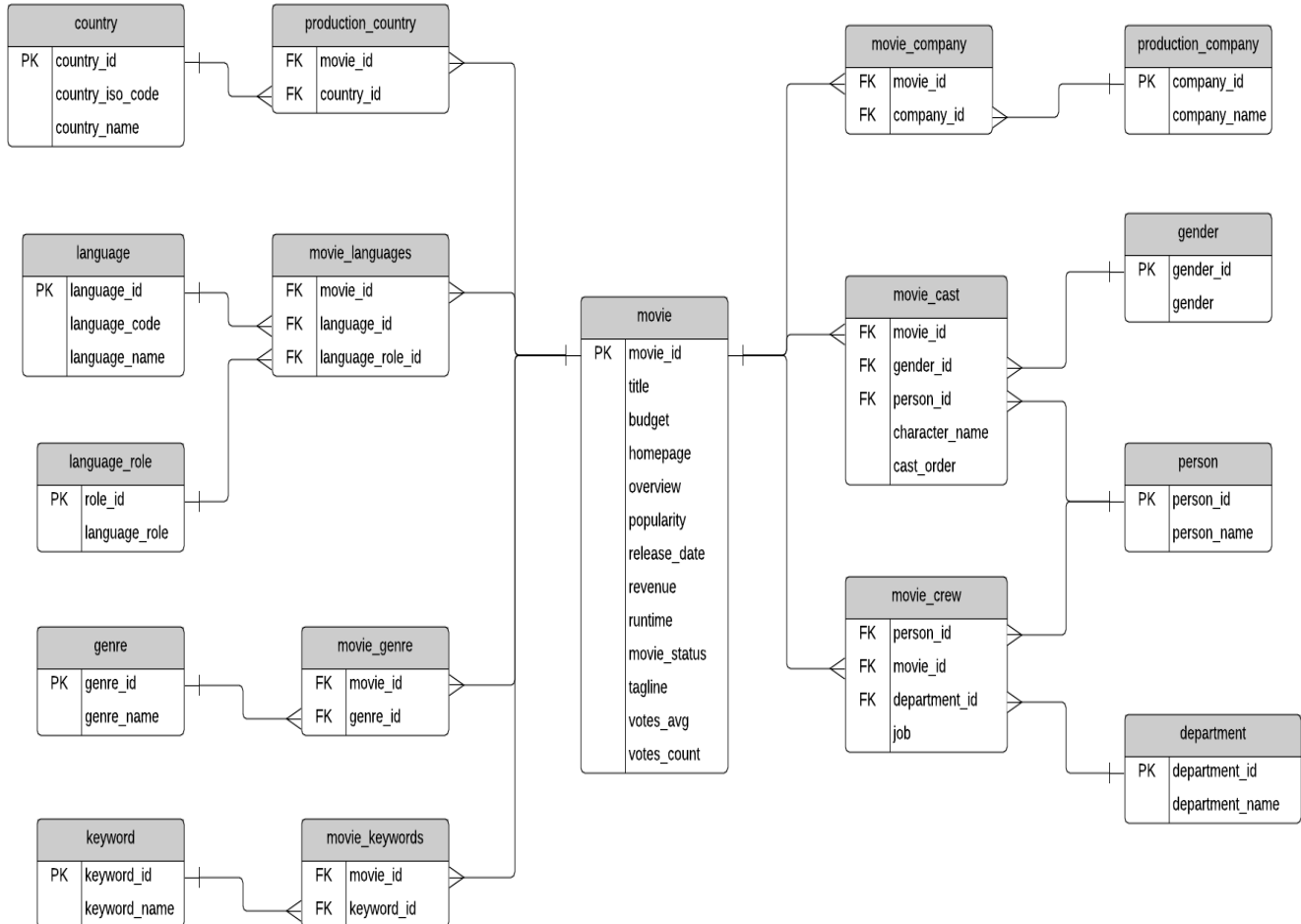


WORKSHEET 5 SQL

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 inPost-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 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, charactername, cast order.

Ans

```
select title, gender, character_name, cast_order, person_name from movie
m INNER JOIN movie_cast mc ON m.movie_id=mc.movie_id INNER JOIN
gender g ON g.gender_id=mc.gender_id INNER JOIN person p ON
p.person_id= mc.person_id;
```

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

Ans

```
select country_name, count(country_name) as count from country as c inner
join production_country as pc on pc.country_id=c.country_id
groupby 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.

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, title from language l INNER JOIN movie_languages  
ml ON l.language_id=ml.language_id INNER JOIN movie m ON  
ml.movie_id=m.movie_id;
```

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 movie_id, title, count(cast.person_id) as cast_count,  
count(crew.person_id) as crew_count  
from movie as m inner join movie_cast as cast on  
cast.movie_id=m.movie_id  
inner join movie_crew as crew on crew.movie_id=m.movie_id  
groupby movie_id;
```

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

Ans

```
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.

Ans

```
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.

Ans

```
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.

Ans

```
select c.country_name, p.movie_id, m.title from country as c JOIN  
production_country as p USING(country_id) JOIN movie as m USING  
(movie_id) 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 m.movie_id, p.company_name from movie_company as m JOIN  
production_company as p USING (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 desc limit 20;
```
