Overview

Wednesday, August 7, 2024 10:01 PM

1. **MongoDB in Data Science**

2. **PyMongo**

3. **Project 1 - Data Analysis**

4. **Project 2 - Machine Learning**

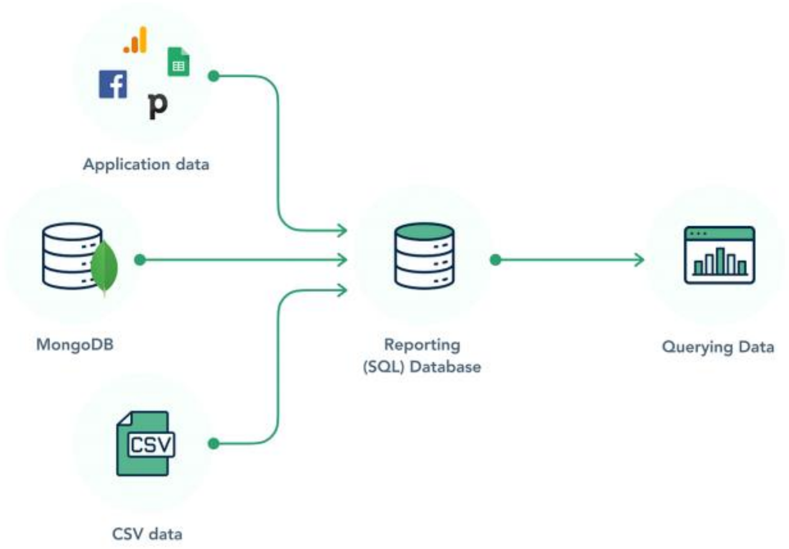
5. **Next Steps**

MongoDB in Data Science

Wednesday, August 7, 2024 10:23 PM

MongoDB is a popular NoSQL database that can be highly effective in data science workflows due to its flexibility, scalability, and ability to handle large volumes of unstructured data.

Typical usage in Data Science workflows:



1. **Data Storage:**
    - ○ MongoDB stores data in a flexible, JSON-like format (BSON), which allows for a dynamic schema. This is ideal for storing diverse datasets without the need for a predefined structure.
    - ○ It can easily scale out by distributing data across multiple servers.
    - ○ MongoDB provides powerful indexing capabilities that enable efficient querying, which is crucial for data retrieval and analysis.

2. **Data Ingestion:**
    - ○ Capable of handling real-time data ingestion from various sources such as IoT devices, social media feeds, and logs. This is useful for real-time analytics and monitoring.

- MongoDB can be integrated into ETL (Extract, Transform, Load) workflows to transform and load data from various sources into a unified format for analysis.

3. **Data Analysis:**
    - MongoDB's aggregation framework allows for powerful data transformation and processing. It can be used to filter, group, and manipulate data directly within the database.
    - Also helps in cleaning data by removing duplicates, filling missing values, and standardizing data formats.
    - MongoDB can be integrated with Python and statistical analysis tools like Pandas, to perform in-depth analysis.

4. **Data Visualization:**
    - MongoDB can be integrated with various data visualization tools like Tableau, Power BI, and MongoDB Charts.
    - These tools help in creating interactive dashboards and visualizing the results of data analysis.
    - Using frameworks like Flask or Django, data scientists can build custom dashboards that fetch data directly from MongoDB and display it in real-time.

5. **Machine Learning Integration:**
    - Data can be seamlessly pulled from MongoDB for training, testing, and validating models.
    - MongoDB integrates well with machine learning libraries and frameworks such as TensorFlow, PyTorch, and scikit-learn.
    - This allows data scientists to leverage MongoDB for storing training data, model parameters, and results.

6. **Collaboration:**
    - By storing different versions of datasets, MongoDB allows for tracking changes and collaborating on data analysis projects.

PyMongo

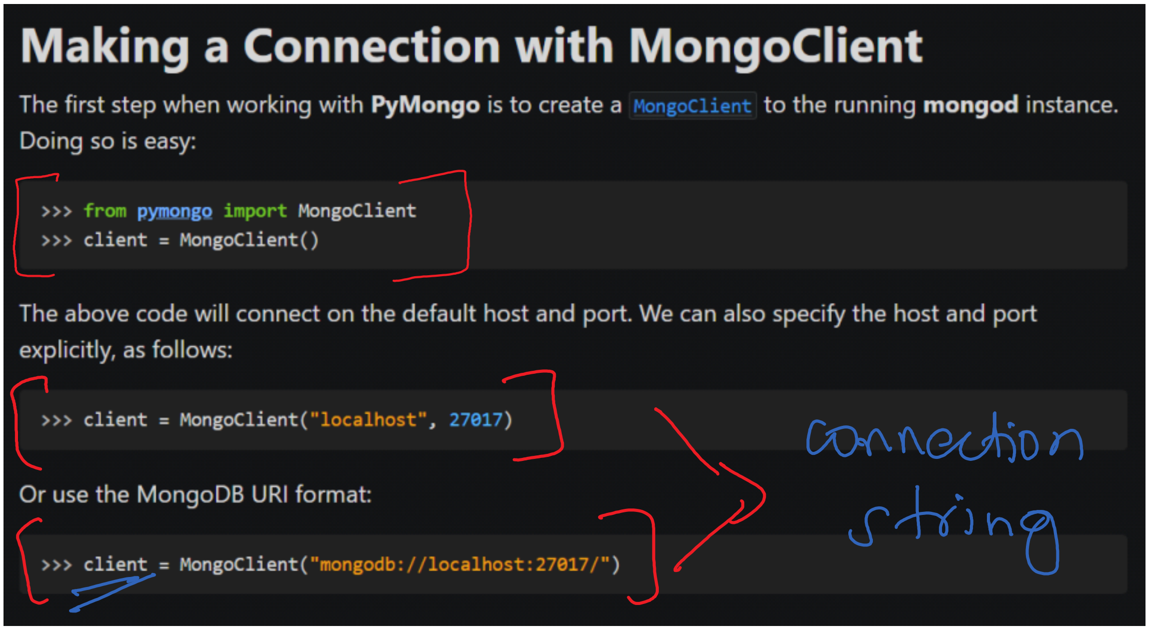
Wednesday, August 7, 2024 10:38 PM

PyMongo is the official Python driver for MongoDB. It provides tools to connect to a MongoDB instance, perform CRUD (Create, Read, Update, Delete) operations, and handle more advanced database interactions such as aggregations and indexing.

Core Components of PyMongo:

1. MongoClient:

- The MongoClient class is the starting point for all interactions with the database.
- It allows you to establish a connection to a MongoDB instance or cluster.



Making a Connection with MongoClient

The first step when working with **PyMongo** is to create a `MongoClient` to the running **mongod** instance. Doing so is easy:

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

The above code will connect on the default host and port. We can also specify the host and port explicitly, as follows:

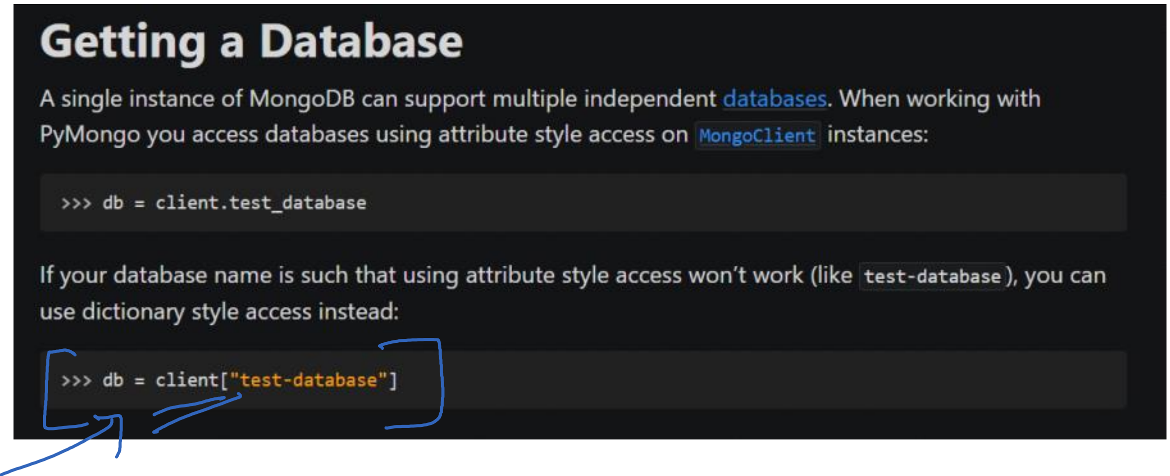
```
>>> client = MongoClient("localhost", 27017)
```

Or use the MongoDB URI format:

```
>>> client = MongoClient("mongodb://localhost:27017/")
```

connection string

2. Database and Collection:



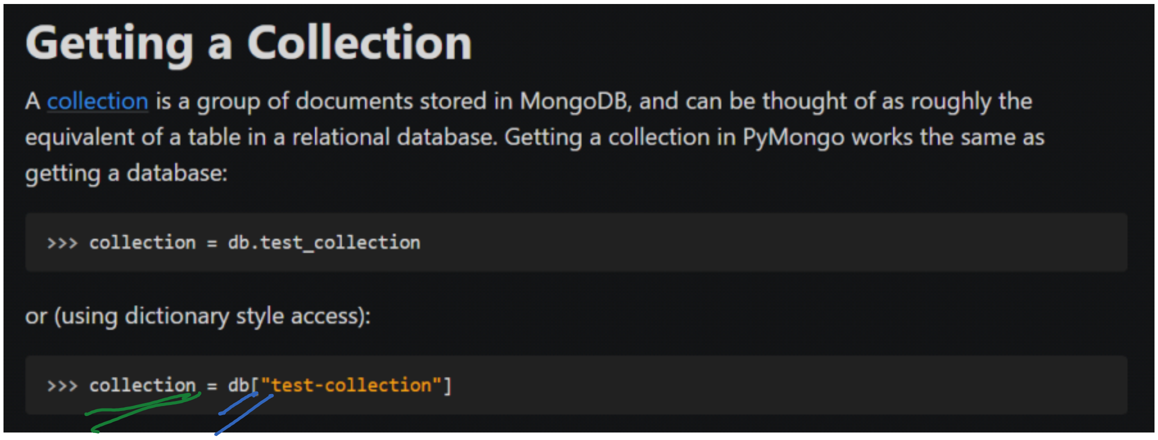
Getting a Database

A single instance of MongoDB can support multiple independent databases. When working with PyMongo you access databases using attribute style access on `MongoClient` instances:

```
>>> db = client.test_database
```

If your database name is such that using attribute style access won't work (like `test-database`), you can use dictionary style access instead:

```
>>> db = client["test-database"]
```



Getting a Collection

A collection is a group of documents stored in MongoDB, and can be thought of as roughly the equivalent of a table in a relational database. Getting a collection in PyMongo works the same as getting a database:

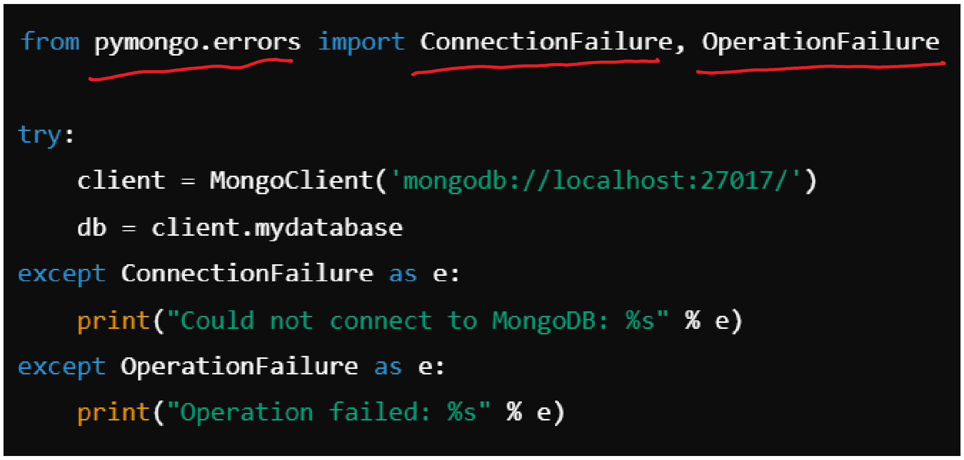
```
>>> collection = db.test_collection
```

or (using dictionary style access):

```
>>> collection = db["test-collection"]
```

3. **Error Handling:**

    ○ PyMongo provides robust error handling mechanisms to catch and handle exceptions such as ConnectionFailure, OperationFailure, and others.



```
from pymongo.errors import ConnectionFailure, OperationFailure

try:
    client = MongoClient('mongodb://localhost:27017/')
    db = client.mydatabase
except ConnectionFailure as e:
    print("Could not connect to MongoDB: %s" % e)
except OperationFailure as e:
    print("Operation failed: %s" % e)
```

Project Workflow

Wednesday, August 7, 2024 11:09 PM

Description:

- Analyze the NYC Taxi Trip Data to uncover insights such as peak hours, popular pickup/drop-off locations, and trip durations.

Dataset:

- **Title:** *New York City Taxi Trip Data*
- **Link:** https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- This dataset includes information about taxi trips in New York City, such as pickup and drop-off dates/times, locations, trip distances, fares, and payment types.

Questions:

1. What are the peak hours for taxi pickups?
2. What is the average trip distance by passenger count?
3. What is the distribution of payment types?
4. What are the top 10 days with the highest number of trips?
5. What is the total fare amount collected by each payment type?
6. What are the average trip distances for each hour of the day?
7. What is the distribution of trips by passenger count?

Workflow:

1. **Instantiate Compass Local Server**

2. **Virtual Environment & IDE Setup**

3. **Install Libraries**

- *pymongo*
- *pyarrow*

4. **Jupyter Notebook Instance**

   - *Import Libraries*
   - *Confirm Python Executable is from Virtual Environment*

5. **Import the Data**

6. **Summarize the Data**

   - *Shape of the Data*
   - *Data Types of Features*
   - *High-level Info on Data*
   - *Detect Missing Values*
   - *Check for Duplicates*

7. **Load Data into MongoDB**

   - *Establish connection to server*
   - *Convert data to JSON*
   - *Insert data into DB*

8. **Query Data to answer all Questions**

   - *aggregate*
   - *group*
   - *project*
   - *sort*
   - *limit*

9. **Visualize the Data using Plots**

   - *Pandas*
   - *Matplotlib*

Project Workflow

Wednesday, August 7, 2024   11:10 PM

Description:

- Train multiple machine learning models on the Car Details dataset and store the results using MongoDB

Dataset:

- **Title:** *Car Details*
- **Link:** click here
- This dataset includes information about various car models and prices.

Workflow:

1. **Instantiate Compass Local Server**
2. **Virtual Environment & IDE Setup**
3. **Install Libraries**
4. **Jupyter Notebook Instance**
5. **Load Data into MongoDB**
6. **Read the Data from MongoDB**
7. **Summarize the Data**
    - *Shape of the Data*
    - *Data Types of Features*
    - *High-level Info on Data*
    - *Detect Missing Values*

- *Check for Duplicates*

8. **Train and Evaluate several Models**

9. **Store the results back into MongoDB**

Next Steps

Saturday, August 10, 2024 10:59 AM

1. **Try this yourself**
2. **Practice with different Datasets**
3. **Explore Advanced MongoDB Features**
4. **Work on implementing Projects**
5. **Enhance Skills with Certification**
    - https://learn.mongodb.com/pages/certification-program