

Assignment 3

WebSocialytics is a company focused on collecting customer reviews for different products manufactured by different vendors and sold by different retailers

WebSocialytics is your imaginary fast-growing social media company that has a business model similar to yelp.

Every review filled out by the customer online and collected and maintained by Socialytics has the fields listed below(though customers might submit messy reviews, inaccurate review, and yet faked (spam) reviews are real challenge as well.

However, for the purpose of this assignment assume we have the following fields for the review form that is filled out by the customer online:

The Potential of Reviews Analytics

Reviews can be used by customers, retailers, and manufacturers to create predictive models (Clustering, Associations, Classification, Sentiment Analysis, etc.) in order to support the decision making process.

For example,

- How likely a certain product will receive a bad rating in certain zip code
- Clustering of the products based on their ratings
- Analysis of successful marketing campaigns by retailers/manufacturers
- Increase cross-selling by retailers
- How likely an existing customer will buy a newly introduced product to the market
- When customers buy the newly introduced product to the market, what other products do they tend to buy?

Data Preparation & Analysis

The dataset of the reviews given to us is in a plain textfile.

There are a number of tasks that we need to work in order to prepare and present the data in a form that could be used by a data analytics tool or database engine.

We will use Python to prepare and process the given dataset file.

Steps to consider:

1. Inspect and Analyze the data
2. Process and Prepare the data
3. Present the data

Review Form Fields

Product/ModelName:
Product/Category:
Product/Price:
Retailer/Name:
Retailer/ZipCode:
Retailer/City:
Retailer/State:
Retailer/On Sale:
Manufacturer/Name:
Manufacturer/Rebate:
Review/UserId:
Review/Rating:
Review/Date:
Review/Text:

Example of Review Form Filled out

ProductModelName: Galaxy S4
ProductCategory: Smart Phone
ProductPrice: 499
RetailerName: Bestbuy
RetailerZipCode: 44114
RetailerCity: Cleveland
RetailerState: OH
RetailerOnSale: No
ManufacturerName: Samsung
ManufacturerRebate: No
ReviewUserId: xmm473
ReviewRating: 3

```
ReviewDate: 6/10/2012
ReviewText: overheats after 2 hours
```

Data Set File: CustomerReviews.txt

The dataset that we have received from Socialytics has 100 reviews filled by different customers for different products bought from different retailers and manufactured by different vendors.

The data set is saved in plainntext format in the file CustomerReviews.txt which is available with this assignment.

Different reviews, in the file, are separated by BLANK LINES.

In []:

```
In [1]: import os

import cPickle as pickle

import pandas as pd

import numpy as np

from pandas import DataFrame, Series
```

```
In [2]: # 1. Open the fiel to read it
f = open(r'/Users/Zeeshan/Desktop/PREDICT 420/Week 5/Sync Session 5-20/A

# 2. Read the entire file in one step as a single GIANT string
raw_giant_string_data = f.read()
```

```
In [3]: # Verify the datatype you got?
type(raw_giant_string_data)
```

Out[3]: str

```
In [4]: # 3. Split the GIANT string you read into LIST of lines
raw_list_of_strings_data = raw_giant_string_data.splitlines()
```

```
In [5]: # Verify the datatype you got?
        type(raw_list_of_strings_data)
```

```
Out[5]: list
```

```
In [6]: # 4. Print the raw_list_data
        # How does the data look like in the raw_list_data?
        # Every Line will be represented as an item of type (String) in the result
        raw_list_of_strings_data[:10]
```

```
Out[6]: ['ProductModelName: Samsung TV 60 LED',
        'ProductCategory: TV',
        'ProductPrice: 1200',
        'RetailerName: Bestbuy',
        'RetailerZipCode: 60585',
        'RetailerCity: Naperville',
        'RetailerState: IL',
        'RetailerOnSale: No',
        'ManufacturerName: Samsung',
        'ManufacturerRebate: No']
```

```
In [7]: # Does the data list you got have similar patterns?
        # Did you get a list of strings, where the PATTERN of the string could be
        # (a) 'parameterName: parameterValue'
        # (b) ''
        # Note that '' string is the BLANK line in the textfile
```

```
In [8]: # 5. Create the LOL - a List Of two-item Lists
        # So we can have something like this:
        # [['Product/ModelName', 'Samsung TV 60 LED'],
        #   ['Product/Category', 'TV'],
        #   ['Product/Price', '1200'],
        #   .....
        #   .....]
```

```
In [ ]:
```

```
In [9]: # Empty list: The [] characters denote an empty list.
        # Python evaluates zero-element collections to False.
        # In our data list, the blank line is represented by Empty List []
```

```
raw_list_of_lists_data = []
for row in raw_list_of_strings_data:
    if row:
        raw_list_of_lists_data = raw_list_of_lists_data + [row.split(':
```

```
In [10]: type (raw_list_of_lists_data)
```

```
Out[10]: list
```

```
In [11]: raw_list_of_lists_data[:32]
```

```
Out[11]: [['ProductModelName', 'Samsung TV 60 LED'],
          ['ProductCategory', 'TV'],
          ['ProductPrice', '1200'],
          ['RetailerName', 'Bestbuy'],
          ['RetailerZipCode', '60585'],
          ['RetailerCity', 'Naperville'],
          ['RetailerState', 'IL'],
          ['RetailerOnSale', 'No'],
          ['ManufacturerName', 'Samsung'],
          ['ManufacturerRebate', 'No'],
          ['ReviewUserId', 'abc234'],
          ['ReviewRating', '1'],
          ['ReviewDate', '3/14/2013'],
          ['ReviewText', 'Bluetooth did not work for my sound system'],
          ['ProductModelName', 'Surface 3'],
          ['ProductCategory', 'Tablet'],
          ['ProductPrice', '399'],
          ['RetailerName', 'Hhgregg'],
          ['RetailerZipCode', '90012'],
          ['RetailerCity', 'Los Angeles'],
          ['RetailerState', 'CA'],
          ['RetailerOnSale', 'Yes'],
          ['ManufacturerName', 'Microsoft'],
          ['ManufacturerRebate', 'No'],
          ['ReviewUserId', 'chj787'],
          ['ReviewRating', '5'],
          ['ReviewDate', '7/1/2014'],
          ['ReviewText', 'Much better screen than the samsung galaxy tablet'],
          ['ProductModelName', 'Sony TV 42 LED'],
          ['ProductCategory', 'TV'],
          ['ProductPrice', '800'],
          ['RetailerName', 'Frys']]
```

```
In [12]: # 6. Create a generator Method for our partitions of the reviews in the r
```

```
def partition_generator(reviews_list, n):
    def reviews_partitions():
        for i in xrange(0, len(reviews_list), n):
            yield reviews_list[i:i+n]
    return [i for i in reviews_partitions()]
```

```
In [13]: partitioned_list_of_reviews = partition_generator(raw_list_of_lists_data,
partitioned_list_of_reviews[:32])
```

```
Out[13]: [[['ProductModelName', 'Samsung TV 60 LED'],
['ProductCategory', 'TV'],
['ProductPrice', '1200'],
['RetailerName', 'Bestbuy'],
['RetailerZipCode', '60585'],
['RetailerCity', 'Naperville'],
['RetailerState', 'IL'],
['RetailerOnSale', 'No'],
['ManufacturerName', 'Samsung'],
['ManufacturerRebate', 'No'],
['ReviewUserId', 'abc234'],
['ReviewRating', '1'],
['ReviewDate', '3/14/2013'],
['ReviewText', 'Bluetooth did not work for my sound system']],
[['ProductModelName', 'Surface 3'],
['ProductCategory', 'Tablet'],
['ProductPrice', '399'],
['RetailerName', 'Hhgregg'],
['RetailerZipCode', '90012'],
['RetailerCity', 'Los Angeles'],
['RetailerState', 'CA'],
['RetailerOnSale', 'No'],
['ManufacturerName', 'Microsoft'],
['ManufacturerRebate', 'No'],
['ReviewUserId', 'def567'],
['ReviewRating', '2'],
['ReviewDate', '4/15/2013'],
['ReviewText', 'Great tablet, but the screen is a bit blurry']]]
```

```
In [14]: # 7. Create Column Headers
# Read the FIRST list ONLY in the partitioned_list_of_reviews
# and extract from it column_headers for our review_table
# All other lists have the SAME header/pattern names

column_headers_for_reviews_table = [row[0] for row in partitioned_list_of_reviews[0]]
```

```
In [15]: column_headers_for_reviews_table
```

```
Out[15]: [['ProductModelName',  
          'ProductCategory',  
          'ProductPrice',  
          'RetailerName',  
          'RetailerZipCode',  
          'RetailerCity',  
          'RetailerState',  
          'RetailerOnSale',  
          'ManufacturerName',  
          'ManufacturerRebate',  
          'ReviewUserId',  
          'ReviewRating',  
          'ReviewDate',  
          'ReviewText']]
```

```
In [16]:
```

```
# 8. Create a Row in the table for every review  
# Though remember, every review has the pairs (parameterName, parametValu  
# We already know teh parameter NAMES in the Header we created in teh pri  
# so we will read only the SECOND COLUMN of every list because that is te  
# of the parameter
```

```
rows_for_reviews_table = [[col[1] for col in row] for row in partitioned_
```

```
In [17]: rows_for_reviews_table
```

```
Out[17]: [['Samsung TV 60 LED',
          'TV',
          '1200',
          'Bestbuy',
          '60585',
          'Naperville',
          'IL',
          'No',
          'Samsung',
          'No',
          'abc234',
          '1',
          '3/14/2013',
          'Bluetooth did not work for my sound system'],
          ['Surface 3',
          'Tablet',
          '399',
          'Hhgregg',
          '90012',
          'No',
          'Hhgregg',
          'No',
          'abc234',
          '1',
          '3/14/2013',
          'Bluetooth did not work for my sound system']]
```

```
In [18]: # 9. Add the Rows and Columns we created
          # And that will be our hard-work for the Reviews Table

          reviews_table = column_headers_for_reviews_table + rows_for_reviews_table
```

```
In [19]: type(reviews_table)
```

```
Out[19]: list
```



```
In [20]: reviews_table[:3]
```

```
Out[20]: [['ProductModelName',
           'ProductCategory',
           'ProductPrice',
           'RetailerName',
           'RetailerZipCode',
           'RetailerCity',
           'RetailerState',
           'RetailerOnSale',
           'ManufacturerName',
           'ManufacturerRebate',
           'ReviewUserId',
           'ReviewRating',
           'ReviewDate',
           'ReviewText'],
          ['Samsung TV 60 LED',
           'TV',
           '1200',
           'Bestbuy',
           '60585',
           'Naperville',
           'IL',
           'No',
           'Samsung',
           'No',
           'abc234',
           '1',
           '3/14/2013',
           'Bluetooth did not work for my sound system'],
          ['Surface 3',
           'Tablet',
           '399',
           'Hhgregg',
           '90012',
           'Los Angeles',
           'CA',
           'Yes',
           'Microsoft',
           'No',
           'chj787',
           '5',
           '7/1/2014',
           'Much better screen than the samsung galaxy tablet']]
```

```
In [21]: # 10. Write our Reviews Table (Remember it is LOL) into a CSV File

csvOutFile=open('CustomerReviews.csv','wt') # this will be a text file

csvOutFile.write("\n".join(",".join(row) for row in reviews_table))

csvOutFile.close()
```

```
In [22]: #11 Read the CSV file adn load it into DataFrame

custreview100=pd.read_csv('CustomerReviews.csv')
```

```
In [23]: type(custreview100)
```

```
Out[23]: pandas.core.frame.DataFrame
```

```
In [24]: custreview100.dtypes
```

```
Out[24]: ProductModelName      object
ProductCategory               object
ProductPrice                   int64
RetailerName                   object
RetailerZipCode                int64
RetailerCity                   object
RetailerState                  object
RetailerOnSale                 object
ManufacturerName               object
ManufacturerRebate             object
ReviewUserId                   object
ReviewRating                   int64
ReviewDate                     object
ReviewText                     object
dtype: object
```

In [25]: `custreview100.head()`

Out[25]:

	ProductModelName	ProductCategory	ProductPrice	RetailerName	RetailerZipCode
0	Samsung TV 60 LED	TV	1200	Bestbuy	60585
1	Surface 3	Tablet	399	Hhgregg	90012
2	Sony TV 42 LED	TV	800	Frys	60616
3	LG 65	TV	2250	Bestbuy	2110
4	Dell XP 15	Laptop	1349	Hhgregg	60616

+++++

+++++

+++++ Now lets execute different queries for our analysis +++++

+++++

+++++

Query 1: Print the list of all retailers along with the products and the rating

```
In [26]: custreview100[['ProductModelName', 'RetailerName', 'ReviewRating']].head()
```

```
Out[26]:
```

	ProductModelName	RetailerName	ReviewRating
0	Samsung TV 60 LED	Bestbuy	1
1	Surface 3	Hhgregg	5
2	Sony TV 42 LED	Frys	4
3	LG 65	Bestbuy	5
4	Dell XP 15	Hhgregg	3

Query 2: Print a list of reviews where rating greater than 3 - TWO WAYS TO DO IT - SEE BELOW

```
In [27]: df4 = custreview100[custreview100['ReviewRating'] > 3]
```

```
In [28]: df4[['ProductModelName', 'RetailerName', 'ReviewRating']].head()
```

```
Out[28]:
```

	ProductModelName	RetailerName	ReviewRating
1	Surface 3	Hhgregg	5
2	Sony TV 42 LED	Frys	4
3	LG 65	Bestbuy	5
6	Samsung TV 65 Curved	Bestbuy	5
7	HP Pavilion 15.6	Hhgregg	5

```
In [29]: custreview100[custreview100['ReviewRating'] > 3][['ProductModelName', 'Re
```

```
Out[29]:
```

	ProductModelName	RetailerName	ReviewRating
1	Surface 3	Hhgregg	5
2	Sony TV 42 LED	Frys	4
3	LG 65	Bestbuy	5
6	Samsung TV 65 Curved	Bestbuy	5
7	HP Pavilion 15.6	Hhgregg	5

Query 3: Get a list of products that got review rating 5 and price more than thousand

In [30]: `custreview100[(custreview100['ReviewRating'] == 5) & (custreview100['Prod`

Out[30]:

	ProductModelName	ProductCategory	ProductPrice	RetailerName	RetailerZipCode
3	LG 65	TV	2250	Bestbuy	2110
6	Samsung TV 65 Curved	TV	2199	Bestbuy	94102
11	Samsung TV 65 Curved	TV	2199	Bestbuy	2108
29	Dell XP 13	Laptop	1150	Walmart	94158
31	Samsung TV 65 Curved	TV	1899	Hhgregg	30134

```
In [31]: custreview100.head()
```

Out[31]:

	ProductModelName	ProductCategory	ProductPrice	RetailerName	RetailerZipCode	F
0	Samsung TV 60 LED	TV	1200	Bestbuy	60585	N
1	Surface 3	Tablet	399	Hhgregg	90012	L
2	Sony TV 42 LED	TV	800	Frys	60616	C
3	LG 65	TV	2250	Bestbuy	2110	E
4	Dell XP 15	Laptop	1349	Hhgregg	60616	C

Query 4: How many products reviewed for every retailer?

```
In [32]: custreview100.RetailerName.value_counts()
```

```
Out[32]: Walmart      26
Target      25
Hhgregg     20
Bestbuy     19
Frys        10
Name: RetailerName, dtype: int64
```

Query 5: How many reviews for every product?

```
In [33]: custreview100.ProductModelName.value_counts()
```

```
Out[33]: Samsung TV 65 Curved      15
          HTC One                  13
          Dell XP 13                12
          Surface 3                 10
          Lenevo Y50                 7
          Galaxy S4                  6
          Samsung Galaxy Tab 4       6
          iPhone 6                   6
          iPad Air                   6
          HP Pavilion 15.6           4
          LG 65                      3
          Samsung TV 60 LED          3
          Sony TV 42 LED             3
          Dell XP 15                 3
          HP Pavilion 15.6           3
          Name: ProductModelName, dtype: int64
```

Query 6: How many reviews for every product category?

```
In [34]: custreview100.ProductCategory .value_counts()
```

```
Out[34]: Laptop      29
          Smart Phone 25
          TV          24
          Tablet      21
          Walmart      1
          Name: ProductCategory, dtype: int64
```

Query 7: Get the list of reviews for shoppers in Chicago

```
In [35]: ChicagoShoppers=custreview100[custreview100.RetailerCity=='Chicago']
```


In [36]: `ChicagoShoppers.head()`

Out[36]:

	ProductModelName	ProductCategory	ProductPrice	RetailerName	RetailerZipCode
2	Sony TV 42 LED	TV	800	Frys	60616
4	Dell XP 15	Laptop	1349	Hhgregg	60616
5	HP Pavilion 15.6	Laptop	1399	Hhgregg	60603
9	Dell XP 13	Laptop	1349	Walmart	60616
21	Dell XP 13	Laptop	1200	Walmart	60616

Query 8: Set the index of the ChicagoShoppers to be RetailerName

In [37]: `ChicagoShoppers=ChicagoShoppers.set_index('RetailerName')`

```
In [38]: ChicagoShoppers.head()
```

```
Out[38]:
```

	ProductModelName	ProductCategory	ProductPrice	RetailerZipCode	Re
RetailerName					
Frys	Sony TV 42 LED	TV	800	60616	Ch
Hhgregg	Dell XP 15	Laptop	1349	60616	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1399	60603	Ch
Walmart	Dell XP 13	Laptop	1349	60616	Ch
Walmart	Dell XP 13	Laptop	1200	60616	Ch

Query 9: Sort the dataframe based on the row-label index -- that is the retailer name

```
In [39]: ChicagoShoppers=ChicagoShoppers.sort_index()
```

```
In [40]: ChicagoShoppers
```

```
Out[40]:
```

	ProductModelName	ProductCategory	ProductPrice	RetailerZipCode	Re
RetailerName					
Bestbuy	Dell XP 13	Laptop	1349	60616	Ch
Bestbuy	Dell XP 13	Laptop	1349	60616	Ch
Frys	Sony TV 42 LED	TV	800	60616	Ch
Frys	Sony TV 42 LED	TV	800	60616	Ch

Hhgregg	Dell XP 15	Laptop	1349	60616	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1399	60603	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1599	60603	Ch
Hhgregg	Dell XP 15	Laptop	1499	60616	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1899	60603	Ch
Target	Sony TV 42 LED	TV	799	60616	Ch
Target	Dell XP 15	Laptop	1899	60616	Ch
Walmart	Dell XP 13	Laptop	1349	60616	Ch
Walmart	Dell XP 13	Laptop	1200	60616	Ch
Walmart	Dell XP 13	Laptop	999	60616	Ch
Walmart	Dell XP 13	Laptop	1200	60616	Ch

Query 10: Sorting based on column values using order field. It returns a series

```
In [41]: ChicagoShoppers.ProductPrice.order(ascending=False)
```

```
/Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-
packages/ipykernel/__main__.py:1: FutureWarning: order is deprecated,
use sort_values(...)
    if __name__ == '__main__':
```

```
Out[41]: RetailerName
Target      1899
Hhgregg     1899
Hhgregg     1599
Hhgregg     1499
Hhgregg     1399
Walmart    1349
Hhgregg     1349
Bestbuy     1349
Bestbuy     1349
Walmart    1200
Walmart    1200
Walmart     999
Frys        800
Frys        800
Target      799
Name: ProductPrice, dtype: int64
```

```
In [42]: ChicagoShoppers
```

```
Out[42]:
```

	ProductModelName	ProductCategory	ProductPrice	RetailerZipCode	Re
RetailerName					
Bestbuy	Dell XP 13	Laptop	1349	60616	Ch
Bestbuy	Dell XP 13	Laptop	1349	60616	Ch
Frys	Sony TV 42 LED	TV	800	60616	Ch
Frys	Sony TV 42 LED	TV	800	60616	Ch
Hhgregg	Dell XP 15	Laptop	1349	60616	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1399	60603	Ch

Hhgregg	HP Pavilion 15.6	Laptop	1599	60603	Ch
Hhgregg	Dell XP 15	Laptop	1499	60616	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1899	60603	Ch
Target	Sony TV 42 LED	TV	799	60616	Ch
Target	Dell XP 15	Laptop	1899	60616	Ch
Walmart	Dell XP 13	Laptop	1349	60616	Ch
Walmart	Dell XP 13	Laptop	1200	60616	Ch
Walmart	Dell XP 13	Laptop	999	60616	Ch
Walmart	Dell XP 13	Laptop	1200	60616	Ch

Query 11: Sort Column values using sort_index method to return a Dataframe rather than Series

In [43]: ChicagoShoppersSortedPrice=ChicagoShoppers.sort_index(ascending=False, by
 /Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-
 packages/ipykernel/__main__.py:1: FutureWarning: by argument to sort_i
 ndex is deprecated, pls use .sort_values(by=...)
 if __name__ == '__main__':

In [44]: ChicagoShoppersSortedPrice

Out[44]:

	ProductModelName	ProductCategory	ProductPrice	RetailerZipCode	Re
RetailerName					
Hhgregg	HP Pavilion 15.6	Laptop	1899	60603	Ch
Target	Dell XP 15	Laptop	1899	60616	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1599	60603	Ch
Hhgregg	Dell XP 15	Laptop	1499	60616	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1399	60603	Ch
Bestbuy	Dell XP 13	Laptop	1349	60616	Ch
Bestbuy	Dell XP 13	Laptop	1349	60616	Ch
Hhgregg	Dell XP 15	Laptop	1349	60616	Ch
Walmart	Dell XP 13	Laptop	1349	60616	Ch
Walmart	Dell XP 13	Laptop	1200	60616	Ch
Walmart	Dell XP 13	Laptop	1200	60616	Ch

Walmart	Dell XP 13	Laptop	999	60616	Ch
Frys	Sony TV 42 LED	TV	800	60616	Ch
Frys	Sony TV 42 LED	TV	800	60616	Ch
Target	Sony TV 42 LED	TV	799	60616	Ch

Query 12: Sort based on two column values. We are sorting based on product prices then based on manufacturer name

In [45]: ChicagoShoppersSortedPriceAndManufacturer=ChicagoShoppers.sort_index(ascending=True, by=[

```

/Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-packages/ipykernel/__main__.py:2: FutureWarning: by argument to sort_index is deprecated, pls use .sort_values(by=...)
from ipykernel import kernelapp as app

```

In [46]: ChicagoShoppersSortedPriceAndManufacturer

Out[46]:

	ProductModelName	ProductCategory	ProductPrice	RetailerZipCode	Re
RetailerName					
Target	Dell XP 15	Laptop	1899	60616	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1899	60603	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1599	60603	Ch

Hhgregg	Dell XP 15	Laptop	1499	60616	Ch
Hhgregg	HP Pavilion 15.6	Laptop	1399	60603	Ch
Bestbuy	Dell XP 13	Laptop	1349	60616	Ch
Bestbuy	Dell XP 13	Laptop	1349	60616	Ch
Hhgregg	Dell XP 15	Laptop	1349	60616	Ch
Walmart	Dell XP 13	Laptop	1349	60616	Ch
Walmart	Dell XP 13	Laptop	1200	60616	Ch
Walmart	Dell XP 13	Laptop	1200	60616	Ch
Walmart	Dell XP 13	Laptop	999	60616	Ch
Frys	Sony TV 42 LED	TV	800	60616	Ch
Frys	Sony TV 42 LED	TV	800	60616	Ch
Target	Sony TV 42 LED	TV	799	60616	Ch


```
In [47]: Top10LikedItemsInChicago=ChicagoShoppers.set_index('ProductModelName')
Top10LikedItemsInChicago.head()
```

```
Out[47]:
```

	ProductCategory	ProductPrice	RetailerZipCode	RetailerCity	Reta
ProductModelName					
Dell XP 13	Laptop	1349	60616	Chicago	IL
Dell XP 13	Laptop	1349	60616	Chicago	IL
Sony TV 42 LED	TV	800	60616	Chicago	IL
Sony TV 42 LED	TV	800	60616	Chicago	IL
Dell XP 15	Laptop	1349	60616	Chicago	IL

Query 13: Get the top 10 list of items liked in Chicago

```
In [48]: Top10LikedItemsInChicago=Top10LikedItemsInChicago.ReviewRating.order(asce
/Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-
packages/ipykernel/__main__.py:1: FutureWarning: order is deprecated,
use sort_values(...)
if __name__ == '__main__':
```

```
In [49]: Top10LikedItemsInChicago
```

```
Out[49]: ProductModelName
Dell XP 15          5
HP Pavilion 15.6    5
Sony TV 42 LED     4
Dell XP 13          4
Dell XP 13          3
Sony TV 42 LED     3
Dell XP 15          3
Dell XP 13          3
HP Pavilion 15.6    2
HP Pavilion 15.6    2
Name: ReviewRating, dtype: int64
```

Query 14: groupby returns a group of dataframe objects indexed based on a key you specify more or less dictionary-like

```
In [50]: ItemsReviewedByCity=custreview100.groupby('RetailerCity')
```

```
In [51]: type(ItemsReviewedByCity)
```

```
Out[51]: pandas.core.groupby.DataFrameGroupBy
```

```
In [52]: for key, group in ItemsReviewedByCity:
          print key
          print group
```

Atlanta

	ProductModelName	ProductCategory	ProductPrice	RetailerName	\
10	iPhone 6	Smart Phone	299	Frys	
18	HTC One	Smart Phone	299	Target	
19	Samsung TV 65 Curved	TV	1899	Target	
22	iPhone 6	Smart Phone	399	Target	
30	HTC One	Smart Phone	199	Target	
31	Samsung TV 65 Curved	TV	1899	Hhgregg	
42	iPhone 6	Smart Phone	399	Frys	
50	HTC One	Smart Phone	199	Target	
51	Samsung TV 65 Curved	TV	2099	Walmart	
54	iPhone 6	Smart Phone	399	Target	
62	HTC One	Smart Phone	199	Bestbuy	
63	Samsung TV 65 Curved	TV	1899	Walmart	
74	iPhone 6	Smart Phone	299	Hhgregg	
82	HTC One	Smart Phone	299	Target	
83	Samsung TV 65 Curved	TV	1899	Target	
86	iPhone 6	Smart Phone	399	Frys	
94	HTC One	Smart Phone	199	Frys	
95	Samsung TV 65 Curved	TV	1899	Walmart	

Query 15: Find highest price product reviewed/sold in every city

```
In [53]: #generator comprehension for retrieving highest product sold by city
gl = (group.sort_index(by='ProductPrice', ascending=False)[:1] for rtc, g
```

```
In [54]: gl
```

```
Out[54]: <generator object <genexpr> at 0x10ed51690>
```

```
In [55]: topProductsByCity = pd.DataFrame()
```

```
In [56]: for line in gl:
          topProductsByCity = topProductsByCity.append(line)
```

```
/Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-
packages/ipykernel/__main__.py:2: FutureWarning: by argument to sort_i
ndex is deprecated, pls use .sort_values(by=...)
from ipykernel import kernelapp as app
```

In [57]: topProductsByCity

Out[57]:

	ProductModelName	ProductCategory	ProductPrice	RetailerName	RetailerZipCode
51	Samsung TV 65 Curved	TV	2099	Walmart	30134
3	LG 65	TV	2250	Bestbuy	2110
36	Dell XP 15	Laptop	1899	Target	60616
8	Galaxy S4	Smart Phone	499	Bestbuy	44114
1	Surface 3	Tablet	399	Hhgregg	90012
14	Lenevo Y50	Laptop	1349	Walmart	33129
0	Samsung TV 60 LED	TV	1200	Bestbuy	60585
6	Samsung TV 65 Curved	TV	2199	Bestbuy	94102

Query 16: Plot median product prices per

city

```
In [58]: ItemsReviewedByCity['ProductPrice'].median().plot()
```

```
Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x112688a10>
```

Query 17: Get the top 5 list of liked products for every city

```
In [59]: #generator for retrieving top 5 products liked by city
gTopRating5 = (group.sort_index(by='ReviewRating', ascending=False)[:5] for group in ItemsReviewedByCity.groupby('City'))
```

```
In [60]: topLikedProductsByCity = pd.DataFrame()
```

```
In [61]: for line in gTopRating5 :
        topLikedProductsByCity = topLikedProductsByCity.append(line)
```

```
/Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-packages/ipykernel/__main__.py:2: FutureWarning: by argument to sort_index is deprecated, pls use .sort_values(by=...)
  from ipykernel import kernelapp as app
```

```
In [62]: topLikedProductsByCity
```

```
Out[62]:
```

	ProductModelName	ProductCategory	ProductPrice	RetailerName	RetailerZipCode	ReviewRating
10	iPhone 6	Smart Phone	299	Frys	30303	4.5
94	HTC One	Smart Phone	199	Frys	30303	4.5
22	iPhone 6	Smart Phone	399	Target	30303	4.5
30	HTC One	Smart Phone	199	Target	30303	4.5

Query 18: Get top 5 list of most liked and expensive products sorted by retailer name for every city

```
In [63]: #generator for retrieving top 5 products most expensive and liked by city
gTop5LikedExpensive = (group.sort_index(ascending=[False, False, True],
                                     by=['ReviewRating', 'ProductPrice']
                                     for rtc, group in ItemsReviewedByCity )
```

```
In [64]: top5LikedExpensiveProductsByCity = pd.DataFrame()
```

```
In [65]: for line in gTop5LikedExpensive :
          top5LikedExpensiveProductsByCity = top5LikedExpensiveProductsByCity.append(
          /Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-packages/ipykernel/__main__.py:4: FutureWarning: by argument to sort_index is deprecated, pls use .sort_values(by=...)
```

```
In [66]: top5LikedExpensiveProductsByCity[['RetailerCity', 'RetailerName', 'ProductModelName', 'ProductPrice', 'ReviewRating']]
```

Out[66]:

	RetailerCity	RetailerName	ProductModelName	ProductPrice	ReviewRating
51	Atlanta	Walmart	Samsung TV 65 Curved	2099	5
31	Atlanta	Hhgregg	Samsung TV 65 Curved	1899	5
86	Atlanta	Frys	iPhone 6	399	5
22	Atlanta	Target	iPhone 6	399	5
10	Atlanta	Frys	iPhone 6	299	5
3	Boston	Bestbuy	LG 65	2250	5
35	Boston	Bestbuy	LG 65	2250	5
11	Boston	Bestbuy	Samsung TV 65 Curved	2199	5
43	Boston	Frys	Samsung TV 65 Curved	2099	5
55	Boston	Target	Samsung TV 65 Curved	1900	5
36	Chicago	Target	Dell XP 15	1899	5
37	Chicago	Hhgregg	HP Pavilion 15.6	1599	5
41	Chicago	Bestbuy	Dell XP 13	1349	4
2	Chicago	Frys	Sony TV 42 LED	800	4
73	Chicago	Bestbuy	Dell XP 13	1349	3

84	Cleveland	Bestbuy	Galaxy S4	499	5
52	Cleveland	Walmart	Galaxy S4	399	5
40	Cleveland	Frys	Galaxy S4	499	4
8	Cleveland	Bestbuy	Galaxy S4	499	3
20	Cleveland	Bestbuy	Galaxy S4	499	3
1	Los Angeles	Hhgregg	Surface 3	399	5
65	Los Angeles	Hhgregg	Surface 3	399	5
33	Los Angeles	Target	Surface 3	399	5
12	Los Angeles	Walmart	HTC One	199	5
88	Los Angeles	Walmart	HTC One	199	5
58	Miami	Hhgregg	Lenevo Y50	1149	5
39	Miami	Bestbuy	HP Pavilion 15.6	599	5
7	Miami	Hhgregg	HP Pavilion 15.6	599	5
14	Miami	Walmart	Lenevo Y50	1349	3
78	Miami	Walmart	Lenevo Y50	1349	3
32	Naperville	Walmart	Samsung TV 60 LED	1100	5
64	Naperville	Bestbuy	Samsung TV 60 LED	1200	4
0	Naperville	Bestbuy	Samsung TV 60 LED	1200	1
6	San Francisco	Bestbuy	Samsung TV 65 Curved	2199	5
29	San Francisco	Walmart	Dell XP 13	1150	5
81	San Francisco	Hhgregg	Dell XP 13	1099	5
49	San Francisco	Frys	Dell XP 13	999	5
17	San Francisco	Hhgregg	Dell XP 13	1099	4

+++++

+++++

Write Python code for the following

requirements (ONE-CELL-FOR-EVERY-REQUIREMENT):

+++++

+++++

Requirement #1:

Get the top 5 list of most Disliked products sorted by retailer name for every city


```
In [67]: ItemsReviewedByRetailer=custreview100.groupby('RetailerName')

gBottomRating5 = (group.sort_index(by=['ReviewRating','RetailerName'], as

TopDislikedProductsByRetailer = pd.DataFrame()

for line in gBottomRating5 :
    TopDislikedProductsByRetailer = TopDislikedProductsByRetailer.append(

TopDislikedProductsByRetailer
```

```
/Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-
packages/ipykernel/__main__.py:3: FutureWarning: by argument to sort_i
ndex is deprecated, pls use .sort_values(by=...)
app.launch_new_instance()
```

```
Out[67]:
```

	ProductModelName	ProductCategory	ProductPrice	RetailerName	RetailerZipCode	
0	Samsung TV 60 LED	TV	1200	Bestbuy	60585	↑
62	HTC One	Smart Phone	199	Bestbuy	30303	↓
72	Galaxy S4	Smart Phone	499	Bestbuy	44114	↓

Requirement #2:

Get the top 5 list of cheapest products sorted by retailer name for every city

```
In [68]: ItemsReviewedByRetailer=custreview100.groupby('RetailerName')

gCheapProd5 = (group.sort_index(by=['ProductPrice','RetailerName'], ascen

CheapestProductsByRetailer = pd.DataFrame()

for line in gCheapProd5 :
    CheapestProductsByRetailer = CheapestProductsByRetailer.append(line)

CheapestProductsByRetailer
```

```
/Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-
packages/ipykernel/__main__.py:3: FutureWarning: by argument to sort_i
ndex is deprecated, pls use .sort_values(by=...)
app.launch_new_instance()
```

```
Out[68]:
```

	ProductModelName	ProductCategory	ProductPrice	RetailerName	RetailerZipCode	
44	HTC One	Smart Phone	99	Bestbuy	90033	L
62	HTC One	Smart Phone	199	Bestbuy	30303	/
92	Samsung Galaxy Tab 4	Tablet	399	Bestbuy	2109	E

Requirement #3:

Get the total number of products reviewed and got Rating 5 in Every City

```
In [69]: Total5RatedProducts=custreview100[custreview100.ReviewRating==5]
Total5RatedProducts.RetailerCity.value_counts()
```

```
Out[69]: Boston          13
Atlanta          7
Los Angeles      7
San Francisco    4
Miami            3
Chicago          2
Cleveland        2
Naperville       1
Name: RetailerCity, dtype: int64
```

Requirement #4:

Get the top 2 list of zip-codes where highest number of products got review rating 5

```
In [70]: ZipCodeWithHighestRated = Total5RatedProducts['RetailerZipCode'].value_co
ZipCodeWithHighestRated.head(2)
```

```
Out[70]: 2109          8
30303         5
Name: RetailerZipCode, dtype: int64
```

Requirement #5:

which City got the most liked (rating >2) products sorted by product names

```
In [71]: MostLikedProducts=custreview100[custreview100.ReviewRating>2]

MostLikedProductsByProductName=MostLikedProducts.sort_index(ascending=True
MostLikedProductsByProductName

items_counts = MostLikedProductsByProductName['RetailerCity'].value_count
items_counts.head(1)
```

```
/Users/Zeeshan/Library/Enthought/Canopy_64bit/User/lib/python2.7/site-
packages/ipykernel/__main__.py:3: FutureWarning: by argument to sort_i
ndex is deprecated, pls use .sort_values(by=...)
  app.launch_new_instance()
```

```
Out[71]: Boston      17
         Name: RetailerCity, dtype: int64
```

```
In [ ]:
```