MAPÚA UNIVERSITY
SCHOOL OF ELECTRICAL, ELECTRONICS, AND COMPUTER ENGINEERING

Experiment 5: Data Modeling and Database

CPE106L (Software Design Laboratory)

Danikka Villaron
Jessa Abigaile Tongol
Mark Vincent De Villa
Matthew Benedict Nepomuceno

Group No.: **2**
Section: **E03**

PreLab

Readings, Insights, and Reflection

Insights and Reflections
A Guide to SQL
9780357419830
De Villa (Chapter 1)

Databases are structures that contain multiple categories of information and their relations.

The TAL Distributors database contains five tables: the customer table – which contains information regarding the customers – the orders table – holding information about the order date and the customer who places the order – the items table – holding necessary information about a product–, and the order lines – linking the information from the order, items, and customer table.

The Colonial Adventure Tours contains four entities in their database: the customer table – comprising details related to a customer's personal information and the booking details – the guide table – composed of information regarding the tour guides, like their personal details and assigned tours – the tours table – presenting information regarding a tour's schedule, id, costs, and locations – and the booking table – presenting the tour bookings made by customers.

The Solmaris Condominium Group contains four tables: the condominium table – presents information on the condominium's vacancy status, price, size, and locations – the owner table – presents information on the property owners' personal and contact information – the renter table – presents information on those that currently reside in specific condominiums provided by the group– and the leases table – presenting the rental agreement details between the owners, renters, and the unit.

De Villa (Chapter 2)

A Relational Database is a collection of tables in a database, formally known as relations. Databases usually contain tables, which are also known as entities, and these entities contain columns, which are known as attributes. The connection between these entities and their attributes is called relationships. The relationships between entities come in many forms. The one-to-many relationship, for example, refers to the relationship of one attribute from a select entity associated with multiple attributes from another entity.

Database Designs require knowing the involved entities, the related attributes, the functional dependencies of attributes, and the relationship between the entities in a database.

After designing and creating a database design, a process called "normalization" takes place, where redundant attributes in the entities of a database become converted to Normal Forms. The First Normal Form simplifies the data of attributes by creating multiple rows for attributes holding numerous information in a single row. The Second Normal Form eliminates partial dependencies of attributes, splitting the entity into multiple simpler entities with one key attribute. The Third Normal Form removes transitive dependencies on non-key attributes, ensuring that dependencies reach only one key attribute.

Core Python Programming
9789351198918
<Nepomuceno> (Chapter 24)

Python supports a wide range of databases, including MySQL, PostgreSQL, and SQLite. Both commercial databases like Oracle and free, open-source options are compatible, providing flexibility for various development needs. Adapters are essential for establishing these database connections

To connect Python with MySQL, the MySQLdb adapter is commonly used. It enables creating tables, inserting, updating, and retrieving data through SQL commands. With functions like connect() and cursor(), developers can efficiently manage database operations

Python Programming
9781118908891
Nepomuceno (Chapter 3)

Chapter 3 is all about Relational Database Concepts and it dives into managing data with SQL, focusing on constraints, relationships, and how SQLite handles data types. It explains how constraints like NOT NULL or PRIMARY KEY help keep the data consistent and clean. For instance, linking tables using FOREIGN KEY ensures proper relationships, like an employee always having a valid manager.

What stands out is how SQLite treats data types as more of a guideline rather than strict rules, which makes it flexible but can also cause unexpected results. The chapter also talks about designing tables and relationships, like a book-author example to show how many-to-many links work.

Overall, this part of the chapter shows how important it is to think carefully about database design to avoid issues later on. The practical examples make it easier to see how these ideas work in real-life databases.

De Villa (Chapter 3)

The Structured Query Language (SQL) is the selected standard software tool for relational database manipulation, where their expressions are referred to as queries regardless of whether the functions return data or not.

SQL is composed of a Data Definition Language (DDL), which contains the set of commands to create and change the database structure, and a Data Manipulation Language (DML), which includes the set of commands to manipulate the content of databases.

Questions & Answers

1. What are DML and DDL statements in Structured Query Language? Give examples of each.

DML is a statement in Structured Query Language to manage data stored in a database. These statements allow users to access, insert, update and delete data in database tables. For example, the SELECT statement is used to query and extract specific data from a table, while the INSERT statement adds new records to a table. The UPDATE statement changes existing records, while the DELETE statement removes data from the table. Meanwhile, DDL commands are used to define and manage the structure of databases and their objects (such as tables, schemes, indexes). These statements are responsible for creating, modifying and deleting database structures, but do not manipulate actual data. Examples of DDL statements are CREATE TABLE, which is used to create new tables; ALTER TABLE, which changes the structure of existing tables; and DROP TABLE, which removes a table from the database.

2. What are the categories of SQLite Functions? Give 3 examples of each category.

SQLite functions can be divided into five main types: aggregate functions, date and time functions, string functions, mathematical functions and system functions. The aggregate function calculates multiple data rows and returns a single result, such as SUM() to calculate the number of rows, AVG() to calculate the average, and COUNT() to calculate the number of rows. The date and time function is used to manipulate and format the date and time values, such as date() for e-extracting the date, strftime() for formatting the date, and now() for obtaining the current date and time. String functions operate on text data, for example, LENGTH() finds the length of a string, UPPER() converts text to uppercase, and SUBSTR() extracts substrings. These categories help improve SQLite functionality by enabling various data operations.

3. How do you check if you have SQLite installed in a system using the Linux terminal?

Open the terminal and enter the command sqlite3 to see if SQLite is installed on a Linux machine. This command will start the SQLite command-line interface if SQLite is installed, and a prompt similar to sqlite> will appear. The error message "the command was not found" will appear if SQLite is not installed. In these situations, you can use your package manager to install SQLite; for instance, sudo apt install sqlite3 for Debian-based distributions.

PostLab

Programming Problems

Note: Leaders should assign the problems to members Download SQL Scripts here >> SQL Scripts (Colonial, etc.)

A. Machine Problems

1. Colonial Adventure Tours is considering offering outdoor adventure classes to prepare people to participate in hiking, biking, and paddling adventures. Only one class is taught on any given day. Participants can enroll in one or more classes. Classes are taught by the guides that Colonial Adventure employs. Participants do not know who the instructor for a particular class will be until the day of the class. Colonial Adventure Tours needs your help with the database design for this new venture. In each step, represent your answer using the shorthand representation and a diagram. Use crow's foot notation for the diagram. Follow the sample SQLite chinook database ERD (Download it from Blackboard Course Materials)

a) For each participant, list his or her number, last name, first name, address, city, state, postal code, telephone number, and date of birth.

b) For each adventure class, list the class number, class description, maximum number of people in the class, and class fee.

c) For each participant, list his or her number, last name, first name, and the class number, class description, and date of the class for each class in which the participant is enrolled.

d) For each class, list the class date, class number, and class description; and the number, last name, and first name of each participant in the class.

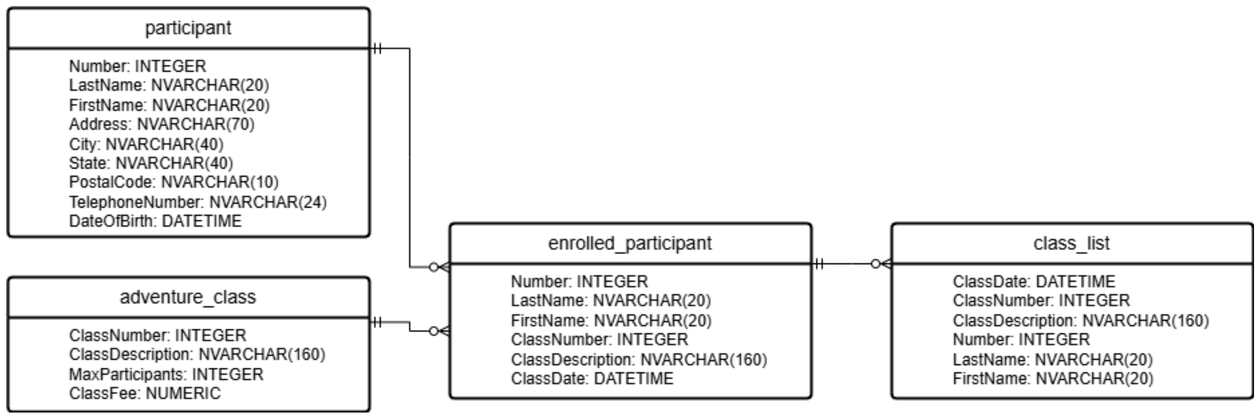
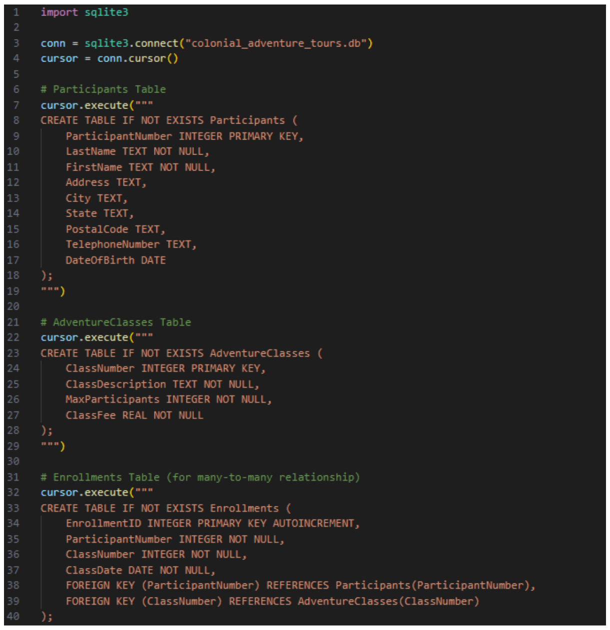


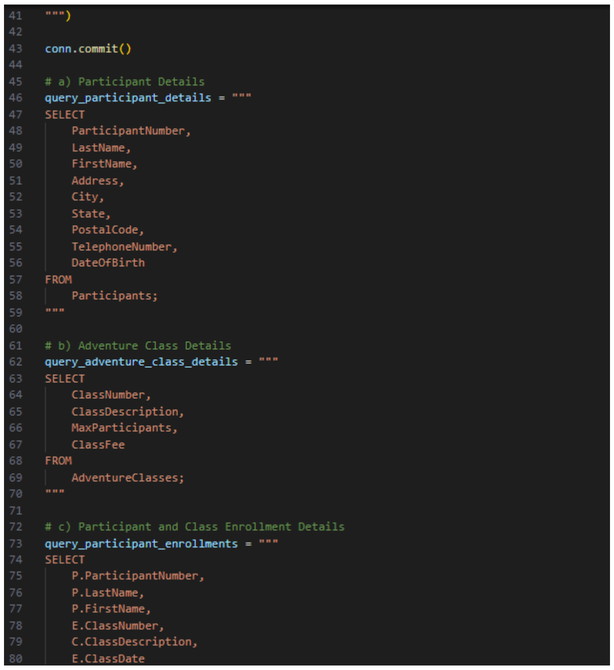
Figure 1.1. Colonial Adventure Tours Database Diagram

Figure 1.1 presents the relations between the tables in the Colonial Adventure Tours database. The participant table provides multiple mandated data for the enrolled_participant table and obtains the rest from the adventure_class table. The class_list table also receives data from the enrolled_participant table.



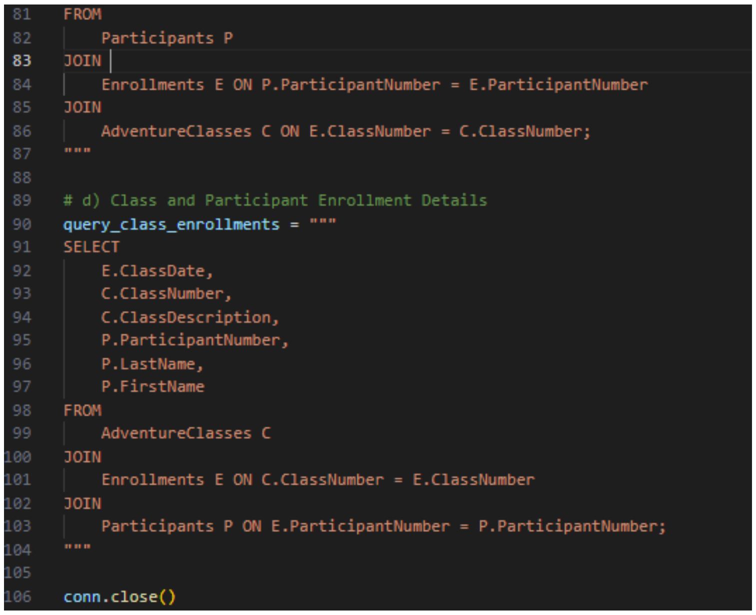
```python
import sqlite3

conn = sqlite3.connect("colonial_adventure_tours.db")
cursor = conn.cursor()

# Participants Table
cursor.execute("""
CREATE TABLE IF NOT EXISTS Participants (
    ParticipantNumber INTEGER PRIMARY KEY,
    LastName TEXT NOT NULL,
    FirstName TEXT NOT NULL,
    Address TEXT,
    City TEXT,
    State TEXT,
    PostalCode TEXT,
    TelephoneNumber TEXT,
    DateOfBirth DATE
);
""")

# AdventureClasses Table
cursor.execute("""
CREATE TABLE IF NOT EXISTS AdventureClasses (
    ClassNumber INTEGER PRIMARY KEY,
    ClassDescription TEXT NOT NULL,
    MaxParticipants INTEGER NOT NULL,
    ClassFee REAL NOT NULL
);
""")

# Enrollments Table (for many-to-many relationship)
cursor.execute("""
CREATE TABLE IF NOT EXISTS Enrollments (
    EnrollmentID INTEGER PRIMARY KEY AUTOINCREMENT,
    ParticipantNumber INTEGER NOT NULL,
    ClassNumber INTEGER NOT NULL,
    ClassDate DATE NOT NULL,
    FOREIGN KEY (ParticipantNumber) REFERENCES Participants(ParticipantNumber),
    FOREIGN KEY (ClassNumber) REFERENCES AdventureClasses(ClassNumber)
);
```

Figure 1.2.1 Colonial Adventure Tours Database Code Part 1



```python
""")

conn.commit()

# a) Participant Details
query_participant_details = """
SELECT
    ParticipantNumber,
    LastName,
    FirstName,
    Address,
    City,
    State,
    PostalCode,
    TelephoneNumber,
    DateOfBirth
FROM
    Participants;
"""

# b) Adventure Class Details
query_adventure_class_details = """
SELECT
    ClassNumber,
    ClassDescription,
    MaxParticipants,
    ClassFee
FROM
    AdventureClasses;
"""

# c) Participant and Class Enrollment Details
query_participant_enrollments = """
SELECT
    P.ParticipantNumber,
    P.LastName,
    P.FirstName,
    E.ClassNumber,
    C.ClassDescription,
    E.ClassDate
```

Figure 1.2.2 Colonial Adventure Tours Database Code Part 2



```
FROM
    Participants P
JOIN
    Enrollments E ON P.ParticipantNumber = E.ParticipantNumber
JOIN
    AdventureClasses C ON E.ClassNumber = C.ClassNumber;
"""

# d) Class and Participant Enrollment Details
query_class_enrollments = """
SELECT
    E.ClassDate,
    C.ClassNumber,
    C.ClassDescription,
    P.ParticipantNumber,
    P.LastName,
    P.FirstName
FROM
    AdventureClasses C
JOIN
    Enrollments E ON C.ClassNumber = E.ClassNumber
JOIN
    Participants P ON E.ParticipantNumber = P.ParticipantNumber;
"""

conn.close()
```

Figure 1.2.3 Colonial Adventure Tours Database Code Part 3

Figures 1.2.1, 1.2.2, and 1.2.3 present the code that declares the Colonial Adventure Tours Database. The code utilizes SQLite3 to create a database file, and the following codes create the tables with the columns as presented in Figure 1.1.

2. Solmaris Condominium Group has many condos that are available as weekly vacation rentals. Design a database to meet the following requirements:

a) For each renter, list his or her number, first name, middle initial, last name, address, city, state, postal code, telephone number, and email address.

b) For each property, list the condo location number, condo location name, address, city, state, postal code, condo unit number, square footage, number of bedrooms, number of bathrooms, maximum number of persons that can sleep in the unit, and the base weekly rate.

c) For each rental agreement, list the renter number, first name, middle initial, last name, address, city, state, postal code, telephone number, start date of the rental, end date of the rental, and the weekly rental amount. The rental period is one or more weeks.

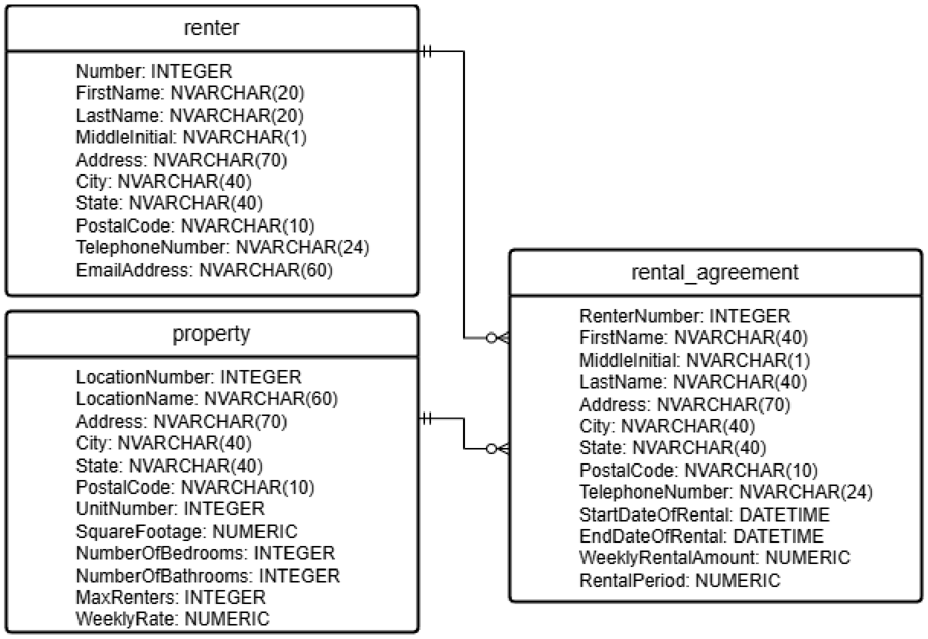
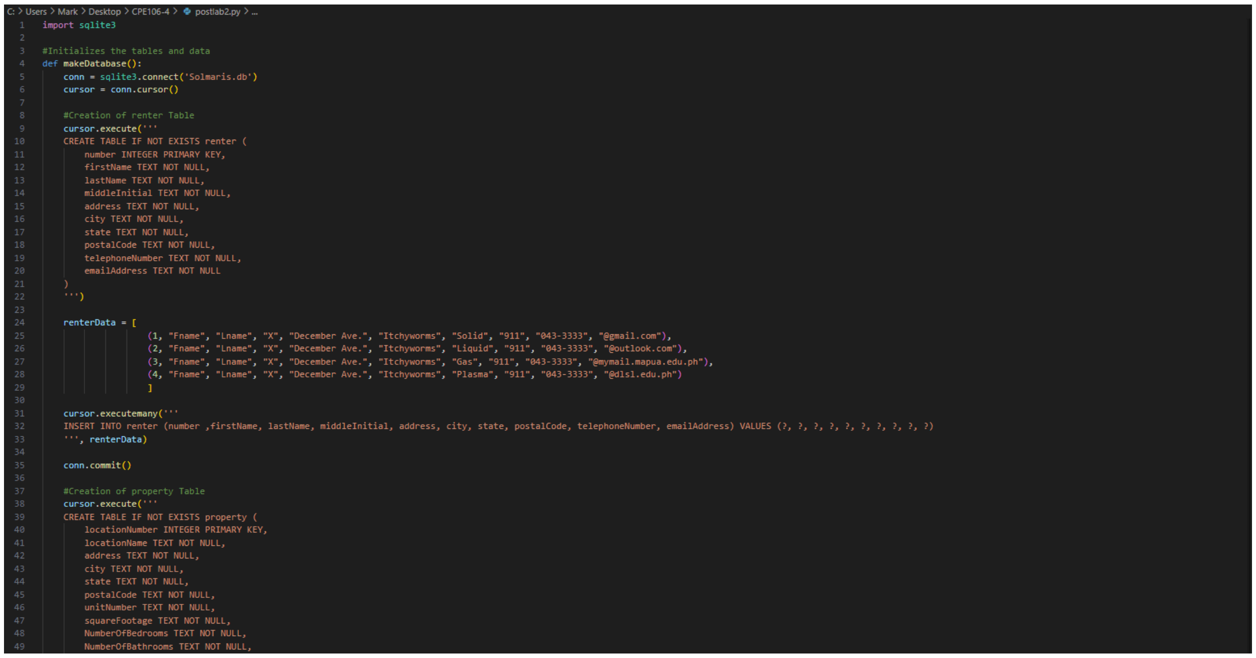


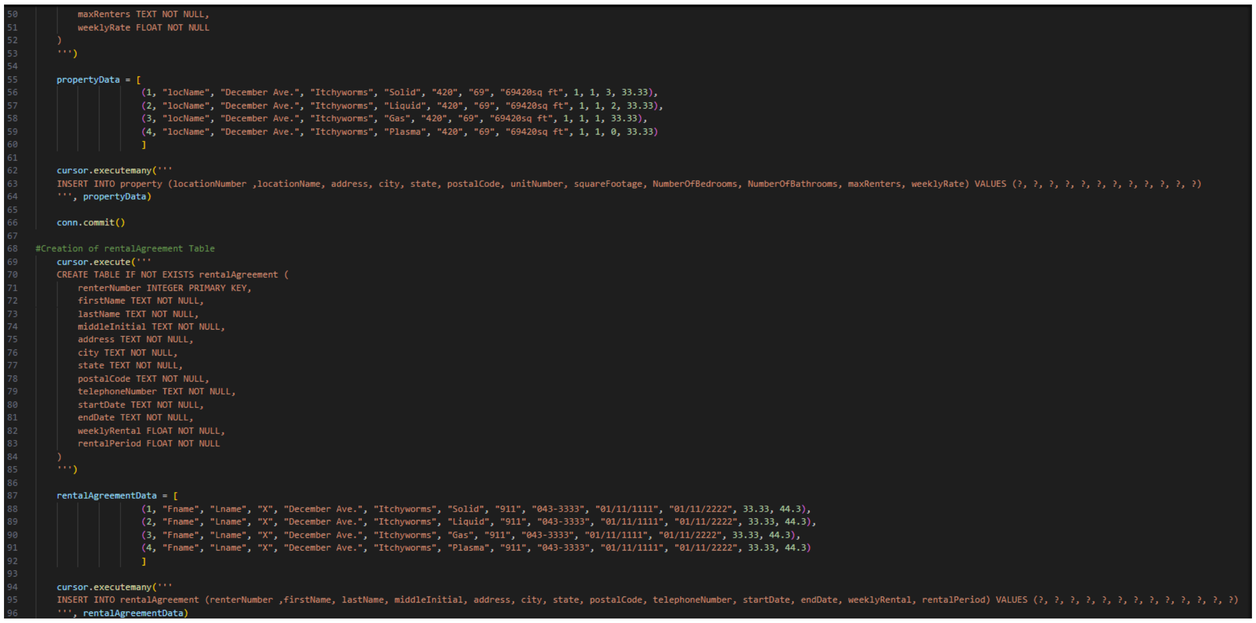
Figure 2.1. Solmaris Condominium Group Database Diagram

Figure 2.1 presents the Solmaris Condominium Group Database Diagram to visualize the tables and their respective columns present in the database, along with the relation of each table. The renter table and the property table provide data necessary for the rental agreement table.



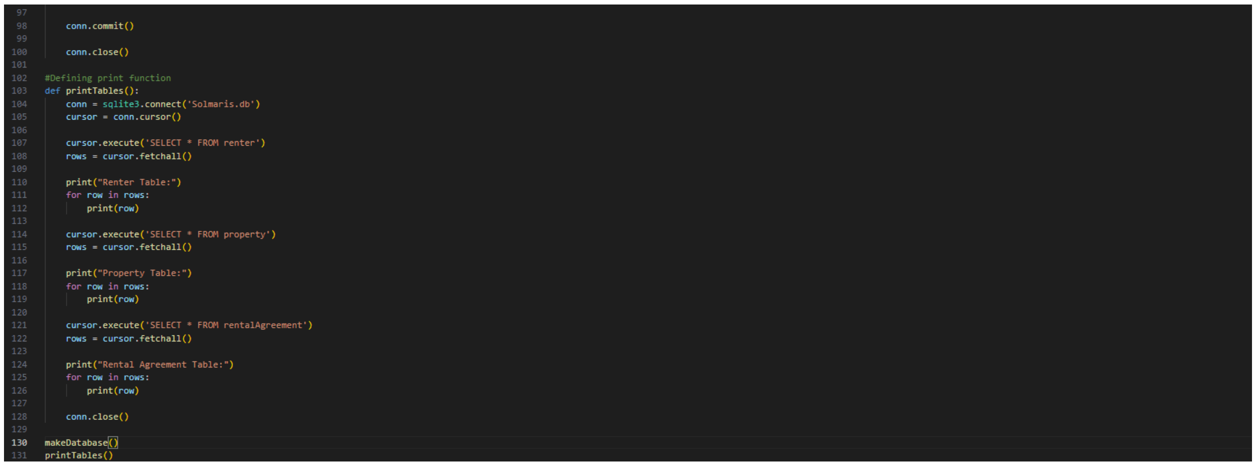
```python
import sqlite3

#Initializes the tables and data
def makeDatabase():
    conn = sqlite3.connect('Solmaris.db')
    cursor = conn.cursor()

    #Creation of renter Table
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS renter (
        number INTEGER PRIMARY KEY,
        firstName TEXT NOT NULL,
        lastName TEXT NOT NULL,
        middleInitial TEXT NOT NULL,
        address TEXT NOT NULL,
        city TEXT NOT NULL,
        state TEXT NOT NULL,
        postalCode TEXT NOT NULL,
        telephoneNumber TEXT NOT NULL,
        emailAddress TEXT NOT NULL
    )
    ''')

    renterData = [
                        (1, "Fname", "Lname", "X", "December Ave.", "Itchyworms", "Solid", "911", "043-3333", "@gmail.com"),
                        (2, "Fname", "Lname", "X", "December Ave.", "Itchyworms", "Liquid", "911", "043-3333", "@outlook.com"),
                        (3, "Fname", "Lname", "X", "December Ave.", "Itchyworms", "Gas", "911", "043-3333", "@mymail.mapua.edu.ph"),
                        (4, "Fname", "Lname", "X", "December Ave.", "Itchyworms", "Plasma", "911", "043-3333", "@dlsl.edu.ph")
                        ]

    cursor.executemany('''
    INSERT INTO renter (number ,firstName, lastName, middleInitial, address, city, state, postalCode, telephoneNumber, emailAddress) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', renterData)

    conn.commit()

    #Creation of property Table
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS property (
        locationNumber INTEGER PRIMARY KEY,
        locationName TEXT NOT NULL,
        address TEXT NOT NULL,
        city TEXT NOT NULL,
        state TEXT NOT NULL,
        postalCode TEXT NOT NULL,
        unitNumber TEXT NOT NULL,
        squareFootage TEXT NOT NULL,
        NumberOfBedrooms TEXT NOT NULL,
        NumberOfBathrooms TEXT NOT NULL,
```

Figure 2.2.1. Solmaris Condominium Group Database Code Part 1



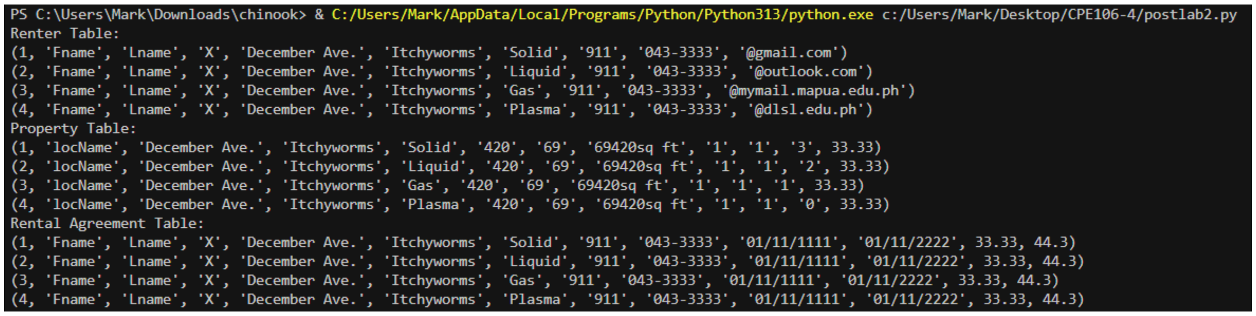
```python
        maxRenters TEXT NOT NULL,
        weeklyRate FLOAT NOT NULL
    )
    ''')

    propertyData = [
                    (1, "locName", "December Ave.", "Itchyworms", "Solid", "420", "69", "69420sq ft", 1, 1, 3, 33.33),
                    (2, "locName", "December Ave.", "Itchyworms", "Liquid", "420", "69", "69420sq ft", 1, 1, 2, 33.33),
                    (3, "locName", "December Ave.", "Itchyworms", "Gas", "420", "69", "69420sq ft", 1, 1, 1, 33.33),
                    (4, "locName", "December Ave.", "Itchyworms", "Plasma", "420", "69", "69420sq ft", 1, 1, 0, 33.33)
                    ]

    cursor.executemany('''
    INSERT INTO property (locationNumber ,locationName, address, city, state, postalCode, unitNumber, squareFootage, NumberOfBedrooms, NumberOfBathrooms, maxRenters, weeklyRate) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', propertyData)

    conn.commit()

#Creation of rentalAgreement Table
    cursor.execute('''
    CREATE TABLE IF NOT EXISTS rentalAgreement (
        renterNumber INTEGER PRIMARY KEY,
        firstName TEXT NOT NULL,
        lastName TEXT NOT NULL,
        middleInitial TEXT NOT NULL,
        address TEXT NOT NULL,
        city TEXT NOT NULL,
        state TEXT NOT NULL,
        postalCode TEXT NOT NULL,
        telephoneNumber TEXT NOT NULL,
        startDate TEXT NOT NULL,
        endDate TEXT NOT NULL,
        weeklyRental FLOAT NOT NULL,
        rentalPeriod FLOAT NOT NULL
    )
    ''')

    rentalAgreementData = [
                    (1, "Fname", "Lname", "X", "December Ave.", "Itchyworms", "Solid", "911", "043-3333", "01/11/1111", "01/11/2222", 33.33, 44.3),
                    (2, "Fname", "Lname", "X", "December Ave.", "Itchyworms", "Liquid", "911", "043-3333", "01/11/1111", "01/11/2222", 33.33, 44.3),
                    (3, "Fname", "Lname", "X", "December Ave.", "Itchyworms", "Gas", "911", "043-3333", "01/11/1111", "01/11/2222", 33.33, 44.3),
                    (4, "Fname", "Lname", "X", "December Ave.", "Itchyworms", "Plasma", "911", "043-3333", "01/11/1111", "01/11/2222", 33.33, 44.3)
                    ]

    cursor.executemany('''
    INSERT INTO rentalAgreement (renterNumber ,firstName, lastName, middleInitial, address, city, state, postalCode, telephoneNumber, startDate, endDate, weeklyRental, rentalPeriod) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', rentalAgreementData)
```

Figure 2.2.2. Solmaris Condominium Group Database Code Part 2



```python

    conn.commit()

    conn.close()

#Defining print function
def printTables():
    conn = sqlite3.connect('Solmaris.db')
    cursor = conn.cursor()

    cursor.execute('SELECT * FROM renter')
    rows = cursor.fetchall()

    print("Renter Table:")
    for row in rows:
        print(row)

    cursor.execute('SELECT * FROM property')
    rows = cursor.fetchall()

    print("Property Table:")
    for row in rows:
        print(row)

    cursor.execute('SELECT * FROM rentalAgreement')
    rows = cursor.fetchall()

    print("Rental Agreement Table:")
    for row in rows:
        print(row)

    conn.close()

makeDatabase()
printTables()
```

Figure 2.2.3. Solmaris Condominium Group Database Code Part 3



```
PS C:\Users\Mark\Downloads\chinook> & C:/Users/Mark/AppData/Local/Programs/Python/Python313/python.exe c:/Users/Mark/Desktop/CPE106-4/postlab2.py
Renter Table:
(1, 'Fname', 'Lname', 'X', 'December Ave.', 'Itchyworms', 'Solid', '911', '043-3333', '@gmail.com')
(2, 'Fname', 'Lname', 'X', 'December Ave.', 'Itchyworms', 'Liquid', '911', '043-3333', '@outlook.com')
(3, 'Fname', 'Lname', 'X', 'December Ave.', 'Itchyworms', 'Gas', '911', '043-3333', '@mymail.mapua.edu.ph')
(4, 'Fname', 'Lname', 'X', 'December Ave.', 'Itchyworms', 'Plasma', '911', '043-3333', '@dlsl.edu.ph')
Property Table:
(1, 'locName', 'December Ave.', 'Itchyworms', 'Solid', '420', '69', '69420sq ft', '1', '1', '3', 33.33)
(2, 'locName', 'December Ave.', 'Itchyworms', 'Liquid', '420', '69', '69420sq ft', '1', '1', '2', 33.33)
(3, 'locName', 'December Ave.', 'Itchyworms', 'Gas', '420', '69', '69420sq ft', '1', '1', '1', 33.33)
(4, 'locName', 'December Ave.', 'Itchyworms', 'Plasma', '420', '69', '69420sq ft', '1', '1', '0', 33.33)
Rental Agreement Table:
(1, 'Fname', 'Lname', 'X', 'December Ave.', 'Itchyworms', 'Solid', '911', '043-3333', '01/11/1111', '01/11/2222', 33.33, 44.3)
(2, 'Fname', 'Lname', 'X', 'December Ave.', 'Itchyworms', 'Liquid', '911', '043-3333', '01/11/1111', '01/11/2222', 33.33, 44.3)
(3, 'Fname', 'Lname', 'X', 'December Ave.', 'Itchyworms', 'Gas', '911', '043-3333', '01/11/1111', '01/11/2222', 33.33, 44.3)
(4, 'Fname', 'Lname', 'X', 'December Ave.', 'Itchyworms', 'Plasma', '911', '043-3333', '01/11/1111', '01/11/2222', 33.33, 44.3)
```

Figure 2.3. Solmaris Condominium Group Database Output

Figures 2.2.1, 2.2.2, and 2.2.3 present the code of the Solmaris Condominium Group Database. 2.2.1 and 2.2.2 present the database's declaration and data, whereas 2.2.3 presents the printing function for the data in the tables from the Solmaris Condominium Group Database.

3. Use SQLite commands to complete the following exercises.

a) Create a table named ADVENTURE_TRIP. The table has the same structure as the TRIP table shown in Figure 3-2 below except the TRIP_NAME column should use the VARCHAR data type and the DISTANCE and MAX_GRP_SIZE columns should use the NUMBER data type. Execute the command to describe the layout and characteristics of the ADVENTURE_TRIP table.

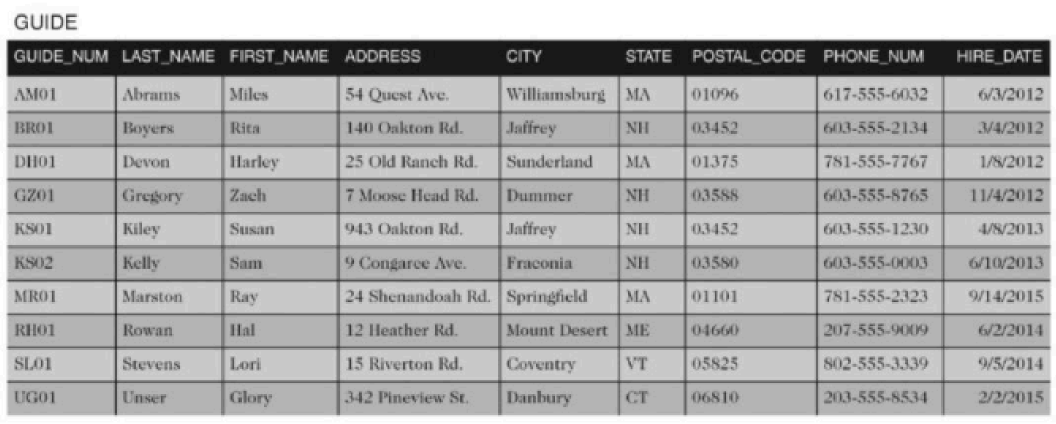
b) Add the following row to the ADVENTURE_TRIP table: trip ID: 45; trip name: Jay Peak; start location: Jay; state: VT; distance: 8; maximum group size: 8; type: Hiking and sea- son: Summer. Display the contents of the ADVENTURE_TRIP table.

c) Delete the ADVENTURE_TRIP table.

d) Open the script file (SQLServerColonial.sql) to create the six tables and add records to the tables. Revise the script file so that it can be run in the DB Browser.

e) Confirm that you have created the tables correctly by describing each table and comparing the results to the figures shown below. Confirm that you have added all data correctly by viewing the data in each table and comparing the results to Figures 1-4 through 1-8 shown below.

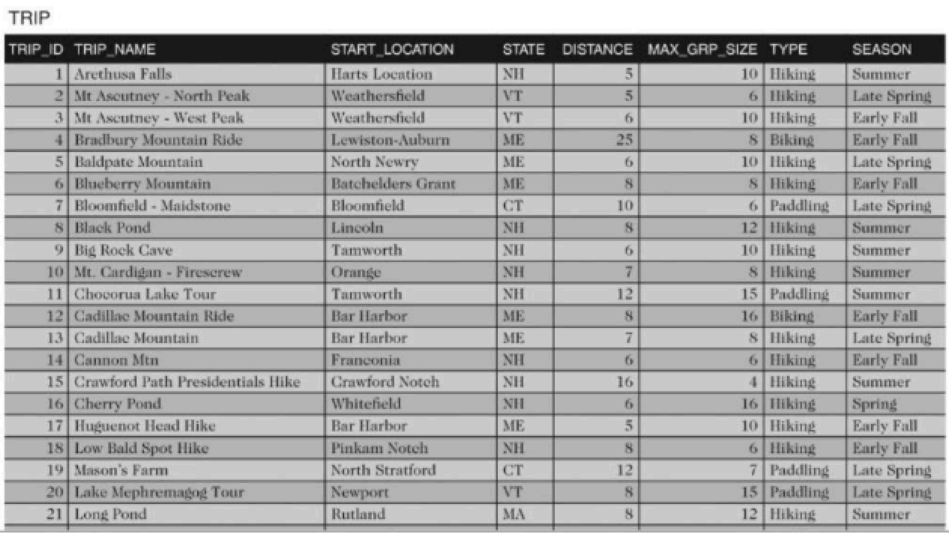
Figure 3-2. Colonial Adventure Tours Database GUIDE Table



GUIDE

GUIDE_NUM	LAST_NAME	FIRST_NAME	ADDRESS	CITY	STATE	POSTAL_CODE	PHONE_NUM	HIRE_DATE
AM01	Abrams	Miles	54 Quest Ave.	Williamsburg	MA	01096	617-555-6032	6/3/2012
BR01	Boyers	Rita	140 Oakton Rd.	Jaffrey	NH	03452	603-555-2134	3/4/2012
DH01	Devon	Harley	25 Old Ranch Rd.	Sunderland	MA	01375	781-555-7767	1/8/2012
GZ01	Gregory	Zach	7 Moose Head Rd.	Dummer	NH	03588	603-555-8765	11/4/2012
KS01	Kiley	Susan	943 Oakton Rd.	Jaffrey	NH	03452	603-555-1230	4/8/2013
KS02	Kelly	Sam	9 Congaree Ave.	Fraconia	NH	03580	603-555-0003	6/10/2013
MR01	Marston	Ray	24 Shenandoah Rd.	Springfield	MA	01101	781-555-2323	9/14/2015
RH01	Rowan	Hal	12 Heather Rd.	Mount Desert	ME	04660	207-555-9009	6/2/2014
SL01	Stevens	Lori	15 Riverton Rd.	Coventry	VT	05825	802-555-3339	9/5/2014
UG01	Unser	Glory	342 Pineview St.	Danbury	CT	06810	203-555-8534	2/2/2015

Figure 3-2. Colonial Adventure Tours Database TRIP Table (1)



TRIP

TRIP_ID	TRIP_NAME	START_LOCATION	STATE	DISTANCE	MAX_GRP_SIZE	TYPE	SEASON
1	Arethusa Falls	Harts Location	NH	5	10	Hiking	Summer
2	Mt Ascutney - North Peak	Weathersfield	VT	5	6	Hiking	Late Spring
3	Mt Ascutney - West Peak	Weathersfield	VT	6	10	Hiking	Early Fall
4	Bradbury Mountain Ride	Lewiston-Auburn	ME	25	8	Biking	Early Fall
5	Baldpate Mountain	North Newry	ME	6	10	Hiking	Late Spring
6	Blueberry Mountain	Batchelders Grant	ME	8	8	Hiking	Early Fall
7	Bloomfield - Maidstone	Bloomfield	CT	10	6	Paddling	Late Spring
8	Black Pond	Lincoln	NH	8	12	Hiking	Summer
9	Big Rock Cave	Tamworth	NH	6	10	Hiking	Summer
10	Mt. Cardigan - Firescrew	Orange	NH	7	8	Hiking	Summer
11	Chocorua Lake Tour	Tamworth	NH	12	15	Paddling	Summer
12	Cadillac Mountain Ride	Bar Harbor	ME	8	16	Biking	Early Fall
13	Cadillac Mountain	Bar Harbor	ME	7	8	Hiking	Late Spring
14	Cannon Mtn	Franconia	NH	6	6	Hiking	Early Fall
15	Crawford Path Presidentials Hike	Crawford Notch	NH	16	4	Hiking	Summer
16	Cherry Pond	Whitefield	NH	6	16	Hiking	Spring
17	Huguenot Head Hike	Bar Harbor	ME	5	10	Hiking	Early Fall
18	Low Bald Spot Hike	Pinkam Notch	NH	8	6	Hiking	Early Fall
19	Mason's Farm	North Stratford	CT	12	7	Paddling	Late Spring
20	Lake Mephremagog Tour	Newport	VT	8	15	Paddling	Late Spring
21	Long Pond	Rutland	MA	8	12	Hiking	Summer
22	Long Pond Tour	Greenville	ME	12	10	Paddling	Summer
23	Lower Pond Tour	Poland	ME	8	15	Paddling	Late Spring
24	Mt Adams	Randolph	NH	9	6	Hiking	Summer
25	Mount Battie Ride	Camden	ME	20	8	Biking	Early Fall
26	Mount Cardigan Hike	Cardigan	NH	4	16	Hiking	Late Fall
27	Mt. Chocorua	Albany	NH	6	10	Hiking	Spring
28	Mount Garfield Hike	Woodstock	NH	5	10	Hiking	Early Fall
29	Metacomet-Monadnock Trail Hike	Pelham	MA	10	12	Hiking	Late Spring
30	McLennan Reservation Hike	Tyringham	MA	6	16	Hiking	Summer
31	Missisquoi River - VT	Lowell	VT	12	10	Paddling	Summer
32	Northern Forest Canoe Trail	Stark	NH	15	10	Paddling	Summer
33	Park Loop Ride	Mount Desert Island	ME	27	8	Biking	Late Spring
34	Pontook Reservoir Tour	Dummer	NH	15	14	Paddling	Late Spring
35	Pisgah State Park Ride	Northborough	NH	12	10	Biking	Summer
36	Pondicherry Trail Ride	White Mountains	NH	15	16	Biking	Late Spring
37	Seal Beach Harbor	Bar Harbor	ME	5	16	Hiking	Early Spring
38	Sawyer River Ride	Mount Carrigain	NH	10	18	Biking	Early Fall
39	Welch and Dickey Mountains Hike	Thorton	NH	5	10	Hiking	Summer
40	Wachusett Mountain	Princeton	MA	8	8	Hiking	Early Spring
41	Westfield River Loop	Fort Fairfield	ME	20	10	Biking	Late Spring

Figure 3-2. Colonial Adventure Tours Database TRIP Table (2)

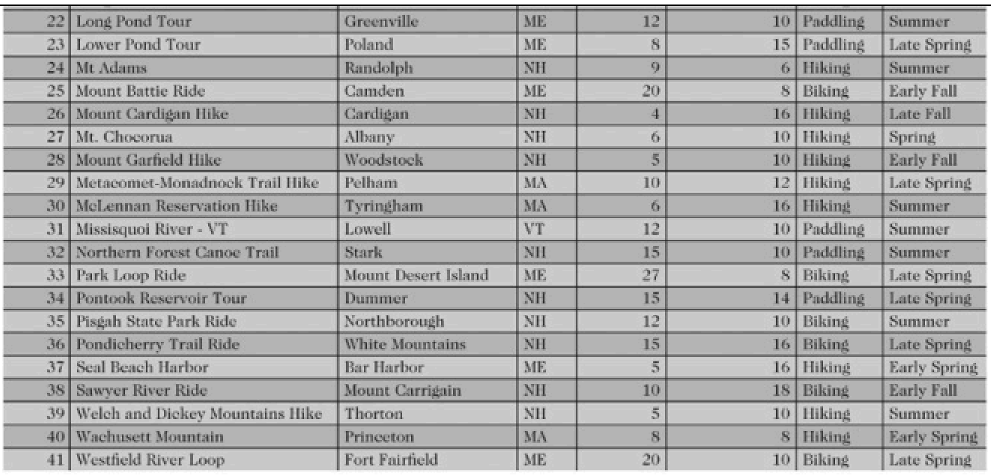
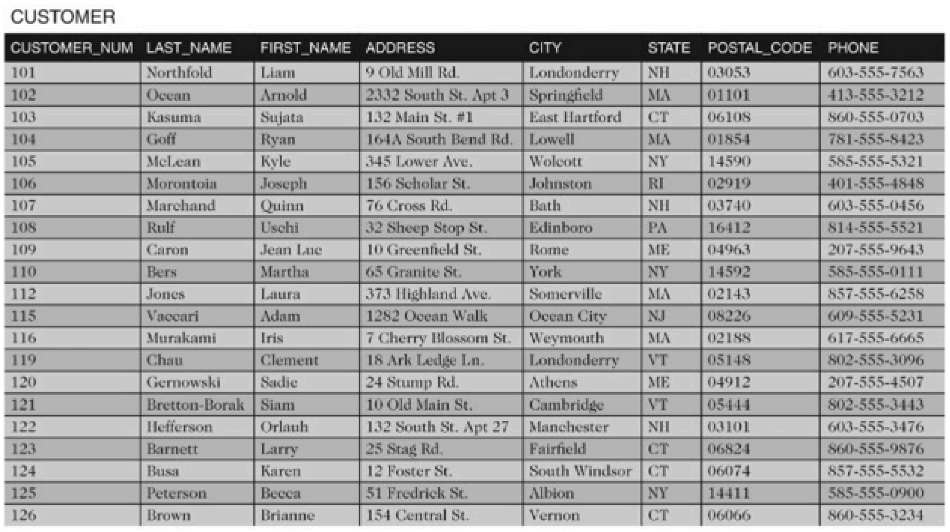


Figure 3-2. Colonial Adventure Tours Database CUSTOMER Table



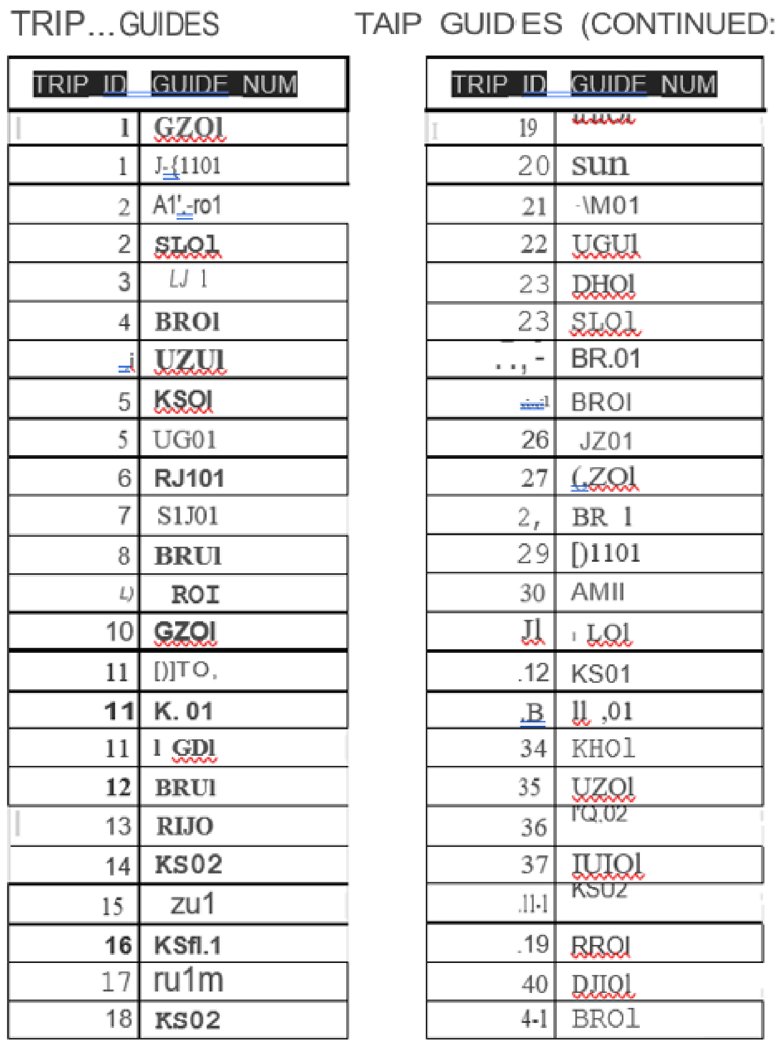
CUSTOMER

CUSTOMER_NUM	LAST_NAME	FIRST_NAME	ADDRESS	CITY	STATE	POSTAL_CODE	PHONE
101	Northfold	Liam	9 Old Mill Rd.	Londonderry	NH	03053	603-555-7563
102	Ocean	Arnold	2332 South St. Apt 3	Springfield	MA	01101	413-555-3212
103	Kasuma	Sujata	132 Main St. #1	East Hartford	CT	06108	860-555-0703
104	Goff	Ryan	164A South Bend Rd.	Lowell	MA	01854	781-555-8423
105	McLean	Kyle	345 Lower Ave.	Wolcott	NY	14590	585-555-5321
106	Morontoia	Joseph	156 Scholar St.	Johnston	RI	02919	401-555-4848
107	Marchand	Quinn	76 Cross Rd.	Bath	NH	03740	603-555-0456
108	Rulf	Uschi	32 Sheep Stop St.	Edinboro	PA	16412	814-555-5521
109	Caron	Jean Luc	10 Greenfield St.	Rome	ME	04963	207-555-9643
110	Bers	Martha	65 Granite St.	York	NY	14592	585-555-0111
112	Jones	Laura	373 Highland Ave.	Somerville	MA	02143	857-555-6258
115	Vaccari	Adam	1282 Ocean Walk	Ocean City	NJ	08226	609-555-5231
116	Murakami	Iris	7 Cherry Blossom St.	Weymouth	MA	02188	617-555-6665
119	Chau	Clement	18 Ark Ledge Ln.	Londonderry	VT	05148	802-555-3096
120	Gernowski	Sadie	24 Stump Rd.	Athens	ME	04912	207-555-4507
121	Bretton-Borak	Siam	10 Old Main St.	Cambridge	VT	05444	802-555-3443
122	Hefferson	Orlauh	132 South St. Apt 27	Manchester	NH	03101	603-555-3476
123	Barnett	Larry	25 Stag Rd.	Fairfield	CT	06824	860-555-9876
124	Busa	Karen	12 Foster St.	South Windsor	CT	06074	857-555-5532
125	Peterson	Becca	51 Fredrick St.	Albion	NY	14411	585-555-0900
126	Brown	Brianne	154 Central St.	Vernon	CT	06066	860-555-3234

Figure 3-2. Colonial Adventure Tours Database RESERVATION Table (1)

RESERVATION

RESERVATION_ID	TRIP_ID	TRIP_DATE	NUM_PERSONS	TRIP_PRICE	OTHER_FEES	CUSTOMER_NUM
1600001	-10	J/26/2016	2	$55.00	$0.00	JJJI
16Q()002	21	6/1'/2016	2	$')5.00	$0.IJO	11)1
16Q()00.1	2R	9112/2016		$]500	$000	rn. 1
160!)00-I	26	10/lt./2016	-l	$-15.00	$15.00	10-1
1600005	=f)	(/25/2016	5	$55.00	$0.0o	105
[(.0000<.	J2	(/18/2016	1	$80.00	$20.IJO	1()(
160000i	22	i/9/2016	8	$iS.0o	$10,00	107
±(X)l)()S	2tl	W12/201h	2	$.'\5,00	$0,111)	11),
1600009	.18	9/11/2016	2	$90.00	$40.00	109
16±±)01()	2	5/H/2016	3	$r.oo	$0.011	102
[60()01]	±l	WJ5/20H,	J	$25,00	$0.00	W2
1600012	1	G/12/2016	4	$15,00	$0,00	115
16Q(10D	8	71'-)/2016	1	$20.00	$5,011	116



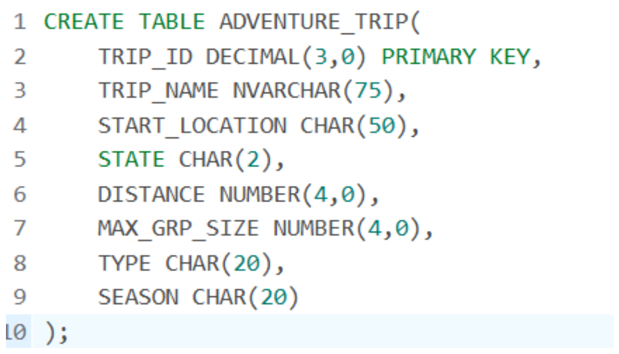
TRIP...GUIDES

TRIP_ID	GUIDE_NUM
1	GZOl
1	J-{1101
2	A1'-ro1
2	SLO1
3	LJ 1
4	BROI
-i	UZUl
5	KSOl
5	UG01
6	RJ101
7	S1J01
8	BRUl
L)	ROI
10	GZOl
11	D)TO,
11	K. 01
11	I GDl
12	BRUl
13	RIJO
14	KS02
15	zu1
16	KSfl.1
17	ru1m
18	KS02

TAIP GUID ES (CONTINUED:

TRIP_ID	GUIDE_NUM
19	[illegible]
20	sun
21	-\M01
22	UGUl
23	DHOl
23	SLO1
. . , -	BR.01
[illegible]	BROI
26	JZ01
27	(ZOl
2,	BR 1
29	[)1101
30	AMII
Jl	LOl
.12	KS01
.B	Jl ,01
34	KHO1
35	UZOl
36	rQ.02
37	IUIOl
.Il-l	KSU2
.19	RROl
40	DJIOl
4-1	BRO1

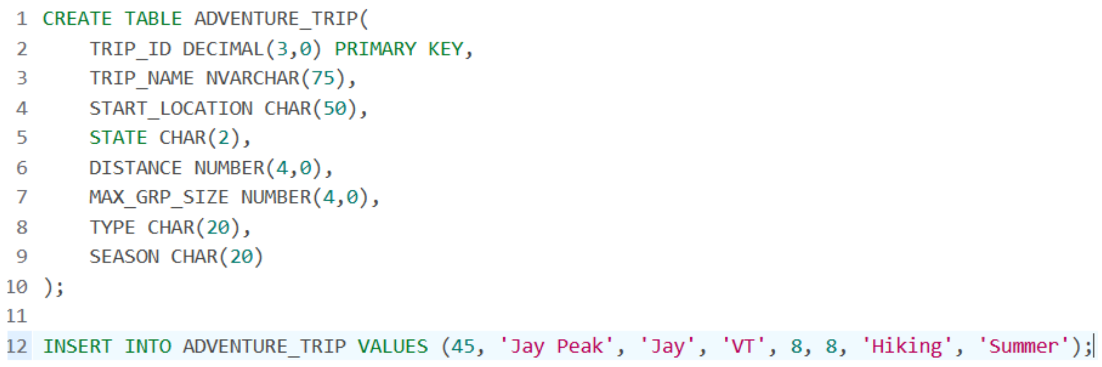
B. Debugging and Sample Run of Python program connection to your created SQLite database (with edited screengrabs and discussion)



```
CREATE TABLE ADVENTURE_TRIP(
    TRIP_ID DECIMAL(3,0) PRIMARY KEY,
    TRIP_NAME NVARCHAR(75),
    START_LOCATION CHAR(50),
    STATE CHAR(2),
    DISTANCE NUMBER(4,0),
    MAX_GRP_SIZE NUMBER(4,0),
    TYPE CHAR(20),
    SEASON CHAR(20)
);
```

Figure 4.1. Creation of ADVENTURE_TRIP Entity and Changes in Attributes

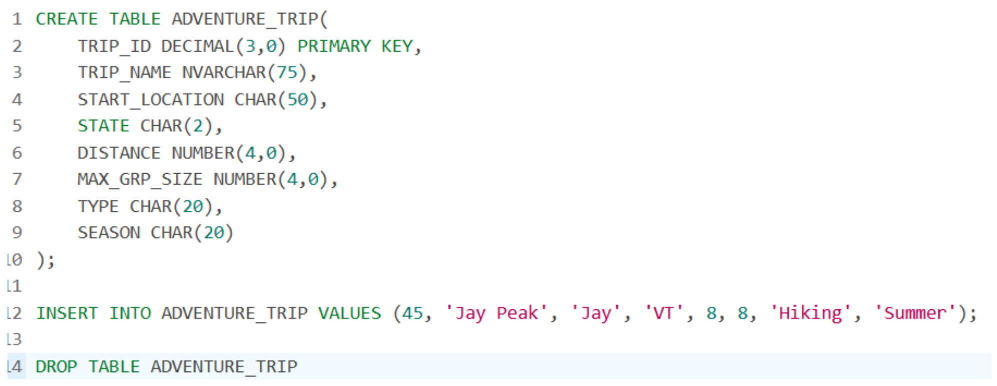
Figure 4.1 presents the creation of an entity named ADVENTURE_TRIP and its declaration of attributes similar to the TRIP table of Figure 3-2 but with altered data types of TRIP_NAME, DISTANCE, and MAX_GRP_SIZE.



```
CREATE TABLE ADVENTURE_TRIP(
    TRIP_ID DECIMAL(3,0) PRIMARY KEY,
    TRIP_NAME NVARCHAR(75),
    START_LOCATION CHAR(50),
    STATE CHAR(2),
    DISTANCE NUMBER(4,0),
    MAX_GRP_SIZE NUMBER(4,0),
    TYPE CHAR(20),
    SEASON CHAR(20)
);

INSERT INTO ADVENTURE_TRIP VALUES (45, 'Jay Peak', 'Jay', 'VT', 8, 8, 'Hiking', 'Summer');
```

Figure 4.2. Inserting a Value Into ADVENTURE_TRIP Entity

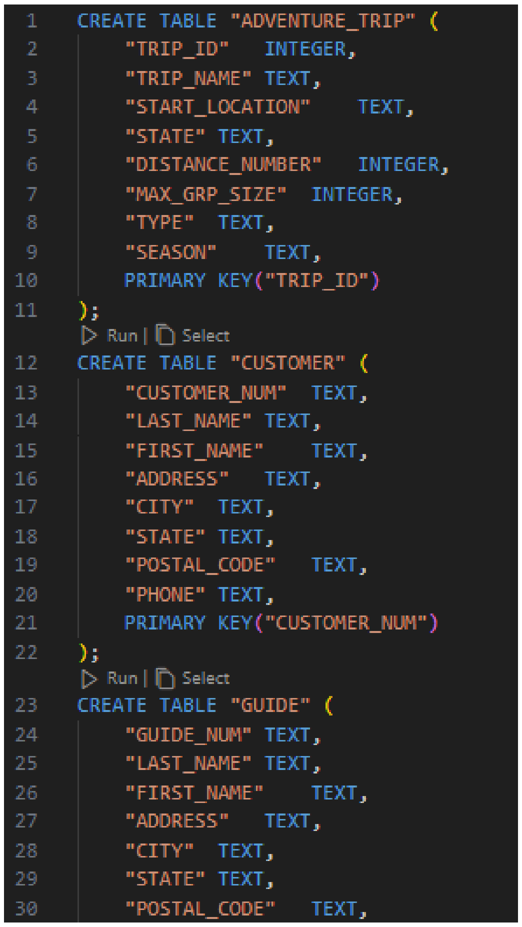
Figure 4.2 presents the addition of a line to add a value to the ADVENTURE_TRIP entity based on the provided values for each attribute.



```
CREATE TABLE ADVENTURE_TRIP(
    TRIP_ID DECIMAL(3,0) PRIMARY KEY,
    TRIP_NAME NVARCHAR(75),
    START_LOCATION CHAR(50),
    STATE CHAR(2),
    DISTANCE NUMBER(4,0),
    MAX_GRP_SIZE NUMBER(4,0),
    TYPE CHAR(20),
    SEASON CHAR(20)
);

INSERT INTO ADVENTURE_TRIP VALUES (45, 'Jay Peak', 'Jay', 'VT', 8, 8, 'Hiking', 'Summer');

DROP TABLE ADVENTURE_TRIP
```

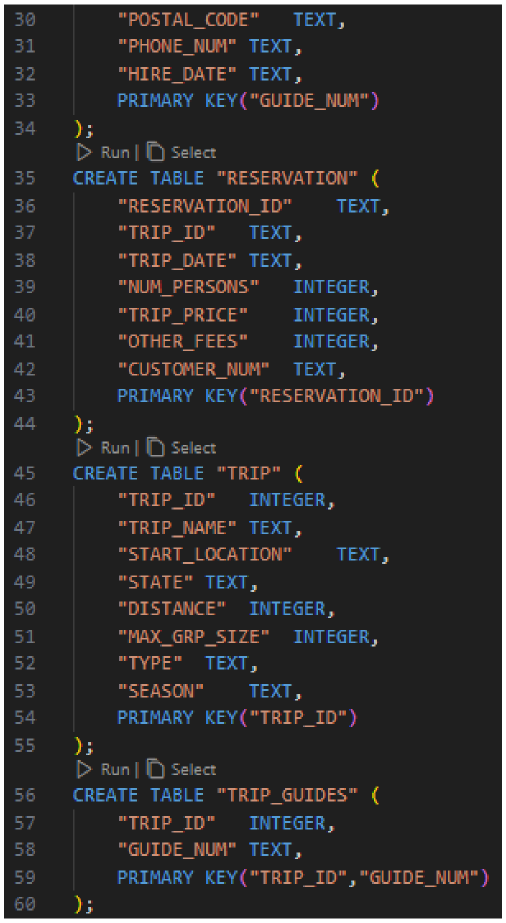
Figure 4.3. Adding a DROP TABLE ADVENTURE_TRIP Query

Figure 4.3 presents the addition of a DROP TABLE query to delete the ADVENTURE_TRIP entity.



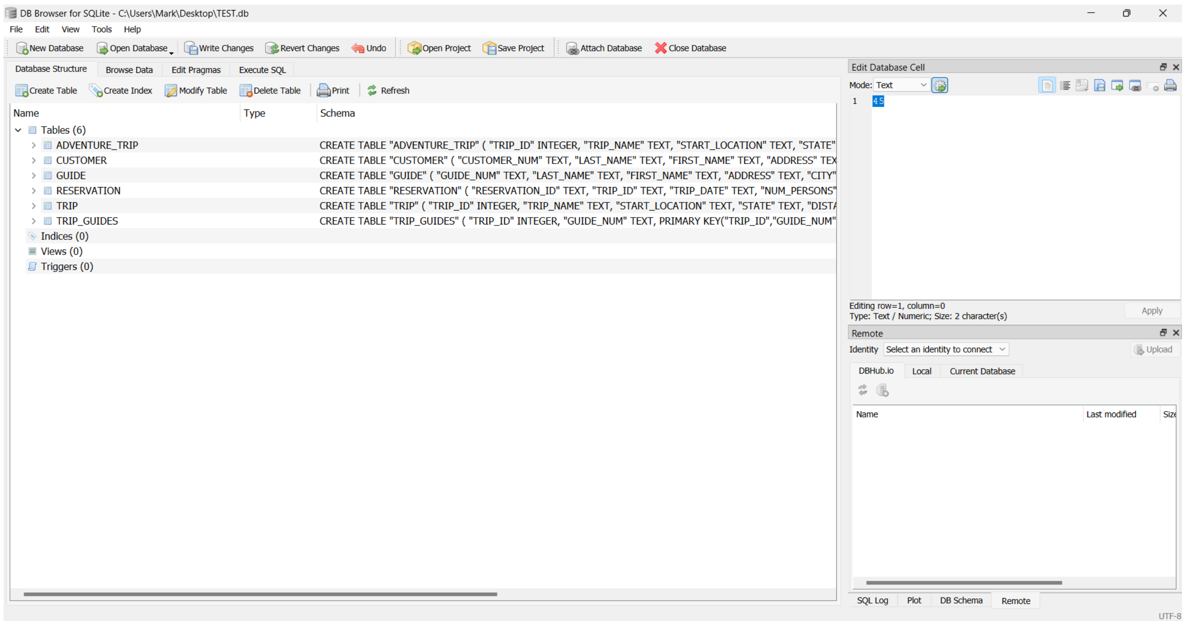
```
CREATE TABLE "ADVENTURE_TRIP" (
    "TRIP_ID"   INTEGER,
    "TRIP_NAME" TEXT,
    "START_LOCATION"    TEXT,
    "STATE" TEXT,
    "DISTANCE_NUMBER"   INTEGER,
    "MAX_GRP_SIZE"  INTEGER,
    "TYPE"  TEXT,
    "SEASON"    TEXT,
    PRIMARY KEY("TRIP_ID")
);
▷ Run | Select
CREATE TABLE "CUSTOMER" (
    "CUSTOMER_NUM"  TEXT,
    "LAST_NAME" TEXT,
    "FIRST_NAME"    TEXT,
    "ADDRESS"   TEXT,
    "CITY"  TEXT,
    "STATE" TEXT,
    "POSTAL_CODE"   TEXT,
    "PHONE" TEXT,
    PRIMARY KEY("CUSTOMER_NUM")
);
▷ Run | Select
CREATE TABLE "GUIDE" (
    "GUIDE_NUM" TEXT,
    "LAST_NAME" TEXT,
    "FIRST_NAME"    TEXT,
    "ADDRESS"   TEXT,
    "CITY"  TEXT,
    "STATE" TEXT,
    "POSTAL_CODE"   TEXT,
```

Figure 4.4.1. Revised Table Declaration Part 1



```
    "POSTAL_CODE"   TEXT,
    "PHONE_NUM" TEXT,
    "HIRE_DATE" TEXT,
    PRIMARY KEY("GUIDE_NUM")
);
▷ Run | Select
CREATE TABLE "RESERVATION" (
    "RESERVATION_ID"    TEXT,
    "TRIP_ID"   TEXT,
    "TRIP_DATE" TEXT,
    "NUM_PERSONS"   INTEGER,
    "TRIP_PRICE"    INTEGER,
    "OTHER_FEES"    INTEGER,
    "CUSTOMER_NUM"  TEXT,
    PRIMARY KEY("RESERVATION_ID")
);
▷ Run | Select
CREATE TABLE "TRIP" (
    "TRIP_ID"   INTEGER,
    "TRIP_NAME" TEXT,
    "START_LOCATION"    TEXT,
    "STATE" TEXT,
    "DISTANCE"  INTEGER,
    "MAX_GRP_SIZE"  INTEGER,
    "TYPE"  TEXT,
    "SEASON"    TEXT,
    PRIMARY KEY("TRIP_ID")
);
▷ Run | Select
CREATE TABLE "TRIP_GUIDES" (
    "TRIP_ID"   INTEGER,
    "GUIDE_NUM" TEXT,
    PRIMARY KEY("TRIP_ID","GUIDE_NUM")
);
```

Figure 4.4.2. Revised Table Declaration Part 2



DB Browser for SQLite - C:\Users\Mark\Desktop\TEST.db

Name	Type	Schema
Tables (6)		
ADVENTURE_TRIP		CREATE TABLE "ADVENTURE_TRIP" ("TRIP_ID" INTEGER, "TRIP_NAME" TEXT, "START_LOCATION" TEXT, "STATE"
CUSTOMER		CREATE TABLE "CUSTOMER" ("CUSTOMER_NUM" TEXT, "LAST_NAME" TEXT, "FIRST_NAME" TEXT, "ADDRESS" TEX
GUIDE		CREATE TABLE "GUIDE" ("GUIDE_NUM" TEXT, "LAST_NAME" TEXT, "FIRST_NAME" TEXT, "ADDRESS" TEXT, "CITY"
RESERVATION		CREATE TABLE "RESERVATION" ("RESERVATION_ID" TEXT, "TRIP_ID" TEXT, "TRIP_DATE" TEXT, "NUM_PERSONS"
TRIP		CREATE TABLE "TRIP" ("TRIP_ID" INTEGER, "TRIP_NAME" TEXT, "START_LOCATION" TEXT, "STATE" TEXT, "DISTA
TRIP_GUIDES		CREATE TABLE "TRIP_GUIDES" ("TRIP_ID" INTEGER, "GUIDE_NUM" TEXT, PRIMARY KEY("TRIP_ID","GUIDE_NUM"
Indices (0)		
Views (0)		
Triggers (0)		

Edit Database Cell
Mode: Text
1 45
Editing row=1, column=0
Type: Text / Numeric; Size: 2 character(s)

Figure 4.5. DB Browser with Six Tables

Figures 4.4.1 and 4.4.2 present the revised codes to declare the six tables into the DB Browser. Figure 4.5 presents the UI of the DB Browser with the declared tables.

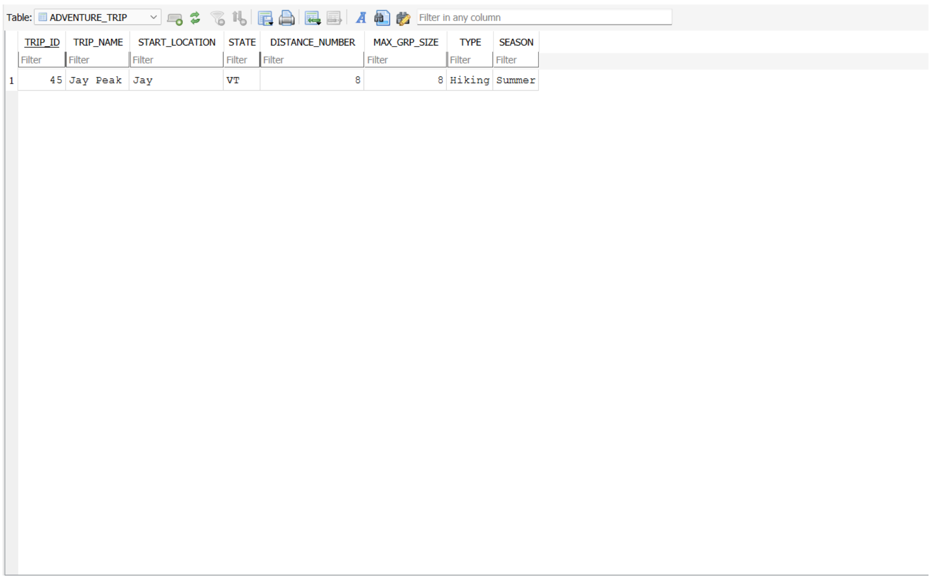


Table: ADVENTURE_TRIP

Filter in any column

	TRIP_ID	TRIP_NAME	START_LOCATION	STATE	DISTANCE_NUMBER	MAX_GRP_SIZE	TYPE	SEASON
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	45	Jay Peak	Jay	VT	8	8	Hiking	Summer

Figure 4.6. ADVENTURE_TRIP Table

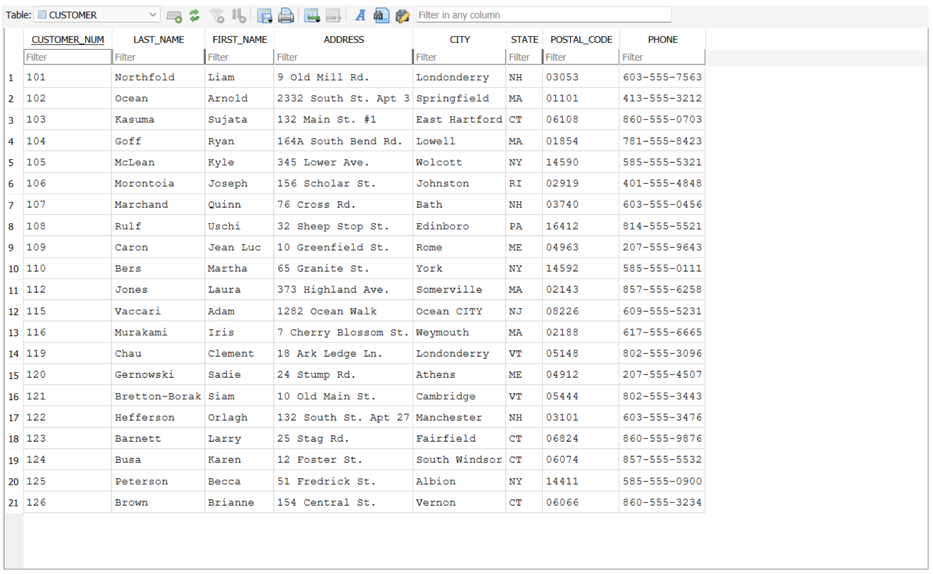


Table: CUSTOMER

Filter in any column

	CUSTOMER_NUM	LAST_NAME	FIRST_NAME	ADDRESS	CITY	STATE	POSTAL_CODE	PHONE
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	101	Northfold	Liam	9 Old Mill Rd.	Londonderry	NH	03053	603-555-7563
2	102	Ocean	Arnold	2332 South St. Apt 3	Springfield	MA	01101	413-555-3212
3	103	Kasuma	Sujata	132 Main St. #1	East Hartford	CT	06108	860-555-0703
4	104	Goff	Ryan	164A South Bend Rd.	Lowell	MA	01854	781-555-8423
5	105	McLean	Kyle	345 Lower Ave.	Wolcott	NY	14590	585-555-5321
6	106	Morontoia	Joseph	156 Scholar St.	Johnston	RI	02919	401-555-4848
7	107	Marchand	Quinn	76 Cross Rd.	Bath	NH	03740	603-555-0456
8	108	Rulf	Uschi	32 Sheep Stop St.	Edinboro	PA	16412	814-555-5521
9	109	Caron	Jean Luc	10 Greenfield St.	Rome	ME	04963	207-555-9643
10	110	Bers	Martha	65 Granite St.	York	NY	14592	585-555-0111
11	112	Jones	Laura	373 Highland Ave.	Somerville	MA	02143	857-555-6258
12	115	Vaccari	Adam	1282 Ocean Walk	Ocean CITY	NJ	08226	609-555-5231
13	116	Murakami	Iris	7 Cherry Blossom St.	Weymouth	MA	02188	617-555-6665
14	119	Chau	Clement	18 Ark Ledge Ln.	Londonderry	VT	05148	802-555-3096
15	120	Gernowski	Sadie	24 Stump Rd.	Athens	ME	04912	207-555-4507
16	121	Bretton-Borak	Siam	10 Old Main St.	Cambridge	VT	05444	802-555-3443
17	122	Hefferson	Orlagh	132 South St. Apt 27	Manchester	NH	03101	603-555-3476
18	123	Barnett	Larry	25 Stag Rd.	Fairfield	CT	06824	860-555-9876
19	124	Busa	Karen	12 Foster St.	South Windsor	CT	06074	857-555-5532
20	125	Peterson	Becca	51 Fredrick St.	Albion	NY	14411	585-555-0900
21	126	Brown	Brianne	154 Central St.	Vernon	CT	06066	860-555-3234

Figure 4.7. CUSTOMER Table

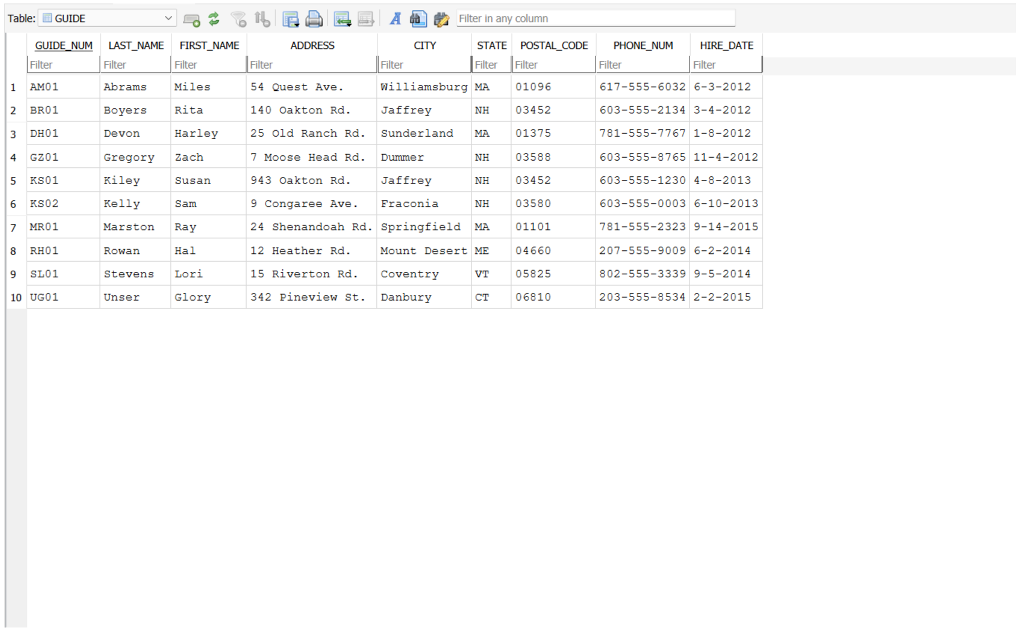


Table: GUIDE

	GUIDE_NUM	LAST_NAME	FIRST_NAME	ADDRESS	CITY	STATE	POSTAL_CODE	PHONE_NUM	HIRE_DATE
1	AM01	Abrams	Miles	54 Quest Ave.	Williamsburg	MA	01096	617-555-6032	6-3-2012
2	BR01	Boyers	Rita	140 Oakton Rd.	Jaffrey	NH	03452	603-555-2134	3-4-2012
3	DH01	Devon	Harley	25 Old Ranch Rd.	Sunderland	MA	01375	781-555-7767	1-8-2012
4	GZ01	Gregory	Zach	7 Moose Head Rd.	Dummer	NH	03588	603-555-8765	11-4-2012
5	KS01	Kiley	Susan	943 Oakton Rd.	Jaffrey	NH	03452	603-555-1230	4-8-2013
6	KS02	Kelly	Sam	9 Congaree Ave.	Fraconia	NH	03580	603-555-0003	6-10-2013
7	MR01	Marston	Ray	24 Shenandoah Rd.	Springfield	MA	01101	781-555-2323	9-14-2015
8	RH01	Rowan	Hal	12 Heather Rd.	Mount Desert	ME	04660	207-555-9009	6-2-2014
9	SL01	Stevens	Lori	15 Riverton Rd.	Coventry	VT	05825	802-555-3339	9-5-2014
10	UG01	Unser	Glory	342 Pineview St.	Danbury	CT	06810	203-555-8534	2-2-2015

Figure 4.7. GUIDE Table

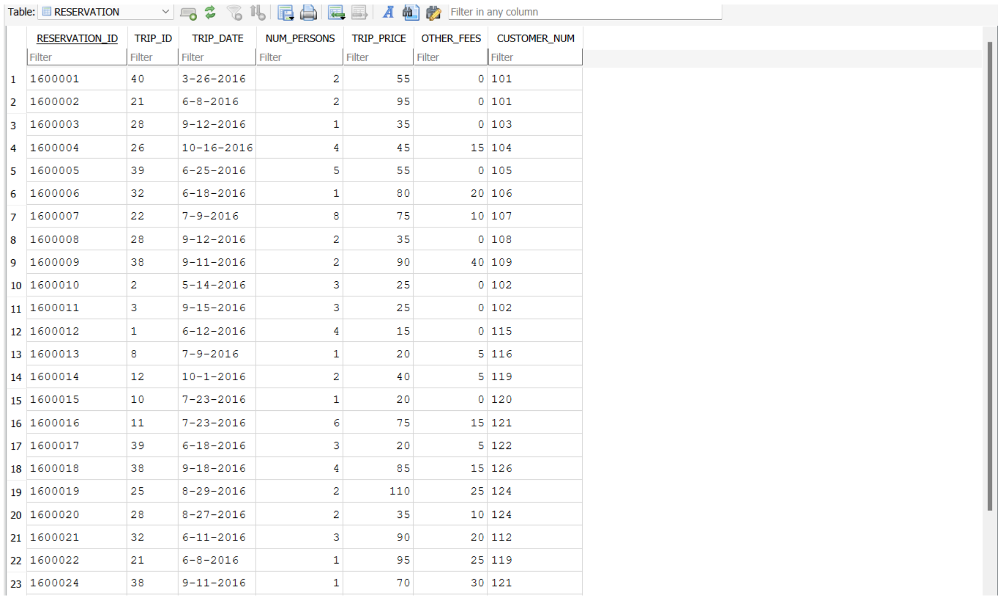
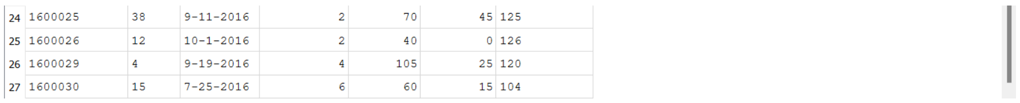


Table: RESERVATION

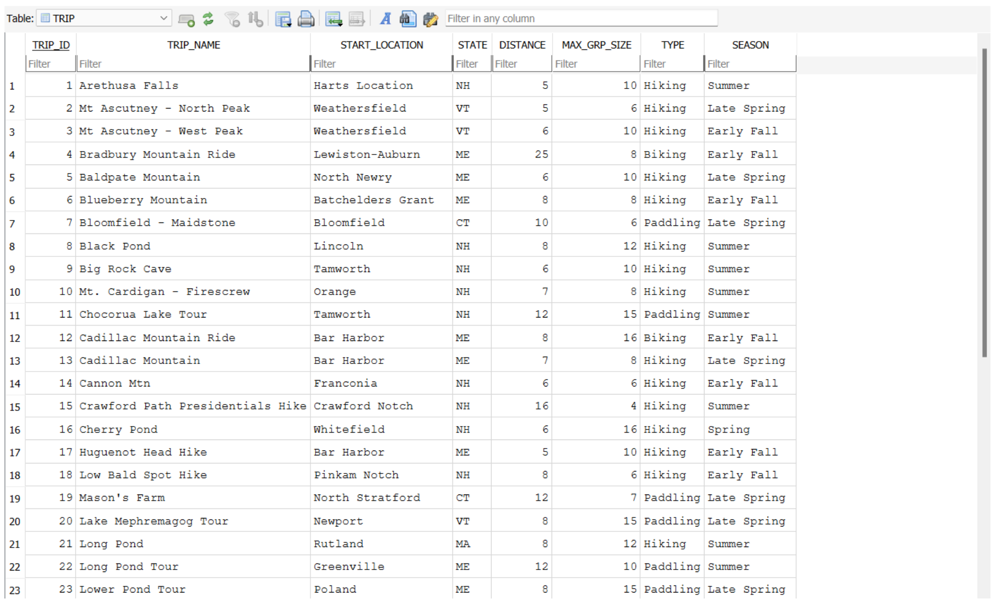
	RESERVATION_ID	TRIP_ID	TRIP_DATE	NUM_PERSONS	TRIP_PRICE	OTHER_FEES	CUSTOMER_NUM
1	1600001	40	3-26-2016	2	55	0	101
2	1600002	21	6-8-2016	2	95	0	101
3	1600003	28	9-12-2016	1	35	0	103
4	1600004	26	10-16-2016	4	45	15	104
5	1600005	39	6-25-2016	5	55	0	105
6	1600006	32	6-18-2016	1	80	20	106
7	1600007	22	7-9-2016	8	75	10	107
8	1600008	28	9-12-2016	2	35	0	108
9	1600009	38	9-11-2016	2	90	40	109
10	1600010	2	5-14-2016	3	25	0	102
11	1600011	3	9-15-2016	3	25	0	102
12	1600012	1	6-12-2016	4	15	0	115
13	1600013	8	7-9-2016	1	20	5	116
14	1600014	12	10-1-2016	2	40	5	119
15	1600015	10	7-23-2016	1	20	0	120
16	1600016	11	7-23-2016	6	75	15	121
17	1600017	39	6-18-2016	3	20	5	122
18	1600018	38	9-18-2016	4	85	15	126
19	1600019	25	8-29-2016	2	110	25	124
20	1600020	28	8-27-2016	2	35	10	124
21	1600021	32	6-11-2016	3	90	20	112
22	1600022	21	6-8-2016	1	95	25	119
23	1600024	38	9-11-2016	1	70	30	121

Figure 4.8.1. RESERVATION Table Part 1



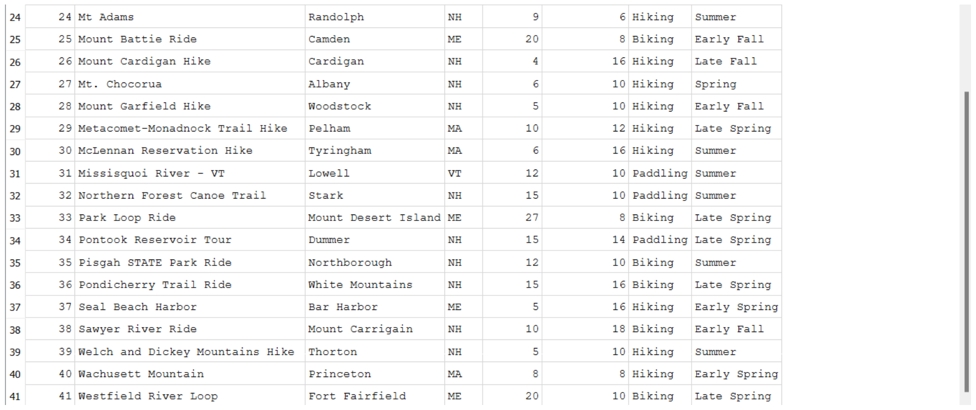
24	1600025	38	9-11-2016	2	70	45	125
25	1600026	12	10-1-2016	2	40	0	126
26	1600029	4	9-19-2016	4	105	25	120
27	1600030	15	7-25-2016	6	60	15	104

Figure 4.8.2. RESERVATION Table Part 2



	TRIP_ID	TRIP_NAME	START_LOCATION	STATE	DISTANCE	MAX_GRP_SIZE	TYPE	SEASON
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Arethusa Falls	Harts Location	NH	5	10	Hiking	Summer
2	2	Mt Ascutney - North Peak	Weathersfield	VT	5	6	Hiking	Late Spring
3	3	Mt Ascutney - West Peak	Weathersfield	VT	6	10	Hiking	Early Fall
4	4	Bradbury Mountain Ride	Lewiston-Auburn	ME	25	8	Biking	Early Fall
5	5	Baldpate Mountain	North Newry	ME	6	10	Hiking	Late Spring
6	6	Blueberry Mountain	Batchelders Grant	ME	8	8	Hiking	Early Fall
7	7	Bloomfield - Maidstone	Bloomfield	CT	10	6	Paddling	Late Spring
8	8	Black Pond	Lincoln	NH	8	12	Hiking	Summer
9	9	Big Rock Cave	Tamworth	NH	6	10	Hiking	Summer
10	10	Mt. Cardigan - Firescrew	Orange	NH	7	8	Hiking	Summer
11	11	Chocorua Lake Tour	Tamworth	NH	12	15	Paddling	Summer
12	12	Cadillac Mountain Ride	Bar Harbor	ME	8	16	Biking	Early Fall
13	13	Cadillac Mountain	Bar Harbor	ME	7	8	Hiking	Late Spring
14	14	Cannon Mtn	Franconia	NH	6	6	Hiking	Early Fall
15	15	Crawford Path Presidentials Hike	Crawford Notch	NH	16	4	Hiking	Summer
16	16	Cherry Pond	Whitefield	NH	6	16	Hiking	Spring
17	17	Huguenot Head Hike	Bar Harbor	ME	5	10	Hiking	Early Fall
18	18	Low Bald Spot Hike	Pinkam Notch	NH	8	6	Hiking	Early Fall
19	19	Mason's Farm	North Stratford	CT	12	7	Paddling	Late Spring
20	20	Lake Mephremagog Tour	Newport	VT	8	15	Paddling	Late Spring
21	21	Long Pond	Rutland	MA	8	12	Hiking	Summer
22	22	Long Pond Tour	Greenville	ME	12	10	Paddling	Summer
23	23	Lower Pond Tour	Poland	ME	8	15	Paddling	Late Spring

Figure 4.9.1. TRIP Table Part 1



24	24	Mt Adams	Randolph	NH	9	6	Hiking	Summer
25	25	Mount Battie Ride	Camden	ME	20	8	Biking	Early Fall
26	26	Mount Cardigan Hike	Cardigan	NH	4	16	Hiking	Late Fall
27	27	Mt. Chocorua	Albany	NH	6	10	Hiking	Spring
28	28	Mount Garfield Hike	Woodstock	NH	5	10	Hiking	Early Fall
29	29	Metacomet-Monadnock Trail Hike	Pelham	MA	10	12	Hiking	Late Spring
30	30	McLennan Reservation Hike	Tyringham	MA	6	16	Hiking	Summer
31	31	Missisquoi River - VT	Lowell	VT	12	10	Paddling	Summer
32	32	Northern Forest Canoe Trail	Stark	NH	15	10	Paddling	Summer
33	33	Park Loop Ride	Mount Desert Island	ME	27	8	Biking	Late Spring
34	34	Pontook Reservoir Tour	Dummer	NH	15	14	Paddling	Late Spring
35	35	Pisgah STATE Park Ride	Northborough	NH	12	10	Biking	Summer
36	36	Pondicherry Trail Ride	White Mountains	NH	15	16	Biking	Late Spring
37	37	Seal Beach Harbor	Bar Harbor	ME	5	16	Hiking	Early Spring
38	38	Sawyer River Ride	Mount Carrigain	NH	10	18	Biking	Early Fall
39	39	Welch and Dickey Mountains Hike	Thorton	NH	5	10	Hiking	Summer
40	40	Wachusett Mountain	Princeton	MA	8	8	Hiking	Early Spring
41	41	Westfield River Loop	Fort Fairfield	ME	20	10	Biking	Late Spring

Figure 4.9.2. TRIP Table Part 2

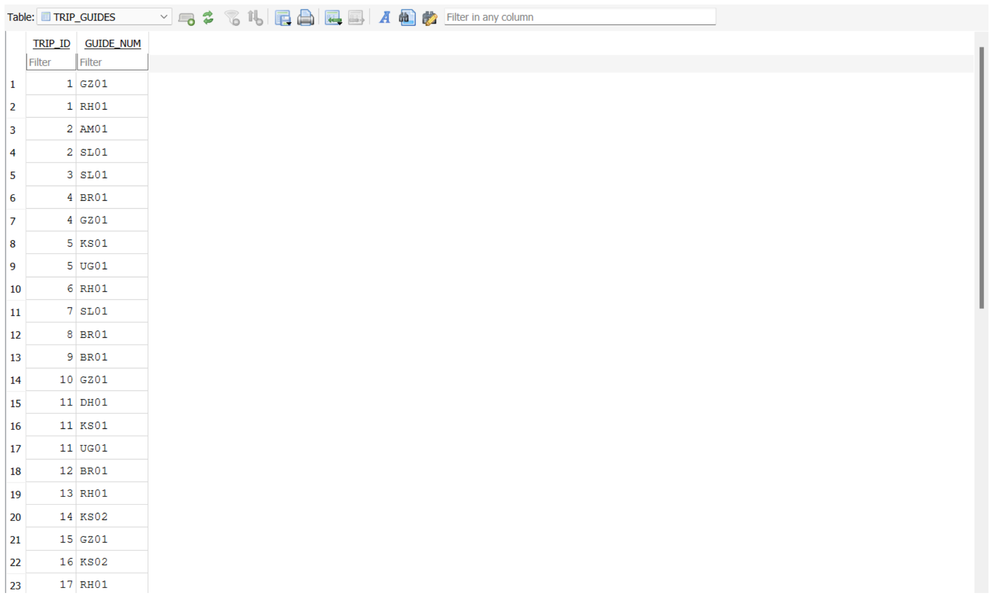
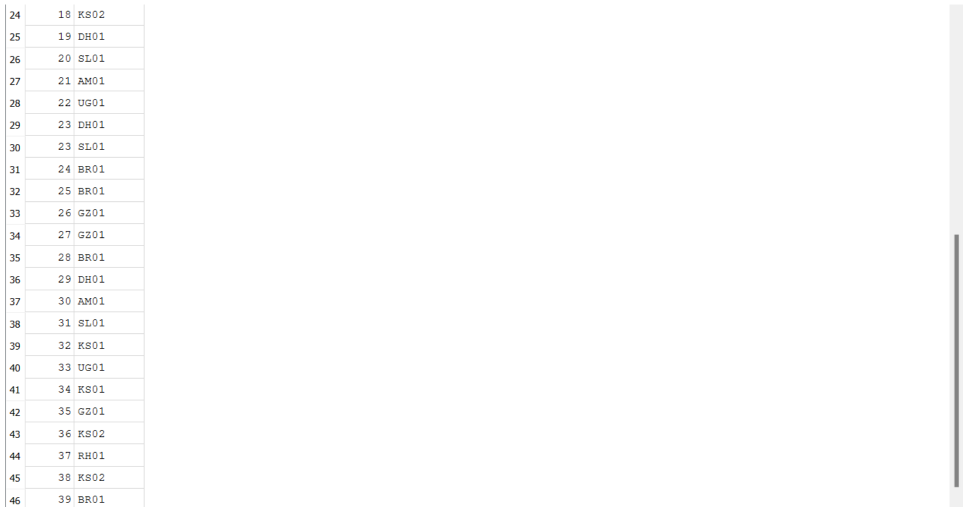


Table: TRIP_GUIDES

Filter in any column

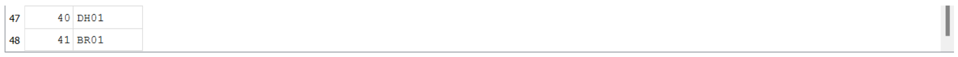
	TRIP_ID	GUIDE_NUM
	Filter	Filter
1	1	GZ01
2	1	RH01
3	2	AM01
4	2	SL01
5	3	SL01
6	4	BR01
7	4	GZ01
8	5	KS01
9	5	UG01
10	6	RH01
11	7	SL01
12	8	BR01
13	9	BR01
14	10	GZ01
15	11	DH01
16	11	KS01
17	11	UG01
18	12	BR01
19	13	RH01
20	14	KS02
21	15	GZ01
22	16	KS02
23	17	RH01

Figure 4.10.1. TRIP_GUIDES Table Part 1



24	18	KS02
25	19	DH01
26	20	SL01
27	21	AM01
28	22	UG01
29	23	DH01
30	23	SL01
31	24	BR01
32	25	BR01
33	26	GZ01
34	27	GZ01
35	28	BR01
36	29	DH01
37	30	AM01
38	31	SL01
39	32	KS01
40	33	UG01
41	34	KS01
42	35	GZ01
43	36	KS02
44	37	RH01
45	38	KS02
46	39	BR01

Figure 4.10.2. TRIP_GUIDES Table Part 2



47	40	DH01
48	41	BR01

Figure 4.10.3. TRIP_GUIDES Table Part 3

Figures 4.6, 4.7, 4.8.1, 4.8.2, 4.9.1, 4.9.2, 4.10.1, 4.10.2, and 4.10.3 present the tables and their contents after running the revised table declarations and the copied "INSERT INTO" queries from OracleColonial.sql. The tables presented in Figures 4.7, 4.8.1, 4.8.2, 4.9.1, 4.9.2, 4.10.1, 4.10.2, and 4.10.3 all show similar table and data outputs as those in Figure 3-2, namely the data in the GUIDE, TRIP, CUSTOMER, RESERVATION, and TRIP_GUIDES tables provided.