# SQL PRIME COURSE COMMANDS

## SQL Commands

This file contains all the commands discussed in the SQL course.

USE imdb;

SHOW TABLES;

DESCRIBE movies;

## SELECT Statement:

**Questions:** Select all columns from movies

SELECT * FROM movies;

**Questions:** Select column name &

SELECT name, year

FROM movies;

**Questions:** Select column rank from movies:

SELECT rankscore, name

FROM movies;

## LIMIT

**Questions:** Show name and ra 0 rows from movies table.

SELECT name, ranks

FROM movies LIM

**Questions:** Show n

SELECT name,rankscore

FROM movies LIMIT 20 OFFSE

## ORDER BY

**Question:** 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;

# SQL PRIME COURSE COMMANDS

## DISTINCT

**Question:** Return a list of all genres.

```
SELECT DISTINCT genre
FROM movies_genres;

SELECT DISTINCT first_name, last_name
FROM directors;
WHERE:

**Question:** list all movies with rankscore
SELECT name, year,rankscore
FROM movies
WHERE rankscore>9;

**Question:** list all movies with ra                    y first 20 rows by top rankscore first.
SELECT name, year,rankscore
FROM movies
WHERE rankscore>9
ORDER BY rankscore DESC
LIMIT 20;

# Condition's output
# Comparison Ope

**Question:** List all movies wher                    Comedy.
SELECT * FROM movies_genres
WHERE genre = 'Comedy';

**Question:** List all movies where the movie genre is not Horror.
SELECT * FROM movies_genres
WHERE genre <> 'Horror';


NULL => does not-exist/unknown/missing
# "=" does not work with NULL, will give you an empty result-set.
```

# SQL PRIME COURSE COMMANDS

**Question:** List all movies where the rankscore of the movie is NULL and print the name, year, and rankscore of the movie.

```
SELECT name, year,rankscore
FROM movies
WHERE rankscore = NULL;
```

**Question:** List all movies where rankscore of the movie is NULL and print the name, year, and rankscore of the movie display first 2o movies.

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

**Question:** List all movies where movie is not NULL and print the nan scor of the movie display first movies.

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

## LOGICAL OPERAT

AND, OR, NOT, ALL

**Question:** Print th the year 2000 and rankscore of the movie is greater than 9

```
SELECT name, year,rankscore
FROM movies
WHERE rankscore>9 AND year>
```

**Question:** Display first 20 movies which released after 2000. Print the name of the movie, the year it was released, and the rankscore the movie.

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

# SQL PRIME COURSE COMMANDS

**Questions:** Display movies having rankscore more than 9, and the movie should be released after 2007.

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

**Question:** Display the movies which were released between the years 1999 and 2000. Both years should be inclusive.

```
SELECT name, year,rankscore
FROM movies
WHERE year BETWEEN 1999 AND 2000;
# This will throw an error
SELECT name, year,rankscore
FROM movies
WHERE year BETWEEN 2000 AN
Because low value cannot be hi                    value.

This is the same as genre='Com                    Horror'
SELECT director_id, genre
FROM directors_genres
WHERE genre IN ('Comedy','Ho

 %  wildcard character to imply                    aracters
SELECT name, year,ran
FROM movies
WHERE name LIK
```

**Question:** Display the                    st name of the actor ends with 'es'.

```
SELECT first_name, last_name
FROM actors
WHERE first_name LIKE '%es';
```

**Question:** Display the first name and last name of the actors where the first name of the actor contains with 'es'.

```
SELECT first_name, last_name
FROM actors
WHERE first_name LIKE '%es%';

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

# SQL PRIME COURSE COMMANDS

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%’;

## Window Functions:

Aggregate window functions (Count, MIN, MAX, AV

**Question:** Select the minimum value of

SELECT MIN(year) FROM movies;

**Question:** Select the maximum ovie year released.

SELECT MAX(year) FROM movi

**Question:** Return the total nur n the dataset.

SELECT COUNT(*) FROM movie

**Question:** How many movies v er the year 2000?

SELECT COUNT(*)

FROM movies

WHERE year>200

**Question:** Count the year prese se, excluding the null values.

SELECT COUNT(year) FROM mo

Analytical window functions (ROW_NUMBER, RANK & DENSE RANK)

**Question:** Select movie with highest rankscore

SELECT id, name, rankscore

FROM movies ORDER BY rankscore DESC;

# Use of ROW_NUMBER

# SQL PRIME COURSE COMMANDS

SELECT id, name, rankscore,

ROW_NUMBER() OVER (ORDER BY rankscore DESC) as row_number

FROM movies;

# USE of PARTITION on year and ROW number: As soon as the year changes it will reset the row number.

SELECT id, name, year, rankscore,

ROW_NUMBER() OVER

(PARTITION BY year ORDER BY rankscore DE

FROM movies;

# RANK & DENSE RANK

SELECT id, name, rankscore,

RANK() OVER (ORDER BY ranks ank_number`,

DENSE_RANK() OVER (ORDER SC) as `dense_rank_number` FROM r

## Group By & Havin

**Question:** Find the number of per year.

SELECT year, COUNT(

FROM movies

GROUP BY year;

# Order by the results

SELECT year, COUNT(year)

FROM movies

GROUP BY year ORDER BY year;

# year_count is an alias, often used with COUNT, MIN, MAX or SUM.

SELECT year, COUNT(year) year_count

FROM movies

GROUP BY year ORDER BY year_count;

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

# SQL PRIME COURSE COMMANDS

## Having

**Question:** Print all the years where the number of movies released in the year is greater than 1000.

```
SELECT year, COUNT(year) year_count
FROM movies
GROUP BY year HAVING year_count>1000;

# having clause without using the group b

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

**Question:** Print year of movies  ar there are 20 movies which have rar  han 9.

```
SELECT year, COUNT(year) yea
FROM movies
WHERE rankscore>9 GROUP B  ear_count>20;
```

## JOIN's

**Question:** For eac

```
SELECT m.name, g.
FROM movies m  JOIN movies  d=g.movie_id
LIMIT 20;

# Left Join

SELECT m.name, g.genre
FROM movies m  LEFT JOIN movies_genres g ON m.id=g.movie_id
LIMIT 20;

# 3 - way join
#Practical note about joins: Joins can be expensive computationally when we have large tables.
```

# SQL PRIME COURSE COMMANDS

**Question:** Return the Name and Last name of all the actors who have worked in the movie ‘officer 444’.

```
SELECT a.first_name, a.last_name
FROM actors a
JOIN roles r ON a.id=r.actor_id
JOIN movies m on m.id=r.movie_id AND m.name=‘Officer 444’;
```

## SubQueries

**Question:** List all the actors in the movie

```
SELECT first_name, last_name
FROM actors
WHERE id IN
( SELECT actor_id from roles W                N
( SELECT id FROM movies whe                dler’s List”));
```

**Question:** Select all the movies                e is same as the maximum rankscore                ne database.

```
SELECT * FROM movies
WHERE rankscore >= ALL
(SELECT MAX(rankscor
```

## DML: Inser

```
# insert single value
INSERT INTO movies(id, name,                VALUES (412321, ‘Tho’’, 2011, 7);

#insert multiple values
INSERT INTO movies(id, name, year, rankscore) VALUES (412321, ‘Thor’, 2011, 7), (412322, ‘Iron Man’, 2008, 7.9), (412323, ‘Iron Man 2’, 2010,
```

## DML: Update

**Question:** Update the rankscore of a movie to value 9.

```
UPDATE movies SET rankscore=9 where id=412321;
```

# SQL PRIME COURSE COMMANDS

## DML: Delete

**Question:** Delete a movie from the database.

DELETE FROM movies WHERE id=412321;

## DDL: Create Table

CREATE TABLE language (id INT PRIMARY

## DDL: Alter, Modify, D

ALTER TABLE language ADD cou

ALTER TABLE language MODIF HAR(60);

ALTER TABLE language DROP

ALTER TABLE language DROP

## DDL:  DROP TABL ATE, DELETE

DROP TABLE <tablename>;

DROP TABLE Tabl

TRUNCATE TABLE