SUBQUERY

- DEFINITION
- GUIDELINES
- TYPES
  - Single row
  - Multiple row
  - Correlated
  - Nested
- SUBQUERY WITH
  - Exists and not exists
  - Insertion
  - Update
  - Delete
- LAB EXERCISE

Subquery/Inner query

- a query within another SQL query
- It can be used in a SELECT, INSERT, DELETE, or UPDATE statement
- It perform the following tasks:
  - Compare an expression to the result of the query.
  - Determine if an expression is included in the results of the query.
  - Check whether the query selects any rows.
  - The subquery executes once before the main query (outer query) executes.
  - The main query use the subquery result.

Subqueries: Guidelines

- must be enclosed in parentheses.
- must be placed on the right side of the comparison operator
- An ORDER BY command cannot be used in a subquery.
- Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.
- The BETWEEN operator cannot be used with a subquery. However, the BETWEEN operator can be used within the subquery.

Types

Single row subquery : Returns zero or one row.

Multiple row subquery : Returns one or more rows.

Correlated subqueries : Reference one or more columns in the outer SQL statement. The subquery is known as a correlated subquery because the subquery is related to the outer SQL statement.

Nested subqueries : Subqueries are placed within another subquery.

Single Row Subqueries

A single row subquery returns zero or one row to the outer SQL statement.

```
mysql> select * from oldcafeteria;
+--------+------------+-----------------+------------+---------------+
| ord_no | ord_amount | advaance_amount | agent_code | customer_code |
+--------+------------+-----------------+------------+---------------+
|      1 |        200 |            2000 | oc001      | cus001        |
|      2 |        300 |            3000 | oc002      | cus002        |
|      3 |       3000 |           30000 | oc007      | cus009        |
|      3 |       7000 |           20000 | oc009      | cus010        |
|      4 |        600 |            5000 | oc010      | cus011        |
+--------+------------+-----------------+------------+---------------+
5 rows in set (0.00 sec)

mysql> select ord_no, ord_amount from oldcafeteria where ord_no=(select ord_no from oldcafeteria where agent_code='oc001');
+--------+------------+
| ord_no | ord_amount |
+--------+------------+
|      1 |        200 |
+--------+------------+
1 row in set (0.00 sec)
```

Multiple Row Subqueries

- returns one or more rows to the outer SQL statement
- use the IN, ANY, or ALL operator in outer query to handle a subquery that returns multiple rows

```
mysql> select ord_no, ord_amount from oldcafeteria where ord_no in (select ord_no from oldcafeteria where advaance_amount<=20000);
+--------+------------+
| ord_no | ord_amount |
+--------+------------+
|      1 |        200 |
|      2 |        300 |
|      3 |       3000 |
|      3 |       7000 |
|      4 |        600 |
+--------+------------+
5 rows in set (0.10 sec)
```

Correlated Subqueries

- a subquery is correlated with the outer query using outerr keyword
- subquery uses information from the outer query
- subquery executes once for every row in the outer query

```
mysql> select * from employees1;
+-------------+------------+-----------+--------+---------------+
| employee_id | first_name | last_name | salary | department_id |
+-------------+------------+-----------+--------+---------------+
|           4 | anil       | sarkar    |  40000 | 80            |
|           5 | sunil      | malhotra  |  50000 | 80            |
|           6 | Dinesh     | Kartik    |  60000 | 90            |
|           7 | Dinesh1    | Kartik1   |  65000 | 90            |
|           8 | Ramesh1    | Jaiswal1  |  75000 | 90            |
+-------------+------------+-----------+--------+---------------+
5 rows in set (0.00 sec)

mysql> SELECT last_name, salary, department_id FROM employees1 outerr
    -> WHERE salary > (SELECT AVG(salary) FROM employees1
    -> WHERE department_id = outerr.department_id);
+-----------+--------+---------------+
| last_name | salary | department_id |
+-----------+--------+---------------+
| malhotra  |  50000 | 80            |
| Jaiswal1  |  75000 | 90            |
+-----------+--------+---------------+
2 rows in set (0.00 sec)
```

find all employees who earn more than the average salary in their department

Nested subqueries

```
mysql> select employee_id from employees1 where exists(select * from employees1 where exists
    -> (select* from employees1 where salary>1000)
    -> );
+-------------+
| employee_id |
+-------------+
|           4 |
|           5 |
|           6 |
|           7 |
|           8 |
+-------------+
5 rows in set (0.05 sec)
```

subquery with EXISTS and NOT EXISTS

subquery returns a Boolean value of True or false

TRUE: Subquery return any rows
False : Doesn't return any rows

SELECT * FROM table_name **WHERE EXISTS**(subquery);

Find the employee_id whose salary is greater than 1000.

```
mysql> select employee_id from employees1 where exists(select * from employees1 where salary>1000);
+-------------+
| employee_id |
+-------------+
|           4 |
|           5 |
|           6 |
|           7 |
|           8 |
+-------------+
5 rows in set (0.00 sec)

mysql> select employee_id from employees1 where exists(select * from employees1 where salary<1000);
Empty set (0.00 sec)
```

Inserting records using subqueries

INSERT INTO table_name [(column1 [, column2])] SELECT [*|column1 [, column2] FROM table1 [, table2] [WHERE VALUE OPERATOR];

```
mysql> select * from oldcafeteria;
+--------+------------+-----------------+------------+---------------+
| ord_no | ord_amount | advaance_amount | agent_code | customer_code |
+--------+------------+-----------------+------------+---------------+
|      1 |        200 |            2000 | oc001      | cus001        |
|      2 |        300 |            3000 | oc002      | cus002        |
|      3 |       3000 |           30000 | oc007      | cus009        |
|      3 |       7000 |           20000 | oc009      | cus010        |
|      4 |        600 |            5000 | oc010      | cus011        |
+--------+------------+-----------------+------------+---------------+
```

```
mysql> insert into newcafeteria select * from oldcafeteria where advaance_amount in (2000,25000);
Query OK, 1 row affected (0.14 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

```
mysql> select * from newcafeteria;
+--------+------------+-----------------+------------+---------------+
| ord_no | ord_amount | advaance_amount | agent_code | customer_code |
+--------+------------+-----------------+------------+---------------+
|      1 |        200 |            2000 | oc001      | cus001        |
+--------+------------+-----------------+------------+---------------+
1 row in set (0.05 sec)
```

Old cafeteria and newcafeteria has same attributes

Subqueries with UPDATE statement

UPDATE table SET column_name = new_value [WHERE OPERATOR [VALUE]
(SELECT COLUMN_NAME FROM TABLE_NAME) [WHERE)]

```
mysql> update newcafeteria set ord_no =3 where advaance_amount-ord_amount>(select min(ord_amount)from oldcafeteria);
Query OK, 2 rows affected (0.30 sec)
Rows matched: 3  Changed: 2  Warnings: 0

mysql> select* from newcafeteria;
+--------+------------+-----------------+------------+---------------+
| ord_no | ord_amount | advaance_amount | agent_code | customer_code |
+--------+------------+-----------------+------------+---------------+
|      3 |        200 |            2000 | oc001      | cus001        |
|      3 |        200 |            2000 | oc001      | cus001        |
|      3 |       3000 |           30000 | oc007      | cus009        |
+--------+------------+-----------------+------------+---------------+
3 rows in set (0.00 sec)
```

Update the rows of newcafeteria whose (advance_amount-ord_amount) is greater than minimum ord_amount from old cafeteria

Subqueries with DELETE statement

DELETE FROM TABLE_NAME [WHERE OPERATOR [VALUE] (SELECT COLUMN_NAME FROM TABLE_NAME) [WHERE)]

```
mysql> select* from newcafeteria;
+--------+------------+-----------------+------------+---------------+
| ord_no | ord_amount | advaance_amount | agent_code | customer_code |
+--------+------------+-----------------+------------+---------------+
|      3 |        200 |            2000 | oc001      | cus001        |
|      3 |        200 |            2000 | oc001      | cus001        |
|      3 |       3000 |           30000 | oc007      | cus009        |
+--------+------------+-----------------+------------+---------------+
3 rows in set (0.00 sec)
```

delete those orders from 'newcafeteria' table which advance_amount are less than the maximum advance_amount of ‘oldcafeteria' table

```
mysql> DELETE FROM newcafeteria WHERE advaance_amount< (SELECT MAX(advaance_amount) FROM oldcafeteria);
Query OK, 2 rows affected (0.32 sec)

mysql> select*from newcafeteria;
+--------+------------+-----------------+------------+---------------+
| ord_no | ord_amount | advaance_amount | agent_code | customer_code |
+--------+------------+-----------------+------------+---------------+
|      3 |       3000 |           30000 | oc007      | cus009        |
+--------+------------+-----------------+------------+---------------+
1 row in set (0.07 sec)
```

LAB EXERCISE

Ord_num	Ord_amount	Advance_amount	Ord_date	Cust_code	Agent_code	Description
004	200	3000	15-aug-2020	C004	Ac001	Masala kulcha
007	600	5000	17-sept-2020	C006	Ac003	Biriyani
008	700	100	19-feb-2019	C007	Ac005	
009	10000	600	21-march-2010	C009	Ac008	Masala dosa
010	20	600	21-april-2012	C006	Ac005	

Table: orders

Agent_code	Agent_name	Working_area	commision	Phone_no	country
Ac001	Ramesh	Bangalore	.15	0331234567	India
Ac002	Dinesh	Bangalore	.25	0331234568	
Ac003	Suresh	Mumbai	.35	0331234569	London
Ac004	Kamlesh	New jersey	.68	0331234564	
Ac005	Kartik	Chennai	.73	0331234563	India

Table: Agent

1. Consider the following table Agent(AGENT_CODE, AGENT_NAME, WORKING_AREA ,|COMMISSION , PHONE_NO, COUNTRY) and Orders(ORD_NUM, ORD_AMOUNT, ADVANCE_AMOUNT, ORD_DATE,CUST_CODE,AGENT_CODE,ORD_DESCRIPTION)
   a. Find ord_num, ord_amount, ord_date, cust_code and agent_code from the table Orders working_area of Agent table must be Bangalore.
   b. Retrive ord_num, ord_amount, cust_code and agent_code from the table orders where the agent_code of orders table must be the same agent_code of agents table and agent_name of agents table must be Ramesh.

Lab Exercise

Salesman_id	Name	City	commission
si123@06	Lakshmi	Kolkata	.5
si123@09	Ganesh	London	.6
si123@90	Dinesh	London	.3
si123@10	Joseph	Chennai	.6
si123@19	Mahesh	Hyderabad	.65
si123@26	Paul Adam	London	.1
si123@67	Rahul	Delhi	.4

Table: salesman

ord_no	Purch_amt	Ord_date	Customer_id	Salesman_id
123	600	20-aug-2010	003cd	si123@19
576	750	20-feb-2018	004cd	si123@19
579	800	20-may-20120	004cd	si123@26
600	60000	20-jan-2021	006cd	si123@10
700	745	26-jan-2021	007cd	si123@09
800	860	29-jan-2019	007cd	si123@26

Table: orders

2. Consider the tables salesman(salesman_id, name ,city ,commission) and Orders(ord_no, purch_amt, ord_date, customer_id, salesman_id)

a. Display all the orders from the orders table issued by the salesman 'Paul Adam'.

b. Display all the orders for the salesman who belongs to the city London