Stephen Cardone

CS 340

Final Project

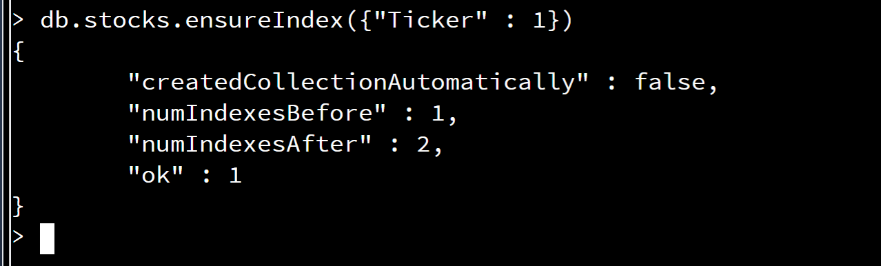
**Collection Management:**

**A.** Importing the market database: To import the database (db), I used the mongoimport command. I added the –db and –collections parameters to set the db name and collection names. I also passed in the stocks.json dataset that contains the json documents that will make up the market.stocks db collection. This is needed as it creates the database that the internal stakeholders will be retrieving data from when they use the API’s I create.



**B.** Indexing: The indexing needs for the market db will vary depending on what we need to do with the data. Internal stakeholders will benefit from having indexes because it will speed up query times in many situations. In this case, we do not have any fields that contain a large amount of text, so a full text index is not necessary. Looking ahead to the goals of this project, we will benefit from the following:

**Ticker index:** Since we will be searching for equities by ticker in this project, I decided to set up an index on the ticket.

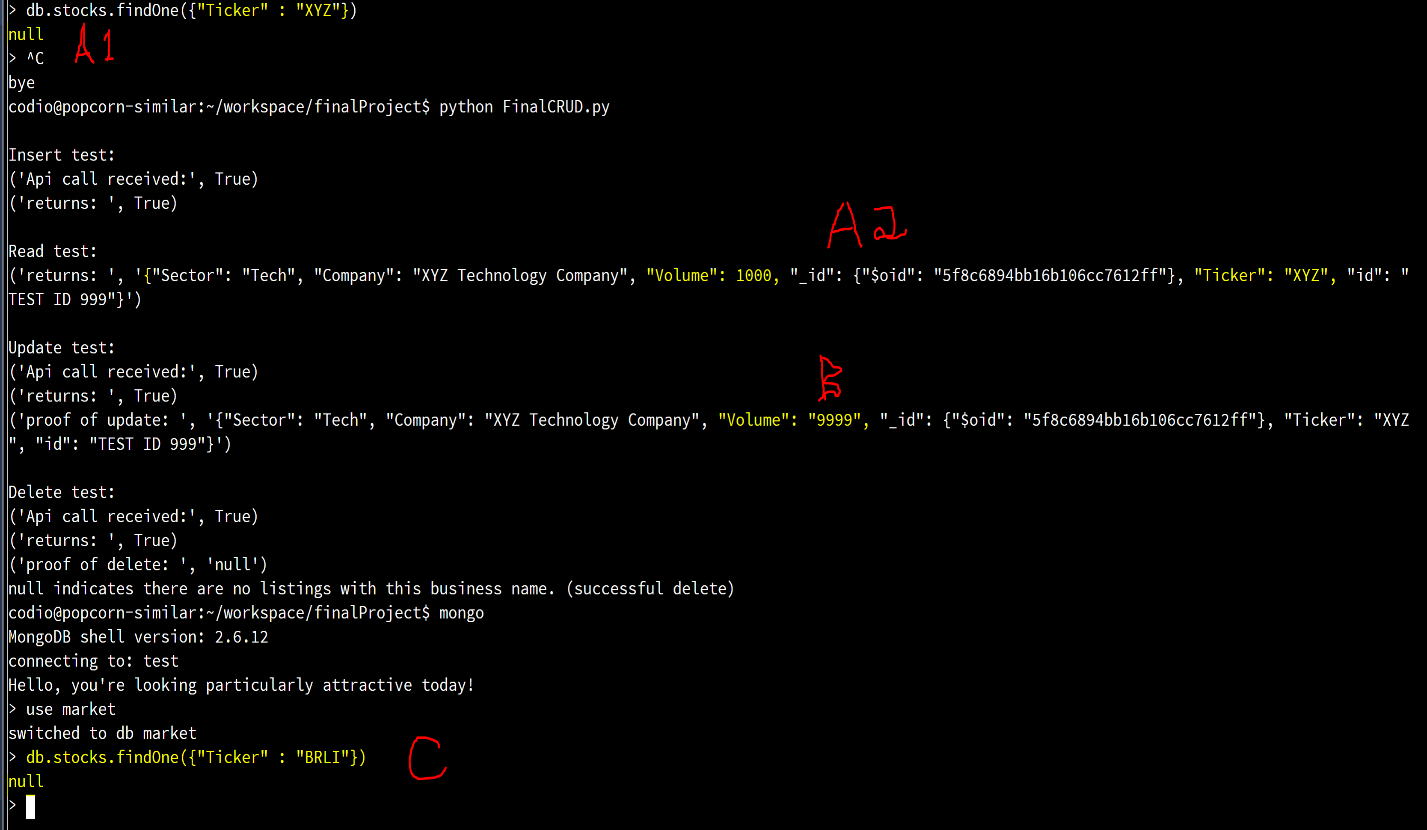


**Document Manipulation:  
Insert, Update, Delete:** I’ve included all the code for the Insert, Update, and Delete functionality in 1 python script: FinalCRUD.py. To inspect the actual code, please look at that script directly which was submitted with this project. In conjunction with the supplied .py script, we can observe the output below.

**A1,A2:** A1 queries the db for ticker “XYZ”. It returns null indicating it does not exist. A2 shows the results from the document creation in FinalCRUD.py. As we can see, a document was created for ticker XYZ with a volume of 1000.

**B:** FinalCRUD.py then executes the update test function which updates volume to “9999”

**C:** FinalCRUD.py then executes the delete test function on Ticker “BRLI”. To confirm this delete, I enter the mongo shell, and query for Ticker:BRLI which returns null confirming the delete.

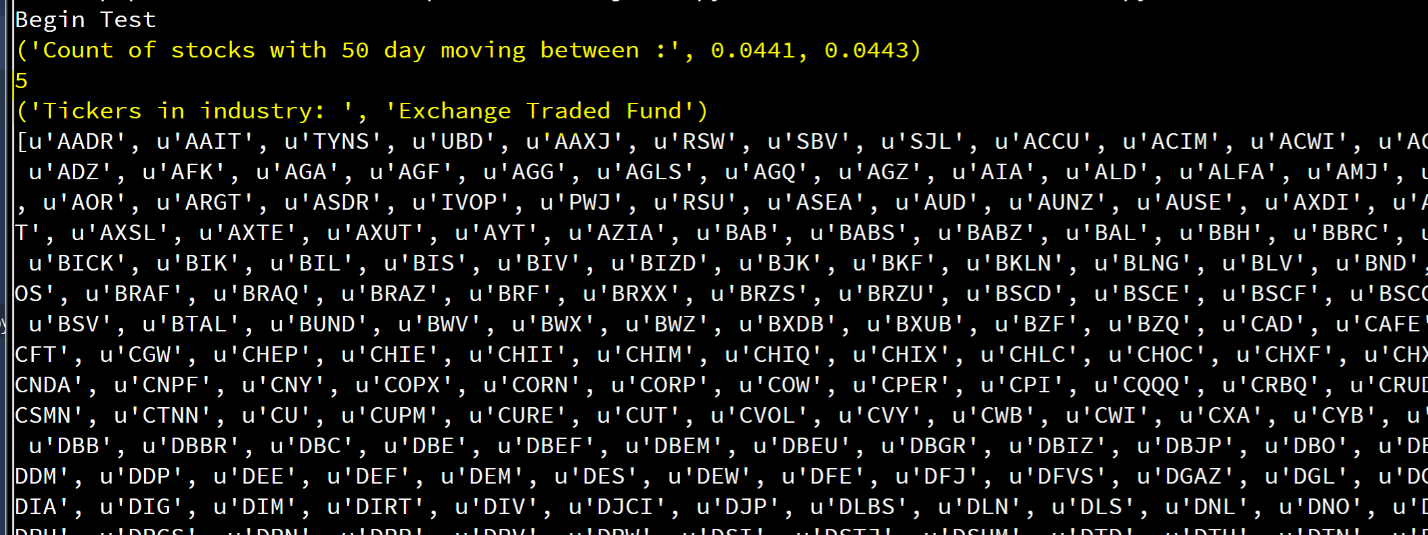


**Document Retrieval:**

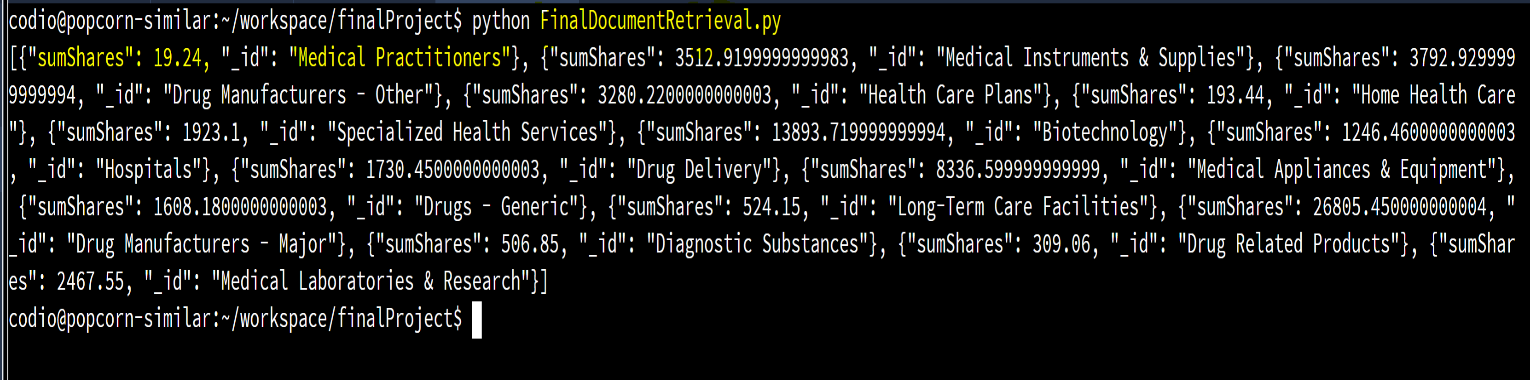
**A, Find Statement Arguments:** All code for this section is found in FinalDocumentRetrival.py

1: The count\_50\_day\_moving\_average\_in\_range(low, high) function accepts 2 params and returns a count of all stocks who have a 50 day moving average within that range. As you can see, in the screenshot below, there are 5 stocks with moving averages between 0.0441 and 0.0443.

2: The list\_tickers\_in\_industry(industry) function accepts 1 parameter for industry and returns the list of all tickets in that industry. In the screenshot below, you can see there are a very large number of tickers in the industry of “Exchange Traded Fund”. Not all are pictured here.

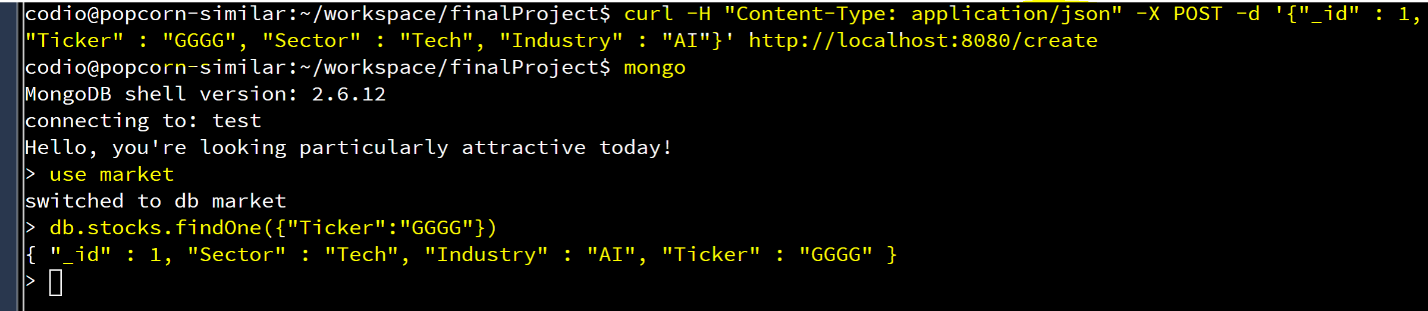


**B, Aggregate Pipeline Statements:** Code for the aggregate pipeline is found in FinalDocumentRetrival.py. Using an Aggregate Pipeline, we can create queries that return data with some manipulation, such as grouping or summation. For example, in the screenshot below, you can see the results for the aggregate\_pipeline(sector) function that accepts a sector argument. It takes that sector, and groups the contents by Industry. Then for each industry it sums the Outstanding Shares. This allows us to supply internal stakeholders with more useful data on outstanding shares in a sector by industry.

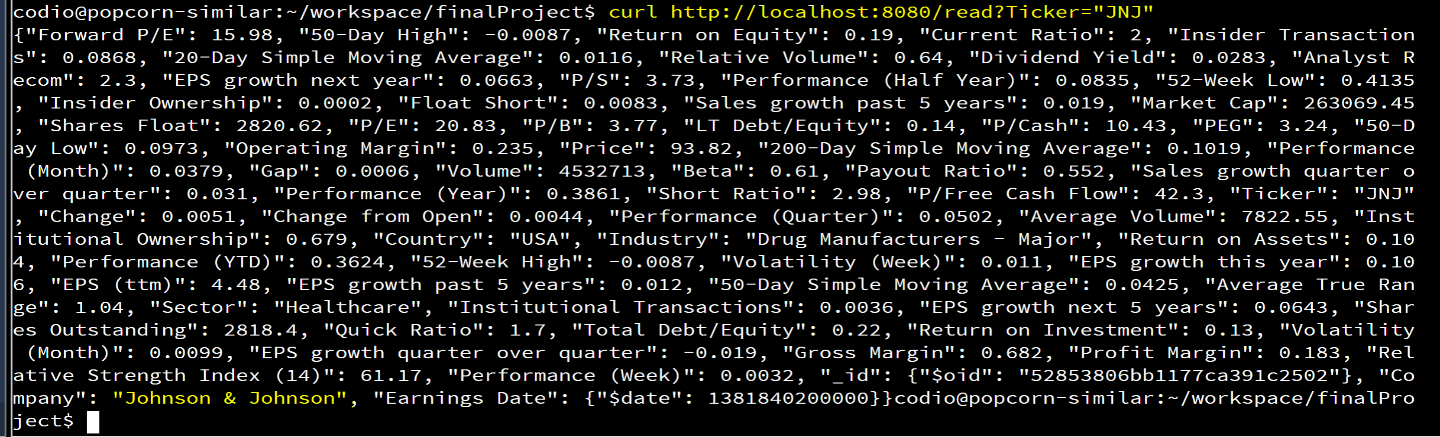


**4: Advanced Programming Project**

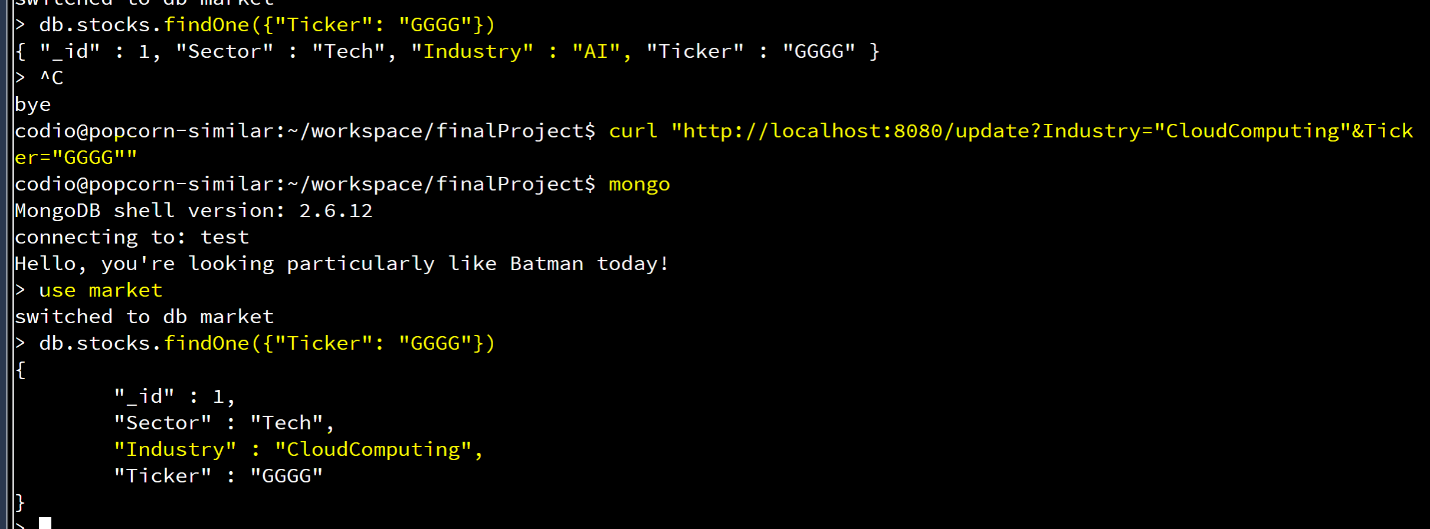
**CRUD: Create:** With curl I requested the /create api and passed in a new stock to be added. In the example below, I created a Stock with the ticker “GGGG”, then entered the mongo shell to confirm its creation.



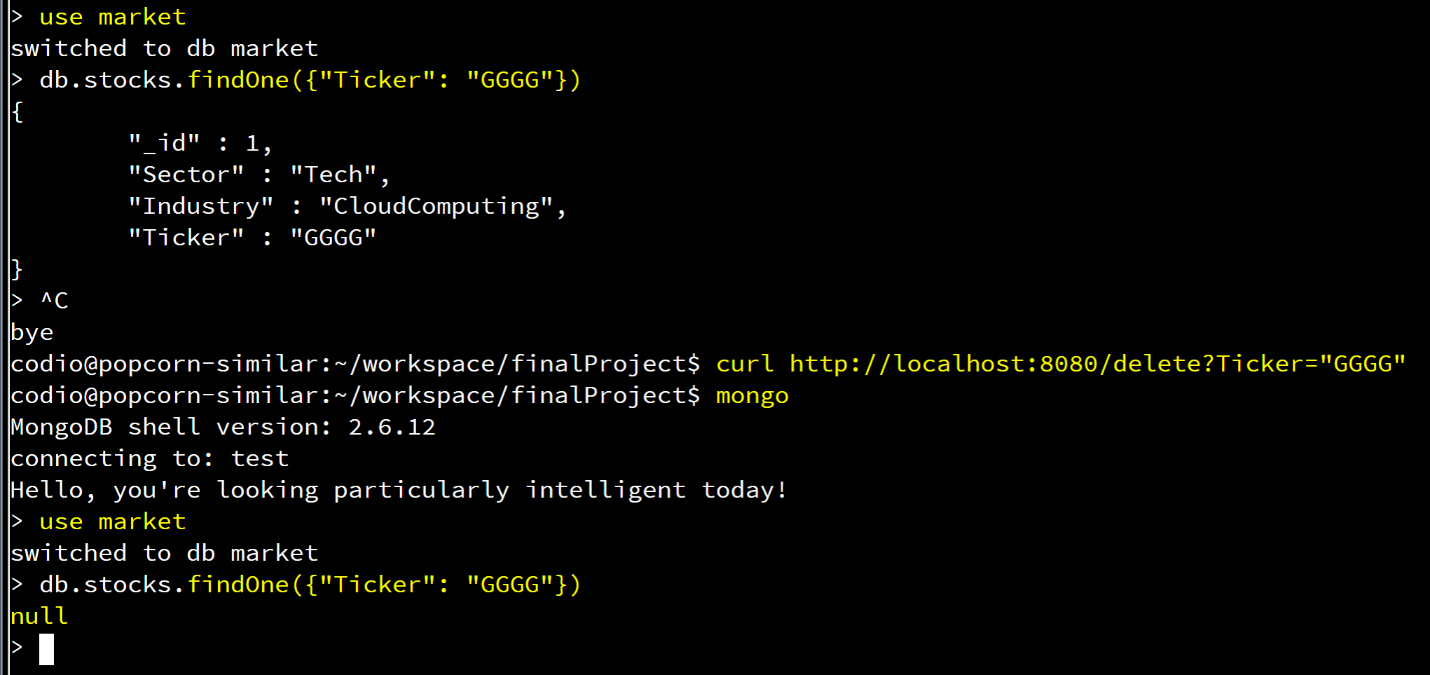
**Read:** With curl, I request the /read api and passing criteria. In this example, I request the “JNJ” ticker. The JNJ document is returned



**Update:** With curl, I request the /update api passing in the ticker for the target document, and the value I want to update. In this case, I change the industry for GGGG to “CloudComputing” I then enter the mongo shell and query GGGG to confirm the update.

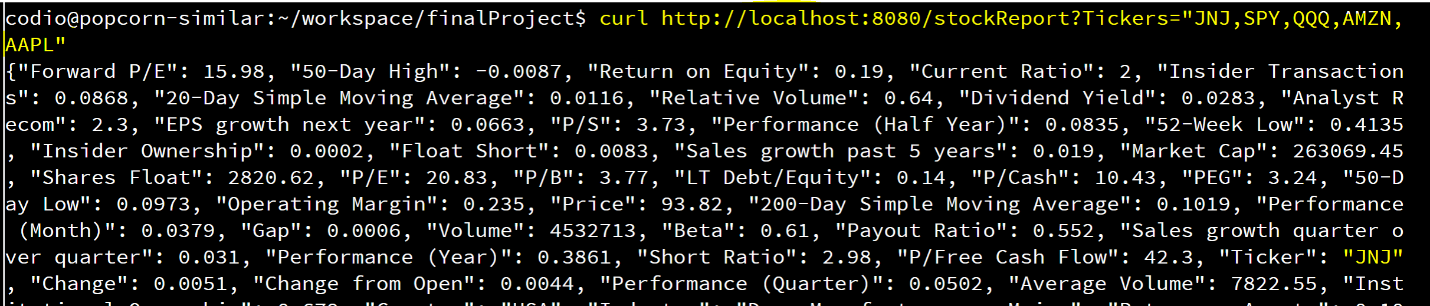


**Delete:** With curl, I request the /delete api passing in the target document to drop. I then enter the mongo shell to confirm that it was removed. In this case, I remove the GGGG document.



**Advanced Programming Project C: (Final\_server.py)**

**C1 5 Stock Ticker info.** With curl, I request the /stockReport api which returns the document for the 5 tickers that I supplied in the url. A large amount of data was returned, so I did not include all of it in the screenshot.



**C2 Industry Report.** With curl, I request the /industryReport api and return data for the top 5 stocks in the industry. It was not clear how to identify the “Top 5” so, I just returned data for the top 5 by market cap for the “Diversified Communication Services” Industry.

