

SQL

Content:-

max

count

having

inner join

left join

join

union

case

Employee Table

<u>id</u>	<u>name</u>	<u>salary</u>	<u>department</u>	<u>branch</u>
<u>1</u>	<u>Alice</u>	<u>60000</u>	<u>HR</u>	<u>1</u>
<u>2</u>	<u>Bob</u>	<u>40000</u>	<u>IT</u>	<u>2</u>
<u>3</u>	<u>Charlie</u>	<u>30000</u>	<u>IT</u>	<u>2</u>
<u>4</u>	<u>Diana</u>	<u>70000</u>	<u>HR</u>	<u>1</u>
<u>5</u>	<u>Edward</u>	<u>20000</u>	<u>Sales</u>	<u>NULL</u>

Branch Table

<u>ID</u>	<u>Branch_Name</u>
<u>1</u>	<u>New York</u>
<u>2</u>	<u>San Francisco</u>

1. MAX

Definition: The **MAX** function returns the highest value in a column.

Query: Find the employee with the highest salary.

```
SELECT MAX(salary) AS highest_salary  
FROM employee;
```

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL query:

```
1 • SELECT MAX(salary) AS highest_salary  
2 FROM employee;  
3
```

The query results are displayed in the Result Grid, showing a single row with the value 70000 for the column highest_salary.

The Output tab shows the execution log with the following entries:

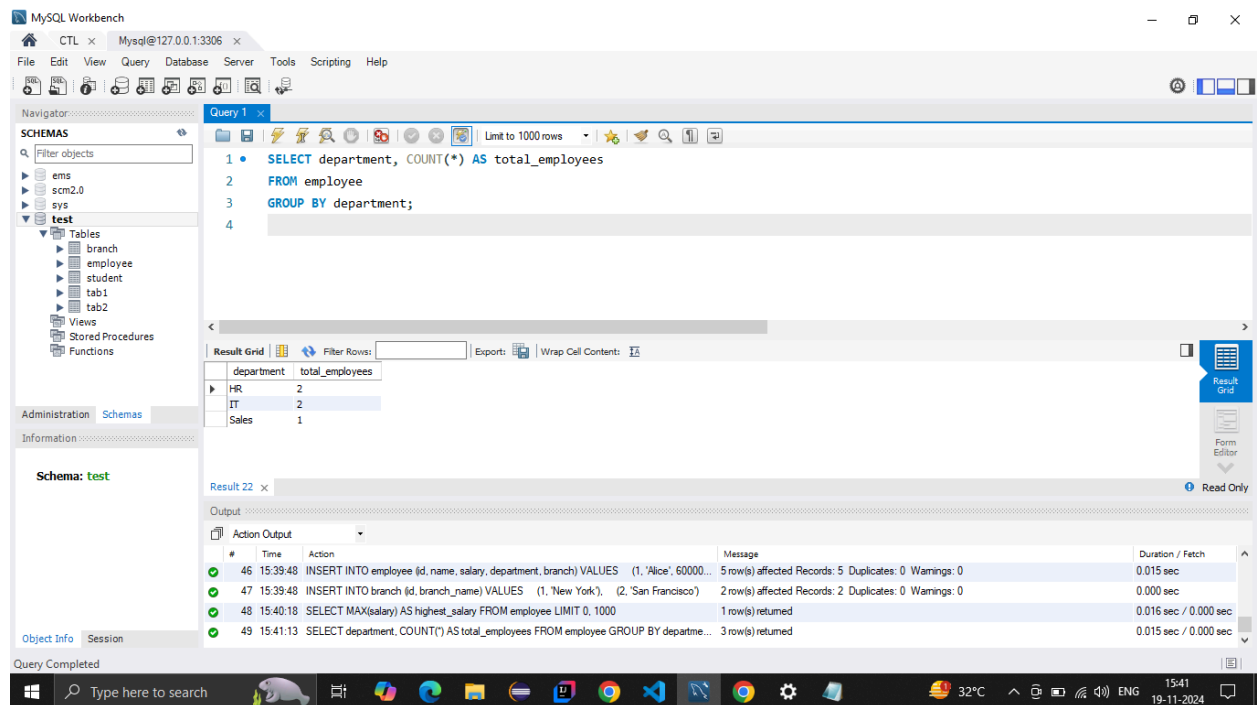
#	Time	Action	Message	Duration / Fetch
45	15:39:22	CREATE TABLE branch (id INT PRIMARY KEY, branch_name VARCHAR(50))	0 row(s) affected	0.046 sec
46	15:39:48	INSERT INTO employee (id, name, salary, department, branch) VALUES (1, 'Alice', 60000, ...)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
47	15:39:48	INSERT INTO branch (id, branch_name) VALUES (1, 'New York'), (2, 'San Francisco')	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
48	15:40:18	SELECT MAX(salary) AS highest_salary FROM employee LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

2. COUNT

Definition: The **COUNT** function calculates the number of rows that match a condition.

Query: Count the total number of employees in each department.

```
SELECT department, COUNT(*) AS total_employees
FROM employee
GROUP BY department;
```



The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying the following SQL query:

```
1 • SELECT department, COUNT(*) AS total_employees
2 FROM employee
3 GROUP BY department;
```

The 'Result Grid' shows the output of the query:

department	total_employees
HR	2
IT	2
Sales	1

The 'Output' tab shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
46	15:39:48	INSERT INTO employee (id, name, salary, department, branch) VALUES (1, 'Alice', 60000...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
47	15:39:48	INSERT INTO branch (id, branch_name) VALUES (1, 'New York'), (2, 'San Francisco')	2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0	0.000 sec
48	15:40:18	SELECT MAX(salary) AS highest_salary FROM employee LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
49	15:41:13	SELECT department, COUNT(*) AS total_employees FROM employee GROUP BY departme...	3 row(s) returned	0.015 sec / 0.000 sec

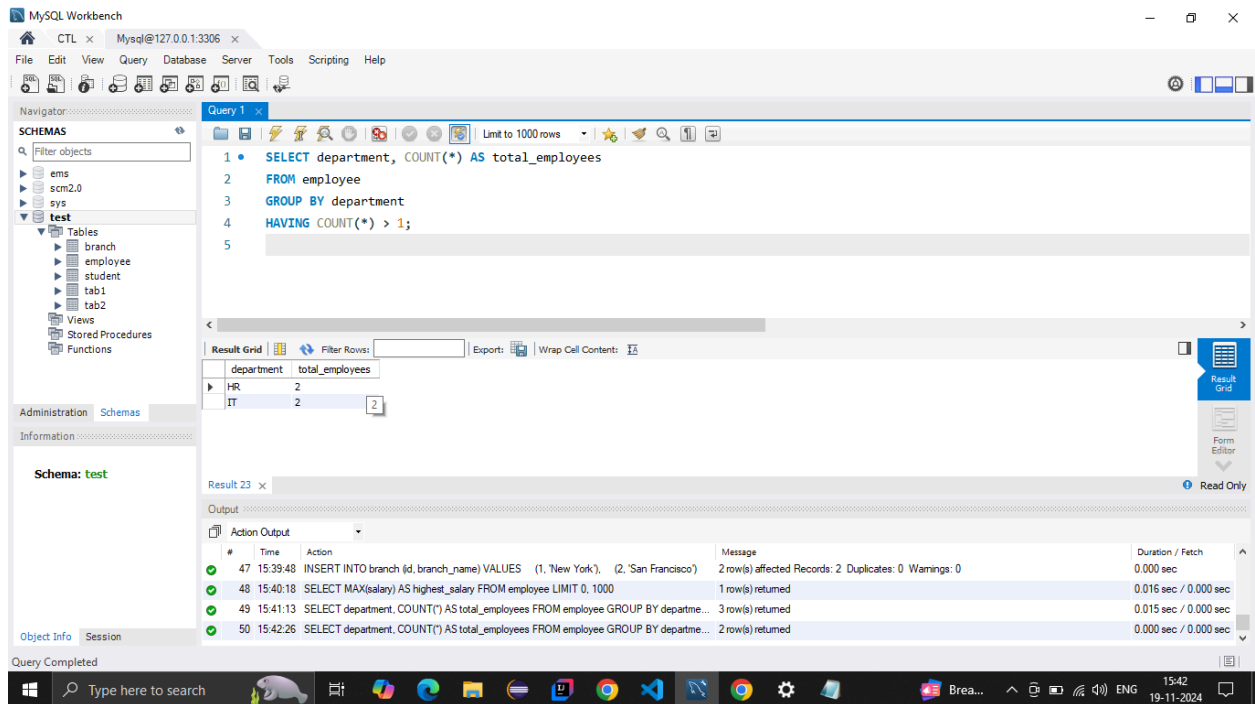
3. HAVING

Definition: The **HAVING** clause filters groups after aggregation (unlike **WHERE**, which filters rows).

Query: List departments with more than 5 employees.

```
SELECT department, COUNT(*) AS total_employees
```

```
FROM employee
GROUP BY department
HAVING COUNT(*) > 5;
```

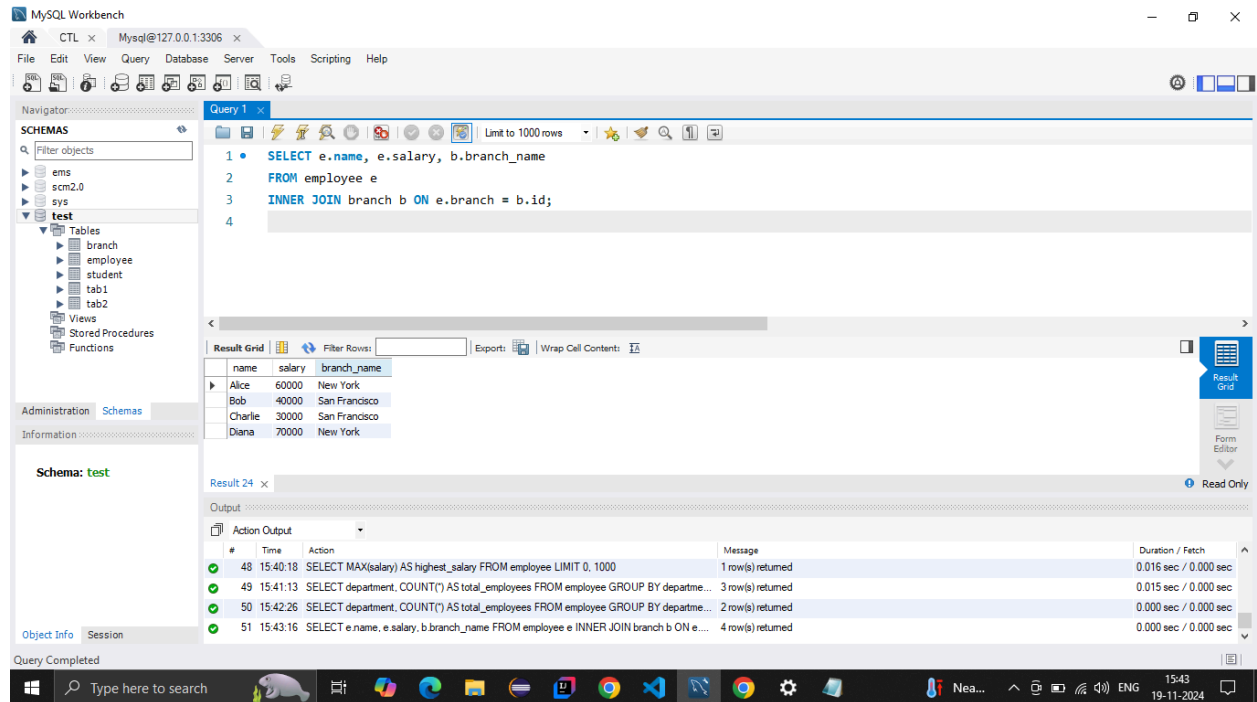


4. INNER JOIN

Definition: Combines rows from two tables where there is a match in the specified columns.

Query: Get employees' salaries along with their branch information from a **branch** table.

```
SELECT e.name, e.salary, b.branch_name
FROM employee e
INNER JOIN branch b ON e.branch = b.id;
```

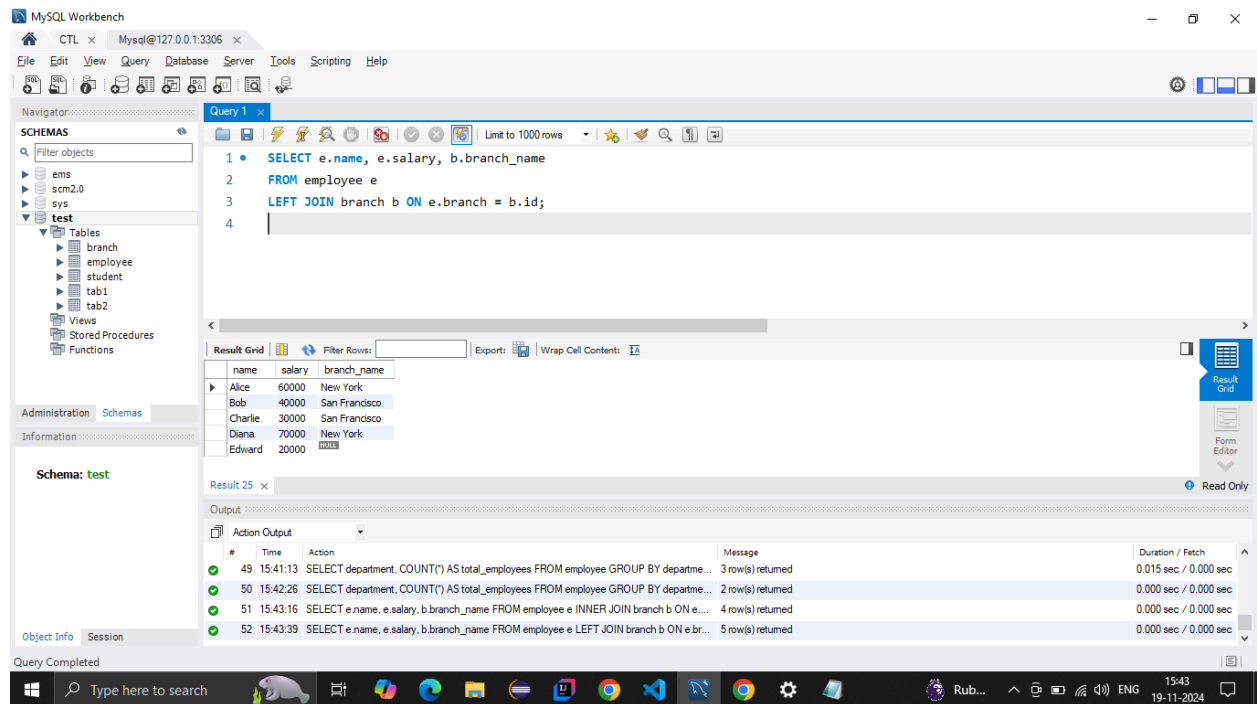


5. LEFT JOIN

Definition: Returns all rows from the left table and the matching rows from the right table. If no match, **NULL** is returned for the right table's columns.

Query: Get all employees and their branch information, including employees not assigned to any branch.

```
SELECT e.name, e.salary, b.branch_name
FROM employee e
LEFT JOIN branch b ON e.branch = b.id;
```



6. JOIN

Definition: A general term referring to the combination of rows from two or more tables. SQL supports multiple types of joins like **INNER JOIN**, **LEFT JOIN**, **RIGHT JOIN**, etc.

7. UNION

Definition: Combines the results of two or more **SELECT** queries into a single result set. Duplicate rows are removed unless **UNION ALL** is used.

Query: Combine employees earning above 50,000 with those in the "HR" department.

SELECT name, salary, department

FROM employee

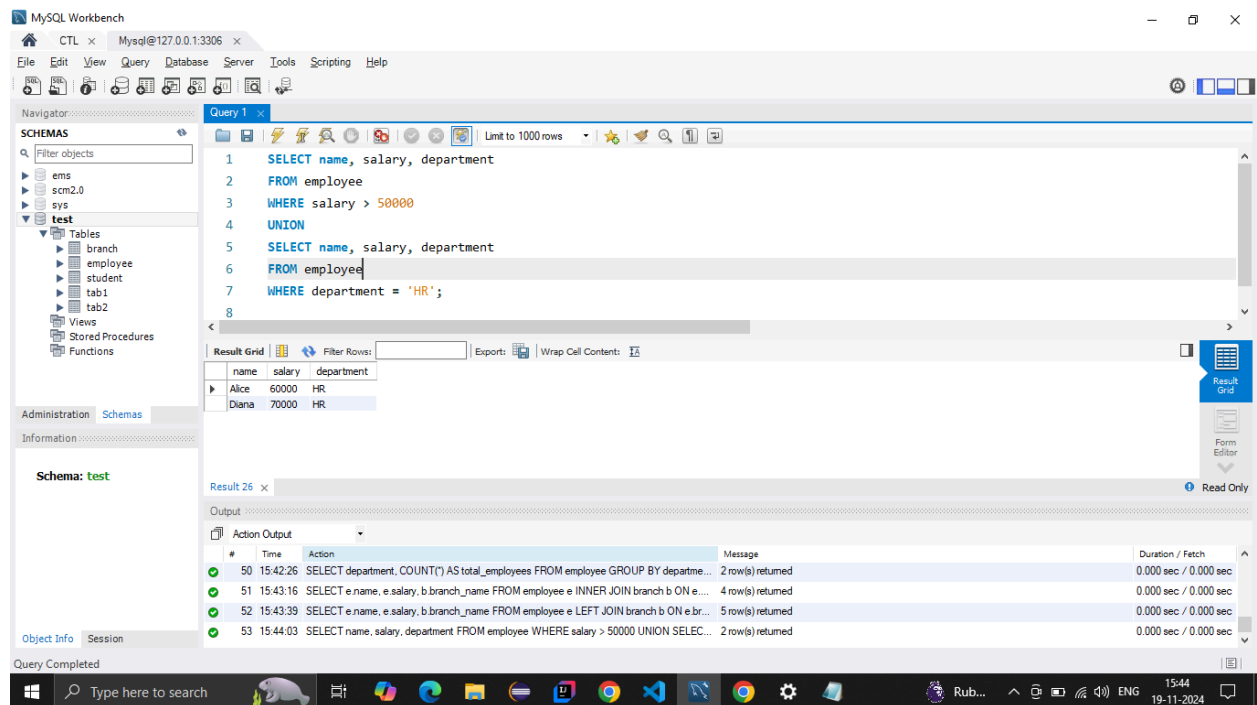
WHERE salary > 50000

UNION

SELECT name, salary, department

FROM employee

WHERE department = 'HR';



8. CASE

Definition: Provides conditional logic to return different values based on specific conditions.

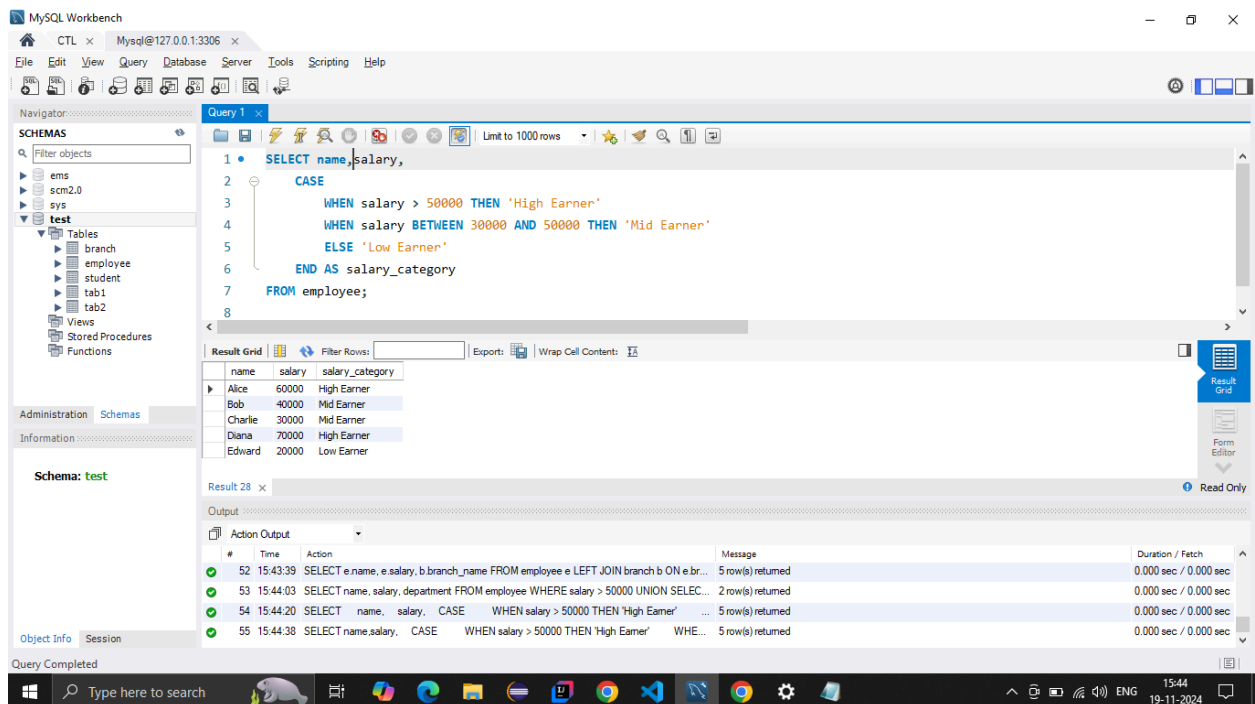
Query: Categorize employees based on their salary.

SELECT

```

name,
salary,
CASE
    WHEN salary > 50000 THEN 'High Earner'
    WHEN salary BETWEEN 30000 AND 50000 THEN 'Mid
Earner'
    ELSE 'Low Earner'
END AS salary_category
FROM employee;

```



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```

1 SELECT name,salary,
2 CASE
3     WHEN salary > 50000 THEN 'High Earner'
4     WHEN salary BETWEEN 30000 AND 50000 THEN 'Mid Earner'
5     ELSE 'Low Earner'
6 END AS salary_category
7 FROM employee;
8

```

The Results window displays the output of the query:

	name	salary	salary_category
▶	Alice	60000	High Earner
	Bob	40000	Mid Earner
	Charlie	30000	Mid Earner
	Diana	70000	High Earner
	Edward	20000	Low Earner

The bottom panel shows the Output window with the following log entries:

#	Time	Action	Message	Duration / Fetch
52	15:43:39	SELECT e.name, e.salary, b.branch_name FROM employee e LEFT JOIN branch b ON e.br...	5 row(s) returned	0.000 sec / 0.000 sec
53	15:44:03	SELECT name, salary, department FROM employee WHERE salary > 50000 UNION SELEC...	2 row(s) returned	0.000 sec / 0.000 sec
54	15:44:20	SELECT name, salary, CASE WHEN salary > 50000 THEN 'High Earner' ...	5 row(s) returned	0.000 sec / 0.000 sec
55	15:44:38	SELECT name,salary, CASE WHEN salary > 50000 THEN 'High Earner' WHE...	5 row(s) returned	0.000 sec / 0.000 sec