Basic SQL

What is SQL?

- SQL stands for Structured Query Language.
- It is a standardized programming language used for managing and manipulating relational databases.
- SQL allows users to perform tasks such as querying data, inserting or updating records, and defining database structures.

What SQL can do?

- SQL can execute queries against a database.
- SQL can create and delete a database and tables in a database.
- SQL can create,modify or update, delete records in a database.
- SQL can create stored procedures,functions, views in a database.
- SQL can set permissions on tables,procedures,functions and views.

Advantages and Disadvantages of SQL:

Advantages:

1. SQL provides a standardized way to interact with databases, making it easier to work with different database systems.
2. It offers powerful querying capabilities for retrieving and manipulating data efficiently.
3. SQL supports data integrity constraints, ensuring data consistency and reliability.

Disadvantages:

1. SQL can be complex, especially for beginners, and may require a learning curve to master.
2. Some database systems may have proprietary extensions to SQL, leading to potential compatibility issues when migrating between systems.
3. SQL queries can sometimes be slow, especially when dealing with large datasets or complex queries.

What is data types in SQL? Why is it required?

- Each column in a database table is required to have a name and a data type. An SQL developer must decide what type of data that will be stored inside each column when creating a table.
- Data types in SQL define the type of data that can be stored in a column of a table.
- Data types are required to ensure data integrity, optimize storage space, and enable efficient querying and processing of data.

Different data types in SQL:

There are three main data types: string, numeric, and date and time.

1. **String Data Types:**

- VARCHAR(length): Variable-length character string with a maximum length specified by "length".
- CHAR(length): Fixed-length character string with a specified length.
- TEXT(size) : Holds a string with a maximum length of 65,535 bytes (MySQL)

2. **Numeric Data Types:**

- INT: Integer value (whole number).
- DECIMAL(precision, scale): Fixed-point number with a specified total number of digits (precision) and number of digits after the decimal point (scale).
- FLOAT(size, d) : A floating point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter.

3. **Date and Time Data Types:**

- DATE: Date value in the format YYYY-MM-DD.
- DATETIME: Date and time value in the format YYYY-MM-DD HH:MM:SS.
- TIMESTAMP : Format: YYYY-MM-DD hh:mm:ss

Each data type has specific attributes and storage requirements, and choosing the appropriate data type is essential for efficient data storage and retrieval in a database.

(Ref : https://www.w3schools.com/sql/sql_datatypes.asp)

Constraints

- Constraints in SQL are rules that are enforced on columns or tables to maintain data integrity and consistency within a database.
- They define limits or conditions on the data that can be stored in a database table.

Why constraints are used in SQL:

1. Data Integrity: Constraints ensure that data entered into a database meets certain criteria, preventing incorrect or inconsistent data from being stored.
2. Consistency: Constraints help maintain consistency across the database by enforcing rules and relationships between tables. For example, a FOREIGN KEY constraint ensures that a value in one table matches a value in another table, maintaining referential integrity.
3. Prevention of Data Anomalies: Constraints help prevent data anomalies such as insertion, deletion, or update anomalies by enforcing rules on the data.
4. Enforcement of Business Rules: Constraints can enforce business rules and requirements, ensuring that the data stored in the database adheres to specific criteria set by the organization.

5. Improved Data Quality: By enforcing constraints, SQL ensures that only valid and accurate data is stored in the database, leading to improved data quality and reliability.

Overall, Constraints play a critical role in ensuring the **accuracy**, **consistency**, and **reliability** of data stored in a database, making it an essential aspect of database design and management.

Different types of Constraints:

1. **PRIMARY KEY**: Ensures each row in a table is uniquely identified.A combination of a NOT NULL and UNIQUE. (Use: Primary Key as Constraint_Name)

2. **FOREIGN KEY**: Establishes a relationship between two tables and enforces referential integrity. Prevents actions that would destroy links between tables. (Use: Foreign Key Reference Ref_Table(Ref_Column) as Constraint_Name)

3. UNIQUE: Ensures that all values in a column are unique. (Use: Unique as Constraint_Name)

4. **NOT NULL**: Ensures that a column cannot contain NULL values. (Use: Not Null as Constraint_Name)

5. **CHECK**: Ensures that the values in a column satisfies a specific condition. Limits the range of values that a column can contain based on a specified condition. (Use: Check(Condition) as Constraint_Name)

6. **DEFAULT** - Sets a default value for a column if no value is specified. (Use: Default "Default Value" as Constraint_Name)

7. CREATE **INDEX** - Used to create and retrieve data from the database very quickly. (Syntax: Create Index index_name On table_name (column/columns);

General Syntax of adding Constraints:

The following SQL ensures that the "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created:

❖ **During Table Creation:**

❖

```
CREATE TABLE Persons (
   ID int Constraint_Name1,
   LastName varchar(255) Constraint_Name2,
   FirstName varchar(255) Constraint_Name3,
   Age int );
```

Or

```
CREATE TABLE Persons (
   ID int,
   LastName varchar(255),
    FirstName varchar(255),
    Age int
Constraint Constraint_Name (Column_Name));  (Can use single or multiple columns or constraints)
```

- **Alter or Modify :**
-

```
Alter Table Table_Name
Add Constraint Constraint_Name Constraint_Type (Column_Name);
Or
Alter Table Table_Name
Modify Column_Name datatype Constraint_Type ;
```

- **Drop Constraint :**

```
Alter Table Table_Name
Drop Constraint Constraint_Name;
```

Key in SQL:

- A key in a database is an attribute or a set of attributes that uniquely identifies a tuple (row) in a table.
- Keys play a crucial role in ensuring the integrity and reliability of a database by enforcing unique constraints on the data and establishing relationships between tables.

Types of Key:

1. Super Key -A Super key is a combination of columns that uniquely identifies any row within a relational database management system (RDBMS) table.
2. Candidate key - A candidate key is a minimal Super key, meaning it has no redundant attributes. In other words, it's the smallest set of attributes that can be used to uniquely identify a tuple (row) in the table.
3. Primary Key - A primary key is a unique identifier for each tuple in a table. There can only be one primary key in a table, and it cannot contain null values.
4. Alternate Key - An alternate key is a candidate key that is not used as the primary key.
5. Composite Key - A composite key is a primary key that is made up of two or more attributes. Composite keys are used when a single attribute is not sufficient to uniquely identify a tuple in a table.
6. Unique Key - A Unique Key is key that uniquely identify a record.
7. Foreign Key - A foreign key is a primary key from one table that is used to establish a relationship with another table.

(Ref : https://www.geeksforgeeks.org/types-of-keys-in-relational-model-candidate-super-primary-alternate-and-foreign/)

Operators in SQL:

- SQL operators are symbols or keywords that perform operations on one or more operands to produce a result.
- They are used to perform comparisons, arithmetic operations, logical operations, and more within SQL queries.
- Here's why they are used:

1. Data Manipulation: Operators allow you to manipulate and transform data within SQL queries. For example, arithmetic operators (+, -, *, /) enable mathematical calculations, while string operators (|| for concatenation) allow you to manipulate text data.
2. Data Filtering: Comparison operators (>, <, =, !=, etc.) are used to filter rows based on specific conditions, enabling the selection of relevant data from a database table.
3. Logical Operations: Logical operators (AND, OR, NOT) are used to combine conditions in SQL queries, allowing for complex criteria to be applied when retrieving data.

By using operators, you can perform a wide range of operations within SQL queries, enabling you to retrieve, manipulate, and filter data as needed for various data analysis and reporting tasks.

Diffrent types of Operators:

There are total 6 types of operators in SQL: Arithmetic, Bitwise, Comparison, Compound, Logical and String.

Arithmetic Operator:

+ : Addition, It is used to perform addition operation on the data items, items include either single column or multiple columns.

- : Subtraction, It is use to perform subtraction operation on the data items, items include either single column or multiple columns.

/ : Division, Computation of Division : R(x,y) div S(y) ,R & S two table, x,y columns.

* : Multiplication, It is use to perform multiplication of data items.

% : Modulus, It is use to get remainder when one data is divided by another.

Bitwise Operators:

&	: Bitwise AND
|	: Bitwise OR
^	: Bitwise exclusive OR

Comparison Operators:

=	: Equal to
>	: Greater than
<	: Less than
>=	: Greater than or equal to
<=	: Less than or equal to
<>	: Not equal to

Compound Operators:

+= : Add equals

-= : Subtract equals

*= : Multiply equals

/= : Divide equals

%= : Modulo equals

&= : Bitwise AND equals

^-= : Bitwise exclusive equals

|*= : Bitwise OR equals

Logical Operators:

ALL : TRUE if all of the subquery values meet the condition

AND : TRUE if all the conditions separated by AND is TRUE

ANY : TRUE if any of the subquery values meet the condition

BETWEEN : TRUE if the operand is within the range of comparisons (BETWEEN THIS AND THAT)

EXISTS : TRUE if the subquery returns one or more records

IN : TRUE if the operand is equal to one of a list of expressions (IN LIST)

LIKE : TRUE if the operand matches a pattern

NOT : Displays a record if the condition(s) is NOT TRUE

OR : TRUE if any of the conditions separated by OR is TRUE

SOME : TRUE if any of the subquery values meet the condition

I will learn String related operator later.

SQL CASE Expression :

- The Case Expression goes through conditions and return a value when the first conditions is met (like an if_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 will return null.

General Syntax:

```
SELECT *
CASE
   WHEN  Condition_1 THEN Result_1
    WHEN  Condition_2 THEN Result_2
      ELSE Result_3
END                  (if you want to alias the result , you can use END AS ‘Column_ Name’)
FROM Table_Name;
```