XII - COMPUTER SCIENCE
CHAPTER 15
DATA MANIPULATION THROUGH SQL

PART-II

1. Mention the user who uses the Database:

- Users of database can be human users, other programs or applications

2. Which method is used to connect a database? Give an example.

- **connect () method** is used to Connect a database and passing the name of the database to be accessed.
- Example:
- import sqlite3
- **connection = sqlite3.connect ("Academy.db")**
- cursor = connection.cursor()

3. What is the advantage of declaring a column as "INTEGER PRIMARY KEY"

- If a column of a table is declared to be an **INTEGER PRIMARY KEY**, then whenever a **NULL** will be used as an input for this column, the NULL will be automatically converted into an integer which will one larger than the highest value so far used in that column.
- If the table is empty, the value 1 will be used.

4. Write the command to populate record in a table. Give an example.

- To populate (add record) the table "INSERT" command is passed to SQLite.
- "execute" method executes the SQL command to perform some action.
- **For Example:**

```
sql_command = """INSERT INTO Student (Rollno, Sname, Grade) VALUES (NULL, "Akshay",
"B");"""
cursor.execute(sql_ command)
```

5. Which method is used to fetch all rows from the database table?

- The fetchall() method is used to fetch all rows from the database table

Example:

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("fetchall:")
result = cursor.fetchall()
for r in result:
    print(r)
```

OUTPUT:

```
fetchall:
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')
```

```
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')
(5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')
(6, 'PRIYA', 'A', 'F', 98.6, '2002-01-01')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```

PART - III

1. What is SQLite? What is it advantage?

- SQLite is a simple relational database system, which saves its data in regular data files or even in the internal memory of the computer.
- It is designed to be embedded in applications, instead of using a separate database server program such as MySQL or Oracle.
- SQLite is fast, rigorously tested, and flexible, making it easier to work.
- Python has a native library for SQLite.

2. Mention the difference between fetchone() and fetchmany()

fetchone()	fetchmany()
The **fetchone()** method returns the next row of a query result set or None in case there is no row left	The **fetchmany()** method returns the next number of rows (n) of the result set.
Using while loop and fetchone() method we can display all the records from a table.	Displaying specified number of records is done by using **fetchmany().**

3. What is the use of Where Clause? Give a python statement Using the where clause.

- The WHERE clause is used to extract only those records that fulfil a specified condition.
- In this example we are going to display the different grades scored by male students from “student table”
- Example:

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT DISTINCT (Grade) FROM student where gender='M'")
result = cursor.fetchall()
print(*result,sep="\n")
```

OUTPUT:

```
('B',)
('A',)
('C',)
('D',)
```

4. Read the following details. Based on that write a python script to display department wise

records
database name :- organization.db
Table name :- Employee
Columns in the table :- Eno, EmpName, Esal, Dept
PYTHON SCRIPT:

```
import sqlite3
connection = sqlite3.connect(“organization.db”)
cursor =connection.cursor()
cursor.execute(“SELECT * FROM Employee GROUP BY Dept;”)
result = cursor.fetchall()
print(*result, sep=”\n”)
```

5. Read the following details.Based on that write a python script to display records in desending order of Eno :

database name :- organization.db

Table name :- Employee

Columns in the table :- Eno, EmpName, Esal, Dept

PYTHON SCRIPT:

```
import sqlite3
connection = sqlite3.connect(“organization.db”)
cursor =connection.cursor()
cursor.execute(“SELECT * FROM Employee ORDER BY Eno DESC;”)
result = cursor.fetchall()
print(*result, sep=”\n”)
```

PART-IV

1. Write in brief about SQLite and the steps used to use it:

- SQLite is a simple relational database system, which saves its data in regular data files or even in the internal memory of the computer.
- It is designed to be embedded in applications, instead of using a separate database server program such as MySQL or Oracle.
- SQLite is fast, rigorously tested, and flexible, making it easier to work.
- Python has a native library for SQLite.

To use SQLite:
Step 1: import sqlite3
Step 2: Create a connection using connect () method and pass the name of the database File.
Step 3: Set the cursor object cursor = connection. cursor ()

- Connecting to a database in step2 means passing the name of the database to be accessed. If the database already exists the connection will open the same. Otherwise, Python will open a new database file with the specified name.

- Cursor in step 3: is a control structure used to traverse and fetch the records of the database.
- Cursor has a major role in working with Python. All the commands will be executed using cursor object only.
- To create a table in the database, create an object and write the SQL command in it.
- **Example:** sql_comm = "SQL statement"
- For executing the command use the cursor method and pass the required sql command as a parameter.
- Many number of commands can be stored in the sql_command can be executed one after other.
- Any changes made in the values of the record should be saved by the command "Commit" before closing the "Table connection".
- For Example:
- *# importing module*
- import sqlite3
- *# connecting to the database*
- connection = sqlite3.connect ("Academy.db")
- *# cursor*
- cursor = connection.cursor()

2. Write the Python script to display all the records of the following table using fetchmany()

Icode	ItemName	Rate
1003	Scanner	10500
1004	Speaker	3000
1005	Printer	8000
1008	Monitor	15000
1010	Mouse	700

PYTHON SCRIPT:

```
import sqlite3
connection = sqlite3.connect(“Materials.db”)
cursor =connection.cursor()
cursor.execute(“SELECT * FROM Materials;”)
print(“Displaying All the Records”)
result=  cursor.fetchmany(5)
print(*result, Sep=”\n”)
```

OUTPUT:

```
Displaying All The Records
(1003, “Scanner”, 10500)
(1004, “Speaker”, 3000)
(1005, “Printer”, 8000)
```

```
(1008, “Monitor”, 15000)
(1010, “Mouse”, 700)
```

3. What is the use of HAVING clause. Give an example python script

- Having clause is used to filter data based on the group functions.
- This is similar to WHERE condition but can be used only with group functions.
- Group functions cannot be used in WHERE Clause but can be used in HAVING clause.
- **Example**

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT GENDER,COUNT(GENDER) FROM Student GROUP BY GENDER
HAVING COUNT(GENDER)>3;")
result = cursor.fetchall()
co = [i[0] for i in cursor.description]
print(co)
print(result)
```

OUTPUT:

```
['gender', 'COUNT(GENDER)']
[('M', 5)]
```

4. Write a Python script to create a table called ITEM with following specification.
Add one record to the table.

Name of the database :- ABC
Name of the table :- Item
Column name and specification :-

Icode :-	**integer and act as primary key**
Item Name :-	**Item Name :-**
Rate :-	**Integer**
Record to be added :-	**1008, Monitor,15000**

PYTHON SCRIPT:

```
import sqlite3
connection = sqlite3.connect(“ABC.db”)
cursor =connection.cursor()
sql_command =”””CREATE TABLE Item (Icode INTEGER PRIMARY KEY,
                ItemName VARCHAR(25), Rate INTEGER);”””
cursor.execute(sql_comamnd)
sql_command=”””INSERT INTO Item(Icode, ItemName, Rate)
                        VALUES(1008, “Monitor”, 15000);”””
cursor.execute(sql_comamnd)
```

```
connection.commit()
connection.close()
print(“TABLE CREATED”)
print(“Displaying All the Records”)
result  cursor.fetchmany(5)
print(*result, Sep=”\n”)
```

OUTPUT:

```
TABLE CREATED
```

5.Consider the following table Supplier and item .Write a python script for (i) to (ii)

SUPPLIER				
Suppno	**Name**	**City**	**Icode**	**SuppQty**
S001	**Prasad**	**Delhi**	**1008**	**100**
S002	**Anu**	**Bangalore**	**1010**	**200**
S003	**Shahid**	**Bangalore**	**1008**	**175**
S004	**Akila**	**Hydrabad**	**1005**	**195**
S005	**Girish**	**Hydrabad**	**1003**	**25**
S006	**Shylaja**	**Chennai**	**1008**	**180**
S007	**Lavanya**	**Mumbai**	**1005**	**325**

i) Display Name, City and Itemname of suppliers who do not reside in Delhi.

```
import sqlite3
connection = sqlite3.connect(“ABC.db”)
cursor.execute(“SELECT Supplier.Name, Supplier.City,Item.ItemName  FROM Supplier,Item
                WHERE Supplier.Icode = Item.Icode AND Supplier.City NOT In Delhi ”)
co = [i[0] for i in cursor.description]
print(co)
result = cursor.fetchall()
print(*result, Sep=”\n”)
```

OUTPUT:

```
[„Name‟,     „City‟,        „ItemName‟]
[„Anu‟,      „Bangalore‟,   „Scanner‟]
[„Shahid‟,   „Bangalore‟,   „Speaker‟]
[„Akila‟,    „Hydrabad‟,    „Printer‟]
[„Girish‟,   „Hydrabad‟,    „Monitor‟]
[„Shylaja‟,  „Chennai‟,     „Mouse‟]
[„Lavanya‟,  „Mumbai‟,      „CPU‟]
```

ii) Increment the SuppQty of Akila by 40

```
import sqlite3
connection = sqlite3.connect(“ABC.db”)
cursor.execute(“UPDATE Supplier SET SuppQty = SuppQty +40 WHERE Name = “Akila”)
cursor.commit()
result = cursor.fetchall()
print(result)
```

OUTPUT:

```
(S004, “Akila”, “Hydrabad”, 1005,235)
```

EXTRA QUESTION ANSWER:

Database:

- A database is an organized collection of data. The term "database" can both refer to the data themselves or to the database management system.

Database Management System:

- The Database management system is a software application for the interaction between users and the databases.

Creating a Database using SQLite:

```
import sqlite3
connection = sqlite3.connect ("Academy.db")
cursor = connection.cursor()
```

- In the above example a database with the name "Academy" would be created.

Creating a Table:

- The SQL syntax for creating a table "Student" in the database "Academy".
- CREATE TABLE Student (Rollno INTEGER, Sname VARCHAR(20), Grade CHAR(1), gender CHAR(1), Average float(5.2), birth_date DATE, PRIMARY KEY (Rollno));

SELECT Query:

- **“Select”** is the most commonly used statement in SQL.
- The SELECT Statement in SQL is used to retrieve or fetch data from a table in a database.
- Syntax: **“Select * from table_name”** and all the table data can be fetched in an object in the form of list of lists.
- For Example: “Select * from student”

Displaying A record using fetchone():

The fetchone() method returns the next row of a query result set or None in case there is no row left.

Example:

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("\nfetch one:")
res = cursor.fetchone()
print(res)
```
OUTPUT
```
fetch one:
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
```

Displaying all records using fetchone() :

Using while loop and fetchone() method we can display all the records from a table.

Example:

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("fetching all records one by one:")
result = cursor.fetchone()
while result is not None:
    print(result)
    result = cursor.fetchone()
```
OUTPUT:
```
    fetching all records one by one:
    (1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
    (2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')
    (3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
    (4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')
    (5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')
    (6, 'PRIYA', 'A', 'F', 98.6, '2002-01-01')
    (7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```

Displayingusing fetchmany():

Displaying specified number of records is done by using fetchmany(). This method returns the next number of rows (n) of the result set.

Example : Program to display the content of tuples using fetchmany()

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("fetching first 3 records:")
```

```
result = cursor.fetchmany(3)
print(result)
```

OUTPUT:

```
fetching first 3 records:
[(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12'), (2, 'Aravind', 'A', 'M', 92.5, '2000-08-17'), (3,
'BASKAR', 'C', 'M', 75.2, '1998-05-17')]
```

Example : Program to display the content of tuples in newline without using loops

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("fetching first 3 records:")
result = cursor.fetchmany(3)
print(*result,sep="\n")
```

OUTPUT:

```
fetching first 3 records:
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
```

CLAUSES IN SQL:

- SQL provides various clauses that can be used in the SELECT statements.

  DISTINCT

  WHERE

  GROUP BY

  ORDER BY.

  HAVING

SQL DISTINCT CLAUSE

- The distinct clause is helpful when there is need of avoiding the duplicate values present in any specific columns/table.
- When we use distinct keyword only the unique values are fetched.
- In this example we are going to display the different grades scored by students from “student table”.
- **Example:**

  ```
  import sqlite3
  connection = sqlite3.connect("Academy.db")
  cursor = connection.cursor()
  cursor.execute("SELECT DISTINCT (Grade) FROM student")
  ```

```
result = cursor.fetchall()
print(result)
```

OUTPUT

```
[('B',), ('A',), ('C',), ('D',)]
```

SQL Group By Clause:

- The SELECT statement can be used along with GROUP BY clause.
- The GROUP BY clause groups records into summary rows. It returns one records for each group.
- It is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.
- The following example count the number of male and female from the student table and display the result.

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT gender,count(gender) FROM student Group BY gender")
result = cursor.fetchall()
print(*result,sep="\n")
```

<u>OUTPUT:</u>

```
('F', 2)
('M', 5)
```

SQL ORDER BY Clause:

- The ORDER BY Clause can be used along with the SELECT statement to sort the data of specific fields in an ordered way.
- It is used to sort the result-set in ascending or descending order.
- In this example name and Rollno of the students are displayed in alphabetical order of names

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT Rollno,sname FROM student Order BY sname")
result = cursor.fetchall()
print(*result,sep="\n")
```

<u>OUTPUT:</u>

```
(1, 'Akshay')
(2, 'Aravind')
(3, 'BASKAR')
```

```
(6, 'PRIYA')
(4, 'SAJINI')
(7, 'TARUN')
```

The SQL AND, OR and NOT Operators:

- The WHERE clause can be combined with AND, OR, and NOT operators.
- The AND and OR operators are used to filter records based on more than one condition.
- In this example you are going to display the details of students who have scored other than ‘A’ or ‘B’ from the “student table”
- **<u>Example for WHERE WITH NOT Operator:</u>**

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student where grade<>'A' and Grade<>'B'")
result = cursor.fetchall()
print(*result,sep="\n")
```

<u>OUTPUT:</u>

```
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```

- **<u>Example for WHERE WITH AND Operator:</u>**
- In this example we are going to display the name, Rollno and Average of students who have scored an average between 80 to 90% (both limits are inclusive)

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT Rollno,Same,Average FROM student WHERE (Average>=80 AND
Average<=90)")
result = cursor.fetchall()
print(*result,sep="\n")
```

<u>OUTPUT:</u>

```
(1, 'Akshay', 87.8)
(5, 'VARUN', 80.6)
```

- **<u>Example for WHERE WITH OR Operator:</u>**
- In this example we are going to display the name and Rollno of students who have not scored an average between 60 to 70%

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT Rollno,sname FROM student WHERE (Average<60 OR
Average>70)")
result = cursor.fetchall()
```

```
print(*result,sep="\n")
```

OUTPUT:

```
(1, 'Akshay')
(2, 'Aravind')
(3, 'BASKAR')
(4, 'SAJINI')
(5, 'VARUN')
(6, 'PRIYA')
```

Querying A Date Column:

- In this example we are going to display the name and grade of students who have born in the year 2001

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT Rollno,sname FROM student
WHERE(Birth_date>='2001-01-01' ANDBirth_date<='2001-12-01')")
result = cursor.fetchall()
print(*result,sep="\n")
```

OUTPUT:

```
(5, 'VARUN')
```

Aggregate Functions:

- These functions are used to do operations from the values of the column and a single value is returned.
  1. COUNT()
  2. AVG()
  3. SUM()
  4. MIN()
  5. MAX()

COUNT() function:

- The SQL COUNT() function returns the number of rows in a table satisfying the criteria specified in the WHERE clause. COUNT() returns 0 if there were no matching rows.
- **Example1: In this example we are going to count the number of records(rows)**

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT COUNT(*) FROM student ")
result = cursor.fetchall()
print(result)
```

Output:

```
[(7,)]
```

- **Example 2 : In this example we are going to count the number of records by specifying a column**

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT COUNT(AVERAGE) FROM student ")
result = cursor.fetchall()
print(result)
```

Output:

```
[(7,)]
```

- Note: NULL values are not counted. In case if we had null in one of the records in student table for example in Average field then the output would be 6.

AVG():

- The following SQL statement in the python program finds the average mark of all students.
- **Example:**

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT AVG(AVERAGE) FROM student ")
result = cursor.fetchall()
print(result)
```

OUTPUT:

```
[(84.65714285714286,)]
```

- Note: NULL values are ignored.

SUM():

- The following SQL statement in the python program finds the sum of all average in the Average field of “Student table”.
- **Example:**

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT SUM(AVERAGE) FROM student ")
result = cursor.fetchall()
print(result)
```

OUTPUT :

```
[(592.6,)]
```

- **Note:** NULL values are ignored.

MAX() AND MIN() FUNCTIONS:

- The MAX() function returns the largest value of the selected column.
- The MIN() function returns the smallest value of the selected column.
- The following example show the highest and least average student's name.

```
import sqlite3
connection = sqlite3.connect("Organization.db")
cursor = connection.cursor()
print("Displaying the name of the Highest Average")
cursor.execute("SELECT sname,max(AVERAGE) FROM student ")
result = cursor.fetchall()
print(result)
print("Displaying the name of the Least Average")
cursor.execute("SELECT sname,min(AVERAGE) FROM student ")
result = cursor.fetchall()
print(result)
```

OUTPUT:

```
Displaying the name of the Highest Average
[('PRIYA', 98.6)]
Displaying the name of the Least Average
[('TARUN', 62.3)]
```

Updating A Record:

- You can even update a record (tuple) in a table through python script.
- The following example change the name "Priya" to "Priyanka" in a record in "student table"

```
import sqlite3
conn = sqlite3.connect("Academy.db")
conn.execute("UPDATE Student SET sname ='Priyanka' where Rollno='6'")
conn.commit()
print ("Total number of rows updated :", conn.total_changes)
cursor = conn.execute("SELECT * FROM Student")
for row in cursor:
        print (row)
conn.close()
```

OUTPUT:

```
Total number of rows updated : 1
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')
```

```
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')
(5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')
(6, 'Priyanka', 'A', 'F', 98.6, '2002-01-01')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```

Deletion Operation:

- Similar to Sql command to delete a record, Python also allows to delete a record.
- The following example delete the content of Rollno 2 from "student table"

```
import sqlite3
conn = sqlite3.connect("Academy.db")
conn.execute("DELETE from Student where Rollno='2'")
conn.commit()
print("Total number of rows deleted :", conn.total_changes)
cursor =conn.execute("SELECT * FROM Student")
for row in cursor:
        print(row)
conn.close()
```

OUTPUT:

```
Total number of rows deleted : 1
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')
(5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')
(6, 'Priyanka', 'A', 'F', 98.6, '2002-01-01')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```

Data input by User:

- In this example we are going to accept data using Python input() command during runtime and then going to write in the Table called "Person"

```
import sqlite3
con =sqlite3.connect("Academy.db")
cur =con.cursor()
cur.execute("DROP Table person")
cur.execute("create table person (name, age, id)")
print("Enter 5 students names:")
who =[input() for i in range(5)]
print("Enter their ages respectively:")
age =[int(input()) for i in range(5)]
print("Enter their ids respectively:")
p_id =[int(input())for i in range(5)]
n =len(who)
for i in range(n):
        cur.execute("insert into person values (?, ?, ?)", (who[i], age[i], p_id[i]))
        cur.execute("select * from person")
print("Displaying All the Records From Person Table")
print (*cur.fetchall(), sep='\n' )
```

OUTPUT :

```
Enter 5 students names:
RAM
KEERTHANA
KRISHNA
HARISH
GIRISH
Enter their ages respectively:
28
12
21
18
16
Enter their ids respectively:
1
2
3
4
5
Displaying All the Records From Person Table
('RAM', 28, 1)
('KEERTHANA', 12, 2)
('KRISHNA', 21, 3)
('HARISH', 18, 4)
('GIRISH', 16, 5)
```

Using Multiple Table for Querying:

- Python allows to query more than one table by joining them. In the following example a new table called "Appointments" which contain the details of students Rollno, Duty, Age is created.
- The tables "student" and "Appointments" are joined and displayed the result with the column headings.
- **Example:**

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("""DROP TABLE Appointment;""")
sql_command = """
CREATE TABLE Appointment(rollnointprimarkey,Dutyvarchar(10),age int)"""
cursor.execute(sql_command)
sql_command = """INSERT INTO Appointment (Rollno,Duty ,age )
VALUES ("1", "Prefect", "17");"""
cursor.execute(sql_command)
sql_command = """INSERT INTO Appointment (Rollno, Duty, age)
VALUES ("2", "Secretary", "16");"""
cursor.execute(sql_command)
connection.commit()
cursor.execute("SELECT student.rollno,student.sname,Appointment.
Duty,Appointment.Age FROM student,Appointment where student.
rollno=Appointment.rollno")
co = [i[0] for i in cursor.description]
print(co)
result = cursor.fetchall()
for r in result:
        print(r)
```

OUTPUT:

```
['Rollno', 'Sname', 'Duty', 'age']
(1, 'Akshay', 'Prefect', 17)
(2, 'Aravind', 'Secretary', 16)
```

Integrating Query With Csv File:

- You can even store the query result in a CSV file.
- This will be useful to display the query output in a tabular format.
- Using Python script the student table is sorted "gender" wise in descending order and then arranged the records alphabetically.
- The output of this Query will be written in a CSV file called "SQL.CSV", again the content is read from the CSV file and displayed the result.
- **Example:**

```
import sqlite3
import io # to access replace()
import csv
d=open('c:/pyprg/sql.csv','w')
c=csv.writer(d)
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student ORDER BY GENDER DESC,SNAME")
co = [i[0] for i in cursor.description]
c.writerow(co)
data=cursor.fetchall()
for item in data:
c.writerow(item)
d.close()
with open('c:/pyprg/sql.csv', "r", newline=None) as fd:
# r = csv.reader(fd)
for line in fd:
        line = line.replace("\n", "")
        print(line)
cursor.close()
connection.close()
```

OUTPUT:

```
Rollno,Sname,Grade,gender,Average,birth_date
1, Akshay, B, M, 87.8, 2001-12-12
2, Aravind, A, M, 92.5, 2000-08-17
3, BASKAR, C, M, 75.2, 1998-05-17
7, TARUN, D, M, 62.3, 1999-02-01
5, VARUN, B, M, 80.6, 2001-03-14
6, PRIYA, A, F, 98.6, 2002-01-01
4, SAJINI, A, F, 95.6, 2002-11-01
```

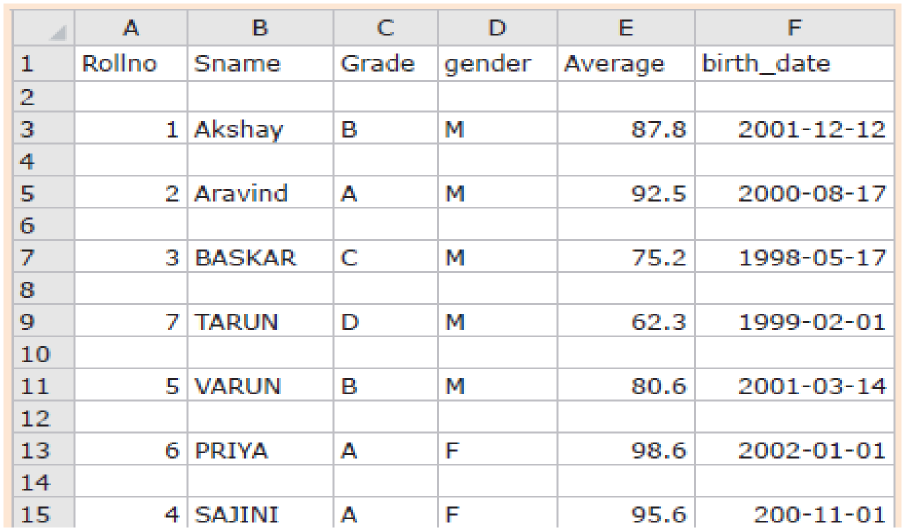
- **Example Opening the file ("sqlexcel.csv") through MS-Excel and view the result (Program is same similar to EXAMPLE 15.12 -1 script)**

```
import sqlite3
import io
import csv
conn = sqlite3.connect("Academy.db")
print(“Content of the table before sorting and writing in CSV file”)
cursor = conn.execute("SELECT * FROM Student")
for row in cursor:
        print (row)
d=open('c:\\pyprg\\sqlexcel.csv','w')
c=csv.writer(d)
cursor = conn.cursor()
cursor.execute("SELECT * FROM student ORDER BY GENDER DESC,SNAME")
co = [i[0] for i in cursor.description]
c.writerow(co)
data=cursor.fetchall()
for item in data:
        c.writerow(item)
d.close()
print(”sqlexcel.csv File is created open by visiting c:\\pyprg\\sqlexcel.csv”)
conn.close()
```

OUTPUT:

```
Content of the table before sorting and writing in CSV file
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')
(5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')
(6, 'Priyanka', 'A', 'F', 98.6, '2002-01-01')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
sqlexcel.csv File is created open by visiting c:\\pyprg\\sqlexcel.csv
```

OUTPUT THROUGH EXCEL:



	A	B	C	D	E	F
1	Rollno	Sname	Grade	gender	Average	birth_date
2						
3	1	Akshay	B	M	87.8	2001-12-12
4						
5	2	Aravind	A	M	92.5	2000-08-17
6						
7	3	BASKAR	C	M	75.2	1998-05-17
8						
9	7	TARUN	D	M	62.3	1999-02-01
10						
11	5	VARUN	B	M	80.6	2001-03-14
12						
13	6	PRIYA	A	F	98.6	2002-01-01
14						
15	4	SAJINI	A	F	95.6	200-11-01

Table List:

- To show (display) the list of tables created in a database the following program can be used.
- **Example :**

```
import sqlite3
con = sqlite3.connect('Academy.db')
cursor = con.cursor()
cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
print(cursor.fetchall())
```

OUTPUT:

```
[('Student',), ('Appointment',), ('Person',)]
```

The above program display the names of all tables created in 'Academy. db' database.

sqlite_master:

The master table holds the key information about your database tables and it is called sqlite_master

HANDS ON EXPERIENCE:

1. Create an interactive program to accept the details from user and store it in a csv file using Python for the following table.
Database name;- DB1
Table name : Customer

Cust_Id	Cust_Name	Address	Phone_no	City
C008	Sandeep	14/1 Pritam Pura	41206819	Delhi
C010	Anurag Basu	15A, Park Road	61281921	Kolkata
C012	Hrithik	7/2 Vasant Nagar	26121949	Delhi

2. Consider the following table GAMES. Write a python program to display the records for question (i) to (iv) and give outputs for SQL queries (v) to (viii)
Table: GAMES

Gcode	Name	GameName	Number	PrizeMoney	ScheduleDate
101	Padmaja	Carom Board	2	5000	01-23-2014
102	Vidhya	Badminton	2	12000	12-12-2013
103	Guru	Table Tennis	4	8000	02-14-2014
105	Keerthana	Carom Board	2	9000	01-01-2014
108	Krishna	Table Tennis	4	25000	03-19-2014

(i) To display the name of all Games with their Gcodes in descending order of their schedule date.
(ii) To display details of those games which are having Prize Money more than 7000.
(iii) To display the name and gamename of the Players in the ascending order of Gamename.
(iv) To display sum of PrizeMoney for each of the Numberof participation groupings (as shown in column Number 4)
(v) Display all the records based on GameName

Choose the best answer (1 Marks):

1. Which of the following is an organized collection of data?

(A) Database

(B) DBMS

(C) Information

(D) Records

2. SQLite falls under which database system?

(A) Flat file database system

(B) Relational Database system

(C) Hierarchical database system

(D) Object oriented Database system

3. Which of the following is a control structure used to traverse and fetch the records of the database?

(A) Pointer

(B) Key

(C) Cursor

(D) Insertion point

4. Any changes made in the values of the record should be saved by the command

(A) Save

(B) Save As

(C) Commit

(D) Oblige

5. Which of the following executes the SQL command to perform some action?

(A) Execute()

(B) Key()

(C) Cursor()

(D) run()

6. Which of the following function retrieves the average of a selected column of rows in a table?

(A) Add()

(B) SUM()

(C) AVG()

(D) AVERAGE()

7. The function that returns the largest value of the selected column is

(A) MAX()

(B) LARGE()

(C) HIGH()

(D) MAXIMUM()

8. Which of the following is called the master table?

(A) sqlite_master

(B) sql_master

(C) main_master

(D) master_main

9. The most commonly used statement in SQL is

(A) cursor

(B) select

(C) execute

(D) commit

10. Which of the following clause avoid the duplicate?

(A) Distinct

(B) Remove
(C) Where
(D) GroupBy
11. A **database** is an organized collection of data.
12. Users of database can be **human users, other programs or applications**
13. **SQLite** is a simple relational database system, which saves its data in regular data files.
14. **Cursor** is a control structure used to traverse and fetch the records of the database.
15. All the SQL commands will be executed using **cursor object** only.
16. As data in a table might contain **single or double quotes**, SQL commands in Python are denoted as **triple quoted string**.
17. **"Select"** is the most commonly used statement in SQL
18. The **SELECT Statement** in SQL is used to retrieve or fetch data from a table in a database
19. The **GROUP BY** clause groups records into summary rows
20. The **ORDER BY** Clause can be used along with the SELECT statement to sort the data of specific fields in an ordered way
21. **Having clause** is used to filter data based on the group functions.
22. **Where clause** cannot be used along with 'Group by'
23. The **WHERE clause** can be combined with AND, OR, and NOT operators
24. The **'AND' and 'OR' operators** are used to filter records based on more than one condition
25. **Aggregate functions** are used to do operations from the values of the column and a single value is returned.
26. **COUNT() function** returns the number of rows in a table.
27. **AVG() function** retrieves the average of a selected column of rows in a table.
28. **SUM() function** retrieves the sum of a selected column of rows in a table.
29. **MAX() function** returns the largest value of the selected column.
30. **MIN() function** returns the smallest value of the selected column
31. **sqlite_master** is the master table which holds the key information about your database tables.
32. The path of a file can be either represented as **'/' or using '\\'** in Python.
33. For example the path can be specified either as **'c:/pyprg/sql.csv', or c:\\pyprg\\sql.csv'.**
34. **Cursor** is used for performing all SQL commands.
35. The cursor object is created by calling the **cursor() method** of connection.
36. The **cursor** is used to traverse the records from the result set.
37. **fetchall ()** method is to fetch all rows from the database table
38. The **fetchone()** method returns the next row of a query result set or None in case there is no row left.
39. **cursor.fetchmany()** method that returns the next number of rows (n) of the result set
40. To print all elements in new lines or separated by space use **sep= "\n" or sep= ","** respectively.
41. The sqlite3 module supports **two kinds** of placeholders: **question marks?** ("qmark style") and **named placeholders :name** ("named style").
42. **cursor. description** contain the details of each column headings .It will be stored as a **tuple and the first one that is 0(zero)** index refers to the column name. Using this command you can display the **table's Field names.**

1. Which of the following is fast, flexible and easy to work?

a. CSV
b. SQLite
c. Perl
d. Ruby
2. Which method in SQLite is used to create a connection with a database file created?
a. cursor()
b. lite()
c. connect()
d. connection()
3. Which method has a major role in working with python?
a. cursor()
b. connect()
c. execure()
d. close()
4. The command to populate the table is
a. ADD
b. APPEND
c. INSERT
d. ADDROW
5. Which of the following statement in SQL is used to retrieve or fetch data from a table in a database?
a. select
b. insert
c. create
d. fetch
6. Which sqlite method is used to fetch all rows from the database table?
a. fetch()
b. fetchrowsall()
c. fetchmany()
d. fetchall()
7. Which sqlite method is used to fetch required number of rows from the database table?
a. fetch()
b. fetchrowsall()
c. fetchmany()
d. fetchall()
8. Which SQLite keyword is used to fetch only the unique values from the database table?
a. UNIQUE
b. DISTINCT

c. GROUPBY

d. HAVING

9. Which SQLite keyword is used to extract only those records that fulfill a specified condition?

a. WHERE

b. EXTRACT

c. CONNECT

d. CURSOR

10. Which of the following clause is often used with aggregate functions to group the result?

a. ORDER BY

b. WHERE

c. GROUP BY

d. DISTINCT