1. **Draw a normalized entity-relationship diagram that corresponds to the specification and business rules in the box below. Follow these five steps while drawing your model on the next page:**

   a. **Draw a single box for each entity: give each an appropriate name.**

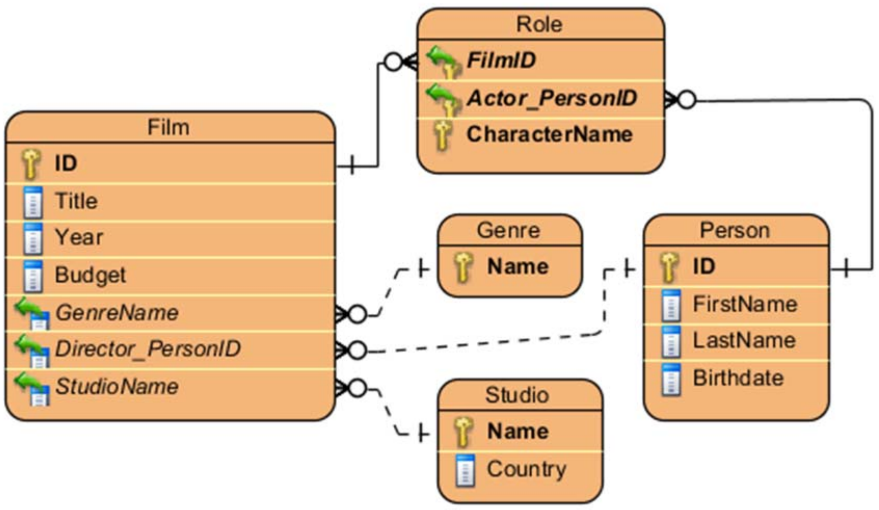
   b. **List the attributes in the box for each entity (data types are <u>not</u> necessary). Indicate the primary key for each entity by writing "PK" next to the appropriate attribute(s).**

   c. **Draw all relationships between the entities in the model. Indicate foreign keys by writing "FK" next to attributes that are foreign keys. For many-to-many relationships, show the associative/intermediate table explicitly; don't use the many-to-many notation.**

   d. **Indicate the cardinality of each relationship (one-to-many, many-to-one, etc.) Any time a cardinality of "many" occurs at either end of a relationship, you do <u>not</u> have to specify whether that cardinality is "zero-or-more" versus "one-or-more." Use crow's-foot notation; if you use any other notation, define it.**

You are hired by a movie theater chain to develop an online ticket-sales and movie-information application. For the first spiral, you must design the data model for a simplified version of the movie-information database, which must adhere to the following:

- Each film has a title, a year, a genre, a director, a studio, a budget, zero or more cast members, and a numeric ID.
- A person can direct zero or more films and can act in zero or more films. (I.e., a person may act or direct or do both.) Each person has a first name, a last name, a birthdate, and a numeric ID.
- A person can play multiple characters in the same film.
- A film's genre must be selected from a predefined list.
- Each studio has a single location, and all films are made on a studio's premises. The website must be able to specify the country in which each film was made.
- Although later spirals of this system will contain movie times and locations, box office figures, and reviews, we are only interested in modeling the data specified in the previous bullets at this time.



2. **Film Cast. List the cast of the 1958 version of the film *Vertigo*, including the names of the actors and their characters. Sort the list by the actors' last names. Assume this film is in the database.** (8 points)

```
SELECT FirstName, LastName, CharacterName FROM Film
INNER JOIN Role ON Role.FilmID = Film.ID
INNER JOIN Person ON Person.ID = Role.Actor_PersonID
WHERE Title = 'Vertigo' AND Year = 1958
ORDER BY LastName, FirstName, CharacterName
```

3. **Youngest Big-Budget Directors. List the names, in order of increasing age, of people who have directed films budgeted over $150 million, and display the average film budget of each of these directors. You may assume the combination of first name and last name are unique.**

```
SELECT FirstName, LastName, AVG(Budget) AS AvgBudget
FROM Person
INNER JOIN Film ON Film.Director_PersonID = Person.ID
GROUP BY FirstName, LastName
HAVING MAX(Budget) >= 150000000
ORDER BY Birthdate DESC
```

4. **Currency Correction. Although the database is intended to display film budgets in US dollars, data has erroneously been loaded in the film studios' local currencies. Write a single query to edit the records of all films made in the UK, converting pounds sterling to dollars. (1 pound = 1.6 dollars. UK film studios are denoted by a Country value of "UK".)**

```
UPDATE Film
SET Budget = Budget * 1.6
FROM Film
INNER JOIN Studio ON Studio.Name = Film.StudioName
WHERE Country = 'UK'
```