import pandas

import pandasql

def num\_rainy\_days(filename):

'''

This function should run a SQL query on a dataframe of

weather data. The SQL query should return one column and

one row - a count of the number of days in the dataframe where

the rain column is equal to 1 (i.e., the number of days it

rained).

'''

weather\_data = pandas.read\_csv(filename)

q = """

SELECT COUNT(\*) FROM weather\_data WHERE rain = 1;

"""

#Execute your SQL command against the pandas frame

rainy\_days = pandasql.sqldf(q.lower(), locals())

return rainy\_days

count(\*)

1. 10

def max\_temp\_aggregate\_by\_fog(filename):

'''

This function should run a SQL query on a dataframe of

weather data. The SQL query should return two columns and

two rows - whether it was foggy or not (0 or 1) and the max

maxtempi for that fog value (i.e., the maximum max temperature

for both foggy and non-foggy days).

weather\_data = pandas.read\_csv(filename)

q = """

SELECT fog, MAX(cast(maxtempi as integer)) FROM weather\_data GROUP BY fog;

"""

#Execute your SQL command against the pandas frame

foggy\_days = pandasql.sqldf(q.lower(), locals())

return foggy\_days

fog max(cast(maxtempi as integer))

0 0 86

1 1 81

def avg\_weekend\_temperature(filename):

'''

This function should run a SQL query on a dataframe of

weather data. The SQL query should return one column and

one row - the average meantempi on days that are a Saturday

or Sunday (i.e., the the average mean temperature on weekends).

SELECT AVG(cast (meantempi as integer))

FROM weather\_data

WHERE cast(strftime('%w', date) as integer) =0 OR cast(strftime('%w', date) as integer)=6

avg(cast (meantempi as integer))

0 65.111111

def avg\_min\_temperature\_on\_rainy\_days(filename):

'''

This function should run a SQL query on a dataframe of

weather data. More specifically you want to find the average

minimum temperature (mintempi column of the weather dataframe) on

rainy days where the minimum temperature is greater than 55 degrees.

weather\_data = pandas.read\_csv(filename)

q = """

SELECT AVG(cast(mintempi as integer))

FROM weather\_data

WHERE cast(mintempi as integer)>55 AND rain = 1

"""

#Execute your SQL command against the pandas frame

avg\_min\_temp\_rainy = pandasql.sqldf(q.lower(), locals())

return avg\_min\_temp\_rainy

avg(cast(mintempi as integer))

0 61.25

import csv

def fix\_turnstile\_data(filenames):

'''

Filenames is a list of MTA Subway turnstile text files. A link to an example

MTA Subway turnstile text file can be seen at the URL below:

http://web.mta.info/developers/data/nyct/turnstile/turnstile\_110507.txt

As you can see, there are numerous data points included in each row of the

a MTA Subway turnstile text file.

You want to write a function that will update each row in the text

file so there is only one entry per row. A few examples below:

A002,R051,02-00-00,05-28-11,00:00:00,REGULAR,003178521,001100739

A002,R051,02-00-00,05-28-11,04:00:00,REGULAR,003178541,001100746

A002,R051,02-00-00,05-28-11,08:00:00,REGULAR,003178559,001100775

Write the updates to a different text file in the format of "updated\_" + filename.

For example:

1) if you read in a text file called "turnstile\_110521.txt"

2) you should write the updated data to "updated\_turnstile\_110521.txt"

The order of the fields should be preserved. Remember to read through the

Instructor Notes below for more details on the task.

In addition, here is a CSV reader/writer introductory tutorial:

http://goo.gl/HBbvyy

You can see a sample of the turnstile text file that's passed into this function

and the the corresponding updated file by downloading these files from the resources:

Sample input file: turnstile\_110528.txt

Sample updated file: solution\_turnstile\_110528.txt

'''

for name in filenames:

# your code here

# create file input object f\_in to work with

f\_in = open (name, 'r')

# create file output object f\_out to write to the new file

f\_out = open ('updated\_'+name, 'w')

#creater CSV readers and writers based on file objects

reader\_in = csv.reader(f\_in, delimiter = ',')

writer\_out = csv.writer(f\_out, delimiter =',')

for line in reader\_in:

line\_len = len(line)

c1 = line [0]

c2 = line [1]

c3 = line [2]

i = 3

while i < line\_len:

line\_updated = [c1, c2, c3, line[i], line[i+1], line[i+2], line[i+3], line[i+4]]

writer\_out.writerow(line\_updated)

i = i +5

f\_in.close()

f\_out.close()

A002,R051,02-00-00,05-21-11,00:00:00,REGULAR,003169391,001097585,05-21-11,04:00:00,REGULAR,003169415,001097588,05-21-11,08:00:00,REGULAR,003169431,001097607,05-21-11,12:00:00,REGULAR,003169506,001097686,05-21-11,16:00:00,REGULAR,003169693,001097734,05-21-11,20:00:00,REGULAR,003169998,001097769,05-22-11,00:00:00,REGULAR,003170119,001097792,05-22-11,04:00:00,REGULAR,003170146,001097801

This 1 line makes 8 lines

1. A002,R051,02-00-00,05-21-11,00:00:00,REGULAR,003169391,001097585
2. A002,R051,02-00-00,05-21-11,04:00:00,REGULAR,003169415,001097588
3. A002,R051,02-00-00,05-21-11,08:00:00,REGULAR,003169431,001097607
4. A002,R051,02-00-00,05-21-11,12:00:00,REGULAR,003169506,001097686
5. A002,R051,02-00-00,05-21-11,16:00:00,REGULAR,003169693,001097734
6. A002,R051,02-00-00,05-21-11,20:00:00,REGULAR,003169998,001097769
7. A002,R051,02-00-00,05-22-11,00:00:00,REGULAR,003170119,001097792
8. A002,R051,02-00-00,05-22-11,04:00:00,REGULAR,003170146,001097801

import csv

def create\_master\_turnstile\_file(filenames, output\_file):

'''

Write a function that takes the files in the list filenames, which all have the

columns 'C/A, UNIT, SCP, DATEn, TIMEn, DESCn, ENTRIESn, EXITSn', and consolidates

them into one file located at output\_file. There should be ONE row with the column

headers, located at the top of the file. The input files do not have column header

rows of their own.

For example, if file\_1 has:

line 1 ...

line 2 ...

and another file, file\_2 has:

line 3 ...

line 4 ...

line 5 ...

We need to combine file\_1 and file\_2 into a master\_file like below:

'C/A, UNIT, SCP, DATEn, TIMEn, DESCn, ENTRIESn, EXITSn'

line 1 ...

line 2 ...

line 3 ...

line 4 ...

line 5 ...

'''

with open(output\_file, 'w') as master\_file:

master\_file.write('C/A,UNIT,SCP,DATEn,TIMEn,DESCn,ENTRIESn,EXITSn\n')

for filename in filenames:

with open(filename, 'r') as datafile:

datafile.readline()

for line in datafile:

master\_file.write(line)

C/A,UNIT,SCP,DATEn,TIMEn,DESCn,ENTRIESn,EXITSn

A002,R051,02-00-00,05-21-11,04:00:00,REGULAR,003169415,001097588

A002,R051,02-00-00,05-21-11,08:00:00,REGULAR,003169431,001097607

A002,R051,02-00-00,05-21-11,12:00:00,REGULAR,003169506,001097686

A002,R051,02-00-00,05-21-11,16:00:00,REGULAR,003169693,001097734

A002,R051,02-00-00,05-21-11,20:00:00,REGULAR,003169998,001097769

A002,R051,02-00-00,05-22-11,00:00:00,REGULAR,003170119,001097792

import pandas

def filter\_by\_regular(filename):

'''

This function should read the csv file located at filename into a pandas dataframe,

and filter the dataframe to only rows where the 'DESCn' column has the value 'REGULAR'.

For example, if the pandas dataframe is as follows:

,C/A,UNIT,SCP,DATEn,TIMEn,DESCn,ENTRIESn,EXITSn

0,A002,R051,02-00-00,05-01-11,00:00:00,REGULAR,3144312,1088151

1,A002,R051,02-00-00,05-01-11,04:00:00,DOOR,3144335,1088159

2,A002,R051,02-00-00,05-01-11,08:00:00,REGULAR,3144353,1088177

3,A002,R051,02-00-00,05-01-11,12:00:00,DOOR,3144424,1088231

The dataframe will look like below after filtering to only rows where DESCn column

has the value 'REGULAR':

0,A002,R051,02-00-00,05-01-11,00:00:00,REGULAR,3144312,1088151

2,A002,R051,02-00-00,05-01-11,08:00:00,REGULAR,3144353,1088177

'''

turnstile\_data = pandas.read\_csv(filename)

turnstile\_data = turnstile\_data[(turnstile\_data.DESCn == 'REGULAR')]

return turnstile\_data

def get\_hourly\_entries(df):

'''

The data in the MTA Subway Turnstile data reports on the cumulative

number of entries and exits per row. Assume that you have a dataframe

called df that contains only the rows for a particular turnstile machine

(i.e., unique SCP, C/A, and UNIT). This function should change

these cumulative entry numbers to a count of entries since the last reading

(i.e., entries since the last row in the dataframe).

More specifically, you want to do two things:

1) Create a new column called ENTRIESn\_hourly

2) Assign to the column the difference between ENTRIESn of the current row

and the previous row. If there is any NaN, fill/replace it with 1.

You may find the pandas functions shift() and fillna() to be helpful in this exercise.

Examples of what your dataframe should look like at the end of this exercise:

C/A UNIT SCP DATEn TIMEn DESCn ENTRIESn EXITSn ENTRIESn\_hourly

0 A002 R051 02-00-00 05-01-11 00:00:00 REGULAR 3144312 1088151 1

1 A002 R051 02-00-00 05-01-11 04:00:00 REGULAR 3144335 1088159 23

2 A002 R051 02-00-00 05-01-11 08:00:00 REGULAR 3144353 1088177 18

3 A002 R051 02-00-00 05-01-11 12:00:00 REGULAR 3144424 1088231 71

4 A002 R051 02-00-00 05-01-11 16:00:00 REGULAR 3144594 1088275 170

5 A002 R051 02-00-00 05-01-11 20:00:00 REGULAR 3144808 1088317 214

6 A002 R051 02-00-00 05-02-11 00:00:00 REGULAR 3144895 1088328 87

7 A002 R051 02-00-00 05-02-11 04:00:00 REGULAR 3144905 1088331 10

8 A002 R051 02-00-00 05-02-11 08:00:00 REGULAR 3144941 1088420 36

9 A002 R051 02-00-00 05-02-11 12:00:00 REGULAR 3145094 1088753 153

10 A002 R051 02-00-00 05-02-11 16:00:00 REGULAR 3145337 1088823 243

...

...

'''

df['ENTRIESn\_hourly'] = (df['ENTRIESn']-df['ENTRIESn'].shift(1)).fillna(1)

return df

(1088177-1088159=18)

def get\_hourly\_exits(df):

'''

The data in the MTA Subway Turnstile data reports on the cumulative

number of entries and exits per row. Assume that you have a dataframe

called df that contains only the rows for a particular turnstile machine

(i.e., unique SCP, C/A, and UNIT). This function should change

these cumulative exit numbers to a count of exits since the last reading

(i.e., exits since the last row in the dataframe).

More specifically, you want to do two things:

1) Create a new column called EXITSn\_hourly

2) Assign to the column the difference between EXITSn of the current row

and the previous row. If there is any NaN, fill/replace it with 0.

You may find the pandas functions shift() and fillna() to be helpful in this exercise.

Example dataframe below:

Unnamed: 0 C/A UNIT SCP DATEn TIMEn DESCn ENTRIESn EXITSn EXITSn\_hourly

0 0 A002 R051 02-00-00 05-01-11 00:00:00 REGULAR 3144312 1088151 0

1 1 A002 R051 02-00-00 05-01-11 04:00:00 REGULAR 3144335 1088159 8

2 2 A002 R051 02-00-00 05-01-11 08:00:00 REGULAR 3144353 1088177 18

3 3 A002 R051 02-00-00 05-01-11 12:00:00 REGULAR 3144424 1088231 54

4 4 A002 R051 02-00-00 05-01-11 16:00:00 REGULAR 3144594 1088275 44

5 5 A002 R051 02-00-00 05-01-11 20:00:00 REGULAR 3144808 1088317 42

6 6 A002 R051 02-00-00 05-02-11 00:00:00 REGULAR 3144895 1088328 11

7 7 A002 R051 02-00-00 05-02-11 04:00:00 REGULAR 3144905 1088331 3

8 8 A002 R051 02-00-00 05-02-11 08:00:00 REGULAR 3144941 1088420 89

9 9 A002 R051 02-00-00 05-02-11 12:00:00 REGULAR 3145094 1088753 333

'''

df['EXITSn\_hourly'] = (df['EXITSn']-df['EXITSn'].shift(1)).fillna(0)

return df

def time\_to\_hour(time):

'''

Given an input variable time that represents time in the format of:

"00:00:00" (hour:minutes:seconds)

Write a function to extract the hour part from the input variable time

and return it as an integer. For example:

1) if hour is 00, your code should return 0

2) if hour is 01, your code should return 1

3) if hour is 21, your code should return 21

Please return hour as an integer.

'''

hour = int(time[0:2])

return hour

0 UNIT DATEn TIMEn DESCn ENTRIESn\_hourly EXITSn\_hourly Hour

0 0 R022 05-01-11 00:00:00 REGULAR 0.0 0.0 0

1 1 R022 05-01-11 04:00:00 REGULAR 562.0 173.0 4

2 2 R022 05-01-11 08:00:00 REGULAR 160.0 194.0 8

3 3 R022 05-01-11 12:00:00 REGULAR 820.0 1025.0 12

4 4 R022 05-01-11 16:00:00 REGULAR 2385.0 1954.0 16

5 5 R022 05-01-11 20:00:00 REGULAR 3631.0 2010.0 20

6 6 R022 05-02-11 00:00:00 REGULAR 1788.0 733.0 0

import datetime

def reformat\_subway\_dates(date):

'''

The dates in our subway data are formatted in the format month-day-year.

The dates in our weather underground data are formatted year-month-day.

In order to join these two data sets together, we'll want the dates formatted

the same way. Write a function that takes as its input a date in the MTA Subway

data format, and returns a date in the weather underground format.

Hint:

There are a couple of useful functions in the datetime library that will

help on this assignment, called strptime and strftime.

More info can be seen here and further in the documentation section:

http://docs.python.org/2/library/datetime.html#datetime.datetime.strptime

'''

date\_formatted = str(datetime.datetime.strptime(date, "%m-%d-%y"))[:10]

return date\_formatted

Unnamed: 0 UNIT DATEn TIMEn DESCn ENTRIESn\_hourly EXITSn\_hourly Hour

0 0 R022 2011-05-01 00:00:00 REGULAR 0.0 0.0 0

1 1 R022 2011-05-01 04:00:00 REGULAR 562.0 173.0 4

2 2 R022 2011-05-01 08:00:00 REGULAR 160.0 194.0 8

3 3 R022 2011-05-01 12:00:00 REGULAR 820.0 1025.0 12

4 4 R022 2011-05-01 16:00:00 REGULAR 2385.0 1954.0 16

5 5 R022 2011-05-01 20:00:00 REGULAR 3631.0 2010.0 20