Santa Ana College
CMPR114
m10 Ch 16 Databases + class exercise #8
Objectives:

- SQLite and Python

There are 7 print screens each worth 14.2%

Project 1 (Create a database and label it as **CustomerDB** then create the **Contacts** table below with a Primary key on the ContactID column). Use the *SQLite Studio Management* software.

Figure 14-3 The Contacts table with data entered

ContactID	Name	Phone
1	Katie Allen	555-1234
2	Jill Ammons	555-5678
3	Kevin Brown	555-9012
4	Elisa Garcia	555-3456
5	Jeff Jenkins	555-7890
6	Leo Killian	555-1122
7	Marcia Potemkin	555-3344
8	Kelsey Rose	555-5566

```
Create table Contacts
(ContactID int primary key,
 Name text not null,
 Phone text not null);

insert into Contacts
values
(1,'Katie Allen','555-1234'),
(2,'Jill Ammons','555-5678'),
(3,'Kevin Brown','555-9012'),
(4,'Elsa Garcia','555-3456'),
(5,'Jeff Jenkins','555-7890'),
(6,'Leo Killian','555-1122'),
(7,'Marcia Potemkin','555-3344')
(8,'Kelsey Rose','555-5566');

select * from Contacts;
```

Santa Ana College
CMPR114
m10 Ch 16 Databases + class exercise #8
Project #2 (using Python to extract all names that end with the letter s) uses the like statement.

```
import sqlite3

con = sqlite3.connect("C:\\Users\\techi\\OneDrive - Rancho Santiago Community College Distr
cur = con.cursor()

cur.execute('''select * from Contacts where name like '%s' ''')
# Fetch the results of the SELECT statement.
results = cur.fetchall()

print ('ContactID  Name            Phone')
# Iterate over the rows and display the results.
for row in results:
 print(f'{row[0]:1}          {row[1]:1}     {row[2]:1}')

con.commit()
# Close the connection.
con.close()
```

Project #3 (Using Database Triggers)

A database trigger will execute, or fire based on the DML statements such as Delete, Insert and Update.

The company table will be used as the main table.

```
CREATE TABLE COMPANY(
  ID INT PRIMARY KEY     NOT NULL,
  NAME           TEXT    NOT NULL,
  AGE            INT     NOT NULL,
  ADDRESS        CHAR(50),
  SALARY         REAL);
```

The audit table will be used to transfer the data using the database trigger.

```
CREATE TABLE AUDIT(
   EMP_ID INT NOT NULL,
   ENTRY_DATE TEXT NOT NULL);
```

This trigger will transfer the data into the audit table from the company table after it is inserted into the company table.

```
CREATE TRIGGER audit_log AFTER INSERT
ON COMPANY
BEGIN
   INSERT INTO AUDIT(EMP_ID, ENTRY_DATE) VALUES (new.ID, datetime('now'));
END;
```

This is the syntax to enter new information.

Santa Ana College
CMPR114
m10 Ch 16 Databases + class exercise #8

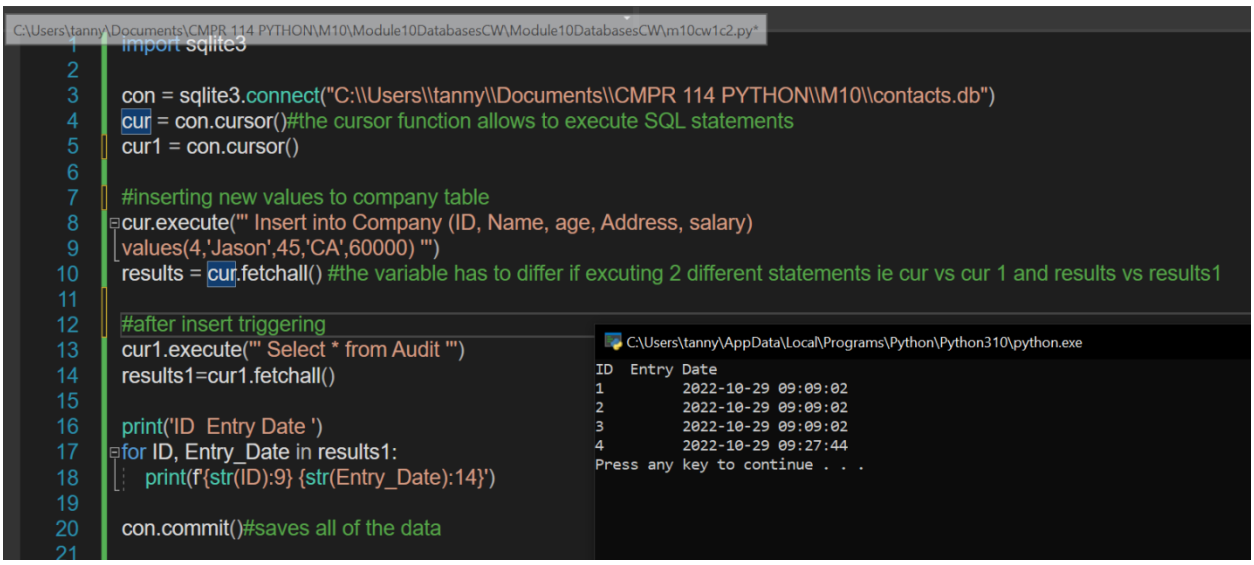
```
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Paul', 32, 'California', 20000.00 ),
(2, 'Jake', 22, 'New Jersey', 40000.00 ),
(3, 'Jill', 42, 'Texas', 50000.00 );
```

To verify the information from the audit table.

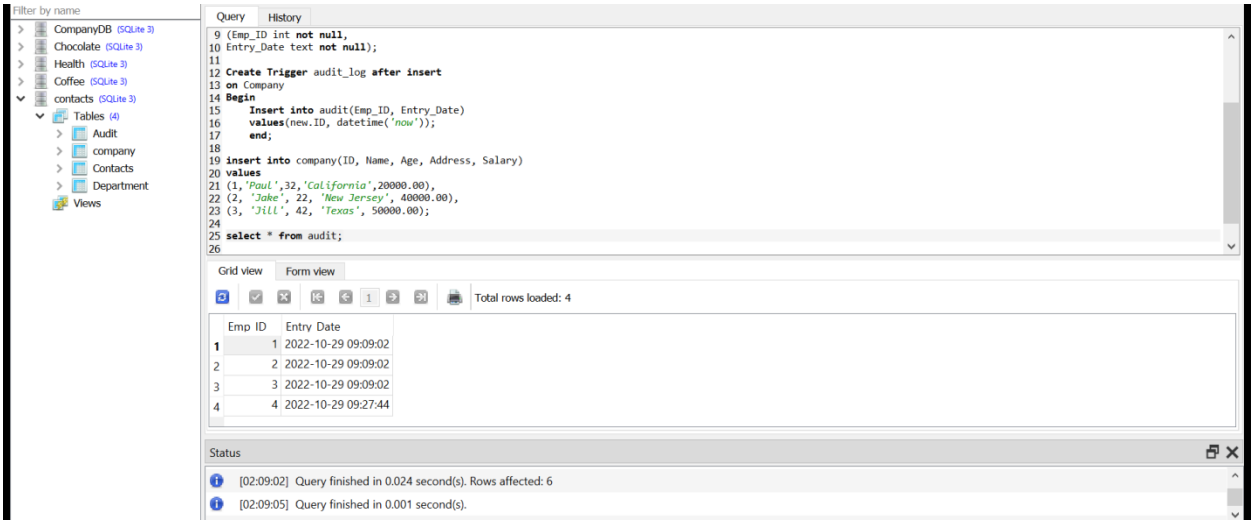
```
select * From audit;
```

Challenge Exercise #1: connect the company table to python and from the python, the application executes the trigger just created above.

#1 print screen the code with output on the python application below here



#2 on the SQLite studio, execute the select statement against the audit table to verify that the database trigger has worked. Print screen the output below here.



Santa Ana College
CMPR114
m10 Ch 16 Databases + class exercise #8

Project #4 (Using Database Views), a database view is used to view and or read the data, and NOT update, delete, or insert data.

```
Create View ContactsView
as
select * from Contacts;

select * from ContactsView;
```

Try to use the delete statement to delete a row from the view.

```
delete from ContactsView where ContactID=1;
```

Notice, we get the following error below, because a view is used for read-only purposes.

[22:05:13] Error while executing SQL query on database 'CustomersDBNew': cannot modify ContactsView because it is a view

Project #5 (Using the Update, Delete Statements, and the Alter command)

```
delete from Contacts where contactID=1;
```

Deleting from a range

```
delete from Contacts where contactID >=6;
```

Using the update statement

```
update contacts set Name = 'Jason Sim' where ContactID = 5;
```

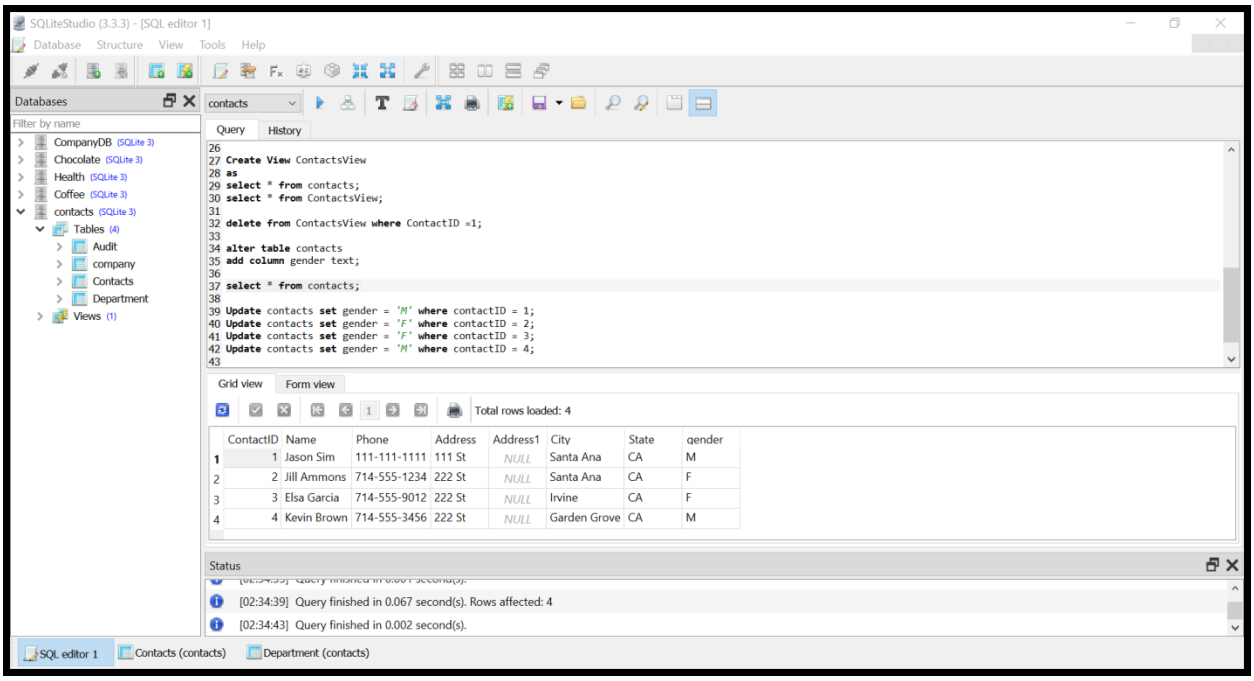
Using the Alter Statement to add a column

```
Alter table contacts
add column gender text;
```

Challenge Exercise #2: Using the update statement, update the gender column with either M or F.

#3 print screen the updated SQL code with the output below here.

Santa Ana College
CMPR114
m10 Ch 16 Databases + class exercise #8



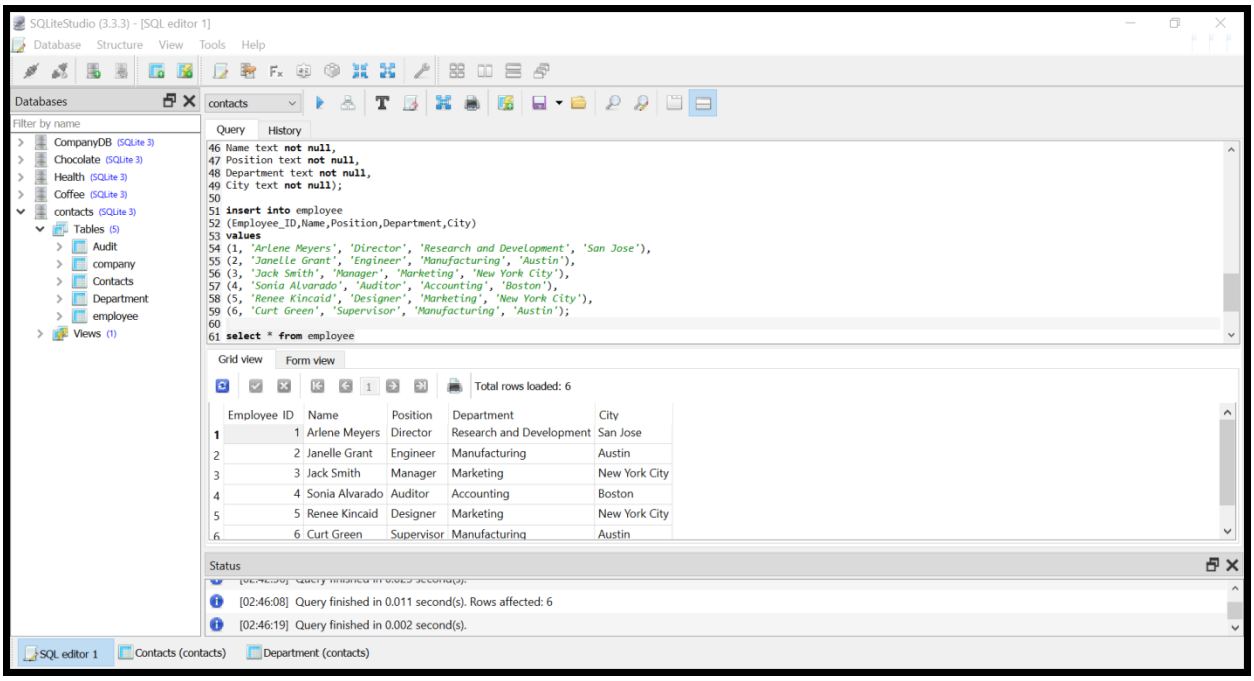
Challenge Exercise #3: Using the SQL script design the following employee data table below, and then enter the information using the SQL script.

Figure 14-6 Employee Data

Employee ID	Name	Position	Department	City
1	Arlene Meyers	Director	Research and Development	San Jose
2	Janelle Grant	Engineer	Manufacturing	Austin
3	Jack Smith	Manager	Marketing	New York City
4	Sonia Alvarado	Auditor	Accounting	Boston
5	Renee Kincaid	Designer	Marketing	New York City
6	Curt Green	Supervisor	Manufacturing	Austin

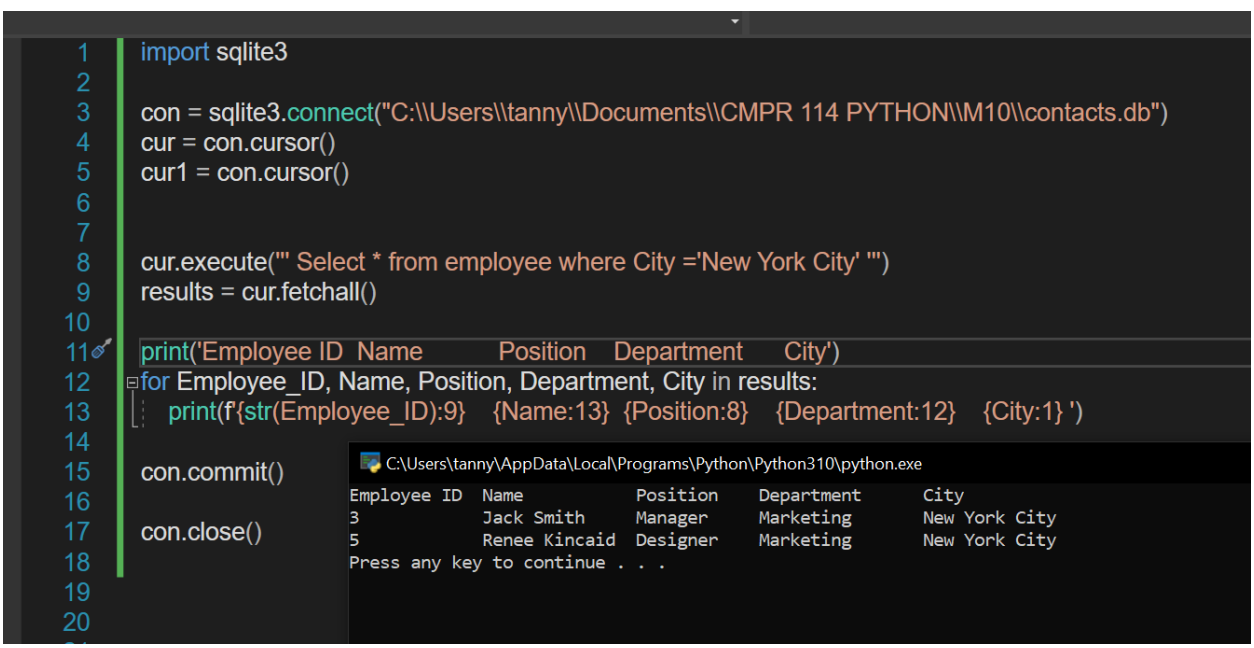
#4 print screen the create and insert SQL code with the output below here. Be sure to execute the select command to show the data in the table.

Santa Ana College
CMPR114
m10 Ch 16 Databases + class exercise #8



Challenge Exercise #4: Create a new python application and extract all employees who live in New York City. Be sure to extract all the rows with columns.

#5 Print screen below here.



Extract employee id greater than 3 and above. Be sure to extract all the rows with columns.

Santa Ana College
CMPR114
m10 Ch 16 Databases + class exercise #8

#6 Print screen below here.

```python
import sqlite3

con = sqlite3.connect("C:\\Users\\tanny\\Documents\\CMPR 114 PYTHON\\M10\\contacts.db")
cur = con.cursor()
cur1 = con.cursor()

cur.execute(''' Select * from employee where City ='New York City' ''')
results = cur.fetchall()

cur1.execute(''' Select * from employee
where Employee_ID > 3; ''')
results1 =cur1.fetchall()

print('Employee ID  Name          Position      Department      City')
#for Employee_ID, Name, Position, Department, City in results:
    #print(f'{str(Employee_ID):9}   {Name:13}  {Position:8}   {Department:12}   {City:1} ')

for Employee_ID, Name,Position, Department, City in results1:
    print(f'{str(Employee_ID):9}   {Name:14}  {Position:10}   {Department:13}   {City:1} ')

con.commit()
```

No issues found

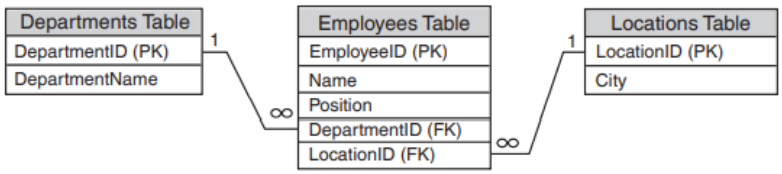
Autos Locals Watch 1

C:\Users\tanny\AppData\Local\Programs\Python\Python310\python.exe

```
Employee ID  Name            Position      Department      City
4            Sonia Alvarado  Auditor       Accounting      Boston
5            Renee Kincaid   Designer      Marketing       New York City
6            Curt Green      Supervisor    Manufacturing   Austin
Press any key to continue . . .
```

Challenge Exercise #5: Using SQL creates the relationship as shown in the diagram below.

Figure 14-7 Entity Relationship Diagram



Then enter the information as shown in the diagram below.

Figure 14-8 Contents of the `employees.db` database before a new employee is added

Employees Table

EmployeeID	Name	Position	DepartmentID	LocationID
1	Arlene Meyers	Director	4	4
2	Janelle Grant	Engineer	2	1
3	Jack Smith	Manager	3	3
4	Sonia Alvarado	Auditor	1	2
5	Renee Kincaid	Designer	3	3
6	Curt Green	Supervisor	2	1

Departments Table

DepartmentID	DepartmentName
1	Accounting
2	Manufacturing
3	Marketing
4	Research and Development

Locations Table

LocationID	City
1	Austin
2	Boston
3	New York City
4	San Jose

Santa Ana College
CMPR114
m10 Ch 16 Databases + class exercise #8
Then join the three tables and print the screen below.

#7 print screen all the SQL scripts below here.

```sql
create table Department2
(DepartmentID int primary key,
DepartmentName text not null);

create table Location
(LocationID int primary key,
City text not null);

create table Employee2
(EmployeeID int primary key,
Name text not null,
Position text not null,
DepartmentID int not null,
LocationID int not null,
Constraint DepartmentID_FK foreign key (DepartmentID)
references Department2 (DepartmentID),
Constraint LocationID_FK foreign key (LocationID)
references Location (LocationID));

insert into Employee2
(EmployeeID,Name,Position,DepartmentID,LocationID)
values
(1, 'Arlene Meyers','Director',4,4),
(2, 'Janelle Grant','Engineer',2,1),
(3, 'Jack Smith','Manager',3,3),
(4, 'Sonia Alvarado','Auditor',1,2),
(5, 'Renee Kincaid','Designer',3,3),
(6, 'Curt Green','Supervisor',2,1);

insert into Department2
(DepartmentID,DepartmentName)
values
(1,'Accounting'),
(2,'Maufacturing'),
(3,'Marketing'),
(4,'Research and Development');

insert into Location
(LocationID, City)
values
(1, 'Austin'),
(2,'Boston'),
(3,'New York City'),
(4,'San Jose');

select Location.LocationID,City, Department2.DepartmentID, DepartmentName,EmployeeID,Name,Position,Employee2.DepartmentID, Employee2.LocationID
from Location,Department2,Employee2
where Location.LocationID = Employee2.LocationID and Department2.DepartmentID = Employee2.DepartmentID;
```

Grid view | Form view

Total rows loaded: 6

	LocationI	City	DepartmentID	DepartmentName	Employee	Name	Position	DepartmentID:1	LocationID:1
1	4	San Jose	4	Research and Development	1	Arlene Meyers	Director	4	4
2	1	Austin	2	Maufacturing	2	Janelle Grant	Engineer	2	1
3	3	New York City	3	Marketing	3	Jack Smith	Manager	3	3
4	2	Boston	1	Accounting	4	Sonia Alvarado	Auditor	1	2
5	3	New York City	3	Marketing	5	Renee Kincaid	Designer	3	3
6	1	Austin	2	Maufacturing	6	Curt Green	Supervisor	2	1

Status

[03:36:43] Query finished in 0.030 second(s). Rows affected: 6

[03:42:55] Query finished in 0.001 second(s).

Submit this document to the Module 10 class exercise.