

Author: Oluwaseun Tope Oyero

Title: Working with MySQL and Python

Installing MySQL Workbench

- Download MySQL application. Go to page: <https://dev.mysql.com/downloads/installer/>

Click on the link highlighted below:

④ MySQL Community Downloads

◀ MySQL Installer

General Availability (GA) Releases
Archives
不远

MySQL Installer 8.0.19

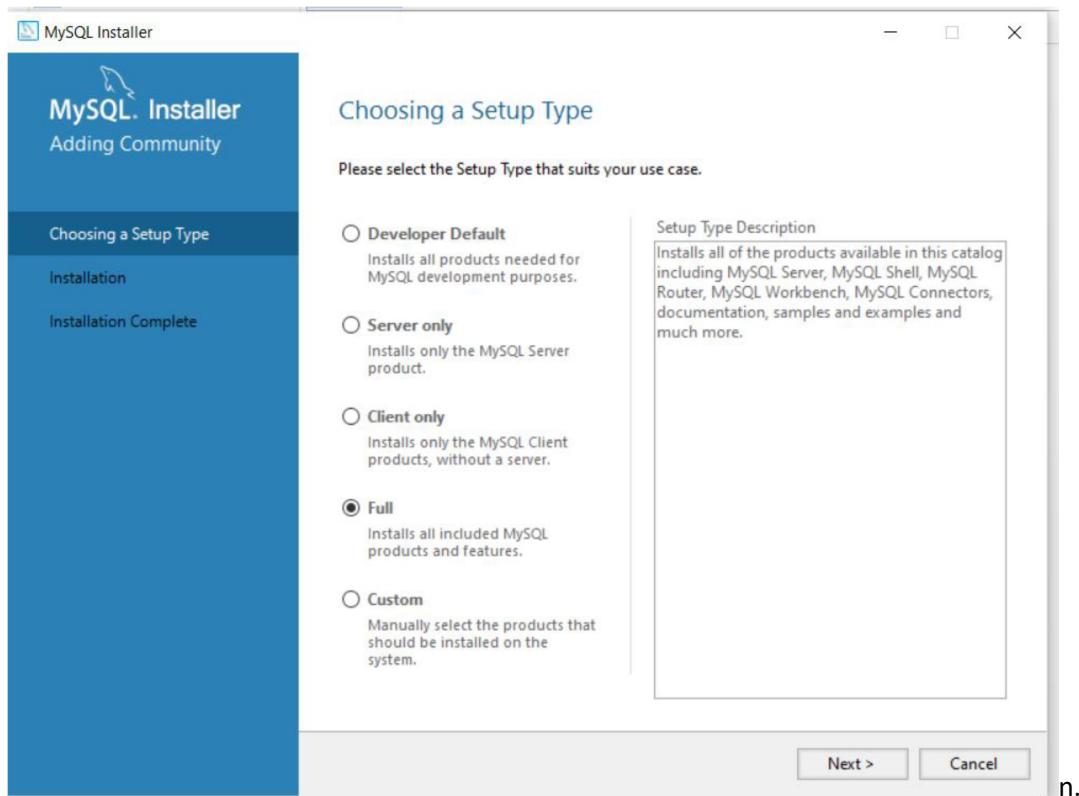
Select Operating System: Microsoft Windows

Looking for previous GA versions?

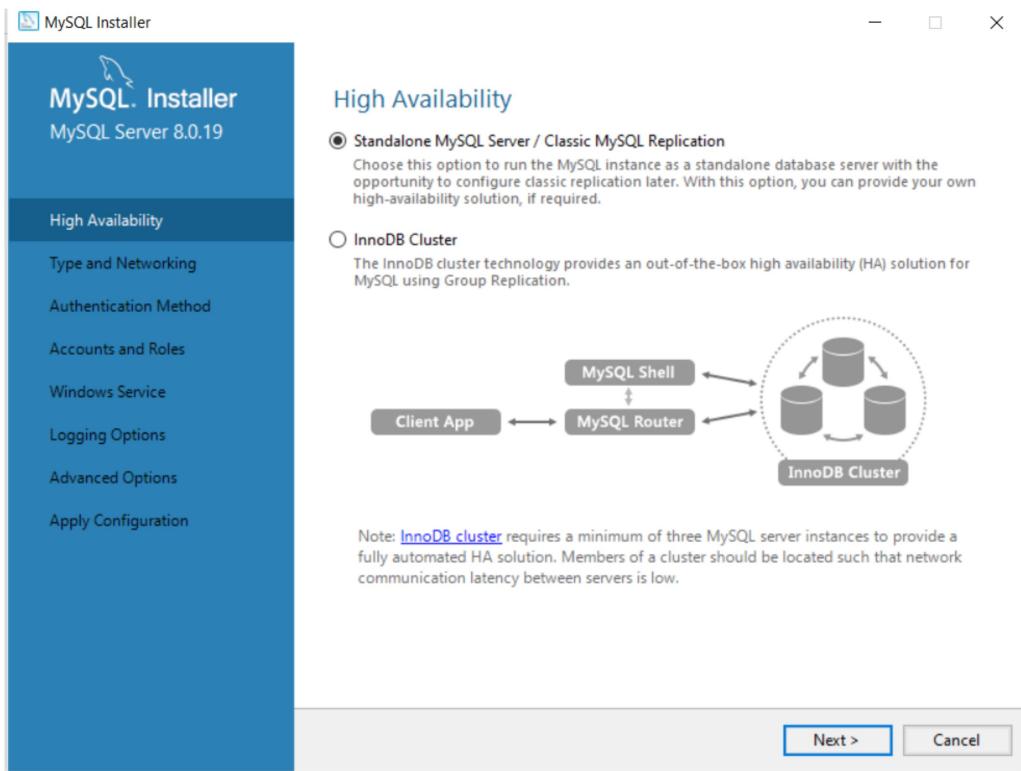
Windows (x86, 32-bit), MSI Installer	8.0.19	18.6M	Download
(mysql-installer-web-community-8.0.19.0.msi)		MD5: 32043776cb2239db45fddaa86dc0ad61 Signature	
Windows (x86, 32-bit), MSI Installer	8.0.19	398.9M	Download
(mysql-installer-community-8.0.19.0.msi)		MD5: 1a882015da7fb93f28c4717e63b6817c Signature	

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

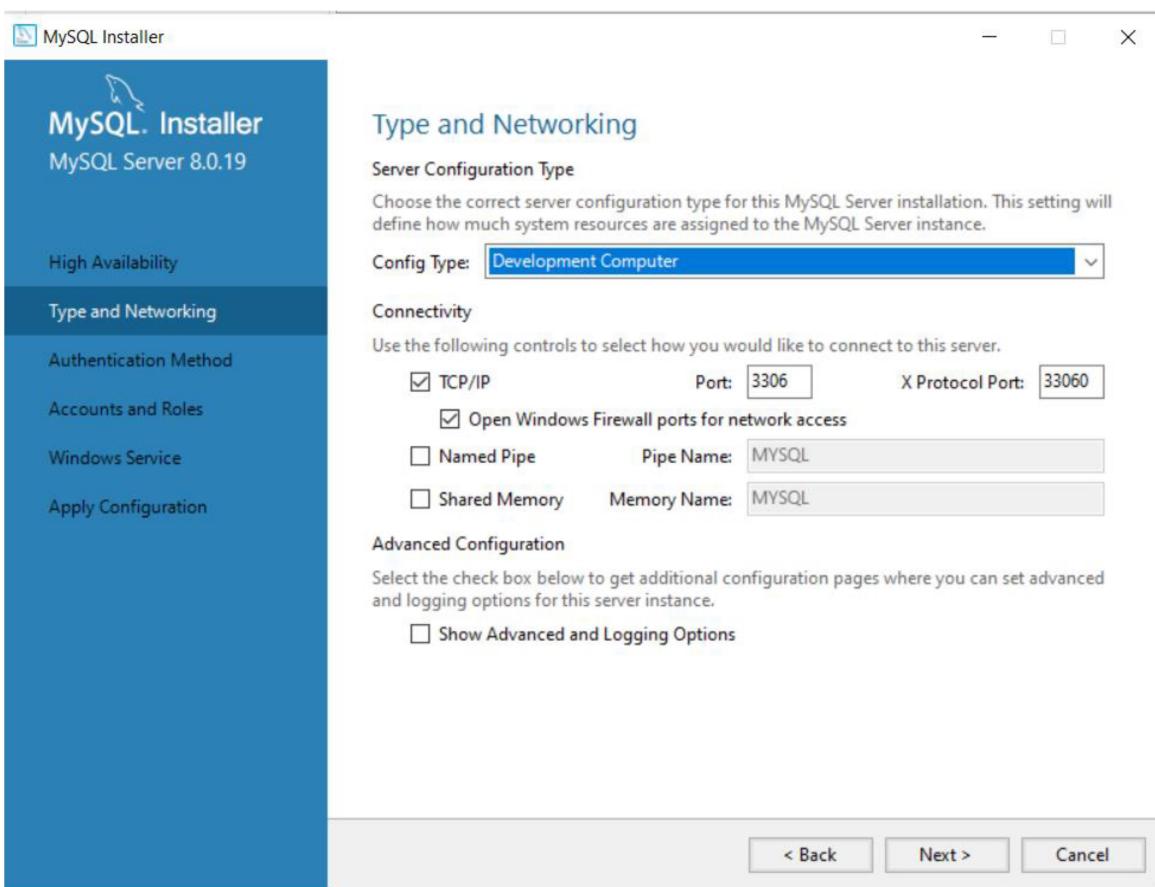
- Once downloaded, install the application. Follow the instructions and continue with installation.

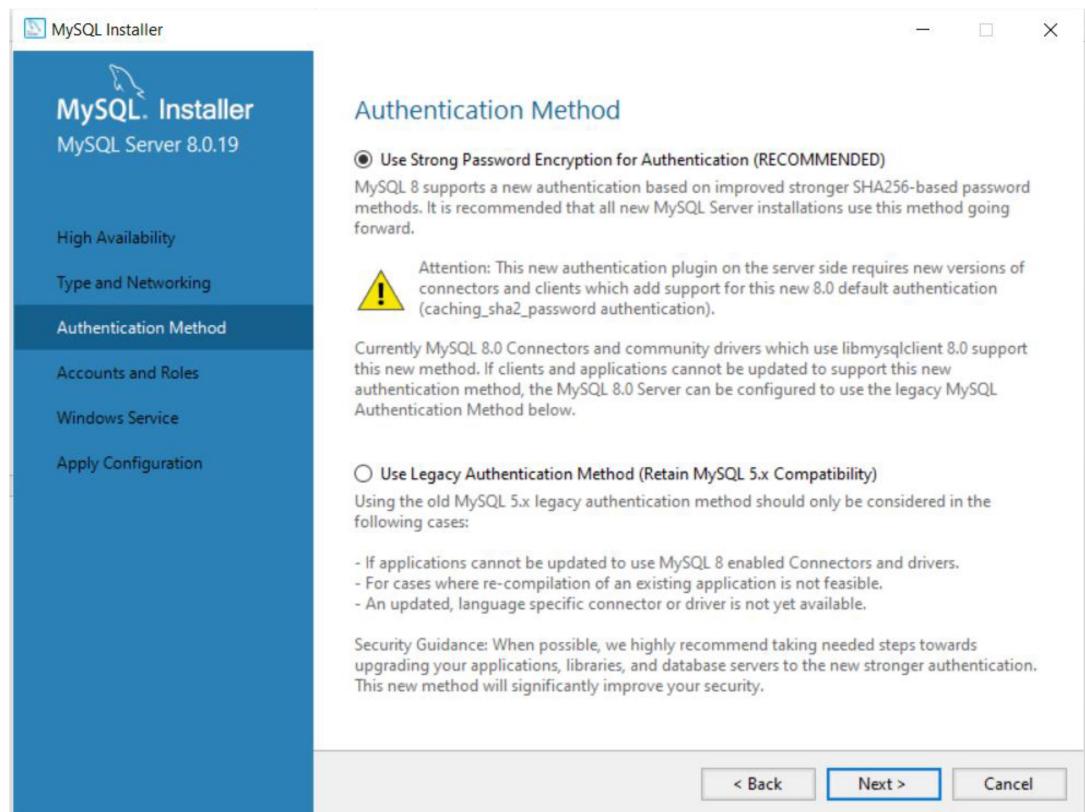


- Select Standalone MySQL server option.

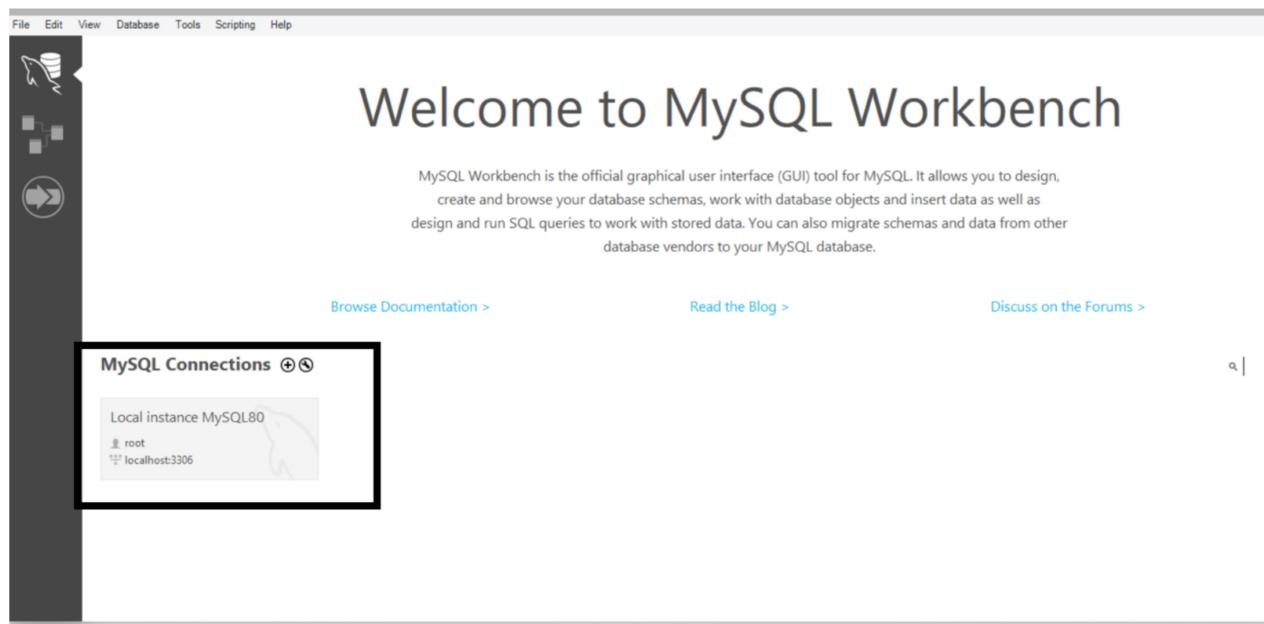


- Keep the default values for “Type and Networking” section.



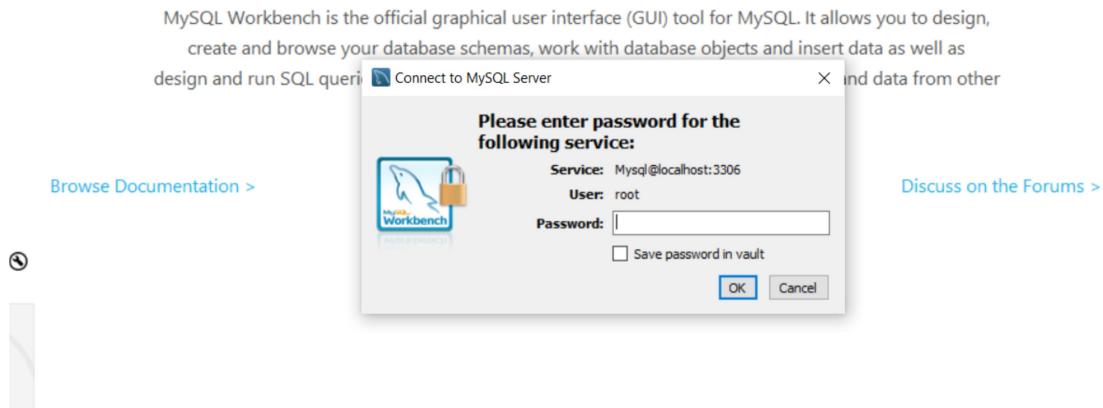


- Finish the installation by pressing next.
- Once the installation is complete, open the MySQL workbench and click on the local instance connection. The window will look something like this:



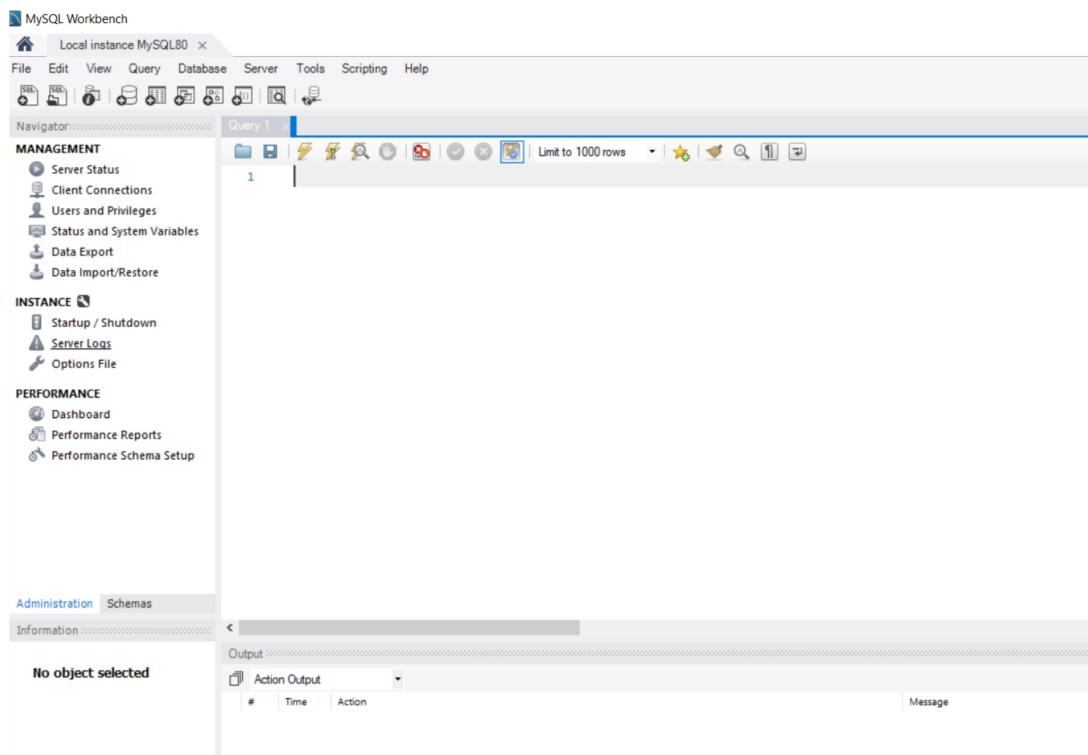
- Provide the password that you setup during installation:

Welcome to MySQL Workbench



Remember the user name mentioned above as we will need it to connect python to Mysql.

- Once the connection is established, you will see a window like this:



Connecting MySQL With Python

Let's use python to connect with this database.

- First we need to install “mysql-connector-python” package to establish a connection with Mysql.

“`pip install mysql-connector-python`”

```
(pyMysql) D:\Project_\PythonToMySQL>pip install mysql-connector-python
Collecting mysql-connector-python
  Using cached mysql_connector_python-8.0.19-cp36-cp36m-win_amd64.whl (4
Requirement already satisfied: protobuf==3.6.1 in c:\programdata\anacond
Requirement already satisfied: dnspython==1.16.0 in c:\programdata\anaco
Requirement already satisfied: six>=1.9 in c:\programdata\anaconda3\envs
Requirement already satisfied: setuptools in c:\programdata\anaconda3\en
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-8.0.19
```

- Once the package is installed, we can go ahead with establishing the connection.

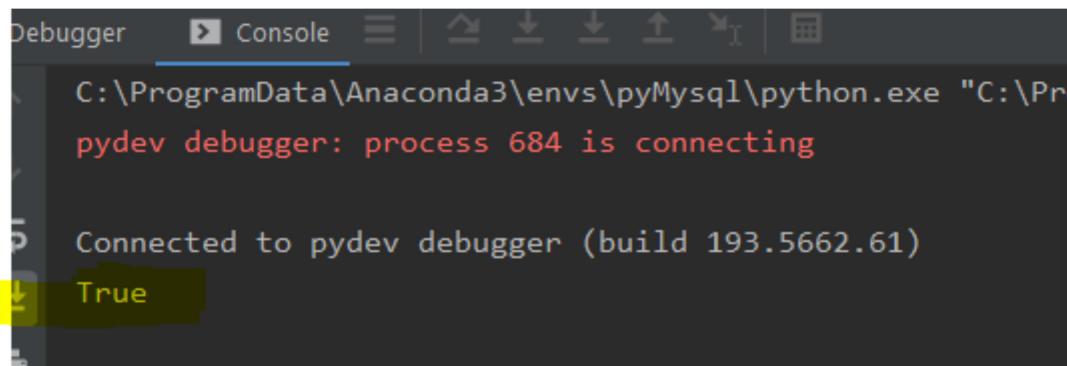
```
import mysql.connector as connection

try:
    mydb = connection.connect(host="localhost", user="root", passwd="password", use_pure=True)
    # check if the connection is established
    print(mydb.is_connected())
    mydb.close()
except Exception as e:
    print(str(e))
```

- Enter the details as shown above, your mysql server is running locally so host is “localhost”, enter the **username** and **password** as was setup during installation.

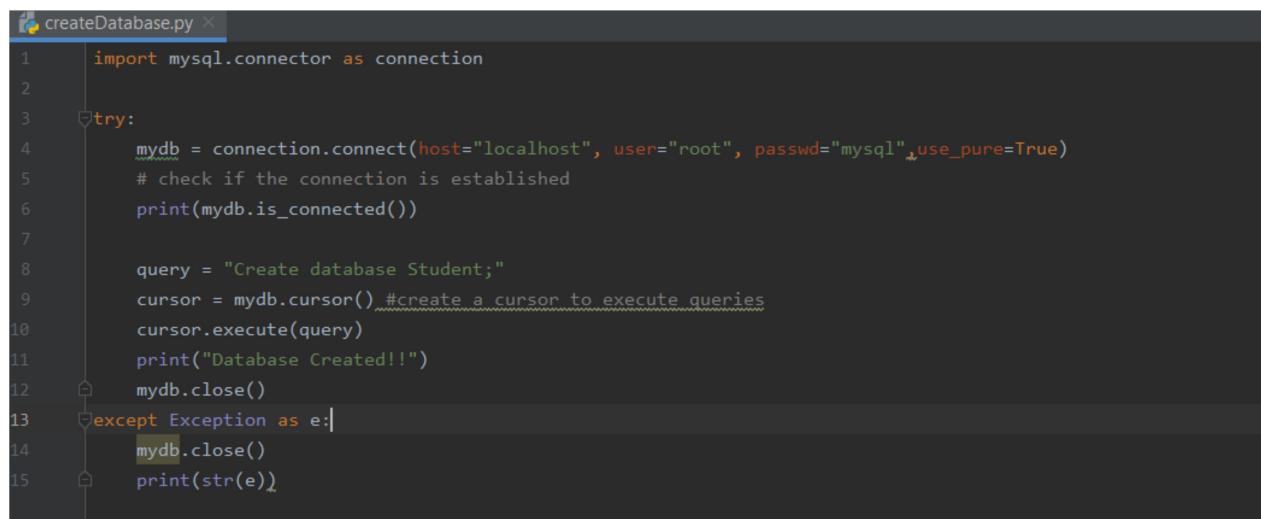
Note: “use_pure” argument forces mysqlConnector to user pure python connection instead of C extensions which leads to SSL error.

- To check if the connection is established, we can use `print (mydb.is_connected)`. It will return **TRUE** if the connection is established else **FALSE**.

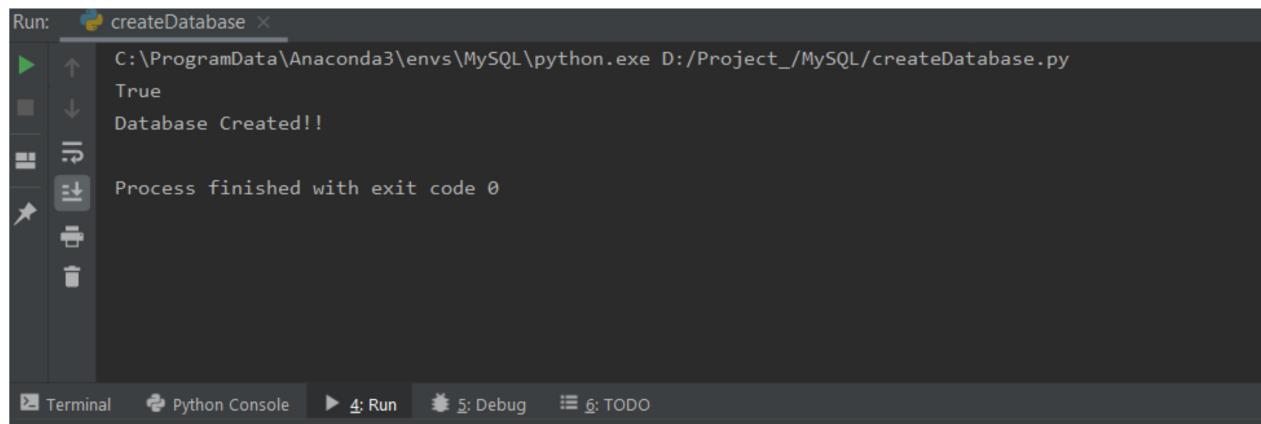


The screenshot shows the PyCharm IDE's Python Console tab. At the top, it says "Connected to pydev debugger (build 193.5662.61)". Below that, the output shows "True". The status bar at the bottom indicates "pydev debugger: process 684 is connecting".

- Our connection is established now.
- Let's start with creating a database with name Student.



```
createDatabase.py ×
1 import mysql.connector as connection
2
3 try:
4     mydb = connection.connect(host="localhost", user="root", passwd="mysql", use_pure=True)
5     # check if the connection is established
6     print(mydb.is_connected())
7
8     query = "Create database Student;"
9     cursor = mydb.cursor() #create a cursor to execute queries
10    cursor.execute(query)
11    print("Database Created!!")
12    mydb.close()
13 except Exception as e:
14     mydb.close()
15     print(str(e))
```



The screenshot shows the PyCharm IDE's Run tab. It displays the command "C:\ProgramData\Anaconda3\envs\MySQL\python.exe D:/Project/_MySQL/createDatabase.py" and its output: "True", "Database Created!!", and "Process finished with exit code 0".

- Let's check if database is created in our MySQL Workbench.

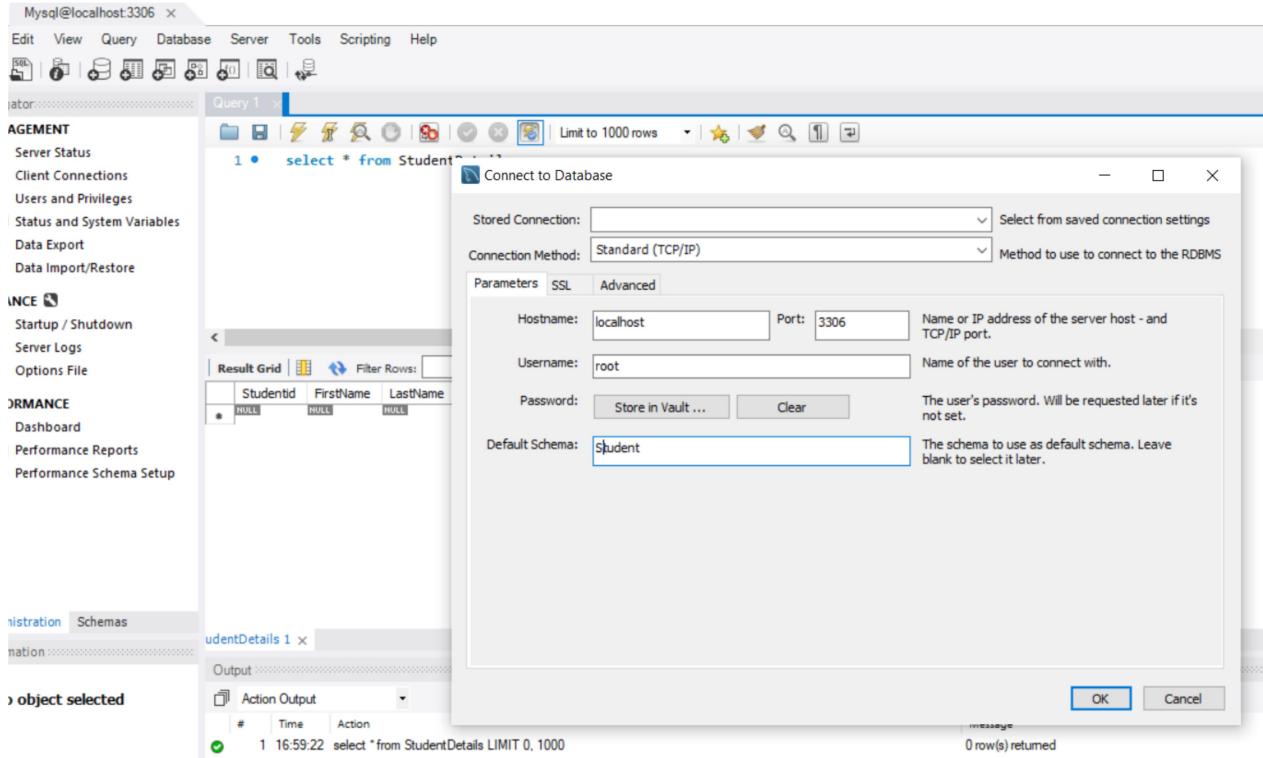
The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query "show databases" has been run, and the results are displayed in a table titled "Result Grid". The table lists several databases: information_schema, mysql, performance_schema, sakila, student (which is highlighted with a yellow background), sys, and world. The "student" database is the one that was created in the previous step.

Database
information_schema
mysql
performance_schema
sakila
student
sys
world

- Great! Database is created.
- Let's start with creating tables. Let's first connect to the created database in our workbench so that we can view the tables, once we create them from python.

The screenshot shows the MySQL Workbench interface with the "Database" menu open. The "Connect to Database..." option is highlighted with a yellow box. This is the next step in connecting to the newly created "student" database.

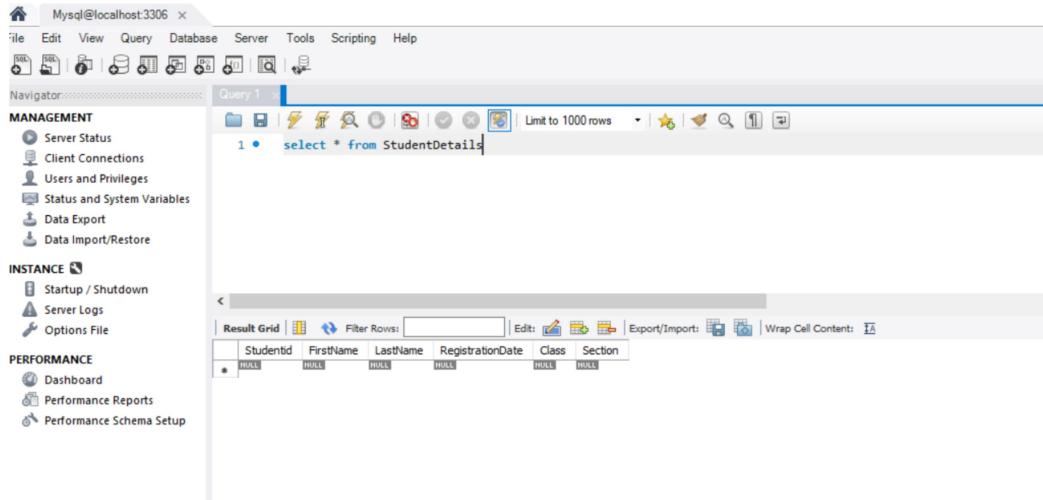
- Give the database name:



- Now we are connected to the created database. Let's start with creating tables.

```
createDatabase.py x createTableInDB.py x
1 import mysql.connector as connection
2
3 try:
4     mydb = connection.connect(host="localhost", database="Student", user="root", passwd="mysql", use_pure=True)
5     # check if the connection is established
6     print(mydb.is_connected())
7
8     query = "CREATE TABLE StudentDetails (Studentid INT(10) AUTO_INCREMENT PRIMARY KEY, FirstName VARCHAR(60), LastName VARCHAR(60), RegistrationDate DATE, Class Varchar(20), Section Varchar(10))"
9
10    cursor = mydb.cursor() #create a cursor to execute queries
11    cursor.execute(query)
12    print("Table Created!!")
13    mydb.close()
14
15 except Exception as e:
16     mydb.close()
17     print(str(e))
18
19 try
```

- We need to pass an additional parameter, database name, while connecting to server. We have passed “Student” database in which we are going to create the table.
 - Let’s see if the table is connected in our Mysql.



➤ Our table is now created in the mentioned database.

A screenshot of the PyCharm IDE. The project structure on the left shows a 'MySQL' folder containing several Python files: connectToMySQL.py, createDatabase.py, createTableInDB.py, insertToTable.py, and a few others. The 'insertToTable.py' file is open in the editor, displaying the following Python code:

```
1 import mysql.connector as connection
2
3 try:
4     mydb = connection.connect(host="localhost", database='Student', user="root", passwd="mysql", use_pure=True)
5     # check if the connection is established
6     print(mydb.is_connected())
7
8     query = "INSERT INTO StudentDetails VALUES (1120, 'Rohit', 'Sharma', '1999-03-21', 'Tenth', 'B')"
9
10    cursor = mydb.cursor() #create a cursor to execute queries
11    cursor.execute(query)
12    print("Values inserted into the table!!")
13    mydb.commit()
14    mydb.close()
15 except Exception as e:
16     mydb.close()
17     print(str(e))
```

The 'Console' tab at the bottom shows the output of running the script: 'Connected to pydev debugger (build 193.5662.61)', 'True', 'Values inserted into the table!!', and 'Process finished with exit code 0'.

- Let's check if the values are inserted:

StudentId	FirstName	LastName	RegistrationDate	Class	Section
1120	Rohit	Sharma	1999-03-21	Tenth	B

- Let's now insert values into a new table in a new database from a file.
- We are loading all the values in the file "glass. Data" into our table.
- We created a new database named "Glass Data" and a new table "Glass Data" in it.
- We are reading each row from the file and inserting into the table.

```

try:
    mydb = connection.connect(host="localhost", user="root", passwd="mysql", use_pure=True)
    #check if the connection is established
    print(mydb.is_connected())
    #create a new database
    query = "Create database GlassData;"
    cursor = mydb.cursor() #create a cursor to execute queries
    cursor.execute(query)
    print("Database Created!!")
    mydb.close() #close the connection

    #Establish a new connection to the database created above
    mydb = connection.connect(host="localhost", database='GlassData', user="root", passwd="mysql", use_pure=True)

    #create a new table to store glass data
    query = "CREATE TABLE IF NOT EXISTS GlassData (Index Number INT(10), RI float(10,5), Na float(10,5), Mg float(10,5), Al float(10,5), " \
            " Si float(10,5), K float(10,5), Ca float(10,5), Ba float(10,5), Fe float(10,5), Class INT(5))"
    cursor = mydb.cursor() # create a cursor to execute queries
    cursor.execute(query)
    print("Table Created!!")

    #read from the file
    with open('glass.data', "r") as f:
        next(f)
        glass_data = csv.reader(f, delimiter="\n")
        for line in enumerate(glass_data):
            for list_ in (line[1]):
                cursor.execute('INSERT INTO GlassData values ({values})'.format(values=(list_)))
    print("Values inserted!!")
    mydb.commit()
    cursor.close()
    mydb.close()

```

```
Run: insertFromFile x
C:\ProgramData\Anaconda3\envs\MySQL\python.exe D:/Project-/MySQL/insertFromFile.py
True
Database Created!!
Table Created!!
Values inserted!!
Process finished with exit code 0
```

➤ Let's check if the values are inserted in the table.

```
Query 1 x
Limit to 1000 rows
1   select * from GlassData.GlassData
```

Index_Number	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Class
1	1.52101	13.64000	4.49000	1.10000	71.78000	0.06000	8.75000	0.00000	0.00000	1
2	1.51761	13.89000	3.60000	1.36000	72.73000	0.48000	7.83000	0.00000	0.00000	1
3	1.51618	13.53000	3.55000	1.54000	72.99000	0.39000	7.78000	0.00000	0.00000	1
4	1.51766	13.21000	3.69000	1.29000	72.61000	0.57000	8.22000	0.00000	0.00000	1
5	1.51742	13.27000	3.62000	1.24000	73.08000	0.55000	8.07000	0.00000	0.00000	1
6	1.51596	12.79000	3.61000	1.62000	72.97000	0.64000	8.07000	0.00000	0.26000	1
7	1.51743	13.30000	3.60000	1.14000	73.09000	0.58000	8.17000	0.00000	0.00000	1
8	1.51756	13.15000	3.61000	1.05000	73.24000	0.57000	8.24000	0.00000	0.00000	1
9	1.51918	14.04000	3.58000	1.37000	72.08000	0.56000	8.30000	0.00000	0.00000	1
10	1.51755	13.00000	3.60000	1.36000	72.99000	0.57000	8.40000	0.00000	0.11000	1
11	1.51571	12.72000	3.46000	1.56000	73.20000	0.67000	8.09000	0.00000	0.24000	1
12	1.51750	12.00000	3.66000	1.37000	72.01000	0.58000	8.55000	0.00000	0.00000	1

➤ The values are inserted.

Let's see some other commands:

- 1) Selecting from table

Oluwaseun Tope Oyero

The screenshot shows the PyCharm IDE interface. The left sidebar displays a project structure for a MySQL application, including files like `checkListOfDatabases.py`, `connectToMySQL.py`, `createDatabase.py`, `createTableInDB.py`, `Download MySQL application.docx`, `glass.data`, `InsertFromFile.py`, `insertToTable.py`, and `selectFromDbIntoDataframe.py`. The right pane shows the Python code for `selectFromDbIntoDataframe.py`:

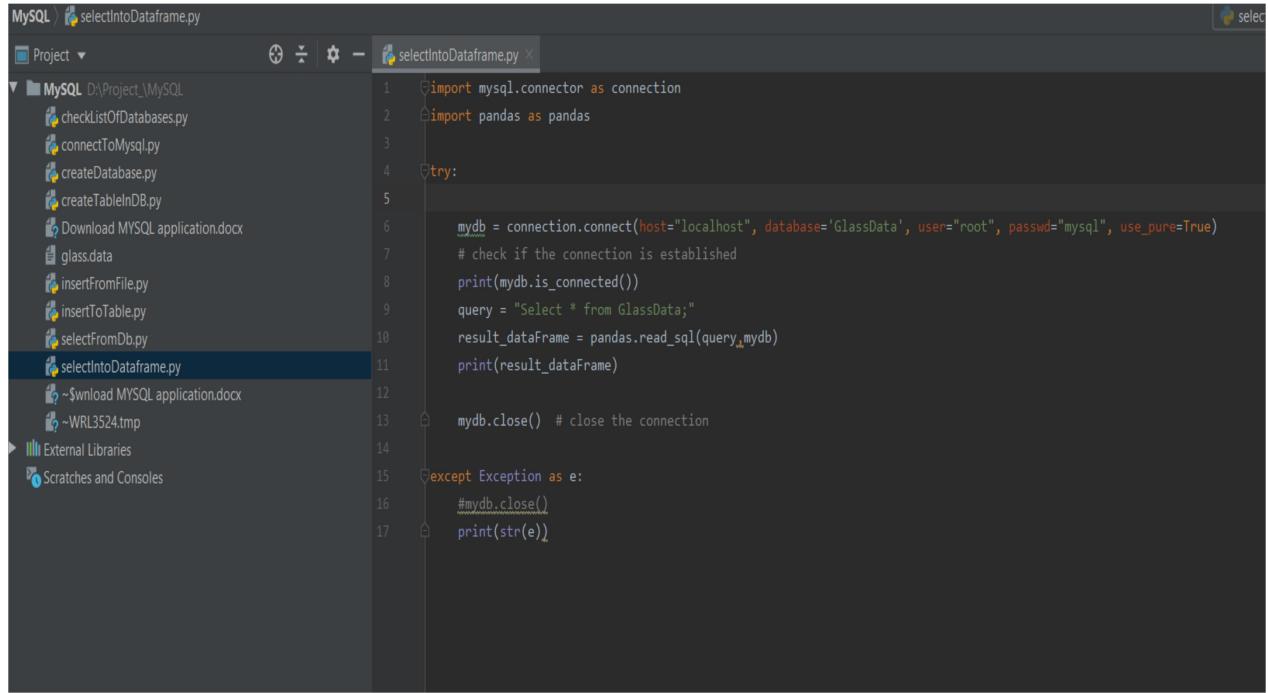
```
1 import mysql.connector as connection
2 import pandas as pandas
3
4 try:
5     mydb = connection.connect(host="localhost", database='GlassData', user="root", passwd="mysql", use_pure=True)
6     #check if the connection is established
7     print(mydb.is_connected())
8     query = "Select * from GlassData;"
9     cursor = mydb.cursor() #create a cursor to execute queries
10    cursor.execute(query)
11    for result in cursor.fetchall():
12        print(result)
13    mydb.close() #close the connection
14
15 except Exception as e:
16     #mydb.close()
17     print(str(e))
```

Result:

The screenshot shows the Jupyter Notebook interface with the output of the executed Python code. The command run was `C:\ProgramData\Anaconda3\envs\MySQL\python.exe D:/Project_/MySQL/selectFromDbIntoDataframe.py`. The output displays 27 rows of data, each representing a row from the MySQL `GlassData` table. The columns correspond to the variables in the code: host, database, user, passwd, use_pure, mydb, is_connected, query, cursor, result, mydb.close, e, and str(e). The data includes numerical values for each column.

host	database	user	passwd	use_pure	mydb	is_connected	query	cursor	result	mydb.close	e	str(e)
"localhost"	"GlassData"	"root"	"mysql"	True	None	True	"Select * from GlassData;"	None	(1, 1.52101, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1)	None	None	
									(2, 1.51761, 13.89, 3.6, 1.36, 72.73, 0.48, 7.83, 0.0, 0.0, 1)			
									(3, 1.51618, 13.53, 3.55, 1.54, 72.99, 0.39, 7.78, 0.0, 0.0, 1)			
									(4, 1.51766, 13.21, 3.69, 1.29, 72.61, 0.57, 8.22, 0.0, 0.0, 1)			
									(5, 1.51742, 13.27, 3.62, 1.24, 73.08, 0.55, 8.07, 0.0, 0.0, 1)			
									(6, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1)			
									(7, 1.51743, 13.3, 3.6, 1.14, 73.09, 0.58, 8.17, 0.0, 0.0, 1)			
									(8, 1.51756, 13.15, 3.61, 1.05, 73.24, 0.57, 8.24, 0.0, 0.0, 1)			
									(9, 1.51918, 14.04, 3.58, 1.37, 72.08, 0.56, 8.3, 0.0, 0.0, 1)			
									(10, 1.51755, 13.0, 3.6, 1.36, 72.99, 0.57, 8.4, 0.0, 0.11, 1)			
									(11, 1.51571, 12.72, 3.46, 1.56, 73.2, 0.67, 8.09, 0.0, 0.24, 1)			
									(12, 1.51763, 12.8, 3.66, 1.27, 73.01, 0.6, 8.56, 0.0, 0.0, 1)			
									(13, 1.51589, 12.88, 3.43, 1.4, 73.28, 0.69, 8.05, 0.0, 0.24, 1)			
									(14, 1.51748, 12.86, 3.56, 1.27, 73.21, 0.54, 8.38, 0.0, 0.17, 1)			
									(15, 1.51763, 12.61, 3.59, 1.31, 73.29, 0.58, 8.5, 0.0, 0.0, 1)			
									(16, 1.51761, 12.81, 3.54, 1.23, 73.24, 0.58, 8.39, 0.0, 0.0, 1)			
									(17, 1.51784, 12.68, 3.67, 1.16, 73.11, 0.61, 8.7, 0.0, 0.0, 1)			
									(18, 1.52196, 14.36, 3.85, 0.89, 71.36, 0.15, 9.15, 0.0, 0.0, 1)			
									(19, 1.51911, 13.9, 3.73, 1.18, 72.12, 0.06, 8.89, 0.0, 0.0, 1)			
									(20, 1.51735, 13.02, 3.54, 1.69, 72.73, 0.54, 8.44, 0.0, 0.07, 1)			
									(21, 1.5175, 12.82, 3.55, 1.49, 72.75, 0.54, 8.52, 0.0, 0.19, 1)			
									(22, 1.51966, 14.77, 3.75, 0.29, 72.02, 0.03, 9.0, 0.0, 0.0, 1)			
									(23, 1.51736, 12.78, 3.62, 1.29, 72.79, 0.59, 8.7, 0.0, 0.0, 1)			
									(24, 1.51751, 12.81, 3.57, 1.35, 73.02, 0.62, 8.59, 0.0, 0.0, 1)			
									(25, 1.5172, 13.38, 3.5, 1.15, 72.85, 0.5, 8.43, 0.0, 0.0, 1)			
									(26, 1.51764, 12.98, 3.54, 1.21, 73.0, 0.65, 8.53, 0.0, 0.0, 1)			
									(27, 1.51793, 13.21, 3.48, 1.41, 72.64, 0.59, 8.43, 0.0, 0.0, 1)			

- Let's try to store all the select values into a Dataframe.



```
MySQL | selectIntoDataframe.py
Project ▾          selectIntoDataframe.py
MySQL D:\Project\MySQL
  checkListOfDatabases.py
  connectToMysql.py
  createDatabase.py
  createTableInDB.py
  Download MYSQL application.docx
  glass.data
  insertFromFile.py
  insertToTable.py
  selectFromDb.py
  selectIntoDataframe.py
  ~$wnload MYSQL application.docx
  ~WRL3524tmp
External Libraries
Scratches and Consoles
```

```
1 import mysql.connector as connection
2 import pandas as pandas
3
4 try:
5
6     mydb = connection.connect(host="localhost", database="GlassData", user="root", passwd="mysql", use_pure=True)
7     # check if the connection is established
8     print(mydb.is_connected())
9     query = "Select * from GlassData;"
10    result_dataFrame = pandas.read_sql(query,mydb)
11    print(result_dataFrame)
12
13    mydb.close() # close the connection
14
15 except Exception as e:
16     #mydb.close()
17     print(str(e))
```

We can use `pandas.read_sql` to store the values in a **dataframe**.

Let's see the result:

```
True
   Index_Number      RI      Na      Mg      Al      ...      K      Ca      Ba      Fe      Class
0           1  1.52101  13.64  4.49  1.10      ...  0.06  8.75  0.00  0.0       1
1           2  1.51761  13.89  3.60  1.36      ...  0.48  7.83  0.00  0.0       1
2           3  1.51618  13.53  3.55  1.54      ...  0.39  7.78  0.00  0.0       1
3           4  1.51766  13.21  3.69  1.29      ...  0.57  8.22  0.00  0.0       1
4           5  1.51742  13.27  3.62  1.24      ...  0.55  8.07  0.00  0.0       1
...
209        210  1.51623  14.14  0.00  2.88      ...  0.08  9.18  1.06  0.0       7
210        211  1.51685  14.92  0.00  1.99      ...  0.00  8.40  1.59  0.0       7
211        212  1.52065  14.36  0.00  2.02      ...  0.00  8.44  1.64  0.0       7
212        213  1.51651  14.38  0.00  1.94      ...  0.00  8.48  1.57  0.0       7
213        214  1.51711  14.23  0.00  2.08      ...  0.00  8.62  1.67  0.0       7

[214 rows x 11 columns]
```

```
Process finished with exit code 0
```

2) Update

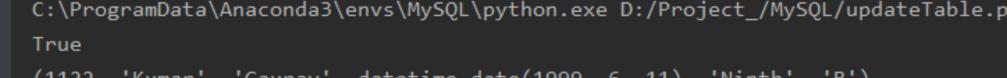
The screenshot shows the PyCharm IDE interface with a MySQL project open. The left sidebar displays the project structure with files like checklistOfDatabases.py, connectToMySQL.py, createDatabase.py, createTableInDB.py, Download MYSQL application.docx, glass.data, insertFromFile.py, insertToTable.py, MySQL application.docx, selectFromDb.py, selectIntoDataFrame.py, updateTable.py, ~\$SQL application.docx, and ~WRL524tmp. Below the sidebar are tabs for External Libraries and Scratches and Consoles.

```
import mysql.connector as connection
import pandas as pandas

try:
    mydb = connection.connect(host="localhost", database='Student', user="root", passwd="mysql", use_pure=True)
    # check if the connection is established
    print(mydb.is_connected())
    query = "UPDATE studentdetails SET FirstName = 'Kumar', LastName = 'Gaurav' WHERE Studentid = 1122"
    cursor = mydb.cursor() # create a cursor to execute queries
    cursor.execute(query)
    mydb.commit()

    #let's check if the value is updated in the table.
    query = "Select * from studentdetails where Studentid=1122;"
    cursor = mydb.cursor() # create a cursor to execute queries
    cursor.execute(query)
    for result in cursor.fetchall():
        print(result)

    mydb.close() # close the connection
except Exception as e:
    #mydb.close()
    print(str(e))
try > for result in cursor.fetchall()
```



The screenshot shows the PyCharm IDE interface. The top bar displays the file name "updateTable". The left sidebar has sections for "Favorite" (with a star icon) and "Structure" (with a database icon). The main area shows the command line output of the Python script:

```
C:\ProgramData\Anaconda3\envs\MySQL\python.exe D:/Project/_MySQL/updateTable.py
True
(1122, 'Kumar', 'Gaurav', datetime.date(1999, 6, 11), 'Ninth', 'B')
Process finished with exit code 0
```

At the bottom, there are tabs for "Terminal", "Python Console", "Run", "Debug", and "TODO".

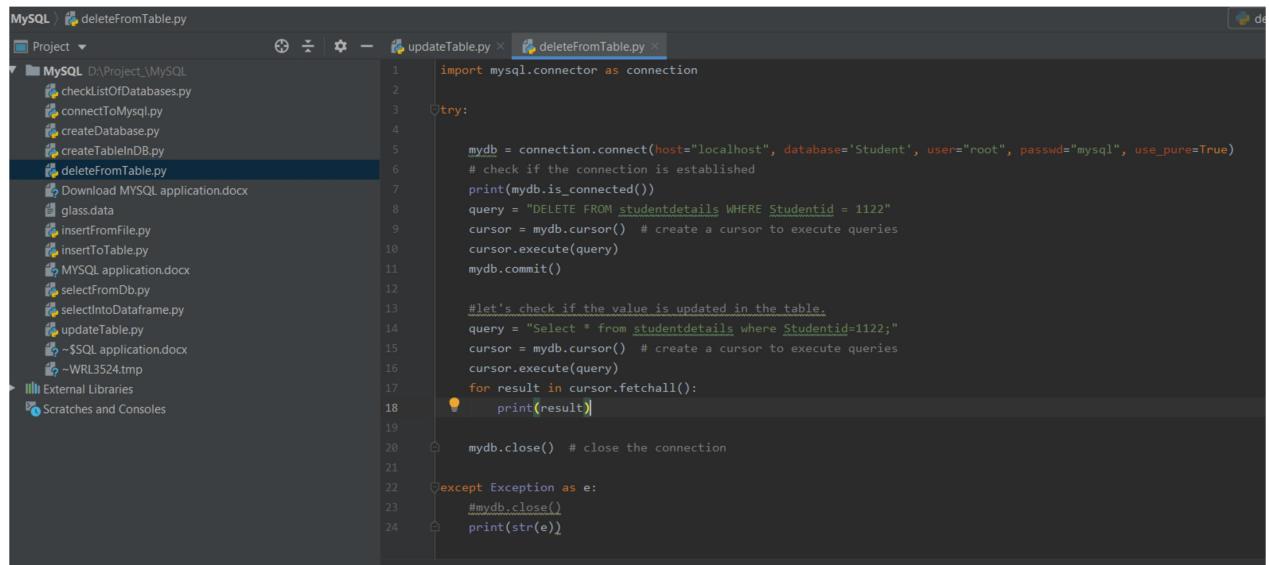
➤ Let's check in MySql workbench:

```
Query 1 | Limit to 1000 rows |
```

1 • select * from studentdetails

	StudentId	FirstName	LastName	RegistrationDate	Class	Section
▶	1120	Rohit	Sharma	1999-03-21	Tenth	B
	1121	Kapil	Sharma	1999-04-18	Ninth	A
	1122	Kumar	Gaurav	1999-06-11	Ninth	B
	1132	Sachin	Kumar	1997-11-11	Eleventh	A
*	NULL	HULL	HULL	HULL	HULL	HULL

3) Delete statement



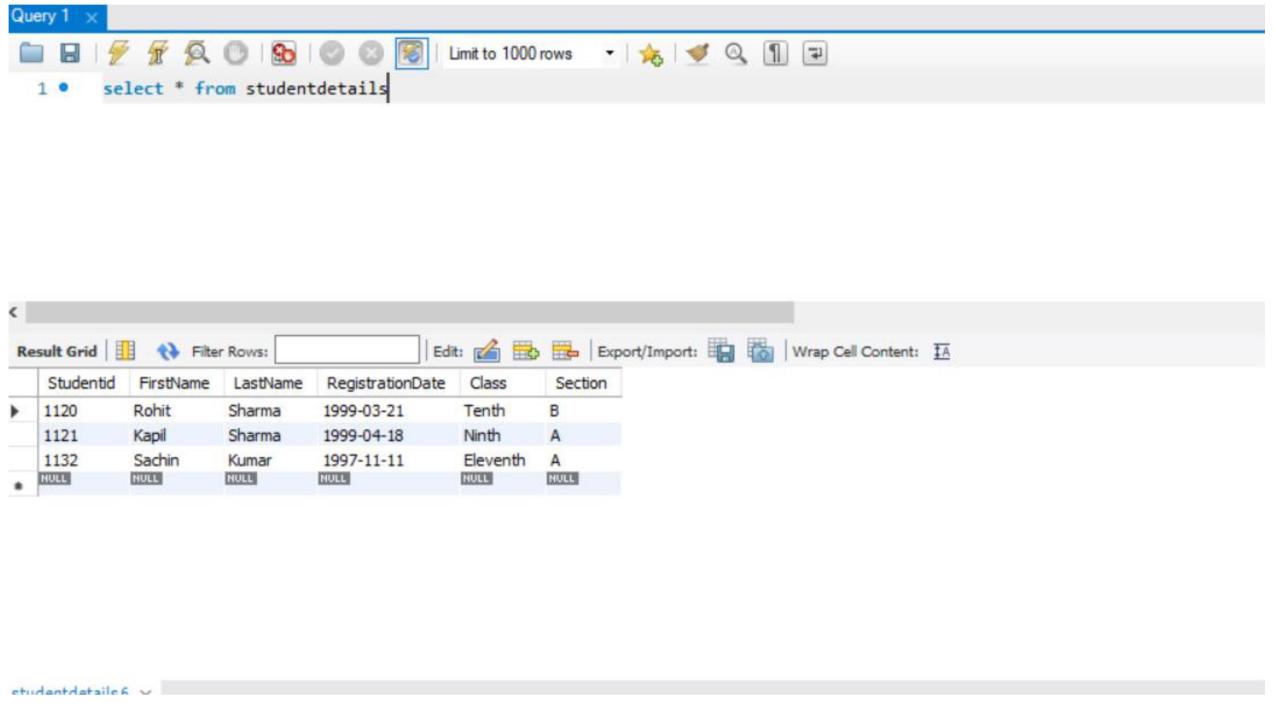
```
import mysql.connector as connection

try:
    mydb = connection.connect(host="localhost", database='Student', user="root", passwd="mysql", use_pure=True)
    # check if the connection is established
    print(mydb.is_connected())
    query = "DELETE FROM studentdetails WHERE Studentid = 1122"
    cursor = mydb.cursor() # create a cursor to execute queries
    cursor.execute(query)
    mydb.commit()

    #let's check if the value is updated in the table.
    query = "Select * from studentdetails where Studentid=1122;"
    cursor = mydb.cursor() # create a cursor to execute queries
    cursor.execute(query)
    for result in cursor.fetchall():
        print(result)

    mydb.close() # close the connection
except Exception as e:
    #mydb.close()
    print(str(e))
```

➤ Let's check in MySql workbench:



Studentid	FirstName	LastName	RegistrationDate	Class	Section
1120	Rohit	Sharma	1999-03-21	Tenth	B
1121	Kapil	Sharma	1999-04-18	Ninth	A
1132	Sachin	Kumar	1997-11-11	Eleventh	A
NULL	NULL	NULL	NULL	NULL	NULL

4) Group by, Order by

The screenshot shows the PyCharm IDE interface with a project named 'MySQL' open. The 'groupByOrderBy.py' file is the active editor. The code uses MySQL connector and pandas to connect to a 'GlassData' database, execute a query to group by 'Si' and order by 'Class', and print the results. It also includes exception handling for closing the connection.

```
import mysql.connector as connection
import pandas as pandas

try:
    mydb = connection.connect(host="localhost", database="GlassData", user="root", passwd="mysql", use_pure=True)
    # check if the connection is established
    print(mydb.is_connected())
    query = "select si, Count(Si), Class from GlassData.GlassData group by Class order by Si;" #group by class
    result_dataFrame = pandas.read_sql(query,mydb)
    print(result_dataFrame)

    mydb.close() # close the connection
except Exception as e:
    #mydb.close()
    print(str(e))

try:
```

The run output window shows the execution results:

```
C:\ProgramData\Anaconda3\envs\MySQL\python.exe D:/Project/_MySQL/groupByOrderBy.py
True
   si  Count(si)  Class
0  69.89        13      5
1  71.78        70      1
2  71.87        76      2
3  72.37         9      6
4  72.77        17      3
5  72.81        29      7

Process finished with exit code 0
```