Olusola Ogun

Final Exam

CSC 455

PART 1

1. The following database contains information about actors, plays, and roles performed.

   a. Write a SQL query that returns the number of actors who have performed in three or more different plays written by the author “August Wilson”.

   ```
   SELECT'count(r.actor_id) FROM'Role'r,'Play'p WHERE'p.author'=''August'Wilson''AND p.play_id'='r.play
   _id GROUP'BY r.actor_id HAVING'count(DISTINCT r.play_id)'>=3
   ```

   b. Write a SQL query that returns the names of all actors who have performed some play by the author “Chekhov” and have never performed in any play written by author “Shakespeare”.The list should not contain duplicates but does not need to be ordered.

   ```
   SELECT'DISTINCT a.name FROM'Actor'a,'Play'p,'Role'r WHERE'p.author'=''Chekhov''AND'p.play_id'='r.p
   lay_id'AND'r.actor_id'='a.actor_id'AND a.actor_id NOT'IN'(SELECT'r2.actor_id FROM'Role'r2,'Play'p2 WH
   ERE'p2.author'=''Shakespeare'' AND 'p2.play_id'='r2.play_id)
   ```

2. Use Python SQL to complete the below. (25 points)

In [ ]:
```
# Use Python SQL to complete the below. (25 points)
import sqlite3
import urllib.request
import json
import time

# Identify a primary key generate and execute a python SQL DDL command to Create table.
conn = sqlite3.connect("final.db")
c = conn.cursor()
sql = 'CREATE TABLE Books (ISBN CHAR(10) NOT NULL, NAME CHAR(30),PRICE INT, NumberOfCopies IN
T)'
c.execute('drop table Books')
c.execute(sql)
# ISBN is selected as the Primary Key

c.execute('create table books (ISBN NUMBER(20) NOT NULL, NAME VARCHAR2(100), PRICE NUMBER(2
0), NUMBEROFCOPIES NUMBER(20) CONSTRAINT BOOK_PK PRIMARY KEY (ISBN))')

# Make your own data and insert 10 rows in the table created using python SQLite.
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('1232633434', 'MATH', 10,
5 )")
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('4874533434', 'ADVENG
LISH', 210,100)")
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('1202633432', 'ADVMAT
H', 100,5 )")
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('1242633434', 'FRENCH',
 10,5 )")
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('1532633130', 'PYTHON',
 10,5 )")
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('1232693434', 'MATH', 10,
5 )")
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('4874533434', 'ADVENG
LISH', 210,100)")
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('2202633432', 'ADVPYT
HON', 190,5 )")
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('1249633434', 'CIVIL', 19
0,5 )")
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('1539633434', 'CPLUS', 2
10,5 )")

# Retrieve all the records from the table Books
c.execute("SELECT * from books")

# Insert another record in the table as ('9875465248','Programming With Python',100,50)
c.execute("INSERT INTO Books(ISBN, NAME,PRICE, NumberOfCopies) VALUES ('9875465248', 'Program
ming With Python', 100,50 )")

# Update the record inserted above – Price -> new Value -> 60 and Number of copies -> new value -> 1
00
c.execute("UPDATE Books set PRICE = 60 ,NumberOfCopies=100 where ISBN = 9875465248")

# Delete the record from the above table where number of copies = 100
c.execute("DELETE from books where NumberOfCopies=100;")

# Retrieve all the records from the table Books
c.execute("SELECT * from books")
```

```
# Retrieve sum of all copies and the avg price from the above table.
c.execute("SELECT sum(NumberOfCopies) AS Total,(avg(sum(PRICE)*sum(NumberOfCopies))) AS Aver
age priceFROM Books;")
```

Part 2

1. We will use a full day worth of tweets as an input (there are total of 4.4M tweets in this file): http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt (http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt)

   a. Create a 3rd table incorporating the Geo table (in addition to tweet and user tables that you already have) and extend your schema accordingly. You will need to generate an ID for the Geo table primary key (you may use any value or combination of values as long as it is unique) for that table and link it to the Tweet table (foreign key should be in the Tweet table because there can be multiple tweets sent from the same location). In addition to the primary key column, the geo table should have “type”, “longitude” and “latitude” columns

In [ ]:
```
conn = sqlite3.connect('Tweets_Database.db')
c = conn.cursor()
webFD=urllib.request.urlopen("http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt")

Line = webFD.readline()

tweetsdata= (Line.decode('utf8')).split('EndOfTweet')

c.execute('DROP TABLE IF EXISTS Tweets');
c.execute('DROP TABLE IF EXISTS USER);

# Create Table Tweets
c.execute('''CREATE TABLE tweet(created_at DATETIME, user_id INT,id_str INT, text TEXT, source TEXT, in
_reply_to_user_id INT,in_reply_to_screen_name TEXT,
        in_reply_to_status_id INT, retweet_count INT,
        contributors TEXT CONSTRAINT tweet_FK FOREIGN KEY (user_id) REFERENCES user(id), CONSTR
AINT tweet_FK2 FOREIGN KEY (user_str) REFERENCES Geo(user_str)''')


# Create Table User
c.execute('''CREATE TABLE user (id INT, screen_name TEXT, description TEXT, friends_count INT, contrib
utors TEXT CONSTRAINT USER_PK PRIMARY KEY(ID))''')

# Create Table Geo
c.execute('''CREATE TABLE Geo (id_str INT, type TEXT, longitude INT, latitude INT, CONSTRAINT GEO_PK
 PRIMARY KEY(id_str)''')
```

   b. Use python to download from the web and save to a local text file (not into database yet, just to text file) at le ast 500,000 lines worth of tweets. Test your code with fewer rows first – you can reduce the number of tweets if your computer is running too slow to handle 500K tweets. Report how long did it take

In [ ]:
```
tweet_data=urllib.request.urlopen("http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/OneDayOfTweets.txt").read(1000000)
str_data = str(tweet_data)
lines = str_data.split('\\n') # then split it into lines
fx = open('downloadedfile.txt','w')
for line in lines:
   fx.write(line+'\n')
fx.close()
```

c. Repeat what you did in part-b, but instead of saving tweets to the file, populate the 3-table schema that you created in SQLite. Be sure to execute commit and verify that the data has been successfully loaded (report row counts for each of the 3 tables).

In [ ]:
```
# Creating Tables and error.txt file.

datatweet = []
datauser = []
errors = open('exam_errors.txt', 'w')

for OneTweetline in tweetdata:
   try:
      tweetrecord= json.loads(OneTweetline, encoding='utf-8')
      datatweet.append((tweetrecord["created_at"], tweetrecord["user_id "],  tweetrecord["id_str"], tweet
record["text"],tweetrecord["source"], tweetrecord["in_reply_to_user_id"],tweetrecord["in_reply_to_scree
n_name"], tweetrecord["in_reply_to_status_id"],tweetrecord['retweet_count'],tweetrecord['contributor
s']))
      datauser.append((tweetrecord["id"], tweetrecord["screen_name "],  tweetrecord["desription"], twe
etrecord["friends_count"]))
   except ValueError:
      print(tweet)
      errors.write(tweet)
errors.close()

#inserting values in to table
c.executemany(
'''INSERT INTO tweet (created_at, user_id, id_str, text, source, in_reply_to_user_id, in_reply_to_screen_na
me,
   in_reply_to_status_id,retweet_count, contributors) VALUES (?,?,?,?,?,?,?,?,?,?,?)''', datatweet)
c.executemany(
'INSERT INTO User (id, screen_name, description, source, friends_count) VALUES (?,?,?,?,?)', datauser)
c.executemany(
'INSERT INTO GEO (id_str, type, longitude, source, longitude ) VALUES (?,?,?,?,?)',   datauser)

print("Created table in tweetdata.db")

conn.commit()
conn.close()


start = time.time()

for i in range(100000):
   line = webFD.readline()

tweetsdata= (Line.decode('utf8')).split('EndOfTweet')

end = time.time()
print ('difference is', (end-start), 'seconds')
print ('perfomane:', 100000/(end-start), 'operation per second')
For i in range(100000):
   line = webFD.readline()
   tweetsdata= (line.decode('utf8')).split('EndOfTweet')
   fx.write(tweetsdata+'\n'
```

Write and execute SQL queries to do the following

i. Find tweets where tweet id_str contains “44” or “77” anywhere in the column

Select id_str from tweet Where id_str = %44% or id_str = %77%

ii. Find how many unique values are there in the “in_reply_to_user_id” column

Select count(distinct in_reply_to_user_id) as reply_count FROM tweet

iii. Find the average longitude and latitude value for each user name

Select screen_name longitude latitude from user inner join Geo inner join user where user_id= id Order by te
xt