1.College

-- 1. Create the database

CREATE DATABASE IF NOT EXISTS CollegeDB;

USE CollegeDB;

-- 2. Drop tables if they already exist (for re-runs)

DROP TABLE IF EXISTS Student;

DROP TABLE IF EXISTS Department;

-- 3. Create Department table

CREATE TABLE Department (

dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(50) NOT NULL,

intake INT CHECK (intake >= 0)

);

-- 4. Create Student table with foreign key

CREATE TABLE Student (

sid INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

gender VARCHAR(10) CHECK (gender IN ('Male', 'Female')),

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES Department(dept\_id)

);

-- 5. Insert 3 departments

INSERT INTO Department (dept\_id, dept\_name, intake) VALUES

(101, 'Computer Science', 60),

(102, 'Electronics', 50),

(103, 'Mechanical', 55);

-- 6. Insert 5 students

INSERT INTO Student (sid, name, gender, dept\_id) VALUES

(1, 'Rahul Sharma', 'Male', 101),

(2, 'Priya Mehta', 'Female', 102),

(3, 'Aman Verma', 'Male', 101),

(4, 'Sneha Patil', 'Female', 103),

(5, 'Ravi Kumar', 'Male', 103);

-- 7. Display names of all male students and their department names

SELECT s.name AS student\_name, d.dept\_name

FROM Student s

JOIN Department d ON s.dept\_id = d.dept\_id

WHERE s.gender = 'Male';

-- 8. List departments with more than 2 students

SELECT d.dept\_name, COUNT(s.sid) AS student\_count

FROM Student s

JOIN Department d ON s.dept\_id = d.dept\_id

GROUP BY d.dept\_id, d.dept\_name

HAVING COUNT(s.sid) > 2;

-- 9. Update the intake to increase by 10% for all departments

UPDATE Department

SET intake = CEIL(intake \* 1.10);

2.retail

-- 1. Create the database

CREATE DATABASE IF NOT EXISTS RetailDB;

USE RetailDB;

-- 2. Drop tables if they already exist (for re-runs)

DROP TABLE IF EXISTS Orders;

DROP TABLE IF EXISTS Customers;

-- 3. Create Customers table

CREATE TABLE Customers (

cust\_id INT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

city VARCHAR(30)

);

-- 4. Create Orders table

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

cust\_id INT,

amount DECIMAL(10,2),

order\_date DATE,

FOREIGN KEY (cust\_id) REFERENCES Customers(cust\_id)

);

-- 5. Insert 4 customers

INSERT INTO Customers (cust\_id, name, city) VALUES

(1, 'Anjali Singh', 'Mumbai'),

(2, 'Rohan Mehta', 'Delhi'),

(3, 'Neha Kapoor', 'Pune'),

(4, 'Vikas Reddy', 'Hyderabad');

-- 6. Insert 5 orders

INSERT INTO Orders (order\_id, cust\_id, amount, order\_date) VALUES

(101, 1, 7200.50, '2024-11-01'),

(102, 2, 3000.00, '2024-11-03'),

(103, 1, 4500.00, '2024-11-05'),

(104, 3, 5500.75, '2024-11-07'),

(105, 2, 1500.00, '2024-11-09');

-- 7. Display customer names who placed orders above ₹5000

SELECT DISTINCT c.name

FROM Customers c

JOIN Orders o ON c.cust\_id = o.cust\_id

WHERE o.amount > 5000;

-- 8. List total order amount placed by each customer in descending order

SELECT c.name, SUM(o.amount) AS total\_amount

FROM Customers c

JOIN Orders o ON c.cust\_id = o.cust\_id

GROUP BY c.cust\_id, c.name

ORDER BY total\_amount DESC;

-- 9. Retrieve customers who haven’t placed any orders

SELECT c.name

FROM Customers c

LEFT JOIN Orders o ON c.cust\_id = o.cust\_id

WHERE o.order\_id IS NULL;

3.bookstorrrre

-- 1. Create the database

CREATE DATABASE IF NOT EXISTS BookstoreDB;

USE BookstoreDB;

-- 2. Drop tables if they already exist

DROP TABLE IF EXISTS Sales;

DROP TABLE IF EXISTS Books;

-- 3. Create Books table

CREATE TABLE Books (

book\_id INT PRIMARY KEY,

title VARCHAR(100) NOT NULL,

price DECIMAL(8,2) CHECK (price >= 0),

pub\_year INT

);

-- 4. Create Sales table with foreign key

CREATE TABLE Sales (

sale\_id INT PRIMARY KEY,

book\_id INT,

quantity INT CHECK (quantity > 0),

sale\_date DATE,

FOREIGN KEY (book\_id) REFERENCES Books(book\_id)

);

-- 5. Insert 4 books

INSERT INTO Books (book\_id, title, price, pub\_year) VALUES

(1, 'Learn MySQL', 499.50, 2022),

(2, 'Mastering Python', 750.00, 2021),

(3, 'Web Dev Basics', 399.00, 2020),

(4, 'Data Science Handbook', 899.99, 2023);

-- 6. Insert 5 sales records

INSERT INTO Sales (sale\_id, book\_id, quantity, sale\_date) VALUES

(101, 1, 3, '2024-01-15'),

(102, 2, 2, '2023-12-10'),

(103, 3, 5, '2024-03-20'),

(104, 1, 4, '2024-04-02'),

(105, 4, 1, '2023-11-30');

-- 7. Display titles of books sold in the year 2024

SELECT DISTINCT b.title

FROM Books b

JOIN Sales s ON b.book\_id = s.book\_id

WHERE YEAR(s.sale\_date) = 2024;

-- 8. Show total sales revenue for each book

SELECT b.title, SUM(b.price \* s.quantity) AS total\_revenue

FROM Books b

JOIN Sales s ON b.book\_id = s.book\_id

GROUP BY b.book\_id, b.title;

-- 9. Find the title of the most sold book

SELECT b.title, SUM(s.quantity) AS total\_quantity

FROM Books b

JOIN Sales s ON b.book\_id = s.book\_id

GROUP BY b.book\_id, b.title

ORDER BY total\_quantity DESC

LIMIT 1;

4.airline

-- 1. Create database

CREATE DATABASE IF NOT EXISTS AirlineDB;

USE AirlineDB;

-- 2. Drop tables if they already exist

DROP TABLE IF EXISTS Passengers;

DROP TABLE IF EXISTS Flights;

-- 3. Create Flights table

CREATE TABLE Flights (

flight\_id INT PRIMARY KEY,

source VARCHAR(30),

destination VARCHAR(30),

fare DECIMAL(6,2) CHECK (fare >= 0)

);

-- 4. Create Passengers table with foreign key

CREATE TABLE Passengers (

pid INT PRIMARY KEY,

name VARCHAR(50),

flight\_id INT,

travel\_date DATE,

FOREIGN KEY (flight\_id) REFERENCES Flights(flight\_id)

);

-- 5. Insert 3 flights

INSERT INTO Flights (flight\_id, source, destination, fare) VALUES

(1, 'Mumbai', 'Delhi', 4500.00),

(2, 'Chennai', 'Bangalore', 3200.00),

(3, 'Kolkata', 'Delhi', 4700.00);

-- 6. Insert 5 passenger bookings

INSERT INTO Passengers (pid, name, flight\_id, travel\_date) VALUES

(101, 'Aman Verma', 1, '2024-05-10'),

(102, 'Sneha Iyer', 2, '2024-05-12'),

(103, 'Ritika Mehta', 1, '2024-05-15'),

(104, 'Raj Malhotra', 3, '2024-05-18'),

(105, 'Ankit Sharma', 1, '2024-05-20');

-- 7. List all passengers travelling to ‘Delhi’

SELECT p.name, f.destination

FROM Passengers p

JOIN Flights f ON p.flight\_id = f.flight\_id

WHERE f.destination = 'Delhi';

-- 8. Show flight-wise passenger count

SELECT f.flight\_id, COUNT(p.pid) AS passenger\_count

FROM Flights f

LEFT JOIN Passengers p ON f.flight\_id = p.flight\_id

GROUP BY f.flight\_id;

-- 9. Increase fare by 10% for flights having more than 2 bookings

UPDATE Flights

SET fare = fare \* 1.10

WHERE flight\_id IN (

SELECT f.flight\_id

FROM Flights f

JOIN Passengers p ON f.flight\_id = p.flight\_id

GROUP BY f.flight\_id

HAVING COUNT(p.pid) > 2

);

5.Employee performance

-- 1. Create the database

CREATE DATABASE IF NOT EXISTS EmployeeDB;

USE EmployeeDB;

-- 2. Drop tables if they already exist

DROP TABLE IF EXISTS Performance;

DROP TABLE IF EXISTS Employee;

-- 3. Create Employee table

CREATE TABLE Employee (

emp\_id INT PRIMARY KEY,

name VARCHAR(50),

designation VARCHAR(30),

salary INT CHECK (salary > 0)

);

-- 4. Create Performance table

CREATE TABLE Performance (

emp\_id INT,

month VARCHAR(15),

rating INT CHECK (rating BETWEEN 1 AND 5),

FOREIGN KEY (emp\_id) REFERENCES Employee(emp\_id)

);

-- 5. Insert sample data into Employee

INSERT INTO Employee (emp\_id, name, designation, salary) VALUES

(1, 'Ravi Kumar', 'Manager', 60000),

(2, 'Anjali Mehta', 'Developer', 45000),

(3, 'Suresh Singh', 'Analyst', 40000),

(4, 'Pooja Sharma', 'Manager', 65000),

(5, 'Nikhil Jain', 'Developer', 42000);

-- 6. Insert sample data into Performance

INSERT INTO Performance (emp\_id, month, rating) VALUES

(1, 'January', 5),

(2, 'January', 4),

(3, 'January', 3),

(1, 'February', 4),

(2, 'February', 5),

(4, 'February', 5),

(2, 'March', 5),

(4, 'March', 4);

-- 7. Find employees with average rating > 4

SELECT e.name, AVG(p.rating) AS avg\_rating

FROM Employee e

JOIN Performance p ON e.emp\_id = p.emp\_id

GROUP BY e.emp\_id, e.name

HAVING AVG(p.rating) > 4;

-- 8. Display highest rated employee each month

SELECT p.month, e.name, p.rating

FROM Performance p

JOIN Employee e ON e.emp\_id = p.emp\_id

WHERE (p.month, p.rating) IN (

SELECT month, MAX(rating)

FROM Performance

GROUP BY month

);

-- 9. List employees who never received a rating using NOT IN

SELECT name

FROM Employee

WHERE emp\_id NOT IN (SELECT DISTINCT emp\_id FROM Performance);

-- 10. Display total salary to be paid for ‘Manager’ designation employees

SELECT SUM(salary) AS total\_manager\_salary

FROM Employee

WHERE designation = 'Manager';