SQL

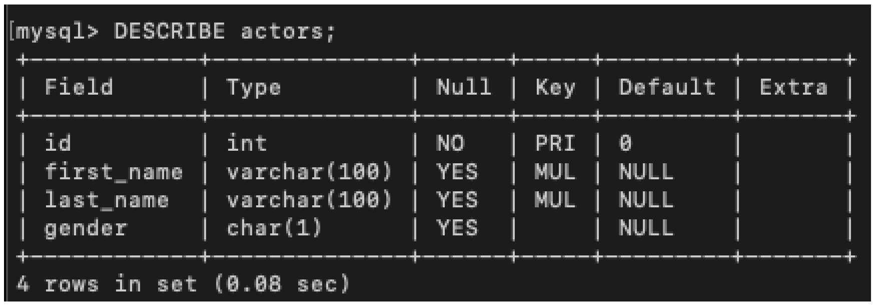
Password: sql14*****1

- **How to create a MySQL database (imdb) from sql file (imbd.sql)?**
  - mysql> CREATE DATABASE imdb; → It will create a database called "imdb"
  - mysql> USE imdb;
  - mysql> SOURCE /user/Downloads/SQL/imbd.sql; → It has to dump the "imdb.sql" data into the "imdb" database.

- **What are MySQL Server and MySQL databases?**
  - MySQL Server:
    - It is your computer or "local host" (/usr/local/sql-9….). MySQL server refers to the computer MySQL is running on or the MySQL application.
    - "MySQL Server" is a running MySQL database program, while a MySQL database is the data managed by a MySQL database program. To be pedantic, a single MySQL server can manage many local MySQL databases.)
  - MySQL database refers to an individual database that is running on a MySQL server. A single MySQL server may contain multiple MySQL databases.

Some Commands of SQL:

- *mysql -u root -p* → It is saying that you want to enter the database as a root user. Note: lower case letters will also work (use imdb). It is just a good habit as a programmer to use capital letters to specify that it is a SQL command.
- *USE imdb* → You want to use the "imdb" database for the work. Here the "USE" can be used to change the current database to another one like "*USE amazon*". Note: lower case letters will also work (use imdb). It is just a good habit as a programmer to use capital letters to specify that it is a SQL command.
- *control + L* → Clear the screen.
- *SHOW TABLES;* → Don't forget to use the semicolon at the end. Otherwise, it will wait for it by showing this sign "→". This command will show the "tables" present in "imdb" database.
- *DESCRIBE actors;* → To describe one of the tables from the database, you can use this command.



```
mysql> DESCRIBE actors;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| id         | int          | NO   | PRI | 0       |       |
| first_name | varchar(100) | YES  | MUL | NULL    |       |
| last_name  | varchar(100) | YES  | MUL | NULL    |       |
| gender     | char(1)      | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
4 rows in set (0.08 sec)
```

  - **Field:** There are four columns in the actor's table. (id, first_name, last_name, gender)
  - **Type:** It tells you the datatypes of the content present in the each table. For example, id → all the rows in this column is integer. varchar(100) → There can be variable lengths of characters in this column row, like Arjun, Ajay etc.
  - **Null:** It tells you whether the rows in this column can have "NULL" values or not. For example, id → It can be NULL. It must have some value. gender → It can have null values.
  - **Key:** It shows which of the columns is Primary and Mul. By primary, it means every row in the column will have unique values. For example, id → So, for actors, there will be only one unique id. Mul → Multiple values can be present, such as the first_name of many actors being the same.
  - **Default:** It shows what the given value should be if no value is present—for example, 0 for id and NULL for others.

- *SELECT * FROM movies;* → It will show all the columns and the content present in the table movies.

But what if I want to select only two columns (year, name) from the imdb database?

- *SELECT name,year FROM movies;* → The result it gives is another table having name and year as a column. This query is faster than the above query, so always recommend asking for those queries that you need.
  Note: If you want to change the order of the columns of the result table, you can just switch the name in the query. For example, *SELECT year, name FROM movies;*.

- *SELECT name, year FROM movies LIMIT 20;* → In this, we are getting only the top 20 rows of the column names and years.

But what if I want to get only rows from 20 to 40?

- *SELECT name, year FROM movies LIMIT 40 OFFSET 20;* → It says in my result table I want 40 rows but offset (ignore) the first 20 rows and then give the result table. OFFSET: It says how many rows from the start you want to ignore.

If I want to see the latest 20 movies' names.

- *SELECT name, year FROM movies ORDER BY year DESC LIMIT 20;* → Here, I want my result table to have two columns, name and year. And I want to see only 20 rows where the year is in descending order. You can see that in every website you use, for example, Amazon, when you sort the items based on price or ratings.
  Note: If you do not mention the "DESC" then the default order is the ascending order.

If I want to see how many types of genres of movies I have.

- *SELECT DISTINCT genre FROM movies_generes;* → Here, I am getting "genre" table from the movies_geners table. It will only show distinct genres present in the column genre. For example, On Amazon, if you select a particular color or size of shirt, it only shows that.

Note: If I want to see the distinct director name. *SELECT DISTINCT first_name, last_name FROM directors ORDER BY first_name LIMIT 20;* Here, we are getting the table which has unique directors based on their first_name and last_name.

If I want to watch movies whose ratings are more than 9.5.

- *SELECT name, year, rankscore FROM movies WHERE rankscore>9.5 ORDER BY year DESC LIMIT 20;* It will give me the table having 20 rows where all the movies are sorted by latest year and having more than 9.5 rating.
- *SELECT movie_id, genre FROM movies_genres WHERE genre = 'comedy' LIMIT 20;* It will give us the table which has 20 rows of only comedy movies.

Note: We can use other comparison operators too with WHERE "comparison Operators: = , <> or != , < , <= , >, >= ".

NULL → does not exist/unknown/missing values in the row.

Note: '=' doesn't work with NULL, it will give you an empty result set. For example, *SELECT * FROM movies WHERE rankscore = NULL;* → It will give you an "Empty set". You can not compare NULL with the "=" symbol. Then how can we work with that? If you want to get all the NULL rankscore what you can do is use "IS NULL" instead of "=".
*SELECT * FROM movies WHERE rankscore IS NULL LIMIT 20;* → This will give us the top 20 rows of the table where the rankscore is NULL. Similarly, you can use "IS NOT NULL".

If I want to filter all the T-shirt items based on material: cotton, brand: levis, color: maroon, and other attributes. How can we do that? Filtering more than one attribute.

Logical Operators: AND, OR, NOT, ALL, ANY, BETWEEN, EXIST, LIKE, SOME, IN
For example,

- *SELECT * FROM movies WHERE year>2000* ***AND*** *rankscore>9.5 LIMIT 20;* → It will give you all the movies that are released after 2000 and have a rankscore of more than 9.5. Similarly, you can use "***NOT*** *year <= 2000*".
- *SELECT * FROM movies WHERE year>2007* ***OR*** *rankscore>9.9 LIMIT 20;*

If I want to select movies released between 2001 and 2002 inclusive.

- *SELECT * FROM movies WHERE year* ***BETWEEN*** *2001 AND 2002;* → Here, the order matters.
    - *SELECT * FROM movies WHERE year >= 2001 AND year <= 2002;*

Note: *SELECT * FROM movies WHERE year* ***BETWEEN*** *2002 AND 2001;* Here you will get "Empty result set".

Is there a shorter form of doing this?

- *SELECT director_id, genre FROM directors_genre WHERE genre='Comedy' OR genre='Horror'.*
- *SELECT director_id, genre FROM directors_genre WHERE genre* ***IN*** *('Comedy',' Horror').*

If I want to select all the movie name which starts with "Tis".

- *SELECT name, year FROM movies WHERE name LIKE 'Tis%';* → Here, the **"%"** refers to a wildcard character which means it can be anything. It will give the result table having the movie name start with "Tis."
- *SELECT name, year FROM movies WHERE name LIKE '%es';* → Here we are getting the name of the movies ending with "es" and the released date.
- What if we don't use the "%"? Our task here is to find the first name of the directors, which starts with "Agn" and then after one character, it has "s." *SELECT name, year FROM movies WHERE name LIKE 'Agn_s%';* → Here **"_" represents the one wildcard character.**

Escape character: "\"

- *SELECT name, percentage FROM result WHERE percentage LIKE '96%';* → Here is one problem, the "%" is the wildcard character. It will show the rows where the percentage starts with "96". To overcome this problem, we can use "\".
    - *SELECT name, percentage FROM result WHERE percentage LIKE '96\%';*

Note: So, we can use the "\" if we want to use "_" or "%" as a symbol, not as a wildcard we must use "\".

Aggregate Functions: Computes a single value on a set of rows and returns the aggregate.

COUNT, MIN, MAX, AVG, SUM

How many movies or a number of movies were released after 2000?

- *SELECT COUNT(*) FROM movies WHERE year > 2000;* → It will give you all the movies which are released after 2000.

To find the MIN or MAX year.

- *SELECT MIN(year) FROM movies;*

To find the sum and average of the rank scores of all the movies.

- *SELECT SUM(rankscore) FROM movies;*

Group BY: Ofter used with Aggregate operators (COUNT(), MIN, MAX,......)

How do we get the number of movies released per year?

- *SELECT year, COUNT(year) FROM movies GROUP BY year ORDER BY year;* → Internally, it makes all the rows which are of the same year group together after grouping together, it performs the count operation of every row and gives you the table.

If I want to show a table in ascending order of COUNT(year), how can I do that? – alias

- *SELCET year, COUNT(year) count_year FROM movies GROUP BY year ORDER BY count_year;* → It will create an alias "count_year" and use it to order by. Here you can not use order by on the COUNT(year), because COUNT() is the aggregator operator.

Note: if grouping columns contain NULL values, all null values are grouped together.