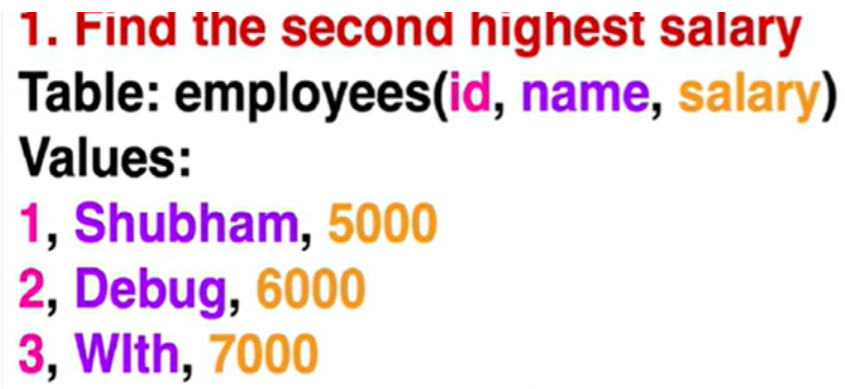


SELECT DISTINCT salary
FROM employees
ORDER BY salary DESC
LIMIT 1 OFFSET 1;

(`LIMIT` ka use hum **number of rows restrict karne** ke liye karte hain jo query return karegi.

> **"Mujhe sirf top 3 rows chahiye."**

🧠 Syntax:

SELECT * FROM table_name
LIMIT 3;

`OFFSET` ka use hota hai to **skip rows** from the top.

> **"Top 2 skip karke next 1 row chahiye."**

🧠 Syntax:

sql
CopyEdit

```
SELECT * FROM table_name
LIMIT 1 OFFSET 2;)
```

Make sure yaha debug, 5000

SELECT name, COUNT(*) // Yaha count salary wise count krega because of group by salary
FROM employees
GROUP BY salary
HAVING COUNT(*) > 1;

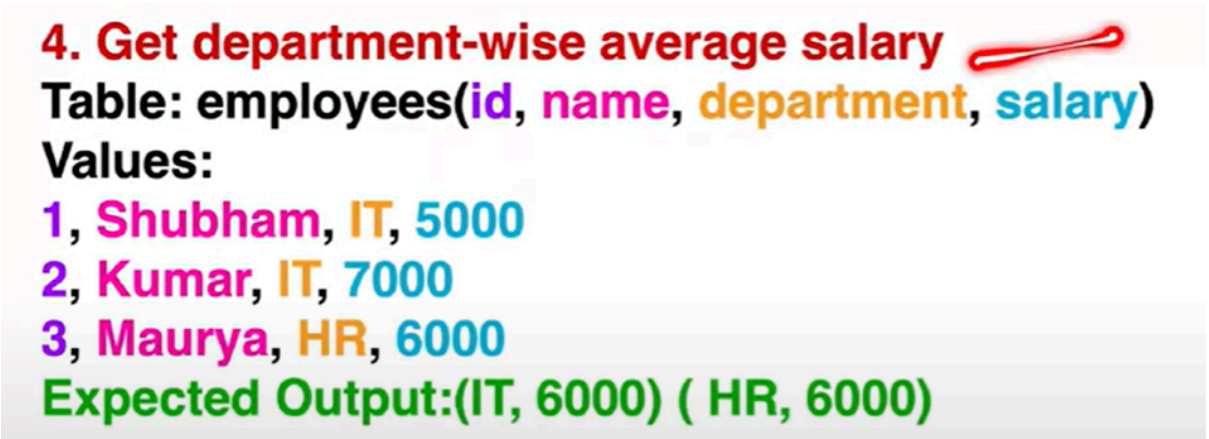
SELECT name
FROM employees
WHERE join_date >= CURDATE() - INTERVAL 3 MONTH;

✔️ `CURDATE()` ek MySQL function hai jo **aaj ki current date** deta hai.

For example: `2025-07-10` (agar aaj ki date hai).

✔️ `INTERVAL` ek MySQL ka keyword hai jo tumhe date/time se kuch subtract/add karne deta hai.

👉 `CURDATE() - INTERVAL 3 MONTH` ka matlab hai:
Aaj ki date se **3 mahine pehle ki date** nikaal lo.



SELECT department, AVG(salary) as AVG_SALARY

FROM employees

GROUP BY department;

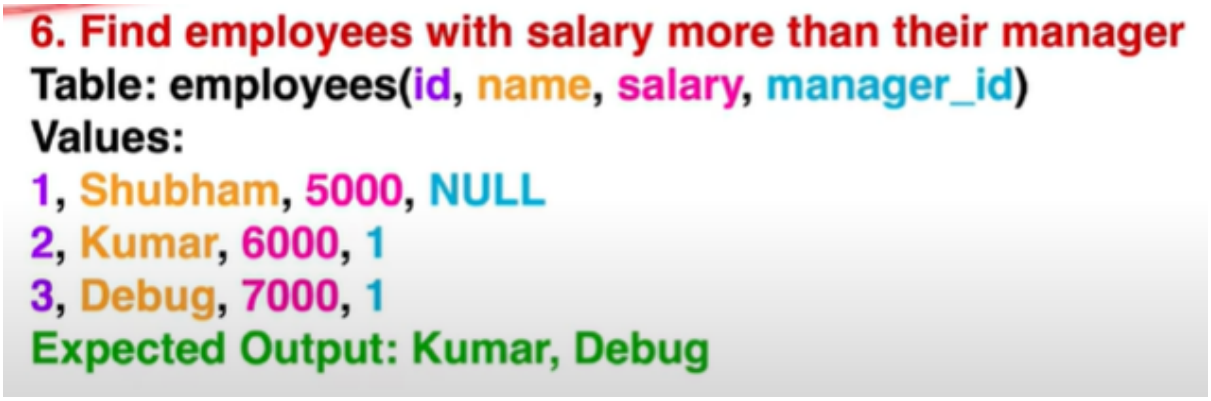
Here we have to find the 3rd highest

SELECT DISTINCT salary

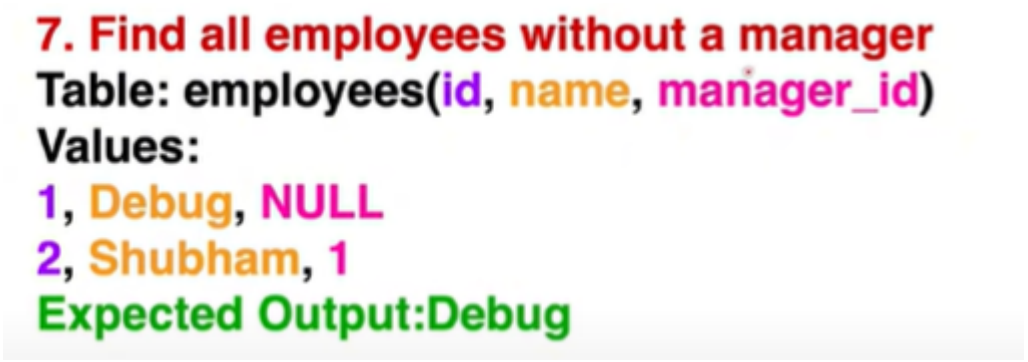
FROM employees

ORDER BY salary DESC

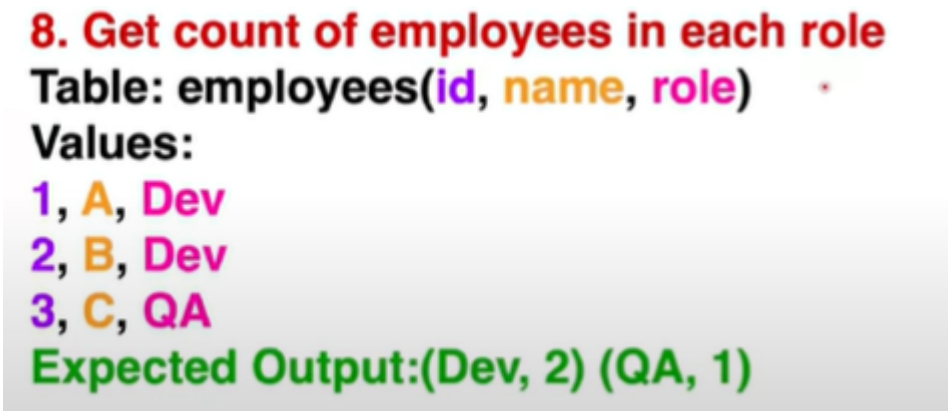
LIMIT 1 OFFSET 2;



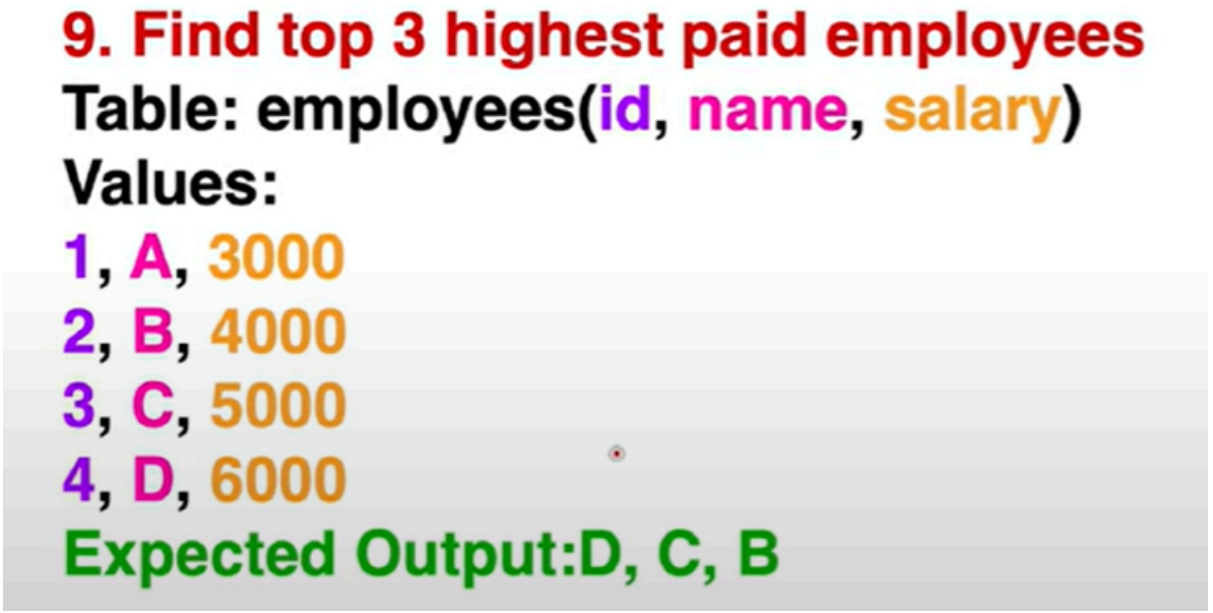

```
SELECT e.name

FROM employees e

JOIN employees m ON e.manager_id = m.id

WHERE e.salary > m.salary;
```




```
SELECT *

FROM employees

WHERE manager_id IS NULL;
```




```
SELECT role, COUNT(*)

FROM employees

GROUP BY role;
```




```
SELECT name

FROM employees

ORDER BY salary DESC

LIMIT 3 ;
```

---

10. Find common employees in two tables
Table: employees(name), contractors(name)
Values:
employees: A, B
contractors: B, C
Expected Output: B

SELECT e.name

FROM employees e

INNER JOIN contactors c

ON e.name = c.name;

---

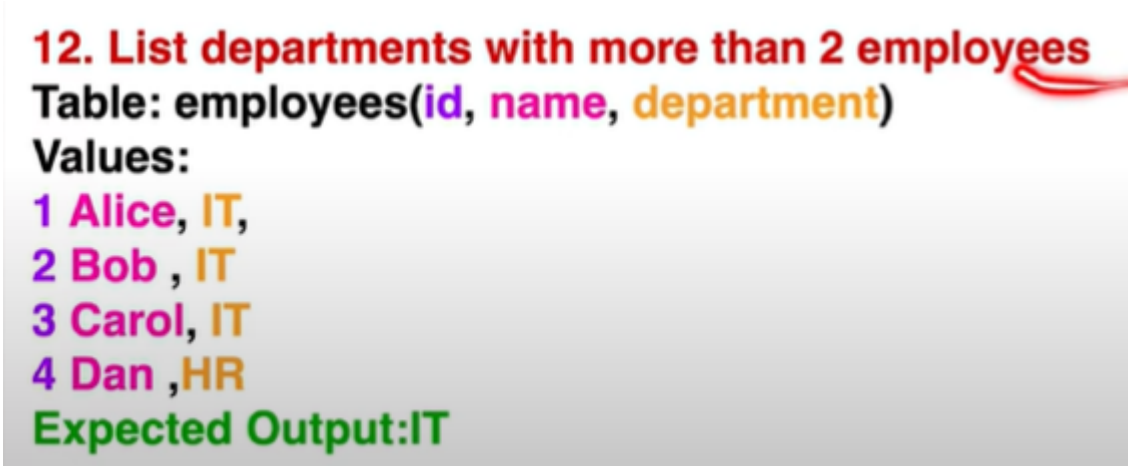
11. Find employees who earn the same as someone in 'Sales' department
Table: employees(id, name, salary, department)
Values:
1, A, 5000, Sales
2, B, 5000, IT
3, C, 6000, HR
Expected Output:A, B

SELECT name

FROM employee

WHERE salary IN

(SELECT salary FROM employee WHERE department =“Sales”);

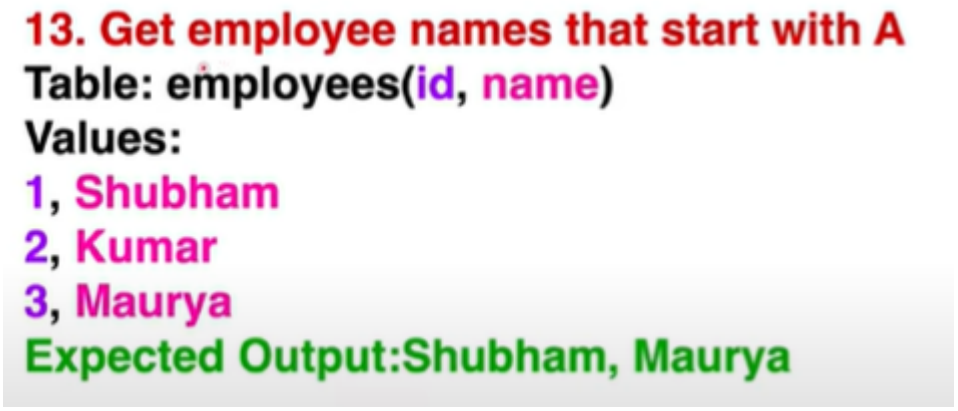


SELECT department

FROM employees

GROUP BY department

HAVING COUNT(*) > 2;



O/P is wrong here

SELECT name

FROM employees

WHERE name LIKE ‘A%’;

---

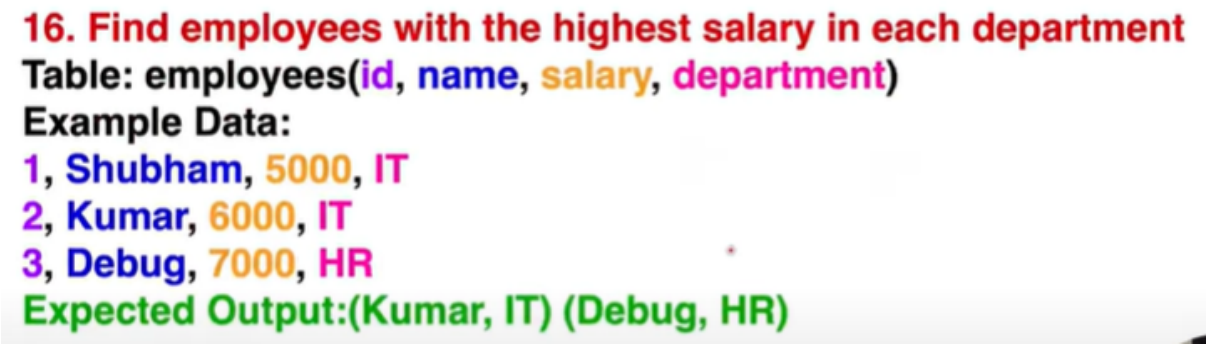
14. Get employees who dont belong to any department
Table: employees(id, name, department_id)
Values:
1, A, NULL
2, B, 1
Expected Output:A

```
SELECT name

FROM employees

WHERE department_id IS NULL;
```

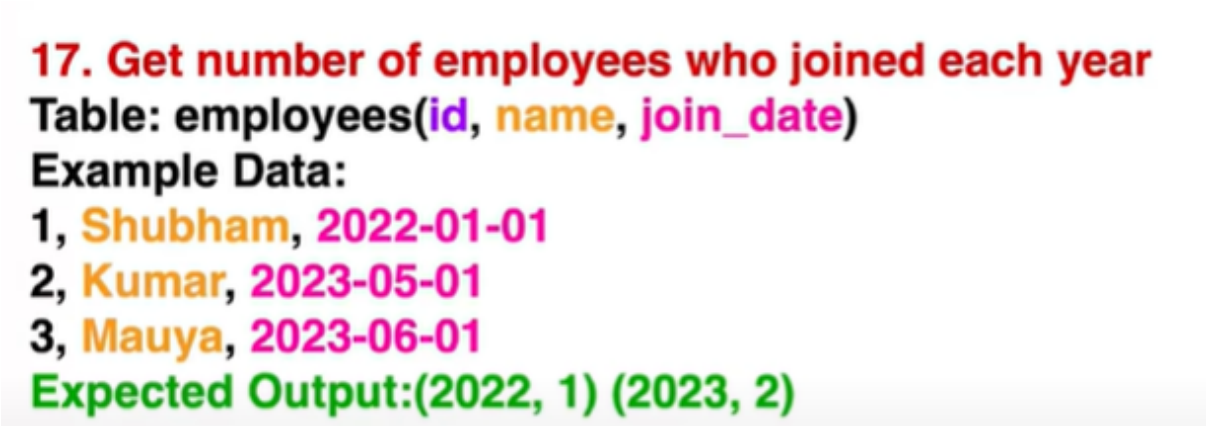
---

15. Join employees with department names
Table: employees(id, name, department_id), departments(id, department_name)
Values:
employees: 1, A, 10
employees: 2, B, 20
departments: 10, IT
departments:20, HR
Expected Output:(A, IT) (B, HR)

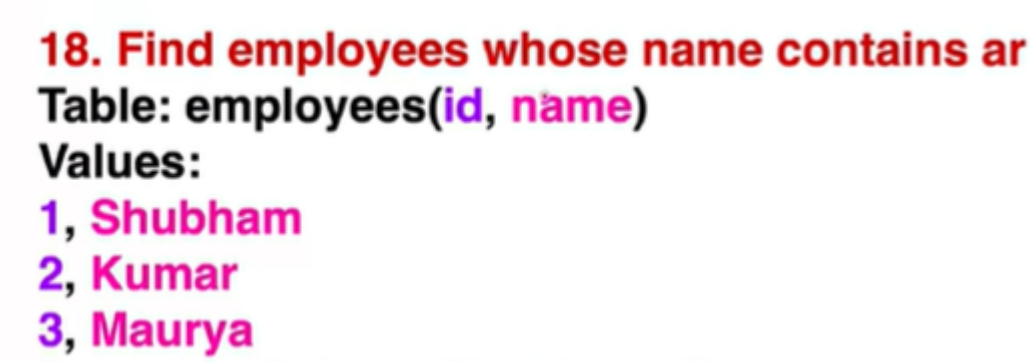
```
SELECT e.name, d.department_name

FROM employees e

INNER JOIN department d

ON e.department_id = d.id;
```




```
SELECT name, department

FROM employees e

WHERE salary = (SELECT  MAX(salary)

FROM employees

WHERE department = e.department);
```



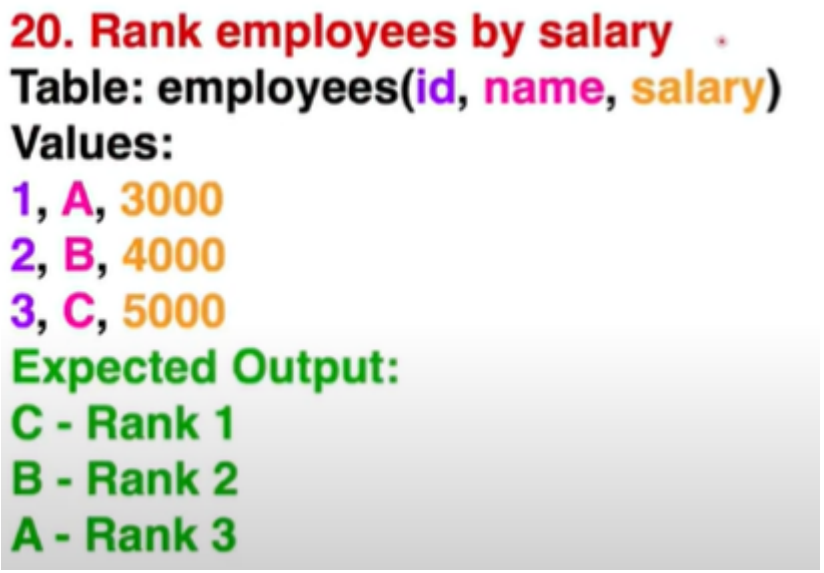

```
SELECT YEAR(join_date) AS join_year, COUNT(*)

FROM employees

GROUP BY YEAR(join_date);
```



SELECT name

FROM employees

WHERE name LIKE ‘%ar%’;



SELECT name, RANK() OVER (ORDER BY salary DESC)

AS salary_rank

FROM employees;


```
DELETE FROM employees

WHERE id NOT IN(

SELECT MIN(id)

FROM employees

GROUP BY name, email);
```
