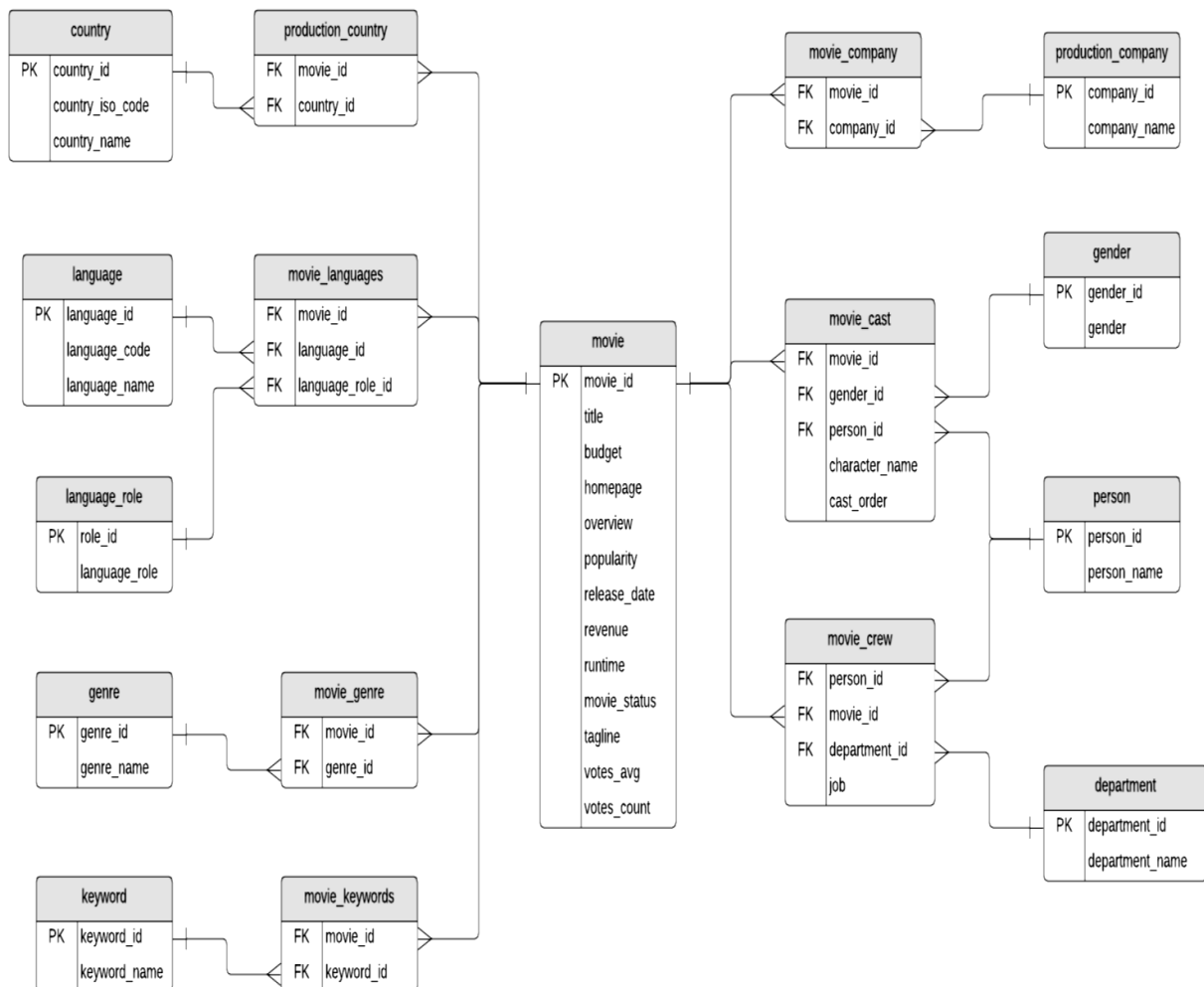


### WORKSHEET - 5 SQL

Refer the following ERD and answer all the questions in this worksheet. You have to write the queries using MYSQL for the required Operation.



### Table Explanations:

- The **movie** table contains information about each movie. There are text descriptions such as title and overview. Some fields are more obvious than others: revenue (the amount of money the movie made), budget (the amount spent on creating the movie). Other fields are calculated based on data used to create the data source: popularity, votes\_avg, and votes\_count. The status indicates if the movie is Released, Rumoured, or in Post-Production.
- The **country** list contains a list of different countries, and the **movie\_country** table contains a record of which countries a movie was filmed in (because some movies are filmed in multiple countries). This is a standard many-to-many table, and you'll find these in a lot of databases.
- The same concept applies to the **production\_company** table. There is a list of production companies and a many-to-many relationship with movies which is captured in the **movie\_company** table.
- The **languages** table has a list of languages, and the **movie\_languages** captures a list of languages in a movie. The difference with this structure is the addition of a **language\_role** table.
- This **language\_role** table contains two records: Original and Spoken. A movie can have an original language (e.g. English), but many Spoken languages. This is captured in the **movie\_languages** table along with a role.
- Genres define which category a movie fits into, such as Comedy or Horror. A movie can have multiple genres, which is why the **movie\_genres** table exists.
- The same concept applies to keywords, but there are a lot more keywords than genres. I'm not sure what qualifies as a keyword, but you can explore the data and take a look. Some examples as "paris", "gunslinger", or "saving the world".
- The cast and crew section of the database is a little more complicated. Actors, actresses, and crew members are all people, playing different roles in a movie. Rather than have separate lists of names for crew and cast, this database contains a table called person, which has each person's name.

## WORKSHEET-5

- The **movie\_cast** table contains records of each person in a movie as a cast member. It has their character name, along with the **cast\_order**, which I believe indicates that lower numbers appear higher on the cast list.
- The **movie\_cast** table also links to the gender table, to indicate the gender of each character. The gender is linked to the **movie\_cast** table rather than the **person** table to cater for characters which may be a different gender than the person, or characters of unknown gender. This means that there is no gender table linked to the person table, but that's because of the sample data.
- The **movie\_crew** table follows a similar concept and stores all crew members for all movies. Each crew member has a job, which is part of a **department** (e.g. Camera).

### QUESTIONS:

1. Write SQL query to show all the data in the Movie table.

Select \* from movie;

2. Write SQL query to show the title of the longest runtime movie.

Select title from movie order by runtime desc limit 1;

3. Write SQL query to show the highest revenue generating movie title.

Select title from movie order by revenue desc limit 1;

4. Write SQL query to show the movie title with maximum value of revenue/budget.

Select title from movie order by budget desc limit 1;

5. Write a SQL query to show the movie title and its cast details like name of the person, gender, character name, cast order.

## WORKSHEET-5

Select title, gender, character\_name, cast\_order, person\_name from movie a inner join movie\_cast b on a.movie\_id=b.movie\_id inner join gender c on c.gender\_id=b.gender\_id inner join person d on d.person\_id= b.person\_id;

- 6. Write a SQL query to show the country name where maximum number of movies has been produced, along with the number of movies produced.**

Select country\_name, count(country\_name) as count from country as a inner join production\_country as b on b.country\_id=a.country\_id group by country\_name order by count desc limit 1;

- 7. Write a SQL query to show all the genre\_id in one column and genre\_name in second column.**

Select \* from genre;

- 8. Write a SQL query to show name of all the languages in one column and number of movies in that particular column in another column.**

Select language\_name,movie\_id,count(language\_name) from movie\_languages as a join language as b on a.language\_id=b.language\_id group by language\_name order by count(language\_name) desc;

- 9. Write a SQL query to show movie name in first column, no. of crew members in second column and number of cast members in third column.**

Select m.title as movie\_name, count(cr.person\_id) as no\_of\_crews, count(ca.person\_id) as no\_of\_cast from movie as m inner join movie\_crew as cr on cr.movie\_id=m.movie\_id inner join movie\_cast ca on ca.person\_id=cr\_person\_id;

- 10. Write a SQL query to list top 10 movies title according to popularity column in decreasing order.**

Select title from movie order by popularity desc limit 10;

**11. Write a SQL query to show the name of the 3rd most revenue generating movie and its revenue.**

Select title from movie order by revenue desc offset 3 limit 1;

**12. Write a SQL query to show the names of all the movies which have “rumoured” movie status.**

Select title from movie where movie\_status like ‘rumored’;

**13. Write a SQL query to show the name of the “United States of America” produced movie which generated maximum revenue.**

Select title, revenue from movie a inner join production\_country b on b.movie\_id = a.movie\_id inner join country c on c.country\_id = b. country\_id where country\_name= ‘United State of America’;

**14. Write a SQL query to print the movie\_id in one column and name of the production company in the second column for all the movies.**

Select m.movie\_id, pc.company\_name from movie m inner join movie\_company mc on mc.movie\_id = m.movie\_id inner join production\_company pc on pc.company\_id =mc.company\_id;

**15. Write a SQL query to show the title of top 20 movies arranged in decreasing order of their budget.**

Select title from movie order by budget desc limit 20;