



## **ASSIGNMENT TITLE**

### **SUBQUERIES ASSIGNMENT**

**Submitted By : Abhilasha Pareek**

**Course : Data Analytics With AI – September batch Live**

**Institute : PW Skills**

**Date : 12 December 2025**

## TABLE 1: Employee

Create Table Query:

Query	Query History
1	<pre>CREATE TABLE Employee (     emp_id INT PRIMARY KEY,     name VARCHAR(50),     department_id VARCHAR(10),     salary INT );</pre>
2	
3	
4	
5	
6	
7	

  

Data Output	Messages	Notifications
	CREATE TABLE	

Query returned successfully in 19 secs 867 msec.

Insert Data :

Query	Query History
1	<pre>INSERT INTO Employee(emp_id, name, department_id, salary) VALUES (101, 'Abhishek', 'D01', 62000), (102, 'Shubham', 'D01', 58000), (103, 'Priya', 'D02', 67000), (104, 'Rohit', 'D02', 64000), (105, 'Neha', 'D03', 72000), (106, 'Aman', 'D03', 55000), (107, 'Ravi', 'D04', 60000), (108, 'Sneha', 'D04', 75000), (109, 'Kiran', 'D05', 70000), (110, 'Tanuja', 'D05', 65000);</pre>
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

  

Data Output	Messages	Notifications
	INSERT 0 10	

Query returned successfully in 17 secs 755 msec.

## TABLE 2: Department

## Create Table:

```
Query  Query History  
1  CREATE TABLE Department (  
2      department_id VARCHAR(10) PRIMARY KEY,  
3      department_name VARCHAR(50),  
4      location VARCHAR(50)  
5  );  
6
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 17 secs 642 msec.

## Insert Data:

```
Query  Query History  
1  INSERT INTO Department VALUES  
2      ('D01', 'Sales', 'Mumbai'),  
3      ('D02', 'Marketing', 'Delhi'),  
4      ('D03', 'Finance', 'Pune'),  
5      ('D04', 'HR', 'Bengaluru'),  
6      ('D05', 'IT', 'Hyderabad');  
7
```

Data Output Messages Notifications

INSERT 0 5

Query returned successfully in 9 secs 103 msec.

### TABLE 3: Sales

Create Table:

Query	Query History
1	<pre>CREATE TABLE Sales ( 2     sale_id INT PRIMARY KEY, 3     emp_id INT, 4     sale_amount INT, 5     sale_date DATE 6 ); 7 8</pre>
Data Output	Messages Notifications
CREATE TABLE	
Query returned successfully in 13 secs 197 msec.	

Insert Data:

Query    Query History

```
1  INSERT INTO Sales VALUES
2  (201, 101, 4500, '2025-01-05'),
3  (202, 102, 7800, '2025-01-10'),
4  (203, 103, 6700, '2025-01-14'),
5  (204, 104, 12000, '2025-01-20'),
6  (205, 105, 9800, '2025-02-02'),
7  (206, 106, 10500, '2025-02-05'),
8  (207, 107, 3200, '2025-02-09'),
9  (208, 108, 5100, '2025-02-15'),
10 (209, 109, 3900, '2025-02-20'),
11 (210, 110, 7200, '2025-03-01');
12
```

Data Output    Messages    Notifications

INSERT 0 10

Query returned successfully in 21 secs 834 msec.

## 1. Retrieve employees who earn more than the average salary of all employees.

Query:

Query Query History

```
1  SELECT name, salary
2  FROM Employee
3  WHERE salary > (SELECT AVG(salary) FROM Employee);
4
```

Data Output Messages Notifications

The screenshot shows a database interface with a toolbar at the top containing various icons for file operations. Below the toolbar is a message bar stating "Showing rows: 1 to 5". The main area displays a table with two columns: "name" and "salary". The "name" column contains five entries: Priya, Neha, Sneha, Kiran, and Tanuja. The "salary" column contains the corresponding values: 67000, 72000, 75000, 70000, and 65000 respectively. The table has a header row with column names and data types: "name character varying (50)" and "salary integer".

	name character varying (50)	salary integer
1	Priya	67000
2	Neha	72000
3	Sneha	75000
4	Kiran	70000
5	Tanuja	65000

**Explanation:**

- Subquery calculates the **average salary** of all employees.
- Outer query returns employees whose salary is **greater** than this average.
- This is a scalar subquery because it returns a single value.

**2. Find employees who belong to the department with the highest average salary.**

**Query:**

Query    Query History

```

1  SELECT *
2  FROM Employee
3  WHERE department_id = (
4      SELECT department_id
5      FROM Employee
6      GROUP BY department_id
7      ORDER BY AVG(salary) DESC
8      LIMIT 1
9  );
10
11

```

Data Output    Messages    Notifications

Showing rows: 1 to 2

	emp_id [PK] integer	name character varying (50)	department_id character varying (10)	salary integer
1	109	Kiran	D05	70000
2	110	Tanuja	D05	65000

Total rows: 2    Query complete 00:00:13.523

### Explanation:

- First, subquery finds the **department with the highest avg salary**.
- Outer query returns all employees in that department.
- This is a nested analytical subquery.

### 3. List employees who have made at least one sale.

Query:

Query    Query History

```

1  SELECT *
2  FROM Employee
3  WHERE emp_id IN (SELECT emp_id FROM Sales);
4
5

```

Data Output   Messages   Notifications

Showing rows: 1 to 10

	emp_id [PK] integer	name character varying (50)	department_id character varying (10)	salary integer
1	101	Abhishek	D01	62000
2	102	Shubham	D01	58000
3	103	Priya	D02	67000
4	104	Rohit	D02	64000
5	105	Neha	D03	72000
6	106	Aman	D03	55000
7	107	Ravi	D04	60000
8	108	Sneha	D04	75000
9	109	Kiran	D05	70000
10	110	Tanuja	D05	65000

### Explanation:

- Subquery fetches all employee IDs appearing in Sales.
- IN checks if employee appears at least once.
- Used for activity/existence checks.

### 4. Find the employee with the highest sale amount.

Query:

Query    Query History

```
1  SELECT *
2  FROM Employee
3  WHERE emp_id = (
4      SELECT emp_id
5      FROM Sales
6      ORDER BY sale_amount DESC
7      LIMIT 1
8  );
9
10
11
```

Data Output    Messages    Notifications

The screenshot shows a database interface with a toolbar at the top containing icons for file operations and SQL. The main area displays a table titled 'Employee' with the following columns: emp\_id [PK] integer, name character varying (50), department\_id character varying (10), and salary integer. A single row is shown with values: 1, 104, Rohit, D02, and 64000. A status bar at the bottom right indicates 'Showing rows: 1 to 1'.

	emp_id [PK] integer	name character varying (50)	department_id character varying (10)	salary integer
1	104	Rohit	D02	64000

**Explanation:**

- Subquery finds the emp\_id with **highest sale**.
- Outer query returns employee details.
- Classic top-performer identification query.

## 5. Retrieve employees whose salaries are higher than Shubham's salary.

**Query:**

Query    Query History

```
1  SELECT name, salary
2  FROM Employee
3  WHERE salary > (SELECT salary FROM Employee WHERE name = 'Shubham');
4
5
6
```

Data Output Messages Notifications



	name character varying (50)	salary integer
1	Abhishek	62000
2	Priya	67000
3	Rohit	64000
4	Neha	72000
5	Ravi	60000
6	Sneha	75000
7	Kiran	70000
8	Tanuja	65000

Total rows: 8 Query complete 00:00:17.906

#### Explanation:

- Subquery returns Shubham's salary.
- Outer query compares everyone's salary with it.
- Example of single-value comparison subquery.

## INTERMEDIATE LEVEL

### 6. Find employees who work in the same department as Abhishek.

Query:

Query Query History

```
1  SELECT *
2  FROM Employee
3  WHERE department_id = (
4      SELECT department_id
5      FROM Employee
6      WHERE name = 'Abhishek'
7  );
8
9
10
11
```

Data Output Messages Notifications

The screenshot shows a database interface with a toolbar at the top containing various icons for file operations like new, open, save, and export. Below the toolbar is a table with five columns: emp\_id, name, department\_id, and salary. The table has two rows of data. Row 1 contains emp\_id 101, name Abhishek, department\_id D01, and salary 62000. Row 2 contains emp\_id 102, name Shubham, department\_id D01, and salary 58000.

	emp_id [PK] integer	name character varying (50)	department_id character varying (10)	salary integer
1	101	Abhishek	D01	62000
2	102	Shubham	D01	58000

Total rows: 2 | Query complete 00:00:20.875

### Explanation:

- Subquery identifies Abhishek's department.
- Outer query lists all employees in that department.
- Used for team-based filtering.

## 7. List departments with at least one employee earning more than ₹60,000.

Query:

Query Query History

```
1 SELECT *
2 FROM Department
3 WHERE department_id IN (
4     SELECT department_id
5     FROM Employee
6     WHERE salary > 60000
7 );
8
9
10
11
12
```

Data Output Messages Notifications

The screenshot shows a database interface with a toolbar at the top containing various icons for file operations like new, open, save, and export. Below the toolbar is a table with four columns: department\_id, department\_name, and location. The table has 5 rows of data. At the bottom of the interface, there is a status bar showing 'Total rows: 5' and 'Query complete 00:00:21.761'.

	department_id [PK] character varying (10)	department_name character varying (50)	location character varying (50)
1	D01	Sales	Mumbai
2	D02	Marketing	Delhi
3	D03	Finance	Pune
4	D04	HR	Bengaluru
5	D05	IT	Hyderabad

### Explanation:

- Subquery returns department IDs where salary exceeds 60,000.
- Outer query returns matching departments.

## 8. Find department name of the employee with the highest sale.

Query:

Query History

Query

```
1 FROM Department
2 WHERE department_id = (
3     SELECT department_id
4     FROM Employee
5     WHERE emp_id = (
6         SELECT emp_id
7         FROM Sales
8         ORDER BY sale_amount DESC
9         LIMIT 1
10    )
11  );
12
13
```

Data Output Messages Notifications

Showing rows: 1 to 1

	department_name
1	Marketing

### Explanation:

3-step subquery chain:

1. Find employee with highest sale
2. Get their department
3. Fetch department name

## 9. Employees who made sales greater than the average sale amount.

Query:

Query Query History

```
1  SELECT name
2  FROM Employee
3  WHERE emp_id IN (
4      SELECT emp_id
5      FROM Sales
6      WHERE sale_amount > (SELECT AVG(sale_amount) FROM Sales)
7  );
8
9
10
11
12
```

Data Output Messages Notifications

The screenshot shows a SQL query results interface. At the top, there are tabs for 'Data Output', 'Messages', and 'Notifications'. Below the tabs is a toolbar with various icons for file operations like new, open, save, print, and export. To the right of the toolbar, it says 'Showing rows: 1 to 5'. The main area displays a table with one column labeled 'name' and a type annotation 'character varying (50)'. There are five rows of data: 1. Shubham, 2. Rohit, 3. Neha, 4. Aman, and 5. Tanuja. At the bottom left, it says 'Total rows: 5', and at the bottom right, it says 'Query complete 00:00:11.461'.

	name character varying (50)
1	Shubham
2	Rohit
3	Neha
4	Aman
5	Tanuja

### Explanation:

- Subquery calculates average sale.
- Then finds employees whose sale exceeds this average.

## 10. Total sales made by employees earning above average salary.

Query:

Query Query History

```
1  SELECT SUM(sale_amount) AS total_sales
2  FROM Sales
3  WHERE emp_id IN (
4      SELECT emp_id
5      FROM Employee
6      WHERE salary > (SELECT AVG(salary) FROM Employee)
7  );
8
9
10
11
12
```

Data Output Messages Notifications

The screenshot shows a database query results window. At the top, there are tabs for 'Data Output', 'Messages', and 'Notifications'. Below the tabs is a toolbar with various icons. To the right of the toolbar, it says 'Showing rows: 1 to 1'. The main area displays a table with one row of data. The table has a single column labeled 'total\_sales' with a data type of 'bigint'. The value in the first row is '32700'. There is also a small lock icon next to the column header.

	total_sales
1	32700

Total rows: 1 | Query complete 00:00:14.945

### Explanation:

- First find employees with above-average salary.
- Then sum their total sales.
- Good example of cross-table metric analysis.

## ADVANCED LEVEL

### 11. Employees who have NOT made any sales.

Query:

Query Query History

```
1  SELECT name
2  FROM Employee
3  WHERE emp_id NOT IN (SELECT emp_id FROM Sales);
4
5
6
7
8
9
```

Data Output Messages Notifications

	name	
	character varying (50)	🔒

Total rows: 0    Query complete 00:00:24.659

#### Explanation:

- Subquery returns employees who made sales.
- NOT IN filters out everyone except non-performers.

## 12. Departments where average salary is above ₹55,000.

Query:

Query    Query History

```
1  SELECT department_name
2  FROM Department
3  WHERE department_id IN (
4      SELECT department_id
5      FROM Employee
6      GROUP BY department_id
7      HAVING AVG(salary) > 55000
8  );
9
10
11
12
```

Data Output    Messages    Notifications

	department_name
1	Sales
2	Marketing
3	Finance
4	HR
5	IT

Total rows: 5    Query complete 00:00:21.267

#### Explanation:

- Group employees department-wise.
- Apply HAVING for avg salary filter.
- Outer query matches department names.

### 13. Departments where total sales exceed ₹10,000.

Query:

Query Query History

```
1  SELECT department_name
2  FROM Department
3  WHERE department_id IN (
4      SELECT E.department_id
5      FROM Employee E
6      JOIN Sales S ON E.emp_id = S.emp_id
7      GROUP BY E.department_id
8      HAVING SUM(S.sale_amount) > 10000
9  );
10
11
12
```

Data Output Messages Notifications

The screenshot shows a database query results interface. At the top, there are tabs for 'Data Output', 'Messages', and 'Notifications'. Below the tabs is a toolbar with various icons for file operations like copy, paste, and save. To the right of the toolbar, it says 'Showing rows: 1 to 4'. The main area displays a table with four rows, each containing a number from 1 to 4 and a corresponding department name: Sales, Marketing, Finance, and IT. The first column is numerical (1, 2, 3, 4), and the second column is labeled 'department\_name' with a data type of 'character varying (50)'.

	department_name
1	Sales
2	Marketing
3	Finance
4	IT

Total rows: 4 | Query complete 00:00:21.711

### Explanation:

- Join employees with their sales.
- Group by department.
- Filter total sales > 10,000.

## 14. Employee who made the second-highest sale.

Query:

Query Query History

```
1  SELECT name
2  FROM Employee
3  WHERE emp_id = (
4      SELECT emp_id
5      FROM Sales
6      ORDER BY sale_amount DESC
7      LIMIT 1 OFFSET 1
8  );
9
10
11
12
```

Data Output Messages Notifications



Showing rows: 1 to 1

	name
	character varying (50)
1	Aman

### Explanation:

- ORDER BY DESC → highest at top
- OFFSET 1 → skip the highest
- LIMIT 1 → pick second highest
- Outer query fetches employee name.

## 15. Employees whose salary is greater than the highest sale amount.

Query:

Query Query History

```
1  SELECT name
2  FROM Employee
3  WHERE salary > (SELECT MAX(sale_amount) FROM Sales);
4
5
6
```

Data Output    Messages    Notifications



Showing rows: 1 to 10

	name character varying (50)
1	Abhishek
2	Shubham
3	Priya
4	Rohit
5	Neha
6	Aman
7	Ravi
8	Sneha
9	Kiran
10	Tanuja

### Explanation:

- Subquery computes highest sale amount.
- Outer query finds employees whose salary exceeds that amounts.