Employee

EmpId	Name	ManagerId	DOJ	City
121	John	321	1/31/2016	Hyd
321	David	986	1/30/2018	Chennai
421	Scott	876	27/11/2020	Mumbai

Ques.1. Write an SQL query to fetch the EmpId and Name of all the employees working under Manager with id – '986'.

SELECT EmpId, Name FROM Employee WHERE ManagerId = 986;

Salary

EmpId	Project	Salary	Variable
121	P1	20000	0
321	P2	35000	1000
421	P1	50000	3000

Ques.2. Write an SQL query to fetch the different projects available from the Salary table.

SELECT DISTINCT(Project) FROM Salary;

Salary

EmpId	Project	Salary	Variable
121	P1	20000	0
321	P2	35000	1000
421	P1	50000	3000

Ques.3. Write an SQL query to fetch the count of employees working in project 'P1'.

SELECT COUNT(*) FROM Salary WHERE Project = 'P1';

Salary

EmpId	Project	Salary	Variable
121	P1	20000	0
321	P2	35000	1000
421	P1	50000	3000

Ques.4. Write an SQL query to find the maximum, minimum, and average salary of the employees.

```
SELECT Max(Salary), Min(Salary), AVG(Salary) FROM Salary;
```

Salary

EmpId	Project	Salary	Variable
121	P1	20000	0
321	P2	35000	1000
421	P1	50000	3000

Ques.5. Write an SQL query to find the employee id whose salary lies in the range of 30000 and 40000.

```
SELECT EmpId, Salary FROM Salary
WHERE Salary BETWEEN 30000 AND 40000;
```

Employee

EmpId	Name	ManagerId	DOJ	City
121	John	321	1/31/2016	Hyd
321	David	986	1/30/2018	Chennai
421	Scott	876	27/11/2020	Mumbai

Ques.6. Write an SQL query to fetch those employees who live in Chennai and work under manager with ManagerId – 986.

```
SELECT EmpId, City, ManagerId FROM Employee
WHERE City='Chennai' AND ManagerId='986';
```

Employee

EmpId	Name	ManagerId	DOJ	City
121	John	321	1/31/2016	Hyd
321	David	986	1/30/2018	Chennai
421	Scott	876	27/11/2020	Mumbai

Ques.7. Write an SQL query to fetch all the employees who either live in Chennai or work under a manager with ManagerId – 321.

```
SELECT EmpId, City, ManagerId FROM Employee
WHERE City='Chennai' OR ManagerId='321';
```

Salary

EmpId	Project	Salary	Variable
121	P1	20000	0
321	P2	35000	1000
421	P1	50000	3000

Ques.8. Write an SQL query to fetch all those employees who work on Project other than P1.

```
SELECT EmpId FROM Salary WHERE NOT Project='P1';
```

(Or)

```
SELECT EmpId FROM Salary WHERE Project <> 'P1';
```

Salary

EmpId	Project	Salary	Variable
121	P1	20000	0
321	P2	35000	1000
421	P1	50000	3000

Ques.9. Write an SQL query to display the total salary of each employee adding the Salary with Variable value.

SELECT EmpId, Salary+Variable as TotalSalary FROM Salary;

Employee

EmpId	Name	ManagerId	DOJ	City
121	John	321	1/31/2016	Hyd
321	David	986	1/30/2018	Chennai
421	Scott	876	27/11/2020	Mumbai

Ques.10. Write an SQL query to fetch the employees whose name begins with any two characters, followed by a text "vi" and ending with any sequence of characters.

SELECT Name FROM Employee WHERE Name LIKE '__vi%';

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

Ques.11. List out the employees who are not receiving commission.

Select * from employee where COMM is Null

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

Ques.12. List out the employees who are working in department 10 and draw the salaries more than 3500

Select * from employee where department_id=10 and salary>3500

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

Ques.13. List out the employee id, name in descending order based on salary column

Select employee_id, last_name, salary from employee order by salary desc

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

Ques.14. How many employees who are working in different departments wise in the organization.

Select department_id, count(*), from employee group by department_id;

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

Ques.15. List out the department id having at least 3 employees.

Select department_id, count(*) from employee group by department_id having count(*)>=3

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

Ques.16. Display the employees who got the maximum salary.

Select * from employee where salary=(select max(salary) from employee)

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

DEPARTMENT

Department_ID	Name	Location_ID
10	ACCOUNTING	122
20	RESEARCH	124
30	SALES	123
40	OPERATIONS	167

Ques.17. Display the employees who are working in Sales department.

Select * from employee where department_id IN (select department_id from department where name='SALES')

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

DEPARTMENT

Department_ID	Name	Location_ID
10	ACCOUNTING	122
20	RESEARCH	124
30	SALES	123
40	OPERATIONS	167

Location_ID

Location_ID	Regional_Group
122	NEW YORK
123	DALLAS
124	CHICAGO
167	BOSTON

Ques.18. Display the employees who are working in "New York"

Select * from employee where department_id=(select department_id from department where location_id=(select location_id from location where regional_group='New York'))

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

JOB

Job_ID	Function
667	CLERK
668	STAFF
669	ANALYST
670	SALESPERSON
671	MANAGER
672	PRESIDENT

Ques.19. Update the employees salaries, who are working as Manager on the basis of 10%.

Update employee set salary=salary*10/100 wehre job_id=(select job_id from job where function='MANAGER')

EMPLOYEE

EMPLOYEE_ID	LAST_NAME	FIRST_NAM	MIDDLE_NAME	JOB_ID	MANAGER_ID	HIRE DATE	SALARY	COMM	DEPARTMENT_ID
7369	SMITH	JOHN	Q	667	7902	17-Dec-84	800	NULL	20
7499	ALLEN	KEVIN	J	670	7698	20-Feb-85	1600	300	30
7505	DOYLE	JEAN	K	671	7839	4-Apr-85	2850	NULL	30
7506	DENNIS	LYNN	S	671	7839	15-May-85	2750	NULL	30
7507	BAKER	LESLIE	D	671	7839	10-Jun-85	2200	NULL	40
7521	WARK	CYNTHIA	D	670	7698	22-Feb-85	1250	NULL	40

DEPARTMENT

Department_ID	Name	Location_ID
10	ACCOUNTING	122
20	RESEARCH	124
30	SALES	123
40	OPERATIONS	167

Ques.20. Delete the employees who are working in accounting department.

delete from employee where department_id=(select department_id from department where name='ACCOUNTING')

21. Display the highest salary from Employee table

Your query:

sql Copy code

```sql
SELECT *
FROM Employee
ORDER BY Salary DESC
LIMIT 1;
```

✔ Correct (with a note)

- This returns **the employee row** having the highest salary.
- If the question strictly asks for **highest salary value only**, then better is:

sql Copy code

```sql
SELECT MAX(Salary)
FROM Employee;
```

✅ Conclusion:

Your query is correct if you want **employee details**, not just the salary.

22. Display the second highest salary from Employee table

Your query:

sql Copy code

```sql
SELECT *
FROM Employee
ORDER BY Salary DESC
OFFSET 1
LIMIT 1;
```

⚠️ Partially correct

This works **only if:**

- Salaries are **unique**
- Database supports `OFFSET` (PostgreSQL, MySQL 8+, etc.)

❌ Problem:

If **duplicate salaries** exist, this may still return the highest salary again.

✅ Correct & safer version:

sql Copy code

```sql
SELECT *
FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET 1;
```

✅ Best practice (handles duplicates):

sql Copy code

```sql
SELECT *
FROM Employee
WHERE Salary = (
    SELECT DISTINCT Salary
    FROM Employee
    ORDER BY Salary DESC
    LIMIT 1 OFFSET 1
);
```

23. Display the n-th highest salary from Employee table

Your query:

```sql
SELECT *
FROM Employee
ORDER BY Salary DESC
OFFSET n-1
LIMIT 1;
```

❌ Not fully correct

- SQL does **not understand** `n-1` **directly**
- Same **duplicate salary issue**

✅ Correct generic version (unique salaries assumed):

```sql
SELECT *
FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET (n - 1);
```

✅ Best & interview-safe version:

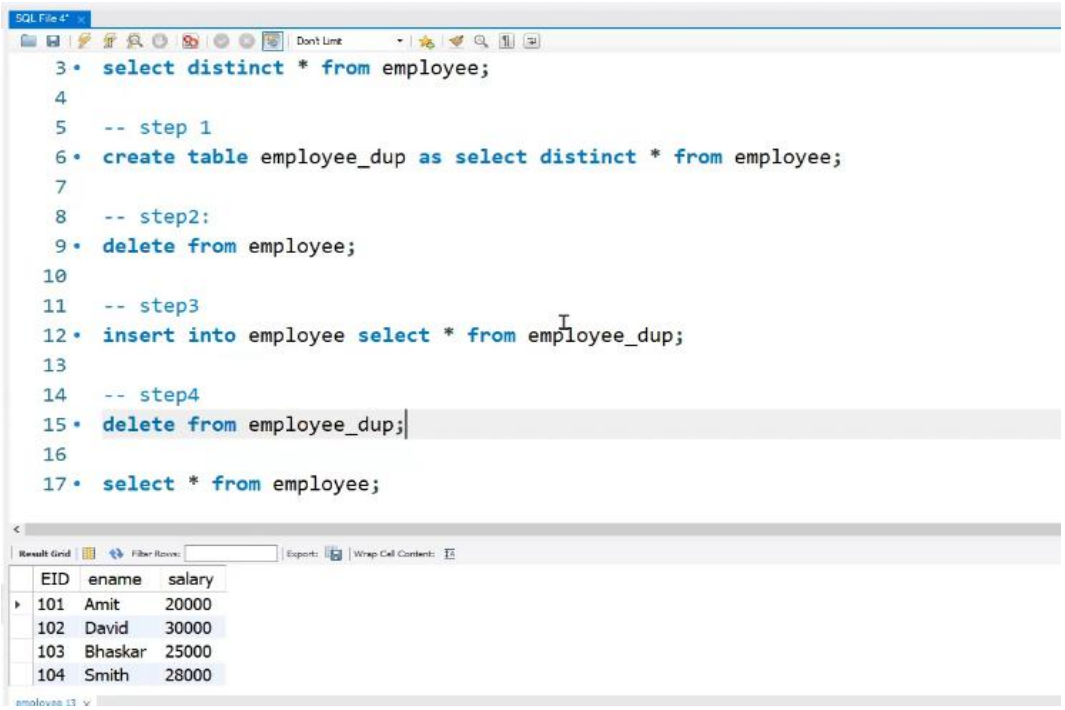
```sql
SELECT *
FROM Employee
WHERE Salary = (
    SELECT DISTINCT Salary
    FROM Employee
    ORDER BY Salary DESC
    LIMIT 1 OFFSET (n - 1)
);
```

✅ Final Summary

Query	Correct?	Notes
Highest salary	✅	Use `MAX()` if only salary needed
Second highest salary	⚠️	Fails with duplicates
Nth highest salary	❌	Needs correction & DISTINCT

⭐ Interview Tip (important)

> **Always handle duplicate salaries** unless the problem explicitly says salaries are unique.


SQL File 4*
Don't Limit
3 • select distinct * from employee;
4
5 -- step 1
6 • create table employee_dup as select distinct * from employee;
7
8 -- step2:
9 • delete from employee;
10
11 -- step3
12 • insert into employee select * from employee_dup;
13
14 -- step4
15 • delete from employee_dup;
16
17 • select * from employee;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
EID ename salary
101 Amit 20000
102 David 30000
103 Bhaskar 25000
104 Smith 28000

```sql
-- Step 0: View unique (distinct) rows from Employee table
-- This does NOT delete duplicates, it only displays them
SELECT DISTINCT *
FROM Employee;


-- Step 1: Create a temporary table containing only unique rows
-- NOTE: This copies only data, not indexes, constraints, or keys
CREATE TABLE emp_unique_data AS
SELECT DISTINCT *
FROM Employee;


-- Step 2: Remove all data from the original Employee table
-- TRUNCATE is faster than DELETE and resets the table
TRUNCATE TABLE Employee;


-- Step 3: Insert unique records back into the Employee table
-- Ensure column order matches exactly
INSERT INTO Employee
SELECT *
FROM emp_unique_data;


-- Step 4: Drop the temporary table as it is no longer required
DROP TABLE emp_unique_data;


-- Step 5: Verify that Employee table now contains only unique rows
SELECT DISTINCT *
FROM Employee;
```

```sql
-- Step 1: View all data from Employee table
SELECT *
FROM Employee;


-- Step 2: Use ALTER TABLE when we need to MODIFY the structure of an existing table
-- Here, ALTER is used to ADD a new IDENTITY column (auto_id)
-- This column helps uniquely identify each row so that duplicates can be removed safely
ALTER TABLE Employee
ADD auto_id INTEGER GENERATED BY DEFAULT AS IDENTITY;


-- Step 3: Identify the rows to KEEP
-- For each duplicate group (eid, ename), we keep the row
-- that has the minimum auto_id value
SELECT MIN(auto_id)
FROM Employee
GROUP BY eid, ename;


-- Step 4: Identify duplicate rows to be DELETED
-- Rows whose auto_id is NOT the minimum within their duplicate group
SELECT *
FROM Employee
WHERE auto_id NOT IN (
    SELECT MIN(auto_id)
    FROM Employee
    GROUP BY eid, ename
);
```

```sql
-- Step 5: Delete duplicate rows
-- This deletes all extra duplicate records and keeps one unique row per group
DELETE FROM Employee
WHERE auto_id NOT IN (
    SELECT MIN(auto_id)
    FROM Employee
    GROUP BY eid, ename
);


-- Step 6: Verify that only unique rows remain in Employee table
SELECT *
FROM Employee;
```