CLASS: CSE7345

Name: Randeep Hanspal

SMUID: 47812509

Quest 7 MongoDB2

Part A

In [1]:
```
import pymongo
import json
import pprint
```

In [2]:
```
from pymongo import MongoClient
```

In [3]:
```
client=MongoClient('mongodb://rhanspal:ar43hRn4@smgo7db01.smu.edu:27017/rhanspaldb')
```

In [4]:
```
db=client.rhanspaldb
```

In [5]:
```
#Imported zipcodes.states.gps.csv to create zipstates collection
#print all the collections
db.collection_names()
```

Out[5]:
```
[u'products',
 u'system.js',
 u'test',
 u'zorro',
 u'blog',
 u'csvdata',
 u'zipstates']
```

In [6]:
```
query = {'zip_code': 10463}

#use projection to display city and state without _id
projection={'_id':0,'city': 1, 'state': 1 }
```

In [8]:
```
cur=db.zipstates.find(query,projection)
for doc in cur:
    print ("City and State for zipcode 10463 is {}".format(doc))

#dictStats contains result of execution Stats
dictStats = cur.explain()["executionStats"]

#storing the result in variables
TotalDocsWithoutIndex=dictStats['totalDocsExamined']
TotalExectionWithoutIndex=dictStats['executionTimeMillis']

#displaying the result without Index
print ("Number of records examined is {} ".format(TotalDocsWithoutIndex))
print ("Execution time in millis is {}".format(TotalExectionWithoutIndex))
```

```
City and State for zipcode 10463 is {u'city': u'Bronx', u'state': u'NY'}
Number of records examined is 42741
Execution time in millis is 39
```

In [13]:
```
#create index for zipcode
indexCreation = db.zipstates.create_index([('zip_code', pymongo.ASCENDING)], u
nique=True)
```

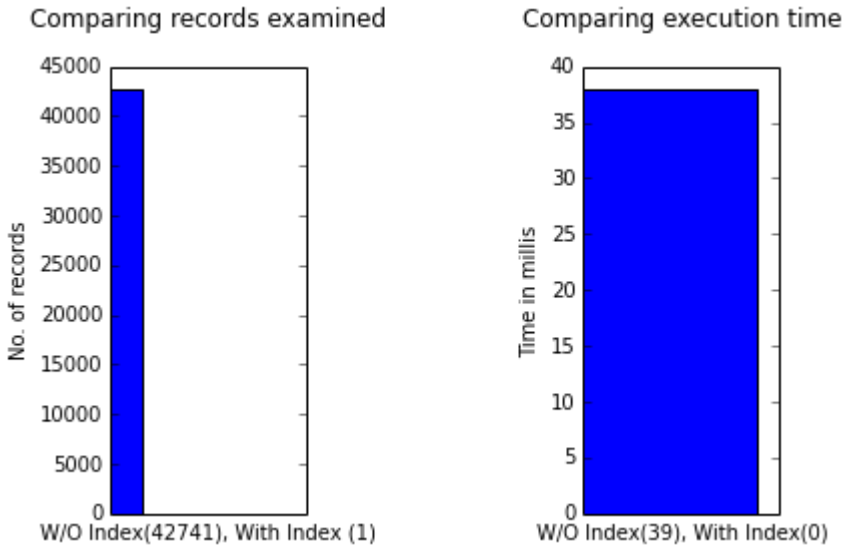
In [14]:
```
#using the above code to run for index
query = {'zip_code': 10463}
projection={'_id':0,'city': 1, 'state': 1 }
cur=db.zipstates.find(query,projection)
for doc in cur:
    print ("City and State for zipcode 10463 is {}".format(doc))
dictStats = cur.explain()["executionStats"]

#storing the result in variables
TotalDocsWithIndex = dictStats['totalDocsExamined']
TotalExectionWithIndex = dictStats['executionTimeMillis']

#displaying the result with Index
print ("Number of records examined is {} ".format(TotalDocsWithIndex))
print ("Execution time in millis is {}".format(TotalExectionWithIndex))
```

```
City and State for zipcode 10463 is {u'city': u'Bronx', u'state': u'NY'}
Number of records examined is 1
Execution time in millis is 0
```

In [109]:
```
#plot the bar graph
import matplotlib.pyplot as plt
%matplotlib inline
#customizing the bar chart
plt.subplot(1,3,1)
methods = ['TotalDocsWithoutIndex','TotalDocsWithIndex']
plt.xlabel("W/O Index(42741), With Index (1)")
plt.ylabel("No. of records")
plt.title("Comparing records examined\n")
plt.xticks([])
#bar chart for execution time
plt.bar(range(len(methods)),[TotalDocsWithoutIndex,TotalDocsWithIndex],width=
0.2)
plt.subplot(1,3,3)
methods1 = ['TotalExectionWithoutIndex','TotalExectionWithIndex']
plt.xlabel("W/O Index(39), With Index(0)")
plt.ylabel("Time in millis")
plt.title("Comparing execution time\n")
plt.bar(range(len(methods)),[TotalExectionWithoutIndex,TotalExectionWithIndex
])
plt.xticks([])
plt.show()
```



Part B. Find Location of Zip Codes

In [113]:
```
#read all the zipcodes from zipcodes.txt
f = open("zipcodes.txt",'r')
zipcode = f.readlines()
zipcode = zipcode[0].split(',')

zipcodestxt=[]
for i in zipcode:
    stripedvalue = i.strip()
    zipcodestxt.append(stripedvalue)
print (zipcodestxt)
```

```
['10463', '06520', '00603', '75225', '90210', '99999']
```

In [127]:
```
#print the column name
print "{:<6} {:<13} {:<8}".format('Zip','City','State')

for i in range(len(zipcodestxt)):
    #if loop to check if the zip_code is present in the db
    if db.zipstates.find({'zip_code': int(zipcodestxt[i])}).count() > 0:
        query = {'zip_code': int(zipcodestxt[i])}
        projection={'_id':0,'city': 1, 'state': 1 }
        cur=db.zipstates.find(query,projection)
    #loop to print each row
        for doc in cur:
        #format checks for length of zip as 5, else add leading 0
            print "{:<6} {:<13} {:<8}".format(str(zipcodestxt[i]).rjust(5,'0'
),
                                              doc['city'], doc['state'])
    else:
        #if zip_code not present
        print "{:<6} 'Not Valid Zip'".format(zipcodestxt[i])
```

```
Zip    City          State
10463  Bronx         NY
06520  New Haven     CT
00603  Aguadilla     PR
75225  Dallas        TX
90210  Beverly Hills CA
99999  'Not Valid Zip'
```

Part C

In [10]:
```
#reading the lat lon from mysteryLatLong.txt
f = open("mysteryLatLong.txt",'r')
listmys = f.readlines()
listmys = listmys[0].split(',')
#list, to store the lat lon
listmystery=[]
for i in listmys:
    listmystery.append(i)
print (listmystery)
```

```
['40.8276', ' -73.92614', ' 41.94887', ' -87.65778', ' 32.7582813', ' -97.110
5791']
```

In [11]:
```
from math import radians, cos, sin, asin, sqrt
#python function that computes the distance in km
def haversine(lon1, lat1, lon2, lat2):
    lon1, lat1, lon2, lat2 = map(radians, [float(lon1), float(lat1), float(lon
2), float(lat2)])
    dlon = lon2-lon1
    dlat = lat2-lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    km = 6371*c
    return km
```

In [12]:
```
#assuming the largest value
closetPoint1 = 999999999999.9999
closetPoint2 = 999999999999.9999
closetPoint3 = 999999999999.9999

dict1 = {}
dict2 = {}
dict3 = {}

#print the column names
print "{}\t{}\t{}\t{}".format('GPS-Coordinates','City','State','Zipcode')

for p in db.zipstates.find({}):
    #if values present in db
    if(p["longitude"] and p["latitude"]):
        #haversine fn takes lon,lat
        if(haversine(listmystery[1],listmystery[0],p["longitude"],p["latitude"
])<closetPoint1):
            dict1 = p
            closetPoint1 = haversine(listmystery[1],listmystery[0],p["longitud
e"],p["latitude"])
        if(haversine(listmystery[3],listmystery[2],p["longitude"],p["latitude"
])<closetPoint2):
            dict2 = p
            closetPoint2 = haversine(listmystery[3],listmystery[2],p["longitud
e"],p["latitude"])
        if(haversine(listmystery[5],listmystery[4],p["longitude"],p["latitude"
])<closetPoint3):
            dict3 = p
            closetPoint3 = haversine(listmystery[5],listmystery[4],p["longitud
e"],p["latitude"])

print "{},{}\t{}\t{}\t{}".format(listmystery[0],listmystery[1],dict1["city"],d
ict1["state"],dict1["zip_code"])
print "{},{}\t{}\t{}\t{}".format(listmystery[2],listmystery[3],dict2["city"],d
ict2["state"],dict2["zip_code"])
print "{},{}\t{}\t{}\t{}".format(listmystery[4],listmystery[5],dict3["city"],d
ict3["state"],dict3["zip_code"])
```

```
GPS-Coordinates City    State   Zipcode
40.8276, -73.92614      Bronx   NY      10451
 41.94887, -87.65778    Chicago IL      60659
 32.7582813, -97.1105791        Arlington       TX      76012
```

In [ ]:
```
What do these mystery zip codes have in common?
Ans: The above zip codes belong to the US cities.
```