CC5067NI

60% Individual Coursework

2022-23 Spring

Student Name: Sandesh Prasad Paudel

London Met ID: 22015762

College ID: NP01CP4S220035

Assignment Due Date: 2023-05-04

Assignment Submission Date: Thursday, May 4, 2023

Word Count: 4000 words

Table of Contents

Table of Figures

List of Tables

Abstract

For this coursework, data analysis activities on the 2019 sales data of the ABC Company must be executed with programming techniques and skills. The preparation of the data for subsequent data mining and analysis is the main goal. Understanding, gathering, exploring, and preliminary analysis of data are all part of the coursework Understanding the data resources and their characteristics is the first step. Following that, a Python program is created to combine the data from every month into a single CSV and read an updated dataframe. In order to complete data analysis tasks, this coursework exhibits problem-solving, critical thinking, and assessment abilities. A Python program is built to present summary statistics of selected variables, compute and display the correlation of all variables, and analyse the data. Finally, for data exploration- bar graph, pie-chart and histogram plot are made with proper labels.

Acknowledgment

I would like to express my gratitude to our lecturer Mr. Dipeshwor Silwal and tutor Mr. Sarun Dahal for their guidance, support, and feedback throughout this coursework. Their expertise and insight were invaluable in shaping the direction and scope of this research. My completion of this coursework could not have been accomplished without the support of my classmates, Abhisan, Riwaj, and Rohan. Thank you everyone who assisted in completion of my coursework and by allowing me time away from you to research and write.

Introduction

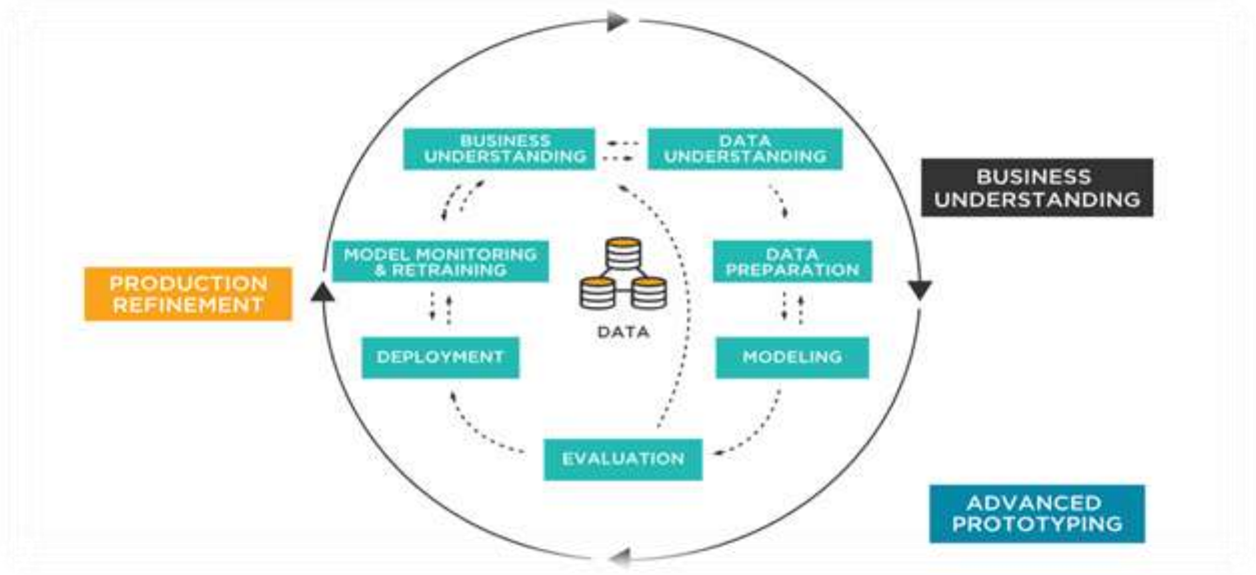
The sections on data preparation and interpretation used the Python packages NumPy and pandas. NumPy is a library that provides access to a great number of mathematical functions that may be used to do complex computations on enormous datasets in addition to supporting multi-dimensional arrays and matrices. Given that NumPy provides a speedy and efficient method for performing numerical operations, it is the best choice for data analysis and scientific computing.

On the other hand, the Python package Pandas provides features for data analysis and manipulation. It provides a reliable data structure called a Data Frame that is similar to a table in a relational database and allows programmers to easily manage and analyse large datasets using a number of functions and techniques. For data exploration, we used the matplotlib and seaborn visualization libraries. Python with the adaptable package matplotlib may be used to create a range of static, interactive, and animated displays. It provides a wide range of plotting options, including line plots, scatter plots, bar charts, histograms, heat maps, and more. Programmers are able to change the color, labels, and axis bounds of the representations since matplotlib provides a wide range of customization choices.

Seaborn is a Python data visualization package that builds on matplotlib. It provides a sophisticated user interface for creating statistical graphics that are both enlightening and visually beautiful. Seaborn can generate a wide variety of charts, including those with uniform and bivariate distributions, categorical data plots, matrix plots, and regression graphs.

1. Data Understanding

Data understanding comes next after enterprise understanding. This contains a list of all the data that is available. Here, you must closely collaborate with the business group because they are aware of the information that is available. The facts that should be applied to this business issue, and other relevant data. In this step, the data are described together with their structure, relevancy, and record type. Utilize graphical graphs to investigate the data. Basically, extracting any information you can about the information by merely looking through the data (Dikshamulchandani, 2023).



Understanding about types of data or variables

Data are generally divided into two types which are qualitative data and quantitative data.

1. Qualitative or categorical data

Nominal and ordinal categorical data are the two categories that can be separated. Nominal data has an inherent order, and the categories are exclusive of one another. Nominal data examples include gender, eye color, and car kind. On the other hand, ordinal data has a natural order or ranking. Education level (e.g., high school, college, graduate school) and rating scales (e.g., highly agree, agree, neutral, disagree, disagree, strongly disagree) are two examples of ordinal data.

2. Quantitative or numerical data

Numerical data are values or observations that come from measurements. There are two types of different numerical values: discrete and continuous numbers. Discrete values are values that can be counted and that are distinct and separated from each other. Continuous values, on the other hand, are values produced by measurements or observations that assume any value within a defined range (Nelli, 2015).

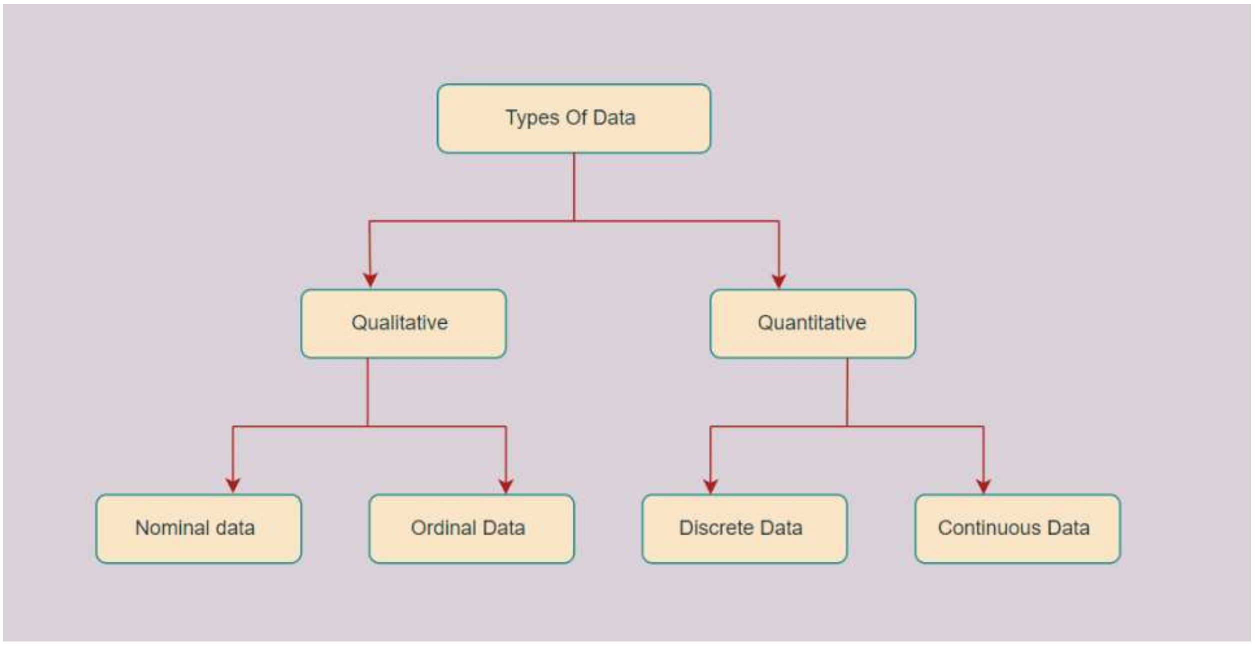


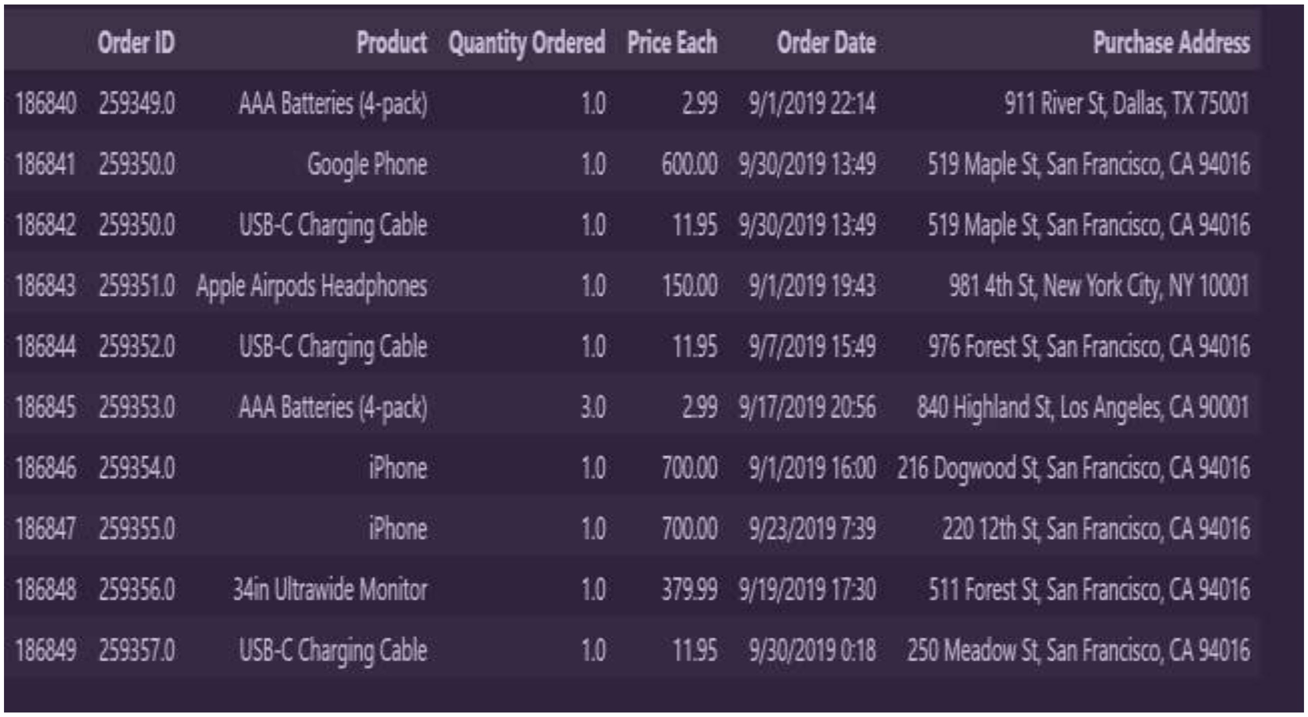
Figure 1 Types of data in data analysis

1.1 Description of Datasets

Datasets are a group of data pertaining to a certain subject, theme, or sector. Datasets can be saved in a variety of formats, such as CSV, JSON, or SQL, and include numerous forms of information, including numbers, text, photos, videos, and audio. As a result, a dataset often contains structured data that is connected to the same issue and is used for that purpose (Siman, 2022).

1.1.1 Our Datasets of Monthly Sales 2019

We have a dataset containing sales information for the year 2019 about the ABC Company. The dataset includes various attributes for each sales such as order ID, product sold, quantity ordered, price per unit, and date plus purchase address of the sale. We can use these data to obtain valuable insights which can help in business decisions and drive growth.



	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
186840	259349.0	AAA Batteries (4-pack)	1.0	2.99	9/1/2019 22:14	911 River St, Dallas, TX 75001
186841	259350.0	Google Phone	1.0	600.00	9/30/2019 13:49	519 Maple St, San Francisco, CA 94016
186842	259350.0	USB-C Charging Cable	1.0	11.95	9/30/2019 13:49	519 Maple St, San Francisco, CA 94016
186843	259351.0	Apple Airpods Headphones	1.0	150.00	9/1/2019 19:43	981 4th St, New York City, NY 10001
186844	259352.0	USB-C Charging Cable	1.0	11.95	9/7/2019 15:49	976 Forest St, San Francisco, CA 94016
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

Figure 2 Sample of given datasets of 2019

1.1.2 Nature of our variable given in datasets

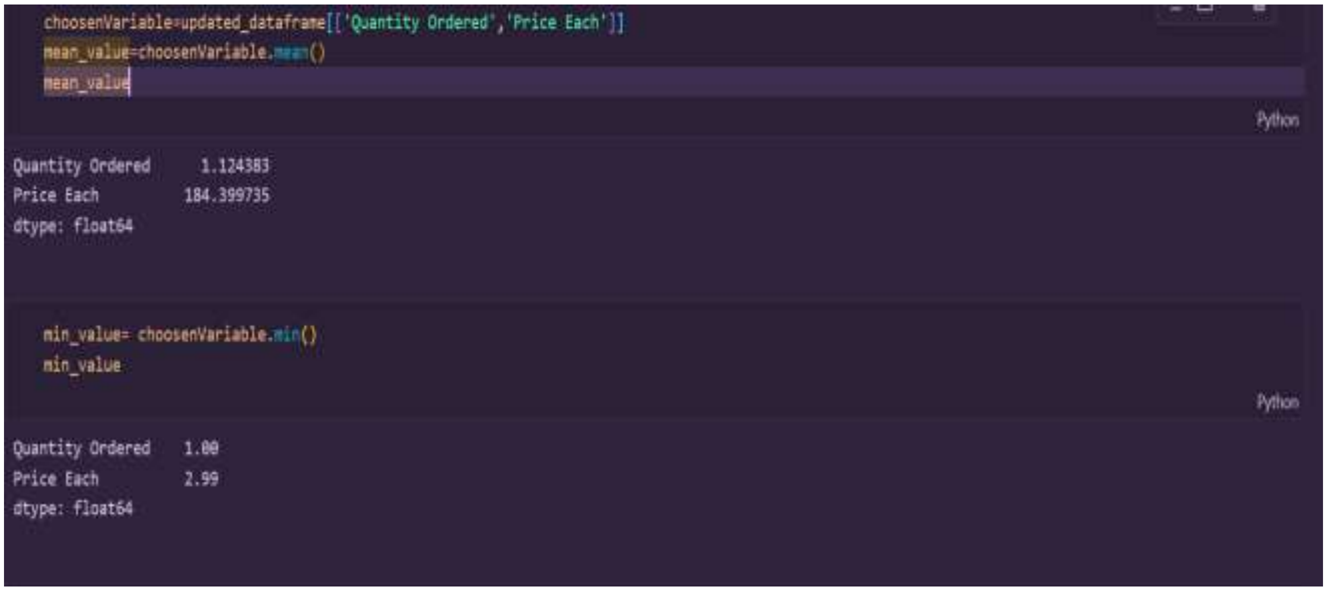
A variable in statistics is a quality or attribute of a thing or a person that can have distinct values. The suitable techniques for data analysis and interpretation are determined by the kind and properties of a variable, which are referred to as its nature.

S.No	Column Name	Description	Data Type	Nature of Variable
1.	Order ID	Unique identifier for every order of products that the user have done.	Integer	Discrete Variable.
2.	Product	Name or type of product that was sold in every transaction.	Object	Nominal Variable.
3.	Quantity Ordered	Quantity of the product ordered in each transaction.	Float	Discrete Variable.
4.	Price Each	Price per unit of the product in each transaction.	Float	Continuous Variable.
5.	Order Date	Date of each transaction.	Object	Nominal Variable.
6.	Purchase Address	Address of the purchaser in each transaction.	Object	Nominal Variable.

Table 1 Data Description of our Monthly Sales Datasets

A. Quantitative variable in our datasets

Since quantitative or discrete variable can be used to find the minimum, maximum, range and mean. So I have used two of the variable as ‘Quantity Ordered’ and ‘Price Each’ for that purpose.

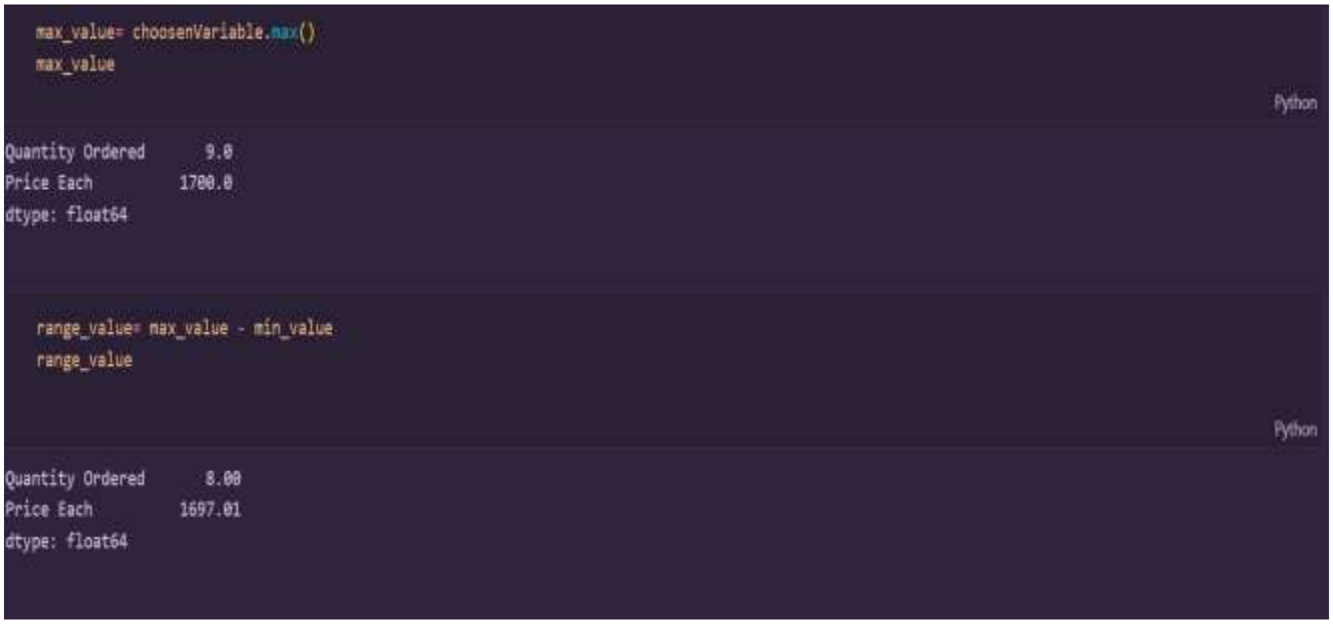


```
choosenVariable=updated_dataframe[['Quantity Ordered','Price Each']]
mean_value=choosenVariable.mean()
mean_value
```

```
Quantity Ordered      1.124383
Price Each          184.399735
dtype: float64
```

```
min_value= choosenVariable.min()
min_value
```

```
Quantity Ordered    1.00
Price Each          2.99
dtype: float64
```



```
max_value= choosenVariable.max()
max_value
```

```
Quantity Ordered       9.0
Price Each          1700.0
dtype: float64
```

```
range_value= max_value - min_value
range_value
```

```
Quantity Ordered       8.00
Price Each          1697.01
dtype: float64
```

Figure 3 Use of quantitative variable for calculation

B. Qualitative variable in our datasets

The variables 'Product' and 'Purchase Address' in this dataset stand in for the nominal under categorical or qualitative data. The names of the items that were purchased are shown in the 'Product' variable, including "AAA Batteries (4-pack), "Google Phone," "USB-C Charging Cable" , "Apple Airpods Headphones," "iPhone," and "34in Ultrawide Monitor." This variable has no intrinsic ordering or numerical value because it reflects several product categories.

'Purchase Address' variable holds the address of the store where the item was bought, such as "911 River St, Dallas, TX 75001", "519 Maple St, San Francisco, CA 94016", "981 4th St, New York City, NY 10001", "840 Highland St, Los Angeles, CA 90001", and "250 Meadow St, San Francisco, CA 94016". This variable is also nominal because it reflects several address types without having any inherent numerical value or ordering.

2. Data Preparation

Prior to processing and analysis, raw data must be cleaned and transformed, which is known as data preparation. It is a crucial stage before processing and frequently include reformatting data, correcting data, and integrating data sets to enrich data. For analysts or business users, preparing data can be a time-consuming task, but it is necessary to put data in context before turning it into insights and removing bias brought on by bad data quality (Bandgar, 2021).

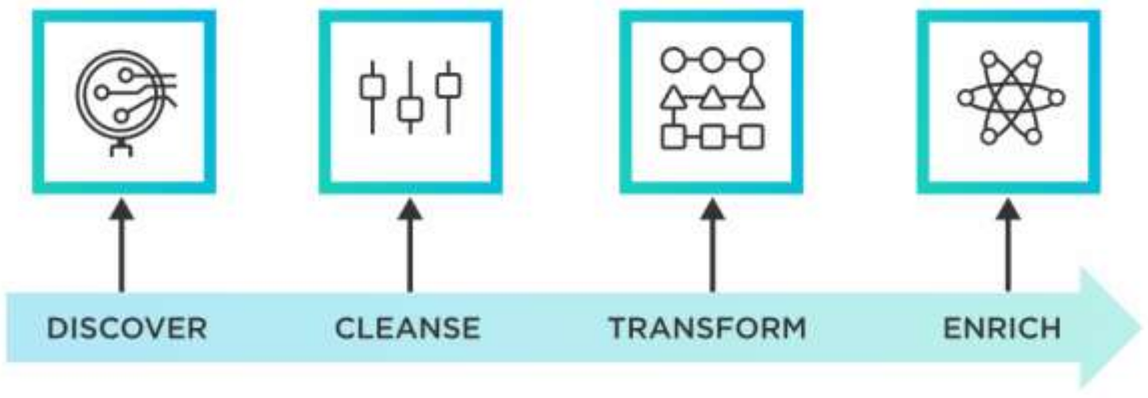


Figure 4 Data Preparation Phase

The following actions are included in the pipeline for data preparation

1. Obtain the data.

2. Data to be ingested (or fetched).

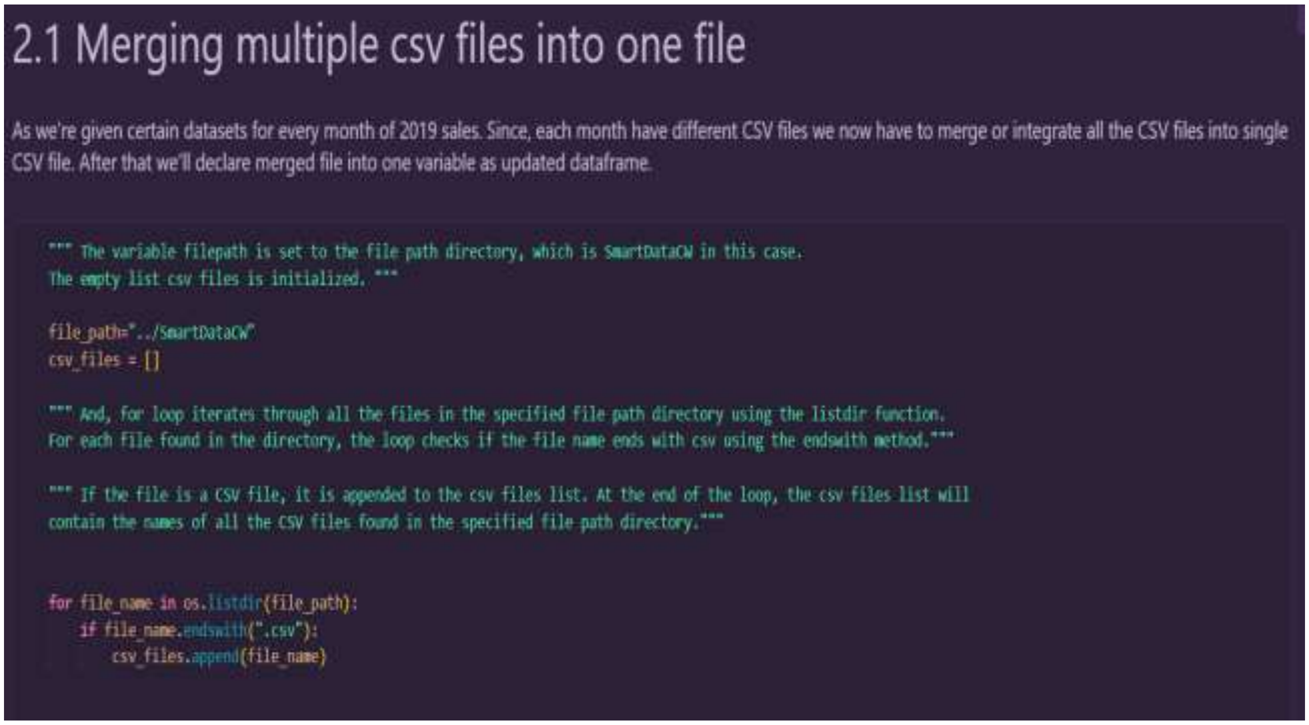
3. Make the data clean.

4. Prepare the data.

5. Add the data together and examine them

Data preparation aims to ensure that data is accurate, complete, consistent, and suitable for analysis. By adopting best practices in data preparation, organizations can derive meaningful insights, make informed decisions, and achieve their business objectives. Effective data preparation can save time and resources in the long run by minimizing errors and improving the accuracy of analysis.

2.1 Python Program to merge multiple csv files into one file.



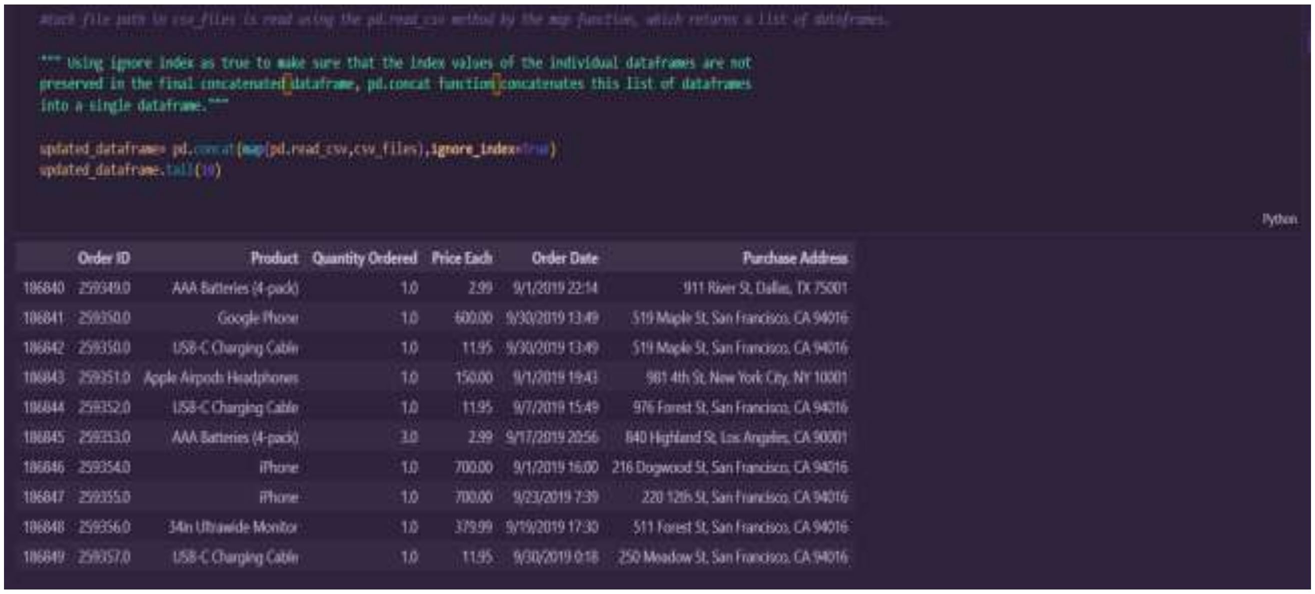
2.1 Merging multiple csv files into one file

As we're given certain datasets for every month of 2019 sales. Since, each month have different CSV files we now have to merge or integrate all the CSV files into single CSV file. After that we'll declare merged file into one variable as updated dataframe.

```python
""" The variable filepath is set to the file path directory, which is SmartDataCW in this case.
The empty list csv files is initialized. """

file_path="../SmartDataCW"
csv_files = []

""" And, for loop iterates through all the files in the specified file path directory using the listdir function.
For each file found in the directory, the loop checks if the file name ends with csv using the endswith method."""

""" If the file is a CSV file, it is appended to the csv files list. At the end of the loop, the csv files list will
contain the names of all the CSV files found in the specified file path directory."""


for file_name in os.listdir(file_path):
    if file_name.endswith(".csv"):
        csv_files.append(file_name)
```

Figure 5 Extracting csv extensions file

Here, we have to set the file path and an empty list. Then, we will have to use for loop in order to extract all the csv files. Then append those csv files to empty list and print the csv files.



```python
#each file path in csv_files is read using the pd.read_csv method by the map function, which returns a list of dataframes.

""" Using ignore index as true to make sure that the index values of the individual dataframes are not
preserved in the final concatenated dataframe, pd.concat function concatenates this list of dataframes
into a single dataframe."""

updated_dataframe= pd.concat(map(pd.read_csv,csv_files),ignore_index=True)
updated_dataframe.tail(10)
```

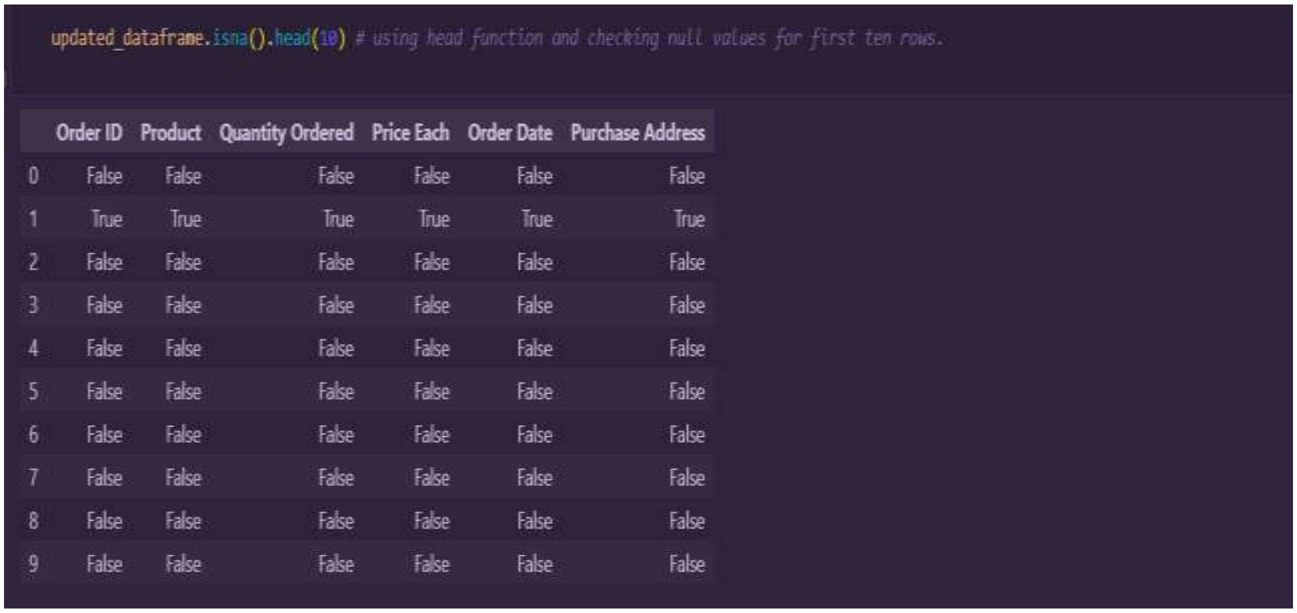
Python

```
        Order ID                  Product  Quantity Ordered  Price Each       Order Date                        Purchase Address
186840  259349.0     AAA Batteries (4-pack)               1.0        2.99   9/1/2019 22:14            911 River St, Dallas, TX 75001
186841  259350.0             Google Phone               1.0      600.00  9/30/2019 13:49     519 Maple St, San Francisco, CA 94016
186842  259350.0       USB-C Charging Cable               1.0       11.95  9/30/2019 13:49     519 Maple St, San Francisco, CA 94016
186843  259351.0  Apple Airpods Headphones               1.0      150.00   9/1/2019 19:43       981 4th St, New York City, NY 10001
186844  259352.0       USB-C Charging Cable               1.0       11.95   9/7/2019 15:49    976 Forest St, San Francisco, CA 94016
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
```

Figure 6 Concatenation of multiple dataframe

Through map function, we have to read individual csv files and convert each file into dataframe. Then, we have to concatenate each dataframe into one by ignoring index.

2.2 Python Program to remove missing values from updated dataframe.

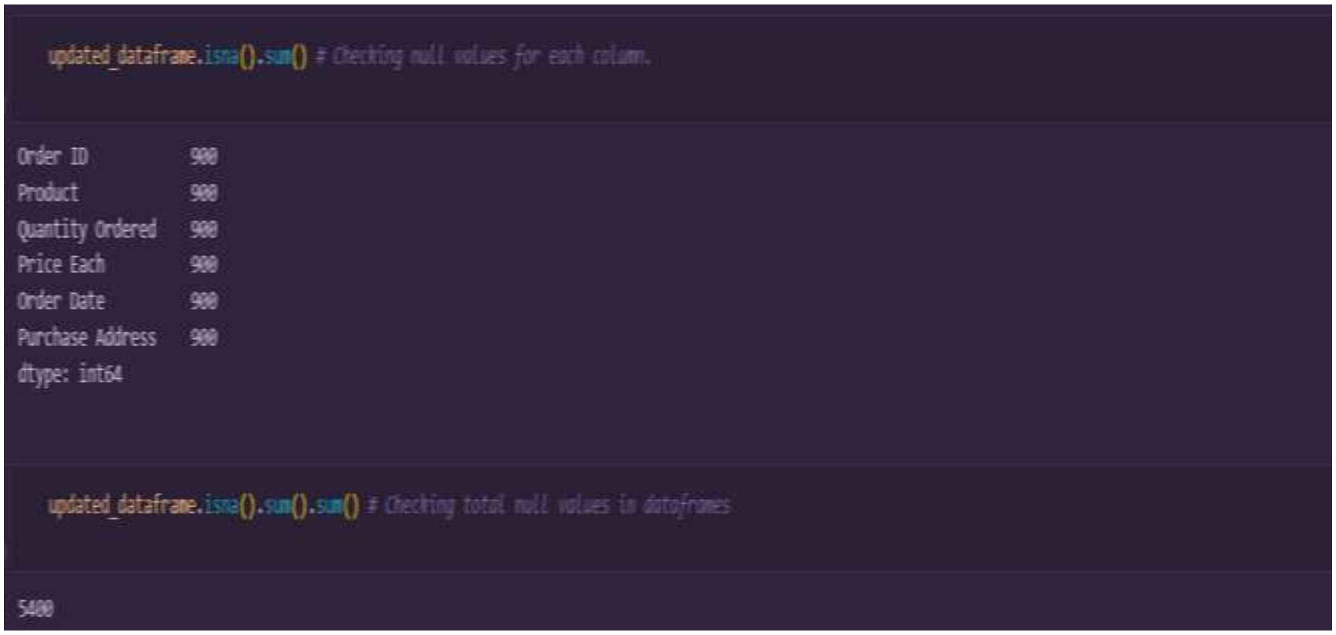


```
updated_dataframe.isna().head(10) # using head function and checking null values for first ten rows.
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	False	False	False	False	False	False
1	True	True	True	True	True	True
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	False	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	False
8	False	False	False	False	False	False
9	False	False	False	False	False	False

Figure 7 Returns boolean while checking null values

First we have to check the null value and display first ten rows that returns boolean values.



```
updated_dataframe.isna().sum() # Checking null values for each column.
```

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

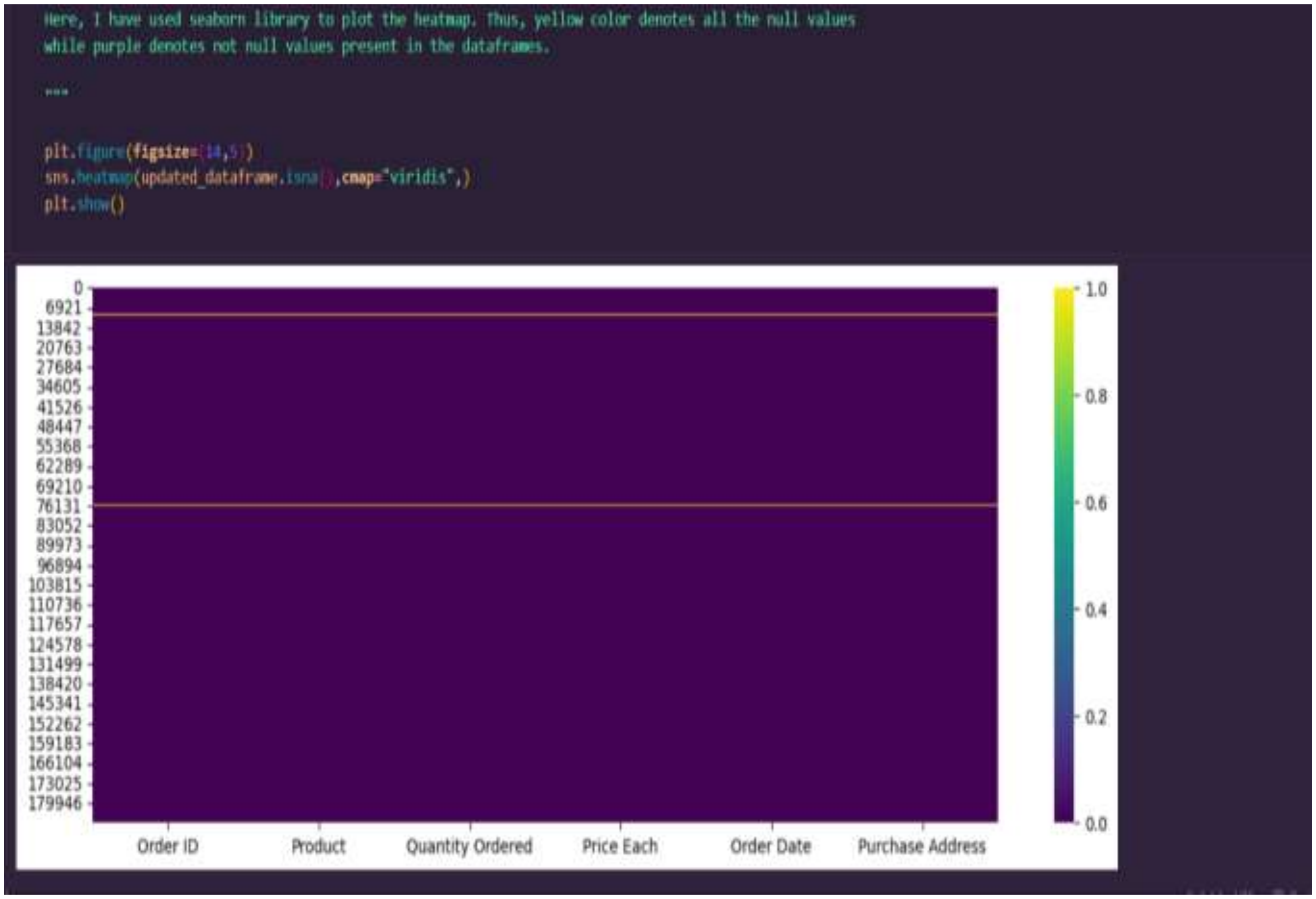
```
updated_dataframe.isna().sum().sum() # Checking total null values in dataframes
```

```
5400
```

Figure 8 Overall null values in our dataframe

And, check the total null values in each column and overall we have to use the sum method for that to be displayed.

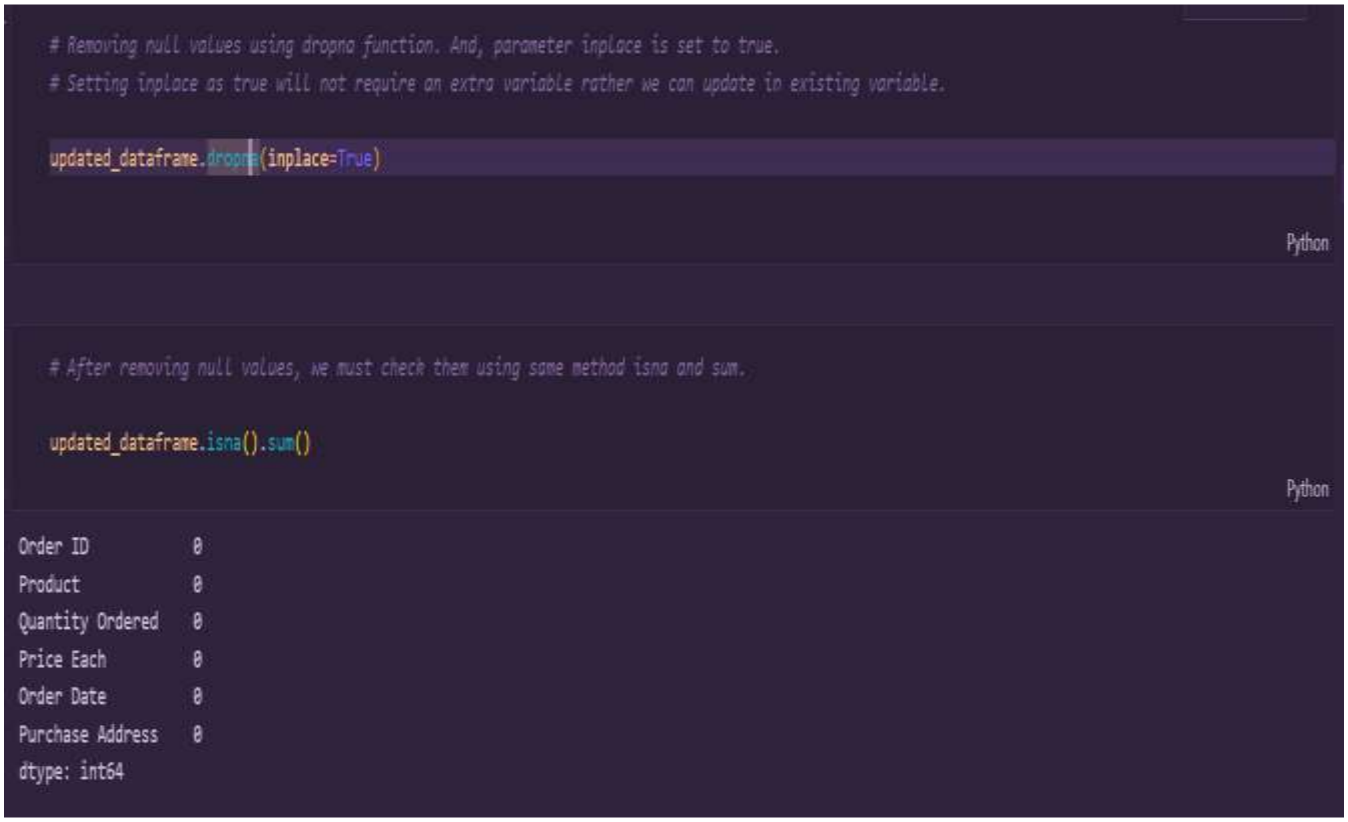
Heatmap for looking null values before removing



```
Here, I have used seaborn library to plot the heatmap. Thus, yellow color denotes all the null values
while purple denotes not null values present in the dataframes.

"""

plt.figure(figsize=(14,5))
sns.heatmap(updated_dataframe.isna(),cmap="viridis",)
plt.show()
```

Figure 9 Heatmap to show null values before removing

Now, for visualizing null values in a heatmap we have use seaborn library. Then, display it using show method.



```
# Removing null values using dropna function. And, parameter inplace is set to true.
# Setting inplace as true will not require an extra variable rather we can update in existing variable.

updated_dataframe.dropna(inplace=True)
```

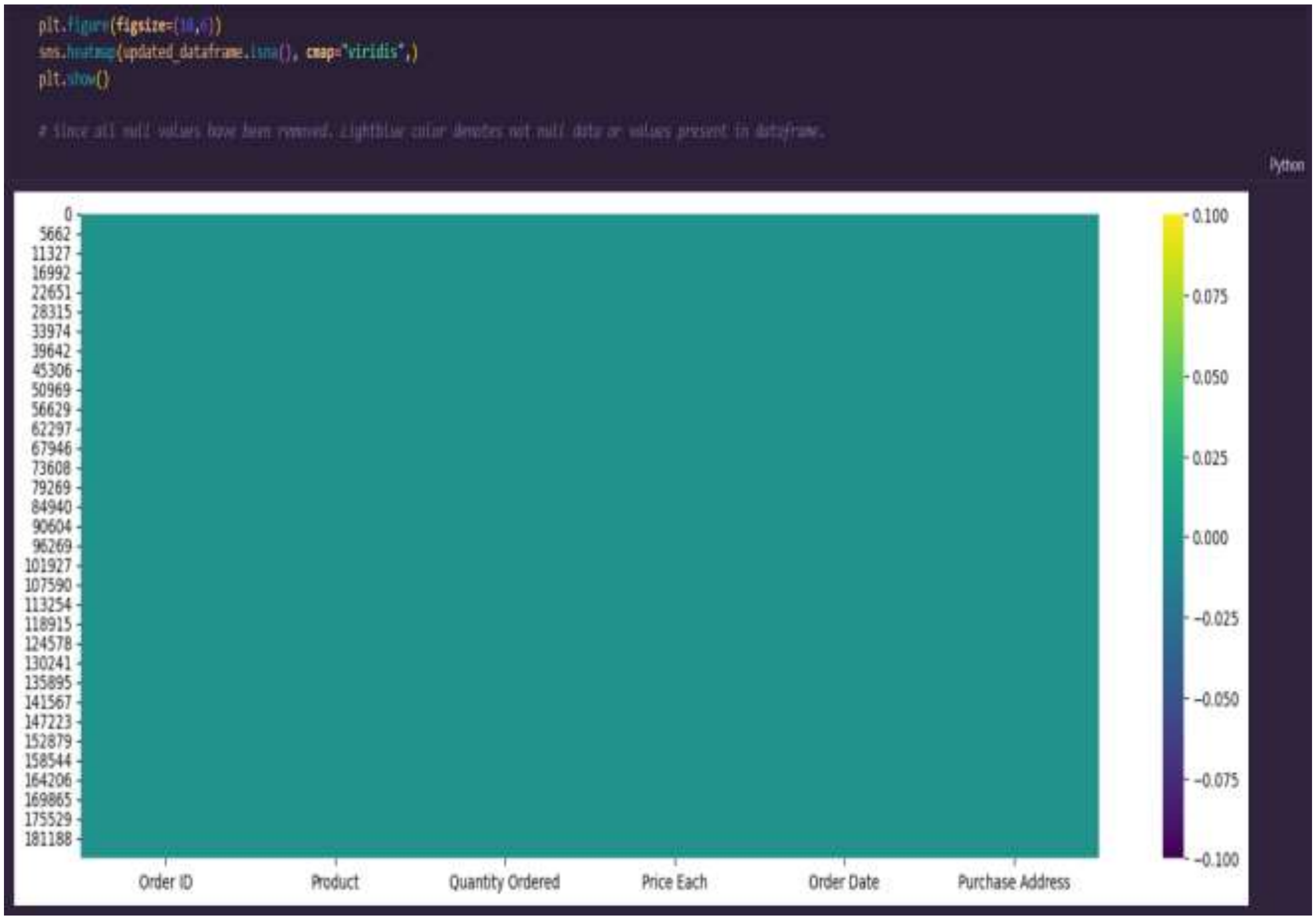
```
# After removing null values, we must check them using same method isna and sum.

updated_dataframe.isna().sum()
```

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

Figure 10 Removing null values from our dataframe

Then, we have to remove null values using dropna method with inplace as true. So, that there will be no need of extra variable. Then looking if all the null values in a column have been removed.

Heatmap for looking null values after removing

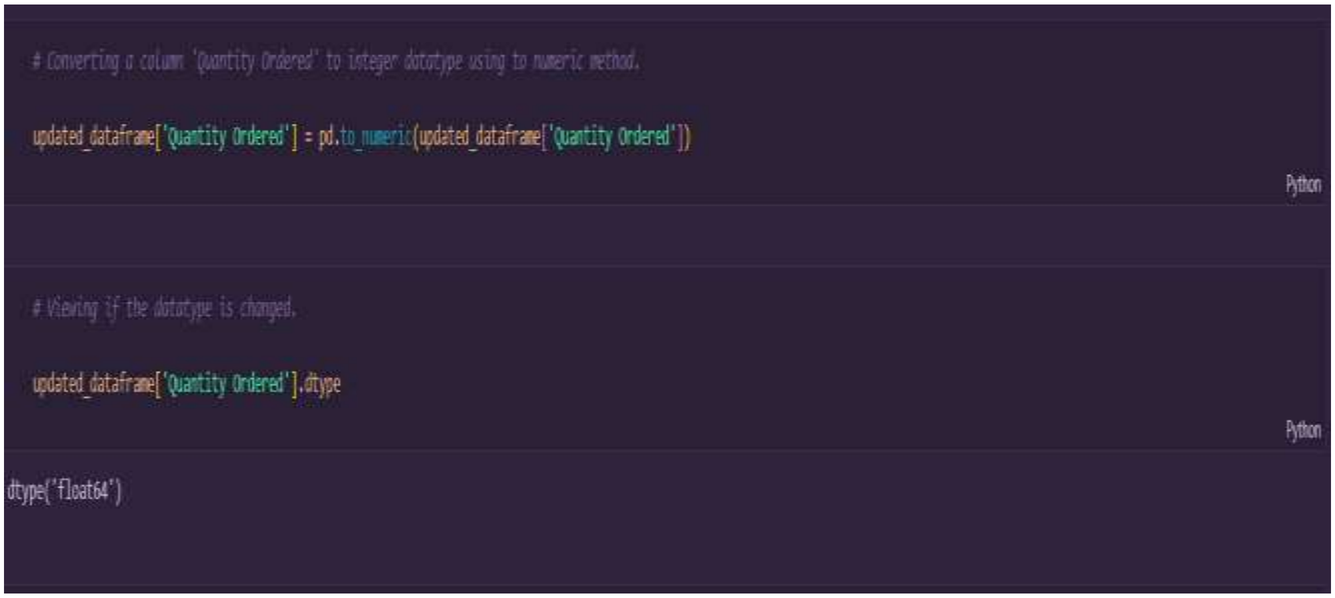


```
plt.figure(figsize=(10,6))
sns.heatmap(updated_dataframe.isna(), cmap="viridis",)
plt.show()

# Since all null values have been removed. Lightblue color denotes not null data or values present in dataframe.
```

Figure 11 Use of heatmap for null values after removing them

Now, checking if all the null values have been removed or not by using a heatmap. Then, displaying using show method.

2.3 Python program to convert quantity ordered and price to numeric.



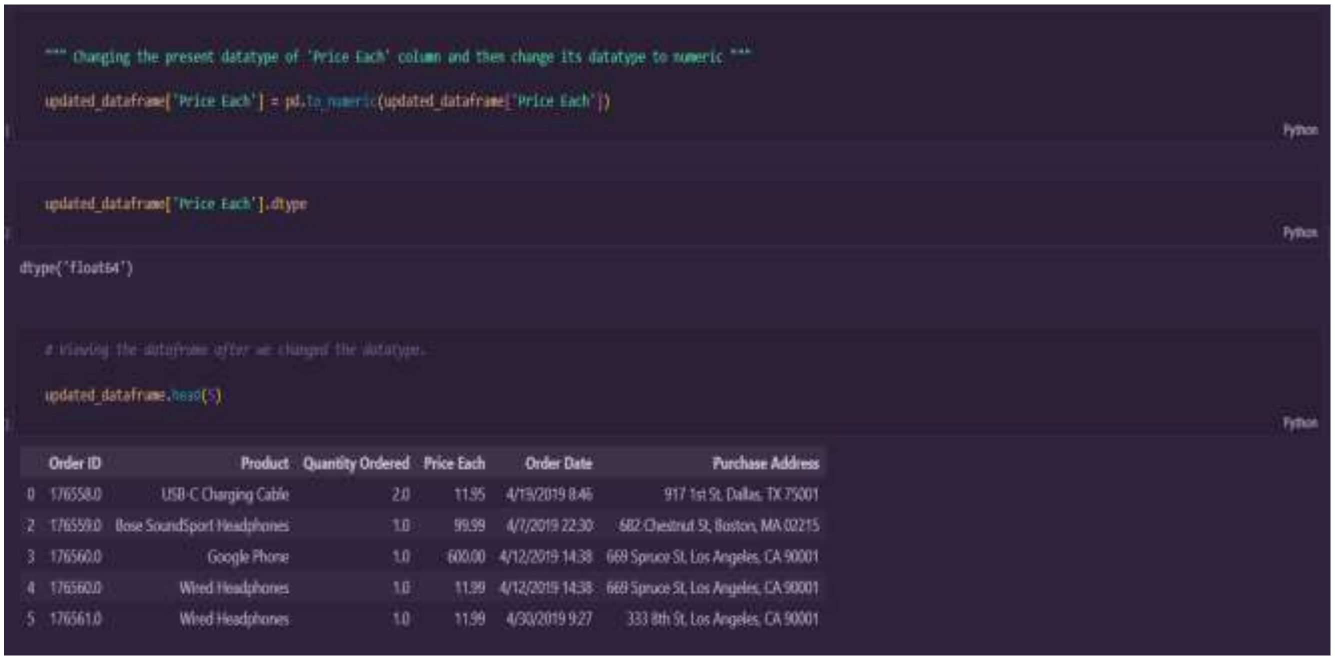
```
# Converting a column 'Quantity Ordered' to integer datatype using to numeric method.

updated_dataframe['Quantity Ordered'] = pd.to_numeric(updated_dataframe['Quantity Ordered'])
```

```
# Viewing if the datatype is changed.

updated_dataframe['Quantity Ordered'].dtype
```

```
dtype('float64')
```

Figure 12 Changing datatype of quantity ordered to numeric

Here, by using to numeric method, we have to change the datatype of quantity ordered and price each.



```
""" Changing the present datatype of 'Price Each' column and then change its datatype to numeric """

updated_dataframe['Price Each'] = pd.to_numeric(updated_dataframe['Price Each'])
```

```
updated_dataframe['Price Each'].dtype
```

```
dtype('float64')
```

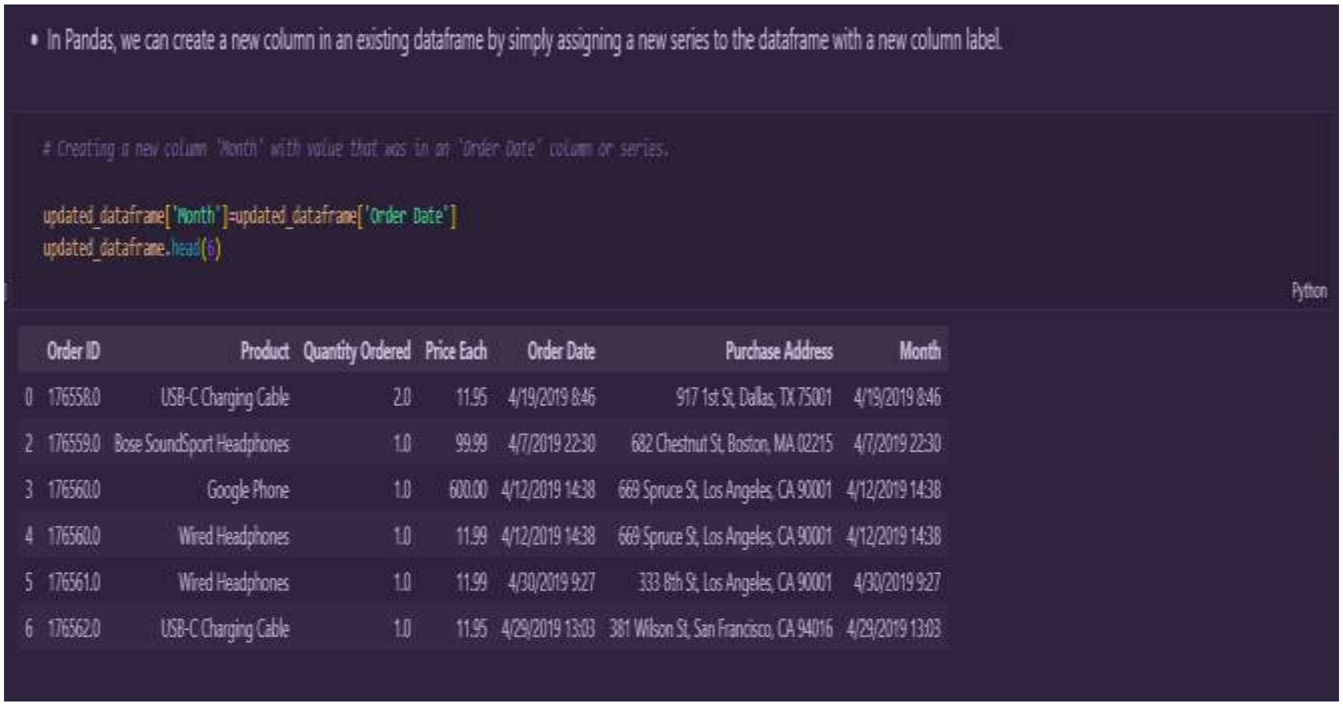
```
# Viewing the dataframe after we changed the datatype.

updated_dataframe.head(5)
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558.0	USB-C Charging Cable	2.0	11.95	4/19/2019 8:46	917 1st St, Dallas, TX 75001
2	176559.0	Bose SoundSport Headphones	1.0	99.99	4/7/2019 22:30	682 Chestnut St, Boston, MA 02215
3	176560.0	Google Phone	1.0	600.00	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560.0	Wired Headphones	1.0	11.99	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561.0	Wired Headphones	1.0	11.99	4/30/2019 9:27	333 8th St, Los Angeles, CA 90001

Figure 13 Changing datatype of price each to numeric

After changing the datatype of price each, we then check the datatype of it and display the first five rows of the updated dataframe.

2.4 Create a new column named month from ordered date of updated dataframe and convert it to integer as data type.



- In Pandas, we can create a new column in an existing dataframe by simply assigning a new series to the dataframe with a new column label.

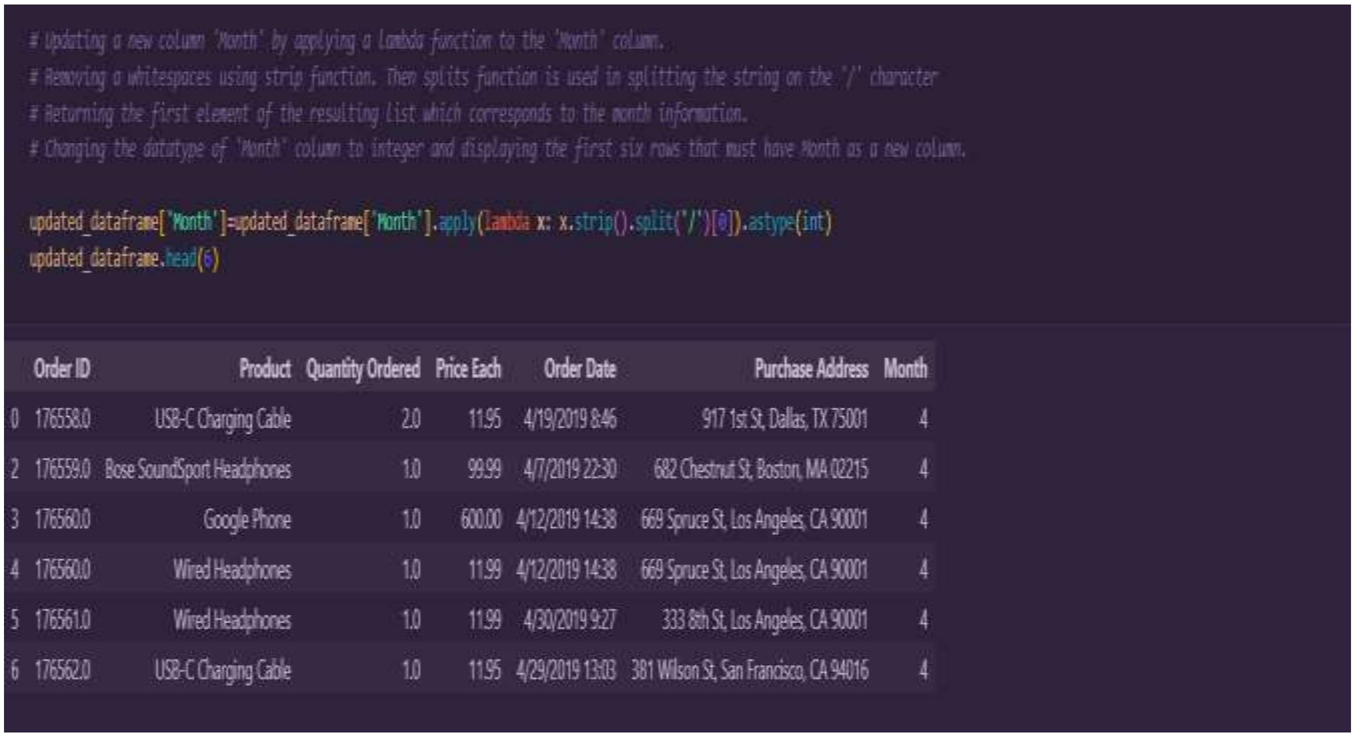
```
# Creating a new column 'Month' with value that was in an 'Order Date' column or series.

updated_dataframe['Month']=updated_dataframe['Order Date']
updated_dataframe.head(6)
```

Python

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558.0	USB-C Charging Cable	2.0	11.95	4/19/2019 8:46	917 1st St, Dallas, TX 75001	4/19/2019 8:46
2	176559.0	Bose SoundSport Headphones	1.0	99.99	4/7/2019 22:30	682 Chestnut St, Boston, MA 02215	4/7/2019 22:30
3	176560.0	Google Phone	1.0	600.00	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	4/12/2019 14:38
4	176560.0	Wired Headphones	1.0	11.99	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	4/12/2019 14:38
5	176561.0	Wired Headphones	1.0	11.99	4/30/2019 9:27	333 8th St, Los Angeles, CA 90001	4/30/2019 9:27
6	176562.0	USB-C Charging Cable	1.0	11.95	4/29/2019 13:03	381 Wilson St, San Francisco, CA 94016	4/29/2019 13:03

Figure 14 Creating a new column named Month

Here, we first have to create a new column named month with the value used in an ordered date. Then by printing the dataframe with six rows we check whether the new column has been added to an existing dataframe.



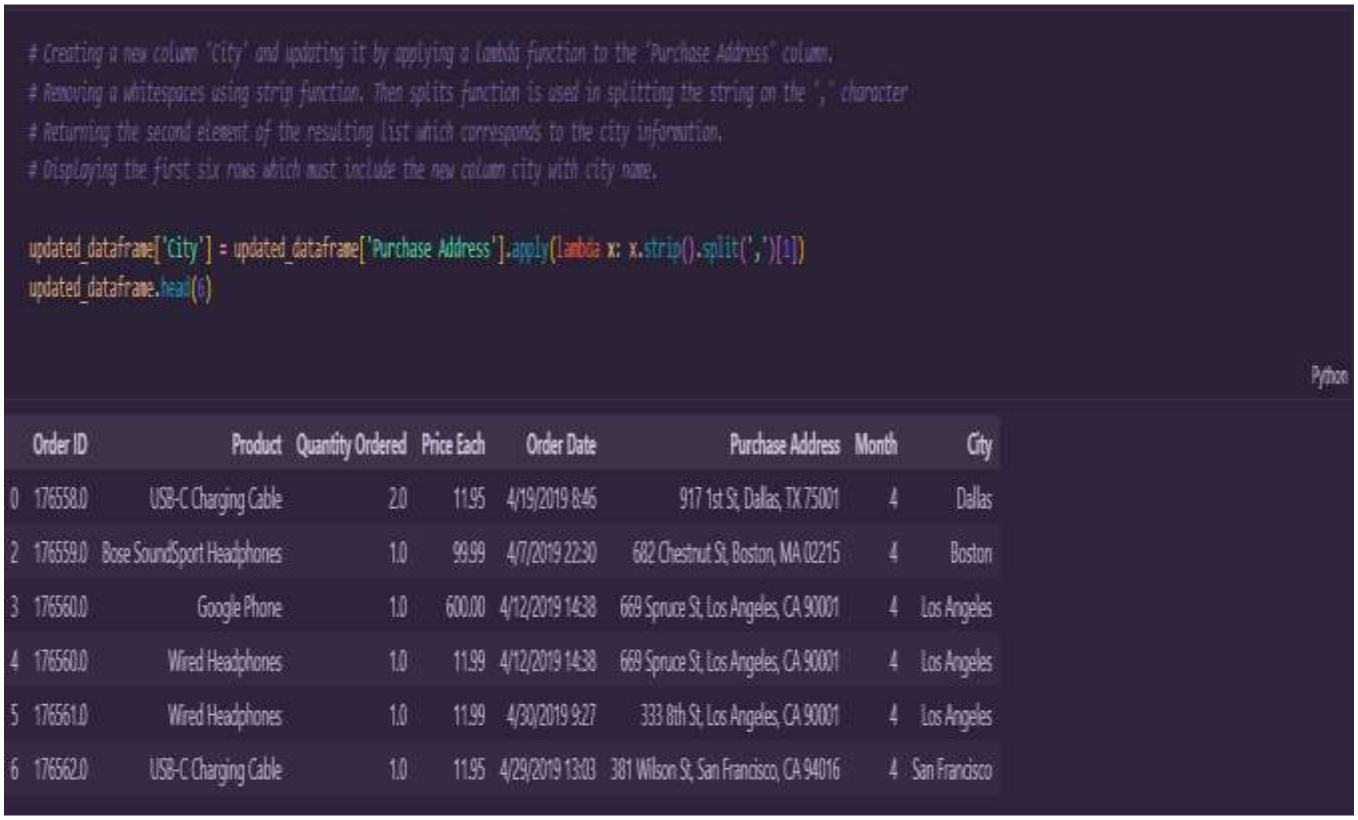
```python
# Updating a new column 'Month' by applying a lambda function to the 'Month' column.
# Removing a whitespaces using strip function. Then splits function is used in splitting the string on the '/' character
# Returning the first element of the resulting list which corresponds to the month information.
# Changing the datatype of 'Month' column to integer and displaying the first six rows that must have Month as a new column.

updated_dataframe['Month']=updated_dataframe['Month'].apply(lambda x: x.strip().split('/')[0]).astype(int)
updated_dataframe.head(6)
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558.0	USB-C Charging Cable	2.0	11.95	4/19/2019 8:46	917 1st St, Dallas, TX 75001	4
2	176559.0	Bose SoundSport Headphones	1.0	99.99	4/7/2019 22:30	682 Chestnut St, Boston, MA 02215	4
3	176560.0	Google Phone	1.0	600.00	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176560.0	Wired Headphones	1.0	11.99	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	176561.0	Wired Headphones	1.0	11.99	4/30/2019 9:27	333 8th St, Los Angeles, CA 90001	4
6	176562.0	USB-C Charging Cable	1.0	11.95	4/29/2019 13:03	381 Wilson St, San Francisco, CA 94016	4

Figure 15 Extracting only month number from existing month column

Since, we have to print only the month number. Apply lambda function that goes up to every existing month value and avoid whitespaces as well as split each value separated by '/' character into lists. Then, fetch the index that have only month number. After that change the existing datatype of Month to integer using astype method. And, finally display first six rows of that updated column to see if the month number is retrieved.

2.5 Create a new column named city from purchase address based on the value in updated dataframe.



```python
# Creating a new column 'City' and updating it by applying a lambda function to the 'Purchase Address' column.
# Removing a whitespaces using strip function. Then splits function is used in splitting the string on the ',' character
# Returning the second element of the resulting list which corresponds to the city information.
# Displaying the first six rows which must include the new column city with city name.

updated_dataframe['City'] = updated_dataframe['Purchase Address'].apply(lambda x: x.strip().split(',')[1])
updated_dataframe.head(6)
```

Python

```
   Order ID                     Product  Quantity Ordered  Price Each       Order Date                       Purchase Address  Month           City
0  176558.0        USB-C Charging Cable               2.0       11.95   4/19/2019 8:46           917 1st St, Dallas, TX 75001      4          Dallas
2  176559.0  Bose SoundSport Headphones               1.0       99.99   4/7/2019 22:30      682 Chestnut St, Boston, MA 02215      4          Boston
3  176560.0                Google Phone               1.0      600.00  4/12/2019 14:38   669 Spruce St, Los Angeles, CA 90001      4     Los Angeles
4  176560.0            Wired Headphones               1.0       11.99  4/12/2019 14:38   669 Spruce St, Los Angeles, CA 90001      4     Los Angeles
5  176561.0            Wired Headphones               1.0       11.99   4/30/2019 9:27      333 8th St, Los Angeles, CA 90001      4     Los Angeles
6  176562.0        USB-C Charging Cable               1.0       11.95  4/29/2019 13:03  381 Wilson St, San Francisco, CA 94016      4   San Francisco
```

Figure 16 Creating a new column named city in an existing dataframe

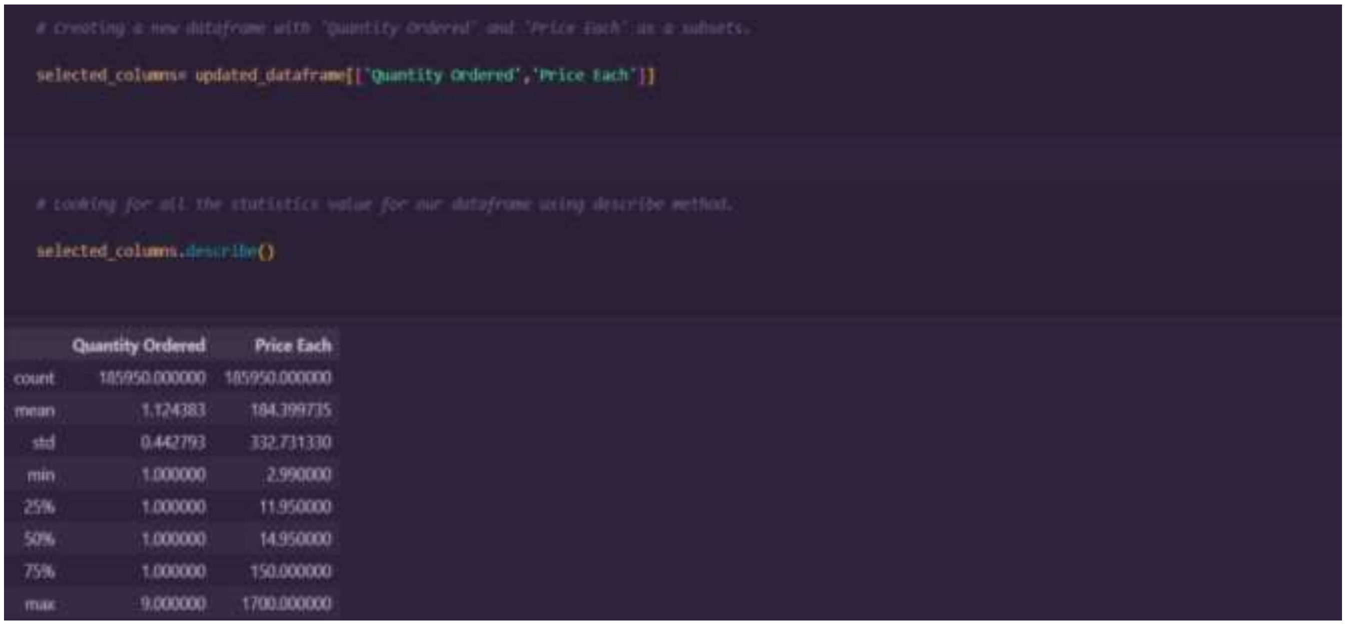
As we have to print only name of city. First create a column named city and then apply lambda function that goes up to every existing city value and avoid whitespaces as well as split each value separated by ‘,’ character into lists. Then, fetch the index that have only city name. And, finally display first six rows of that updated column to see if the city name is retrieved.

3. Data Analysis

The process of cleaning, transforming, and modelling data in order to find relevant information for business decision-making is known as data analysis. Gathering usable information from data and making decisions based on that analysis are the main objectives of data analysis (Johnson, 2023). Different types of data analysis techniques and methods are text analysis, statistical analysis, diagnostic analysis, predictive analysis and prescriptive analysis.

3.1 Summary Statistics

The information about your sample data is summarized and provided by summary statistics. It provides information regarding the values in your data set. This covers the distribution of the mean and the skewness of your data. It includes mean, median, sum, kurtosis, skewness, standard deviation and so on.



```
# Creating a new dataframe with 'Quantity Ordered' and 'Price Each' as a subsets.
selected_columns= updated_dataframe[['Quantity Ordered','Price Each']]

# Looking for all the statistics value for our dataframe using describe method.
selected_columns.describe()
```

```
       Quantity Ordered     Price Each
count     185950.000000  185950.000000
mean           1.124383     184.399735
std            0.442793     332.731330
min            1.000000       2.990000
25%            1.000000      11.950000
50%            1.000000      14.950000
75%            1.000000     150.000000
max            9.000000    1700.000000
```

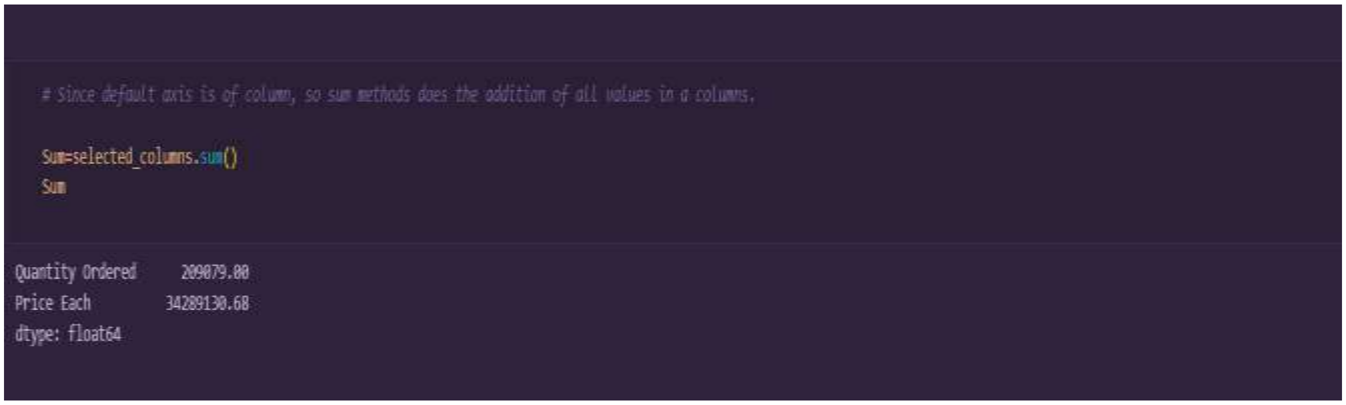
Figure 17 Summary statistics of chosen variable

I have chosen the quantity ordered and price each column and stored it in a selected column variable. Then using describe method, all the summary statistics can be obtained.

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

A. Finding sum of chose variable

Sum method in pandas gives the total values for the axis that was requested. To calculate the total of the columns, use axis=1 and for rows we can take the axis as 0.



```
# Since default axis is of column, so sum methods does the addition of all values in a columns.

Sum=selected_columns.sum()
Sum
```

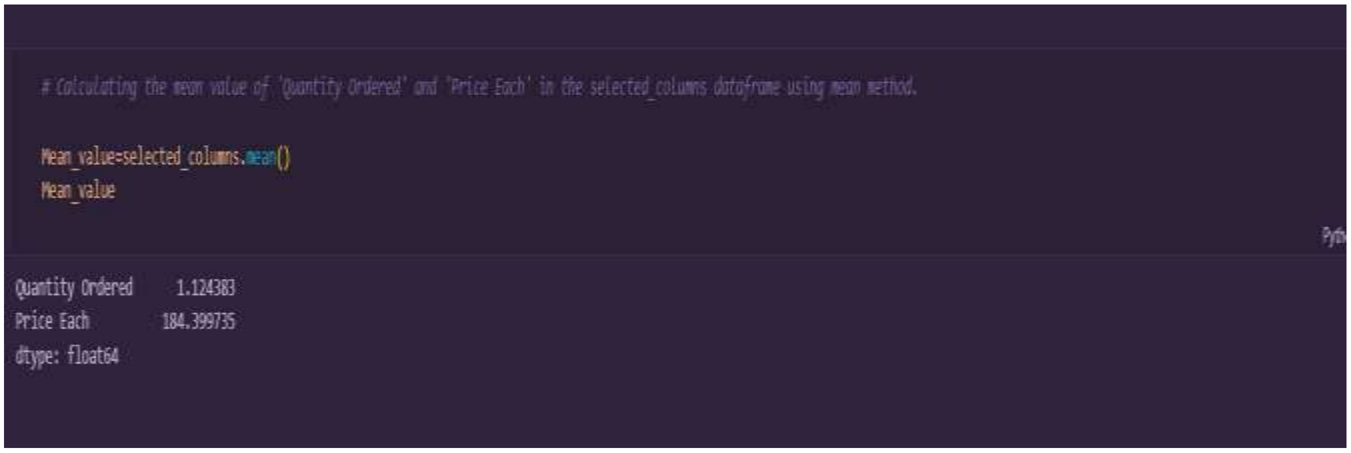
```
Quantity Ordered     209079.00
Price Each         34289130.68
dtype: float64
```

Figure 18 Use of sum method

Since, we have chosen two columns in a variable. We will receive the sum for that columns after using sum method.

B. Finding mean of chose variable

Mean tells us the average of a dataset, which can help us to understand the data and make predictions about the future. It is used for various purposes in business, finance, research, and everyday life.



```
# Calculating the mean value of 'Quantity Ordered' and 'Price Each' in the selected_columns dataframe using mean method.

Mean_value=selected_columns.mean()
Mean_value
```

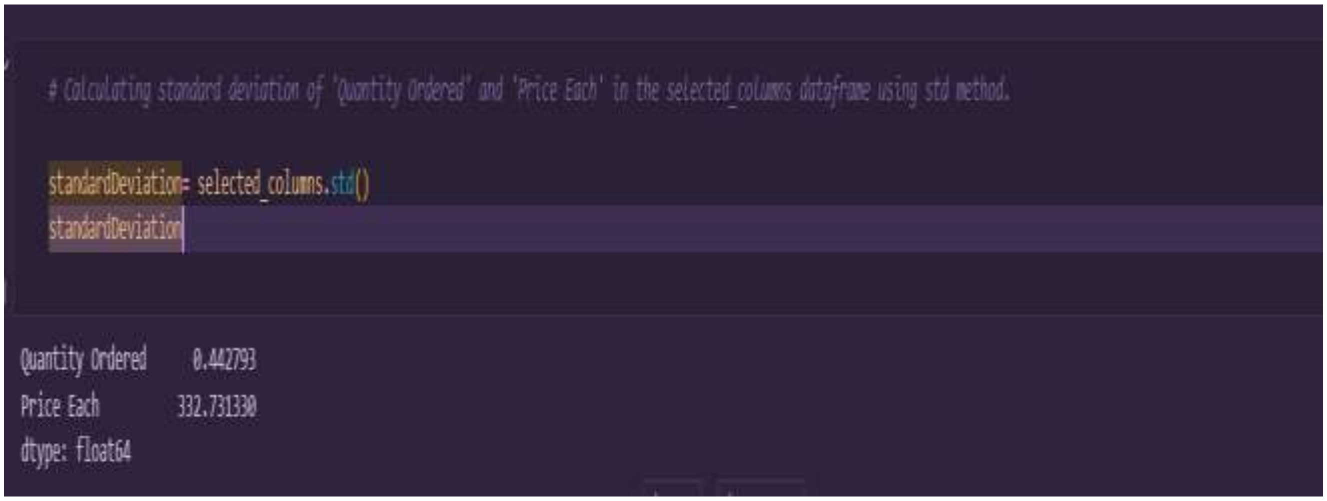
```
Quantity Ordered      1.124383
Price Each          184.399735
dtype: float64
```

Figure 19 Use of mean method

As, the default axis will be the one. So, on using mean method we will get the mean of all those columns that are selected.

C. Finding standard deviation of chose variable

Standard deviation is a measure of how dispersed the data is in relation to the mean. Low standard deviation means data are clustered around the mean, and high standard deviation indicates data are more spread out. A standard deviation close to zero indicates that data points are close to the mean, whereas a high or low standard deviation indicates data points are respectively above or below the mean.



```
# Calculating standard deviation of 'Quantity Ordered' and 'Price Each' in the selected_columns dataframe using std method.

standardDeviation= selected_columns.std()
standardDeviation
```

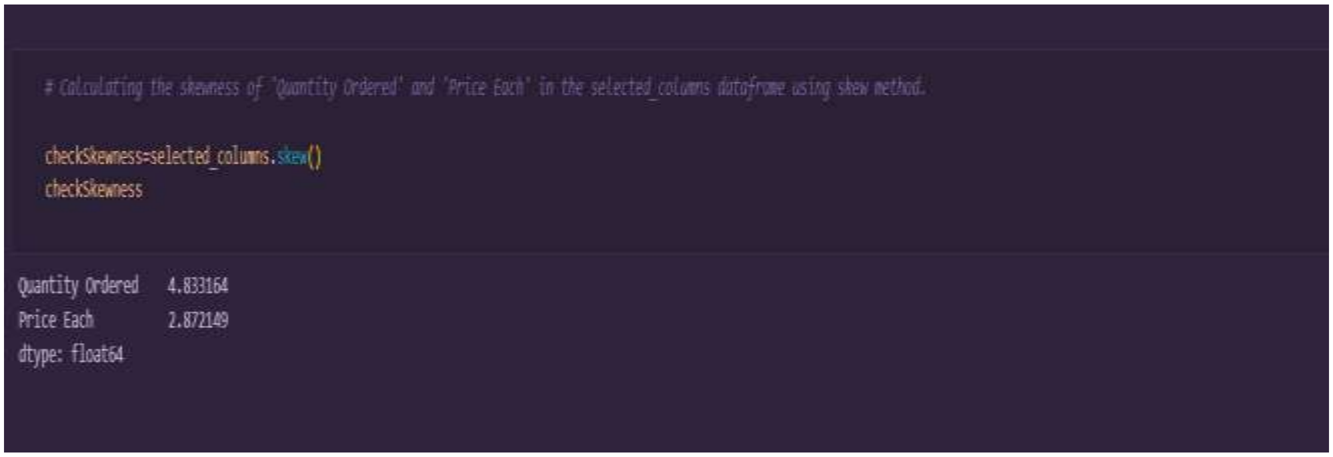
```
Quantity Ordered      0.442793
Price Each          332.731330
dtype: float64
```

Figure 20 Use of standard deviation method

Using std method we can get the standard deviation of the chosen variable.

D. Finding skewness of chose variable.

Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right of the center point (NIST, 2023).



```
# Calculating the skewness of 'Quantity Ordered' and 'Price Each' in the selected_columns dataframe using skew method.

checkSkewness=selected_columns.skew()
checkSkewness
```

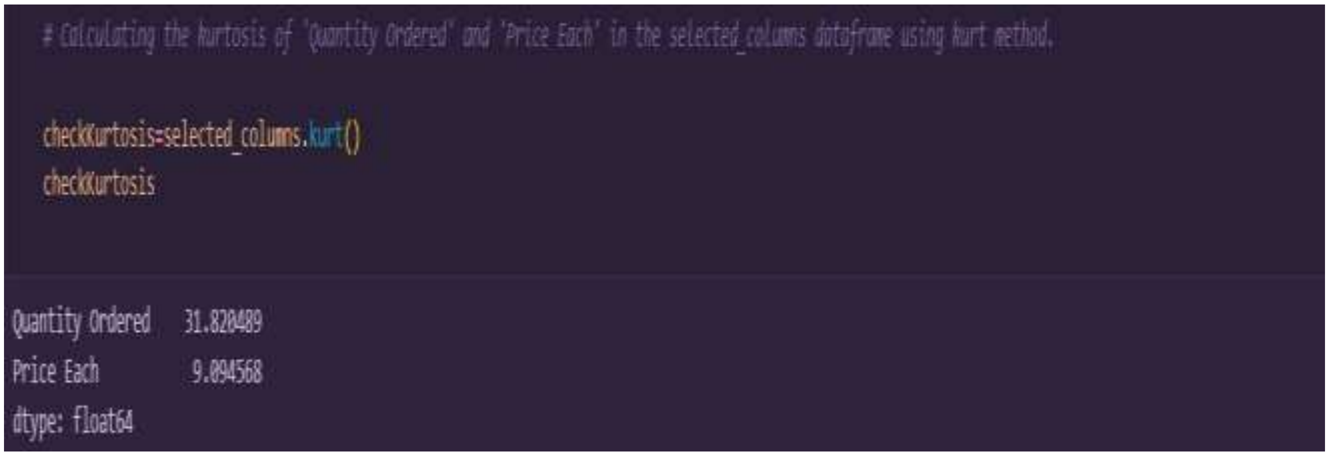
```
Quantity Ordered    4.833164
Price Each          2.872149
dtype: float64
```

Figure 21 Use of skew method

For finding skewness we use skew method for selected column.

E. Finding kurtosis of chose variable.

Kurtosis is a statistic that reflects how intense-tailed or light-tailed the data are in comparison to a normal distribution. In other words, data sets with a high kurtosis tend to have large outliers or heavy tails. Data sets with low kurtosis frequently lack outliers and have light tails. The worst-case scenario would be a uniform distribution.



```
# Calculating the kurtosis of 'Quantity Ordered' and 'Price Each' in the selected_columns dataframe using kurt method.

checkKurtosis=selected_columns.kurt()
checkKurtosis
```

```
Quantity Ordered    31.820489
Price Each           9.094568
dtype: float64
```

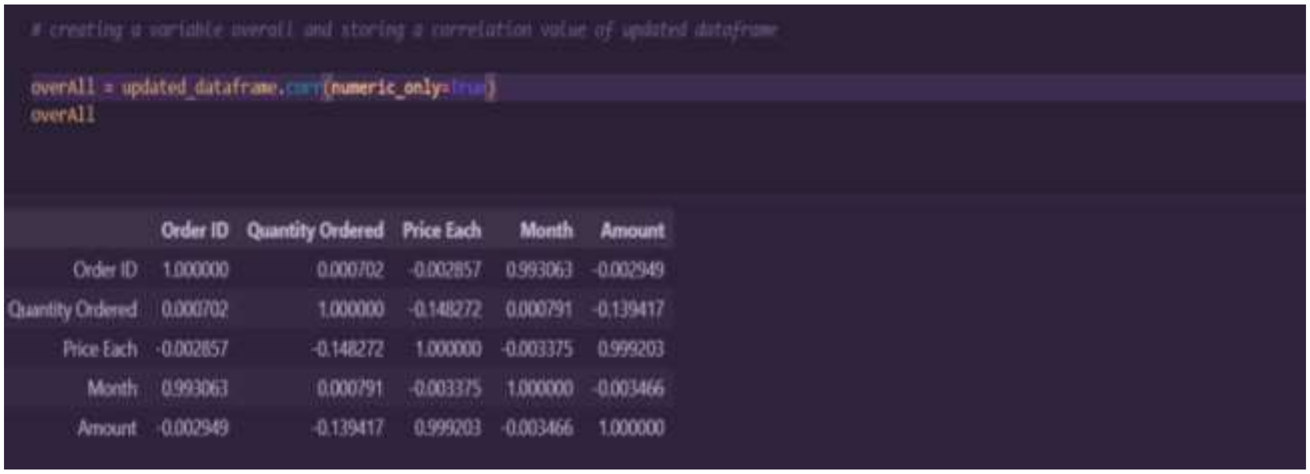
Figure 22 Use of kurtosis method

To find the kurtosis of our selected columns, we use kurt method in pandas.

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

3.2.1 Description of Correlation

A statistical measure called correlation shows how much two or more variables fluctuate in connection to one another. When two variables rise or decrease simultaneously, there is a positive correlation; when there is a negative correlation, one variable increases as the other falls (Singh, 2021).



```python
# creating a variable overall and storing a correlation value of updated dataframe

overAll = updated_dataframe.corr(numeric_only=True)
overAll
```

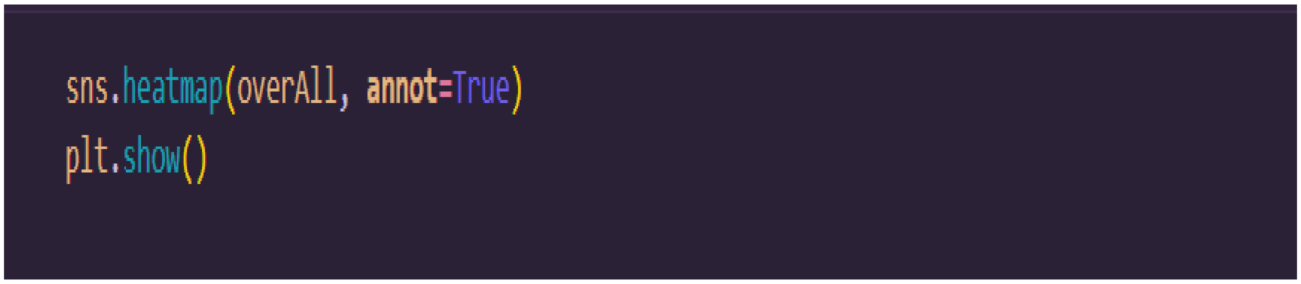
```
                  Order ID  Quantity Ordered  Price Each     Month    Amount
        Order ID  1.000000          0.000702   -0.002857  0.993063 -0.002949
Quantity Ordered  0.000702          1.000000   -0.148272  0.000791 -0.139417
      Price Each -0.002857         -0.148272    1.000000 -0.003375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

Figure 23 Correlation value of dataframe

Here for finding the correlation, we have to use corr method. Thus, for only numeric value we have to set parameter to True.

3.2.2 Heatmap for Correlation

Heatmap are commonly used to visualize the correlations between variables because they provide an easy-to-interpret graphical representation of the correlation matrix. A correlation matrix is a table that shows the correlation coefficients between all possible pairs of variables in a dataset.



```python
sns.heatmap(overAll, annot=True)
plt.show()
```

Figure 24 Use of heatmap from sns library

Using seaborn library, we have visualize heatmap for all variable in our dataframe and by taking parameter as overall and annot is set to true so that all the value can be displayed easily in the boxes.

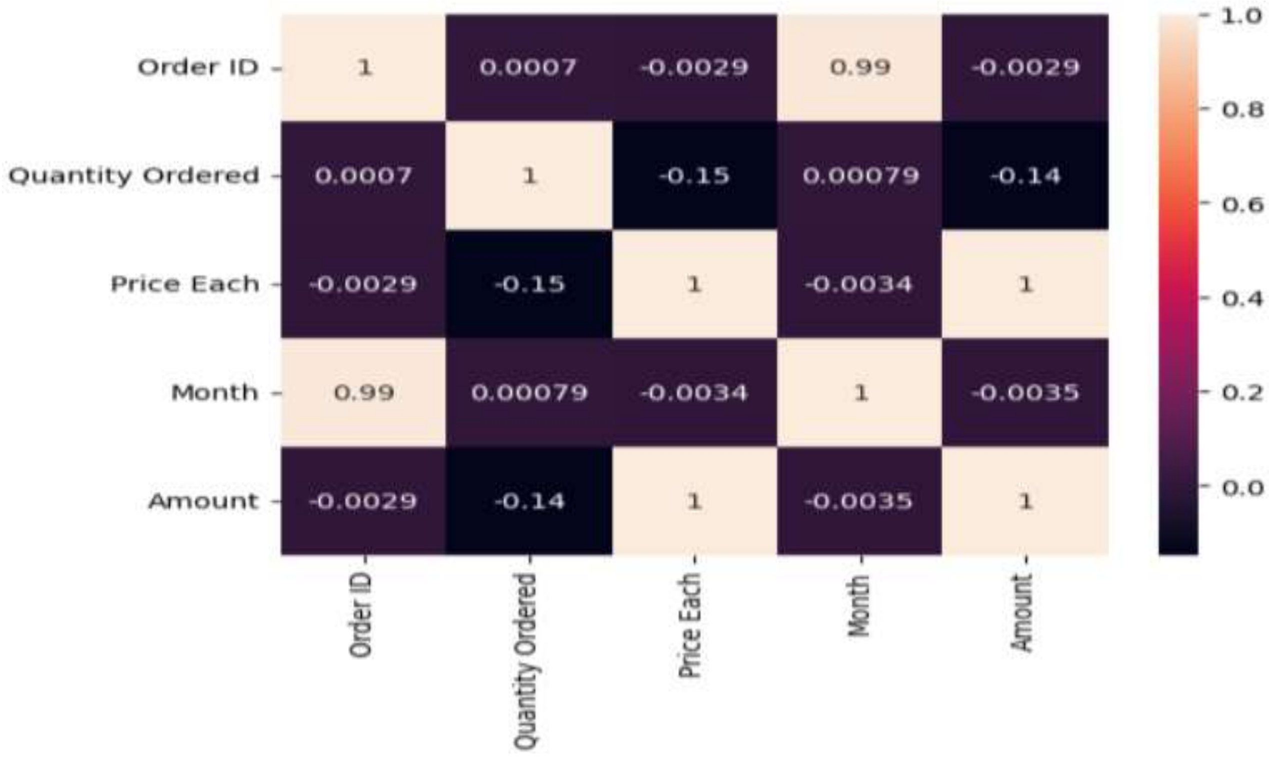


Figure 25 Representation of heatmap for all variable

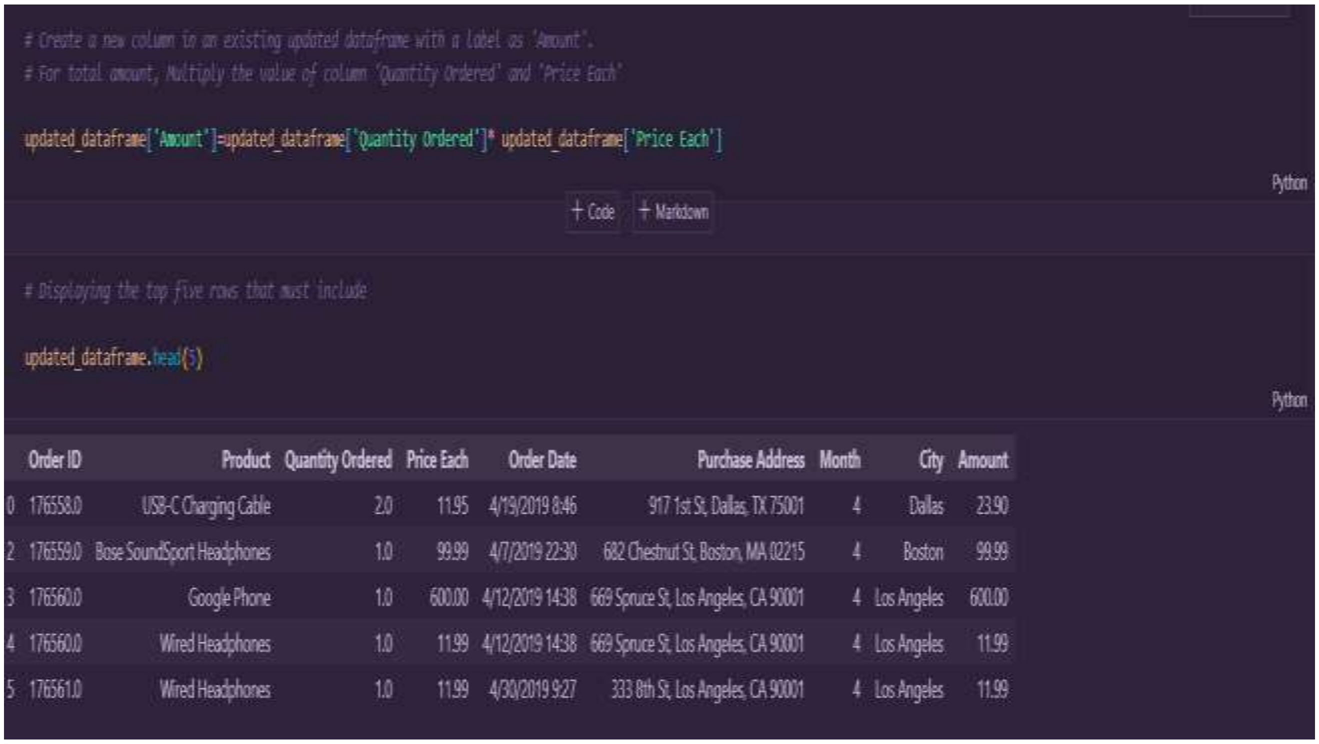
In this heatmap, we can see that there are two axes x axis and y axis with the value of each other comparing to next variable. Such as we can see the price each and month comparison as -0.0034. While, the diagonal value of all represents 1 since they are correlated to themselves. We can also see the bars at the right end with scale to 1 having a gap of 0.2 value. Then we can observe the color with light skin as the positive indication of value and purple or haze as the negative.

4. Data Exploration

Data analysts utilize statistical methods and data visualization to characterize dataset characterizations, such as size, number, and correctness, in order to better comprehend the nature of the data. Data exploration is the first stage of data analysis.

A dataset may be better understood through exploration, which also makes it simpler to explore and utilise the data in the future. The quality of an analyst's analysis will depend on how well they understand the data they are using. An open mind is necessary for successful research since it opens up new avenues for learning and aids in defining and addressing future analytics queries and issues.

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



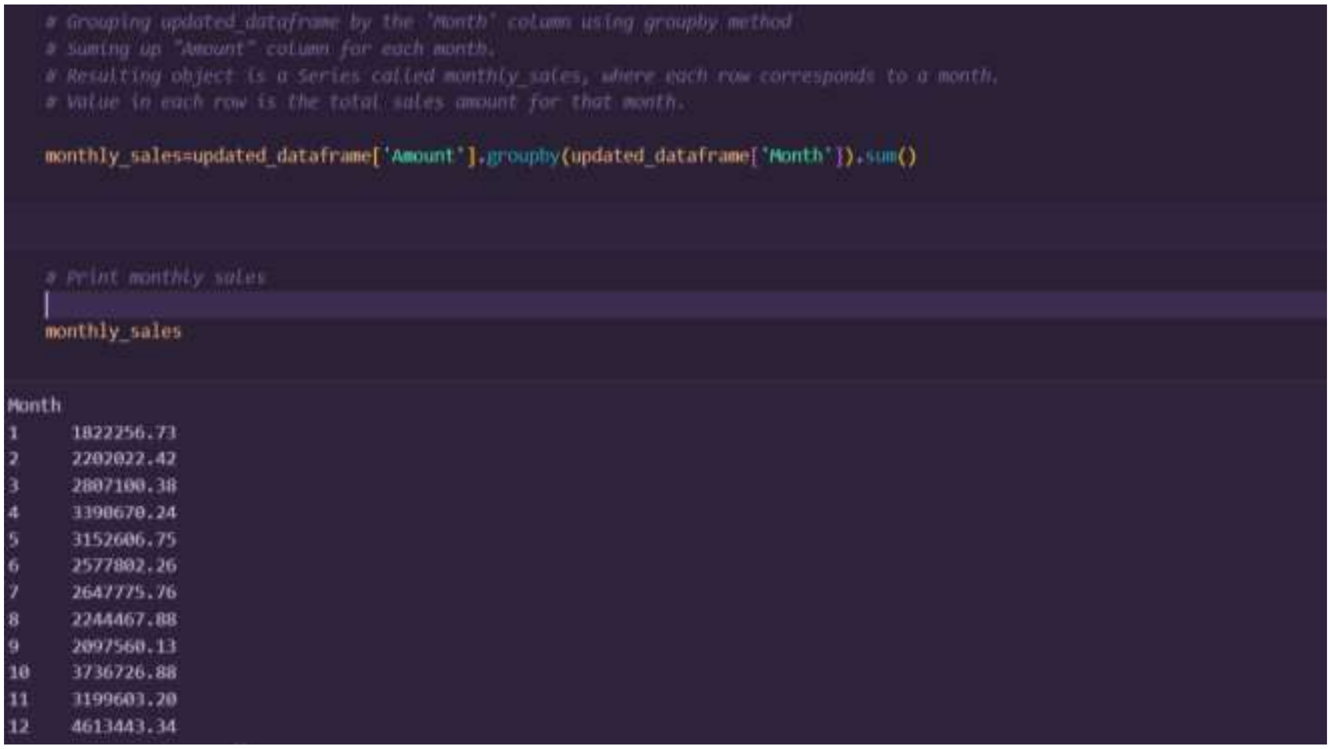
```python
# Create a new column in an existing updated dataframe with a label as 'Amount'.
# For total amount, Multiply the value of column 'Quantity Ordered' and 'Price Each'

updated_dataframe['Amount']=updated_dataframe['Quantity Ordered']* updated_dataframe['Price Each']
```

```python
# Displaying the top five rows that must include

updated_dataframe.head(5)
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City	Amount
0	176558.0	USB-C Charging Cable	2.0	11.95	4/19/2019 8:46	917 1st St, Dallas, TX 75001	4	Dallas	23.90
2	176559.0	Bose SoundSport Headphones	1.0	99.99	4/7/2019 22:30	682 Chestnut St, Boston, MA 02215	4	Boston	99.99
3	176560.0	Google Phone	1.0	600.00	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles	600.00
4	176560.0	Wired Headphones	1.0	11.99	4/12/2019 14:38	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles	11.99
5	176561.0	Wired Headphones	1.0	11.99	4/30/2019 9:27	333 8th St, Los Angeles, CA 90001	4	Los Angeles	11.99

Figure 26 Creating column named Amount in a dataframe

Here, we first calculated the total amount simply by multiplying quantity ordered and price each. Then, store that as variable and finally as a column in our existing dataframe. Then, we displayed the first five rows and checked if the column named amount is created.

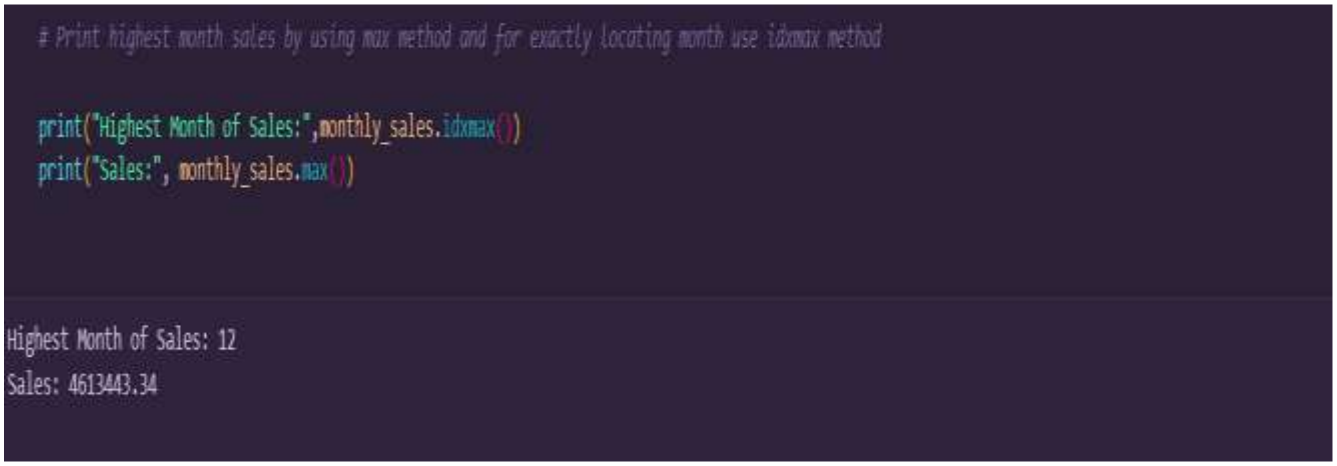


```
# Grouping updated_dataframe by the 'Month' column using groupby method
# Suming up "Amount" column for each month.
# Resulting object is a Series called monthly_sales, where each row corresponds to a month.
# Value in each row is the total sales amount for that month.

monthly_sales=updated_dataframe['Amount'].groupby(updated_dataframe['Month']).sum()
```

```
# Print monthly sales

monthly_sales
```

```
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
```

Figure 27 Calculation of monthly sales

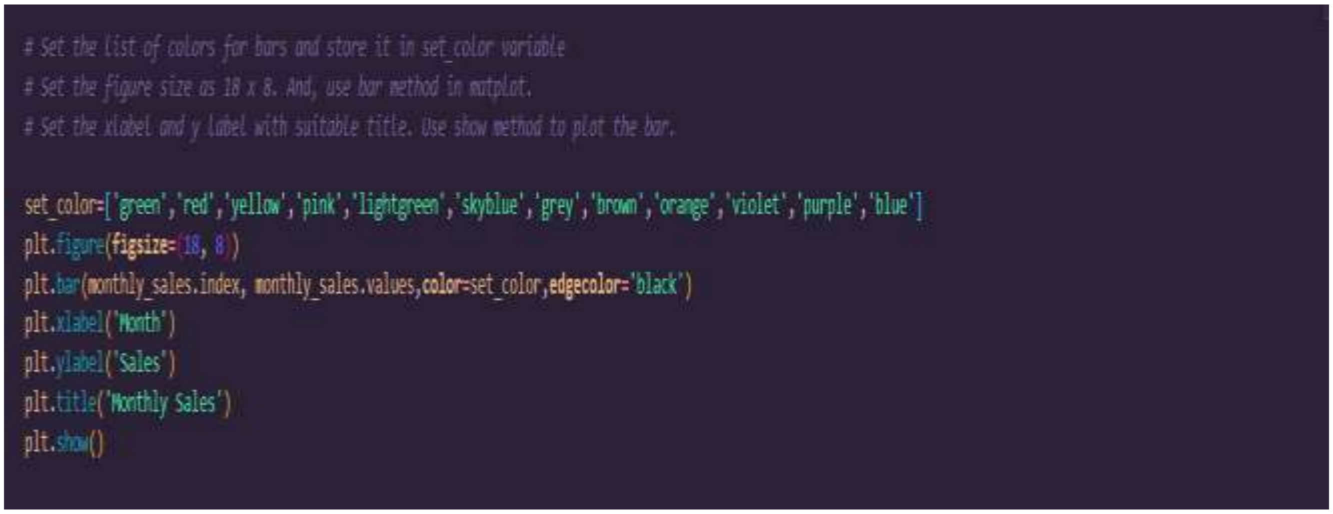


```
# Print highest month sales by using max method and for exactly locating month use idxmax method

print("Highest Month of Sales:",monthly_sales.idxmax())
print("Sales:", monthly_sales.max())
```

```
Highest Month of Sales: 12
Sales: 4613443.34
```

Figure 28 Printing highest month of sales with their month number

After that on using group by function, we grouped the month on the basis of total amount that results us in total monthly sales after we use sum method. For printing the month with highest sales, idxmax method is used and max method for highest monthly sales.



```
# Set the list of colors for bars and store it in set_color variable
# Set the figure size as 18 x 8. And, use bar method in matplot.
# Set the xlabel and y label with suitable title. Use show method to plot the bar.

set_color=['green','red','yellow','pink','lightgreen','skyblue','grey','brown','orange','violet','purple','blue']
plt.figure(figsize=(18, 8))
plt.bar(monthly_sales.index, monthly_sales.values,color=set_color,edgecolor='black')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.title('Monthly Sales')
plt.show()
```

Figure 29 Code for barplot

Here, for barplot we have to set the bar color, figure size, x-label, y-label and title and then draw the bar graph.

Bargraph for data visualization

Use bar graphs/charts to represent comparisons among discrete categories visually when categories are qualitative. Such categorical data is a grouping of data into discrete groups, such as the months of the year, age group, shoe sizes, and animals. Also use bar charts as an alternative to column charts for showing a larger set of data, where the vertical alignment of labels give more space for text to be easily read for each category (Think Design, 2022).

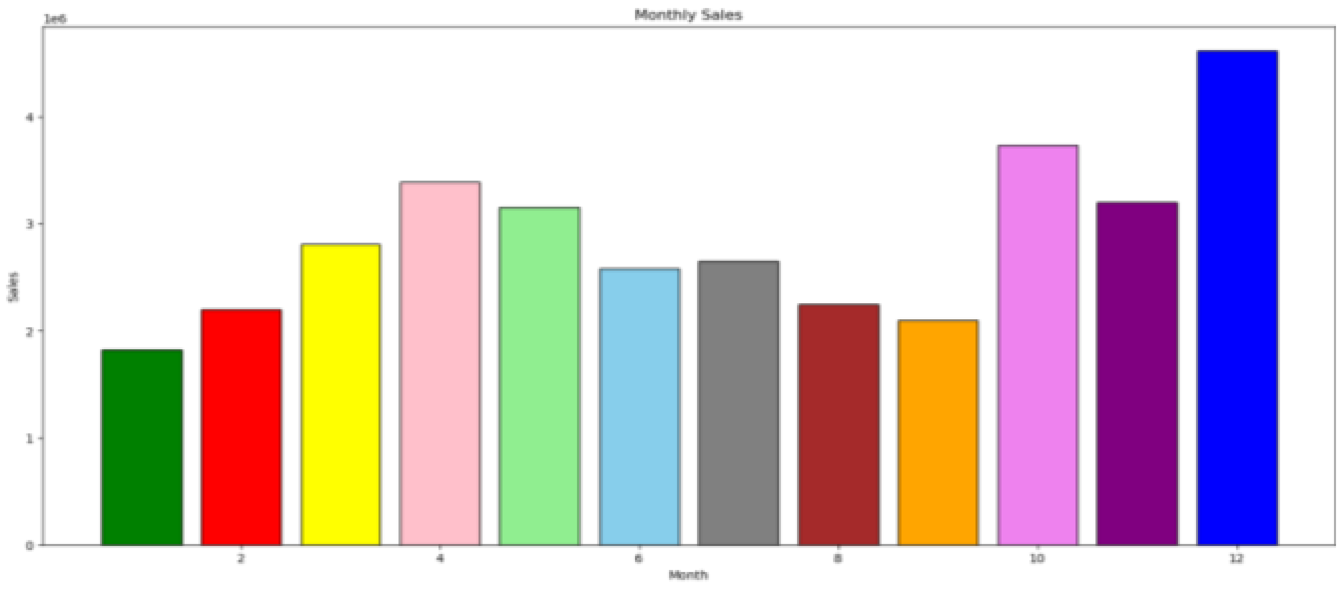
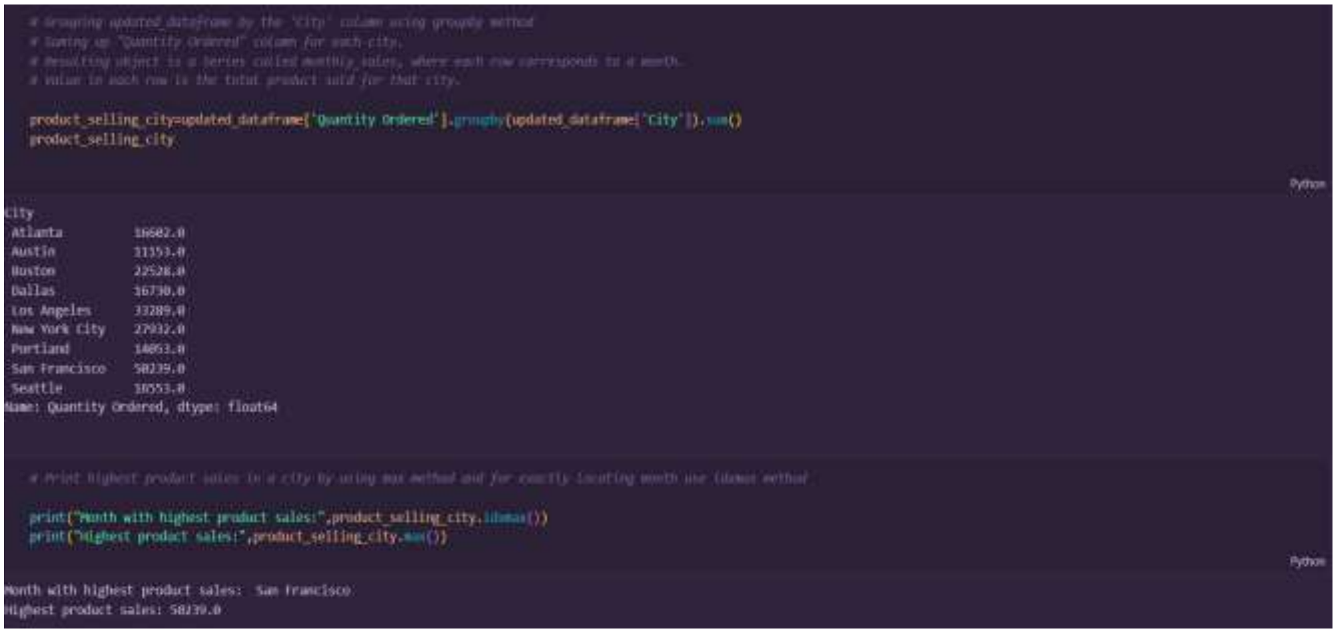


Figure 30 Barplot for monthly sales

In this barplot, we have set month as the x-axis value and sales as the y axis where we have set different color. Title is set as Monthly sales here. Looking to the graph we can say that 12th month had the best monthly sales while the very 1st month of the year had the least sales. And, depending upon different bars we can visualize our data clearly.

4.2 Which city has sold the highest product?



```
# Grouping updated_dataframe by the 'City' column using groupby method
# Summing up "Quantity Ordered" column for each city.
# Resulting object is a Series called monthly_sales, where each row corresponds to a month.
# Value in each row is the total product sold for that city.

product_selling_city=updated_dataframe['Quantity Ordered'].groupby(updated_dataframe['City']).sum()
product_selling_city
```

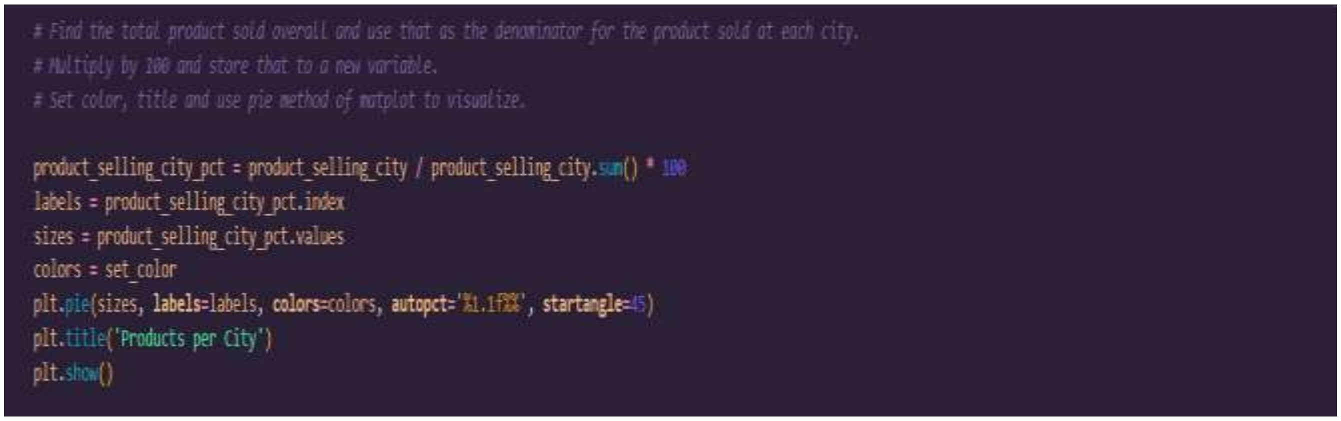
```
City
 Atlanta          16602.0
 Austin           11153.0
 Boston           22528.0
 Dallas           16730.0
 Los Angeles      33289.0
 New York City    27932.0
 Portland         14053.0
 San Francisco    50239.0
 Seattle          16553.0
Name: Quantity Ordered, dtype: float64
```

```
# Print highest product sales in a city by using max method and for exactly locating month use idxmax method

print("Month with highest product sales:",product_selling_city.idxmax())
print("Highest product sales:",product_selling_city.max())
```

```
Month with highest product sales:  San Francisco
Highest product sales: 50239.0
```

Figure 31 Month with highest sales

Firstly we have to groupby quantity ordered and city then use sum to find total quantity for those city. After that, for displaying city with highest quantity we can use idxmax method and max method for the value.



```
# Find the total product sold overall and use that as the denominator for the product sold at each city.
# Multiply by 100 and store that to a new variable.
# Set color, title and use pie method of matplot to visualize.

product_selling_city_pct = product_selling_city / product_selling_city.sum() * 100
labels = product_selling_city_pct.index
sizes = product_selling_city_pct.values
colors = set_color
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=45)
plt.title('Products per City')
plt.show()
```

Figure 32 Code for pie chart representing city

Since I have to use pie chart to display the city with highest number of products sold. So, first find the total product sold and use that as denominator for each slice of products that was sold in that city. Then, assigning index and values to the main variable product selling city. After, that we use pie method which takes different parameter ranging from color, axis to start angle.

Pie chart to visualize city with highest products sold

A pie chart is a visual representation of data that looks like a pie with the slices representing the size of the data. To show data as a pie chart, a list of numerical variables and category variables are required. Each slice in a pie chart has an arc that is proportionate to the quantity it depicts, and as a result, the area and center angle it generates are also proportional.

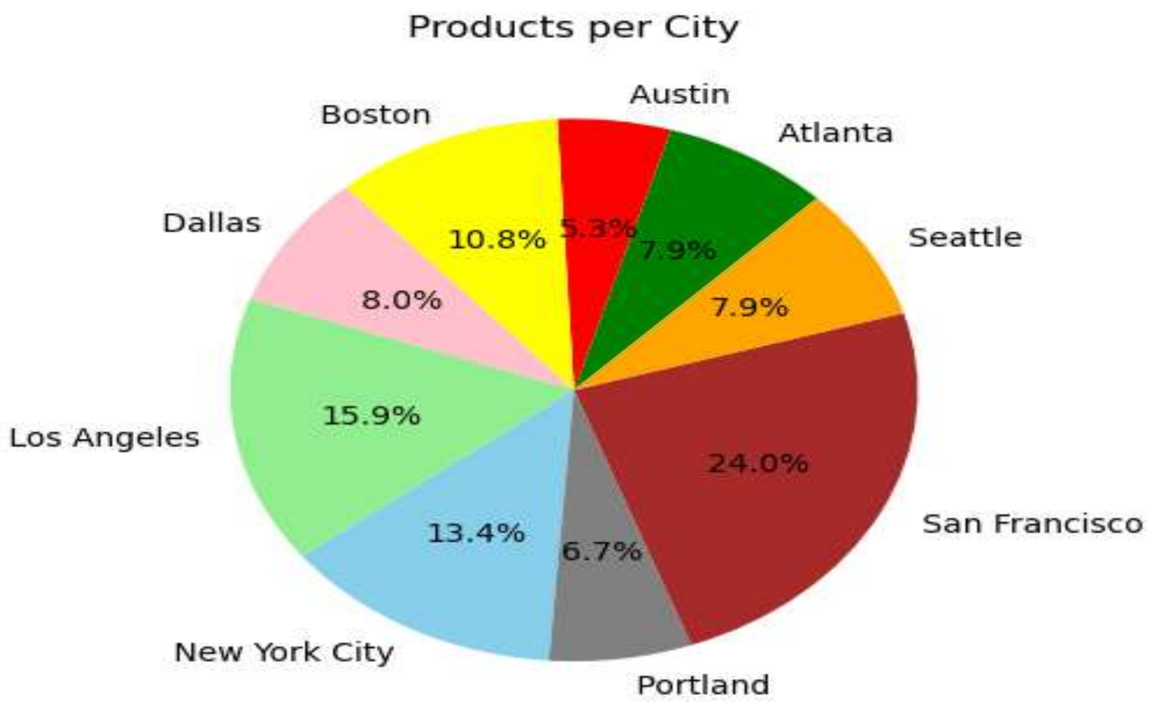
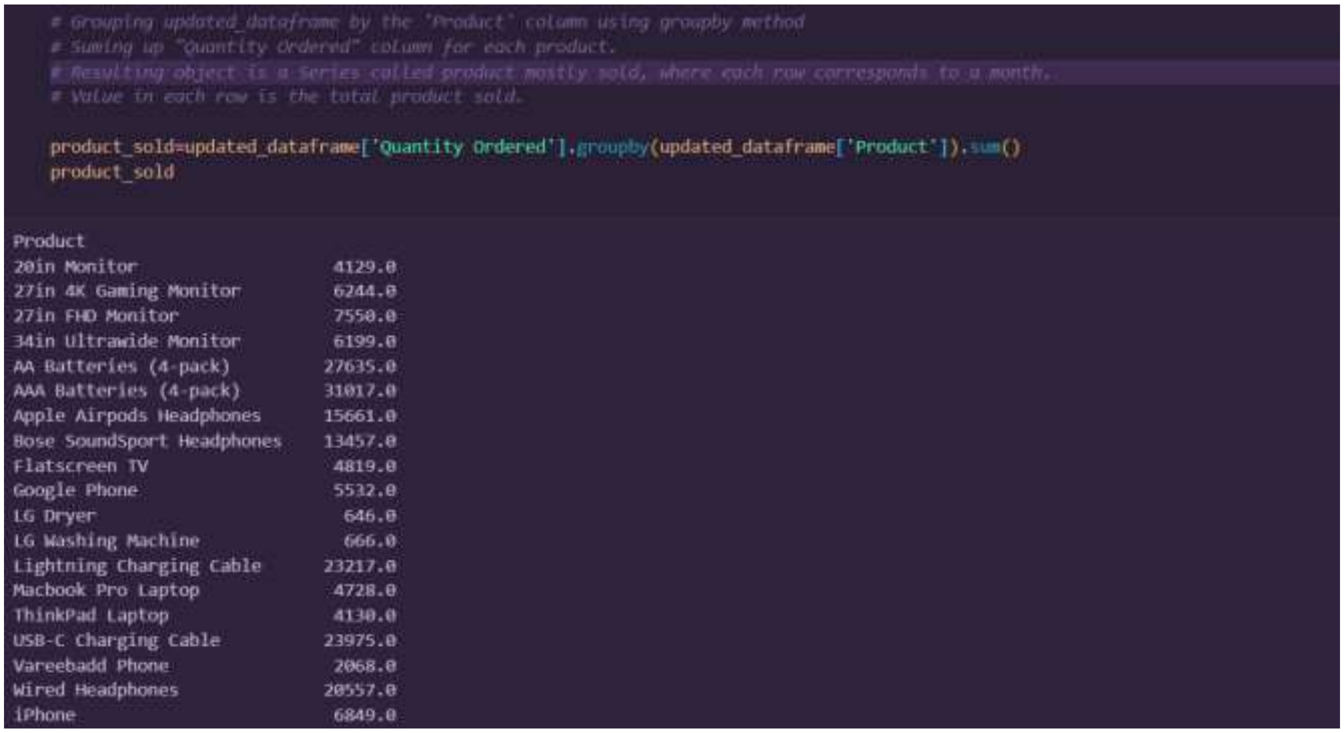


Figure 33 Pie chart for products sold by city

From this pie chart, we can easily visualize the city with most products sold which is San Francisco with 24 percent of the overall products then in second its Los Angeles with 15.9 percentage. And the least of it is contributed by Austin with 5.3 percentage. This implies that percentage of product sold by the city.

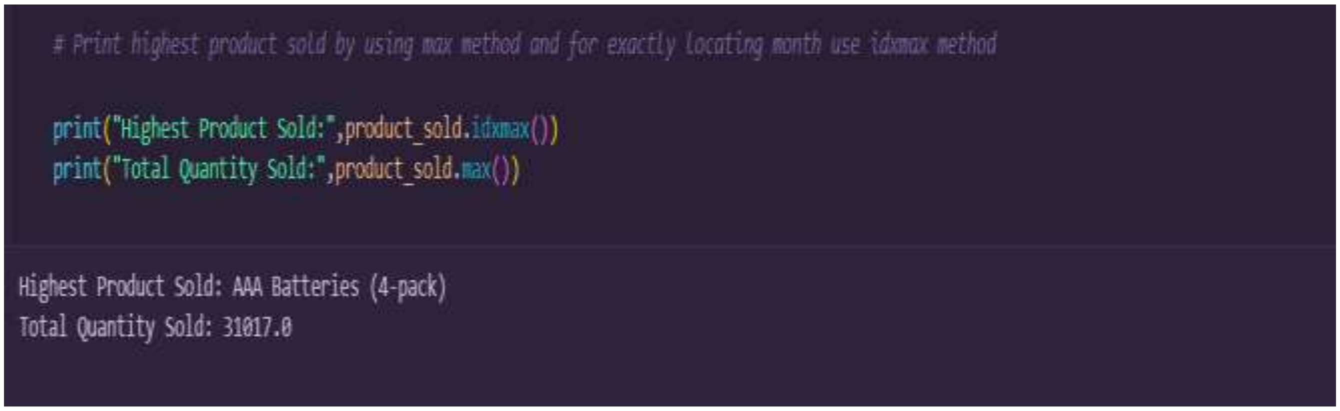
4.3 Which product was sold the most in overall? Illustrate it through bar graph.



```
# Grouping updated_dataframe by the 'Product' column using groupby method
# Suming up "Quantity Ordered" column for each product.
# Resulting object is a Series called product mostly sold, where each row corresponds to a month.
# Value in each row is the total product sold.

product_sold=updated_dataframe['Quantity Ordered'].groupby(updated_dataframe['Product']).sum()
product_sold
```

```
Product
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
```

Here, quantity ordered is grouped with product and then each product sold quantity is displayed with the use of sum method.



```
# Print highest product sold by using max method and for exactly locating month use idxmax method

print("Highest Product Sold:",product_sold.idxmax())
print("Total Quantity Sold:",product_sold.max())
```

```
Highest Product Sold: AAA Batteries (4-pack)
Total Quantity Sold: 31017.0
```

Then, we can display the name of product that was highly sold using idxmax method and quantity as max method.

Bargraph to visualize mostly sold products

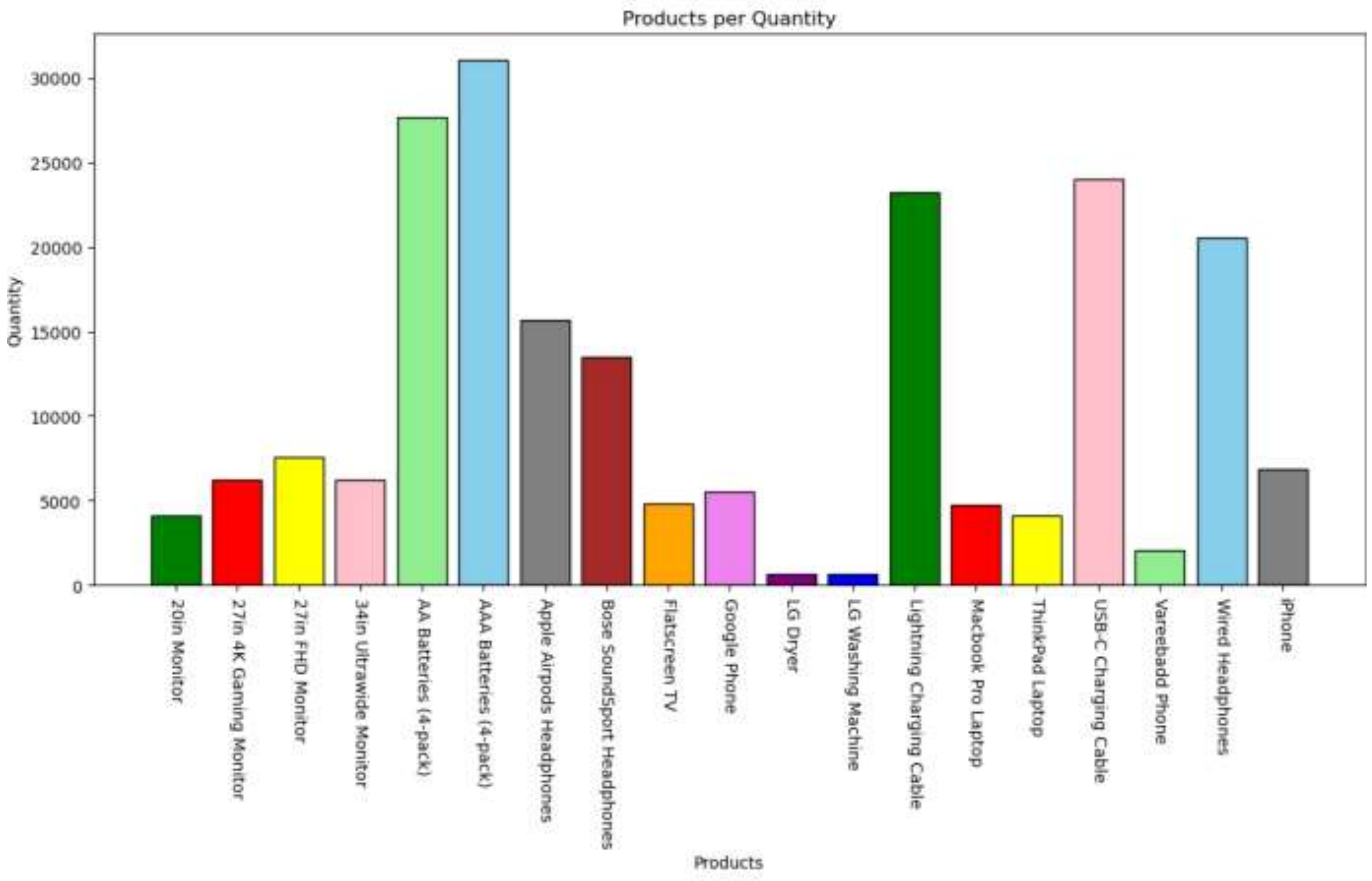
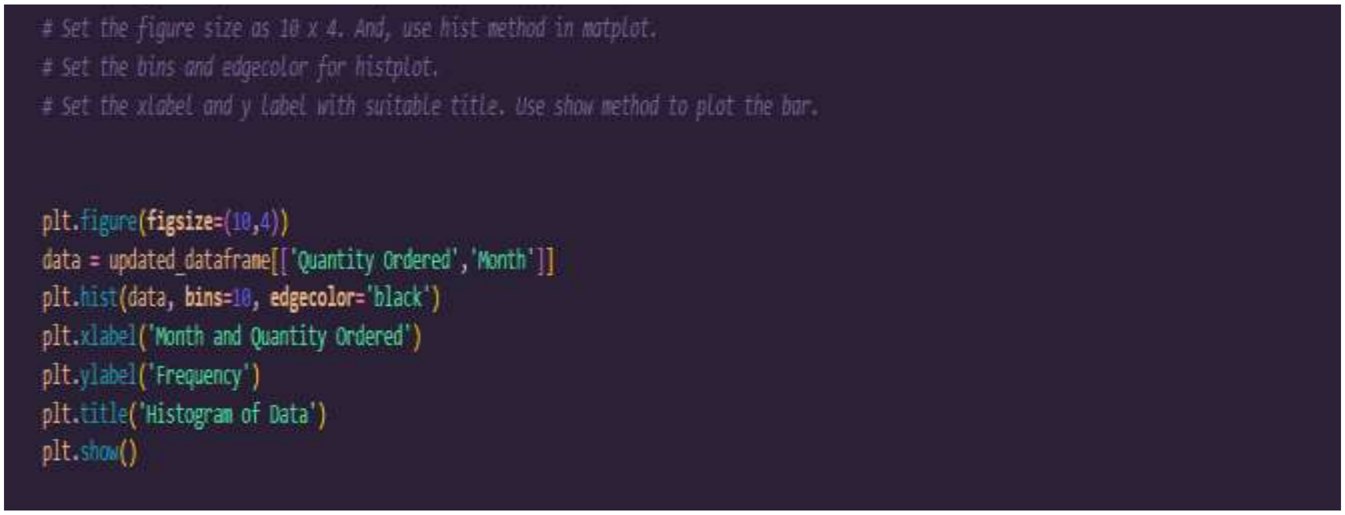


Figure 34 Bar graph to find overall product sold.

Since AAA Batteries with 4 pack was highly sold, its bar is highest among all. The lowest sold products were the LG dryer and LG Washing Machine as the bar goes least in them. And, other different bars have been set to visualize our data properly. Similarly we have x axis for products and y for quantity of particular products sold.

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



```python
# Set the figure size as 10 x 4. And, use hist method in matplot.
# Set the bins and edgecolor for histplot.
# Set the xlabel and y label with suitable title. Use show method to plot the bar.



plt.figure(figsize=(10,4))
data = updated_dataframe[['Quantity Ordered','Month']]
plt.hist(data, bins=10, edgecolor='black')
plt.xlabel('Month and Quantity Ordered')
plt.ylabel('Frequency')
plt.title('Histogram of Data')
plt.show()
```

Figure 35 Code for histogram plot for chosen variable

Firstly, we have to select the variable. Here, we have used two variables for the visualization purpose. Similar to other, we have to set the figure size here and hist method of matplot library. Then we have to set x label, y label, title and in the parameter for hist we have to use bins according to our choice. Finally setting edge color is also done here.

Histogram plot to visualize any variable

Histograms are a type of graph that is used to represent the distribution of a dataset. They are commonly used to visualize the frequency distribution of a continuous variable. The x-axis of a histogram represents the range of values of the variable being measured, and the y-axis represents the frequency of the observations falling within each bin (or interval) on the x-axis.

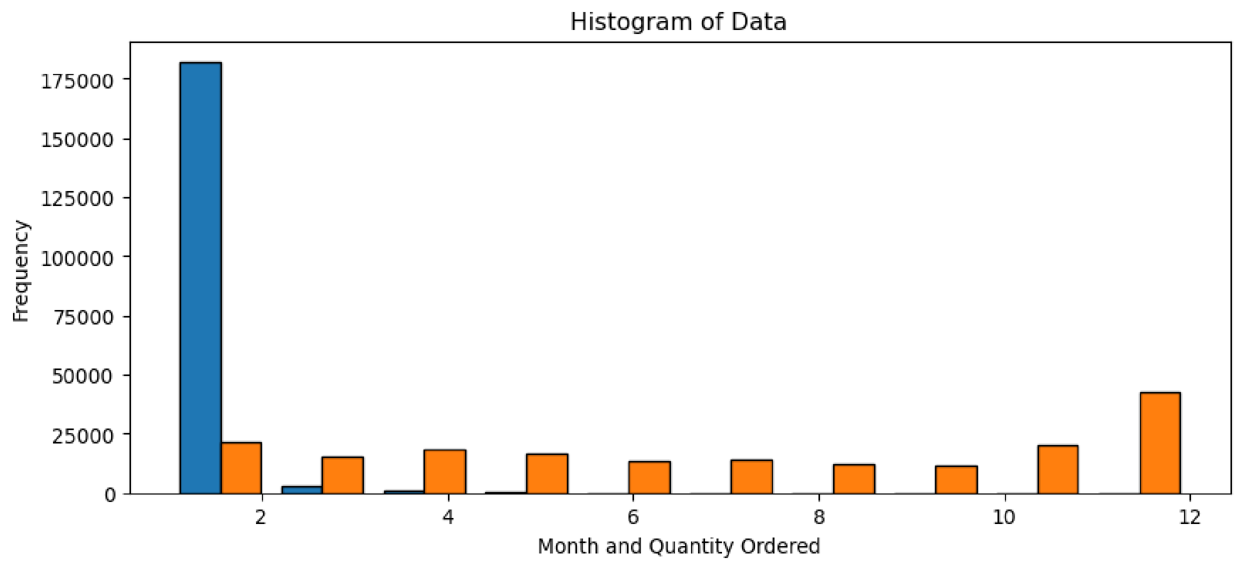


Figure 36 Histogram plot for month and quantity ordered

In this histogram plot, we can observe two attached bars with orange and blue color. Here, the blue color bars represent the quantity ordered and orange one as month. Then looking for the frequency of each data we can state the number of quantity ordered between 0 and 2 seems to be maximum while between 2 and 4 have lesser appearance and finally after 4 to 6 it seems to disappear since no customer have ordered quantity more than 4 to 6. For the months, there are every bars present and maximum is between 10 to 12 while minimum is between 8 to 10.

Bibliography

Bandgar, S., 2021. Data Preparation. *Data Preparation in Data Science*, 28 May.

Dikshamulchandani, 2023. Data Understanding. *Data Science Lifecycle*, 20 Feb.

Johnson, D., 2023. Data Analysis. *What is Data Analysis? Research, Types & Example,* pp. 2-3 pages.

Nelli, F., 2015. *Python Data Analytics.* s.l.: Library of Congress.

NIST, 2023. *nist.com.* [Online]
Available at: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm#

Siman, E., 2022. Web Data.

Singh, S., 2021. *Correlation in statistics.* California: HarperOne.

Think Design, 2022. *think.design.* [Online]
Available at: https://think.design/services/data-visualization-data-design/bar-chart/#:
[Accessed 02 05 2023].