CERTIFIED QUIZ

Introduction to SQL

SQL (Structured Query Language) is a standard programming language specifically designed for managing and manipulating relational databases. SQL enables users to create, read, update, and delete (CRUD) data in a database. It is widely used by database administrators, developers, and data analysts. SQL was initially developed at IBM in the 1970s and has since become the industry standard for database management.

Basic SQL Syntax

SQL syntax is composed of commands and keywords used to interact with a database. The most common commands include:

- SELECT: Retrieves data from the database.
- INSERT: Adds new data into the database.
- UPDATE: Modifies existing data in the database.
- DELETE: Removes data from the database.
- CREATE: Creates a new table or database.

Example:
sql

```
SELECT * FROM Employees WHERE Age > 30;
```

Data Types

SQL provides various data types to define the kind of data that can be stored in a table's columns. Common data types include:

- INT: Integer data (whole numbers).
- VARCHAR(size): Variable-length string data.
- DATE: Date values.
- FLOAT: Floating-point numbers (decimals).
- BOOLEAN: TRUE or FALSE values.

Example:
sql

```
CREATE TABLE Students (
ID INT,
Name VARCHAR(50),
Birthdate DATE
);
```

SQL Joins

SQL joins are used to combine records from two or more tables in a relational database based on a related column between them. Types of joins include:

- INNER JOIN: Returns records with matching values in both tables.
- LEFT JOIN: Returns all records from the left table and matching records from the right table.
- RIGHT JOIN: Returns all records from the right table and matching records from the left table.
- FULL OUTER JOIN: Returns records when there is a match in either table.

Example:

sql

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

SQL Aggregation Functions

SQL offers several aggregation functions to perform calculations on data, including:

- COUNT: Returns the number of rows that match a condition.
- SUM: Returns the sum of a numeric column.
- AVG: Returns the average value of a numeric column.
- MIN and MAX: Returns the minimum or maximum value of a column.

Example:
sql

```
SELECT COUNT(*) FROM Orders WHERE OrderDate = '2024-01-01';
```

SQL Constraints

SQL constraints are rules applied to columns in a table to ensure the integrity and accuracy of the data. Common constraints include:

- NOT NULL: Ensures that a column cannot have a NULL value.
- UNIQUE: Ensures that all values in a column are different.
- PRIMARY KEY: Uniquely identifies each record in a table.
- FOREIGN KEY: Ensures referential integrity between tables.
- CHECK: Ensures that values in a column satisfy a specific condition.

Example:
sql

```
CREATE TABLE Employees (
ID INT PRIMARY KEY,
Name VARCHAR(100) NOT NULL,
Salary FLOAT CHECK (Salary > 0)
);
```

Subqueries

A subquery is a query nested inside another SQL query. Subqueries can be used to return data that will be used in the main query as a condition for filtering results.

Example:
sql

```
SELECT Name
FROM Employees
WHERE Salary > (SELECT AVG(Salary) FROM Employees);
```

Transactions

Transactions in SQL are used to ensure that a series of operations are executed as a single unit. If any operation fails, the entire transaction is rolled back, ensuring data integrity. A transaction has the following properties, known as ACID:

- Atomicity: All operations are completed or none are.
- Consistency: Ensures that data moves from one valid state to another.

- Isolation: Transactions are executed independently of others.
- Durability: Once a transaction is committed, it remains permanent.

Example:

sql

```
BEGIN TRANSACTION;
UPDATE Accounts SET Balance = Balance - 100 WHERE AccountID = 1;
UPDATE Accounts SET Balance = Balance + 100 WHERE AccountID = 2;
COMMIT;
```