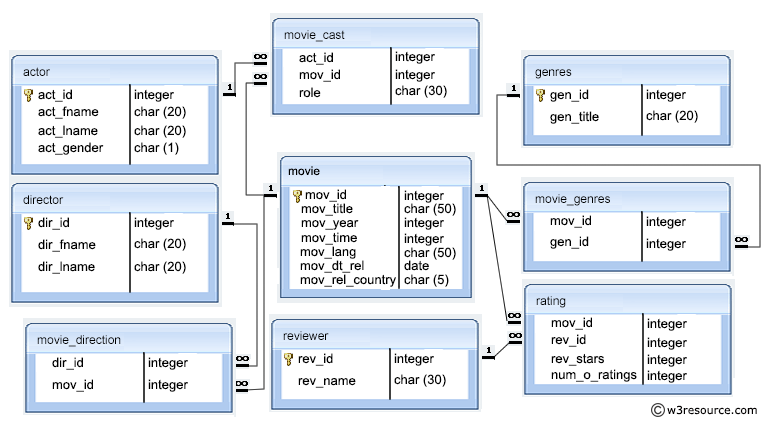
**MOVIE DATABASE**

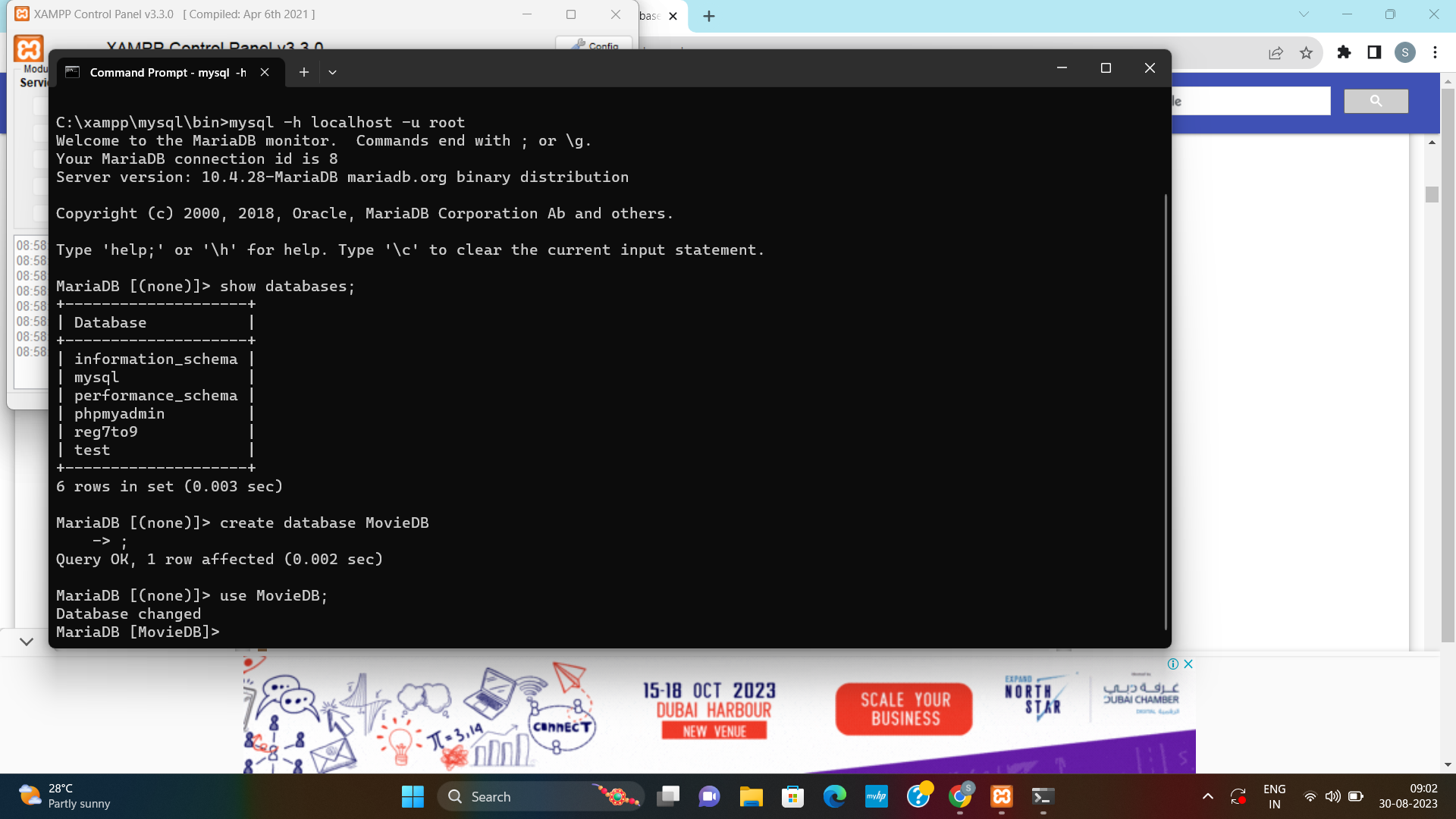


Firstly, define the main entities in Movie database. The main entities for ERD are Movie, Genres, Actor, Director and Reviewer

**TO CREATE DATABASE MovieDB**

**Syntax:**

Create Database MovieDB;



**CREATING TABLE – MOVIE**

The **Movie** table contains information about each movie. It stores information like movie id, movie title, movie year, movie time, movie lang, movie release date, movie release country.

**Syntax:**

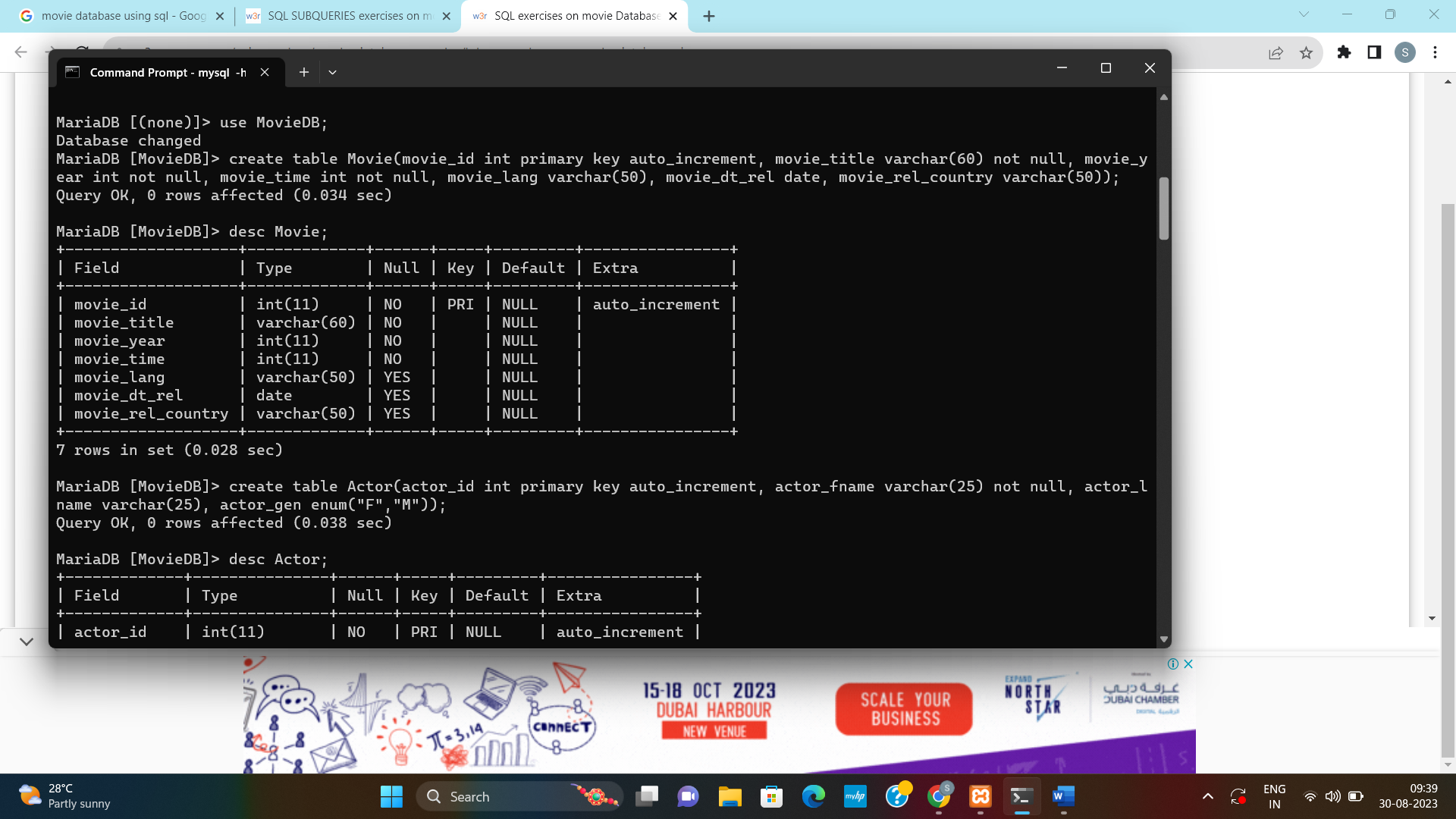
Create table Movie(movie\_id int primary key auto\_increment, movie\_title varchar(60) not null, movie\_year int not null, movie\_time int not null, movie\_lang varchar(50), movie\_dt\_rel date, movie\_rel\_country varchar(50));

**CREATING TABLE – ACTOR**

The **Actor** table contains information about each actor. It stores information like actor id, actor first name, actor last name, actor gender.

**Syntax:**

Create table Actor(actor\_id int primary key auto\_increment, actor\_fname varchar(25) not null, actor\_lname varchar(25), actor\_gen enum(“F”,”M”));



**CREATING TABLE – DIRECTOR**

The **Director** table contains information about each director. It stores information like director id, director first name, director last name.

**Syntax:**

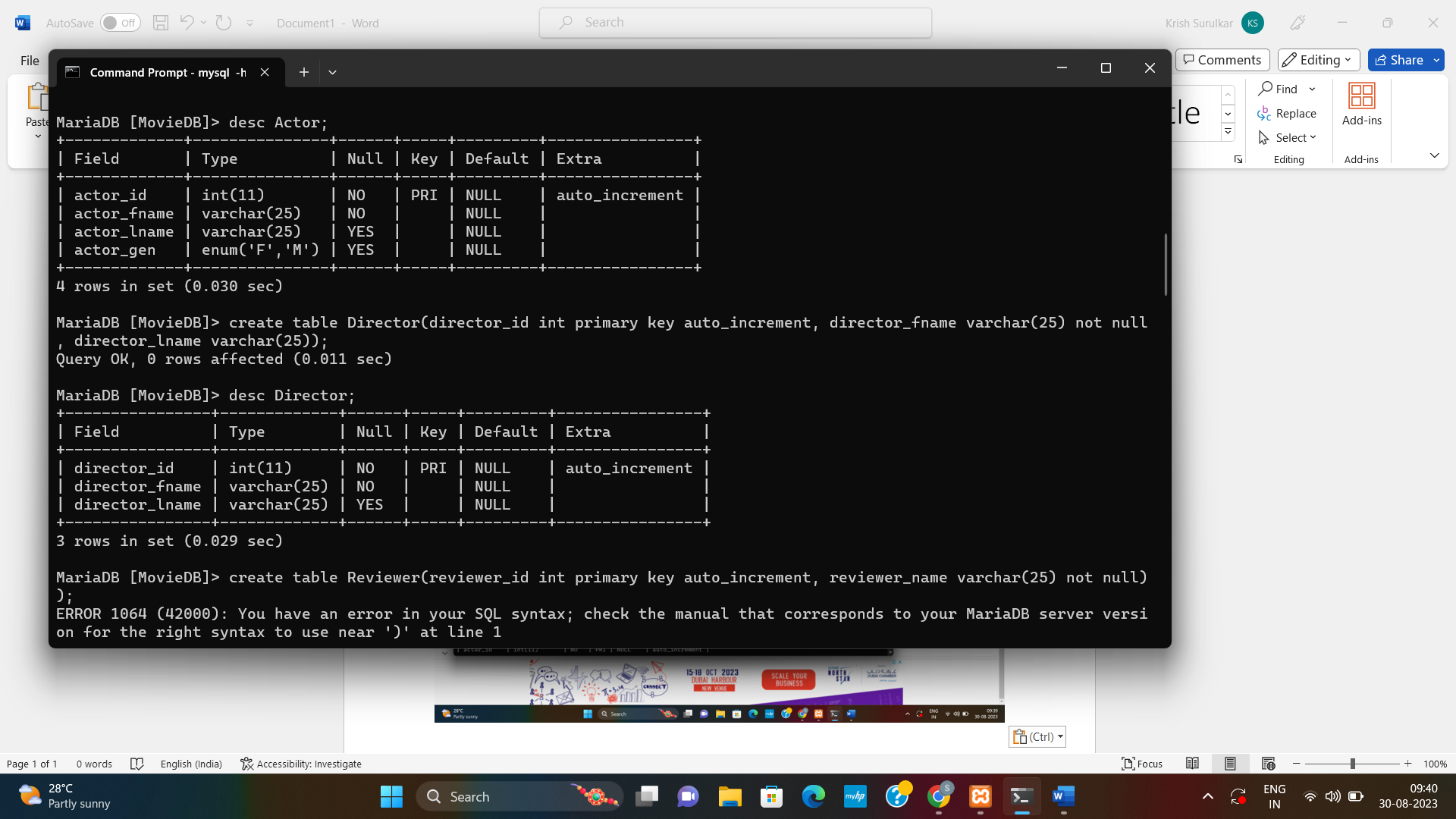
Create table Director(director\_id int primary key auto\_increment, director\_fname varchar(25) not null, director\_lname varchar(25));

**CREATING TABLE – REVIEWER**

The **Reviewer** table contains information about each actor. It stores information like reviewer id, reviewer first name.

**Syntax:**

Create table Reviewer(reviewer\_id int primary key auto\_increment, reviewer\_fname varchar(25) not null);

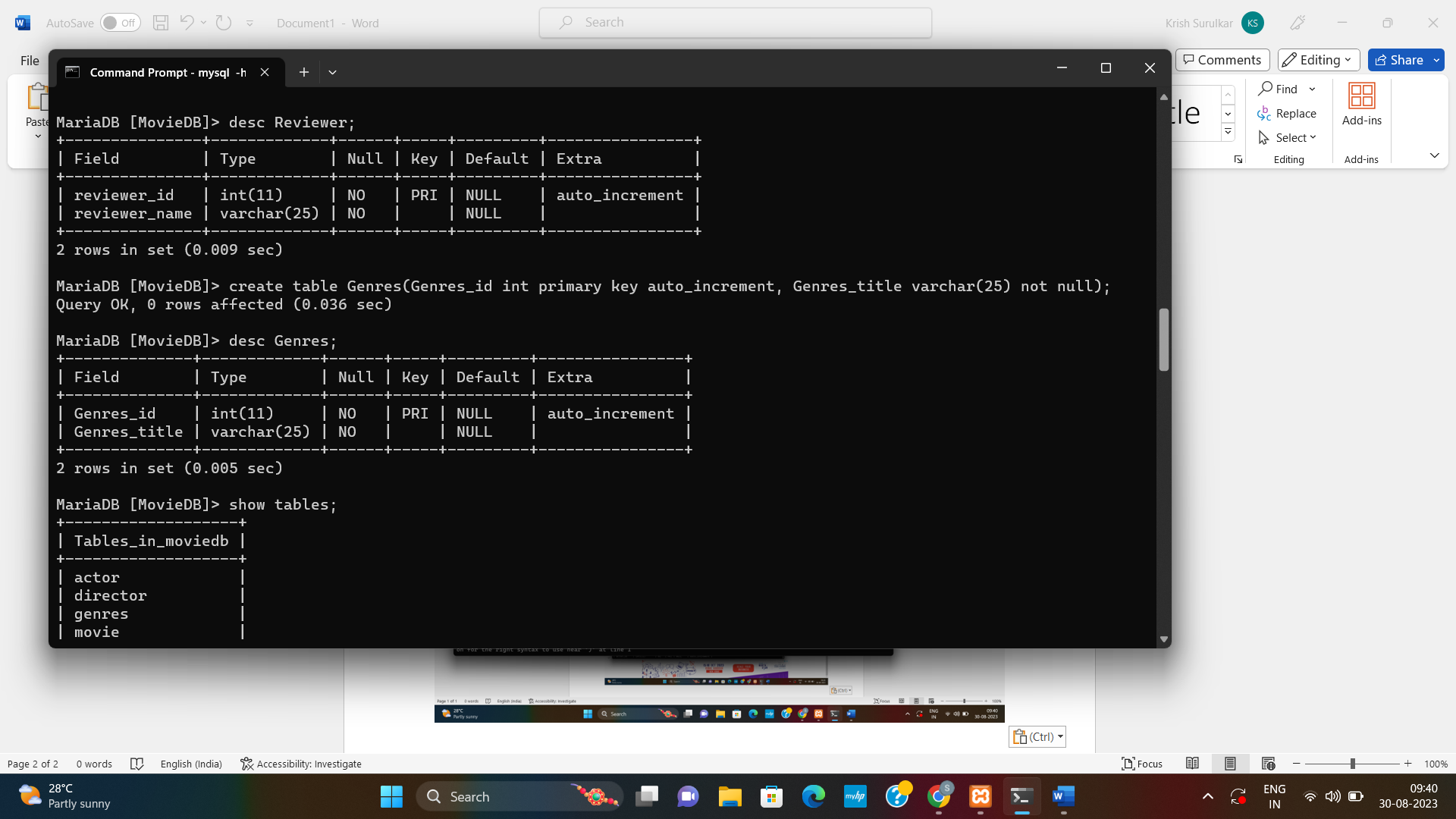


**CREATING TABLE – GENRES**

**Genres** define which category a movie fits into, such as Comedy or Horror. The **Genres** table contains information about each genres. It stores information like genres id, genres title.

**Syntax:**

Create table Genres(genres\_id int primary key auto\_increment, genres\_title varchar(25) not null);

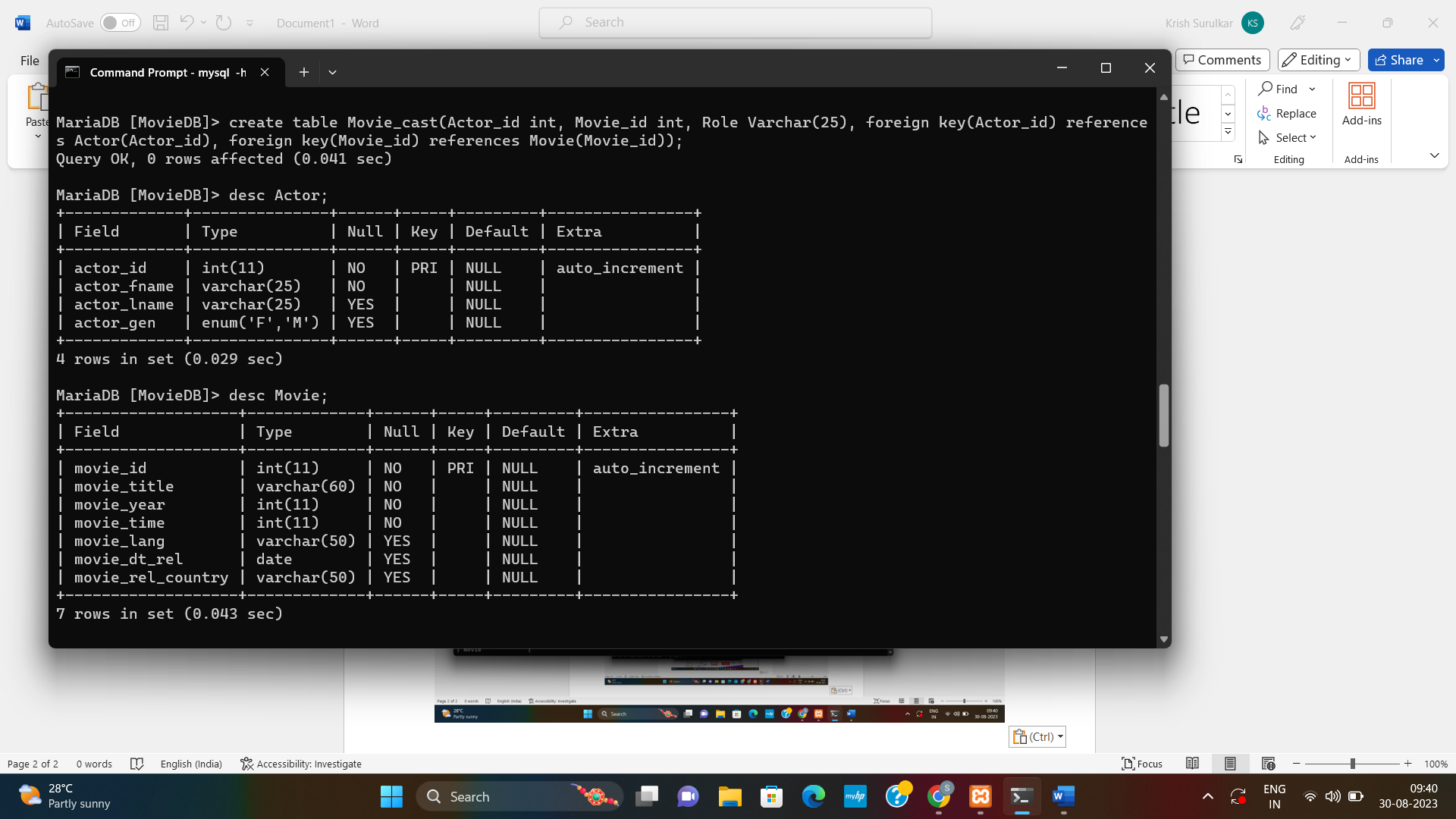


**CREATING TABLE – MOVIE\_CAST**

The **movie\_cast** table contains records of each person in a movie as a cast member. It has their character name, along with actor id and movie id.

**Syntax:**

Create table Movie\_Cast(actor\_id int, movie\_id int, role varchar(25), foreign key(actor\_id) references Actor(actor\_id), foreign key(movie\_id) references Movie(movie\_id));

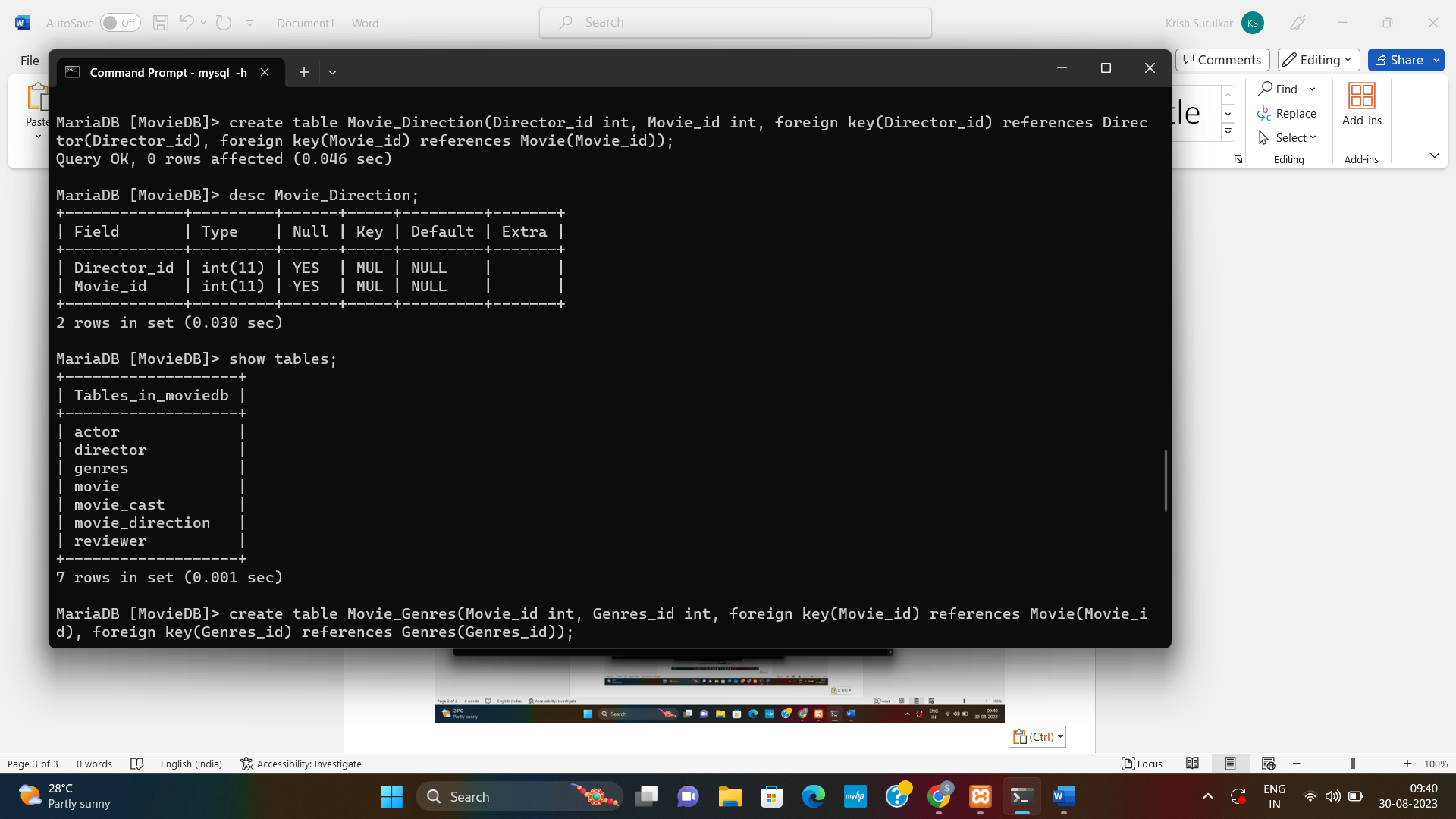


**CREATING TABLE – MOVIE\_DIRECTION**

The **Movie\_direction** table stores information like director id, movie id.

**Syntax:**

Create table Movie\_Direction(director\_id int, movie\_id int, foreign key(director\_id) references Director(director\_id), foreign key(movie\_id) references Movie(movie\_id));



**CREATING TABLE – MOVIE\_GENRES**

The **Movie\_genres** table stores information like movie id, genres id.

**Syntax:**

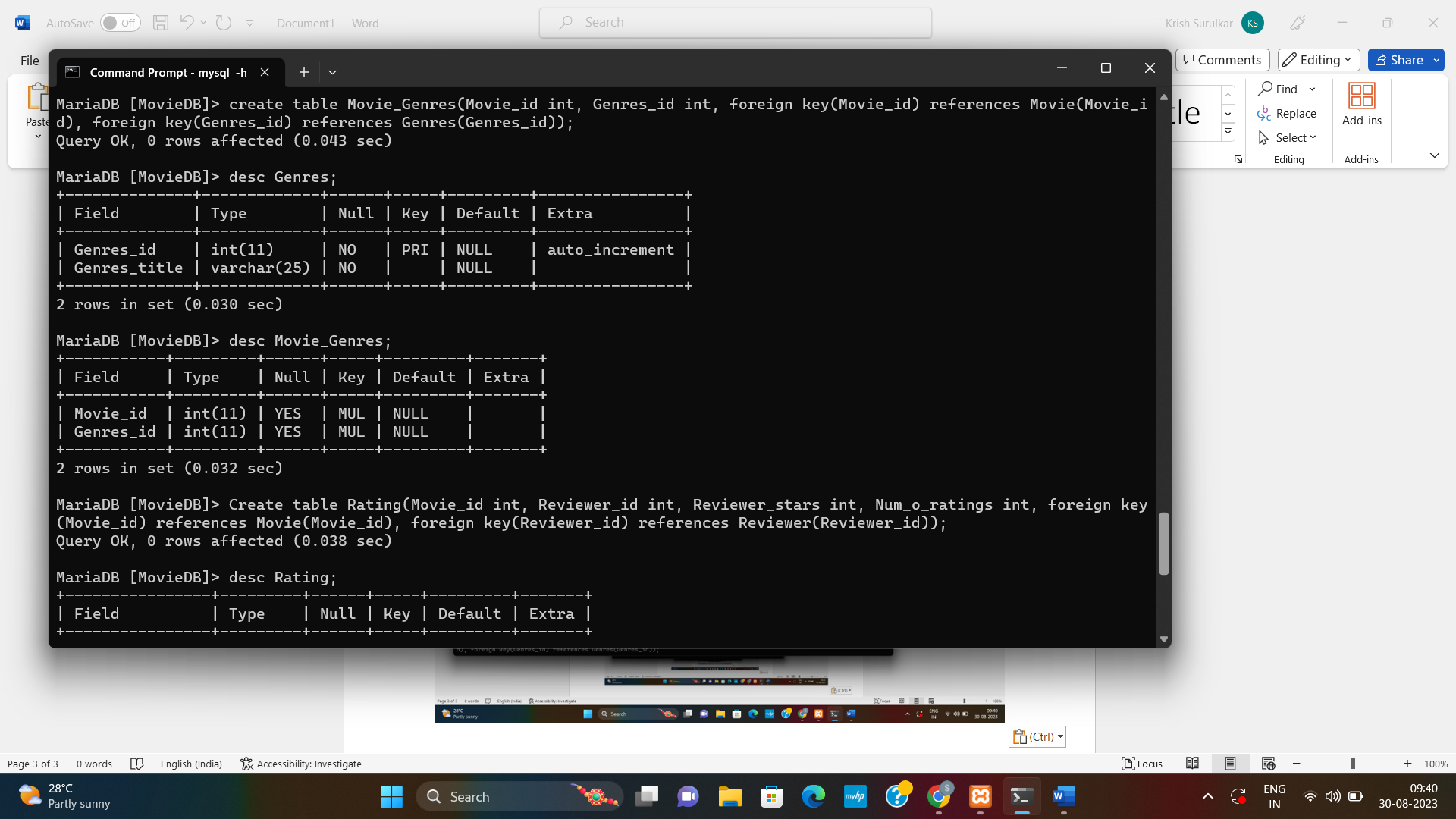
Create table Movie\_Genres(movie\_id int, genres\_id int, foreign key(movie\_id) references Movie(movie\_id), foreign key(genres\_id) references Genres(genres\_id));

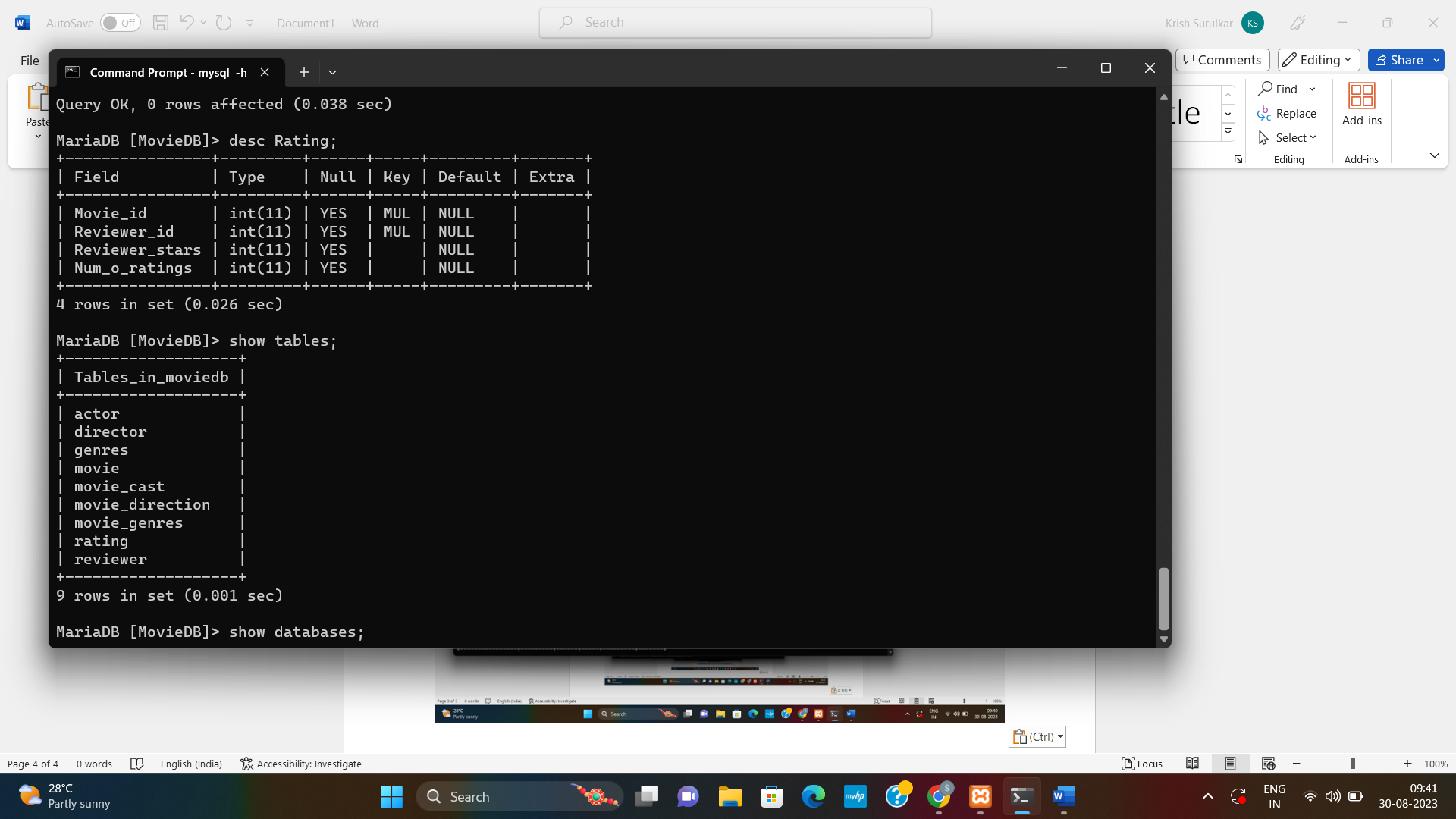
**CREATING TABLE – RATING**

The ratingtable stores information like movie id, reviewer id, reviewer stars and the number of ratings.

**Syntax:**

Create table Rating(movie\_id int, reviewer\_id int, reviewer\_stars int, num\_o\_ratings int, foreign key(movie\_id) references Movie(movie\_id), foreign key(reviewer\_id) references Reviewer(reviewer\_id));





After creation of database and all the tables of the Movie database, insert rows into all the tables.

**INSERTING VALUES IN MOVIE TABLE**

**Syntax:**

Insert into movie values(101,’Mary Kom’, 2014, 122,’Hindi’);

Insert into movie values(102,’Seven Samurai’, 1954, 207,’Japanese’);

Insert into movie values(103,’Anandi Gopal’, 2019, 134,’Marathi’);

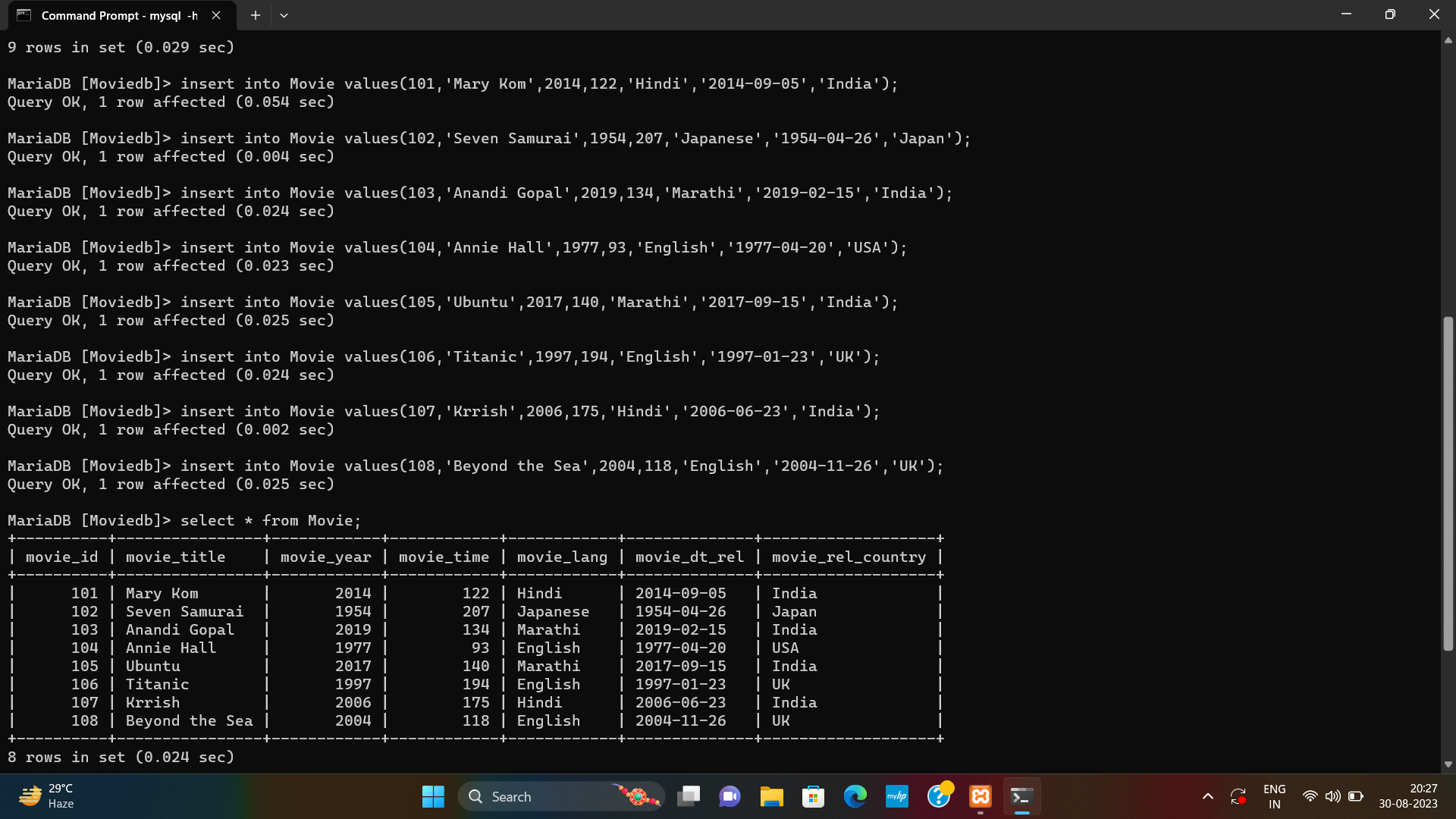
Insert into movie values(104,’Annie Hall’, 1977, 93,’English’);

Insert into movie values(105,’Ubuntu’, 2017, 140,’Marathi’);

Insert into movie values(106,’Titanic’, 1997, 194,’English’);

Insert into movie values(107,’Krrish’, 2006, 175,’Hindi’);

Insert into movie values(108,’Beyond the Sea’, 2004, 118,’English’);



**INSERTING VALUES IN DIRECTOR TABLE**

**Syntax:**

Insert into director values(201,’Omung’, ‘Kumar’);

Insert into director values(202,’Akira’, ‘Kurosawa’);

Insert into director values(203,’Sameer’, ‘Vidwans’);

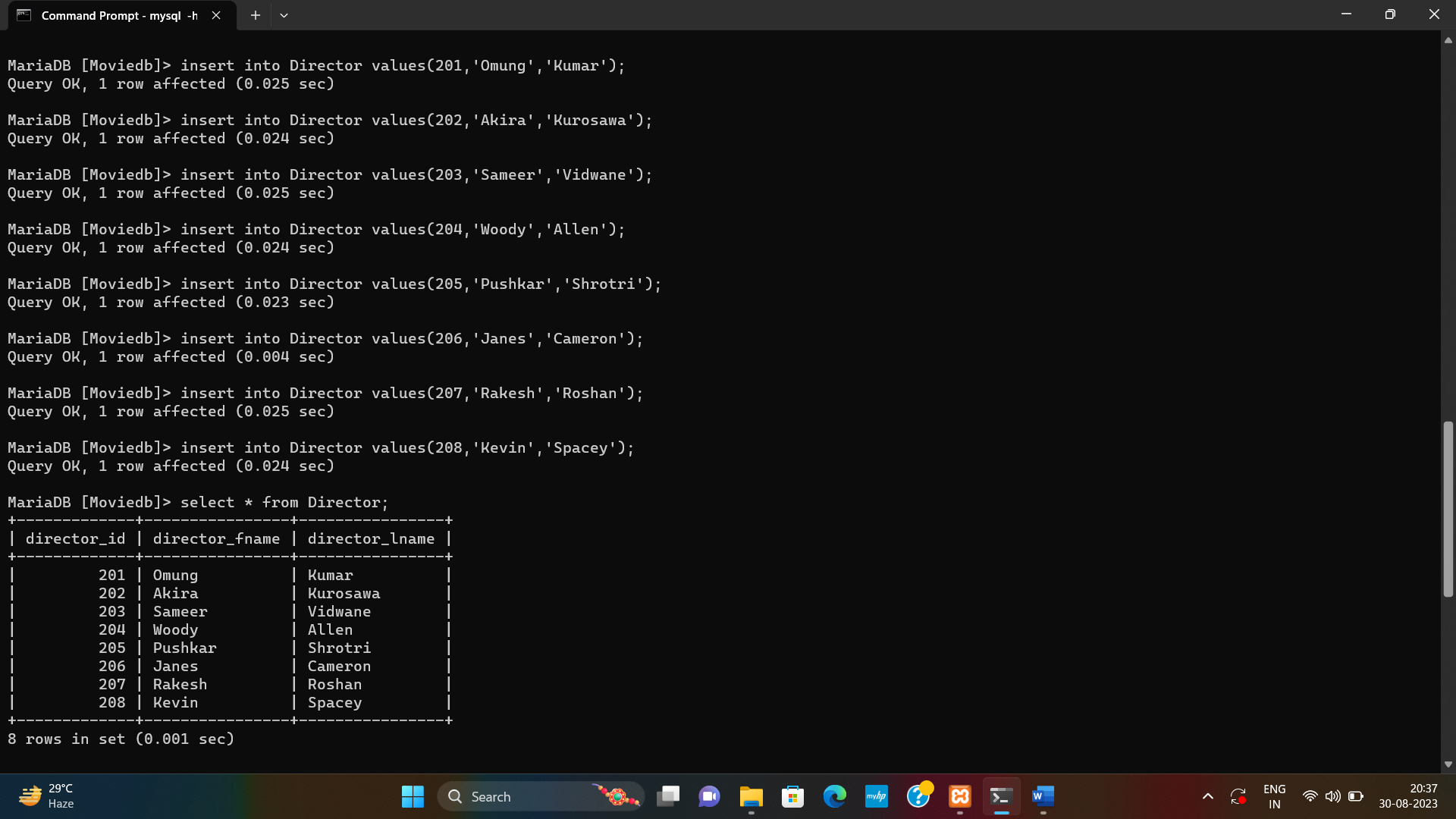
Insert into director values(204,’Woody’, ‘Allens’);

Insert into director values(205,’Pushkar’, ‘Shrotri’);

Insert into director values(206,’Janes’, ‘Cameron’);

Insert into director values(207,’Rakesh’, ‘Roshan’);

Insert into director values(208,’Kevin’, ‘Spacey’);



**INSERTING VALUES IN ACTOR TABLE**

**Syntax:**

Insert into Actor values(301,’Priyanka’, ‘Chopra’,’F’);

Insert into Actor values(302,’Takashi’, ‘Shimura’,’M’);

Insert into Actor values(303,’Bhagyashree’, ‘Milind’,’F’);

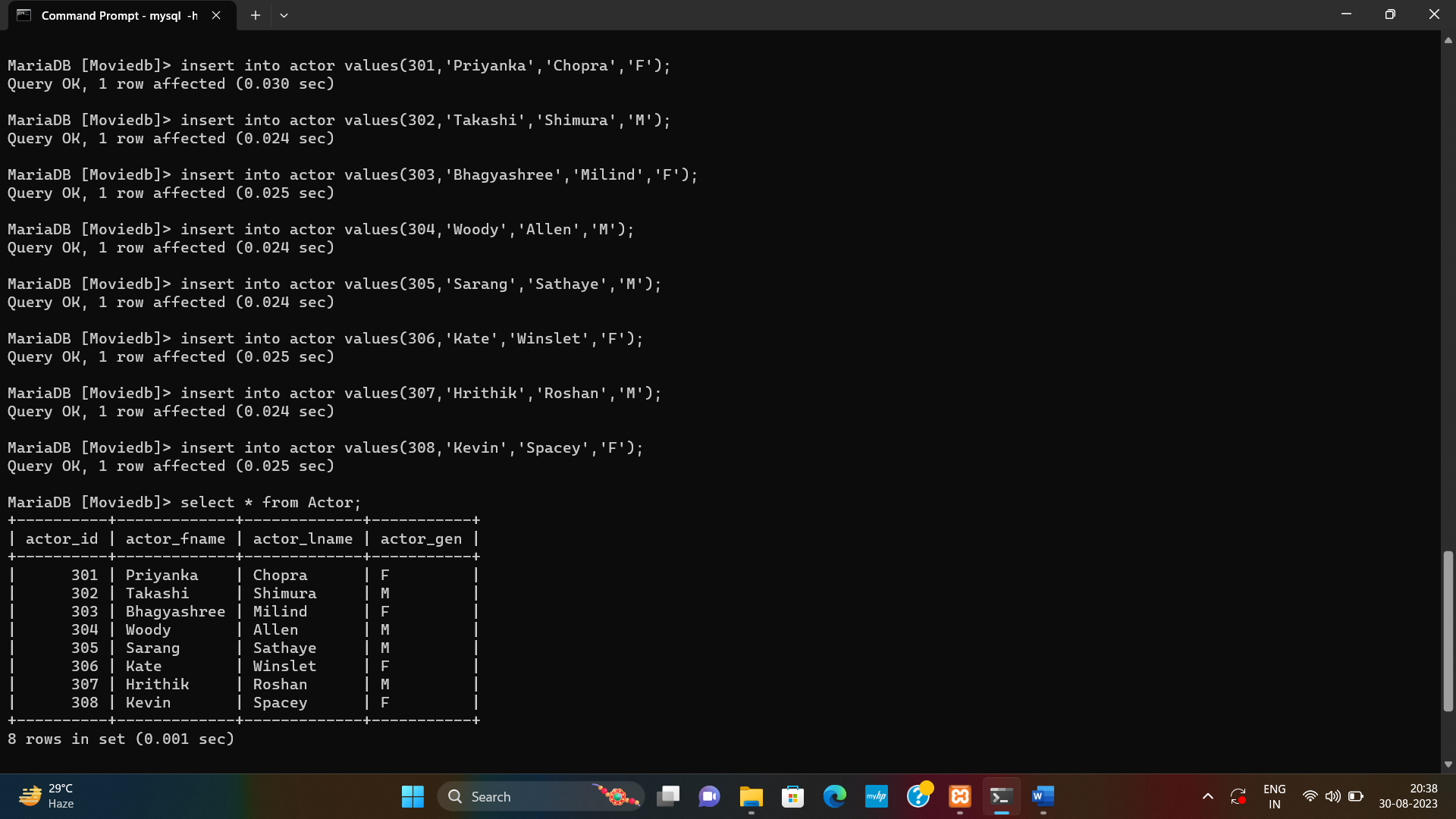
Insert into Actor values(304,’Woody’, ‘Allen’,’M’);

Insert into Actor values(305,’Sarang’, ‘Sathaye’,’M’);

Insert into Actor values(306,’Kate’, ‘Winslet’,’F’);

Insert into Actor values(307,’Hrithik’, ‘Roshan’,’M’);

Insert into Actor values(308,’Kevin’, ‘Spacey’,’F’);



**INSERTING VALUES IN REVIEWER TABLE**

**Syntax:**

Insert into Reviewer values(401,’Sana Sayed’);

Insert into Reviewer values(402,’Solid Gold’);

Insert into Reviewer values(403,’Vivekrevan Shete’);

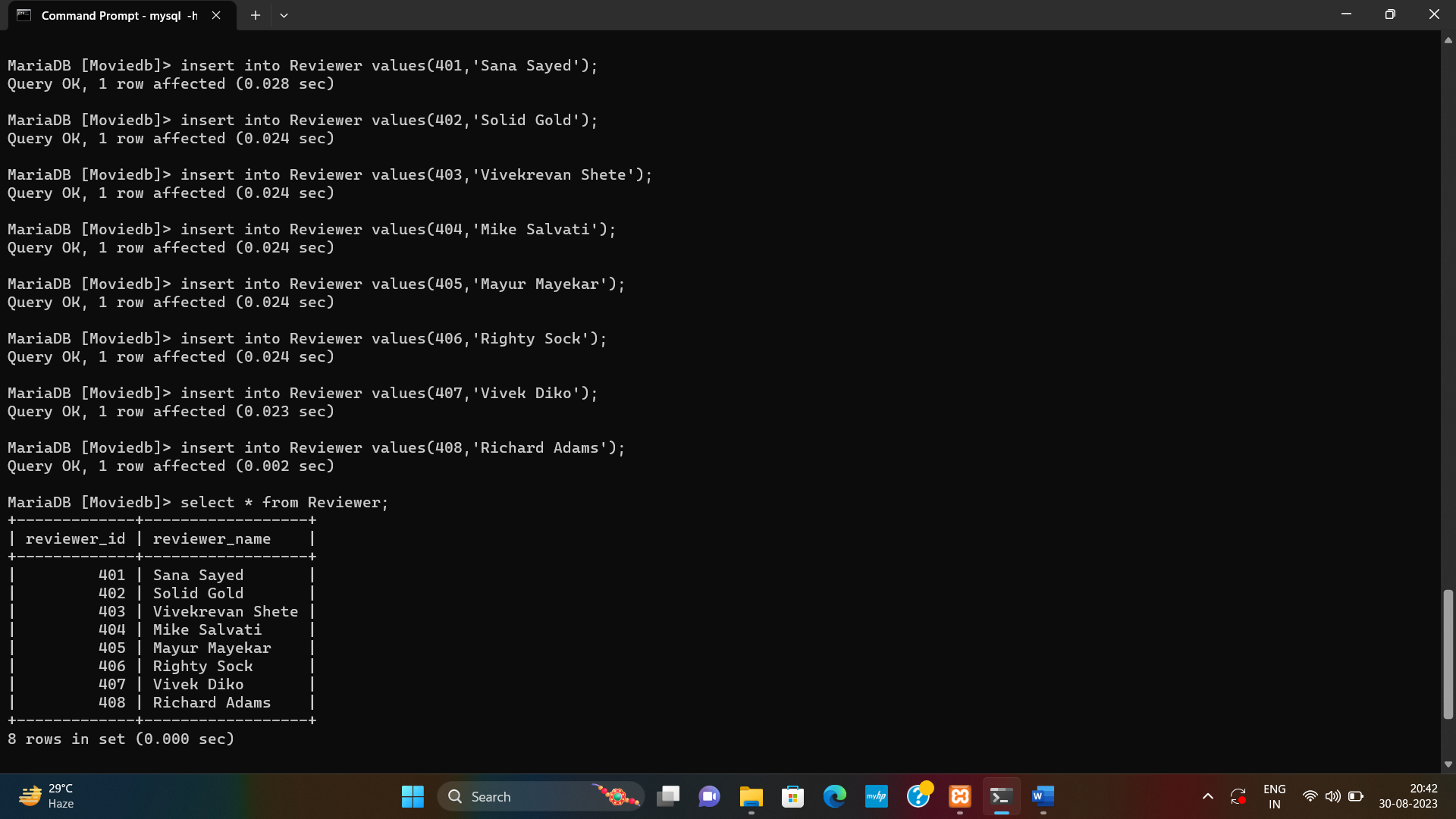
Insert into Reviewer values(404,’Mike Salvati’);

Insert into Reviewer values(405,’Mayur Mayekar’);

Insert into Reviewer values(406,’Righty Sock’);

Insert into Reviewer values(407,’Vivek Diko’);

Insert into Reviewer values(408,’Richard Adams’);



**INSERTING VALUES IN GENRES TABLE**

**Syntax:**

Insert into Genres values(501,’Biography’);

Insert into Genres values(502,’Action’);

Insert into Genres values(503,’Biography’);

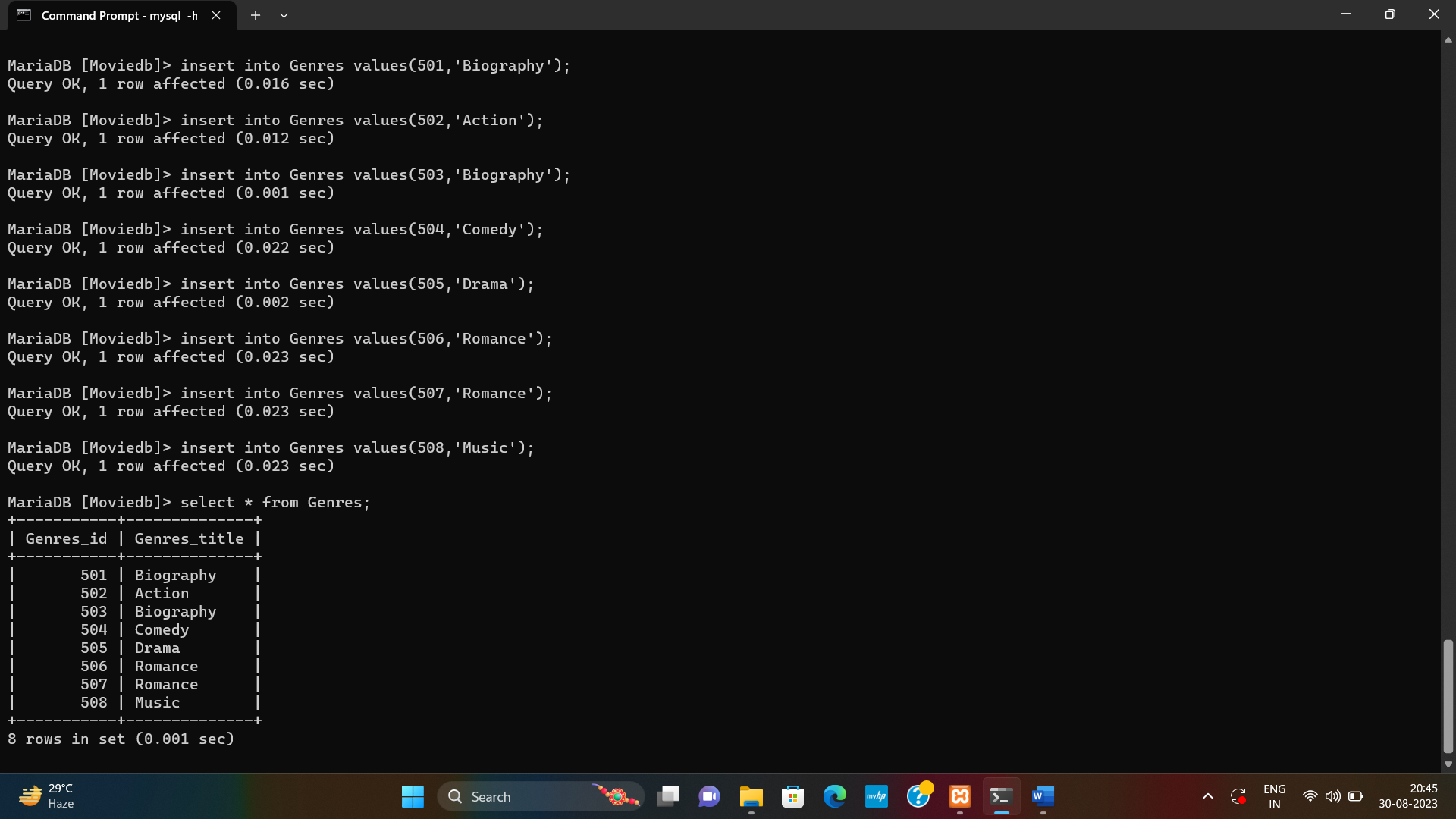
Insert into Genres values(504,’Comedy’);

Insert into Genres values(505,’Drama’);

Insert into Genres values(506,’Romance’);

Insert into Genres values(507,’Romance’);

Insert into Genres values(508,’Music’);



**INSERTING VALUES IN MOVIE\_DIRECTION TABLE**

**Syntax:**

Insert into Movie\_Direction values(201,101);

Insert into Movie\_Direction values(202,102);

Insert into Movie\_Direction values(203,103);

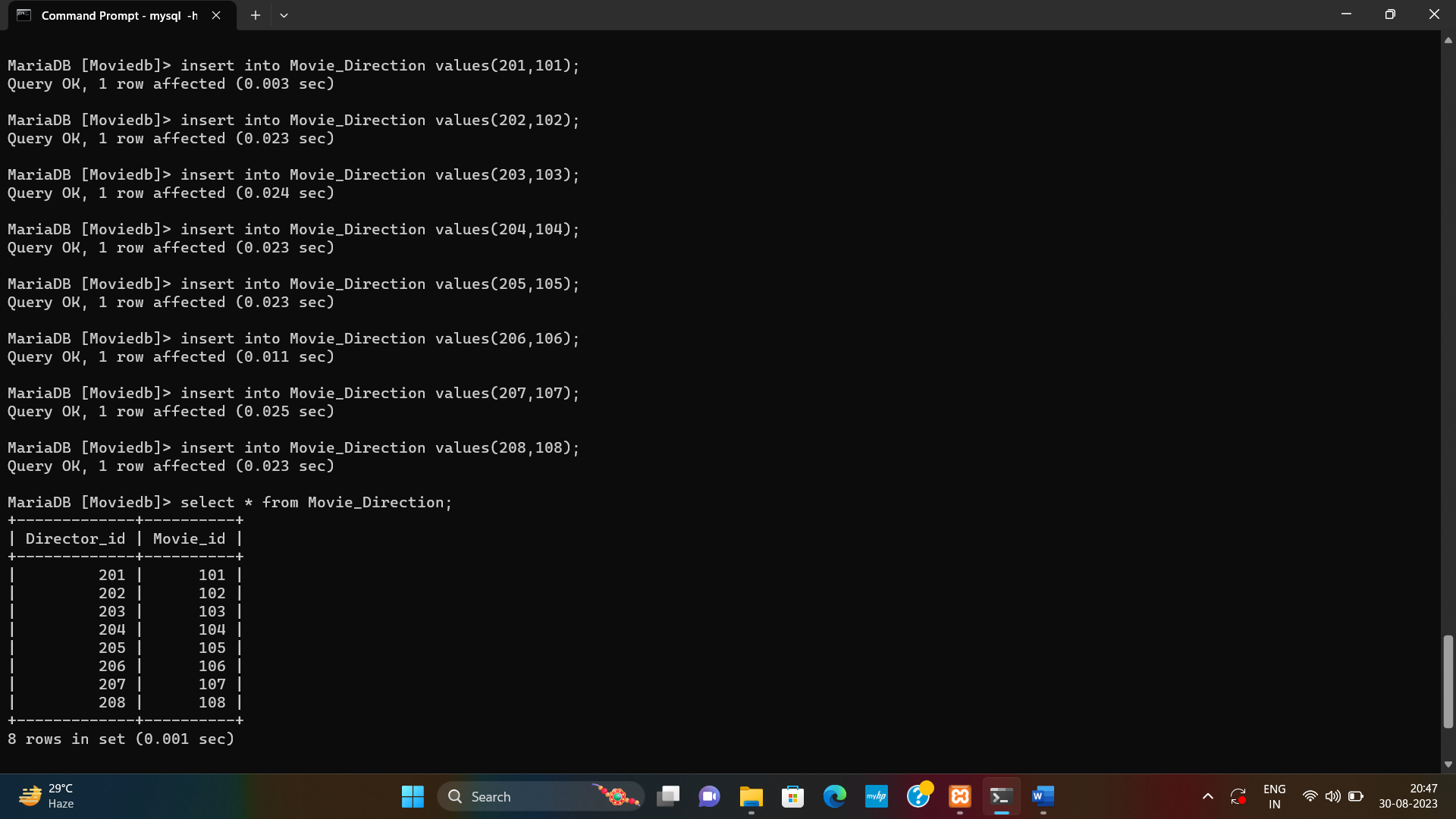
Insert into Movie\_Direction values(204,104);

Insert into Movie\_Direction values(205,105);

Insert into Movie\_Direction values(206,106);

Insert into Movie\_Direction values(207,107);

Insert into Movie\_Direction values(208,108);



**INSERTING VALUES IN MOVIE\_GENRES TABLE**

**Syntax:**

Insert into Movie\_Genres values(101, 501);

Insert into Movie\_Genres values(102, 502);

Insert into Movie\_Genres values(103, 503);

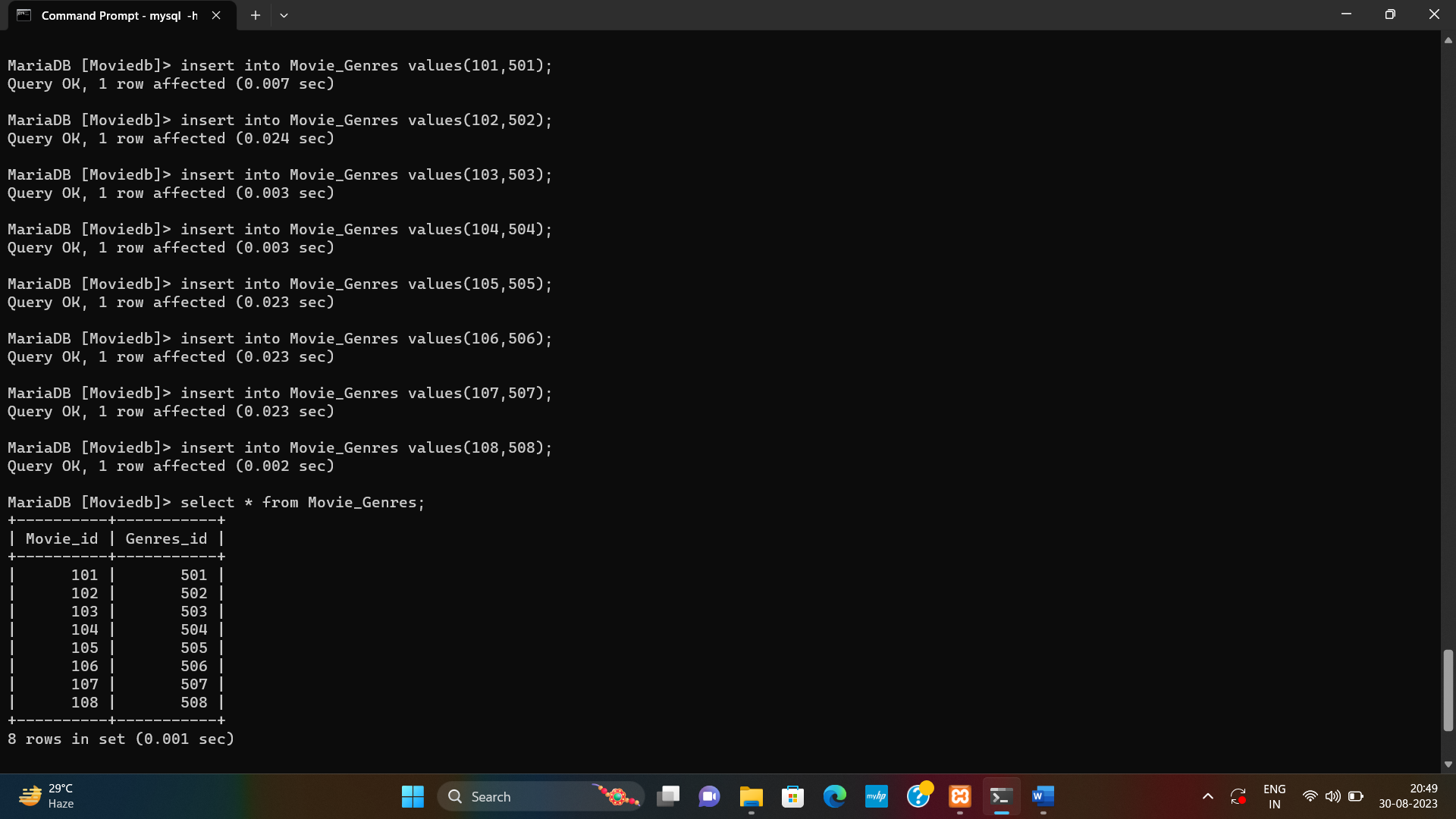
Insert into Movie\_Genres values(104,504);

Insert into Movie\_Genres values(105,505);

Insert into Movie\_Genres values(106,506);

Insert into Movie\_Genres values(107,507);

Insert into Movie\_Genres values(108,508);



**INSERTING VALUES IN MOVIE\_CAST TABLE**

**Syntax:**

Insert into Movie\_Cast values(301, 101,’Mary Kom’);

Insert into Movie\_Cast values(302, 102,’Kamber Shimada’);

Insert into Movie\_Cast values(303, 103,’Anandi Gopal’);

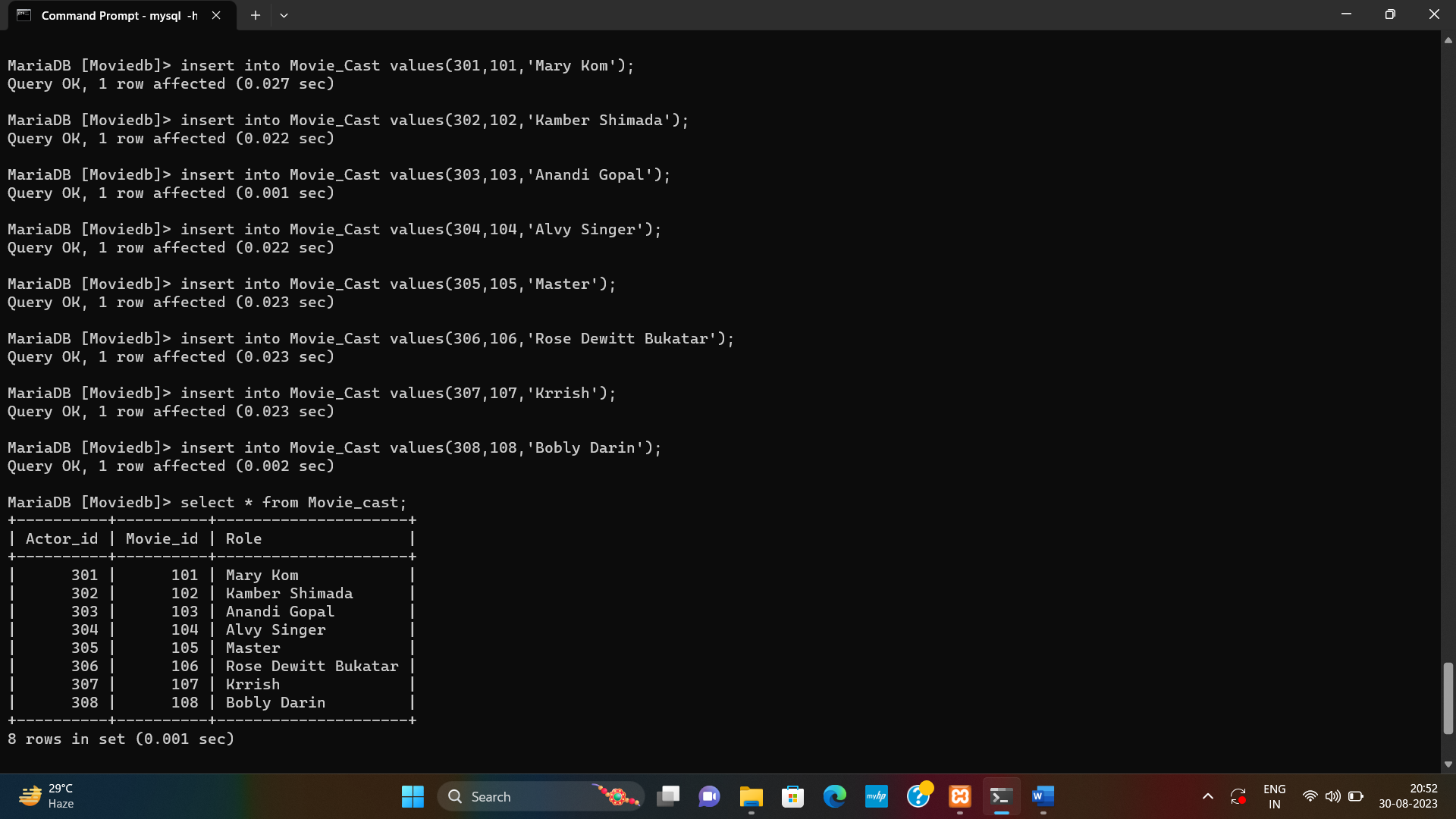
Insert into Movie\_Cast values(304, 104,’Alvy Singer’);

Insert into Movie\_Cast values(305, 105,’Master’);

Insert into Movie\_Cast values(306, 106,’Rose Dewitt Bukatar’);

Insert into Movie\_Cast values(307, 107,’Krrish’);

Insert into Movie\_Cast values(308, 108,’Bobly Darin’);



**INSERTING VALUES IN RATING TABLE**

**Syntax:**

Insert into Rating values(101, 401,8,700101);

Insert into Rating values(102, 402,9,85000);

Insert into Rating values(103, 403,9,85012);

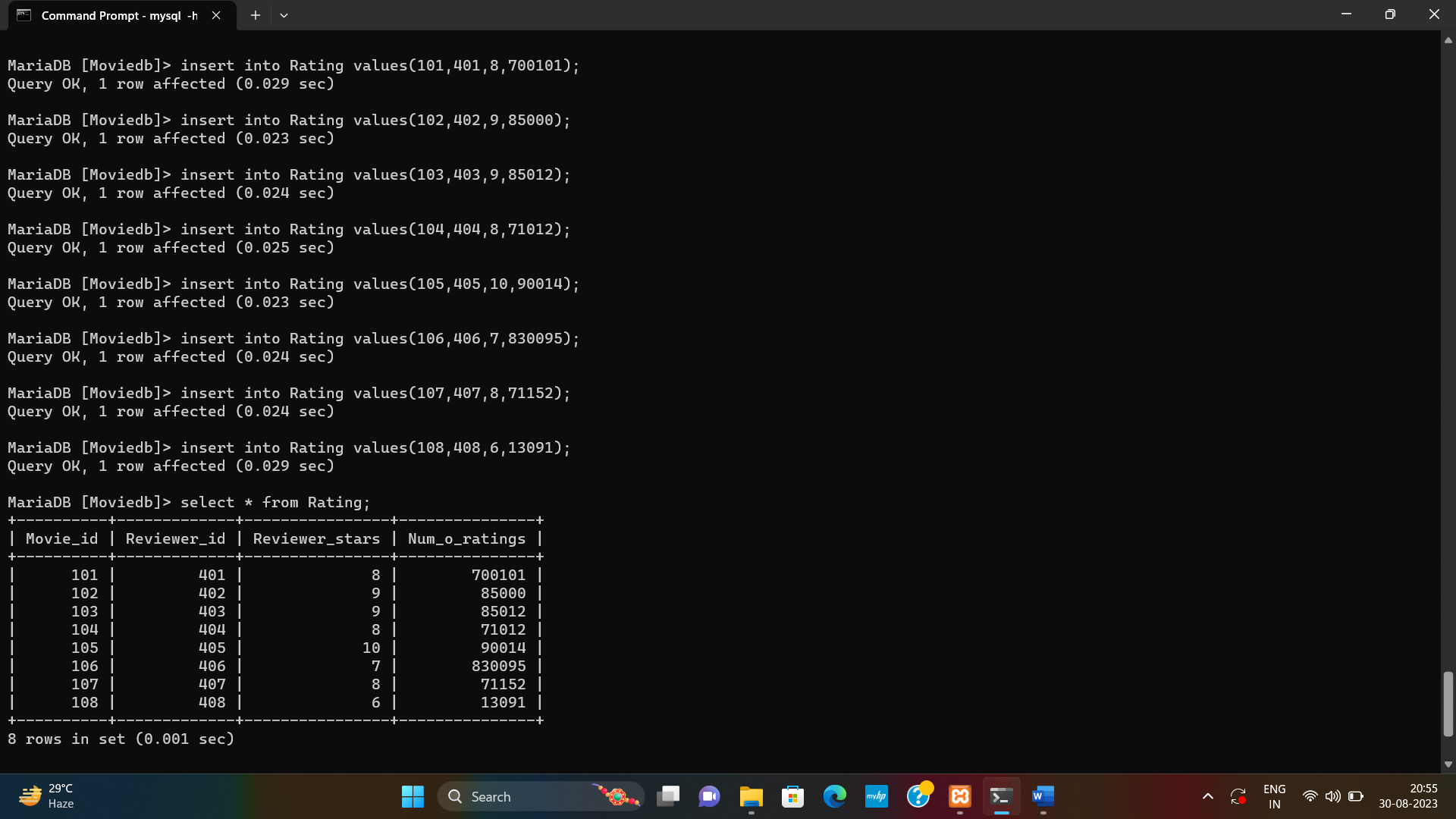
Insert into Rating values(104, 404,8,71012);

Insert into Rating values(105, 405,10,90014);

Insert into Rating values(106, 406,7,830095);

Insert into Rating values(107, 407,8,71152);

Insert into Rating values(108, 408,6,13091);



**QUERIES FOR PRACTICE**

1. **Write a SQL query to find all reviewers whose ratings contain a NULL value. Return reviewer name.**

Select reviewer\_name from reviewer inner join rating using(rev\_id) where reviewer\_stars is null;

Or

Select r.reviewer\_name from reviewer r inner join rating ra on r.reviewer\_id=ra.reviewer\_id where reviewer\_stars is null;

1. **write a SQL query to find out who was cast in the movie 'Annie Hall'. Return actor first name, last name and role.**

Select a.Actor\_fname, a.Actor\_lname, mc.role from Actor a inner join movie\_cast mc using(Actor\_id) inner join movie mo using(Movie\_id) where mo.movie\_title=’Annie Hall’;

Or

Select a.Actor\_fname, a.Actor\_lname, mc.role from Actor a inner join movie\_cast mc on a.actor\_id=mc.actor\_id inner join movie mo on mc.movie\_id=mo.movie\_id where mo.movie\_title=’Annie Hall’;

1. **write a SQL query to find the director who directed a movie that featured a role in 'Master'. Return director first name, last name and movie title.**

Select d.director\_fname, d.director\_lname, mo.movie\_title from director d inner join movie\_direction md using(director\_id) inner join movie mo using(movie\_id) inner join movie\_cast mc using(movie\_id) where mc.role=’Master’;

Or

Select d.director\_fname, d.director\_lname, mo.movie\_title from director d inner join movie\_direction md on d.director\_id=md.director\_id inner join movie mo on md.movie\_id=mo.movie\_id inner join movie\_cast mc on mo.movie\_id=mc.movie\_id where mc.role=’Master’;

1. **write a SQL query to find the director of a movie that cast a role as Krrish. Return director first name, last name and movie title.**

Select d.director\_fname, d.director\_lname, mo.movie\_title from director d inner join movie\_direction md using(director\_id) inner join movie mo using(movie\_id) inner join movie\_cast mc using(movie\_id) where mc.role=’Krrish’;

Or

Select d.director\_fname, d.director\_lname, mo.movie\_title from director d inner join movie\_direction md on d.director\_id=md.director\_id inner join movie mo on md.movie\_id=mo.movie\_id inner join movie\_cast mc on mo.movie\_id=mc.movie\_id where mc.role=’Krrish’;

1. **write a SQL query to find out which actors have not appeared in any movies between 1990 and 2000 (Begin and end values are included.). Return actor first name, last name, movie title and release year.**

Select a.actor\_fname, a.actor\_lname, mo.movie\_title, mo.movie\_year from actor a inner join movie\_cast mc using(actor\_id) inner join movie mo using(movie\_id) where movie\_year not between 1990 and 2000;

Or

Select a.actor\_fname, a.actor\_lname, mo.movie\_title, mo.movie\_year from actor a inner join movie\_cast mc on a.actor\_id=mc.actor\_id inner join movie mo on mo.movie\_id=mc.movie\_id where movie\_year not between 1990 and 2000;

1. **write a SQL query to find the directors who have directed films in a variety of genres. Group the result set on director first name, last name and generic title. Sort the result-set in ascending order by director first name and last name. Return director first name, last name and number of genres movies.**

Select d.director\_fname, d.director\_lname, g.genres\_title, count(g.genres\_title) from director d inner join movie\_direction md using(director\_id) inner join movie\_genres mg using(movie\_id) inner join genres g using(genres\_id) group by d.director\_fname, d.director\_lname, g.genres\_title order by d.director\_fname, d.director\_lname;

1. **write a SQL query to find the movies with year and genres. Return movie title, movie year and generic title.**

Select mo.movie\_title, mo.movie\_year, g.genres\_title from movie mo inner join movie\_genres mg using(movie\_id) inner join genres g using(genres\_id);

or

Select mo.movie\_title, mo.movie\_year, g.genres\_title from movie mo inner join movie\_genres mg on mo.movie\_id=mg.movie\_id inner join genres g on mg.genres\_id=g.genres\_id;

1. **write a SQL query to find all the movies with year, genres, and name of the director.**

Select mo.movie\_year, g.genres\_title, d.director\_fname, d.director\_lname from movie mo inner join movie\_genres mg using(movie\_id) inner join genres g using(genres\_id) inner join movie\_direction md using(movie\_id) inner join director d using(director\_id);

1. **write a SQL query to find the movies released before 1st January 1989. Sort the result-set in descending order by date of release. Return movie title, release year, date of release, duration, and first and last name of the director.**

Select mo.Movie\_title, mo.movie\_year, mo.movie\_dt\_rel, mo.movie\_time, d.director\_fname, d.director\_lname from movie mo inner join movie\_direction using(movie\_id) inner join director d using(director\_id) order by mo.movie\_dt\_rel;

1. **write a SQL query to calculate the average movie length and count the number of movies in each genre. Return genre title, average time and number of movies for each genre.**

Select g.genres\_title, avg(movie\_time), count(movie\_id) from movie mo inner join movie\_genres mg using(movie\_id) inner join genres g using(genres\_id) group by g.genres\_title;

1. **write a SQL query to find movies with the shortest duration. Return movie title, movie year, director first name, last name, actor first name, last name and role.**

Select mo.movie\_title, mo.movie\_year, d.director\_fname, d.director\_lname,a.actor\_fname,a.actor\_lname, mc.role from movie mo inner join movie\_direction md using(movie\_id) inner join director d using(director\_id) inner join movie\_cast mc using(movie\_id) inner join actor a using(actor\_id) where mo.movie\_time=(select min(movie\_time) from movie);

1. **write a SQL query to find the years in which a movie received a rating of 3 or 4. Sort the result in increasing order on movie year.**

Select mo.movie\_year from movie mo inner join rating r using(movie\_id) where r.reviewer\_stars in (7,8) order by mo.movie\_year;

1. **write a SQL query to get the reviewer name, movie title, and stars in an order that reviewer name will come first, then by movie title, and lastly by number of stars.**

Select r.reviewer\_name, mo.movie\_title,ra.reviewer\_stars from movie mo inner join rating ra using(movie\_id) inner join reviewer r using(reviewer\_id) order by r.reviewer\_name,mo.movie\_title,ra.reviewer\_stars;

1. **write a SQL query to find those movies that have at least one rating and received the most stars. Sort the result-set on movie title. Return movie title and maximum review stars.**

Select mo.movie\_title, max(reviewer\_stars) from movie mo inner join rating ra using(movie\_id) group by mo.movie\_title having max(reviewer\_stars)>0 order by mo.movie\_title;

1. **write a SQL query to find out which movies have received ratings. Return movie title, director first name, director last name and review stars.**

Select mo.movie\_title, d.director\_fname, d.director\_lname, r.reviewer\_stars from movie mo inner join movie\_direction md using(movie\_id) inner join director d using(director\_id) inner join rating r using(movie\_id) where reviewer\_star is not null;

1. **write a SQL query to find movies in which one or more actors have acted in more than one film. Return movie title, actor first and last name, and the role.**

Select mo.movie\_title, a.actor\_fname, a.actor\_lname, mc.role from movie mo inner join movie\_cast mc using(movie\_id) inner join actor a using(actor\_id) where a.actor\_id in (select actor\_id from movie\_cast group by actor\_id having count(\*)>=1);

1. **write a SQL query to find the actor whose first name is 'Priyanka' and last name is 'Chopra'. Return director first name, last name, movie title, actor first name and last name, role.**

Select d.director\_fname, d.director\_lname, mo.movie\_title, a.actor\_fname, a.actor\_lname, mc.role from actor a inner join movie\_cast mc using(actor\_id) inner join movie\_direction md using(movie\_id) inner join director d using(director\_id) inner join movie mo using(movie\_id) where actor\_fname=’Priyanka’ and actor\_lname=’Chopra’;

1. **write a SQL query to find for actors whose films have been directed by them. Return actor first name, last name, movie title and role.**

Select a.actor\_fname, a.actor\_lname, mo.movie\_title, mc.role from actor a inner join movie\_cast mc using(actor\_id) inner join movie\_direction md using(movie\_id) inner join director d using(director\_id) inner join movie mo using(movie\_id) where actor\_fname=director\_fname and actor\_lname=director\_lname;