# 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.**

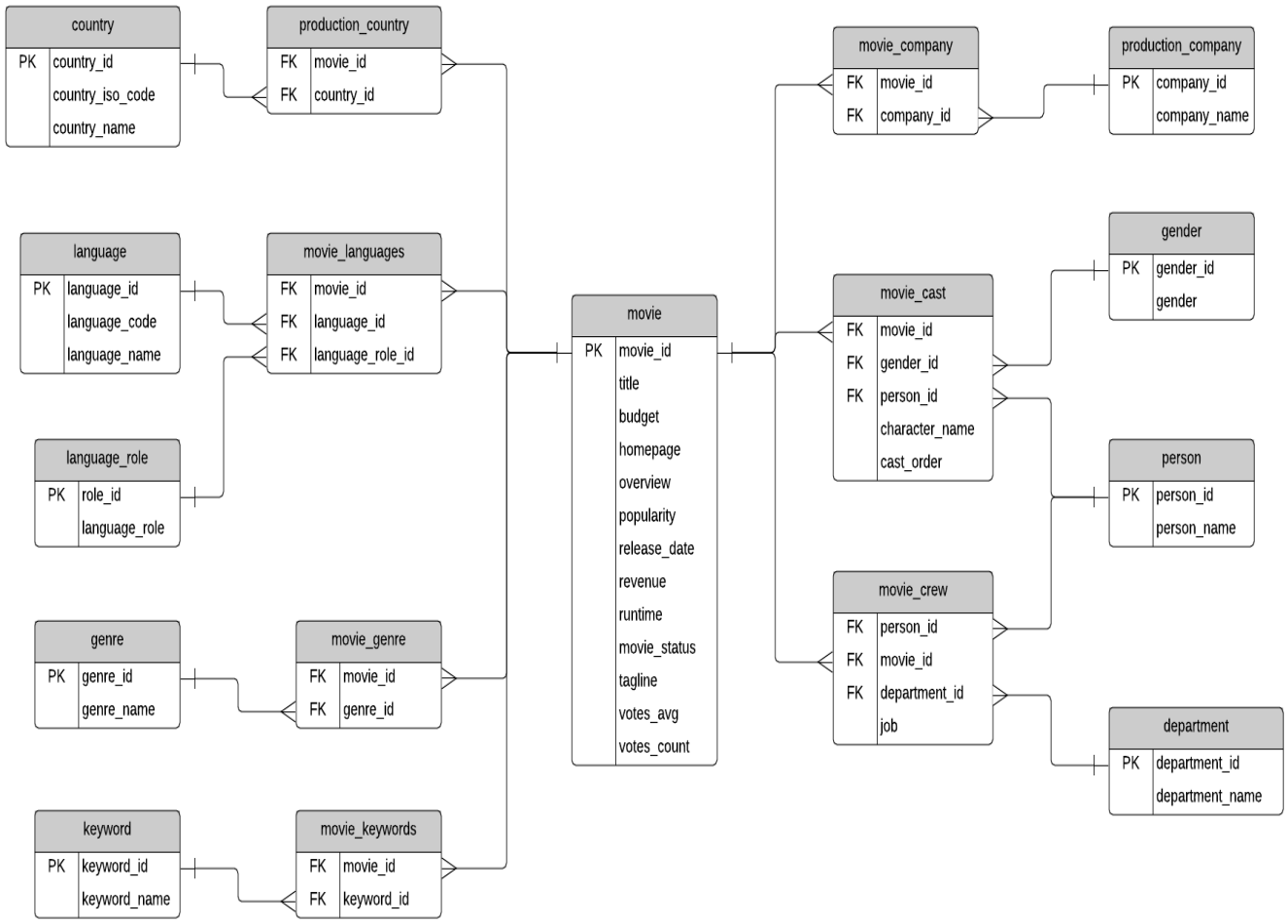


**Table Explanations:**

- 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.
- 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.

   Ans: select*from movie;
2. Write SQL query to show the title of the longest runtime movie.
   Ans: select title from movie order by runtime desc limit 1;
3. Write SQL query to show the highest revenue generating movie title.
   Ans: 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.
   Ans: select title, budget 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.
   Ans: select movie.title, movie_cast.person_id, gender.gender, movie_cast.character_name, movie_cast.cast_order from movie join movie_cast on movie.movie_id=movie_cast.movie_id join gender on movie_cast.gender_id=gender.gender_id where gender.gender_id=2limit3\G
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.
   Ans: select country_name, count(country_name) as no_of_movies_produced from movie as a join production_country as c on a.movie_id=c.movie_id join country on c.country_id=country.country_id group by country_name order by count(country_name)desc;
7. Write a SQL query to show all the genre_id in one column and genre_name in second column.
   Ans: 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.
   Ans: select language_name,movie_id,count(language_name) from movie_languages as m join language as l on m.language_id=l.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.
   Ans: select title,count(cr.person_id),count(ca.person_id) from movie as m join movie_crew as cr on m.movie_id= cr.movie_id full outer join movie_cast as ca on m.movie_id=ca.movie_id group by title;
10. Write a SQL query to list top 10 movies title according to popularity column in decreasing order.
    Ans: select title, revenue from movie order by popularity desc limit 1;
11. Write a SQL query to show the name of the 3rd most revenue generating movie and its revenue.
    Ans: select title from movie order by revenue desc limit 2,1;
12. Write a SQL query to show the names of all the movies which have "rumoured" movie status.
    Ans: select title from movie where movie_status='Rumoured';

FLIP ROBO

13. Write a SQL query to show the name of the “United States of America” produced movie which generated maximum revenue.
    Ans: select title, revenue from movie as m join production_country as pc on m.movie_id=pc.movie_id join country as c on
    Pc.country_id=c.country_id where country_name=’United States of America’ order by revenue desc limit 1;
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.
    Ans: select movie_id, company_name from movie_company as mc join production_company as pc on mc.company_id=pc.company_id;
15. Write a SQL query to show the title of top 20 movies arranged in decreasing order of their budget.
    Ans: select title from movie order by budget limit 20;