SQL

What is SQL?

SQL (Structured Query Language) is a programming language used to interact with database.

CRUD is an acronym for CREATE, READ(SELECT), UPDATE, and DELETE statements in SQL

SQL v/s NoSQL

Relational Database Non-Relational Database

1. SQL database

   NoSQL database

2. Data stored in tables

   Data stored are either key-value pairs, document-based, graph databases or wide column stores

3. These databases have fixed or static or predefined schema

   They have dynamic schema

4. Low performance with huge volumes of data
   Easily work with huge volumes of data

5. Eg: PostgreSQL, MySQL, MS SQL Server
   Eg: MongoDB, Cassandra, Hbase

SQL Commands

There are mainly 3 types of SQL commands:

• DDL (Data Definition Language): create, alter, and drop

• DML (Data Manipulation Language): select, insert, update and delete

• DCL (Data Control Language): grant and revoke permission to users

What is Database?

Database is a system that allow users to store and organize data.

Data Types

•Data type of a column defines what value the column can store in table

•Defined while creating tables in database

•Data types mainly classified into three categories most used

String: char, varchar, etc

Numeric: int, float, bool, etc

Date and time: date, datetime, etc

Commonly Used data types in SQL:

• int: used for the integer value

• float: used to specify a decimal point number

• bool: used to specify Boolean values true and false

• char: fixed length string that can contain numbers, letters, and special characters

• varchar: variable length string that can contain numbers, letters, and special characters

• date: date format YYYY-MM-DD

• datetime: date & time combination, format is YYYY-MM-DD hh:mm:ss

SELECT Statement

The SELECT statement is used to select data from a database.

• Syntax

SELECT column_name FROM table_name;

To select all the fields available in the table

• Syntax

SELECT * FROM table_name;

To select distinct/unique fields available in the table

• Syntax

SELECT DISTINCT Column_name FROM table_name;

WHERE Clause

The WHERE clause is used to filter records.

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

• Syntax

SELECT column_name FROM table_name

WHERE conditions;

Operators In SQL

The SQL reserved words and characters are called operators, which are

used with a WHERE clause in a SQL query

Most used operators:

1. **Arithmetic operators** : arithmetic operations on numeric values

Example: Addition (+), Subtraction (-), Multiplication (*), Division (/), Modulus (%)

2. **Comparison operators**: compare two different data of SQL table

• Example: Equal (=), Not Equal (!=), Greater Than (>), Greater Than Equals to (>=)

3**. Logical operators**: perform the Boolean operations

• Example: ALL, IN, BETWEEN, LIKE, AND, OR, NOT, ANY

4. **Bitwise operators:** perform the bit operations on the Integer values

• Example: Bitwise AND (&), Bitwise OR(|)

Like operator

Pattern	Meaning
‘a%’	Match strings that start with ‘a’
‘%a’	Match strings with end with ‘a’
‘a%t’	Match strings that contain the start with ‘a’ and end with ‘t’.
‘%wow%’	Match strings that contain the substring ‘wow’ in them at any position.
‘_wow%’	Match strings that contain the substring ‘wow’ in them at the second position.
‘_a%’	Match strings that contain ‘a’ at the second position.
‘a_ _%’	Match strings that start with ‘a and contain at least 2 more characters.

LIMIT Clause

The LIMIT clause is used to set an upper limit on the number of tuples returned by SQL.

Example: below code will return 5 rows of data

```
SELECT column_name FROM table_name
LIMIT 5;
```

ORDER BY Clause

The ORDER BY is used to sort the result-set in ascending (ASC) or descending order (DESC).

Example: below code will sort the output data by column name in ascending order

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

Functions In SQL

Functions in SQL are the database objects that contains a set of SQL statements to perform a specific task. A function accepts input parameters, perform actions, and then return the result.

Types of Function:

1. **System Defined Function** : these are built-in functions

• Example: rand(), round(), upper(), lower(), count(), sum(), avg(), max(), etc

2. **User-Defined Function** : Once you define a function, you can call it in the same way as the built-in functions

String functions are used to perform an operation on input string and return an output string

- UPPER() converts the value of a field to uppercase
- LOWER() converts the value of a field to lowercase
- LENGTH() returns the length of the value in a text field
- SUBSTRING() extracts a substring from a string
- NOW() returns the current system date and time
- FORMAT() used to set the format of a field
- CONCAT() adds two or more strings together
- REPLACE() Replaces all occurrences of a substring within a string, with a new substring
- TRIM() removes leading and trailing spaces (or other specified characters) from a string

Aggregate function performs a calculation on multiple values and returns a single value.

And Aggregate function are often used with GROUP BY & SELECT statement

- COUNT() returns number of values
- SUM() returns sum of all values
- AVG() returns average value
- MAX() returns maximum value
- MIN() returns minimum value
- ROUND() Rounds a number to a specified number of decimal places

GROUP BY Statement

The GROUP BY statement group rows that have the same values into summary rows.

It is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns

• Syntax

```
SELECT column_name(s)

FROM table_name

GROUP BY column_name(s);
```

• Example

```
SELECT mode, SUM(amount) AS total

FROM payment

GROUP BY mode
```

HAVING Clause

The HAVING clause is used to apply a filter on the result of GROUP BY based on the specified condition.

The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause

Syntax

```
SELECT column_name(s)

FROM table_name

WHERE condition(s)

GROUP BY column_name(s)

HAVING condition(s)
```

• **Example**

```
SELECT mode, COUNT(amount) AS total

FROM payment

GROUP BY mode

HAVING COUNT(amount) >= 3

ORDER BY total DESC
```

TIMESTAMP

The TIMESTAMP data type is used for values that contain both date and time parts

• TIME contains only time, format HH:MI:SS

• DATE contains on date, format YYYY-MM-DD

• YEAR contains on year, format YYYY or YY

• TIMESTAMP contains date and time, format YYYY-MM-DD HH:MI:SS

• TIMESTAMPTZ contains date, time and time zone

TIMESTAMP functions/operators

Below are the TIMESTAMP functions and operators in SQL:

• **SHOW** TIMEZONE

• **SELECT** NOW()

• **SELECT** TIMEOFDAY()

• **SELECT** CURRENT_TIME

• **SELECT** CURRENT_DATE

EXTRACT Function

The EXTRACT() function extracts a part from a given date value.

Syntax: SELECT EXTRACT(MONTH FROM date_field) FROM Table

- YEAR
- QUARTER
- MONTH
- WEEK
- DAY
- HOUR
- MINUTE
- DOW – day of week
- DOY – day of year

SQL JOIN

•JOIN means to combine something.

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

TYPES OF JOINS

• INNER JOIN

• LEFT JOIN

• RIGHT JOIN

• FULL JOIN

INNER JOIN

Returns records that have matching values in both tables

• **Syntax**

```
SELECT column_name(s)

FROM TableA

INNER JOIN TableB

ON TableA.col_name = TableB.col_name
```

• **Example**

```
SELECT *

FROM customer AS c

INNER JOIN payment AS p

ON c.customer_id = p.customer_id
```

LEFT JOIN

• Returns all records from the left table, and the matched records from the right table

• **Syntax**

```
SELECT column_name(s)

FROM TableA

LEFT JOIN TableB

ON TableA.col_name = TableB.col_name
```

• **Example**

```
SELECT *

FROM customer AS c

LEFT JOIN payment AS p

ON c.customer_id = p.customer_id
```

RIGHT JOIN

• Returns all records from the right table, and the matched records from the left table

• Syntax

```
SELECT column_name(s)
FROM TableA
RIGHT JOIN TableB
ON TableA.col_name = TableB.col_name
```

• Example

```
SELECT *
FROM customer AS c
RIGHT JOIN payment AS p
ON c.customer_id = p.customer_id
```

FULL JOIN

• Returns all records when there is a match in either left or right table

• Syntax

```
SELECT column_name(s)
FROM TableA
FULL OUTER JOIN TableB
ON TableA.col_name = TableB.col_name
```

• Example

```
SELECT *
FROM customer AS c
FULL OUTER JOIN payment AS p
ON c.customer_id = p.customer_id
```

SELF JOIN

• A self join is a regular join in which a table is joined to itself

• SELF Joins are powerful for comparing values in a column of rows with the same table

• Syntax

```
SELECT column_name(s)
FROM Table AS T1
JOIN Table AS T2
ON T1.col_name = T2.col_name
```

SELF JOIN example

• Find the name of respective managers for each of the employees?

UNION

The SQL UNION clause/operator is used to combine/concatenate the results of two or more SELECT statements without returning any duplicate rows and keeps unique records

To use this UNION clause, each SELECT statement must have

- The same number of columns selected and expressions
- The same data type and
- Have them in the same order

• Syntax

```
SELECT column_name(s) FROM TableA
UNION
SELECT column_name(s) FROM TableB
```

• Example

```
SELECT cust_name, cust_amount from custA
UNION
SELECT cust_name, cust_amount from custB
```

UNION ALL

In UNION ALL everything is same as UNION, it combines/concatenate two or more table but keeps all records, including duplicates

• **Syntax**

```
SELECT column_name(s) FROM TableA

UNION ALL

SELECT column_name(s) FROM TableB
```

• **Example**

```
SELECT cust_name, cust_amount from custA

UNION ALL

SELECT cust_name, cust_amount from custB
```

SUB QUERY

A Subquery or Inner query or a Nested query allows us to create complex query on the output of another query

• Sub query syntax involves two SELECT statements

• **Syntax**

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator
( SELECT column_name FROM table_name WHERE ... );
```

Primary and Foreign Keys:

Primary key (PK):

• A Primary key is a unique column we set in a table to easily identify and locate data in queries

• A table can have only one primary key, which should be unique and NOT NULL

Foreign keys (FK):

• A Foreign key is a column used to link two or more tables together

• A table can have any number of foreign keys, can contain duplicate and NULL values

Constraints

• Constraints are used to specify rules for data in a table

• This ensures the accuracy and reliability of the data in the table

• Constraints can be specified when the table is created with the CREATE TABLE statement, or

• after the table is created with the ALTER TABLE statement

• Syntax

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

Constraints

Commonly used constraints in SQL:

• NOT NULL - Ensures that a column cannot have a NULL value

• UNIQUE - Ensures that all values in a column are different

• PRIMARY KEY - A combination of a NOT NULL and UNIQUE

• FOREIGN KEY - Prevents actions that would destroy links between tables (used to link multiple tables together)

• CHECK - Ensures that the values in a column satisfies a specific condition

• DEFAULT - Sets a default value for a column if no value is specified

• CREATE INDEX - Used to create and retrieve data from the database very quickly

Create Table

The CREATE TABLE statement is used to create a new table in a database

- **Syntax**

```
CREATE TABLE table_name
(
column_name1 datatype constraint,
column_name2 datatype constraint,
column_name3 datatype constraint,
);
```

- **Example**

```
CREATE TABLE customer
(
CustID int8 PRIMARY KEY,
CustName varchar(50) NOT NULL,
Age int NOT NULL,
City char(50),
Salary numeric
);
```

Insert Values In Table

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

• **Syntax**

```
INSERT INTO TABLE_NAME
(column1, column2, column3,...columnN)
VALUES
(value1, value2, value3,...valueN);
```

• **Example**

```
INSERT INTO customer
(CustID, CustName, Age, City, Salary)
VALUES
(1, 'Sam', 26, 'Delhi', 9000),
(2, 'Ram', 19, 'Bangalore', 11000),
(3, 'Pam', 31, 'Mumbai', 6000),
(4, 'Jam', 42, 'Pune', 10000);
```

Update Values In Table

The UPDATE command is used to update existing rows in a table

• Syntax

```
UPDATE TABLE_NAME
SET “Column_name1” = ‘value1’, “Column_name2” = ‘value2’
WHERE “ID” = ‘value’
```

• Example

```
UPDATE customer
SET CustName = 'Xam', Age= 32
WHERE CustID = 4;
```

ALTER Table

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table

• ALTER TABLE - ADD Column Syntax

```
ALTER TABLE table_name
ADD COLUMN column_name ;
```

• ALTER TABLE - DROP COLUMN Syntax

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

• ALTER TABLE - ALTER/MODIFY COLUMN Syntax

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

ALTER Table Example

• ADD Column Syntax: Adding new ‘Gender’ column to customer table

```
ALTER TABLE customer
ADD COLUMN Gender varchar(10);
```

• ALTER/MODIFY COLUMN Syntax: changing Gender column data type from varchar(10) to char(10)

Delete Values In Table

The DELETE statement is used to delete existing records in a table

• Syntax

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

• Example

```
DELETE FROM customer
WHERE CustID = 3;
```

Drop & Truncate Table

The DROP TABLE command deletes a table in the database

• Syntax

```
DROP TABLE table_name;
```

The TRUNCATE TABLE command deletes the data inside a table, but not the table itself

• Syntax

```
TRUNCATE TABLE table_name;
```

CASE Expression

• The CASE expression goes through conditions and returns a value when the first condition is met (like if-then-else statement). 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.

• It also called CASE STATEMENT

CASE Statement Syntax

• **General CASE Syntax**

```
CASE
        WHEN condition1 THEN result1
        WHEN condition2 THEN result2
        WHEN conditionN THEN resultN
        ELSE other_result
END;
```

• Example:

```
SELECT customer_id, amount,
CASE
        WHEN amount > 100 THEN 'Expensive product'
        WHEN amount = 100 THEN 'Moderate product'
        ELSE 'Inexpensive product'
        END AS ProductStatus
FROM payment
```

Common Table Expression (CTE)

• A common table expression, or CTE, is a temporary named result set created from a simple SELECT statement that can be used in a subsequent SELECT statement

• We can define CTEs by adding a WITH clause directly before SELECT, INSERT, UPDATE, DELETE, or MERGE statement.

• The WITH clause can include one or more CTEs separated by commas

• Syntax

```
WITH my_cte AS (
                        SELECT a,b,c
                        FROM Table1 )
SELECT a,c
FROM my_cte
```

The name of this CTE is my_cte, and the CTE query is SELECT a,b,c FROM Table1. The CTE starts with the WITH keyword, after which you specify the name of your CTE, then the content of the query in parentheses.

The main query comes after the closing parenthesis and refers to the CTE. Here, the main query (also known as the outer query) is SELECT a,c FROM my_cte