

Create an SQL query

In this tutorial, you learn how to create an SQL query. Using the SQL Builder, which is a visual interface for creating and running SQL queries, you build a SELECT statement that includes the following elements:

- A table alias
- A column alias
- A database function expression
- A CASE expression
- A grouping clause



30 minutes



[Start the tutorial](#)



[View the PDF version](#)

Introduction: Create an SQL query

Time required

To complete this tutorial, you will need approximately **30 minutes**. If you decide to explore other facets of creating an SQL query while working on the tutorial, it could take longer to finish.

Prerequisites

To perform the exercises in this tutorial, the following software must be installed on your system:

- DB2^(R) Universal Database Version 8.1 or later

Learning objectives

This tutorial teaches you how to build an SQL query by using the SQL Builder. The tutorial has several exercises that must be completed in sequence.

You will work with a database for a video store called VIDEOS. The database holds data that is related to the store customers, inventory, video rentals, and employees. The video store application needs a query that lets the store employees look up the video titles that a particular customer has rented and what day of the week the videos are due. The query involves two joins, a query condition (WHERE clause), and a GROUP BY clause. The SQL statement also needs a CASE expression, a function, and a host variable to get the day of the week, and to substitute the name of the customer at run time.

In this tutorial, you perform the following tasks:

1. Create the VIDEOS database
2. Connect to the database and import a local copy of it
3. Create a SELECT statement
4. Add tables to the statement
5. Add table aliases
6. Specify the result columns
7. Add a column alias
8. Join tables
9. Create a query condition
10. Add a GROUP BY clause
11. Run the SQL query

When you are ready, begin [Exercise 1.1: Creating and connecting to the VIDEOS database](#)

[Terms of use](#) | [Feedback](#)

(C) Copyright IBM Corporation 2000, 2005. All Rights Reserved.

Exercise 1.1: Creating and connecting to the VIDEOS database

In this exercise, you create the VIDEOS database, connect to the database, and then import a copy of the database to your workspace.

Creating the VIDEOS database

The exercises in this tutorial use tables in the VIDEOS database. You will create the database by running a batch file. The batch file and the source files that contain the data for the VIDEOS database tables are included in the createVideos.zip file, which is in the following directory:

installdir\rwd\eclipse\plugins\com.ibm.etools.sqlbuilder_*version*\samples, where *installdir* is your installation directory.

Linux The scripts to create the VIDEOS database for this tutorial are not currently provided for the Linux operating system. You can create your own database and follow along with most of the steps in the exercises, adjusting them as necessary for your database and statements.

1. Open the createVideos.zip file and extract the files to a new directory.
2. Open a DB2 command window. Click **Start > Programs > IBM DB2 > Command Line Tools > Command Window**.
3. In the DB2 command window, change to the directory that you extracted the files to.
4. Enter the following command: `createVideoStore.bat`.
After messages are displayed that show progress in creating the database, the following message is displayed:
`DB200001 The SQL command completed successfully`
5. To test the creation of the VIDEOS database, enter the following commands in the DB2 command window:
`db2 connect to videos`
`db2 select * from xmltest.rentals`

The values in the xmltest.rentals table (which is part of the VIDEOS database) are listed, showing you that the database was created successfully.

Connecting to the database

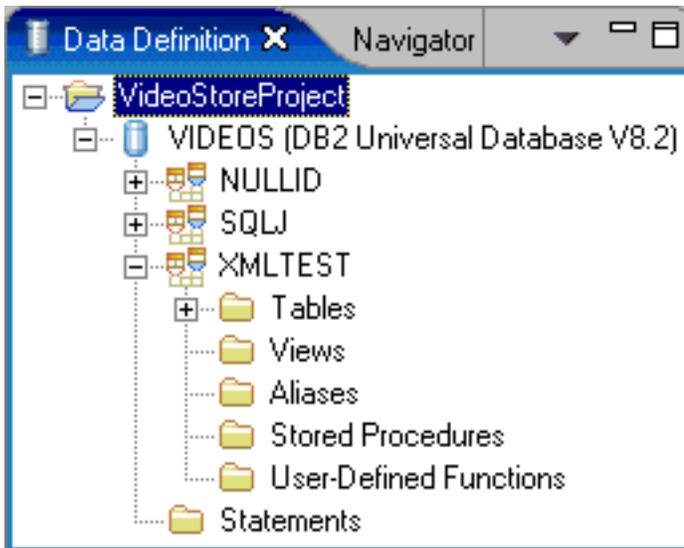
Before you can create an SQL query, you must connect to the VIDEOS database. You must then import a copy of the database to your workspace so that you can work on it locally.

1. Switch to the Data perspective (**Window > Open Perspective > Data**).
2. Right-click in the Database Explorer view, and then click **New Connection** on pop-up menu.
3. On the first page of the New Database Connection wizard, click **Choose a DB2 alias**, and then click **Next**.
4. On the Specify connection parameters page, provide the information for your database connection:
 - a. Select the JDBC driver that you want to use.
 - b. In the **Alias** list, click **VIDEOS**.
 - c. If you do not want to use your operating system user ID and password (which is the default), clear the check box and then specify the user ID and password that you want to use.
 - d. Click **Finish**.

The **VideoStore** connection that you created now is shown in the Database Explorer view.

5. When you are prompted to copy the database metadata to a project, click **Yes**.
To work with a database locally, you must import a copy of the database to your workspace.
6. In the Copy to Project window, type `VideoStoreProject` for the folder name, and then click **Finish**.

7. When you are prompted to create the folder, click **Yes**.
8. In the Data Definition view, expand the **VideoStoreProject** folder and the items below it in the tree. The VIDEOS database is shown as part of the VideoStoreProject project.



Now you are ready to begin [Exercise 1.2: Creating a SELECT statement with tables](#).

[Terms of use](#) | [Feedback](#)

(C) Copyright IBM Corporation 2000, 2005. All Rights Reserved.

Exercise 1.2: Creating a SELECT statement with tables

Before you begin, you must complete [Exercise 1.1: Creating and connecting to the VIDEOS database](#).

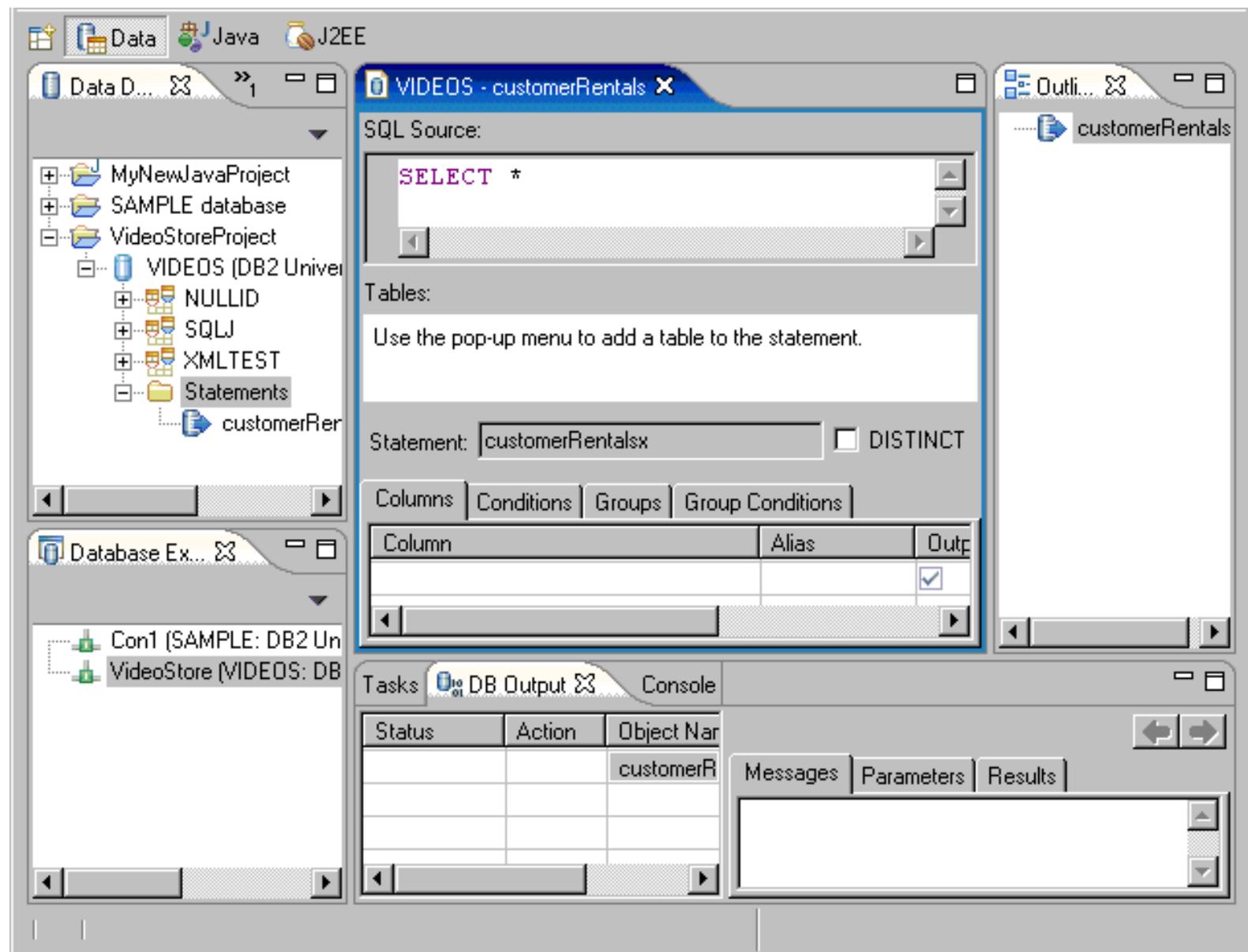
In this exercise, use the SQL Builder to create a SELECT statement for the VIDEOS database, add the necessary database tables to the statement, and then assign aliases to each of the tables.

Creating a SELECT statement

You can add SELECT, INSERT, DELETE, UPDATE, WITH, and FULLSELECT statements by using the **Statements** folder pop-up menu in the Data Definition view. You will now create a SELECT statement for the VIDEOS database.

1. In the Data Definition view, right-click the **Statements** folder and then click **New > Select Statement**.
2. In the New Select Statement window, type `customerRentals` for the statement name, and then click **OK**.

The SELECT statement template is created and opens in the SQL Builder, as shown here:



The SQL Builder has three panes:

SQL Source

The top pane shows the SQL source for your statement as it is being built.

Tables

The middle pane shows the tables and table columns that are added to the statement.

Design

The bottom pane is used to provide details for statement clauses. This pane changes depending on the statement type being edited.

You will use all of these panes throughout this tutorial.

In addition to the Data Definition view, you can use the following two views with the SQL Builder:

Outline

Shows the statement that you are currently working on. For more complex statements such as WITH and FULLSELECT that can include, for example, subselects and common table expressions, the Outline view shows the structure of the statement.

DB Output

Shows the messages, parameters, and results that are related to running your SQL statement.

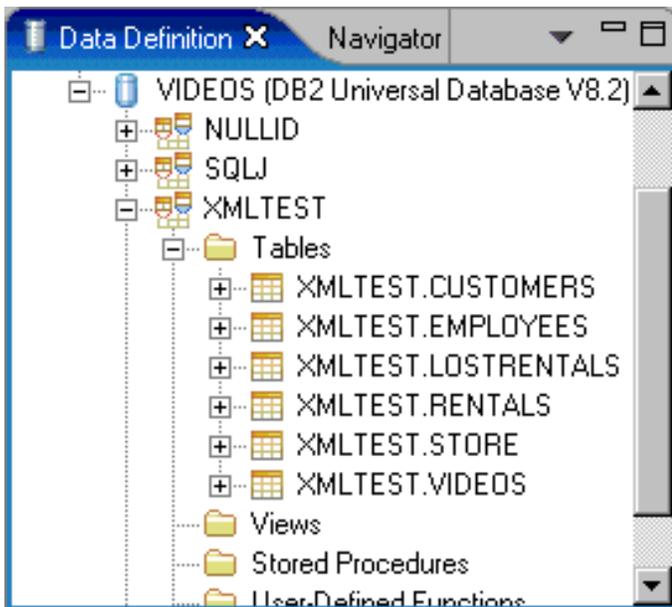
The SELECT statement that you created is already open in the SQL Builder, so you can continue to construct it. You are creating only a SELECT statement in this tutorial, but you can also use the SQL Source pane, the Tables pane, and the Design pane of the SQL Builder to create INSERT, DELETE, UPDATE, FULLSELECT, and WITH SQL statements.

Adding tables to the statement

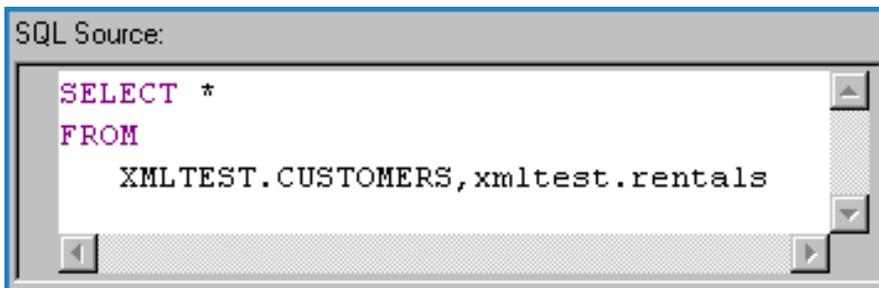
You will now add three tables to the SELECT statement for the **customerRentals** query. You are creating a query that lets the video store employees look up the video titles that a particular customer has rented and the day of the week that the videos are due. The query needs to include the CUSTOMERS, RENTALS, and VIDEOS tables.

The CUSTOMERS table contains the name and ID (identification number) of each person who rents from the video store. The data in the VIDEOS table includes the ID (identification number) and title of each video. The data in the RENTALS table includes the following information for each video that is currently rented: the customer ID, video ID, and the date that the video is due.

1. In the Data Definition view, expand the **XMLTEST** schema in the **VIDEOS** database tree, and then expand the **Tables** folder. You will see the tables for the database.



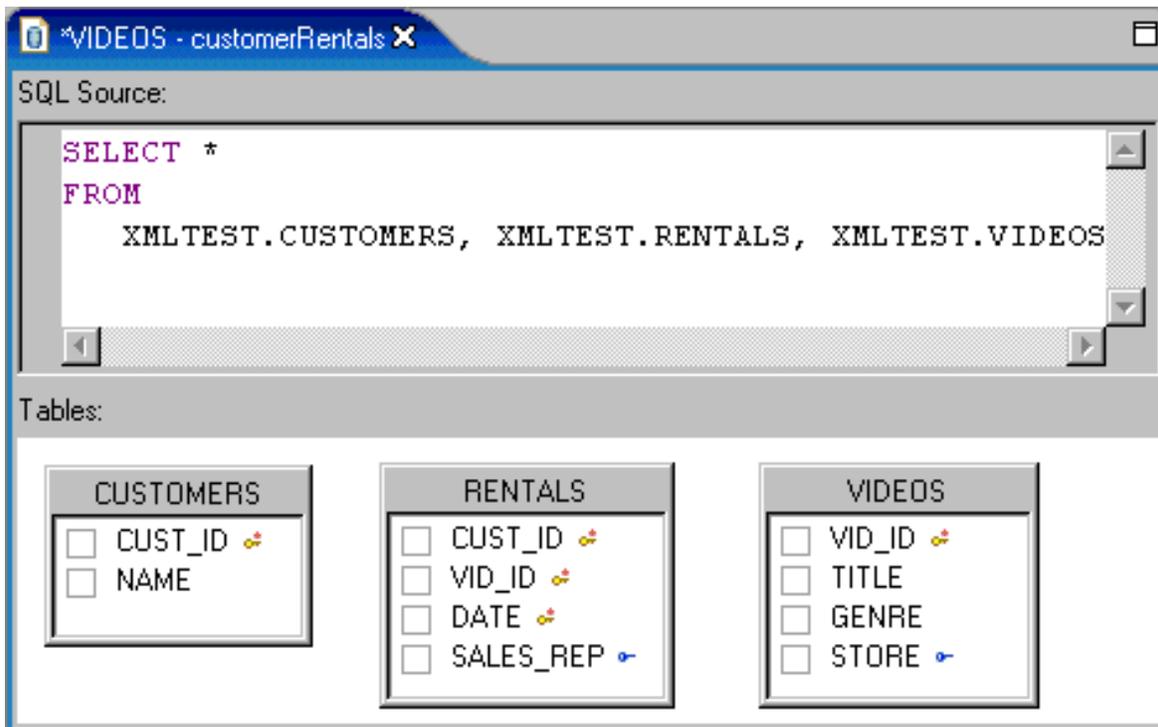
2. Drag the **XMLTEST.CUSTOMERS** table from the Data Definition view to the Tables pane in the SQL Builder. The CUSTOMERS table is shown in the Tables pane, and the source code in the SQL Source pane shows the addition of the CUSTOMERS table in the SELECT statement.
3. In the SQL Source pane of the SQL Builder, type `,xmltest.rentals` (including the leading comma) after `XMLTEST.CUSTOMERS`, as shown in the following screen capture, and then click anywhere outside the SQL Source pane.



After the changed statement is parsed and determined to be valid, the RENTALS table is added in the Tables pane, and the code in the SQL SOURCE pane is formatted.

4. Right-click in the Tables pane, and then click **Add Table** on the pop-up menu.
5. In the **Table name** list, select **XMLTEST.VIDEOS** and then click **OK**.
The VIDEOS table is added in the Tables pane, and the source code in the SQL Source pane shows the addition of the VIDEOS table in the SELECT statement.

All three tables are now shown in the Tables pane. Note the corresponding changes to the source code in the SQL Source pane.



Adding table aliases

Next, you will create an alias for each of the tables in the SELECT statement. An alias is an indirect method of referencing a table so that an SQL statement can be independent of the qualified name of that table. If the table name changes, only the alias definition must be changed.

Table aliases can be added when you add the table to the statement, or after the table is added, by using the table pop-up menu in the Tables pane. You can also use the table pop-up menu to delete tables or create joins between tables.

The aliases for the CUSTOMERS, RENTALS, and VIDEOS tables will be **C**, **R**, and **V**, respectively.

1. In the Tables pane, right-click the header in the CUSTOMERS table, and then click **Update Alias** on the pop-up menu.
2. In the Change table alias window, type **C** as the alias for the table, and then click **OK**.
In the Tables pane, the alias is shown in the header for the CUSTOMERS table. In the SQL Source pane, the alias is represented by the `AS C` code for the CUSTOMERS table.
3. Repeat steps 1 and 2 to add aliases for the RENTALS (R) and VIDEOS (V) tables.

Now you are ready to begin [Exercise 1.3: Specifying the result columns](#).

[Terms of use](#) | [Feedback](#)

(C) Copyright IBM Corporation 2000, 2005. All Rights Reserved.

Exercise 1.3: Specifying the result columns

Before you begin, you must complete [Exercise 1.2: Creating a SELECT statement with tables](#).

In this exercise, you will specify what columns to show in the query result. A store employee needs to see the customer name, the title of each video that the customer has rented, and the day of the week that each video is due. You will include two of the columns and add a CASE expression for the third result column, and then assign an alias to the CASE column expression.

Selecting columns for the result column set

You will add the following columns to the result column set for the customer name and the video title:

- The NAME column in the CUSTOMERS table
- The TITLE column in the VIDEOS table

1. In the Tables pane, select the **NAME** check box in the **C** (CUSTOMER) table.
2. On the Columns page in the Design pane, double-click the first empty cell in the **Column** column, click **V.TITLE** in the list, and then press Enter.
The columns are added to the SQL source code in the SQL Source pane.

Adding a CASE expression to the result column set

The third column for the query result set will be the result of a column expression. The video store database stores the date that the video is due in DATE format. The DATE format needs to be translated into a character string that contains the name of the day that the video is due. A database function will get an integer from the date that corresponds to the day of the week. You will use a CASE expression to evaluate the function and convert the integers that are returned from the function into character strings for each day of the week (for example, 1 will be changed to "Sunday" in the query result). You will perform the following activities to build the CASE expression:

- Open the Expression Builder wizard.
 - Create a simple type CASE expression.
 - Add the DAYOFWEEK function for evaluation by the CASE expression.
The DAYOFWEEK function lets you get the day of the week from the DATE column. This function returns an integer that corresponds to the day of the week.
 - Add WHEN clauses to produce the results for the CASE expression.
1. On the Columns page in the Design pane, double-click the third cell in the **Column** column (the first empty cell), click **Build Expression** at the end of the list, and then press Enter.
The Expression Builder wizard opens.
 2. Create a simple type CASE expression.
 - a. On the Expression Builder page, click **CASE - search or simple type**, and then click **Next**.
 - b. On the CASE Options page, click **Simple-When-Clause**, and then click **Next**.
 3. Add the DAYOFWEEK function:
 - a. On the Simple Type Case page, in the **CASE** list, click **Build function expression**. The Function Expression Builder wizard opens.
 - b. In the **Select a function category** list, click **Date and time**.

- c. In the **Select a function** list, click **DAYOFWEEK**.
- d. In the **Select a function signature** list, click **DAYOFWEEK(DATE) --> INTEGER**.
The function signature shows that the function requires one argument.
- e. In the **Value** column of the argument table, click the cell, click **R.DATE** in the list, and then press Enter.
The syntax of the function expression is `DAYOFWEEK (R . DATE)` , as shown in section 5 of the following screen capture:

Function Expression Builder

Function Builder Page

Use this page to build a function expression.

1. Select a function category:
Date and time
2. Select a function:
DAYOFWEEK
3. Select a function signature:
DAYOFWEEK(DATE) --> INTEGER
4. Select an expression for the argument(s):

Parameter	Value	Operator	
#1	R.DATE		

5. Preview of function expression:
DAYOFWEEK(R.DATE)

Finish **Cancel**

- f. Click **Finish** to return to constructing the CASE expression.

The new function is now displayed in the **CASE** list at the top of the Simple Type Case page.

4. Add seven WHEN clauses to the CASE expression (one for each day of the week) to translate the integer returned from the DAYOFWEEK function into character strings. The following table shows the character string that is needed for each value returned from the DAYOFWEEK function:

DAYOFWEEK value	Resulting character string
1	'Sunday'
2	'Monday'

3	'Tuesday'
4	'Wednesday'
5	'Thursday'
6	'Friday'
7	'Saturday'

- In the table, double-click the empty cell in the **Expression** column to the right of **WHEN**, and then enter the integer 1.
- Double-click the empty cell in the **Result Expression** column to the right of **THEN** and enter the string 'Sunday'.
- For each of the remaining six days of the week, click **Add WHEN Clause**, and then repeat steps 4a and 4b, so that your table looks similar to the screen capture shown here:

Build up when-then clauses and/or add an else clause.

CASE

	Expression		Result Expression
WHEN	1	THEN	'Sunday'
WHEN	2	THEN	'Monday'
WHEN	3	THEN	'Tuesday'
WHEN	4	THEN	'Wednesday'
WHEN	5	THEN	'Thursday'
WHEN	6	THEN	'Friday'
WHEN	7	THEN	'Saturday'

END

Add ELSE clause

- Click **Finish** to close the Expression Builder wizard.

The completed CASE expression is shown in the list of column expressions on the Columns page in the Design pane, and is also shown in the SQL statement source.

Adding a column alias

In the customerRentals SQL statement, you will add a column alias for the CASE column expression. You will use the Columns page in the Design pane of the SQL Builder.

- Click the **Columns** tab in the Design pane.
- Click the cell in the **Alias** column next to the CASE column expression and then enter `DUEDAY`.
In the SQL Source pane, the column alias `AS DUEDAY` is shown after the CASE expression. When you run the query, this alias appears as the title for the column in the result table.

The Columns page now looks like this:

Column	Alias	Output	Sort Type
C.NAME		<input checked="" type="checkbox"/>	
V.TITLE		<input checked="" type="checkbox"/>	
CASE DAYOFWEEK(R.DATE) WH...	DUEDAY	<input checked="" type="checkbox"/>	
		<input checked="" type="checkbox"/>	

Now you are ready to begin [Exercise 1.4: Adding joins, a query condition, and a GROUP BY clause.](#)

[Terms of use](#) | [Feedback](#)

(C) Copyright IBM Corporation 2000, 2005. All Rights Reserved.

Exercise 1.4: Adding joins, a query condition, and a GROUP BY clause

Before you begin, you must complete [Exercise 1.3: Specifying the result columns](#).

In this exercise, you qualify the query by performing the following steps:

- Restrict the query results to the customers who are currently renting videos and to the videos that are currently being rented.
- Limit the query results to a specific customer whose name you specify when you run the query.
- Organize the query results by the day of the week that the rented videos are due, and within each day by the video title, and for each video title by customer name.

Joining tables

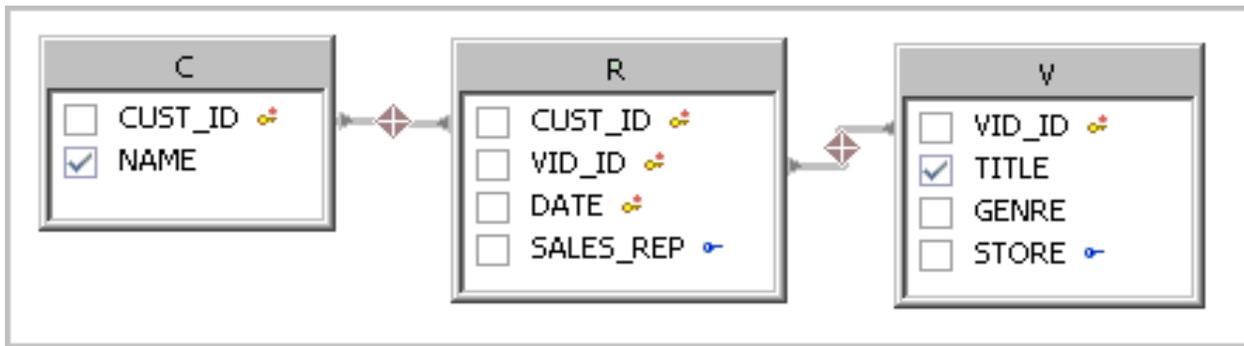
A join operation lets you retrieve data from two or more tables based on matching column values. The data in the tables is linked into a single result. Two joins are needed for this query. The query result needs to include the RENTALS and CUSTOMERS table entries that have matching CUST_ID column values. The query result must also include the RENTALS and VIDEOS table entries that have matching video IDs (VID_ID column values).

1. In the Tables pane, right-click the header of the **R** table, and then click **Create Join** on the pop-up menu.
2. In the Create Join window under **Source**, make the following selections:
 - a. In the **Table (alias)** list, click **RENTALS (R)**.
 - b. In the **Column** list, click **CUST_ID**.
3. Under **Target**, make the following selections:
 - a. In the **Table (alias)** list, click **CUSTOMERS (C)**.
 - b. In the **Column** list, click **CUST_ID**.
4. Click **OK**. A join connector appears between the two columns.
5. In the Tables pane, drag your pointer from the **VID_ID** column in the **R** (RENTALS) table to the **VID_ID** column in the **V** (VIDEOS) table.

Look at the SQL Source pane to see the joins in the source code:

```
WHERE
    R.CUST_ID = C.CUST_ID
    AND R.VID_ID = V.VID_ID
```

In the Tables pane, both joins have been created as shown here:



You can change the join type (for example, from the default inner join to a left, right, or full outer join) in the Tables pane by right-clicking the connector, clicking **Specify Join Type** on the pop-up menu, and then selecting the join type that you want in the Specify Join window.

Creating a query condition

Next, the query needs a query condition so that the query extracts only result rows that have the customer name that you want. You add conditions to the query by using the Conditions page in the Design pane.

1. Click the **Conditions** tab to see the Conditions page.
2. Double-click in the **Column** column and then click **C.NAME** in the list.
3. Double-click in the **Operator** column and then click the = operator.
4. Double-click in the **Value** column and enter :custName. The colon followed by a variable name is the SQL syntax for a variable that will be substituted with a value when you run the query. You will see how this works when you run the SQL query.

The Conditions page now looks like the image shown here:

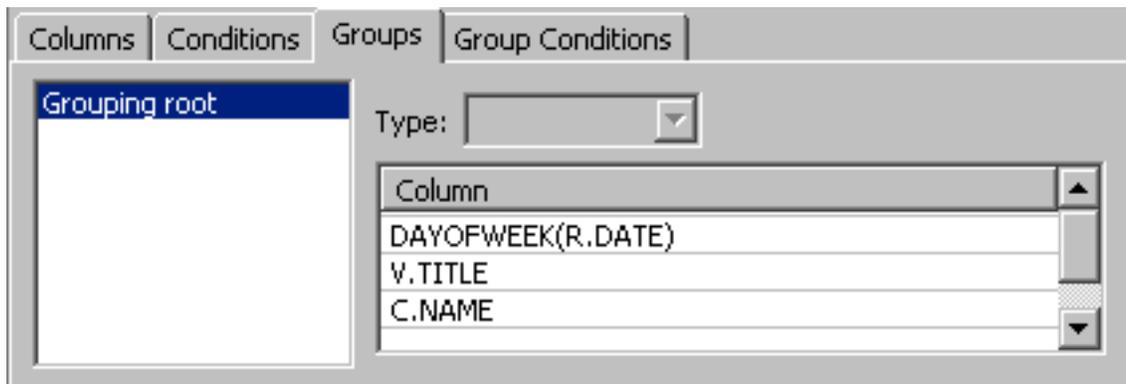
Column	Operator	Value	And/Or
C.NAME	=	:custname	

Adding a GROUP BY clause

You will group the query by the day of the week, then by title, and then by customer name. To create a GROUP BY clause in the SQL Builder, use the Groups page in the Design pane. In this view, you can also create more advanced groupings in your query result by using column expressions, nested groups, grouping sets (in DB2 Universal Database only), and the ROLLUP and CUBE grouping functions (in Oracle and DB2 Universal Database).

1. In the Design pane, click the **Groups** tab.
2. Add the DAYOFWEEK function as a result column.
 - a. In the **Column** table, double-click the first row, click **Build Expression** in the list, and then press Enter.

- b. On the Expression Builder page of the wizard, click **Function**. The Function Expression Builder page opens.
 - c. In the **Select a function category** list, click **Date and time**.
 - d. In the **Select a function** list, click **DAYOFWEEK**.
 - e. In the **Select a function signature** list, click **DAYOFWEEK(DATE) --> INTEGER**.
The function signature shows that the function requires one argument.
 - f. In the **Value** column of the argument table, click the cell, click **R.DATE** in the list, and then press Enter.
 - g. Click **Finish**. The DAYOFWEEK function is shown in the first cell of the **Column** table.
3. In the second row of the **Column** table, select the **V.TITLE** column from the list, and then in the third row, select **C.NAME**. The Groups page now looks like the following image:



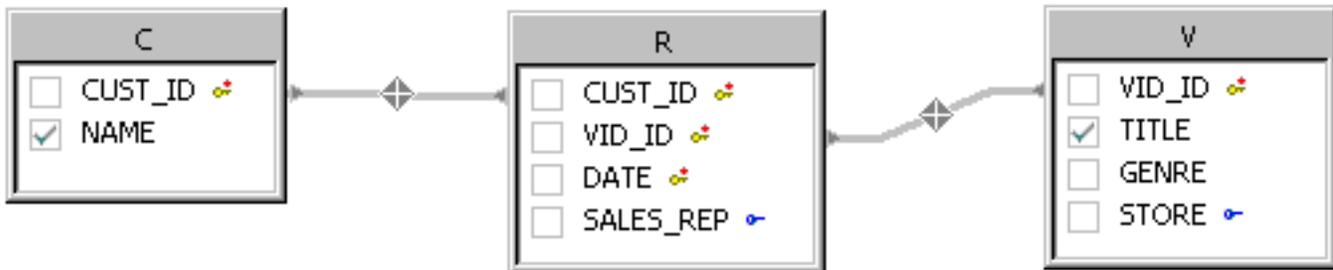
The query is now complete. The query looks like this in the SQL Builder:

SQL Source:

```

SELECT
  C.NAME,
  V.TITLE,
  CASE DAYOFWEEK(R.DATE)
    WHEN 1 THEN 'Sunday'
    WHEN 2 THEN 'Monday'
    WHEN 3 THEN 'Tuesday'
    WHEN 4 THEN 'Wednesday'
    WHEN 5 THEN 'Thursday'
    WHEN 6 THEN 'Friday'
    WHEN 7 THEN 'Saturday'
  END AS DUEDAY
FROM
  XMLTEST.RENTALS AS R, XMLTEST.CUSTOMERS AS C, XMLTEST.VIDEOS
WHERE
  R.CUST_ID = C.CUST_ID
  AND R.VID_ID = V.VID_ID
  AND C.NAME = :custname
GROUP BY
  DAYOFWEEK(R.DATE),
  V.TITLE,
  C.NAME
    
```

Tables:



Statement: customerRentals

Columns | Conditions | Groups | Group Conditions

Column	Alias	Output	Sort Type	Sort Order
C.NAME		<input checked="" type="checkbox"/>		
V.TITLE		<input checked="" type="checkbox"/>		
CASE DAYOFWEEK(R.DATE) WHE...	DUEDAY	<input checked="" type="checkbox"/>		

Now you are ready to begin [Exercise 1.5: Running the SQL query](#).

[Terms of use](#) | [Feedback](#)

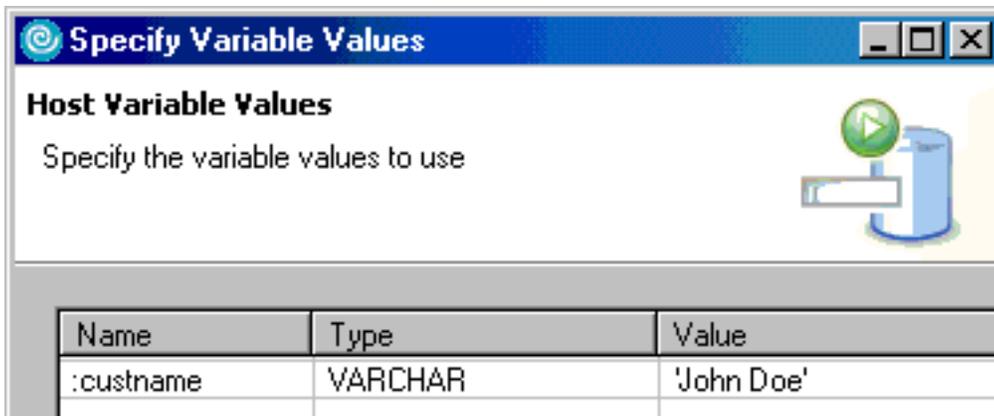
(C) Copyright IBM Corporation 2000, 2005. All Rights Reserved.

Exercise 1.5: Running the SQL query

Before you begin, you must complete [Exercise 1.4: Adding joins, a query condition, and a GROUP BY clause](#).

Before you incorporate the SQL query into the video store application, you need to test the query to ensure that it returns correct results. You can do this by using the SQL Builder.

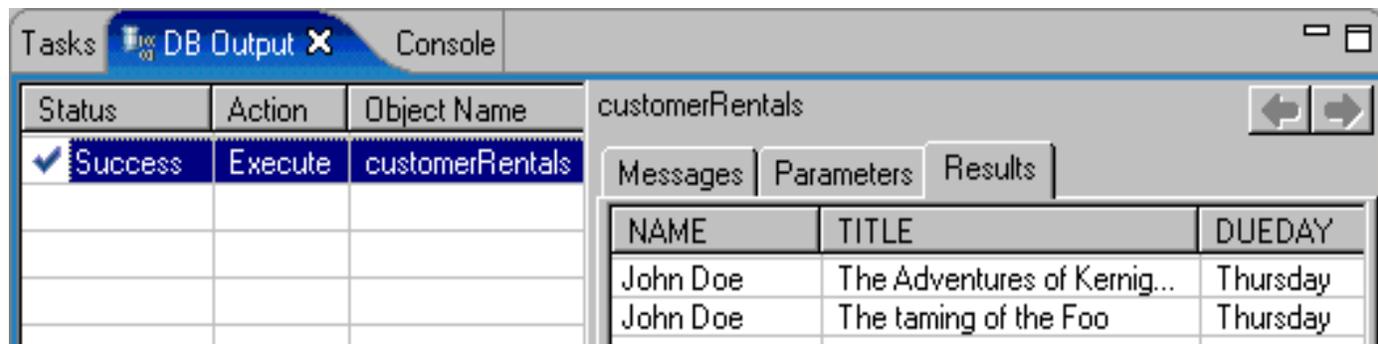
1. Click anywhere in the SQL Builder to enable the **Execute** command on the SQL menu.
2. Click **SQL > Execute**. The Specify Variable Values window opens.
3. Double-click the **Value** cell in the :custname row, and then enter the string `John Doe` in the cell.



After you press Enter, the name is automatically embedded in single quotation marks.

4. Click **Finish**.

Your query runs, and the results are displayed in the DB Output view.



Finish your tutorial by reviewing the information in the [Summary](#).

[Terms of use](#) | [Feedback](#)

(C) Copyright IBM Corporation 2000, 2005. All Rights Reserved.

Summary: Create an SQL query

Good work! This tutorial taught you how to create a simple SQL query by using the SQL Builder.

Completed learning objectives

If you completed all of the exercises, you learned how to:

- Construct SQL queries in the SQL Builder, including joins, expressions, conditions, and groups
- Run SQL queries on the database and pass in host variables for the query

Next step

If you want to learn more about the topics covered in this tutorial, consider the following source:

- View help topics on [Working with SQL statements](#)

[Terms of use](#) | [Feedback](#)

(C) Copyright IBM Corporation 2000, 2005. All Rights Reserved.