# 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.**

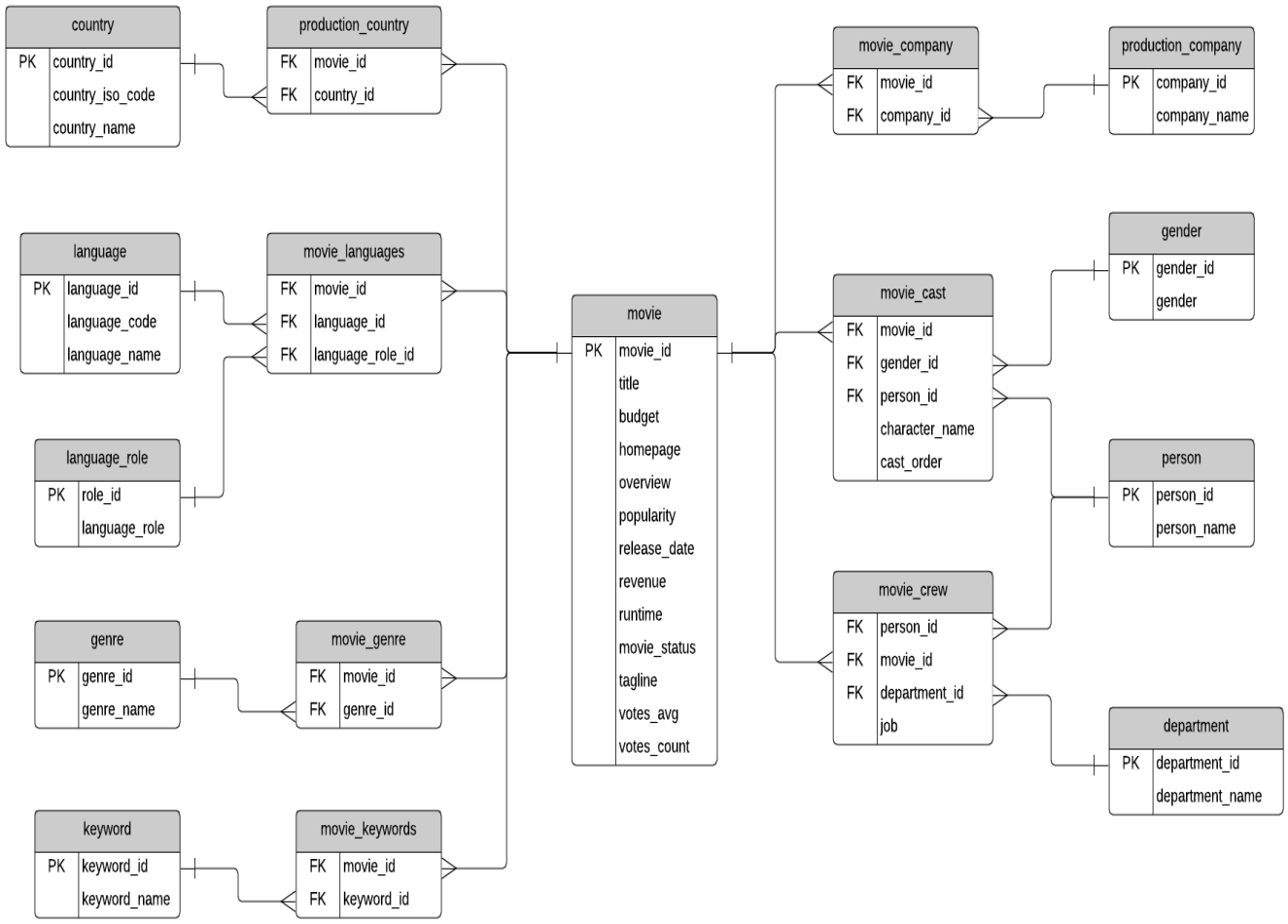


**Table Explanations:**

- 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.

```
CREATE TABLE movie (
movie_id varchar(500) NOT NULL,
title varchar(500) NOT NULL,
budget int NOT NULL,
homepage varchar(500) NOT NULL,
popularity varchar(500) NOT NULL,
release_date DATE NOT NULL,
revenue int NOT NULL,
runtime TIME NOT NULL,
movie_status varchar(500) NOT NULL,
tagline varchar(500) NOT NULL,
votes_avg float NOT NULL,
votes_count varchar(500) NOT NULL,
primary key (movie_id));
SELECT * from movie;
```

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

```
SELECT title from movie
where runtime =  (SELECT max(runtime) from movie);
```

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

```
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.

```
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, character

name, cast order.

```
SELECT
movie.title,person.person_name,gender.gender,movie_cast.character_name,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, along with the number of movies produced.

```
SELECT (country.country_name),count(country.country_name) AS NO_OF_MOVIES
from production_country
INNER JOIN country
ON production_country.country_id = country.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.

```
CREATE TABLE genre (
genre_id int NOT NULL,
genre_name varchar(500) NOT NULL,
PRIMARY KEY (genre_id));

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.

```
SELECT language.language_name,count(language.language_name) AS
NO_OF_MOVIES
from movie_languages
INNER JOIN movie
ON movie_languages.movie_id = movie.movie_id
INNER JOIN language
ON movie_languages.language_id = language.language_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.

```
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.

```
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.

```
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.

```
select * 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.

```
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.

```
select movie.movie_id,production_company.company_name
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
```

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

```
SELECT title from movie
order by budget desc
limit 10;
```