1.

CREATE DATABASE EmployeeProjectDB;

USE EmployeeProjectDB;

CREATE TABLE employee (

eid INT PRIMARY KEY,

ename VARCHAR(100),

salary INT

);

CREATE TABLE manager (

eid INT PRIMARY KEY,

ename VARCHAR(100),

FOREIGN KEY (eid) REFERENCES employee(eid)

);

CREATE TABLE project (

projectid INT PRIMARY KEY,

project\_name VARCHAR(255),

manager INT,

FOREIGN KEY (manager) REFERENCES manager(eid)

);

CREATE TABLE assignment (

projectid INT,

eid INT,

PRIMARY KEY (projectid, eid),

FOREIGN KEY (projectid) REFERENCES project(projectid),

FOREIGN KEY (eid) REFERENCES employee(eid)

);

ALTER TABLE employee ADD address VARCHAR(255);

SELECT e.ename, p.project\_name

FROM employee e

JOIN assignment a ON e.eid = a.eid

JOIN project p ON a.projectid = p.projectid;

SELECT p.projectid, p.project\_name, m.ename AS manager\_name

FROM project p

JOIN manager m ON p.manager = m.eid;

CREATE VIEW Bank\_Management\_Employees AS

SELECT e.eid, e.ename, e.salary

FROM employee e

JOIN assignment a ON e.eid = a.eid

JOIN project p ON a.projectid = p.projectid

WHERE p.project\_name = 'Bank Management';

SELECT ename FROM employee WHERE salary > 40000;

UPDATE employee SET salary = salary + 2000;

2.

CREATE DATABASE EmployeeProjectDB;

USE EmployeeProjectDB;

CREATE TABLE employee (

eid INT AUTO\_INCREMENT PRIMARY KEY,

ename VARCHAR(100),

salary INT

);

CREATE TABLE manager (

eid INT PRIMARY KEY,

ename VARCHAR(100),

FOREIGN KEY (eid) REFERENCES employee(eid)

);

CREATE TABLE project (

projectid INT PRIMARY KEY,

project\_name VARCHAR(255),

manager INT,

FOREIGN KEY (manager) REFERENCES manager(eid)

);

CREATE TABLE assignment (

projectid INT,

eid INT,

PRIMARY KEY (projectid, eid),

FOREIGN KEY (projectid) REFERENCES project(projectid),

FOREIGN KEY (eid) REFERENCES employee(eid)

);

ALTER TABLE employee MODIFY eid INT AUTO\_INCREMENT;

SELECT e.eid, e.ename

FROM employee e

JOIN assignment a1 ON e.eid = a1.eid

JOIN project p1 ON a1.projectid = p1.projectid

JOIN assignment a2 ON e.eid = a2.eid

JOIN project p2 ON a2.projectid = p2.projectid

WHERE p1.project\_name = 'Bank Management'

AND p2.project\_name = 'Content Management';

SELECT AVG(salary) AS average\_salary FROM employee;

SELECT e.eid, e.ename

FROM employee e

WHERE e.eid NOT IN (

SELECT a.eid

FROM assignment a

JOIN project p ON a.projectid = p.projectid

WHERE p.project\_name = 'Bank Management'

);

DELETE FROM employee WHERE eid = 5;

SELECT ename, salary

FROM employee

WHERE salary = (SELECT MAX(salary) FROM employee);

3.

CREATE DATABASE SupplierPartsDB;

USE SupplierPartsDB;

CREATE TABLE supplier (

supplierid INT PRIMARY KEY,

sname VARCHAR(100),

saddress VARCHAR(255)

);

CREATE TABLE parts (

part\_id INT PRIMARY KEY,

part\_name VARCHAR(100),

color VARCHAR(50)

);

CREATE TABLE catalog (

supplierid INT,

part\_id INT,

cost DECIMAL(10,2),

PRIMARY KEY (supplierid, part\_id),

FOREIGN KEY (supplierid) REFERENCES supplier(supplierid),

FOREIGN KEY (part\_id) REFERENCES parts(part\_id)

);

SELECT DISTINCT s.sname

FROM supplier s

JOIN catalog c ON s.supplierid = c.supplierid

JOIN parts p ON c.part\_id = p.part\_id

WHERE p.color = 'green';

SELECT s.sname

FROM supplier s

JOIN catalog c1 ON s.supplierid = c1.supplierid

JOIN parts p1 ON c1.part\_id = p1.part\_id

JOIN catalog c2 ON s.supplierid = c2.supplierid

JOIN parts p2 ON c2.part\_id = p2.part\_id

WHERE p1.color = 'blue' AND p2.color = 'green';

SELECT s.supplierid, s.sname

FROM supplier s

WHERE NOT EXISTS (

SELECT p.part\_id

FROM parts p

WHERE NOT EXISTS (

SELECT c.supplierid

FROM catalog c

WHERE c.supplierid = s.supplierid AND c.part\_id = p.part\_id

)

);

SELECT SUM(c.cost) AS total\_red\_parts\_cost

FROM catalog c

JOIN parts p ON c.part\_id = p.part\_id

WHERE p.color = 'red';

SELECT s.sname, c.cost

FROM supplier s

JOIN catalog c ON s.supplierid = c.supplierid

JOIN parts p ON c.part\_id = p.part\_id

WHERE p.color = 'green'

ORDER BY c.cost ASC

LIMIT 1;

UPDATE parts

SET color = 'new\_color'

WHERE part\_id = 4;

4.

CREATE DATABASE EmployeeCompanyDB;

USE EmployeeCompanyDB;

CREATE TABLE emp (

eid INT PRIMARY KEY,

ename VARCHAR(100),

street VARCHAR(255),

city VARCHAR(100)

);

CREATE TABLE company (

company\_name VARCHAR(100) PRIMARY KEY,

city VARCHAR(100)

);

CREATE TABLE works (

eid INT,

company\_name VARCHAR(100),

salary INT,

PRIMARY KEY (eid, company\_name),

FOREIGN KEY (eid) REFERENCES emp(eid),

FOREIGN KEY (company\_name) REFERENCES company(company\_name)

);

CREATE TABLE manages (

eid INT PRIMARY KEY,

manager\_id INT,

FOREIGN KEY (manager\_id) REFERENCES emp(eid)

);

UPDATE works

SET company\_name = 'TCS'

WHERE eid = (SELECT eid FROM emp WHERE ename = 'Prashant')

AND company\_name = 'Infosys';

SELECT e.ename, e.city

FROM emp e

JOIN works w ON e.eid = w.eid

WHERE w.company\_name = 'Infosys';

SELECT e.ename, e.street

FROM emp e

JOIN works w ON e.eid = w.eid

JOIN company c ON w.company\_name = c.company\_name

WHERE c.company\_name = 'TCS'

AND w.salary > 20000;

SELECT DISTINCT e.ename

FROM emp e

WHERE e.eid NOT IN (

SELECT w.eid FROM works w WHERE w.company\_name = 'Infosys'

);

SELECT company\_name, SUM(salary) AS total\_salary

FROM works

GROUP BY company\_name;

SELECT e.ename

FROM emp e

JOIN works w ON e.eid = w.eid

WHERE w.company\_name = 'Accenture';

5.

CREATE DATABASE ProjectManagementDB;

USE ProjectManagementDB;

CREATE TABLE employee (

eid INT AUTO\_INCREMENT PRIMARY KEY,

ename VARCHAR(100),

salary INT

);

CREATE TABLE project (

projectid INT PRIMARY KEY,

project\_name VARCHAR(255),

manager INT,

FOREIGN KEY (manager) REFERENCES employee(eid)

);

CREATE TABLE assignment (

projectid INT,

eid INT,

PRIMARY KEY (projectid, eid),

FOREIGN KEY (projectid) REFERENCES project(projectid),

FOREIGN KEY (eid) REFERENCES employee(eid)

);

CREATE TABLE manager (

eid INT PRIMARY KEY,

ename VARCHAR(100),

FOREIGN KEY (eid) REFERENCES employee(eid)

);

ALTER TABLE employee MODIFY eid INT AUTO\_INCREMENT;

SELECT e.ename

FROM employee e

JOIN assignment a1 ON e.eid = a1.eid

JOIN project p1 ON a1.projectid = p1.projectid

JOIN assignment a2 ON e.eid = a2.eid

JOIN project p2 ON a2.projectid = p2.projectid

WHERE p1.project\_name = 'Bank Management'

AND p2.project\_name = 'Content Management';

SELECT AVG(salary) AS average\_salary

FROM employee;

SELECT e.ename

FROM employee e

WHERE e.eid NOT IN (

SELECT a.eid

FROM assignment a

JOIN project p ON a.projectid = p.projectid

WHERE p.project\_name = 'Bank Management'

);

DELETE FROM employee WHERE eid = 5;

SELECT ename, salary

FROM employee

WHERE salary = (SELECT MAX(salary) FROM employee);

6.

CREATE DATABASE SupplierPartsDB;

USE SupplierPartsDB;

CREATE TABLE supplier (

supplierid INT PRIMARY KEY,

sname VARCHAR(100),

saddress VARCHAR(255)

);

CREATE TABLE parts (

part\_id INT PRIMARY KEY,

part\_name VARCHAR(100),

color VARCHAR(50)

);

CREATE TABLE catalog (

supplierid INT,

part\_id INT,

cost INT,

PRIMARY KEY (supplierid, part\_id),

FOREIGN KEY (supplierid) REFERENCES supplier(supplierid),

FOREIGN KEY (part\_id) REFERENCES parts(part\_id)

);

SELECT DISTINCT s.sname

FROM supplier s

JOIN catalog c ON s.supplierid = c.supplierid

JOIN parts p ON c.part\_id = p.part\_id

WHERE p.color = 'green';

SELECT s.sname

FROM supplier s

JOIN catalog c1 ON s.supplierid = c1.supplierid

JOIN parts p1 ON c1.part\_id = p1.part\_id

JOIN catalog c2 ON s.supplierid = c2.supplierid

JOIN parts p2 ON c2.part\_id = p2.part\_id

WHERE p1.color = 'blue' AND p2.color = 'green';

SELECT s.sname

FROM supplier s

WHERE NOT EXISTS (

SELECT p.part\_id

FROM parts p

WHERE NOT EXISTS (

SELECT c.part\_id

FROM catalog c

WHERE c.supplierid = s.supplierid AND c.part\_id = p.part\_id

)

);

SELECT SUM(c.cost) AS total\_red\_parts\_cost

FROM catalog c

JOIN parts p ON c.part\_id = p.part\_id

WHERE p.color = 'red';

SELECT s.sname, c.cost

FROM supplier s

JOIN catalog c ON s.supplierid = c.supplierid

JOIN parts p ON c.part\_id = p.part\_id

WHERE p.color = 'green'

ORDER BY c.cost ASC

LIMIT 1;

UPDATE parts

SET color = 'new\_color'

WHERE part\_id = 4;

7.

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15),

City VARCHAR(50)

);

CREATE TABLE Cars (

CarID INT PRIMARY KEY,

Model VARCHAR(100),

Brand VARCHAR(50),

Year INT,

RentalPricePerDay DECIMAL(10,2),

AvailabilityStatus VARCHAR(20) CHECK (AvailabilityStatus IN ('Available', 'Rented'))

);

CREATE TABLE Rentals (

RentalID INT PRIMARY KEY,

CustomerID INT,

CarID INT,

StartDate DATE,

EndDate DATE,

TotalAmount DECIMAL(10,2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (CarID) REFERENCES Cars(CarID)

);

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY,

RentalID INT,

PaymentDate DATE,

AmountPaid DECIMAL(10,2),

PaymentMethod VARCHAR(50),

FOREIGN KEY (RentalID) REFERENCES Rentals(RentalID)

);

UPDATE Cars

SET AvailabilityStatus = 'Rented'

WHERE CarID = (SELECT CarID FROM Rentals WHERE CustomerID = 1 AND CarID = 10);

SELECT c.Name, ca.Model, r.StartDate

FROM Rentals r

JOIN Customers c ON r.CustomerID = c.CustomerID

JOIN Cars ca ON r.CarID = ca.CarID

WHERE ca.RentalPricePerDay > 1000;

SELECT ca.Brand, SUM(r.TotalAmount) AS TotalRentalAmount

FROM Rentals r

JOIN Cars ca ON r.CarID = ca.CarID

GROUP BY ca.Brand;

SELECT c.Name, SUM(r.TotalAmount) AS TotalSpent

FROM Rentals r

JOIN Customers c ON r.CustomerID = c.CustomerID

GROUP BY c.Name

ORDER BY TotalSpent DESC

LIMIT 3;

8.

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15),

Address VARCHAR(255)

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

Name VARCHAR(100),

Category VARCHAR(50),

Price DECIMAL(10,2),

StockQuantity INT

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10,2),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT,

Subtotal DECIMAL(10,2),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY,

OrderID INT,

PaymentDate DATE,

AmountPaid DECIMAL(10,2),

PaymentMethod VARCHAR(50),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

UPDATE Products

SET StockQuantity = StockQuantity - (

SELECT Quantity FROM OrderDetails WHERE OrderID = 101 AND ProductID = 5

)

WHERE ProductID = 5;

SELECT c.Name, o.OrderDate, o.TotalAmount

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

WHERE o.TotalAmount > 5000;

SELECT p.Category, SUM(od.Subtotal) AS TotalSales

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY p.Category;

SELECT c.Name, SUM(o.TotalAmount) AS TotalSpent

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

GROUP BY c.Name

ORDER BY TotalSpent DESC

LIMIT 5;

9.

CREATE TABLE Members (

MemberID INT PRIMARY KEY,

Name VARCHAR(100),

Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15),

MembershipDate DATE

);

CREATE TABLE Books (

BookID INT PRIMARY KEY,

Title VARCHAR(200),

Author VARCHAR(100),

Genre VARCHAR(50),

CopiesAvailable INT

);

CREATE TABLE BorrowedBooks (

BorrowID INT PRIMARY KEY,

MemberID INT,

BookID INT,

BorrowDate DATE,

ReturnDate DATE,

FOREIGN KEY (MemberID) REFERENCES Members(MemberID),

FOREIGN KEY (BookID) REFERENCES Books(BookID)

);

CREATE TABLE Fines (

FineID INT PRIMARY KEY,

MemberID INT,

Amount DECIMAL(10,2),

Status VARCHAR(20) CHECK (Status IN ('Unpaid', 'Paid')),

FineDate DATE,

FOREIGN KEY (MemberID) REFERENCES Members(MemberID)

);

UPDATE Books

SET CopiesAvailable = CopiesAvailable - 1

WHERE BookID = 5;

UPDATE Books

SET CopiesAvailable = CopiesAvailable + 1

WHERE BookID = 5;

SELECT m.Name, b.Title, bb.BorrowDate

FROM BorrowedBooks bb

JOIN Members m ON bb.MemberID = m.MemberID

JOIN Books b ON bb.BookID = b.BookID

WHERE bb.BorrowDate >= DATE\_SUB(CURDATE(), INTERVAL 1 MONTH);

SELECT b.Genre, COUNT(bb.BorrowID) AS BooksBorrowed

FROM BorrowedBooks bb

JOIN Books b ON bb.BookID = b.BookID

GROUP BY b.Genre;

SELECT m.Name, COUNT(bb.BorrowID) AS BooksBorrowed

FROM BorrowedBooks bb

JOIN Members m ON bb.MemberID = m.MemberID

GROUP BY m.Name

ORDER BY BooksBorrowed DESC

LIMIT 5;

10.

CREATE DATABASE HospitalManagement;

USE HospitalManagement;

CREATE TABLE Patients (

PatientID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Age INT CHECK (Age > 0),

Gender VARCHAR(10) CHECK (Gender IN ('Male', 'Female', 'Other')),

Contact VARCHAR(15) UNIQUE NOT NULL

);

CREATE TABLE Doctors (

DoctorID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Specialization VARCHAR(100) NOT NULL,

Contact VARCHAR(15) UNIQUE NOT NULL

);

CREATE TABLE Appointments (

AppointmentID INT PRIMARY KEY,

PatientID INT NOT NULL,

DoctorID INT NOT NULL,

AppointmentDate DATE NOT NULL,

Status VARCHAR(20) CHECK (Status IN ('Scheduled', 'Completed', 'Cancelled')) DEFAULT 'Scheduled',

FOREIGN KEY (PatientID) REFERENCES Patients(PatientID) ON DELETE CASCADE,

FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID) ON DELETE CASCADE

);

CREATE TABLE Bills (

BillID INT PRIMARY KEY,

PatientID INT NOT NULL,

Amount DECIMAL(10,2) CHECK (Amount >= 0),

PaymentStatus VARCHAR(20) CHECK (PaymentStatus IN ('Paid', 'Unpaid', 'Pending')) DEFAULT 'Pending',

FOREIGN KEY (PatientID) REFERENCES Patients(PatientID) ON DELETE CASCADE

);

CREATE TABLE MedicalRecords (

RecordID INT PRIMARY KEY,

PatientID INT NOT NULL,

Diagnosis TEXT NOT NULL,

Prescription TEXT NOT NULL,

RecordDate DATE NOT NULL,

FOREIGN KEY (PatientID) REFERENCES Patients(PatientID) ON DELETE CASCADE

);

UPDATE Appointments

SET Status = 'Completed'

WHERE AppointmentID = 10;

SELECT p.Name AS PatientName, d.Name AS DoctorName, a.AppointmentDate

FROM Appointments a

JOIN Patients p ON a.PatientID = p.PatientID

JOIN Doctors d ON a.DoctorID = d.DoctorID

WHERE d.Specialization = 'Cardiology';

SELECT d.Name AS DoctorName, SUM(b.Amount) AS TotalRevenue

FROM Bills b

JOIN Appointments a ON b.PatientID = a.PatientID

JOIN Doctors d ON a.DoctorID = d.DoctorID

GROUP BY d.Name;

SELECT d.Name AS DoctorName, COUNT(a.AppointmentID) AS AppointmentCount

FROM Appointments a

JOIN Doctors d ON a.DoctorID = d.DoctorID

GROUP BY d.Name

ORDER BY AppointmentCount DESC

LIMIT 3;

11.

CREATE DATABASE UniversityManagement;

USE UniversityManagement;

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Age INT CHECK (Age > 0),

Gender VARCHAR(10) CHECK (Gender IN ('Male', 'Female', 'Other')),

Department VARCHAR(50) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL

);

CREATE TABLE Courses (

CourseID INT PRIMARY KEY,

CourseName VARCHAR(100) NOT NULL,

Credits INT CHECK (Credits > 0),

Department VARCHAR(50) NOT NULL

);

CREATE TABLE Enrollments (

EnrollmentID INT PRIMARY KEY,

StudentID INT NOT NULL,

CourseID INT NOT NULL,

EnrollmentDate DATE NOT NULL,

Grade CHAR(2),

Semester VARCHAR(20) NOT NULL,

FOREIGN KEY (StudentID) REFERENCES Students(StudentID) ON DELETE CASCADE,

FOREIGN KEY (CourseID) REFERENCES Courses(CourseID) ON DELETE CASCADE

);

CREATE TABLE Professors (

ProfessorID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Department VARCHAR(50) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL

);

SELECT Department,

COUNT(StudentID) AS StudentCount,

(COUNT(StudentID) \* 100.0 / (SELECT COUNT(\*) FROM Students)) AS Percentage

FROM Students

GROUP BY Department;

SELECT StudentID, CourseID, Semester, COUNT(\*) AS DuplicateCount

FROM Enrollments

GROUP BY StudentID, CourseID, Semester

HAVING COUNT(\*) > 1;

SELECT Semester,

(COUNT(EnrollmentID) \* 1.0 / COUNT(DISTINCT CourseID)) AS AvgEnrollmentsPerCourse

FROM Enrollments

GROUP BY Semester

ORDER BY AvgEnrollmentsPerCourse DESC

LIMIT 1;

SELECT s.StudentID, s.Name, COUNT(e.EnrollmentID) AS EnrollmentCount

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

GROUP BY s.StudentID, s.Name

HAVING COUNT(e.EnrollmentID) > 3;

SELECT c.CourseID, c.CourseName, COUNT(e.StudentID) AS StudentCount

FROM Courses c

LEFT JOIN Enrollments e ON c.CourseID = e.CourseID

GROUP BY c.CourseID, c.CourseName

ORDER BY StudentCount DESC;

12.

CREATE DATABASE UniversityDB;

USE UniversityDB;

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Age INT CHECK (Age > 0),

Gender VARCHAR(10) CHECK (Gender IN ('Male', 'Female', 'Other')),

Department VARCHAR(50) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL

);

CREATE TABLE Courses (

CourseID INT PRIMARY KEY,

CourseName VARCHAR(100) NOT NULL,

Credits INT CHECK (Credits > 0),

Department VARCHAR(50) NOT NULL

);

CREATE TABLE Enrollments (

EnrollmentID INT PRIMARY KEY AUTO\_INCREMENT,

StudentID INT NOT NULL,

CourseID INT NOT NULL,

EnrollmentDate DATE NOT NULL,

Grade DECIMAL(3,2),

FOREIGN KEY (StudentID) REFERENCES Students(StudentID) ON DELETE CASCADE,

FOREIGN KEY (CourseID) REFERENCES Courses(CourseID) ON DELETE CASCADE

);

CREATE TABLE Professors (

ProfessorID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Department VARCHAR(50) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL

);

INSERT INTO Students (StudentID, Name, Age, Gender, Department, Email) VALUES

(1, 'Pooja', 20, 'Female', 'Computer Science', 'pooja@example.com'),

(2, 'Raj', 22, 'Male', 'Mechanical', 'raj@example.com'),

(3, 'Aisha', 21, 'Female', 'Electronics', 'aisha@example.com'),

(4, 'Vikas', 23, 'Male', 'Computer Science', 'vikas@example.com'),

(5, 'Neha', 22, 'Female', 'Civil', 'neha@example.com');

INSERT INTO Courses (CourseID, CourseName, Credits, Department) VALUES

(101, 'Database Systems', 4, 'Computer Science'),

(102, 'Machine Learning', 3, 'Computer Science'),

(103, 'Thermodynamics', 4, 'Mechanical'),

(104, 'Structural Analysis', 3, 'Civil'),

(105, 'Embedded Systems', 4, 'Electronics');

INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate, Grade) VALUES

(1, 101, '2024-02-10', 3.5),

(1, 102, '2024-02-15', 2.8),

(2, 103, '2024-02-12', 3.9),

(3, 105, '2024-02-18', 1.5),

(4, 101, '2024-02-11', 3.0),

(5, 104, '2024-02-20', 2.7),

(3, 102, '2024-02-21', 1.2);

INSERT INTO Professors (ProfessorID, Name, Department, Email) VALUES

(201, 'Dr. Sharma', 'Computer Science', 'sharma@example.com'),

(202, 'Dr. Mehta', 'Mechanical', 'mehta@example.com'),

(203, 'Dr. Reddy', 'Civil', 'reddy@example.com'),

(204, 'Dr. Gupta', 'Electronics', 'gupta@example.com');

SELECT c.CourseID, c.CourseName

FROM Courses c

JOIN Enrollments e ON c.CourseID = e.CourseID

JOIN Students s ON e.StudentID = s.StudentID

WHERE s.Name = 'Pooja';

SELECT s.StudentID, s.Name, COUNT(e.CourseID) AS FailedCourses

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

WHERE e.Grade < 2.0

GROUP BY s.StudentID, s.Name

HAVING COUNT(e.CourseID) > 2;

SELECT Department, COUNT(StudentID) AS StudentCount

FROM Students

GROUP BY Department;

SELECT c.CourseID, c.CourseName

FROM Courses c

LEFT JOIN Enrollments e ON c.CourseID = e.CourseID

WHERE e.CourseID IS NULL;

SELECT c.CourseID, c.CourseName, COUNT(e.StudentID) AS EnrollmentCount

FROM Courses c

JOIN Enrollments e ON c.CourseID = e.CourseID

GROUP BY c.CourseID, c.CourseName

ORDER BY EnrollmentCount DESC

LIMIT 1;

13.

CREATE DATABASE UniversityDB;

USE UniversityDB;

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Age INT CHECK (Age > 0),

Gender VARCHAR(10) CHECK (Gender IN ('Male', 'Female', 'Other')),

Department VARCHAR(50) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL

);

CREATE TABLE Courses (

CourseID INT PRIMARY KEY,

CourseName VARCHAR(100) NOT NULL,

Credits INT CHECK (Credits > 0),

Department VARCHAR(50) NOT NULL

);

CREATE TABLE Enrollments (

EnrollmentID INT PRIMARY KEY AUTO\_INCREMENT,

StudentID INT NOT NULL,

CourseID INT NOT NULL,

EnrollmentDate DATE NOT NULL,

Grade DECIMAL(3,2),

FOREIGN KEY (StudentID) REFERENCES Students(StudentID) ON DELETE CASCADE,

FOREIGN KEY (CourseID) REFERENCES Courses(CourseID) ON DELETE CASCADE

);

CREATE TABLE Professors (

ProfessorID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Department VARCHAR(50) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL

);

INSERT INTO Students (StudentID, Name, Age, Gender, Department, Email) VALUES

(1, 'Pooja', 20, 'Female', 'Computer Science', 'pooja@example.com'),

(2, 'Raj', 22, 'Male', 'Mechanical', 'raj@example.com'),

(3, 'Aisha', 21, 'Female', 'Electronics', 'aisha@example.com'),

(4, 'Vikas', 23, 'Male', 'Computer Science', 'vikas@example.com'),

(5, 'Neha', 22, 'Female', 'Civil', 'neha@example.com'),

(6, 'Sahil', 24, 'Male', 'Mechanical', 'sahil@example.com');

INSERT INTO Courses (CourseID, CourseName, Credits, Department) VALUES

(101, 'Database Systems', 4, 'Computer Science'),

(102, 'Machine Learning', 3, 'Computer Science'),

(103, 'Thermodynamics', 4, 'Mechanical'),

(104, 'Structural Analysis', 3, 'Civil'),

(105, 'Embedded Systems', 4, 'Electronics');

INSERT INTO Enrollments (StudentID, CourseID, EnrollmentDate, Grade) VALUES

(1, 101, '2024-02-10', 3.5),

(1, 102, '2024-02-15', 2.8),

(2, 103, '2024-02-12', 3.9),

(3, 105, '2024-02-18', 1.5),

(4, 101, '2024-02-11', 3.0),

(5, 104, '2024-02-20', 2.7),

(3, 102, '2024-02-21', 1.2),

(1, 103, '2024-02-22', 3.7),

(2, 101, '2024-02-23', 3.9),

(4, 102, '2024-02-24', 3.1),

(5, 101, '2024-02-25', 2.9);

INSERT INTO Professors (ProfessorID, Name, Department, Email) VALUES

(201, 'Dr. Sharma', 'Computer Science', 'sharma@example.com'),

(202, 'Dr. Mehta', 'Mechanical', 'mehta@example.com'),

(203, 'Dr. Reddy', 'Civil', 'reddy@example.com'),

(204, 'Dr. Gupta', 'Electronics', 'gupta@example.com');

SELECT s.StudentID, s.Name

FROM Students s

LEFT JOIN Enrollments e ON s.StudentID = e.StudentID

WHERE e.StudentID IS NULL;

SELECT s.StudentID, s.Name, COUNT(e.CourseID) AS EnrolledCourses

FROM Students s

JOIN Enrollments e ON s.StudentID = e.StudentID

GROUP BY s.StudentID, s.Name

HAVING COUNT(e.CourseID) > 3;

SELECT c.CourseID, c.CourseName, AVG(e.Grade) AS AverageGrade

FROM Courses c

JOIN Enrollments e ON c.CourseID = e.CourseID

GROUP BY c.CourseID, c.CourseName;

SELECT c.CourseID, c.CourseName, MAX(e.Grade) AS HighestGrade

FROM Courses c

JOIN Enrollments e ON c.CourseID = e.CourseID

GROUP BY c.CourseID, c.CourseName;

SELECT Department, COUNT(StudentID) AS StudentCount

FROM Students

GROUP BY Department

ORDER BY StudentCount DESC

LIMIT 1;

14.

CREATE DATABASE BankDB;

USE BankDB;

CREATE TABLE Customer (

customer\_id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

address VARCHAR(255) NOT NULL,

phone VARCHAR(15) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL

);

CREATE TABLE Branch (

branch\_id INT PRIMARY KEY,

branch\_name VARCHAR(100) NOT NULL,

location VARCHAR(255) NOT NULL,

manager\_id INT

);

CREATE TABLE Account (

account\_id INT PRIMARY KEY,

customer\_id INT NOT NULL,

account\_type VARCHAR(50) CHECK (account\_type IN ('Savings', 'Current', 'Fixed Deposit')),

balance DECIMAL(15,2) CHECK (balance >= 0),

branch\_id INT NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id) ON DELETE CASCADE,

FOREIGN KEY (branch\_id) REFERENCES Branch(branch\_id) ON DELETE CASCADE

);

CREATE TABLE Transaction (

transaction\_id INT PRIMARY KEY AUTO\_INCREMENT,

account\_id INT NOT NULL,

transaction\_type VARCHAR(50) CHECK (transaction\_type IN ('Deposit', 'Withdrawal', 'Transfer')),

amount DECIMAL(15,2) CHECK (amount > 0),

transaction\_date DATE NOT NULL,

FOREIGN KEY (account\_id) REFERENCES Account(account\_id) ON DELETE CASCADE

);

CREATE TABLE Loan (

loan\_id INT PRIMARY KEY AUTO\_INCREMENT,

customer\_id INT NOT NULL,

amount DECIMAL(15,2) CHECK (amount > 0),

loan\_type VARCHAR(50) CHECK (loan\_type IN ('Home Loan', 'Car Loan', 'Personal Loan', 'Education Loan')),

status VARCHAR(50) CHECK (status IN ('Approved', 'Pending', 'Rejected')),

FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id) ON DELETE CASCADE

);

CREATE TABLE Employee (

employee\_id INT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

position VARCHAR(50) NOT NULL,

branch\_id INT NOT NULL,

salary DECIMAL(10,2) CHECK (salary > 0),

FOREIGN KEY (branch\_id) REFERENCES Branch(branch\_id) ON DELETE CASCADE

);

INSERT INTO Customer (customer\_id, name, address, phone, email) VALUES

(1, 'Amit Sharma', 'Delhi', '9876543210', 'amit@example.com'),

(2, 'Neha Verma', 'Mumbai', '9876543211', 'neha@example.com'),

(3, 'Rohan Singh', 'Bangalore', '9876543212', 'rohan@example.com'),

(4, 'Priya Mehta', 'Kolkata', '9876543213', 'priya@example.com');

INSERT INTO Branch (branch\_id, branch\_name, location, manager\_id) VALUES

(101, 'Delhi Branch', 'Delhi', 201),

(102, 'Mumbai Branch', 'Mumbai', 202),

(103, 'Bangalore Branch', 'Bangalore', 203);

INSERT INTO Account (account\_id, customer\_id, account\_type, balance, branch\_id) VALUES

(1001, 1, 'Savings', 150000.00, 101),

(1002, 2, 'Current', 20000.00, 102),

(1003, 3, 'Fixed Deposit', 500000.00, 103),

(1004, 4, 'Savings', 80000.00, 101);

INSERT INTO Transaction (account\_id, transaction\_type, amount, transaction\_date) VALUES

(1001, 'Deposit', 5000.00, '2024-03-01'),

(1002, 'Withdrawal', 2000.00, '2024-03-02'),

(1003, 'Deposit', 10000.00, '2024-03-03'),

(1001, 'Transfer', 1500.00, '2024-03-04');

INSERT INTO Loan (customer\_id, amount, loan\_type, status) VALUES

(1, 250000.00, 'Home Loan', 'Approved'),

(2, 50000.00, 'Car Loan', 'Pending'),

(3, 120000.00, 'Personal Loan', 'Approved'),

(4, 90000.00, 'Education Loan', 'Rejected');

INSERT INTO Employee (employee\_id, name, position, branch\_id, salary) VALUES

(201, 'Suresh Kumar', 'Manager', 101, 75000.00),

(202, 'Anita Desai', 'Manager', 102, 72000.00),

(203, 'Vikram Rao', 'Manager', 103, 78000.00),

(204, 'Kavita Shah', 'Clerk', 101, 35000.00);

SELECT c.customer\_id, c.name, a.account\_id, a.account\_type, a.balance, a.branch\_id

FROM Customer c

JOIN Account a ON c.customer\_id = a.customer\_id;

SELECT b.branch\_id, b.branch\_name, SUM(a.balance) AS total\_balance

FROM Branch b

JOIN Account a ON b.branch\_id = a.branch\_id

GROUP BY b.branch\_id, b.branch\_name;

SELECT customer\_id, name, amount

FROM Customer c

JOIN Loan l ON c.customer\_id = l.customer\_id

WHERE l.amount > 100000;

SELECT \* FROM Transaction WHERE account\_id = 1001;

SELECT c.customer\_id, c.name

FROM Customer c

JOIN Account a ON c.customer\_id = a.customer\_id

JOIN Loan l ON c.customer\_id = l.customer\_id;

CREATE VIEW HighValueCustomers AS

SELECT c.customer\_id, c.name, a.balance

FROM Customer c

JOIN Account a ON c.customer\_id = a.customer\_id

WHERE a.balance > 100000;

15.

```sql

CREATE DATABASE BankDB;

USE BankDB;

CREATE TABLE Customer (

customer\_id INT PRIMARY KEY,

name VARCHAR(100),

address VARCHAR(255),

phone VARCHAR(15),

email VARCHAR(100) UNIQUE

);

CREATE TABLE Branch (

branch\_id INT PRIMARY KEY,

branch\_name VARCHAR(100),

location VARCHAR(255),

manager\_id INT

);

CREATE TABLE Account (

account\_id INT PRIMARY KEY,

customer\_id INT,

account\_type VARCHAR(50),

balance DECIMAL(10,2),

branch\_id INT,

FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id),

FOREIGN KEY (branch\_id) REFERENCES Branch(branch\_id)

);

CREATE TABLE Transaction (

transaction\_id INT PRIMARY KEY,

account\_id INT,

transaction\_type VARCHAR(50),

amount DECIMAL(10,2),

transaction\_date DATE,

FOREIGN KEY (account\_id) REFERENCES Account(account\_id)

);

CREATE TABLE Loan (

loan\_id INT PRIMARY KEY,

customer\_id INT,

amount DECIMAL(10,2),

loan\_type VARCHAR(50),

status VARCHAR(20) CHECK (status IN ('Approved', 'Pending', 'Rejected')),

FOREIGN KEY (customer\_id) REFERENCES Customer(customer\_id)

);

CREATE TABLE Employee (

employee\_id INT PRIMARY KEY,

name VARCHAR(100),

position VARCHAR(50),

branch\_id INT,

salary DECIMAL(10,2),

FOREIGN KEY (branch\_id) REFERENCES Branch(branch\_id)

);

INSERT INTO Customer VALUES

(1, 'Amit Sharma', 'Mumbai', '9876543210', 'amit@gmail.com'),

(2, 'Pooja Mehta', 'Pune', '9823456789', 'pooja@gmail.com'),

(3, 'Rahul Singh', 'Delhi', '9867896543', 'rahul@gmail.com');

INSERT INTO Branch VALUES

(1, 'Main Branch', 'Mumbai', 101),

(2, 'Pune Branch', 'Pune', 102),

(3, 'Delhi Branch', 'Delhi', 103);

INSERT INTO Account VALUES

(101, 1, 'Savings', 150000.00, 1),

(102, 2, 'Current', 80000.00, 2),

(103, 3, 'Savings', 4500.00, 3),

(104, 1, 'Current', 25000.00, 1),

(105, 2, 'Savings', 3000.00, 2);

INSERT INTO Transaction VALUES

(1, 101, 'Deposit', 5000.00, '2025-03-01'),

(2, 102, 'Withdrawal', 10000.00, '2025-03-02'),

(3, 103, 'Deposit', 2000.00, '2025-03-03'),

(4, 101, 'Deposit', 30000.00, '2025-03-04'),

(5, 104, 'Withdrawal', 5000.00, '2025-03-05');

INSERT INTO Loan VALUES

(201, 1, 200000.00, 'Home Loan', 'Approved'),

(202, 2, 50000.00, 'Personal Loan', 'Pending'),

(203, 3, 120000.00, 'Car Loan', 'Approved'),

(204, 1, 100000.00, 'Education Loan', 'Rejected'),

(205, 2, 300000.00, 'Home Loan', 'Approved');

INSERT INTO Employee VALUES

(101, 'Rajesh Kumar', 'Manager', 1, 75000.00),

(102, 'Sonia Gupta', 'Clerk', 2, 35000.00),

(103, 'Vikas Patel', 'Manager', 3, 80000.00),

(104, 'Asha Verma', 'Clerk', 1, 36000.00);

SELECT c.customer\_id, c.name, a.account\_id, a.account\_type, a.balance, a.branch\_id

FROM Customer c

JOIN Account a ON c.customer\_id = a.customer\_id;

SELECT b.branch\_id, b.branch\_name, SUM(a.balance) AS total\_balance

FROM Branch b

JOIN Account a ON b.branch\_id = a.branch\_id

GROUP BY b.branch\_id, b.branch\_name;

SELECT c.customer\_id, c.name, l.amount, l.loan\_type

FROM Customer c

JOIN Loan l ON c.customer\_id = l.customer\_id

WHERE l.amount > 100000;

SELECT \* FROM Transaction WHERE account\_id = 101;

SELECT DISTINCT c.customer\_id, c.name

FROM Customer c

JOIN Account a ON c.customer\_id = a.customer\_id

JOIN Loan l ON c.customer\_id = l.customer\_id;

CREATE VIEW HighValueCustomers AS

SELECT c.customer\_id, c.name, a.account\_id, a.balance

FROM Customer c

JOIN Account a ON c.customer\_id = a.customer\_id

WHERE a.balance > 100000;

SELECT \* FROM Employee WHERE branch\_id = 3;

SELECT \* FROM Transaction ORDER BY amount DESC LIMIT 1;

SELECT \* FROM Account WHERE balance < 5000;

UPDATE Account SET balance = balance + 2000 WHERE account\_id = 105;

DELETE FROM Loan WHERE status = 'Rejected';

SELECT loan\_type, SUM(amount) AS total\_loan\_amount FROM Loan GROUP BY loan\_type;

```