Programming for Data Science - DS-GA 1007

Homework 10: Pandas - Part III

Due Date: Monday 12/02, 11:59 PM ¶

We will explore some aspects of the pandas package along with SQL. We will need to query a database, work with dates, group and pivot tables. By completing Homework 10, you should take away...

- Pratice converting strings with month, day, year to a date data type
- Gain experience with the grouping and pivoting operations in pandas

We will use data provided by the New York City Department of Health and Mental Hygiene. The table contains information about inspections of each of New York City's 24,000 restaurants. Each restaurant receives a grade of A,B,C based on a score that tallies points from health violations. Please see the following link (https://www1.nyc.gov/assets/doh/downloads/pdf/rii/how-we-score-grade.pdf) for information.

Submission Instructions

Please submit your notebook through the Assignments tab. Additionally, you need to submit a copy to Gradescope. Follow these steps

1. Download as HTML (`File->Download As->HTML(.html)`).
2. Open the HTML in the browser. Print to .pdf
3. Upload to Gradescope. Tag your answers.

Note that

- Please map your answers to our questions. Otherwise you may lose points. Please see the rubric below.
- You should break long lines of code into multiple lines. Otherwise your code will extend out of view from the cell. Consider using `\` followed by a new line.
- For each textual response, please include relevant code that informed your response. For each plotting question, please include the code used to generate the plot.
- You should not display large output cells such as all rows of a table. Instead convert the input cell from Code to Markdown back to Code to remove the output cell.

Collaboration Policy

Data science is a collaborative activity. While you may talk with others about the homework, we ask that you **write your solutions individually**. If you do discuss the assignments with others please **include their names** at the top of your solution.

Rubric

Question	Points
Gradescope	2
Question 0	3
Question 1	2
Question 2	1
Question 3	1

Question	Points
Question 4	1
Question 5	4

In [ ]:
```
from IPython.display import Image

import sqlalchemy
import pymysql

import numpy as np
import pandas as pd
pd.options.display.max_rows = 20
pd.options.display.max_columns = 15
pd.set_option('precision', 2)

import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (8, 5)
plt.rcParams['figure.dpi'] = 150
```

Question 0. (3 points) Use the connection details below to access the MySQL database containing the data.

In [ ]:
```
# RUN
database_handle = sqlalchemy.create_engine('mysql+pymysql://dbreader:WuE8c1TF@
35.245.55.145:3306/cp126', echo=False)
```

You can access the data in the table `nyc`. Use the handle to the database, `database_handle`, to submit a SQL query with pandas `read_sql`.

1. Determine the number of rows in the table `nyc`. Call it `count`.

In [ ]:
```
count_query = ...

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
# TEST
count = pd.read_sql(count_query, database_handle)
```

In [ ]:
```

```

2. Determine the distinct number of grades in the table `nyc`. Call it `grades`.

In [ ]:
```
grades_query = ...

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
# Test
grades = pd.read_sql(grades_query, database_handle)
```

In [ ]:
```

```

3. Read all rows from the `GRADE DATE`, `GRADE` and `BORO` column into a DataFrame called `inspection`. Note that `GRADE DATE` needs to be enclosed in tick marks (```) not apostrophes.

In [ ]:
```
inspection_query = ...

# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
# TEST
inspection = pd.read_sql(inspection_query, database_handle)
assert inspection.shape[0] == 381834
assert inspection.shape[1] == 3
```

Question 1. (2 points) We have to process the data to remove invalid entries.

Note that pandas has a function called `to_datetime` that converts strings to date data type. If the string does not match the format, then the function returns `NaT`, which is similar to `NaN`.

Here, we replace the `GRADE DATE` column with `DATE` column containing date data type.

In [ ]:
```
# RUN
inspection_cleaned = inspection \
    .assign(DATE = pd.to_datetime(inspection['GRADE DATE'], errors='coerce', f
ormat="%m/%d/%Y")) \
    .drop(columns='GRADE DATE') \
    .replace(to_replace=pd.NaT, value=np.NaN) \
    .dropna()
```

We learned that `inspection` table contains over three hundred distinct entries in the grade column. Filter `inspection_cleaned` to contain rows with grade A,B, or C in the `GRADE` column.

In [ ]:
```
# YOUR CODE HERE
raise NotImplementedError()
```

Some of the entries in `BORO` column are `Missing`. Filter `inspection_cleaned` to contain rows with boroughs 'QUEENS', 'BROOKLYN', 'BRONX', 'MANHATTAN', or 'STATEN ISLAND' in the `BORO` column

In [ ]:
```
# YOUR CODE HERE
raise NotImplementedError()
```

How many rows were removed from `inspection`?

In [ ]:
```
# RUN
print('Number of Rows Before Cleaning:',len(inspection))
print('Number of Rows After Cleaning:',len(inspection_cleaned))
print('Number of Removed Rows :',len(inspection) - len(inspection_cleaned))
```

In [ ]:
```

```

Question 2. (1 points) Write a function called `filter_boro` with input

- DataFrame `inspection_cleaned`
- borough 'QUEENS', 'BROOKLYN', 'BRONX', 'MANHATTAN', 'STATEN ISLAND' as string or `None`

and output

- DataFrame containing rows in the borough.

The default borough should be `None` which returns a copy of the entire table.

In [ ]:
```
def filter_boro(restaurants, boro = None):
    # YOUR CODE HERE
    raise NotImplementedError()

boro = filter_boro(inspection_cleaned, 'STATEN ISLAND')
```

In [ ]:
```
# TEST
assert len(boro) == 2969
```

In [ ]:
```

```

Question 3. (1 points) Write a function called `group_years` with input `inspection_cleaned` and output a pandas DataFrameGroupBy object with entries grouped according to years in `DATE` column.

In [ ]:
```
def group_years(restaurants):
    # YOUR CODE HERE
    raise NotImplementedError()

year_groups = group_years(inspection_cleaned)
```

In [ ]:
```
# TEST
assert len(list(year_groups.groups.keys())) == 6
```

In [ ]:
```

```

Question 4. (1 points) Write a function called `restaurant_grades` with inputs

- DataFrameGroupBy object from Question 2
- year

and outputs

- number of restaurants graded as A,B, or C in year

The output should be a pandas Series with index A,B,C. If there is not information for that year, then the function should return value 0 for A,B,C.

In [ ]:
```
def restaurant_grades(data_frame_groupby, year):
    # YOUR CODE HERE
    raise NotImplementedError()
```

In [ ]:
```
# RUN
for year,grp in year_groups:
    numb_per_grade =restaurant_grades(year_groups,year)
    print(year)
    for i in numb_per_grade.index:
        print(i,': ',numb_per_grade.loc[i])
```

In [ ]:
```
# TEST
assert np.all(restaurant_grades(year_groups, 2016).values == [17446,  3771,
894])
```

In [ ]:
```

```

Question 5. (4 points) Generate three bar plots comparing the number of grades A,B,C between 2013 and 2018. Distinguish the plots by borough

1. NYC: restaurants in all boroughs
2. Brooklyn: restaurants in the Brooklyn borough
3. Manhattan: restaurants in the Manhattan borough

We can combine the functions `filter_boro` , `group_years` and `restaurant_grades` to generate the bar charts.

In [ ]:
```
boros = [None, 'BROOKLYN', 'MANHATTAN']
years = [2013, 2014, 2015, 2016, 2017, 2018]
grades = ['A','B','C']

fig, axes = plt.subplots(nrows=1, ncols=3, figsize = (10,5))
plt.subplots_adjust(wspace=0.3)

for idx, entry in enumerate(boros):
    df = pd.DataFrame(data = np.zeros((len(years), len(grades))),
                      index = years,
                      columns= grades)

    for year, grade in [(y,g) for y in years for g in grades]:
        df.loc[year,grade] = (inspection_cleaned
                                  .pipe(filter_boro, entry)
                                  .pipe(group_years)
                                  .pipe(restaurant_grades, year)
                                  .get(grade, default = 0) )

    df.plot(kind='bar', ax=axes[idx], title=entry if entry else "NYC")
```

Note that we use `pipe` to organize the operations on `inspection_cleaned`. Please read through the code to understand putting together the three functions from Question 2, Question 3, and Question 4.

Another approach is pivot tables. Fill in the missing code below using the `pivot_table` function in pandas. You should create the same bar charts.

In [ ]:
```
boros = [None, 'BROOKLYN', 'MANHATTAN']

fig, axes = plt.subplots(nrows=1, ncols=3, figsize = (10,5))
plt.subplots_adjust(wspace=0.3)

for idx, entry in enumerate(boros):
    df_filtered = filter_boro(inspection_cleaned, entry)
    df_filtered['DATE'] = df_filtered['DATE'].map(lambda y: y.year)
    df_pivot = ...

    # YOUR CODE HERE
    raise NotImplementedError()
    df_pivot.plot(kind='bar', ax=axes[idx], title=entry if entry else "NYC")
    plt.savefig('plots.png')
```

In [ ]:
```
# RUN
Image('plots.png')
```

Based on the plots, does the quality of restaurants in New York City appear to be improving over years?