# SQL – WORKSHEET 5

## https://www.databasestar.com/wp-content/uploads/2019/11/movies_erd.pngRefer 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;

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

ANS: select title from movie where runtime= (select max(runtime) from movie);

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

ANS: select title from movie where revenue= (select max(revenue) from movie);

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

ANS: select title from movie where budget= (select max(budget) from movie);

* 1. **Write a SQL query to show the movie title and its cast details like name of the person, gender, character name, cast order**
  2. **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\_name from country c INNER JOIN production\_country p ON  
c.country\_id = p.country\_id INNER JOIN movie m ON p.movie\_id = m.movie\_id;

* 1. **Write a SQL query to show all the genre\_id in one column and genre\_name in second column.**

ANS: select genere\_id, genere\_name from genere;

* 1. **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 language.language\_name,movies.movies\_name from ((language inner join movie\_languages on language.lanuage\_id=movie\_languages.language\_id) inner join movie\_languages.movie\_id= movie.movie\_id));

* 1. **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 movie.title,movie\_crew.count(person\_id),movie\_cast.count(person\_id) from ((movie inner join crew on movie.movie\_id=crew.movie\_id) inner join movie\_cast on movie.movie\_id=movie\_cast.movie\_id);

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

ANS: select top 10 title from movie order by popularity DESC;

* 1. **Write a SQL query to show the name of the 3rd most revenue generating movie and its revenue.**

ANS:select title from movie where revenue= (select top 3 revenue from movie order by revenue DESC limit 3,1);

* 1. **Write a SQL query to show the names of all the movies which have “rumored” movie status.**

select title from movie where movie\_status= ‘rumored’;

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

select title, from movie where movie\_id = ( select movie\_id from movie\_company where company\_id = (select company\_id from production\_company where company\_name = 'United States of America'));

* 1. **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.**

ANS: select movie\_company.movie\_id, production\_company.company\_name from ovie\_company inner join production\_company on movie\_company.company\_id=production\_company.company\_id;

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

ANS: select title from movie order by budget DESC LIMIT 20;