**1. Basic SQL Knowledge – 20 mins**

* **CRUD Operations**:
  + SELECT: Retrieve data.
  + INSERT: Add data to tables.
  + UPDATE: Modify existing data.
  + DELETE: Remove data from tables.
* **Filtering**:
  + WHERE clause with conditions (=, >, <, >=, <=, !=, BETWEEN, IN, LIKE).
* **Sorting and Limiting**:
  + ORDER BY, LIMIT, and OFFSET.

**Resource** : https://www.w3schools.com/sql/

**2. Understanding Database Structure**

* **Schemas**:
  + Understand the structure of the database and relationships between tables.
* **Primary and Foreign Keys**:
  + Validate referential integrity.
* **Indexes**:
  + Understand indexing and its impact on query performance.
* **Constraints**:
  + Primary key, unique key, not null, default, foreign key constraints.
* **Resource** : https://www.w3schools.com/sql/sql\_constraints.asp

**3. Joins and Relationships**

* **Types of Joins**:
  + INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN.
* **Cross Joins and Self Joins**:
  + Test scenarios requiring multiple relationships.

**4. Data Validation**

* **Data Integrity**:
  + Verify data correctness between the application and database.
* **Data Consistency**:
  + Ensure data remains consistent after operations like insert/update/delete.
* **Boundary Value Testing**:
  + Test numeric and string boundaries.

Resource below the document

**5. Transactions**

* **ACID Properties**:
  + Understand Atomicity, Consistency, Isolation, Durability.
* **Commit and Rollback**:
  + Test transaction management in the database.

**6. Database Testing Types**

* **Data-Driven Testing**:
  + Validate test cases using data from the database.
* **ETL Testing** (for testers working with data warehousing):
  + Verify data extraction, transformation, and loading processes.
* **Performance Testing**:
  + Check query execution time and optimize SQL.
* **Security Testing**:
  + Validate permissions, SQL injection vulnerability, and data protection.

**7. Stored Procedures, Functions, and Triggers**

* **Stored Procedures**:
  + Test input and output parameters.
* **Functions**:
  + Verify calculations and return values.
* **Triggers**:
  + Ensure triggers execute correctly during data modification.

**8. Advanced SQL**

* **Subqueries**:
  + Single-row and multi-row subqueries.
* **Set Operations**:
  + UNION, UNION ALL, INTERSECT, EXCEPT.
* **Window Functions**:
  + ROW\_NUMBER, RANK, DENSE\_RANK, OVER.

**9. Testing Data Integrity**

* **Duplicate Records**:
  + Identify and handle duplicates.
* **Null Handling**:
  + Test columns with nullable constraints.
* **Foreign Key Integrity**:
  + Ensure child records cannot exist without parent records.

**10. Testing Tools and Techniques**

* **Database Query Tools**:
  + Familiarity with tools like MySQL Workbench, SQL Server Management Studio (SSMS), pgAdmin, etc.
* **Automation Tools**:
  + Use automation tools (e.g., Selenium with database, or Python libraries like pyodbc, psycopg2, etc.) for data validation.
* **SQL Profilers and Monitors**:
  + Monitor query performance.

**11. Knowledge of Non-Relational Databases (Optional)**

* **NoSQL Databases**:
  + Basic understanding of MongoDB, Cassandra, etc., if the application uses them.

**12. Real-World Scenarios**

* **Testing Data Migration**:
  + Validate data correctness when migrating between databases.
* **Testing Backend APIs**:
  + Ensure APIs interacting with the database return the expected results.
* **Audit Logs**:
  + Validate database logging mechanisms.

**13. Practical Knowledge**

* **Write and Execute Test Cases**:
  + Define test cases for database testing based on requirements.
* **Data Backup and Recovery**:
  + Test the backup and restore processes.

[**Understanding Data Integrity, Data Consistency, and Boundary Value Testing in Database Testing**](#_top)

**1. Data Integrity**

**Definition**: Data integrity ensures that data is accurate, consistent, and valid in the database. It checks that the data adheres to defined rules, constraints, and relationships.

**How to Test:**

1. **Primary Key and Foreign Key Validation**:
   * Ensure that primary keys are unique and not null.
   * Verify that foreign keys correctly reference the primary keys of related tables.
   * Example Test:

sql

CopyEdit

SELECT COUNT(\*)

FROM Orders

WHERE CustomerID NOT IN (SELECT CustomerID FROM Customers);

* + - This checks for Orders without a valid CustomerID.

1. **Check Constraints**:
   * Verify that constraints (e.g., CHECK, NOT NULL, DEFAULT) are enforced.
   * Example:

sql

CopyEdit

INSERT INTO Employees (Name, Age)

VALUES ('John', -5); -- Should fail due to a check constraint

1. **Data Type Validation**:
   * Ensure that data matches the defined data types.
   * Example:

sql

CopyEdit

INSERT INTO Orders (OrderDate)

VALUES ('InvalidDate'); -- Should fail due to a data type mismatch

1. **Unique Constraints**:
   * Validate columns with UNIQUE constraints.
   * Example Test:

sql

CopyEdit

INSERT INTO Users (Email)

VALUES ('test@example.com'), ('test@example.com'); -- Should fail

**2. Data Consistency**

**Definition**: Data consistency ensures that data remains accurate and consistent across the application and database, even after operations like INSERT, UPDATE, or DELETE.

**How to Test:**

1. **Verify Data After Operations**:
   * Ensure data matches expected values after INSERT, UPDATE, or DELETE.
   * Example Test:

sql

CopyEdit

INSERT INTO Products (ProductName, Price)

VALUES ('New Product', 50);

SELECT \* FROM Products WHERE ProductName = 'New Product';

-- Verify that 'New Product' exists with Price = 50

1. **Check Cascading Updates/Deletes**:
   * Ensure foreign key relationships handle cascading updates or deletes correctly.
   * Example Test:

sql

CopyEdit

DELETE FROM Customers WHERE CustomerID = 1;

SELECT \* FROM Orders WHERE CustomerID = 1; -- Should return no rows if cascade delete is enabled

1. **ACID Properties**:
   * Verify that transactions maintain **Atomicity**, **Consistency**, **Isolation**, and **Durability**.
   * Example Test:

sql

CopyEdit

BEGIN TRANSACTION;

UPDATE Account SET Balance = Balance - 100 WHERE AccountID = 1;

UPDATE Account SET Balance = Balance + 100 WHERE AccountID = 2;

ROLLBACK;

-- Verify that balances remain unchanged

**3. Boundary Value Testing**

**Definition**: Boundary value testing ensures that numeric or string fields behave correctly at their boundaries (e.g., minimum, maximum, or specific edge cases).

**How to Test:**

1. **Numeric Boundaries**:
   * Test lower and upper limits for numeric fields.
   * Example Test:

sql

CopyEdit

INSERT INTO Products (ProductName, Price)

VALUES ('Boundary Test', -1); -- Should fail due to a constraint

INSERT INTO Products (ProductName, Price)

VALUES ('Boundary Test', 100000); -- Test upper limit

1. **String Length**:
   * Test minimum, exact, and maximum lengths of strings.
   * Example Test:

sql

CopyEdit

INSERT INTO Users (Username)

VALUES (''); -- Test empty string

INSERT INTO Users (Username)

VALUES (REPEAT('A', 51)); -- Test exceeding max length (e.g., 50 characters)

1. **Date Boundaries**:
   * Test valid and invalid dates.
   * Example Test:

sql

CopyEdit

INSERT INTO Events (EventDate)

VALUES ('2023-02-29'); -- Should fail (invalid date for a non-leap year)

1. **Special Characters**:
   * Test how the database handles special characters in fields.
   * Example Test:

sql

CopyEdit

INSERT INTO Comments (Content)

VALUES ('<script>alert("XSS")</script>'); -- Check for injection vulnerabilities

**Summary Table:**

| **Aspect** | **Objective** | **Examples** |
| --- | --- | --- |
| **Data Integrity** | Verify correctness of data, adherence to constraints, and relationships. | Primary key/foreign key checks, data type validation, unique constraints. |
| **Data Consistency** | Ensure data remains consistent across operations. | Test cascading updates/deletes, verify transaction behavior (ACID). |
| **Boundary Testing** | Validate behavior at the edges of acceptable input values. | Test numeric limits, string lengths, date ranges, and handling of special characters. |

**Practice Example:**

1. Test that the Price column in the Products table allows only values between 0 and 1000.
2. Verify that deleting a Customer also removes all their related Orders from the Orders table.
3. Check if a Username column allows special characters like @ or #.