Assignment 2: MongoDB and Cassandra

By

Rachael Joan Dias

0651897

Addressed to,

Professor Brian Srivastava

Trent University

AMOD-5410H-A-2019GW-PTBO Big Data

**Objective:** Data was gathered from 2 different sources Twitter and Coinmarketcap. The information from Twitter consisted of all tweets related to a cryptocurrency posted by a twitter user, while coinmarketcap keeps track of the latest value of a cryptocurrency. The data extracted from both the API were stored on MongoDB in the form of documents. Similarly, data was extracted from the same 2 sources and stored on tables in Cassandra.

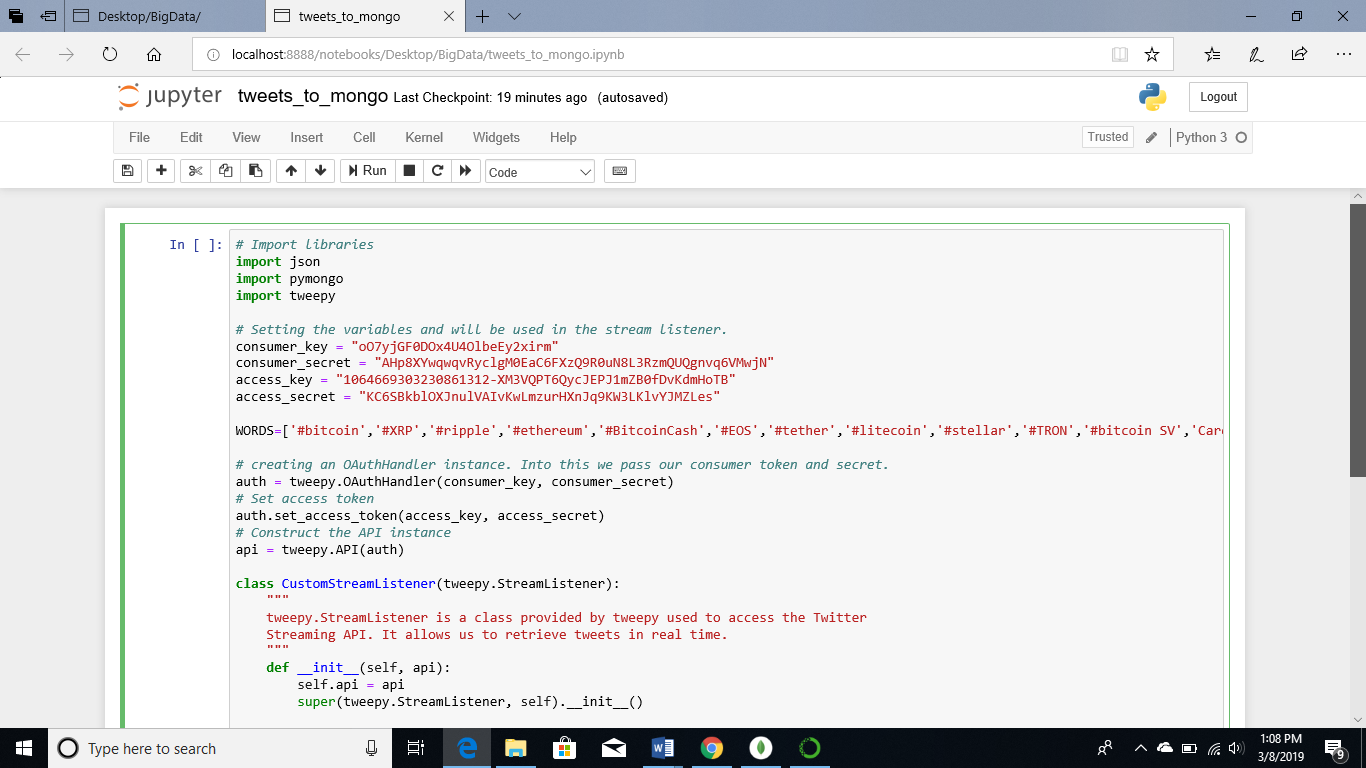
After gathering and storing the data on 2 different databases multiple data manipulation queries were run to search for keywords, determine the number of documents containing a particular keyword and order data based on the timestamp.

**Part 1:** Data Gathering and Storage

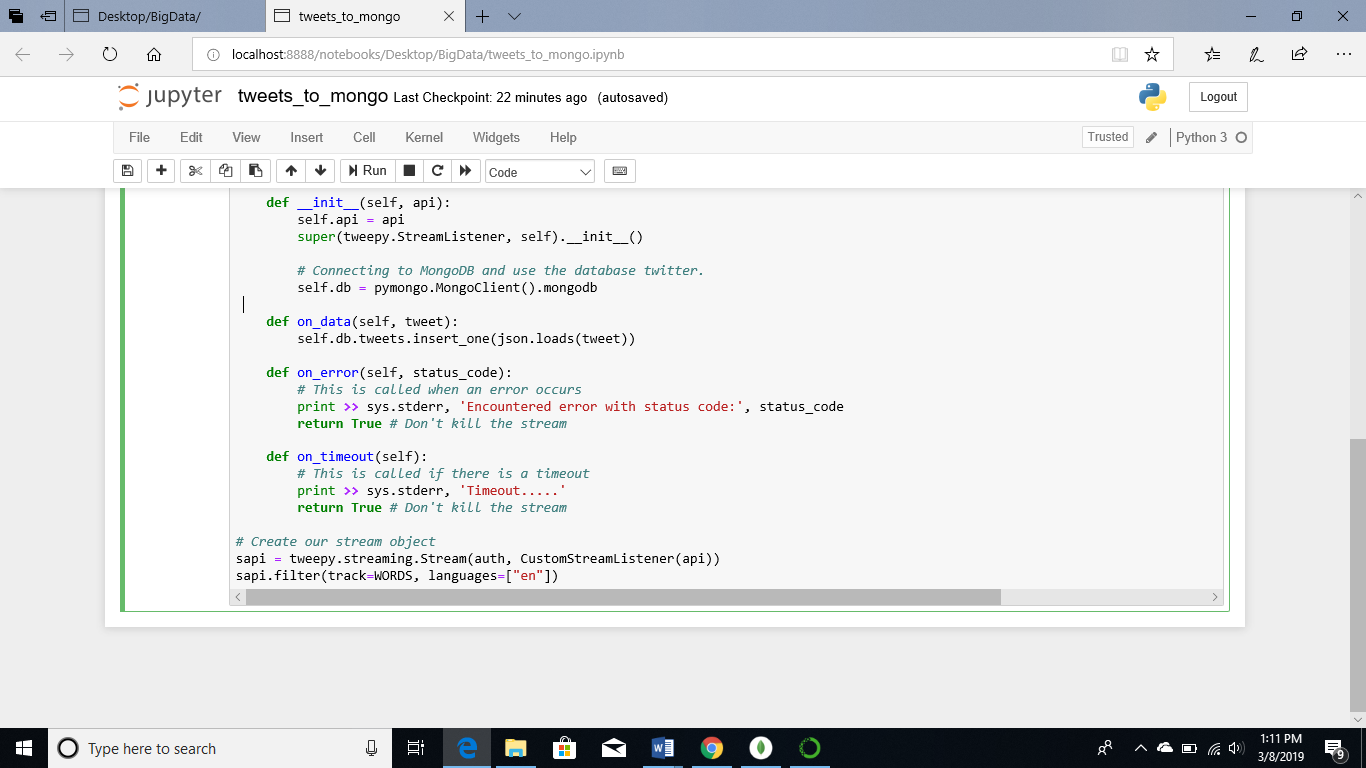
**MongoDB Data Gathering and Storage**

All tweets that contained names of cryptocurrencies such as “Bitcoin”, “TRON”, “Ethereum” etc. were stored on a database “mongodb” on MongoDB. A python script “tweets\_to\_mongo” was executed to load streaming tweets into MongoDB using twitter’s streaming API.

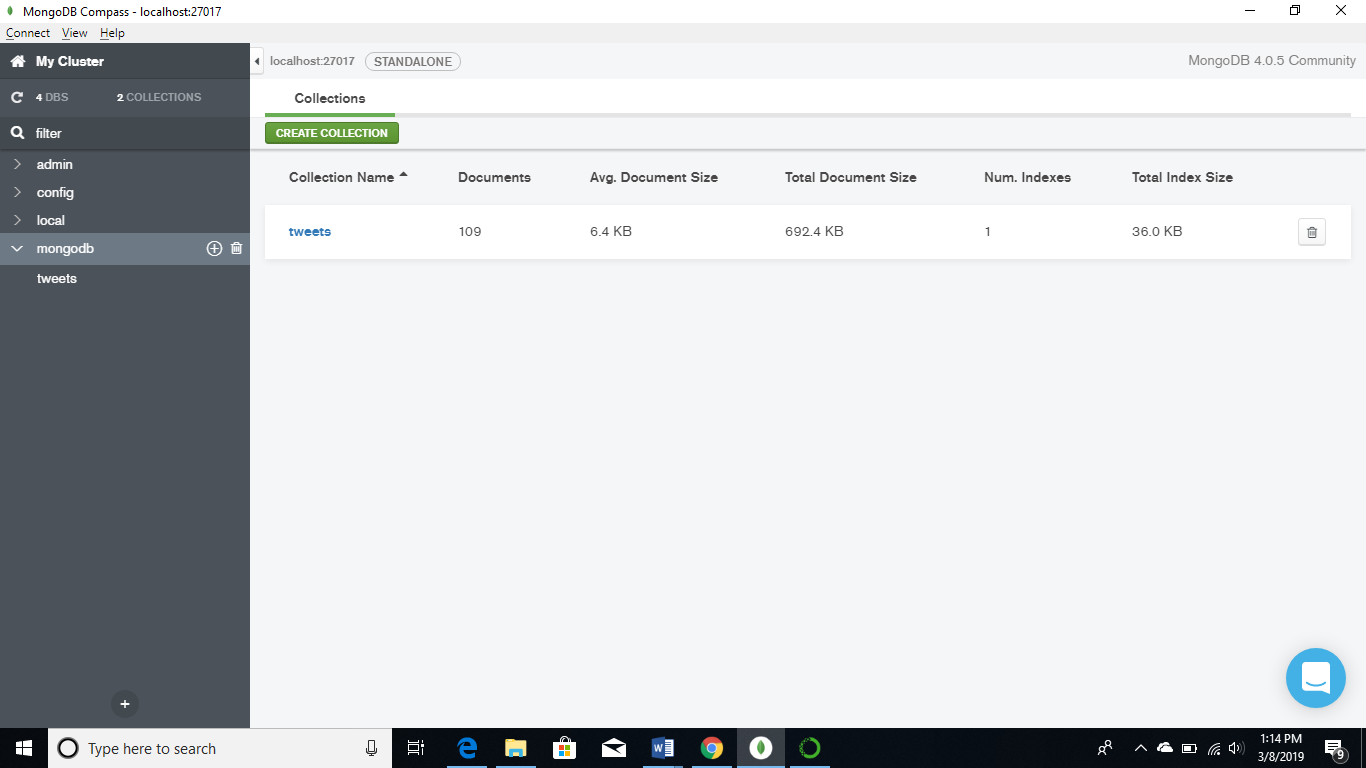
The 3 libraries used were json, pymongo and tweepy. Json library is used to read the tweets, pymongo is the python connector used to insert the tweets into MongoDB and tweepy is the library used to extract streaming tweets. First, the consumer and access keys are specified to set-up the connection, these keys are passed through the OAuthHandler() and set\_access\_token() functions of the tweepy library to initiate the authentication. Once auth is set-up it is passed through the API() function, using the CustomerStreamerListener(tweepy.StreamListener) class streaming tweets are pulled from twitter. Please see below screenshots of the script.



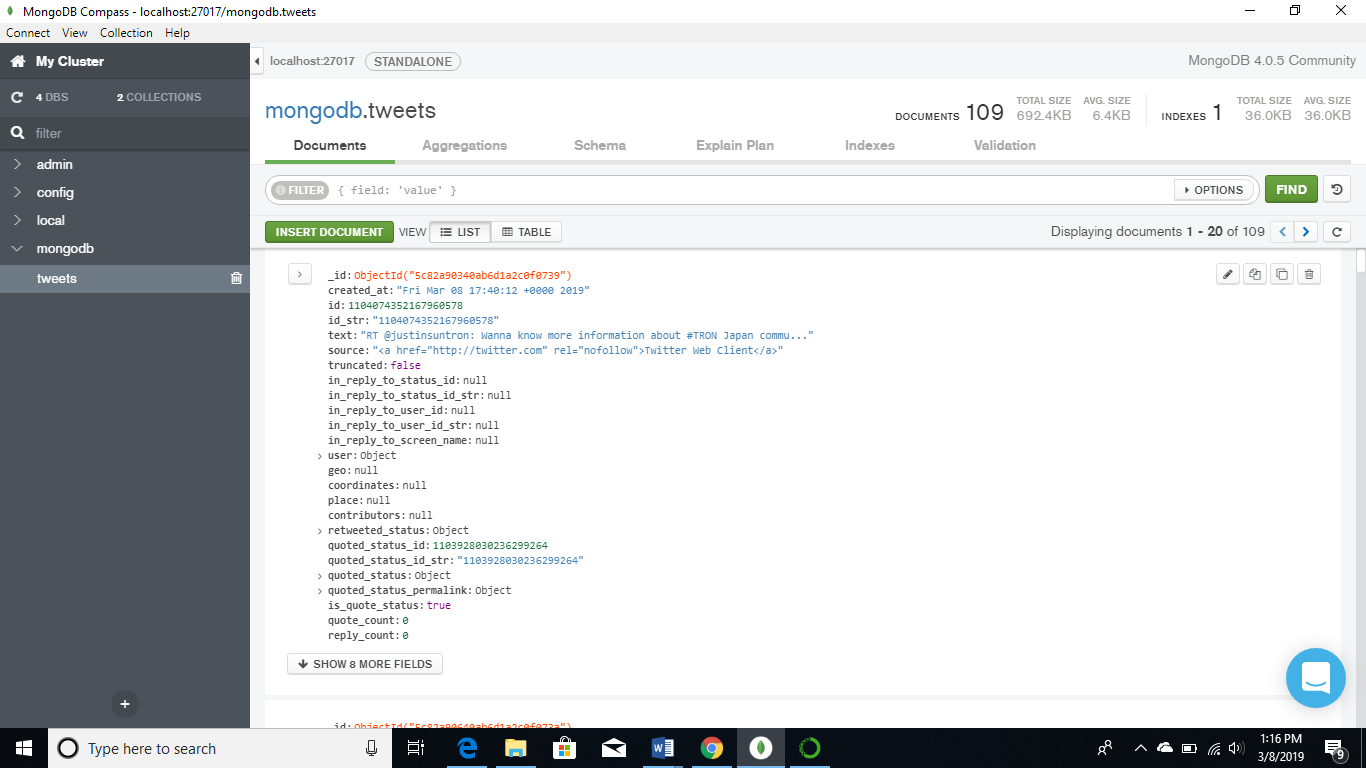
Using the pymongo library a connection is established between python and MongoDB. The self.db.tweets.insert\_one(json.loads(tweet)) inserts streaming tweets into MongoDB. Only English tweets are filter by specifying languages=[“en”] in the sapi.filter() function filter, tweets related to keywords mentioned in the list “WORDS” are inserted.



Below is a screenshot which shows the database “mongodb”, the collection “tweets” and the 109 documents that are stored within the collection.



Below is an example depicting the way information related to a single tweet is stored on MongoDB.

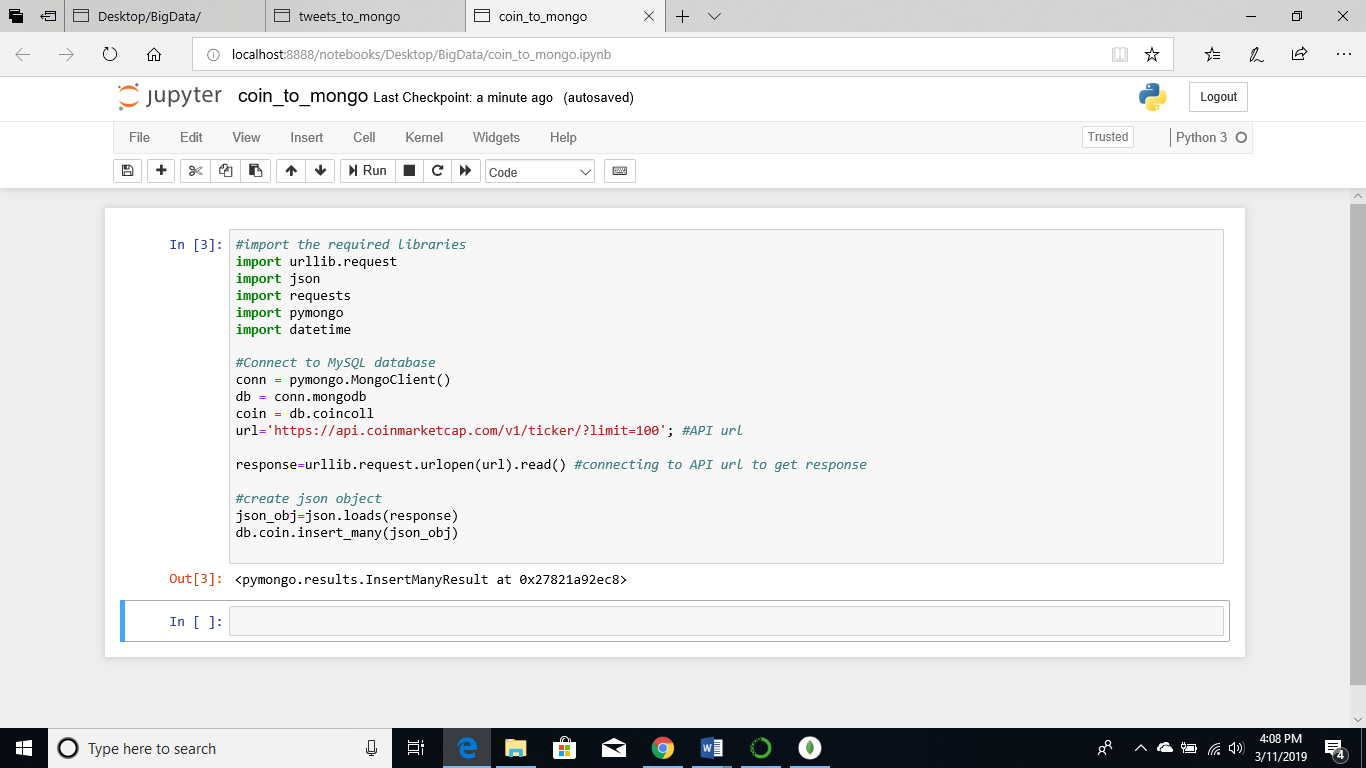


MongoDB is a NoSQL database, it stores information in the form of documents, unlike SQL databases that stores information as rows. Multiple documents form a collection, all the documents within a collection may not contain the same information. In order to search for patterns queries should be run within a field or column of a document.

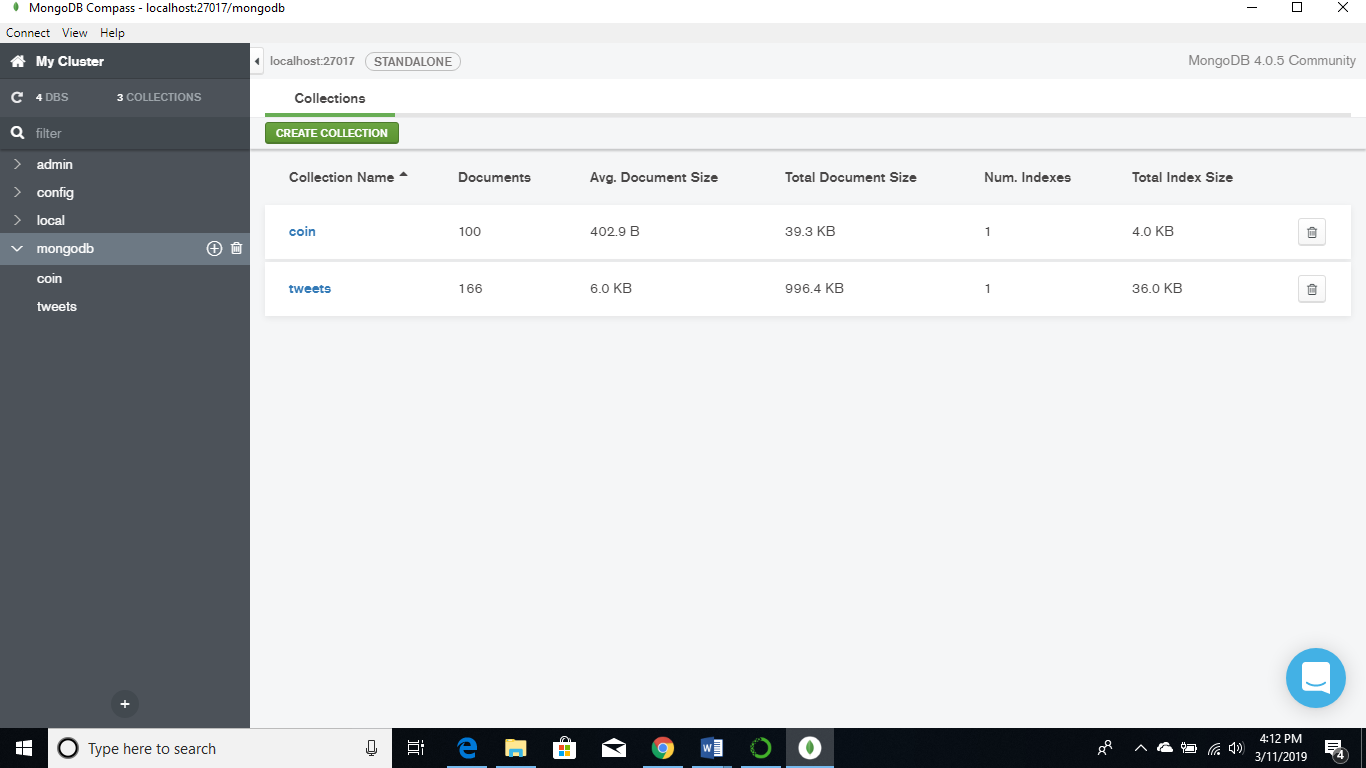
Python script to load Twitter data to MongoDB: tweets\_to\_mongo.ipynb

In a similar manner data from coinmarketcap was loaded into MongoDB. Below is a screenshot the script used to load data into MongoDB.

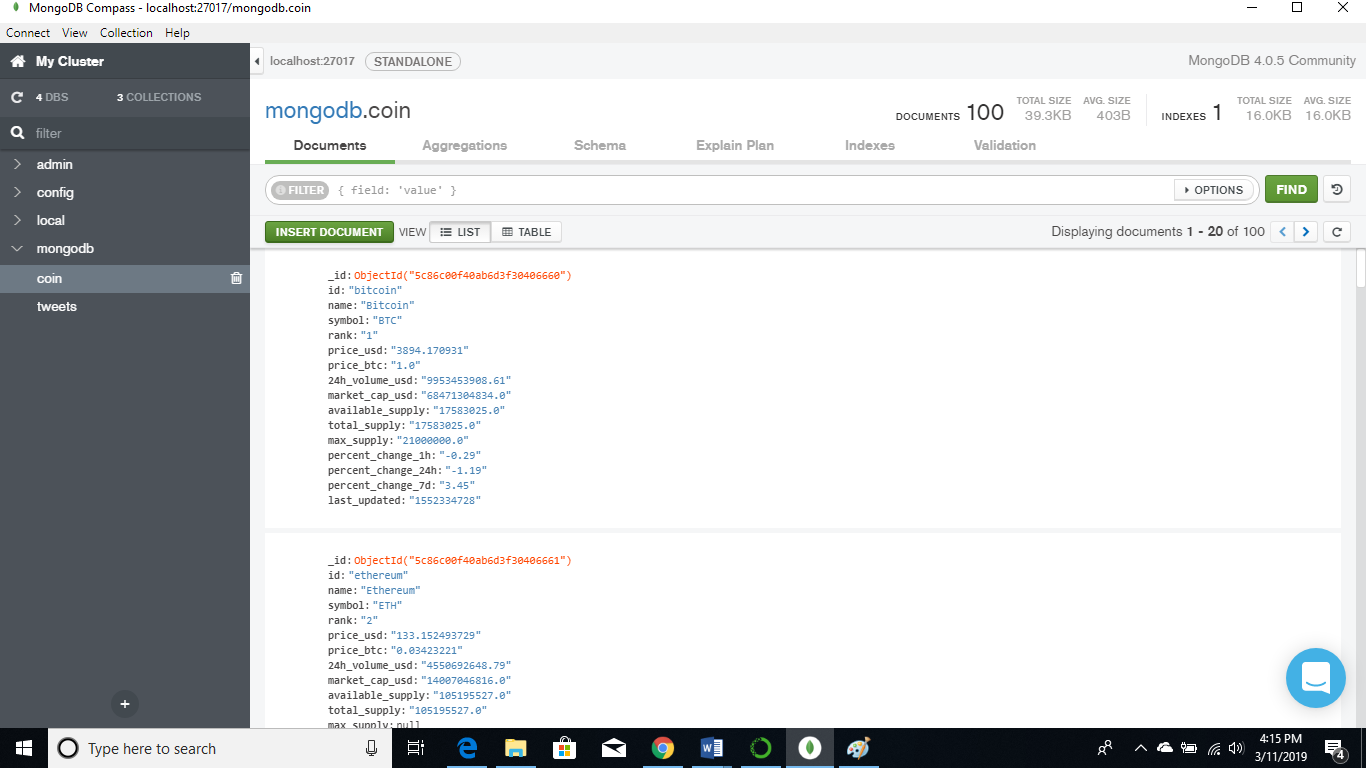
First, a connection is established between MongoDB and Python by using the MongoClient() function from the pymongo library. Once the connection is made, the database “mongodb” is specified followed by the collection “coin”. In the case of coinmarketcap data is extracted from a url, the url link specifies the number of entries we would like to extract, the top 100 entries are selected. Using the json library the top 100 entries are loaded into a json object and then with the insert\_many() function of pymongo library they are loaded into the collection “coin” on MongoDB.



The collection “coin” is stored on the database “mongodb”. There are 100 documents present under the collection coin.



All information related to a cryptocurrency is stored as follows.

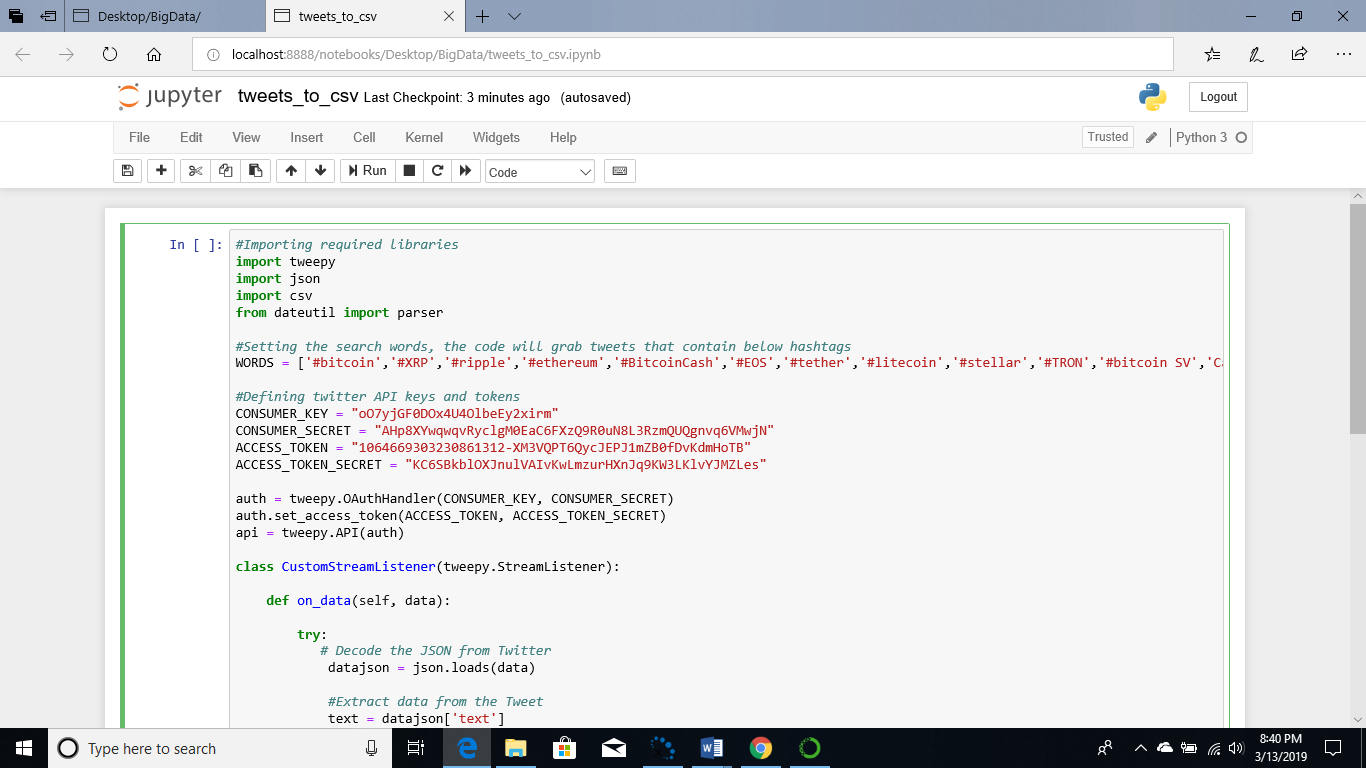


Python script to load coinmarketcap data to MongoDB: coin\_to\_mongo.ipynb

**Cassandra Data Gathering and Storage**

Twitter and coinmarketcap data were extracted into a .csv files and then import into tables on Cassandra.

Streaming tweets were extracted using the tweepy library based on keywords related to cyptocurrencies. With the help of the csv library data was loaded into a csv file, since twitter data is in json format, only some fields for each tweet were extracted and loaded into the csv file. Below are screenshots of the script which loads extracts tweets into a csv file



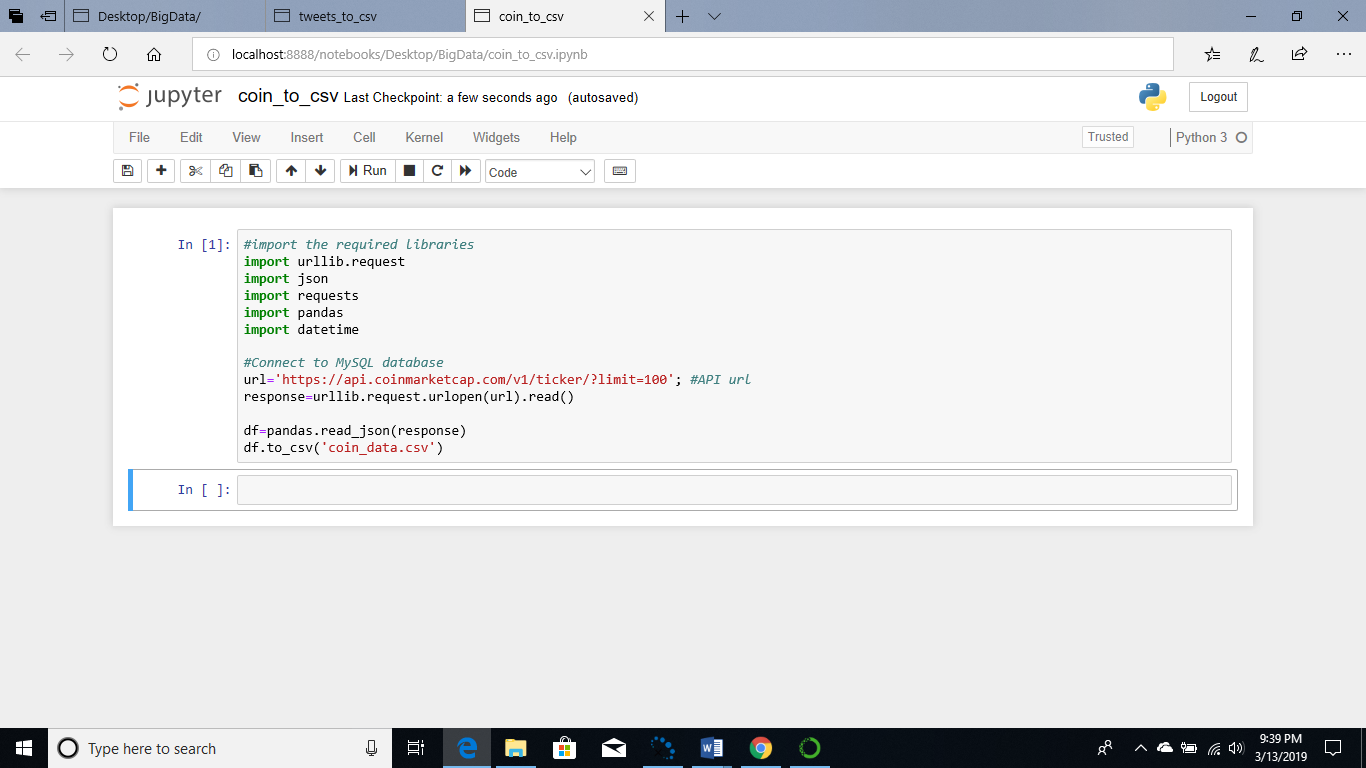
First json data is loaded into a json object using the json.loads(data) command. Then, certain fields of the json object text, name, screen\_name, tweet\_id, created\_at, follwers and friends are extracted from the tweets.

With the help of the writer function from the csv library, a file tweets.csv is opened and encoding=”utf-8” is specified so that tweets or usernames containing special characters can be inserted. The writerrow() function appends the tweets one at a time to the csv file. Finally a header is hardcoded to csv file so that each column can be identified.



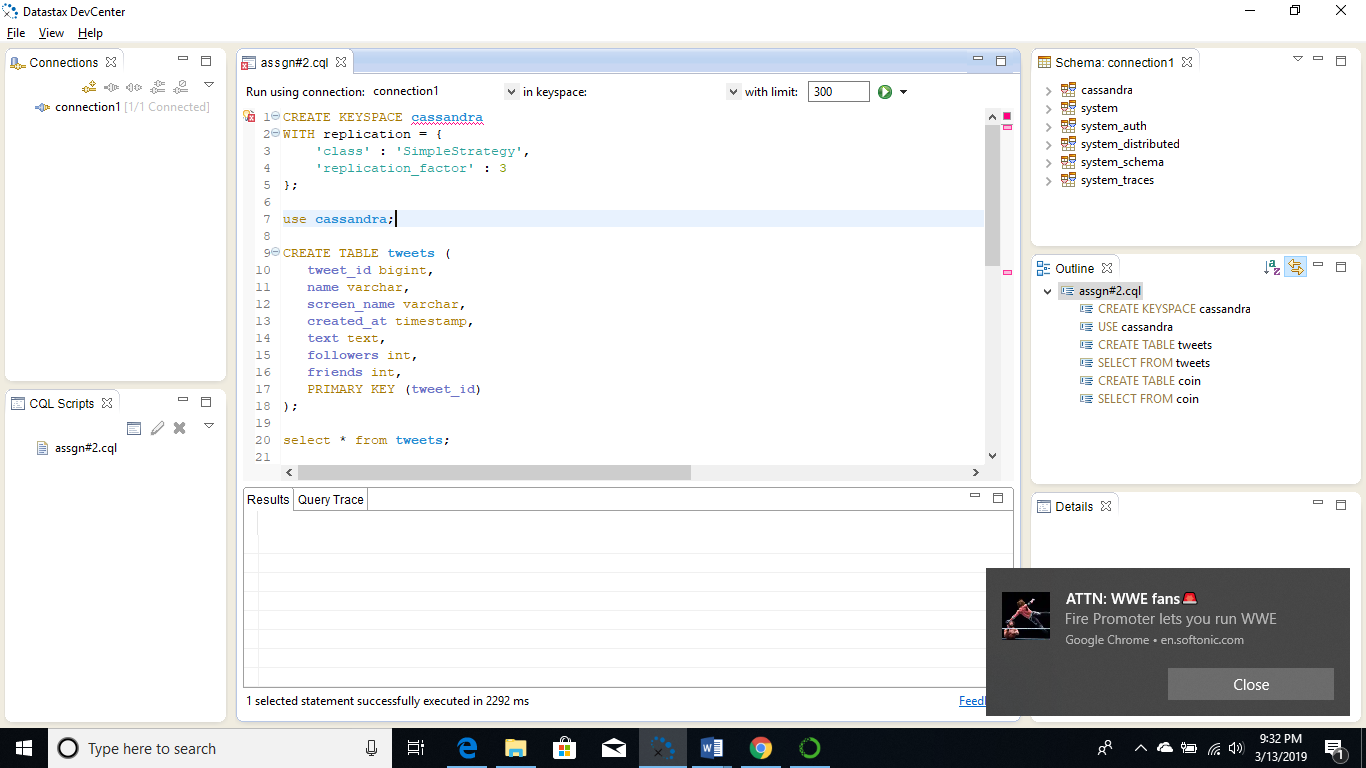
Python script to load tweets to a csv file: tweets\_to\_csv.ipynb

Using the request library the data is read from the url specified, in this case top 100 cryptocurrencies are extracted and saved in a json object called response. Since the data is in json format using the read\_json() from the pandas library we can convert the data to a dataframe and then store it on a csv file.

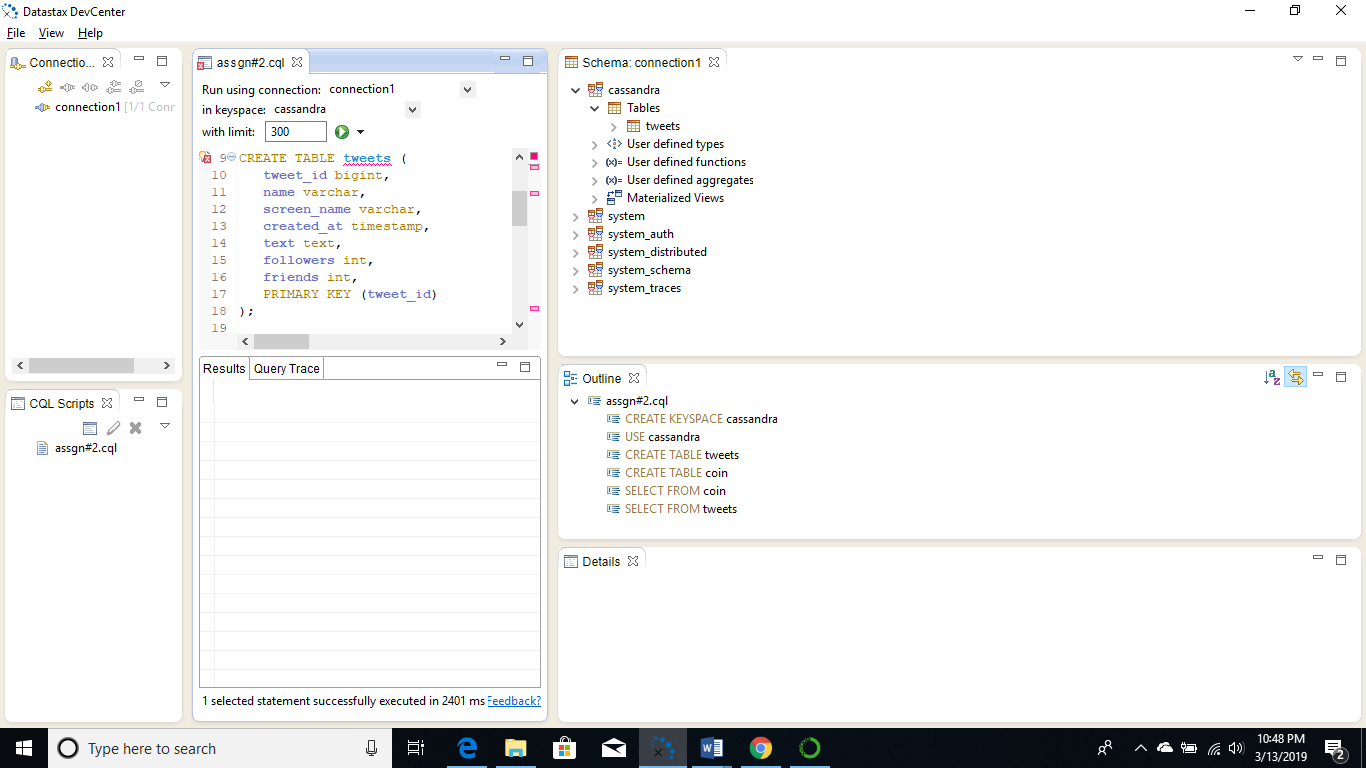


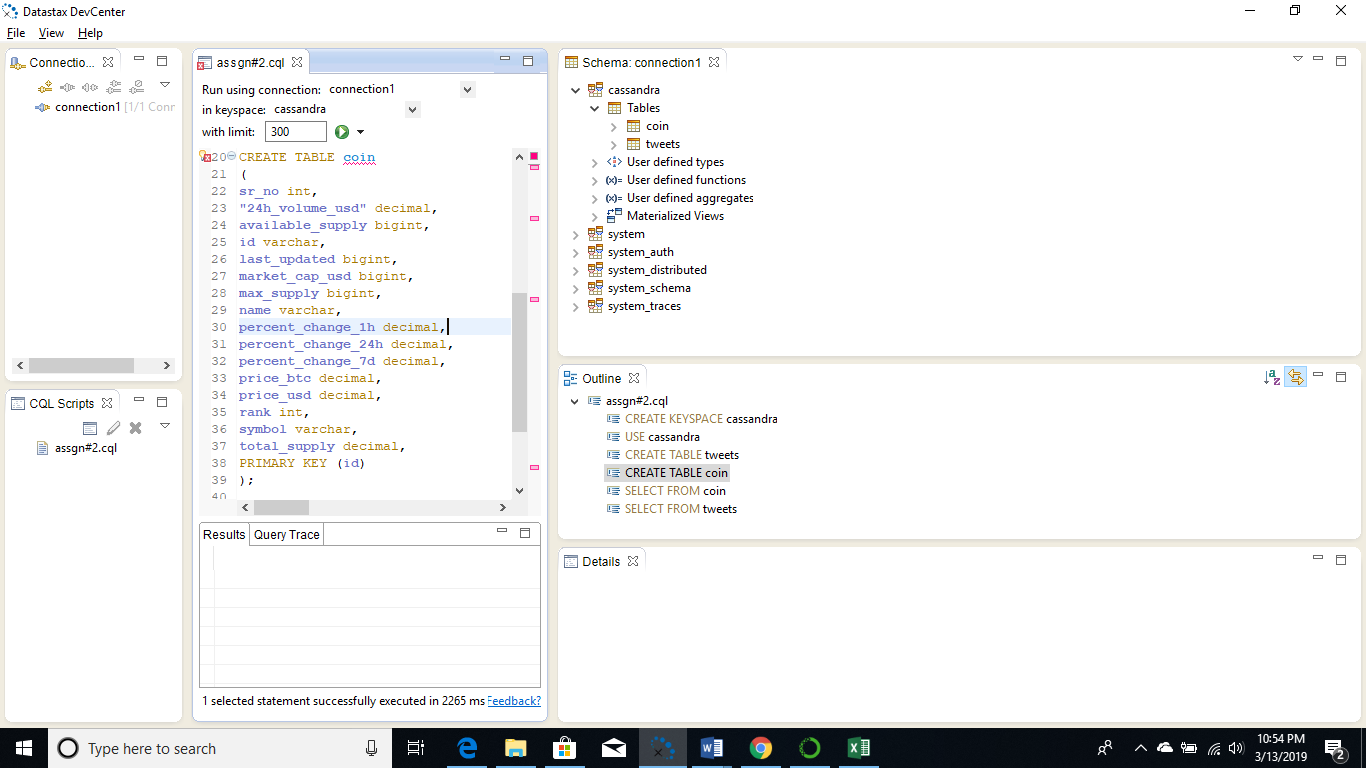
Python script to load tweets to a csv file: coin\_to\_csv.ipynb

A keyspace is created in Cassandra, keyspace is like a database which stores all the tables, in the query below a keyspace named “cassandra” is created the replication factor is set to 3 which means that 3 copies of the data are kept in a given data center, “SimpleStrategy” means that the same replication factor is assigned to the whole cluster.

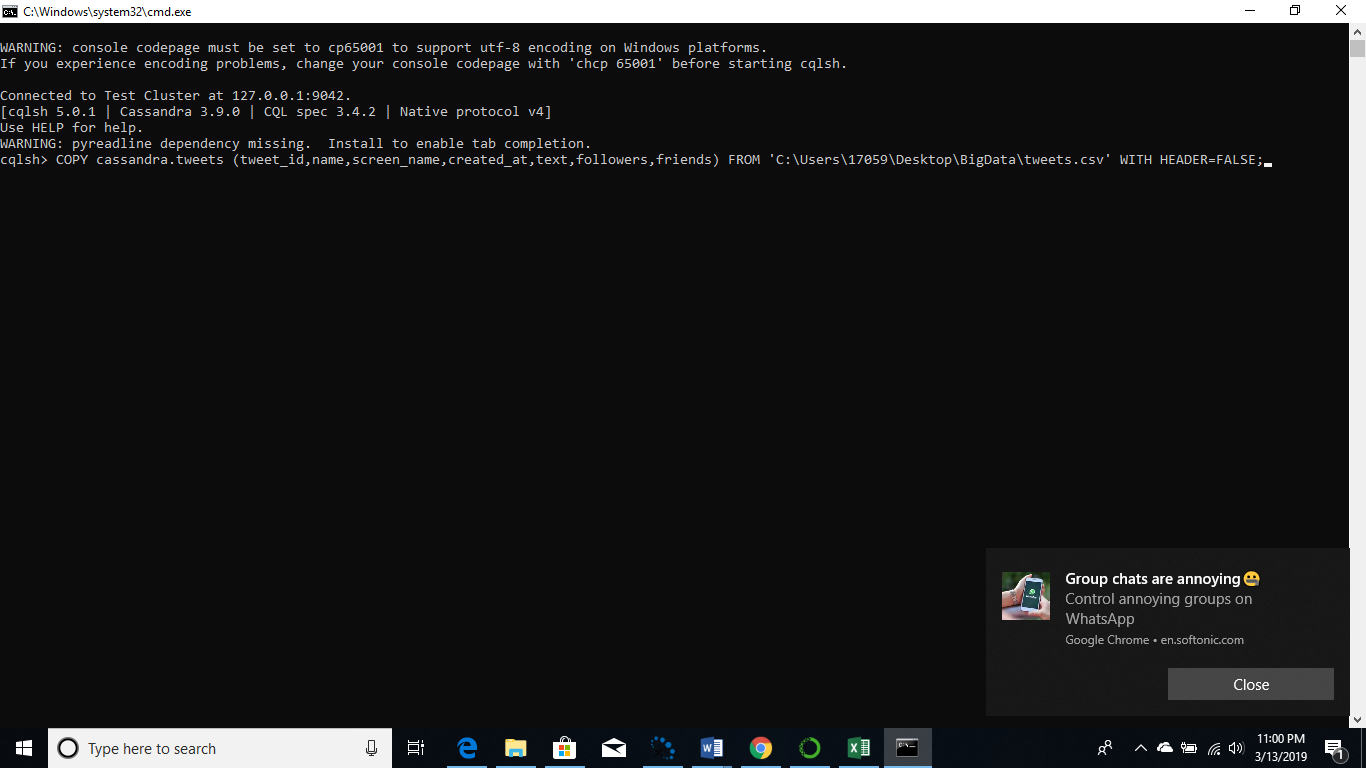


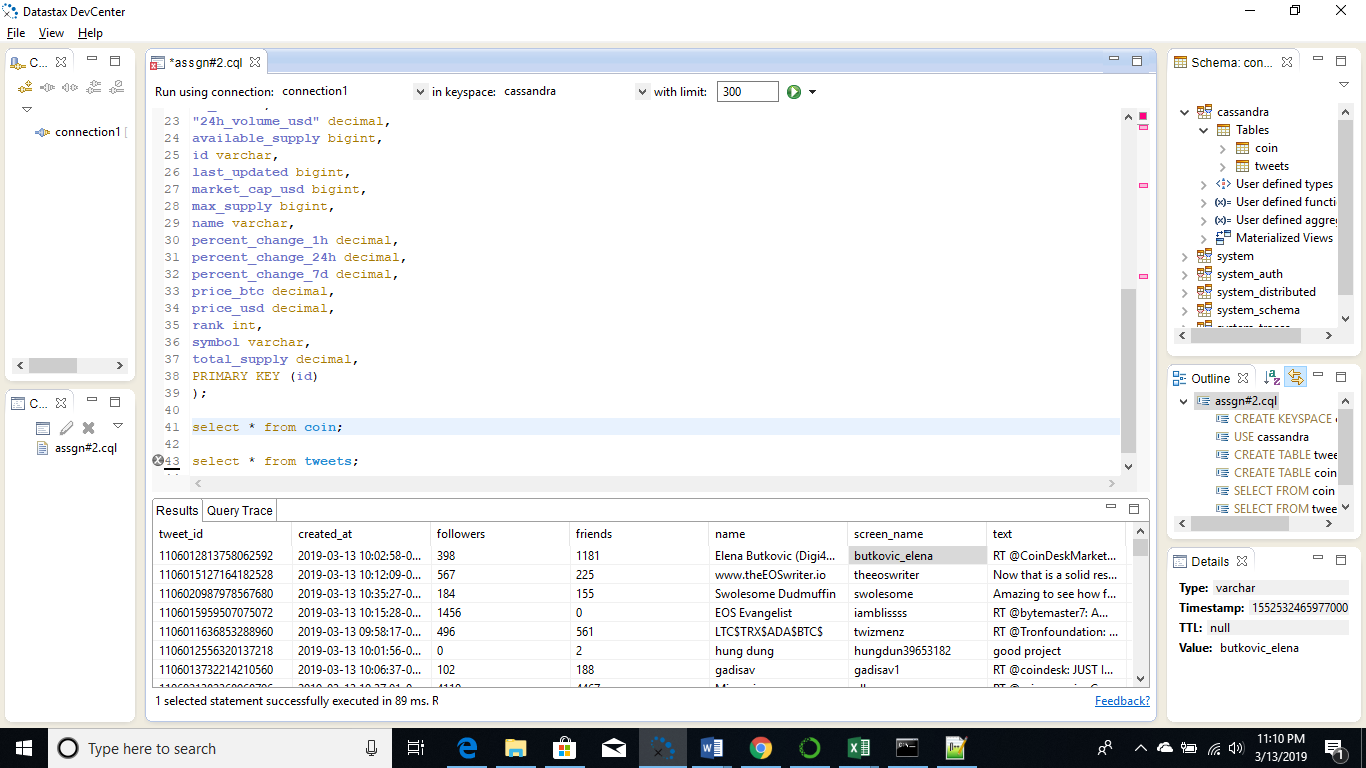
Now, that the database is created, tables need to be created to store data from the csv files. A table “tweets” is created to store twitter data and “coin” stores all the data from coinmarketcap. Below are the data definition commands used to create the tables.

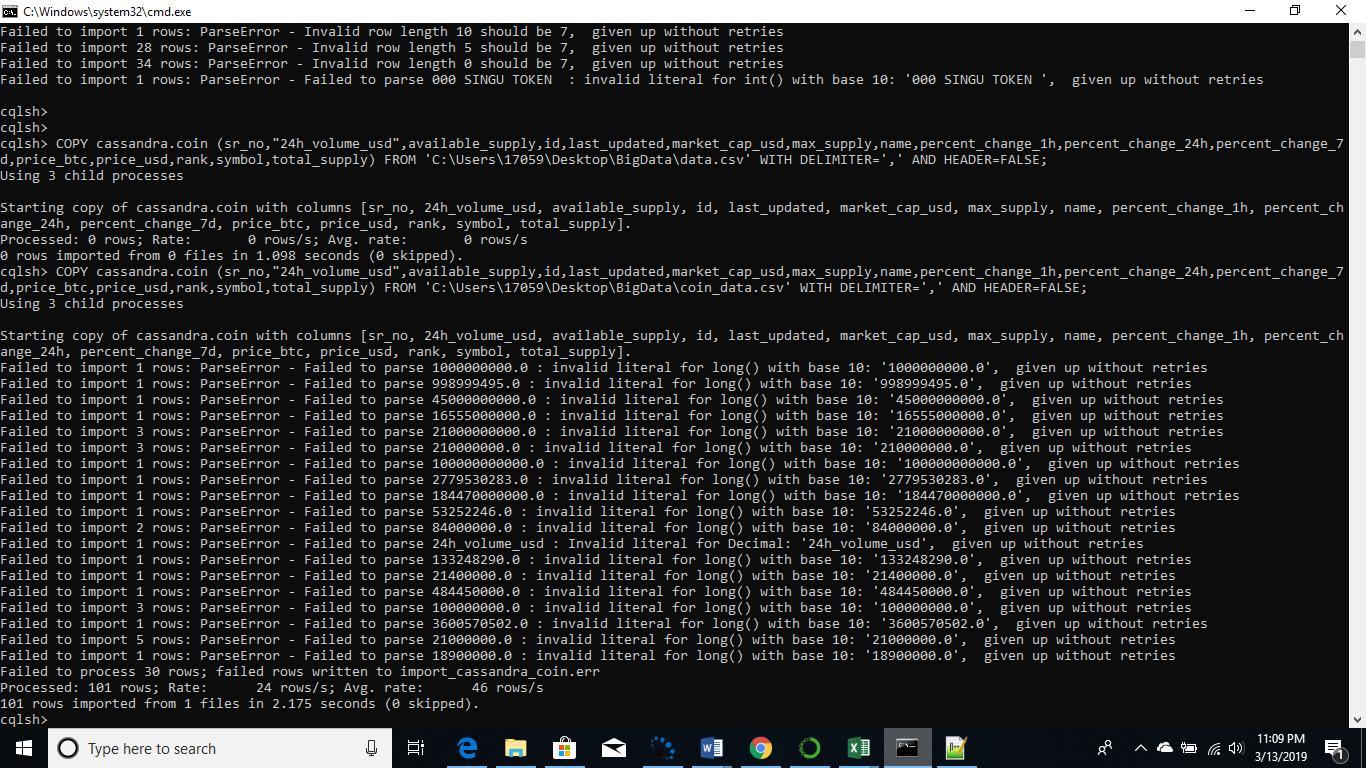




Now that we have tables data has to be imported from the csv files to the tables. Below commands were executed in cqlsh to import data into Cassandra tables “tweets” and “coin”.









**Data Gathering and Storage Challenges**

The first challenge encountered was inserting data directly into Cassandra. I tried to use multiple libraries that were python connectors for Cassandra. But since they didn’t work, I eventually settled on extracting the data to a csv filed and importing it using commands in cqlsh.

It was harder to load data into Cassandra as compared to MongoDB, since data is stored in the form of tables in Cassandra. Data had to be extracted from the JSON object and inserted as rows. MongoDB on the other hand reads the data in JSON format and saves it in JSON format.

Later, I also faced issues with connecting to Cassandra because of multiple versions of Java, Cassandra abruptly stopped working. After uninstalling version 11 of Java and installing only version 8 Cassandra started working properly. Another issue encountered was while inserting data into the tables in Cassandra the .csv had to be stored with utf-8 encoding so that it could be imported to the tables.

**Part 2:** Data exploration

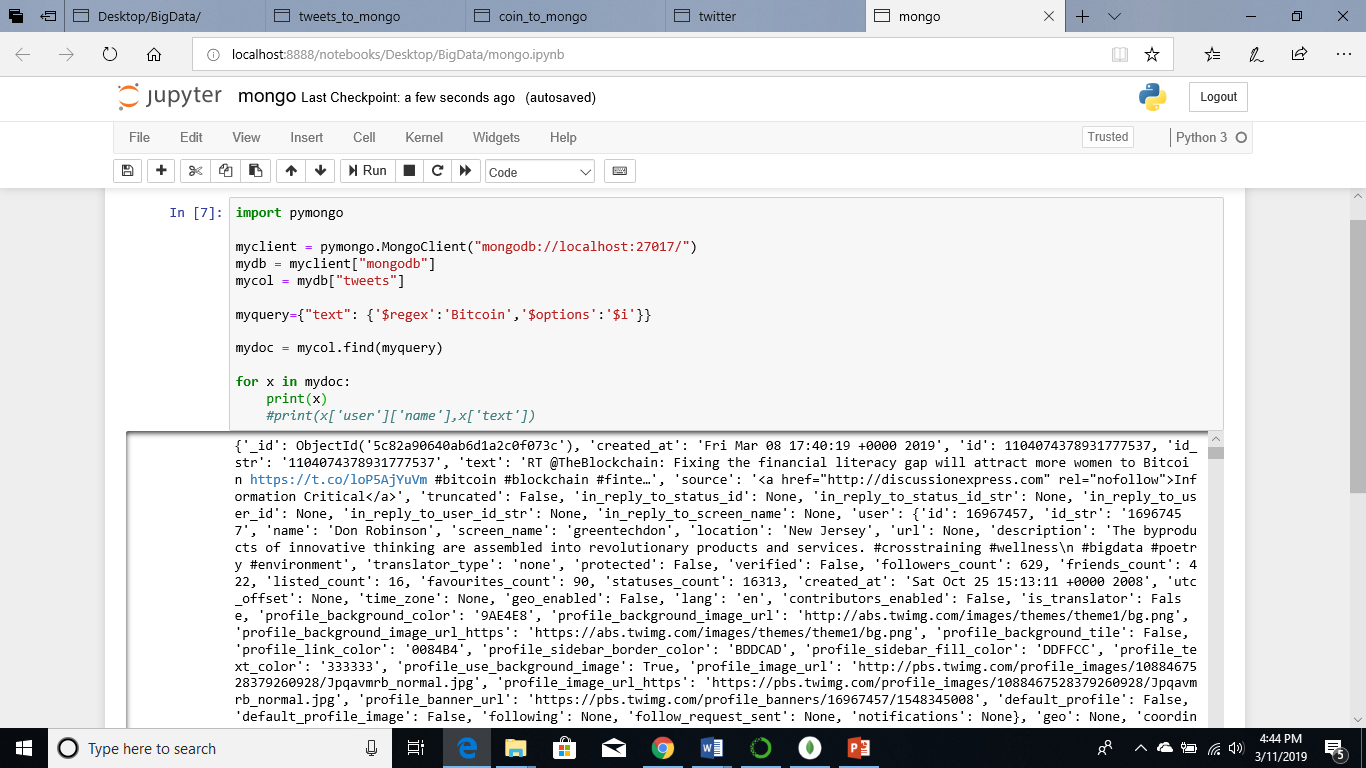
After storing all the tweets and coinmarketcap currency data on MongoDB a few queries were executed.

**Data Exploration in MongoDB**

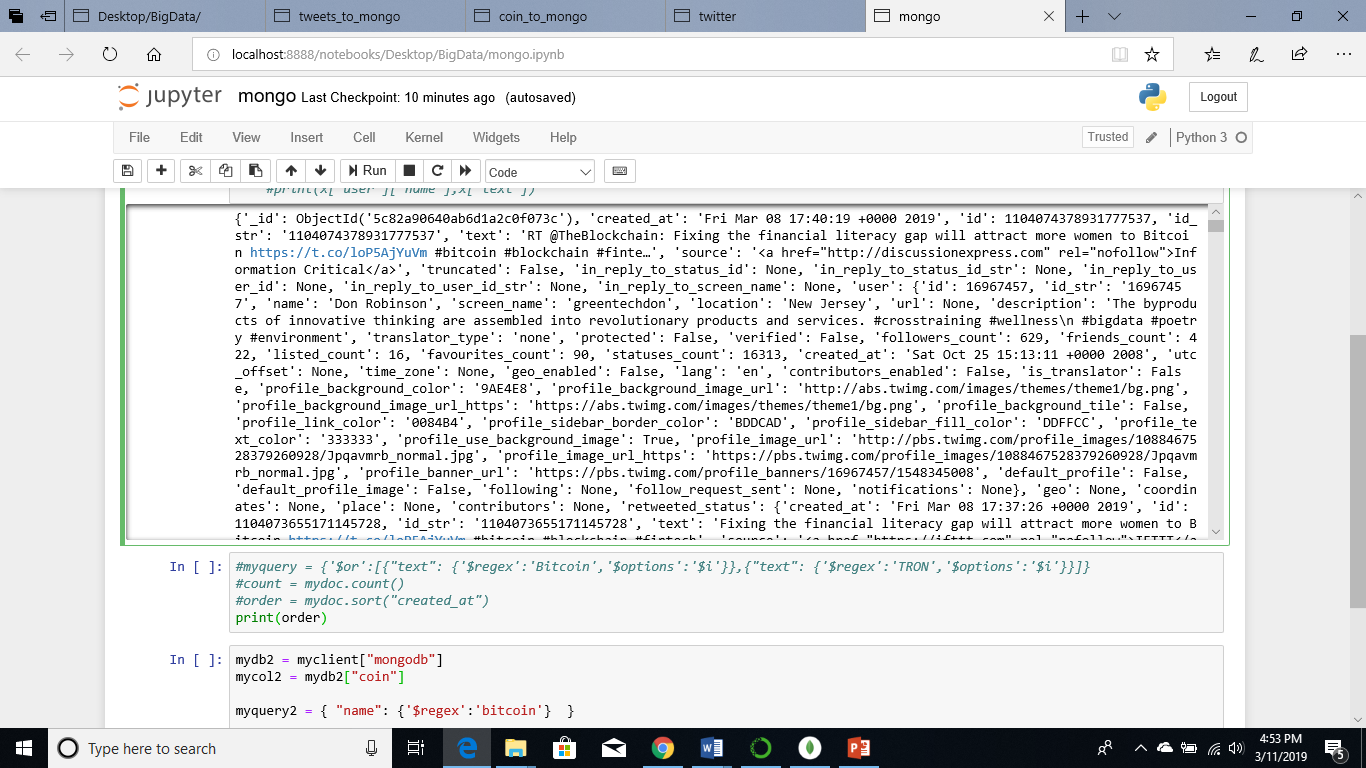
1. **Searching for keywords**

With the following query, all tweets that contain the keyword ‘Bitcoin’ in the text column were filtered out, ‘$i’ means that the keyword is case insensitive, so it will search for both lower- and upper-case words.

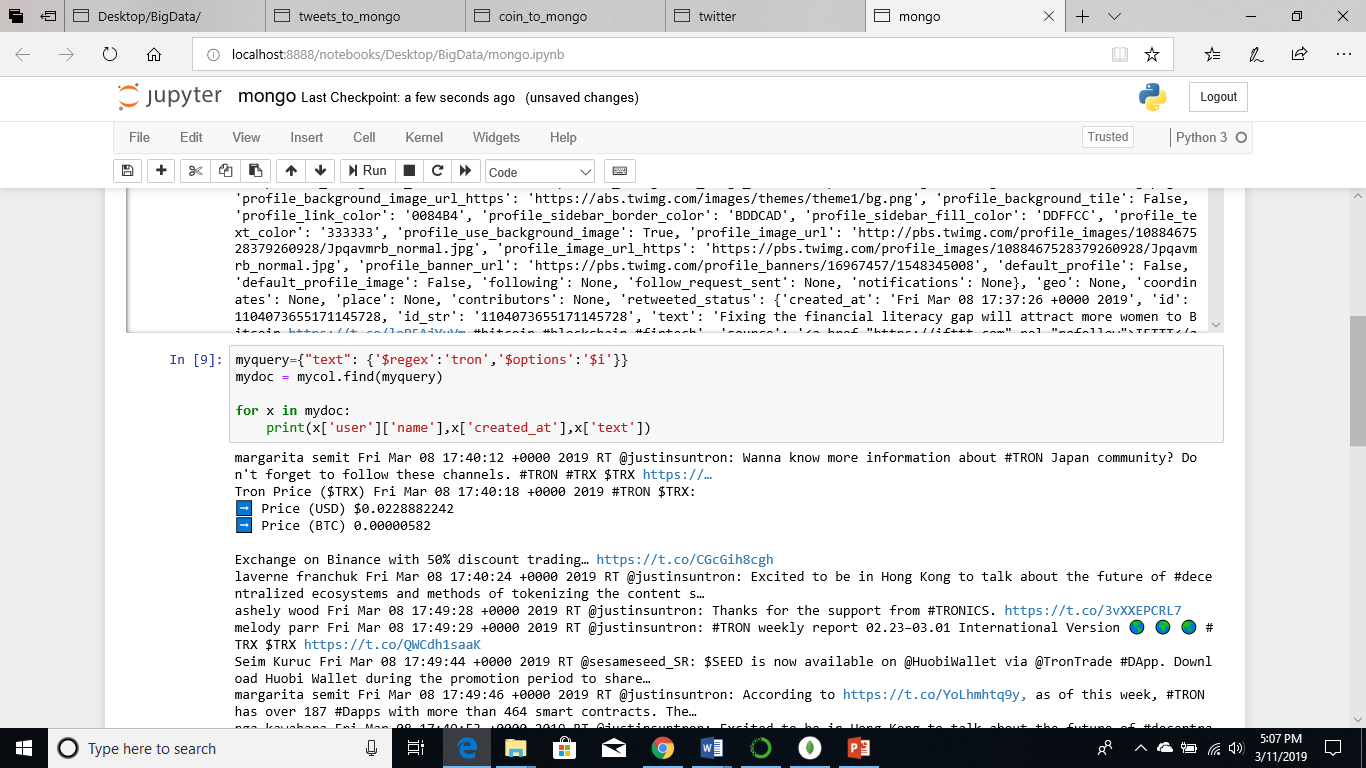
*myquery={"text": {'$regex':'Bitcoin','$options':'$i'}}*



Below is the output of a tweet that contains the word bitcoin all the fields of the tweet are printed to the screen

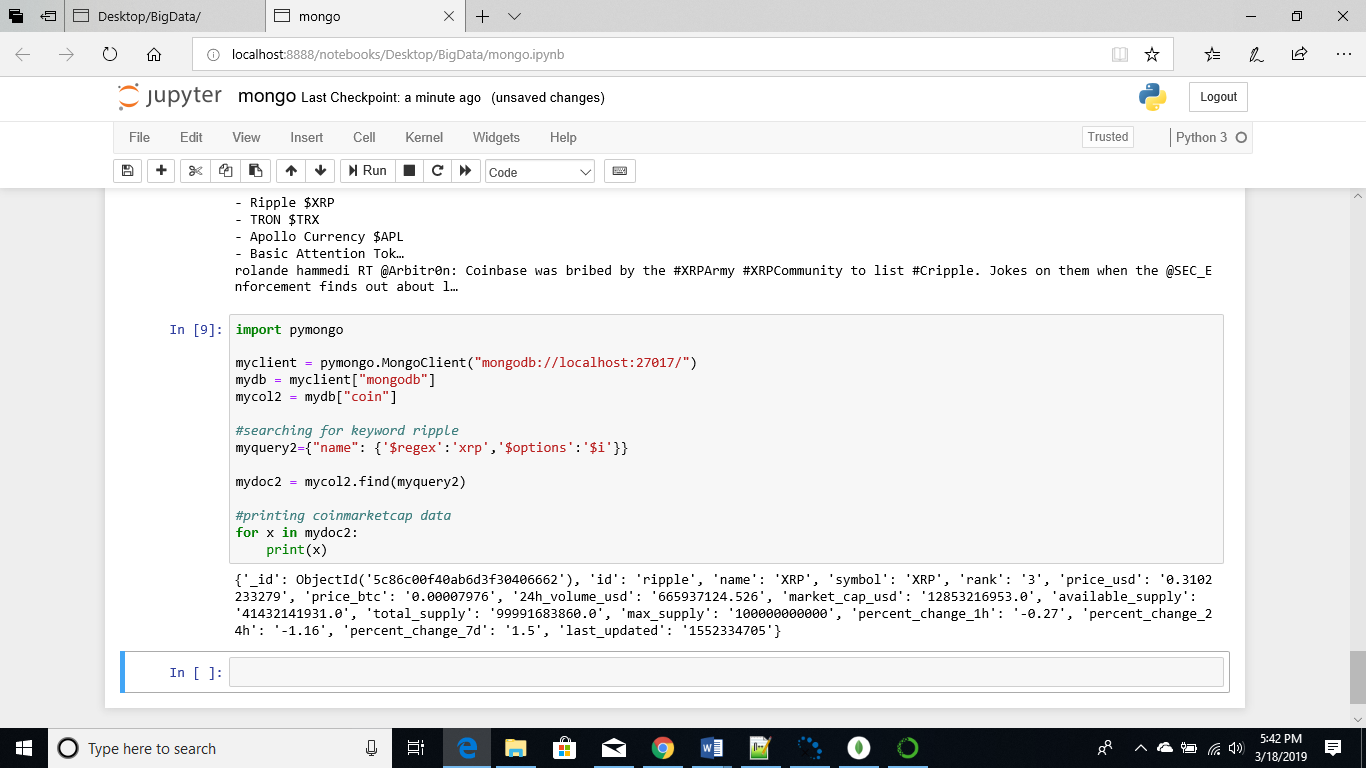


In the example below only the fields username, created\_at, and text are printed for tweets containing the keyword “tron”





Similarly, for coinmarketcap the keyword search was done using the “$regex” operator. Since the data consist of only the top 100 cryptocurrencies, the query returns only one entry which is related to the Ripple currency. The coin collection contains only one entry for each cryptocurrency.



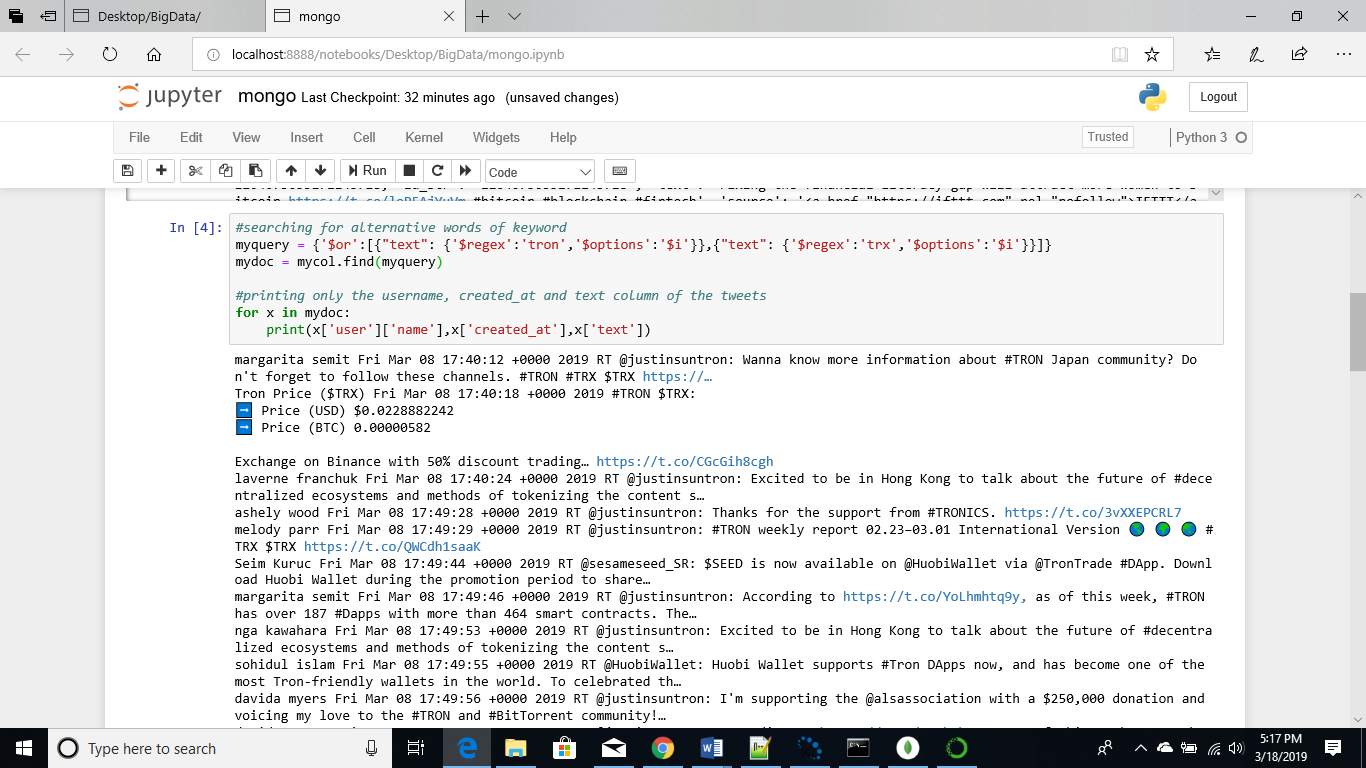
Keyword search or pattern searching is relatively easier in MongoDB because of the functionality provided by the string pattern matching “$regex” operator.

1. **Searching for alternate forms of the same keyword**

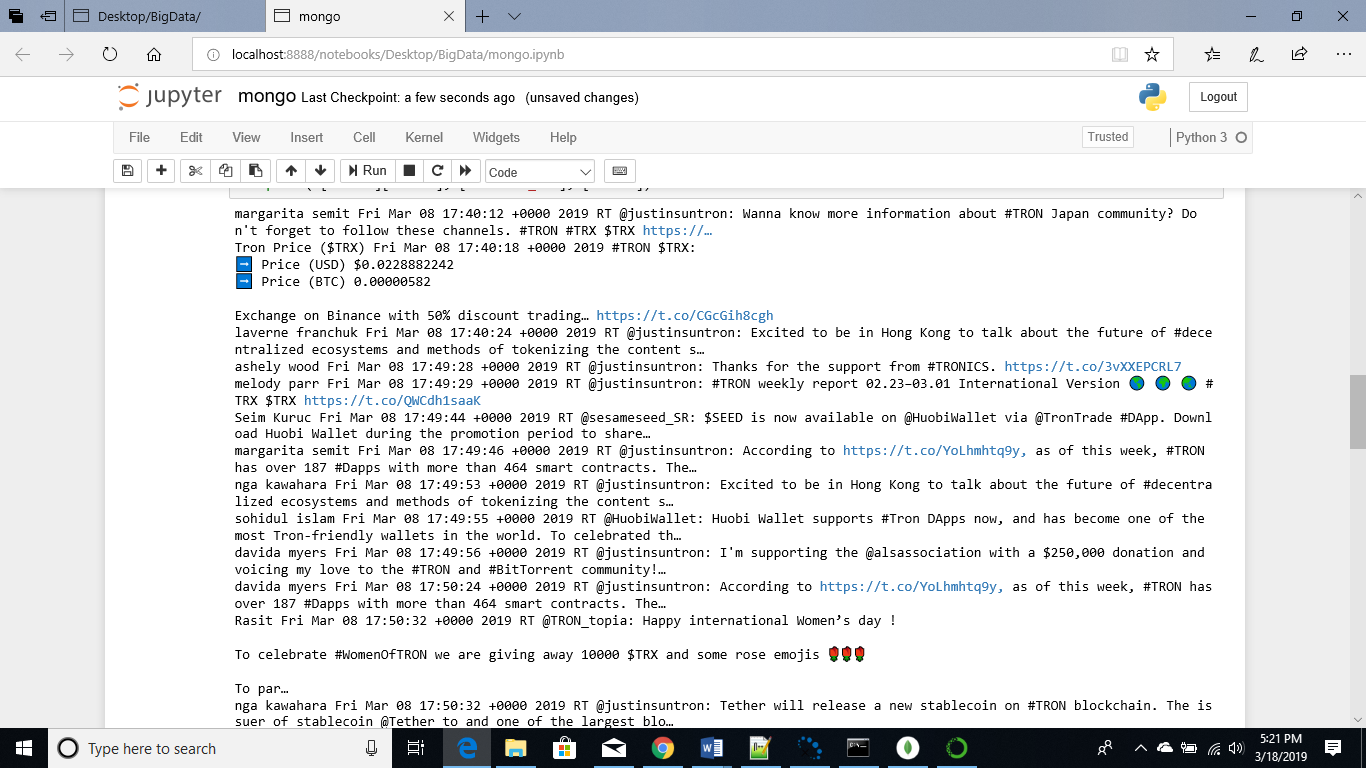
Alternate forms of the same keyword can be searched using the “$or” operator

In the example below, I searched for the cryptocurrency TRON, another word used to identify the TRON currency is TRX

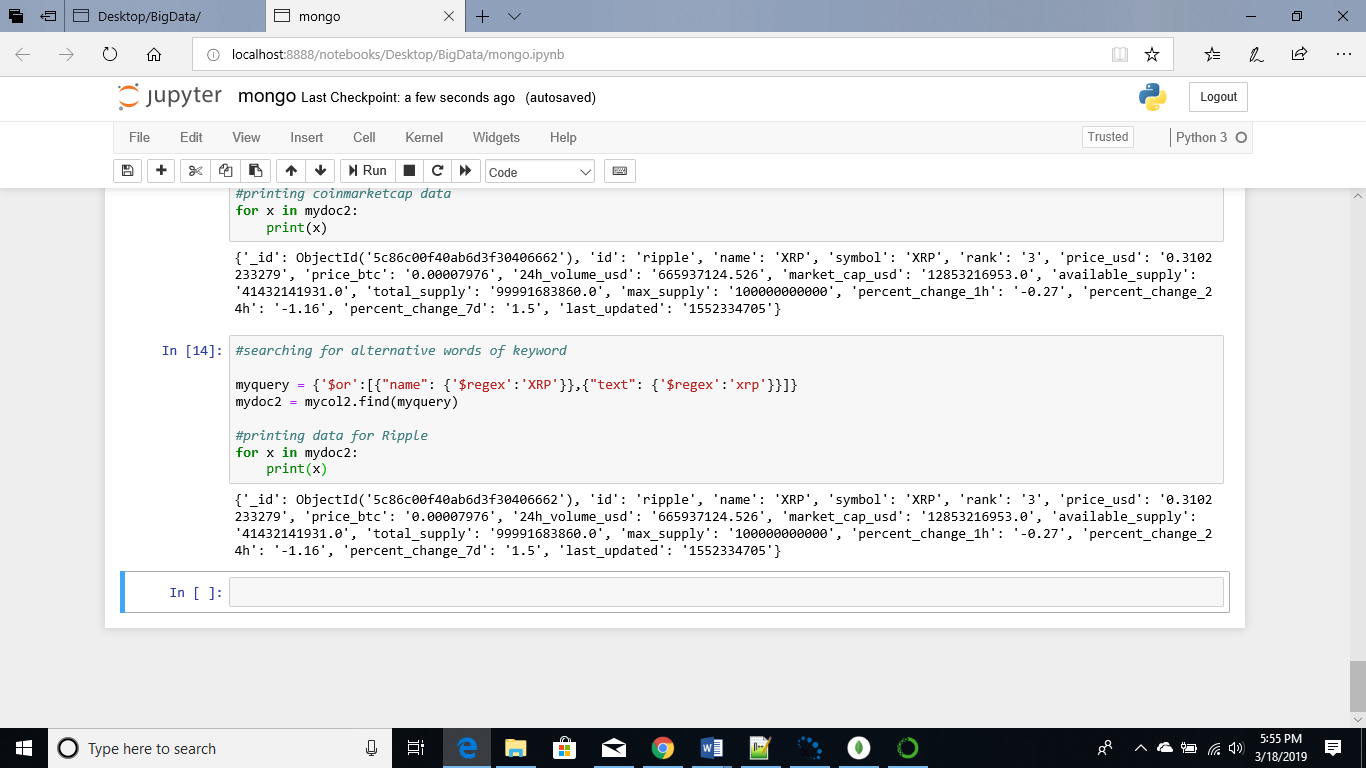
*myquery = {'$or':[{"text": {'$regex':'tron','$options':'$i'}},{"text": {'$regex':'trx','$options':'$i'}}]}*



The output contains all the tweets that have both TRON and TRX in the text field. Similarly, for other cryptocurrencies we can search for other terms that are used to talk about those currencies using the “$or” operator.

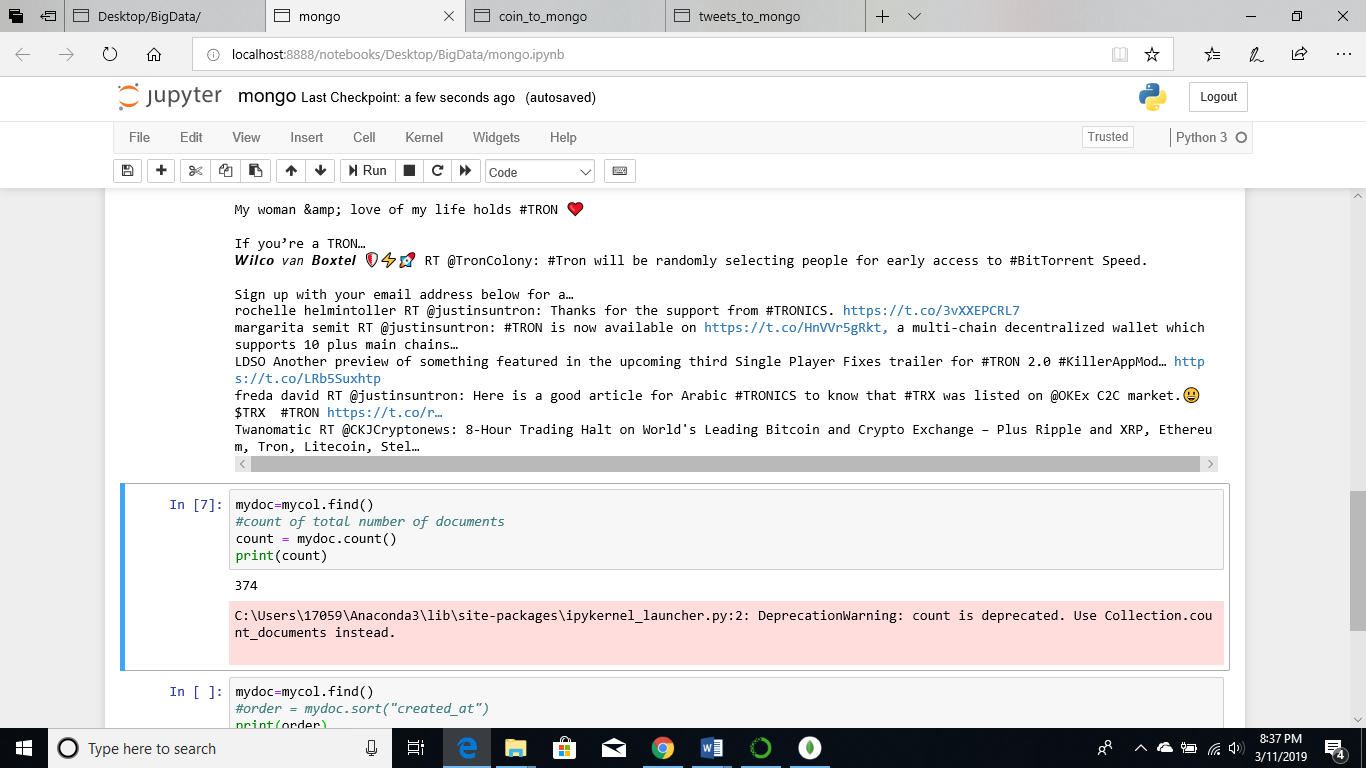


The query below returns data from the collection “coin” that was extracted from coinmarketcap, to demonstrate searching for alternate forms of the same keyword I searched for XRP in both upper and lower case and removed the option field which is used to specify case-insensitive. Since Ripple is represented as “XRP” querying only “xrp” will return results results.

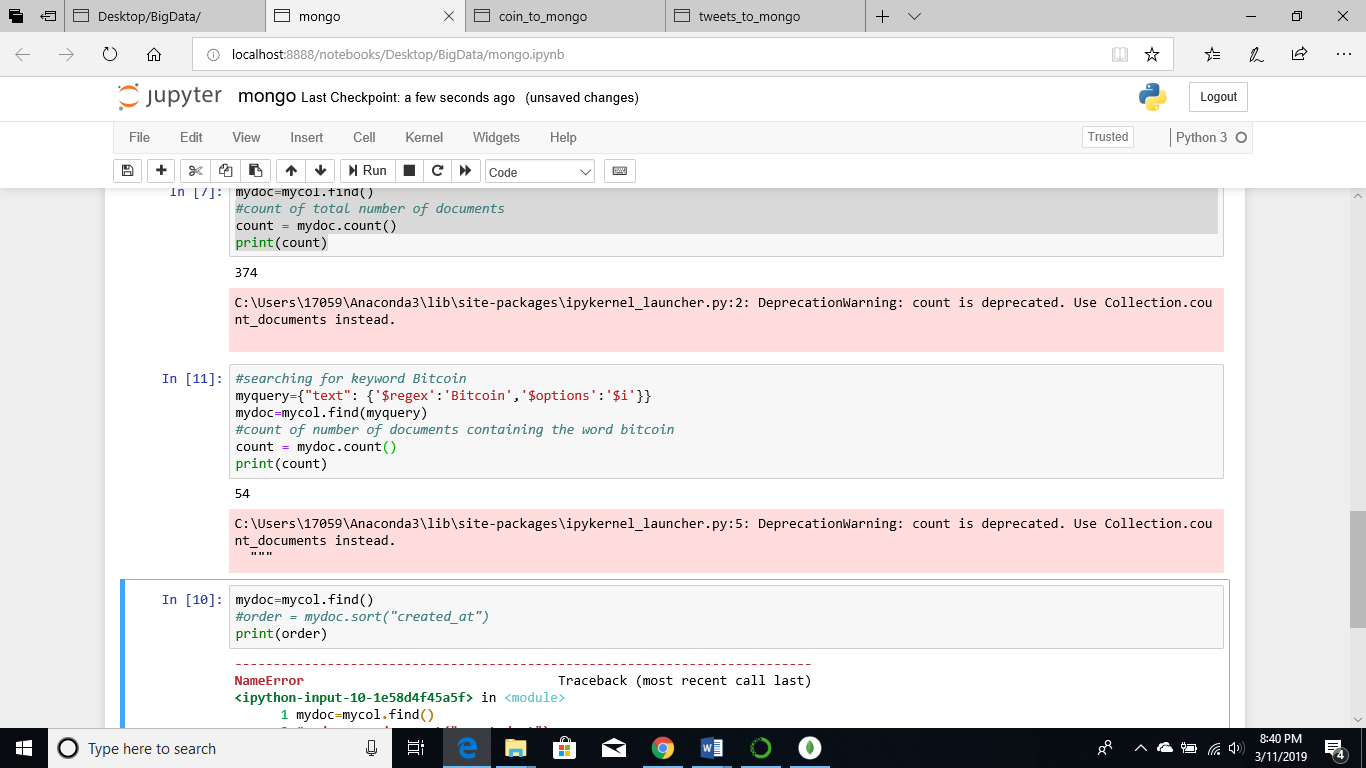


1. **Aggregation sums and count**

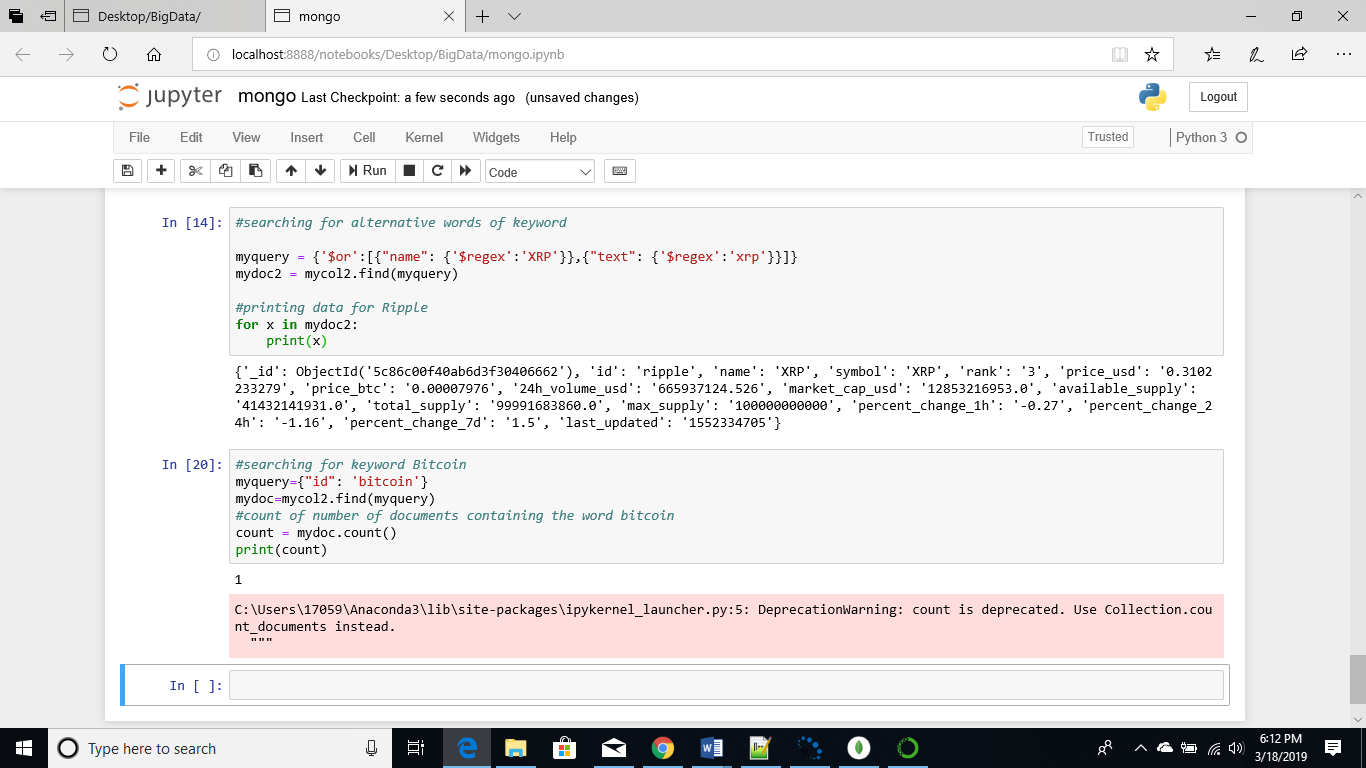
In the below query with the count() function the total number of documents present in the collection can be validated.



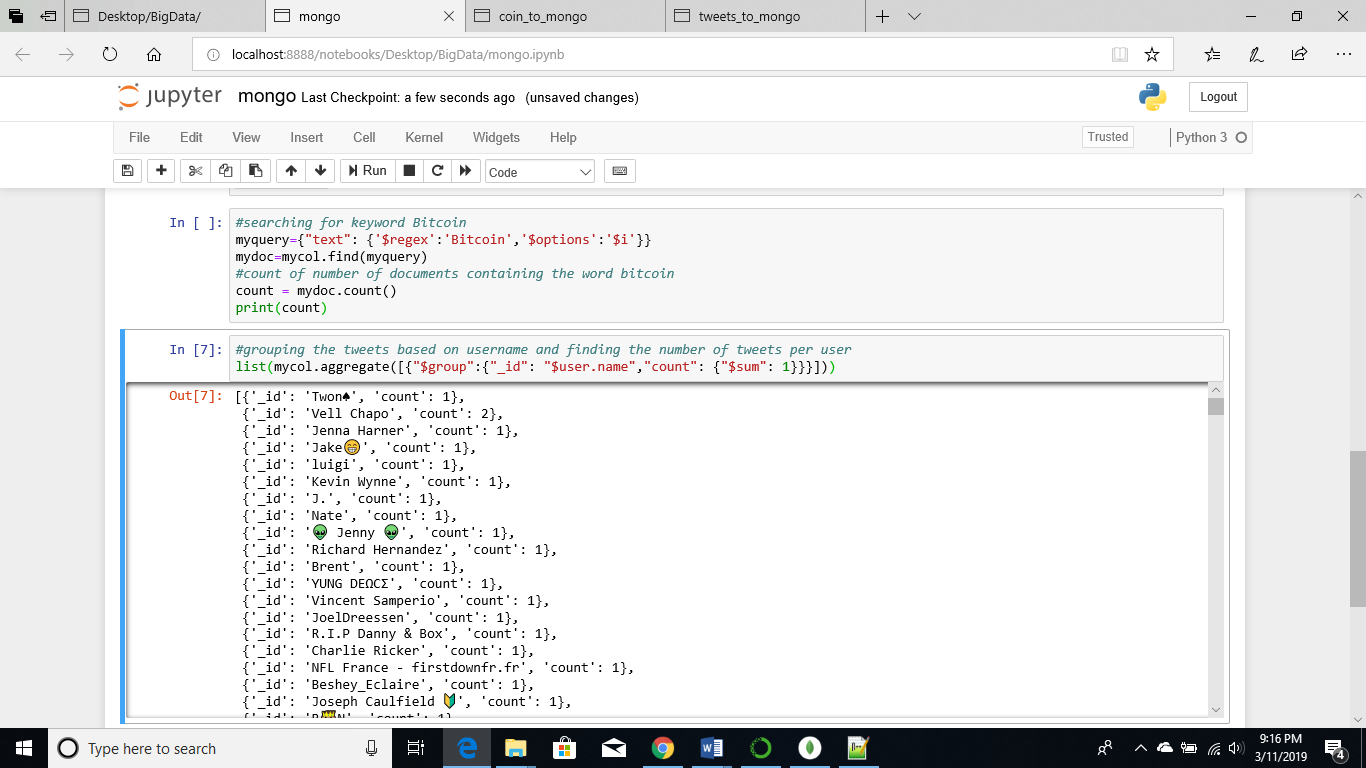
There are 54 documents containing the term “Bitcoin”

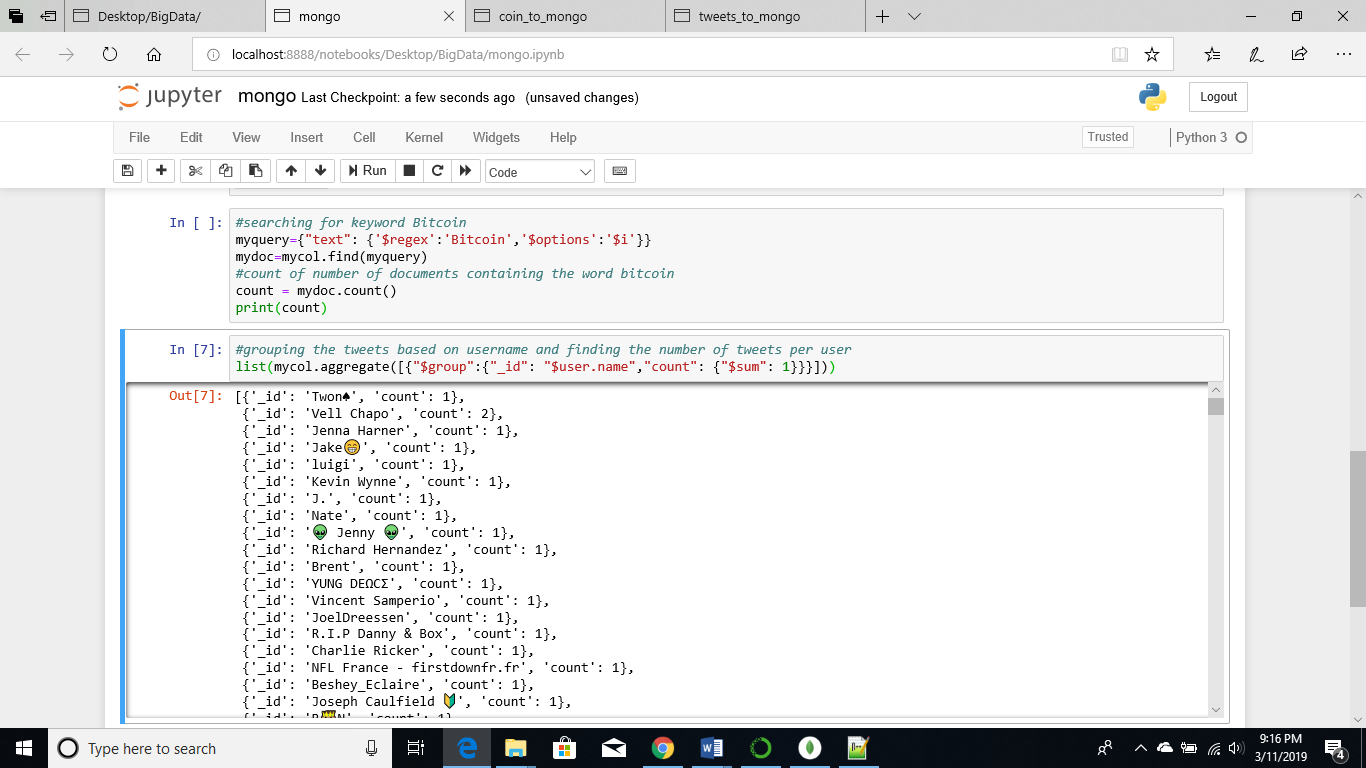


For coinmarketcap data the id field was searched for “bitcoin”

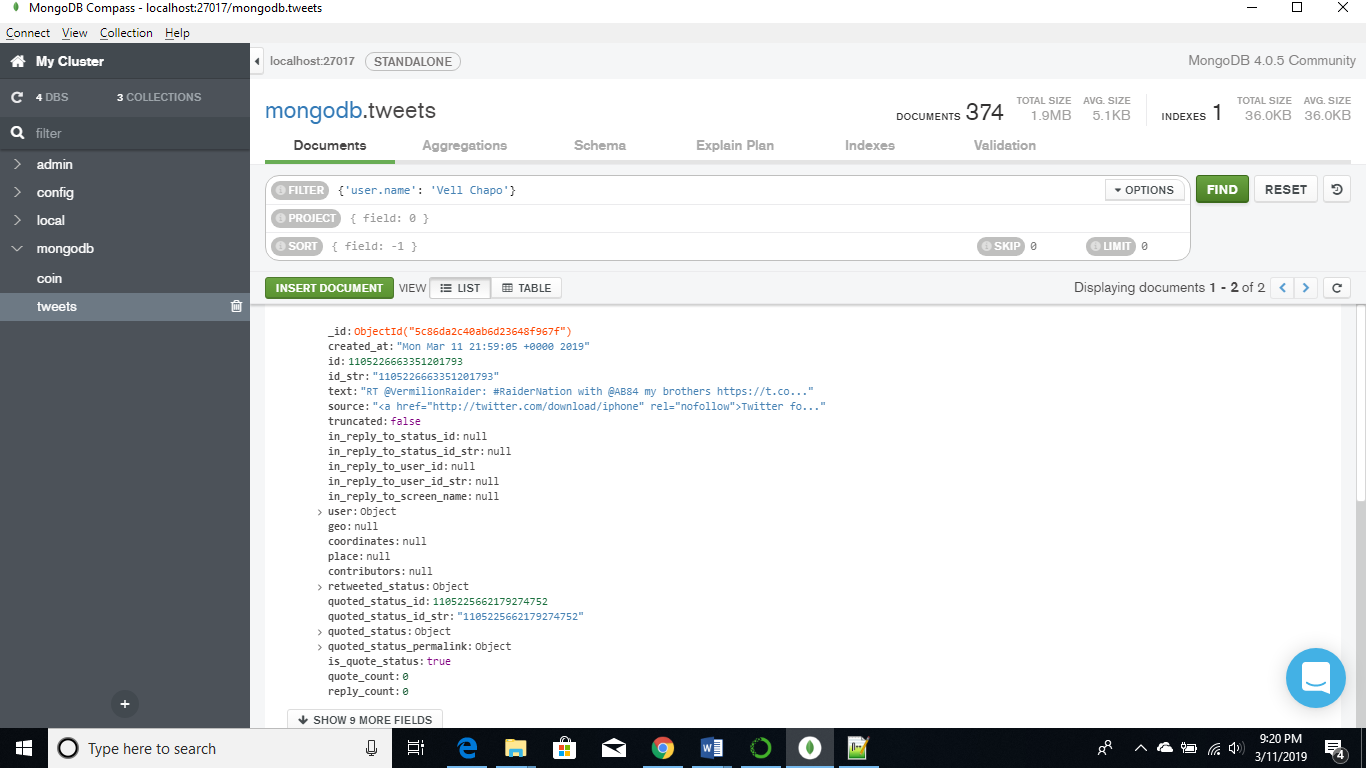


By grouping the tweets based on username we can get the count of the number of tweets posted by the users. From, the screenshot below we can see that “Vell Chapo” has 2 tweets.

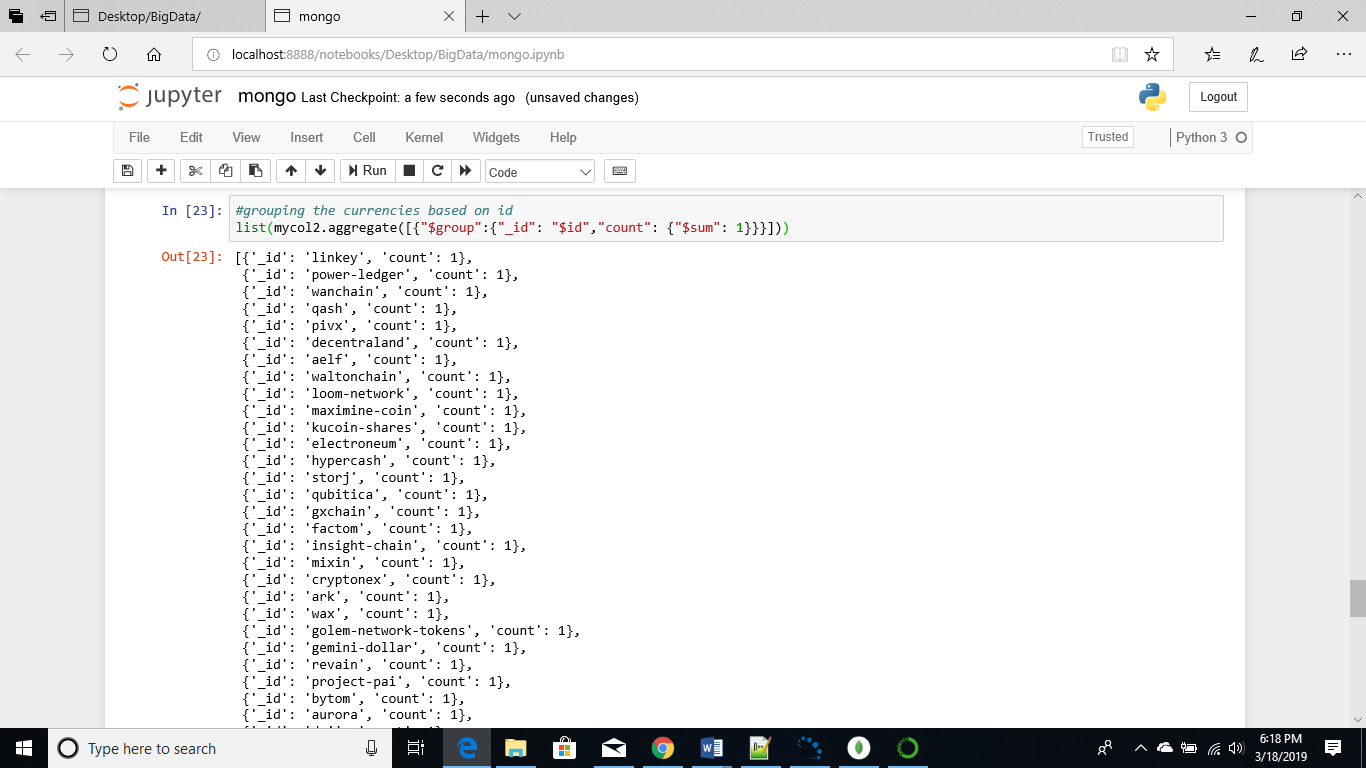




After checking MongoDB for the same user, we can see that there are 2 documents for “Vell Chapo”

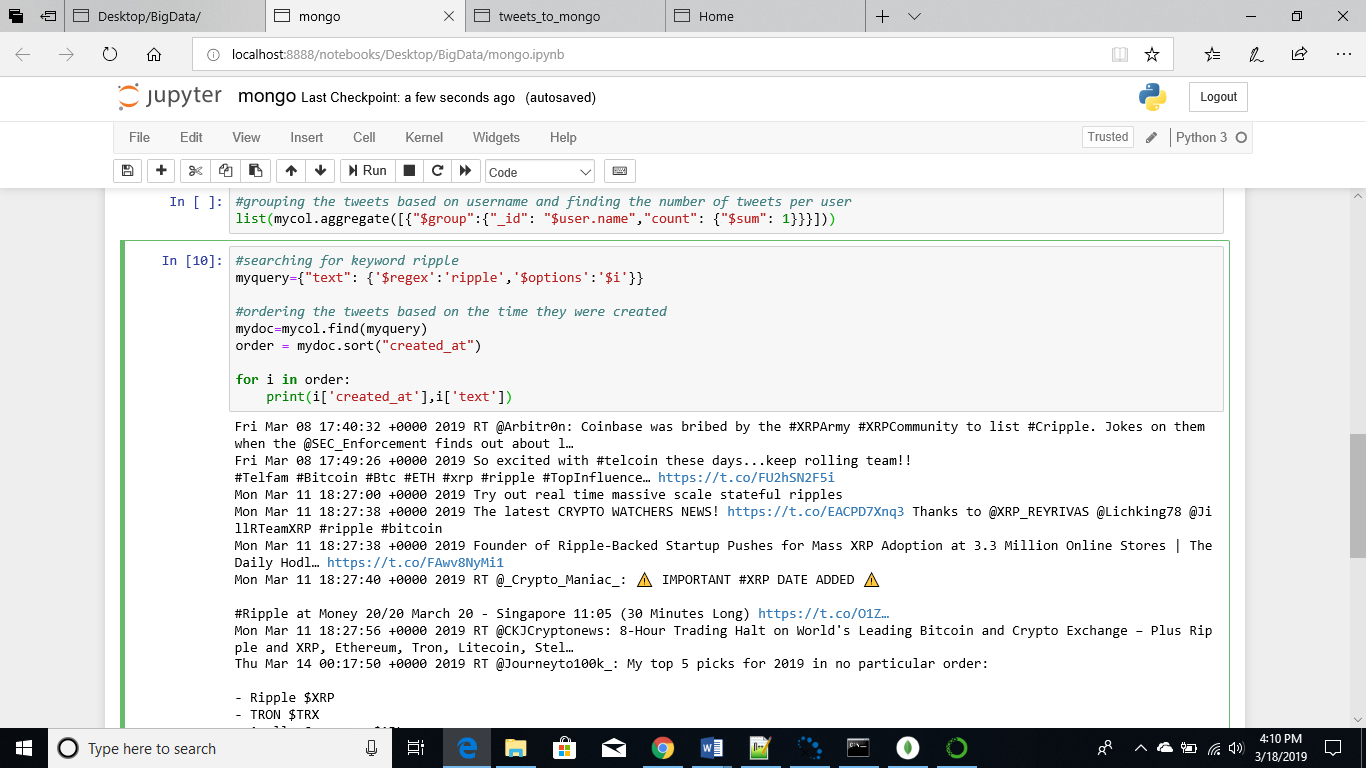


The query below aggregates the currencies from coinmarketcap based on “id” and counts how many times each has occurred

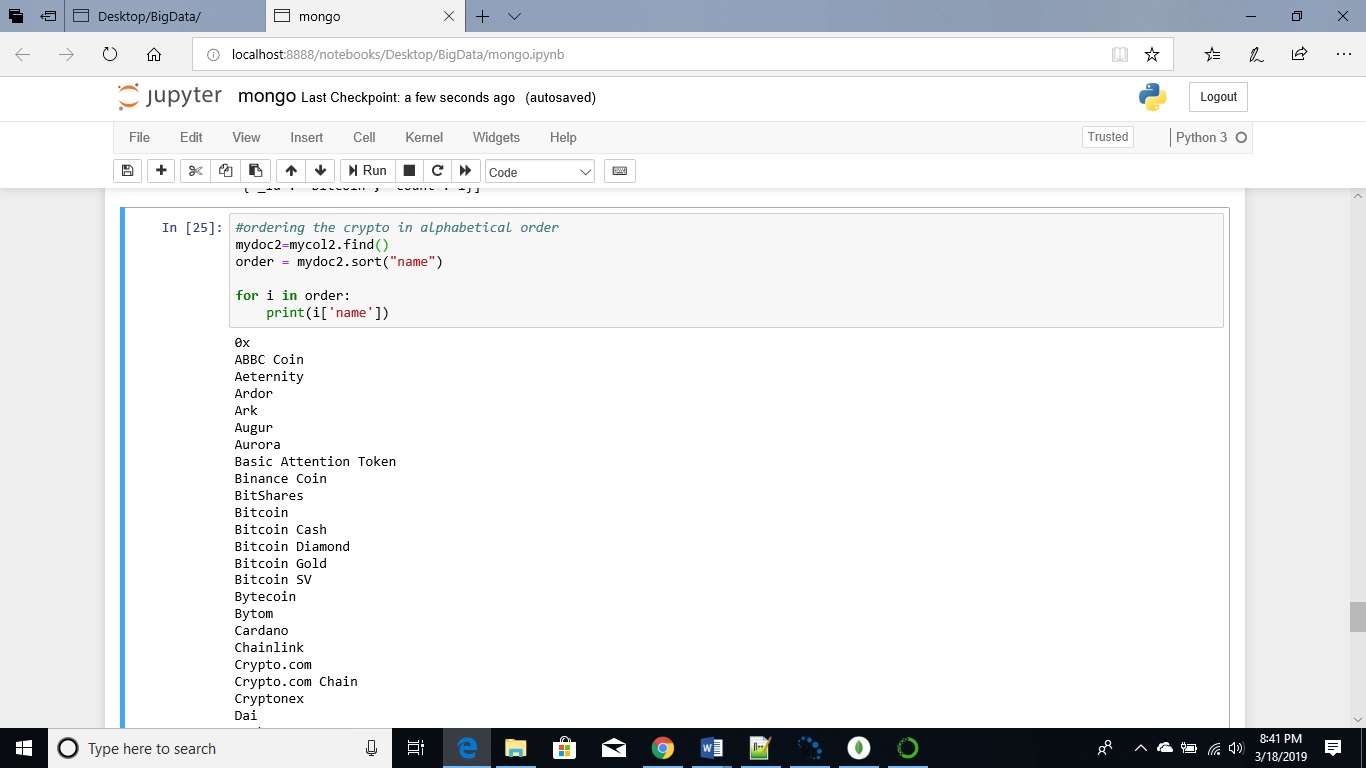


1. **Conditions and order**

The query below orders the tweets based on the time they were posted and prints the fields created\_at and text



The query below sorts the names of the cryptocurrencies from the coin collection in alphabetical order



1. **Code complexity and query time**

Pattern or keyword searches weren’t too complex in MongoDB mainly because of the “$regex” operator, by specifying the field and keywords code could be easily written to query the database of tweets. Since very few documents were extracted the queries did not take much time to fetch results, they were instantaneous. However, in real world scenarios for actual data analysis the datasets are incredibly large, and queries will take a much longer time to execute.

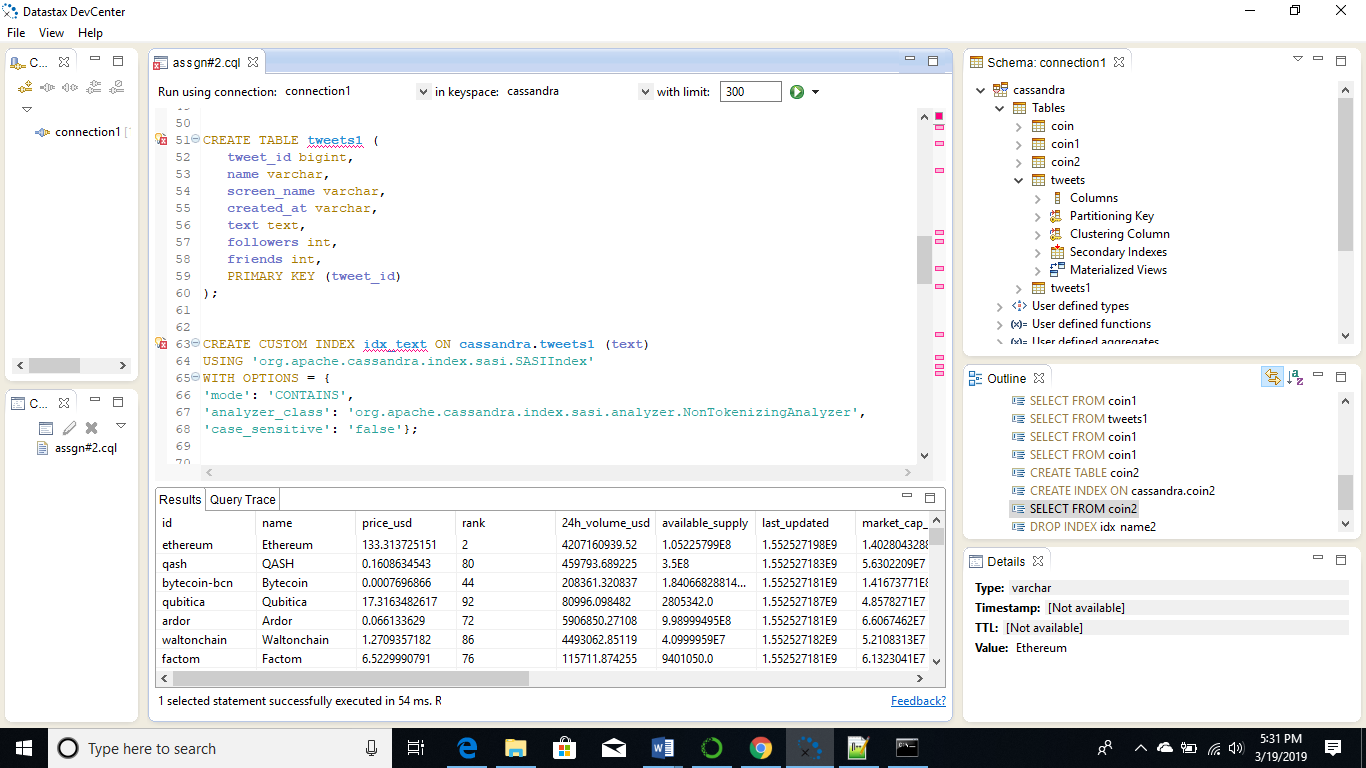
Python script for MongoDB queries: mongo.ipynb

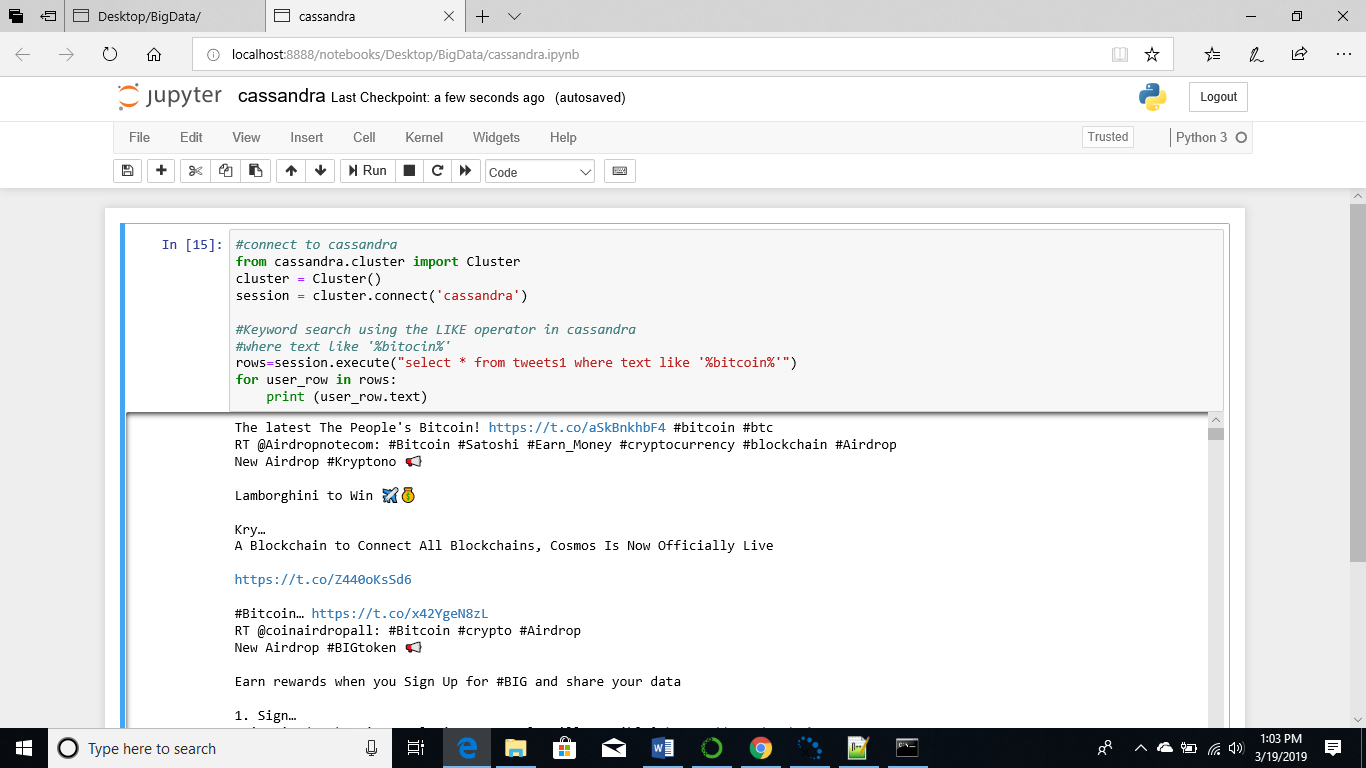
**Data Exploration in Cassandra**

First, we connect to Cassandra in Python using cluster from the cassandra library and calling the cluster() function

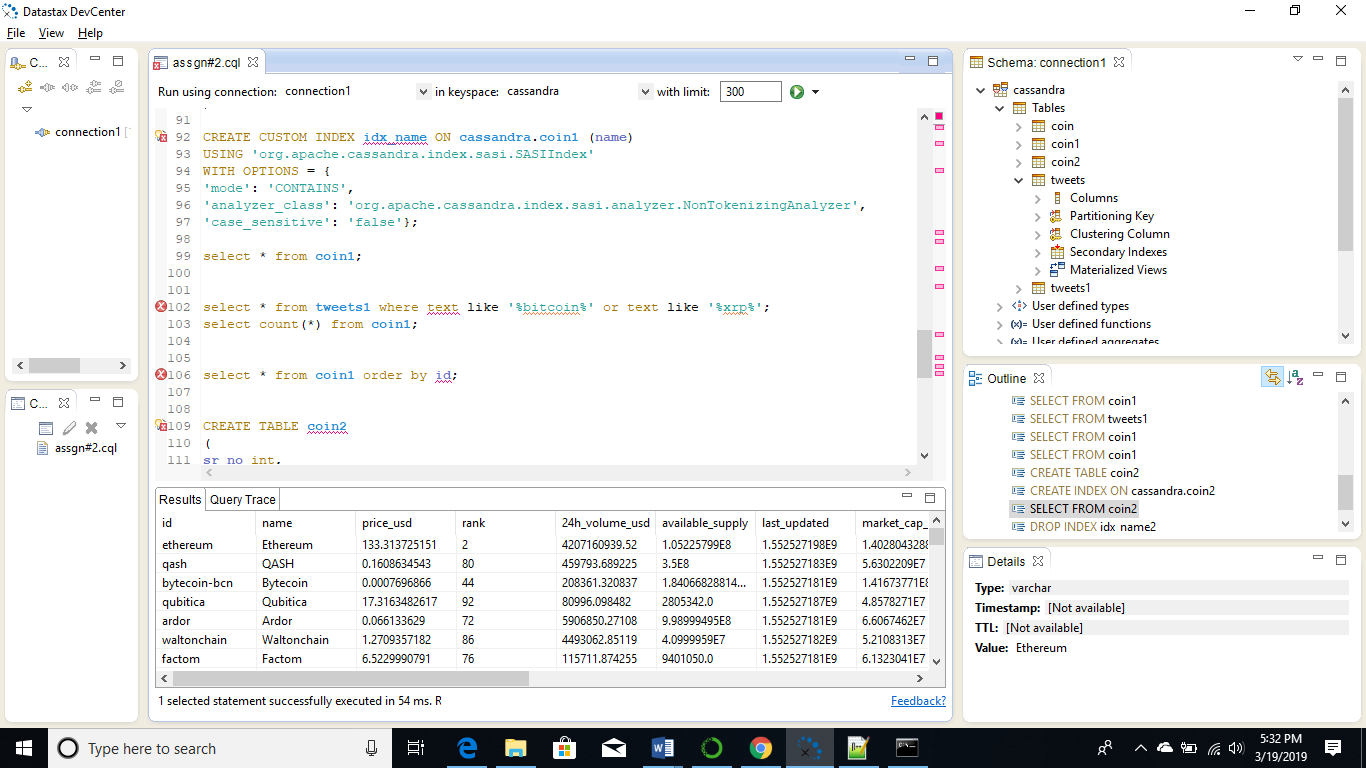
1. **Searching for keywords**

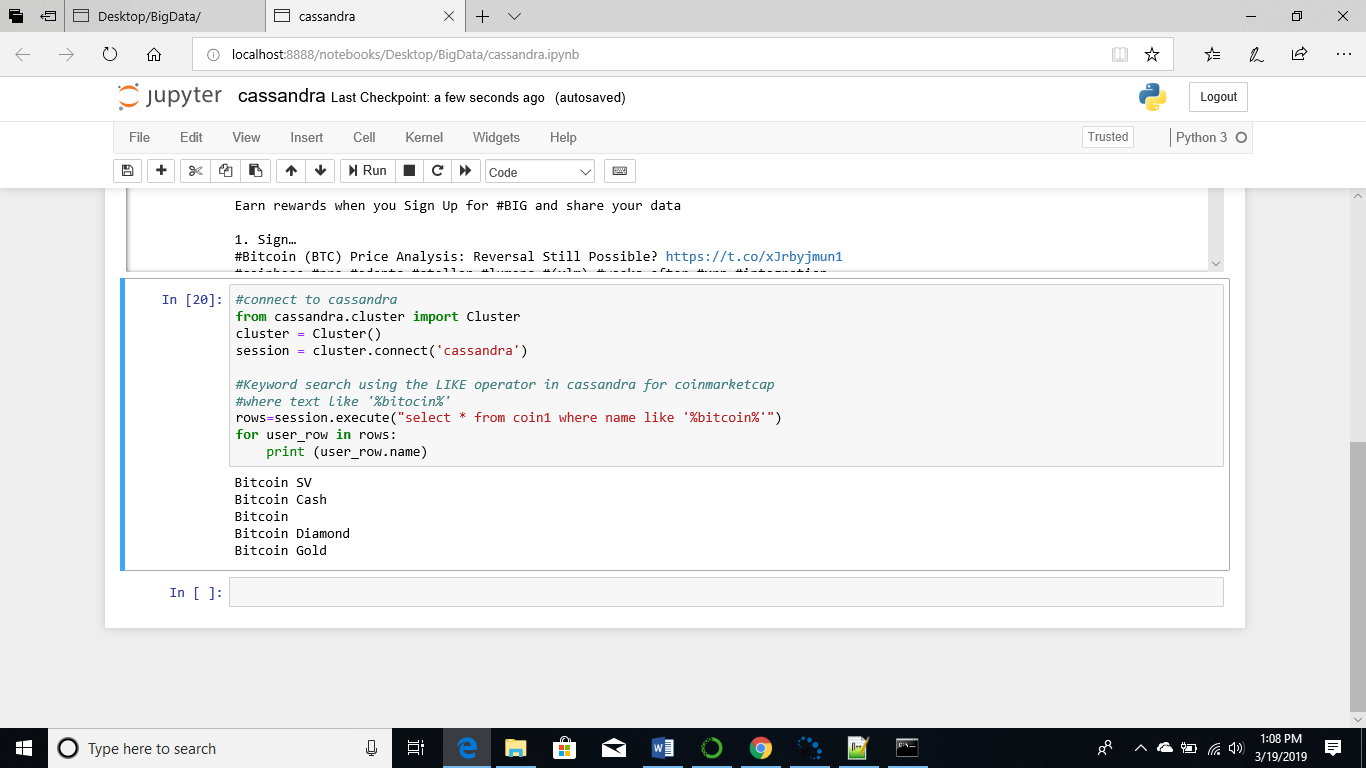
In order to perform a keyword search in Cassandra the column on which the search is done has to be indexed first, then using the LIKE operator a query can be written in cql. I indexed the “text” column of the tweets table and then searched for tweets that contained the term “Bitcoin”.





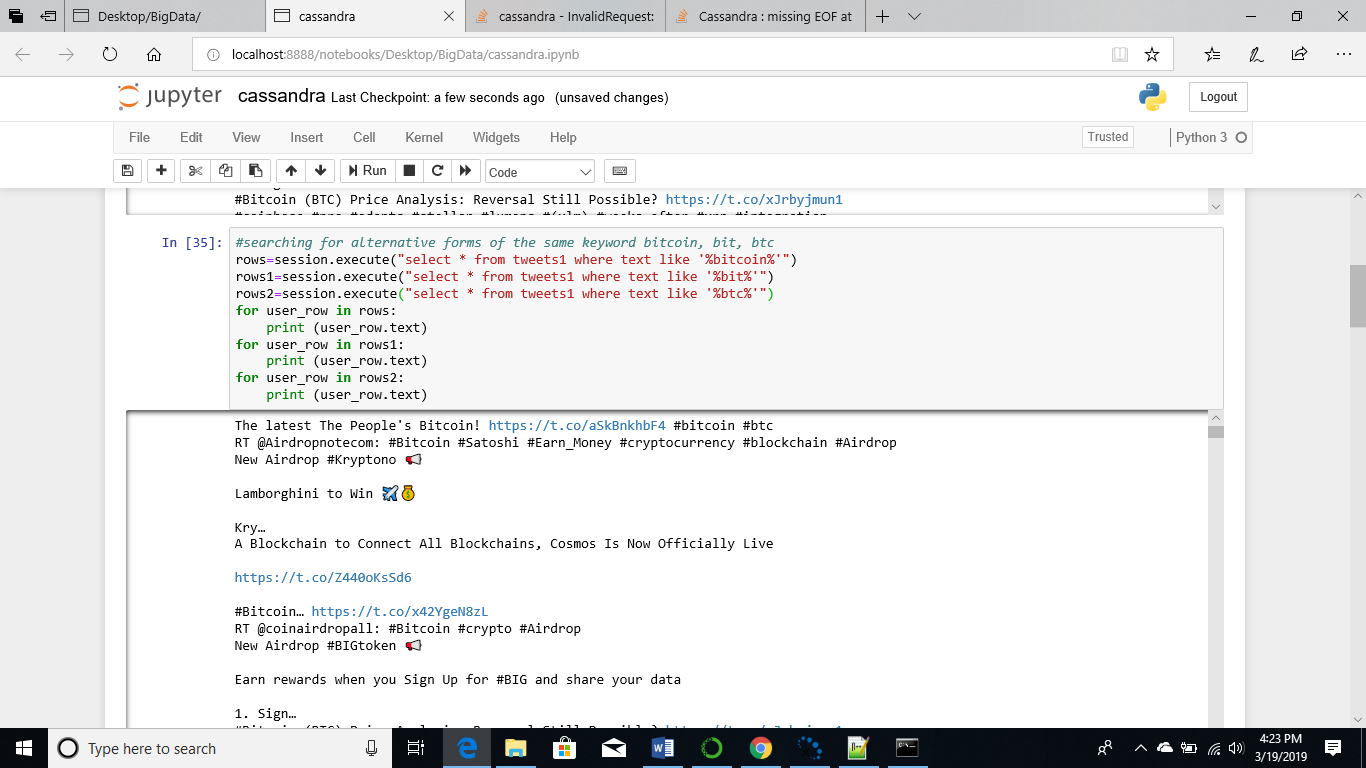
Similarly, for data stored on the coin table the name column was indexed and a query using the like operator was used to search for currency that have the term Bitcoin, 5 currencies were returned.

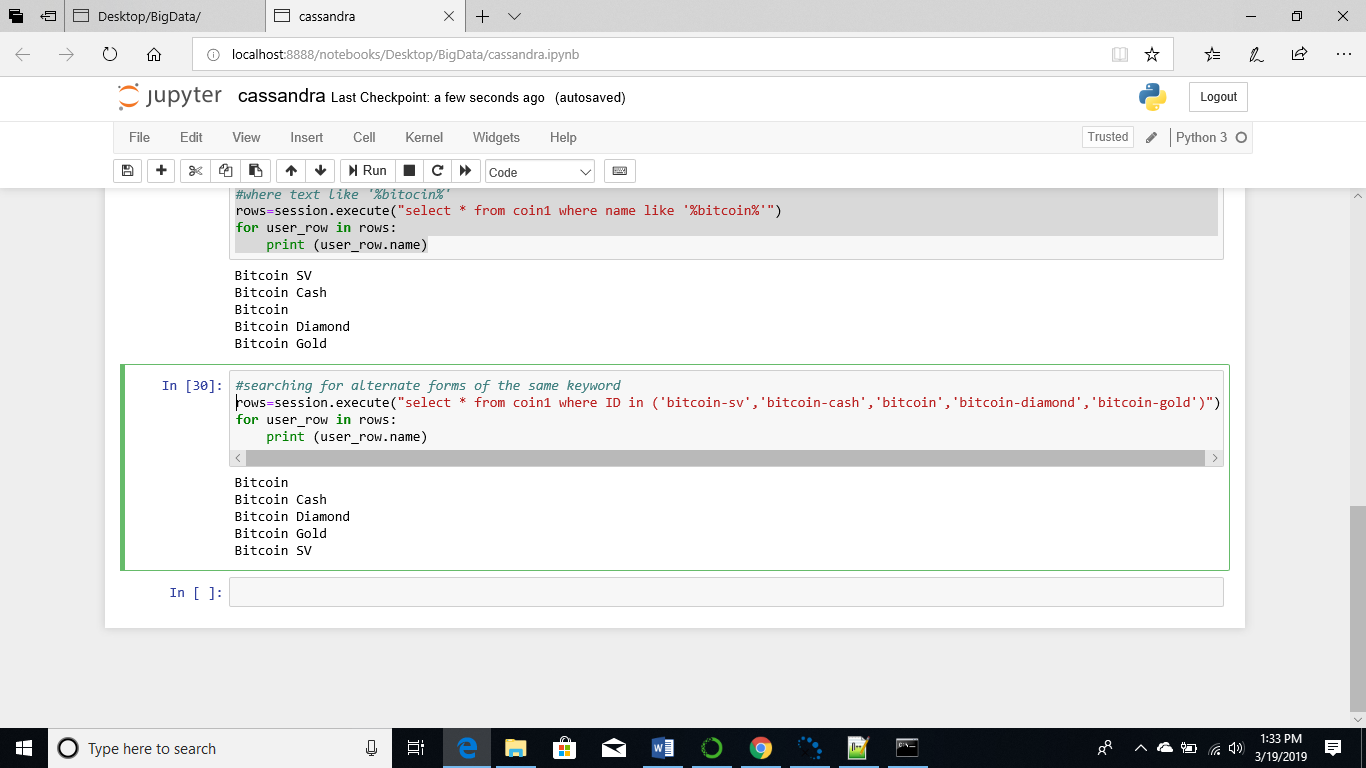




1. **Searching for alternate forms of the same keyword**

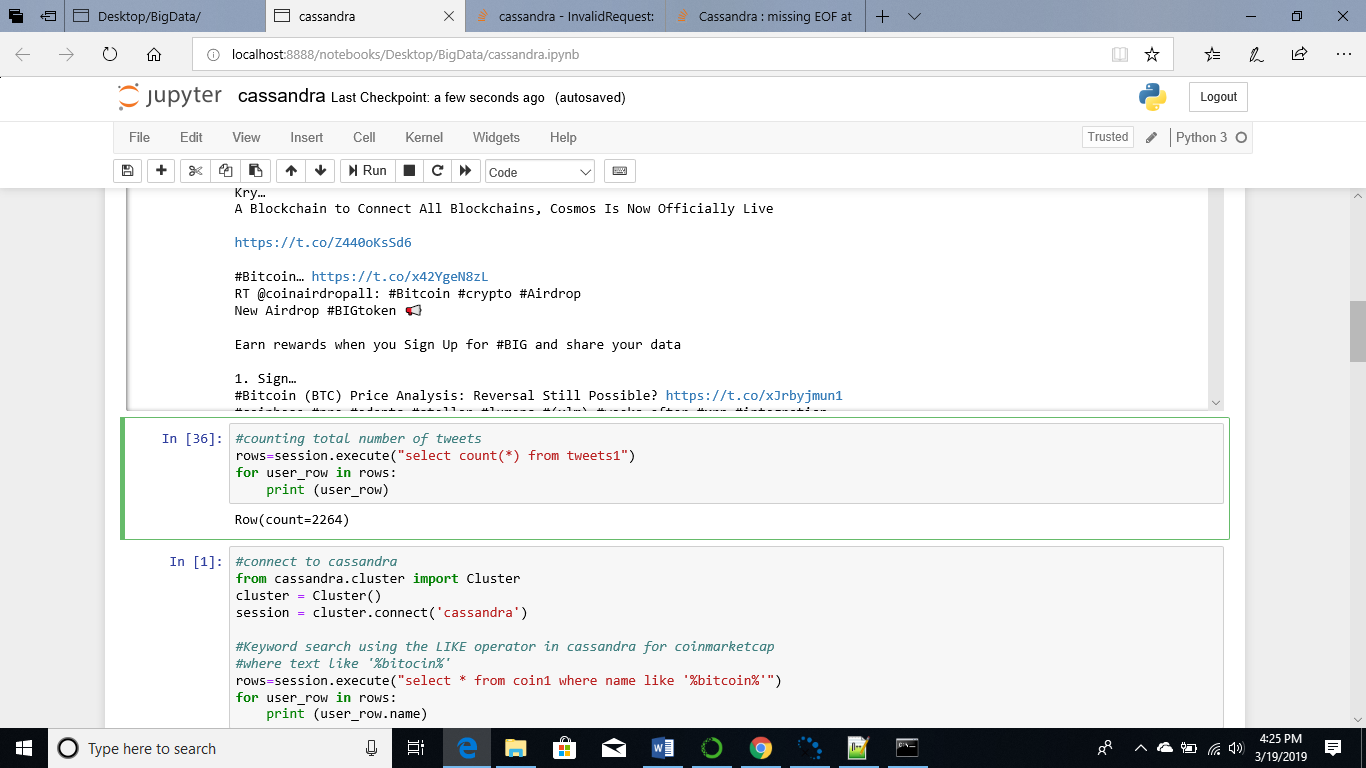
It was difficult to search for alternate forms of the same keyword. So, individual queries had to be written to look for different keywords such as “Bitcoin”,” bit” and “BTC” from the tweets.



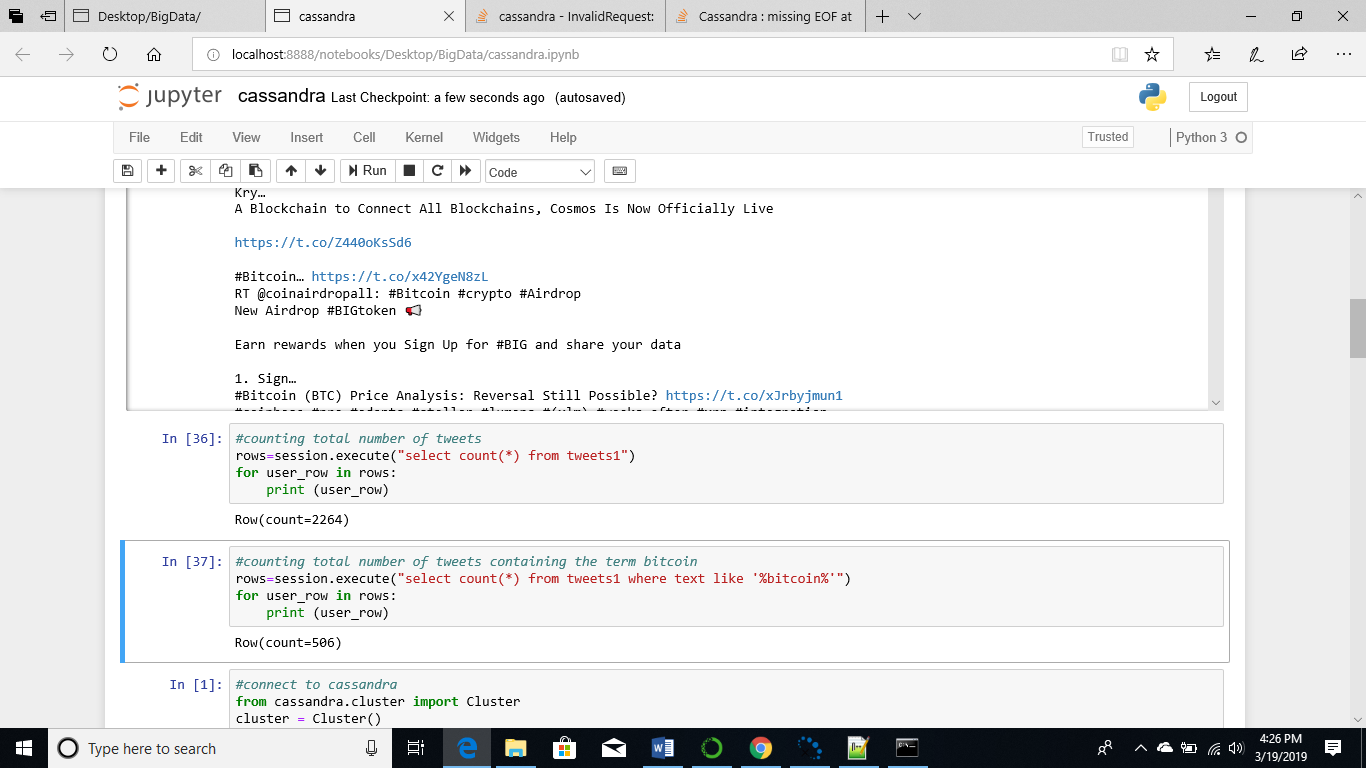
I was also able to use the IN operator to search for specific terms from the table coin 

1. **Aggregation sums and count**

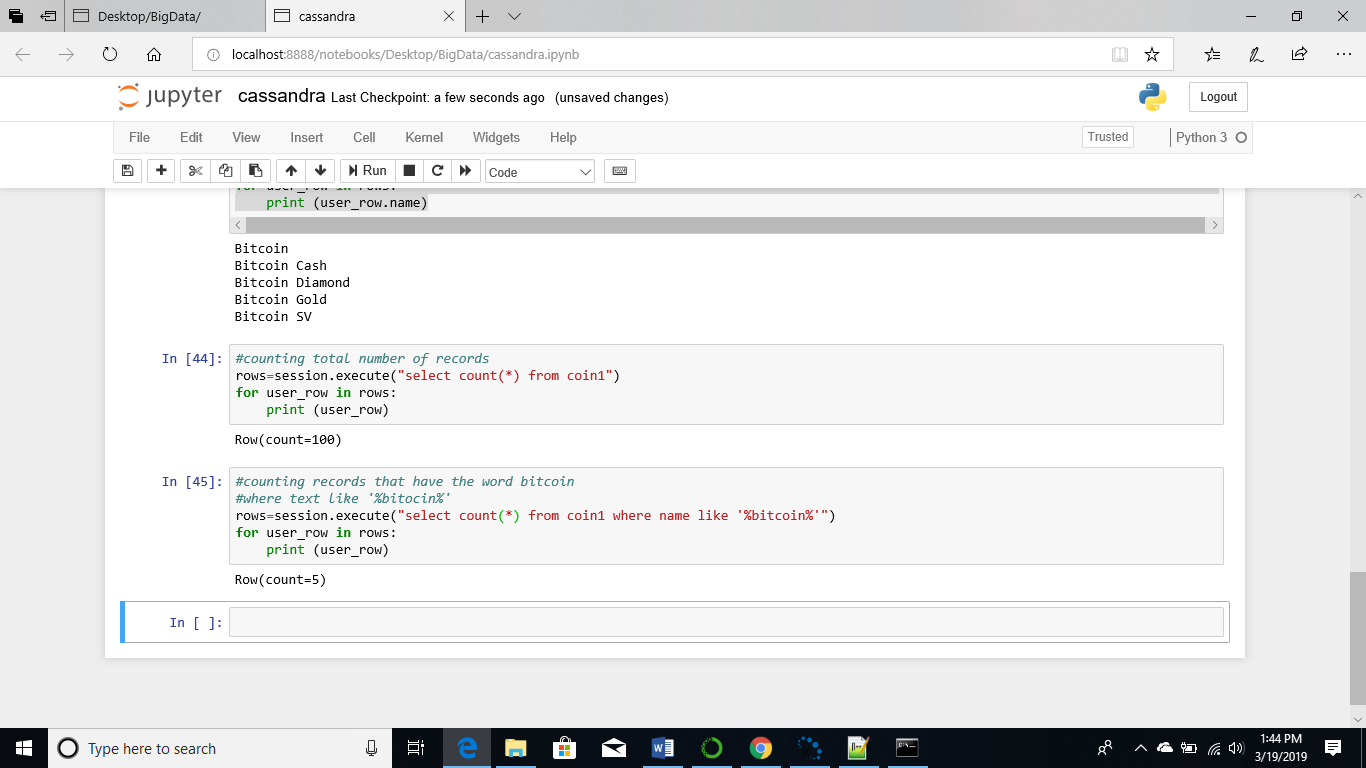
The query below returns the count of the total number of tweets stored, in total 2264 tweets were stored on the table tweets



The query below returns the count of all the tweets that contain the term “Bitcoin”, 506 tweets contained the term Bitcoin



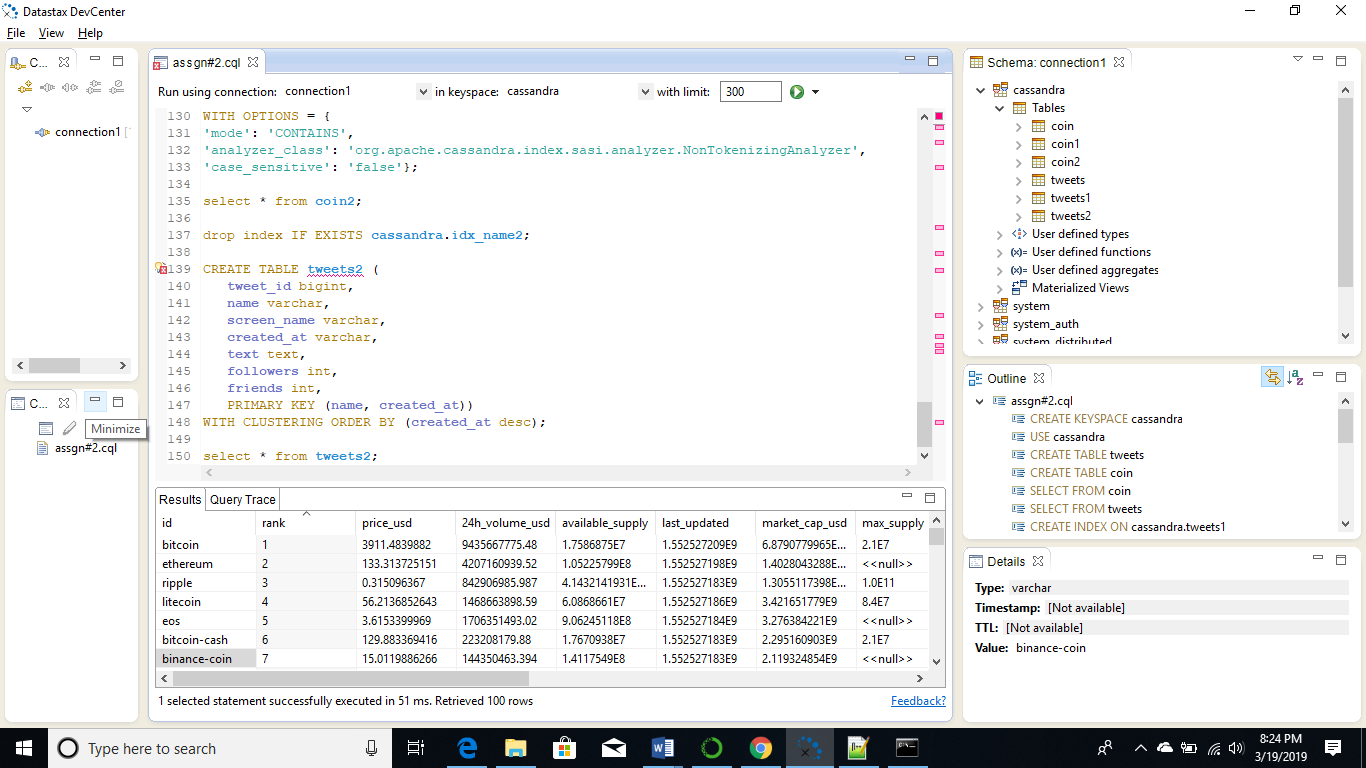
The top 100 cryptocurrencies were retrieved from coinmarkercap and stored on the coin1 table on Cassandra. Out of those currencies 5 of them contain the term Bitcoin in their name

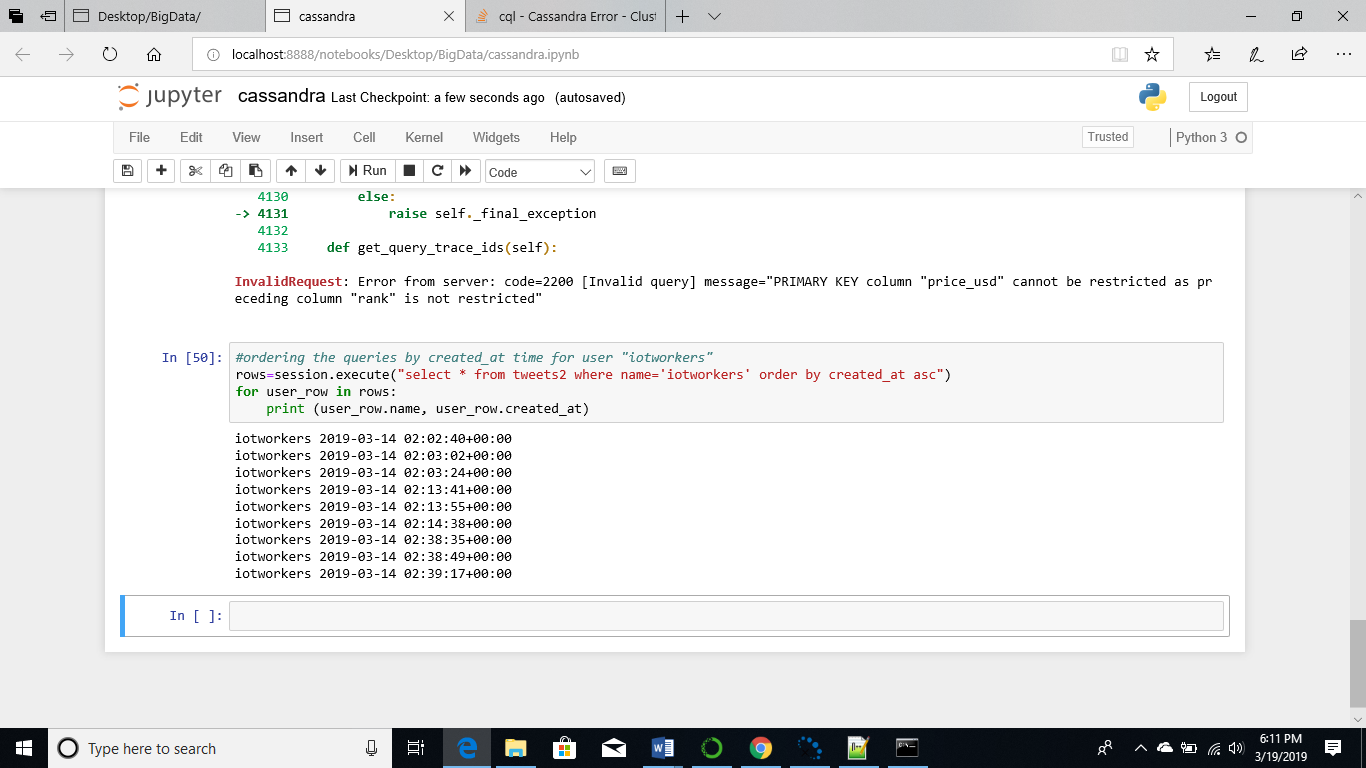


1. **Conditions and order**

In the query below the username “iotworkers” is filtered and the tweets are arranged in increasing order of timestamp. In Cassandra order by is not supported with secondary index, therefore searching for patterns and ordering the timestamp was not possible

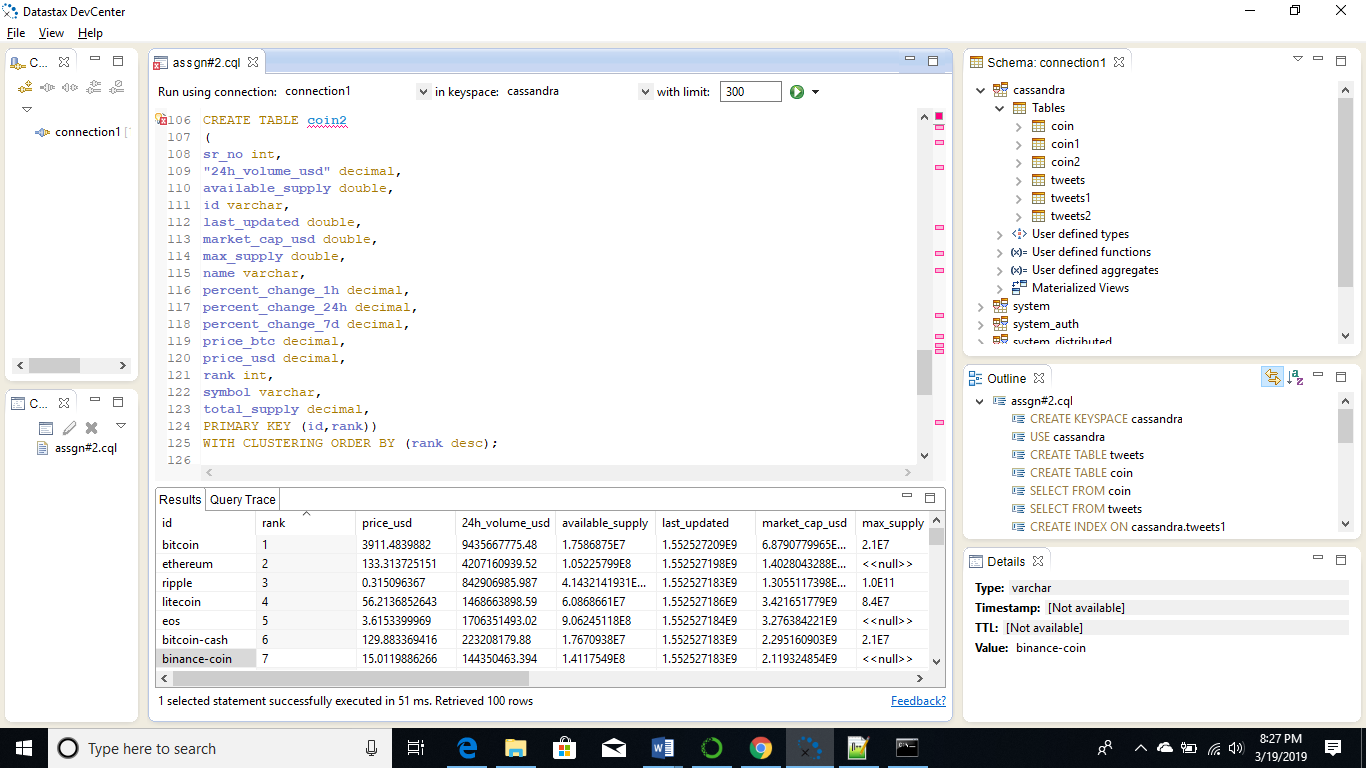
In order to be able to sort with the created\_at column the created\_at column had to be specified as primary key and then with clustering order by.

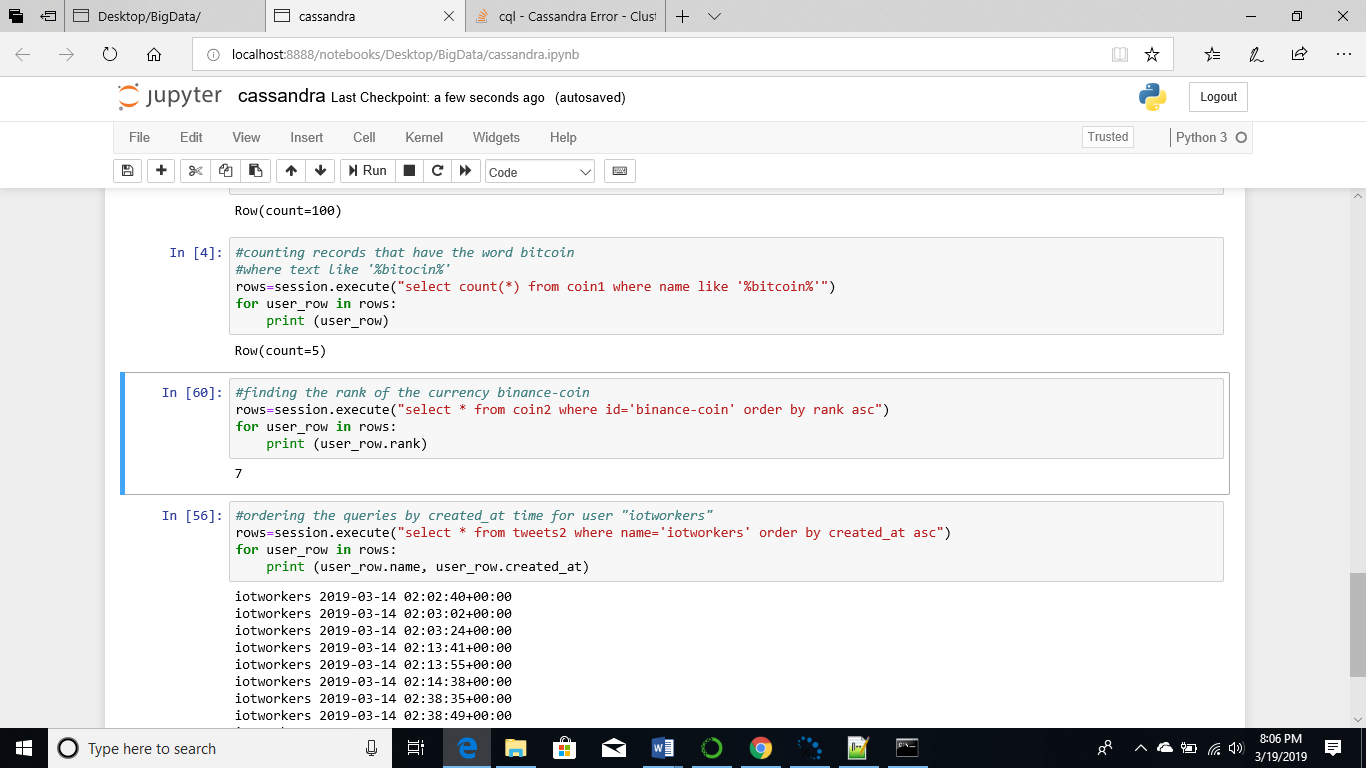




The data from coin market cap consist of only 100 records, however they all the records could not be ordered because Cassandra only supports ordering after filtering the data with an IN or = clause and there was not way to filter group of cryptocurrencies

Similarly, for the coin2 table rank was specified as a primary key and then in the clustering order by statement





1. **Code complexity and query time**

Since the syntax of CQL is like SQL it was quite easy to write queries. However, Cassandra is not a relational database management system it was difficult to query non-key columns and perform task such as pattern searching and sorting. In order to be able to search for patterns the columns had to be indexed first and to sort columns they must be included as primary keys. Since, only a few records were stored the queries did not take much time to execute, however when dealing with real-time data query might take much longer to run.

Python script for Cassandra queries: cassandra.ipynb

**Data Exploration Challenges**

It was relatively easy to store and query data in MongoDB, on the other hand performing the same operations in Cassandra were quite challenging. Primarily because Cassandra appears as a relational database but behaves differently. The first challenge was performing a keyword or a pattern search, the “LIKE” clause could not be applied to columns that are not indexed, to resolve this I indexed the text column of tweets and the name column of the coin table, thus creating 2 secondary indexes. This simplified the querying enabled me to search for keywords in the tweets.

The other major challenge was sorting the data based on a column, in order to do this the column had to be specified as a primary key and then with clustering specify the column which should be used for ordering. It was not possible to order all the data of one table.

**Conclusion:** It was easier to store and retrieve data from MongoDB, since MongoDB can store unstructured data all the information gathered was in JSON format and stored directly on MongoDB in JSON format. Pattern or keywords searches were quite easy to perform because of the regex operator, even searching for multiple forms of the same keyword could be easily done with the or operator. On the other hand, Cassandra is not a very flexible database and there were lots of challenges right from data gathering, storage and retrieval. The data gathering had to be done through .csv files and then imported through cqlsh, while extracting and importing the data care had to be taken of the encoding to ensure that data could be properly loaded into the tables. In addition, running queries on non-indexed columns was a major challenge especially for while performing keyword searches and sorting data.