CSC-252: Database Management System
Semester IV(CS, SE) Section(A, B) (Spring 2020)
Course Instructor(s): Khalid Hussain

Lab 05: Working with JOINs

Objective(s):

1. Introduction of JOIN
2. INNER JON
3. LEFT JOIN
4. RIGHT JOIN
5. FULL OUTER JOIN/ FULL JOIN
6. CROSS JOIN
7. FULL JOIN vs CROSS JOIN

1: Introduction of JOIN

A relational database consists of multiple related tables linking together using common columns which are known as foreign key columns. Because of this, data in each table is incomplete from the business perspective.

By using joins, you can retrieve data from two or more tables based on logical relationships between the tables. Joins indicate how SQL should use data from one table to select the rows in another table.

A join condition defines the way two tables are related in a query by:

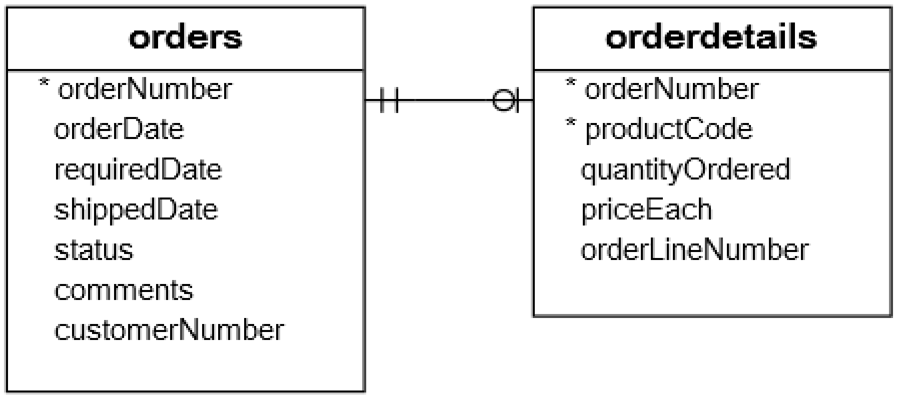
- Specifying the column from each table to be used for the join. A typical join condition specifies a foreign key from one table and its associated key in the other table.
- Specifying a logical operator (for example, = or <>,) to be used in comparing values from the columns.

There are various forms of the JOIN clause. These will include:

1. INNER JOIN
2. OUTER JOIN (both LEFT and RIGHT)
3. FULL OUTER JOIN
4. CROSS JOIN

A JOIN does just what it sounds like—it puts the information from two tables together into one result set. We can think of a result set as being a "virtual" table. It has both columns and rows, and the columns have data types. How exactly does a JOIN put the information from two tables into a single result set? Well, that depends on how you tell it to put the data together—that's why there are four different kinds of JOINs. The thing that all JOINs have in common is that they match one record up with one or more other records to make a record that is a superset created by the combined columns of both records.

For example we have the orders and orderdetails tables that are linked using the orderNumber column:

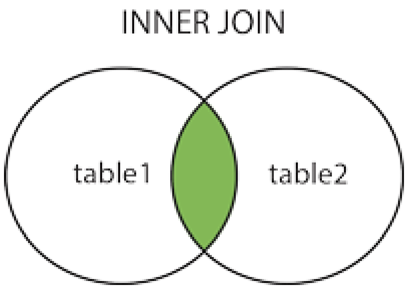


To get complete orders' information, you need to query data from both orders and orderdetails tables. That's why joins come into the play.

A join is a method of linking data between one (self-join) or more tables based on values of the common column between the tables.

2: INNER JOIN/JOIN

The INNER JOIN keyword selects records that have matching values in both tables.



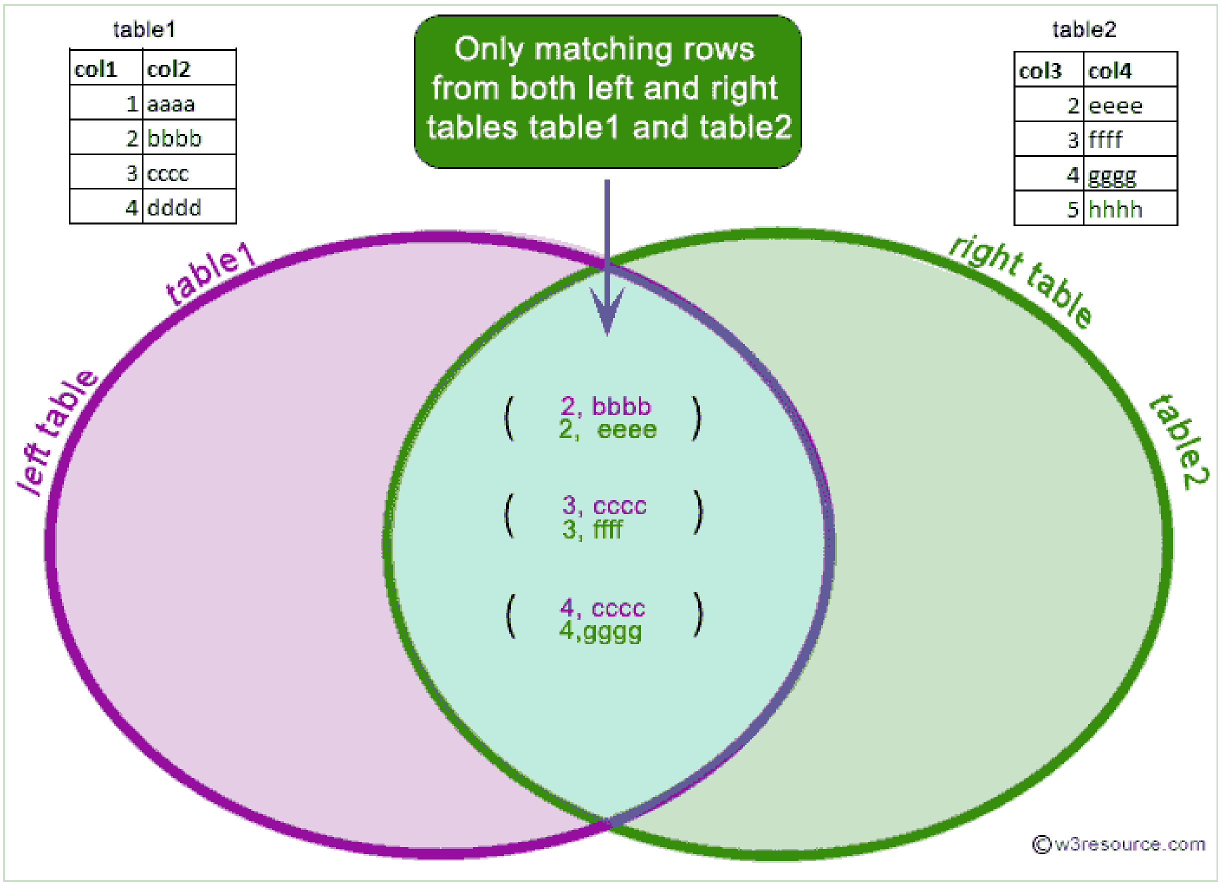
Syntax:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

If the join condition uses the equal operator (=) and the column names in both tables used for matching are the same, you can use the USING clause instead:

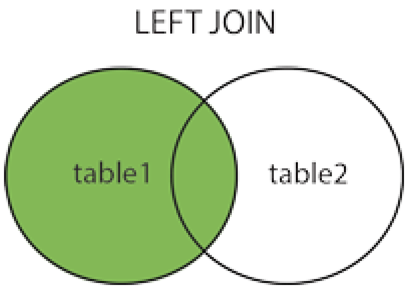
Syntax:

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
USING(column_name);
```



3: LEFT JOIN

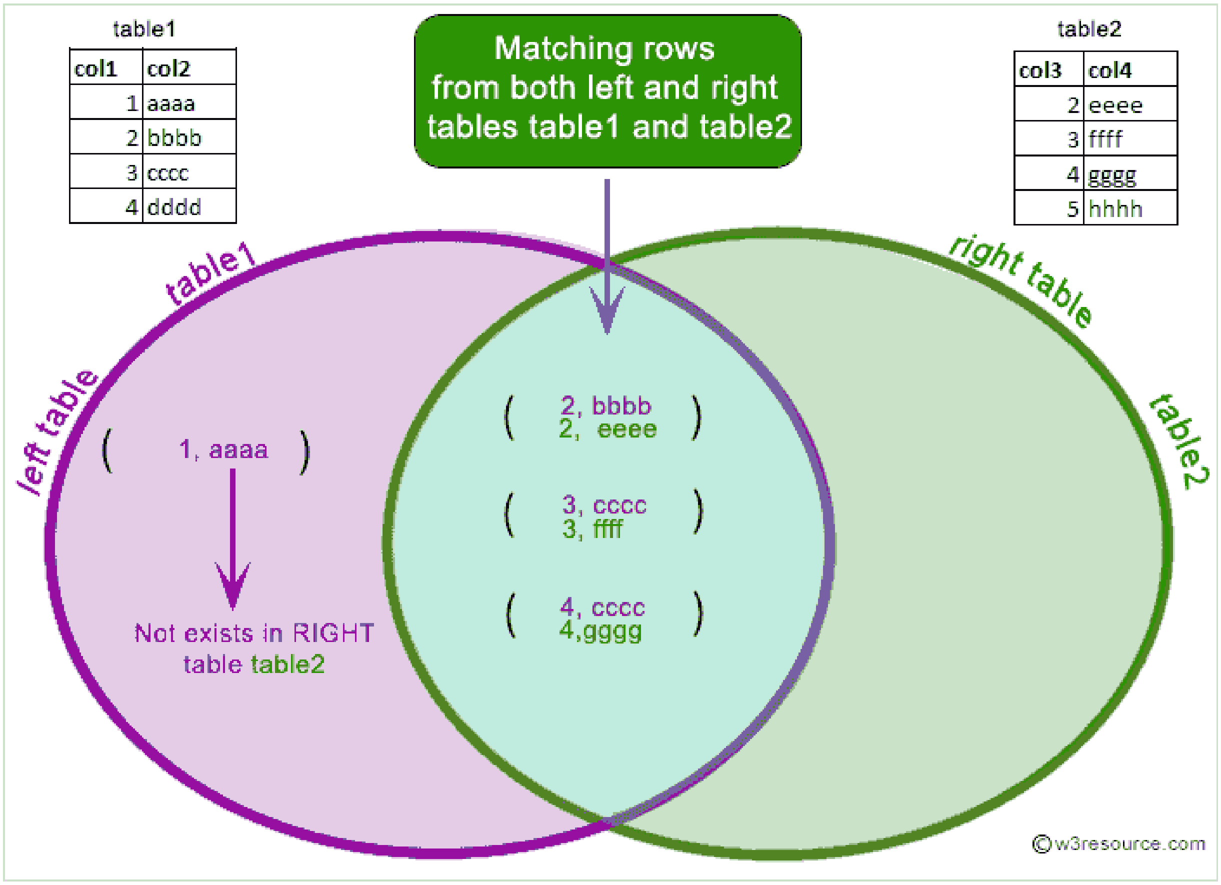
The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.



Syntax:

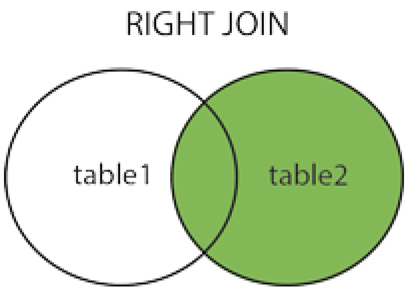
```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

Note: In some databases LEFT JOIN is called LEFT OUTER JOIN.



4: RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

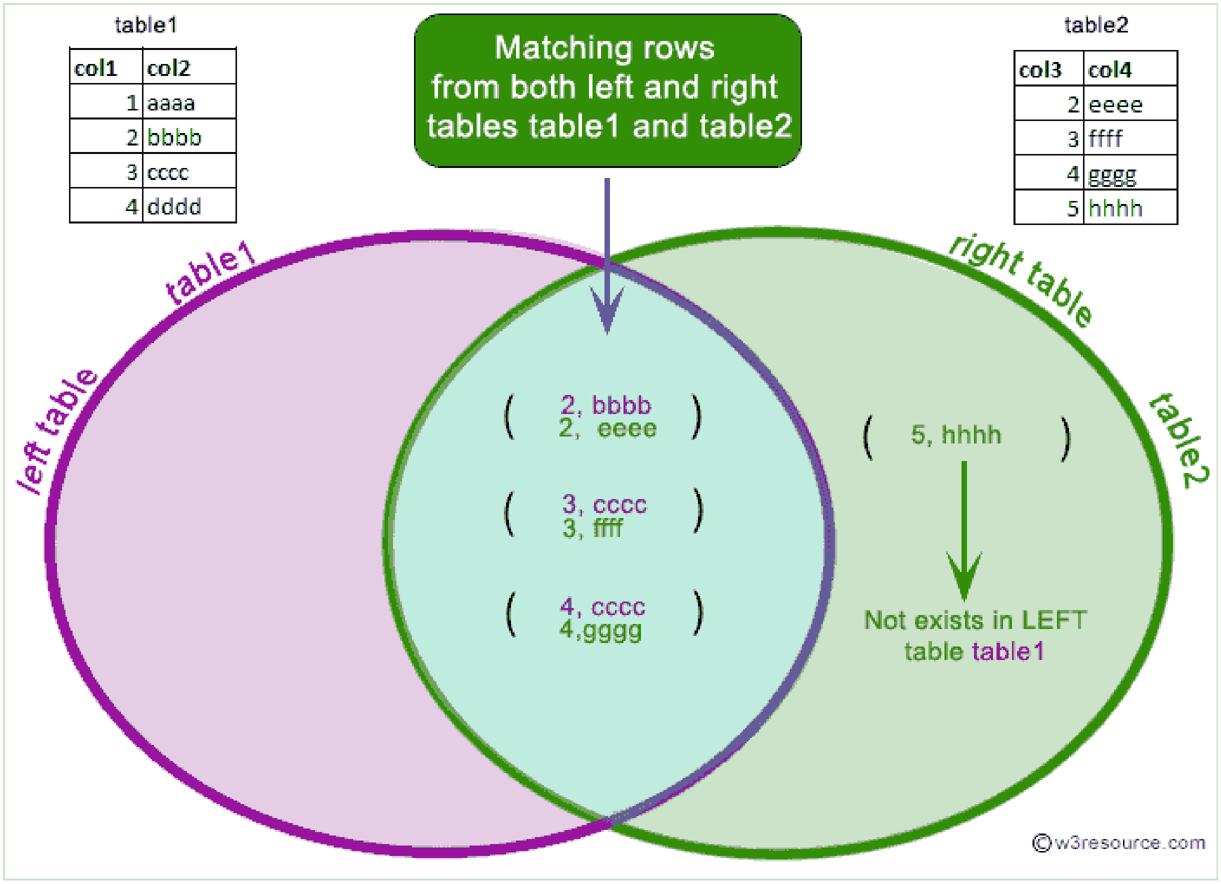


Syntax:

```
SELECT column_name(s)
FROM table1
```

```
RIGHT JOIN table2
ON table1.column_name = table2.column_name;
```

Note: In some databases RIGHT JOIN is called RIGHT OUTER JOIN.

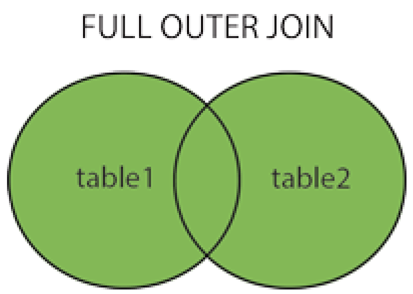


5: FULL JOIN

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

FULL OUTER JOIN and FULL JOIN are the same.

Note: FULL OUTER JOIN can potentially return very large result-sets!



```
SELECT column_name(s)
FROM table1
FULL JOIN/FULL OUTER JOIN table2
ON table1.column_name = table2.column_name;
```

6: CROSS JOIN

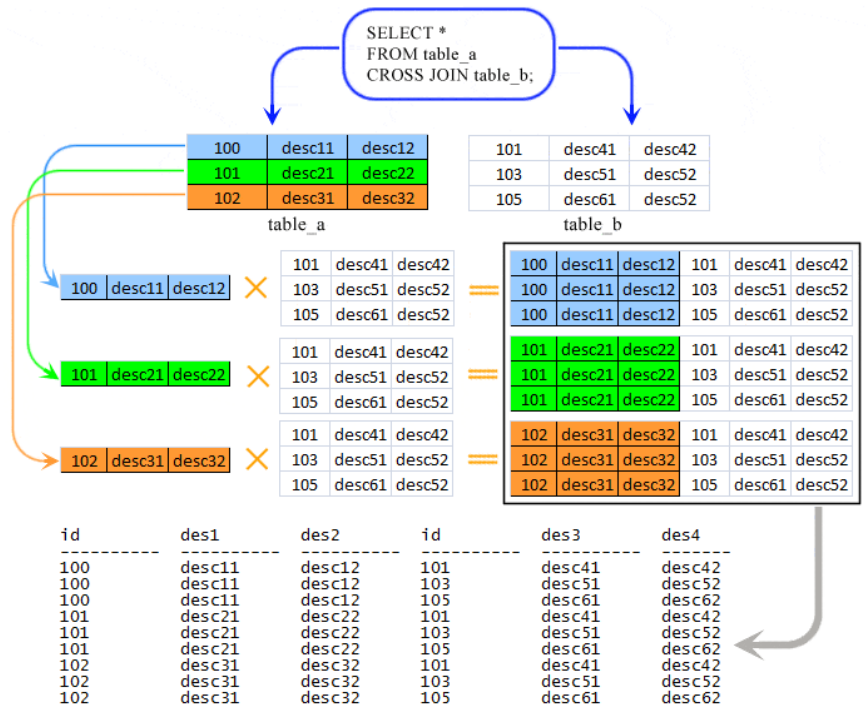
In MySQL, the CROSS JOIN produced a result set which is the product of rows of two associated tables when no WHERE clause is used with CROSS JOIN.

In this join, the result set appeared by multiplying each row of the first table with all rows in the second table if no condition introduced with CROSS JOIN.

This kind of result is called as Cartesian Product.

In MySQL, the CROSS JOIN behaves like JOIN and INNER JOIN without using any condition.

In standard SQL the difference between INNER JOIN and CROSS JOIN is ON clause can be used with INNER JOIN on the other hand ON clause can't be used with CROSS JOIN.



```
SELECT *
FROM table_a
CROSS JOIN table_b;
```

table_a

100	desc11	desc12
101	desc21	desc22
102	desc31	desc32

table_b

101	desc41	desc42
103	desc51	desc52
105	desc61	desc52

100	desc11	desc12	101	desc41	desc42
100	desc11	desc12	103	desc51	desc52
100	desc11	desc12	105	desc61	desc52
101	desc21	desc22	101	desc41	desc42
101	desc21	desc22	103	desc51	desc52
101	desc21	desc22	105	desc61	desc52
102	desc31	desc32	101	desc41	desc42
102	desc31	desc32	103	desc51	desc52
102	desc31	desc32	105	desc61	desc52

```
id          des1        des2        id          des3        des4
----------  ----------  ----------  ----------  ----------  --------
100         desc11      desc12      101         desc41      desc42
100         desc11      desc12      103         desc51      desc52
100         desc11      desc12      105         desc61      desc62
101         desc21      desc22      101         desc41      desc42
101         desc21      desc22      103         desc51      desc52
101         desc21      desc22      105         desc61      desc62
102         desc31      desc32      101         desc41      desc42
102         desc31      desc32      103         desc51      desc52
102         desc31      desc32      105         desc61      desc62
```

Syntax:

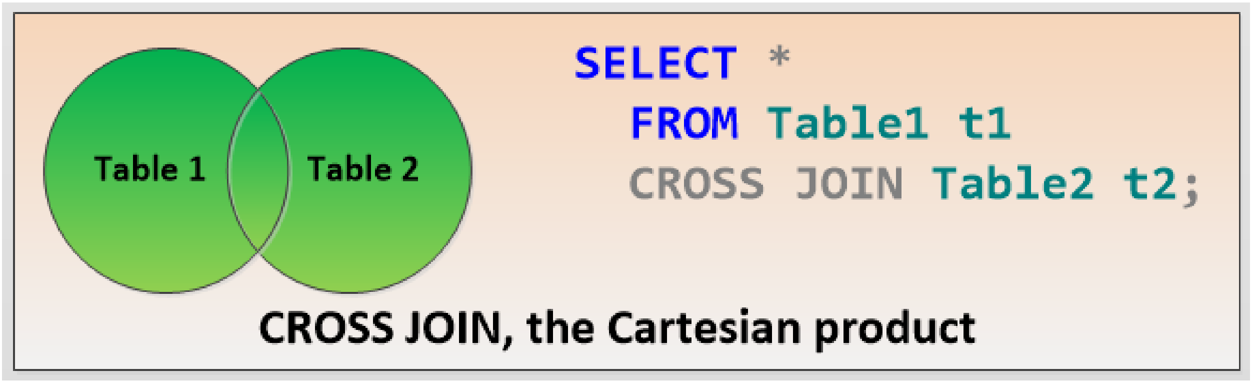
```
SELECT table1.col1, table2.col1, table1.col2, table2.col3, …
FROM table1
CROSS JOIN table2;
```

7: FULL JOIN vs CROSS JOIN

After looking at the Venn diagrams for the two, they are both shown as the same, however they are not the same by any means.

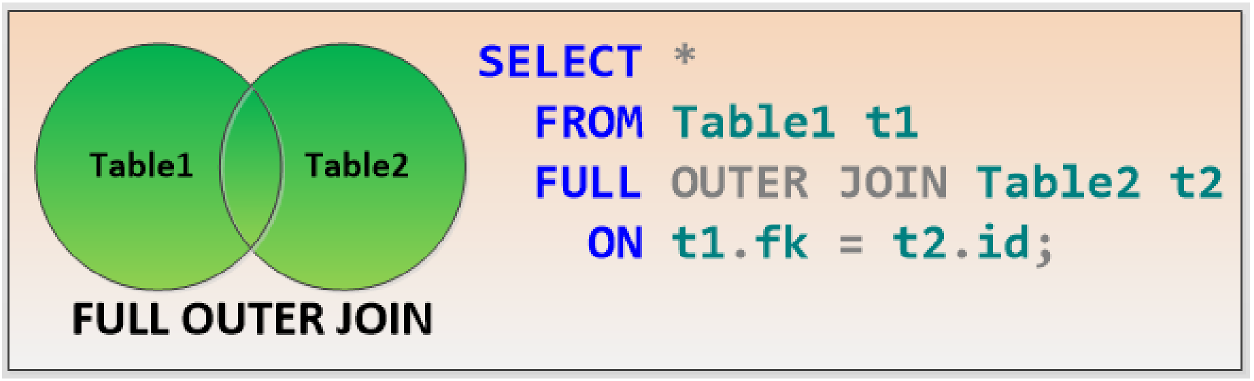
Yes, they both include all rows from both the LEFT and RIGHT side of the JOIN, however they are matched up or JOINed in a very different way.

Lets take a look, first at the Venn diagrams, you can see below that the Venn diagrams show that all rows from Table1 are included in the results, and all rows from Table2 are included in the results. This is indeed correct, the differences come in the number of rows and how the results are matched up.



```
SELECT *
  FROM Table1 t1
  CROSS JOIN Table2 t2;
```

The CROSS JOIN gives you the Cartesian product of the two tables, by matching every row from one table with every row from another table. If there are X rows in the first table, and Y rows in the second table, the result set will have exactly X times Y rows. Notice on the CROSS JOIN, there is no ON clause specified. This is because there is not matching.



```
SELECT *
  FROM Table1 t1
  FULL OUTER JOIN Table2 t2
    ON t1.fk = t2.id;
```

Now for the full outer JOIN, this is like asking for a LEFT OUTER JOIN and a RIGHT OUTER JOIN in the same query. You are asking to get every row from Table1 matching what could be matched in Table2, for those that don’t match on the Table1 side, just display the Table2 side, with NULLs on the LEFT side, and for those that don’t match on the TABLE2 side, just display the Table1 side with NULLs on the RIGHT side.

Lab Task(s):

Exercise

1. Write a query in SQL to display the first name, last name, department number, and department name for each employee. (**Sample tables:** employees & departments)
2. Write a query to find the name (first_name, last_name), job, department ID and name of the department who works in London. (**Sample tables:** employees , locations & departments)
3. Write a query in SQL to display the first and last name, department, city, and state province for each employee. (**Sample tables:** employees , locations & departments)
4. Write a query to find the employee id, name (last_name) along with their manager_id and name (last_name). (**Sample tables:** employees)
5. Write a query to find the name (first_name, last_name) and hire date of the employees who was hired after 'Jones'. (**Sample tables:** employees)
6. Write a query to get the department name and number of employees in the department. (**Sample tables:** employees & departments)
7. Write a query to display the department ID and name and first name of manager. (**Sample tables:** employees & departments)
8. Write a query to display the department name, manager name, and city. (**Sample tables:** employees , locations & departments)
9. Write a query to display the job history that were done by any employee who is currently drawing more than 10000 of salary. (**Sample tables:** employees & job_history)
10. Write a query to display the first name, last name, hire date, salary of the manager for all managers whose experience is more than 15 years. (**Sample tables:** employees & departments)
11. Write a query in SQL to display the name of the department, average salary and number of employees working in that department who got commission. (**Sample tables:** employees & departments)
12. Write a query in SQL to display the name of the country, city, and the departments which are running there. (**Sample tables:** countries , locations & departments)
13. Write a query in SQL to display department name and the full name (first and last name) of the manager. (**Sample tables:** employees & departments)
14. Write a query in SQL to display the details of jobs which was done by any of the employees who is presently earning a salary on and above 12000. (**Sample tables:** employees & job_history)
15. Write a query in SQL to display the full name (first and last name), and salary of those employees who working in any department located in London. (**Sample tables:** employees , locations & departments)

END