Exercise for Week 12: Association Analysis

This week's work will be a little different (and a little shorter) than the formal "Assignments" for the other weeks.

This exercise asks functionally a single question. A few hints and functions are provided which should pave the way to use the apriori algorithm to find the two (2) most frequent collections of items in 250,000 orders.

Ultimately you will be asked to give to give the two lists of the four items that are found together in more than 300 orders.

An Important Note On Grading

A number of these cells could take a long time to run. **To decrease grading time, comment out any cells that take a long time to run.**

The Data

The data today is a subset of data from the "Insta-Cart" dataset on Kaggle

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

```
file_path = "../resource/asnlib/publicdata/insta_cart_subset.csv"

data = pd.read_csv(file_path)
data.head()
```

As you can see above, the data consists of two columns, an `order_id` and a `product_id`.

Looking at the first 5 rows, we can see that order #7 consisted of items 34050, and 46802.

```
unique_orders = len(data.order_id.unique())
print(unique_orders)
```

Count appearances of each product, and delete products which do not appear at least 500 times.

```
count_of_each_product = data.product_id.value_counts()
products_to_keep = count_of_each_product[count_of_each_product > 500].index
products_to_keep[:10]
```

Keep only those products

```
compacted_data = data[data.product_id.isin(products_to_keep)]
compacted_data.shape
```

Helper Function

The below function finds all of the product groupings of a specified size, and counts how many times they appear.

The solution for this exercise could theoretically be found by running "`find_groups_of_size_n(compacted_data, 4)`". However, without a fantastic amount of computing power and memory, that would take a very long time and/or result in a memory error.

Thus the trick to this exercise is determining how to cut-down the size of the data being investigated.

If the data is not cut-down enough, you will see a MemoryError

A few ideas:

1. We know we are looking for product groups of four items. Thus all orders of fewer than 4 items may be disregarded.
2. Use the theories which support the apriori algorithm to further reduce the orders / products which are grouped.

```
import itertools as it
def find_groups_of_size_n(data, size):
    group_by = data.groupby("order_id")['product_id'].unique()
    group_by = group_by.apply(lambda x: sorted(x))
    group_by = pd.DataFrame(group_by)
    def groupings(x):
        return list(it.combinations(x,size))

    group_by['groups'] = group_by['product_id'].apply(groupings)
    counts = pd.Series(list(it.chain.from_iterable(group_by['groups'].values))).value_counts()
    return counts
```

Example of using above function

The below cell is an example of code that should be commented out to decrease grading time

```
%%time
# find_groups_of_size_n(compacted_data,2).head()
```

Work Area Below

Graded Response

Note: the answer is hard coded, your responses should be hard coded as well.

If this cell contains long-running code, it may fail

```
### GRADED
### What are the two sets of the four items which are found together more than 300 times
### in the above dataset.

### Assign one list of four product_ids to set_1,
### and one list of four product_ids to set_2

### For example, after running the cell above "find_groups_of_size_n(compacted_data,2).head()",
### If the questions were about the sets of two items which appear more than 5000 times together
### The answers would be:

### set_1 = [13176, 47209]
### set_2 = [13176, 21137]

### YOUR ANSWER BELOW

set_1 = list((13176, 21137, 27966, 47209))
set_2 = list((13176, 21137, 21903, 47209))
```

Lookup function for product names

If you are interested, a single product ID or a list of product IDs may be looked up using the function below

```
prod_filepath = "../resource/asnlib/publicdata/products.csv"
products = pd.read_csv(prod_filepath)
def product_lookup(product_ids):
    try:
        len(product_ids)
        names = [products[products.product_id == pid].iloc[0,1] for pid in product_ids]
    except:
        names = products[products.product_id == product_ids].iloc[0,1]

    return names
```

```
product_lookup(13176)
```