LAMINA STUDIOS, LLC
DATA ANALYTICS INTERNSHIP

Create and Manipulate SQL Databases using Python

Documentation

Jewel Anne A. Reyes
BS Computer Science
Polytechnic University of the Philippines

September 1, 2023

Table of Contents

Task Overview

For Data Analysts and Data Scientists, Python has many advantages. A huge range of open-source libraries make it an incredibly useful tool for any Data Analyst.

We have pandas, NumPy and Vaex for data analysis, Matplotlib, seaborn and Bokeh for visualisation, and TensorFlow, scikit-learn and PyTorch for machine learning applications.

With its (relatively) easy learning curve and versatility, it's no wonder that Python is one of the fastest-growing programming languages out there.

While there is a massive variety of sources for datasets, in many cases - particularly in enterprise businesses - data is going to be stored in a relational database. Relational databases are an extremely efficient, powerful and widely-used way to create, read, update and delete data of all kinds.

The most widely used relational database management systems (RDBMSs) - Oracle, MySQL, Microsoft SQL Server, PostgreSQL, IBM DB2 - all use the Structured Query Language (SQL) to access and make changes to the data.

In this task, I used Jupyter Notebook and Oracle MySQL to create a database and manipulate its data using Python. This task also required the python libraries MySQL connector and pandas.

Codes

The following are the steps with codes that was used for this task.

Step 1: Installation of Python and Oracle MySQL Server, Workbench, & Connector

Step 2: Installation of Jupyter on a terminal

```
python -m pip install jupyter
```

Step 3: Launching of Jupyter Notebook

```
Jupyter notebook
```

Step 4: Install MySQL Connector Python Library

```
!pip install mysql-connector-python
```

Step 5: Install Panda Python Library

```
!pip install pandas
```

Step 6: Importing the installed libraries

```
#Import libraries
import mysql.connector
from mysql.connector import Error
import pandas as pd
```

Step 7: Connecting to MySQL server

```
import mysql.connector
from mysql.connector import Error

# Connecting to MySQL Server
def create_server_connection(host_name, user_name, user_password):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password
        )
        print("MySQL Database connection successful")
    except Error as err:
        print(f"Error: '{err}'")

    return connection

# MySQL terminal password
pw = "password123"

# Database name
db = "dummy_db"
connection = create_server_connection("localhost", "root", pw)
```

Step 8: Creating new database called “dummy_db”

```
#Create dummy_db
def create_database(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        print("Database created successfully")
    except Error as err:
        print(f"Error: '{err}'")
create_database_query = "Create database dummy_db"
create_database(connection, create_database_query)
```

Step 9: Connecting to the “dummy_db” database

```
#Connect to Database
def create_db_connection(host_name, user_name, user_password, db_name):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password,
            database=db_name
        )
        print("MySQL Database connection successful")
    except Error as err:
        print(f"Error: '{err}'")

    return connection
```

Step 10: Creating a Query Execution Function

```
#Execute SQL Queries
def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        print("Query successful")
    except Error as err:
        print(f"Error: '{err}'")
```

Step 11: Creating a table for “dummy_db” database called “books” table. The data was retrieved from https://www.encodedna.com/2012/12/create-dummy-database-tables.htm

```
#Create books table from dummy database
create_books_table = """
CREATE TABLE Books(
    BookID int primary key NOT NULL,
    BookName varchar(50) NULL,
    Category varchar(50) NULL,
    Price numeric(18, 2) NULL,
    Price_Range varchar(20) NULL
)
 """


# Connect to the Database
connection = create_db_connection("localhost", "root", pw, db)
execute_query(connection, create_books_table)
```

Step 12: Inserting data to “books” table. The data was retrieved from https://www.encodedna.com/2012/12/create-dummy-database-tables.htm

```
#Insert data to table
insert_books = """
INSERT INTO Books
    (BookID, BookName, Category, Price, Price_Range)
VALUES
    ('1', 'Computer Architecture', 'Computers', 125.6, '100-150'),
    ('2', 'Advanced Composite Materials', 'Science', 172.56, '150-200'),
    ('3', 'Asp.Net 4 Blue Book', 'Programming', 56.00, '50-100'),
    ('4', 'Strategies Unplugged', 'Science', 99.99, '50-100'),
    ('5', 'Teaching Science', 'Science', 164.10, '150-200'),
    ('6', 'Challenging Times', 'Business', 150.70, '150-200'),
    ('7', 'Circuit Bending', 'Science', 112.00, '100-150'),
    ('8', 'Popular Science', 'Science', 210.40, '200-250'),
    ('9', 'ADOBE Premiere', 'Computers', 62.20, '50-100')
"""

connection = create_db_connection("localhost", "root", pw, db)
execute_query(connection, insert_books)
```

Step 13: Reading data from the database “dummy_db”

```
#Reading data
def read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
        return result
    except Error as err:
        print(f"Error: '{err}'")
connection = create_db_connection("localhost", "root", pw, db)
execute_query(connection, insert_books)
```

Step 14: Using SELECT statement to display data

```
#Using SELECT statement to display data
q1 = """
SELECT *
FROM books;
"""

connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q1)
for result in results:
    print(result)
```

Step 15: Getting the BookName and Price data from the database

```
#Getting only the BookName and the Price
q2 = """
SELECT BookName, Price
FROM books;
"""

connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q2)
for result in results:
    print(result)
```

Step 16: Getting all the distinct book category from the database

```
#Getting all the distinct book Category
q3 = """
SELECT distinct Category
FROM books;
"""

connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q3)
for result in results:
    print(result)
```

Step 17: Getting all the data which has greater the 150 price

```
#Getting the data in which the Price is greater than 150
q4 = """
SELECT *
FROM books
WHERE Price > 150;
"""

connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q4)
for result in results:
    print(result)
```

Step 18: Getting all the data which has lesser the 150 price

```
#Getting the BookID and BookName in which the Price is less than 150
q5 = """
SELECT BookID, BookName
FROM books
WHERE Price < 150;
"""

connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q5)
for result in results:
    print(result)
```

Step 19: Arrange the data BookID, BookName, and Price in ascending order based from the Price

```
q6 = """
SELECT BookID, BookName, Price
FROM books
ORDER BY Price;
"""
connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q6)
for result in results:
    print(result)
```

Step 20: Using SELECT statement to display data

```
#Using SELECT statement to display data
q7 = """
SELECT *
FROM books;
"""

connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q7)
for result in results:
    print(result)
```

Step 21: Returns a list of lists and then creates a pandas dataframe

```
# Returns a list of lists and then creates a pandas DataFrame
from_db = []

for result in results:
  result = list(result)
  from_db.append(result)


columns = ["BookID", "BookName", "Category", "Price", "Price_Range"]
df = pd.DataFrame(from_db, columns=columns)

display(df)
```

Step 22: Using UPDATE command to update specific data

```
#Using Update command
update = """
UPDATE books
SET Price = '143.00'
WHERE BookID = 1;
"""

connection = create_db_connection("localhost", "root", pw, db)
execute_query(connection, update)
```

Step 23: Checking the changed data if it is updated

```
#Check the updated data
q8 = """
SELECT *
FROM books
WHERE BookID = 1;
"""

connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q8)
for result in results:
    print(result)
```

Step 24: Using the DELETE command to delete a row

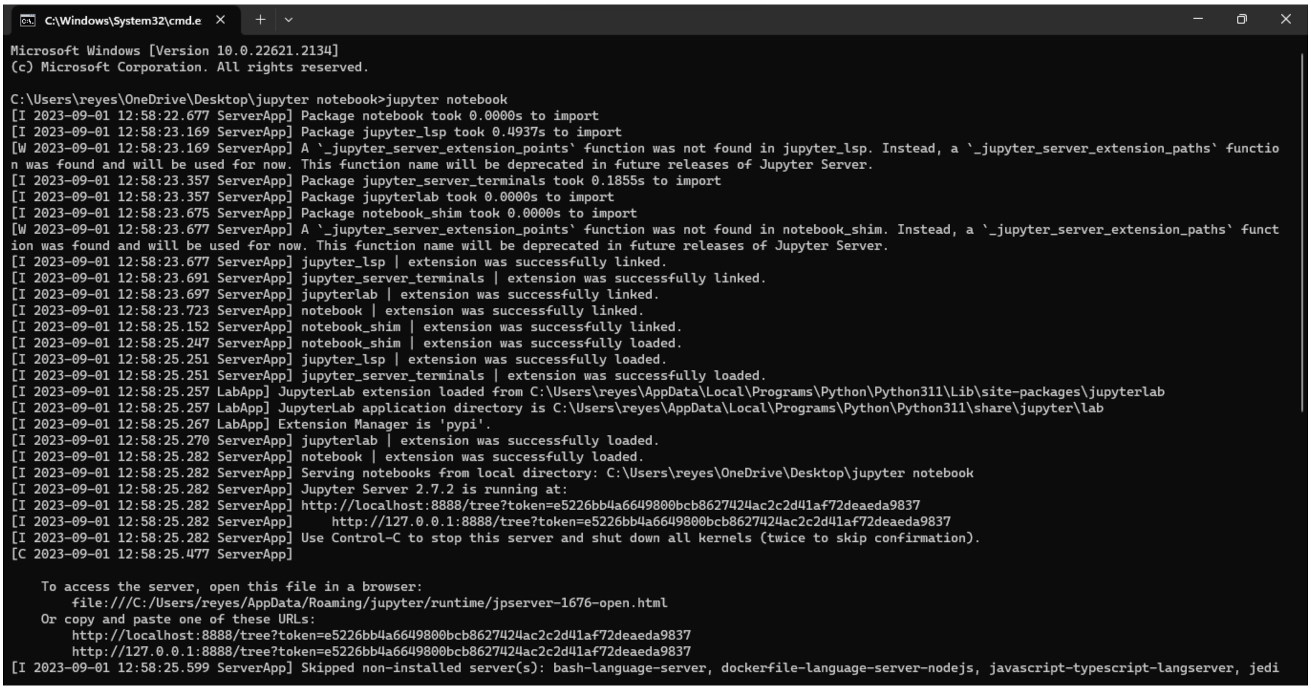
```
#Using the DELETE command
delete_book = """
DELETE FROM books
WHERE BookID = 8;
"""

connection = create_db_connection("localhost", "root", pw, db)
execute_query(connection, delete_book)
```

Step 25: Checking if the row was deleted

```
#Check if the BookID 8 was deleted
q9 = """
SELECT *
FROM books
"""

connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q9)
for result in results:
    print(result)
```

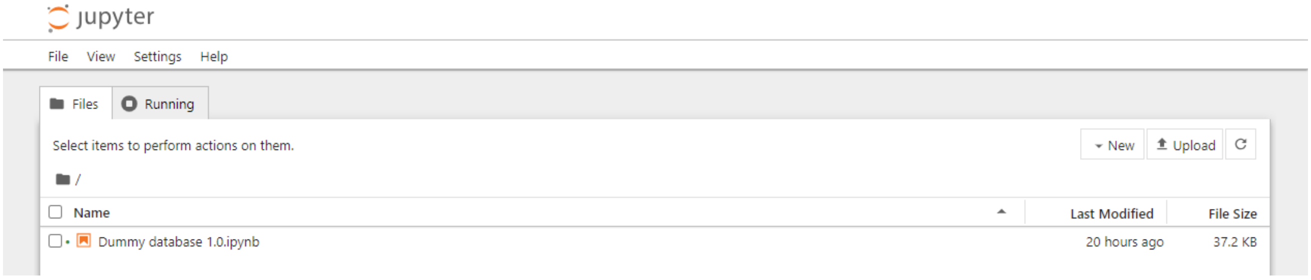
Screenshots of Work

Figure 1. Launching of Jupyter Notebook



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\reyes\OneDrive\Desktop\jupyter notebook>jupyter notebook
[I 2023-09-01 12:58:22.677 ServerApp] Package notebook took 0.0000s to import
[I 2023-09-01 12:58:23.169 ServerApp] Package jupyter_lsp took 0.4937s to import
[W 2023-09-01 12:58:23.169 ServerApp] A `_jupyter_server_extension_points` function was not found in jupyter_lsp. Instead, a `_jupyter_server_extension_paths` functio
n was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-09-01 12:58:23.357 ServerApp] Package jupyter_server_terminals took 0.1855s to import
[I 2023-09-01 12:58:23.357 ServerApp] Package jupyterlab took 0.0000s to import
[I 2023-09-01 12:58:23.675 ServerApp] Package notebook_shim took 0.0000s to import
[W 2023-09-01 12:58:23.677 ServerApp] A `_jupyter_server_extension_points` function was not found in notebook_shim. Instead, a `_jupyter_server_extension_paths` funct
ion was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-09-01 12:58:23.677 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2023-09-01 12:58:23.691 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2023-09-01 12:58:23.697 ServerApp] jupyterlab | extension was successfully linked.
[I 2023-09-01 12:58:23.723 ServerApp] notebook | extension was successfully linked.
[I 2023-09-01 12:58:25.152 ServerApp] notebook_shim | extension was successfully linked.
[I 2023-09-01 12:58:25.247 ServerApp] notebook_shim | extension was successfully loaded.
[I 2023-09-01 12:58:25.251 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2023-09-01 12:58:25.251 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2023-09-01 12:58:25.257 LabApp] JupyterLab extension loaded from C:\Users\reyes\AppData\Local\Programs\Python\Python311\Lib\site-packages\jupyterlab
[I 2023-09-01 12:58:25.257 LabApp] JupyterLab application directory is C:\Users\reyes\AppData\Local\Programs\Python\Python311\share\jupyter\lab
[I 2023-09-01 12:58:25.267 LabApp] Extension Manager is 'pypi'.
[I 2023-09-01 12:58:25.270 ServerApp] jupyterlab | extension was successfully loaded.
[I 2023-09-01 12:58:25.282 ServerApp] notebook | extension was successfully loaded.
[I 2023-09-01 12:58:25.282 ServerApp] Serving notebooks from local directory: C:\Users\reyes\OneDrive\Desktop\jupyter notebook
[I 2023-09-01 12:58:25.282 ServerApp] Jupyter Server 2.7.2 is running at:
[I 2023-09-01 12:58:25.282 ServerApp] http://localhost:8888/tree?token=e5226bb4a6649800bcb8627424ac2c2d41af72deaeda9837
[I 2023-09-01 12:58:25.282 ServerApp]     http://127.0.0.1:8888/tree?token=e5226bb4a6649800bcb8627424ac2c2d41af72deaeda9837
[I 2023-09-01 12:58:25.282 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2023-09-01 12:58:25.477 ServerApp]

    To access the server, open this file in a browser:
        file:///C:/Users/reyes/AppData/Roaming/jupyter/runtime/jpserver-1676-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/tree?token=e5226bb4a6649800bcb8627424ac2c2d41af72deaeda9837
        http://127.0.0.1:8888/tree?token=e5226bb4a6649800bcb8627424ac2c2d41af72deaeda9837
[I 2023-09-01 12:58:25.599 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi
```

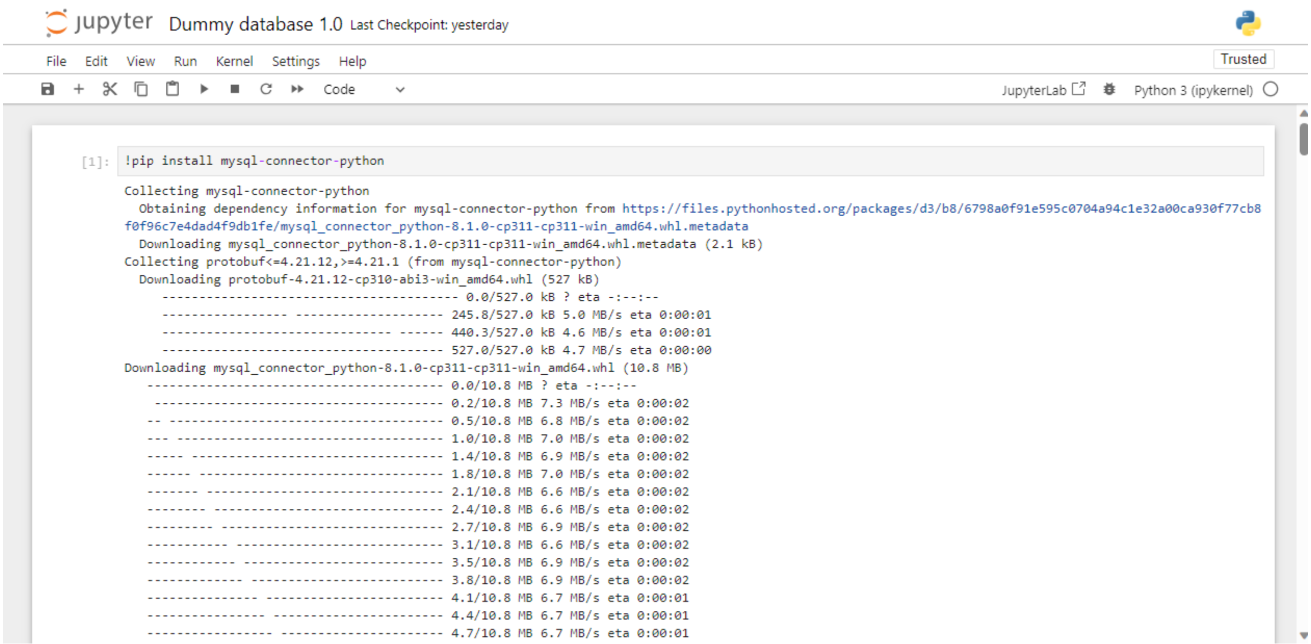
Figure 2. Jupyter Notebook localhost



Select items to perform actions on them.

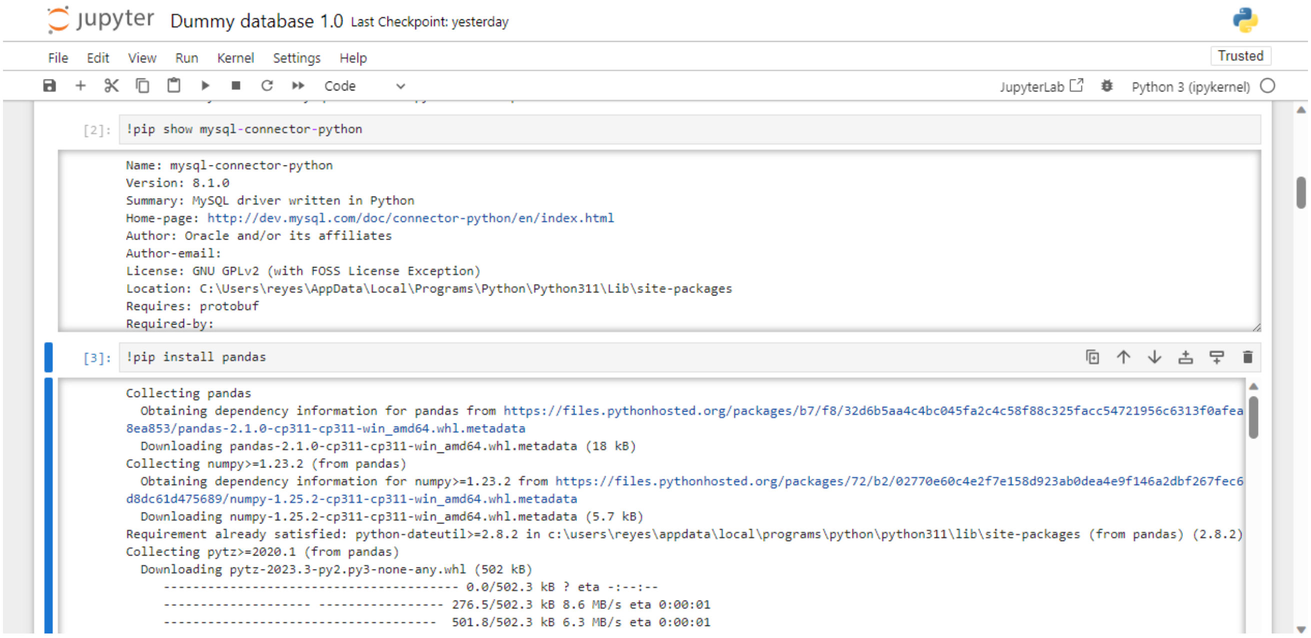
□ Name	Last Modified	File Size
□ Dummy database 1.0.ipynb	20 hours ago	37.2 KB

Figure 3. Installing the MySQL Connector Python Library



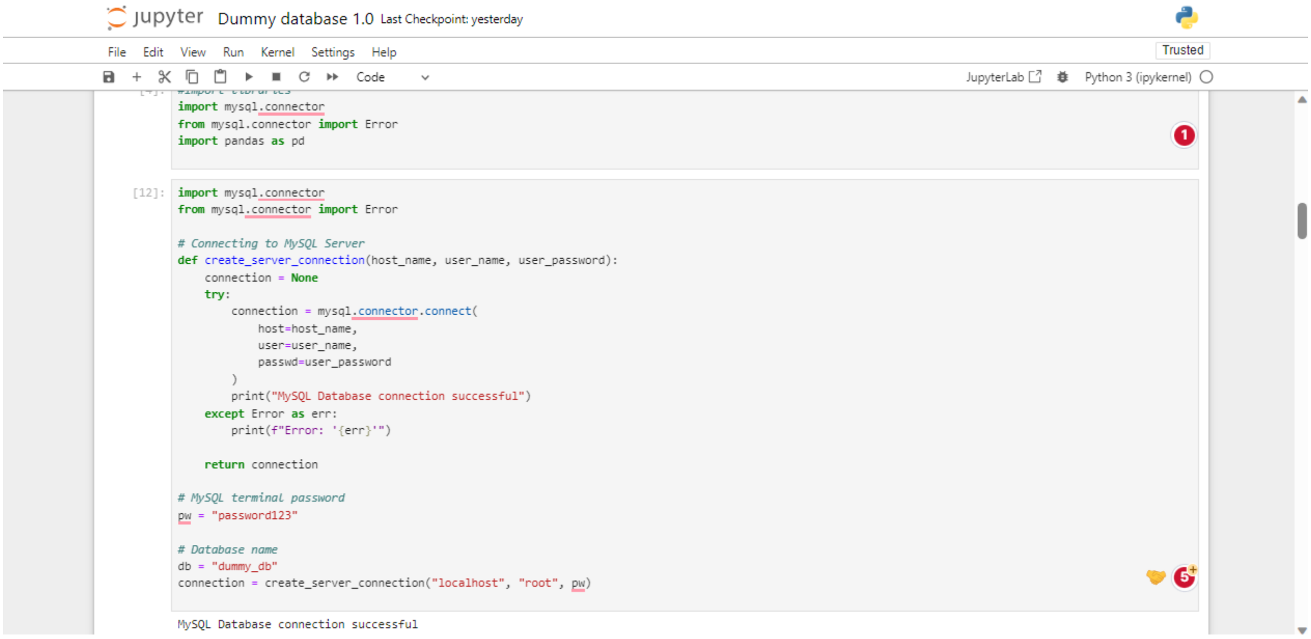
```
[1]: !pip install mysql-connector-python

Collecting mysql-connector-python
  Obtaining dependency information for mysql-connector-python from https://files.pythonhosted.org/packages/d3/b8/6798a0f91e595c0704a94c1e32a00ca930f77cb8
f0f96c7e4dad4f9db1fe/mysql_connector_python-8.1.0-cp311-cp311-win_amd64.whl.metadata
  Downloading mysql_connector_python-8.1.0-cp311-cp311-win_amd64.whl.metadata (2.1 kB)
Collecting protobuf<=4.21.12,>=4.21.1 (from mysql-connector-python)
  Downloading protobuf-4.21.12-cp310-abi3-win_amd64.whl (527 kB)
     ---------------------------------------- 0.0/527.0 kB ? eta -:--:--
     ----------------- -------------------- 245.8/527.0 kB 5.0 MB/s eta 0:00:01
     ------------------------------ ------ 440.3/527.0 kB 4.6 MB/s eta 0:00:01
     -------------------------------------- 527.0/527.0 kB 4.7 MB/s eta 0:00:00
Downloading mysql_connector_python-8.1.0-cp311-cp311-win_amd64.whl (10.8 MB)
   ---------------------------------------- 0.0/10.8 MB ? eta -:--:--
    --------------------------------------- 0.2/10.8 MB 7.3 MB/s eta 0:00:02
   -- ------------------------------------- 0.5/10.8 MB 6.8 MB/s eta 0:00:02
   --- ------------------------------------ 1.0/10.8 MB 7.0 MB/s eta 0:00:02
   ----- ---------------------------------- 1.4/10.8 MB 6.9 MB/s eta 0:00:02
   ------ --------------------------------- 1.8/10.8 MB 7.0 MB/s eta 0:00:02
   ------- -------------------------------- 2.1/10.8 MB 6.6 MB/s eta 0:00:02
   -------- ------------------------------- 2.4/10.8 MB 6.6 MB/s eta 0:00:02
   --------- ------------------------------ 2.7/10.8 MB 6.9 MB/s eta 0:00:02
   ----------- ---------------------------- 3.1/10.8 MB 6.6 MB/s eta 0:00:02
   ------------ --------------------------- 3.5/10.8 MB 6.9 MB/s eta 0:00:02
   ------------- -------------------------- 3.8/10.8 MB 6.9 MB/s eta 0:00:02
   --------------- ------------------------ 4.1/10.8 MB 6.7 MB/s eta 0:00:01
   ---------------- ----------------------- 4.4/10.8 MB 6.7 MB/s eta 0:00:01
   ----------------- ---------------------- 4.7/10.8 MB 6.7 MB/s eta 0:00:01
```

Figure 4. Installing the Pandas Python Library



```
[2]: !pip show mysql-connector-python

Name: mysql-connector-python
Version: 8.1.0
Summary: MySQL driver written in Python
Home-page: http://dev.mysql.com/doc/connector-python/en/index.html
Author: Oracle and/or its affiliates
Author-email:
License: GNU GPLv2 (with FOSS License Exception)
Location: C:\Users\reyes\AppData\Local\Programs\Python\Python311\Lib\site-packages
Requires: protobuf
Required-by:

[3]: !pip install pandas

Collecting pandas
  Obtaining dependency information for pandas from https://files.pythonhosted.org/packages/b7/f8/32d6b5aa4c4bc045fa2c4c58f88c325facc54721956c6313f0afea
8ea853/pandas-2.1.0-cp311-cp311-win_amd64.whl.metadata
  Downloading pandas-2.1.0-cp311-cp311-win_amd64.whl.metadata (18 kB)
Collecting numpy>=1.23.2 (from pandas)
  Obtaining dependency information for numpy>=1.23.2 from https://files.pythonhosted.org/packages/72/b2/02770e60c4e2f7e158d923ab0dea4e9f146a2dbf267fec6
d8dc61d475689/numpy-1.25.2-cp311-cp311-win_amd64.whl.metadata
  Downloading numpy-1.25.2-cp311-cp311-win_amd64.whl.metadata (5.7 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\reyes\appdata\local\programs\python\python311\lib\site-packages (from pandas) (2.8.2)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2023.3-py2.py3-none-any.whl (502 kB)
     ---------------------------------------- 0.0/502.3 kB ? eta -:--:--
     -------------------- ----------------- 276.5/502.3 kB 8.6 MB/s eta 0:00:01
     -------------------------------------  501.8/502.3 kB 6.3 MB/s eta 0:00:01
```

Figure 5. Importing the installed libraries and connecting to MySQL Server

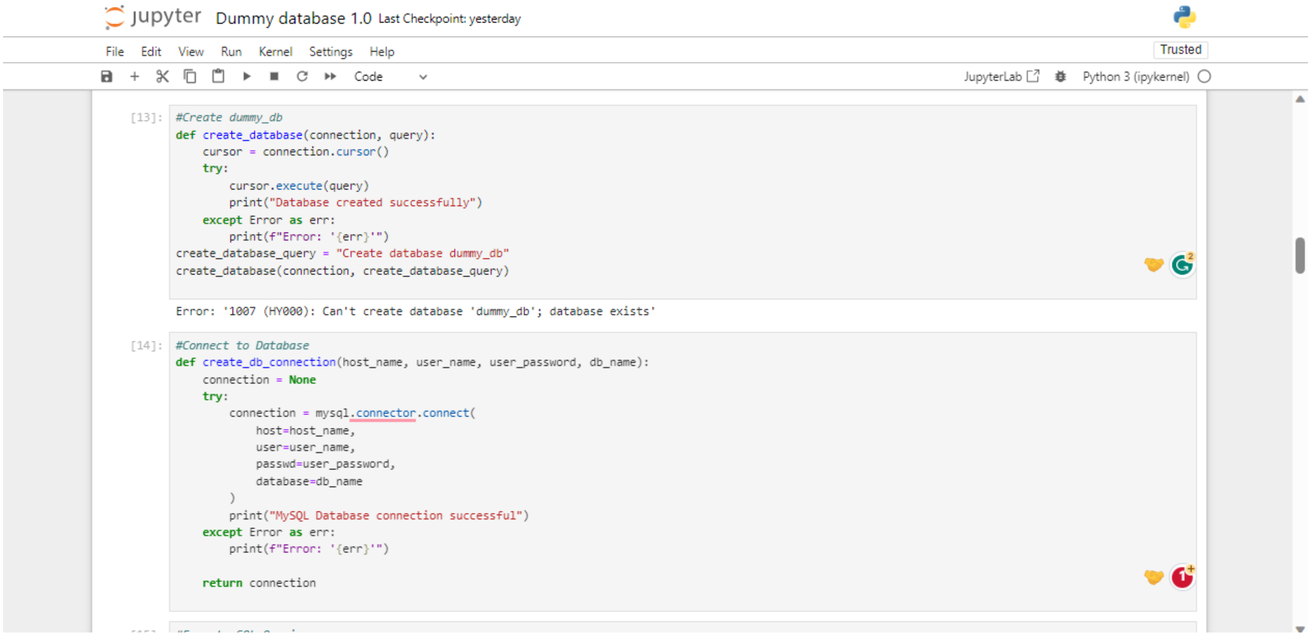


```
[4]: #import libraries
import mysql.connector
from mysql.connector import Error
import pandas as pd
```

```
[12]: import mysql.connector
from mysql.connector import Error

# Connecting to MySQL Server
def create_server_connection(host_name, user_name, user_password):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password
        )
        print("MySQL Database connection successful")
    except Error as err:
        print(f"Error: '{err}'")

    return connection

# MySQL terminal password
pw = "password123"

# Database name
db = "dummy_db"
connection = create_server_connection("localhost", "root", pw)
```

```
MySQL Database connection successful
```

Figure 6. Creating and connecting to a database called “dummy_db”

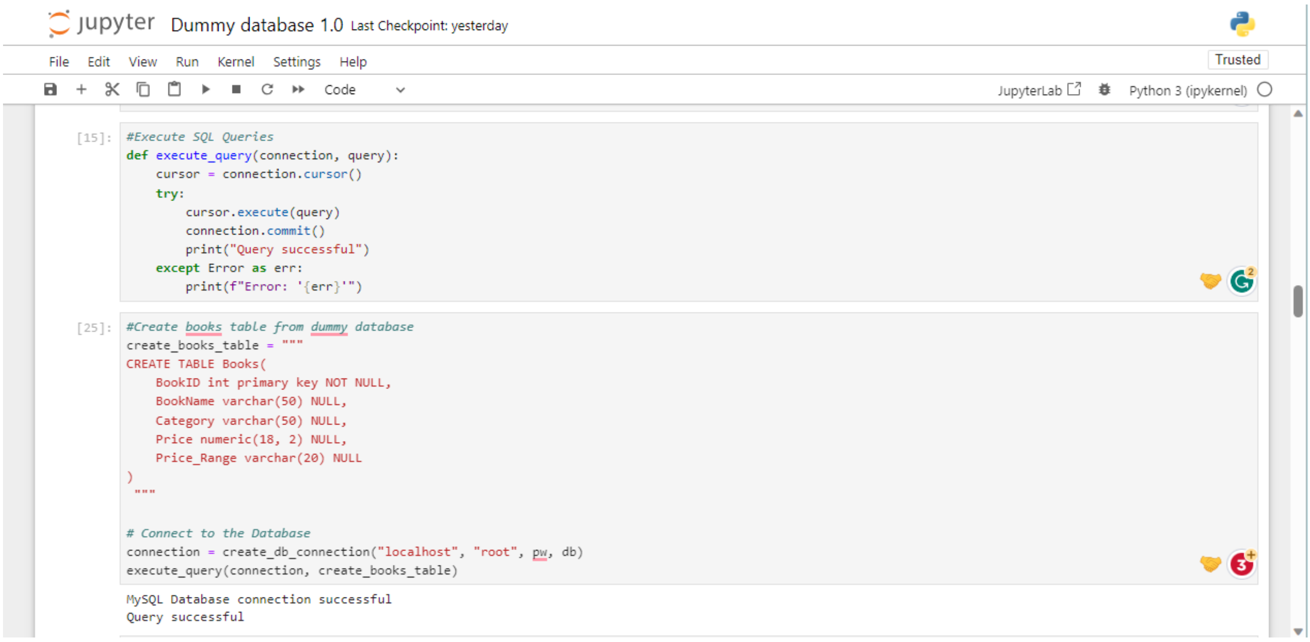


```
[13]: #Create dummy_db
def create_database(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        print("Database created successfully")
    except Error as err:
        print(f"Error: '{err}'")
create_database_query = "Create database dummy_db"
create_database(connection, create_database_query)
```

```
Error: '1007 (HY000): Can't create database 'dummy_db'; database exists'
```

```
[14]: #Connect to Database
def create_db_connection(host_name, user_name, user_password, db_name):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password,
            database=db_name
        )
        print("MySQL Database connection successful")
    except Error as err:
        print(f"Error: '{err}'")

    return connection
```

Figure 7. SQL Queries Execution Connection and creating table called “books”

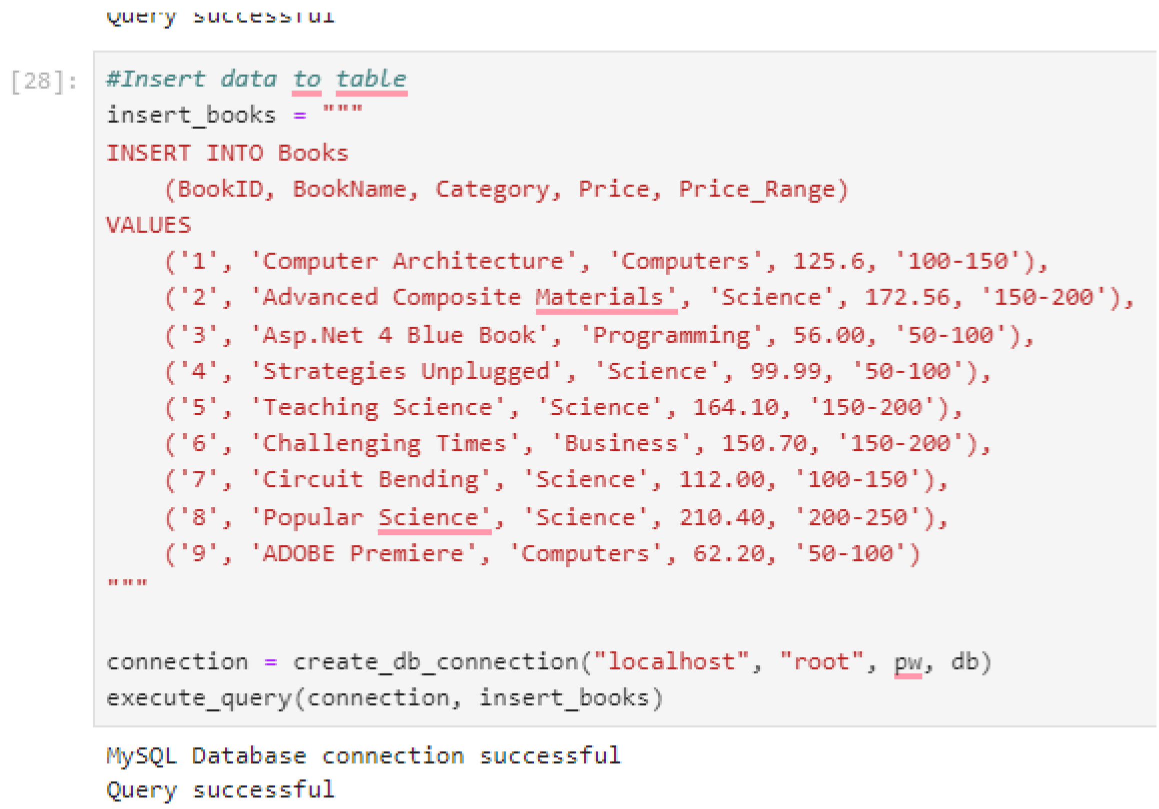


```
[15]: #Execute SQL Queries
      def execute_query(connection, query):
          cursor = connection.cursor()
          try:
              cursor.execute(query)
              connection.commit()
              print("Query successful")
          except Error as err:
              print(f"Error: '{err}'")
```

```
[25]: #Create books table from dummy database
      create_books_table = """
      CREATE TABLE Books(
          BookID int primary key NOT NULL,
          BookName varchar(50) NULL,
          Category varchar(50) NULL,
          Price numeric(18, 2) NULL,
          Price_Range varchar(20) NULL
      )
       """

      # Connect to the Database
      connection = create_db_connection("localhost", "root", pw, db)
      execute_query(connection, create_books_table)
```

```
MySQL Database connection successful
Query successful
```

Figure 8. Inserting data into the books table

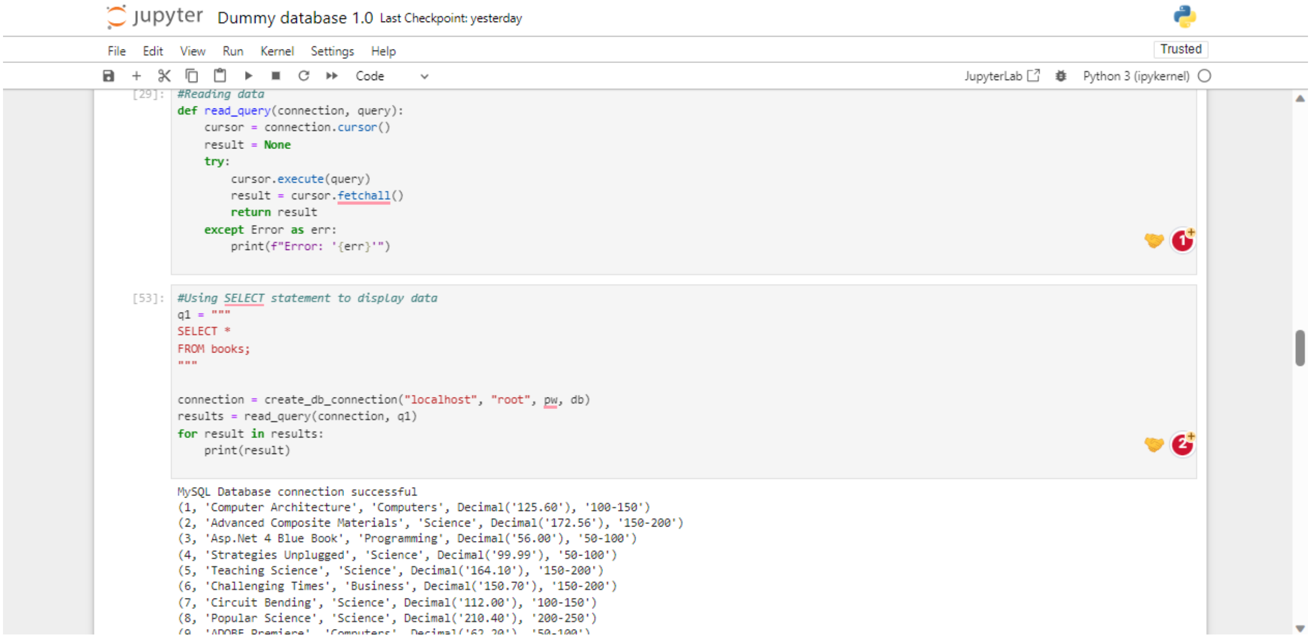


```
Query successful
```

```
[28]: #Insert data to table
      insert_books = """
      INSERT INTO Books
          (BookID, BookName, Category, Price, Price_Range)
      VALUES
          ('1', 'Computer Architecture', 'Computers', 125.6, '100-150'),
          ('2', 'Advanced Composite Materials', 'Science', 172.56, '150-200'),
          ('3', 'Asp.Net 4 Blue Book', 'Programming', 56.00, '50-100'),
          ('4', 'Strategies Unplugged', 'Science', 99.99, '50-100'),
          ('5', 'Teaching Science', 'Science', 164.10, '150-200'),
          ('6', 'Challenging Times', 'Business', 150.70, '150-200'),
          ('7', 'Circuit Bending', 'Science', 112.00, '100-150'),
          ('8', 'Popular Science', 'Science', 210.40, '200-250'),
          ('9', 'ADOBE Premiere', 'Computers', 62.20, '50-100')
      """

      connection = create_db_connection("localhost", "root", pw, db)
      execute_query(connection, insert_books)
```

```
MySQL Database connection successful
Query successful
```

Figure 9. Reading and siplaying data using SQL command

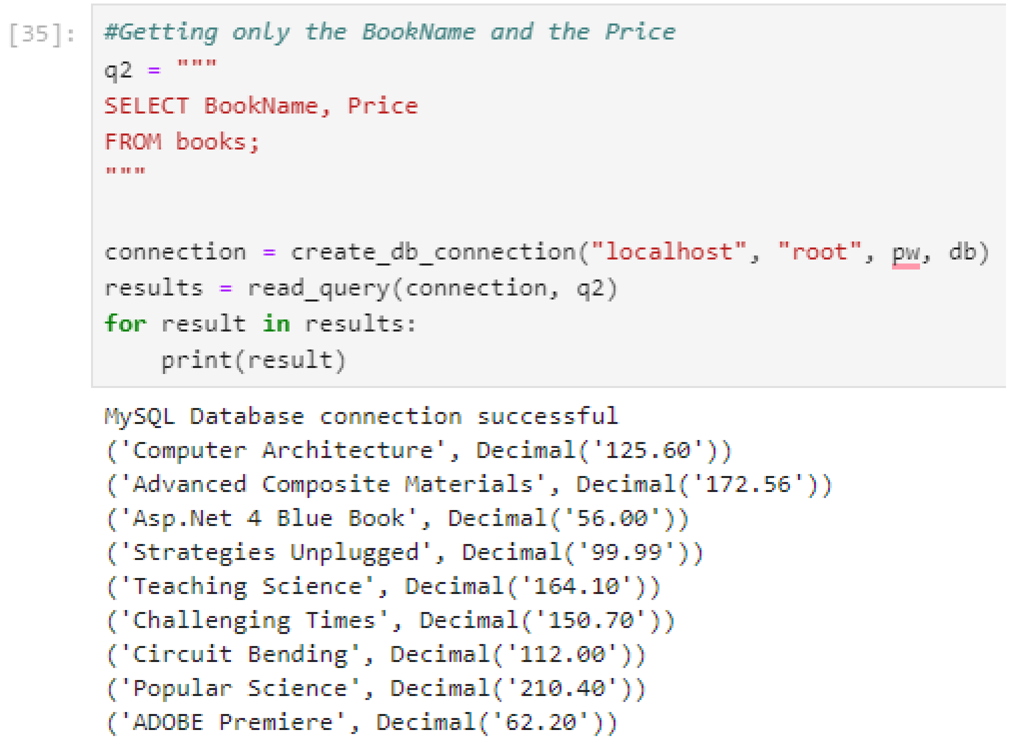

jupyter Dummy database 1.0 Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[29]: #Reading data
      def read_query(connection, query):
          cursor = connection.cursor()
          result = None
          try:
              cursor.execute(query)
              result = cursor.fetchall()
              return result
          except Error as err:
              print(f"Error: '{err}'")
```

```
[53]: #Using SELECT statement to display data
      q1 = """
      SELECT *
      FROM books;
      """

      connection = create_db_connection("localhost", "root", pw, db)
      results = read_query(connection, q1)
      for result in results:
          print(result)
```

```
MySQL Database connection successful
(1, 'Computer Architecture', 'Computers', Decimal('125.60'), '100-150')
(2, 'Advanced Composite Materials', 'Science', Decimal('172.56'), '150-200')
(3, 'Asp.Net 4 Blue Book', 'Programming', Decimal('56.00'), '50-100')
(4, 'Strategies Unplugged', 'Science', Decimal('99.99'), '50-100')
(5, 'Teaching Science', 'Science', Decimal('164.10'), '150-200')
(6, 'Challenging Times', 'Business', Decimal('150.70'), '150-200')
(7, 'Circuit Bending', 'Science', Decimal('112.00'), '100-150')
(8, 'Popular Science', 'Science', Decimal('210.40'), '200-250')
(9, 'ADOBE Premiere', 'Computers', Decimal('62.20'), '50-100')
```

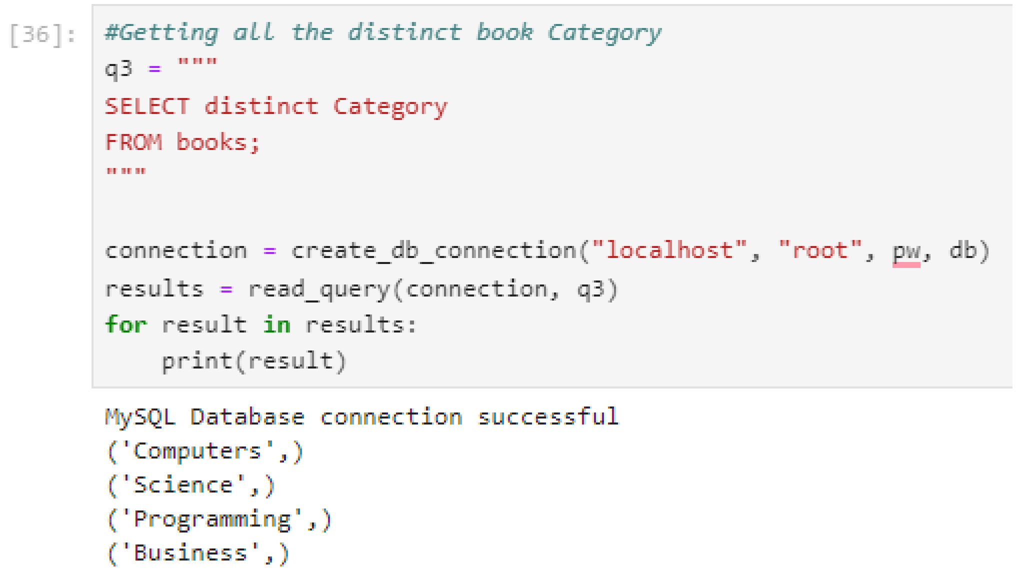
Figure 10. Getting the BookName and Price from the table



```
[35]: #Getting only the BookName and the Price
      q2 = """
      SELECT BookName, Price
      FROM books;
      """

      connection = create_db_connection("localhost", "root", pw, db)
      results = read_query(connection, q2)
      for result in results:
          print(result)
```

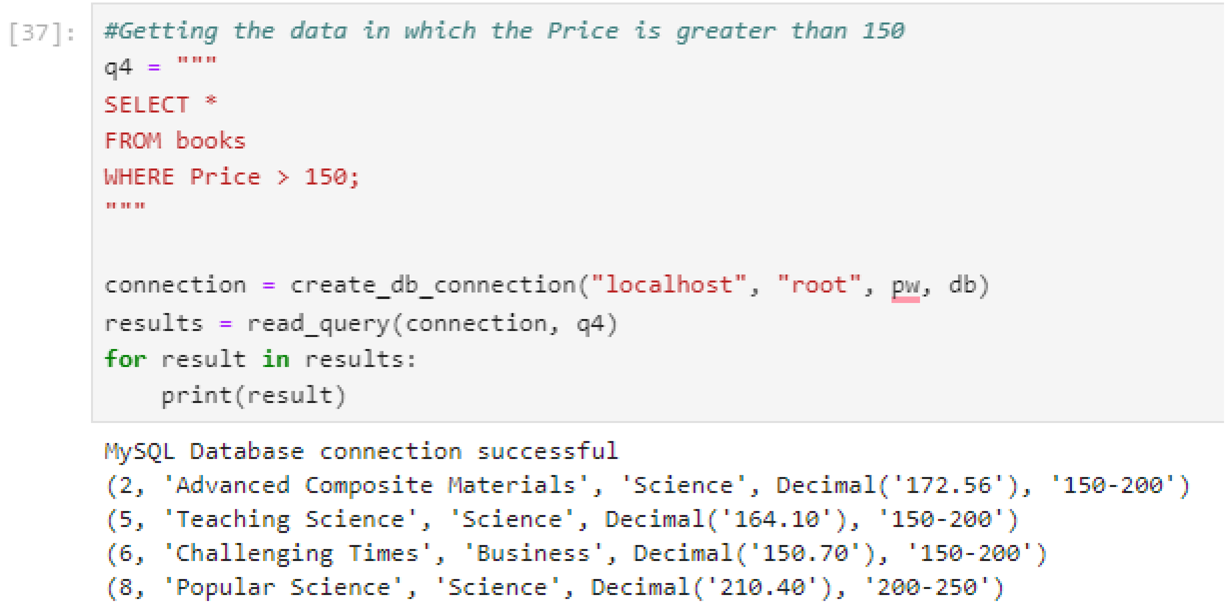
```
MySQL Database connection successful
('Computer Architecture', Decimal('125.60'))
('Advanced Composite Materials', Decimal('172.56'))
('Asp.Net 4 Blue Book', Decimal('56.00'))
('Strategies Unplugged', Decimal('99.99'))
('Teaching Science', Decimal('164.10'))
('Challenging Times', Decimal('150.70'))
('Circuit Bending', Decimal('112.00'))
('Popular Science', Decimal('210.40'))
('ADOBE Premiere', Decimal('62.20'))
```

Figure 11. Getting all the distinct book category



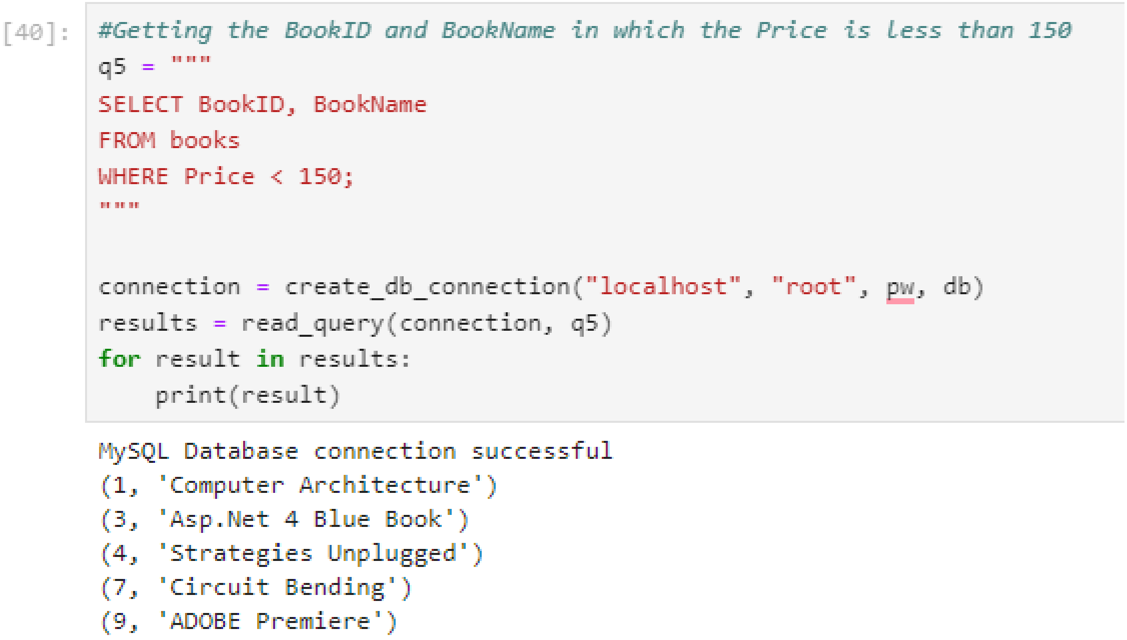
```
[36]: #Getting all the distinct book Category
      q3 = """
      SELECT distinct Category
      FROM books;
      """

      connection = create_db_connection("localhost", "root", pw, db)
      results = read_query(connection, q3)
      for result in results:
          print(result)

      MySQL Database connection successful
      ('Computers',)
      ('Science',)
      ('Programming',)
      ('Business',)
```

Figure 12. Getting the data that has Price over 150



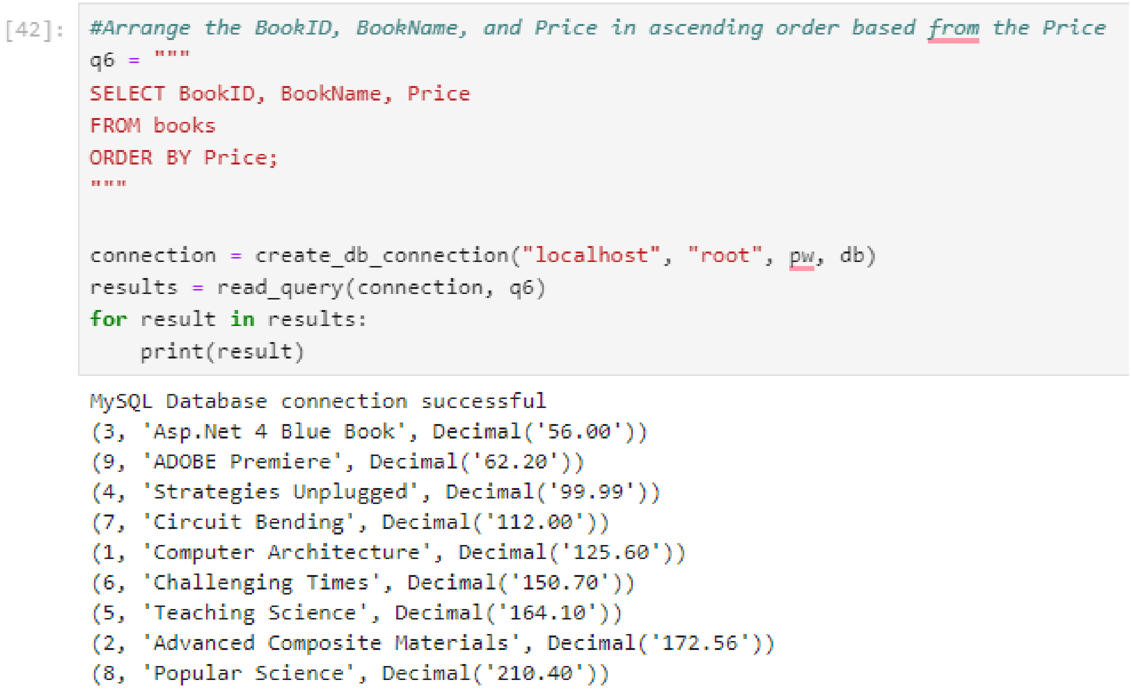
```
[37]: #Getting the data in which the Price is greater than 150
      q4 = """
      SELECT *
      FROM books
      WHERE Price > 150;
      """

      connection = create_db_connection("localhost", "root", pw, db)
      results = read_query(connection, q4)
      for result in results:
          print(result)

      MySQL Database connection successful
      (2, 'Advanced Composite Materials', 'Science', Decimal('172.56'), '150-200')
      (5, 'Teaching Science', 'Science', Decimal('164.10'), '150-200')
      (6, 'Challenging Times', 'Business', Decimal('150.70'), '150-200')
      (8, 'Popular Science', 'Science', Decimal('210.40'), '200-250')
```

Figure 13. Getting the data that has Price lesser than 150



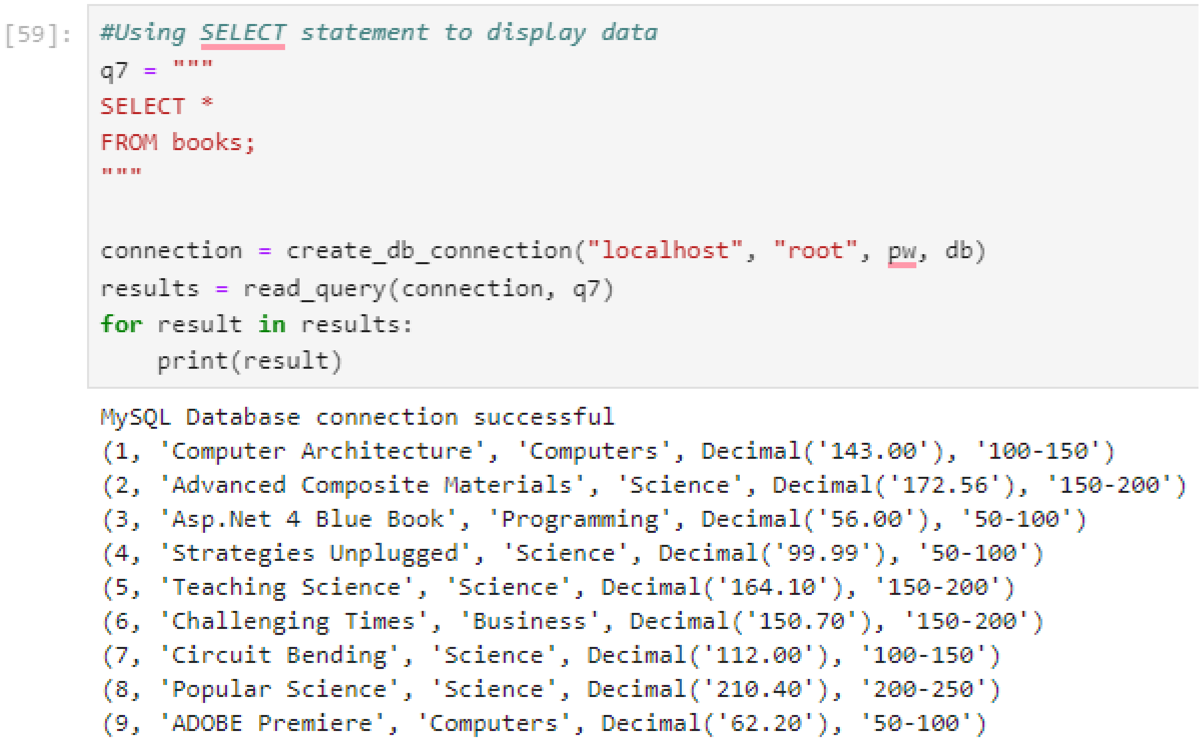
```
[40]: #Getting the BookID and BookName in which the Price is less than 150
      q5 = """
      SELECT BookID, BookName
      FROM books
      WHERE Price < 150;
      """

      connection = create_db_connection("localhost", "root", pw, db)
      results = read_query(connection, q5)
      for result in results:
          print(result)

      MySQL Database connection successful
      (1, 'Computer Architecture')
      (3, 'Asp.Net 4 Blue Book')
      (4, 'Strategies Unplugged')
      (7, 'Circuit Bending')
      (9, 'ADOBE Premiere')
```

Figure 14. Arrange in ascending order based from the price



```
[42]: #Arrange the BookID, BookName, and Price in ascending order based from the Price
      q6 = """
      SELECT BookID, BookName, Price
      FROM books
      ORDER BY Price;
      """

      connection = create_db_connection("localhost", "root", pw, db)
      results = read_query(connection, q6)
      for result in results:
          print(result)

      MySQL Database connection successful
      (3, 'Asp.Net 4 Blue Book', Decimal('56.00'))
      (9, 'ADOBE Premiere', Decimal('62.20'))
      (4, 'Strategies Unplugged', Decimal('99.99'))
      (7, 'Circuit Bending', Decimal('112.00'))
      (1, 'Computer Architecture', Decimal('125.60'))
      (6, 'Challenging Times', Decimal('150.70'))
      (5, 'Teaching Science', Decimal('164.10'))
      (2, 'Advanced Composite Materials', Decimal('172.56'))
      (8, 'Popular Science', Decimal('210.40'))
```

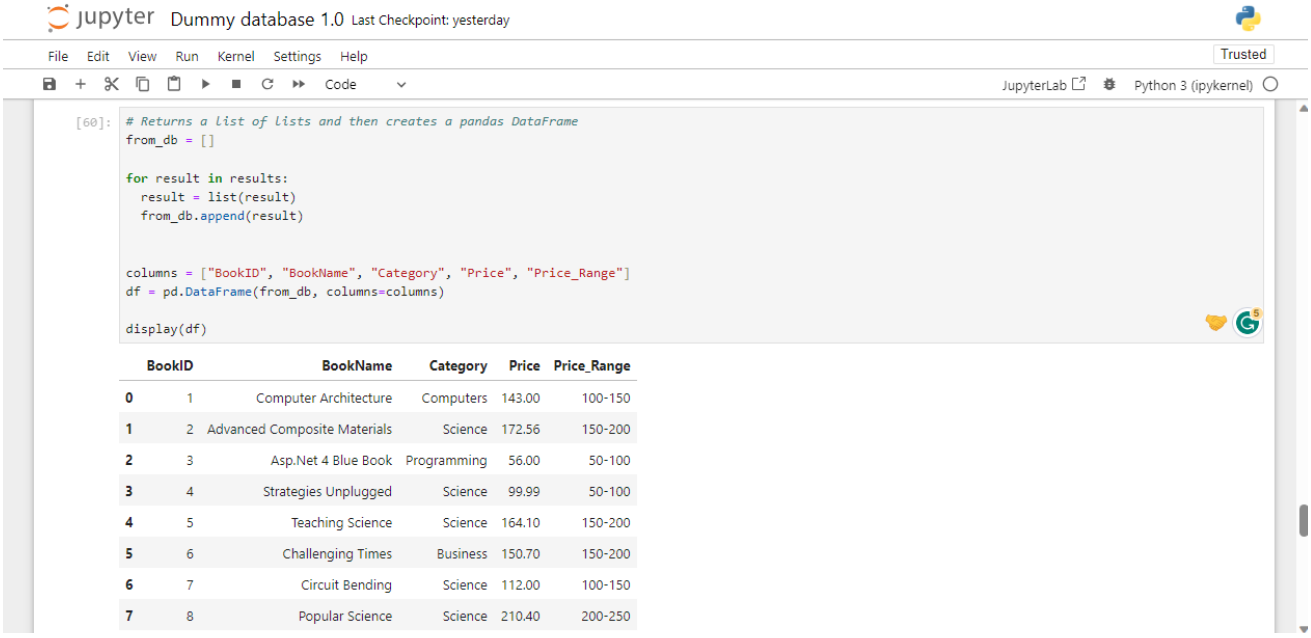
Figure 15. Display all the data in the books table



```
[59]: #Using SELECT statement to display data
      q7 = """
      SELECT *
      FROM books;
      """

      connection = create_db_connection("localhost", "root", pw, db)
      results = read_query(connection, q7)
      for result in results:
          print(result)
```

```
MySQL Database connection successful
(1, 'Computer Architecture', 'Computers', Decimal('143.00'), '100-150')
(2, 'Advanced Composite Materials', 'Science', Decimal('172.56'), '150-200')
(3, 'Asp.Net 4 Blue Book', 'Programming', Decimal('56.00'), '50-100')
(4, 'Strategies Unplugged', 'Science', Decimal('99.99'), '50-100')
(5, 'Teaching Science', 'Science', Decimal('164.10'), '150-200')
(6, 'Challenging Times', 'Business', Decimal('150.70'), '150-200')
(7, 'Circuit Bending', 'Science', Decimal('112.00'), '100-150')
(8, 'Popular Science', 'Science', Decimal('210.40'), '200-250')
(9, 'ADOBE Premiere', 'Computers', Decimal('62.20'), '50-100')
```

Figure 16. Returns list then create panda dataframe and displaying it

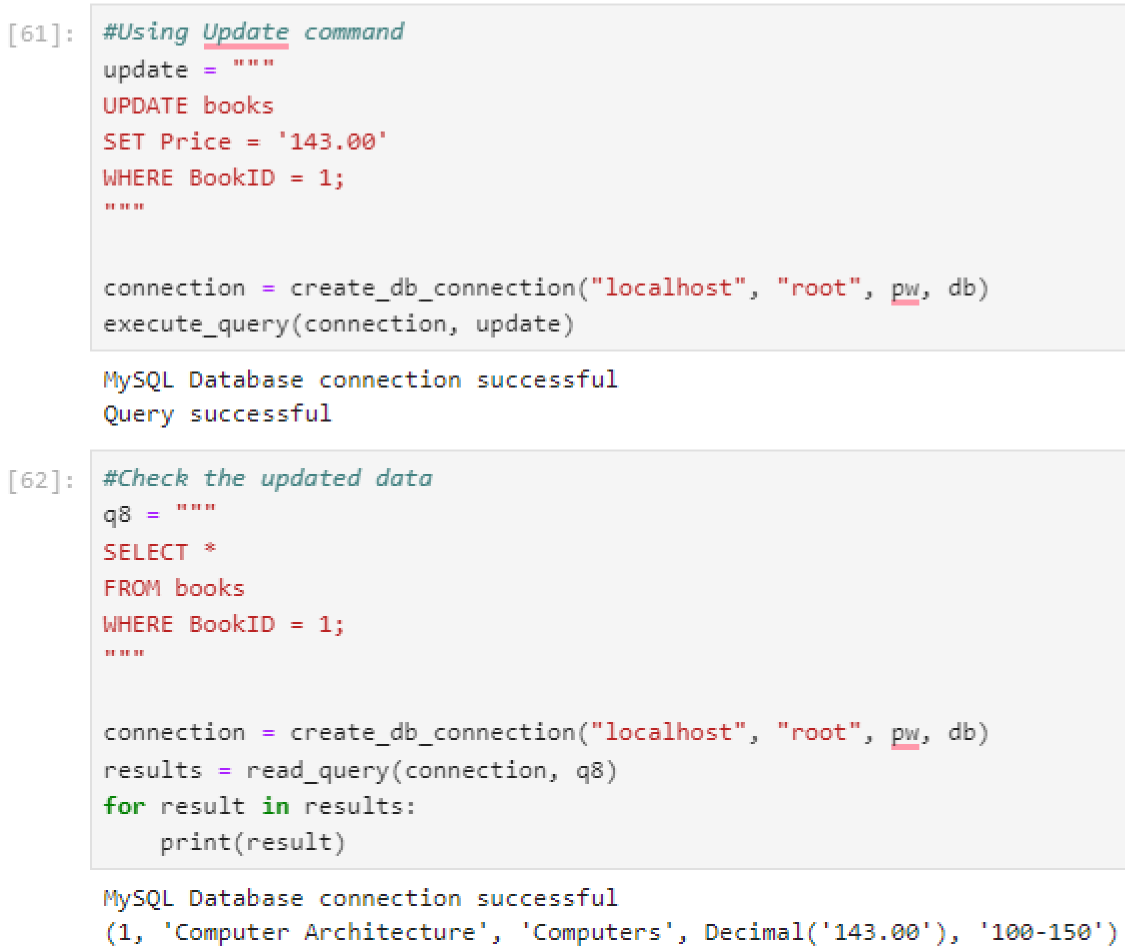


Jupyter Dummy database 1.0 Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[60]: # Returns a list of lists and then creates a pandas DataFrame
      from_db = []

      for result in results:
        result = list(result)
        from_db.append(result)


      columns = ["BookID", "BookName", "Category", "Price", "Price_Range"]
      df = pd.DataFrame(from_db, columns=columns)

      display(df)
```

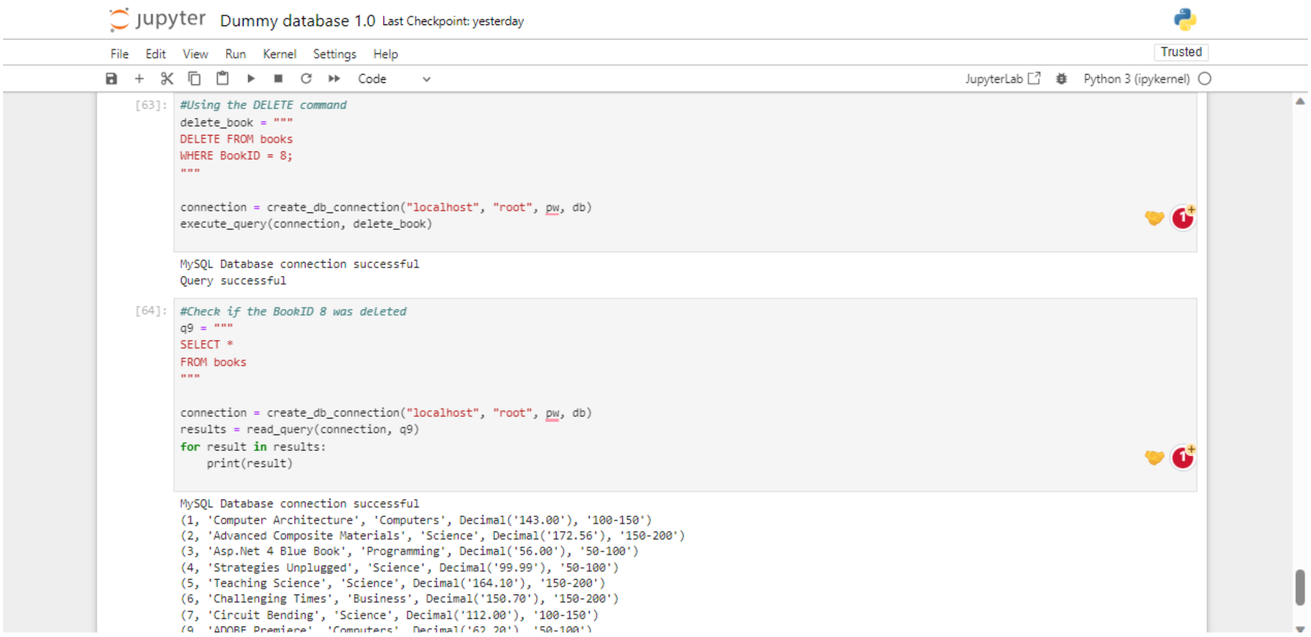
	BookID	BookName	Category	Price	Price_Range
0	1	Computer Architecture	Computers	143.00	100-150
1	2	Advanced Composite Materials	Science	172.56	150-200
2	3	Asp.Net 4 Blue Book	Programming	56.00	50-100
3	4	Strategies Unplugged	Science	99.99	50-100
4	5	Teaching Science	Science	164.10	150-200
5	6	Challenging Times	Business	150.70	150-200
6	7	Circuit Bending	Science	112.00	100-150
7	8	Popular Science	Science	210.40	200-250

Figure 17. Using SQL UPDATE command



```
[61]: #Using Update command
      update = """
      UPDATE books
      SET Price = '143.00'
      WHERE BookID = 1;
      """

      connection = create_db_connection("localhost", "root", pw, db)
      execute_query(connection, update)

      MySQL Database connection successful
      Query successful

[62]: #Check the updated data
      q8 = """
      SELECT *
      FROM books
      WHERE BookID = 1;
      """

      connection = create_db_connection("localhost", "root", pw, db)
      results = read_query(connection, q8)
      for result in results:
          print(result)

      MySQL Database connection successful
      (1, 'Computer Architecture', 'Computers', Decimal('143.00'), '100-150')
```

Figure 18. Using SQL DELETE command


jupyter Dummy database 1.0 Last Checkpoint: yesterday
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[63]:
[64]:

```
#Using the DELETE command
delete_book = """
DELETE FROM books
WHERE BookID = 8;
"""

connection = create_db_connection("localhost", "root", pw, db)
execute_query(connection, delete_book)
```

```
MySQL Database connection successful
Query successful
```

```
#Check if the BookID 8 was deleted
q9 = """
SELECT *
FROM books
"""

connection = create_db_connection("localhost", "root", pw, db)
results = read_query(connection, q9)
for result in results:
    print(result)
```

```
MySQL Database connection successful
(1, 'Computer Architecture', 'Computers', Decimal('143.00'), '100-150')
(2, 'Advanced Composite Materials', 'Science', Decimal('172.56'), '150-200')
(3, 'Asp.Net 4 Blue Book', 'Programming', Decimal('56.00'), '50-100')
(4, 'Strategies Unplugged', 'Science', Decimal('99.99'), '50-100')
(5, 'Teaching Science', 'Science', Decimal('164.10'), '150-200')
(6, 'Challenging Times', 'Business', Decimal('150.70'), '150-200')
(7, 'Circuit Bending', 'Science', Decimal('112.00'), '100-150')
(9, 'ADOBE Premiere', 'Computers', Decimal('62.20'), '50-100')
```