MovieShows Database

## DATABASE MANAGEMENT STSTEM

A Database Management System (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, Constructing, manipulating and sharing databases among various users and applications.

# INTRODUCTION TO MOVIESHOWS

MovieShows database provide basic information about currently released and upcoming movies like title, description, cast, language, genre, etc. It also provides information about theatres that screened currently released movies that user requests and the user can book the tickets for the movies. It also gives the booking information of the shows booked by the user.

The database will have user reviews about the movies from their experiences.

To implement the above database, we will use **MySQL**.

## MYSQL

**SQL** is a database computer language designed for the retrieval and management of data in a relational database. **SQL** stands for **Structured Query Language**.

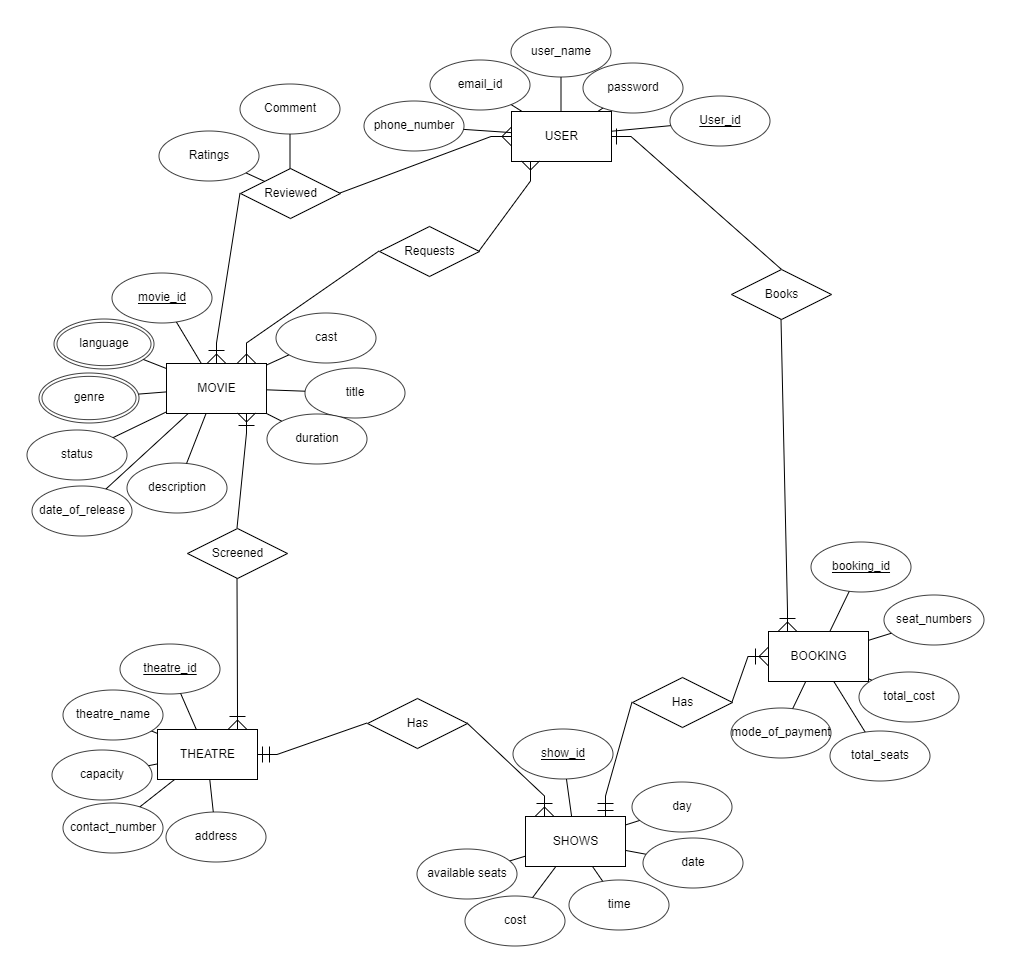
## MODULES IN DATABASE

1. USER: Requests the movies and book the tickets the movie.
2. MOVIE: Provide basic information about movies like title, date of release, description, etc.
3. THEATRE: Provide the information about theatre like theatre name, capacity, address, etc.
4. SHOW: Contain information about the shows shown in theatre.
5. BOOKING: Contain record of user bookings.

DESIGN OF DATABASE

The database has been developed using MySQL software which is queries oriented.

# ER-DAIGRAM



# SCHEMA DAIGRAM

USER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User\_id | User\_name | Password | Phone\_number | Email\_id |

MOVIE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Movie\_id | Title | Date\_of\_release | Duration | Description | Cast | Status |

THEATRE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Theatre\_id | Theatre\_name | Capacity | Contact\_number | Address |

SHOWS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Show\_id | Available\_seats | Cost | Day | Date | Time | T\_id |

BOOKING

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Booking\_id | Seat\_number | Total\_seats | Total\_cost | Mode\_of\_payment | U\_id | S\_id |

REVIEWED

|  |  |  |  |
| --- | --- | --- | --- |
| U\_id | M\_id | Comment | Rating |

REQUESTS

|  |  |
| --- | --- |
| U\_id | M\_id |

SCREENED

|  |  |
| --- | --- |
| M\_id | T\_id |

MOVIE\_GENRE

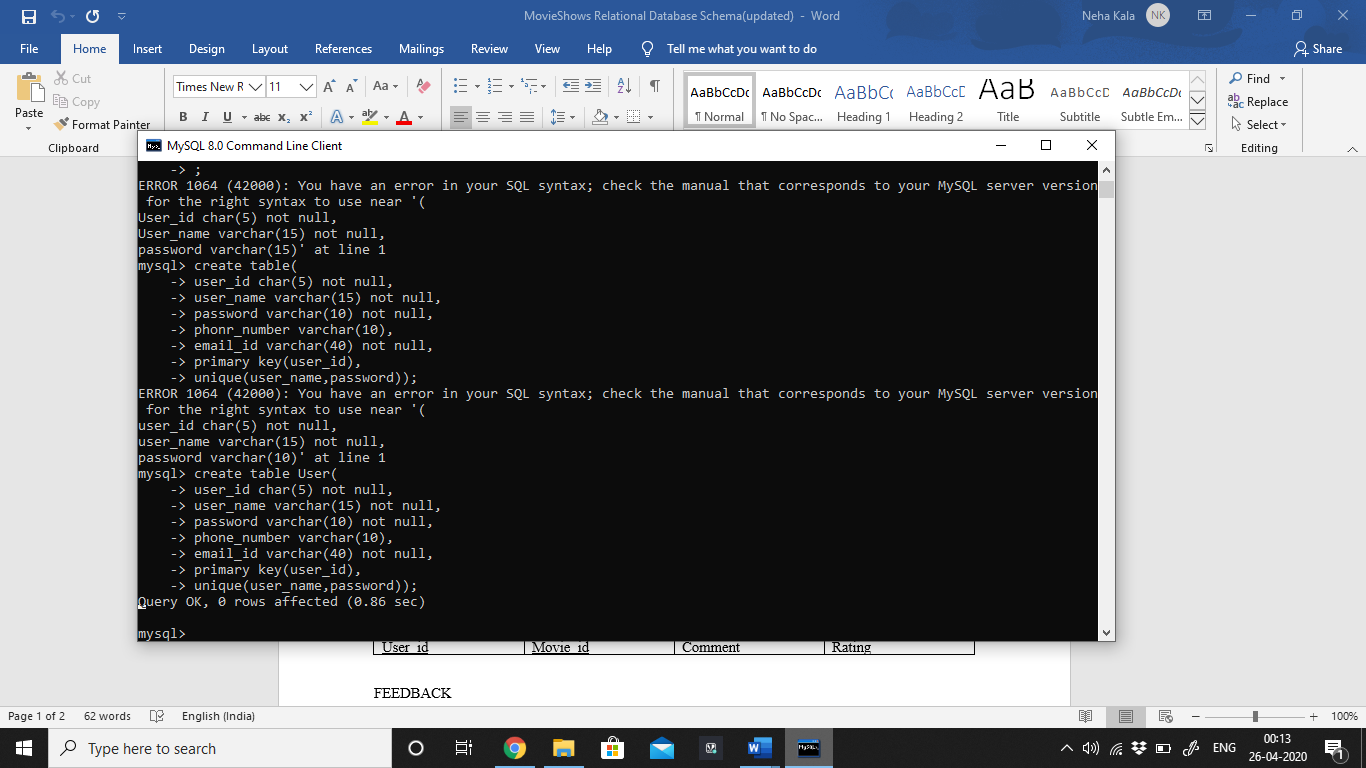
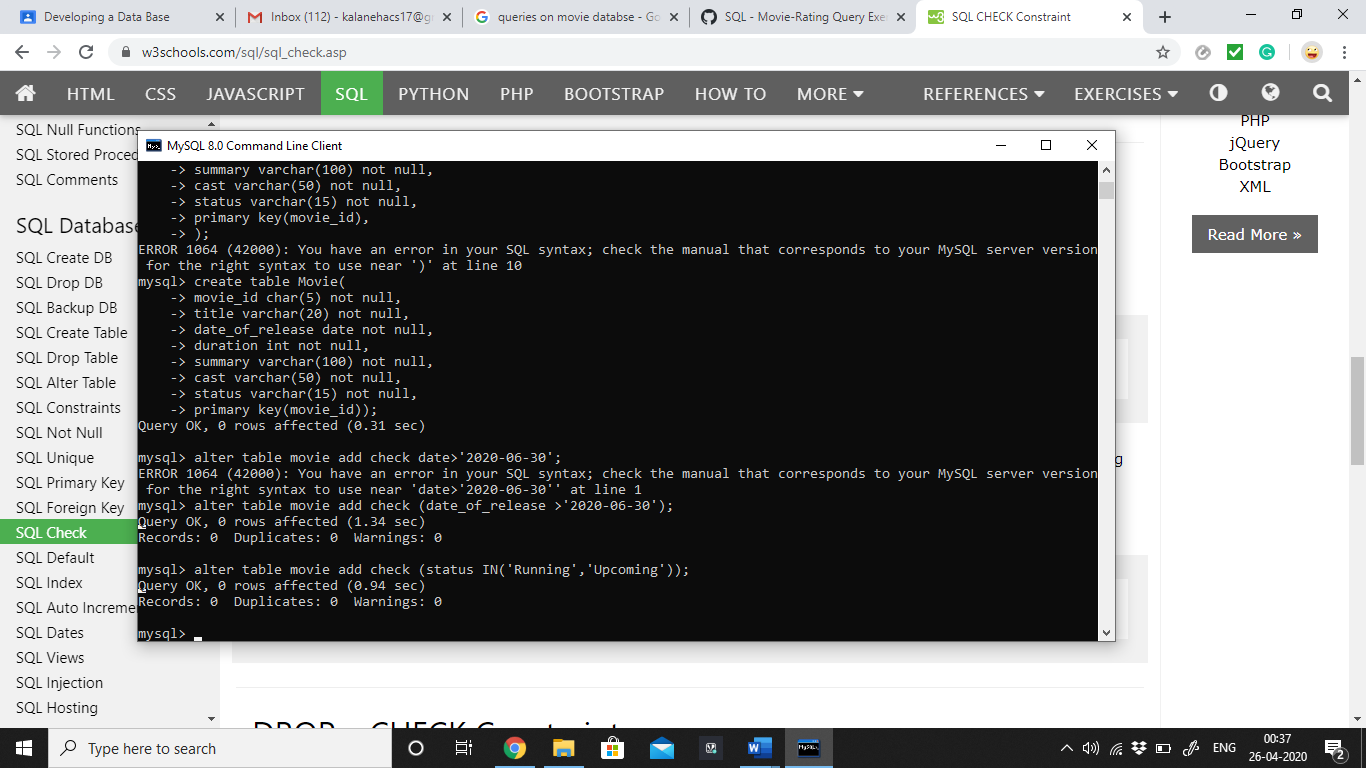
|  |  |
| --- | --- |
| M\_id | Genre |

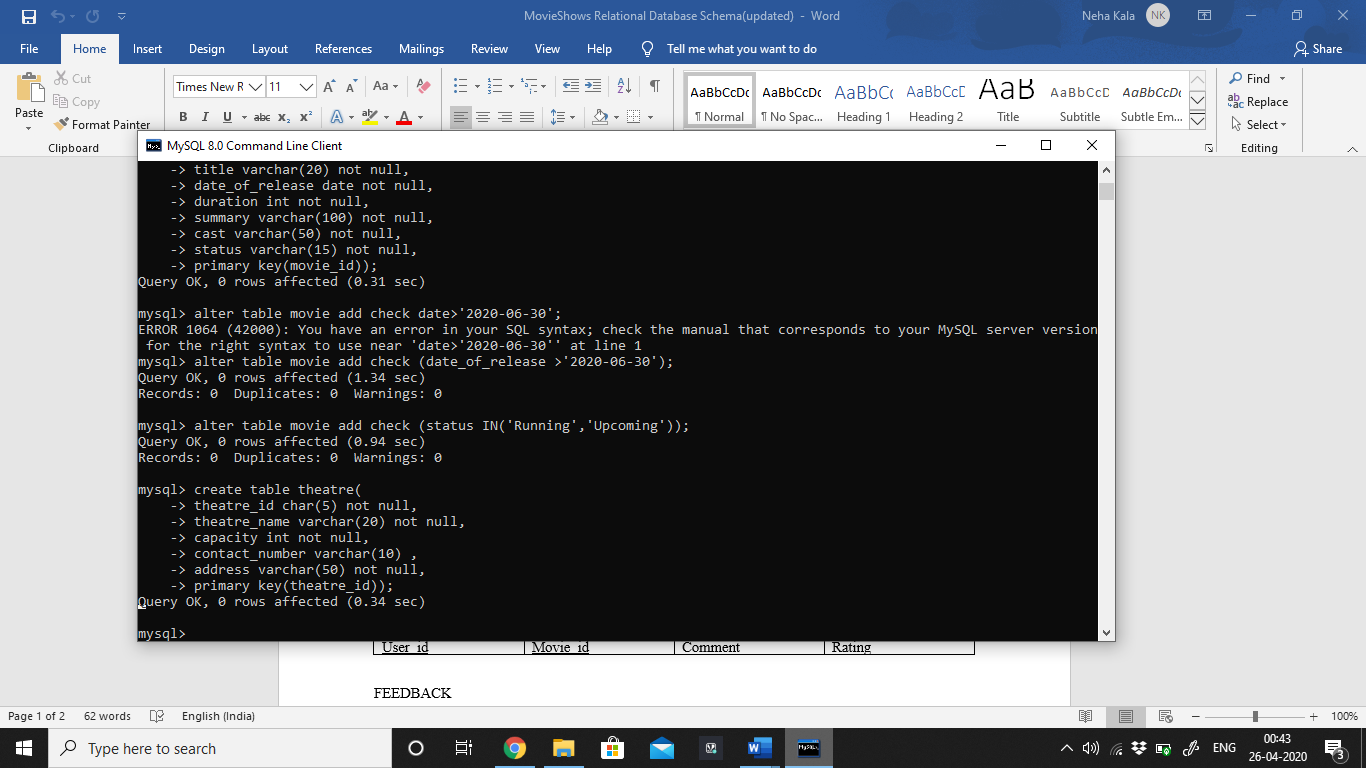
MOVIE\_LANGUAGE

|  |  |
| --- | --- |
| M\_id | Language |

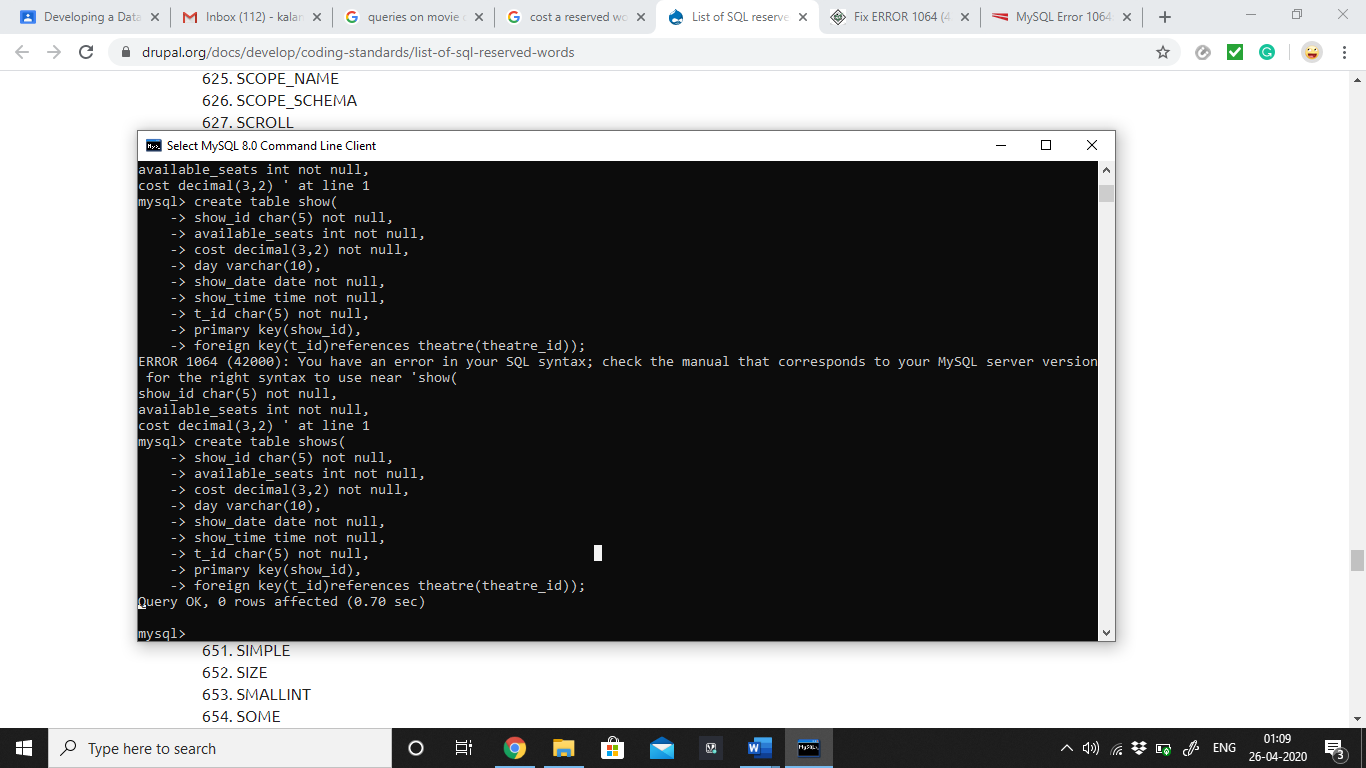
## **U\_id, T\_id ,S\_id and M\_id are foreign keys references to User, Theatre, Shows and Movie respectively.**

# CREATING TABLES

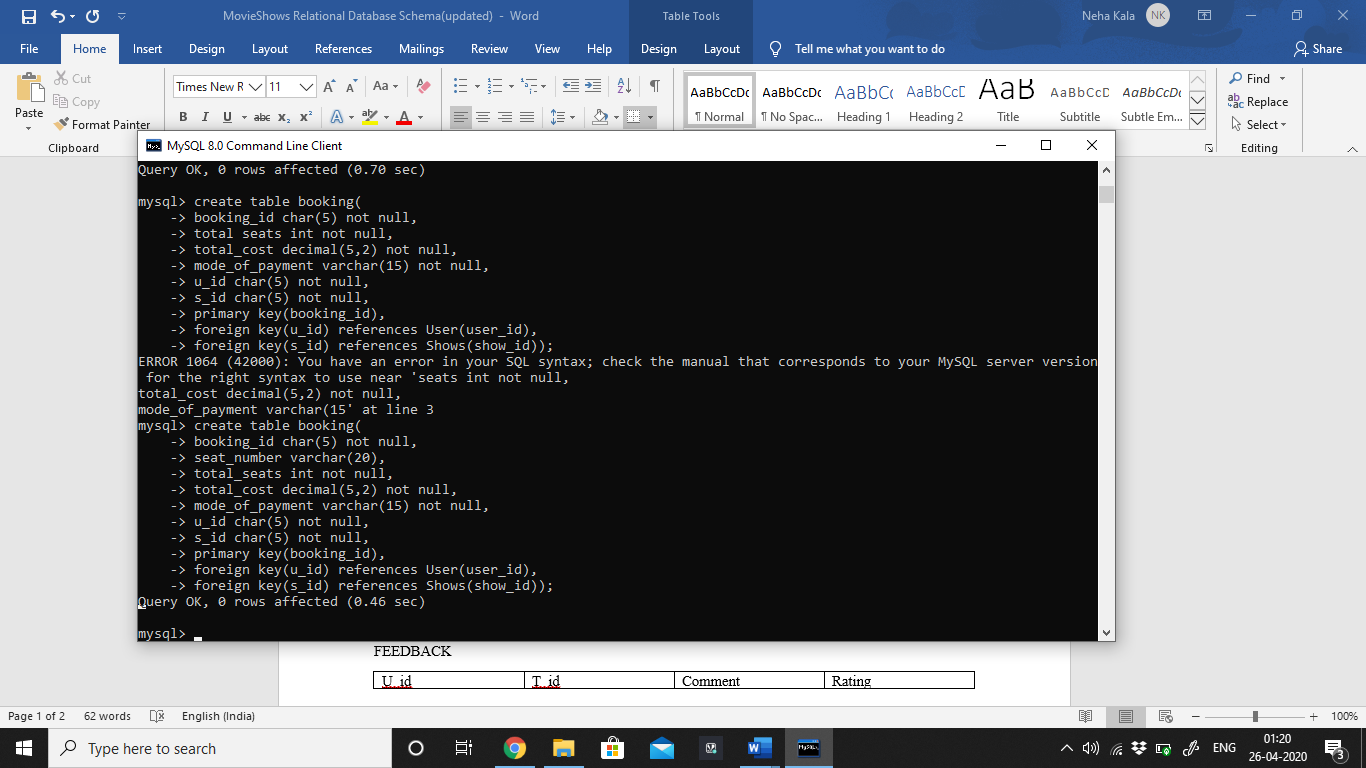
1. USER
2. MOVIE 
3. THEATRE



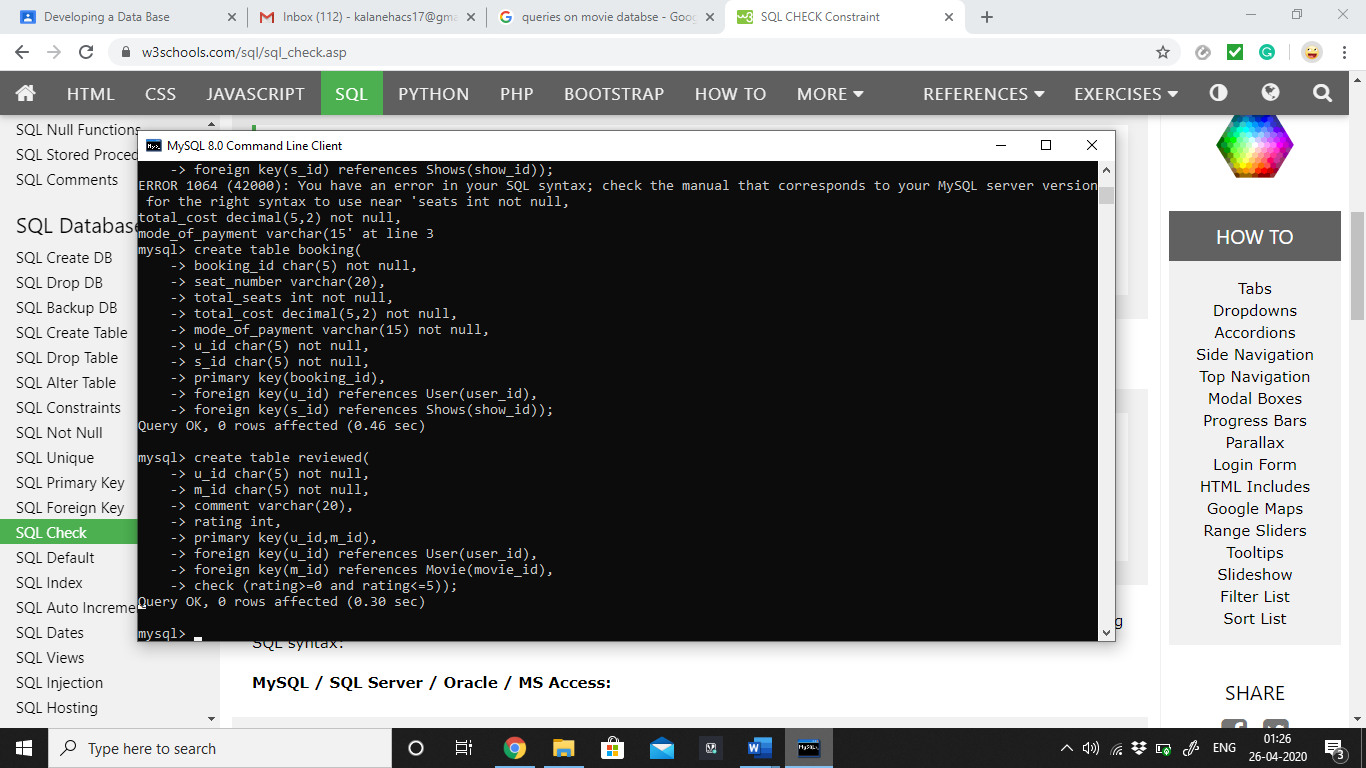
1. SHOWS



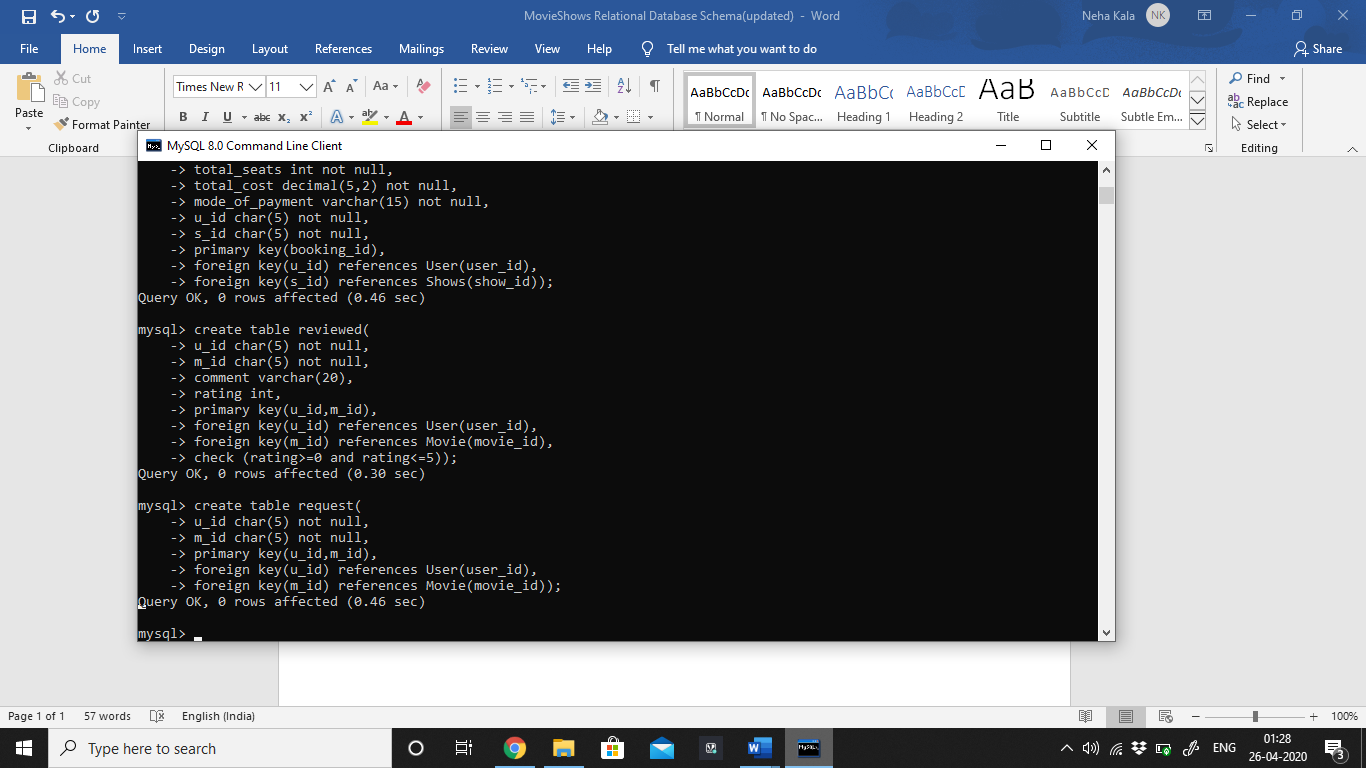
1. BOOKING



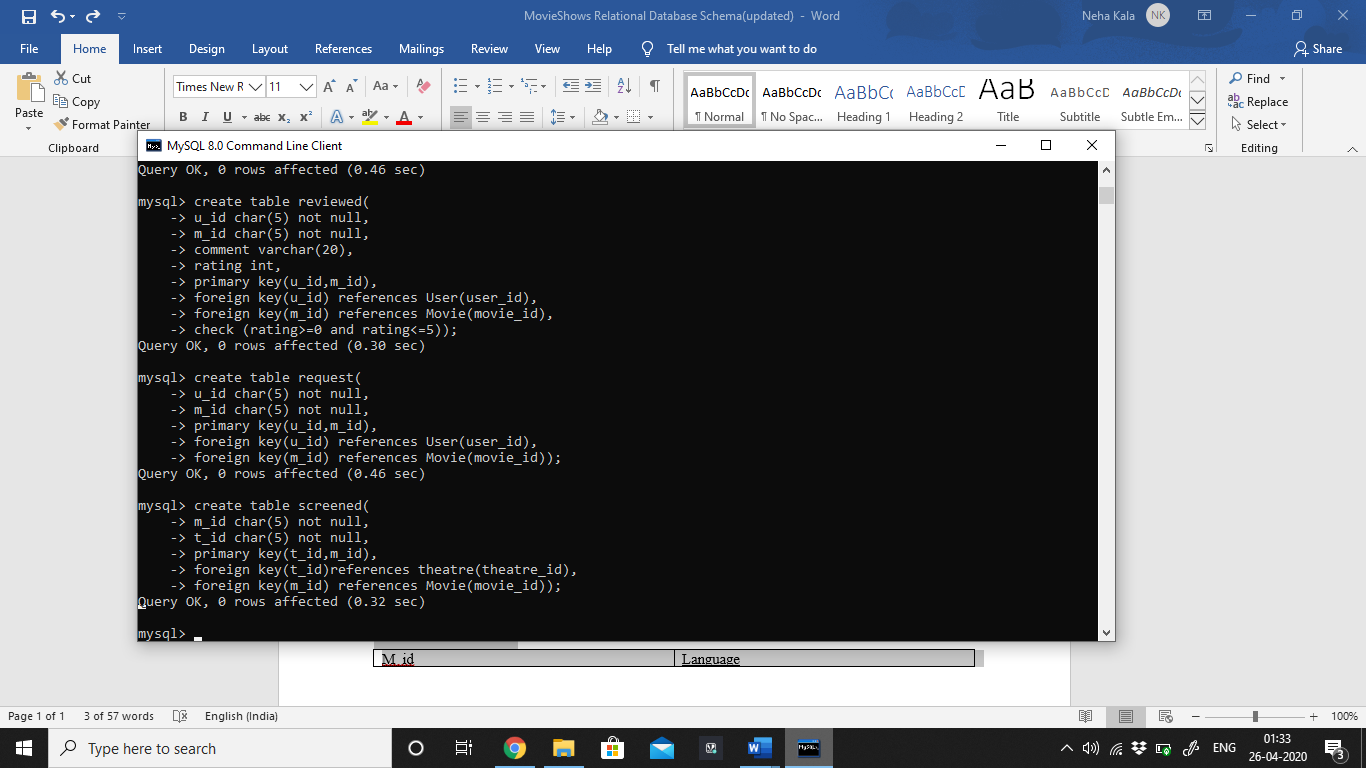
1. REVIEWED



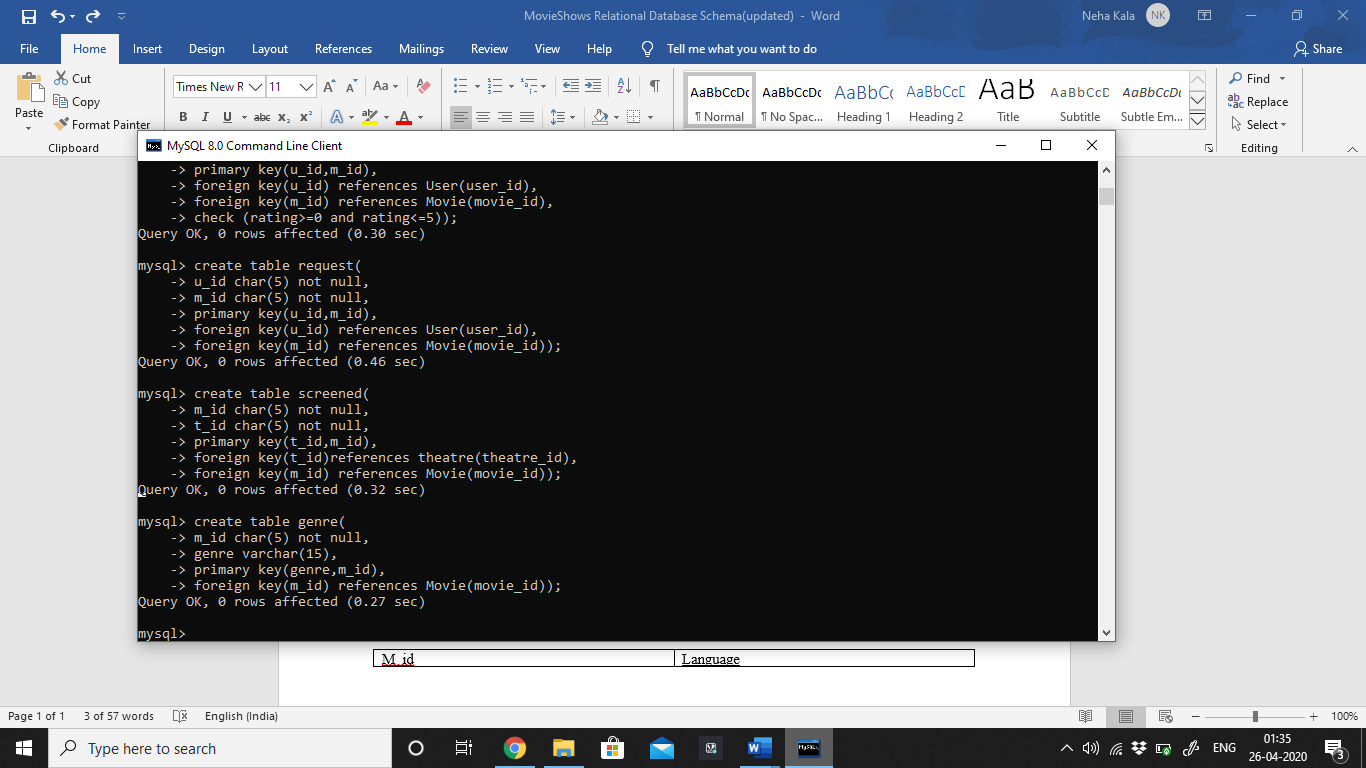
1. REQUEST



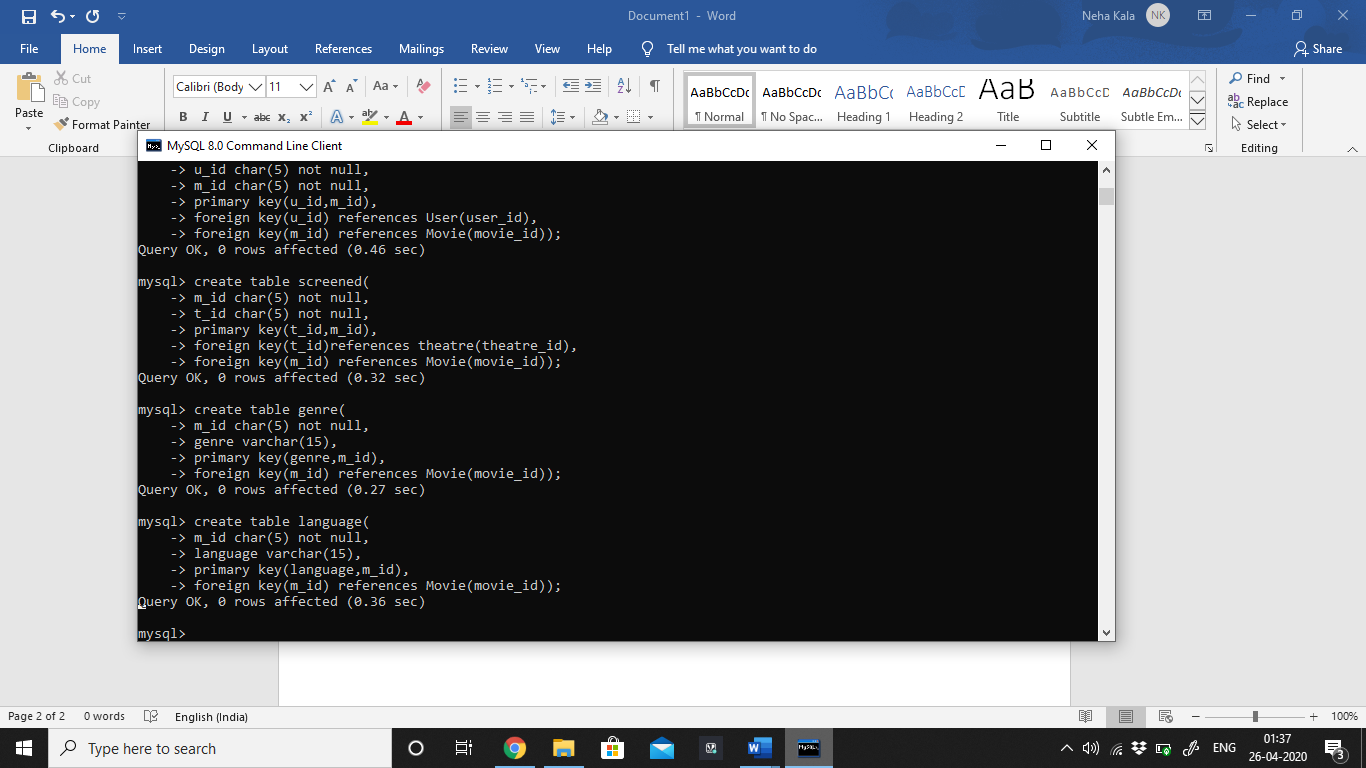
1. SCREENED



1. GENRE



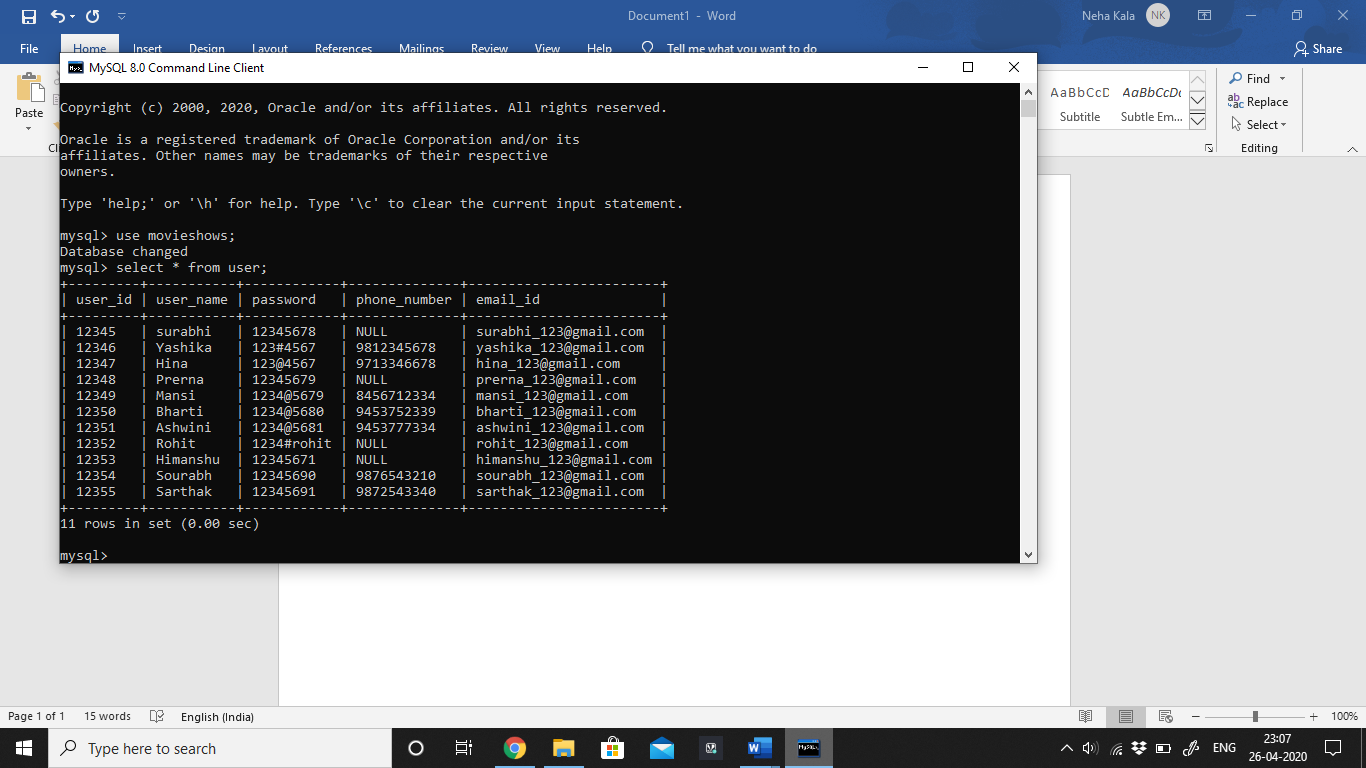
1. LANGUAGE



# INSERTION INTO TABLE: USING INSERT COMMAND

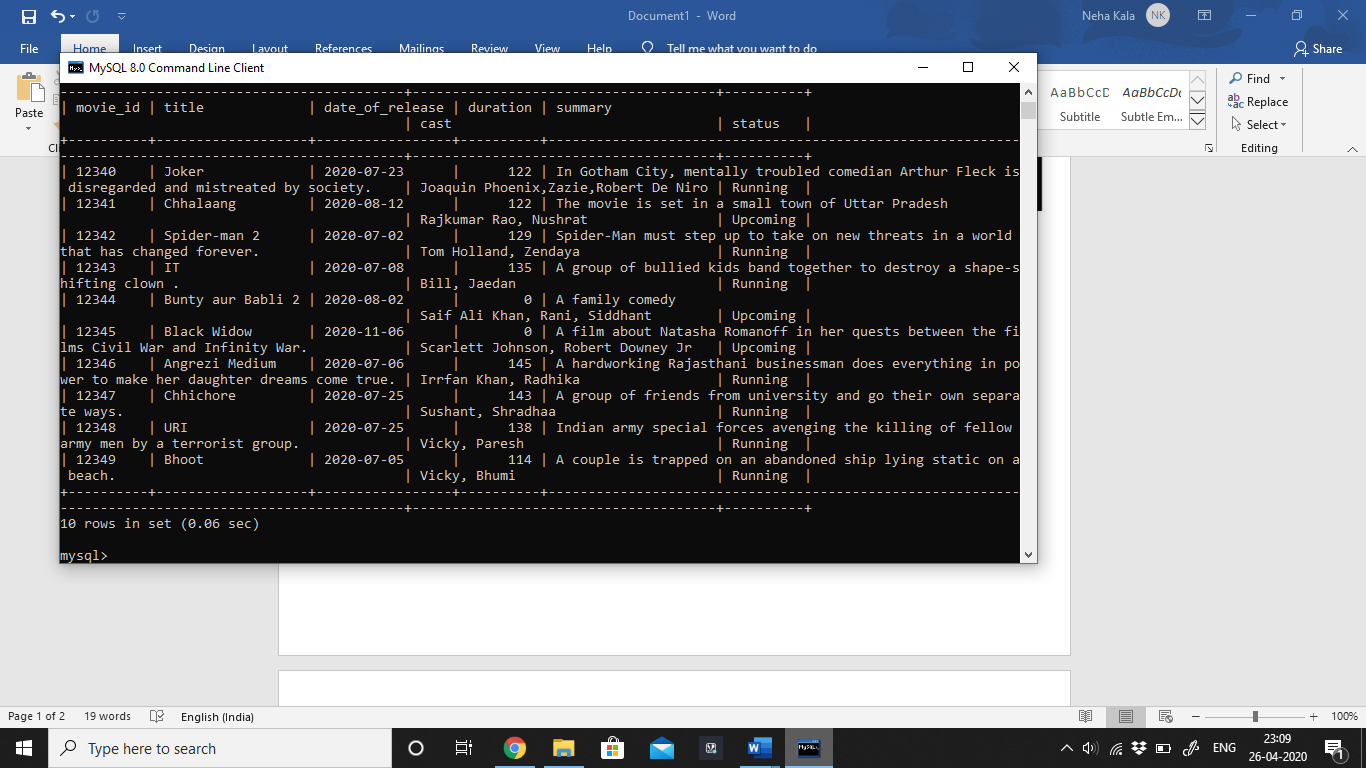
1. USER:

**E.g.: Insert into user values (‘12345’, ‘Surabhi’, ‘12345678’,’surabhi\_123@gmail.com’);**



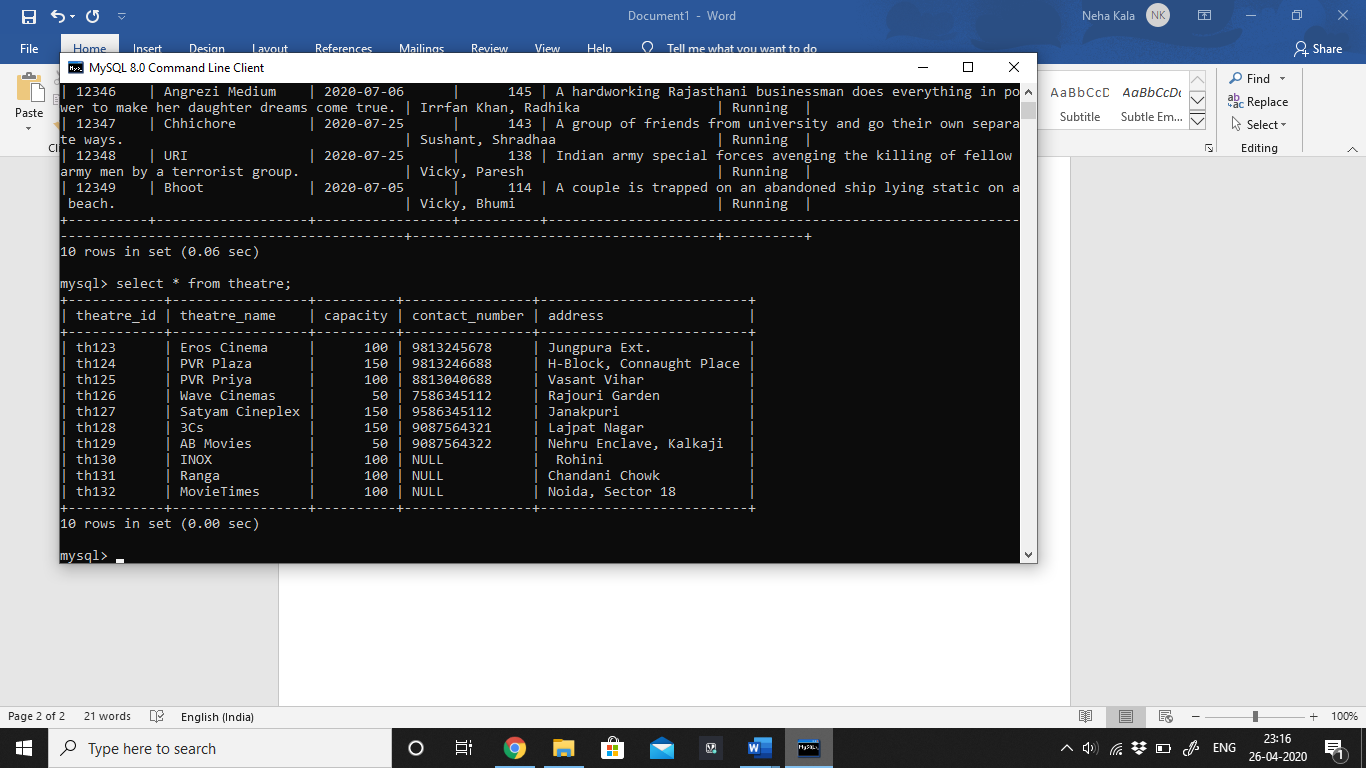
1. Movie

**E.G: insert into movie values (‘12340’,’Joker’, ‘2020-07-23’, 122,’summary’, ‘Running’);**



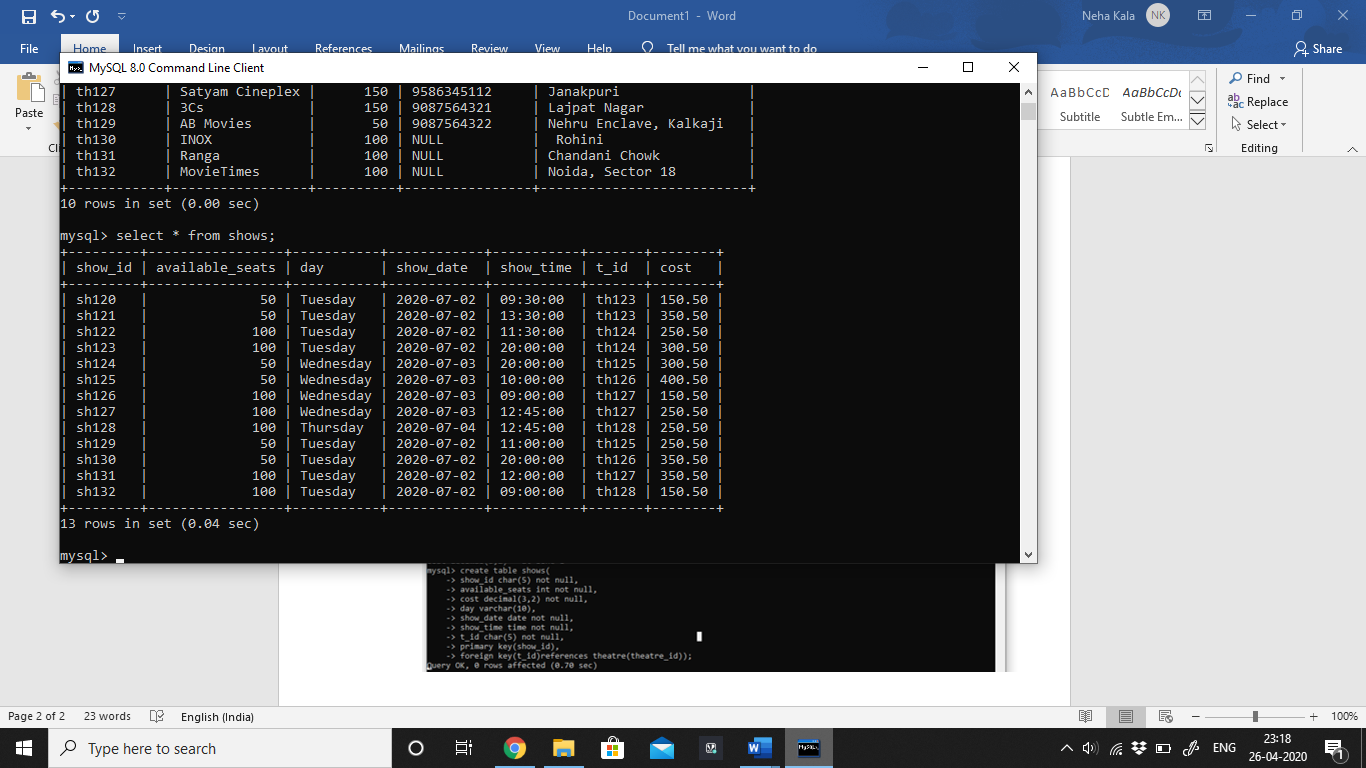
1. Theatre

**E.G: insert into movie values (‘th123’,’Eros Cinema’, 100, ‘9813245678’,’summary’, ‘Jung Pura Ext’);**



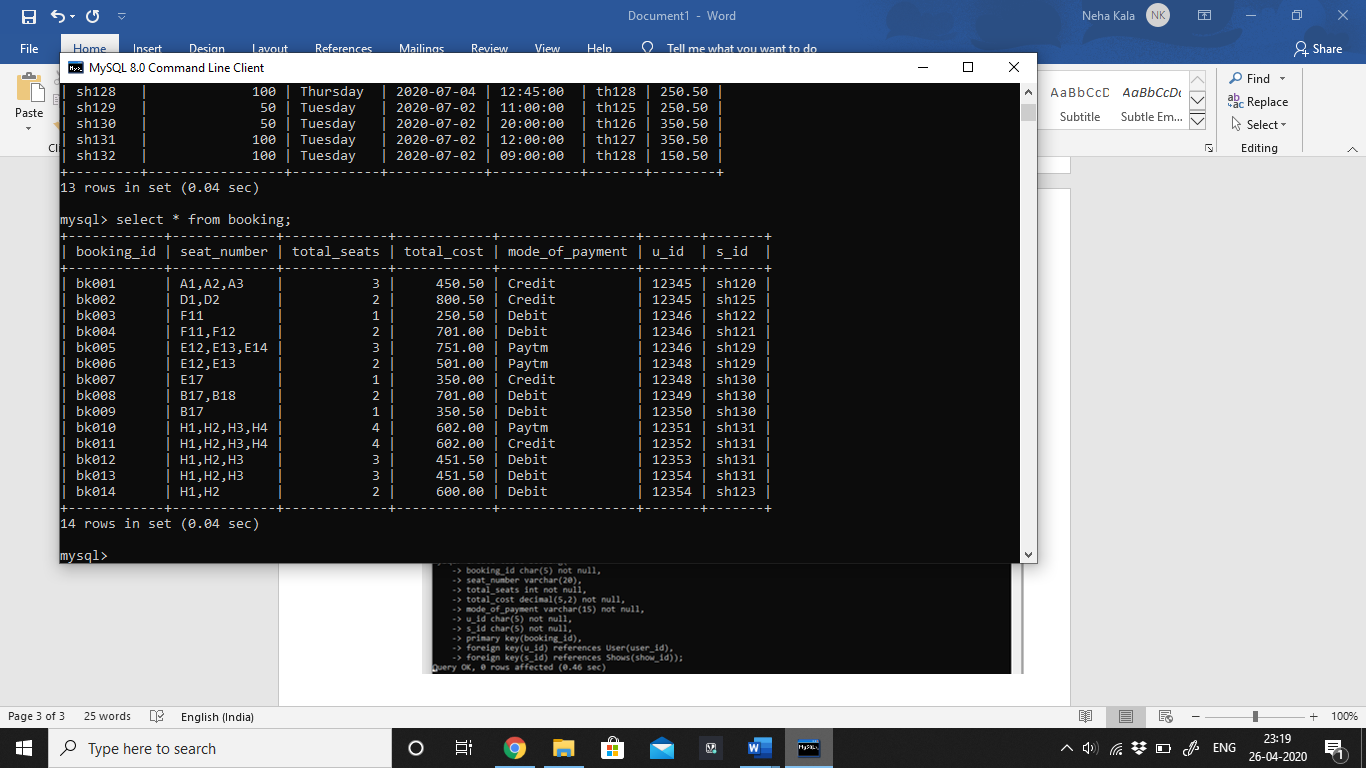
1. Shows

**E.G: insert into movie values (‘sh120’,50,’Tuesday’, ‘2020-07-02’, ’09:30:00’,’th123’,150.50);**



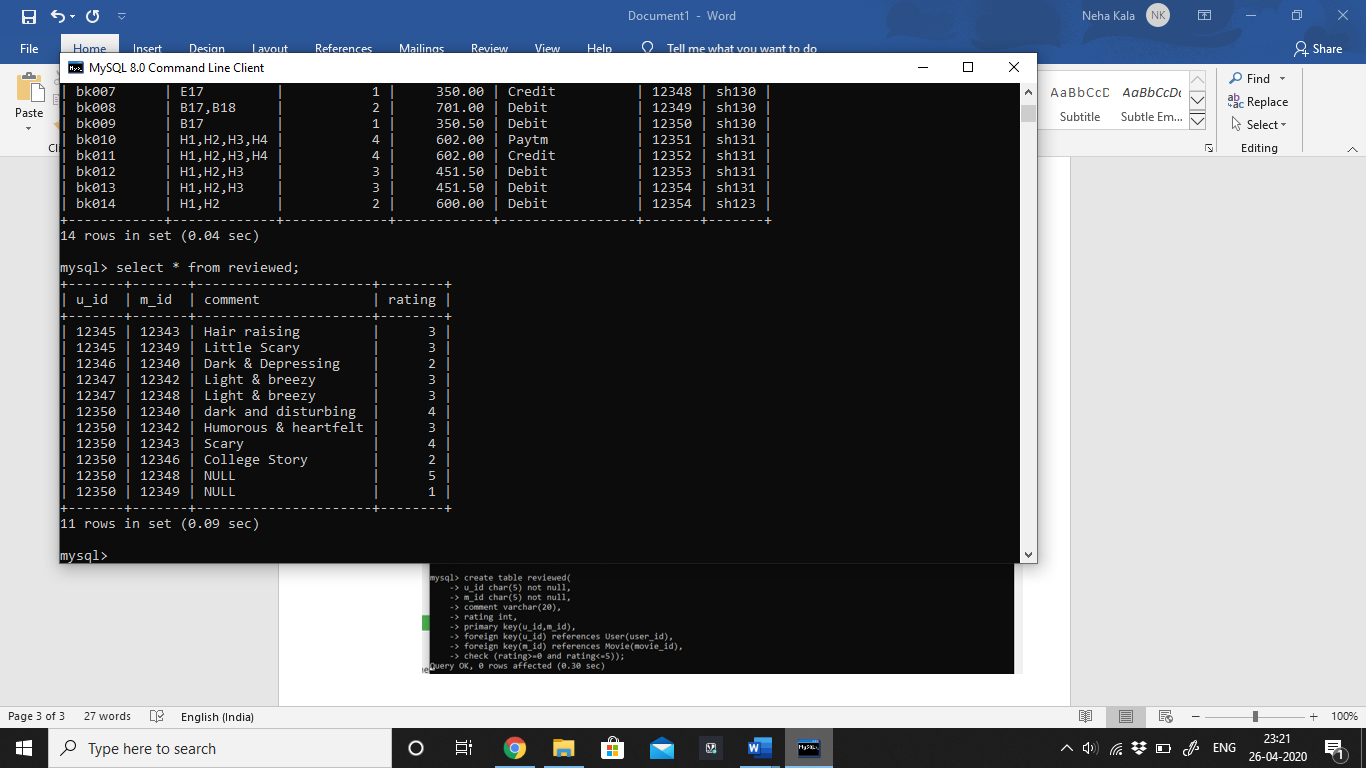
1. Booking

**E.G: insert into movie values (‘bk001’,’A1, A2,A3’, 3, 450.50,’credit’, ‘12345’,’sh120’);**



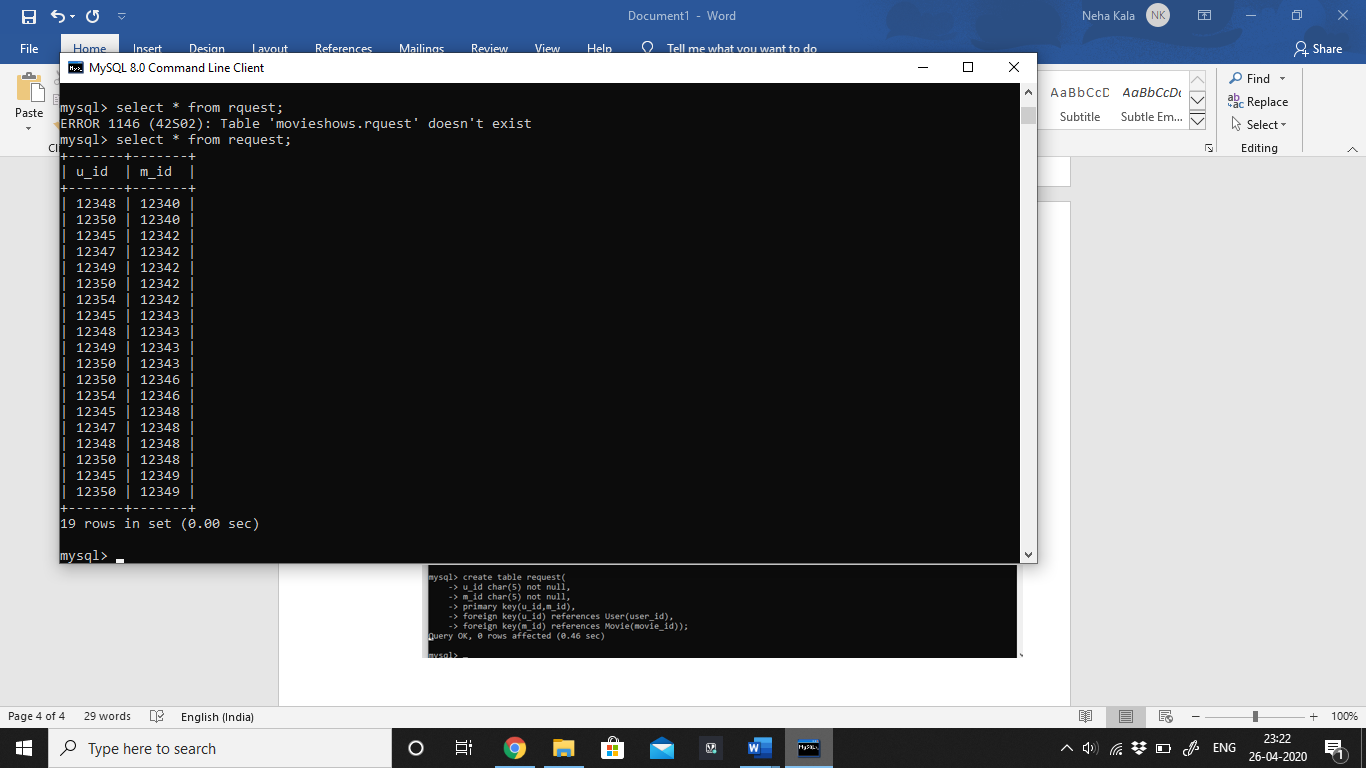
1. Reviewed

**E.G: insert into movie values (‘12345’,’12343’, ‘Hair raising’, 3);**



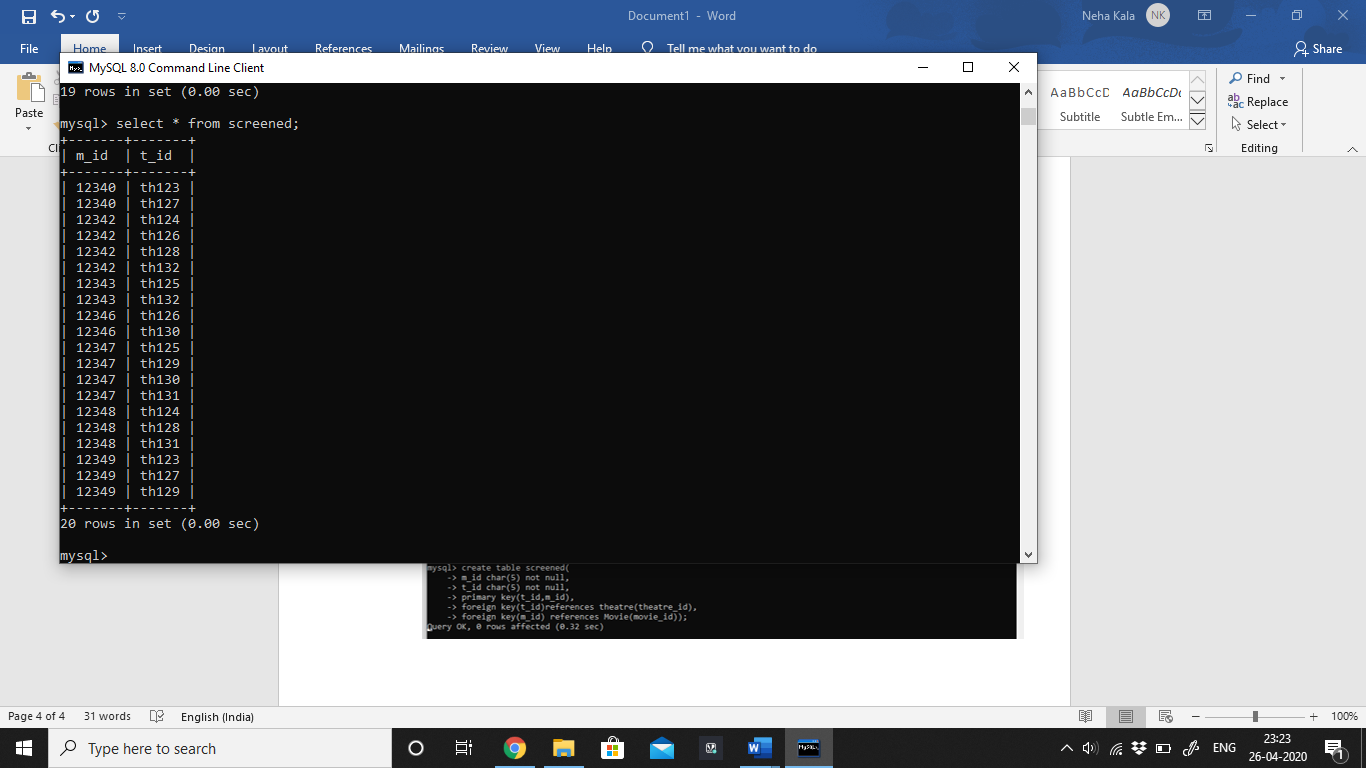
1. Request

**E.G: insert into movie values (‘12348’,’12340’);**



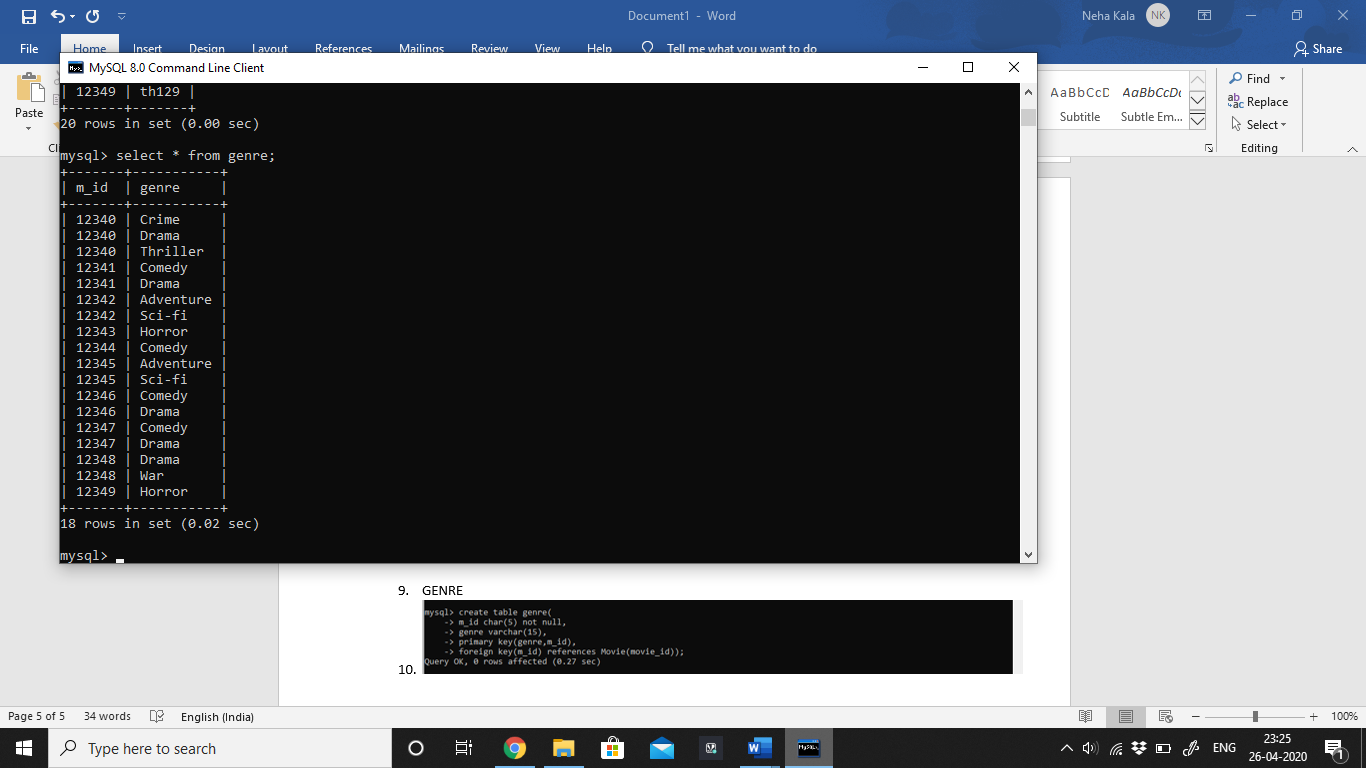
1. Screened

**E.G: insert into movie values (‘12340’,’th123’);**



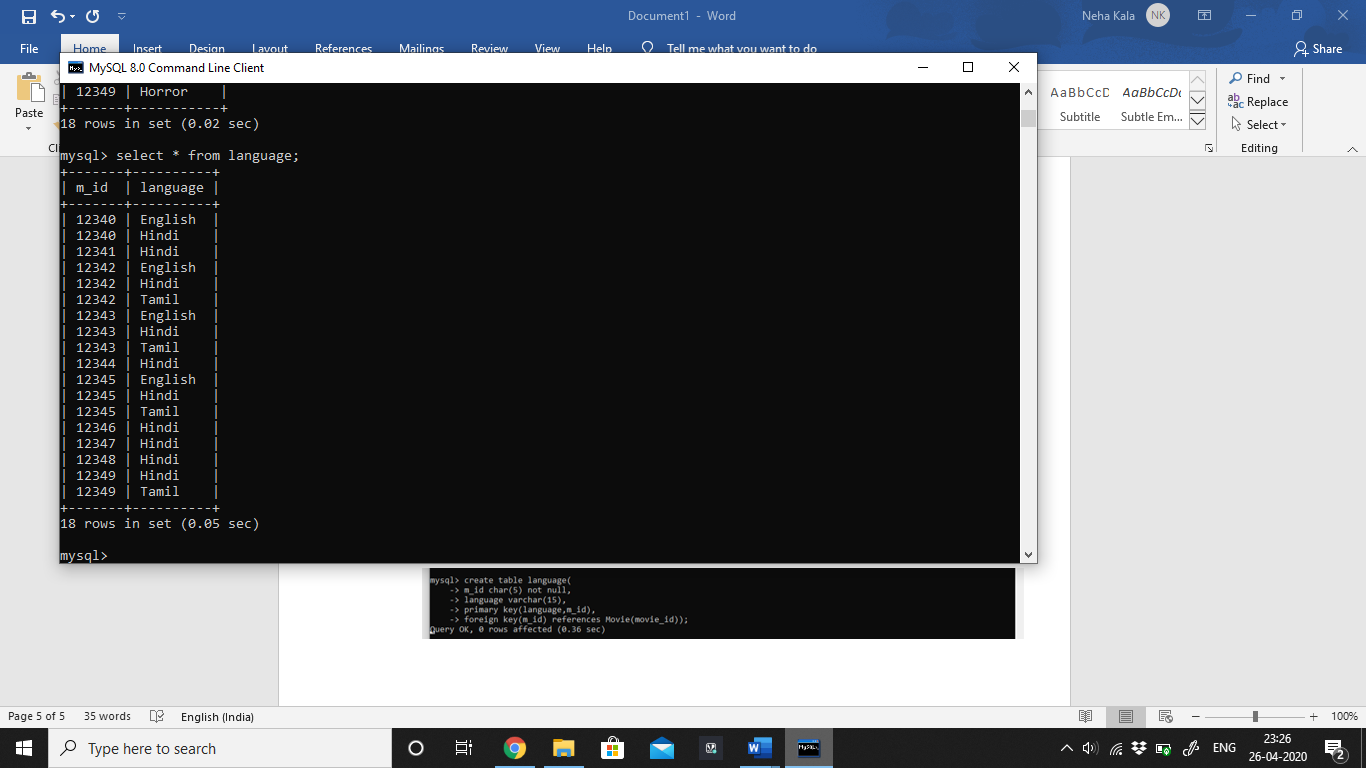
1. Genre

**E.G: insert into movie values (‘12340’,’Crime’);**



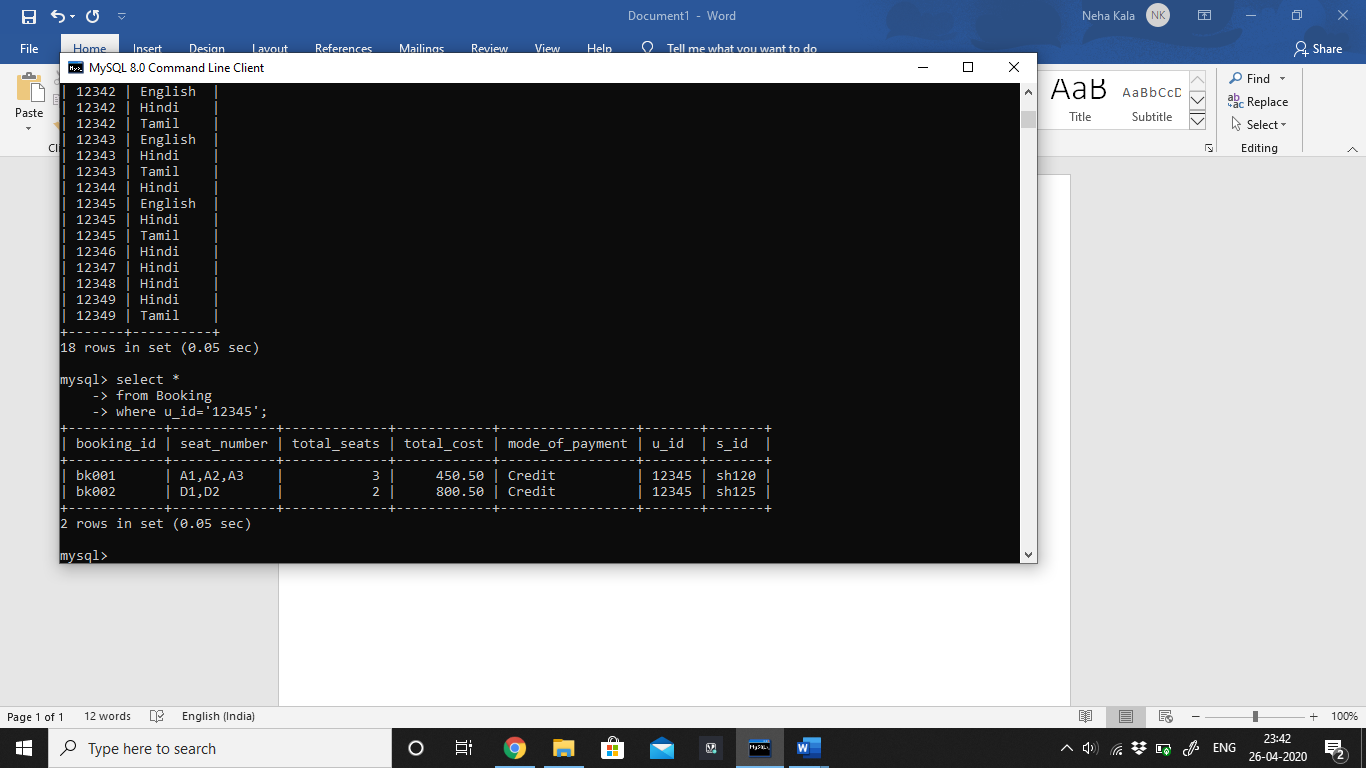
1. Language

**E.G: insert into movie values (‘12340’,’English’);**

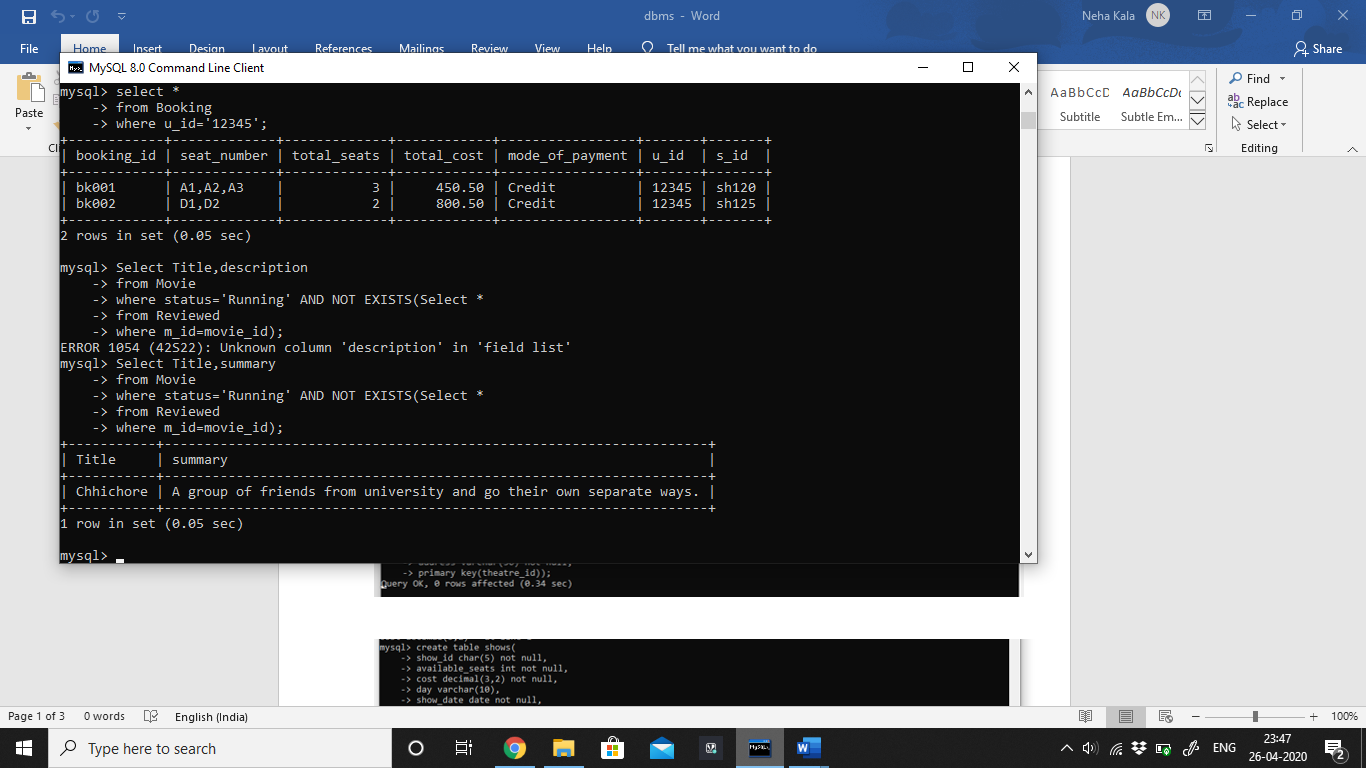


# Write 10 SQL queries.

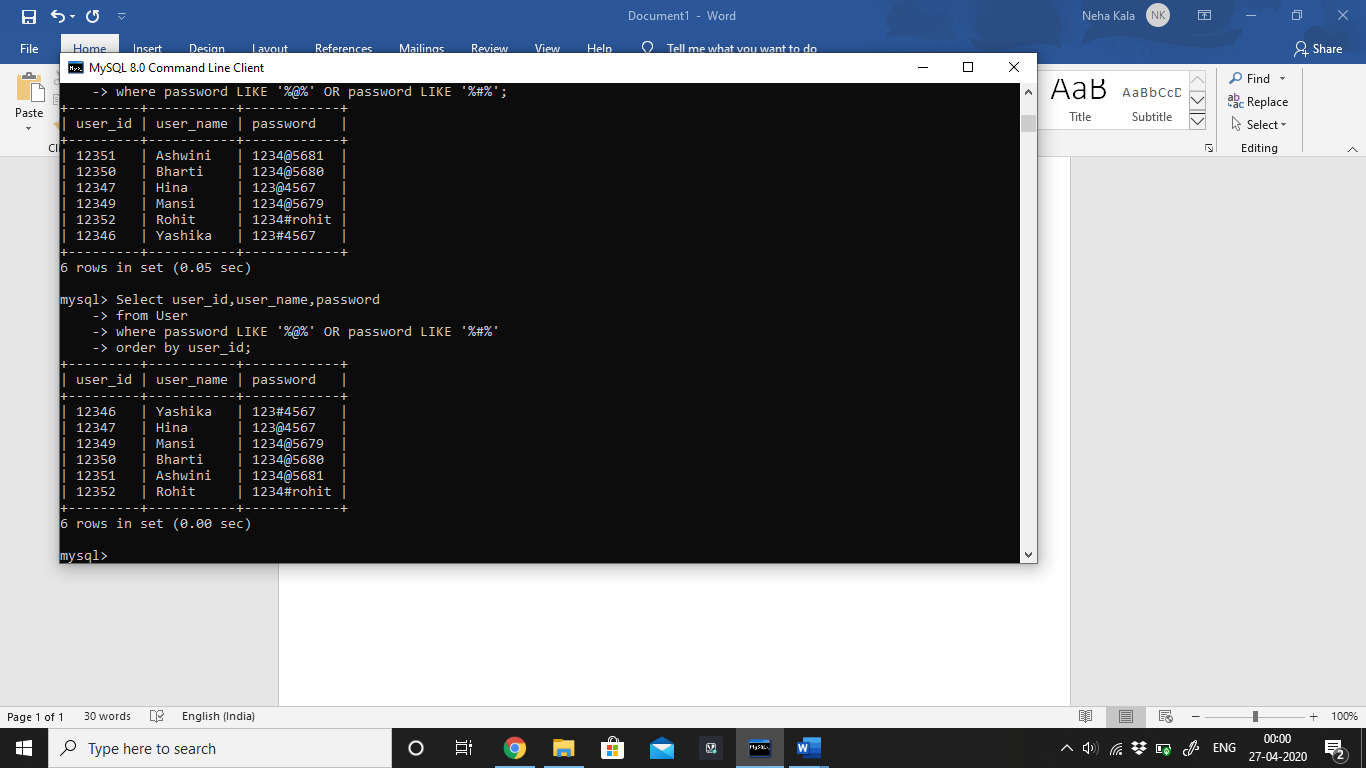
## Show booking details of User\_id=’12345’.



## List running movies have no reviews.

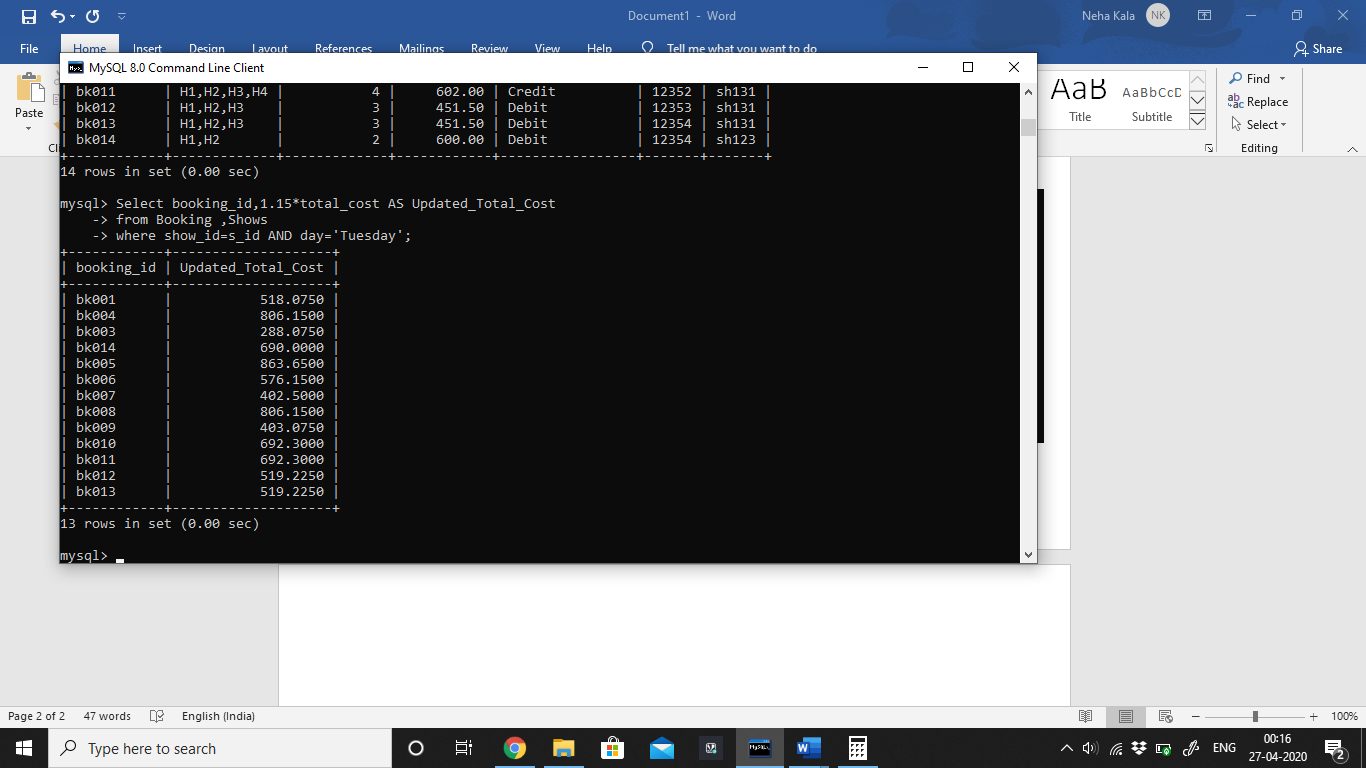


## Retrieve all users have @ or # in their password.

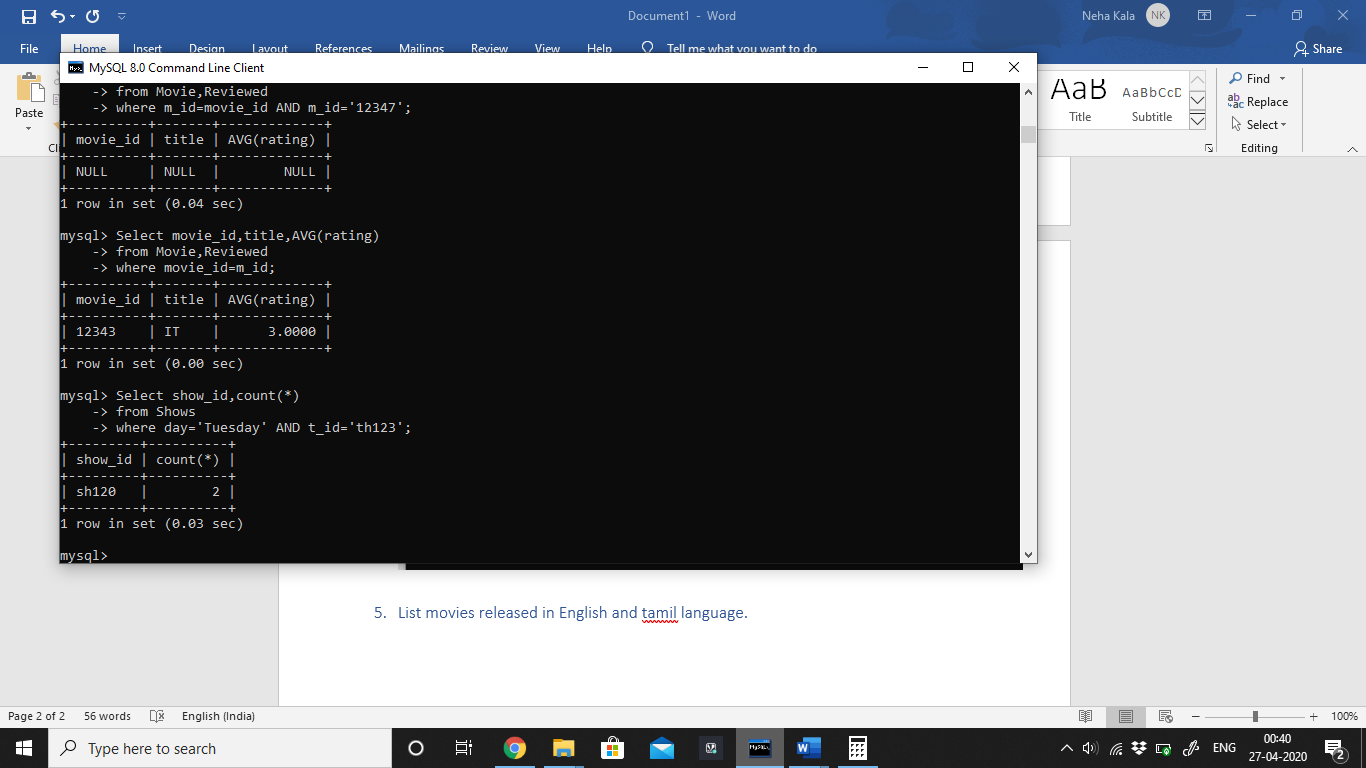


## Show updated total cost after applying 15% tax on the movie bookings of shows on Tuesday.

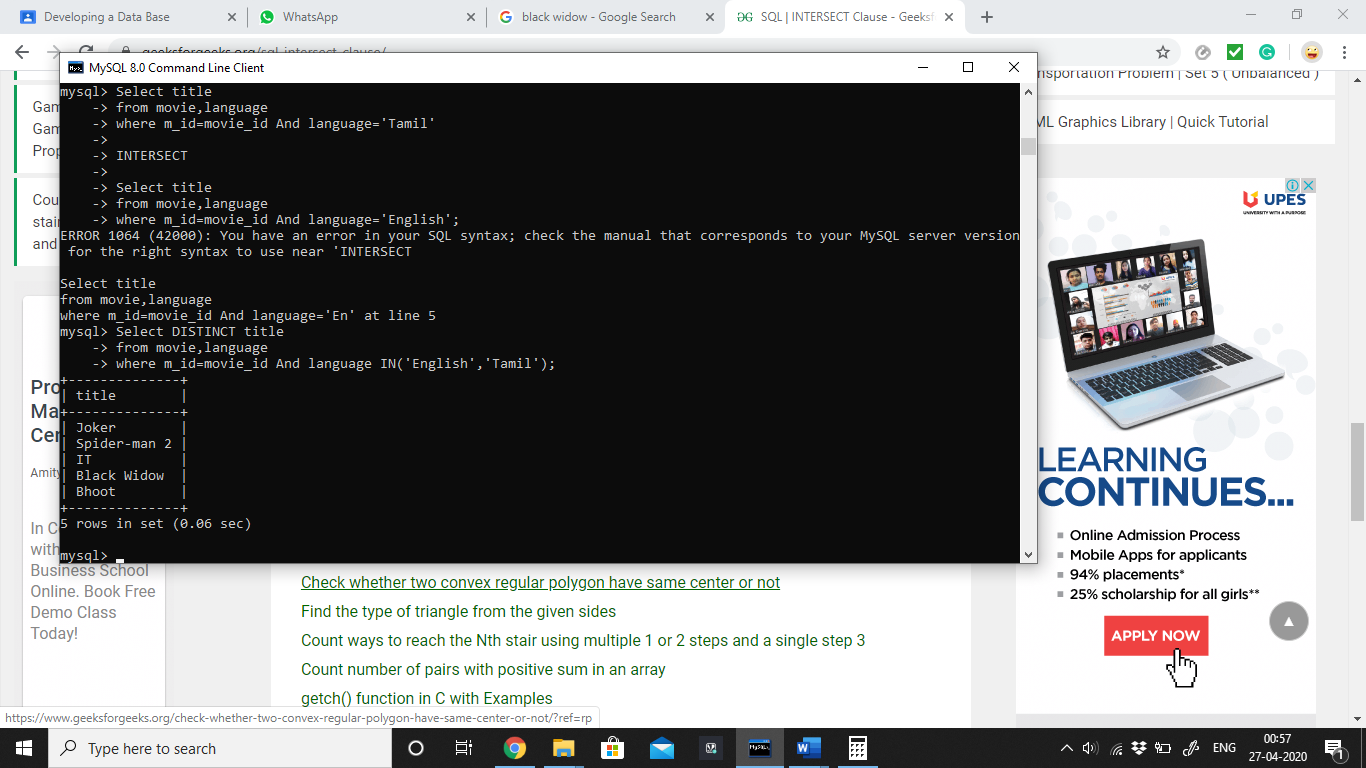




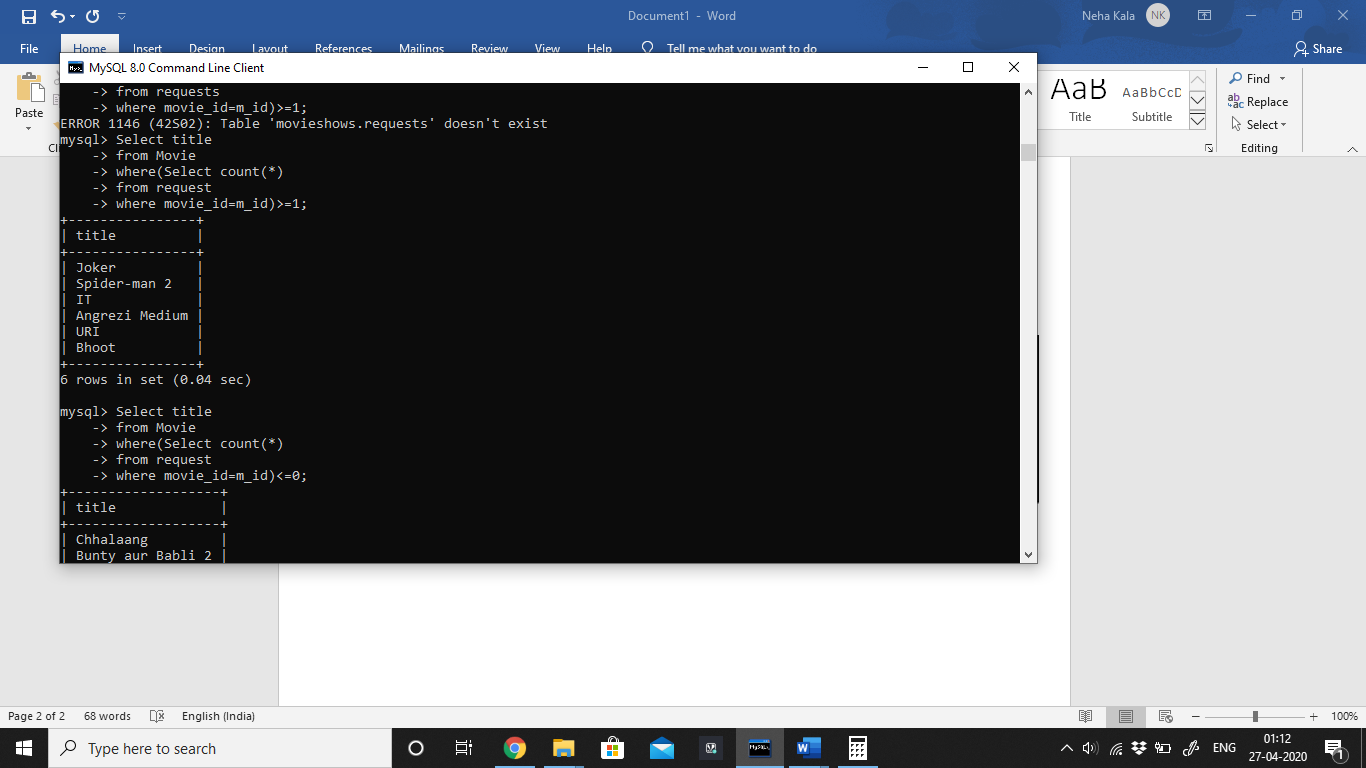
## Count the number of shows of theatre\_id=’th123’ on Tuesday.



## List movie titles released in English or Tamil.



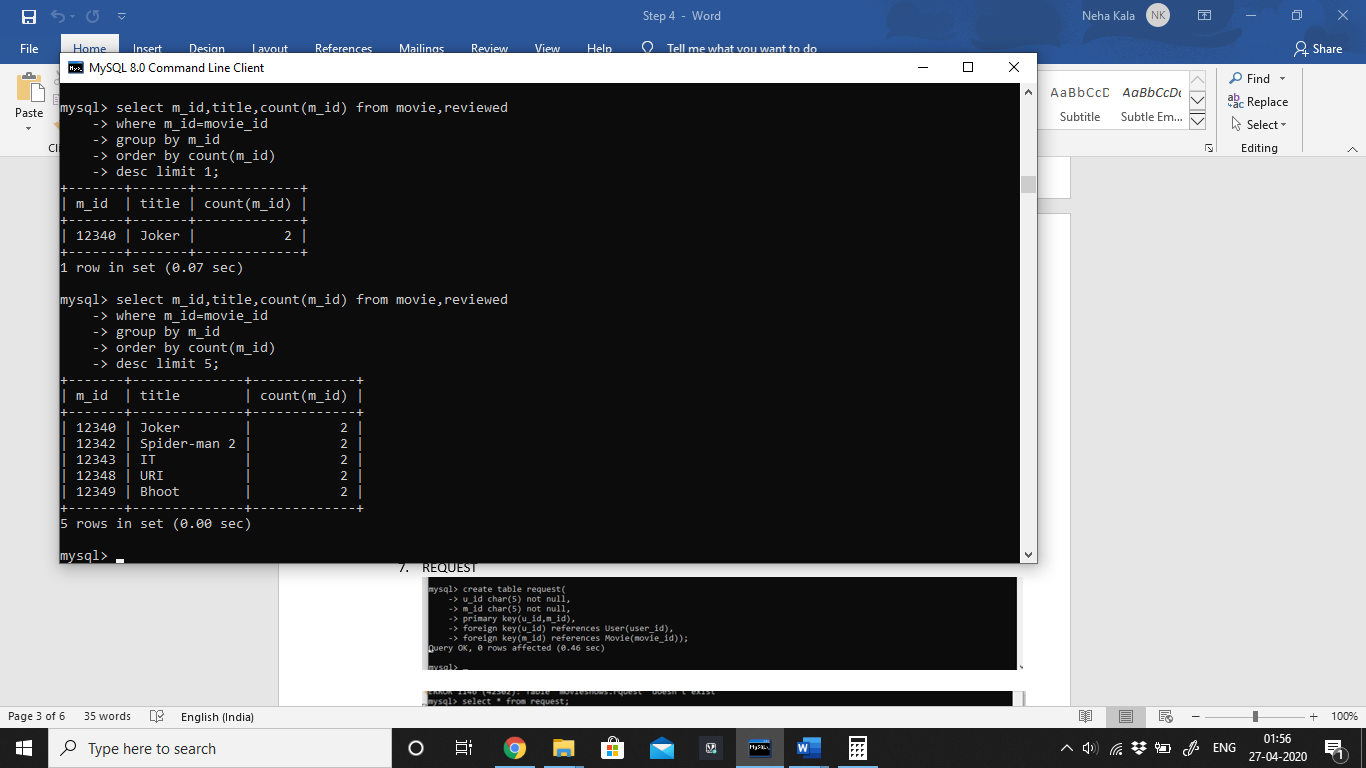
## List the movies that are requested at least once.



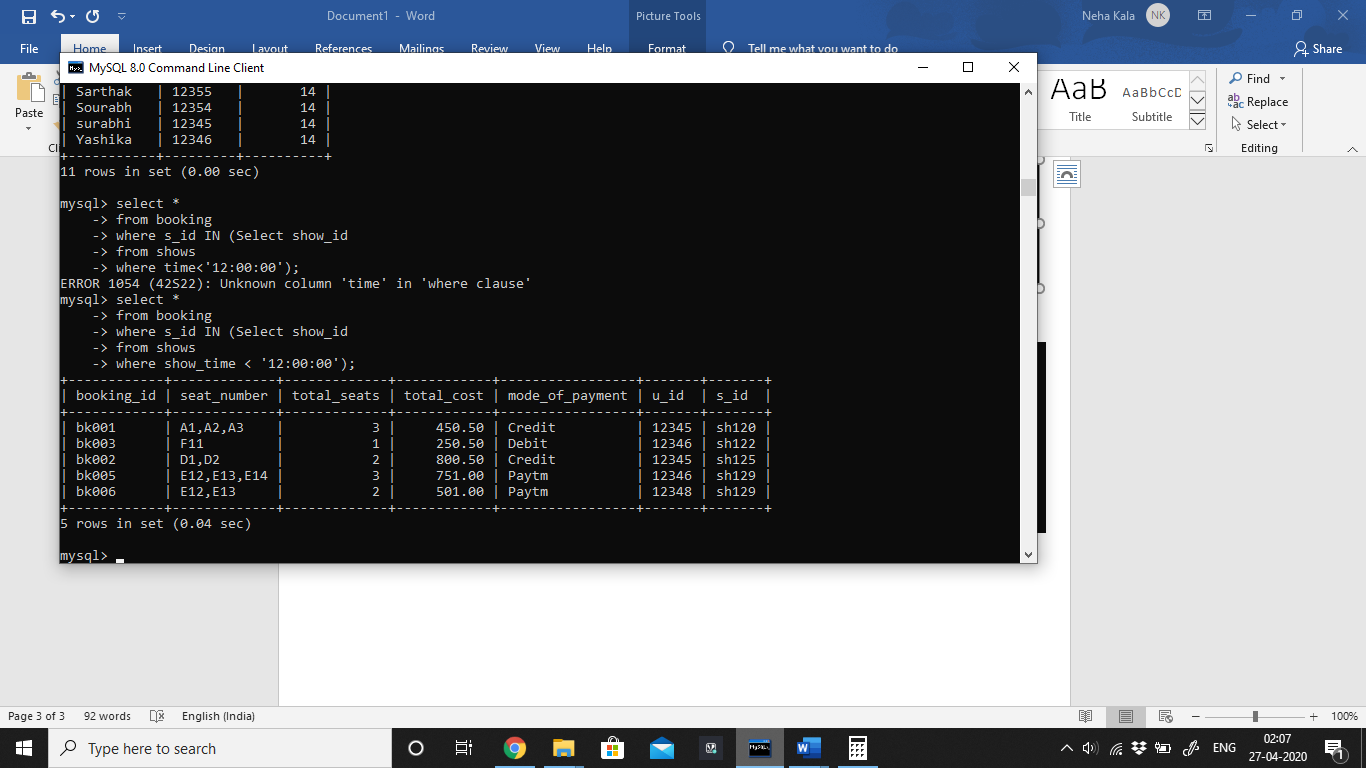
## Retrieve total number of users for each Mode\_of\_payment.

## 

## List first five most searched movies.



## Select the booking details of shows run in the morning.



# Write 10 relational algebra queries.

## Find title of all movies having rating more than 3.

R1 <-ρ(Movie\_id) (π m\_id (σ rating>3(Reviewed)))

R2 <- R1 \* Movie

RESULT <- πtitle(R2)

## Retrieve theatre details that screen movies released in English language.

R1 <-( σ language=’English’(Language))

R2 <- ρ(Theatre\_id) (π t\_id ( R1 \* Screened))

R3 <- R2\*Theatre

RESULT <- π theatre\_id,theatre\_name,capacity,address,contact\_number(R3)

## Show number of shows in each theatre.

t\_idᴲcount (show\_id) (Shows)

## List all movies of horror genre.

M\_genre <-ρ(Movie\_id) (σ genre=’Horror’(Genre))

RESULT<- πtitle (M\_genre \* Movie)

## List Theatre\_name with capacity more than 100.

R1 <-(σ capacity>100(Theatre))

RESULT <- πTheatre\_name, capacity(R1)

## Retrieve details of movies that are released in language hindi and have genre drama.

R1 <- π m\_id( σ language=’Hindi’(Language))

R2 <- (π m\_id ( σ genre=’Drama’(Genre))

R3 <- ρ(Movie\_id) ( R1 **∩ R2)**

RESULT <- π movie\_id,title(R3\*Movie)

## Retrieve details of users done no booking.

R1 <- π user\_id( (User))

R2 <- ρ(user\_id) ( π u\_id( (Booking)))

R3 <- ( R1 **- R2)**

RESULT <- π user\_id,user\_name(R3\* User)

## Retrieve booking\_id of all bookings where payment mode is credit.

R1 <-ρ(User\_id) (σ mode\_of\_payment=’Credit’(Booking))

RESULT <- πbooking\_id(R1)

## Retrieve title of movies released on 17th of June 2020.

RESULT <- πtitle (σ date\_of\_release=’2020-06-17’(Movie))

## List users who request and booked the movie.

R1 <- π u\_id( (Booking))

R2 <- (π u\_id ( (Request))

R3 <- ρ(user\_id) ( R1 **∩ R2)**

RESULT <- π user\_id,user\_name(R3\*User)