



System i  
Database  
Database overview

*Version 6 Release 1*







System i  
Database  
Database overview

*Version 6 Release 1*

**Note**

Before using this information and the product it supports, read the information in “Notices,” on page 43.

This edition applies to version 6, release 1, modification 0 of IBM i5/OS (product number 5761-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

© Copyright International Business Machines Corporation 2004, 2008. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Database overview</b> . . . . .	<b>1</b>
What's new for V6R1 . . . . .	1
PDF file for Database overview . . . . .	1
DB2 for i5/OS . . . . .	2
Terminology: SQL versus traditional file access . . . . .	2
Getting started with System i Navigator . . . . .	2
Starting System i Navigator . . . . .	3
Creating a schema . . . . .	3
Editing the list of schemas displayed . . . . .	3
Creating and using a table . . . . .	4
Defining additional columns on a table . . . . .	6
Creating the supplier table . . . . .	7
Copying column definitions . . . . .	7
Inserting information into a table . . . . .	7
Viewing the contents of a table . . . . .	8
Changing information in a table . . . . .	9
Deleting information from a table . . . . .	9
Copying and moving a table . . . . .	10
Copying a table . . . . .	10
Moving a table . . . . .	10
Creating and using a view . . . . .	11
Creating a view over a single table . . . . .	11
Creating a WHERE clause . . . . .	12
Creating a view that combines data from multiple tables . . . . .	13
Deleting database objects . . . . .	15
System i Navigator database tasks . . . . .	15
Database objects creation tasks . . . . .	16
Database objects operation tasks . . . . .	18

Database performance optimization tasks . . . . .	21
Mapping your database . . . . .	24
Querying your database by running SQL scripts . . . . .	24
Stopping SQL scripts . . . . .	25
Viewing the job log . . . . .	25
Generating SQL for existing objects . . . . .	25
Building SQL statements with SQL Assist . . . . .	25
Starting System i5 Debugger . . . . .	26
Managing check pending constraints . . . . .	26
Importing and exporting data . . . . .	26
Getting started with SQL . . . . .	27
Creating a schema . . . . .	27
Creating and using a table . . . . .	28
Using the LABEL ON statement . . . . .	29
Inserting information into a table . . . . .	31
Getting information from a single table . . . . .	33
Getting information from multiple tables . . . . .	35
Changing information in a table . . . . .	37
Deleting information from a table . . . . .	40
Creating and using a view . . . . .	40
Creating a view on a single table . . . . .	41
Creating a view that combines data from multiple tables . . . . .	41

<b>Appendix. Notices</b> . . . . .	<b>43</b>
Programming interface information . . . . .	44
Trademarks . . . . .	45
Terms and conditions . . . . .	45



---

## Database overview

DB2® for i5/OS® shares characteristics with many other DB2 implementations. But if you have just migrated to the System i™ product, you might wonder how differently DB2 for i5/OS works on the system and what advantages the system brings to database development.

The System i product as a database platform has various strengths. This topic collection describes how to assess which data access methods make the most sense for your organization and how to build a rough framework for developing and maintaining your database implementation.

You can also explore other database information using the main navigation tree.

---

## What's new for V6R1

Read about new or significantly changed information for the Database overview topic collection.

Various System i Navigator database tasks have been added or updated this release. For more information, see the following topics:

- “Database objects creation tasks” on page 16
- “Database objects operation tasks” on page 18
- “Database performance optimization tasks” on page 21

## How to see what's new or changed

To help you see where technical changes have been made, the information center uses:

- The  image to mark where new or changed information begins.
- The  image to mark where new or changed information ends.

In PDF files, you might see revision bars (|) in the left margin of new and changed information.

To find other information about what's new or changed this release, see the Memo to users.

---

## PDF file for Database overview

You can view and print a PDF file of this information.

To view or download the PDF version of this document, select Database overview (about 680 KB).

## Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF link in your browser.
2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

## Downloading Adobe Reader

You need Adobe® Reader installed on your system to view or print these PDFs. You can download a free copy from the Adobe Web site ([www.adobe.com/products/acrobat/readstep.html](http://www.adobe.com/products/acrobat/readstep.html)) .

---

## DB2 for i5/OS

DB2 for i5/OS is the relational database manager that is fully integrated on your System i product. Because it is integrated on the system, DB2 for i5/OS is easy to use and manage.

DB2 for i5/OS also provides many functions and features, such as triggers, stored procedures, and dynamic bitmapped indexing, that serve a wide variety of application types. These applications range from traditional host-based applications to client/server solutions to business intelligence applications.

As an interface to DB2 for i5/OS, the IBM® DB2 Query Manager and SQL Development Kit for i5/OS licensed program adds an interactive query and report writing interface, together with precompilers and tools, to help you write Structured Query Language (SQL) application programs in high-level languages. Conforming to the industry standard SQL, the SQL implementation for the i5/OS operating system allows you to define, manipulate, query, and control access to your i5/OS data. It works equally well with i5/OS files and SQL tables.

---

### Terminology: SQL versus traditional file access

DB2 for i5/OS provides two access methods for manipulating database tables and data: SQL and system file access methods. These access methods use different words to describe some similar concepts.

SQL term	Traditional file access term
<b>Schema.</b> A group of related objects that consists of a library, a journal, a journal receiver, an SQL catalog, and an optional data dictionary. A schema enables the user to find the objects by name. Another name for a schema is <i>collection</i> .	<b>Library.</b> A group of related objects that enables the user to find the objects by name.
<b>Table.</b> A set of columns and rows.	<b>Physical file.</b> A set of records.
<b>Row.</b> The horizontal part of a table containing a serial set of columns.	<b>Record.</b> A set of fields.
<b>Column.</b> The vertical part of a table of one data type.	<b>Field.</b> One of more bytes of related information of one data type.
<b>View.</b> A subset of columns and rows of one or more tables.	<b>Logical file.</b> A subset of fields or records of up to 32 physical files.
<b>Index.</b> A collection of data in the columns of a table, logically arranged in ascending or descending order.	<b>Index.</b> A type of logical file.
<b>Package.</b> An object that contains control structures for SQL statements to be used by an application server.	<b>SQL package.</b> An object that contains control structures for SQL statements to be used by an application server.
<b>Catalog.</b> A set of tables and views that contain information about tables, packages, views, indexes, and constraints.	No similar object. However, the Display File Description (DSPFD) and Display File Field Description (DSPFFD) commands provide some of the same information that querying an SQL catalog provides.

---

### Getting started with System i Navigator

System i Navigator is a graphical interface that you can use to perform many common administrative database operations. Most of the System i Navigator operations are based on SQL, but you do not need to fully understand SQL to perform them.

#### Related concepts

“System i Navigator database tasks” on page 15

You can perform many database tasks with System i Navigator, including creating, modifying, and deleting various database objects.

#### Related tasks

## Starting System i Navigator

You need to start System i Navigator before using it to manage your database objects.

To start System i Navigator, follow these steps:

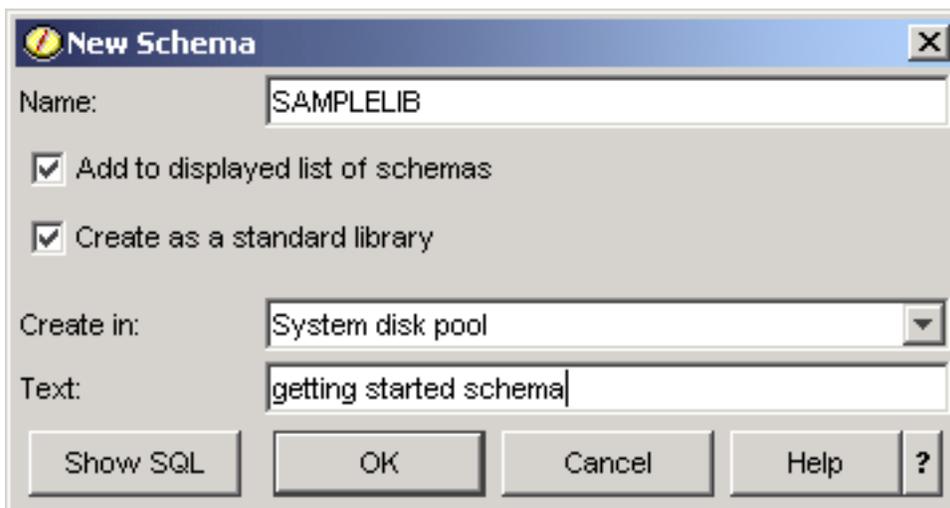
1. Double-click the **System i Navigator** icon.
2. Expand the system you want to use.

## Creating a schema

A database schema provides a logical classification of database objects. After you successfully create a schema, you can create tables, views, indexes, stored procedures, user-defined functions, and user-defined types in the schema.

To create a schema, follow these steps. The procedure uses SAMPLELIB for the schema name.

1. From System i Navigator, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Right-click **Schemas** and select **New Schema**.
4. In the New Schema window, type SAMPLELIB in the name field.
5. To add the newly created schema to the list of schemas to be displayed, select **Add to displayed list of schemas**.
6. Select **Create as a standard library**.
7. Specify a disk pool to contain the schema. Choose one so that the schema is created on the system disk pool.
8. Optional: Specify a description.
9. Click **OK**.



### Related concepts

Creating database objects

### Related tasks

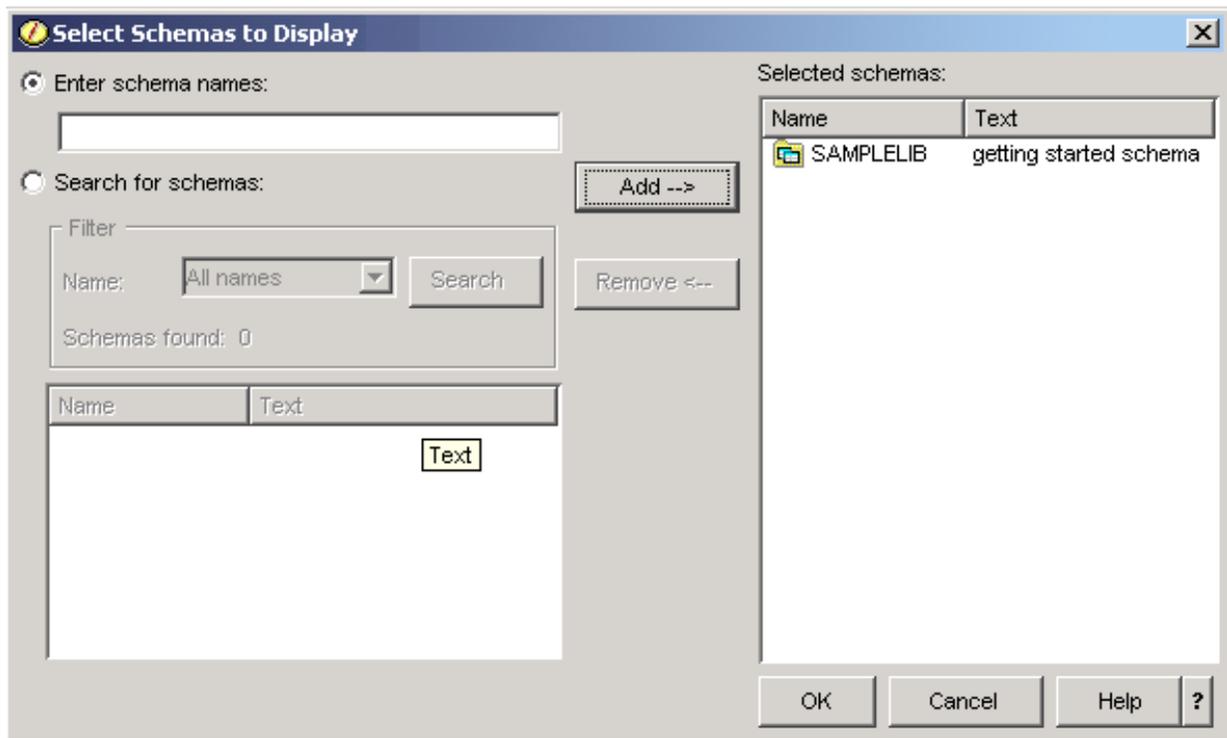
Working with multiple databases

## Editing the list of schemas displayed

By editing the list of schemas displayed, you can hide from your view those schemas that you do not use frequently.

To edit the list of schemas that is displayed when you click the **Schemas** folder, follow these steps:

1. Right-click **Schemas** and select **Select Schemas to Display**.
2. In the Select Schemas to Display window, you can edit the list by selecting **Enter schema names** and specifying a schema, or by selecting **Search for schemas** and performing a search. Select the schema that you want to display and click **Add**.
3. To remove a schema from the list of schemas to display, select the schema from the list and click **Remove**.



4. For now, leave SAMPLELIB as the schema displayed.

## Creating and using a table

A table is a basic database object that is used to store information. After you create a table, you can define columns, create indexes, and add triggers and constraints.

When you are creating a table, you need to understand the concepts of null value and default value. A *null value* indicates the absence of a column value for a row. It is not the same as a value of zero or all blanks. It means unknown. A null value is not equal to any value, not even to other null values. If a column does not allow the null value, a value must be assigned to the column. This value is either a default value or a user-supplied value.

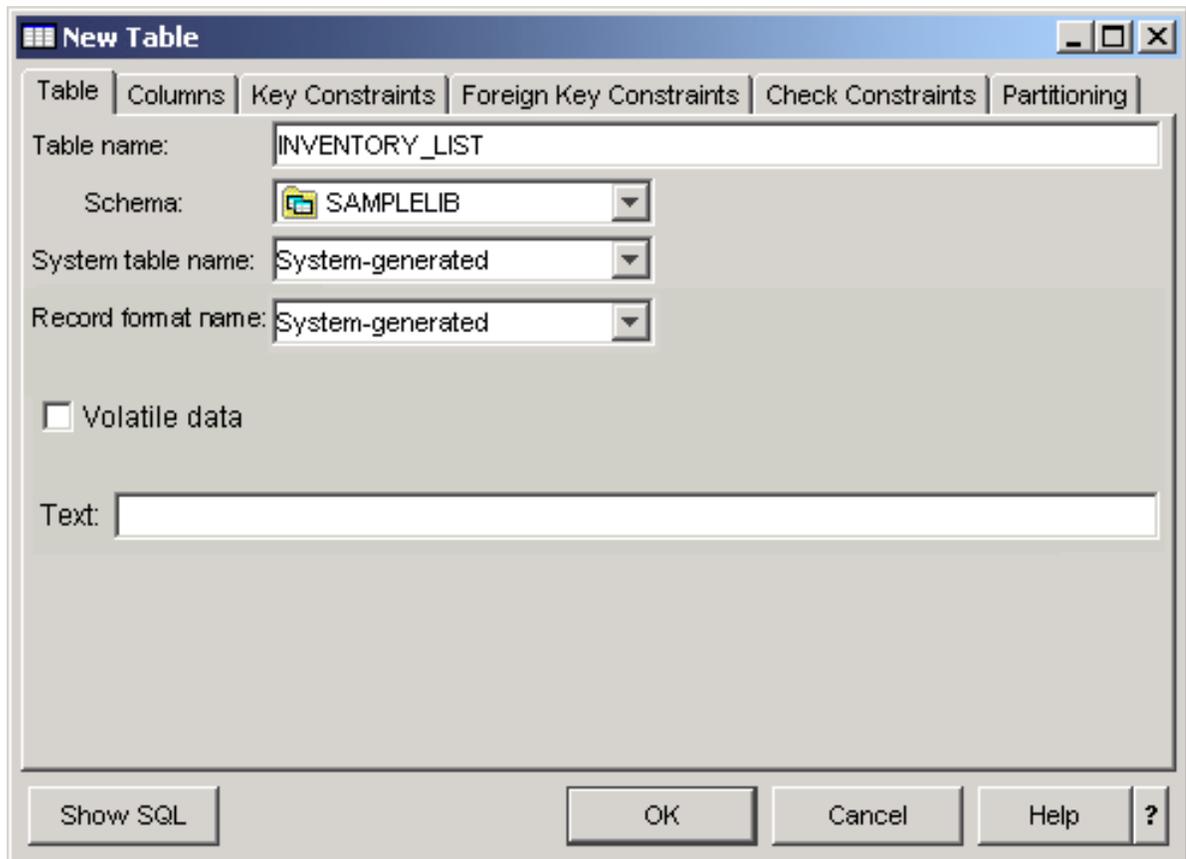
If no value is specified for a column when a row is added to a table, the row is assigned a *default value*. If the column is not assigned a specific default value, the column uses the system default value.

This example shows how to create a table to maintain information about the current inventory of a business. The table contains information about the items kept in the inventory, their cost, quantity currently on hand, the last order date, and the number last ordered. The item number is a required value. It cannot be null. The item name, quantity on hand, and order quantity have user-supplied default values. The last order date and quantity allow null values.

To create the table, follow these steps:

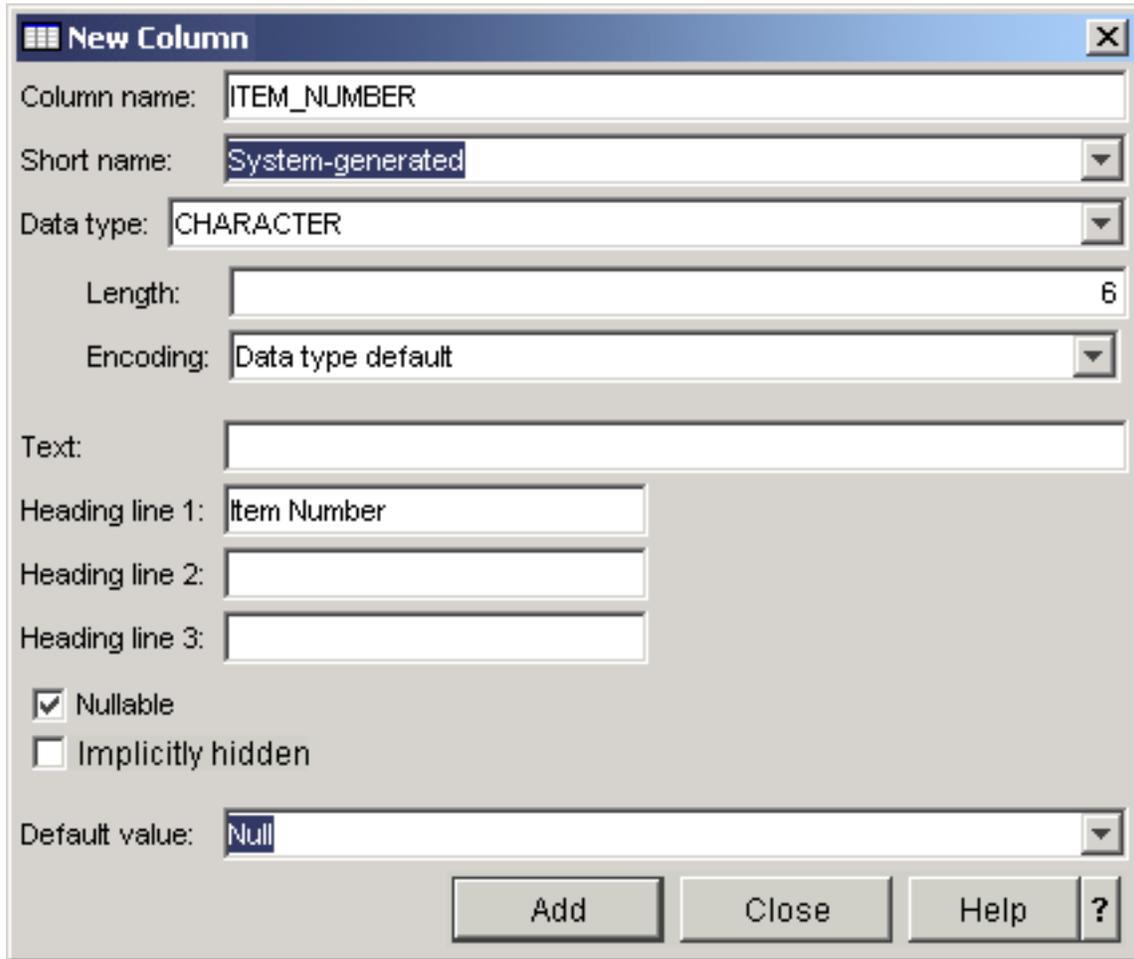
1. From System i Navigator, expand the system that you want to use.

2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Right-click **SAMPLELIB** and select **New** → **Table** → **Table**. The New Table window is shown.
5. On the **Table** tab, provide the following information for the new table:
  - a. Specify **INVENTORY\_LIST** as the table name.
  - b. In the **Schema** field, select **SAMPLELIB**.
  - c. In the **System table name** field and the **Record format name** field, select **System-generated**.
  - d. Optional: In the **Text** field, specify a description for the table.



6. On the **Columns** tab, click the **Add** button to define a column for the new table. The New Column window is shown.
  - a. In the **Column name** field, type **ITEM\_NUMBER**.
  - b. Optional: In the **Short name** field, specify a short name for the column. If you do not specify a short name, the system automatically generates a name. If the column name is 10 characters or less, then the short name is the same as the column name. You can perform queries by using either column name. For this example, leave this space as the default, **System-generated**.
  - c. Select **CHARACTER** as the data type.
  - d. Specify a length of 6 for this column. For data types of which the size is predetermined, the size is provided and you cannot change the value.
  - e. Leave the **Encoding** option as the default, **Data type default**.
  - f. Optional: In the **Text** field, specify a description for the column.
  - g. In the **Heading** fields, type a column heading. The heading is the label that shows at the top of the column for displaying or printing. The heading is limited to 60 characters, 20 per line.
  - h. Clear the **Nullable** check box. This ensures that a value must be placed in this column for the row insertion to be successful.

- i. In the **Default value** field, type 0.
- j. Click **Add** to add the column.
- k. Click **Close** to close the New Column window.



- 7. Click **OK** to create the table.

The new table INVENTORY\_LIST is shown.

### Defining additional columns on a table

After you create a table, you can still add new columns to the table.

To add columns to the INVENTORY\_LIST table that you created, follow these steps:

1. Navigate to the table by expanding *your database* → **Schemas** → **SAMPLELIB** → **Tables**.
2. In the detail pane, right-click **INVENTORY\_LIST** and select **Definition**.
3. In the Table Definition window, click the **Columns** tab and click **Add**.
4. Define the following new columns.

Column name	Type	Length	Precision	Scale	Nullable	Default value
ITEM_NAME	VARCHAR	20			No	UNKNOWN
UNIT_COST	DECIMAL		8	2	No	0
QUANTITY_ON_HAND	SMALLINT				Yes	NULL
LAST_ORDER_DATE	DATE				Yes	NULL

Column name	Type	Length	Precision	Scale	Nullable	Default value
ORDER_QUANTITY	SMALLINT				Yes	20

5. Click **OK** to add these columns.

### Creating the supplier table

Assume that later you need a second table. This table contains information about suppliers of inventory items, which items they supply, and the cost of the item from that supplier.

Create a table called SUPPLIERS in SAMPLELIB. This table has three columns: SUPPLIER\_NUMBER, ITEM\_NUMBER, and SUPPLIER\_COST.

**Note:** This table has a common column with the INVENTORY\_LIST table: ITEM\_NUMBER. Instead of creating a new ITEM\_NUMBER column, you can copy the column definition used for ITEM\_NUMBER in INVENTORY\_LIST.

### Copying column definitions

Copying column definitions saves you from defining columns in multiple tables by sharing the same definitions across the tables.

In this example, the ITEM\_NUMBER column in the SUPPLIERS table shares the definition of the ITEM\_NUMBER column in the INVENTORY\_LIST table. To copy column definitions, follow these steps:

1. In the New Table window or in the SUPPLIERS table definition window, click **Browse** on the **Columns** tab.
2. In the Browse Columns window, expand **SAMPLELIB**.
3. Click **INVENTORY\_LIST**. The columns in this table are listed, along with their data types, sizes, and descriptions.
4. Select **ITEM\_NUMBER**.
5. Click **Add** to copy this column definition to the SUPPLIERS table.
6. Close the Browse Columns window.

You can then add the following additional columns to the SUPPLIERS table.

Column name	Type	Length	Precision	Scale	Nullable	Default value
SUPPLIER_NUMBER	CHAR	4			No	0
SUPPLIER_COST	DECIMAL		8	2	Yes	NULL

### Inserting information into a table

After you create a table and define its columns, you can insert data into your table.

To add data to the INVENTORY\_LIST table, follow these steps:

1. From System i Navigator, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Select **SAMPLELIB**.
5. Double-click **Tables**.
6. Right-click **INVENTORY\_LIST** and select **Edit Contents**.
7. From the **Rows** menu, select **Insert**. A new row is shown.
8. Type the information in the following table under the appropriate headings.

**Note:** The values that you insert must satisfy all constraints and satisfy the type of each column. If there is a unique constraint or an index over the table, the values that you insert must be unique key values. If you do not insert a value in a column, the default value is used, if allowed. For this example, insert only those values shown in the following table so that the default values are used.

ITEM_NUMBER	ITEM_NAME	UNIT_COST	QUANTITY_ON_HAND
153047	PENCILS, RED	10.00	25
229740	LINED TABLETS	1.50	120
544931		5.00	
303476	PAPER CLIPS	2.00	100
559343	ENVELOPES, LEGAL	3.00	500
291124	ENVELOPES, STANDARD		
775298	CHAIRS, SECRETARY	225.00	6
073956	PENS, BLACK	20.00	25

9. From the **File** menu, select **Save**.

10. Add the following rows to the SAMPLELIB.SUPPLIERS table.

ITEM_NUMBER	SUPPLIER_NUMBER	SUPPLIER_COST
153047	1234	10.00
229740	1234	1.00
303476	1234	3.00
153047	9988	8.00
559343	9988	3.00
153047	2424	9.00
303476	2424	2.50
775298	5546	225.00
303476	3366	1.50
073956	3366	17.00

11. From the **File** menu, select **Save**. The example schema now contains two tables with rows of data in each table.

## Viewing the contents of a table

You can display the contents of your tables and views. When viewing the contents of a table, you cannot make changes to the table. To make changes to a table, you must edit the table.

To view the contents of INVENTORY\_LIST, follow these steps:

1. From System i Navigator, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Click **SAMPLELIB**.
5. Double-click **Tables**.
6. Right-click **INVENTORY\_LIST** and select **View Contents**.

	ITEM_NUMBER	ITEM_NAME	UNIT_COST	QUANTITY_ON_HAND	LAST_ORDER_DATE	ORDER_QUANTITY
1	153047	PENCILS, RED	10.00	25	-	20
2	229740	LINED TABLETS	1.50	120	-	20
3	544931	UNKNOWN	5.00	-	-	20
4	303476	PAPER CLIPS	2.00	100	-	20
5	559343	ENVELOPES, LEGAL	3.00	500	-	20
6	291124	ENVELOPES, STANDARD	0	-	-	20
7	775298	CHAIRS, SECRETARY	225.00	6	-	20
8	073956	PENS, BLACK	20.00	25	-	20

### Changing information in a table

You can use System i Navigator to change the values in the columns of a table. The value that you give must be valid for that column.

Suppose that you want to update a column to indicate that you received an order for more paper clips today.

1. Navigate to the table INVENTORY\_LIST. Right-click the table and select **Edit Contents**.
2. Type the current date in the LAST\_ORDER\_DATE column for paper clips. Be sure to use the correct date format for your system.
3. Change the ORDER\_QUANTITY value to 50.
4. Save the changes and view the table contents by using **View Contents**.

The paper clip row reflects the changes you made.

### Deleting information from a table

You can delete information from a single column in a row or delete the row entirely. If a column requires a value, you cannot delete it without deleting the entire row.

To delete some information in the INVENTORY\_LIST table, follow these steps:

1. Open the INVENTORY\_LIST table by double-clicking it.
2. Delete the ORDER\_QUANTITY value for the ENVELOPES, STANDARD row. Because this column allows null values, you can delete the value.
3. Delete the UNIT\_COST value for the LINED TABLETS row. Because this column does not allow null values, the deletion is not allowed.

You can also delete an entire row without removing all of the column values one at a time.

1. Open table INVENTORY\_LIST by double-clicking it.
2. Click the cell to the left of the UNKNOWN row. This highlights the entire row.
3. Select **Delete** from the **Rows** menu or press the Delete key on your keyboard. The UNKNOWN row is deleted.
4. Delete all of the rows that do not have a value in the QUANTITY\_ON\_HAND column from the INVENTORY\_LIST table.
5. Save the changes and view the contents by using **View Contents**.

You should have a table that contains the following data.

ITEM_ NUMBER	ITEM_ NAME	UNIT_ COST	QUANTITY_ ON_ HAND	LAST_ ORDER_ DATE	ORDER_ QUANTITY
153047	PENCILS, RED	10.00	25		20
229740	LINED TABLETS	1.50	120		20
303476	PAPER CLIPS	2.00	100	2007-09-22	50
559343	ENVELOPES, LEGAL	3.00	500		20
775298	CHAIRS, SECRETARY	225.00	6		20
073956	PENS, BLACK	20.00	25		20

## Copying and moving a table

You can copy or move tables from one schema or system to another. By copying a table, you create more than one instance of the table. By moving a table, you transfer the table to its new location while removing the instance from its former location.

### Copying a table:

In this example, you copy the INVENTORY\_LIST table to another schema.

Create a new schema called LIBRARY1 and add it to the list of schemas displayed. After you have created this new schema, copy INVENTORY\_LIST to the LIBRARY1 schema. To copy a table, follow these steps:

1. From System i Navigator, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Click **SAMPLELIB**.
5. Double-click **Tables**.
6. Right-click **INVENTORY\_LIST** and select **Copy**.
7. Right-click **LIBRARY1** and select **Paste**.

### Moving a table:

Now that you have copied the INVENTORY\_LIST table to the LIBRARY1 schema, move the SUPPLIERS table to LIBRARY1.

To move a table, follow these steps:

1. From System i Navigator, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Click **SAMPLELIB**.
5. Double-click **Tables**.
6. Right-click **SUPPLIERS** and select **Cut**.
7. Right-click **LIBRARY1** and select **Paste**.

**Note:** You can move a table by dragging and dropping the table on the new schema. Moving a table to a new location does not always remove it from the source system. For example, if you have read authority but not delete authority to the source table, you can move the table to the target system. However, you cannot delete the table from the source system, causing two instances of the table to exist.

## Creating and using a view

You might find that no single table in the database contains all the information that you need. You might also want to give users access to only part of the data in a table. Views provide a way to divide the table so that you deal with only the data that you need.

A view reduces complexity and, at the same time, restricts access. To create a view, you must have the correct authority to the tables or physical files on which the view is based. See the CREATE VIEW statement for a list of authorities needed.

If you did not specify column names in the view definition, the column names are the same as those for the table on which the view is based.

You can make changes to a table through a view even if the view has a different number of columns or rows than the table. For INSERT, columns in the table that are not in the view must have a default value.

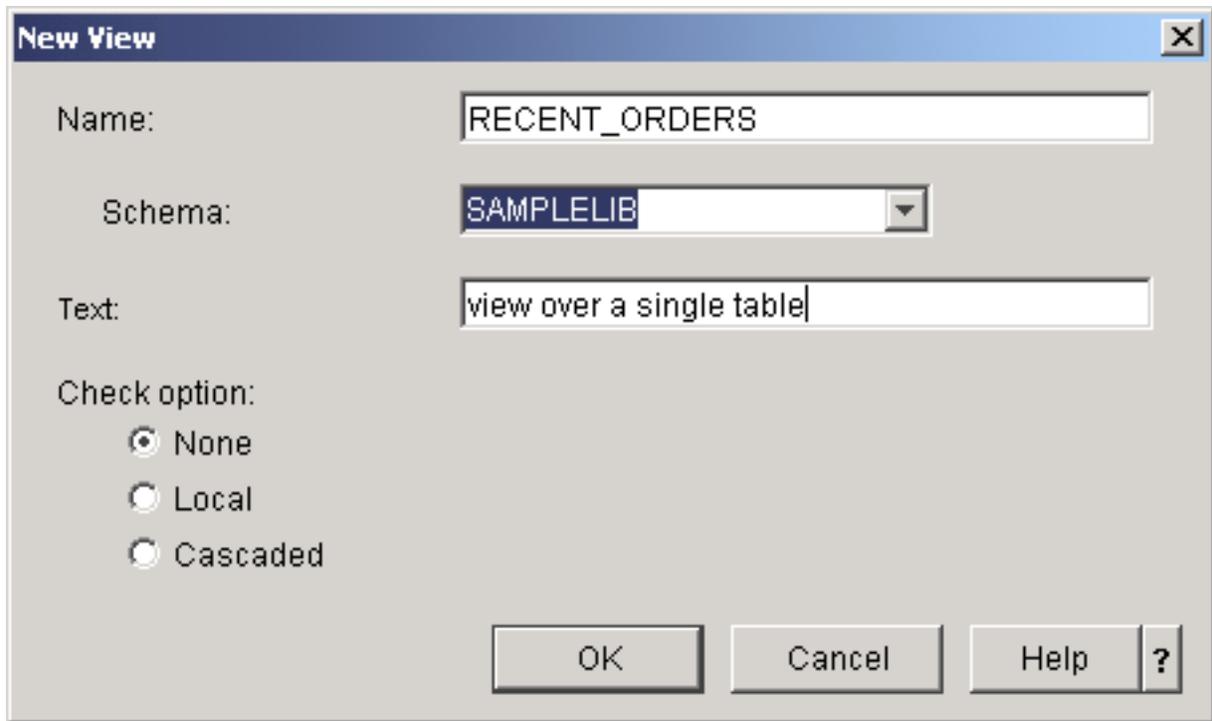
You can use the view as though it were a table, even though the view is totally dependent on one or more tables for data. The view has no data of its own and therefore requires no storage for the data. Because a view is derived from a table that exists in storage, when you update the view data, you are really updating data in the table. Therefore, views are automatically kept up-to-date as the tables they depend on are updated.

### Creating a view over a single table

This example shows how to create a view over a single table. The view is built on the INVENTORY\_LIST table. The table has six columns, but the view uses only three of them: ITEM\_NUMBER, LAST\_ORDER\_DATE, and QUANTITY\_ON\_HAND.

To create a view over a single table, follow these steps:

1. From System i Navigator, expand the system that you want to use.
2. Expand **Databases** and the database that you want to work with.
3. Expand **Schemas**.
4. Right-click **SAMPLELIB** and select **New** → **View**.
5. In the New View window, type RECENT\_ORDERS in the **Name** field.
6. In the **Schema** field, specify SAMPLELIB.
7. Optional: In the **Text** field, specify a description.
8. Select a check option. A check option on a view specifies that the values inserted or updated into a row must conform to the conditions of the view. For this view, select **None**.
9. Click **OK**.



The New View definition window then opens.

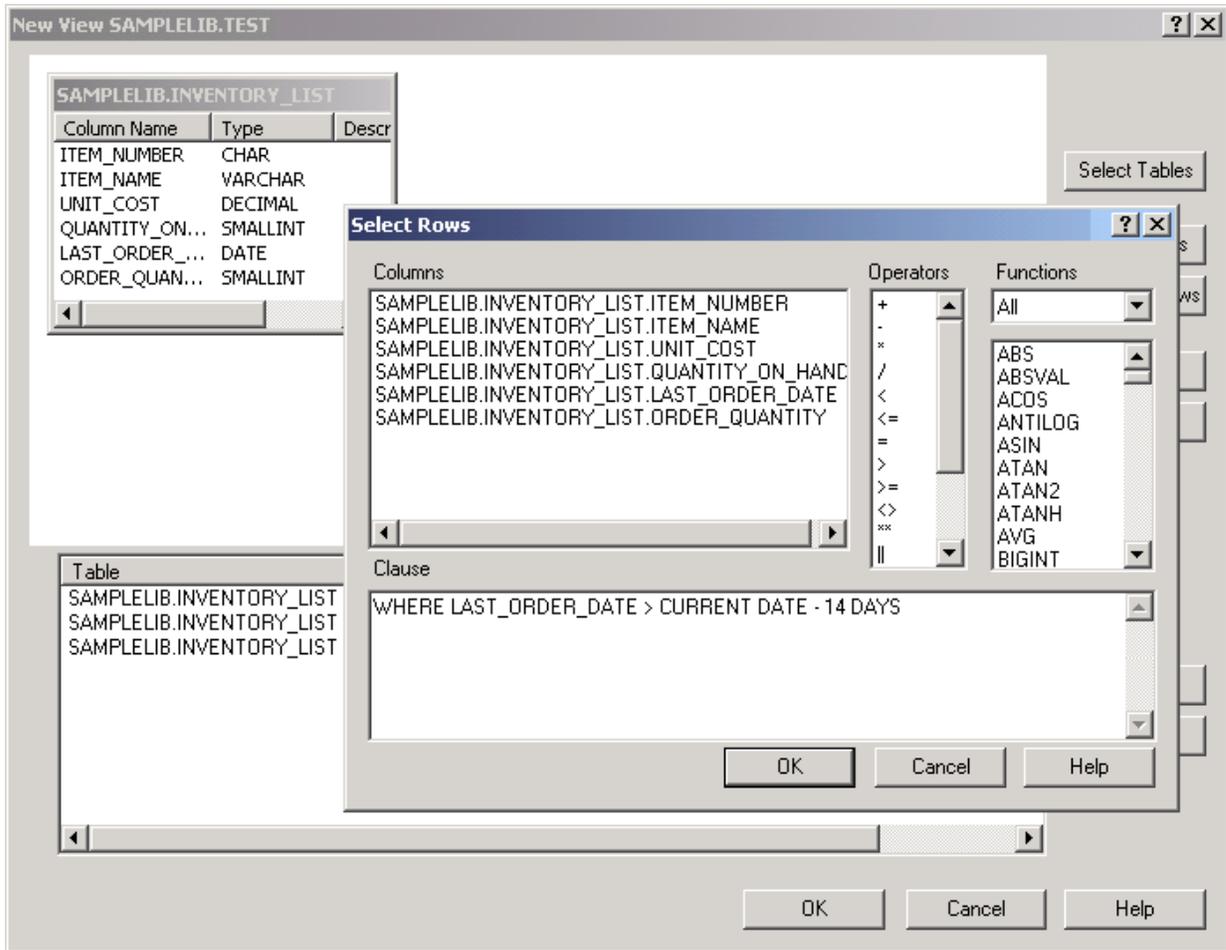
10. In the New View definition window, click **Select Tables**.
11. In the Browse Tables window, expand **SAMPLELIB**, and then select **INVENTORY\_LIST**.
12. Click **Add**.
13. Click **OK**. **INVENTORY\_LIST** should now be in the work area of the New View window.
14. To choose the columns that you want in the new view, click them in the selected tables and drag them to the selection grid in the bottom half of the window. Select **ITEM\_NUMBER**, **LAST\_ORDER\_DATE**, and **QUANTITY\_ON\_HAND**.
15. The order that the columns appear in the selection grid is the order that they will appear in the view. To change the order, select a column and drag it to its new position. Put the columns in the following order: **ITEM\_NUMBER**, **LAST\_ORDER\_DATE**, **QUANTITY\_ON\_HAND**.

### Creating a WHERE clause

With a WHERE clause on a table or a view, you can display only records that match certain criteria.

In this example, you only want to view those items that have been ordered in the last 14 days. To specify this information, you need to create a WHERE clause:

1. Click **Select Rows**.
2. In the Select Rows window, enter the following command: `WHERE LAST_ORDER_DATE > CURRENT DATE - 14 DAYS`. You can select the elements that make up this WHERE clause by selecting them from the options shown, or you can enter them in the **Clause** field.
3. Click **OK**.
4. To view the SQL used to generate this view, click **Show SQL**.
5. Click **OK** to create the view.



- To display the contents of the RECENT\_ORDERS view, right-click **RECENT\_ORDERS** and select **View Contents**.

You should see the following information displayed.

ITEM_NUMBER	LAST_ORDER_DATE	QUANTITY_ON_HAND
303476	2007-09-22	100

In the preceding example, the columns in the view have the same names as the columns in the table because you did not specify new names. You can create the view into any schema, not necessarily into the schema that contains the table it is built over.

### Creating a view that combines data from multiple tables

If you want to display data from multiple tables together, create a single view that combines data from these tables.

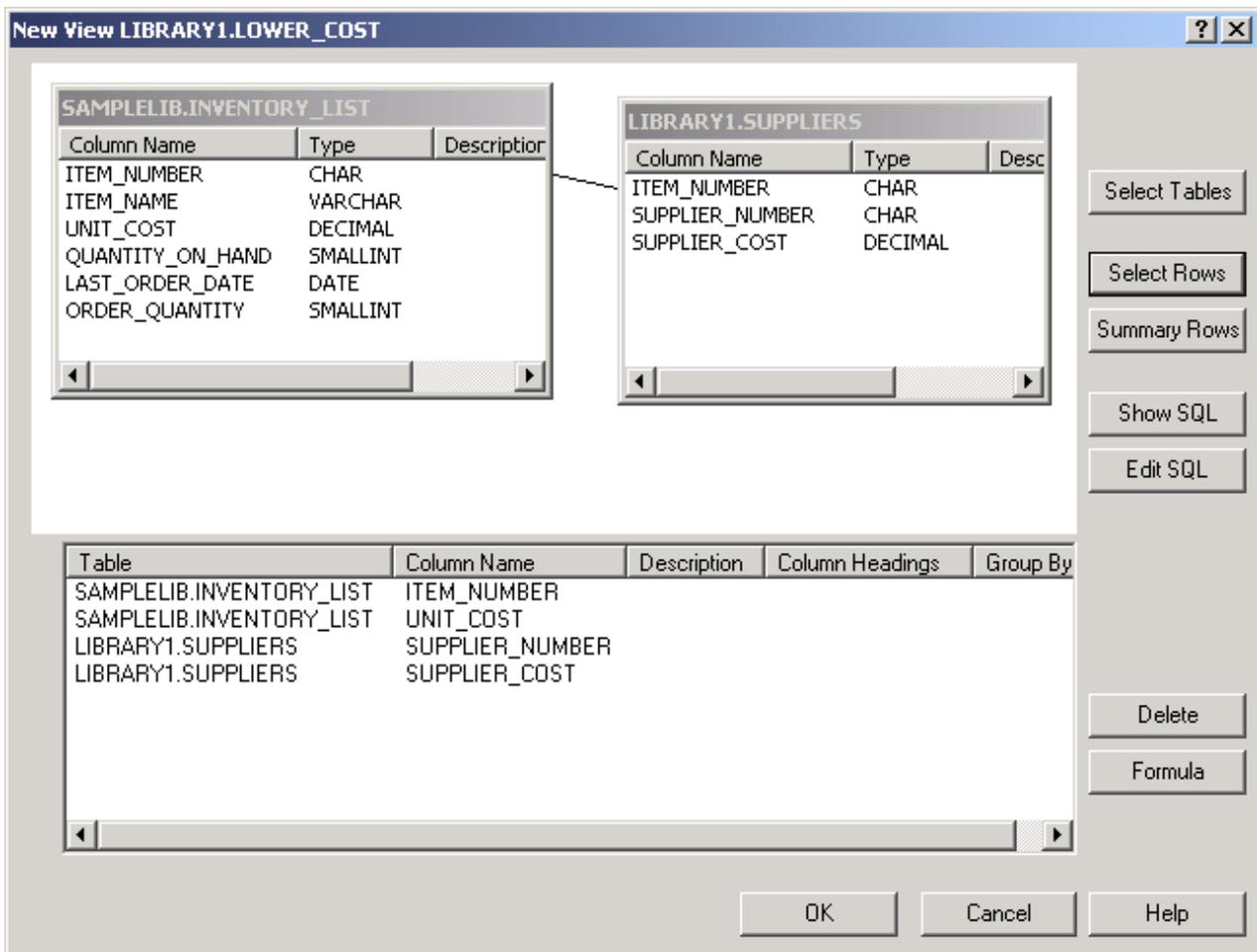
You can create a view that combines information from more than one table by selecting more than one table in the work area of the New View window. You can create a simple view from more than one table by selecting the columns that you want to include from different tables and clicking **OK**. However, this example shows how to create a view that joins information from two different tables and returns only those rows that you want to see, much like using a **WHERE** clause.

In this example, you create a view that contains only those item numbers for suppliers that can supply an item at lower cost than the current unit cost. This requires selecting **ITEM\_NUMBER** and **UNIT\_COST**

from the INVENTORY\_LIST table and joining them with SUPPLIER\_NUMBER and SUPPLIER\_COST from the SUPPLIERS table. A WHERE clause is used to limit the number of rows returned.

To create such a view called LOWER\_COST, follow these steps:

1. Navigate to the LIBRARY1 schema.
2. Right-click **Views** and select **New**.
3. Select **INVENTORY\_LIST** from SAMPLELIB and **SUPPLIERS** from LIBRARY1, and then click **OK**. Both tables should be shown in the work area of the window.
4. Select **ITEM\_NUMBER** and **UNIT\_COST** from INVENTORY\_LIST. Select **SUPPLIER\_NUMBER** and **SUPPLIER\_COST** from SUPPLIERS.
5. To define the join, select **ITEM\_NUMBER** from INVENTORY\_LIST and drag it to **ITEM\_NUMBER** in SUPPLIERS. The Join Properties window opens.
6. In the Join Properties window, select **Return rows with a matching condition (Inner Join)** and click **OK**. A line is drawn from the **ITEM\_NUMBER** column in INVENTORY\_LIST to the **ITEM\_NUMBER** column in SUPPLIERS.
7. Click **Select Rows** to create a WHERE clause for the view. Double-click **LIBRARY1.SUPPLIERS.SUPPLIER\_COST**, and then the **<** operator, and finally, **SAMPLELIB.INVENTORY\_LIST.UNIT\_COST**. When you click the items, they are shown in the window. You can also type them directly.
8. Click **OK** to create the view LOWER\_COST.



**Note:** You can view the SQL used to create this view by selecting **Show SQL**. You can also edit the SQL by selecting **Edit SQL**. Edit SQL starts Run SQL Scripts, where you can edit your SQL statement. Be aware, however, that if you change the SQL, you will need to run the statement from Run SQL Scripts rather than returning to the New View window. If you return to the New View window, your changes are not saved.

To display the contents of this new view, right-click **LOWER\_COST** and select **View Contents**. The rows that you see through this view are only those rows that have a supplier cost that is lower than the unit cost.

ITEM_NUMBER	UNIT_COST	SUPPLIER_NUMBER	SUPPLIER_COST
153047	10.00	9988	8.00
153047	10.00	2424	9.00
229740	1.50	1234	1.00
303476	2.00	3366	1.50
073956	20.00	3366	17.00

## Deleting database objects

After you create database objects on your system, you might want to delete them to save system resources. You need delete authority to perform these tasks.

**Note:** To keep the information in these tables, create a third schema and copy the tables and views to it.

1. To delete the INVENTORY\_LIST table from the LIBRARY1 schema, follow these steps:
  - a. From System i Navigator, expand the system that you want to use.
  - b. Expand **Databases** and the database that you want to work with.
  - c. Expand **Schemas** and select LIBRARY1.
  - d. Select **Tables**.
  - e. Right-click **INVENTORY\_LIST** and select **Delete** or press the Delete key.
  - f. In the Object Deletion Confirmation window, select **Delete**. The INVENTORY\_LIST table is deleted.
2. To delete the SUPPLIERS table from LIBRARY1 and delete LIBRARY1, follow these steps:
  - a. Right-click **SUPPLIERS** and select **Delete** or press the Delete key.
  - b. In the Object Deletion Confirmation window, select **Yes**. A new window opens, indicating that the view LOWER\_COST is dependent on SUPPLIERS. The view should also be deleted.
  - c. Click **Delete**. SUPPLIERS and LOWER\_COST are deleted, and LIBRARY1 is empty.
  - d. Right-click **LIBRARY1** and select **Delete**.
  - e. In the Object Deletion Confirmation window, select **Yes**. LIBRARY1 is deleted.
3. To delete the SAMPLELIB schema, follow these steps:
  - a. Navigate to **SAMPLELIB** in the **Schemas** menu.
  - b. Right-click **SAMPLELIB** and select **Delete**.
  - c. In the Object Deletion Confirmation window, select **Delete**. A new window opens, indicating that SAMPLELIB contains the INVENTORY\_LIST table and the RECENT\_ORDERS view and that RECENT\_ORDERS is dependent on INVENTORY\_LIST.
  - d. Click **Yes** to delete SAMPLELIB, INVENTORY\_LIST, and RECENT\_ORDERS.

---

## System i Navigator database tasks

You can perform many database tasks with System i Navigator, including creating, modifying, and deleting various database objects.

In addition to the tasks described in “Getting started with System i Navigator” on page 2, you can use System i Navigator with DB2 for i5/OS in many other ways. With System i Navigator, you can perform a task in the following ways:

- In System i Navigator, right-click an object and then select the appropriate function from the context menu.
- In the System i Navigator Web interface, select the appropriate function from an object context menu.
- In System i Navigator, use a System i Navigator utility, such as the Run SQL Scripts window.

**Note:** Most of the System i Navigator utilities are not available from the Web interface.

**Related tasks**

Defining public authority using System i Navigator  
 Setting a default public authority for new files

**Related reference**

Examining query optimizer debug messages in the job log

**Related information**

Working with journals

## Database objects creation tasks

This table shows how you can create various database objects in System i Navigator.

*Table 1. Creating database objects*

Tasks	Objects whose context menu supports the task	Available from object context menus in the Web interface	System i Navigator utilities
Create a constraint on a table		N/A, but the task is available from the Table Definition window in the Web interface.	<ul style="list-style-type: none"> <li>• The Table Definition window</li> <li>• A Database Navigator map</li> </ul>
Create a distinct type	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The Distinct Types folder</li> </ul>	Yes	
Create a function	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The Functions folder</li> <li>• A function object</li> </ul>	Yes	
Create a journal	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The Journals folder</li> </ul>	Yes	A Database Navigator map
Create an alias	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The Alias folder</li> <li>• A table object</li> <li>• A table partition</li> <li>• A view object</li> </ul>	Yes	A Database Navigator map

Table 1. Creating database objects (continued)

Tasks	Objects whose context menu supports the task	Available from object context menus in the Web interface	System i Navigator utilities
Create an index	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The Indexes folder</li> <li>• A table object</li> <li>• A table partition</li> </ul>	Yes	<ul style="list-style-type: none"> <li>• A Database Navigator map</li> <li>• Index Advisor</li> <li>• An SQL performance monitor analysis result</li> <li>• Visual Explain</li> </ul>
Create an SQL package	A database folder	No	The Run SQL Scripts window <sup>1</sup>
Create a procedure	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The Procedures folder</li> </ul>	Yes	
Create a schema	The Schemas folder	Yes	
Create a sequence	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The Sequences folder</li> </ul>	Yes	
Create a table	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The Tables folder</li> </ul>	Yes	A Database Navigator map
Create a trigger	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The Triggers folder</li> <li>• A table object</li> <li>• A view object</li> </ul>	Yes	A Database Navigator map
Create a view	<ul style="list-style-type: none"> <li>• A schema object</li> <li>• The All Objects folder</li> <li>• The views folder</li> </ul>	Yes	A Database Navigator map
Create multiple indexes with Run SQL Scripts <sup>2</sup>			Index Advisor

**Note:**

<sup>1</sup> To create an SQL package from the Run SQL Scripts window, select **Connection** → **JDBC Settings** from the menu, and then select the **Enable extended dynamic (SQL package) support** check box on the **Package** tab.

<sup>2</sup> To create multiple indexes with Run SQL Scripts, right-click the advised indexes in the Index Advisor window and select **Show SQL**. This launches the Run SQL Scripts window that contains the CREATE INDEX statements for each index selected.

**Related concepts**

System i Navigator URL parameters and available Web tasks

Creating a library

**Related tasks**

“Creating and using a table” on page 4

A table is a basic database object that is used to store information. After you create a table, you can define columns, create indexes, and add triggers and constraints.

“Creating and using a view” on page 11

You might find that no single table in the database contains all the information that you need. You might also want to give users access to only part of the data in a table. Views provide a way to divide the table so that you deal with only the data that you need.

Adding triggers using System i Navigator

## Database objects operation tasks

This table shows how you can work with various database objects in System i Navigator.

Table 2. Working with database objects

Tasks	Objects whose context menu supports the task	Available from object context menus in the Web interface	System i Navigator utilities
Add a comment to a database object	All database objects	Yes	
Build SQL scripts with SQL Assist			<ul style="list-style-type: none"> <li>• The Run SQL Scripts window</li> <li>• The Create SQL Trigger window</li> <li>• The New Materialized Query Table window</li> </ul>
Capture information about your database with the Database Health Center	A database folder	Yes	
Clear data from a table	<ul style="list-style-type: none"> <li>• A table object</li> <li>• A table partition</li> </ul>	No	
Copy data from a table	<ul style="list-style-type: none"> <li>• A table object</li> <li>• A table partition</li> <li>• A view object</li> </ul>	No	
Delete a database object	Most objects	Yes	A Database Navigator map
Export data from a table or a view	<ul style="list-style-type: none"> <li>• A table object</li> <li>• A table partition</li> <li>• A view object</li> </ul>	Yes	Taskpad
Generate SQL statements for existing database objects	Most objects and object folders	Yes	<ul style="list-style-type: none"> <li>• A Database Navigator map</li> <li>• The Run SQL Scripts window</li> </ul>
Import data into a table	A table object	Yes	Taskpad
Initialize data in a table	<ul style="list-style-type: none"> <li>• A table object</li> <li>• A table partition</li> </ul>	No	
Manage check pending constraints	A database folder	No	
Manage index rebuilds	A database folder	No	

Table 2. Working with database objects (continued)

Tasks	Objects whose context menu supports the task	Available from object context menus in the Web interface	System i Navigator utilities
Modify the definition of a table	A table object	Yes	<ul style="list-style-type: none"> <li>• A Database Navigator map</li> <li>• Visual Explain</li> <li>• An SQL performance monitor analysis report</li> </ul>
Reset usage counts for indexes	<ul style="list-style-type: none"> <li>• One or more index objects</li> <li>• One or more table objects</li> <li>• One or more indexes, constraints, keyed logical files, and keyed physical files in the Show Indexes window</li> </ul>	Yes	The Show Indexes window
Reset usage counts for materialized query tables	One or more materialized query table objects	Yes	The Show Materialized Query Tables window
Run an SQL script <sup>1</sup>	A database folder	No	Taskpad
Select schemas to be displayed in the Schemas folder	The Schemas folder	Yes	Taskpad
Show database objects relations	A table object	Yes	A Database Navigator map
Show indexes for all tables in a schema	The Tables folder	Yes	
Show indexes for a specific table	A table object	Yes	
Show locked rows	<ul style="list-style-type: none"> <li>• A table object</li> <li>• A table partition</li> <li>• A job that locks some rows in the Work Management folder<sup>2</sup></li> </ul>	Yes	
Show materialized query tables for all tables in a schema	The Tables folder	Yes	
Show materialized query tables for all views in a schema	The Views folder	Yes	
Show materialized query tables for a specific table	A table object	Yes	
Show materialized query tables for a specific view	A view object	Yes	
Show partitions of a table	A partitioned table	Yes	
Show transaction information	<ul style="list-style-type: none"> <li>• The Database Transactions folder</li> <li>• The Global Transactions folder</li> </ul>	No	
Specify the JDBC settings			The Run SQL Scripts window

Table 2. Working with database objects (continued)

Tasks	Objects whose context menu supports the task	Available from object context menus in the Web interface	System i Navigator utilities
View lock holders	<ul style="list-style-type: none"> <li>• An alias object</li> <li>• An index object</li> <li>• A table object</li> <li>• A view object</li> </ul>	Yes	
<p><b>Notes:</b></p> <p><sup>1</sup> You can start the Run SQL Scripts window by double-clicking an SQL file.</p> <p><sup>2</sup> To show the locked rows of an object from the Work Management folder, right-click a job in the <b>Work Management folder</b> and select <b>Details</b> → <b>Locked Objects</b>. You can then right-click the locked object and show the locked rows of that object.</p>			

**Related concepts**

System i Navigator URL parameters and available Web tasks

Copying a file

Moving a file

Displaying information with Database Health Center

**Related tasks**

“Building SQL statements with SQL Assist” on page 25

You can build SELECT, INSERT, UPDATE, and DELETE SQL statements interactively in the SQL Assist window of System i Navigator.

“Importing and exporting data” on page 26

System i Navigator provides Import and Export wizards for you to import and export data between files and database tables. These wizards use the Copy from Import File (CPYFRMIMPF) and Copy to Import File (CPYTOIMPF) commands to process the requests.

“Generating SQL for existing objects” on page 25

You can reconstruct the SQL that was used to create existing database objects in the Generate SQL window of System i Navigator.

“Managing check pending constraints” on page 26

You can view and change constraints that the system has placed in a check pending state.

Displaying locked rows using System i Navigator

“Mapping your database” on page 24

Database Navigator is a System i Navigator function that you can use to visually represent the relationships of database objects on your system. This representation is called a *map*. In essence, a Database Navigator map is a snapshot of your database and the relationships that exist among all of the objects in the map.

Displaying attributes of a file using System i Navigator

Reorganizing a table using System i Navigator

“Deleting database objects” on page 15

After you create database objects on your system, you might want to delete them to save system resources. You need delete authority to perform these tasks.

**Related reference**

Managing index rebuilds

Determining unnecessary indexes

## Database performance optimization tasks

This table shows how you can access various tools in System i Navigator to optimize your database performance.

Table 3. Optimizing database performance

Tasks	Objects whose context menu supports the task	Available from object context menus in the Web interface	System i Navigator utilities
Analyze an SQL plan cache event monitor	An SQL plan cache event monitor object	Yes	
Analyze an SQL plan cache snapshot	An SQL plan cache snapshot object	Yes	
Analyze monitor data	An SQL performance monitor object	Yes	The Run SQL Scripts window
Change query attributes for a job	A database folder	Yes	The Run SQL Scripts window
Change the plan score for an access plan			The SQL Plan Cache Statements window
Change the SQL plan cache size threshold <b>Note:</b> The resize is temporary and is lost at the next initial program load.			The SQL Plan Cache Properties window
Compare data collected by performance monitors	One or more SQL performance monitor objects	Yes	
Compare data collected by SQL plan cache event monitors	One or more SQL plan cache event monitor objects	Yes	
Compare data collected by SQL plan cache snapshots	One or more SQL plan cache snapshot objects	Yes	
Create an SQL performance monitor	The SQL Performance Monitors folder	Yes	<ul style="list-style-type: none"> <li>• Health Center–Environment Limits tab</li> <li>• Taskpad</li> <li>• The Run SQL Scripts window</li> <li>• Visual Explain</li> </ul>
Create an SQL plan cache snapshot	The SQL Plan Cache Snapshots folder	Yes	
Create a subset of an existing SQL performance monitor for easier analysis	An SQL performance monitor object (analyze or show statements)	Yes	Visual Explain
Delete a plan from the SQL plan cache			The SQL Plan Cache Statements window

Table 3. Optimizing database performance (continued)

Tasks	Objects whose context menu supports the task	Available from object context menus in the Web interface	System i Navigator utilities
Display indexes advised by the system	<ul style="list-style-type: none"> <li>• A database folder</li> <li>• A schema object</li> <li>• A table object</li> </ul>	Yes	<ul style="list-style-type: none"> <li>• Visual Explain</li> <li>• An SQL performance monitor object</li> <li>• An SQL plan cache snapshot object</li> </ul>
Display SQL information for an object	<ul style="list-style-type: none"> <li>• A function object</li> <li>• An SQL package object</li> <li>• A procedure object</li> <li>• A trigger object</li> <li>• A program object that contains precompiled SQL information in the Integrated File System folder</li> </ul>	Yes	
Display SQL statements collected by an SQL plan cache event monitor	An SQL plan cache event monitor	Yes	
Display SQL statements collected by an SQL plan cache snapshot	An SQL plan cache snapshot object	Yes	
Display SQL statements collected by a performance monitor	An SQL performance monitor object	Yes	
Display the most recent statement and other SQL details for a job	<ul style="list-style-type: none"> <li>• The Databases folder</li> <li>• A job that contains SQL statements in the Work Management folder<sup>2</sup></li> </ul>	Yes	
Manage statistics data	<ul style="list-style-type: none"> <li>• A table object</li> <li>• A table partition</li> </ul>	No	<ul style="list-style-type: none"> <li>• A Database Navigator map</li> <li>• Visual Explain</li> </ul>
Manage the background statistics process	The Databases folder	No	
Remove the longest runs of SQL statements			SQL Plan Cache Show Statements window
Show active jobs that are using the selected SQL statements			The SQL Plan Cache Statements window
Show an SQL statement	<ul style="list-style-type: none"> <li>• A detailed SQL performance monitor object</li> <li>• An SQL plan cache snapshot object</li> <li>• A job that contains SQL statements in the Work Management folder<sup>2</sup></li> </ul>	Yes	<ul style="list-style-type: none"> <li>• The SQL Details for Jobs window</li> <li>• The Compare SQL Performance Data window</li> </ul>
Show the longest runs of SQL statements			The SQL Plan Cache Statements window

Table 3. Optimizing database performance (continued)

Tasks	Objects whose context menu supports the task	Available from object context menus in the Web interface	System i Navigator utilities
Show the picture of a query execution using Visual Explain	<ul style="list-style-type: none"> <li>• SQL performance monitor objects (analyze or compare)</li> <li>• SQL plan cache snapshot objects (analyze or compare)</li> <li>• A job that contains SQL statements in the Work Management folder<sup>1</sup></li> </ul>	No	<ul style="list-style-type: none"> <li>• The Run SQL Scripts window</li> <li>• The Show statements window</li> <li>• The SQL Details for Jobs window</li> <li>• The SQL Plan Cache Statements window</li> </ul>
Show the user history for the selected SQL statements			The SQL Plan Cache Statements window
Start an SQL plan cache event monitor to log plans that are removed from the plan cache	The SQL Plan Cache Event Monitors folder	Yes	
Start the statistics advisor			<ul style="list-style-type: none"> <li>• Visual Explain</li> <li>• An SQL performance monitor analysis report</li> <li>• An SQL plan cache snapshot analysis report</li> </ul>
View the SQL plan cache	<ul style="list-style-type: none"> <li>• The SQL Plan Cache folder<sup>1</sup></li> <li>• An index object</li> <li>• An advised index in the Index Advisor window</li> <li>• A condensed advised index in the Condensed Index Advice window</li> </ul>	Yes	Index Advisor

**Notes:**

<sup>1</sup> To view the SQL plan cache, right-click the **SQL Plan Cache** folder and select **Show Statements**.

<sup>2</sup> To show the SQL statement that a job contains from the Work Management folder, right-click a job in the **Work Management** folder and select **Details** → **SQL**.

**Related concepts**

System i Navigator URL parameters and available Web tasks

Memory Resident Database Monitor: DDS

**Related reference**

Viewing the plan cache with System i Navigator

Viewing the implementation of your queries with Visual Explain

Displaying index advisor information

Collecting statistics with the Statistics Manager

## Mapping your database

Database Navigator is a System i Navigator function that you can use to visually represent the relationships of database objects on your system. This representation is called a *map*. In essence, a Database Navigator map is a snapshot of your database and the relationships that exist among all of the objects in the map.

Using Database Navigator, you can explore the complex relationships of your database objects using a graphical representation that presents the tables in your database, the relationships between tables, and indexes and constraints that are attached to tables. The primary workspace for Database Navigator is a window that is divided into several main areas. The map is displayed in the right pane. You can perform a variety of tasks by right-clicking an object. The Locator pane is found on the left side of the window. You can use this pane to locate specific objects to include in the map or to specify a type of object to include in the map.

To access Database Navigator maps, expand the system name, **Databases**, and the database that you want to use.

To display a list of existing maps in the right pane, click **Database Navigator Maps**.

To create a new map, right-click **Database Navigator Maps** and select **New** → **Map**.

### Tips for using Database Navigator:

- To change the size of either side of the window, drag the bar (splitter) that separates the two sides.
- Be sure to right-click the objects on both the left and right sides of the window. The right-click menus give you quick access to common functions.
- To quickly open a schema and display the objects in it, double-click the schema.
- To access the various Database Navigator commands, use either the Menu bar or the Toolbar.

## Querying your database by running SQL scripts

You can create, edit, run, and troubleshoot scripts of SQL statements in the Run SQL Scripts window of System i Navigator. When you finish working with the scripts, you can save the statements to your PC.

To open the Run SQL Scripts window, expand the system name and **Databases** and right-click the database to which you want to connect.

You can use the Examples list to build your scripts, manually create your statement, retrieve the SQL for an existing object using the generate SQL function, or build a script using SQL Assist.

You can check the syntax of your SQL by clicking **Check Syntax**. Additional ways of debugging your programs and scripts include debugging messages in the job log and starting System i5™ Debugger. When syntax checking is complete, you can save the script by selecting **Save** from the **File** menu.

To run an SQL script, select one of the following options from the **Run** menu:

- **All**: Run your SQL script from the beginning to the end. If an error occurs and the **Stop on Error** option is turned on, the program stops and the statement where the error occurred remains selected.
- **From Selected**: Start your SQL script from the first statement that is selected or from the current cursor position to the end of the script.
- **Selected**: Run the statements that are selected.

The results are added to the end of the **Messages** tab. If the **Smart Statement Selection** option on the **Options** menu is not checked, the text that is selected is run as a single SQL statement.

### Related concepts

Using interactive SQL

### Related tasks

“Creating a view that combines data from multiple tables” on page 41

A view that combines data from multiple tables enables you to show relevant information in multiple tables together. You can create a view that combines data from two or more tables by naming more than one table in the FROM clause.

## Stopping SQL scripts

You can stop or cancel a running SQL script from System i Navigator.

To stop or cancel SQL scripts that are running, select one of the following options from the **Run** menu:

- **Stop After Current:** Stop running the SQL script after the currently running statement ends.
- **Cancel Request:** Request that the system cancel the current SQL statement. However, because not all SQL statements can be canceled, an SQL statement might continue to completion even after this option is used. SQL statements that have already completed host processing before Cancel Request is pressed also continue to completion. For example, SELECT statements that have already completed query processing but have not returned results to the client typically cannot be canceled.

## Viewing the job log

The job log contains the messages that are related to your job. You can view the job log from System i Navigator.

To see query optimizer and other database debugging messages, follow these steps:

1. Select **Include Debug Messages in Job Log** from the **Options** menu.
2. Run the statement again.
3. If the Job Log dialog box is open when you do this, refresh the view to see new messages.

To view the job log, select **Job Log** from the **View** menu.

The job log is not cleared when **Clear Run History** is used, so you can use the job log to see messages that are no longer in the Output pane.

- | To view job details, select **Job Details** from the **View** menu.

## Generating SQL for existing objects

You can reconstruct the SQL that was used to create existing database objects in the Generate SQL window of System i Navigator.

You can generate SQL for most database objects. Additionally, if you generate SQL for a table that has constraints or triggers associated with it, the SQL is generated for those constraints or triggers as well. You can generate SQL for one object or many objects at a time. You can also send the generated SQL to the Run SQL Scripts window for running or editing, or you can write the generated SQL directly to a database or PC file.

To generate SQL for an object, right-click the object and select **Generate SQL**.

You can also open the Generate SQL window by selecting **Insert Generated SQL** from the **Edit** menu in the Run SQL Scripts window.

## Building SQL statements with SQL Assist

You can build SELECT, INSERT, UPDATE, and DELETE SQL statements interactively in the SQL Assist window of System i Navigator.

To start the SQL Assist window, follow these steps:

1. Select **SQL Assist** from the **Edit** menu in the Run SQL Scripts window. In the SQL Assist window, you can choose tables to work with and build selection criteria. The statement is built in the bottom portion of the window.
2. Click **OK** to return the statement that you built to the Run SQL Scripts window.
3. Edit, run, and save your statement.

## Starting System i5 Debugger

System i5 Debugger provides a graphical user debugging environment on your system. You can use System i5 Debugger to debug and test programs that run on your system, including the programs that run in i5/OS Portable Application Solutions Environment (PASE).

To start System i5 Debugger from the Run SQL Scripts window, select **Debugger** from the **Run** menu.

### Related concepts

System i5 Debugger

## Managing check pending constraints

You can view and change constraints that the system has placed in a check pending state.

*Check pending* refers to a state in which a mismatch exists either between a parent and foreign key in the case of a referential constraint or between the column value and the check constraint definition in the case of a check constraint.

To view and change constraints that have been placed in a check pending state, follow these steps:

1. Expand the system name and **Databases**.
2. Right-click the database that you want to use and select **Manage Check Pending Constraints**. From this interface, you can view the definition of each constraint and the rows that are in violation of the constraint rules.
3. Select the constraint that you want to work with and then select **Edit Check Pending Constraint** from the **File** menu.
4. Change or delete the rows that are in violation.

### Related concepts

Check pending status in referential constraints

## Importing and exporting data

System i Navigator provides Import and Export wizards for you to import and export data between files and database tables. These wizards use the Copy from Import File (CPYFRMIMPF) and Copy to Import File (CPYTOIMPF) commands to process the requests.

Here are the files and database tables that you can import data from or export data to:

- Integrated file systems files
- Source physical files
- Program-described files
- Database tables with a single nonnumeric column that is not data type LOB

System i Navigator refers to an import file as a data file.

To start the Import or Export wizard, follow these steps:

1. From System i Navigator, expand the system that you want to use.
2. Expand **Databases**.
3. Expand the database and schema that you want to work with.
4. Click the **Tables** container.

5. If you want to import data from a data file, right-click the table that you want to import data to and select **Data** → **Import**. If you want to export data in a table to a file, right-click the table that you want to export data to and select **Data** → **Export**.

#### **Related tasks**

Copying between different systems

#### **Related reference**

Copy From Import File (CPYFRMIMPF) command

Copy To Import File (CPYTOIMPF) command

---

## **Getting started with SQL**

SQL is a standardized language for defining and manipulating data in a relational database. You can create and work with schemas, tables, and views by using SQL statements in interactive SQL directly.

The syntax for each of the SQL statements used in these topics is described in detail and descriptions of how to use SQL statements and clauses in more complex situations are provided in the DB2 for i5/OS SQL reference topic collection.

In these topics, the examples use the interactive SQL interface to show the use of SQL statements. Each SQL interface provides methods for defining tables, views, and other objects; for updating the objects; and for reading data from the objects.

First, start interactive SQL:

1. Type STRSQL NAMING(\*SQL).
2. Press Enter.

When the Enter SQL Statements display appears, you are ready to start typing SQL statements.

If you are reusing an existing interactive SQL session, make sure that you set the naming mode to **SQL naming**. You can specify this on the F13 (Services) panel, option 1 (Change session attributes).

#### **Related reference**

SQL programming

## **Creating a schema**

A *schema* (also known as a *collection*) is a basic object in which tables, views, indexes, and packages are placed. To create a schema, use the CREATE SCHEMA statement.

To create a schema named SAMPLECOLL, follow these steps:

1. Enter the following SQL statement on the Enter SQL Statements display: CREATE SCHEMA SAMPLECOLL.
2. Press Enter.

**Note:** Running this statement causes several objects to be created and takes several seconds.

After you have successfully created a schema, you can create tables, views, and indexes in it. Tables, views, and indexes can also be created in libraries instead of schemas.

#### **Related concepts**

Creating database objects

#### **Related reference**

CREATE SCHEMA

## Creating and using a table

You can use the CREATE TABLE statement to create a table, to define the physical attributes of the columns in a table, and to define constraints to restrict the values that are allowed in a table.

When creating a table, you need to understand the concepts of null value and default value. A *null value* indicates the absence of a column value for a row. It is not the same as a value of zero or all blanks. It means unknown. A null value is not equal to any value, not even to other null values. If a column does not allow the null value, a value must be assigned to the column, either a default value or a user-supplied value.

A *default value* is assigned to a column when a row is added to a table and no value is specified for that column. If a specific default value was not defined for a column, the system default value is used.

You are going to create a table to maintain information about the current inventory of a business. The table contains information about the items kept in the inventory, their cost, quantity currently on hand, the last order date, and the number last ordered. The item number is a required value. It cannot be null. The item name, quantity on hand, and order quantity have user-supplied default values. The last order date and quantity ordered allow null values.

You also need to create a second table. This table contains information about suppliers of your inventory items, which items they supply, and the cost of the item from that supplier.

1. Create the first table named INVENTORY\_LIST:
  - a. On the Enter SQL Statements display, type CREATE TABLE and press F4 (Prompt). The following display is shown (with the input areas not yet filled in).

Specify CREATE TABLE Statement

Type information, press Enter.

Table . . . . . **INVENTORY\_LIST** \_\_\_\_\_ Name  
 Collection . . . . . **SAMPLECOLL** \_\_\_\_\_ Name, F4 for list

Nulls: 1=NULL, 2=NOT NULL, 3=NOT NULL WITH DEFAULT

Column	FOR Column	Type	Length	Scale	Nulls
<b>ITEM_NUMBER</b> _____	_____	<b>CHAR</b> _____	<b>6</b> _____	_____	<b>2</b>
<b>ITEM_NAME</b> _____	_____	<b>VARCHAR</b> _____	<b>20</b> _____	_____	<b>3</b>
<b>UNIT_COST</b> _____	_____	<b>DECIMAL</b> _____	<b>8</b> _____	<b>2</b> _____	<b>3</b>
<b>QUANTITY_ON_HAND</b> _____	_____	<b>SMALLINT</b> _____	_____	_____	<b>1</b>
<b>LAST_ORDER_DATE</b> _____	_____	<b>DATE</b> _____	_____	_____	<b>1</b>
_____	_____	_____	_____	_____	<b>3</b>

Bottom

Table CONSTRAINT . . . . . **N** Y=Yes, N=No  
 Distributed Table . . . . . **N** Y=Yes, N=No

F3=Exit F4=Prompt F5=Refresh F6=Insert line F10=Copy line  
 F11=Display more attributes F12=Cancel F14=Delete line F24=More keys

- b. Type the table name INVENTORY\_LIST and schema name SAMPLECOLL at the **Table** and **Collection** prompts, as shown.
- c. Each column you want to define for the table is represented by an entry in the list on the lower part of the display. For each column, type the name of the column, the data type of the column, its length and scale, and the null attribute.
- d. Press F11 (Display more attributes) to see more attributes that can be specified for the columns. This is where a default value can be specified.

Specify CREATE TABLE Statement

Type information, press Enter.

Table . . . . . INVENTORY\_LIST \_\_\_\_\_ Name  
Collection . . . . . SAMPLECOLL\_\_ \_\_\_\_\_ Name, F4 for list

Data: 1=BIT, 2=SBCS, 3=MIXED, 4=CCSID

Column	Data	Allocate	CCSID	CONSTRAINT	Default
ITEM_NUMBER _____	-	_____	_____	N	_____
ITEM_NAME _____	-	_____	_____	N	'***UNKNOWN***' _____
UNIT_COST _____	-	_____	_____	N	_____
QUANTITY_ON_HAND _____	-	_____	_____	N	NULL _____
LAST_ORDER_DATE _____	-	_____	_____	N	_____
ORDER_QUANTITY _____	-	_____	_____	N	20 _____
_____	-	_____	_____	-	_____

Bottom

Table CONSTRAINT . . . . . N Y=Yes, N=No  
Distributed Table . . . . . N Y=Yes, N=No

F3=Exit F4=Prompt F5=Refresh F6=Insert line F10=Copy line  
F11=Display more attributes F12=Cancel F14=Delete line F24=More keys

**Note:** Another way of entering column definitions is to press F4 (Prompt) with your cursor on one of the column entries in the list. A display that shows all of the attributes for defining a single column appears.

- e. When all the values have been entered, press Enter to create the table. The Enter SQL Statements display is shown again with a message indicating that the table has been created.

**Note:** You can type this CREATE TABLE statement on the Enter SQL Statements display as follows:

```
CREATE TABLE SAMPLECOLL.INVENTORY_LIST
(ITEM_NUMBER CHAR(6) NOT NULL,
ITEM_NAME VARCHAR(20) NOT NULL WITH DEFAULT '***UNKNOWN***',
UNIT_COST DECIMAL(8,2) NOT NULL WITH DEFAULT,
QUANTITY_ON_HAND SMALLINT DEFAULT NULL,
LAST_ORDER_DATE DATE,
ORDER_QUANTITY SMALLINT DEFAULT 20)
```

2. Create a second table named SUPPLIERS. There are two methods you can use:
  - a. Type the following command directly on the Enter SQL Statements display.
  - b. Press F4 (Prompt) to use the interactive SQL displays to create the definition.

```
CREATE TABLE SAMPLECOLL.SUPPLIERS
(SUPPLIER_NUMBER CHAR(4) NOT NULL,
ITEM_NUMBER CHAR(6) NOT NULL,
SUPPLIER_COST DECIMAL(8,2))
```

**Related concepts**

Altering and managing database objects

Creating database objects

**Related reference**

INSERT

**Using the LABEL ON statement**

Normally, the column name is used as the column heading when the output of a SELECT statement is shown in interactive SQL. By using the LABEL ON statement, you can create a more descriptive label for the column name.

Because you run your examples in interactive SQL, you use the LABEL ON statement to change the column headings. Even though the column name is descriptive, it is easier to read if the column heading shows each part of the name on a single line. It also allows you to see more columns of data on a single display.

To change the labels for your columns, follow these steps:

1. Enter LABEL ON COLUMN on the Enter SQL Statements display.
2. Press F4 (Prompt). The following display appears.

```

Specify LABEL ON Statement

Type choices, press Enter.

Label on . . . . 2                                1=Table or view
                                                    2=Column
                                                    3=Package
                                                    4=Alias

Table or view   INVENTORY_LIST_____           Name, F4 for list
Collection . .  SAMPLECOLL__                   Name, F4 for list

Option . . . . . 1                                1=Column heading
                                                    2=Text

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F20=Display entire name
F21=Display statement

```

3. Type the name of the table and schema that contains the columns for which you want to add labels.
4. Press Enter. The following display is shown, prompting you for each of the columns in the table.

```

Specify LABEL ON Statement

Type information, press Enter.

Column          Column Heading
.....+.....1.....+.....2.....+.....3.....+.....4.....+.....5.....
ITEM_NUMBER     'ITEM          NUMBER' _____
ITEM_NAME       'ITEM          NAME' _____
UNIT_COST       'UNIT          COST' _____
QUANTITY_ON_HAND 'QUANTITY      ON          HAND' _____
LAST_ORDER_DATE 'LAST          ORDER          DATE' _____
ORDER_QUANTITY  'NUMBER        ORDERED' _____

Bottom

F3=Exit  F5=Refresh  F6=Insert line  F10=Copy line  F12=Cancel
F14=Delete line  F19=Display system column names  F24=More keys

```

5. Type the column heading for each of the columns. Column headings are defined in 20-character sections. Each section is displayed on a different line when the output of a SELECT statement is shown. The ruler across the top of the column heading entry area can be used to easily space the headings correctly.
6. Press Enter.

The following message indicates that the LABEL ON statement was successful:  
 LABEL ON for INVEN00001 in SAMPLECOLL completed.

The table name in the message is the system table name for this table, not the name that was actually specified in the statement. DB2 for i5/OS maintains two names for tables with names longer than 10 characters.

**Note:** The LABEL ON statement can also be typed directly on the Enter SQL Statements display as follows:

```
LABEL ON SAMPLECOLL.INVENTORY_LIST
(ITEM_NUMBER      IS 'ITEM      NUMBER ',
 ITEM_NAME        IS 'ITEM      NAME ',
 UNIT_COST        IS 'UNIT      COST ',
 QUANTITY_ON_HAND IS 'QUANTITY  ON      HAND ',
 LAST_ORDER_DATE  IS 'LAST      ORDER   DATE ',
 ORDER_QUANTITY   IS 'NUMBER    ORDERED  ')
```

**Related reference**

CREATE TABLE

### Inserting information into a table

After you create a table, you can insert or add information (data) into the table by using the SQL INSERT statement.

To insert information into a table, follow these steps:

1. On the Enter SQL Statements display, type INSERT and press F4 (Prompt). The Specify INSERT Statement display is shown.

Specify INSERT Statement

Type choices, press Enter.

INTO table . . . . .	INVENTORY_LIST _____	Name, F4 for list
Collection . . . . .	SAMPLECOLL__	Name, F4 for list

Select columns to insert

INTO . . . . .	Y	Y=Yes, N=No
Insertion method . . . . .	1	1=Input VALUES 2=Subselect

Type choices, press Enter.

WITH isolation level . . .	1	1=Current level, 2=NC (NONE) 3=UR (CHG), 4=CS, 5=RS (ALL) 6=RR
----------------------------	---	--

F3=Exit F4=Prompt F5=Refresh F12=Cancel F20=Display entire name  
 F21=Display statement

2. Type the table name and schema name in the input fields as shown.
3. Change the **Select columns to insert INTO** prompt to **Yes**.
4. Press Enter to see the display where the columns you want to insert values into can be selected.

```

Specify INSERT Statement

Type sequence numbers (1-999) to make selections, press Enter.

Seq  Column                Type                Length  Scale
1__  ITEM_NUMBER            CHARACTER           6
2__  ITEM_NAME              VARCHAR             20
3__  UNIT_COST              DECIMAL            8      2
4__  QUANTITY_ON_HAND      SMALLINT           4
___  LAST_ORDER_DATE       DATE
___  ORDER_QUANTITY        SMALLINT           4

Bottom

F3=Exit   F5=Refresh   F12=Cancel   F19=Display system column names
F20=Display entire name   F21=Display statement

```

In this example, insert four of the columns. Allow the other columns to have their default values inserted. The sequence numbers on this display indicate the order that the columns and values are listed in the INSERT statement.

- 5. Press Enter to show the display where values for the selected columns can be typed.

```

Specify INSERT Statement

Type values to insert, press Enter.

Column                Value
ITEM_NUMBER           '153047'
ITEM_NAME              'Pencils, red'
UNIT_COST              10.00
QUANTITY_ON_HAND      25

Bottom

F3=Exit   F5=Refresh   F6=Insert line   F10=Copy line   F11=Display type
F12=Cancel   F14=Delete line   F15=Split line   F24=More keys

```

**Note:** To see the data type and length for each of the columns in the insert list, press F11 (Display type). This shows a different view of the insert values display, providing information about the column definition.

- 6. Type the values to be inserted for all of the columns and press Enter. A row containing these values is added to the table. The values for the columns that were not specified have a default value inserted. For LAST\_ORDER\_DATE it is the null value because no default was provided and the column allows the null value. For ORDER\_QUANTITY it is 20, the value specified as the default value on the CREATE TABLE statement.
- 7. Type the INSERT statement on the Enter SQL Statements display as follows:  
**INSERT INTO** SAMPLECOLL.INVENTORY\_LIST (ITEM\_NUMBER, ITEM\_NAME, UNIT\_COST, QUANTITY\_ON\_HAND)  
**VALUES** ('153047', 'Pencils, red', 10.00, 25)
- 8. To add the next row to the table, press F9 (Retrieve) on the Enter SQL Statements display. This copies the previous INSERT statement to the typing area. You can either type over the values from the previous INSERT statement or press F4 (Prompt) to use the Interactive SQL displays to enter data.
- 9. Continue using the INSERT statement to add the following rows to the table.

Values not shown in the following chart should not be inserted so that the default is used. In the INSERT statement column list, specify only the column names for which you want to insert a value. For example, to insert the third row, specify only ITEM\_NUMBER and UNIT\_COST for the column names and only the two values for these columns in the VALUES list.

ITEM_NUMBER	ITEM_NAME	UNIT_COST	QUANTITY_ON_HAND
153047	Pencils, red	10.00	25
229740	Lined tablets	1.50	120
544931		5.00	
303476	Paper clips	2.00	100
559343	Envelopes, legal	3.00	500
291124	Envelopes, standard		
775298	Chairs, secretary	225.00	6
073956	Pens, black	20.00	25

Add the following rows to the SAMPLECOLL.SUPPLIERS table.

SUPPLIER_NUMBER	ITEM_NUMBER	SUPPLIER_COST
1234	153047	10.00
1234	229740	1.00
1234	303476	3.00
9988	153047	8.00
9988	559343	3.00
2424	153047	9.00
2424	303476	2.50
5546	775298	225.00
3366	303476	1.50
3366	073956	17.00

The sample schema now contains two tables with several rows of data in each.

## Getting information from a single table

After inserting information into a table, you can use the SELECT statement to display some or all the information in the table.

The SELECT statement is the most complex of all SQL statements. This statement is composed of the following main clauses:

1. The SELECT clause, which specifies those columns that contain the data.
2. The FROM clause, which specifies the table or tables that contain the columns with the data.
3. The WHERE clause, which supplies conditions that determine which rows of data are retrieved.

In addition to these main clauses, several other clauses that affect the final form of returned data are described in the SQL programming and DB2 for i5/OS SQL reference topic collections.

1. To see the values that you inserted into the INVENTORY\_LIST table, type SELECT and press F4 (Prompt). The following display is shown.

```

Specify SELECT Statement

Type SELECT statement information. Press F4 for a list.

FROM tables . . . . . SAMPLECOLL.INVENTORY_LIST_____
SELECT columns . . . . . *_____
WHERE conditions . . . . . _____
GROUP BY columns . . . . . _____
HAVING conditions . . . . . _____
ORDER BY columns . . . . . _____
FOR UPDATE OF columns . . . . . _____

Bottom

Type choices, press Enter.

DISTINCT rows in result table . . . . . N Y=Yes, N=No
UNION with another SELECT . . . . . N Y=Yes, N=No
Specify additional options . . . . . N Y=Yes, N=No

F3=Exit      F4=Prompt  F5=Refresh  F6=Insert line  F9=Specify subquery
F10=Copy line F12=Cancel F14>Delete line F15=Split line F24=More keys

```

2. Type the table name in the **FROM tables** field on the display. To select all columns from the table, type \* for the **SELECT columns** field on the display.
3. Press Enter and the statement runs to select all of the data for all of the columns in the table. The following output is shown.

```

Display Data
Data width . . . . . : 71
Position to line . . . . . Shift to column . . . . .
.....1.....2.....3.....4.....5.....6.....7.
ITEM  ITEM          UNIT  QUANTITY  LAST  NUMBER
NUMBER NAME          COST   ON        ORDER  ORDERED
      HAND        DATE
153047 Pencils, red      10.00   25   -      20
229740 Lined tablets     1.50  120   -      20
544931 ***UNKNOWN***    5.00   -    -      20
303476 Paper clips       2.00  100   -      20
559343 Envelopes, legal  3.00  500   -      20
291124 Envelopes, standard .00   -    -      20
775298 Chairs, secretary 225.00  6    -      20
073956 Pens, black      20.00  25   -      20
***** End of data *****

F3=Exit      F12=Cancel  F19=Left    F20=Right   F21=Split

```

The column headings that were defined with the LABEL ON statement are shown. The ITEM\_NAME column for the third entry contains the default value that was specified in the CREATE TABLE statement. The QUANTITY\_ON\_HAND column contains a null value for the rows where no value was inserted. The LAST\_ORDER\_DATE column contains all null values because that column is not in any of the INSERT statements and the column was not defined to have a default value. Similarly, the ORDER\_QUANTITY column contains the default value for all rows.

This statement can be entered on the Enter SQL Statements display as:

```

SELECT *
FROM SAMPLECOLL.INVENTORY_LIST

```

4. To limit the number of columns returned by the SELECT statement, the columns you want to see must be specified. To restrict the number of output rows returned, the WHERE clause is used. To see only the items that cost more than 10 dollars, and only have the values for the columns ITEM\_NUMBER, UNIT\_COST, and ITEM\_NAME returned, type SELECT and press F4 (Prompt). The Specify SELECT Statement display is shown.

```

Specify SELECT Statement

Type SELECT statement information. Press F4 for a list.

FROM tables . . . . . SAMPLECOLL.INVENTORY_LIST
SELECT columns . . . . . ITEM_NUMBER, UNIT_COST, ITEM_NAME
WHERE conditions . . . . . UNIT_COST > 10.00
GROUP BY columns . . . . .
HAVING conditions . . . . .
ORDER BY columns . . . . .
FOR UPDATE OF columns . . . . .

Bottom

Type choices, press Enter.

DISTINCT rows in result table . . . . . N Y=Yes, N=No
UNION with another SELECT . . . . . N Y=Yes, N=No
Specify additional options . . . . . N Y=Yes, N=No

F3=Exit      F4=Prompt  F5=Refresh  F6=Insert line  F9=Specify subquery
F10=Copy line F12=Cancel F14=Delete line F15=Split line F24=More keys

```

Although only one line is initially shown for each prompt on the Specify SELECT Statement display, you can add more lines to any of the input areas on the top part of the display by pressing F6 (Insert line). F6 can be used if more columns need to be entered in the SELECT columns list or if a longer, more complex WHERE condition is needed.

- Complete the information on the display, as shown.
- Press Enter to run the SELECT statement. The following output is shown.

```

Display Data
Data width . . . . . : 41
Position to line . . . . . Shift to column . . . . .
.....1.....2.....3.....4.
ITEM          UNIT   ITEM
NUMBER       COST   NAME
775298       225.00 Chairs, secretary
073956       20.00 Pens, black
***** End of data *****

F3=Exit      F12=Cancel  F19=Left    F20=Right   F21=Split

```

The only rows returned are those whose data values satisfy the condition specified in the WHERE clause. Furthermore, the only data values returned are from the columns you explicitly specified in the SELECT clause. Data values of columns other than those explicitly identified are not returned.

This statement can be entered on the Enter SQL Statements display as:

```

SELECT ITEM_NUMBER,UNIT_COST,ITEM_NAME
FROM SAMPLECOLL.INVENTORY_LIST
WHERE UNIT_COST > 10.00

```

**Getting information from multiple tables**

With SQL, you can get information from columns in more than one table. This operation is called a *join* operation.

In SQL, a join operation is specified by placing the names of those tables that you want to join in the same FROM clause of a SELECT statement.

Suppose that you want to see a list of all the suppliers and the item numbers and item names for their supplied items. The item name is not in the SUPPLIERS table; it is in the INVENTORY\_LIST table. Using the common column, ITEM\_NUMBER, you can see all of the columns as if they were from a single table.

Whenever the same column name exists in two or more tables being joined, the column name must be qualified by the table name to specify which column is being referenced. In this SELECT statement, the column name ITEM\_NUMBER is defined in both tables, so it needs to be qualified by the table name. If the columns have different names, no qualification is needed.

To perform this join operation, enter the following SELECT statement by typing it directly on the Enter SQL Statements display or by prompting:

```
SELECT SUPPLIER_NUMBER, SAMPLECOLL.INVENTORY_LIST.ITEM_NUMBER, ITEM_NAME
      FROM SAMPLECOLL.SUPPLIERS, SAMPLECOLL.INVENTORY_LIST
      WHERE SAMPLECOLL.SUPPLIERS.ITEM_NUMBER
            = SAMPLECOLL.INVENTORY_LIST.ITEM_NUMBER
```

If you use prompting, you need to type both table names on the FROM tables input line.

Another way to enter the same statement is to use a correlation name. A correlation name provides another name for a table name to use in a statement. A correlation name must be used when the table names are the same. It can be specified by following each table name in the FROM list. The previous statement can be rewritten as:

```
SELECT SUPPLIER_NUMBER, Y.ITEM_NUMBER, ITEM_NAME
      FROM SAMPLECOLL.SUPPLIERS X, SAMPLECOLL.INVENTORY_LIST Y
      WHERE X.ITEM_NUMBER = Y.ITEM_NUMBER
```

In this example, SAMPLECOLL.SUPPLIERS is given a correlation name of X and SAMPLECOLL.INVENTORY\_LIST is given a correlation name of Y. The names X and Y are then used to qualify the ITEM\_NUMBER column name.

Running this example returns the following output.

```

                                Display Data
                                Data width . . . . . : 45
Position to line . . . . .      Shift to column . . . . .
...+...1...+...2...+...3...+...4...+
SUPPLIER_NUMBER  ITEM   ITEM
                NUMBER  NAME
1234            153047 Pencils, red
1234            229740 Lined tablets
1234            303476 Paper clips
9988            153047 Pencils, red
9988            559343 Envelopes, legal
2424            153047 Pencils, red
2424            303476 Paper clips
5546            775298 Chairs, secretary
3366            303476 Paper clips
3366            073956 Pens, black
***** End of data *****

F3=Exit      F12=Cancel    F19=Left     F20=Right    F21=Split
```

**Note:** Because no ORDER BY clause was specified for the query, the order of the rows returned by your query may be different.

The data values in the result table represent a composite of the data values contained in the two tables INVENTORY\_LIST and SUPPLIERS. This result table contains the supplier number from the SUPPLIER table and the item number and item name from the INVENTORY\_LIST table. Any item numbers that do not appear in the SUPPLIER table are not shown in this result table. The results are not guaranteed to be in any order unless the ORDER BY clause is specified for the SELECT statement. Because you did not change any column headings for the SUPPLIER table, the SUPPLIER\_NUMBER column name is used as the column heading.

The following example shows how to use ORDER BY to guarantee the order of the rows. The statement first sorts the result table by the SUPPLIER\_NUMBER column. Rows with the same value for SUPPLIER\_NUMBER are sorted by their ITEM\_NUMBER.

```
SELECT SUPPLIER_NUMBER,Y.ITEM_NUMBER,ITEM_NAME
FROM SAMPLECOLL.SUPPLIERS X,SAMPLECOLL.INVENTORY_LIST Y
WHERE X.ITEM_NUMBER = Y.ITEM_NUMBER
ORDER BY SUPPLIER_NUMBER,Y.ITEM_NUMBER
```

Running the previous statement produces the following output.

```

                                Display Data
                                Data width . . . . . :    45
Position to line . . . . .      Shift to column . . . . .
....+....1....+....2....+....3....+....4....+
SUPPLIER_NUMBER  ITEM      ITEM
                NUMBER  NAME
    1234         153047  Pencils, red
    1234         229740  Lined tablets
    1234         303476  Paper clips
    2424         153047  Pencils, red
    2424         303476  Paper clips
    3366         073956  Pens, black
    3366         303476  Paper clips
    5546         775298  Chairs, secretary
    9988         153047  Pencils, red
    9988         559343  Envelopes, legal
***** End of data *****
F3=Exit      F12=Cancel    F19=Left    F20=Right   F21=Split

```

**Related reference**

DB2 for i5/OS SQL reference

**Changing information in a table**

The SQL UPDATE statement changes the values in some or all of the columns of a table. If you want to limit the number of rows being changed during the processing of a single statement, use the WHERE clause with the UPDATE statement.

If you do not specify the WHERE clause, all of the rows in the specified table are changed. However, if you use the WHERE clause, the system changes only the rows that satisfy the specified conditions.

Suppose that you want to place an order for more paper clips today.

1. To update the LAST\_ORDER\_DATE and ORDER\_QUANTITY columns for item number 303476, type UPDATE and press F4 (Prompt). The Specify UPDATE Statement display is shown.

Specify UPDATE Statement

Type choices, press Enter.

Table . . . . .	INVENTORY_LIST_____	Name, F4 for list
Collection . . . . .	SAMPLECOLL__	Name, F4 for list
Correlation . . . . .	_____	Name

F3=Exit F4=Prompt F5=Refresh F12=Cancel F20=Display full names  
F21=Display statement

2. Type the table name and schema name, as shown on the previous display.
3. Press Enter. The display is shown again with the list of columns in the table.

Specify UPDATE Statement

Type choices, press Enter.

Table . . . . .	INVENTORY_LIST_____	Name, F4 for list
Collection . . . . .	SAMPLECOLL__	Name, F4 for list
Correlation . . . . .	_____	Name

Type information, press Enter.

Column	Value
ITEM_NUMBER	_____
ITEM_NAME	_____
UNIT_COST	_____
QUANTITY_ON_HAND	_____
LAST_ORDER_DATE	<b>CURRENT DATE</b> _____
ORDER_QUANTITY	<b>50</b> _____

Bottom

F3=Exit F4=Prompt F5=Refresh F6=Insert line F10=Copy line  
F11=Display type F12=Cancel F14=Delete line F24=More keys

4. Specify CURRENT DATE in the LAST\_ORDER\_DATE field to change the value to today's date.
5. Type the updated values, as shown.
6. Press Enter to see the display on which the WHERE condition can be specified. If a WHERE condition is not specified, all the rows in the table are updated with the values from the previous display.

```

Specify UPDATE Statement

Type WHERE conditions, press Enter. Press F4 for a list.
ITEM_NUMBER = '303476'
_____
_____

Bottom

Type choices, press Enter.

WITH isolation level . . . 1          1=Current level, 2=NC (NONE)
                                     3=UR (CHG), 4=CS, 5=RS (ALL)
                                     6=RR

F3=Exit      F4=Prompt  F5=Refresh  F6=Insert line  F9=Specify subquery
F10=Copy line F12=Cancel  F14=Delete line F15=Split line  F24=More keys

```

7. Type ITEM\_NUMBER = '303476' in the WHERE condition field.
8. Press Enter to perform the update on the table. A message indicates that the function is complete.

Running a SELECT statement to get all the rows from the table (SELECT \* FROM SAMPLECOLL.INVENTORY\_LIST) returns the following result.

```

Display Data
Data width . . . . . : 71
Position to line . . . . . Shift to column . . . . .
.....+.....1.....+.....2.....+.....3.....+.....4.....+.....5.....+.....6.....+.....7.
ITEM  ITEM          UNIT  QUANTITY  LAST  NUMBER
NUMBER NAME          COST  ON        ORDER  ORDERED
      HAND        DATE
153047 Pencils, red      10.00    25  -        20
229740 Lined tablets    1.50    120  -        20
544931 ***UNKNOWN***    5.00     -    -        20
303476 Paper clips      2.00    100  05/30/07  50
559343 Envelopes, legal  3.00    500  -        20
291124 Envelopes, standard .00     -    -        20
775298 Chairs, secretary 225.00    6  -        20
073956 Pens, black      20.00    25  -        20
***** End of data *****

Bottom

F3=Exit      F12=Cancel  F19=Left    F20=Right   F21=Split

```

Only the entry for paper clips is changed. The LAST\_ORDER\_DATE column is changed to be the current date. This date is always the date the update is run. The NUMBER\_ORDERED column shows its updated value.

This statement can be typed on the Enter SQL Statements display as:

```

UPDATE SAMPLECOLL.INVENTORY_LIST
SET LAST_ORDER_DATE = CURRENT DATE,
    ORDER_QUANTITY = 50
WHERE ITEM_NUMBER = '303476'

```

**Related reference**

SQL programming

## Deleting information from a table

The SQL DELETE statement deletes data from a table. You can delete all rows in a table when they no longer contain needed information, or you can use the WHERE clause with the DELETE statement to identify rows to be deleted during the processing of a single statement.

To remove all the rows in a table that have the null value for the QUANTITY\_ON\_HAND column, follow these steps:

1. Enter the following statement on the Enter SQL Statements display:

```
DELETE
FROM SAMPLECOLL.INVENTORY_LIST
WHERE QUANTITY_ON_HAND IS NULL
```

To check a column for the null value, the IS NULL comparison is used.

2. After the delete operation is completed, run another SELECT statement. This results in the following table.

Display Data						
Position to line . . . . .		Data width . . . . . : 71			Shift to column . . . . .	
.....1.....+.....2.....+.....3.....+.....4.....+.....5.....+.....6.....+.....7.	ITEM	UNIT	QUANTITY	LAST	NUMBER	
NUMBER	NAME	COST	ON	ORDER	ORDERED	
			HAND	DATE		
153047	Pencils, red	10.00	25	-	20	
229740	Lined tablets	1.50	120	-	20	
303476	Paper clips	2.00	100	05/30/07	50	
559343	Envelopes, legal	3.00	500	-	20	
775298	Chairs, secretary	225.00	6	-	20	
073956	Pens, black	20.00	25	-	20	
***** End of data *****						
						Bottom
F3=Exit	F12=Cancel	F19=Left	F20=Right	F21=Split		

The rows with a null value for QUANTITY\_ON\_HAND are deleted.

## Creating and using a view

Views provide a way to divide a table or multiple tables so that you deal with only the data that you need. A view reduces complexity and, at the same time, restricts access. You can create a view using the SQL CREATE VIEW statement.

Using the CREATE VIEW statement, you define a view on a table just as you create a new table that contains only the columns and rows that you want. When your application uses a view, it cannot access rows or columns of the table that are not included in the view. However, rows that do not match the selection criteria can still be inserted through a view if WITH CHECK OPTION is not used.

To create a view, you must have the appropriate authority to the tables or physical files on which the view is based.

If you did not specify column names in the view definition, the column names are the same as those for the table on which the view is based.

You can make changes to a table through a view even if the view has a different number of columns or rows than the table. For INSERT, columns in the table that are not in the view must have a default value.

You can use the view as though it were a table, even though the view is totally dependent on one or more tables for data. The view has no data of its own and therefore requires no storage for the data.

Because a view is derived from a table that exists in storage, when you update the view data, you are really updating data in the table. Therefore, views are automatically kept up-to-date as the tables they depend on are updated.

**Related concepts**

WITH CHECK OPTION on a view

**Related reference**

CREATE VIEW

**Creating a view on a single table**

You can create a view on a single table to show a subset of the data that the table contains. Compared with the original table, the view can have fewer records and fewer columns, and the columns in the view can have a different order.

The following example procedure shows how to create a view on a single table. The view is built on the INVENTORY\_LIST table. The table has six columns, but the view uses only three of the columns: ITEM\_NUMBER, LAST\_ORDER\_DATE, and QUANTITY\_ON\_HAND. The order of the columns in the SELECT clause is the order in which they appear in the view. The view contains only the rows for items that were ordered in the last two weeks. The CREATE VIEW statement looks like this:

1. Use the following command to create the view:

```
CREATE VIEW SAMPLECOLL.RECENT_ORDERS AS
SELECT ITEM_NUMBER, LAST_ORDER_DATE, QUANTITY_ON_HAND
FROM SAMPLECOLL.INVENTORY_LIST
WHERE LAST_ORDER_DATE > CURRENT DATE - 14 DAYS
```

In the preceding example, the columns in the view have the same name as the columns in the table because no column list follows the view name. The schema that the view is created into does not need to be the same schema as the table it is built over. Any schema or library can be used.

2. Run this statement:

```
SELECT *FROM SAMPLECOLL.RECENT_ORDERS
```

The result looks like this.

Display Data		
Position to line . . . . .	Data width . . . . .	26
....+....1.....+....2.....+	Shift to column . . . . .	
ITEM	LAST	QUANTITY
NUMBER	ORDER	ON
	DATE	HAND
303476	05/30/07	100
***** End of data *****		
Bottom		
F3=Exit	F12=Cancel	F19=Left F20=Right F21=Split

The only row selected by the view is the row that you updated to have the current date. All other dates in the table still have the null value so they are not returned.

**Creating a view that combines data from multiple tables**

A view that combines data from multiple tables enables you to show relevant information in multiple tables together. You can create a view that combines data from two or more tables by naming more than one table in the FROM clause.

In the following example procedure, the INVENTORY\_LIST table contains a column of item numbers called ITEM\_NUMBER and a column of item cost called UNIT\_COST. These columns are joined with the ITEM\_NUMBER column and the SUPPLIER\_COST column of the SUPPLIERS table. A WHERE clause is used to limit the number of rows returned. The view contains only the item numbers for suppliers that can supply an item at lower cost than the current unit cost.

1. Use the following statement to create the view:

```
CREATE VIEW SAMPLECOLL.LOWER_COST AS
SELECT SUPPLIER_NUMBER, A.ITEM_NUMBER,UNIT_COST, SUPPLIER_COST
FROM SAMPLECOLL.INVENTORY_LIST A, SAMPLECOLL.SUPPLIERS B
WHERE A.ITEM_NUMBER = B.ITEM_NUMBER
AND UNIT_COST > SUPPLIER_COST
```

2. Run this statement:

```
SELECT *FROM SAMPLECOLL.LOWER_COST
```

The results look like this.

Display Data			
Position to line . . . . .		Data width . . . . . : 51	
.....1.....2.....3.....4.....5.		Shift to column . . . . .	
SUPPLIER_NUMBER	ITEM NUMBER	UNIT COST	SUPPLIER_COST
1234	229740	1.50	1.00
9988	153047	10.00	8.00
2424	153047	10.00	9.00
3366	303476	2.00	1.50
3366	073956	20.00	17.00
***** End of data *****			
			Bottom
F3=Exit	F12=Cancel	F19=Left	F20=Right F21=Split

**Note:** Because no ORDER BY clause was specified for the query, the order of the rows that is returned by the query might be different.

Only rows that contain a supplier cost that is lower than the unit cost can be seen through this view.

**Related tasks**

“Querying your database by running SQL scripts” on page 24

You can create, edit, run, and troubleshoot scripts of SQL statements in the Run SQL Scripts window of System i Navigator. When you finish working with the scripts, you can save the statements to your PC.

---

## Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Software Interoperability Coordinator, Department YBWA  
3605 Highway 52 N  
Rochester, MN 55901  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

- | The licensed program described in this document and all licensed material available for it are provided
- | by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement,
- | IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. \_enter the year or years\_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## | **Programming interface information**

This Database overview publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

---

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

DB2  
i5/OS  
IBM  
IBM (logo)  
System i  
System i5

- | Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks
- | of Adobe Systems Incorporated in the United States, and/or other countries.

Other company, product, or service names may be trademarks or service marks of others.

---

## Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.







Printed in USA