Alex Teboul

DSC 450: Databases for Analytics

Assignment 3

Due Sunday, May 12th.

Part 1

In this and the next part we will use an extended version of the schema from Assignment 2. You can find it in a file ZooDatabase.sql posted with this assignment on D2L.Once again, it is up to you to write the SQL queries to answer the following questions:

1. **List the animals (animal names) and the ID of the zoo keeper assigned to them.**

```
SELECT Animal.AName, Zookeeper.ZName
  FROM Animal
    JOIN Handles ON Animal.AID=Handles.AnimalID
    JOIN ZooKeeper ON Handles.ZOOKEEPID=Zookeeper.ZID
    ;
```

AName	ZName
Galapagos Penguin	Jim Carrey
Emperor Penguin	Jim Carrey
Alpaca	Jim Carrey
Sri Lankan sloth bear	Tina Fey
Grizzly bear	Tina Fey
Giant Panda bear	Tina Fey
Siberian tiger	Rob Schneider
Bengal tiger	Rob Schneider
South China tiger	Rob Schneider
Alpaca	Rob Schneider

2. **Now repeat the previous query and make sure that the animals without an assigned handler also appear in the answer.**

```
SELECT Animal.AName, ZooKeeper.ZName
    FROM Animal
      LEFT OUTER JOIN Handles ON Animal.AID=Handles.AnimalID
      LEFT OUTER JOIN ZooKeeper ON Handles.ZookeepID=Zookeeper.ZID
      ;
```

AName	ZName
Galapagos Penguin	Jim Carrey
Emperor Penguin	Jim Carrey
Alpaca	Jim Carrey
Sri Lankan sloth bear	Tina Fey
Grizzly bear	Tina Fey
Giant Panda bear	Tina Fey
Siberian tiger	Rob Schneider
Bengal tiger	Rob Schneider
South China tiger	Rob Schneider
Alpaca	Rob Schneider
Florida black bear	(null)
Llama	(null)

3. **Report, for every zoo keeper name, the average number of hours they spend feeding all animals in their care.**

```
SELECT Zookeeper.ZName, AVG(Animal.TimeToFeed)
  FROM Animal
    JOIN Handles ON Animal.AID=Handles.AnimalID
    JOIN ZooKeeper ON Handles.ZOOKEEPID=Zookeeper.ZID
  GROUP BY ZooKeeper.ZName
  ORDER BY AVG(Animal.TimeToFeed)
  ;
```

ZName	AVG(Animal.TimeToFeed)
Jim Carrey	0.5
Rob Schneider	2.1875
Tina Fey	2.33333333333333333333333333333333333333

4. **Report every handling assignment (as a list of assignment date, zoo keeper name and animal name). Sort the result of the query by the assignment date in an ascending order.**

```
SELECT Handles.Assigned, ZooKeeper.ZName, Animal.AName
  FROM Handles
    JOIN Animal ON Animal.AID=Handles.AnimalID
    JOIN ZooKeeper ON ZooKeeper.ZID=Handles.ZooKeepID
    ORDER BY Handles.Assigned ASC
    ;
```

Assigned	ZName	AName
01-JAN-00	Jim Carrey	Galapagos Penguin
01-JAN-00	Jim Carrey	Alpaca
01-JAN-00	Rob Schneider	Siberian tiger
02-JAN-00	Jim Carrey	Emperor Penguin
02-JAN-00	Tina Fey	Sri Lankan sloth bear
03-JAN-00	Tina Fey	Giant Panda bear
03-JAN-00	Rob Schneider	Bengal tiger
04-JAN-00	Tina Fey	Grizzly bear
04-JAN-00	Rob Schneider	Alpaca
05-JAN-00	Rob Schneider	South China tiger

5. **Find the names of animals that have at least 1 zoo keeper assigned to them.**

```
SELECT Animal.AName --, Handles.AnimalID,COUNT(Handles.AnimalID)
  FROM Handles
    JOIN Animal ON Animal.AID=Handles.AnimalID
  GROUP BY Handles.AnimalID, Animal.AName
  HAVING COUNT(Handles.AnimalID)>=1
  ORDER BY Handles.AnimalID
  ;
```

AName
Galapagos Penguin
Emperor Penguin
Sri Lankan sloth bear
Grizzly bear
Giant Panda bear
Siberian tiger
Bengal tiger
South China tiger
Alpaca

6. **Find the names of animals that have 0 or 1 (i.e., less than 2) zoo keepers assigned to them.**

```
SELECT Animal.AName --, Handles.AnimalID,COUNT(Handles.AnimalID)
   FROM Animal
      LEFT OUTER JOIN Handles ON Animal.AID=Handles.AnimalID
   GROUP BY Handles.AnimalID, Animal.AName
   HAVING COUNT(Handles.AnimalID)<=2
   ORDER BY Handles.AnimalID
   ;
```

AName
Galapagos Penguin
Emperor Penguin
Sri Lankan sloth bear
Grizzly bear
Giant Panda bear
Siberian tiger
Bengal tiger
South China tiger
Alpaca
Florida black bear
Llama

Part 2

A. Write a python script that is going to read the queries that you have created in Part-1 from a SQL file, execute each SQL query against a SQLite database and print the output of that query (i.e., given a queries.sql, execute each query from that file automatically). You must read your SQL queries from a file, please do not copy SQL directly into the python code. The code that runs commands from the ZooDatabase.sql file is included with the homework (runSQL.py). All you have to do is to change it so that it reads *your* queries from a SQL file (that just means you need to rename the file) and also *prints* the output of your queries. You must print every row individually using a for-loop.

```
import sqlite3
from sqlite3 import OperationalError

conn = sqlite3.connect('csc455_HW3.db')
c = conn.cursor()

# Open and read the file as a single buffer
fd = open('ZooDatabase_Alex.sql', 'r')
# Read as a single document (not individual lines)
sqlFile = fd.read()
fd.close()

# all SQL commands (split on ';' which separates them)
sqlCommands = sqlFile.split(';')

# Execute every command from the input file (separated by ";")
for command in sqlCommands:
    # This will skip and report errors
    # For example, if the tables do not yet exist, this will skip over
    # the DROP TABLE commands
    try:
        c.execute(command)
        for row in c.fetchall():
            print(row)
        print()
    except OperationalError as msg:
        print ("Command skipped: ", msg)

c.close()
conn.commit()
conn.close()
```

B. Create the table and use python to automate loading of the following file into SQLite: http://rasinsrv07.cstcis.cti.depaul.edu/CSC455/Public_Chauffeurs_Short_hw3.csv
It contains comma-separated data, with two changes: NULL may now be represented by NULL string or an empty string (e.g., either ,NULL, or „) and some of the names have the following form "Last, First" instead of "First Last", which is problematic because when you split the string on a comma, you end up with too many values to insert.

1) Implement loading the data using .split() on comma, similarly to the previous homework and report any issues that you encountered.

```
import sqlite3
#create the table:
Chauffeurs = '''CREATE TABLE Chauffeurs(
   License_Number NUMBER(8=10),
   Renewed VARCHAR(10),
   Status VARCHAR(10),
   Status_Date DATE(MM-DD-YY),
   Driver_Type VARCHAR(20),
   License_Type VARCHAR(10),
   Original_Issue_Date DATE(MM-DD-YY),
   Name VARCHAR(32),
   Sex VARCHAR(10),
   Chauffeur_City VARCHAR(32),
   Chauffeur_State VARCHAR(4),
   Record_Number VARCHAR(16),
   PRIMARY KEY(License_Number)
);'''

connection = sqlite3.connect('csc455.db')
cursor = connection.cursor()
#cursor.execute('DROP TABLE Chauffeurs')
cursor.execute(Chauffeurs)

os.getcwd()
fd = open('Public_Chauffeurs_Short_hw3.csv', 'r')
allLines = fd.readlines()
for line in allLines:
   vals = line.strip().split(', ')
   cursor.execute("INSERT OR IGNORE INTO Chauffeurs VALUES(?,?,?,?,?,?,?,?,?,?,?,?)", [vals[0], vals[1],
vals[2],vals[3], vals[4], vals[5], vals[6], vals[7], vals[8], vals[9], vals[10], vals[11]]);

connection.commit()   # finalize inserted data
```

```
connection.close()    # close the connection
fd.close()          # close the open file
```

- Encountered the issued sqlite3.OperationalError: near "=": syntax error in Line 25 (when file opened)
- Believe this is due to that First, Last problem as indicated in the problem statement.

2) Once you do part-1, you can use csvreader to resolve the problem, like so:

```
import csv
fd = open('Public_Chauffeurs_Short_hw3.csv', 'r')
reader = csv.reader(fd)
for row in reader:
   print(row)
fd.close()
```

- By not splitting on the commas and going row by row, there are the correct number of values to insert.

Part 3

Using the company.sql database (posted in with this assignment on D2L), write the following SQL queries.

1. Find the names of all employees who are directly supervised by 'Franklin T Wong' (your SQL query must use the name to match the name, not the SSN).

```
SELECT FName, Minit, LName
  FROM EMPLOYEE
    WHERE super_ssn = (SELECT SSN
                          FROM Employee
                             WHERE FName = 'Franklin' AND Minit='T' AND LName='Wong')
       ;
```

FName	Minit	LName
John	B	Smith
Ramesh	K	Narayan
Joyce	A	English
Melissa	M	Jones

2. For each project, list the project name, project number, and the total hours per week (by all employees) spent on that project.

```
SELECT project.pname as project_name, works_on.pno as project_number,
SUM(works_on.hours) as total_hours
   FROM works_on
      JOIN project ON works_on.pno=project.pnumber
      group by works_on.pno, project.pname
      ORDER BY works_on.pno
         ;
```

project_name	project_number	total_hours
ProductX	1	62
ProductY	2	38
ProductZ	3	5
Computerization	10	55
Reorganization	20	35
Newbenefits	30	60

3. For each department, retrieve the department name and the average salary of all employees working in that department. Order the output by department number in ascending order.

```
SELECT department.dname AS department_name, avg(employee.salary) AS average_salary
   FROM employee
      JOIN department ON employee.dno=department.dnumber
      GROUP BY department.dname, department.dnumber
      ORDER BY department.dnumber ASC
         ;
```

department_name	average_salary
Headquarters	55000
Administration	28000
Research	32100

4. Retrieve the average salary of all female employees.

```
SELECT avg(salary) AS female_average_salary
  FROM employee
    WHERE sex='F'
    GROUP BY sex
  ;
```

female_average_salary
28625

5. For each department whose average salary is greater than $42,000, retrieve the department name and the number of employees in that department.

```
SELECT department.dname AS department_name, count(employee.ssn) AS
number_of_employees --, avg(salary) AS average_salary
  FROM employee
    JOIN department ON employee.dno=department.dnumber
    HAVING avg(salary)>42000
    GROUP BY department.dname
  ;
```

department_name	number_of_employees
Headquarters	1

6. Retrieve the names of employees whose salary is within $25,000 of the salary of the employee who is paid the most in the company (e.g., if the highest salary in the company is $85,000, retrieve the names of all employees that make at least $60,000.).

```
SELECT FName AS first_name, Minit AS middle_initial, LName AS last_name, Salary
  FROM Employee
    WHERE Salary BETWEEN (SELECT MAX(salary)-25000 FROM employee) AND (SELECT
MAX(salary)+25000 FROM employee)
  ;
--SELECT MAX(salary) FROM employee; --check
```

first_name	middle_initial	last_name	Salary
James	E	Borg	55000
Jennifer	S	Wallace	37000
Franklin	T	Wong	40000
John	B	Smith	30000
Ramesh	K	Narayan	38000