Python Project - Marvel Mart Project Cam Nguyen 3/13/22

In [222…

```
import numpy as np
import pandas as pd
from pandas import DataFrame, Series
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
import seaborn as sns
sns.set(style='ticks', palette='Set2')
%matplotlib inline
pd.set_option('display.float_format', lambda x: '%.3f' % x)
import warnings
warnings.filterwarnings('ignore')
import csv
```

Part 1: Cleaning the Data

In [3]:

```
#finding which columns have empty values and placing 'Null' into the empty spot
print('Printing columns and the sum of their missing values')
sales = pd.read_csv('DataSamples/MM_Sales.csv')
sumNA = sales.isna().sum()
print(sumNA)
salesClean = sales.copy()
salesClean.fillna('NULL', inplace = True)
print('\n')

#for country need to find numbers disguised as strings and replace with Null
#country: has to be string can't be number or negative but number will be disgui
#convert string to int. then test to see if it turns into a int if it does repla
print('Prints numbers disguised as strings in Country column')
count = 0
for k,v in salesClean.iterrows():
    try:
        v.loc['Country'] = float(v.loc['Country'])
        print(v.loc['Country'])
        salesClean.at[k,'Country'] = 'NULL'
        #does v.loc actually change the value or just in the loop
    except:
        count += 1
print(str(50000-count) + ' incorrect values in Country column')
print('\n')

#for ItemType need to group the itemtypes and find the ones that don't make sens
print('Searching for unique item types')
print(salesClean['Item Type'].value_counts())
print('No singular item types')
print('\n')


#For orderpriority need to find ones that aren't C, H, M, L, or Null and replace
#For orderpriority need to find ones that aren't C, H, M, L, or Null and replace
priorityLabels = ['C', 'H', 'M', 'L', 'NULL']
labels = pd.DataFrame(priorityLabels)
count = 0
for k,v in salesClean.iterrows():
    if v.loc['Order Priority'] not in labels.values:
        print(salesClean.at[k,'Order Priority'])
```

```
        salesClean.at[k,'Order Priority'] = 'NULL'
    else:
        count +=1
print(str(50000 - count) + ' incorrect values in Order Priority column')

#OrderID - want a number and not negative
print('\nTesting incorrect data in Order ID')
count = 0
for index, row in salesClean.iterrows():
    try:
        row.loc['Order ID'] = float(row.loc['Order ID'])
        row.loc['Order ID'] < 0
    except:
        print(row.loc['Order ID'])
        row.loc['Order ID'] = 0
        count += 1
print('Number of incorrect Order ID: ' + str(count))

#removing rows with 'Null' in them
salesClean = salesClean[salesClean.Country != "NULL"]
salesClean = salesClean[salesClean['Order Priority'] != "NULL"]
salesClean = salesClean[salesClean['Item Type'] != 'NULL']

#removing rows with 0
salesClean = salesClean[salesClean['Order ID'] != 0]
```

```
Printing columns and the sum of their missing values
Region              0
Country             0
Item Type           6
Sales Channel       0
Order Priority     15
Order Date          0
Order ID            0
Ship Date           0
Units Sold          0
Unit Price          0
Unit Cost           0
Total Revenue       0
Total Cost          0
Total Profit        0
dtype: int64


Prints numbers disguised as strings in Country column
154.06
437.2
651.21
3 incorrect values in Country column


Searching for unique item types
Meat                4221
Fruits              4221
Cosmetics           4192
Vegetables          4189
Personal Care       4185
Beverages           4173
Snacks              4163
Clothes             4155
```

```
Cereal             4141
Household          4138
Office Supplies    4138
Baby Food          4078
NULL                  6
Name: Item Type, dtype: int64
No singular item types


0 incorrect values in Order Priority column

Testing incorrect data in Order ID
Cosmetics
Fruits
Snacks
Meat
Snacks
Number of incorrect Order ID: 5
```

In [5]:
```
salesClean.to_csv('DataSamples/MM_Sales_clean.csv')
```

In [6]:
```
newSal = pd.read_csv('DataSamples/MM_Sales_clean.csv')
```

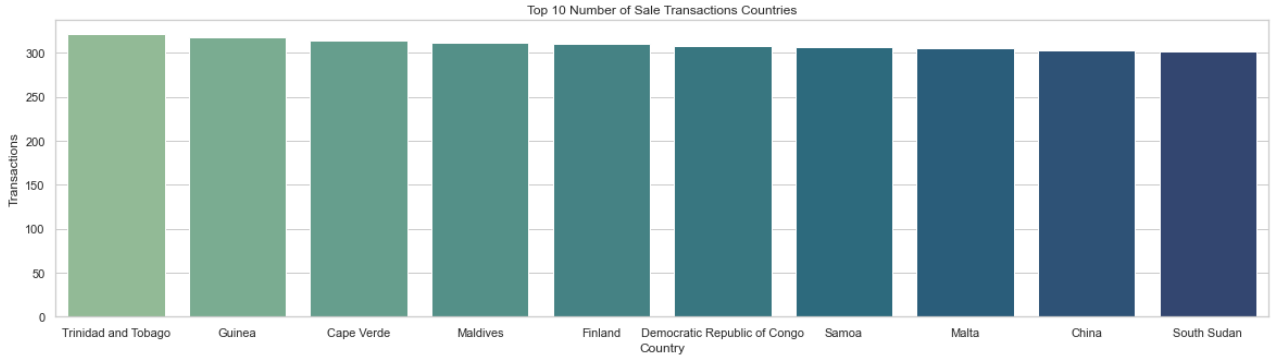
Part 2: General Statistics

1(A)

In [119…
```
#find top 10 countries with the most number of sales transactions and the number
salesClean = pd.read_csv('DataSamples/MM_Sales_clean.csv', sep=',')

countryDF = pd.DataFrame(salesClean['Country'].value_counts())
print(countryDF['Country'].nlargest(n=10))
top10 = ({'Country':['Trinidad and Tobago', 'Guinea', 'Cape Verde', 'Maldives',
                     'Samoa','Malta', 'China', 'South Sudan'], 'Transactions':[3
rankedDF = pd.DataFrame(top10)

sns.set(style='whitegrid')
plt.figure(figsize=(20, 5))
ax = sns.barplot(rankedDF['Country'], rankedDF['Transactions'], palette = 'crest
ax.set_title('Top 10 Number of Sale Transactions Countries')
```

```
Trinidad and Tobago                321
Guinea                             318
Cape Verde                         314
Maldives                           311
Finland                            310
Democratic Republic of the Congo   308
Samoa                              306
Malta                              305
China                              303
South Sudan                        302
Name: Country, dtype: int64
```

Out[119…
```
Text(0.5, 1.0, 'Top 10 Number of Sale Transactions Countries')
```



1(B)

In [135…

```
with open('DataSamples/MM_Rankings.txt', 'a+') as appender:
    appender.write('Countries Most Sale Transactions:\n')
    for k in range(len(rankedDF)):
        appender.write(f'{rankedDF.iloc[k,0]}: {rankedDF.iloc[k,1]}\n')
    appender.write('The country we should build our shipping center is Cape Verd
```

2(A)

In [67]:

```
channelType = salesClean['Sales Channel'].value_counts()
print(channelType)
```

```
Online     30185
Offline    19791
Name: Sales Channel, dtype: int64
```

2(B)

In [65]:

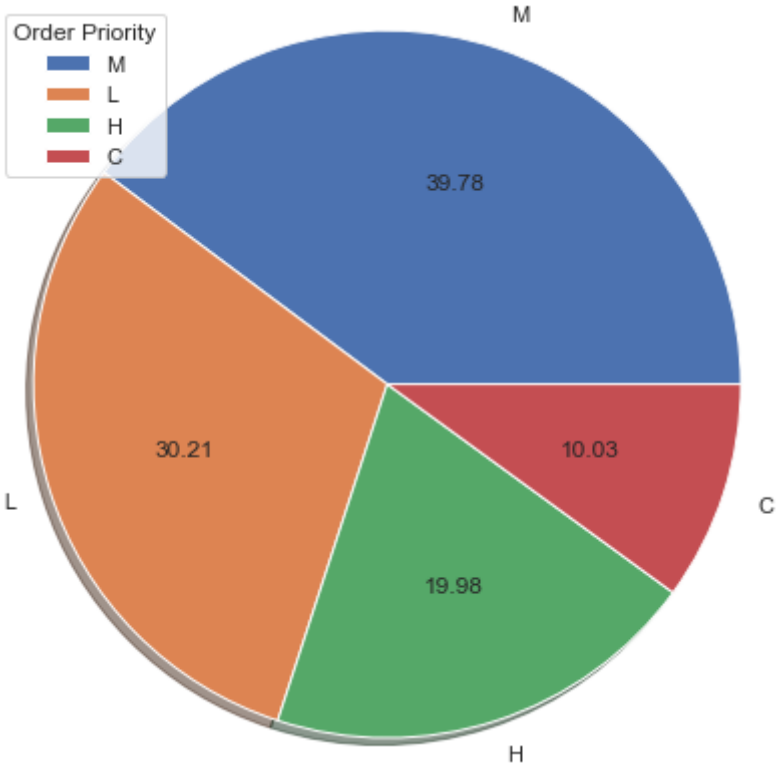
```
priorTypes = salesClean['Order Priority'].value_counts()
print(priorTypes)
```

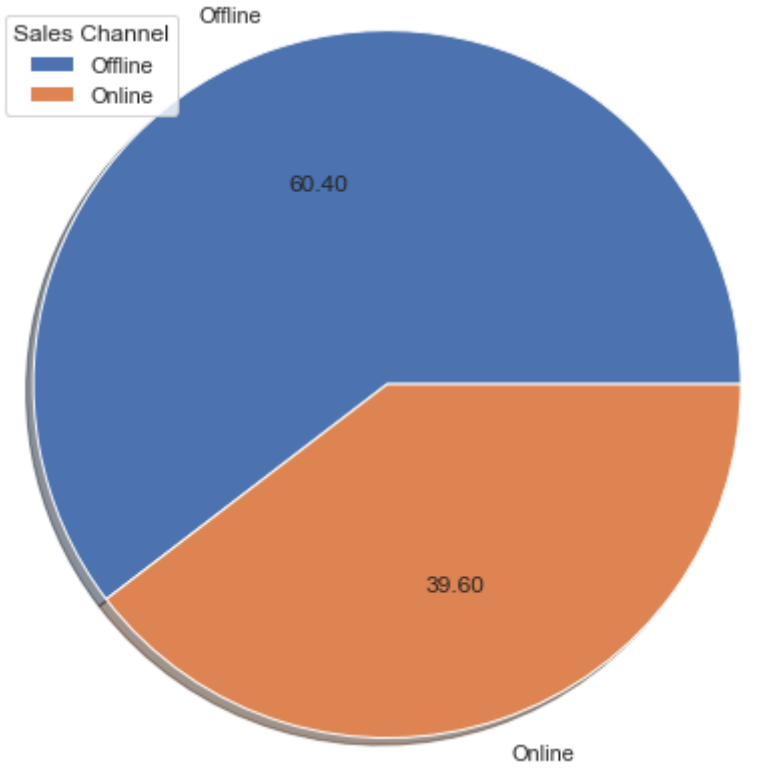
```
M    19882
H    15097
L     9985
C     5012
Name: Order Priority, dtype: int64
```

2(C)

In [73]:

```
plt.figure(figsize=(7, 7))
plt.pie(priorTypes, labels= salesClean['Order Priority'].unique() , shadow=True,
plt.axis('equal')  # centers pie chart
plt.legend(loc=2, title='Order Priority')
plt.show()

plt.figure(figsize=(7, 7))
plt.pie(channelType, labels= salesClean['Sales Channel'].unique() , shadow=True,
plt.axis('equal')  # centers pie chart
plt.legend(loc=2, title='Sales Channel')
plt.show()
```





2(D)

In [136…

```
with open('DataSamples/MM_Rankings.txt', 'a+') as appender:
    chanType = ['\nSales Channels:\n', 'Offline: 19791\nOnline: 30185\n', 'We do
    appender.writelines(chanType)
    appender.write('\n')

    appender.write('\nOrder Priorities:\n')
    priorOrders = {'M': 19882, 'H': 15097, 'L': 9985, 'C': 5012}
    for k,v in priorOrders.items():
```

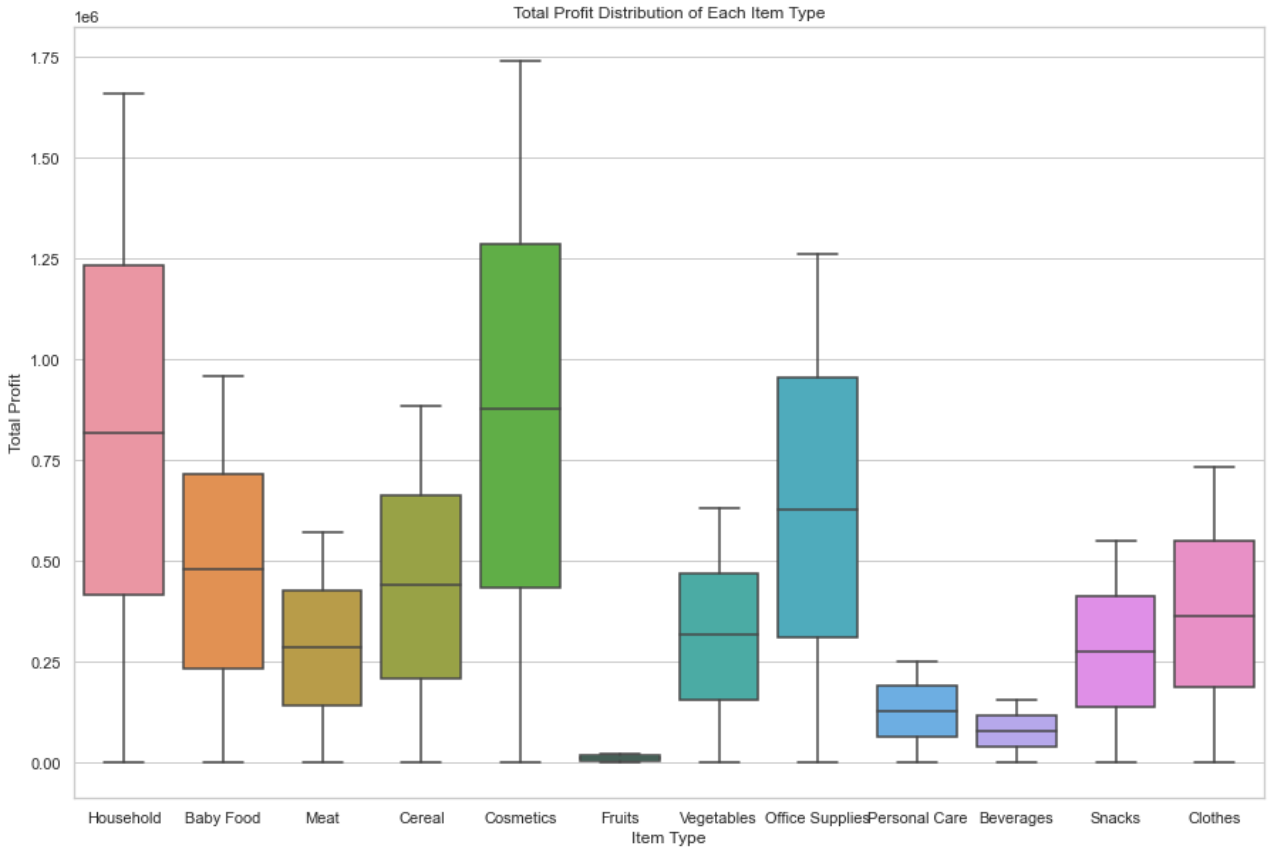
```
        appender.write(k + ': ' + str(v) + '\n')
    appender.write('We do more M order priorities')
```

3(A)

In [116…

```
plt.figure(figsize=(15, 10))
ax = sns.boxplot(x=salesClean['Item Type'], y=salesClean['Total Profit'], data=s
ax.set_title('Total Profit Distribution of Each Item Type')
```

Out[116… Text(0.5, 1.0, 'Total Profit Distribution of Each Item Type')



3(B)

In [201…

```
typeProfit = (salesClean.groupby(['Item Type'])['Total Profit'].sum())
print(typeProfit)
```

```
Item Type
Baby Food          1942865748.120
Beverages           327559249.080
Cereal             1824726412.290
Clothes            1520832019.680
Cosmetics          3638645299.300
Fruits               51025156.240
Household          3401180998.060
Meat               1196826774.000
Office Supplies    2605440187.500
Personal Care       535250525.600
Snacks             1150281274.560
Vegetables         1322639660.240
Name: Total Profit, dtype: float64
```

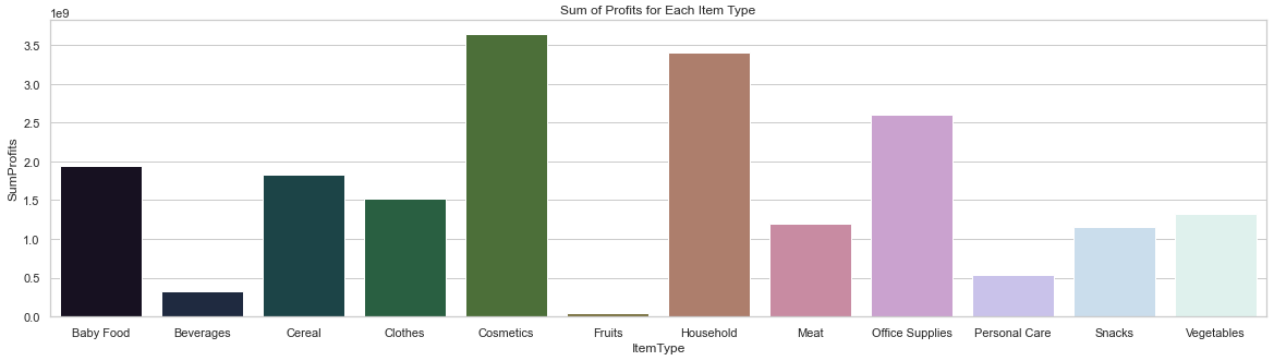
3(C)

In [210…

```
sumDict = ({'ItemType':['Baby Food', 'Beverages', 'Cereal', 'Clothes', 'Cosmetic
                        'Personal Care', 'Snacks', 'Vegetables'], 'SumProfits':[194
                                                                                1520
                                                                                1196
                                                                                1322

sumDF = pd.DataFrame(sumDict)

sns.set(style='whitegrid')
plt.figure(figsize=(20, 5))
ax = sns.barplot(y=sumDF['SumProfits'], x=sumDF['ItemType'], palette = 'cubeheli
ax.set_title('Sum of Profits for Each Item Type')
```

Out[210…

```
Text(0.5, 1.0, 'Sum of Profits for Each Item Type')
```



3(D)

In [123…

```
DF = pd.DataFrame(salesClean.groupby(['Item Type'])['Total Profit'].sum().nlarge
print(DF)
```

```
                Total Profit
Item Type
Cosmetics       3638645299.300
Household       3401180998.060
Office Supplies 2605440187.500
```

3(E)

In [137…

```
with open('DataSamples/MM_Rankings.txt', 'a+') as appender:
    appender.write('\n')
    topItems = ['\nHigh Selling Items:\n', 'Cosmetics: 3638645299.30\n', 'Househ
                'Office Supplies: 2605440187.50\n','We profited from Cosmetics t
    appender.writelines(topItems)
    appender.write('\n')
```

3(F)

In [ ]:

```
'''Provide a markdown section discussing the results of the boxplots. Discuss wh
amd do some business analytics around what sort of use this sort of chart might
Are there any unexpected results? Discuss them.'''
```

4(A)

In [149…
```
#sum of Units Sold, Unit Cost, Total Revenue, Total Cost and Total Profit
print('Sum:')
print('Units sold: ' + str(salesClean['Units Sold'].sum()))
print('Units cost: ' + str(salesClean['Unit Cost'].sum()))
print('Total Revenue: ' + str(salesClean['Total Revenue'].sum()))
print('Total Cost: ' + str(salesClean['Total Cost'].sum()))
print('Total Profit: ' + str(salesClean['Total Profit'].sum()))
```

```
Sum:
Units sold: 249844291
Units cost: 9361598.14
Total Revenue: 66150795085.33
Total Cost: 46633521780.659996
Total Profit: 19517273304.67
```

4(B)

In [148…
```
#averageof Units Sold, Unit Cost, Total Revenue, Total Cost and Total Profit
print('Mean:')
print('Units sold: ' + str(salesClean['Units Sold'].mean()))
print('Units cost: ' + str(salesClean['Unit Cost'].mean()))
print('Total Revenue: ' + str(salesClean['Total Revenue'].mean()))
print('Total Cost: ' + str(salesClean['Total Cost'].mean()))
print('Total Profit: ' + str(salesClean['Total Profit'].mean()))
```

```
Mean:
Units sold: 4999.285477028974
Units cost: 187.32187730109186
Total Revenue: 1323651.25430866
Total Cost: 933118.3324127578
Total Profit: 390532.921895912
```

4(C)

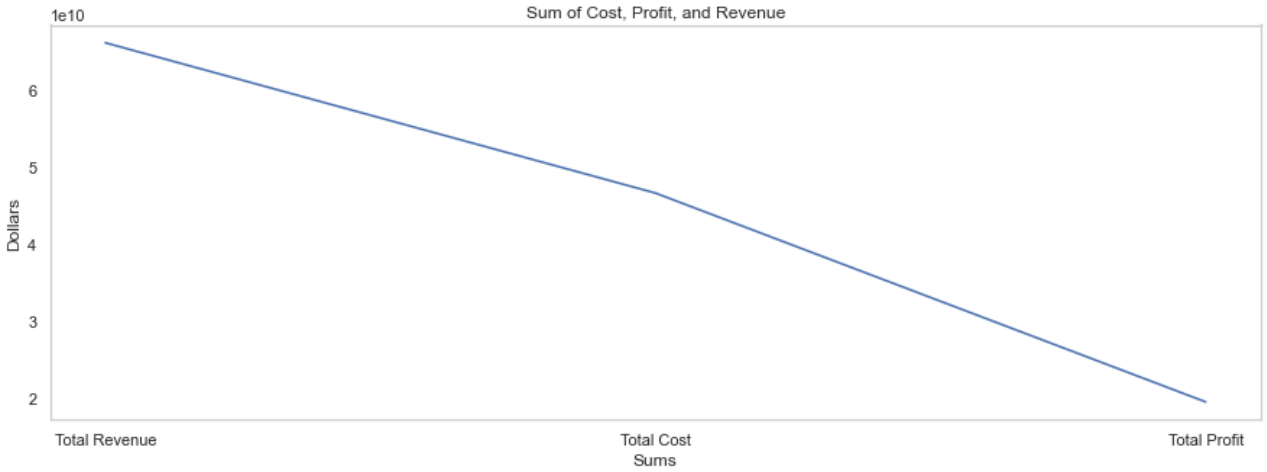
In [147…
```
#max Units Sold, Unit Cost, Total Revenue, Total Cost and Total Profit
print('Max:')
print('Units sold: ' + str(salesClean['Units Sold'].max()))
print('Units cost: ' + str(salesClean['Unit Cost'].max()))
print('Total Revenue: ' + str(salesClean['Total Revenue'].max()))
print('Total Cost: ' + str(salesClean['Total Cost'].max()))
print('Total Profit: ' + str(salesClean['Total Profit'].max()))
```

```
Max:
Units sold: 10000
Units cost: 524.96
Total Revenue: 6682031.73
Total Cost: 5249075.04
Total Profit: 1738178.39
```

4(D)

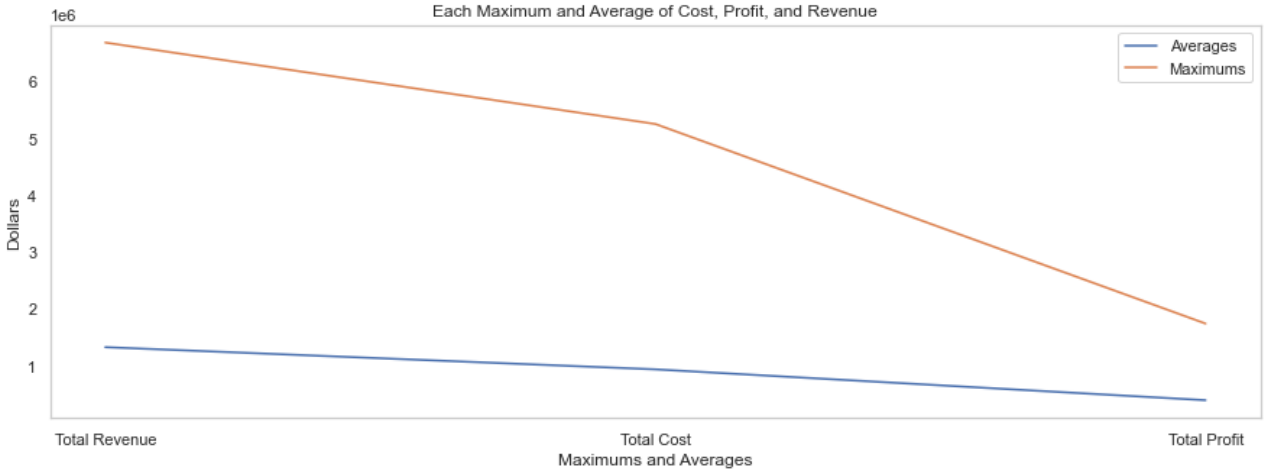
In [157…
```
#Create two line plots using Seaborn or Matplotlib, one for the sums and one for
#DO NOT INCLUDE UNITS SOLD OR UNITS COST.

# plot xy coordinates
# (1, 1), (2, 4), (3, 2), (4, 3)
```

```
sums = [salesClean['Total Revenue'].sum(), salesClean['Total Cost'].sum(), sales
columns = ['Total Revenue', 'Total Cost', 'Total Profit']
plt.figure(figsize=(15, 5))
plt.plot(columns, sums)
plt.title('Sum of Cost, Profit, and Revenue')
plt.xlabel('Sums')
plt.ylabel('Dollars')
plt.grid(False)
plt.show()

#line plot for both average and max
```



In [160…

```
means = [salesClean['Total Revenue'].mean(), salesClean['Total Cost'].mean(), sa
maximums = [salesClean['Total Revenue'].max(), salesClean['Total Cost'].max(), s
columns = ['Total Revenue', 'Total Cost', 'Total Profit']

plt.figure(figsize=(15, 5))
plt.plot(columns, means, label = "Averages")
plt.plot(columns, maximums, label = "Maximums")
plt.title('Each Maximum and Average of Cost, Profit, and Revenue')
plt.xlabel('Maximums and Averages')
plt.ylabel('Dollars')
plt.grid(False)
plt.legend()
plt.show()
```



4(E)

In [164…
```
with open('DataSamples/MM_Calc.txt', 'a+') as appender:
    appender.write('Sums: \n')
    sumList = ['Units Sold: 249844291', 'Units Cost: 9361598.14', 'Total Revenue
               'Total Cost: 46633521780.66', 'Total Profit: 19517273304.67']
    for x in sumList:
        appender.write(x +'\n')

    appender.write('\nAverages: \n')
    meansList = ['Units Sold: 4999.29', 'Units Cost: 187.32', 'Total Revenue: 13
                 'Total Profit: 390532.92']
    for x in meansList:
        appender.write(x + '\n')

    appender.write('\nMaximums: \n')
    maxList = ['Units Sold: 10000', 'Units Cost: 524.96', 'Total Revenue: 668203
               'Total Profit: 1738178.39']
    for x in maxList:
        appender.write(x + '\n')
```

Part 3: Cross-Reference Statistics

3(A)

In [257…
```
#list of regions and countries we sell to in that region
regionGroup = salesClean.groupby(['Region','Country'])['Country'].count()
regionDF = pd.DataFrame(regionGroup)
print(regionGroup)
```

```
Region              Country
Asia                Bangladesh    275
                    Bhutan        258
                    Brunei        252
                    Cambodia      278
                    China         303
                                  ...
Sub-Saharan Africa  The Gambia    281
                    Togo          248
                    Uganda        281
                    Zambia        262
                    Zimbabwe      260
Name: Country, Length: 185, dtype: int64
```

3(B)

In [258…
```
regionDF.to_csv('DataSamples/Countries_By_Region.csv')
```

In [ ]:
```

```