ASSIGNMENT TITLE

SUBQUERIES ASSIGNMENT

Submitted By : Bhaskar Singh Rajput

Course : Data Analytics With AI – September batch Live

Institude : PW Skills

Date : 23 December 2025

TABLE 1: Employee

Create Table Query:

Query   Query History

```
1  CREATE TABLE Employee (
2      emp_id INT PRIMARY KEY,
3      name VARCHAR(50),
4      department_id VARCHAR(10),
5      salary INT
6  );
7
```

Data Output   Messages   Notifications

```
CREATE TABLE

Query returned successfully in 19 secs 867 msec.
```

Insert Data :

Query   Query History

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

```sql
1   CREATE TABLE Sales (
2       sale_id INT PRIMARY KEY,
3       emp_id INT,
4       sale_amount INT,
5       sale_date DATE
6   );
7
8
```

Data Output   Messages   Notifications

```
CREATE TABLE

Query returned successfully in 13 secs 197 msec.
```

Insert Data:

```
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

Showing rows: 1 to 5

	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.
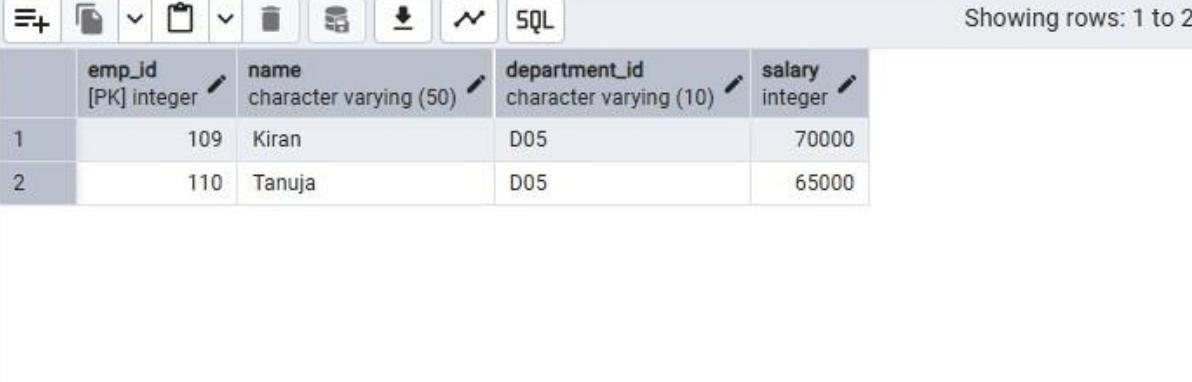
Query:

```
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
```

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

Showing rows: 1 to 1

emp_id [PK] integer	name character varying (50)	department_id character varying (10)	salary integer
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
```

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

	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

	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

Total rows: 5   Query complete 00:00:21.761

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:

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

Data Output   Messages   Notifications

Showing rows: 1 to 1

	department_name character varying (50)
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

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

Showing rows: 1 to 5

Total rows: 5   Query complete 00:00:11.461

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

=+ ▣ ∨ ▢ ∨ ▮   ▤   ± ∿ SQL   Showing rows: 1 to 1

	total_sales 🔒 bigint
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:

```sql
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 character varying (50) 🔒
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

Showing rows: 1 to 4

department_name character varying (50)
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

	name character varying (50) 🔒
1	Aman

Showing rows: 1 to 1

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
```

	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.