**SQL- Structured Query Language**

* SQL stands for Structured Query Language.
* SQL is a standard language for storing, accessing, manipulating and retrieving data in databases.
* We can use SQL in MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.
* SELECT \* FROM Customers;

**Database Tables**

* A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.

**The SQL SELECT Statement**

* The SELECT statement is used to select data from a database.
* The data returned is stored in a result table, called the result-set.
* SELECT column1, column2, ...  
  FROM table\_name;

**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.

**Note:** A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

## How to Test for NULL Values?

* It is not possible to test for NULL values with comparison operators, such as =, <, or <>.
* We will have to use the IS NULL and IS NOT NULL operators instead.

### **IS NULL Syntax**

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NULL;

### **IS NOT NULL Syntax**

SELECT column\_namesFROM table\_name  
WHERE column\_name IS NOT NULL;

## The SQL ORDER BY Keyword

* The ORDER BY keyword is used to sort the result-set in ascending or descending order.
* It sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

### ORDER BY Syntax

SELECT column1, column2, ...  
FROM table\_name  
ORDER BY column1, column2, ... ASC|DESC;

**The SQL SELECT TOP Clause**

The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact on performance.

**Note:** Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses ROWNUM.

**SQL Server / MS Access Syntax:**

SELECT TOP *number*|*percent* *column\_name(s)*  
FROM *table\_name*WHERE *condition*;

## The SQL MIN() and MAX() Functions

* The MIN() function returns the smallest value of the selected column.
* The MAX() function returns the largest value of the selected column.

### **MIN() Syntax**

SELECT MIN(column\_name)  
FROM table\_name  
WHERE condition;

### **MAX() Syntax**

SELECT MAX(column\_name)  
FROM table\_name  
WHERE condition;

**The SQL COUNT(), AVG() and SUM() Functions**

* The COUNT() function returns the number of rows that matches a specified criteria.
* The AVG() function returns the average value of a numeric column.
* The SUM() function returns the total sum of a numeric column.

### **COUNT() Syntax**

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;

## SQL Wildcard Characters

* A wildcard character is used to substitute one or more characters in a string.
* Wildcard characters are used with the [SQL LIKE](https://www.w3schools.com/sql/sql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.
* **Symbol** -\* ? [] ! # -

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Description** | **Example** |
| \* | Represents zero or more characters | bl\* finds bl, black, blue, and blob |
| ? | Represents a single character | h?t finds hot, hat, and hit |
| [] | Represents any single character within the brackets | h[oa]t finds hot and hat, but not hit |

## The SQL IN Operator

* The IN operator allows you to specify multiple values in a WHERE clause.
* The IN operator is a shorthand for multiple OR conditions.

### **IN Syntax**

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (value1, value2, ...);

or:

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name IN (*SELECT* STATEMENT);

## The SQL BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

### **BETWEEN Syntax**

SELECT column\_name(s)  
FROM table\_name  
WHERE column\_name BETWEEN value1 AND value2;

## SQL Aliases

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of the query.

### **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;

## SQL JOIN

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

## SQL INNER JOIN Keyword

The INNER JOIN keyword selects records that have matching values in both tables.

### **INNER JOIN Syntax**

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2ON table1.column\_name = table2.column\_name;



## SQL LEFT JOIN Keyword

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

### **LEFT JOIN Syntax**

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2ON table1.column\_name = table2.column\_name;

**Note:** In some databases LEFT JOIN is called LEFT OUTER JOIN.



## SQL RIGHT JOIN Keyword

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

### **RIGHT JOIN Syntax**

SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

**Note:** In some databases RIGHT JOIN is called RIGHT OUTER JOIN.



## SQL FULL OUTER JOIN Keyword

The FULL OUTER JOIN keyword return all records when there is a match in either left (table1) or right (table2) table records.

**Note:** FULL OUTER JOIN can potentially return very large result-sets!

**Tip:** FULL OUTER JOIN and FULL JOIN are the same.

### **FULL OUTER JOIN Syntax**

SELECT column\_name(s)  
FROM table1  
FULL OUTER JOIN table2ON table1.column\_name = table2.column\_nameWHERE condition;



## SQL Self JOIN

A self JOIN is a regular join, but the table is joined with itself.

### **Self JOIN Syntax**

SELECT column\_name(s)  
FROM table1 T1, table1 T2  
WHERE condition;

## The SQL UNION Operator

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

* Each SELECT statement within UNION must have the same number of columns
* The columns must also have similar data types
* The columns in each SELECT statement must also be in the same order

### **UNION Syntax**

SELECT column\_name(s) FROM table1  
UNION  
SELECT column\_name(s) FROM table2;

### **UNION ALL Syntax**

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;

**Note:** The column names in the result-set are usually equal to the column names in the first SELECT statement in the UNION.

**The SQL GROUP BY Statement**

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.

### **GROUP BY Syntax**

SELECT column\_name(s)  
FROM table\_name  
WHERE condition  
GROUP BY column\_name(s)ORDER BY column\_name(s);

## SQL UNIQUE Constraint

* The UNIQUE constraint ensures that all values in a column are different.
* Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
* A PRIMARY KEY constraint automatically has a UNIQUE constraint.
* However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

## SQL UNIQUE Constraint on CREATE TABLE

The following SQL creates a UNIQUE constraint on the "ID" column when the "Persons" table is created:

**SQL Server / Oracle / MS Access:**

CREATE TABLE Persons (  
    ID int NOT NULL UNIQUE,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int  
);

**MySQL:**

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    UNIQUE (ID)  
);

To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

**MySQL / SQL Server / Oracle / MS Access:**

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    CONSTRAINT UC\_Person UNIQUE (ID,LastName)  
);

## SQL CREATE VIEW Statement

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 functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

### **CREATE VIEW Syntax**

CREATE VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

**Note:** A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.

## SQL CREATE VIEW Examples

The following SQL creates a view that shows all customers from Brazil:

**Example**

CREATE VIEW [Brazil Customers] AS  
SELECT CustomerName, ContactName  
FROM Customers  
WHERE Country = "Brazil";

**SQL PRIMARY KEY Constraint**

* The PRIMARY KEY constraint uniquely identifies each record in a table.
* Primary keys must contain UNIQUE values, and cannot contain NULL values.
* A table can have only one primary key, which may consist of single or multiple fields.

**SQL PRIMARY KEY on CREATE TABLE**

The following SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

**MySQL:**

CREATE TABLE Persons (  
    ID int NOT NULL,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int,  
    PRIMARY KEY (ID)  
);

**SQL Server / Oracle / MS Access:**

CREATE TABLE Persons (  
    ID int NOT NULL PRIMARY KEY,  
    LastName varchar(255) NOT NULL,  
    FirstName varchar(255),  
    Age int  
);

## SQL FOREIGN KEY Constraint

A FOREIGN KEY is a key used to link two tables together.

A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

**What is Trigger In SQL?**

* A **trigger** is a special kind of stored procedure that automatically executes when an event occurs in the database server.
* DML **triggers** execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT , UPDATE , or DELETE statements on a table or view.

**The TRUNCATE TABLE statement** is used to remove all records from a table in **SQL Server**. It performs the same function as a DELETE statement without a WHERE clause.

**In SQL, the TRUNCATE TABLE statement** is a Data **Definition** Language (DDL) operation that marks the extents of a table for deallocation (empty for reuse). The result of this operation quickly removes all data from a table, typically bypassing a number of integrity enforcing mechanisms.

**What is difference between delete and truncate in SQL?**

**Delete -Return but truncate- Not Return**

The **DROP** command removes a table from the database. All the tables' rows, indexes and privileges will also be removed.

**DROP and TRUNCATE** are DDL commands, whereas **DELETE** is a DML command.

**DELETE** operations can be rolled back (undone), while **DROP and TRUNCATE** operations cannot be rolled back.

## The SQL DROP DATABASE Statement

* The DROP DATABASE statement is used to drop an existing SQL database.
* **Syntax**

DROP DATABASE databasename;

**What is Drop?**

SQL provides **DROP DATABASE statement** to allow you to remove existing databases.

The SQL **DROP DATABASE statement** deletes all tables, views, triggers, and other related objects inside a database, and also removes the database permanently.

**What is rollback in database?**

In **database** technologies, a **rollback** is an operation which returns the **database** to some previous state.

**Rollbacks** are important for **database** integrity, because they mean that the **database** can be restored to a clean copy even after erroneous operations are performed.

A transaction is a unit of work that is performed against a database. Transactions are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

A transaction is the propagation of one or more changes to the database. For example, if you are creating a record or updating a record or deleting a record from the table, then you are performing a transaction on that table. It is important to control these transactions to ensure the data integrity and to handle database errors.

Practically, you will club many SQL queries into a group and you will execute all of them together as a part of a transaction.

## Properties of Transactions

Transactions have the following four standard properties, usually referred to by the acronym **ACID**.

* **Atomicity** − ensures that all operations within the work unit are completed successfully. Otherwise, the transaction is aborted at the point of failure and all the previous operations are rolled back to their former state.
* **Consistency** − ensures that the database properly changes states upon a successfully committed transaction.
* **Isolation** − enables transactions to operate independently of and transparent to each other.
* **Durability** − ensures that the result or effect of a committed transaction persists in case of a system failure.

### **Transaction Control**

The following commands are used to control transactions.

* **COMMIT** − to save the changes.
* **ROLLBACK** − to roll back the changes.
* **SAVEPOINT** − creates points within the groups of transactions in which to ROLLBACK.
* **SET TRANSACTION** − Places a name on a transaction.

**A relational database**

is a set of formally described tables from which data can be accessed or reassembled in many different ways without having to reorganize the**database** tables. The standard user and application programming interface (API) of a **relational database** is the Structured Query Language (**SQL**).

**What is the difference between database and relational database?**

The key **difference** is that RDBMS (**relational database** management system) applications store data **in a** tabular form, while DBMS applications store data as files.

**In a** RDBMS, the tables will have an identifier called primary key. Data values will be stored **in the** form of tables.

**What is the MS Access and SQL database?**

**SQL** stands for Structured Query Language.

**Microsoft Access** is an application, that stores definition of work into a file (MDB/ACCDB) and that one can use to create and manage desktop **databases** and create forms on top of it to gather the data and reports to present them.

# **What is MongoDB?**

MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need.

* MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
* The document model maps to the objects in your application code, making data easy to work with
* Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data
* MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use

**Why MongoDB is called NoSQL?**

* The **NoSQL** database is a way of storing data in means other than the relational database.
* The **NoSQL** varieties are further categorized into key-value, column, graph, and document, to name a few.
* **MongoDB** is an open source software that falls under the "Document" category.

QA’s responsibility is retrieval operation and data base validation . It means QA is to retrieve data from data base and use these data to implement in the project .

Like I want to log in and I have 50 sets of user ID and password in the table of the data base, then Read these data(User ID and password) from the table, I can use these data to implement my project. This process is called Data Driven Testing.

For data validation, we use data base. Like Sign up.

QA’s responsibility is

* Is to confirm all data property store in data base.
* To confirm the real data, from backend, which is displayed in the UI level
* To ensure the matching data in the UI level which data comes from backend system

**How to create JDBC Connection?**

First of all I need a my SQL java connector jar file, then as you know java cannot communicate/talk directly to data base, that’s why we need driver class or I have to initialize the driver class as a interpreter or media .

Then I will create connection using driver manager class as a interpreter or media.

Then in order to create a connection

Then to fail a connection we have to/ I will invoke get /create connection method using driver manager class which will return connection statement interface.

Once I have a connection then have to invoke create statement method using the reference of connection interface

Then I will invoke execute query method using the reference of statement interface

Where I will write my query.

Once you have result set then I will iterate through one by one and store that into any data structure bases on result set.

Then I will use those data into my project based on my need.

First of all I have the driver as a media or interpreter then I will invoke get connection method using driver manager class which will return me get connection method then I have to invoke as a interpreter or media.

Then to fail a connection we have to/ I will invoke get /create connection method using driver manager class which will return connection statement interface.

Once I have a connection then have to invoke create statement method using the reference of connection interface

Then I will invoke execute query method using the reference of statement interface

Where I will write my query.

Once you have result set then I will iterate through one by one and store that into any data structure bases on result set.

Then I will use those data into my project based on my need.