Data Engineering Batch 1 **Date: 25/01/2024**

Name: Pradip Bochare

SQL Coding Challenge

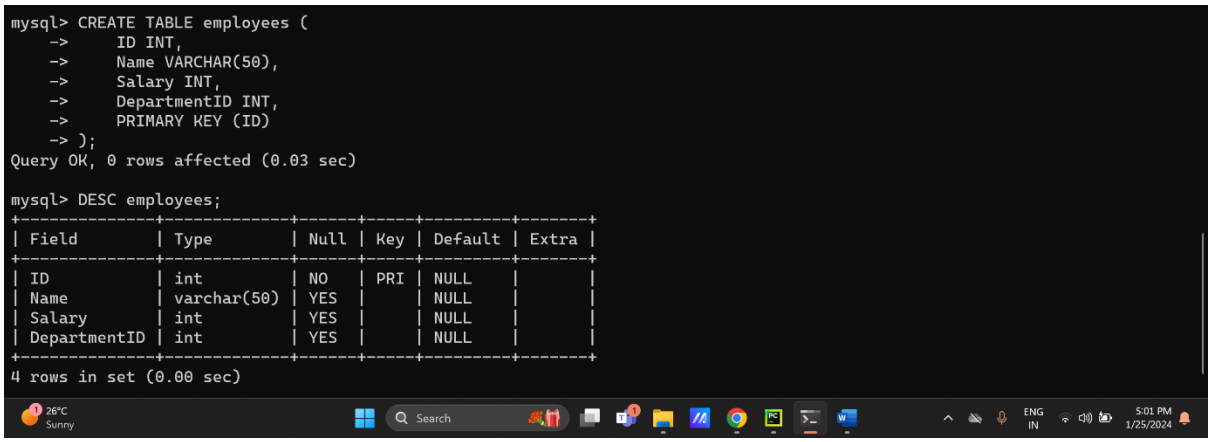
❖ SQL JOIN:

SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Following are different types of joins.

Inner join
Left join
Right join
Full join
Cross join
Equi join

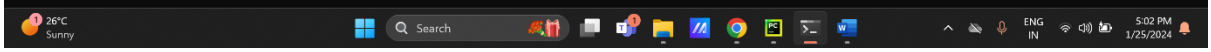
- Here we have created a database ‘sqlcodinchaleng’ to implement Joins.

- Created Table ‘employees’ and ‘departments’ in database



```
mysql> CREATE TABLE departments (
    ->     DepartmentID INT,
    ->     DepartmentName VARCHAR(50),
    ->     PRIMARY KEY (DepartmentID)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> DESC departments;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| DepartmentID   | int         | NO   | PRI | NULL    |       |
| DepartmentName | varchar(50) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```



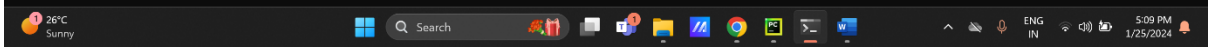
- Inserted Sample data in ‘employees’ and ‘departments’ table.

```
mysql> INSERT INTO employees (ID, Name, Salary, DepartmentID) VALUES
    -> (1, 'Pradip', 80000, 2),
    -> (2, 'Ram', 50000, 3),
    -> (3, 'Kaustubh', 55000, 1),
    -> (4, 'Sarvesh', 40000, 3),
    -> (5, 'Tejas' , 45000, 1),
    -> (6, 'Suraj', 70000, 2),
    -> (7, 'Yash', 60000, 1);
Query OK, 7 rows affected (0.01 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM employees;
+----+----------+--------+--------------+
| ID | Name     | Salary | DepartmentID |
+----+----------+--------+--------------+
|  1 | Pradip   |  80000 |            2 |
|  2 | Ram      |  50000 |            3 |
|  3 | Kaustubh |  55000 |            1 |
|  4 | Sarvesh  |  40000 |            3 |
|  5 | Tejas    |  45000 |            1 |
|  6 | Suraj    |  70000 |            2 |
|  7 | Yash     |  60000 |            1 |
+----+----------+--------+--------------+
7 rows in set (0.00 sec)
```

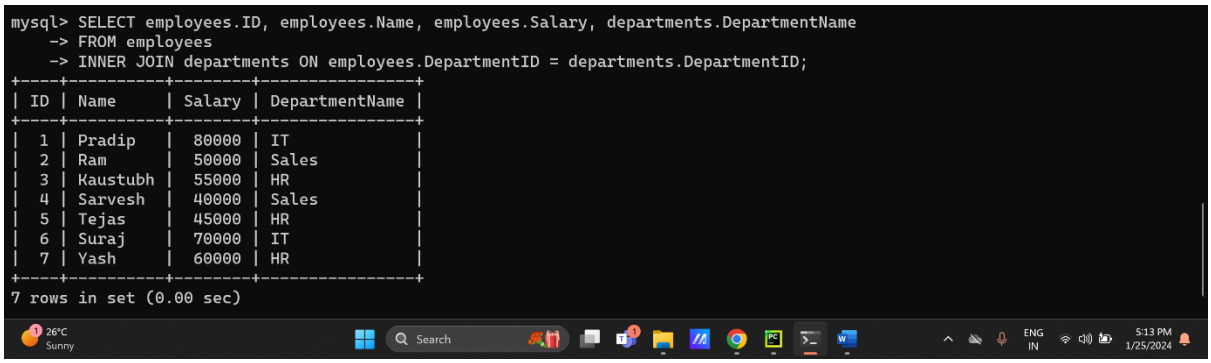
```
mysql> INSERT INTO departments (DepartmentID, DepartmentName) VALUES
    -> (1, 'HR'),
    -> (2, 'IT'),
    -> (3, 'Sales');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM departments;
+--------------+----------------+
| DepartmentID | DepartmentName |
+--------------+----------------+
|            1 | HR             |
|            2 | IT             |
|            3 | Sales          |
+--------------+----------------+
3 rows in set (0.00 sec)
```



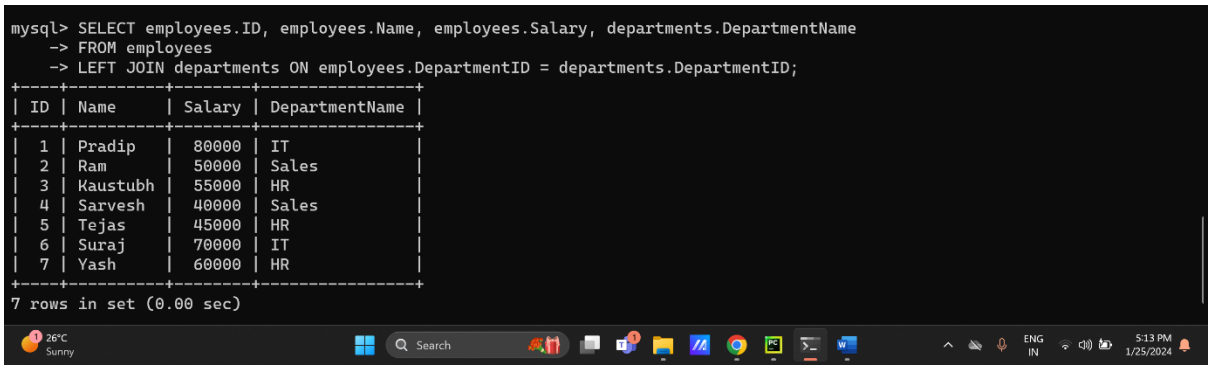
❖ Inner Join:

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



❖ 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.



❖ Right 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.

```
mysql> SELECT employees.ID, employees.Name, employees.Salary, departments.DepartmentName
    -> FROM employees
    -> RIGHT JOIN departments ON employees.DepartmentID = departments.DepartmentID;
+------+----------+--------+----------------+
| ID   | Name     | Salary | DepartmentName |
+------+----------+--------+----------------+
|    7 | Yash     |  60000 | HR             |
|    5 | Tejas    |  45000 | HR             |
|    3 | Kaustubh |  55000 | HR             |
|    6 | Suraj    |  70000 | IT             |
|    1 | Pradip   |  80000 | IT             |
|    4 | Sarvesh  |  40000 | Sales          |
|    2 | Ram      |  50000 | Sales          |
+------+----------+--------+----------------+
7 rows in set (0.00 sec)
```

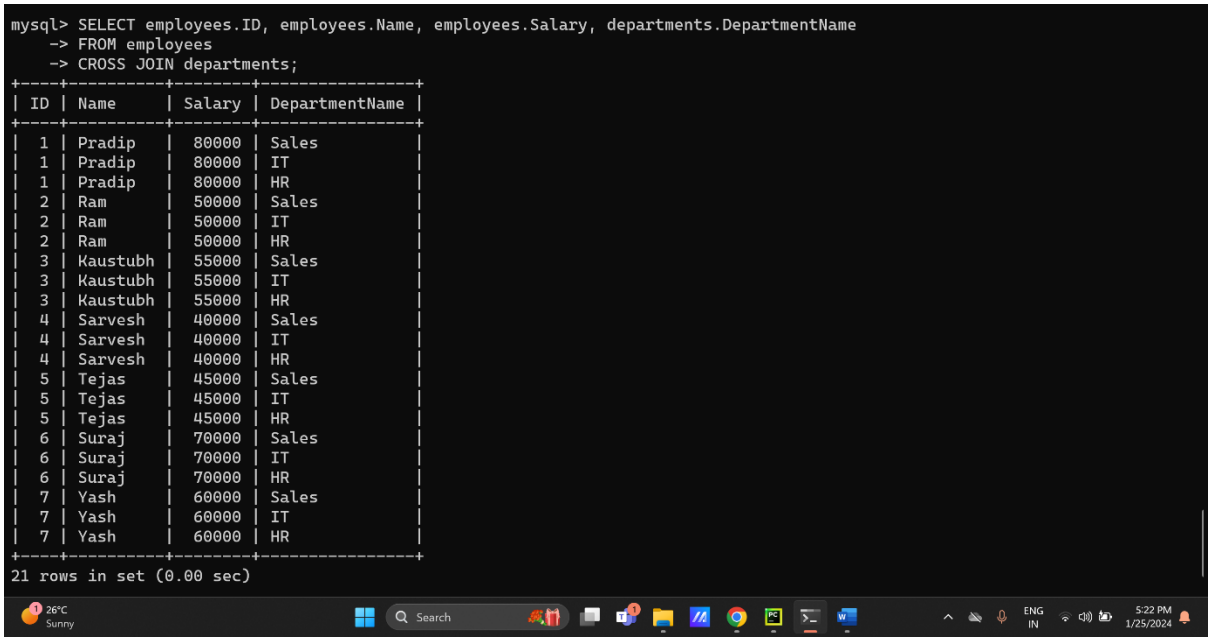
❖ **Full Join:**

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.

```
mysql> SELECT employees.ID, employees.Name, employees.Salary, departments.DepartmentName
    -> FROM employees
    -> LEFT JOIN departments ON employees.DepartmentID = departments.DepartmentID
    -> UNION
    -> SELECT employees.ID, employees.Name, employees.Salary, departments.DepartmentName
    -> FROM employees
    -> RIGHT JOIN departments ON employees.DepartmentID = departments.DepartmentID
    -> ;
+------+----------+--------+----------------+
| ID   | Name     | Salary | DepartmentName |
+------+----------+--------+----------------+
|    1 | Pradip   |  80000 | IT             |
|    2 | Ram      |  50000 | Sales          |
|    3 | Kaustubh |  55000 | HR             |
|    4 | Sarvesh  |  40000 | Sales          |
|    5 | Tejas    |  45000 | HR             |
|    6 | Suraj    |  70000 | IT             |
|    7 | Yash     |  60000 | HR             |
+------+----------+--------+----------------+
7 rows in set (0.00 sec)
```

❖ **Cross Join**

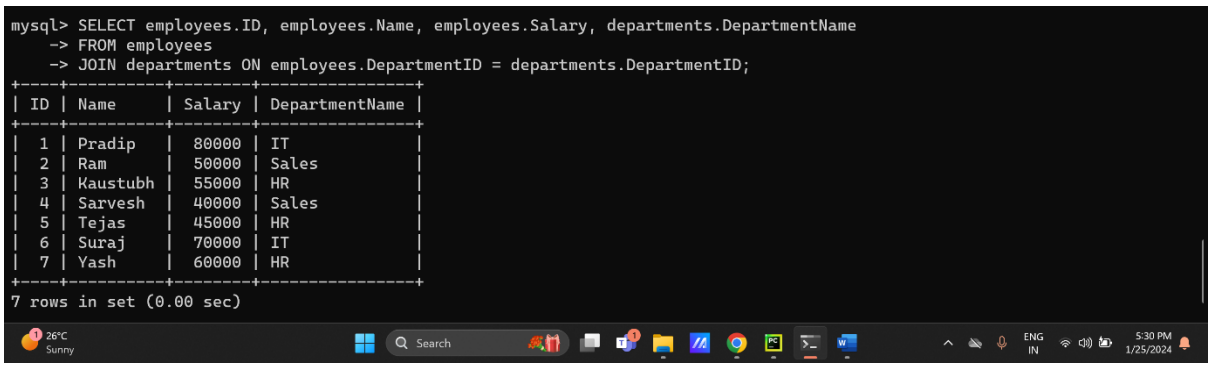
Cross Join returns the Cartesian product of two tables meaning it produces all the possible combinations of rows between the two tables.



```
mysql> SELECT employees.ID, employees.Name, employees.Salary, departments.DepartmentName
    -> FROM employees
    -> CROSS JOIN departments;
+----+----------+--------+----------------+
| ID | Name     | Salary | DepartmentName |
+----+----------+--------+----------------+
|  1 | Pradip   |  80000 | Sales          |
|  1 | Pradip   |  80000 | IT             |
|  1 | Pradip   |  80000 | HR             |
|  2 | Ram      |  50000 | Sales          |
|  2 | Ram      |  50000 | IT             |
|  2 | Ram      |  50000 | HR             |
|  3 | Kaustubh |  55000 | Sales          |
|  3 | Kaustubh |  55000 | IT             |
|  3 | Kaustubh |  55000 | HR             |
|  4 | Sarvesh  |  40000 | Sales          |
|  4 | Sarvesh  |  40000 | IT             |
|  4 | Sarvesh  |  40000 | HR             |
|  5 | Tejas    |  45000 | Sales          |
|  5 | Tejas    |  45000 | IT             |
|  5 | Tejas    |  45000 | HR             |
|  6 | Suraj    |  70000 | Sales          |
|  6 | Suraj    |  70000 | IT             |
|  6 | Suraj    |  70000 | HR             |
|  7 | Yash     |  60000 | Sales          |
|  7 | Yash     |  60000 | IT             |
|  7 | Yash     |  60000 | HR             |
+----+----------+--------+----------------+
21 rows in set (0.00 sec)
```

❖ Equi Join

An equi join is a type of join that combines rows from two or more tables based on a matching condition in a specified column. Equi join uses the equality operator (=) to match rows where specified columns have equal values.



```
mysql> SELECT employees.ID, employees.Name, employees.Salary, departments.DepartmentName
    -> FROM employees
    -> JOIN departments ON employees.DepartmentID = departments.DepartmentID;
+----+----------+--------+----------------+
| ID | Name     | Salary | DepartmentName |
+----+----------+--------+----------------+
|  1 | Pradip   |  80000 | IT             |
|  2 | Ram      |  50000 | Sales          |
|  3 | Kaustubh |  55000 | HR             |
|  4 | Sarvesh  |  40000 | Sales          |
|  5 | Tejas    |  45000 | HR             |
|  6 | Suraj    |  70000 | IT             |
|  7 | Yash     |  60000 | HR             |
+----+----------+--------+----------------+
7 rows in set (0.00 sec)
```