

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.

The screenshot shows an email interface. The main area displays an email from 'freemysqlhosting.net' with the subject 'Your Database is setup'. The email content includes a greeting, account details, and database connection information. The sidebar on the right contains a 'Welcome' dropdown menu with links to Donations, Translations, Changelog, Imprint, Database, phpMyAdmin, Twitter, mpop.net blog, and a Switch Language button.

Search mail

← [Icons] ⋮

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

Welcome ▾

- Donations
- Translations
- Changelog
- Imprint
- Database ▾
- phpMyAdmin »**
- Twitter
- mpopp.net blog
- Switch Language [+]

Input your username and password.



Language

English ▼

Log in ?

Username:

Password:

Log in







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

Table: STUDENT.

The screenshot shows the phpMyAdmin interface. On the left is a sidebar with a tree view of databases and tables. The main area displays the table structure for the 'STUDENT' table in the 'georgew' database. The table has four columns: REGNO, NAME, POINTS, and GRADE. Each column has a checkbox, a primary key icon for REGNO, and a 'More' link. The table structure is shown in a table format with columns for #, Name, Type, Collation, Attributes, Null, Default, Comments, Extra, and Action.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 REGNO	varchar(30)	utf8mb4_0900_ai_ci		No	None			Change Drop More
<input type="checkbox"/>	2 NAME	varchar(30)	utf8mb4_0900_ai_ci		No	None			Change Drop More
<input type="checkbox"/>	3 POINTS	int			No	None			Change Drop More
<input type="checkbox"/>	4 GRADE	varchar(10)	utf8mb4_0900_ai_ci		No	None			Change Drop More

phpMyAdmin



Recent

Favorites

db4free_sys

georgew

New

STUDENT

information_schema

performance_schema

Server: MySQL 8.2 Server:3306 » Database: georgew » Table: STUDENT

Browse



















Structure

SQL

Search

Insert

Export

				REGNO	NAME	POINTS	GRADE
<input type="checkbox"/>				R1	George Wainaina	65	B
<input type="checkbox"/>				R2	Joyce Wanjiru	72	A
<input type="checkbox"/>				R3	Peter Kinyanjui	77	A
<input type="checkbox"/>				R4	John Omolo	51	C
<input type="checkbox"/>				R5	Esther Wafula	69	B
<input type="checkbox"/>				R6	Juliet Mohammed	56	C

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

!pip install mysql-connector-python

!pip install matplotlib

```

+ Code + Text All changes saved
[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 = "db4free.net"
database = "georgew"
user = "georgew"
password = "george123"
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"
wainaina = pd.read_sql(query, conn)
# Display the retrieved data
print(wainaina)
# Example: Bar chart of grades with y-axis starting from 0
plt.bar(wainaina['NAME'], wainaina['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')
plt.xticks(rotation=45)
# Set the y-axis limit to start from 0
plt.ylim(0, max(wainaina['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()

```

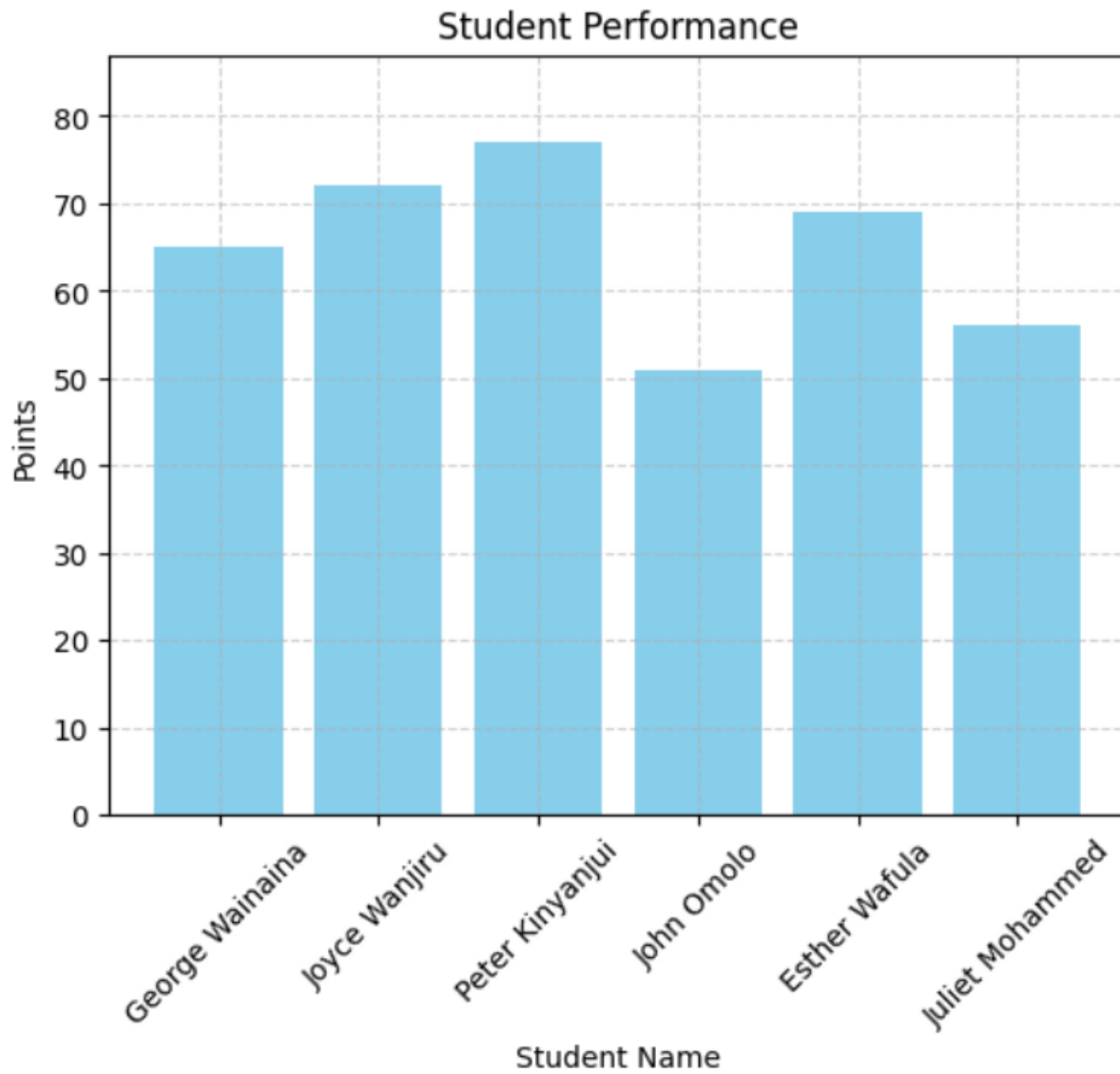
Line	Code Statement	Explanation
1	import mysql.connector	Import the MySQL Connector module for Python.
2	import pandas as pd	Import the Pandas library and alias it as 'pd'.
3	import matplotlib.pyplot as plt	Import the Matplotlib library and alias it as 'plt'.
4	host = "db4free.net"	Define variables for MySQL database connection details: host, database, user, password, and port.
5	conn = mysql.connector.connect(...)	Establish a connection to the MySQL database using the provided credentials.
6	cursor = conn.cursor()	Create a cursor object to execute SQL queries.
7	query = "SELECT * FROM STUDENT"	Define an SQL query to select all rows from the 'STUDENT' table.
8	wainaina = pd.read_sql(query, conn)	Use Pandas to execute the SQL query and store the result in a DataFrame named 'wainaina'.

9	<code>print(wainaina)</code>	Display the retrieved data (all rows from the 'STUDENT' table) stored in the 'wainaina' DataFrame.
10	<code>plt.bar(wainaina['NAME'], wainaina['POINTS'], color='skyblue')</code>	Create a bar chart using Matplotlib with student names on the x-axis and points on the y-axis.
11	<code>plt.title('Student Performance')</code>	Set the title of the plot to 'Student Performance'.
12	<code>plt.xlabel('Student Name')</code>	Label the x-axis as 'Student Name'.
13	<code>plt.ylabel('Points')</code>	Label the y-axis as 'Points'.
14	<code>plt.xticks(rotation=45)</code>	Rotate x-axis labels by 45 degrees (for better readability).
15	<code>plt.ylim(0, max(wainaina['POINTS']) + 10)</code>	Set the y-axis limit to start from 0 and extend to the maximum points + 10.
16	<code>plt.grid(True, linestyle='--', alpha=0.6)</code>	Show grid with a dashed line style and transparency.
17	<code>plt.show()</code>	Display the plot.
18	<code>cursor.close()</code>	Close the cursor.
18	<code>conn.close()</code>	Close the database connection.

OUTPUT:

+ Code + Text All changes saved

```
wainaina = pd.read_sql(query, conn)
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
```



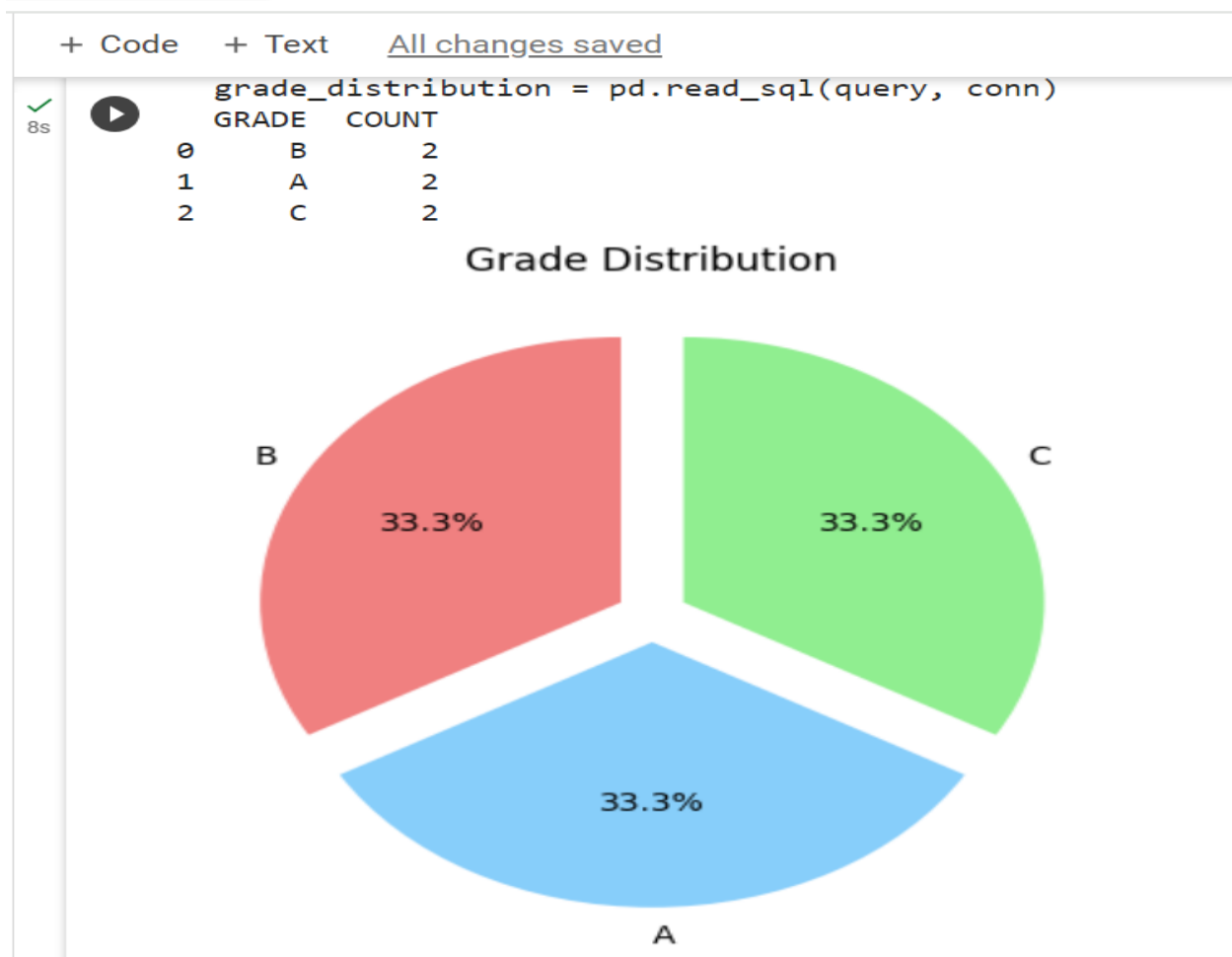
Step 6: Create a pie chart from the data in table STUDENT.

SOURCE CODE:

Line	Code Statement	Explanation
1	import mysql.connector	Import the MySQL Connector module for Python.
2	import pandas as pd	Import the Pandas library and alias it as 'pd'.
3	import matplotlib.pyplot as plt	Import the Matplotlib library and alias it as 'plt'.
4	host = "db4free.net"	Define variables for MySQL database connection details: host, database, user, password, and port.
5	conn = mysql.connector.connect(...)	Establish a connection to the MySQL database using the provided credentials.
6	cursor = conn.cursor()	Create a cursor object to execute SQL queries.

7	query = "SELECT GRADE, COUNT(*) AS COUNT FROM STUDENT GROUP BY GRADE"	Define an SQL query to retrieve grade distribution from the 'STUDENT' table.
8	grade_distribution = pd.read_sql(query, conn)	Use Pandas to execute the SQL query and store the result in a DataFrame named 'grade_distribution'.
9	print(grade_distribution)	Display the retrieved grade distribution data.
10	plt.pie(grade_distribution['COUNT'], labels=grade_distribution['GRADE'], autopct='%1.1f%%', startangle=90, colors=colors, explode=explode)	Create a pie chart using Matplotlib with grade distribution data.
11	plt.title('Grade Distribution')	Set the title of the pie chart.
12	plt.show()	Display the pie chart.
13	cursor.close()	Close the cursor.
14	conn.close()	Close the database connection.

OUTPUT:



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 = "db4free.net"
database = "georgew"
user = "georgew"
password = "george123"
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"
wainaina = pd.read_sql(query, conn)
# Display the retrieved data
print(wainaina)
# Example: Line graph of points over student names with a specific color
plt.plot(wainaina['NAME'], wainaina['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')
plt.xticks(rotation=45)
# 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	Import the MySQL Connector module for Python.
2	import pandas as pd	Import the Pandas library and alias it as 'pd'.
3	import matplotlib.pyplot as plt	Import the Matplotlib library and alias it as 'plt'.
4	host = "db4free.net"	Define variables for MySQL database connection details: host, database, user, password, and port.
5	conn = mysql.connector.connect(...)	Establish a connection to the MySQL database using the provided credentials.
6	cursor = conn.cursor()	Create a cursor object to execute SQL queries.
7	query = "SELECT * FROM STUDENT"	Define an SQL query to select all rows from the 'STUDENT' table.
8	wainaina = pd.read_sql(query, conn)	Use Pandas to execute the SQL query and store the result in a DataFrame named 'wainaina'.
9	print(wainaina)	Display the retrieved data (all rows from the 'STUDENT' table) stored in the 'wainaina' DataFrame.
10	plt.plot(wainaina['NAME'], wainaina['POINTS'], marker='o', color='orange', label='Points')	Create a line graph using Matplotlib with student names on the x-axis and points on the y-axis.
11	plt.title('Student Performance')	Set the title of the line graph.
12	plt.xlabel('Student Name')	Label the x-axis as 'Student Name'.
13	plt.ylabel('Points')	Label the y-axis as 'Points'.
14	plt.xticks(rotation=45)	Rotate x-axis labels by 45 degrees (for better readability).
15	plt.grid(True, linestyle='--', alpha=0.6)	Show grid with a dashed line style and transparency.
16	plt.legend	Show legend to identify the plotted line.
17	(Note: plt.show() is missing in the provided code.)	Add plt.show() to actually display the line graph.
18	cursor.close()	Close the cursor.
19	conn.close()	Close the database connection.

OUTPUT:

+ Code

+ Text

All changes saved

✓

7s

	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

✓
7s

