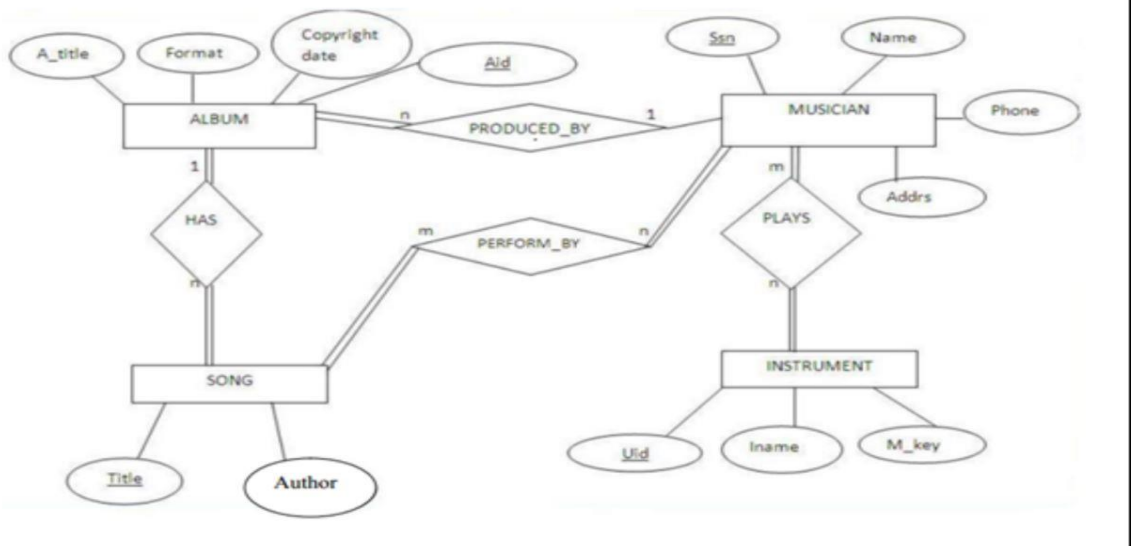
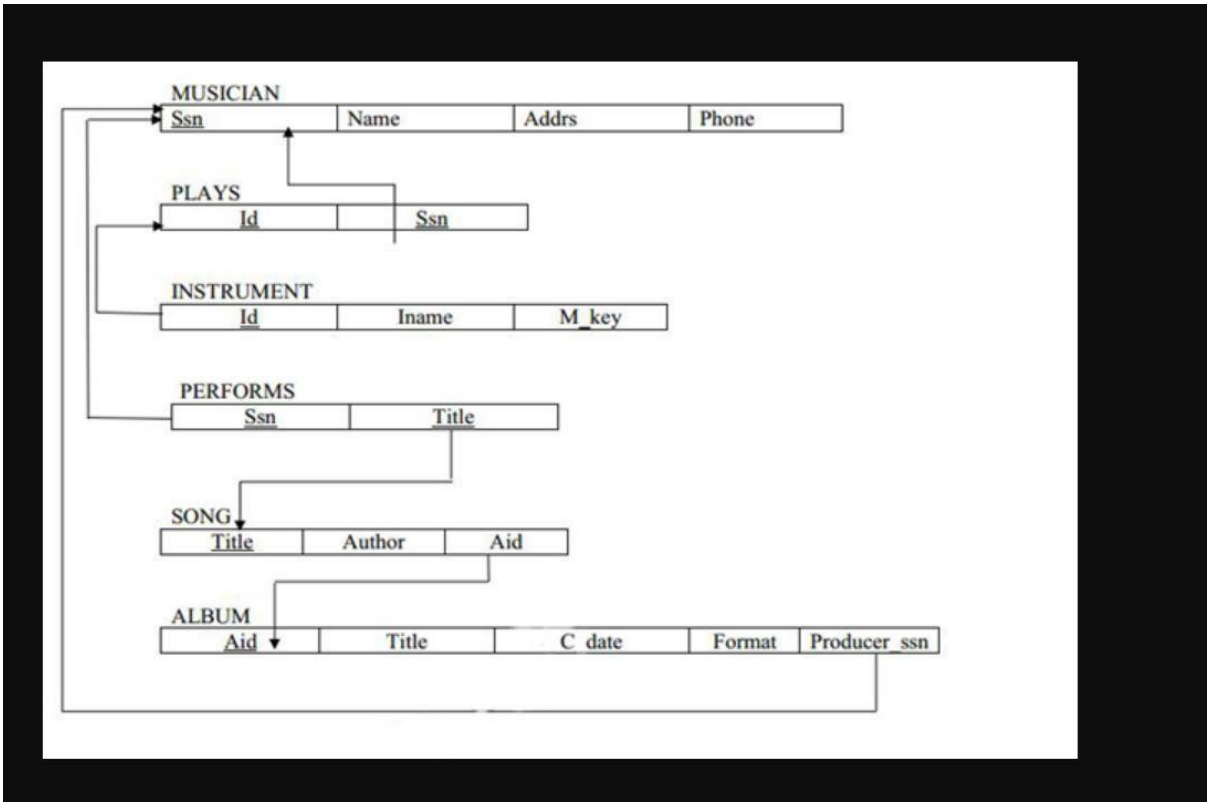
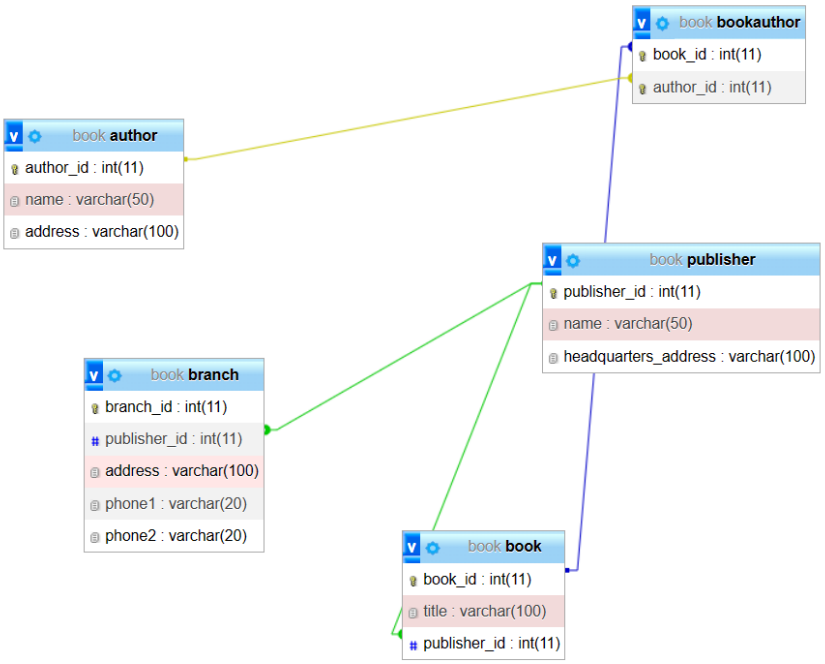
Q1)


A_title
Format
Copyright date
Aid
ALBUM
n
PRODUCED_BY
1
Ssn
Name
MUSICIAN
Phone
1
HAS
n
m
PERFORM_BY
n
m
PLAYS
n
Addrs
SONG
INSTRUMENT
Title
Author
Uid
Iname
M_key


MUSICIAN
Ssn
Name
Addrs
Phone
PLAYS
Id
Ssn
INSTRUMENT
Id
Iname
M_key
PERFORMS
Ssn
Title
SONG
Title
Author
Aid
ALBUM
Aid
Title
C_date
Format
Producer_ssn

Q2)



Q3)

```
-- Create Movie table
CREATE TABLE Movie (
    movie_id INT PRIMARY KEY,
    title VARCHAR(255),
    genre VARCHAR(255),
    year INT,
    budget DECIMAL(10,2),
    Hit_status VARCHAR(255),
    Production_name VARCHAR(255)
);

-- Create Director table
CREATE TABLE Director (
```

```
    did INT PRIMARY KEY,
    Dname VARCHAR(255),
    Gender VARCHAR(10)
);

-- Create Directs table with foreign keys referencing Movie and Director tables
CREATE TABLE Directs (
    did INT,
    movie_id INT,
    No_of_days INT,
    PRIMARY KEY(did, movie_id),
    FOREIGN KEY(did) REFERENCES Director(did),
    FOREIGN KEY(movie_id) REFERENCES Movie(movie_id)
);

-- Insert data into Movie table
INSERT INTO Movie VALUES
(1, 'Titanic', 'Romance', 1997, 200000000, 'Superhit', 'Paramount Pictures'),
(2, 'The Dark Knight', 'Action', 2008, 185000000, 'Superhit', 'Warner Bros'),
(3, 'Inception', 'Thriller', 2010, 160000000, 'Superhit', 'Warner Bros'),
(4, 'Joker', 'Thriller', 2019, 55000000, 'Superhit', 'Warner Bros'),
(5, 'Avengers: Endgame', 'Action', 2019, 356000000, 'Superhit', 'Marvel
Studios');

-- Insert data into Director table
INSERT INTO Director VALUES
(1, 'Christopher Nolan', 'Male'),
(2, 'Todd Phillips', 'Male'),
```

```
(3, 'Russo Brothers', 'Male'),
(4, 'Joss Whedon', 'Male'),
(5, 'Patty Jenkins', 'Female');

-- Insert data into Directs table
INSERT INTO Directs VALUES
(1, 2, 180),
(1, 3, 120),
(2, 4, 120),
(3, 5, 210);

-- 1. Add a new Movie to Movie table
INSERT INTO Movie VALUES
(6, 'Hobbit', 'Fantasy', 2015, 1500000, 'flop', 'Warner Bros');

-- 2. Add a new Movie to Movie table
INSERT INTO Movie VALUES
(7, 'Black Panther', 'Sci fi', 2020, 1000000, 'flop', 'Marvel Studios');

-- 3. Add a new Movie to Movie table
INSERT INTO Movie VALUES
(8, 'Die Hard', 'Action', 2001, 800000, 'Superhit', 'Warner Bros');

-- 4. Movie Hobbit is directed by Christopher Nolan, make appropriate entry in appropriate table.
INSERT INTO Directs VALUES
(1, 6, 90);
```

```
-- 5. Movie Black Panther is directed by James Wan, make appropriate entry in appropriate table.
INSERT INTO Director VALUES
(6, 'James Wan', 'Male');
INSERT INTO Directs VALUES
(6, 7, 150);

-- 6. Movie Die Hard is directed by Martha Coolidge. She is a female Director.
INSERT INTO Director VALUES
(7, 'Martha Coolidge', 'Female');
INSERT INTO Directs VALUES
(7, 8, 90);

-- 7. Update the Hit status for movie Hobbit to Superhit.
UPDATE Movie SET Hit_status = 'Superhit' WHERE title = 'Hobbit';

-- 8. Delete entry for movie Black Panther.
DELETE FROM Movie WHERE title = 'Black Panther';

Q4)
CREATE TABLE worker (
  worker_id INT PRIMARY KEY,
  firstname VARCHAR(50),
  lastname VARCHAR(50),
  salary DECIMAL(10,2),
  joiningdate DATE,
  department VARCHAR(50)
);
```

```
CREATE TABLE bonus (
  worker_ref_id INT,
  bonus_amount DECIMAL(10,2),
  bonus_date DATE
);

INSERT INTO worker VALUES
  (1, 'John', 'Doe', 5000.00, '2022-01-01', 'IT'),
  (2, 'Jane', 'Smith', 6000.00, '2022-01-01', 'Sales'),
  (3, 'Bob', 'Johnson', 5500.00, '2022-01-02', 'IT'),
  (4, 'Alice', 'Brown', 7000.00, '2022-01-03', 'Marketing'),
  (5, 'Mike', 'Davis', 6500.00, '2022-01-03', 'IT');

INSERT INTO bonus VALUES
  (1, 1000.00, '2022-03-01'),
  (2, 1500.00, '2022-04-01'),
  (3, 2000.00, '2022-04-15'),
  (5, 1000.00, '2022-05-01');
```

1. List the workers who got bonus. Display their name, bonus amount, and date on which the bonus was given.

```
SELECT w.firstname, w.lastname, b.bonus_amount, b.bonus_date
FROM worker w
JOIN bonus b ON w.worker_id = b.worker_ref_id;
```

2. List all the workers who did not get a bonus.

```
SELECT *
```

```
FROM worker
WHERE worker_id NOT IN (SELECT worker_ref_id FROM bonus);
```

3. Get a list of all the workers and the bonus amount if any.

```
SELECT w.worker_id, w.firstname, w.lastname, b.bonus_amount, b.bonus_date
FROM worker w
LEFT JOIN bonus b ON w.worker_id = b.worker_ref_id;
```

4. Display the total number of employees working for each department.

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

Q5)

Here is the SQL code to create the employee table and insert 10 random values:

```
CREATE TABLE employee (
    employee_number INT PRIMARY KEY,
    last_name VARCHAR(50),
    first_name VARCHAR(50),
    salary DECIMAL(10, 2),
    dept_id INT,
    job_title VARCHAR(50)
);

INSERT INTO employee VALUES
    (1, 'Smith', 'John', 50000.00, 1, 'Manager'),
```

(2, 'Doe', 'Jane', 60000.00, 1, 'Engineer'),

(3, 'Johnson', 'Tom', 55000.00, 1, 'Engineer'),

(4, 'Garcia', 'Maria', 45000.00, 2, 'Technician'),

(5, 'Lee', 'David', 65000.00, 2, 'Manager'),

(6, 'Chen', 'Amy', 55000.00, 2, 'Technician'),

(7, 'Kim', 'Jin', 70000.00, 3, 'Manager'),

(8, 'Wang', 'Ling', 50000.00, 3, 'Engineer'),

(9, 'Park', 'Min', 60000.00, 3, 'Engineer'),

(10, 'Singh', 'Raj', 45000.00, 3, 'Technician');

Count number of employees in each department. Rename the column in the result as Number_of_Employees.

```
SELECT dept_id, COUNT(*) AS Number_of_Employees
FROM employee
GROUP BY dept_id;
```

Find the name/s of the employee/s who earn minimum salary.

```
SELECT first_name, last_name
FROM employee
WHERE salary = (SELECT MIN(salary) FROM employee);
```

Give 10% salary hike to all the managers.

```
UPDATE employee
SET salary = salary * 1.1
WHERE job_title = 'Manager';
```

Find average salary of Engineering department.

```
SELECT AVG(salary) AS avg_salary
FROM employee
WHERE dept_id = 1;
```

Q6)

```
CREATE TABLE movie (
  movie_id INT PRIMARY KEY,
  title VARCHAR(255),
  genre VARCHAR(255),
  year INT,
  budget DECIMAL(10,2),
  hit_status VARCHAR(255),
  production_name VARCHAR(255)
);

INSERT INTO movie (movie_id, title, genre, year, budget, hit_status, production_name)
VALUES
  (1, 'The Dark Knight', 'Action', 2008, 185000000.00, 'Block Buster', 'Warner Bros'),
  (2, 'Inception', 'Sci-Fi', 2010, 160000000.00, 'Block Buster', 'Warner Bros'),
  (3, 'The Shawshank Redemption', 'Drama', 1994, 25000000.00, 'Superhit', 'Castle Rock Entertainment'),
  (4, 'The Godfather', 'Drama', 1972, 7000000.00, 'Superhit', 'Paramount Pictures'),
  (5, 'Pulp Fiction', 'Crime', 1994, 8000000.00, 'Block Buster', 'Miramax Films');

CREATE TABLE director (
```

```
  DID INT PRIMARY KEY,
  dname VARCHAR(255),
  gender VARCHAR(255)
);

INSERT INTO director (DID, dname, gender)
VALUES
  (1, 'Christopher Nolan', 'Male'),
  (2, 'Frank Darabont', 'Male'),
  (3, 'Francis Ford Coppola', 'Male'),
  (4, 'Quentin Tarantino', 'Male'),
  (5, 'Patty Jenkins', 'Female');

CREATE TABLE directs (
  d_id INT,
  movie_id INT,
  no_of_days INT,
  PRIMARY KEY (d_id, movie_id),
  FOREIGN KEY (d_id) REFERENCES director (DID),
  FOREIGN KEY (movie_id) REFERENCES movie (movie_id)
);

INSERT INTO directs (d_id, movie_id, no_of_days)
VALUES
  (1, 1, 120),
  (1, 2, 100),
  (2, 3, 150),
```

```
  (3, 4, 200),
  (4, 5, 90),
  (5, 2, 80),
  (5, 5, 110);
```

1)

```
SELECT d.dname
FROM movie m
INNER JOIN directs di ON m.movie_id = di.movie_id
INNER JOIN director d ON di.d_id = d.did
WHERE m.hit_status = 'Block Buster';
```

2)

```
SELECT m.title
FROM movie m
INNER JOIN directs di ON m.movie_id = di.movie_id
INNER JOIN director d ON di.d_id = d.did
WHERE d.gender = 'Female';
```

3)

```
SELECT m.production_name, MAX(m.budget) AS max_budget,
AVG(m.budget) AS avg_budget, COUNT(*) AS num_movies
FROM movie m
GROUP BY m.production_name;
```

4)

```
SELECT m.title
FROM movie m
```

```
WHERE m.genre LIKE '%Action%';
```

Q7)

```
CREATE TABLE Movie (
    movie_id INT PRIMARY KEY,
    title VARCHAR(255),
    genre VARCHAR(50),
    year INT,
    budget INT,
    Hit_status VARCHAR(10),
    Production_name VARCHAR(255)
);

CREATE TABLE Director (
    did INT PRIMARY KEY,
    Dname VARCHAR(255),
    Gender VARCHAR(10)
);

CREATE TABLE Directs (
    did INT,
    movie_id INT,
    No_of_days INT,
    PRIMARY KEY (did, movie_id),
    FOREIGN KEY (did) REFERENCES Director(did),
    FOREIGN KEY (movie_id) REFERENCES Movie(movie_id)
);
```

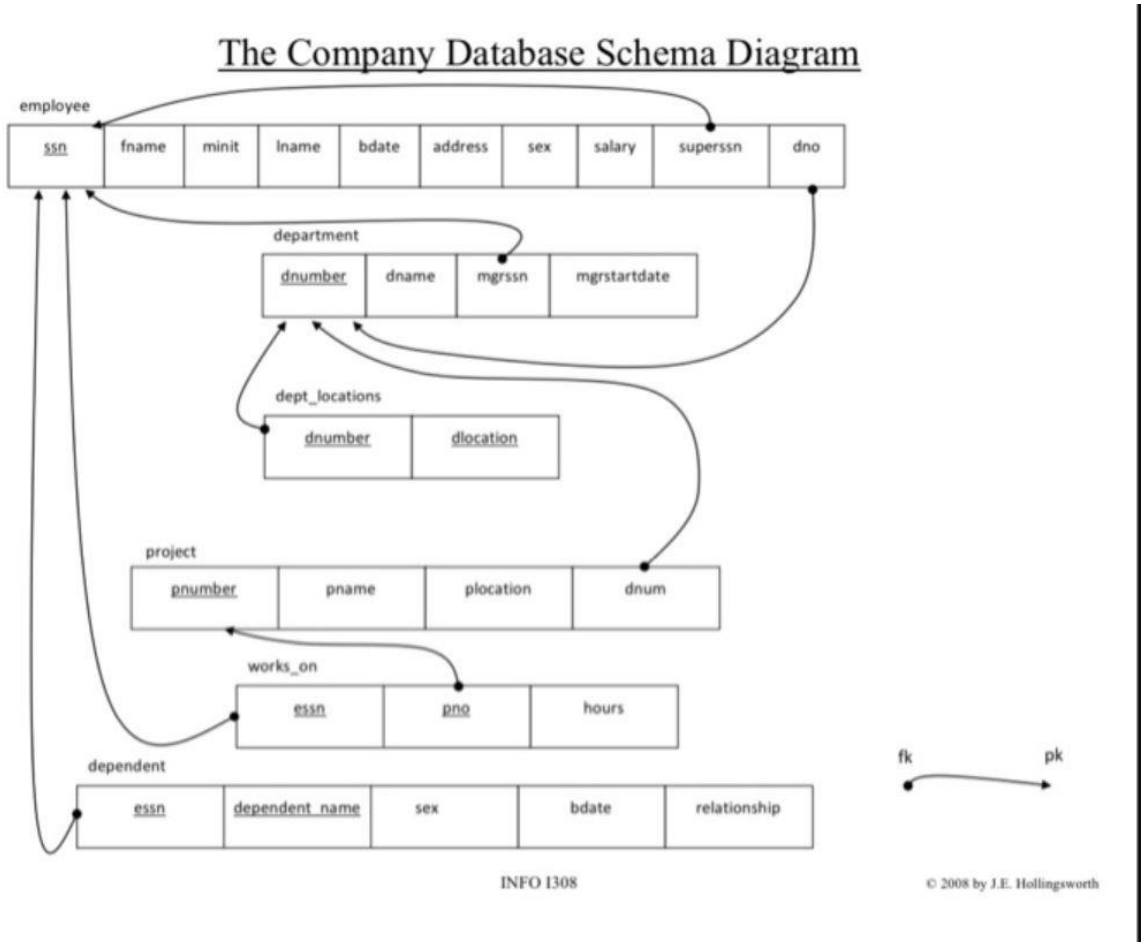
```
INSERT INTO Movie VALUES
(1, 'The Shawshank Redemption', 'Drama', 1994, 25000000, 'Hit', 'Castle Rock Entertainment'),
(2, 'The Godfather', 'Crime', 1972, 7000000, 'Hit', 'Paramount Pictures'),
(3, 'The Dark Knight', 'Action', 2008, 185000000, 'Hit', 'Warner Bros.'),
(4, 'Jurassic Park', 'Adventure', 1993, 63000000, 'Super Hit', 'Universal Pictures'),
(5, 'Avatar', 'Science Fiction', 2009, 237000000, 'Super Hit', '20th Century Fox');

INSERT INTO Director VALUES
(1, 'Frank Darabont', 'Male'),
(2, 'Francis Ford Coppola', 'Male'),
(3, 'Christopher Nolan', 'Male');

INSERT INTO Directs VALUES
(1, 1, 300),
(2, 2, 400),
(3, 3, 500),
(3, 4, 600),
(3, 5, 700);
```

Query)

```
CREATE VIEW Hit_movies AS
SELECT Movie.title, Movie.genre, Movie.Hit_status, Movie.Production_name, Director.Dname
FROM Movie
INNER JOIN Directs ON Movie.movie_id = Directs.movie_id
```

```
INNER JOIN Director ON Directs.did = Director.did
WHERE Movie.Hit_status LIKE '%Hit%';

Q8)
CREATE DATABASE Company;

USE Company;

CREATE TABLE Employee (
    emp_id INT PRIMARY KEY,
    emp_name VARCHAR(255) NOT NULL,
    job_title VARCHAR(255) NOT NULL,
    manager_id INT,
    hire_date DATE NOT NULL,
    salary DECIMAL(10,2) NOT NULL,
    dept_id INT,
    FOREIGN KEY (manager_id) REFERENCES Employee(emp_id),
    FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
);

CREATE TABLE Department (
    dept_id INT PRIMARY KEY,
    dept_name VARCHAR(255) NOT NULL,
    location VARCHAR(255) NOT NULL
);

CREATE TABLE Project (
```

```
    proj_id INT PRIMARY KEY,
    proj_name VARCHAR(255) NOT NULL,
    budget DECIMAL(10,2) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    dept_id INT,
    FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
);

CREATE TABLE Dependent (
    dep_id INT PRIMARY KEY,
    dep_name VARCHAR(255) NOT NULL,
    relation VARCHAR(255) NOT NULL,
    emp_id INT,
    FOREIGN KEY (emp_id) REFERENCES Employee(emp_id)
);
```



Q9)

```
CREATE TABLE Bank (
  bank_name varchar(50) NOT NULL,
  bank_code varchar(10) PRIMARY KEY,
  address varchar(100) NOT NULL,
  PRIMARY KEY (bank_code)
);

CREATE TABLE Branch (
  branch_no int PRIMARY KEY,
  branch_name varchar(50) NOT NULL,
  address varchar(100) NOT NULL,
```

```
  bank_code varchar(10) NOT NULL,
  FOREIGN KEY (bank_code) REFERENCES Bank(bank_code)
);

CREATE TABLE Customer (
  cust_id int PRIMARY KEY,
  cust_name varchar(50) NOT NULL,
  phone_no varchar(20) NOT NULL,
  address varchar(100) NOT NULL
);

CREATE TABLE Account (
  account_no int PRIMARY KEY,
  acc_type varchar(20) NOT NULL,
  balance decimal(10,2) NOT NULL,
  cust_id int NOT NULL,
  branch_no int NOT NULL,
  FOREIGN KEY (cust_id) REFERENCES Customer(cust_id),
  FOREIGN KEY (branch_no) REFERENCES Branch(branch_no)
);

CREATE TABLE Loan (
  loan_id int PRIMARY KEY,
  loan_type varchar(20) NOT NULL,
  amount decimal(10,2) NOT NULL,
  cust_id int NOT NULL,
  branch_no int NOT NULL,
```

```
  FOREIGN KEY (cust_id) REFERENCES Customer(cust_id),
  FOREIGN KEY (branch_no) REFERENCES Branch(branch_no)
);
```

@

Bank(bank_code(PK), bank_name, address)

Branch(branch_no(PK), branch_name, address, bank_code(FK references Bank))

Customer(cust_id(PK), cust_name, phone_no, address)

Account(account_no(PK), acc_type, balance, cust_id(FK references Customer), branch_no(FK references Branch))

Loan(loan_id(PK), loan_type, amount, cust_id(FK references Customer), branch_no(FK references Branch))

Q10)

```
CREATE TABLE Movie (
    movie_id INT PRIMARY KEY,
    title VARCHAR(255),
    genre VARCHAR(50),
    year INT,
    budget INT,
    Hit_status VARCHAR(10),
    Production_name VARCHAR(255)
);
```

```
INSERT INTO Movie VALUES
(1, 'The Shawshank Redemption', 'Drama', 1994, 25000000, 'Hit', 'Castle Rock Entertainment'),
(2, 'The Godfather', 'Crime', 1972, 7000000, 'Hit', 'Paramount Pictures'),
(3, 'The Dark Knight', 'Action', 2008, 185000000, 'Hit', 'Warner Bros.'),
(4, 'Jurassic Park', 'Adventure', 1993, 63000000, 'Super Hit', 'Universal Pictures'),
(5, 'Avatar', 'Science Fiction', 2009, 237000000, 'Super Hit', '20th Century Fox');

CREATE TABLE Actor (
    A_id INT PRIMARY KEY,
    A_name VARCHAR(255),
    Gender VARCHAR(10)
);

CREATE TABLE Acts (
    movie_id INT,
    A_id INT,
    role VARCHAR(50),
    fees INT,
    PRIMARY KEY (movie_id, A_id),
    FOREIGN KEY (movie_id) REFERENCES Movie(movie_id),
    FOREIGN KEY (A_id) REFERENCES Actor(A_id)
);
```

To list the movie names and female leads of those movies:

sql

Copy code

```
SELECT Movie.title, Actor.A_name
```

```
FROM Movie
JOIN Acts ON Movie.movie_id = Acts.movie_id
JOIN Actor ON Acts.A_id = Actor.A_id
WHERE Actor.Gender = 'Female' AND Acts.role = 'Lead';
```

This query joins the Movie, Acts, and Actor tables, filters for rows where the actor's gender is female and role is lead, and selects the movie title and actor name.

To find the actor name/s who charged the highest fees and the movie/s they acted in:

sql

Copy code

```
SELECT Actor.A_name, Movie.title, Acts.fees
FROM Acts
JOIN Actor ON Acts.A_id = Actor.A_id
JOIN Movie ON Acts.movie_id = Movie.movie_id
WHERE Acts.fees = (
    SELECT MAX(fees)
    FROM Acts
);
```

This query first selects the maximum fee from the Acts table, and then joins the Actor, Acts, and Movie tables to retrieve the actor name, movie title, and fee where the fee matches the maximum fee.

To list the names of actors who have acted in blockbuster movies:

sql

Copy code

```
SELECT DISTINCT Actor.A_name
FROM Actor
```

JOIN Acts ON Actor.A_id = Acts.A_id

JOIN Movie ON Acts.movie_id = Movie.movie_id

WHERE Movie.Hit_status LIKE '%Hit%';

This query joins the Actor, Acts, and Movie tables, filters for rows where the movie hit status includes the word "Hit", and selects the distinct actor names.