53 SQL Developer Interview Questions-Answers

Watch Full Video On Youtube:
https://youtu.be/pKFo8Mqp-cU

Connect with me:
Youtube: https://www.youtube.com/c/nitmantalks
Instagram: https://www.instagram.com/nitinmangotra/
LinkedIn: https://www.linkedin.com/in/nitin-mangotra-9a075a149/
Facebook: https://www.facebook.com/NitManTalks/
Twitter: https://twitter.com/nitinmangotra07/
Telegram: https://t.me/nitmantalks/

1. What Is The SQL Query Used For Creating A Database And A Table?

SQL Query To Create A Database:

```
CREATE DATABASE NitManDB;
```

If successful, the command will generate output similar to the following:

```
Query OK, 1 row affected (0.00 sec)
```

If the database name specified conflicts with an existing database MySQL will display and error message reporting this fact:

```
ERROR 1007 (HY000): Can't create database 'NitManDB'; database exists
```

In this situation, a different database name should be selected, or the IF NOT EXISTS option should be used. This option only creates the database if it does not already exist:

```
CREATE DATABASE IF NOT EXISTS NitManDB;
```

1. What Is The SQL Query Used For Creating A Database And A Table?

SQL Query To Create A Table:

- New Tables are added to an existing database using the **CREATE TABLE** statement.
- But Before a table can be created a database must first be selected so that MySQL knows where to create the table.
- This is achieved using the USE SQL statement:

```
USE NitManDB;
```

Syntax For Creating A Table:

```
CREATE TABLE "tablename"
        ("column1" "data type",
         "column2" "data type",
         ...
         "columnN" "data type");
```

Example:

```
CREATE TABLE DEPARTMENT
      (ID INT NOT NULL,
       NAME VARCHAR(250) NOT NULL,
       LOCATION VARCHAR(250)
);
```

```
mysql> CREATE TABLE DEPARTMENT
    ->        (ID int NOT NULL,
    -> NAME VARCHAR(250) NOT NULL,
    -> LOCATION VARCHAR(250))
    ->    ;
Query OK, 0 rows affected (0.01 sec)
```

2. What Is The SQL Query Used To Create A Table With Same Structure Of Another Table?

```
mysql> select * from department;
+----+----------------+-----------+
| ID | NAME           | LOCATION  |
+----+----------------+-----------+
|  1 | Development    | Pune      |
|  2 | Finance        | Gurgaon   |
|  3 | Sales          | Delhi     |
|  4 | Purchase       | Mumbai    |
|  5 | Marketing      | Bangalore |
|  6 | Human Resource | Noida     |
|  7 | Operations     | Hyderabad |
|  8 | Quality        | Chennai   |
+----+----------------+-----------+
8 rows in set (0.00 sec)

mysql> create table department_copy
    -> as (select * from department where 1=2);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from department_copy;
Empty set (0.00 sec)
```

SQL QUERY:

```
CREATE TABLE department_copy
as (SELECT * FROM department WHERE 1=2);
```

```
mysql> describe department_copy;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| ID       | int         | NO   |     | NULL    |       |
| NAME     | varchar(250)| NO   |     | NULL    |       |
| LOCATION | varchar(250)| YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> describe department;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| ID       | int         | NO   | PRI | NULL    |       |
| NAME     | varchar(250)| NO   |     | NULL    |       |
| LOCATION | varchar(250)| YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

3. What Is The SQL Query Used To Create A Table With Same Structure With Data Of Another Table

SQL QUERY:

CREATE TABLE Emp SELECT * FROM Employee;

```
mysql> create table emp
    -> select * from employee;
Query OK, 20 rows affected (0.02 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

4. What Is The SQL Query Used To Find The 2nd / 3rd / nth Highest Salary?

1. USING SUB-QUERY:

```
SELECT MAX(Salary) FROM Employee
WHERE Salary<(SELECT MAX(Salary) FROM Employee
WHERE Salary<(SELECT MAX(Salary) FROM Employee));
```

```
mysql> select max(salary) from employee
    -> where salary<(select max(salary) from employee
    -> where salary<(select max(salary) from employee));
+-------------+
| max(salary) |
+-------------+
|    51000.00 |
+-------------+
1 row in set (0.00 sec)
```

```
mysql> select name, salary
    -> from employee
    -> order by salary desc;
+----------+----------+
| name     | salary   |
+----------+----------+
| Mohit    | 70000.00 |
| Shreya   | 65000.00 |
| Satish   | 51000.00 |
| Sweety   | 45000.00 |
| Divya    | 35000.00 |
| Shivam   | 35000.00 |
| Saurabh  | 33000.00 |
| Raghav   | 31000.00 |
| Swati    | 30000.00 |
| Arjun    | 28000.00 |
| Utkarsh  | 27000.00 |
| Pranshul | 26000.00 |
| Srishti  | 26000.00 |
| Nitin    | 25000.00 |
| Kushal   | 25000.00 |
| Kiran    | 25000.00 |
| Divya    | 22000.00 |
| Sonali   | 20000.00 |
| Rohit    | 15000.00 |
| Manav    | 11000.00 |
+----------+----------+
20 rows in set (0.00 sec)
```

4. What Is The SQL Query Used To Find The 2nd / 3rd / nth Highest Salary?

2. USING LIMIT:

SELECT Salary FROM Employee
ORDER BY Salary DESC LIMIT N-1,1;

```
mysql> select salary from employee
    -> order by salary desc limit 2,1;
+----------+
| salary   |
+----------+
| 51000.00 |
+----------+
1 row in set (0.00 sec)

mysql> select salary from employee
    -> order by salary desc limit 1,1;
+----------+
| salary   |
+----------+
| 65000.00 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> select salary from employee
    -> order by salary desc limit 2,2;
+----------+
| salary   |
+----------+
| 51000.00 |
| 45000.00 |
+----------+
2 rows in set (0.00 sec)

mysql> select salary from employee
    -> order by salary desc limit 1,3;
+----------+
| salary   |
+----------+
| 65000.00 |
| 51000.00 |
| 45000.00 |
+----------+
3 rows in set (0.00 sec)

mysql> select salary from employee
    -> order by salary desc limit 0,3;
+----------+
| salary   |
+----------+
| 70000.00 |
| 65000.00 |
| 51000.00 |
+----------+
3 rows in set (0.00 sec)
```

```
mysql> select name, salary
    -> from employee
    -> order by salary desc;
+----------+----------+
| name     | salary   |
+----------+----------+
| Mohit    | 70000.00 |
| Shreya   | 65000.00 |
| Satish   | 51000.00 |
| Sweety   | 45000.00 |
| Divya    | 35000.00 |
| Shivam   | 35000.00 |
| Saurabh  | 33000.00 |
| Raghav   | 31000.00 |
| Swati    | 30000.00 |
| Arjun    | 28000.00 |
| Utkarsh  | 27000.00 |
| Pranshul | 26000.00 |
| Srishti  | 26000.00 |
| Nitin    | 25000.00 |
| Kushal   | 25000.00 |
| Kiran    | 25000.00 |
| Divya    | 22000.00 |
| Sonali   | 20000.00 |
| Rohit    | 15000.00 |
| Manav    | 11000.00 |
+----------+----------+
20 rows in set (0.00 sec)
```

4. What Is The SQL Query Used To Find The 2nd / 3rd / nth Highest Salary?

3. USING LIMIT-OFFSET:

SELECT Salary FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET N-1;

For Example:
// 3rd Highest Salary//

SELECT Salary FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET 2;

```
mysql> select salary from employee
    -> order by salary desc
    -> limit 1 offset 2;
+----------+
| salary   |
+----------+
| 51000.00 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> select salary from employee
    -> order by salary desc
    -> limit 2 offset 2;
+----------+
| salary   |
+----------+
| 51000.00 |
| 45000.00 |
+----------+
2 rows in set (0.00 sec)
```

```
mysql> select name, salary
    -> from employee
    -> order by salary desc;
+----------+----------+
| name     | salary   |
+----------+----------+
| Mohit    | 70000.00 |
| Shreya   | 65000.00 |
| Satish   | 51000.00 |
| Sweety   | 45000.00 |
| Divya    | 35000.00 |
| Shivam   | 35000.00 |
| Saurabh  | 33000.00 |
| Raghav   | 31000.00 |
| Swati    | 30000.00 |
| Arjun    | 28000.00 |
| Utkarsh  | 27000.00 |
| Pranshul | 26000.00 |
| Srishti  | 26000.00 |
| Nitin    | 25000.00 |
| Kushal   | 25000.00 |
| Kiran    | 25000.00 |
| Divya    | 22000.00 |
| Sonali   | 20000.00 |
| Rohit    | 15000.00 |
| Manav    | 11000.00 |
+----------+----------+
20 rows in set (0.00 sec)
```

4. What Is The SQL Query Used To Find The 2nd / 3rd / nth Highest Salary?

4. USING DISTINCT:

```
SELECT Salary FROM Employee e1
WHERE (N-1) = (
   SELECT COUNT(DISTINCT(e2.Salary)) FROM Employee e2
   WHERE e2.Salary > e1.Salary);
```

```
mysql> SELECT salary
    -> FROM Employee e1
    -> WHERE (2) = (
    ->     SELECT COUNT(DISTINCT(e2.Salary))
    ->     FROM Employee e2
    ->     WHERE e2.Salary > e1.Salary);
+----------+
| salary   |
+----------+
| 51000.00 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> select name, salary
    -> from employee
    -> order by salary desc;
+----------+----------+
| name     | salary   |
+----------+----------+
| Mohit    | 70000.00 |
| Shreya   | 65000.00 |
| Satish   | 51000.00 |
| Sweety   | 45000.00 |
| Divya    | 35000.00 |
| Shivam   | 35000.00 |
| Saurabh  | 33000.00 |
| Raghav   | 31000.00 |
| Swati    | 30000.00 |
| Arjun    | 28000.00 |
| Utkarsh  | 27000.00 |
| Pranshul | 26000.00 |
| Srishti  | 26000.00 |
| Nitin    | 25000.00 |
| Kushal   | 25000.00 |
| Kiran    | 25000.00 |
| Divya    | 22000.00 |
| Sonali   | 20000.00 |
| Rohit    | 15000.00 |
| Manav    | 11000.00 |
+----------+----------+
20 rows in set (0.00 sec)
```

5. What Is The SQL Query Used To Find All Employees Who Also Hold The Managerial Position?

SQL QUERY:

```
SELECT * FROM EMPLOYEES
WHERE (EMPLOYEE_ID IN (SELECT MANAGER_ID FROM EMPLOYEES));
```

```
mysql> select * from employee
    -> where id IN(select manager_id from employee);
+----+--------+----------+---------+------------+------------+------------+----------+---------+--------+
| ID | EMPNO  | NAME     | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE   | SALARY   | BONUS   | COMM   |
+----+--------+----------+---------+------------+------------+------------+----------+---------+--------+
|  1 | 100001 | Nitin    |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 | 500.00 |
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |  50.00 |
|  3 | 100003 | Utkarsh  |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |   5.00 |
|  6 | 100006 | Rohit    |       1 |          8 | 9876543215 | 2020-02-27 | 15000.00 | 1000.00 | 600.00 |
|  7 | 100007 | Shreya   |       4 |         16 | 9876543216 | 2020-08-27 | 65000.00 |  600.00 |   NULL |
|  8 | 100008 | Satish   |       2 |         16 | 9876543216 | 2020-02-03 | 51000.00 | 2000.00 |   NULL |
| 12 | 200012 | Sweety   |       2 |          2 | 9876543219 | 2022-09-24 | 45000.00 | 2000.00 | 800.00 |
| 13 | 200013 | Mohit    |       6 |          3 | 9876543210 | 2022-04-11 | 70000.00 | 4000.00 | 400.00 |
| 16 | 200016 | Kushal   |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |  20.00 |
+----+--------+----------+---------+------------+------------+------------+----------+---------+--------+
9 rows in set (0.01 sec)
```

```
mysql> SELECT id, name, manager_id,salary
    -> from employee;
+----+----------+------------+----------+
| id | name     | manager_id | salary   |
+----+----------+------------+----------+
|  1 | Nitin    |          3 | 25000.00 |
|  2 | Pranshul |          8 | 26000.00 |
|  3 | Utkarsh  |          8 | 27000.00 |
|  4 | Arjun    |          8 | 28000.00 |
|  5 | Divya    |          8 | 35000.00 |
|  6 | Rohit    |          8 | 15000.00 |
|  7 | Shreya   |         16 | 65000.00 |
|  8 | Satish   |         16 | 51000.00 |
|  9 | Shivam   |          1 | 35000.00 |
| 10 | Sonali   |          1 | 20000.00 |
| 11 | Swati    |          1 | 30000.00 |
| 12 | Sweety   |          2 | 45000.00 |
| 13 | Mohit    |          3 | 70000.00 |
| 14 | Manav    |          3 | 11000.00 |
| 15 | Divya    |          3 | 22000.00 |
| 16 | Kushal   |          3 | 25000.00 |
| 17 | Kiran    |          6 | 25000.00 |
| 18 | Srishti  |         12 | 26000.00 |
| 19 | Raghav   |          7 | 31000.00 |
| 20 | Saurabh  |         13 | 33000.00 |
+----+----------+------------+----------+
20 rows in set (0.00 sec)
```

6. What Is The SQL Query Used To Find The Names Of The Employees That Begin With ‘A’?

SQL QUERY:

SELECT Name FROM Employee WHERE Name LIKE 'A%';

To Find Names Starting From Multiple Characters:

SELECT Name FROM Employee
WHERE Name LIKE 'A%’ OR Name LIKE 'N%’;

```
mysql> select name from employee where name LIKE 'A%' OR name LIKE 'N%';
+-------+
| name  |
+-------+
| Nitin |
| Arjun |
+-------+
2 rows in set (0.00 sec)
```

```
mysql> select name from employee where name LIKE 'A%';
+-------+
| name  |
+-------+
| Arjun |
+-------+
1 row in set (0.00 sec)

mysql> select name from employee where name LIKE 'N%';
+-------+
| name  |
+-------+
| Nitin |
+-------+
1 row in set (0.00 sec)

mysql> select name from employee where name LIKE 'S%';
+---------+
| name    |
+---------+
| Shreya  |
| Satish  |
| Shivam  |
| Sonali  |
| Swati   |
| Sweety  |
| Srishti |
| Saurabh |
+---------+
8 rows in set (0.00 sec)
```

7. What Is The SQL Query Used To Display The Current Date?

These Functions Return the current date.
The date is returned as "YYYY-MM-DD" (string) or as YYYYMMDD (numeric).

```
SELECT CURRENT_DATE;
SELECT CURRENT_DATE();
SELECT CURDATE();
SELECT DATE(NOW());
SELECT DATE(CURRENT_TIMESTAMP());
```

NOTE: The **NOW**() & **CURRENT_TIMESTAMP**() functions returns the current date and time in "YYYY-MM-DD HH-MM-SS" (string) or as YYYYMMDDHHMMSS (numeric) format. So we need to fetch date using date()

```
mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2022-02-26 02:29:21 |
+---------------------+
1 row in set (0.00 sec)

mysql> SELECT CURRENT_TIMESTAMP();
+---------------------+
| CURRENT_TIMESTAMP() |
+---------------------+
| 2022-02-26 02:29:31 |
+---------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT CURRENT_DATE, CURRENT_DATE(), CURDATE(), DATE(NOW()), DATE(CURRENT_TIMESTAMP());
+--------------+----------------+------------+-------------+--------------------------+
| CURRENT_DATE | CURRENT_DATE() | CURDATE()  | DATE(NOW()) | DATE(CURRENT_TIMESTAMP()) |
+--------------+----------------+------------+-------------+--------------------------+
| 2022-02-26   | 2022-02-26     | 2022-02-26 | 2022-02-26  | 2022-02-26                |
+--------------+----------------+------------+-------------+--------------------------+
1 row in set (0.00 sec)
```

8(i). What Is The SQL Query Used To Fetch Alternate Records From A Table?

To Fetch The Even Number Records:-

```
SELECT * FROM Employee WHERE id%2=0;
```

To Fetch The Odd Number Records:-

```
SELECT * FROM Employee WHERE id%2=1;
```

```
mysql> SELECT * FROM Employee WHERE id%2=0;
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+
| ID | EMPNO  | NAME     | DEPT_ID | manager_id | PHONENO    | HIREDATE   | SALARY  | BONUS   | COMM    |
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |  50.00 |
|  4 | 100004 | Arjun    |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |  50.00 |
|  6 | 100006 | Rohit    |       1 |          8 | 9876543215 | 2020-02-27 | 15000.00 | 1000.00 | 600.00 |
|  8 | 100008 | Satish   |       2 |         16 | 9876543216 | 2020-02-03 | 51000.00 | 2000.00 |    NULL |
| 10 | 100010 | Sonali   |       6 |          1 | 9876543218 | 2022-02-15 | 20000.00 | 1000.00 |    NULL |
| 12 | 200012 | Sweety   |       2 |          2 | 9876543219 | 2022-09-24 | 45000.00 | 2000.00 | 800.00 |
| 14 | 200014 | Manav    |       3 |          3 | 9876543211 | 2022-11-11 | 11000.00 | 1000.00 | 600.00 |
| 16 | 200016 | Kushal   |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |  20.00 |
| 18 | 200018 | Srishti  |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |  80.00 |
| 20 | 200020 | Saurabh  |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 | 950.00 |
| 22 | 200022 | Satish   |       2 |         16 | 9876543216 | 2020-02-03 | 44500.00 | 2000.00 |    NULL |
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+
11 rows in set (0.00 sec)

mysql> SELECT * FROM Employee WHERE id%2=1;
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+
| ID | EMPNO  | NAME     | DEPT_ID | manager_id | PHONENO    | HIREDATE   | SALARY  | BONUS   | COMM    |
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+
|  1 | 100001 | Nitin    |       1 |          3 | 9876543210 | 2022-02-17 | 25500.00 | 1000.00 |  500.00 |
|  3 | 100003 | Utkarsh  |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |    5.00 |
|  5 | 100005 | Divya    |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL |  430.00 |
|  7 | 100007 | Shreya   |       4 |         16 | 9876543216 | 2020-08-27 | 25500.00 |  600.00 |    NULL |
|  9 | 100009 | Shivam   |       5 |          1 | 9876543217 | 2021-02-05 | 35000.00 |    NULL |    NULL |
| 11 | 200011 | Swati    |       3 |          1 | 9876543218 | 2022-02-23 | 30000.00 | 8000.00 | 1000.00 |
| 13 | 200013 | Mohit    |       6 |          3 | 9876543210 | 2022-04-11 | 70000.00 | 4000.00 |  400.00 |
| 15 | 200015 | Divya    |       7 |          3 | 9876543220 | 2022-10-21 | 22000.00 | 1000.00 |   10.00 |
| 17 | 200017 | Kiran    |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |    NULL |
| 19 | 200019 | Raghav   |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 |  505.00 |
| 21 | 200021 | Divya    |       4 |          8 | 9876543214 | 2019-02-14 | 34500.00 |    NULL |  430.00 |
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+
11 rows in set (0.00 sec)
```

8(i). What Is The SQL Query Used To Fetch Alternate Records From A Table?

To Fetch The Even Number Records:-

```
SELECT * FROM (SELECT *,
    Row_Number() OVER(ORDER BY id)
    AS RowNumber FROM Employee) e
        WHERE e.RowNumber % 2 = 0;
```

To Fetch The Odd Number Records:-

```
SELECT * FROM (SELECT *,
    Row_Number() OVER(ORDER BY id)
    AS RowNumber FROM Employee) e
        WHERE e.RowNumber % 2 = 1;
```

```
mysql> SELECT * FROM (SELECT *, Row_Number() OVER(ORDER BY id) AS RowNumber FROM Employee) e WHERE e.RowNumber % 2 = 0;
+----+--------+----------+---------+------------+------------+------------+---------+---------+-----------+
| ID | EMPNO  | NAME     | DEPT_ID | manager_id | PHONENO    | HIREDATE   | SALARY  | BONUS   | COMM    | RowNumber |
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+-----------+
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |  50.00 |         2 |
|  4 | 100004 | Arjun    |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |  50.00 |         4 |
|  6 | 100006 | Rohit    |       1 |          8 | 9876543215 | 2020-02-27 | 15000.00 | 1000.00 | 600.00 |         6 |
|  8 | 100008 | Satish   |       2 |         16 | 9876543216 | 2020-02-03 | 51000.00 | 2000.00 |    NULL |         8 |
| 10 | 100010 | Sonali   |       6 |          1 | 9876543218 | 2022-02-15 | 20000.00 | 1000.00 |    NULL |        10 |
| 12 | 200012 | Sweety   |       2 |          2 | 9876543219 | 2022-09-24 | 45000.00 | 2000.00 | 800.00 |        12 |
| 14 | 200014 | Manav    |       3 |          3 | 9876543211 | 2022-11-11 | 11000.00 | 1000.00 | 600.00 |        14 |
| 16 | 200016 | Kushal   |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |  20.00 |        16 |
| 18 | 200018 | Srishti  |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |  80.00 |        18 |
| 20 | 200020 | Saurabh  |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 | 950.00 |        20 |
| 22 | 200022 | Satish   |       2 |         16 | 9876543216 | 2020-02-03 | 44500.00 | 2000.00 |    NULL |        22 |
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+-----------+
11 rows in set (0.00 sec)

mysql> SELECT * FROM (SELECT *, Row_Number() OVER(ORDER BY id) AS RowNumber FROM Employee) e WHERE e.RowNumber % 2 = 1;
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+-----------+
| ID | EMPNO  | NAME     | DEPT_ID | manager_id | PHONENO    | HIREDATE   | SALARY  | BONUS   | COMM    | RowNumber |
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+-----------+
|  1 | 100001 | Nitin    |       1 |          3 | 9876543210 | 2022-02-17 | 25500.00 | 1000.00 |  500.00 |         1 |
|  3 | 100003 | Utkarsh  |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |   5.00 |         3 |
|  5 | 100005 | Divya    |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL |  430.00 |         5 |
|  7 | 100007 | Shreya   |       4 |         16 | 9876543216 | 2020-08-27 | 25500.00 |  600.00 |    NULL |         7 |
|  9 | 100009 | Shivam   |       5 |          1 | 9876543217 | 2021-02-05 | 35000.00 |    NULL |    NULL |         9 |
| 11 | 200011 | Swati    |       3 |          1 | 9876543218 | 2022-02-23 | 30000.00 | 8000.00 | 1000.00 |        11 |
| 13 | 200013 | Mohit    |       6 |          3 | 9876543210 | 2022-04-11 | 70000.00 | 4000.00 |  400.00 |        13 |
| 15 | 200015 | Divya    |       7 |          3 | 9876543220 | 2022-10-21 | 22000.00 | 1000.00 |  10.00 |        15 |
| 17 | 200017 | Kiran    |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |    NULL |        17 |
| 19 | 200019 | Raghav   |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 |  505.00 |        19 |
| 21 | 200021 | Divya    |       4 |          8 | 9876543214 | 2019-02-14 | 34500.00 |    NULL |  430.00 |        21 |
+----+--------+----------+---------+------------+------------+------------+---------+---------+---------+-----------+
11 rows in set (0.01 sec)
```

8(ii). What Is The SQL Query Used To Fetch The Common Records From Two Tables?

SQL Query:-

```
SELECT * FROM Employee
    INNER JOIN Emp ON Employee.ID = Emp.ID;
```

```
mysql> SELECT * FROM EMPLOYEE INNER JOIN EMP ON EMPLOYEE.ID = EMP.ID;
+----+--------+--------+---------+------------+------------+----------+---------+---------+----+--------+--------+---------+------------+------------+----------+---------+---------+
| ID | EMPNO  | NAME   | DEPT_ID | manager_id | PHONENO    | HIREDATE | SALARY  | BONUS   | COMM | ID | empno  | NAME   | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE | SALARY  | BONUS   | COMM    |
+----+--------+--------+---------+------------+------------+----------+---------+---------+------+----+--------+--------+---------+------------+------------+----------+---------+---------+---------+
|  1 | 100001 | Nitin  |       1 |          3 | 9876543210 | 2022-02-17 | 25500.00 | 1000.00 |  500.00 |  1 | 100001 | Nitin  |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 |  500.00 |
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |   50.00 |  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |   50.00 |
|  3 | 100003 | Utkarsh |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |    5.00 |  3 | 100003 | Utkarsh |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |    5.00 |
|  4 | 100004 | Arjun  |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |   50.00 |  4 | 100004 | Arjun  |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |   50.00 |
|  5 | 100005 | Divya  |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL |  430.00 |  5 | 100005 | Divya  |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL |  430.00 |
|  6 | 100006 | Rohit  |       1 |          8 | 9876543215 | 2020-02-27 | 15000.00 | 1000.00 |  600.00 |  6 | 100006 | Rohit  |       1 |          8 | 9876543215 | 2020-02-27 | 15000.00 | 1000.00 |  600.00 |
|  7 | 100007 | Shreya |       4 |         16 | 9876543216 | 2020-08-27 | 25500.00 |  600.00 |    NULL |  7 | 100007 | Shreya |       4 |         16 | 9876543216 | 2020-08-27 | 65000.00 |  600.00 |    NULL |
|  8 | 100008 | Satish |       2 |         16 | 9876543216 | 2020-02-03 | 51000.00 | 2000.00 |    NULL |  8 | 100008 | Satish |       2 |         16 | 9876543216 | 2020-02-03 | 45000.00 | 2000.00 |    NULL |
|  9 | 100009 | Shivam |       5 |          1 | 9876543217 | 2021-02-05 | 35000.00 |    NULL |    NULL |  9 | 100009 | Shivam |       5 |          1 | 9876543217 | 2021-02-05 | 35000.00 |    NULL |    NULL |
| 10 | 100010 | Sonali |       6 |          1 | 9876543218 | 2022-02-15 | 20000.00 | 1000.00 |    NULL | 10 | 100010 | Sonali |       6 |          1 | 9876543218 | 2022-02-15 | 20000.00 | 1000.00 |    NULL |
| 11 | 200011 | Swati  |       3 |          1 | 9876543218 | 2022-02-23 | 30000.00 | 8000.00 | 1000.00 | 11 | 200011 | Swati  |       3 |          1 | 9876543218 | 2022-02-23 | 30000.00 | 8000.00 | 1000.00 |
| 12 | 200012 | Sweety |       2 |          2 | 9876543219 | 2022-09-24 | 45000.00 | 2000.00 |  800.00 | 12 | 200012 | Sweety |       2 |          2 | 9876543219 | 2022-09-24 | 45000.00 | 2000.00 |  800.00 |
| 13 | 200013 | Mohit  |       6 |          3 | 9876543210 | 2022-04-11 | 70000.00 | 4000.00 |  400.00 | 13 | 200013 | Mohit  |       6 |          3 | 9876543210 | 2022-04-11 | 70000.00 | 4000.00 |  400.00 |
| 14 | 200014 | Manav  |       3 |          3 | 9876543211 | 2022-11-11 | 11000.00 | 1000.00 |  600.00 | 14 | 200014 | Manav  |       3 |          3 | 9876543211 | 2022-11-11 | 11000.00 | 1000.00 |  600.00 |
| 15 | 200015 | Divya  |       7 |          3 | 9876543220 | 2022-10-21 | 22000.00 | 1000.00 |   10.00 | 15 | 200015 | Divya  |       7 |          3 | 9876543220 | 2022-10-21 | 22000.00 | 1000.00 |   10.00 |
| 16 | 200016 | Kushal |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |   20.00 | 16 | 200016 | Kushal |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |   20.00 |
| 17 | 200017 | Kiran  |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |    NULL | 17 | 200017 | Kiran  |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |    NULL |
| 18 | 200018 | Srishti |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |   80.00 | 18 | 200018 | Srishti |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |   80.00 |
| 19 | 200019 | Raghav |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 |  505.00 | 19 | 200019 | Raghav |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 |  505.00 |
| 20 | 200020 | Saurabh |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 |  950.00 | 20 | 200020 | Saurabh |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 |  950.00 |
+----+--------+--------+---------+------------+------------+----------+---------+---------+------+----+--------+--------+---------+------------+------------+----------+---------+---------+---------+
20 rows in set (0.00 sec)
```

9(i). What Is The SQL Query Used To Find Duplicate Rows In Table?

SQL Query:-

```
SELECT column, COUNT(column)
    FROM table_name
    GROUP BY column
    HAVING COUNT(column) > 1;

SELECT name, COUNT(*)
    FROM EMPLOYEE
    GROUP BY name
    HAVING COUNT(name) > 1;

SELECT salary, COUNT(*)
    FROM EMPLOYEE
    GROUP BY salary
    HAVING COUNT(*) > 1;
```

```
mysql> SELECT name, COUNT(*)
    -> FROM EMPLOYEE
    -> GROUP BY name
    -> HAVING COUNT(name) > 1;
+--------+----------+
| name   | COUNT(*) |
+--------+----------+
| Divya  |        3 |
| Satish |        2 |
+--------+----------+
2 rows in set (0.00 sec)

mysql> SELECT salary, COUNT(*)
    -> FROM EMPLOYEE
    -> GROUP BY salary
    -> HAVING COUNT(*) > 1;
+----------+----------+
| salary   | COUNT(*) |
+----------+----------+
| 25500.00 |        2 |
| 26000.00 |        2 |
| 35000.00 |        2 |
| 25000.00 |        2 |
+----------+----------+
4 rows in set (0.00 sec)
```

9(ii). What Is The SQL Query Used To Remove The Duplicate Rows In Table?

SQL Query:-

```
DELETE E1 FROM EMPLOYEE E1
    INNER JOIN EMPLOYEE E2
    WHERE
      E1.id < E2.id AND
      E1.Name = E2.Name;
```

```
mysql> select * from employee;
+----+--------+---------+---------+------------+------------+------------+---------+---------+---------+
| ID | EMPNO  | NAME    | DEPT_ID | manager_id | PHONENO    | HIREDATE   | SALARY  | BONUS   | COMM    |
+----+--------+---------+---------+------------+------------+------------+---------+---------+---------+
|  1 | 100001 | Nitin   |       1 |          3 | 9876543210 | 2022-02-17 | 25500.00 | 1000.00 |  500.00 |
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |   50.00 |
|  3 | 100003 | Utkarsh |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |    5.00 |
|  4 | 100004 | Arjun   |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |   50.00 |
|  5 | 100005 | Divya   |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL |  430.00 |
|  6 | 100006 | Rohit   |       1 |          8 | 9876543215 | 2020-02-27 | 15000.00 | 1000.00 |  600.00 |
|  7 | 100007 | Shreya  |       4 |         16 | 9876543216 | 2020-08-27 | 25500.00 |  600.00 |    NULL |
|  8 | 100008 | Satish  |       2 |         16 | 9876543216 | 2020-02-03 | 51000.00 | 2000.00 |    NULL |
|  9 | 100009 | Shivam  |       5 |          1 | 9876543217 | 2021-02-05 | 35000.00 |    NULL |    NULL |
| 10 | 100010 | Sonali  |       6 |          1 | 9876543218 | 2022-02-15 | 20000.00 | 1000.00 |    NULL |
| 11 | 200011 | Swati   |       3 |          1 | 9876543218 | 2022-02-23 | 30000.00 | 8000.00 | 1000.00 |
| 12 | 200012 | Sweety  |       2 |          2 | 9876543219 | 2022-09-24 | 45000.00 | 2000.00 |  800.00 |
| 13 | 200013 | Mohit   |       6 |          3 | 9876543210 | 2022-04-11 | 70000.00 | 4000.00 |  400.00 |
| 14 | 200014 | Manav   |       3 |          3 | 9876543211 | 2022-11-11 | 11000.00 | 1000.00 |  600.00 |
| 15 | 200015 | Divya   |       7 |          3 | 9876543220 | 2022-10-21 | 22000.00 | 1000.00 |   10.00 |
| 16 | 200016 | Kushal  |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |   20.00 |
| 17 | 200017 | Kiran   |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |    NULL |
| 18 | 200018 | Srishti |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |   80.00 |
| 19 | 200019 | Raghav  |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 |  505.00 |
| 20 | 200020 | Saurabh |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 |  950.00 |
| 21 | 200021 | Divya   |       4 |          8 | 9876543214 | 2019-02-14 | 34500.00 |    NULL |  430.00 |
| 22 | 200022 | Satish  |       2 |         16 | 9876543216 | 2020-02-03 | 44500.00 | 2000.00 |    NULL |
+----+--------+---------+---------+------------+------------+------------+---------+---------+---------+
22 rows in set (0.00 sec)
```

```
mysql> SELECT name, COUNT(*)
    -> FROM EMPLOYEE
    -> GROUP BY name
    -> HAVING COUNT(name) > 1;
+--------+----------+
| name   | COUNT(*) |
+--------+----------+
| Divya  |        3 |
| Satish |        2 |
+--------+----------+
2 rows in set (0.00 sec)

mysql> DELETE E1 FROM EMPLOYEE E1
    -> INNER JOIN EMPLOYEE E2
    -> WHERE
    ->      E1.id < E2.id AND
    ->      E1.Name = E2.Name;
Query OK, 3 rows affected (0.01 sec)

mysql> SELECT name, COUNT(*)
    -> FROM EMPLOYEE
    -> GROUP BY name
    -> HAVING COUNT(name) > 1;
Empty set (0.00 sec)

mysql> select count(*) from employee;
+----------+
| count(*) |
+----------+
|       19 |
+----------+
1 row in set (0.00 sec)
```

10. What Is The SQL Query Used To Find The nth Record From A Table?

SQL Query:-

Syntax : SELECT * FROM <table_name> LIMIT N-1,1;
Example : SELECT * FROM Employee LIMIT 0,1;

Syntax : SELECT * FROM <table_name> LIMIT 1 OFFSET N-1;
Example : SELECT * FROM Employee LIMIT 1 OFFSET 10;

```
mysql> SELECT * FROM Employee LIMIT 0,1;
+----+--------+-------+---------+------------+------------+----------+---------+--------+
| ID | EMPNO  | NAME  | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE | SALARY  | BONUS  | COMM   |
+----+--------+-------+---------+------------+------------+----------+---------+--------+
|  1 | 100001 | Nitin |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 | 500.00 |
+----+--------+-------+---------+------------+------------+----------+---------+--------+
1 row in set (0.00 sec)

mysql> SELECT * FROM Employee LIMIT 3,1;
+----+--------+-------+---------+------------+------------+----------+---------+--------+--------+
| ID | EMPNO  | NAME  | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE | SALARY  | BONUS  | COMM   |
+----+--------+-------+---------+------------+------------+----------+---------+--------+--------+
|  4 | 100004 | Arjun |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |  NULL  | 50.00  |
+----+--------+-------+---------+------------+------------+----------+---------+--------+--------+
1 row in set (0.01 sec)

mysql> SELECT * FROM Employee  LIMIT 1 OFFSET 10;
+----+--------+-------+---------+------------+------------+----------+---------+--------+---------+
| ID | EMPNO  | NAME  | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE | SALARY  | BONUS  | COMM    |
+----+--------+-------+---------+------------+------------+----------+---------+--------+---------+
| 11 | 200011 | Swati |       3 |          1 | 9876543218 | 2022-02-23 | 30000.00 | 8000.00 | 1000.00 |
+----+--------+-------+---------+------------+------------+----------+---------+--------+---------+
1 row in set (0.00 sec)

mysql> SELECT * FROM Employee  LIMIT 1 OFFSET 3;
+----+--------+-------+---------+------------+------------+----------+---------+--------+--------+
| ID | EMPNO  | NAME  | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE | SALARY  | BONUS  | COMM   |
+----+--------+-------+---------+------------+------------+----------+---------+--------+--------+
|  4 | 100004 | Arjun |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |  NULL  | 50.00  |
+----+--------+-------+---------+------------+------------+----------+---------+--------+--------+
1 row in set (0.00 sec)
```

11(i). What Is The SQL Query Used To Find The First 5 Records From A Table?

SQL Query:-

```
SELECT * FROM Employee LIMIT 5;
SELECT * FROM Employee ORDER BY ID LIMIT 5;
```

```
mysql> SELECT * FROM Employee LIMIT 5;
+----+--------+----------+---------+------------+------------+------------+----------+---------+--------+
| ID | EMPNO  | NAME     | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE   | SALARY   | BONUS   | COMM   |
+----+--------+----------+---------+------------+------------+------------+----------+---------+--------+
|  1 | 100001 | Nitin    |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 | 500.00 |
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |  50.00 |
|  3 | 100003 | Utkarsh  |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |   5.00 |
|  4 | 100004 | Arjun    |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |  50.00 |
|  5 | 100005 | Divya    |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL | 430.00 |
+----+--------+----------+---------+------------+------------+------------+----------+---------+--------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Employee ORDER BY ID LIMIT 5;
+----+--------+----------+---------+------------+------------+------------+----------+---------+--------+
| ID | EMPNO  | NAME     | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE   | SALARY   | BONUS   | COMM   |
+----+--------+----------+---------+------------+------------+------------+----------+---------+--------+
|  1 | 100001 | Nitin    |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 | 500.00 |
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |  50.00 |
|  3 | 100003 | Utkarsh  |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |   5.00 |
|  4 | 100004 | Arjun    |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |  50.00 |
|  5 | 100005 | Divya    |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL | 430.00 |
+----+--------+----------+---------+------------+------------+------------+----------+---------+--------+
5 rows in set (0.00 sec)
```

11(ii). What Is The SQL Query Used To Find The Last 5 Records From A Table?

SQL Query:-

```
(SELECT * FROM Employee
ORDER BY ID DESC LIMIT 5)
ORDER BY ID ASC;
```

```
SELECT * FROM Employee
WHERE ID > (SELECT MAX(ID) - 5
FROM Employee);
```

```
SELECT * FROM Employee
WHERE ID > (SELECT COUNT(*)
FROM Employee) - 5;
```

```
mysql> (SELECT * FROM Employee ORDER BY ID DESC LIMIT 5) ORDER BY ID ASC;
+----+--------+---------+---------+------------+------------+------------+---------+--------+--------+
| ID | EMPNO  | NAME    | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE   | SALARY  | BONUS  | COMM   |
+----+--------+---------+---------+------------+------------+------------+---------+--------+--------+
| 16 | 200016 | Kushal  |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |  20.00 |
| 17 | 200017 | Kiran   |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |   NULL |
| 18 | 200018 | Srishti |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |  80.00 |
| 19 | 200019 | Raghav  |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 | 505.00 |
| 20 | 200020 | Saurabh |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 | 950.00 |
+----+--------+---------+---------+------------+------------+------------+---------+--------+--------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Employee WHERE ID > (SELECT MAX(ID) - 5 FROM Employee);
+----+--------+---------+---------+------------+------------+------------+---------+--------+--------+
| ID | EMPNO  | NAME    | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE   | SALARY  | BONUS  | COMM   |
+----+--------+---------+---------+------------+------------+------------+---------+--------+--------+
| 16 | 200016 | Kushal  |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |  20.00 |
| 17 | 200017 | Kiran   |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |   NULL |
| 18 | 200018 | Srishti |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |  80.00 |
| 19 | 200019 | Raghav  |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 | 505.00 |
| 20 | 200020 | Saurabh |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 | 950.00 |
+----+--------+---------+---------+------------+------------+------------+---------+--------+--------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM Employee WHERE ID > (SELECT COUNT(*) FROM Employee) - 5;
+----+--------+---------+---------+------------+------------+------------+---------+--------+--------+
| ID | EMPNO  | NAME    | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE   | SALARY  | BONUS  | COMM   |
+----+--------+---------+---------+------------+------------+------------+---------+--------+--------+
| 16 | 200016 | Kushal  |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |  20.00 |
| 17 | 200017 | Kiran   |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |   NULL |
| 18 | 200018 | Srishti |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |  80.00 |
| 19 | 200019 | Raghav  |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 | 505.00 |
| 20 | 200020 | Saurabh |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 | 950.00 |
+----+--------+---------+---------+------------+------------+------------+---------+--------+--------+
5 rows in set (0.00 sec)
```

12. What Is The SQL Query Used To Find The First Or Last Record From A Table?

To Find First Record:-

```
SELECT *  FROM employee  LIMIT 1;

SELECT * FROM Employee
      WHERE ID=(SELECT min(ID) FROM Employee);
```

To Find Last Record:-

```
SELECT *  FROM Employee ORDER BY ID desc LIMIT 1;

SELECT * FROM Employee
      WHERE ID=(SELECT max(ID) FROM employee);
```

```
mysql> SELECT *  FROM employee  LIMIT 1;
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
| ID | EMPNO  | NAME   | DEPT_ID | MANAGER_ID | PHONENO   | HIREDATE      | SALARY  | BONUS   | COMM    |
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
|  1 | 100001 | Nitin  |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 | 500.00 |
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
1 row in set (0.00 sec)

mysql> SELECT * FROM Employee WHERE ID=(SELECT min(ID) FROM Employee);
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
| ID | EMPNO  | NAME   | DEPT_ID | MANAGER_ID | PHONENO   | HIREDATE      | SALARY  | BONUS   | COMM    |
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
|  1 | 100001 | Nitin  |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 | 500.00 |
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
1 row in set (0.00 sec)
```

```
mysql> SELECT *  FROM Employee ORDER BY ID desc LIMIT 1;
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
| ID | EMPNO  | NAME   | DEPT_ID | MANAGER_ID | PHONENO   | HIREDATE      | SALARY  | BONUS   | COMM    |
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
| 20 | 200020 | Saurabh |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 | 900.00 | 950.00 |
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
1 row in set (0.00 sec)

mysql> SELECT * FROM Employee WHERE ID=(SELECT max(ID) FROM employee);
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
| ID | EMPNO  | NAME   | DEPT_ID | MANAGER_ID | PHONENO   | HIREDATE      | SALARY  | BONUS   | COMM    |
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
| 20 | 200020 | Saurabh |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 | 900.00 | 950.00 |
+----+--------+--------+---------+------------+-----------+---------------+---------+---------+---------+
1 row in set (0.00 sec)
```

13. What Is The SQL Query Used To Find The Distinct Records Without Using Distinct Keyword?

USING DISTINCT KEYWORD:

SELECT DISTINCT dept_id FROM Employee;

```
mysql> SELECT DISTINCT dept_id FROM Employee;
+---------+
| dept_id |
+---------+
|       1 |
|       2 |
|       3 |
|       4 |
|       5 |
|       6 |
|       7 |
|       8 |
+---------+
8 rows in set (0.01 sec)
```

13. What Is The SQL Query Used To Find The Distinct Records Without Using Distinct Keyword?

1. USING GROUP BY:

```
SELECT dept_id FROM Employee GROUP BY dept_id;

SELECT salary FROM  Employee GROUP BY salary;
```

```
mysql> SELECT dept_id FROM Employee GROUP BY dept_id;
+---------+
| dept_id |
+---------+
|       1 |
|       2 |
|       3 |
|       4 |
|       5 |
|       6 |
|       7 |
|       8 |
+---------+
8 rows in set (0.00 sec)

mysql> SELECT salary FROM  Employee GROUP BY salary;
+----------+
| salary   |
+----------+
| 25000.00 |
| 26000.00 |
| 27000.00 |
| 28000.00 |
| 35000.00 |
| 15000.00 |
| 65000.00 |
| 51000.00 |
| 20000.00 |
| 30000.00 |
| 45000.00 |
| 70000.00 |
| 11000.00 |
| 22000.00 |
| 31000.00 |
| 33000.00 |
+----------+
16 rows in set (0.00 sec)
```

13. What Is The SQL Query Used To Find The Distinct Records Without Using Distinct Keyword?

2. USING SET UNION OPERATOR:

```
SELECT dept_id FROM Employee
UNION
SELECT dept_id FROM Employee;


SELECT salary FROM Employee
UNION
SELECT salary FROM Employee;
```

```
mysql> SELECT dept_id FROM Employee UNION SELECT dept_id FROM Employee;
+---------+
| dept_id |
+---------+
|       1 |
|       2 |
|       3 |
|       4 |
|       5 |
|       6 |
|       7 |
|       8 |
+---------+
8 rows in set (0.01 sec)

mysql> SELECT salary FROM Employee UNION SELECT salary FROM Employee;
+----------+
| salary   |
+----------+
| 25000.00 |
| 26000.00 |
| 27000.00 |
| 28000.00 |
| 35000.00 |
| 15000.00 |
| 65000.00 |
| 51000.00 |
| 20000.00 |
| 30000.00 |
| 45000.00 |
| 70000.00 |
| 11000.00 |
| 22000.00 |
| 31000.00 |
| 33000.00 |
+----------+
16 rows in set (0.00 sec)
```

13. What Is The SQL Query Used To Find The Distinct Records Without Using Distinct Keyword?

3. USING SUB-QUERY:

```
SELECT dept_id FROM Employee A
    WHERE A.id >=
        ALL(SELECT B.id FROM Employee B
                WHERE A.dept_id = B.dept_id)
                ORDER BY dept_id ;
```

```
mysql> SELECT dept_id FROM Employee A
    -> WHERE A.id >=
    -> ALL(SELECT B.id FROM Employee B
    -> WHERE A.dept_id = B.dept_id)
    -> ORDER BY dept_id ;
+---------+
| dept_id |
+---------+
|       1 |
|       2 |
|       3 |
|       4 |
|       5 |
|       6 |
|       7 |
|       8 |
+---------+
8 rows in set (0.01 sec)
```

14. What Is The SQL Query Used To Find The Maximum Salary Of Each Department?

SQL QUERY:

```
SELECT dept_id, MAX(salary) FROM Employee
    GROUP BY dept_id;

SELECT dept_id, MAX(salary) FROM Employee E
    RIGHT OUTER JOIN Department D
    ON (E.dept_id = D.id)
    GROUP BY dept_id;

SELECT dept_id, MAX(salary) FROM department D
    LEFT OUTER JOIN employee E
    ON (D.id = E.dept_id)
    GROUP BY dept_id;
```

```
mysql> SELECT dept_id, MAX(salary) FROM Employee
    -> GROUP BY dept_id;
+---------+-------------+
| dept_id | MAX(salary) |
+---------+-------------+
|       1 |    25500.00 |
|       2 |    51000.00 |
|       3 |    33000.00 |
|       4 |    35000.00 |
|       5 |    35000.00 |
|       6 |    70000.00 |
|       7 |    25000.00 |
|       8 |    31000.00 |
+---------+-------------+
8 rows in set (0.00 sec)
```

```
mysql> SELECT dept_id, MAX(salary) FROM Employee E
    -> RIGHT OUTER JOIN Department D
    -> ON (E.dept_id = D.id)
    -> GROUP BY dept_id;
+---------+-------------+
| dept_id | MAX(salary) |
+---------+-------------+
|       1 |    25500.00 |
|       2 |    51000.00 |
|       3 |    33000.00 |
|       4 |    35000.00 |
|       5 |    35000.00 |
|       6 |    70000.00 |
|       7 |    25000.00 |
|       8 |    31000.00 |
+---------+-------------+
8 rows in set (0.00 sec)
```

```
mysql> SELECT dept_id, MAX(salary) FROM department D
    -> LEFT OUTER JOIN employee E
    -> ON (D.id = E.dept_id)
    -> GROUP BY dept_id;
+---------+-------------+
| dept_id | MAX(salary) |
+---------+-------------+
|       1 |    25500.00 |
|       2 |    51000.00 |
|       3 |    33000.00 |
|       4 |    35000.00 |
|       5 |    35000.00 |
|       6 |    70000.00 |
|       7 |    25000.00 |
|       8 |    31000.00 |
+---------+-------------+
8 rows in set (0.00 sec)
```

15. What Is The SQL Query Used To Find The Department-Wise Count Of Employees Sorted By Department's Count In Ascending Order?

SQL QUERY:

```
SELECT dept_id, COUNT(dept_id) FROM Employee
    GROUP BY dept_id
    ORDER BY COUNT(dept_id);


SELECT  D.name, COUNT(*) as Count FROM Employee E
    INNER JOIN Department D on E.dept_id = D.id
    GROUP BY D.name
    ORDER BY Count;
```

```
mysql> SELECT dept_id, COUNT(dept_id) FROM Employee
    -> GROUP BY dept_id
    -> ORDER BY COUNT(dept_id);
+---------+----------------+
| dept_id | COUNT(dept_id) |
+---------+----------------+
|       5 |              2 |
|       6 |              2 |
|       7 |              2 |
|       8 |              2 |
|       1 |              3 |
|       3 |              3 |
|       2 |              4 |
|       4 |              4 |
+---------+----------------+
8 rows in set (0.00 sec)

mysql> SELECT  D.name, COUNT(*) as Count
    -> FROM Employee E
    -> INNER JOIN Department D on E.dept_id = D.id
    -> GROUP BY D.name
    -> ORDER BY Count;
+----------------+-------+
| name           | Count |
+----------------+-------+
| Marketing      |     2 |
| Human Resource |     2 |
| Operations     |     2 |
| Quality        |     2 |
| Development    |     3 |
| Sales          |     3 |
| Finance        |     4 |
| Purchase       |     4 |
+----------------+-------+
8 rows in set (0.00 sec)
```

16. How Will You Change The Datatype Of A Column?

By Using ALTER TABLE & MODIFY:

Syntax:-

ALTER TABLE [table_name] MODIFY
[column_name] [new_data_type];

The syntax shows that we are using **ALTER TABLE** statement.
Change datatype of a column using **MODIFY**

For Example:

ALTER TABLE Employee MODIFY manager_id BIGINT;

```
mysql> describe employee;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| ID         | int          | NO   |     | NULL    |       |
| EMPNO      | char(6)      | NO   | PRI | NULL    |       |
| NAME       | varchar(250) | NO   |     | NULL    |       |
| DEPT_ID    | int          | YES  | MUL | NULL    |       |
| MANAGER_ID | int          | YES  |     | NULL    |       |
| PHONENO    | char(10)     | YES  |     | NULL    |       |
| HIREDATE   | date         | YES  |     | NULL    |       |
| SALARY     | decimal(9,2) | YES  |     | NULL    |       |
| BONUS      | decimal(9,2) | YES  |     | NULL    |       |
| COMM       | decimal(9,2) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
10 rows in set (0.01 sec)

mysql> ALTER TABLE employee MODIFY manager_id BIGINT;
Query OK, 22 rows affected (0.06 sec)
Records: 22  Duplicates: 0  Warnings: 0

mysql> describe employee;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| ID         | int          | NO   |     | NULL    |       |
| EMPNO      | char(6)      | NO   | PRI | NULL    |       |
| NAME       | varchar(250) | NO   |     | NULL    |       |
| DEPT_ID    | int          | YES  | MUL | NULL    |       |
| manager_id | bigint       | YES  |     | NULL    |       |
| PHONENO    | char(10)     | YES  |     | NULL    |       |
| HIREDATE   | date         | YES  |     | NULL    |       |
| SALARY     | decimal(9,2) | YES  |     | NULL    |       |
| BONUS      | decimal(9,2) | YES  |     | NULL    |       |
| COMM       | decimal(9,2) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

17. What Are Tables And Fields In The Database?

Table:

- A **Table** is an organized collection of data stored in the form of rows and columns.
- It enables users to store and display records in the structure format.
- It is similar to worksheets in the spreadsheet application.

Employee Table ⇒>

ID	Name	Department	Salary
1	Nitin	Testing	25000
2	Pranshul	Marketing	28000
3	Utkarsh	Sales	24000

17. What Are Tables And Fields In The Database?

Field:

- ❑ Fields are the components to provide the structure for the table.
- ❑ It stores the same category of data in the same data type.
- ❑ A table contains a fixed number of columns but can have any number of rows known as the record.

- ★ The columns in a table are called FIELDS while the rows can be referred to as RECORDS.
- ★ The columns can be categorized as vertical and rows as horizontal.

ID	Name	Department	Salary
1	Nitin	Testing	25000
2	Pranshul	Marketing	28000
3	Utkarsh	Sales	24000

17. What Are Tables And Fields In The Database?

Table:

- A **Table** is an organized collection of data stored in the form of rows and columns.
- It enables users to store and display records in the structure format.
- It is similar to worksheets in the spreadsheet application.

ID	Name	Department	Salary
1	Nitin	Testing	25000
2	Pranshul	Marketing	28000
3	Utkarsh	Sales	24000

Field:

- Fields are the components to provide the structure for the table.
- It stores the same category of data in the same data type.
- A table contains a fixed number of columns but can have any number of rows known as the record.

- ★ The columns in a table are called FIELDS while the rows can be referred to as RECORDS.
- ★ The columns can be categorized as vertical and rows as horizontal.

Example:

Table: Employee.
Field: ID, Name, Department, Salary
Record: 1, Nitin, Development, 25000

18. What Is The Difference Between Unique Key, Primary key And Foreign key?

UNIQUE KEY:

- The role of the Unique key is to make sure that each column and row are unique.
- The Unique key cannot accept the duplicate values.
- It can accept a null value but only one null value per column.
- It ensures the integrity of the column or group of columns to store different values into a table

ID	Name	Department	Salary
1	Nitin	Testing	25000
2	Pranshul	Marketing	28000
3	Utkarsh	Sales	24000

Syntax: UNIQUE constraint on **single column**

```
CREATE TABLE Employee(
    ID int NOT NULL,
    Name varchar(255) NOT NULL,
    Department varchar(255),
    Salary int,
    UNIQUE  (ID));
```

Syntax: UNIQUE constraint on **multiple columns**

```
CREATE TABLE Employee (
    ID int NOT NULL,
    Name varchar(255) NOT NULL,
    Department varchar(255),
    Salary int,
    CONSTRAINT UC_Employee UNIQUE (ID, Name));
```

18. What Is The Difference Between Primary Key, Foreign key And Unique key?

PRIMARY KEY:

- A Primary key is used to uniquely identify all table records.
- You can consider Primary Key constraint to be a combination of UNIQUE and NOT NULL constraint.

  i.e Primary Key = UNIQUE + NOT NULL
- This means that if a column is set as a primary key, then this particular column cannot have any null values present in it and also all the values present in this column must be unique.
- A table can have only one primary key that consists of single or multiple fields.

ID	Name	Department	Salary
1	Nitin	Testing	25000
2	Pranshul	Marketing	28000
3	Utkarsh	Sales	24000

Syntax: PRIMARY constraint on **single column**

```
CREATE TABLE Employee(
    ID int NOT NULL,
    Name varchar(255) NOT NULL,
    Department varchar(255),
    Salary int,
    PRIMARY KEY (ID));
```

Syntax: PRIMARY constraint on **multiple columns**

```
CREATE TABLE Employee (
    ID int NOT NULL,
    Name varchar(255) NOT NULL,
    Department varchar(255),
    Salary int,
    CONSTRAINT PK_Employee PRIMARY KEY (ID, Name));
```

18. What Is The Difference Between Primary Key, Foreign key And Unique key?

There cannot be more than one Primary key but unique keys can be multiple.

S.NO	PRIMARY KEY	UNIQUE KEY
1.	The primary key act as a unique identifier for each record in the table.	The unique key is also a unique identifier for records when the primary key is not present in the table.
2.	We cannot store NULL values in the primary key column	We can store NULL value in the unique key column, but only one NULL is allowed.
3.	We cannot change or delete the primary key column values.	We can modify the unique key column values.

18. What Is The Difference Between Primary Key, Foreign key And Unique key?

FOREIGN KEY:

- The foreign key is used to link one or more tables together. It is also known as the **Referencing Key**.
- A Foreign key is specified as a key that is related to the primary key of another table. It means a foreign key field in one table refers to the primary key field of the other table.
- A Foreign key is a field that can uniquely identify each row in another table.
- The primary key-foreign key relationship is a very crucial relationship as it maintains the ACID properties of the database sometimes.

Syntax:

```
CREATE TABLE Employee (
    ID int NOT NULL,
    Name varchar(255) NOT NULL,
    Dept_ID int,
    Salary int,
    PRIMARY KEY (ID),
    FOREIGN KEY (Dept_ID) REFERENCES Department(ID));
```

18. What Is The Difference Between Primary Key, Foreign key And Unique key?

FOREIGN KEY:

Consider the two tables as shown:
As we can clearly see, that the field Dept_Id in the Employee table is the primary key in the Department table, i.e. it uniquely identifies each row in the Department table. Therefore, it is a Foreign Key in the Employee table.

Employee Table

ID	Name	Dept_Id	Salary
1	Nitin	1	25000
2	Pranshul	2	28000
3	Utkarsh	1	24000
4	Rohit	3	27000

Department Table

ID	Name	Location
1	Development	Gurgaon
2	Marketing	Bangalore
3	Sales	Pune

19. What Is 'SELECT' Statement? What Are Some Common Clauses Used With 'SELECT' Query In SQL?

'SELECT' Statement:

- SELECT operator is used to select data from a database.
- The data returned is stored in a result table, called the result-set.
- SELECT is a data manipulation language (DML) command.
- **Syntax**:

```
SELECT * FROM EMPLOYEE;
```

Example:

```
SELECT dept_id, COUNT(id)
    FROM employee
    WHERE dept_id!= "2"
      GROUP BY dept_id
      HAVING COUNT(id) > 2
          ORDER BY dept_id ASC;
```

Some Common Clauses Used With SELECT Query in SQL:

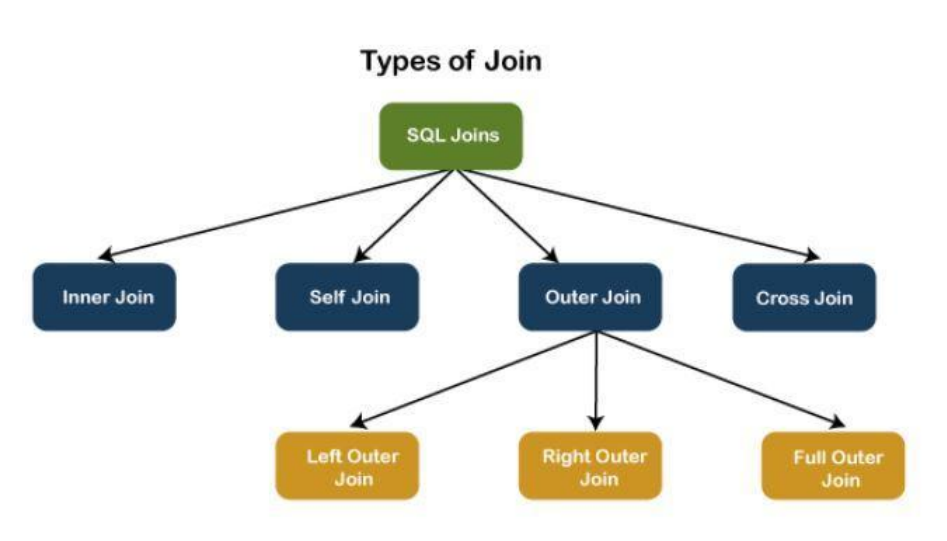
FROM Clause: It defines the tables and views from which data can be interpreted.
WHERE Clause: It is used to filter records that are required depending on certain criteria.
ORDER BY Clause: It is used to sort data in ascending (ASC) or descending (DESC) order depending on specific field(s).
GROUP BY Clause: It is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.
HAVING Clause: It is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records.

```
mysql> SELECT dept_id, COUNT(id) FROM employee
    -> WHERE dept_id!= "2"
    -> GROUP BY dept_id
    -> HAVING COUNT(id) > 2
    -> ORDER BY dept_id ASC;
+---------+-----------+
| dept_id | COUNT(id) |
+---------+-----------+
|       1 |         3 |
|       3 |         3 |
|       4 |         3 |
+---------+-----------+
3 rows in set (0.00 sec)
```

20. What Is A Join? List Different Types Of Joins.

JOINS:

- JOINS are used to retrieve data from multiple tables into a meaningful result set.
- It is performed whenever you need to fetch records from two or more tables.



Source of the image: https://www.javatpoint.com

20. What Is A Join? List Different Types Of Joins.

There are 4 types of JOINS, Which are:-:

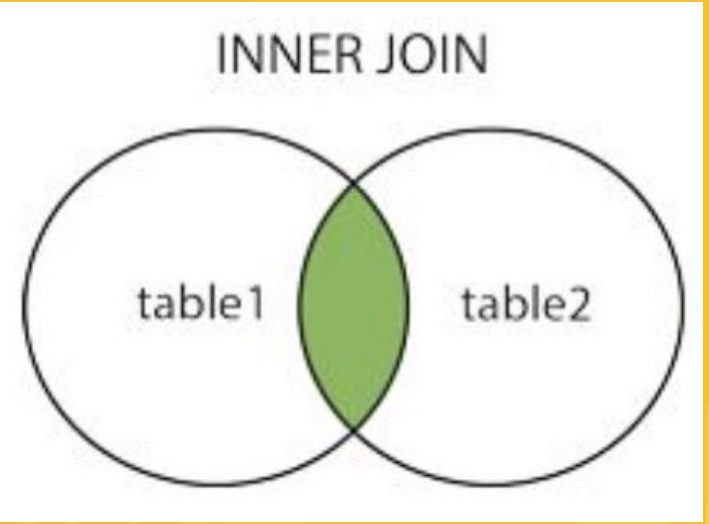
- INNER JOIN
- CROSS JOIN
- SELF JOIN
- OUTER JOIN (LEFT, RIGHT, FULL)

INNER JOIN:

- Inner Join basically returns records that have matching values in both tables.

Syntax:

```
SELECT * FROM Table_A INNER JOIN Table_B;
```



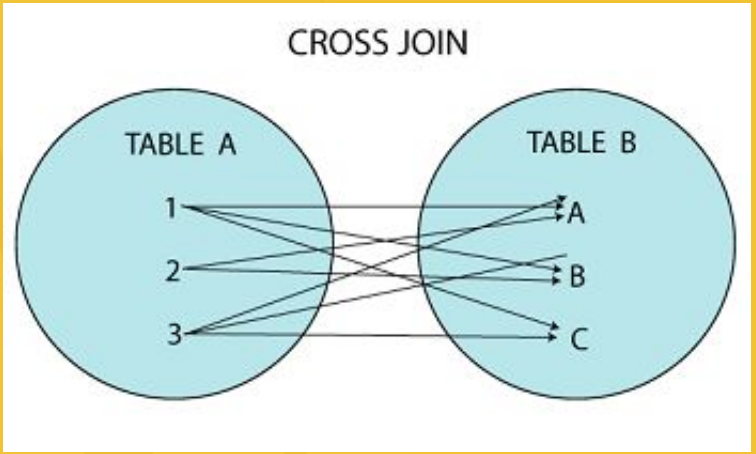
20. What Is A Join? List Different Types Of Joins.

CROSS JOIN:

- Cross join can be defined as a cartesian product of the two tables included in the join.
- It returns rows combining each row from the first table with each row of the second table.
- For Example - If we join two tables having 10 and 15 columns the Cartesian product of two tables will be 10×15=150 rows.

Syntax:

```
SELECT a.name, d.name FROM EMPLOYEE AS a
CROSS JOIN DEPARTMENT as d;
```



20. What Is A Join? List Different Types Of Joins.

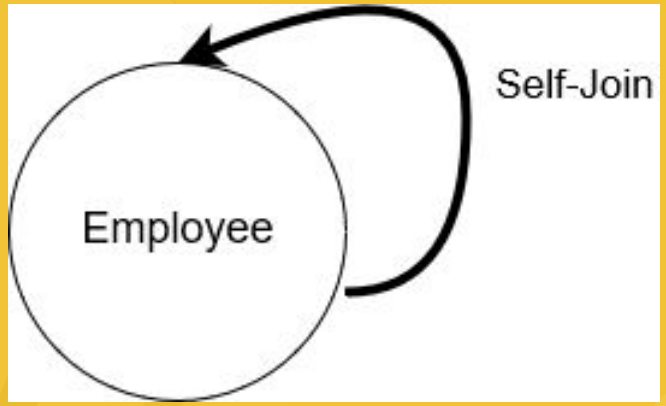
SELF JOIN:

- A SELF JOIN is used to join a table with itself.
- This join can be performed using table aliases, which allow us to avoid repeating the same table name in a single sentence.
- It will throw an error if we use the same table name more than once in a single query without using table aliases.
- A SELF JOIN is required when we want to combine data with other data in the same table itself.

Syntax:

```
SELECT column_lists
FROM TABLE1 AS T1, TABLE1 AS T2
WHERE join_conditions;
```

```
SELECT A.id AS "Emp_ID",A.name AS "Employee Name",
B.manager_id AS "Manager_ID",B.name AS "Manager Name"
FROM Employee A, Employee B
WHERE A.id = B.manager_id;
```



20. What Is A Join? List Different Types Of Joins.

OUTER JOIN: There are 3 types of Outer Joins.

LEFT OUTER JOIN:

Left join return rows which are common between the tables and all rows of Left hand side table. Simply, it returns all the rows from Left hand side table even though there are no matches in the Right hand side table.

```
SELECT * FROM Table_A A LEFT JOIN Table_B B ON A.col = B.col;
```

RIGHT OUTER JOIN:

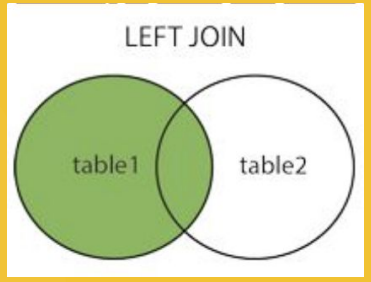
Left join return rows which are common between the tables and all rows of Left hand side table. Simply, it returns all the rows from Left hand side table even though there are no matches in the Right hand side table.

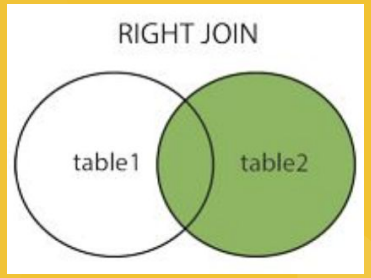
```
SELECT * FROM Table_A A RIGHT JOIN Table_B B ON A.col = B.col;
```

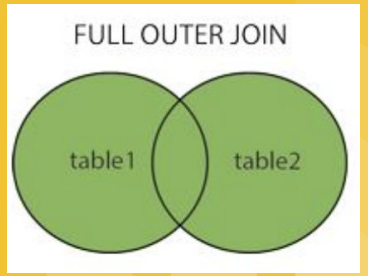
FULL OUTER JOIN:

Left join return rows which are common between the tables and all rows of Left hand side table. Simply, it returns all the rows from Left hand side table even though there are no matches in the Right hand side table.

```
SELECT * FROM Table_A A FULL JOIN Table_B B ON A.col = B.col;
```







21. What Are ‘TRUNCATE’, ‘DELETE’ And ‘DROP’ Statements In SQL?

‘DELETE’ Statement:

DELETE removes some or all rows from a table based on the condition.
It can be rolled back.

Example:

```
DELETE FROM Emp
WHERE id > 10;
```

```
mysql> select * from emp;
+----+--------+---------+---------+------------+----------+------------+---------+---------+---------+
| ID | empno  | NAME    | DEPT_ID | MANAGER_ID | PHONENO  | HIREDATE   | SALARY  | BONUS   | COMM    |
+----+--------+---------+---------+------------+----------+------------+---------+---------+---------+
|  1 | 100001 | Nitin   |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 |  500.00 |
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |   50.00 |
|  3 | 100003 | Utkarsh |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |    5.00 |
|  4 | 100004 | Arjun   |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |   50.00 |
|  5 | 100005 | Divya   |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL |  430.00 |
|  6 | 100006 | Rohit   |       1 |          8 | 9876543215 | 2020-02-27 | 15000.00 | 1000.00 |  600.00 |
|  7 | 100007 | Shreya  |       4 |         16 | 9876543216 | 2020-08-27 | 65000.00 |  600.00 |    NULL |
|  8 | 100008 | Satish  |       2 |         16 | 9876543216 | 2020-02-03 | 45000.00 | 2000.00 |    NULL |
|  9 | 100009 | Shivam  |       5 |          1 | 9876543217 | 2021-02-05 | 35000.00 |    NULL |    NULL |
| 10 | 100010 | Sonali  |       6 |          1 | 9876543218 | 2022-02-15 | 20000.00 | 1000.00 |    NULL |
| 11 | 200011 | Swati   |       3 |          1 | 9876543218 | 2022-02-23 | 30000.00 | 8000.00 | 1000.00 |
| 12 | 200012 | Sweety  |       2 |          2 | 9876543219 | 2022-09-24 | 45000.00 | 2000.00 |  800.00 |
| 13 | 200013 | Mohit   |       6 |          3 | 9876543210 | 2022-04-11 | 70000.00 | 4000.00 |  400.00 |
| 14 | 200014 | Manav   |       3 |          3 | 9876543211 | 2022-11-11 | 11000.00 | 1000.00 |  600.00 |
| 15 | 200015 | Divya   |       7 |          3 | 9876543220 | 2022-10-21 | 22000.00 | 1000.00 |   10.00 |
| 16 | 200016 | Kushal  |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |   20.00 |
| 17 | 200017 | Kiran   |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |    NULL |
| 18 | 200018 | Srishti |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |   80.00 |
| 19 | 200019 | Raghav  |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 |  505.00 |
| 20 | 200020 | Saurabh |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 |  950.00 |
+----+--------+---------+---------+------------+----------+------------+---------+---------+---------+
20 rows in set (0.00 sec)

mysql> delete from emp
    -> where id>10;
Query OK, 10 rows affected (0.04 sec)

mysql> select * from emp;
+----+--------+---------+---------+------------+----------+------------+---------+---------+---------+
| ID | empno  | NAME    | DEPT_ID | MANAGER_ID | PHONENO  | HIREDATE   | SALARY  | BONUS   | COMM    |
+----+--------+---------+---------+------------+----------+------------+---------+---------+---------+
|  1 | 100001 | Nitin   |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 |  500.00 |
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |   50.00 |
|  3 | 100003 | Utkarsh |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |    5.00 |
|  4 | 100004 | Arjun   |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |   50.00 |
|  5 | 100005 | Divya   |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL |  430.00 |
|  6 | 100006 | Rohit   |       1 |          8 | 9876543215 | 2020-02-27 | 15000.00 | 1000.00 |  600.00 |
|  7 | 100007 | Shreya  |       4 |         16 | 9876543216 | 2020-08-27 | 65000.00 |  600.00 |    NULL |
|  8 | 100008 | Satish  |       2 |         16 | 9876543216 | 2020-02-03 | 45000.00 | 2000.00 |    NULL |
|  9 | 100009 | Shivam  |       5 |          1 | 9876543217 | 2021-02-05 | 35000.00 |    NULL |    NULL |
| 10 | 100010 | Sonali  |       6 |          1 | 9876543218 | 2022-02-15 | 20000.00 | 1000.00 |    NULL |
+----+--------+---------+---------+------------+----------+------------+---------+---------+---------+
10 rows in set (0.01 sec)
```

21. What Are ‘TRUNCATE’, ‘DELETE’ And ‘DROP’ Statements In SQL?

‘TRUNCATE’ Statement:

TRUNCATE command is used to delete all the rows from the table and free the space containing the table. Truncate operation cannot be rolled back.

Example:

```
TRUNCATE TABLE emp;
```

```
mysql> select * from emp;
+----+--------+----------+---------+------------+------------+------------+---------+---------+--------+
| ID | empno  | NAME     | DEPT_ID | MANAGER_ID | PHONENO    | HIREDATE   | SALARY  | BONUS   | COMM   |
+----+--------+----------+---------+------------+------------+------------+---------+---------+--------+
|  1 | 100001 | Nitin    |       1 |          3 | 9876543210 | 2022-02-17 | 25000.00 | 1000.00 | 500.00 |
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |  50.00 |
|  3 | 100003 | Utkarsh  |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |   5.00 |
|  4 | 100004 | Arjun    |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |  50.00 |
|  5 | 100005 | Divya    |       4 |          8 | 9876543214 | 2019-02-14 | 35000.00 |    NULL | 430.00 |
|  6 | 100006 | Rohit    |       1 |          8 | 9876543215 | 2020-02-27 | 15000.00 | 1000.00 | 600.00 |
|  7 | 100007 | Shreya   |       4 |         16 | 9876543216 | 2020-08-27 | 65000.00 |  600.00 |   NULL |
|  8 | 100008 | Satish   |       2 |         16 | 9876543216 | 2020-02-03 | 45000.00 | 2000.00 |   NULL |
|  9 | 100009 | Shivam   |       5 |          1 | 9876543217 | 2021-02-05 | 35000.00 |    NULL |   NULL |
| 10 | 100010 | Sonali   |       6 |          1 | 9876543218 | 2022-02-15 | 20000.00 | 1000.00 |   NULL |
+----+--------+----------+---------+------------+------------+------------+---------+---------+--------+
10 rows in set (0.00 sec)

mysql> TRUNCATE TABLE emp;
Query OK, 0 rows affected (0.03 sec)

mysql> select * from emp;
Empty set (0.01 sec)
```

21. What Are ‘TRUNCATE’, ‘DELETE’ And ‘DROP’ Statements In SQL?

‘DROP’ Statement:

DROP command is used to remove an object from the database. If you drop a table, all the rows in the table are deleted and the table structure is removed from the database. It also cannot be retrieved back.

Example:

DROP TABLE emp;

```
mysql> show tables;
+------------------+
| Tables_in_testdb |
+------------------+
| Department       |
| department1      |
| department_copy  |
| emp              |
| EMP1             |
| employee         |
+------------------+
6 rows in set (0.01 sec)

mysql> DROP TABLE emp;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from emp;
ERROR 1146 (42S02): Table 'testdb.emp' doesn't exist
mysql> show tables;
+------------------+
| Tables_in_testdb |
+------------------+
| Department       |
| department1      |
| department_copy  |
| EMP1             |
| employee         |
+------------------+
5 rows in set (0.00 sec)
```

21. What Are 'TRUNCATE', 'DELETE' And 'DROP' Statements In SQL?

DELETE removes some or all rows from a table based on the condition. It can be rolled back.

TRUNCATE command is used to delete all the rows from the table and free the space containing the table. Truncate operation cannot be rolled back.

DROP command is used to remove an object from the database. If you drop a table, all the rows in the table are deleted and the table structure is removed from the database. It also cannot be retrieved back.

21. What Are 'TRUNCATE', 'DELETE' And 'DROP' Statements In SQL?

S.NO	TRUNCATE	DELETE
1.	The Truncate command deletes the whole contents of an existing table without the table itself. It preserves the table structure or schema.	The Delete statement removes single or multiple rows from an existing table depending on the specified condition.
2.	TRUNCATE is a DML command.	DELETE is a DML command.
3.	We cannot use the WHERE clause with TRUNCATE.	We can use the WHERE clause in the DELETE command.
4.	TRUNCATE is used to remove all the rows from a table.	DELETE is used to delete a row from a table.
5.	TRUNCATE is faster than DELETE as it deletes entire data at a time without maintaining transaction logs.	DELETE statement is slower because it maintained the log.
6.	It is not possible to roll back after using the TRUNCATE.	You can roll back data after using the DELETE.
7.	TRUNCATE query occupies less space.	DELETE query takes more space.

Source: https://www.javatpoint.com/

22. What Is An ‘Alias’ In SQL?

ALIAS:

- This command provides another name to a table or a column.
- It can be used in WHERE clause of a SQL query using the “AS” keyword.
- Aliases are created to make table or column names more readable.
- The renaming is just a temporary change and the table name does not change in the original database.
- Aliases are useful when table or column names are big or not very readable.

Example:

```
SELECT e.name AS "Employee Name",
       d.name AS "Department Name",
       e.salary AS "Monthly Salary",
       e.hiredate AS "Date Of Joining"
       FROM Employee AS e
       LEFT OUTER JOIN Department d
              on e.dept_id = d.id;
```

```
mysql> SELECT
    -> e.name AS "Employee Name",
    -> d.name AS "Department Name",
    -> e.salary AS "Monthly Salary",
    -> e.hiredate AS "Date Of Joining"
    -> FROM Employee AS e
    -> LEFT OUTER JOIN Department d
    -> on e.dept_id = d.id;
+---------------+-----------------+---------------+-----------------+
| Employee Name | Department Name | Monthly Salary | Date Of Joining |
+---------------+-----------------+---------------+-----------------+
| Nitin         | Development     |      25500.00 | 2022-02-17      |
| Pranshul      | Purchase        |      26000.00 | 2018-02-17      |
| Utkarsh       | Quality         |      27000.00 | 2019-03-13      |
| Arjun         | Marketing       |      28000.00 | 2019-07-17      |
| Rohit         | Development     |      15000.00 | 2020-02-27      |
| Shreya        | Purchase        |      25500.00 | 2020-08-27      |
| Shivam        | Marketing       |      35000.00 | 2021-02-05      |
| Sonali        | Human Resource  |      20000.00 | 2022-02-15      |
| Swati         | Sales           |      30000.00 | 2022-02-23      |
| Sweety        | Finance         |      45000.00 | 2022-09-24      |
| Mohit         | Human Resource  |      70000.00 | 2022-04-11      |
| Manav         | Sales           |      11000.00 | 2022-11-11      |
| Kushal        | Operations      |      25000.00 | 2022-08-03      |
| Kiran         | Development     |      25000.00 | 2022-07-09      |
| Srishti       | Finance         |      26000.00 | 2022-05-01      |
| Raghav        | Quality         |      31000.00 | 2019-04-17      |
| Saurabh       | Sales           |      33000.00 | 2021-03-17      |
| Divya         | Purchase        |      34500.00 | 2019-02-14      |
| Satish        | Finance         |      44500.00 | 2020-02-03      |
+---------------+-----------------+---------------+-----------------+
19 rows in set (0.00 sec)
```

23. What Are Constraints In SQL?

CONSTRAINTS:

- Constraints are used to specify the rules that we can apply to the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.
- It can be applied for single or multiple fields in an SQL table during the creation of the table or after creating using the ALTER TABLE command.
- It enforces us to store valid data and prevents us from storing irrelevant data.

SQL categories the constraints into two levels:

- **Column Level Constraints:** These constraints are only applied to a single column and limit the type of data that can be stored in that column.
- **Table Level Constraints:** These constraints are applied to the entire table and limit the type of data that can be entered.

The following are the constraints in SQL:

NOT NULL - Restricts NULL value from being inserted into a column.
CHECK - Verifies that all values in a field satisfy a condition.
DEFAULT - Automatically assigns a default value if no value has been specified for the field.
INDEX - Indexes a field providing faster retrieval of records.
UNIQUE - Ensures unique values to be inserted into the field.
PRIMARY - Uniquely identifies each record in a table.
FOREIGN - Ensures referential integrity for a record in another table.

24. What Is The Difference Between 'WHERE' And 'HAVING' Clauses In SQL?

SQL clause helps to limit the result set by providing a condition to the query. A clause helps to filter the rows from the entire set of records.

For example – WHERE, HAVING clause.

S. No.	Where Clause	Having Clause
1	It can be used without the GROUP by clause	It cannot be used without the GROUP BY clause
2	It selects rows before grouping	It selects rows after grouping
3	It cannot contain aggregate functions	It can contain aggregate functions
4	It is used to impose a condition on SELECT statement as well as single row function and is used before GROUP BY clause	It is used to impose a condition on GROUP Function and is used after GROUP BY clause in the query
6	It is implemented in row operations.	It is implemented in column operations.
7	It does not allow to work with aggregate functions.	It can work with aggregate functions.
8	It can be used with the SELECT, UPDATE, & DELETE statements.	It can only be used with the SELECT statement.

25. What Is The Difference Between ‘IN’ And ‘BETWEEN’ Operators In SQL?

BETWEEN operator is used to display rows based on a range of values in a row whereas **IN** condition operator is used to check for values contained in a specific set of values.

Example of BETWEEN:

```
SELECT * FROM employee
    WHERE salary
    BETWEEN 25000 and 35000;
```

(Note: Both Values will be included)

Example of IN:

```
SELECT * FROM employee
    WHERE salary
    IN(23000,25000,26000,30000);
```

```
mysql> SELECT * FROM employee WHERE salary BETWEEN 25000 and 35000;
+----+--------+----------+---------+------------+------------+------------+----------+---------+
| ID | EMPNO  | NAME     | DEPT_ID | manager_id | PHONENO    | HIREDATE   | SALARY   | BONUS   | COMM    |
+----+--------+----------+---------+------------+------------+------------+----------+---------+
|  1 | 100001 | Nitin    |       1 |          3 | 9876543210 | 2022-02-17 | 25500.00 | 1000.00 |  500.00 |
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |   50.00 |
|  3 | 100003 | Utkarsh  |       8 |          8 | 9876543212 | 2019-03-13 | 27000.00 | 1000.00 |    5.00 |
|  4 | 100004 | Arjun    |       5 |          8 | 9876543213 | 2019-07-17 | 28000.00 |    NULL |   50.00 |
|  7 | 100007 | Shreya   |       4 |         16 | 9876543216 | 2020-08-27 | 25500.00 |  600.00 |    NULL |
|  9 | 100009 | Shivam   |       5 |          1 | 9876543217 | 2021-02-05 | 35000.00 |    NULL |    NULL |
| 11 | 200011 | Swati    |       3 |          1 | 9876543218 | 2022-02-23 | 30000.00 | 8000.00 | 1000.00 |
| 16 | 200016 | Kushal   |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |   20.00 |
| 17 | 200017 | Kiran    |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |    NULL |
| 18 | 200018 | Srishti  |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |   80.00 |
| 19 | 200019 | Raghav   |       8 |          7 | 9876543223 | 2019-04-17 | 31000.00 |  200.00 |  505.00 |
| 20 | 200020 | Saurabh  |       3 |         13 | 9876543224 | 2021-03-17 | 33000.00 |  900.00 |  950.00 |
| 21 | 200021 | Divya    |       4 |          8 | 9876543214 | 2019-02-14 | 34500.00 |    NULL |  430.00 |
+----+--------+----------+---------+------------+------------+------------+----------+---------+
13 rows in set (0.00 sec)

mysql> SELECT * FROM employee WHERE salary IN(23000,25000,26000,30000);
+----+--------+----------+---------+------------+------------+------------+----------+---------+---------+
| ID | EMPNO  | NAME     | DEPT_ID | manager_id | PHONENO    | HIREDATE   | SALARY   | BONUS   | COMM    |
+----+--------+----------+---------+------------+------------+------------+----------+---------+---------+
|  2 | 100002 | Pranshul |       4 |          8 | 9876543211 | 2018-02-17 | 26000.00 | 1000.00 |   50.00 |
| 11 | 200011 | Swati    |       3 |          1 | 9876543218 | 2022-02-23 | 30000.00 | 8000.00 | 1000.00 |
| 16 | 200016 | Kushal   |       7 |          3 | 9876543220 | 2022-08-03 | 25000.00 | 3000.00 |   20.00 |
| 17 | 200017 | Kiran    |       1 |          6 | 9876543222 | 2022-07-09 | 25000.00 |  100.00 |    NULL |
| 18 | 200018 | Srishti  |       2 |         12 | 9876543221 | 2022-05-01 | 26000.00 |  100.00 |   80.00 |
+----+--------+----------+---------+------------+------------+------------+----------+---------+---------+
5 rows in set (0.00 sec)
```

25. What Is The Difference Between ‘IN’ And ‘BETWEEN’ Operators In SQL?

S.No	BETWEEN OPERATOR	IN OPERATOR
1.	This operator is used to selects the range of data between two values. The values can be numbers, text, and dates as well.	It is a logical operator to determine whether or not a specific value exists within a set of values. This operator reduces the use of multiple OR conditions with the query.
2.	It returns records whose column value lies in between the defined range.	It compares the specified column's value and returns the records when the match exists in the set of values.
3.	Syntax: **SELECT * FROM employee** **WHERE salary** **BETWEEN 25000 and 35000;**	Syntax: **SELECT * FROM employee** **WHERE salary** **IN(23000,25000,26000,30000);**

26. What Is The Default Ordering Of Data Using The ORDER BY Clause? How Could It Be Changed?

ORDER BY:

- ❏ The ORDER BY clause is used to sort the table data either in ascending or descending order.
- ❏ By default ORDER BY sorts the data in ASCENDING ORDER.
- ❏ If we want to change its default behavior, we need to use the DESC keyword after the column name in the ORDER BY clause.

Syntax:

```
SELECT expressions FROM tables
ORDER BY expression [ASC | DESC];
```

Example:

```
SELECT name, salary FROM EMPLOYEE
ORDER BY SALARY ASC;

SELECT name, salary FROM EMPLOYEE
ORDER BY SALARY DESC;
```

```
mysql> SELECT name, salary FROM EMPLOYEE
    -> ORDER BY SALARY ASC;
+----------+----------+
| name     | salary   |
+----------+----------+
| Manav    | 11000.00 |
| Rohit    | 15000.00 |
| Sonali   | 20000.00 |
| Kushal   | 25000.00 |
| Kiran    | 25000.00 |
| Nitin    | 25500.00 |
| Shreya   | 25500.00 |
| Pranshul | 26000.00 |
| Srishti  | 26000.00 |
| Utkarsh  | 27000.00 |
| Arjun    | 28000.00 |
| Swati    | 30000.00 |
| Raghav   | 31000.00 |
| Saurabh  | 33000.00 |
| Divya    | 34500.00 |
| Shivam   | 35000.00 |
| Satish   | 44500.00 |
| Sweety   | 45000.00 |
| Mohit    | 70000.00 |
+----------+----------+
19 rows in set (0.00 sec)
```

```
mysql> SELECT name, salary FROM EMPLOYEE
    -> ORDER BY SALARY DESC;
+----------+----------+
| name     | salary   |
+----------+----------+
| Mohit    | 70000.00 |
| Sweety   | 45000.00 |
| Satish   | 44500.00 |
| Shivam   | 35000.00 |
| Divya    | 34500.00 |
| Saurabh  | 33000.00 |
| Raghav   | 31000.00 |
| Swati    | 30000.00 |
| Arjun    | 28000.00 |
| Utkarsh  | 27000.00 |
| Pranshul | 26000.00 |
| Srishti  | 26000.00 |
| Nitin    | 25500.00 |
| Shreya   | 25500.00 |
| Kushal   | 25000.00 |
| Kiran    | 25000.00 |
| Sonali   | 20000.00 |
| Rohit    | 15000.00 |
| Manav    | 11000.00 |
+----------+----------+
19 rows in set (0.00 sec)
```

27. What Is DISTINCT Statement?

DISTINCT:

- The DISTINCT keyword is used to ensure that the fetched value always has unique values.
- It does not allow to have duplicate values.
- The DISTINCT keyword is used with the SELECT statement and retrieves different values from the table's column.

Syntax:

SELECT DISTINCT column FROM table;

Example:

SELECT DISTINCT dept_id FROM Employee;

```
mysql> SELECT DISTINCT dept_id FROM Employee;
+---------+
| dept_id |
+---------+
|       1 |
|       2 |
|       3 |
|       4 |
|       5 |
|       6 |
|       7 |
|       8 |
+---------+
8 rows in set (0.00 sec)
```

28. What Are Aggregate Functions. How many Aggregate Functions Are Available In SQL?

AGGREGATE FUNCTIONS:

- Aggregate functions are used to evaluate mathematical calculation and return single values.
- Aggregate functions are often used with the GROUP BY and HAVING clauses of the SELECT statement.
- This can be calculated from the columns in a table.

SQL provides Five(5) aggregate functions:

COUNT(): returns the number of table rows, including rows with null values.
SELECT COUNT(*) FROM Employee;

AVG(): returns the average value from specified columns.
SELECT AVG(salary) FROM Employee;

MAX(): returns the largest value among the group.
SELECT MAX(salary) FROM Employee;

MIN(): returns the smallest value among the group.
SELECT MIN(salary) FROM Employee;

SUM(): returns the total summed values(non-null) of the specified column.
SELECT SUM(salary) FROM Employee;

```
mysql> SELECT COUNT(*) FROM Employee;
+----------+
| COUNT(*) |
+----------+
|       19 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT AVG(salary) FROM Employee;
+--------------+
| AVG(salary)  |
+--------------+
| 30368.421053 |
+--------------+
1 row in set (0.00 sec)

mysql> SELECT MIN(salary) FROM Employee;
+-------------+
| MIN(salary) |
+-------------+
|    11000.00 |
+-------------+
1 row in set (0.00 sec)

mysql> SELECT MAX(salary) FROM Employee;
+-------------+
| MAX(salary) |
+-------------+
|    70000.00 |
+-------------+
1 row in set (0.00 sec)

mysql> SELECT SUM(salary) FROM Employee;
+-------------+
| SUM(salary) |
+-------------+
|   577000.00 |
+-------------+
1 row in set (0.00 sec)
```

29. What Are The Subsets Of SQL?

DDL (Data Definition Language):

- DDL defines the data structure.
- DDL deals with database schemas and descriptions of how the data should reside in the database.
- DDL Commands: CREATE, ALTER, DROP, TRUNCATE, COMMENT, RENAME

DML (Data Manipulation Language):

- DML is used to store, modify, retrieve, delete and update data in a database.
- DML deals with data manipulation of existing data in the database.
- DML Commands: SELECT, INSERT, UPDATE, DELETE

DCL (Data Control Language):

- DCL controls access to the data stored in the database.
- DCL mostly concerned with rights, permissions and other controls of the database system.
- DCL Commands: GRANT, REVOKE

TCL (Transaction Control Language):

- TCL is used to deal with the transaction operations in the database.
- TCL Commands: COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION

30. List Different Types Of Relationships In SQL.

Database Relationship:

Database Relationship is defined as the connection between the tables in a database.
Following are the relationships in SQL.

- **One-to-One relationship** - This can be defined as the relationship between two tables where each record in one table is associated with the maximum of one record in the other table.
- **One-to-Many & Many-to-One relationships** - This is the most commonly used relationship where a record in a table is associated with multiple records in the other table.
- **Many-to-Many relationship**- This is used in cases when multiple instances on both sides are needed for defining a relationship.
- **Self-Referencing Relationships** - This is used when a table needs to define a relationship with itself.

31. What Is RDBMS? How Is It Different From DBMS?

RDBMS:

- RDBMS stands for Relational Database Management System that stores data in the form of a collection of tables, and relations can be defined between the common fields of these tables.
- It also provides relational operators to manipulate the data stored into the tables.
- Examples of relational database management systems are Microsoft Access, MySQL, SQL Server, Oracle database, etc.

31. What Is RDBMS? How Is It Different From DBMS?

S.No	DBMS	RDBMS
1.	DBMS applications store data as file.	RDBMS applications store data in a tabular form.
2.	In DBMS, data is generally stored in either a hierarchical form or a navigational form.	In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables.
3.	Normalization is not present in DBMS.	Normalization is present in RDBMS.
4.	DBMS does not apply any security with regards to data manipulation.	RDBMS defines the integrity constraint for the purpose of ACID (Atomicity, Consistency, Isolation and Durability) property.
5.	DBMS uses file system to store data, so there will be no relation between the tables.	in RDBMS, data values are stored in the form of tables, so a relationship between these data values will be stored in the form of a table as well.
7.	DBMS does not support distributed database.	RDBMS supports distributed database.
8.	Examples of DBMS are file systems, xml etc.	Example of RDBMS are mysql, postgre, sql server, oracle etc.

Source: https://www.javatpoint.com/difference-between-dbms-and-rdbms

32. What Is A QUERY?

QUERY:

- An SQL query is used to retrieve the required data from the database.
- A query is a code written in order to get the information back from the database.
- However, there may be multiple SQL queries that yield the same results but with different levels of efficiency.
- An inefficient query can drain the database resources, reduce the database speed or result in a loss of service for other users.
- So it is very important to optimize the query to obtain the best database performance.
- **Syntax**:

```
SELECT * FROM EMP;
```

33. What Is A SUB-QUERY? What Are Its Types?

SUB-QUERY:

- A Subquery can be simply defined as a query within another query.
- The outer query is called as **MAIN QUERY,** and inner query is called **SUB-QUERY.**
- SubQuery is always executed first, and the result of subquery is passed on to the main query.
- A subquery can also use any comparison operators such as >,< or =.
- For example:

SELECT MAX(salary) FROM Employee
WHERE Salary<(SELECT MAX(Salary) FROM Employee
WHERE Salary<(SELECT MAX(Salary) FROM Employee));

```
mysql> select max(salary) from employee
    -> where salary<(select max(salary) from employee
    -> where salary<(select max(salary) from employee));
+-------------+
| max(salary) |
+-------------+
|    51000.00 |
+-------------+
1 row in set (0.00 sec)
```

33. What Is A SUB-QUERY? What Are Its Types?

SUB-QUERY:

- A Subquery can be simply defined as a query within another query.
- The outer query is called as **MAIN QUERY,** and inner query is called **SUB-QUERY.**
- SubQuery is always executed first, and the result of subquery is passed on to the main query.
- A subquery can also use any comparison operators such as >,< or =.
- For example:

```
SELECT MAX(salary) FROM Employee
WHERE Salary<(SELECT MAX(Salary) FROM Employee
WHERE Salary<(SELECT MAX(Salary) FROM  Employee));
```

```
mysql> select max(salary) from employee
    -> where salary<(select max(salary) from employee
    -> where salary<(select max(salary) from employee));
+-------------+
| max(salary) |
+-------------+
|    51000.00 |
+-------------+
1 row in set (0.00 sec)
```

Types Of SUB-QUERY:

There are two types of subquery – Correlated and Non-Correlated.

- **Correlated Subquery** - A correlated subquery cannot be considered as independent query, but it can refer the column in a table listed in the FROM the list of the main query.
- **Non-Correlated Subquery** - It can be considered as independent query and the output of subquery are substituted in the main query.

34. What Is A Self-Join. What Is Its Requirement.

SELF JOIN:

- A SELF JOIN is used to join a table with itself.
- This join can be performed using table aliases, which allow us to avoid repeating the same table name in a single sentence.
- It will throw an error if we use the same table name more than once in a single query without using table aliases.
- A SELF JOIN is required when we want to combine data with other data in the same table itself.

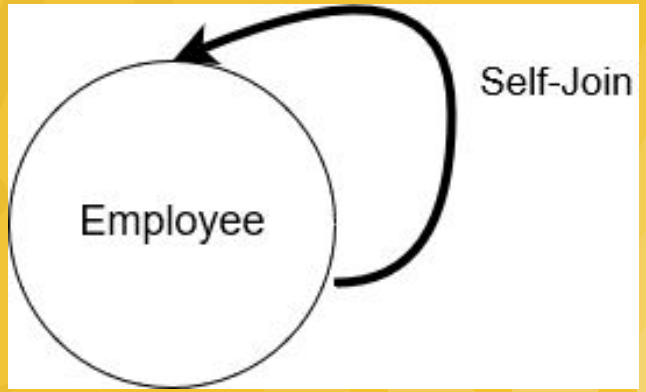
Syntax:

```
SELECT column_lists
FROM TABLE1 AS T1, TABLE1 AS T2
WHERE join_conditions;

SELECT A.id AS "Emp_ID",A.name AS "Employee Name",
B.manager_id AS "Manager_ID",B.name AS "Manager Name"
FROM Employee A, Employee B
WHERE A.id = B.manager_id;
```

REQUIREMENT OF SELF JOIN:

- A SELF JOIN is required when we want to combine data with other data in the same table itself.
- It is often very useful to convert a hierarchical structure to a flat structure.



Source: https://www.javatpoint.com/

35. What Is The Difference Between NOW() And CURRENT_DATE()?

NOW():

NOW() Command will fetch the current date and time both in format 'YYYY-MM-DD HH:MM:SS'

```
SELECT NOW();
```

CURRENT_DATE():

CURRENT_DATE() Command will fetch the date of the current day in format 'YYYY-MM-DD'.

```
SELECT CURRENT_DATE();
```

```
mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2022-02-27 00:48:39 |
+---------------------+
1 row in set (0.00 sec)

mysql> SELECT CURRENT_DATE();
+----------------+
| CURRENT_DATE() |
+----------------+
| 2022-02-27     |
+----------------+
1 row in set (0.00 sec)
```

36. Which Function Is Used To Return Remainder In A Division Operator In SQL?

The **MOD function** returns the remainder in a division operation.
Example:

SELECT MOD(27,5) AS “Value”;

```
mysql> Select MOD(27,5) as "Remainder Value";
+-----------------+
| Remainder Value |
+-----------------+
|               2 |
+-----------------+
1 row in set (0.00 sec)

mysql> Select MOD(31,7) as "Remainder Value";
+-----------------+
| Remainder Value |
+-----------------+
|               3 |
+-----------------+
1 row in set (0.01 sec)

mysql> Select MOD(-25,7) as "Remainder Value";
+-----------------+
| Remainder Value |
+-----------------+
|              -4 |
+-----------------+
1 row in set (0.01 sec)
```

37. What Is The Main Disadvantage Of Deleting Data From An Existing Table Using The DROP TABLE Command?

- DROP TABLE command deletes complete data from the table along with removing the complete table structure too.
- In case our requirement was only just remove the data, then we would need to recreate the table to store data in it.
- In such cases, it is advised to use the TRUNCATE command.

38. What Is The Difference Between CHAR And VARCHAR2 Datatype In SQL?

SR.NO	CHAR	VARCHAR
1.	CHAR datatype is used to store character string of fixed length	VARCHAR datatype is used to store character string of variable length
2.	In CHAR, If the length of string is less than set or fixed length then it is padded with extra memory space.	In VARCHAR, If the length of string is less than set or fixed length then it will store as it is without padded with extra memory spaces.
3.	CHAR stands for “Character”	VARCHAR stands for “Variable Character”
4.	Storage size of CHAR datatypes is equal to n bytes i.e. set length	Storage size of VARCHAR datatype is equal to the actual length of the entered string in bytes.
5.	We should use CHAR datatype when we expect the data values in a column are of same length.	We should use VARCHAR datatype when we expect the data values in a column are of variable length.
6.	CHAR take 1 byte for each character	VARCHAR take 1 byte for each character and some extra bytes for holding length information
7.	Better performance than VARCHAR	Performance is not good as compared to CHAR

Source: https://www.geeksforgeeks.org/char-vs-varchar-in-sql/

39. What Are Scalar Functions?

SCALAR FUNCTIONS():

Scalar functions are the built-in functions in SQL, and whatever be the input provided to the scalar functions, the output returned by these functions will always be a single value.

Following are the widely used SQL scalar functions:

LENGTH() - Calculates the total length of the given field (column).
SELECT name, LENGTH(name) FROM Department;

UCASE() - Converts a collection of string values to uppercase characters.
SELECT UCASE(name) FROM Department;

LCASE() - Converts a collection of string values to lowercase characters.
SELECT LCASE(name) FROM Department;

MID() - Extracts substrings from a collection of string values in a table.
SELECT MID(name,2,4) FROM Department;

ROUND() - Calculates the round-off integer value for a numeric field (or decimal point values).
SELECT Round(26.837352835283528,4) as Value;

NOW() - Returns the current date & time.
SELECT NOW() AS Today;

CONCAT() - Concatenates two or more strings.
SELECT CONCAT(id, name, location) FROM Department;

RAND() - Generates a random collection of numbers of a given length.

FORMAT() - Sets the format to display a collection of values.

```
mysql> Select name, Length(name) from Department;
+----------------+--------------+
| name           | Length(name) |
+----------------+--------------+
| Development    |           11 |
| Finance        |            7 |
| Sales          |            5 |
| Purchase       |            8 |
| Marketing      |            9 |
| Human Resource |           14 |
| Operations     |           10 |
| Quality        |            7 |
+----------------+--------------+
8 rows in set (0.03 sec)
```

```
mysql> Select UCASE(name) from Department;
+----------------+
| UCASE(name)    |
+----------------+
| DEVELOPMENT    |
| FINANCE        |
| SALES          |
| PURCHASE       |
| MARKETING      |
| HUMAN RESOURCE |
| OPERATIONS     |
| QUALITY        |
+----------------+
8 rows in set (0.01 sec)
```

```
mysql> Select LCASE(name) from Department;
+----------------+
| LCASE(name)    |
+----------------+
| development    |
| finance        |
| sales          |
| purchase       |
| marketing      |
| human resource |
| operations     |
| quality        |
+----------------+
8 rows in set (0.00 sec)
```

```
mysql> Select MID(name,2,4) from Department;
+----------------+
| MID(name,2,4)  |
+----------------+
| evel           |
| inan           |
| ales           |
| urch           |
| arke           |
| uman           |
| pera           |
| uali           |
+----------------+
8 rows in set (0.00 sec)
```

```
mysql> Select Round(26.837352835283528,4) as Value;
+---------+
| Value   |
+---------+
| 26.8374 |
+---------+
1 row in set (0.00 sec)
```

```
mysql> SELECT NOW() AS Today;
+---------------------+
| Today               |
+---------------------+
| 2022-02-27 00:00:29 |
+---------------------+
1 row in set (0.00 sec)
```

```
mysql> select CONCAT(id,name,location) from department;
+--------------------------+
| CONCAT(id,name,location) |
+--------------------------+
| 1DevelopmentPune         |
| 2FinanceGurgaon          |
| 3SalesDelhi              |
| 4PurchaseMumbai          |
| 5MarketingBangalore      |
| 6Human ResourceNoida     |
| 7OperationsHyderabad     |
| 8QualityChennai          |
+--------------------------+
8 rows in set (0.00 sec)
```

40. What is A Database?

DATABASE:

- A database is an organized collection of data for easy access, storing, retrieval and managing of data.
- This is also known as structured form of data which can be accessed in many ways.
- Almost every organization uses the database for storing the data due to its easily accessible and high operational ease.
- The database provides perfect access to data and lets us perform required tasks.
- It is also the collection of schemas, tables, queries, views, etc.
- Example: School Management Database, Bank Management Database.

The following are the common features of a database:

- Manages large amounts of data
- Accurate
- Easy to update
- Security
- Data integrity
- Easy to research data

41. What Is SQL? What Is The Difference Between NoSQL vs SQL Databases?

SQL:

- SQL is a structured query language used in a database.
- It is used for query and operating database system.
- SQL is the core of the relational database which is used for accessing and managing database.

41. What Is SQL? What Is The Difference Between NoSQL vs SQL Databases?

Parameter	SQL	NoSQL
Define	SQL databases are a type of system software that supports management, analysis, capturing and querying the structured data in a relational format.	NoSQL databases are a type of software that allows to maintain and retrieve structured, unstructured, polymorphic data for different purposes.
Purpose	A language used to communicate with databases for storage, deletion, updation, insertion and retrieval of data.	A software to retrieve, store and manage scalability of databases.
Development Year	SQL was developed in the year 1970 for flat file storage problems.	NoSQL was developed in 2000 as an enhanced version for SQL databases for unstructured and semi-structured data.
Query Language	SQL databases support Structured Query Languages.	NonSQL does not have any declarative query language.
Type	Supports table based data type.	Supports document oriented, graph databases, key value pair-based.

Source: https://www.interviewbit.com/blog/sql-vs-nosql/

41. What Is SQL? What Is The Difference Between NoSQL vs SQL Databases?

Parameter	SQL	NoSQL
Scalability	Vertically Scalable (Add resources to increase the capacity of the existing hardware and software).	Horizontally Scalable (Changing small nodes with larger nodes to increase the capacity of the existing hardware and software).
Schemas	SQL supports predefined schemas, making the storage of data restrictive to structured type only.	Nosql supports dynamic schemas to store different forms of data.
Architecture	SQL is relational.	Non-SQL is non relational.
Ideal Use Cases	SQL is best suitable for complex queries, multi-row transactions.	NoSQL is best suited for unstructured data or documents. Not ideal for complex queries.
Hardware	Databases that support SQL require powerful hardware to support vertical scaling.	NonSQL databases require commodity hardware for horizontal scaling.
Properties	SQL enables ACID(atomicity, consistency, isolation, and durability) properties	NonSQL follows CAP (consistency, availability, partition tolerance) properties.

Source: https://www.interviewbit.com/blog/sql-vs-nosql/

41. What Is SQL? What Is The Difference Between NoSQL vs SQL Databases?

Parameter	SQL	NoSQL
Data Storage	SQL does not support hierarchical storage of data.	NoSQL is best suited for hierarchical storage of data.
Distributed Data	SQL databases can only be run on a single system and hence, does not follow distribution of data.	NoSQL databases are designed to follow data distribution features like repetition, partition.
Best Features	• Secure • Cross Platform Support • Free	• High Performance • Flexible • Easy to use
Top Companies Using	Microsoft, Dell, Cognizant, etc.	Amazon, Capgemini, Adobe, etc.
Examples	SQL supports databases like MySQL, SQL Server, Oracle, etc.	Nosql databases are Hbase, MongoDB, Redis, etc.

Source: https://www.interviewbit.com/blog/sql-vs-nosql/

42. What Is The Difference Between SQL And MySQL?

S.No.	SQL	MySQL
1.	It is a structured query language used in a database	It is a database management system
2.	It is used for query and operating database system	It allows data handling, storing, and modifying data in an organized manner.
3.	SQL is always the same	MySQL keeps updating
4.	Only a single storage engine is supported in SQL.	MySQL supports multiple storage engines.
5.	The server is independent in SQL	During backup sessions, the server blocks the database.
6.	SQL is a standard language which stands for Structured Query Language based on the English language	MySQL is a database management system.
7.	SQL is the core of the relational database which is used for accessing and managing database	MySQL is an RDMS (Relational Database Management System) such as SQL Server, Informix etc.
8.	SQL is a programming language, so that it does not get any updates. Its commands are always fixed and remain the same.	MySQL is software, so it gets frequent updation.

43. What Do You Mean By DBMS? What Are Its Different Types?

DBMS:

- A Database Management System (DBMS) is a software application that interacts with the user, applications, and the database itself to capture and analyze data. (A database is a structured collection of data.)
- DBMS is a system software responsible for the creation, retrieval, updation, and management of the database.
- A DBMS allows a user to interact with the database. The data stored in the database can be modified, retrieved and deleted and can be of any type like strings, numbers, images, etc.
- Without the database management system, it would be far more difficult for the user to access the database's data.

There are two types of DBMS:

- **Relational Database Management System:** The data is stored in relations (tables). Example – MySQL, PostGres etc
- **Non-Relational Database Management System:** There is no concept of relations, tuples and attributes. Example – MongoDB

44. Is A Blank Space or Zero Same As A NULL Value?

No. The NULL value is not the same as zero or a blank space.
The following points explain their main differences:

- A NULL value is a value, which is 'unavailable, unassigned, unknown or not applicable.' It would be used in the absence of any value. We can perform arithmetic operations on it. On the other hand, zero is a number, and a blank space is treated as a character.

- In SQL, zero or blank space can be compared with another zero or blank space. whereas one null may not be equal to another null. null means data might not be provided or there is no data.

45. What Is Union, Union ALL, Minus And Intersect?

UNION:

The UNION operator combines the results of two or more Select statements by removing duplicate rows. The columns and the data types must be the same in the SELECT statements.

```
SELECT columns FROM table1   UNION   SELECT columns FROM table2;
```

UNION ALL:

This operator is similar to the Union operator, but it does not remove the duplicate rows from the output of the SELECT statements.

```
SELECT columns FROM table1   UNION ALL   SELECT columns FROM table2;
```

INTERSECT:

This operator returns the common records from two or more SELECT statements.

```
SELECT columns FROM table1   INTERSECT   SELECT columns FROM table2;
```

MINUS:

This operator returns the records from the first query, which is not found in the second query. It does not return duplicate values.

```
SELECT columns FROM table1   MINUS   SELECT columns FROM table2;
```

Before running either of the above SQL statements, certain requirements must be satisfied –

- ❏ Within the clause, each SELECT query must have the same amount of columns.
- ❏ The data types in the columns must also be comparable.
- ❏ In each SELECT statement, the columns must be in the same order

46. What Are The ACID Properties In A Database?

ACID PROPERTIES:

- ❏ The ACID properties are meant for the transaction that goes through a different group of tasks.
- ❏ A transaction is a single logical order of data.
- ❏ It provides properties to maintain consistency before and after the transaction in a database.
- ❏ It also ensures that the data transactions are processed reliably in a database system.

The ACID property is an acronym for **Atomicity, Consistency, Isolation, and Durability.**

Atomicity: It ensures that all statements or operations within the transaction unit must be executed successfully. If one part of the transaction fails, the entire transaction fails, and the database state is left unchanged. Its main features are COMMIT, ROLLBACK, and AUTO-COMMIT.

Consistency: This property ensures that the data must meet all validation rules. In simple words, we can say that the database changes state only when a transaction will be committed successfully. It also protects data from crashes.

Isolation: This property guarantees that the concurrent property of execution in the transaction unit must be operated independently. It also ensures that statements are transparent to each other. The main goal of providing isolation is to control concurrency in a database.

Durability: This property guarantees that once a transaction has been committed, it persists permanently even if the system crashes, power loss, or failed.

Source: https://www.javatpoint.com/

47. What Is Normalization? What Are The Various Forms Of Normalization?

NORMALIZATION:

- Generally, in a table, we will have a lot of redundant information which is not required, so it is better to divide this complex table into multiple smaller tables which contains only unique information.
- The process of table design to minimize the data redundancy is called normalization.
- In other words, Normalization in SQL is the process of organizing data to avoid duplication and redundancy.
- It includes the creation of tables, establishing relationships between them, and defining rules for those relationships.
- The main aim of Normalization is to add, delete or modify field that can be made in a single table.

Some of the advantages are:

- Reduction of redundant and duplicate data
- Quickly find the information
- More Tables with smaller rows
- Better Database organization

47. What Is Normalization In A Database? What Are The Various Forms Of Normalization?

VARIOUS FORMS OF NORMALIZATION:

There are many successive levels of normalization. These are called **Normal Forms.**
Each consecutive normal form depends on the previous one.

FIRST NORMAL FORM (1NF):

A relation schema is in 1NF, if and only if:

- ❏ All attributes in the relation are atomic(indivisible value)
- ❏ And there are no repeating elements or group of elements.

SECOND NORMAL FORM (2NF):

A relation is said to be in 2NF, if and only if:

- ❏ It is in 1NF.
- ❏ And creates & places data subsets in an individual table and defines relationship between tables using primary key.

THIRD NORMAL FORM (3NF):

A relation is said to be in 3NF if and only if:

- ❏ It is in 2NF.
- ❏ And removes those columns which are not related through the primary key.

FOURTH NORMAL FORM (4NF):

- ❏ Meeting all the requirements of 3NF and does not define multi-valued dependencies.
- ❏ 4NF is also known as BCNF.

48. What Are The Disadvantages Of Not Performing Database Normalization?

THE DISADVANTAGES OF NOT PERFORMING NORMALIZATION:

- The occurrence of redundant terms in the database causes the waste of space in the disk.
- Due to redundant terms, inconsistency may also occur.
- If any change is made in the data of one table but not made in the same data of another table, then inconsistency will occur.
- This inconsistency will lead to the maintenance problem and effects the ACID properties as well.

49. What Is Denormalization?

DENORMALIZATION:

- Denormalization is the inverse process of normalization, where the normalized schema is converted into a schema that has redundant information.
- Denormalization is a technique used to access the data from higher to lower normal forms of database.
- The performance is improved by using redundancy and keeping the redundant data consistent.
- The reason for performing denormalization is the overheads produced in the query processor by an over-normalized structure.
- Denormalization doesn't mean that normalization will not be done. It is an optimization strategy that takes place after the normalization process.
- It adds redundant terms into the tables to avoid complex joins and many other complex operations.

50. What Is An Index? Explain Different Types Of Indexes In SQL?

DATABASE INDEX:

- A database index is a data structure that provides a quick lookup of data in a column or columns of a table.
- An index creates an entry for each value and hence it will be faster to retrieve data.
- It also has a unique value meaning that the index cannot be duplicated.
- It is used to increase the performance and allow faster retrieval of records from the table.
- Indexes help speed up searching in the database.
- If there is no index on any column in the WHERE clause, then SQL Server has to skim through the entire table and check each and every row to find matches, which might result in slow operation on large data.
- **Syntax**:

```
CREATE INDEX INDEX_NAME ON TABLE_NAME (COLUMN)
```

DIFFERENT TYPES OF INDEX:

- Clustered Index
- Non-Clustered Index

51. What Is The Difference Between Clustered And Non-Clustered Index?

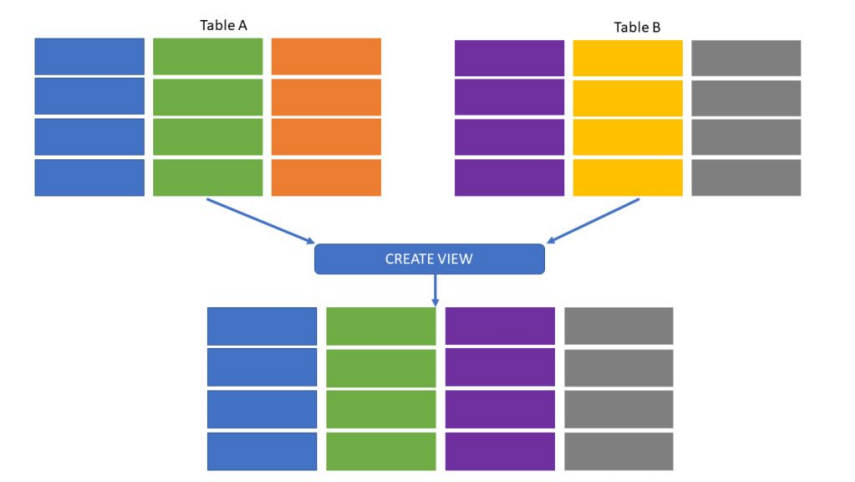
S.NO	Clustered Index	Non-Clustered Index
1.	A clustered index is a table or view where the data for the rows are stored. In a relational database, if the table column contains a primary key, MySQL automatically creates a clustered index named PRIMARY.	The indexes other than PRIMARY indexes (clustered indexes) are called non-clustered indexes. It has a structure separate from the data row. The non-clustered indexes are also known as secondary indexes.
2.	Clustered indexes store the data information and the data itself.	Non-clustered indexes stores only the information, and then it will refer you to the data stored in clustered data.
3.	There can only be one clustered index per table.	There can be one or more non-clustered indexes in a table.
4.	A clustered index determines how data is stored physically in the table. Therefore, reading from a clustered index is faster.	It creates a logical ordering of data rows and uses pointers for accessing the physical data files. Therefore, reading from a clustered index is slower.
5.	A clustered index always contains an index id of 0.	A non-clustered index always contains an index id>0.

Source: https://www.javatpoint.com/

52. What Is A View In Database?

DATABASE VIEW:

- A view in SQL is a virtual table based on the result-set of SQL statement.
- A view contains rows and columns, just like a real table.
- The fields in a view are fields from one or more real tables in the database.
- View can have data of one or more tables combined, and it is depending on the relationship.
- If any changes occur in the existing table, the same changes reflected in the views also.



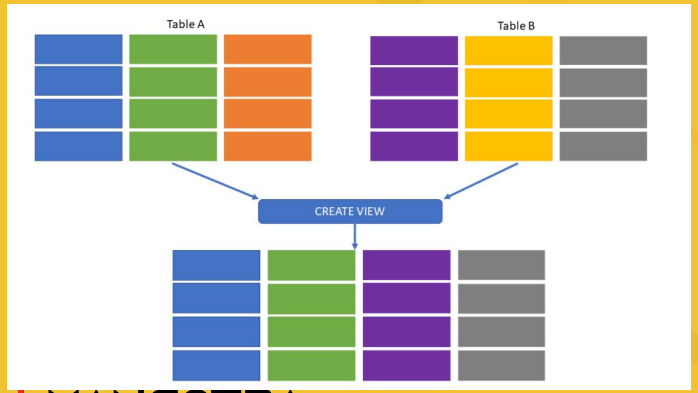
52. What Is A View In Database?

DATABASE VIEW:

- A view in SQL is a virtual table based on the result-set of SQL statement.
- A view contains rows and columns, just like a real table.
- The fields in a view are fields from one or more real tables in the database.
- View can have data of one or more tables combined, and it is depending on the relationship.
- If any changes occur in the existing table, the same changes reflected in the views also.

Example:

```
SELECT e.name AS "Employee Name",
    d.name AS "Department Name",
    e.salary AS "Monthly Salary",
    e.hiredate AS "Date Of Joining"
    FROM Employee AS e
       LEFT OUTER JOIN Department d
       on e.dept_id = d.id;
```




```
mysql> SELECT
    -> e.name AS "Employee Name",
    -> d.name AS "Department Name",
    -> e.salary AS "Monthly Salary",
    -> e.hiredate AS "Date Of Joining"
    -> FROM Employee AS e
    -> LEFT OUTER JOIN Department d
    -> on e.dept_id = d.id;
+---------------+-----------------+----------------+-----------------+
| Employee Name | Department Name | Monthly Salary | Date Of Joining |
+---------------+-----------------+----------------+-----------------+
| Nitin         | Development     |       25500.00 | 2022-02-17      |
| Pranshul      | Purchase        |       26000.00 | 2018-02-17      |
| Utkarsh       | Quality         |       27000.00 | 2019-03-13      |
| Arjun         | Marketing       |       28000.00 | 2019-07-17      |
| Rohit         | Development     |       15000.00 | 2020-02-27      |
| Shreya        | Purchase        |       25500.00 | 2020-08-27      |
```

53. What Are The Difference Between OLAP And OLTP?

S. No	OLAP (ONLINE ANALYTICAL PROCESSING	OLTP (ONLINE TRANSACTION PROCESSING)
1.	It Consists of historical data from various Databases.	It Consists only operational current data.
2.	It is subject oriented. Used for Data Mining, Analytics, Decision making,etc.	It is application oriented. Used for business tasks.
3.	The data is used in planning, problem solving and decision making.	The data is used to perform day to day fundamental operations.
4.	It reveals a snapshot of present business tasks.	It provides a multi-dimensional view of different business tasks.
5.	Large amount of data is stored typically in TB, PB	The size of the data is relatively small as the historical data is archived. For ex MB, GB
6.	Relatively slow as the amount of data involved is large. Queries may take hours.	Very Fast as the queries operate on 5% of the data.
7.	It only need backup from time to time as compared to OLTP	Backup and recovery process is maintained religiously
8.	This data is generally managed by CEO, MD, GM.	This data is managed by clerks, managers.
9.	Only read and rarely write operation.	Both read and write operations.

Source: https://www.geeksforgeeks.org/difference-between-olap-and-oltp-in-dbms/

Thanks! Hope It Helps You!

Watch The Answers For The Remaining Questions On My Youtube Channel.
Link For The Remaining Questions : https://youtu.be/pKFo8Mqp-cU

Connect with me:
Youtube: https://www.youtube.com/c/nitmantalks
Instagram: https://www.instagram.com/nitinmangotra/
LinkedIn: https://www.linkedin.com/in/nitin-mangotra-9a075a149/
Facebook: https://www.facebook.com/NitManTalks/
Twitter: https://twitter.com/nitinmangotra07/
Telegram: https://t.me/nitmantalks/

Do Connect With Me!!!!

Please Do Comment Your Feedback In Comment Section Of My Video On Youtube.
Here Is The Link: https://youtu.be/pKFo8Mqp-cU

When You Get Placed In Any Company Because Of My Video, DO Let Me Know.
It will Give Me More Satisfaction and Will Motivate me to make more such video Content!!

Thanks
PS: Don't Forget To Connect WIth ME.
Regards,
Nitin Mangotra (NitMan)

Connect with me:
Youtube: https://www.youtube.com/c/nitmantalks
Instagram: https://www.instagram.com/nitinmangotra/
LinkedIn: https://www.linkedin.com/in/nitin-mangotra-9a075a149/
Facebook: https://www.facebook.com/NitManTalks/
Twitter: https://twitter.com/nitinmangotra07/
Telegram: https://t.me/nitmantalks/