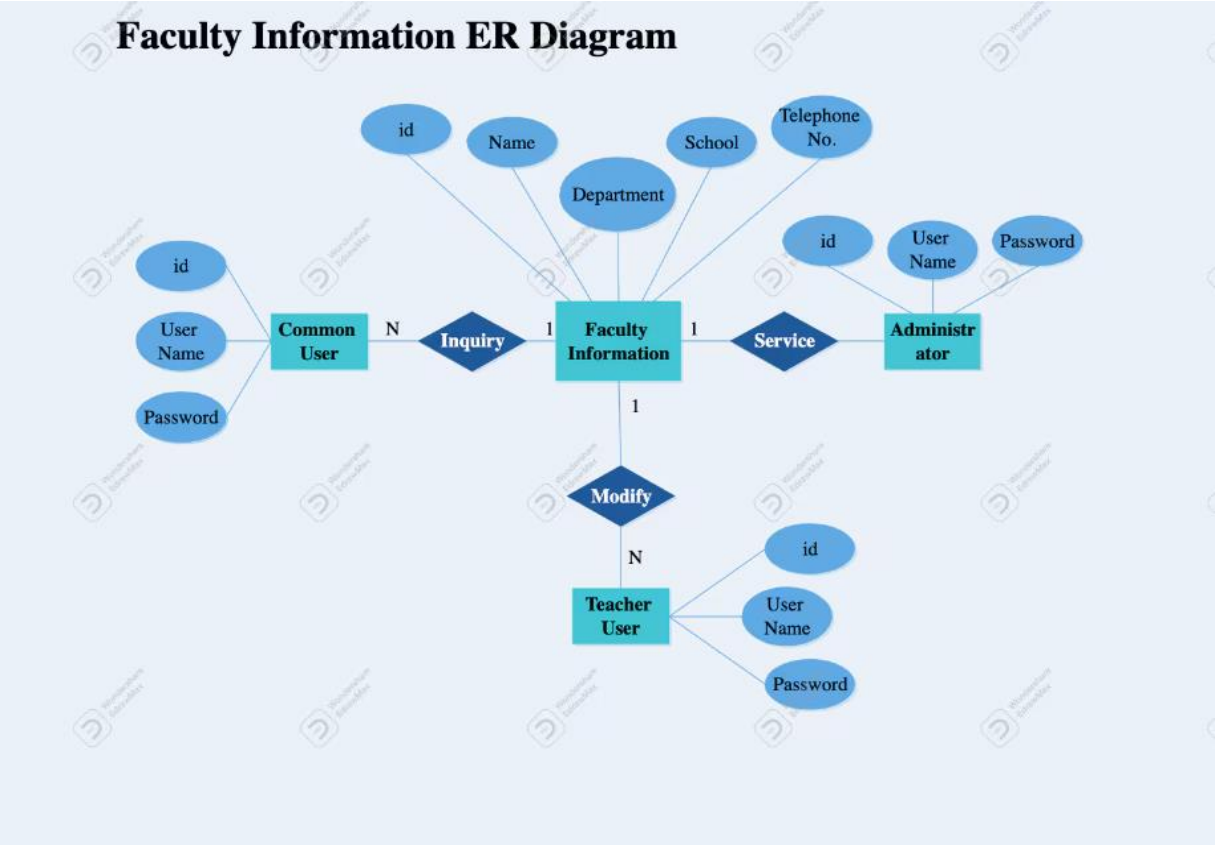
1.




```
create database faculty;
use faculty;
--------------------------------------------------------------------------------
create table facultyInformation(
id int primary key,
name varchar(50),
department varchar(50),
telephoneNo varchar(15)
);
select *
from facultyInformation;
alter table facultyInformation
add school varchar(50);

CREATE TABLE CommonUser (
    id INT PRIMARY KEY,
    UserName VARCHAR(50),
    Password VARCHAR(50)
);
```

```
CREATE TABLE Administrator (
    id INT PRIMARY KEY,
    UserName VARCHAR(50),
    Password VARCHAR(50)
);
-----------------------------------------------------------------------
CREATE TABLE TeacherUser (
    id INT PRIMARY KEY,
    UserName VARCHAR(50),
    Password VARCHAR(50)
);
-------------------------------------------------------------------------------
CREATE TABLE Inquiry (
    FacultyId INT,
    CommonUserId INT,
    FOREIGN KEY (FacultyId) REFERENCES FacultyInformation(id),
    FOREIGN KEY (CommonUserId) REFERENCES CommonUser(id),
    PRIMARY KEY (FacultyId, CommonUserId)
);
-----------------------------------------------------------------------------------------------------------
CREATE TABLE Service (
    FacultyId INT,
    AdministratorId INT,
    FOREIGN KEY (FacultyId) REFERENCES FacultyInformation(id),
    FOREIGN KEY (AdministratorId) REFERENCES Administrator(id),
    PRIMARY KEY (FacultyId, AdministratorId)
);
-----------------------------------------------------------------------------------------------------------
CREATE TABLE Modify (
    FacultyId INT,
    TeacherUserId INT,
    FOREIGN KEY (FacultyId) REFERENCES FacultyInformation(id),
    FOREIGN KEY (TeacherUserId) REFERENCES TeacherUser(id),
    PRIMARY KEY (FacultyId, TeacherUserId)
);
```

Stored procedure

```
DELIMITER //
CREATE PROCEDURE InsertFaculty (
   IN faculty_id INT,
   IN faculty_name VARCHAR(50),
   IN faculty_dept VARCHAR(50),
   IN faculty_tel VARCHAR(15),
         IN faculty_school VARCHAR(50)
)
BEGIN
   INSERT INTO FacultyInformation (id, Name, Department,
   TelephoneNo,school)
   VALUES (faculty_id, faculty_name, faculty_dept,
   faculty_tel, faculty_school);
END //
DELIMITER ;
```

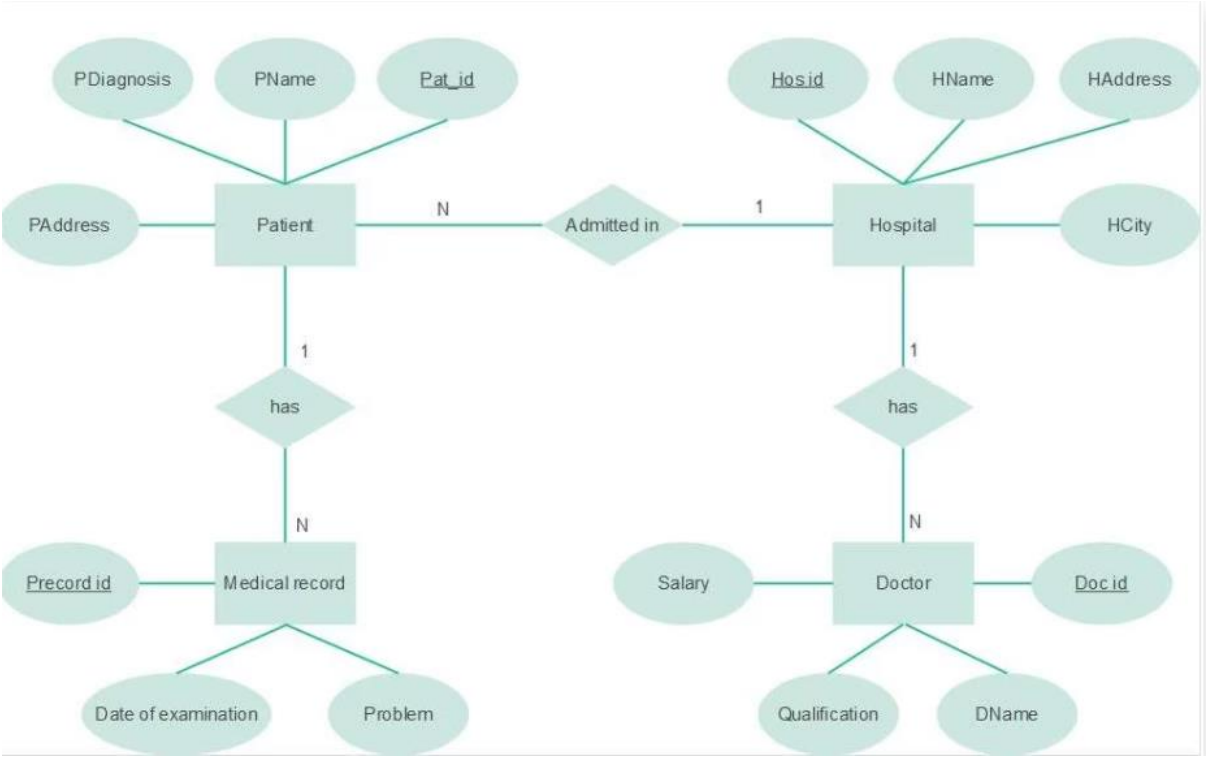
```
call insertFaculty(101, 'John Doe', 'Computer Science',  '1234567890','Engineering');
call insertFaculty(103, 'Jane Smith', 'Mathematics', '9876543210','Science');
call insertfaculty(105, 'Emily Davis', 'Chemistry', '7891234560', 'Science');

select *from facultyInformation;
delete from facultyInformation where id=1;

INSERT INTO CommonUser (id, UserName, Password)
VALUES
(101, 'common_user1', 'password123'),
(102, 'common_user2', 'password456'),
(103, 'common_user3', 'password789'),
(104, 'common_user4', 'password101');

INSERT INTO CommonUser (id, UserName, Password)
VALUES
(105, 'common_user5', 'password105');
delete from CommonUser where id in(1,2,3,4);
```

```
INSERT INTO Administrator (id, UserName, Password)
VALUES
(101, 'admin_user1', 'adminpass123'),
(102, 'admin_user2', 'adminpass456'),
(103, 'admin_user3', 'adminpass789'),
(104, 'admin_user4', 'adminpass101');
INSERT INTO Administrator (id, UserName, Password)
VALUES
(105, 'admin_user5', 'adminpass105');
-------------------------------------------------------------------------------
INSERT INTO TeacherUser (id, UserName, Password)
VALUES
(101, 'teacher_user1', 'teachpass123'),
(102, 'teacher_user2', 'teachpass456'),
(103, 'teacher_user3', 'teachpass789'),
(104, 'teacher_user4', 'teachpass101');
INSERT INTO TeacherUser (id, UserName, Password)
VALUES
(105, 'teacher_user5', 'teachpass105');
-----------------------------------------------------------------------------------------------
-- Insert records into Inquiry table
INSERT INTO Inquiry (FacultyId, CommonUserId)
VALUES
(101, 101),
(102, 102),
(103,103),
(104,104),
(105,105);
-----------------------------------------------------------------------------------------------
-- Insert records into Modify table
INSERT INTO Modify (FacultyId, TeacherUserId)
VALUES
(101, 101),
(102, 102),
(103,103),
(104,104),
(105,105);
```

```
-----------------------------------------------------------------------------------------------------
-- Insert records into Service table
INSERT INTO Service (FacultyId, AdministratorId)
VALUES
(101, 101),
(102, 102),
(103,103),
(104,104),
(105,105);
-----------------------------------------------------------------------------------------------
select *from commonuser;
select* from administrator;
select * from teacheruser;
select*  from inquiry;
select* from modify;
select* from service;
-----------------------------------------------------------------------------------------------------------
-- join conditions
-- Verify FacultyInformation with Inquiry and CommonUser
SELECT
    f.id AS FacultyId,
    f.Name AS FacultyName,
    c.id AS CommonUserId,
    c.UserName AS CommonUserName
FROM
    Inquiry i
JOIN
    FacultyInformation f ON i.FacultyId = f.id
JOIN
    CommonUser c ON i.CommonUserId = c.id;
```

```
-- Verify Modify Relationship with FacultyInformation and TeacherUser
SELECT
    f.id AS FacultyId,
    f.Name AS FacultyName,
    t.id AS TeacherUserId,
    t.UserName AS TeacherUserName
FROM
    Modify m
JOIN
    FacultyInformation f ON m.FacultyId = f.id
JOIN
    TeacherUser t ON m.TeacherUserId = t.id;

-- Verify Service Relationship with FacultyInformation and Administrator
SELECT
    f.id as ServiceId,
    f.id AS FacultyId,
    f.Name AS FacultyName,
    f.Department AS FacultyDepartment,
    a.id AS AdministratorId,
    a.UserName AS AdministratorName
FROM
    Service s
JOIN
    FacultyInformation f ON s.FacultyId = f.id
JOIN
    Administrator a ON s.AdministratorId = a.id;
```

2.

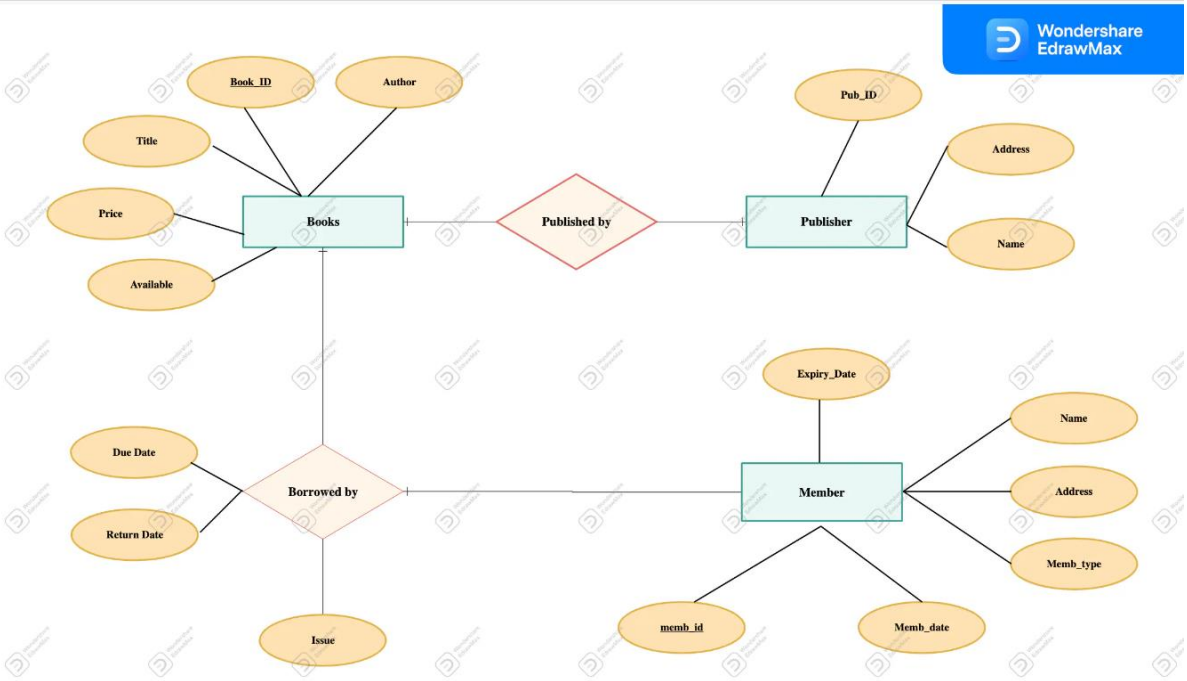



```
create database hospitaldb;
use hospitaldb;

create table hospital(
hosid int primary key,
hname varchar(50),
haddress varchar(115),
hcity varchar(25)
);
-----------------------------------------------------------------------------------------------
create table patient(
pat_id int primary key,
pname varchar(50) not null,
pdiagnosis varchar(100)not null,
paddress varchar(115),
hosid int,
foreign key(hosid)references hospital(hosid)
);
-----------------------------------------------------------------------------------------------
create table doctor(
docid int primary key,
dname varchar(50) not null,
```

```
qualifictaion varchar(100),
salary decimal(10,2),
hosid int,
foreign key (hosid) references hospital(hosid)
);
-----------------------------------------------------------------------------------------------
create table medicalrecord(
precordid int primary key,
date_of_examination date,
problem varchar(200),
patid int ,
foreign key (patid) references patient(pat_id)
);
-----------------------------------------------------------------------------------------------------------
select*from hospital;
insert into hospital(hosid,hname,haddress,hcity) values
(1,'city hospital','123main set','new york'),
(2,'green valley hospita','345 oak ave','los angeles'),
(3,'sunrise hospital','789 pine set','chicago');

insert into patient (pat_id,pname,pdiagnosis,paddress,hosid) values
(101,'john','diabetes','123 main st',1),
(102,'jane smith','hypertension','200 pine set',2),
(103,'robert brown','asthma','300 maple st',1);

INSERT INTO Doctor (Docid, DName,qualifictaion, Salary, Hosid) VALUES
 (201, 'Dr. Alice Johnson', 'MD - Cardiology', 150000, 1),
(202, 'Dr. Mark Wilson', 'MD - Neurology', 130000, 2),
 (203, 'Dr. Sarah Lee', 'MD - Pediatrics', 120000, 1);

 INSERT INTO MedicalRecord (precordid, Date_of_examination, Problem, patid)
VALUES
(301, '2024-09-01', 'Routine Check-up', 101),
(302, '2024-09-10', 'Blood Pressure Monitoring', 102),
(303, '2024-09-15', 'Asthma Attack Treatment', 103);
```

```
SELECT * FROM Hospital;
SELECT * FROM Patient;
SELECT * FROM Doctor;
SELECT * FROM MedicalRecord;

select h.hname,p.pat_id,p.pname,p.pdiagnosis
from patient as p
join hospital as h on p.hosid=h.hosid;

select d.docid,d.dname,d.qualifictaion,d.salary,h.hname,h.hcity
from doctor  as d
join hospital h on
d.hosid=h.hosid;



DELIMITER //
CREATE TRIGGER PreventLowSalaryDoctor
BEFORE INSERT ON Doctor
FOR EACH ROW
BEGIN
   -- If the new doctor's salary is less than 100,000, raise an error
   IF NEW.salary < 100000 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Doctor salary must be at least 100,000.';
   END IF;
END //
DELIMITER ;

-- This should fail since the salary is less than 100,000
INSERT INTO Doctor (docid, dname, qualifictaion, salary, hosid)
VALUES (204, 'Dr. Test', 'Test Qualification', 90000, 1);
```

3.




```
create database library;
use library;
CREATE TABLE Publisher (
    Pub_ID INT PRIMARY KEY,
    Name VARCHAR(100),
    Address VARCHAR(255)
);
```

```
CREATE TABLE Books (
    Book_ID INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(100),
    Price DECIMAL(10, 2),
    Available BOOLEAN,
    Pub_ID INT,
    FOREIGN KEY (Pub_ID) REFERENCES Publisher(Pub_ID)
);
```

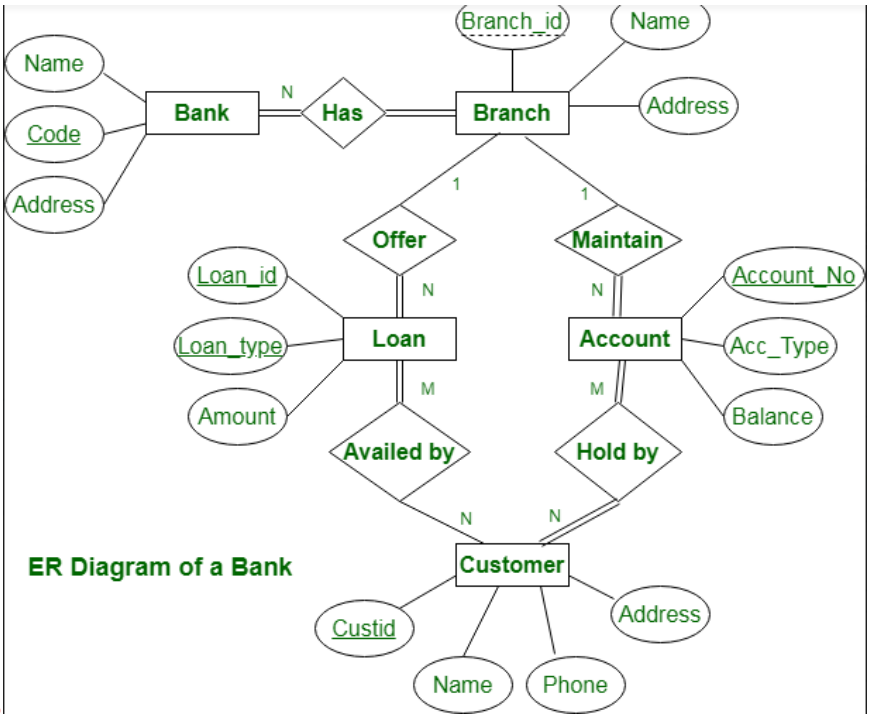
```
CREATE TABLE Member (
    memb_id INT PRIMARY KEY,
    Name VARCHAR(100),
    Address VARCHAR(255),
    Memb_type VARCHAR(50),
```

```
    Memb_date DATE,
    Expiry_Date DATE
);
-----------------------------------------------------------------------------------------------------------
CREATE TABLE Borrowed_by (
    Book_ID INT,
    memb_id INT,
    Issue DATE,
    DueDate DATE,
    ReturnDate DATE,
    PRIMARY KEY (Book_ID, memb_id),
    FOREIGN KEY (Book_ID) REFERENCES Books(Book_ID),
    FOREIGN KEY (memb_id) REFERENCES Member(memb_id)
);
-----------------------------------------------------------------------------------------------
INSERT INTO Publisher (Pub_ID, Name, Address)
VALUES
(101, 'Penguin Books', '123 Penguin St, NY'),
(102, 'HarperCollins', '456 Harper Ave, CA'),
(103, 'Simon & Schuster', '789 Simon Rd, TX');

INSERT INTO Books (Book_ID, Title, Author, Price, Available, Pub_ID)
    VALUES(1, 'The Great Gatsby', 'F. Scott Fitzgerald', 300.00, TRUE, 101),
(2, '1984', 'George Orwell', 350.00, TRUE, 102),
(3, 'To Kill a Mockingbird', 'Harper Lee', 400.00, TRUE, 102);

INSERT INTO Member (memb_id, Name, Address, Memb_type, Memb_date, Expiry_Date)
VALUES
(1, 'John Doe', '100 Main St, NY', 'Gold', '2024-01-01', '2024-12-31'),
(2, 'Jane Smith', '200 Maple Ave, CA', 'Silver', '2024-03-15', '2024-12-15'),
(3, 'Emily Johnson', '300 Oak Blvd, TX', 'Platinum', '2024-05-20', '2024-11-20');

INSERT INTO Borrowed_by (Book_ID, memb_id, Issue, DueDate, ReturnDate)
VALUES
(1, 1, '2024-10-01', '2024-10-15', NULL),
(2, 2, '2024-10-02', '2024-10-16', NULL),
(3, 3, '2024-10-03', '2024-10-17', NULL);
```

```
select*from publisher;
select*from books;
select*from member;

-- join's condition to check relation
SELECT B.Book_ID, B.Title, B.Author, B.Price, P.Name AS Publisher_Name, P.Address
FROM Books B
INNER JOIN Publisher P ON B.Pub_ID = P.Pub_ID;

-- Join with Filtering:  books that have been borrowed

SELECT B.Book_ID, B.Title, M.Name AS Borrowed_By, BB.Issue, BB.DueDate
FROM Books B
INNER JOIN Borrowed_by BB ON B.Book_ID = BB.Book_ID
INNER JOIN Member M ON BB.memb_id = M.memb_id
WHERE B.Available = FALSE;

-- Stored Procedure to Count Total Books Borrowed by a Member
DELIMITER //

CREATE PROCEDURE GetTotalBooksBorrowedByMember(
    IN p_memb_id INT
)
BEGIN
    SELECT COUNT(*) AS TotalBooksBorrowed
    FROM Borrowed_by
    WHERE memb_id = p_memb_id;
END //

DELIMITER ;

CALL GetTotalBooksBorrowedByMember(1);

-- TRIGGER

DELIMITER //
```

```
CREATE TRIGGER UpdateBookAvailability
AFTER INSERT ON Borrowed_by
FOR EACH ROW
BEGIN
   UPDATE Books
   SET Available = FALSE
   WHERE Book_ID = NEW.Book_ID;
END //

DELIMITER ;
-- TESTING THE SETUP BY ADDIG ENTRIES.
INSERT INTO Books (Book_ID, Title, Author, Price, Available, Pub_ID)
VALUES
(4, 'The Catcher in the Rye', 'J.D. Salinger', 500.00, TRUE, 101);
-- INSERT IN BORROWED
INSERT INTO Borrowed_by (Book_ID, memb_id, Issue, DueDate, ReturnDate)
VALUES (4, 1, '2024-10-03', '2024-10-17', NULL);

-- check if the Trigger Updated the Books Table:

SELECT Book_ID, Title, Available
FROM Books
WHERE Book_ID = 4;

**********table 3 compeleted**********
```

4.




```
create database BankDB;
use BankDB;

CREATE TABLE Bank (
    Code VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(50),
    Address VARCHAR(100)
);
-----------------------------------------------------------------------------------------------------------
CREATE TABLE Branch (
    Branch_id INT PRIMARY KEY,
    Name VARCHAR(50),
    Address VARCHAR(100),
    Bank_Code VARCHAR(10),
    FOREIGN KEY (Bank_Code) REFERENCES Bank(Code)
);
-----------------------------------------------------------------------------------------------------------
CREATE TABLE Loan (
    Loan_id INT PRIMARY KEY,
    Loan_type VARCHAR(50),
    Amount DECIMAL(15,2),
    Branch_id INT,
    FOREIGN KEY (Branch_id) REFERENCES Branch(Branch_id)
);
```

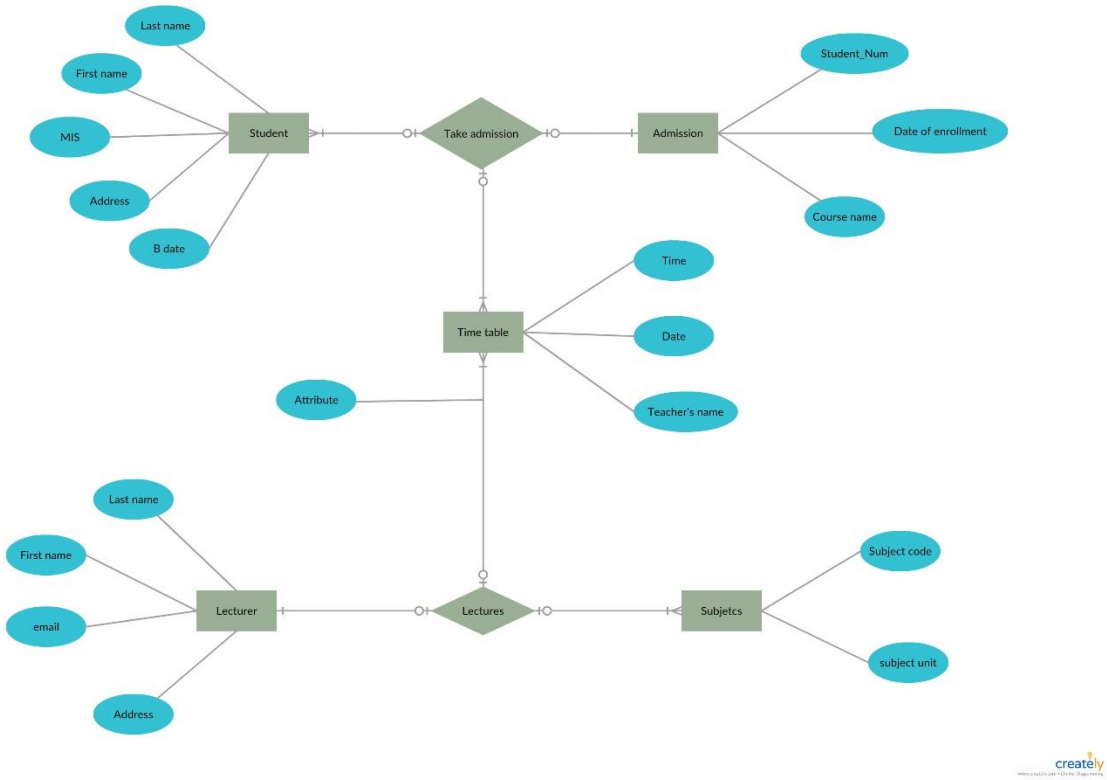
```
CREATE TABLE Account (
    Account_No INT PRIMARY KEY,
    Acc_Type VARCHAR(50),
    Balance DECIMAL(15,2),
    Branch_id INT,
    FOREIGN KEY (Branch_id) REFERENCES Branch(Branch_id)
);
-----------------------------------------------------------------------------------------------------------------
CREATE TABLE Customer (
    Custid INT PRIMARY KEY,
    Name VARCHAR(50),
    Address VARCHAR(100),
    Phone VARCHAR(15)
);
-----------------------------------------------------------------------------------------------------------
-- Create Junction table for Loan-Customer relationship (Availed by)
CREATE TABLE Customer_Loan (
    Custid INT,
    Loan_id INT,
    PRIMARY KEY (Custid, Loan_id),
    FOREIGN KEY (Custid) REFERENCES Customer(Custid),
    FOREIGN KEY (Loan_id) REFERENCES Loan(Loan_id)
);
-----------------------------------------------------------------------------------------------------------------
-- Create Junction table for Account-Customer relationship (Hold by)
CREATE TABLE Customer_Account (
    Custid INT,
    Account_No INT,
    PRIMARY KEY (Custid, Account_No),
    FOREIGN KEY (Custid) REFERENCES Customer(Custid),
    FOREIGN KEY (Account_No) REFERENCES Account(Account_No)
);
-----------------------------------------------------------------------------------------
INSERT INTO Bank (Code, Name, Address)
VALUES ('B001', 'Global Bank', '123 Main St'),
       ('B002', 'City Bank', '456 Oak St'),
```

```
    ('B003', 'Metro Bank', '789 Pine St');

INSERT INTO Branch (Branch_id, Name, Address, Bank_Code)
VALUES (101, 'Downtown Branch', '456 Elm St', 'B001'),
    (102, 'Uptown Branch', '789 Maple St', 'B002'),
    (103, 'Suburban Branch', '321 Birch St', 'B003');

INSERT INTO Customer (Custid, Name, Address, Phone)
VALUES (1, 'John Doe', '789 Maple St', '1234567890'),
    (2, 'Jane Smith', '101 Oak St', '0987654321'),
    (3, 'Alice Johnson', '102 Pine St', '1122334455');

INSERT INTO Loan (Loan_id, Loan_type, Amount, Branch_id)
VALUES (1001, 'Home Loan', 50000, 101),
    (1002, 'Car Loan', 15000, 102),
    (1003, 'Education Loan', 30000, 103);

INSERT INTO Account (Account_No, Acc_Type, Balance, Branch_id)
VALUES (5001, 'Savings', 1000, 101),
    (5002, 'Checking', 2000, 102),
    (5003, 'Fixed Deposit', 5000, 103);
-- insert into relationship
INSERT INTO Customer_Loan (Custid, Loan_id)
VALUES (1, 1001),
    (2, 1002),
    (3, 1003);

-- Insert into Customer_Account (Hold by)
INSERT INTO Customer_Account (Custid, Account_No)
VALUES (1, 5001),
    (2, 5002),
    (3, 5003);
select *from bank;
select*from branch;
select*from loan;
select*from account;
select*from Customer_Loan;
```

```
select*from Customer_Account;

-- Find customers who have loans from a specific branch
SELECT c.Name, l.Loan_type, l.Amount, b.Name AS Branch_Name
FROM Customer c
JOIN Customer_Loan cl ON c.Custid = cl.Custid
JOIN Loan l ON cl.Loan_id = l.Loan_id
JOIN Branch b ON l.Branch_id = b.Branch_id
WHERE b.Branch_id = 101;

-- store procedure
DELIMITER //

CREATE PROCEDURE GetCustomersByLoan(IN loanId INT)
BEGIN
    SELECT c.Name, c.Address, c.Phone
    FROM Customer c
    JOIN Customer_Loan cl ON c.Custid = cl.Custid
    WHERE cl.Loan_id = loanId;
END //

DELIMITER ;

CALL GetCustomersByLoan(1001);

-- trigger
-- Create Loan_Audit Table
CREATE TABLE Loan_Audit (
    Audit_id INT AUTO_INCREMENT PRIMARY KEY,
    Loan_id INT,
    Old_Amount DECIMAL(15,2),
    New_Amount DECIMAL(15,2),
    Updated_At TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
DELIMITER //

CREATE TRIGGER loan_update_audit
```

```
AFTER UPDATE ON Loan
FOR EACH ROW
BEGIN
   IF OLD.Amount <> NEW.Amount THEN
      INSERT INTO Loan_Audit (Loan_id, Old_Amount, New_Amount)
      VALUES (NEW.Loan_id, OLD.Amount, NEW.Amount);
   END IF;
END //

DELIMITER ;

SELECT * FROM Loan_Audit;
UPDATE Loan
SET Amount = 60000
WHERE Loan_id = 1002;
SELECT * FROM Loan_Audit;
```

5.



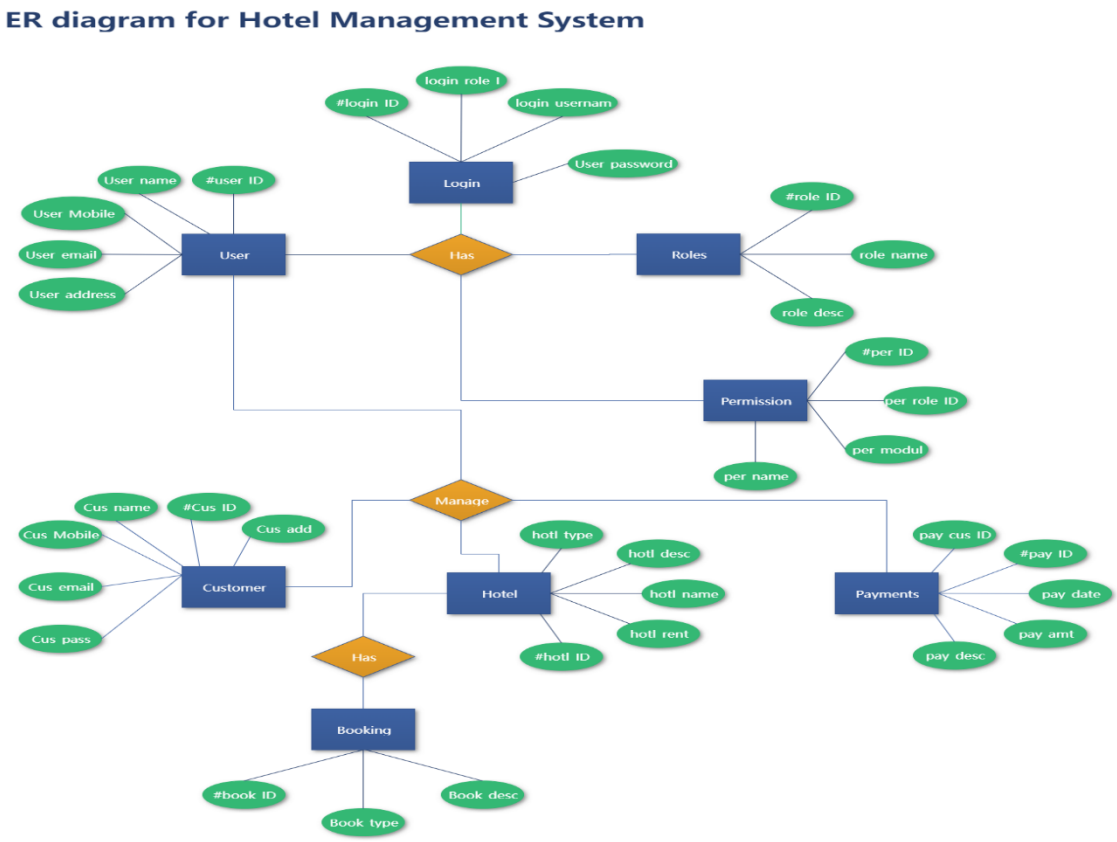

```
CREATE DATABASE UniversityDB;
USE UniversityDB;

-- Create Student Table
CREATE TABLE Student (
    MIS INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Address VARCHAR(100),
    BirthDate DATE
);

-- Create Admission Table
CREATE TABLE Admission (
    Student_Num INT,
    Date_of_Enrollment DATE,
    Course_Name VARCHAR(100),
    PRIMARY KEY (Student_Num),
    FOREIGN KEY (Student_Num) REFERENCES Student(MIS)
);
```

```
-- Create TimeTable Table
CREATE TABLE TimeTable (
    Attribute INT AUTO_INCREMENT PRIMARY KEY,
    Time TIME,
    Date DATE,
    TeacherName VARCHAR(100)
);

-- Create Lecturer Table
CREATE TABLE Lecturer (
    email VARCHAR(100) PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    Address VARCHAR(100)
);

-- Create Subjects Table
CREATE TABLE Subjects (
    SubjectCode INT PRIMARY KEY,
    SubjectUnit VARCHAR(100),
    LecturerEmail VARCHAR(100),
    FOREIGN KEY (LecturerEmail) REFERENCES Lecturer(email)
);
-- Insert records into Student
INSERT INTO Student (MIS, FirstName, LastName, Address, BirthDate)
VALUES (1, 'John', 'Doe', '123 Main St', '2000-01-15'),
       (2, 'Jane', 'Smith', '456 Oak St', '2001-03-22'),
       (3, 'Alice', 'Johnson', '789 Pine St', '1999-06-10');

-- Insert records into Admission
INSERT INTO Admission (Student_Num, Date_of_Enrollment, Course_Name)
VALUES (1, '2020-09-01', 'Computer Science'),
       (2, '2021-01-10', 'Mathematics'),
       (3, '2020-09-01', 'Physics');

-- Insert records into TimeTable
INSERT INTO TimeTable (Time, Date, TeacherName)
```

```
VALUES ('09:00:00', '2024-01-01', 'Dr. Brown'),
    ('11:00:00', '2024-01-01', 'Prof. Green'),
    ('13:00:00', '2024-01-01', 'Ms. White');

-- Insert records into Lecturer
INSERT INTO Lecturer (email, FirstName, LastName, Address)
VALUES ('brown@university.edu', 'Robert', 'Brown', '101 Elm St'),
    ('green@university.edu', 'Linda', 'Green', '202 Birch St'),
    ('white@university.edu', 'Helen', 'White', '303 Cedar St');

-- Insert records into Subjects
INSERT INTO Subjects (SubjectCode, SubjectUnit, LecturerEmail)
VALUES (101, 'Database Systems', 'brown@university.edu'),
    (102, 'Calculus', 'green@university.edu'),
    (103, 'Physics 101', 'white@university.edu');

SELECT * FROM Student;
SELECT * FROM Admission;
SELECT * FROM TimeTable;
SELECT * FROM Lecturer;
SELECT * FROM Subjects;

-- join Query 1: Join Student and Admission
SELECT s.FirstName, s.LastName, a.Course_Name, a.Date_of_Enrollment
FROM Student s
JOIN Admission a ON s.MIS = a.Student_Num;
-- Join Lecturer and Subjects
SELECT l.FirstName, l.LastName, s.SubjectUnit
FROM Lecturer l
JOIN Subjects s ON l.email = s.LecturerEmail;
-- Join Student, Admission, and TimeTable
SELECT st.FirstName, st.LastName, ad.Course_Name, tt.TeacherName, tt.Date
FROM Student st
JOIN Admission ad ON st.MIS = ad.Student_Num
JOIN TimeTable tt ON tt.TeacherName = 'Dr. Brown'; -- You can modify this to match other
relationships
```

```
DELIMITER //

CREATE PROCEDURE GetStudentCourses()
BEGIN
   SELECT s.FirstName, s.LastName, a.Course_Name
   FROM Student s
   JOIN Admission a ON s.MIS = a.Student_Num;
END //

DELIMITER ;

call GetStudentCourses();
-- trigger
DELIMITER //

CREATE TRIGGER Check_BirthDate_Before_Update
BEFORE UPDATE ON Student
FOR EACH ROW
BEGIN
   IF NEW.BirthDate > CURDATE() THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'BirthDate cannot be a future date.';
   END IF;
END //

DELIMITER ;
SELECT * FROM Student;
UPDATE Student SET BirthDate = '2025-01-01' WHERE MIS = 1;  -- BirthDate cannot be a future date.
UPDATE Student SET BirthDate = '1999-12-31' WHERE MIS = 1;
```

6.




```
CREATE DATABASE hotel_management;
USE hotel_management;

CREATE TABLE Login (
    login_id INT PRIMARY KEY,
    login_username VARCHAR(50),
    login_password VARCHAR(50)
);

CREATE TABLE User (
    user_id INT PRIMARY KEY,
    user_name VARCHAR(50),
    user_mobile VARCHAR(20),
    user_email VARCHAR(50),
    user_address VARCHAR(100),
    login_id INT,
    FOREIGN KEY (login_id) REFERENCES Login(login_id)
);

CREATE TABLE Roles (
```

```
    role_id INT PRIMARY KEY,
    role_name VARCHAR(50),
    role_desc TEXT
);

CREATE TABLE Permission (
    per_id INT PRIMARY KEY,
    per_role_id INT,
    per_modul VARCHAR(50),
    per_name VARCHAR(50),
    FOREIGN KEY (per_role_id) REFERENCES Roles(role_id)
);

CREATE TABLE Customer (
    cus_id INT PRIMARY KEY,
    cus_name VARCHAR(50),
    cus_mobile VARCHAR(20),
    cus_email VARCHAR(50),
    cus_address VARCHAR(100),
    cus_pass VARCHAR(50)
);

CREATE TABLE Hotel (
    hotel_id INT PRIMARY KEY,
    hotel_name VARCHAR(50),
    hotel_type VARCHAR(20),
    hotel_desc TEXT,
    hotel_rent DECIMAL(10, 2)
);

CREATE TABLE Payments (
    pay_id INT PRIMARY KEY,
    pay_cus_id INT,
    pay_date DATE,
    pay_amt DECIMAL(10, 2),
    pay_desc TEXT,
    FOREIGN KEY (pay_cus_id) REFERENCES Customer(cus_id)
```

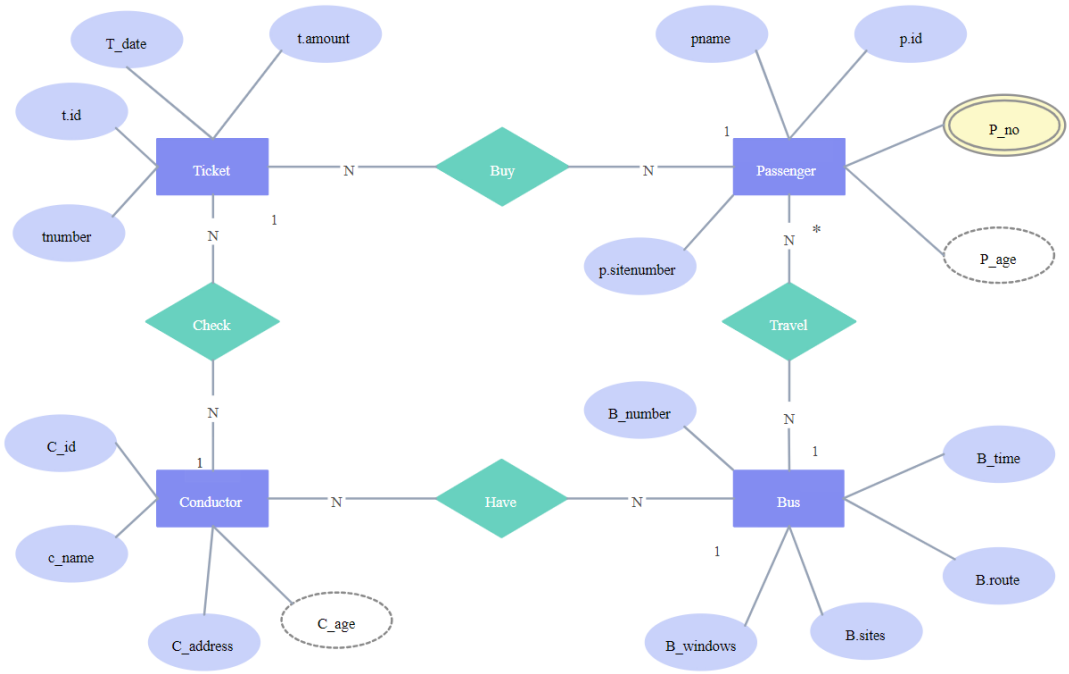
```
);

CREATE TABLE Booking (
    book_id INT PRIMARY KEY,
    book_desc TEXT,
    book_type VARCHAR(20),
    user_id INT,
    hotel_id INT,
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (hotel_id) REFERENCES Hotel(hotel_id)
);

-- Create a junction table for the many-to-many relationship between User and Roles
CREATE TABLE User_Roles (
    user_id INT,
    role_id INT,
    PRIMARY KEY (user_id, role_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (role_id) REFERENCES Roles(role_id));

-- Login table
INSERT INTO Login (login_id, login_username, login_password) VALUES
(1, 'user1', 'password1'),
(2, 'user2', 'password2'),
(3, 'user3', 'password3');

-- User table
INSERT INTO User (user_id, user_name, user_mobile, user_email, user_address, login_id) VALUES
(1, 'John Doe', '1234567890', 'johndoe@example.com', '123 Main St', 1),
(2, 'Jane Smith', '9876543210', 'janesmith@example.com', '456 Elm St', 2),
(3, 'Alice Johnson', '5555555555', 'alicejohnson@example.com', '789 Oak St', 3);

-- Roles table
INSERT INTO Roles (role_id, role_name, role_desc) VALUES
(1, 'Admin', 'Administrator role'),
(2, 'Manager', 'Manager role'),
(3, 'Guest', 'Guest role');
```

```
-- Permission table
INSERT INTO Permission (per_id, per_role_id, per_modul, per_name) VALUES
(1, 1, 'Hotel', 'Add Hotel'),
(2, 2, 'Booking', 'View Bookings'),
(3, 3, 'Payment', 'Make Payment');

-- Customer table
INSERT INTO Customer (cus_id, cus_name, cus_mobile, cus_email, cus_address, cus_pass) VALUES
(1, 'Customer1', '1111111111', 'customer1@example.com', 'Address1', 'pass1'),
(2, 'Customer2', '2222222222', 'customer2@example.com', 'Address2', 'pass2'),
(3, 'Customer3', '3333333333', 'customer3@example.com', 'Address3', 'pass3');

-- Hotel table
INSERT INTO Hotel (hotel_id, hotel_name, hotel_type, hotel_desc, hotel_rent) VALUES
(1, 'Hotel A', 'Luxury', 'Description for Hotel A', 100.00),
(2, 'Hotel B', 'Budget', 'Description for Hotel B', 50.00),
(3, 'Hotel C', 'Mid-Range', 'Description for Hotel C', 75.00);

-- Payments table
INSERT INTO Payments (pay_id, pay_cus_id, pay_date, pay_amt, pay_desc) VALUES
(1, 1, '2023-01-01', 100.00, 'Payment for Hotel A'),
(2, 2, '2023-02-01', 50.00, 'Payment for Hotel B'),
(3, 3, '2023-03-01', 75.00, 'Payment for Hotel C');

-- Booking table
INSERT INTO Booking (book_id, book_desc, book_type, user_id, hotel_id) VALUES
(1, 'Booking for Hotel A', 'Single Room', 1, 1),
(2, 'Booking for Hotel B', 'Double Room', 2, 2),
(3, 'Booking for Hotel C', 'Family Room', 3, 3);

-- User_Roles table
INSERT INTO User_Roles (user_id, role_id) VALUES
(1, 1),
(2, 2),
(3, 3);
-- join
```

```
SELECT User.user_id, User.user_name, Booking.book_id, Booking.book_desc
FROM User
INNER JOIN Booking ON User.user_id = Booking.user_id;

SELECT User.user_id, User.user_name, Booking.book_id, Booking.book_desc, Hotel.hotel_name
FROM User
INNER JOIN Booking ON User.user_id = Booking.user_id
INNER JOIN Hotel ON Booking.hotel_id = Hotel.hotel_id;

DELIMITER //

CREATE PROCEDURE GetAllUsers()
BEGIN
   SELECT * FROM User;
END //

DELIMITER ;
call GetAllUsers();

CREATE TABLE CustomerLog (
   log_id INT PRIMARY KEY AUTO_INCREMENT,
   cus_id INT,
   log_message VARCHAR(255),
   log_date DATETIME DEFAULT CURRENT_TIMESTAMP
);
DELIMITER //

CREATE TRIGGER after_customer_insert
AFTER INSERT ON Customer
FOR EACH ROW
BEGIN
   INSERT INTO CustomerLog (cus_id, log_message)
   VALUES (NEW.cus_id, CONCAT('New customer added: ', NEW.cus_name));
END;
//

DELIMITER ;
```

```
INSERT INTO Customer (cus_id, cus_name, cus_mobile, cus_email, cus_address, cus_pass)
VALUES (4, 'Bob Brown', '5557654321', 'bob@example.com', '321 Maple Lane', 'password101');


SELECT * FROM CustomerLog;
```

7.




```
CREATE DATABASE BusTicketSystem;

USE BusTicketSystem;

-- Passenger table

CREATE TABLE Passenger (

    p_id INT PRIMARY KEY,

    pname VARCHAR(50),

    P_no VARCHAR(15),

    P_age INT

);


-- Ticket table

CREATE TABLE Ticket (

    t_id INT PRIMARY KEY,

    T_date DATE,

    t_amount DECIMAL(10, 2),

    tnumber VARCHAR(15)

);


-- Conductor table

CREATE TABLE Conductor (

    C_id INT PRIMARY KEY,

    c_name VARCHAR(50),

    C_address VARCHAR(100),
```

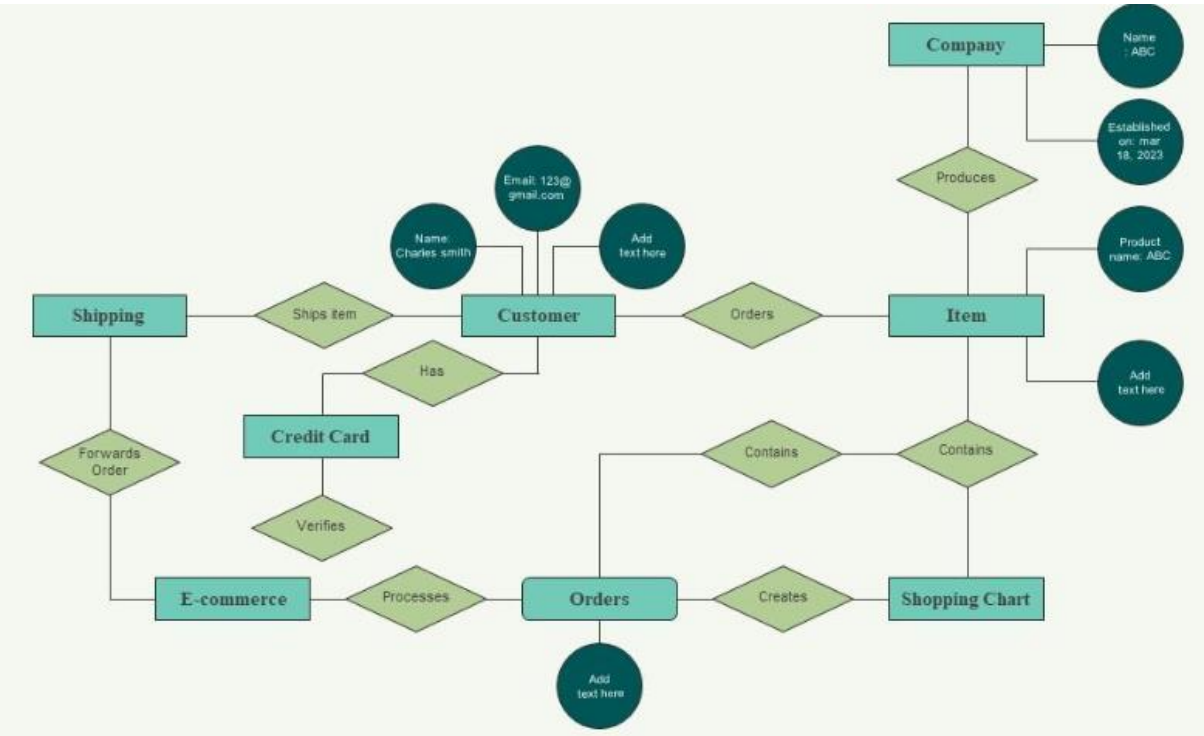
```
    C_age INT
);

-- Bus table
CREATE TABLE Bus (
    B_number VARCHAR(15) PRIMARY KEY,
    B_time TIME,
    B_windows INT,
    B_sites INT,
    B_route VARCHAR(50)
);

-- Relationship table for Passenger-Bus (many-to-many relation)
CREATE TABLE Travel (
    p_id INT,
    B_number VARCHAR(15),
    PRIMARY KEY (p_id, B_number),
    FOREIGN KEY (p_id) REFERENCES Passenger(p_id),
    FOREIGN KEY (B_number) REFERENCES Bus(B_number)
);

-- Relationship table for Ticket-Passenger (many-to-many relation)
CREATE TABLE Buy (
    t_id INT,
    p_id INT,
    PRIMARY KEY (t_id, p_id),
    FOREIGN KEY (t_id) REFERENCES Ticket(t_id),
    FOREIGN KEY (p_id) REFERENCES Passenger(p_id)
);
-- Relationship table for Conductor-Bus (many-to-many relation)
CREATE TABLE Have (
    C_id INT,
    B_number VARCHAR(15),
    PRIMARY KEY (C_id, B_number),
    FOREIGN KEY (C_id) REFERENCES Conductor(C_id),
    FOREIGN KEY (B_number) REFERENCES Bus(B_number)
);
```

```
-- Insert sample data into Passenger
INSERT INTO Passenger (p_id, pname, P_no, P_age) VALUES (1, 'John Doe', '1234567890', 30);
INSERT INTO Passenger (p_id, pname, P_no, P_age) VALUES (2, 'Jane Smith', '0987654321', 25);
INSERT INTO Passenger (p_id, pname, P_no, P_age) VALUES (3, 'Alice Johnson', '1112223334', 28);

-- Insert sample data into Ticket
INSERT INTO Ticket (t_id, T_date, t_amount, tnumber) VALUES (1, '2024-10-01', 50.00, 'T001');
INSERT INTO Ticket (t_id, T_date, t_amount, tnumber) VALUES (2, '2024-10-02', 75.00, 'T002');
INSERT INTO Ticket (t_id, T_date, t_amount, tnumber) VALUES (3, '2024-10-03', 100.00, 'T003');

-- Insert sample data into Conductor
INSERT INTO Conductor (C_id, c_name, C_address, C_age) VALUES (1, 'Mark Lee', '123 Elm Street',
40);
INSERT INTO Conductor (C_id, c_name, C_address, C_age) VALUES (2, 'Paul Green', '456 Oak Avenue',
35);
INSERT INTO Conductor (C_id, c_name, C_address, C_age) VALUES (3, 'Sara White', '789 Pine Road',
38);

-- Insert sample data into Bus
INSERT INTO Bus (B_number, B_time, B_windows, B_sites, B_route) VALUES ('B001', '09:00:00', 10,
40, 'Route 1');
INSERT INTO Bus (B_number, B_time, B_windows, B_sites, B_route) VALUES ('B002', '11:00:00', 12,
45, 'Route 2');
INSERT INTO Bus (B_number, B_time, B_windows, B_sites, B_route) VALUES ('B003', '13:00:00', 8, 35,
'Route 3');

INSERT INTO Buy (t_id, p_id) VALUES (1, 1);  -- Ticket ID 1 bought by Passenger ID 1
INSERT INTO Buy (t_id, p_id) VALUES (2, 2);  -- Ticket ID 2 bought by Passenger ID 2
INSERT INTO Buy (t_id, p_id) VALUES (3, 3);

INSERT INTO Travel (p_id, B_number) VALUES (1, 'B001');  -- Passenger ID 1 traveled on Bus B001
INSERT INTO Travel (p_id, B_number) VALUES (2, 'B002');  -- Passenger ID 2 traveled on Bus B002
INSERT INTO Travel (p_id, B_number) VALUES (3, 'B003');

INSERT INTO Have (C_id, B_number) VALUES (1, 'B001');  -- Conductor ID 1 is assigned to Bus B001
INSERT INTO Have (C_id, B_number) VALUES (2, 'B002');  -- Conductor ID 2 is assigned to Bus B002
INSERT INTO Have (C_id, B_number) VALUES (3, 'B003');
select*from passenger;
```

```
select*from ticket;
select*from conductor;
select*from bus;
select*from buy;
select*from travel;
select*from have;

select p.p_id,p.pname,b.B_number,b.B_time
from travel t
join passenger p on t.p_id=p.p_id
join bus b on t.B_number=b.B_number;

SELECT p.pname, t.T_date, t.t_amount
FROM Passenger p
INNER JOIN Buy b ON p.p_id = b.p_id
INNER JOIN Ticket t ON b.t_id = t.t_id;

-- store procedure

DELIMITER //
CREATE PROCEDURE GetTotalTicketAmount (IN passenger_id INT)
BEGIN
   SELECT Passenger.pname, SUM(Ticket.t_amount) AS Total_Amount
   FROM Ticket
   INNER JOIN Buy ON Ticket.t_id = Buy.t_id
   INNER JOIN Passenger ON Buy.p_id = Passenger.p_id
   WHERE Passenger.p_id = passenger_id
   GROUP BY Passenger.pname;
END //
DELIMITER ;

call GetTotalTicketAmount(2);
 -- trigger
 ALTER TABLE Passenger ADD COLUMN ticket_count INT DEFAULT 0;
 DELIMITER //
CREATE TRIGGER UpdateTicketCount AFTER INSERT ON Buy
FOR EACH ROW
```

```
BEGIN
    DECLARE ticket_count INT;

    -- Calculate the total number of tickets for the passenger
    SELECT COUNT(*) INTO ticket_count FROM Buy WHERE p_id = NEW.p_id;

    -- Update the passenger's ticket count (assuming there's a column to hold this data)
    UPDATE Passenger SET ticket_count = ticket_count WHERE p_id = NEW.p_id;
END //
DELIMITER ;
-- test trigger
select*from buy;
INSERT INTO Buy (t_id, p_id) VALUES (1, 2);
select*from passenger;
```

8.




```
-- Create the database
CREATE DATABASE EcommerceDB;
USE EcommerceDB;

-- Create Customer table
CREATE TABLE Customer (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL
);

-- Create Item table
CREATE TABLE Item (
    item_id INT AUTO_INCREMENT PRIMARY KEY,
    item_name VARCHAR(255) NOT NULL,
    item_price DECIMAL(10, 2) NOT NULL
);

-- Create Orders table
CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
```

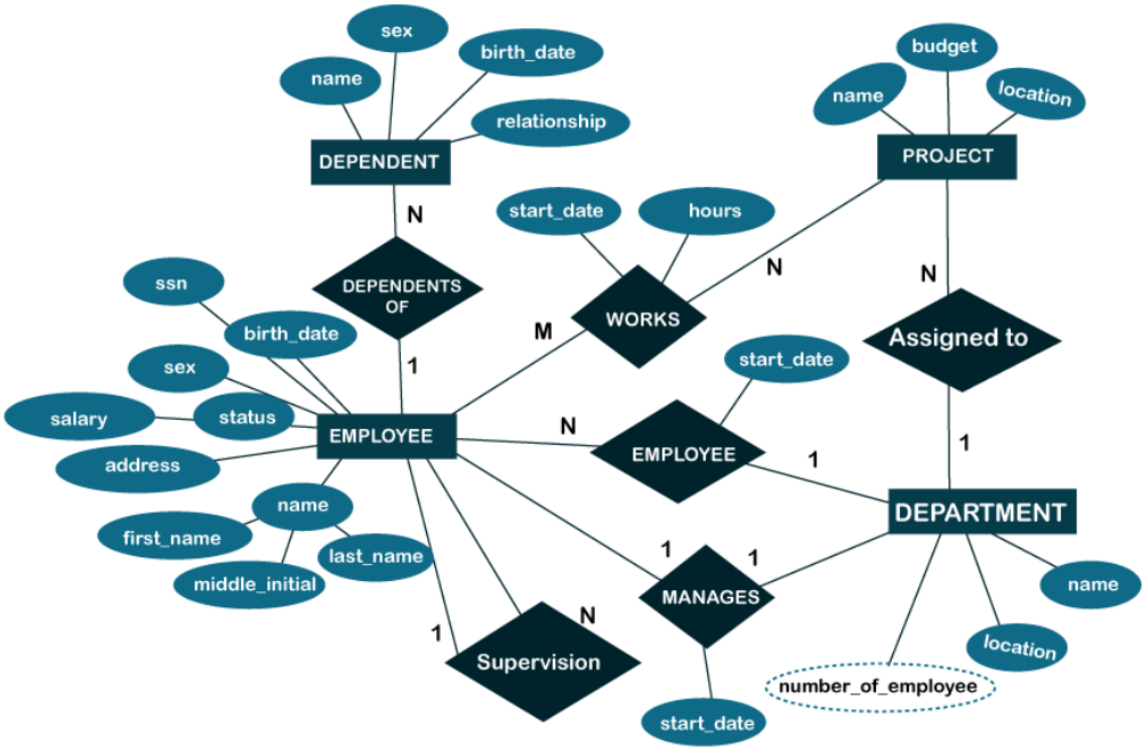
```
    order_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

-- Create ShoppingCart table (a junction table to handle many-to-many relation between Orders and Item)
CREATE TABLE ShoppingCart (
    order_id INT,
    item_id INT,
    quantity INT,
    PRIMARY KEY (order_id, item_id),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (item_id) REFERENCES Item(item_id)
);

-- Create CreditCard table
CREATE TABLE CreditCard (
    card_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    card_number VARCHAR(16) UNIQUE NOT NULL,
    expiry_date DATE,
    FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

-- Create Company table
CREATE TABLE Company (
    company_id INT AUTO_INCREMENT PRIMARY KEY,
    company_name VARCHAR(255) NOT NULL,
    established_date DATE
);

-- Create Shipping table
CREATE TABLE Shipping (
    shipping_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    shipping_date DATE,
    status VARCHAR(50),
```

```
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);

-- Create ECommerce table
CREATE TABLE ECommerce (
    ecommerce_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255),
    url VARCHAR(255)
);
-- Insert records into Customer table
INSERT INTO Customer (name, email) VALUES ('Alice Johnson', 'alice@example.com');
INSERT INTO Customer (name, email) VALUES ('Bob Smith', 'bob@example.com');
INSERT INTO Customer (name, email) VALUES ('Charlie Brown', 'charlie@example.com');

-- Insert records into Item table
INSERT INTO Item (item_name, item_price) VALUES ('Laptop', 1000.00);
INSERT INTO Item (item_name, item_price) VALUES ('Smartphone', 500.00);
INSERT INTO Item (item_name, item_price) VALUES ('Tablet', 300.00);

-- Insert records into Orders table
INSERT INTO Orders (customer_id, order_date) VALUES (1, '2024-10-01');
INSERT INTO Orders (customer_id, order_date) VALUES (2, '2024-10-02');
INSERT INTO Orders (customer_id, order_date) VALUES (3, '2024-10-03');

-- Insert records into ShoppingCart (junction table)
INSERT INTO ShoppingCart (order_id, item_id, quantity) VALUES (1, 1, 2);  -- Order 1 contains 2
Laptops
INSERT INTO ShoppingCart (order_id, item_id, quantity) VALUES (2, 2, 1);  -- Order 2 contains 1
Smartphone
INSERT INTO ShoppingCart (order_id, item_id, quantity) VALUES (3, 3, 3);  -- Order 3 contains 3
Tablets

-- Insert records into CreditCard table
INSERT INTO CreditCard (customer_id, card_number, expiry_date) VALUES (1, '1234567812345678',
'2025-12-01');
INSERT INTO CreditCard (customer_id, card_number, expiry_date) VALUES (2, '8765432187654321',
'2026-06-01');
INSERT INTO CreditCard (customer_id, card_number, expiry_date) VALUES (3, '1122334455667788',
'2027-09-01');
```

```
-- Insert records into Company table
INSERT INTO Company (company_name, established_date) VALUES ('TechCorp', '2020-05-01');
INSERT INTO Company (company_name, established_date) VALUES ('SoftSolutions', '2018-09-10');
INSERT INTO Company (company_name, established_date) VALUES ('GadgetHub', '2015-03-15');

-- Insert records into Shipping table
INSERT INTO Shipping (order_id, shipping_date, status) VALUES (1, '2024-10-02', 'Shipped');
INSERT INTO Shipping (order_id, shipping_date, status) VALUES (2, '2024-10-03', 'In Process');
INSERT INTO Shipping (order_id, shipping_date, status) VALUES (3, '2024-10-04', 'Delivered');

-- Insert records into ECommerce table
INSERT INTO ECommerce (name, url) VALUES ('Amazon', 'https://www.amazon.com');
INSERT INTO ECommerce (name, url) VALUES ('eBay', 'https://www.ebay.com');
INSERT INTO ECommerce (name, url) VALUES ('Shopify', 'https://www.shopify.com');

select*from Customer;
select*from Item;
select*from Orders;
select*from ShoppingCart;
select*from  CreditCard;
select*from Company  ;
select*from Shipping;

-- Join Customer, Orders, and ShoppingCart Table
SELECT Customer.name, Orders.order_id, Orders.order_date, Item.item_name,
ShoppingCart.quantity
FROM Customer
JOIN Orders ON Customer.customer_id = Orders.customer_id
JOIN ShoppingCart ON Orders.order_id = ShoppingCart.order_id
JOIN Item ON ShoppingCart.item_id = Item.item_id;

--  Join Orders and ShoppingCart Table
SELECT Orders.order_id, Orders.order_date, ShoppingCart.item_id, ShoppingCart.quantity
FROM Orders
JOIN ShoppingCart ON Orders.order_id = ShoppingCart.order_id;

-- store procedure
```

```
DELIMITER //
CREATE PROCEDURE GetCustomerOrders(IN cust_id INT)
BEGIN
   SELECT Orders.order_id, Orders.order_date, Item.item_name, ShoppingCart.quantity
   FROM Orders
   JOIN ShoppingCart ON Orders.order_id = ShoppingCart.order_id
   JOIN Item ON ShoppingCart.item_id = Item.item_id
   WHERE Orders.customer_id = cust_id;
END //
DELIMITER ;

call GetCustomerOrders(1);
-- trigger
DELIMITER //
CREATE TRIGGER UpdateShippingStatus
BEFORE UPDATE ON Shipping
FOR EACH ROW
BEGIN
   -- Check if the new shipping date is less than or equal to the current date
   IF NEW.shipping_date <= CURDATE() THEN
      -- Set the status to 'Shipped' if the condition is met
      SET NEW.status = 'Shipped';
   ELSE
      -- Optionally, you can handle the case where the status remains 'In Process'
      SET NEW.status = 'In Process';
   END IF;
END //
DELIMITER ;
-- Update shipping_date to today's date
UPDATE Shipping SET shipping_date = CURDATE() WHERE shipping_id = 2;

-- test trigger
-- Assume shipping_id 2 is still in process; update to today's date

SELECT * FROM Shipping WHERE shipping_id = 2;
```

9.




```
create database employeedb;
use employeedb;
CREATE TABLE EMPLOYEE (
  ssn INT PRIMARY KEY,
  first_name VARCHAR(50),
  middle_initial CHAR(1),
  last_name VARCHAR(50),
  address VARCHAR(100),
  sex CHAR(1),
  birth_date DATE,
  status VARCHAR(20),
  salary DECIMAL(10, 2)
);
CREATE TABLE DEPENDENT (
  dependent_id INT PRIMARY KEY,
  ssn INT,
  name VARCHAR(50),
  sex CHAR(1),
  birth_date DATE,
  relationship VARCHAR(50),
  FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn)
```

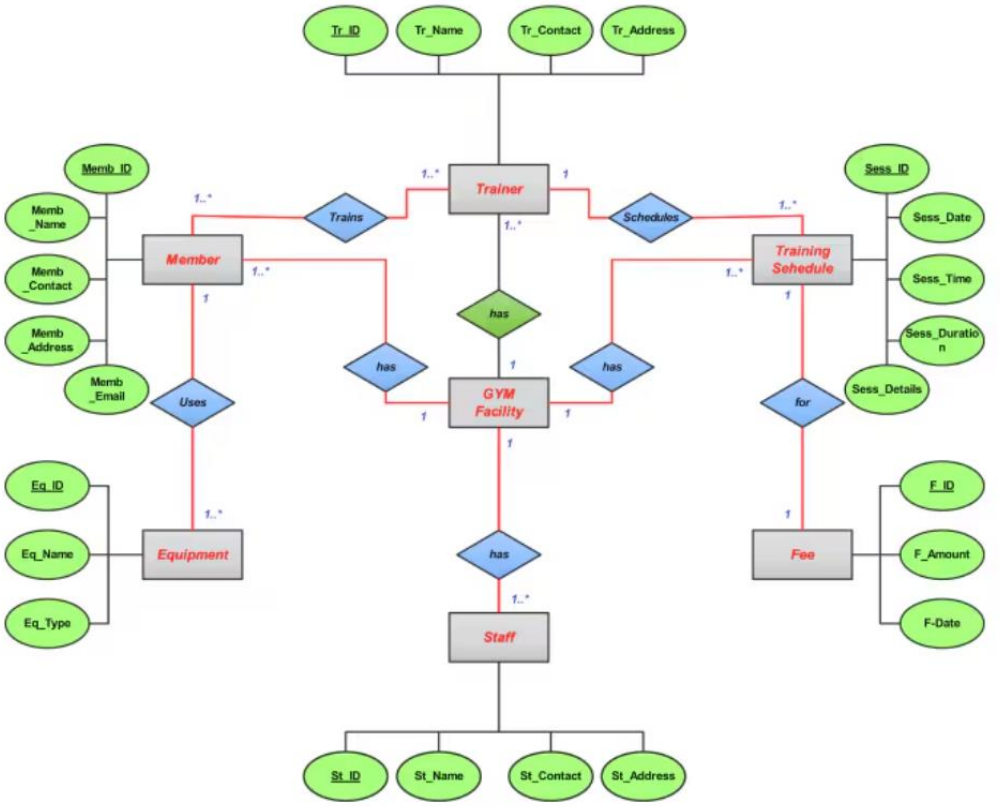
```
);
CREATE TABLE PROJECT (
  project_id INT PRIMARY KEY,
  name VARCHAR(100),
  budget DECIMAL(10, 2),
  location VARCHAR(100)
);
CREATE TABLE DEPARTMENT (
  dept_id INT PRIMARY KEY,
  name VARCHAR(100),
  location VARCHAR(100),
  number_of_employees INT
);
CREATE TABLE WORKS (
  ssn INT,
  project_id INT,
  start_date DATE,
  hours INT,
  PRIMARY KEY (ssn, project_id),
  FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn),
  FOREIGN KEY (project_id) REFERENCES PROJECT(project_id)
);
CREATE TABLE DEPENDENTS_OF (
  ssn INT,
  dependent_id INT,
  PRIMARY KEY (ssn, dependent_id),
  FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn),
  FOREIGN KEY (dependent_id) REFERENCES DEPENDENT(dependent_id)
);
INSERT INTO EMPLOYEE (ssn, first_name, middle_initial, last_name, address, sex, birth_date, status,
salary) VALUES
(1, 'John', 'A', 'Doe', '123 Elm St', 'M', '1980-01-01', 'Full-Time', 60000),
(2, 'Jane', 'B', 'Smith', '456 Oak St', 'F', '1985-05-10', 'Full-Time', 65000),
(3, 'Michael', 'C', 'Johnson', '789 Pine St', 'M', '1990-02-15', 'Part-Time', 45000);
INSERT INTO DEPENDENT (dependent_id, ssn, name, sex, birth_date, relationship) VALUES
(1, 1, 'Alice', 'F', '2005-03-20', 'Daughter'),
(2, 1, 'Bob', 'M', '2008-06-15', 'Son'),
```

```
(3, 2, 'Charlie', 'M', '2010-11-12', 'Son');
INSERT INTO PROJECT (project_id, name, budget, location) VALUES
(1, 'Project Alpha', 100000, 'New York'),
(2, 'Project Beta', 150000, 'Los Angeles'),
(3, 'Project Gamma', 200000, 'San Francisco');
INSERT INTO DEPARTMENT (dept_id, name, location, number_of_employees) VALUES
(1, 'HR', 'New York', 10),
(2, 'Finance', 'Los Angeles', 15),
(3, 'Engineering', 'San Francisco', 20);
INSERT INTO WORKS (ssn, project_id, start_date, hours) VALUES
(1, 1, '2021-01-01', 40),
(2, 2, '2022-03-15', 35),
(3, 3, '2023-07-20', 30);
INSERT INTO DEPENDENTS_OF (ssn, dependent_id) VALUES
(1, 1),
(1, 2),
(2, 3);
create table manages(
ssn int,
dept_id int,
start_date date,
primary key(ssn,dept_id),
foreign key(ssn) references employee(ssn),
foreign key(dept_id) references department(dept_id));

INSERT INTO MANAGES (ssn, dept_id, start_date) VALUES
(1, 1, '2018-05-01'),
(2, 2, '2019-06-10'),
(3, 3, '2020-11-15');
SELECT * FROM EMPLOYEE;
SELECT * FROM DEPENDENT;
SELECT * FROM PROJECT;
SELECT * FROM DEPARTMENT;
SELECT * FROM WORKS;
SELECT * FROM DEPENDENTS_OF;
SELECT * FROM MANAGES;
```

```
-- join
SELECT E.first_name, E.last_name, P.name AS Project_Name, W.start_date, W.hours
FROM EMPLOYEE E
JOIN WORKS W ON E.ssn = W.ssn
JOIN PROJECT P ON W.project_id = P.project_id
WHERE P.name = 'Project X';

-- store procedure
DELIMITER //

CREATE PROCEDURE GetProjectDetails(IN projectName VARCHAR(100))
BEGIN
   SELECT E.first_name, E.last_name, W.start_date, W.hours
   FROM EMPLOYEE E
   JOIN WORKS W ON E.ssn = W.ssn
   JOIN PROJECT P ON W.project_id = P.project_id
   WHERE P.name = projectName;
END //

DELIMITER ;

call GetProjectDetails('Project Alpha');

-- trigger
DELIMITER $$

CREATE TRIGGER UpdateEmployeeCount
AFTER INSERT ON MANAGES
FOR EACH ROW
BEGIN
  UPDATE DEPARTMENT
  SET number_of_employees = number_of_employees + 1
  WHERE dept_id = NEW.dept_id;
END $$

DELIMITER ;
```

```
INSERT INTO EMPLOYEE (ssn, first_name, middle_initial, last_name, address, sex, birth_date, status, salary) VALUES

(4, 'John', 'A', 'Doe', '123 Elm St', 'M', '1980-01-01', 'Full-Time', 60000);


INSERT INTO DEPARTMENT (dept_id, name, location, number_of_employees) VALUES

(4, 'HR', 'New York', 10);

INSERT INTO MANAGES (ssn, dept_id, start_date) VALUES

(4, 4, '2024-02-01');


SELECT * FROM DEPARTMENT;
```

10.




```
-- Create Database
CREATE DATABASE GymManagement;
USE GymManagement;

-- Create Tables
CREATE TABLE Member (
    Mem_ID INT PRIMARY KEY,
    Mem_Name VARCHAR(50),
    Mem_Contact VARCHAR(15),
    Mem_Address VARCHAR(100),
    Mem_Email VARCHAR(50)
);

CREATE TABLE Trainer (
    Tr_ID INT PRIMARY KEY,
    Tr_Name VARCHAR(50),
    Tr_Contact VARCHAR(15),
    Tr_Address VARCHAR(100)
```

```
);

CREATE TABLE Equipment (
    Eq_ID INT PRIMARY KEY,
    Eq_Name VARCHAR(50),
    Eq_Type VARCHAR(50)
);

CREATE TABLE TrainingSchedule (
    Sess_ID INT PRIMARY KEY,
    Sess_Date DATE,
    Sess_Time TIME,
    Sess_Duration INT,
    Sess_Details VARCHAR(200)
);

CREATE TABLE Fee (
    F_ID INT PRIMARY KEY,
    F_Amount DECIMAL(10, 2),
    F_Date DATE
);

CREATE TABLE Staff (
    St_ID INT PRIMARY KEY,
    St_Name VARCHAR(50),
    St_Contact VARCHAR(15),
    St_Address VARCHAR(100)
);

CREATE TABLE GymFacility (
    Gym_ID INT PRIMARY KEY AUTO_INCREMENT
);

-- Create Junction Tables for Many-to-Many Relationships
CREATE TABLE MemberTrainer (
```

```
    Mem_ID INT,
    Tr_ID INT,
    PRIMARY KEY (Mem_ID, Tr_ID),
    FOREIGN KEY (Mem_ID) REFERENCES Member(Mem_ID),
    FOREIGN KEY (Tr_ID) REFERENCES Trainer(Tr_ID)
);

CREATE TABLE MemberEquipment (
    Mem_ID INT,
    Eq_ID INT,
    PRIMARY KEY (Mem_ID, Eq_ID),
    FOREIGN KEY (Mem_ID) REFERENCES Member(Mem_ID),
    FOREIGN KEY (Eq_ID) REFERENCES Equipment(Eq_ID)
);

CREATE TABLE TrainerGymFacility (
    Tr_ID INT,
    Gym_ID INT,
    PRIMARY KEY (Tr_ID, Gym_ID),
    FOREIGN KEY (Tr_ID) REFERENCES Trainer(Tr_ID),
    FOREIGN KEY (Gym_ID) REFERENCES GymFacility(Gym_ID)
);

-- Create relationships for Trainer and Training Schedule
ALTER TABLE TrainingSchedule
ADD COLUMN Tr_ID INT,
ADD FOREIGN KEY (Tr_ID) REFERENCES Trainer(Tr_ID);

-- Create relationship between Fee and Training Schedule
ALTER TABLE Fee
ADD COLUMN Sess_ID INT,
ADD FOREIGN KEY (Sess_ID) REFERENCES TrainingSchedule(Sess_ID);

-- Insert 3 Sample Records for Each Table
INSERT INTO Member VALUES (1, 'Alice', '1234567890', '123 Elm St', 'alice@example.com');
```

```
INSERT INTO Member VALUES (2, 'Bob', '0987654321', '456 Maple St', 'bob@example.com');
INSERT INTO Member VALUES (3, 'Charlie', '1122334455', '789 Oak St', 'charlie@example.com');

INSERT INTO Trainer VALUES (1, 'John Doe', '1234567890', '101 Main St');
INSERT INTO Trainer VALUES (2, 'Jane Smith', '2345678901', '202 Park Ave');
INSERT INTO Trainer VALUES (3, 'Sam Brown', '3456789012', '303 Broadway');

INSERT INTO Equipment VALUES (1, 'Treadmill', 'Cardio');
INSERT INTO Equipment VALUES (2, 'Dumbbell', 'Strength');
INSERT INTO Equipment VALUES (3, 'Yoga Mat', 'Flexibility');

INSERT INTO TrainingSchedule VALUES (1, '2024-10-01', '08:00:00', 60, 'Morning Yoga', 1);
INSERT INTO TrainingSchedule VALUES (2, '2024-10-02', '09:00:00', 45, 'Cardio Blast', 2);
INSERT INTO TrainingSchedule VALUES (3, '2024-10-03', '10:00:00', 30, 'Strength Training', 3);

INSERT INTO Fee VALUES (1, 500.00, '2024-10-01', 1);
INSERT INTO Fee VALUES (2, 300.00, '2024-10-02', 2);
INSERT INTO Fee VALUES (3, 200.00, '2024-10-03', 3);

INSERT INTO Staff VALUES (1, 'Emma White', '1234567890', '456 North St');
INSERT INTO Staff VALUES (2, 'Oliver Green', '2345678901', '789 South St');
INSERT INTO Staff VALUES (3, 'Ava Brown', '3456789012', '123 East St');

INSERT INTO GymFacility VALUES (1);
INSERT INTO GymFacility VALUES (2);
INSERT INTO GymFacility VALUES (3);

-- Insert into Junction Tables
INSERT INTO MemberTrainer VALUES (1, 1);
INSERT INTO MemberTrainer VALUES (2, 2);
INSERT INTO MemberTrainer VALUES (3, 3);

INSERT INTO MemberEquipment VALUES (1, 1);
INSERT INTO MemberEquipment VALUES (2, 2);
INSERT INTO MemberEquipment VALUES (3, 3);
```

```
INSERT INTO TrainerGymFacility VALUES (1, 1);
INSERT INTO TrainerGymFacility VALUES (2, 2);
INSERT INTO TrainerGymFacility VALUES (3, 3);

select* from Member;
select * from trainer;
select*from equipment;
select*from gymfacility;
select*from trainingschedule;
select*from fee;
select*from staff;
select*from membertrainer;
select*from memberequipment;
select*from trainergymfacility;
-- Create Stored Procedure to View All Trainers
DELIMITER //
CREATE PROCEDURE GetAllTrainers()
BEGIN
   SELECT * FROM Trainer;
END //
DELIMITER ;

call getalltrainers();

-- Create Trigger to Prevent Duplicate Email Entries in Member
DELIMITER //
CREATE TRIGGER PreventDuplicateEmail
BEFORE INSERT ON Member
FOR EACH ROW
BEGIN
   DECLARE email_count INT;
   SELECT COUNT(*) INTO email_count FROM Member WHERE Mem_Email = NEW.Mem_Email;
   IF email_count > 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Duplicate Email Entry';
```

```
    END IF;
END //
DELIMITER ;


-- Test Existing Trigger - Trying to Insert a Duplicate Email
INSERT INTO Member (Mem_ID, Mem_Name, Mem_Contact, Mem_Address, Mem_Email)
VALUES (4, 'David', '5566778899', '100 Cedar St', 'alice@example.com');




-- Example Join Query to Check Member and Trainer Relationships
SELECT m.Mem_Name, t.Tr_Name
FROM Member m
JOIN MemberTrainer mt ON m.Mem_ID = mt.Mem_ID
JOIN Trainer t ON mt.Tr_ID = t.Tr_ID;
```

……