**Phase 3 Submission Document**

**Project Title:** covid-19 Analysis

**Phase 3 : *Development part 1***

# **Data link:**[**https://www.kaggle.com/datasets/ksabishek/product-sales-data**](https://www.kaggle.com/datasets/ksabishek/product-sales-data)

# Import Necessary Libraries

In [1]:

import pandas as pd

import os

import numpy as np

import seaborn as sns

import warnings

from matplotlib import pylab as plt

from statsmodels.graphics.gofplots import qqplot

from IPython.core.interactiveshell import InteractiveShell

### **Covid-19 into a single CSV file**

*# let's make a list compreension for all the data in the folder*

files = [file for file **in** os.listdir('../input/covid-19-')]

*# let's make a pandas DataFrame*

all\_months\_data = pd.DataFrame()

*# makes a loop for concat the data*

for file **in** files:

data = pd.read\_csv("../input/covid-19/" + file)

all\_months\_data = pd.concat([all\_months\_data, data])

*# export all data to csv*

all\_months\_data.to\_csv("all\_data.csv", index=False)

# Read in updated DataFrame

Let's see the data and how it looks.

sales\_data = pd.read\_csv('all\_data.csv') *# read data*

sales\_data *# show data*

|  | Order ID | Product | Quantity Ordered | Price Each | Order Date | Purchase Address |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 236670 | Wired Headphones | 2 | 11.99 | 08/31/19 22:21 | 359 Spruce St, Seattle, WA 98101 |
| 1 | 236671 | Bose SoundSport Headphones | 1 | 99.99 | 08/15/19 15:11 | 492 Ridge St, Dallas, TX 75001 |
| 2 | 236672 | iPhone | 1 | 700.0 | 08/06/19 14:40 | 149 7th St, Portland, OR 97035 |
| 3 | 236673 | AA Batteries (4-pack) | 2 | 3.84 | 08/29/19 20:59 | 631 2nd St, Los Angeles, CA 90001 |
| 4 | 236674 | AA Batteries (4-pack) | 2 | 3.84 | 08/15/19 19:53 | 736 14th St, New York City, NY 10001 |
| ... | ... | ... | ... | ... | ... | ... |
| 186845 | 319666 | Lightning Charging Cable | 1 | 14.95 | 12/11/19 20:58 | 14 Madison St, San Francisco, CA 94016 |
| 186846 | 319667 | AA Batteries (4-pack) | 2 | 3.84 | 12/01/19 12:01 | 549 Willow St, Los Angeles, CA 90001 |
| 186847 | 319668 | Vareebadd Phone | 1 | 400 | 12/09/19 06:43 | 273 Wilson St, Seattle, WA 98101 |
| 186848 | 319669 | Wired Headphones | 1 | 11.99 | 12/03/19 10:39 | 778 River St, Dallas, TX 75001 |
| 186849 | 319670 | Bose SoundSport Headphones | 1 | 99.99 | 12/21/19 21:45 | 747 Chestnut St, Los Angeles, CA 90001 |

"Head"

sales\_data.head() *# Checking the first 5 rows of data*

"Tail"

sales\_data.tail() *# Checking the last 5 rows of data*

Out[5]:

'Head'

Out[5]:

|  | Order ID | Product | Quantity Ordered | Price Each | Order Date | Purchase Address |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 236670 | Wired Headphones | 2 | 11.99 | 08/31/19 22:21 | 359 Spruce St, Seattle, WA 98101 |
| 1 | 236671 | Bose SoundSport Headphones | 1 | 99.99 | 08/15/19 15:11 | 492 Ridge St, Dallas, TX 75001 |
| 2 | 236672 | iPhone | 1 | 700.0 | 08/06/19 14:40 | 149 7th St, Portland, OR 97035 |
| 3 | 236673 | AA Batteries (4-pack) | 2 | 3.84 | 08/29/19 20:59 | 631 2nd St, Los Angeles, CA 90001 |
| 4 | 236674 | AA Batteries (4-pack) | 2 | 3.84 | 08/15/19 19:53 | 736 14th St, New York City, NY 10001 |

Out[5]:

'Tail'

Out[5]:

|  | Order ID | Product | Quantity Ordered | Price Each | Order Date | Purchase Address |
| --- | --- | --- | --- | --- | --- | --- |
| 186845 | 319666 | Lightning Charging Cable | 1 | 14.95 | 12/11/19 20:58 | 14 Madison St, San Francisco, CA 94016 |
| 186846 | 319667 | AA Batteries (4-pack) | 2 | 3.84 | 12/01/19 12:01 | 549 Willow St, Los Angeles, CA 90001 |
| 186847 | 319668 | Vareebadd Phone | 1 | 400 | 12/09/19 06:43 | 273 Wilson St, Seattle, WA 98101 |
| 186848 | 319669 | Wired Headphones | 1 | 11.99 | 12/03/19 10:39 | 778 River St, Dallas, TX 75001 |
| 186849 | 319670 | Bose SoundSport Headphones | 1 | 99.99 | 12/21/19 21:45 | 747 Chestnut St, Los Angeles, CA 90001 |

# Data Preprocessing

Data preprocessing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance, and is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects.

In [6]:

*# getting the information*

sales\_data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 186850 entries, 0 to 186849

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Order ID 186305 non-null object

1 Product 186305 non-null object

2 Quantity Ordered 186305 non-null object

3 Price Each 186305 non-null object

4 Order Date 186305 non-null object

5 Purchase Address 186305 non-null object

dtypes: object(6)

memory usage: 8.6+ MB

### **Uniqueness Categorical Variables**

Let's have a look at categorical variables. How many unique values of these variables.

In [7]:

categorical = sales\_data.select\_dtypes(['category', 'object']).columns *# getting the Uniqueness catrgorical variable*

for col **in** categorical:

print('**{}** : **{}** unique value(s)'.format(col, sales\_data[col].nunique()))

Order ID : 178438 unique value(s)

Product : 20 unique value(s)

Quantity Ordered : 10 unique value(s)

Price Each : 24 unique value(s)

Order Date : 142396 unique value(s)

Purchase Address : 140788 unique value(s)

### **missing data points**

Ok, now we know that we do have some missing values. Let's see how many we have in each column.

In [8]:

*# get the number of missing data points per column*

missing\_values\_count = sales\_data.isnull().sum()

*# look at the # of missing points in the first ten columns*

missing\_values\_count[0:10]

Out[8]:

Order ID 545

Product 545

Quantity Ordered 545

Price Each 545

Order Date 545

Purchase Address 545

dtype: int64

It might be helpful to see what percentage of the values in our dataset were missing to give us a better sense of the scale of this problem:

In [9]:

*# how many total missing values do we have?*

total\_cells = np.product(sales\_data.shape)

total\_missing = missing\_values\_count.sum()

*# percent of data that is missing*

percent\_missing = (total\_missing / total\_cells) \* 100

print(f"**{**percent\_missing**:**.2f**}**%")

0.29%

Looks like the percent missing of the data is not too big.

since the data is big enough.

### **Clean up the Data!**

The first step in this is figuring out what we need to clean. I have found in practice, that you find things you need to clean as you perform operations and get errors. Based on the error, you decide how you should go about cleaning the data.

In [10]:

*# let's drop the rows of NaN data!*

sales\_data = sales\_data.dropna(how='all')

*# okay, let's check it again!*

"NaN Value:"

sales\_data[sales\_data.isna().any(axis=1)]

*# future warning! ValueError: invalid literal for int() with base 10: 'Or'*

"Clean Future Warnings:"

sales\_data = sales\_data[sales\_data['Order Date'].str[0:2] != 'Or']

sales\_data

Out[10]:

'NaN Value:'

Out[10]:

|  | Order ID | Product | Quantity Ordered | Price Each | Order Date | Purchase Address |
| --- | --- | --- | --- | --- | --- | --- |

Out[10]:

'Clean Future Warnings:'

Out[10]:

|  | Order ID | Product | Quantity Ordered | Price Each | Order Date | Purchase Address |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 236670 | Wired Headphones | 2 | 11.99 | 08/31/19 22:21 | 359 Spruce St, Seattle, WA 98101 |
| 1 | 236671 | Bose SoundSport Headphones | 1 | 99.99 | 08/15/19 15:11 | 492 Ridge St, Dallas, TX 75001 |
| 2 | 236672 | iPhone | 1 | 700.0 | 08/06/19 14:40 | 149 7th St, Portland, OR 97035 |
| 3 | 236673 | AA Batteries (4-pack) | 2 | 3.84 | 08/29/19 20:59 | 631 2nd St, Los Angeles, CA 90001 |
| 4 | 236674 | AA Batteries (4-pack) | 2 | 3.84 | 08/15/19 19:53 | 736 14th St, New York City, NY 10001 |
| ... | ... | ... | ... | ... | ... | ... |
| 186845 | 319666 | Lightning Charging Cable | 1 | 14.95 | 12/11/19 20:58 | 14 Madison St, San Francisco, CA 94016 |
| 186846 | 319667 | AA Batteries (4-pack) | 2 | 3.84 | 12/01/19 12:01 | 549 Willow St, Los Angeles, CA 90001 |
| 186847 | 319668 | Vareebadd Phone | 1 | 400 | 12/09/19 06:43 | 273 Wilson St, Seattle, WA 98101 |
| 186848 | 319669 | Wired Headphones | 1 | 11.99 | 12/03/19 10:39 | 778 River St, Dallas, TX 75001 |
| 186849 | 319670 | Bose SoundSport Headphones | 1 | 99.99 | 12/21/19 21:45 | 747 Chestnut St, Los Angeles, CA 90001 |

### **Convert Quantity Ordered column and Price Each column**

Let's convert the Quantity Ordered column and Price Each column to Numeric Type, because we will add some future features, and we need to multiply this two column.

In [11]:

*# convert the data*

sales\_data['Quantity Ordered'], sales\_data['Price Each'] = sales\_data['Quantity Ordered'].astype('int64'), sales\_data['Price Each'].astype('float')

*# and check it*

sales\_data.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 185950 entries, 0 to 186849

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Order ID 185950 non-null object

1 Product 185950 non-null object

2 Quantity Ordered 185950 non-null int64

3 Price Each 185950 non-null float64

4 Order Date 185950 non-null object

5 Purchase Address 185950 non-null object

dtypes: float64(1), int64(1), object(4)

memory usage: 9.9+ MB

### **Convert Order Date column**

And let's convert Order Date column too, so we can take the Year, Month, and the other date easily.

In [12]:

*# convert it using to\_datetime() funct*

sales\_data['Order Date'] = pd.to\_datetime(sales\_data['Order Date'])

*# let's see it*

sales\_data

Out[12]:

|  | Order ID | Product | Quantity Ordered | Price Each | Order Date | Purchase Address |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 236670 | Wired Headphones | 2 | 11.99 | 2019-08-31 22:21:00 | 359 Spruce St, Seattle, WA 98101 |
| 1 | 236671 | Bose SoundSport Headphones | 1 | 99.99 | 2019-08-15 15:11:00 | 492 Ridge St, Dallas, TX 75001 |
| 2 | 236672 | iPhone | 1 | 700.00 | 2019-08-06 14:40:00 | 149 7th St, Portland, OR 97035 |
| 3 | 236673 | AA Batteries (4-pack) | 2 | 3.84 | 2019-08-29 20:59:00 | 631 2nd St, Los Angeles, CA 90001 |
| 4 | 236674 | AA Batteries (4-pack) | 2 | 3.84 | 2019-08-15 19:53:00 | 736 14th St, New York City, NY 10001 |
| ... | ... | ... | ... | ... | ... | ... |
| 186845 | 319666 | Lightning Charging Cable | 1 | 14.95 | 2019-12-11 20:58:00 | 14 Madison St, San Francisco, CA 94016 |
| 186846 | 319667 | AA Batteries (4-pack) | 2 | 3.84 | 2019-12-01 12:01:00 | 549 Willow St, Los Angeles, CA 90001 |
| 186847 | 319668 | Vareebadd Phone | 1 | 400.00 | 2019-12-09 06:43:00 | 273 Wilson St, Seattle, WA 98101 |
| 186848 | 319669 | Wired Headphones | 1 | 11.99 | 2019-12-03 10:39:00 | 778 River St, Dallas, TX 75001 |
| 186849 | 319670 | Bose SoundSport Headphones | 1 | 99.99 | 2019-12-21 21:45:00 | 747 Chestnut St, Los Angeles, CA 90001 |

# Covid-19 Preparation

Data preparation is the act of manipulating raw data into a form that can readily and accurately be analysed, e.g. for business purposes. Data Preparation is a pre-processing step in which data from one or more sources is cleaned and transformed to improve its quality prior to its use in business analytics.

### **Add Month, Hour, Minute, Sales, Cities Column**

In [13]:

def augment\_data(data):

*"""*

*Adding new features to*

*our data, adding Month Data,*

*Hour Data, Minute Data, Sales Data,*

*and Cities Column*

*Returning:*

*data with new features*

*"""*

*# funtction to get the city in the data*

def get\_city(address):

return address.split(',')[1]

*# funtction to get the state in the data*

def get\_state(address):

return address.split(',')[2].split(' ')[1]

*# let's get the year data in order date column*

data['Year'] = data['Order Date'].dt.year

*# let's get the month data in order date column*

data['Month'] = data['Order Date'].dt.month

*# let's get the houe data in order date column*

data['Hour'] = data['Order Date'].dt.hour

*# let's get the minute data in order date column*

data['Minute'] = data['Order Date'].dt.minute

*# let's make the sales column by multiplying the quantity ordered colum with price each column*

data['Sales'] = data['Quantity Ordered'] \* data['Price Each']

*# let's get the cities data in order date column*

data['Cities'] = data['Purchase Address'].apply(lambda x: f"**{**get\_city(x)**}** (**{**get\_state(x)**}**)")

return data *# returning data*

*# and see it*

sales\_data = augment\_data(sales\_data)

sales\_data.head()

Out[13]:

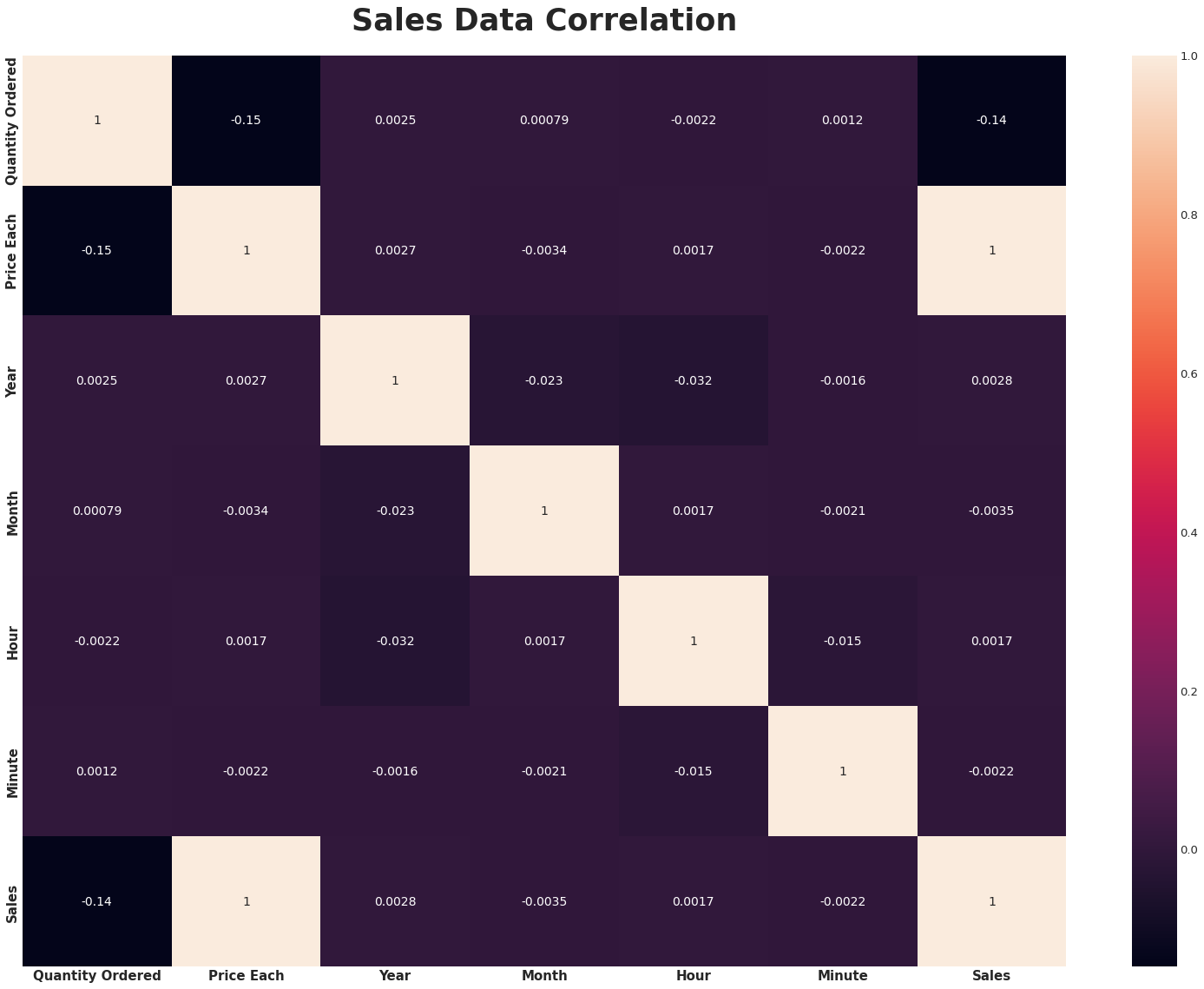
|  | Order ID | Product | Quantity Ordered | Price Each | Order Date | Purchase Address | Year | Month | Hour | Minute | Sales | Cities |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 236670 | Wired Headphones | 2 | 11.99 | 2019-08-31 22:21:00 | 359 Spruce St, Seattle, WA 98101 | 2019 | 8 | 22 | 21 | 23.98 | Seattle (WA) |
| 1 | 236671 | Bose SoundSport Headphones | 1 | 99.99 | 2019-08-15 15:11:00 | 492 Ridge St, Dallas, TX 75001 | 2019 | 8 | 15 | 11 | 99.99 | Dallas (TX) |
| 2 | 236672 | iPhone | 1 | 700.00 | 2019-08-06 14:40:00 | 149 7th St, Portland, OR 97035 | 2019 | 8 | 14 | 40 | 700.00 | Portland (OR) |
| 3 | 236673 | AA Batteries (4-pack) | 2 | 3.84 | 2019-08-29 20:59:00 | 631 2nd St, Los Angeles, CA 90001 | 2019 | 8 | 20 | 59 | 7.68 | Los Angeles (CA) |
| 4 | 236674 | AA Batteries (4-pack) | 2 | 3.84 | 2019-08-15 19:53:00 | 736 14th St, New York City, NY 10001 | 2019 | 8 | 19 | 53 | 7.68 | New York City (NY) |

# COVID-19 Analysis

Covid-19 Analysis is the process of systematically applying statistical and/or logical techniques to describe and illustrate, condense and recap, and evaluate data. Indeed, researchers generally analyze for patterns in observations through the entire data collection phase *(Savenye, Robinson,*20042004*)*. analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

Or, the easier, you can say in Data Analysis we (Data Scientist or Data Analyst) what ever you want to call that, in this section, we're looking for the correlation and also the relationships between every data (features and labels) or the variables using and applying the statistical and visualization methods for looking some patterns.

unfold\_moreShow hidden code



In [15]:

*# Let's see the correlation from `sales\_data`*

(sales\_data.corr()['Sales'] *# transform it into data corr*

.sort\_values(ascending=False) *# sort values*

.to\_frame() *# change it into data frame*

.T) *# transpose it*

Out[15]:

|  | Sales | Price Each | Year | Hour | Minute | Month | Quantity Ordered |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sales | 1.0 | 0.999203 | 0.002824 | 0.001668 | -0.002162 | -0.003466 | -0.139417 |

In [16]:

*# statistical measure of sales data without object type of data*

sales\_data\_numeric = sales\_data.describe(include=[np.number])

"Statistical Measure of Sales Data in Numeric Data"

sales\_data\_numeric

Out[16]:

'Statistical Measure of Sales Data in Numeric Data'

Out[16]:

|  | Quantity Ordered | Price Each | Year | Month | Hour | Minute | Sales |
| --- | --- | --- | --- | --- | --- | --- | --- |
| count | 185950.000000 | 185950.000000 | 185950.000000 | 185950.000000 | 185950.000000 | 185950.000000 | 185950.000000 |
| mean | 1.124383 | 184.399735 | 2019.000183 | 7.059140 | 14.413305 | 29.481361 | 185.490917 |
| std | 0.442793 | 332.731330 | 0.013521 | 3.502996 | 5.423416 | 17.317573 | 332.919771 |
| min | 1.000000 | 2.990000 | 2019.000000 | 1.000000 | 0.000000 | 0.000000 | 2.990000 |
| 25% | 1.000000 | 11.950000 | 2019.000000 | 4.000000 | 11.000000 | 14.000000 | 11.950000 |
| 50% | 1.000000 | 14.950000 | 2019.000000 | 7.000000 | 15.000000 | 29.000000 | 14.950000 |
| 75% | 1.000000 | 150.000000 | 2019.000000 | 10.000000 | 19.000000 | 45.000000 | 150.000000 |
| max | 9.000000 | 1700.000000 | 2020.000000 | 12.000000 | 23.000000 | 59.000000 | 3400.000000 |

In [17]:

*# statistical measure of sales data without numeric type of data*

sales\_data\_object = sales\_data.describe(exclude=[np.number])

"Statistical Measure of Sales Data in Object / Str Data"

sales\_data\_object

Out[17]:

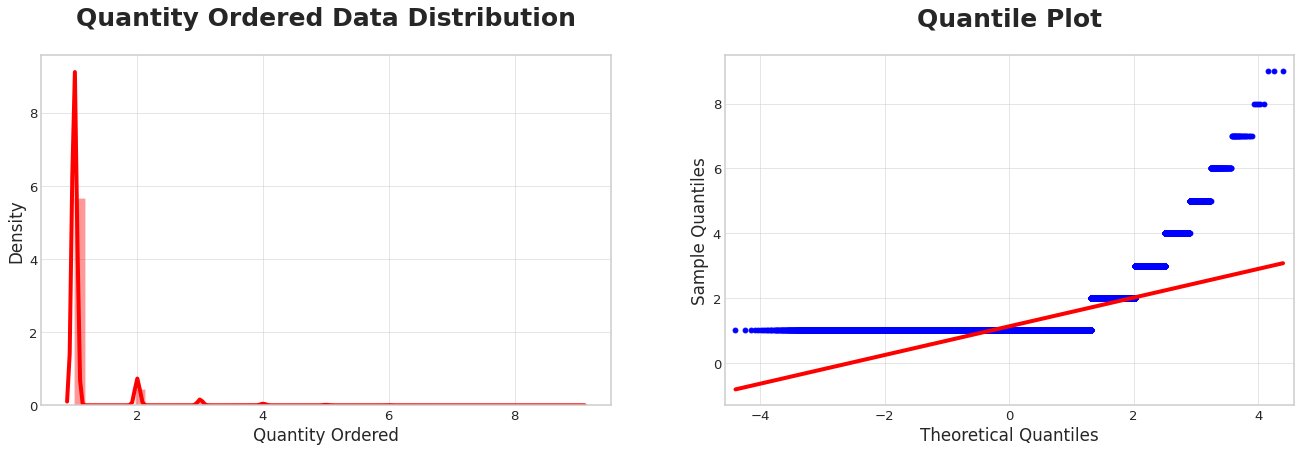
'Statistical Measure of Sales Data in Object / Str Data'

Out[17]:

|  | Order ID | Product | Order Date | Purchase Address | Cities |
| --- | --- | --- | --- | --- | --- |
| count | 185950 | 185950 | 185950 | 185950 | 185950 |
| unique | 178437 | 19 | 142395 | 140787 | 10 |
| top | 160873 | USB-C Charging Cable | 2019-12-15 20:16:00 | 193 Forest St, San Francisco, CA 94016 | San Francisco (CA) |
| freq | 5 | 21903 | 8 | 9 | 44732 |
| first | NaN | NaN | 2019-01-01 03:07:00 | NaN | NaN |
| last | NaN | NaN | 2020-01-01 05:13:00 | NaN | NaN |

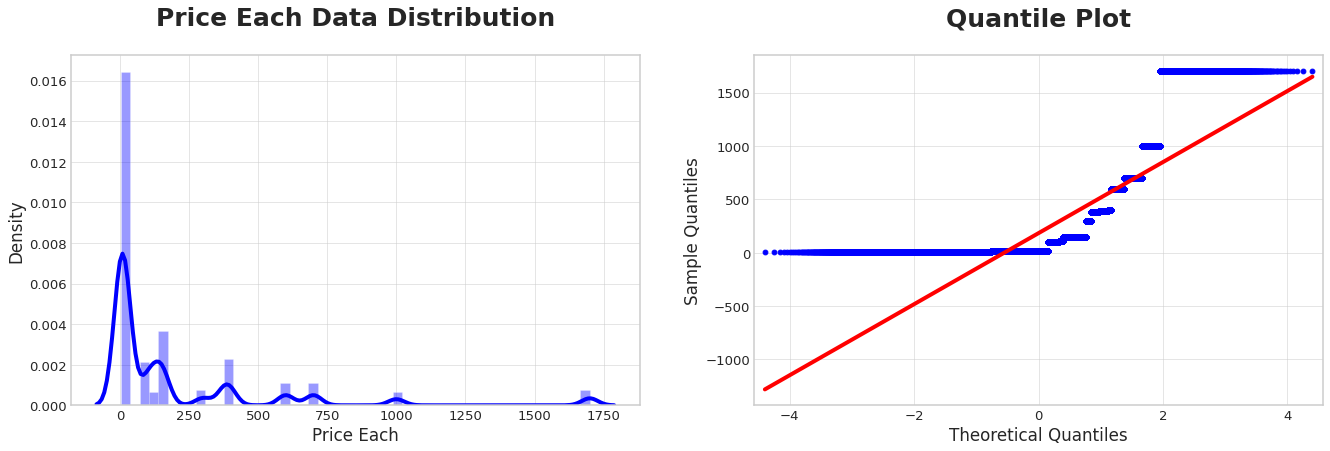
## Univariate Analysis

Univariate analysis is perhaps the simplest form of statistical analysis. Like other forms of statistics, it can be inferential or descriptive. The key fact is that only one variable is involved. Univariate analysis can yield misleading results in cases in which multivariate analysis is more appropriate.



Here we can see it, the average customer buys 11 item/product more often, there are also a few customers who buy 22 or 44 items/product at once, more than that it is very rare.

At least 75%75% of the Sales Data Quantity Ordered population in the USA has a Quantity Ordered range from 0−20−2 item/product.



## Price Each

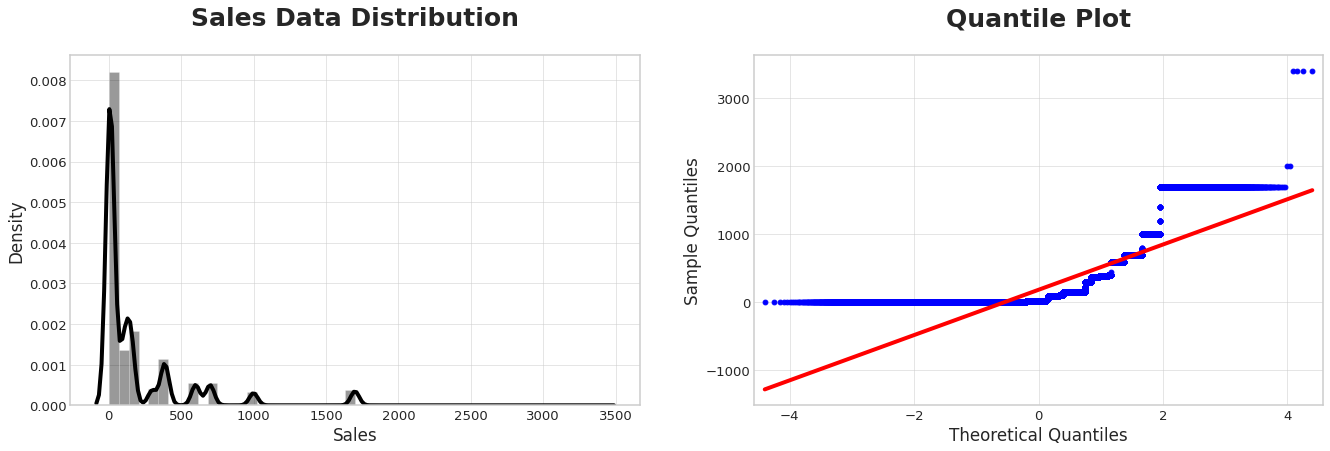
Find the proportion that lies in between two standard deviation (σ) from mean (μ), and let's try to interprete that. and in the Price Each Data, the μ=184.3=184.3 and the σ=332.7=332.7, then without further ado let's calculate it.

#### **Calculation:**

* 184.3−2(332.7)=−481184.3−2(332.7)=−481
* 184.3+2(332.7)=849.7184.3+2(332.7)=849.7

#### *Interpretation:*

At least 75%75% of the population Sales Price data for each item/product in the USA has a price range for each item/product from 0−849.70−849.7 (USD).



**Covid-19**

Find the proportion that lies in between two standard deviation (σ) from mean (μ), and let's try to interprete that. and in the Sales Data, the μ=185.4=185.4 and the σ=332.9=332.9, then without further ado let's calculate it.

#### **Calculation:**

* 185.4−2(332.9)=−480185.4−2(332.9)=−480
* 185.4+2(332.9)=851.19185.4+2(332.9)=851.19

#### *Interpretation:*

At least 75%75% of population Sales Data customers in USA have Sales range from 0−851.190−851.19 (USD).

In [22]:

*# checking skewness value*

*# if value lies between -0.5 to 0.5 then it is normal otherwise skewed*

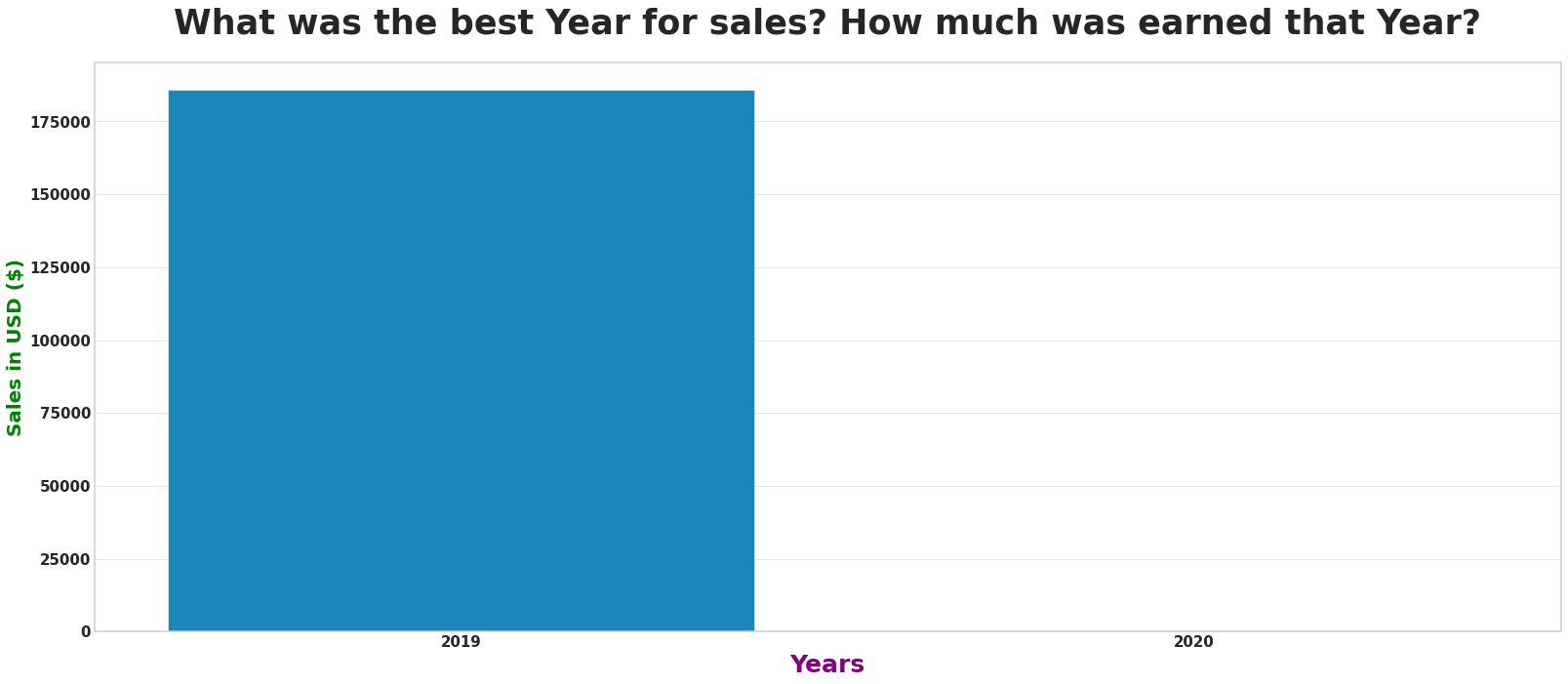
skew\_value = sales\_data.skew().sort\_values(ascending=False).to\_frame()

skew\_value

Out[22]:

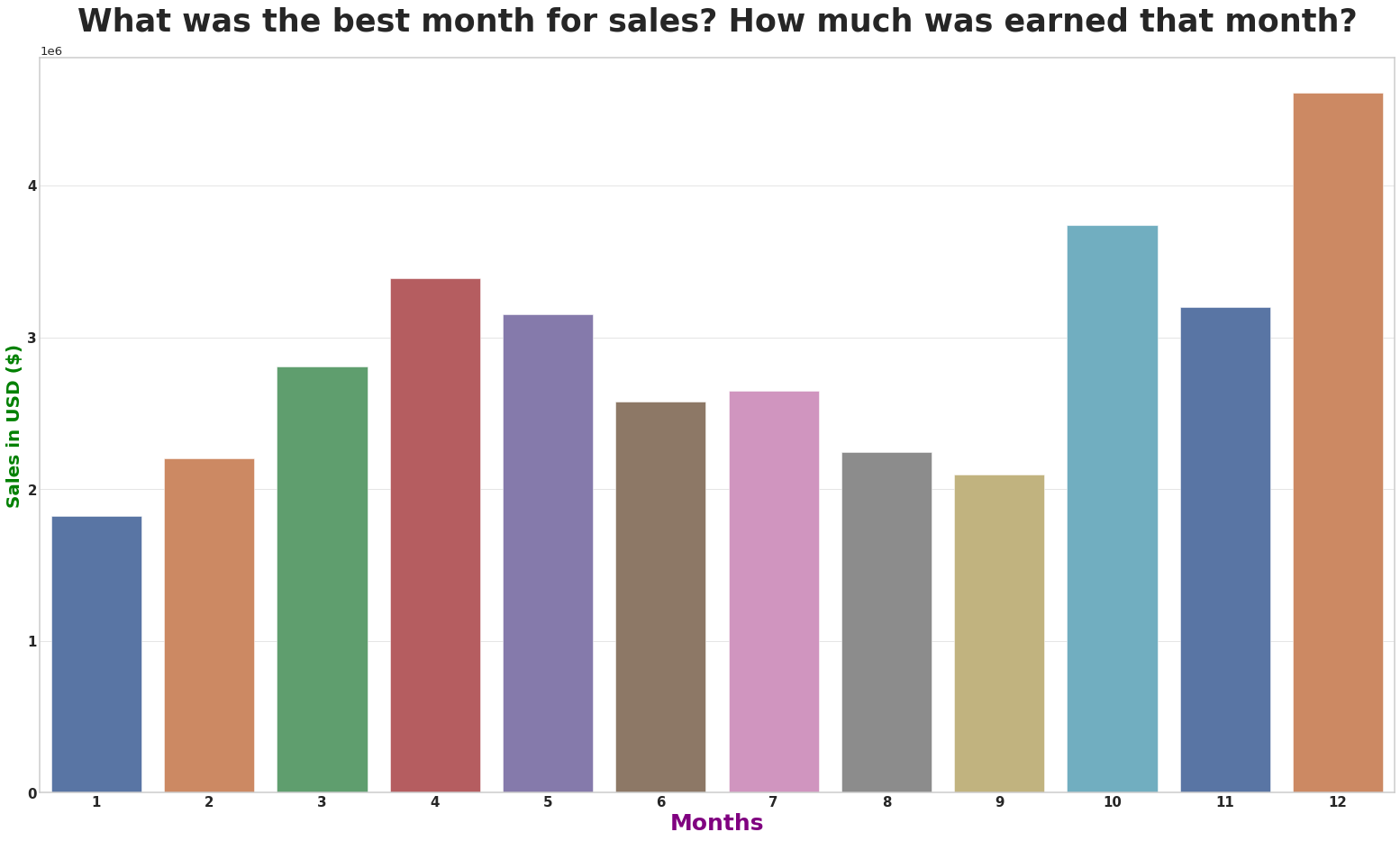
|  | 0 |
| --- | --- |
| Year | 73.933795 |
| Quantity Ordered | 4.833164 |
| Sales | 2.881913 |
| Price Each | 2.872149 |
| Minute | 0.002580 |
| Order ID | 0.000719 |
| Month | -0.088588 |
| Hour | -0.530377 |

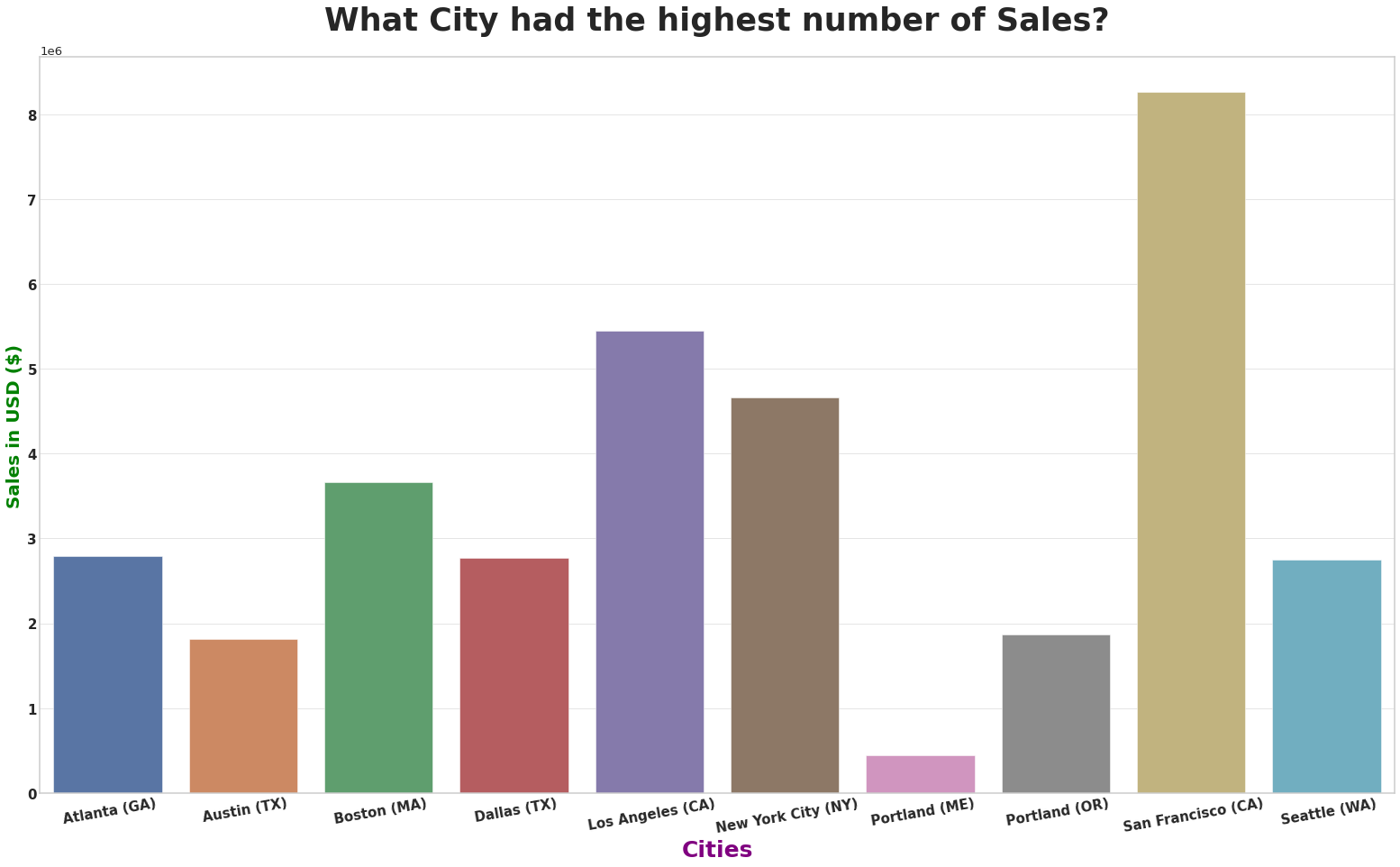
It can be seen that most of the data we have are in the form of a normal distribution, and there are two skewed data



### *Answer:*

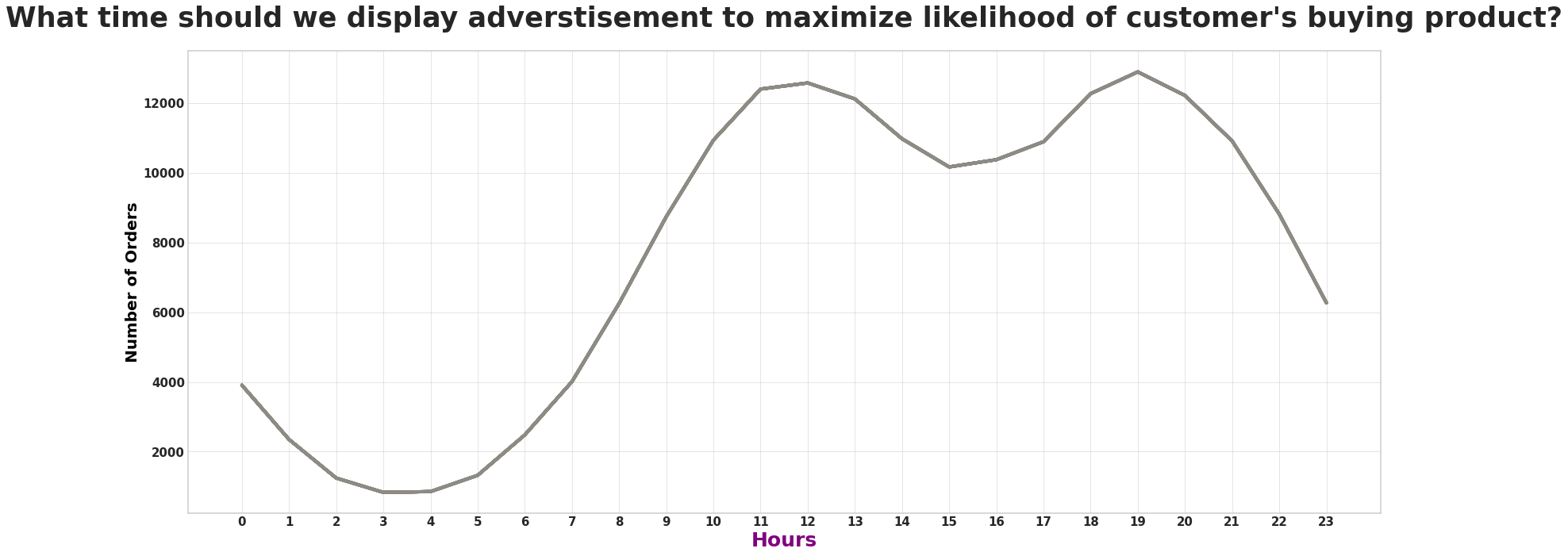
When viewed from the data above, 20192019 was the best year that had the highest number of sales, which was $34,483,365$34,483,365, compared to 20202020 which only had $8,670$8,670 in sales, this is due to the lack of data in 20202020 which caused a data imbalance.





### *Answer:*

The city that has the most sales data in the above visualization is San Francisco, with total sales reaching $8,262,203$8,262,203.



### *Answer:*

My recommendation if you want to place an ad, place the ad at 99 am or 1010 am, because there is an increase in the number of orders at that time.

**What products are most often sold together?**

('iPhone', 'Lightning Charging Cable') 1005

('Google Phone', 'USB-C Charging Cable') 987

('iPhone', 'Wired Headphones') 447

('Google Phone', 'Wired Headphones') 414

('Vareebadd Phone', 'USB-C Charging Cable') 361

('iPhone', 'Apple Airpods Headphones') 360

('Google Phone', 'Bose SoundSport Headphones') 220

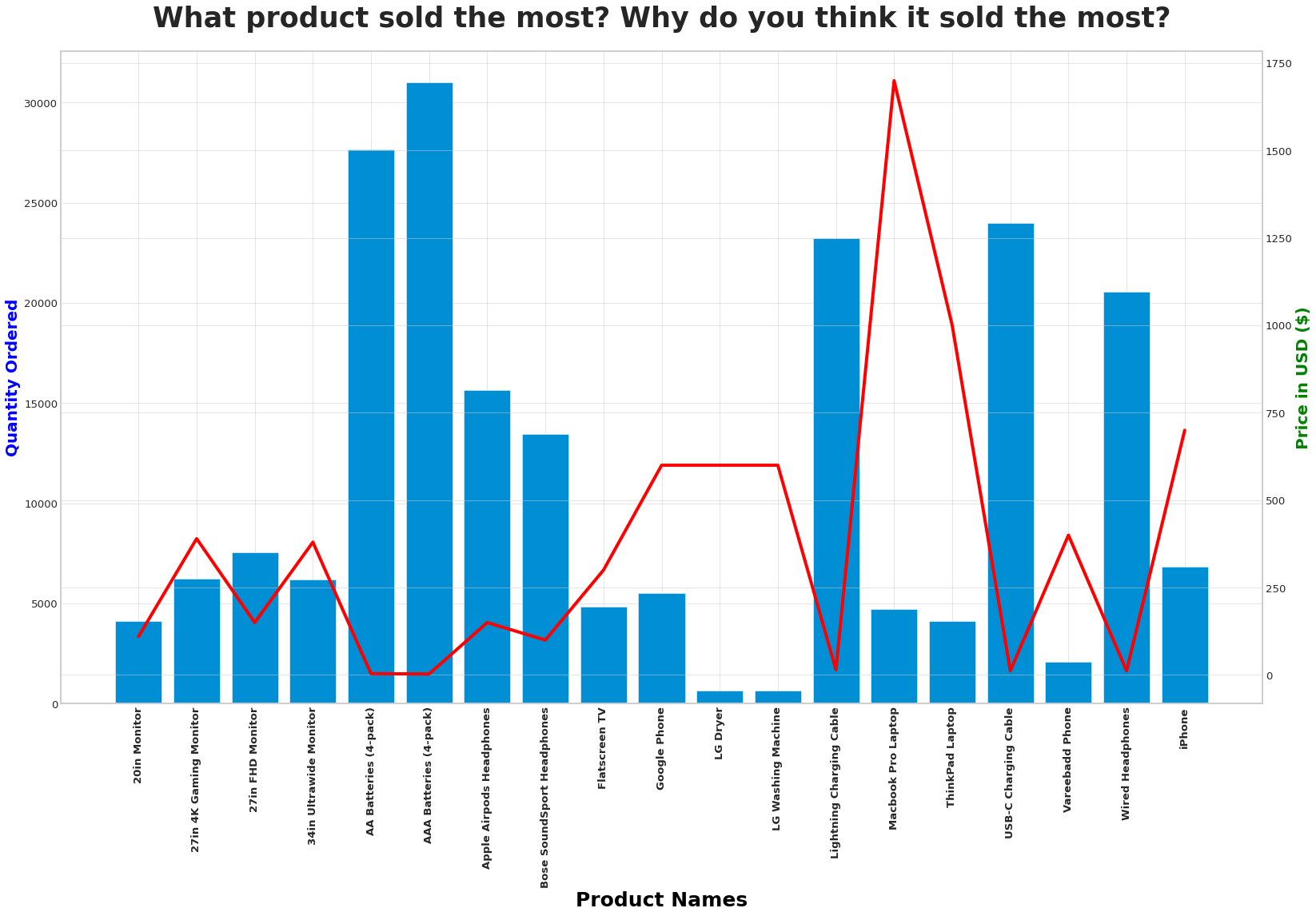
('USB-C Charging Cable', 'Wired Headphones') 160

('Vareebadd Phone', 'Wired Headphones') 143

('Lightning Charging Cable', 'Wired Headphones') 92

### *Answer:*

Products that are often sold simultaneously are iPhone and Lightning Charging Cable which sold 10051005 orders, and Google Phone, USB-C Charging Cable ranked second with 987987 orders.



### **probability for next people will order iPhone?**

In [31]:

iphone = sales\_data[sales\_data.Product == 'iPhone'].value\_counts().sum()

*# Calculating iPhone Probability*

P\_iphone = statistical\_probability(iphone, product)

Pprime\_iphone = 1 - P\_iphone

print('Probability for next people will order iPhone: **%.2f%%**' % P\_iphone)

print('Probability for next people will not order iPhone: **%.2f%%**' % Pprime\_iphone)

Probability for next people will order iPhone: 0.04%

Probability for next people will not order iPhone: 0.96%

#### **Answer**

The probability for next people will order iPhone is 4%4%, Because there are only 4%4% chance we can say this an **unusual event**.

### **probability for next people will order Google Phone?**

In [32]:

google\_phone = sales\_data[sales\_data.Product == 'Google Phone'].value\_counts().sum()

*# Calculating Google Phone Probability*

P\_google\_phone = statistical\_probability(google\_phone, product)

Pprime\_google\_phone = 1 - P\_google\_phone

print('Probability for next people will order Google Phone: **%.2f%%**' % P\_google\_phone)

print('Probability for next people will not order Google Phone: **%.2f%%**' % Pprime\_google\_phone)

Probability for next people will order Google Phone: 0.03%

Probability for next people will not order Google Phone: 0.97%

#### **Answer**

The probability for next people will order Google Phone is 3%3%, Because there are only 3%3% chance we can say this an **unusual event**.

### **probability for next people will order Wired Headphones?**

In [33]:

wired\_headphones = sales\_data[sales\_data.Product == 'Wired Headphones'].value\_counts().sum()

*# Calculating Wired Headphones Probability*

P\_wired\_headphones = statistical\_probability(wired\_headphones, product)

Pprime\_wired\_headphones = 1 - P\_wired\_headphones

print('Probability for next people will order Wired Headphones: **%.2f%%**' % P\_wired\_headphones)

print('Probability for next people will not order Wired Headphones: **%.2f%%**' % Pprime\_wired\_headphones)

Probability for next people will order Wired Headphones: 0.10%

Probability for next people will not order Wired Headphones: 0.90%

#### **Answer**

The probability for next people will order Wired Headphones is 10%10%, Because there are only 10%10% chance we can say this an **unusual event**.