SQL Assignment – 1

Submitted by:

Name: Vanaja
Roll No: 22KQ1A6102
Batch: AIML
College Name: Pace Institute of Technology and Sciences

Subject: MY SQL

Submitted to:

Trainer's Name: _____Alekhya_____

TASK-1

emp_id	name	department	salary	Joining_date	city
1	Alice Smith	HR	45000	2020-02-15	New York
2	Bob Johnson	IT	60000	2019-08-23	Chicago
3	Carol White	IT	75000	2021-01-10	New York
4	David Brown	Finance	50000	2020-06-01	San Diego
5	Eva Adams	HR	47000	2018-04-12	Chicago
6	Frank Lee	Finance	65000	2021-11-05	New York
7	Grace Kim	IT	80000	2017-09-18	San Diego
8	Henry Clark	Marketing	55000	2019-12-29	Chicago
9	Irene Scott	Marketing	53000	2021-03-22	New York
10	Jack Davis	IT	72000	2020-08-14	Chicago

Sample Queries to Practice

1. Show all employees who work in the IT department.
2. Find employees whose salary is greater than 60,000.
3. Get all employees who joined after 1st Jan 2020.
4. Find the total salary paid to employees in each department.
5. Show the highest paid employee in the Finance department.
6. List employees who are from Chicago and earn more than 55,000.
7. Find the number of employees in each city
8. Show employees in the IT department ordered by salary in descending order
9. Display the average salary of all employees.
10. Get the details of the top 3 highest paid employees

Main Content – For Each Query

```
mysql> create database vanaja;
Query OK, 1 row affected (0.01 sec)

mysql> use vanaja;
Database changed
```

Query – 1: Create the Employee Table

```
mysql> create table employees(empid int primary key,name varchar(20),department
    -> varchar(20),salary decimal(10,2),joining_date date,city varchar(30));
Query OK, 0 rows affected (0.06 sec)

mysql> desc employees;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| empid        | int           | NO   | PRI | NULL    |       |
| name         | varchar(20)   | YES  |     | NULL    |       |
| department   | varchar(20)   | YES  |     | NULL    |       |
| salary       | decimal(10,2) | YES  |     | NULL    |       |
| joining_date | date          | YES  |     | NULL    |       |
| city         | varchar(30)   | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

Explanation:
This query creates a table named Employee with six columns:

- Emp_id → stores student roll number (unique for each employe).
- name → stores Employee's name.
- department → stores Employee's department
- Salary→ stores Employee's salary
- Joining_date→stores to Employee's starting date
- city→ To stores Employee's city

Output:
Table created successfully.

Insert into all employee's data

1. mysql> insert into employees values(1,'Alice Smith','HR',45000,'2020-02-15','New York');

Query OK, 1 row affected (0.02 sec)

2.mysql> insert into employees values(2,'Bob Johnson','IT',60000,'2019-08-23','Chicago');

Query OK, 1 row affected (0.02 sec)

3.mysql> insert into employees values(3,'Carol White','IT',75000,'2021-08-10','New York');

Query OK, 1 row affected (0.01 sec)

4.mysql> insert into employees values(4,'David Brown','Finance',50000,'2020-06-01','San Diego');

Query OK, 1 row affected (0.01 sec)

5.mysql> insert into employees values(5,'Eva Adams','HR',47000,'2018-04-12','Chicago');

Query OK, 1 row affected (0.01 sec)

6.mysql> insert into employees values(6,'Frank Lee','HR',65000,'2021-11-05','New York');

Query OK, 1 row affected (0.01 sec)

7.mysql> insert into employees values(7,'Grace Kim','IT',80000,'2017-09-18','San Diego');

Query OK, 1 row affected (0.01 sec)

8.mysql> insert into employees values(8,'Henry Clark','Marketing',55000,'2019-12-29','Chicago');

Query OK, 1 row affected (0.01 sec)

9.mysql> insert into employees values(9,'Irene Scott','Marketing',53000,'2021-03-22','New York');

Query OK, 1 row affected (0.01 sec)

10.mysql> insert into employees values(10,'Jack Davis','IT',72000,'2020-08-14','Chicago');

Query OK, 1 row affected (0.02 sec)

Select Employee's data

mysql> select * from employees;

```
+-------+-------------+------------+----------+--------------+-----------+
| empid | name        | department | salary   | joining_date | city      |
+-------+-------------+------------+----------+--------------+-----------+
|     1 | Alice Smith | HR         | 45000.00 | 2020-02-15   | New York  |
|     2 | Bob Johnson | IT         | 60000.00 | 2019-08-23   | Chicago   |
|     3 | Carol White | IT         | 75000.00 | 2021-08-10   | New York  |
|     4 | David Brown | Finance    | 50000.00 | 2020-06-01   | San Diego |
|     5 | Eva Adams   | HR         | 47000.00 | 2018-04-12   | Chicago   |
|     6 | Frank Lee   | HR         | 65000.00 | 2021-11-05   | New York  |
|     7 | Grace Kim   | IT         | 80000.00 | 2017-09-18   | San Diego |
|     8 | Henry Clark | Marketing  | 55000.00 | 2019-12-29   | Chicago   |
|     9 | Irene Scott | Marketing  | 53000.00 | 2021-03-22   | New York  |
|    10 | Jack Davis  | IT         | 72000.00 | 2020-08-14   | Chicago   |
```

+-------+-------------+------------+----------+--------------+-----------+

10 rows in set (0.01 sec)

SQL QUERIES+ EXPLANATION + OUTPUT

Query title1: Show all employees who work in the IT department

SQL QUERY: mysql> SELECT * FROM employees

-> WHERE department = 'IT';

EXPLANATION: This query uses a **WHERE condition** to filter only those employees whose department is "IT".

OUTPUT:+-------+-------------+------------+----------+--------------+-----------+

| empid | name | department | salary | joining_date | city |

+-------+-------------+------------+----------+--------------+-----------+

| 2 | Bob Johnson | IT | 60000.00 | 2019-08-23 | Chicago |

| 3 | Carol White | IT | 75000.00 | 2021-08-10 | New York |

| 7 | Grace Kim | IT | 80000.00 | 2017-09-18 | San Diego |

| 10 | Jack Davis | IT | 72000.00 | 2020-08-14 | Chicago |

+-------+-------------+------------+----------+--------------+-----------+

4 rows in set (0.00 sec)

Query title 2: Find employees whose salary is greater than 60,000

SQL QUERY: mysql> SELECT * FROM employees

-> WHERE salary > 60000;

EXPLANATION: Here we apply a salary condition (> 60000) to get only high-paid employees.

OUTPUT: +-------+-------------+------------+----------+--------------+-----------+

| empid | name | department | salary | joining_date | city |

+-------+-------------+------------+----------+--------------+-----------+

| 3 | Carol White | IT | 75000.00 | 2021-08-10 | New York |

| 6 | Frank Lee | HR | 65000.00 | 2021-11-05 | New York |

| 7 | Grace Kim | IT | 80000.00 | 2017-09-18 | San Diego |

| 10 | Jack Davis | IT | 72000.00 | 2020-08-14 | Chicago |

+-------+-------------+------------+----------+--------------+-----------+

4 rows in set (0.01 sec)

Query title 3: Get all employees who joined after 1st Jan 2020

SQL QUERY: mysql> SELECT * FROM employees

-> WHERE joining_date > '2020-01-01';

EXPLANATION: We use the Joining_date column and check with > operator (> '2020-01-01').

OUTPUT: +-------+-------------+------------+----------+--------------+-----------+

| empid | name | department | salary | joining_date | city |

+-------+-------------+------------+----------+--------------+-----------+

| 1 | Alice Smith | HR | 45000.00 | 2020-02-15 | New York |

| 3 | Carol White | IT | 75000.00 | 2021-08-10 | New York |

| 4 | David Brown | Finance | 50000.00 | 2020-06-01 | San Diego |

| 6 | Frank Lee | HR | 65000.00 | 2021-11-05 | New York |

| 9 | Irene Scott | Marketing | 53000.00 | 2021-03-22 | New York |

| 10 | Jack Davis | IT | 72000.00 | 2020-08-14 | Chicago |

+-------+-------------+------------+----------+--------------+-----------+

6 rows in set (0.00 sec)

Query title 4: Find the total salary paid to employees in each department

SQL QUERY: mysql> SELECT department, SUM(salary) AS total_salary

-> FROM employees

-> GROUP BY department**;**

EXPLANATION: Here we use GROUP BY department and SUM(salary) to calculate department-wise salary totals**.**

OUTPUT: +------------+--------------+

| department | total_salary |

+------------+--------------+

| HR | 157000.00 |

| IT | 287000.00 |

| Finance | 50000.00 |

| Marketing | 108000.00 |

+------------+--------------+

4 rows in set (0.01 sec)

Query title 5: Show the highest paid employee in the Finance department

SQL QUERY: mysql> SELECT * FROM employees

-> WHERE department = 'Finance'

-> ORDER BY salary DESC

-> LIMIT 1;

EXPLANATION: We filter department = "Finance" and use MAX(salary) to find the top salary.

OUTPUT: +-------+-------------+------------+----------+--------------+-----------+

| empid | name | department | salary | joining_date | city |

+-------+-------------+------------+----------+--------------+-----------+

| 4 | David Brown | Finance | 50000.00 | 2020-06-01 | San Diego |

+-------+-------------+------------+----------+--------------+-----------+

1 row in set (0.00 sec)

Query title 6: List employees who are from Chicago and earn more than 55,000

SQL QUERY: mysql> SELECT * FROM employees

-> WHERE city = 'Chicago' AND salary > 55000;

EXPLANATION: Combine two conditions using AND:city = 'Chicago' and salary>55000.

OUTPUT:

```
+-------+-------------+------------+----------+--------------+---------+
| empid | name        | department | salary   | joining_date | city    |
+-------+-------------+------------+----------+--------------+---------+
|     2 | Bob Johnson | IT         | 60000.00 | 2019-08-23   | Chicago |
|    10 | Jack Davis  | IT         | 72000.00 | 2020-08-14   | Chicago |
+-------+-------------+------------+----------+--------------+---------+
```

2 rows in set (0.00 sec)

Query title 7: Find the number of employees in each city.

SQL QUERY: mysql> SELECT city, COUNT(*) AS total_employees

-> FROM employees

-> GROUP BY city;

EXPLANATION: Use GROUP BY city with COUNT(*) to count employees city-wise.

OUTPUT:

```
+-----------+-----------------+
| city      | total_employees |
+-----------+-----------------+
| New York  |               4 |
| Chicago   |               4 |
| San Diego |               2 |
+-----------+-----------------+
```

3 rows in set (0.00 sec)

Query title 8: Show employees in the IT department ordered by salary in descending order

SQL OUERY: mysql> SELECT * FROM employees

-> WHERE department = 'IT'

-> ORDER BY salary DESC;

EXPLANATION: Filter with WHERE department = 'IT' and sort using ORDER BY salary DESC.

OUTPUT:

```
+-------+-------------+------------+----------+--------------+-----------+
| empid | name        | department | salary   | joining_date | city      |
+-------+-------------+------------+----------+--------------+-----------+
|     7 | Grace Kim   | IT         | 80000.00 | 2017-09-18   | San Diego |
|     3 | Carol White | IT         | 75000.00 | 2021-08-10   | New York  |
|    10 | Jack Davis  | IT         | 72000.00 | 2020-08-14   | Chicago   |
|     2 | Bob Johnson | IT         | 60000.00 | 2019-08-23   | Chicago   |
+-------+-------------+------------+----------+--------------+-----------+
4 rows in set (0.00 sec)
```

Query title 9: Display the average salary of all employees.

SQLQUERY: mysql> SELECT AVG(salary) AS avg_salary

-> FROM employees;

EXPLANATION: Use **AVG(salary)** function to find the overall average.

(Sum of all salaries = 607000 ÷ 10 employees = 60700)

OUTPUT:

```
+--------------+
| avg_salary   |
+--------------+
```

| 60200.000000 |

+--------------+

1 row in set (0.00 sec)

Query title 10: Get the details of the top 3 highest paid employees.

SQL QUERY: mysql> SELECT * FROM employees

-> ORDER BY salary DESC

-> LIMIT 3;

EXPLANATION: Order employees by salary DESC and use LIMIT 3 (or TOP 3 in SQL Server).

OUTPUT:

mysql> SELECT * FROM employees

-> ORDER BY salary DESC

-> LIMIT 3;

+-------+-------------+------------+----------+--------------+-----------+

| empid | name | department | salary | joining_date | city |

+-------+-------------+------------+----------+--------------+-----------+

| 7 | Grace Kim | IT | 80000.00 | 2017-09-18 | San Diego |

| 3 | Carol White | IT | 75000.00 | 2021-08-10 | New York |

| 10 | Jack Davis | IT | 72000.00 | 2020-08-14 | Chicago |

+-------+-------------+------------+----------+--------------+-----------+

3 rows in set (0.00 sec).