

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.

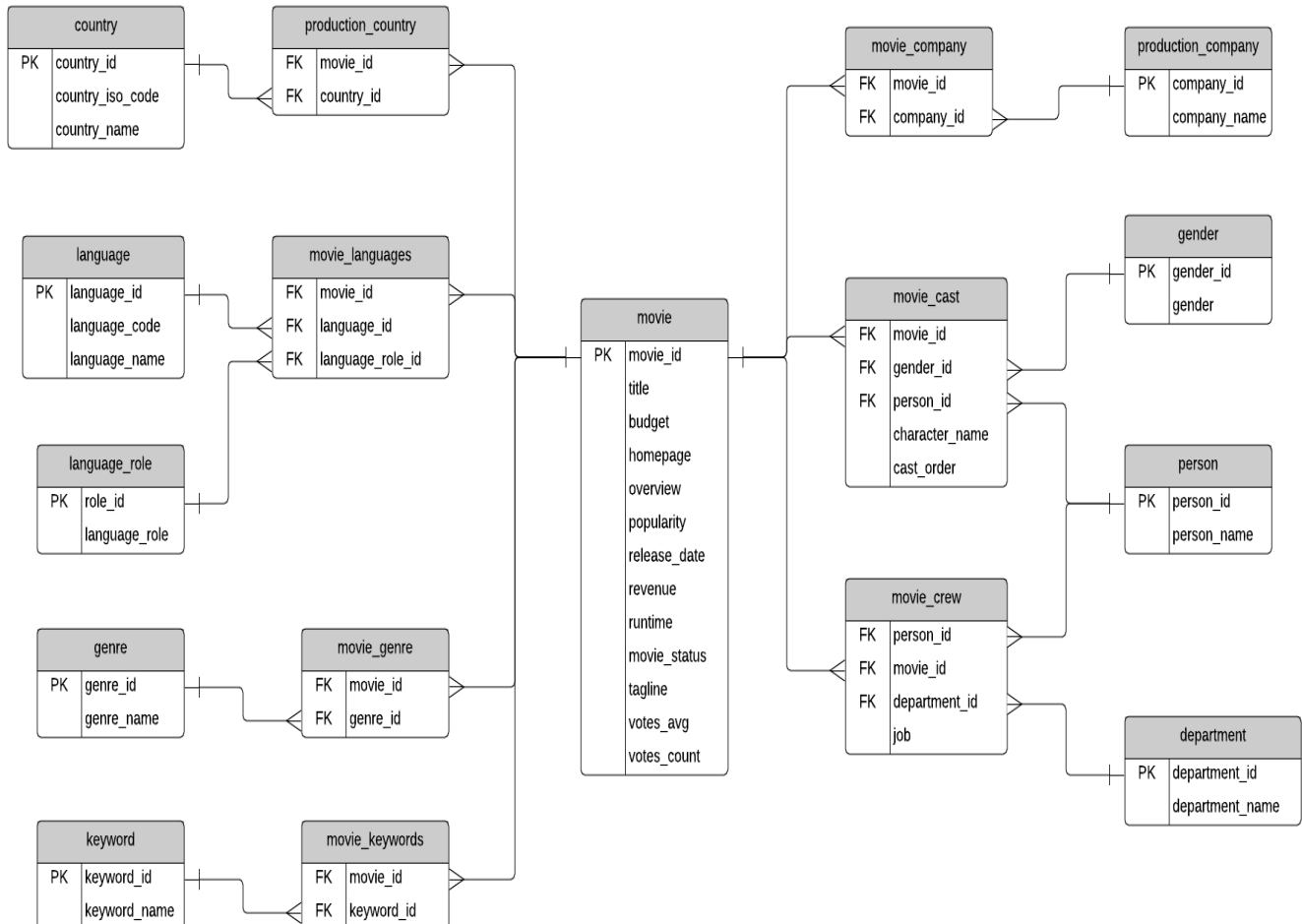


Table Explanations:

- 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.

Answer:

```
SELECT * FROM movie;
```

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

Answer:

```
SELECT title FROM movie WHERE runtime= (SELECT MAX (runtime) FROM movie;
```

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

Answer:

```
SELECT title FROM movie WHERE revenue= (SELECT MAX (revenue) FROM movie;
```

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

Answer:

```
SELECT title FROM movie WHERE revenue= (SELECT MAX (revenue) FROM movie OR budget= (SELECT  
MAX (budget) FROM movie;
```

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

Answer:

```
SELECT movie.title, person.person_name, gender.gender, movie_cast.cast_order  
FROM movie_cast  
INNER JOIN movie  
ON movie_cast.movie_id= movie. movie_id
```

```
INNER JOIN person
ON movie_cast. person_id= person. person _id
INNER JOIN gender
ON movie_cast.gender_id= gender. gender_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.**

Answer:

```
SELECT country_name, count(country_name) AS no_mov_prd
FROM country
INNER JOIN production_country
ON country. country_id= production. country _id
GROUP BY country_name
ORDER BY count(country.country_name) desc limit 1;
```

- 7. Write a SQL query to show all the genre_id in one column and genre_name in second column.**

Answer:

```
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.**

Answer:

```
SELECT language_name, count (language_name) AS no_of_movie
FROM language
INNER JOIN movie_language
ON movie_language. language_id= language.language _id
INNER JOIN movie
ON movie_language.movie_id=movie. movie_id
GROUP BY language_name;
```

- 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.**

Answer:

```
SELECT movie.title, count (movie_crew.job), count(movie_cast.character_name)
FROM movie_crew
INNER JOIN movie
ON movie_crew.movie_id= movie.movie_id
INNER JOIN movie_cast
ON movie_crew.movie_id=movie_cast.movie_id
GROUP BY movie.title;
```

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

Answer:

```
SELECT title, popularity 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.

Answer:

```
SELECT title, revenue 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.

Answer:

```
SELECT title FROM movie WHERE movie_status= “rumoured”;
```

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

Answer:

```
SELECT movie.title, production_company.company_name, max(movie.revenue)
FROM movie_company
INNER JOIN movie
ON movie_company.movie_id= movie.movie_id
INNER JOIN production_company
ON movie_company.company_id=production_company.company_id
WHERE production_company.company_name= “United States of America”
ORDER BY revenue DESC;
```

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.

Answer:

```
SELECT movie.movie_id, production_company.company_name
FROM movie_company.movie_id=movie.movie_id
INNER JOIN production_company
ON movie_company. company_id= production_company. company_id;
```

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

Answer:

```
SELECT title from movie ORDER BY budget DESC LIMIT 10;
```
