**SELECT, WHERE, DISTINCT, LIKE, ORDER BY**

**SELECT** \* FROM moviesdb.movies;

**SELECT** title, industry FROM movies;

**SELECT** \* FROM movies **WHERE** industry="bollywood";

**SELECT** **count**(\*) FROM movies **WHERE** industry="bollywood";

**SELECT** **distinct** industry FROM movies;

https://www.imdb.com/

**SELECT** \* FROM movies **WHERE** title **LIKE** "%THOR%";

**SELECT** \* FROM movies **WHERE** title **LIKE** "america%";

**SELECT** \* FROM movies **WHERE** studio="";

**Takeaways**

- **SELECT**, **FROM** and **WHERE** are the basic SQL functions

- **'\*'** means all columns. Using **'\*'** after database after the SELECT query will select all columns of

- With the help of the **USE function**, you can indicate the query to use a particular database, especially when there are multiple databases

- The **COUNT function** will provide the numerical count of rows

-The **DISTINCT function** will help you see the unique values present in a given column

- **'%'** is a **wild card** search

- Use **LIKE function** and **'%'** to filter the rows based on a text value

**SELECT** \* FROM movies **WHERE** imdb\_rating>=9

**SELECT** \* FROM movies **WHERE** imdb\_rating<=5

**SELECT** \* FROM movies **WHERE** imdb\_rating>=6 **AND** imdb\_rating<=8

**SELECT** \* FROM movies **WHERE** imdb\_rating **BETWEEN** 6 AND 8

**SELECT** \* FROM movies **WHERE** release\_year=2022 **OR** release\_year=2019 **OR** release\_year=2018

**SELECT** \* FROM movies **WHERE** release\_year **IN** (2022,2019,2018)

**SELECT** \* FROM movies **WHERE** studio **IN** ("Marvel Studios","Zee Studios")

**SELECT** \* FROM movies **WHERE** imdb\_rating **IS** **NULL**

**SELECT** \* FROM movies **WHERE** imdb\_rating **IS NOT NULL**

**SELECT** \*FROM movies **WHERE** industry="bollywood" **ORDER BY** imdb\_rating

**SELECT \*** FROM movies **WHERE** industry="bollywood" **ORDER BY** imdb\_rating **desc**

**SELECT \*** FROM movies **WHERE** industry="bollywood" **ORDER BY** imdb\_rating **asc**

**SELECT \*** FROM movies **WHERE** industry="bollywood" **ORDER BY** imdb\_rating **desc LIMIT** 5

Export the data

**SELECT \*** FROM movies **WHERE** industry="bollywood" **ORDER BY** imdb\_rating **desc LIMIT** 5 **OFFSET** 1

**Takeaways**

• **<, <=, >, >=** are the basic numerical operators used in SQL.

• You can also use **AND, OR, BETWEEN, IN** to perform numerical queries.

• You can sort the table by using **'ORDER BY'** clause.

• By default, it sorts the data in ascending order but you can specify the sort order.

• **LIMIT** clause can be used to fetch the top **'N'** or bottom **'N'** amount of records. **'N'** can be any numerical value.

queries

• **OFFSET** clause help you to skip a certain number of rows in your final result.

**Summary Analytics (MIN, MAX, AVG, GroupBy)**

**SELECT** **MAX** (imdb\_rating) FROM movies **WHERE** industry="bollywood";

**SELECT** **MIN** (imdb\_rating) FROM movies **WHERE** industry="bollywood";

**SELECT** **AVG** (imdb\_rating) FROM movies **WHERE** studio="Marvel studios";

**SELECT** **ROUND** (**AVG** (imdb\_rating),2) FROM movies **WHERE** studio="Marvel studios";

**SELECT** **ROUND** (**AVG** (imdb\_rating),2) **as** avg\_rating FROM movies **WHERE** studio="Marvel studios";

**SELECT** **MIN**(imdb\_rating) **as** min\_rating, **MAX**(imdb\_rating) **as** max\_rating,

**ROUND**(**AVG** (imdb\_rating),2) **as** avg\_rating FROM movies **WHERE** studio="Marvel studios";

**SELECT** **COUNT** (\*) FROM movies **WHERE** industry="bollywood";

**SELECT** industry, **COUNT** (\*) FROM movies **GROUP BY** industry;

**SELECT** studio, **COUNT** (\*) FROM movies **GROUP BY** studio;

**SELECT** studio, **COUNT**(\*) **as** cnt FROM movies **GROUP BY** studio **ORDER BY cnt DESC**;

**SELECT** industry, **COUNT**(industry) **as** cnt, **avg**(imdb\_rating) **as** avg\_rating FROM movies **GROUP BY** industry;

**SELECT** industry, **COUNT**(industry) **as** cnt, **ROUND(avg**(imdb\_rating),1) **as** avg\_rating FROM movies **GROUP BY** industry;

**SELECT** studio, **COUNT**(studio) **as** cnt, **ROUND**(**avg**(imdb\_rating),1) **as** avg\_rating FROM movies **GROUP BY** studio **ORDER BY** avg\_rating **DESC**;

**HAVING clause, Calculated Columns**

#Print all the years where more than 2 movies were released

**SELECT** release\_year, **COUNT**(\*) FROM movies **GROUP BY** release\_year;

**SELECT** release\_year, **COUNT**(\*) **as** movies\_count FROM movies **GROUP BY** release\_year **ORDER BY** movies\_count **DESC**;

**SELECT** release\_year, **COUNT**(\*) **as** movies\_count FROM movies **WHERE** movies\_count>2 **GROUP BY** release\_year **ORDER BY** movies\_count **DESC**;

# Error Code: 1054. Unknown column 'movies\_count' in 'where clause'

FROM ---> WHERE ---> GROUP BY ---> HAVING ---> ORDER BY

**SELECT** release\_year, **COUNT**(\*) **as** movies\_count FROM movies **GROUP BY** release\_year **HAVING** movies\_count>2 **ORDER BY** movies\_count **DESC**;

**Takeaways**

• The order of query execution in - **SQL** is **FROM** **WHERE → GROUP BY HAVING ORDER BY**

• **GROUP BY** and **HAVING** clauses are often used together

• The Column you use in **HAVING** should be present in **SELECT** clause whereas **WHERE** can use columns that is not present in select clause as well

**SELECT** \* FROM moviesdb.actors;

**SELECT** **CURDATE()**;

**SELECT** **YEAR**(**CURDATE())**;

**SELECT** \*, **YEAR**(**CURDATE())-birth\_year as age** FROM actors;

**SELECT** \* FROM moviesdb.financials;

**SELECT** \*, (revenue-budget) **as** profit **FROM** moviesdb.financials;