

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.

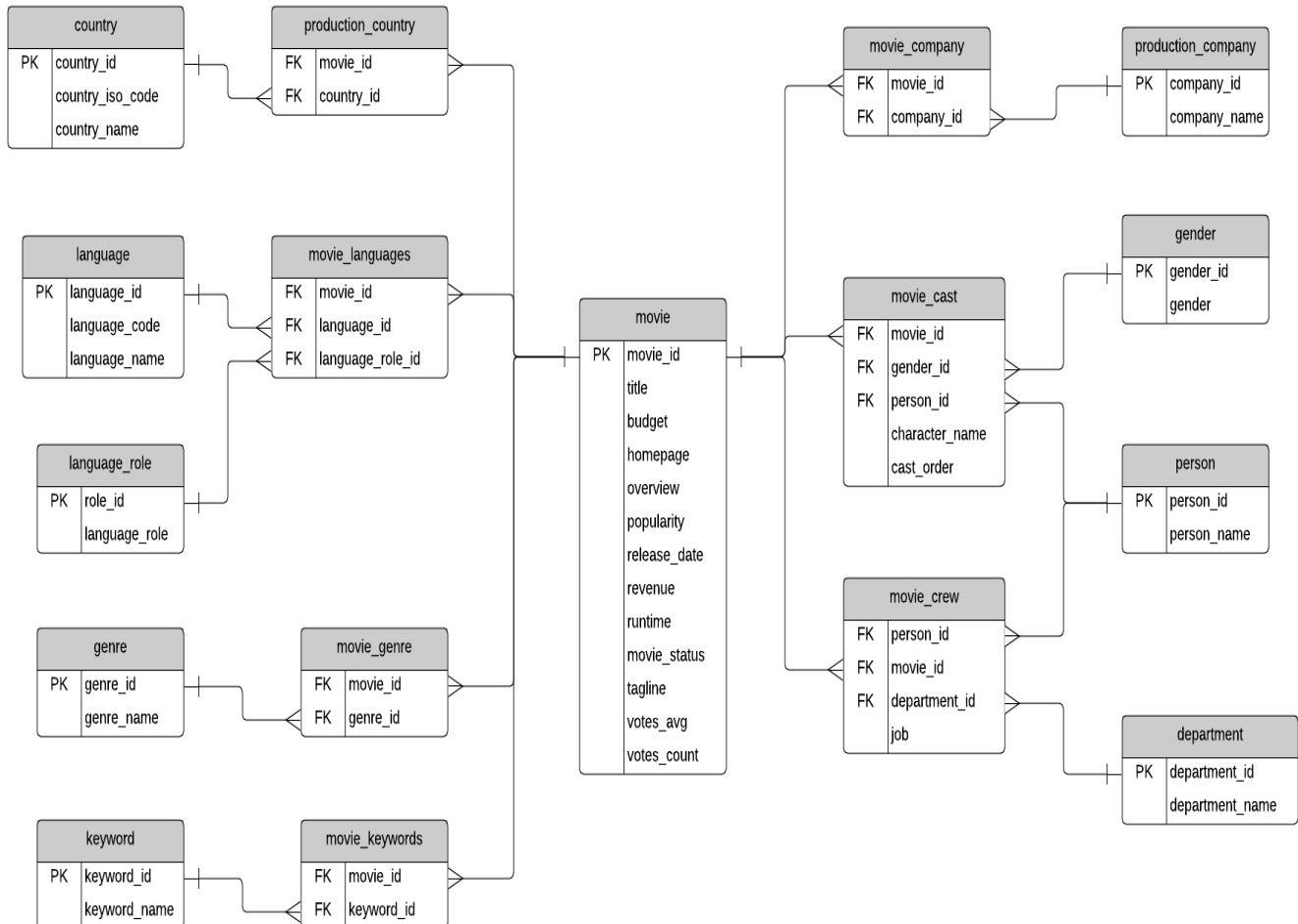


Table Explanations:

- 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.
2. Write SQL query to show the title of the longest runtime movie.
3. Write SQL query to show the highest revenue generating movie title.
4. Write SQL query to show the movie title with maximum value of revenue/budget.
5. Write a SQL query to show the movie title and its cast details like name of the person, gender, character name, cast order.
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.
7. Write a SQL query to show all the genre_id in one column and genre_name in second column.
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.
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.
10. Write a SQL query to list top 10 movies title according to popularity column in decreasing order.
11. Write a SQL query to show the name of the 3rd most revenue generating movie and its revenue.
12. Write a SQL query to show the names of all the movies which have "rumoured" movie status.
13. Write a SQL query to show the name of the "United States of America" produced movie which generated maximum revenue.
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.
15. Write a SQL query to show the title of top 20 movies arranged in decreasing order of their budget.

```
1. SELECT * FROM Movie;
```

```
2. SELECT title FROM Movie  
ORDER BY runtime DESC  
LIMIT 1;
```

```
3. SELECT title FROM Movie  
ORDER BY revenue DESC  
LIMIT 1
```

4. `SELECT title FROM Movie
ORDER BY revenue/budget DESC
LIMIT 1;`

 5. `SELECT m.title, p.name, g.gender, mc.character_name, mc.cast_order
FROM Movie m
JOIN movie_cast mc ON m.id = mc.movie_id
JOIN person p ON mc.person_id = p.id
JOIN gender g ON mc.gender_id = g.id`

 6. `SELECT c.name, COUNT(*) AS movie_count
FROM country c
JOIN movie_country mc ON c.id = mc.country_id
GROUP BY c.id
ORDER BY movie_count DESC
LIMIT 1;`

 7. `SELECT id AS genre_id, name AS genre_name FROM genre;`

 8. `SELECT l.name AS language, COUNT(*) AS movie_count
FROM language l
JOIN movie_languages ml ON l.id = ml.language_id
GROUP BY l.id;`

 9. `SELECT m.title, COUNT(DISTINCT mc.person_id) AS cast_members, COUNT(DISTINCT mcw.person_id)
AS crew_members
FROM Movie m
LEFT JOIN movie_cast mc ON m.id = mc.movie_id
LEFT JOIN movie_crew mcw ON m.id = mcw.movie_id
GROUP BY m.id;`

 10. `SELECT title FROM Movie
ORDER BY popularity DESC
LIMIT 10;`

 11. `SELECT title, revenue FROM Movie
ORDER BY revenue DESC
LIMIT 1 OFFSET 2;`

 12. `SELECT title FROM Movie
WHERE status = 'Rumoured';`
-

13. SELECT title FROM Movie
JOIN movie_country mc ON Movie.id = mc.movie_id
JOIN country c ON mc.country_id = c.id
WHERE c.name = 'United States of America'
ORDER BY revenue DESC
LIMIT 1;
14. SELECT m.id AS movie_id, pc.name AS production_company
FROM Movie m
JOIN movie_company mc ON m.id = mc.movie_id
JOIN production_company pc ON mc.company_id = pc.id;
15. SELECT title
FROM movie
ORDER BY budget DESC
LIMIT 20;
-