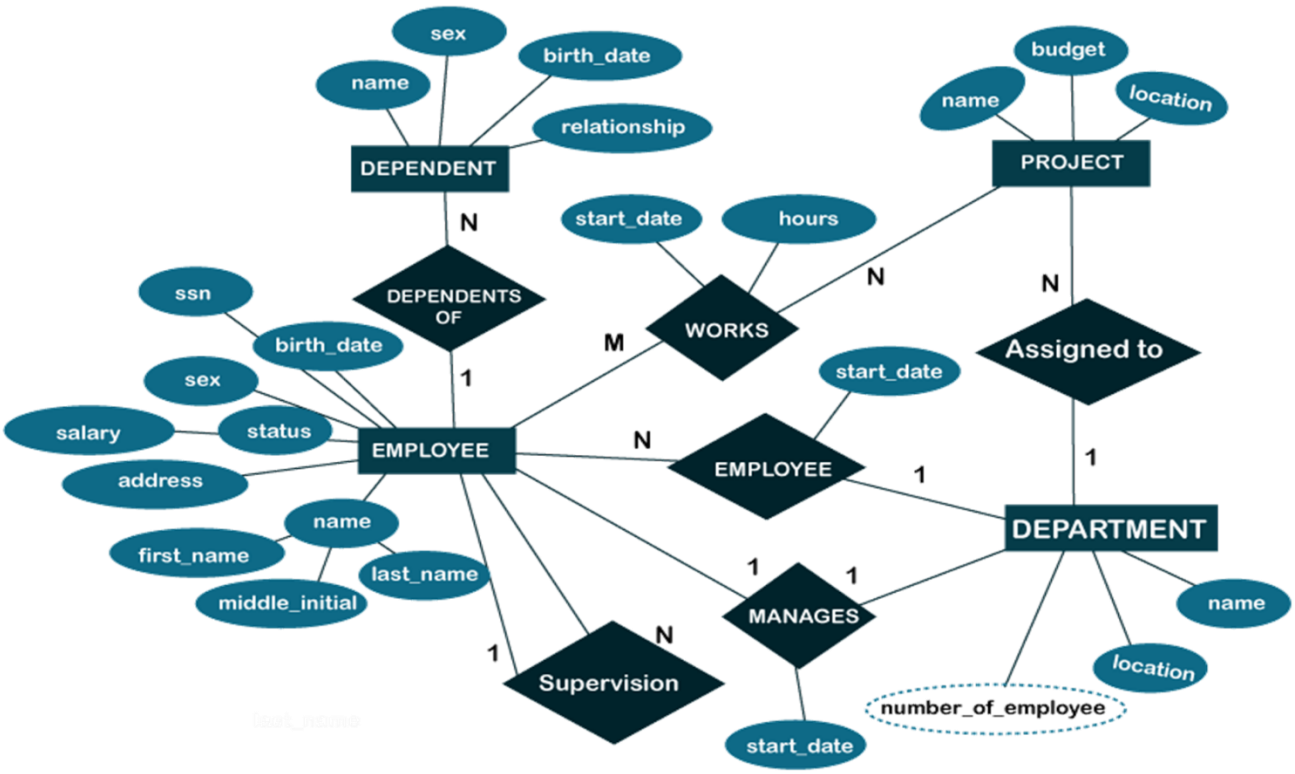
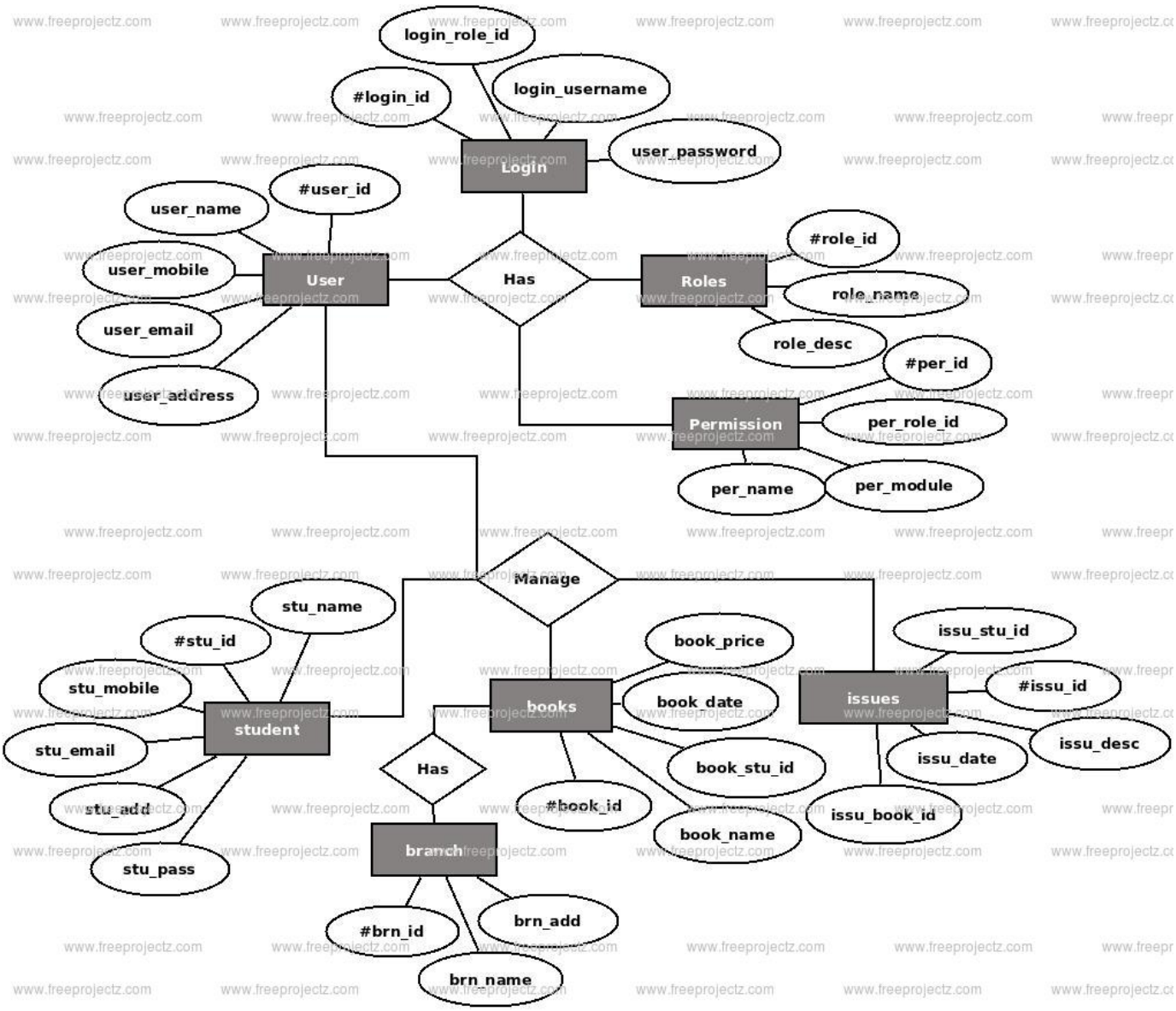
DATABASE MANAGEMENT SYSTEMS LAB

Part A

1. Draw E-R diagram and convert entities and relationships to relation table for a given scenario.
a. Two assignments shall be carried out i.e. consider two different scenarios (e.g. bank, college)
Consider the Company database with following Schema
EMPLOYEE (FNAME, MINIT, LNAME, SSN,
BDATE,ADDRESS,SEX,SALARY, SUPERSSN,
DNO)
DEPARTMENT (DNAME, DNUMBER,MGRSSN,MSRSTARTDATE)
DEPT_LOCATIONS (DNUMBER, DLOCATION)
PROJECT (PNAME, PNUMBER, PLOCATION, DNUM)
WORKS_ON (ESSN, PNO<HOURS)
DEPENDENT (ESSN, DEPENDENT_NAME, SEX, BDATE,
RELATIONSHIP)





ER Diagram For College Management System

2. Perform the following:
a. Viewing all databases, Creating a Database, Viewing all Tables in a Database,
Creating Tables (With and Without Constraints),
Inserting/Updating/Deleting
Records in a Table, Saving (Commit) and Undoing (rollback)

Viewing all databases:

```
SHOW DATABASES;
```

Creating a Database:

```
CREATE DATABASE your_database_name;
CREATE DATABASE Employee;
```

Viewing all tables in a Database:

```
USE your_database_name;
SHOW TABLES;
```

Creating Tables:

Without Constraints:

```
CREATE TABLE employees (
    id INT,
    name VARCHAR(100),
    position VARCHAR(50),
    salary DECIMAL(10, 2)
);
```

With Constraints (Primary Key, Not Null, Unique):

```
CREATE TABLE employees (
    id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE,
    salary DECIMAL(10, 2),
    hire_date DATE
);
```

Inserting Records into a Table:
INSERT INTO employees (id, name, position, salary, hire_date)
VALUES (1, 'John Doe', 'Manager', 65000, '2024-01-15');

Or

INSERT INTO employees
VALUES (2, 'Jane Smith', 'Engineer', 55000, '2023-11-12');

Updating Records in a Table:
UPDATE employees SET salary = 70000 WHERE id = 1;

Deleting Records in a Table:
DELETE FROM employees WHERE id = 1;

Saving (Commit):
COMMIT;

Undoing Changes (Rollback):
ROLLBACK;

3 Perform the following:
a. Altering a Table, Dropping/Truncating/Renaming Tables, Backing up / Restoring a Database.

Altering a Table:
You can modify a table structure by adding, removing, or changing columns.

- **Add a new column:**
  ALTER TABLE employees ADD COLUMN phone_number VARCHAR(15);

Modify a column (changing data type):
ALTER TABLE employees
MODIFY COLUMN salary DECIMAL(12, 2);

Drop (remove) a column:
ALTER TABLE employees
DROP COLUMN phone_number;

Rename a column:
ALTER TABLE employees
CHANGE COLUMN name full_name VARCHAR(100);

Dropping/Truncating/Renaming Tables:

- **Drop a table (completely removes it):**
  DROP TABLE employees;

Truncate a table (removes all data but keeps the structure):
TRUNCATE TABLE employees;

Rename a table:
RENAME TABLE employees TO staff;

Backing up a Database:
You can back up a MySQL database using the mysqldump utility.
mysqldump -u your_username -p your_database_name > backup.sql
his will create a backup of the database in a .sql file named backup.sql.

Restoring a Database:
To restore a database from a backup file:
mysql -u your_username -p your_database_name < backup.sql
This will restore the data from backup.sql back into the specified database.

4. For a given set of relation schemes, create tables and perform the following Simple Queries, Simple Queries with Aggregate functions, Queries with Aggregate functions (group by and having clause).

Sample Relation Schemes:

1. **Employees** (employee_id, name, department, salary, hire_date)
2. **Departments** (department_id, department_name)
3. **Projects** (project_id, project_name, department_id)
4. **Assignments** (employee_id, project_id, hours_worked)

Step 1: Create Tables

```
-- Create Employees Table
CREATE TABLE Employees (
    employee_id INT PRIMARY KEY,
    name VARCHAR(100),
    department VARCHAR(50),
    salary DECIMAL(10, 2),
    hire_date DATE
);

-- Create Departments Table
CREATE TABLE Departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(100)
);

-- Create Projects Table
CREATE TABLE Projects (
    project_id INT PRIMARY KEY,
    project_name VARCHAR(100),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES
Departments(department_id)
);
```

-- Create Assignments Table

```
CREATE TABLE Assignments (
    employee_id INT,
    project_id INT,
    hours_worked INT,
    PRIMARY KEY (employee_id, project_id),
    FOREIGN KEY (employee_id) REFERENCES
Employees(employee_id),
    FOREIGN KEY (project_id) REFERENCES Projects(project_id)
);
```

Step 2: Insert Sample Data

-- Insert data into Employees table

```
INSERT INTO Employees (employee_id, name, department, salary,
hire_date) VALUES
(1, 'Alice', 'IT', 60000, '2021-01-15'),
(2, 'Bob', 'HR', 45000, '2020-07-01'),
(3, 'Charlie', 'Finance', 50000, '2019-10-23'),
(4, 'David', 'IT', 55000, '2018-05-16'),
(5, 'Alice', 'IT', 60000, '2021-01-15'),
(6, 'Bob', 'HR', 45000, '2020-07-01'),
(7, 'Charlie', 'Research', 60000, '2019-10-23'),
(8, 'David', 'Accounts', 57000, '2018-05-16'),
(9, 'Alice', ' Accounts ', 80000, '2021-01-15'),
(10, 'Bob', ' Research ', 55000, '2020-07-01'),
(11, 'Charlie', ' Research ', 40000, '2019-10-23'),
(12, 'David', ' Accounts ', 35000, '2018-05-16'),
(13, 'Alice', ' Accounts ', 20000, '2021-01-15'),
(14, 'Bob', ' Research ', 15000, '2020-07-01'),
(15, 'Charlie', ' Research ', 70000, '2019-10-23'),
(16, 'David', ' Accounts ', 58000, '2018-05-16');
```

-- Insert data into Departments table

```
INSERT INTO Departments (department_id, department_name)
VALUES
(1, 'IT'),
(2, 'HR'),
(3, 'Finance');
```

-- Insert data into Projects table

```
INSERT INTO Projects (project_id, project_name, department_id)
VALUES
(101, 'Website Upgrade', 1),
(102, 'Employee Training', 2),
(103, 'Budget Analysis', 3);
```

-- Insert data into Assignments table

```
INSERT INTO Assignments (employee_id, project_id, hours_worked)
VALUES
(1, 101, 35),
(2, 102, 20),
(3, 103, 40),
(4, 101, 45);
```

Step 3: Perform Simple Queries

Get all employees in the IT department:

```
SELECT * FROM Employees WHERE department = 'IT';
```

Get details of employees hired after 2020:

```
SELECT name, hire_date FROM Employees WHERE hire_date >
'2020-01-01';
```

Get all projects associated with the Finance department:

```
SELECT project_name FROM Projects
WHERE department_id = (SELECT department_id FROM Departments
WHERE department_name = 'Finance');
```

Simple Queries with Aggregate Functions

Find the average salary of all employees:

```
SELECT AVG(salary) AS avg_salary FROM Employees;
```

Find the maximum salary in the IT department:

```
SELECT MAX(salary) AS max_salary FROM Employees
WHERE department = 'IT';
```

Find the total hours worked by all employees:

```
SELECT SUM(hours_worked) AS total_hours FROM Assignments;
```

Count the number of employees in each department:

```
SELECT department, COUNT(*) AS num_employees FROM
Employees GROUP BY department;
```

3. **Queries with Aggregate Functions (GROUP BY and HAVING Clause)**

Find the total hours worked by employees per project:

```
SELECT project_id, SUM(hours_worked) AS total_hours FROM
Assignments GROUP BY project_id;
```

Find the average salary of employees in each department:

SELECT department, AVG(salary) AS avg_salary FROM Employees GROUP BY department;

Find departments where the average salary is greater than 50,000:

SELECT department, AVG(salary) AS avg_salary FROM Employees GROUP BY department HAVING AVG(salary) > 50000;

Get the projects with more than 30 total hours worked:

SELECT project_id, SUM(hours_worked) AS total_hours
FROM Assignments GROUP BY project_id HAVING
SUM(hours_worked) > 30;

5. Execute the fallowing queries
a. How the resulting salaries if every employee working on the „Research“ Departments is given a 10% raise.
b. Find the sum of the salaries of all employees of the „Accounts“ department, as well as the maximum salary, the minimum salary, and the average salary in this department

a. Increase Salaries by 10% for Employees in the "Research" Department
To update the salaries of employees in the "Research" department, you would execute the following query:

UPDATE employees
SET salary = salary * 1.10
WHERE department = 'Research';

This query multiplies the current salary by 1.10 (which adds a 10% raise) for all employees whose department is "Research".

b. Sum, Max, Min, and Average Salaries in the "Accounts" Department

To calculate the sum, maximum, minimum, and average salaries of employees in the "Accounts" department, you can use the following query**:**

```
SELECT
    SUM(salary) AS total_salary,
    MAX(salary) AS max_salary,
    MIN(salary) AS min_salary,
    AVG(salary) AS average_salary
FROM employees
WHERE department = 'Accounts';
```

This query retrieves the total, maximum, minimum, and average salaries for all employees working in the "Accounts" department.

6,7,8,9,10

To execute the queries you've mentioned, we first need to set up the necessary database schema, including tables for employees, departments, and projects. Let's create a simplified version of these tables and then run the queries as specified. Below, I'll outline the steps involved.

Step 1: Create Tables

We will create the following tables:

1. Employees: Contains employee details.

2. Departments: Contains department details.

3. Projects: Contains project details.

4. Employee_Project: A junction table to represent the many-to-many relationship between employees and projects.

Create Departments Table

```
CREATE TABLE Departments (
    DeptID INT PRIMARY KEY,
    DeptName VARCHAR(50)
);
```

-Create Employees Table

```
CREATE TABLE Employees (
    EmpID INT PRIMARY KEY,
    EmpName VARCHAR(50),
    Salary DECIMAL(10, 2),
    DeptID INT,
    DOB DATE,
    FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);
```

-- Create Projects Table

```
CREATE TABLE Projects (
    ProjectID INT PRIMARY KEY,
    ProjectName VARCHAR(50)
);
```

-- Create Employee_Project Table

```
CREATE TABLE Employee_Project (
    EmpID INT,
    ProjectID INT,
    PRIMARY KEY (EmpID, ProjectID),
    FOREIGN KEY (EmpID) REFERENCES Employees(EmpID),
    FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID)
);
```

Step 2: Insert Sample Data

-- Insert Departments

INSERT INTO Departments (DeptID, DeptName) VALUES

(1, 'Research'),

(2, 'Accounts'),

(3, 'HR'),

(4, 'IT'),

(5, 'Sales');

-- Insert Employees

INSERT INTO Employees (EmpID, EmpName, Salary, DeptID, DOB) VALUES

(1, 'Alice', 50000, 1, '1990-01-12'),

(2, 'Bob', 60000, 2, '1999-02-23'),

(3, 'Charlie', 55000, 2, '1990-03-20'),

(4, 'David', 45000, 3, '1999-04-15'),

(5, 'Eve', 40000, 1, '1990-05-16'),

(6, 'Frank', 70000, 5, '1999-06-17'),

(7, 'Grace', 80000, 1, '1998-07-18'),

(8, 'Heidi', 65000, 2, '1997-08-19'),

(9, 'Ivan', 30000, 4, '2000-09-23'),

(10, 'Judy', 90000, 5, '2001-08-2'),

(11, 'Judy', 90000, 5, '2005-09-3'),

(12, 'Judy', 100000, 5, '2004-10-4'),

(13, 'Judy', 200000, 5, '2003-11-5'),

(14, 'Judy', 300000, 5. '2002-12-6');

-- Insert Projects

INSERT INTO Projects (ProjectID, ProjectName) VALUES

(1, 'Project A'),

(2, 'Project B'),

(3, 'Project C');

-- Insert Employee_Project Relationships

INSERT INTO Employee_Project (EmpID, ProjectID) VALUES

(1, 1),

(1, 2),

(2, 1),

(3, 1),

(4, 2),

(5, 3),

(6, 2),

(7, 3),

(8, 1);

Step 3: Execute the Queries

6. Execute the fallowing queries

a. Retrieve the name of each employee Controlled by department number 5 (use EXISTS operator).

b. Retrieve the name of each dept and number of employees working in each department which has at least 2 employees.

a.

```
SELECT e.EmpName
FROM employees e
WHERE EXISTS (
SELECT 1
FROM departments d
WHERE d.DeptID = e.DeptID
AND d.DeptID = 5
);
```

b.

```
SELECT d.DeptName, COUNT(e.EmpID) AS employee_count
FROM departments d
JOIN employees e ON d.DeptID = e.DeptID
GROUP BY d.DeptName
HAVING COUNT(e.EmpID) >= 2;
```

7. Execute the fallowing queries

a. For each project, retrieve the project number, the project name, and the number of employee who work on that project.(use GROUP BY)

b. Retrieve the name of employees who born in the year 1990‟s

a. Retrieve project number, project name, and number of employees working on each project.

SELECT p.ProjectID,

p.ProjectName, COUNT(ep.EmpID) AS NumberOfEmployees

FROM Projects p

LEFT JOIN Employee_Project ep ON p.ProjectID = ep.ProjectID

GROUP BY p.ProjectID, p.ProjectName;

b. Retrieve names of employees born in the 1990s (assuming a date of birth field).

Note: We'll need to assume there’s a `DOB` (date of birth) field in the `Employees` table.

SELECT EmpName

FROM Employees

WHERE YEAR(DOB) BETWEEN 1990 AND 1999;

8. For each department that has more than five employees, retrieve the department number and number of employees who are making salary more than 40000.

Retrieve department number and number of employees making more than 40,000.

```
SELECT DeptID, COUNT(EmpID) AS NumberOfEmployees

FROM Employees

WHERE Salary > 40000

GROUP BY DeptID

HAVING COUNT(EmpID) > 5;
```

9. For each project on which more than two employees work, retrieve the project number, project name and the number of employees who work on that project.

For projects with more than two employees.

```
SELECT p.ProjectID, p.ProjectName, COUNT(EmpID) AS NumberOfEmployees

FROM Projects p

JOIN Employee_Project ep ON p.ProjectID = ep.ProjectID

GROUP BY ProjectID, ProjectName

HAVING COUNT(EmpID) > 2;
```

10. For a given set of relation tables perform the following

a. Creating Views (with and without check option), Dropping views, Selecting from a view

a. Creating a View

```
CREATE VIEW HighSalaryEmployees AS

SELECT EmpName, Salary

FROM Employees

WHERE Salary > 60000;
```

b. Dropping a View

```
DROP VIEW IF EXISTS HighSalaryEmployees;
```

c. Selecting from a View

```
SELECT * FROM HighSalaryEmployees;
```

PART B

Create the following tables with properly specifying Primary keys, Foreign keys and solve the
following queries.
BRANCH (Branchid, Branchname, HOD)
STUDENT (USN, Name, Address, Branchid, sem)
BOOK (Bookid, Bookname, Authorid, Publisher, Branchid)
AUTHOR (Authorid, Authorname, Country, age)
BORROW (USN, Bookid, Borrowed_Date)

1. Perform the following: a. Viewing all databases, Creating a Database, Viewing all Tables in a Database, Creating Tables (With and Without Constraints), Inserting/Updating/Deleting Records in a Table, Saving (Commit) and Undoing (rollback) Execute the following Queries:

   View all databases:
   SHOW DATABASES;

   Create a new database:
   CREATE DATABASE libraryDB;

   View all tables in a database:
   USE libraryDB;

   SHOW TABLES;

```
CREATE TABLE BRANCH (
   Branchid INT PRIMARY KEY,
   Branchname VARCHAR(50),
   HOD VARCHAR(50)
);
```

```
CREATE TABLE STUDENT (
    USN VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(50),
    Address VARCHAR(100),
    Branchid INT,
    Sem INT,
    FOREIGN KEY (Branchid) REFERENCES BRANCH(Branchid)
);


CREATE TABLE AUTHOR (
    Authorid INT PRIMARY KEY,
    Authorname VARCHAR(50),
    Country VARCHAR(50),
    Age INT
);

CREATE TABLE BOOK (
    Bookid INT PRIMARY KEY,
    Bookname VARCHAR(100),
    Authorid INT,
    Publisher VARCHAR(50),
    Branchid INT,
    FOREIGN KEY (Branchid) REFERENCES BRANCH(Branchid),
    FOREIGN KEY (Authorid) REFERENCES AUTHOR(Authorid)
);
```

```
CREATE TABLE BORROW (
    USN VARCHAR(10),
    Bookid INT,
    Borrowed_Date DATE,
    PRIMARY KEY (USN, Bookid),
    FOREIGN KEY (USN) REFERENCES STUDENT(USN),
    FOREIGN KEY (Bookid) REFERENCES BOOK(Bookid));
```

Inserting records:

Insert into BRANCH Table:

```
INSERT INTO BRANCH (Branchid, Branchname, HOD) VALUES
(1, 'MCA', 'Dr. Smith'),
(2, 'MBA', 'Dr. Johnson'),
(3, 'BCA', 'Dr. Williams'),
(4, 'BBA', 'Dr. Brown'),
(5, 'B.Tech', 'Dr. Taylor');
```

Insert into STUDENT Table:

```
INSERT INTO STUDENT (USN, Name, Address, Branchid, Sem)
VALUES
('1RV18MCA01', 'Sam Smith', 'Bangalore', 1, 2),
('1RV18MCA02', 'Sara Connor', 'Mysore', 1, 2),
('1RV18MBA01', 'John Doe', 'Mangalore', 2, 1),
('1RV18BCA01', 'Alice Cooper', 'Hubli', 3, 3),
('1RV18BBA01', 'Tom Hanks', 'Dharwad', 4, 1);
```

Insert into AUTHOR Table:

```
INSERT INTO AUTHOR (Authorid, Authorname, Country, Age)
VALUES
(1, 'Alice Adams', 'USA', 35),
(2, 'Bob Brown', 'UK', 40),
(3, 'Charlie Chaplin', 'Canada', 50),
(4, 'Diana Prince', 'USA', 30),
(5, 'Ethan Hunt', 'Australia', 45);
```

Insert into BOOK Table:

```
INSERT INTO BOOK (Bookid, Bookname, Authorid, Publisher,
Branchid) VALUES
(1, 'Database Management', 1, 'TechBooks', 1),
(2, 'Business Strategy', 2, 'MarketBooks', 2),
(3, 'Programming 101', 3, 'CodePublishers', 3),
(4, 'Marketing Basics', 4, 'AdBooks', 4),
(5, 'Engineering Fundamentals', 5, 'EduBooks', 5);
```

Insert into BORROW Table:

```
INSERT INTO BORROW (USN, Bookid, Borrowed_Date) VALUES
('1RV18MCA01', 1, '2024-01-15'),
('1RV18MCA01', 2, '2024-01-15'),
('1RV18MCA01', 3, '2024-02-20'),
('1RV18MCA02', 2, '2024-01-10'),
('1RV18MBA01', 4, '2024-02-25'),
('1RV18BCA01', 5, '2024-03-05');
```

Insert a record into the STUDENT table:

```
INSERT INTO STUDENT (USN, Name, Address, Branchid, Sem)

VALUES ('1RV18MCA01', 'John Doe', 'Bangalore', 1, 2);
```

Update a record in the STUDENT table:

```
UPDATE STUDENT
SET Name = 'Jane Doe' WHERE USN = '1RV18MCA01';
```

Delete a record from the STUDENT table:

```
DELETE FROM STUDENT WHERE USN = '1RV18MCA01';
```

Commit transaction:

```
COMMIT;
```

Rollback transaction:

```
ROLLBACK;
```

2. a. List the details of Students who are all studying in 2nd Sem MCA.
   b. List the students who are not borrowed any books.

List the details of Students who are studying in 2nd sem MCA:

```
SELECT * FROM STUDENT
WHERE Sem = 2 AND Branchid = (SELECT Branchid FROM
BRANCH WHERE Branchname = 'MCA');
```

b. **List the students who have not borrowed any books:**

```
SELECT * FROM STUDENT
WHERE USN NOT IN (SELECT USN FROM BORROW);
```

3. a. Display the USN, Student name, Branch_name, Book_name, Author_name, Books_Borrowed_ Date of 2nd sem MCA Students who borrowed books.
   b. Display the number of books written by each Author.

Display the USN, Student name, Branch_name, Book_name, Author_name, Books_Borrowed_Date of 2nd sem MCA Students who borrowed books:

```
SELECT S.USN, S.Name, B.Branchname, BK.Bookname, A.Authorname, BO.Borrowed_Date
FROM STUDENT S
JOIN BRANCH B ON S.Branchid = B.Branchid
JOIN BORROW BO ON S.USN = BO.USN
JOIN BOOK BK ON BO.Bookid = BK.Bookid
JOIN AUTHOR A ON BK.Authorid = A.Authorid
WHERE S.Sem = 2 AND B.Branchname = 'MCA';
```

Display the number of books written by each Author:

```
SELECT A.Authorname, COUNT(BK.Bookid) AS NumBooks
FROM AUTHOR A
JOIN BOOK BK ON A.Authorid = BK.Authorid
GROUP BY A.Authorname;
```

4. a. Display the student details who borrowed more than two books.
   b. Display the student details who borrowed books of more than one Author.

   Display the student details who borrowed more than two books:

```
SELECT S.USN, S.Name, COUNT(BO.Bookid) AS NumBooks
FROM STUDENT S
JOIN BORROW BO ON S.USN = BO.USN
GROUP BY S.USN, S.Name
HAVING COUNT(BO.Bookid) > 2;
```

   Display the student details who borrowed books from more than one author:

```
SELECT S.USN, S.Name
FROM STUDENT S
JOIN BORROW BO ON S.USN = BO.USN
JOIN BOOK BK ON BO.Bookid = BK.Bookid
GROUP BY S.USN, S.Name
HAVING COUNT(DISTINCT BK.Authorid) > 1;
```

5. a. Display the Book names in descending order of their names.
   b. List the details of students who borrowed the books which are all published by the same publisher.

Display the Book names in descending order of their names:

```
SELECT Bookname
FROM BOOK
ORDER BY Bookname DESC;
```

List the details of students who borrowed books published by the same publisher:

```
SELECT S.USN, S.Name, BK.Bookname, BK.Publisher
FROM STUDENT S
JOIN BORROW BO ON S.USN = BO.USN
JOIN BOOK BK ON BO.Bookid = BK.Bookid
WHERE BK.Publisher = 'PublisherName';
-- Replace with the actual publisher name
```

Consider the following schema:
STUDENT (USN, name, date_of_birth, branch, mark1, mark2, mark3, total, GPA)

6. **Perform the following:**
   a. Creating Tables (With and Without Constraints), Inserting/Updating/Deleting Records in a Table, Saving (Commit) and Undoing (rollback).

Schema for STUDENT (with GPA):

```
CREATE TABLE STUDENT (
    USN VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(50),
    Date_of_Birth DATE,
    Branch VARCHAR(50),
    Mark1 INT,
    Mark2 INT,
    Mark3 INT,
    Total INT,
    GPA DECIMAL(4, 2)
);
```

Insert into STUDENT Table:

```
INSERT INTO STUDENT (USN, Name, Date_of_Birth, Branch, Mark1, Mark2, Mark3, Total, GPA)
VALUES ('1RV18MCA01', 'John Doe', '2000-04-15', 'MCA', 85, 90, 88, 263, 9.1),
    ('1RV18MCA02', 'Jane Smith', '1999-12-20', 'MCA', 80, 82, 78, 240, 8.5),
    ('1RV18CSE01', 'Alice Johnson', '2001-01-10', 'CSE', 75, 85, 80, 240, 8.0),
    ('1RV18CSE02', 'Bob Brown', '2000-06-25', 'CSE', 88, 89, 90, 267, 9.2),
    ('1RV18ECE01', 'Charlie Davis', '2001-03-05', 'ECE', 70, 75, 80, 225, 7.5),
    ('1RV18ECE02', 'David Wilson', '1999-09-17', 'ECE', 65, 72, 70, 207, 7.0),
    ('1RV18ME01', 'Emma Thomas', '2000-11-01', 'ME', 78, 80, 82, 240, 8.3),
    ('1RV18ME02', 'Frank Martin', '2000-02-12', 'ME', 82, 85, 87, 254, 8.7),
    ('1RV18EEE01', 'Grace Lee', '1999-08-08', 'EEE', 77, 79, 76, 232, 8.1),
('1RV18MCA11', 'SJohn Doe', '2000-04-15', 'MCA', 85, 90, 88, 263, 9.1),
('1RV18MCA12', 'SJohn Doear', '2000-04-15', 'MCA', 85, 90, 88, 263, 9.1),
('1001', 'SJohn Doear', '2000-04-15', 'MCA', 85, 90, 88, 263, 9.1),
    ('1RV18EEE02', 'Hannah White', '2000-07-23', 'EEE', 90, 88, 85, 263, 9.0);
```

7. Execute the following queries:
   a. Find the GPA score of all the students.
   b. Find the students who born on a particular year of birth from the date_of_birth column.

   Find the GPA score of all the students:

```
SELECT USN, Name, GPA FROM STUDENT;
```

   Find the students born in a particular year:

```
SELECT * FROM STUDENT
WHERE YEAR(Date_of_Birth) = 2000;
 -- Replace with the actual year
```

8. a. List the students who are studying in a particular branch of study.
   b. Find the maximum GPA score of the student branch-wise.

   List the students who are studying in a particular branch of study (e.g., MCA):

```
SELECT * FROM STUDENT
WHERE Branch = 'MCA';
 -- Replace 'MCA' with the desired branch name
```

   Find the maximum GPA score of the student branch-wise:

```
SELECT Branch, MAX(GPA) AS MaxGPA
FROM STUDENT
GROUP BY Branch;
```

9. a. List the students who are studying in a particular branch of study. b. Find the maximum GPA score of the student branch-wise. a. Find the students whose name starts with the alphabet “S”. b. Update the column total by adding the columns mark1, mark2, mark3.

Find the students whose name starts with the alphabet “S”:

```
SELECT * FROM STUDENT
WHERE Name LIKE 'S%';
```

Update the column total by adding the columns mark1, mark2, mark3:

```
UPDATE STUDENT
SET Total = Mark1 + Mark2 + Mark3;
```

10. Execute the following queries:
a. Find the students whose name ends with the alphabets “AR”.
b. Delete the student details whose USN is given as 1001.

Find the students whose name ends with the alphabets “AR”:

```
SELECT * FROM STUDENT WHERE Name LIKE '%AR';
```

b. **Delete the student details whose USN is given as 1001:**

```
DELETE FROM STUDENT WHERE USN = '1001';
```