EX NO.: 04 SUB QUERIES AND JOINS

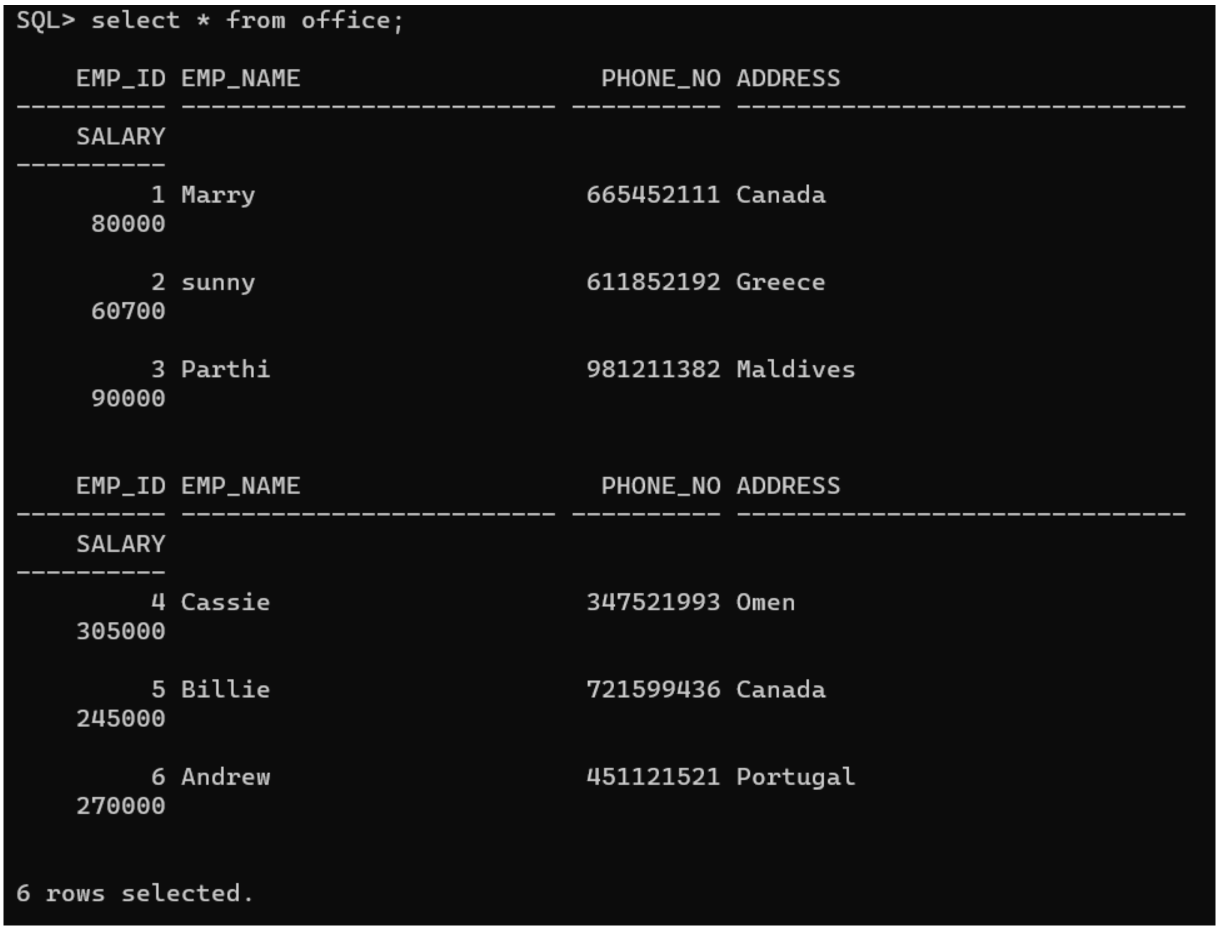
AIM:

To work with Sub queries and joins

SUB QUERIES:

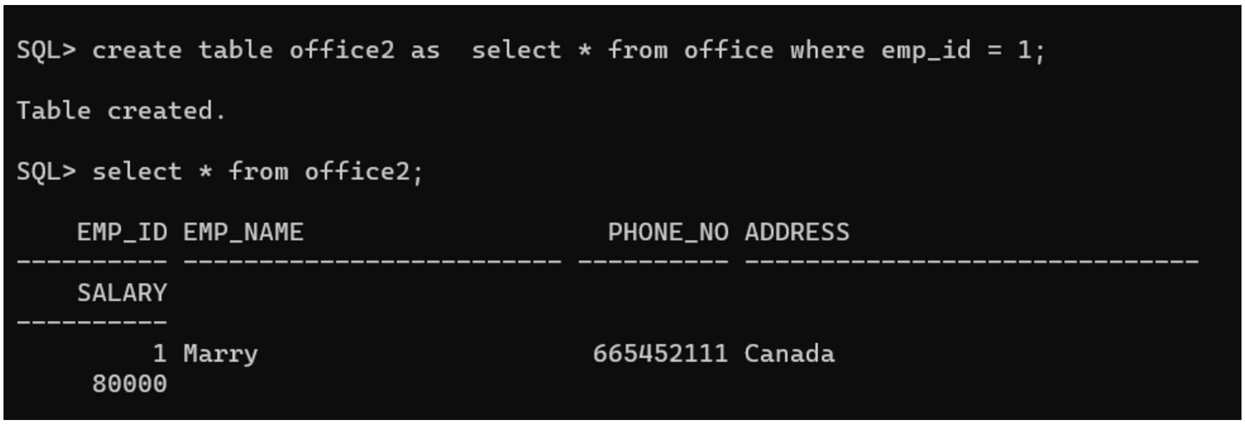
In SQL a Subquery can be simply defined as a query within another query. In other words we can say that a Subquery is a query that is embedded in WHERE clause of another SQL query.

TABLE 1 – OFFICE:



```
SQL> select * from office;

    EMP_ID EMP_NAME                    PHONE_NO ADDRESS
---------- ------------------------- ---------- ------------------------------
    SALARY
----------
         1 Marry                      665452111 Canada
     80000

         2 sunny                      611852192 Greece
     60700

         3 Parthi                     981211382 Maldives
     90000


    EMP_ID EMP_NAME                    PHONE_NO ADDRESS
---------- ------------------------- ---------- ------------------------------
    SALARY
----------
         4 Cassie                     347521993 Omen
    305000

         5 Billie                     721599436 Canada
    245000

         6 Andrew                     451121521 Portugal
    270000


6 rows selected.
```

TABLE 2 – OFFICE 2:



```
SQL> create table office2 as  select * from office where emp_id = 1;

Table created.

SQL> select * from office2;

    EMP_ID EMP_NAME                        PHONE_NO ADDRESS
---------- ------------------------- ---------- ------------------------------
    SALARY
----------
         1 Marry                          665452111 Canada
     80000
```

SUB QUERIES WITH SELECT STATEMENT:

Subqueries are most frequently used with the SELECT statement.

The basic syntax is as follows –

SELECT column_name [, column_name]

FROM   table1 [, table2]

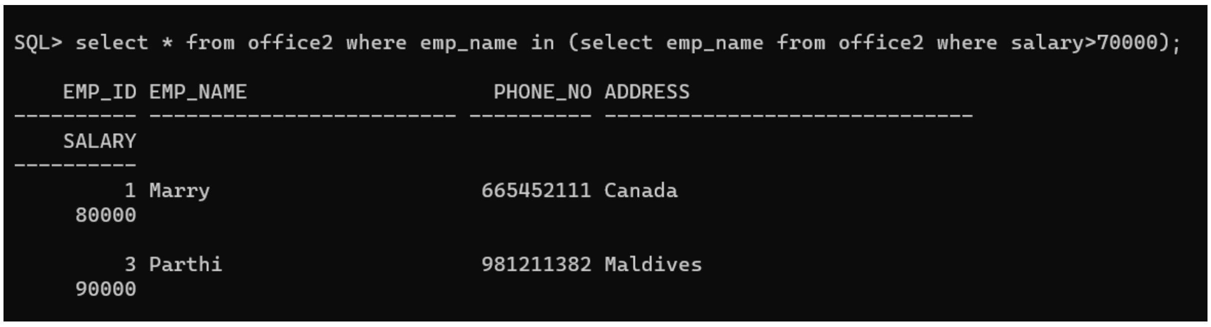
WHERE column_name OPERATOR

(SELECT column_name [, column_name]

FROM table1 [, table2]

[WHERE])

OUTPUT:



```
SQL> select * from office2 where emp_name in (select emp_name from office2 where salary>70000);

    EMP_ID EMP_NAME                        PHONE_NO ADDRESS
---------- ------------------------- ---------- ------------------------------
    SALARY
----------
         1 Marry                          665452111 Canada
     80000

         3 Parthi                         981211382 Maldives
     90000
```

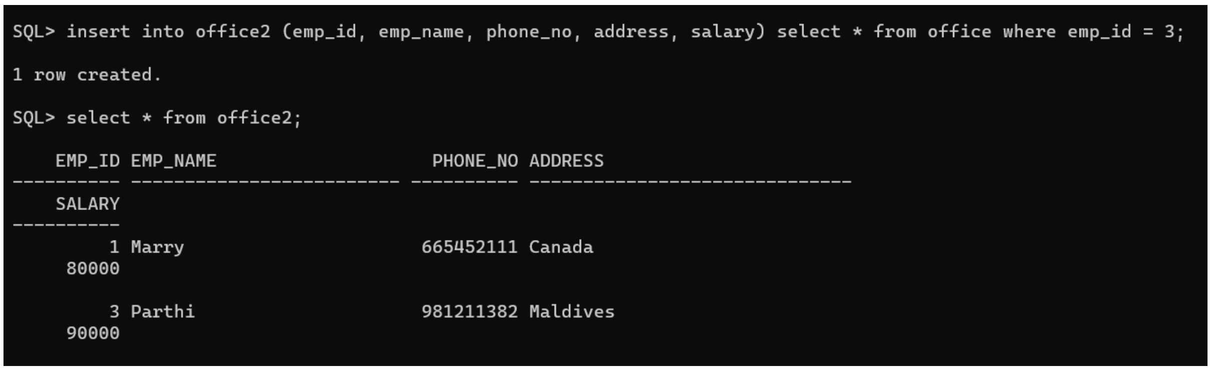
SUB QUERIES WITH INSERT STATEMENT:

Subqueries also can be used with INSERT statements. The INSERT statement uses the data returned from the subquery to insert into another table. The selected data in the subquery can be modified with any of the character, date or number functions.

The basic syntax is as follows.

```
INSERT INTO table_name [ (column1 [, column2 ]) ]
  SELECT [ *|column1 [, column2 ]
  FROM table1 [, table2 ]
  [ WHERE VALUE OPERATOR ]
```

OUTPUT:



```
SQL> insert into office2 (emp_id, emp_name, phone_no, address, salary) select * from office where emp_id = 3;

1 row created.

SQL> select * from office2;

    EMP_ID EMP_NAME                  PHONE_NO ADDRESS
---------- ------------------------- ---------- ------------------------------
    SALARY
----------
         1 Marry                      665452111 Canada
     80000

         3 Parthi                     981211382 Maldives
     90000
```

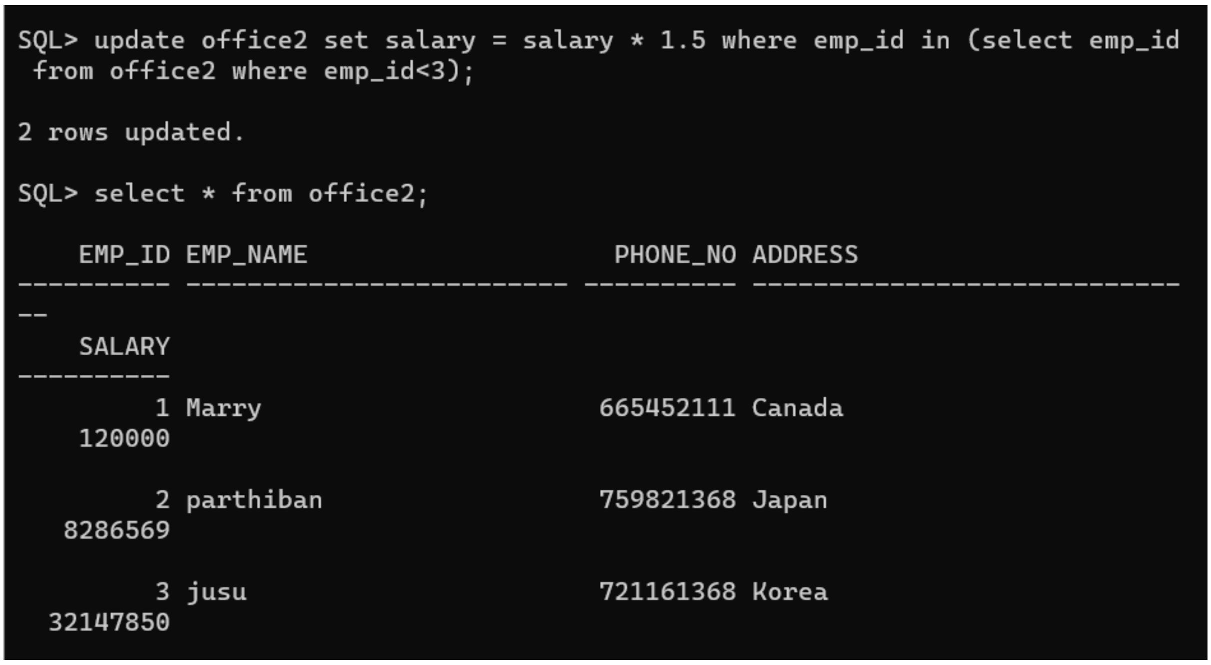
SUB QUERIES WITH UPDATE STATEMENT:

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

The basic syntax is as follows

```
UPDATE table
SET column_name = new_value
[ WHERE OPERATOR [ VALUE ]
  (SELECT COLUMN_NAME
  FROM TABLE_NAME)
  [ WHERE) ]
```

OUTPUT:



```
SQL> update office2 set salary = salary * 1.5 where emp_id in (select emp_id
 from office2 where emp_id<3);

2 rows updated.

SQL> select * from office2;

    EMP_ID EMP_NAME                  PHONE_NO ADDRESS
---------- ------------------------- ---------- ----------------------------
--
    SALARY
----------
         1 Marry                      665452111 Canada
    120000

         2 parthiban                  759821368 Japan
   8286569

         3 jusu                       721161368 Korea
  32147850
```

SUB QUERIES WITH DELETE STATEMENT:

The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above.

The basic syntax is as follows.

DELETE FROM TABLE_NAME

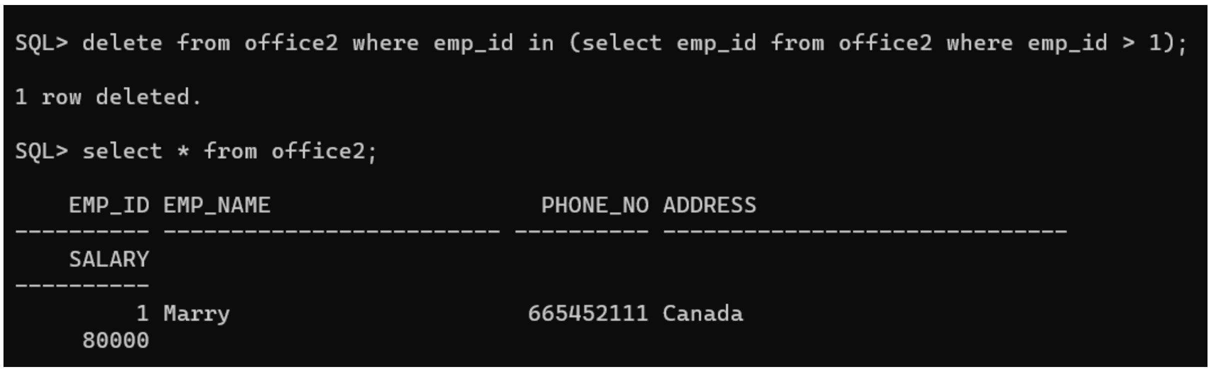
[WHERE OPERATOR [VALUE]

  (SELECT COLUMN_NAME

  FROM TABLE_NAME)

  [WHERE)]

OUTPUT:



```
SQL> delete from office2 where emp_id in (select emp_id from office2 where emp_id > 1);

1 row deleted.

SQL> select * from office2;

    EMP_ID EMP_NAME                  PHONE_NO ADDRESS
---------- ------------------------- ---------- ------------------------------
    SALARY
----------
         1 Marry                      665452111 Canada
     80000
```

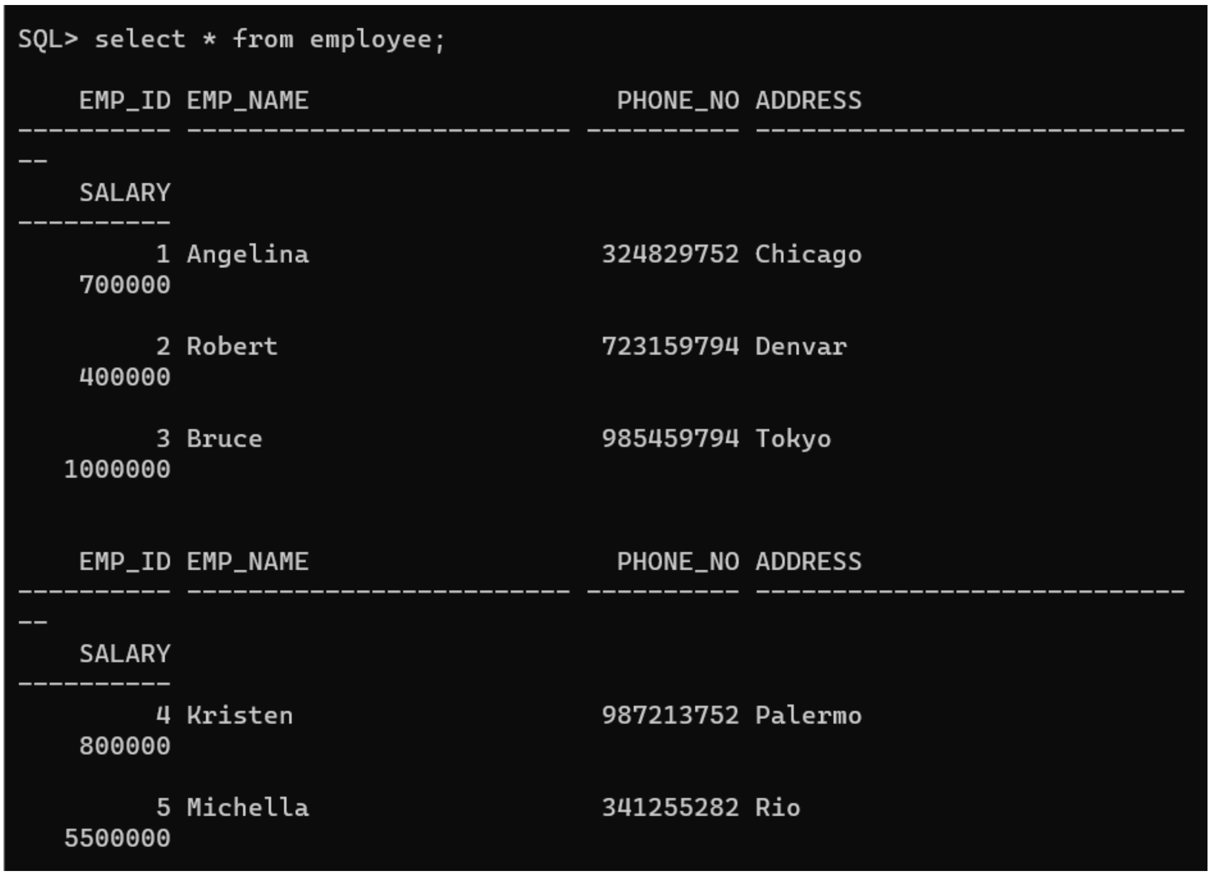
JOINS:

Different types of Joins are as follows:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

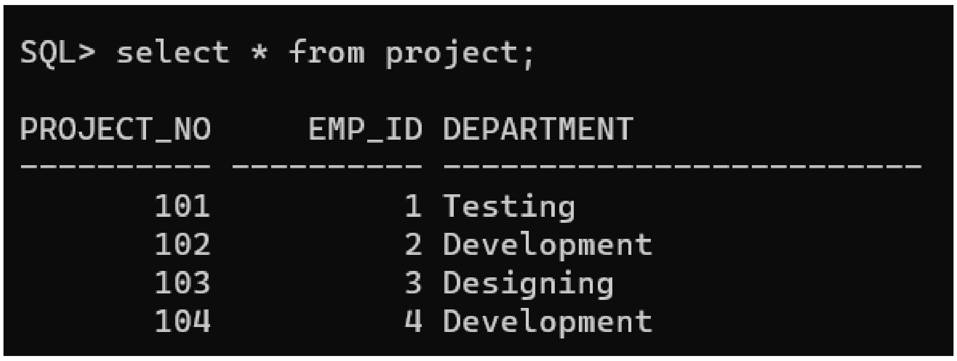
Consider the two tables below:

EMPLOYEE TABLE:



```
SQL> select * from employee;

    EMP_ID EMP_NAME                    PHONE_NO ADDRESS
---------- ------------------------- ---------- ----------------------------
--
    SALARY
----------
         1 Angelina                   324829752 Chicago
    700000

         2 Robert                     723159794 Denvar
    400000

         3 Bruce                      985459794 Tokyo
   1000000


    EMP_ID EMP_NAME                    PHONE_NO ADDRESS
---------- ------------------------- ---------- ----------------------------
--
    SALARY
----------
         4 Kristen                    987213752 Palermo
    800000

         5 Michella                   341255282 Rio
   5500000
```

PROJECT TABLE:



```
SQL> select * from project;

PROJECT_NO     EMP_ID DEPARTMENT
---------- ---------- -------------------------
       101          1 Testing
       102          2 Development
       103          3 Designing
       104          4 Development
```

A. INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

INNER JOIN table2

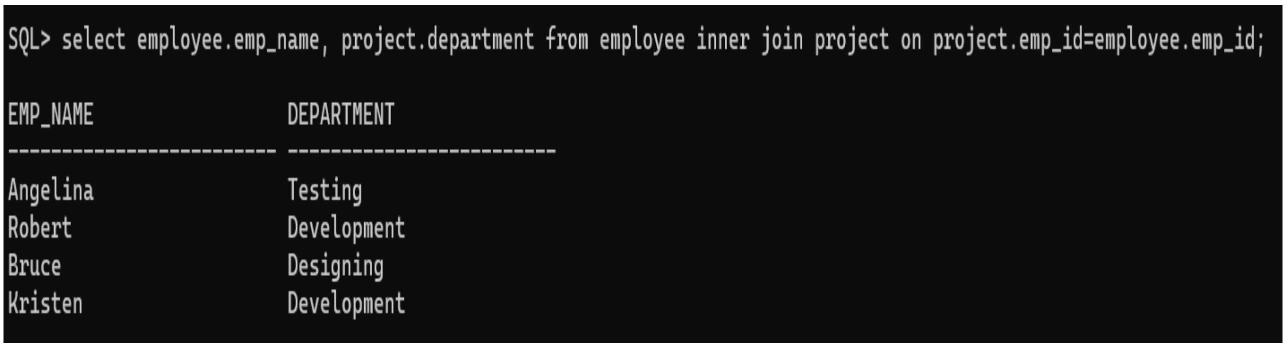
ON table1.matching_column = table2.matching_column;

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

OUTPUT:



```
SQL> select employee.emp_name, project.department from employee inner join project on project.emp_id=employee.emp_id;

EMP_NAME                  DEPARTMENT
------------------------- -------------------------
Angelina                  Testing
Robert                    Development
Bruce                     Designing
Kristen                   Development
```

B. LEFT JOIN

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

LEFT JOIN table2

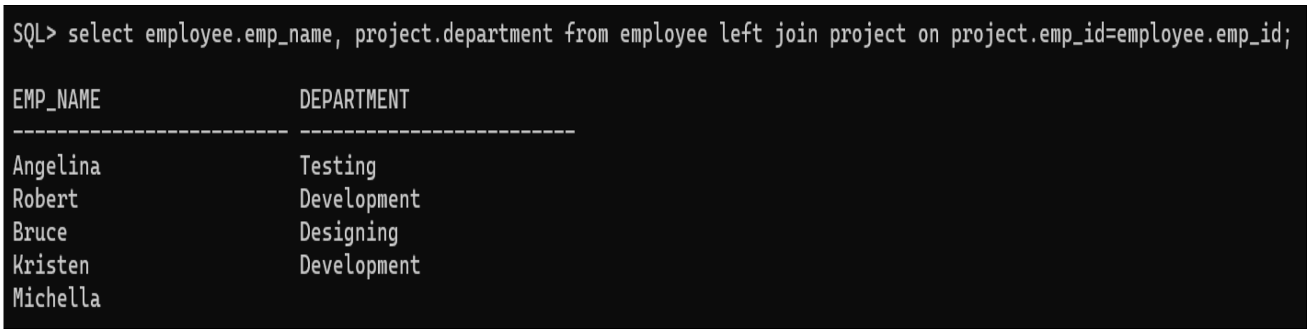
ON table1.matching_column = table2.matching_column;

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

OUTPUT:



```
SQL> select employee.emp_name, project.department from employee left join project on project.emp_id=employee.emp_id;

EMP_NAME                  DEPARTMENT
------------------------- ------------------------
Angelina                  Testing
Robert                    Development
Bruce                     Designing
Kristen                   Development
Michella
```

C. RIGHT JOIN

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

RIGHT JOIN table2

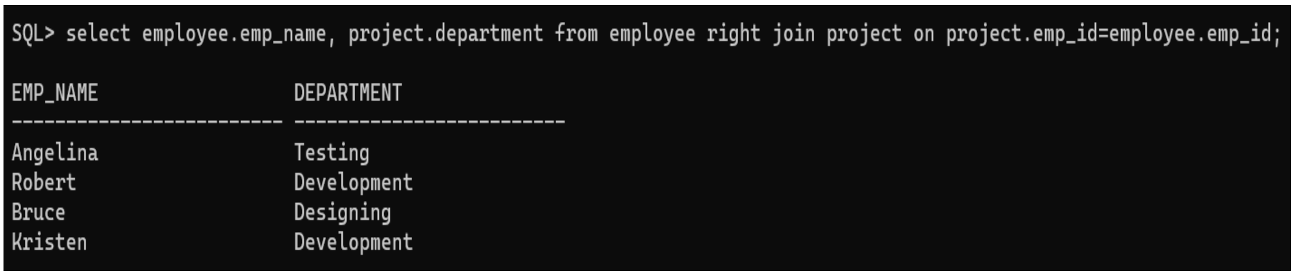
ON table1.matching_column = table2.matching_column;

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

OUTPUT:



```
SQL> select employee.emp_name, project.department from employee right join project on project.emp_id=employee.emp_id;

EMP_NAME                 DEPARTMENT
------------------------ ------------------------
Angelina                 Testing
Robert                   Development
Bruce                    Designing
Kristen                  Development
```

D. FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.

Syntax:

SELECT table1.column1,table1.column2,table2.column1,....

FROM table1

FULL JOIN table2

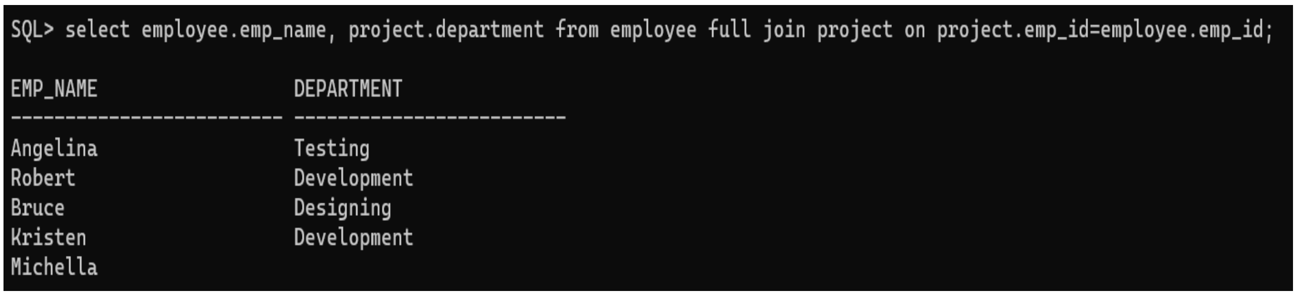
ON table1.matching_column = table2.matching_column;

table1: First table.

table2: Second table

matching_column: Column common to both the tables.

OUTPUT:



```
SQL> select employee.emp_name, project.department from employee full join project on project.emp_id=employee.emp_id;

EMP_NAME                 DEPARTMENT
------------------------ ------------------------
Angelina                 Testing
Robert                   Development
Bruce                    Designing
Kristen                  Development
Michella
```

RESULT:

The queries for Sub queries and Joins were successfully executed and the output is noted.