Sql Commands

SELECT

SELECT statements are used to fetch data from a database. Every query will begin with SELECT.

```
SELECT column_name FROM table_name;
```

SELECT DISTINCT

SELECT DISTINCT specifies that the statement is going to be a query that returns unique values in the specified column(s).

```
SELECT DISTINCT column_name FROM table_name;
```

AS

AS is a keyword in SQL that allows you to rename a column or table using an alias.

```
SELECT column_name AS 'Alias' FROM table_name;
```

LIMIT

LIMIT is a clause that lets you specify the maximum number of rows the result set will have.

```
SELECT column_name(s) FROM table_name LIMIT number;
```

WHERE

WHERE is a clause that indicates you want to filter the result set to include only rows where the following condition is true.

Select names from Person where gender = 'Female'

IS NULL / IS NOT NULL

IS NULL and IS NOT NULL are operators used with the WHERE clause to test for empty values.

```
SELECT column_name(s)
```

```
FROM table_name

WHERE column_name IS NULL;
```

BETWEEN

The BETWEEN operator is used to filter the result set within a certain range. The values can be numbers, text or dates. The border values are inclusive

```
SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value_1 AND value_2;
```

LIKE

LIKE is a special operator used with the WHERE clause to search for a specific pattern in a column. ***LIKE*** *matches* ***case-insensitive*** *records*

```
SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern;
```

AND

AND is an operator that combines two conditions. Both conditions must be true for the row to be included in the result set.

```
SELECT column_name(s) FROM table_name WHERE column_1 = value_1 AND column_2 = value_2;

Select gender from Person
```

OR

OR is an operator that filters the result set to only include rows where either condition is true.

```
SELECT column_name FROM table_name WHERE column_name = value_1 OR column_name = value_2;
```

ORDER BY

ORDER BY is a clause that indicates you want to sort the result set by a particular column either alphabetically or numerically.

```
SELECT column_name FROM table_name ORDER BY column_name ASC | DESC;
```

GROUP BY

GROUP BY is a clause in SQL that is only used with aggregate functions. It is used in collaboration with the SELECT statement to arrange identical data into groups.

```
SELECT column_name, COUNT(*)

FROM table_name

GROUP BY column_name;
```

```
Note : how can we group on data ranges ?
```

HAVING

HAVING was added to SQL because the WHERE keyword could not be used with aggregate functions.

```
SELECT column_name, COUNT(*)

FROM table_name

GROUP BY column_name

HAVING COUNT(*) > value;
```

CASE

CASE statements are used to create different outputs (usually in the SELECT statement). It is SQL's way of handling if-then logic.

```
SELECT column_name,

  CASE

    WHEN condition1 THEN 'Result_1'

    WHEN condition2 THEN 'Result_2'
```

```
    ELSE 'Result_3'
  END
FROM table_name;
```

Aggregate Functions

COUNT()

COUNT() is a function that takes the name of a column as an argument and counts the number of rows where the column is not NULL.

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

SUM

SUM() is a function that takes the name of a column as an argument and returns the sum of all the values in that column.

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

MAX()

MAX() is a function that takes the name of a column as an argument and returns the largest value in that column.

```
SELECT MAX(column_name) FROM table_name;
```

MIN()

MIN() is a function that takes the name of a column as an argument and returns the smallest value in that column.

```
SELECT MIN(column_name) FROM table_name;
```

AVG()

AVG() is an aggregate function that returns the average value for a numeric column.

```
SELECT AVG(column_name) FROM table_name;
```

ROUND()

ROUND() is a function that takes a column name and an integer as an argument. It rounds the values in the column to the number of decimal places specified by the integer.

```
SELECT ROUND(column_name, integer) FROM table_name;
```

CREATE TABLE

CREATE TABLE creates a new table in the database. It allows you to specify the name of the table and the name of each column in the table.

```
CREATE TABLE table_name (
  column_1 datatype,
  column_2 datatype,
  column_3 datatype
);
```

Change/delete/insert table values

UPDATE

UPDATE statements allow you to edit rows in a table.

```
UPDATE table_name SET some_column = some_value WHERE some_column = some_value;
```

```
UPDATE movies SET movie_title = "NewMovieTitle" WHERE movie_year = "1977"
```

DELETE

DELETE statements are used to remove rows from a table.

```
DELETE FROM table_name WHERE some_column = some_value;
```

INSERT

INSERT statements are used to add a new row to a table.

```
INSERT INTO table_name (column_1, column_2, column_3) VALUES (value_1, 'value_2', value_3);
```

```
INSERT INTO PERSONS (name,age,gender) VALUES (‘YASH’, 23, ‘m’);
```

ALTER TABLE:

ALTER TABLE lets you add columns to a table in a database.

```
ALTER TABLE table_name ADD column_name datatype;
```

```
ALTER TABLE table_name DROP COLUMN column_name; table_name.
```

Joins

can we filter two table before applying join ??

INNER JOIN

An inner join will combine rows from different tables if the join condition is true.

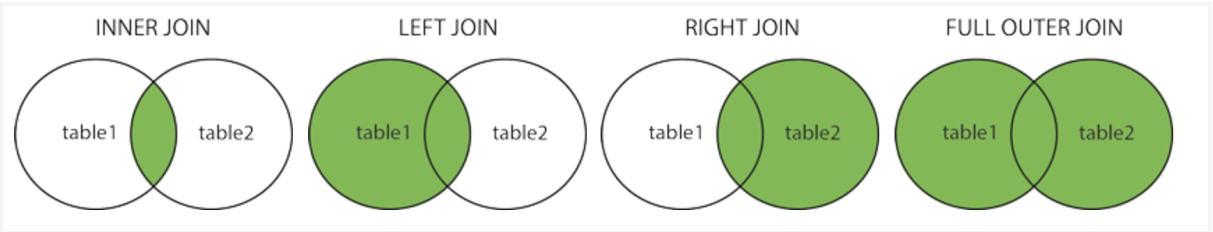
```
SELECT column_name(s)
FROM table_1
JOIN table_2
ON table_1.column_name = table_2.column_name;
```

OUTER JOIN

An outer join will combine rows from different tables even if the join condition is not met. Every row in the left table is returned in the result set, and if the join condition is not met, then NULL values are used to fill in the columns from the right table.

LEFT OUTER JOIN

```
SELECT column_name(s) FROM table_1
LEFT JOIN table_2
ON table_1.column_name = table_2.column_name;
```



Views

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;

How a Sql query is structured

1. **SELECT** [**DISTINCT**] Attribute_List **FROM** R1,R2….RM
2. [**WHERE** condition]
3. [**GROUP BY** (Attributes)[**HAVING** condition]]
4. [**ORDER BY**(Attributes)[**DESC**]];
5. **LIMIT N**

How query is executed in the Backend

- FROM : ***SQL CHECKS IF THE TABLE EXISTS***
- WHERE : **Filters**
- GROUP BY : **segregates the data**
- HAVING :
- SELECT
- DISTINCT
- ORDER BY

Database Indexing

Imagine a textbook index.
What is the way to find the occurrence of a particular word in a book ?

Difference between top and limit performance wise

If you are using SQL Server use TOP if you are using MySQL or Postgres use Limit!

Ref : Stackoverflow

Resources

Explain Keyword

MYSQL Architecture

DBMS Indexing

How is data stored in sql database

How do Joins Work

Order Of Execution of an SQL Query

Count(*) vs Count(1) - SQL Server

Inner join or subquery? Which is optimized? Which to use when?

MySQL Trigger

Temporary Tables

What are projection and selection?

Offset

Partitions

SQL update from one Table to another based on a ID match

What is Reverse Indexing

Views

Temporary Tables