# IMDb Data Dump

[https://drive.google.com/file/d/1KLH4ENuC-TLeigo0tFa5rKN7tNvpZDXX/view?usp=sharing](https://drive.google.com/file/d/1KLH4ENuC-TLeigo0tFa5rKN7tNvpZDXX/view?usp=sharing)

# SQL Commands

USE imdb;

SHOW TABLES;

DESCRIBE movies;

**************************************************************************************************

SELECT * FROM movies;

# more data transfer

#result-set: a set of rows that form the result of a query along with column-names and meta-data.

SELECT name,year FROM movies;

SELECT rankscore,name FROM movies;
#row order same as the one in the table

**************************************************************************************************

## LIMIT:

SELECT name,rankscore FROM movies LIMIT 20;

SELECT name,rankscore FROM movies LIMIT 20 OFFSET 40;

**************************************************************************************************

## ORDER BY:

```
# list recent movies first

SELECT name,rankscore,year FROM movies ORDER BY year DESC LIMIT 10;

# default:ASC

SELECT name,rankscore,year FROM movies ORDER BY year LIMIT 10;

# The output row order maynot be the same as the one in the table due to query optimizer and
internal data-structures/indices.

****************************************************************************************************
```

## DISTINCT:

```
# list all genres of
SELECT DISTINCT genre FROM movies_genres;


# multiple-column DISTINCT
SELECT DISTINCT first_name, last_name FROM directors;

****************************************************************************************************
```

## WHERE:

```
# list all movies with rankscore>9
SELECT name,year,rankscore FROM movies WHERE rankscore>9 ;

SELECT name,year,rankscore FROM movies WHERE rankscore>9 ORDER BY rankscore
DESC LIMIT 20;
```

## # Condition's outputs: TRUE, FALSE, NULL

## # Comparison Operators: = , <> or != , < , <= , >, >=

SELECT * FROM movies_genres WHERE genre = 'Comedy';

SELECT * FROM movies_genres WHERE genre <> 'Horror';

## NULL => doesnot-exist/unknown/missing

# "=" does not work with NULL, and will give you an empty result-set.
SELECT name,year,rankscore FROM movies WHERE rankscore = NULL;

SELECT name,year,rankscore FROM movies WHERE rankscore IS NULL LIMIT 20;

SELECT name,year,rankscore FROM movies WHERE rankscore IS NOT NULL LIMIT 20;

****************************************************************************************************

## BREAK

# LOGICAL OPERATORS: AND, OR, NOT, BETWEEN, IN, LIKE

```sql
# website search filters
SELECT name,year,rankscore FROM movies WHERE rankscore>9 AND year>2000;


SELECT name,year,rankscore FROM movies WHERE NOT year<=2000 LIMIT 20;


SELECT name,year,rankscore FROM movies WHERE rankscore>9 OR year>2007;



SELECT name,year,rankscore FROM movies WHERE year BETWEEN 1999 AND 2000;
#inclusive: year>=1999 and year<=2000


SELECT name,year,rankscore FROM movies WHERE year BETWEEN 2000 AND 1999;
#low value <= high value else you will get an empty result set


SELECT director_id, genre FROM directors_genres WHERE genre IN ('Comedy','Horror');
# same as genre='Comedy' OR genre='Horror'


SELECT name,year,rankscore FROM movies WHERE name LIKE 'Tis%';
# % => wildcard character to imply zero or more characters


SELECT first_name, last_name FROM actors WHERE first_name LIKE '%es';
# first name ending in 'es'


SELECT first_name, last_name FROM actors WHERE first_name LIKE '%es%';
#first name contains 'es'


SELECT first_name, last_name FROM actors WHERE first_name LIKE 'Agn_s';
# '_' implies exactly one character.


# If we want to match % or _, we should use the backslash as the escape character: \% and \_
```

```
SELECT first_name, last_name FROM actors WHERE first_name LIKE 'L%' AND first_name
NOT LIKE 'Li%';
```

**************************************************************************************************

## Aggregate functions: Computes a single value on a set of rows and returns the aggregate

COUNT, MIN, MAX, SUM, AVG

```
SELECT MIN(year) FROM movies;
```

```
SELECT MAX(year) FROM movies;

SELECT COUNT(*) FROM movies;

SELECT COUNT(*) FROM movies where year>2000;

SELECT COUNT(year) FROM movies;
```

**************************************************************************************************

## GROUP-BY

```
# find number of movies released per year

SELECT year, COUNT(year) FROM movies GROUP BY year;

SELECT year, COUNT(year) FROM movies GROUP BY year ORDER BY year;
SELECT year, COUNT(year) year_count FROM movies GROUP BY year ORDER BY
year_count;
# year_count is an alias.

# often used with COUNT, MIN, MAX or SUM.
# if grouping columns contain NULL values, all null values are grouped together.
```

**************************************************************************************************

## HAVING:

```
# Print years which have >1000 movies in our DB [Data Scientist for Analysis]

SELECT year, COUNT(year) year_count FROM movies GROUP BY year HAVING
year_count>1000;
# specify a condition on groups using HAVING.
```

### Order of execution:

1. GROUP BY to create groups
2. apply the AGGREGATE FUNCTION
3. Apply HAVING condition.

```
# often used along with GROUP BY. Not Mandatory.

SELECT name, year  FROM movies HAVING year>2000;
```

```
# HAVING without GROUP BY is same as WHERE
```

```
SELECT year, COUNT(year) year_count FROM movies WHERE rankscore>9 GROUP BY year
HAVING year_count>20;
```

# HAVING vs WHERE

## WHERE is applied on individual rows while HAVING is applied on groups.
## HAVING is applied after grouping while WHERE is used before grouping.

*********************************************************************************************************

## Installation: Windows (Optional)

```
Steps to Install MySQL 8.0.13 on windows operating system
=========================================================

1. Go to mysql.com website
2. Select Downloads option
3. Select MySQL community Edition  (https://dev.mysql.com/downloads/)
4. Download MySQL community server 8.0.13
   (https://dev.mysql.com/downloads/windows/installer/8.0.html)

While installing

-> choose setup Type: custom  (Next)
-> + MySQL servers
         + MySQL server 8
               + MySQL server 8.0.13 (select)
   + Applications
         + MySQL workbench
               + MySQL server 8.0.13 (select)

   Next
-> Install other essential software required

                                                                  Requirement
   select           MySQL Server 8.0.13      Microsoft visual c++ 2015
   Redistribution
   Select     MySQL Workbench 8.0.13   Microsoft visual c++ 2015
   Redistribution

   Next

-> Click Execute Button

-> Product Configuration
   Click Next

-> Group Replication
   Select Standalone MySQL server / Classic MySQL replication

   Click Next

-> Type and Networking

   Config Type: Development Computer

   Connectivity:
```

```
   Check TCP/IP           PORT 3306 (Default)

   Click Next

-> Authentication Method

   Use strong password Encryption for Authentication (select)

   Click Next

-> Account And Roles

   MySQL root password : Enter password
   Repeat Password: reenter it

   Optional : You can add new user and set the roles here

   Click Next

-> Windows Service

   check the box: start the MySQL server at system start
   Run Windows service as:

   standard system account

   Click Next

-> Execute it

-> Open MySQL command Line client
Enter password:
```

## Installation: Other (Optional)

### MySQL installation:

https://www.digitalocean.com/community/tutorial_collections/how-to-install-mysql

### Software to interact with DB and run SQL queries:

1. Windows: https://www.heidisql.com/download.php

2. Linux: https://dbeaver.io/download/

3. Mac: https://apps.apple.com/us/app/sequel-ace/id1518036000

## Link to Doc:

https://docs.google.com/document/d/1VbZzwS0N-TcFPRqTBqRTiXjTwTGEDdi3sItYktwONMY/edit?usp=sharing