

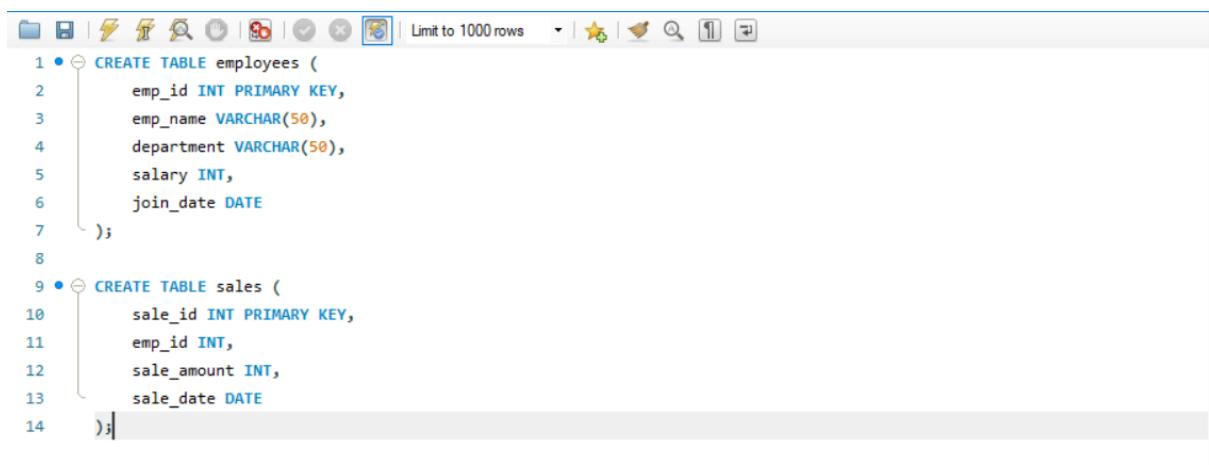
CODTECH Internship

TASK 2

Data Analysis With Complex Queries

NAME: S.Arun Ganesh

1.Table Creation



The screenshot shows a MySQL Workbench interface with two SQL queries in the editor:

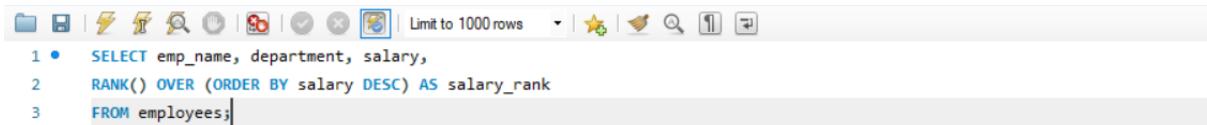
```
1 • Ⓜ CREATE TABLE employees (
2     emp_id INT PRIMARY KEY,
3     emp_name VARCHAR(50),
4     department VARCHAR(50),
5     salary INT,
6     join_date DATE
7 );
8
9 • Ⓜ CREATE TABLE sales (
10    sale_id INT PRIMARY KEY,
11    emp_id INT,
12    sale_amount INT,
13    sale_date DATE
14 );
```

2. Insertion values

```
1 • INSERT INTO employees VALUES  
2     (1,'Arun','IT',50000,'2022-01-10'),  
3     (2,'Priya','HR',45000,'2021-03-15'),  
4     (3,'Rahul','IT',60000,'2020-07-21'),  
5     (4,'Sneha','Sales',40000,'2022-06-01');  
6  
7 • INSERT INTO sales VALUES  
8     (101,1,20000,'2024-01-10'),  
9     (102,1,15000,'2024-02-12'),  
10    (103,3,30000,'2024-01-05'),  
11    (104,4,18000,'2024-02-20');
```

3.Window Function

Query:

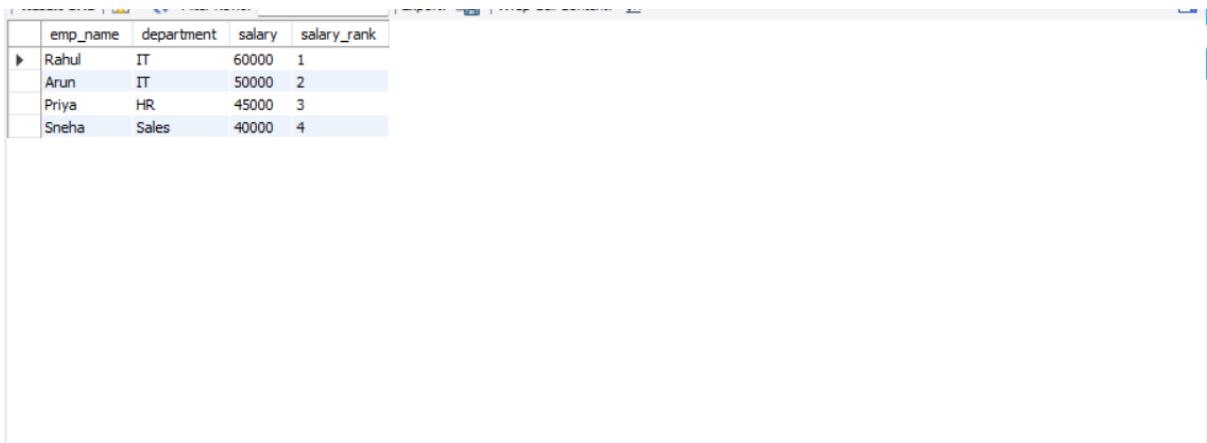


The screenshot shows a MySQL query editor interface. The query window contains the following SQL code:

```
1 •  SELECT emp_name, department, salary,
2      RANK() OVER (ORDER BY salary DESC) AS salary_rank
3  FROM employees;
```

The interface includes standard database navigation buttons (File, Edit, View, etc.) and a toolbar with icons for search, refresh, and other functions. A status bar at the bottom indicates "Limit to 1000 rows".

Output:

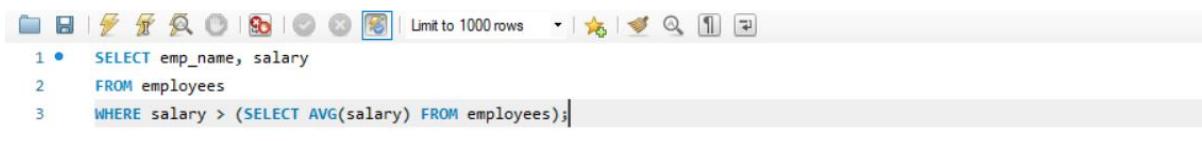


The screenshot shows the results of the executed query. The output is a table with four columns: emp_name, department, salary, and salary_rank. The data is as follows:

| | emp_name | department | salary | salary_rank |
|---|----------|------------|--------|-------------|
| ▶ | Rahul | IT | 60000 | 1 |
| | Arun | IT | 50000 | 2 |
| | Priya | HR | 45000 | 3 |
| | Sneha | Sales | 40000 | 4 |

4. Subquery

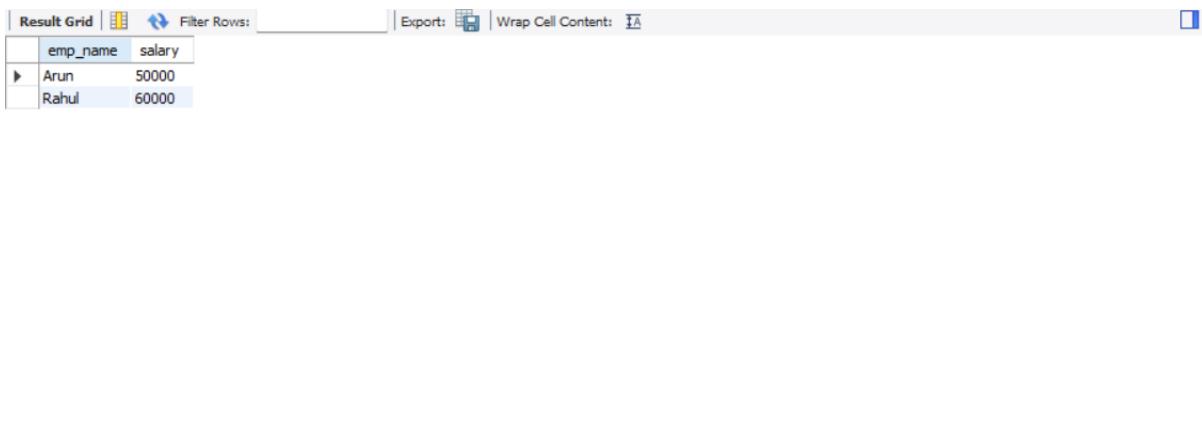
Query:



A screenshot of a MySQL query editor. The toolbar at the top includes icons for file operations, search, and help. A dropdown menu says "Limit to 1000 rows". The query window contains the following code:

```
1 •  SELECT emp_name, salary
2   FROM employees
3 WHERE salary > (SELECT AVG(salary) FROM employees);
```

Output:

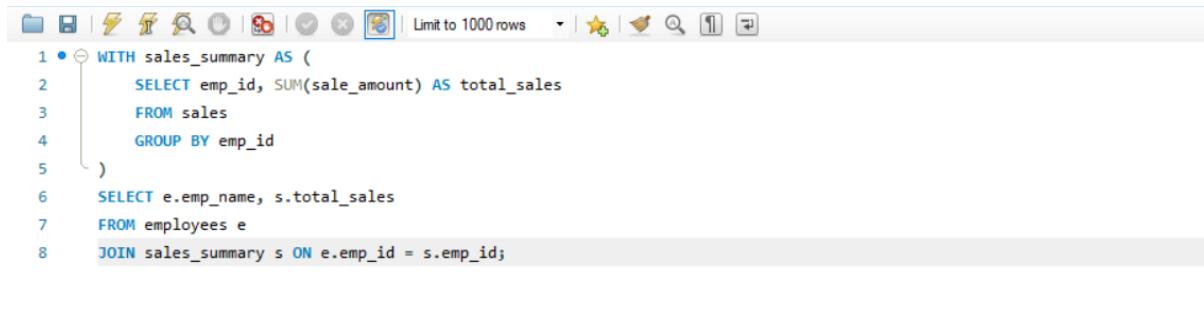


A screenshot of a MySQL query editor showing the results of the executed query. The results are displayed in a grid format:

| emp_name | salary |
|----------|--------|
| Arun | 50000 |
| Rahul | 60000 |

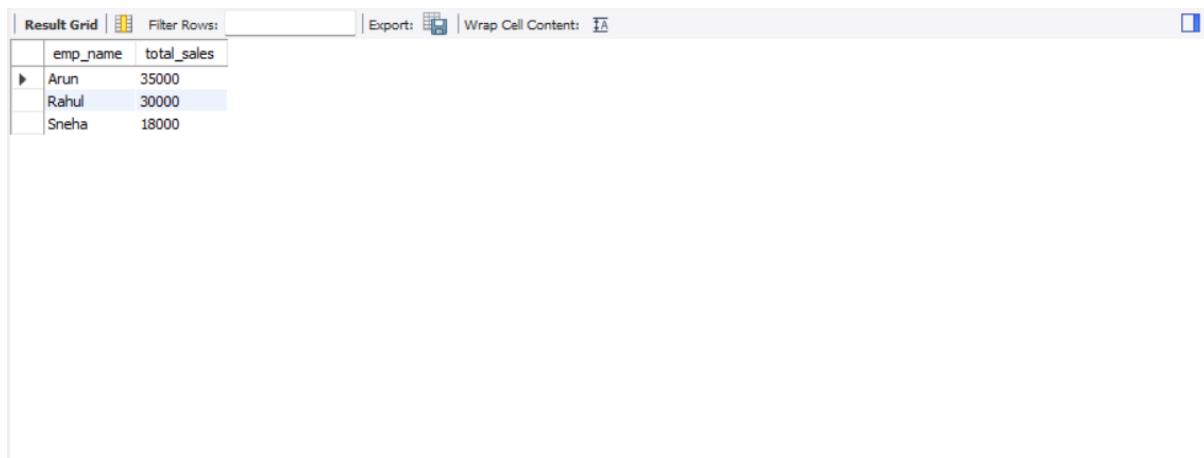
5.CTE (Common Table Expression)

Query:



```
1 • Ⓜ WITH sales_summary AS (
2     SELECT emp_id, SUM(sale_amount) AS total_sales
3     FROM sales
4     GROUP BY emp_id
5 )
6     SELECT e.emp_name, s.total_sales
7     FROM employees e
8     JOIN sales_summary s ON e.emp_id = s.emp_id;
```

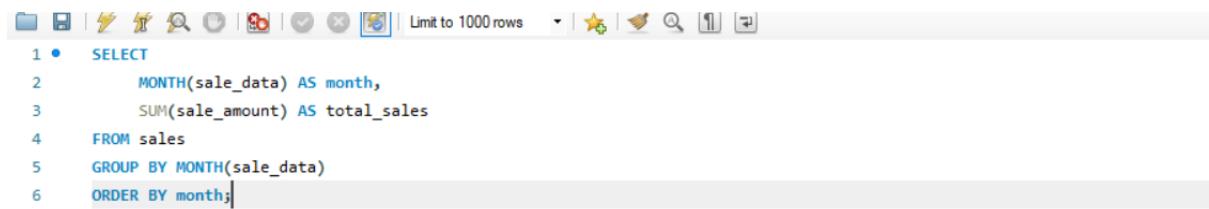
Output:



| emp_name | total_sales |
|----------|-------------|
| Arun | 35000 |
| Rahul | 30000 |
| Sneha | 18000 |

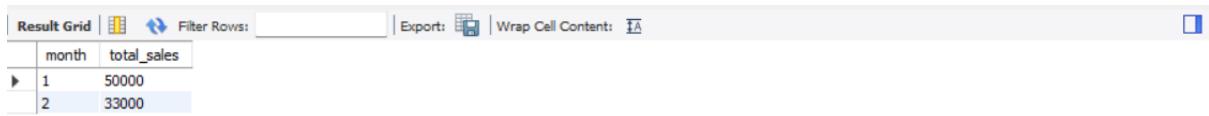
6. Trend or Pattern Analysis

Query:



```
1 •   SELECT
2       MONTH(sale_data) AS month,
3       SUM(sale_amount) AS total_sales
4   FROM sales
5   GROUP BY MONTH(sale_data)
6   ORDER BY month;
```

Output:



| month | total_sales |
|-------|-------------|
| 1 | 50000 |
| 2 | 33000 |