

MySQL

1

Topics

➤ Subqueries

- Subqueries with IN, EXISTS
- Subqueries restrictions
- Nested subqueries
- ANY/ALL clause
- Correlated subqueries

➤ Views

- Creating
- Altering
- Dropping
- Renaming
- Manipulating views

What is a Query?

Query

- A database "**query**" is basically a "**question in a predefined format**" that you ask the database.
- The **results** of the query is the **information** that is **returned** by the **DBMS**.
- Queries are constructed using **SQL** (structured query language).

Subquery

- A *subquery* is a query within a query. (nesting)
- Subqueries enable you to write queries that select data rows for criteria that are actually developed while the query is executing at *run time*.
- A **SELECT** statement within another statement.
- A **SELECT** statement that is most often used as a part of another **SELECT** statement, but could also be used with an **INSERT**, **UPDATE**, or **DELETE** and other statements.
- Subqueries are used in order to achieve very **complex searches** and **complex reports**, as well as for **various optimizations**.
- **Results** of **one query** can be used **in another** SQL statement.
[Subquery is useful if more than one tables are involved.]

Types of Subquery

- There are three basic types of subqueries.
 1. Subqueries that operate on lists by use of the **IN** operator or with a comparison operator modified by the **ANY** or **ALL** optional keywords. These subqueries can return a **group of values**, but the values must be from a single column of a table.
 2. Subqueries that use an unmodified **comparison operator** (**=, <, >, <>**), these subqueries must return only a **single, scalar value**.
 3. Subqueries that use the **EXISTS** operator to test the **existence** of data rows satisfying specified criteria.
 4. Correlated Subqueries - A *correlated subquery* is one where the inner query depends on values provided by the outer query.
- This means the inner query is executed repeatedly, **once for each row that might be selected by the outer query.**

Subquery - GuideLines

- A subquery must be enclosed in **parentheses**.
- Use **single-row** operators with single-row subqueries, and use **multiple-row** operators with multiple-row subqueries.
- If a subquery (inner query) returns a **null value** to the outer query, the outer query will not return any rows when using certain comparison operators in a WHERE clause.

Subquery

➤ A subquery may return

- a scalar (single value, e.g. `count()`, `sum()`, etc)
- a single column
- a single row
- a table

➤ Operations

- Comparison operator can be used on scalar (e.g., `'='`, `'>'`, `'<'`)
- `IN` or `NOT IN` for single row or column
- `EXISTS` or `NOT EXIST` to test for empty set

Example - 1

- A subquery is called an **inner query** & the query that contains the subquery is called an **outer query**.
- A subquery is evaluated in an first “**inner**” then “**outer**” manner
- Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	35	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00

Example - 1

Select *select_list*
From *table*
Where *expr operator*

(Select *select_list*
From *table*);

- Find all customers whose salary is greater than 4500.

=> **SELECT ***
FROM CUSTOMERS
WHERE ID IN (SELECT ID
FROM CUSTOMERS
WHERE SALARY > 4500) ;

- The subquery (inner query) executes once before the main query (outer query) executes.
- The main query (outer query) use the subquery result.

Example - 2

- The following query returns the customer who has the maximum payment.

```
=> SELECT customerNumber,  
      checkNumber, amount  
      FROM payments  
      WHERE amount = (  
        SELECT MAX(amount)  
        FROM payments  
      );
```

	customerNumber	checkNumber	amount
▶	141	JE105477	120166.58

Subquery with EXISTS

- When a subquery is used with EXISTS or NOT EXISTS operator, a subquery returns a Boolean value of TRUE or FALSE. The subquery acts as an existence check.

⇒ **SELECT**
elastname, efirstname
FROM employee
WHERE EXISTS
(SELECT * FROM department
WHERE emp_pid = dep_emp_pid)
;

elastname	efirstname
-----	-----
Joyner	Suzanne
Zhu	Waiman
Bock	Douglas

Subquery with ANY/ALL

- ▶ ANY and ALL keywords are used with a WHERE or HAVING clause.
- ▶ ANY and ALL operate on subqueries that return multiple values.
- ▶ ANY returns true if any of the subquery values meet the condition.
- ▶ ALL returns true if all of the subquery values meet the condition.
- ▶ The ALL and ANY keywords can modify a comparison operator to allow an outer query to accept multiple values from a subquery.

The general ANY syntax is:

```
=> SELECT column-names  
FROM table-name  
WHERE column-name operator ANY  
      (SELECT column-name  
        FROM table-name  
        WHERE condition)
```

The general ALL syntax is:

```
=> SELECT column-names  
FROM table-name  
WHERE column-name operator ALL  
      (SELECT column-name  
        FROM table-name  
        WHERE condition)
```

Example - 1

- The ALL keyword modifies the greater than comparison operator to mean greater than all values.

```
=> SELECT emp_last_name "Last Name",  
       emp_first_name "First Name",  
       emp_salary "Salary"  
FROM employee  
WHERE emp_salary > ALL  
  
      (SELECT emp_salary  
FROM employee  
WHERE emp_dpt_number = 7);
```

Example - 2

- The ANY keyword is not as restrictive as the ALL keyword.
- When used with the greater than comparison operator, "> ANY" means greater than some value.

```
=> SELECT emp_last_name "Last Name",  
       emp_first_name "First Name",  
       emp_salary "Salary"  
FROM employee  
WHERE emp_salary > ANY  
      (SELECT emp_salary FROM employee  
       WHERE emp_salary > 30000);
```

Example - 1

- To find customer who has not ordered any products as follows:

```
=> SELECT customername  
      FROM customers  
      WHERE customerNumber NOT IN(  
        SELECT DISTINCT customernumber  
        FROM orders  
      );
```


Example - 2

```
mysql> SELECT * FROM students;
```

sid	sname	address	marks	classid
1	Raj	Mumbai	80	1
2	Prem	Bandra	85	2
3	Aryan	Bandra	82	2

```
mysql> SELECT * FROM CLASSES;
```

classid	cname
1	C1
2	C2

Example - 3


```
mysql> SELECT sname, marks  
-> FROM students  
-> WHERE classid IN (SELECT classid  
-> FROM classes  
-> WHERE cname = 'C2');
```


sname	marks
Prem	85
Aryan	82

Example - 4

List products with order quantities greater than 100.

```
=> SELECT ProductName
      FROM Product
      WHERE Id IN (SELECT ProductId
                   FROM OrderItem
                   WHERE Quantity > 100)
```

PRODUCT	
Id	
ProductName	
SupplierId	
UnitPrice	
Package	
IsDiscontinued	



ORDERITEM	
Id	
OrderId	
ProductId	
UnitPrice	
Quantity	

Example – 5 (Correlated Subquery)

List all customers with their total number of orders

```
=> SELECT FirstName, LastName,  
      OrderCount = (SELECT COUNT(O.Id)  
                    FROM [Order] O WHERE O.CustomerId = C.Id)  
      FROM Customer C
```

This is a **correlated subquery** because the subquery references the enclosing query (i.e. the C.Id in the WHERE clause).

ORDER	CUSTOMER
Id 	Id 
OrderDate	FirstName
OrderNumber	LastName
CustomerId	City
TotalAmount	Country
	Phone

Subquery Restrictions

- Refer: <https://dev.mysql.com/doc/mysql-reslimits-excerpt/5.1/en/subquery-restrictions.html>

Views

- A **database view** is known as a “**virtual table**” that allows you to query the data in it.
- Views can be effective copies of base tables.
- Views can have column names and expressions.
- You can use any clauses in views.
- Views can be used in INSERT/UPDATE/DELETE.
- Views can contain expressions in the select list.
- Views can be views of views.

CREATE VIEW

- `CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition`
- `CREATE VIEW [Current Product List] AS
SELECT ProductID,ProductName
FROM Products
WHERE Discontinued=No`
- `SELECT * FROM [Current Product List]`

UPDATE VIEW

- CREATE OR REPLACE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
- CREATE OR REPLACE VIEW [Current Product List] AS
SELECT ProductID,ProductName,Category
FROM Products
WHERE Discontinued=No

DROP VIEW

- You can delete a view with the DROP VIEW command.
- DROP VIEW view_name

QUESTIONS?



THANK YOU