Name:Kevin Thomas

Last 4 digits ID:4593

Computer IP address:

CSE 6331, Cloud Computing

Quiz Q3, 4 Summer 2018 (c) DL, UTA, 2018

C&P means cut and paste only those relevant lines from your program(s) into this quiz.

1. I understand that I am on my honor during this quiz, I will not collaborate, use

non-allowed sources, and I will not discuss or share this quiz with anyone for the next

4 hours.

You MAY: use Google to search, use previous source code,

YOU MAY NOT use:

Email, Facebook, Hangouts, IM, chats, Skype or ANY other human connection.

This is a timed test. Late tests will have points deducted for being late.

Very late tests will not be graded.

When you are complete, with any part, please raise your hand, so we may visually inspect that part.

The second part of the test, you should electronically submit, you will need to copy and paste

only those lines of code to implement that part of the test, usually a few (two to eight) lines of code.

Place it immediately after the question.

Submit this Quiz (renamed) with code cut and pasted, ONLY text. DO NOT submit zips, binaries, libraries,

or anything other than text.

When any parts(questions) are complete complete, submit this test, you may make multiple submissions.

If you have computer, or other, problems, please raise your hand immediately.

If you understand and agree, please initial here:

Kevin Thomas

2. Get files from this same folder.

3. Name your program with your name and last digits of your ID.

4. Using the cloud services provider, all functionality possible should be implemented on that provider.(Of course, displaying web pages through a browser and the user interface is "local")edata.csv is earthquake data from USGS.

Show and submit code: (Create and show a web interface to perform the following on the cloud service provider)

5. Show a web page and interface (which resides on the cloud provider) with your name and student ID in large font at the top of every web page dispayed (for this quiz)Import edata into a Relational Data Base (SQL), if you have already done this(briefly) explain/describe how you did this. Please note that for some parts of the following you will need to create indexes (keys)

appropriately.

Then, for all rms values less than, or equal, 0.25, remove all those entries.

@app.route('/')

def home():

return render\_template('home.html')

@app.route('/enternew')

def upload\_csv():

return render\_template('upload.html')

@app.route('/addrec',methods = ['POST', 'GET'])

def addrec():

if request.method == 'POST':

con = sql.connect("database.db")

csv = request.files['myfile']

file = pd.read\_csv(csv)

file.to\_sql('Earthquake', con, schema=None, if\_exists='replace', index=True, index\_label=None, chunksize=None, dtype=None)

con.close()

return render\_template("result.html",msg = "Record inserted successfully")

@app.route('/list')

def list():

con = sql.connect("database.db")

cur = con.cursor()

cur.execute("select \* from Earthquake where rms > 0.25")

rows = cur.fetchall();

con.close()

return render\_template("list.html",rows = rows)

6. Allow a user, through a web form, to give the first letter of a net id (these are two letters, for example "nc" or "nn", we will give you "n") and a loop value LOOP, and you will generate LOOP

number of random queries distributed between the net id that start with that letter (in this case"nn", "nm", "nc"). (For LOOP = 100, and NetIDFirstLetter = "n", you will generate 100 selects randomly

generated with net id = "nn", "nm", "nc").On the web page show us the first result time and location for the first select, and the total time used to do ALL queries on the cloud service provider.

@app.route('/records')

def records():

return render\_template('records.html')

@app.route('/options' , methods = ['POST', 'GET'])

def options():

con = sql.connect("database.db")

start\_time = time.time()

num =int(request.form['num'])

loc = (request.form['loc'])

rows = []

d =[]

for i in range(num):

cur = con.cursor()

b = 'select \* from Earthquake WHERE net LIKE ?', ('%'+loc+'%',)

cur.execute("select \* from Earthquake WHERE net LIKE ?", ('%'+loc+'%',))

get = cur.fetchall();

rows.append(get)

cur.execute("select \* from Earthquake WHERE net LIKE ?", ('%'+loc+'%',))

get = cur.fetchall();

con.close()

end\_time = time.time()

elapsed\_time = end\_time - start\_time

return render\_template("list1.html",rows = [rows,elapsed\_time])

html

label><h2>Display the no of records:</h2></label><br>

<form action = "{{ url\_for('options') }}" method = "POST">

<label>Enter number:</label>

<input type = "text"

name = "num"

/>

<label>Enter First letter of the net id:</label>

<input type = "text"

name = "loc"

/>

<br>

<br>

<input type = "submit" value = "submit" /><br>

</form>

7. Using an in-memory caching mechanism (NOT an in-memory database) repeat the previous step.

@app.route('/restricted')

def restricted():

return render\_template('rest.html')

@app.route('/options2' , methods = ['POST', 'GET'])

def options2():

con = sql.connect("database.db")

start\_time = time.time()

num =int(request.form['num'])

loc = (request.form['loc'])

rows = []

d =[]

for i in range(num):

cur = con.cursor()

b = 'select \* from Earthquake WHERE net LIKE ?', ('%'+loc+'%',)

cur.execute("select \* from Earthquake WHERE net LIKE ?", ('%'+loc+'%',))

get = cur.fetchall();

rows.append(get)

if r.get(b):

print ('Cached')

d.append('Cached')

else:

print('Not Cached')

d.append('Not Cached')

cur.execute("select \* from Earthquake WHERE net LIKE ?", ('%'+loc+'%',))

get = cur.fetchall();

r.set(b,get)

con.close()

end\_time = time.time()

elapsed\_time = end\_time - start\_time

print (elapsed\_time)

return render\_template("list2.html",rows = [d,elapsed\_time])

<header> <h1> Displaying time taken for execution for queries with caching </h1>

</header>

<br> <br>

<h1>Execution time with caching:{{rows[1]}}</h1>

<table border = 1>

{% for row in rows[0] %}

<tr>

<h1> </h1>

<br>

<h1>{{row}}</h1>

</tr>

{% endfor %}

</table>

8. Show GTA parts 5,6, 7

9. When complete, return (send) this quiz

If you finish early, send this immediately, otherwise send between the end of class and no more than 1 minute after that.