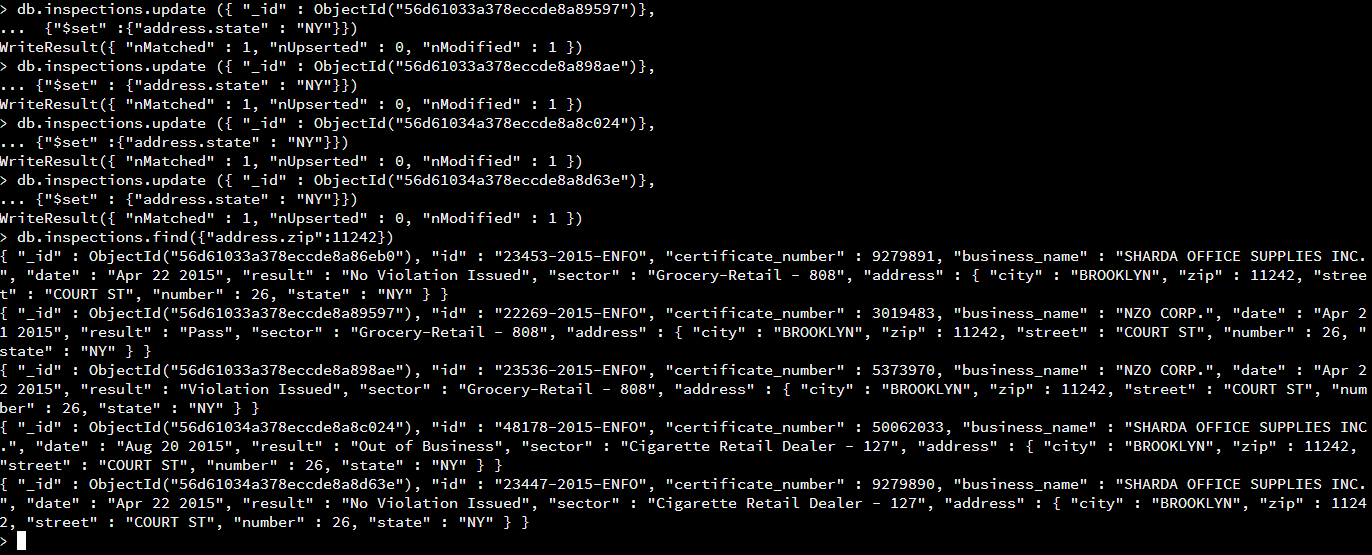
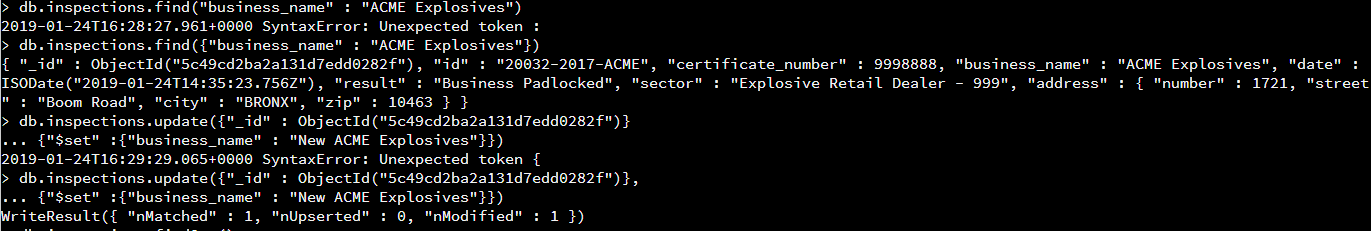
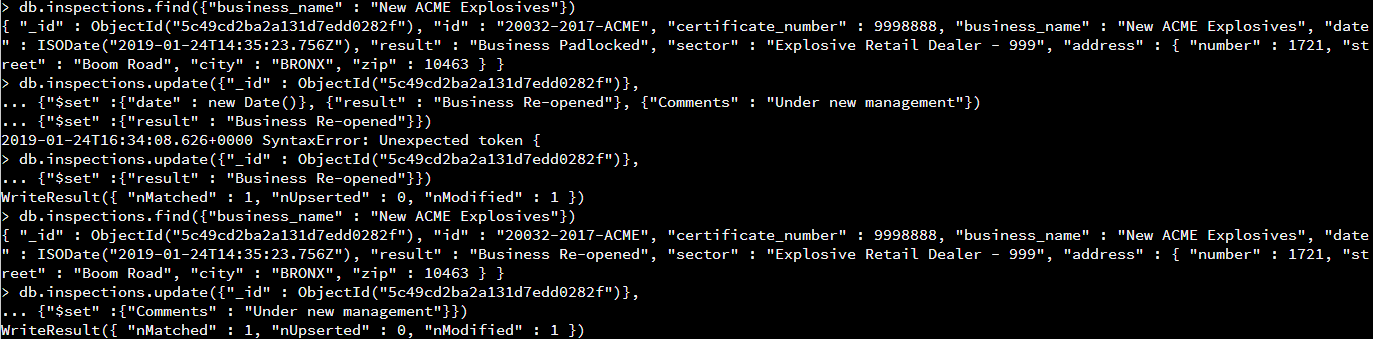
This artifact was done in a previous class of mine, it shows some of the work that I had done using MongoDB, Pymongo on Codio. Everything that I did was done by coding into the program which is seen at the bottom, using the code and using a series of different methods I was able to produce various results, such as; using the update and $set method to update businesses state to NY. I was able to delete a business from the database.

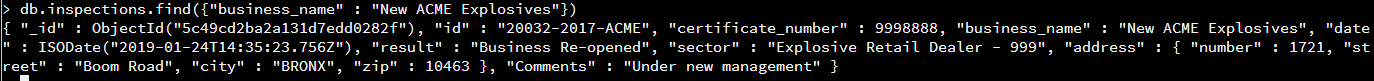
I feel that this artifact is justified in being included as it does show that I do have an understanding of how coding works, and how it works with different methods to reach the desired result and works within the database.

All businesses with the zip code “11242” updated to include the state of “NY” using the .update and $set method

ACME explosives was updated to include the new name of “New ACME Explosives which went from a business that was padlocked to a business that was re-opened and the comment added that it was under new management using the .update function.

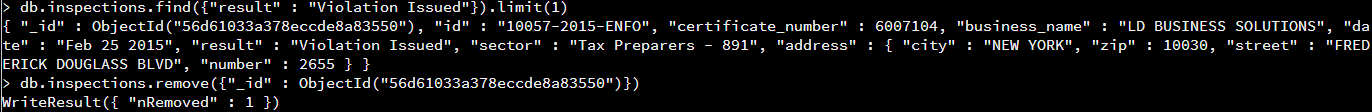




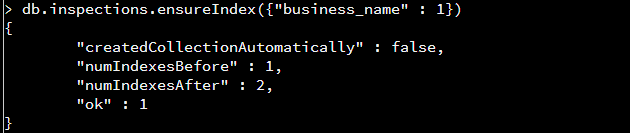


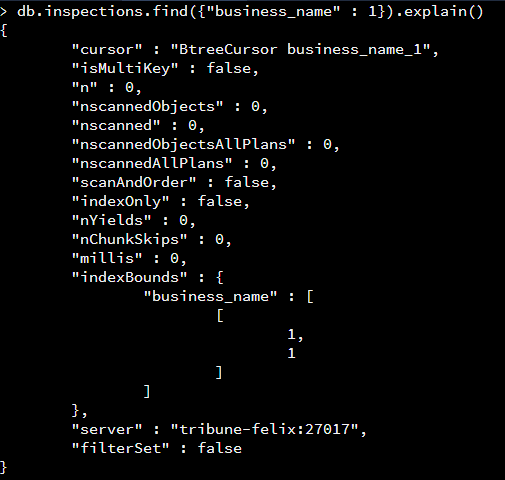
The first document on record was the “LD BUSINESS SOLUTIONS” this business was deleted using the .remove function.

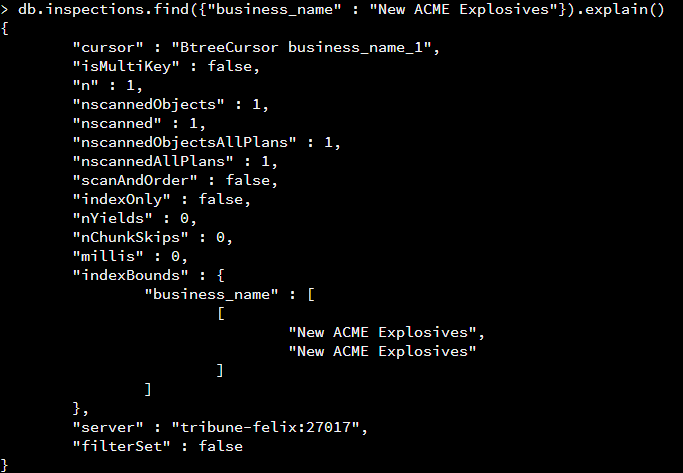




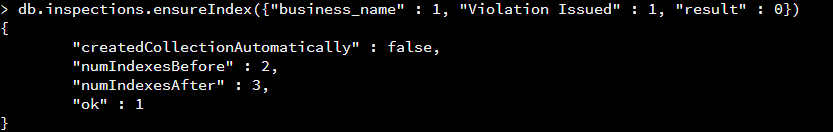
Index was created using the key “business\_name” using the .ensureIndex function, this was then found by using the .find function along with the .explain() function which shows the index and explains the usage.



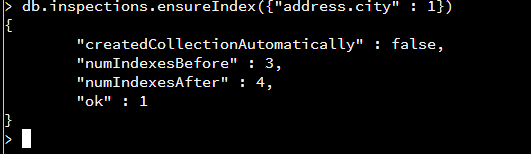




Compound index was created to show the business name with the result of a violation that had been issued.



Index was created to help improve the performance of finding a business by the companies city location.



import json

from bson import json\_util

from pymongo import MongoClient

from bottle import abort

import pymongo

connection = MongoClient('localhost', 27017)

db = connection['City']

collection = db['inspections']

def insert\_document(document):

try:

result=collection.save(document)

except ValidationError as ve:

abort(400, str(ve))

return result

def get\_document(document, name):

try:

query = {document: name}

result=collection.find\_one(query)

except Exception as ve:

return False

return result

def update\_document(document, business, updateData):

try:

result = collection.update({"id":"inspections"},{ '$set': updateData }, upsert=False, multi=False)

except Exception:

abort(400, str(ve))

return result

def main():

myDocument = {"id" : "inspections", "business\_name" : "InspectionsRus", "date" : "February 8 2019", "result" : "New Business", "sector" : "City Inspectors", "address" : {"city" : "Richmond", "zip":22235, "street" : "First Ave", "state" : "Virginia"}}

print insert\_document(myDocument)

print get\_document("id", "inspections")

updateData = { "business\_name": "VA Inspectors" }

print update\_document("id", "inspections", updateData)

main()

#!/usr/bin/python

import json

from bson import json\_util

import bottle

from bottle import route, run, request, abort

import milestone

import pymongo

# myclient = pymongo.MongoClient("mongodb://localhost:8080/")

# mydb = myclient["mydatabase"]

# mycol = mydb["customers"]

mydict = {"id" : "10011-2017-TEST","certificate\_number" : 9278833,"business\_name" : "ACME TEST INC.","date" : "Feb 20 2017","result" : "No Violation Issued","sector" : "Test Retail Dealer - 101"}

# set up URI paths for REST service

@route('/currentTime', method='GET')

def get\_currentTime():

dateString=datetime.datetime.now().strftime("%Y-%m-%d")

timeString=datetime.datetime.now().strftime("%H:%M:%S")

string="{\"date\":"+dateString+",\"time\":"+timeString+"}"

return json.loads(json.dumps(string, indent=4, default=json\_util.default))

@route('/create', method='POST')

def put\_document():

data = request.json

if not data:

abort(400, 'No data received')

#entity = json.loads(data)

if not data.has\_key('id'):

abort(400, 'No id specified')

try:

milestone.insert\_document(data)

except NameError:

abort(404, 'No parameter for id %s' % id)

if not string:

abort(404, 'No id %s' % id)

return "True"

@route('/read', method='GET')

def get\_document():

b\_name = request.query.business\_name

result = milestone.get\_document("business\_name", b\_name)

return json.dumps(result, indent=4, default=json\_util.default)

return "true"

@route('/update', method='GET')

def update\_document():

id = request.query.id

result = milestone.update\_document("result", id, updateData)

return json.dumps(result, indent=4, default=json\_util.default)

if \_\_name\_\_ == '\_\_main\_\_':

#app.run(debug=True)

run(host='localhost', port=8080)

#from bottle import route, run