Joins and Subqueries

Joins are used to query data from two or more tables, based on a relationship between certain columns in these tables. There are different types of JOINing methods used in MySQL database, which are given below.

INNER JOIN

Probably the most common join operation MySQL supports is an inner join. It identifies and combines only matching rows which are stored in two or more related tables.

A join condition, which indicates how the tables are related, is added with the keywords ON or USING.

NOTE:

ON is used when the relationship column names are different.

USING is used when the relationship column names are same.

INNER JOIN with ON clause

Syntax

```
SELECT <column list> FROM tableA a INNER JOIN tableB b ON a.somecolumn = b.othercolumn;
```

Example

```
mysql> SELECT d.deptno, d.dname, e.empno, e.ename FROM dept d INNER JOIN emp e ON d.deptno = e
```

INNER JOIN with USING clause syntax

Syntax

```
SELECT * FROM tableA a INNER JOIN tableB b USING(columnname);
```

Example

```
SELECT d.deptno, d.dname, e.empno, e.ename FROM dept d INNER JOIN emp e USING (deptno);
```

OUTER JOIN

OUTER JOIN allows us to retrieve all values in a certain table regardless of whether these values are present in other tables.

The difference between inner and outer join is: An outer join can identify rows without a match in the joined table. When no match was found, MySQL sets the value of columns from the joined table to NULL.

LEFT OUTER JOIN

This type of join will display matching records from both tables and all unmatched records of the left table. Left table means which table name is listed on the left side of the JOIN keywords.

Syntax

```
SELECT <column list> FROM table1 a LEFT OUTER JOIN table2 b ON a.somecolumn = b.othercolumn;
```

Example

```
SELECT d.deptno, d.dname, e.empno, e.ename FROM dept d LEFT OUTER JOIN employees e
    ON d.deptno = e.deptnumber;
```

RIGHT OUTER JOIN

This type of join will display matching records from both tables and all unmatched records of the right table. Right table means which table name is listed on the right side of the JOIN keywords.

Syntax

```
SELECT <column list> FROM table1 a RIGHT OUTER JOIN table2 b ON a.somecolumn = b.othercolumn;
```

Example

```
SELECT d.deptno, d.dname, e.empno, e.ename FROM dept d RIGHT OUTER JOIN employees e
    ON d.deptno = e.deptnumber;
```

SELF JOIN

A self-join, also known as an inner join, is a structured query language (SQL) statement where a queried table is joined to itself. The self-join statement is necessary when two sets of data, within the same table, are compared.

Example Suppose you're tasked with writing a SQL query to retrieve a list of employees and their managers.

```
mysql> SELECT e.firstname AS ‘Employee FN’, e.lastname AS ‘Employee LN’,
    m.firstname AS ‘Manager FN’, m.lastname AS ‘Manager LN’
    FROM employees AS e LEFT OUTER JOIN employees AS m ON e.manager =m.id;
```

CROSS JOIN

The cross join operation retrieves data between two tables as a Cartesian product of set theory in mathematics. Each row will get multiplied by other rows. If one table has three rows and the second row has two rows, then the Cartesian of two table will be six.

Example

```
mysql> SELECT * FROM dept CROSS JOIN employee;
```

SubQueries

A subquery is a SQL query nested inside a larger query.

A subquery may occur in :

A SELECT clause
A FROM clause
A WHERE clause

In MySQL subquery can be nested inside a SELECT, INSERT, UPDATE, DELETE, SET, or DO statement or inside another subquery.

A subquery is usually added within the WHERE Clause of another SQL SELECT statement. You can use the comparison operators, such as $>$, $<$, or $=$. The comparison operator can also be a multiple-row operator, such as IN, ANY, SOME, or ALL. A subquery can be treated as an inner query, which is a SQL query placed as a part of another query called as outer query. The inner query executes first before its parent query so that the results of inner query can be passed to the outer query.

Syntax

```
SELECT select_list from outerTable where expr_operator(SELECT select_list from innerTable);
```

In the above statement the inner query will be parsed first and the result will be passed to the outer query.

Types of SubQueries

- Scalar SubQueries
- SubQueries with ANY,ALL,IN

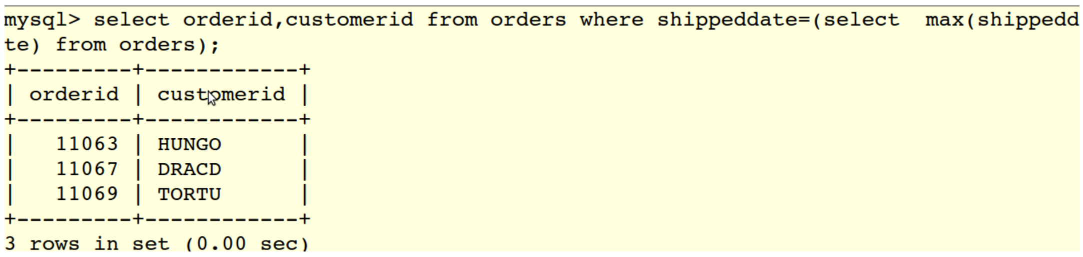
- SubQueries with EXISTS and NOT EXISTS
- Correlated SubQueries

Scalar SubQueries

- A scalar subquery is a subquery that returns exactly one column value from one row.
- If the subquery returns 0 rows then the value of scalar subquery expression in NULL.
- If the subquery returns more than one row then MySQL returns an error.
- The subquery can be used in either SELECT statement or WHERE clause.
- In either case, an aggregate function or an expression is normally used in the subquery.
- When the subquery is used in WHERE clause, a comparison operator is always used

Subquery in WHERE clause with an aggregate function

```
mysql>select OrderID, CustomerID from orders where
    ShippedDate = (select max(ShippedDate) from orders);
```



```
mysql> select orderid,customerid from orders where shippeddate=(select  max(shippedd
te) from orders);
+---------+------------+
| orderid | customerid |
+---------+------------+
|   11063 | HUNGO      |
|   11067 | DRACD      |
|   11069 | TORTU      |
+---------+------------+
3 rows in set (0.00 sec)
```

Figure 1: figure
Scalar SubQuery using Max function

NOTE:The above query returns data for all customers and their orders where the orders were shipped on the most recent recorded day.

```
mysql>select distinct ProductName, UnitPrice from products
    where UnitPrice>(select avg(UnitPrice) from products) order by UnitPrice desc;
```

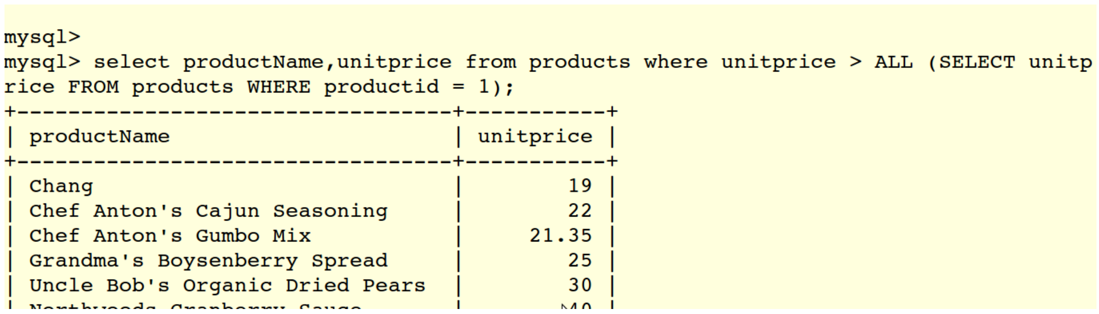
NOTE:This query returns all products whose unit price is greater than average unit price of all Products.

SubQueries using ALL

The ALL keyword specifies that the search condition is TRUE if the comparison is TRUE for every value that the subquery returns.

In the following example, the first condition tests whether each unitprice is greater than the unitprice of productid=1.

```
mysql>select productname,unitprice from products where unitprice > ALL
    (SELECT unitprice FROM products WHERE productid = 1);
```



```
mysql>
mysql> select productName,unitprice from products where unitprice > ALL (SELECT unitp
rice FROM products WHERE productid = 1);
+-----------------------------------+-----------+
| productName                       | unitprice |
+-----------------------------------+-----------+
| Chang                             |        19 |
| Chef Anton's Cajun Seasoning      |        22 |
| Chef Anton's Gumbo Mix            |     21.35 |
| Grandma's Boysenberry Spread      |        25 |
| Uncle Bob's Organic Dried Pears   |        30 |
```

Figure 2: figure
Subquery using ALL

Subquery using ANY

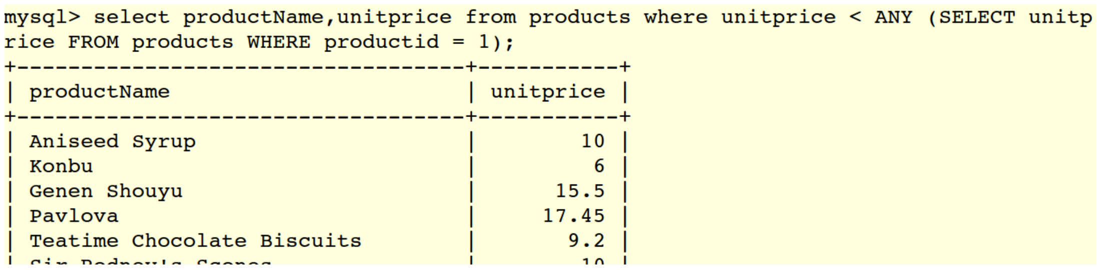
The ANY keyword denotes that the search condition is TRUE if the comparison is TRUE for at least one of the values that is returned. If the subquery returns no value, the search condition is FALSE. The SOME keyword is a synonym for ANY.

```
mysql>select productName,unitprice from products
    where unitprice < ANY (SELECT unitprice FROM products WHERE productid = 1);
```

Subquery using EXISTS

The MySQL EXISTS condition is used in combination with a subquery and is considered "to be met" if the subquery returns at least one row. It can be used in a SELECT, INSERT, UPDATE, or DELETE statement.

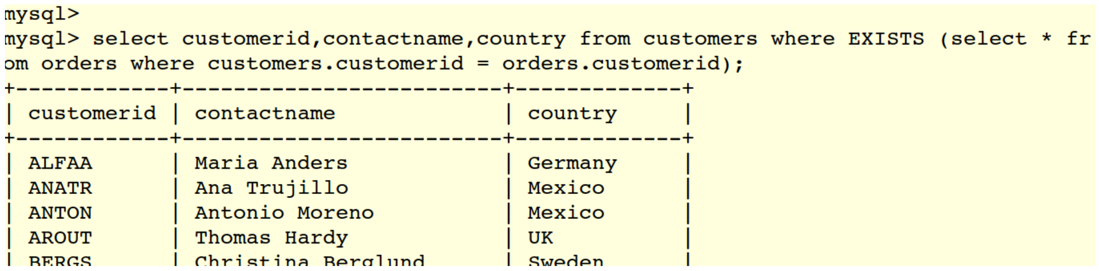
NOTE:SQL statements that use the EXISTS Condition in MySQL are very inefficient since the sub-query is RE-RUN for EVERY row in the outer query's table.



```
mysql> select productName,unitprice from products where unitprice < ANY (SELECT unitp
rice FROM products WHERE productid = 1);
+------------------------------------+-----------+
| productName                        | unitprice |
+------------------------------------+-----------+
| Aniseed Syrup                      |        10 |
| Konbu                              |         6 |
| Genen Shouyu                       |      15.5 |
| Pavlova                            |     17.45 |
| Teatime Chocolate Biscuits         |       9.2 |
| Sir Rodney's Scones                |        10 |
```

Figure 3: figure
SubQuery using ANY

```
mysql>select * from customers where EXISTS (select * from order_details
    where customers.customer_id = orders.customer_id);
```



```
mysql>
mysql> select customerid,contactname,country from customers where EXISTS (select * fr
om orders where customers.customerid = orders.customerid);
+------------+-------------------------+-------------+
| customerid | contactname             | country     |
+------------+-------------------------+-------------+
| ALFAA      | Maria Anders            | Germany     |
| ANATR      | Ana Trujillo            | Mexico      |
| ANTON      | Antonio Moreno          | Mexico      |
| AROUT      | Thomas Hardy            | UK          |
| BERGS      | Christina Berglund      | Sweden      |
```

Figure 4: figure
Exists Condition

NOTE:The above example will return all records from the customers table where there is at least one record in the order_details table with the matching customer_id.

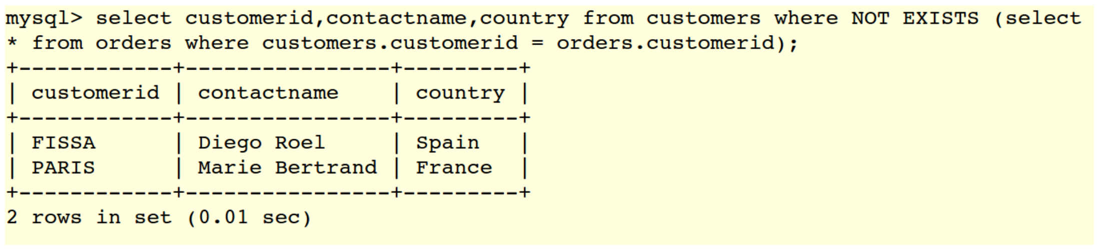
Subquery using NOT EXISTS

```
select * from customers where NOT EXISTS (select * from order_details
    where customers.customer_id = orders.customer_id);
```

NOTE:The above example will return all records from the customers table when there are no records in the order_details table for the given customer_id.

Subquery using EXISTS and INSERT

```
INSERT into contacts (contact\_id, contact\_name)
```



```
mysql> select customerid,contactname,country from customers where NOT EXISTS (select
* from orders where customers.customerid = orders.customerid);
+------------+----------------+---------+
| customerid | contactname    | country |
+------------+----------------+---------+
| FISSA      | Diego Roel     | Spain   |
| PARIS      | Marie Bertrand | France  |
+------------+----------------+---------+
2 rows in set (0.01 sec)
```

Figure 5: figure
Not Exists Condition

```
    select supplier\_id, supplier\_name from suppliers
    where EXISTS (select * from orders
    where suppliers.supplier\_id = orders.supplier\_id);
```

NOTE:The above example will insert records into contacts table if supplier_id of suppliers and orders table matches.

Subquery using EXISTS and DELETE

```
DELETE from suppliers where EXISTS
    (select * from orders where suppliers.supplier_id = orders.supplier_id);
```

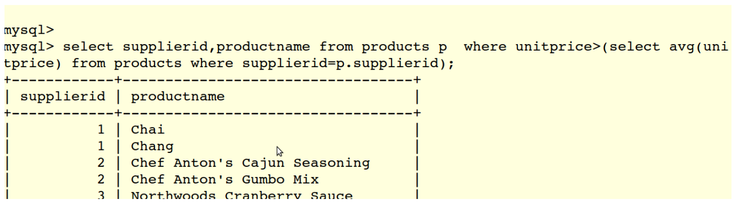
NOTE:The above example will delete records from suppliers table if supplier_id of suppliers and orders table matches.

Correlated SubQueries

A correlated subquery is a subquery that contains a reference to a table (in the parent query) that also appears in the outer query.
MySQL evaluates from inside to outside.

```
mysql> select supplierid, productname from products p
    where unitprice > (select AVG(unitprice) from products where supplierid = p.supplierid);
```

Displaying all employees whose salary is greater than the average salary of their own department.



```
mysql>
mysql> select supplierid,productname from products p  where unitprice>(select avg(uni
tprice) from products where supplierid=p.supplierid);
+------------+---------------------------------+
| supplierid | productname                     |
+------------+---------------------------------+
|          1 | Chai                            |
|          1 | Chang                           |
|          2 | Chef Anton's Cajun Seasoning    |
|          2 | Chef Anton's Gumbo Mix          |
|          3 | Northwoods Cranberry Sauce      |
```

Figure 6: figure
Correlated SubQuery