1.WAP TO CREATE EMPLOYEE TABLE (E_ID INTEGER ,E_NAME VARCHAR(20) , E_CITY VARCHAR(20) ,E_MOB INTEGER)

```
SQL> CREATE TABLE EMPLOYEE (
E_ID        INT                 NOT NULL,
E_NAME VARCHAR (20) NOT NULL,
E_CITY    VARCHAR (20) NOT NULL,
E_MOB     INT                  NOT NULL,
PRIMARY KEY (E_ID)
);
```

2.WAP TO USED DISTINCT CLAUSE IN EMPLOYEE TABLE ON E_NAME

The distinct keyword is used in conjunction with the select keyword. It is helpful when there is a need to avoid duplicate values present in any specific columns/table. When we use distinct keywords only the unique values are fetched.

```
SQL> CREATE TABLE EMPLOYEE (
E_ID        INT                 NOT NULL,
E_NAME VARCHAR (20) NOT NULL,
E_CITY    VARCHAR (20) NOT NULL,
E_MOB     INT(10)           NOT NULL,
PRIMARY KEY (E_ID)
);
```

```
SELECT DISTINCT E_NAME FROM EMPLOYEE;
```

3.WAP TO IMPLEMENT WHERE CLAUSE IN EMPLOYEE TABLE

```
INSERT INTO EMPLOYEE (E_ID,E_NAME,E_CITY,E_MOB)
VALUES (1,'Shubham','SURAT','9090909090');
SELECT * FROM EMPLOYEE WHERE E_CITY=’SURAT’;
```

4. WAP TO PUT BETWEEN CLAUSE ON EMPLOYEE TABLE

SELECT * FROM EMPLOYEE WHERE E_NAME BETWEEN 'Shubham' AND 'Rakesh';

5. WAP TO IMPLIMENT IN AND NOT IN CLAUSE ON EMPLOYEE TABLE

SELECT E_NAME FROM EMPLOYEE WHERE E_CITY NOT IN ('VADODARA', 'VAPI');

6.WAP TO USE LIKE CLAUSE ON EMPLOYEE TABLE IN NAME OTHERWISE ON ADDRESS

SELECT * FROM EMPLOYEE WHERE E_NAME LIKE 'Shubham';

7.WAP TO IMPLIMENT UNION ON EMPLOYEE AND STUDENT TABLE

SELECT E_ID FROM EMPLOYEE UNION SELECT E_ID FROM STUDENT;

8. WAP TO IMPLIMENT LIMIT CLAUSE ON EMPLOYEE TABLE

SELECT * FROM EMPLOYEE LIMIT 3;

9.WAP TO USE GROUP BY CLAUSE ON EMPLOYE AND MAKE GROUP ON E_NAME ON EMPLOYEE

SELECT E_NAME SUM(SALARY) FROM EMPLOYEE GROUP BY E_NAME;

10. WAP TO IMPLIMENT HAVING CLAUSE WITH GROUP BY CLAUSE

SELECT COUNT(EMP_ID), E_CITY FROM EMPLOYEE GROUP BY E_CITY HAVING COUNT(EMP_ID) > 5;

11.WAP TO PUT CASE ON EMPLOYEE TABLE IN E_MOBILE PUT CASE WHICH IS NOT GREATER THAN 10 AND NOT LESS THAN 10

SELECT E_MOB FROM EMPLOYEE WHERE E_MOB NOT BETWEEN E_MOB >10 AND E_MOB <10;

12. WAP TO PERFORM JOINS IN SQLITE SUCH AS INNER , OUTER (LEFT OUTER,RIGHT OUTER,FULL OUTER), CROSS JOIN

INNER JOIN

SELECT ProductID, ProductName, CategoryName FROM Products

INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;

LEFT OUTER JOIN

SELECT Customers.CustomerName, Orders.OrderID FROM CustomersLEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID ORDER BY Customers.CustomerName;

RIGHT OUTER JOIN

SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;

FULL OUTER JOIN

SELECT Customers.CustomerName, Orders.OrderID FROM Customers FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID ORDER BY Customers.CustomerName;

CROSS JOIN

SELECT Customers.CustomerName, Orders.OrderID FROM Customers CROSS JOIN Orders;

13. WAP TO CREATE TRIGGER (BEFORE | AFTER] [INSERT | UPDATE |DELETE] ALL TASK PERFORM AND MAKE OUTPUT

```
CREATE TABLE Employee
(
Id INT PRIMARY KEY,
Name VARCHAR(45),
Salary INT,
Gender VARCHAR(12),
 DepartmentId INT
)
INSERT INTO Employee VALUES (1,'Steffan', 82000, 'Male', 3),
(2,'Amelie', 52000, 'Female', 2),
(3,'Antonio', 25000, 'male', 1),
(4,'Marco', 47000, 'Male', 2),
(5,'Eliana', 46000, 'Female', 3)
CREATE TABLE Employee_Audit_Test
(
Id int IDENTITY,
Audit_Action text
)
```

```
CREATE TRIGGER trInsertEmployee
ON Employee
FOR INSERT
AS
BEGIN
Declare @Id int
SELECT @Id = Id from inserted
INSERT INTO Employee_Audit_Test
 VALUES ('New employee with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is added at ' +
CAST(Getdate() AS VARCHAR(22)))
END
INSERT INTO Employee VALUES (6,'Peter', 62000, 'Male', 3)
```

If no error is found, execute the SELECT statement to check the audit records. We will get the output as follows:

Id	Audit_Action
1	New employee with Id = 6 is added at Mar 24 2021 2:08PM

14. WAP TO DROP TRIGGER WHICH IS EXIST

```
DROP TRIGGER IF EXISTS trInsertEmployee ;
```

15. WAP TO (.OUTPUT) STORE AND BACKUP DATABSE INTO FILE BY .DUMP COMMAND

sqlite> .output c:/sqlite/chinook.sql

sqlite> .dump

sqlite> .exit

sqlite> .output c:/sqlite/chinook_structure.sql

sqlite> .schema

sqlite> .quit

sqlite> .mode insert

sqlite> .output data.sql

sqlite> select * from artists;

16. WAP TO (.OUTPUT) STORE AND BACKUP PERTICULAR EMPLOYEE TABLE INTO PERTICULAR FILE.

sqlite> .output d:/sqlite/backup/test1.txt

sqlite> SELECT * FROM doctors;

Sample Output:

sqlite> .print "This command is used to print the text."

This command is used to print the text.

17. WAP TO IMPORT CSV FILE AND TRANSFORM CSV FILE INTO A PERTICULAR TABLE.

Download the city.csv file

To import the c:\sqlite\city.csv file into the cities table:

sqlite>.import c:/sqlite/city.csv cities

```
sqlite> .schema cities
CREATE TABLE cities(
  "name" TEXT,
  "population" TEXT
);

SELECT
   name,
   population
FROM
   Cities;
```

18. WAP TO EXPORT TABLE INTO CSV FILE.

```
sqlite> .header on
sqlite> .mode csv
sqlite> .output Employee.csv
sqlite> SELECT * FROM emp_master;
sqlite> .quit
```

19.WAP TO CREATE TWO MODULE AND USED OTHER MODULE IN ONE MODULE USING TWO METHOD(IMPORT AND FROM IMPORT)

```
.mode csv
.import /Users/javatpoint1/Desktop/sqlite/student.csv EMPLOYEE
```

20. WAP TO CONNECTING THE PYTHON WITH SQLITE BY USING CONNECTION QUERY AND MAKE STUDENT DATABSE(STUDENT.DB)

```
import sqlite3

try:
    sqliteConnection = sqlite3.connect('STUDENT.db')
    cursor = sqliteConnection.cursor()
    print('DB Init')

    query = 'select sqlite_version();'
    cursor.execute(query)

    result = cursor.fetchall()
    print('SQLite Version is {}'.format(result))

    cursor.close()

except sqlite3.Error as error:
    print('Error occurred - ', error)

finally:
    if sqliteConnection:
        sqliteConnection.close()
        print('SQLite Connection closed')
```

21.WAP TO CREATE TABLE USING PYTHON WITH CONN.EXECUTE() METHOD. (THE TABLE ID EMPLOYEE WHICH CONTAIN FOLLOWING FIELD E_ID,E_NAME,E_CITY,E_MOBNO,E-DESIGNATION) PUT PRIMARY KEY IN E_ID

```
import sqlite3

conn = sqlite3.connect('EMPLOYEE.db')

cursor = conn.cursor()

cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")




sql ='''CREATE TABLE EMPLOYEE(

   E_ID INT PRIMARY KEY,

   E_NAME CHAR(20) NOT NULL,

   E_CITY  CHAR(20),

   MOBNO INT(10),

   E-DESIGNATION CHAR(20)

)'''

cursor.execute(sql)

print("Table created successfully.....")

conn.commit()

conn.close()
```

22.WAP TO INSERT RECORD INTO EMPLOYEE TABLE USING EXECUTE()

METHOD.(THE VALUE IS STATIC).

```
import sqlite3
conn = sqlite3.connect('geeks2.db')
cursor = conn.cursor()


table ="""CREATE TABLE STUDENT(NAME VARCHAR(255), CLASS VARCHAR(255),
SECTION VARCHAR(255));"""
cursor.execute(table)

cursor.execute("'INSERT INTO STUDENT VALUES ('Raju', '7th', 'A')"')
cursor.execute("'INSERT INTO STUDENT VALUES ('Shyam', '8th', 'B')"')
cursor.execute("'INSERT INTO STUDENT VALUES ('Baburao', '9th', 'C')"')

print("Data Inserted in the table: ")
data=cursor.execute("'SELECT * FROM STUDENT"')
for row in data:
   print(row)

conn.commit()

conn.close()
```

23.WAP TO INSERT RECORD INTO EMPLOYEE TABLE USING EXECUTE()
METHOD. (THE VALUE IS NOT STATIC).AT LEAST 10 RECOD(5 RECORD INSERT BY STATIC AND 5 RECORD INSERT BY DYNAMIC)

```
import sqlite3

def insertVaribleIntoTable(id, name, email, joinDate, salary):

   try:

      sqliteConnection = sqlite3.connect('SQLite_Python.db')

      cursor = sqliteConnection.cursor()

      print("Connected to SQLite")
```

```
        sqlite_insert_with_param = """INSERT INTO SqliteDb_developers
                          (id, name, email, joining_date, salary)
                          VALUES (?, ?, ?, ?, ?);"""

        data_tuple = (id, name, email, joinDate, salary)
        cursor.execute(sqlite_insert_with_param, data_tuple)
        sqliteConnection.commit()
        print("Python Variables inserted successfully into SqliteDb_developers table")

            cursor.close()

    except sqlite3.Error as error:
        print("Failed to insert Python variable into sqlite table", error)
    finally:
        if sqliteConnection:
            sqliteConnection.close()
            print("The SQLite connection is closed")

insertVaribleIntoTable(2, 'Joe', 'joe@pynative.com', '2019-05-19', 9000)
insertVaribleIntoTable(3, 'Ben', 'ben@pynative.com', '2019-02-23', 9500)
```

24. WAP TO DELETE RECORD FROM THE EMPLOYEE TABLE BY EXECUTE() METHOD(THE DELETE VALUE IS STATIC) DELETE RECORD IS 101

```
import sqlite3
conn = sqlite3.connect('example.db')
cursor = conn.cursor()

print("Contents of the table: ")
cursor.execute("'SELECT * from EMPLOYEE"')
print(cursor.fetchall())

cursor.execute("'DELETE FROM EMPLOYEE WHERE E_ID=101"')

print("Contents of the table after delete operation ")
cursor.execute("SELECT * from EMPLOYEE")
print(cursor.fetchall())

conn.commit()

conn.close()
```

25.WAP TO DELETE RECORD FROM THE EMPLOYEE TABLE BY EXECUTE() METHOD(THE DELETE VALUE IS DYNAMIC) DELETE RECORD IS 102

```
import sqlite3

def deleteSqliteRecord(id):
    try:
        sqliteConnection = sqlite3.connect('SQLite_Python.db')
        cursor = sqliteConnection.cursor()
        print("Connected to SQLite")

        sql_update_query = """DELETE from SqliteDb_developers where id = ?"""
        cursor.execute(sql_update_query, (id,))
        sqliteConnection.commit()
```

```
        print("Record deleted successfully")

        cursor.close()

    except sqlite3.Error as error:
        print("Failed to delete reocord from a sqlite table", error)
    finally:
        if sqliteConnection:
            sqliteConnection.close()
            print("sqlite connection is closed")

deleteSqliteRecord(5)
```

26.WAP TO DELETE RECORD OF E_NAME BY EXECUTE() METHOD AND DISPLAY THE TOTAL DELETE RECORD IN NUMBER(TOTAL_CHANGE) (DELETE VALUE WHICH IS DUPLICAT)

```
import sqlite3
conn = sqlite3.connect('example.db')
cursor = conn.cursor()

print("Contents of the table: ")
cursor.execute("'SELECT * from EMPLOYEE"')
print(cursor.fetchall())

cursor.execute("'DELETE FROM EMPLOYEE WHERE E_NAME=’RAKESH’"')

print("Contents of the table after delete operation ")
cursor.execute("SELECT * from EMPLOYEE")
print(cursor.fetchall())

conn.commit()

conn.close()
```

27.WAP TO DISPLAY THE ALL RECORD OF THE EMPLOYEE TABLE BY USING THE SELECT QUERY.

```
import sqlite3
connection_obj = sqlite3.connect('geek.db')
cursor_obj = connection_obj.cursor()

statement = '''SELECT * FROM GEEK'''

cursor_obj.execute(statement)

print("All the data")
output = cursor_obj.fetchall()
for row in output:
 print(row)

connection_obj.commit()
connection_obj.close()
```

28. WAP TO FATCH ONE RECORD FROM THE EMPLOYEE TABLE.(DISPLAY AT LEAST 8 RECORD FROM THE DATABSE)

```
import sqlite3
connection_obj = sqlite3.connect('geek.db')
cursor_obj = connection_obj.cursor()

statement = '''SELECT * FROM GEEK WHERE E_ID=101’’’;

cursor_obj.execute(statement)

print("All the data")
output = cursor_obj.fetchall()
for row in output:
 print(row)

connection_obj.commit()
connection_obj.close()
```

29.WAP TO FATCH ALL RECORD FROM THE EMPLOYE TABLE.(DISPLAY ALL RECORD)

```
import sqlite3
connection_obj = sqlite3.connect('geek.db')
cursor_obj = connection_obj.cursor()

statement = '''SELECT * FROM GEEK'''

cursor_obj.execute(statement)

print("All the data")
output = cursor_obj.fetchall()
for row in output:
 print(row)

connection_obj.commit()
connection_obj.close()
```

30.WAP TO UPDATE RECORD BY USING EXECUTE() METHOD AND DISPLAY THE TOTAL CHANGES IN THE DATABSE IN NUMBER.

```
import sqlite3
conn = sqlite3.connect('example.db')
cursor = conn.cursor()


print("Contents of the table: ")
cursor.execute('''SELECT * from EMPLOYEE''')
print(cursor.fetchall())


cursor.execute('''UPDATE EMPLOYEE SET E_CITY='SURAT' WHERE E_ID=101''')


print("Contents of the table after update operation ")
cursor.execute("SELECT * from EMPLOYEE")
print(cursor.fetchall())


conn.commit()


conn.close()
```