MySQL Commands and Descriptions

USE

Select the database to use for subsequent operations.

```
USE moviesdb;
```

SELECT

Retrieve all records from the movies table.

```
SELECT * FROM moviesdb.movies;
```

Retrieve specific columns from the movies table.

```
SELECT title, industry FROM movies;
```

DISTINCT

Retrieve unique values in a specified column.

```
SELECT DISTINCT industry FROM movies;
```

LIKE

Retrieve records where the title contains 'america'.

```
SELECT * FROM movies WHERE title LIKE '%america%';
```

WHERE

Retrieve records where the industry is 'Hollywood'.

```
SELECT * FROM movies WHERE industry='Hollywood';
```

Retrieve records where the studio column is empty (NULL).

```
SELECT * FROM movies WHERE studio='';
```

COUNT

Count the number of records where the industry is 'Hollywood'.

```
SELECT COUNT(*) FROM movies WHERE industry='Hollywood';
```

Comparison Operators

Retrieve records where the IMDb rating is 9 or higher.

```
SELECT * FROM movies WHERE imdb_rating >= 9;
```

AND

Retrieve records where the IMDb rating is greater than 8 and less than or equal to 9.

```
SELECT * FROM movies WHERE imdb_rating > 8 AND imdb_rating <= 9;
```

BETWEEN

Retrieve records where the IMDb rating is between 8 and 9.

```
SELECT * FROM movies WHERE imdb_rating BETWEEN 8 AND 9;
```

OR

Retrieve records where the release year is 2022, 2018, or 2019.

```
SELECT * FROM movies WHERE release_year = 2022 OR release_year = 2018 OR
release_year = 2019;
```

IN

Retrieve records where the release year is in the list (2018, 2019, 2022).

```
SELECT * FROM movies WHERE release_year IN (2018, 2019, 2022);
```

NULL

Retrieve records where the IMDb rating is NULL.

```
SELECT imdb_rating FROM movies WHERE imdb_rating IS NULL;
```

NOT NULL

Retrieve records where the IMDb rating is NOT NULL.

```
SELECT imdb_rating FROM movies WHERE imdb_rating IS NOT NULL;
```

ORDER BY

Retrieve records where the industry is 'Hollywood', ordered by IMDb rating (ascending).

```
SELECT * FROM movies WHERE industry='Hollywood' ORDER BY imdb_rating;
```

Retrieve records where the industry is 'Hollywood', ordered by IMDb rating (descending).

```
SELECT * FROM movies WHERE industry='Hollywood' ORDER BY imdb_rating DESC;
```

LIMIT

Retrieve the top 5 records where the industry is 'Hollywood', ordered by IMDb rating (descending).

```
SELECT * FROM movies WHERE industry='Hollywood' ORDER BY imdb_rating DESC
LIMIT 5;
```

MAX

Retrieve the maximum IMDb rating from the movies table.

```
SELECT MAX(imdb_rating) FROM movies;
```

MIN

Retrieve the minimum IMDb rating from the movies table.

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

AVG with ROUND

Retrieve the average IMDb rating from the movies table, rounded to 2 decimal places, where the studio is 'Marvel studios'.

```
SELECT ROUND(AVG(imdb_rating), 2) FROM movies WHERE studio='Marvel
studios';
```

AS

Retrieve the maximum, minimum, and average IMDb rating, renaming the columns.

```
SELECT MAX(imdb_rating) AS max_rating, MIN(imdb_rating) AS min_rating,
ROUND(AVG(imdb_rating), 2) AS avg_rating FROM movies WHERE studio='Marvel
studios';
```

GROUP BY

Group records by studio and count them, ordering by count (descending).

```
SELECT studio, COUNT(*) AS Count FROM movies GROUP BY studio ORDER BY Count
DESC;
```

Group records by studio, count them, and calculate the average IMDb rating, ordering by average rating (descending).

```
SELECT studio, COUNT(*) AS Count, ROUND(AVG(imdb_rating), 1) AS avg_rating
FROM movies WHERE studio != '' GROUP BY studio ORDER BY avg_rating DESC;
```

HAVING

Group records by release year and count them, only including groups with more than 2 records.

```
SELECT release_year, COUNT(*) AS Count FROM movies GROUP BY release_year
HAVING Count > 2 ORDER BY Count DESC;
```

YEAR() & CURDATE()

Retrieve all actors and calculate their age based on the current year, ordering by age.

```
SELECT *, YEAR(CURDATE()) - birth_year AS age FROM actors ORDER BY age;
```

Profit

Retrieve all records from the financials table and calculate profit as revenue minus budget.

```
SELECT *, (revenue - budget) AS profit FROM financials;
```

USE to INR

Convert revenue to INR if the currency is USD.

```
SELECT *, IF(currency = 'USD', revenue * 77, revenue) AS revenue_inr FROM
financials;
```

Unit Conversion

Convert revenue to millions based on unit.

```
SELECT *, CASE WHEN unit = 'Billions' THEN revenue * 1000 WHEN unit =
'Thousands' THEN revenue / 1000 ELSE revenue END AS revenue_mil FROM
financials;
```

INNER JOIN

Perform an inner join to combine movies and financials tables.

```
SELECT m.movie_id, title, budget, revenue, unit, currency FROM movies m
JOIN financials f ON m.movie_id = f.movie_id;
```

LEFT JOIN

Perform a left join to combine movies and financials tables.

```
SELECT m.movie_id, title, budget, revenue, unit, currency FROM movies m
LEFT JOIN financials f ON m.movie_id = f.movie_id;
```

RIGHT JOIN

Perform a right join to combine movies and financials tables.

```
SELECT f.movie_id, title, budget, revenue, unit, currency FROM movies m
RIGHT JOIN financials f ON m.movie_id = f.movie_id;
```

UNION (FULL JOIN)

Combine the results of a left join and a right join to simulate a full join.

```
SELECT m.movie_id, title, budget, revenue, unit, currency FROM movies m
LEFT JOIN financials f ON m.movie_id = f.movie_id UNION SELECT f.movie_id,
title, budget, revenue, unit, currency FROM movies m RIGHT JOIN financials
f ON m.movie_id = f.movie_id;
```

USING

Perform a join using the column that has the same name in both tables.

```
SELECT movie_id, title, budget, revenue, unit, currency FROM movies RIGHT
JOIN financials USING (movie_id);
```

INSERT

Insert new records into the actors table.

```
INSERT INTO actors (name, birth_year) VALUES ('Sriram', 2000), ('Sushanth',
1998), ('Dheeraj', 2003);
```

Disable Safe Updates

Disable safe updates for the session.

```
SET _SAFE_UPDATES = 0;
```

Enable Safe Updates

Re-enable safe updates after modifications.

```
SET _SAFE_UPDATES = 1;
```

DELETE

Delete a specific actor from the actors table.

```
DELETE FROM actors WHERE name = 'Sushanth';
```

UPDATE

Update the name and birth year of a specific actor.

```
UPDATE actors SET name = 'Sushanth', birth_year = 1999 WHERE actor_id =
173;
```