LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CC5067NI-Smart Data Discovery

60% Individual Coursework

2022-23 Autumn

Student Name: Kishan Raj Malla

London Met ID: 22015698

College ID: np01cp4s220043

Assignment Due Date: Thursday, May 4, 2023

Assignment Submission Date: Thursday, May 4, 2023

Word Count: 3182

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 figure

Table of tables

1. Introduction

- The course work is about exploring the given set of data in order to extract required information or even predict the future results of a ABC company. A data set is a organize collection of database. Every column describes a particular variable and each row corresponds to a given member of the data set. the the data sets are often used in statistics to gain the insight into patterns and trends in the data (Anon., n.d.).

  This assignment involves the sales analysis from ABC company of 2019. You are expected to write Python programs and technical report on data understanding, preparation, exploration, and initial analysis.

2. Data Understanding

- The database contains the information of sales analysis of ABC company of year 2019. There are 1868950 data which represents the attributes such as Order ID, Product, Quantity Ordered, price of the product, Order Date, Purchase Address.

  The data set represents the total amount order made by the customer in in year 2019. There are hundreds of thousand of electronic product stored in the table which also represent its price where customer paid to bought it. the dataset also gives information about the total amount of quantity ordered by the customer in the exact date. The company has multiple stores in the country so there is another column name purchase address to gain the sale location of the item, these addresses are provided with its unique state name such as ST, CA and TX etc to remove the data redundancy. these The database does contains null sets which should be removed in the further preparation of the data. By further understanding the data there are all together 19 products which are sold by the company. By describing the tables, the three columns (Order ID, Quantity Ordered and Price Each) have float as datatype while other three (Product, Order Date, Purchase Address) have object as the datatype.

  The data contains hundreds of thousands purchases broken down by date, product type, cost, purchase address etc. in order to data exploration. The data in the table need to be refined in order to display data in graph. For example, those rows which contains null values need to be removed. The tables need to contains extra column such as month, total product sale to illustrate in data exploration.

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1	176558	USB-C Charging Cable	2	11.95	4/19/2019 8:46	917 1st St, Dallas, TX…
2						
3	176559	Bose SoundSport He…	1	99.99	4/7/2019 22:30	682 Chestnut St, Bost…
4	176560	Google Phone	1	600	4/12/2019 14:38	669 Spruce St, Los A…
5	176560	Wired Headphones	1	11.99	4/12/2019 14:38	669 Spruce St, Los A…
6	176561	Wired Headphones	1	11.99	4/30/2019 9:27	333 8th St, Los Angel…
7	176562	USB-C Charging Cable	1	11.95	4/29/2019 13:03	381 Wilson St, San Fr…
8	176563	Bose SoundSport He…	1	99.99	4/2/2019 7:46	668 Center St, Seattl…
9	176564	USB-C Charging Cable	1	11.95	4/12/2019 10:58	790 Ridge St, Atlanta,…
10	176565	Macbook Pro Laptop	1	1700	4/24/2019 10:38	915 Willow St, San Fr…
11	176566	Wired Headphones	1	11.99	4/8/2019 14:05	83 7th St, Boston, MA…
12	176567	Google Phone	1	600	4/18/2019 17:18	444 7th St, Los Angel…
13	176568	Lightning Charging C…	1	14.95	4/15/2019 12:18	438 Elm St, Seattle, …
14	176569	27in 4K Gaming Monitor	1	389.99	4/16/2019 19:23	657 Hill St, Dallas, TX…
15	176570	AA Batteries (4-pack)	1	3.84	4/22/2019 15:09	186 12th St, Dallas, T…
16	176571	Lightning Charging C…	1	14.95	4/19/2019 14:29	253 Johnson St, Atlan…
17	176572	Apple Airpods Headp…	1	150	4/4/2019 20:30	149 Dogwood St, Ne…
18	176573	USB-C Charging Cable	1	11.95	4/27/2019 18:41	214 Chestnut St, San …
19	176574	Google Phone	1	600	4/3/2019 19:42	20 Hill St, Los Angele…
20	176574	USB-C Charging Cable	1	11.95	4/3/2019 19:42	20 Hill St, Los Angele…
21	176575	AAA Batteries (4-pack)	1	2.99	4/27/2019 0:30	433 Hill St, New York …
22	176576	Apple Airpods Headp…	1	150	4/28/2019 11:42	771 Ridge St, Los An…
23	176577	Apple Airpods Headp…	1	150	4/4/2019 19:25	260 Spruce St, Dallas…
24	176578	Apple Airpods Headp…	1	150	4/9/2019 23:35	513 Church St, Bosto…
25	176579	AA Batteries (4-pack)	1	3.84	4/11/2019 10:23	886 Jefferson St, New…

Figure 1: CSV files

The following tables shows the detail information about each column in the dataset.

Table 1: column information

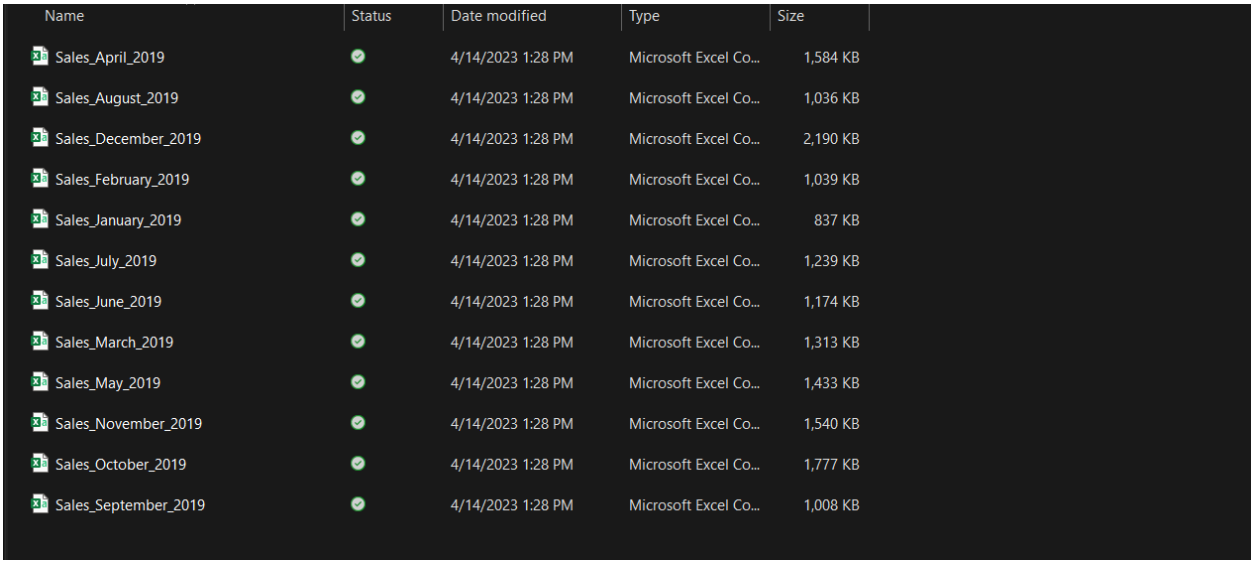
S.no	Column Name	Description	Data Type
I.	Order ID	Order Id is the unique id which represents the purchase of the product. The value in this column is numeric. This column contains null values. There are about 1886850 orders in a year.	Float
II.	Product	This column in the database represents the product that was sold to the customers. It was found that there is total 19 products in the table which were repeatedly sold.	Object
III.	Quantity Ordered	The total amount of product ordered by the customer is represented in this column. These column stores numeric data as well as null values.	Float
IV.	Price Each	This column represents the product price. The data stored in this column shows the individual price rather then displaying the total price of the products purchased.	Float
V.	Order Date	The purchase date of the product is given in this column. This column contains repeated data and also represents time of purchase	Object
VI.	Purchase Address	This column represents the address from where the products are bought.	Object

3. Data Preparation

➢

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

→ The course work contains multiple csv file which represents the monthly sales of the company ABC. These csv files should merge into one csv file to conduct yearly data analysis.



Name	Status	Date modified	Type	Size
Sales_April_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,584 KB
Sales_August_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,036 KB
Sales_December_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	2,190 KB
Sales_February_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,039 KB
Sales_January_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	837 KB
Sales_July_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,239 KB
Sales_June_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,174 KB
Sales_March_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,313 KB
Sales_May_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,433 KB
Sales_November_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,540 KB
Sales_October_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,777 KB
Sales_September_2019	✓	4/14/2023 1:28 PM	Microsoft Excel Co...	1,008 KB

Figure 2: csv files

```
# importing the module pandas, os and warning from python library
import pandas as pd
import os as os
import warnings
```

Figure 3: importing

Modules like pandas, OS and warnings are imported from the pandas library. Pandas is imported so that data frame can be used and help organize data into rows and column. Os module is imported to create file path for reading a data file. Warning module to ignore the warning.

```
merged_data=pd.DataFrame()          # creating a datagframe merged_data
warnings.filterwarnings('ignore')
#  reading all csv file in the given path in directory csv_files
csv_files=r"C:\Users\malla\OneDrive\Documents\year2 sem2\Smart data discovery\drive-download-20230414T074205Z-001"
os.chdir(csv_files)  #changing the path for current working directory from download to CSV_files directory
Fdirection=os.getcwd()   # assiging the current working directory conatainng the all csv files in variable Fdirection
```

Figure 4: changing the direction

In the above code A new dataframe is created named as "merged_data". all the csv file which located in the given path is read in the variable 'CSV_files'. 'Os.chdiv()' function is used to change the directory to the 'csv_files' which contains csv files. Getting path of directory containing csv files and Storing the path in variable 'Fdirection'.

```
for each in os.listdir(Fdirection):                    # using for loop
    if each.endswith(".csv"):                        # choosing csv files
        transferred_csv=pd.read_csv(each)                        # reads
        print(each)                                # print to ckeck if a
        merged_data=merged_data.append(transferred_csv)
merged_data
```

Figure 5: append

Here forloop is used to read all the files from the variable Fdirection. Now to extract csv file if else condition is use to filter the files of .csv extensions. Those files are stored in 'transferred_file' variable. Then files are added or append in dataframe merged_data. All the csv are now combined into one csv file and finally merged_data is written to display updated table.

So

```
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
```

[84]:

	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 6: merged_data putput

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

→ The database contains nan value which should be removed so that data exploration can be performed without any error.

```
display(merged_data)
b=merged_data.dropna()      # droping all the null value in the table
b
```

Figure 7: drop nan value code

In the above code, the updated dataframe is displayed to show the current data in the dataframe. Here the function dropna() is used to remove all the rows that

contain at least one NAN value in the merged_data. The resulting dataframe is assigned back to variable ‘b’. now the merged_data dataframe is replaced by new dataframe ‘b’.

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

[13]:

	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

Figure 8: nan output

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

→ In order to calculate the sales we would need to convert the data type of the quantity ordered and Price each into numeric.

```
[41]: b['Quantity Ordered'] = pd.to_numeric(b['Quantity Ordered']  )    # using .to_n
      b['Price Each'] = pd.to_numeric(b['Price Each'] )      #  using .to_numeric con
      b
```

Figure 9: convert result

In the above code, the .to_numeric() function is used to convert the values in the columns of ‘Quantity Ordered’ and ‘Price Each’ from ___ to ____ . this new column with numeri data is then assigned back to the ‘Quantity Ordered’ column of the dataframe b.

```
[43]: b.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 11685
Data columns (total 7 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 
 6   city              185950 non-null  object 
dtypes: float64(3), object(4)
memory usage: 11.3+ MB
```

Figure 10: convertt result

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

→ In order calculate the exact month in which the product is ordered and extract other information according to the month evaluation. A new column month is added from the ordered date column.

```
[45]:
b['month'] = pd.DatetimeIndex(b['Order Date']).month     # adding the month value of orderdate in month
b['month'] = b['month'].astype('int')            # converting datatype as integer
b
```

Figure 11: Month

A new column is added called ”month” in the dataframe b which contains the data where pd.to_datetimeIndex function converts the ‘Order Date’ into a pandas DatetimeIndex. now the month value is extracted from column the ‘Order Date’ column by the help of the attribute .month().

Here, the month datatype is converted into integer with the method month and again the new data is again saved to the month column of the dataframe.

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

```
[45]:
b['month'] = pd.DatetimeIndex(b['Order Date']).month     # adding the month value of orderdate in month
b['month'] = b['month'].astype('int')            # converting datatype as integer
b
```

[45]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	city	month
0	176558.0	USB-C Charging Cable	2.0	11.95	4/19/2019 8:46	917 1st St, Dallas, TX 75001	DallasTX	4
2	176559.0	Bose SoundSport Headphones	1.0	99.99	4/7/2019 22:30	682 Chestnut St, Boston, MA 02215	BostonMA	4
3	176560.0	Google Phone	1.0	600.00	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	Los AngelesCA	4
4	176560.0	Wired Headphones	1.0	11.99	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	Los AngelesCA	4
5	176561.0	Wired Headphones	1.0	11.99	4/30/2019 9:27	333 8th St, Los Angeles, CA 90001	Los AngelesCA	4
...	...	...	...	...	...	...	...	...
11681	259353.0	AAA Batteries (4-pack)	3.0	2.99	9/17/2019 20:56	840 Highland St, Los Angeles, CA 90001	Los AngelesCA	9
11682	259354.0	iPhone	1.0	700.00	9/1/2019 16:00	216 Dogwood St, San Francisco, CA 94016	San FranciscoCA	9
11683	259355.0	iPhone	1.0	700.00	9/23/2019 7:39	220 12th St, San Francisco, CA 94016	San FranciscoCA	9
11684	259356.0	34in Ultrawide Monitor	1.0	379.99	9/19/2019 17:30	511 Forest St, San Francisco, CA 94016	San FranciscoCA	9
11685	259357.0	USB-C Charging Cable	1.0	11.95	9/30/2019 0:18	250 Meadow St, San Francisco, CA 94016	San FranciscoCA	9

185950 rows × 8 columns

Figure 12: month output

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

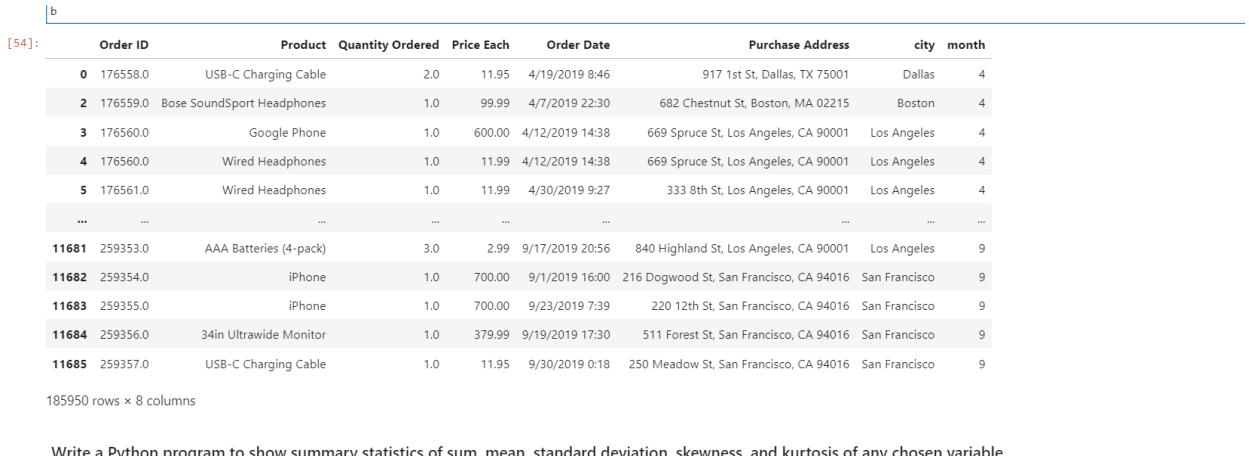
→ In order extract or analyse which city has sold the most product, a new column city needs to be created from the value of purchase address. Since the purchase address have city, state sign and state number which cannot be used to extract data.

```
[52]:
# adding the value in the city
b['city'] = b['Purchase Address'].apply(lambda x: x.split(',')[1])   # apply used to split the value in purchase address where there is ','
b
```

Figure 13: city

Here, a lambda function is applied on the column(‘Purchase Address”) of the dataframe ‘b’ here. The lambda function splits each string in the purchase address where there is comma and return the second value of the resulting list. Here x passed as the argument for the each row of the purchase address. Here

.apply method is used for returning the new series which contains the second element of each string in the “purchase address column”) .



b

[54]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	city	month
0	176558.0	USB-C Charging Cable	2.0	11.95	4/19/2019 8:46	917 1st St, Dallas, TX 75001	Dallas	4
2	176559.0	Bose SoundSport Headphones	1.0	99.99	4/7/2019 22:30	682 Chestnut St, Boston, MA 02215	Boston	4
3	176560.0	Google Phone	1.0	600.00	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	Los Angeles	4
4	176560.0	Wired Headphones	1.0	11.99	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	Los Angeles	4
5	176561.0	Wired Headphones	1.0	11.99	4/30/2019 9:27	333 8th St, Los Angeles, CA 90001	Los Angeles	4
...	...	...	...	...	...	...	...	...
11681	259353.0	AAA Batteries (4-pack)	3.0	2.99	9/17/2019 20:56	840 Highland St, Los Angeles, CA 90001	Los Angeles	9
11682	259354.0	iPhone	1.0	700.00	9/1/2019 16:00	216 Dogwood St, San Francisco, CA 94016	San Francisco	9
11683	259355.0	iPhone	1.0	700.00	9/23/2019 7:39	220 12th St, San Francisco, CA 94016	San Francisco	9
11684	259356.0	34in Ultrawide Monitor	1.0	379.99	9/19/2019 17:30	511 Forest St, San Francisco, CA 94016	San Francisco	9
11685	259357.0	USB-C Charging Cable	1.0	11.95	9/30/2019 0:18	250 Meadow St, San Francisco, CA 94016	San Francisco	9

185950 rows × 8 columns

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

Figure 14: city result

The city column is formed at the right side of the table.

4. Data Analysis

➢

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

→ All the data after refining can be proceed into data analysis stage.

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

[ ]:

47]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import skew, kurtosis    #importing required modules from the python library
```

Figure 15: import data for sum()

Numpy module is imported from the python library as np. Here numpy imported to use the function like .sum(), .mean(), .std(), .array() to analyse the statistics of the chosen variable. Scipy.stats module is imported in the code to use the function like skew and kurtosis in order to calculate the asymmetry in the distribution as well as peakedness of the distribution.

```
datA=np.array(b['Quantity Ordered'])          # adding value in array datA from
sum_A= np.sum(datA)                          # calculating the sum
mean_A= np.mean(datA)                       # calculating the mean
standard_A= np.std(datA)                    # calculating the standard deviation
skew_A= skew(datA)                          # measuring the assemetry in the distr
kurtosis_A= kurtosis(datA)                  # measuring the peakedness of the dist
print("sum of Quantity Ordered : ", sum_A)
print("mean of Quantity Ordered: ", mean_A)
print("std of Quantity Ordered: ", standard_A)
print("skew of Quantity Ordered: ", skew_A)
print("kortosis of Quantity Ordered: ", kurtosis_A)
```

Figure 16: sum

The column name ‘Quantity Ordered’ od the dataframe b is converted into array and the new data are in the variable ‘datA’. This variable is used in the further process to calculate the sum and standard Deviation of the column.

```
•[60]: import numpy as np
       from scipy.stats import skew, kurtosis    #importing required modules from the python library

       datA=np.array(b['Quantity Ordered'])         # adding value in array datA from data extracted from dataframe b
       sum_A= np.sum(datA)                        # calculating the sum
       mean_A= np.mean(datA)                     # calculating the mean
       standard_A= np.std(datA)                  # calculating the standard deviation
       skew_A= skew(datA)                        # measuring the assemetry in the distribution
       kurtosis_A= kurtosis(datA)                # measuring the peakedness of the distribution
       print("sum of Quantity Ordered : ", sum_A)
       print("mean of Quantity Ordered: ", mean_A)
       print("std of Quantity Ordered: ", standard_A)
       print("skew of Quantity Ordered: ", skew_A)
       print("kortosis of Quantity Ordered: ", kurtosis_A)

       sum of Quantity Ordered :  209079.0
       mean of Quantity Ordered:  1.1243828986286637
       std of Quantity Ordered:  0.44279143340425636
       skew of Quantity Ordered:  4.833125184879626
       kortosis of Quantity Ordered:  31.81960104066001
```

Figure 17: sum relation

Array(), one of the function of the numpy is used to convert the column ’Quantity Ordered’ of the dataframe b to a numpy array. Now it is very important to convert the column in array as it made the calculation of the sum mean, standard deviation much easier. And numpy has more advantage on operating on mathematical calculations because of the many optimized functions.

.sum() is the functions of numpy module that allowed to calculate the sum of the all value of the ‘Quantity Ordered’ column. The total amount of product ordered by the customer In the year 2019 is about 209079.

.mean() is the another function in numpy which allowed to calculate the mean of ‘Quantity Ordered. This value shows the amount of product order by the single customers.

The .std() is also one of the function of the numpy module that allows to calculate the standard deviation. The standard deviation of the quantity ordered

is 0.44279 which means the quantity ordered may differ from the mean. Here 0.44279 indicates that the individual quantity ordered are spread over the given range of approximately plus or minus.

The skewness and kurtosis is the function of scipy.stats. all the values were printed accordingly. the values of the skewness Is 4.833125184879626 which means that the data in the column have higher values than the lower values in the distribution.

Similarly kurtosis has a value of 31.81960104066001 which shows that the distribution is more peaked than a normal distribution.

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

→ Corelation is the statistical measure of the relation between the two variables. If the relation between the variables is positive then it means if one of the variable increases then other variable also increases. While negative denotes that if one variable increase then other variable decreases.

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

```
[48]: import seaborn as sns
corelation = b.corr()                # showing the relation between the orderid, quantu=ity ordered and price each
plt.imshow(corelation)
print( corelation)
sns.heatmap(corelation)
```

Figure 18: corelation

In the above code seaborn module is imported from the python library to a scatter plot. In order to show the corelation between all variables the function .corr() is used to show the corelation between all the variables in the dataframe b and the updated data is stored in the variable ‘corelation’. The third line helps

to display the corelation matrix using imshow() function of the matplotlib module. The print is used to print the corelation matrix.

	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

[71]: <AxesSubplot:>

Figure 19: correlation output

According to the output, the correlation between the columns is nearly 0 which means these columns have very weak relation with each other. So the value remains unchanged even if one of the column data increases.

While the sns.heatmapp() function used to display the heatmaps.

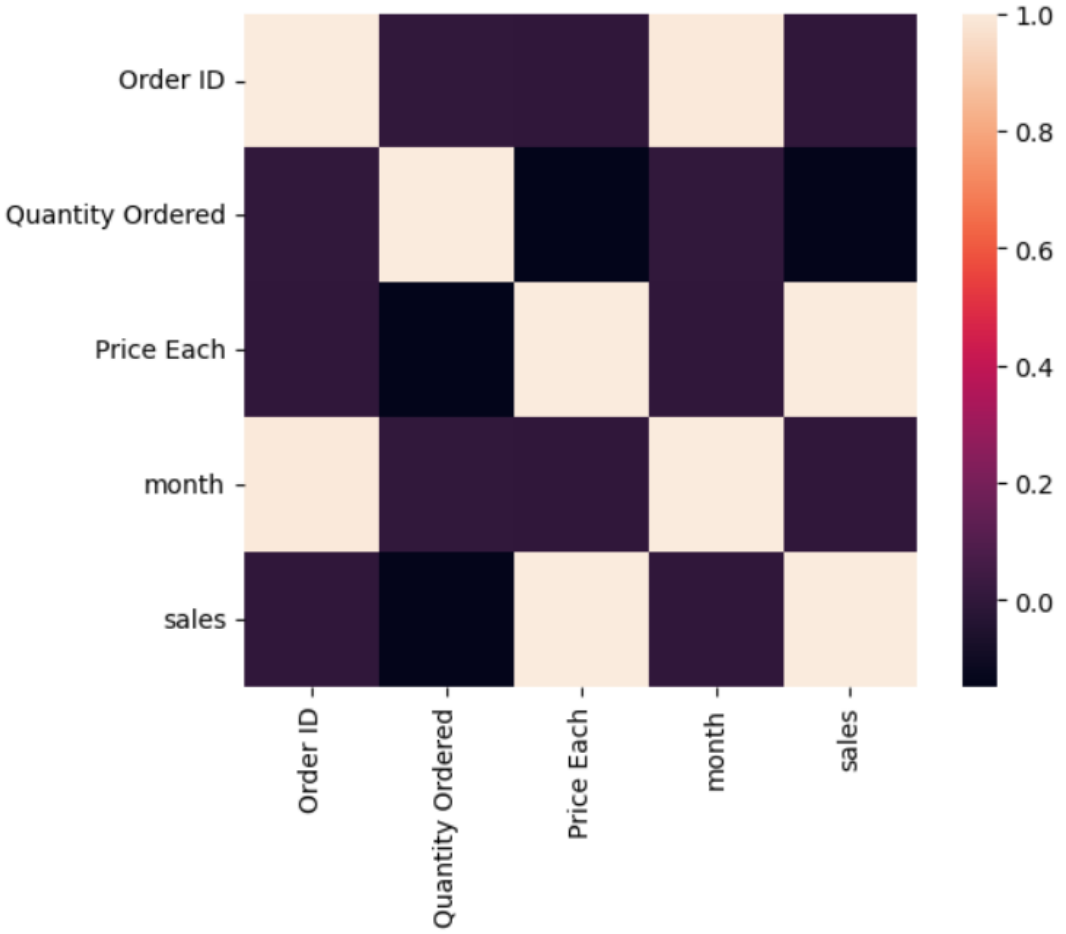


Figure 20: heatmap

The heatmap shows that the relation between the two columns in positive. According to the diagram, those relation which have the positive relations are denoted by the white color(which means the correlation is 1) and purple colors denotes the positive value(which means the correlation is near to 0) while the black color denotes the negative relation between the column(which means the corelation between them is less then 0) for example Between the Quantity Ordered and order ID the box is filled with purple color which is true since the correlation between the them is 0.000702 which nearly equal to zero. And between the “quantity Ordered and sales the box is filled with black color which denotes the negative relation which is true given the relation(-0.139417)

5. Data Exploration

➢

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

→ In order to calculate the total sales in a single month and to compare those sales we need to add a new columns month and sales. In the above data month column is already added. now a sale column must be added to calculate and the sale in the month. The following is the code written in the jupternoebook for the above question.

```
Import matplotlib.pyplot as plt
b[‘sales’] = b[‘Quantity Ordered’] * b[‘Price Each’]
```

Step1: a new sale column is created in dataframe ‘b’ which carries the data of the multiplications of the two column ‘Quantity Ordered’ and ‘Price Each’.

```
amount = b.groupby('month')['sales'].sum()
```

Step 2: New variable is defined that contains the list of data, the ‘amount’ variable contains the information created by grouping the data of the dataframe ‘b’ where .sum() function is applied on the ‘sales’ column and month. The resulting ‘amount’ variable has the ‘month’ as its index and ‘sales’ as value for each month.

```
plt.bar(amount.index, amount.values, color="green", alpha=0.3, edgecolor='black')

plt.xticks(range(1,13),["Jan","Feb","Mar","Apr","May","jun","Jul","Aug","Sep","Oct","Nov","Dec"])

plt.yticks([0, 1000000, 2000000, 3000000, 4000000] ,['0', '1M', '2M', '3M', '4M'])

plt.title(" monthly sales")

plt.xlabel("Month")

plt.ylabel("Amount")

plt.show()
```

Step 3: here ‘plt.bar’ function used to plot a bar graph to display the amount of sales in the single month. Here the amount.index is the month and amount.values is sales column . the colour of the bar is set to be ‘green’ and the distance between the bar is ‘0.3’ and border line is black. The xticks is the function that helps to change the index name to be the numbers 1 through 12 corresponding to the months of the year. Similarly the yticks is used for seting the sales value. The name of the x-axis is “Month” and name of the y-axis is “Amount”.

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

```
[41]: import matplotlib.pyplot as plt
b['sales'] = b['Quantity Ordered'] *b['Price Each']     # adding the value which is the multiplication of two column in sales

amount = b.groupby('month')['sales'].sum()                  # group by month and adding sales
plt.bar(amount.index,amount.values,color="green",alpha=0.3, edgecolor='black')
plt.xticks(range(1,13),["Jan","Feb","Mar","Apr","May","jun","Jul","Aug","Sep","Oct","Nov","Dec"])
plt.yticks([0, 1000000, 2000000, 3000000, 4000000,5000000] ,['0', '1M', '2M', '3M', '4M','5M'])
plt.title(" monthly sales")
plt.xlabel("Month")
plt.ylabel("Amount")
plt.show()
```

Figure 21: month sale

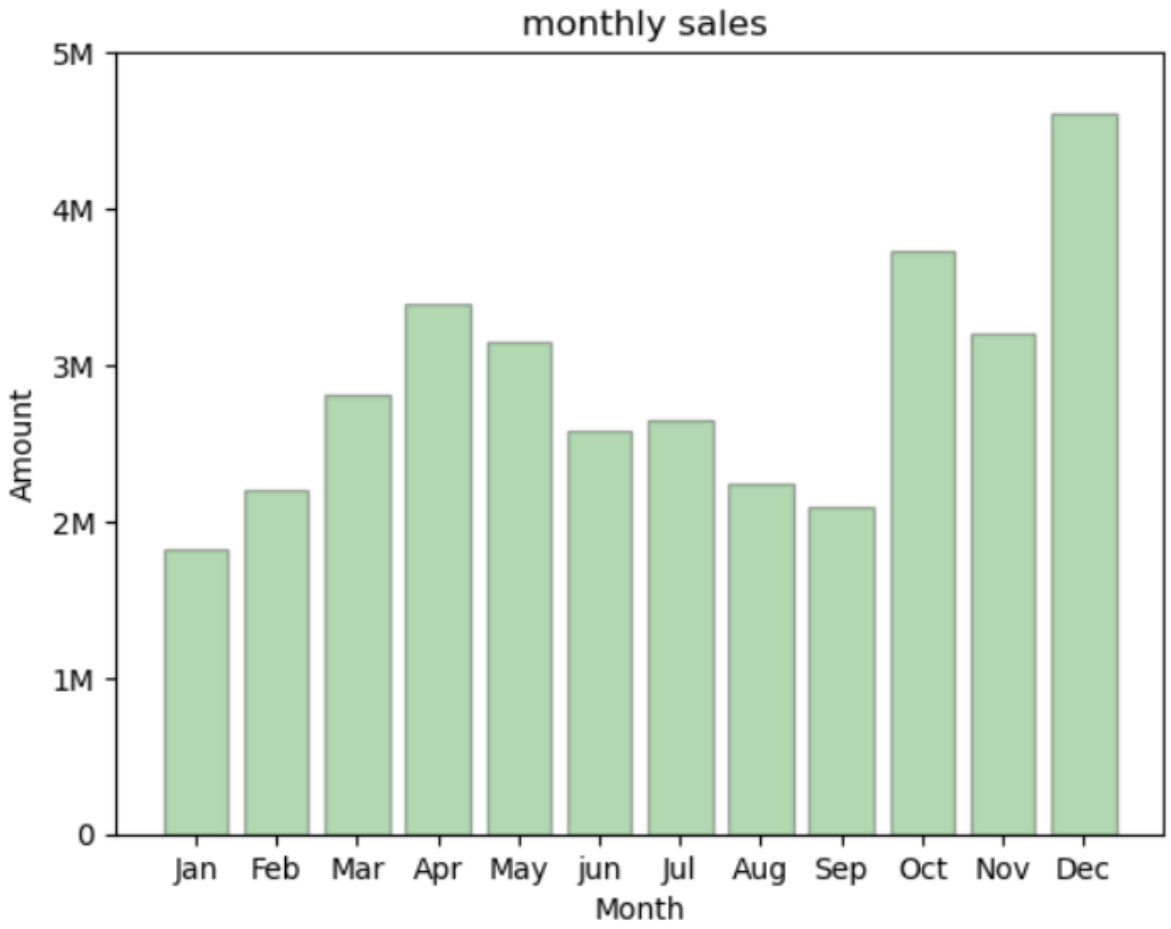


Figure 22: month sale output

It is seen that the December month has the highest amount of sale in a whole year. The sale in the December is 4613443.34. these can be observed thourgh the given code.

```
[40]: b.groupby("month")['sales'].sum()    # to see the total sales in month

[40]: month
      1     1822256.73
      2     2202022.42
      3     2807100.38
      4     3390670.24
      5     3152606.75
      6     2577802.26
      7     2647775.76
      8     2244467.88
      9     2097560.13
      10    3736726.88
      11    3199603.20
      12    4613443.34
      Name: sales, dtype: float64
```

Figure 23: sales in the month

The data shows the total sales in a single month.

5.2. Which city has sold the highest product?

→ For this question a new column needs to be created called ac ity which has already created in above data preparation. The column ‘city’ and ‘Quantity Ordered’ column is used to extract the highest amount of product sell in the city.

```
max_sale = b.groupby("city")["Quantity Ordered"].sum()
```

Step 1: In the above code the dataframe ‘b’ is grouped by the column ‘month’ and a ‘.sum()’ function is applied to the ‘Quanitity Ordered’ column, which returns the total product sales in each month. The ‘max_sales’ variable contains the total product sales in each month.

```
plt.bar(max_sale.index, max_sale.values)
plt.xticks(rotation = 90)
plt.xlabel('city')
plt.ylabel('number of product saled')
plt.title("Product Sold in City")
plt.show()
```

Step 2: In the above code ‘plt.bar()’ function is used to plot a bar graph between the ‘city’ as index and ‘Quantity Ordered’ as values. The name of the of the bar is rotated into 90 Degree to clear the name of the city. The name of the X-axis is ‘city’ and the name of the y-axis is “number of product saled”. The name of the bar graph is “Product Sold in City”

Which city has sold the highest product?

```
[42]: # visualizing
      max_sale = b.groupby("city")["Quantity Ordered"].sum()
      max_sale
```

```
[42]: city
      Atlanta          16602.0
      Austin           11153.0
      Boston           22528.0
      Dallas           16730.0
      Los Angeles      33289.0
      New York City    27932.0
      Portland         14053.0
      San Francisco    50239.0
      Seattle          16553.0
      Name: Quantity Ordered, dtype: float64
```

Figure 24: product sold

The above diagram shows the product sold in a single month. The san Francisco has sold 50239.

```
[45]: max_sale = b.groupby("city")["Quantity Ordered"].sum()    # adding the value city and quantity ordered in datafram max_sale
      plt.bar(max_sale.index, max_sale.values)          #graph plot where city is the x-axis and values are the quantity ordered
      plt.xticks(rotation = 90)                 # to display the name
      plt.xlabel('city')
      plt.ylabel('number of product saled')
      plt.title("Product Sold in City")
      plt.show()                                  # to display the plot
```

Figure 25: Product sold2

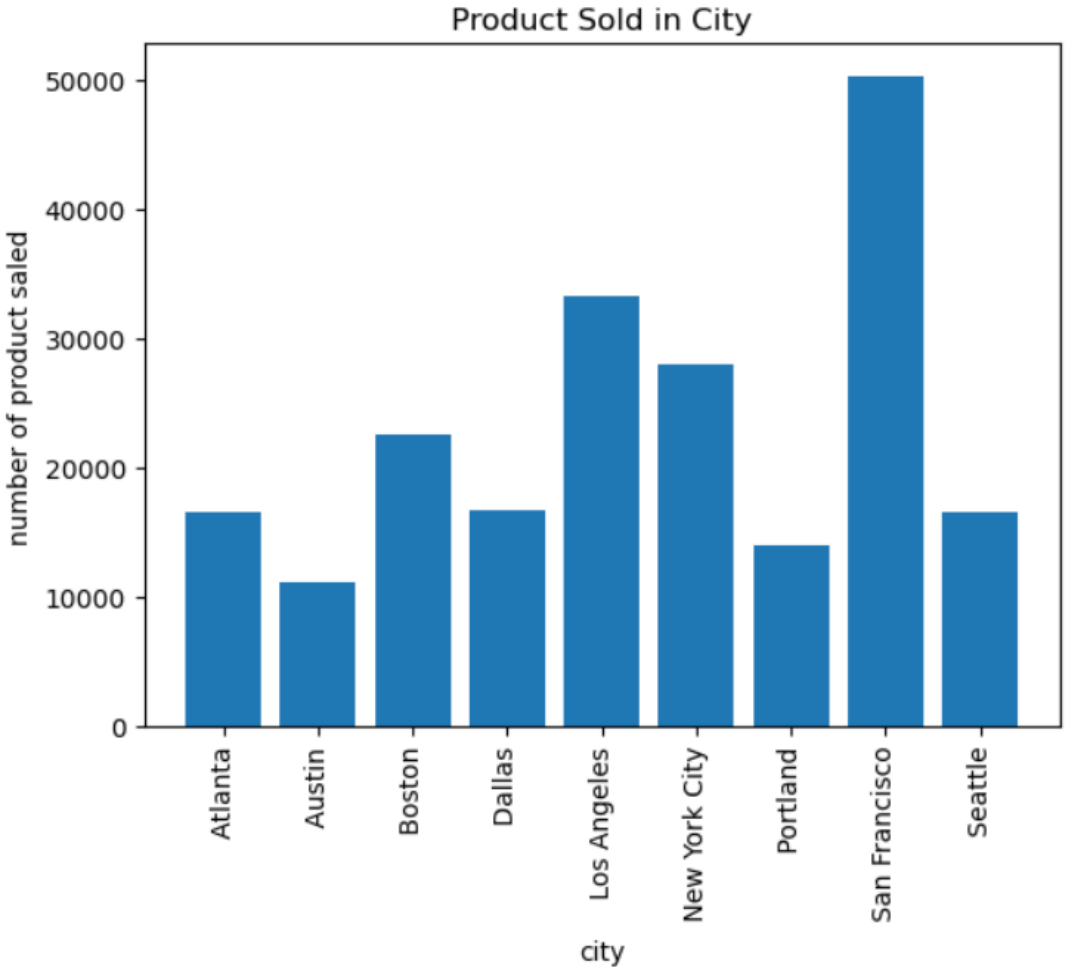


Figure 26: product sold output

The above diagram shows that san Francisco had the highest amount of product sold then most of the cities. While Austin sold the least amount of product.

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

→ In order to display the most sold product in a year. A data needs to be calculated where the quantity ordered should be sum for each product. According to the data there are about 19 products.

```
values= b.groupby("Product")["Quantity Ordered"].sum()
```

Step1: In the first step the column of the dataframe b is grouped by 'Product' column and then the sum of the 'Quantity Ordered' is assigned to each product. The The 'values' variable contains the total product ordered in each month.

```
plt.bar(values.index,yvalues.values)
```

Step2: plt.bar() function is used to plot the graph between the product and Quantity ordered here. The "Product" is considered as index and 'Quantity Ordered' as values.

```
plt.xticks(rotation=90)
```

Step3: the plt.xticks is used for rotating the name of the products to cear the names.

```
plt.xlabel("Product")
```

```
plt.ylabel("Product sold")
```

Step4: The name of the x-axis is set as "product" and the name of the y-axis is set as "product sold"

```
plt.legend()
```

```
plt.title("highest product sold")
```

```
plt.show()
```

Step5: the plt.legend() is used to denote the color of the bar graph and tht title of the whole bar graph is set as “highest product sold”

```
[37]: values= b.groupby("Product")["Quantity Ordered"].sum()
      plt.bar(values.index,yvalues.values,label="product")
      plt.xticks(rotation=90)
      plt.xlabel("Product")
      plt.ylabel("Product sold")
      plt.legend()
      plt.title("highest product sold")
      plt.show()
```

Figure 27: highest product

The code written in the anaconda ide.

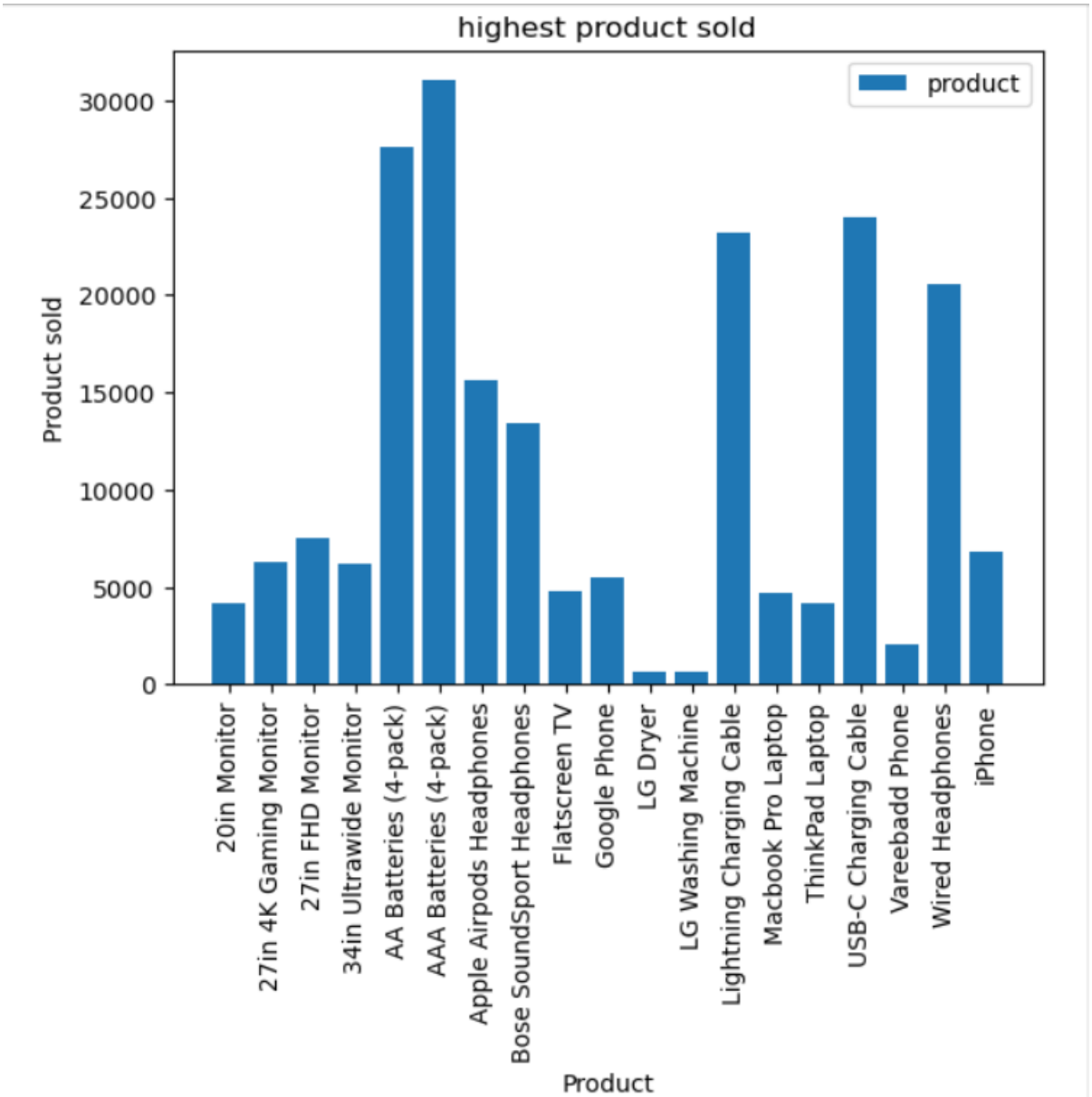


Figure 28: highest product output.

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

→ A frequency distribution shows how often each different value in a set of data occurs. A histogram is the most used graph to show frequency distributions. It looks very much like a bar chart, but there are important differences between them. This helpful data collection and analysis tool is considered one of the seven basic quality tools. (Anon., n.d.)

```
plt.hist(b['month'], bins= 11,edgecolor='black', color= "green" )
```

Step 1: here plt.hist() function of the matplotlib.pylot module is used to plot the histrogram. The ‘month’ column is chosen for the graphical representation. It has 11 bins to represent each month correctly which 12. The color of the bar is set to green. And border line is black.

```
plt.xticks(np.arange(1,13),["January","february","March","April","May","June","
July","August","September","October","November","December"], rotation=90 )
```

Step 2: in order to display all the month x-ticks labels of the plot to be the number 1 through 12 corresponding to the months of the year and setting the x-ticks to be located with each month and g with their respective names.

```
plt.xlabel("Months")
plt.ylabel("number of orders")
plt.show()
```

Step3: the name of the x-axis is set as Months. The name of the y-axis is set to number of orders.

```
[69]:
plt.hist(b['month'], bins= 11,edgecolor='black', color= "green" )       # bin used for managing number of bars
plt.xticks(np.arange(1,13),["January","february","March","April","May","June","July","August","September","October","November","December"], rotation=90 )
plt.xlabel("Months")
plt.ylabel("number of orders")
plt.show()
```

Figure 29: histogram code

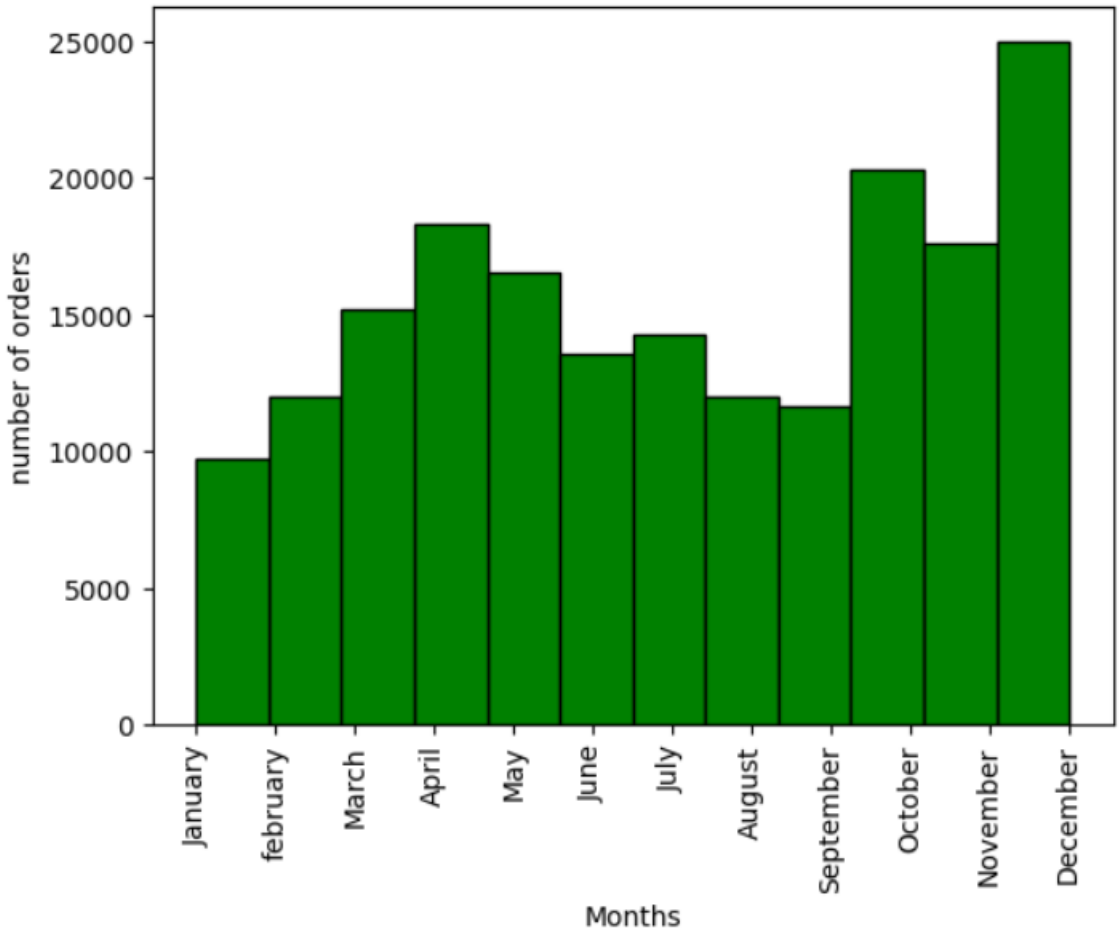


Figure 30:histogram output

From the above output, it is seen that the highest amount of product order is from in December which means the frequency of the December is high because December s mostly repeated in the column.

6. Conclusion

- In conclusion using a python programming language to prepare, explore and analysing the data proved to be very effective in carrying the various task like analysing the total sale in a month and total product sold in a last year. Exploring which product was sold the most. All these analyses are effective for a data analysist. Overall, this assignment provided knowledge to apply to various data analysis techniques and tools to real world sales data.

7. Bibliography

Anon., n.d. *asq.org.* [Online]
Available at: https://asq.org/quality-resources/histogram

Anon., n.d. *wikipedia.* [Online]
Available at: https://en.wikipedia.org/wiki/Data_set