LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Smart Data Discovery

60% Individual Coursework

2022-23 Spring

Student Name: Arbaaz Alam

London Met ID: 22015655

College ID: NP01CP4S220108

Assignment Due Date: Thursday, May 4, 2023

Assignment Submission Date: Thursday, May 4, 2023

Word Count: 2930

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.

Acknowledgement

I am grateful to “Islington college” for giving us an amazing opportunity to explore about Python libraries like Pandas, NumPy and Matplotlib. I would like to express gratitude to all the tutor and module leader for their noble guidance throughout the module. I am thankful to our module leader Mr. Dipeshor Silwal sir, for wonderful teaching methods in lecture and grateful to our tutors Mr. Sarun Dahal sir, who guided me throughout the module with keen supervision and suggestion for this coursework. I would also like to express my thankfulness to my friends and my family who supported me directly or indirectly to complete this coursework.

Abstract

This project presents the use of various python libraries like Pandas, Operating System, NumPy, Matplotlib and Seaborn. Using all these libraries and python functions, data analysis is carried out. The data is of ABC company which sells electronic product in the USA. There are CSV file of each month with 6 columns which have data of sales of product. The report is divided into five heading. Data understanding heading contain information about data columns and what kinds of column are there in CSV file. Data preparation heading contains how all the CSV file are merge, how data can be converted to integer, how new columns are created and null values are removed. Data analysis heading contains how coefficient of correlation is studied using heatmap and various statistic values are calculated. Data exploration heading contains graphs, pie charts, histogram of the sales of product. Finding heading contains overall finding of the data.

Table of Contents

List of Figures

List of Tables

1. Data Understanding

The data contains the information of sales analysis of ABC company of year 2019. It includes the attributes such as Order ID, Product, Quantity Ordered, Price Each, Order Date, and Purchased Address. According to the dataset, this data is of an electronic store which sells various products including phones, TV's, laptop, monitors, batteries, headphone, dryer and charging cables. This store is located in USA and is sold in 9 cities of USA.

S.N.	Column Name	Description	Data Type
1	Order ID	Order ID is unique id given to each order that takes place.	float
2	Product	Product are the types good that are company sells.	string
3	Quantity Ordered	Quantity Ordered is the total number of same products ordered at a time.	float
4	Price Each	Price Each is the price of each product.	float
5	Order Date	Order Date is the date that the product was ordered.	string
6	Purchased Address	Purchased Address is the address that the product was ordered.	string

Table 1 Data information table

```
In [4]: df.dtypes

Out[4]: Order ID            float64
        Product              object
        Quantity Ordered    float64
        Price Each          float64
        Order Date           object
        Purchase Address     object
        dtype: object
```

Figure 1 Datatype of columns

The above image shows the data type of all the columns after reading it in data frame.

```
In [66]: df.describe()#describe function show all the statistic of numeric data
```

Out[66]:

	Order ID	Quantity Ordered	Price Each
count	185950.000000	185950.000000	185950.000000
mean	230417.569379	1.124383	184.399735
std	51512.737110	0.442793	332.731330
min	141234.000000	1.000000	2.990000
25%	185831.250000	1.000000	11.950000
50%	230367.500000	1.000000	14.950000
75%	275035.750000	1.000000	150.000000
max	319670.000000	9.000000	1700.000000

Figure 2 Dataframe basic statistical

In the above figure using describe function the mean, standard deviation, median quartile, max and min value was extracted from the dataframe. The data shows that the maximum price product is of 1700 dollars where as minimum price product is of 2.99 dollars. The mean or average of quantity ordered is 1.124383.

1.1. Order ID

Order ID is a column that contains unique id, each order placed is given different id which helps to extract data on the basic of order. There are 186850 order id in total. Order ID is discrete quantitative variable as it gives value of only one order and is unique and it is countable, finite number of values.

1.2. Product

Product is a column that contains name of the product that has been ordered. There are 19 products in total in the dataset including phones, TV's, laptop, monitors, batteries, headphone, dryer and charging cables.

```
In [8]: df1=df['Product'].unique()#Using unique function all the name of product are extracted
        df2=pd.DataFrame(df1)#array is converted into dataframe
        df2.columns =['Products']#column name is give to dataframe
        df2
```

Out[8]:

	Products
0	USB-C Charging Cable
1	Bose SoundSport Headphones
2	Google Phone
3	Wired Headphones
4	Macbook Pro Laptop
5	Lightning Charging Cable
6	27in 4K Gaming Monitor
7	AA Batteries (4-pack)
8	Apple Airpods Headphones
9	AAA Batteries (4-pack)
10	iPhone
11	Flatscreen TV
12	27in FHD Monitor
13	20in Monitor
14	LG Dryer
15	ThinkPad Laptop
16	Vareebadd Phone
17	LG Washing Machine
18	34in Ultrawide Monitor

Figure 3 Unique product

The above image shows all the product that is sold by the company. The unique function gives all the unique data array which is converted to dataframe and column name is given.

1.3. Quantity Ordered

Quantity Ordered is the column that contain the amount of product that has been ordered. The highest amount of quantity ordered is 9 whereas least amount of quantity order is 1. Quantity Ordered is discrete quantitative variable.

```
In [9]: df1=df['Quantity Ordered'].unique()#Using unique function all the name of quantity Ordered are extracted
        df2=pd.DataFrame(df1)#array is converted into dataframe
        df2.columns =['Quantity Ordered']#column name is give to dataframe
        df2
```

Out[9]:

	Quantity Ordered
0	2.0
1	1.0
2	3.0
3	5.0
4	4.0
5	7.0
6	6.0
7	8.0
8	9.0

Figure 4 Quantity ordered values

The above image shows the all the unique quantity ordered in the dataframe using unique function.

```
In [67]: df1=df.groupby('Product').sum()#Using groupby function by product all the other numeric data are sum
         df2=df1['Quantity Ordered']#only sum of quantity ordered is extracted by prodct
         df3=pd.DataFrame(df2)#array is converted into dataframe
         df3
```

Out[67]:

Product	Quantity Ordered
20in Monitor	4129.0
27in 4K Gaming Monitor	6244.0
27in FHD Monitor	7550.0
34in Ultrawide Monitor	6199.0
AA Batteries (4-pack)	27635.0
AAA Batteries (4-pack)	31017.0
Apple Airpods Headphones	15661.0
Bose SoundSport Headphones	13457.0
Flatscreen TV	4819.0
Google Phone	5532.0
LG Dryer	646.0
LG Washing Machine	666.0
Lightning Charging Cable	23217.0
Macbook Pro Laptop	4728.0
ThinkPad Laptop	4130.0
USB-C Charging Cable	23975.0
Vareebadd Phone	2068.0
Wired Headphones	20557.0
iPhone	6849.0

Figure 5 Product with total quantity ordered

In the above image the total quantity ordered of each product is displayed with its corresponding product. The highest ordered product is AAA Batteries (4-pack) of 31017

And least ordered product is LG Dryer with 649 orders. Using group by function by product all the data are sum and only quantity order is extracted.

1.4. Price Each

Price Each column contains the price of product that has been ordered. The highest price product is of 1700 dollar and least price product is of 2.99 dollars. Price each is continuous quantitative variable as it can be taken in any value within an interval.

```
In [63]: df1=df.groupby('Product').mean()['Price Each']#using mean the price each with its product name is extracted
         df2=pd.DataFrame(df1)#array is converted into dataframe
         df2
```

Out[63]:

Product	Price Each
20in Monitor	109.99
27in 4K Gaming Monitor	389.99
27in FHD Monitor	149.99
34in Ultrawide Monitor	379.99
AA Batteries (4-pack)	3.84
AAA Batteries (4-pack)	2.99
Apple Airpods Headphones	150.00
Bose SoundSport Headphones	99.99
Flatscreen TV	300.00
Google Phone	600.00
LG Dryer	600.00
LG Washing Machine	600.00
Lightning Charging Cable	14.95
Macbook Pro Laptop	1700.00
ThinkPad Laptop	999.99
USB-C Charging Cable	11.95
Vareebadd Phone	400.00
Wired Headphones	11.99
iPhone	700.00

Figure 6 Product with its price

In the above figure the price of each product is shown corresponding to product name. Using groupby function product unique product was extracted and mean helps to find price of the product.

1.5. Order Date

Order Date column contains the date and time the product was ordered. It contains data of the year 2019 and has all 12 months sales data.

1.6. Purchased Address

Purchased address column contains the address the product was ordered. There is total 186850 address in the dataset.

```
In [68]: df1=df['Purchase Address'].unique()#Using unique function all the unique are extracted
         df2=pd.DataFrame(df1)#array is converted into dataframe
         df2.columns =['Unique Purchased Address']#column name is give to dataframe
         df2
```

Out[68]:

	Unique Purchased Address
0	917 1st St, Dallas, TX 75001
1	NaN
2	682 Chestnut St, Boston, MA 02215
3	669 Spruce St, Los Angeles, CA 90001
4	333 8th St, Los Angeles, CA 90001
...	...
140783	260 Spruce St, Boston, MA 02215
140784	911 River St, Dallas, TX 75001
140785	981 4th St, New York City, NY 10001
140786	840 Highland St, Los Angeles, CA 90001
140787	220 12th St, San Francisco, CA 94016

140788 rows × 1 columns

In the above figure the unique address of the purchased address is extracted so total unique address is 140788. Out of 186850 purchase address, 46062 are repeated addresses. So, 24% of order are repeated.

2. Data Preparation

Data collection, combination, structure, and organization are all steps in the process of preparing data for use in business intelligence (BI), analytics, and data visualization applications. Data profiling, cleaning, validation, and transformation are all parts of data preparation. It frequently includes entails combining data from various internal systems and outside sources (Stedman, 2022).

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

In order to complete this question, the csv files of all months must me merge and update in dataframe of ABC company. Pandas is a library of python programming language which contains various function. One of them is dataframe, dataframe is a two-dimensional data and the labels that go with them are stored in a structure (Stojiljković, 2020).

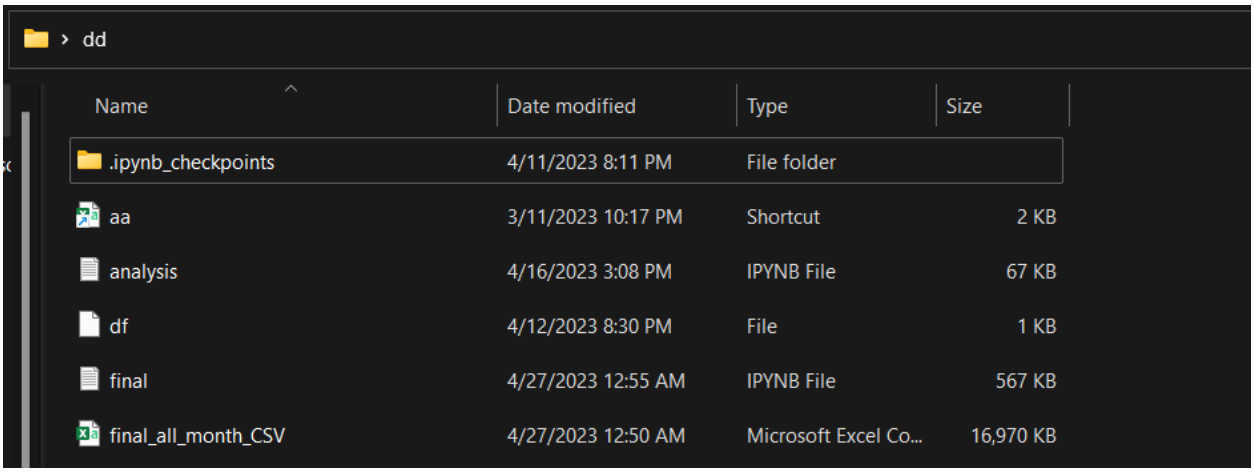
```
In [1]: #importing all the needed libraries
        import pandas as pd
        import os
        import matplotlib.pyplot as plt
        import seaborn as sns

In [2]: df=pd.DataFrame()#creating empty dataframe
        files = os.listdir("C:/Users/lucky/OneDrive/Desktop/sales")#listdir lists all the file in that directory
        for file in files:#for each loop is used to iteration
            #read_csv is used to read excel csv file and is concatenate with file
            df1 = pd.read_csv("C:/Users/lucky/OneDrive/Desktop/sales/"+file)
            """concat is used to merge, empty dataframe is merged with csv file that is store in df1,
            ignore index helps to continue the index"""
            df = pd.concat([df, df1], ignore_index=True)
        df.to_csv('final_all_month_CSV.csv', index=False)#the dataframe of all 12 month is converted to csv file
```

Figure 7 Merging data of all months into one CSV file

In the above figure before reading data, all the needed libraries were imported. First empty data frame was created by the name df. In files variable using listdir(it is function of os libraries which helps to list all the file in the directory) the file was listed giving the path where file is stored. Using for each loop files variable was iterated and was read using pandas read_csv function which helps to read csv file and was concatenated with path and file that come from for each loop. Using pandas concat function empty dataframe and the read_csv file was merge which will eventually merge all the files. Concat is a pandas function which does all the heavy lifting of performing concatenation

operations whereas ignore index helps to continue the index while files are merged (Yash_R, 2023). Using to_csv file function the file merge file was exported to computer and index false helped to remove the index column.



Name	Date modified	Type	Size
.ipynb_checkpoints	4/11/2023 8:11 PM	File folder	
aa	3/11/2023 10:17 PM	Shortcut	2 KB
analysis	4/16/2023 3:08 PM	IPYNB File	67 KB
df	4/12/2023 8:30 PM	File	1 KB
final	4/27/2023 12:55 AM	IPYNB File	567 KB
final_all_month_CSV	4/27/2023 12:50 AM	Microsoft Excel Co...	16,970 KB

Figure 8 CSV file creation to current directory

The above image shows the file was successfully imported to computer.

```
In [5]: df = pd.read_csv('final_all_month_CSV.csv')#dataframe of all 12 months
        df
```

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
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 9 Reading all month's new CSV file

The above image shows the merge file contains all the data in which there are 186850 rows and 6 columns.

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

In the dataframe there are multiple NaN values which means not a number, which appears when there in no data in certain column. In order to remove that, dropna function is used NaN values.

```
In [69]: df.isna().sum()#isna function helps to find null values in the dataframe using sum function all the null values are counted.

Out[69]: Order ID            900
         Product             900
         Quantity Ordered    900
         Price Each          900
         Order Date          900
         Purchase Address    900
         dtype: int64

In [70]: plt.figure(figsize=(10,6))#figure size is set
         sns.heatmap(df.isna().transpose(), cbar_kws={'label': 'Missing Data'})#using heat map null values are plotted

Out[70]: <AxesSubplot:>
```

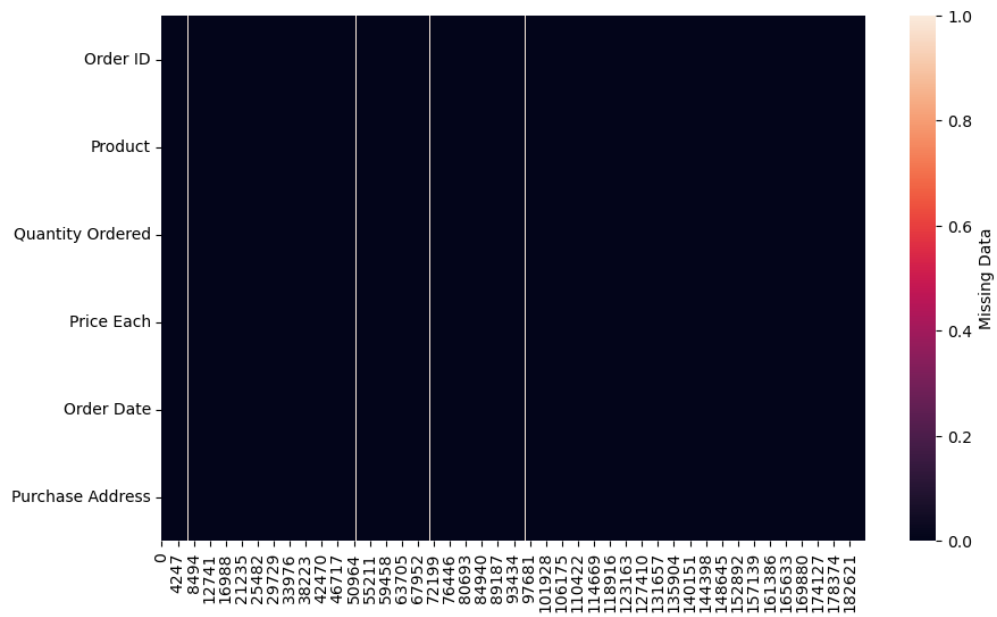


Figure 10 Null value heatmap and sum

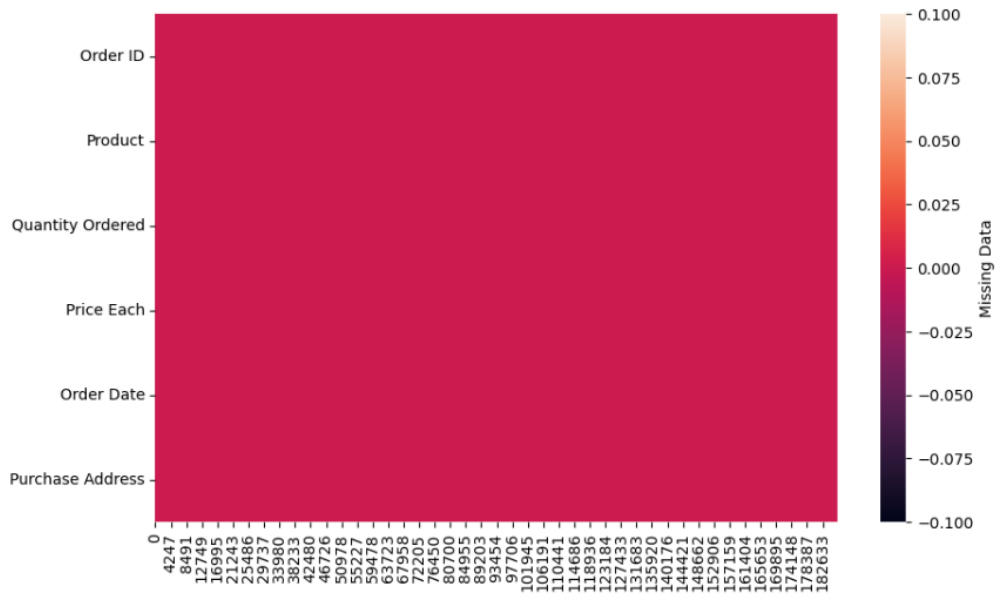
In the above image using isna() function the NaN values are sum which give the sum of NaN values in the column. Using the heatmap() function of seaborn which is also library of python the missing data is plotted. The white line shows that there are missing data whereas black background means there are no NaN data. Heatmap function helps to find two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colours (Alok, 2022).

```
In [19]: """Using dropna all the null values are removed where how helps to remove all null values and inplace true helps to permanent
         remove it in this dataframe"""
         df.dropna(how = 'all', inplace = True)
```

Figure 11 Dropna to remove NaN value

Using dropna function the NaN value are remove 'how=all' helps to remove all null values and inplace true helps to permanent remove it in this dataframe.

```
In [5]: plt.figure(figsize=(10,6))
        sns.heatmap(df.isna().transpose(), cbar_kws={'label': 'Missing Data'})#using heat map null values are plotted if there is
Out[5]: <AxesSubplot:>
```




```
In [8]: df.isna().sum()#isna function helps to find null values in the dataframe using sum function all the null values are counted.
Out[8]: Order ID            0
        Product             0
        Quantity Ordered    0
        Price Each          0
        Order Date          0
        Purchase Address    0
        dtype: int64
```

Figure 12 Heatmap and NaN value sum after removing it

In the above figure using isna() and heatmap the NaN values has been removed is shown.

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

In order to convert the Quantity Ordered and Price Each to numeric we have to use pandas' function to_numeric which is used to convert data into numeric type.

```
In [74]: #Qunatity ordered and price each columns data is changed to numeric using apply function
         df[['Quantity Ordered','Price Each']] = df[['Quantity Ordered','Price Each']].apply(pd.to_numeric)
         df
```

Out[74]:

	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 13 Conversion of Quantity Ordered and Price Each to numeric

In the above figure the Quantity Ordered and Price Each were converted to numeric type using to_numeric() function. By apply function we can apply another function in the multiple columns.

```
In [75]: df.dtypes#dtypes helps to show data type of columns

Out[75]: Order ID            float64
         Product              object
         Quantity Ordered    float64
         Price Each          float64
         Order Date           object
         Purchase Address     object
         dtype: object
```

Figure 14 Data type after conversion

In the above figure using dtypes the data type of each column is shown.

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

In order to add new column named month the Order Date data should be converted pandas date and time so that month can be simply accessed.

```
In [76]: df['Order Date']=pd.to_datetime(df['Order Date'])#it converts order date to pandas date and time
         df['Month']=df['Order Date'].dt.month.astype(int)#month column is extracted from it and covenverted to int using astype
         df
```

Out[76]:

	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
...	...	...	...	...	...	...	...
186845	259353.0	AAA Batteries (4-pack)	3.0	2.99	2019-09-17 20:56:00	840 Highland St, Los Angeles, CA 90001	9
186846	259354.0	iPhone	1.0	700.00	2019-09-01 16:00:00	216 Dogwood St, San Francisco, CA 94016	9
186847	259355.0	iPhone	1.0	700.00	2019-09-23 07:39:00	220 12th St, San Francisco, CA 94016	9
186848	259356.0	34in Ultrawide Monitor	1.0	379.99	2019-09-19 17:30:00	511 Forest St, San Francisco, CA 94016	9
186849	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

Figure 15 Creation of month column as integer data type

In the above image, first order date column was changed to pandas date and time, then new column month was created by extracting month from the order date. It was also converted into integer using astype which helps to convert data type.

```
In [77]: df.dtypes#dtypes helps to show data type of columns

Out[77]: Order ID                  float64
         Product                    object
         Quantity Ordered          float64
         Price Each                float64
         Order Date          datetime64[ns]
         Purchase Address           object
         Month                       int32
         dtype: object
```

Figure 16 Data type of month column

The above image shows that the month column was converted to integer.

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

In order to add new column named city the Purchased Address column data should be separated as it contains all the details.

```
In [78]: seperate=df['Purchase Address'].str.split(",")#The purchase address is splited using comma
         df["City"] = seperate.str[1]#the first index of the splited string is converted to new column city
         df
```

Out[78]:

	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
...	...	...	...	...	...	...	...	...
186845	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
186846	259354.0	iPhone	1.0	700.00	2019-09-01 16:00:00	216 Dogwood St, San Francisco, CA 94016	9	San Francisco
186847	259355.0	iPhone	1.0	700.00	2019-09-23 07:39:00	220 12th St, San Francisco, CA 94016	9	San Francisco
186848	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
186849	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 17 Creation of city column

In the above image, first the purchase address column was converted to string and split function is used to breakdown string. By comma the string was separated and was store in variable separate then new column city was created in string by extracting first index value of the split string list.

3. Data Analysis

Data analysis is the process of modifying, processing, and cleansing raw data in order to obtain useful, pertinent information that supports commercial decision-making. The process offers helpful insights and data, frequently displayed in charts, graphics, tables, and graphs, which lessen the risks associated with decision-making (Kelley, 2023).

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

In python there are various math function which helps to find statistic values. Among them is mean, standard deviation, skewness and kurtosis.

```
In [20]: print("Mean of Price Each is",df['Price Each'].mean())
         print("Mean sum of Quantity Ordered is",df['Quantity Ordered'].sum())
         print("Standard Deviation of Price Each is",df['Price Each'].std())
         print("Skewness of Price Each is",df['Price Each'].skew())
         print("Kurtosis of Price Each is", df['Price Each'].kurtosis())

         Mean of Price Each is 184.3997347670135
         Mean sum of Quantity Ordered is 209079.0
         Standard Deviation of Price Each is 332.7313298840936
         Skewness of Price Each is 2.8721487292935257
         Kurtosis of Price Each is 9.094568341148197
```

Figure 18 Statistic values of price each

In the above figure the mean, sum, standard deviation, skewness and kurtosis was calculated by the functions mean(), sum(), std(), skew(), kurtosis of column price each. Mean gives the average value of price each, sum gives that total value of quantity ordered. Standard deviation is a statistic that measures the dispersion of a dataset relative to its mean and is calculated as the square root of the variance which is 332.7 of price each column. Kurtosis is a statistical measure used to describe the degree to which scores cluster in the tails or the peak of a frequency distribution which is at 9.09 of price each. Skewness is used to describes how much statistical data distribution is asymmetrical from the normal distribution which is 2.87 of price each column.

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

A statistical metric called correlation shows how much two or more variables change in connection to one another. Correlation refers to the degree of linear relationship between the two variables which is measured by Correlation Coefficient (r), values ranges from +1 to -1 in which +1 denotes strong positive correlation and -1 denotes strong negative correlation (Wigmore, 2020).

```
In [18]: df.corr()
```

Out[18]:

	Order ID	Quantity Ordered	Price Each
Order ID	1.000000	0.000702	-0.002857
Quantity Ordered	0.000702	1.000000	-0.148272
Price Each	-0.002857	-0.148272	1.000000

Figure 19 Correlation coefficient of dataframe

```
In [29]: corrs=df.corr()#using corr() function correlation cofficient of each column is extracted
         plt.figure(figsize=(6, 4), facecolor='w', edgecolor='k')#color and size of graph is set
         sns.set(font_scale=1.1)#font size of the columns are given
         sns.heatmap(corrs, cmap='coolwarm', center = 0,  annot=True, fmt='.1g')#heatmap is plotted using seaborn library
Out[29]: <AxesSubplot:>
```

Figure 20 Heatmap of correlation coefficient of dataframe

In above dataframe correlation coefficient is calculated using corr() function which shows that mostly +1 or strong postive correlation is between same data which doesn't mean anything. Price each and quantity ordered has negative correlation coefficient

which show if one increases the other one decrease. So, if the price of product is more the amount of quantity ordered is less, whereas if price of product is less than the quantity ordered is more. That's why the most ordered product is AAA Batteries (4-Pack) which cost 2.99 dollars where as most one of the most least ordered is MacBook pro laptop which cost 1700 dollars. The heatmap of the correlation coefficient of dataframe is also plotted to get graphical representation of the data.

4. Data Exploration

Data exploration is the first stage of data analysis, during which users examine a sizable data collection in an unstructured manner in order to find the first patterns, traits, and points of interest. It facilitates the creation of a comprehensive picture of key developments and focal topics for further investigation.

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.

In order to calculate the best sales, we need to multiple the number of quantities by price of product to get total order value.

```
In [33]: #total price column is created by multiplying quantity ordered and price each
         df['Total Price']=df['Quantity Ordered']*df['Price Each']
         df.head()
```

Out[33]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City	Total Price
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	23.90
2	176559.0	Bose SoundSport Headphones	1.0	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	Boston	99.99
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	600.00
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	11.99
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	11.99

Figure 21 Creation of new column total price

In the above figure new column was created by the name total price which gives total value of order. The quantity ordered column is multiplied by price each column.

```
In [35]: df.groupby('Month').sum()#Using groupby function, by month all the other numeric data are sum
Out[35]:
```

Month	Order ID	Quantity Ordered	Price Each	Total Price
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 22 Groupby of all columns sum by month

In the above figure using groupby function the sum of all the column was extracted for each month. It shows that best sells month is December whereas least sales is January.

```
In [31]: sumall=df.groupby('Month')#Using groupby function, month are extracted
price=sumall.sum()['Total Price']#Using groupby function, by month sum of only total price column is extracted
months=[month for month , df in sumall]#for loop is used access same month by the total price
colors=['b','b','b','b','b','b','b','b','b','b','b','r']#color for each bar is given
#Using groupby function, by month all the other numeric data are sum and only total price column is selected
plt.bar(months, price, color=colors)#using matplotlib the bar graph is plotted
plt.xlabel('Months')#x-axis label is given
plt.ylabel('Total Sales')#y-axis label is given
plt.title("Best Sales of Month")
plt.show()
```

Figure 23 Bar graph of best sales of month

In the above figure bar graph of months by its total sales is plotted. Using groupby function month data is extracted and then sum of total price is plotted as y-axis value and the months is plotted as x-axis values. The above graph shows that the best sales month is December where the total sales is of 461344.34 dollars and second highest sells is on October.

4.2. Which city has sold the highest product?

In order to calculate the highest sold product by city, the total of quantity ordered should be taken calculated by the city.

```
In [33]: df.groupby('City').sum()#Using groupby function, by city all the other numeric data are sum
Out[33]:
```

City	Order ID	Quantity Ordered	Price Each	Month	Total Price
Atlanta	3.423838e+09	16602.0	2779908.20	104794	2795498.58
Austin	2.280982e+09	11153.0	1809873.61	69829	1819581.75
Boston	4.598265e+09	22528.0	3637409.77	141112	3661642.01
Dallas	3.415644e+09	16730.0	2752627.82	104620	2767975.40
Los Angeles	6.811085e+09	33289.0	5421435.23	208325	5452570.80
New York City	5.736334e+09	27932.0	4635370.83	175741	4664317.43
Portland	2.868861e+09	14053.0	2307747.47	87765	2320490.61
San Francisco	1.030444e+10	50239.0	8211461.74	315520	8262203.91
Seattle	3.406694e+09	16553.0	2733296.01	104941	2747755.48

Figure 24 Groupby of all columns sum by city

In the above figure the using the groupby function by the city the sum of all the columns was extracted. The city with highest quantity of product sold is San Fransciaco where 50239 products were sold and second highest city is

```
In [50]: #Using groupby function, by city total price is shown using pie chart, explode helps to give space between each
         df.groupby(['City']).sum().plot(kind='pie', y='Quantity Ordered', explode=(0.1, 0.1, 0.1,0.1,0.1,0.1,0.2,0.1,0.1), autopct='%1.0
         plt.legend(bbox_to_anchor=(1.2, 1), loc='upper left', borderaxespad=0)
         plt.title("City with Highest Product Sold")

Out[50]: Text(0.5, 1.0, 'City with Highest Product Sold')
```

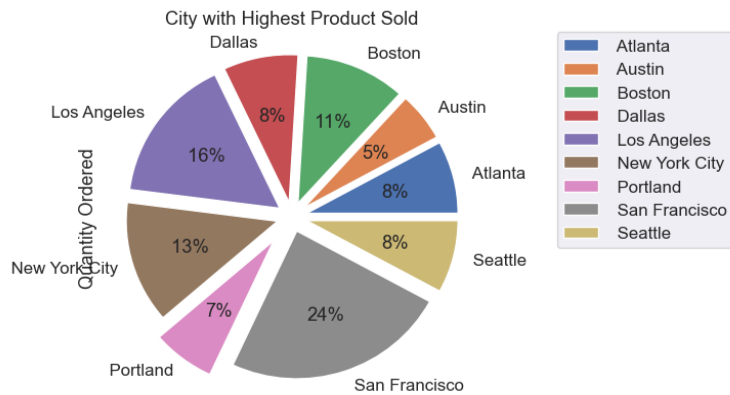


Figure 25 Pie chart of highest sold product

In the above figure using the pie chart the percentage of total quantity is distributed by city. The highest quantity ordered city is San Francisco which cover 24% of total quantity ordered of the year whereas second highest quantity ordered city is Los Angeles which cover 16% of total quantity ordered of the year.

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

In order to calculate the most sold product, the total of quantity ordered should be taken calculated by the product.

```
In [35]: df.groupby('Product').sum()#Using groupby function, by product all the other numeric data are sum
```

Out[35]:

Product	Order ID	Quantity Ordered	Price Each	Month	Total Price
20in Monitor	9.508897e+08	4129.0	451068.99	29336	454148.71
27in 4K Gaming Monitor	1.442589e+09	6244.0	2429637.70	44440	2435097.56
27in FHD Monitor	1.724224e+09	7550.0	1125974.93	52558	1132424.50
34in Ultrawide Monitor	1.418986e+09	6199.0	2348718.19	43304	2355558.01
AA Batteries (4-pack)	4.744174e+09	27635.0	79015.68	145558	106118.40
AAA Batteries (4-pack)	4.764959e+09	31017.0	61716.59	146370	92740.83
Apple Airpods Headphones	3.579120e+09	15661.0	2332350.00	109477	2349150.00
Bose SoundSport Headphones	3.071496e+09	13457.0	1332366.75	94113	1345565.43
Flatscreen TV	1.110943e+09	4819.0	1440000.00	34224	1445700.00
Google Phone	1.262237e+09	5532.0	3315000.00	38305	3319200.00
LG Dryer	1.465563e+08	646.0	387600.00	4383	387600.00
LG Washing Machine	1.507187e+08	666.0	399600.00	4523	399600.00
Lightning Charging Cable	4.994091e+09	23217.0	323787.10	153092	347094.15
Macbook Pro Laptop	1.091958e+09	4728.0	8030800.00	33548	8037600.00
ThinkPad Laptop	9.487932e+08	4130.0	4127958.72	28950	4129958.70
USB-C Charging Cable	5.049538e+09	23975.0	261740.85	154819	286501.25
Vareebadd Phone	4.725325e+08	2068.0	826000.00	14309	827200.00
Wired Headphones	4.350952e+09	20557.0	226395.18	133397	246478.43
iPhone	1.571390e+09	6849.0	4789400.00	47941	4794300.00

Figure 26 Groupby of all columns sum by product

In the above figure the using the groupby function by the product the sum of all the columns was extracted. The product with highest sold quantity is AAA Batteries (4-pack) which was ordred in 31017 times.

```
In [62]: sumall=df.groupby('Product')#Using groupby function, month are extracted
quantity=sumall.sum()['Quantity Ordered']#Using groupby function, by month sum of only total price column is extracted
products=[product for product , df in sumall]#for loop is used access same month by the total price
colors=['b','b','b','b','b','r','b','b','b','b','b','b','b','b','b']#color for each bar is given
#Using groupby function, by month all the other numeric data are sum and only total price column is selected
plt.bar(products, quantity, color=colors)#using matplotlib the bar graph is plotted
plt.xlabel('Products')#x-axis label is given
plt.ylabel('Total Quantity Sold')#y-axis label is given
plt.title("Most Sold Product")
plt.xticks(rotation=90)
plt.show()
```

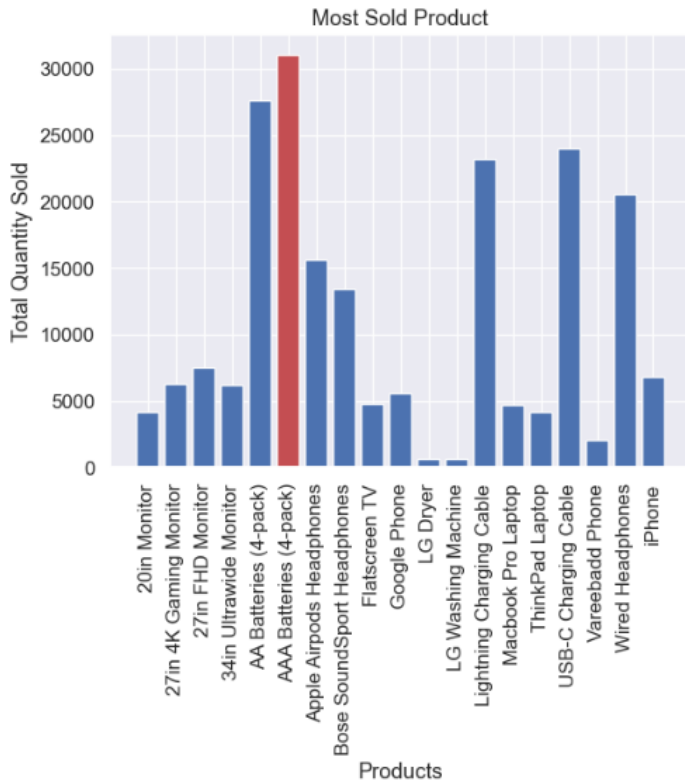


Figure 27 Bar graph of most sold product

In the above figure the bar graph of the product sold by total quantity ordered is plotted which shows that AAA Batteries (4-pack) was the most sold product which is also one of the cheapest products whereas second most sold product is AA Batteries (4-pack) and least sold product is LG Dryer.

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

In order of create a histogram an hour's column was added and according to hour the sales was plotted. Histogram is representation numeric data of a range of outcomes into columns formation along the x-axis.

```
In [52]: df['Hours']=df['Order Date'].dt.hour#hours column is extracted the dataframe
         df.head()
```

Out[52]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City	Total Price	Hours
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	23.90	8
2	176559.0	Bose SoundSport Headphones	1.0	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	Boston	99.99	22
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	600.00	14
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	11.99	14
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	11.99	9

Figure 28 Creation of new column hour

In the above image, new column Hours was created by extracting hour from the order date. Using the hours column, the sales can be calculated hourly.

```
In [56]: df1=df['Hours']#column hours from dataframe is stored in a variable
         plt.hist([df1], bins=34)#histogram was plotted on basic of hour whereas bins helps to set distance between lines
         plt.title("Hourly Sales Histogram")#tite of histogram is given
         plt.xlabel("Hours")#label of x axis is given
         plt.ylabel("Sales")#label of y axis is given
         plt.show()
```

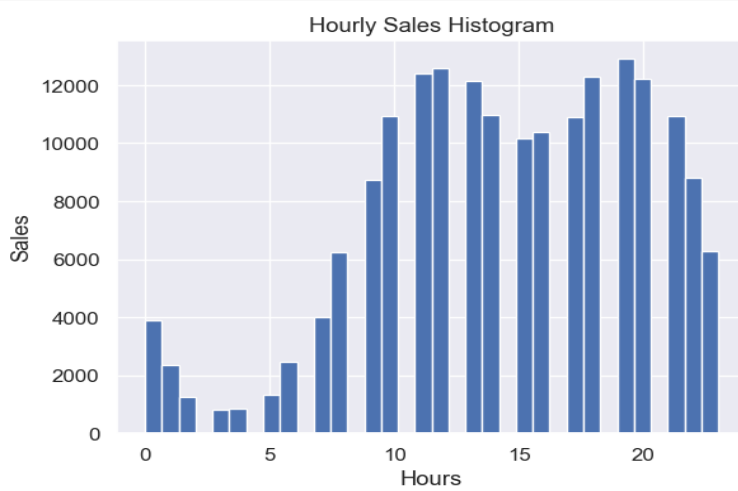


Figure 29 Histogram of hourly sales

In the above figure the sales of by hour is shown in the histogram. From above figure, it is observed that the most of products are sold after 9am to 10pm. The highest frequency data is on 7pm whereas least frequency data is at 4am in the morning. So, most at night and evening hours the goods are ordered the most.

5. Findings

There were various findings that were discovered after the going through various phases of data understanding. They are listed below

- According to the data, this data is of ABC company sells various products including phones, TV's, laptop, monitors, batteries, headphone, dryer and charging cables. This store is located in USA and is sold in 9 cities of USA.
- The company sells this year is of total 34492035.97 dollars.
- The store sells most in month of December which was total of 4613443.34 in which total quantity order was 10903 dollars whereas most least sells in month of January which was total of 1822256.73 dollars in which total quantity order was 28114. So, as Christmas is celebrated in December the sells is observed to be high so quantity of product should be increased in that month.
- In the city of San Francisco total quantity ordered highest of 50239 with highest total sales of 8262203.91 dollars. So, the company should open more its branch in San Francisco.
- In the city of Austin total quantity ordered lowest of 11153 with lowest total sales of 1819581.75 dollars.
- Highest quantity ordered product is AAA Batteries (4-pack) of 31017 and LG Dryer is least quantity order though the total sales of LG Dryer is way more than AAA Batteries due to difference in price.
- In 3AM in the morning least quantity of product is ordered which is 831 whereas in 7PM in the evening most quantity of product is ordered which is 12905
- 68552 people have order only one product at a time and only 3 times people have ordered most product which is 9.

References

Alok, U., 2022. *Creating Heatmap Using Python Seaborn.* [Online]
Available at: https://blog.quantinsti.com/creating-heatmap-using-python-seaborn/
[Accessed 26 04 2023].

Kelley, K., 2023. *What is Data Analysis? Methods, Process and Types Explained.* [Online]
Available at: https://www.simplilearn.com/data-analysis-methods-process-types-article
[Accessed 29 04 2023].

Stedman, C., 2022. *What is data preparation? An in-depth guide to data prep.* [Online]
Available at: https://www.techtarget.com/searchbusinessanalytics/definition/data-preparation
[Accessed 27 04 2023].

Stojiljković, M., 2020. *The pandas DataFrame: Make Working With Data Delightful.* [Online]
Available at: https://realpython.com/pandas-dataframe/
[Accessed 27 04 2023].

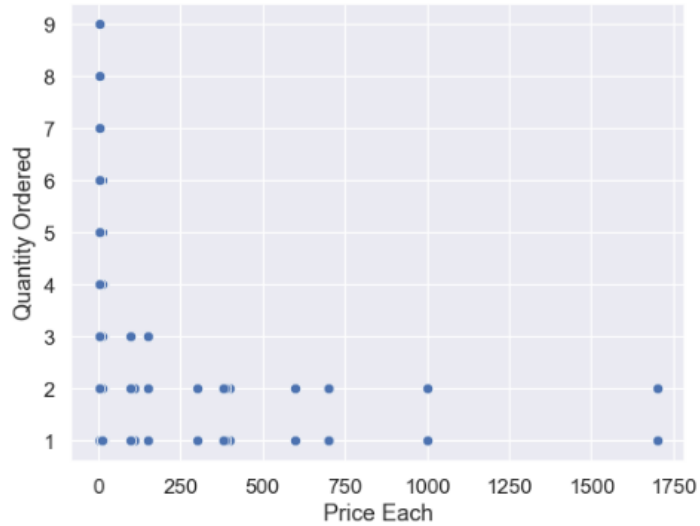
Wigmore, I., 2020. *Correlation.* [Online]
Available at: https://www.techtarget.com/whatis/definition/correlation
[Accessed 27 04 2023].

Yash_R, 2023. *pandas.concat() function in Python.* [Online]
Available at: https://www.geeksforgeeks.org/pandas-concat-function-in-python/
[Accessed 26 04 2023].

Appendix

Extra graph and data analysis is shown here.

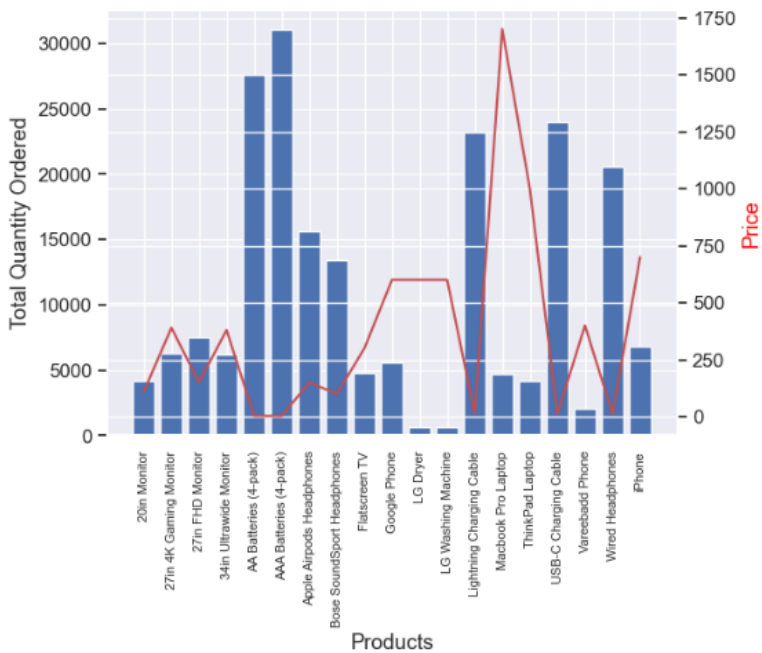
```
In [30]: sns.scatterplot(x="Price Each", y="Quantity Ordered", data=df)#scatter plot helps to show correlation between data
Out[30]: <AxesSubplot:xlabel='Price Each', ylabel='Quantity Ordered'>
```



In the above figure scatter plot between price each and quantity ordered is plotted. A scatter plot is used show relation between two numeric variables and it uses dots to represent values for two different numeric variables. From the above figure we can observed that only the least price product is ordered in quantity more than 3. High price product is only ordered in the quantity of 1 or 2 which shows that costly price ordered in least amount whereas low-cost product is ordered in large quantity.

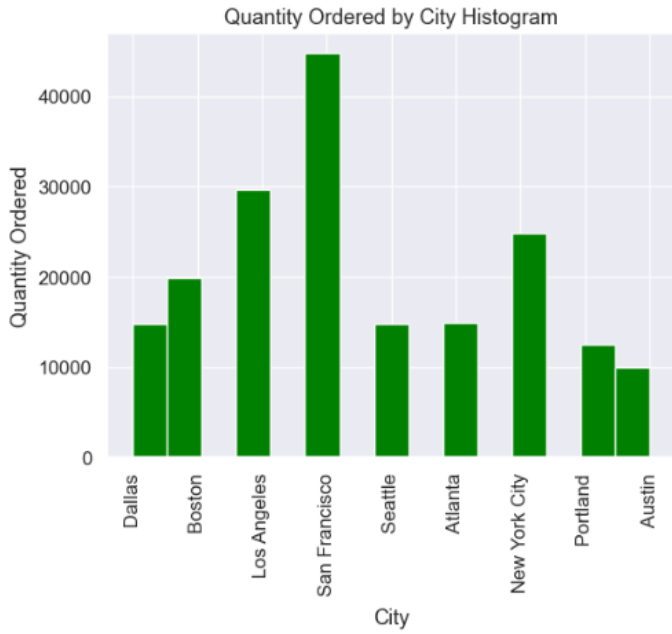
```
In [57]: sump=df.groupby('Product')#Using groupby function, product is extracted
         price=df.groupby('Product').mean()['Price Each']#Using groupby function, by product sum of only total price column is extracted
         fig, axis1=plt.subplots()
         axis2=axis1.twinx()
         quantity=sump.sum()['Quantity Ordered']
         products=[product for product , df in sump]#for loop is used access same product by the total price
         axis1.bar(products, quantity)#using product name and quantity order bar graph is plotted
         axis2.plot(products, price, 'r-')#using product and its price a line graph is plotted
         #using label, labels are given to following axis
         axis1.set_xlabel('Products')
         axis1.set_ylabel('Total Quantity Ordered')
         axis2.set_ylabel('Price', color='red')
         #xticklabels helps to increase height of the graph
         axis1.set_xticklabels(products, rotation='vertical', size=8,)
         plt.show()

C:\Users\lucky\AppData\Local\Temp\ipykernel_16404\2801539207.py:14: UserWarning: FixedFormatter should only be used together wi
th FixedLocator
  axis1.set_xticklabels(products, rotation='vertical', size=8,)
```



In the above image, it is observed that products with its total quantity is plotted as bar graph whereas the line is plotted using plot of price of each product. It shows that least price product is ordered the most whereas highest price product is also sold in good quantity rather than medium price product which are least ordered product. AAA batteries (4-pack) has least price but ordered the most whereas LG Dryer is least ordered product but its price is medium.

```
In [64]: df1=df['City']#column city from dataframe is stored in a variable
         colors = ['green']
         plt.hist([df1], bins=15, color = colors)#histogram was plotted on basic of city whereas bins helps to set distance between lines
         plt.title("Quantity Ordered by City Histogram")#tite of histogram is given
         plt.xticks(rotation=90)#helps to increase height of the graph
         plt.xlabel("City")#label of x axis is given
         plt.ylabel("Quantity Ordered")#label of y axis is given
         plt.show()
```



In the above image the histogram of quantity ordered are shown on the basic of city. The highest frequency of quantity ordered is in San Francisco whereas lowest frequency quantity ordered is in Austin.

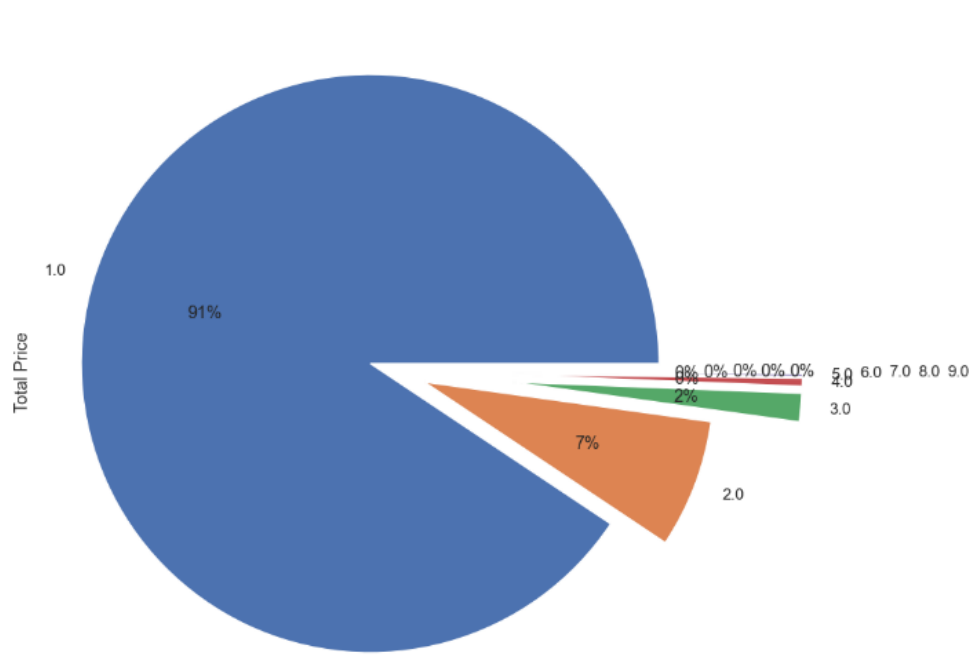
```
In [67]: df.groupby(['Quantity Ordered']).count()
```

Out[67]:

Quantity Ordered	Order ID	Product	Price Each	Order Date	Purchase Address	Month	City	Total Price	Hours
1.0	168552	168552	168552	168552	168552	168552	168552	168552	168552
2.0	13324	13324	13324	13324	13324	13324	13324	13324	13324
3.0	2920	2920	2920	2920	2920	2920	2920	2920	2920
4.0	806	806	806	806	806	806	806	806	806
5.0	236	236	236	236	236	236	236	236	236
6.0	80	80	80	80	80	80	80	80	80
7.0	24	24	24	24	24	24	24	24	24
8.0	5	5	5	5	5	5	5	5	5
9.0	3	3	3	3	3	3	3	3	3

Using the above figure, it can be seen that highest ordered quantity is 1 with 168552 total order records and least ordered quantity at a time is 9 with 3 total order records.

```
In [70]: df.groupby(['Quantity Ordered']).count().plot(kind='pie', y='Total Price', explode=(0.1, 0.1, 0.4,0.4,0.4,0.5,0.6,0.7,0.8), auto
         plt.legend(bbox_to_anchor=(1.4, 1), loc='upper left', borderaxespad=0)
Out[70]: <matplotlib.legend.Legend at 0x25643e8c070>
```



From pie it can observed that 91% of order has only one quantity ordered whereas 7% of order has two quantities ordered.