Version 1.0.1

Pandas basics

Hi! In this programming assignment you need to refresh your `pandas` knowledge. You will need to do several `groupby` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.groupby.html)s and `join` ()`s to solve the task.

In [ ]:

```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
%matplotlib inline

from grader import Grader
```

In [ ]:

```
DATA_FOLDER = '../readonly/final_project_data/'

transactions    = pd.read_csv(os.path.join(DATA_FOLDER, 'sales_train.csv.gz'))
items           = pd.read_csv(os.path.join(DATA_FOLDER, 'items.csv'))
item_categories = pd.read_csv(os.path.join(DATA_FOLDER, 'item_categories.csv'))
shops           = pd.read_csv(os.path.join(DATA_FOLDER, 'shops.csv'))
```

The dataset we are going to use is taken from the competition, that serves as the final project for this course. You can find complete data description at the competition web page (https://www.kaggle.com/c/competitive-data-science-final-project/data). To join the competition use this link (https://www.kaggle.com/t/1ea93815dca248e99221df42ebde3540).

Grading

We will create a grader instace below and use it to collect your answers. When function `submit_tag` is called, grader will store your answer *locally*. The answers will *not* be submited to the platform immediately so you can call `submit_tag` function as many times as you need.

When you are ready to push your answers to the platform you should fill your credentials and run `submit` function in the last paragraph of the assignment.

In [ ]:

```
grader = Grader()
```

Task

Let's start with a simple task.

0. **Print the shape of the loaded dataframes and use `df.head` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.head.html) function to print several rows. Examine the features you are given.**

In [ ]:

```
transactions.shape
items.shape
item_categories.shape
shops.shape
transactions.head(1)
items.head(2)
shops.head(2)
```

Now use your `pandas` skills to get answers for the following questions.

The first question is:

1. **What was the maximum total revenue among all the shops in September, 2014?**

- Hereinafter *revenue* refers to total sales minus value of goods returned.

Hints:

- Sometimes items are returned, find such examples in the dataset.
- It is handy to split `date` field into [`day`, `month`, `year`] components and use `df.year == 14` and `df.month == 9` in order to select target subset of dates.
- You may work with `date` feature as with srings, or you may first convert it to `pd.datetime` type with `pd.to_datetime` function, but do not forget to set correct `format` argument.

The data is transformed into datetime type

In [ ]:

```
transactions['date'] = pd.to_datetime(transactions['date'],format='%d.%m.%Y')
transactions = transactions.set_index('date')
```

In [ ]:

```
df = transactions['2014-09']
df.head()
```

In [ ]:

```
df.insert(2, column='itemsum', value = df['item_price'].mul(df['item_cnt_day']))
df.head()
```

In [ ]:

```
groupby_shop = df['itemsum'].groupby(df['shop_id'])
k = groupby_shop.sum()
```

In [ ]:

```
max_revenue = k.max()
grader.submit_tag('max_revenue', max_revenue)
```

Great! Let's move on and answer another question:

2. **What item category generated the highest revenue in summer 2014?**

- Submit `id` of the category found.
- Here we call "summer" the period from June to August.

Hints:

- Note, that for an object `x` of type `pd.Series`: `x.argmax()` returns **index** of the maximum element. `pd.Series` can have non-trivial index (not `[1, 2, 3, ...]`).

In [ ]:

```
df_summer = transactions['2014-06':'2014-08']
df_summersum = df_summer['item_price'].mul(df_summer['item_cnt_day'])
```

In [ ]:

```
df_summer.insert(2, column='itemsum', value=df_summersum)
df_summer.size
```

In [ ]:

```
items.head()
```

In [ ]:

```
def func1(x):
    return items['item_category_id'][x]

item_cate = df_summer['item_id'].map(func1)
item_cate
```

In [ ]:

```
df_summer.insert(2, column='categories', value=item_cate)
df_summer.head()
```

In [ ]:

```
grouped = df_summer['itemsum'].groupby(df_summer['categories'])
s = grouped.sum()
s
```

In [ ]:

```
category_id_with_max_revenue = s.argmax()
grader.submit_tag('category_id_with_max_revenue', category_id_with_max_revenue)
```

3. **How many items are there, such that their price stays constant (to the best of our knowledge)**

during the whole period of time?

- Let's assume, that the items are returned for the same price as they had been sold.

In [ ]:

```
grouped2 = transactions['item_price'].groupby(transactions['item_id'])
grouped2 = grouped2.unique()
grouped2.head(5)
```

In [ ]:

```
def cal_num(x):
    a = len(x)
    if a > 1:
        a = 0
    return a
grouped3 = grouped2.map(cal_num)
```

In [ ]:

```
num_items_constant_price = grouped3.sum()
grader.submit_tag('num_items_constant_price', num_items_constant_price)
```

Remember, the data can sometimes be noisy.

4. **What was the variance of the number of sold items per day sequence for the shop with `shop_id = 25` in December, 2014? Do not count the items, that were sold but returned back later.**

- Fill `total_num_items_sold` and `days` arrays, and plot the sequence with the code below.
- Then compute variance. Remember, there can be differences in how you normalize variance (biased or unbiased estimate, see link (https://math.stackexchange.com/questions/496627/the-difference-between-unbiased-biased-estimator-variance)). Compute ***unbiased*** estimate (use the right value for `ddof` argument in `pd.var` or `np.var`).

In [ ]:

```
df_sold = transactions['2014-12']
df_sold.head()
```

In [ ]:

```
shop_id = 25
df_soldshop = df_sold.loc[df_sold['shop_id'] == shop_id]
```

In [ ]:

```
df_soldshop.insert(2, column='date', value=df_soldshop.index)
df_soldshop
```

In [ ]:

```
groupbydate = df_soldshop['item_cnt_day'].groupby(df_soldshop['date']);
s = groupbydate.sum()
s.head
```

In [ ]:

```
total_num_items_sold = np.array(s)
days = np.array(s.index)

# Plot it
plt.plot(days, total_num_items_sold)
plt.ylabel('Num items')
plt.xlabel('Day')
plt.title("Daily revenue for shop_id = 25")
plt.show()
```

In [ ]:

```
total_num_items_sold_var = np.var(total_num_items_sold,ddof=1)
grader.submit_tag('total_num_items_sold_var', total_num_items_sold_var)
```

Authorization & Submission

To submit assignment to Cousera platform, please, enter your e-mail and token into the variables below. You can generate token on the programming assignment page. *Note:* Token expires 30 minutes after generation.

In [ ]:

```
STUDENT_EMAIL = 'yaohongxiang543@sina.com'
STUDENT_TOKEN = 'RBGJYj55bn8kX4ZC'
grader.status()
```

In [ ]:

```
grader.submit(STUDENT_EMAIL, STUDENT_TOKEN)
```

Well done! :)