RED BRICK® WAREHOUSE
Version 5.1

RISQL® ENTRY TOOL AND
RISQL REPORTER USER’S GUIDE
Contents

About This Document
   Purpose vii
   Audience vii
   Organization viii
   Related Documentation ix
   Conventions xi
      Syntax Notation xi
      Syntax Diagrams xii
      Keywords and Punctuation xiv
      Identifiers and Names xiv
      Filenames xiv
   Customer Support xv
      Support Solutions Warehouse xv
      General and Technical Questions xv
      Troubleshooting Tips xvi
      Documentation Questions and Comments xvii

1 Introduction to the RISQL Entry Tool and RISQL Reporter
   The RISQL Entry Tool 1-2
   The RISQL Reporter 1-2
   The Red Brick Warehouse Environment 1-3

2 Using the RISQL Entry Tool and RISQL Reporter
   Accessing Warehouse Databases 2-2
   Defining a Locale 2-3
   Starting the Tool 2-3
   Connecting to a Warehouse Database 2-7
      Connecting with the RB_PATH Environment Variable 2-8
      Connecting on the Command Line 2-8
      Connecting with the SQL CONNECT Command 2-10
   Ending Your Tool Session 2-12
   Submitting Queries to the Warehouse 2-12
   Interrupting Command Execution 2-14
   Displaying Statistics 2-14
Contents

Using File Redirection     2-15
   Using System Redirection  2-16
   Using Redirection Within the Tool  2-16
Using an Editor          2-22
   Changing the Default Editor    2-22
   Starting an Editor             2-22
Adding Comments to Statements    2-27
Accessing the System Shell   2-29
Changing the Tool Prompt     2-30
Changing Database Passwords   2-31
Administering the Database  2-31
   Defining Database Objects    2-32
   Controlling Database and Table Access   2-32
   Changing Information in a Database  2-33
Creating Data Extraction Applications  2-34
   Redirecting Extraction Results from the Shell  2-34
   Redirecting Extraction Results from Within the Tool  2-35
Exit Status Codes        2-35

3 Formatting Reports
Laying Out Reports     3-2
   Using the SET COLUMN Commands  3-2
      Column Specification    3-3
      Column Variable Precedence    3-3
      Column Display Options   3-4
Saving Command Settings  3-5
Creating a Report Title    3-6
Defining the Page Length    3-7
Skipping a Line or Starting a New Page When Data Changes  3-9
Changing a Column Title   3-10
Underlining Column Titles   3-11
Justifying Data in Columns   3-13
Changing the Width of a Column Field   3-14
Widening Space Between Columns   3-15
Hiding a Column        3-16
Left-Filling Data        3-17
Changing the Format of Numeric Data    3-18
Displaying Signs for Positive Numbers    3-19
Changing the Position of Signs    3-20
Suppressing Duplicate Data in Columns   3-21
Breaking Row Data Across Display Lines   3-22
Defining a Data Label        3-23
Displaying Nulls           3-25
Displaying Current Settings   3-26
Resetting Display Options to Default Values  3-27
Sample Report 1  3-28
Sample Report 2  3-33

4 Command Reference
!(shell escape)  4-4
CONNECT  4-5
EDIT  4-7
QUERY  4-9
QUERY COLUMN  4-11
QUIT or EXIT  4-12
RESET  4-13
RESET COLUMN  4-15
RUN  4-18
SAVE  4-20
SET COLUMN  4-21
SET ECHO  4-51
SET EDITOR  4-53
SET ERROR OUTPUT  4-54
SET OUTPUT  4-55
SET PAGE LENGTH  4-57
SET PROMPT  4-60
SET ROW DATA  4-62
SET TITLE  4-64

A risql and risqlrpt Command Reference
B .rbretrc Files
   Tool Initialization  B-1
   Sample Initialization File  B-2

C Locales
   Overriding the Warehouse Locale  C-1
   Defined Locale Components  C-3

Index
About This Document

Purpose

This user’s guide describes how to use the RISQL® Entry Tool and the RISQL Reporter.

- The RISQL Entry Tool lets you access Red Brick® Warehouse databases from computers running UNIX-based, Windows NT-based, and Windows 95-based operating systems.
- The RISQL Reporter provides all of the features of the RISQL Entry Tool and lets you format reports with data retrieved from warehouse databases.

Most of the features described in this guide apply to both the RISQL Entry Tool and the RISQL Reporter. Those features that are specific to the RISQL Reporter are clearly noted. The phrase “tool command” refers to a command supported by both products.

Audience

The intended users of this guide are warehouse database administrators and SQL application developers. Knowledge of SQL, RISQL extensions, and your host operating system is assumed.
About This Document

Organization

This guide is organized as follows:

Chapter 1, “Introduction to the RISQL Entry Tool and RISQL Reporter,” provides overviews of Red Brick Warehouse, the RISQL Entry Tool, and the RISQL Reporter.

Chapter 2, “Using the RISQL Entry Tool and RISQL Reporter,” describes how to start the RISQL Entry Tool or RISQL Reporter, connect to a warehouse database, submit SQL statements, use file redirection, and use various other RISQL Entry Tool commands and features.

Chapter 3, “Formatting Reports,” describes how to modify the display of result sets with the RISQL Reporter.

Chapter 4, “Command Reference,” describes RISQL Entry Tool and RISQL Reporter commands in alphabetical order.


Appendix B, “.rbretrc Files,” describes the RISQL Entry Tool and RISQL Reporter initialization file.

Appendix C, “Locales,” explains how to override the warehouse locale for Red Brick client tools and identifies the languages, territories, character sets, and sort components supported by Red Brick products.
**Related Documentation**

The standard documentation set for Red Brick Warehouse includes the following documents:

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation and Configuration Guide</td>
<td>Installation and configuration information, as well as platform-specific material, about Red Brick Warehouse and related products. Customized for either UNIX-based or Windows NT systems.</td>
</tr>
<tr>
<td>Warehouse Administrator’s Guide</td>
<td>Description of warehouse architecture, supported schemas, and other concepts relevant to warehouse databases. Procedural information for designing and implementing a warehouse database, maintaining a database, and tuning a database for performance. Includes a description of the system tables and the configuration file (<code>rbw.config</code>). Customized for either UNIX-based or Windows NT systems.</td>
</tr>
<tr>
<td>Table Management Utility Reference Guide</td>
<td>Description of the Table Management Utility, including all activities related to loading and maintaining data. Also includes information about data replication and the <code>rb_cm</code> copy management utility.</td>
</tr>
<tr>
<td>SQL Self-Study Guide</td>
<td>Example-based review of SQL and introduction to the RISQL extensions, the macro facility, and Aroma, the sample database.</td>
</tr>
<tr>
<td>RISQL Entry Tool and RISQL Reporter User’s Guide</td>
<td>Complete guide to the RISQL Entry Tool, a command-line tool used to enter SQL statements, and the RISQL Reporter, an enhanced version of the RISQL Entry Tool with report-formatting capabilities.</td>
</tr>
<tr>
<td>Messages and Codes Reference Guide</td>
<td>Complete listing of all informational, warning, and error messages generated by warehouse products, including probable causes and recommended responses. Also includes event log messages that are written to the log files.</td>
</tr>
<tr>
<td>Release Notes</td>
<td>Information pertinent to the current release that was unavailable when the documents were printed.</td>
</tr>
</tbody>
</table>
In addition to the standard documentation set, the following documents are included for specific sites:

- **Red Brick Vista User’s Guide**: Description of the Red Brick Vista™ aggregate navigation and advice system, including procedures for rewriting queries and getting advice on the best set of aggregate tables and views to create. Includes detailed examples of queries whose performance can be dramatically increased by using aggregate navigation.

- **SQL-BackTrack for Red Brick Warehouse User’s Guide**: Complete guide to SQL-BackTrack™ for Red Brick Warehouse, a command-line interface for backing up and recovering warehouse databases. Includes procedures for defining backup configuration files, performing online and checkpoint backups, and recovering the database to a consistent state.

- **Client Connector Pack Installation Guide**: Procedures for installing and configuring the Red Brick ODBC Driver, the RISQL Entry Tool, and the RISQL Reporter on client systems. Included for those sites that purchase the Client Connector Pack.

- **ODBC Connectivity Guide**: Information about ODBC conformance levels as well as instructions for compiling and linking an ODBC application using the Red Brick ODBClib SDK.

- **Red Brick Data Mine User’s Guide**: Description of the data mining process, and procedural information for using the Red Brick Data Mine™ SQL-based interface to find hidden or unpredictable relationships among the data in a data set. Included for those sites that purchase the Red Brick Data Mine option.


Additional references you might find helpful include:

- An introductory-level book on SQL
- An introductory-level book on relational databases
- Documentation for your hardware platform and operating system

**Online Documentation**

The English version of the Red Brick Warehouse documentation is also available in Adobe Acrobat format (PDF) on a separate CD-ROM.
Conventions

Throughout Red Brick Systems technical publications, the following notation and syntax conventions are used:

- Computer input and output, including commands, code, and examples, appear in Courier.
- Information that you enter or that is being emphasized in an example appears in Courier bold to help you distinguish it from other text.
- Filenames, system-level commands, and variables appear in Palatino italic or Courier italic, depending on the context.
- Document titles always appear in Palatino italic.
- Names of database tables and columns are capitalized (Sales table, Dollars column). Names of system tables and columns are in all uppercase (RBW_INDEXES table, TNAME column).

Syntax Notation

This guide uses the following conventions to describe the syntax of operating-system commands:

<table>
<thead>
<tr>
<th>Command Element</th>
<th>Example</th>
<th>Convention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values and parameters</td>
<td>table_name</td>
<td>Items that you replace with an appropriate name, value, or expression are in italic type style.</td>
</tr>
<tr>
<td>Optional items</td>
<td>[ ]</td>
<td>Optional items are enclosed by square brackets. Do not type the brackets.</td>
</tr>
<tr>
<td>Choices</td>
<td>ONE</td>
<td>TWO</td>
</tr>
<tr>
<td>Required choices</td>
<td>{ONE</td>
<td>TWO}</td>
</tr>
<tr>
<td>Default values</td>
<td>ONE</td>
<td>TWO</td>
</tr>
<tr>
<td>Repeating items</td>
<td>name, ...</td>
<td>Items that can be repeated are followed by a comma and an ellipsis. Separate the items with commas.</td>
</tr>
<tr>
<td>Language elements</td>
<td>( ) , ; .</td>
<td>Parentheses, commas, semicolons, and periods are language elements. Use them exactly as shown.</td>
</tr>
</tbody>
</table>
Syntax Diagrams

This guide uses diagrams built with the following components to describe the syntax for statements and all commands other than system-level commands:

<table>
<thead>
<tr>
<th>Component</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Statement begins.</td>
</tr>
<tr>
<td></td>
<td>Statement syntax continues on next line. Syntax elements other than complete statements end with this symbol.</td>
</tr>
<tr>
<td></td>
<td>Statement continues from previous line. Syntax elements other than complete statements begin with this symbol.</td>
</tr>
<tr>
<td></td>
<td>Statement ends.</td>
</tr>
<tr>
<td>SELECT</td>
<td>Required item in statement.</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>Optional item.</td>
</tr>
<tr>
<td>DBA TO</td>
<td>Required item with choice. One and only one item must be present.</td>
</tr>
<tr>
<td>CONNECT TO</td>
<td>Optional item with choice. If a default value exists, it is printed in bold.</td>
</tr>
<tr>
<td>ASC</td>
<td>Optional items. Several items are allowed; a comma must precede each repetition.</td>
</tr>
<tr>
<td>DESC</td>
<td></td>
</tr>
</tbody>
</table>
The syntax elements shown above are combined to form a diagram as follows:

![Diagram of syntax elements](image)

Complex syntax diagrams such as the one for the following statement are repeated as point-of-reference aids for the detailed diagrams of their components. Point-of-reference diagrams are indicated by their shadowed corners, gray lines, and reduced size:

![Point-of-reference diagram](image)

The point-of-reference diagram is then followed by an expanded diagram of the shaded portion—in this case, the *select_list*:

![Expanded diagram of select_list](image)
Keywords and Punctuation

Keywords are words reserved for statements and all commands except system-level commands. When a keyword appears in a syntax diagram, it is shown in uppercase. You can write a keyword in upper- or lowercase, but you must spell the keyword exactly as it appears in the syntax diagram.

Any punctuation that occurs in a syntax diagram must also be included in your statements and commands exactly as shown in the diagram.

Identifiers and Names

Metavariables serve as placeholders for identifiers and names in the syntax diagrams and examples. A metavariable can be replaced by an arbitrary name, identifier, or literal, depending on the context. Metavariables are also used to represent complex syntax elements that are expanded in additional syntax diagrams. When a metavariable appears in a syntax diagram, an example, or text, it is shown in *lowercase italic*.

The following syntax diagram uses metavariables to illustrate the general form of a simple SELECT statement:

```
|
```

```
| SELECT — column_name — FROM — table_name |
```

When you write a SELECT statement of this form, you replace the metavariables `column_name` and `table_name` with the name of a specific column and table.

Filenames

Throughout this document, file and directory names in examples are shown in UNIX style, using forward slashes (/) as separators (for example, `/spock/risql/query1`). Windows NT and Windows 95 users can choose to use this format, or the more common Windows-style backslash (\) separator (for example, `c:\spock\risql\query1`); both formats work equally well.
Customer Support

Please review the following information before contacting the Customer Support Center at Red Brick Systems.

Support Solutions Warehouse

The Support Solutions Warehouse is the Customer Support Center’s external web site, an online resource that registered Red Brick customers can use to:

- Submit new cases.
- Read release notes.
- Find answers to frequently asked questions (FAQs).
- Search the Problems and Solutions database.

To use the Support Solutions Warehouse, point your web browser to the following URL and enter your registered username and password:

http://www.redbrick.com/RBCustomer/index.htm

If you do not have a registered username and password, contact the Customer Support Center by telephone, fax, or e-mail.

General and Technical Questions

If you have general sales-related questions or technical questions about Red Brick products or services, contact Red Brick Systems as follows:

Telephone
General Questions (408) 399-3200 or 1 (800) 777-2585
Technical Questions (408) 399-7100 or 1 (800) 727-1866

FAX
General Questions (408) 399-3277
Technical Questions (408) 399-3297

Internet e-mail
General Questions info@redbrick.com
Technical Questions support@redbrick.com

World Wide Web www.redbrick.com
Existing Cases

If you want to inquire about the status of an existing case, please have the case number ready. The case number will always be given to you by the support engineer who logs the case or first contacts you. This number is used to keep track of all the activities performed during the resolution of each problem.

New Cases

If you want to log a new case, please have the following information ready:

- Red Brick Warehouse version
- Platform and operating-system version
- Error messages returned by Red Brick Warehouse or the operating system
- Concise description of the problem, including any commands or operations performed prior to the occurrence of the error message
- List of Red Brick Warehouse and/or operating-system configuration changes made prior to the occurrence of the error message

If you think the problem concerns “client/server” connectivity, please have the following additional information ready:

- Name and version of the client tool in use
- Version of Red Brick ODBC Driver in use (if applicable)
- Name and version of client network and/or TCP/IP stack in use
- Error messages returned by the client application
- Warehouse and client locale specifications
**Troubleshooting Tips**

You can often reduce the time it takes to close your case by providing the smallest possible reproducible example of your problem. The more you can isolate the cause of the problem, the more quickly the support engineer can help you resolve it.

- For SQL query problems, try removing columns or functions, or restating WHERE, ORDER BY, or GROUP BY clauses until you can isolate the part of the statement causing the problem.
- For TMU load problems, verify the datatype mapping between the source file and the target table to ensure compatibility. Try loading a small test set of data to determine whether the problem concerns volume or data format.
- For connectivity problems, verify that the network is up and running by issuing the `ping` command from the client to the host. If possible, try another client tool to see if the same problem arises.

**Documentation Questions and Comments**

If you have questions or comments about the Red Brick Warehouse documentation, please contact the Technical Publications Department at Red Brick Systems as follows:

**Telephone**
- + 1 408 399 3200
- + 1 800 727 1866 (USA only)

**Internet e-mail**
docs@redbrick.com
Red Brick® Warehouse is a relational database management system (RDBMS) designed for data warehouse, data mart, and online analytical processing (OLAP) applications. Compared to online transaction processing (OLTP) or “universal” database products, Red Brick Warehouse delivers higher query-processing and data-loading performance, greater ease of administration, and a richer set of specialized features for applications that range from a few gigabytes to well over a terabyte, and from a few users to thousands of users.

Red Brick Warehouse can scale from the workgroup to the enterprise, is built for an open client/server environment using industry-standard open database connectivity (ODBC), and is accessed using industry-standard SQL. The server’s RISQL® extensions simplify analyses that require ranks, ratios, and other commonly used business calculations, while the Red Brick Vista™ STARjoin™, STARindex™, TARGETjoin™, and TARGETindex™ technologies provide unparalleled ad hoc query and analysis performance against very large databases with various schema designs. Managers and analysts can pose numerous and creative queries to quickly receive the information they need, and make good business decisions with similar speed and confidence.
Introduction to the RISQL Entry Tool and RISQL Reporter

The RISQL Entry Tool

The RISQL Entry Tool provides interactive access to Red Brick Warehouse databases. You can use the RISQL Entry Tool for writing and executing queries and for database administration tasks.

The RISQL Entry Tool accepts three forms of input:

- Standard SQL statements including those that contain RISQL extensions
- RISQL Entry Tool commands
- Operating-system commands through shell-escape commands

The RISQL Reporter

The RISQL Reporter provides all of the functionality of the RISQL Entry Tool plus the ability to format query output into reports. You can use RISQL Reporter features to define titles for your reports that include a date, time, and page numbers, add spaces or start new pages based on changes in data values, and set format characteristics such as data justification and column width on individual columns.

Most of the features described in this guide apply to both the RISQL Entry Tool and the RISQL Reporter. Those features that are specific to the RISQL Reporter are clearly noted.

Note: The RISQL Reporter product must be licensed before it can be used. For more information about licensing optional Red Brick products, refer to the Installation and Configuration Guide for your platform.
The Red Brick Warehouse Environment

The RISQL Entry Tool and RISQL Reporter can be used from a UNIX-based client, Windows 95 client, or Windows NT client. The RISQL Entry Tool, RISQL Reporter, and other client query tools communicate with Red Brick Warehouse databases through ODBC connectivity software.

For more information about the Red Brick Warehouse environment and query logging, refer to the Warehouse Administrator’s Guide.
Using the RISQL Entry Tool and RISQL Reporter

The RISQL Entry Tool and RISQL Reporter are interactive programs that allow you to access warehouse databases. You can use these tools to perform database and server administration tasks and to write queries. The RISQL Reporter also allows you to format query results into attractive reports. All features described in this chapter apply to both the RISQL Entry Tool and the RISQL Reporter, unless Reporter-only support is specifically noted.

This chapter contains the following sections:

• Accessing Warehouse Databases
• Defining a Locale
• Starting the Tool
• Connecting to a Warehouse Database
• Ending Your Tool Session
• Submitting Queries to the Warehouse
• Interrupting Command Execution
• Displaying Statistics
• Using File Redirection
• Using an Editor
• Adding Comments to Statements
• Accessing the System Shell
• Changing the Tool Prompt
• Changing Database Passwords
• Administering the Database
• Creating Data Extraction Applications
Identifying System Shell Prompts

The system default prompt is shell-specific:

- C shell: `%`
- Korn and Bourne shells: `$`
- Windows NT shell and Windows 95 shell: `C:\>`

The Korn-shell prompt (`$`) is used in examples throughout this document.

Accessing Warehouse Databases

Only the system administrator can provide database access to users. Database access authorizations are granted with the SQL GRANT CONNECT command, which is described in detail in both the Warehouse Administrator’s Guide and the SQL Reference Guide.

The RB_CONFIG environment variable must be defined before you attempt to run the RISQL Entry Tool or RISQL Reporter. This environment variable can be set in your shell-startup file or on the operating-system command line. The following error messages are displayed when the RB_CONFIG environment variable is either not defined or defined incorrectly when you start the RISQL Entry Tool:

- Undefined RB_CONFIG environment variable:
  ** FATAL ** (803) RB_CONFIG must be defined.
- Incorrectly defined RB_CONFIG environment variable:
  Error in opening config file; unable to continue.
  ** FATAL ** (10502) Message base lookup failed for message 10502, error code 200040.

When you start the RISQL Reporter, the tool checks the rbw.config file for license verification. If a valid license key is not found, the following message is displayed:

** FATAL ** (802) Not licensed for 'RISQL_REPORTER'.

For more information about Red Brick Warehouse error messages, refer to the Messages and Codes Reference Guide. For more information about the RB_CONFIG environment variable, refer to page A-3 and the Warehouse Administrator’s Guide.
Defining a Locale

The warehouse locale, defined when the Red Brick Warehouse software is installed, specifies the language and location for all the databases in that installation. RISQL Entry Tool and RISQL Reporter users can specify which language they want to use by using the RB_NLS_LOCALE environment variable to override the warehouse locale. For information about setting this environment variable, refer to Appendix C, “Locales.”

For detailed information about locale specifications and incompatibilities that might arise when the client locale differs from the database locale, refer to the Warehouse Administrator’s Guide.

Starting the Tool

Configuration information specific to the RISQL Entry Tool or RISQL Reporter is required before you can start either tool. Database administrators typically provide this information so that you have easy access to one or more warehouse databases. If the information has not been provided, consult your database administrator.

Syntax

The tool startup syntax is as follows:

```
risql [-q] [-s dsn] [-h host] [-d database] [db_username [db_password]]
```

where:

- `-q` Quiet mode in which the copyright notice is not displayed when the tool starts.
- `-s dsn` Data source name (DSN) defined in the _odbc.ini_ file on UNIX systems, _odbc.ini_ file on Windows 95, and the Registry on Windows NT. A DSN can also be specified with the RB_DSN environment variable.
- `-h host` RB_HOST value defined in the _rbw.config_ file. This RB_HOST value can also be specified with the RB_HOST environment variable.
Using the RISQL Entry Tool and RISQL Reporter
Starting the Tool

-d database Logical warehouse database name defined in the server's rbw.config file. A logical database name can also be specified with the RB_PATH environment variable or with an SQL CONNECT command, as described on page 2-7.

db_username Valid username for the specified database. If db_username is not specified, the tool prompts for the database username.

db_password Valid password for the specified database. If db_password is not specified, the tool prompts for the database password.

To start a RISQL Reporter session, substitute risqlrpt for risql above.

For more detailed information about risql and risqlrpt commands, command-line options, and environment variables, refer to Appendix A, “risql and risqlrpt Command Reference.”

Example

To start a RISQL Entry Tool session and connect to a database:

1. Enter the following command at the system prompt:

   $ risql

   Note: If the RISQL Entry Tool program (risql) or the RISQL Reporter program (risqlrpt) is not in your search path, the operating system displays a message, similar to the following, when you try to start the tool:

   risql: command not found
   risqlrpt: command not found

   If this message appears, either type the full pathname for the risql or risqlrpt program on the command line when you start the tool or add the pathname to your search path. For further assistance, consult your operating-system administrator.

The RISQL Entry Tool displays a copyright notice and prompts for your database username.

(C) Copyright 1991-1998 Red Brick Systems, Inc. Los Gatos, California, USA
All rights reserved.
RISQL Entry Tool Version 5.1.x(xxxxx)
Please type username:
If you are starting the RISQL Reporter with the `risqlrpt` command, the following copyright notice is displayed:

(C) Copyright 1991-1998 Red Brick Systems, Inc. Los Gatos, California, USA
All rights reserved.
RISQL Reporter Version 5.1.x(xxxxx)
Please type username:

2. Type your database username and press Return.

Please type username: **curly**
The RISQL Entry Tool prompts for your database user password.

3. Type your database password and press Return.

Please type password: **xxxxxxxx**

When your password is accepted, you are connected to the database and the tool displays the following prompt:

RISQL>

**Usage Notes**

When you omit `-d database` on the command line as specified here, the tool searches for a logical database name defined by the environment variable RB_PATH and connects to that database. If RB_PATH has not been defined, the tool starts but is not connected to a database. In this case, you can use the CONNECT command to access a warehouse database. Both the RB_PATH environment variable and the CONNECT command are described in “Connecting to a Warehouse Database” on page 2-7.

The optional `-q` parameter causes the RISQL Entry Tool or RISQL Reporter to run in quiet mode. In quiet mode, the copyright notice and the column headings are neither displayed nor printed in a file when output is redirected. This option is useful when you are using the RISQL Entry Tool or RISQL Reporter in data extraction scripts intended for TMU load statements. For more information about operating in quiet mode, refer to “Creating Data Extraction Applications” on page 2-34.

**Note:** You might need to change your password when you log into a database for the first time. For more information, refer to “Changing Database Passwords” on page 2-31.
Valid Input

When you see the RISQL prompt, you can enter:

- RISQL Entry Tool commands
- Operating-system commands through shell-escape commands

After you make a database connection, you can also enter SQL statements, including those that contain RISQL extensions, to query a warehouse database.

Command Syntax

All SQL statements and RISQL Entry Tool or RISQL Reporter commands except the shell-escape command (!) must be followed by a semicolon (;). The examples in this guide show a space character between the last element of a statement or command declaration and the semicolon. This space is allowed by SQL, but not required.

Semicolons do not terminate statements when found inside:

- Quoted strings
- C-program-style comments (/ *...*/)
- SQL-style comments (--...)

A semicolon occurring anywhere else terminates the command or statement. If a semicolon as the last character of a line fails to terminate the statement or command, the statement contains either a quoted string that is missing the closing quote or a C-style comment that is missing the end-comment characters (*/).

For more information about using comments in your statements, refer to “Adding Comments to Statements” on page 2-27.

New-Line Character

Sometimes, quoted literals span multiple lines. To avoid displaying the INCOMPLETE STRING prompt in these cases, you can insert a line break in a quoted string of characters with a C-program-style new-line character (\n). For more information about the INCOMPLETE STRING prompt, refer to page 2-25.
.rbretc file

RISQL Entry Tool and RISQL Reporter commands can be entered interactively at the tool prompt or in a tool initialization file named .rbretc. Setting command preferences interactively changes the default settings for the immediate session only. Changing command preferences in the .rbretc file changes the default settings each time a RISQL Entry Tool or RISQL Reporter session is started or until you modify the .rbretc file. Settings made in the .rbretc file during a tool session (for example, during a shell-escape sequence) are not applied until the tool is restarted. For more information about the initialization file, refer to Appendix B, “.rbretc Files.”

Error Messages

Error messages generated by the RISQL Entry Tool, RISQL Reporter, and other components of Red Brick Warehouse are documented in the Messages and Codes Reference Guide.

Connecting to a Warehouse Database

To make a warehouse database available to the RISQL Entry Tool or RISQL Reporter, the database administrator must assign a logical database name that describes the full pathname to the database in the warehouse configuration file (rbw.config). Users can then specify this logical name when requesting a database connection.

After logical database names are defined in the server’s rbw.config file, there are four ways to connect to a database:

• Users or system administrators can define an environment variable named RB_PATH, which is set to a logical database name. With this method, the RISQL Entry Tool or RISQL Reporter defaults to the database defined in RB_PATH when users start the tool without specifying a database name on the command line.
• Users can enter the -d option to specify a logical database name on the command line when they start the RISQL Entry Tool or RISQL Reporter.
• Users can enter the -s option to specify a data source name (DSN) or the RB_DSN environment variable. To access warehouse databases on remote servers through the Data Sources dialog box, you must install the Red Brick Client Connector Pack on your machine.
• Users can issue a CONNECT command from inside a RISQL Entry Tool or RISQL Reporter session and specify a logical database name or a data source name (DSN).
Connecting with the RB_PATH Environment Variable

If you do not specify a database on the command line when you start the RISQL Entry Tool or RISQL Reporter, the tool attempts to connect to a database specified by the environment variable RB_PATH. Defining a database with RB_PATH lets you set a default database, which is useful when you access the same warehouse database most of the time.

To set the RB_PATH environment variable:

1. Ask your database administrator for the logical database name of the database you want to access.
2. For Korn shell and Bourne shell, enter the following command at the system prompt:
   ```bash
   $ RB_PATH=db_name; export RB_PATH
   ```
   For C shell, enter:
   ```csh
   % setenv RB_PATH db_name
   ```
   For Windows NT and Windows 95 shell, enter:
   ```cmd
   C:\> set RB_PATH=db_name
   ```
   where `db_name` is a logical name for the database you want to access. For more information about defining logical database names, refer to the Warehouse Administrator's Guide.

Environment variables are specific to your login session. To avoid having to reset RB_PATH each time you log in, you might prefer to set it in your shell startup file.

Connecting on the Command Line

The `-d database` option, when used at startup, connects you to the database specified by the argument `database_name`, which can be any logical database name. You can specify your database username and password after `database_name` on the command line, or you can use the `-d` option alone and then respond to the tool prompts for your username and password.

The `-s dsn` option can be used to connect and log in to the database specified in the Data Source dialog box. A DSN is the logical name for the combination of a specific warehouse server installed on a host system, the name of a database on that host, and a default username. The Data Source dialog box contains fields for entering this information. After definition, the DSN is displayed in the
Using the RISQL Entry Tool and RISQL Reporter
Connecting to a Warehouse Database

connectivity component of an ODBC-compliant tool, making it easy for the user to access warehouse databases that are not located on the local server. For more information about DSN, refer to the Client Connector Pack Installation Guide.

Note: The RISQL Entry Tool and RISQL Reporter prompt you for username and password information only if standard input is coming from the terminal and if standard output and error output have not been redirected. For more information about redirecting input and output, refer to “Using File Redirection” on page 2-15.

For more information about starting the risql program, refer to Appendix A, “risql and risqlrpt Command Reference.”

Examples

To connect to but not log in to the Aroma database (a logical database name) when you start the RISQL Entry Tool, enter:

$ risql -d aroma

where aroma is a logical database name. The system responds:

(C) Copyright 1991-1998 Red Brick Systems, Inc. Los Gatos, California, USA
All rights reserved.
RISQL Entry Tool Version 5.1.x(xxx)
Please type username: curly
Please type password: xxxxxxxx
RISQL>

Note: Passwords are limited to 8 characters when prompted for by the RISQL Entry Tool or RISQL Reporter. Passwords entered on the command line, as shown in the following example, are limited to 128 characters.

To connect to and log into the Aroma database when you start the RISQL Entry Tool, enter:

$ risql -d aroma curly mysecret

(C) Copyright 1991-1998 Red Brick Systems, Inc. Los Gatos, California, USA
All rights reserved.
RISQL Entry Tool Version 5.1.x(xxx)
RISQL>

Caution: With this method, on UNIX platforms, your password is displayed on your screen and can be displayed on the screens of other users if they list processes with the operating system ps command.
To connect and log into the Aroma database by using a DSN when you start the RISQL Entry Tool, enter:

```
$ risql -s aroma
```

where `aroma` is a logical DSN defined in your environment.

### Connecting with the SQL CONNECT Command

When you start the RISQL Entry Tool or RISQL Reporter without connecting to a database, you can use the SQL CONNECT command to specify a logical database name. When you omit your database username on the command line, the tool prompts you for both username and password.

If you issue the CONNECT command while a session with another database is still active, the current session is closed before the new session is opened. If your second connection attempt fails because you entered an incorrect password, the tool displays the RISQL prompt but you are no longer connected to the first database. In this situation, you must issue a valid CONNECT statement before you can access a database.

If you have installed the Red Brick Client Connector Pack, you can use the CONNECT command to connect to a DSN after invoking the RISQL Entry Tool or the RISQL Reporter.

For complete syntax information for the CONNECT command, refer to page 4-5.

### Examples

To connect to a database during a RISQL Entry Tool or RISQL Reporter session, enter the CONNECT command followed by the database name and a valid database username. For example:

```
RISQL> connect to aroma as moe ;
Please type password: xxxxxxxx
RISQL>
```

In this example, the logical database name `aroma` has been defined by the database administrator.
Suppose your company has a group database account and password for a particular database. If you start the RISQL Entry Tool and connect with your personal account, you can connect again to the same database under another username. For example:

```
RISQL> connect as 3stooges ;
Please type password: xxxxxxxx
RISQL>
```

If you have installed the Red Brick Client Connector Pack on a Windows machine and want to use the DSN, enter:

```
RISQL> connect to aroma ;
```

where _aroma_ is a DSN defined in your environment.

If you want to use the RB_DSN environment variable on a UNIX platform, enter at the prompt:

```
% setenv RB_DSN dsn_name
% risql
RISQL>
```

where _dsn_name_ is a DSN defined in your environment.

If you want to use the RB_DSN environment variable on a Windows NT platform, enter at the prompt:

```
C: \> set RB_DSN=dsn_name
C: \> risql
```

where _dsn_name_ is a DSN defined in your environment.
Ending Your Tool Session

The EXIT and QUIT commands terminate a RISQL Entry Tool or RISQL Reporter session and return you to the system prompt. If you have a current database connection when you issue one of these commands, the connection is closed before the tool terminates.

Example

To terminate the session, enter:

```
RISQL> quit ;
$
```

or:

```
RISQL> exit ;
$
```

Submitting Queries to the Warehouse

To retrieve data from a database with the RISQL Entry Tool or RISQL Reporter:

1. Start the tool and connect to a database.
2. Type an SQL query.
3. Type a semicolon(;) at the end of the statement.
4. Press Return to submit the query to the database.

For a complete description of warehouse-supported SQL and RISQL extensions, refer to the SQL Reference Guide. For more query examples, refer to the SQL Self-Study Guide.

Note: SQL keywords including those specific to the RISQL Entry Tool and RISQL Reporter (such as SET, RUN, and EDIT) are always in English, regardless of the current locale.
Examples

A simple SELECT statement lists all the cities and districts for which data is stored in the Aroma sample database. This sample database is usually created during the installation process.

```
RISQL> select * from market;
MKTKEY    HQ_CITY       HQ_STATE        DISTRICT       REGION
 1  Atlanta   GA        Atlanta        South
 2   Miami    FL        Atlanta        South
 3  New Orleans LA        New Orleans    South
 4    Houston TX        New Orleans    South
 5   New York NY        New York       North
 6 Philadelphia PA        New York       North
 7    Boston  MA        Boston         North
 8   Hartford CT        Boston         North
 9   Chicago   IL        Chicago        Central
10   Detroit   MI        Chicago        Central
11 Minneapolis MN        Minneapolis    Central
12    Milwaukee WI        Minneapolis    Central
13    San Jose CA        San Francisco   West
14  San Francisco CA        San Francisco   West
15     Oakland CA        San Francisco   West
16  Los Angeles CA        Los Angeles     West
17  Phoenix  AZ        Los Angeles     West
```

SELECT * returns all information from a table. Be careful when using it to view large tables, because doing so might inadvertently reduce server warehouse performance for other users and make your terminal inaccessible until the query result is returned. The SQL SET ROWCOUNT command is useful for limiting the number of rows returned by a query. For more information about this command, refer to the SQL Reference Guide.

To display the annual sales stored in the Aroma database (including the number of items sold, the average sales, and total sales) for Lotta Latte brand coffee in Los Angeles during 1995, enter the following query:

```
RISQL> select count(*) as num_items, 
> string (avg(dollars), 7,2) as avg_sale, 
> sum (dollars) as total_sales 
> from store st 
> join sales sa on st. storekey = sa. storekey 
> join product pr on pr. prodkey = sa. prodkey 
> and pr.classkey = sa.classkey 
> join period pe on pe.perkey = sa. perkey 
> where prod_name = 'Lotta Latte' 
> and year = 1995 
> and city = 'Los Angeles';
NUM_ITEMS   AVG_SAL TOTAL_SALES
 80 171.33        13706.50
```
Interrupting Command Execution

To interrupt an SQL or RISQL statement before it returns data, enter Control-C. (that is, hold down the key marked “control” and press the key marked “C” at the same time.) This keystroke sequence interrupts the current command execution and returns you to the RISQL prompt.

Displaying Statistics

The SET STATS command allows you to display statistics after each statement is executed. You can choose to display:

• No statistics, with SET STATS OFF (this is the default).
• Basic reporting, with SET STATS ON, which typically includes the following types of statistical information:
  – Number of data rows returned
  – Elapsed time during statement processing
  – CPU time spent on statement execution
  – Logical I/O counts
• More detailed reporting, with SET STATS INFO, which includes additional information about how queries are processed.

Statistics reporting varies from statement to statement and from platform to platform; for example, when you display statistics for a CREATE TABLE statement, no data rows are returned so there are no values to report.

The RISQL Entry Tool and RISQL Reporter send statistical output to the same destination as error output, which is the terminal by default. You can redirect statistics output with the SET ERROR OUTPUT command. For more information about redirecting statistical output, refer to “Redirecting or Discarding Error Messages” on page 2-20.

For information about the SET STATS command, refer to the SQL Reference Guide.
Using File Redirection

File redirection is useful when you want to execute long SQL statements or generate reports with the RISQL Reporter. There are two methods for using file redirection:

- You can use standard system file-redirection symbols (>, <, >>, <<, |) to redirect input that comes from the terminal and output that goes to the screen.
- You can use tool commands (RUN, SET OUTPUT, SET ERROR OUTPUT) to read from and write to files.
Using System Redirection

The RISQL Entry Tool and RISQL Reporter use standard syntax for redirecting standard system input and output. To direct file input to the RISQL Entry Tool from the system command line, enter:

```
$ risql -d database db_username db_password < input_file
```
or:

```
$ cat input_file | risql -d database db_username db_password
```

where `input_file` is a file containing SQL or tool commands. (The `cat` command may not be available on your Windows NT system.)

To read from and write to a file, enter:

```
$ risql -d database db_username db_password < input_file > output_file
```

where `output_file` contains the output.

**Note:** To avoid typing the database name, username, and password on each command line, you can include a CONNECT statement on the first line of the input file. If you use this method, you must enter your database username and password on one line. For more information about the CONNECT command, refer to “Connecting to a Warehouse Database” on page 2-7.

When redirecting output with system-redirection commands, you might want to use the `-q` option to run in quiet mode. In quiet mode, the copyright notice and the column headings are suppressed and only data output is redirected to the output file. For more information about quiet mode, refer to “Creating Data Extraction Applications” on page 2-34.

Using Redirection Within the Tool

The following tool commands control input and output:

- **RUN**
- **SET OUTPUT**
- **SET ERROR OUTPUT**

Controlling input and output during a tool session is described in the following sections. The SET ECHO command is also described as it relates to input and output.
Reading Input

You can read input from the tool’s command buffer or from a file. The RUN command, when issued at the RISQL prompt, executes the contents of the command buffer or a specified file. The output is directed to the screen or to another file depending on the output setting. Input stored in the command buffer is read in the following order:

- If the most recent command was EDIT, issuing the RUN command without a filename executes the previously edited file.
- If the most recent command was not EDIT, issuing the RUN command executes the command buffer, which contains the last command that was executed from standard input.

For complete syntax information for the RUN command, refer to page 4-18.

Examples

To execute the contents of the command buffer, which contains the most recently executed command or SQL statement, enter:

```
RISQL> run ;
```

Input stored in files can be saved and executed many times.

To execute the contents of a file named `store_rank` located in the directory from which you started the RISQL Entry Tool or RISQL Reporter, enter:

```
RISQL> run store_rank ;
RISQL> select store_name, sum(dollars) as total_sales, rank(sum(dollars)) as sales_rank
> from market m join store r on m.mktkey = r.mktkey
> join sales s on s.storekey = r.storekey
> join period p on s.perkey = p.perkey
> where year = 1995
> and month = 'MAR'
> and region = 'West'
> group by store_name;
```

<table>
<thead>
<tr>
<th>STORE_NAME</th>
<th>TOTAL_SALES</th>
<th>SALES_RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cupertino Coffee Supply</td>
<td>18709.50</td>
<td>1</td>
</tr>
<tr>
<td>San Jose Roasting Company</td>
<td>17867.90</td>
<td>2</td>
</tr>
<tr>
<td>Java Judy’s</td>
<td>17826.25</td>
<td>3</td>
</tr>
<tr>
<td>Beaches Brew</td>
<td>17197.25</td>
<td>4</td>
</tr>
<tr>
<td>Roasters, Los Gatos</td>
<td>15159.05</td>
<td>5</td>
</tr>
<tr>
<td>Instant Coffee</td>
<td>14841.75</td>
<td>6</td>
</tr>
</tbody>
</table>

If you want to execute a file from another directory, you can specify the full or relative pathname of the file on the RUN command line.
Including Input in Output

You can display input before output. The SET ECHO command controls command echoing when you execute the contents of the command buffer or a file. Command echoing is turned on by default.

For complete syntax information for the SET ECHO command, refer to page 4-51.

Examples

To display SQL statements before the display of output, enter:

```sql
RISQL> set echo on;
```

To display the current echo setting, enter:

```sql
RISQL> query echo;
```

To restore default echoing, enter:

```sql
RISQL> reset echo;
```

If you use the RUN command to execute the contents of the command buffer or a file with SET ECHO ON, input is displayed before the output. To submit a query stored in a file named `lotta_latte_LA_95`, enter:

```sql
RISQL> run lotta_latte_LA_95;

RISQL> select count(*) as num_items,>
     > string (avg(dollars), 7,2) as avg_sale,>
     > sum (dollars) as total_sales>
     > from market ma join store st>
     > on ma.mktkey = st.mktkey>
     > join sales sa on st.storekey = sa.storekey>
     > join product pr on pr.prodkey = sa.prodkey>
     > and pr.classkey = sa.classkey>
     > join period pe on pe.perkey = sa.perkey>
     > where prod_name = 'Lotta Latte'>
     > and year = 1995>
     > and city = 'Los Angeles';
```

```
   NUM_ITEMS   AVG_SALE   TOTAL_SALES
         80     171.33     13706.50
```

To display output only, when input comes from the command buffer or a file, enter:

```sql
RISQL> set echo off;
```
When you start the RISQL Entry Tool or RISQL Reporter in quiet mode, the
SET ECHO command is set to OFF by default.

**Saving or Discarding Output**

The SET OUTPUT command directs input to a specified file or discards it
altogether. The SET OUTPUT command has the following options:

- Sends output to a specified file and overwrites the contents of a file, if the
  file exists when the command is issued or when the tool starts.
- Appends output to the contents of a specified file.
- Sends output to standard output, which, depending on whether output
  redirection is in effect at the operating-system level, is either the terminal
  screen or a file.
- Sends output to a specified program, which modifies output when it is
  displayed on a terminal.
- Discards all output.

This command sends all output to the specified file until the output setting is
changed or until the RISQL Entry Tool or RISQL Reporter session is ended.
Sending output to a file logs all statements executed during the current session.
If the same filename is specified in a subsequent tool session, the contents of
the file are overwritten with output from the current session.

**Examples**

To send output to a file named `test.risql` in the directory `/home/curly`, enter:

```
RISQL> set output /home/curly/test.risql;
```

To send output to a file named `test.risql` and append the output to the contents
of an existing file:

```
RISQL> set output /home/curly/test.risql append;
```

To send output to standard output (the default setting), enter:

```
RISQL> set output stdout;
```
Using the RISQL Entry Tool and RISQL Reporter
Using File Redirection

Specifying an Output Program

Specifying an output program is useful when you want to control the display of output on a terminal. When you specify the PROGRAM option of the SET OUTPUT command, the tool sends output to a system program, which processes and displays it. Typical system output programs include `more` and `cat`.

Note: The `cat` command may not be available on your Windows system.

The system program `more` displays output one screen at a time, scrolling additional rows after the initial screen.

For syntax information for the SET OUTPUT command, refer to page 4-55.

Examples

To specify the `more` program, enter:

```
RISQL> set output program 'more';
```

The name of the program must be enclosed in single quotes.

The `-c` option of the `more` command clears the display area and displays continued output a screen at a time, rather than scrolling it. To specify `more` with the `-c` option, enter:

```
RISQL> set output program 'more' -c;
```

To discard all output, enter:

```
RISQL> set output off;
```

To restore the output setting to the default (standard output), enter:

```
RISQL> reset output;
```

Redirecting or Discarding Error Messages

The SET ERROR OUTPUT command directs error, warning, and informational messages to a specified file or discards error messages. This command has the following options:

- Sends error messages to a file and overwrites the contents of the file if the file exists when the command is issued or when the tool starts.
- Appends error messages to the contents of a file.
• Sends error messages to standard output, which, depending on whether output redirection is in effect at the operating-system level, is either the terminal or a file.

Note: Fatal messages, which indicate that your server connection has been lost, are always displayed in standard output and cannot be redirected with the SET ERROR OUTPUT command.

This command sends all informational messages to the specified file until the setting is changed or the tool session is ended. Sending output to a file logs all error messages for the current session. If the same filename is specified in a subsequent session, the contents of the file are overwritten with error output from the new session.

For complete syntax information for the SET ERROR OUTPUT command, refer to page 4-54.

Examples

To send error messages to a file named errors in the directory /home/curly, enter:

RISQL> set error output /home/curly/errors ;

To send error messages to a file named errors and append the new messages to the list of messages in the file from a previous tool session, enter:

RISQL> set error output /home/curly/errors append ;

Sending error messages to standard error is useful when you have output directed to a file. To send error messages to standard error (the default setting), enter:

RISQL> set error output stderr ;

or:

RISQL> reset error output ;

To send error messages to standard output (the terminal screen or a file, depending on whether output redirection is in effect at the operating-system level), enter:

RISQL> set error output stdout ;
Using an Editor

You can use an editor with the RISQL Entry Tool or RISQL Reporter to modify and correct SQL statements and tool commands located in existing files or in the command buffer. Both tools support any editor installed on your system. The default editor is *vi* for UNIX platforms and Notepad for Windows platforms.

Changing the Default Editor

You can change the default editor to another system editor with the SET EDITOR command.

For complete syntax information for the SET EDITOR command, refer to page 4-53.

**Examples**

To change the editor to *ed*, enter:

```
RISQL> set editor 'ed' ;
RISQL>
```

The editor command declaration must be enclosed in single quotation marks.

To display the current editor, enter:

```
RISQL> query editor ;
```

To restore the default editor, enter:

```
RISQL> reset editor ;
```

Starting an Editor

The EDIT command starts the default editor or the editor you have specified with the SET EDITOR command. You can use the editor to open:

- An existing file that contains SQL statements or tool commands
- A temporary buffer, which can be used for query testing and saved to a permanent file.

If you do not specify a filename, the contents of the command buffer are placed in a temporary file for editing. The command buffer contains the most recently executed SQL statement or tool command, with the exception of the EDIT and RUN commands. These cannot be edited in the command buffer.
After editing files, you can use the RUN command to execute them. For more information about the RUN command, refer to “Reading Input” on page 2-17.

To end an editing session, enter the editor’s termination command. The editor quits and the tool prompt is displayed.

**Examples**

Suppose your query named *store_rank* retrieves a ranking of stores in the Western region based on sales during the month of March, and you want to see the same figures for the Southern region. You can open your query text file with an editor, change the search condition on the region column, and use the RUN command to submit the query to the database.

1. First run the existing query and examine the result.

   ```sql
   RISQL> run store_rank ;
   RISQL> select store_name, sum(dollars) as total_sales,
   > rank(sum(dollars)) as sales_rank
   > from market m join store s on m.mktkey = r.mktkey
   > join sales s on s.storekey = r.storekey
   > join period p on s.perkey = p.perkey
   > where year = 1995
   > and month = 'MAR'
   > and region = 'West'
   > group by store_name;
   
<table>
<thead>
<tr>
<th>STORE_NAME</th>
<th>TOTAL_SALES</th>
<th>SALES_RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cupertino Coffee Supply</td>
<td>18709.50</td>
<td>1</td>
</tr>
<tr>
<td>San Jose Roasting Company</td>
<td>17867.90</td>
<td>2</td>
</tr>
<tr>
<td>Java Judy’s</td>
<td>17826.25</td>
<td>3</td>
</tr>
<tr>
<td>Beaches Brew</td>
<td>17197.25</td>
<td>4</td>
</tr>
<tr>
<td>Roasters, Los Gatos</td>
<td>15159.05</td>
<td>5</td>
</tr>
<tr>
<td>Instant Coffee</td>
<td>14841.75</td>
<td>6</td>
</tr>
</tbody>
</table>
   
   SET ECHO has been turned on in this example to display the contents of *store_rank* as it executes. For more information about echoing input, refer to “Including Input in Output” on page 2-18.
2. Now use your editor to open and edit the file. (This example shows the vi editor.)

```
RISQL> edit store_rank ;
```

The file opens and you see the query.

```
select store_name, sum(dollars) as total_sales,
    rank(sum(dollars)) as sales_rank
from market m join store r on m.mktkey = r.mktkey
    join sales s on s.storekey = r.storekey
    join period p on s.perkey = p.perkey
where year = 1995
    and month = 'MAR'
    and region = 'West'
group by store_name;
```

3. Change the search condition from West to South.

```
select store_name, sum(dollars) as total_sales,
    rank(sum(dollars)) as sales_rank
from market m join store r on m.mktkey = r.mktkey
    join sales s on s.storekey = r.storekey
    join period p on s.perkey = p.perkey
where year = 1995
    and month = 'MAR'
    and region = 'South'
group by store_name;
```

4. Terminate the editing session and issue the RUN command to submit the new query to the database.

RISQL> run ;

**Note:** If the last command issued was EDIT, you need not specify a filename as an argument to the RUN command.

RISQL> select store_name, sum(dollars) as total_sales,
> rank(sum(dollars)) as sales_rank
> from market m join store r on m.mktkey = r.mktkey
> join sales s on s.storekey = r.storekey
> join period p on s.perkey = p.perkey
> where year = 1995
> and month = 'MAR'
> and region = 'South'
> group by store_name;

<table>
<thead>
<tr>
<th>STORE_NAME</th>
<th>TOTAL_SALES</th>
<th>SALES_RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moulin Rouge Roasting</td>
<td>22013.05</td>
<td>1</td>
</tr>
<tr>
<td>Miami Espresso</td>
<td>15875.40</td>
<td>2</td>
</tr>
<tr>
<td>Olympic Coffee Company</td>
<td>14889.75</td>
<td>3</td>
</tr>
<tr>
<td>Texas Teahouse</td>
<td>14401.75</td>
<td>4</td>
</tr>
</tbody>
</table>

Editing the temporary buffer is useful when you make typing errors while entering statements at the tool prompt. When you use the EDIT command without specifying a filename, the command buffer contains the last RISQL statement that was executed. This buffer is placed in a temporary file, which can be submitted again with the RUN command.

**Incomplete-String Prompt**

When your statement contains unmatched quotes, the tool displays an incomplete string prompt (INCOMPLETE STRING). This prompt indicates that your statement has not been executed and that the tool is waiting for more input.

For example, a long string literal used in a WHERE clause might be broken across multiple lines. If the tool comes to the end of a line and does not find a closing quote, it displays the incomplete string prompt. This behavior does not indicate that an error has occurred; the prompt is simply sending you a status message about the current line of code. When the tool reads the closing quote on a subsequent line, the continuation or default prompt is displayed.
Suppose you forget to type a closing quote on a literal string in a query statement. You can use the EDIT and RUN commands to correct the statement and execute it again.

1. Type the following query, omitting the closing quote on West as shown, and press Return:

```
RISQL> select store_name, sum(dollars) as total_sales,
>          rank(sum(dollars)) as sales_rank
> from market m join store r on m.mktkey = r.mktkey
> join sales s on s.storekey = r.storekey
> join period p on s.perkey = p.perkey
> where year = 1995
> and month = 'MAR'
> and region = 'West ;
```

2. The tool responds with the incomplete string prompt:

```
INCOMPLETE STRING>
```

The tool has not read the statement-terminating semicolon because it is placed inside a quoted literal. Instead, the tool is waiting for more input.

3. To execute the statement, enter a closing quote and another semicolon:

```
INCOMPLETE STRING>' ;
```

The warehouse returns an incorrect result because it does not find a match for the quoted literal, which included the semicolon.

```
STORE_NAME                     TOTAL_SALES     SALES_RANK
RISQL>
```

4. To correct the query statement, edit a temporary copy in the command buffer:

```
RISQL> edit ;
```

```
select store_name, sum(dollars) as total_sales,
    rank(sum(dollars)) as sales_rank
from market m join store r on m.mktkey = r.mktkey
    join sales s on s.storekey = r.storekey
    join period p on s.perkey = p.perkey
where year = 1995
    and month = 'MAR'
    and region = 'West ;
```

5. Type the missing quote at the end of line 8:

```
    and region = 'West' ;
```
Adding Comments to Statements

6. Quit the editor. You see the RISQL prompt.
7. Enter the RUN command to execute the contents of the temporary buffer:

```
RISQL> run ;
```

```
STORE_NAME                     TOTAL_SALES     SALES_RANK
Cupertino Coffee Supply               18709.50           1
San Jose Roasting Company             17867.90           2
Java Judy’s                           17826.25           3
Beaches Brew                          17197.25           4
Roasters, Los Gatos                   15159.05           5
Instant Coffee                        14841.75           6
```

You must issue the RUN command immediately after editing the file, closing it, and submitting it to the database, because the command buffer is overwritten with each data retrieval or command execution.

If you want to save the contents of the buffer, you can do so before terminating the editor session by saving it with a permanent filename in your own directory.

For complete syntax information for the EDIT command, refer to page 4-7. For more information about the RUN command, refer to “Reading Input” on page 2-17.

Adding Comments to Statements

When you write a complex SQL statement that you plan to use more than once, you might want to describe portions of the code by including comments. Annotating statements is useful when preparing them for less technical users. It is also useful when you use statements only occasionally and want to include reminders about the specific elements of the code.

The RISQL Entry Tool and RISQL Reporter accept two comment formats:

- C-program-style comments (`/* ... */`), which require a beginning marker and an ending marker and can be embedded in a line or span multiple lines.
- SQL-style comments (`-- ...`), which require only a beginning marker, must be placed at the end of a line, and are limited to one line.
Examples

Use C-program-style comments as shown below:

```sql
select city,
    sum(quantity) as yr_1995,
    tertile(sum(quantity)) as qty_rank
/* calculates H, M, L ranking of quantity totals */
from market ma join store st
    on ma.mktkey = st.mktkey
    join sales sa on st.storekey = sa.storekey
    join product pr on pr.prodkey = sa.prodkey
    join period pe on pe.perkey = sa.perkey
where year = 1995
    and prod_name = 'Expresso XO'
    and region in ('West', 'South')
group by city;
```

Note: If you use a C-program-style comment (/*...*/) after the semicolon, the ending marker must occur on the same line as the beginning marker.

Use SQL-style comments as shown below:

```sql
select city,
    sa.perkey, --Period key column from Sales table
    month,
    dollars,
    ratiotoreport (dollars)*100 as ratio
--Ratio of $ to total (as percentage)
from market ma join store st
    on ma.mktkey = st.mktkey
    join sales sa on st.storekey = sa.storekey
    join product pr on pr.prodkey = sa.prodkey
    and pr.classkey = sa.classkey
    join period pe on pe.perkey = sa.perkey
where prod_name like 'Xalapa%' 
    and qtr in ('Q3%', 'Q4%')
    and year = 1995
    and city in ('San Jose', 'Los Angeles')
order by city, sa.perkey
--Orders data alphabetically by city, then by period key
reset by city;
```
Accessing the System Shell

The shell-escape command (!) suspends the current tool session, starts a new shell process, and lets you execute system commands. You need not terminate your shell-escape command statement with a semicolon (;).

You can issue a single system command by using the command as an argument to the ! command, or you can start an interactive shell session that suspends your tool session until you indicate that you want to resume it.

The exit command ends the shell session and reactivates the tool session. Always remember to terminate the interactive shell session and return to your RISQL Entry Tool or RISQL Reporter session. Otherwise, you might inadvertently leave yourself connected to a database, which might interfere with the work of other users.

For complete syntax information for the ! command, refer to page 4-4.

Examples

To list all files in the current UNIX directory that end with the suffix .risql, enter:

```
RISQL> !ls -l *.risql
-rw-rw-rw- 1 curly  pubs  13 Oct 23 13:23 cume.risql
-rw-rw-rw- 1 curly  pubs  13 Oct 23 13:24 mvgavg.risql
-rw-rw-rw- 1 curly  pubs  13 Oct 23 13:23 ratio.risql
-rw-rw-rw- 1 curly  pubs  13 Oct 23 13:24 share.risql
RISQL>
```

The operating system executes the command and returns you to the tool prompt.

To start an interactive shell session, enter the ! command and a system shell identifier:

```
RISQL> !csh
%
```
Using the RISQL Entry Tool and RISQL Reporter

Changing the Tool Prompt

To list all files in the current Windows NT directory that end with the suffix .risql, enter:

```
RISQL> !dir * .risql
CUME~1   RIS             80  10-31-97 12:54p cume.risql
AVG~1    RIS             80  10-31-97 12:54p avg.risql
SUM~1    RIS  80  10-31-97 12:54p sum.risql
MOVAG~1  RIS             80  10-31-97 12:55p movag.risql
4 file(s)              320 bytes
```

Changing the Tool Prompt

The RISQL Entry Tool and RISQL Reporter display three unique prompts: the first-line prompt (RISQL), the continuation prompt (>), and the incomplete string prompt (INCOMPLETE STRING). You can change these prompts with the SET PROMPT command. The prompt you specify must be enclosed in single quotes.

You can change the first-line prompt without changing the continuation or incomplete string prompts. However, you must change the first-line and continuation prompts or restate their default values if you want to change subsequent prompts.

For complete syntax information for changing these prompts, refer to page 4-60.

Examples

To change the first-line prompt to >>>, the continuation prompt to >>, and the incomplete string prompt to UNMATCHED QUOTE, enter:

```
RISQL> set prompt '>>> ' '>> ' 'UNMATCHED QUOTE> ' ;
```

The prompts are displayed as follows until you change them or end the tool session:

```
>>> select * from sales
>> where market = 'Los Angeles ;
UNMATCHED QUOTE>
```

To display the current prompts, enter:

```
RISQL> query prompt ;
```

To restore the default prompts, enter:

```
RISQL> reset prompt ;
```
Changing Database Passwords

The warehouse administrator uses the SQL GRANT command to authorize user access to a database and create usernames and passwords. For database security, users must often change assigned passwords to private ones.

For a complete discussion of the GRANT command and character limits on the lengths of usernames and passwords, refer to the SQL Reference Guide. For information about site restrictions imposed by the password security feature in the Enterprise Control and Coordination option, refer to the Warehouse Administrator’s Guide.

**Example**

User *curly* can change his current password *mysecret* to *mystery*, by entering:

```
RISQL> grant connect to curly with mystery ;
```

Administering the Database

Provided you have the correct database authorization and table privileges, you can use the RISQL Entry Tool or RISQL Reporter for database administration tasks. These tasks fall into three categories:

- Data definition
- Data control
- Data manipulation

The following sections contain examples of commonly used data administration commands. For complete syntax and option information for all warehouse-supported SQL commands and RISQL extensions, refer to the SQL Reference Guide. For a discussion of database administration tasks, refer to the Warehouse Administrator’s Guide.
Defining Database Objects

You can use the RISQL Entry Tool or RISQL Reporter to create, alter, and drop database objects such as tables, views, indexes, synonyms, and macros.

Examples

To create a table named product, enter:

```
RISQL> create table product 
> classkey integer not null, 
> prodkey integer not null, 
> prod_name char(30), 
> pkg_type char(20), 
> constraint prod_pk primary key (classkey, prodkey) 
> constraint prod_fkc foreign key (classkey) references class 
> (classkey) 
> maxrows per segment 2500;
```

To create a view containing only data for the Lotta Latte product, enter:

```
RISQL> create view l_latte 
> as select * 
> from product 
> where prod_name = 'Lotta Latte' ;
```

To drop a view, enter:

```
RISQL> drop view l_latte ;
```

Controlling Database and Table Access

As database administrator, you give database access to users and control access to database tables. As the creator of a table, you grant access to other users. These tasks are referred to as granting database authorization and granting table privileges.

Examples

To create the database user curly with the password mysecret, enter:

```
RISQL> grant connect to curly with mysecret ;
```

To give curly query SELECT privileges on the Product table, enter:

```
RISQL> grant select on product to curly ;
```
Changing Information in a Database

If you are the database administrator, or if the database administrator has given you data modification privileges, you can use the RISQL Entry Tool or RISQL Reporter to insert new rows, update data in rows, and delete rows.

Examples

To define a row in the Product table for a new brand of coffee named Aloha Kona, enter:

```
RISQL> insert into product
> (classkey,
>  prodkey,
>  prod_name,
>  pkg_type)
> values
>  (1,
>  5,
>  'Aloha Kona',
>  'No pkg'));
```

To update the Pkg_type data value for Aloha Kona, enter:

```
RISQL> update product
> set pkg_type = 'One-pound bag'
> where product = 'Aloha Kona' ;
```

To delete the Aloha Kona row from the Product table, enter:

```
RISQL> delete from product
> where product = 'Aloha Kona' ;
```
Creating Data Extraction Applications

A useful application of the RISQL Entry Tool or RISQL Reporter is to extract selected data from a warehouse database and place that data into a file for transfer to another program. You can also use a pipe (|) to send row data output directly from the RISQL Entry Tool or RISQL Reporter to another program, such as the Table Management Utility (TMU).

When using the RISQL Entry Tool or RISQL Reporter in data extraction applications, you typically want to restrict data output to only query result rows, omitting the product copyright notice, column headings, blank lines, and so on. Quiet mode restricts output to query results only.

Redirecting Extraction Results from the Shell

You can perform data extractions at the operating-system level directly from the UNIX or Windows NT shell (for example, in a shell script).

When you perform data extraction from the shell by specifying quiet mode (-q tool-startup option), all non-error output from the program is redirected to a file or pipe with standard UNIX or Windows NT file-redirection commands. Using the -q option also changes the default setting of the following commands:

- SET ECHO is set to OFF.
- SET ROW DATA is set to ONLY.

Examples

Use quiet mode (-q command-line option) to extract only query result rows to a file as shown below:

$s risql -q user_name password < weekly_extract > excel_import

To pipe extracted data to another program, such as the warehouse TMU, enter:

$s risql -q username password < weekly_extract | rb_tmu load.tmu username password
Redirecting Extraction Results from Within the Tool

When used together, the following commands capture query results in a file for transfer to another program during a RISQL Entry Tool or RISQL Reporter session:

- SET ROW DATA ONLY causes the query result columns to be returned without column headings or other extraneous information.
- SET ECHO OFF suppresses the RISQL prompt and input commands, as described on page 2-20.
- SET OUTPUT filename sends row data output to a file instead of the terminal (standard output), as described on page 2-19.

Example

To send only the row data for the specified period to a file named weekly_extract, enter:

```
RISQL> set row data only;
RISQL> set echo off;
RISQL> set output weekly_extract;
RISQL> select * > from period join sales
    > on period.perkey = sales.perkey
    > and quarter = 'Q1_95' ;
```

Exit Status Codes

The RISQL Entry Tool and RISQL Reporter use the following exit status codes:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Information or statistics messages might have been issued during execution, but no warning, error, or fatal messages were issued.</td>
</tr>
<tr>
<td>1</td>
<td>Warning messages cause an exit status of 1 to be returned.</td>
</tr>
<tr>
<td>2</td>
<td>Error messages cause an exit status of 2 to be returned.</td>
</tr>
<tr>
<td>3</td>
<td>Fatal messages cause an exit status of 3 to be returned.</td>
</tr>
</tbody>
</table>
The RISQL Entry Tool and RISQL Reporter return the highest status code encountered during processing. For example, if the RISQL Entry Tool generates only informational messages, it will return an exit status code of 0; if, however, it generates both information messages and a fatal message, it will return an exit status code of 3.
Formatting Reports

This chapter describes report-formatting commands included with the RISQL Reporter. These features are not available with the RISQL Entry Tool.

This chapter is divided into the following sections:

• Laying Out Reports
• Using the SET COLUMN Commands
• Saving Command Settings
• Creating a Report Title
• Defining the Page Length
• Skipping a Line or Starting a New Page When Data Changes
• Changing a Column Title
• Underlining Column Titles
• Justifying Data in Columns
• Changing the Width of a Column Field
• Widening Space Between Columns
• Hiding a Column
• Left-Filling Data
• Changing the Format of Numeric Data
• Displaying Signs for Positive Numbers
• Changing the Position of Signs
• Suppressing Duplicate Data in Columns
• Breaking Row Data Across Display Lines
• Defining a Data Label
• Displaying Nulls
• Displaying Current Settings
• Resetting Display Options to Default Values
• Sample Report 1
• Sample Report 2


**Laying Out Reports**

The following procedure describes how to format a report with the RISQL Reporter:

1. Write and execute your SQL query and verify the results.
2. Specify RISQL Reporter SET COLUMN commands to define the layout.
3. Execute the query to view the formatted results.

If you are developing a brand new report format on a large amount of data, you might find it useful to experiment with formatting commands on a subset of the output (for example, one product). When you have designed a satisfactory layout, submit the entire query to display the complete report.

After you have developed a layout, you can use the SAVE command to write the SET COLUMN commands for the format to a file. This file can then be executed with the RUN command in subsequent RISQL Reporter sessions and used as a template for similar reports in the future. For more information about the RUN command, refer to “Reading Input” on page 2-17.

**Using the SET COLUMN Commands**

Most of the RISQL Reporter formatting features are options of the SET COLUMN command. The syntax for the SET COLUMN command is straightforward:

1. Specify the command.
2. Specify one or more columns on which to work.
3. Specify a formatting option.

SET COLUMN commands can be set interactively at the RISQL prompt, in input files, or in RISQL Reporter initialization files (.rbretrace files).
### Column Specification

The following table describes valid column variables:

<table>
<thead>
<tr>
<th>Column Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Global column list</td>
</tr>
<tr>
<td>n</td>
<td>Relative column number</td>
</tr>
<tr>
<td>table.*</td>
<td>Table column list (sales.*)</td>
</tr>
<tr>
<td>column_name</td>
<td>Unqualified column name (dollars)</td>
</tr>
<tr>
<td>table.column_name</td>
<td>Table-qualified column name (sales.dollars)</td>
</tr>
</tbody>
</table>

### Column Variable Precedence

The following list describes each valid column specification and gives its level of precedence when more than one option is specified:

- Options specified for the global column list apply to all columns retrieved unless overwritten by a more specific column option specified later in the command syntax.
- Options specified for a relative column number are applied to the column in that position in the query. Relative column number options modify the characteristics of columns that are literals, subqueries, expressions, or display functions. Relative column numbers are retained even when columns are suppressed with the NO PRINT option of the SET COLUMN command. Relative column number options override global options.
- Options specified for a table column list are applied to all columns in that table. Table column list options override global and relative column number options.
- Options specified for unqualified column names are applied to all columns with that name, regardless of the table to which the column belongs. Column aliases defined in a SELECT statement can also be used as unqualified column names. Unqualified column options override global column lists, table column lists, and relative column number options.

**Note:** Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the SQL Reference Guide or the SQL Self-Study Guide.
• Options specified for table-qualified column names are applied only to the specific column. Column aliases defined in a SELECT statement cannot be used in table-qualified names. Table-qualified column name options take precedence over any other option specified for the column.

**Column Display Options**

The following table lists and describes column display options.

<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILL</td>
<td>Sets the left-fill character for data in a column.</td>
</tr>
<tr>
<td>FOLD LINE</td>
<td>Breaks a data row into two or more lines.</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Sets the format of numeric data in a column (DECIMAL, INTEGER, EXPONENTIAL).</td>
</tr>
<tr>
<td>IF NULL</td>
<td>Defines the string that is displayed when null is returned.</td>
</tr>
<tr>
<td>JUSTIFY</td>
<td>Sets the justification of data in a column.</td>
</tr>
<tr>
<td>NO PRINT</td>
<td>Prevents a column from being displayed in the result.</td>
</tr>
<tr>
<td>NO REPEAT</td>
<td>Suppresses the display of duplicate values in a column.</td>
</tr>
<tr>
<td>POSITIVE SIGN</td>
<td>Specifies whether to display or suppress the plus sign (+) on positive numeric data.</td>
</tr>
<tr>
<td>SKIP LINE</td>
<td>Inserts a blank row between data rows when a value in a column changes.</td>
</tr>
<tr>
<td>SKIP PAGE</td>
<td>Starts a new page when a value in a column changes.</td>
</tr>
<tr>
<td>SIGN</td>
<td>Defines position, either leading or trailing, for signs (+ or −) on numeric data.</td>
</tr>
<tr>
<td>SPACE</td>
<td>Sets lead spacing for data in a column.</td>
</tr>
<tr>
<td>TITLE</td>
<td>Defines a column title.</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>Displays a string of repeating characters between column heading and data.</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Sets the width of a column.</td>
</tr>
</tbody>
</table>
Some of the RISQL Reporter SET COLUMN commands affect others, so the sequence in which you issue them and the combination you use might be significant. The examples in this chapter are not sequential; to achieve the results shown here, you must reset any options that were previously set during your session or in your .rbretrc files so you begin with default settings.

For specific information about command syntax and a complete list of arguments to all options, refer to SET COLUMN beginning on page 4-21.

**Saving Command Settings**

You can save RISQL Reporter command settings made during a session with the SAVE command. Saving formatting settings lets you define and reuse a report style for different queries during different RISQL Reporter sessions. Files created with the SAVE command are standard ASCII files, and can be modified with a system editor.

SAVE writes declarations for the following commands to a specified file:

- SET PAGE LENGTH
- SET TITLE
- SET COLUMN (all options)

SAVE does not save settings for the following commands:

- SET ECHO
- SET EDITOR
- SET ERROR OUTPUT
- SET PROMPT
- SET ROW DATA
- SET STATS

If you execute the SAVE command and specify an existing file, the contents of the file are overwritten with the current settings.

The contents of the saved file can be executed with the RUN command (refer to “Reading Input” on page 2-17) to change default format settings for the current session, or copied to the RISQL Reporter initialization files (.rbretrc) to change defaults for all subsequent sessions. For complete syntax information for the SAVE command, refer to page 4-20. For more information about the initialization files, refer to Appendix B, “.rbretrc Files.”
Creating a Report Title

The SET TITLE command defines a single-line title for a report. Titles are left-justified and can contain one or more of the following predefined title variables, which display current values in the report:

- $DATE displays the date.
- $TIME displays the time.
- $PAGE displays the page number.

Title variables are embedded in text strings and are case-sensitive.

For complete syntax information for the SET TITLE command, refer to page 4-64.

Examples

The following command defines a title for the report shown below:

```sql
RISQL> set title 'WEIGHT RANKING IN WEST & SOUTH Report run: $DATE at $TIME' ;
RISQL> run wgt.rank ;
WEIGHT RANKING IN WEST & SOUTH Report run: 10/15/95 at 05:12:52

CITY         YR_1995   W_RK
Miami         5694.00  M
New Orleans   3578.00  M
Houston       367.00   L
San Francisco 11056.00 H
Los Angeles   9570.00  H
```

Notice that a blank line is displayed between the report title and the column titles.

To display the current title, enter:

```sql
RISQL> query title ;
```

To delete the title, enter:

```sql
RISQL> reset title ;
```
Defining the Page Length

The SET PAGE LENGTH command lets you specify the maximum number of lines printed on each page of a report. Changing the page length is useful when you want to customize your display for a specific terminal screen size, terminal emulator window size, or default printer page size.

The following lines can be displayed at the top of each report page:

- Report title followed by a blank line
- Column title
- Column underline

The default page length is unlimited; the report title, column titles, and column title underlines are printed only once at the beginning of the report. To break the report into several pages, you can provide a line-number argument to the SET PAGE LENGTH command. The column title information is then repeated at the top of each new page.

The RISQL Reporter displays lines of data until either a page break is indicated by a SKIP PAGE option or the maximum number of report lines, as defined by the SET PAGE LENGTH command, is reached. For more information about the SKIP PAGE option, refer to page 3-9.

The number of report lines displayed on a page is determined by subtracting the heading option lines (report title lines, the column title line, and the column underscore line) and the top and bottom margin allowance lines (4) from the specified page length. The minimum allowable page length is the number needed for the report title and column titles.

The SET PAGE LENGTH command is also affected by the FOLD LINE option of the SET COLUMN command. The precedence of these format settings is important and is described in detail on page 4-57.
Examples

The following setting produces a report that prints 20 lines, then breaks and prints the column titles at the top of the next page before printing another 20 lines.

\[ \text{RISQL}> \text{set page size 20 ;} \]

For an example of page-defined output, refer to the sample reports at the end of this chapter.

To display the current page length setting, enter:

\[ \text{RISQL}> \text{query page length ;} \]

To restore default page length, enter:

\[ \text{RISQL}> \text{reset page length ;} \]
Skipping a Line or Starting a New Page When Data Changes

The SKIP option of the SET COLUMN command skips a line or starts a new page in the result each time data changes in a specified column.

The RISQL Reporter compares columns from left to right. The leftmost column encountered with a SKIP setting causes the line to be skipped or the new page to be started.

If both a SKIP LINE and a SKIP PAGE are used on the same column, SKIP PAGE takes precedence. In general, SKIP PAGE should be used on columns to the left of columns with SKIP LINE settings.

Note: The RISQL Reporter automatically skips to a new page when the number of lines on a page, as specified with SET PAGE LENGTH (refer to page 3-7), is exceeded.

For complete syntax information for the SKIP option, refer to page 4-40.

Examples

The following report lists cities in the southern and western regions by district. Each time a value changes in the District column, the RISQL Reporter skips a line.

```
RISQL> set column district skip line;
RISQL> select hq_city, district
> from market
> where region = 'South'
> or region = 'West'
> order by district;

HQ_CITY  DISTRICT
Atlanta   Atlanta
Miami     Atlanta

Phoenix   Los Angeles
Los Angeles Los Angeles

New Orleans New Orleans
Houston   New Orleans

San Jose   San Francisco
San Francisco
Oakland   San Francisco
```

To print the data for each district on a separate page, enter:

```
RISQL> set column district skip page;
```
By default, internal column names stored in a warehouse database are displayed as titles in a result. Internal names are often abbreviated strings and cannot contain spaces; therefore, they are not always appropriate for report writing. Calculated columns resulting from SQL functions or RISQL extensions have no title by default.

You can use the TITLE option of the SET COLUMN command to define single- or multi-word column titles in a result. You must enclose your column title definition in single quotes.

**Note:** Column titles also can be defined with column aliases in a query statement, but column aliases are displayed in uppercase and cannot contain spaces.

For complete syntax information for the column TITLE option, refer to page 4-45.

**Example**

To display the title RUNNING TOTAL above the CUME column, enter:

```
RISQL> set column 4 title 'RUNNING TOTAL' right;
RISQL> select prod_name, city, quantity, cume(quantity)
> from store st
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
> or prod_name like 'Cafe%')
> and city like 'New O%'
> and week = 32
> and year = 1995
> order by prod_name, quantity asc;
```

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>RUNNING TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>43</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>58</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>92</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>111</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>130</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>145</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>160</td>
</tr>
</tbody>
</table>
Underlining Column Titles

The UNDERLINE option of the SET COLUMN command lets you underline one or more column titles with a horizontal character string. The hyphen character (-) is displayed by default, but you can change it to any single character you choose. For example, an equal sign (=) provides a double underline.

The UNDERLINE option has two formats:

- LONG underscores the title across the entire width of the column.
- SHORT underscores only the displayed width of the title text, no matter what the width of the column. When you customize the underline character, SHORT is the default format.

For complete syntax information for the UNDERLINE option, refer to page 4-47.

Examples

To display a double underline across the entire width of all columns, enter:

```sql
RISQL> set column * underline '=' long;
RISQL> select prod_name, city, quantity,
>           cume(quantity) as running_total
> from store st
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
>       or prod_name like 'Cafe%')
>       and city like 'New O%
>       and week = 32
>       and year = 1995
>       order by prod_name, quantity asc;
```

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>RUNNING TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>43</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>58</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>92</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>111</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>130</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>145</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>160</td>
</tr>
</tbody>
</table>
To display asterisks that underline column titles only, enter:

RISQL> set column * underline '*' short;
RISQL> select prod_name, city, quantity,
>    cume(quantity) as running_total
> from store st
>   join sales sa on st.storekey = sa.storekey
>   join product pr on pr.prodkey = sa.prodkey
>   join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
>   or prod_name like 'Cafe%')
>   and city like 'New O%'
>   and week = 32
>   and year = 1995
> order by prod_name, quantity asc;

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>RUNNING TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>43</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>58</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>92</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>111</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>130</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>145</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>160</td>
</tr>
</tbody>
</table>
**Justifying Data in Columns**

The `JUSTIFY` option of the `SET COLUMN` command lets you right- or left-justify data in a column. By default, numeric data is right-justified and character data is left-justified.

For complete syntax information for the `JUSTIFY` option, refer to page 4-33.

**Example**

To left-and right-justify numeric columns, enter:

```sql
RISQL> set column 3 justify right;
RISQL> set column 4 justify left;
RISQL> select prod_name, city, quantity, rank(quantity) as qty_rank
from store st
join sales sa on st.storekey = sa.storekey
join product pr on pr.prodkey = sa.prodkey
join period pe on pe.perkey = sa.perkey
where (prod_name like 'Expresso%' or prod_name like 'Cafe%')
and city like 'New O%' and week = 32 and year = 1995
order by prod_name, quantity asc;
```

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>QTY_RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>5</td>
</tr>
</tbody>
</table>
Changing the Width of a Column Field

The default width of a column field is the datatype width that was defined by a table-definition statement when the table was created in the database. For example, if a column is defined as `char (10)`, the default column width is 10 characters in the result.

You can change the width of a column field in your report with the WIDTH option of the SET COLUMN command. This option accepts a numeric value for the number of display spaces in a column and is valid for any column type.

Column width settings can be influenced by the JUSTIFY option of the SET COLUMN command. If the alignment of data appears irregular in your report, consider the effect of justification and column width settings.

Column titles returned from the database, defined with the TITLE option of the SET COLUMN command, or specified as column aliases in the select list are truncated when they exceed the defined width of a column. For information about column aliases, refer to the SQL Reference Guide.

For complete syntax information for the WIDTH option, refer to page 4-49.

**Example**

The following command sets a width of 9 display spaces for the Dollars column.

```
RISQL> set column dollars width 9 ;
```

**Note:** If you set the SIGN TRAILING option on a right-justified decimal or integer column, the rightmost character position in the column is always reserved for a sign character. Therefore, the number of characters of a numeric result that this column can contain is reduced by one. For more information about the SIGN option, refer to page 3-19.
Widening Space Between Columns

The SPACE option of the SET COLUMN command defines lead spacing, which widens the margin between columns and makes output easier to read. The default column spacing is one space for all columns except the leftmost column, which has no leading spaces.

For complete syntax information for the SPACE option, refer to page 4-42.

Example

To indent the result display, set the lead spacing on the leftmost column to 5 as follows:

```
RISQL> set column prod_name space 5;
RISQL> select prod_name, city, quantity, dollars
> from store st
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%' 
>     or prod_name like 'Cafe%')
>     and city like 'New O%' 
>     and week = 32 
>     and year = 1995 
> order by prod_name, quantity asc;
```

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>DOLLARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>112.00</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>112.00</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>63.75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>63.75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>72.25</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>72.25</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>161.50</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>161.50</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>90.00</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>90.00</td>
</tr>
</tbody>
</table>

You can set spacing to zero to simulate concatenated columns. For example, you might want to concatenate columns when working with a database of address information.
Hiding a Column

The NO PRINT option of the SET COLUMN command hides a column in the result. Hiding columns is often useful when you want to order your data by a particular column, but do not want that column to be displayed or printed in the report.

When using relative column numbers to replace column variables, remember that all columns are counted, even those that are hidden. For example, when you hide column 3, column 4 of the result is actually displayed in the column 3 position. Any settings on the displayed column in position 3 must be set on column 4.

For complete syntax information for the NO PRINT option, refer to page 4-33.

Example

Suppose you do not want certain columns of data to display in your report. You can use the NO PRINT option to hide those columns:

```
RISQL> set column state no print;
RISQL> set column store_type no print;
RISQL> select store_name, store_type, state,
    >       sum(dollars) as sales
    > from store natural join sales
    > where store_type = 'Large'
    > group by store_name, store_type, state;
STORE_NAME                SALES
Beaches Brew              459248.50
San Jose Roasting Company 659385.35
Olympic Coffee Company   451983.05
Miami Espresso           465057.70
```
**Left-Filling Data**

The FILL option of the SET COLUMN command left-fills data in a column. Left-filling data is often useful when you have numbers of unequal length in a column (for example, employee numbers). The default fill character is a space, but you can change it to a character of your choice.

When you format integer or decimal data with leading zero characters, the plus and minus signs (+ and –) are placed in the leftmost character position. When you fill with any other character, the minus sign is displayed immediately to the left of the leftmost numeric data character and the plus sign is not displayed at all.

When the FILL option is in use, the JUSTIFY LEFT option has no effect on numeric data in the column because every space in a data field is filled. For more information about justifying data, refer to “Justifying Data in Columns” on page 3-13.

For complete syntax information for the FILL option, refer to page 4-24.

**Example**

To left-fill a column with leading zeros, enter:

```
RISQL> set column emp_no fill 0;
RISQL> select emp_no from employees;
```

Data in the Emp_No column is displayed in the result as follows:

```
EMP_NO
00012
00036
00037
00052
00123
00320
```
Changing the Format of Numeric Data

Data is displayed in its original datatype by default. The FORMAT option of the SET COLUMN command offers three options for changing the format of numeric data:

- DECIMAL format displays all real, float, double-precision, decimal, and numeric types in typical decimal format (2 decimal places, by default). The number of decimal places can be changed by supplying a numeric argument to the DECIMAL option. The maximum number of decimal places is 99. Data values that have more decimal places than specified by this option are rounded off.

- INTEGER format displays all numbers as integers. In this format, decimal values are rounded off and no decimal places are displayed. Negative numbers are also rounded off; –1.2 becomes –1 and –1.5 becomes –2.

- EXPONENTIAL format displays all numbers in exponential format. The number of decimal places displayed is equal to the column width minus 8. If the column width is less than 9, asterisks are displayed.

If the value exceeds the width or format limitation of the column, asterisks (*) are printed in the field. Asterisks occur most often in columns that are formatted as decimal or integer. However, if exponential format is specified and the column width is less than 9, asterisks are displayed.

For complete syntax information for the numeric format options, refer to page 4-28.

Examples

RISQL> set column weight format decimal 4 ;
123.4567

RISQL> set column weight format integer ;
123

RISQL> set column weight format exponential ;
1.2345E+2
Displaying Signs for Positive Numbers

By default, negative integer and decimal values in a result set are shown with a minus sign (–), but no sign is shown for positive numbers. The POSITIVE SIGN option of the SET COLUMN command lets you specify that plus signs (+) for positive numbers should also be shown.

For complete syntax information for the POSITIVE SIGN option, refer to page 4-36.

Example

To display the plus sign, specify the SHOW keyword in your SET COLUMN command, as follows:

RISQL> set column dollars show positive sign;
RISQL> select dollars from sales_chg;
   DOLLARS
   -132.13
   +566.00
   -212.98

To hide the plus sign (after specifying that it is to be displayed), specify the HIDE keyword in your SET COLUMN command, as follows:

RISQL> set column dollars hide positive sign;
RISQL> select dollars from sales_chg;
   DOLLARS
   -132.13
   566.00
   -212.98
Changing the Position of Signs

By default, any displayed plus (+) and minus (–) signs for integer and decimal values in a result set are placed to the left of the numbers in a column. The SIGN option of the SET COLUMN command lets you reposition the signs to the right of a displayed number.

When you specify the TRAILING option, the rightmost character position in a column is reserved for the sign. When positive values are present in this column, either a plus sign or a space character occupies the rightmost space, depending on the setting of the SET COLUMN POSITIVE SIGN command. Minus signs are always displayed for negative numbers.

For complete syntax information for the SIGN option, refer to page 4-38.

Example

To move the sign to the right of column data, specify the TRAILING keyword in your SET COLUMN command, as shown below:

```
RISQL> set column dollars sign trailing;
RISQL> select dollars from sales_chg;
DOLLARS
132.13–
566.00
212.98–
```

Note that the plus signs are not displayed in this example.
Suppressing Duplicate Data in Columns

The NO REPEAT option of the SET COLUMN command suppresses the display of duplicate values in a result column. When you specify NO REPEAT and data in a named column is the same as the data in the preceding row, the RISQL Reporter hides succeeding identical data in the result. Thus, the NO REPEAT option is useful in reports ordered by a column with multiple identical entries.

When multiple NO REPEAT columns are specified, the RISQL Reporter searches for NO REPEAT values from left to right.

If a page break occurs when data in a column is suppressed, the suppressed values are displayed at the top of the new report page.

For complete syntax information for the NO REPEAT option, refer to page 4-34.

Example

To prevent the state and store type from repeating on each row, enter:

```
RISQL> set column state no repeat;
RISQL> set column store_type no repeat;
RISQL> select store_name, state, store_type, sum(dollars) as sales
  > from store natural join sales
  > where store_type = 'Large'
  > group by store_name, store_type, state;
```

<table>
<thead>
<tr>
<th>STORE_NAME</th>
<th>STATE</th>
<th>STORE_TYPE</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beaches Brew</td>
<td>CA</td>
<td>Large</td>
<td>459248.50</td>
</tr>
<tr>
<td>San Jose Roasting Company</td>
<td>CA</td>
<td>Large</td>
<td>659385.35</td>
</tr>
<tr>
<td>Olympic Coffee Company</td>
<td>GA</td>
<td>Large</td>
<td>451983.05</td>
</tr>
<tr>
<td>Miami Espresso</td>
<td>FL</td>
<td>Large</td>
<td>465057.70</td>
</tr>
</tbody>
</table>
Breaking Row Data Across Display Lines

The FOLD LINE option of the SET COLUMN command wraps data rows across multiple lines. This command causes the RISQL Reporter to insert a new-line character ("\n") before displaying a specified column, letting you break a long row of data into several rows so the total length of the line does not exceed the width of your output device (such as a terminal screen or printer).

For complete syntax information for the FOLD LINE option, refer to page 4-26.

Example

To display the product name on a line above the city, quantity, and running total information in the following report, set the FOLD LINE option on the City column as follows:

RISQL> set column city fold line;
RISQL> select prod_name, city, quantity,
> cume(quantity) as running_total
> from store st
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
> or prod_name like 'Cafe%')
> and city like 'New O%
> and week = 32
> and year = 1995
> order by prod_name, quantity asc;

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>RUNNING_TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>43</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>111</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>130</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>145</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>160</td>
</tr>
</tbody>
</table>
Defining a Data Label

The LABEL option of the SET COLUMN SPACE command lets you annotate data in the result. For example, suppose your result returns dollar sales figures; when you define a dollar-sign label for the Dollars column, each value in the column is prefixed with a dollar sign ($).

Data labels are displayed to the left of each value in a column. The label string you define must be enclosed in single quotes.

Note: The width of a column is automatically expanded by the length of your label string. If you use the LABEL option of the SET COLUMN command with the WIDTH option, you need not include the length of the label string in the width specification. For information about the WIDTH option, refer to “Changing the Width of a Column Field” on page 3-14.

For complete syntax information for the SPACE and LABEL options, refer to page 4-42.

Examples

To include dollar signs in the sales figures column, first narrow the width of the column so that the $ symbol appears immediately to the left of the data, then define the label as follows:

RISQL> set column total width 9;
RISQL> set column total space 1 '$';
RISQL> select district,
> sum(dollars) as total
> from market ma join store st
> on ma.mktkey = st.mktkey
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
> where prod_name like 'Cafe%'
> and year = 1995
> group by district
> order by total asc;
The LABEL option also works well for custom report formatting when used with SET COLUMN * FOLD LINE and SET ROW DATA ONLY, which puts each column value on a separate line.

RISQL> set column * fold line;
RISQL> set row data only;
RISQL> set column mktkey space 1 'Mkt: ';  
RISQL> set column hq_city space 4 'City: '; 
RISQL> set column hq_state space 4 'State: '; 
RISQL> set column district space 4 'District: '; 
RISQL> set column region space 4 'Region: '; 
RISQL> set column city skip line;
RISQL> select * from market 
> where region = 'West';

Mkt: 14
    City: San Jose
    State: CA
    District: San Francisco
    Region: West

Mkt: 15
    City: San Francisco
    State: CA
    District: San Francisco
    Region: West

Mkt: 16
    City: Oakland
    State: CA
    District: San Francisco
    Region: West

Mkt: 17
    City: Los Angeles
    State: CA
    District: Los Angeles
    Region: West

Mkt: 19
    City: Phoenix
    State: AZ
    District: Los Angeles
    Region: West
Displaying Nulls

The IF NULL option of the SET COLUMN command defines a null-replacement string. The null string you supply must be enclosed in single quotes when defined. The default string is NULL.

Example

To display nulls as “N/A,” enter:

```
RISQL> set column mgrno if null 'N/A';
RISQL> select name, department, title, mgrno, empno
> from dept natural join employees
> where department like 'Sales%';
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>DEPARTMENT</th>
<th>TITLE</th>
<th>MGRNO</th>
<th>EMPNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blake, Joe</td>
<td>Sales</td>
<td>AcctRep</td>
<td>N/A</td>
<td>1874</td>
</tr>
<tr>
<td>Peach, Etta</td>
<td>Sales</td>
<td>AcctRep</td>
<td>N/A</td>
<td>1051</td>
</tr>
<tr>
<td>Slo, Monica</td>
<td>Sales</td>
<td>Mgr</td>
<td>089</td>
<td>0498</td>
</tr>
<tr>
<td>Tepio, Curtis</td>
<td>Sales</td>
<td>AcctRep</td>
<td>N/A</td>
<td>1179</td>
</tr>
</tbody>
</table>

For complete syntax information for the IF NULL option, refer to page 4-30.
Displaying Current Settings

The QUERY COLUMN command displays settings for SET COLUMN options defined for the current session.

You can display option settings for:

- A single column, which displays the specified options set on that column.
- A column list, which displays options set on columns with that list specification (for example, market.*).
- All columns with options set with the global column wildcard (*).

Note: Displaying global options displays only those options that were set with the global column wildcard character (*), not all column settings.

Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the SQL Reference Guide.

For complete syntax information for the QUERY COLUMN command, refer to page 4-11.

Example

To display the format settings on the Dollars column, enter:

```
RISQL> query column dollars;
set column dollars width 8;
set column dollars title 'Dollars';
```
Resetting Display Options to Default Values

The RESET COLUMN command restores one or more of the column display options to its default value.

You can reset options on:

- A single column, which resets the specified options set on that column.
- A column list, which resets options set on columns with that list specification (for example, market.*).
- All columns with options set with the global column wildcard (*).

When you reset an option for a single column, options on other columns in the result are left intact. When you reset an option on a column list, only the specified option that was set with that particular column-list variable is reset. Resetting global options affects only those options that were set with the global column wildcard character (*), not all column settings.

Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the SQL Reference Guide.

For complete syntax information for the RESET COLUMN command, refer to page 4-15.

Examples

To reset an option, you must use the same type of column variable and option argument that you used when setting the option. For example, the following command sets the width of the sales column:

```
set column sales.* width 8;
```

to reset the default column width for the sales column, enter:

```
reset column sales.* width;
```

However, it does not reset the column width if it was set with the following command:

```
set column sales.dollars width 8;
```
Sample Report 1

The following report lists coffee sales in San Jose for the first quarter of 1994. The data is listed by product. Percentages are provided to show the most profitable month for each product for the quarter.

Query

RISQL> > select pj.prod_name, dj.month, sum(dollars) as mon_sales_94,  
> dec(100 * sum(dollars)/  
> (select sum(si.dollars)  
> from store ri natural join sales si  
> natural join product pi  
> natural join period di  
> where di.qtr = dj.qtr  
> and di.year = dj.year  
> and pi.prod_name = pj.prod_name  
> and pi.pkg_type = pj.pkg_type  
> and ri.city = rj.city), 7, 2) as pct_qtr1  
> from store rj natural join sales sj  
> natural join product pj  
> natural join period dj  
> where rj.city = 'San Jose'  
> and dj.year = 1994  
> and dj.qtr = 'Q1_94'  
> and pj.pkg_type = 'One-pound bag'  
> group by pj.prod_name, pj.pkg_type, dj.month,  
> dj.qtr, dj.year, rj.city  
> order by pj.prod_name, pct_qtr1 desc;

Query Output

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>MONTH</th>
<th>MON_SALES_94</th>
<th>PCT_QTR1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aroma Roma</td>
<td>FEB</td>
<td>688.75</td>
<td>39.91</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>JAN</td>
<td>594.50</td>
<td>34.45</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>MAR</td>
<td>442.25</td>
<td>25.63</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>MAR</td>
<td>742.00</td>
<td>40.61</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>JAN</td>
<td>600.50</td>
<td>32.86</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>FEB</td>
<td>484.50</td>
<td>26.51</td>
</tr>
<tr>
<td>Colombiano</td>
<td>JAN</td>
<td>770.75</td>
<td>41.66</td>
</tr>
<tr>
<td>Colombiano</td>
<td>FEB</td>
<td>593.50</td>
<td>32.08</td>
</tr>
<tr>
<td>Colombiano</td>
<td>MAR</td>
<td>485.75</td>
<td>26.25</td>
</tr>
<tr>
<td>Demitasse Ms</td>
<td>MAR</td>
<td>789.25</td>
<td>35.98</td>
</tr>
<tr>
<td>Demitasse Ms</td>
<td>JAN</td>
<td>768.75</td>
<td>35.04</td>
</tr>
<tr>
<td>Demitasse Ms</td>
<td>FEB</td>
<td>635.50</td>
<td>28.97</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>JAN</td>
<td>849.75</td>
<td>50.99</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>MAR</td>
<td>528.00</td>
<td>31.68</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>FEB</td>
<td>288.75</td>
<td>17.32</td>
</tr>
<tr>
<td>La Antigua</td>
<td>MAR</td>
<td>1131.50</td>
<td>39.70</td>
</tr>
</tbody>
</table>
### Formatting Reports

**Sample Report 1**

<table>
<thead>
<tr>
<th>Product</th>
<th>Month</th>
<th>Sales</th>
<th>Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td>La Antigua</td>
<td>JAN</td>
<td>888.75</td>
<td>31.18</td>
</tr>
<tr>
<td>La Antigua</td>
<td>FEB</td>
<td>829.25</td>
<td>29.10</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>JAN</td>
<td>901.00</td>
<td>36.55</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>MAR</td>
<td>892.50</td>
<td>36.20</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>FEB</td>
<td>671.50</td>
<td>27.24</td>
</tr>
<tr>
<td>NA Lite</td>
<td>FEB</td>
<td>531.00</td>
<td>43.38</td>
</tr>
<tr>
<td>NA Lite</td>
<td>JAN</td>
<td>441.00</td>
<td>36.02</td>
</tr>
<tr>
<td>NA Lite</td>
<td>MAR</td>
<td>252.00</td>
<td>20.58</td>
</tr>
<tr>
<td>Veracruzano</td>
<td>FEB</td>
<td>1008.00</td>
<td>45.32</td>
</tr>
<tr>
<td>Veracruzano</td>
<td>JAN</td>
<td>632.00</td>
<td>28.41</td>
</tr>
<tr>
<td>Veracruzano</td>
<td>MAR</td>
<td>584.00</td>
<td>26.25</td>
</tr>
<tr>
<td>Xalapa Lapa</td>
<td>MAR</td>
<td>864.00</td>
<td>50.52</td>
</tr>
<tr>
<td>Xalapa Lapa</td>
<td>JAN</td>
<td>639.00</td>
<td>37.36</td>
</tr>
<tr>
<td>Xalapa Lapa</td>
<td>FEB</td>
<td>207.00</td>
<td>12.10</td>
</tr>
</tbody>
</table>
Format Settings

To improve the readability of the preceding report, format settings are used to make the following changes:

- Define a default page length and a report title.
- Underline the column titles.
- Suppress repeated values in the Prod_Name column.
- Skip a line after each Product.
- Specify titles for each column that include upper- and lowercase characters and spaces.
- Adjust the column width and spacing.

The format settings are as follows:

```plaintext
set page length 43 using blank lines;
set title 'Q1 Sales in San Jose $DATE $TIME Pg: $PAGE';
set column * underline '-' short;
set column prod_name no repeat;
set column prod_name skip line;
set column 1 title 'Product';
set column 1 width 15;
set column 1 space 2;
set column 2 title 'Month';
set column 2 space 3;
set column 3 title 'Monthly Sales';
set column 3 space 5;
set column 3 width 8;
set column 4 title 'Pct. of Qtr';
set column 4 space 5;
set column 4 width 12;
```
## Formatting Reports

### Sample Report 1

**Report**

Q1 Sales in San Jose 10/28/97 05:13:34 Pg: 1

<table>
<thead>
<tr>
<th>Product</th>
<th>Month</th>
<th>Monthly</th>
<th>Pct. of Qtr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aroma Roma</td>
<td>FEB</td>
<td>688.75</td>
<td>39.91</td>
</tr>
<tr>
<td></td>
<td>JAN</td>
<td>594.50</td>
<td>34.45</td>
</tr>
<tr>
<td></td>
<td>MAR</td>
<td>442.25</td>
<td>25.63</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>MAR</td>
<td>742.00</td>
<td>40.61</td>
</tr>
<tr>
<td></td>
<td>JAN</td>
<td>600.50</td>
<td>32.86</td>
</tr>
<tr>
<td></td>
<td>FEB</td>
<td>484.50</td>
<td>26.51</td>
</tr>
<tr>
<td>Colombiano</td>
<td>JAN</td>
<td>770.75</td>
<td>41.66</td>
</tr>
<tr>
<td></td>
<td>FEB</td>
<td>593.50</td>
<td>32.08</td>
</tr>
<tr>
<td></td>
<td>MAR</td>
<td>485.75</td>
<td>26.25</td>
</tr>
<tr>
<td>Demitasse Ms</td>
<td>MAR</td>
<td>789.25</td>
<td>35.98</td>
</tr>
<tr>
<td></td>
<td>JAN</td>
<td>768.75</td>
<td>35.04</td>
</tr>
<tr>
<td></td>
<td>FEB</td>
<td>635.50</td>
<td>28.97</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>JAN</td>
<td>849.75</td>
<td>50.99</td>
</tr>
<tr>
<td></td>
<td>MAR</td>
<td>528.00</td>
<td>31.68</td>
</tr>
<tr>
<td></td>
<td>FEB</td>
<td>288.75</td>
<td>17.32</td>
</tr>
<tr>
<td>La Antigua</td>
<td>MAR</td>
<td>1131.50</td>
<td>39.70</td>
</tr>
<tr>
<td></td>
<td>JAN</td>
<td>888.75</td>
<td>31.18</td>
</tr>
<tr>
<td></td>
<td>FEB</td>
<td>829.25</td>
<td>29.10</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>JAN</td>
<td>901.00</td>
<td>36.55</td>
</tr>
<tr>
<td></td>
<td>MAR</td>
<td>892.50</td>
<td>36.20</td>
</tr>
<tr>
<td></td>
<td>FEB</td>
<td>671.50</td>
<td>27.24</td>
</tr>
<tr>
<td>NA Lite</td>
<td>FEB</td>
<td>531.00</td>
<td>43.38</td>
</tr>
<tr>
<td></td>
<td>JAN</td>
<td>441.00</td>
<td>36.02</td>
</tr>
<tr>
<td></td>
<td>MAR</td>
<td>252.00</td>
<td>20.58</td>
</tr>
<tr>
<td>Veracruzano</td>
<td>FEB</td>
<td>1008.00</td>
<td>45.32</td>
</tr>
<tr>
<td></td>
<td>JAN</td>
<td>632.00</td>
<td>28.41</td>
</tr>
<tr>
<td></td>
<td>MAR</td>
<td>584.00</td>
<td>26.25</td>
</tr>
</tbody>
</table>
### Q1 Sales in San Jose 10/28/97 05:13:34 Pg: 2

<table>
<thead>
<tr>
<th>Product</th>
<th>Month</th>
<th>Monthly</th>
<th>Pct. of Qtr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xalapa Lapa</td>
<td>MAR</td>
<td>864.00</td>
<td>50.52</td>
</tr>
<tr>
<td></td>
<td>JAN</td>
<td>639.00</td>
<td>37.36</td>
</tr>
<tr>
<td></td>
<td>FEB</td>
<td>207.00</td>
<td>12.10</td>
</tr>
</tbody>
</table>
### Sample Report 2

The following report lists quarterly coffee sales in San Jose for the Aroma Roma, Colombiano, and Lotta Latte brands. This report includes subtotals for each quarter and each year and shows the average cost per unit by quarter.

**Query**

```
select prod_name, year, qtr, sum(dollars), sum(quantity), dec(sum(dollars)/sum(quantity),7,2)
> from market ma join store st
> on ma.mktkey = st.mktkey
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
where prod_name in ('Aroma Roma', 'Lotta Latte', 'Colombiano')
> and city = 'San Jose'
> and year in (1994, 1995)
group by prod_name, year, qtr
order by prod_name, year, qtr
break by year summing 4, 5;
```

**Query Output**

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>YEAR</th>
<th>QTR</th>
<th>Dollars</th>
<th>Quantity</th>
<th>Avg. Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aroma Roma</td>
<td>1994</td>
<td>Q1_94</td>
<td>14417.50</td>
<td>2398</td>
<td>6.01</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>1994</td>
<td>Q2_94</td>
<td>12975.00</td>
<td>2220</td>
<td>5.84</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>1994</td>
<td>Q3_94</td>
<td>16423.50</td>
<td>2594</td>
<td>6.33</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>1994</td>
<td>Q4_94</td>
<td>14242.00</td>
<td>2304</td>
<td>6.18</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>1994</td>
<td>NULL</td>
<td>58058.00</td>
<td>9516</td>
<td>NULL</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>1995</td>
<td>Q1_95</td>
<td>17172.50</td>
<td>2766</td>
<td>6.20</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>1995</td>
<td>Q2_95</td>
<td>16658.00</td>
<td>2666</td>
<td>6.24</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>1995</td>
<td>Q3_95</td>
<td>15858.00</td>
<td>2582</td>
<td>6.14</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>1995</td>
<td>Q4_95</td>
<td>15942.50</td>
<td>2614</td>
<td>6.09</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>1995</td>
<td>NULL</td>
<td>65631.00</td>
<td>10628</td>
<td>NULL</td>
</tr>
<tr>
<td>Aroma Roma</td>
<td>NULL</td>
<td>NULL</td>
<td>123689.00</td>
<td>20144</td>
<td>NULL</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1994</td>
<td>Q1_94</td>
<td>16048.00</td>
<td>2726</td>
<td>5.88</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1994</td>
<td>Q2_94</td>
<td>15498.00</td>
<td>2628</td>
<td>5.89</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1994</td>
<td>Q3_94</td>
<td>13278.50</td>
<td>2260</td>
<td>5.87</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1994</td>
<td>Q4_94</td>
<td>12390.50</td>
<td>2176</td>
<td>5.69</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1994</td>
<td>NULL</td>
<td>57215.00</td>
<td>9790</td>
<td>NULL</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1995</td>
<td>Q1_95</td>
<td>16006.50</td>
<td>2752</td>
<td>5.81</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1995</td>
<td>Q2_95</td>
<td>17382.00</td>
<td>2882</td>
<td>6.03</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1995</td>
<td>Q3_95</td>
<td>15299.50</td>
<td>2624</td>
<td>5.83</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1995</td>
<td>Q4_95</td>
<td>17457.00</td>
<td>2702</td>
<td>6.46</td>
</tr>
<tr>
<td>Colombiano</td>
<td>1995</td>
<td>NULL</td>
<td>66145.00</td>
<td>10960</td>
<td>NULL</td>
</tr>
<tr>
<td>Colombiano</td>
<td>NULL</td>
<td>NULL</td>
<td>123360.00</td>
<td>20750</td>
<td>NULL</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>1994</td>
<td>Q1_94</td>
<td>24633.00</td>
<td>3258</td>
<td>7.56</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>1994</td>
<td>Q2_94</td>
<td>17433.00</td>
<td>2266</td>
<td>7.69</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>1994</td>
<td>Q3_94</td>
<td>17842.00</td>
<td>2482</td>
<td>7.18</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>1994</td>
<td>Q4_94</td>
<td>18047.00</td>
<td>2528</td>
<td>7.13</td>
</tr>
</tbody>
</table>
### Formatting Reports

#### Sample Report 2

<table>
<thead>
<tr>
<th>Product</th>
<th>Year</th>
<th>Quarter</th>
<th>Sales</th>
<th>Profit</th>
<th>Margin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lotta Latte</td>
<td>1994</td>
<td>NULL</td>
<td>77955.00</td>
<td>10534</td>
<td>NULL</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>1995</td>
<td>Q1_95</td>
<td>18753.00</td>
<td>2650</td>
<td>7.07</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>1995</td>
<td>Q2_95</td>
<td>24777.00</td>
<td>3310</td>
<td>7.48</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>1995</td>
<td>Q3_95</td>
<td>17091.00</td>
<td>2306</td>
<td>7.41</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>1995</td>
<td>Q4_95</td>
<td>14935.00</td>
<td>2096</td>
<td>7.12</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>1995</td>
<td>NULL</td>
<td>75556.00</td>
<td>10362</td>
<td>NULL</td>
</tr>
<tr>
<td>Lotta Latte</td>
<td>NULL</td>
<td>NULL</td>
<td>153511.00</td>
<td>20896</td>
<td>NULL</td>
</tr>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>400560.00</td>
<td>61790</td>
<td>NULL</td>
</tr>
</tbody>
</table>
**Format Settings**

To improve the readability of the preceding report, format settings are used to make the following changes:

- Define a title for the report.
- Specify titles for each column that include upper- and lowercase characters and spaces.
- Underline column titles.
- Adjust column widths and spacing.
- Skip a line between each year.
- Add labels.
- Suppress nulls (which appear in some BREAK BY fields).
- Suppress repeated values in the Product column.
- Custom-justify data in specified columns.
- Print data for each product on a separate page.

The format settings are as follows:

```sql
set title '1994-1995 Sales in S. J. by Brand $DATE Pg: $PAGE';
set column * if null ' ';
set column * underline '=' short;
set column prod_name title 'Product';
set column 1 no repeat;
set column 1 skip page;
set column year title 'Year';
set column year width 4;
set column year no repeat;
set column year skip line;
set column qtr title 'Qtr';
set column qtr width 5;
set column qtr space 2;
set column qtr justify left;
set column qtr no repeat;
set column 4 title 'Sales';
set column 4 width 10;
set column 4 space 2 '$';
set column 5 title 'Quantity';
set column 5 width 8;
set column 5 space 2;
set column 6 title 'Unit Cost';
set column 6 width 9;
set column 6 space 2;
```
<table>
<thead>
<tr>
<th>Product</th>
<th>Year</th>
<th>Qtr</th>
<th>Sales</th>
<th>Quantity</th>
<th>Unit Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aroma Roma</td>
<td>1994</td>
<td>Q1_94</td>
<td>$14417.50</td>
<td>2398</td>
<td>6.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q2_94</td>
<td>$12975.00</td>
<td>2220</td>
<td>5.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q3_94</td>
<td>$16423.50</td>
<td>2594</td>
<td>6.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q4_94</td>
<td>$14242.00</td>
<td>2304</td>
<td>6.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$58058.00</td>
<td>9516</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1995</td>
<td>Q1_95</td>
<td>$17172.50</td>
<td>2766</td>
<td>6.20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q2_95</td>
<td>$16658.00</td>
<td>2666</td>
<td>6.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q3_95</td>
<td>$15858.00</td>
<td>2582</td>
<td>6.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q4_95</td>
<td>$15942.50</td>
<td>2614</td>
<td>6.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$65631.00</td>
<td>10628</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$123689.00</td>
<td>20144</td>
<td></td>
</tr>
</tbody>
</table>
### 1994-1995 Sales in S. J. by Brand 10/28/97 Pg: 2

<table>
<thead>
<tr>
<th>Product</th>
<th>Year</th>
<th>Qtr</th>
<th>Sales</th>
<th>Quantity</th>
<th>Unit Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colombiano</td>
<td>1994</td>
<td>Q1</td>
<td>$16048.00</td>
<td>2726</td>
<td>5.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q2</td>
<td>$15498.00</td>
<td>2628</td>
<td>5.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q3</td>
<td>$13278.50</td>
<td>2260</td>
<td>5.87</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$57215.00</td>
<td>9790</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1995</td>
<td>Q1</td>
<td>$16006.50</td>
<td>2752</td>
<td>5.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q2</td>
<td>$17382.00</td>
<td>2882</td>
<td>6.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q3</td>
<td>$15299.50</td>
<td>2624</td>
<td>5.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q4</td>
<td>$17457.00</td>
<td>2702</td>
<td>6.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$66145.00</td>
<td>10960</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$123360.00</td>
<td>20750</td>
<td></td>
</tr>
</tbody>
</table>

### 1994-1995 Sales in S. J. by Brand 10/28/97 Pg: 3

<table>
<thead>
<tr>
<th>Product</th>
<th>Year</th>
<th>Qtr</th>
<th>Sales</th>
<th>Quantity</th>
<th>Unit Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lotta Latte</td>
<td>1994</td>
<td>Q1</td>
<td>$24633.00</td>
<td>3258</td>
<td>7.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q2</td>
<td>$17433.00</td>
<td>2266</td>
<td>7.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q3</td>
<td>$17842.00</td>
<td>2482</td>
<td>7.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q4</td>
<td>$18047.00</td>
<td>2528</td>
<td>7.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$77955.00</td>
<td>10534</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1995</td>
<td>Q1</td>
<td>$18753.00</td>
<td>2650</td>
<td>7.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q2</td>
<td>$24777.00</td>
<td>3310</td>
<td>7.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q3</td>
<td>$17091.00</td>
<td>2306</td>
<td>7.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Q4</td>
<td>$14935.00</td>
<td>2096</td>
<td>7.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$75556.00</td>
<td>10362</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$153511.00</td>
<td>20896</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$400560.00</td>
<td>61790</td>
<td></td>
</tr>
</tbody>
</table>
Formatting Reports
Sample Report 2
This chapter contains reference pages in alphabetical order that describe RISQL Entry Tool and RISQL Reporter commands. These commands are not case-sensitive.

Many of the commands documented in this chapter are available only with RISQL Reporter, and some commands have options that apply only to RISQL Reporter. These restrictions are noted in the right column of the following table and at the top of each applicable reference page.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>RISQL Reporter Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Starts a system shell session or executes a single shell command (shell escape).</td>
<td></td>
</tr>
<tr>
<td>CONNECT</td>
<td>Connects to a warehouse database with the current username, an alternative username, or a DSN.</td>
<td></td>
</tr>
<tr>
<td>EDIT</td>
<td>Edits the current command buffer or a specified file.</td>
<td></td>
</tr>
<tr>
<td>QUERY</td>
<td>Displays current RISQL Entry Tool and RISQL Reporter settings.</td>
<td>✓ (some options)</td>
</tr>
<tr>
<td>QUERY COLUMN</td>
<td>Displays RISQL Reporter settings for the SET COLUMN command and options.</td>
<td>✓</td>
</tr>
<tr>
<td>QUIT EXIT</td>
<td>Ends the current tool session.</td>
<td></td>
</tr>
</tbody>
</table>
## Command Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>RISQL Reporter Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>Restores current RISQL Entry Tool and RISQL Reporter settings to default values.</td>
<td>✓  (some options)</td>
</tr>
<tr>
<td>RESET COLUMN</td>
<td>Removes column display options.</td>
<td>✓</td>
</tr>
<tr>
<td>RUN</td>
<td>Executes the current command buffer or a specified file.</td>
<td></td>
</tr>
<tr>
<td>SAVE</td>
<td>Saves all current RISQL Reporter settings to a specified file.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN</td>
<td>Sets display options for a result column.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column FILL</td>
<td>Sets the left-fill character for data in a column.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column FOLD LINE</td>
<td>Breaks a query row into two or more print lines.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column FORMAT</td>
<td>Sets the format of numeric data (DECIMAL, INTEGER, EXPONENTIAL).</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column IF NULL</td>
<td>Defines string that is displayed when nulls are returned.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column JUSTIFY</td>
<td>Sets the justification of data in a column.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column NO PRINT</td>
<td>Prevents a column from being displayed in the result.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column NO REPEAT</td>
<td>Suppresses the display of duplicate values in the result column.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column POSITIVE SIGN</td>
<td>Controls the display of the plus sign (+) for integer and decimal data.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column SIGN</td>
<td>Controls the placement of the displayed signs (+ or −) for integer and decimal data.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column SKIP LINE</td>
<td>Skips a line or starts a new page when a value in a column changes.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column SPACE</td>
<td>Sets the lead spacing before a column.</td>
<td>✓</td>
</tr>
</tbody>
</table>
### Command Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>RISQL Reporter Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET COLUMN column TITLE</td>
<td>Sets a column title.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column UNDERLINE</td>
<td>Displays a string of repeating characters underneath the column title.</td>
<td>✓</td>
</tr>
<tr>
<td>SET COLUMN column WIDTH</td>
<td>Sets the width of each column.</td>
<td>✓</td>
</tr>
<tr>
<td>SET ECHO</td>
<td>Turns command echoing on or off.</td>
<td></td>
</tr>
<tr>
<td>SET EDITOR</td>
<td>Specifies the system editor invoked by the EDIT command.</td>
<td></td>
</tr>
<tr>
<td>SET ERROR OUTPUT</td>
<td>Sends error messages to a file or discards them.</td>
<td></td>
</tr>
<tr>
<td>SET OUTPUT</td>
<td>Sends output to a file or discards it.</td>
<td></td>
</tr>
<tr>
<td>SET PAGE LENGTH</td>
<td>Sets the number of lines on each report page.</td>
<td>✓</td>
</tr>
<tr>
<td>SET PROMPT</td>
<td>Defines the prompt strings displayed by the RISQL Entry Tool or RISQL Reporter</td>
<td></td>
</tr>
<tr>
<td>SET ROW DATA</td>
<td>Suppresses output of column headings and any other non-data row information in result set.</td>
<td></td>
</tr>
<tr>
<td>SET TITLE</td>
<td>Creates a report title.</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Note:** All user-defined object names (except filenames and passwords) specified in RISQL Entry Tool or RISQL Reporter commands can contain single-byte or multibyte characters.
Suspends the tool session and starts a system shell session or executes a single shell command and reactivates the tool session.

\[ \text{os\_shell} \]

Shell startup command that starts an interactive shell session. Any valid system shell, for example, \textit{ksh}, \textit{sh}, or \textit{csh}, is acceptable.

\[ \text{os\_command} \]

Single system command to be executed.

**Interactive Shell Session**

When a shell process is initialized, your RISQL Entry Tool or RISQL Reporter session is suspended and you can execute one or more shell commands. Unlike RISQL Entry Tool or RISQL Reporter commands, system commands executed from the shell are always case-sensitive. The \textit{exit} command ends the shell session and reactivates the tool session.

Always remember to terminate the interactive shell session and return to your RISQL Entry Tool or RISQL Reporter session. Otherwise, you might inadvertently leave yourself connected to a database, which might interfere with the work of other users.

**Single Shell-Command Execution**

When used with a system command, the ! command suspends the current tool session, starts a new shell process, executes the specified command, and returns to the tool prompt. You can issue only one shell command at a time with this method and the operating-system command controls the output.

**Example**

RISQL>!ls -l *.risql
"-rw-rw-rw- 1 curly pubs 13 Oct 16 13:23 cume.risql"
"-rw-rw-rw- 1 curly pubs 13 Oct 16 13:24 mvgavg.risql"
"-rw-rw-rw- 1 curly pubs 13 Oct 16 13:23 ratio.risql"
"-rw-rw-rw- 1 curly pubs 13 Oct 16 13:24 share.risql"
RISQL>
CONNECT

Connects to a warehouse database with the current username or an alternative username.

```
CONNECT
  [TO database_name] [dsn_name]
  [AS db_username] [db_password]
```

database_name
Logical database name or full pathname of the file directory where the database name is located.

dsn_name
Data source name (DSN) defined in the .odbc.ini file (UNIX system), the odbc.ini file (Windows 95), or the Registry (Windows NT).

db_username
Valid database username.

db_password
Password associated with the specified database username.

TO, AS
The TO and AS keywords are optional, but if you omit them and provide only one argument, the tool assumes that the argument is -d database_name. If -d database_name is a valid warehouse database name, the tool prompts for db_username and db_password.

Database Connection

When you issue the CONNECT command during a session, any current database connection is automatically disconnected. If the new connection fails, the tool session remains active, but your previous database connection is lost. You cannot access a database until you make a valid connection. You can connect to the same database under the new db_username by using the CONNECT AS command.
You can connect to a database on the local server with a DSN or the CONNECT command. However, to connect to a database on a remote server, you must use a DSN.

**Prompting for Username and Password**

When input comes from a terminal:

- If you omit your username, the tool prompts for `db_username` and `db_password`.
- If you omit your password, the tool prompts for `db_password` alone.

When you redirect input from a file, you must provide your `db_username` and `db_password` in the file.

**Examples**

```plaintext
RISQL> connect to aroma as curly ;
Please enter password: xxxxxxxx
RISQL>

RISQL> connect to aroma_DSN;
Please enter password: xxxxxxxx
RISQL>
```

**See Also**

Appendix A, “risql and risqlrpt Command Reference.”
**EDIT**

Edits the current command buffer or a specified file.

```
EDIT filename ;
```

*filename*

Name of a file that contains one or more SQL statements or RISQL Entry Tool or RISQL Reporter commands. This filename can be a full or relative pathname.

**Editing a File**

Issue the `EDIT` command with a `filename` to start an editor and open `filename`. After you terminate the editing session, you can execute the contents of `filename` with the `RUN` command. The default editor on UNIX platforms is `vi` and the default editor on Windows platforms is Notepad.

**Editing the Buffer**

When you omit `filename`, the contents of the command buffer are placed in a temporary file for editing. The command buffer contains the most recently executed SQL statement or tool command, with the exception of the `EDIT` and `RUN` commands. `EDIT` and `RUN` declarations cannot be edited in the command buffer. You can save any changes you make to the temporary file by writing them to the original file with an editor command. Otherwise, any changes are made in the temporary copy only and are not retained.

**Displaying the Current Editor**

`QUERY EDITOR` displays the current editor.
Example

RISQL> **edit store_rank** ;

```sql
select store_name, sum(dollars) as total_sales,
    rank(sum(dollars)) as sales_rank
from market m join store r on m.mktkey = r.mktkey
    join sales s on s.storekey = r.storekey
    join period p on s.perkey = p.perkey
where year = 1995
    and month = 'MAR'
    and region = 'West'
group by store_name;
```

See Also

RUN
SET EDITOR
SET OUTPUT
QUERY
(Some options RISQL Reporter only)

Displays settings for various RISQL Entry Tool and RISQL Reporter commands.

QUERY ;

QUERY

ECHO

EDITOR

PAGE LENGTH

PROMPT

ROW DATA

TITLE

ECHO
Displays status of the SET ECHO command for the current session.

EDITOR
Displays status of the SET EDITOR command for the current session.

PAGE LENGTH
(RISQL Reporter only)
Displays status of the SET PAGE LENGTH command for the current session.

PROMPT
Displays status of the SET PROMPT command for the current session.

ROW DATA
Displays status of the SET ROW DATA command for the current session.

TITLE
(RISQL Reporter only)
Displays status of the SET TITLE command for the current session.

Example

RISQL> query echo ;
SET ECHO ON ;
Command Reference

QUERY

See Also

SET ECHO
SET EDITOR
SET PAGE LENGTH
SET PROMPT
SET ROW DATA
SET TITLE
QUERY COLUMN
(RISQL Reporter only)

Displays RISQL Reporter settings for the SET COLUMN command and its display options.

Example

RISQL> query column dollars ;
COLUMN DOLLARS WIDTH 8
COLUMN DOLLARS FORMAT DECIMAL

See Also

RESET COLUMN
SET COLUMN
QUIT or EXIT

Closes the current tool session and returns to the system prompt.

QUIT
EXIT

EXIT
EXIT can be used as an alternative to QUIT.

Example

RISQL> quit ;
$

See Also

None.
RESET
(Some options RISQL Reporter only)

Restores settings for various RISQL Entry Tool and RISQL Reporter commands to default values.

<table>
<thead>
<tr>
<th>SET ECHO</th>
<th>SET EDITOR</th>
<th>SET ERROR OUTPUT</th>
<th>SET OUTPUT</th>
<th>SET PAGE LENGTH</th>
<th>SET PROMPT</th>
<th>SET ROW DATA</th>
<th>SET TITLE</th>
</tr>
</thead>
</table>

ECHO
Restores SET ECHO to its default value.

EDITOR
Restores SET EDITOR to its default value.

ERROR OUTPUT
Restores SET ERROR OUTPUT to its default value.

OUTPUT
Restores SET OUTPUT and SET OUTPUT PROGRAM to their default values.

PAGE LENGTH
(RISQL Reporter only)
Restores SET PAGE LENGTH to its default value.

PROMPT
Restores SET PROMPT to its default value.

ROW DATA
Restores SET ROW DATA command to its default value.
Command Reference

RESET

TITLE
(RISQL Reporter only)
Restores SET TITLE to its default value.

Example

RISQL> reset editor ;

See Also

SET ECHO
SET EDITOR
SET ERROR OUTPUT
SET OUTPUT
SET PROMPT
SET TITLE
SET ROW DATA
SET PAGE LENGTH
RESET COLUMN

*(RISQL Reporter only)*

Removes column display settings and restores options to default values.

```
RESET COLUMN * \( n \) \( \text{table.*} \) \( \text{column_name} \) \( \text{table.column_name} \) \( \text{FILL} \) \( \text{FOLD LINE} \) \( \text{IF NULL} \) \( \text{JUSTIFY} \) \( \text{NO PRINT} \) \( \text{NO REPEAT} \) \( \text{FORMAT} \) \( \text{POSITIVE SIGN} \) \( \text{SIGN} \) \( \text{SKIP LINE} \) \( \text{SKIP PAGE} \) \( \text{SPACE} \) \( \text{TITLE} \) \( \text{UNDERLINE} \) \( \text{WIDTH} \) \( \text{ALL} \)
```

**Column Declaration**

The following table describes valid column variables:

<table>
<thead>
<tr>
<th>Column Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Global column list</td>
</tr>
<tr>
<td>( n )</td>
<td>Relative column number (3)</td>
</tr>
<tr>
<td>( \text{table.*} )</td>
<td>Table column list (sales.*)</td>
</tr>
<tr>
<td>( \text{column_name} )</td>
<td>Unqualified column name (dollars)</td>
</tr>
<tr>
<td>( \text{table.column_name} )</td>
<td>Table-qualified column name (sales.dollars)</td>
</tr>
</tbody>
</table>

When you reset an option for a single column, options on other columns in the result are left intact. When you reset an option on a column list, only the specified option that was set with that particular column-list variable is reset. Resetting global options resets only those options that were set with the global column wildcard character (*), not all column settings.
Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the SQL Reference Guide.

**Option Declaration**

The following table briefly describes each valid column display option. Format options are described in detail on individual SET COLUMN reference pages.

<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILL</td>
<td>Sets the left-fill character for data in a column.</td>
</tr>
<tr>
<td>FOLD LINE</td>
<td>Breaks a data row into two or more lines.</td>
</tr>
<tr>
<td>IF NULL</td>
<td>Defines the string that is displayed when null is returned.</td>
</tr>
<tr>
<td>JUSTIFY</td>
<td>Sets the justification of data in a column.</td>
</tr>
<tr>
<td>NO PRINT</td>
<td>Prevents a column from being displayed in the result.</td>
</tr>
<tr>
<td>NO REPEAT</td>
<td>Suppresses the display of duplicate values in a column.</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Sets the format of numeric data in a column (DECIMAL, INTEGER, EXPONENTIAL)</td>
</tr>
<tr>
<td>POSITIVE SIGN</td>
<td>Controls the display of the plus sign (+) for integer and decimal data.</td>
</tr>
<tr>
<td>SIGN</td>
<td>Controls the placement of the plus and minus signs (+ and –) for integer and decimal data.</td>
</tr>
<tr>
<td>SKIP LINE</td>
<td>Inserts a blank row between data rows when a value in a column changes.</td>
</tr>
<tr>
<td>SKIP PAGE</td>
<td>Starts a new page when a value in a column changes.</td>
</tr>
<tr>
<td>SPACE</td>
<td>Sets lead spacing for data in a column.</td>
</tr>
<tr>
<td>TITLE</td>
<td>Defines a column title.</td>
</tr>
<tr>
<td>UNDERLINE</td>
<td>Displays a string of repeating characters between column headings and data.</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Sets the width of a column.</td>
</tr>
<tr>
<td>ALL</td>
<td>Resets options for all columns to their default values.</td>
</tr>
</tbody>
</table>
**Examples**

To reset an option, you must use the same type of column variable and option argument that you used when setting the option. For example, the following command sets the width of the perkey column in the sales table:

```
RISQL> set column 1 width 1;
```

To reset the default column width for the quantity column, enter:

```
RISQL> reset column 1 width;
```

However, it does not reset the column width if it was set with the following command:

```
RISQL> set column perkey width 1;
```

The following command restores the JUSTIFY option for the sales.dollars column to its default value (RIGHT, in this case):

```
RISQL> reset column sales.dollars justify;
```

The following RESET command restores all display options set with the sales.* column variable to their default values:

```
RISQL> reset column sales.* all;
```

**See Also**

QUERY COLUMN

SET COLUMN
**RUN**

Executes the current command buffer or a specified file.

```
RUN [filename] ;
```

**filename**
Name of a file that contains one or more SQL statements or RISQL Entry Tool or RISQL Reporter commands. This filename can be a full or relative pathname.

When you specify *filename*, the RUN command reads and executes the commands in the file.

When you omit *filename*, the tool reads the temporary buffer in the following order of precedence:

- If the most recent tool command was EDIT, issuing the RUN command without *filename* executes the previously edited file.
- If the most recent tool command was not EDIT, issuing the RUN command without *filename* executes the command buffer, which contains the last executed command.
Example

RISQL> run store_rank ;
RISQL> select store_name, sum(dollars) as total_sales,
>       rank(sum(dollars)) as sales_rank
> from market m join store r on m.mktkey = r.mktkey
>       join sales s on s.storekey = r.storekey
>       join period p on s.perkey = p.perkey
> where year = 1995
>       and month = 'MAR'
>       and region = 'West'
> group by store_name;

<table>
<thead>
<tr>
<th>STORE_NAME</th>
<th>TOTAL_SALES</th>
<th>SALES_RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cupertino Coffee Supply</td>
<td>18709.50</td>
<td>1</td>
</tr>
<tr>
<td>San Jose Roasting Company</td>
<td>17867.90</td>
<td>2</td>
</tr>
<tr>
<td>Java Judy's</td>
<td>17826.25</td>
<td>3</td>
</tr>
<tr>
<td>Beaches Brew</td>
<td>17197.25</td>
<td>4</td>
</tr>
<tr>
<td>Roasters, Los Gatos</td>
<td>15159.05</td>
<td>5</td>
</tr>
<tr>
<td>Instant Coffee</td>
<td>14841.75</td>
<td>6</td>
</tr>
</tbody>
</table>

Note: SET ECHO ON is in effect in this example.

See Also

EDIT
SET ECHO
SAVE

(RISQL Reporter only)

Saves current RISQL Reporter format settings in a specified file.

```
SAVE filename ;
```

**filename**
Name of a file to which RISQL Reporter format settings for the current session are written. The file is overwritten each time the SAVE command is executed. The file contains standard ASCII text and can be modified with an editor.

**Valid Commands**

The following commands and options can be saved in a file with the SAVE command:

- SET PAGE LENGTH
- SET TITLE
- SET COLUMN (all options)

**Changing Default Settings**

You can use the RUN command to execute saved format settings in a file, which lets you develop reusable templates for commonly used report formats. You can also copy the contents of a saved file to the RISQL Reporter initialization files (.rbretrc) to set customized default settings for all subsequent RISQL Reporter sessions.

**Example**

```
RISQL> save set_commands ;
```

**See Also**

RUN
Appendix B, ".rbretrc Files"
**SET COLUMN**  
*(RISQL Reporter only)*

Sets column display options.

```
<table>
<thead>
<tr>
<th>Column Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Global column list</td>
</tr>
<tr>
<td>n</td>
<td>Relative column number</td>
</tr>
<tr>
<td>table.*</td>
<td>Table column list (sales.*)</td>
</tr>
<tr>
<td>column_name</td>
<td>Unqualified column name (dollars)</td>
</tr>
<tr>
<td>table.column_name</td>
<td>Table-qualified column name (sales.dollars)</td>
</tr>
</tbody>
</table>
```

**Column Variable Precedence**

The following list describes each valid column specification and gives its level of precedence when more than one option is specified:

- Options specified for the global column list apply to all columns retrieved unless overwritten by a more specific column option specified later in the command syntax.
- Options specified for a relative column number are applied to the column in that position in the query. Relative column number options modify the characteristics of columns that are literals, subqueries, expressions, or display functions. Relative column numbers are retained even when columns are suppressed with the NO PRINT option of the SET COLUMN command. Relative column number options override global options.
- Options specified for a table column list are applied to all columns in that table. Table column list options override global and relative column number options.
Options specified for unqualified column names are applied to all columns with that name, regardless of the table to which the column belongs. Column aliases defined in a SELECT statement can also be used as unqualified column names. Unqualified column options override global column lists, table column lists, and relative column number options.

**Note:** Correlation names are not accepted as column identifiers. For more information about correlation names, refer to the *SQL Reference Guide*.

Options specified for table-qualified column names are applied to only the specific column. Column aliases defined in a SELECT statement cannot be used in table-qualified names. Table-qualified column name options take precedence over any other option specified for the column.

### Option Declaration

The following table briefly describes each valid column display option:

<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILL</td>
<td>Sets the left-fill character for data in a column.</td>
</tr>
<tr>
<td>FOLD LINE</td>
<td>Breaks a data row into two or more lines.</td>
</tr>
<tr>
<td>IF NULL</td>
<td>Defines the string that is displayed when null is returned.</td>
</tr>
<tr>
<td>JUSTIFY</td>
<td>Sets the justification of data in a column.</td>
</tr>
<tr>
<td>NO PRINT</td>
<td>Prevents a column from being displayed in the result.</td>
</tr>
<tr>
<td>NO REPEAT</td>
<td>Suppresses the display of duplicate values in a column.</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Sets the format of numeric data in a column (DECIMAL, INTEGER, EXPONENTIAL).</td>
</tr>
<tr>
<td>POSITIVE SIGN</td>
<td>Controls the display of the plus sign (+) for integer and decimal data.</td>
</tr>
<tr>
<td>SIGN</td>
<td>Controls the placement of the plus and minus signs (+ and −) for integer and decimal data.</td>
</tr>
<tr>
<td>SKIP LINE</td>
<td>Inserts a blank row between data rows when a value in a column changes.</td>
</tr>
<tr>
<td>SKIP PAGE</td>
<td>Starts a new page when a value in a column changes.</td>
</tr>
<tr>
<td>SPACE</td>
<td>Sets lead spacing for data in a column.</td>
</tr>
<tr>
<td>TITLE</td>
<td>Defines a column title.</td>
</tr>
</tbody>
</table>
### Format Option | Description
--- | ---
UNDERLINE | Displays a string of repeating characters between column headings and data.
WIDTH | Sets the width of a column.

SET COLUMN display options are described in detail on the reference pages listed under “See Also” below.

**Displaying Current Settings**

Use the QUERY COLUMN command to display current column settings.

**Restoring Defaults**

Use the RESET COLUMN command to return column settings to their default values.

**See Also**

QUERY COLUMN
RESET COLUMN
SET COLUMN column FILL
SET COLUMN column FOLD LINE
SET COLUMN column IF NULL
SET COLUMN column JUSTIFY
SET COLUMN column NO PRINT
SET COLUMN column NO REPEAT
SET COLUMN column FORMAT
SET COLUMN column POSITIVE SIGN
SET COLUMN column SIGN
SET COLUMN column SKIP
SET COLUMN column SPACE
SET COLUMN column TITLE
SET COLUMN column UNDERLINE
SET COLUMN column WIDTH
**SET COLUMN column FILL**  
*(RISQL Reporter only)*

Defines a left-fill character for a result column.

```
  SET COLUMN -- * -- FILL -- 'fill_char' -- ;
```

*fill_char*

One single-byte character that is used to left-fill a column field. This character is a space character by default and must be enclosed in single quotation marks ('). Multibyte characters cannot be used for this argument.

**Using FILL with the Justify Option**

When the FILL option is in use, the JUSTIFY LEFT option has no effect on data in the column because every character space in a column field is filled.

**Using FILL with Negative Numeric Data**

The RISQL Reporter makes a special provision for negative numbers when you format integer or decimal data with leading zero characters. In this case, the leading minus sign (−) indicating a negative number is placed in the leftmost character position. When you fill with any other character, the minus sign displays immediately to the left of the leftmost numeric data character.

**Using FILL with Positive Numeric Data**

If you format integer or decimal data with leading zero characters and display the plus sign, the leading plus sign (+) indicating a positive number is placed in the leftmost character position. However, if you fill a column containing integer or decimal data with non-zero characters, the plus sign in a leading position is not displayed.

**Displaying Current Setting**

QUERY COLUMN column displays all RISQL Reporter options set on column.
**Command Reference**

**SET COLUMN column FILL**

### Restoring Default

RESET COLUMN *column* FILL returns the FILL option to its default state.

### Example

```
RISQL> set column emp_no fill '0' ;
RISQL> select emp_no from employees;
```

Data in the Emp_No column is displayed in the result as follows:

```
EMP_NO
00012
00036
00037
00052
00123
00320
```

### See Also

- QUERY COLUMN
- RESET COLUMN
- SET COLUMN
- SET COLUMN *column* JUSTIFY
- SET COLUMN *column* SIGN
**SET COLUMN column FOLD LINE**

*(RISQL Reporter only)*

Breaks a query row into two or more lines.

```
SET COLUMN column FOLD LINE
```

**\n Character**

The FOLD LINE option of the SET COLUMN command causes the RISQL Reporter to insert a new-line (“\n”) character before displaying the specified column. This option lets you break a long row into several rows, so the total width of the line does not exceed the width of your anticipated output device (such as a small terminal screen or a printer).

**Multiple Columns**

Multiple FOLD LINE columns are permitted.

**Stacking Data**

Placing FOLD LINE on every column (set column * fold line) stacks data, displaying a column-oriented report rather than a row-oriented report. Using the FOLD LINE option in this manner is particularly useful with the LABEL option. Including a SET ROW DATA ONLY statement prevents stacked column headings from being displayed.

**Displaying Current Setting**

QUERY COLUMN column displays all RISQL Reporter options set on column.

**Restoring Default**

RESET COLUMN column FOLD LINE returns the FOLD LINE option to its default state.
SET COLUMN column FOLD LINE

Example

RISQL> set column city fold line;
RISQL> select prod_name, city, quantity,
> cume(quantity) as running_total
> from market ma join store st
> on ma.mktkey = st.mktkey
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
>    or prod_name like 'Cafe%')
>    and city like 'New O%
>    and week = 32
>    and year = 1995
> order by prod_name, quantity asc;

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>RUNNING_TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>43</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>58</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>92</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>111</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>130</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>145</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>160</td>
</tr>
</tbody>
</table>

See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN column SPACE
SET ROW DATA ONLY
Command Reference

SET COLUMN column FORMAT

SET COLUMN column FORMAT
(RISQL Reporter only)

Changes the numeric format of data in a result column.

\[
\text{SET COLUMN } \quad \begin{array}{c|c|c|c}
\hline
\text{column} & \text{table} & \text{column name} \\
\hline
\text{n} & \text{FORMAT} & \text{DECIMAL} & \text{INTEGER} \\
\text{EXPONENTIAL} \\
\hline
\end{array}
\]

**DECIMAL** \( n \)

Displays all real, float, double-precision, decimal, and numeric types in typical decimal format. The integer \( n \) specifies the number of displayed decimal points. Data values that have more decimal places than specified by \( n \) are rounded off.

\[
123.45
\]

The default value for \( n \) is 2. The maximum value for \( n \) is 99.

**INTEGER**

Displays all numbers as integers; decimal values are rounded off and no decimal places are displayed.

\[
123
\]

Negative numbers are also rounded off; \(-1.2\) becomes \(-1\) and \(-1.5\) becomes \(-2\).

**EXPONENTIAL**

Displays all numbers in exponential format. The number of decimal places displayed is equal to the column width minus 8. If the column width is less than 9, asterisks are displayed.

\[
1.2345E+2
\]

**Default**

Data is displayed in its original datatype by default.
Displayed Asterisks

If the value exceeds the width or format limitation of the column, asterisks (*) are printed in the field. Asterisks occur most often in columns that are formatted as decimal or integer. However, if exponential format is specified and the column width is less than 9, asterisks are displayed.

Displaying Current Setting

QUERY COLUMN column displays all RISQL Reporter options set on column.

Restoring Default

RESET COLUMN column FORMAT returns the FORMAT option to its default state.

Example

The following example rounds off decimal values and display all numbers as integers in column number 3:

```
RISQL> set column 3 format integer;
RISQL> select month, sum(dollars) as dols,
>       cume(sum(dollars)) as run_dollars,
>       sum(quantity) as qty,
>       cume(sum(quantity)) as run_quantity
> from market ma join store st
> on ma.mktkey = st.mktkey
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
> where year = 1995
> and qtr = 'Q2_95'
> and prod_name = 'Colombiano'
> and city = 'Los Angeles'
> group by month;
```

<table>
<thead>
<tr>
<th>MONTH</th>
<th>DOLS</th>
<th>RUN_DOLLARS</th>
<th>QTY</th>
<th>RUN_QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUN</td>
<td>4011.50</td>
<td>4011</td>
<td>660</td>
<td>660</td>
</tr>
<tr>
<td>MAY</td>
<td>2853.50</td>
<td>6865</td>
<td>482</td>
<td>1142</td>
</tr>
<tr>
<td>APR</td>
<td>2736.50</td>
<td>9601</td>
<td>452</td>
<td>1594</td>
</tr>
</tbody>
</table>

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN column WIDTH
SET COLUMN column IF NULL

(RISQL Reporter only)

Defines a string that is displayed when nulls are returned from the database.

```
SET COLUMN column IF NULL 'null_string' ;
```

**null_string**

String to be displayed when nulls are returned. The string can consist of single-byte or multibyte characters. The null_string must be enclosed in single quotation marks ('). The default value is NULL. To display blanks in place of nulls, use an empty null_string (' ').

**Displaying Current Setting**

QUERY COLUMN column displays all RISQL Reporter options set on column.

**Restoring Default**

RESET COLUMN column IF NULL returns the IF NULL option to its default setting.

**Example**

```
RISQL> set column mgrno if null 'N/A' ;
RISQL> select name, department, title, mgrno, empno
   > from dept natural join employees
   > where department like 'Sales%';
NAME       DEPARTMENT  TITLE      MGRNO  EMPNO
Blake, Joe  Sales       AcctRep   N/A    1874
Peach, Etta Sales       AcctRep   N/A    1051
Slo, Monica Sales       Mgr      089    0498
Tepio, Curtis Sales     AcctRep   N/A    1179
... 
```

**See Also**

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN column JUSTIFY

(RISQL Reporter only)

Justifies data, either left or right, in a result column.

LEFT
Left-justifies data in a column.

Note: When the FILL option is in use, the JUSTIFY LEFT option has no effect on data in the column because every character space in a column field is filled.

RIGHT
Right-justifies data in a column.

Defaults
Default justification is determined by datatype; numeric data is right-aligned and character data is left-aligned. Use the RESET COLUMN statement to reset options to default values.

Displaying Current Setting
QUERY COLUMN column displays all RISQL Reporter options set on column.

Restoring Default
RESET COLUMN column JUSTIFY returns the JUSTIFY option to its default state.
Example

In the following example, the employee number column (emp_no) is of character datatype, and is therefore left-justified by default. You can right-justify employee numbers with the following command:

```
RISQL> set column emp_no justify right
RISQL> select emp_no from employees;

EMP_NO
  12
  36
  37
  52
  123
  320
```

See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
**SET COLUMN column NO PRINT**

*(RISQL Reporter only)*

Suppresses the display of a result column.

```
SET COLUMN column NO PRINT
```

### Displaying Current Setting

`QUERY COLUMN column` displays all RISQL Reporter options set on `column`.

### Restoring Default

`RESET COLUMN column NO PRINT` returns the NO PRINT option to its default state.

### Example

The results of the following query apply only to stores of type Large, but the Store_Type column of data (as well as the State column) is hidden in the report:

```
RISQL> set column state no print;
RISQL> set column store_type no print;
RISQL> select store_name, store_type, state, 
>      sum(dollars) as sales
> from store natural join sales 
> where store_type = 'Large'
> group by store_name, store_type, state;
```

<table>
<thead>
<tr>
<th>STORE_NAME</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beaches Brew</td>
<td>459248.50</td>
</tr>
<tr>
<td>San Jose Roasting Company</td>
<td>659385.35</td>
</tr>
<tr>
<td>Olympic Coffee Company</td>
<td>451983.05</td>
</tr>
<tr>
<td>Miami Espresso</td>
<td>465057.70</td>
</tr>
</tbody>
</table>

### See Also

- `QUERY COLUMN`  
- `RESET COLUMN`  
- `SET COLUMN`
Command Reference

SET COLUMN column NO REPEAT

SET COLUMN column NO REPEAT
(RISQL Reporter only)

Suppresses the display of duplicate values in a result column.

Displaying Data Hierarchy

When you specify NO REPEAT on a column, the RISQL Reporter displays duplicate data values as blanks in the result. This option is useful for reports that order data by a column containing non-unique values.

Multiple Columns

When you specify multiple NO REPEAT columns, the RISQL Reporter compares values for each NO REPEAT column starting with the leftmost column. The first column that changes value ends further checking for repeated values in remaining columns to the right in that row.

New Page

If a page break occurs, hidden values are displayed at the top of the new page. Subsequent repeating values are suppressed until another page break occurs.

Displaying Current Setting

QUERY COLUMN column displays all RISQL Reporter options set on column.

Restoring Defaults

RESET COLUMN column NO REPEAT returns the NO REPEAT option to its default state.
Example

RISQL> set column state no repeat;
RISQL> set column store_type no repeat;
RISQL> select store_name, state, store_type,
    > sum(dollars) as sales
    > from store natural join sales
    > where store_type = 'Large'
    > group by store_name, store_type, state;

<table>
<thead>
<tr>
<th>STORE_NAME</th>
<th>STATE</th>
<th>STORE_TYPE</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beaches Brew</td>
<td>CA</td>
<td>Large</td>
<td>459248.50</td>
</tr>
<tr>
<td>San Jose Roasting Co.</td>
<td>GA</td>
<td>Large</td>
<td>659385.35</td>
</tr>
<tr>
<td>Olympic Coffee Co.</td>
<td>GA</td>
<td>Large</td>
<td>451983.05</td>
</tr>
<tr>
<td>Miami Espresso</td>
<td>FL</td>
<td>Large</td>
<td>465057.70</td>
</tr>
</tbody>
</table>

See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN column POSITIVE SIGN

Specifies whether to display or hide plus signs (+) for positive numbers.

<table>
<thead>
<tr>
<th>SET COLUMN</th>
<th>*</th>
<th>HIDE</th>
<th>POSITIVE SIGN</th>
<th>SHOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>table.*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>column_name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>table.column_name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HIDE
Hides the plus sign for integer or decimal data. HIDE is the default setting.

SHOW
Displays the plus sign for integer or decimal data.

Using with SET COLUMN column FILL

The RISQL Reporter makes a special provision for positive numbers when you format integer or decimal data with leading zero characters and show the positive sign. In this case, the leading plus sign (+) indicating a positive number is placed in the leftmost character position.

If you fill a column containing integer or decimal data with non-zero characters, leading plus signs are not displayed, regardless of the POSITIVE SIGN option. Trailing plus signs, however, are displayed when the POSITIVE SIGN option is set to SHOW.

Using with SET COLUMN column SIGN

The SET COLUMN column POSITIVE SIGN command can be used with the SET COLUMN column SIGN TRAILING command to display the plus sign (+) at the rightmost character position for the columns.

Displaying Current Setting

QUERY COLUMN column displays all RISQL Reporter options set on column.
**Command Reference**

SET COLUMN column POSITIVE SIGN

**Restoring Default**

RESET COLUMN column POSITIVE SIGN returns the POSITIVE SIGN option to its default state, HIDE.

**Example**

To display the plus sign, specify the SHOW keyword in the SET COLUMN command, as follows:

```sql
RISQL> set column dollars show positive sign;
RISQL> select dollars from sales_new;

DOLLARS
-132.13
+566.00
-212.98
```

**See Also**

SET COLUMN column SIGN
Command Reference

SET COLUMN column SIGN

**SET COLUMN column SIGN**

*(RISQL Reporter only)*

Controls the placement of the signs (+ and –) for integer and decimal data.

```
> SET COLUMN * SIGN TRAILING LEADING

n

table.*

-column_name-

-table.column_name
```

**LEADING**

Displays the sign to the left of data values in named column. LEADING is the default setting.

**TRAILING**

Displays the sign to the right of data values in named column, reserving the rightmost character position in a column for the sign. If plus signs are not displayed, a space character occupies the rightmost space. The sign or space character reduces by one the number of characters in a numeric result.

**Using with SET COLUMN FILL**

If you format integer or decimal data with leading zero characters, leading plus and minus signs are placed in the leftmost character position. When you fill with any other character, leading minus signs display immediately to the left of the leftmost numeric data character; and leading plus signs are not displayed, regardless of the POSITIVE SIGN option.

**Using with SET COLUMN POSITIVE SIGN**

Plus signs (+) for positive numbers are displayed only if specified by the POSITIVE SIGN option.

**Displaying Current Setting**

QUERY COLUMN column displays all RISQL Reporter options set on column.

**Restoring Default**

RESET COLUMN column SIGN returns the SIGN option to its default state, LEADING.
**Example**

To move the minus sign to the right of column data, specify the TRAILING keyword in the SET COLUMN command:

```
RISQL> set column dollars sign trailing;
RISQL> select dollars from sales_new;
DOLLARS
132.13-
566.00
212.98-
```

**See Also**

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN column FILL
SET COLUMN column JUSTIFY
SET COLUMN column POSITIVE SIGN
SET COLUMN column SKIP

(RISQL Reporter only)

Skips a line or goes forward to a new page when a column value changes in the result.

```
SET COLUMN * SKIP LINE PAGE;
```

**LINE**
Skips a line when a value in a column changes.

**PAGE**
Starts a new page when a value in a column changes.

**Multiple Columns**

When you use the SKIP option on multiple columns, the RISQL Reporter compares values for each SKIP column starting with the leftmost column in the query. The first column encountered that changes value causes the RISQL Reporter to skip a line or begin a new page, as specified. Remaining SKIP columns in the row are not checked.

If both a SKIP LINE and a SKIP PAGE are used on the same column, SKIP PAGE takes precedence.

**Note:** In general, SKIP PAGE should be used on columns to the left of columns with SKIP LINE settings.

**Using with SET PAGE LENGTH**

The RISQL Reporter automatically starts a new page when the number of lines on a page, as specified with SET PAGE LENGTH, is exceeded. If a column value change occurs concurrently with a skip induced by the SET PAGE LENGTH option, only one page is skipped.

**Displaying Current Setting**

QUERY COLUMN column displays all RISQL Reporter options set on column.


**Restoring Default**

RESET COLUMN column SKIP returns the SKIP option to its default state.

**Examples**

```sql
RISQL> set column district skip line;
RISQL> select hq_city, district
> from market
> where region = 'South'
> or region = 'West'
> order by district ;

<table>
<thead>
<tr>
<th>HQ_CITY</th>
<th>DISTRICT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlanta</td>
<td>Atlanta</td>
</tr>
<tr>
<td>Miami</td>
<td></td>
</tr>
<tr>
<td>Phoenix</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>Los Angeles</td>
</tr>
<tr>
<td>New Orleans</td>
<td>New Orleans</td>
</tr>
<tr>
<td>Houston</td>
<td></td>
</tr>
<tr>
<td>San Jose</td>
<td>San Francisco</td>
</tr>
<tr>
<td>San Francisco</td>
<td>San Francisco</td>
</tr>
<tr>
<td>Oakland</td>
<td></td>
</tr>
</tbody>
</table>

The following command prints data for each district on a separate page:

```sql
RISQL> set column district skip page;
```

**See Also**

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET PAGE LENGTH
SET COLUMN column SPACE

(RISQL Reporter only)

Defines the lead spacing and data labels for result columns.

```
SET COLUMN * SPACE n 'label_string'
```

**SPACE n**
Number of spaces displayed to the left of a column.

**label_string**
Column label, which is displayed to the left of each column of the result. For example, you can set `label_string` to a locale-specific currency symbol. The `label_string` must be enclosed in single quotation marks ('). The string can consist of single-byte or multibyte characters. There is no default value.

**Changing Column Margins**

The `SPACE` option changes the space between columns and makes data columns easier to read. The default is one space for all columns except the leftmost column, which has no leading spaces. You can set spacing to zero to simulate concatenated columns.

**Column Width**

When you use the `SPACE` option with the `WIDTH` option of the `SET COLUMN` command, do not include the length of `label_string` in the width specification; the column expands automatically to include `label_string`.

**Wrapping Lines**

The `label_string` modifier is often used in conjunction with `SET COLUMN * FOLD LINE`, which displays each column value on a separate line, as illustrated on page 3-24.
**Command Reference**

*SET COLUMN column SPACE*

**Displaying Current Setting**

`QUERY COLUMN column` displays all RISQL Reporter options set on `column`.

**Restoring Default**

`RESET COLUMN column SPACE` returns the SPACE option to its default state.

**Examples**

To indent the result display, set the leading space on the leftmost column to 5 as follows:

```
RISQL> set column prod_name space 5;
RISQL> select prod_name, city, quantity, dollars
> from store st
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
> or prod_name like 'Cafe%')
> and city like 'New O%
> and week = 32
> and year = 1995
> order by prod_name, quantity asc;
```

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>DOLLARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14 112.00</td>
<td>112.00</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15 63.75</td>
<td>63.75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15 63.75</td>
<td>63.75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17 72.25</td>
<td>72.25</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19 161.50</td>
<td>161.50</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19 161.50</td>
<td>161.50</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15 90.00</td>
<td>90.00</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15 90.00</td>
<td>90.00</td>
</tr>
</tbody>
</table>
To precede each value in the Total column with a dollar sign ($), set the
\textit{label\_string} as follows:

```sql
RISQL> set column total width 9;
RISQL> set column total space 1 '$';
RISQL> select district,
       > sum(dollars) as total
       > from market ma join store st
       > on ma.mktkey = st.mktkey
       > join sales sa on st.storekey = sa.storekey
       > join product pr on pr.prodkey = sa.prodkey
       > join period pe on pe.perkey = sa.perkey
       > where prod\_name like 'Cafe%'
       > and year = 1995
       > group by district
       > order by total asc;
```

<table>
<thead>
<tr>
<th>DISTRICT</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston</td>
<td>$ 59623.00</td>
</tr>
<tr>
<td>Minneapolis</td>
<td>$ 62439.00</td>
</tr>
<tr>
<td>New Orleans</td>
<td>$ 62515.50</td>
</tr>
<tr>
<td>New York</td>
<td>$ 64072.00</td>
</tr>
<tr>
<td>Chicago</td>
<td>$ 66264.00</td>
</tr>
<tr>
<td>Atlanta</td>
<td>$ 69294.00</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>$ 71858.00</td>
</tr>
<tr>
<td>San Francisco</td>
<td>$141803.00</td>
</tr>
</tbody>
</table>

\textbf{Note:} A label is displayed at the left margin of a column by default. In this example, the column width is modified so the dollar signs are displayed close to the data values.

\textbf{See Also}

\texttt{QUERY COLUMN}
\texttt{RESET COLUMN}
\texttt{SET COLUMN}
\texttt{SET COLUMN column FOLD LINE}
\texttt{SET COLUMN column WIDTH}
**SET COLUMN column TITLE**  
*(RISQL Reporter only)*

Defines a column title and its position in the result.

```
<table>
<thead>
<tr>
<th>SET COLUMN column</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>n</td>
</tr>
<tr>
<td>table.*</td>
<td></td>
</tr>
<tr>
<td>column_name</td>
<td></td>
</tr>
<tr>
<td>table.column_name</td>
<td></td>
</tr>
</tbody>
</table>
```

**LEFT**
Left-justifies the title. This is the default value.

**RIGHT**
Right-justifies the title.

**title_string**
Column title, which is displayed above the specified column at the top of each page. The title_string must be enclosed in single quotation marks ('). The string can consist of single-byte or multibyte characters.

**Default Title String**

The default title_string is one of the following:

- Internal name of the column, as stored in the database
- Column alias defined in the SELECT statement
- Literal value of character literals
- Blanks for all other columns

**Macros**

You cannot use macros in column titles.

**Displaying Current Setting**

QUERY COLUMN column displays all RISQL Reporter options set on column.
Restoring Default

RESET COLUMN column TITLE returns the TITLE option to its default state.

Example

The following command right-justifies the existing title of the Percent column:

RISQL> set column percent title right;

The following command changes the title of the Cume (Quantity) column and right-justifies it:

RISQL> set column 4 title 'RUNNING TOTAL' right;
> select prod_name, city, quantity, cume(quantity)
> from store st
> join sales sa on st.storekey = sa.storekey
> join product pr on pr.prodkey = sa.prodkey
> join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
> or prod_name like 'Cafe%')
> and city like 'New O%'
> and week = 32
> and year = 1995
> order by prod_name, quantity asc;

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>RUNNING TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>43</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>58</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>92</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>111</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>130</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>145</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>160</td>
</tr>
</tbody>
</table>

See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN column UNDERLINE

(RISQL Reporter only)

Displays a horizontal line composed of a specified character string underneath column titles.

LONG
Displays an underscore (_) across the entire width of the column regardless of the length of the title string.

SHORT
Displays an underscore under the title text only.

u_char
One single-byte character that is used to underscore the column title. This character must be enclosed in single quotation marks ('). The default value is a hyphen (-). SHORT is the default when u_char is specified. Multibyte characters cannot be used for this argument.

Displaying Current Setting

QUERY COLUMN column displays all RISQL Reporter options set on column.

Restoring Default

RESET COLUMN column UNDERLINE returns the UNDERLINE option to its default state.
Command Reference
SET COLUMN column UNDERLINE

Example
RISQL> set column * underline '*' short;
RISQL> select prod_name, city, quantity,
>    cume(quantity) as running_total
> from store st
>    join sales sa on st.storekey = sa.storekey
>    join product pr on pr.prodkey = sa.prodkey
>    join period pe on pe.perkey = sa.perkey
> where (prod_name like 'Expresso%'
>    or prod_name like 'Cafe%')
>    and city like 'New O'
>    and week = 32
>    and year = 1995
> order by prod_name, quantity asc;

<table>
<thead>
<tr>
<th>PROD_NAME</th>
<th>CITY</th>
<th>QUANTITY</th>
<th>RUNNING TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>14</td>
<td>28</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>43</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>15</td>
<td>58</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>75</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>17</td>
<td>92</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>111</td>
</tr>
<tr>
<td>Cafe Au Lait</td>
<td>New Orleans</td>
<td>19</td>
<td>130</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>145</td>
</tr>
<tr>
<td>Expresso XO</td>
<td>New Orleans</td>
<td>15</td>
<td>160</td>
</tr>
</tbody>
</table>

See Also
QUERY COLUMN
RESET COLUMN
SET COLUMN
**SET COLUMN column WIDTH**

(RISQL Reporter only)

 Defines a display width for a result column.

![Command Syntax Diagram](image)

```
SET COLUMN  column_name  table.column_name  *  WIDTH  n  ;
```

**WIDTH n**

Display width of a column. The width is interpreted as spaces displayed on the screen, not as characters or bytes.

**Default Width**

The default width of a column field is the datatype width that was defined in the table definition statement. For example, if a column is defined as `character(10)`, the column is displayed as 10 spaces wide in the result. Titles are truncated when they are longer than the defined width.

**Default Justification**

By default, numeric data is right-justified and character data is left-justified.

**Displaying Current Setting**

`QUERY COLUMN column` displays all RISQL Reporter options set on `column`.

**Restoring Default**

`RESET COLUMN column WIDTH` returns the WIDTH option to its default state.

**Example**

The following command sets the width of the dollars column to 9, which provides two leading spaces before the expected 7-digit data.

```
RISQL> set column dollars width 9;
```
Command Reference

SET COLUMN column WIDTH

See Also

QUERY COLUMN
RESET COLUMN
SET COLUMN
SET COLUMN column JUSTIFY
SET COLUMN column TITLE
SET ECHO

Turns command echoing on and off.

<table>
<thead>
<tr>
<th>SET ECHO</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
</table>

**ON**
Displays all executed commands, such as a query statement, in standard output before results.

**OFF**
Suppresses the display of input from a file before output.

**Defaults**

Default settings differ depending on standard input and standard output settings. If both standard input and standard output are directed to the terminal, the default for SET ECHO is OFF. Otherwise, the default is ON. The following table summarizes the default state for command echoing:

<table>
<thead>
<tr>
<th>Input from Terminal</th>
<th>Input from File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output to Terminal</td>
<td>ECHO off</td>
</tr>
<tr>
<td>Output to File</td>
<td>ECHO on</td>
</tr>
</tbody>
</table>

**Displaying Current Setting**

The QUERY ECHO command displays the current status of the SET ECHO command.

**Restoring Default**

The RESET ECHO command returns the SET ECHO command to its default state.
**Example**

If you use the RUN command to execute the contents of the command buffer or a file and ECHO is set to ON, then the contents of the input file are displayed before the output. To submit a query stored in a file named *lotta_latte_LA_95*, enter:

```ql
RISQL> set echo on;
RISQL> run lotta_latte_LA_95;
RISQL> select count(*) as num_items,
     > string(avg(dollars), 7, 2) as avg_sale,
     > sum(dollars) as total_sales
     > from store st
     > join sales sa on st.storekey = sa.storekey
     > join product pr on pr.prodkey = sa.prodkey
     > join period pe on pe.perkey = sa.perkey
     > where prod_name = 'Lotta Latte'
     > and year = 1995
     > and city = 'Los Angeles';
```

```
<table>
<thead>
<tr>
<th>NUM_ITEMS</th>
<th>AVG_SALE</th>
<th>TOTAL_SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>360</td>
<td>106.69</td>
<td>38405.00</td>
</tr>
</tbody>
</table>
```

**See Also**

QUERY  
RESET  
RUN
SET EDITOR

Specifies the system editor, which is invoked by the EDIT command.

```
SET EDITOR 'edit_command' ;
```

**edit_command**

Command that starts an operating-system editor. This command must be enclosed in single quotation marks (') and can be a full or relative pathname or a name defined in your search path. The string can consist of single-byte or multibyte characters. The default system editor is *vi* for UNIX platforms, and Notepad for Windows platforms. You can invoke the editor with the EDIT command.

**Displaying Current Setting**

The QUERY EDITOR command displays the current status of the SET EDITOR command.

**Restoring Default**

The RESET EDITOR command returns the SET EDITOR command to its default state.

**Examples**

```
RISQL> query editor;
EDITOR 'vi'
RISQL> set editor 'ed';
RISQL>
RISQL> set editor 'ed -s';
RISQL> query editor;
EDITOR 'ed -s'
```

**See Also**

EDIT
QUERY
RESET
SET ERROR OUTPUT

Sends all error, warning, and informational messages to a designated file.

```
SET ERROR OUTPUT filename [APPEND | STDERR | STDOUT] ;
```

**filename**
Name of a file to which error output is sent. The `filename` can be a full or relative path name. If the specified file exists when the command is issued or the tool starts, its contents are overwritten.

**APPEND**
Appends output to the specified file.

**STDERR**
Sends all error, warning, and informational messages to standard error, except for fatal errors, which are always displayed in standard output.

**STDOUT**
Sends all error, warning, and informational messages to standard output. Depending on whether output redirection is in effect at the system level, standard output is either the terminal or another file. This is the default setting.

**Using SET ERROR OUTPUT with SET STATS**

The error output destination that you specify with the SET ERROR OUTPUT command is also the destination for any statistics output. You obtain statistics output by issuing the SET STATS ON or SET STATS INFO command, as described in the SQL Reference Guide.

**Restoring Default**

The RESET ERROR OUTPUT command resets message output to `stdout`, the default setting.

**Example**

```
RISQL> set error output error_messages;
```

**See Also**

RESET
SET OUTPUT

Sends query output to a designated file.

\[
\text{SET OUTPUT} \quad \text{filename} \quad ; \quad \begin{array}{c}
\text{APPEND} \\
\text{PROGRAM} \quad \text{program\_name} \\
\text{OFF} \\
\text{STDOUT}
\end{array}
\]

filename
Name of a file to which output is sent. The file can be a full or relative pathname. If the specified file exists when the command is issued or the tool starts, its contents are overwritten. Filenames must consist of single-byte characters.

APPEND
Appends output to the specified file.

PROGRAM
Specifies the name of a program that processes output displayed on a terminal. When the PROGRAM option is active, all output lines are sent to the program’s standard input.

program\_name
Name of output program. The program name can include parameters, and must be enclosed in single quotation marks ('). The program name can consist of single-byte or multibyte characters. Typical system output programs include \textit{more} and \textit{cat}. There is no default program\_name.

STDOUT
Directs all query output to standard output. Depending on whether output is redirected at the system level, standard output is either the terminal or a file. STDOUT is the default setting.

OFF
Discards all query output. When you discard output, query statistics are retained and can be displayed with the SET STATS command.
**Restoring Default**

The `RESET OUTPUT` command returns the `SET OUTPUT` command to its default state.

**Examples**

UNIX:

```
RISQL> set output /home/curly/test.output append;
RISQL> set output program 'more';
```

Windows:

```
RISQL> set output e: /home/curly/test.output;
```

**See Also**

`RESET`

`RUN`

`SET ECHO`

`SET ERROR OUPUT`
**SET PAGE LENGTH**  
*(RISQL Reporter only)*

Sets the number of lines printed on each report page.

```
> SET PAGE LENGTH  n  USING  FORM FEED  BLANK LINES
```

n
The desired number of lines printed on each page of a report.

**USING FORM FEED**

Inserts a form-feed character (ASCII decimal 12 or Control-L) when \( n \) lines is reached.

**USING BLANK LINES**

Fills remainder of page with blank lines when \( n \) lines is reached. This is the default setting.

**Report Heading Options**

The following lines can be displayed at the top of each report page:

- Report title followed by a blank line
- Column title
- Column underline

**Determining Page Length**

The RISQL Reporter prints lines of data until either a page break is indicated by a SKIP PAGE option or the maximum number of report lines, as defined by the SET PAGE LENGTH command, is reached.

The number of report lines displayed on a page is determined by subtracting the heading option lines (report title lines, the column title line, and the column underscore line) and the top and bottom margin allowance lines (4) from \( n \). Depending on the various SET options, some or all of these lines might not be displayed and, therefore, might not be needed in \( n \).
**SET PAGE LENGTH**

Setting \( n \) to 0 creates a page of infinite length or, in other words, one page. This is the default page length.

You must set \( n \) to at least one line greater than the total of the report title line, the column title line, and the column underscore line. Otherwise, the RISQL Reporter will automatically adjust \( n \) to zero (0).

**Using with SET COLUMN SKIP**

If the SET COLUMN SKIP option is specified when \( n \) is 0, page skipping is ignored.

When SET COLUMN SKIP is in use:

- If USING BLANK LINES is specified and a page skip is required, the tool completes a partial page by printing the number of blank lines required to fill the page. USING BLANK LINES is the default setting.
- If USING FORM FEED is specified and a page skip is required, a form-feed character is generated to skip to the next page. This character is an ASCII decimal 12 (equivalent to Control-L).

**Using with SET COLUMN FOLD LINE**

If the SET COLUMN column FOLD LINE option is used and results in multiple report lines per row, report lines are printed without breaking a row across two pages. If the number of report lines required for a single row is greater than the number of report lines available on a page, the RISQL Reporter automatically adjusts \( n \) to 0.

**Displaying Current Setting**

The QUERY PAGE LENGTH command displays the current status of the SET PAGE LENGTH command.

**Restoring Default**

The RESET PAGE LENGTH command returns the SET PAGE LENGTH command to its default state.

**Example**

```sql
RISQL> set page length 20 using form feed;
```
See Also

QUERY
RESET
SET COLUMN column SKIP PAGE
SET COLUMN column FOLD LINE
SET PROMPT

Defines the first-line, continuation, and incomplete string prompts displayed by the RISQL Entry Tool or RISQL Reporter.

\[
\text{SET PROMPT} \quad \leftarrow \quad \text{'prompt'} \quad \leftarrow \quad \text{'cont_promt'} \quad \leftarrow \quad \text{'incomplete_string'}
\]

**prompt**
Prompt string displayed on the first line during input. The string can consist of single-byte or multibyte characters. The default is RISQL.

**cont_prompt**
Continuation prompt string displayed for all subsequent lines during input. The string can consist of single-byte or multibyte characters. The default cont_prompt is >. If cont_prompt is not specified, the default prompt is used.

**incomplete_string**
Prompt string displayed when the user submits a query that contains an incomplete string. The string can consist of single-byte or multibyte characters. The default is INCOMPLETE STRING. If incomplete_string is not provided, the default prompt is used.

**Note:** All prompt strings must be enclosed in single quotation marks (').

**Changing Prompts**
You can change the first-line prompt without changing the continuation or incomplete string prompts. However, you must change the first-line and continuation prompts, or restate their default values, if you want to change subsequent prompts.

**Displaying Current Setting**
The QUERY PROMPT command displays the current status of the SET PROMPT command.

**Restoring Default**
The RESET PROMPT command returns the SET PROMPT command to its default state.
Command Reference

SET PROMPT

Example

RISQL> set prompt '>>> ' '>> ' 'UNMATCHED QUOTE> ';
>>> select * from store
>> where city = 'Los Angeles ;
UNMATCHED QUOTE>

See Also

QUERY
RESET
**SET ROW DATA**

Limits query results to the display of row data only (no column headings, blank lines, or so on).

```
 SET ROW DATA ONLY WITH HEADINGS
```

WITH HEADINGS
Displays data rows, column headings, and other non-data text in the result set.

ONLY
Displays only data rows in the result set.

**Default Settings**

WITH HEADINGS is the default setting in regular mode. ONLY is the default setting in quiet mode. For more information about quiet mode, refer to page 4-63.

**Using Row-Only Data**

The SET ROW DATA command can be used to extract selected data from a warehouse database and then place that data into a file for transfer to another program. This command can also be used to redirect row data output directly from the RISQL Entry Tool or RISQL Reporter to another program, such as the Table Management Utility (TMU).

**Redirecting Extraction Results from Within the Tool**

When used together, the following commands capture query results in a file for transfer to another program during a RISQL Entry Tool or RISQL Reporter session:

- SET ROW DATA ONLY causes the query result columns to be returned without column headings of other non-data text.
- SET ECHO OFF suppresses the RISQL prompt and input commands, as described on page 2-18 and page 4-51.
- SET OUTPUT filename sends row data output to a file rather than to the terminal (standard output), as described on page 2-19.
**Operating in Quiet Mode**

You can perform data-only extractions at the operating-system level directly from the UNIX shell by specifying quiet mode (`-q`) in the tool startup command. For more information about quiet mode, refer to “Redirecting Extraction Results from the Shell” on page 2-34 and Appendix B, “.rbretrc Files.”

**Example**

To send only the San Francisco Aroma Roma sales figures for the first quarter of 1995 to a file named `qtr_extract`, enter:

```
RISQL> set row data only;
RISQL> set echo off;
RISQL> set output qtr_extract;
RISQL> select * from store st
>   join sales sa on st.storekey = sa.storekey
>   join product pr on pr.prodkey = sa.prodkey
>   join period pe on pe.perkey = sa.perkey
> where year = 1995
> and qtr = 'Q1_95'
> and city = 'San Francisco'
> and prod_name = 'Aroma Roma';
```

The result of this query is sent directly to the `qtr_extract` file.

**See Also**

RESET ROW DATA
SET ECHO OFF
SET ERROR OUTPUT
SET OUTPUT
**SET TITLE**  
*(RISQL Reporter only)*

Defines a report title.

```
SET TITLE 'title_string'
```

**title_string**  
Character string that defines a title for a report. The string can consist of single-byte or multibyte characters. Report titles are left-justified. The `title_string` must be enclosed in single quotation marks (').

**Title Variables**

The `title_string` can contain one or more of the following predefined title variables:

- `$DATE` displays the date. The datetime data is formatted in accordance with the current client locale.
- `$TIME` displays the time. The timestamp data is formatted in accordance with the current client locale.
- `$PAGE` displays the page number.

Title variables are embedded in text strings and are case-sensitive. A blank line is displayed under the title and before column titles and data.

**Displaying Current Setting**

The `QUERY TITLE` command displays the current title.

**Restoring Default**

The `RESET TITLE` command deletes the current title.
Example

RISQL> set title 'SALES RANKING IN WEST REGION Report Run: $DATE at $TIME';

SALES RANKING IN WEST REGION Report Run: 5/23/96 at 05:12:52

<table>
<thead>
<tr>
<th>STORE_NAME</th>
<th>TOTAL_SALES</th>
<th>SALES_RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cupertino Coffee Supply</td>
<td>18709.50</td>
<td>1</td>
</tr>
<tr>
<td>San Jose Roasting Company</td>
<td>17867.90</td>
<td>2</td>
</tr>
<tr>
<td>Java Judy’s</td>
<td>17826.25</td>
<td>3</td>
</tr>
<tr>
<td>Beaches Brew</td>
<td>17197.25</td>
<td>4</td>
</tr>
<tr>
<td>Roasters, Los Gatos</td>
<td>15159.05</td>
<td>5</td>
</tr>
<tr>
<td>Instant Coffee</td>
<td>14841.75</td>
<td>6</td>
</tr>
</tbody>
</table>

See Also

QUERY
RESET
SET PAGE LENGTH
This appendix describes the `risql` and `risqlrpt` commands and their command-line options and environment variables.

The following syntax defines the command-line use of the `risql` and `risqlrpt` commands to invoke the RISQL Entry Tool and RISQL Reporter:

```
risql [-d database] [-q] [-s dsn] [-h host] [db_username [db_password]]
```

```
risqlrpt [-d database] [-q] [db_username [db_password]]
```

where:

- **risql** Invokes the RISQL Entry Tool.
- **risqlrpt** Invokes the RISQL Reporter.
- **-d database** Specifies a logical warehouse database name, as defined in the `rbw.config` file. Alternatively, a logical database name can be supplied by the environment variable `RB_PATH` or with an SQL CONNECT command and not specified on the command line. The `-d` option overrides `RB_PATH`.
- **-q** Causes the RISQL Entry Tool or RISQL Reporter to run in quiet mode. In quiet mode, the copyright notice is neither displayed on the screen nor printed to a file when output is redirected; the default setting for SET ECHO is OFF; and the default setting for SET ROW DATA is ONLY.
**risql and risqlrpt Command Reference**

**-s dsn**  
Data source name (DSN) defined in the .odbc.ini file on UNIX systems, odbc.ini file on Windows 95, and the Registry on Windows NT. A DSN can also be specified with the RB_DSN environment variable.

**-h host**  
RB_HOST value defined in the rbw.config file. This RB_HOST value can also be specified with the RB_HOST environment variable.

**db_username**  
Valid warehouse username.

**db_password**  
Warehouse password associated with the specified warehouse username.

**Database Connection**

If you neither provide the logical database name on the command line nor specify a database name with the RB_PATH environment variable, the RISQL Entry Tool or RISQL Reporter will start, but it will not be connected to a database. Before a database connection is established, the tool accepts only tool-specific or operating-system commands—not SQL statements.

When the `-d` option is provided, the tool searches for a logical database name definition in the rbw.config file.

When the `-s` option is provided, the tool searches for a logical data source name (DSN) definition in the .odbc.ini file (UNIX), odbc.ini file (Windows 95), or the Registry (Windows NT).

If you enter input from your terminal and you provide the `database` option at startup but fail to provide any login information, the tool prompts for both `db_username` and `db_password`. If you provide `db_username` but not `db_password`, the tool prompts for `db_password`.

If command input is redirected from a file before a database connection has been established, you must include a CONNECT statement on the first line of the input file before any SQL statements. The tool prompts for `db_username` and `db_password` only when input comes from a terminal, so you must include these arguments in your CONNECT statement. If either argument is missing from the source file, the warehouse connection attempt fails.

If a database connection is not made when the tool starts, you can use the CONNECT command at the RISQL prompt to establish one.
When a database connection is requested, the tool reads two initialization files named `.rbretrc` from the directory where the Red Brick Warehouse configuration files are located and from your home directory. Any valid SQL statement, RISQL statement, or tool command can be executed from either `.rbretrc` file. For details, refer to Appendix B, “.rbretrc Files.”

**Quiet Mode**

The optional `-q` parameter causes the RISQL Entry Tool or RISQL Reporter to run in quiet mode. In quiet mode, the copyright notice is neither displayed on the screen nor printed to a file when output is redirected. The default settings of the following SET commands are also changed:

- SET ECHO is set to OFF.
- SET ROW DATA is set to ONLY.

Quiet mode is helpful when you are using the RISQL Entry Tool or RISQL Reporter in data extraction scripts. For more information about operating in quiet mode, refer to page 2-34.

**Environment Variables**

The RISQL Entry Tool and RISQL Reporter use the following environment variables:

- RB_CONFIG, a pathname to the directory containing the `rbw.config` file, which is usually the directory that contains the `bin` subdirectory of warehouse executable files. RB_CONFIG is required.

The following error messages are displayed when the RB_CONFIG environment variable is either not defined or defined incorrectly when you start the RISQL Entry Tool:

- Undefined RB_CONFIG environment variable:
  
  **FATAL** **(803)** RB_CONFIG must be defined.

- Incorrectly defined RB_CONFIG environment variable:

  Error in opening config file; unable to continue.

  **FATAL** **(10502)** Message base lookup failed for message 10502, error code 200040.
When you start the RISQL Reporter, the tool checks the `rbw.config` file for license verification. If the RB_CONFIG environment variable is not set when the tool is started, the file is not found and the following message is displayed:

```
** FATAL ** (802) Not licensed for 'RISQL_REPORTER'.
```

For more information about Red Brick Warehouse error messages, refer to the `Messages and Codes Reference Guide`.

- RB_PATH, an environment variable which defines a logical database name in the `rbw.config` file. RB_PATH is optional.
- RB_HOST, an environment variable which identifies the warehouse daemon process. RB_HOST is optional.
- RB_DSN, an environment variable which defines a logical data source name.

**Option Precedence**

If both a command-line option and an environment variable are specified for any argument, the command-line option takes precedence.

**Input/Output Redirection**

Standard system input and output redirection is permitted (`>`, `<`, `>>`, `<<`, `|`).

Redirection settings affect the default ECHO state, which controls whether input is echoed with output.

**Examples**

The following example shows the username and password prompts when the RISQL Entry Tool is started with a logical database name.

```
$ risql -d aroma
(C) Copyright 1991-1998 Red Brick Systems, Inc. Los Gatos, California, USA All rights reserved.
RISQL Entry Tool Version 5.1.x (xxx)
Please enter username: curly
Please enter password: xxxxxxxx
RISQL>
```
The following example shows the username and password prompts when the RISQL Reporter is started with a logical database name.

$ risqlrpt -d aroma

Please enter username: curly
Please enter password: xxxxxxxx
RISQL>

See Also

“Using File Redirection” on page 2-15
“Creating Data Extraction Applications” on page 2-34
“CONNECT” on page 4-5
“SET ECHO” on page 4-51
Appendix B, “.rbretrc Files”
risql and risqlrpt Command Reference
This appendix describes the RISQL Entry Tool and RISQL Reporter initialization files (.rbretrc files) and provides an example to illustrate the file format.

The appendix is divided into the following sections:

- Tool Initialization
- Sample Initialization File

**Tool Initialization**

On startup, the RISQL Entry Tool and RISQL Reporter read an initialization file named .rbretrc. Any valid SQL or RISQL statement or tool command can be executed from .rbretrc. Only one command is permitted per line. Settings made in the .rbretrc file during a tool session (for example, during a shell-escape sequence) are not applied until the tool is restarted.

There is no default .rbretrc file; database administrators and users must create initialization files with an operating-system editor.

An .rbretrc file can exist in one or both of the following locations:

- Global .rbretrc file, which is located in the directory that contains the Red Brick Warehouse configuration file rbw.config and affects all sessions. This file is usually created by the warehouse administrator and must have global read permission set so that all users can access it.
- Local .rbretrc file, which is located in a user’s home directory and overrides the settings of the global file when both exist. This file is usually created by individual users.
If both global and local .rbretc files exist, the tool reads the global file first and then reads the local file. Therefore, statements and commands in local .rbretc files take precedence over statements and commands in the global .rbretc file. Any statements or commands in a global .rbretc file that are not present in a local .rbretc file are retained.

**Note:** On Windows platforms, the $HOME environment variable must be set to an existing directory that contains the .rbretc file.

**Sample Initialization File**

The following commands provide a customized user environment for either the RISQL Entry Tool or the RISQL Reporter; therefore, they are good candidates for a .rbretc file:

```bash
set prompt 'Enter Statement Here>' ;
set editor 'ed' ;
set echo on ;
set output program 'more -c' ;
```

The following commands are suitable for a “default” report format when you use the RISQL Reporter for report writing:

```bash
set page length 66 ;
set output /home/user/curly/report.test ;
set column * underline '=' ;
set column dollars decimal ;
set column dollars space 2 '$' ;
set column if null 'N/A' ;
```
This appendix explains how to override the warehouse locale for Red Brick client tools and lists the defined locale components supported by the Red Brick products.

**Overriding the Warehouse Locale**

RISQL Entry Tool and RISQL Reporter users can override the warehouse locale by setting the RB_NLS_LOCALE environment variable.

For example, from the UNIX C shell prompt:

```sh
% setenv RB_NLS_LOCALE German_Austria.Latin1@Default
```

From the Windows MS-DOS shell:

```cmd
C:\>set RB_NLS_LOCALE=German_Austria.Latin1@Default
```

where

- **German** = language
- **Austria** = territory
- **Latin1** = character set
- **Default** = collation sequence

It is not necessary to specify all four parts of a locale; however, it is strongly recommended that if the locale is not fully specified, the language should be one of the specified components. Otherwise, the unspecified components might default to incompatible values.
Locales
Overriding the Warehouse Locale

Note the following rules regarding default values for unspecified locale components:

• If only the language is specified, the omitted components are set to the
default values for that language. For example, if the locale is set to
Japanese
the complete locale specification will be as follows:

Japanese_Japan.JapanEUC@Binary

For a list of default components for each language, refer to Appendix A.

• If only the territory is specified, the language defaults to English, the
character set to US-ASCII, and the sort to Binary. For example, if the locale
is set to:

_Japan
the complete, but impractical, locale specification will be as follows:

English_Japan.US-ASCII@Binary

• Similarly, if only the character set is specified, the language defaults to
English, the territory defaults to UnitedStates, and the sort component
defaults to Binary.

• Finally, if only the sort component is specified, the language defaults to
English, the territory defaults to UnitedStates, and the character set defaults
to US-ASCII.

It is not necessary to specify all the separator characters (the underscore, the
period, and the @ character) in a partial locale specification. Only the character
that immediately precedes the component(s) is required—such as the
underscore character ( _ ) in the previous territory example.

For more information about the warehouse locale, refer to the Warehouse
Administrator’s Guide.
Defined Locale Components

The following table identifies the languages, territories, character sets, and sort components of a locale and lists the logical combinations of these components; however, any combination can be used in a locale specification.

The values shown in boldface are the default values for the corresponding language when an incomplete locale is specified. For example, the default character set for German is Latin1.

<table>
<thead>
<tr>
<th>Language</th>
<th>Territory</th>
<th>Character Set</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>United States</td>
<td>US-ASCII</td>
<td>Binary</td>
</tr>
<tr>
<td></td>
<td>Australia</td>
<td>Latin1</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Canada</td>
<td>MS1252</td>
<td></td>
</tr>
<tr>
<td></td>
<td>United Kingdom</td>
<td>UTF-8</td>
<td></td>
</tr>
<tr>
<td>German</td>
<td>Germany</td>
<td>Latin1</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Austria</td>
<td>MS1252</td>
<td>Binary</td>
</tr>
<tr>
<td></td>
<td>German-Switzerland</td>
<td>UTF-8</td>
<td></td>
</tr>
<tr>
<td>French</td>
<td>France</td>
<td>Latin1</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>French-Canada</td>
<td>MS1252</td>
<td>Binary</td>
</tr>
<tr>
<td></td>
<td>French-Switzerland</td>
<td>UTF-8</td>
<td></td>
</tr>
<tr>
<td>Spanish</td>
<td>Spain</td>
<td>Latin1</td>
<td>Spanish</td>
</tr>
<tr>
<td></td>
<td>Argentina</td>
<td>MS1252</td>
<td>Traditional</td>
</tr>
<tr>
<td></td>
<td>Chile</td>
<td>UTF-8</td>
<td>Spanish</td>
</tr>
<tr>
<td></td>
<td>Mexico</td>
<td></td>
<td>Binary</td>
</tr>
<tr>
<td>Portuguese</td>
<td>Portugal</td>
<td>Latin1</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>Brazil</td>
<td>MS1252</td>
<td>Binary</td>
</tr>
<tr>
<td>Italian</td>
<td>Italy</td>
<td>Latin1</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS1252</td>
<td>Binary</td>
</tr>
<tr>
<td>Norwegian</td>
<td>Norway</td>
<td>Latin1</td>
<td>Danish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS1252</td>
<td>Binary</td>
</tr>
</tbody>
</table>
## Locales

### Defined Locale Components

<table>
<thead>
<tr>
<th>Language</th>
<th>Territory</th>
<th>Character Set</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swedish</td>
<td>Sweden</td>
<td>Latin1</td>
<td>Swedish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS1252</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UTF-8</td>
<td></td>
</tr>
<tr>
<td>Danish</td>
<td>Denmark</td>
<td>Latin1</td>
<td>Danish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS1252</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UTF-8</td>
<td></td>
</tr>
<tr>
<td>Finnish</td>
<td>Finland</td>
<td>Latin1</td>
<td>Swedish</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS1252</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UTF-8</td>
<td></td>
</tr>
<tr>
<td>Japanese</td>
<td>Japan</td>
<td>JapanEUC</td>
<td>Binary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS932</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UTF-8</td>
<td></td>
</tr>
</tbody>
</table>

### Notes

- The component strings in this table must be spelled exactly as shown, but they are not case-sensitive.
- In the Sort column, any value that is not “Binary” is a linguistic sort definition. “Default” refers to the sort definition specified by the CAN/CSA Z243.4.1 Canadian ordering standard, which covers English and several Western European languages.
- All character sets include US-ASCII as a subset, so any of the listed character sets can be used when the language is English; however, the character sets listed for each language are the most appropriate choices.
- Character set conversions can reliably be performed between any two character sets listed for a given language. Conversions outside the scope of each language row in the table are not supported. For example, characters can be converted from Latin1 to MS1252 but not from Latin1 to JapanEUC.
- The MS932 character set, listed for Japanese, is a superset of Shift-JIS.
- Unicode is not a supported character set, nor are any shifted encoding schemes.
Index

Symbols
!, shell escape command 2-29, 4-4
$DATE 3-6, 4-64
$PAGE 3-6, 4-64
$TIME 3-6, 4-64
*, column variable wildcard 4-15
*, displayed in column 3-18, 4-29
+, displaying 4-36
+, placement 3-20, 4-38
-, placement 3-20, 4-38
.rbretrc file, processing 2-7
; in SQL syntax 2-6
>, <, file redirection symbols 2-15
\n, new-line character 2-6, 4-26
|, file redirection symbol 2-15

A
access, database 2-31
administration, database 2-31
asterisks(*)
  column variable 4-15
  displayed in column 3-18, 4-29
authorization, databases 2-2, 2-32

C
cases, tracked by technical support xvi
correspondence, sort methods C-4
currency symbol 3-23, 4-44
customer support xvi
customers, tracked by technical support xvi
currency symbol 3-23, 4-44
Customer Support Center xv

columns
  changing width 3-14, 4-49
  hiding 3-16, 4-33
  left-fill 3-17
  setting report format 4-21 to 4-50
  specification precedence 3-3, 4-21
  widening space between 3-15, 4-42
See also format settings, displaying and
  RESET COLUMN
commands, RISQL Entry Tool and RISQL
  Reporter
    command buffer 2-25
    SAVE syntax 4-20
    saving 3-2, 3-5

comments
  C-style 2-6, 2-27
  inserting 2-27
  SQL-style 2-6, 2-27

CONNECT command, SQL 4-5
conventions
  form of SELECT statements xiv
  syntax diagrams xii
  syntax notation xi
  csh prompt 2-2
currency symbol 3-23, 4-44
Customer Support Center xv
Index

D
  -d option 2-8, A-1
  data
    format, changing 3-18
    hiding 3-16, 4-33
    justifying 3-13, 4-31
    labeling 3-23
    left-fill 3-17, 4-24
    numeric format 3-18, 4-28
    numeric rounding 3-18, 4-28
    suppressing duplicate 3-21, 4-34
    See also format settings, displaying and
    RESET COLUMN command
  data source name (DSN) 4-5
  databases
    access to 2-2, 2-3
    administering 2-31
    authorizations for 2-2, 2-32
    changing information in 2-33
    connecting to 2-7 to 2-11, 4-5, A-2
    disconnecting from 2-10
  db_password
    CONNECT syntax 4-5
    startup syntax 2-3
  db_username
    CONNECT syntax 4-5
    startup syntax 2-3
  DECIMAL numeric format, SET COLUMN
    command 3-18, 4-28
  default editor 2-22, 4-53
  default report format, See initialization file
defaults, setting 4-13, 4-15
documentation
  list of Red Brick Systems ix
  support xvii
double-spaced reports 3-9, 4-40
  DSN
    See data source name
duplicate data, suppressing 3-21

E
  EDIT command, RISQL Entry Tool and
  RISQL Reporter 2-22, 4-7
  editors
    changing default 2-22, 4-53
    ending a session 2-23
    starting 2-22, 4-7
  e-mail addresses, for Red Brick
    Systems xv
  environment variables
    RB_CONFIG 2-2
    RB_HOST A-4
    RB_PATH 2-7, 2-8, A-2
  error messages
    discarding 2-20, 4-54
    identifying 2-7
    saving 2-20, 4-54
  execution, interrupting 2-14
  EXIT command, RISQL Entry Tool and
  RISQL Reporter 2-12, 4-12
  EXPONENTIAL numeric format, SET
  COLUMN command 3-18, 4-28

F
  file redirection, See redirection, file
  file, saving output 2-15, 4-55
  FILL option, SET COLUMN
    command 3-17, 4-24, 4-36, 4-38
  filter, specifying 2-20, 4-55
  FOLD LINE option, SET COLUMN
    command 3-7, 3-22, 4-26
  FORMAT option, SET COLUMN
    command 3-18, 4-28
  format settings
    displaying 3-26, 4-9
    saving 3-5, 4-20

G
  GRANT command 2-31

H
  headings, suppressing in output 4-62
  hiding data 4-33
    See also format settings, displaying and
    RESET COLUMN
Index

I
IF NULL option, SET COLUMN command 3-25, 4-30
INCOMPLETE STRING> prompt 2-25, 2-30, 4-60
initialization file 2-7, B-1, 4-11, 4-21 to 4-50
input
including in output 2-18, 4-51
reading 2-17
valid 2-6
INTEGER numeric format, SET COLUMN command 3-18, 4-28
invoking tool, See starting tool

J
JUSTIFY option, SET COLUMN command 3-13, 4-31

K
keywords, in syntax diagrams xiv
ksh prompt 2-2

L
layout, report 3-2
literals in SQL syntax 2-6
locales
default C-2
defining 2-3
list of supported C-3
logging in 2-4
logging out 2-12
logical database names 2-7

M
metavariables, in syntax diagrams xiv
minus sign, placement 3-20
more output program 2-20
multibyte characters 4-3

N
new-line character 2-6, 4-26
NO PRINT option, SET COLUMN command 3-16, 4-33
NO REPEAT option, SET COLUMN command 3-21, 4-34
notation conventions xi
NULL, displaying 3-25, 4-30
numeric format 3-18, 4-28

O
object names 4-3
output
appending 2-19, 4-55
discarding 2-19, 4-55
including input 2-18, 4-51
restoring default 2-20
saving 2-19, 4-55
output program, specifying 2-20, 4-55

P
page breaks, in reports 3-7
page layout
default, for reports 3-7
formatting reports 3-7, 4-57
passwords
changing 2-31
hiding 2-16
plus sign
displaying 4-36
placement 3-20
POSITIVE SIGN option, SET COLUMN command 4-36, 4-38
precedence, in column settings 3-3, 4-21
PROGRAM option, SET OUTPUT command 4-55
prompts
changing 2-30, 4-60
INCOMPLETE STRING> 2-25
Index

Q
-q option  2-5, 2-16, 2-19, A-1, A-3
queries  2-12, 2-14, 4-18
executing  2-16, 2-18
canceling  2-19
QUERY COLUMN command, RISQL  Reporter  3-26, 4-11
QUERY command, RISQL Entry Tool and  Reporter  4-9
RISQL Reporter  4-1
quiet mode  description  2-5, A-3
in data extraction applications  2-34
with redirection  2-16
QUIT command, RISQL Entry Tool and  Reporter  2-12, 4-12
quoted strings in SQL syntax  2-6

R
RB_CONFIG environment variable  2-2, A-3
RB_DSN environment variable  2-11, A-4
RB_HOST environment variable  A-4
RB_PATH environment variable  2-7, 2-8, A-2
.rbretrc file  2-7, B-1
redirection, file
overview  2-15
system-level  2-16
tool commands  2-15, 2-16
report format  building templates  3-5
column title  3-10, 4-45
creating templates  4-20
displaying + sign  4-36
displaying settings  3-26, 4-9, 4-11
laying out  3-2
overview  3-2
sample reports  3-28
saving defaults  3-5, 4-20
setting defaults  3-27
settings  4-21 to 4-50
sign placement  4-38
skipping lines  3-9, 4-40
wrapping lines  3-22, 4-26
report format, plus sign placement  4-36
report titles  3-6, 4-64
RESET COLUMN command, RISQL  Reporter  3-27, 4-15
RESET command, RISQL Entry Tool and  Reporter  4-13
risql and risqlrpt programs, in search
path  2-4
RISQL Entry Tool  description  1-2, 2-1
displaying settings  4-9
identifying version  2-4
initialization file  B-1
prompt  2-30, 4-60
restoring default settings  4-13
starting  2-4
suspending session  4-4
valid input  2-6
RISQL Entry Tool commands  CONNECT  4-5
EDIT  2-22, 4-7
EXIT  2-12, 4-12
QUERY  4-9
QUIT  2-12, 4-12
RESET  4-13
RUN  2-16
SET ECHO  2-18, 4-51
SET EDITOR  2-22, 4-53
SET ERROR OUTPUT  2-16, 2-20, 4-54
SET OUTPUT  2-16, 2-19, 4-55
SET PROMPT  2-30, 4-60
SET STATS  2-14
summary  4-1
syntax  2-6
RISQL Reporter  description  1-2, 2-1
displaying format settings  4-11
displaying settings  4-9
identifying version  2-5
initialization file  B-1
prompt  2-30, 4-60
restoring default settings  4-13
starting  2-4
supported features  1-2
suspending features  4-4
terminating session  4-12
valid input  2-6
Index

RISQL Reporter commands
CONNECT 4-5
EDIT 2-22, 4-7
EXIT 2-12, 4-12
QUERY 4-9
QUERY COLUMN 3-26, 4-11
QUIT 2-12, 4-12
RESET 4-13
RESET COLUMN 3-27, 4-15
RUN 2-16
SET COLUMN 4-21 to 4-50
SET ECHO 2-18, 4-51
SET EDITOR 2-22, 4-53
SET ERROR OUTPUT 2-16, 2-20, 4-54
SET OUTPUT 2-16, 2-19, 4-55
SET PAGE LENGTH 3-7, 4-57
SET PROMPT 2-30, 4-60
SET STATS 2-14
SET TITLE 3-6
summary 4-1
syntax 2-6
risql, command syntax 2-4, A-1
RISQL> prompt 2-30
risqlrpt, command syntax 2-4, A-1
RUN command, RISQL Entry Tool and
RISQL Reporter 2-16, 2-17, 2-25, 4-18

S
sample reports 3-28
SAVE command, RISQL Entry Tool and
RISQL Reporter 3-2, 3-5, 4-20
search path, modifying 2-4
SELECT statements, basic form xiv
semicolon, in SQL syntax 2-6
session, ending 2-12

SET COLUMN command, RISQL
FILL option 3-17, 4-24
FOLD LINE option 3-7, 3-22, 4-26
FORMAT option 3-18, 4-28
IF NULL option 3-25, 4-30
JUSTIFY option 3-13, 4-31
NO PRINT option 3-16, 4-33
NO REPEAT option 3-21, 4-34
POSITIVE SIGN option 4-36
SIGN option 4-38
SKIP LINE option 3-9, 4-40
SKIP PAGE option 3-7, 4-40
SPACE option 3-15, 3-23, 4-42
TITLE option 3-10, 4-45
UNDERLINE option 3-11, 4-47
usage 3-2
WIDTH option 3-14, 4-49

SET commands, RISQL Entry Tool and
RISQL Reporter
ECHO 2-18, 2-35, 4-51
EDITOR 2-22, 4-53
ERROR OUTPUT 2-16, 2-20, 4-54
OUTPUT 2-16, 2-19, 2-35, 4-55
PAGE LENGTH 3-7, 4-57
PROGRAM option, OUTPUT 4-55
PROMPT 2-30, 4-60
ROW DATA ONLY 2-35, 4-62
TITLE 3-6

SET STATS command 2-14
sh prompt 2-2
shell escape command 2-29, 4-4
interactive session 2-29
prompt 2-2
suspending 2-29, 4-4
termination 2-6
SIGN option, SET COLUMN
command 3-20, 4-36, 4-38
single-byte characters 4-3
SKIP LINE option, SET COLUMN
command 3-9, 4-40
SKIP PAGE option, SET COLUMN
command 3-7, 4-40
SPACE option, SET COLUMN
command 3-15, 3-23, 4-42
starting tool
See also initialization file
Index

statistics
  - reporting 2-14
  - turning off 2-15
status codes, exiting 2-35
support
  - documentation xvii
  - solutions warehouse xv
  - technical xv
suspending
  - shell 2-29, 4-4
  - tool session 4-4
syntax diagrams
  - conventions for xii
  - keywords in xiv
  - metavariables in xiv
syntax notation xi
system commands operation,
  - running 2-15 to 2-16, 2-29
  - system prompt 2-2

T
  - technical support xv
  - templates, report 3-5, B-2
  - TITLE option, SET COLUMN command 3-10, 3-14, 4-45
titles, report 3-6, 4-64
tool session
  - ending 2-12, 4-12
  - starting 2-12, 4-12
troubleshooting, general problems xvii

U
  - UNDERLINE option, SET COLUMN command 3-11, 4-47
  - UNIX prompt 2-2
  - user access 2-31
  - user name, hiding 2-16

W
  - widening space between columns 4-42
  - WIDTH option, SET COLUMN command 3-14, 4-49
  - Windows NT prompt 2-2
  - World Wide Web address, for Red Brick Systems xv
USA SALES OFFICES
1040 Crowne Point Parkway, Suite 250, Atlanta, GA 30338 +1 770 804 2440
2215 York Road, Suite 409, Oak Brook, IL 60521 +1 630 472 8500
1120 Avenue of the Americas, 4th Floor, New York, NY 10036 +1 212 626 6815
5314 Arapaho Road, Dallas, TX 75248 +1 972 702 1750
2010 Corporate Ridge, 7th Floor, McLean, VA 22102 +1 703 883 9310

UK SALES OFFICE
Red Brick Systems UK Ltd., 45 Berkeley Square, Mayfair, London W1A 1EB
United Kingdom +44 171 290 8373

AUSTRALASIA HEADQUARTERS
Red Brick Systems Australasia Pty. Ltd., Level 20, 99 Walker Street,
North Sydney, NSW 2060 Australia +61 02 9911 7744

JAPAN HEADQUARTERS
Red Brick Japan Co. Ltd., Level 16 Shiroyama Hills, 4-3-1 Toranomon,
Minato-ku, Tokyo 105 Japan +81 3 5403 4638