SQL Aggregate Function

SQL is excellent at aggregating data the way you might in a pivot table in Excel. You will use aggregate functions all the time, so it's important to get comfortable with them. The functions themselves are the same ones you will find in Excel or any other analytics program.

- COUNT counts how many rows are in a particular column.
- SUM adds together all the values in a particular column.
- MIN and MAX return the lowest and highest values in a particular column, respectively.
- AVG calculates the average of a group of selected values.

Example :-

```
SELECT COUNT(*)
  FROM Sales ;
```

Example :-

```
SELECT COUNT (column_name)
FROM table_name
WHERE condition ;
```

Example :-

```
SELECT SUM (column_name)
FROM table_name
WHERE condition ;
```

Example :-
```
SELECT MIN (column_name)
FROM table_name
WHERE condition;
```

Example :-
```
SELECT MAX (column_name)
FROM table_name
WHERE condition ;
```

Example :-
```
SELECT AVG (column_name)
FROM table_name
WHERE condition ;
```

The SQL GROUP BY clause

GROUP BY allows you to separate data into groups, which can be aggregated independently of one another.

```
SELECT year,
       COUNT (*) AS count
  FROM sales
GROUP BY year ;
```

Multiple column

```
SELECT year,
       month,
       COUNT (*) AS count
  FROM  sales
GROUP BY year, month;
```

GROUP BY Column numbers

```
SELECT year,
       month,
       COUNT(*) AS count
  FROM sales
GROUP BY 1,2 ;
```

Using GROUP BY with ORDER BY

```
SELECT year,
       month,
       COUNT(*) AS count
  FROM sales
GROUP BY year, month
ORDER BY month, year ;
```

Using GROUP BY with LIMIT

```
SELECT column_name,
 FROM table_name
WHERE condition
GROUP BY column_name
LIMIT number ;
```

HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

Example :-
```
SELECT column_name (s)
FROM table_name
WHERE condition
GROUP BY column_name (s)
HAVING condition
ORDER BY column_name (s) ;
```

-
```
SELECT year,
       month,
       MAX (high) AS month_high
  FROM sales
 GROUP BY year, month
HAVING MAX (high) > 400
 ORDER BY year, month ;
```

The SQL CASE statement

The CASE statement is SQL's way of handling if/ then logic. The CASE statement is followed by at least one pair of WHEN and THEN statements - SQL's equivalent of IF/THEN in Excel. Because of this pairing, you might be tempted to call this SQL CASE WHEN, but CASE is the accepted term.

Every CASE statement must end with the END statement. The ELSE statement is optional, and provides a way to capture values not specified in the WHEN / THEN statement. CASE is easiest to understand in the context of an example.

Syntax

```
CASE
    WHEN condition1 THEN result 1
    WHEN condition 2  THEN result2
    WHEN conditionN  THEN resultN
    ELSE result
END ;
```

Example :-
```
SELECT orderID , Quantity,
   CASE
      WHEN Quantity > 30 THEN "The quantity is greater than 30"
      WHEN quantity= 30 THEN "The quantity is 30"
      ELSE "The quantity is under 30"
   END AS QuantityText
   FROM sales ;
```

SQL DISTINCT

You'll occasionally want to look at only the unique values in a particular column. You can do this using SELECT DISTINCT syntax.

Example :-

-
  ```
  SELECT DISTINCT month
      FROM Sales ;
  ```

-
  ```
  SELECT DISTINCT year, month
      FROM Sales ;
  ```

Using DISTINCT in aggregations

```
SELECT COUNT(DISTINCT month) AS unique_months
    FROM Sales;
```

MySQL JOINS

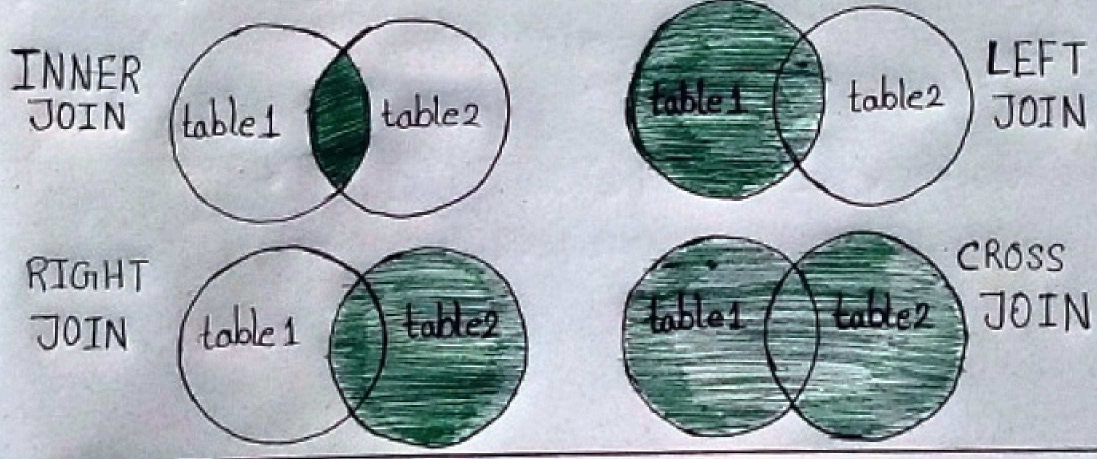
A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Example :-

```
SELECT *
 FROM benn.college_football_players players
JOIN benn. college_football_teams teams
   ON teams. school_name = players .school_name
```

Supported Types of JOINS in MySQL

- INNER JOIN: Returns records that have matching values in both tables.
- LEFT JOIN : Returns all records from the left table, and the matched records from the right table.
- RIGHT JOIN : Returns all records from the right table, and the matched records from the left table.
- CROSS JOIN: Returns all records from both tables.


INNER JOIN
table1
table2
LEFT JOIN
table1
table2
RIGHT JOIN
table1
table2
CROSS JOIN
table1
table2