MAPÚA UNIVERSITY
SCHOOL OF ELECTRICAL, ELECTRONICS, AND COMPUTER ENGINEERING

Lab 4: Data Modeling and Database Systems

CPE106L (Software Design Laboratory)

Hannah Mae Antaran

Darrel Umali Virtusio

Kathleen Joy Tupas

Group: 01

Section: E01

PreLab

Readings, Insights, and Reflection

- A Guide to SQL. Philip J. Pratt; et al. 9781337668880

For this experiment, the group will focus mainly on databases. A database is a structure that contains different categories of information and the relationships between these categories. An example of a database is the TAL distributor database. The database contains information about sales representatives, customers, orders, and items. The next example of a database is the Colonial Adventure Tours database. It contains information such as the guide's last name, first name, address, city, state, postal code, telephone number, and hire date. And lastly, the Solmaris Condominium group database. It includes the information about the location name, address, city, state, and postal code. These categories involved in different databases are collected by the programmer so that data can be easily organized, managed, accessed, and updated.

The most common type of database is the relational database. Relational database is a collection of tables based on the relational model of data. Some concepts are important in databases such as entity, attribute, and relationship. Entity is like a noun; it can be a person, location, thing, or an event. The attribute is a property of an entity. And relationship is the association between those entities. To create a database design, a set of requirements is needed. First is to understand the given data and unique identifiers. Next step is to identify the attributes for all the entities. After that, identify the functional dependencies that exist among the attributes and use them to identify the tables by placing each attribute with the attribute on which it is functionally dependent. Last is to identify any relationships between the tables. After creating a database design, normalization is a must. Normalization is a process in which you identify the existence of potential problems, such as data duplication and redundancy, and implement ways to correct these problems.

- Core Python Programming. R. Nageswara Rao. 9789351198918

In python there are different types of databases. These are MySQL, Microsoft SQL Server, Sybase, Informix, Firebase, IBM DB2, Ingres, PostgreSQL, SAP DB, and Microsoft access. Before working with these databases, we should have picked the most suitable for your needs and install it in our computer system. In this experiment, we will focus on MySQL in python. To use MySQL, we need to install it first and follow these steps. First is to click start, click the MySQL folder, and open the MySQL Command Line Client. It will prompt you to the command line and can start entering SQL commands. Some SQL commands are show, create, use, describe, drop, and many more useful commands.

- Python Projects. Laura Cassell. 9781118908891 Chapter 3

A relational database is a database that access and store data in a structured format wherein its data are organized into rows known as records and columns that are called fields. The data of relational databases are based from their relations with one another. Example of relational databases are IBM DB2, Microsoft SQL Server, MySQL, and Oracle Database. The

Structured Query Language (SQL) is another concept related to relational databases wherein, through SQL, users can communicate and manipulate databases.

B. Answer to Questions

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

The DML, Data Manipulating Language, statements are used to manipulate contents of databases. Example commands are insert, delete, and update. On the other hand, DDL or Data Definition Language deals with creation and alteration of a database where the structure of the database is considered. Example commands of DDL are alter, create, drop, and rename.

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

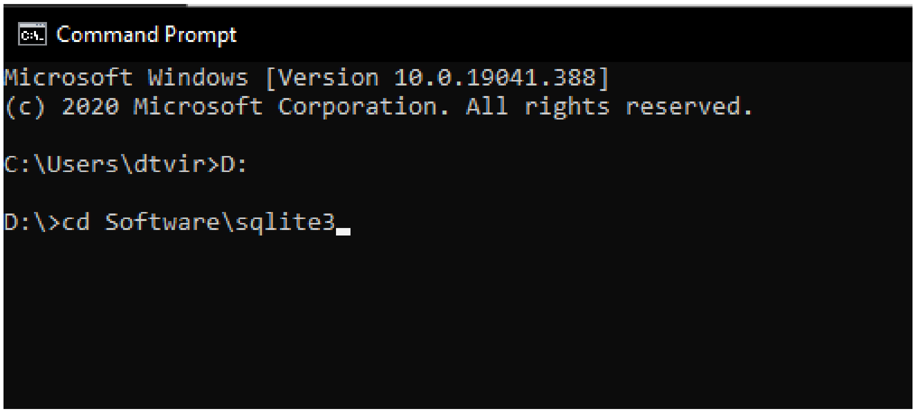
The categories of SQLite Functions are string functions, numeric functions, control flow functions, and date or time functions. Examples of string functions are length, trim, and lower. For numeric functions, examples of this are abs, random, and round. For control flow functions, examples are iif, ifnull, and nullif. For date or time functions, examples are date, time, and datetime.

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

In the Linux terminal, type "sqlite3" and the SQLite version will appear in the terminal if SQLite is installed.

InLab

- ## Objectives
  1. To familiarize ourselves about database
  2. To understand the importance of database in handling data
  3. To learn the different types of database run in python
  4. To demonstrate different SQLite commands
- ## Tools Used
  1. SQLite
- ## Procedure



Command Prompt

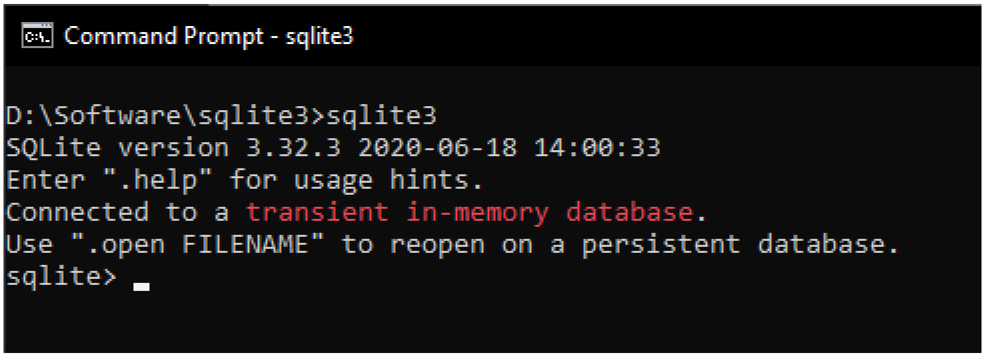
```
Microsoft Windows [Version 10.0.19041.388]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\dtvir>D:

D:\>cd Software\sqlite3
```

Figure 1. Navigating to the SQLite

First step in running SQLite is to navigate to the folder where the sqlite3.exe file is located. This is important so that there are no errors occurred if you run SQLite and enter commands.

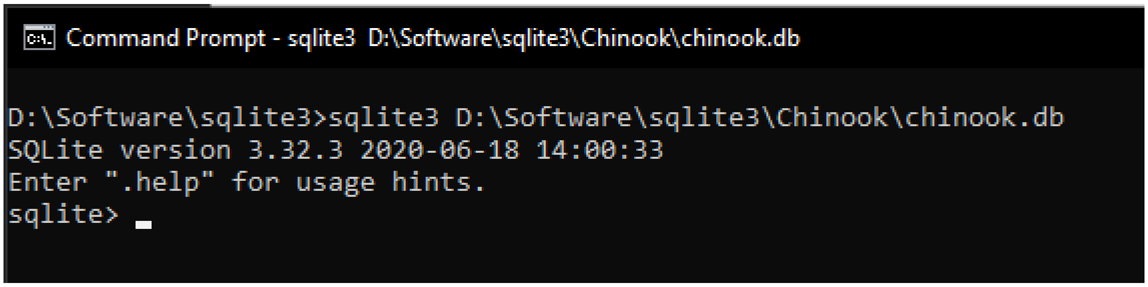


Command Prompt - sqlite3

```
D:\Software\sqlite3>sqlite3
SQLite version 3.32.3 2020-06-18 14:00:33
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

Figure 2. Running SQLite

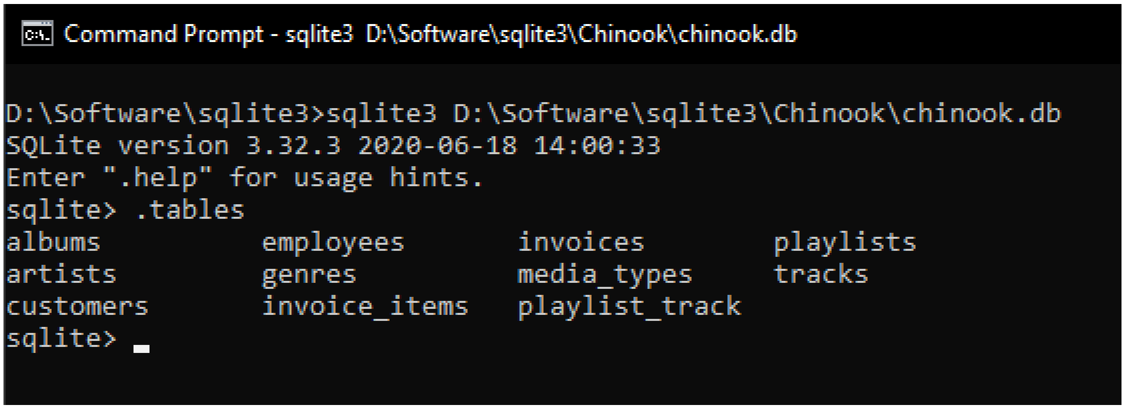
After navigating to the SQLite directory, we use the command sqlite3 in the command line to run the SQLite in our computer system. There are other ways to run SQLite, but this is the fastest and the most convenient way.


Command Prompt - sqlite3 D:\Software\sqlite3\Chinook\chinook.db

```
D:\Software\sqlite3>sqlite3 D:\Software\sqlite3\Chinook\chinook.db
SQLite version 3.32.3 2020-06-18 14:00:33
Enter ".help" for usage hints.
sqlite>
```

Figure 3. Running the Example Database using SQLite

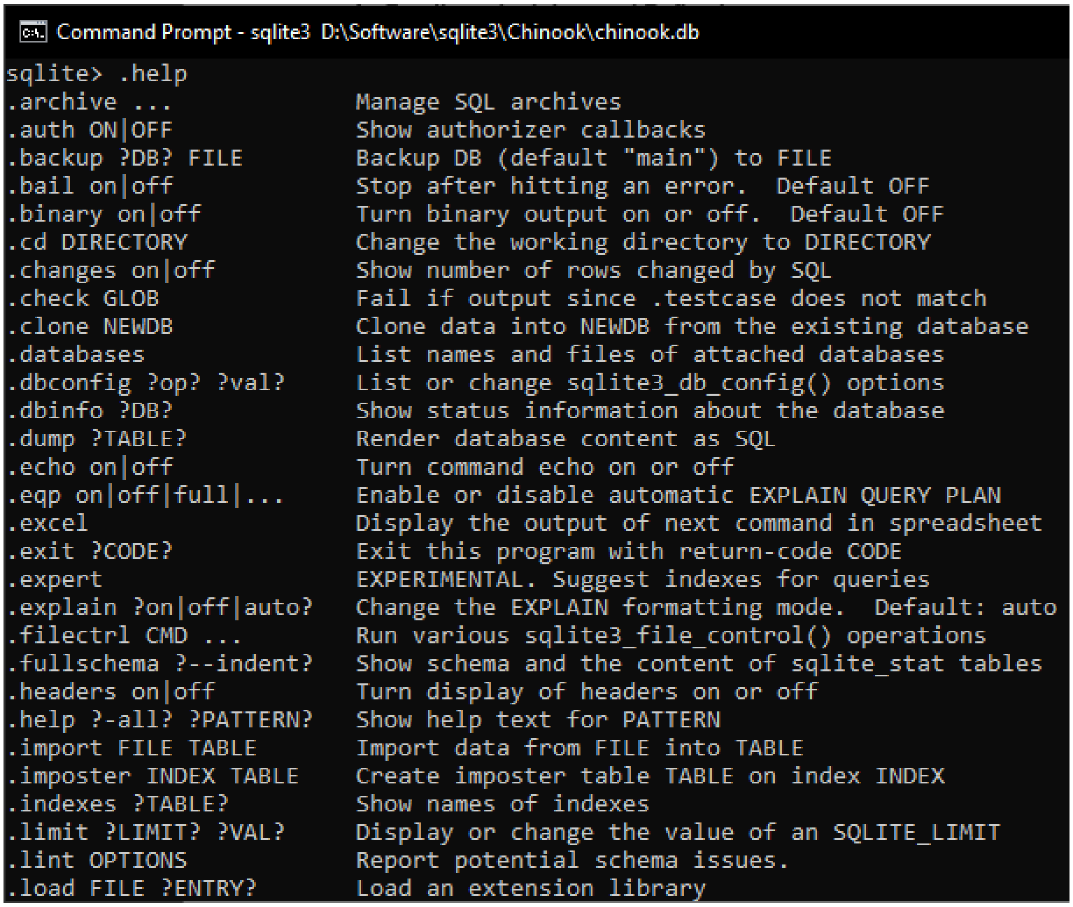
For this experiment, we will run the sample chinook database using the command shown in the figure above. If there are no errors, it will prompt you directly to SQLite and should show "sqlite>"


Command Prompt - sqlite3 D:\Software\sqlite3\Chinook\chinook.db

```
D:\Software\sqlite3>sqlite3 D:\Software\sqlite3\Chinook\chinook.db
SQLite version 3.32.3 2020-06-18 14:00:33
Enter ".help" for usage hints.
sqlite> .tables
albums          employees       invoices        playlists
artists         genres          media_types     tracks
customers       invoice_items   playlist_track
sqlite>
```

Figure 4. .tables command in SQLite

In this figure above, .tables is used. This command views all the tables available in the database. It consists of 11 tables like albums, artists, customers, employees, genres, invoice_items, invoices, media_types, playlist_track, playlists, and tracks.

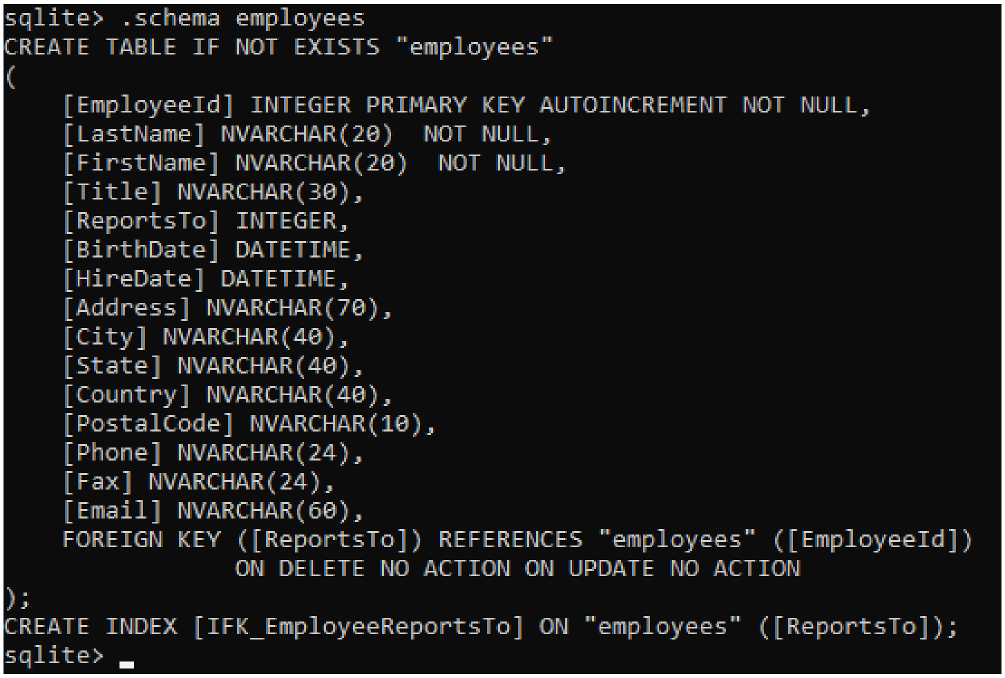


Command Prompt - sqlite3 D:\Software\sqlite3\Chinook\chinook.db

```
sqlite> .help
.archive ...             Manage SQL archives
.auth ON|OFF             Show authorizer callbacks
.backup ?DB? FILE        Backup DB (default "main") to FILE
.bail on|off             Stop after hitting an error.  Default OFF
.binary on|off           Turn binary output on or off.  Default OFF
.cd DIRECTORY            Change the working directory to DIRECTORY
.changes on|off          Show number of rows changed by SQL
.check GLOB              Fail if output since .testcase does not match
.clone NEWDB             Clone data into NEWDB from the existing database
.databases               List names and files of attached databases
.dbconfig ?op? ?val?     List or change sqlite3_db_config() options
.dbinfo ?DB?             Show status information about the database
.dump ?TABLE?            Render database content as SQL
.echo on|off             Turn command echo on or off
.eqp on|off|full|...     Enable or disable automatic EXPLAIN QUERY PLAN
.excel                   Display the output of next command in spreadsheet
.exit ?CODE?             Exit this program with return-code CODE
.expert                  EXPERIMENTAL. Suggest indexes for queries
.explain ?on|off|auto?   Change the EXPLAIN formatting mode.  Default: auto
.filectrl CMD ...        Run various sqlite3_file_control() operations
.fullschema ?--indent?   Show schema and the content of sqlite_stat tables
.headers on|off          Turn display of headers on or off
.help ?-all? ?PATTERN?   Show help text for PATTERN
.import FILE TABLE       Import data from FILE into TABLE
.imposter INDEX TABLE    Create imposter table TABLE on index INDEX
.indexes ?TABLE?         Show names of indexes
.limit ?LIMIT? ?VAL?     Display or change the value of an SQLITE_LIMIT
.lint OPTIONS            Report potential schema issues.
.load FILE ?ENTRY?       Load an extension library
```

Figure 5. .help command in SQLite

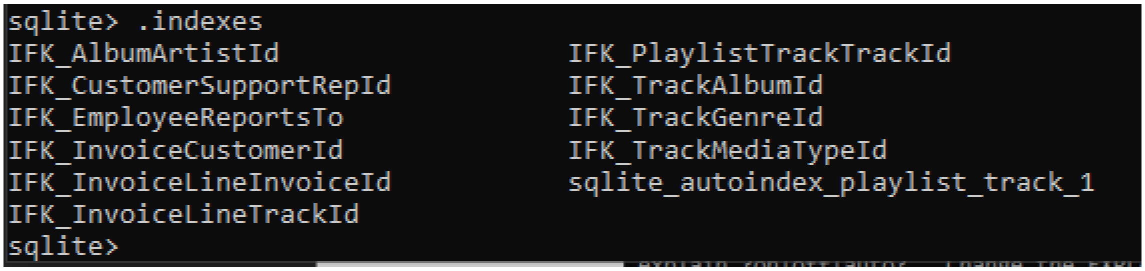
The command .help was entered in this part. This command shows all of the available commands that can be used by the user in SQLite. If the user can’t remember a specific command or the usage of the command, .help can be used to help with those problems.



```
sqlite> .schema employees
CREATE TABLE IF NOT EXISTS "employees"
(
    [EmployeeId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    [LastName] NVARCHAR(20)  NOT NULL,
    [FirstName] NVARCHAR(20)  NOT NULL,
    [Title] NVARCHAR(30),
    [ReportsTo] INTEGER,
    [BirthDate] DATETIME,
    [HireDate] DATETIME,
    [Address] NVARCHAR(70),
    [City] NVARCHAR(40),
    [State] NVARCHAR(40),
    [Country] NVARCHAR(40),
    [PostalCode] NVARCHAR(10),
    [Phone] NVARCHAR(24),
    [Fax] NVARCHAR(24),
    [Email] NVARCHAR(60),
    FOREIGN KEY ([ReportsTo]) REFERENCES "employees" ([EmployeeId])
                ON DELETE NO ACTION ON UPDATE NO ACTION
);
CREATE INDEX [IFK_EmployeeReportsTo] ON "employees" ([ReportsTo]);
sqlite> _
```

Figure 6. .schema Command in SQLite

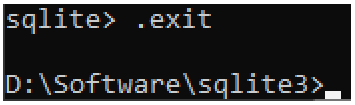
This command is used if you want to display the structure of the table. In this example, we used the employees table in our sample database and displayed its structure using .schema employees.



```
sqlite> .indexes
IFK_AlbumArtistId                   IFK_PlaylistTrackTrackId
IFK_CustomerSupportRepId            IFK_TrackAlbumId
IFK_EmployeeReportsTo               IFK_TrackGenreId
IFK_InvoiceCustomerId               IFK_TrackMediaTypeId
IFK_InvoiceLineInvoiceId            sqlite_autoindex_playlist_track_1
IFK_InvoiceLineTrackId
sqlite>
```

Figure 7. .indexes Command in SQLite

To show all of the indexes in the database, you use the .indexes command. If you want to show the indexes of a specific table, the command must be .indexes <name of table>.



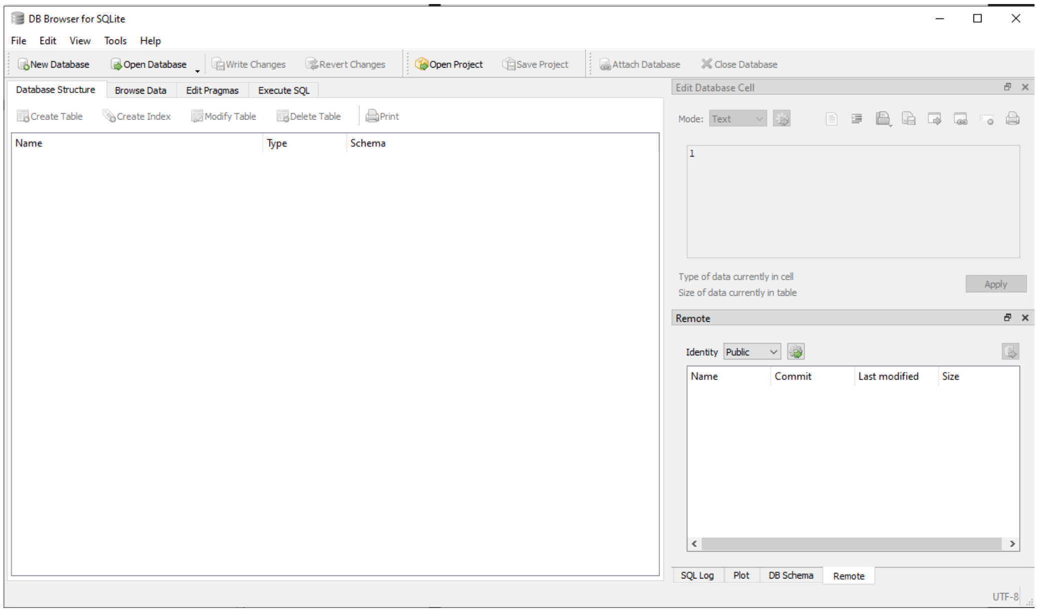
```
sqlite> .exit

D:\Software\sqlite3>_
```

Figure 8. .exit Command in SQLite

If the user is done using the sqlite3 tool, .exit is the command to quit and go back to the regular command line.

- DB Browser

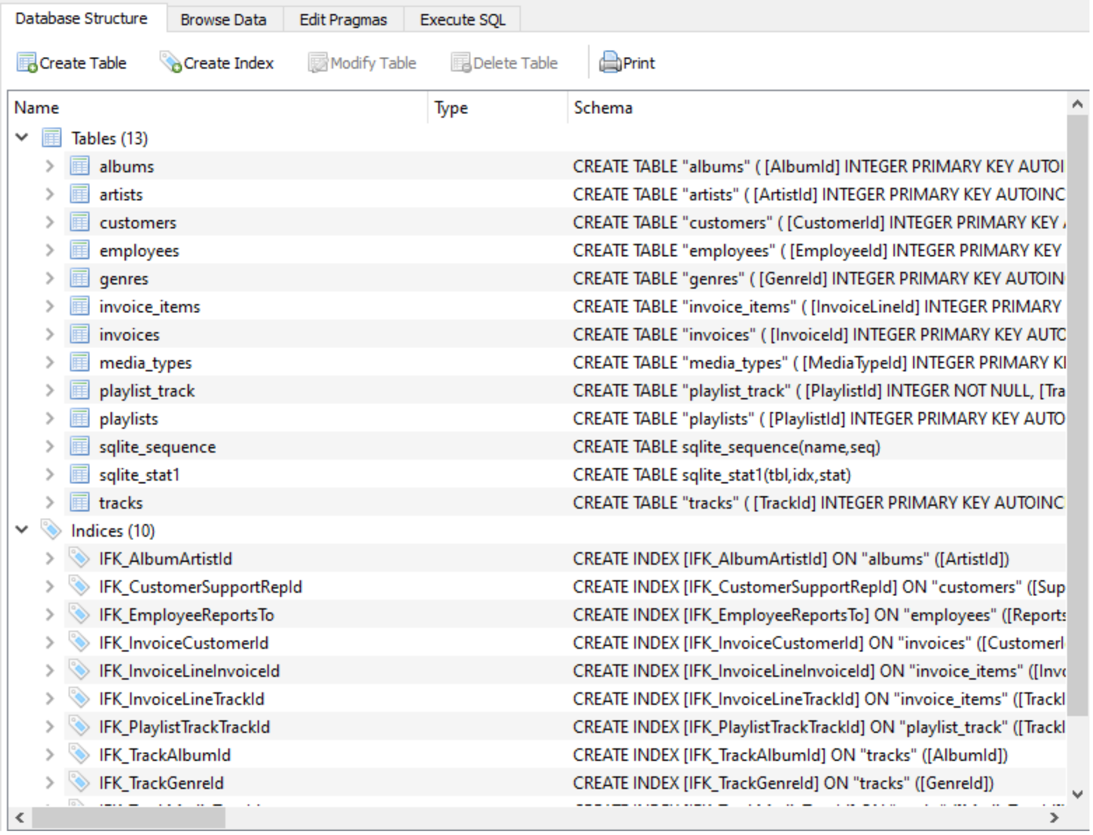


Mode: Text

Identity Public

Figure 1. User Interface of the DB Browser

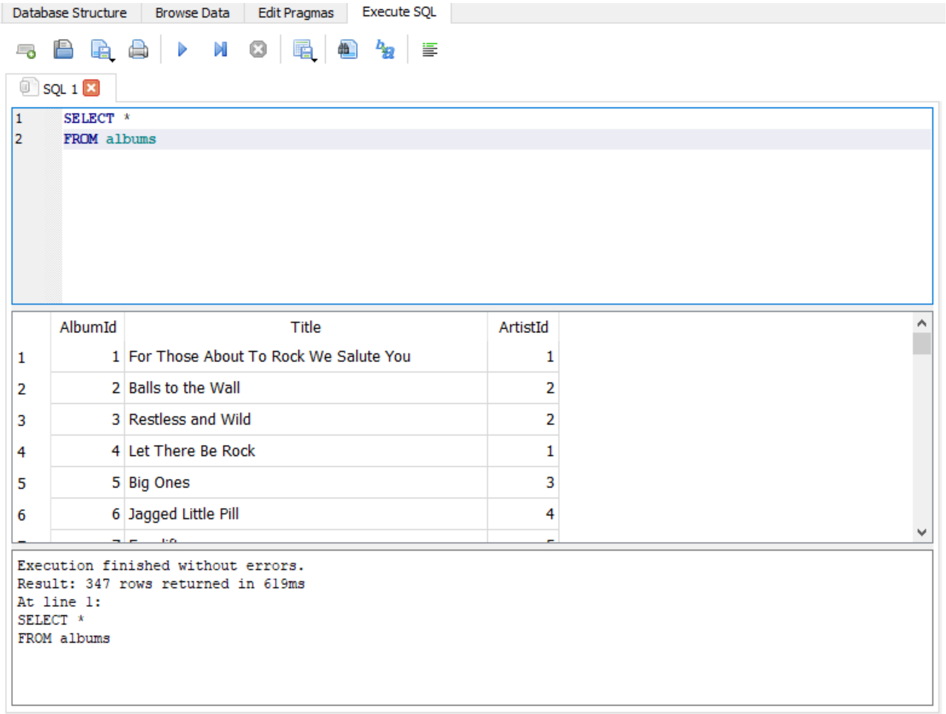
This is the initial screen of DB browser. On top is the menu bar and below it is another toolbar with four options: New Database, Open Database, Write Changes and Revert Changes. Below the toolbar is a 4-tabbed pane for; Database Structure, Browse Data, Edit Pragmas and Execute SQL. Initially these will be quite empty as we haven't created or opened a database yet. In general, we will see how each of these are used as we go through the lesson with the exception of the Edit Pragmas tab which deals with system wide parameters which we won't want to change. On the right-hand side there are two further panes, at the top is the Edit Database Cell pane which is grayed out. Below it is a 3-tabbed pane for DB Schema, SQL log and Remote.



Name	Type	Schema
Tables (13)		
albums		CREATE TABLE "albums" ([AlbumId] INTEGER PRIMARY KEY AUTOI
artists		CREATE TABLE "artists" ([ArtistId] INTEGER PRIMARY KEY AUTOINC
customers		CREATE TABLE "customers" ([CustomerId] INTEGER PRIMARY KEY
employees		CREATE TABLE "employees" ([EmployeeId] INTEGER PRIMARY KEY
genres		CREATE TABLE "genres" ([GenreId] INTEGER PRIMARY KEY AUTOIN
invoice_items		CREATE TABLE "invoice_items" ([InvoiceLineId] INTEGER PRIMARY
invoices		CREATE TABLE "invoices" ([InvoiceId] INTEGER PRIMARY KEY AUTC
media_types		CREATE TABLE "media_types" ([MediaTypeId] INTEGER PRIMARY KI
playlist_track		CREATE TABLE "playlist_track" ([PlaylistId] INTEGER NOT NULL, [Tra
playlists		CREATE TABLE "playlists" ([PlaylistId] INTEGER PRIMARY KEY AUTO
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
sqlite_stat1		CREATE TABLE sqlite_stat1(tbl,idx,stat)
tracks		CREATE TABLE "tracks" ([TrackId] INTEGER PRIMARY KEY AUTOINC
Indices (10)		
IFK_AlbumArtistId		CREATE INDEX [IFK_AlbumArtistId] ON "albums" ([ArtistId])
IFK_CustomerSupportRepId		CREATE INDEX [IFK_CustomerSupportRepId] ON "customers" ([Sup
IFK_EmployeeReportsTo		CREATE INDEX [IFK_EmployeeReportsTo] ON "employees" ([Reports
IFK_InvoiceCustomerId		CREATE INDEX [IFK_InvoiceCustomerId] ON "invoices" ([CustomerI
IFK_InvoiceLineInvoiceId		CREATE INDEX [IFK_InvoiceLineInvoiceId] ON "invoice_items" ([Invo
IFK_InvoiceLineTrackId		CREATE INDEX [IFK_InvoiceLineTrackId] ON "invoice_items" ([TrackI
IFK_PlaylistTrackTrackId		CREATE INDEX [IFK_PlaylistTrackTrackId] ON "playlist_track" ([TrackI
IFK_TrackAlbumId		CREATE INDEX [IFK_TrackAlbumId] ON "tracks" ([AlbumId])
IFK_TrackGenreId		CREATE INDEX [IFK_TrackGenreId] ON "tracks" ([GenreId])

Figure 2. Opening the chinook.db in DB Browser

In this part, we opened the chinook.db sample database. It shows in the panel the different table and indices that are available in the sample database. It consists of 13 tables and 10 indices.



```
SELECT *
FROM albums
```

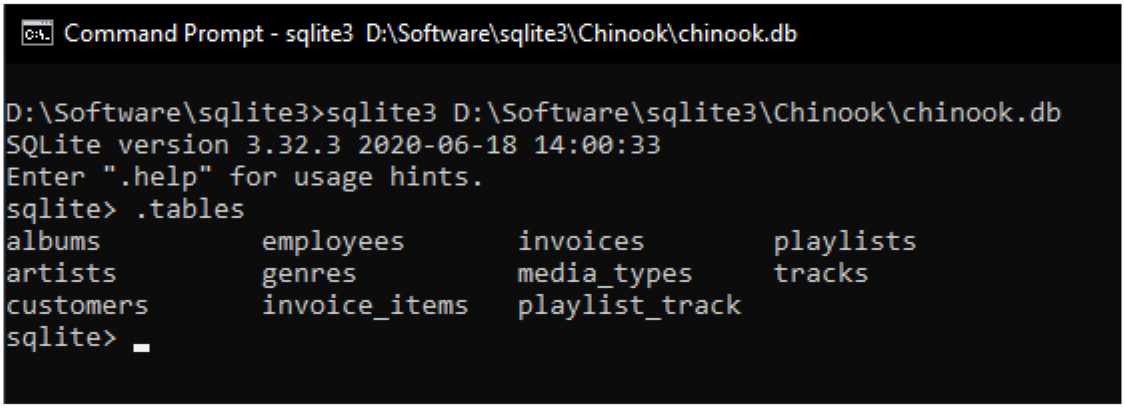
	AlbumId	Title	ArtistId
1	1	For Those About To Rock We Salute You	1
2	2	Balls to the Wall	2
3	3	Restless and Wild	2
4	4	Let There Be Rock	1
5	5	Big Ones	3
6	6	Jagged Little Pill	4

```
Execution finished without errors.
Result: 347 rows returned in 619ms
At line 1:
SELECT *
FROM albums
```

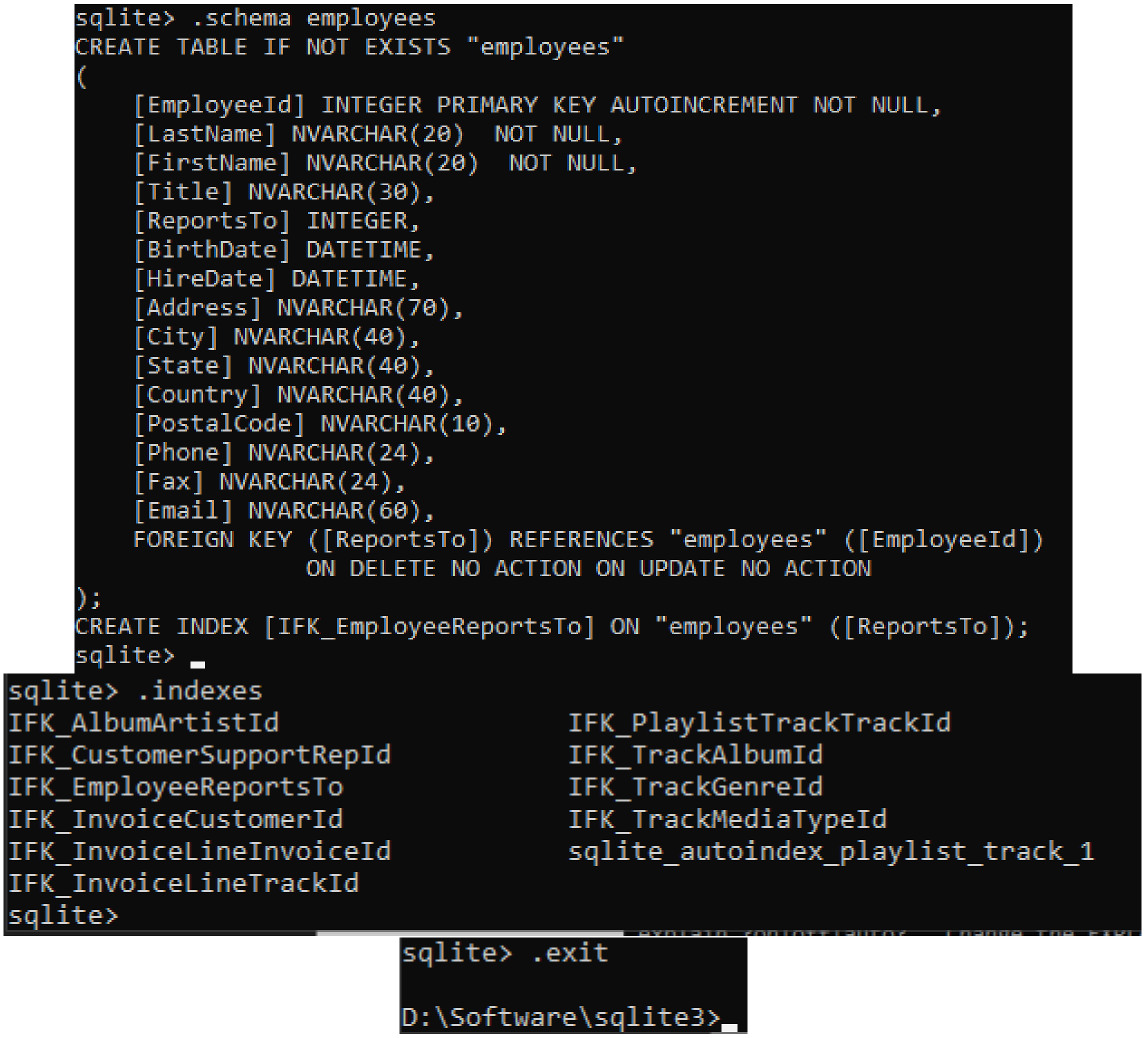
Figure 3. Running SQL Queries

In the Execute SQL tab, we are able to run queries in the environment. Below is a simple query that displays the contents of the albums table in the chinook database using the SELECT * FROM albums command. As observed, the albums consist of AlbumID as integer, Title as varchar, and ArtistID as integer.

Python SQLite Database Connection


Command Prompt - sqlite3 D:\Software\sqlite3\Chinook\chinook.db

```
D:\Software\sqlite3>sqlite3 D:\Software\sqlite3\Chinook\chinook.db
SQLite version 3.32.3 2020-06-18 14:00:33
Enter ".help" for usage hints.
sqlite> .tables
albums          employees       invoices        playlists
artists         genres          media_types     tracks
customers       invoice_items   playlist_track
sqlite> _
```



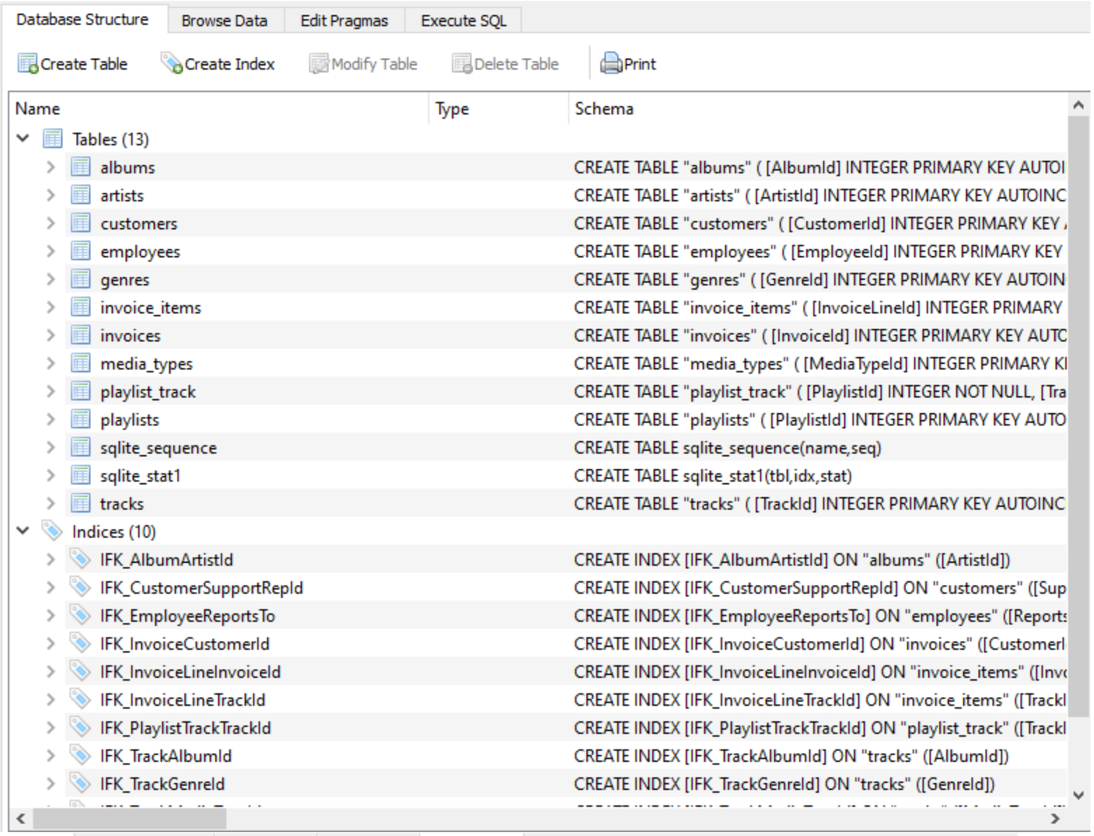
```
sqlite> .schema employees
CREATE TABLE IF NOT EXISTS "employees"
(
    [EmployeeId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    [LastName] NVARCHAR(20)  NOT NULL,
    [FirstName] NVARCHAR(20)  NOT NULL,
    [Title] NVARCHAR(30),
    [ReportsTo] INTEGER,
    [BirthDate] DATETIME,
    [HireDate] DATETIME,
    [Address] NVARCHAR(70),
    [City] NVARCHAR(40),
    [State] NVARCHAR(40),
    [Country] NVARCHAR(40),
    [PostalCode] NVARCHAR(10),
    [Phone] NVARCHAR(24),
    [Fax] NVARCHAR(24),
    [Email] NVARCHAR(60),
    FOREIGN KEY ([ReportsTo]) REFERENCES "employees" ([EmployeeId])
                ON DELETE NO ACTION ON UPDATE NO ACTION
);
CREATE INDEX [IFK_EmployeeReportsTo] ON "employees" ([ReportsTo]);
sqlite>
```

```
sqlite> .indexes
IFK_AlbumArtistId                   IFK_PlaylistTrackTrackId
IFK_CustomerSupportRepId            IFK_TrackAlbumId
IFK_EmployeeReportsTo               IFK_TrackGenreId
IFK_InvoiceCustomerId               IFK_TrackMediaTypeId
IFK_InvoiceLineInvoiceId            sqlite_autoindex_playlist_track_1
IFK_InvoiceLineTrackId
sqlite>
```

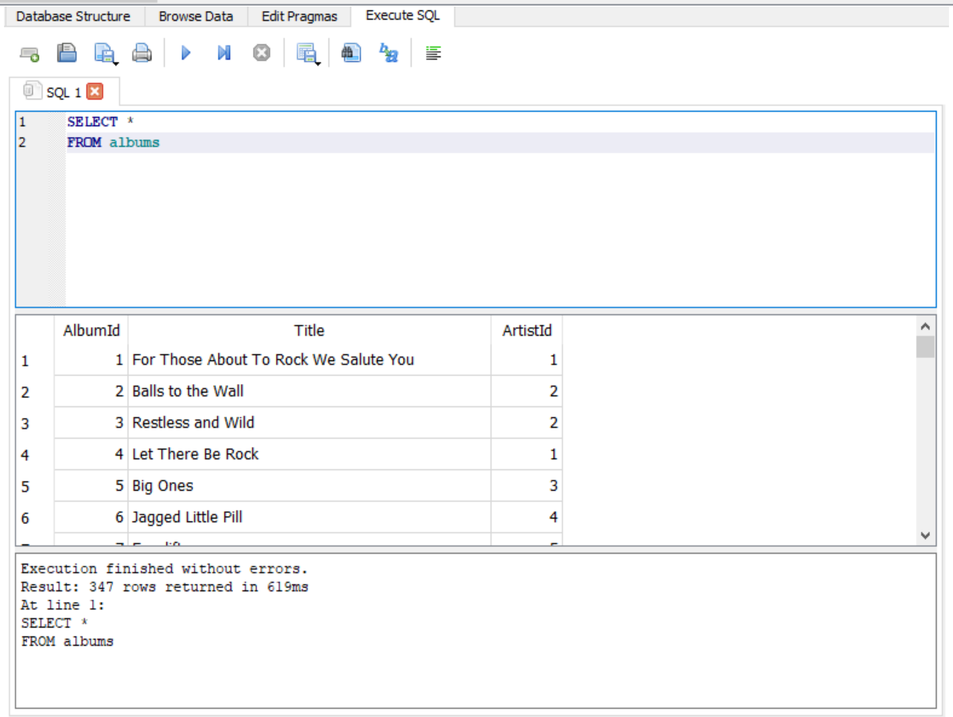
```
sqlite> .exit

D:\Software\sqlite3>
```

Figure 1. Running SQLite in the Command Line



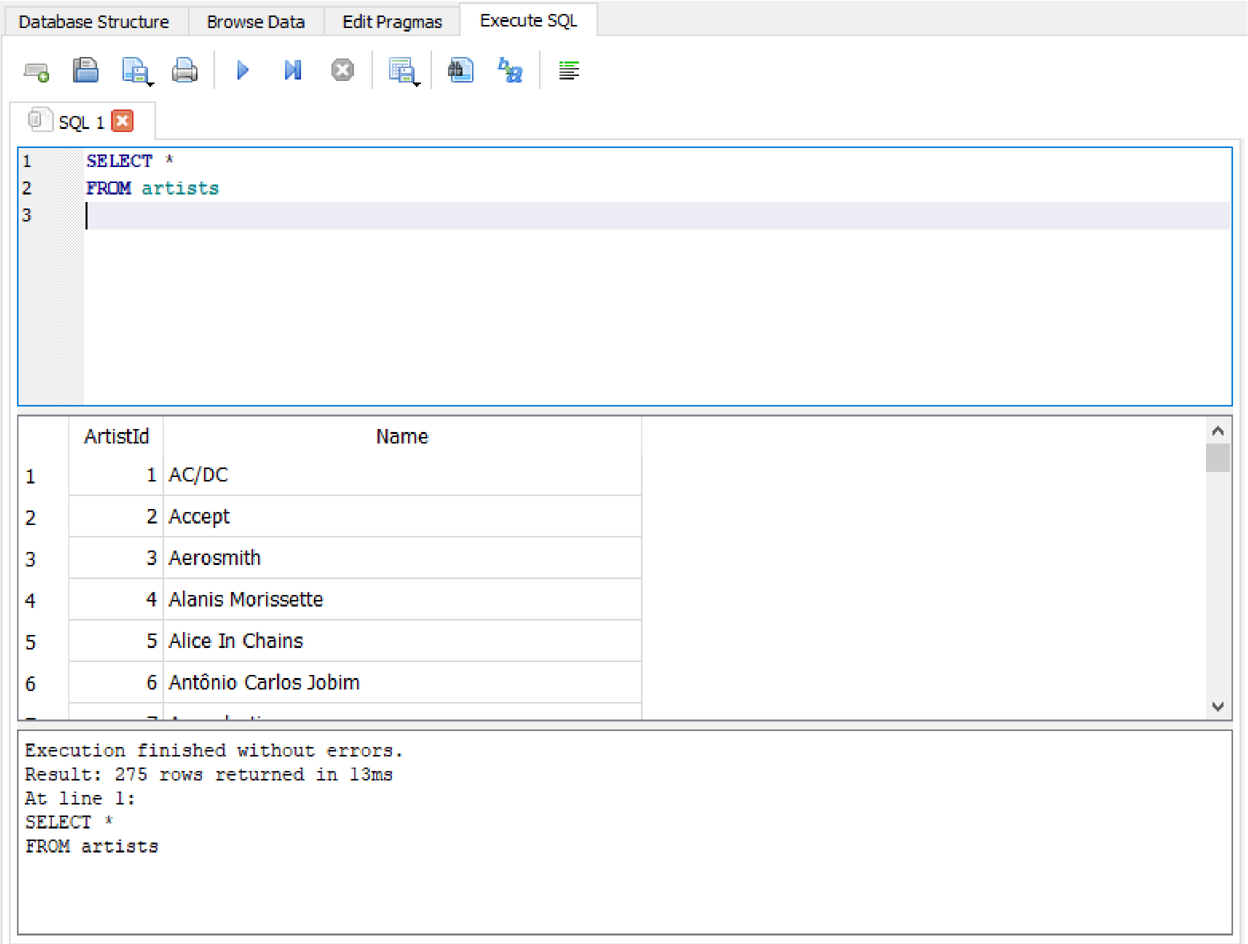
Name	Type	Schema
Tables (13)		
albums		CREATE TABLE "albums" ([AlbumId] INTEGER PRIMARY KEY AUTOI
artists		CREATE TABLE "artists" ([ArtistId] INTEGER PRIMARY KEY AUTOINC
customers		CREATE TABLE "customers" ([CustomerId] INTEGER PRIMARY KEY
employees		CREATE TABLE "employees" ([EmployeeId] INTEGER PRIMARY KEY
genres		CREATE TABLE "genres" ([GenreId] INTEGER PRIMARY KEY AUTOIN
invoice_items		CREATE TABLE "invoice_items" ([InvoiceLineId] INTEGER PRIMARY
invoices		CREATE TABLE "invoices" ([InvoiceId] INTEGER PRIMARY KEY AUTO
media_types		CREATE TABLE "media_types" ([MediaTypeId] INTEGER PRIMARY KI
playlist_track		CREATE TABLE "playlist_track" ([PlaylistId] INTEGER NOT NULL, [Tra
playlists		CREATE TABLE "playlists" ([PlaylistId] INTEGER PRIMARY KEY AUTO
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
sqlite_stat1		CREATE TABLE sqlite_stat1(tbl,idx,stat)
tracks		CREATE TABLE "tracks" ([TrackId] INTEGER PRIMARY KEY AUTOINC
Indices (10)		
IFK_AlbumArtistId		CREATE INDEX [IFK_AlbumArtistId] ON "albums" ([ArtistId])
IFK_CustomerSupportRepId		CREATE INDEX [IFK_CustomerSupportRepId] ON "customers" ([Sup
IFK_EmployeeReportsTo		CREATE INDEX [IFK_EmployeeReportsTo] ON "employees" ([Reports
IFK_InvoiceCustomerId		CREATE INDEX [IFK_InvoiceCustomerId] ON "invoices" ([CustomerI
IFK_InvoiceLineInvoiceId		CREATE INDEX [IFK_InvoiceLineInvoiceId] ON "invoice_items" ([Invo
IFK_InvoiceLineTrackId		CREATE INDEX [IFK_InvoiceLineTrackId] ON "invoice_items" ([TrackI
IFK_PlaylistTrackTrackId		CREATE INDEX [IFK_PlaylistTrackTrackId] ON "playlist_track" ([TrackI
IFK_TrackAlbumId		CREATE INDEX [IFK_TrackAlbumId] ON "tracks" ([AlbumId])
IFK_TrackGenreId		CREATE INDEX [IFK_TrackGenreId] ON "tracks" ([GenreId])



```sql
SELECT *
FROM albums
```

	AlbumId	Title	ArtistId
1	1	For Those About To Rock We Salute You	1
2	2	Balls to the Wall	2
3	3	Restless and Wild	2
4	4	Let There Be Rock	1
5	5	Big Ones	3
6	6	Jagged Little Pill	4

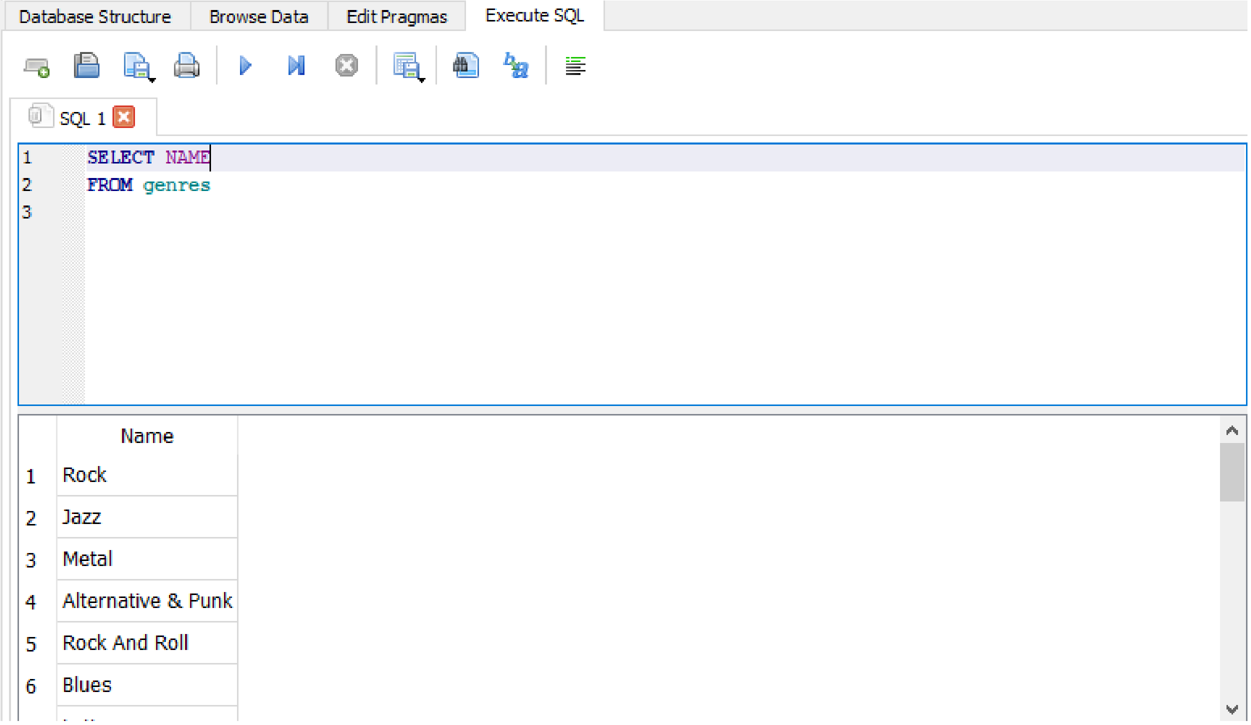
```
Execution finished without errors.
Result: 347 rows returned in 619ms
At line 1:
SELECT *
FROM albums
```



```
SELECT *
FROM artists
```

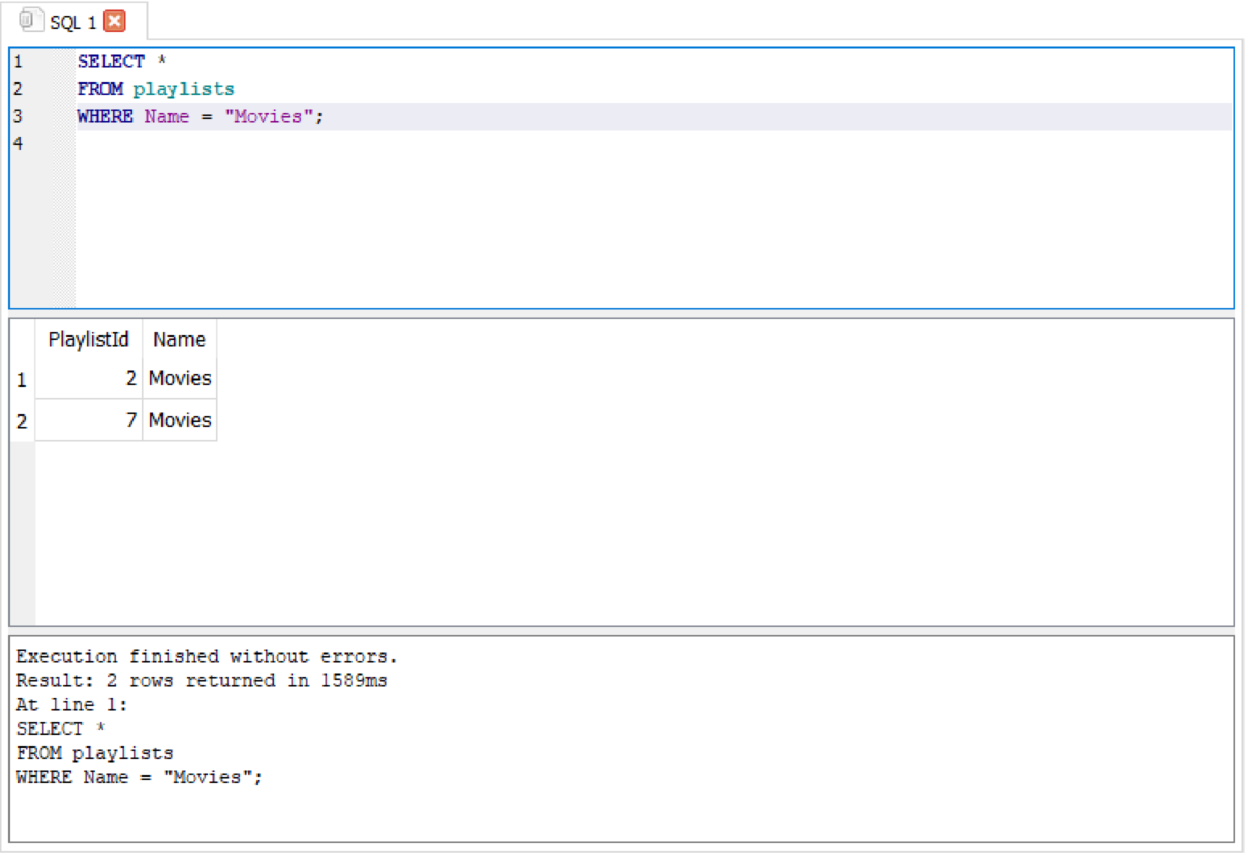
	ArtistId	Name
1	1	AC/DC
2	2	Accept
3	3	Aerosmith
4	4	Alanis Morissette
5	5	Alice In Chains
6	6	Antônio Carlos Jobim

```
Execution finished without errors.
Result: 275 rows returned in 13ms
At line 1:
SELECT *
FROM artists
```



```
SELECT NAME
FROM genres
```

	Name
1	Rock
2	Jazz
3	Metal
4	Alternative & Punk
5	Rock And Roll
6	Blues



```sql
SELECT *
FROM playlists
WHERE Name = "Movies";
```

	PlaylistId	Name
1	2	Movies
2	7	Movies

```
Execution finished without errors.
Result: 2 rows returned in 1589ms
At line 1:
SELECT *
FROM playlists
WHERE Name = "Movies";
```

Figure 2. Running SQLite in DB

PostLab

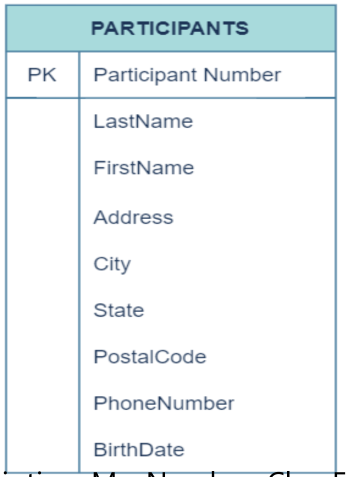
A. Machine Problems

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.

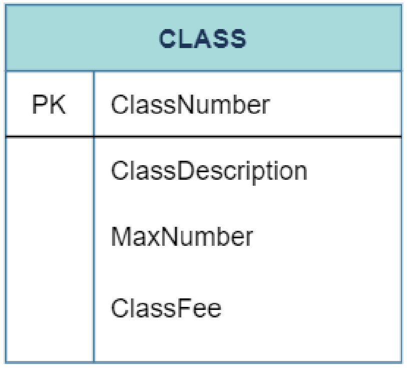
ANSWER

a) PARTICIPANTS (ParticipantNumber, LastName, FirstName, Address, City, State, PostalCode, PhoneNumber, BirthDate)

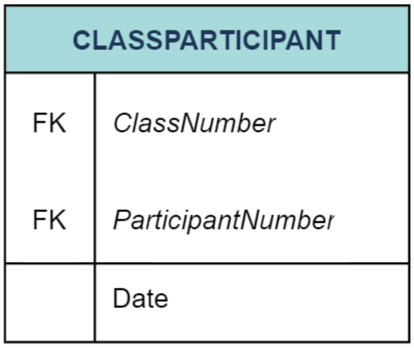


PARTICIPANTS	
PK	Participant Number
	LastName
	FirstName
	Address
	City
	State
	PostalCode
	PhoneNumber
	BirthDate

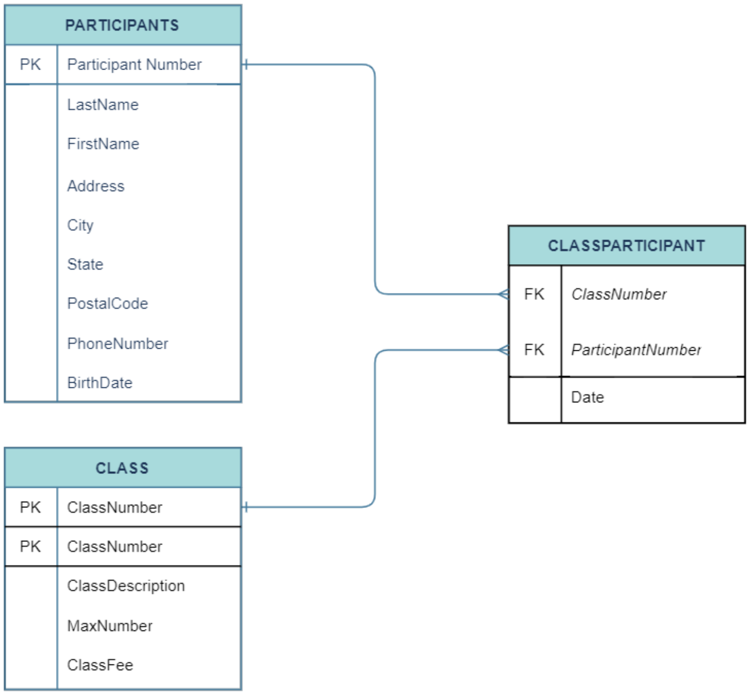
b) CLASS (ClassNumber, ClassDescription, MaxNumber, ClassFee)



c) CLASSPARTICIPANT (*ClassNumber*, *ParticpantNumber,* Date)



d) CLASSPARTICIPANT (*ClassNumber*, *ParticpantNumber*, Date, ActualParticipant)



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

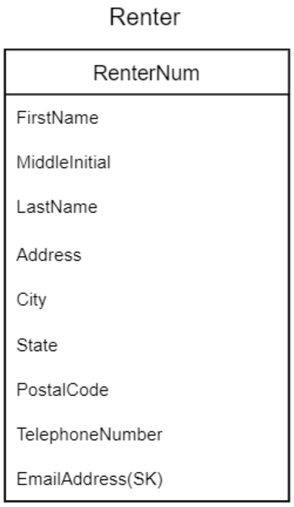
a.) For each renter, list his or her number, first name, middle initial, last name, address, city, state, postal code, telephone number, and email address.

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.

ANSWER

a.) Renter (RenterNum, FirstName, MiddleInitial, LastName, Address, City, State, PostalCode, TelephoneNum, EmailAddress)



Renter

RenterNum

FirstName

MiddleInitial

LastName

Address

City

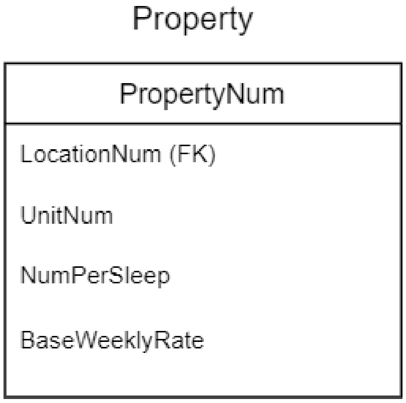
State

PostalCode

TelephoneNumber

EmailAddress(SK)

b.) Property (PropertyNum, LocationNum, UnitNum, NumPerSleep, BaseWeeklyRate)



Property

PropertyNum

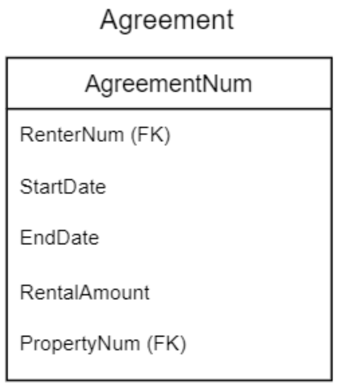
LocationNum (FK)

UnitNum

NumPerSleep

BaseWeeklyRate

c.) Agreement (AgreementNum, RenterNum, StartDate, EndDate, RenTalAmount, PropertyNum)



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.

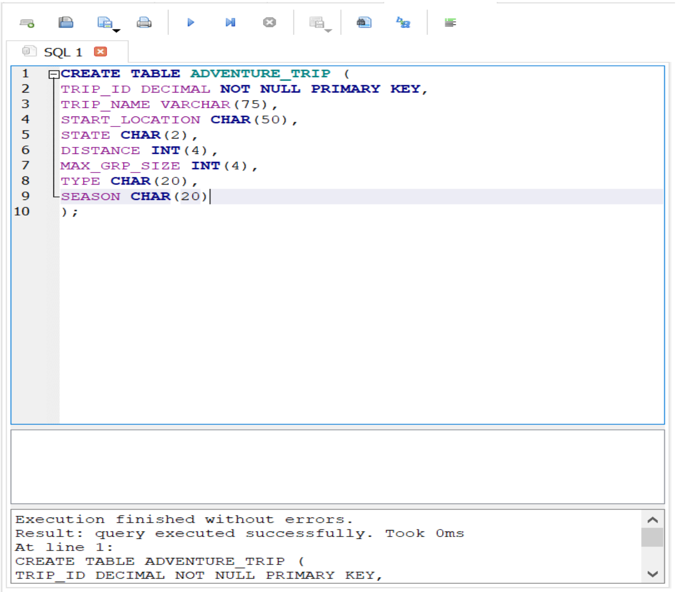
b.) Add the following row to the ADVENTURE_TRIP table: trip ID: 45; trip name: Jay Peak; start location: Jay; state: VT; distance: 8; maximum group size: 8; type: Hiking and sea- son: Summer. Display the contents of the ADVENTURE_TRIP table.

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

ANSWER

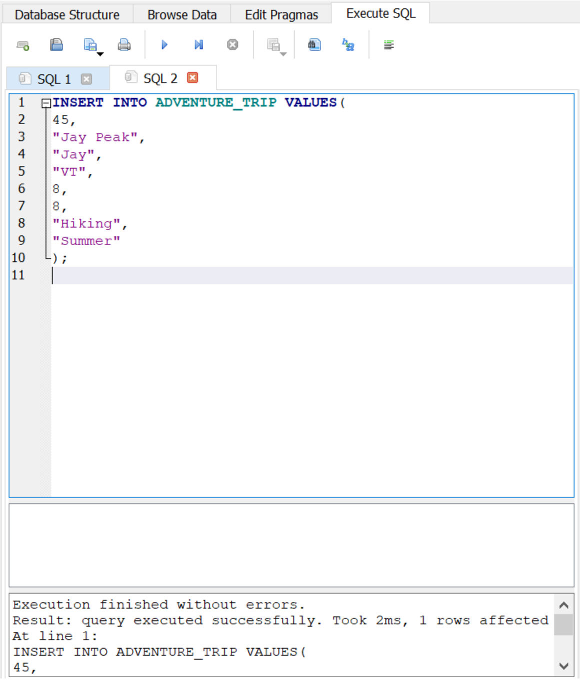


```
CREATE TABLE ADVENTURE_TRIP (
TRIP_ID DECIMAL NOT NULL PRIMARY KEY,
TRIP_NAME VARCHAR(75),
START_LOCATION CHAR(50),
STATE CHAR(2),
DISTANCE INT(4),
MAX_GRP_SIZE INT(4),
TYPE CHAR(20),
SEASON CHAR(20)
);
```

```
Execution finished without errors.
Result: query executed successfully. Took 0ms
At line 1:
CREATE TABLE ADVENTURE_TRIP (
TRIP_ID DECIMAL NOT NULL PRIMARY KEY,
```

Figure 1. Create ADVENTURE_TRIP table

This figure shows how the instructed table is created. If the program runs and does not recognize any table within the directory, the program will create its own table.

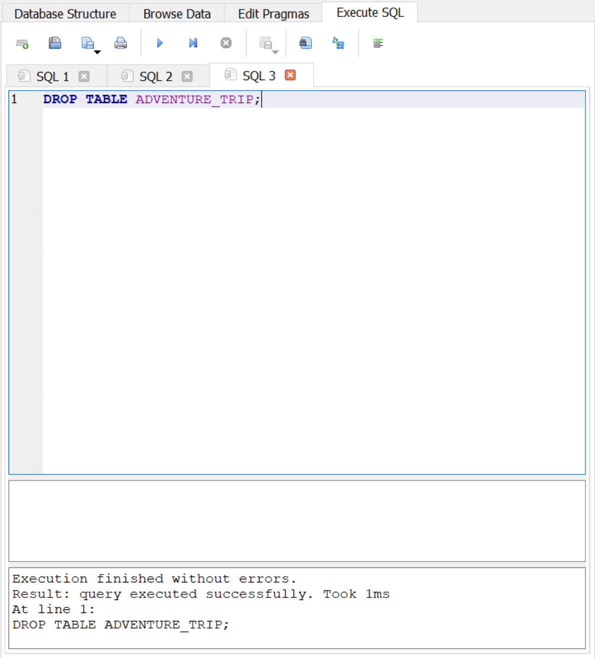


```
INSERT INTO ADVENTURE_TRIP VALUES(
45,
"Jay Peak",
"Jay",
"VT",
8,
8,
"Hiking",
"Summer"
);
```

```
Execution finished without errors.
Result: query executed successfully. Took 2ms, 1 rows affected
At line 1:
INSERT INTO ADVENTURE_TRIP VALUES(
45,
```

Figure 2. Insert data to ADVENTURE_TRIP Table

The figure shows how the data, namely 45, Jay Peak, Jay, Vt, 8, 8, hiking, and summer are inserted into the columns of the ADVENTURE_TRIP table.

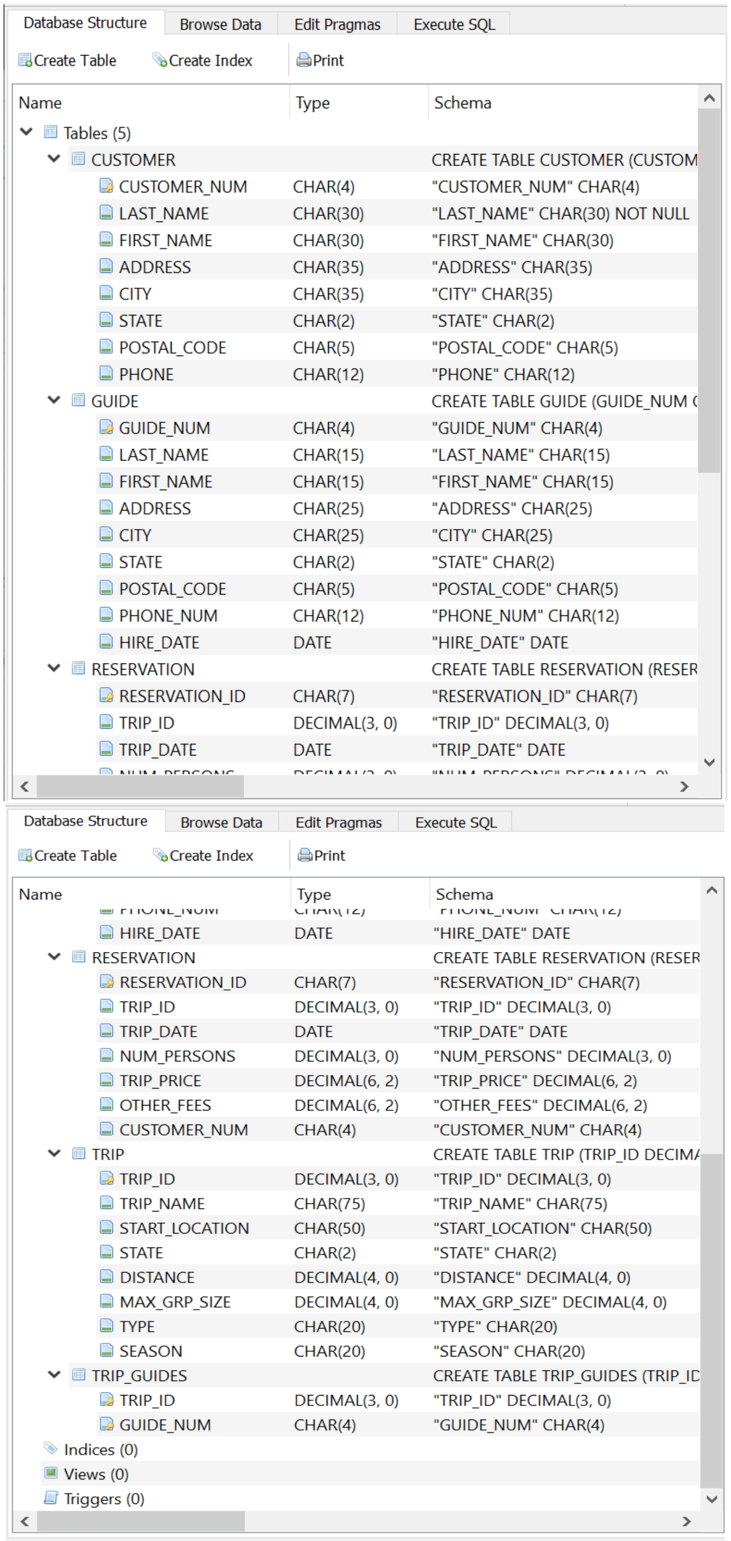


```
DROP TABLE ADVENTURE_TRIP;
```

```
Execution finished without errors.
Result: query executed successfully. Took 1ms
At line 1:
DROP TABLE ADVENTURE_TRIP;
```

Figure 3. Delete ADVENTURE_TRIP table

The figure above shows the code line on how to delete a table, the user must use the drop syntax.

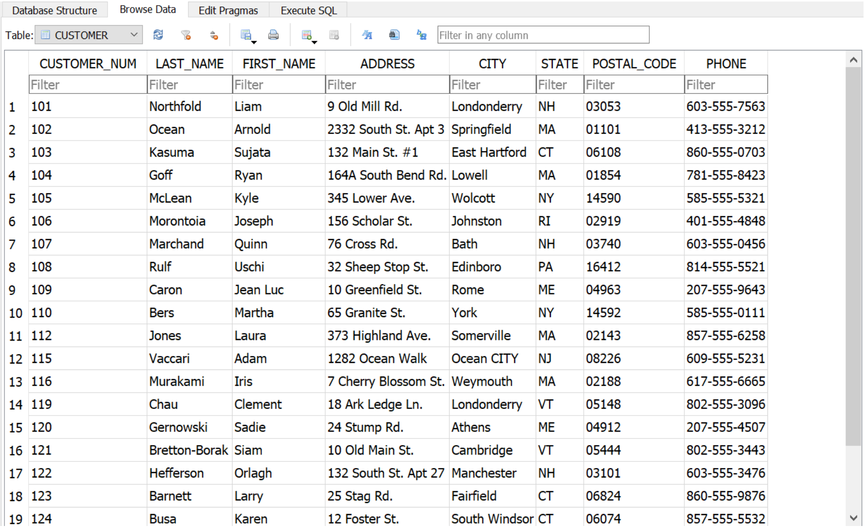


Name	Type	Schema
Tables (5)		
CUSTOMER		CREATE TABLE CUSTOMER (CUSTOM
CUSTOMER_NUM	CHAR(4)	"CUSTOMER_NUM" CHAR(4)
LAST_NAME	CHAR(30)	"LAST_NAME" CHAR(30) NOT NULL
FIRST_NAME	CHAR(30)	"FIRST_NAME" CHAR(30)
ADDRESS	CHAR(35)	"ADDRESS" CHAR(35)
CITY	CHAR(35)	"CITY" CHAR(35)
STATE	CHAR(2)	"STATE" CHAR(2)
POSTAL_CODE	CHAR(5)	"POSTAL_CODE" CHAR(5)
PHONE	CHAR(12)	"PHONE" CHAR(12)
GUIDE		CREATE TABLE GUIDE (GUIDE_NUM (
GUIDE_NUM	CHAR(4)	"GUIDE_NUM" CHAR(4)
LAST_NAME	CHAR(15)	"LAST_NAME" CHAR(15)
FIRST_NAME	CHAR(15)	"FIRST_NAME" CHAR(15)
ADDRESS	CHAR(25)	"ADDRESS" CHAR(25)
CITY	CHAR(25)	"CITY" CHAR(25)
STATE	CHAR(2)	"STATE" CHAR(2)
POSTAL_CODE	CHAR(5)	"POSTAL_CODE" CHAR(5)
PHONE_NUM	CHAR(12)	"PHONE_NUM" CHAR(12)
HIRE_DATE	DATE	"HIRE_DATE" DATE
RESERVATION		CREATE TABLE RESERVATION (RESER
RESERVATION_ID	CHAR(7)	"RESERVATION_ID" CHAR(7)
TRIP_ID	DECIMAL(3, 0)	"TRIP_ID" DECIMAL(3, 0)
TRIP_DATE	DATE	"TRIP_DATE" DATE
NUM_PERSONS	DECIMAL(3, 0)	"NUM_PERSONS" DECIMAL(3, 0)

Name	Type	Schema
PHONE_NUM	CHAR(12)	"PHONE_NUM" CHAR(12)
HIRE_DATE	DATE	"HIRE_DATE" DATE
RESERVATION		CREATE TABLE RESERVATION (RESER
RESERVATION_ID	CHAR(7)	"RESERVATION_ID" CHAR(7)
TRIP_ID	DECIMAL(3, 0)	"TRIP_ID" DECIMAL(3, 0)
TRIP_DATE	DATE	"TRIP_DATE" DATE
NUM_PERSONS	DECIMAL(3, 0)	"NUM_PERSONS" DECIMAL(3, 0)
TRIP_PRICE	DECIMAL(6, 2)	"TRIP_PRICE" DECIMAL(6, 2)
OTHER_FEES	DECIMAL(6, 2)	"OTHER_FEES" DECIMAL(6, 2)
CUSTOMER_NUM	CHAR(4)	"CUSTOMER_NUM" CHAR(4)
TRIP		CREATE TABLE TRIP (TRIP_ID DECIMA
TRIP_ID	DECIMAL(3, 0)	"TRIP_ID" DECIMAL(3, 0)
TRIP_NAME	CHAR(75)	"TRIP_NAME" CHAR(75)
START_LOCATION	CHAR(50)	"START_LOCATION" CHAR(50)
STATE	CHAR(2)	"STATE" CHAR(2)
DISTANCE	DECIMAL(4, 0)	"DISTANCE" DECIMAL(4, 0)
MAX_GRP_SIZE	DECIMAL(4, 0)	"MAX_GRP_SIZE" DECIMAL(4, 0)
TYPE	CHAR(20)	"TYPE" CHAR(20)
SEASON	CHAR(20)	"SEASON" CHAR(20)
TRIP_GUIDES		CREATE TABLE TRIP_GUIDES (TRIP_ID
TRIP_ID	DECIMAL(3, 0)	"TRIP_ID" DECIMAL(3, 0)
GUIDE_NUM	CHAR(4)	"GUIDE_NUM" CHAR(4)
Indices (0)		
Views (0)		
Triggers (0)		

Figure 4. Script File

Here in figure 4, the script file was modified for it to run in the DB Browser.

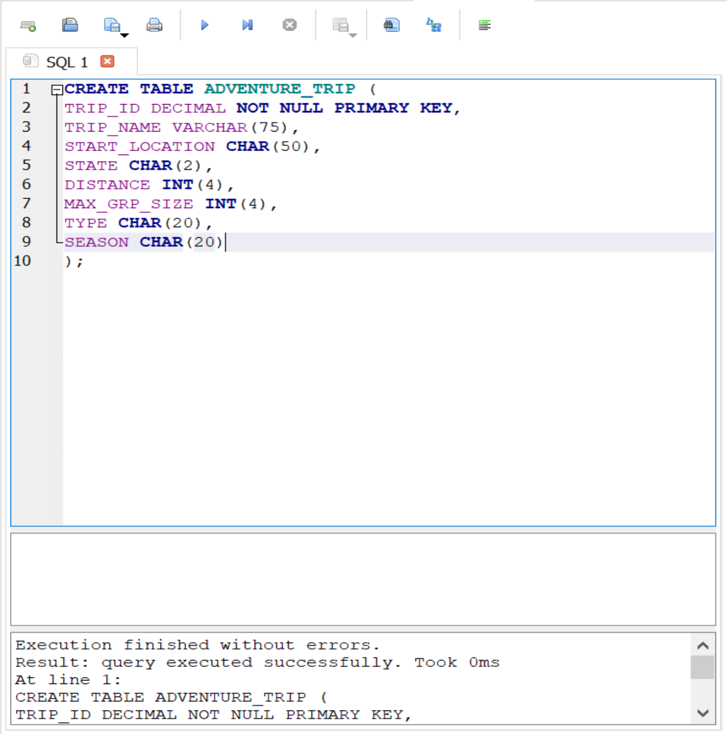


	CUSTOMER_NUM	LAST_NAME	FIRST_NAME	ADDRESS	CITY	STATE	POSTAL_CODE	PHONE
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	101	Northfold	Liam	9 Old Mill Rd.	Londonderry	NH	03053	603-555-7563
2	102	Ocean	Arnold	2332 South St. Apt 3	Springfield	MA	01101	413-555-3212
3	103	Kasuma	Sujata	132 Main 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

Figure 5. Run the file in DB Browser

In figure 5, the script file was run in the DB Browser

B. Debugging and Sample Run

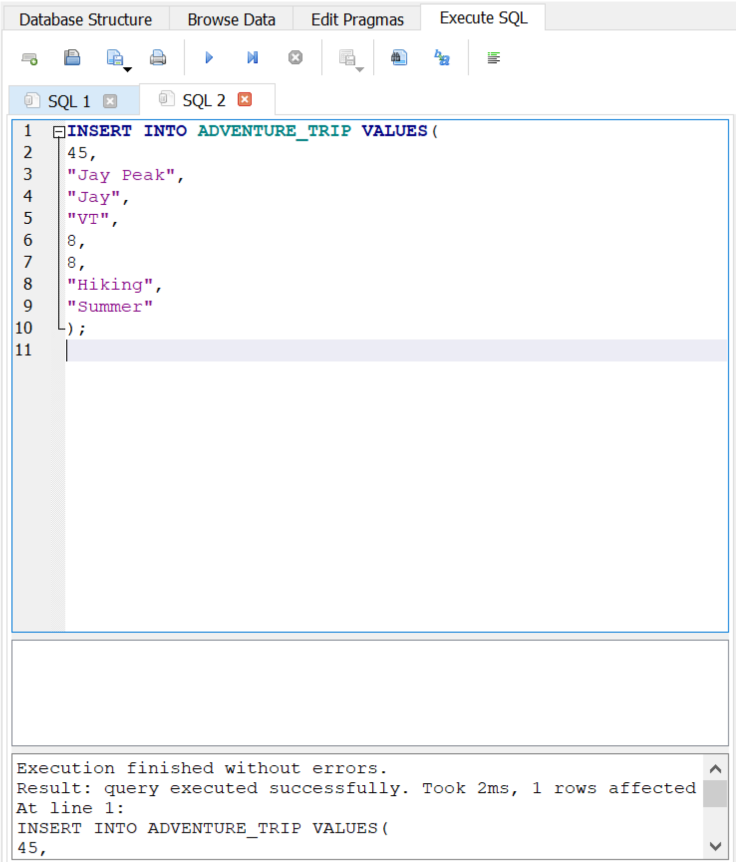


```
CREATE TABLE ADVENTURE_TRIP (
TRIP_ID DECIMAL NOT NULL PRIMARY KEY,
TRIP_NAME VARCHAR(75),
START_LOCATION CHAR(50),
STATE CHAR(2),
DISTANCE INT(4),
MAX_GRP_SIZE INT(4),
TYPE CHAR(20),
SEASON CHAR(20)
);
```

```
Execution finished without errors.
Result: query executed successfully. Took 0ms
At line 1:
CREATE TABLE ADVENTURE_TRIP (
TRIP_ID DECIMAL NOT NULL PRIMARY KEY,
```

Figure 1. Table ADVENTURE_TRIP

Figure 1 shows how the table ADVENTUR_TRIP is created.

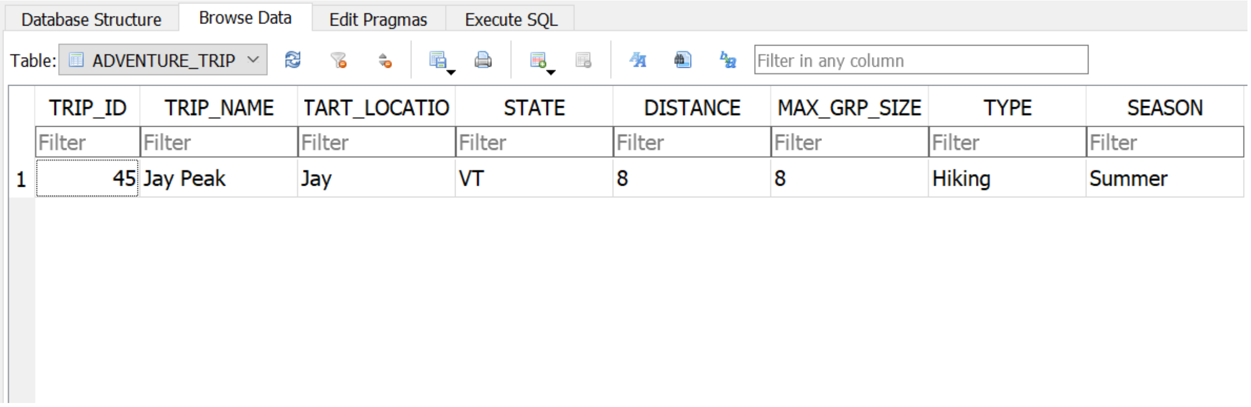


```
INSERT INTO ADVENTURE_TRIP VALUES(
45,
"Jay Peak",
"Jay",
"VT",
8,
8,
"Hiking",
"Summer"
);
```

```
Execution finished without errors.
Result: query executed successfully. Took 2ms, 1 rows affected
At line 1:
INSERT INTO ADVENTURE_TRIP VALUES(
45,
```

Figure 2. Inserting into ADVENTURE_TRIP

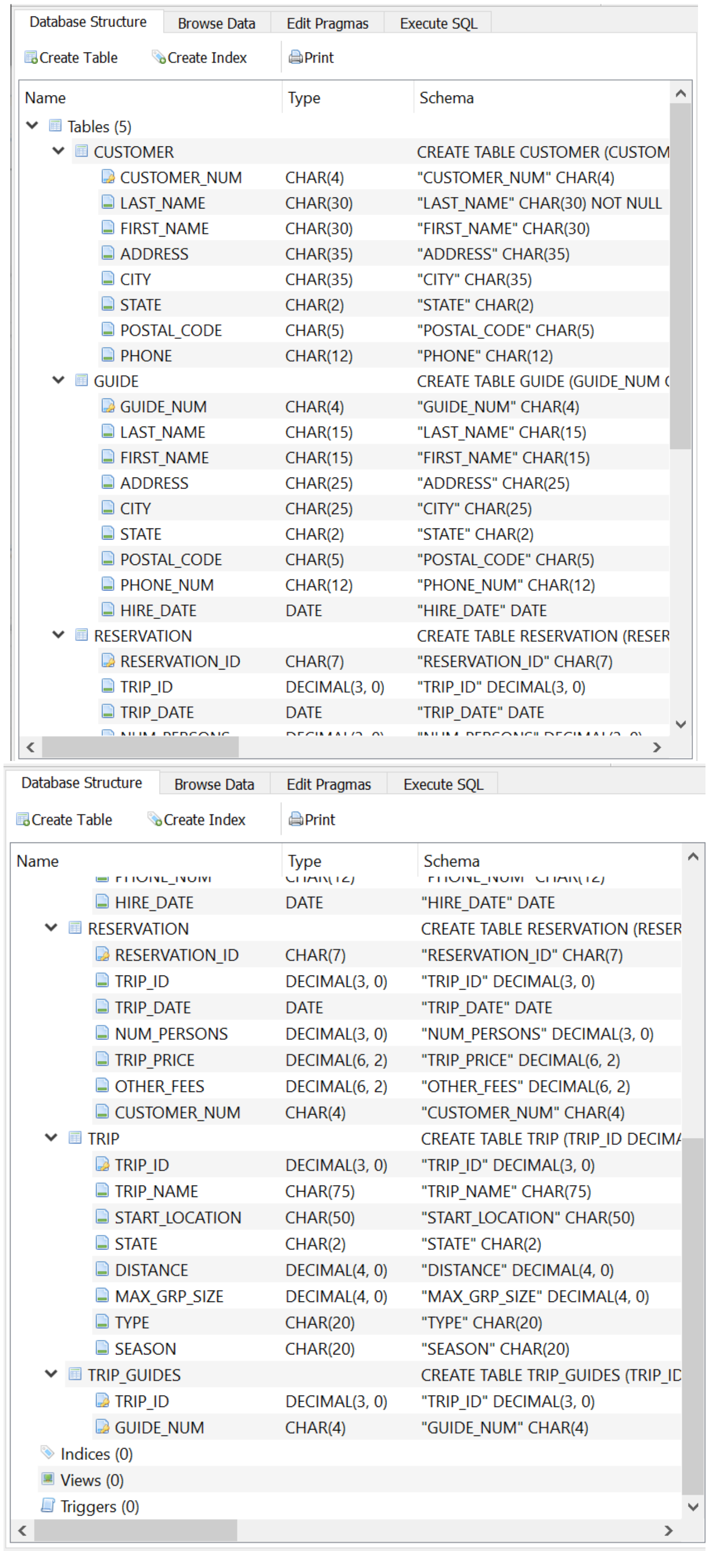
For figure 2, the group has set values and inserted it into the create table named ADVENTURE_TRIP.



	TRIP_ID	TRIP_NAME	TART_LOCATIO	STATE	DISTANCE	MAX_GRP_SIZE	TYPE	SEASON
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	45	Jay Peak	Jay	VT	8	8	Hiking	Summer

Figure 3. ADVENTURE_TRIP Table with contents

In figure 3, the table is shown with the values inserted in the code as seen in Figure 2.

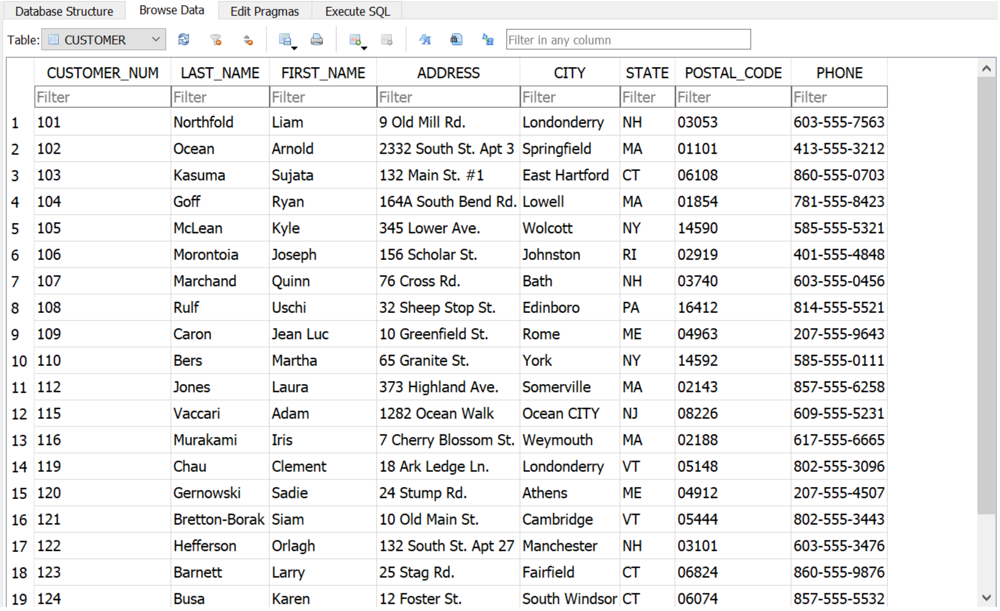


Name	Type	Schema
Tables (5)		
CUSTOMER		CREATE TABLE CUSTOMER (CUSTOM
CUSTOMER_NUM	CHAR(4)	"CUSTOMER_NUM" CHAR(4)
LAST_NAME	CHAR(30)	"LAST_NAME" CHAR(30) NOT NULL
FIRST_NAME	CHAR(30)	"FIRST_NAME" CHAR(30)
ADDRESS	CHAR(35)	"ADDRESS" CHAR(35)
CITY	CHAR(35)	"CITY" CHAR(35)
STATE	CHAR(2)	"STATE" CHAR(2)
POSTAL_CODE	CHAR(5)	"POSTAL_CODE" CHAR(5)
PHONE	CHAR(12)	"PHONE" CHAR(12)
GUIDE		CREATE TABLE GUIDE (GUIDE_NUM (
GUIDE_NUM	CHAR(4)	"GUIDE_NUM" CHAR(4)
LAST_NAME	CHAR(15)	"LAST_NAME" CHAR(15)
FIRST_NAME	CHAR(15)	"FIRST_NAME" CHAR(15)
ADDRESS	CHAR(25)	"ADDRESS" CHAR(25)
CITY	CHAR(25)	"CITY" CHAR(25)
STATE	CHAR(2)	"STATE" CHAR(2)
POSTAL_CODE	CHAR(5)	"POSTAL_CODE" CHAR(5)
PHONE_NUM	CHAR(12)	"PHONE_NUM" CHAR(12)
HIRE_DATE	DATE	"HIRE_DATE" DATE
RESERVATION		CREATE TABLE RESERVATION (RESER
RESERVATION_ID	CHAR(7)	"RESERVATION_ID" CHAR(7)
TRIP_ID	DECIMAL(3, 0)	"TRIP_ID" DECIMAL(3, 0)
TRIP_DATE	DATE	"TRIP_DATE" DATE
NUM_PERSONS	DECIMAL(3, 0)	"NUM_PERSONS" DECIMAL(3, 0)

Name	Type	Schema
PHONE_NUM	CHAR(12)	"PHONE_NUM" CHAR(12)
HIRE_DATE	DATE	"HIRE_DATE" DATE
RESERVATION		CREATE TABLE RESERVATION (RESER
RESERVATION_ID	CHAR(7)	"RESERVATION_ID" CHAR(7)
TRIP_ID	DECIMAL(3, 0)	"TRIP_ID" DECIMAL(3, 0)
TRIP_DATE	DATE	"TRIP_DATE" DATE
NUM_PERSONS	DECIMAL(3, 0)	"NUM_PERSONS" DECIMAL(3, 0)
TRIP_PRICE	DECIMAL(6, 2)	"TRIP_PRICE" DECIMAL(6, 2)
OTHER_FEES	DECIMAL(6, 2)	"OTHER_FEES" DECIMAL(6, 2)
CUSTOMER_NUM	CHAR(4)	"CUSTOMER_NUM" CHAR(4)
TRIP		CREATE TABLE TRIP (TRIP_ID DECIMA
TRIP_ID	DECIMAL(3, 0)	"TRIP_ID" DECIMAL(3, 0)
TRIP_NAME	CHAR(75)	"TRIP_NAME" CHAR(75)
START_LOCATION	CHAR(50)	"START_LOCATION" CHAR(50)
STATE	CHAR(2)	"STATE" CHAR(2)
DISTANCE	DECIMAL(4, 0)	"DISTANCE" DECIMAL(4, 0)
MAX_GRP_SIZE	DECIMAL(4, 0)	"MAX_GRP_SIZE" DECIMAL(4, 0)
TYPE	CHAR(20)	"TYPE" CHAR(20)
SEASON	CHAR(20)	"SEASON" CHAR(20)
TRIP_GUIDES		CREATE TABLE TRIP_GUIDES (TRIP_ID
TRIP_ID	DECIMAL(3, 0)	"TRIP_ID" DECIMAL(3, 0)
GUIDE_NUM	CHAR(4)	"GUIDE_NUM" CHAR(4)
Indices (0)		
Views (0)		
Triggers (0)		

Figure 4. Opening Script File

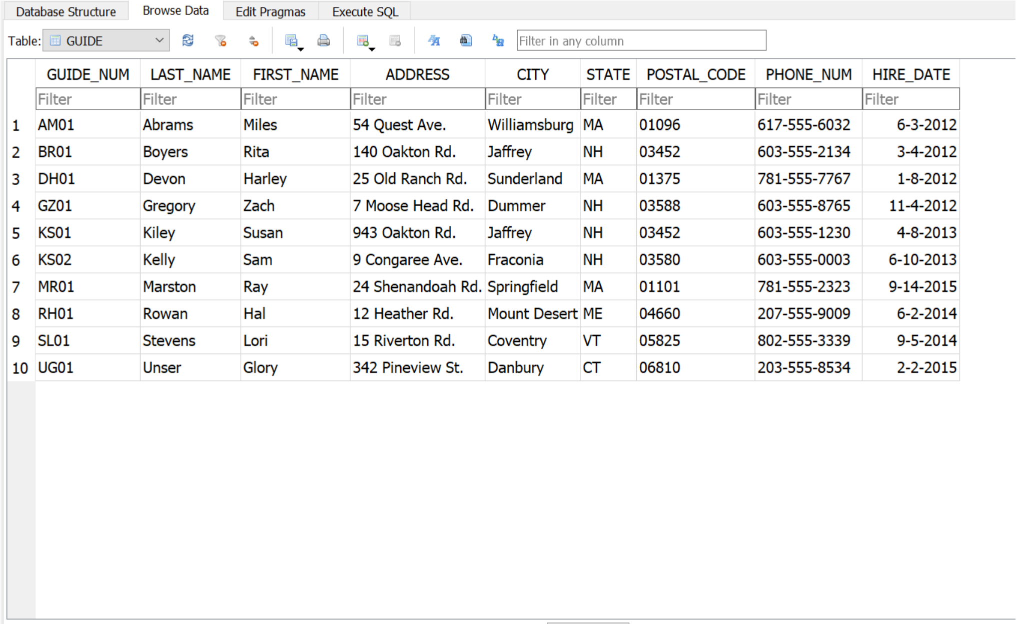
Figure 4 shows the entirety of the script files to be run in the DB Browser.



	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

Figure 5. Opened Customer File

Figure 5 shows the table and its contents in the Customer File.



	GUIDE_NUM	LAST_NAME	FIRST_NAME	ADDRESS	CITY	STATE	POSTAL_CODE	PHONE_NUM	HIRE_DATE
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
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 6. Opened Guide File

Figure 6 shows the table and its contents in the Guide File.

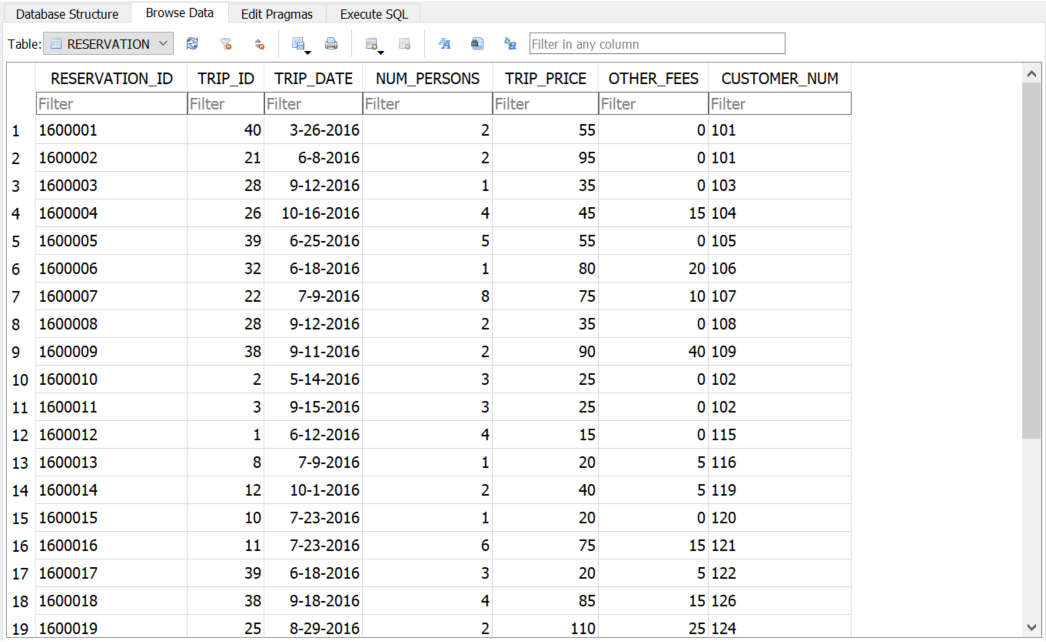


Table: RESERVATION

	RESERVATION_ID	TRIP_ID	TRIP_DATE	NUM_PERSONS	TRIP_PRICE	OTHER_FEES	CUSTOMER_NUM
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
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

Figure 7. Opened Reservation File

Figure 7 shows the table and its contents in the Reservation File.

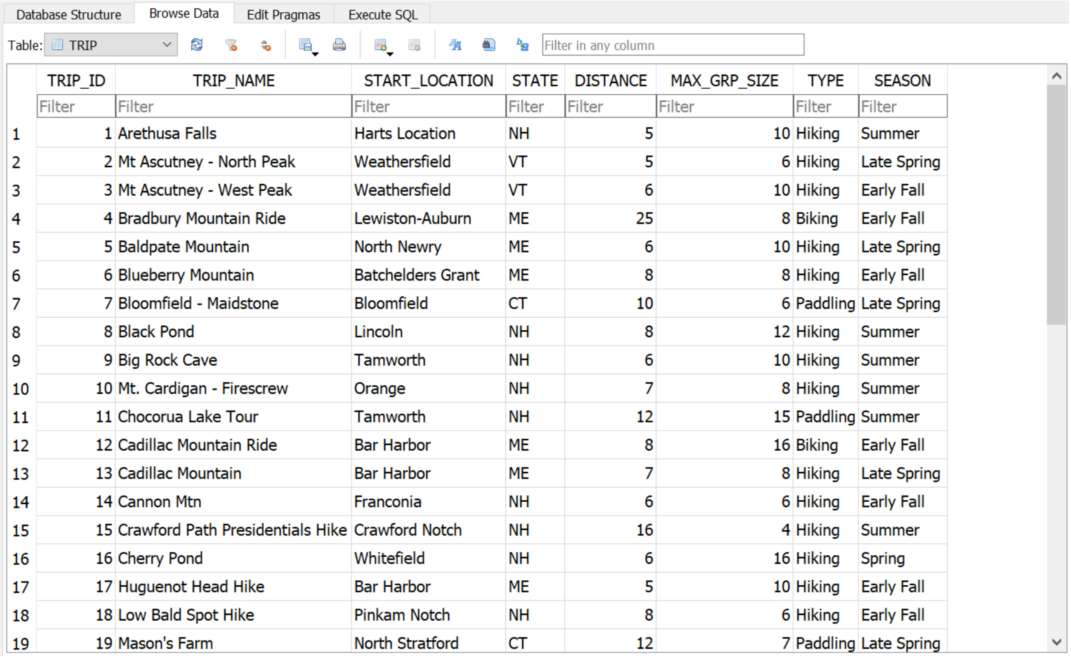
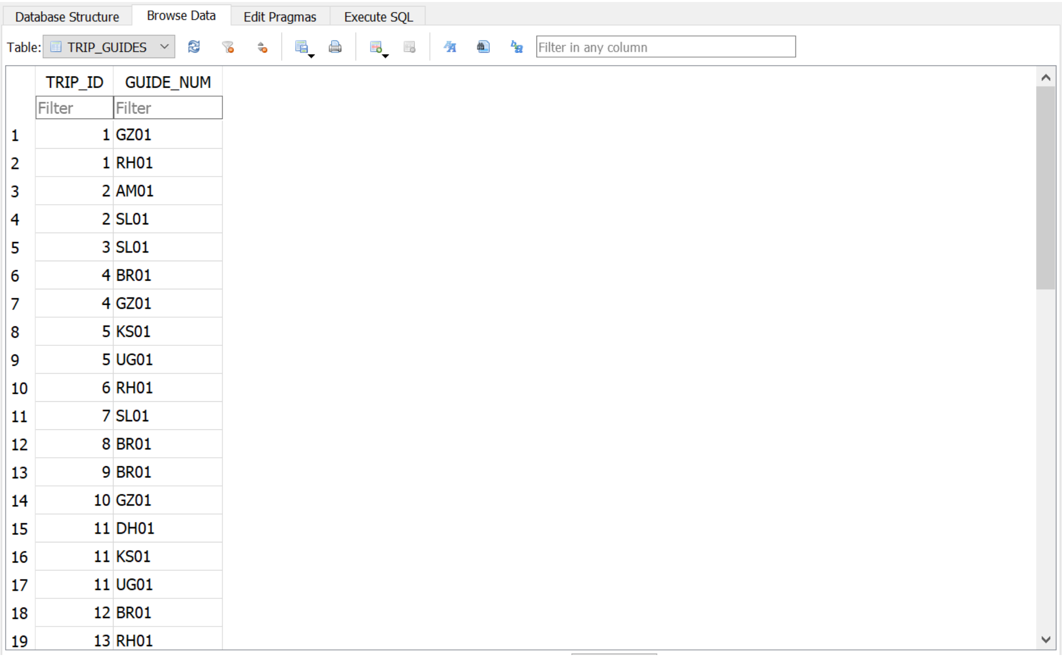


Table: TRIP

	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

Figure 8. Opened Trip File

Figure 8 shows the table and its contents in the Trip File.



	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

Figure 9. Opened Trip_Guides File

Figure 9 shows the table and its contents in the Trip_Guides File.

OneDrive:
Group 1: https://bit.ly/2DaQ58r

Github:
Darrel Virtusio: https://bit.ly/2PgDOSn
Hannah Antaran: https://bit.ly/30niUrb
Kathleen Tupas: https://bit.ly/2DrsGzk