



## Overview of SQL

By proceeding with this courseware you agree with [these terms and conditions](#). Interskill Learning Pty. Ltd. © 2019





## Objectives

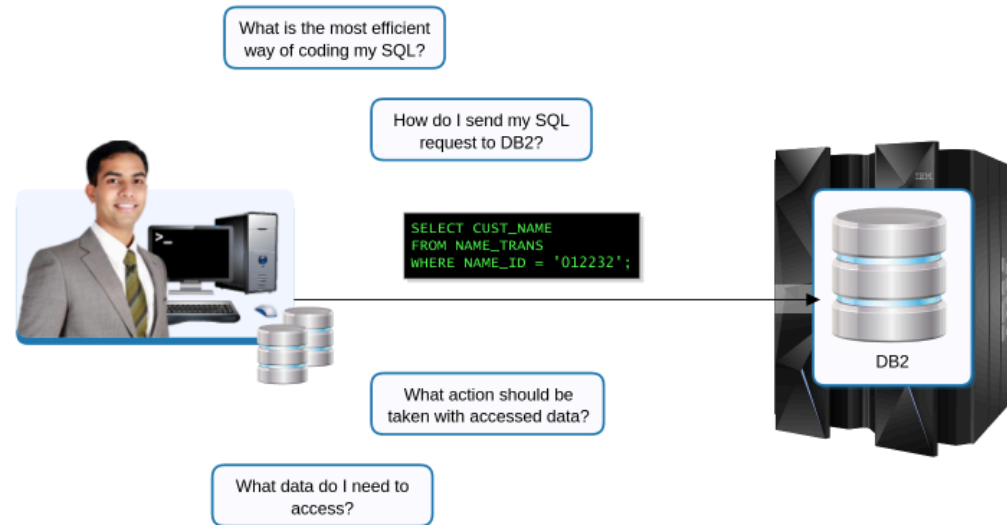
---

### Overview of SQL

In this module, you will revisit the concepts relating to how SQL is used to access Db2 data, and see how the SQL SELECT statement is used to obtain Db2 data. You will also look at some of the other commonly used SQL statements used when creating and modifying table data.

After completing this module, you will be able to:

- Code a SELECT Statement to Obtain Table Data
- Identify Statement Clauses and Predicates That Further Qualify Data to be Selected
- Explain the Function of Other Commonly Used SQL Statements



---

SQL is used to obtain and manipulate data that is stored in Db2 tables. SQL consists of over 100 different statements that can be used to insert, query, update, delete, and authorize access to Db2 data.

In this section you will look at the use and syntax of the SELECT statement and how it is used to obtain data from Db2.

```
SPUFI                                SSID: DB11
====>
Enter the input data set name:         (Can be sequential)
1 DATA SET NAME ... ==> 'BAY1.SPUFI.INPUT'
2 VOLUME SERIAL ... ==>
3 DATA SET PASSWORD ==>             (Enter if not cataloged)
                                       (Enter if password required)

Enter the output data set name:        (Must be a sequential data set)
4 DATA SET NAME ... ==> 'BAY1.SPUFI.OUTPUT'

Specify processing options:
5 CHANGE DEFAULTS ... ==> YES
6 EDIT INPUT ... ==> YES
7 EXECUTE ... ==> YES
8 AUTOCOMMIT ... ==> YES
9 BROWSE OUTPUT ... ==> YES

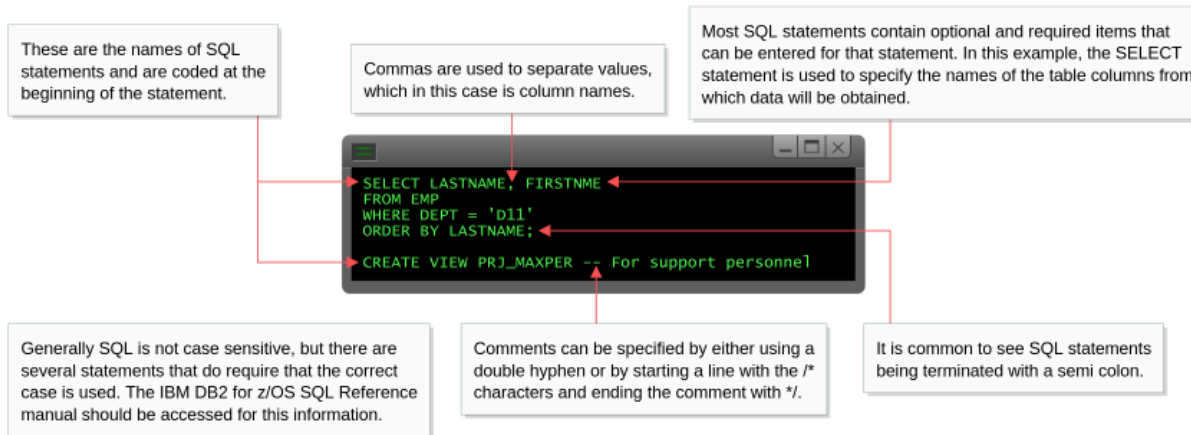
For remote SQL processing:
10 CONNECT LOCATION ==>

PRESS:  ENTER to process   END to exit           HELP for more information
```

The input data set or PDS member contains either existing SQL statements or is empty allowing you to enter your own statements.

The result from processing the SQL statements is placed in the output data set, which is automatically opened in browse mode following the processing.

SQL statements can be inserted into source application programs and invoked when they are executed or they can be run interactively using a TSO product such as SPUFI. The examples displayed in this module are created using this method.



When coding SQL you need to adhere to the syntax rules, otherwise your code will not execute.



```
BROWSE USER01234.RESULT -----Columns 001 072
COMMAND INPUT ==> SCROLL ==> PAGE
-----+-----+-----+-----+-----+
SELECT *                               000100000
FROM ADDR01.TABLE;                     000200000
-----+-----+-----+-----+
LASTNAME  FIRSTNAME  ADDRESS  PHONE  GENDER
ADDISON   PAUL          123 RUSHDALE STREET, COOTAMUNDRA  4819  M
BAKER     GAIL          456 SMITH STREET, EASTWOOD        5765  F
DAVIS     MICHELLE      12 TINTERN AVENUE, ASHFIELD        7673  F
D'SILVA  MARK          25 HIGH STREET, SEAFORD            5538  M
FISHER    PHILLIP       2/144 POWER STREET, HANTHORN       2236  M
MACKINTOSH HARRY        154 SYDNEY DRIVE, SWAN VIEW        7673  M
PETERSON  BRETT         16 CHARLIES PLACE, TOONOOMBA       4231  M
RICHARDSON PETER        97 STURT HIGHWAY, GLENELG         6582  F
SMITH     EYVONNE      17/125 GRANGE ROAD, SOUTH YARRA    7164  M
SMITH     GREG         49 EVANS ROAD, PENRITH
-----+-----+-----+-----+
DSNE610I NUMBER OF ROWS DISPLAYED IS 10.
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 0.
-----+-----+-----+-----+

```

SQL statement that was entered

Table data

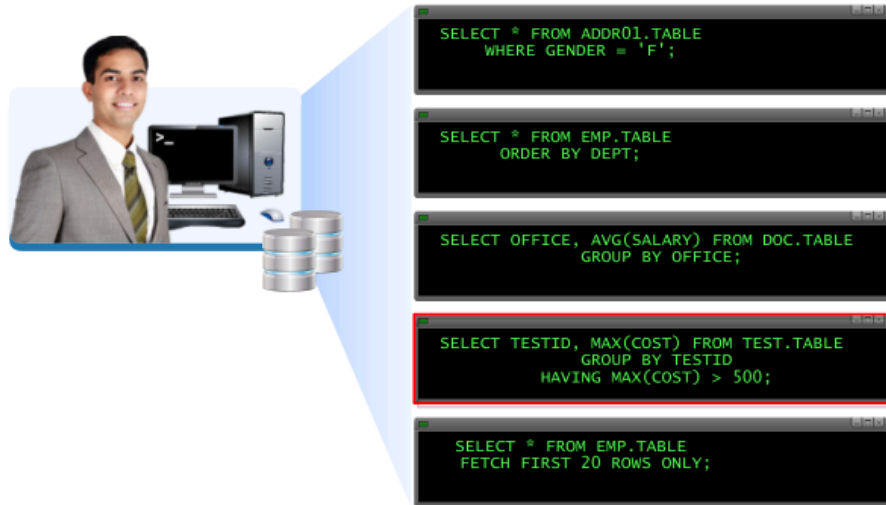
General invocation messages

One of the most commonly used SQL statements is SELECT. This statement is used to query a table and produce results that are in a table format.

The simple example above selects and displays all entries from the ADDR01 table.







```
SELECT * FROM ADDR01.TABLE
WHERE GENDER = 'F';
```

```
SELECT * FROM EMP.TABLE
ORDER BY DEPT;
```

```
SELECT OFFICE, AVG(SALARY) FROM DOC.TABLE
GROUP BY OFFICE;
```

```
SELECT TESTID, MAX(COST) FROM TEST.TABLE
GROUP BY TESTID
HAVING MAX(COST) > 500;
```

```
SELECT * FROM EMP.TABLE
FETCH FIRST 20 ROWS ONLY;
```

The HAVING clause is used in conjunction with the GROUP BY clause to further refine the data to be displayed. In this example, the MAX function is used to return the name of each TESTID and maximum cost of that test. The HAVING clause will only display those TESTIDs that have a cost greater than 500.

When the data you require needs to be further qualified, several types of clauses can be used with the SELECT statement.

**Mouse-over** the statements above for a brief description of their purpose.







Extract from EMP.TABLE			
EMPID	SALARY	FIRSTNAME	LASTNAME
110	63827.50	DONALD	D'SILVA
111	49912.50	GAIL	BAKER
113	46282.50	SONYA	PETERSON
114	48611.75	TOM	SMITH
116	28798.00	NARI	RICHARDSON
117	74388.20	RITA	DAVIS
118	30588.80	JOHN	ADDISON

```
SELECT EMPID, LASTNAME FROM EMP.TABLE  
WHERE (SALARY / 52) > 1000;
```

EMPID	LASTNAME
110	D'SILVA
117	DAVIS

There will be times when you need to select rows from a table based on a mathematical calculation. For example, you may need to select all the employees in the organization that get paid more than \$1000 a week. Arithmetical operators such as + (plus), - (minus), \* (multiply) and / (divide) can be used with any column name, whether or not they are specified to be displayed in the SELECT statement, provided that the column's data type is numeric.



The following comparison operators can be used:

=	Equal to
<>	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

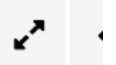
```
SELECT * FROM BUSINESS.TABLE  
WHERE CUSTNAME = 'GH LIMITED';
```

```
SELECT ORDER, COST, MARKUP FROM SALES.TABLE  
WHERE MARKUP > 100;
```

```
SELECT INGRED, SHOP, COST FROM RECIPE.TABLE  
WHERE COST <= 20;
```

There have been several examples already shown in this module where values are compared to produce a subset of the original table data. SQL supports a number of different types of comparison operators that can be used.

Above are some examples.



You need to select employees from a table who have a salary between 30,000 and 40,000.

You only require unique names to be selected from a lastname column in an address table.

You need to display all patients that live in a specific area where at least one of those patients has returned a specific test result.



You need to compare results from a SELECT statement with one or more values.

You need to display all clients that live in a suburb with 'SOUTH' in its name.

You need to display certain users that have not completed a test.

```
SELECT PATID, AREA, TESTRES FROM CUST.TABLE A
WHERE EXISTS (SELECT * FROM CUST.TABLE
WHERE A.AREA=AREA AND TESTRES=K108);
```

---

Another item that can be added to a SQL statement is a predicate. A predicate is used against a row or group and returns a value of true, false or unknown. Where the result is true, that data is selected for the result table.

**Mouse-over** the scenarios above to see how predicates are used.



---

There are a number of built-in and user-defined functions that you can also use to further modify the data captured through your SELECT statement.

The CONCAT function shown at the top of this page is concatenating character strings, while MAX and MIN can be used to return the maximum and minimum value from a set of values. In this example the MAX function is being used to display the largest salary within each department.

Combining all records from two tables containing the same column names.

```
SELECT * FROM SALES.EMP  
UNION  
SELECT * FROM MANU.EMP;
```

This will create a result table that contains the USERID that has not completed a course or passed an assessment.

```
SELECT USERID FROM STUDY.DET  
WHERE COURSE <> 'COMP'  
UNION  
SELECT USERID FROM ASSESS.DET  
WHERE SCORE <> 'PASS';
```



---

The majority of examples shown so far have dealt with a SELECT statement creating data in a result table. The merging of data with that from a separate SELECT statement is possible using the UNION keyword.



If your SQL has a simple syntax error, you can resolve the problem.

You can code a SQL query to select all entries from a specific table.



You can create a subset of data containing only certain columns, from a table.

You can code clauses to further refine the data that you need to select from a table.

---

This section introduced you to the SQL SELECT statement, showing the general syntax and the coding required to create subsets of table data based on supplied criteria.



How do you create a table?

Can you delete table data based on selection criteria you provide?

Can you insert your own data into an existing table?

Can you copy data from another table into your table?

How can you update table data values?



---

In this section you will discover additional keywords that can be used with the SELECT statement to create specific result table data. You will also look at statements that are used to create a table, insert data into it and update its data.



```
CREATE TABLE STAFF_DATA  
(STAFFID CHAR(4) NOT NULL,  
FIRSTNAME VARCHAR(15) NOT NULL,  
LASTNAME VARCHAR(25) NOT NULL,  
SALARY DECIMAL(8,2),  
DEPTCODE CHAR(3),  
GENDER CHAR(1) NOT NULL,  
PRIMARY KEY (STAFFID));
```

These are the names of the columns to be created within the table.

So far you have looked at creating a result set of data from existing tables but there are many more tasks you will undoubtedly need to perform over time.

The CREATE TABLE statement is used to create the structure into which data is added. A number of data value types can be assigned to columns you define within the table.

**Mouse-over** the CREATE TABLE coding for a description of its purpose.





```
INSERT INTO RESULT.TAB
VALUES ('002', 'SALES', '109', 'SA', 'UNICORN');
```

This will create a new row within the RESULT.TAB table and add the specified values to the existing columns.

```
INSERT INTO EMP.DAT (DEPTNO, DEPTNAME, MGRNO, SALARY)
VALUES ('020', 'PRODUCTION', '7099', '52000');
```

This will create a new row in the EMP.DAT table but only add the values to the corresponding column names specified in the first line.

```
INSERT INTO SALES.EMP
SELECT * FROM MARKET.EMP;
```

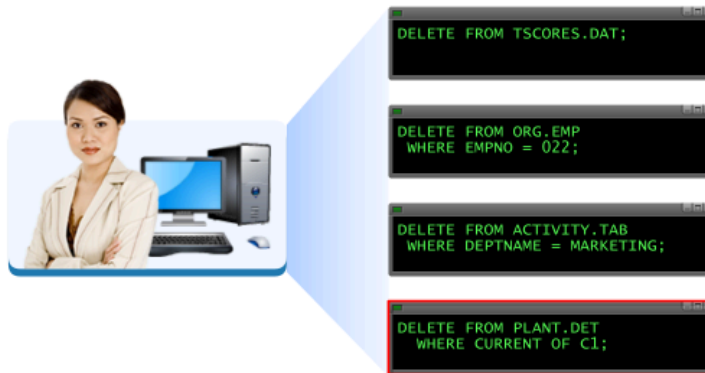
This will insert all the data from the MARKET.EMP table into the SALES.EMP table.

```
INSERT INTO ORDER.TAB
SELECT * FROM OSORDER.TAB
WHERE LANG = 'ENGLISH';
```

This will only select the records from the OSORDER.TAB table that contain ENGLISH in the LANG column, and then insert them into the ORDER.TAB table.

---

There are several ways of inserting data into a table. You can add a single row of data by specifying values to be inserted into columns, or populate an entire table by inserting all or selected rows from another table.



In this example, a previous statement has declared C1 as the cursor. This DELETE statement will delete the row on which the cursor C1 is currently positioned.

Just as rows can be added, they can also be removed using the DELETE statement. A search argument can be provided to remove specific rows or they can be deleted relative to where the cursor is positioned.

**Mouse-over** the DELETE statements for a description of their purpose.

```
UPDATE USORG.EMP  
SET MANAGER = 'YOUNG'  
WHERE DEPTNO = '005';
```

This will update only those records in table USORG.EMP where the DEPTNO column contains the 005 value. In those records, the value in the MANAGER column will be changed to YOUNG.



```
UPDATE TAX.EMP  
SET SALARY = 1.05 * SALARY;
```

This will update all records in table TAX.EMP changing the value in the SALARY column to reflect a 5% increase.

---

When existing table data needs to be modified, the UPDATE statement can be used. This statement can update data on one or multiple rows and can be used in conjunction with search criteria.

You can add data to a table using the INSERT statement.

You know how a basic table is created.



You know how to delete a single row or multiple rows of data.

You can update values in a table.

---

In this section you have looked at several statements that can be used to create tables and table data, remove unwanted data and update existing data. SQL modules in Interskill's programming curriculum expand the use of these statements and introduce you to other statements and keywords used to manipulate tables and indexes.



## Summary

---

### Overview of SQL

In this module, you looked at the SELECT statement and the clauses, predicates and functions used to define specific data to be obtained. You also looked at the syntax of the code used to create a table, insert and delete data and modify existing data.

You should now be able to:

- Code a SELECT Statement to Obtain Table Data
- Identify SQL Statement Clauses and Predicates That Allow you to Define Specific Data to be Selected
- Explain the Function of the CREATE TABLE, INSERT, DELETE and UPDATE Statements