**ADA Assignment#2:**

**Question.1**

**Answer:**

Although indexes are "generally immutable," their name attribute can be set and modified. These properties can be directly set using the rename and set names functions, and by default, they return a copy.

Axis lebels were also stored in indexes.

**Example:**

import pandas as pd

ind = pd.Index([1, 2, 3])

ind.rename("apple")

**output**: Int64Index[1, 2, 3], dtype='int64', name='apple')

But when we print ‘ind’ index again it returns original array, instead of renamed.

**input:** ind

**output:** Int64Index([1, 2, 3], dtype='int64')

**Question.2**

**Answer:**

**.corr:** From code in textbookthe piece of code **“returns[‘MSFT’].corr(returns[‘IBM’])”**

The overlapping, non-NA, aligned-by-index values in two Series are computed to determine their correlation using the corr() technique of Series

Correlation essentially assesses how strongly two variables are related. The correlation between the percent change in prices for Microsoft and IBM is "0.499766114415114," which shows that the two columns are strongly correlated in the positive direction. If the percent change in prices for Microsoft increases, the percentage changes in price for IBM also increases, and opposite.

**.cov:** The another piece of code to get cov() is “**returns['MSFT'].cov(returns['IBM'])**”

Similar corr() function the cov() function calculates the covariance. On the other hand, the whole correlation or covariance matrix is returned as a DataFrame by the corr() and cov() methods of DataFrame.

Only covariance can decide which way the variables move. The percent change in the prices of Microsoft and IBM tend to change in the same direction, according to this positive "8.8706554797035462e-05" covariance. Covariance measures the total deviation of two random variables from their expected values.

**Question.3**

**Answer:**

The corr() function, which determines the correlation between two numerical columns contained in the same DataFrame, differs from this one. The columns which is not numerical is skipped by itself. On the other hand, corrwith() is used to compare columns with same name.

Example:

import pandas as pd

#First DataFrame

Subject1 = pd.DataFrame({'Grade': ['A', 'B', 'C', 'D', 'E', 'F'],

                    'ClassA\_count': [20, 22, 20, 25, 15, 11],

                    'ClassB\_count': [10, 5, 20, 10, 9, 12],

                    'ClassC\_count': [8, 9, 9, 7, 2, 5]})

print("Subject1\n", Subject1)

#Second DataFrame

Subject2 = pd.DataFrame({'Grade': ['A', 'B', 'C', 'D', 'E', 'F'],

                    'ClassA\_count': [22, 25, 27, 35, 25, 20],

                    'ClassB\_count': [15, 16, 8, 8, 5, 8],

                    'ClassC\_count': [4, 12, 12, 8, 7, 10]})

print("Subject2\n", Subject2)

Output:

Subject1

Grade ClassA\_count ClassB\_count ClassC\_count

0 A 20 10 8

1 B 22 5 9

2 C 20 20 9

3 D 25 10 7

4 E 15 9 2

5 F 11 12 5

Subject2

Grade ClassA\_count ClassB\_count ClassC\_count

0 A 22 15 4

1 B 25 16 12

2 C 27 8 12

3 D 35 8 8

4 E 25 5 7

5 F 20 8 10

Input:

#calculate correlation between Subject colums grades count in each DataFrame

Subject1.corrwith(Subject2)

Output:

ClassA\_count 0.760731

ClassB\_count -0.444500

ClassC\_count 0.343499

dtype: float64

**Question.4**

**Answer.1**

index = index.drop([“Colorado”, “Ohio”])

**Answer.2**

data.drop(data.columns[[1]], axis=1, inplace=True)

**Question.5**

**Answer.1**

“NaN” stands for “Not A Number” and it represents that the data is missing.

**Answer.2**

Because no value has been set to or specified for obj1, it is NaN. As a result, California's value is NaN.