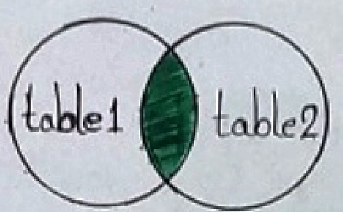
INNER JOIN

The INNER JOIN keyword selects records that have matching values in both tables.

INNER JOIN



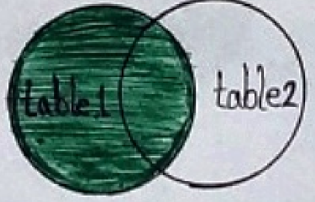
Example :-

```
SELECT column_name(s)
FROM table 1
INNER JOIN table 2
ON table1. column_name = table2.column_name;
```

LEFT JOIN

The LEFT JOIN keyword returns all records from the left table (table 1), and the matching records (if any) from the right table (table 2).

LEFT JOIN



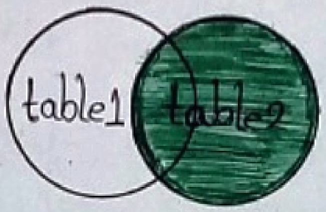
Example :-

```
SELECT column_name (s)
FROM table 1
LEFT JOIN table 2
ON table1.column_name = table2. column_name ;
```

RIGHT JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records (if any) from the left table (table 1).

RIGHT JOIN



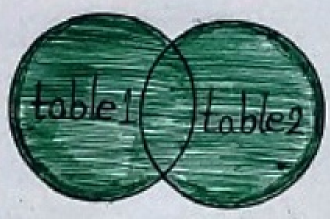
Example :-

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table 2
ON table1.column_name = table2.column_name;
```

CROSS JOIN

The CROSS JOIN keyword returns all records from both tables (table 1 and table 2).

CROSS JOIN



Example :-

```
SELECT column_name(s)
FROM table 1
CROSS JOIN table2;
```

SELF JOIN

A self Join is a regular join, but the table is joined with itself.

Example :-

```
SELECT column_name (s)
FROM table1 T1 , table1.T2
WHERE condition ;
```

UNION Operator

SQL joins allow you to combine two datasets side-by-side, but UNION allows you to stack one dataset on top of the other. Put differently, UNION allows you to write two separate SELECT statements, and to have the results of one statement display in the same table as the results from the other statement.

Example :-

-
  ```
  SELECT column_name (s)  FROM table 1
  UNION
  SELECT column_name (s)  FROM table 2 ;
  ```

-
  ```
  SELECT column_name (s)  FROM table 1
  UNION ALL
  SELECT column_name (s)  FROM table 2 ;
  ```

IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.
The IN operator is a shorthand for multiple OR conditions.

Example :-

-
  ```
  SELECT * FROM Sales
  WHERE country IN ("India", "Nepal", "UK");
  ```

-
  ```
  SELECT * FROM sales
  WHERE country NOT IN ("India", "Nepal", "UK");
  ```

-
  ```
  SELECT * FROM sales
  WHERE country IN (SELECT country FROM suppliers);
  ```

EXISTS Operator

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns TRUE if the subquery returns one or more records.

Example :-

```
SELECT column_name (s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

ANY and ALL Operator

The ANY and ALL operator allow you to perform a comparison between a single column value and a range of other values.

ANY Operator

- It returns a boolean value as a result.
- It returns TRUE if ANY of the subquery values meet the condition.

ANY means that the condition will be true if the operation is true for any of the values in the range.

Example:-

```
SELECT ProductName FROM sales
WHERE ProductID = ANY
   (SELECT ProductID FROM OrderDetails
    WHERE Quantity > 99 ) ;
```

ALL Operator

- It returns a boolean value as a result.
- It returns TRUE if ALL of the subquery values meet the condition.
- It is used with SELECT, WHERE and HAVING statements.

ALL means that the condition will be true only if the operation is true for all values in the range.

Example:-

-
  ```
  SELECT ALL ProductName
  FROM Sales
  WHERE TRUE ;
  ```

-
  ```
  SELECT ProductName FROM sales
  WHERE ProductID = ALL
    (SELECT ProductID FROM OrderDetails
     WHERE Quantity = 10) ;
  ```

INSERT INTO SELECT

The INSERT INTO SELECT statement copies data from one table and inserts it into anothertable.
The INSERT INTO SELECT statement requires that the data types in source and target tables matches.

The existing records in the target table are unaffected.

Example:-

-
  ```
  INSERT INTO table2
  SELECT * FROM table1
  WHERE condition ;
  ```

-
  ```
  INSERT INTO table2(column1, column2, column3, ...)
  SELECT column1, column2, column3, ...
  FROM table1
  WHERE condition ;
  ```

INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

It is possible to write the INSERT INTO statement in two ways.

- Specify both the column names and the values to be inserted.

```
INSERT INTO table_name(column1, column2, column3, ...)
VALUES (value1, value2, value3, ...) ;
```

- If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows.

```
INSERT INTO table_name
VALUES (value1, value2, value3,...) ;
```

IFNULL() Function

IFNULL(). function lets you return an alternative value if an expression is NULL.
The example below returns 0 if the value is NULL.

-
  ```
  SELECT contactname,
     IFNULL (bizphone, homephone) AS phone
  FROM contacts ;
  ```

-
  ```
  SELECT name,
     IFNULL (officephone , mobilephone) AS contact
  FROM employee ;
  ```