Problem Set 2:Wrangling Subway Data.

****

1.) Number of Rainy Days.

```
q = """
SELECT
count(*)
FROM
weather_data
WHERE rain =1;
"""
```

2.) Temp on Foggy and NonFoggy Days.

```
q = """

select fog,MAX(maxtempi)
from weather_data
GROUP BY fog;

"""
```

3.) Mean Temp on Weekends.

```
q = """
select AVG(meantempi)
from weather_data
WHERE strftime('%w', date) IN('0', '6');
"""
```

4.) Mean Temp on Rainy Days.

```
q = """
select AVG(mintempi)
FROM weather_data
WHERE rain =1
and mintempi > 55;
"""
```

5.) Fixing Turnstile Data.

```
     for name in_filenames:
         solutionName = "updated_" + name
         originalData = open(name, 'r')
         solutionDataFile = open(solutionName, 'w')

         solutionData = csv.writer(solutionDataFile
)
         csvOriginalData = csv.reader(originalData)

         for row in csvOriginalData:
            baseData = row[0:3]
            date = row[3::5]
            initHour = row[4::5]
            typeTicket = row[5::5]
            initCode = row[6::5]
            endCode = row[7::5]
            for entry in zip(date, initHour, typeTi
cket, initCode, endCode):
                solutionData.writerow(baseData + li
st(entry))

        originalData.close()
        solutionDataFile.close()
```

6.) Combining Turnstile Data

```
       with open(output_file, 'w') as master_file
          master_file.write('C/A,UNIT,SCP,DATEn,TIM
En,DESCn,ENTRIESn,EXITSn\n')
          for filename in filenames:
                     with open(filename,'rb') as f:

                            for row in f:
                             master_file.write(row)
```

7.) Filtering Irregular Data.

```
     turnstile_data = pandas.read_csv(filename)

     return turnstile_data[turnstile_data['DESCn']
== 'REGULAR']
```

8.) Get Hourly Entries.

```
    df['ENTRIESn_hourly'] = df.ENTRIESn.shift(-1) -
 df.ENTRIESn
    df.ENTRIESn_hourly = df.ENTRIESn_hourly.shift(1
)
    df.ENTRIESn_hourly = df.ENTRIESn_hourly.fillna(
1)
```

9.) Get Hourly Exits.

```
    df['EXITSn_hourly'] = df['EXITSn'] - df['EXITSn
'].shift()
    return df.fillna(0)
```

10.) Time to Hour.

```
     return int(time.split(':')[0])
```

11.) Reformat Subway Dates

```
     date_formatted = datetime.datetime.strptime(da
te, "%m-%d-%y").strftime("%Y-%m-%d")

     return date_formatted
```

Problem Set 3: Analyzing Subway Data
****

1.) Exploratory Data Analysis

```
    plt.figure()
    rainy = turnstile_weather['ENTRIESn_hourly'][turnstile_weather['rain']==1].hist(bins=20, alpha = 0.8)
    clear = turnstile_weather['ENTRIESn_hourly'][turnstile_weather['rain']==0].hist(bins=20, alpha = 0.3)
    plt.xlabel('ENTRIESn_hourly', fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.show()
```

2.) Welch's t-Test?

Does entries data from the previous exercise seem normally distributed?
Answer is No.

Can we run Welch T test on entries data? Why or why not?
Answer is tes we can to check if their means are equal.

3.) Mann-Whitney U-Test

```
    rain = turnstile_weather['ENTRIESn_hourly'][turnstile_weather['rain']==1]
    no_rain = turnstile_weather['ENTRIESn_hourly'][turnstile_weather['rain']==0]
    with_rain_mean = rain.mean()
    without_rain_mean = no_rain.mean()
    U, p = scipy.stats.mannwhitneyu(rain, no_rain)
```

4.) Ridership on Rainy vs. Nonrainy Days

Is the distribution of the number of entries statistically different between rainy & non rainy days?

Answer is Yes as explained below.

Rainy days has more ridership.

5.) Linear Regression

Computing the cost function.
****

```
def compute_cost(features, values, theta):
m = len(values)
sum_of_square_errors = np.square(np.dot(features, theta) - values).sum()
cost = sum_of_square_errors / (2*m)

return cost
```

Perform gradient descent.
****

```
def gradient_descent(features, values, theta, alpha, num_iterations):

m = len(values)
cost_history = []

for i in range(num_iterations):
     cost = compute_cost(features, values, theta)
     cost_history.append(cost)
     theta = theta - (alpha/m) * np.dot((np.dot(features,theta) - values),features)
return theta, pandas.Series(cost_history)
```

6.) Plotting Residuals

```
plt.figure()
(turnstile_weather['ENTRIESn_hourly'] - predictions).hist()
plt.show
```

7.) Compute R^2

```
    numerator = np.square(data - predictions).sum()

    mean = np.mean(data)
    denominator = np.square(data - mean).sum()
    r_squared = 1 - (numerator / denominator)
```

Problem Set 4:Visualizing Subway Data

****

1.) Exercise - Visualization 1

```
    plot = ggplot(turnstile_weather, aes('Hour', '
ENTRIESn_hourly')) + geom_bar(alpha=0.8, stat="bar
") + \
           theme(text = element_text(size=30)) + \

           ggtitle('Subway Usage') + xlab('Hour') +
 ylab('Number of Entries')

    return plot
```

2.) Exercise - Visualization 2

```
      plot = ggplot(turnstile_weather, aes(x = 'Ho
ur', y = 'ENTRIESn_hourly',fill ='rain')) + \
               geom_bar(stat="bar") + \
               theme(text = element_text(size=30))
+ \
               ylab("Number of Entries") + \
               xlab("Hour of the day") + \
               ggtitle('Subway ridership on a rainy
 and non rainy day')

     return plot
```

Problem Set 5: MapReduce on Subway Data

```
***************************************

1.) Ridership per station

    riders_per_station_mapper.py

    for line in sys.stdin:

        data = line.strip().split(",")
        if len(data) !=22 or data[6] == 'ENTRIESn_h
ourly':
            continue
        print "{0}\t{1}".format(data[1], data[6])


    riders_per_station_reducer.py

    entries_hourly_count = 0
    old_key = None

    for line in sys.stdin:
        data = line.strip().split("\t")

         if len(data) != 2:
           continue
         this_key, count = data

         if old_key and old_key != this_key:
            print "{0}\t{1}".format(old_key, entrie
s_hourly_count)
            entries_hourly_count = 0


          old_key = this_key
          entries_hourly_count += float(count)

    if old_key != None:
        print "{0}\t{1}".format(old_key, entries_ho
urly_count)


2.) Ridership by Weather Type
```

ridership_by_weather_mapper.py

```
    def format_key(fog, rain):
        return '{}fog-{}rain'.format(
            '' if fog else 'no',
            '' if rain else 'no'
        )


    for line in sys.stdin:
        data = line.strip().split(",");

        if len(data) !=22 or data[6] == "ENTRIESn_h
ourly":
            continue

        print "{0}\t{1}".format(format_key(float(da
ta[14]),float(data[15])), data[6])
        logging.info("{0}\t{1}".format(format_key(f
loat(data[14]),float(data[15])), data[6]))
```

ridership_by_weather_reducer.py

```
    riders = 0      # The number of total riders fo
r this key
    num_hours = 0   # The number of hours with this
 key
    old_key = None
    avg = 0.0


    for line in sys.stdin:
       data = line.strip().split("\t")

       if len(data) !=2:
           continue
       this_key, count = data

       if old_key and old_key != this_key:
           print "{0}\t{1}".format(old_key,avg)
           riders = 0
           num_hours = 0

        old_key = this_key
        riders += float(count)
        num_hours += 1
        avg =  entries / num

    if old_key != None:
       print "{0}\t{1}".format(old_key, avg)
       logging.info("{0}\t{1}".format(old_key, avg)
)
```

3.) busiest_hour_mapper.py

```
    for line in sys.stdin:
        data = line.strip().split(",")
        if len(data) !=22 or data[6] == 'ENTRIESn_h
ourly':
           continue
        print "{0}\t{1}\t{2}\t{3}".format(data[1],d
ata[6],data[2],data[3])
```

    busiest_houre_reducer.py

```
    max_entries = 0
    old_key = None
    datetime = ''

    for line in sys.stdin:
        data = line.strip().split('\t')
        if len(data) != 4:
            continue
        this_key, count, date, time  = data
        count = float(count)

        if old_key and old_key != this_key:
            print "{0}\t{1}\t{2}".format(old_key, d
atetime, max_entries)
            max_entries = 0
            datetime = ''

        old_key = this_key
        if count >= max_entries:
            max_entries = count
            datetime = str(date) + ' ' + str(time)


     if old_key != None:
        print "{0}\t{1}\t{2}".format(old_key, datet
ime, max_entries)
        logging.info("{0}\t{1}\t{2}".format(old_key
, datetime, max_entries))
```

List of Websites used

http://pandas.pydata.org/pandas-docs/stable/tutorials.html

http://www.python-course.eu/numpy.php

https://github.com/allanbreyes/udacity-data-science/blob/master/p1/ps5/

http://matplotlib.org/users/pyplot_tutorial.html