LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CC5051NI Database System

Assessment Weightage & Type

50% Individual Coursework

Year and Semester

2018-19 Autumn / 2018-19 Spring

Student Name: Rajat Shrestha

London Met ID: 17030954

College ID: np01cp4a170021

Assignment Due Date: 28 Jan 2019

Assignment Submission Date: 28 Jan 2019

Submitted to

Mr. Rohit Pandey (Module Leader)

Mr. Prashant Lal Shrestha (Lecturer)

Mr. Sagun Man Tamrakar (Tutor)

Ms. Yunisha Bajracharya (Tutor)

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Contents

Table of Figures:

Table of tables:

Company Information

1. Introduction

BSR Movies a relatively new cinema operator located at BG Mall in Gongabu, Kathmandu which provides cinema and entertainment related services such as bookings, accommodations, and meals with cinema to the general consumers. BSR movie is the first installment from Sarthak Entertainment Pvt. Ltd. It has two halls which preview latest and popular movies from all over the world to satisfy its audience who are willing to pay to watch their desired show. Both the halls are fully equipped with high-end projectors and sound systems which previews various movies in 2K resolution and with Dolby Atmos audio systems and can contain 160 spectators. As it has become more and more popular and gaining new customers, the management team have decided to enlarge the business by adding online booking system. This will satisfy the needs of the customers while the business will also profit due to bigger sales. The enlargement of the business will increase the number of employees, customers, available meals, shows and reservations which will complicate storing and retrieving information tedious. The aim of this project is to manage a database which will host a platform for a good booking system so that the customer can easily book their preferred movie with desired features.

So, to handle this huge amount of information a good database model is required to be implemented so, various data can be stored efficiently with minimal data redundancy within its various tables. The booking system is supposed to handle multiple bookings entered by the employee from online bookings forms and request from the customer. So this project is based on developing a database for a theatre operating system to manage information regarding its services, employees, customers, bookings, halls, shows, movies and mainly involves the tickets bought by the customers directly or booked. The database will include various entities with numerous attributes which were created so that the database is normalized till the 3rd normalization form, to reduce anomalies while maintaining and updating the database with new information. This all process is to be done and documented properly so that one can easily understand the design and flow of the database if it needs to be redesigned or viewed for inspection.

2. Business activities & Operations

BSR Movies provides various entertainment related services such as previewing of new/popular movies of mainly three different languages: Hindi, English, and Nepali. The company targets a wide range of customers, from teenagers to senior citizens, but the customer base was observed to be dominated by youths who have more exposure to smartphones and the internet, an online system for booking and buying ticket seems mandatory. The entertainment company provides accommodation services at various rates with different features. The company will also implement an online booking system which will require an employee handling the database who will take requests from the company's website where users can log in/register to get access to the related services. The show and accommodation type can be booked so that customers can enjoy their movie with desired features. The company is also proposing to hold a discount event for a week after announcing the online booking system to book a ticket. This event will advertise the company while also spreading the news of a new online booking system. The event offers 25% off on any tickets booked via the online procedure.

The customers should register to book the tickets. Then the customer can book available seats for a show allowing them to pick the desired accommodation and show. Then the booking must be confirmed by the customer within an hour by submitting the total amount for the ticket. Customers can also buy multiple tickets at the counter which will also be stored as a booking event as it is done before the actual show. The theatre also manages small vendors for selling food items such as candy, popcorn, and drinks. These vendors also provide free meals as specified by the show.

The accommodations consist of Platinum, Gold, and Silver categories allocated in each hall. Platinum seats contain luxurious seats with many features is priced at Rs 500. Gold Seats are normal seating without additional features which costs Rs 400. Silver seats are regular seating which costs Rs 300. Any employee can get a single ticket for themselves for free by submitting their Id at the counter which will also be recorded which will be treated as Exclusive Accommodation.

3. Rules to be followed

The company has already defined a set of rules to dictate the process of buying the tickets and various activities that can be held in its jurisdiction. The terms and conditions are presented to the customers in both the websites and at the hardcopy of the registration form. For the company to operate the new online system new rules must be followed which are:

3.1. Registration Rules:

- The customer should fill all the required details in the registration process,
- To register the customer should be above 15 years,
- For customers below the age of 15 years, parental guidance is required,
- Phone number, address, name, and sex must be specified while registering.

3.2. Ticket & Booking Policy:

- The customer should be registered book tickets online,
- Tickets temporary booked must be cleared before an hour of previewing the show,
- The temporary bookings will be dismissed if not cleared,
- The temporary bookings can be canceled but tickets cannot be refunded,
- If required to be refunded the customer must have a valid reason,
- The ticket for a particular show is only eligible for that show,
- The exclusive seat also require a ticket but is free,
- One ticket for Exclusive seat can only be redeemed by the employee for One show,
- Employees have to buy tickets if no Exclusive seats are available,

4. Database Design

4.1. Creation of Objects – Entities and Attributes:

Entities are any object or event that contains various information about them. Entities in the Database can be both physical (people, ticket, accommodation etc.) or abstract (booking, show, shifts etc.) (Scottish Qualifications Authority, 2010). Attributes are simply characteristics of the Entity, it stores the information about different aspects of an entity (IBM Corporation, 2018). For example, taking a person (Entity), a person will have a name, age, sex, and address (attributes).

Scenario analysis.

The scenario states that the database will have to record a person's addresses as well as mailing addresses. In which each address consists of a country, zone, city, street and street number. the person's contact is also stored which will have phone-no, cell-no, e-mail, and fax-no (fax-no can be null). The type of person, sex, age and DOB (optional) is also stored in a person entity. The type will determine if the person is an employee or a customer which will further categorize the person into new-customer, old-customer or an employee. The employee of any type can have benefits such as free tickets and meals.

As the scenario only emphasizes the detail of an individual customer or an employee, the attributes for the ticket and booking is also important as this database will deal with cinemas. So, practically it has a previewing movie, preview date/time, usher, hall, meals, and accommodation-information. Assuming these as the required information, the initial ER-Diagram becomes like the following:

4.2. Identifying relations and Defining Keys using Data Dictionary:

Assuming the required attributes of all the entities, a database can be formed, which will be functional but not quite practical. To design a better database, this base database model must be observed, so several data dictionaries are created to define and relate these entities and attributes. Data dictionary describes the structure of the whole database which is represented in a table by including detailed information about an entity in a database and defines the nature of attributes which provides all the required information about a database. Data dictionary simply put is a collection of tables which gives us detailed information on the inner mechanics of a database (Kreines, 2009). The three separate entities data dictionaries for this table are as follows:

Table 1: Data Dictionary for Person table

Entity	Attributes	Datatype	Length	Description	Constraints
Person	Person_ID	NUMBER	10	Unique Identifier of the person	PRIMARY_KEY
	Name	VARCHAR2	20	Name of person	NOT_NULL
	Age	NUMBER	5	Age of person	NOT_NULL
	Sex	VARCHAR2	20	Sex of person	NOT_NULL
	DOB	DATE		DOB of person	
	Type	VARCHAR2	20	Type of person	NOT_NULL
	Job	VARCHAR2	20	Job of person	
	Salary	NUMBER	10	Salary of person	
	E_mail	VARCHAR2	50	e-mail of person	UNIQUE
	Phone_No	NUMBER	10	phone number of the person	
	Cell_No	NUMBER	10	the cell number of the person	UNIQUE
	Fax_No	NUMBER	10	fax number of the person	
	Location_ID	NUMBER	10	Unique Identifier of the location	NOT_NULL
	Country	VARCHAR2	20	Country of person	NOT_NULL
	Zone	VARCHAR2	20	Zone of person	NOT_NULL
	City	VARCHAR2	20	City of person	NOT_NULL
	Street	VARCHAR2	20	Street of person	NOT_NULL
	Street_Name	VARCHAR2	20	street number of the person	NOT_NULL
	POB_Address	VARCHAR2	1	The mailing address of the person	DEFAULT = Y

Table 2: Data Dictionary for Booking table

Entity	Attributes	Datatype	Length	Description	Constraints
Booking	*Person_ID**	NUMBER	10	Unique Identifier of the person	PRIMARY_KEY, FOREIGN_KEY
	*Show_ID**	NUMBER	10	Unique Identifier for the show	FOREIGN_KEY
	No_Tickets	NUMBER	10	Number of tickets booked	DEFAULT=1
	Booking_Date	DATE		Date of booking	NOT_NULL
	Accommodation	VARCHAR2	20	Accommodation type	NOT_NULL
	Price	NUMBER	10	Price of each ticket	DEFAULT=0

Table 3: Data Dictionary for Show table

Entity	Attributes	Datatype	Length	Description	Constraints
Show	Show_ID	NUMBER	10	Unique Identifier for the show	NOT_NULL
	Meal	VARCHAR2	20	Meal served in the show	
	*Usher**	NUMBER	10	An employee who ushers the show	FOREIGN_KEY
	Date	DATE		Date of show	NOT_NULL
	Time	NUMBER	10	Time of show	NOT_NULL
	Hall	VARCHAR2	20	Hall of show	NOT_NULL
	Movie_Name	VARCHAR2	20	Name of the movie	NOT_NULL
	Duration	NUMBER	10	Duration of the movie	
	Rating	NUMBER	10	Rating of the movie	

The purposed data dictionaries will help understand the type of data and its relation to various entities. The final data dictionaries will also have the same data types but the constraints might differ due to the creation and assigning of new prime attributes or minor changes.

5. Entity Relationship Diagram

The main purpose of ER-Diagrams is to clarify the connection between the entities and their properties (Song, et al., 1994). After a lot of assuming and correcting the entities and their attributes, the basic structure of the unnormalized database is created below. The notation used in this model to describe the relation between the entities is Crows-foot notation which utilizes look across cardinality and participation. So for a better understanding of the base model of the database, an ER-Diagram is created to state the relationship between various entities.

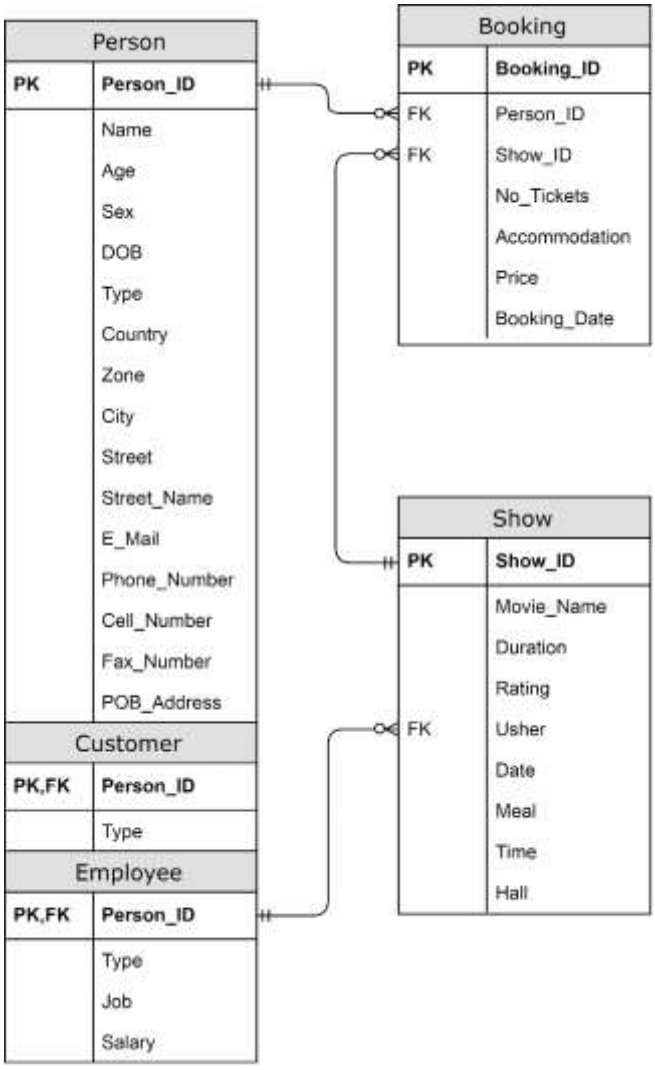


Figure 1: Initial ER-Diagram

Assumptions

- A booking can have multiple tickets but needs to be of a certain accommodation.
- A booking cannot specify two separate accommodations for a single show.
- The employee gets a free ticket for Exclusive seat.
- Tickets shouldn't be issued more than 180 for each show due to hall capacity.
- The booking must first be confirmed before being entering into the database.
- Any booking without payment is treated as a temporary booking.
- Temporary bookings will be dismissed before an hour of the show.
- the POB_Address is a Boolean value denoted by a string 'y' or 'n'.
- When the address is being specified, one address is required to be POB_Address.
- The database is not accessible by customers.
- All of the information stored in the database is handled by an employee.
- Any modification on the records is required to be handled by the supervision of the manager.
- The job of an employee isn't related to the salary, i.e. the same job might have different salaries for different individuals.

Normalization

1. Un-Normalized form (UNF)

In the un-normalized form, all the attributes are listed. This un-normalized form the person is taken a reference entity. When Person is taken as a reference, according to the scenario a person can have many addresses and can also watch many shows. If the data within each repeating group is not related to any other data within any other repeating group, it is considered as multiple repeating group. So according to that logic, the person will have multiple repeating groups which are addresses and show detail as shown in a simple ER Diagram below:

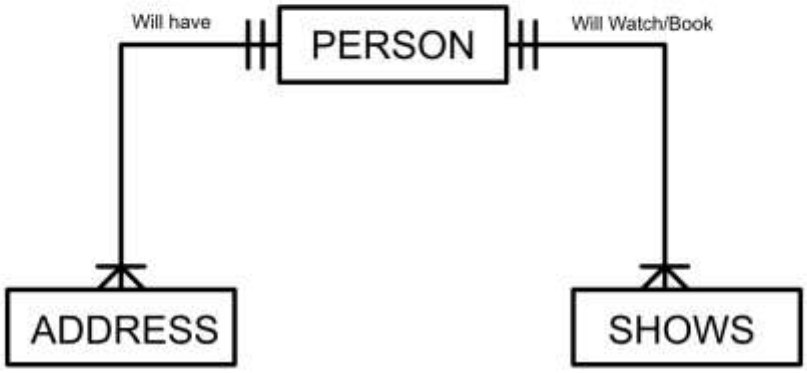


Figure 2: Possible repeating group for a person

UNF:

Person (**Person_ID**, Type, Name, Age, Sex, DOB, Job, Salary, E_mail, Phone_No, Cell_No, Fax_No,{Address_ID, Country, Zone, City, Street, Street_Name, POB_Address} , {, No_Tickets, Booking_Date, Show_ID, Meal, Usher, Show_Date, Show_Time, Hall, Movie_Name, Duration, Rating, Accommodation, price})

2. First-Normal Form (1NF)

To normalize the entities in the first normal form we must remove all the repeating group from the person table, which is: multiple addresses and multiple bookings. However the person can also be an employee or a customer, so these subtypes of person will also be separated in the First Normal form. The primary Key of the person will be also be invoked by the Customer and Employee subtypes as their own primary key while the Address and booking will have composite primary keys to define the addresses and the bookings of a single person.

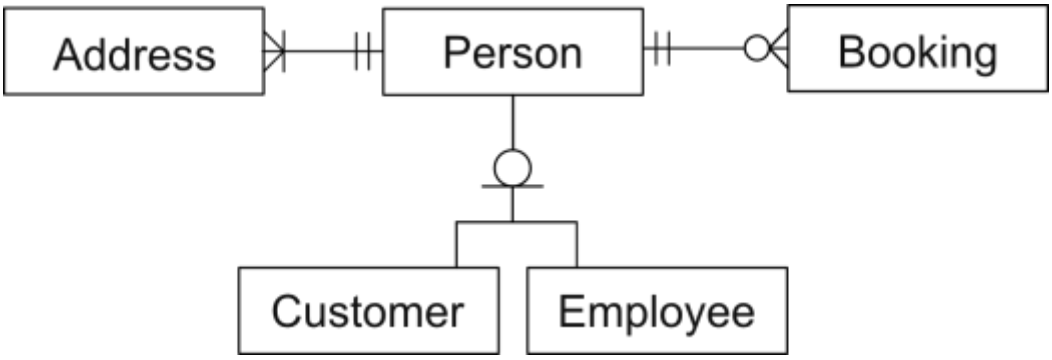


Figure 3: Every relation with person entity

1NF:

Person (**Person_ID**, Name, Age, Sex, DOB, E_mail, Phone_No, Cell_No, Fax_No)

Employee (***Person_ID****, Job, Salary)

Customer (***Person_ID****, Type)

Address (***Person_ID****, **Location_ID**, Country, Zone, City, Street, Street_Name, POB_Address)

Booking (***Person_ID****, **Show_ID**, No_Tickets, Booking_Date, Accommodation, price, Meal, Usher, Show_Date, Show_Time, Hall, Movie_Name, Duration, Rating)

3. Second-Normal Form (2NF)

To normalize the database model into 2nf all of the tables must be in 1NF and then all the partial dependencies must be analyzed and removed. Partial dependency is simply a functional dependency when a non-prime attribute is linked to fewer prime attributes than being fully linked to all of the prime attributes. For example, IF A & B is a prime attribute in a table with C which is fully dependent on A (i.e. A can give C) but B is not linked to C then it is a partial dependency. To remove partial dependencies all of the tables are analyzed.

1. **Person**: Every non-prime attributes is dependent on Person_ID

   Person_ID === Name, Age, Sex, DOB, E_mail, Phone_No, Cell_No, Fax_No

2. **Employee**: Every non-prime attributes is dependent on Person_ID

   Person_ID === Job, Salary

3. **Customer**: Every non-prime attributes is dependent on Person_ID

   Person_ID === Type

4. **Address**: The geographical location details are dependent on Location_ID and partially dependent on Person_ID, while the POB_Address is fully dependent on both Location and Person

   Person_ID*, Location_ID === POB_Address

   Location_ID === Country, Zone, City, Street, Street_Name

5. **Booking**: The Person_ID is not required to give the show and movie details.

   Person_ID*, Show_ID === No_Tickets, Booking_Date, Accommodation, price

   Person_ID* ===

   Show_ID === Meal, Usher, Show_Date, Show_Time, Hall, Movie_Name, Duration, Rating

Removing all the partial dependencies from the address and booking tables we get:

2NF:

Location (**Location_ID**, Country, Zone, City, Street, Street_Name)

Address (***Person_ID****, ***Location_ID****, POB_No)

Person (**Person_ID**, Type, Name, Age, Sex, DOB, Job, Salary, E_mail, Phone_No, Cell_No, Fax_No)

Employee (***Person_ID****, Job, Salary)

Customer (***Person_ID****, Type)

Booking (***Person_ID****, ***Show_ID****, No_Tickets, Booking_Date, price, Accommodation)

Show (**Show_ID**, Meal, Usher, Show_Date, Show_Time, Hall, Movie_Name, Duration, Rating)

4. Third-Normal Form (3NF)

To normalize the database model into 3NF all of the tables must be in 2NF and then all the transitive dependencies must be analyzed and removed. A transitive dependency is simply a functional dependency when a non-prime attribute is functionally dependent on another non-prime attribute. For example If A is a prime attribute in a table with B which is fully dependent on A and there also exists an attribute C which could be given by B then the partial dependency occurs between A and C as A gives B and B gives C. To remove all of the transitive dependencies form the tables above analyzing all the transitive dependencies in each table.

1. Location: Every non-prime attribute is fully functionally dependent on location ID.

2. Address: POB_Address is functionally dependent on Location and person.

3. Person: Every non-prime attribute is fully functionally dependent on location ID.

4. Employee: Every non-prime attribute is fully functionally dependent on location ID.

5. Customer: Every non-prime attribute is fully functionally dependent on location ID.

6. Booking: Person_ID and Show_ID will define Accommodation which can also give a price.

    Person_ID* , Show_ID === Accommodation === Price

    Accommodation === Price

7. Show: the only non-key relation is how Movie details are given by Movie_Name

    Show_ID == Movie_Name == Duration, Rating

    Movie_Name == Duration, Rating

By removing all of these transitive dependencies we get:

3NF:

Location (**Location_ID**, Country, Zone, City, Street, Street_Name)

Address (***Person_ID****, ***Location_ID****, POB_Address)

Person (**Person_ID**, Name, Age, Sex, DOB, E_mail, Phone_No, Cell_No, Fax_No)

Employee (***Person_ID****, Job, Salary)

(note: the Different Employee can have different salaries for the same job)

Customer (***Person_ID****, Type)

Booking (***Person_ID****, ***Show_ID****, No_Tickets, Booking_Date, *Accommodation**)

Accommodation (**Accommodation**, Price)

Show (**Show_ID**, Meal, *Usher**, Show_Date, Show_Time, Hall, *Movie_Name**)

Movie (**Movie_Name**, Duration, Rating)

Since all of the attributes are fully dependent on the prime attributes and no repeating groups occur so the following database model is in 3NF.

5. Entity Relationship Diagram after Normalization

After Normalization, there will be nine entities and their relations are clarified below:

1. Location can be related to none to many addresses.
2. The address must have a location and person.
3. The person will have at least one address.
4. A person can be, employee, customer or both.
5. A person can have none to multiple bookings
6. A booking must have Person, Show, and Accommodation specified.
7. Accommodation can be booked none to many in one show.
8. The show must have an usher and a movie.
9. The show could be booked none to multiple times.
10. A movie could be in zero to many shows.
11. An usher could usher in zero to many shows.

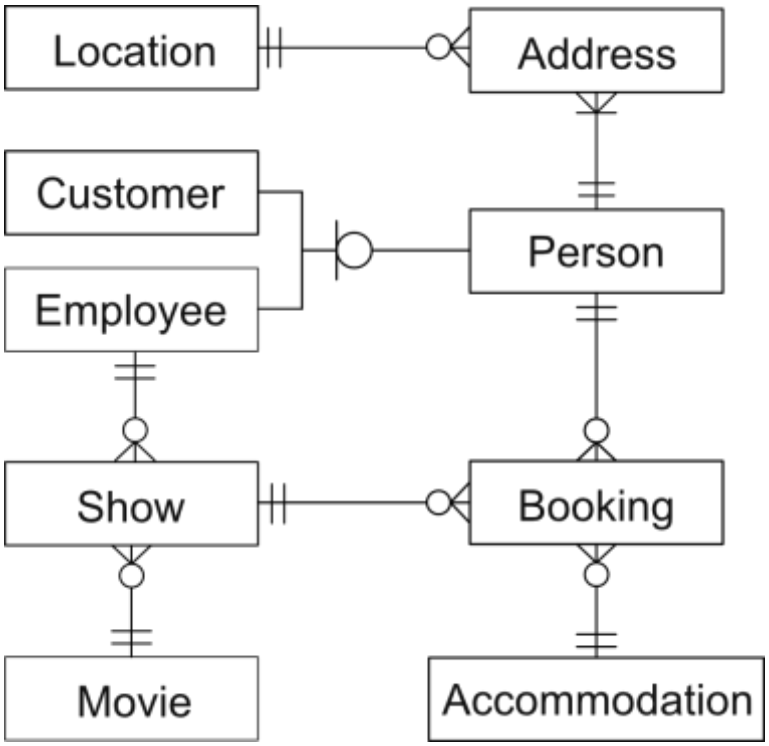


Figure 4: Minimized ER diagram to clarify the relationship between entities

The entity full entity relationship diagram with relation with attributes will look as follows:

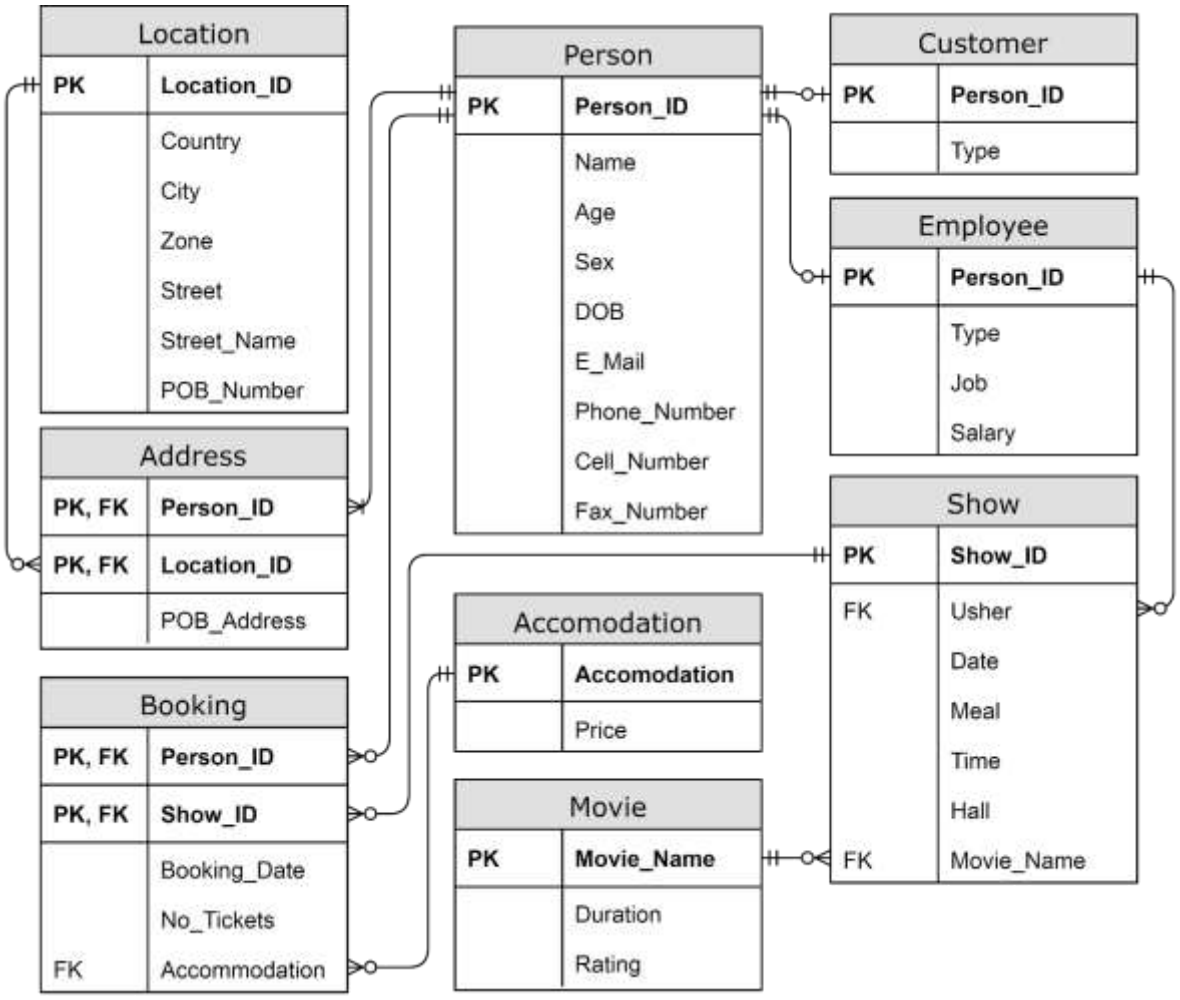


Figure 5: ER Diagram After Normalization

6. Data Dictionaries:

After normalization the same entities are now divided within different entities which will finally form a functioning database system, so the entities and its attributes are now listed with detailed constraint specification for each attribute in the tables:

Table 4: Data-Dictionary for Location table

Entity	Attributes	Datatype	Length	Description	Constraints
Location	Location_ID	NUMBER	10	Location identifier	PRIMARY_KEY
	Country	VARCHAR2	20	Country of person	NOT_NULL
	Zone	VARCHAR2	20	Zone of person	NOT_NULL
	City	VARCHAR2	20	City of person	NOT_NULL
	Street	VARCHAR2	20	Street of person	NOT_NULL
	Street_Name	VARCHAR2	20	street name of the person	NOT_NULL

Table 5: Data-Dictionary for the Address table

Entity	Attributes	Datatype	Length	Description	Constraints
Address	*Person_ID**	NUMBER	10	Unique Identifier of the person	PRIMARY_KEY, FOREIGN_KEY
	*Location_ID**	NUMBER	10	Unique Identifier of the location	PRIMARY_KEY, FOREIGN_KEY
	POB_Address	VARCHAR2	1	The mailing address of the person	

Table 6: Data-Dictionary for Person table

Entity	Attributes	Datatype	Length	Description	Constraints
Person	Person_ID	NUMBER	10	Unique Identifier of the person	PRIMARY_KEY
	Name	VARCHAR2	20	Name of person	NOT_NULL
	Age	NUMBER	5	Age of person	NOT_NULL
	Sex	VARCHAR2	20	Sex of person	NOT_NULL
	DOB	DATE		DOB of person	
	E_mail	VARCHAR2	50	e-mail of person	UNIQUE
	Phone_No	NUMBER	10	phone number of the person	
	Cell_No	NUMBER	10	the cell number of the person	UNIQUE
	Fax_No	NUMBER	10	fax number of the person	

Table 7: Data-Dictionary for Employee table

Entity	Attributes	Datatype	Length	Description	Constraints
Employee	*Person_ID**	NUMBER	10	Unique Identifier of the person	PRIMARY_KEY
	Job	VARCHAR2	20	Job of person	NOT_NULL
	Salary	NUMBER	10	Salary of person	NOT_NULL

Table 8: Data-Dictionary for the Customer table

Entity	Attributes	Datatype	Length	Description	Constraints
Customer	*Person_ID**	NUMBER	10	Unique Identifier of the person	PRIMARY_KEY
	Type	VARCHAR2	20	Type of person	DEFAULT = "New Customer"

Table 9: Data-Dictionary for Booking table

Entity	Attributes	Datatype	Length	Description	Constraints
Booking	*Person_ID**	NUMBER	10	Unique Identifier of the person	PRIMARY_KEY, FOREIGN_KEY
	*Show_ID**	NUMBER	10	Unique Identifier for the show	PRIMARY_KEY, FOREIGN_KEY
	No_Tickets	NUMBER	10	Number of tickets booked	DEFAULT=1
	Booking_Date	DATE		Date of booking	NOT_NULL
	*Accommodation**	VARCHAR	20	Accommodation type	NOT_NULL

Table 10: Data-Dictionary for Accommodation table

Entity	Attributes	Datatype	Length	Description	Constraints
Accommodation	Accommodation	VARCHAR2	20	Accommodation type	PRIMARY_KEY
	Price	NUMBER	10	Price of each ticket	DEFAULT=0

Table 11: Data-Dictionary for Movie table

Entity	Attributes	Datatype	Length	Description	Constraints

Movie	Movie_Name	VARCHAR2	20	Name of the movie	PRIMARY_KEY
	Duration	NUMBER	10	Duration of the movie in minutes	
	Rating	NUMBER	10	Rating of the movie out of 10	

Table 12: Data-Dictionary for Show table

Entity	**Attributes**	**Datatype**	**Length**	**Description**	**Constraints**
Show	Show_ID	NUMBER	10	Unique Identifier for the show	PRIMARY_KEY
	*Movie_Name**	VARCHAR	20	Name of the movie	FOREIGN_KEY
	Meal	VARCHAR	20	Meal served in the show	
	*Usher**	NUMBER	10	An employee who ushers the show	FOREIGN_KEY
	Date	DATE		Date of show	NOT_NULL
	Time	NUMBER	10	Time of show in military time(24-hour)	NOT_NULL
	Hall	VARCHAR	20	Hall of show	NOT_NULL

Database Implementation

1. Create User

To create various tables, add data on the tables and to run queries, a user must be created at first. A user is created to test the database design as follows:

```
CONNECT sys as sysdba;
CREATE USER ShresthaRajat IDENTIFIED BY pwrd;
ALTER USER ShresthaRajat ACCOUNT UNLOCK;
GRANT resource To ShresthaRajat;
GRANT create session TO ShresthaRajat;
CONNECT ShresthaRajat
```

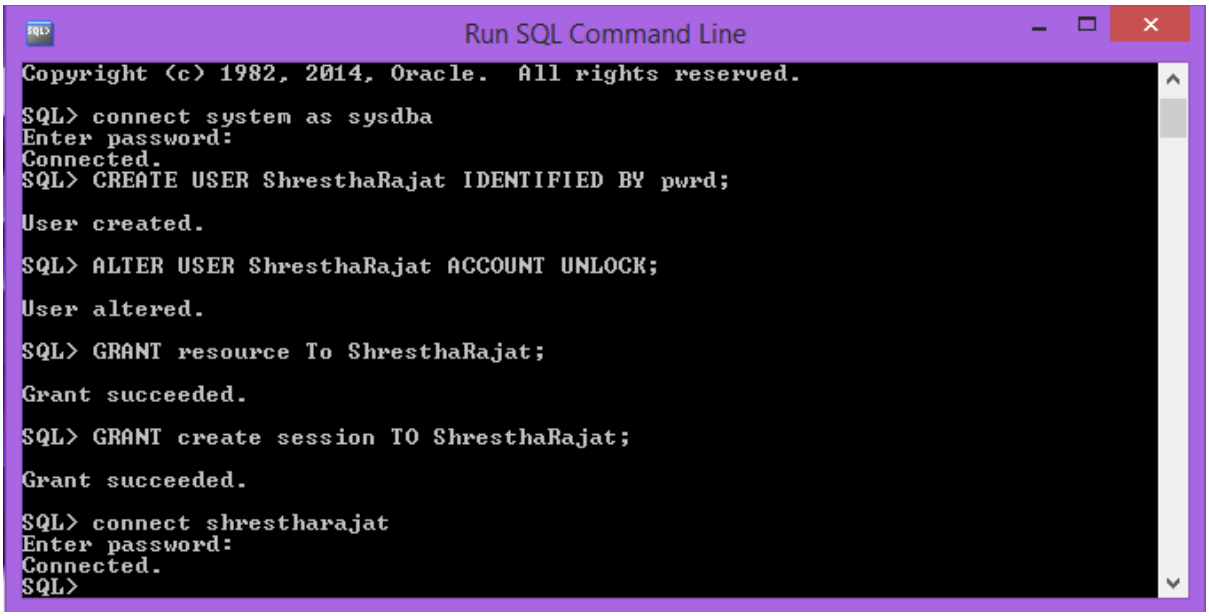


Figure 6: Creating a user

2. Table Generation

All of the tables are created at Oracle Command line, The tables are created in such an order that all of the tables will have its relation intact:

2.1. Person

```
CREATE TABLE Person(
    Person_ID NUMBER(10) NOT NULL,
    Name VARCHAR2(20) NOT NULL,
    Age  NUMBER(5) NOT NULL,
    sex VARCHAR2(20) NOT NULL,
    DOB DATE,
    E_mail VARCHAR2(50) UNIQUE,
    Cell_No NUMBER(10) UNIQUE,
    Fax_No NUMBER(10),
    Phone_No NUMBER(10),
    CONSTRAINT Person_PK
       PRIMARY KEY (Person_ID)
);
```

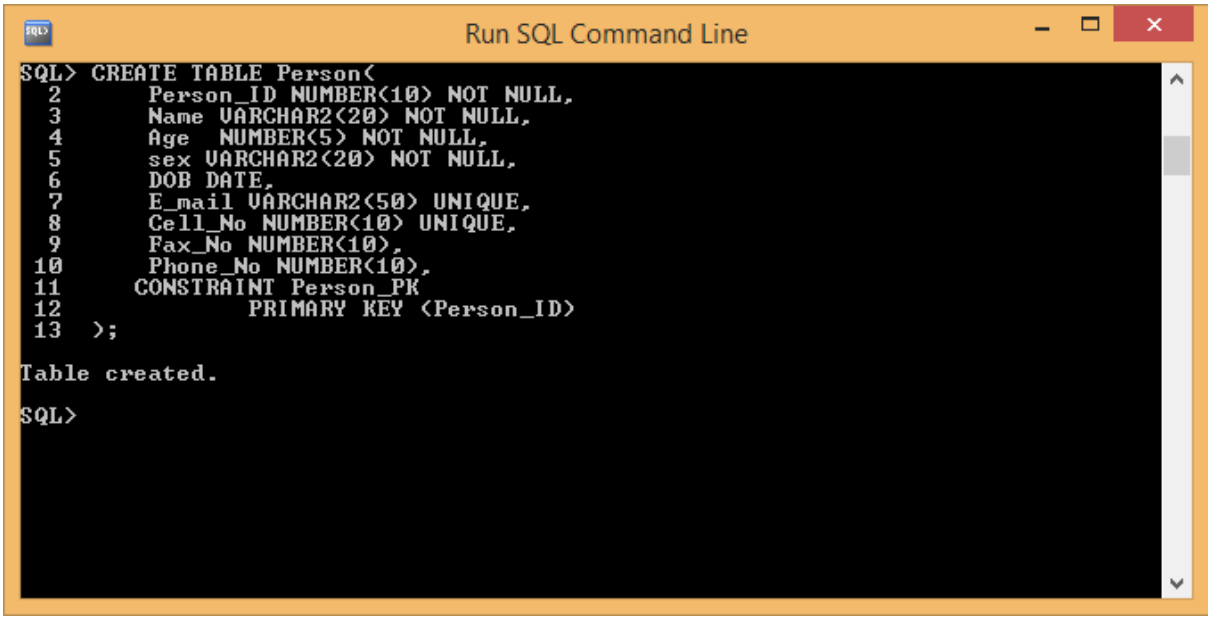


Figure 7:Creating Table Person

2.2. Location

```
CREATE TABLE Location(
    Location_ID NUMBER(10) NOT NULL,
    Country VARCHAR2(20) NOT NULL,
    Zone VARCHAR2(20) NOT NULL,
    City VARCHAR2(20) NOT NULL,
    Street VARCHAR2(20) NOT NULL,
    Street_Name VARCHAR2(20) NOT NULL,
    CONSTRAINT Location_PK
        PRIMARY KEY (Location_ID)
);
```

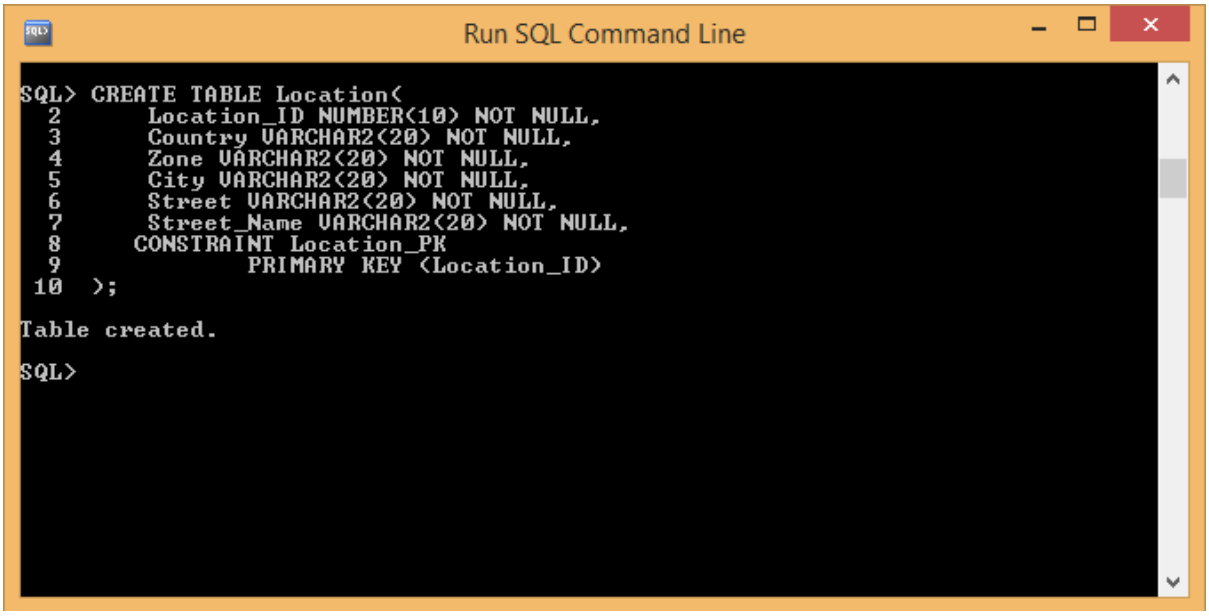


Figure 8:Creating Table Location

2.3. Address

```
CREATE TABLE Address(
      Person_ID NUMBER(10) NOT NULL,
      Location_ID NUMBER(10) NOT NULL,
      POB_Address VARCHAR2(1) DEFAULT 'T',
      CONSTRAINT Address_PK
            PRIMARY KEY (Person_ID, Location_ID),
      CONSTRAINT Address_Location_FK
            FOREIGN KEY (Location_ID)
            REFERENCES Location (Location_ID),
      CONSTRAINT Address_Person_FK
            FOREIGN KEY (Person_ID)
            REFERENCES Person (Person_ID)
);
```

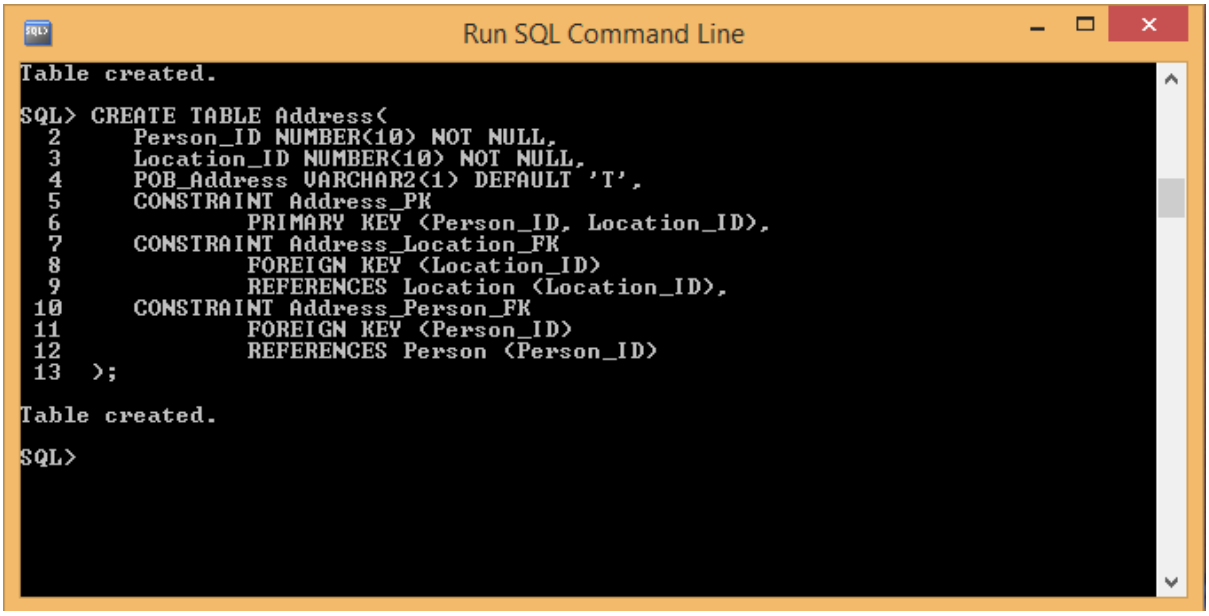


Figure 9:Creating Table Address

2.4. Customer

```
CREATE TABLE Customer(
      Person_ID NUMBER(10) NOT NULL,
      Type VARCHAR2(20) DEFAULT 'New',
      CONSTRAINT Customer_PK
            PRIMARY KEY (Person_ID),
      CONSTRAINT Customer_Person_FK
            FOREIGN KEY (Person_ID)
            REFERENCES Person (Person_ID)
);
```

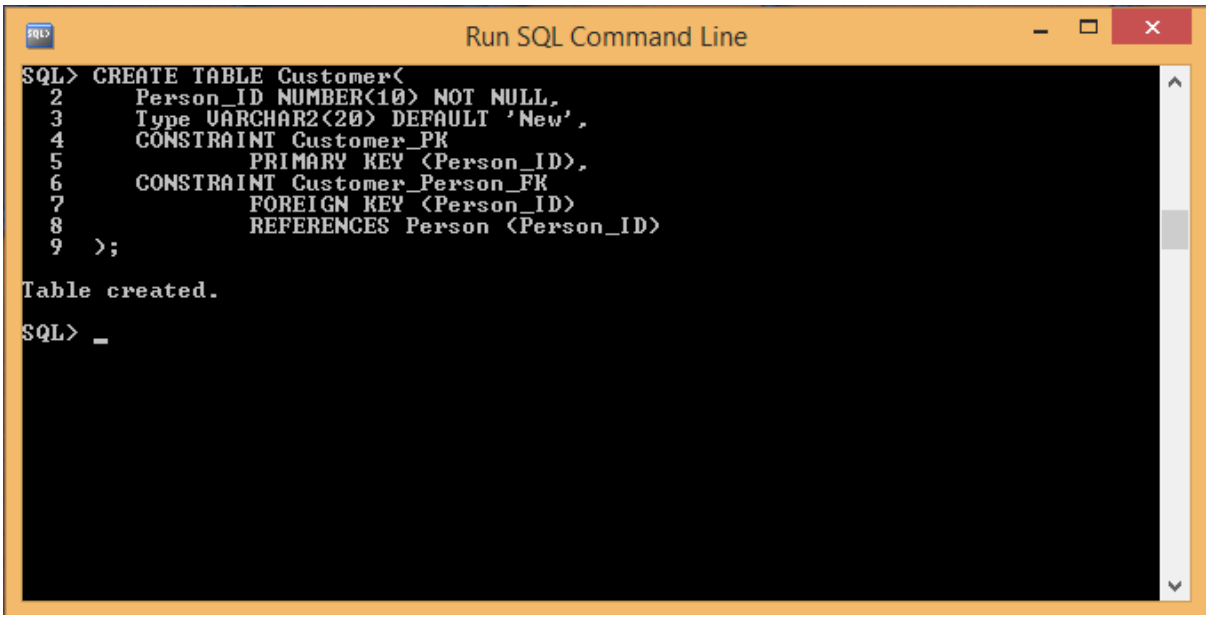


Figure 10:Creating Table Customer

2.5. Employee

```
CREATE TABLE Employee(
      Person_ID NUMBER(10) NOT NULL,
      Job VARCHAR2(20) NOT NULL,
      Salary NUMBER(10) NOT NULL,
      CONSTRAINT Employee_PK
            PRIMARY KEY (Person_ID),
      CONSTRAINT Employee_Person_FK
            FOREIGN KEY (Person_ID)
            REFERENCES Person (Person_ID)
);
```

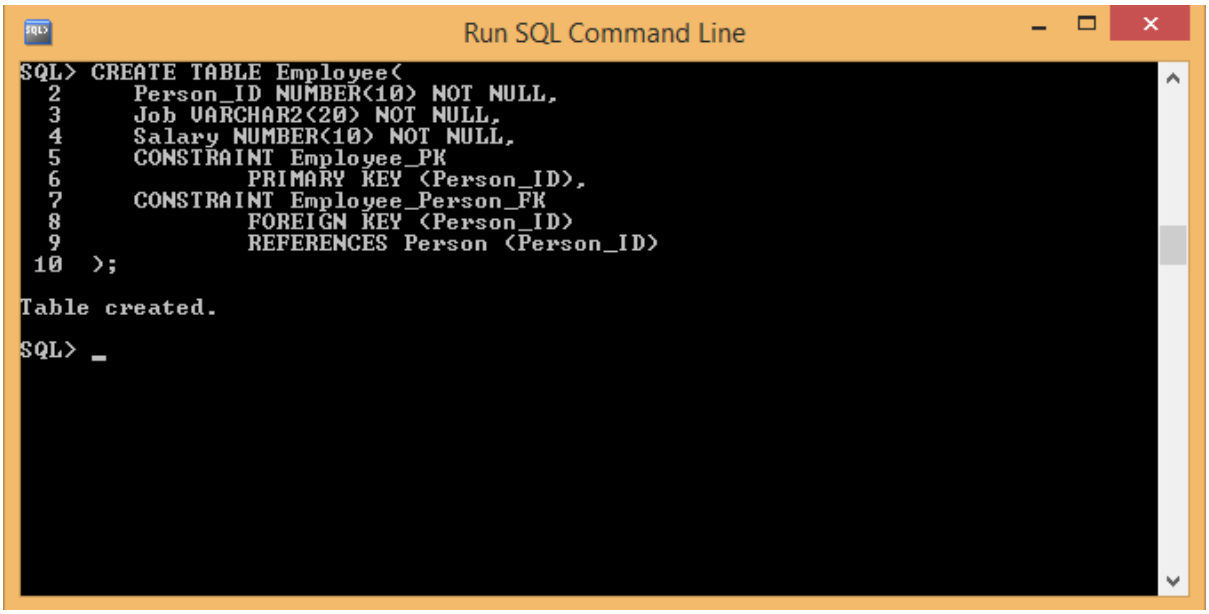


Figure 11: Creating Table Employee

2.6. Movie

```
CREATE TABLE Movie(
      Movie_Name VARCHAR2(20) NOT NULL,
      Rating NUMBER(10) ,
      Duration NUMBER(10) ,
      CONSTRAINT Movie_PK
            PRIMARY KEY (Movie_Name)
);
```

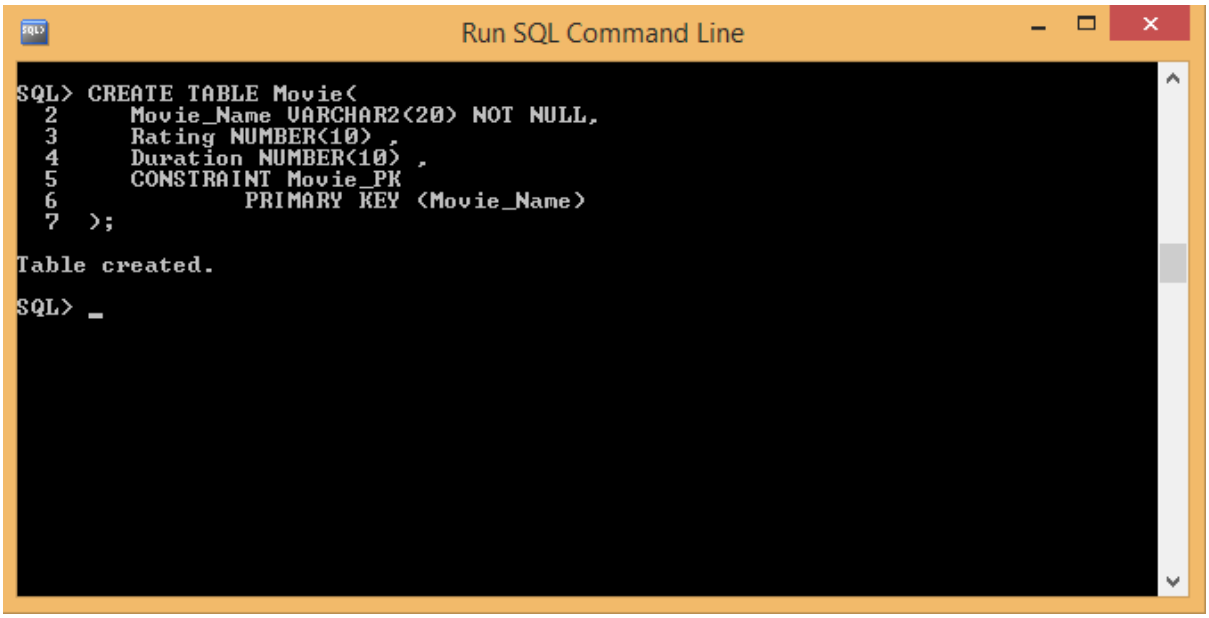


Figure 12: Creating Table Movie

2.7. Accommodation

```
CREATE TABLE Accommodation(
      Accommodation VARCHAR2(20) NOT NULL,
      Price NUMBER(10) NOT NULL,
      CONSTRAINT Accommodation_PK
            PRIMARY KEY (Accommodation)
);
```

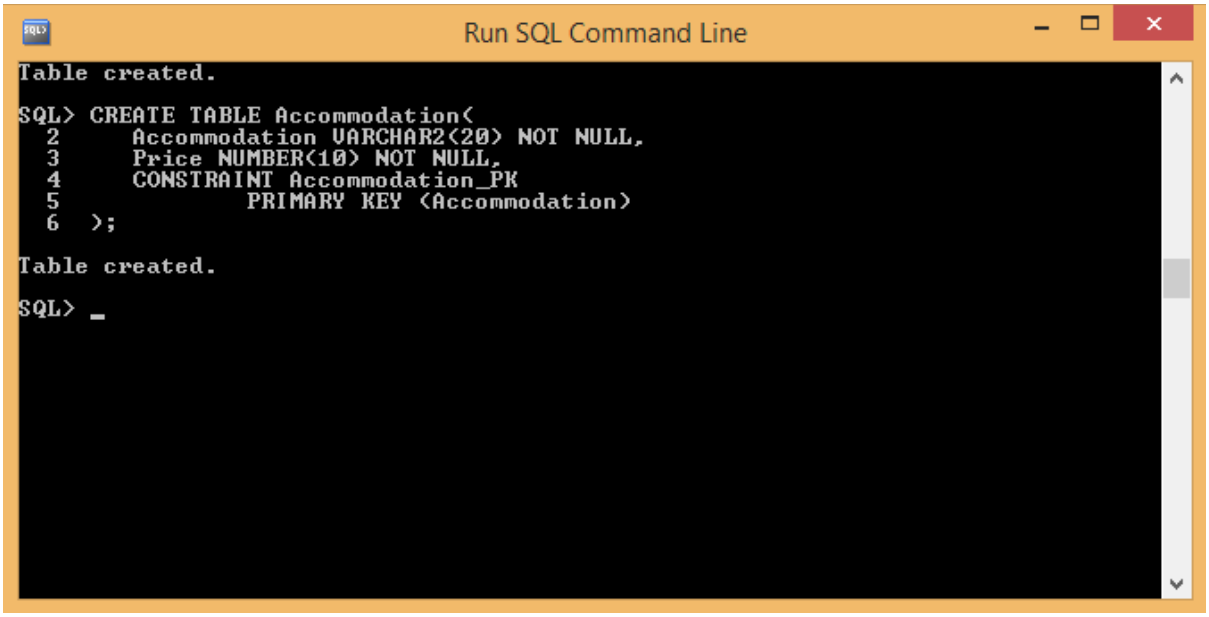


Figure 13: Creating Table Accommodation

2.8. Show

```
CREATE TABLE Show(
      Show_ID NUMBER(10) NOT NULL,
      Meal VARCHAR2(20) NOT NULL,
      Usher NUMBER(10) NOT NULL,
      Show_Date DATE NOT NULL,
      Show_Time NUMBER(10) NOT NULL,
      Hall VARCHAR2(20) NOT NULL,
      Movie_Name VARCHAR2(20) NOT NULL,
      CONSTRAINT Show_PK
            PRIMARY KEY (Show_ID),
      CONSTRAINT Show_Employee_FK
            FOREIGN KEY (Usher)
            REFERENCES Employee (Person_ID),
      CONSTRAINT Show_Movie_FK
            FOREIGN KEY (Movie_Name)
            REFERENCES Movie (Movie_Name)
);
```

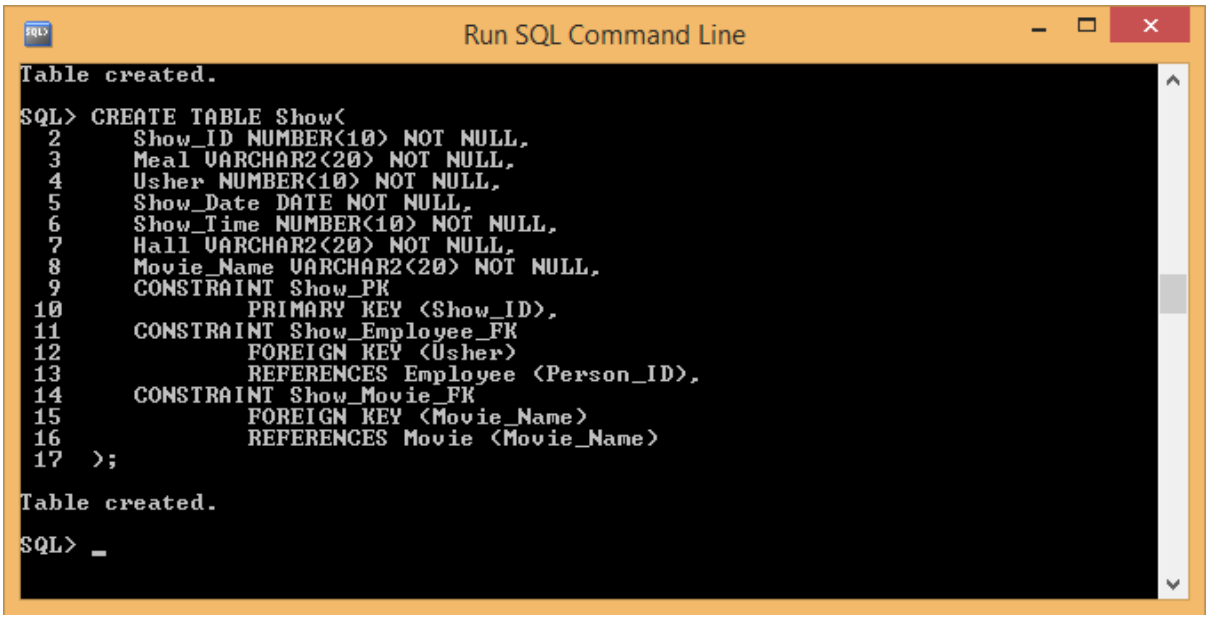


Figure 14: Creating Table Show

2.9. Booking

```
CREATE TABLE Booking(
      Person_ID NUMBER(10) NOT NULL,
      Show_ID NUMBER(10) NOT NULL,
      No_Tickets NUMBER(10) NOT NULL,
      Booking_Date DATE NOT NULL,
      Accommodation VARCHAR2(20) NOT NULL,
      CONSTRAINT Booking_PK
            PRIMARY KEY (Person_ID, Show_ID),
      CONSTRAINT Booking_Person_FK
            FOREIGN KEY (Person_ID)
            REFERENCES Person (Person_ID),
      CONSTRAINT Booking_Show_FK
            FOREIGN KEY (Show_ID)
            REFERENCES Show (Show_ID),
      CONSTRAINT Booking_Accommodation_FK
            FOREIGN KEY (Accommodation)
            REFERENCES Accommodation (Accommodation)
);
```

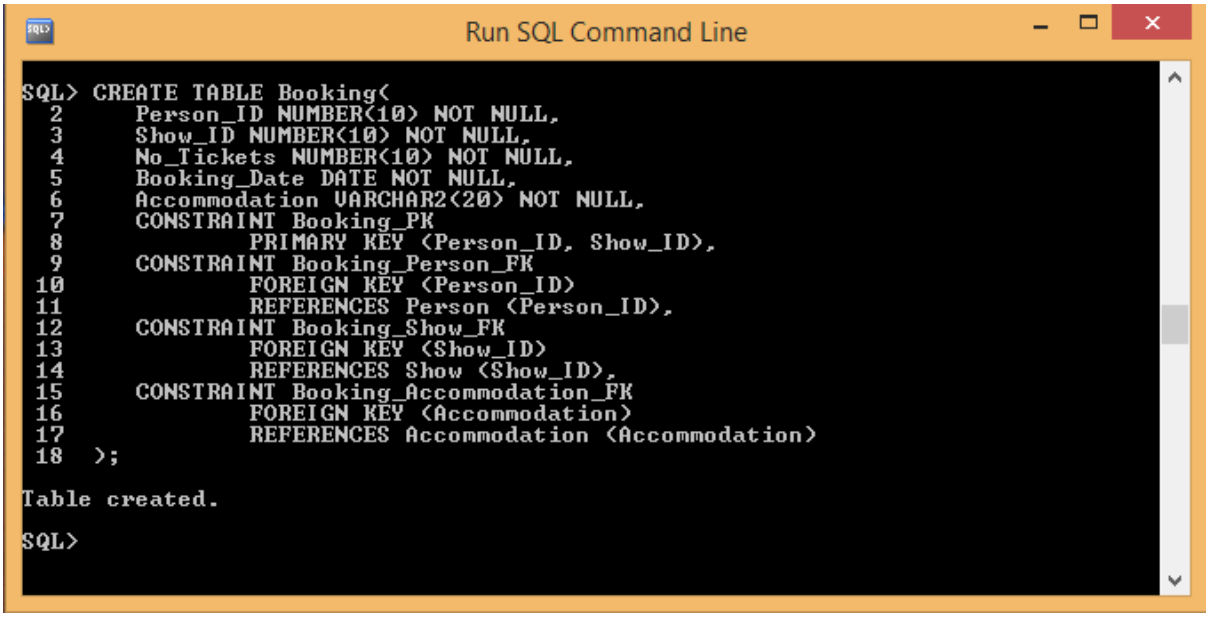


Figure 15: Creating Table Booking

2.10. Dropping the tables:

The booking table must be dropped at first because it is not referenced by any other entities in the relation. Then the show should be dropped as it was only referenced by booking. Accommodation and Movie can be dropped after the show and booking tables are dropped. The address table stores the address info of the person by calling the Person_ID and the Location_ID so it must be removed. Then location table can also be removed. As the employee and Customer are subtypes of person so Employee and customer should be removed before removing person. Simply the command to drop all of the tables without any error will be:

```
DROP TABLE Booking;
DROP TABLE Show;
DROP TABLE Movie;
DROP TABLE Accommodation;
DROP TABLE Address;
DROP TABLE Location;
DROP TABLE Employee;
DROP TABLE Customer;
DROP TABLE Person;
```

This command is stored as an sql file so that it will be easier to drop all of the tables by simply calling the file in the SQL command line.

3. Populating database tables

The database must have some sort of data so that we can understand it's functionality. So to understand the inner mechanics of the database the tables are populated by direct inserting values. The other methods of inserting values are further discussed in the appendix.

3.1. Person

```
INSERT INTO Person VALUES (1, 'Rajat Shretha', 19, 'male', '11-mar-1999',
'rajat@mail.com', 984399999, 909090, 89891232 );
INSERT INTO Person VALUES (2, 'Saubhagya Sharma', 19, 'male', '11-dec-
1998', 'sau@mail.com', 984399998, 909091, 89891233 );
INSERT INTO Person VALUES (3, 'Saurav Khadka', 19, 'male', '16-feb-1999',
'saurav@mail.com', 984399997, 909092, 89891234 );
INSERT INTO Person VALUES (4, 'Anu Shretha', 21, 'female', '11-mar-1999',
'anu@mail.com', 984399996, 909093, 89891235 );
INSERT INTO Person VALUES (5, 'Amira Shakya', 14, 'female', '11-dec-
1998', 'Amira@mail.com', 984399995, 909094, 89861233 );
```

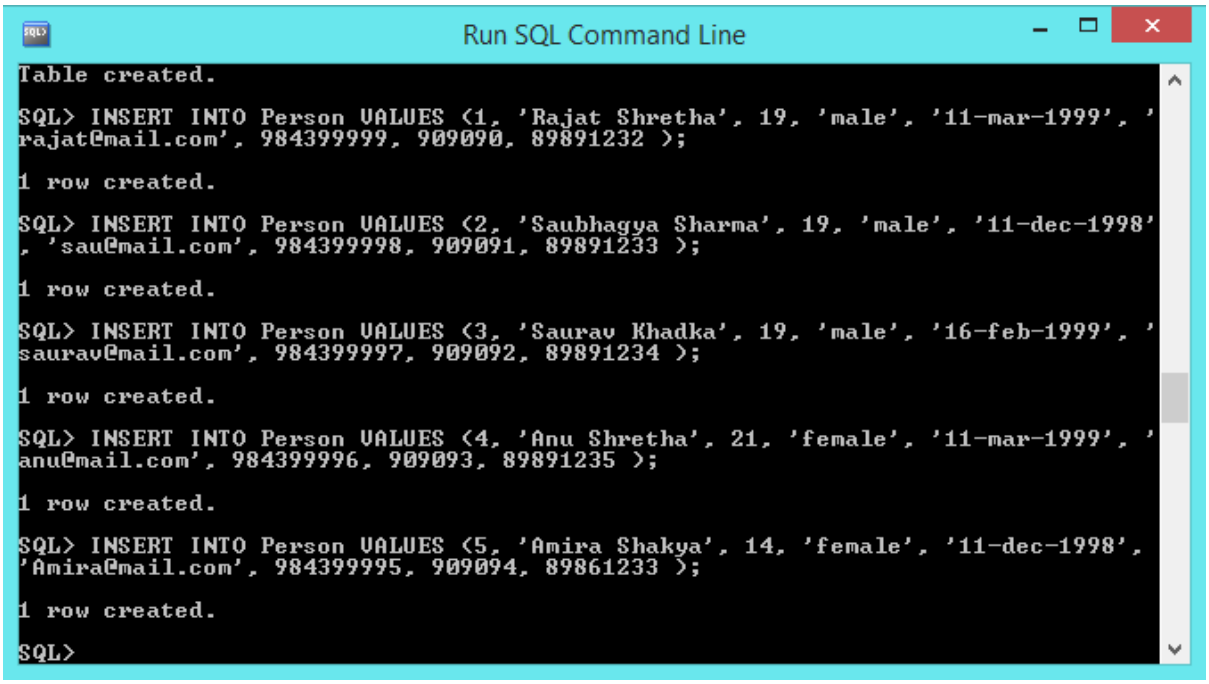


Figure 16: Populating Table Person

3.2. Location

```
INSERT INTO Location VALUES (1, 'Nepal', 'Bagmati', 'Kathmandu',
'Samakhushi', 'Town-planning Rd');
INSERT INTO Location VALUES (2, 'Nepal', 'Bagmati', 'Kathmandu',
'Baneshwor', 'Bhawan');
INSERT INTO Location VALUES (3, 'Nepal', 'Bagmati', 'Kathmandu',
'Gyaneshwor', 'Kilagal');
INSERT INTO Location VALUES (4, 'Nepal', 'Bagmati', 'Kathmandu',
'Soyambu', 'Bhagwan Pau');
```

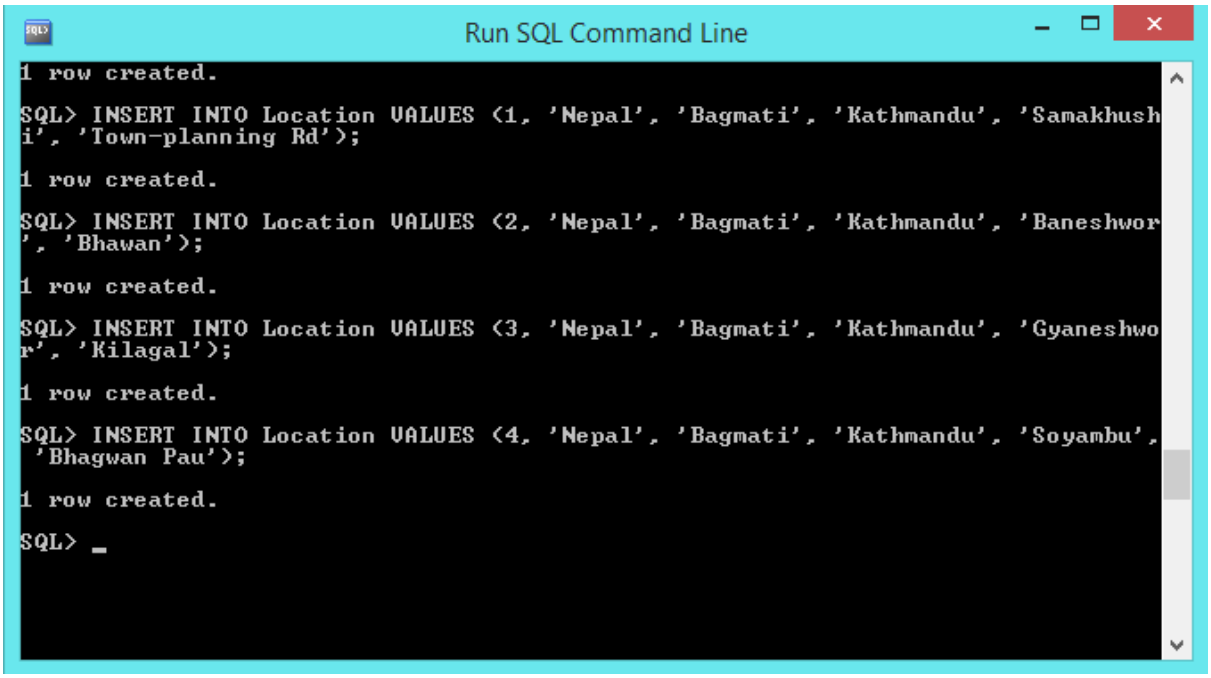


Figure 17: Populating Table Location

3.3. Address

```
INSERT INTO Address VALUES (1, 1, 'y');
INSERT INTO Address VALUES (2, 2, 'y');
INSERT INTO Address VALUES (3, 3, 'y');
INSERT INTO Address VALUES (1, 4, 'n');
INSERT INTO Address VALUES (4, 4, 'y');
INSERT INTO Address VALUES (5, 4, 'y');
```

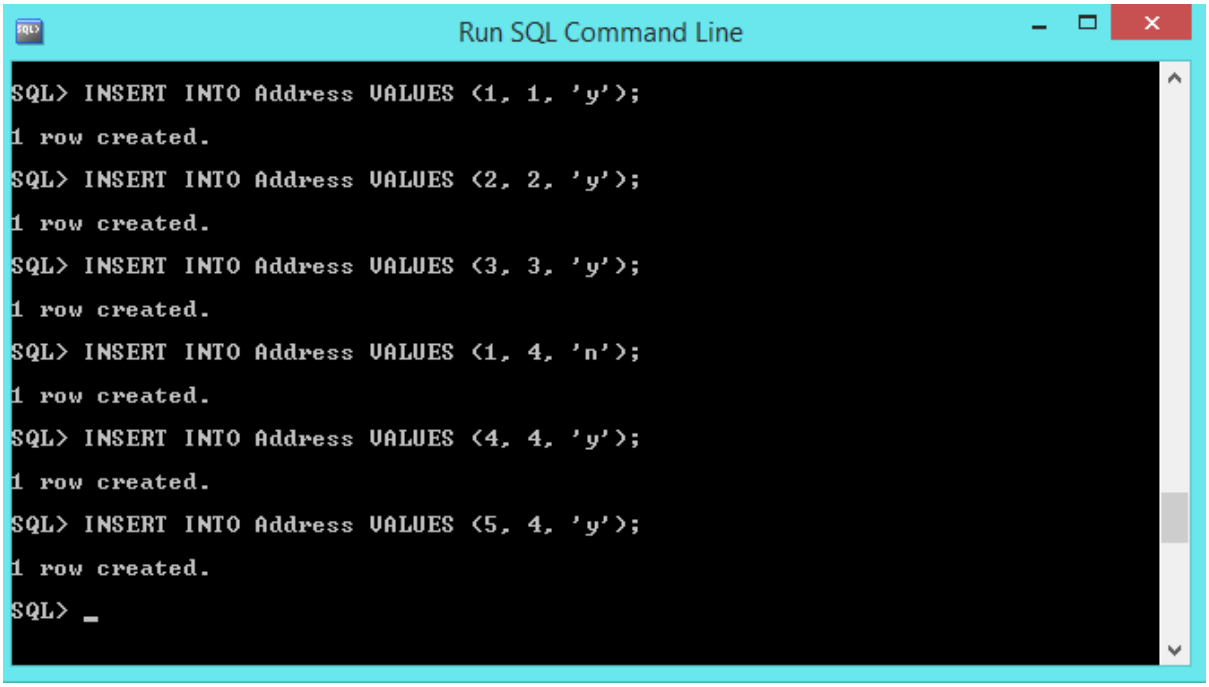


Figure 18: Populating Table Address

3.4. Employee

```
INSERT INTO Employee VALUES (1, 'Usher', 20000);
INSERT INTO Employee VALUES (2, 'Usher', 20000);
INSERT INTO Employee VALUES (5, 'Accountant', 40000);
```

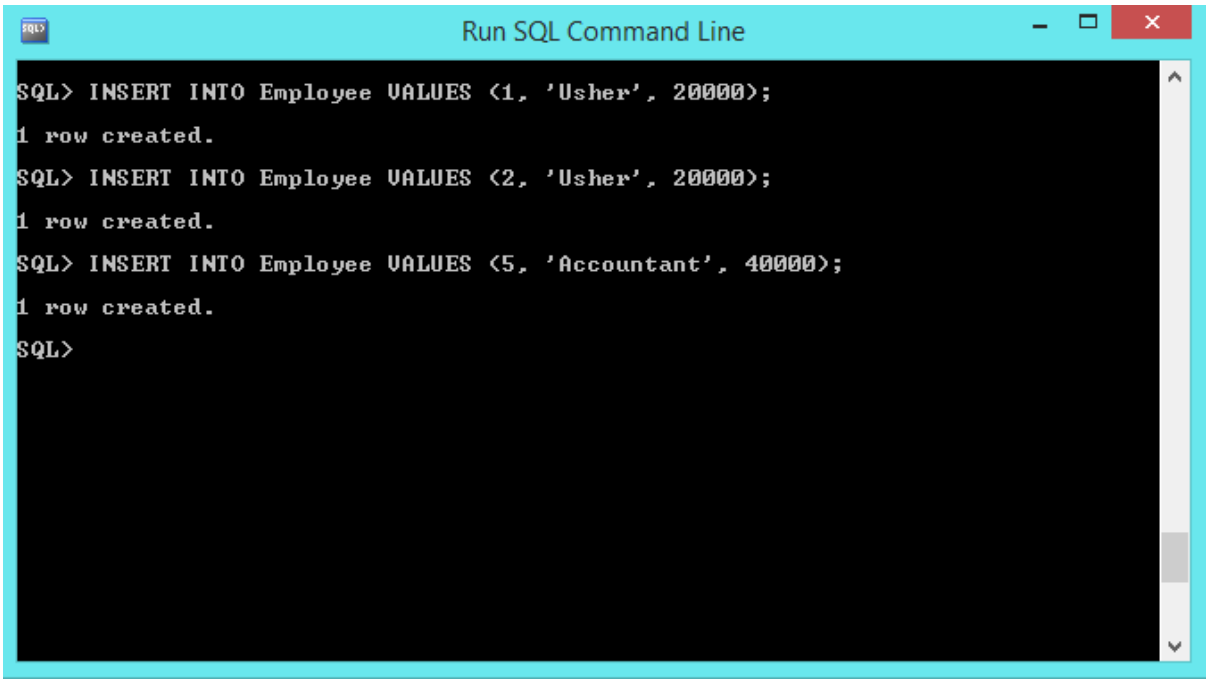


Figure 19: Populating Table Employee

3.5. Customer

```
INSERT INTO Customer VALUES (1, 'Old Customer');
INSERT INTO Customer VALUES (2, 'New Customer');
INSERT INTO Customer VALUES (3, 'New Customer');
INSERT INTO Customer VALUES (4, 'New Customer');
INSERT INTO Customer VALUES (5, 'New Customer');
```

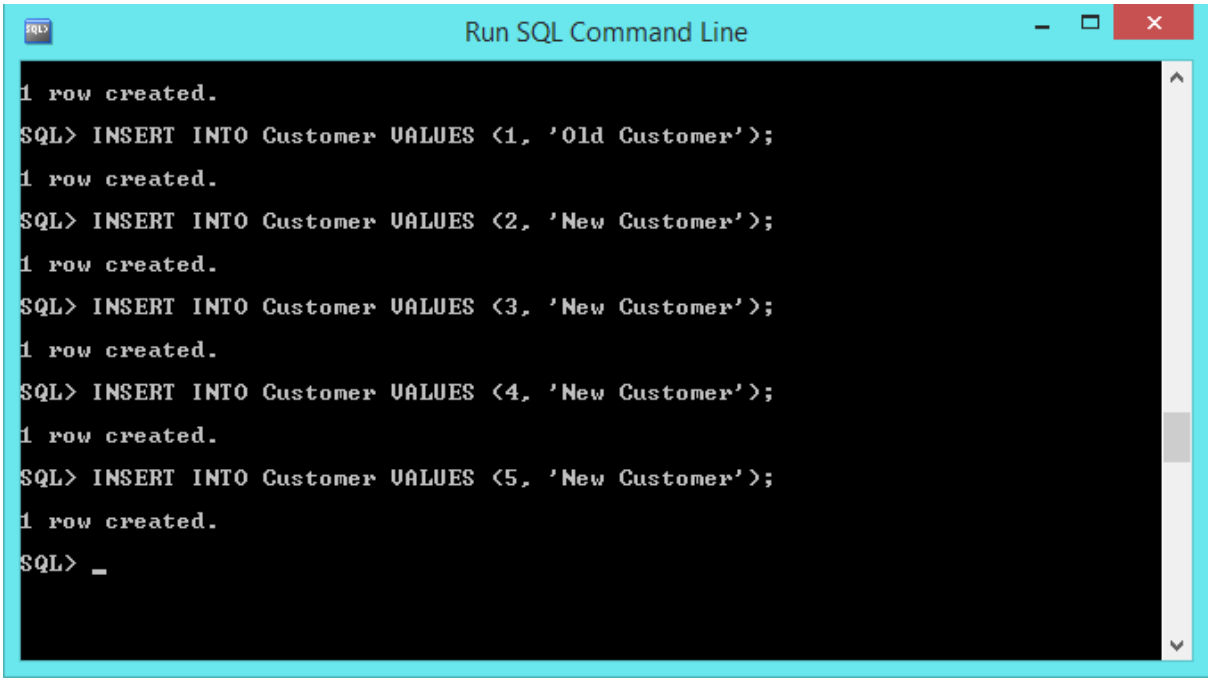


Figure 20: Populating Table Customer

3.6. Movie

```
INSERT INTO Movie VALUES ('Aquawoman', 9, 100);
INSERT INTO Movie VALUES ('Bohemien', 10, 130);
INSERT INTO Movie VALUES ('Sinatra', 7, 140);
```

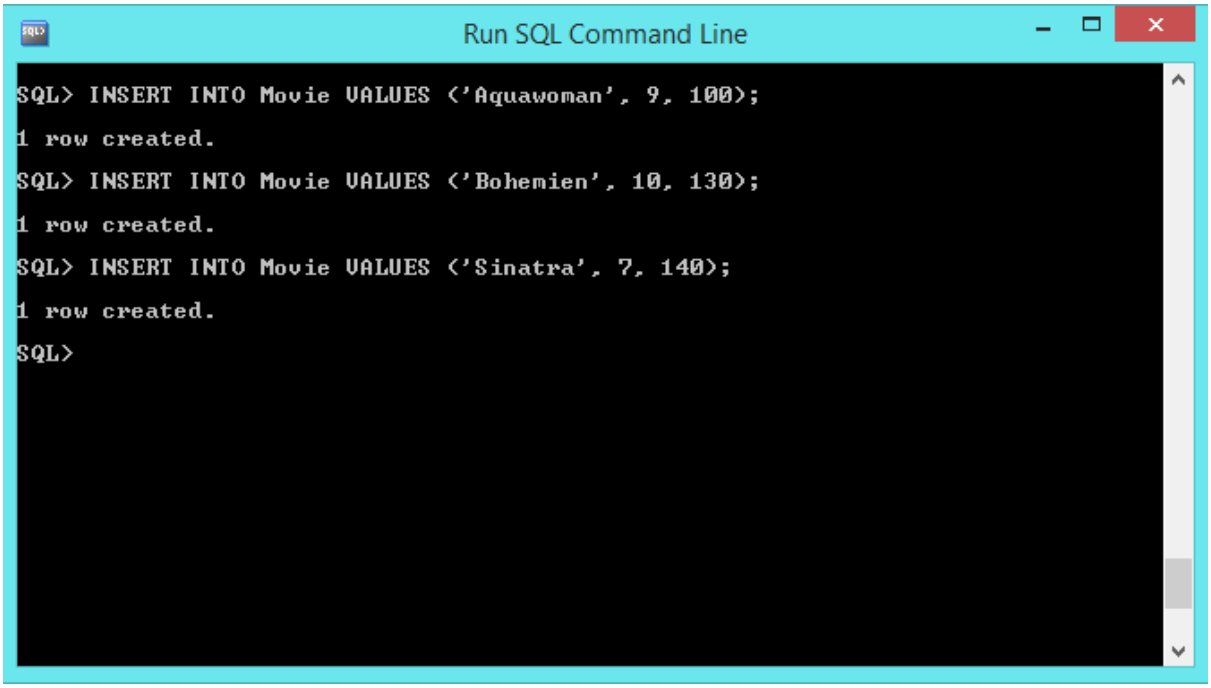


Figure 21: Populating Table Movie

3.7. Accommodation

```
INSERT INTO Accommodation VALUES ('Platinium', 500);
INSERT INTO Accommodation VALUES ('Gold', 400);
INSERT INTO Accommodation VALUES ('Silver', 300);
INSERT INTO Accommodation VALUES ('Exclusive', 0);
```

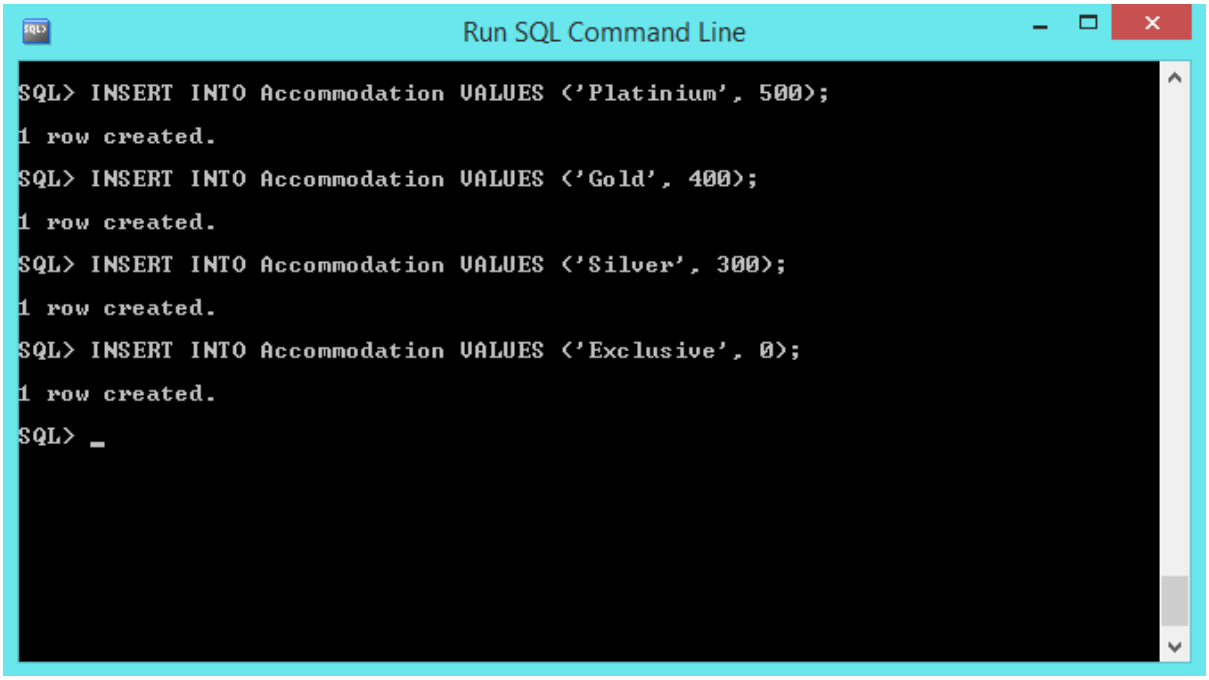


Figure 22: Populating Table Accommodation

3.8. Show

```
INSERT INTO Show VALUES (1, 'breakfast', 1, '11-jan-2019', 0900,
'A','Aquawoman');
INSERT INTO Show VALUES (2, 'dinner', 1, '11-jan-2019', 2100,
'B','Bohemien');
INSERT INTO Show VALUES (3, 'dinner', 1, '11-jan-2019', 2100,
'A','Sinatra');
INSERT INTO Show VALUES (4, 'lunch', 1, '12-jan-2019', 1200,
'B','Sinatra');
```

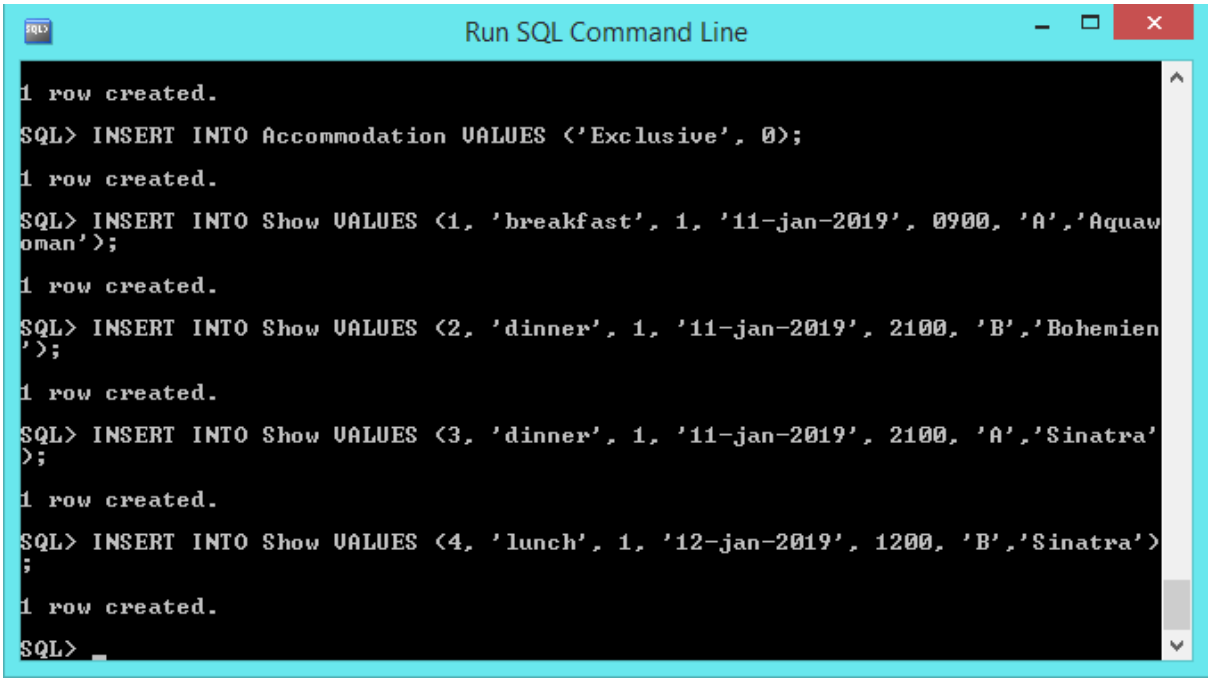


Figure 23: Populating Table Show

3.9. Booking

```
INSERT INTO Booking VALUES (1, 2, 1, '2-jan-2019', 'Exclusive');
INSERT INTO Booking VALUES (2, 2, 1, '3-jan-2019', 'Exclusive');
INSERT INTO Booking VALUES (3, 2, 1, '3-jan-2019', 'Silver');
INSERT INTO Booking VALUES (3, 1, 2, '3-jan-2019', 'Platinium');
INSERT INTO Booking VALUES (3, 3, 1, '3-jan-2019', 'Gold');
INSERT INTO Booking VALUES (1, 4, 1, '12-jan-2019', 'Exclusive');
```

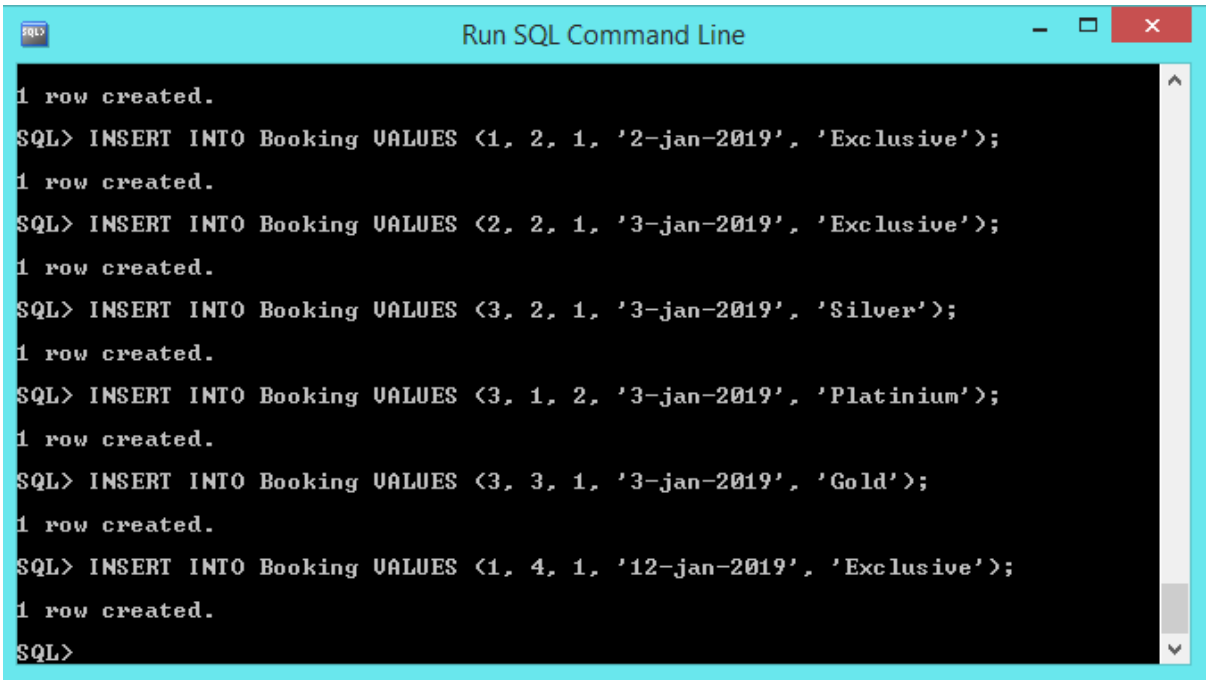


Figure 24: Populating Table Booking

(note: The detailed specification of each tables and the entries are discussed at the appendix)

Queries

1. Information Queries

1.1. List all customers, old and current

```
SELECT Customer.Person_Id, Customer.Type, Person.Name
FROM Customer
JOIN Person
ON Customer.Person_Id = Person.Person_Id;
```

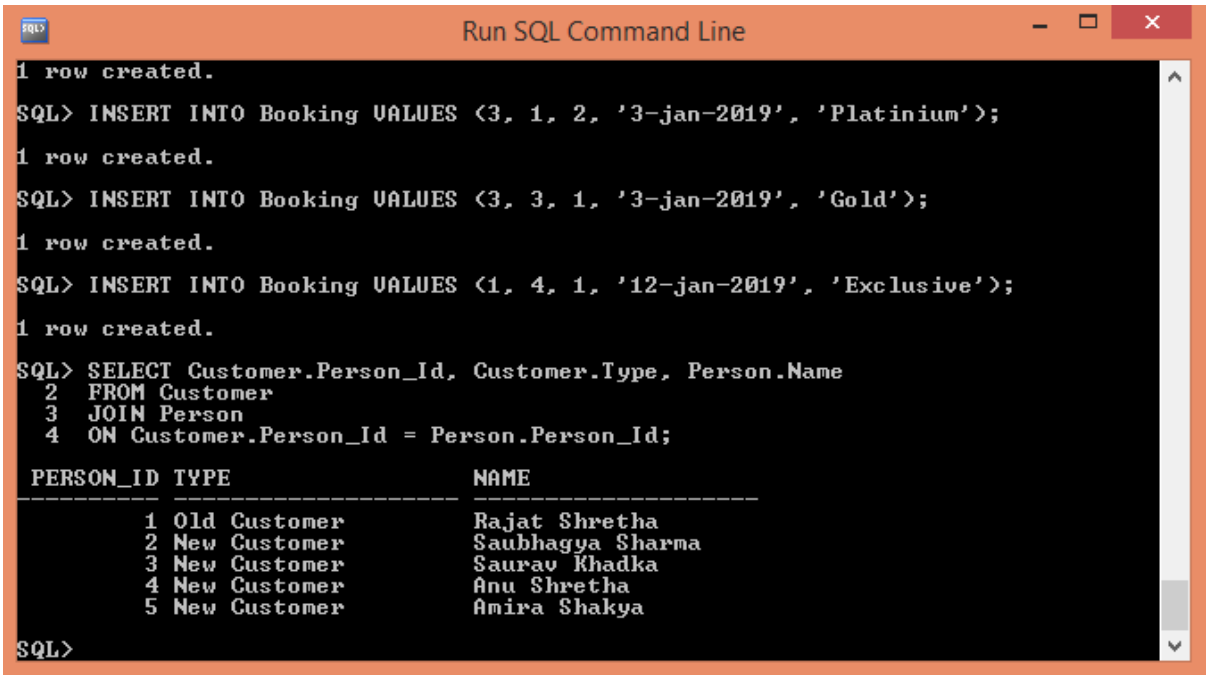


Figure 25: Information Query1: Customer listing

1.2. List all customers with all their addresses

```
SELECT Customer.Person_Id as Customer,
Location.City, Location.Street , Location.Street_Name
FROM Address
JOIN Customer
ON Address.Person_Id = Customer.Person_Id
JOIN Location
ON Address.Location_Id = Location.Location_ID;
```

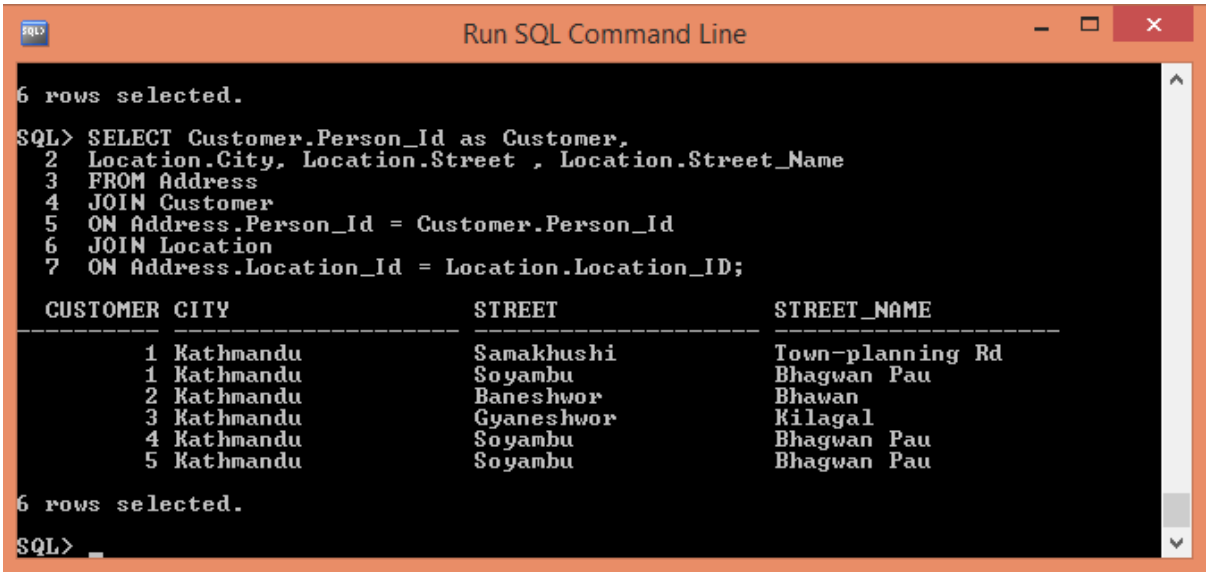


Figure 26: Information Query2: Customer and their addresses

1.3. For a given usher, find all the shows he/she ushered or will usher and the amount he/she got/will get for ushering the show.

```
SELECT Show.Show_Id, Show.Movie_Name, Show.Usher, Employee.Salary,
Person.Name
FROM Show
Join Employee
ON Show.Usher = Employee.Person_ID
Join Person
ON Show.Usher = Person.Person_ID
WHERE Show.Usher = 1;
```

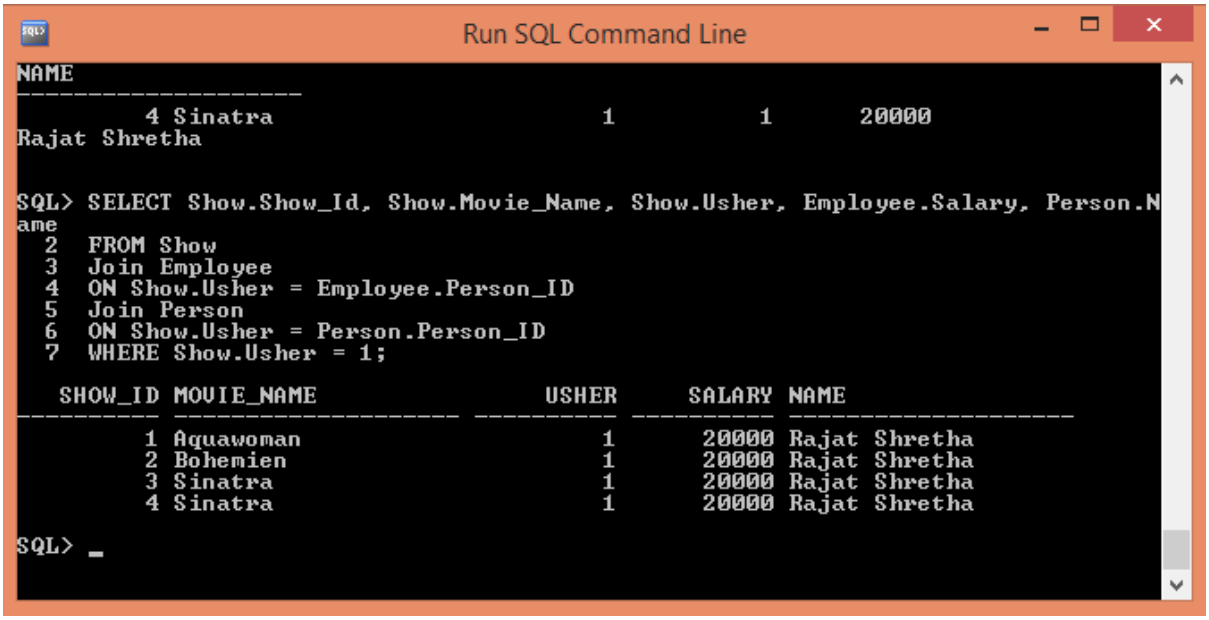


Figure 27: Information Query3: usher and shows with the salary

1.4. List all customers that are also ushers.

```
SELECT Customer.Person_ID as Customer,
Person.Name, Employee.Job
FROM Employee
JOIN Person
ON Employee.Person_ID = Person.Person_ID
INNER JOIN Customer
ON Employee.Person_Id = Customer.Person_Id
WHERE Employee.job = 'Usher';
```

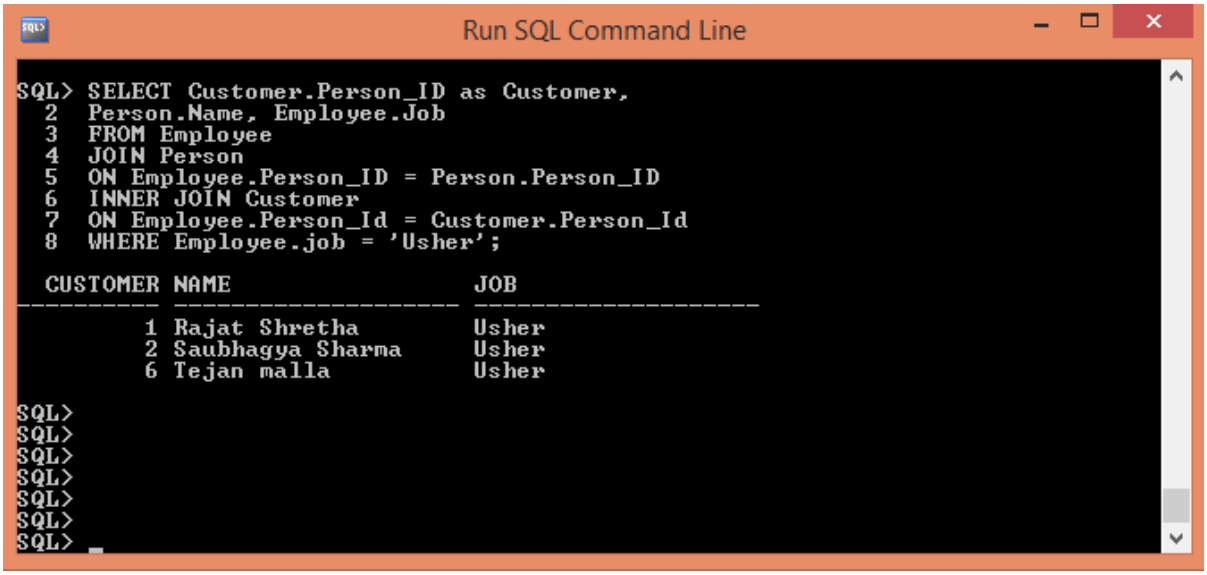


Figure 28: Information Query4: usher who are also customer

2. Transaction Queries

2.1. List all ushers that attended a show that had lunch in a given place.

```
SELECT Employee.Person_ID, Employee.Job, Person.Name, Show.Meal,
Show.Hall, Booking.Show_ID, Booking.Person_ID
FROM Booking
JOIN Employee
ON Booking.Person_ID = Employee.Person_ID
JOIN Person
ON Booking.Person_ID = Person.Person_ID
JOIN Show
ON Booking.Show_ID = Show.Show_ID
WHERE Show.Meal = 'lunch' and Show.Hall = 'B';
```

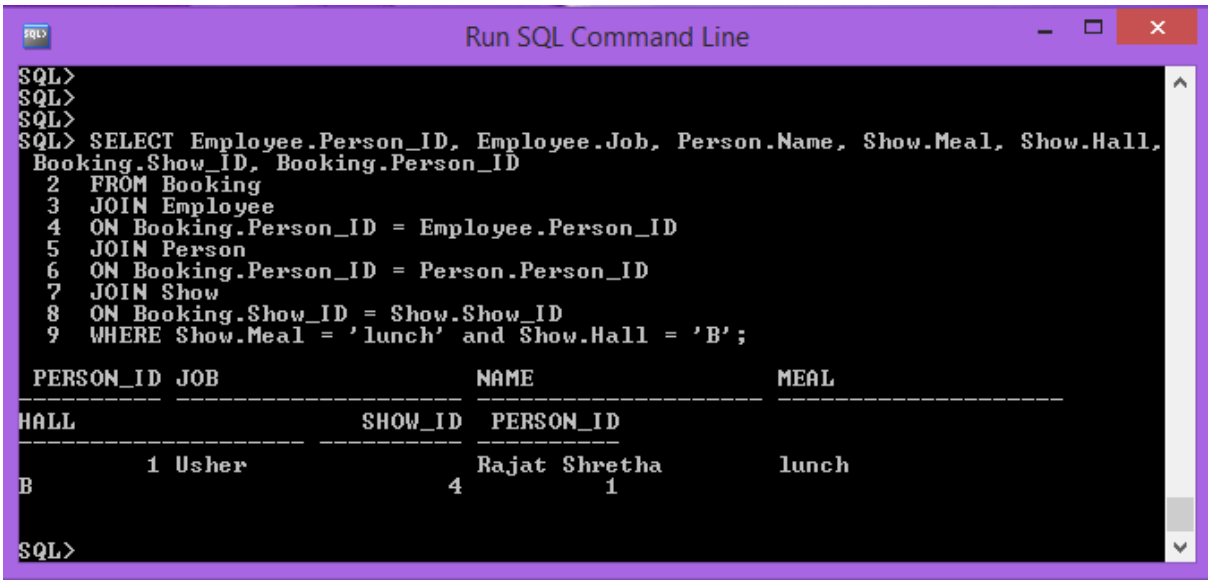


Figure 29: Transaction Query1: Usher show and lunch

2.2. List the shows that will have breakfast at a given place on a given date.

```
SELECT Show_ID, Movie_Name, Meal, Hall
FROM Show
WHERE Meal = 'breakfast' and Hall = 'A' and Show_Date = '11-jan-2019';
```

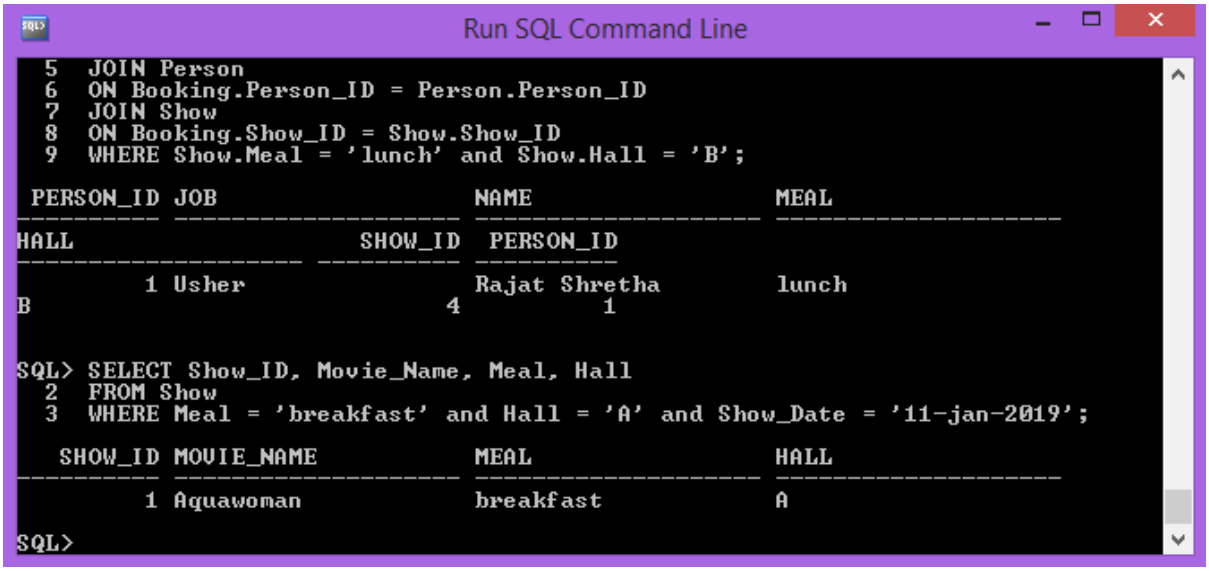


Figure 30: Transaction Query2: meal and place by show

2.3. List all employees that have worked as an usher or will work as an usher for a show or who have attended or will attend a show.

```
SELECT Employee.Person_ID, Employee.Job, Person.Name, Booking.Show_ID
FROM Employee
JOIN Person
ON Employee.Person_ID = Person.Person_ID
JOIN Booking
ON Employee.Person_ID = Booking.Person_ID;
```

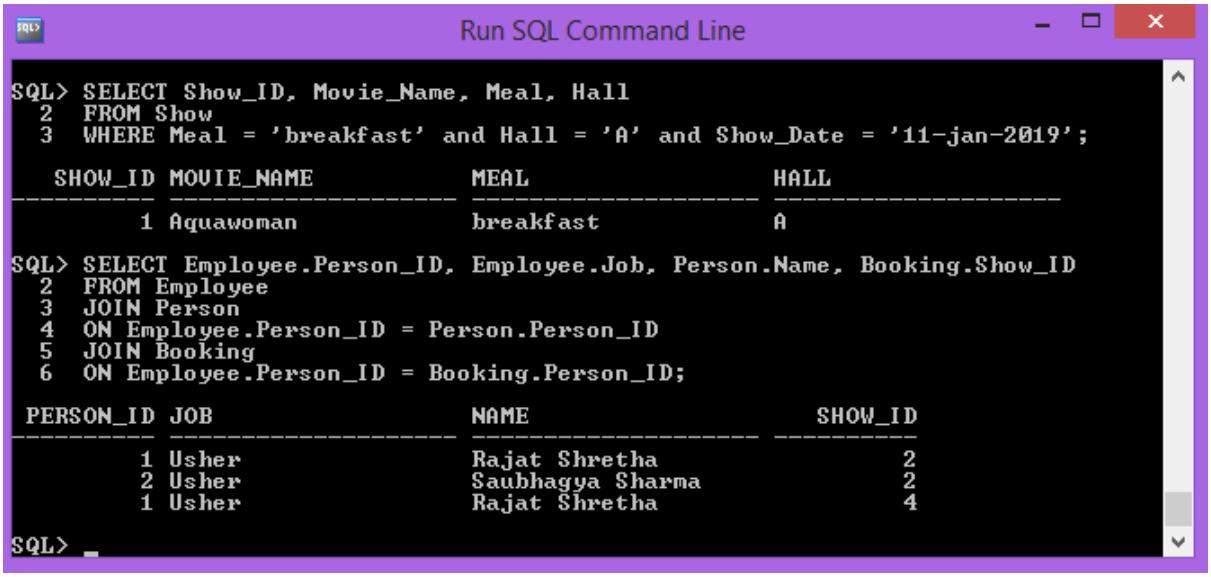


Figure 31: Transaction Query3: Employees working as a usher or booked a show

2.4. List all customers booked for a show starting later or on a given date.

```
SELECT Person.Person_Id, Person.Name, Booking.Show_ID, Show.Movie_Name
FROM Booking
JOIN Person
ON Booking.Person_Id = Person.Person_Id
JOIN Show
ON Booking.Show_ID = Show.Show_ID
WHERE Show_Date >= '12-jan-2019';
```

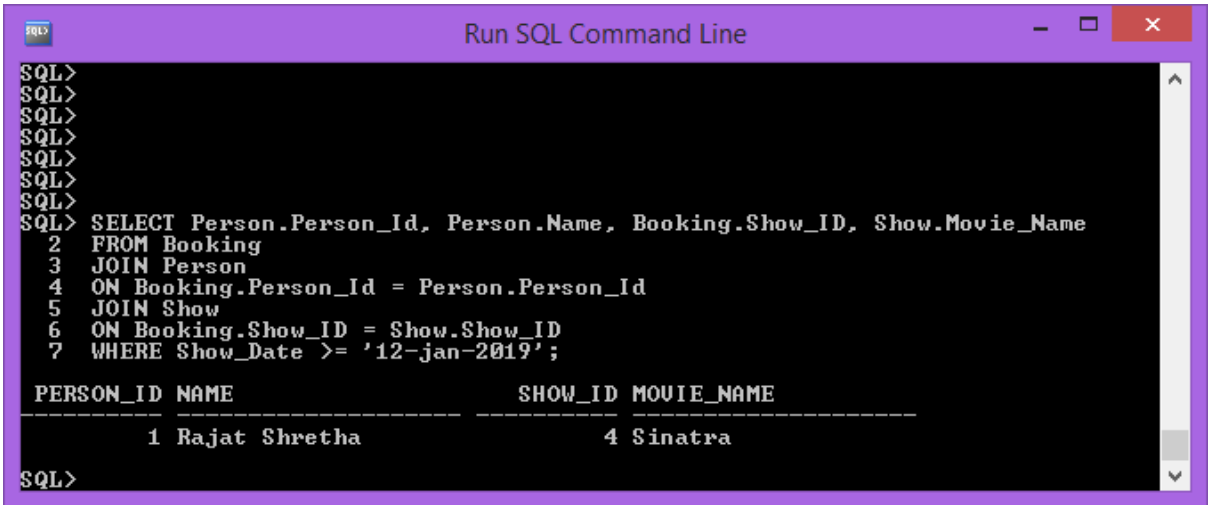


Figure 32: Transaction Query4: Customer booked shows on a given date

Critical Evaluation

1. Further discussion on Learning Experience

While developing a database model for the given scenario, various methods were used so that the database could be properly explained and created. To do so at first all of the required entities and attributes were listed out and a simple database model was created. This however wasn't the most ideal database model as data redundancy was not minimized. So to minimize the data redundancy and to remove anomalies while Updating, deleting or adding any data in the database, Normalization was done. Normalization is simply the process which will help in reducing data redundancy. In Un-normalized form all of the attributes required for the database was listed out and the repeating groups were identified. All of the repeating group was removed to create new tables and referenced the primary key from the main entity. The super type and sub type was also separated in this step. Now to be in the Second normal form, The entities must satisfy all the rules of the first normal form and all the partial functional dependencies was removed. After that the entities were again checked for any transitive functional dependencies and the attributes were separated into another table. Finally all of the entities in the database model was normalized to third normal form.

After normalization, the relationship between the entities were presented in an ER-Diagram and all of the attributes of each entities were described in detail in form of data-dictionaries. Then the data dictionaries for the entities were used to create various tables interlinking them. The tables were then populated with various data and several queries were run on it. Various queries stated by the scenario was tested and each of them gave the desired information as output. All of the steps succeeded which concludes that the implementation of the so created database model was correct and can run in real world environments. So by finalizing the database the dump file and the '.sql' files containing all the necessary commands, queries, and steps were put in the development folder.

2. Critical Assessment of coursework

As database has become more and more useful due to its ability to manage any automatic task of recording various activities and information, It has become very important to understand the mechanics of the databases. As database is very versatile and can be used to store any kind of information it is also quite important to understand the scenario and implement a proper database to store the information provided and to make retrieving data more easy and efficient. In this coursework a database model was designed to store information about every employee, customer, movies and shows and all of the bookings shows how a real life problem of storing complex interconnected information can be easily stored in a database with minimal redundancy and how easily this information can be retrieved when required. All of the mentioned reasons are why databases are swiftly taking over the information handling process and why this module is important to understand the basic methods of designing and understanding the mechanics of a database.

Database Systems module is a very interesting and important module as it deals with designing and implementing databases. This module mostly consists of the theoretical part dealing with the creation of entities and relation of the database in real life scenarios and also provides practical insight on how to solve them. As a right database model can make a huge impact on managing data so designing a database model correctly is crucial for any system. To implement the database in this module, oracle database management system was used which is quite similar to the previous MySQL DBMS. This module teaches various ways to use oracle DBMS such as to create tables, populate the tables, search for specific information stored in the database. This module also deals with most of the datatypes that can be used in oracle DBMS and also various alternative methods for doing tasks so that managing database will be easier in the future. This module and the coursework will greatly increase both the practical and thermotical knowledge on databases so it is quite important for one’s carrier.

References

IBM Corporation, 2018. *Key concepts: Entity, attribute, and entity type.* [Online]
Available at:
https://www.ibm.com/support/knowledgecenter/en/SSWSR9_11.6.0/com.ibm.mdmhs.overview.doc/entityconcepts.html
[Accessed 12 1 2019].

Kreines, D., 2009. *Oracle Data Dictionary Pocket Reference.* 1st ed. Chicago: O'Reilly Media.

Scottish Qualifications Authority, 2010. *Outcome 1: Fundamentals of Database Design.* [Online]
Available at: https://www.sqa.org.uk/e-learning/MDBS01CD/page_06.htm
[Accessed 12 1 2019].

Song, I.-Y., Evans, M. & Park, E. K., 1994. A Comparative Analysis of Entity-Relationship Diagrams. *Journal of Computer and Software Engineering,* 3(4), pp. 427-459.

Watt, A., 2012. *Database Design.* 2nd ed. Vancouver: BCcampus Open Textbook.

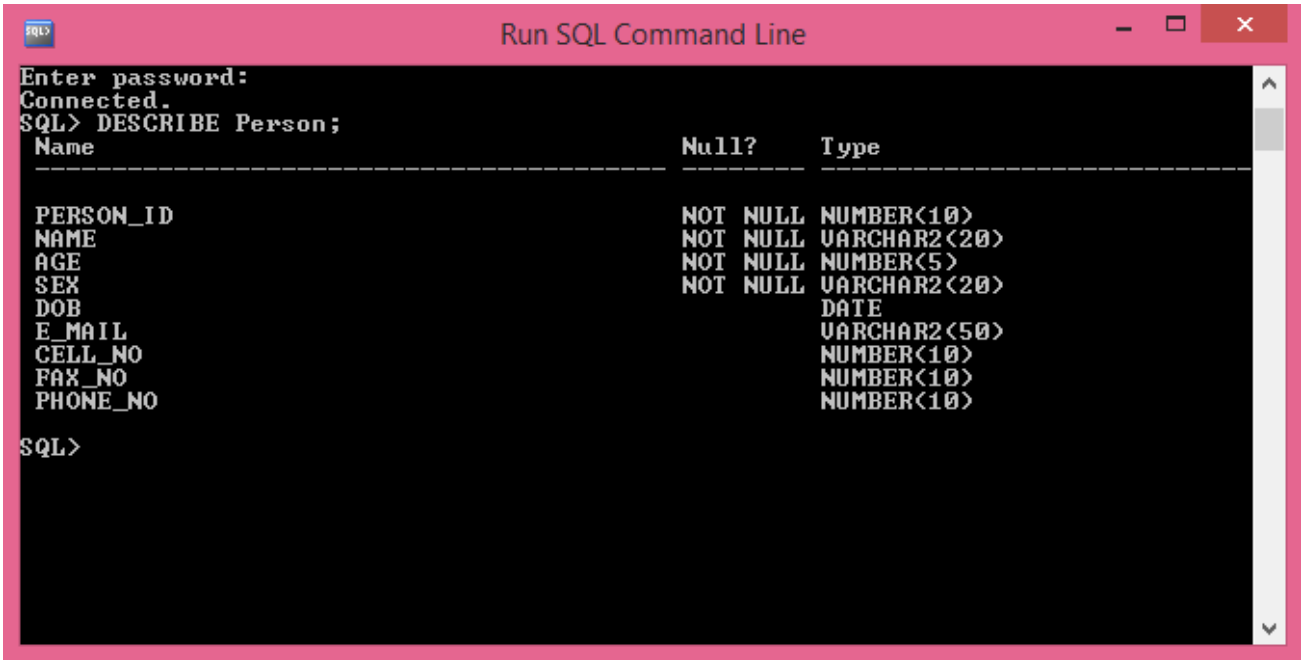
Appendix

User: ShresthaRajat

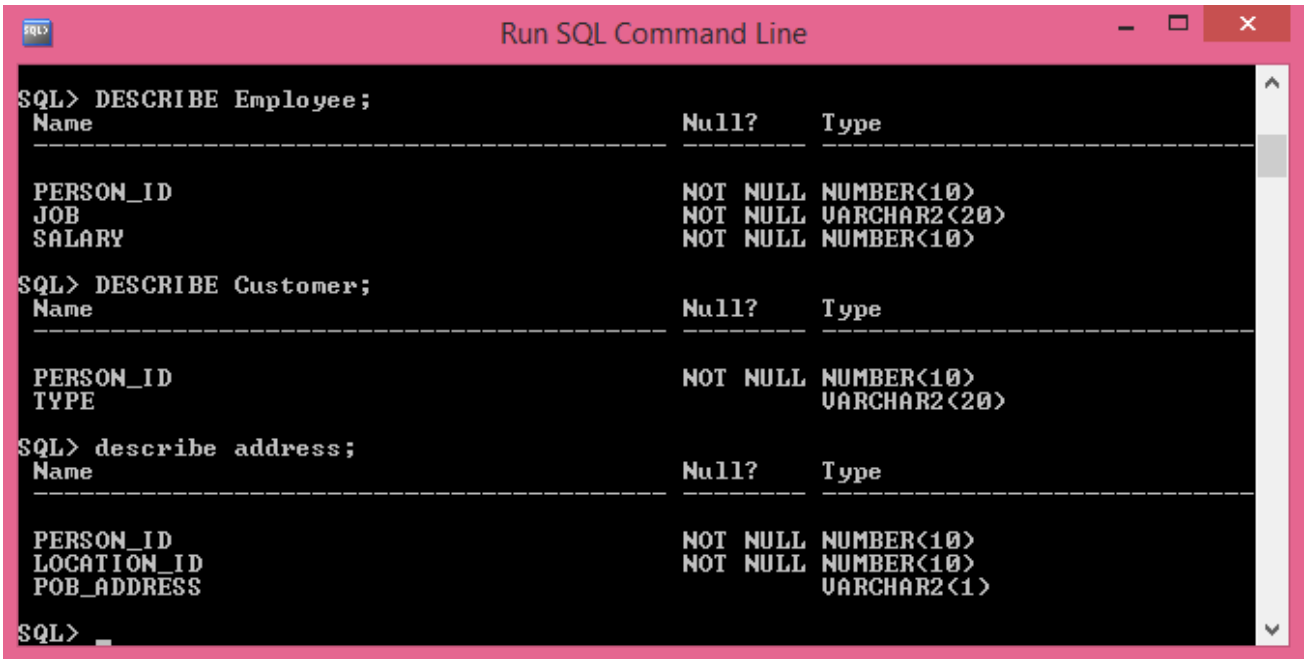
Password: pwrd

Detailed Specification of each table:

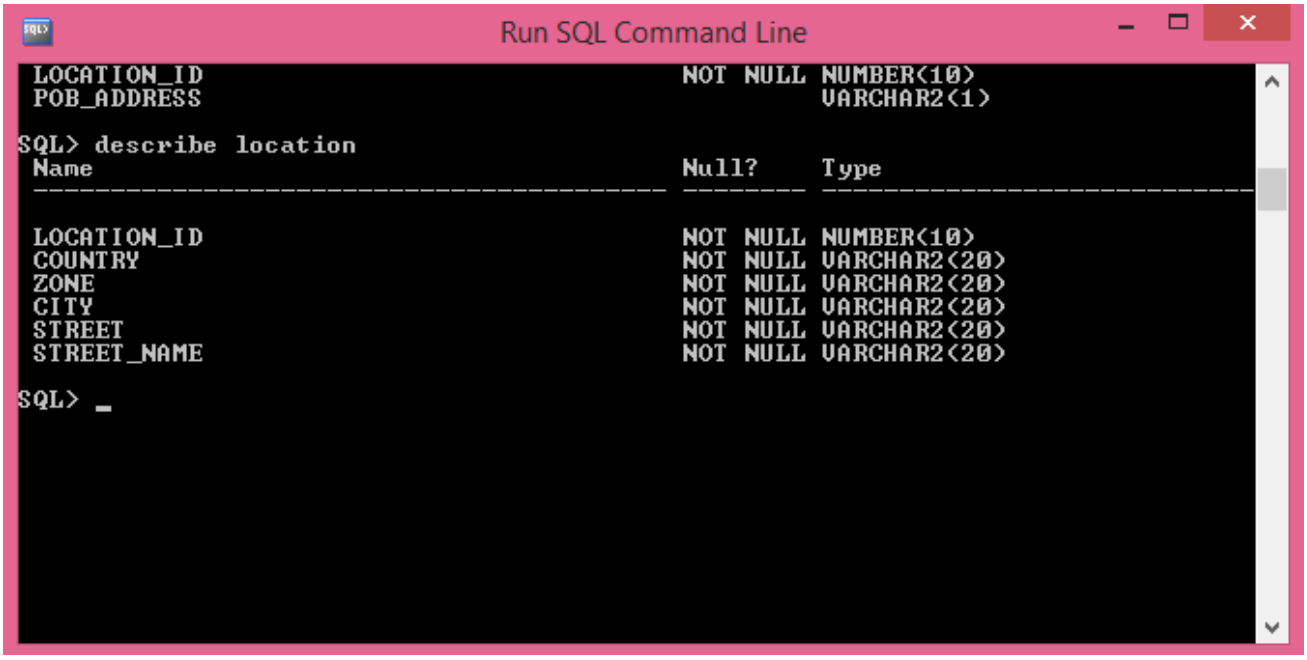
- **Person**



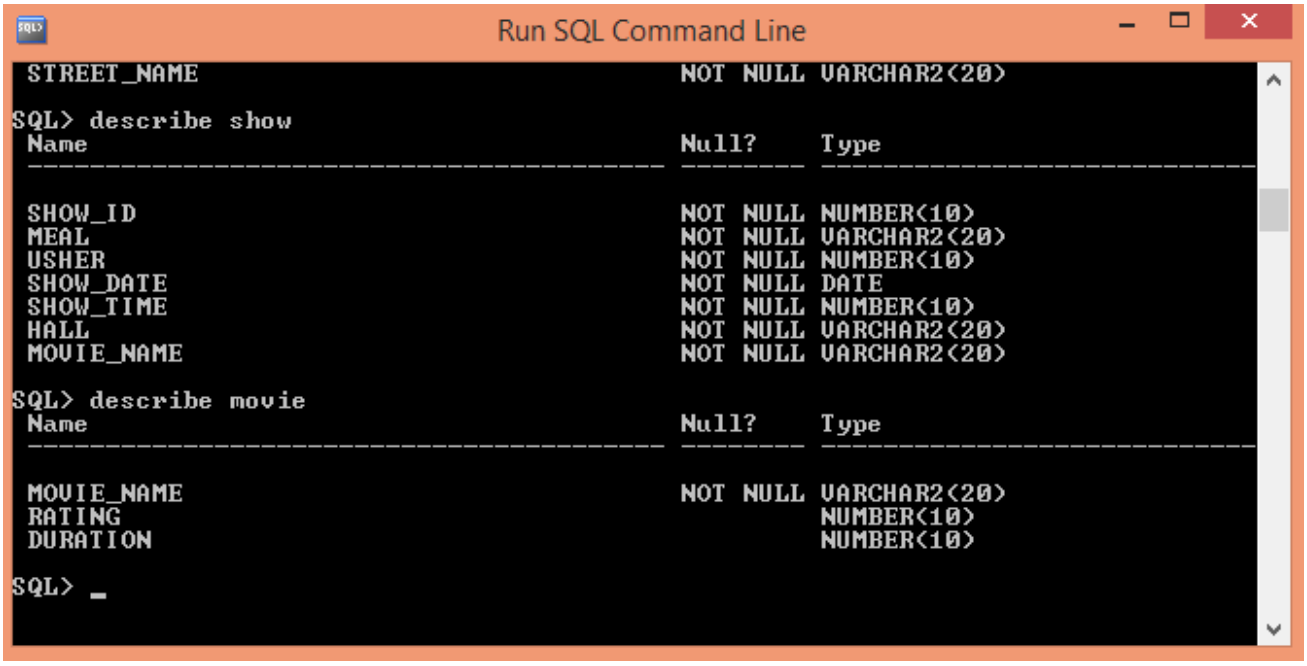
- **Employee, Customer and Address**



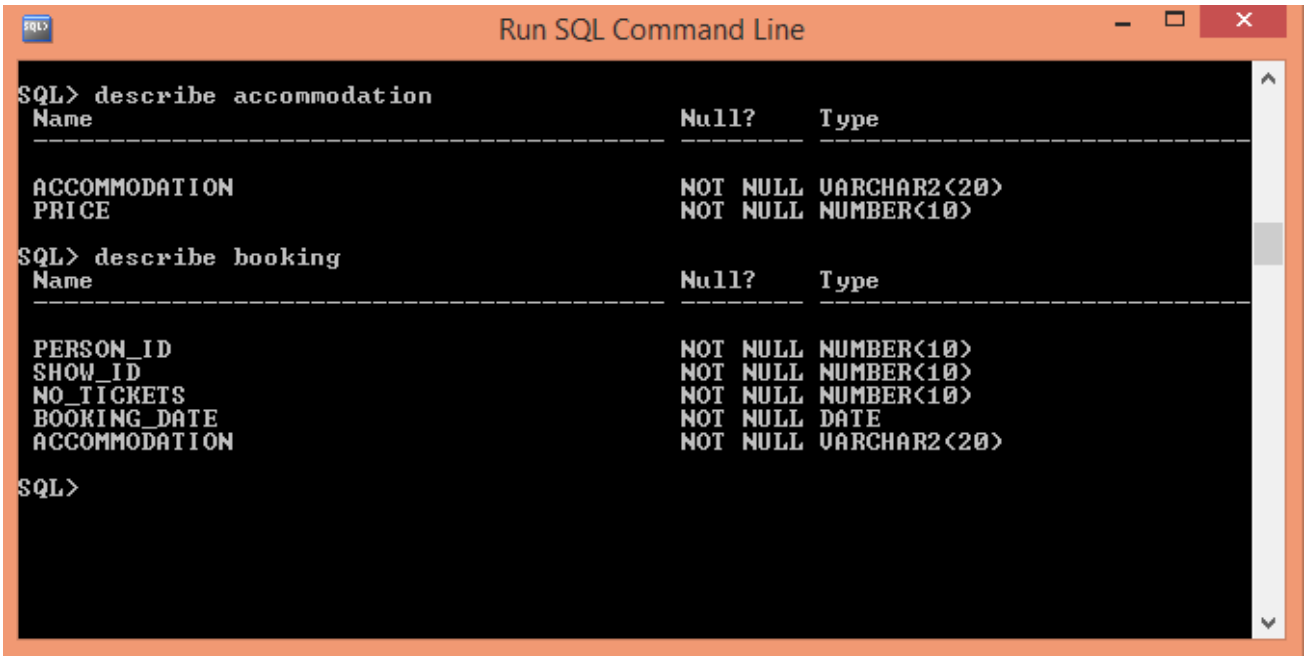
- **Location**



- **Show and movie**



- **Accommodation and Booking**

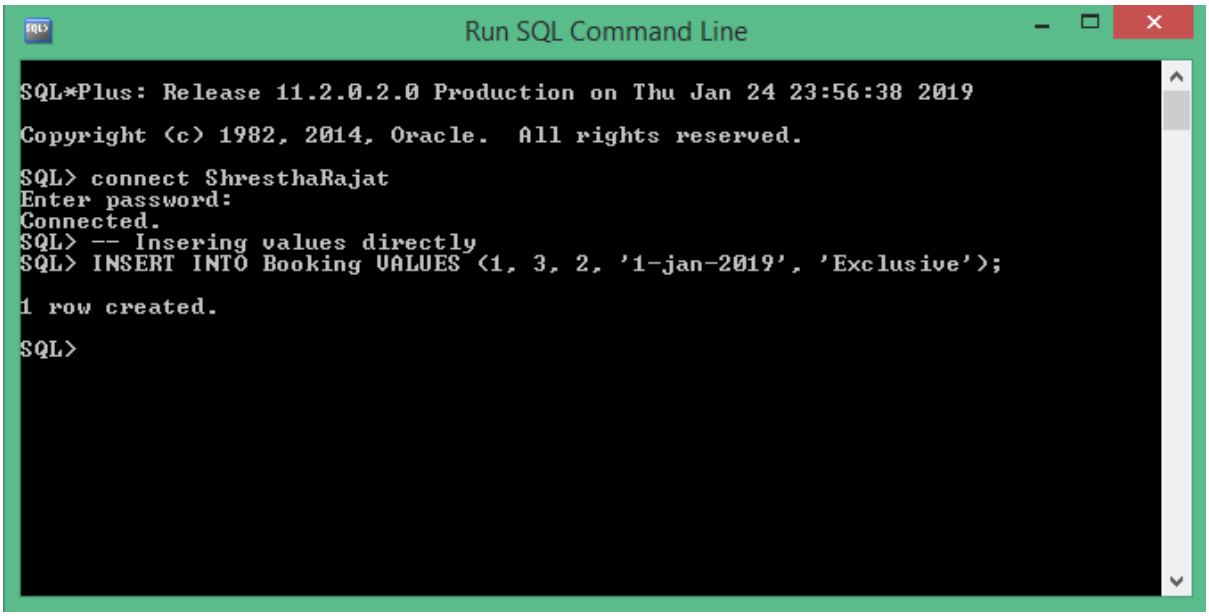


Inserting values by different methods:

While entering the data into the tables there are many methods by which we can insert data which are:

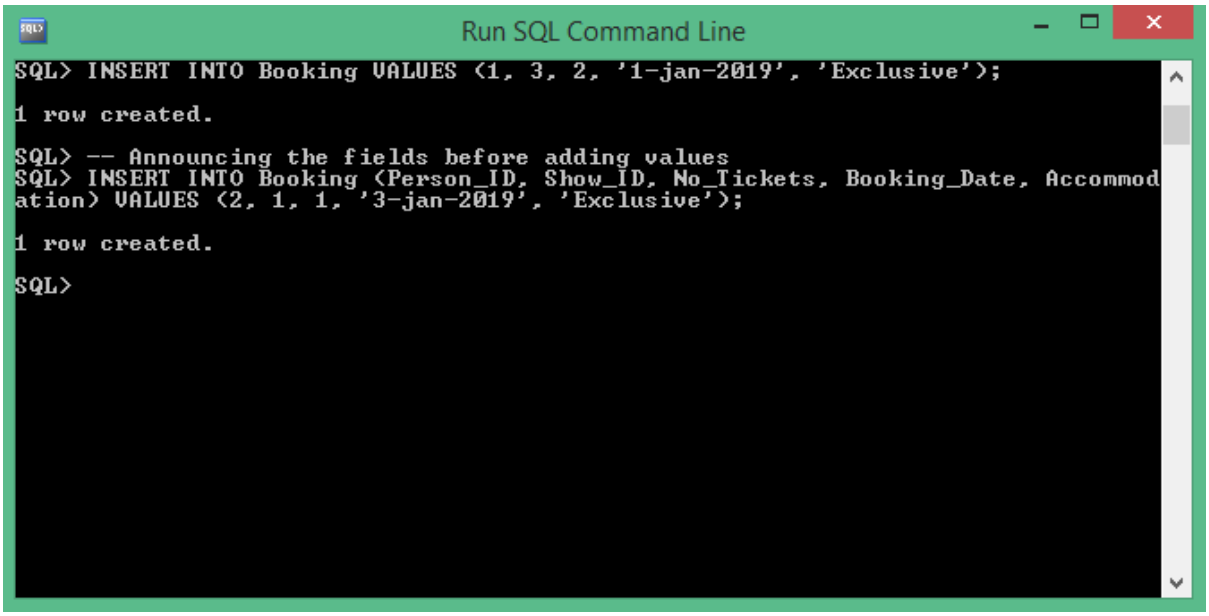
- **Directly Inserting values:**

```
INSERT INTO Booking VALUES (1, 3, 2, '1-jan-2019', 'Exclusive');
```



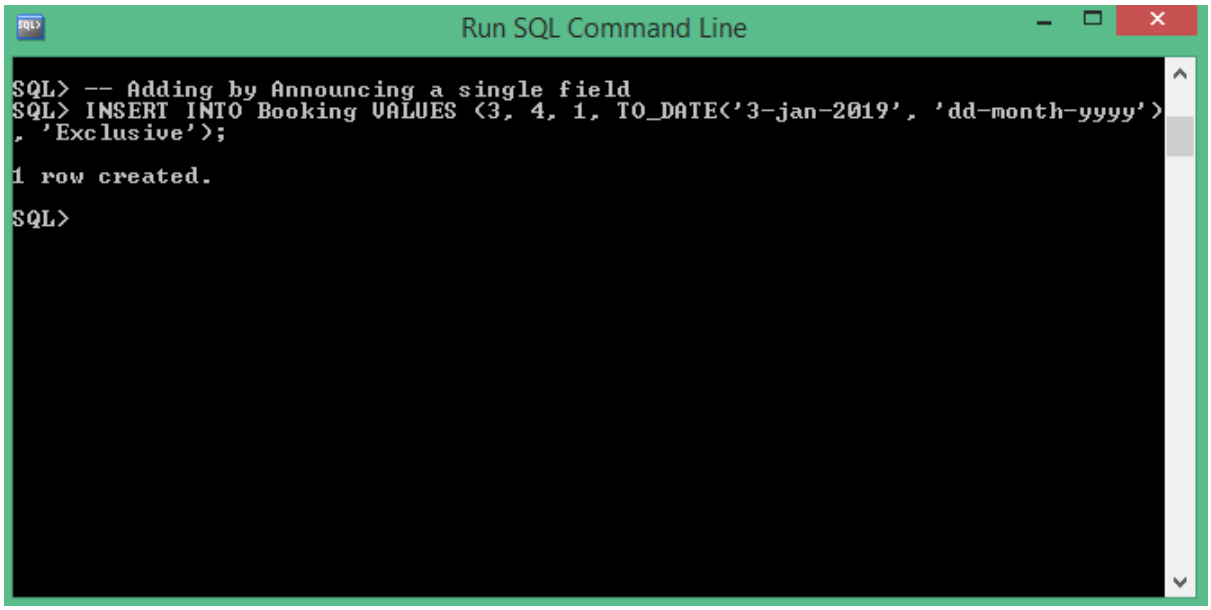
- **Announcing the fields before adding values:**

```
INSERT INTO Booking (Person_ID, Show_ID, No_Tickets, Booking_Date,
Accommodation) VALUES (2, 1, 1, '3-jan-2019', 'Exclusive');
```



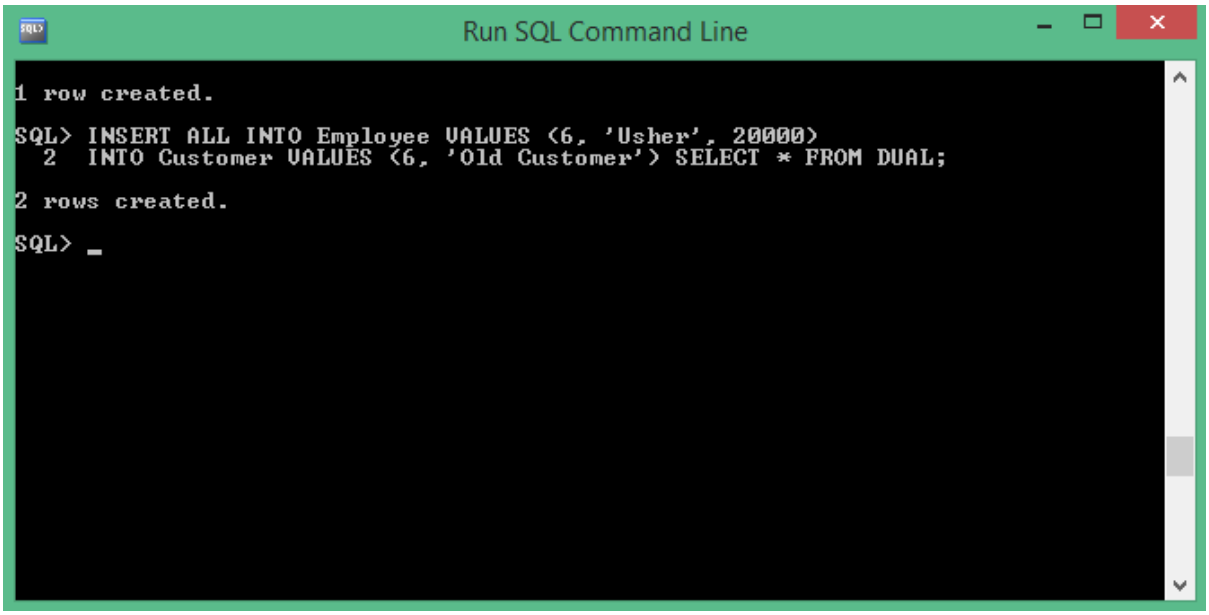
- **Adding by Announcing a single field**

```
INSERT INTO Booking VALUES (3, 4, 1, TO_DATE('3-jan-2019', 'dd-month-
yyyy'), 'Exclusive');
```



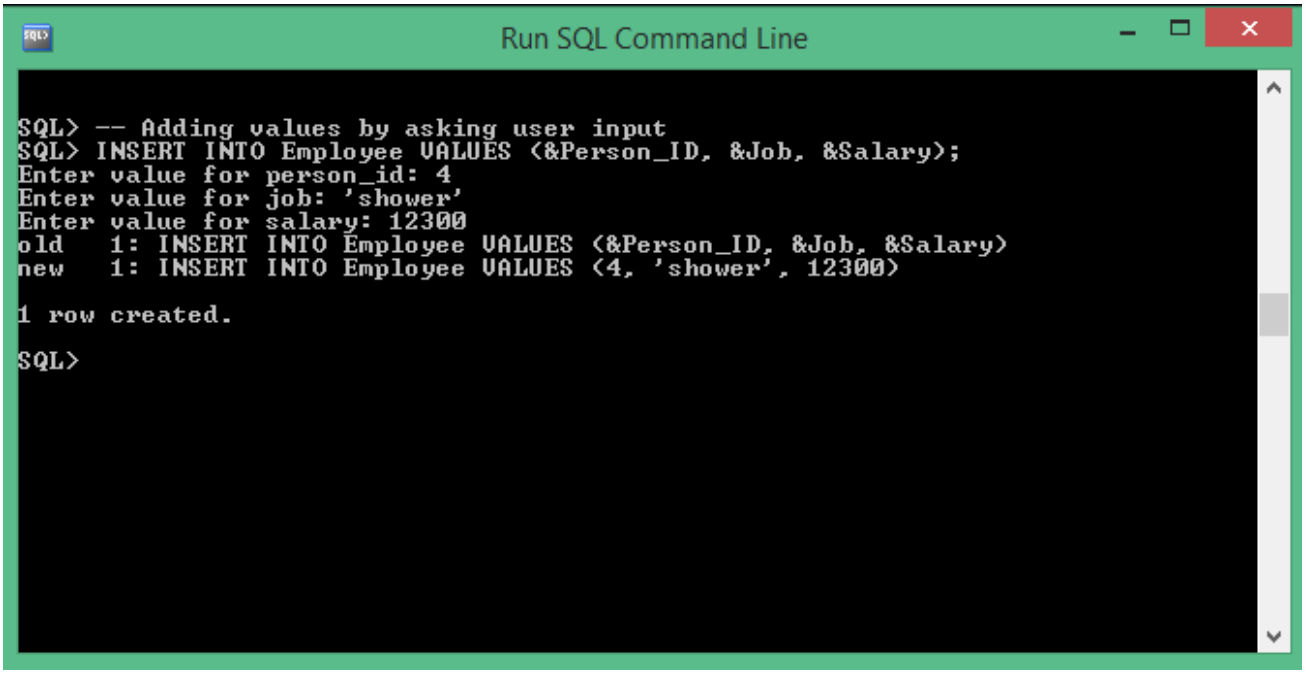
- **Adding into two separate tables directly**

```
INSERT ALL INTO Employee VALUES (6, 'Usher', 20000)
INTO Customer VALUES (6, 'Old Customer') SELECT * FROM DUAL;
```



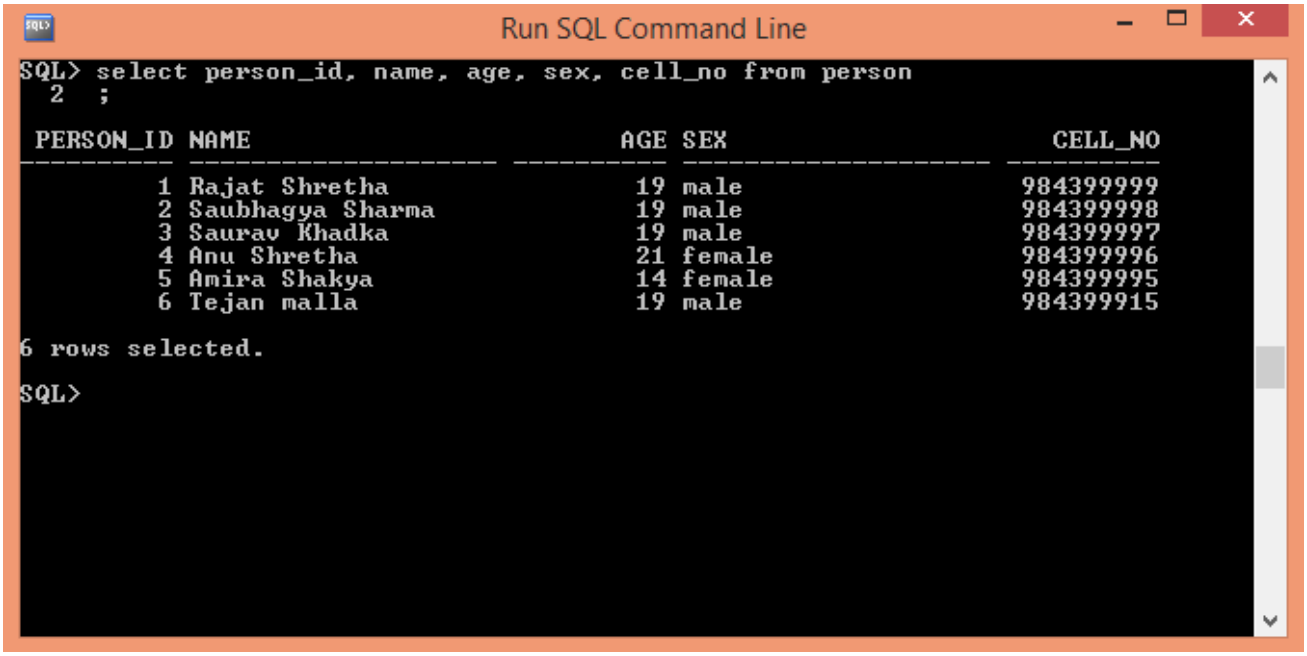
- **Adding values by asking user input:**

```
INSERT INTO Employee VALUES (&Person_ID, &Job, &Salary);
```

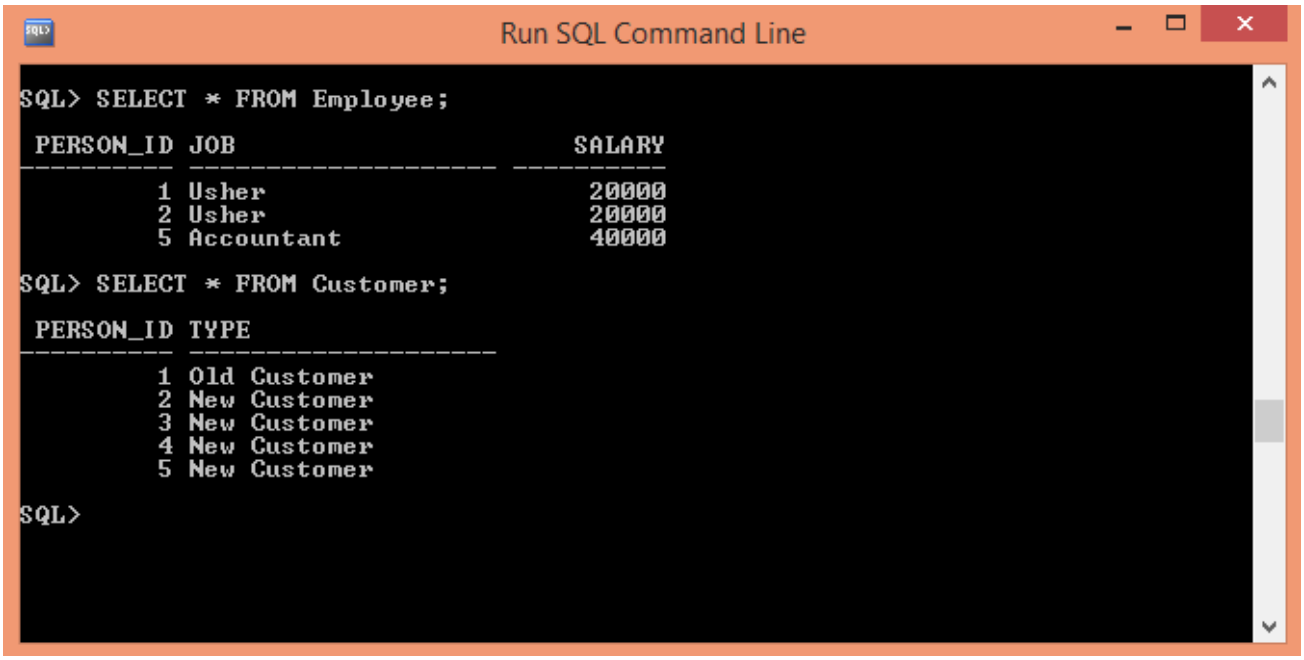


Contents of the tables after adding values:

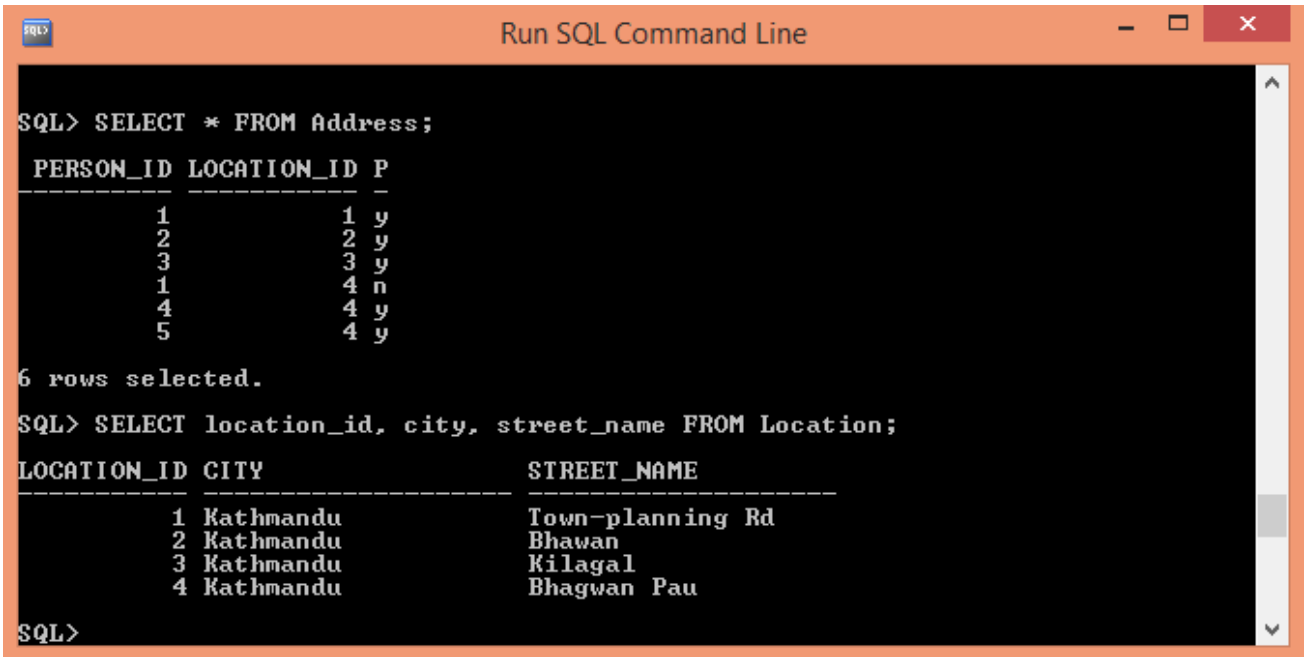
- **Person**



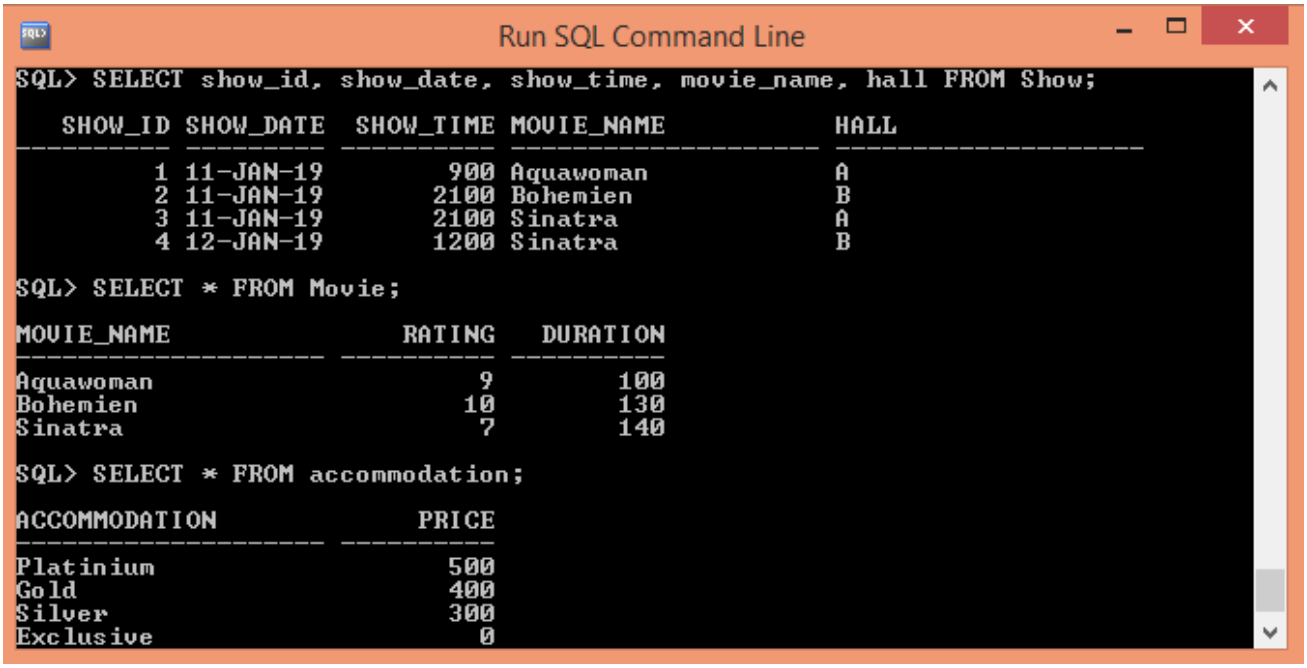
- **Employee and Customer**



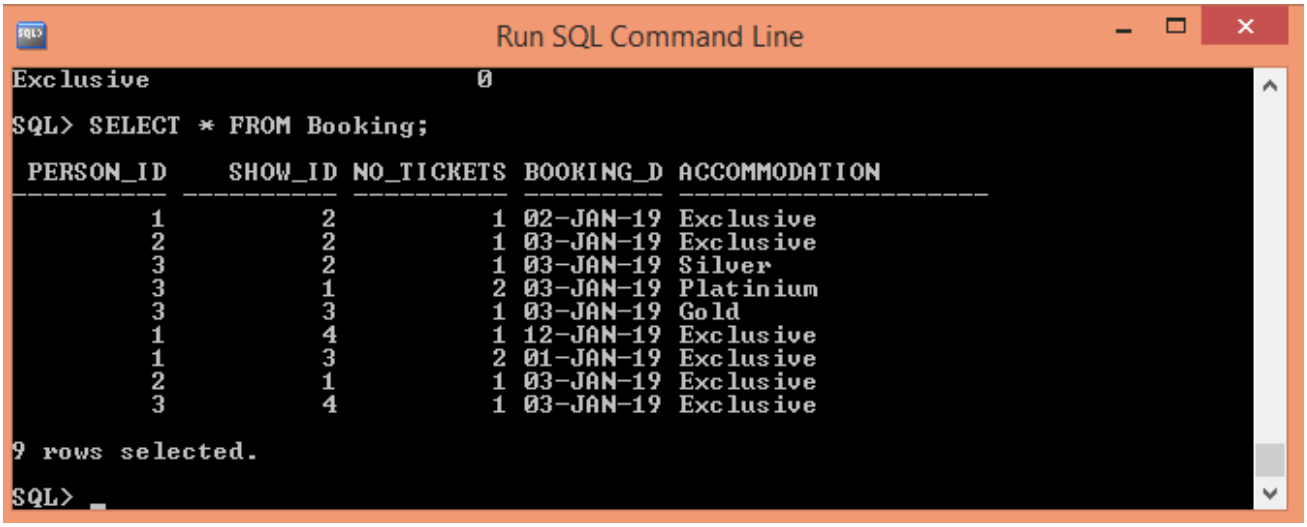
- **Address and Location**



- **Show, Movie and Accommodation**



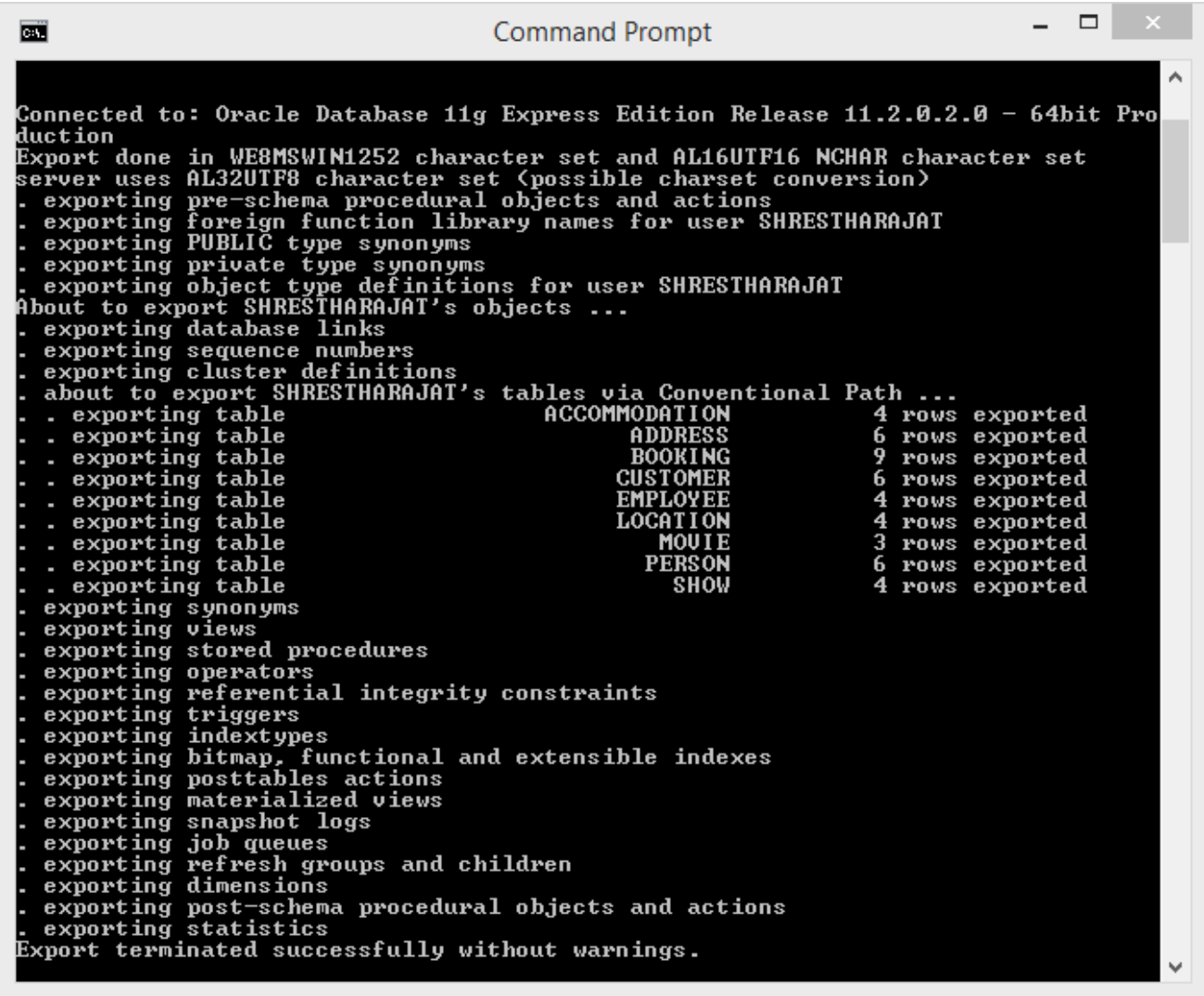
- **Booking**


Run SQL Command Line
Exclusive 0
SQL> SELECT * FROM Booking;
PERSON_ID SHOW_ID NO_TICKETS BOOKING_D ACCOMMODATION
1 2 1 02-JAN-19 Exclusive
2 2 1 03-JAN-19 Exclusive
3 2 1 03-JAN-19 Silver
3 1 2 03-JAN-19 Platinium
3 3 1 03-JAN-19 Gold
1 4 1 12-JAN-19 Exclusive
1 3 2 01-JAN-19 Exclusive
2 1 1 03-JAN-19 Exclusive
3 4 1 03-JAN-19 Exclusive
9 rows selected.
SQL>

Exporting and importing the dump files

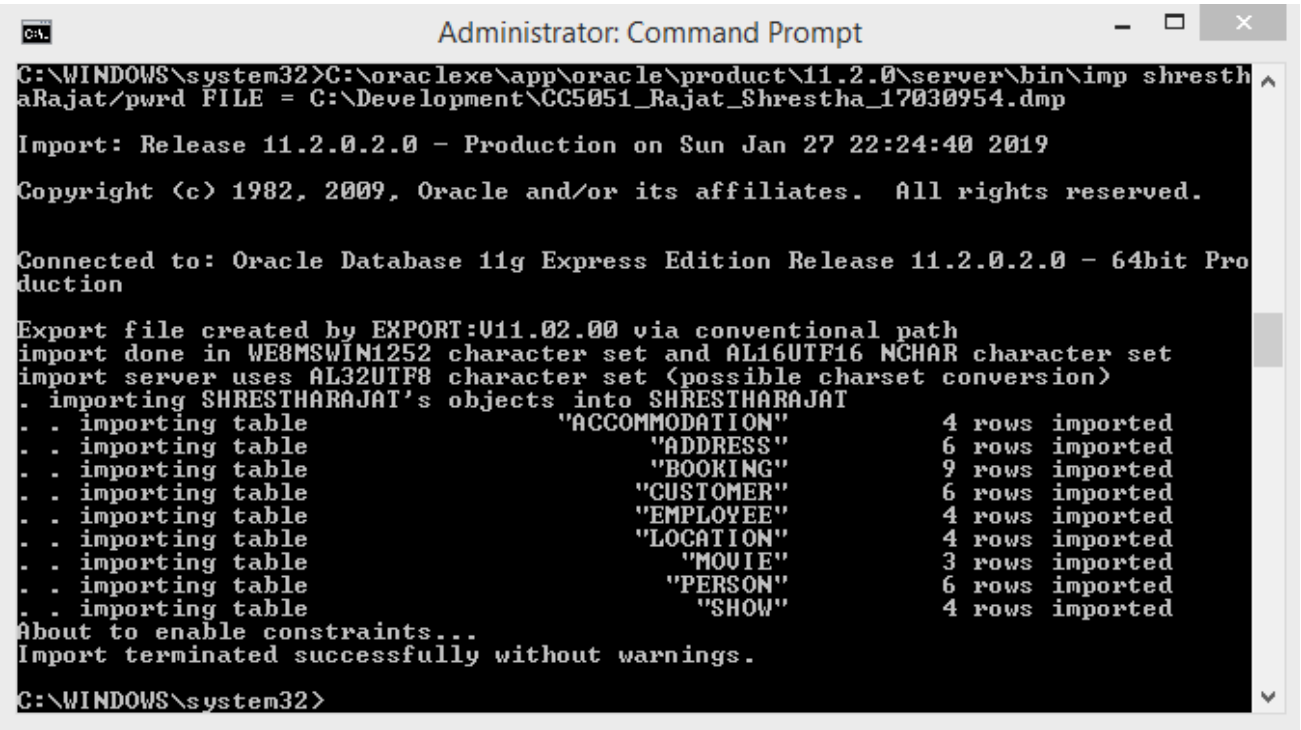
To export:

```
C:\oraclexe\app\oracle\product\11.2.0\server\bin\exp shresthaRajat/pwrd
FILE = C:\Development\CC5051_Rajat_Shrestha_17030954.dmp LOG =
C:\Development\CC5051_Rajat_Shrestha_17030954.log
```



To import:

```
C:\oraclexe\app\oracle\product\11.2.0\server\bin\imp shresthaRajat/pwrd
FILE = C:\Development\CC5051_Rajat_Shrestha_17030954.dmp
```



Development Folder with sql codes, dump files and miscellaneous text files: