**Bon Voyage Database Management System**

**Summary**

* Bon Voyage is a relational database with 17 entities, 2 multi-table views, 2 computed columns and a data encryption trigger
* Bon voyage is a powerful reservation database designed for tour & activity of all sizes, built to bring ease and efficiency to every aspect of your tour

**Overview**

A hotel's reservation system is the heart of the hotel itself. Think about it: if you can't get rooms booked, what's the function of a hotel? For many hotels, a reservations system does more than just book rooms. It also organizes reservations, saves details about guest preferences, and can even keep contact with some special event and restaurant to yield maximum returns. Many reservation solutions can also help you craft hotel deal packages for busy seasons or certain holidays, while still enabling you to come out on the end with a remarkable price.

**Doing this by hand would take a ton of time.**

Again, hotel reservations database or booking software can consolidate all of these booking across a multitude of platforms so that they are organized and available right in front of you in one user-friendly interface. No overbooking or mistakes made on your end. You can even know about guest preference before they come as well. Not to mention, it can also let prospective guests book online at all hours, meaning front desk employees can spend more time assisting checkins and focus on keeping the current guests happy. Now imagine if this could be done for multiple hotels whereby they act as franchises and you get a lot more options to choose from.

*Bon Voyage* does all this and more. We let you connect with private event organizers and you can get access to a multitude of events across your city of choice. Each restaurant, room, event organizer is rated by you and this helps all prospective guests to know the pros and cons of each service. We even provide a list of some of the local tourist attractions that may interest you during your visit to the city.

**Mission Statement:**

A project to create a database to view and manage bookings by various customers. It searches through all the hotels or properties and events in the locality that are registered in the system and returns the deals as per the customer requirements and date availability.

**Objective:**

To make traveling an even more of a delightful experience, we will also store the details or profiles of the customers and once a booking is made the customer will get an option to view/match with other travelers staying in the same area or hotel during the same duration of stay. We will have the details of various hotels, restaurants, pubs in and around the city. We will also have details, like things to do around the city and store the information of event organizers to help customers to explore the city.

We created the following tables:

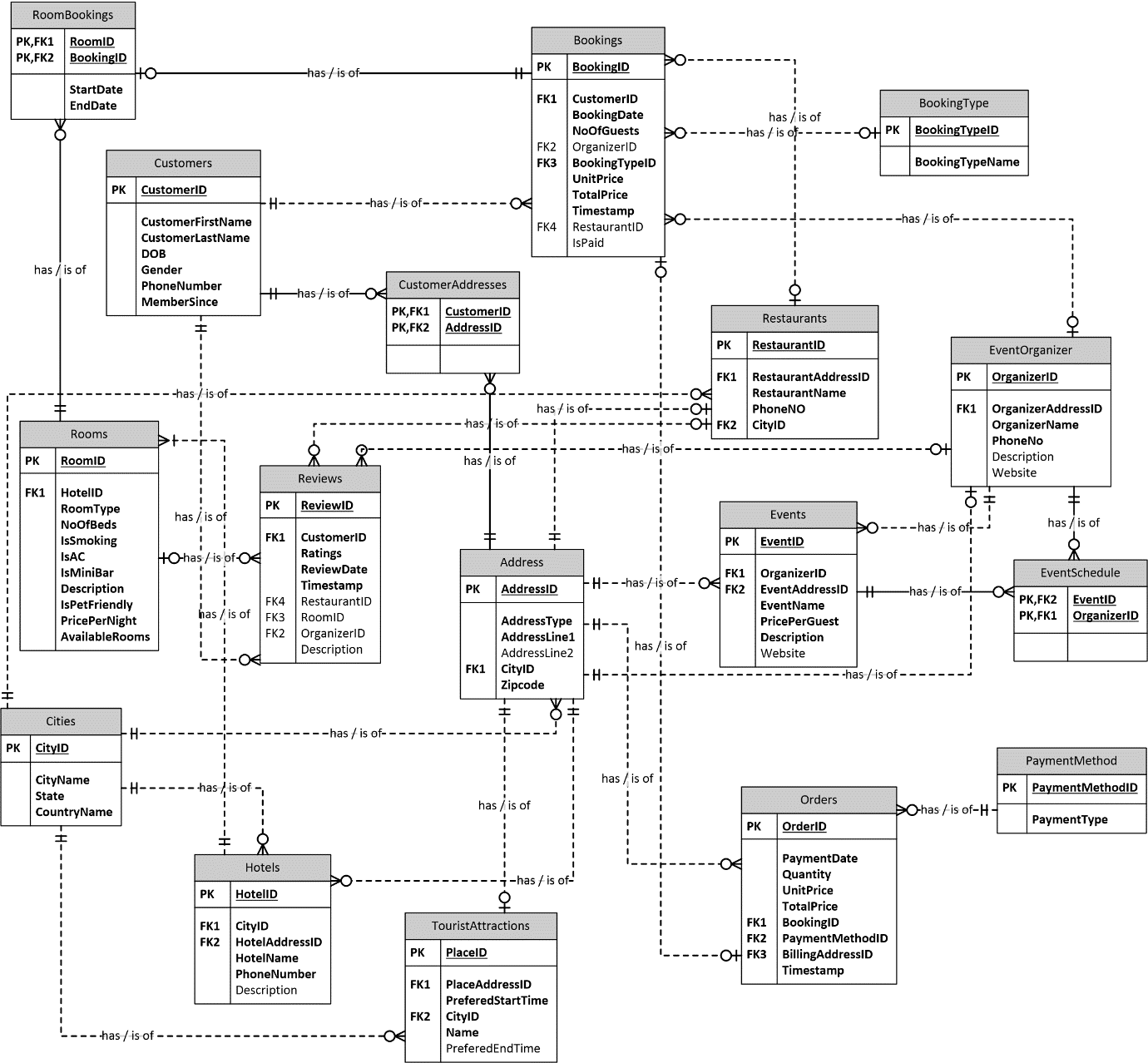
* Hotels
* Rooms
* Bookings
* Room Bookings
* Customers
* Tourist Attractions
* Reviews
* Orders
* Restaurants
* Payment Methods
* Cities
* Event Organizers
* Address
* Booking Type

*Included with xls document (in the root), here are explainations about important tables. The project contains entertainment items as well such as restaurants and tourist attractions, hosts called Event Organizers would manage special events.*

**Key Business Rules:**

1. **Bookings** are converted to **Order** and so, *Order* contains *BookingID* as FK only ***AFTER*** the customer has paid for the booking.
2. **Order** can also have some orders like ticket to some event etc. In this case, it will have the *reference (OrganizerId or HotelID)* which tells us who provided the service.
3. Any person who uses our services has to be a registered customer ie no guests.
4. **Address** has columns namely *CustomerID, HotelID or RestaurantID* corresponding to whose address it is.

**ERD Diagram:**

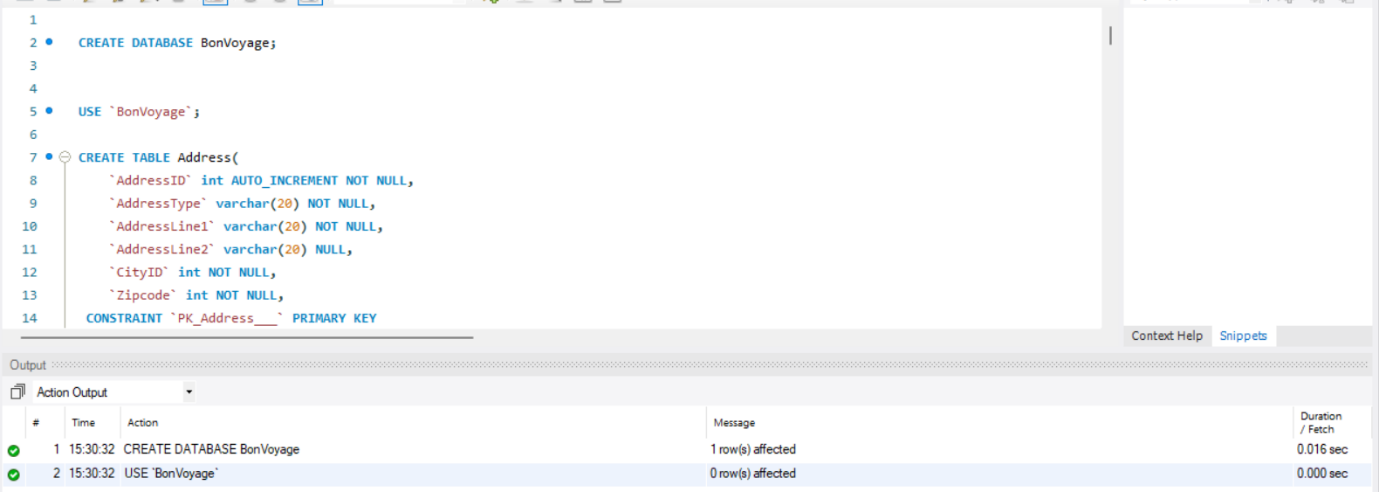


**Relational Database Design Process**

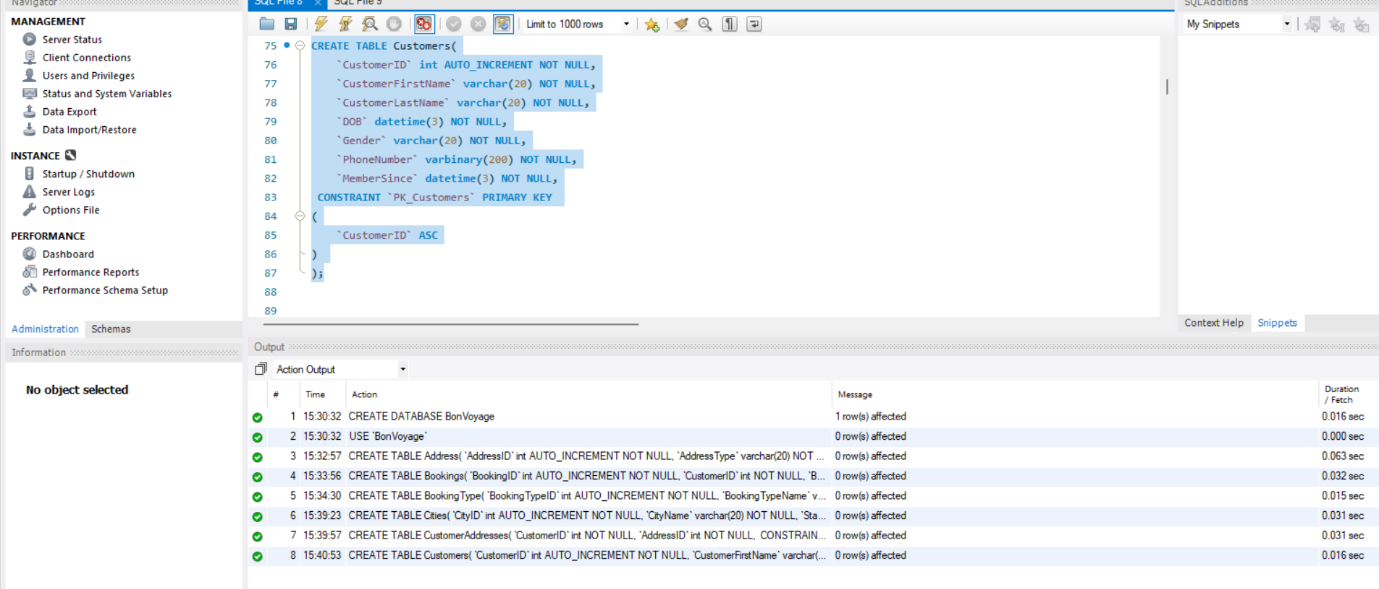
| **Entity Name** | **Why Entity is Included** | **How Entity is Related to Other Entities** |
| --- | --- | --- |
| Cities | Since there would many different hotels with various locations in our system, we will set up a city table to represent the geographic information. | The primary key(PK) *CityID* relates to *Hotels*, *Address*, *Restaurants* and *TouristAttractions* entities to tell us which city it belongs to. |
| Hotels | Each Hotel will be assigned a unique ID, and some detailed information such as Name and Contact Number and other basic descriptions. | Hotels’ PK *HotelID* is a foreign key(FK) of *Rooms* which tells us which hotel the room belongs to. |
| Rooms | This entity will have all the information about the room such as ac/non-ac, room type, no. of beds, pricing, etc. | PK *RoomID* is used as a FK in *RoomBookings*, *Bookings*, and *Reviews* to tell us if a room is booked or available, reviews and ratings about the room. |
| Customers | This stores the details of all the customers who uses the services. | PK *CustomerID* is used as a FK in *Bookings* and *Reviews*. From *Bookings* we can extract the information of the customer who has made a booking. Customer can submit a review in the *Reviews* table to help other travelers. |
| Bookings | Bookings gives the information about the booking such as who has made the booking, when the booking was made, what was booked, no. of visitors, etc. | The PK *BookingID* is a FK in *Orders* to tell us if a booking was completed and if the payment is made. |
| TouristAttractions | This will have the details about places to visit, best time to visit and other relevant information |  |
| Restaurants | This will have details about the restaurant such as address, phone no., city located in etc. | PK *RestaurantID* is used as FK in *Reviews* to allow users to review or rate a restaurant. |
| Reviews | Ratings and reviews always help fellow travelers. Reviews will contain details such as ratings and comments about all the hotels, restaurants and organizers. |  |
| EventOrganizers | It will have all the details of the organizers, events, prices etc. to make the travel easy and enjoyable. | PK *OrganizerID* is a FK in *Bookings* and *Reviews*. Customers can book an event with an organizer and rate and review them as well, according to their experiences. |
| Address | It is used to store all the addresses of customers, hotels, restaurants, and organizers. | PK *AddressID* is used as FK in *Customers*, *Restaurants*, *Orders*, *Organizers*, *TouristAttractions* and *Hotels* which stores all the contact information. |
| PaymentMethods | It tells us the type of the payment when a booking is made by a customer. | PK *PaymentMethodID* is a FK in *Orders* to track a payment if the need be. |
| Orders | Orders will have all the details when a booking is confirmed and a payment is made. |  |
| BookingType | This tells us what is the type of booking such as a restaurant booking, hotel booking or an event booking. | *BookingTypeID* is a FK in *Bookings* which stores all the information about a booking. |
| RoomBookings | This a look-up table that keeps track of the room bookings between a date interval. |  |

**Data Definition Language Scripts**

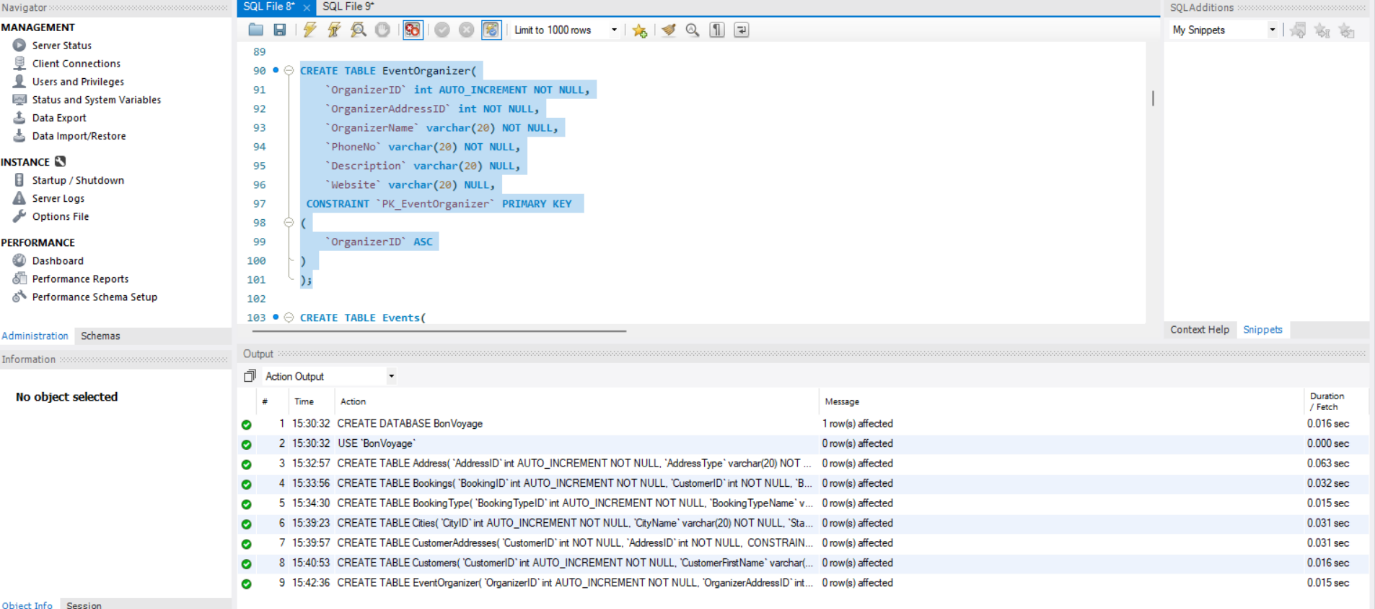
**Database creation:**

****

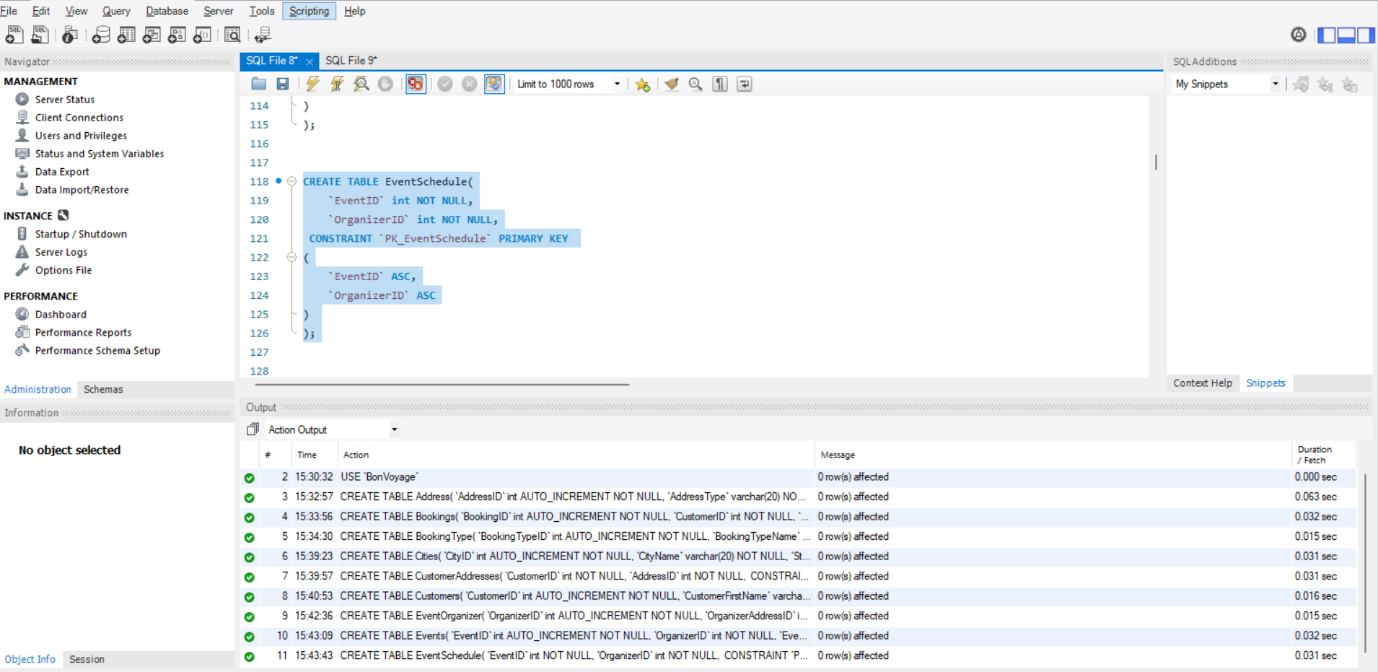
**Customers Table:**



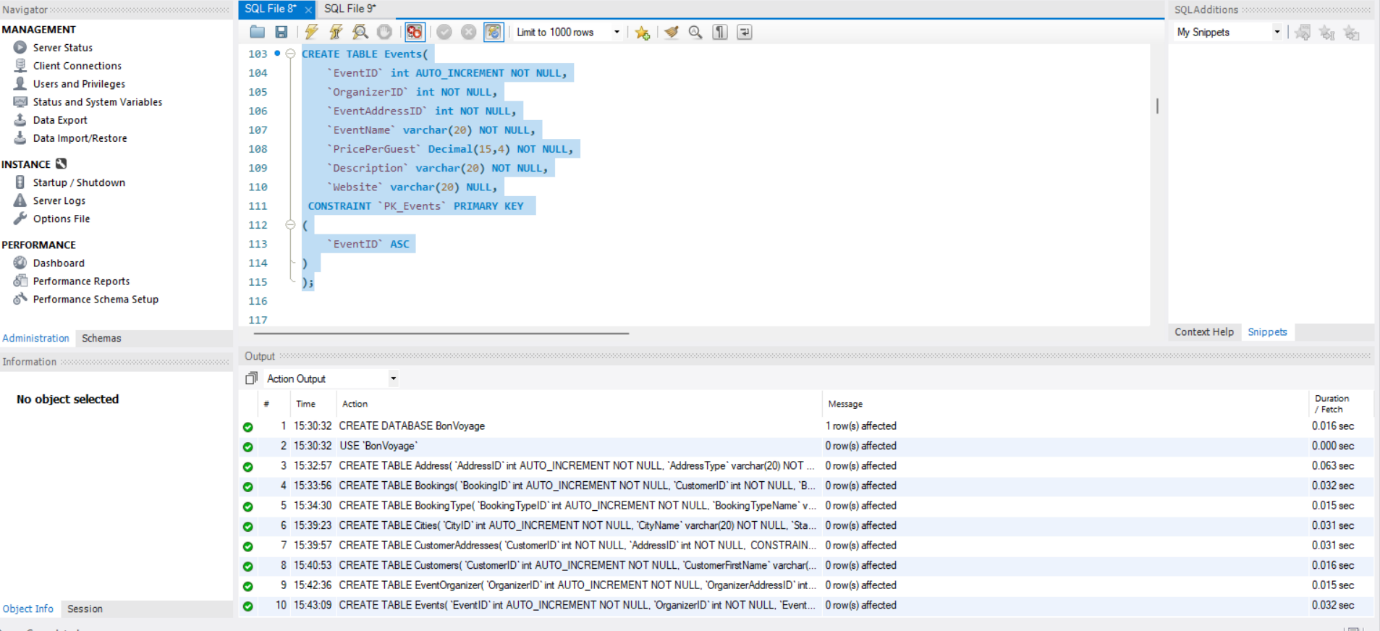
**EventOrganizer Table:**



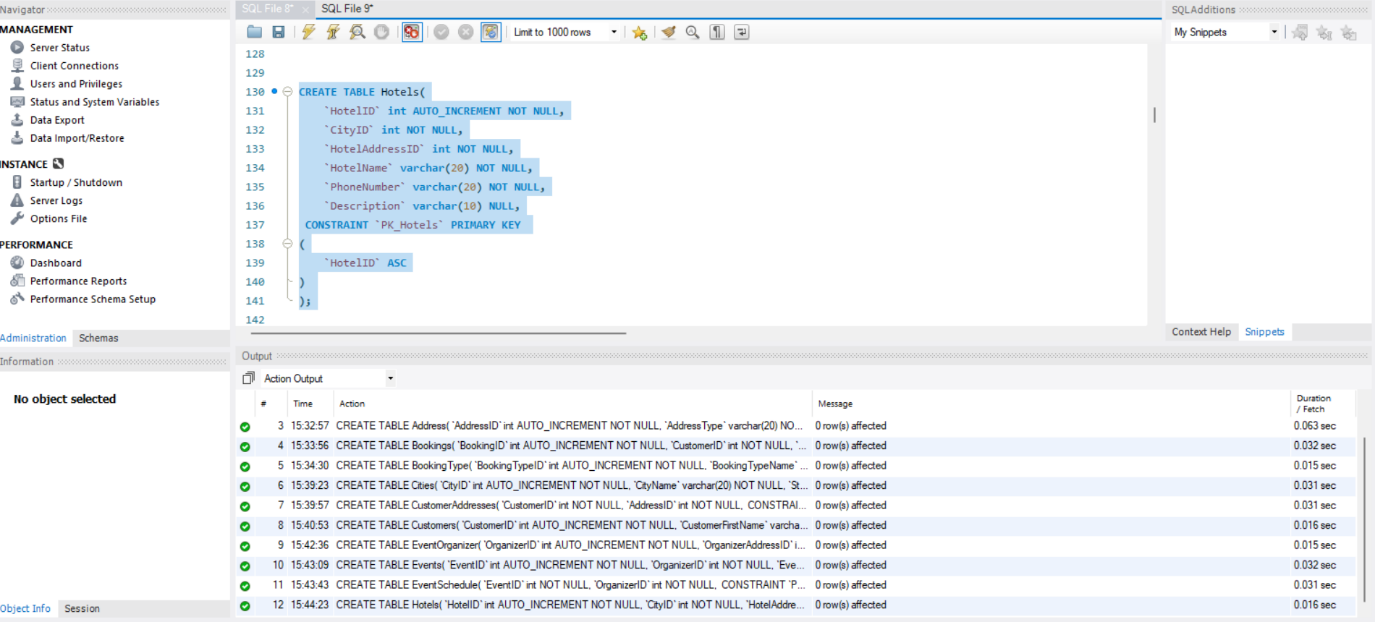
**Event Organizer Table:**



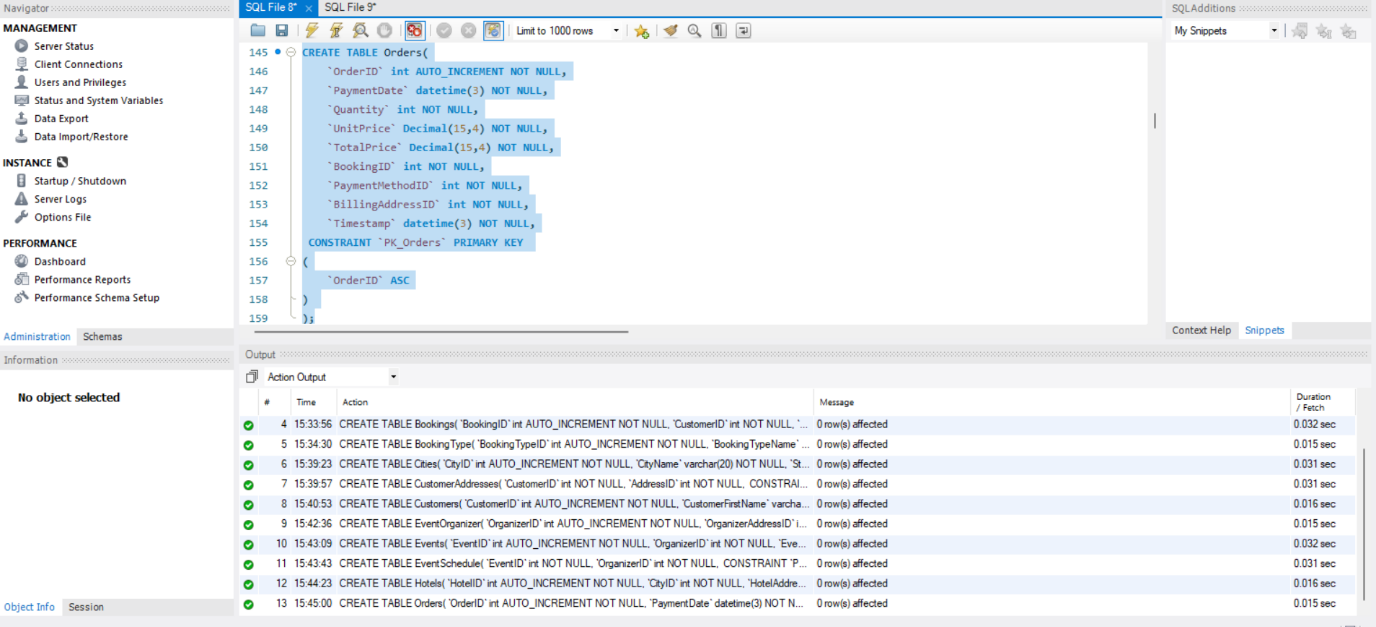
**Event Table:**



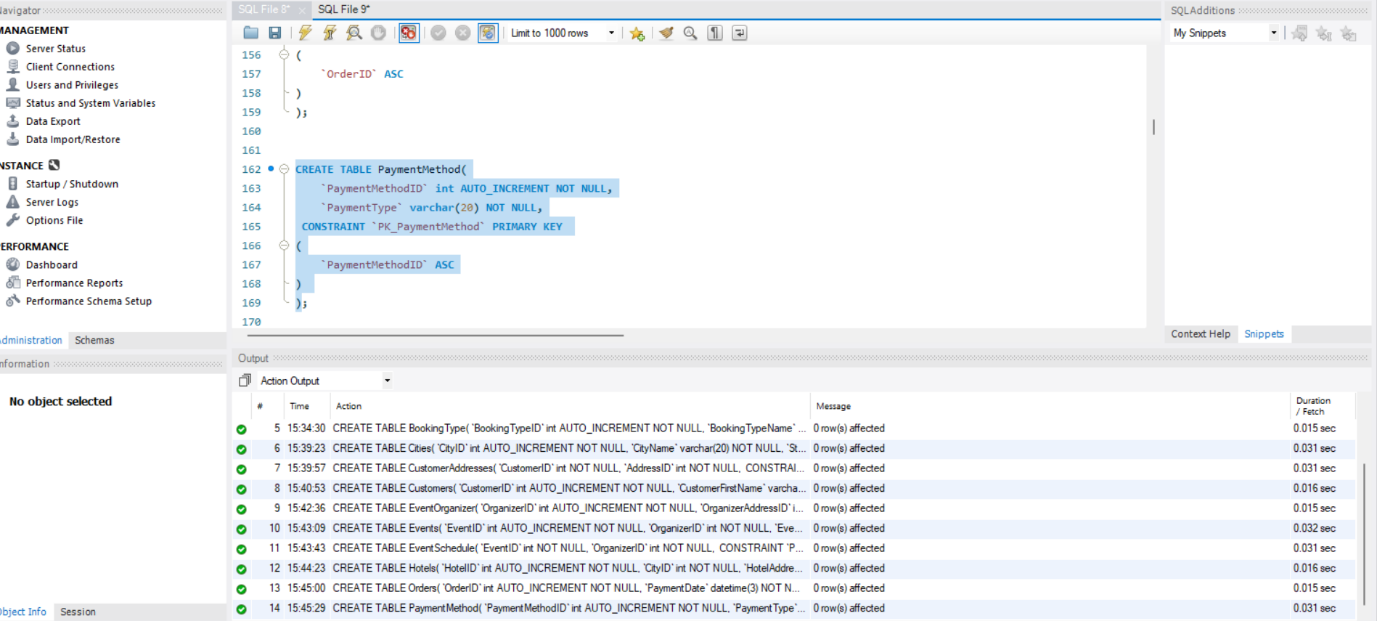
**Hotels Table:**



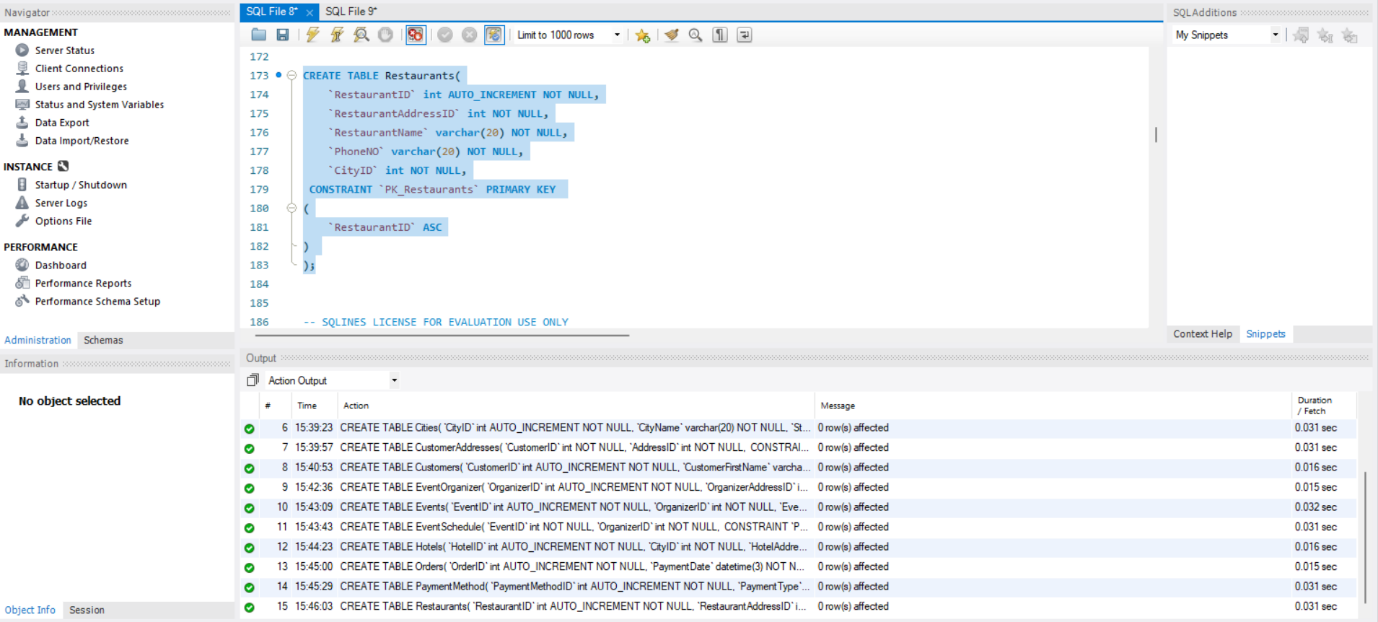
**Orders Table:**



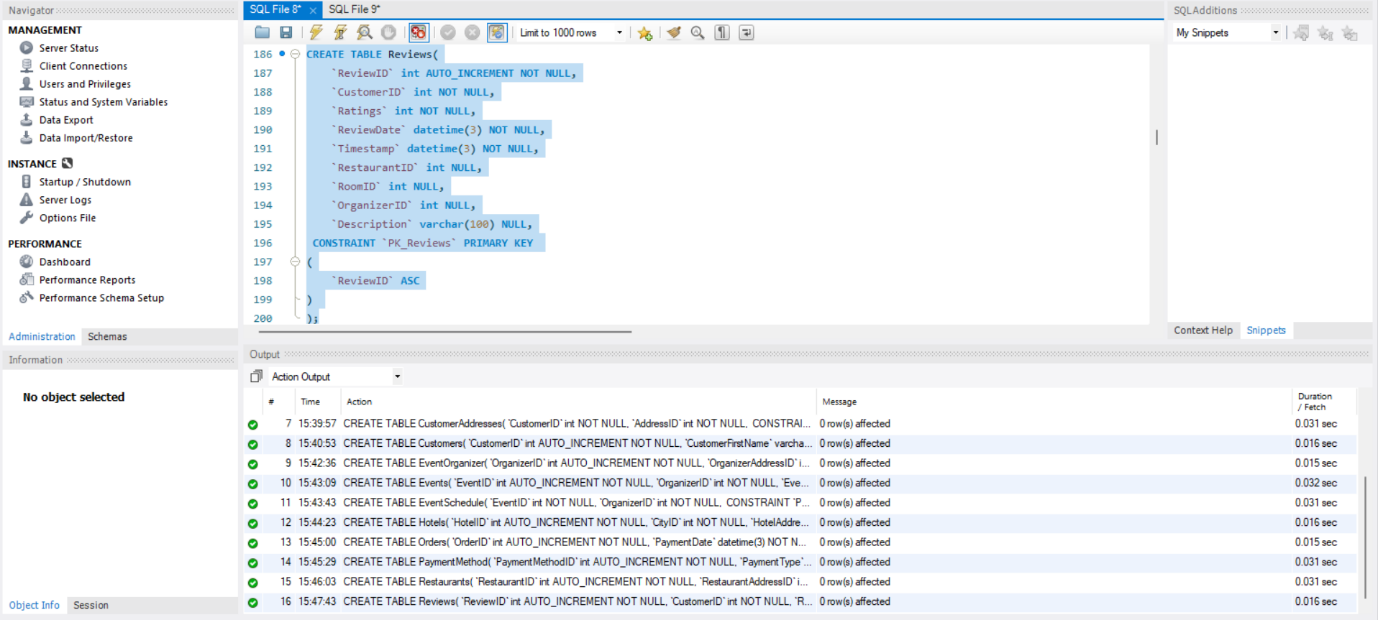
**PaymentMethod Table:**



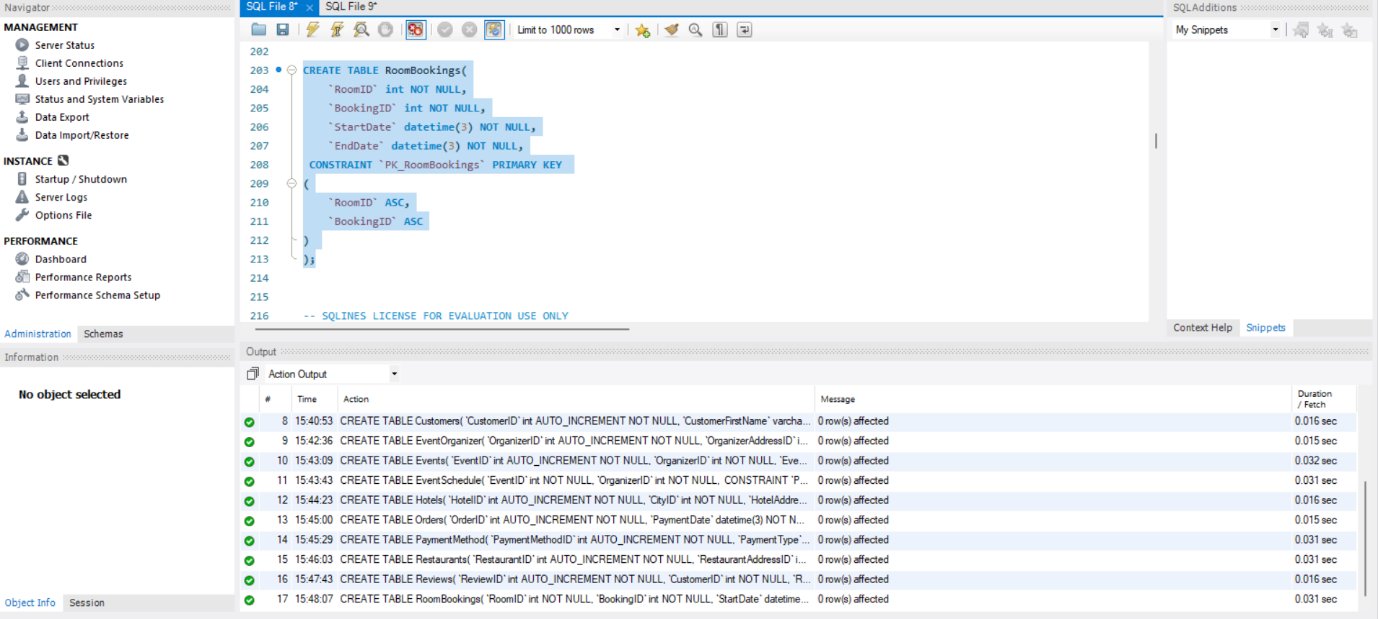
**Restaurants Table:**



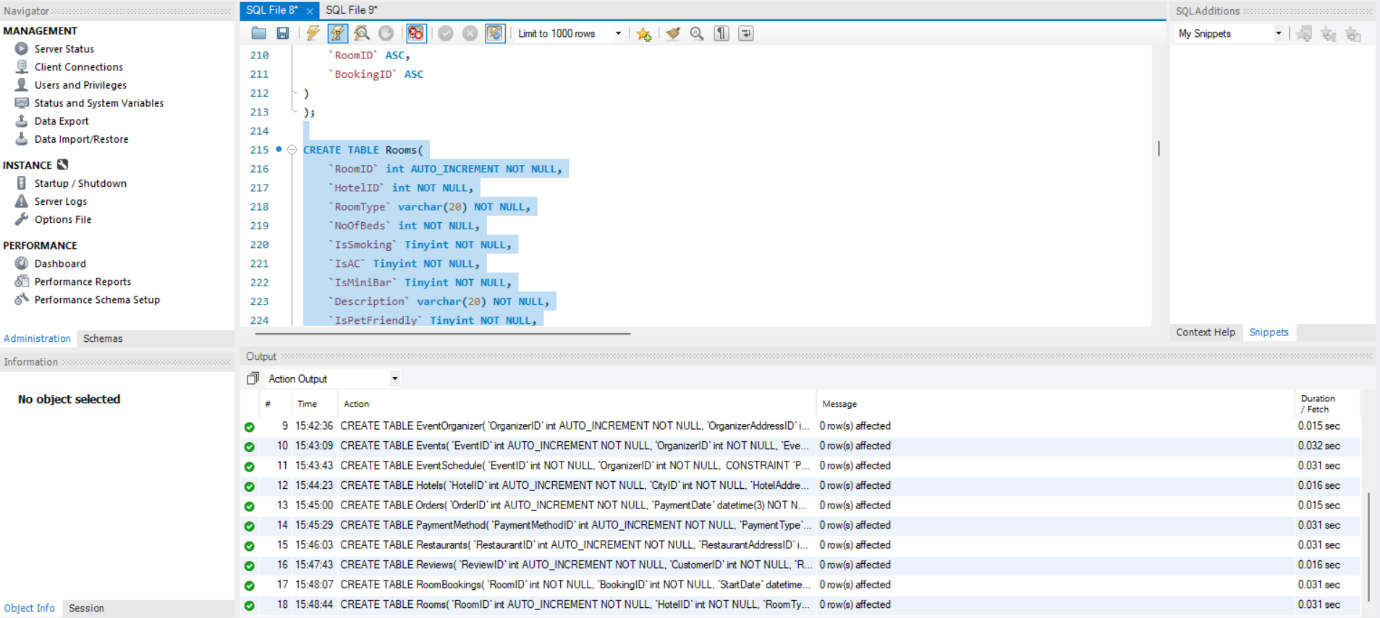
**Reviews Table:**



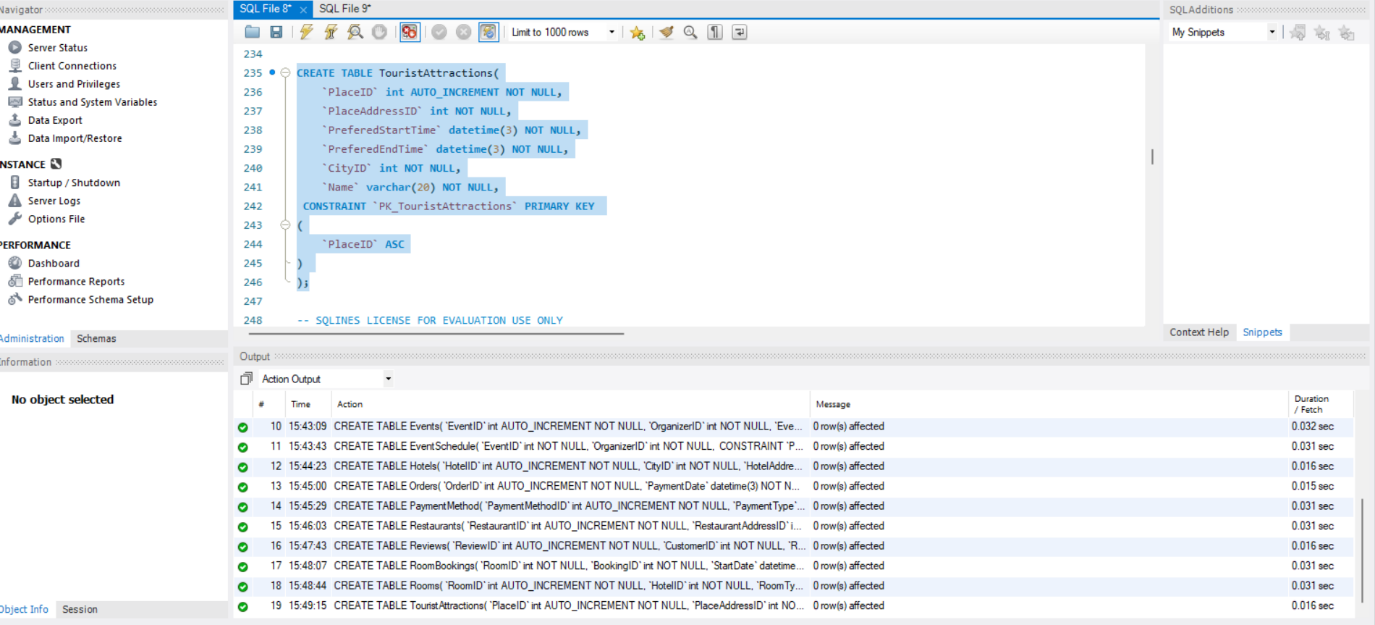
**Bookings Table:**



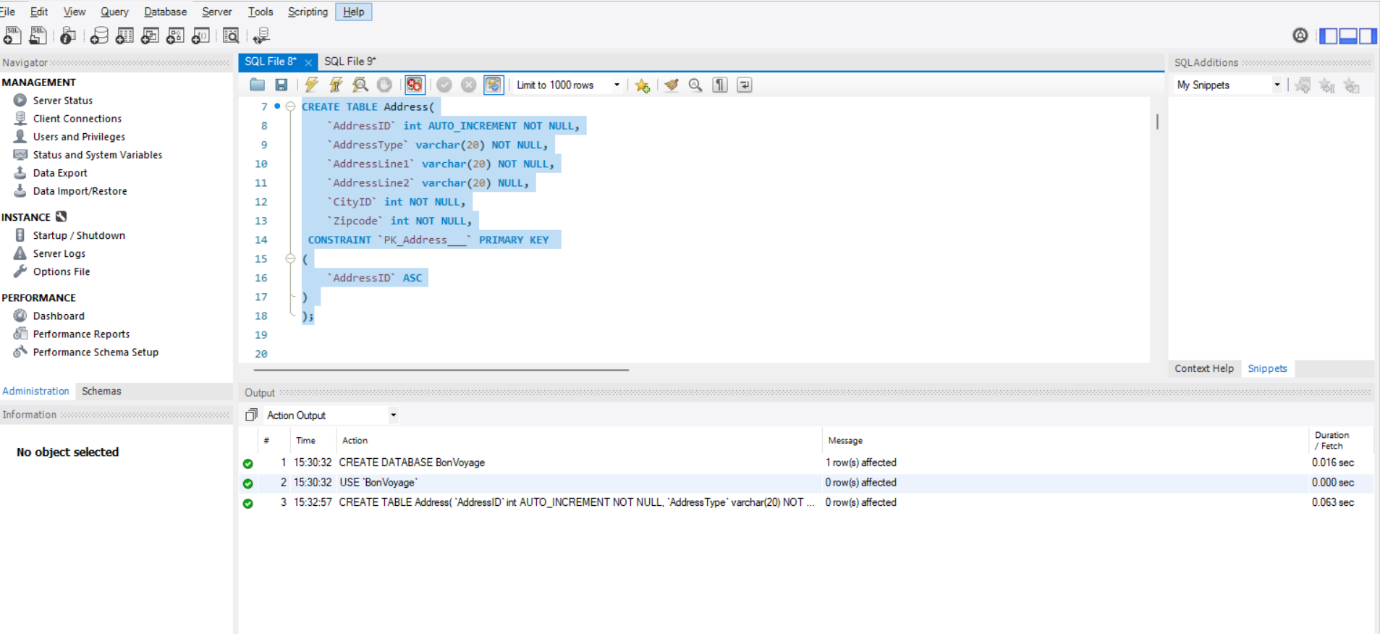
**Rooms Table:**



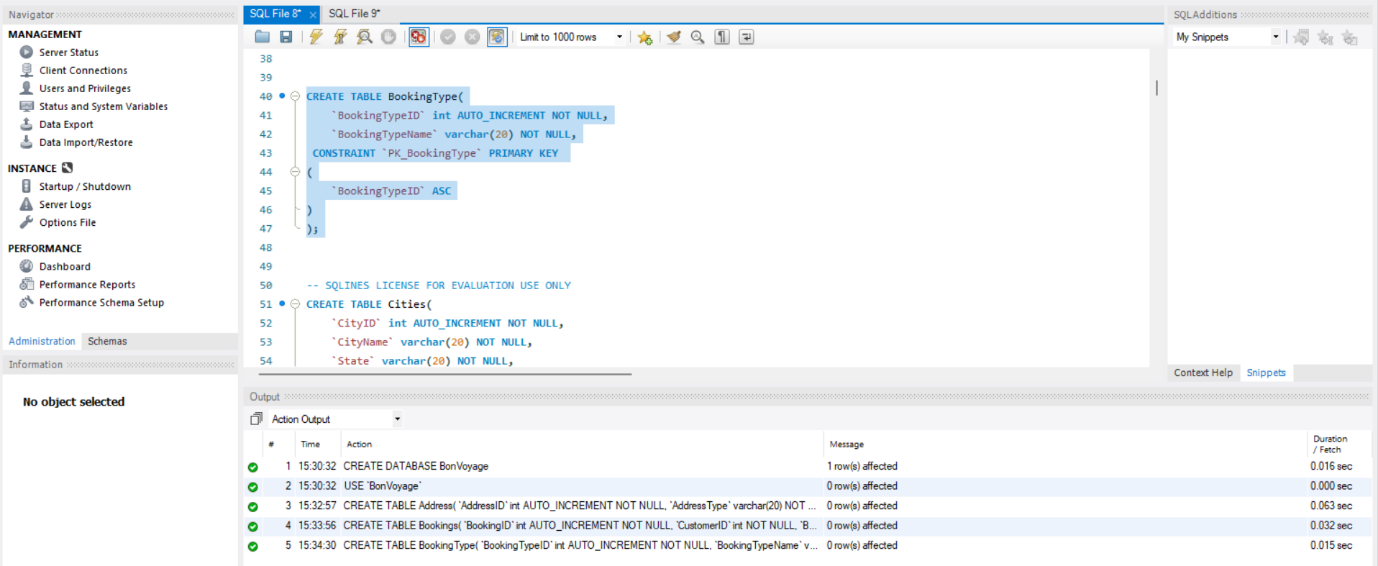
**Tourists Attraction Table:**



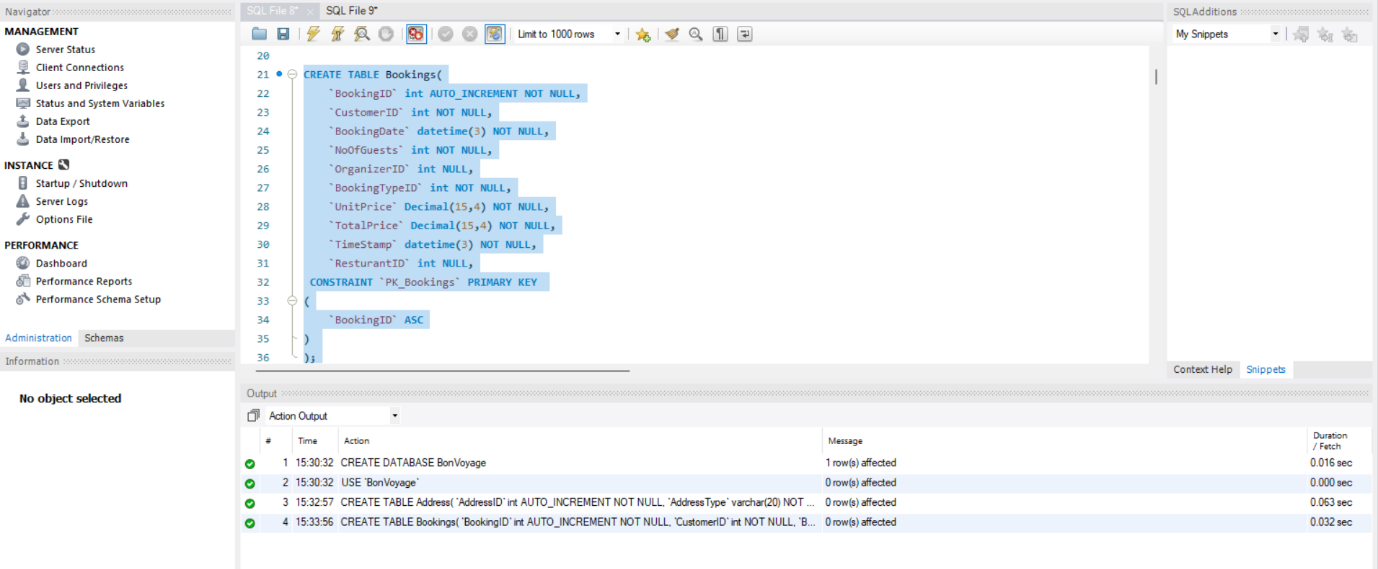
**Address Table:**



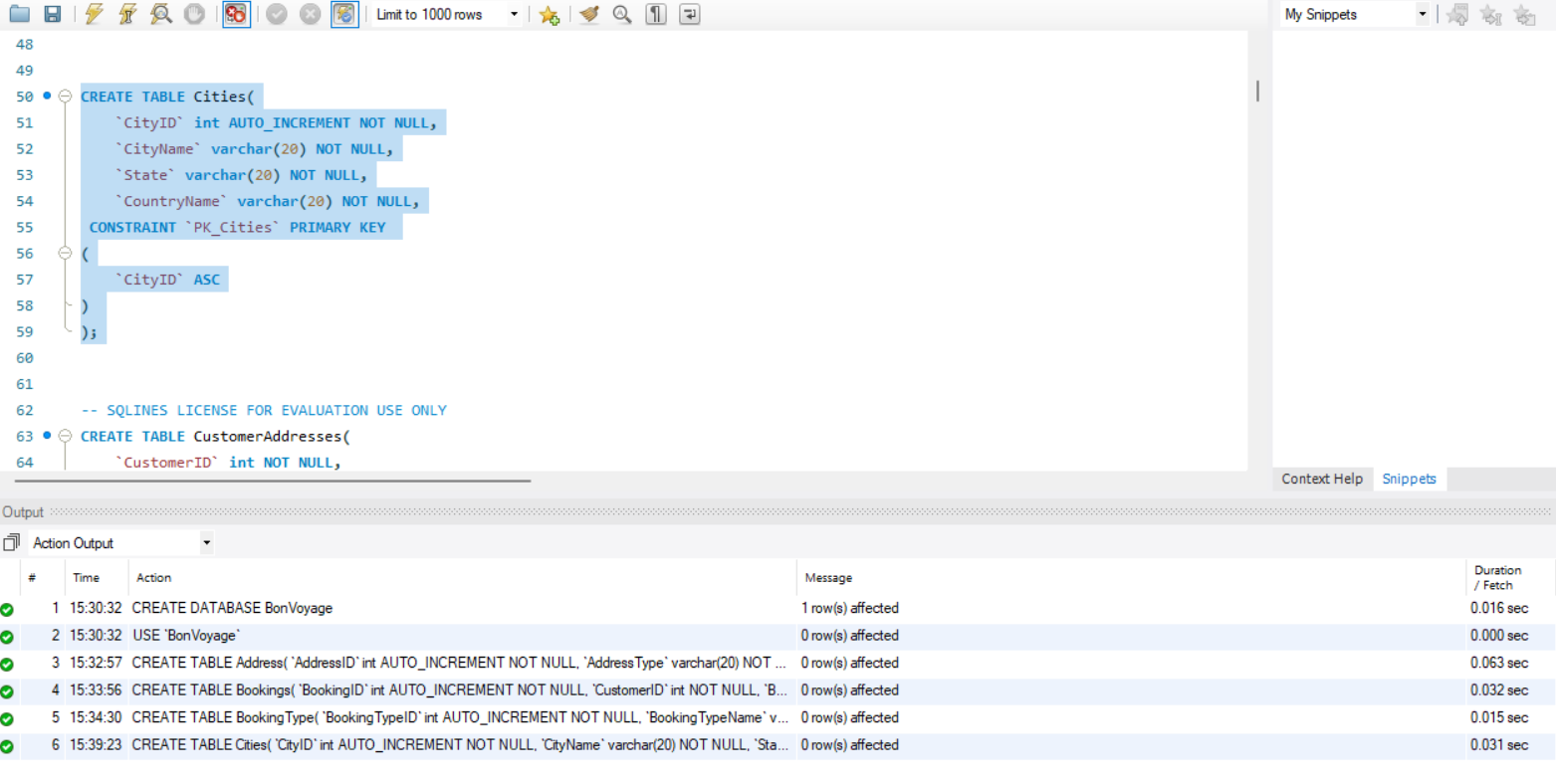
**Booking Type Table:**



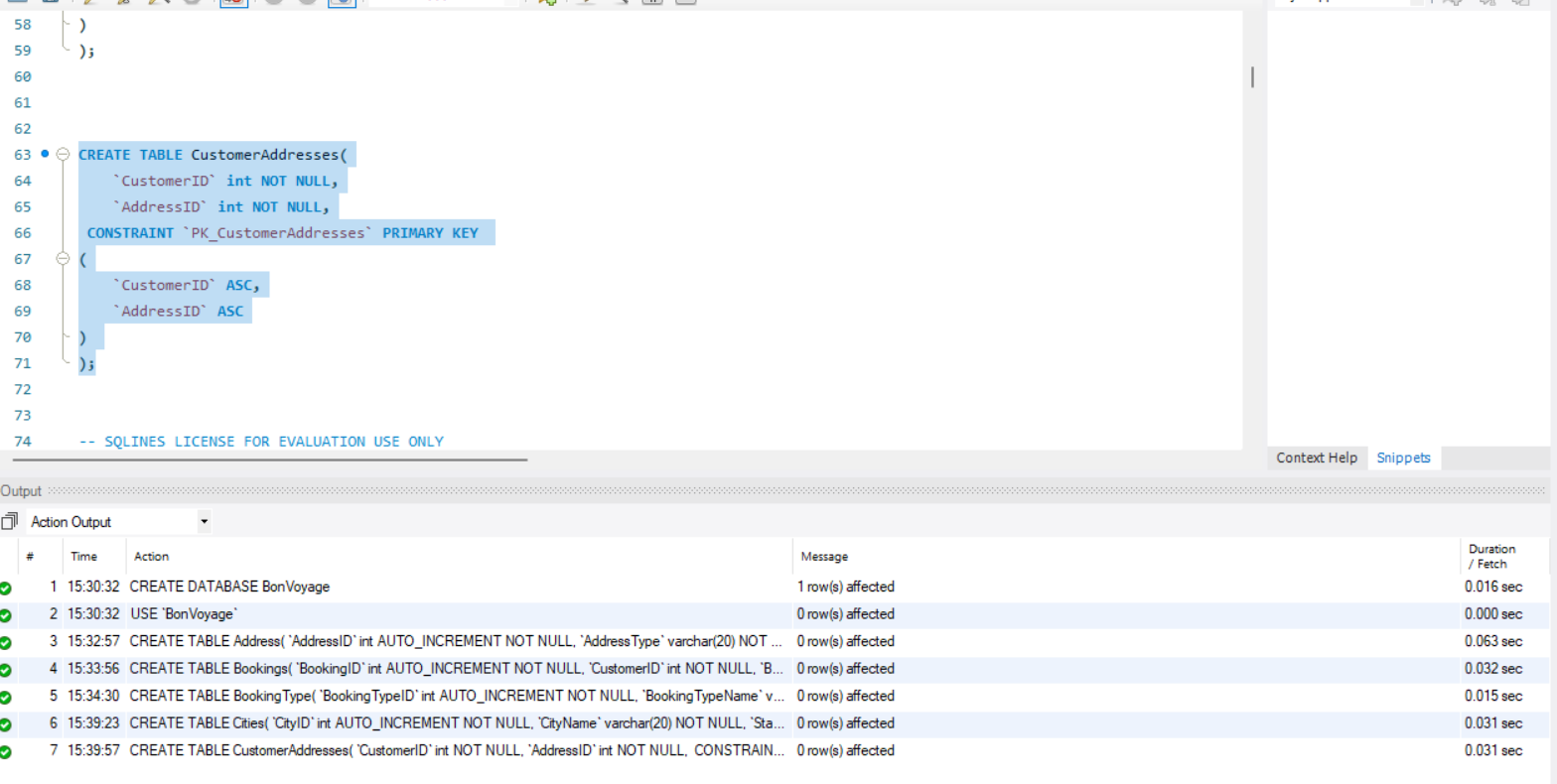
**Bookings Table:**



**Cities Table:**



**Customer Address Table:**



**Data Manipulation Language Scripts**

**Insert statements:**

**Insering into Booking type table:**

INSERT BookingType (`BookingTypeID`, `BookingTypeName`) VALUES (1, N'resturant booking');

INSERT BookingType (`BookingTypeID`, `BookingTypeName`) VALUES (2, N'hotel booking');

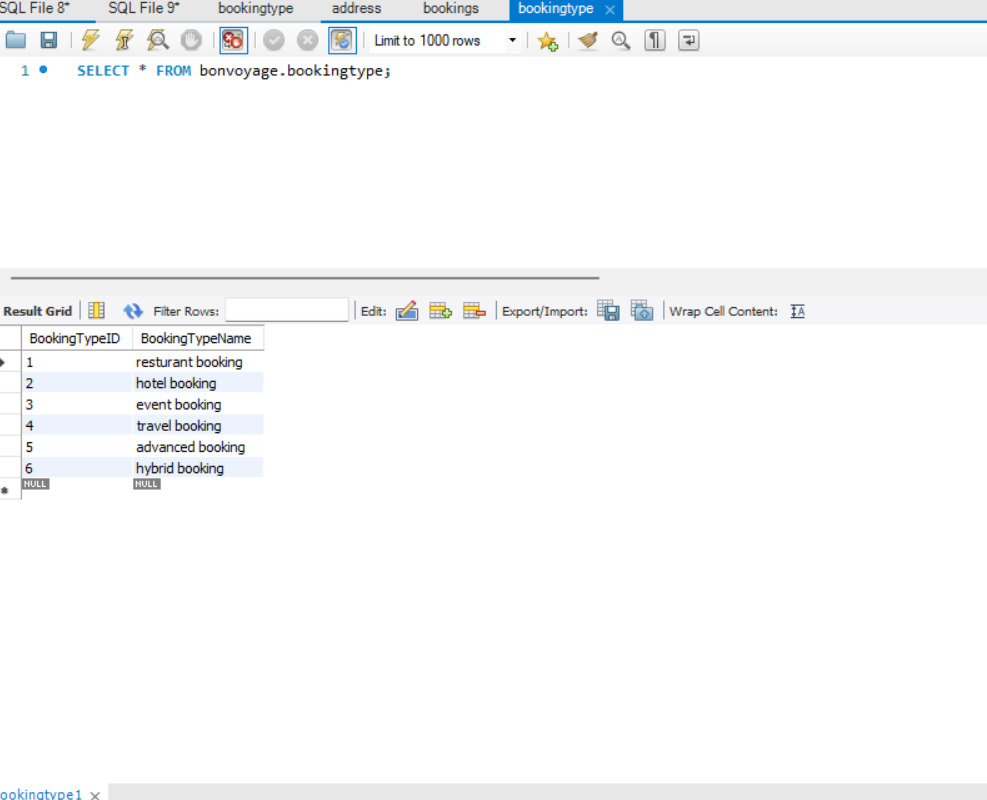
INSERT BookingType (`BookingTypeID`, `BookingTypeName`) VALUES (3, N'event booking');

INSERT BookingType (`BookingTypeID`, `BookingTypeName`) VALUES (4, N'travel booking');

INSERT BookingType (`BookingTypeID`, `BookingTypeName`) VALUES (5, N'advanced booking');

INSERT BookingType (`BookingTypeID`, `BookingTypeName`) VALUES (6, N'hybrid booking');

**Output:**



**Inserting into Events Table:**

INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (1, 1, 25, N'New year parade', 16.0000, N'new year ', N'www.newyear.com');

INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (2, 3, 26, N'duty free shopping', 8000.0000, N'pure shopping', N'www.letsbuy.com');

INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (3, 4, 27, N'herb', 40.0000, N'health and pill', N'www.herb.com');

INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (4, 5, 28, N'valentine journey', 2000.0000, N'romantic', N'www.couple.com');

INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (6, 6, 25, N'elder people', 300.0000, N'elder people', N'www.randytour.com');

INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (7, 7, 26, N'big fun', 300.0000, N'500 people on beach', N'www.petetravel.com');

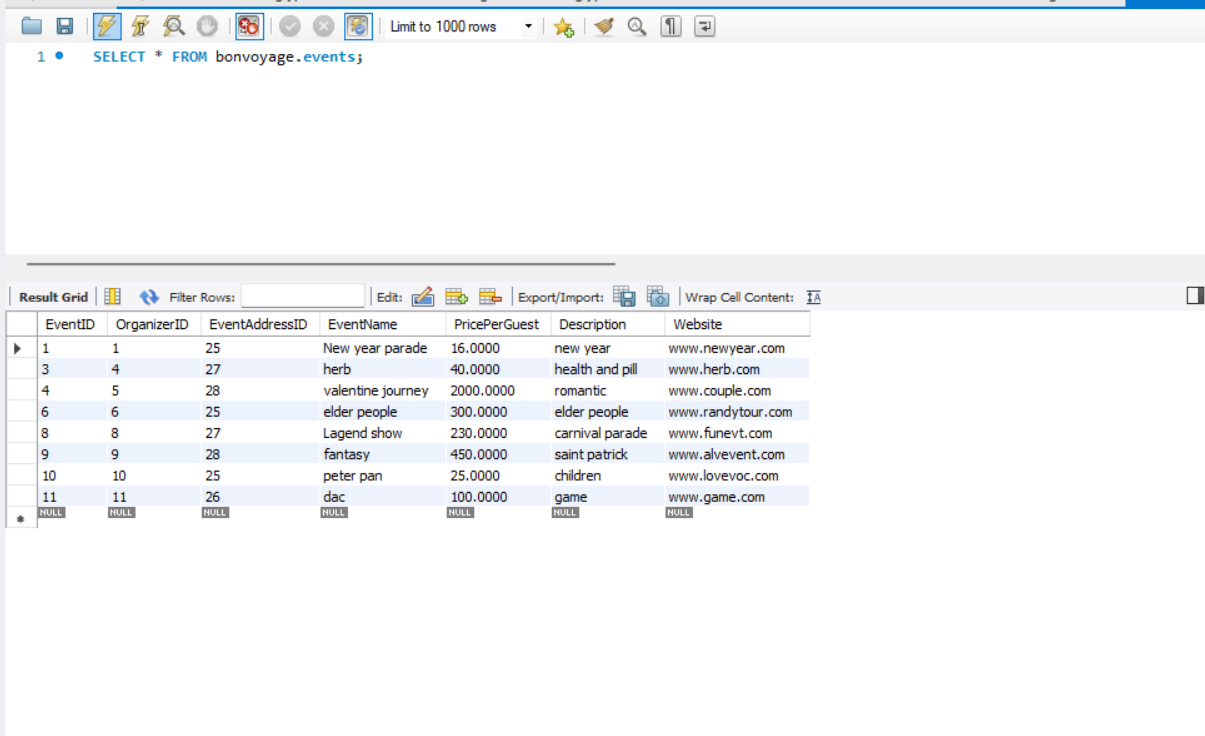
INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (8, 8, 27, N'Lagend show', 230.0000, N'carnival parade', N'www.funevt.com');

INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (9, 9, 28, N'fantasy', 450.0000, N'saint patrick', N'www.alvevent.com');

INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (10, 10, 25, N'peter pan', 25.0000, N'children', N'www.lovevoc.com');

INSERT Events (`EventID`, `OrganizerID`, `EventAddressID`, `EventName`, `PricePerGuest`, `Description`, `Website`) VALUES (11, 11, 26, N'dac', 100.0000, N'game', N'www.game.com');

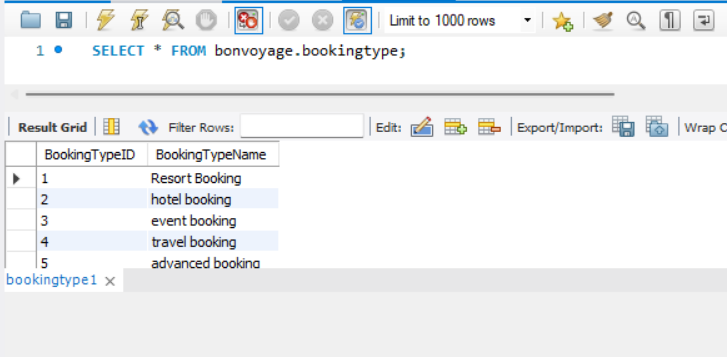
**Output:**



**Update Statements:**

1. UPDATE BookingType

SET BookingTypeName = "Resort Booking" WHERE BookingTypeID=1;

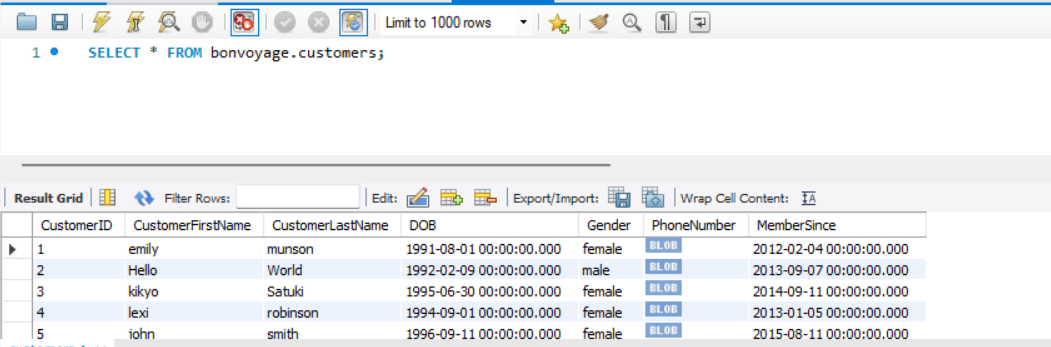


2)

UPDATE Customers

SET CustomerFirstName = "Hello", CustomerLastName="World"

WHERE CustomerID=2;

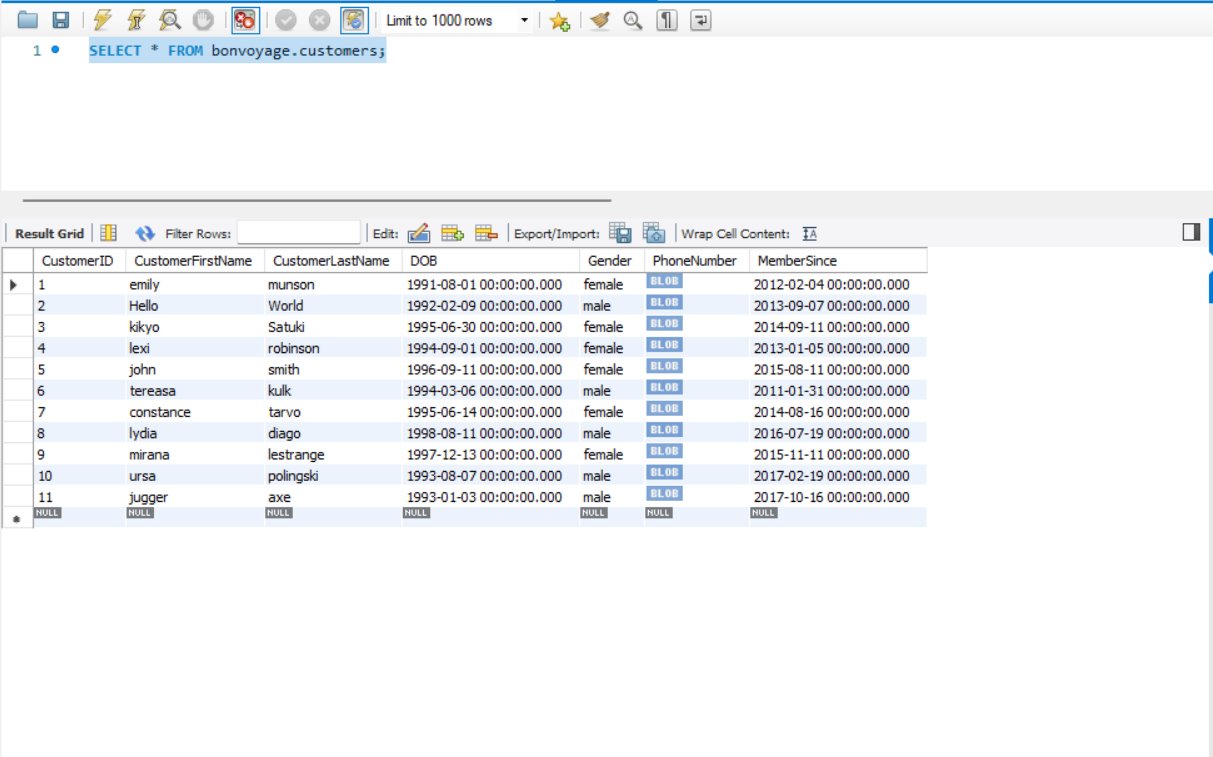


**Delete Statements:**

DROP TABLE rooms;

**Select Statement:**

SELECT \* FROM bonvoyage.customers;

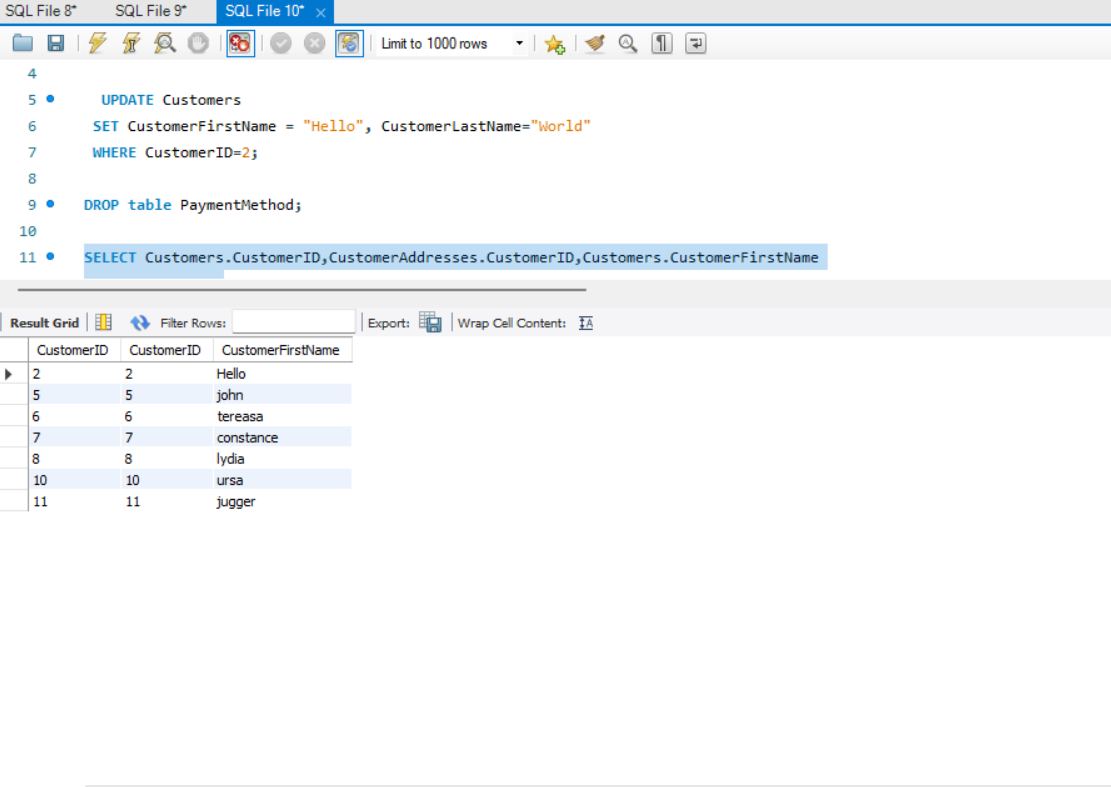


**Join Statement:**

SELECT Customers.CustomerID,CustomerAddresses.CustomerID,Customers.CustomerFirstName

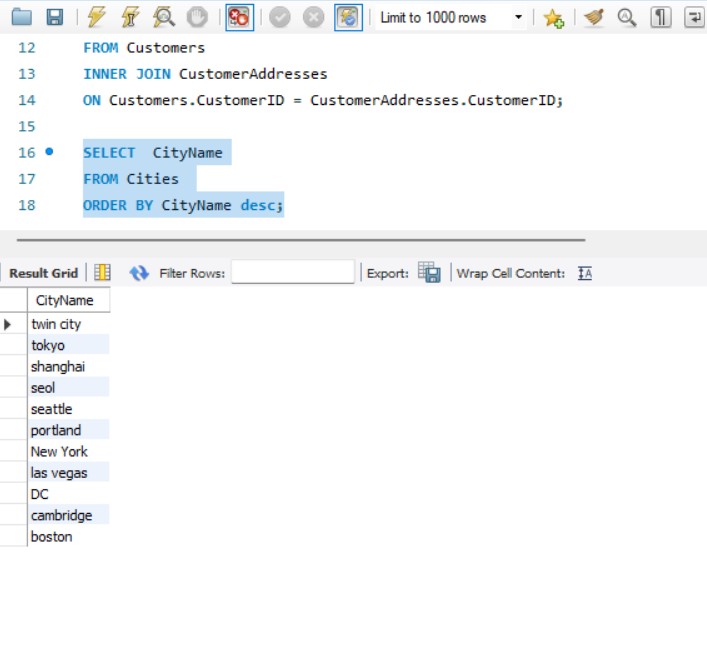
FROM Customers INNER JOIN CustomerAddresses

ON Customers.CustomerID = CustomerAddresses.CustomerID;

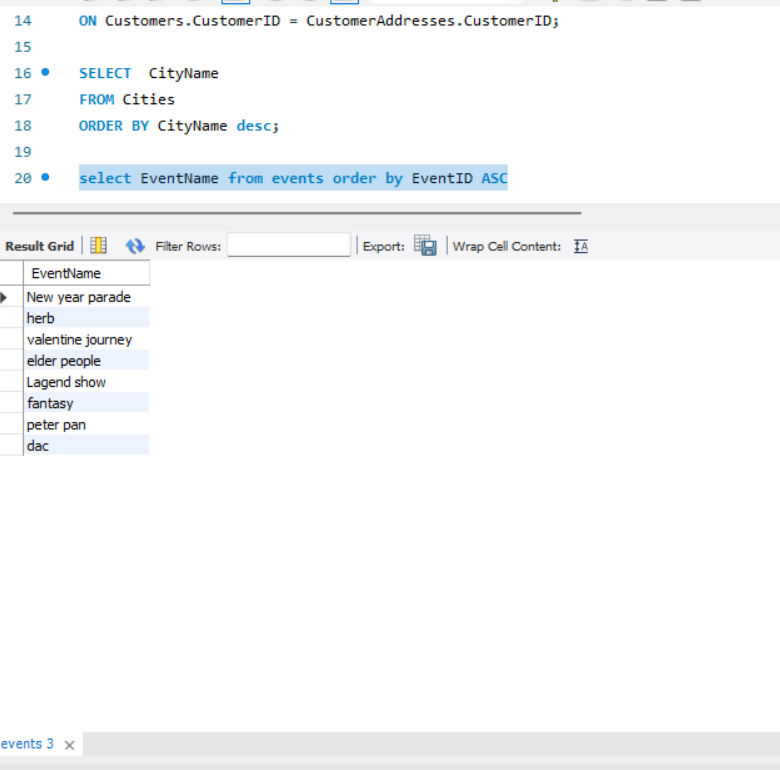


**Own Queries:**

SELECT CityName FROM Cities ORDER BY CityName desc;



select EventName from events order by EventID ASC;



**Views:**

**View 1 Organizers and their details**

CREATE VIEW vw\_EventOrganizerDetails

AS

SELECT eo.OrganizerName,

ad.AddressLine1 AS Address,

eo.PhoneNo,

ct.CityName,

eo.Website

FROM EventOrganizer AS eo

JOIN Address AS ad

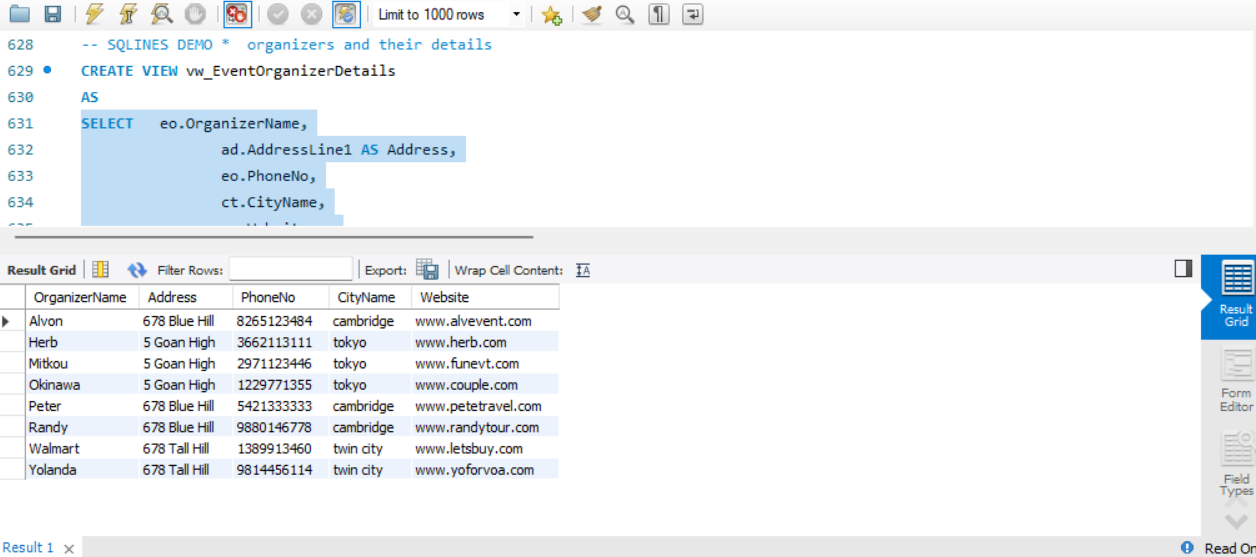
ON eo.OrganizerAddressID = ad.AddressID

JOIN Cities AS ct

ON ad.CityID = ct.CityID

ORDER BY `OrganizerName` ASC

LIMIT 20;



**View 2: 100 hotel bookings based on hotels, ordered by the no. of bookings**

CREATE VIEW vw\_HotelMaxBookings

AS

SELECT h.HotelName,

(SELECT CityName FROM Cities city

WHERE city.CityID = h.CityID) CityName,

a.AddressLine1 Address,

h.PhoneNumber,

(SELECT COUNT(booking.BookingID)

FROM RoomBookings booking

WHERE r.RoomID = booking.RoomID) AS 'NumberOfBookings'

FROM Hotels h,

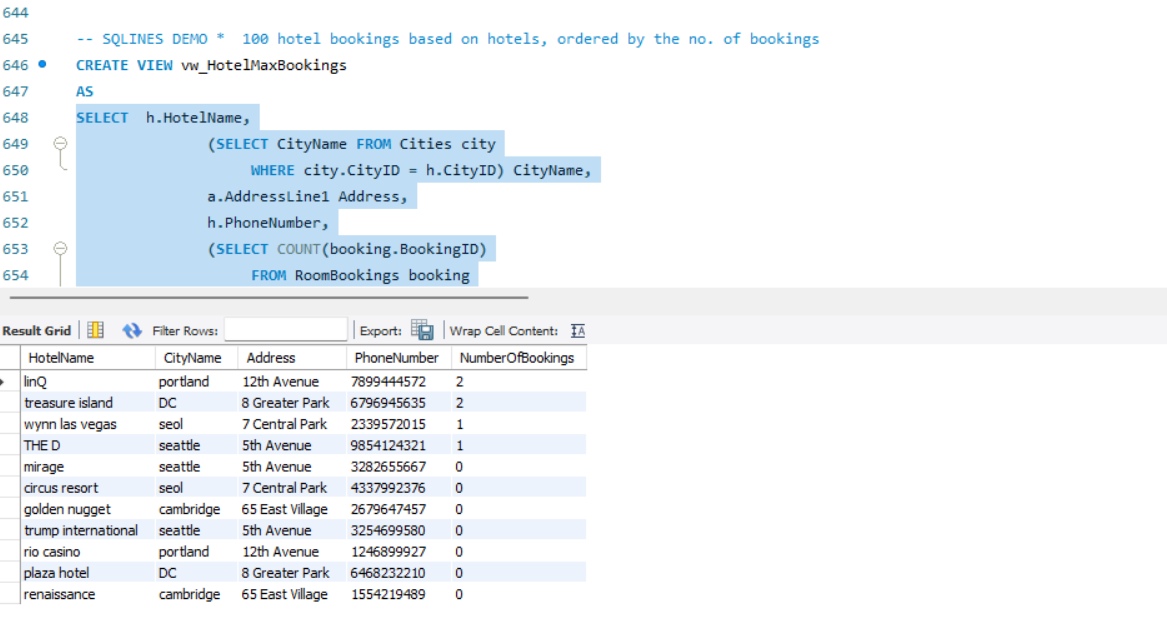
Rooms r,

Address a

WHERE h.HotelID = r.HotelID AND h.HotelAddressID = a.AddressID

ORDER BY `NumberOfBookings` DESC

LIMIT 20;



**Triggers:**

CREATE TRIGGER tr\_EncryptCustPhoneNo

ON Customers

AFTER INSERT AS

BEGIN

-- Create DMK

CREATE MASTER KEY

ENCRYPTION BY PASSWORD = 'Test\_P@sswOrd';

-- SQLINES DEMO \*\*\* e to protect symmetric key

CREATE CERTIFICATE TestCertificate

WITH SUBJECT = 'BonVoyage Test Certificate',

EXPIRY\_DATE = '2026-12-31';

-- SQLINES DEMO \*\*\* key to encrypt data

CREATE SYMMETRIC KEY TestSymmetricKey

WITH ALGORITHM = AES\_128

ENCRYPTION BY CERTIFICATE TestCertificate;

-- Open symmetric key

OPEN SYMMETRIC KEY TestSymmetricKey DECRYPTION BY CERTIFICATE TestCertificate;

-- Insert

UPDATE Customers SET `PhoneNumber`=(EncryptByKey(Key\_GUID(N'TestSymmetricKey'), (SELECT `PhoneNumber` FROM inserted)))

WHERE CustomerID=(SELECT CustomerID FROM inserted);

-- SQLINES DEMO \*\*\* able with decrypted names

-- SQLINES DEMO \*\*\* omers]

-- SQLINES DEMO \*\*\* = DecryptByKey( [PhoneNumber]);

-- SQLINES DEMO \*\*\* ic key

CLOSE SYMMETRIC KEY TestSymmetricKey;

-- SQLINES DEMO \*\*\* c key

DROP SYMMETRIC KEY TestSymmetricKey;

-- SQLINES DEMO \*\*\* ate

DROP CERTIFICATE TestCertificate;

-- Drop the DMK

DROP MASTER KEY;

**Python Programming:**

from os import close

import sys

import mysql.connector

from mysql.connector import errorcode

try:

   mydb = mysql.connector.connect(

      user="root",

      password="root",

      host="localhost",

      database="BonVoyage")

except mysql.connector.Error as err:

   if err.errno == errorcode.ER\_ACCESS\_DENIED\_ERROR:

      print("Invalid credentials")

   elif err.errno == errorcode.ER\_BAD\_DB\_ERROR:

      print("Database not found")

   else:

      print("Cannot connect to database:", err)

else:

    cursor1 = mydb.cursor()

# Events Table Information

    query1 = ("SELECT \* FROM Events")

    cursor1.execute(query1)

    print("-------------------------------------Events Table Information---------------------------------------\n")

    for row in cursor1.fetchall():

        print(row,"\n")

# Address Table

    query2=("SELECT \* FROM address")

    cursor1.execute(query2)

    print("-------------------------------------------Address Table-------------------------------------------- \n")

    for row1 in cursor1.fetchall():

        print(row1,"\n")

# Bookings Table

    query3=("SELECT \* FROM Bookings")

    cursor1.execute(query3)

    print("--------------------------------------------Bookings Table-------------------------------------------- \n")

    for row in cursor1.fetchall():

       print(row,"\n")

# Booking Type Table

    query4=("SELECT \* FROM BookingType")

    cursor1.execute(query4)

    print("-------------------------------------------Booking Type Table------------------------------------------- \n")

    for row in cursor1.fetchall():

       print(row,"\n")

# Cities Table

    query5=("SELECT \* FROM Cities")

    cursor1.execute(query5)

    print("-------------------------------------------Cities Table------------------------------------------------- \n")

    for row in cursor1.fetchall():

       print(row,"\n")

# Customer Address Table

    query6=("SELECT \* FROM CustomerAddresses")

    cursor1.execute(query6)

    print("---------------------------------------Customer Address Table--------------------------------------------")

    for row in cursor1.fetchall():

       print(row,"\n")

# Event Organizer Table

    query8=("SELECT \* FROM EventOrganizer")

    cursor1.execute(query8)

    print("-----------------------------------------Event Organizer Table--------------------------------------------")

    for row in cursor1.fetchall():

       print(row,"\n")

# Event Schedule Table

    query9=("SELECT \* FROM EventSchedule")

    cursor1.execute(query9)

    print("---------------------------------------------Event Schedule Table-------------------------------------------")

    for row in cursor1.fetchall():

       print(row,"\n")

# Hotels table

    query10=("SELECT \* FROM Hotels")

    cursor1.execute(query10)

    print("--------------------------------------------------Hotels table------------------------------------------------")

    for row in cursor1.fetchall():

       print(row,"\n")

# Orders Table

    query11=("SELECT \* FROM Orders")

    cursor1.execute(query11)

    print("------------------------------------------------Orders Table--------------------------------------------------")

    for row in cursor1.fetchall():

       print(row,"\n")

# Payment Method Table

    query12=("SELECT \* FROM PaymentMethod")

    cursor1.execute(query12)

    print("---------------------------------------------Payment Method Table---------------------------------------------")

    for row in cursor1.fetchall():

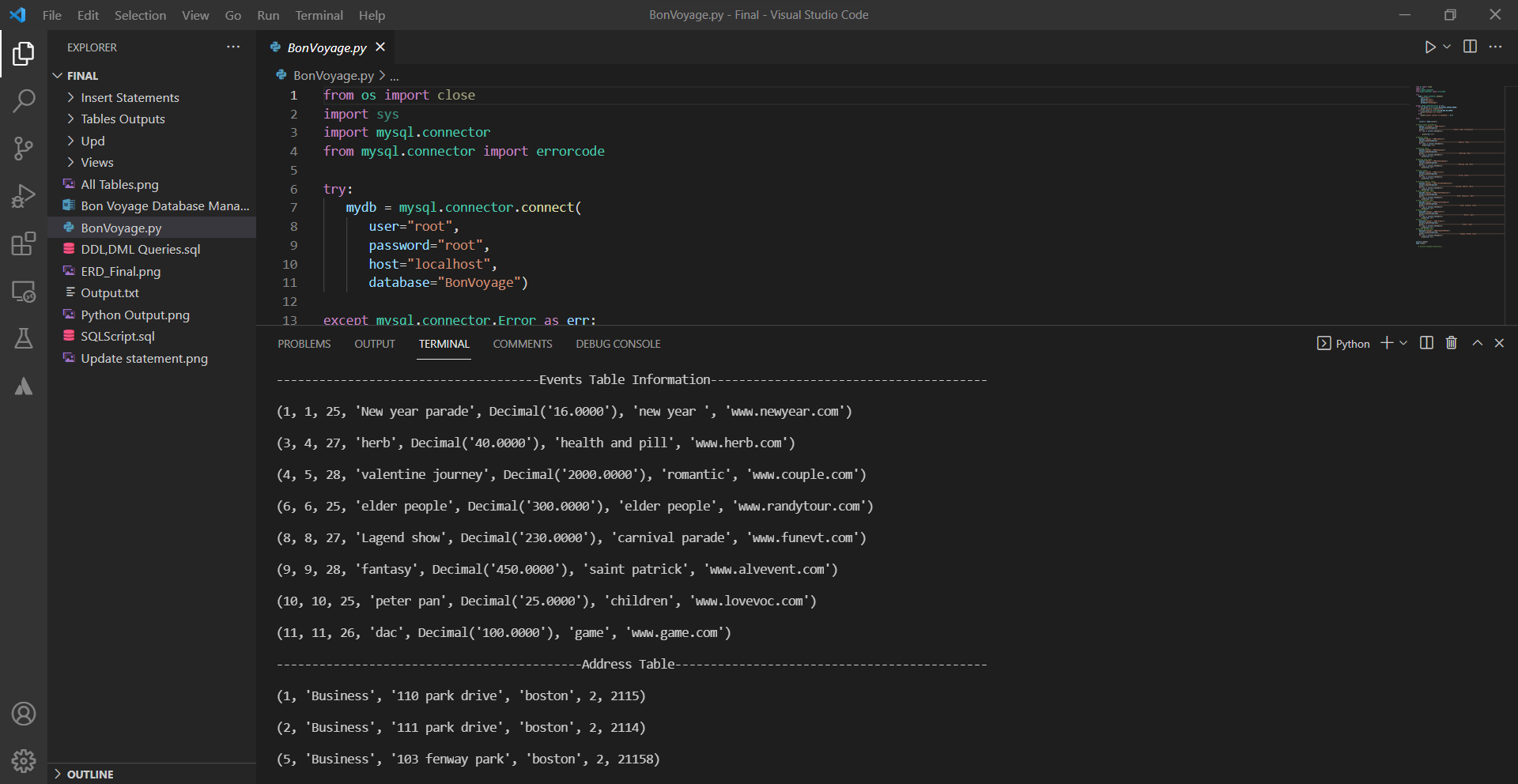
       print(row,"\n")

cursor1.close()

mydb.close()

   # Execute database operations...

**Output:**

****