Name= Hritik yadav

Internship=34

**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.

Answer:- SELECT \* FROM Movie;

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

Answer:- SELECT title FROM Movie WHERE runtime = (SELECT MAX(runtime) FROM Movie);

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

Answer:-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.

Answer:-SELECT title FROM Movie WHERE revenue/budget = (SELECT MAX(revenue/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.

Answer:-SELECT m.title, c.person\_name, c.gender, c.character\_name, c.cast\_order

FROM Movie m

JOIN Cast c ON m.id = c.movie\_id;

1. Write a SQL query to show the country name where maximum number of movies has been produced, along with the number of movies produced.

Answer:-SELECT c.country\_name, COUNT(m.id) AS num\_movies

FROM Movie m

JOIN Country c ON m.id = c.movie\_id

GROUP BY c.country\_name

ORDER BY num\_movies DESC

LIMIT 1;

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

Answer:-SELECT genre\_id, genre\_name

FROM Genre;

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.

Answer:-SELECT l.language\_name, COUNT(m.id) AS num\_movies

FROM Language l

JOIN Movie\_Language ml ON l.id = ml.language\_id

JOIN Movie m ON m.id = ml.movie\_id

GROUP BY l.language\_name;

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.

Answer:-SELECT m.title AS movie\_name, COUNT(DISTINCT c.id) AS num\_cast\_members, COUNT(DISTINCT cr.id) AS num\_crew\_members

FROM Movie m

LEFT JOIN Cast c ON c.movie\_id = m.id

LEFT JOIN Crew cr ON cr.movie\_id = m.id

GROUP BY m.title;

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

Answer:- SELECT title, popularity

FROM movie

ORDER BY popularity DESC

LIMIT 10;

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

Answer:-SELECT title, revenue

FROM movie

ORDER BY revenue DESC

LIMIT 1 OFFSET 2;

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

Answer:-SELECT title

FROM movie

WHERE status = 'Rumored';

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

Answer:-SELECT m.title, m.revenue

FROM Movie m

INNER JOIN ProductionCountry pc ON m.id = pc.movie\_id

WHERE pc.name = 'United States of America'

ORDER BY m.revenue DESC

LIMIT 1;

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.

Answer:-SELECT movies.id AS movie\_id, production\_companies.name

FROM movies

JOIN movie\_production\_companies ON movies.id = movie\_production\_companies.movie\_id

JOIN production\_companies ON movie\_production\_companies.production\_company\_id = production\_companies.id;

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

Answer:-SELECT title

FROM movies

ORDER BY budget DESC

LIMIT 20;