**PRACTICAL - 5**

**AIM: Displaying data from Multiple Tables (join)**

**THEORY:**

**JOIN IN SQL**

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. **Join** Keyword is used in SQL queries for joining two or more tables.

**DIFFERENT TYPES OF JOIN**

* (INNER) JOIN: Select records that have matching values in both tables.
* LEFT (OUTER) JOIN: Select records from the first (left-most) table with matching right table records.
* RIGHT (OUTER) JOIN: Select records from the second (right-most) table with matching left table records.
* FULL (OUTER) JOIN: Selects all records that match either left or right table records.

All INNER and OUTER keywords are optional.

**INNER JOIN (SIMPLE JOIN)**

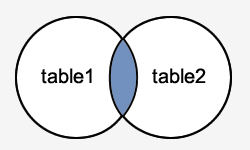
It is the most common type of SQL join. SQL INNER JOINS return all rows from multiple tables where the join condition is met.

**Syntax:**

SELECT COLUMN-NAMES FROM TABLE-NAME1 JOIN TABLE-NAME2 ON COLUMN-NAME1 = COLUMN-NAME2 WHERE CONDITION

**Visual Illustration:**

In this visual diagram, the SQL INNER JOIN returns the shaded area:



The SQL INNER JOIN would return the records where *table1* and *table2* intersect.

**Example:**

We have a table called *customers* with the following data:

|  |  |  |  |
| --- | --- | --- | --- |
| **customer\_id** | **last\_name** | **first\_name** | **favorite\_website** |
| 4000 | Jackson | Joe | techonthenet.com |
| 5000 | Smith | Jane | digminecraft.com |
| 6000 | Ferguson | Samantha | bigactivities.com |
| 7000 | Reynolds | Allen | checkyourmath.com |
| 8000 | Anderson | Paige | NULL |
| 9000 | Johnson | Derek | techonthenet.com |

And a table called *orders* with the following data:

|  |  |  |
| --- | --- | --- |
| **order\_id** | **customer\_id** | **order\_date** |
| 1 | 7000 | 2016/04/18 |
| 2 | 5000 | 2016/04/18 |
| 3 | 8000 | 2016/04/19 |
| 4 | 4000 | 2016/04/20 |
| 5 | NULL | 2016/05/01 |

Enter the following SQL statement:

SELECT CUSTOMERS.CUSTOMER\_ID, ORDERS.ORDER\_ID, ORDERS.ORDER\_DATEFROM CUSTOMERS INNER JOIN ORDERSON CUSTOMERS.CUSTOMER\_ID = ORDERS.CUSTOMER\_IDORDER BY CUSTOMERS.CUSTOMER\_ID;

There will be 4 records selected. These are the results that you should see:

|  |  |  |
| --- | --- | --- |
| customer\_id | order\_id | order\_date |
| 4000 | 4 | 2016/04/20 |
| 5000 | 2 | 2016/04/18 |
| 7000 | 1 | 2016/04/18 |
| 8000 | 3 | 2016/04/19 |

This example would return all rows from the *customers* and *orders* tables where there is a matching *customer\_id* value in both the *customers* and *orders* tables.

The rows where *customer\_id* is equal to 6000 and 9000 in the *customers* table would be omitted, since they do not exist in both tables. The row where the *order\_id* is 5 from the *orders* table would be omitted, since the *customer\_id* of NULL does not exist in the *customers* table.

**Old Syntax**

The above query be rewritten using the older implicit syntax as follows

SELECT CUSTOMERS.CUSTOMER\_ID, ORDERS.ORDER\_ID, ORDERS.ORDER\_DATEFROM CUSTOMERS, ORDERSWHERE CUSTOMERS.CUSTOMER\_ID = ORDERS.CUSTOMER\_IDORDER BY CUSTOMERS.CUSTOMER\_ID;

OR

SELECT C.CUSTOMER\_ID, O.ORDER\_ID, O.ORDER\_DATEFROM CUSTOMERS C, ORDERS O WHERE C.CUSTOMER\_ID = O.CUSTOMER\_ID ORDER BY C.CUSTOMER\_ID;

**LEFT (OUTER) JOIN**

This type of join returns all rows from the LEFT-hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal (join condition is met).

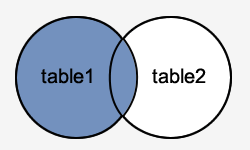
**Syntax:**

SELECT COLUMNSFROM TABLE1LEFT [OUTER] JOIN TABLE2ON TABLE1.COLUMN = TABLE2.COLUMN;

In some databases, the OUTER keyword is omitted and written simply as LEFT JOIN.

**Visual Illustration:**

In this visual diagram, the SQL LEFT OUTER JOIN returns the shaded area:



The SQL LEFT OUTER JOIN would return the all records from *table1* and only those records from *table2* that intersect with *table1*.

**Example:**

Using the same *customers* table as the previous example:

|  |  |  |  |
| --- | --- | --- | --- |
| **customer\_id** | **last\_name** | **first\_name** | **favorite\_website** |
| 4000 | Jackson | Joe | techonthenet.com |
| 5000 | Smith | Jane | digminecraft.com |
| 6000 | Ferguson | Samantha | bigactivities.com |
| 7000 | Reynolds | Allen | checkyourmath.com |
| 8000 | Anderson | Paige | NULL |
| 9000 | Johnson | Derek | techonthenet.com |

And the *orders* table with the following data:

|  |  |  |
| --- | --- | --- |
| **order\_id** | **customer\_id** | **order\_date** |
| 1 | 7000 | 2016/04/18 |
| 2 | 5000 | 2016/04/18 |
| 3 | 8000 | 2016/04/19 |
| 4 | 4000 | 2016/04/20 |
| 5 | NULL | 2016/05/01 |

Enter the following SQL statement:

SELECT CUSTOMERS.CUSTOMER\_ID, ORDERS.ORDER\_ID, ORDERS.ORDER\_DATE FROM CUSTOMERS LEFT OUTER JOIN ORDERSON CUSTOMERS.CUSTOMER\_ID = ORDERS.CUSTOMER\_IDORDER BY CUSTOMERS.CUSTOMER\_ID;

There will be 6 records selected. These are the results that you should see:

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **order\_id** | **order\_date** |
| 4000 | 4 | 2016/04/20 |
| 5000 | 2 | 2016/04/18 |
| 6000 | NULL | NULL |
| 7000 | 1 | 2016/04/18 |
| 8000 | 3 | 2016/04/19 |
| 9000 | NULL | NULL |

This LEFT OUTER JOIN example would return all rows from the *customers* table and only those rows from the *orders* table where the joined fields are equal.

If a *customer\_id* value in the *customers* table does not exist in the *orders* table, all fields in the *orders* table will display as NULL in the result set. As you can see, the rows where *customer\_id* is 6000 and 9000 would be included with a LEFT OUTER JOIN but the *order\_id* and *order\_date* fields display NULL.

**RIGHT (OUTER) JOIN**

Another type of join is called a SQL RIGHT OUTER JOIN. This type of join returns all rows from the RIGHT-hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal (join condition is met).

**Syntax:**

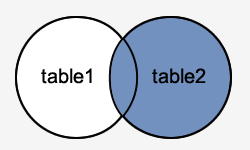
The syntax for the RIGHT OUTER JOIN in SQL is:

SELECT COLUMNSFROM TABLE1RIGHT [OUTER] JOIN TABLE2ON TABLE1.COLUMN = TABLE2.COLUMN;

In some databases, the OUTER keyword is omitted and written simply as RIGHT JOIN.

**Visual Illustration:**

In this visual diagram, the SQL RIGHT OUTER JOIN returns the shaded area:



The SQL RIGHT OUTER JOIN would return the all records from *table2* and only those records from *table1* that intersect with *table2*.

**Example:**

Using the same *customers* table as the previous example:

|  |  |  |  |
| --- | --- | --- | --- |
| **customer\_id** | **last\_name** | **first\_name** | **favorite\_website** |
| 4000 | Jackson | Joe | techonthenet.com |
| 5000 | Smith | Jane | digminecraft.com |
| 6000 | Ferguson | Samantha | bigactivities.com |
| 7000 | Reynolds | Allen | checkyourmath.com |
| 8000 | Anderson | Paige | NULL |
| 9000 | Johnson | Derek | techonthenet.com |

And the *orders* table with the following data:

|  |  |  |
| --- | --- | --- |
| **order\_id** | **customer\_id** | **order\_date** |
| 1 | 7000 | 2016/04/18 |
| 2 | 5000 | 2016/04/18 |
| 3 | 8000 | 2016/04/19 |
| 4 | 4000 | 2016/04/20 |
| 5 | NULL | 2016/05/01 |

Enter the following SQL statement:

SELECT CUSTOMERS.CUSTOMER\_ID, ORDERS.ORDER\_ID, ORDERS.ORDER\_DATE FROM CUSTOMERS RIGHT OUTER JOIN ORDERSON CUSTOMERS.CUSTOMER\_ID = ORDERS.CUSTOMER\_IDORDER BY CUSTOMERS.CUSTOMER\_ID;

There will be 5 records selected. These are the results that you should see:

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **order\_id** | **order\_date** |
| NULL | 5 | 2016/05/01 |
| 4000 | 4 | 2016/04/20 |
| 5000 | 2 | 2016/04/18 |
| 7000 | 1 | 2016/04/18 |
| 8000 | 3 | 2016/04/19 |

This RIGHT OUTER JOIN example would return all rows from the *orders* table and only those rows from the *customers* table where the joined fields are equal.

If a *customer\_id* value in the *orders* table does not exist in the *customers* table, all fields in the *customers* table will display as NULL in the result set. As you can see, the row where *order\_id* is 5 would be included with a RIGHT OUTER JOIN but the *customer\_id* field displays NULL.

**FULL (OUTER) JOIN**

Another type of join is called a SQL FULL OUTER JOIN. This type of join returns all rows from the LEFT-hand table and RIGHT-hand table with NULL values in place where the join condition is not met.

**Syntax:**

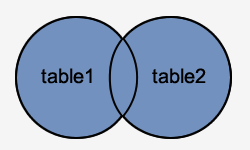
The syntax for the SQL **FULL OUTER JOIN** is:

SELECT COLUMNSFROM TABLE1FULL [OUTER] JOIN TABLE2ON TABLE1.COLUMN = TABLE2.COLUMN;

In some databases, the OUTER keyword is omitted and written simply as FULL JOIN.

**Visual Illustration:**

In this visual diagram, the SQL FULL OUTER JOIN returns the shaded area:



The SQL FULL OUTER JOIN would return the all records from both *table1* and *table2*.

**Example:**

Using the same *customers* table as the previous example:

|  |  |  |  |
| --- | --- | --- | --- |
| **customer\_id** | **last\_name** | **first\_name** | **favorite\_website** |
| 4000 | Jackson | Joe | techonthenet.com |
| 5000 | Smith | Jane | digminecraft.com |
| 6000 | Ferguson | Samantha | bigactivities.com |
| 7000 | Reynolds | Allen | checkyourmath.com |
| 8000 | Anderson | Paige | NULL |
| 9000 | Johnson | Derek | techonthenet.com |

And the *orders* table with the following data:

|  |  |  |
| --- | --- | --- |
| **order\_id** | **customer\_id** | **order\_date** |
| 1 | 7000 | 2016/04/18 |
| 2 | 5000 | 2016/04/18 |
| 3 | 8000 | 2016/04/19 |
| 4 | 4000 | 2016/04/20 |
| 5 | NULL | 2016/05/01 |

Enter the following SQL statement:

SELECT CUSTOMERS.CUSTOMER\_ID, ORDERS.ORDER\_ID, ORDERS.ORDER\_DATEFROM CUSTOMERS FULL OUTER JOIN ORDERSON CUSTOMERS.CUSTOMER\_ID = ORDERS.CUSTOMER\_IDORDER BY CUSTOMERS.CUSTOMER\_ID;

There will be 7 records selected. These are the results that you should see:

|  |  |  |
| --- | --- | --- |
| **customer\_id** | **order\_id** | **order\_date** |
| NULL | 5 | 2016/05/01 |
| 4000 | 4 | 2016/04/20 |
| 5000 | 2 | 2016/04/18 |
| 6000 | NULL | NULL |
| 7000 | 1 | 2016/04/18 |
| 8000 | 3 | 2016/04/19 |
| 9000 | NULL | NULL |

This FULL OUTER JOIN example would return all rows from the *orders* table and all rows from the *customers* table. Whenever the joined condition is not met, a NULL value would be extended to those fields in the result set. This means that if a *customer\_id* value in the *customers* table does not exist in the *orders* table, all fields in the *orders* table will display as NULL in the result set. Also, if a *customer\_id* value in the *orders* table does not exist in the *customers* table, all fields in the *customers* table will display as NULL in the result set.

As you can see, the rows where the *customer\_id* is 6000 and 9000 would be included but the *order\_id* and *order\_date* fields for those records contains a NULL value. The row where the *order\_id* is 5 would be also included but the *customer\_id* field for that record has a NULL value.

**PROGRAM EXECUTION:**

**Creating Tables:**

To create table Department:

|  |
| --- |
| C:\Users\Desktop\Desktop\photo\DEPARTMENT.PNG  C:\Users\Desktop\Desktop\photo\DEPATMENT_VALUE.PNG |

**Inserting Data into Tables:**

Insert Data in Employee:

|  |
| --- |
| 1  2 |

**Queries to be performed:**

* 1. Give details of customers ANIL.

|  |
| --- |
| SELECT D.ACTNO,D.CNAME,D.BNAME,D.AMOUNT,D.ADATE, C.CITY FROM DEPOSIT D JOIN CUSTOMER C ON D.CNAME=’ANIL’; |

**Output:**

|  |
| --- |
| C:\Users\Desktop\Desktop\photo\P1.PNG |

* 1. Give name of customer who are borrowers and depositors and having living city nagpur.

|  |
| --- |
| SELECT D.CNAME,C.CITY,B.LOAN\_NO FROM DEPOSIT D JOIN CUSTOMER C ON D.CNAME=C.CNAME JOIN BORROW B ON C.CNAME=B.CNAME WHERE C.CITY=’NAGPUR’; |

**Output:**

|  |
| --- |
| C:\Users\Desktop\Desktop\photo\P2.PNG |

* 1. Give city as their city name of customers having same living branch.

|  |
| --- |
| SELECT C.CNAME,C.CITY FROM CUSTOMER C JOIN DEPOSIT D ON D.CNAME=C.CNAME; |

**Output:**

|  |
| --- |
| C:\Users\Desktop\Desktop\photo\P3.PNG |

* 1. Write a query to display the last name, department number, and department name for all employees.

|  |
| --- |
| SELECT E.NAME,D.DEPTNO,D.DNAME FROM EMPLOYEE1 E LEFT JOIN DEPARTMENT D ON E.DEPTNO=D.DEPTNO; |

**Output:**

|  |
| --- |
| C:\Users\Desktop\Desktop\photo\P4.PNG |

* 1. Create a unique listing of all jobs that are in department 30. Include the location of the department in the output.

|  |
| --- |
| SELECT DISTINCT E.JOB,D.LOC FROM EMPLOYEE1 E JOIN DEPARTMENT D ON E.DEPTNO=D.DEPTNO WHERE E.DEPTNO=30; |

**Output:**

|  |
| --- |
| C:\Users\Desktop\Desktop\photo\P5.PNG |

* 1. Write a query to display the employee name, department number, and department name for all employees who work in NEW YORK.

|  |
| --- |
| SELECT E.NAME,D.DEPTNO,D.DNAME,D.LOC FROM EMPLOYEE1 E JOIN DEPARTMENT D ON E.DEPTNO=D.DEPTNO WHERE D.LOC=’NEW YORK’; |

**Output:**

|  |
| --- |
| **C:\Users\Desktop\Desktop\photo\P6.PNG** |

* 1. Display the employee last name and employee number along with their manager’s last name and manager number. Label the columns Employee, Emp#, Manager, and Mgr#, respectively.

|  |
| --- |
| SELECT E1.NAME “EMPLOYEE”,E1.EMPNO “EMP#”,E1.MGR “MGR#”,E2.NAME “MANAGER” FROM EMPLOYEE1 E1 LEFT OUTER JOIN EMPLOYEE1 E2 ON E1.MGR=E2.EMPNO; |

**Output:**

|  |
| --- |
| **C:\Users\Desktop\Desktop\photo\P7.PNG** |

* 1. Create a query to display the name and hire date of any employee hired after employee SCOTT**.**

|  |
| --- |
| SELECT A.NAME,A.HIREDATE FROM EMPLOYEE1 A JOIN EMPLOYEE1 B ON A.HIREDATE>B.HIREDATE WHERE B.NAME=”SCOTT”; |

**Output:**

|  |
| --- |
| C:\Users\Desktop\Desktop\photo\P8.PNG |

**VIVA QUESTIONS:**

* **You want to display a result query from joining two tables with 20 and 10 rows respectively. Erroneously you forget to write the WHERE clause. What would be the result?**
* The Cartesian product of two tables with 20 x 10 = 200 rows.
* **What is the difference between cross joins and natural joins?**
* Natural Join joins two tables based on same attribute name and datatypes. The resulting table will contain all the attributes of both the tables but only one copy of each common column.
* Cross Join will produce cross or Cartesian product of two tables if there is no condition specifies. The resulting table will contain all the attributes of both the tables including duplicate or common columns also.