Lesson 10: MySQL knowledge required for Data Science.

Data science often involves working with databases to store, retrieve, and analyze data. MySQL is a popular relational database management system that is commonly used in conjunction with data science tasks.

Here are some ways data science and MySQL databases are interconnected:

- **Data Storage:** MySQL as a Data Warehouse: MySQL databases are frequently used to store large volumes of structured data. In data science, you might store datasets in MySQL databases for easy retrieval and management.
- **Data Retrieval:** SQL Queries for Data Extraction: Data scientists often use SQL queries to extract relevant data from a MySQL database. This could involve filtering data, joining tables, or aggregating information.
- **Data Cleaning and Transformation:** Data Cleaning with SQL: SQL queries can be used to clean and preprocess data directly in the database. This might include handling missing values, removing duplicates, or transforming data types.
- **Data Integration:** Combining Data from Multiple Sources: Data scientists frequently work with data from various sources. MySQL databases can be used to integrate and consolidate data from different places, making it easier to analyze.
- **Collaboration:** Database as a Shared Resource: MySQL databases serve as a centralized repository that multiple data scientists can access. This facilitates collaboration and ensures that everyone is working with the same set of data.
- **Data Analysis:** Performing Statistical Analysis: Once data is retrieved from a MySQL database, data scientists can use statistical methods and machine learning algorithms to derive insights and patterns.
- **Model Deployment:** Storing Trained Models: After creating and training machine learning models, you might store the results or models in a MySQL database for future use or integration into applications.
- **Data Security:** Ensuring Data Security: MySQL provides mechanisms for securing data, which is crucial when dealing with sensitive information. Data scientists need to be aware of and implement security best practices.
- **Scalability:** Handling Large Datasets: MySQL is designed to handle large datasets efficiently. As data science tasks often involve working with big data, MySQL can be scaled to meet these demands.
- **Real-time Data Analysis:** Real-time Analytics: MySQL can be used in scenarios where real-time data analysis is required. Data scientists might perform continuous queries to analyze streaming data.

MySQL databases play a vital role in the data science workflow, from storing and managing data to facilitating analysis and collaboration among data science teams. Understanding how to interact with and leverage databases is a valuable skill for any data scientist.

Loading a database from XAMPP into Google Colab.

Since Google Colab is an online environment, it doesn't have direct access to your local XAMPP server. However, you can use cloud-based databases or other online storage solutions.

Important Notes:

- Ensure your MySQL server is accessible from the internet or use a cloud-based database if needed.
- Handle database credentials securely and avoid sharing them in public notebooks.
- The provided code assumes that you already have a table named 'students' with appropriate columns in your database.

How to set a Set Up a Cloud-Based MySQL Database:

- ✓ Consider free MySQL database hosting services that are not part of major cloud providers. Remember to carefully review the terms and conditions, limitations, and features of each service to ensure they meet your project requirements.

- ✓ Additionally, be aware that free hosting services may have limitations in terms of storage, bandwidth, and support compared to paid services. If your project grows, you might need to consider upgrading to a paid plan or explore other hosting options.

- ✓ Here are the options:

  1) ***db4free.net:***

     - db4free.net offers free MySQL database hosting. It's suitable for development and testing purposes.

  2) ***FreeMySQLHosting:***

     - FreeMySQLHosting provides free MySQL databases with a limited storage capacity. It's suitable for small projects and testing.

- ✓ Please note that these services might have limitations in terms of storage, bandwidth, or uptime. Always review the terms of service of any free hosting provider to ensure it meets your project's requirements.

Keep in mind that the availability of free services can change, and new services may become available. It's a good practice to verify the current offerings on the respective websites or contact the service providers directly for the most up-to-date information.

db4free.net, MYSQL and Google Colab for Data Science:

1) db4free.net.

Overview:

- db4free.net is a free MySQL database hosting service.
- It provides users with an environment to test and develop MySQL databases without the need for a local server.

Key Features:

- ***Free Hosting:*** db4free.net offers free hosting for MySQL databases.
- ***Web-based Interface:*** Users can manage their databases through a web-based interface.
- ***Testing and Development:*** Ideal for testing and developing database-driven applications.

Considerations:

- ***Limited Resources:*** As a free service, db4free.net has limitations on resources, making it suitable for small projects and testing.
- ***Uptime:*** While it provides a reliable service, users should be aware of occasional downtime.

2) dbMySQL:

Overview:

- MySQL is an open-source relational database management system (RDBMS).
- Widely used for web applications, data warehousing, and e-commerce.

Key Features:

- ***Scalability:*** MySQL is known for its scalability, making it suitable for applications of various sizes.
- ***Community Support:*** It has a large and active community, offering support and resources.
- ***Open Source:*** MySQL is open-source, allowing users to modify and distribute the software.

Considerations:

- ***Configuration:*** Users need to configure and manage MySQL installations.
- ***Security:*** Security measures need to be implemented to protect databases.

3) Google Colab for Data Science:

Overview:

Google Colab is a cloud-based platform provided by Google that allows users to write and execute Python code in a collaborative environment.

Key Features:

- ***Free Access to GPUs:*** Colab provides free access to GPUs, enabling users to perform machine learning tasks.
- ***Integration with Google Drive:*** Seamless integration with Google Drive for storage and sharing of notebooks.
- ***Data Visualization:*** Supports data visualization libraries like Matplotlib and Seaborn.

Considerations:

- ***Session Limits:*** Colab sessions have time and resource limits, and users may need to reconnect after a certain duration.
- ***Internet Access:*** Requires internet access to run and save notebooks.

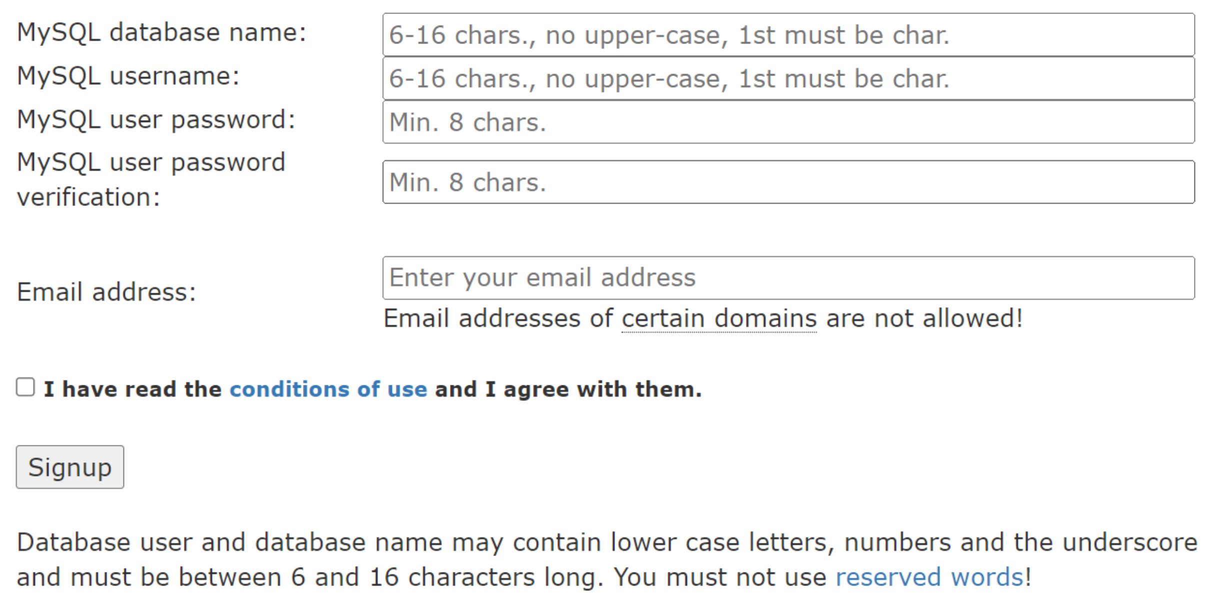
Integration for Data Science Workflow:

- *Connect db4free.net Database to Google Colab:* Use MySQL connectors in Python to establish a connection between Colab and the db4free.net MySQL database.
- *Data Retrieval:* Fetch data from the MySQL database using SQL queries.
- *Data Analysis and Visualization:* Leverage Python libraries in Colab, such as Pandas for data analysis and Matplotlib/Seaborn for visualization.
- *Machine Learning:* Utilize Colab's GPU support for machine learning tasks, incorporating data from the MySQL database.
- *Collaboration:* Share Colab notebooks via Google Drive, facilitating collaborative work on data science projects.

Note: Always handle sensitive information, such as database credentials, securely in any data science workflow.

IMPLEMENTATION IN GOOGLE COLAB:

Step 1: ***Create an account with dbfree platform*(https://www.db4free.net/signup.php).**



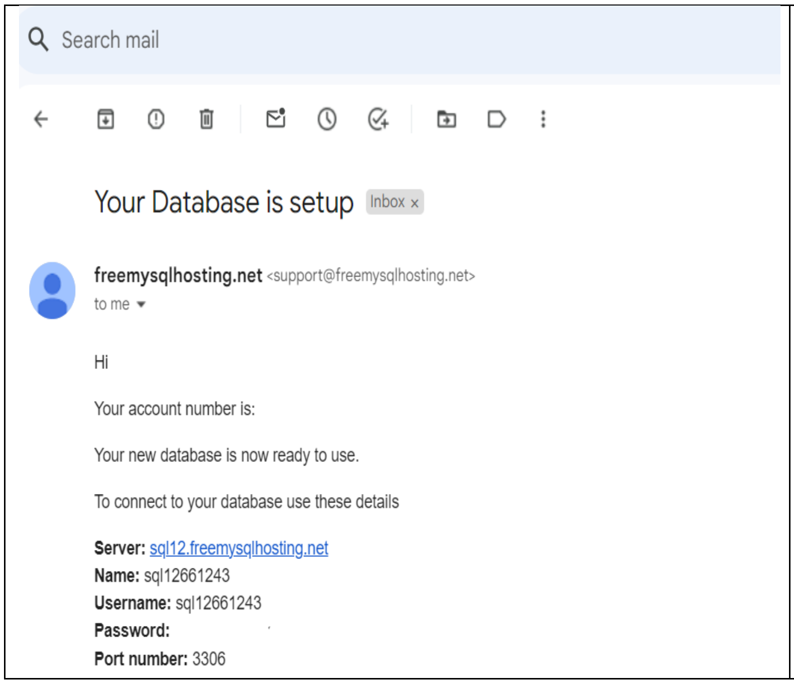
MySQL database name:	6-16 chars., no upper-case, 1st must be char.
MySQL username:	6-16 chars., no upper-case, 1st must be char.
MySQL user password:	Min. 8 chars.
MySQL user password verification:	Min. 8 chars.
Email address:	Enter your email address Email addresses of certain domains are not allowed!

☐ **I have read the conditions of use and I agree with them.**

Signup

Database user and database name may contain lower case letters, numbers and the underscore and must be between 6 and 16 characters long. You must not use reserved words!

Step 2: ***Log in to your account after activation (Link is sent to your email address). Click PHPMyAdmin.***



Search mail

Your Database is setup Inbox x

freemysqlhosting.net <support@freemysqlhosting.net>
to me

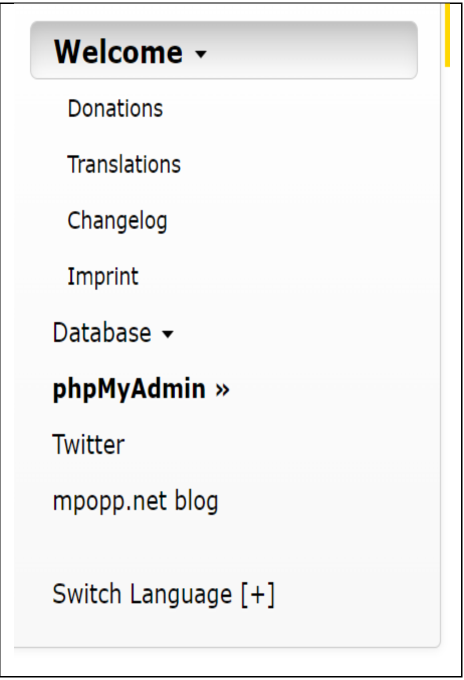
Hi

Your account number is:

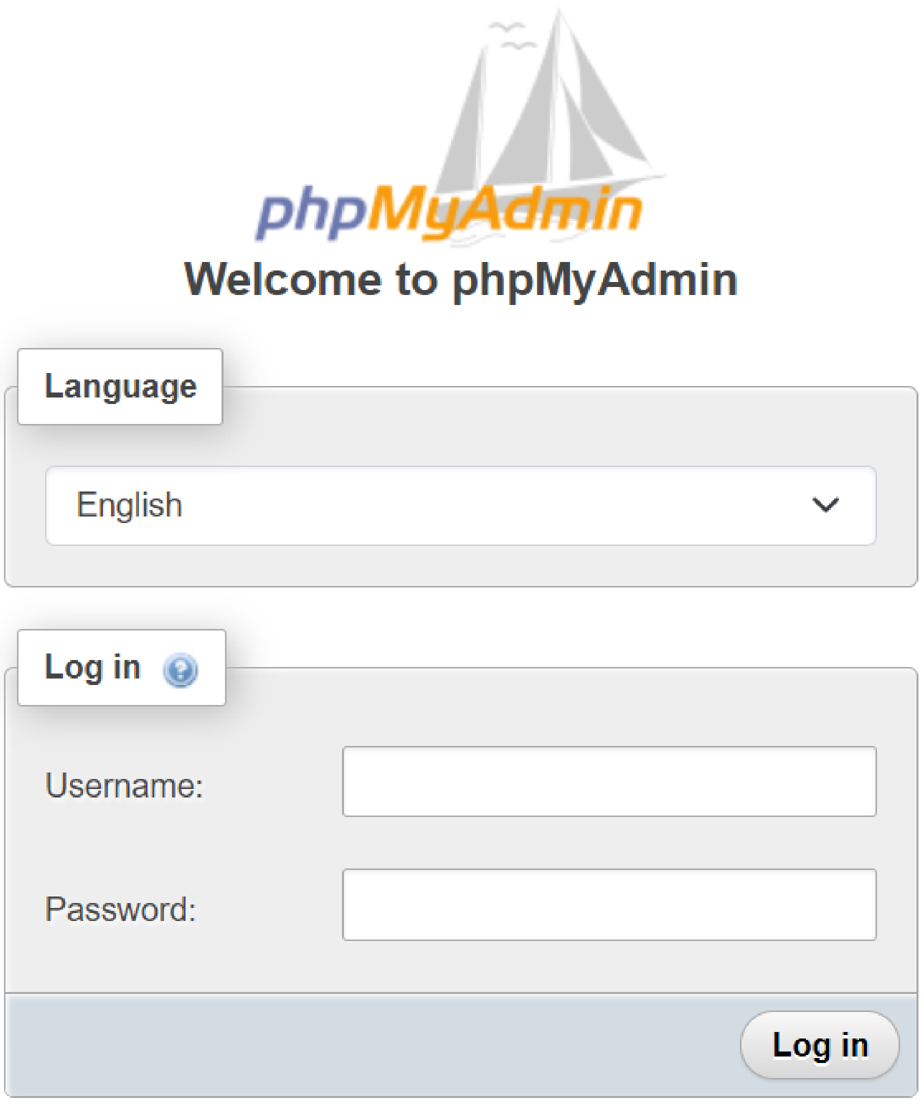
Your new database is now ready to use.

To connect to your database use these details

Server: sql12.freemysqlhosting.net
Name: sql12661243
Username: sql12661243
Password:
Port number: 3306

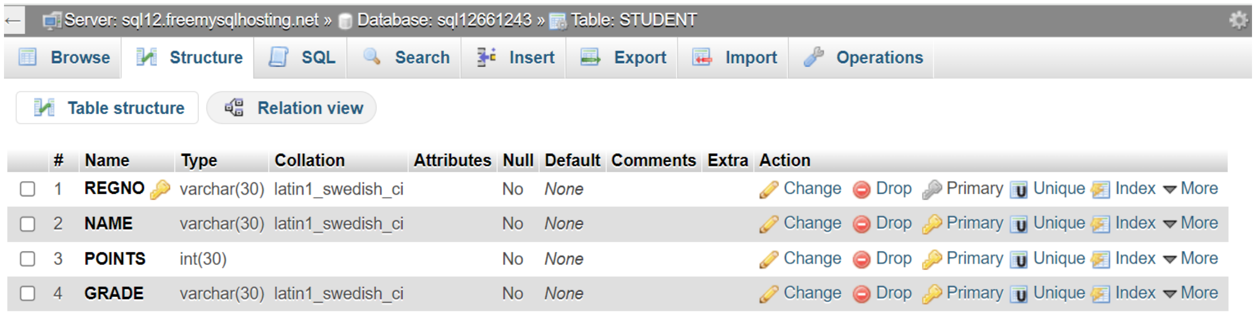


Input your username and password.

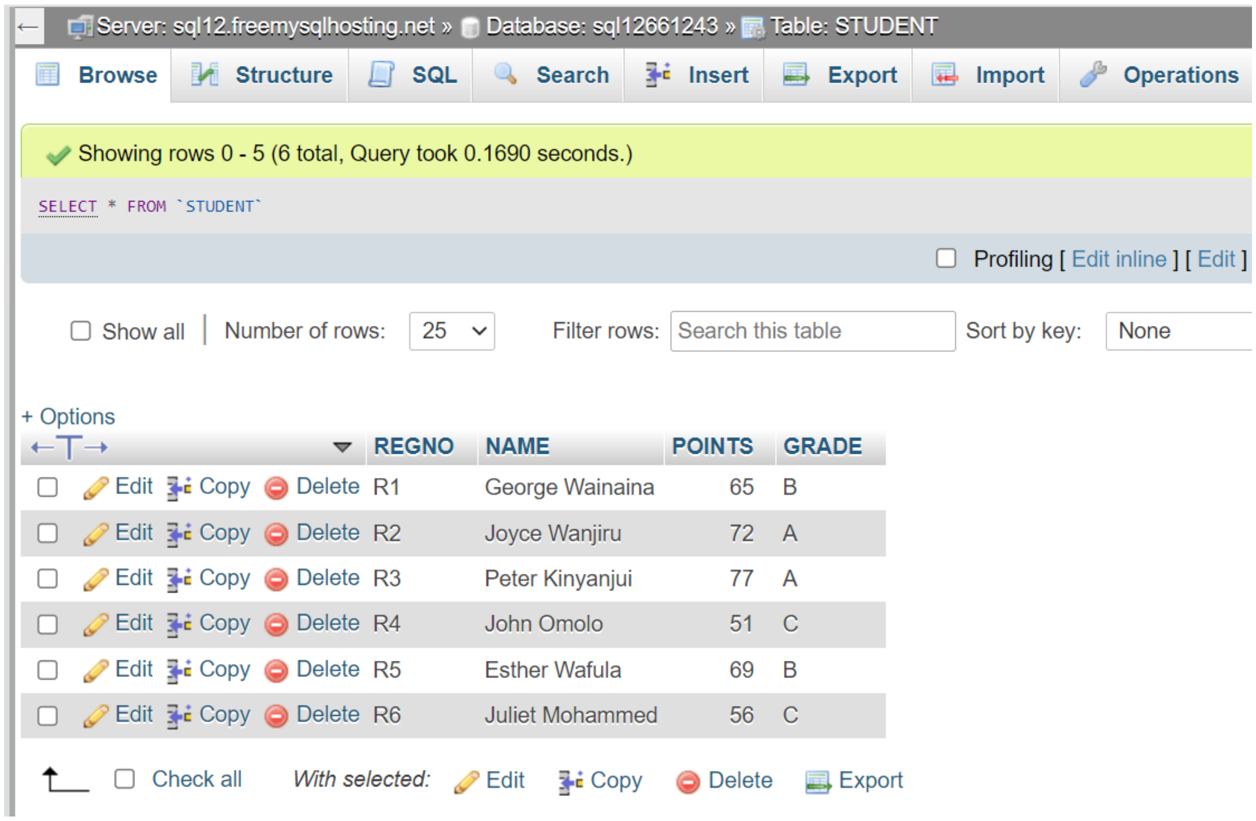


Step 3: Create table structure and populate as per the screen shots below.

Table: STUDENT.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	REGNO	varchar(30)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
2	NAME	varchar(30)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More
3	POINTS	int(30)			No	None			Change Drop Primary Unique Index More
4	GRADE	varchar(30)	latin1_swedish_ci		No	None			Change Drop Primary Unique Index More


Server: sql12.freemysqlhosting.net » Database: sql12661243 » Table: STUDENT
Browse
Structure
SQL
Search
Insert
Export
Import
Operations

Showing rows 0 - 5 (6 total, Query took 0.1690 seconds.)

```
SELECT * FROM `STUDENT`
```

Profiling [Edit inline] [Edit]
Show all | Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

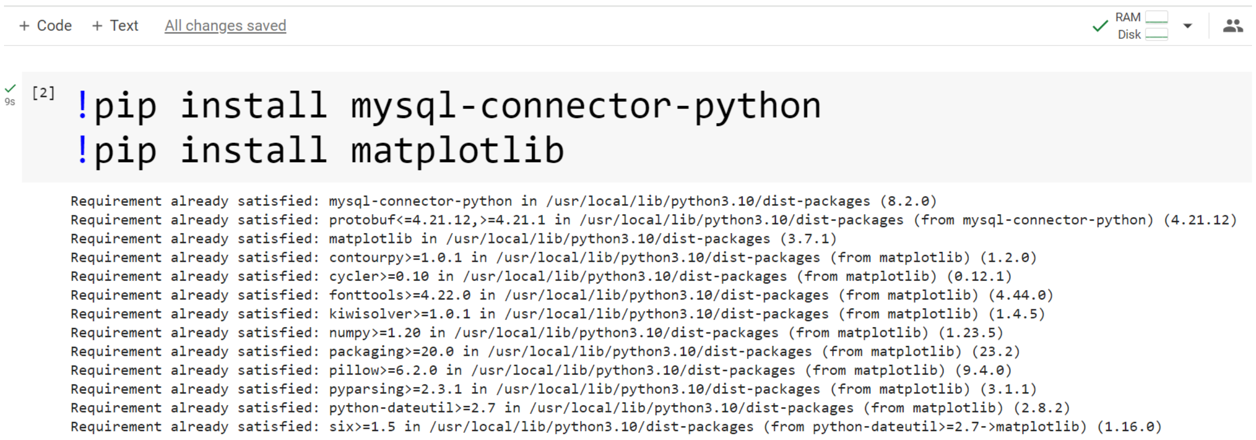
REGNO	NAME	POINTS	GRADE
R1	George Wainaina	65	B
R2	Joyce Wanjiru	72	A
R3	Peter Kinyanjui	77	A
R4	John Omolo	51	C
R5	Esther Wafula	69	B
R6	Juliet Mohammed	56	C

Check all
With selected:
Edit
Copy
Delete
Export

Step 4: Access google colab and install the necessary libraries.

!pip install mysql-connector-python

!pip install matplotlib



```
[2] !pip install mysql-connector-python
    !pip install matplotlib
```

```
Requirement already satisfied: mysql-connector-python in /usr/local/lib/python3.10/dist-packages (8.2.0)
Requirement already satisfied: protobuf<=4.21.12,>=4.21.1 in /usr/local/lib/python3.10/dist-packages (from mysql-connector-python) (4.21.12)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.44.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

Command	*Purpose*
!pip install mysql-connector-python	Installs the MySQL Connector/Python package, which is a Python driver for MySQL databases. This package allows Python programs to connect to and interact with MySQL databases.
!pip install matplotlib	Installs the Matplotlib library, a popular 2D plotting library for Python. Matplotlib is widely used for creating various types of plots and charts, including line plots, scatter plots, bar plots, and more.

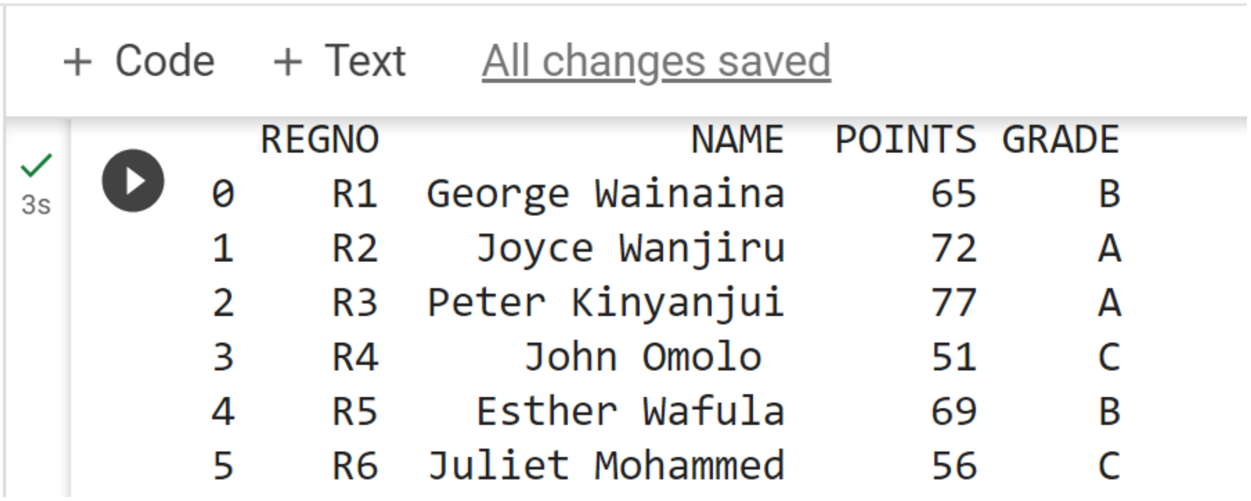
Step 5: Create a bar graph from the data in table STUDENT.

SOURCE CODE:

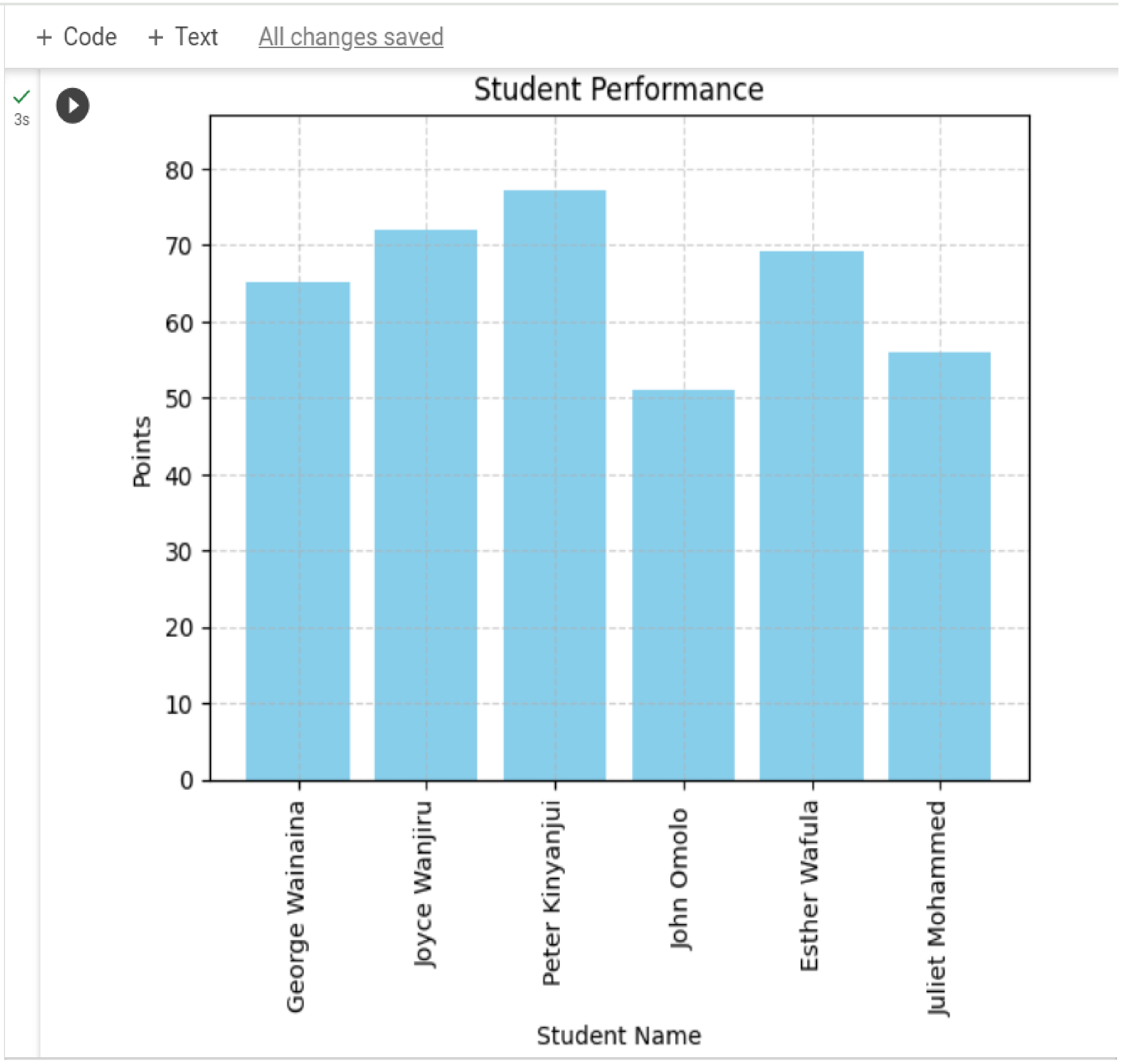
```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
# Replace with your MySQLHosting credentials
host = "sql12.freemysqlhosting.net"
database = "sql12661243"
user = "sql12661243"
password = "wainaina"
port = 3306
# Connect to the MySQL database
conn = mysql.connector.connect(
    host=host,
    user=user,
    password=password,
    database=database,
    port=port
)
# Create a cursor object to execute SQL queries
cursor = conn.cursor()
# Example query to retrieve all rows from the student table
query = "SELECT * FROM STUDENT"
df = pd.read_sql(query, conn)
# Display the retrieved data
print(df)
# Example: Bar chart of grades with y-axis starting from 0
plt.bar(df['NAME'], df['POINTS'], color='skyblue')  # Change the color as needed
plt.title('Student Performance')
plt.xlabel('Student Name')
plt.ylabel('Points')
# Rotate x-axis labels vertically
plt.xticks(rotation='vertical')
# Set the y-axis limit to start from 0
plt.ylim(0, max(df['POINTS']) + 10)  # You can adjust the additional value as needed
# Show grid
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
# Close the cursor and connection
cursor.close()
conn.close()
```

Line	*Code*	*Explanation*
1	import mysql.connector	Imports the MySQL Connector module, allowing Python to connect to and interact with MySQL databases.
2	import pandas as pd	Imports the pandas library with the alias pd. Pandas is a powerful data manipulation and analysis library in Python.
3	import matplotlib.pyplot as plt	Imports the pyplot module from the Matplotlib library with the alias plt. Matplotlib is a plotting library for creating visualizations in Python.
4	# Replace with your MySQLHosting credentials	Comments providing a reminder to replace the placeholder values with actual MySQL database credentials.
5	host = "sql12.freemysqlhosting.net"	Defines the MySQL database host.
6	database = "sql12661243"	Defines the name of the MySQL database.
7	user = "sql12661243"	Defines the MySQL database user.
8	password = "wainaina"	Defines the password for the MySQL database user.
9	port = 3306	Defines the port number for the MySQL database connection.
10	conn = mysql.connector.connect(...)	Establishes a connection to the MySQL database using the provided credentials. The connection object is stored in the variable conn.
11	cursor = conn.cursor()	Creates a cursor object that allows executing SQL queries on the MySQL database. The cursor object is stored in the variable cursor.
12	query = "SELECT * FROM STUDENT"	Defines an SQL query to select all rows from the "STUDENT" table in the MySQL database.
13	df = pd.read_sql(query, conn)	Executes the SQL query and reads the result into a pandas DataFrame (df). The DataFrame contains the retrieved data from the database.
14	print(df)	Prints the contents of the DataFrame (df), displaying the retrieved data from the "STUDENT" table.
15	plt.bar(df['NAME'], df['POINTS'], color='skyblue')	Creates a bar chart using Matplotlib. The chart represents student points with student names on the x-axis and points on the y-axis. The bars are colored sky blue.
16	plt.title('Student Performance')	Sets the title of the chart to "Student Performance".
17	plt.xlabel('Student Name')	Sets the label for the x-axis to "Student Name".
18	plt.ylabel('Points')	Sets the label for the y-axis to "Points".
19	plt.xticks(rotation='vertical')	Rotates the x-axis labels vertically for better readability.
20	plt.ylim(0, max(df['POINTS']) + 10)	Sets the y-axis limit to start from 0, adjusting the additional value based on the maximum points in the DataFrame.
21	plt.grid(True, linestyle='--', alpha=0.6)	Adds a grid to the chart with a dashed line style and alpha (transparency) value of 0.6.
22	plt.show()	Displays the created chart.
23	cursor.close()	Closes the cursor, releasing associated resources.
24	conn.close()	Closes the database connection, releasing associated resources.

OUTPUT:



```
  REGNO              NAME  POINTS  GRADE
0    R1   George Wainaina      65      B
1    R2     Joyce Wanjiru      72      A
2    R3   Peter Kinyanjui      77      A
3    R4        John Omolo      51      C
4    R5     Esther Wafula      69      B
5    R6   Juliet Mohammed      56      C
```



Step 6: Create a pie chart from the data in table STUDENT.

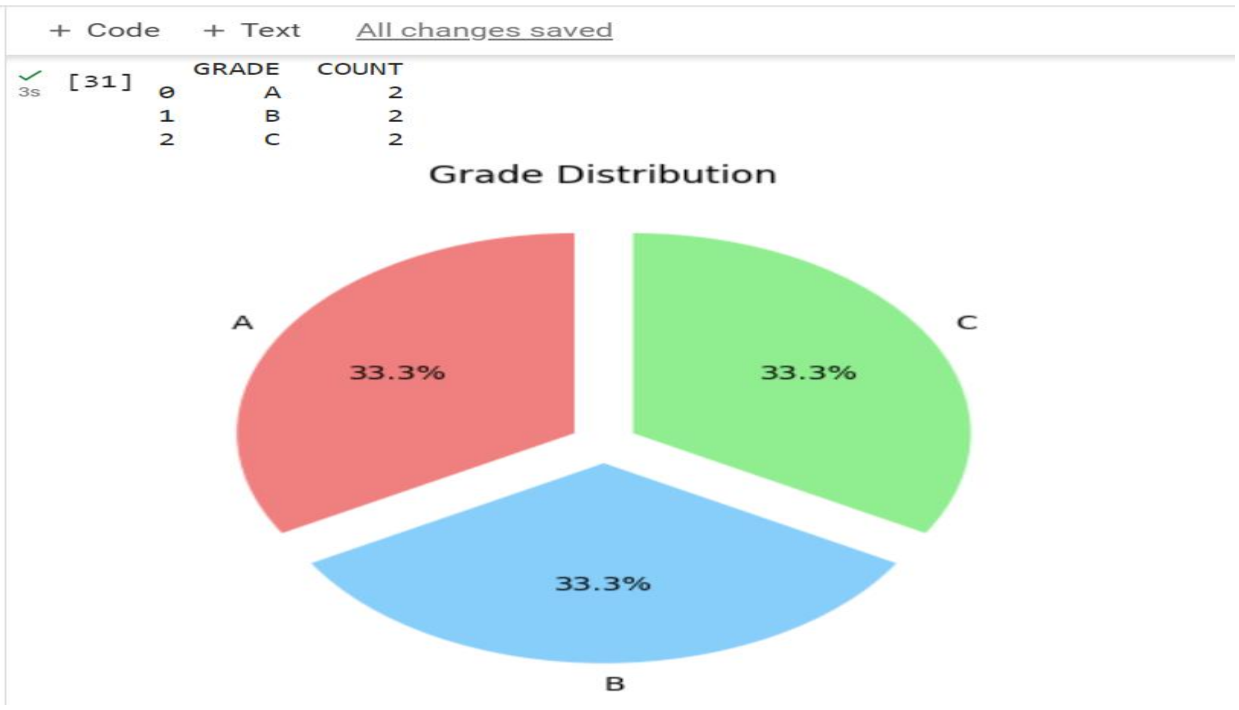
SOURCE CODE:

```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
# Replace with your MySQLHosting credentials
host = "sql12.freemysqlhosting.net"
database = "sql12661243"
user = "sql12661243"
password = "wainaina"
port = 3306
# Connect to the MySQL database
conn = mysql.connector.connect(
   host=host,
   user=user,
   password=password,
   database=database,
   port=port
)
# Create a cursor object to execute SQL queries
cursor = conn.cursor()
# Example query to retrieve grade distribution from the STUDENT table
query = "SELECT GRADE, COUNT(*) AS COUNT FROM STUDENT GROUP BY
GRADE"
grade_distribution = pd.read_sql(query, conn)
# Display the retrieved data
print(grade_distribution)
# Example: Pie chart of grade distribution with explode effect
explode = tuple([0.1] * len(grade_distribution))  # Dynamic explode values based on the
number of unique grades
colors = ['lightcoral', 'lightskyblue', 'lightgreen']  # Set custom colors for each sector
plt.pie(grade_distribution['COUNT'], labels=grade_distribution['GRADE'],
autopct='%1.1f%%', startangle=90, colors=colors, explode=explode)
plt.title('Grade Distribution')
plt.show()
# Close the cursor and connection
cursor.close()
conn.close()
```

Line	Code Statement	Explanation
1	import mysql.connector	Import the MySQL connector for Python.
2	host = "sql12.freemysqlhosting.net"	Set the MySQL database host.
3	database = "sql12661243"	Set the name of the MySQL database.
4	user = "sql12661243"	Set the username for MySQL authentication.
5	password = "wainaina"	Set the password for MySQL authentication.
6	port = 3306	Set the port for MySQL connection.
7	conn = mysql.connector.connect(...)	Establish a connection to the MySQL database using the provided host, user, password, database, and port.
8	cursor = conn.cursor()	Create a cursor object to execute SQL queries on the connected database.

9	query = "SELECT GRADE, COUNT(*) AS COUNT FROM STUDENT GROUP BY GRADE"	SQL query to retrieve grade distribution from the STUDENT table.
10	grade_distribution = pd.read_sql(query, conn)	Execute the SQL query and store the result in a Pandas DataFrame named grade_distribution.
11	print(grade_distribution)	Print the retrieved grade distribution data.
12	explode = tuple([0.1] * len(grade_distribution))	Create a tuple of explode values for the pie chart, based on the number of unique grades.
13	colors = ['lightcoral', 'lightskyblue', 'lightgreen']	Define custom colors for each sector of the pie chart.
14	plt.pie(...)	Create a pie chart using Matplotlib, specifying labels, autopct, startangle, colors, and explode values.
15	plt.title('Grade Distribution')	Set the title for the pie chart.
16	plt.show()	Display the pie chart.
17	cursor.close()	Close the cursor to release database resources.
18	conn.close()	Close the database connection.

OUTPUT:



```
  GRADE  COUNT
0     A      2
1     B      2
2     C      2
```

Step 7: Create a line graph from the data in table STUDENT.

SOURCE CODE:

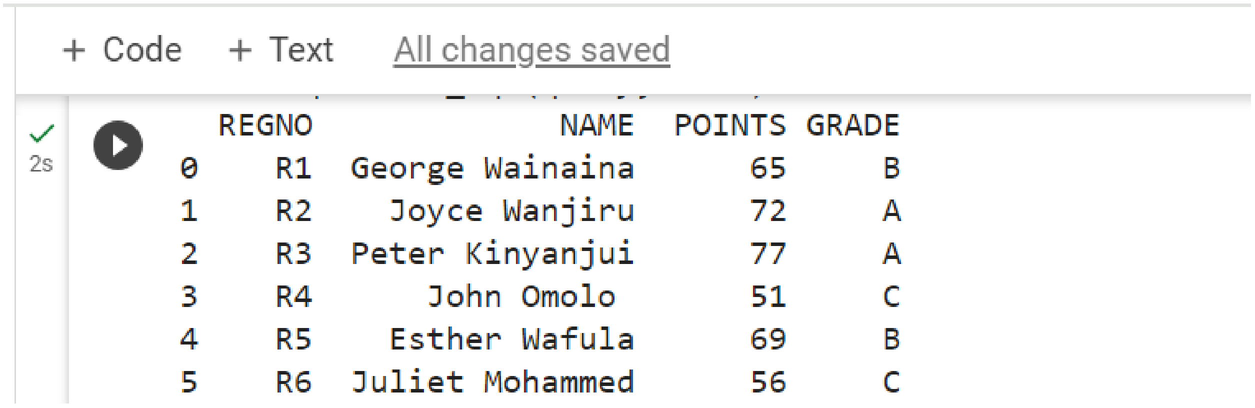
```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
# Replace with your MySQLHosting credentials
host = "sql12.freemysqlhosting.net"
database = "sql12661243"
```

```
user = "sql12661243"
password = "wainaina"
port = 3306
# Connect to the MySQL database
conn = mysql.connector.connect(
    host=host,
    user=user,
    password=password,
    database=database,
    port=port
)
# Create a cursor object to execute SQL queries
cursor = conn.cursor()
# Example query to retrieve all rows from the student table
query = "SELECT * FROM STUDENT"
df = pd.read_sql(query, conn)
# Display the retrieved data
print(df)
# Example: Line graph of points over student names with a specific color
plt.plot(df['NAME'], df['POINTS'], marker='o', color='orange', label='Points')  # Change the color as needed
plt.title('Student Performance')
plt.xlabel('Student Name')
plt.ylabel('Points')
# Rotate x-axis labels vertically
plt.xticks(rotation='vertical')
# Set the y-axis limit to start from 0
#plt.ylim(0)
# Show grid
plt.grid(True, linestyle='--', alpha=0.6)
# Show legend
plt.legend
```

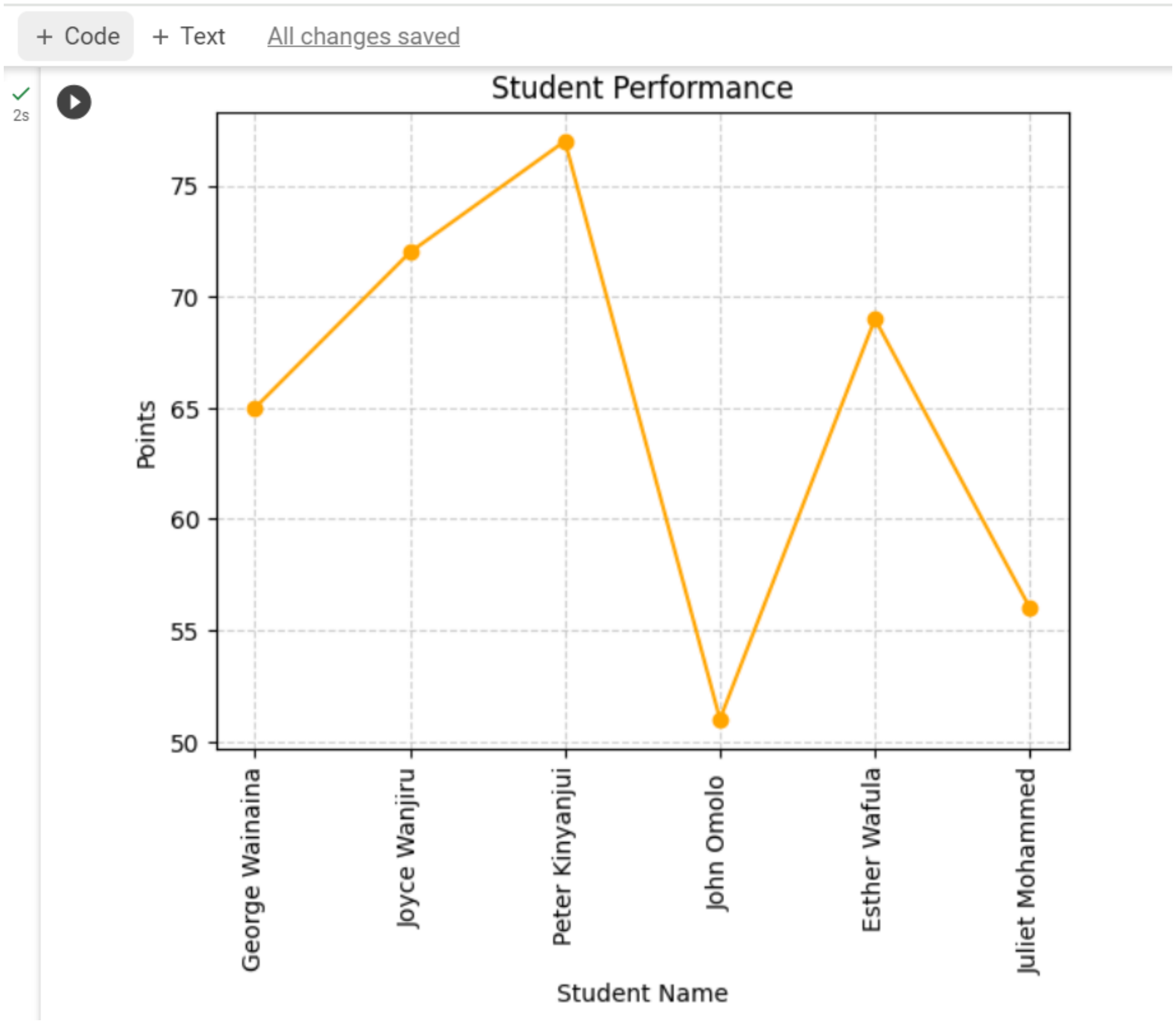
Line	***Code Statement***	***Explanation***
1	import mysql.connector to port = 3306	Import necessary libraries and define MySQL database credentials.
2	conn = mysql.connector.connect(...)	Establish a connection to the MySQL database using the provided credentials.

3	cursor = conn.cursor()	Create a cursor object to execute SQL queries.
4	query = "SELECT * FROM STUDENT"	Define an SQL query to retrieve all rows from the STUDENT table.
5	df = pd.read_sql(query, conn)	Execute the query and store the result in a Pandas DataFrame (df).
6	print(df)	Print the retrieved data (all rows from the STUDENT table).
7	plt.plot(df['NAME'], df['POINTS'], marker='o', color='orange', label='Points')	Plot a line graph with student names on the x-axis and points on the y-axis.
8	plt.title('Student Performance')	Set the title of the line graph.
9	plt.xlabel('Student Name') and plt.ylabel('Points')	Set labels for the x-axis and y-axis, respectively.
10	plt.xticks(rotation='vertical')	Rotate x-axis labels vertically for better readability.
11	plt.grid(True, linestyle='--', alpha=0.6)	Show a grid on the graph with a dashed linestyle and 60% transparency.
12	plt.legend	Show the legend in the plot.

OUTPUT:



```
  REGNO             NAME  POINTS GRADE
0    R1  George Wainaina      65     B
1    R2    Joyce Wanjiru      72     A
2    R3  Peter Kinyanjui      77     A
3    R4       John Omolo      51     C
4    R5    Esther Wafula      69     B
5    R6  Juliet Mohammed      56     C
```


+ Code
+ Text
All changes saved
2s
Student Performance
Points
75
70
65
60
55
50
George Wainaina
Joyce Wanjiru
Peter Kinyanjui
John Omolo
Esther Wafula
Juliet Mohammed
Student Name