

SSD Monsoon 2024

SQL Commands- SubQueries, UNION, EXISTS, INTERSECT, JOINs
(INNER, LEFT, RIGHT, CROSS), Common Table Expressions

SubQueries

SubQueries are most commonly in the FROM statement as a table to query from

When using it in the WHERE or HAVING statement, the subquery has to produce a single value table if you are using comparison operators such as =, <, <=, >, >=, !=.

Multi-value subquery tables will result in an error since it would try to compare a single field to every row in a column or every cell within table at once.

```
SELECT *  
FROM facebook  
WHERE num_of_friends =  
  (SELECT num_of_connections  
   FROM linkedin)
```

```
SELECT *  
FROM facebook  
WHERE num_of_friends IN  
  (SELECT num_of_connections  
   FROM linkedin)
```

Common Table Expressions (CTEs)

Subqueries can also be written in a **WITH** statement instead of in the main query.

Write **WITH** and then name the resulting table from the subquery and then in parentheses write the subquery.

Write a query that uses that subquery by referring to the name given to it

```
SELECT *  
FROM (SELECT State,SUM(num_of_friends)  
      FROM facebook  
      GROUP BY state)
```

```
WITH subQ AS (  
    SELECT State, SUM(num_of_friends)  
    FROM facebook  
    GROUP BY state)  
  
SELECT * FROM subQ
```

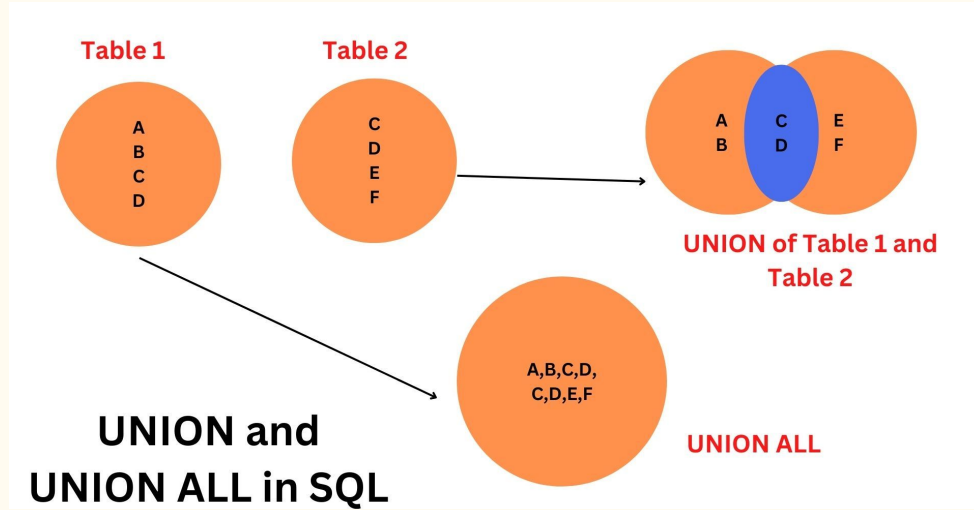
UNION

UNION combines distinct (default) results from multiple SELECT statements.

Each SELECT must have the same number of columns

The columns must have the same data types.

The columns in each table must be in the same order.



```
SELECT 'Customer' AS Type, ContactName, City,
Country
FROM Customers
UNION
SELECT 'Supplier', ContactName, City, Country
FROM Suppliers;
```

EXISTS

EXISTS is a conditional operator that checks if rows are returned by a subquery.

Returning TRUE when matches are found, it filters data with precision.

Commonly utilized in conjunction with WHERE clauses for targeted data retrieval.

```
SELECT fname, lname
FROM Customers
WHERE EXISTS (SELECT *
              FROM Orders
              WHERE Customers.customer_id =
Orders.c_id);
```

```
SELECT lname, fname
FROM Customers
WHERE NOT EXISTS (SELECT *
                  FROM Orders
                  WHERE Customers.customer_id =
Orders.c_id);
```

INTERSECT

Both SELECT statements must match in column count and data types.

MySQL database does not support the INTERSECT operator.

DISTINCT allows only such pairs of customer_id and name to exist which are unique and different from the others.

INTERSECT Operator using DISTINCT and INNER JOIN

```
SELECT DISTINCT customers.customer_id,  
customers.name  
FROM customers  
INNER JOIN premium_customers ON  
customers.customer_id=  
premium_customers.customer_id;
```

INTERSECT Operator using IN and Subquery

```
SELECT customer_id, name  
FROM customers  
WHERE customer_id IN (  
SELECT customer_id  
FROM premium_customers  
);
```

INNER JOIN

INNER JOIN retrieves rows that have matching values in both tables.

It's essential for combining related records across different datasets.

The result set includes only those rows where a link exists.

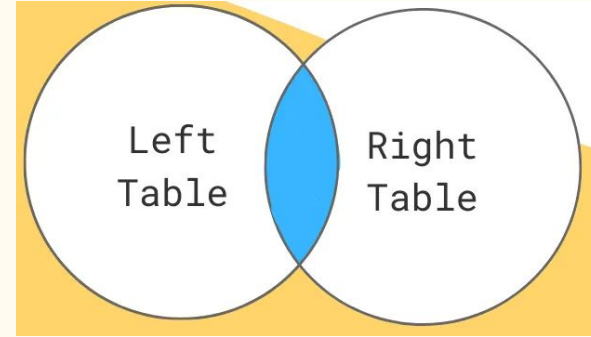


Table 1

1		
2		

Table 2

1		
3		
4		

Inner Join

1				

LEFT JOIN

LEFT JOIN, or LEFT OUTER JOIN, retrieves all records from the left table.

It includes matched rows from the right table; unmatched records appear as NULL.

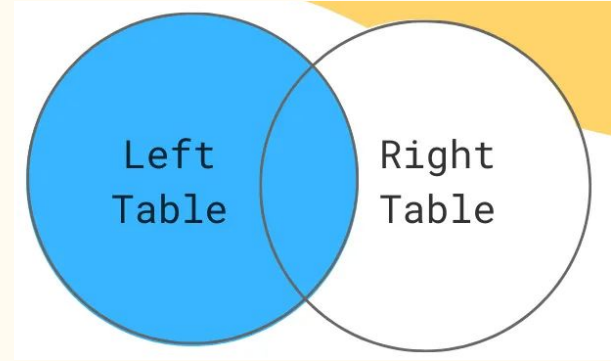


Table 1 ●

1		
2		

Table 2 ●

1		
3		
4		

Left Join ●●

1				
2				

RIGHT JOIN

RIGHT JOIN, or RIGHT OUTER JOIN, is similar to LEFT JOIN but focuses on the right table.

It ensures all right-side records are included, regardless of matches in the left table.

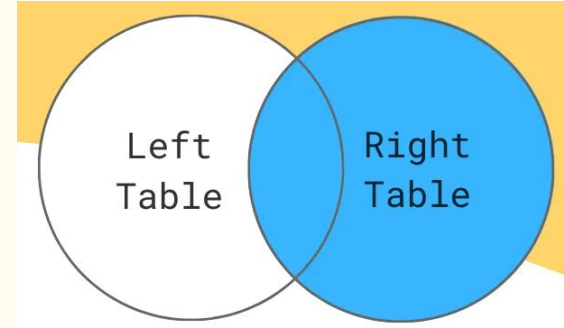


Table 1 ●

1		
2		

Table 2 ●


1		
3		
4		

Outer Join ●●


1				
2				
3				
4				

CROSS JOIN

Cartesian product of two tables, pairing each row from one with all from the other.

Table 1 


1		
2		

Table 2 

1		
3		
4		

Union  + 

1		
2		
1		
3		
4		

Table 1 

1		
2		

Table 2 

1		
3		
4		

Cross Join 

1			1		
1			3		
1			4		
2			1		
2			3		
2			4		

References

<https://learnsql.com/blog/learn-and-practice-sql-joins/>

<https://dataschool.com/how-to-teach-people-sql/how-sql-subqueries-work/>

https://www.w3schools.com/sql/sql_union.asp

<https://www.geeksforgeeks.org/sql-exists/>

<https://www.geeksforgeeks.org/mysql-intersect-operator/>

<https://www.atlassian.com/data/sql/sql-join-types-explained-visually>