1. Create a student Record

```
create database collegeDB;
use collegeDB;

-- Create student table
CREATE TABLE student (
    student_id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT,
    department VARCHAR(50)
);

-- Insert student data
INSERT INTO student VALUES
(1, 'Rahul Sharma', 20, 'Computer Science'),
(2, 'Anita Das', 21, 'Information Technology');

select * from student;
```

2. Convert a ER diagram into table.

```
-- Create database
CREATE DATABASE HospitalDB;
USE HospitalDB;

-- Patient table
CREATE TABLE Patient (
    patient_id INT PRIMARY KEY,
    name VARCHAR(50),
    age INT,
    gender VARCHAR(10)
);

-- Doctors table
CREATE TABLE Doctors (
    doctor_id INT PRIMARY KEY,
    name VARCHAR(50),
    specialization VARCHAR(50)
);

-- Tests table
CREATE TABLE Tests (
    test_id INT PRIMARY KEY,
    test_name VARCHAR(50),
    cost DECIMAL(8,2)
);

-- TestLog table (relationship table)
CREATE TABLE TestLog (
    log_id INT PRIMARY KEY,
    patient_id INT,
    doctor_id INT,
    test_id INT,
    test_date DATE,

    FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id),
    FOREIGN KEY (test_id) REFERENCES Tests(test_id)
);
```

3. Create database College and create tables student, course, instructor, enrollment and teacher.

```
-- Create and use database
CREATE DATABASE College;
USE College;

-- Student table
CREATE TABLE student (
    sid INT PRIMARY KEY,
    sname VARCHAR(30),
    age INT
);

-- Course table
CREATE TABLE course (
    cid INT PRIMARY KEY,
    cname VARCHAR(30),
    credits INT
);

-- Instructor table
CREATE TABLE instructor (
    iid INT PRIMARY KEY,
    iname VARCHAR(30)
);

-- Teacher table
CREATE TABLE teacher (
    tid INT PRIMARY KEY,
    tname VARCHAR(30),
    dept VARCHAR(30)
);

-- Enrollment table
CREATE TABLE enrollment (
    eid INT PRIMARY KEY,
    sid INT,
    cid INT,

    FOREIGN KEY (sid) REFERENCES student(sid),
    FOREIGN KEY (cid) REFERENCES course(cid)
);
```

4. Create a collegeDB, a student table, specifying realational data types, constraints and indices.

```
-- Create database
CREATE DATABASE collegeDB;
USE collegeDB;

-- Create student table
CREATE TABLE student (
    student_id INT AUTO_INCREMENT PRIMARY KEY,
    roll_no VARCHAR(15) UNIQUE NOT NULL,
    name VARCHAR(50) NOT NULL,
    age INT CHECK (age >= 17),
    email VARCHAR(60) UNIQUE,
    dept VARCHAR(30) NOT NULL,
    admission_date DATE DEFAULT CURRENT_DATE
);

-- Create indexes
CREATE INDEX idx_student_name ON student(name);
CREATE INDEX idx_student_dept ON student(dept);
```

5. Create a database name ComapanyDB and create tables Department, Employees and

insert 4 data of employee.

```
-- Create database
CREATE DATABASE CompanyDB;
USE CompanyDB;

-- Create Department table
CREATE TABLE Department (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(30)
);

-- Create Employees table
CREATE TABLE Employees (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(50),
    dept_id INT,
    salary DECIMAL(10,2),

    FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
);

-- Insert department data
INSERT INTO Department VALUES
(1, 'HR'),
(2, 'IT'),
(3, 'Finance'),
(4, 'Marketing');

-- Insert employee data
INSERT INTO Employees VALUES
(1, 'Amit Kumar', 1, 35000),
(2, 'Neha Singh', 2, 50000),
(3, 'Rahul Das', 3, 42000),
(4, 'Priya Sharma', 4, 38000);
```

6. Employee Management system SQL statement select, insert, update, delete, truncate, drop, alter operations.

```
CREATE DATABASE EMS;
USE EMS;

CREATE TABLE Employee (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(50),
    dept VARCHAR(30),
    salary DECIMAL(10,2)
);

INSERT INTO Employee VALUES
(1, 'Amit', 'HR', 30000),
(2, 'Neha', 'IT', 50000),
(3, 'Rahul', 'Finance', 42000);

SELECT (View Data)
SELECT * FROM Employee;

SELECT emp_name, salary FROM Employee WHERE dept = 'IT';

UPDATE (Modify Data)
UPDATE Employee
SET salary = 55000
WHERE emp_id = 2;
```

```
DELETE (Remove Specific Record)
DELETE FROM Employee
WHERE emp_id = 3;

ALTER (Modify Table Structure)
ALTER TABLE Employee
ADD email VARCHAR(50);

TRUNCATE (Delete All Records)
TRUNCATE TABLE Employee;

DROP (Delete Table)
DROP TABLE Employee;
```

7. Employee Management System where caluse with logical operator.

```
CREATE DATABASE EMS;
USE EMS;

CREATE TABLE Employee (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(50),
    dept VARCHAR(30),
    salary DECIMAL(10,2)
);

INSERT INTO Employee VALUES
(1, 'Amit', 'HR', 30000),
(2, 'Neha', 'IT', 50000),
(3, 'Rahul', 'Finance', 42000);

SELECT * FROM Employee;

WHERE with AND
SELECT * FROM Employee
WHERE dept = 'IT' AND salary > 40000;

WHERE with OR
SELECT * FROM Employee
WHERE dept = 'HR' OR dept = 'Finance';

WHERE with NOT
SELECT * FROM Employee
WHERE NOT dept = 'HR';

WHERE with AND + OR
SELECT emp_name, salary
FROM Employee
WHERE dept = 'IT' AND (salary > 45000 OR emp_id = 1);

UPDATE with WHERE + Logical Operator
UPDATE Employee
SET salary = salary + 5000
WHERE dept = 'IT' AND salary < 50000;

DELETE with WHERE + Logical Operator
DELETE FROM Employee
WHERE dept = 'HR' OR salary < 30000;
```