MAPÚA UNIVERSITY
SCHOOL OF ELECTRICAL, ELECTRONICS, AND COMPUTER ENGINEERING

Experiment 5:
Data Modeling and Database Systems

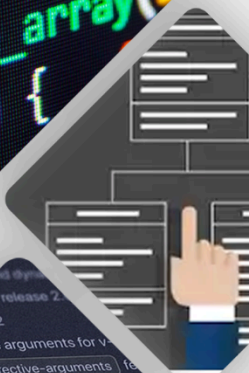## CPE106L (Software Design Laboratory)

Member 1: ASIDAO, Shane Eowyn R.
Member 2: GONZALES, Benedick B.
Member 3: LULU, Neal Axel M.
Member 4: VARGAS, Imar Colt S.

Group No.: **4**
Section: **E04**

PreLab

A. Readings, Insights, and Reflection

METIS ebooks:
- A Guide to SQL. Philip J. Pratt; et al.9780357419830
- VBID: 9780357419830

- Chapter 1:
    1. What is a Database
    2. Database Requirements of TAL Distributors, Colonial Adventure Tours, and Solmaris Condominium Group
- Chapter 2:
    1. Database Concepts
    2. Database Design Fundamentals
    3. Normalization

- Core Python Programming. R. Nageswara Rao. (Not available in METIS)

- VBID: 9789351198918
- Chapter 24:
    1. Types of Databases Used with Python
    2. Using MySQL from Python

- Python Projects. Laura Cassell. 9781118908891 Chapter 3
  Chapter 3:
    1. Relational Database Concepts
    2. Structured Query Language. DML and DDL SQL command

Website(s):
- https://www.sqlite.org/index.html
- https://www.sqlitetutorial.net/sqlite-sample-database/

Insights and Reflections

Gonzales and Vargas

Lulu and Asidao

B. Answer to Questions
● Short Answer
1. **What are DML and DDL statements in Structured Query Language? Give examples of each.**

DML (Data Manipulation Language) and DDL (Data Definition Language) are two types of SQL statements. DML statements are used to manipulate data within tables, such as INSERT, UPDATE, and DELETE, while DDL statements define and manage database structures, including CREATE, ALTER, and DROP. For example, INSERT INTO students VALUES (1, 'John Doe'); is a DML statement, while CREATE TABLE students (id INT, name TEXT); is a DDL statement.

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

SQLite functions are categorized into Aggregate Functions, Scalar Functions, and Window Functions. Aggregate functions perform calculations on multiple rows and return a single result, such as SUM(), AVG(), and COUNT(). Scalar functions operate on a single value and return a single value, like UPPER(), LOWER(), and LENGTH(). Window functions perform calculations across a set of table rows related to the current row, including ROW_NUMBER(), RANK(), and DENSE_RANK().

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

To check if SQLite is installed on a Linux system, you can use the terminal command sqlite3 --version. If SQLite is installed, this command will return the installed version number; otherwise, it will show a "command not found" error. Another way is to use sqlite3 to check the installation path of SQLite.

InLab

- **Objectives**
    1. **Modify** the sample programs provided on the OneDrive Source code folder.
    2. **Use** Visual Studio Code in modifying the code.
    3. **Run** a query using MySQLite extension in VSCode.

- **Steps Performed with screenshots of tools used:**
    - Visual Studio Code
    - GitHub
    - SQLite commands running in VSCode terminal
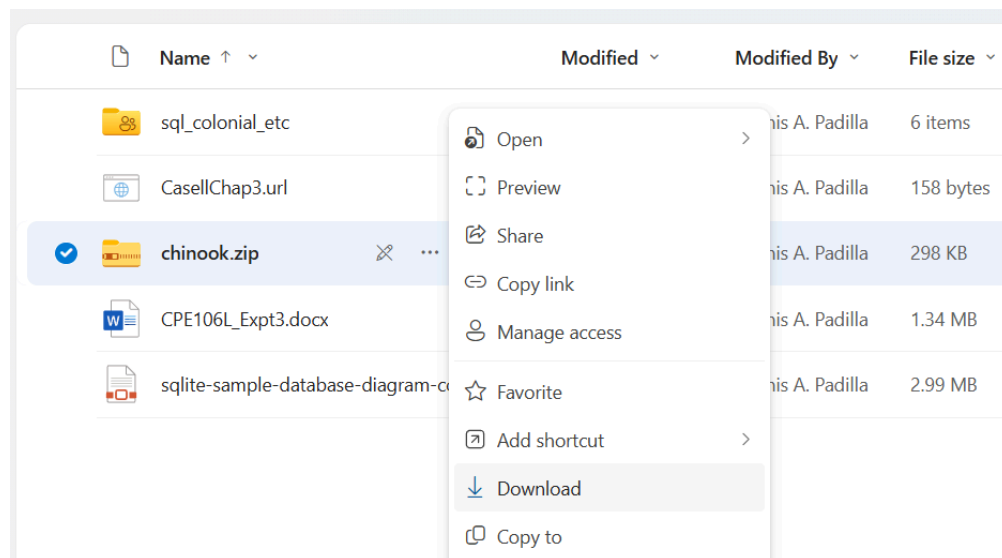   - SQLite commands running in DB Browser
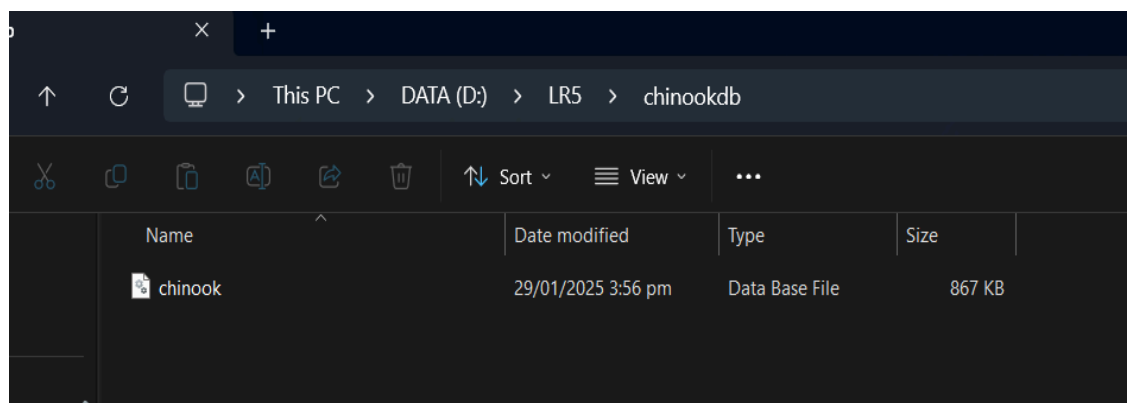

Figure 1.1 Download the database

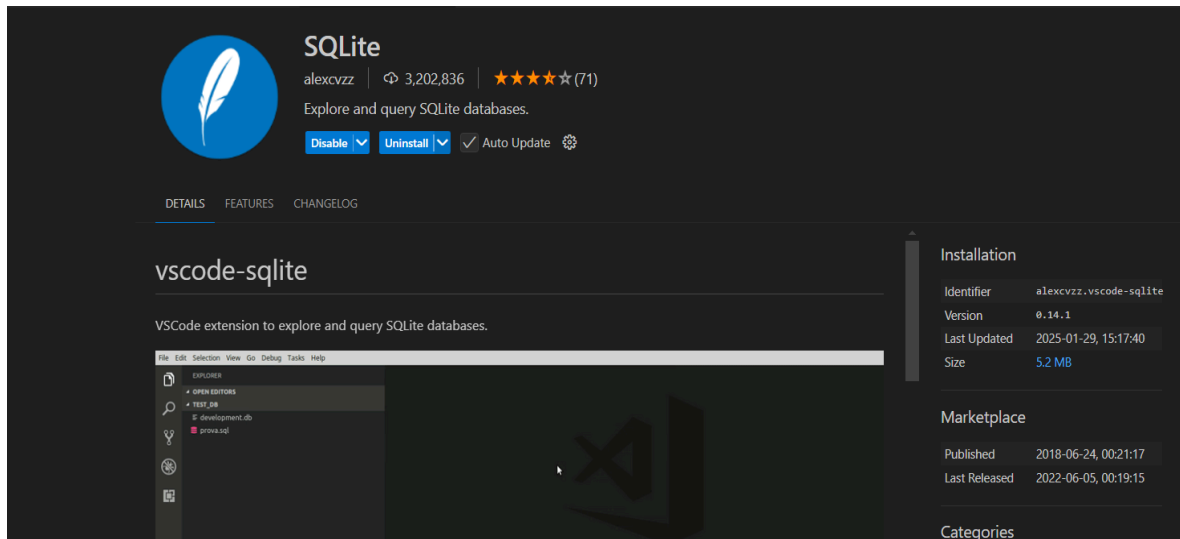**Figure 1.2** Save the database file in a folder

Figure 1.3 Install SQLite extension from Visual Studio Code

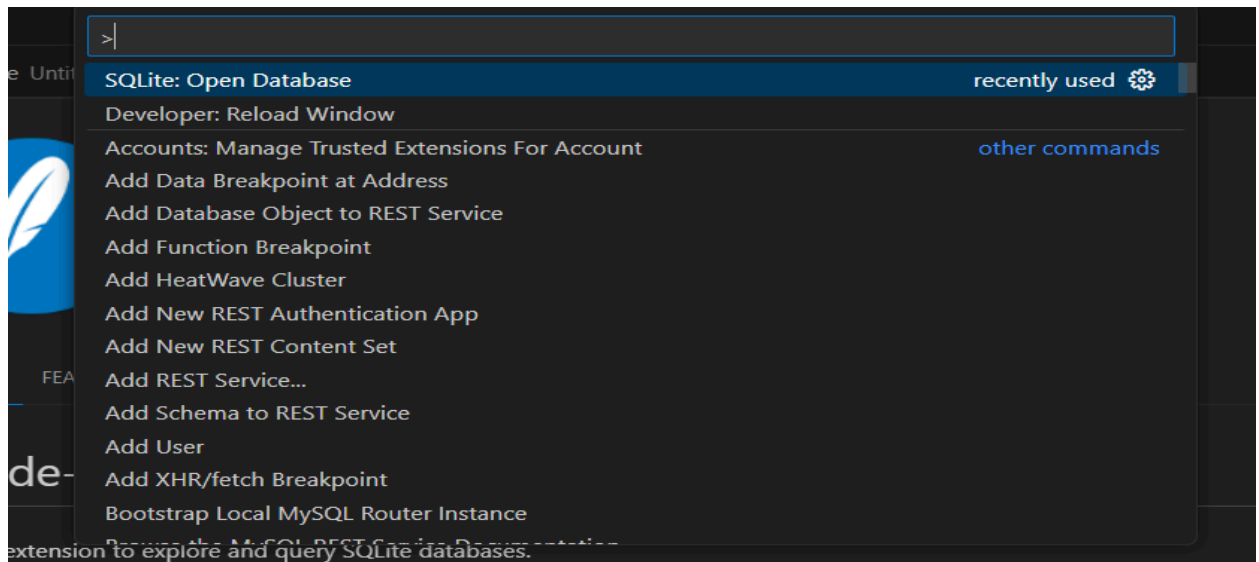After opening Visual Studio Code, go towards the 'extensions' tab and download SQLite by alexcvzz.

Figure 1.4 Press [Shift] + [Ctrl] + [P], to open the Command Palette

After opening the Command Palette, select the "SQLite: Open Database" and select the file directory of your database. Afterwards, select the 'Explorer' tab from your Visual Studio Code and you should be able to see the SQLite Explorer from within the tab.
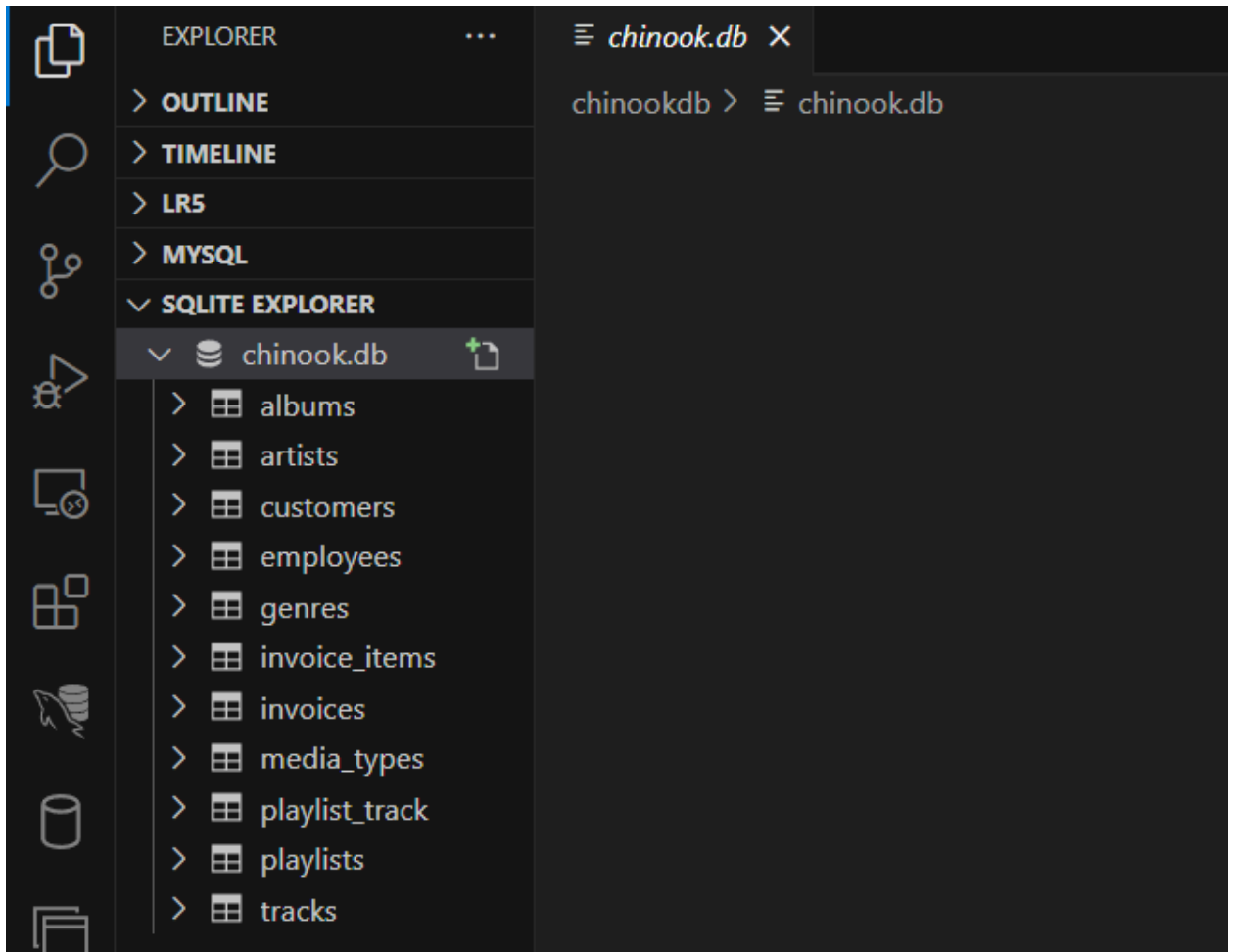
Figure 1.5 SQLite Explorer

After navigating to the SQLite Explorer, open the database and this will show you the list of tables from within the database.
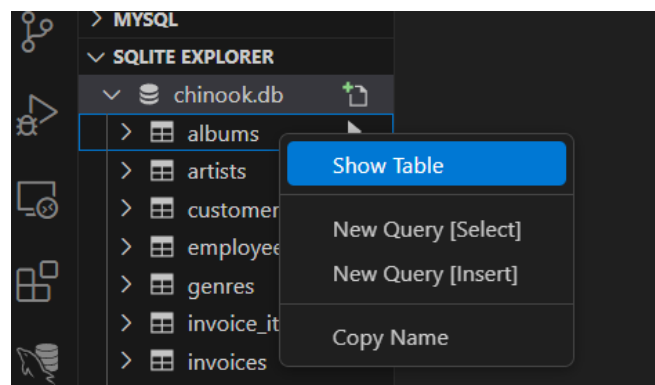

Figure 1.6 Show Table

By right clicking the table, you will be able to "Show" the table.
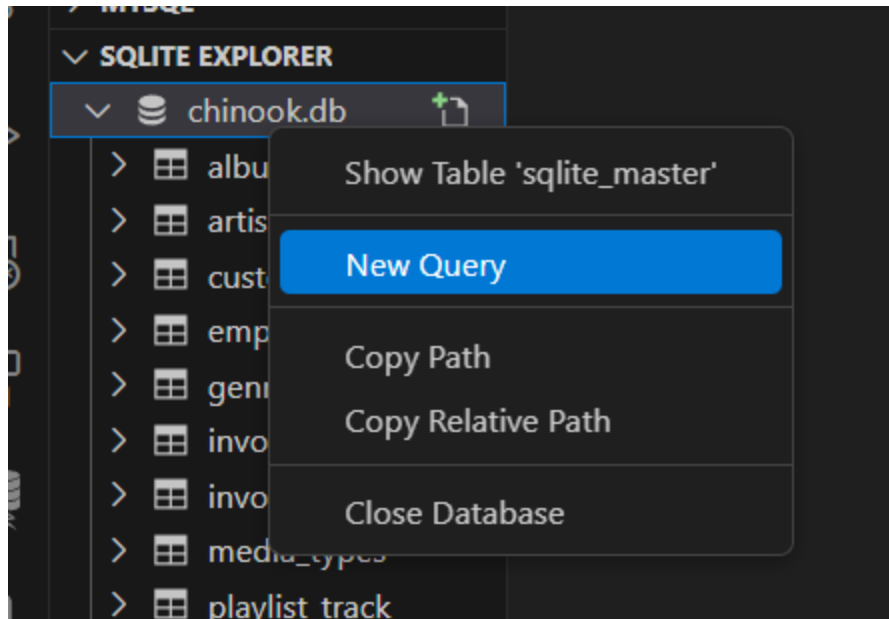
Figure 1.7 Creating a Query

Right click the database and select 'New Query' to create a query for the database.
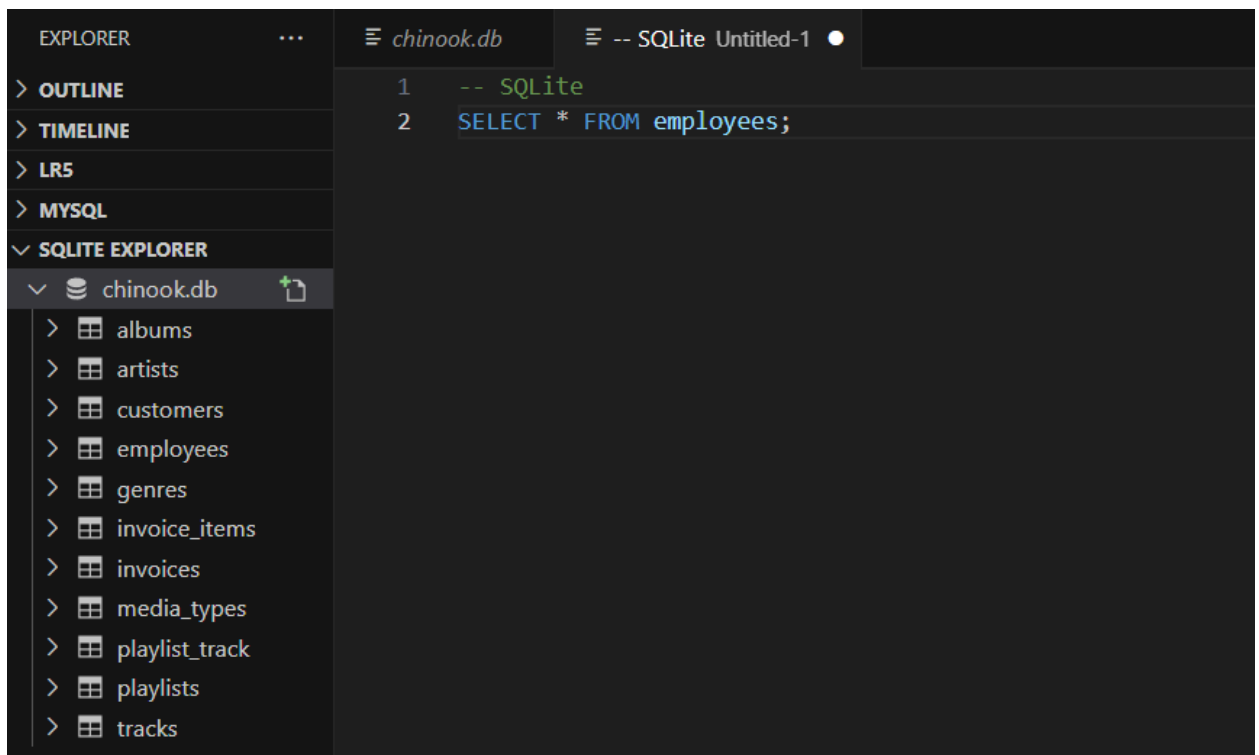

Figure 1.8 Running SQLite Commands

After writing the SQLite command, press [Ctrl] + [Shift] + [Q] to execute the SQLite command.
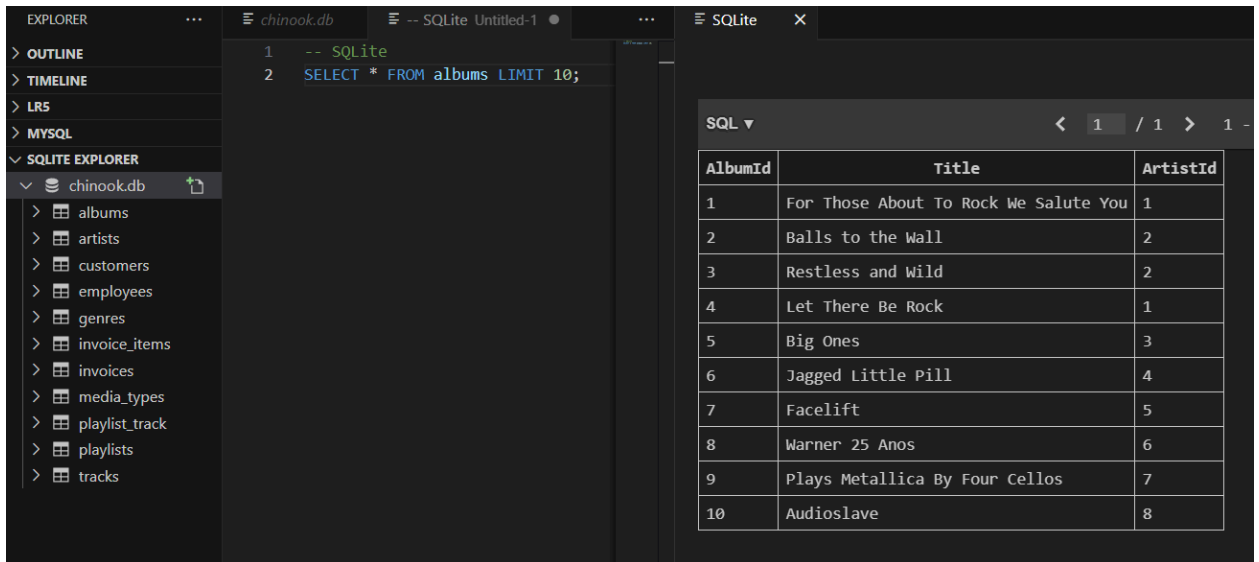
Figure 1.9 SQLite Command output for chinook database
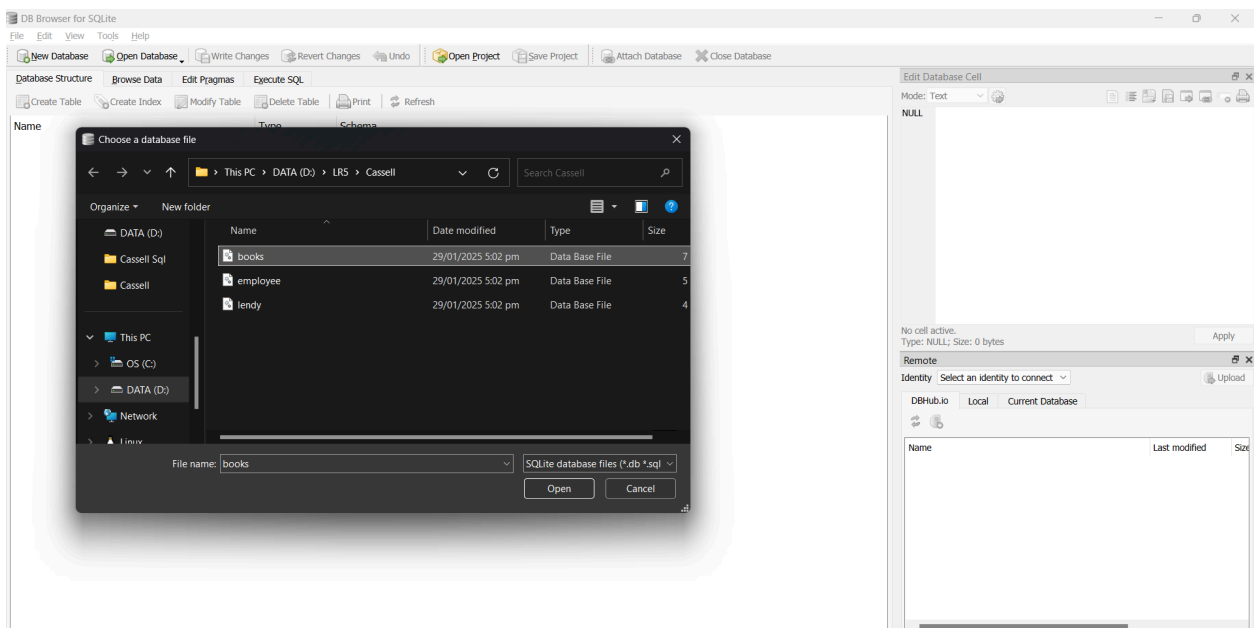
Figure 1.10 Opening the Database in DB Browser

Following the same procedure but this time for DB Browser. First, select "Open Database" and select the 'books' database then open
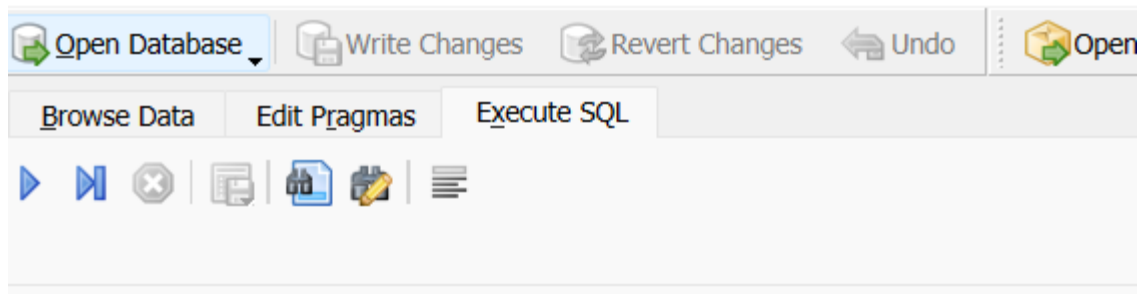
Figure 1.11 Execute SQL Tab

At the 'Execute SQL' tab start by importing all the SQL files from the CassellChap3 file directory given by clicking on 'File' and selecting all the SQL files.
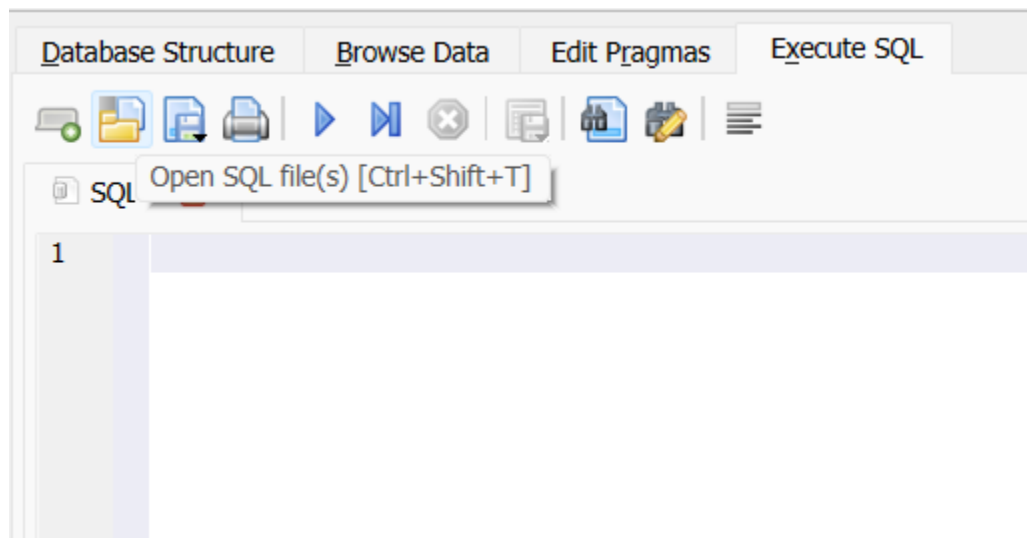
Figure 1.12 Importing SQL files

Select all of the SQL files and import it into the database.

Figure 1.13 Run the employee.sql file

Figure 1.14 Select a new database 'employee'

Figure 1.15 Run load_employee.sql

Figure 1.16 Open 'lendy' database

Figure 1.17 Run the 'lendydb.sql' file

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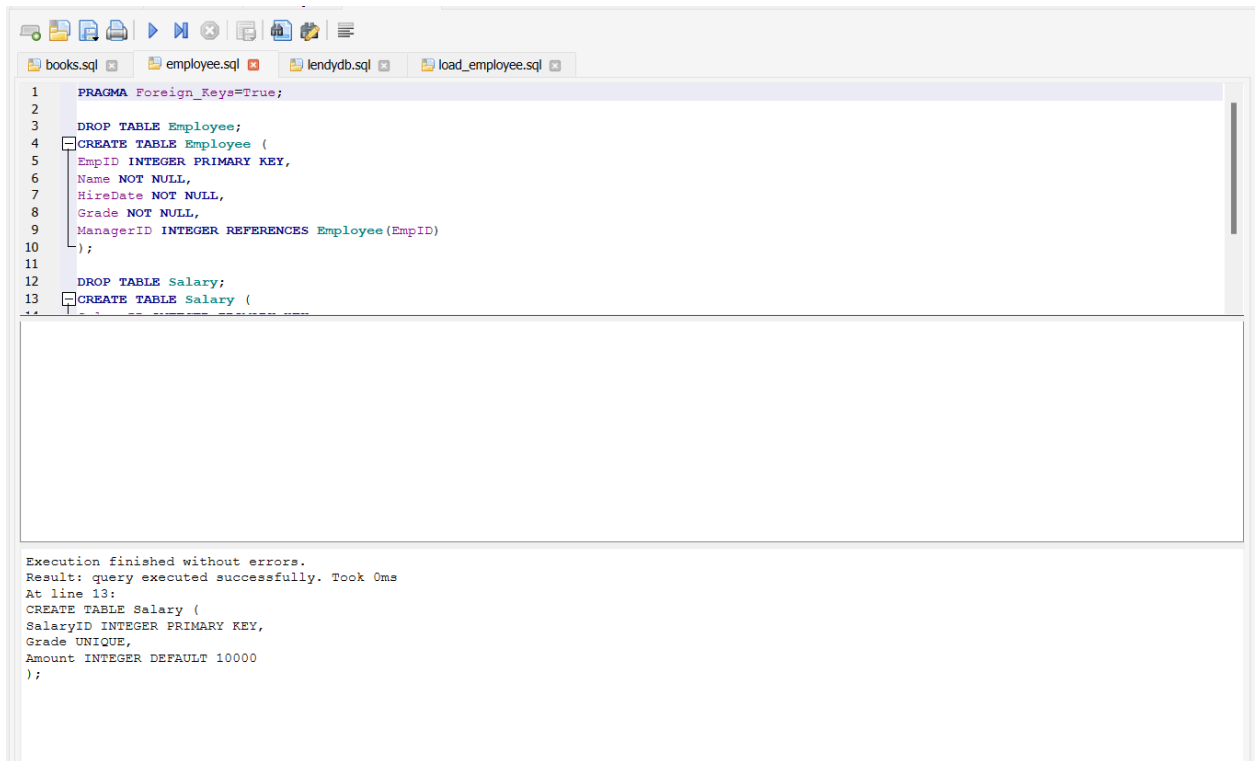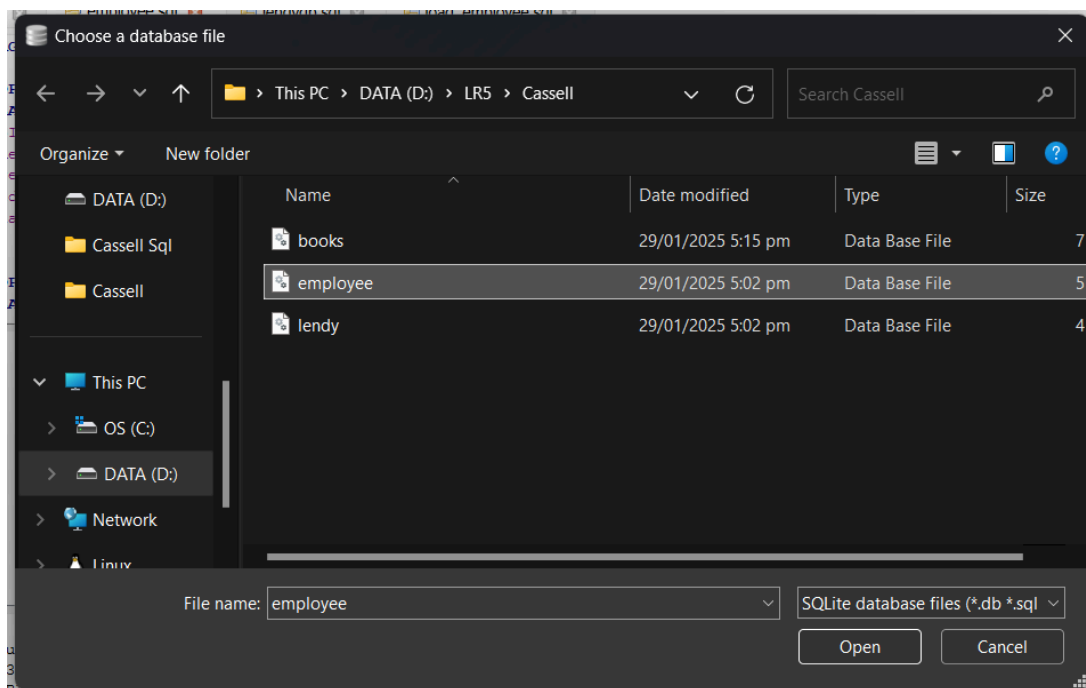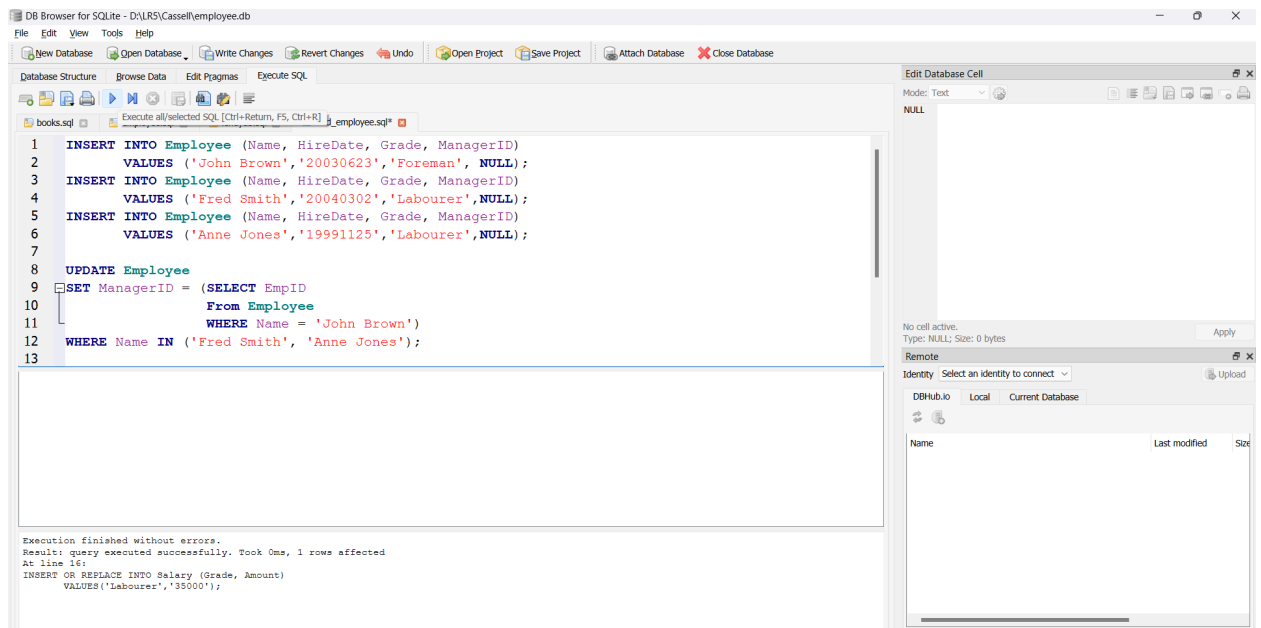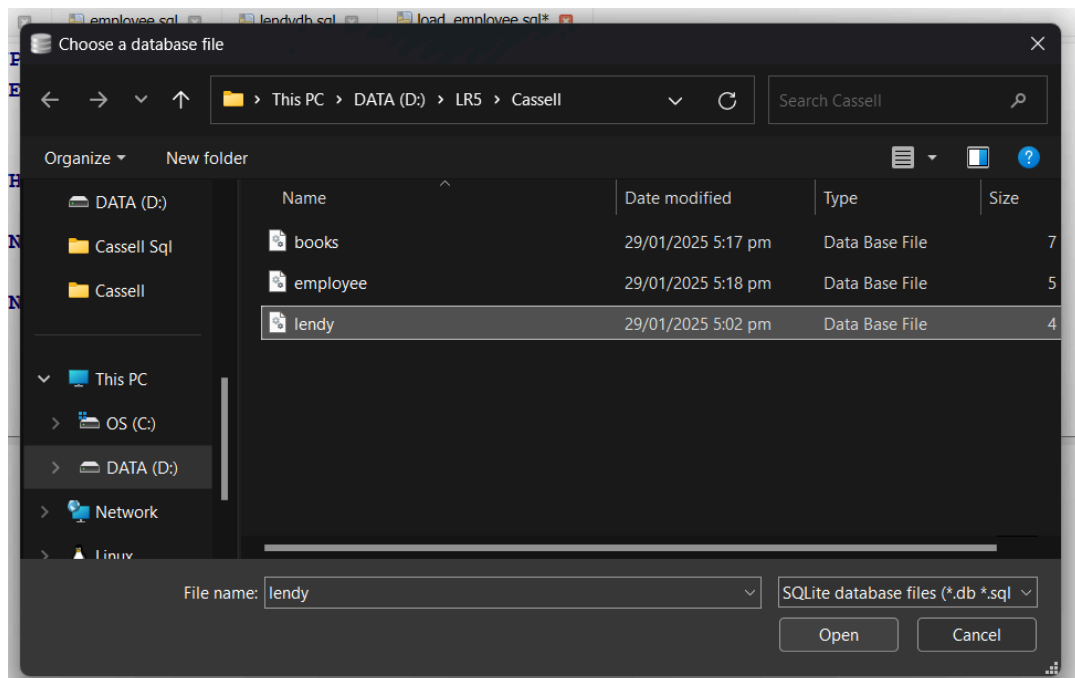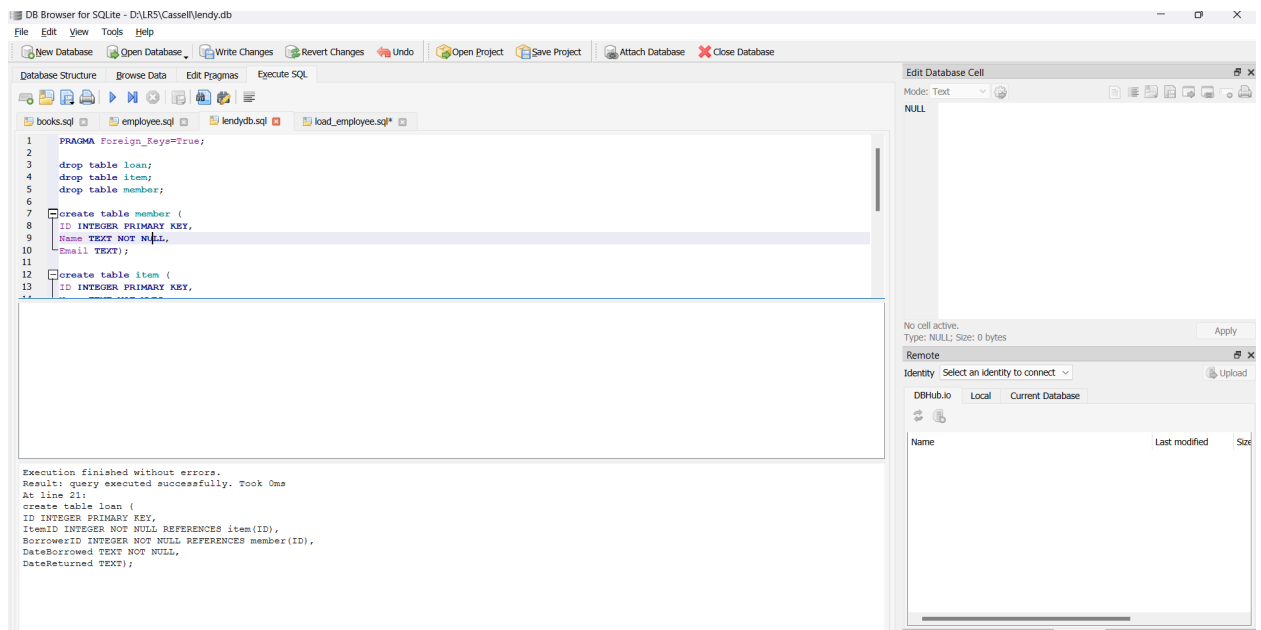 not know who the instructor for a particular class will be until the day of the class. Colonial Adventure Tours needs your help with the database design for this new venture. In each step, represent your answer using the shorthand representation and a diagram. Use crow's foot notation for the diagram. 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.

1.1 Shorthand representation

Participant (participant_id, last_name, first_name, address, city, state, postal_code, telephone, date_of_birth)

Class (class_id, class_description, max_participants, class_fee)

ClassSchedule (schedule_id, class_id [FK], class_date)

Enrollment (enrollment_id, participant_id [FK], schedule_id [FK])

Guide (guide_id, last_name, first_name)

ClassAssignment (assignment_id, guide_id [FK], schedule_id [FK])

1.2 Crow's Foot ERD

Participant
(participant_id,
last_name,
first_name,

address, city,
state,

postal_code,
telephone,
date_of_birth)

Class
(class_id,
class_description,

max_participants,

class_fee)

(M:N)

(M:1)

Enrollment
(enrollment_id,

participant_id
[FK],

schedule_id
[FK])

(M:1)

ClassSchedule
(schedule_id,

class_id [FK],

class_date,
guide_id,)

(M:1)

Guide
(guide_id,
last_name,
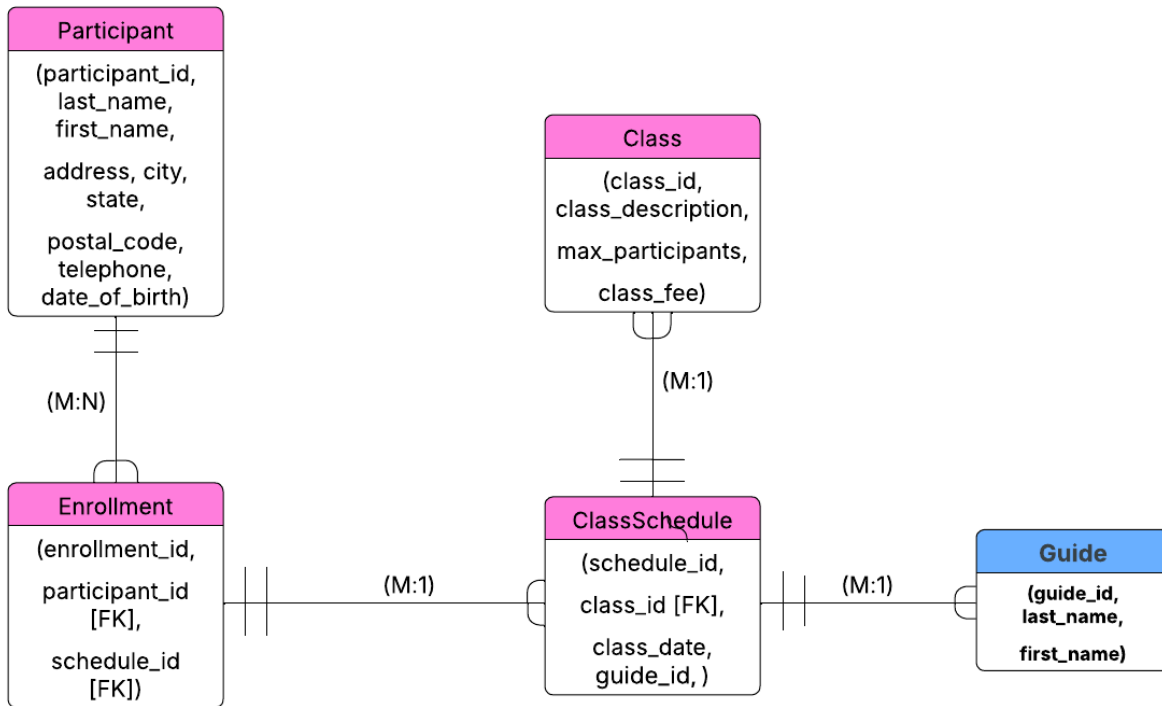
first_name)

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. 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. 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.

	RenterID	FirstName	MiddleInitial	LastName	Address	city	state	PostalCode	TelephoneNumber	Email
	Filter	Filter	Filter	Filter	Filter	Fil...	Filter	Filter	Filter	Filter
1	1	John	A	Doe	123 ...	O...	FL	32801	407-555-1234	joh...
2	2	Jane	B	Smith	456 ...	M...	FL	33101	305-555-5678	jan...
3	3	Michael	C	Brown	789 ...	T...	FL	33601	813-555-2468	mic...
4	4	Sarah	D	Johnson	321 ...	J...	FL	32201	904-555-3698	sar...
5	5	Emily	E	Davis	654 ...	N...	FL	34101	239-555-7890	emi...

Figure 2.1 Creating a table named "Renters"

	PropertyID	LocationNumber	LocationName	Address	City	State	PostalCode	CondoUnitNumber	SquareFootage	Bedrooms	Bathrooms	MaxOccupants	BaseWeeklyRate
	Filter	Filter	Filter	Filter	Filt...	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	L001	Beachfron...	100 ...	M...	FL	33101	A101	1200	2	2	6	1500.0
2	2	L002	City View...	200 ...	O...	FL	32801	B202	900	1	1	4	1200.0
3	3	L003	Golf ...	300 ...	T...	FL	33601	C303	1400	3	2	8	1800.0
4	4	L004	Lakehouse...	400 ...	N...	FL	34101	D404	1600	3	3	8	2000.0
5	5	L005	Sunset ...	500 ...	J...	FL	32201	E505	1100	2	1	5	1300.0

Figure 2.2 Creating a table named "Property"

	AgreeementID	RenterID	PropertyID	StartDate	EndDate	WeeklyRentalAmount
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	2	2024-06-01	2024-06-08	1200.0
2	2	2	4	2024-07-15	2024-07-22	2000.0
3	3	3	1	2024-08-10	2024-08-17	1500.0
4	4	4	5	2024-09-05	2024-09-12	1300.0
5	5	5	3	2024-10-20	2024-10-27	1800.0

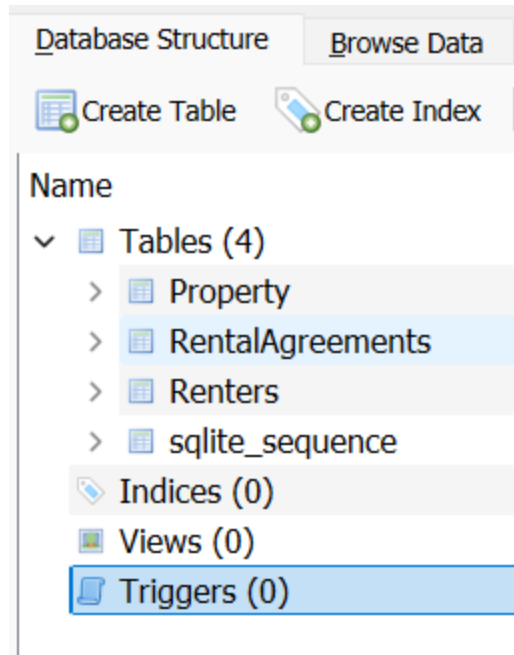**Figure 2.3** Creating a table named "RentalAgreements"

Figure 2.4 The Database Structure Solmaris Condominium Group

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.

```
1   CREATE TABLE ADVENTURE_TRIP (
2       TRIP_ID INT PRIMARY KEY,
3       TRIP_NAME VARCHAR(50),
4       START_LOCATION TEXT,
5       STATE TEXT,
6       DISTANCE INT,
7       MAX_GRP_SIZE INT,
8       TYPE TEXT,
9       SEASON TEXT
10  );
11
12  PRAGMA table_info(TRIP);
13
14  INSERT INTO ADVENTURE_TRIP (TRIP_ID, TRIP_NAME, START_LOCATION, STATE, DISTANCE, MAX_GRP_SIZE, TYPE, SEASON)
15  VALUES (45, 'Jay Peak', 'Jay', 'VT', 8, 8, 'Hiking', 'Summer');
16
17  SELECT * FROM TRIP;
18
19  DROP TABLE ADVENTURE_TRIP;
```

	TRIP_ID	TRIP_NAME	START_LOCATION	STATE	DISTANCE	MAX_GRP_SIZE	TYPE	SEASON
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

```
Execution finished without errors.
Result: 41 rows returned in 31ms
At line 29:
SELECT * FROM TRIP;
```

Figure 3.1 Creating a table named "ADVENTURE_TRIP"

Database Structure | Browse Data | Edit Pragmas | Execute SQL

ADVENTURE_TRIP | ADVENTURE_TRIP | ADVENTURE_TRIP | ADVENTURE_TRIP | ADVENTURE_TRIP | ADVE

ADVENTURE

Table: ADVENTURE_TRIP | Filter in any column

	TRIP_ID	TRIP_NAME	START_LOCATION	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 3.2 Creating and inserting

Figure 3.3 Database Tables

Figure 3.4 Colonial Adventure Tours Database 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 3.5 Colonial Adventure Tours Database CUSTOMER Table

	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
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

Figure 3.6 Colonial Adventure Tours Database RESERVATION Table

Table: TRIP_GUIDES

	TRIP_ID	GUIDE_NUM
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

Table: TRIP_GUIDES

	TRIP_ID	GUIDE_NUM
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
47	40	DH01
48	41	BR01

Figure 3.7 Colonial Adventure Tours Database TRIP_GUIDES Table