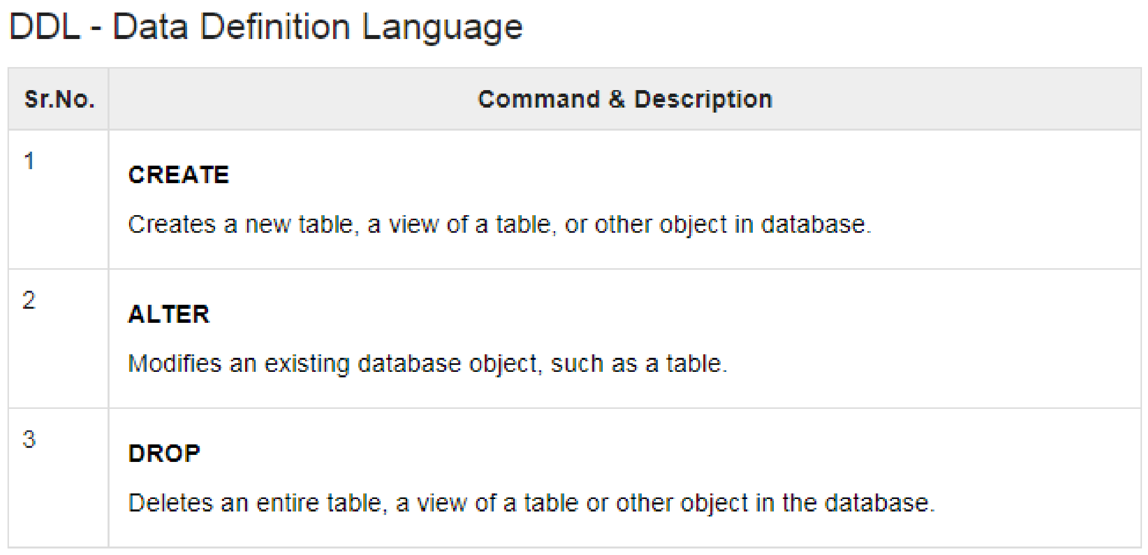
SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system. SQLite accesses its storage files directly.The sqlite3 module was written by **Gerhard Häring.**

SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).The important point to be noted is that SQLite is **case insensitive**

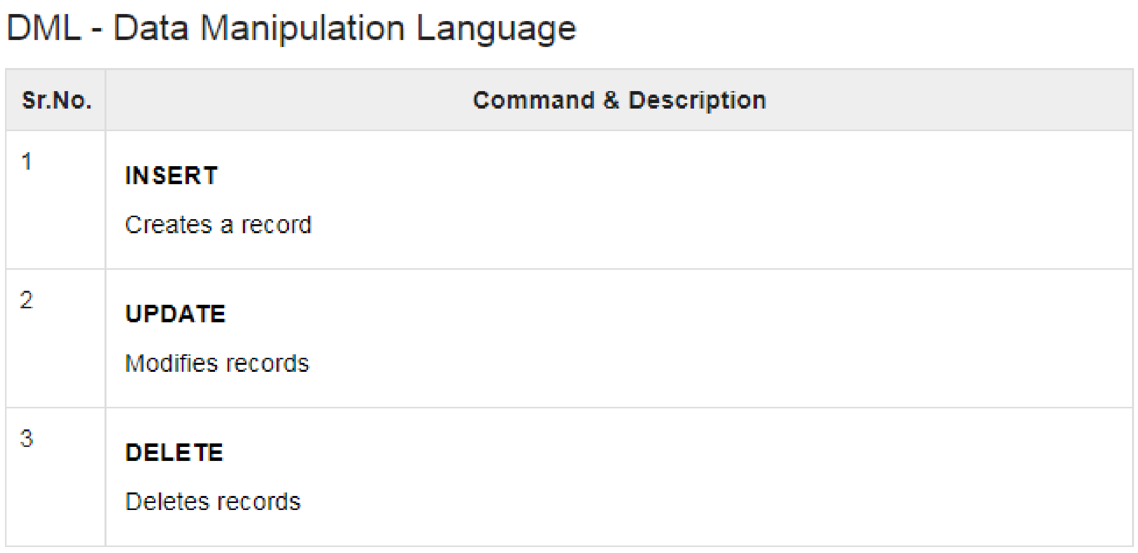
SQLite Commands

The standard SQLite commands to interact with relational databases are similar to SQL. They are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their operational nature .



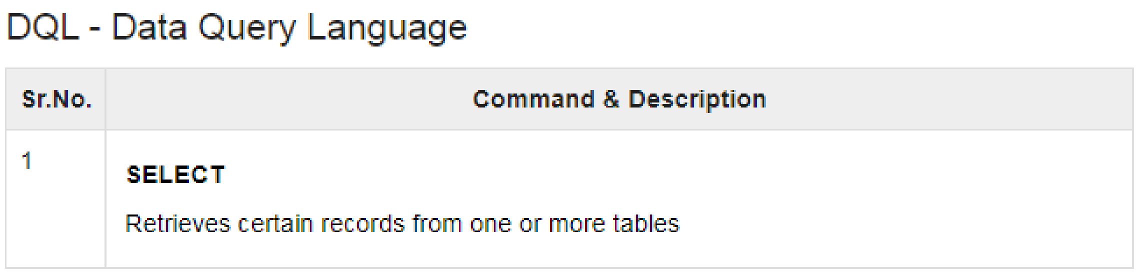
DDL - Data Definition Language

Sr.No.	Command & Description
1	**CREATE** Creates a new table, a view of a table, or other object in database.
2	**ALTER** Modifies an existing database object, such as a table.
3	**DROP** Deletes an entire table, a view of a table or other object in the database.



DML - Data Manipulation Language

Sr.No.	Command & Description
1	**INSERT** Creates a record
2	**UPDATE** Modifies records
3	**DELETE** Deletes records



DQL - Data Query Language

Sr.No.	Command & Description
1	**SELECT** Retrieves certain records from one or more tables

To use the module, you must first create a Connection object that represents the database. Here the data will be stored in the student.db file:

```
import sqlite3
conn = sqlite3.connect('student.db')
print ("opened database successfully")
```

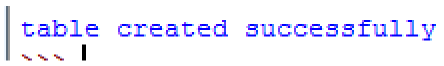
Output

```
opened database successfully
```

CREATE Table

```
import sqlite3

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

conn .execute('CREATE TABLE EMP(ID INT PRIMARY KEY NOT NULL,NAME TEXT NOT NULL,AGE INT NOT NULL,ADDRESS CHAR(50),SALARY real NOT NULL,date text not null)' )

print ("table created successfully")
```

Output

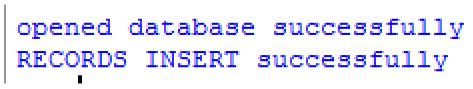


```
table created successfully
```

INSERT INTO Table

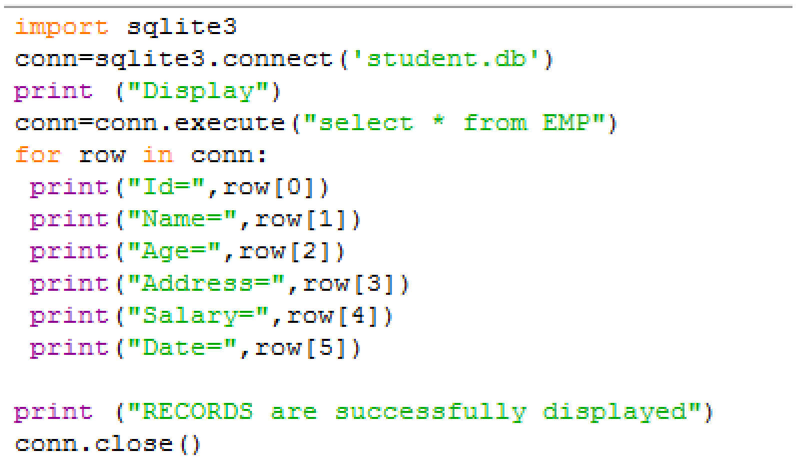
```
import sqlite3

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

print ("opened database successfully")

conn.execute("INSERT INTO EMP(ID,NAME,AGE,ADDRESS,SALARY,date)VALUES(1,'APPU',10,'ADDR1',22000.10,'2020-10-05')")

# Save (commit) the changes

conn.commit()

print ("RECORDS INSERT successfully" )

# We can also close the connection if we are done with it.

conn.close()
```

Output



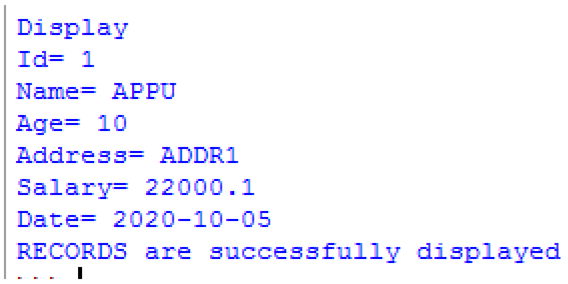
```
opened database successfully
RECORDS INSERT successfully
```

Display using select command



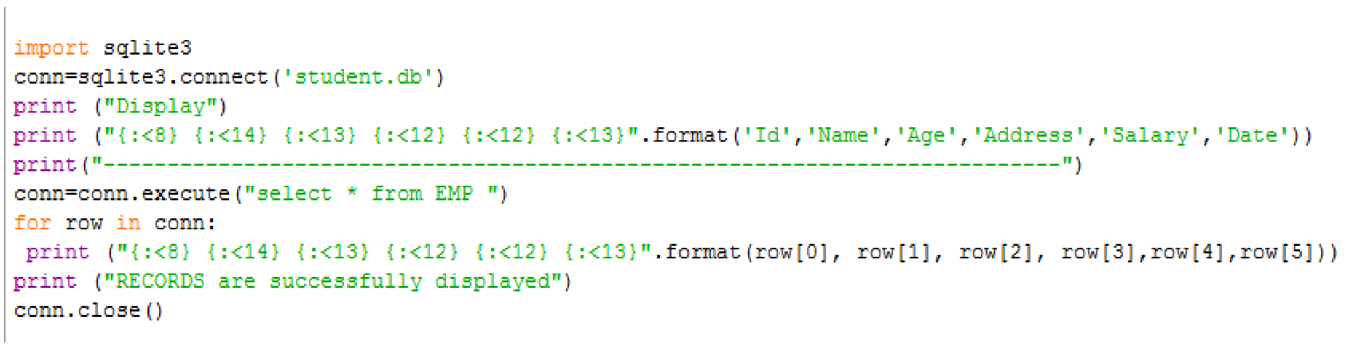
```
import sqlite3
conn=sqlite3.connect('student.db')
print ("Display")
conn=conn.execute("select * from EMP")
for row in conn:
 print("Id=",row[0])
 print("Name=",row[1])
 print("Age=",row[2])
 print("Address=",row[3])
 print("Salary=",row[4])
 print("Date=",row[5])

print ("RECORDS are successfully displayed")
conn.close()
```

Output



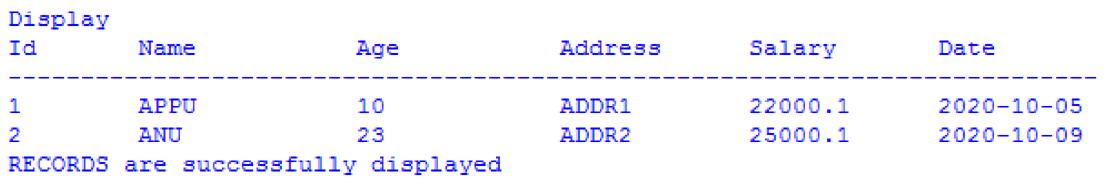
```
Display
Id= 1
Name= APPU
Age= 10
Address= ADDR1
Salary= 22000.1
Date= 2020-10-05
RECORDS are successfully displayed
```

Display result as a Table



```python
import sqlite3
conn=sqlite3.connect('student.db')
print ("Display")
print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format('Id','Name','Age','Address','Salary','Date'))
print("--------------------------------------------------------------------------")
conn=conn.execute("select * from EMP ")
for row in conn:
 print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format(row[0], row[1], row[2], row[3],row[4],row[5]))
print ("RECORDS are successfully displayed")
conn.close()
```

Output



```
Display
Id       Name           Age           Address      Salary       Date
-------------------------------------------------------------------------
1        APPU           10            ADDR1        22000.1      2020-10-05
2        ANU            23            ADDR2        25000.1      2020-10-09
RECORDS are successfully displayed
```

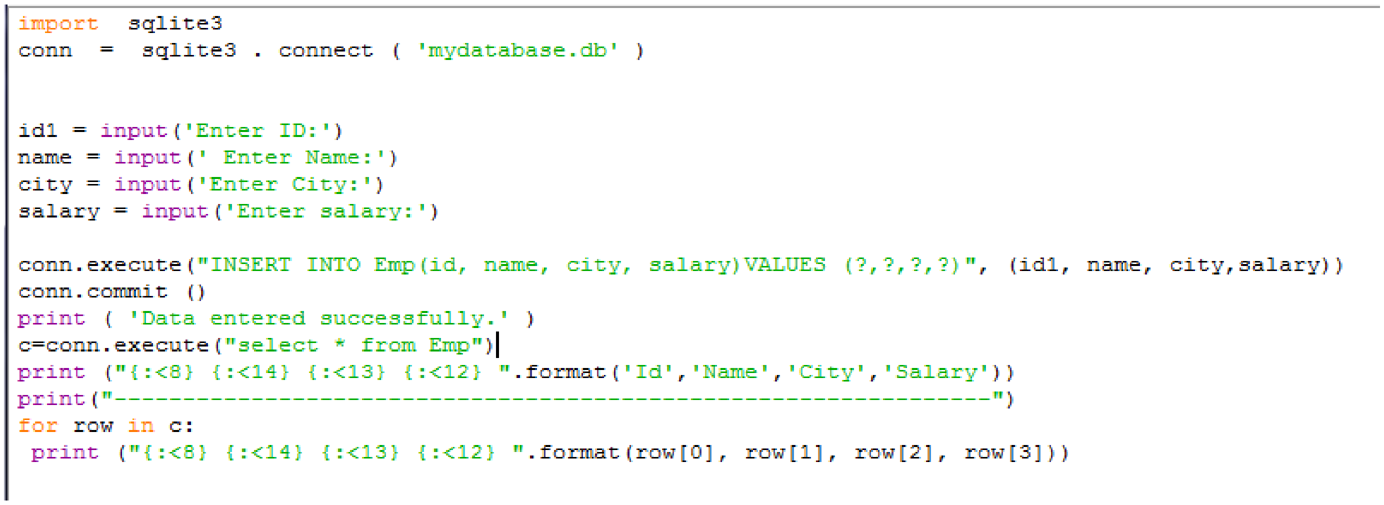
Select * from T1

Select id,name,age from T1

Select id,name,age from T1 where id=1

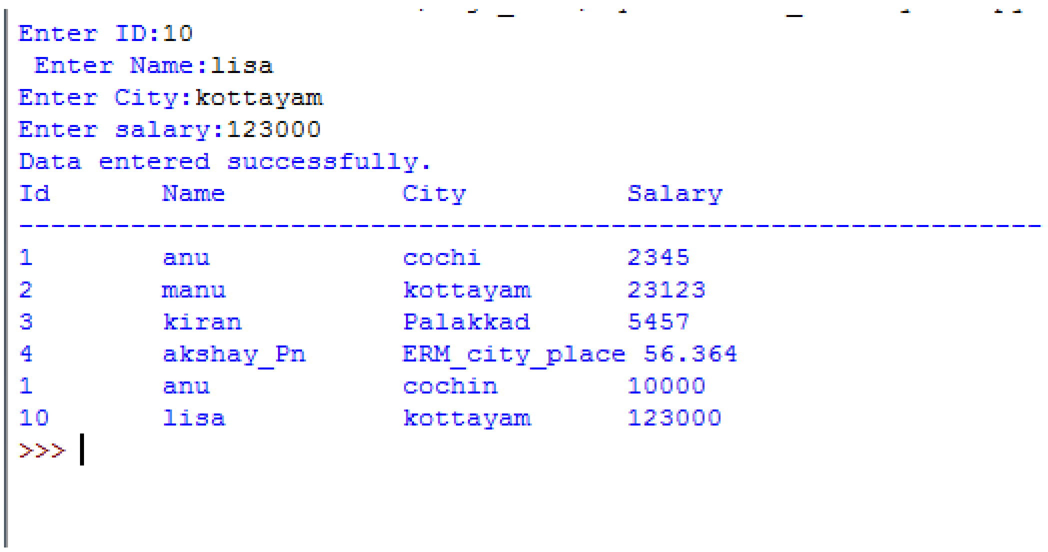
Select id,name,age from T1 where name='APPU'

insert values From keybord



```
import  sqlite3
conn  =  sqlite3 . connect ( 'mydatabase.db' )


id1 = input('Enter ID:')
name = input(' Enter Name:')
city = input('Enter City:')
salary = input('Enter salary:')

conn.execute("INSERT INTO Emp(id, name, city, salary)VALUES (?,?,?,?)", (id1, name, city,salary))
conn.commit ()
print ( 'Data entered successfully.' )
c=conn.execute("select * from Emp")
print ("{:<8} {:<14} {:<13} {:<12} ".format('Id','Name','City','Salary'))
print("-----------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} ".format(row[0], row[1], row[2], row[3]))
```

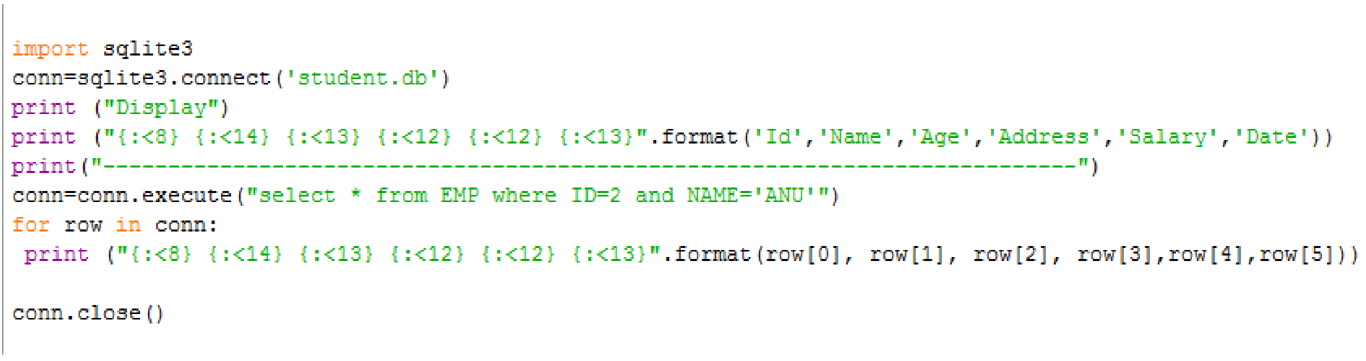
Output



```
Enter ID:10
 Enter Name:lisa
Enter City:kottayam
Enter salary:123000
Data entered successfully.
Id       Name           City          Salary
-----------------------------------------------------------------
1        anu            cochi         2345
2        manu           kottayam      23123
3        kiran          Palakkad      5457
4        akshay_Pn      ERM_city_place 56.364
1        anu            cochin        10000
10       lisa           kottayam      123000
>>> 
```

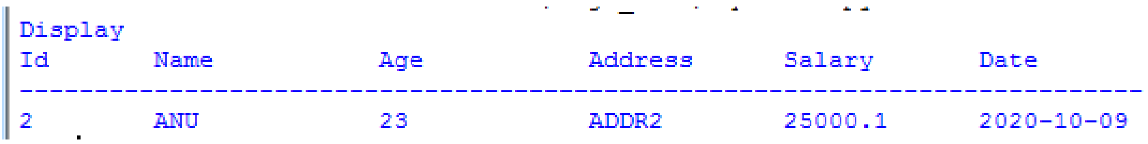
WHERE Clause

SQLite WHERE clause is used to specify a condition while fetching the data from one table or multiple tables.The WHERE clause not only is used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc



```python
import sqlite3
conn=sqlite3.connect('student.db')
print ("Display")
print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format('Id','Name','Age','Address','Salary','Date'))
print("------------------------------------------------------------------------------")
conn=conn.execute("select * from EMP where ID=2 and NAME='ANU'")
for row in conn:
 print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format(row[0], row[1], row[2], row[3],row[4],row[5]))

conn.close()
```

Output



```
Display
Id       Name           Age           Address      Salary       Date
-----------------------------------------------------------------------------
2        ANU            23            ADDR2        25000.1      2020-10-09
```

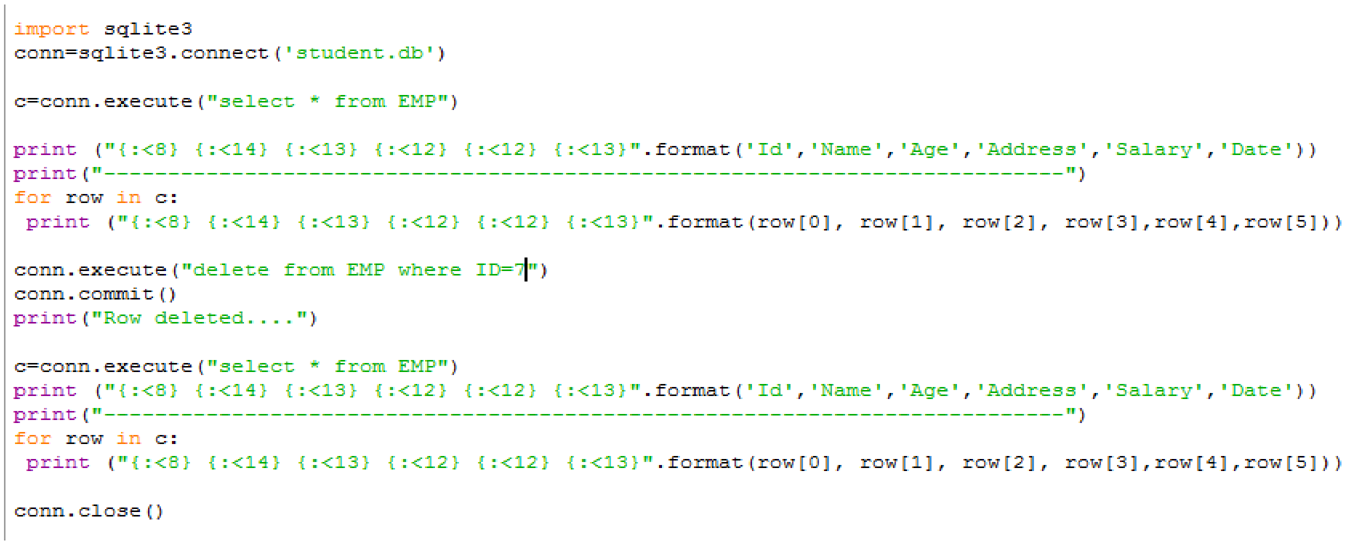
DELETE Query

SQLite DELETE Query is used to delete the existing records from a table. You can use WHERE clause with DELETE query to delete the selected rows, otherwise all the records would be deleted.

Syntax

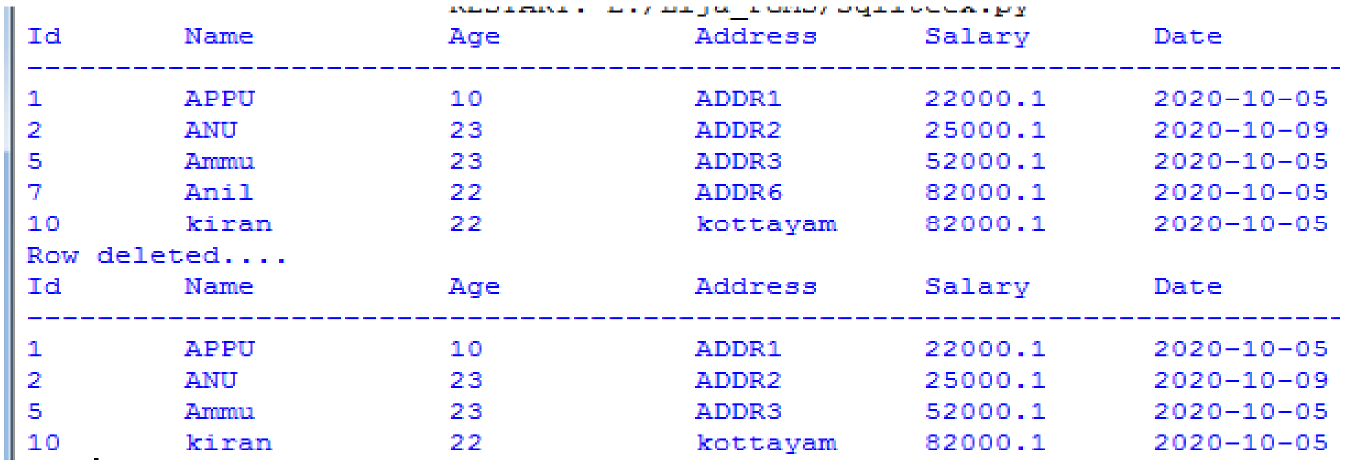
DELETE FROM table_name

WHERE [condition];



```python
import sqlite3
conn=sqlite3.connect('student.db')

c=conn.execute("select * from EMP")

print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format('Id','Name','Age','Address','Salary','Date'))
print("-------------------------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format(row[0], row[1], row[2], row[3],row[4],row[5]))

conn.execute("delete from EMP where ID=7")
conn.commit()
print("Row deleted....")

c=conn.execute("select * from EMP")
print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format('Id','Name','Age','Address','Salary','Date'))
print("-------------------------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format(row[0], row[1], row[2], row[3],row[4],row[5]))

conn.close()
```

Output



```
Id       Name           Age           Address      Salary       Date
---------------------------------------------------------------------------
1        APPU           10            ADDR1        22000.1      2020-10-05
2        ANU            23            ADDR2        25000.1      2020-10-09
5        Ammu           23            ADDR3        52000.1      2020-10-05
7        Anil           22            ADDR6        82000.1      2020-10-05
10       kiran          22            kottayam     82000.1      2020-10-05
Row deleted....
Id       Name           Age           Address      Salary       Date
---------------------------------------------------------------------------
1        APPU           10            ADDR1        22000.1      2020-10-05
2        ANU            23            ADDR2        25000.1      2020-10-09
5        Ammu           23            ADDR3        52000.1      2020-10-05
10       kiran          22            kottayam     82000.1      2020-10-05
```

UPDATE Query

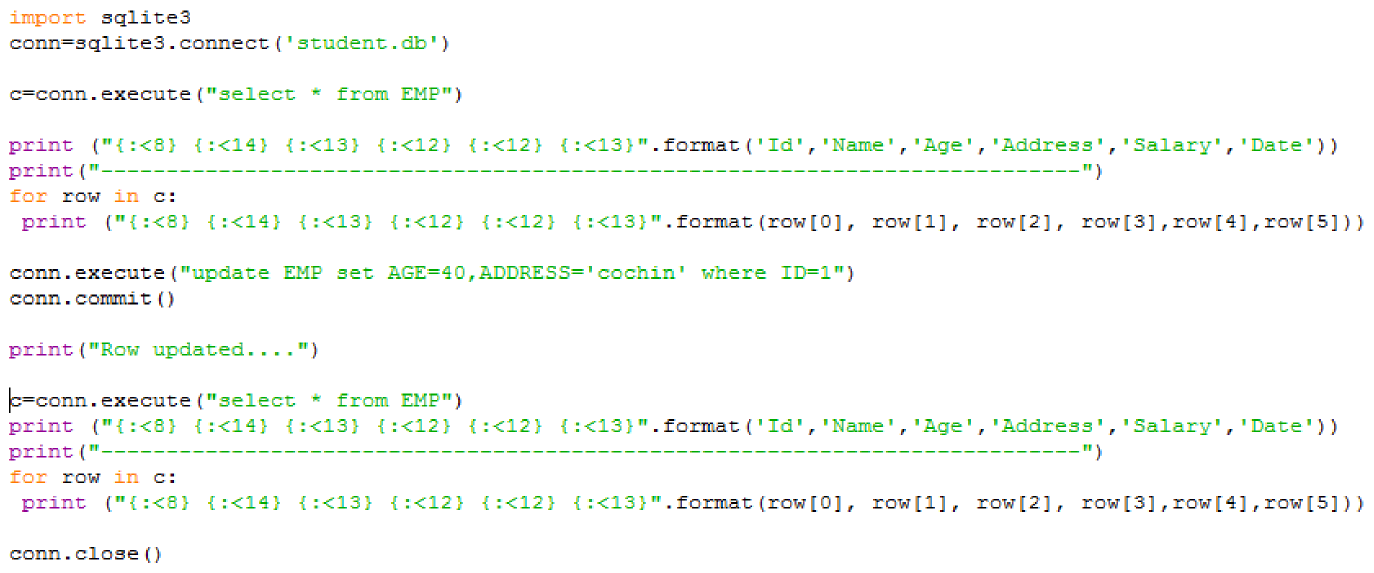
SQLite UPDATE Query is used to modify the existing records in a table. You can use WHERE clause with UPDATE query to update selected rows, otherwise all the rows would be updated.

Syntax

UPDATE table_name

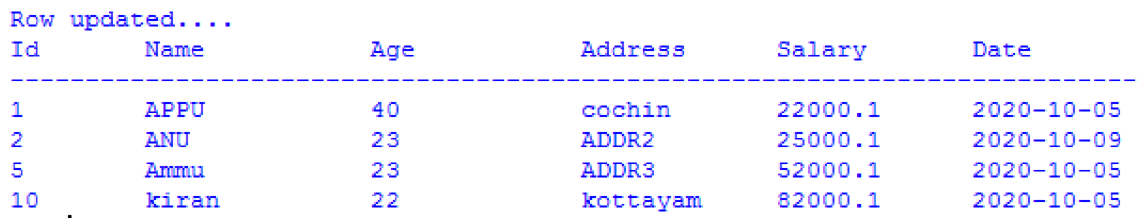
SET column1 = value1, column2 = value2...., columnN = valueN

WHERE [condition];



```
import sqlite3
conn=sqlite3.connect('student.db')

c=conn.execute("select * from EMP")

print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format('Id','Name','Age','Address','Salary','Date'))
print("---------------------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format(row[0], row[1], row[2], row[3],row[4],row[5]))

conn.execute("update EMP set AGE=40,ADDRESS='cochin' where ID=1")
conn.commit()

print("Row updated....")

c=conn.execute("select * from EMP")
print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format('Id','Name','Age','Address','Salary','Date'))
print("---------------------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format(row[0], row[1], row[2], row[3],row[4],row[5]))

conn.close()
```

Output



```
Row updated....
Id       Name           Age           Address      Salary       Date
---------------------------------------------------------------------------
1        APPU           40            cochin       22000.1      2020-10-05
2        ANU            23            ADDR2        25000.1      2020-10-09
5        Ammu           23            ADDR3        52000.1      2020-10-05
10       kiran          22            kottayam     82000.1      2020-10-05
```

ORDER BY Clause

ORDER BY clause is used to sort the data in an ascending or descending order, based on one or more columns.

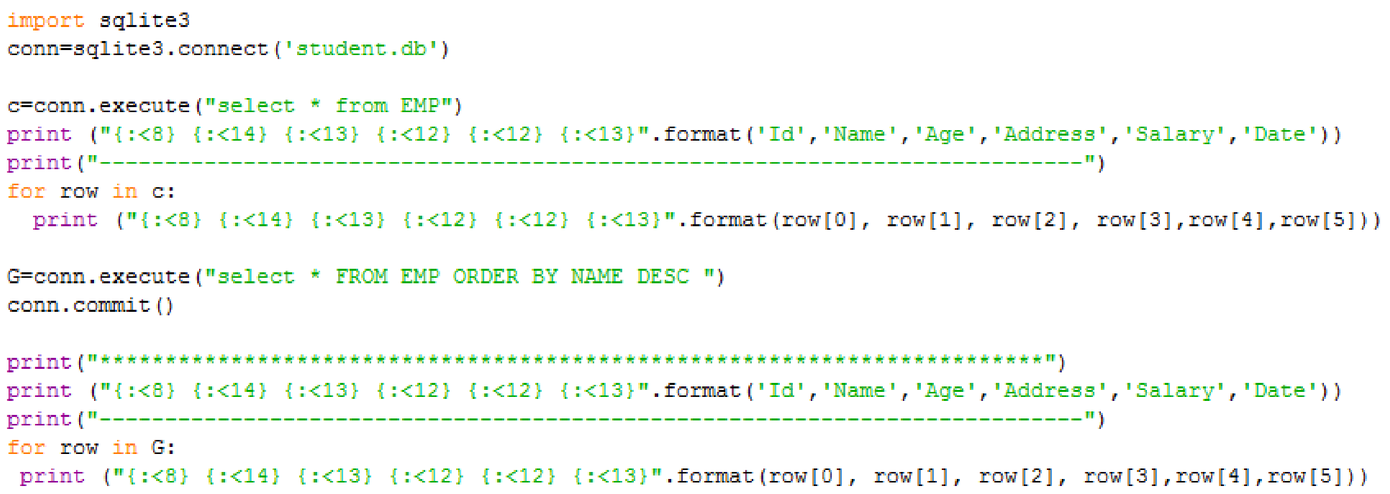
Syntax

SELECT column-list

FROM table_name

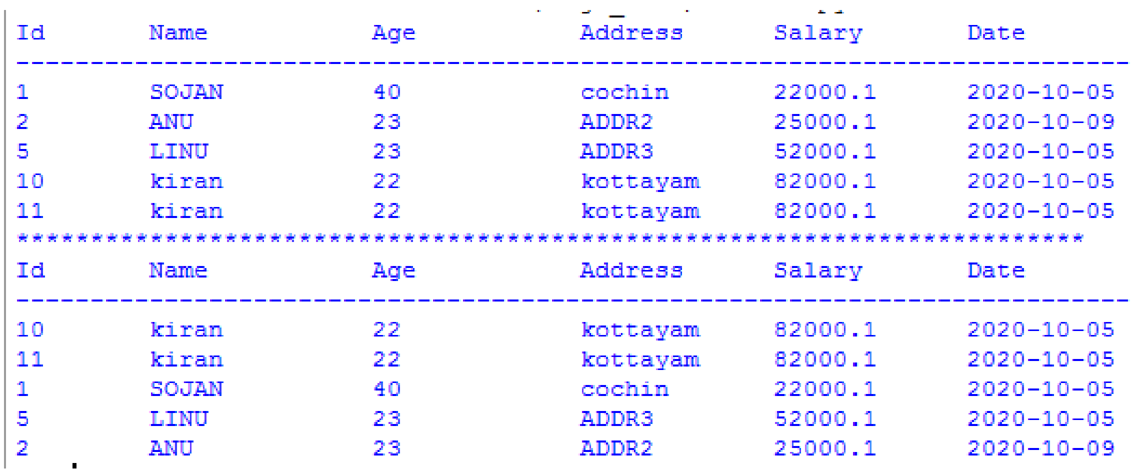
[WHERE condition]

[ORDER BY column1, column2, .. columnN] [ASC | DESC];



```
import sqlite3
conn=sqlite3.connect('student.db')

c=conn.execute("select * from EMP")
print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format('Id','Name','Age','Address','Salary','Date'))
print("---------------------------------------------------------------------------")
for row in c:
  print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format(row[0], row[1], row[2], row[3],row[4],row[5]))

G=conn.execute("select * FROM EMP ORDER BY NAME DESC ")
conn.commit()

print("************************************************************************")
print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format('Id','Name','Age','Address','Salary','Date'))
print("---------------------------------------------------------------------------")
for row in G:
 print ("{:<8} {:<14} {:<13} {:<12} {:<12} {:<13}".format(row[0], row[1], row[2], row[3],row[4],row[5]))
```

Output



```
Id       Name           Age           Address      Salary       Date
---------------------------------------------------------------------------
1        SOJAN          40            cochin       22000.1      2020-10-05
2        ANU            23            ADDR2        25000.1      2020-10-09
5        LINU           23            ADDR3        52000.1      2020-10-05
10       kiran          22            kottayam     82000.1      2020-10-05
11       kiran          22            kottayam     82000.1      2020-10-05
************************************************************************
Id       Name           Age           Address      Salary       Date
---------------------------------------------------------------------------
10       kiran          22            kottayam     82000.1      2020-10-05
11       kiran          22            kottayam     82000.1      2020-10-05
1        SOJAN          40            cochin       22000.1      2020-10-05
5        LINU           23            ADDR3        52000.1      2020-10-05
2        ANU            23            ADDR2        25000.1      2020-10-09
```

DISTINCT Keyword

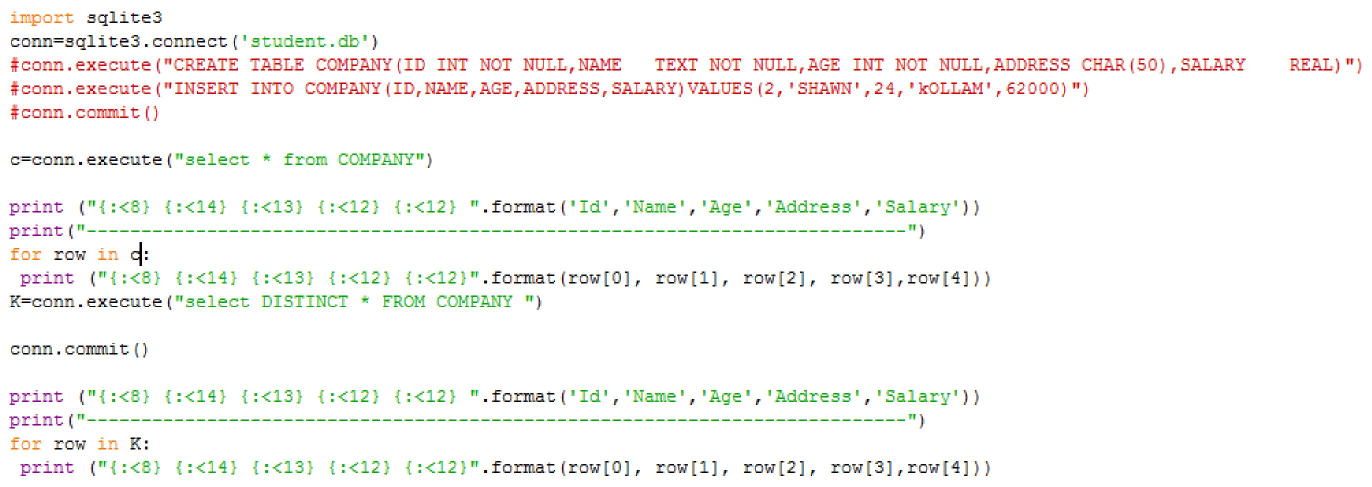
DISTINCT keyword is used in conjunction with SELECT statement to eliminate all the duplicate records and fetching only the unique records.

Syntax

SELECT DISTINCT column1, column2,.....columnN

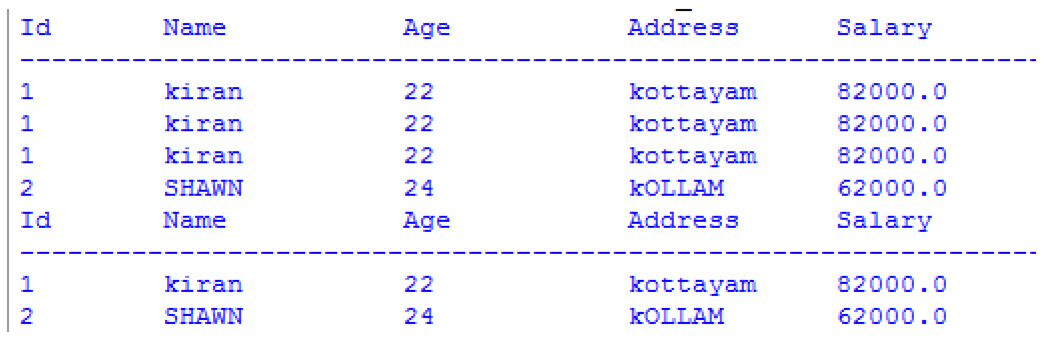
FROM table_name

WHERE [condition]



```
import sqlite3
conn=sqlite3.connect('student.db')
#conn.execute("CREATE TABLE COMPANY(ID INT NOT NULL,NAME   TEXT NOT NULL,AGE INT NOT NULL,ADDRESS CHAR(50),SALARY    REAL)")
#conn.execute("INSERT INTO COMPANY(ID,NAME,AGE,ADDRESS,SALARY)VALUES(2,'SHAWN',24,'kOLLAM',62000)")
#conn.commit()

c=conn.execute("select * from COMPANY")

print ("{:<8} {:<14} {:<13} {:<12} {:<12} ".format('Id','Name','Age','Address','Salary'))
print("--------------------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} {:<12}".format(row[0], row[1], row[2], row[3],row[4]))
K=conn.execute("select DISTINCT * FROM COMPANY ")

conn.commit()

print ("{:<8} {:<14} {:<13} {:<12} {:<12} ".format('Id','Name','Age','Address','Salary'))
print("--------------------------------------------------------------------------")
for row in K:
 print ("{:<8} {:<14} {:<13} {:<12} {:<12}".format(row[0], row[1], row[2], row[3],row[4]))
```

Output



```
Id       Name           Age           Address      Salary
------------------------------------------------------------------
1        kiran          22            kottayam     82000.0
1        kiran          22            kottayam     82000.0
1        kiran          22            kottayam     82000.0
2        SHAWN          24            kOLLAM       62000.0
Id       Name           Age           Address      Salary
------------------------------------------------------------------
1        kiran          22            kottayam     82000.0
2        SHAWN          24            kOLLAM       62000.0
```

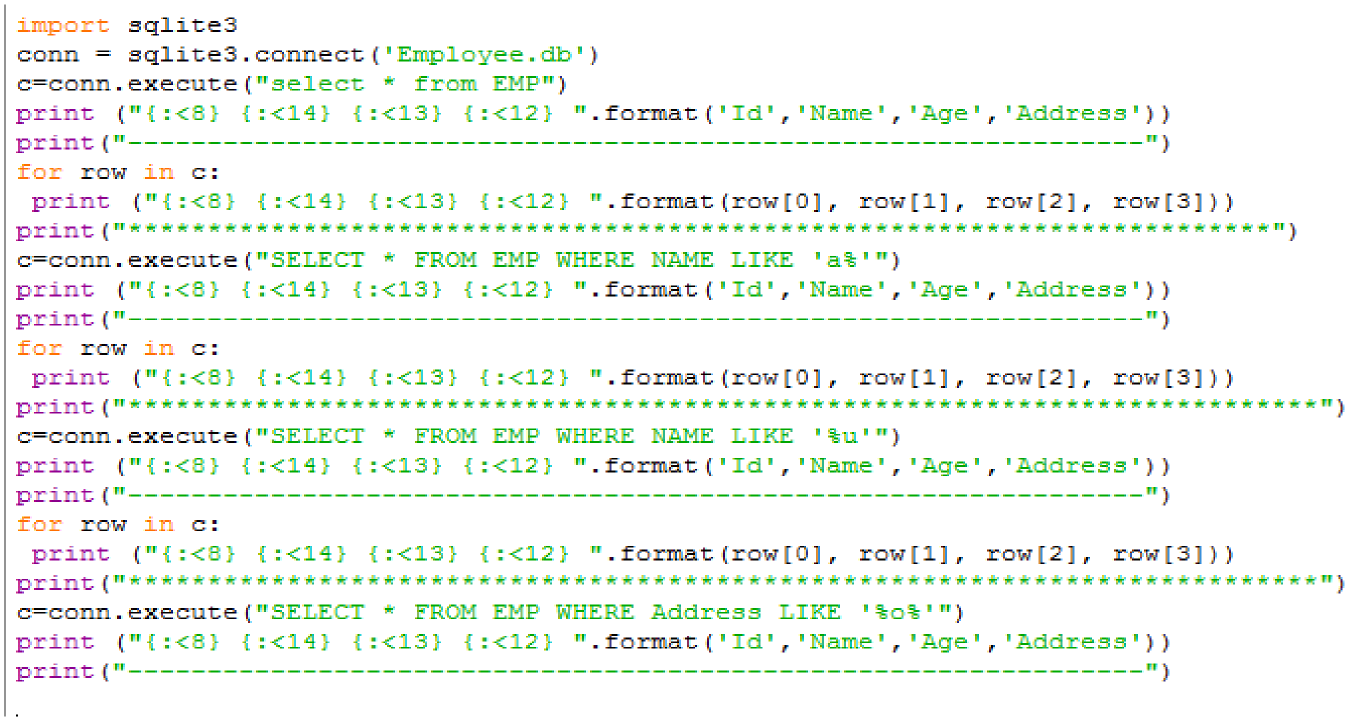
LIKE Clause

SQLite LIKE operator is used to match text values against a pattern using wildcards. If the search expression can be matched to the pattern expression, the LIKE operator will return true, which is 1. There are two wildcards used in conjunction with the LIKE operator −

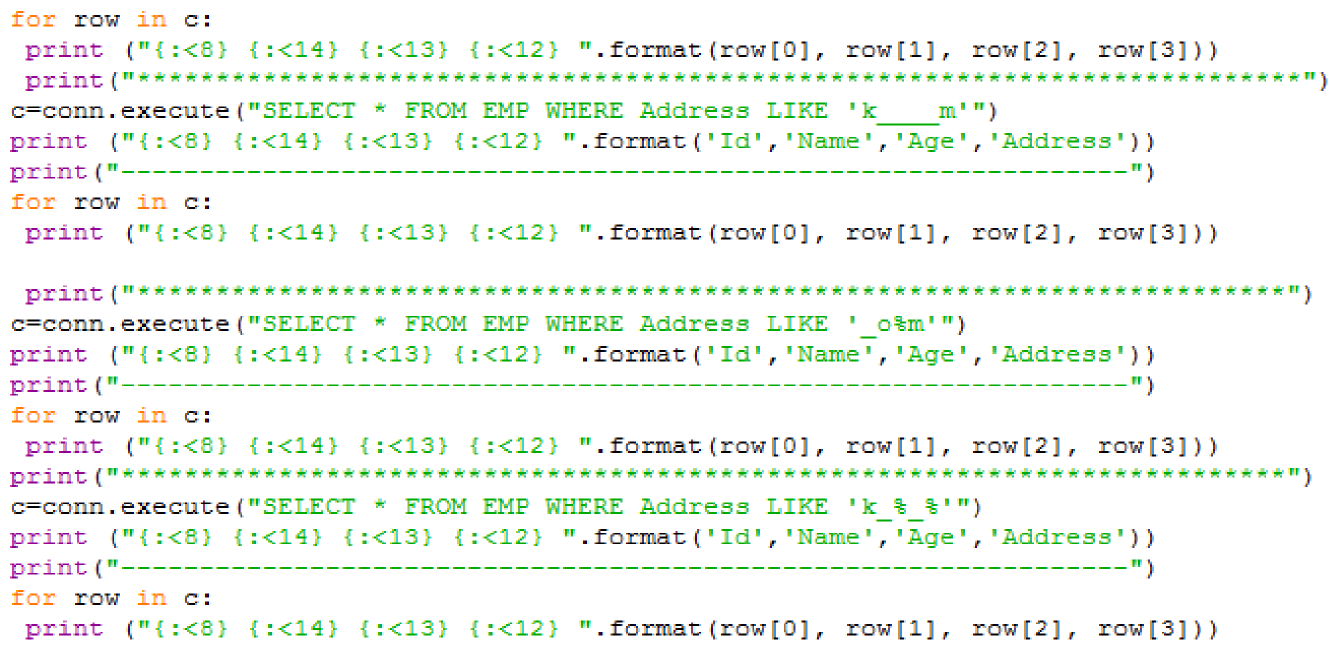
The percent sign (%)

The underscore (_)

Sr.No.	Statement & Description
1	**WHERE SALARY LIKE '200%'** Finds any values that start with 200
2	**WHERE SALARY LIKE '%200%'** Finds any values that have 200 in any position
3	**WHERE SALARY LIKE '_00%'** Finds any values that have 00 in the second and third positions
4	**WHERE SALARY LIKE '2_%_%'** Finds any values that start with 2 and are at least 3 characters in length
5	**WHERE SALARY LIKE '%2'** Finds any values that end with 2
6	**WHERE SALARY LIKE '_2%3'** Finds any values that has a 2 in the second position and ends with a 3
7	**WHERE SALARY LIKE '2___3'** Finds any values in a five-digit number that starts with 2 and ends with 3

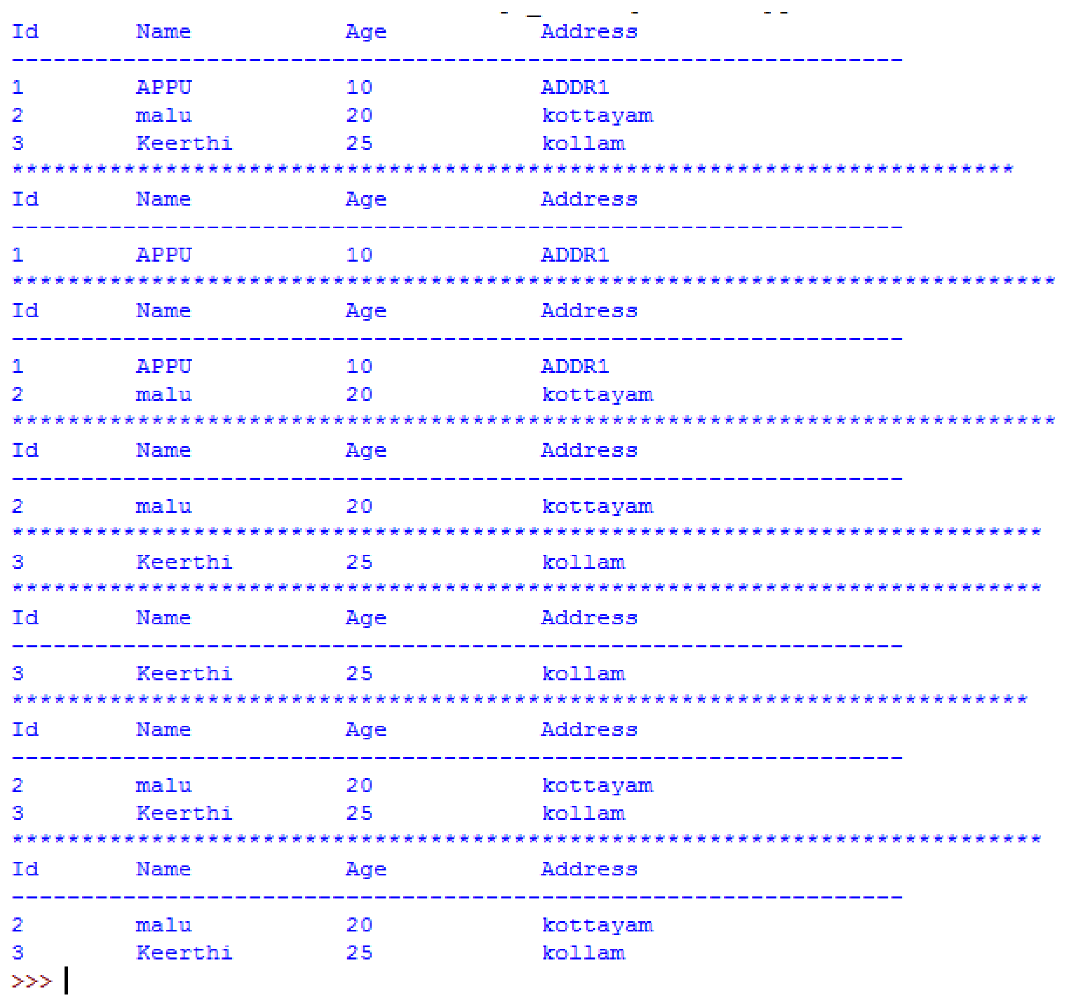


```
import sqlite3
conn = sqlite3.connect('Employee.db')
c=conn.execute("select * from EMP")
print ("{:<8} {:<14} {:<13} {:<12} ".format('Id','Name','Age','Address'))
print("-----------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} ".format(row[0], row[1], row[2], row[3]))
print("************************************************************************")
c=conn.execute("SELECT * FROM EMP WHERE NAME LIKE 'a%'")
print ("{:<8} {:<14} {:<13} {:<12} ".format('Id','Name','Age','Address'))
print("-----------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} ".format(row[0], row[1], row[2], row[3]))
print("***************************************************************************")
c=conn.execute("SELECT * FROM EMP WHERE NAME LIKE '%u'")
print ("{:<8} {:<14} {:<13} {:<12} ".format('Id','Name','Age','Address'))
print("-----------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} ".format(row[0], row[1], row[2], row[3]))
print("***************************************************************************")
c=conn.execute("SELECT * FROM EMP WHERE Address LIKE '%o%'")
print ("{:<8} {:<14} {:<13} {:<12} ".format('Id','Name','Age','Address'))
print("-----------------------------------------------------------------")
```



```
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} ".format(row[0], row[1], row[2], row[3]))
 print("***************************************************************************")
c=conn.execute("SELECT * FROM EMP WHERE Address LIKE 'k____m'")
print ("{:<8} {:<14} {:<13} {:<12} ".format('Id','Name','Age','Address'))
print("-----------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} ".format(row[0], row[1], row[2], row[3]))

 print("***************************************************************************")
c=conn.execute("SELECT * FROM EMP WHERE Address LIKE '_o%m'")
print ("{:<8} {:<14} {:<13} {:<12} ".format('Id','Name','Age','Address'))
print("-----------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} ".format(row[0], row[1], row[2], row[3]))
print("***************************************************************************")
c=conn.execute("SELECT * FROM EMP WHERE Address LIKE 'k_%_%'")
print ("{:<8} {:<14} {:<13} {:<12} ".format('Id','Name','Age','Address'))
print("-----------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<12} ".format(row[0], row[1], row[2], row[3]))
```

<u>Output</u>



```
Id       Name           Age           Address
----------------------------------------------------------------
1        APPU           10            ADDR1
2        malu           20            kottayam
3        Keerthi        25            kollam
************************************************************************
Id       Name           Age           Address
----------------------------------------------------------------
1        APPU           10            ADDR1
***************************************************************************
Id       Name           Age           Address
----------------------------------------------------------------
1        APPU           10            ADDR1
2        malu           20            kottayam
***************************************************************************
Id       Name           Age           Address
----------------------------------------------------------------
2        malu           20            kottayam
**************************************************************************
3        Keerthi        25            kollam
**************************************************************************
Id       Name           Age           Address
----------------------------------------------------------------
3        Keerthi        25            kollam
*************************************************************************
Id       Name           Age           Address
----------------------------------------------------------------
2        malu           20            kottayam
3        Keerthi        25            kollam
**************************************************************************
Id       Name           Age           Address
----------------------------------------------------------------
2        malu           20            kottayam
3        Keerthi        25            kollam
>>> |
```

SQLite Joins

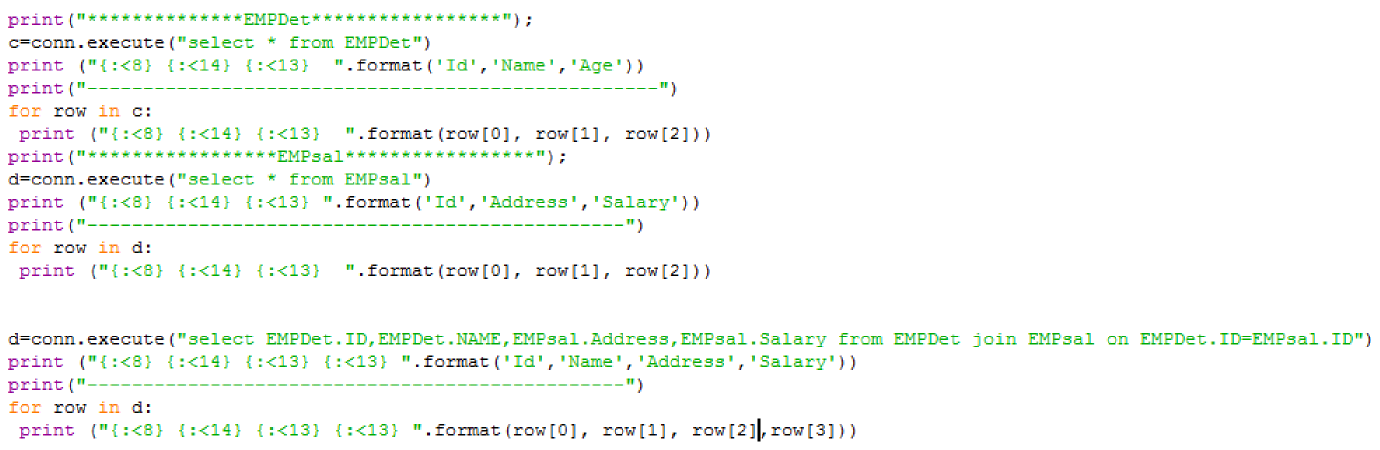
In SQLite, JOIN clause is used to combine records from two or more tables in a database. It unites fields from two tables by using the common values of the both table.

There are mainly three types of Joins in SQLite:

SQLite INNER JOIN/JOIN

SQLite OUTER JOIN

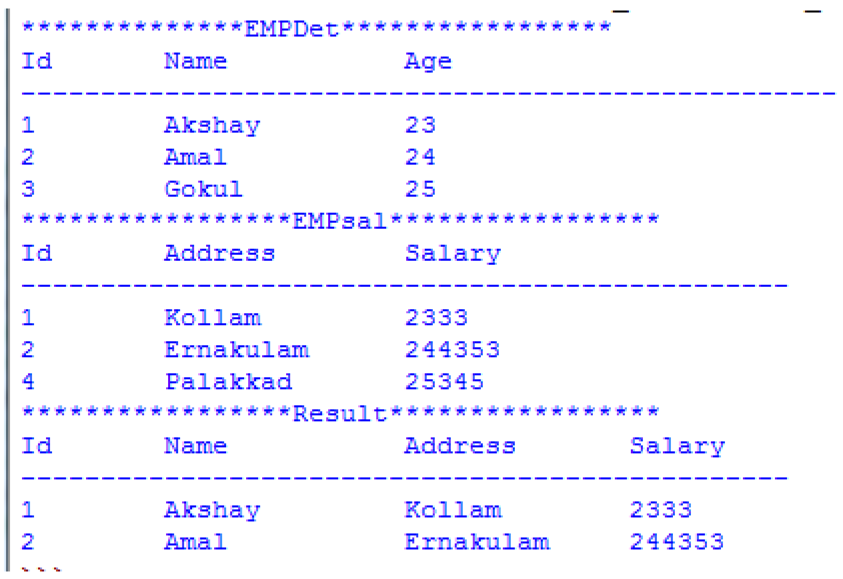
SQLite CROSS JOIN



```
print("**************EMPDet*****************");
c=conn.execute("select * from EMPDet")
print ("{:<8} {:<14} {:<13}  ".format('Id','Name','Age'))
print("---------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13}  ".format(row[0], row[1], row[2]))
print("*****************EMPsal*****************");
d=conn.execute("select * from EMPsal")
print ("{:<8} {:<14} {:<13} ".format('Id','Address','Salary'))
print("-----------------------------------------------")
for row in d:
 print ("{:<8} {:<14} {:<13}  ".format(row[0], row[1], row[2]))


d=conn.execute("select EMPDet.ID,EMPDet.NAME,EMPsal.Address,EMPsal.Salary from EMPDet join EMPsal on EMPDet.ID=EMPsal.ID")
print ("{:<8} {:<14} {:<13} {:<13} ".format('Id','Name','Address','Salary'))
print("-----------------------------------------------")
for row in d:
 print ("{:<8} {:<14} {:<13} {:<13} ".format(row[0], row[1], row[2],row[3]))
```

d=conn.execute("select EMPDet.ID,EMPDet.NAME,EMPsal.Address,EMPsal.Salary from EMPDet inner join EMPsal on EMPDet.ID=EMPsal.ID ")

<u>Output</u>



```
**************EMPDet*****************
Id       Name           Age
---------------------------------------------------
1        Akshay         23
2        Amal           24
3        Gokul          25
*****************EMPsal*****************
Id       Address        Salary
-----------------------------------------------
1        Kollam         2333
2        Ernakulam      244353
4        Palakkad       25345
*****************Result*****************
Id       Name           Address       Salary
-----------------------------------------------
1        Akshay         Kollam        2333
2        Amal           Ernakulam     244353
```

<u>GROUP BY Clause</u>

The SQLite GROUP BY clause is used with SELECT statement to collaborate the same identical elements into groups.

The GROUP BY clause is used with WHERE clause in SELECT statement and precedes the ORDER BY clause.

Syntax:

SELECT column-list

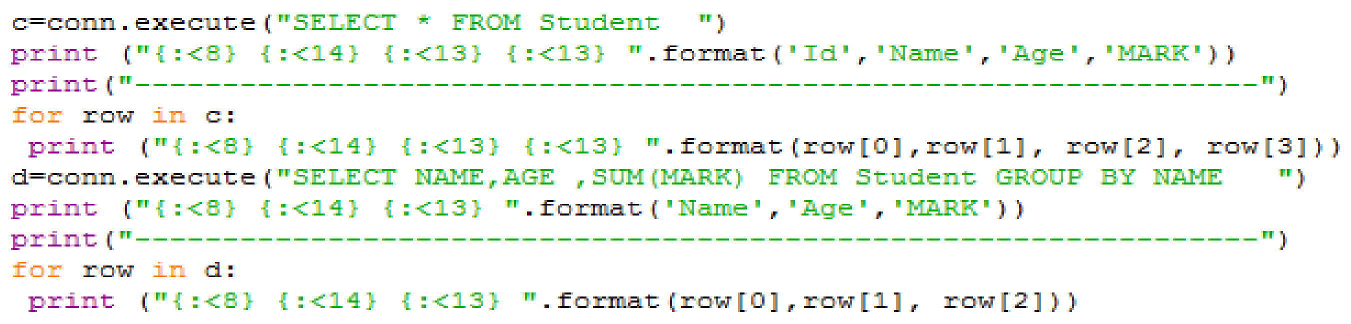
FROM table_name

WHERE [conditions]

GROUP BY column1, column2....columnN

ORDER BY column1, column2....columnN

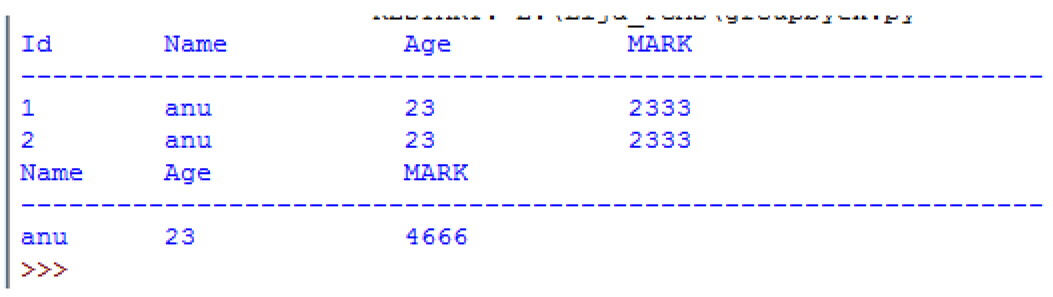
Example 1



```
c=conn.execute("SELECT * FROM Student  ")
print ("{:<8} {:<14} {:<13} {:<13} ".format('Id','Name','Age','MARK'))
print("-----------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<13} ".format(row[0],row[1], row[2], row[3]))
d=conn.execute("SELECT NAME,AGE ,SUM(MARK) FROM Student GROUP BY NAME   ")
print ("{:<8} {:<14} {:<13} ".format('Name','Age','MARK'))
print("-----------------------------------------------------------------")
for row in d:
 print ("{:<8} {:<14} {:<13} ".format(row[0],row[1], row[2]))
```

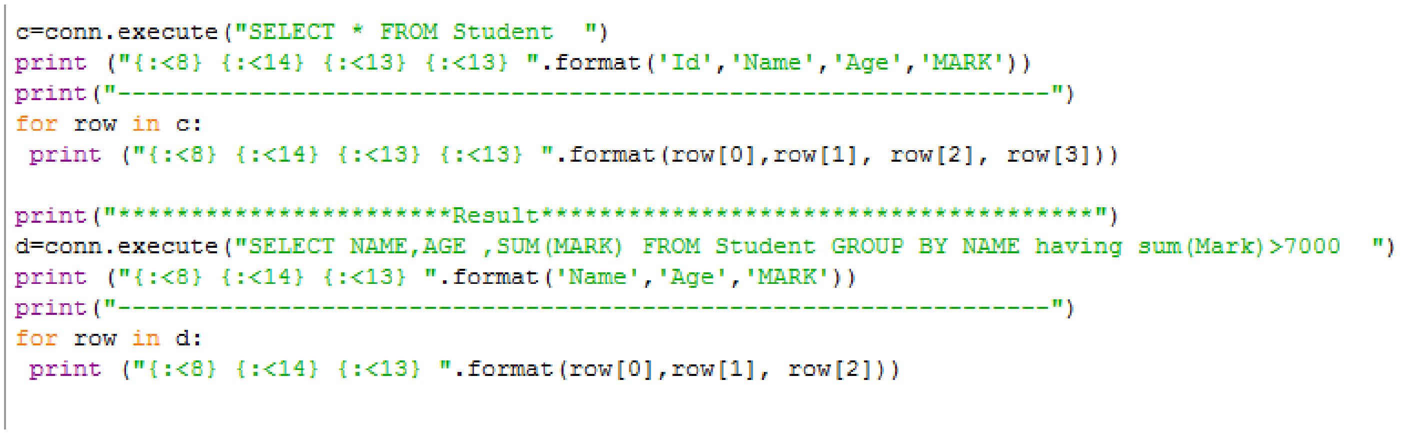
Output

Sum(),count(),avg(),min(),max()➔aggregate functions



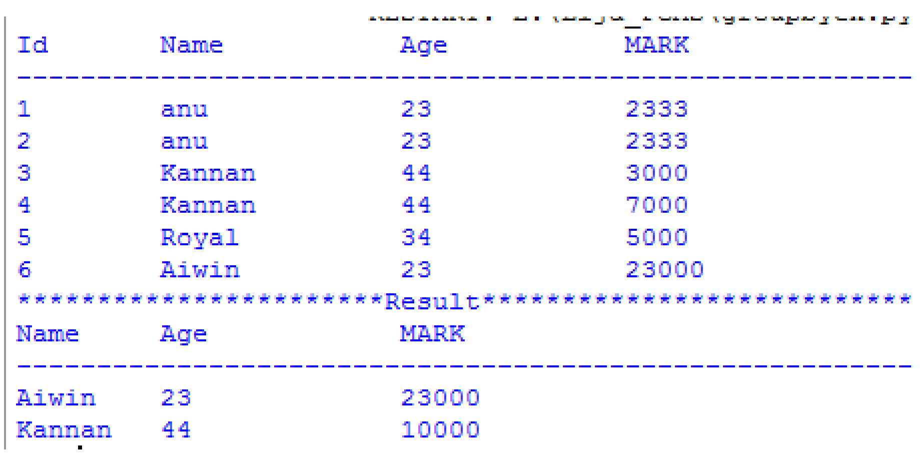
```
Id       Name           Age           MARK
-----------------------------------------------------------------
1        anu            23            2333
2        anu            23            2333
Name     Age            MARK
-----------------------------------------------------------------
anu      23             4666
>>>
```

Example 2



```
c=conn.execute("SELECT * FROM Student  ")
print ("{:<8} {:<14} {:<13} {:<13} ".format('Id','Name','Age','MARK'))
print("----------------------------------------------------------------")
for row in c:
 print ("{:<8} {:<14} {:<13} {:<13} ".format(row[0],row[1], row[2], row[3]))

print("***********************Result**************************************")
d=conn.execute("SELECT NAME,AGE ,SUM(MARK) FROM Student GROUP BY NAME having sum(Mark)>7000  ")
print ("{:<8} {:<14} {:<13} ".format('Name','Age','MARK'))
print("----------------------------------------------------------------")
for row in d:
 print ("{:<8} {:<14} {:<13} ".format(row[0],row[1], row[2]))
```

Output



```
Id       Name           Age           MARK
--------------------------------------------------------
1        anu            23            2333
2        anu            23            2333
3        Kannan         44            3000
4        Kannan         44            7000
5        Royal          34            5000
6        Aiwin          23            23000
***********************Result***************************
Name     Age            MARK
--------------------------------------------------------
Aiwin    23             23000
Kannan   44             10000
```