SQL queries are essential in database testing, enabling testers to interact with the database to ensure its functionality, integrity, and performance. Here, I will demonstrate various SQL queries for testing purposes, focusing on retrieving accurate data through different methods.

Database testing involves confirming the accuracy, reliability, and performance of a database system. This includes examining data integrity, consistency, validation, and overall database functionality.

Here are some key aspects of database testing:

- **Data Integrity:** Database testing ensures the accuracy, completeness, and consistency of the data stored within the database. It involves verifying primary key constraints, foreign key relationships, unique constraints, and data validations specified in the database schema.
- **Data Manipulation:** Database testing involves examining various operations performed on the database, including inserting, updating, and deleting records. It ensures these operations are carried out correctly and yield the expected results.
- **Security and Access Control:** Database testing verifies the security measures implemented in the database system. It includes testing access controls, user permissions, authentication mechanisms, and data encryption to ensure that sensitive data is protected and unauthorized access is prevented.
- **Data Retrieval:** Testing the retrieval of data from the database is an important aspect of database testing. It involves verifying the accuracy and completeness of data fetched using queries and ensuring that the retrieved data matches the expected results.
- **Performance and Scalability:** Database testing includes evaluating the performance of the database under different loads and stress conditions. It involves measuring response times, throughput, and concurrency to ensure that the database can handle the expected workload efficiently. Scalability testing is performed to assess the performance of the database as the data volume and user load increase.
- **Data Migration and Integration:** Testing database migration and integration processes is essential when moving data between databases or merging data from various sources. This involves validating data mappings and transformations, and ensuring data accuracy and integrity post-migration or integration.
- **Error Handling and Recovery:** Database testing includes testing error handling mechanisms and recovery procedures. It involves simulating various error scenarios, such as network failures, system crashes, and validating the database's ability to recover data and maintain data integrity.
- **Backup and Recovery:** Database testing also involves testing backup and recovery procedures to ensure that data can be successfully backed up and restored in case of data loss or system failures. It includes testing backup schedules, recovery mechanisms, and data consistency after recovery. To perform effective database testing, various techniques and tools are used, such as SQL queries, test data generation, database comparison tools, and automation frameworks. It is important to have a comprehensive test strategy and test cases that cover all aspects of the database system to ensure its reliability and performance.

When conducting database testing, it's crucial to account for various scenarios such as boundary values, null values, error conditions, performance under different loads, and concurrency. By running a range of queries and evaluating the results, testers can determine the quality and reliability of the database system. Database testing verifies the accuracy, integrity, and performance of a database. Queries play a vital role in this process as they are used to retrieve and manipulate data. Here are some common types of queries utilized in database testing:

Example:

First, There are three tables Employee, Manager, Department with some values inside them:

A **"CREATE TABLE"** query is used to create a new table in a database. It defines the structure of the table by specifying the columns and their data types, constraints, and any additional properties.

--Create Employee Table:

```
CREATE TABLE EMPLOYEE (
   Employee_Id int PRIMARY KEY,
   First_Name varchar(255),
   Middle_Name varchar(255),
   Last_Name varchar(255),
   Age int,
   CONSTRAINT Age_Check CHECK (Age >= 18),
   DOJ DATE,
   Address varchar(255),
   Department_Id int NOT NULL,
   Manager_Id int,
   Progress_Value int,
   FOREIGN KEY (Manager_Id) references MANAGER(Manager_Id)
);
```

--Create Department Table:

```
CREATE TABLE DEPARTMENT (
  Department_Id int,
  Name varchar(255),
);
```

--Create Manager Table:

```
CREATE TABLE MANAGER (
  Manager_Id INT PRIMARY KEY,
  Name varchar(255),
  Location varchar(255),
);
```

The table needs to have some values in order for the testers to test the accuracy of the data. To insert values inside the tables we use the Insert Into query.

The **"INSERT INTO"** query is used to add new records or data into an existing table in a database. It allows you to specify the table name and provide values for the columns in the table. It allows you to specify the table name and provide values for the columns in the table.

-- Insert sample data into the EMPLOYEE table:

```
INSERT INTO EMPLOYEE (Employee_Id, First_Name, Middle_Name, Last_Name, Age, DOJ, Address, Department_Id, Manager_Id, Progress_Value)
  VALUES
    (1, 'Rahim', 'Uddin', 'Chowdhury', 25, '2020-01-15', 'Dhaka, Bangladesh', 101, 1, 5),
    (2, 'Karim', 'Hasan', 'Ahmed', 30, '2019-07-22', 'Chittagong, Bangladesh', 102, 2, NULL),
    (3, 'Farhana', ‘Rahman’, 'Sultana', 28, '2021-03-18', 'Sylhet, Bangladesh', 103, 3, 7),
    (4, 'Tanvir', 'Rashid', 'Hossain', 35, '2018-11-05', 'Khulna, Bangladesh', 101, 4, 9),
    (5, 'Sumi', 'Akter', 'Khan', 26, '2020-09-14', 'Rajshahi, Bangladesh', 104, 5, NULL),
    (6, 'Nusrat','Jahan',’Bristy’, 29, '2017-02-28', 'Barisal, Bangladesh', 102, 4, NULL),
    (7, 'Imran', 'Hossain', 'Sheikh', 32, '2016-05-12', 'Dhaka, Bangladesh', 103, 3, 8),
    (8, 'Alamgir', 'Kabir', 'Rahman', 40, '2015-08-09', 'Chittagong, Bangladesh', 104, 1, NULL),
    (9, 'Mitu', 'Begum', 'Ali', 27, '2021-12-30', 'Sylhet, Bangladesh', 101, 2, 8),
    (10, 'Rifat', 'Bin', 'Kamal', 34, '2019-10-25', 'Khulna, Bangladesh', 103, 2, 9);
```

-- Insert sample data into the DEPARTMENT table:

```
INSERT INTO DEPARTMENT VALUES
    (101, 'ACCOUNTING'),
    (102, 'Development'),
    (103, 'SALES'),
    (104, 'OPERATIONS');
```

-- Insert sample data into the EMPLOYEE table:

```
INSERT INTO MANAGER (Manager_Id, Name, Location) VALUES
     (1, ‘Ayesha Rahman’ , ‘Dhaka’),
     (2, ‘Hasan Mahmud’ , ‘Chittagong’),
     (3, ‘Shakib Al Hasan’, ‘Sylhet’),
     (4, ‘Nusrat Jahan’, ‘Khulna’),
     (5, ‘Fahim Ahmed’, ‘Rajshahi’),
     (6, ‘Sadia Karim’ , ‘Barisal’),
     (7, ‘Rahim Ullah’ , ‘Rangpur’),
     (8, ‘Tariq Aziz’, ‘Comilla’)
     (9, ‘Mariam Akhtar’, ‘Gazipur’)
```

Now, I am getting the values from these tables by using (SELECT * from EMPLOYEE) , (SELECT * from DEPARTMENT), (SELECT * from MANAGER)

```sql
CREATE TABLE EMPLOYEE (
    Employee_Id int PRIMARY KEY,
    First_Name varchar(255),
    Middle_Name varchar(255),
    Last_Name varchar(255),
    Age int,
    CONSTRAINT Age_Check CHECK (Age >= 18),
    DOJ DATE,
    Address varchar(255),
    Department_Id int NOT NULL,
    Manager_Id int,
    Progress_Value int,
    FOREIGN KEY (Manager_Id) references MANAGER(Manager_Id)
);
SELECT * FROM EMPLOYEE;
```

Employee_Id	First_Name	Middle_Name	Last_Name	Age	DOJ	Address	Department_Id	Manager_Id	Progress_Value

Pic: Employee Table Create

```sql
INSERT INTO EMPLOYEE (Employee_Id, First_Name, Middle_Name, Last_Name, Age, DOJ, Address, Department_Id, Manager_Id, Progress_Value
VALUES
(1, 'Rahim', 'Uddin', 'Chowdhury', 25, '2020-01-15', 'Dhaka, Bangladesh', 101, 1, 5),
(2, 'Karim', 'Hasan', 'Ahmed', 30, '2019-07-22', 'Chittagong, Bangladesh', 102, 2, NULL),
(3, 'Farhana', 'Rahman', 'Sultana', 28, '2021-03-18', 'Sylhet, Bangladesh', 103, 3, 7),
(4, 'Tanvir', 'Rashid', 'Hossain', 35, '2018-11-05', 'Khulna, Bangladesh', 101, 4, 9),
(5, 'Sumi', 'Akter', 'Khan', 26, '2020-09-14', 'Rajshahi, Bangladesh', 104, 5, NULL),
(6, 'Nusrat', 'Jahan', 'Bristy', 29, '2017-02-28', 'Barisal, Bangladesh', 102, 4, NULL),
(7, 'Imran', 'Hossain', 'Sheikh', 32, '2016-05-12', 'Dhaka, Bangladesh', 103, 3, 8),
(8, 'Alamgir', 'Kabir', 'Rahman', 40, '2015-08-09', 'Chittagong, Bangladesh', 104, 1, NULL),
(9, 'Mitu', 'Begum', 'Ali', 27, '2021-12-30', 'Sylhet, Bangladesh', 101, 2, 8),
(10, 'Rifat', 'Bin', 'Kamal', 34, '2019-10-25', 'Khulna, Bangladesh', 103, 2, 9);
SELECT * FROM EMPLOYEE;
```

Employee_Id	First_Name	Middle_Name	Last_Name	Age	DOJ	Address	Department_Id	Manager_Id	Progress_Value
1	Rahim	Uddin	Chowdhury	25	202...	Dhaka, ...	101	1	5
2	Karim	Hasan	Ahmed	30	201...	Chittago...	102	2	NULL
3	Farhana	Rahman	Sultana	28	202...	Sylhet, ...	103	3	7

Pic: Insert Data in Employee Table

```sql
CREATE TABLE MANAGER (
  Manager_Id INT PRIMARY KEY,
  Name varchar(255),
  Location varchar(255),
  );
```

Pic: Manager Table Create

```sql
INSERT INTO MANAGER (Manager_Id, Name, Location) VALUES
  (1, 'Ayesha Rahman', 'Dhaka'),
  (2, 'Hasan Mahmud', 'Chittagong'),
  (3, 'Shakib Al Hasan', 'Sylhet'),
  (4, 'Nusrat Jahan', 'Khulna'),
  (5, 'Fahim Ahmed', 'Rajshahi'),
  (6, 'Sadia Karim', 'Barisal'),
  (7, 'Rahim Ullah', 'Rangpur'),
  (8, 'Tariq Aziz', 'Comilla'),
  (9, 'Mariam Akhtar', 'Narayanganj'),
  (10, 'Jamal Khan', 'Gazipur');

SELECT * FROM MANAGER;
```

Manager_Id	Name	Location
1	Ayesha Rahman	Dhaka
2	Hasan Mahmud	Chittagong
3	Shakib Al Hasan	Sylhet

Pic: Insert Data in Manager Table

```sql
INSERT INTO DEPARTMENT VALUES
(101, 'ACCOUNTING'),
(102, 'Development'),
(103, 'SALES'),
(104, 'OPERATIONS');

SELECT * FROM DEPARTMENT;
```

Department_Id	Name
101	ACCOUNTING
102	Development
103	SALES
104	OPERATIONS

Pic: Insert Data in Department Table

I have prepared some questions to work on these tables. They are as follows:

1. List out the employees whose are not achieve any progress value.

➔ **SELECT * FROM EMPLOYEE where Progress_Value is Null**

The "Is Null" operator is used in SQL queries to check whether a column's value is null or contains no value.

```
1 SELECT * FROM EMPLOYEE WHERE Progress_Value is Null
```

Employee_Id	First_Name	Middle_Name	Last_Name	Age	DOJ	Address	Department_Id	Manager...	Progress_Value
2	Karim	Hasan	Ahmed	30	201...	Chittago...	102	2	NULL
5	Sumi	Akter	Khan	26	202...	Rajshah...	104	5	NULL
6	Nusrat	Jahan	Bristy	29	201...	Barisal, ...	102	4	NULL
8	Alamgir	Kabir	Rahman	40	201...	Chittago...	104	1	NULL

Pic: Null Progress Value

2. List out the department_id having at least 2 employees.

➔ **SELECT Department_Id, count(*) from EMPLOYEE GROUP BY Department_Id HAVING COUNT(*)>=2**

```
1 SELECT Department_Id, COUNT(*) FROM EMPLOYEE GROUP BY Department_Id HAVING COUNT(*)>=2
2
```

Department_Id	
101	3
102	2
103	3
104	2

Pic: Which Department Id has more than 2 Employee

3. Display the employees who's age is maximum.

➔ **SELECT * FROM EMPLOYEE WHERE Age = (SELECT max(Age) FROM EMPLOYEE);**

```
SELECT * FROM EMPLOYEE WHERE Age = ( SELECT max(Age)  FROM EMPLOYEE);
```

Employee_Id	First_Name	Middle_Name	Last_Name	Age	DOJ	Address	Department_Id	Manager_Id	Progr...
8	Alamgir	Kabir	Rahman	40	201...	Chittago...	104	1	NULL

Pic: Employee Who's age is max

4. List out the employees who are working in department id 103

➔ **SELECT * FROM EMPLOYEE WHERE Department_Id=103**

```
SELECT * FROM EMPLOYEE WHERE Department_Id=103
```

Employee_Id	First_Name	Middle_Name	Last_Name	Age	DOJ	Address	Department_Id	Manager...	Progress_Value
3	Farhana	Rahman	Sultana	28	202...	Sylhet, ...	103	3	7
7	Imran	Hossain	Sheikh	32	201...	Dhaka, ...	103	3	8
10	Rifat	Bin	Kamal	34	201...	Khulna, ...	103	2	9

Pic: Employee Who's work on 103 Department

5. Write an SQL query to fetch the count of employees live in Dhaka"

➔ **SELECT COUNT(*) FROM EMPLOYEE WHERE Address = 'Dhaka, Bangladesh';**

A "COUNT" query is used to retrieve the number of rows or records that match a specific condition in a database table. It allows you to count the occurrences of a particular value, the total number of rows in a table, or the number of rows that satisfy a given condition. "FROM table_name" The name of the table from which you want to retrieve the data.

```
SELECT COUNT(*) FROM EMPLOYEE WHERE Address = 'Dhaka, Bangladesh';
```

2

Pic: Number of Employee live in Dhaka

6. List out the Employee id, First Name in descending order based on the Manager column.

➔ **SELECT Employee_Id, First_Name, Manager_Id FROM EMPLOYEE ORDER By Manager_id Desc**

```
1 SELECT Employee_Id, First_Name, Manager_Id FROM EMPLOYEE ORDER BY Manager_id DESC;
```

Employee_Id	First_Name	Manager_Id
5	Sumi	5
6	Nusrat	4
4	Tanvir	4
3	Farhana	3
7	Imran	3
2	Karim	2
9	Mitu	2
10	Rifat	2
1	Rahim	1
8	Alamgir	1