Homework #6

Instructions:

--You will essentially copy your Python code into this Word document and then submit the document in Canvas

--Notes about the formatting:

--Use Courier New font (because it is fixed-width)

--****IMPORTANT: As discussed in class, do not extract, copy, move, share, or take screenshots of any of the data in the database. Use copy/paste to directly move your Python code from Jupyter into this Word document. If I, or the IT department, detect any unauthorized access or usage you will automatically receive an F for the course. Please ask if you are not sure if a specific use is authorized.***

--In general, the assignment will be graded for completeness. However, I reserve the right to grade a question or two for correctness.

Hints:

--The following pandas dataframe methods may come in handy:

.replace()

.apply()

.to_numeric

Name: KEY

For all of the problems below, I would like you to recreate the output we generated in SQL using Python and the pandas package. These are all previous HW questions. Below is an example…

HW#1, Question #1:

Query

```
SELECT    COUNT(*)
FROM      health;
```

Output

```
155143
```

Python Code

```
import pymysql
pymysql.install_as_MySQLdb()
%reload_ext sql
%sql mysql://student:twig-7BAG5qj@mqm-db/
%sql USE sanford;
import numpy as np
import pandas as pd
```

```
result = %sql SELECT * FROM health;
df1 = result.DataFrame()
len(df1)
```

<u>In all of your answers, you can exclude all of the code up to and including the SQL query which pulls data into a temporary variable (here, "result"). Simply include the code that generates the requested output. So, for this example, all that I would need to include in my answer is:</u>

```
df1 = result.DataFrame()
len(df1)
```

1.) HW#1, Question #3

<u>Query</u>

```
SELECT    sex,
          COUNT(sex)
FROM      health
GROUP BY  sex
ORDER BY  COUNT(sex) DESC;
```

<u>Output</u>

```
sex       COUNT(sex)
Female    78503
Male      76639
Unknown   1
```

<u>Python Code</u>

```
q1 = result1.DataFrame()

q1_grouped=q1.groupby('sex').agg({'sex':
np.size}).sort_values('sex',ascending=False)
q1_grouped.columns.values[0]='COUNT(sex)'

q1_grouped=q1_grouped.reset_index()
q1_grouped
```

2.) HW#1, Question #4

<u>Query</u>

```
SELECT    sex,
          AVG(hypertension)
FROM      health
```

```
WHERE      sex != 'Unknown'
GROUP BY   sex;
```

Output

```
sex        AVG(hypertension)
Female     0.5623
Male       0.5916
```

Python Code

```
q2 = result1.DataFrame()

q2_grouped=q2[q2['sex'] !=
'Unknown'].groupby('sex').agg({'hypertension': np.mean})

q2_grouped.columns.values[0]='AVG(hypertension)'
q2_grouped=q2_grouped.reset_index()
q2_grouped
```

3.) HW#1, Question #7

Query

```
SELECT     sex,
           payor,
           hypertension,
           COUNT(*),
           ROUND(AVG(age),2)
FROM       health
WHERE      status = 'Alive' AND
           sex != 'Unknown'
GROUP BY   sex,
           payor,
           hypertension
HAVING     COUNT(*) >= 2000
ORDER BY   sex,
           payor,
           hypertension;
```

Output

```
Female     Medicare              0    19409     79.18
Female     Medicare              1    25214     73.31
Female     Private Ins/Other     0    11347     52.36
Female     Private Ins/Other     1    17158     55.69
Male       Medicare              0    13599     76.08
```

```
      Male      Medicare            1    20689     72.16
      Male      Private Ins/Other   0    14583     52.02
      Male      Private Ins/Other   1    22989     54.31
```

Python Code

```
result3 = %sql SELECT * FROM health;
q3 = result3.DataFrame()
q3['age']=q3['age'].replace({"90+": "90"})
q3=q3.apply(pd.to_numeric, errors='ignore')

q3=q3[(q3['status'] == 'Alive') & (q3['sex'] !=
'Unknown')].groupby(['sex','payor','hypertension']).agg({'age':
[np.size,np.mean]})

q3=q3.round(2)
q3=q3[q3['age']['size']>=2000]
q3=q3.reset_index()
q3.columns = q3.columns.map(''.join)
q3=q3.sort_values(['sex','payor','hypertension'])
```

4.) HW#1, Question #8 (here convert the SQL ***NULL*** value to the Python equivalent and then check for it)

Query

```
SELECT    COUNT(*)
FROM      health
WHERE     a1c IS NULL;
```

Output

```
120374
```

Python Code

```
# There is actually no need to convert here--python converts
# the SQL NULL value to the Python NaN value (i.e. the
# python NULL equivalent)
q4 = result3.DataFrame()
len(q4[q4['a1c'].isnull()])
```

5.) Similar to HW#3, Question #2. Here when you pull the “ontime” table into a dataframe, please include a WHERE clause to only pull data for January (ontime.Month = 1). This is simply to prevent us from overloading the data servers.

Query

```
SELECT          airports.Country,
                airports.Name,
                COUNT(*) AS Total_Rec
FROM            ontime
INNER JOIN      airports
ON              ontime.Origin = airports.IATA
WHERE           ontime.Month = 1 AND
                airports.Country != 'United States' AND
                ontime.Cancelled != 1 AND
                ontime.Diverted != 1
GROUP BY        airports.Country,
                airports.Name
ORDER BY        Total_Rec DESC;
```

Output

```
Country         Name                                    Total_Rec
Puerto Rico     Luis Munoz Marin International Airport  2142
Virgin Islands  Cyril E. King Airport                   304
Puerto Rico     Rafael Hernandez Airport                100
Puerto Rico     Mercedita Airport                       61
Virgin Islands  Henry E Rohlsen Airport                 48
```

Python Code

```
%sql USE airline_ontime;

# If you were having difficulty getting the data from the
# database into a dataframe, you could have loaded only the
#necessary columns from "ontime" as I have done below...
result4 = %sql SELECT Origin, Cancelled, Diverted FROM ontime
WHERE Month = 1

q5 = result4.DataFrame()
result5 = %sql SELECT * FROM airports;
q5a = result5.DataFrame()

q5_merged =
q5.merge(q5a,left_on='Origin',right_on='IATA',how='inner')

# Checking the data types reveals that 'Diverted' is actually
# stored as a string...
q5_merged.dtypes

q5_grouped=q5_merged[(q5_merged['Country'].str.strip() !=
'United States') & (q5_merged['Cancelled'] != 1) &
(q5_merged['Diverted'] !=
'1')].groupby(['Country','Name']).agg({'Country':np.size})

q5_grouped.columns.values[0]='COUNT(*)'
q5_grouped=q5_grouped.reset_index()
q5_grouped.sort_values('COUNT(*)',ascending=False)
```