**Manual Database Testing**

**Database**: A database is **an organized collection of structured information, or data, typically stored electronically in a computer system**. A database is usually controlled by a database management system (DBMS).

**SQL**: Structured Query Language

**Schema**: It is the Structure or Organization of Data.

**Tables**: Tables are **database objects that contain all the data in a database**. In tables, data is logically organized in a row-and-column format similar to a spreadsheet. Each row represents a unique record, and each column represents a field in the record.

**Popular Databases**: MySQL, Oracle, MongoDB etc.,

**CRUD Operations**:

Create, Read/Retrieve, Update & Delete.

Data Types:

**CHAR(SIZE):** A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1.

**VARCHAR(SIZE)**: A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum column length in characters - can be from 0 to 65535.

**TEXT(SIZE)**: Holds a string with a maximum length of 65,535 bytes.

**INT(SIZE)**: A medium integer. Signed range is from -2147483648 to 2147483647.

**DATE**: A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'

**DATETIME**: Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.

**TIMESTAMP**: Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.

**AUTO INCREMENT**: Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

**NOT NULL**: The NOT NULL constraint enforces a column to NOT accept NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

**DDL**: Data Definition Language actually consists of the SQL commands that can be used to define the database schema.

* **CREATE**: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).
* **DROP**: This command is used to delete objects from the database.
* **ALTER:**This is used to alter the structure of the database.
* **TRUNCATE:**This is used to remove all records from a table, including all spaces allocated for the records are removed.

**DQL**: **DQL**statements are used for performing queries on the data within schema objects.

* **SELECT:**It is used to retrieve data from the database.

**DML**: The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

* **INSERT**: It is used to insert data into a table.
* **UPDATE:** It is used to update existing data within a table.
* **DELETE**: It is used to delete records from a database table.

**Creating a Database**: create database <databaseName>

**Changing to an existing Database**: use <databaseName>

**Creating a Table**: create table <tableName> (column1 <datatype>, column2 <datatype>, column3 <datatype>);

**Show the list of Databases**: show databases

**Show the list of Tables**: Change to the required Database and type

show tables;

**Show the Columns**: show columns from <tableName>;

**Inserting values to the Tables**: insert into <tableName> values (value1,value2, value3). This will include values directly into all the 3 columns.

If we want to insert into specific Columns only:

Insert into <tableName> (column1, column2) values (value1, value2);

**To insert the Current date & time: We can use the NOW(); function.**

Ex: insert into <tableName> values (email, name, phone, NOW());

**Selecting Data**:

Select \* from <tableName> - Selects all the Columns from the Table.

Select column1, column2 from <tableName> - Selects only particular columns from the table.

We can also combine Select with ‘where’ for various operators.

**Operator Meaning**

**=** Equals

**<** Less than

**>** Greater than

**<=** Less than or equal to

**>=** Greater than or equal to

**!=** Not equal to

**IS NOT NULL -**Has a value

**IS NULL -** Does not have a value

**IS TRUE -**Has a true value

**IS FALSE -**Has a false value

**BETWEEN -**Within a range

**NOT BETWEEN -**Outside of a range

**IN -**Found within a list of values

**NOT IN -**Not found within a list of values

**OR** (also **||**) Where at least one of two conditionals is true

**AND** (also **&&**) Where both conditionals are true

**NOT** (also **!**) Where the condition is not true

Ex: select salary from employees where id between 1 and 3;

select \* from employees where id in (1,2,3);

**LIKE & NOT LIKE**:

SELECT \* FROM users WHERE last\_name LIKE 'Ravi%'

This query will return all rows whose last\_name value begins with Ravi.

SELECT \* FROM users WHERE last\_name LIKE '%Modi'

This query will return all rows whose last\_name value ends with Modi.

**ORDER:**

To Order the results of the query in Ascending or Descending order, we would use:

SELECT \* FROM tablename ORDER BY column

Default ordering is Ascending order.

SELECT \* FROM tablename ORDER BY column desc

**LIMIT**: Used to specify the number of records to return.

Select \* from <column1> where <column2> = <someCondition> limit 1,3;

The above query returns the records from row 1 till row 3.

SELECT \* FROM tablename LIMIT x, y

you can have y records returned, starting at x. To have records 11 through 20 returned, we would write

SELECT \* FROM tablename LIMIT 10, 10

**Update Data**:

To update a single column, type:

UPDATE tablename SET column=value where <someConditon>

We can alter multiple columns at a single time, separating each from the next by a comma.

UPDATE tablename SET column1=valueA, column5=valueB <someCondition>

**Deleting Data**:

We should always use a “where” condition while Deleting certain Data.

DELETE FROM tablename WHERE condition

The command will delete every record in a table, making it empty again.

Once you have deleted a record, there is no way of retrieving it.

**The preferred way to empty a table is to use TRUNCATE as below:**

TRUNCATE TABLE tablename

It deletes all the records from an existing table but not the table itself. The structure or schema of the table is preserved.

**To delete all of the data in a table, as well as the table itself, use DROP TABLE:**

DROP TABLE tablename

Objects deleted using DROP are permanently lost and it cannot be rolled back.

**To delete an entire database, including every table therein and all of its data, use:**

DROP DATABASE databasename

**MIN**: Returns the minimum value from the selected column.

Select min(column1) from tableName where condition;

**MAX**: Returns the maximum value from the selected column.

Select max(column1) from tableName where condition;

**COUNT**: Returns the number of rows that satisfy a particular condition.

Select count(column1) from tableName where condition;

**AVG**: Returns the average value of the numbers in the selected column.

Select avg(column1) from tableName where condition;

**SUM**: Returns the sum of all the values in the selected column.

Select sum(column1) from tableName where condition;

**DISTINCT**: Returns only unique values from the selected column(s).

Select distinct column1, column2 from tableName;

**INNER JOIN**: Returns the common values b/w both the tables.

Select table1.column1, table2.column2 from table1 inner join table2 on table1.column2=table2.column1.

**LEFT OUTER JOIN**: Returns the values from table1 (left table) and also the matching records from table2 (right table).

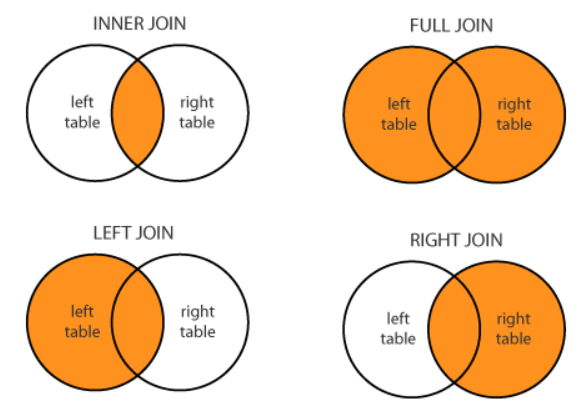
Select table1.column1, table2.column2 from table1 left outer join table2 on table1.column2=table2.column1.

**RIGHT OUTER JOIN**: Returns the values from table2 (right table) and also the matching records from table1 (left table).

Select table1.column1, table2.column2 from table1 right outer join table2 on table1.column2=table2.column1.

**FULL OUTER JOIN**: We can do a UNION to get all the values from both the tables.

Select column(s) from table1 union select column(s) from table2.



**PRIMARY KEY**: The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

**FOREIGN KEY**: The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

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