MongoDB

Assignment Questions

Assignment

Q1. What is MongoDB? Explain non-relational databases in short. In which scenarios it is preferred to use MongoDB over SQL databases?

Q2. State and Explain the features of MongoDB.

Q3. Write a code to connect MongoDB to Python. Also, create a database and a collection in MongoDB.

Q4. Using the database and the collection created in question number 3, write a code to insert one record, and insert many records. Use the find() and find_one() methods to print the inserted record.

Q5. Explain how you can use the find() method to query the MongoDB database. Write a simple code to demonstrate this.

Q6. Explain the sort() method. Give an example to demonstrate sorting in MongoDB. Q7.

Explain why delete_one(), delete_many(), and drop() is used.

Q.1> What is MongoDB? Explain non-relational databases in short. In which scenarios it is preferred to use MongoDB over SQL databases?

ANSWER

MongoDB is a popular document-oriented NoSQL database that stores data in flexible, JSON-like documents, called BSON. It is designed to provide high performance, scalability, and flexibility for modern applications that require dynamic, complex data structures.

Non-relational databases, also known as NoSQL databases, are databases that do not use the traditional table-based relational database model used in SQL databases. Instead, they use flexible data models that can handle semi-structured and unstructured data, making them ideal for handling big data and real-time data processing.

MongoDB is often preferred over SQL databases in situations where the application requires a high degree of scalability and flexibility, real-time processing of large volumes of unstructured data, and the ability to handle complex data models. It is also well-suited for use in modern web and mobile applications that require fast development cycles and agile data structures.

Q.2> State and Explain the features of MongoDB.

ANSWER

MongoDB is a popular document-oriented NoSQL database that provides a wide range of features that make it a popular choice for modern application development.

Document-based data model: MongoDB stores data in documents, which are similar to JSON objects. This allows for a flexible and scalable data model that can easily handle complex data structures.
Scalability: MongoDB is highly scalable and can handle large amounts of data and traffic with ease. It supports sharding, which allows you to distribute data across multiple servers for improved performance and availability.
High availability: MongoDB provides automatic failover and replica sets to ensure that your data is always available, even in the event of hardware or network failures.
Flexible schema: MongoDB has a flexible schema, which means that you can easily modify your data model as your application evolves.
Rich query language: MongoDB provides a powerful and flexible query language that allows you to perform complex queries on your data.
Aggregation framework: MongoDB provides a built-in aggregation framework that allows you to perform complex data processing tasks such as grouping, filtering, and transforming data.
Geospatial support: MongoDB provides built-in support for geospatial data, allowing you to easily perform queries based on location.

.3> Write a code to connect MongoDB to Python. Also, create a database and a collection in MongoDB.

ANSWER

In [1]:

```
import pymongo


client =
pymongo.MongoClient("mongodb+srv://Sachchida:pwskills@sachida.mongodb.net/?retryWrites=true&w=majority")
db = client.test
```

In [3]:

```
pip install pymongo
```

```
Requirement already satisfied: pymongo in /opt/conda/lib/python3.10/site-packages (4.3.3)
Requirement already satisfied: dnspython<3.0.0,>=1.16.0 in /opt/conda/lib/python3.10/site-packages (from pymongo) (2.3.0)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:

```
client =
pymongo.MongoClient("mongodb+srv://ankitdhattarwal:pwskills@ankit.mxc30sz.mongodb.net/?retryWrites
=true&w=majority")
```

In [5]:

```
db = client['pwskills']
```

In [6]:

```
coll_pwskills = db["my_record"]
```

In [ ]:

```
coll_pwskills.insert_one(data)
```

In [ ]:

```
data1 = { "mail_id" : "sachchida2u4utuesday@gmail.com" ,
        "phone" : 9555176166,
        "address" : "varanasi"

}
```

In [ ]:

```
 coll_pwskills.insert_one(data1)
```

Q.4> Using the database and the collection created in question number 3, write a code to insert one record, and insert many records. Use the find() and find_one() methods to print the inserted record.

ANSWER

In [7]:

```
client = pymongo.MongoClient("mongodb+srv://sachchida:pwskill.mongodb.net/?retryWrites=true&w=majority")
```

In [ ]:

```
db = client['pwskills']
```

In [ ]:

```
coll_pwskills = db["my_record"]
```

In [ ]:

```
coll_pwskills.insert_one(data)
```

In [ ]:

```
data1 = { "mail_id" : "sachchida2u4utuesday@gmail.com" ,
        "phone" :9555176166,
        "address" : "Varanasir"


}
```

In [ ]:

```
coll_pwskills.insert_one(data1)
```

In [ ]:

```
coll_pwskills.find_one()
```

In [ ]:

```
for i in coll_pwskills.find():
   print(i)
```

In [ ]:

```
for i in coll_pwskills.find({'name' :'sachchida'}):
    print(i)
```

Q.5> Explain how you can use the find() method to query the MongoDB database. Write a simple code to demonstrate this.

ANSWER

The find() method is used to query a MongoDB database and retrieve documents that match a specific criteria. It takes one or more arguments that define the search criteria, and returns a cursor object that can

be used to iterate over the results.

The find() method supports a wide variety of search criteria, including exact matches, ranges, regular expressions, and more.

In [ ]:

```
import pymongo



client = pymongo.MongoClient("mongodb://localhost:27017/")



mydb = client["mydatabase"]

mycol = mydb["customers"]



for record in mycol.find():

    print(record)



for record in mycol.find({ "name": "John" }):

    print(record)



for record in mycol.find({ "age": 25 }):

    print(record)



for record in mycol.find({ "age": { "$gte": 18 } }):

    print(record)
```

```
for record in mycol.find({ "age": { "$gt": 18, "$lt": 30 } }):
    print(record)

for record in mycol.find({ "name": { "$regex": "^J" } }):
    print(record)
```

Q.6> Explain the sort() method. Give an example to demonstrate sorting in MongoDB.

ANSWER

The sort() method in MongoDB is used to sort the results of a query in ascending or descending order based on one or more fields. The sort() method takes one or more arguments that specify the sorting criteria, and returns a cursor object that can be used to iterate over the sorted results.

By default, the sort() method sorts the results in ascending order based on the specified field(s). To sort the results in descending order, you can pass the value -1 as the sorting criteria for the field(s).

In [ ]:

```
import pymongo

client = pymongo.MongoClient("mongodb://localhost:27017/")

mydb = client["mydatabase"]

mycol = mydb["customers"]

print("Ascending sort by name:")

for record in mycol.find().sort("name"):
    print(record)
```

```
print("Descending sort by age:")

for record in mycol.find().sort("age", -1):

    print(record)



print("Sort by name and age:")

for record in mycol.find().sort([("name", 1), ("age", -1)]):

    print(record)
```

Q.7> Explain why delete_one(), delete_many(), and drop() is used.

ANSWER

delete_one(): This method deletes the first document that matches the specified filter criteria. If multiple documents match the filter, only the first one is deleted.
delete_many(): This method deletes all documents that match the specified filter criteria.

drop(): This method deletes an entire collection and all of its documents.

These methods are used for different purposes:
delete_one(): This method is useful when you want to delete a single document from a collection based on a specific filter criteria. For example, you might use delete_one() to remove a specific order from an orders collection.
delete_many(): This method is useful when you want to delete multiple documents from a collection based on a specific filter criteria. For example, you might use delete_many() to remove all orders that have a status of "cancelled".

drop(): This method is useful when you want to delete an entire collection and all of its documents. For example, you might use drop() to remove a collection that is no longer needed or that contains outdated data.

In [ ]:

Data Science Masters