In [1]:
```
pip install mysql-connector-python
```

```
Requirement already satisfied: mysql-connector-python in c:\users\gadam\anaconda3\lib\site-packages (9.0.0)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
"1.SQL Statement & Operations"
```

In [ ]:
```
SQL(Structured query language ) is a standard language used to comminicate with relational databases like "MYSQ

1.create : Inserting a new records into the databases.

2.Read: Querying the databases to retrieve existing records.

3.Update: Modifying existing records in the databases.

4.Delete : Removing records from the database.
```

In [ ]:
```
"CRUD Operations in "
```

CRUD stands for create,read,update, and delete-the four basic functions that can be performed on databases.

In [ ]:
```
"Python MySql"

Python can be used in databases applications.

One of the most popular databases is MySQL
```

In [ ]:
```
"MySQl Database"
```

In [ ]:
```
"MySQL Driver"
```

In [ ]:
```
Python needs a MySQL driver to access the MySQL databases.

In this we will use the driver "MySQL Connector"

We recommend that you use PIP to install "MySQL Connector"

PIP is most likely already installed in our Python environment.

Navigate our command line to the location of PIP,and type the following:
```

In [ ]:
```
Download and install "MySQL Connector":
```

In [ ]:
```
pip install mysql-connector-python
```

In [ ]:
```
"Test MySQL Connector"
```

TO test if the installation was successful ,or if you already have "MySQL Connector" installed,create a Python page with the follwing content:

In [3]:
```
import mysql.connector as conn
```

If the above code was executed with no errors,"MySQl Connector" is installed and ready to be used.

In [ ]:
```
"Create Connection"
```

In [ ]:
```
Start by creating a connection to the database.

Use the username and password from your MYSQL databases
```

In [1]:
```
import mysql.connector as conn
connection=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="customers")
```

In [ ]:
```
"connection check"
```

In [3]:
```
if connection.is_connected():
    print("connected")
```

```
connected
```

In [ ]:
```
"4.3. Committing Transactions"
```

After executing any operation that modifies the database (like INSERT, UPDATE, DELETE), you need to commit the transaction to save the changes.

In [ ]:
```
"Create a Database"
```

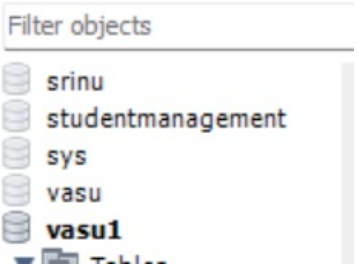
In [ ]:
```
To create a database in MySQL ,use the "CREATE DATABASE" statement
```

In [ ]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789"
)
mycursor=mydb.cursor()

mycursor.execute("create database vasu1") #check in Mysql workbench it is created or not. if it is created it w
connection.close()
```

In [ ]:
```
If the above code was executed with no errors ,we have succesfully created a database.
```

In [47]:
```
from PIL import Image
Image.open('database.png')
```

Out[47]:



In [ ]:
```
"Check if Database Exists or Not"
```

In [ ]:
```
we can check if a database exist by listing all databases in our system by using the "SHOW DATABASES" statement
```

In [21]:
```
import mysql.connector as conn
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789"
)
mycursor=mydb.cursor()
mycursor.execute("Show databases")
for i in mycursor:
    print(i)
```

```
('customers',)
('employee1',)
('information_schema',)
('joins',)
('mani',)
('my',)
('mysql',)
('performance_schema',)
('srinu',)
('studentmanagement',)
('sys',)
('vasu',)
('vasu1',)
('vasu11',)
('vasu122',)
```

In [ ]:
```
Or we can try to access the database when making the connection
```

In [25]:
```
import mysql.connector as conn
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
# mycursor=mydb.cursor()
```

If the above page is executed without any errors ,the database "vasu1" exists in our system

In [ ]:
```
"Python MySql Create Table"
```

In [ ]:
```
"Creating a Table"
```

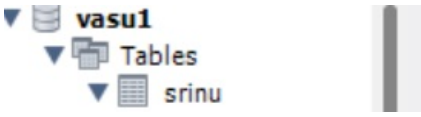
In [ ]:
```
To create a table in MySQL,use the "CREATE TABLE"  statement

Make sure you define the name of the database when you create the connection
```

In [ ]:
```
import mysql.connector as conn
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()
mycursor.execute("create table srinu(id int primary key,name varchar(20),age int ,location varchar(30))")
```

In [ ]:
```
If the above code was executed with no errors ,we have now succesfully created a table.
```

In [49]:
```
Image.open('tablename.png')
```

Out[49]:



In [ ]:
```
"check if Table exists or not"
```

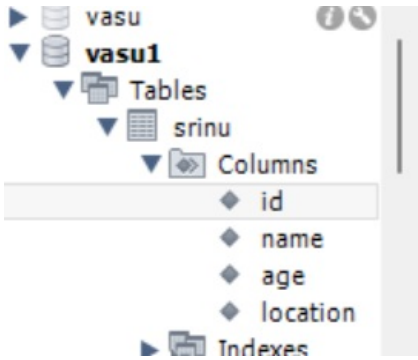
In [ ]:
```
we can check if a table exist by listing all tables in our database with the "show tables" statement:
```

In [7]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()
mycursor.execute("show tables")
for i in mycursor:
    print(i)
```

```
('srinu',)
('vasu',)
('vasu1',)
('vasu2',)
('vasu3',)
('vasu4',)
```

In [51]:
```
Image.open('inserting values.png')
```

Out[51]:



In [ ]:
```
"Primary Key"
```

In [ ]:
```
When creating a table,we should also create a column with a unique key for each record.

This can be done by defining a "Primary Key"

We use the statement,"INT AUTO_INCREMENT PRIMARY KEY"  which will insert a unique number for each record.Starti
```

In [29]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()
mycursor.execute("CREATE TABLE srinu( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100),age int,location VAR
```

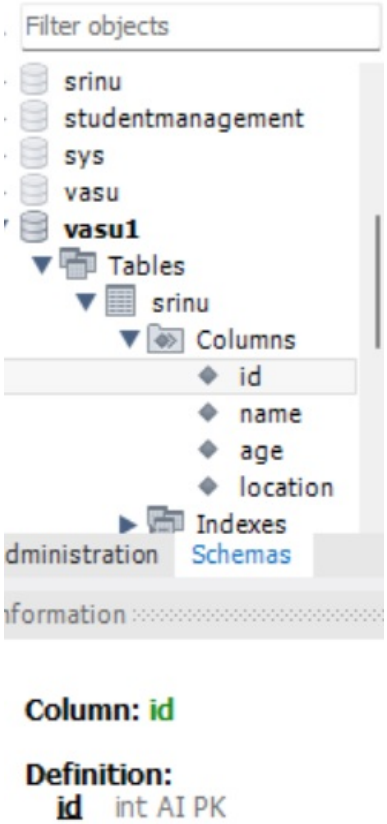
```
---------------------------------------------------------------------------
MySQLInterfaceError                       Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\mysql\connector\connection_cext.py:705, in CMySQLConnection.cmd_query(self, q
uery, raw, buffered, raw_as_string)
    704         query = query.encode("utf-8")
--> 705     self._cmysql.query(
    706         query,
    707         raw=raw,
    708         buffered=buffered,
    709         raw_as_string=raw_as_string,
    710         query_attrs=self.query_attrs,
    711     )
    712 except MySQLInterfaceError as err:

MySQLInterfaceError: Table 'srinu' already exists

The above exception was the direct cause of the following exception:

ProgrammingError                          Traceback (most recent call last)
Cell In[29], line 10
      3 mydb=conn.connect(
      4     host="localhost",
      5     user="root",
      6     password="Gsrinu@789",
      7     database="vasu1"
      8 )
      9 mycursor=mydb.cursor()
---> 10 mycursor.execute("CREATE TABLE srinu( id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100),age int,locat
ion VARCHAR(100))")

File ~\anaconda3\Lib\site-packages\mysql\connector\cursor_cext.py:357, in CMySQLCursor.execute(self, operation,
params, multi)
    352             raise ProgrammingError(
    353                 "Not all parameters were used in the SQL statement"
    354             )
    356 try:
--> 357     result = self._connection.cmd_query(
    358         stmt,
    359         raw=self._raw,
    360         buffered=self._buffered,
    361         raw_as_string=self._raw_as_string,
    362     )
    363 except MySQLInterfaceError as err:
    364     raise get_mysql_exception(
    365         msg=err.msg, errno=err.errno, sqlstate=err.sqlstate
    366     ) from err

File ~\anaconda3\Lib\site-packages\mysql\connector\opentelemetry\context_propagation.py:97, in with_context_prop
agation.<locals>.wrapper(cnx, *args, **kwargs)
     95 # pylint: disable=possibly-used-before-assignment
     96 if not OTEL_ENABLED or not cnx.otel_context_propagation:
---> 97     return method(cnx, *args, **kwargs)
     99 current_span = trace.get_current_span()
    100 tp_header = None

File ~\anaconda3\Lib\site-packages\mysql\connector\connection_cext.py:713, in CMySQLConnection.cmd_query(self, q
uery, raw, buffered, raw_as_string)
    705     self._cmysql.query(
    706         query,
    707         raw=raw,
   (...)
    710         query_attrs=self.query_attrs,
    711     )
    712 except MySQLInterfaceError as err:
--> 713     raise get_mysql_exception(
    714         err.errno, msg=err.msg, sqlstate=err.sqlstate
    715     ) from err
    716 except AttributeError as err:
    717     addr = (
    718         self._unix_socket if self._unix_socket else f"{self._host}:{self._port}"
    719     )

ProgrammingError: 1050 (42S01): Table 'srinu' already exists
```

In [ ]:
```
If the table already exists ,use the ALTER TABLE keyword
```

In [17]:
```
mycursor.execute("Alter table srinu Modify id INT auto_increment ")
```

In [53]:
```
Image.open('alter.png')
```

Out[53]:



In [ ]:
```
"Python MySQL Insert Into Table"
```

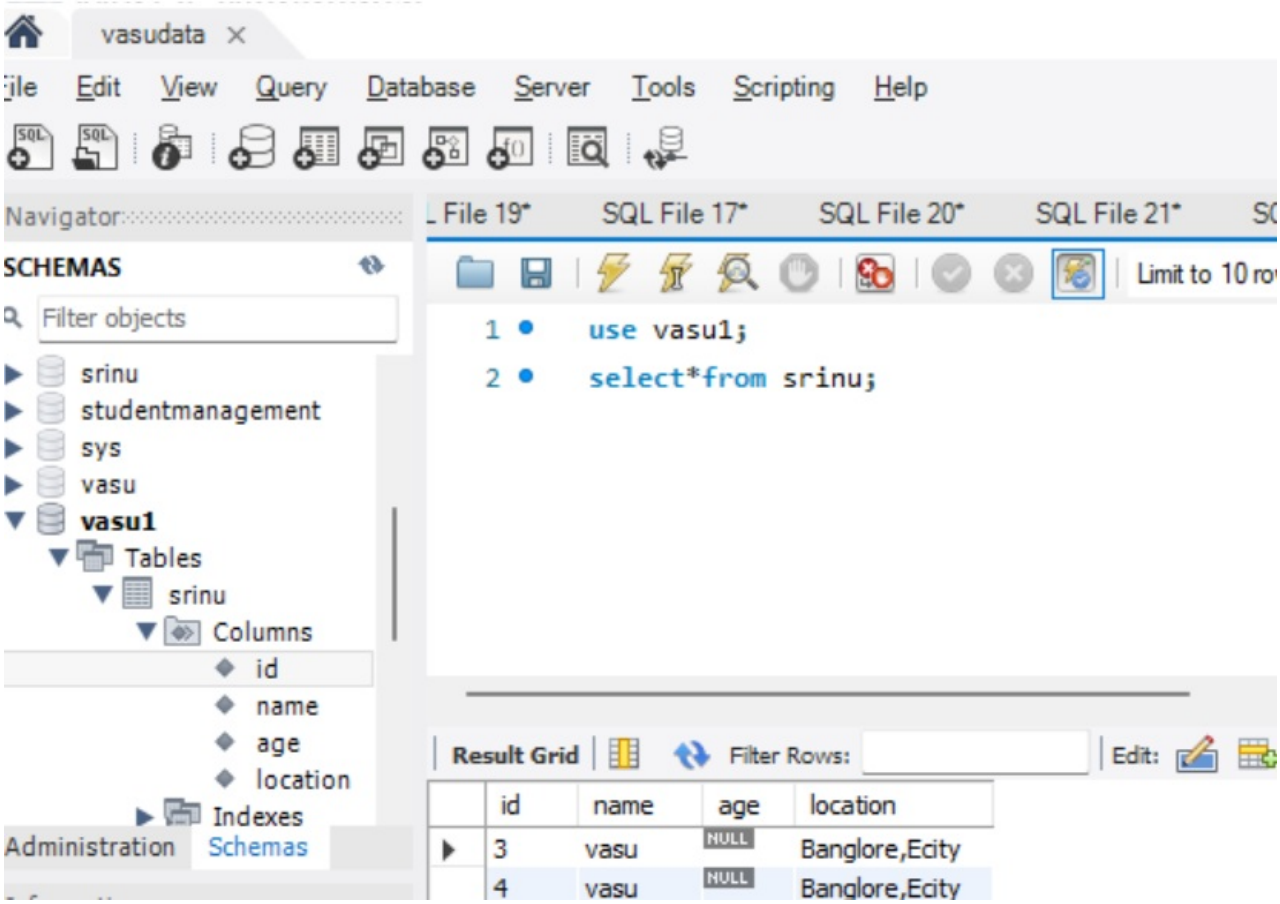
In [ ]:
```
to fill the table in MYSQl ,we have to use the "Insert INTO" statemnt:
```

In [33]:
```
import mysql.connector as conn
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

sql="INSERT INTO srinu (name,age,location) VALUES(%s,%s,%s)"

val=("vasu","21","Banglore,Ecity")

mycursor.execute(sql,val)

mydb.commit()

print(mycursor.rowcount,"record inserted")
```

```
1 record inserted
```

In [57]:
```
Image.open('value1.png')
```

Out[57]:



In [ ]:
```
"Important": Notice the statement: mydb.commit().It is required to make changes ,otherwise bo changes are made
```

In [ ]:
```
"Insert Multiple Rows"
```

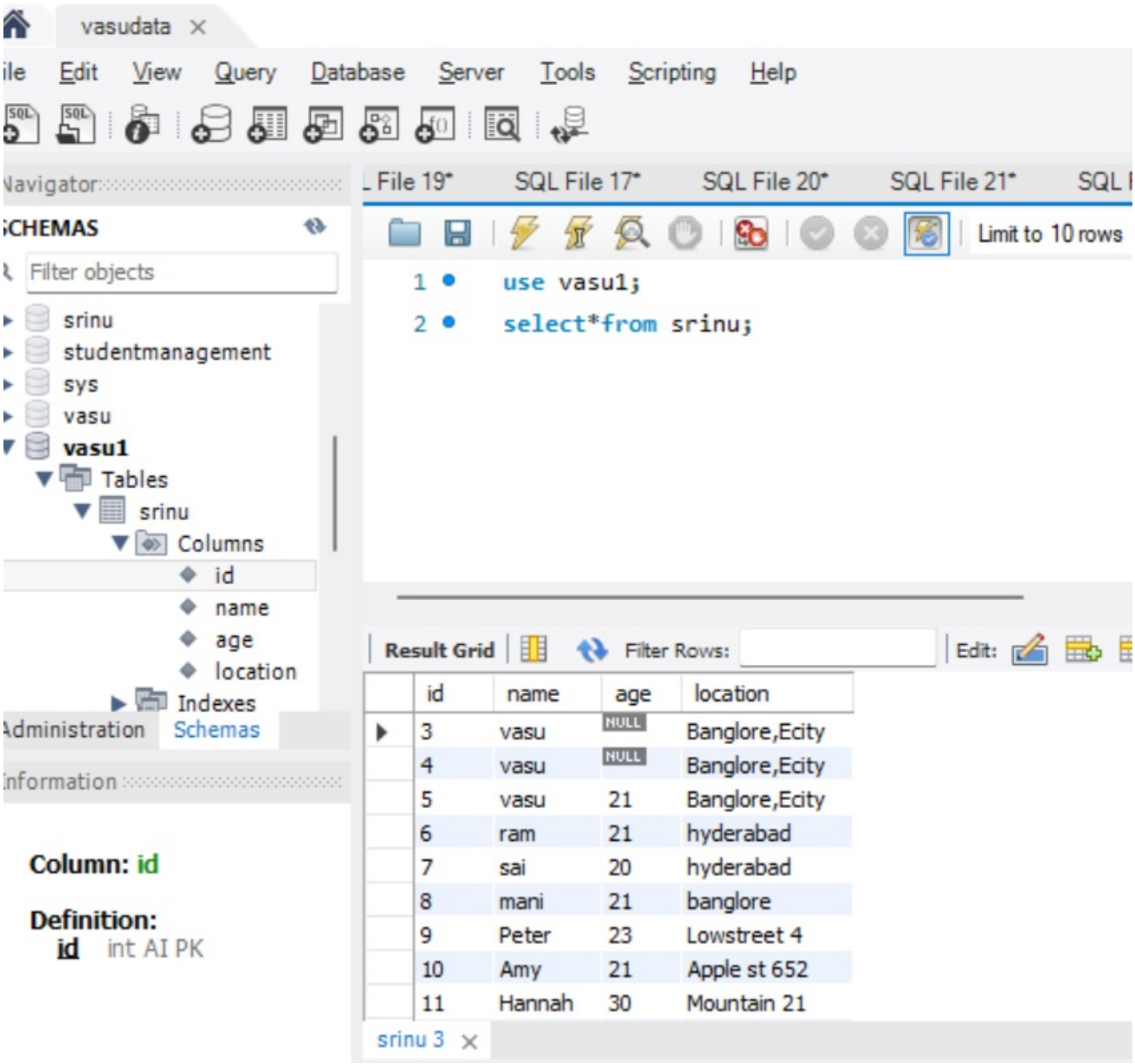
In [ ]:
```
TO insert multiple rows to the table ,we have to use the "executemany()" method.

THE second parameter of the "executemany()" method is list of tuples,containing the data you want to insert.
```

In [43]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
mycursor=mydb.cursor()

sql="INSERT INTO srinu (name,age,location) VALUES(%s,%s,%s)"
val=[
    ('ram',"21","hyderabad"),
    ('sai','20','hyderabad'),
    ('mani','21','banglore'),
    ('Peter','23', 'Lowstreet 4'),
  ('Amy', '21','Apple st 652'),
  ('Hannah', '30','Mountain 21'),
  ('Michael','23', 'Valley 345'),
  ('Sandy','21', 'Ocean blvd 2'),
  ('Betty', '32','Green Grass 1'),
  ('Richard', '32','Sky st 331'),
  ('Susan','24', 'One way 98'),
  ('Vicky','35', 'Yellow Garden 2'),
  ('Ben','35', 'Park Lane 38'),
  ('William','35', 'Central st 954'),
  ('Chuck','36', 'Main Road 989'),
  ('Viola','21', 'Sideway 1633')]

mycursor.executemany(sql,val)
mydb.commit()
print(mycursor.rowcount," rows were inserted")
```

```
16  rows were inserted
```

In [61]:
```
Image.open('values.png')
```

Out[61]:



In [ ]:
```
"Get Inserted ID"
```

In [ ]:
```
we can get "id" of the row you just inserted by asking the cursor object.
```

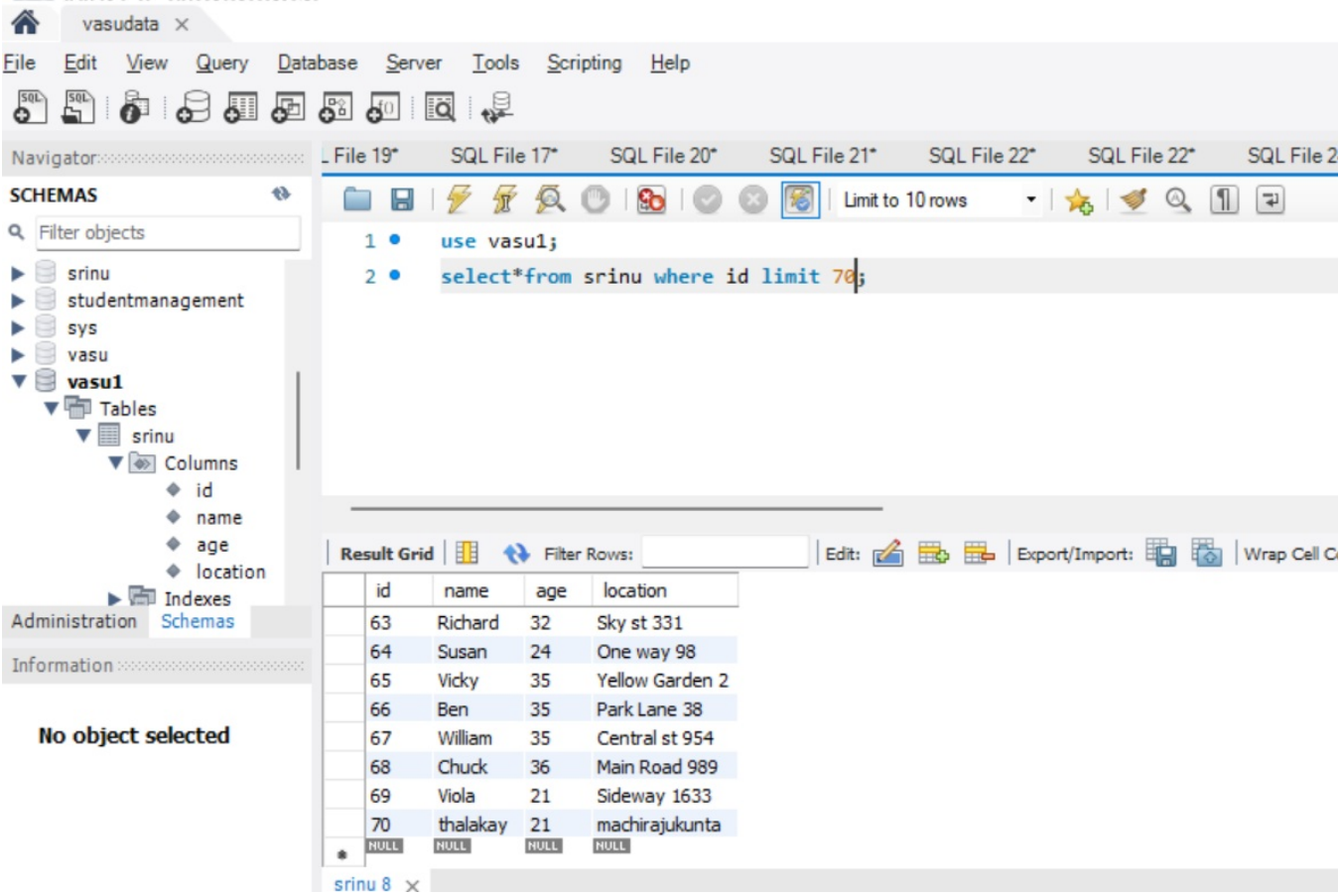
In [ ]:
```
NOte: If we inserted more then one row ,last inserted row will be returned.
```

In [63]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

sql="INSERT INTO srinu(name,age,location) VALUES(%s,%s,%s)"
val=("thalakay","21","machirajukunta")

mycursor.execute(sql,val)
mydb.commit()
print("1 record inserted ,ID:",mycursor.lastrowid)
```

```
1 record inserted ,ID: 70
```

In [65]:
```
Image.open('id.png')
```

Out[65]: MySQL Workbench



In [ ]: "Python MySQL select From"

In [ ]: "SELECT FROM a TABLE"

In [ ]: To select from a table in MySQL ,we have to use the "SELECT" statement:

In [91]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
mycursor=mydb.cursor()

mycursor.execute("SELECT * FROM  srinu")

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
(3, 'vasu', None, 'Banglore,Ecity')
(4, 'vasu', None, 'Banglore,Ecity')
(5, 'vasu', 21, 'Banglore,Ecity')
(6, 'ram', 21, 'hyderabad')
(7, 'sai', 20, 'hyderabad')
(8, 'mani', 21, 'banglore')
(9, 'Peter', 23, 'Lowstreet 4')
(10, 'Amy', 21, 'Apple st 652')
(11, 'Hannah', 30, 'Mountain 21')
(12, 'Michael', 23, 'Valley 345')
(13, 'Sandy', 21, 'Ocean blvd 2')
(14, 'Betty', 32, 'Green Grass 1')
(15, 'Richard', 32, 'Sky st 331')
(16, 'Susan', 24, 'One way 98')
(17, 'Vicky', 35, 'Yellow Garden 2')
(18, 'Ben', 35, 'Park Lane 38')
(19, 'William', 35, 'Central st 954')
(20, 'Chuck', 36, 'Main Road 989')
(21, 'Viola', 21, 'Sideway 1633')
(22, 'ram', 21, 'hyderabad')
(23, 'sai', 20, 'hyderabad')
(24, 'mani', 21, 'banglore')
(25, 'Peter', 23, 'Lowstreet 4')
(26, 'Amy', 21, 'Apple st 652')
(27, 'Hannah', 30, 'Mountain 21')
(28, 'Michael', 23, 'Valley 345')
(29, 'Sandy', 21, 'Ocean blvd 2')
(30, 'Betty', 32, 'Green Grass 1')
(31, 'Richard', 32, 'Sky st 331')
(32, 'Susan', 24, 'One way 98')
(33, 'Vicky', 35, 'Yellow Garden 2')
(34, 'Ben', 35, 'Park Lane 38')
(35, 'William', 35, 'Central st 954')
(36, 'Chuck', 36, 'Main Road 989')
(37, 'Viola', 21, 'Sideway 1633')
(38, 'ram', 21, 'hyderabad')
(39, 'sai', 20, 'hyderabad')
(40, 'mani', 21, 'banglore')
(41, 'Peter', 23, 'Lowstreet 4')
(42, 'Amy', 21, 'Apple st 652')
(43, 'Hannah', 30, 'Mountain 21')
(44, 'Michael', 23, 'Valley 345')
(45, 'Sandy', 21, 'Ocean blvd 2')
(46, 'Betty', 32, 'Green Grass 1')
(47, 'Richard', 32, 'Sky st 331')
(48, 'Susan', 24, 'One way 98')
(49, 'Vicky', 35, 'Yellow Garden 2')
(50, 'Ben', 35, 'Park Lane 38')
(51, 'William', 35, 'Central st 954')
(52, 'Chuck', 36, 'Main Road 989')
(53, 'Viola', 21, 'Sideway 1633')
(54, 'ram', 21, 'hyderabad')
(55, 'sai', 20, 'hyderabad')
(56, 'mani', 21, 'banglore')
(57, 'Peter', 23, 'Lowstreet 4')
(58, 'Amy', 21, 'Apple st 652')
(59, 'Hannah', 30, 'Mountain 21')
(60, 'Michael', 23, 'Valley 345')
(61, 'Sandy', 21, 'Ocean blvd 2')
(62, 'Betty', 32, 'Green Grass 1')
(63, 'Richard', 32, 'Sky st 331')
(64, 'Susan', 24, 'One way 98')
(65, 'Vicky', 35, 'Yellow Garden 2')
(66, 'Ben', 35, 'Park Lane 38')
(67, 'William', 35, 'Central st 954')
(68, 'Chuck', 36, 'Main Road 989')
(69, 'Viola', 21, 'Sideway 1633')
(70, 'thalakay', 21, 'machirajukunta')
```

In [ ]:
```
Note: WE yse the "fetchall()" method,which feteches all rows from the last executed statement.
```

In [ ]:
```
"Selecting Columns"
```

In [ ]:
```
To select onle some of the columns in a table,for that we have to use the "SELECT column_name from table".
```

In [97]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
```

```
mycursor=mydb.cursor()
mycursor.execute("SELECT name,age from srinu")

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
('vasu', None)
('vasu', None)
('vasu', 21)
('ram', 21)
('sai', 20)
('mani', 21)
('Peter', 23)
('Amy', 21)
('Hannah', 30)
('Michael', 23)
('Sandy', 21)
('Betty', 32)
('Richard', 32)
('Susan', 24)
('Vicky', 35)
('Ben', 35)
('William', 35)
('Chuck', 36)
('Viola', 21)
('ram', 21)
('sai', 20)
('mani', 21)
('Peter', 23)
('Amy', 21)
('Hannah', 30)
('Michael', 23)
('Sandy', 21)
('Betty', 32)
('Richard', 32)
('Susan', 24)
('Vicky', 35)
('Ben', 35)
('William', 35)
('Chuck', 36)
('Viola', 21)
('ram', 21)
('sai', 20)
('mani', 21)
('Peter', 23)
('Amy', 21)
('Hannah', 30)
('Michael', 23)
('Sandy', 21)
('Betty', 32)
('Richard', 32)
('Susan', 24)
('Vicky', 35)
('Ben', 35)
('William', 35)
('Chuck', 36)
('Viola', 21)
('ram', 21)
('sai', 20)
('mani', 21)
('Peter', 23)
('Amy', 21)
('Hannah', 30)
('Michael', 23)
('Sandy', 21)
('Betty', 32)
('Richard', 32)
('Susan', 24)
('Vicky', 35)
('Ben', 35)
('William', 35)
('Chuck', 36)
('Viola', 21)
('thalakay', 21)
```

```
In [ ]: "Using the fetchone() Method"
```

```
In [ ]: If we are interested to take only one one for that we have to use "fetchone" method

        THE "fetchone()" method returns the first row of the result.
```

In [101…

```python
import mysql.connector as conn
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

mycursor.execute("select *from srinu")

myresult=mycursor.fetchone()

for i in myresult:
    print(i)
```

```
3
vasu
None
Banglore,Ecity
```

In [ ]:

```python
"Python MySQL Where "
```

In [ ]:

```python
"select with a filter"
```

In [ ]:

```python
when selecting the records from a table,we can filter selection by using the "where" statement
```

In [105…

```python
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

sql="SELECT *FROM srinu WHERE  location='hyderabad'"

mycursor.execute(sql)

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
(6, 'ram', 21, 'hyderabad')
(7, 'sai', 20, 'hyderabad')
(22, 'ram', 21, 'hyderabad')
(23, 'sai', 20, 'hyderabad')
(38, 'ram', 21, 'hyderabad')
(39, 'sai', 20, 'hyderabad')
(54, 'ram', 21, 'hyderabad')
(55, 'sai', 20, 'hyderabad')
```

In [ ]:

```python
"Wildcard Characters"
```

In [ ]:

```python
we can also select the records that starts, includes, or ends with a given letter or phrase.

Use the %  to represent wildcard characters:
```

In [113…

```python
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

sql="SELECT * FROM srinu WHERE name like 's%'"

mycursor.execute(sql)

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
(7, 'sai', 20, 'hyderabad')
(13, 'Sandy', 21, 'Ocean blvd 2')
(16, 'Susan', 24, 'One way 98')
(23, 'sai', 20, 'hyderabad')
(29, 'Sandy', 21, 'Ocean blvd 2')
(32, 'Susan', 24, 'One way 98')
(39, 'sai', 20, 'hyderabad')
(45, 'Sandy', 21, 'Ocean blvd 2')
(48, 'Susan', 24, 'One way 98')
(55, 'sai', 20, 'hyderabad')
(61, 'Sandy', 21, 'Ocean blvd 2')
(64, 'Susan', 24, 'One way 98')
```

In [ ]:
```
"Prevent SQL Injection"
```

In [ ]:
```
when query values are provided by the user,we should escape the values.

This is to prevent SQl injections ,which is a common web hacking technique to destroy or misuse your databases.

THE mysql.connector module has methods to escape query values.
```

In [ ]:
```
Escape query values by using the placholder %s method:
```

In [129…
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

sql="SELECT * FROM srinu WHERE location = %s"

loc=('hyderabad',)

mycursor.execute(sql,loc)

myresult = mycursor.fetchall()

for i in myresult:
    print(i)
```

```
(6, 'ram', 21, 'hyderabad')
(7, 'sai', 20, 'hyderabad')
(22, 'ram', 21, 'hyderabad')
(23, 'sai', 20, 'hyderabad')
(38, 'ram', 21, 'hyderabad')
(39, 'sai', 20, 'hyderabad')
(54, 'ram', 21, 'hyderabad')
(55, 'sai', 20, 'hyderabad')
```

In [ ]:
```
"Python MYSQL Order by"
```

In [ ]:
```
"sort the result"
```

use the "OREDER BY" statement to sort the result in ascending or descending order.

The 'OREDER BY' keyword sorts the result ascending by default.To sort the result in descending order,use the DESC keyword

In [133…
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

sql="SELECT * from srinu ORDER BY name"

mycursor.execute(sql)

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
(26, 'Amy', 21, 'Apple st 652')
(42, 'Amy', 21, 'Apple st 652')
(10, 'Amy', 21, 'Apple st 652')
(58, 'Amy', 21, 'Apple st 652')
(66, 'Ben', 35, 'Park Lane 38')
(18, 'Ben', 35, 'Park Lane 38')
(50, 'Ben', 35, 'Park Lane 38')
(34, 'Ben', 35, 'Park Lane 38')
(46, 'Betty', 32, 'Green Grass 1')
(62, 'Betty', 32, 'Green Grass 1')
(30, 'Betty', 32, 'Green Grass 1')
(14, 'Betty', 32, 'Green Grass 1')
(20, 'Chuck', 36, 'Main Road 989')
(36, 'Chuck', 36, 'Main Road 989')
(68, 'Chuck', 36, 'Main Road 989')
(52, 'Chuck', 36, 'Main Road 989')
(11, 'Hannah', 30, 'Mountain 21')
(43, 'Hannah', 30, 'Mountain 21')
(59, 'Hannah', 30, 'Mountain 21')
(27, 'Hannah', 30, 'Mountain 21')
(24, 'mani', 21, 'banglore')
(40, 'mani', 21, 'banglore')
(56, 'mani', 21, 'banglore')
(8, 'mani', 21, 'banglore')
(12, 'Michael', 23, 'Valley 345')
(28, 'Michael', 23, 'Valley 345')
(60, 'Michael', 23, 'Valley 345')
(44, 'Michael', 23, 'Valley 345')
(9, 'Peter', 23, 'Lowstreet 4')
(25, 'Peter', 23, 'Lowstreet 4')
(41, 'Peter', 23, 'Lowstreet 4')
(57, 'Peter', 23, 'Lowstreet 4')
(6, 'ram', 21, 'hyderabad')
(54, 'ram', 21, 'hyderabad')
(38, 'ram', 21, 'hyderabad')
(22, 'ram', 21, 'hyderabad')
(47, 'Richard', 32, 'Sky st 331')
(15, 'Richard', 32, 'Sky st 331')
(31, 'Richard', 32, 'Sky st 331')
(63, 'Richard', 32, 'Sky st 331')
(7, 'sai', 20, 'hyderabad')
(39, 'sai', 20, 'hyderabad')
(23, 'sai', 20, 'hyderabad')
(55, 'sai', 20, 'hyderabad')
(61, 'Sandy', 21, 'Ocean blvd 2')
(13, 'Sandy', 21, 'Ocean blvd 2')
(29, 'Sandy', 21, 'Ocean blvd 2')
(45, 'Sandy', 21, 'Ocean blvd 2')
(32, 'Susan', 24, 'One way 98')
(16, 'Susan', 24, 'One way 98')
(48, 'Susan', 24, 'One way 98')
(64, 'Susan', 24, 'One way 98')
(70, 'thalakay', 21, 'machirajukunta')
(5, 'vasu', 21, 'Banglore,Ecity')
(4, 'vasu', None, 'Banglore,Ecity')
(3, 'vasu', None, 'Banglore,Ecity')
(33, 'Vicky', 35, 'Yellow Garden 2')
(17, 'Vicky', 35, 'Yellow Garden 2')
(65, 'Vicky', 35, 'Yellow Garden 2')
(49, 'Vicky', 35, 'Yellow Garden 2')
(53, 'Viola', 21, 'Sideway 1633')
(21, 'Viola', 21, 'Sideway 1633')
(37, 'Viola', 21, 'Sideway 1633')
(69, 'Viola', 21, 'Sideway 1633')
(51, 'William', 35, 'Central st 954')
(19, 'William', 35, 'Central st 954')
(67, 'William', 35, 'Central st 954')
(35, 'William', 35, 'Central st 954')
```

In [ ]:
```
"ORDER BY DESC"
```

In [ ]:
```
Use the DESC keyword to sort the result in a descending order.
```

In [135…
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()
```

```python
sql="SELECT * from srinu ORDER BY name DESC"

mycursor.execute(sql)

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
(19, 'William', 35, 'Central st 954')
(67, 'William', 35, 'Central st 954')
(51, 'William', 35, 'Central st 954')
(35, 'William', 35, 'Central st 954')
(69, 'Viola', 21, 'Sideway 1633')
(37, 'Viola', 21, 'Sideway 1633')
(53, 'Viola', 21, 'Sideway 1633')
(21, 'Viola', 21, 'Sideway 1633')
(65, 'Vicky', 35, 'Yellow Garden 2')
(17, 'Vicky', 35, 'Yellow Garden 2')
(33, 'Vicky', 35, 'Yellow Garden 2')
(49, 'Vicky', 35, 'Yellow Garden 2')
(4, 'vasu', None, 'Banglore,Ecity')
(5, 'vasu', 21, 'Banglore,Ecity')
(3, 'vasu', None, 'Banglore,Ecity')
(70, 'thalakay', 21, 'machirajukunta')
(48, 'Susan', 24, 'One way 98')
(64, 'Susan', 24, 'One way 98')
(16, 'Susan', 24, 'One way 98')
(32, 'Susan', 24, 'One way 98')
(29, 'Sandy', 21, 'Ocean blvd 2')
(45, 'Sandy', 21, 'Ocean blvd 2')
(13, 'Sandy', 21, 'Ocean blvd 2')
(61, 'Sandy', 21, 'Ocean blvd 2')
(7, 'sai', 20, 'hyderabad')
(55, 'sai', 20, 'hyderabad')
(23, 'sai', 20, 'hyderabad')
(39, 'sai', 20, 'hyderabad')
(31, 'Richard', 32, 'Sky st 331')
(15, 'Richard', 32, 'Sky st 331')
(47, 'Richard', 32, 'Sky st 331')
(63, 'Richard', 32, 'Sky st 331')
(6, 'ram', 21, 'hyderabad')
(38, 'ram', 21, 'hyderabad')
(22, 'ram', 21, 'hyderabad')
(54, 'ram', 21, 'hyderabad')
(57, 'Peter', 23, 'Lowstreet 4')
(9, 'Peter', 23, 'Lowstreet 4')
(25, 'Peter', 23, 'Lowstreet 4')
(41, 'Peter', 23, 'Lowstreet 4')
(60, 'Michael', 23, 'Valley 345')
(44, 'Michael', 23, 'Valley 345')
(12, 'Michael', 23, 'Valley 345')
(28, 'Michael', 23, 'Valley 345')
(8, 'mani', 21, 'banglore')
(56, 'mani', 21, 'banglore')
(40, 'mani', 21, 'banglore')
(24, 'mani', 21, 'banglore')
(43, 'Hannah', 30, 'Mountain 21')
(59, 'Hannah', 30, 'Mountain 21')
(27, 'Hannah', 30, 'Mountain 21')
(11, 'Hannah', 30, 'Mountain 21')
(36, 'Chuck', 36, 'Main Road 989')
(52, 'Chuck', 36, 'Main Road 989')
(20, 'Chuck', 36, 'Main Road 989')
(68, 'Chuck', 36, 'Main Road 989')
(14, 'Betty', 32, 'Green Grass 1')
(30, 'Betty', 32, 'Green Grass 1')
(46, 'Betty', 32, 'Green Grass 1')
(62, 'Betty', 32, 'Green Grass 1')
(18, 'Ben', 35, 'Park Lane 38')
(34, 'Ben', 35, 'Park Lane 38')
(50, 'Ben', 35, 'Park Lane 38')
(66, 'Ben', 35, 'Park Lane 38')
(10, 'Amy', 21, 'Apple st 652')
(58, 'Amy', 21, 'Apple st 652')
(42, 'Amy', 21, 'Apple st 652')
(26, 'Amy', 21, 'Apple st 652')
```

In [ ]:
```python
"Python MYSQL DELETE from BY"
```

In [ ]:
```python
"Delete Record"
```

In [ ]:
```python
we can delete records from the table existing table by using the 'DELETE FROM' statement
```

In [143…

```python
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

sql="DELETE FROM srinu where location = 'banglore'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount,"records(s) deleted")
```

```
4 records(s) deleted
```

Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

Notice the WHERE clause in the DELETE syntax: The WHERE clause specifies which record(s) that should be deleted. If you omit the WHERE clause, all records will be deleted!

In [ ]:

```python
"Prevent SQL Injection"
```

In [ ]:

```python
It is a considered a good practice to escape the values of any query,also in delete statements.

This is prevent SQL injections ,which is a common web hacking technique to destroy or misuse your database .

the mysql.connector module uses the placeholder '%s' to escape values in the delete statement.
```

In [13]:

```python
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
mycursor=mydb.cursor()

sql="DELETE from srinu where location=%s"
val=('hyderabad',)

mycursor.execute(sql,val)

mydb.commit()
print(mycursor.rowcount,"reccords are delted")
```

```
0 reccords are delted
```

In [33]:

```python
from PIL import Image

Image.open('delete.png')
```

Out[33]:

Result Grid | Filter Rows: | Edit:

id	name	age	location
3	vasu	NULL	Banglore,Ecity
4	vasu	NULL	Banglore,Ecity
5	vasu	21	Banglore,Ecity
9	Peter	23	Lowstreet 4
10	Amy	21	Apple st 652
11	Hannah	30	Mountain 21
12	Michael	23	Valley 345
13	Sandy	21	Ocean blvd 2
14	Betty	32	Green Grass 1

srinu 1

In [21]:

```python
"Python MySQL Drop Table"
```

In [ ]:

```python
"Delete a table"
```

we can delete an existing table by using the "DROP table" statement.

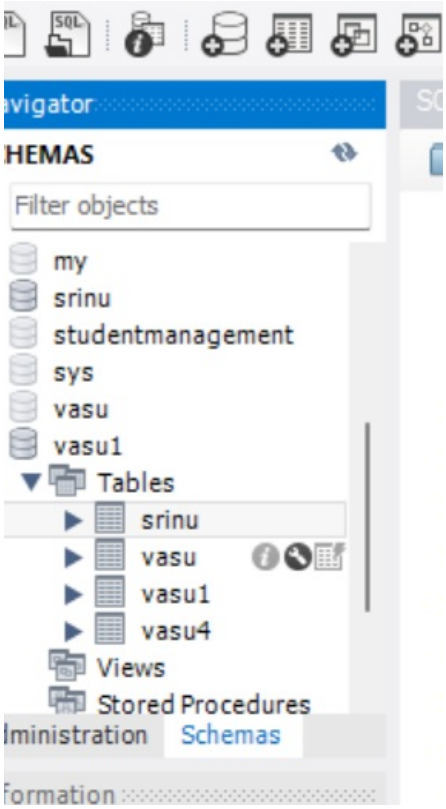
In [23]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
mycursor=mydb.cursor()

sql="show tables"


mycursor.execute(sql)

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
('srinu',)
('vasu',)
('vasu1',)
('vasu2',)
('vasu3',)
('vasu4',)
```

In [31]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
mycursor=mydb.cursor()

sql="DROP table vasu3"


mycursor.execute(sql)


mydb.commit()
```

In [37]:
```
Image.open('drop tabele.png')
```

Out[37]:



In [ ]:
```
"Drop Only if Exist"
```

In [ ]:
```
If the table you want to delete is already deleted ,or for any other reason does not exist ,you can use the IF
```

In [39]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
mycursor=mydb.cursor()

sql="DROP table if exists vasu2"


mycursor.execute(sql)


mydb.commit()
```

If this page was executed with no error(s), you have successfully deleted the "customers" table.

In [ ]:
```
"Python MySQL Update Table"
```

In [ ]:
```
"UPdate Table"
```

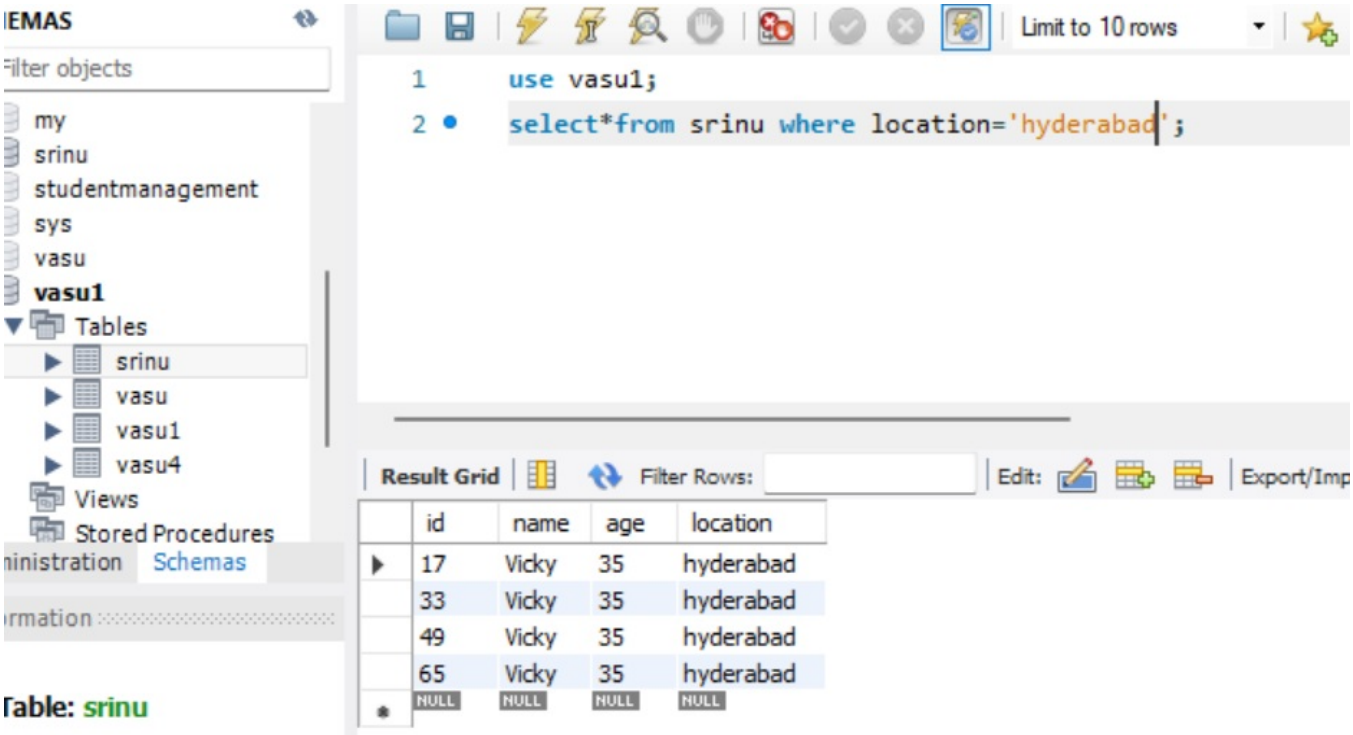
In [ ]:
```
we can update the existing recoords in a table by using the "UPDATE" statement
```

In [42]:
```
import mysql.connector as conn
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)

mycursor=mydb.cursor()

sql="Update srinu set location='hyderabad' where location='Yellow Garden 2'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount,"record(s) affected")
```

```
4 record(s) affected
```

In [44]:
```
Image.open('update.png')
```

Out[44]:



Important!: Notice the statement: mydb.commit(). It is required to make the changes, otherwise no changes are made to the table.

Notice the WHERE clause in the UPDATE syntax: The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

In [ ]:
```
"Prevent SQL Injection"
```

It is considered a good practice to escape the values of any query, also in update statements.

This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.

The mysql.connector module uses the placeholder %s to escape values in the update statement:

In [76]:
```
import mysql.connector as conn
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)

mycursor=mydb.cursor()

sql = "UPDATE srinu SET location = %s WHERE location = %s"
val = ("Valley 345", "Canyon 123")
mycursor.execute(sql,val)

mydb.commit()

print(mycursor.rowcount,"record(s) affected")
```

```
0 record(s) affected
```

In [ ]:
```
"Python MySQL LImit"
```

In [ ]:
```
we can limit the number of records from the query ,by using the "LIMIT" statement.
```

In [80]:
```
import mysql.connector as conn
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)

mycursor=mydb.cursor()

sql = "select * from srinu limit 5"

mycursor.execute(sql)

myresult=mycursor.fetchall()
```

```
for i in myresult:
    print(i)
```

```
(3, 'vasu', None, 'Banglore,Ecity')
(4, 'vasu', None, 'Banglore,Ecity')
(5, 'vasu', 21, 'Banglore,Ecity')
(9, 'Peter', 23, 'Lowstreet 4')
(10, 'Amy', 21, 'Apple st 652')
```

In [ ]:
```
"Start From Another Position"
```

If we wana to return five records starting from the third records,you can use the "Offset" keyword

In [82]:
```
import mysql.connector as conn
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)

mycursor=mydb.cursor()

sql = "select * from srinu limit 5 offset 3"

mycursor.execute(sql)

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
(9, 'Peter', 23, 'Lowstreet 4')
(10, 'Amy', 21, 'Apple st 652')
(11, 'Hannah', 30, 'Mountain 21')
(12, 'Michael', 23, 'Valley 345')
(13, 'Sandy', 21, 'Ocean blvd 2')
```

In [ ]:
```
"Python Mysql Join"
```

In [ ]:
```
"Join Two or More Tables"
```

we can combine rows from two or more tables , based on a related column between them ,by using a JOIN statement.

In [97]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu122"
)

mycursor=mydb.cursor()

sql="select c.firstname,c.address,c1.location from customers as c inner join country as c1 on c.city=c1.city"

mycursor.execute(sql)

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
('Bob', '101 Maple St', 'wer')
('David', '303 Cedar St', 'mrk')
('Leo', '202 Cedar St', 'wer')
('Nina', '404 Fir St', 'mrk')
('Vince', '303 Fir St', 'wer')
('Xander', '505 Maple St', 'mrk')
('Fred', '404 Oak St', 'wer')
('Holly', '606 Cedar St', 'mrk')
('Bob', '101 Maple St', 'wer')
('David', '303 Cedar St', 'mrk')
('Leo', '202 Cedar St', 'wer')
('Nina', '404 Fir St', 'mrk')
('Vince', '303 Fir St', 'wer')
('Xander', '505 Maple St', 'mrk')
('Fred', '404 Oak St', 'wer')
('Holly', '606 Cedar St', 'mrk')
('Bob', '101 Maple St', 'wer')
('David', '303 Cedar St', 'mrk')
('Leo', '202 Cedar St', 'wer')
('Nina', '404 Fir St', 'mrk')
('Vince', '303 Fir St', 'wer')
('Xander', '505 Maple St', 'mrk')
('Fred', '404 Oak St', 'wer')
('Holly', '606 Cedar St', 'mrk')
```

In [ ]:

```
"LEFT JOIN"
```

In the example above, Hannah, and Michael were excluded from the result, that is because INNER JOIN only shows the records where there is a match.

If you want to show all users, even if they do not have a favorite product, use the LEFT JOIN statement:

In [100…

```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu122"
)

mycursor=mydb.cursor()

sql="select c.firstname,c.address,c1.location from customers as c left join country as c1 on c.city=c1.city"

mycursor.execute(sql)

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
('John', '123 Elm St', None)
('Jane', '456 Oak St', None)
('Alice', '789 Pine St', None)
('Bob', '101 Maple St', 'wer')
('Charlie', '202 Birch St', None)
('David', '303 Cedar St', 'mrk')
('Emma', '404 Spruce St', None)
('Frank', '505 Fir St', None)
('Grace', '606 Pine St', None)
('Hank', '707 Maple St', None)
('Ivy', '808 Elm St', None)
('Jack', '909 Oak St', None)
('Karen', '101 Birch St', None)
('Leo', '202 Cedar St', 'wer')
('Mia', '303 Spruce St', None)
('Nina', '404 Fir St', 'mrk')
('Oscar', '505 Pine St', None)
('Paul', '606 Maple St', None)
('Quinn', '707 Elm St', None)
('Rose', '808 Oak St', None)
('Sam', '909 Birch St', None)
('Tina', '101 Cedar St', None)
('Uma', '202 Spruce St', None)
('Vince', '303 Fir St', 'wer')
('Wendy', '404 Pine St', None)
('Xander', '505 Maple St', 'mrk')
('Yara', '606 Elm St', None)
('Zane', '707 Oak St', None)
('Amy', '808 Birch St', None)
('Brian', '909 Cedar St', None)
('Clara', '101 Pine St', None)
```

```
('Dan', '202 Maple St', None)
('Ella', '303 Elm St', None)
('Fred', '404 Oak St', 'wer')
('Gina', '505 Birch St', None)
('Holly', '606 Cedar St', 'mrk')
('Ian', '707 Pine St', None)
('Jackie', '808 Maple St', None)
('Kevin', '909 Elm St', None)
('John', '123 Elm St', None)
('Jane', '456 Oak St', None)
('Alice', '789 Pine St', None)
('Bob', '101 Maple St', 'wer')
('Charlie', '202 Birch St', None)
('David', '303 Cedar St', 'mrk')
('Emma', '404 Spruce St', None)
('Frank', '505 Fir St', None)
('Grace', '606 Pine St', None)
('Hank', '707 Maple St', None)
('Ivy', '808 Elm St', None)
('Jack', '909 Oak St', None)
('Karen', '101 Birch St', None)
('Leo', '202 Cedar St', 'wer')
('Mia', '303 Spruce St', None)
('Nina', '404 Fir St', 'mrk')
('Oscar', '505 Pine St', None)
('Paul', '606 Maple St', None)
('Quinn', '707 Elm St', None)
('Rose', '808 Oak St', None)
('Sam', '909 Birch St', None)
('Tina', '101 Cedar St', None)
('Uma', '202 Spruce St', None)
('Vince', '303 Fir St', 'wer')
('Wendy', '404 Pine St', None)
('Xander', '505 Maple St', 'mrk')
('Yara', '606 Elm St', None)
('Zane', '707 Oak St', None)
('Amy', '808 Birch St', None)
('Brian', '909 Cedar St', None)
('Clara', '101 Pine St', None)
('Dan', '202 Maple St', None)
('Ella', '303 Elm St', None)
('Fred', '404 Oak St', 'wer')
('Gina', '505 Birch St', None)
('Holly', '606 Cedar St', 'mrk')
('Ian', '707 Pine St', None)
('Jackie', '808 Maple St', None)
('Kevin', '909 Elm St', None)
('John', '123 Elm St', None)
('Jane', '456 Oak St', None)
('Alice', '789 Pine St', None)
('Bob', '101 Maple St', 'wer')
('Charlie', '202 Birch St', None)
('David', '303 Cedar St', 'mrk')
('Emma', '404 Spruce St', None)
('Frank', '505 Fir St', None)
('Grace', '606 Pine St', None)
('Hank', '707 Maple St', None)
('Ivy', '808 Elm St', None)
('Jack', '909 Oak St', None)
('Karen', '101 Birch St', None)
('Leo', '202 Cedar St', 'wer')
('Mia', '303 Spruce St', None)
('Nina', '404 Fir St', 'mrk')
('Oscar', '505 Pine St', None)
('Paul', '606 Maple St', None)
('Quinn', '707 Elm St', None)
('Rose', '808 Oak St', None)
('Sam', '909 Birch St', None)
('Tina', '101 Cedar St', None)
('Uma', '202 Spruce St', None)
('Vince', '303 Fir St', 'wer')
('Wendy', '404 Pine St', None)
('Xander', '505 Maple St', 'mrk')
('Yara', '606 Elm St', None)
('Zane', '707 Oak St', None)
('Amy', '808 Birch St', None)
('Brian', '909 Cedar St', None)
('Clara', '101 Pine St', None)
('Dan', '202 Maple St', None)
('Ella', '303 Elm St', None)
('Fred', '404 Oak St', 'wer')
('Gina', '505 Birch St', None)
('Holly', '606 Cedar St', 'mrk')
```

```
('Ian', '707 Pine St', None)
('Jackie', '808 Maple St', None)
('Kevin', '909 Elm St', None)
```

In [ ]:
```
"Right JOIN"
```

If you want to return all products, and the users who have them as their favorite, even if no user have them as their favorite, use the RIGHT JOIN statement:

In [102…
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu122"
)

mycursor=mydb.cursor()

sql="select c.firstname,c.address,c1.location from customers as c right join country as c1 on c.city=c1.city"

mycursor.execute(sql)

myresult=mycursor.fetchall()

for i in myresult:
    print(i)
```

```
('Holly', '606 Cedar St', 'mrk')
('Xander', '505 Maple St', 'mrk')
('Nina', '404 Fir St', 'mrk')
('David', '303 Cedar St', 'mrk')
('Holly', '606 Cedar St', 'mrk')
('Xander', '505 Maple St', 'mrk')
('Nina', '404 Fir St', 'mrk')
('David', '303 Cedar St', 'mrk')
('Holly', '606 Cedar St', 'mrk')
('Xander', '505 Maple St', 'mrk')
('Nina', '404 Fir St', 'mrk')
('David', '303 Cedar St', 'mrk')
('Fred', '404 Oak St', 'wer')
('Vince', '303 Fir St', 'wer')
('Leo', '202 Cedar St', 'wer')
('Bob', '101 Maple St', 'wer')
('Fred', '404 Oak St', 'wer')
('Vince', '303 Fir St', 'wer')
('Leo', '202 Cedar St', 'wer')
('Bob', '101 Maple St', 'wer')
('Fred', '404 Oak St', 'wer')
('Vince', '303 Fir St', 'wer')
('Leo', '202 Cedar St', 'wer')
('Bob', '101 Maple St', 'wer')
(None, None, 'dff')
```

In [ ]:
```
"Handling Errors in MySQL with"
```

When interacting with a MYSQL database using,errors can occurs due to various reasons like connections issues ,SQL syntax errors,or attempting to perform operations on non-existing tables or records ."try-except" block can be used to handle these errors gracefuuly.

In [ ]:
```
Example 1. Handling Errors During Database Operations
```

In [ ]:
```
import mysql.connector as conn

from mysql.connector import Error

try:
    connection=conn.connect(
        host="localhost",
        user="root",
        password="Gsrinu@789",
        database="vasu1"

    )
    if connection.is_connected():
        cursor=connection.cursor()
        sql_insert="Insert into srinu (id,name,age,location) values (%s,%s,%s,%s)"
        values=(71,'srinu',21,'sanikavaram')
        cursor.execute(sql_insert,values)
        connection.commit()
        print("Record inserted successfully into employees table")
```

```
except mysql.connector.Error as error:
    print(f"Failed to insert record into MYSQL table:{error}")

finally:
    if connection.is_connected():
         cursor.close()
         connection.close()
        print("MYSQL connection is closed")
```

In [ ]:

```
Explanation:

1.try: the code block under try is executed.If any error occurs,the control is passed to the except block.

2.except: Catches the mysql.connector.Error and print an error messege .this prevents the program from crashing

3.finally: This block is executed no matter what ,weather an error occured or not.It os often used to close the
```

In [24]:

```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
mycursor=mydb.cursor()

sql="SELECT *FROM srinu"

mycursor.execute(sql)

myresult=mycursor.fetchall()
for i in myresult:
    print(f"Id:{i[0]} ,name:{i[1]} ,age:{i[2]} ,location:{i[3]}")

mycursor.close()
connection.close()
```

```
Id:3 ,name:vasu ,age:None ,location:Banglore,Ecity
Id:4 ,name:vasu ,age:None ,location:Banglore,Ecity
Id:5 ,name:vasu ,age:21 ,location:Banglore,Ecity
Id:9 ,name:Peter ,age:23 ,location:Lowstreet 4
Id:10 ,name:Amy ,age:21 ,location:Apple st 652
Id:11 ,name:Hannah ,age:30 ,location:Mountain 21
Id:12 ,name:Michael ,age:23 ,location:Valley 345
Id:13 ,name:Sandy ,age:21 ,location:Ocean blvd 2
Id:14 ,name:Betty ,age:32 ,location:Green Grass 1
Id:15 ,name:Richard ,age:32 ,location:Sky st 331
Id:16 ,name:Susan ,age:24 ,location:One way 98
Id:17 ,name:Vicky ,age:35 ,location:hyderabad
Id:18 ,name:Ben ,age:35 ,location:Park Lane 38
Id:19 ,name:William ,age:35 ,location:Central st 954
Id:20 ,name:Chuck ,age:36 ,location:Main Road 989
Id:21 ,name:Viola ,age:21 ,location:Sideway 1633
Id:25 ,name:Peter ,age:23 ,location:Lowstreet 4
Id:26 ,name:Amy ,age:21 ,location:Apple st 652
Id:27 ,name:Hannah ,age:30 ,location:Mountain 21
Id:28 ,name:Michael ,age:23 ,location:Valley 345
Id:29 ,name:Sandy ,age:21 ,location:Ocean blvd 2
Id:30 ,name:Betty ,age:32 ,location:Green Grass 1
Id:31 ,name:Richard ,age:32 ,location:Sky st 331
Id:32 ,name:Susan ,age:24 ,location:One way 98
Id:33 ,name:Vicky ,age:35 ,location:hyderabad
Id:34 ,name:Ben ,age:35 ,location:Park Lane 38
Id:35 ,name:William ,age:35 ,location:Central st 954
Id:36 ,name:Chuck ,age:36 ,location:Main Road 989
Id:37 ,name:Viola ,age:21 ,location:Sideway 1633
Id:41 ,name:Peter ,age:23 ,location:Lowstreet 4
Id:42 ,name:Amy ,age:21 ,location:Apple st 652
Id:43 ,name:Hannah ,age:30 ,location:Mountain 21
Id:44 ,name:Michael ,age:23 ,location:Valley 345
Id:45 ,name:Sandy ,age:21 ,location:Ocean blvd 2
Id:46 ,name:Betty ,age:32 ,location:Green Grass 1
Id:47 ,name:Richard ,age:32 ,location:Sky st 331
Id:48 ,name:Susan ,age:24 ,location:One way 98
Id:49 ,name:Vicky ,age:35 ,location:hyderabad
Id:50 ,name:Ben ,age:35 ,location:Park Lane 38
Id:51 ,name:William ,age:35 ,location:Central st 954
Id:52 ,name:Chuck ,age:36 ,location:Main Road 989
Id:53 ,name:Viola ,age:21 ,location:Sideway 1633
Id:57 ,name:Peter ,age:23 ,location:Lowstreet 4
Id:58 ,name:Amy ,age:21 ,location:Apple st 652
Id:59 ,name:Hannah ,age:30 ,location:Mountain 21
Id:60 ,name:Michael ,age:23 ,location:Valley 345
Id:61 ,name:Sandy ,age:21 ,location:Ocean blvd 2
Id:62 ,name:Betty ,age:32 ,location:Green Grass 1
Id:63 ,name:Richard ,age:32 ,location:Sky st 331
Id:64 ,name:Susan ,age:24 ,location:One way 98
Id:65 ,name:Vicky ,age:35 ,location:hyderabad
Id:66 ,name:Ben ,age:35 ,location:Park Lane 38
Id:67 ,name:William ,age:35 ,location:Central st 954
Id:68 ,name:Chuck ,age:36 ,location:Main Road 989
Id:69 ,name:Viola ,age:21 ,location:Sideway 1633
Id:70 ,name:thalakay ,age:21 ,location:machirajukunta
Id:71 ,name:srinu ,age:21 ,location:sanikavaram
```

In [ ]:

```
"Update operation"
```

In [39]:

```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
cursor=mydb.cursor()

sql="update srinu set age=%s where age <%s"

values=(23,30)

cursor.execute(sql,values)

mydb.commit()

print(f"{cursor.rowcount} records are afftected")

cursor.close()
mydb.close()
```

```
47 records are afftected
```

In [43]:

```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
mycursor=mydb.cursor()

sql="SELECT age FROM srinu"

mycursor.execute(sql)

myresult=mycursor.fetchall()
for i in myresult:
    print(i)
mycursor.close()
connection.close()
```

```
(None,)
(None,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
(23,)
```

In [ ]:

```
"delete operation "
```

In [60]:

```
import mysql.connector as conn
```

```
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"

)
mycursor=mydb.cursor()

sql="delete from srinu where age=%s"
value=(23,)

mycursor.execute(sql,value)
mydb.commit()
print(f"{mycursor.rowcount} records are deleted ")
mycursor.close()
mydb.close()
```

```
55 records are deleted
```

In [ ]:
```
"commit()"
```

The commit() functions is used to save all the changes made to the database during a transaction.Without calling commit(),the changes will not be saved in the datbasse ,which could lead to data inconsistencies.

In [ ]:
```
Key Points:
 1. Always call commit() function after INSERT,UPDATE,DELETE operetions.
 2.If commit() is not called ,the changes are not persisited to the database
```

In [5]:
```
import mysql.connector as conn
connection=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="customers")
```

In [31]:
```
cursor=connection.cursor()
cursor.execute("select*from customer3 where customer_id=1")
result=cursor.fetchall()
for row in result:
    print(row)
```

```
(1, 'vasu', 'Doe', 'john.doe@example.com', '555-0101', '123 Elm Street', 'Springfield', 'IL', '62701')
```

In [33]:
```
sql_update=("Update customer3 set first_name=%s where customer_id=%s")
values=("vasu","1")
cursor.execute(sql_update,value)
```

In [35]:
```
connection.commit()
```

In [69]:
```
import mysql.connector as conn
connection=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="customers")
```

In [71]:
```
data=['41', 'saibabu', 'Brown', 'william.brown@example.com', '555-0105', '202 Cedar Street', 'Springfield', 'IL

for i in range(len(data)):
    print(data[i])
```

```
41
saibabu
Brown
william.brown@example.com
555-0105
202 Cedar Street
Springfield
IL
62701
```

In [73]:
```
sql_insert="""Insert into customer3(customer_id ,
    first_name,
    last_name ,
    email,
    phone ,
    address ,
    city ,
    state ,
    zip_code )  values(%s,%s,%s,%s,%s,%s,%s,%s,%s)"""
```

In [75]:
```
cursor=connection.cursor()
```

In [ ]:
```
cursor.execute(sql_insert,data)
```

In [85]:
```
connection.commit()
```

In [87]:
```
print(mycursor.rowcount,"insrted succesfully")
```

```
1 insrted succesfully
```

In [62]:
```
import pandas as pd
df=pd.read_csv('data.csv')
df
```

Out[62]:

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
...	...	...	...	...
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

169 rows × 4 columns

whenever we load data to sql database we need clean data deffinatly we do not clean the data it throws the error showing null values.

In [91]:
```
df1=df.dropna()
df1
```

Out[91]:

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
...	...	...	...	...
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

164 rows × 4 columns

In [105…
```
for _,row in df1.iterrows():
    print(tuple(row))
```

```
(60.0, 110.0, 130.0, 409.1)
(60.0, 117.0, 145.0, 479.0)
(60.0, 103.0, 135.0, 340.0)
(45.0, 109.0, 175.0, 282.4)
(45.0, 117.0, 148.0, 406.0)
(60.0, 102.0, 127.0, 300.0)
(60.0, 110.0, 136.0, 374.0)
(45.0, 104.0, 134.0, 253.3)
(30.0, 109.0, 133.0, 195.1)
(60.0, 98.0, 124.0, 269.0)
(60.0, 103.0, 147.0, 329.3)
(60.0, 100.0, 120.0, 250.7)
(60.0, 106.0, 128.0, 345.3)
(60.0, 104.0, 132.0, 379.3)
```

```
(60.0, 98.0, 123.0, 275.0)
(60.0, 98.0, 120.0, 215.2)
(60.0, 100.0, 120.0, 300.0)
(60.0, 103.0, 123.0, 323.0)
(45.0, 97.0, 125.0, 243.0)
(60.0, 108.0, 131.0, 364.2)
(45.0, 100.0, 119.0, 282.0)
(60.0, 130.0, 101.0, 300.0)
(45.0, 105.0, 132.0, 246.0)
(60.0, 102.0, 126.0, 334.5)
(60.0, 100.0, 120.0, 250.0)
(60.0, 92.0, 118.0, 241.0)
(60.0, 100.0, 132.0, 280.0)
(60.0, 102.0, 129.0, 380.3)
(60.0, 92.0, 115.0, 243.0)
(45.0, 90.0, 112.0, 180.1)
(60.0, 101.0, 124.0, 299.0)
(60.0, 93.0, 113.0, 223.0)
(60.0, 107.0, 136.0, 361.0)
(60.0, 114.0, 140.0, 415.0)
(60.0, 102.0, 127.0, 300.0)
(60.0, 100.0, 120.0, 300.0)
(60.0, 100.0, 120.0, 300.0)
(45.0, 104.0, 129.0, 266.0)
(45.0, 90.0, 112.0, 180.1)
(60.0, 98.0, 126.0, 286.0)
(60.0, 100.0, 122.0, 329.4)
(60.0, 111.0, 138.0, 400.0)
(60.0, 111.0, 131.0, 397.0)
(60.0, 99.0, 119.0, 273.0)
(60.0, 109.0, 153.0, 387.6)
(45.0, 111.0, 136.0, 300.0)
(45.0, 108.0, 129.0, 298.0)
(60.0, 111.0, 139.0, 397.6)
(60.0, 107.0, 136.0, 380.2)
(80.0, 123.0, 146.0, 643.1)
(60.0, 106.0, 130.0, 263.0)
(60.0, 118.0, 151.0, 486.0)
(30.0, 136.0, 175.0, 238.0)
(60.0, 121.0, 146.0, 450.7)
(60.0, 118.0, 121.0, 413.0)
(45.0, 115.0, 144.0, 305.0)
(20.0, 153.0, 172.0, 226.4)
(45.0, 123.0, 152.0, 321.0)
(210.0, 108.0, 160.0, 1376.0)
(160.0, 110.0, 137.0, 1034.4)
(160.0, 109.0, 135.0, 853.0)
(45.0, 118.0, 141.0, 341.0)
(20.0, 110.0, 130.0, 131.4)
(180.0, 90.0, 130.0, 800.4)
(150.0, 105.0, 135.0, 873.4)
(150.0, 107.0, 130.0, 816.0)
(20.0, 106.0, 136.0, 110.4)
(300.0, 108.0, 143.0, 1500.2)
(150.0, 97.0, 129.0, 1115.0)
(60.0, 109.0, 153.0, 387.6)
(90.0, 100.0, 127.0, 700.0)
(150.0, 97.0, 127.0, 953.2)
(45.0, 114.0, 146.0, 304.0)
(90.0, 98.0, 125.0, 563.2)
(45.0, 105.0, 134.0, 251.0)
(45.0, 110.0, 141.0, 300.0)
(120.0, 100.0, 130.0, 500.4)
(270.0, 100.0, 131.0, 1729.0)
(30.0, 159.0, 182.0, 319.2)
(45.0, 149.0, 169.0, 344.0)
(30.0, 103.0, 139.0, 151.1)
(120.0, 100.0, 130.0, 500.0)
(45.0, 100.0, 120.0, 225.3)
(30.0, 151.0, 170.0, 300.0)
(45.0, 102.0, 136.0, 234.0)
(120.0, 100.0, 157.0, 1000.1)
(45.0, 129.0, 103.0, 242.0)
(20.0, 83.0, 107.0, 50.3)
(180.0, 101.0, 127.0, 600.1)
(30.0, 90.0, 107.0, 105.3)
(15.0, 80.0, 100.0, 50.5)
(20.0, 150.0, 171.0, 127.4)
(20.0, 151.0, 168.0, 229.4)
(30.0, 95.0, 128.0, 128.2)
(25.0, 152.0, 168.0, 244.2)
(30.0, 109.0, 131.0, 188.2)
(90.0, 93.0, 124.0, 604.1)
```

```
(20.0, 95.0, 112.0, 77.7)
(90.0, 90.0, 110.0, 500.0)
(90.0, 90.0, 100.0, 500.0)
(90.0, 90.0, 100.0, 500.4)
(30.0, 92.0, 108.0, 92.7)
(30.0, 93.0, 128.0, 124.0)
(180.0, 90.0, 120.0, 800.3)
(30.0, 90.0, 120.0, 86.2)
(90.0, 90.0, 120.0, 500.3)
(210.0, 137.0, 184.0, 1860.4)
(60.0, 102.0, 124.0, 325.2)
(45.0, 107.0, 124.0, 275.0)
(15.0, 124.0, 139.0, 124.2)
(45.0, 100.0, 120.0, 225.3)
(60.0, 108.0, 131.0, 367.6)
(60.0, 108.0, 151.0, 351.7)
(60.0, 116.0, 141.0, 443.0)
(60.0, 97.0, 122.0, 277.4)
(60.0, 103.0, 124.0, 332.7)
(30.0, 112.0, 137.0, 193.9)
(45.0, 100.0, 120.0, 100.7)
(60.0, 119.0, 169.0, 336.7)
(60.0, 107.0, 127.0, 344.9)
(60.0, 111.0, 151.0, 368.5)
(60.0, 98.0, 122.0, 271.0)
(60.0, 97.0, 124.0, 275.3)
(60.0, 109.0, 127.0, 382.0)
(90.0, 99.0, 125.0, 466.4)
(60.0, 114.0, 151.0, 384.0)
(60.0, 104.0, 134.0, 342.5)
(60.0, 107.0, 138.0, 357.5)
(60.0, 103.0, 133.0, 335.0)
(60.0, 106.0, 132.0, 327.5)
(60.0, 103.0, 136.0, 339.0)
(20.0, 136.0, 156.0, 189.0)
(45.0, 117.0, 143.0, 317.7)
(45.0, 115.0, 137.0, 318.0)
(45.0, 113.0, 138.0, 308.0)
(20.0, 141.0, 162.0, 222.4)
(60.0, 108.0, 135.0, 390.0)
(45.0, 100.0, 120.0, 250.4)
(45.0, 122.0, 149.0, 335.4)
(60.0, 136.0, 170.0, 470.2)
(45.0, 106.0, 126.0, 270.8)
(60.0, 107.0, 136.0, 400.0)
(60.0, 112.0, 146.0, 361.9)
(30.0, 103.0, 127.0, 185.0)
(60.0, 110.0, 150.0, 409.4)
(60.0, 106.0, 134.0, 343.0)
(60.0, 109.0, 129.0, 353.2)
(60.0, 109.0, 138.0, 374.0)
(30.0, 150.0, 167.0, 275.8)
(60.0, 105.0, 128.0, 328.0)
(60.0, 111.0, 151.0, 368.5)
(60.0, 97.0, 131.0, 270.4)
(60.0, 100.0, 120.0, 270.4)
(60.0, 114.0, 150.0, 382.8)
(30.0, 80.0, 120.0, 240.9)
(30.0, 85.0, 120.0, 250.4)
(45.0, 90.0, 130.0, 260.4)
(45.0, 95.0, 130.0, 270.0)
(45.0, 100.0, 140.0, 280.9)
(60.0, 105.0, 140.0, 290.8)
(60.0, 110.0, 145.0, 300.0)
(60.0, 115.0, 145.0, 310.2)
(75.0, 120.0, 150.0, 320.4)
(75.0, 125.0, 150.0, 330.4)
```

In [97]:

```
for _,row in df1.iterrows():
    mycursor.execute(insert_data,tuple(row))
    mydb.commit()
```

In [72]:

```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

sql="create table srinu1( Duration int ,Pulse int,Maxpulse varchar(20),Calaries varchar(20))"
```

```
mycursor.execute(sql)
```

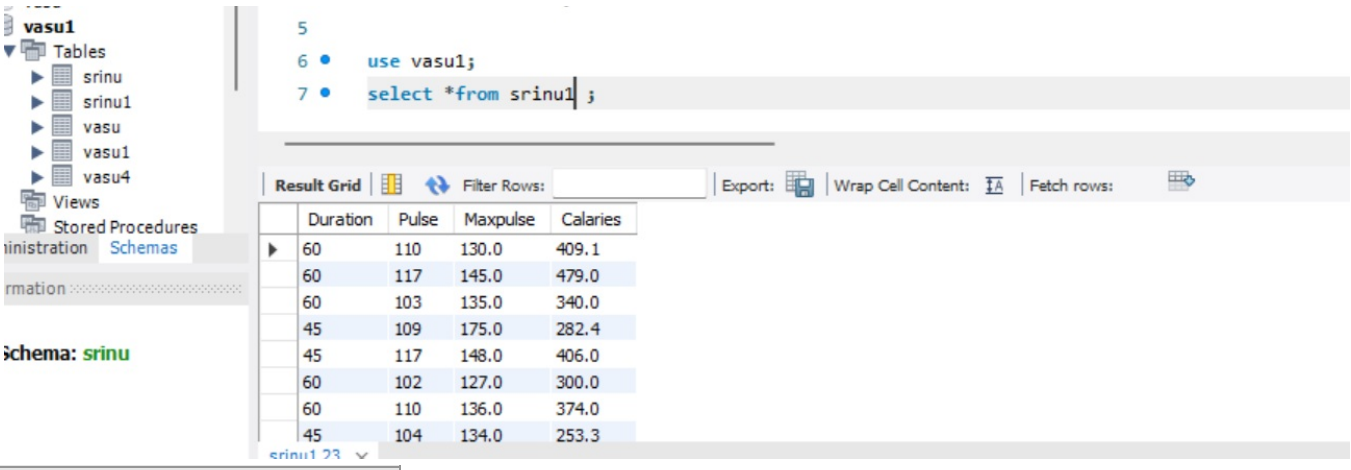
In [83]:
```
import mysql.connector as conn

mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

sql="show tables"
mycursor.execute(sql)

for i in mycursor:
    print(i)
```

```
('srinu',)
('srinu1',)
('vasu',)
('vasu1',)
('vasu4',)
```

In [99]:
```
import mysql.connector as conn
import pandas as pd
mydb=conn.connect(
    host="localhost",
    user="root",
    password="Gsrinu@789",
    database="vasu1"
)
mycursor=mydb.cursor()

insert_data="insert into srinu1 values(%s,%s,%s,%s)"
for _,row in df1.iterrows():
    mycursor.execute(insert_data,tuple(row))
    mydb.commit()
```

In [103…
```
from PIL import Image

Image.open('data23.png')
```

Out[103…



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js