1. Library Management System

```
CREATE TABLE Authors (
   AuthorID INT PRIMARY KEY,
   Name VARCHAR(100)
);

CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(100),
   AuthorID INT,
   Price DECIMAL(10,2),
   FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);

CREATE TABLE Issued (
   BookID INT,
   StudentID INT,
   IssueDate DATE,
   PRIMARY KEY (BookID, StudentID),
   FOREIGN KEY (BookID) REFERENCES Books(BookID)
);

-- Authors
INSERT INTO Authors VALUES (1, 'R.K. Narayan');
INSERT INTO Authors VALUES (2, 'Chetan Bhagat');
INSERT INTO Authors VALUES (3, 'J.K. Rowling');

-- Books
INSERT INTO Books VALUES (101, 'Malgudi Days', 1, 450);
```

```
INSERT INTO Books VALUES (102, '2 States', 2, 550);
INSERT INTO Books VALUES (103, 'Harry Potter', 3, 950);

-- Issued
INSERT INTO Issued VALUES (101, 101, '2025-01-15');
INSERT INTO Issued VALUES (102, 101, '2025-02-10');
INSERT INTO Issued VALUES (103, 102, '2025-03-05');

SELECT * FROM Books WHERE Price > 500;
SELECT * FROM Issued WHERE StudentID = 101;
SELECT COUNT(*) AS TotalIssuedBooks FROM Issued;
SELECT B.Title FROM Books B
JOIN Issued I ON B.BookID = I.BookID
WHERE MONTH(I.IssueDate) = 1
INTERSECT
SELECT B.Title FROM Books B
JOIN Issued I ON B.BookID = I.BookID
WHERE MONTH(I.IssueDate) = 2;

SELECT B.Title, A.Name AS AuthorName
FROM Books B
JOIN Authors A ON B.AuthorID = A.AuthorID;
```

2: Hospital Management System

```
CREATE TABLE Patients (
   PatientID INT PRIMARY KEY,
   Name VARCHAR(100),
   Age INT,
   Gender VARCHAR(10)
);

CREATE TABLE Doctors (
   DoctorID INT PRIMARY KEY,
   Name VARCHAR(100),
   Specialization VARCHAR(50)
);

CREATE TABLE Appointments (
   AppointmentID INT PRIMARY KEY,
   PatientID INT,
   DoctorID INT,
   Date DATE,
   FOREIGN KEY (PatientID) REFERENCES Patients(PatientID),
   FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
);

-- Patients
INSERT INTO Patients VALUES (1, 'Ravi', 65, 'Male');
INSERT INTO Patients VALUES (2, 'Neha', 40, 'Female');
INSERT INTO Patients VALUES (3, 'Amit', 70, 'Male');

-- Doctors
```

```
INSERT INTO Doctors VALUES (101, 'Dr. Shah', 'Cardiology');
INSERT INTO Doctors VALUES (202, 'Dr. Mehta', 'Neurology');
INSERT INTO Doctors VALUES (303, 'Dr. Gupta', 'Orthopedic');

-- Appointments
INSERT INTO Appointments VALUES (1001, 1, 202, '2025-03-15');
INSERT INTO Appointments VALUES (1002, 2, 101, '2025-04-05');
INSERT INTO Appointments VALUES (1003, 3, 202, '2025-04-20');

SELECT * FROM Patients WHERE Age > 60;
SELECT * FROM Appointments WHERE DoctorID = 202;

SELECT DoctorID, COUNT(*) AS TotalAppointments
FROM Appointments
GROUP BY DoctorID;

SELECT * FROM Appointments WHERE MONTH(Date) = 3
INTERSECT
SELECT * FROM Appointments WHERE MONTH(Date) = 4;

SELECT P.Name AS PatientName, D.Name AS DoctorName
FROM Appointments A
JOIN Patients P ON A.PatientID = P.PatientID
JOIN Doctors D ON A.DoctorID = D.DoctorID;
```

3: Online Course Platform

```
CREATE TABLE Courses (
   CourseID VARCHAR(10) PRIMARY KEY,
   Title VARCHAR(100),
   Duration INT  -- in months
);

CREATE TABLE Instructors (
   InstructorID INT PRIMARY KEY,
   Name VARCHAR(100)
);

CREATE TABLE Enrollments (
   CourseID VARCHAR(10),
   StudentID INT,
   EnrollDate DATE,
   PRIMARY KEY (CourseID, StudentID),
   FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
);

-- Courses
INSERT INTO Courses VALUES ('C101', 'DBMS', 4);
INSERT INTO Courses VALUES ('C102', 'Java Programming', 3);
INSERT INTO Courses VALUES ('C103', 'Python', 5);

-- Instructors
INSERT INTO Instructors VALUES (1, 'Prof. Sharma');
INSERT INTO Instructors VALUES (2, 'Prof. Joshi');
INSERT INTO Instructors VALUES (3, 'Prof. Deshmukh');
```

```
-- Enrollments
INSERT INTO Enrollments VALUES ('C101', 101, '2025-04-01');
INSERT INTO Enrollments VALUES ('C102', 102, '2025-04-05');
INSERT INTO Enrollments VALUES ('C101', 103, '2025-04-08');

SELECT StudentID FROM Enrollments WHERE CourseID = 'C101';
SELECT * FROM Courses WHERE Duration > 3;

SELECT CourseID, COUNT(StudentID) AS TotalEnrolled
FROM Enrollments
GROUP BY CourseID;

SELECT StudentID FROM Enrollments WHERE CourseID = 'C101'
INTERSECT
SELECT StudentID FROM Enrollments WHERE CourseID = 'C102';

-- Let's assume this relation temporarily:
-- CourseInstructor(CourseID, InstructorID)

-- Create temporary relation:
CREATE TABLE CourseInstructor (
   CourseID VARCHAR(10),
   InstructorID INT,
   FOREIGN KEY (CourseID) REFERENCES Courses(CourseID),
   FOREIGN KEY (InstructorID) REFERENCES Instructors(InstructorID)
);

-- Sample data
```

```
INSERT INTO CourseInstructor VALUES ('C101', 1);
INSERT INTO CourseInstructor VALUES ('C102', 2);
INSERT INTO CourseInstructor VALUES ('C103', 3);

-- Query:
SELECT C.Title, I.Name AS InstructorName
FROM Courses C
JOIN CourseInstructor CI ON C.CourseID = CI.CourseID
JOIN Instructors I ON CI.InstructorID = I.InstructorID;
```

4: Retail Store Inventory

```
CREATE TABLE Categories (
   CategoryID INT PRIMARY KEY,
   Name VARCHAR(50)
);

CREATE TABLE Products (
   ProductID INT PRIMARY KEY,
   Name VARCHAR(100),
   CategoryID INT,
   Price DECIMAL(10,2),
   FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

CREATE TABLE Sales (
   SaleID INT PRIMARY KEY,
   ProductID INT,
   Quantity INT,
   SaleDate DATE,
   FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);

INSERT INTO Categories VALUES (1, 'Electronics'), (2, 'Furniture'), (3, 'Clothing');

INSERT INTO Products VALUES
(101, 'Laptop', 1, 55000),
(102, 'Chair', 2, 1500),
(103, 'T-Shirt', 3, 500);
```

```
INSERT INTO Sales VALUES
(1, 101, 2, '2025-03-01'),
(2, 102, 5, '2025-03-01'),
(3, 103, 10, '2025-03-10');

-- 1. Products priced above ₹1000
SELECT * FROM Products WHERE Price > 1000;

-- 2. Products sold on '2025-03-01'
SELECT * FROM Sales WHERE SaleDate = '2025-03-01';

-- 3. Total quantity sold per product
SELECT ProductID, SUM(Quantity) AS TotalSold FROM Sales GROUP BY ProductID;

-- 4. Products sold in Store A but not in Store B (EXCEPT)
-- You'll need Store info in a real schema, skipping here as not defined.

-- 5. Product names with their category names
SELECT P.Name AS ProductName, C.Name AS CategoryName
FROM Products P
JOIN Categories C ON P.CategoryID = C.CategoryID;
```

5: College Examination System

```
CREATE TABLE Students (
   StudentID INT PRIMARY KEY,
   Name VARCHAR(100),
   Dept VARCHAR(50)
);

CREATE TABLE Subjects (
   SubjectID INT PRIMARY KEY,
   Name VARCHAR(100)
);

CREATE TABLE Marks (
   StudentID INT,
   SubjectID INT,
   MarksObtained INT,
   PRIMARY KEY (StudentID, SubjectID),
   FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
   FOREIGN KEY (SubjectID) REFERENCES Subjects(SubjectID)
);



INSERT INTO Students VALUES (101, 'Amit', 'CS'), (102, 'Sneha', 'IT'), (103, 'Rahul',
'CS');

INSERT INTO Subjects VALUES (201, 'DBMS'), (202, 'OS'), (203, 'CN');

INSERT INTO Marks VALUES
(101, 201, 78),
```

```
(101, 202, 32),
(102, 201, 88);

-- 1. Students scoring above 75 in any subject
SELECT * FROM Marks WHERE MarksObtained > 75;

-- 2. Subjects where marks are below 35
SELECT SubjectID FROM Marks WHERE MarksObtained < 35;

-- 3. Average marks per subject
SELECT SubjectID, AVG(MarksObtained) AS AverageMarks FROM Marks GROUP BY SubjectID;

-- 4. Students who appeared in both Sub A and B (INTERSECT)
SELECT StudentID FROM Marks WHERE SubjectID = 201
INTERSECT
SELECT StudentID FROM Marks WHERE SubjectID = 202;

-- 5. Student names with subject names and their marks
SELECT S.Name, Sub.Name, M.MarksObtained
FROM Marks M
JOIN Students S ON M.StudentID = S.StudentID
JOIN Subjects Sub ON M.SubjectID = Sub.SubjectID;
```

6: Bank Transactions

```
CREATE TABLE Customers (
   CustomerID INT PRIMARY KEY,
   Name VARCHAR(100),
   Address VARCHAR(100)
);

CREATE TABLE Accounts (
   AccountID INT PRIMARY KEY,
   CustomerID INT,
   Balance DECIMAL(10,2),
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE Transactions (
   TransID INT PRIMARY KEY,
   AccountID INT,
   Amount DECIMAL(10,2),
   TransDate DATE,
   Type VARCHAR(10), -- Debit or Credit
   FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
);

INSERT INTO Customers VALUES (1, 'Rohan', 'Pune'), (2, 'Sneha', 'Mumbai');

INSERT INTO Accounts VALUES (101, 1, 15000), (102, 2, 20000);

INSERT INTO Transactions VALUES
```

```
(1001, 101, 2000, '2025-04-01', 'Debit'),
(1002, 101, 3000, '2025-04-05', 'Credit'),
(1003, 102, 5000, '2025-04-10', 'Debit');


-- 1. Customers from 'Pune'
SELECT * FROM Customers WHERE Address = 'Pune';

-- 2. Transactions for Account ID = 101
SELECT * FROM Transactions WHERE AccountID = 101;

-- 3. Total amount transacted per account
SELECT AccountID, SUM(Amount) AS TotalAmount FROM Transactions GROUP BY AccountID;

-- 4. Accounts with debit but not credit transactions (EXCEPT)
SELECT AccountID FROM Transactions WHERE Type = 'Debit'
EXCEPT
SELECT AccountID FROM Transactions WHERE Type = 'Credit';

-- 5. Customers with their account balances
SELECT C.Name, A.Balance FROM Customers C
JOIN Accounts A ON C.CustomerID = A.CustomerID;
```

7: Movie Booking System

```
CREATE TABLE Movies (
   MovieID VARCHAR(10) PRIMARY KEY,
   Title VARCHAR(100),
   Duration INT -- in minutes
);

CREATE TABLE Theaters (
   TheaterID VARCHAR(10) PRIMARY KEY,
   Name VARCHAR(100),
   City VARCHAR(50)
);

CREATE TABLE Bookings (
   BookingID INT PRIMARY KEY,
   MovieID VARCHAR(10),
   TheaterID VARCHAR(10),
   ShowDate DATE,
   TicketsBooked INT,
   FOREIGN KEY (MovieID) REFERENCES Movies(MovieID),
   FOREIGN KEY (TheaterID) REFERENCES Theaters(TheaterID)
);


INSERT INTO Movies VALUES ('M101', 'Inception', 150), ('M102', 'Interstellar', 180);

INSERT INTO Theaters VALUES ('T1', 'PVR', 'Pune'), ('T2', 'INOX', 'Mumbai');

INSERT INTO Bookings VALUES
```

```
(1, 'M101', 'T1', '2025-04-01', 100),
(2, 'M101', 'T2', '2025-04-02', 150),
(3, 'M102', 'T1', '2025-04-03', 80);

-- 1. Movies longer than 2 hours
SELECT * FROM Movies WHERE Duration > 120;

-- 2. Bookings for movie ID 'M101'
SELECT * FROM Bookings WHERE MovieID = 'M101';

-- 3. Total tickets booked per movie
SELECT MovieID, SUM(TicketsBooked) AS TotalTickets FROM Bookings GROUP BY MovieID;

-- 4. Movies booked in Theater A or B (UNION)
SELECT MovieID FROM Bookings WHERE TheaterID = 'T1'
UNION
SELECT MovieID FROM Bookings WHERE TheaterID = 'T2';

-- 5. Movie titles with theater names
SELECT M.Title, T.Name AS TheaterName
FROM Bookings B
JOIN Movies M ON B.MovieID = M.MovieID
JOIN Theaters T ON B.TheaterID = T.TheaterID;
```

8: Online Shopping Portal

```
CREATE TABLE Customers (
   CustomerID INT PRIMARY KEY,
   Name VARCHAR(100),
   Email VARCHAR(100)
);

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

CREATE TABLE Orders (
   OrderID INT PRIMARY KEY,
   CustomerID INT,
   ProductID INT,
   OrderDate DATE,
   Quantity INT,
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
   FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);


INSERT INTO Customers VALUES (1, 'Amit', 'amit@mail.com'), (2, 'Neha',
'neha@mail.com');

INSERT INTO Products VALUES (101, 'Phone', 15000), (102, 'Headphones', 2000);
```

```
INSERT INTO Orders VALUES
(1001, 1, 101, '2025-04-05', 1),
(1002, 2, 102, '2025-04-02', 6),
(1003, 1, 102, '2025-04-10', 2);

-- 1. Customers who ordered after '2025-04-01'
SELECT DISTINCT C.Name FROM Customers C
JOIN Orders O ON C.CustomerID = O.CustomerID
WHERE OrderDate > '2025-04-01';

-- 2. Products ordered more than 5 times
SELECT ProductID FROM Orders GROUP BY ProductID HAVING SUM(Quantity) > 5;

-- 3. Total revenue per product
SELECT P.Name, SUM(O.Quantity * P.Price) AS TotalRevenue
FROM Products P
JOIN Orders O ON P.ProductID = O.ProductID
GROUP BY P.Name;

-- 4. Customers who ordered Product A or B (UNION)
SELECT CustomerID FROM Orders WHERE ProductID = 101
UNION
SELECT CustomerID FROM Orders WHERE ProductID = 102;

-- 5. Customers with products they ordered
SELECT C.Name AS CustomerName, P.Name AS ProductName
FROM Orders O
JOIN Customers C ON O.CustomerID = C.CustomerID
JOIN Products P ON O.ProductID = P.ProductID;
```

9: Vehicle Service Center

```
CREATE TABLE Vehicles (
   VehicleID INT PRIMARY KEY,
   OwnerName VARCHAR(100),
   Model VARCHAR(50)
);

CREATE TABLE ServiceTypes (
   ServiceID INT PRIMARY KEY,
   Description VARCHAR(100)
);

CREATE TABLE Appointments (
   AppointmentID INT PRIMARY KEY,
   VehicleID INT,
   ServiceID INT,
   ServiceDate DATE,
   FOREIGN KEY (VehicleID) REFERENCES Vehicles(VehicleID),
   FOREIGN KEY (ServiceID) REFERENCES ServiceTypes(ServiceID)
);

INSERT INTO Vehicles VALUES (1, 'Ravi', 'Swift'), (2, 'Anita', 'i20');

INSERT INTO ServiceTypes VALUES (501, 'Oil Change'), (502, 'Wheel Alignment');

INSERT INTO Appointments VALUES
(101, 1, 501, '2025-04-01'),
(102, 1, 502, '2025-04-05'),
(103, 2, 501, '2025-04-03');
```

```
-- 1. Vehicles of model 'Swift'
SELECT * FROM Vehicles WHERE Model = 'Swift';

-- 2. Appointments for Service ID = 501
SELECT * FROM Appointments WHERE ServiceID = 501;

-- 3. Total appointments per service type
SELECT ServiceID, COUNT(*) AS TotalAppointments FROM Appointments GROUP BY ServiceID;

-- 4. Vehicles that had both Service A and B (INTERSECT)
SELECT VehicleID FROM Appointments WHERE ServiceID = 501
INTERSECT
SELECT VehicleID FROM Appointments WHERE ServiceID = 502;

-- 5. Vehicle owners with services done
SELECT V.OwnerName, S.Description
FROM Appointments A
JOIN Vehicles V ON A.VehicleID = V.VehicleID
JOIN ServiceTypes S ON A.ServiceID = S.ServiceID;
```

10: Hotel Reservation System

```
CREATE TABLE Rooms (
   RoomID VARCHAR(10) PRIMARY KEY,
   Type VARCHAR(50),
   Price DECIMAL(10,2)
);

CREATE TABLE Guests (
   GuestID INT PRIMARY KEY,
   Name VARCHAR(100),
   Phone VARCHAR(15)
);

CREATE TABLE Reservations (
   ReservationID INT PRIMARY KEY,
   RoomID VARCHAR(10),
   GuestID INT,
   CheckInDate DATE,
   CheckOutDate DATE,
   FOREIGN KEY (RoomID) REFERENCES Rooms(RoomID),
   FOREIGN KEY (GuestID) REFERENCES Guests(GuestID)
);

INSERT INTO Rooms VALUES ('A1', 'Deluxe', 2500), ('B1', 'Standard', 1800);

INSERT INTO Guests VALUES (1, 'Raj', '9876543210'), (2, 'Simran', '9123456780');

INSERT INTO Reservations VALUES
(1, 'A1', 1, '2025-03-10', '2025-03-12'),
```

(2, 'B1', 2, '2025-03-15', '2025-03-17');

-- 1. Rooms priced above ₹2000

SELECT * FROM Rooms WHERE Price > 2000;

-- 2. Reservations made in March

SELECT * FROM Reservations WHERE MONTH(CheckInDate) = 3;

-- 3. Total bookings per room

SELECT RoomID, COUNT(*) AS TotalBookings FROM Reservations GROUP BY RoomID;

-- 4. Guests who stayed in Room A or B (UNION)

SELECT GuestID FROM Reservations WHERE RoomID = 'A1'

UNION

SELECT GuestID FROM Reservations WHERE RoomID = 'B1';

-- 5. Guest names with room types reserved

SELECT G.Name, R.Type

FROM Reservations Res

JOIN Guests G ON Res.GuestID = G.GuestID

JOIN Rooms R ON Res.RoomID = R.RoomID;

11: Gym Management System

```
CREATE TABLE Members (
   MemberID INT PRIMARY KEY,
   Name VARCHAR(100),
   Age INT
);

CREATE TABLE Trainers (
   TrainerID INT PRIMARY KEY,
   Name VARCHAR(100),
   Expertise VARCHAR(50)
);

CREATE TABLE Subscriptions (
   MemberID INT,
   TrainerID INT,
   Type VARCHAR(50),
   StartDate DATE,
   PRIMARY KEY (MemberID, TrainerID),
   FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
   FOREIGN KEY (TrainerID) REFERENCES Trainers(TrainerID)
);

INSERT INTO Members VALUES (1, 'Arjun', 35), (2, 'Meera', 28), (3, 'Kunal', 40);

INSERT INTO Trainers VALUES (101, 'Ravi', 'Cardio'), (102, 'Nisha', 'Yoga');

INSERT INTO Subscriptions VALUES
(1, 101, 'Basic', '2025-04-01'),
```

```
(1, 102, 'Premium', '2025-04-05'),
(2, 101, 'Basic', '2025-04-10');

SELECT * FROM Members WHERE Age > 30;

SELECT * FROM Trainers WHERE Expertise = 'Cardio';

SELECT Type, COUNT(MemberID) AS MemberCount FROM Subscriptions GROUP BY Type;

SELECT MemberID FROM Subscriptions WHERE Type = 'Basic'
INTERSECT
SELECT MemberID FROM Subscriptions WHERE Type = 'Premium';

SELECT M.Name, T.Name AS TrainerName
FROM Subscriptions S
JOIN Members M ON S.MemberID = M.MemberID
JOIN Trainers T ON S.TrainerID = T.TrainerID;
```

12: Library Fines System

```
CREATE TABLE Students (
   StudentID INT PRIMARY KEY,
   Name VARCHAR(100)
);

CREATE TABLE Books (
   BookID INT PRIMARY KEY,
   Title VARCHAR(100)
);

CREATE TABLE Fines (
   StudentID INT,
   BookID INT,
   ReturnDate DATE,
   FineAmount DECIMAL(10,2),
   PRIMARY KEY (StudentID, BookID),
   FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
   FOREIGN KEY (BookID) REFERENCES Books(BookID)
);



INSERT INTO Students VALUES (1, 'Kiran'), (2, 'Neha');

INSERT INTO Books VALUES (101, 'DBMS'), (102, 'Java');

INSERT INTO Fines VALUES
(1, 101, '2025-04-10', 120),
(1, 102, '2025-04-12', 80),
```

```
(2, 101, '2025-04-15', 150);



SELECT * FROM Fines WHERE FineAmount > 0;


SELECT * FROM Fines WHERE FineAmount > 100;


SELECT StudentID, SUM(FineAmount) AS TotalFine FROM Fines GROUP BY
StudentID;


SELECT StudentID FROM Fines WHERE BookID = 101

INTERSECT

SELECT StudentID FROM Fines WHERE BookID = 102;


SELECT S.Name, B.Title, F.FineAmount

FROM Fines F

JOIN Students S ON F.StudentID = S.StudentID

JOIN Books B ON F.BookID = B.BookID;
```

13: Music Streaming Service

```
CREATE TABLE Songs (
   SongID INT PRIMARY KEY,
   Title VARCHAR(100),
   Duration INT
);

CREATE TABLE Artists (
   ArtistID INT PRIMARY KEY,
   Name VARCHAR(100)
);

CREATE TABLE Playlists (
   UserID INT,
   SongID INT,
   PRIMARY KEY (UserID, SongID),
   FOREIGN KEY (SongID) REFERENCES Songs(SongID)
);


INSERT INTO Songs VALUES (1, 'Track A', 330), (2, 'Track B', 400);

INSERT INTO Artists VALUES (301, 'Arijit Singh'), (302, 'Shreya Ghoshal');

-- Assume Artist info linked with song externally if needed

INSERT INTO Playlists VALUES (101, 1), (101, 2), (102, 2);
```

```
SELECT * FROM Songs WHERE Duration > 300;


SELECT * FROM Songs WHERE SongID IN (SELECT SongID FROM Songs WHERE
SongID = 1 AND EXISTS (SELECT * FROM Artists WHERE ArtistID = 301));


SELECT UserID, COUNT(*) AS TotalSongs FROM Playlists GROUP BY UserID;


SELECT UserID FROM Playlists WHERE SongID = 1

INTERSECT

SELECT UserID FROM Playlists WHERE SongID = 2;


-- Assume Songs table has ArtistID to join:

-- ALTER TABLE Songs ADD ArtistID INT;

-- JOIN query:

-- SELECT S.Title, A.Name

-- FROM Songs S

-- JOIN Artists A ON S.ArtistID = A.ArtistID;
```

14: School Transport System

```
CREATE TABLE Students (
   StudentID INT PRIMARY KEY,
   Name VARCHAR(100),
   Class VARCHAR(20)
);

CREATE TABLE Routes (
   RouteID VARCHAR(10) PRIMARY KEY,
   StartPoint VARCHAR(100),
   EndPoint VARCHAR(100)
);

CREATE TABLE Assignments (
   StudentID INT,
   RouteID VARCHAR(10),
   BusNumber VARCHAR(10),
   PRIMARY KEY (StudentID, RouteID),
   FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
   FOREIGN KEY (RouteID) REFERENCES Routes(RouteID)
);


INSERT INTO Students VALUES (1, 'Rohan', '10A'), (2, 'Anjali', '10B');

INSERT INTO Routes VALUES ('R01', 'Aundh', 'Wakad'), ('R02', 'Kothrud', 'FC Road');

INSERT INTO Assignments VALUES (1, 'R01', 'B001'), (2, 'R02', 'B002');
```

```
SELECT * FROM Assignments WHERE RouteID = 'R01';


-- Driver info not in schema; this question may be skipped or driver column added.


SELECT RouteID, COUNT(StudentID) AS TotalStudents FROM Assignments GROUP
BY RouteID;


SELECT StudentID FROM Assignments WHERE RouteID = 'R01'

INTERSECT

SELECT StudentID FROM Assignments WHERE RouteID = 'R02';


SELECT S.Name, R.StartPoint, R.EndPoint

FROM Assignments A

JOIN Students S ON A.StudentID = S.StudentID

JOIN Routes R ON A.RouteID = R.RouteID;
```

15: Freelance Project Tracker

```
CREATE TABLE Freelancers (
   FID INT PRIMARY KEY,
   Name VARCHAR(100),
   Skill VARCHAR(50)
);

CREATE TABLE Projects (
   PID INT PRIMARY KEY,
   Title VARCHAR(100),
   Deadline DATE
);

CREATE TABLE Assignments (
   FID INT,
   PID INT,
   PRIMARY KEY (FID, PID),
   FOREIGN KEY (FID) REFERENCES Freelancers(FID),
   FOREIGN KEY (PID) REFERENCES Projects(PID)
);
```

INSERT INTO Freelancers VALUES (1, 'Mehul', 'Web Development'), (2, 'Sara', 'Graphic Design');

INSERT INTO Projects VALUES (101, 'Website Redesign', '2025-04-20'), (102, 'Logo Design', '2025-04-18');

INSERT INTO Assignments VALUES (1, 101), (1, 102), (2, 102);

```
SELECT * FROM Freelancers WHERE Skill = 'Web Development';

SELECT * FROM Projects WHERE MONTH(Deadline) = 4;

SELECT FID, COUNT(*) AS TotalProjects FROM Assignments GROUP BY FID;

SELECT FID FROM Assignments WHERE PID = 101
INTERSECT
SELECT FID FROM Assignments WHERE PID = 102;

SELECT F.Name, P.Title
FROM Assignments A
JOIN Freelancers F ON A.FID = F.FID
JOIN Projects P ON A.PID = P.PID;
```

16: College Event Management

```
CREATE TABLE Events (
   EventID INT PRIMARY KEY,
   Name VARCHAR(100),
   Date DATE
);

CREATE TABLE Students (
   StudentID INT PRIMARY KEY,
   Name VARCHAR(100)
);

CREATE TABLE Participation (
   EventID INT,
   StudentID INT,
   PRIMARY KEY (EventID, StudentID),
   FOREIGN KEY (EventID) REFERENCES Events(EventID),
   FOREIGN KEY (StudentID) REFERENCES Students(StudentID)
);
```

```
INSERT INTO Events VALUES (1, 'TechFest', '2025-04-10'), (2, 'CulturalFest', '2025-04-15');

INSERT INTO Students VALUES (101, 'Karan'), (102, 'Aarti');

INSERT INTO Participation VALUES (1, 101), (2, 101), (1, 102);
```

```
SELECT * FROM Events WHERE Date > '2025-04-01';


SELECT S.Name FROM Participation P
JOIN Students S ON P.StudentID = S.StudentID
JOIN Events E ON P.EventID = E.EventID
WHERE E.Name = 'TechFest';


SELECT EventID, COUNT(StudentID) AS Participants FROM Participation GROUP BY
EventID;


SELECT StudentID FROM Participation WHERE EventID = 1
INTERSECT
SELECT StudentID FROM Participation WHERE EventID = 2;


SELECT S.Name AS StudentName, E.Name AS EventName
FROM Participation P
JOIN Students S ON P.StudentID = S.StudentID
JOIN Events E ON P.EventID = E.EventID;
```

17: University Admission Portal

```
CREATE TABLE Applicants (
   ApplicantID INT PRIMARY KEY,
   Name VARCHAR(100),
   Score INT
);

CREATE TABLE Departments (
   DeptID VARCHAR(10) PRIMARY KEY,
   Name VARCHAR(100)
);

CREATE TABLE Admissions (
   ApplicantID INT,
   DeptID VARCHAR(10),
   Status VARCHAR(20),
   PRIMARY KEY (ApplicantID, DeptID),
   FOREIGN KEY (ApplicantID) REFERENCES Applicants(ApplicantID),
   FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);


INSERT INTO Applicants VALUES (1, 'Nikhil', 90), (2, 'Divya', 82);

INSERT INTO Departments VALUES ('CSE', 'Computer Science'), ('ECE', 'Electronics');

INSERT INTO Admissions VALUES (1, 'CSE', 'Admitted'), (2, 'ECE', 'Pending');
```

```
SELECT * FROM Applicants WHERE Score > 85;

SELECT A.Name FROM Admissions AD
JOIN Applicants A ON AD.ApplicantID = A.ApplicantID
JOIN Departments D ON AD.DeptID = D.DeptID
WHERE D.Name = 'Computer Science';

SELECT DeptID, AVG(Score) AS AvgScore
FROM Admissions AD
JOIN Applicants A ON AD.ApplicantID = A.ApplicantID
GROUP BY DeptID;

SELECT ApplicantID FROM Admissions WHERE DeptID = 'CSE'
INTERSECT
SELECT ApplicantID FROM Admissions WHERE DeptID = 'ECE';

SELECT A.Name, D.Name AS Department, AD.Status
FROM Admissions AD
JOIN Applicants A ON AD.ApplicantID = A.ApplicantID
JOIN Departments D ON AD.DeptID = D.DeptID;
```

18: Restaurant Ordering System

```
CREATE TABLE Customers (
   CID INT PRIMARY KEY,
   Name VARCHAR(100)
);

CREATE TABLE MenuItems (
   ItemID INT PRIMARY KEY,
   Name VARCHAR(100),
   Price DECIMAL(10,2)
);

CREATE TABLE Orders (
   OrderID INT PRIMARY KEY,
   CID INT,
   ItemID INT,
   OrderDate DATE,
   Quantity INT,
   FOREIGN KEY (CID) REFERENCES Customers(CID),
   FOREIGN KEY (ItemID) REFERENCES MenuItems(ItemID)
);


INSERT INTO Customers VALUES (1, 'Varun'), (2, 'Priya');

INSERT INTO MenuItems VALUES (101, 'Pizza', 350), (102, 'Burger', 150);

INSERT INTO Orders VALUES (1001, 1, 101, '2025-04-01', 2), (1002, 2, 102, '2025-
04-01', 3);
```

```
SELECT * FROM MenuItems WHERE Price > 300;

SELECT * FROM Orders WHERE OrderDate = '2025-04-01';

SELECT MI.Name, SUM(O.Quantity * MI.Price) AS Revenue
FROM Orders O
JOIN MenuItems MI ON O.ItemID = MI.ItemID
GROUP BY MI.Name;

SELECT CID FROM Orders WHERE ItemID = 101
INTERSECT
SELECT CID FROM Orders WHERE ItemID = 102;

SELECT C.Name AS Customer, MI.Name AS Item
FROM Orders O
JOIN Customers C ON O.CID = C.CID
JOIN MenuItems MI ON O.ItemID = MI.ItemID;
```

19: Employee Leave Management

```
CREATE TABLE Employees (
   EID INT PRIMARY KEY,
   Name VARCHAR(100),
   Dept VARCHAR(50)
);

CREATE TABLE LeaveApplications (
   EID INT,
   LeaveDate DATE,
   Reason VARCHAR(100),
   FOREIGN KEY (EID) REFERENCES Employees(EID)
);

INSERT INTO Employees VALUES (1, 'Rita', 'HR'), (2, 'Aman', 'IT');

INSERT INTO LeaveApplications VALUES
(1, '2025-01-15', 'Medical'),
(1, '2025-02-10', 'Vacation'),
(2, '2025-03-05', 'Travel');
```

```
SELECT * FROM Employees WHERE Dept = 'HR';

SELECT * FROM LeaveApplications WHERE MONTH(LeaveDate) = 3;

SELECT EID, COUNT(*) AS LeaveCount FROM LeaveApplications GROUP BY EID;

SELECT EID FROM LeaveApplications WHERE MONTH(LeaveDate) = 1
INTERSECT
SELECT EID FROM LeaveApplications WHERE MONTH(LeaveDate) = 2;

SELECT E.Name, L.LeaveDate, L.Reason
FROM LeaveApplications L
JOIN Employees E ON L.EID = E.EID;
```

20: NGO Donation Tracking

```
CREATE TABLE Donors (
   DonorID INT PRIMARY KEY,
   Name VARCHAR(100),
   Email VARCHAR(100)
);

CREATE TABLE Beneficiaries (
   BeneficiaryID INT PRIMARY KEY,
   Name VARCHAR(100),
   Cause VARCHAR(100)
);

CREATE TABLE Donations (
   DonationID INT PRIMARY KEY,
   DonorID INT,
   BeneficiaryID INT,
   Amount DECIMAL(10,2),
   Date DATE,
   FOREIGN KEY (DonorID) REFERENCES Donors(DonorID),
   FOREIGN KEY (BeneficiaryID) REFERENCES Beneficiaries(BeneficiaryID)
);
```

```
INSERT INTO Donors VALUES (1, 'Ajay', 'ajay@mail.com'), (2, 'Sneha',
'sneha@mail.com');
```

```
INSERT INTO Beneficiaries VALUES (101, 'ChildCare', 'Education'), (102, 'ElderHelp', 'Healthcare');

INSERT INTO Donations VALUES
(1, 1, 101, 6000, '2025-04-02'),
(2, 2, 102, 3000, '2025-04-03'),
(3, 1, 102, 4000, '2025-04-05');


SELECT * FROM Donations WHERE Amount > 5000;

SELECT * FROM Donations WHERE Date > '2025-04-01';

SELECT B.Name, SUM(D.Amount) AS TotalReceived
FROM Donations D
JOIN Beneficiaries B ON D.BeneficiaryID = B.BeneficiaryID
GROUP BY B.Name;

SELECT DonorID FROM Donations WHERE BeneficiaryID = 101
INTERSECT
SELECT DonorID FROM Donations WHERE BeneficiaryID = 102;

SELECT Donors.Name, Donations.Amount, Beneficiaries.Name AS Beneficiary
FROM Donations
JOIN Donors ON Donations.DonorID = Donors.DonorID
JOIN Beneficiaries ON Donations.BeneficiaryID = Beneficiaries.BeneficiaryID;
```