1 CS 1656 – Introduction to Data Science

1.1 Instructor: Alexandros Labrinidis / Teaching Assistant: Tahereh Arabghalizi

1.1.1 Additional credits: E. Karageorgos, Zuha Agha, Anatoli Shein, Phuong Pham

1.2 ## Lab 4: Data Analysis with Pandas

So far we have encountered basic data manipulation with pandas Dataframes including row and column selection, boolean indexing, working with missing values, groupby and aggregate functions such as mean().But there are many other powerful data manipulation and analysis techniques available in pandas. In this lab, we will learn some more advanced ways for data anylsis in Python using Dataframes.

Begin by importing pandas package.

```
In [1]: import pandas as pd
```

Next load the dataset that we will be playing arround with.

```
In [2]: df = pd.read_csv('http://data.cs1656.org/coffee-chain.csv')
        df.head()
```

Out[2]:

	Area Code	Market	Market Size	Product	Product Line	Product Type	State	Type	Inventory	Budget COGS	Budget Margin	Budget Profit	Budget Sales	COGS	Margin	Marketing	Profit	Sales	Total Expenses
0	985	South	Small Market	Colombian	Beans	Coffee	Louisiana	Regular	845	50	90	70	140	49	71	13	68	128	25
1	985	South	Small Market	Chamomile	Leaves	Herbal Tea	Louisiana	Decaf	540	80	110	70	190	94	120	31	114	228	43
2	985	South	Small Market	Chamomile	Leaves	Herbal Tea	Louisiana	Decaf	552	90	120	80	210	101	130	33	126	246	45
3	985	South	Small Market	Decaf Irish Cream	Beans	Coffee	Louisiana	Decaf	851	70	90	80	160	48	70	13	67	126	25
4	985	South	Small Market	Lemon	Leaves	Herbal Tea	Louisiana	Decaf	599	60	80	30	140	67	83	25	37	160	58

Let's get the subset of the dataframe we need.

```
In [3]: df_small = df[['Area Code','Market', 'Market Size', 'Product', 'Product Line', 'Product
        df_small.head()
```

```
Out[3]:    Area Code Market   Market Size            Product Product Line  \
        0        985  South  Small Market          Colombian        Beans
        1        985  South  Small Market          Chamomile       Leaves
        2        985  South  Small Market          Chamomile       Leaves
        3        985  South  Small Market  Decaf Irish Cream        Beans
        4        985  South  Small Market              Lemon       Leaves

          Product Type      State     Type  Profit  Total Expenses
        0       Coffee  Louisiana  Regular      68              25
        1   Herbal Tea  Louisiana    Decaf     114              43
        2   Herbal Tea  Louisiana    Decaf     126              45
        3       Coffee  Louisiana    Decaf      67              25
        4   Herbal Tea  Louisiana    Decaf      37              58
```

1.3 Slicing & Indexing

What we saw above was slicing. Slicing uses the [] operator selects a set of rows and/or columns from a DataFrame.

Slicing rows

To slice out a set of rows, you use the following syntax: data[start:stop]. When slicing in pandas the start bound is included in the output.

```
In [4]: df_small[0:3]
```

```
Out[4]:    Area Code Market   Market Size    Product Product Line Product Type  \
        0        985  South  Small Market  Colombian        Beans       Coffee
        1        985  South  Small Market  Chamomile       Leaves   Herbal Tea
        2        985  South  Small Market  Chamomile       Leaves   Herbal Tea

              State     Type  Profit  Total Expenses
        0  Louisiana  Regular      68              25
        1  Louisiana    Decaf     114              43
        2  Louisiana    Decaf     126              45
```

Slicing vs Copying

We might have thought that we were creating a fresh copy of df_small when we did slicing. However the statement y = x doesn't create a copy of our DataFrame. It creates a new variable y that refers to the same object x refers to. This means that there is only one object (the DataFrame), and both x and y refer to it. To create a fresh copy of the DataFrame you can use the syntax y=x.copy(). We will see the effect of slicing but not copying in later steps.

** Indexing **

We can select specific ranges of our data in both the row and column directions using either label or integer-based indexing.

- loc: indexing via labels or integers or mixed
- iloc: indexing via integers only

To select a subset of rows AND columns from our DataFrame, we can use the iloc method. For example,

```
In [5]: df_small.loc[0:3, 'Market': 'Product']
```

```
Out[5]:   Market   Market Size            Product
        0  South  Small Market          Colombian
        1  South  Small Market          Chamomile
        2  South  Small Market          Chamomile
        3  South  Small Market  Decaf Irish Cream
```

```
In [6]: df_small.iloc[0:4, 1:4]
```

```
Out[6]:   Market   Market Size            Product
        0  South  Small Market          Colombian
        1  South  Small Market          Chamomile
        2  South  Small Market          Chamomile
        3  South  Small Market  Decaf Irish Cream
```

Notice that indexing in loc is inclusive whereas indexing in iloc is exlusive of the end index

1.4 Statistical Technqiues

1.4.1 Cross-tabulation

Cross tabultaion computes a frequency table of two or more factors. Let's start by making a cross-tab with two variables first.

```
In [7]: df_crosstab = pd.crosstab(df_small["Market"],df_small["Market Size"],margins=True)
        df_crosstab
```

```
Out[7]: Market Size  Major Market  Small Market   All
        Market
        Central               696           648  1344
        East                  552           336   888
        South                 168           504   672
        West                  288          1056  1344
        All                  1704          2544  4248
```

Let'c check the type of the cross-tab

```
In [8]: type(df_crosstab)
```

```
Out[8]: pandas.core.frame.DataFrame
```

Now let's check the value counts of one of our cross-tab's dimensions and see if the totals match?

```
In [9]: pd.value_counts(df_small['Market Size'])
```

```
Out[9]: Small Market    2544
        Major Market    1704
        Name: Market Size, dtype: int64
```

Now let's make a cross-tab with three variables.

```
In [10]: pd.crosstab(df["Product Type"], [df["Market"],df["Market Size"]],margins=True)
```

```
Out[10]: Market            Central                      East                      South  \
         Market Size  Major Market Small Market Major Market Small Market Major Market
         Product Type
         Coffee                192          192           96           72           48
         Espresso              144          144          144           96           72
         Herbal Tea            192          144          144           72           48
         Tea                   168          168          168           96            0
         All                   696          648          552          336          168

         Market                            West                All
         Market Size  Small Market Major Market Small Market
         Product Type
         Coffee                144           72          240  1056
         Espresso              216           72          288  1176
         Herbal Tea            144           72          240  1056
         Tea                     0           72          288   960
         All                   504          288         1056  4248
```

1.4.2 Binning Data

We can bin our data into categorirs by specifying bin widths. Let's define equal width bins as shown below. The bins array specifies 4 bins from -800 to -400, -400 to 0, 0 to 400, 400 to 800. We will also specify a group names to assign as labels to each of our bins later.

```
In [11]: bins = [-800,-400, 0, 400, 800]
         group_names = ['Low', 'Okay', 'Good', 'Great']
```

Now lets bin the data into the categories and add it as a column to the dataframe

```
In [12]: df_small['Categories'] = pd.cut(df_small['Profit'], bins=bins, labels=group_names)
         df_small.head(20)
```

```
Out[12]:     Area Code Market   Market Size            Product Product Line  \
         0         985  South  Small Market          Colombian        Beans
         1         985  South  Small Market          Chamomile       Leaves
         2         985  South  Small Market          Chamomile       Leaves
         3         985  South  Small Market  Decaf Irish Cream        Beans
         4         985  South  Small Market              Lemon       Leaves
         5         985  South  Small Market  Decaf Irish Cream        Beans
```

```
6          985  South  Small Market              Lemon       Leaves
7          985  South  Small Market          Chamomile       Leaves
8          985  South  Small Market        Caffe Mocha        Beans
9          985  South  Small Market        Caffe Latte        Beans
10         985  South  Small Market        Caffe Latte        Beans
11         985  South  Small Market  Decaf Irish Cream        Beans
12         985  South  Small Market     Decaf Espresso        Beans
13         985  South  Small Market              Lemon       Leaves
14         985  South  Small Market     Decaf Espresso        Beans
15         985  South  Small Market              Lemon       Leaves
16         985  South  Small Market        Caffe Mocha        Beans
17         985  South  Small Market        Caffe Latte        Beans
18         985  South  Small Market        Caffe Mocha        Beans
19         985  South  Small Market     Decaf Espresso        Beans
```

	Product Type	State	Type	Profit	Total Expenses	Categories
0	Coffee	Louisiana	Regular	68	25	Good
1	Herbal Tea	Louisiana	Decaf	114	43	Good
2	Herbal Tea	Louisiana	Decaf	126	45	Good
3	Coffee	Louisiana	Decaf	67	25	Good
4	Herbal Tea	Louisiana	Decaf	37	58	Good
5	Coffee	Louisiana	Decaf	87	26	Good
6	Herbal Tea	Louisiana	Decaf	43	58	Good
7	Herbal Tea	Louisiana	Decaf	48	26	Good
8	Espresso	Louisiana	Regular	61	35	Good
9	Espresso	Louisiana	Regular	4	81	Good
10	Espresso	Louisiana	Regular	1	86	Good
11	Coffee	Louisiana	Decaf	70	25	Good
12	Espresso	Louisiana	Decaf	56	39	Good
13	Herbal Tea	Louisiana	Decaf	62	65	Good
14	Espresso	Louisiana	Decaf	61	40	Good
15	Herbal Tea	Louisiana	Decaf	26	59	Good
16	Espresso	Louisiana	Regular	31	35	Good
17	Espresso	Louisiana	Regular	-3	79	Okay
18	Espresso	Louisiana	Regular	58	41	Good
19	Espresso	Louisiana	Decaf	31	36	Good

To find out the value counts for each bin of category, we can use value_counts like we did earlier.

```
In [13]: pd.value_counts(df_small['Categories'])
```

```
Out[13]: Good     3648
         Okay      544
         Great      40
         Low        16
         Name: Categories, dtype: int64
```

1.4.3 Quantiles

Pandas allows an easy way of computing percentiles or quartiles. Let's first specify the quantiles we want to calculate,

```
In [14]: quants = [0.0, 0.05, 0.25, 0.5, 0.75, 0.95, 1.0]
```

To compute the quantiles of Profit and Total Expenses,

```
In [15]: q = df_small[['Profit','Total Expenses']].quantile(quants)
         q
```

```
Out[15]:       Profit  Total Expenses
         0.00  -638.0            10.0
         0.05   -13.0            17.0
         0.25    17.0            33.0
         0.50    40.0            46.0
         0.75    92.0            65.0
         0.95   232.0           125.0
         1.00   778.0           190.0
```

1.4.4 Groupby & Apply

Groupby allows grouping or clustering the dataframe by a particular categorical attribute. Apply can be used to apply a function to a group or the entire dataframe. Let's first define the function that we want to apply,

```
In [16]: def get_stats(group):
             return {'min': group.min(), 'max': group.max(), 'count': group.count(), 'mean': gro
```

This can be applied to a Dataframe or a grouping of the dataframe as shown below

```
In [17]: df_group = df_small['Profit'].groupby(df_small['Categories']).apply(get_stats)
         df_group
```

```
Out[17]: Categories
         Low         count        16.000000
                     max        -404.000000
                     mean       -510.562500
                     min        -638.000000
                     sum       -8169.000000
         Okay        count       544.000000
                     max           0.000000
                     mean        -45.630515
                     min        -392.000000
                     sum      -24823.000000
         Good        count      3648.000000
                     max         397.000000
                     mean         74.514529
                     min           1.000000
```

```
                sum      271829.000000
Great           count        40.000000
                max         778.000000
                mean        517.650000
                min         402.000000
                sum       20706.000000
Name: Profit, dtype: float64
```

The width format of the output above can be fixed by using the unstack() function as shown below.

```
In [18]: df_group.unstack()
```

Out[18]:

	count	max	mean	min	sum
Categories					
Low	16.0	-404.0	-510.562500	-638.0	-8169.0
Okay	544.0	0.0	-45.630515	-392.0	-24823.0
Good	3648.0	397.0	74.514529	1.0	271829.0
Great	40.0	778.0	517.650000	402.0	20706.0

1.4.5 Sorting

Pandas allows nested sorting over mutliple columns of the Dataframe easily as shown below.

```
In [19]: data_sorted = df_small.sort_values(['Total Expenses', 'Profit'], ascending=False)
         data_sorted[['Total Expenses','Profit']].head(20)
```

Out[19]:

	Total Expenses	Profit
959	190	49
2334	189	50
2352	189	-284
3432	181	-266
966	180	45
2224	180	45
632	178	370
1429	178	370
631	178	368
1605	178	368
753	177	357
1622	177	357
1454	177	68
285	176	69
4086	176	-392
3420	168	-367
1461	167	62
3278	167	62
1269	166	511
1596	166	511

1.5 Tasks

For your tasks, use the data file http://data.cs1656.org/bank-data.csv.

Task 1 Compute the mean income of males versus females.

Task 2 Create a cross-tab of save_acct and mortgage.

Task 3 Convert the frequencies in cross-tab to percentages. (Hint: use apply and indexing)