Name: Vishal Shashikant Salvi.

UID: 2019230069

Class: SE Comps

Batch: C

Experiment No 7

Aim: Practicing Joins and Subqueries

Theory:

Joins

Joins help retrieving data from two or more database tables. The tables are mutually related using primary and foreign keys.

Types of joins

Cross Join

Cross JOIN is a simplest form of JOINs which matches each row from one database table to all rows of another.

In other words it gives us combinations of each row of first table with all records in second table.

Syntax:

```
SELECT * FROM T1 CROSS JOIN T2
```

Inner Join

The inner JOIN is used to return rows from both tables that satisfy the given condition.

Syntax:

```
SELECT <column_list>
FROM T1, T2
WHERE <condition>
```

Left Join

The LEFT JOIN returns all the rows from the table on the left even if no matching rows have been found in the table on the right. Where no matches have been found in the table on the right, NULL is returned.

Syntax:

SELECT <column_list>

FROM T1 LEFT JOIN T2

ON <join_condition>

Right Join

RIGHT JOIN is obviously the opposite of LEFT JOIN. The RIGHT JOIN returns all the columns from the table on the right even if no matching rows have been found in the table on the left. Where no matches have been found in the table on the left, NULL is returned.

Syntax:

SELECT <column_list>

FROM T1 RIGHT JOIN T2

ON <join_condition>

7 Page Assignment Join

1. Find out the products which have been sold to 'Ivan Bayross'.

Sol :

```
mysql> select p.description as products
-> from product_master p
-> inner join sales_order_details sod
-> on sod.product_no = p.product_no
-> inner join sales_order so
-> on sod.order_no = so.order_no
-> inner join client_master cm
-> on so.client_no = cm.client_no
-> where cm.name = "Ivan Bayross" ;
```

```
+----------------+
| products       |
 +---------------+
| 1.44 Floppies |
| CD Drive      |
| 540 HDD       |
| Monitors      |
| Mouse         |
 +---------------+
 5    rows in set (0.02 sec)
```

2.Find out the products and their quantities that will have to be delivered in the current month.

Sol :

```
mysql> select p.description , sod.qty_ordered
    -> from product_master p
    -> inner join sales_order_details sod
    -> on p.product_no = sod.product_no
    -> inner join sales_order so
    -> on so.order_no = sod.order_no
    -> where month(dely_date) = 1;
+---------------+-------------+
| description | qty_ordered |
+---------------+-------------+
| 1.44 Floppies          | 4 |
| CD Drive               | 2 |
| 540 HDD                | 2 |
| 1.44 Floppies          | 10 |
+---------------+-------------+
 4    rows in set (0.00 sec)
```

3.Find the product_no and description of constantly sold i.e. rapidly moving products.

Sol : select distinct p.product_no, p.description
from product_master p, sales_order_details d
where p.product_no=d.product_no;

4.Find the names of clients who have purchased 'CD Drive'.

Sol : select name
from client_master c, product_master p, sales_order_details d, sales_order s
where c.client_no=s.client_no and d.order_no=s.order_no and p.product_no=d.product_no and p.description="CD Drive";

5.List the product_no and order_no of customers having qty_ordered less than 5 from sales_order_details table for the product '1.44 Floppies'.

Sol :

```
mysql> select sod.product_no , sod.order_no
-> from sales_order_details sod
-> inner join product_master
-> on product_master.product_no = sod.product_no
-> where sod.qty_ordered < 5 and product_master.description = "1.44 Floppies" ;
+------------+----------+
| product_no | order_no |
+------------+----------+
| P00001     | O19001 |
+------------+----------+
1    row in set (0.00 sec)
```

6.Find the products and their quantities for the orders placed by ‘Ivan Bayross’ and ‘Vandana Saitwal’.

Sol :

```
mysql> select p.description as products, sum(sod.qty_ordered) as quantity from product_master p inner join sales_order_details sod on sod.product_no = p.product_no inner join sales_order so on sod.order_no = so.order_no inner join client_master cm on so.client_no = cm.client_no where cm.name = "Ivan Bayross" or cm.name = "vandana Saitwal" group by p.description;
+---------------+----------+
| products      | quantity |
+---------------+----------+
| 1.44 Floppies | 14 |
| 540 HDD       | 2 |
| CD Drive      | 2 |
| Monitors      | 2 |
| Mouse         | 1 |
+---------------+----------+
5 rows in set (0.00 sec)
```

7.Find the products and their quantities for the orders placed by client_no ‘C00001’ and ‘C00002’.

Sol :

```
select p.product_no, d.qty_ordered
from client_master c, product_master p, sales_order s, sales_order_details d
where c.client_no=s.client_no and s.order_no=d.order_no and d.product_no=p.product_no and c.client_no in ("C00001","C00002");
```

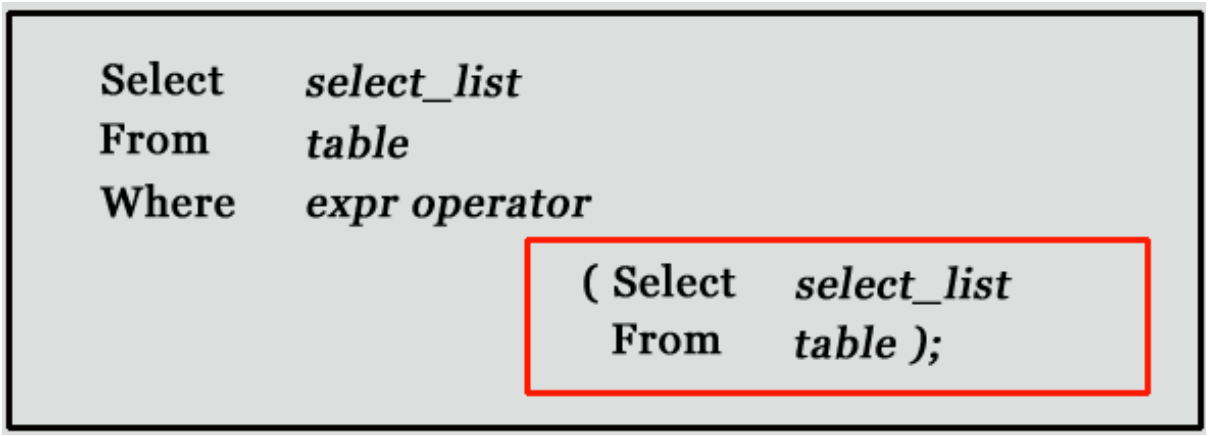
MySQL Subquery

A MySQL subquery is a query nested within another query such as SELECT, INSERT, UPDATE or DELETE. In addition, a MySQL subquery can be nested inside another subquery.

A MySQL subquery is called an inner query while the query that contains the subquery is called an outer query. The inner query executes first before its outer query so that the results of the inner query can be passed to the outer query.

A subquery can be used anywhere that expression is used and must be closed in parentheses.

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 return a scalar (a single value), a single row, a single column, or a table (one or more rows of one or more columns). These are called scalar, column, row, and table subqueries.

7 Page Assignment Subqueries

1. Find the product_no and description of non-moving products i.e. products not being sold.

Sol :

```
mysql> select product_no , description
    -> from product_master
    -> where product_no not in
    -> (select product_no from sales_order_details);

+------------+---------------+
| product_no | description|
+------------+---------------+
| P07865      | 1.22 Floppies |
| P08865      | 1.22 Drive    |
+------------+---------------+

2    rows in set (0.00 sec)
```

2.Find customer name, address1, address2, city and pin code for the client who has placed order no 'O19001'.

Sol :

```
mysql> select name , city , pincode
    -> from client_master
    -> where client_no in
    -> (select client_no from sales_order
    -> where order_no = 'O19001');
+--------------------+-----------+------------+
| name               | city      |   pincode |
+--------------------+-----------+------------+
 | Ivan Bayross      | Bombay | 400054   |
+--------------------+-----------+-----------+
    1    row in set (0.00 sec)
```

3.Find the client names who have placed orders before the month of May'96.

Sol :

```
select name from client_master where client_no in (select client_no from
sales_order where order_date < "1996-05-01");
```

```
+-----------------+
|      name       |
+-----------------+
| Ivan Bayross    |
| Vandana Saitwal |
| Pramada Jasgute |
+-----------------+

2    rows in set (0.00 sec)
```

4.Find out if the product '1.44 Drive' has been ordered by any client and print the client_no, name to whom it was sold.

Sol :

```
mysql> select client_no,name
    -> from client_master
    -> where client_no in (
    -> select client_no from sales_order
    -> where order_no in (
    -> select order_no from sales_order_details
    -> where product_no in (
    -> select product_no from product_master
    -> where description = '1.44 Drive')));
```

```
+--------  +------------ +
| client_no | name    |
+--------  +------------ +
|C0004      | Basu Navindgi   |
| C00005   | Ravi Sreedharan |
+-----------+--------------------+

2 rows in set (0.00 sec)
```

5.Find the names of clients who have placed orders worth Rs. 10000 or more.

Sol :

```
mysql> select name from client_master
    -> where client_no in
    -> (select client_no from sales_order where order_no in (select
order_no from sales_order_details where (qty_ordered*product_rate)
>= 10000));

+-----------------+
| name            |
+-----------------+
| Ivan Bayross    |
| Pramada Jasgute |
+-----------------+
 2 rows in set (0.01 sec)
```

Conclusion:

Thus, we studied different Joins as well as studied how to implement Subqueries in MySQL. Joins and subqueries are both used to combine data from different tables into a single result. They share many similarities and differences. Subqueries can be used to return either a scalar (single) value or a row set; whereas, joins are used to return rows.