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.

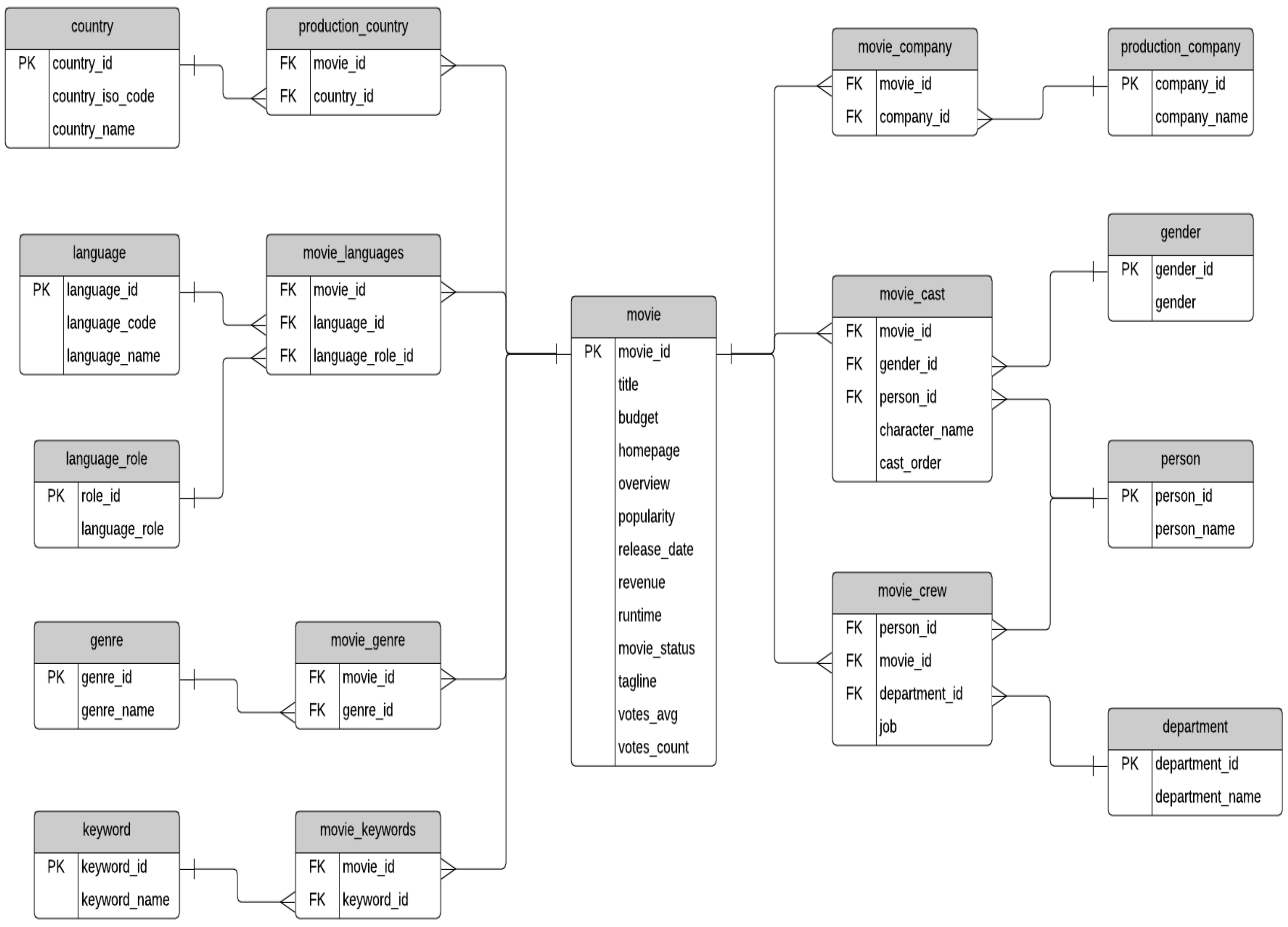

Table Explanations:

- 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.

```
r = cursor.execute("select * from movie")

for row in r:

        print(row)
```

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

```
r = cursor.execute("select title from movie where runtime = (select max(runtime) from movie)")
for row in r:
        print(row)
```

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

```
r = cursor.execute("select title from movie where revenue = (select max(revenue) from movie)")
for row in r:
        print(row)
```

4. Write SQL query to show the movie title with maximum value of revenue/budget.
   Assuming the question says maximum value of revenue or budget hence have chosen budget

```
r = cursor.execute("select title from movie where budget = (select max(budget) from movie)")
for row in r:
        print(row)
```

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

```
r = cursor.execute("select title, person_name, gender, charater_name, cast_order from ( select * from (select
    * from (SELECT * FROM movie JOIN movie_cast using (movie_id)) join person using(person_id))
    join gender using (gender_id))")
for row in r:
        print(row)
```

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.

```
r = cursor.execute(" select country_name, count_CN from (select country_name, count(country_name) as
    count_CN from (select * from country join production_country using (country_id)) group by
    country_name) where count_CN = (select max(count_CN) from (select country_name,
    count(country_name) as count_CN from (select * from country join production_country using
```

(country_id)) group by country_name))")
for row in r:
    print(row)

7. Write a SQL query to show all the genre_id in one column and genre_name in second column.
r = cursor.execute("select * from genre")
for row in r:
    print(row)

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.
r = cursor.execute("select language_name, count(language_name) from (select * from language join movie_language using (language_id)) group by language_name")
for row in r:
    print(row)

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.
r = cursor.execute("select title, count(movie_id), totalcast from (select * from (select title, movie_id, count(movie_id) as totalcast from (select * from movie join movie_cast using (movie_id)) GROUP BY title) join movie_crew using (movie_id)) GROUP BY title")
for row in r:
    print(row)

10. Write a SQL query to list top 10 movies title according to popularity column in decreasing order.
r = cursor.execute("select title, votes_count from movie order by votes_count DESC LIMIT 10")
for row in r:
    print(row)

11. Write a SQL query to show the name of the 3rd most revenue generating movie and its revenue.
r = cursor.execute("select title, revenue from movie order by revenue DESC LIMIT 2,2")
for row in r:
    print(row)

12. Write a SQL query to show the names of all the movies which have "rumoured" movie status.
r = cursor.execute("select title, movie_status from movie where movie_status = 'rumoured'")
for row in r:
    print(row)

13. Write a SQL query to show the name of the "United States of America" produced movie which generated maximum revenue.
r = cursor.execute("select title, country_name, max(revenue) from (select * from movie join (select * from country join production_country using (country_id)) using (movie_id)) where country_name = 'United States of America'")
for row in r:
    print(row)

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.

```
r = cursor.execute("select title, company_name from (select * from movie join (select * from movie_company
    join production_company using (company_id)) using (movie_id))")
for row in r:
        print(row)
```

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

```
r = cursor.execute("select title, budget from movie order by budget DESC LIMIT 20")
for row in r:
        print(row)
```