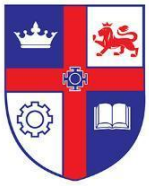
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CC5067NI-Smart Data Discovery

60% Individual Coursework

2022-23 Spring

Student Name: Kushal Bhatta
London Met ID: 21049531
College ID: np01cp4a210186
Assignment Due Date: Thursday, May 4, 2023
Assignment Submission Date: Thursday, May 4, 2023
Word Count: 1294

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 marks of zero will be awarded.

Table of Contents

Table of Tables

Table of Figures

1. Data Understanding

Data understanding is the information you have about data, the demands it will meet, its substance, and placement. There is no tool or artifact that expresses understanding of data as it is described in business glossaries, data dictionaries, models, and other types of metadata, or other locations where data information is maintained (Ladley, 2016).

The dataset below captures the information about the sales analysis of ABC company of year 2019. The data contains the attribute like Order ID, Product name, Quantity Ordered, Price of each product, Ordered Date, and purchase address of each product. Below is the description of each attribute with its data type:

S.no	Column Name	Description	Data Type
1.	Order Id	This attribute contains the order ID of each order. It is also a unique identifier of each order.	Float
2.	Product	This attribute contains the name of each product which are sold.	String
3.	Quantity Ordered	This attribute contains the quantity of products ordered in each order.	Float
4.	Price Each	This attribute contains the price of each product	Float
5.	Ordered Date	This attribute contains the date of order when it was placed.	String
6.	Purchase Address	This attribute contains the shipped address of the product.	String

Table 1 Dataset attributes Information

2. Data Preparation

2.1. Write a python program to merge data from each month into one CSV and read in updated data frame.

Merge data from each month into one CSV file, ‘os’ library was used to locate the CSV files of each month. Additionally, ‘pandas’, ‘NumPy’, ‘matplotlib.pyplot’ was imported to perform necessary calculations and visualizations.

‘os.getcwd()’ was used to locate current working directory and ‘cd’ was used to switch the directory containing CSV files. Then ‘os.listdir()’ was used to list all the files in that directory. Then, the file is confirmed.

```
[183]: import os
       import pandas as pd
       import numpy as np
       import matplotlib.pyplot as mpp
       import warnings
       warnings.filterwarnings('ignore')

[184]: os.getcwd()

[184]: 'E:\\College\\Year 2\\Sem 2\\SDD\\Coursework\\Sales Files'

[185]: cd E:\College\Year 2\Sem 2\SDD\Coursework\Sales Files
       E:\College\Year 2\Sem 2\SDD\Coursework\Sales Files

[186]: os.getcwd()

[186]: 'E:\\College\\Year 2\\Sem 2\\SDD\\Coursework\\Sales Files'

[187]: os.listdir()

[187]: ['Sales_April_2019.csv',
        'Sales_August_2019.csv',
        'Sales_December_2019.csv',
        'Sales_February_2019.csv',
        'Sales_January_2019.csv',
        'Sales_July_2019.csv',
        'Sales_June_2019.csv',
        'Sales_March_2019.csv',
        'Sales_May_2019.csv',
        'Sales_November_2019.csv',
        'Sales_October_2019.csv',
        'Sales_September_2019.csv']
```

Figure 1 Locating and Confirming Files

After that, the dataframe was created to store all the combined CSV files. All the CSV files are merged using for each loop. Inside the loop files are checked if they are CSV files or not. If it was csv file, it was read it and appended to the dataframe. And then the merged dataframe was displayed.

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

```
[188]: dataFrame =pd.DataFrame()
for each in os.listdir():
    if each.endswith('.csv'):
        dataFrame = dataFrame.append(pd.read_csv(each))
dataFrame
```

[188]:

	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
...	...	...	...	...	...	...
11681	259353.0	AAA Batteries (4-pack)	3.0	2.99	9/17/2019 20:56	840 Highland St, Los Angeles, CA 90001
11682	259354.0	iPhone	1.0	700.00	9/1/2019 16:00	216 Dogwood St, San Francisco, CA 94016
11683	259355.0	iPhone	1.0	700.00	9/23/2019 7:39	220 12th St, San Francisco, CA 94016
11684	259356.0	34in Ultrawide Monitor	1.0	379.99	9/19/2019 17:30	511 Forest St, San Francisco, CA 94016
11685	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 2 Merging csv files into one dataframe

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

First the NaN missing values are checked to see if there are NaN missing values or not using isna().Then, how many NaN values are there was checked using ' isna().sum()'.

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

```
[8]: dataFrame[dataFrame.isna().any(axis=1)]
```

[8]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1	NaN	NaN	NaN	NaN	NaN	NaN
356	NaN	NaN	NaN	NaN	NaN	NaN
519	NaN	NaN	NaN	NaN	NaN	NaN
735	NaN	NaN	NaN	NaN	NaN	NaN
1149	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
11384	NaN	NaN	NaN	NaN	NaN	NaN
11399	NaN	NaN	NaN	NaN	NaN	NaN
11468	NaN	NaN	NaN	NaN	NaN	NaN
11574	NaN	NaN	NaN	NaN	NaN	NaN
11662	NaN	NaN	NaN	NaN	NaN	NaN

900 rows × 6 columns

```
[9]: dataFrame.isna().sum()
```

```
[9]: Order ID            900
     Product             900
     Quantity Ordered    900
     Price Each          900
     Order Date          900
     Purchase Address    900
     dtype: int64
```

Figure 3 Checking number of NaN missing values.

Then, the NaN missing values are removed using '.dropna(inplace=True)' and then check if there are any NaN missing values remain using '.isna().sum()'.

```
[189]: dataFrame.dropna(inplace=True)
       dataFrame
```

[189]:

	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
...	...	...	...	...	...	...
11681	259353.0	AAA Batteries (4-pack)	3.0	2.99	9/17/2019 20:56	840 Highland St, Los Angeles, CA 90001
11682	259354.0	iPhone	1.0	700.00	9/1/2019 16:00	216 Dogwood St, San Francisco, CA 94016
11683	259355.0	iPhone	1.0	700.00	9/23/2019 7:39	220 12th St, San Francisco, CA 94016
11684	259356.0	34in Ultrawide Monitor	1.0	379.99	9/19/2019 17:30	511 Forest St, San Francisco, CA 94016
11685	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

```
[181]: dataFrame.isna().sum()
```

```
[181]: Order ID            0
       Product             0
       Quantity Ordered    0
       Price Each          0
       Order Date          0
       Purchase Address    0
       dtype: int64
```

Figure 4 Removing NaN missing values.

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

To convert Quantity Ordered and Price Each to numeric, its data type is checked if they are already numeric or not using '.dtypes '. After confirming they are not in numeric itis converted to numeric '.astype(int)'. Again using '.dtypes' it is confirmed if it was changed or not.

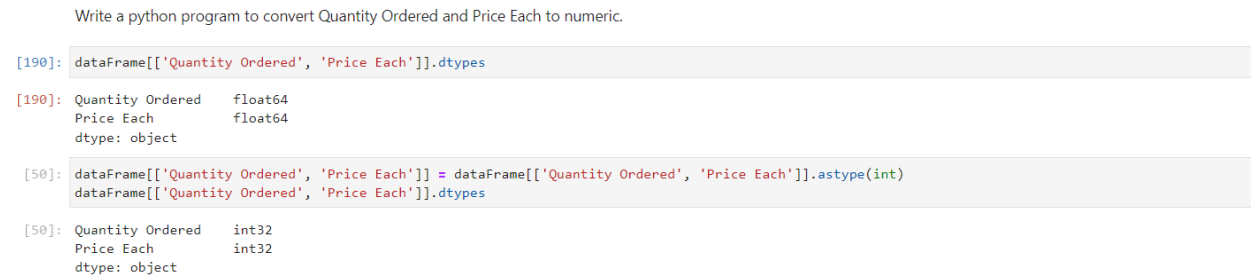


Figure 5 Converting Quantity Ordered and Price Each to numeric.

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

First the 'Order Date' column is converted to datetime data type using 'pd.to_datetime()' then the 'Month' column was created by using dataFrame['Month']. It was created by extracting the month value from the 'Order Date' column using '.dt.month' after that it is converted into integer using '.astype(int)'. Then the dataframe is displayed to show new Month column and its data type is checked using '.dtypes'.

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

```
[193]: dataFrame['Order Date']=pd.to_datetime(dataFrame['Order Date'])
       dataFrame['Month']=dataFrame['Order Date'].dt.month.astype(int)
       dataFrame
```

[193]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558.0	USB-C Charging Cable	2.0	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4
2	176559.0	Bose SoundSport Headphones	1.0	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4
3	176560.0	Google Phone	1.0	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4
4	176560.0	Wired Headphones	1.0	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4
5	176561.0	Wired Headphones	1.0	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	4
...	...	...	...	...	...	...	...
11681	259353.0	AAA Batteries (4-pack)	3.0	2.99	2019-09-17 20:56:00	840 Highland St, Los Angeles, CA 90001	9
11682	259354.0	iPhone	1.0	700.00	2019-09-01 16:00:00	216 Dogwood St, San Francisco, CA 94016	9
11683	259355.0	iPhone	1.0	700.00	2019-09-23 07:39:00	220 12th St, San Francisco, CA 94016	9
11684	259356.0	34in Ultrawide Monitor	1.0	379.99	2019-09-19 17:30:00	511 Forest St, San Francisco, CA 94016	9
11685	259357.0	USB-C Charging Cable	1.0	11.95	2019-09-30 00:18:00	250 Meadow St, San Francisco, CA 94016	9

185950 rows × 7 columns

```
[194]: dataFrame['Month'].dtypes
```

```
[194]: dtype('int32')
```

Figure 6 Creating column named Month from Ordered Date of updated dataframe and converting it to integer as data type.

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

The column ‘Purchase Address’ values was split using ‘.str.split(‘,’).str[1]’ based on the comma and index 1 values are extracted and put it in the ‘City; column.

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

```
[195]: dataFrame['City'] = dataFrame['Purchase Address'].str.split(',').str[1]
       dataFrame
```

[195]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City
0	176558.0	USB-C Charging Cable	2.0	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	Dallas
2	176559.0	Bose SoundSport Headphones	1.0	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	Boston
3	176560.0	Google Phone	1.0	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles
4	176560.0	Wired Headphones	1.0	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles
5	176561.0	Wired Headphones	1.0	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	4	Los Angeles
...	...	...	...	...	...	...	...	...
11681	259353.0	AAA Batteries (4-pack)	3.0	2.99	2019-09-17 20:56:00	840 Highland St, Los Angeles, CA 90001	9	Los Angeles
11682	259354.0	iPhone	1.0	700.00	2019-09-01 16:00:00	216 Dogwood St, San Francisco, CA 94016	9	San Francisco
11683	259355.0	iPhone	1.0	700.00	2019-09-23 07:39:00	220 12th St, San Francisco, CA 94016	9	San Francisco
11684	259356.0	34in Ultrawide Monitor	1.0	379.99	2019-09-19 17:30:00	511 Forest St, San Francisco, CA 94016	9	San Francisco
11685	259357.0	USB-C Charging Cable	1.0	11.95	2019-09-30 00:18:00	250 Meadow St, San Francisco, CA 94016	9	San Francisco

185950 rows × 8 columns

Figure 7 Creating a new column named City from Purchase Address based on the value in updated dataframe.

3. Data Analysis

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

The ‘Quantity Ordered’ is chosen as a variable. Then ‘.sum()’, ‘.mean()’, ‘.std()’, ‘.skew()’, ‘.kurtosis()’ functions are applied to ‘Quantity Ordered’ column. Then different variables are assigned to store summary statistics of sum, mean, standard deviation, skewness, and kurtosis. Then it is displayed.

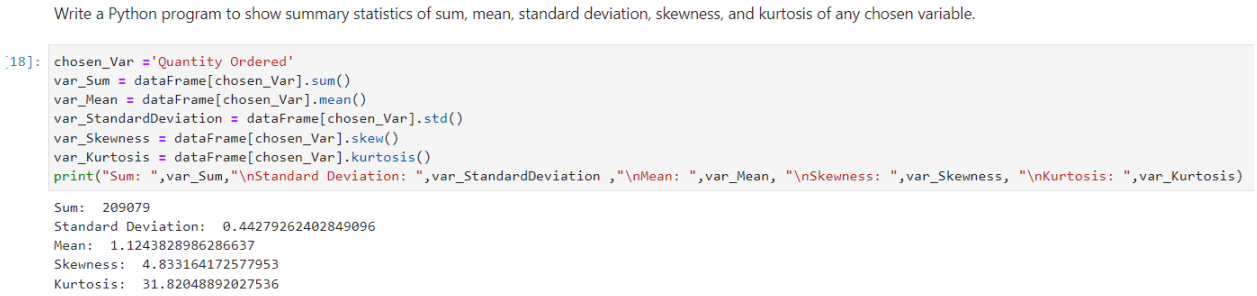


Figure 8 Showing the summary statistics of sum, mean, standard deviation, skewness, and kurtosis of Quantity Ordered.

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

To show the correlation of all the variables ‘.corr()’ is used. Each element represents the correlation between two columns. It ranges from -1(perfect negative correlation) to 1(perfect positive correlation).

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

```
[17]: dataFrame.corr()
```

[17]:

	Order ID	Quantity Ordered	Price Each	Month
Order ID	1.000000	0.000702	-0.002861	0.993063
Quantity Ordered	0.000702	1.000000	-0.148335	0.000791
Price Each	-0.002861	-0.148335	1.000000	-0.003379
Month	0.993063	0.000791	-0.003379	1.000000

Figure 9 Correlation of all variables.

4. Data Exploration

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

First, a new column ‘Sales’ is created using dataFrame[‘Sales’] which represent total sales generated by each order which is calculated by multiplying ‘Quantity Ordered’ and ‘Price Each’ column. Then it is displayed.

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

```
[18]: dataFrame['Sales']=dataFrame['Quantity Ordered']*dataFrame['Price Each']
dataFrame
```

[18]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City	Sales
0	176558.0	USB-C Charging Cable	2	11	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	Dallas	22
2	176559.0	Bose SoundSport Headphones	1	99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	Boston	99
3	176560.0	Google Phone	1	600	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles	600
4	176560.0	Wired Headphones	1	11	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles	11
5	176561.0	Wired Headphones	1	11	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	4	Los Angeles	11
...	...	...	...	...	...	...	...	...	...
11681	259353.0	AAA Batteries (4-pack)	3	2	2019-09-17 20:56:00	840 Highland St, Los Angeles, CA 90001	9	Los Angeles	6
11682	259354.0	iPhone	1	700	2019-09-01 16:00:00	216 Dogwood St, San Francisco, CA 94016	9	San Francisco	700
11683	259355.0	iPhone	1	700	2019-09-23 07:39:00	220 12th St, San Francisco, CA 94016	9	San Francisco	700
11684	259356.0	34in Ultrawide Monitor	1	379	2019-09-19 17:30:00	511 Forest St, San Francisco, CA 94016	9	San Francisco	379
11685	259357.0	USB-C Charging Cable	1	11	2019-09-30 00:18:00	250 Meadow St, San Francisco, CA 94016	9	San Francisco	11

185950 rows × 9 columns

Figure 10 Adding sales column.

Then, ‘Sales’ column is grouped according to ‘Month’ using ‘.groupby(‘Month’)[‘Sales’] and the sum of total sale of the month was get by using ‘.sum()’. And it was stored in t_Sales variable. And it was displayed.

```
[19]: t_Sales = dataFrame.groupby('Month')['Sales'].sum()
      t_Sales
```

```
[19]: Month
      1     1813956
      2     2191696
      3     2794068
      4     3374951
      5     3138287
      6     2566187
      7     2635443
      8     2234194
      9     2087435
      10    3719205
      11    3184394
      12    4591824
      Name: Sales, dtype: int32
```

```
[20]: print("The best sales were in month",t_Sales.idxmax(),"and total earnings in that month is ",t_Sales.max())
```

```
The best sales were in month 12 and total earnings in that month is  4591824
```

Figure 11 Earning of each month.

Finally, a bar chat was created using the ‘mpp.bar()’ where mpp is matplotlib library. The x-axis values are set as ‘t_Sales.index’ which contains months and y-axis values are set as ‘t_Sales.values’ which contains the total sales of months. The ‘mpp.title()’ was used to assign title, ‘mpp.xticks()’ are used to customize the tick lable. And ‘mpp.xlabel()’ and ‘mpp.ylabel()’ are used to show labels. And ‘mpp.show()’ was used to display bar graph.

```
[21]: mpp.bar(t_Sales.index,t_Sales.values)
      mpp.title("Total Sales of Each Month")
      mpp.xticks(t_Sales.index)
      mpp.xlabel("Months")
      mpp.ylabel("Sales")
      mpp.show()
```

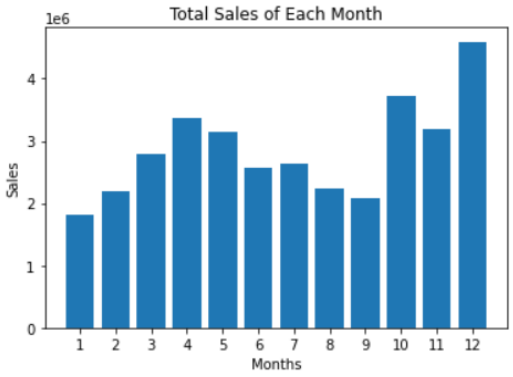


Figure 12 Bar graph showing the sales of each month.

4.2. Which city has sold the highest product?

To find which city has sold the highest product. ‘City’ column is grouped according to ‘Quantity Ordered’ using ‘.groupby(‘City’)[‘Quantity Ordered’] and ‘.sum()’ was used to add the number of product for each city and ‘.idxmax()’ is used to get the city which have sold highest product. And it was printed.

Which city has sold the highest product?

```
[61]: s_Product = dataFrame.groupby('City')['Quantity Ordered'].sum()
print(s_Product)
highest_Selling_City = s_Product.idxmax()
print("The city ",highest_Selling_City," has sold the highest product")
```

```
City
 Atlanta          16602
 Austin           11153
 Boston           22528
 Dallas           16730
 Los Angeles      33289
 New York City    27932
 Portland         14053
 San Francisco    50239
 Seattle          16553
Name: Quantity Ordered, dtype: int32
The city   San Francisco  has sold the highest product
```

Figure 13 City name with quantity of product sold.

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

To find which product was sold the most in overall. The ‘Product’ column is grouped according to ‘Quantity Ordered’ using ‘.groupby(‘Product’)[‘Quantity Ordered’]. And total “Quantity Ordered” of each product was obtained by using ‘.sum()’ and sort its values in descending order using ‘.sort_values(ascending = False). Then it was Printed.

Which product was sold the most in overall? Illustrate it through bargraph.

```
[24]: p_Sales = dataFrame.groupby('Product')['Quantity Ordered'].sum().sort_values(ascending= False)
      p_Sales
```

```
[24]: Product
      AAA Batteries (4-pack)        31017
      AA Batteries (4-pack)         27635
      USB-C Charging Cable          23975
      Lightning Charging Cable      23217
      Wired Headphones              20557
      Apple Airpods Headphones      15661
      Bose SoundSport Headphones    13457
      27in FHD Monitor               7550
      iPhone                         6849
      27in 4K Gaming Monitor         6244
      34in Ultrawide Monitor         6199
      Google Phone                   5532
      Flatscreen TV                  4819
      Macbook Pro Laptop             4728
      ThinkPad Laptop                4130
      20in Monitor                   4129
      Vareebadd Phone                2068
      LG Washing Machine              666
      LG Dryer                        646
      Name: Quantity Ordered, dtype: int32
```

```
[28]: print("The most sold product is ",p_Sales.idxmax(), " and sold quantity is ",p_Sales.max())
```

```
      The most sold product is  AAA Batteries (4-pack)  and sold quantity is  31017
```

Figure 14 Product names with sold quantity.

Finally, a bar chat was created using the ‘mpp.bar()’ where mpp is matplotlib library. The x-axis values are set as ‘p_Sales.index’ which contains product names and y-axis values are set as ‘p_Sales.values’ which contains the total product sales quantity. The ‘mpp.title()’ was used to assign title, ‘mpp.xticks()’ are used to customize the tick lable. And ‘mpp.xlabel()’ and ‘mpp.ylabel()’ are used to show labels. And ‘mpp.show()’ was used to display bar graph.

```
[29]: mpp.bar(p_Sales.index, p_Sales.values)
mpp.xticks(rotation=90)
mpp.xlabel('Products')
mpp.ylabel('Sold Product Quantity')
mpp.title('Total Sold Quantity of Each Product')
mpp.show()
```

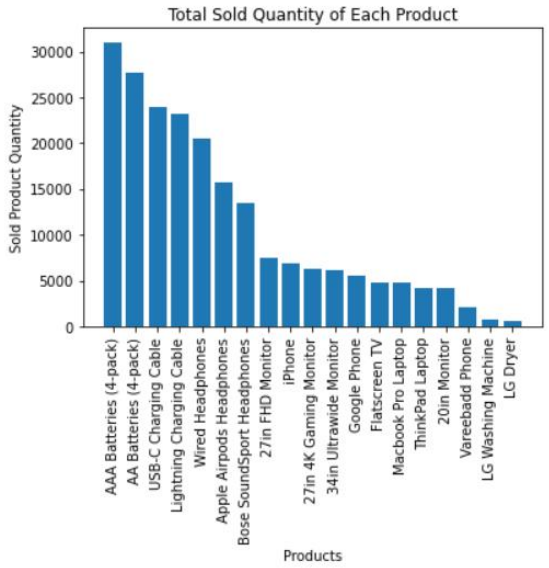


Figure 15 Bar graph showing sold quantity of each product.

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

'Product' was chosen as a variable to make histogram plot. Product and Quantity was used as labels and 'Histogram Plot for Product' was used as title. The edgecolor is used as black and bins is 20. And 'mpp.show()' was used to show the plot.

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

[67]:
```
mpp.hist(dataFrame["Product"],edgecolor='black', bins=20)
mpp.xticks(rotation=90)
mpp.xlabel('Products')
mpp.ylabel('Quantity')
mpp.title('Histogram Plot for Product')
mpp.show()
```

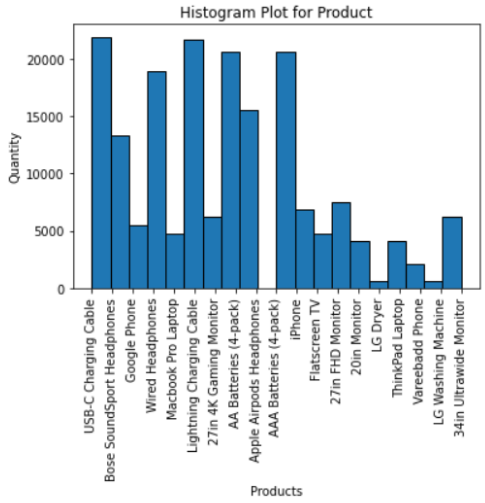


Figure 16 Histogram plot of Product.

5. References

Ladley, J., 2016. *Mastering and managing data understanding.* [Online] Available at: https://www.cio.com/article/238649/mastering-and-managing-data-understanding.html
[Accessed 1 May 2023].