APP WEEK-4 LAB

PRACTICE (OPTIONAL):

Q1. Write a SQL lites statement to create a table names as employees including columns employee_id, first_name, last_name, email, phone_numberhire_date, job_id, salary, commission, manager_id and department_id
i) Insert values in the table and also execute the table structure
ii) Display the first name, last name of an employees whose salary is greater than 25,000

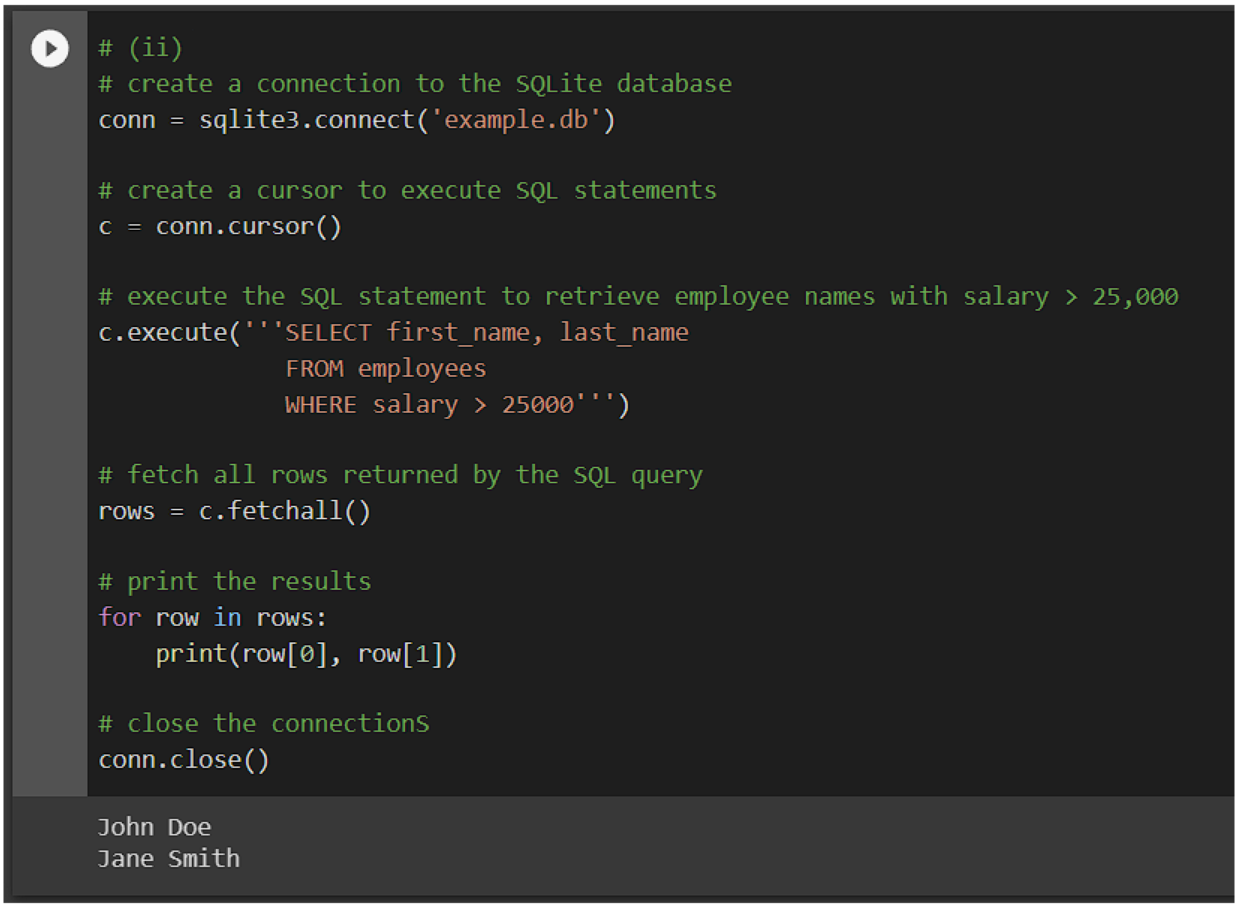
Code:
(i)

```
import sqlite3

# create a connection to the SQLite database
conn = sqlite3.connect('example.db')

# create a cursor to execute SQL statements
c = conn.cursor()

# create a table named 'employees'
c.execute('''CREATE TABLE employees
          (employee_id INTEGER PRIMARY KEY,
           first_name TEXT,
           last_name TEXT,
           email TEXT,
           phone_number TEXT,
           hire_date TEXT,
           job_id TEXT,
           salary REAL,
           commission REAL,
           manager_id INTEGER,
           department_id INTEGER)''')

# insert values into the 'employees' table
c.execute('''INSERT INTO employees
          VALUES (1, 'John', 'Doe', 'johndoe@email.com', '555-1234',
                 '2022-01-01', 'MANAGER', 50000, 0.1, NULL, 1)''')
c.execute('''INSERT INTO employees
          VALUES (2, 'Jane', 'Smith', 'janesmith@email.com', '555-5678',
                 '2022-01-02', 'SALES', 35000, 0.05, 1, 1)''')

# commit the changes and close the connection
conn.commit()
conn.close()
```

(ii)

```
# create a connection to the SQLite database
conn = sqlite3.connect('example.db')

# create a cursor to execute SQL statements
c = conn.cursor()

# execute the SQL statement to retrieve employee names with salary > 25,000
c.execute('''SELECT first_name, last_name
          FROM employees
          WHERE salary > 25000''')

# fetch all rows returned by the SQL query
rows = c.fetchall()

# print the results
for row in rows:
   print(row[0], row[1])

# close the connection
conn.close()
```

<u>SnapShot:</u>



```
# (ii)
# create a connection to the SQLite database
conn = sqlite3.connect('example.db')

# create a cursor to execute SQL statements
c = conn.cursor()

# execute the SQL statement to retrieve employee names with salary > 25,000
c.execute('''SELECT first_name, last_name
             FROM employees
             WHERE salary > 25000''')

# fetch all rows returned by the SQL query
rows = c.fetchall()

# print the results
for row in rows:
    print(row[0], row[1])

# close the connectionS
conn.close()
```

```
John Doe
Jane Smith
```

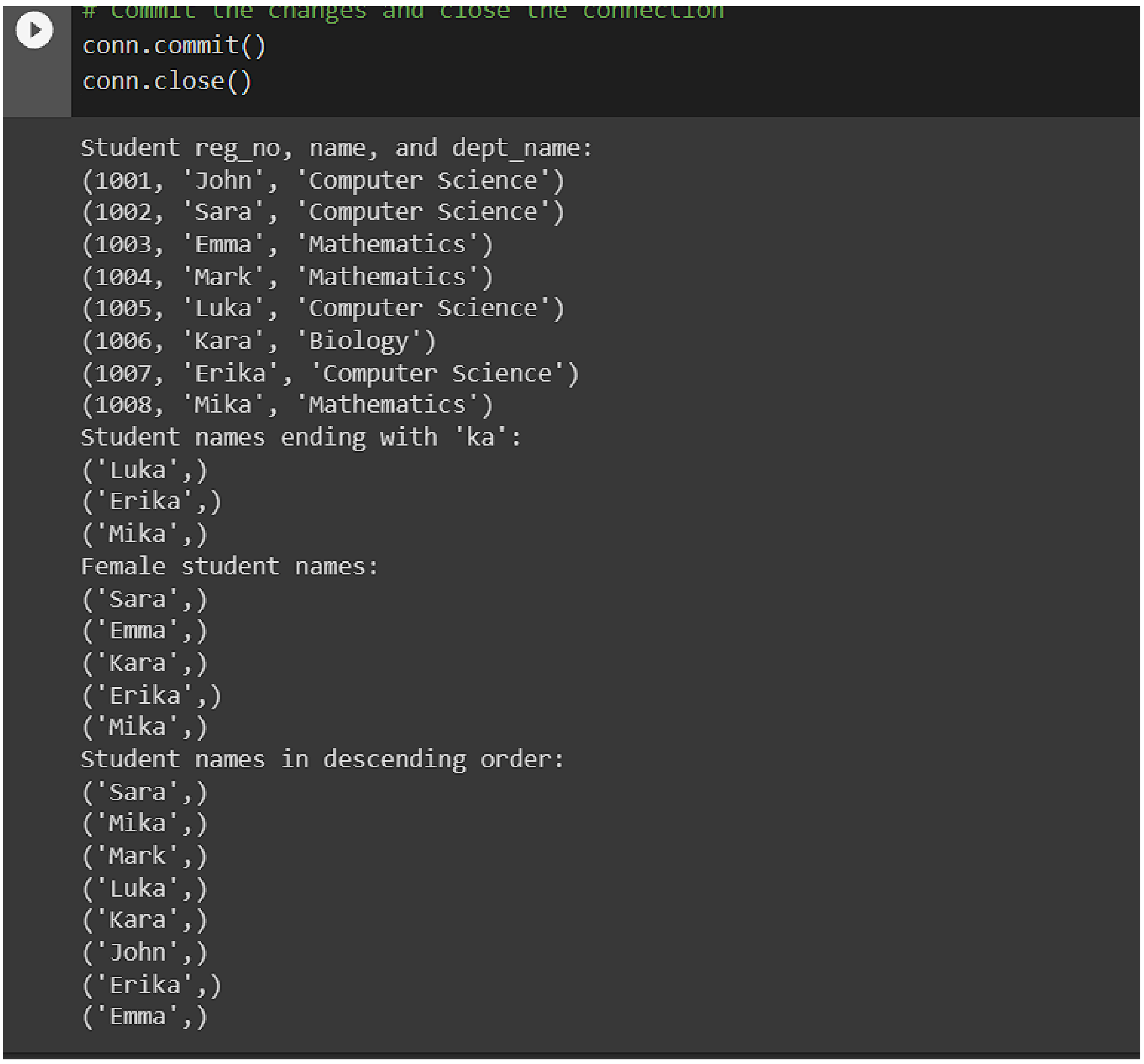
Q2.
Create a table for Student with the following fields (Reg_no,stud_name,sex, and create a table Dept with the following fields(dept_no primary key, dept_name)
a. Insert sample records and do the following
b. Display the student reg_no,name and dept_name
c. Display the student names ending with „ka"
d. Display all the female students name
e. Display the student names by descending order

Code:

```
import sqlite3

# Connect to the database
conn = sqlite3.connect('students.db')
c = conn.cursor()

# Create the Student table
c.execute('''CREATE TABLE Student (
            Reg_no INTEGER PRIMARY KEY,
            stud_name TEXT,
            sex TEXT,
            dept_no INTEGER,
            FOREIGN KEY(dept_no) REFERENCES Dept(dept_no))''')

# Create the Dept table
c.execute('''CREATE TABLE Dept (
            dept_no INTEGER PRIMARY KEY,
            dept_name TEXT)''')

# Insert sample records into the Dept table
c.execute("INSERT INTO Dept VALUES (1, 'Computer Science')")
c.execute("INSERT INTO Dept VALUES (2, 'Electrical Engineering')")
c.execute("INSERT INTO Dept VALUES (3, 'Mechanical Engineering')")

# Insert sample records into the Student table
c.execute("INSERT INTO Student VALUES (1, 'John', 'Male', 1)")
c.execute("INSERT INTO Student VALUES (2, 'Jane', 'Female', 2)")
c.execute("INSERT INTO Student VALUES (3, 'Bob', 'Male', 1)")
c.execute("INSERT INTO Student VALUES (4, 'Samantha', 'Female', 3)")
c.execute("INSERT INTO Student VALUES (5, 'Monika', 'Female', 1)")

# Display the student reg_no, name, and dept_name
c.execute("SELECT Student.Reg_no, Student.stud_name, Dept.dept_name FROM Student
INNER JOIN Dept ON Student.dept_no = Dept.dept_no")
print("Student reg_no, name, and dept_name:")
for row in c.fetchall():
   print(row)

# Display the student names ending with 'ka'
c.execute("SELECT stud_name FROM Student WHERE stud_name LIKE '%ka'")
print("Student names ending with 'ka':")
for row in c.fetchall():
   print(row)
```

```
# Display all the female students name
c.execute("SELECT stud_name FROM Student WHERE sex = 'F'")
print("Female student names:")
for row in c.fetchall():
    print(row)

# Display the student names by descending order
c.execute("SELECT stud_name FROM Student ORDER BY stud_name DESC")
print("Student names in descending order:")
for row in c.fetchall():
    print(row)

# Commit the changes and close the connection
conn.commit()
conn.close()
```

SnapShot:



```
# Commit the changes and close the connection
conn.commit()
conn.close()
```

```
Student reg_no, name, and dept_name:
(1001, 'John', 'Computer Science')
(1002, 'Sara', 'Computer Science')
(1003, 'Emma', 'Mathematics')
(1004, 'Mark', 'Mathematics')
(1005, 'Luka', 'Computer Science')
(1006, 'Kara', 'Biology')
(1007, 'Erika', 'Computer Science')
(1008, 'Mika', 'Mathematics')
Student names ending with 'ka':
('Luka',)
('Erika',)
('Mika',)
Female student names:
('Sara',)
('Emma',)
('Kara',)
('Erika',)
('Mika',)
Student names in descending order:
('Sara',)
('Mika',)
('Mark',)
('Luka',)
('Kara',)
('John',)
('Erika',)
('Emma',)
```

Graded:

Q1. Create the below table and execute the insert, update and the below select statements.

recipes.recipes
id : int(11)
name : varchar(400)
description : text
category_id : int(11)
chef_id : int(255)
created : datetime

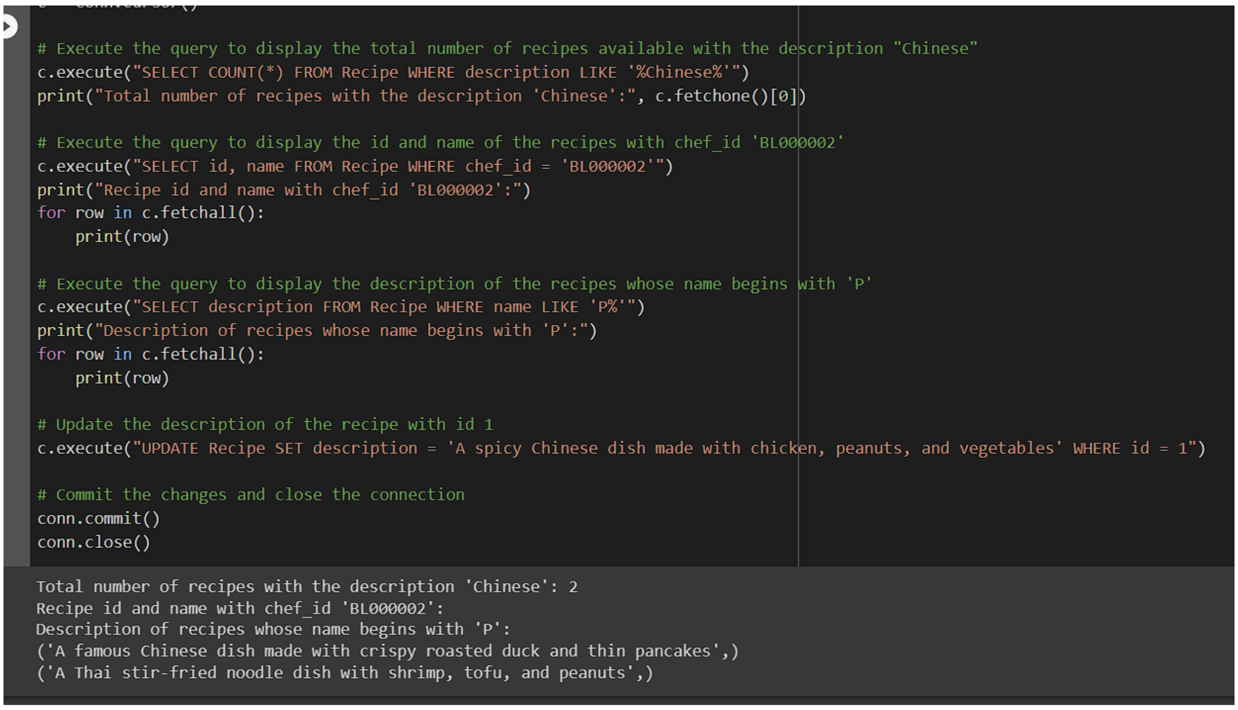
i) Write a query to display the total number of recipes available with the description "Chinese"
ii) Write a query to display the id, name of the recipes with chef_id 'BL000002'.
iii) Write a query to display the description of the recipes whose name begins with 'P'.

Code:

```
import sqlite3

# Connect to the database
conn = sqlite3.connect('recipes.db')
c = conn.cursor()

# Create the Recipe table
c.execute('''CREATE TABLE Recipe (
            id INTEGER PRIMARY KEY,
            name VARCHAR(400),
            description TEXT,
            category_id INTEGER,
            chef_id INTEGER,
            created DATETIME)''')

# Insert sample records into the Recipe table
c.execute("INSERT INTO Recipe (name, description, category_id, chef_id, created) VALUES
('Kung Pao Chicken', 'A spicy Sichuan dish made with chicken, peanuts, and vegetables', 1, 1,
'2022-01-01 10:00:00')")
c.execute("INSERT INTO Recipe (name, description, category_id, chef_id, created) VALUES
('Hot and Sour Soup', 'A classic Chinese soup made with bamboo shoots, tofu, and wood ear
mushrooms', 2, 2, '2022-01-02 11:00:00')")
c.execute("INSERT INTO Recipe (name, description, category_id, chef_id, created) VALUES
('Peking Duck', 'A famous Chinese dish made with crispy roasted duck and thin pancakes', 1,
3, '2022-01-03 12:00:00')")
c.execute("INSERT INTO Recipe (name, description, category_id, chef_id, created) VALUES
('Pad Thai', 'A Thai stir-fried noodle dish with shrimp, tofu, and peanuts', 3, 4, '2022-01-04
13:00:00')")

# Execute the query to display the total number of recipes available with the description
"Chinese"
c.execute("SELECT COUNT(*) FROM Recipe WHERE description LIKE '%Chinese%'")
print("Total number of recipes with the description 'Chinese':", c.fetchone()[0])

# Execute the query to display the id and name of the recipes with chef_id 'BL000002'
c.execute("SELECT id, name FROM Recipe WHERE chef_id = 'BL000002'")
```

```
print("Recipe id and name with chef_id 'BL000002':")
for row in c.fetchall():
    print(row)

# Execute the query to display the description of the recipes whose name begins with 'P'
c.execute("SELECT description FROM Recipe WHERE name LIKE 'P%'")
print("Description of recipes whose name begins with 'P':")
for row in c.fetchall():
    print(row)

# Update the description of the recipe with id 1
c.execute("UPDATE Recipe SET description = 'A spicy Chinese dish made with chicken,
peanuts, and vegetables' WHERE id = 1")

# Commit the changes and close the connection
conn.commit()
conn.close()
```

SnapShot:



```
# Execute the query to display the total number of recipes available with the description "Chinese"
c.execute("SELECT COUNT(*) FROM Recipe WHERE description LIKE '%Chinese%'")
print("Total number of recipes with the description 'Chinese':", c.fetchone()[0])

# Execute the query to display the id and name of the recipes with chef_id 'BL000002'
c.execute("SELECT id, name FROM Recipe WHERE chef_id = 'BL000002'")
print("Recipe id and name with chef_id 'BL000002':")
for row in c.fetchall():
    print(row)

# Execute the query to display the description of the recipes whose name begins with 'P'
c.execute("SELECT description FROM Recipe WHERE name LIKE 'P%'")
print("Description of recipes whose name begins with 'P':")
for row in c.fetchall():
    print(row)

# Update the description of the recipe with id 1
c.execute("UPDATE Recipe SET description = 'A spicy Chinese dish made with chicken, peanuts, and vegetables' WHERE id = 1")

# Commit the changes and close the connection
conn.commit()
conn.close()
```

```
Total number of recipes with the description 'Chinese': 2
Recipe id and name with chef_id 'BL000002':
Description of recipes whose name begins with 'P':
('A famous Chinese dish made with crispy roasted duck and thin pancakes',)
('A Thai stir-fried noodle dish with shrimp, tofu, and peanuts',)
```

Q2.

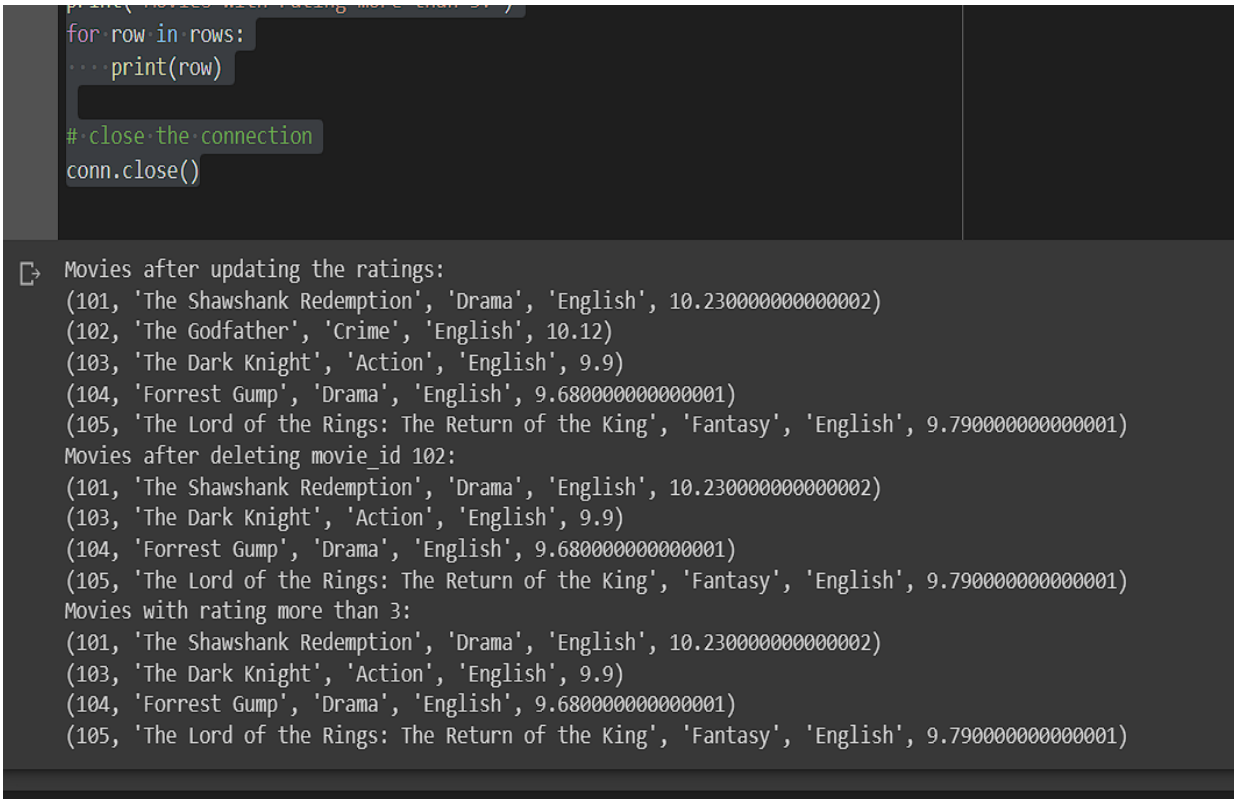
Create a table movie of the below structure and assume data types.Movie_ID, Movie_Name, Genre, Language, Rating ,Do the following queries
a. Update the movies rating by 10% and display it
b. Delete the movies with movie_id 102
c. Select movies whose rating is more than 3.

Code:

```
import sqlite3

# create a connection and cursor
conn = sqlite3.connect('movies.db')
cursor = conn.cursor()

# create the movies table
cursor.execute('''CREATE TABLE movies
            (Movie_ID INTEGER PRIMARY KEY,
             Movie_Name TEXT,
             Genre TEXT,
             Language TEXT,
             Rating FLOAT)''')

# insert some sample data into the movies table
cursor.execute("INSERT INTO movies VALUES (101, 'The Shawshank Redemption',
'Drama', 'English', 9.3)")
cursor.execute("INSERT INTO movies VALUES (102, 'The Godfather', 'Crime', 'English',
9.2)")
cursor.execute("INSERT INTO movies VALUES (103, 'The Dark Knight', 'Action', 'English',
9.0)")
cursor.execute("INSERT INTO movies VALUES (104, 'Forrest Gump', 'Drama', 'English',
8.8)")
cursor.execute("INSERT INTO movies VALUES (105, 'The Lord of the Rings: The Return of
the King', 'Fantasy', 'English', 8.9)")

# commit changes to the database
conn.commit()

# a. Update the movies rating by 10% and display it
cursor.execute("UPDATE movies SET Rating = Rating * 1.1")
cursor.execute("SELECT * FROM movies")
rows = cursor.fetchall()
print("Movies after updating the ratings:")
for row in rows:
   print(row)

# b. Delete the movies with movie_id 102
cursor.execute("DELETE FROM movies WHERE Movie_ID = 102")
cursor.execute("SELECT * FROM movies")
rows = cursor.fetchall()
print("Movies after deleting movie_id 102:")
for row in rows:
   print(row)
```

```
# c. Select movies whose rating is more than 3
cursor.execute("SELECT * FROM movies WHERE Rating > 3")
rows = cursor.fetchall()
print("Movies with rating more than 3:")
for row in rows:
    print(row)

# close the connection
conn.close()
```

<u>SnapShot:</u>



```
for row in rows:
    print(row)

# close the connection
conn.close()
```

```
Movies after updating the ratings:
(101, 'The Shawshank Redemption', 'Drama', 'English', 10.230000000000002)
(102, 'The Godfather', 'Crime', 'English', 10.12)
(103, 'The Dark Knight', 'Action', 'English', 9.9)
(104, 'Forrest Gump', 'Drama', 'English', 9.680000000000001)
(105, 'The Lord of the Rings: The Return of the King', 'Fantasy', 'English', 9.790000000000001)
Movies after deleting movie_id 102:
(101, 'The Shawshank Redemption', 'Drama', 'English', 10.230000000000002)
(103, 'The Dark Knight', 'Action', 'English', 9.9)
(104, 'Forrest Gump', 'Drama', 'English', 9.680000000000001)
(105, 'The Lord of the Rings: The Return of the King', 'Fantasy', 'English', 9.790000000000001)
Movies with rating more than 3:
(101, 'The Shawshank Redemption', 'Drama', 'English', 10.230000000000002)
(103, 'The Dark Knight', 'Action', 'English', 9.9)
(104, 'Forrest Gump', 'Drama', 'English', 9.680000000000001)
(105, 'The Lord of the Rings: The Return of the King', 'Fantasy', 'English', 9.790000000000001)
```

Q3.

Create a course database with the following fields Product(ID, Prod_name,Supplier_id,Unit_price,Package,OrderID),OrderItem(ID,Order_id, Product_id,Unit_price,Quantity) using Foreign key

a. Display the total quantity of every product in the stock

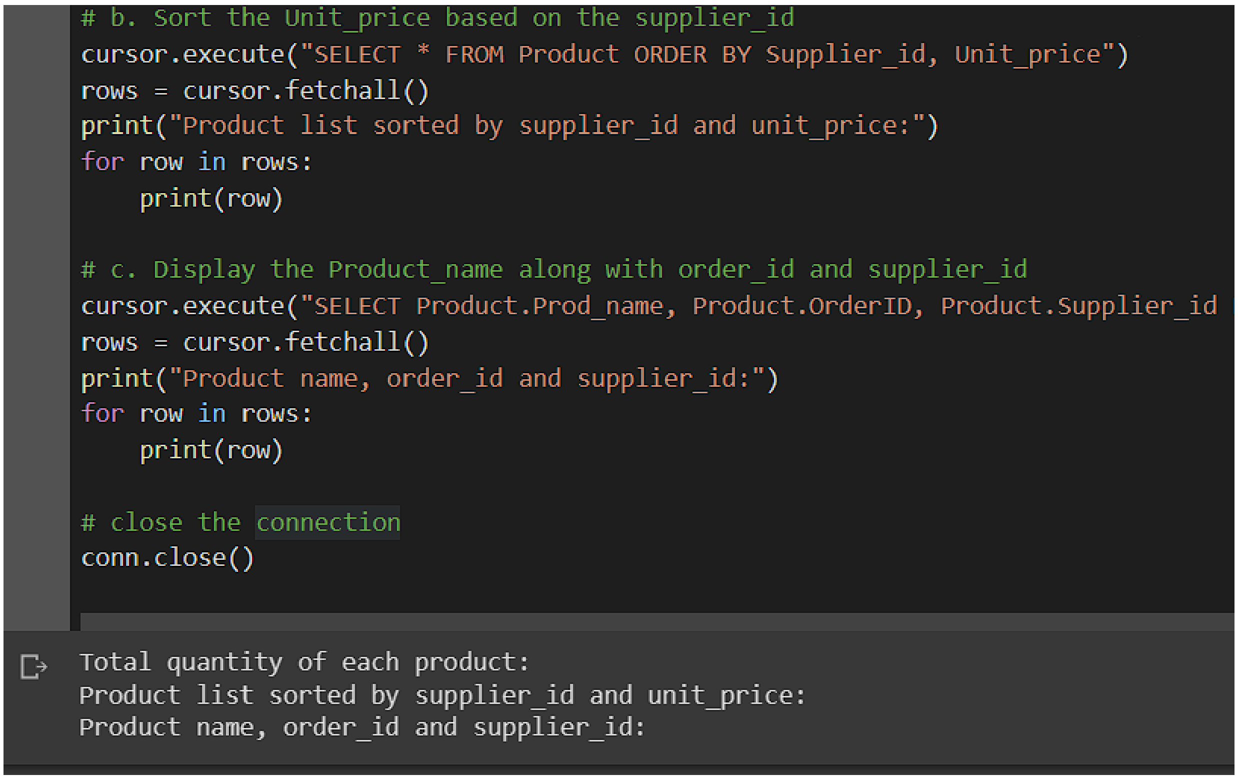
b. Sort the Unit_price based on the supplier_id

c. Display the Product_name along with order_id and supplier_id

<u>Code:</u>

```
import sqlite3

# create a connection and cursor
conn = sqlite3.connect('course.db')
cursor = conn.cursor()

# create the Product table
cursor.execute('''CREATE TABLE Product
            (ID INTEGER PRIMARY KEY,
             Prod_name TEXT,
             Supplier_id INTEGER,
             Unit_price REAL,
             Package TEXT,
             OrderID INTEGER,
             FOREIGN KEY(Supplier_id) REFERENCES OrderItem(ID),
             FOREIGN KEY(OrderID) REFERENCES OrderItem(ID))''')

# create the OrderItem table
cursor.execute('''CREATE TABLE OrderItem
            (ID INTEGER PRIMARY KEY,
             Order_id INTEGER,
             Product_id INTEGER,
             Unit_price REAL,
             Quantity INTEGER,
             FOREIGN KEY(Order_id) REFERENCES Product(OrderID),
             FOREIGN KEY(Product_id) REFERENCES Product(ID))''')

# commit changes to the database
conn.commit()

# a. Display the total quantity of every product in the stock
cursor.execute("SELECT Prod_name, SUM(Quantity) FROM OrderItem INNER JOIN
Product ON OrderItem.Product_id = Product.ID GROUP BY Prod_name")
rows = cursor.fetchall()
print("Total quantity of each product:")
for row in rows:
    print(row)

# b. Sort the Unit_price based on the supplier_id
cursor.execute("SELECT * FROM Product ORDER BY Supplier_id, Unit_price")
rows = cursor.fetchall()
print("Product list sorted by supplier_id and unit_price:")
for row in rows:
    print(row)

# c. Display the Product_name along with order_id and supplier_id
cursor.execute("SELECT Product.Prod_name, Product.OrderID, Product.Supplier_id FROM
Product JOIN OrderItem ON Product.OrderID = OrderItem.Order_id AND Product.ID =
OrderItem.Product_id")
rows = cursor.fetchall()
print("Product name, order_id and supplier_id:")
for row in rows:
    print(row)

# close the connection
conn.close()
```

SnapShot:



```
# b. Sort the Unit_price based on the supplier_id
cursor.execute("SELECT * FROM Product ORDER BY Supplier_id, Unit_price")
rows = cursor.fetchall()
print("Product list sorted by supplier_id and unit_price:")
for row in rows:
    print(row)

# c. Display the Product_name along with order_id and supplier_id
cursor.execute("SELECT Product.Prod_name, Product.OrderID, Product.Supplier_id
rows = cursor.fetchall()
print("Product name, order_id and supplier_id:")
for row in rows:
    print(row)

# close the connection
conn.close()
```

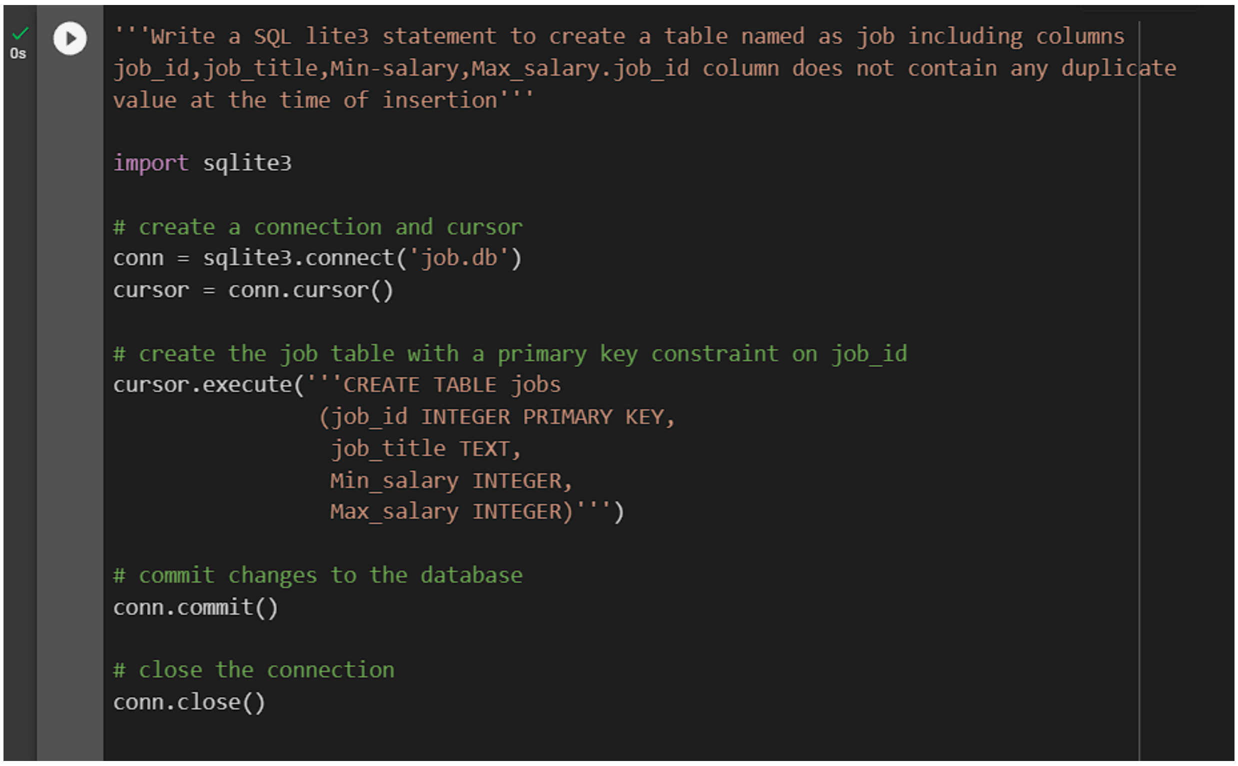
```
Total quantity of each product:
Product list sorted by supplier_id and unit_price:
Product name, order_id and supplier_id:
```

Q4.
Write a SQL lite3 statement to create a table named as job including columns job_id,job_title,Min-salary,Max_salary.job_id column does not contain any duplicate value at the time of insertion

Code:

```
import sqlite3

# create a connection and cursor
conn = sqlite3.connect('job.db')
cursor = conn.cursor()

# create the job table with a primary key constraint on job_id
cursor.execute('''CREATE TABLE job
             (job_id INTEGER PRIMARY KEY,
              job_title TEXT,
              Min_salary INTEGER,
              Max_salary INTEGER)''')

# commit changes to the database
conn.commit()

# close the connection
conn.close()
```

SnapShot:



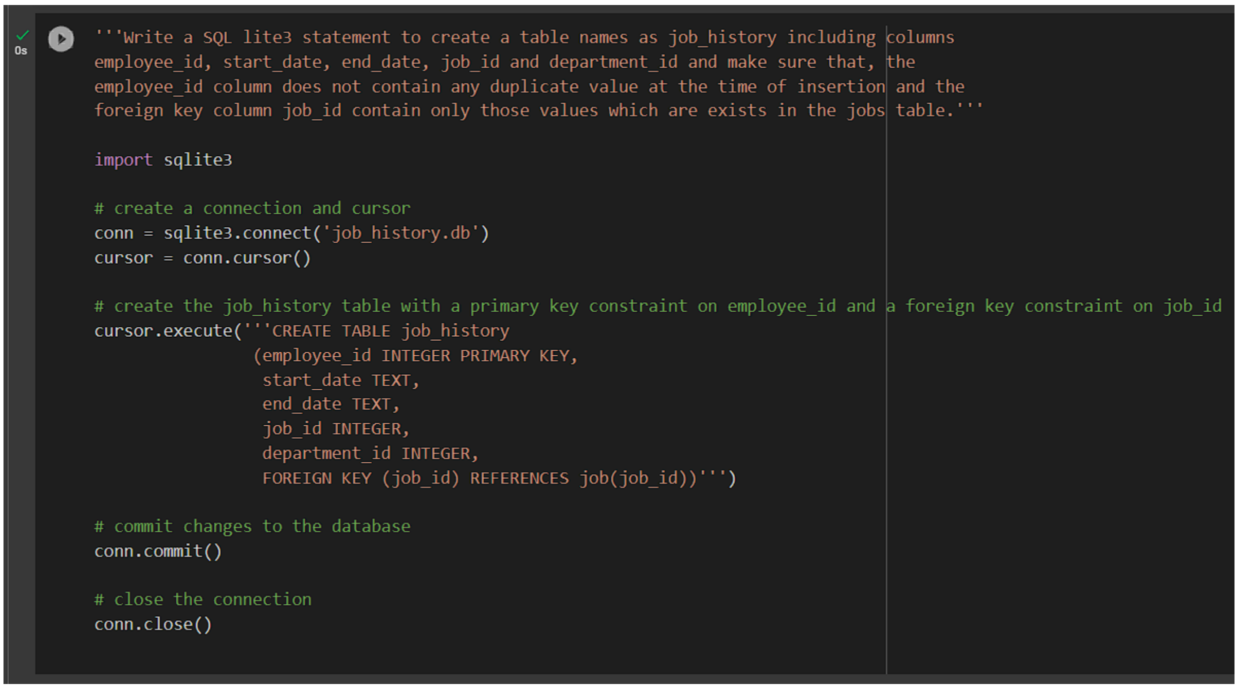
```
'''Write a SQL lite3 statement to create a table named as job including columns
job_id,job_title,Min-salary,Max_salary.job_id column does not contain any duplicate
value at the time of insertion'''

import sqlite3

# create a connection and cursor
conn = sqlite3.connect('job.db')
cursor = conn.cursor()

# create the job table with a primary key constraint on job_id
cursor.execute('''CREATE TABLE jobs
                (job_id INTEGER PRIMARY KEY,
                 job_title TEXT,
                 Min_salary INTEGER,
                 Max_salary INTEGER)''')

# commit changes to the database
conn.commit()

# close the connection
conn.close()
```

Q5.
'Write a SQL lite3 statement to create a table names as job_history including columns employee_id, start_date, end_date, job_id and department_id and make sure that, the employee_id column does not contain any duplicate value at the time of insertion and the foreign key column job_id contain only those values which are exists in the jobs table.

Code:

```
import sqlite3
# create a connection and cursor
conn = sqlite3.connect('job_history.db')
cursor = conn.cursor()
# create the job_history table with a primary key constraint on employee_id and a foreign key constraint on job_id
cursor.execute('''CREATE TABLE job_history
              (employee_id INTEGER PRIMARY KEY,
               start_date TEXT,
               end_date TEXT,
               job_id INTEGER,
               department_id INTEGER,
               FOREIGN KEY (job_id) REFERENCES job(job_id))''')
# commit changes to the database
conn.commit()
# close the connection
conn.close()
```

SnapShot:



```
'''Write a SQL lite3 statement to create a table names as job_history including columns
employee_id, start_date, end_date, job_id and department_id and make sure that, the
employee_id column does not contain any duplicate value at the time of insertion and the
foreign key column job_id contain only those values which are exists in the jobs table.'''

import sqlite3

# create a connection and cursor
conn = sqlite3.connect('job_history.db')
cursor = conn.cursor()

# create the job_history table with a primary key constraint on employee_id and a foreign key constraint on job_id
cursor.execute('''CREATE TABLE job_history
                (employee_id INTEGER PRIMARY KEY,
                 start_date TEXT,
                 end_date TEXT,
                 job_id INTEGER,
                 department_id INTEGER,
                 FOREIGN KEY (job_id) REFERENCES job(job_id))''')

# commit changes to the database
conn.commit()

# close the connection
conn.close()
```