WORKSHEET 5 SQL

Please go through the below ERD before referring the answers.

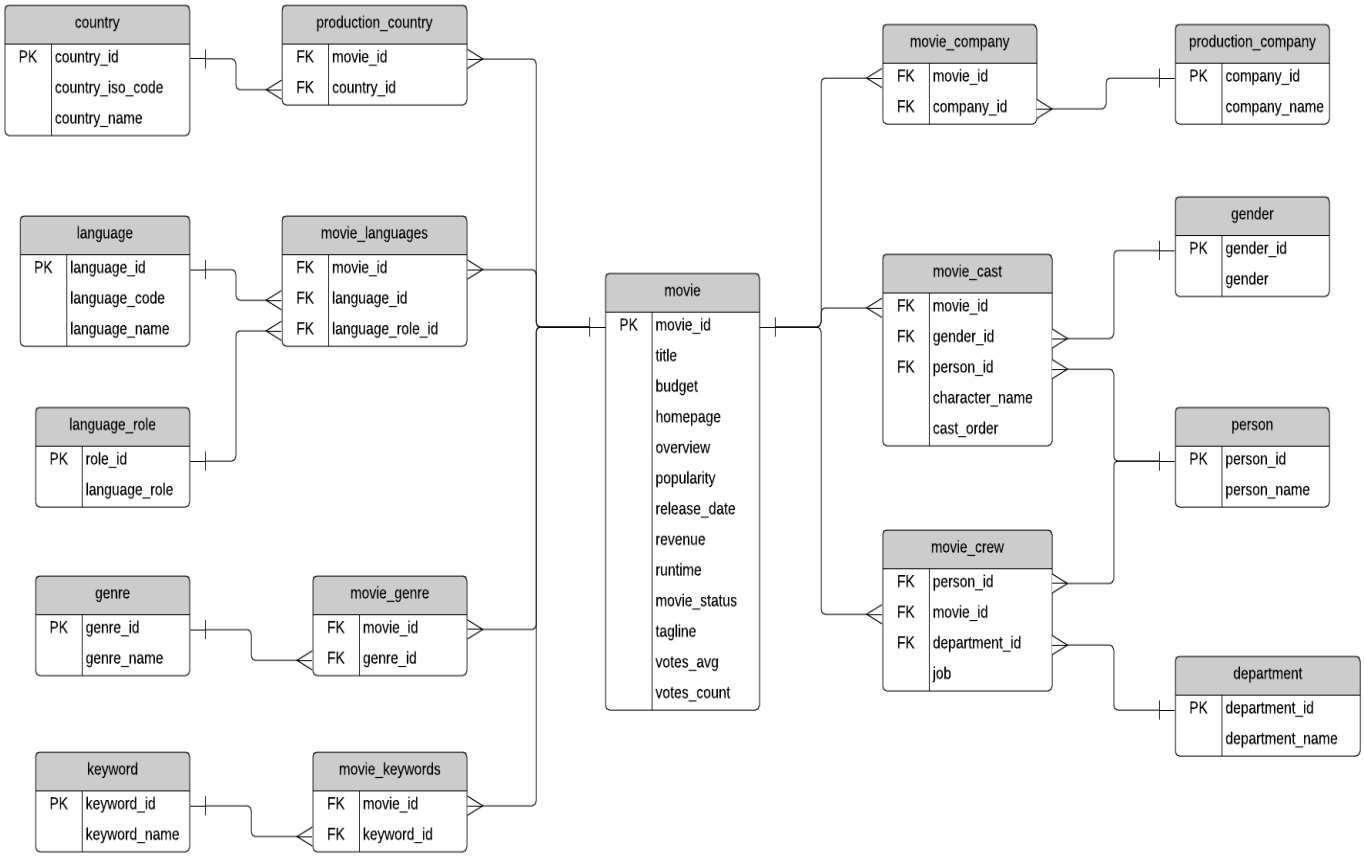

Table Explanations:

- 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).

SOLUTIONS:

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`, `revenue`/`budget` as revenue_budget_ratio FROM movie
   ORDER BY revenue_budget_ratio DESC
   LIMIT 1;
5. SELECT `title`, `person_name`, `gender`, `character_name`, `cast_order`
   FROM movie AS m INNER JOIN movie_cast AS c
   ON m.`movie_id` = c.`movie_id`
   INNER JOIN gender AS g
   ON c.`gender_id` = g.`gender_id`
   INNER JOIN person AS p
   ON c.`person_id` = p.`person_id`;
6. SELECT `country_name`, COUNT(`movie_id`) AS no_of_movies
   FROM production_country AS a INNER JOIN country AS b
   ON a.`country_id` = b.`country_id`
   GROUP BY `country_name`
   ORDER BY no_of_movies DESC
   LIMIT 1;
7. SELECT * FROM genre;
8. SELECT `language_name`, COUNT(`movie_id`) AS no_of_movies
   FROM movie_languages AS a INNER JOIN language AS b
   ON a.`language_id` = b.`language_id`
   GROUP BY b.`language_id`;
9. SELECT `title`, COUNT(`person_id`) AS no_of_cast
   FROM movie AS a INNER JOIN movie_cast AS b
   ON a.`movie_id` = b.`movie_id`
   GROUP BY b.`movie_id`;
10. SELECT `title` FROM movie
    ORDER BY `popularity` DESC
    LIMIT 10;

11. SELECT `title` FROM movie
    ORDER BY `revenue` DESC
    LIMIT 1
    OFFSET 2;
12. SELECT `title` FROM movie
    WHERE `movie_status` = “rumoured”;
13. SELECT `title`, `revenue`
    FROM movie AS a INNER JOIN production_country AS b
    ON a.`movie_id` = b.`movie_id`
    INNER JOIN country AS c
    ON c.`country_id` = b.`country_id`
    WHERE `country_name` = “United States Of America”
    ORDER BY `revenue` DESC
    LIMIT 1;
14. SELECT `movie_id`, `company_name`
    FROM movie_company AS a INNER JOIN production_company AS b
    ON a.`company_id` = b.`company_id`;
15. SELECT `title’, ‘budget` FROM movie
    ORDER BY `budget` DESC
    LIMIT 20;