Practical No. 5

Government College of Engineering, Jalgaon **(An Autonomous Institute of Govt. of Maharashtra)**	
Name of Student:	
PRN:	**Batch:**
Date of Performance:	**Date of Completion:**
Subject: CO307U DBMSL	**Subject Teacher:- Mr. Vinit Kakde**

Aim : Create tables and perform the following

1. How the resulting salaries if every employee working on the 'Research' Departments is given a 10 percent raise.
2. Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department.
3. Retrieve the name of each employee Controlled by department number 5.
4. Retrieve the name of each dept and number of employees working in each department which has atleast 2 employee
5. Retrieve the name of employees who born in the year 1990's.
6. Retrieve the name of employees and their dept name.

Theory:

1. Data Manipulation Language (DML) Overview:

Data Manipulation Language (DML) commands in SQL are used to modify and retrieve data within a database. Common DML commands include:

- SELECT: Used to retrieve data from one or more tables.
- INSERT: Adds new data into tables.
- UPDATE: Modifies existing records.
- DELETE: Removes records from a table.

In complex queries, DML operations can be combined with additional features such as aggregation, conditions, sorting, and subqueries to perform more intricate tasks.

2. Aggregation Functions:

Aggregation functions in SQL allow you to perform calculations on a set of values, and return a single value. Common aggregation functions include:

- COUNT(): Returns the number of rows.
- SUM(): Adds the values of a numeric column.
- AVG(): Returns the average value.
- MIN() and MAX(): Return the smallest and largest values.

3. WHERE Clause:

The WHERE clause is used to filter records based on specific conditions. It defines which rows should be included in the result set before any grouping or aggregation is done.

4. HAVING Clause:

The HAVING clause is used to filter records after the GROUP BY clause. It is similar to the WHERE clause, but operates on aggregated data.

5. ORDER BY Clause:

The ORDER BY clause is used to sort the result set in ascending or descending order. By default, it sorts the data in ascending order.

6. Nested Queries (Subqueries):

A nested query, or subquery, is a query within another SQL query. It can be used in various clauses like SELECT, WHERE, or FROM.

7. EXISTS Statement:

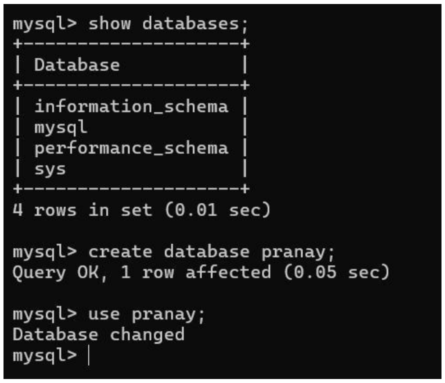
The EXISTS operator is used to check whether a subquery returns any result. It returns TRUE if the subquery returns one or more records, and FALSE if it doesn't.

8. Combining Multiple DML Features in Complex Queries:

In real-world scenarios, complex queries often combine multiple DML features, including aggregation functions, WHERE, HAVING, ORDER BY, nested subqueries, and EXISTS to retrieve or manipulate data efficiently.

Queries and Outputs:

1. Create database and use database

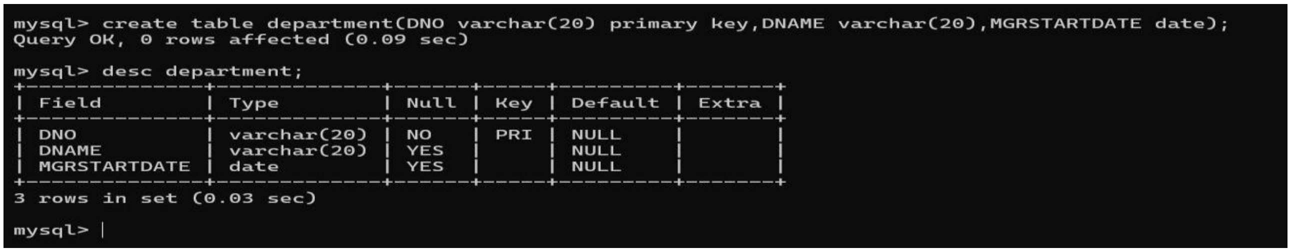


```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.01 sec)

mysql> create database pranay;
Query OK, 1 row affected (0.05 sec)

mysql> use pranay;
Database changed
mysql> |
```

2. Create Table and DESC

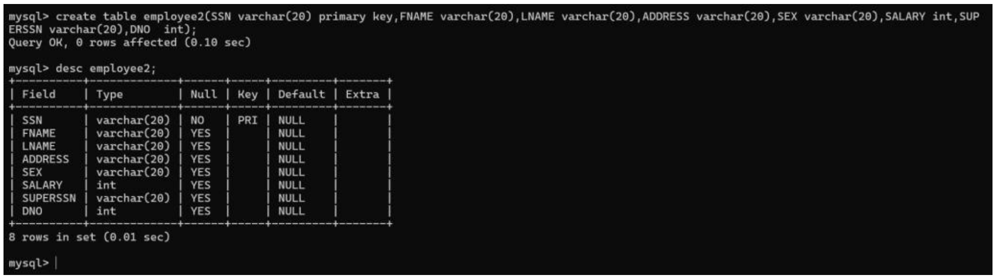
```
CREATE  TABLE  table_name  (

    column1_name datatype constraints,

    column2_name datatype constraints,

    ...

    columnN_name datatype constraints

);
```

create table department(DNO varchar(20) primary key,DNAME varchar(20),MGRSTARTDATE date);



```
mysql> create table department(DNO varchar(20) primary key,DNAME varchar(20),MGRSTARTDATE date);
Query OK, 0 rows affected (0.09 sec)

mysql> desc department;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| DNO          | varchar(20) | NO   | PRI | NULL    |       |
| DNAME        | varchar(20) | YES  |     | NULL    |       |
| MGRSTARTDATE | date        | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
3 rows in set (0.03 sec)

mysql> |
```

create table employee2(SSN varchar(20) primary key,FNAME varchar(20),LNAME varchar(20),ADDRESS varchar(20),SEX varchar(20),SALARY int,SUPERSSN varchar(20),DNO int);

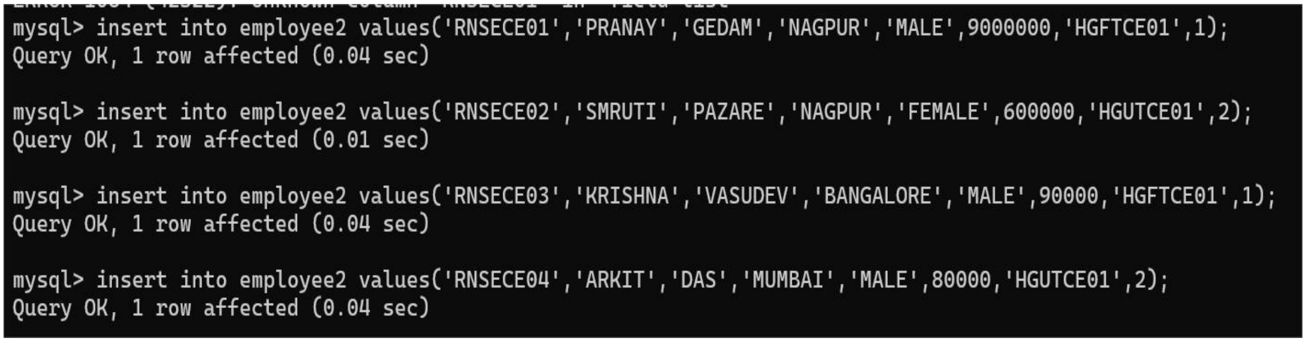


```
mysql> create table employee2(SSN varchar(20) primary key,FNAME varchar(20),LNAME varchar(20),ADDRESS varchar(20),SEX varchar(20),SALARY int,SUP
ERSSN varchar(20),DNO  int);
Query OK, 0 rows affected (0.10 sec)

mysql> desc employee2;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| SSN      | varchar(20) | NO   | PRI | NULL    |       |
| FNAME    | varchar(20) | YES  |     | NULL    |       |
| LNAME    | varchar(20) | YES  |     | NULL    |       |
| ADDRESS  | varchar(20) | YES  |     | NULL    |       |
| SEX      | varchar(20) | YES  |     | NULL    |       |
| SALARY   | int         | YES  |     | NULL    |       |
| SUPERSSN | varchar(20) | YES  |     | NULL    |       |
| DNO      | int         | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
8 rows in set (0.01 sec)

mysql> |
```

3. Insert into values

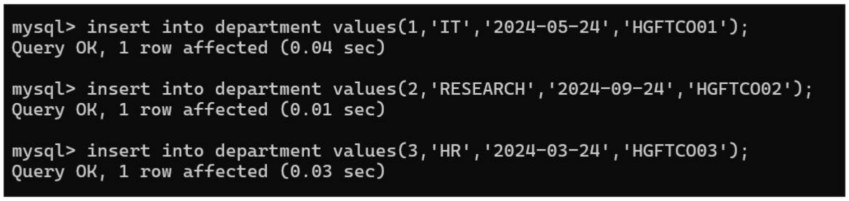
INSERT INTO table_name(column1,column2,column3….column_n)
VALUES (value1,value2,value3,…value_n);

i. insert into employee2 values('RNSECE01','PRANAY','GEDAM','NAGPUR','MALE',9000000,'HGFTCE01',1);
ii. insert into employee2 values('RNSECE02','SMRUTI','PAZARE','NAGPUR','FEMALE',600000,'HGUTCE01',2);
iii. insert into employee2 values('RNSECE04','ARKIT','DAS','MUMBAI','MALE',80000,'HGUTCE01',2);
iv. insert intoemployee2 values('RNSECE03','KRISHNA','VASUDEV','BANGALORE','MALE',90000,'HGFTCE01',1);
v. insert into employee2 values('RNSECE05','RAM','SHARMA','MYSORE','MALE',62000,'HGUTCE01',3);



```
ERROR 1054 (42S22): Unknown column 'RNSECE01' in 'field list'
mysql> insert into employee2 values('RNSECE01','PRANAY','GEDAM','NAGPUR','MALE',9000000,'HGFTCE01',1);
Query OK, 1 row affected (0.04 sec)

mysql> insert into employee2 values('RNSECE02','SMRUTI','PAZARE','NAGPUR','FEMALE',600000,'HGUTCE01',2);
Query OK, 1 row affected (0.01 sec)

mysql> insert into employee2 values('RNSECE03','KRISHNA','VASUDEV','BANGALORE','MALE',90000,'HGFTCE01',1);
Query OK, 1 row affected (0.04 sec)

mysql> insert into employee2 values('RNSECE04','ARKIT','DAS','MUMBAI','MALE',80000,'HGUTCE01',2);
Query OK, 1 row affected (0.04 sec)
```

vi. insert into department values(1,'IT','2024-05-24','HGFTCO01');
vii. insert into department values(2,'RESEARCH','2024-09-24','HGFTCO02');
viii. insert into department values(3,'HR','2024-03-24','HGFTCO03');



```
mysql> insert into department values(1,'IT','2024-05-24','HGFTCO01');
Query OK, 1 row affected (0.04 sec)

mysql> insert into department values(2,'RESEARCH','2024-09-24','HGFTCO02');
Query OK, 1 row affected (0.01 sec)

mysql> insert into department values(3,'HR','2024-03-24','HGFTCO03');
Query OK, 1 row affected (0.03 sec)
```

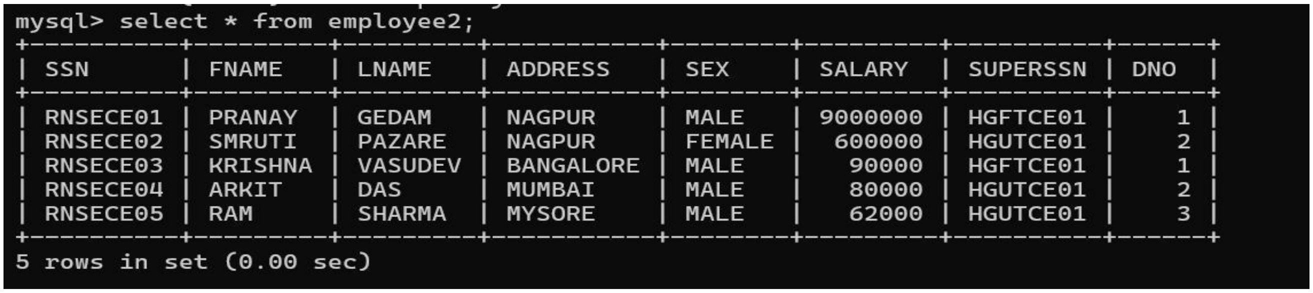
4. Select query

SELECT *column1*, *column2, ...*
FROM *table_name*;

SELECT * FROM

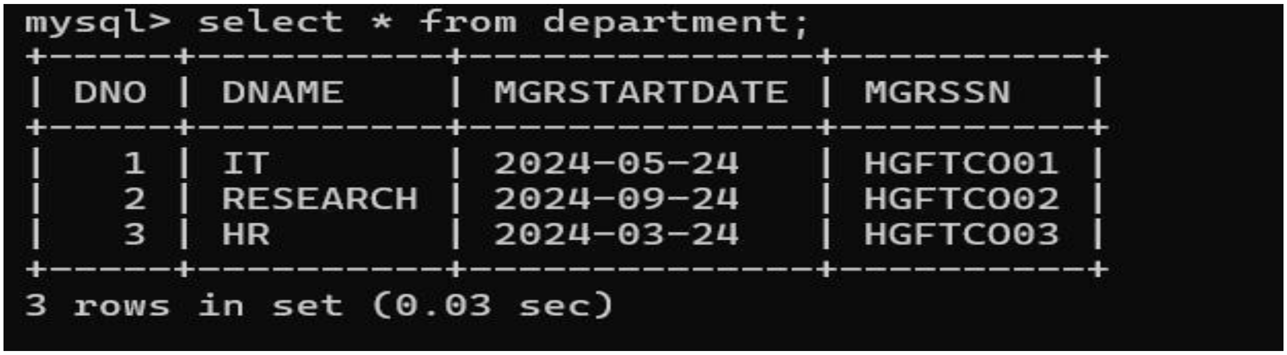
table_name; select * from

employee2;



```
mysql> select * from employee2;
+----------+---------+---------+-----------+--------+---------+----------+-----+
| SSN      | FNAME   | LNAME   | ADDRESS   | SEX    | SALARY  | SUPERSSN | DNO |
+----------+---------+---------+-----------+--------+---------+----------+-----+
| RNSECE01 | PRANAY  | GEDAM   | NAGPUR    | MALE   | 9000000 | HGFTCE01 |   1 |
| RNSECE02 | SMRUTI  | PAZARE  | NAGPUR    | FEMALE |  600000 | HGUTCE01 |   2 |
| RNSECE03 | KRISHNA | VASUDEV | BANGALORE | MALE   |   90000 | HGFTCE01 |   1 |
| RNSECE04 | ARKIT   | DAS     | MUMBAI    | MALE   |   80000 | HGUTCE01 |   2 |
| RNSECE05 | RAM     | SHARMA  | MYSORE    | MALE   |   62000 | HGUTCE01 |   3 |
+----------+---------+---------+-----------+--------+---------+----------+-----+
5 rows in set (0.00 sec)
```

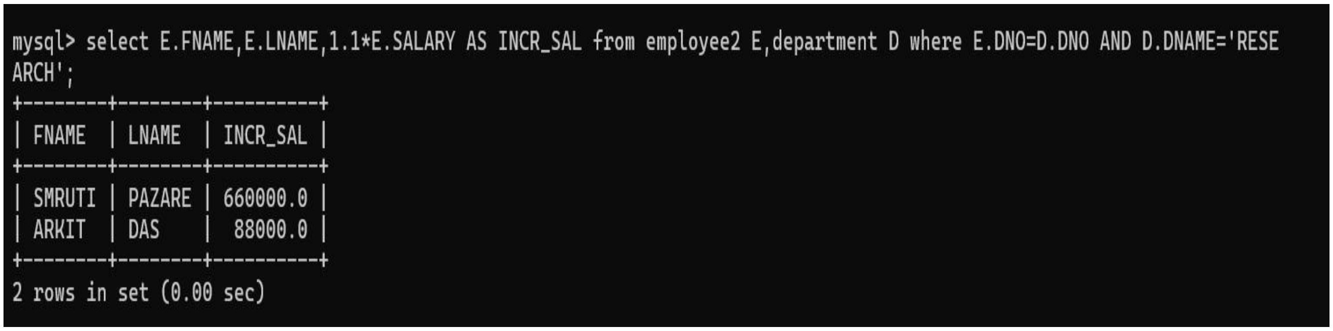
select * from department;



```
mysql> select * from department;
+-----+----------+--------------+----------+
| DNO | DNAME    | MGRSTARTDATE | MGRSSN   |
+-----+----------+--------------+----------+
|   1 | IT       | 2024-05-24   | HGFTCO01 |
|   2 | RESEARCH | 2024-09-24   | HGFTCO02 |
|   3 | HR       | 2024-03-24   | HGFTCO03 |
+-----+----------+--------------+----------+
3 rows in set (0.03 sec)
```

5. How the resulting salaries if every employee working on the "research" departments is given a 10 percent raise.

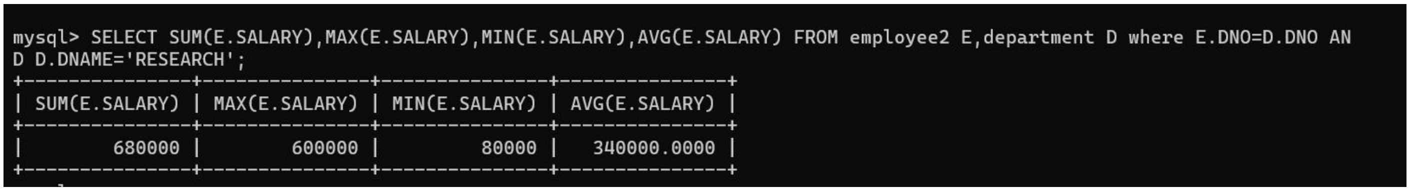
select E.FNAME,E.LNAME,1.1*E.SALARY AS INCR_SAL from employee2 E,department D where E.DNO=D.DNO AND D.DNAME='RESEARCH';



```
mysql> select E.FNAME,E.LNAME,1.1*E.SALARY AS INCR_SAL from employee2 E,department D where E.DNO=D.DNO AND D.DNAME='RESE
ARCH';
+--------+--------+----------+
| FNAME  | LNAME  | INCR_SAL |
+--------+--------+----------+
| SMRUTI | PAZARE | 660000.0 |
| ARKIT  | DAS    |  88000.0 |
+--------+--------+----------+
2 rows in set (0.00 sec)
```

6. **<u>Find the sum of salaries of all employees of all research department as well as maximum salary,the minimum and average salary.</u>**

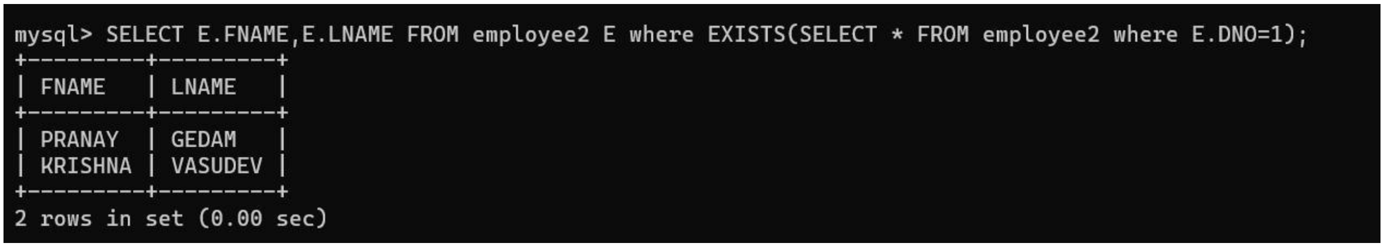
SELECTSUM(E.SALARY),MAX(E.SALARY),MIN(E.SALARY),AVG(E.SALARY)FROM
employee2 E,department D where E.DNO=D.DNO AN D D.DNAME='RESEARCH';



```
mysql> SELECT SUM(E.SALARY),MAX(E.SALARY),MIN(E.SALARY),AVG(E.SALARY) FROM employee2 E,department D where E.DNO=D.DNO AN
D D.DNAME='RESEARCH';
+---------------+---------------+---------------+---------------+
| SUM(E.SALARY) | MAX(E.SALARY) | MIN(E.SALARY) | AVG(E.SALARY) |
+---------------+---------------+---------------+---------------+
|        680000 |        600000 |         80000 |   340000.0000 |
+---------------+---------------+---------------+---------------+
```

7. **<u>Retrieve the name of employees controlled by department number 1.</u>**

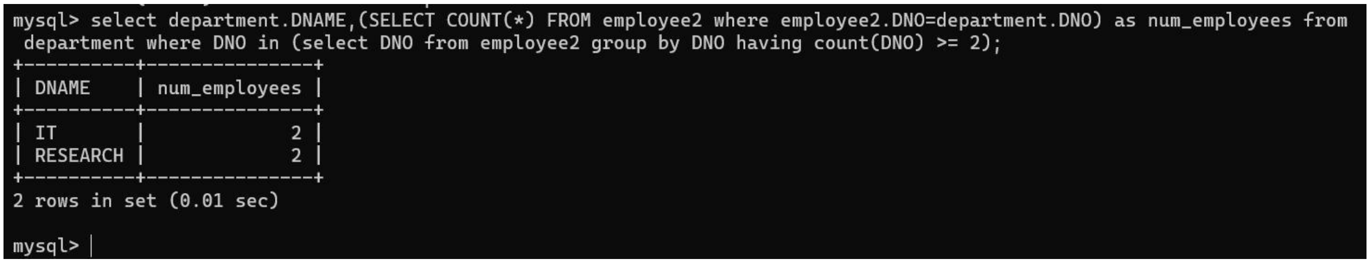
SELECT E.FNAME,E.LNAME FROM employee2 E where EXISTS(SELECT * FROM
employee2 where E.DNO=1);



```
mysql> SELECT E.FNAME,E.LNAME FROM employee2 E where EXISTS(SELECT * FROM employee2 where E.DNO=1);
+---------+---------+
| FNAME   | LNAME   |
+---------+---------+
| PRANAY  | GEDAM   |
| KRISHNA | VASUDEV |
+---------+---------+
2 rows in set (0.00 sec)
```

8. **<u>Retrieve the name of each dept and number of employees working in each department which has at least 2 employess.</u>**

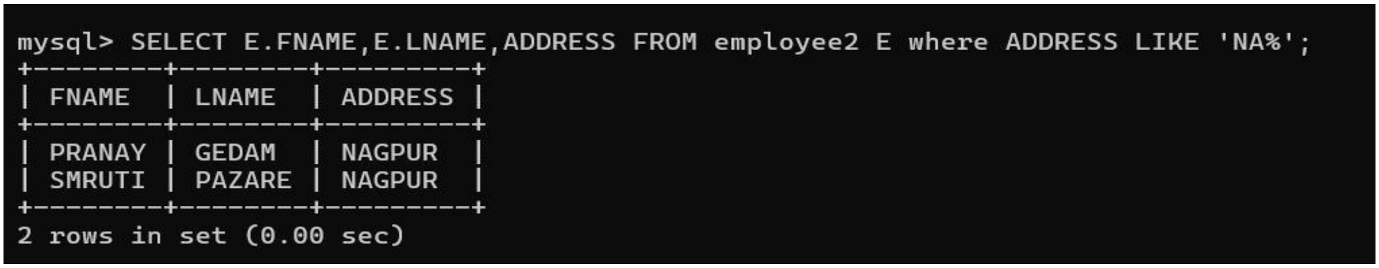
Select department.DNAME,(select count(*) from employee2 where employee2.DNO=department.DNO) as num_employees from department where DNO in (select DNO from employee2 group by DNO having count(DNO) >= 2);



```
mysql> select department.DNAME,(SELECT COUNT(*) FROM employee2 where employee2.DNO=department.DNO) as num_employees from
 department where DNO in (select DNO from employee2 group by DNO having count(DNO) >= 2);
+----------+---------------+
| DNAME    | num_employees |
+----------+---------------+
| IT       |             2 |
| RESEARCH |             2 |
+----------+---------------+
2 rows in set (0.01 sec)

mysql> |
```

9. Retrieve the name of employees who live in nagpur.

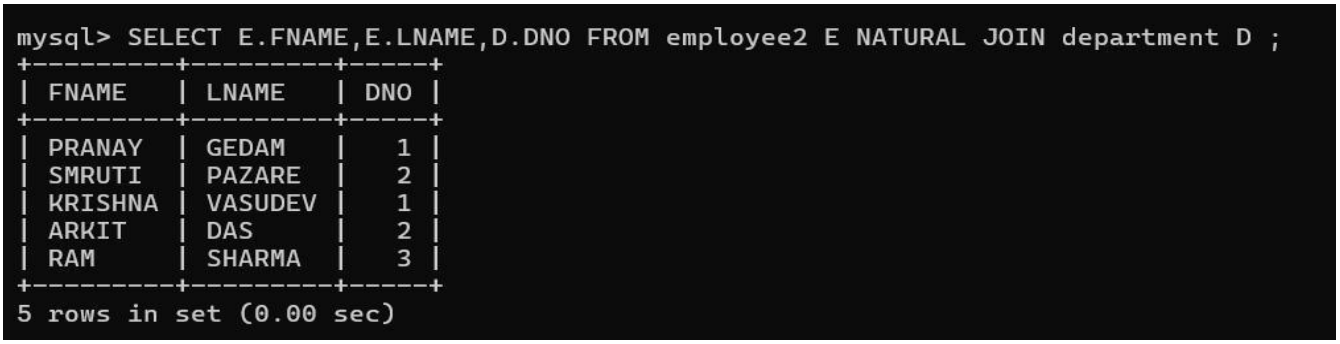
SELECT E.FNAME,E.LNAME,ADDRESS FROM employee2 E where ADDRESS LIKE 'NA%';



```
mysql> SELECT E.FNAME,E.LNAME,ADDRESS FROM employee2 E where ADDRESS LIKE 'NA%';
+--------+--------+---------+
| FNAME  | LNAME  | ADDRESS |
+--------+--------+---------+
| PRANAY | GEDAM  | NAGPUR  |
| SMRUTI | PAZARE | NAGPUR  |
+--------+--------+---------+
2 rows in set (0.00 sec)
```

10. Retrieve the name of the employess and their dept name(using join)

SELECT E.FNAME,E.LNAME,D.DNO FROM employee2 E NATURAL JOIN department D ;



```
mysql> SELECT E.FNAME,E.LNAME,D.DNO FROM employee2 E NATURAL JOIN department D ;
+---------+---------+-----+
| FNAME   | LNAME   | DNO |
+---------+---------+-----+
| PRANAY  | GEDAM   |   1 |
| SMRUTI  | PAZARE  |   2 |
| KRISHNA | VASUDEV |   1 |
| ARKIT   | DAS     |   2 |
| RAM     | SHARMA  |   3 |
+---------+---------+-----+
5 rows in set (0.00 sec)
```

Conclusion:-

Designing complex SQL queries using DML features requires an understanding of how to combine different functions, clauses, and subqueries effectively. Aggregation functions help in summarizing data, the WHERE and HAVING clauses filter records based on specific criteria, and nested queries or EXISTS enable complex conditional logic. Mastering these concepts will allow for efficient retrieval and manipulation of data in a relational database.

Questions:

a) **What is the purpose of the** HAVING **clause in SQL?**

It filters records after aggregation is applied, typically with GROUP BY.

b) **How does the** EXISTS **statement function in SQL?**

It returns TRUE if a subquery produces any rows, otherwise FALSE.

c) **What is the difference between** WHERE **and** HAVING**?**

WHERE filters rows before aggregation, while HAVING filters after aggregation.

d) **What is a nested query (subquery)?**

A query within another query used for complex filtering or calculations.

e) How is the ORDER BY **clause used in SQL?**

It sorts the result set in ascending or descending order based on specified columns.

Name & Sign of Course Teacher
Mr. Vinit Kakde