Q.1) Create a Sales Table and Use Aggregate Functions

a) Create Sales Table

```
CREATE TABLE Sales (
   SaleID INT PRIMARY KEY,
   ProductID INT,
   Quantity INT,
   SaleAmount DECIMAL(10,2),
   SaleDate DATE);
```

b) Insert Sales Records

```
INSERT INTO Sales (SaleID, ProductID, Quantity, SaleAmount, SaleDate) VALUES
(1, 101, 5, 250000.00, '2025-03-10'),
(2, 102, 8, 120000.00, '2025-02-15'),
(3, 103, 2, 60000.00, '2025-03-20'),
(4, 104, 1, 15000.00, '2025-02-20'),
(5, 105, 10, 20000.00, '2025-03-25'),
(6, 106, 4, 50000.00, '2025-03-01'),
(7, 107, 6, 90000.00, '2025-02-05'),
(8, 108, 7, 105000.00, '2025-03-12'),
(9, 109, 3, 45000.00, '2025-03-14'),
(10, 110, 9, 135000.00, '2025-03-18');
```

c) Calculate Total Revenue

```
SELECT SUM(SaleAmount) AS TotalRevenue FROM Sales;
```

d) Find Product with Highest Sale Amount

```
SELECT ProductID, MAX(SaleAmount) AS HighestSale FROM Sales GROUP BY ProductID;
```

e) Retrieve Average Sale Amount Per Transaction

```
SELECT AVG(SaleAmount) AS AverageSaleAmount FROM Sales;
```

Recent Tool/Technology for Database Management:

One of the recent tools in database management that has gained popularity is **GraphQL**.

Key Features of GraphQL:

- **Flexible Queries:** GraphQL allows clients to request only the data they need, which can reduce over-fetching of data compared to traditional REST APIs.
- **Single Endpoint:** Unlike REST, which requires multiple endpoints for different resources, GraphQL uses a single endpoint to handle all queries.
- **Strongly Typed Schema:** GraphQL is based on a schema that defines the structure of data, enabling easier validation and introspection.
- **Real-time Capabilities:** With subscriptions, GraphQL supports real-time data updates, making it suitable for applications that require live data.

Why it's Used in the Industry:

- **Efficiency and Performance:** GraphQL minimizes the amount of data transferred between the server and client, enhancing performance, especially in mobile apps and complex web applications.
- **Developer Productivity:** It simplifies backend and frontend interactions by allowing the frontend to request exactly the data it needs without worrying about backend changes.
- **Ecosystem Integration:** GraphQL is widely used in modern application development frameworks like React, Apollo, and others, making it a good fit for full-stack development.

Q.2) Use DDL and DML Commands

a) Create a Products Table with DDL

```
CREATE TABLE Products (
   ProductID INT PRIMARY KEY,
   ProductName VARCHAR(100),
   Price DECIMAL(10, 2),
   StockQuantity INT);
```

b) Insert Five Product Records and Display All Products

```
INSERT INTO Products (ProductID, ProductName, Price, StockQuantity) VALUES
(1, 'Laptop', 75000.00, 20),
(2, 'Smartphone', 35000.00, 50),
(3, 'Tablet', 15000.00, 30),
(4, 'Headphones', 3000.00, 100),
(5, 'Smartwatch', 5000.00, 60);

-- Display all products
SELECT * FROM Products;
```

c) Update the Price of a Product with ProductID = 3

```
UPDATE Products
SET Price = 16000.00
WHERE ProductID = 3;

-- Check the changes
SELECT * FROM Products WHERE ProductID = 3;
```

d) Delete a Product from the Table and Verify the Changes

```
DELETE FROM Products WHERE ProductID = 4;

-- Verify the changes
SELECT * FROM Products;
```

e) **Alter the Table to Add a New Column Discount and Set a Default Value of 5%**

```
ALTER TABLE Products
ADD Discount DECIMAL(5, 2) DEFAULT 5.00;
-- Verify the change
SELECT * FROM Products;
```

Recent Tool/Technology for Database Management:

A recent tool/technology in database management that has gained traction is NoSQL Databases**, specifically** MongoDB**.**

Key Features of MongoDB:

- **Document-Oriented:** MongoDB stores data in flexible, JSON-like documents (BSON), which can hold arrays, nested objects, and a variety of data types.
- **Scalability:** MongoDB offers horizontal scalability, meaning it can distribute data across multiple servers (sharding), making it suitable for handling large datasets.
- **Schema-less Design:** MongoDB is schema-less, allowing for easy updates to the data model without the need for database schema migrations.
- **High Availability:** Through replica sets, MongoDB ensures high availability, meaning if one server goes down, another replica can take over without service interruption.
- **Aggregation Framework:** MongoDB provides powerful aggregation operations that allow for complex data processing directly within the database.

Why it's Used in the Industry:

- **Flexible Data Models:** MongoDB is perfect for applications that require flexibility, like content management systems, social networks, and real-time analytics.
- **Big Data Handling:** MongoDB can efficiently manage large volumes of structured and unstructured data, making it suitable for big data applications.
- **Rapid Development:** Its schema-less nature allows developers to quickly adapt to changing application requirements without needing to redesign the database structure.
- **Performance:** MongoDB provides high performance for both read and write operations, even with large amounts of data, and can handle high throughput workloads.

Q.3) Create a Customer Table with Integrity Constraints

a) **Create a Customers Table with Constraints**

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    Age INT CHECK (Age > 18),
    Country VARCHAR(50) DEFAULT 'India');
```

b) **Insert a Valid Customer Record and Verify that the Default Country is Assigned if Not Explicitly Provided**

```
INSERT INTO Customers (CustomerID, CustomerName, Email, Age)
VALUES (1, 'John Doe', 'john.doe@example.com', 30);

-- Verify that the default country is assigned
SELECT * FROM Customers WHERE CustomerID = 1;
```

c) **Attempt to Insert a Customer with an Age of 16 and Observe the CHECK Constraint Violation**

```
-- This will result in a violation of the CHECK constraint
INSERT INTO Customers (CustomerID, CustomerName, Email, Age, Country)
VALUES (2, 'Alice Smith', 'alice.smith@example.com', 16, 'USA');
```

- **Error Message:** "CHECK constraint violation" due to age being less than 18.

d) **Try Inserting Two Customers with the Same Email ID and Observe the UNIQUE Constraint Violation**

```
-- Insert the first customer with a valid email
INSERT INTO Customers (CustomerID, CustomerName, Email, Age, Country)
VALUES (3, 'Bob Johnson', 'bob.johnson@example.com', 28, 'Canada');
```

```
-- Insert a second customer with the same email (violates the UNIQUE constraint)
INSERT INTO Customers (CustomerID, CustomerName, Email, Age, Country)
VALUES (4, 'Charlie Brown', 'bob.johnson@example.com', 35, 'UK');
```

- **Error Message:** "UNIQUE constraint violation" due to the duplicate email ID.

e) **Retrieve All Customers Who Are Older Than 25 and Belong to a Country Other Than 'India'**

```
SELECT * FROM Customers
WHERE Age > 25 AND Country != 'India';
```

Recent Tool/Technology for Database Management:

A recent tool/technology in database management that has been widely adopted is **PostgreSQL**.

Key Features of PostgreSQL:

- **Open-Source & Cross-Platform:** PostgreSQL is an open-source relational database system that works on various platforms, including Windows, Linux, and macOS.
- **ACID Compliance:** PostgreSQL is fully ACID-compliant, ensuring reliable transactions and data integrity.
- **Advanced Indexing & Search Capabilities:** It supports advanced indexing techniques like B-trees, GiST, GIN, and full-text search, allowing for efficient querying and searching on large datasets.
- **JSON & NoSQL Support:** PostgreSQL has excellent support for both relational and non-relational data models, including the ability to store and query JSON data. This makes it versatile for various applications, including NoSQL use cases.
- **Extensibility:** It is highly extensible, allowing developers to add custom data types, operators, functions, and even entire languages for stored procedures.

- **Replication & High Availability:** PostgreSQL supports various replication techniques (e.g., streaming replication), ensuring high availability and failover for critical applications.
- **Concurrency & Performance:** PostgreSQL uses MVCC (Multi-Version Concurrency Control) to handle multiple transactions concurrently without locking issues, which significantly boosts performance in multi-user environments.

Why it's Used in the Industry:

- **Data Integrity and Compliance:** PostgreSQL's ACID compliance and advanced data integrity features make it ideal for industries that require strict data governance, such as banking, healthcare, and finance.
- **Scalability and Performance:** Its advanced indexing and optimization techniques allow PostgreSQL to scale horizontally and vertically, handling large-scale applications efficiently.
- **Flexibility in Data Models:** The ability to handle both relational and non-relational data makes PostgreSQL a versatile choice for applications in various domains, from traditional transactional systems to modern data-driven applications requiring NoSQL-like flexibility.
- **Community Support:** As an open-source project, PostgreSQL benefits from a strong community and continuous development, ensuring that it remains at the forefront of database technology.

Q.4) Create a Table with Constraints

a) **Create an EmployeeDetails Table with EmployeeID as the PRIMARY KEY and DepartmentID as a FOREIGN KEY Referencing a Department Table**

First, we need to create the Department table before creating the EmployeeDetails table, as the foreign key depends on it.

```
-- Create the Department table
CREATE TABLE Department (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(100)
);

-- Create the EmployeeDetails table with a FOREIGN KEY reference to the Department table
CREATE TABLE EmployeeDetails (
    EmployeeID INT PRIMARY KEY,
    EmployeeName VARCHAR(100),
    DepartmentID INT,
    Salary DECIMAL(10, 2),
    FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID));
```

b) **Insert a Valid Employee Record with an Existing DepartmentID, Then Attempt to Insert an Employee with a Non-Existent DepartmentID and Observe the Constraint Violation**

```
-- First, insert a valid department record
INSERT INTO Department (DepartmentID, DepartmentName)
VALUES (1, 'HR');

-- Insert a valid employee record with existing DepartmentID
INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (1, 'John Doe', 1, 50000);

-- Attempt to insert an employee with a non-existent DepartmentID (violates the FOREIGN KEY constraint)
```

```
INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (2, 'Jane Smith', 99, 60000);
```

- **Error Message:** "Foreign key constraint violation" because DepartmentID = 99 does not exist in the Department table.

c) **Insert an Employee with a Duplicate EmployeeID and Check How the Primary Key Constraint Prevents Duplicate Entries**

```
-- Attempt to insert an employee with a duplicate EmployeeID (violates the PRIMARY KEY constraint)
INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (1, 'Alice Johnson', 1, 55000);
```

- **Error Message:** "Duplicate entry" because EmployeeID = 1 already exists in the table.

d) **Modify the Salary Column to Have a UNIQUE Constraint and Attempt to Insert Two Employees with the Same Salary to Test the Constraint**

```
-- Modify the Salary column to have a UNIQUE constraint
ALTER TABLE EmployeeDetails
ADD CONSTRAINT unique_salary UNIQUE (Salary);

-- Attempt to insert two employees with the same salary (violates the UNIQUE constraint)
INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (3, 'Charlie Brown', 1, 50000);

INSERT INTO EmployeeDetails (EmployeeID, EmployeeName, DepartmentID, Salary)
VALUES (4, 'David Clark', 1, 50000);
```

- **Error Message:** "Unique constraint violation" because two employees cannot have the same salary.

e) **Write a Query to Delete an Employee from EmployeeDetails and Ensure that the Deletion Does Not Violate Any Referential Integrity Constraints**

```
-- Delete an employee from EmployeeDetails
DELETE FROM EmployeeDetails
WHERE EmployeeID = 1;

-- Verify that the deletion does not violate any referential integrity constraints
SELECT * FROM EmployeeDetails;
```

If the foreign key constraint is set to ON DELETE CASCADE or ON DELETE SET NULL (depending on your specific setup), you need to check if the deletion impacts any related records in the Department table. If there's no cascading action, the deletion should not violate referential integrity.

Recent Tool/Technology for Database Management:

A recent tool that has gained significant traction in database management is **Google Cloud Spanner**.

Key Features of Google Cloud Spanner:

- **Distributed Relational Database:** Google Cloud Spanner is a fully managed, scalable, and relational database service that runs on Google Cloud. It provides strong consistency across distributed systems, making it suitable for both OLTP (Online Transaction Processing) and OLAP (Online Analytical Processing).
- **Horizontal Scalability:** Unlike traditional relational databases, Spanner offers automatic horizontal scaling, meaning it can seamlessly handle growing data without manual intervention. This is ideal for large applications with varying loads.
- **Global Distribution:** Spanner is designed to be a globally distributed database, offering low-latency access to data across regions, making it highly available for applications that require global reach.
- **ACID Compliance:** It maintains full ACID (Atomicity, Consistency, Isolation, Durability) compliance, making it suitable for mission-critical applications that require transaction safety and integrity.

- **SQL Support:** Spanner supports SQL queries, providing a familiar interface for developers who are used to relational databases like MySQL and PostgreSQL.
- **High Availability:** Built for high availability, Cloud Spanner offers automatic replication and failover capabilities, ensuring minimal downtime even in case of server or network failures.

Why it's Used in the Industry:

- **Scalability and Global Reach:** Cloud Spanner is widely used in industries that require global-scale databases with consistent performance, such as e-commerce, financial services, gaming, and telecommunications.
- **Real-Time Analytics:** Companies like Uber, Snap, and Spotify use Google Cloud Spanner to store vast amounts of real-time transactional and analytical data, ensuring high availability and low-latency access across regions.
- **Managed Service:** It is fully managed, reducing the overhead of managing database infrastructure and enabling companies to focus on application development rather than database maintenance.

5) Create an Employee Table with Various Columns

a) **Create a table Employee with attributes: EmployeeID (INT, PRIMARY KEY), Name (VARCHAR), Salary (DECIMAL), JoiningDate (DATE), and ActiveStatus (BOOLEAN)**

```
-- Create the Employee table with specified attributes
CREATE TABLE Employee (
   EmployeeID INT PRIMARY KEY,
   Name VARCHAR(100),
   Salary DECIMAL(10, 2),
   JoiningDate DATE,
   ActiveStatus BOOLEAN);
```

b) **Insert five sample employee records and ensure each employee has a unique EmployeeID**

```
-- Insert sample employee records
INSERT INTO Employee (EmployeeID, Name, Salary, JoiningDate, ActiveStatus)
VALUES
(1, 'Amit Sharma', 50000.00, '2021-05-10', TRUE),
(2, 'Raj Patel', 55000.00, '2022-03-15', TRUE),
(3, 'Suman Gupta', 60000.00, '2020-08-22', TRUE),
(4, 'Ravi Kumar', 45000.00, '2021-11-05', FALSE),
(5, 'Neha Singh', 70000.00, '2019-12-25', TRUE);
```

c) **Write a query to find all employees who joined before January 1, 2023**

```
-- Query to find employees who joined before January 1, 2023
SELECT * FROM Employee
WHERE JoiningDate < '2023-01-01';
```

d) **Update the salary of an employee named ‘Amit Sharma’ by 10% and display the updated record**

```
-- Update the salary of Amit Sharma by 10%
UPDATE Employee
```

```
SET Salary = Salary * 1.10
WHERE Name = 'Amit Sharma';

-- Display the updated record
SELECT * FROM Employee
WHERE Name = 'Amit Sharma';
```

e) **Retrieve all employees who are currently active (ActiveStatus = TRUE)**

```
-- Query to retrieve active employees
SELECT * FROM Employee
WHERE ActiveStatus = TRUE;
```

Recent Tool/Technology for Database Management:

A recent tool I have studied for database management is **Microsoft SQL Server**.

Key Features of Microsoft SQL Server:

- **Relational Database Management System (RDBMS):** SQL Server is an RDBMS designed to manage and store structured data. It uses SQL (Structured Query Language) to communicate with the database.
- **Transactional Integrity:** SQL Server ensures data consistency, integrity, and accuracy by supporting ACID (Atomicity, Consistency, Isolation, Durability) properties.
- **Advanced Security Features:** It provides advanced encryption, authentication, and access control, ensuring that sensitive data is protected from unauthorized access.
- **Data Warehousing and Business Intelligence (BI):** SQL Server includes features for data warehousing, OLAP (Online Analytical Processing), and reporting. It integrates with tools like SQL Server Reporting Services (SSRS) and SQL Server Integration Services (SSIS) to support data analysis and reporting.

- **Scalability and High Availability:** SQL Server supports clustering, replication, and Always On Availability Groups for high availability and disaster recovery solutions.
- **T-SQL:** SQL Server uses Transact-SQL (T-SQL), which is an extension of SQL that includes additional features for procedural programming, error handling, and system functions.

Why it's Used in the Industry:

Microsoft SQL Server is widely used across industries for managing large volumes of transactional data. It's known for its reliability, security, and scalability. SQL Server is ideal for businesses requiring robust, secure, and highly available database solutions, such as banking, healthcare, and e-commerce.

Many large organizations rely on SQL Server due to its integration with other Microsoft technologies (such as Azure and Power BI) and its ability to support both OLTP (Online Transaction Processing) and OLAP systems. Its features for handling complex queries, reporting, and analytics make it a popular choice for enterprises that need a powerful, enterprise-grade database management system.

6) Aggregate Functions (on a single table: Create a Sales table)

a) **From the Sales table, calculate the total sales amount (SUM) generated in the month of February 2025**

```
-- Query to calculate the total sales amount in February 2025
SELECT SUM(SaleAmount) AS TotalSales
FROM Sales
WHERE SaleDate BETWEEN '2025-02-01' AND '2025-02-28';
```

b) **Find the average (AVG) billing amount from the Sales table to assess customer spending behavior**

```
-- Query to find the average sale amount
SELECT AVG(SaleAmount) AS AverageSaleAmount
FROM Sales;
```

c) **Identify the minimum (MIN) quantity of products sold in any transaction using the Sales table**

```
-- Query to find the minimum quantity of products sold
SELECT MIN(Quantity) AS MinQuantitySold
FROM Sales;
```

d) **Determine the highest (MAX) discount applied on any sale using the Sales table**

```
-- Query to find the highest discount applied on any sale
SELECT MAX(Discount) AS MaxDiscount
FROM Sales;
```

e) **Use the COUNT function to find how many transactions were recorded in the Sales table for the product “Laptop”**

```
-- Query to count transactions for the product "Laptop"
SELECT COUNT(*) AS LaptopTransactionCount
FROM Sales
```

```
WHERE ProductName = 'Laptop';
```

Recent Tool/Technology for Database Management:

A recent tool I have studied for database management is **MongoDB**.

Key Features of MongoDB:

- **NoSQL Database:** MongoDB is a NoSQL document-based database that stores data in flexible, JSON-like documents, making it ideal for handling unstructured data, such as logs, social media content, or IoT sensor data.
- **Scalability:** MongoDB is designed for horizontal scaling, making it easy to handle large datasets and high-throughput workloads by distributing data across multiple servers (sharding).
- **High Availability:** MongoDB offers built-in replication, meaning data is automatically duplicated across different servers for redundancy and fault tolerance.
- **Flexible Schema:** The flexible schema allows documents within the same collection to have different structures, which is useful when the structure of data evolves over time or is unpredictable.
- **Aggregation Framework:** MongoDB provides powerful aggregation capabilities, enabling complex data transformations and analyses directly in the database.
- **JSON-like Data Format:** Since MongoDB stores data in BSON format (Binary JSON), it's easier to map data directly to application objects in many programming languages.

Why it's Used in the Industry:

MongoDB is widely used for applications that require high availability, scalability, and flexibility in storing large amounts of unstructured or semi-structured data. It is especially popular in modern web applications, content management systems, social networks, and IoT systems.

Companies like eBay, Uber, and Netflix use MongoDB to handle large-scale data with flexibility and performance, particularly when their requirements outgrow the relational database model. Its capability to scale horizontally and handle complex, real-time queries has made it a key tool for managing big data in various industries.

7) Constraints (on a single table: Employees)

a) **Create the Employees table with EmployeeID as PRIMARY KEY, Email as UNIQUE, and Salary with a CHECK (Salary > 10000) constraint.**

```
sql
CopyEdit
-- Create Employees table with constraints
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,  -- Not null constraint for Name column
    Email VARCHAR(100) UNIQUE,   -- Unique constraint for Email column
    Salary DECIMAL(10, 2) CHECK (Salary > 10000),  -- Check constraint for Salary
    Status VARCHAR(50) DEFAULT 'Active'  -- Default value for Status);
```

b) **Add a NOT NULL constraint on the Name column in the Employees table and try inserting a record without the name.**

```
-- Add NOT NULL constraint on Name column
ALTER TABLE Employees
MODIFY Name VARCHAR(100) NOT NULL;

-- Attempt to insert a record without specifying Name (this will fail)
INSERT INTO Employees (EmployeeID, Email, Salary)
VALUES (1, 'john.doe@example.com', 15000);
```

Error message: This will throw an error because the Name column cannot be NULL.

c) **Add a DEFAULT value 'Active' to the Status column in Employees, and insert a record without specifying the status to verify the default.**

```
-- Add a DEFAULT value of 'Active' to the Status column (already defined during table creation)
-- Insert a record without specifying the status
INSERT INTO Employees (EmployeeID, Name, Email, Salary)
VALUES (2, 'Jane Smith', 'jane.smith@example.com', 20000);

-- Verify the insertion
SELECT * FROM Employees;
```

This will insert the record, and the Status column will default to 'Active'.

d) **Insert a record into Employees where Salary is less than 10000 to test the CHECK constraint.**

```
-- Insert a record with Salary less than 10000 (this will fail)
INSERT INTO Employees (EmployeeID, Name, Email, Salary)
VALUES (3, 'Mark Johnson', 'mark.johnson@example.com', 8000);
```

Error message: This will throw an error because the CHECK (Salary > 10000) constraint is violated.

e) **Try inserting two employees with the same Email ID to verify the enforcement of the UNIQUE constraint.**

```
-- Insert two records with the same Email (this will fail)
INSERT INTO Employees (EmployeeID, Name, Email, Salary)
VALUES (4, 'Alice Brown', 'john.doe@example.com', 12000);

INSERT INTO Employees (EmployeeID, Name, Email, Salary)
VALUES (5, 'Charlie Davis', 'john.doe@example.com', 13000);
```

Error message: This will throw an error because the Email column has a UNIQUE constraint, and both records have the same email address.

Recent Tool/Technology for Database Management:

One of the recent tools I have studied for database management is **Amazon RDS (Relational Database Service)**.

Key Features of Amazon RDS:

- **Managed Service:** Amazon RDS is a fully managed relational database service, which means AWS handles routine database tasks like provisioning, patching, backup, recovery, and scaling.
- **Multiple Database Engines:** Amazon RDS supports several database engines including MySQL, PostgreSQL, Oracle, SQL Server, and Amazon Aurora.
- **Scalability:** It provides easy scalability to accommodate growing databases, whether vertically (increasing instance size) or horizontally (adding read replicas).
- **High Availability:** Amazon RDS offers features like Multi-AZ deployments for high availability and automatic failover.
- **Security:** It integrates with AWS security tools, allowing users to manage encryption, IAM (Identity and Access Management) roles, and VPC (Virtual Private Cloud) security groups.
- **Automated Backups:** RDS automatically backs up your database daily, which is important for data recovery and minimizing downtime in case of failure.
- **Performance Monitoring:** With Amazon RDS, users can monitor their database's performance through tools like Amazon CloudWatch and Performance Insights.

Why it's Used in the Industry:

Amazon RDS is widely used in the industry due to its scalability, ease of use, and fully managed nature. Companies can focus on their applications rather than managing database infrastructure, making RDS a popular choice for businesses ranging from startups to large enterprises. Its high availability, security features, and integration with other AWS services like EC2 and S3 make it suitable for a wide range of applications,

including e-commerce platforms, content management systems, and data warehousing solutions.

8) DDL and DML Commands

a) **Use DDL commands to create a Library database and define a Books table with fields: BookID, Title, Author, Genre, and Price.**

```
-- Create a Library database
CREATE DATABASE Library;

-- Use the Library database
USE Library;

-- Create the Books table
CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(255),
    Author VARCHAR(255),
    Genre VARCHAR(100),
    Price DECIMAL(10, 2));
```

b) **Insert at least five sample records into the Books table using INSERT (DML) and verify them using a SELECT query.**

```
-- Insert sample records into the Books table
INSERT INTO Books (BookID, Title, Author, Genre, Price)
VALUES
(1, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Classic', 10.99),
(2, '1984', 'George Orwell', 'Dystopian', 12.99),
(3, 'The Catcher in the Rye', 'J.D. Salinger', 'Fiction', 8.49),
(4, 'To Kill a Mockingbird', 'Harper Lee', 'Fiction', 9.99),
(5, 'The Lean Startup', 'Eric Ries', 'Business', 15.99);

-- Verify the inserted records using SELECT
```

```
SELECT * FROM Books;
```

c) **A new column PublicationYear needs to be added. Use ALTER TABLE to modify the existing table structure.**

```
-- Add a new column PublicationYear to the Books table
ALTER TABLE Books
ADD COLUMN PublicationYear INT;

-- Verify the updated table structure
DESCRIBE Books;
```

d) **Update the price of all books published before 2020 by increasing 10% using the UPDATE statement.**

```
-- Update the price of books published before 2020 by increasing 10%
UPDATE Books
SET Price = Price * 1.10
WHERE PublicationYear < 2020;

-- Verify the updated prices
SELECT * FROM Books;
```

e) **Use DELETE to remove all books where the genre is 'Outdated Technology' and**

```
-- Delete all books where the genre is 'Outdated Technology'
DELETE FROM Books
WHERE Genre = 'Outdated Technology';

-- Verify the changes by checking the remaining books
SELECT * FROM Books;
```

Recent Tool/Technology for Database Management:

A recent tool I have studied for database management is **Microsoft SQL Server**.

Key Features of Microsoft SQL Server:

- **Comprehensive Data Management:** SQL Server provides robust relational database management capabilities with high security, scalability, and data integrity.
- **T-SQL (Transact-SQL):** SQL Server uses T-SQL, an extension of SQL, offering more procedural programming features, making it suitable for complex queries and stored procedures.
- **Security Features:** It has advanced security features like Transparent Data Encryption (TDE), Always Encrypted, and dynamic data masking to protect sensitive information.
- **High Availability & Disaster Recovery:** Features like Always On Availability Groups, database mirroring, and log shipping ensure high availability and data protection.
- **Data Integration:** It integrates seamlessly with other Microsoft tools like Power BI, Excel, and Azure, making it ideal for business analytics and reporting.
- **Performance Optimization:** It includes features like automatic tuning, query optimization, and in-memory OLTP for improved performance.

Why it's Used in the Industry:

Microsoft SQL Server is widely used in industries ranging from finance and healthcare to retail and government due to its reliability, scalability, and security. It is particularly favored by organizations already using other Microsoft technologies like .NET and Azure. SQL Server's enterprise-level features and strong support for business intelligence make it a popular choice for managing large and complex databases, handling critical workloads, and ensuring high uptime.

9) DDL and DML Commands (on a single table: Books)

a) Create a table Books using DDL with fields: BookID, Title, Author, Price, and StockAvailable.

```
-- Create the Books table with the required fields
CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(255),
    Author VARCHAR(255),
    Price DECIMAL(10, 2),
    StockAvailable INT
);
```

b) Insert 5 book records into the Books table using the INSERT command.

```
-- Insert 5 records into the Books table
INSERT INTO Books (BookID, Title, Author, Price, StockAvailable)
VALUES
(1, 'To Kill a Mockingbird', 'Harper Lee', 12.99, 20),
(2, '1984', 'George Orwell', 15.49, 5),
(3, 'The Great Gatsby', 'F. Scott Fitzgerald', 10.99, 0),
(4, 'Moby Dick', 'Herman Melville', 18.99, 8),
(5, 'War and Peace', 'Leo Tolstoy', 22.49, 2);
```

c) Modify the structure of the Books table by adding a new column Genre using the ALTER TABLE command.

-- Add a new column 'Genre' to the Books table

```
ALTER TABLE Books
ADD Genre VARCHAR(100);

-- Verify the updated table structure
DESCRIBE Books;
```

d) **Use the UPDATE command to increase the price of all books by RS 50 in the Books table.**

```
-- Increase the price of all books by RS 50
UPDATE Books
SET Price = Price + 50;

-- Verify the updated prices
SELECT * FROM Books;
```

e) **Delete all records from the Books table where StockAvailable is 0 using the DELETE command.**

```
-- Delete books where StockAvailable is 0
DELETE FROM Books
WHERE StockAvailable = 0;

-- Verify the deletion
SELECT * FROM Books;
```

Recent Tool/Technology for Database Management:

A recent tool I have studied for database management is **Amazon Aurora**.

Key Features of Amazon Aurora:

- **Managed Service:** Aurora is a fully managed database service by AWS that supports both MySQL and PostgreSQL, removing the complexities of managing database infrastructure.
- **High Performance and Scalability:** Aurora provides up to 5 times the performance of MySQL and 2 times that of PostgreSQL with cost-effectiveness. It scales automatically as per workload demands.
- **High Availability:** Aurora offers built-in fault tolerance and replication across multiple Availability Zones, ensuring high availability and durability.

- **Automated Backups and Snapshots:** Aurora automatically backs up data continuously to Amazon S3, allowing easy restoration to any point in time.
- **Security Features:** It integrates with AWS Identity and Access Management (IAM) for access control, provides encryption at rest and in transit, and supports VPC (Virtual Private Cloud) for network isolation.

Why it's Used in the Industry:

Amazon Aurora is widely used in industries due to its ability to provide a highly available, scalable, and low-cost relational database solution. It is especially beneficial for companies that already use AWS cloud infrastructure as it integrates well with other AWS services like Lambda, S3, and EC2. Aurora's high-performance capabilities make it ideal for businesses with growing data demands and mission-critical applications where high availability, security, and fault tolerance are paramount.

10) Analyze Sales Performance Using Aggregate Functions

a) **Calculate the total quantity of products sold across all transactions in the Sales table.**

```
-- Calculate total quantity of products sold
SELECT SUM(Quantity) AS TotalQuantitySold
FROM Sales;
```

b) **Find the average sale amount for transactions made in March 2025.**

```
-- Calculate the average sale amount for transactions made in March 2025
SELECT AVG(SaleAmount) AS AverageSaleAmount
FROM Sales
```

```
WHERE SaleDate BETWEEN '2025-03-01' AND '2025-03-31';
```

c) **Identify the product with the minimum sale quantity from the Sales table.**

```
-- Identify the product with the minimum sale quantity
SELECT ProductName, MIN(Quantity) AS MinimumQuantitySold
FROM Sales
GROUP BY ProductName
ORDER BY MinimumQuantitySold ASC
LIMIT 1;
```

d) **Determine the maximum discount offered in February 2025.**

```
-- Determine the maximum discount offered in February 2025
SELECT MAX(Discount) AS MaximumDiscount
FROM Sales
WHERE SaleDate BETWEEN '2025-02-01' AND '2025-02-28';
```

e) **Count how many sales were made by each salesperson using GROUP BY SalesPerson.**

```
-- Count the number of sales made by each salesperson
SELECT SalesPerson, COUNT(SaleID) AS SalesCount
FROM Sales
GROUP BY SalesPerson;
```

Recent Tool/Technology for Database Management:

A recent tool I have studied for database management is **Google Cloud Spanner**.

Key Features of Google Cloud Spanner:

- **Globally Distributed Database:** Cloud Spanner is designed to handle large-scale, globally distributed workloads with strong consistency and high availability across multiple regions.

- **Horizontal Scalability:** It provides automatic and horizontal scaling of storage and compute, which allows businesses to grow without worrying about database performance.
- **ACID Transactions:** Cloud Spanner ensures ACID compliance, which guarantees data consistency and integrity even in highly distributed environments.
- **SQL Support:** It supports SQL queries, which makes it easy for developers to interact with the database while maintaining scalability and performance.
- **Integrated Security and Encryption:** Cloud Spanner integrates with Google Cloud's security features, offering end-to-end encryption for data both at rest and in transit.

Why it's Used in the Industry:

Cloud Spanner is used in industries that require both relational database functionality and the scalability of NoSQL databases. Its global distribution and high availability features make it suitable for businesses operating across multiple regions. The strong consistency provided by ACID transactions is ideal for applications requiring reliable and consistent data, such as financial services, e-commerce, and large-scale enterprise applications. It is popular in organizations already leveraging Google Cloud for other services, as it integrates seamlessly into the Google Cloud ecosystem.