What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL keywords are NOT case sensitive

RDBMS

- RDBMS stands for Relational Database Management System.
- RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

Some of The Most Important SQL Commands

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

SELECT Syntax

```
SELECT column1, column2, ...
FROM table_name;
```

- Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

```
SELECT * FROM table_name;
```

SELECT DISTINCT Syntax

- Distinct also count null values but only one if present.

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

```
SELECT DISTINCT * FROM table_name;
```

WHERE Syntax

- It is used to extract only those records that fulfill a specified condition.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

ORDER BY Syntax

- The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword. To sort each column manually write ASC|DESC after each columns.

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

INSERT INTO Syntax

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

- 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

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

What is a NULL Value?

A field with a NULL value is a field with no value.
If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

- Use the IS NULL and IS NOT NULL operators to check the NULL values.

IS NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

IS NOT NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

UPDATE Syntax

- The UPDATE statement is used to modify the existing records in a table.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

DELETE Syntax

- The DELETE statement is used to delete existing records in a table. If no condition is given then all the records get deleted.

```
DELETE FROM table_name WHERE condition;
```

SELECT TOP Syntax

- It is used to specify the number of records to return.MySQL supports the LIMIT clause to select a limited number of records

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

- The LIMIT clause accepts an optional second parameter.When two parameters are specified, the first parameter specifies the offset of the first row to return i.e. the starting point, whereas the second parameter specifies the maximum number of rows to return. The offset of the initial row is 0 (not 1).

```
SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number1, number2;
```

MIN()|MAX() Syntax

```
SELECT MIN|MAX(column_name)
FROM table_name
WHERE condition;
```

COUNT() Syntax

- count(*) gives total no of rows.
- count(column_name) gives total no of Non-NULL values.
- distinct column_name gives all the distinct values in the column even null if present.

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

AVG() Syntax

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

SUM() Syntax

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

LIKE Syntax

- search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator. A wildcard character is used to substitute one or more characters in a string.:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

- 'a%' - Finds any values that start with "a"
- '%a' - Finds any values that end with "a"
- '%or%' - Finds any values that have "or" in any position
- '_r%' - Finds any values that have "r" in the second position
- 'a_%' - Finds any values that start with "a" and are at least 2 characters in length
- 'a__%' - Finds any values that start with "a" and are at least 3 characters in length
- 'a%o' - Finds any values that start with "a" and ends with "o"

IN Syntax

- allows us to specify multiple values in a WHERE clause.
- a shorthand for multiple OR conditions.

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

- If want to exclude use NOT IN instead of IN.

BETWEEN Syntax

- Selects values within a given range. The values can be numbers, text, or dates.
- It is inclusive: begin and end values are included.
- Use NOT BETWEEN if want to select out of this range.

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Alias

- used to give a table, or a column in a table, a temporary name.
- used to make column names more readable.
- only exists for the duration of that query.
- created with the AS keyword.

Alias Column Syntax

```
SELECT column_name AS alias_name
FROM table_name;
```

Alias Table Syntax

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

JOIN

It combine rows from two or more tables, based on a related column between them.

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left - table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

UNION

The UNION operator is used to combine the result-set of two or more SELECT statements.

- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types

- The columns in every SELECT statement must also be in the same order

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

UNION ALL

The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

GROUP BY

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".
The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s);
```

HAVING

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

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition;
```

EXISTS

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.

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

CASE

- The CASE expression goes through conditions and returns a value when the first condition is met (like an if-then-else statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause.
- If there is no ELSE part and no conditions are true, it returns NULL.

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 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 OrderDetails;
```

> Any, All
> null
> count(*) count the row if all the values in that row is null. check it.
> difference between using and on in joins

CREATE DATABASE

- Used to create new SQL database.

```
CREATE DATABASE databasename;
```

DROP DATABASE

- Drop / Delete an existing SQL database.

```
DROP DATABASE databasename;
```

BACKUP DATABASE

- create a full back up of an existing SQL database.

```
BACKUP DATABASE databasename
TO DISK = 'filepath';
```

BACKUP WITH DIFFERENTIAL

- A differential back up only backs up the parts of the database that have changed since the last full database backup.

```
BACKUP DATABASE databasename
TO DISK = 'filepath'
WITH DIFFERENTIAL;
```

CREATE TABLE

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

or

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

DROP TABLE

- Delete the table along with schema.

  ```
  DROP TABLE table_name;
  ```

TRUNCATE TABLE

- Delete the data inside a table, but not the table itself.

```
TRUNCATE TABLE table_name;
```

ALTER TABLE - ADD Column

- Add column in the table.

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

ALTER TABLE - DROP COLUMN

- Delete the column from the table.

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

ALTER TABLE - MODIFY COLUMN

- change the data type of a column in a table.

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```