3) Create a Customer Table with Integrity Constraints a) Create a Customers table with constraints: CustomerID (PRIMARY KEY), Email (UNIQUE), Age (CHECK Age > 18). b) Insert a valid customer record and verify that the default country is assigned if not explicitly provided. c) Attempt to insert a customer with an age of 16 and observe the CHECK constraint violation. d) Try inserting two customers with the same email ID and observe the UNIQUE constraint violation. e) Retrieve all customers who are older than 25 and belong to a country other than 'India'.

Which recent tool or technology have you studied for database management, and can you briefly explain its key features and why it is used in the industry?

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(50),

Email VARCHAR(100) UNIQUE,

Age INT CHECK (Age > 18),

Country VARCHAR(50) DEFAULT 'India'

);

INSERT INTO Customers (CustomerID, Name, Email, Age)

VALUES (1, 'Rahul Sharma', 'rahul.sharma@email.com', 28);

-- Check the record

SELECT \* FROM Customers;

INSERT INTO Customers (CustomerID, Name, Email, Age)

VALUES (2, 'Ankit Patel', 'ankit.patel@email.com', 16);

-- First valid insert

INSERT INTO Customers (CustomerID, Name, Email, Age)

VALUES (2, 'Pooja Verma', 'pooja.verma@email.com', 30);

-- Second insert with same email (should fail)

INSERT INTO Customers (CustomerID, Name, Email, Age)

VALUES (3, 'Ravi Yadav', 'pooja.verma@email.com', 35);

SELECT \* FROM Customers

WHERE Age > 25 AND Country <> 'India';

**📚 Recently Studied Database Management Tool: Amazon RDS (Relational Database Service)**

✅ **Key Features:**

* **Fully Managed:** Automates patching, backups, and database maintenance.
* **Supports Multiple Engines:** MySQL, PostgreSQL, Oracle, SQL Server, MariaDB, and Amazon Aurora.
* **Auto-scaling Storage:** Grows storage as needed without downtime.
* **High Availability:** Multi-AZ deployments for disaster recovery and high durability.
* **Monitoring:** Integrated with AWS CloudWatch for real-time performance tracking.
* **Security:** Encryption at rest and in transit, IAM integration for access control.

✅ **Why it’s used in Industry:**

* Great for businesses wanting **scalable**, **secure**, and **low-maintenance** relational databases.
* Saves a lot of operational overhead — companies can focus more on development than on DB admin tasks.
* Perfect for production-ready apps, from startups to large enterprises