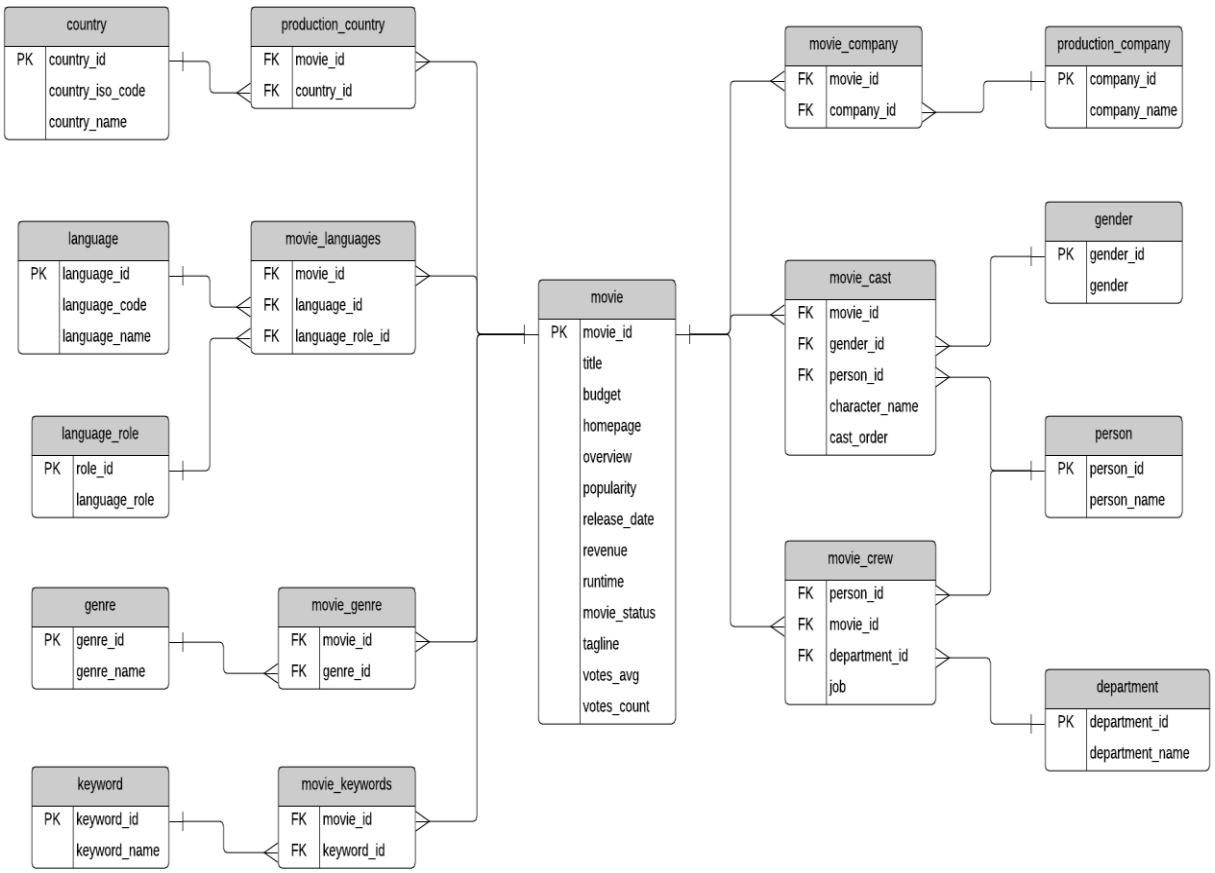
WORKSHEET 5 SQL



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.

ANS: SELECT * FROM MOVIE;

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

ANS:
```
SELECT mov_title, mov_year, dir_fname, dir_lname,
act_fname, act_lname, role
FROM movie
NATURAL JOIN movie_direction
NATURAL JOIN movie_cast
NATURAL JOIN director
NATURAL JOIN actor
WHERE mov_time=(SELECT MIN(mov_time) FROM movie);
```

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

ANS :
```
SELECT title
FROM film
WHERE film_id in (SELECT film_id
FROM inventory
WHERE inventory_id in (
SELECT inventory_id
FROM rental
GROUP BY inventory_id
ORDER BY count(inventory_id) DESC
)) limit 1;
```

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

ANS:
```
SELECT title
FROM film
WHERE film_id IN ("highest revenue")

SELECT film_id
FROM inventory
WHERE inventory_id IN (
SELECT inventory_id
FROM rental
```

```
GROUP BY inventory_id
ORDER BY count(inventory_id) DESC
) limit(highest revenue);
```

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

ANS: SELECT mov_title, act_fname, act_lname, role
FROM movie
JOIN movie_cast
ON movie_cast.mov_id=movie.mov_id
JOIN actor
ON movie_cast.act_id=actor.act_id
WHERE actor.act_id IN (
SELECT act_id
FROM movie_cast
GROUP BY act_id HAVING COUNT(*)>=2);

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

Ans :
```
SELECT m.mov_title
FROM  movie m
JOIN movie_cast c
ON  m.mov_id = c.mov_id
WHERE c.act_id IN (
Select act_id
FROM actor
WHERE act_fname='Harrison'
AND act_lname='Ford');
```

.

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

Ans :
```
select genre.name,
        array_agg(track.name order by trackid) as tracks
    from      track
        join album using(albumid)
        join genre using(genreid)
   where album.title = 'Unplugged'
group by genre.name;
```

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 mov_title, mov_year, mov_time, movie_languages
mov_dt_rel AS Date_of_Release,
mov_rel_country AS Releasing_Country

FROM movie and movie_languages
WHERE language_id and ,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 mov_title, act_fname, act_lname, role
FROM movie
JOIN movie_cast
ON movie_cast.mov_id=movie.mov_id
JOIN actor
ON movie_cast.act_id=actor.act_id
WHERE actor.act_id IN (
SELECT act_id
FROM movie_cast
GROUP BY act_id HAVING COUNT(*)>=2);

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

ANS:
```
SELECT mov_title, mov_year, mov_dt_rel, dir_fname, dir_lname,
       act_fname, act_lname
        FROM movie a, movie_direction b, director c,
                rating d, reviewer e, actor f, movie_cast g
        WHERE a.mov_id=b.mov_id
AND  b.dir_id=c.dir_id
 AND a.mov_id=d.mov_id
  AND  d.rev_id=e.rev_id
   AND  a.mov_id=g.mov_id
    AND g.act_id=f.act_id
         AND e.rev_name IS NULL;
```

11. Write a SQL query to show the name of the 3rd most revenue generating movie and its revenue
SELECT MOVIE
FROM movie a,
WHERE a.mov_id=b.mov_id
AND d.rev_id=e.rev_id

12. Write a SQL query to show the names of all the movies which have “rumoured” movie status.

ANS:

```
SELECT
title,
budget,
release_date,
revenue,
runtime,
vote_average
FROM movie
```

```
ORDER BY revenue DESC;
```

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

```
SELECT country_name,city, department_name
FROM countries
JOIN locations USING (country_id)
JOIN departments USING (location_id);
```

14.Write a SQL query to print the movie_id in one column and name of the production company in the secondcolumn for all the movies.

```
SELECT mov_title, act_fname, act_lname, role
FROM movie
JOIN movie_cast
  ON movie_cast.mov_id=movie.mov_id
JOIN actor
  ON movie_cast.act_id=actor.act_id
WHERE actor.act_id IN (
SELECT act_id
FROM movie_cast
                GROUP BY act_id HAVING COUNT(*)>=2);
```

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

ANS :

```
SELECT movie.mov_title, mov_year, mov_dt_rel,
       mov_time,dir_fname, dir_lname
FROM movie
JOIN  movie_direction
   ON movie.mov_id = movie_direction.mov_id
JOIN director
   ON movie_direction.dir_id=director.dir_id
WHERE mov_dt_rel <'01/01/2022'
```

```
ORDER BY mov_dt_rel desc;
```