SESSION 12
MONGODB DATABASE CONNECTIVITY WITH PYTHON

Learning Objectives

In this session, students will learn to:

- Describe how to connect Python with MongoDB
- Explain how to install `PyMongo` Driver
- Describe how to create a database and collection in MongoDB using Python
- Describe the ways to query, sort, update, and delete documents in MongoDB using Python

Python is a popular computer programming language used to automate processes, analyze data, and create Websites. It is a general-purpose programming language that can be used in various application domains. Python can be integrated with MongoDB to create robust and versatile database applications. These database applications use Python as the application interface and MongoDB as the database server.

This session will provide an overview of connecting Python with MongoDB. It will also explain how to install the PyMongo Driver. This session will explore the methods to create a database and collection in MongoDB using Python. It will also explain the methods to query, sort, update, and delete documents in MongoDB using Python.

12.1 Overview of Connecting Python with MongoDB

As discussed in the previous sessions, MongoDB is a popular choice to build extremely scalable and reliable databases. Python is a versatile programming language with an extensive standard library. Therefore, combination of Python and MongoDB has proven to be excellent for developing database applications.

`PyMongo` is MongoDB's official Python driver. `PyMongo` helps to create a connection between Python and MongoDB. It also provides a wide range of methods that can be used to insert, query, update, and delete data from the database. The documents retrieved using `PyMongo` can be accessed using data structures in Python.

12.2 Install `PyMongo` Driver

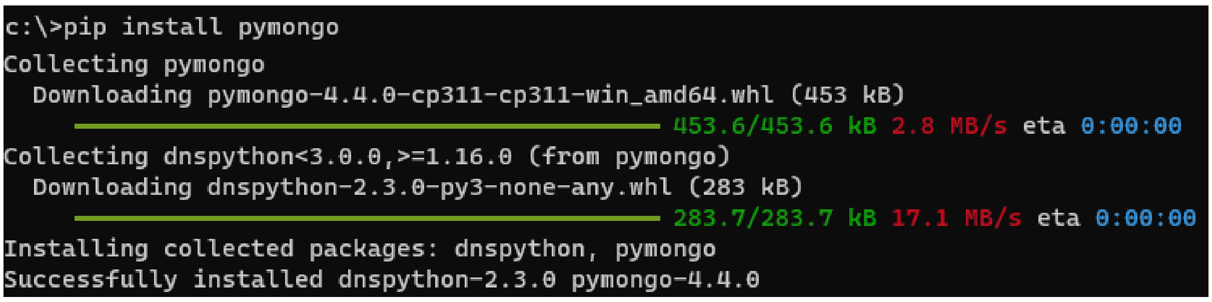
Before installing `PyMongo`, users must download and install the latest version of Python from the URL: https://www.python.org/downloads/

To install `PyMongo`:

1. Open the Command Prompt.
2. Navigate to the path where Python is installed on the local system.
3. Execute the command as:

```
pip install pymongo
```

   The command is executed as shown in Figure 12.1.



```
c:\>pip install pymongo
Collecting pymongo
  Downloading pymongo-4.4.0-cp311-cp311-win_amd64.whl (453 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 453.6/453.6 kB 2.8 MB/s eta 0:00:00
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.3.0-py3-none-any.whl (283 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 283.7/283.7 kB 17.1 MB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.3.0 pymongo-4.4.0
```

Figure 12.1: Install `PyMongo`

The `PyMongo` driver is now installed. The user can now use the driver to connect Python with MongoDB.

12.3 Create a Database and Collection Using Python

To create a database and collection in MongoDB using Python, the user must write Python code for:

1. Establishing a connection with MongoDB
2. Creating a database in MongoDB
3. Creating a collection in the database
4. Inserting documents into the collection

This code must be entered in a .py(Python) file and executed.

To facilitate connection to MongoDB from Python, `PyMongo` includes the `MongoClient` class. This class acts as a client to the MongoDB server. To connect to a MongoDB database, users must first create a `MongoClient` instance by importing it. The command to import `MongoClient` is:

```
from pymongo import MongoClient
client = MongoClient()
```

The command `client = MongoClient()` will create a connection to the default host and default port.

Now, the output from Python applications when used with large databases such as MongoDB can be huge. Also, MongoDB provides the output data in JavaScript Object Notation (JSON) format, which can be difficult to read. To make the output readable, Python offers the `pprint` module. `pprint` stands for 'pretty print'. So, the next step is to import `pprint` using the command as:

```
import pprint
```

Consider that the user wants to create a database named `library`. To do so, the user can use the command as:

```
db = client.library
```

Consider that the user wants to create a collection named `library_details`. To do so, the user can use the command as:

```
lib_collection = db.library_details
```

Consider that the user wants to insert some documents into this collection. Code Snippet 1 lists the code to perform this action:

Code Snippet 1:

```
lib =[
     {"book_id": "1010","book_name": "Python
     Programming", "book_author": "John M Zelle",
     "volume":3},

     {"book_id": "1019","book_name": "Python for Data
     Analysis","book_author": "Wes Mckinney",
     "volume":1},

     {"book_id": "1009","book_name": "Python
     Cookbook","book_author":"David Beazley",
     "volume":3}
     ]
library_details.insert_many(lib)
pprint.pprint("Documents Inserted successfully")
```

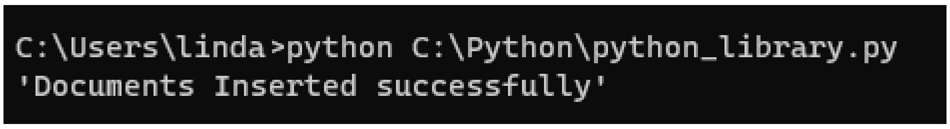
This code will insert three documents into the `library_details` collection. The `pprint` function in this code will print the message on the screen indicating successful insertion of documents.

As discussed earlier, the code to establish the connection, create a database, create a collection, and insert documents into the collection must be placed in a Python file. For example, let us enter the code into a file named `python_library.py` and save the file to the `C:\Python` folder.

To execute the code in the `python_library.py` file, at the command prompt, run the command as:

```
python C:\Python\python_library.py
```

The command executes as shown in Figure 12.2.



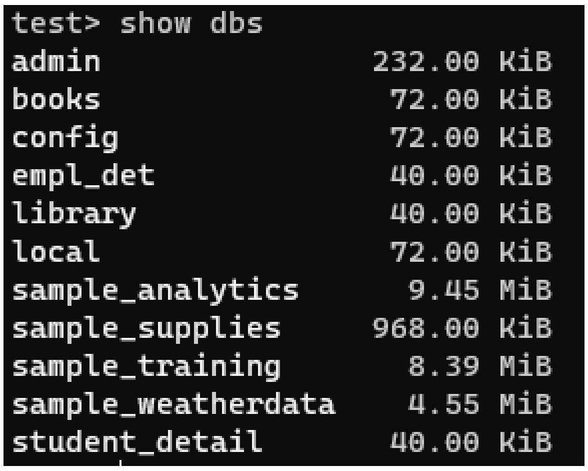
```
C:\Users\linda>python C:\Python\python_library.py
'Documents Inserted successfully'
```

Figure 12.2: Execution of Python File

To view the created database, connect to Mongo Shell and execute the command as:

```
show dbs
```

The command executes as shown in Figure 12.3.



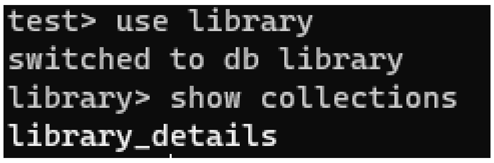
```
test> show dbs
admin                 232.00 KiB
books                  72.00 KiB
config                 72.00 KiB
empl_det               40.00 KiB
library                40.00 KiB
local                  72.00 KiB
sample_analytics        9.45 MiB
sample_supplies       968.00 KiB
sample_training         8.39 MiB
sample_weatherdata      4.55 MiB
student_detail         40.00 KiB
```

Figure 12.3: Show Databases

To verify if the `library_details` collection exists under the `library` database, switch to the library database and execute the command as:

```
show collections
```

The command is executed as shown in Figure 12.4.



```
test> use library
switched to db library
library> show collections
library_details
```

Figure 12.4: View Collections in the Database

12.4 Query Documents Using Python

Users can also write Python code to search for documents from a collection, filter the search results, sort the search results, update a document, or delete a document.

12.4.1 Find a Document

Consider that the user wants to view the first document in the `library_details` collection. Code Snippet 2 lists the code to perform this action:

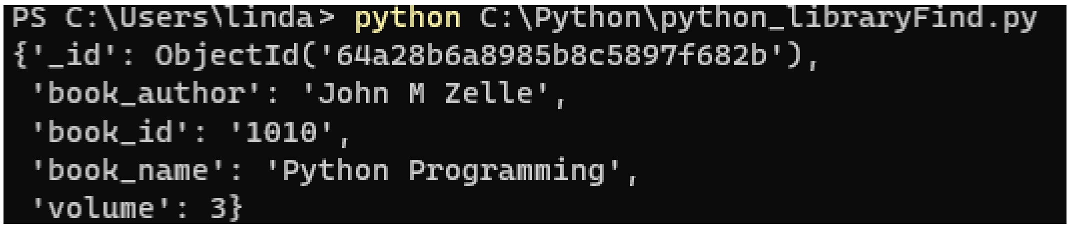
Code Snippet 2:

```
from pymongo import MongoClient
import pprint
client = MongoClient()
db = client.library
library_details = db.library_details
pprint.pprint(library_details.find_one())
```

This code must be saved in a Python file. For example, let us enter this code in a file named as `python_libraryFind.py` and save it to the `C:\Python` folder. To execute the code in the `python_libraryFind.py` folder, at the command prompt, run the command as:

```
python C:\Python\python_libraryFind.py
```

The command executes as shown in Figure 12.5.



```
PS C:\Users\linda> python C:\Python\python_libraryFind.py
{'_id': ObjectId('64a28b6a8985b8c5897f682b'),
 'book_author': 'John M Zelle',
 'book_id': '1010',
 'book_name': 'Python Programming',
 'volume': 3}
```

Figure 12.5: View a Document in the Collection

This command has fetched the first document in the `library_details` collection.

12.4.2 Find All Documents

Consider that the user wants to view all the documents in the `library_details` collection. Code Snippet 3 lists the code to perform this action:

Code Snippet 3:

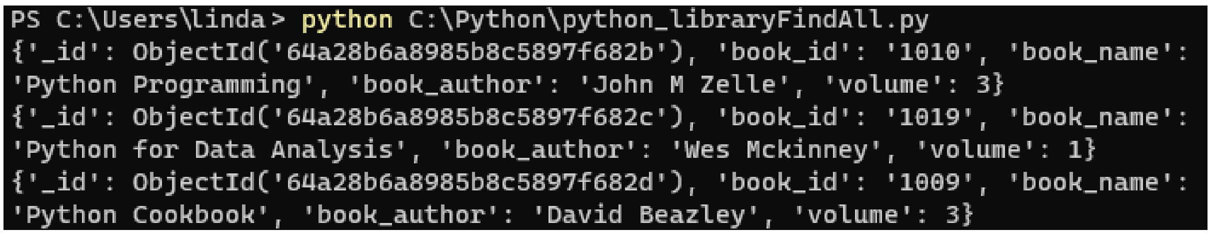
```
from pymongo import MongoClient
import pprint
client = MongoClient()
db = client.library
library_details = db.library_details
for lib in library_details.find({}):
print(lib)
```

The `for` statement in this code will iterate through all the documents and print those on the screen.

This code must be saved in a Python file. For example, let us enter the code in a file named as `python_libraryFindAll.py` and save it to the `C:\Python` folder. To execute the code in the `python_libraryFindAll.py` file, at the command prompt, run the command as:

```
python C:\Python\python_libraryFindAll.py
```

The command executes as shown in Figure 12.6.



```
PS C:\Users\linda> python C:\Python\python_libraryFindAll.py
{'_id': ObjectId('64a28b6a8985b8c5897f682b'), 'book_id': '1010', 'book_name':
'Python Programming', 'book_author': 'John M Zelle', 'volume': 3}
{'_id': ObjectId('64a28b6a8985b8c5897f682c'), 'book_id': '1019', 'book_name':
'Python for Data Analysis', 'book_author': 'Wes Mckinney', 'volume': 1}
{'_id': ObjectId('64a28b6a8985b8c5897f682d'), 'book_id': '1009', 'book_name':
'Python Cookbook', 'book_author': 'David Beazley', 'volume': 3}
```

Figure 12.6: View All Documents in the Collection

This code displays all the documents in the `library_details` collection.

12.4.3 Filter the Result

Consider that the user wants to filter the output of a `find` method to view book details of a specific `book_author`. `PyMongo` provides a query object which can be used to specify the criteria to filter the output of a search. The query object is passed as the first argument to the `find` method. Code Snippet 4 lists the code to perform this action:

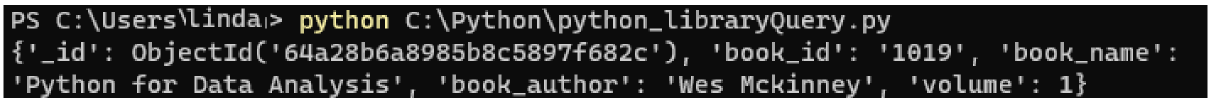
Code Snippet 4:

```
from pymongo import MongoClient
import pprint
client = MongoClient()
db = client.library
library_details = db.library_details
for lib in library_details.find({"book_author":
"Wes Mckinney"}):
print(lib)
```

This code must be saved in a Python file. For example, let us enter the code in a file named as `python_libraryQuery.py` and save it to the `C:\Python` folder. To execute the code in the `python_libraryQuery.py` folder, at the command prompt, run the command as:

```
python C:\Python\python_libraryQuery.py
```

The command executes as shown in Figure 12.7.



```
PS C:\Users\linda> python C:\Python\python_libraryQuery.py
{'_id': ObjectId('64a28b6a8985b8c5897f682c'), 'book_id': '1019', 'book_name':
'Python for Data Analysis', 'book_author': 'Wes Mckinney', 'volume': 1}
```

Figure 12.7: Filter the Result

12.4.4 Filter Query Results Using Comparison Operators

Consider that the user wants to fetch all the documents in the `library_details` collection where the `volume` field is greater than 1. This can be achieved using the '`$gt`' comparison operator. Code Snippet 5 lists the code to perform this action:

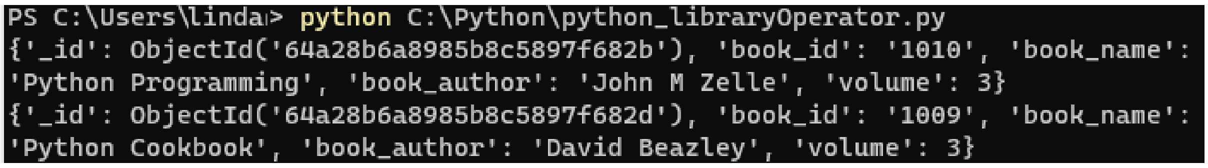
Code Snippet 5:

```
from pymongo import MongoClient
import pprint
client = MongoClient()
db = client.library
library_details = db.library_details
for lib in
library_details.find({"volume":{"$gt":1}}):
print(lib)
```

This code must be saved in a Python file. For example, let us enter the code in a file named as `python_libraryOperator.py` and save it to the `C:\Python` folder. To execute the code in the `python_libraryOperator.py` file, at the command prompt, run the command as:

```
python C:\Python\python_libraryOperator.py
```

The command executes as shown in Figure 12.8.



```
PS C:\Users\linda> python C:\Python\python_libraryOperator.py
{'_id': ObjectId('64a28b6a8985b8c5897f682b'), 'book_id': '1010', 'book_name':
'Python Programming', 'book_author': 'John M Zelle', 'volume': 3}
{'_id': ObjectId('64a28b6a8985b8c5897f682d'), 'book_id': '1009', 'book_name':
'Python Cookbook', 'book_author': 'David Beazley', 'volume': 3}
```

Figure 12.8: Comparison Operator

Note that this code has fetched only those documents where value in the `volume` field is greater than 1.

12.4.5 Filter Query Results Using Boolean Operators

Consider that the user wants to fetch all the documents in the `library_details` collection where the `volume` field is greater than `1` and the book_name is 'Python Programming'. This can be achieved using the and Boolean operator. Code Snippet 6 lists the code to perform this action:

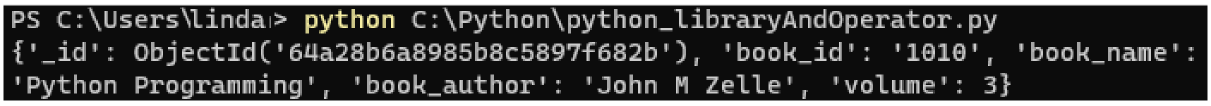
Code Snippet 6:

```
client = MongoClient()
db = client.library
library_details = db.library_details
for lib in
library_details.find({"$and":[{"book_name":
"Python Programming"},{"volume":{"$gt":1}}]}):
 print(lib)
```

This code must be saved in a Python file. For example, let us enter the code in a file named as python_libraryAndOperator.py and save it to the C:\Python folder. To execute the code in the python_libraryAndOperator.py file, at the command prompt, run the command as:

```
python C:\Python\python_libraryAndOperator.py
```

The command executes as shown in Figure 12.9.



```
PS C:\Users\linda> python C:\Python\python_libraryAndOperator.py
{'_id': ObjectId('64a28b6a8985b8c5897f682b'), 'book_id': '1010', 'book_name':
'Python Programming', 'book_author': 'John M Zelle', 'volume': 3}
```

Figure 12.9: Boolean Operator

Note that this code has fetched only those documents from the library_details collection where the volume field is greater than 1 and the book_name is 'Python Programming'.

12.4.6 Sort the Query Results

Consider that the user wants to sort the documents fetched from a find method in descending order on the book_name field. This can be done using the sort method. This method takes two parameters as input. The first parameter specifies the field on which the sort must be performed and the second parameter specifies the sort order (1 for ascending order and -1 for descending order). The default is ascending order.

Code Snippet 7 lists the code to perform this action:

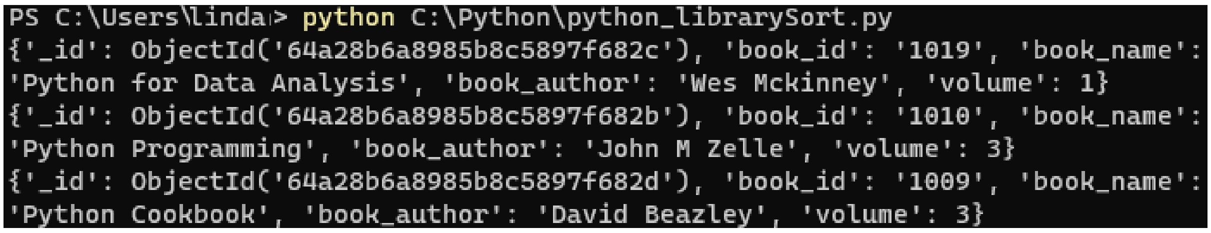
Code Snippet 7:

```
from pymongo import MongoClient
import pprint
client = MongoClient()
db = client.library
library_details = db.library_details
for lib in
library_details.find().sort("book_name", -1):
 print(lib)
```

This code must be saved in a Python file. For example, let us enter the code in a file named as `python_librarySort.py` and save it to the `C:\Python` folder. To execute the code in the `python_librarySort.py` file, at the command prompt, run the command as:

```
python C:\Python\python_librarySort.py
```

The command executes as shown in Figure 12.10.



```
PS C:\Users\linda> python C:\Python\python_librarySort.py
{'_id': ObjectId('64a28b6a8985b8c5897f682c'), 'book_id': '1019', 'book_name':
'Python for Data Analysis', 'book_author': 'Wes Mckinney', 'volume': 1}
{'_id': ObjectId('64a28b6a8985b8c5897f682b'), 'book_id': '1010', 'book_name':
'Python Programming', 'book_author': 'John M Zelle', 'volume': 3}
{'_id': ObjectId('64a28b6a8985b8c5897f682d'), 'book_id': '1009', 'book_name':
'Python Cookbook', 'book_author': 'David Beazley', 'volume': 3}
```

Figure 12.10: Sort Documents

Note that the documents fetched are sorted in descending order of the `book_name` field.

12.4.7 Update a Document

Consider that the user wants to update the value in the `volume` field as `4` for the document where the `book_name` is `Python  Programming`. This can be done using the `update_one` method. This method takes two parameters as input. The first parameter specifies the document that must be updated, and the second parameter specifies the new values.

Code Snippet 8 lists the code to perform this action:

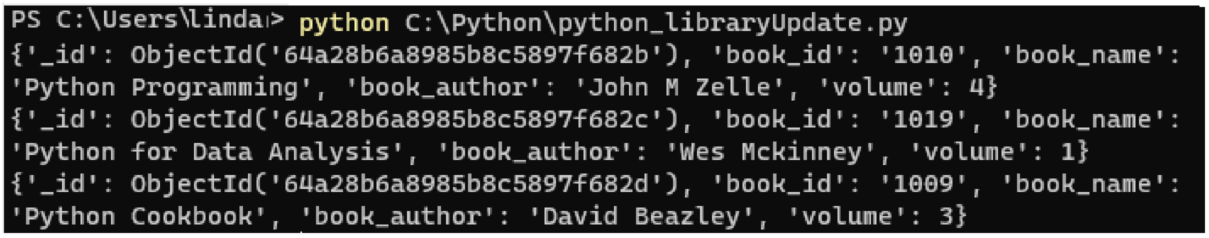
Code Snippet 8:

```
from pymongo import MongoClient
import pprint
client = MongoClient()
db = client.library
library_details = db.library_details
myquery = { "book_name": "Python Programming" }
updatevalue = { "$set":{"volume":4} }
library_details.update_one(myquery, updatevalue)
for lib in library_details.find():
print(lib)
```

This code must be saved in a Python file. For example, let us enter this code in a file named as `python_libraryUpdate.py` and save it to the `C:\Python` folder. To execute the code in the `python_libraryUpdate.py` file, at the command prompt, run the command as:

```
python C:\Python\python_libraryUpdate.py
```

The command executes as shown in Figure 12.11.



```
PS C:\Users\linda> python C:\Python\python_libraryUpdate.py
{'_id': ObjectId('64a28b6a8985b8c5897f682b'), 'book_id': '1010', 'book_name':
'Python Programming', 'book_author': 'John M Zelle', 'volume': 4}
{'_id': ObjectId('64a28b6a8985b8c5897f682c'), 'book_id': '1019', 'book_name':
'Python for Data Analysis', 'book_author': 'Wes Mckinney', 'volume': 1}
{'_id': ObjectId('64a28b6a8985b8c5897f682d'), 'book_id': '1009', 'book_name':
'Python Cookbook', 'book_author': 'David Beazley', 'volume': 3}
```

Figure 12.11: Update a Document

Note that the value of the `volume` field has changed to `4` for the document where the `book_name` is `Python Programming`.

12.4.8 Delete a Document

Consider that the user wants to delete a document in a collection where the `author_name` is '`David Beazley`'. The `delete_one` method in MongoDB can be used to delete a document. The query object is passed as the first argument to the `delete_one` method.

Code Snippet 9 lists the code to perform this action:

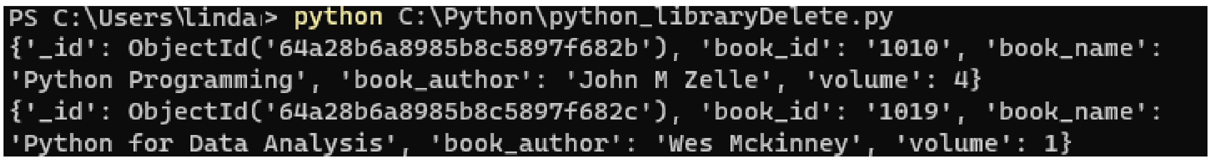
Code Snippet 9:

```
from pymongo import MongoClient
import pprint
client = MongoClient()
db = client. library
library_details = db.library_details
myquery = {"book_author":"David Beazley"}
library_details.delete_one(myquery)
for lib in library_details.find():
print(lib)
```

This code must be saved in a Python file. For example, let us enter this code in a file named as `python_libraryDelete.py` and save it to the `C:\Python` folder. To execute the code in the `python_libraryDelete.py` file, at the command prompt, run the command as:

```
python C:\Python\python_libraryDelete.py
```

The command executes as shown in Figure 12.12.



```
PS C:\Users\linda> python C:\Python\python_libraryDelete.py
{'_id': ObjectId('64a28b6a8985b8c5897f682b'), 'book_id': '1010', 'book_name':
'Python Programming', 'book_author': 'John M Zelle', 'volume': 4}
{'_id': ObjectId('64a28b6a8985b8c5897f682c'), 'book_id': '1019', 'book_name':
'Python for Data Analysis', 'book_author': 'Wes Mckinney', 'volume': 1}
```

Figure 12.12: Delete a Document

Note that the document where the `author_name` is `David Beazley` is deleted from the collection.

The `library_details.delete_many({})` command can be used to delete all the documents from the `library_details` collection.

12.5 Summary

- The combination of Python, a versatile programming language, and MongoDB, an extremely scalable and reliable database, forms a robust database application.
- `PyMongo` is MongoDB's official Python driver that helps to create a connection between Python and MongoDB.
- `PyMongo` can be installed using the command `pip install PyMongo`.
- After establishing the connection between Python and MongoDB, databases and collections can be created in MongoDB using Python.
- The collections in the MongoDB database can be queried to retrieve either a document or all the documents in the collection.
- The results of a query can be filtered using the comparison or Boolean operators.
- The results of a query can be sorted using the `sort` method.
- The documents in a MongoDB collection can be updated or deleted using Python by specifying conditions.

Test Your Knowledge

1. Which driver should you install to interact with the MongoDB database through Python?

   a. `PythonMongo`
   b. `PyMongo`
   c. `MongoPy`
   d. `Phmongo`

2. Which of the following class of the `PyMongo` driver is used to connect Python with MongoDB?

   a. `ConnectMongo`
   b. `MongoConnection`
   c. `PythonClient`
   d. `MongoClient`

3. Which of the following code will allow you to create a `PyMongo` client?

   a. `mongod = MongoClient()`
   b. `server = ClienMongo()`
   c. `client = MongoClient()`
   d. `client = connectMongo()`

4. Consider that you have connected Python with MongoDB using the PyMongo driver. Which of the following code allows you to create a database named `employee`?

   a. `db = client.employee`
   b. `client = db.employee`
   c. `db = MongoClient.employee`
   d. `client= db.client.employee`

5. Which of the following methods are used to return all documents from a collection?

   a. `find()`
   b. `findAll()`
   c. `find({ })`
   d. `findMany()`

Answers to Test Your Knowledge

1	b
2	d
3	c
4	a
5	a, c

Try it Yourself

1. Install the `PyMongo` driver to connect Python with MongoDB.
2. Create a Python file `connectMongo.py` and perform the given tasks:
    a. Import a `MongoClient` class from `PyMongo` driver.
    b. Create a `MongoClient` instance.
    c. Create a database named `Customer_purchase` and a collection named `Purchase_detail`. Three documents in the collection are:

```
[
   {
     Cust_id:94608
     Product_id:112
     Cust_name:Adam Richard
     Purchase_product:Laptop
     Discount:0.05
    },
   {

     Cust_id:94609
     Product_id:118
     Cust_name:Michael Faraday
     Purchase_product:Tablet
     Discount:0.1
     },
   {

     Cust_id:94602
     Product_id:103
     Cust_name:Richard David
     Purchase_product:Smartwatch
     Discount:0.05
   }
]
```

    d. Write Python code to insert the given documents into the `Purchase_detail` collection.
3. Execute the Python file `connectMongo.py`.
4. Connect to Mongo shell to view the `Customer_purchase` database and the `Purchase_detail` collection.
5. To query the documents, write Python code for each of the given tasks and execute the Python codes to view the results.
    a. Find the documents where the `Discount` value is equal to `0.5`.
    b. Find all the documents in the collection `Purchase_detail`.
    c. Update the `Purchase_product` of ‘`Richard David`’ as ‘`Smartphone`’.

Appendix

Sr. No.	Case Studies
1.	In a supermarket, the daily sales of products are stored in a MongoDB database named `product_inventory`. At the end of each day, the manager of the supermarket must analyze sales of each product based on the data stored in the `product_inventory` database. a. Create the `product_inventory` database with a collection named `product_sales`. b. Create a user named `manager` for the `product_inventory` database and grant the `readwrite` role to `manager` on the `product_inventory` database. c. Authenticate `manager` on the `product_inventory` database and insert the given six documents into the `product_sales` collection.

```
[
    {
      cust_id:1001
      cust_name: "John",
      cust_city:"Atlanta "
      current_purchase:"Baby Food",
      quantity:3,
      unit_price:300

last_purchase:["dairy","grocery","snacks"]
  }
{
      cust_id:1006
      cust_name: "Alexander",
      cust_city:"Boston"
      current_purchase:"diary",
      quantity:4,
      unit_price:250,
      last_purchase:["care_products”,"snacks"]
  }
{
      cust_id:1003
      cust_name: "Gabriel",
      cust_city:"Los Angeles"
      current_purchase:"care_products",
      quantity:7,
      unit_price:500,
      last_purchase:["dairy","dried goods”]
  }
{
      cust_id:1004
      cust_name: "Williams",
      cust_city:"Texas"
```

```
            current_purchase:"snacks",
            quantity:2,
            unit_price:250,
            last_purchase:["grocery","diary"]

         }
      {
            cust_id:1007
            cust_name: "Nicholas",
            cust_city:"Boston"
            current_purchase:"Baby Food",
            quantity:5,
            unit_price:150,
            last_purchase:[ "Fruits","snacks"]
         }
      {
            cust_id:1008
            cust_name: "Richard",
            cust_city:"Austin"
            current_purchase:"diary",
            quantity:3,
            unit_price:250,
            last_purchase:[ "Fruits", "vegetables"]

         }
      ]
```

d. Create an index for `product_sales` collection on the `cust_id` field in ascending order.
e. Fetch only the documents where the customer belongs to the `Texas` city.
f. Fetch the city details of all the customers who have currently purchased only dairy products.
g. Add a new field named `Date_of_purchase` which shows the current date to all the documents.
h. Fetch details of all the customers who purchased `grocery` in their last transaction.
i. Exclude the `_id` field and display all the documents of the `product_sales` collection which include only the `cust_name`, `cust_city`, `current_purchase`, and `quantity` fields.
j. Retrieve the distinct `current_purchase` values from the `product_sales` collection.
k. Calculate the total quantity of sales per `current_purchase` item and return only the `current_purchase` item with a total quantity greater than or equal to 5.

	l. Calculate the `total price` (`unit_price*quantity`) of sales per `current_purchase` item for all the documents in the `product_sales` collection. m. Count the number of customers who have purchased '`snacks`' as one of the items in their last transactions.
2.	Consider that the supermarket has three branches all over the country and maintains the data of all three branches on a single server. When the data is accessed from different branches, in order to ensure the high availability of data, create a replica set(P-S-S) in MongoDB where a primary server has two secondary members. a. In the primary server, create a database with the name `product_inventory` that contains a collection named `product_sales`. Insert the six documents into `product_sales` collection and check whether the `product_inventory` database is replicated in the secondary servers. b. In the primary server, create an index for the field `cust_id`. Start a session and start a transaction to update the `current_purchase` field of the customer with `cust_id` as `1008` to `care_products`. Commit this transaction and check whether the update operation executed in this transaction is visible in the secondary servers.
3.	Manager of a supermarket wants to analyze and visualize the data in the `product_sales` collection by generating reports using Business Intelligence (BI) tools. To do this, use MongoDB compass to connect the `product_inventory` database to Microsoft Excel using Open Data Base Connectivity (ODBC). Import data from the `product_sales` collection to Microsoft Excel.