LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS5067NI Smart Data Discovery

Assessment Weightage & Type

60% Individual Coursework

Year and Semester

2022-23 Spring

Student Name: Janawi Shrestha

London Met ID: 21049517

College ID: np01cp4a210171

Assignment Due Date: 4th May 2023

Assignment Submission Date: 3rd May 2023

Word Count: 2673

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Table of Figures

Table of Tables

1. Data Understanding

A dataset containing the sales information about the ABC company is given. The dataset contains the sales information of twelve months of the product purchase history. It comprises of entries of thousands of electronic products broken down by order id, product name, quantity of the products ordered, price of each product, order date of the product, and the addresses of the purchased products.

The csv files are merged into a single file, null values are removed, and the final updated single csv file is stored in a variable, sales_details. To find out the details about the dataset, '.info()' function is called. After calling the function, the details of each column of the dataset are displayed. There are three columns with float datatype and three columns with object datatype. All the columns contain 185950 non-null values. The memory usage of the csv file is almost 10MB.

```
In [6]: sales_details.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 6 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Order ID          185950 non-null  float64
 1   Product           185950 non-null  object 
 2   Quantity Ordered  185950 non-null  float64
 3   Price Each        185950 non-null  float64
 4   Order Date        185950 non-null  object 
 5   Purchase Address  185950 non-null  object 
dtypes: float64(3), object(3)
memory usage: 9.9+ MB
```

Figure 1: Column details

The following table contains the information about each column of the dataset.

SN	Column Name	Description	Data type
1	Order ID	This column stores the id of the product that has been ordered. It is the primary key for this set of data, as a new order id is generated each time an order is made.	Float
2	Product	This column stores the name of the products distributed by the company. By analysing the data, the sales of 19 products have been showcased in the csv files.	Object
3	Quantity Ordered	This column stores the number of the products that has been ordered by the customers.	Float
4	Price Each	This column stores the price of each product that has been ordered.	Float
5	Order Date	This column stores date and time of the order when an order is made by a customer. It stores date in DD/MM/YYYY format and time in H:mm format.	Object
6	Purchase Address	This column stores the address of the order from where the product has been purchased. It stores the data with the street name, city name and city code.	Object

Table 1: Data Understanding

Moreover, three additional columns will be added along the programming process. The columns are Month, City and Sales. The Month column will be separated from the Order Date column which will extract the month that the products were ordered. The City column will be separated from the Purchase Address column which will have the values of the city name from Purchase Address column. The Sales column is the product of Quantity Ordered and Price Each column. The purpose of the sales column is to demonstrate the total sales of each product ordered.

```
In [28]: sales_details.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 9 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   Order ID          185950 non-null  float64       
 1   Product           185950 non-null  object        
 2   Quantity Ordered  185950 non-null  float64       
 3   Price Each        185950 non-null  float64       
 4   Order Date        185950 non-null  datetime64[ns]
 5   Month             185950 non-null  int32         
 6   Purchase Address  185950 non-null  object        
 7   City              185950 non-null  object        
 8   Sales             185950 non-null  float64       
dtypes: datetime64[ns](1), float64(4), int32(1), object(3)
memory usage: 13.5+ MB
```

Figure 2: Final columns

In the process, the datatype of datetime has been changed to datetime from object. The datatype of Month is assigned to int, City to object, and since we have multiplied two columns with float datatype, the datatype of Sales will be float as well. With the increase in the number of columns, the memory usage has also increased.

2. Data Preparation

Data Preparation is the phase of transforming raw data into useful information that will later be used for decision-making. Data sources are merged and filtered. They are finally aggregated, and the raw data are subject to the calculation of additional values. It is mainly the phase that precedes the analysis (Ryax Technologies, 2023).

2.1 Question 1

Question: Write a python program to merge data from each month into one CSV and read in updated dataframe.

Solution:

In [4]:
```
# creating an empty DataFrame to hold the combined contents of all csv files
csv_list = pd.DataFrame()

for file in file_list:
        df_ = pd.read_csv(file)
        csv_list = csv_list.append(df_, ignore_index = True) # adding all of the rows of 12 csv files to the combined dataframe

csv_list
```

Out[4]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558.0	USB-C Charging Cable	2.0	11.95	4/19/2019 8:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559.0	Bose SoundSport Headphones	1.0	99.99	4/7/2019 22:30	682 Chestnut St, Boston, MA 02215
3	176560.0	Google Phone	1.0	600.00	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560.0	Wired Headphones	1.0	11.99	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001
...	...	...	...	...	...	...
186845	259353.0	AAA Batteries (4-pack)	3.0	2.99	9/17/2019 20:56	840 Highland St, Los Angeles, CA 90001
186846	259354.0	iPhone	1.0	700.00	9/1/2019 16:00	216 Dogwood St, San Francisco, CA 94016
186847	259355.0	iPhone	1.0	700.00	9/23/2019 7:39	220 12th St, San Francisco, CA 94016
186848	259356.0	34in Ultrawide Monitor	1.0	379.99	9/19/2019 17:30	511 Forest St, San Francisco, CA 94016
186849	259357.0	USB-C Charging Cable	1.0	11.95	9/30/2019 0:18	250 Meadow St, San Francisco, CA 94016

186850 rows × 6 columns

Figure 3: Merging all csv files

A total of 12 csv files are given in the provided dataset. These files are to be merged into a single csv files and further actions are performed. At first, an empty dataframe ‘csv_list’ is created in which all the rows from each csv file will be appended. Then, for function is run to read the data of the csv files and append them to the empty dataset that was previously created. Lastly, the dataframe is displayed, and a total of 186850 rows and 6 columns are shown in the merged dataframe.

2.2 Question 2

Question: Write a python program to remove the NaN missing values from updated dataframe.

Solution:

```
In [5]: # dropping all of the rows with null values
        sales_details = csv_list.dropna(axis = 0)

        sales_details
```

Out[5]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558.0	USB-C Charging Cable	2.0	11.95	4/19/2019 8:46	917 1st St, Dallas, TX 75001
2	176559.0	Bose SoundSport Headphones	1.0	99.99	4/7/2019 22:30	682 Chestnut St, Boston, MA 02215
3	176560.0	Google Phone	1.0	600.00	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560.0	Wired Headphones	1.0	11.99	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561.0	Wired Headphones	1.0	11.99	4/30/2019 9:27	333 8th St, Los Angeles, CA 90001
...	...	...	...	...	...	...
186845	259353.0	AAA Batteries (4-pack)	3.0	2.99	9/17/2019 20:56	840 Highland St, Los Angeles, CA 90001
186846	259354.0	iPhone	1.0	700.00	9/1/2019 16:00	216 Dogwood St, San Francisco, CA 94016
186847	259355.0	iPhone	1.0	700.00	9/23/2019 7:39	220 12th St, San Francisco, CA 94016
186848	259356.0	34in Ultrawide Monitor	1.0	379.99	9/19/2019 17:30	511 Forest St, San Francisco, CA 94016
186849	259357.0	USB-C Charging Cable	1.0	11.95	9/30/2019 0:18	250 Meadow St, San Francisco, CA 94016

185950 rows × 6 columns

Figure 4: Dropping all null values

While performing and running various function, the function will throw an error because of the presence of null values in the dataset. So, the null values should be removed from all the columns. The null values in the dataset are removed by calling the '.dropna()' function. This function checks the null values, as axis is assigned 0, it will check null values in each row of the dataset and will delete the row if a null value is discovered. As it is seen, the number of rows has been decreased from 186850 to 185950. Hence, the null values from the dataset have been deleted and the updated dataset has been assigned to a new variable 'sales_details'.

2.3 Question 3

Question: Write a python program to convert Quantity Ordered and Price Each to numeric.

Solution:

```
In [7]: # changing the data type of the columns 'Quantity Ordered' and 'Price Each' to numeric data type (i.e. float)
        sales_details['Quantity Ordered'] = pd.to_numeric(sales_details['Quantity Ordered'])
        sales_details['Price Each'] = pd.to_numeric(sales_details['Price Each'])

        sales_details.info()

        <class 'pandas.core.frame.DataFrame'>
        Int64Index: 185950 entries, 0 to 186849
        Data columns (total 6 columns):
         #   Column            Non-Null Count   Dtype
        ---  ------            --------------   -----
         0   Order ID          185950 non-null  float64
         1   Product           185950 non-null  object
         2   Quantity Ordered  185950 non-null  float64
         3   Price Each        185950 non-null  float64
         4   Order Date        185950 non-null  object
         5   Purchase Address  185950 non-null  object
        dtypes: float64(3), object(3)
        memory usage: 9.9+ MB
```

Figure 5: Datatype Conversion

Since pandas library has been imported, the in-built functions of this library can be called to bring changes in the dataset. Here, the datatype of the columns ‘Quantity Ordered’ and ‘Price Each’ has been changed to numeric datatype by the use of the function ‘to_numeric()’.

2.4 Question 4

Question: Create a new column named Month from Ordered Date of updated dataframe and convert it to integer as data type.

Solution:

```
In [8]: # changing the data type of 'Order Date' from object to datetime
sales_details['Order Date'] = pd.to_datetime(sales_details['Order Date'])

# inserting a column with heading 'Month' with datatype 'int' in the 5th index and
# the values will be extracted from month part of 'Order Date'
sales_details.insert(5, "Month", sales_details['Order Date'].dt.strftime('%m').astype(int))

sales_details
```

Out[8]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Month	Purchase Address
0	176558.0	USB-C Charging Cable	2.0	11.95	2019-04-19 08:46:00	4	917 1st St, Dallas, TX 75001
2	176559.0	Bose SoundSport Headphones	1.0	99.99	2019-04-07 22:30:00	4	682 Chestnut St, Boston, MA 02215
3	176560.0	Google Phone	1.0	600.00	2019-04-12 14:38:00	4	669 Spruce St, Los Angeles, CA 90001
4	176560.0	Wired Headphones	1.0	11.99	2019-04-12 14:38:00	4	669 Spruce St, Los Angeles, CA 90001
5	176561.0	Wired Headphones	1.0	11.99	2019-04-30 09:27:00	4	333 8th St, Los Angeles, CA 90001
...	...	...	...	...	...	...	...
186845	259353.0	AAA Batteries (4-pack)	3.0	2.99	2019-09-17 20:56:00	9	840 Highland St, Los Angeles, CA 90001
186846	259354.0	iPhone	1.0	700.00	2019-09-01 16:00:00	9	216 Dogwood St, San Francisco, CA 94016
186847	259355.0	iPhone	1.0	700.00	2019-09-23 07:39:00	9	220 12th St, San Francisco, CA 94016
186848	259356.0	34in Ultrawide Monitor	1.0	379.99	2019-09-19 17:30:00	9	511 Forest St, San Francisco, CA 94016
186849	259357.0	USB-C Charging Cable	1.0	11.95	2019-09-30 00:18:00	9	250 Meadow St, San Francisco, CA 94016

185950 rows × 7 columns

Figure 6: Inserting Month column

The order date column is originally in object datatype format. At first, the datatype of order date column is changed from object to datetime using the ‘to_datetime()’ function of the pandas library. The question has asked to add a new column named month from the column order date. A new column named ‘Month’ is inserted in the sixth index, the value of month will be extracted from the order date column, which was previously converted into datetime datatype, and the ‘Month’ column is assigned the int datatype with the use of ‘astype()’ function.

2.5 Question 5

Question: Create a new column named City from Purchase Address based on the value in updated dataframe.

Solution:

In [10]:
```
# initializing empty list to append the names of cities
cities = []

# loop through each value of Purchase Address to extract the name of the cities only
for x in sales_details['Purchase Address'].values.astype(str):
    city = x.split(',')
    cities.append(city[1])

# inserting Cities column into the dataframe at index 7
sales_details.insert(7, "City", cities)
sales_details
```

Out[10]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Month	Purchase Address	City
0	176558.0	USB-C Charging Cable	2.0	11.95	2019-04-19 08:46:00	4	917 1st St, Dallas, TX 75001	Dallas
2	176559.0	Bose SoundSport Headphones	1.0	99.99	2019-04-07 22:30:00	4	682 Chestnut St, Boston, MA 02215	Boston
3	176560.0	Google Phone	1.0	600.00	2019-04-12 14:38:00	4	669 Spruce St, Los Angeles, CA 90001	Los Angeles
4	176560.0	Wired Headphones	1.0	11.99	2019-04-12 14:38:00	4	669 Spruce St, Los Angeles, CA 90001	Los Angeles
5	176561.0	Wired Headphones	1.0	11.99	2019-04-30 09:27:00	4	333 8th St, Los Angeles, CA 90001	Los Angeles
...	...	...	...	...	...	...	...	...
186845	259353.0	AAA Batteries (4-pack)	3.0	2.99	2019-09-17 20:56:00	9	840 Highland St, Los Angeles, CA 90001	Los Angeles
186846	259354.0	iPhone	1.0	700.00	2019-09-01 16:00:00	9	216 Dogwood St, San Francisco, CA 94016	San Francisco
186847	259355.0	iPhone	1.0	700.00	2019-09-23 07:39:00	9	220 12th St, San Francisco, CA 94016	San Francisco
186848	259356.0	34in Ultrawide Monitor	1.0	379.99	2019-09-19 17:30:00	9	511 Forest St, San Francisco, CA 94016	San Francisco
186849	259357.0	USB-C Charging Cable	1.0	11.95	2019-09-30 00:18:00	9	250 Meadow St, San Francisco, CA 94016	San Francisco

185950 rows × 8 columns

Figure 7: Inserting City column

The question suggests creating a column with name City from Purchase Address. At first, an empty list is created to append the names of the cities. Then, the names of the cities are appended from the for loop that is being run in the program as shown above. In this loop, the street name, city name, and city code are separated by a comma, and the city name which is in the second index is appended to the previously created list. Lastly, a column with name ‘City’ is inserted in the eighth index and the values for this column are taken form the cities list in which the name of the cities are appended.

3. Data Analysis

Data analysis is the process of collecting, modeling, and analyzing data using various statistical and logical methods and techniques. Businesses rely on analytics processes and tools to extract insights that support strategic and operational decision-making (Calzon, 2023).

3.1 Question 1

Question: Write a Python program to show summary statistics of sum, mean, standard deviation, skewness, and kurtosis of any chosen variable.

Solution:

3.1.1 Sum

```
In [33]: # calculating the sum of 'Sales' column
         price_sum = sales_details['Sales'].sum()

         print('The sum of the total sales is ' + str(price_sum) + '.')

         The sum of the total sales is 34492035.97.
```

Figure 8: Sum of the data of 'Sales' variable

The sum of the values in the ‘Sales’ column of the updated merged csv file is taken out with the help of the function, sum(). Then the total sum is displayed, the total sum of the sales is found to be 34492035.97.

3.1.2 Mean

```
In [34]: # taking the mean out of 'Sales' column
         price_mean = sales_details['Sales'].mean()

         print('The mean of the total sales is ' + str(price_mean) + '.')

         The mean of the total sales is 185.49091675146175.
```

Figure 9: Mean of the data of 'Sales' variable

Mean is the average of all given values, it is the sum of all the values divided by the total number of values. The mean of the values in the ‘Sales’ column of the updated merged csv file is taken out with the help of the function, mean(). Then the total mean is displayed, the total mean of the sales is found to be 185.49.

3.1.3 Standard Deviation

```
In [35]: # calculating the Standard Deviation of the values of 'Sales' column
         price_std = sales_details['Sales'].std()

         print('The Standard Deviation of the total sales is ' + str(price_std) + '.')

         The Standard Deviation of the total sales is 332.91977138642756.
```

Figure 10: Standard Deviation of the data of 'Sales' variable

Standard deviation is the degree of dispersion or the scatter of the data points relative to its mean. It tells how the values are spread across the data sample and it is the measure of the variation of the data points from the mean (Cuemath, 2023). The standard deviation of the values in the ‘Sales’ column of the updated merged csv file is taken out with the help of the function, std(). Then the standard deviation is displayed, and the standard deviation of the sales is found to be 332.92.

3.1.4 Skewness

```
In [36]: # calculating the skewness among the values of 'Sales' column
         price_skew = sales_details['Sales'].skew()

         print('The skewness of the total sales is ' + str(price_skew) + '.')

         The skewness of the total sales is 2.8819126688871703.
```

Figure 11: Skewness of the data of 'Sales' variable

Skewness is a measure of asymmetry or distortion of symmetric distribution. It measures the deviation of the given distribution of a random variable from a symmetric distribution (Taylor, 2023). Here, the skewness of the values in the ‘Sales’ column of the updated merged csv file is taken out with the help of the function, skew(). Then the skewness is displayed, and it is found to be 2.88.

3.1.5 Kurtosis

```
In [37]: # creating a new dataframe called "kurtosis" containing only the "Sales" column
kurtosis = pd.DataFrame(sales_details['Sales'])

# calcualting Kurtosis of the values in the kurtosis dataframe
price_kurtosis = kurtosis.kurt()

print('The kurtosis of the total sales is ' + str(price_kurtosis) + '.')
```

```
The kurtosis of the total sales is Sales    9.227075
dtype: float64.
```

Figure 12: Kurtosis of the data of 'Sales' variable

Kurtosis is a statistical measure used to describe the degree to which scores cluster in the tails or the peak of a frequency distribution. The peak is the tallest part of the distribution, and the tails are the ends of the distribution (Mcleod, 2023). Here, the kurosis of the values in the ‘Sales’ column of the updated merged csv file is taken out with the help of the function, kurt(). Then the kurtosis is displayed, and it is found to be 9.22.

3.2 Question 2

Question: Write a Python program to calculate and show correlation of all variables.

Solution:

3.2.1 Correlation

```
In [30]: # finding out the correlation between columns of the dataframe
         # by default Pearson method is being used
         all_corr = sales_details.corr()

         all_corr
```

Out[30]:

	Order ID	Quantity Ordered	Price Each	Month	Sales
Order ID	1.000000	0.000702	-0.002857	0.993063	-0.002949
Quantity Ordered	0.000702	1.000000	-0.148272	0.000791	-0.139417
Price Each	-0.002857	-0.148272	1.000000	-0.003375	0.999203
Month	0.993063	0.000791	-0.003375	1.000000	-0.003466
Sales	-0.002949	-0.139417	0.999203	-0.003466	1.000000

Figure 13: Correlation of all variables

Correlation is the association between two variables. There are three types of correlation: positive correlation, negative correlation, and zero correlation. A positive correlation is a relationship between two variables in which both variables move in the same direction. Therefore, when one variable increases as the other variable increases or one variable decreases while the other decreases. A negative correlation is a relationship between two variables in which an increase in one variable is associated with a decrease in the other. A zero correlation exists when there is no relationship between two variables (Mcleod, 2023).

In this dataset, there is a strong positive correlation between Order ID & Month, which indicates that higher the number of months, higher will be the orders in that month, and Price Each & Sales, which means the presence of the higher number of Price Each in the dataset will automatically increase the number of sales. A weak negative correlation exists between Quantity Ordered & Price Each, which specifies that increase in price of the product will decrease the quantity of the product that will be ordered, and Quantity Ordered & Sales, which directs that if the price of the product is low then the product will be ordered more, i.e., quantity ordered will increase.

4. Data Exploration

Data Exploration is the stage following the preparation phase. The prepared data is then analysed to enable the questions arising from the data preparation to be answered. The data provided is explored interactively. They are reorganized in such a way that they are presented in an understandable way and used by decision-makers. It is therefore a question of exploring data that has not yet been transformed (Ryax Technologies, 2023).

4.1 Question 1

Question: Which Month has the best sales? and how much was the earning in that month? Make a bar graph of sales as well.

Solution:

```
In [26]: bestSales = sales_details.groupby('Month').sum()
         bestSales
```

Out[26]:

Month	Order ID	Quantity Ordered	Price Each	Sales
1	1.421631e+09	10903.0	1811768.38	1822256.73
2	1.871053e+09	13449.0	2188884.72	2202022.42
3	2.564811e+09	17005.0	2791207.83	2807100.38
4	3.387347e+09	20558.0	3367671.02	3390670.24
5	3.345872e+09	18667.0	3135125.13	3152606.75
6	2.932976e+09	15253.0	2562025.61	2577802.26
7	3.284140e+09	16072.0	2632539.56	2647775.76
8	2.899374e+09	13448.0	2230345.42	2244467.88
9	2.948727e+09	13109.0	2084992.09	2097560.13
10	5.457110e+09	22703.0	3715554.83	3736726.88
11	5.047203e+09	19798.0	3180600.68	3199603.20
12	7.685905e+09	28114.0	4588415.41	4613443.34

Figure 14: Best sales month

From the figure above, it is seen that December is the month with the best sales for ABC company with approximately USD 4,610,000. It is the highest among all. By analysing the result, it is almost 20-30% more than the sales of other months.

```
In [31]: # assigning a variable to store the sum of the sales grouped by month
bestSales = sales_details.groupby('Month').sum()
best_sales = bestSales['Sales']

months = range(1, 13)
plt.bar(months, best_sales, color="#ed549b")
plt.title("\nSales stat in each month") # title of the bar graph
plt.xticks(months, ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])
labels, location = plt.yticks()
plt.yticks(labels, (labels/1000000).astype(int))
plt.xlabel("Months")
plt.ylabel("Sales in million(USD)")
plt.show()
```

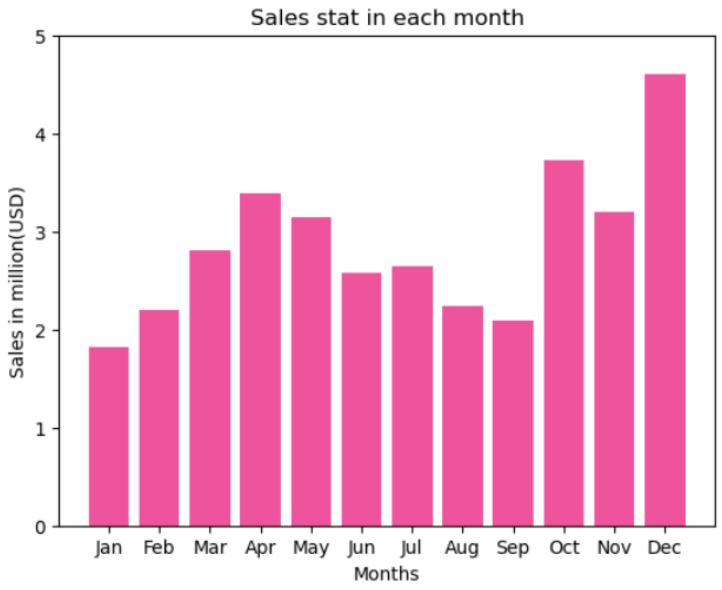

Figure 15: Sales stat in each month

The statistical graph of the sales in each month has been displayed in the figure above. From the bar graph, sales are the highest in the month of **December** reaching almost **5 million USD** and lowest in the month of **January** with approximately **1.8 million USD**. Since, December is the month of Christmas and New Years Eve and people tend to gift stuffs to other people, the sales might have increased due to this in that month. By the end of December, people would have already bought the products to gift, so there might have been not much sale in January.

The bar graph is plotted using the matplotlib library. The colour codes have been decided, the x-axis and y-axis have been labelled as ‘Months’ and ‘Sales in million(USD)’ respectively, and the values of x-axis which were originally the numbers ranging from 1 to 13, have been replaced by 12 months of a year, January to December. The values of y-axis which were originally in exponential form have been converted into million USD.

4.2 Question 2

Question: Which city has sold the highest product?

Solution:

```
In [30]: # assigning a varaible to store the sum of sales grouped by cities
citySales = sales_details.groupby('City').sum()
city_sales = citySales['Quantity Ordered']

plt.bar(city_sales.index, city_sales.values, color="#ed547d", width=0.6)
plt.xlabel("\nCities")
plt.ylabel("Quantity Ordered\n")
plt.title("\nQuantity of the products ordered in different cities")
plt.xticks(cities, rotation='vertical', size=8)
plt.show()
```

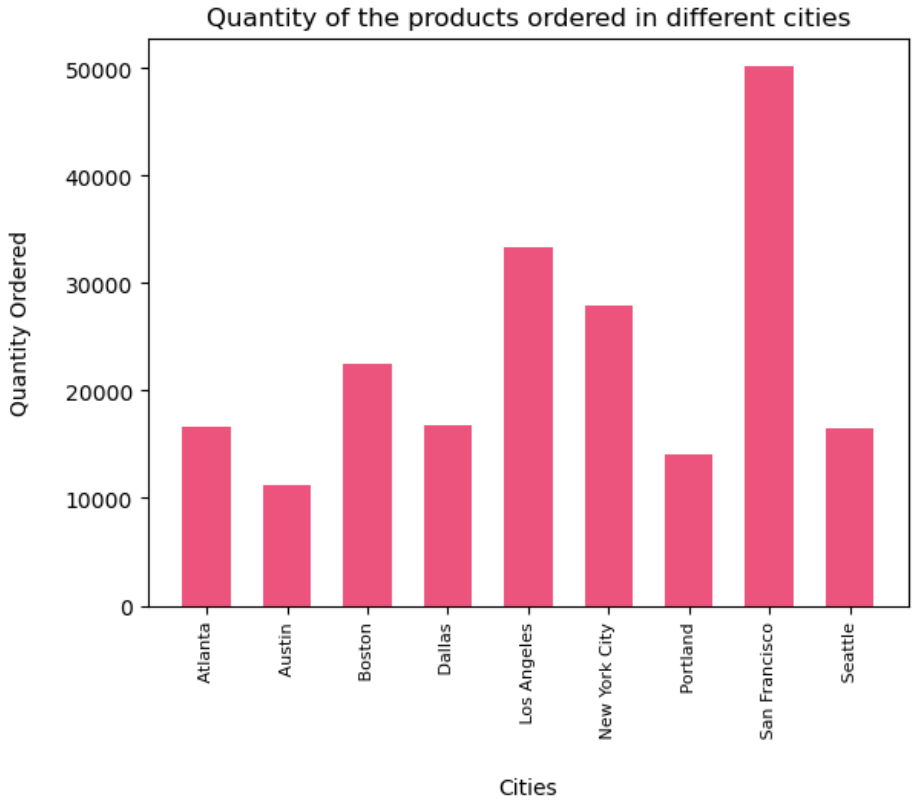

Figure 16: Quantity of products ordered in different cities

The above graph illustrates the quantity of the product ordered in different cities of USA. From the above graph, it is seen that the products are ordered the most from **San Francisco**, California, with more than 50000 orders in a year. San Francisco being the centre of commerce and finance in North California, people might be in more need of

electric products as the world is advancing rapidly towards technology. In the year 2019, the company might have focused on the sales of products in California more than other cities to increase the Californian customers. The branding might have been the best in San Francisco. Hence, the sales might have increased due to these reasons in **San Francisco**.

The bar graph is plotted using the matplotlib library of Python. Pink colour has been given to the bars in the graph, the x-axis and y-axis have been labelled as ‘Cities’ and ‘Quantity Ordered’ respectively, and the values of x-axis are taken from the list that was created beforehand which stored the name of the cities. The values of y-axis which were originally in exponential form have been converted into millions quantity ordered.

4.3 Question 3

Question: Which product was sold the most in overall? Illustrate it through bar graph.

Solution:

```
In [22]: # assigning a variable to store the sum of quantity ordered grouped by Product
products = sales_details.groupby('Product').sum()
product_sales = products['Quantity Ordered']

plt.bar(product_sales.index, product_sales.values, color="#ed548c")
plt.xticks(rotation='vertical', size=8)
plt.yticks(size=8)
plt.title("\nQuantity of the Products Ordered")
plt.xlabel('Products')
plt.ylabel('Quantity Ordered\n')
plt.show()
```

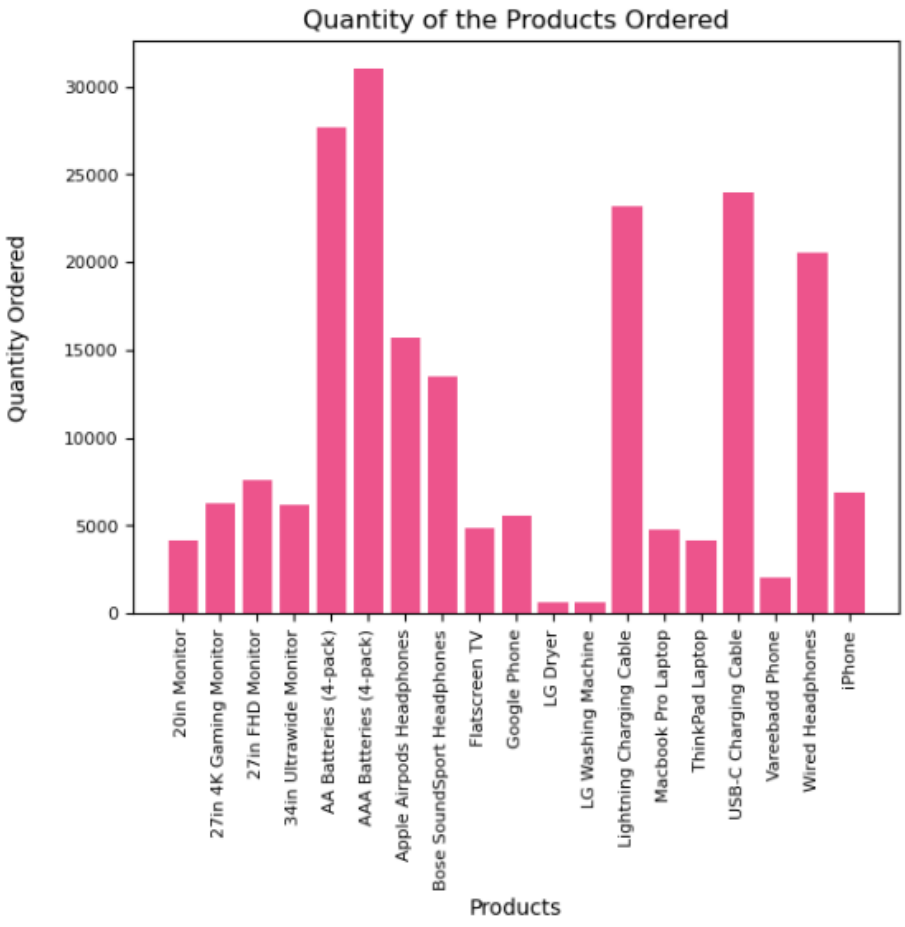

Figure 17: Quantity of the products ordered

The above bar graph demonstrates the quantity of the products ordered. As seen in the graph, **AAA Batteries(4-pack)** has been sold the most among all products. As discussed earlier in the report in the correlation portion, there is a strong negative correlation between sales and quantity ordered. If the price of the product is less, then the product will be ordered more and if the price is more, the sales will be less. Here, the AAA batteries are being sold at USD 2.99 and it is also being ordered the most around USA. So, the AAA batteries are being sold the most as they are cheaper than other products sold by the company.

The bar graph is plotted using the matplotlib library of Python. Pink colour has been given to the bars in the graph, the x-axis and y-axis have been labelled as ‘Products’ and ‘Quantity Ordered’ respectively, and the values of x-axis and y-axis are taken from the variable that was assigned before which stored the name of the products, the index value of the variable is given to x-axis and the values of the variable are given to the values of y-axis.

4.4 Question 4

Question: Write a Python program to show histogram plot of any chosen variables. Use proper labels in the graph.

Solution:

```
In [40]: # creating a histogram of the distribution of price for each product
         plt.hist(sales_details['Price Each'], edgecolor='black', bins=20, color="#ed548c")
         plt.title('Distribution of Price for each product\n')
         plt.xlabel('\nPrice')
         plt.ylabel('Count\n')
         plt.show()
```

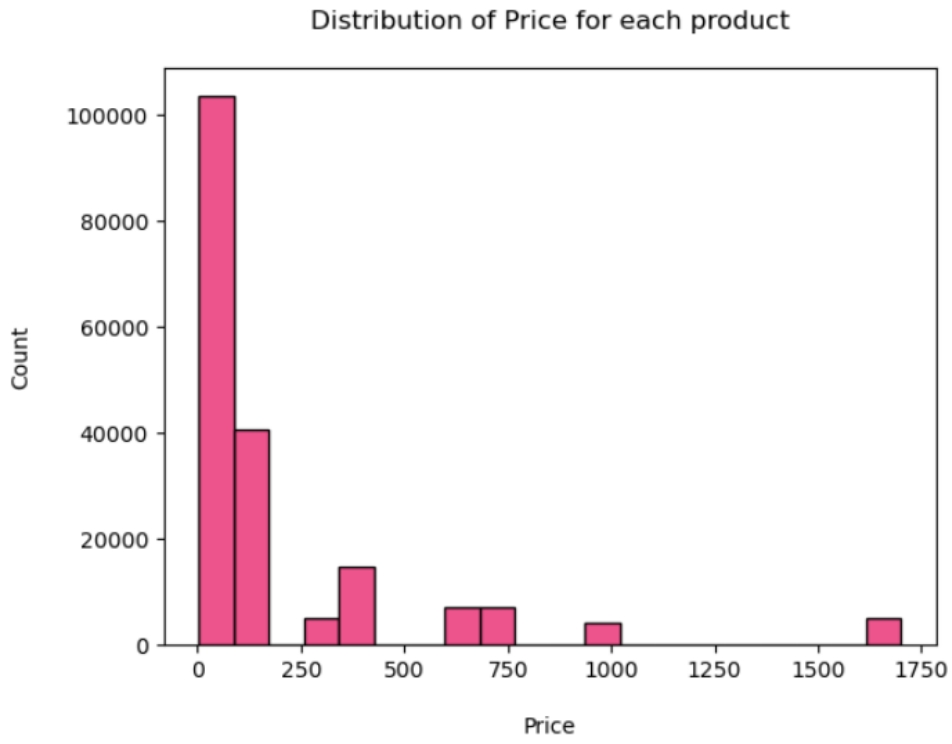

Figure 18: Histogram of price for each product

A histogram is the graphical representation of data where data is grouped into continuous number ranges and each range corresponds to a vertical bar (Cuemath, 2023). Here, a histogram is plotted for price of each product by using '.hist()' function from matplotlib library. The histogram shows that most of the product's price range from 0-250 as there is a count of almost 100k products in that range.

References

Calzon, B., 2023. *Your Modern Business Guide To Data Analysis Methods And Techniques.* [Online]
Available at: https://www.datapine.com/blog/data-analysis-methods-and-techniques/
[Accessed 3 May 2023].

Cuemath, 2023. *Histogram - Graph, Definition, Properties, Examples.* [Online]
Available at: https://www.cuemath.com/data/histograms/
[Accessed 3 May 2023].

Cuemath, 2023. *Standard Deviation.* [Online]
Available at: https://www.cuemath.com/data/standard-deviation/
[Accessed 2 May 2023].

Mcleod, S., 2023. *Correlation Definitions, Examples & Interpretation.* [Online]
Available at: https://www.simplypsychology.org/correlation.html
[Accessed 3 May 2023].

Mcleod, S., 2023. *What Is Kurtosis?* | *Definition, Examples & Formula.* [Online]
Available at: https://www.simplypsychology.org/kurtosis.html
[Accessed 3 May 2023].

Ryax Technologies, 2023. *What is the difference between Data Preparation and Data Exploration?.* [Online]
Available at: https://ryax.tech/what-is-the-difference-between-data-preparation-and-data-exploration/
[Accessed 3 May 223].

Taylor, S., 2023. *Skewness.* [Online]
Available at: https://corporatefinanceinstitute.com/resources/data-science/skewness/
[Accessed 2 May 2023].