**Assignment 2 - Introduction to Google App Engine (and Databases)**

**Name: Shweta Pathak**

**UTA Id: 1001154572**

**Net id: ssp4572**

**Section: 13:00 to 15:00**

**References:**

1. <https://cloud.google.com/appengine/docs/whatisgoogleappengine>
2. <https://cloud.google.com/appengine/docs/python/>
3. <https://cloud.google.com/appengine/docs/python/cloudsql/>
4. <http://stackoverflow.com/questions/2970599/upload-and-parse-csv-file-with-google-app-engine>
5. <https://www.youtube.com/watch?v=VmJSamWsyVs>
6. <http://stackoverflow.com/questions/372885/how-do-i-connect-to-a-mysql-database-in-python>
7. <https://cloud.google.com/appengine/docs/python/googlecloudstorageclient/functions#open>
8. <https://cloud.google.com/appengine/docs/python/googlecloudstorageclient/getstarted>

**#Code to upload the earthquakes\_last30.csv file to google cloud storage**

#import statements.

import argparse

import httplib2

import os

import sys

import json

import time

import datetime

import io

import hashlib

from apiclient import discovery

from oauth2client import file

from oauth2client import client

from oauth2client import tools

from apiclient.http import MediaIoBaseDownload

file\_name = raw\_input("Enter the name of the file :\n")

\_BUCKET\_NAME = 'assg1\_bucket'

\_API\_VERSION = 'v1'

parser = argparse.ArgumentParser(

description=\_\_doc\_\_,

formatter\_class=argparse.RawDescriptionHelpFormatter,

parents=[tools.argparser])

CLIENT\_SECRETS = os.path.join(os.path.dirname(\_\_file\_\_), 'client\_secret.json')

FLOW = client.flow\_from\_clientsecrets(CLIENT\_SECRETS,

scope=[

'https://www.googleapis.com/auth/devstorage.full\_control',

'https://www.googleapis.com/auth/devstorage.read\_only',

'https://www.googleapis.com/auth/devstorage.read\_write',

],

message=tools.message\_if\_missing(CLIENT\_SECRETS))

#Puts a object into file after encryption and deletes the object from the local PC.

def put(service,file\_name):

try:

req = service.objects().insert(

bucket=\_BUCKET\_NAME,

name=file\_name,

media\_body = file\_name)

resp = req.execute()

except client.AccessTokenRefreshError:

print ("Error in the credentials")

def main(argv):

flags = parser.parse\_args(argv[1:])

storage = file.Storage('sample.dat')

credentials = storage.get()

if credentials is None or credentials.invalid:

credentials = tools.run\_flow(FLOW, storage, flags)

http = httplib2.Http()

http = credentials.authorize(http)

service = discovery.build('storage', \_API\_VERSION, http=http)

options\_toselect = {1: put}

option = input("Enter your option 1. put \n")

start\_time = time.clock()

#print start\_time

print " \n Uploading in progress."

options\_toselect[option](service,file\_name)

end\_time = time.clock()

print " \n File uploaded " + file\_name

#print end\_time

total\_time = end\_time - start\_time

print "\n Total time taken in seconds to upload the file : %f" %total\_time

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

main(sys.argv)

**Time taken:**

Total time taken in seconds to upload the file: 10.219065

**# Code to insert data from .csv file uploaded to cloud storage into Cloud SQL database and to execute the queries to find the total earthquakes with magnitude 2,3,4,5 or greater**

**Main.py:**

import webapp2

from google.appengine.ext.webapp.util import run\_wsgi\_app

import MySQLdb

import os

import csv

import time

import cloudstorage

from google.appengine.api import app\_identity

from cloudstorage import storage\_api

# Define your production Cloud SQL instance information.

\_INSTANCE\_NAME = 'adv-database-1:assginment2'

class MainPage(webapp2.RequestHandler):

def get(self):

# Connecting to MySQLdb cloud database

if (os.getenv('SERVER\_SOFTWARE') and

os.getenv('SERVER\_SOFTWARE').startswith('Google App Engine/')):

db = MySQLdb.connect(unix\_socket='/cloudsql/' + \_INSTANCE\_NAME, db='earthquakes', user='root', charset='utf8', passwd='root1234')

else:

db = MySQLdb.connect(host='76.183.83.23', port=3306, db='earthquakes', user='root', charset='utf8', passwd='root1234')

cursor = db.cursor()

# Creating a table in cloud database

cursor.execute("create table if not exists all\_month(time varchar(225), latitude varchar(225),longitude varchar(225), depth varchar(225), mag varchar(225),magType varchar(225), nst varchar(225), gap varchar(225), dmin varchar(225),rms varchar(225), net varchar(225), id varchar(225) PRIMARY KEY, updated varchar(225), place varchar(225), type varchar(225))")

# Reading file from cloud storage bucket

bucket = '/assg1\_bucket/'

file\_name = bucket + 'earthquakes\_last30.csv'

gcs\_file = cloudstorage.open(file\_name, mode='r')

fileReader = csv.reader(gcs\_file)

cursor.execute("delete from all\_month")

start\_time = time.clock()

for row in fileReader:

cursor.execute("INSERT INTO all\_month(time,latitude,longitude,depth,mag,magType,nst,gap,dmin,rms,net,id,updated,place,type) VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)", row)

end\_time = time.clock()

total\_time = end\_time - start\_time

self.response.out.write("<html><body>")

self.response.out.write("<br>")

self.response.write("\n Time taken to load data : ")

self.response.write(total\_time)

# Function calls to calculate the time taken and number of earthquakes

query(str(2),str(2.99),cursor,self)

query(str(3),str(3.99),cursor,self)

query(str(4),str(4.99),cursor,self)

query(str(5),str(5.99),cursor,self)

query(str(6),str(6.99),cursor,self)

query(str(7),str(7.99),cursor,self)

self.response.out.write("</body></html>")

db.commit()

db.close()

class Guestbook(webapp2.RequestHandler):

def post(self):

# Handle the post to create a new guestbook entry.

if (os.getenv('SERVER\_SOFTWARE') and

os.getenv('SERVER\_SOFTWARE').startswith('Google App Engine/')):

db = MySQLdb.connect(unix\_socket='/cloudsql/' + \_INSTANCE\_NAME, db='guestbook', user='root', charset='utf8', passwd='root1234')

else:

db = MySQLdb.connect(host='76.183.83.23', port=3306, db='guestbook', user='root', charset='utf8')

application = webapp2.WSGIApplication([('/', MainPage),

('/sign', Guestbook)],

debug=True)

def query(mag1,mag2,cursor,self):

self.response.out.write("<br>")

self.response.write("\nWeek1\n")

self.response.out.write("<br>")

s1 = time.clock()

cursor.execute("select \* from all\_month where mag between '"+mag1+"' and '"+mag2+"' and time between '2015-05-14T00:00:00Z' and '2015-05-21T23:59:00Z' and type='earthquake'")

self.response.write("Number of earthquakes for the magnitude :\n" +mag1+"and "+mag2)

self.response.out.write("<br>")

row1 = cursor.fetchall()

self.response.out.write(len(row1))

e1 = time.clock()

t1 = e1-s1

self.response.out.write("<br>")

self.response.write("\nTime taken to execute :\n")

self.response.write(t1)

self.response.out.write("<br>")

self.response.write("\nWeek2\n")

s2 = time.clock()

cursor.execute("select \* from all\_month where mag between '"+mag1+"' and '"+mag2+"' and time between '2015-05-22T00:00:00Z' and '2015-05-28T23:59:00Z' and type ='earthquake'")

self.response.out.write("<br>")

self.response.write("Number of earthquakes for the magnitude :\n" +mag1+"and "+mag2)

row2 = cursor.fetchall()

self.response.out.write("<br>")

self.response.out.write(len(row2))

e2 = time.clock()

t2 = e2-s2

self.response.out.write("<br>")

self.response.write("\nTime taken to execute :\n")

self.response.write(t2)

self.response.out.write("<br>")

self.response.write("\nWeek3\n")

s3 = time.clock()

self.response.out.write("<br>")

cursor.execute("select \* from all\_month where mag between '"+mag1+"' and '"+mag2+"' and time between '2015-05-29T00:00:00Z' and '2015-06-04T23:59:00Z' and type='earthquake'")

self.response.write("Number of earthquakes for the magnitude :\n" +mag1+"and "+mag2)

self.response.out.write("<br>")

row3 = cursor.fetchall()

self.response.out.write(len(row3))

e3 = time.clock()

t3 = e3-s3

self.response.out.write("<br>")

self.response.write("\nTime taken to execute :\n")

self.response.write(t3)

self.response.out.write("<br>")

self.response.write("\nWeek4\n")

s4 = time.clock()

self.response.out.write("<br>")

cursor.execute("select \* from all\_month where mag between '"+mag1+"' and '"+mag2+"' and time between '2015-06-05T00:00:00Z' and '2015-06-13T23:59:00Z' and type='earthquake'")

self.response.write("Number of earthquakes for the magnitude :\n" +mag1+"and "+mag2)

self.response.out.write("<br>")

row4 = cursor.fetchall()

self.response.out.write(len(row4))

e4 = time.clock()

t4 = e4-s4

self.response.out.write("<br>")

self.response.write("\nTime taken to execute :\n")

self.response.write(t4)

self.response.out.write("<br>")

def main():

application = webapp2.WSGIApplication([('/', MainPage),

('/sign', Guestbook)],

debug=True)

run\_wsgi\_app(application)

if \_\_name\_\_ == "\_\_main\_\_":

main()

app.yaml:

application: adv-database-1

version: 1

runtime: python27

api\_version: 1

threadsafe: yes

handlers:

- url: /favicon\.ico

static\_files: favicon.ico

upload: favicon\.ico

- url: .\*

script: main.application

libraries:

- name: webapp2

version: "2.5.2"

- name: MySQLdb

version: "latest"

- name: jinja2

version: "latest"

**Time taken:**

Time taken to load data in Cloud SQL database: 1.38

**Time-Magnitude relationship: (Time in seconds)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Week | Mag = 2 – 2.99 | Mag = 3-3.99 | Mag = 4-4.99 | Mag = 5-5.99 | Mag = 6-6.99 | Mag = 7-7.99 |
| 14th May to 21st May | 0.03 | 0.01 | 0.01 | 0.01 | 0.0 | 0.0 |
| 22nd May to 28th May | 0.02 | 0.0 | 0.01 | 0.0 | 0.0 | 0.0 |
| 29th May to 4th June | 0.02 | 0.01 | 0.0 | 0.0 | 0.0 | 0.0 |
| 5th June to 13th June | 0.03 | 0.01 | 0.0 | 0.01 | 0.0 | 0.0 |

Time taken is less where less number of records are retrieved. As in the case of earthquakes with magnitude 5 or greater there are less number of records.  
Time taken to upload file to cloud storage is more as compared to time taken to load data in cloud sql. Earthquakes with magnitude 2 are more frequent than others.