Product Sales Analysis Using Python

About The DataSet

The dataset consists of 12 files which is related to sales for each month of 2019.This dataset contain various details of Products Sold / Purchase by month, product type, cost, purchase address, etc.

Objective of This Data Analysis

The objective of this Analysis as follow:-

1) To find out the patterns in selling structure, 2) Most demanding product, 3) The best selling month for the Sale, 4) The best time to advertisements to increase Sale or buying products, 5) Most busy city to sold the product. 6) And visualise them to obtain important information about the product sales

STEP 1 - Importing Necessary Python Libraries

In [1]:

```python
import pandas as pd # for the data processing,CSV file reading and Data cleaning
import numpy as np # for the N-dimensional array and linear algebra
import seaborn as sns # For the visualization of data set
import matplotlib.pyplot as plt # for the visualization of data set
import plotly.express as px # to visualize a variety of types of data
from datetime import datetime
```

In [2]:

```python
from plotly.offline import iplot # to display the plot when working on offline
import plotly
plotly.offline.init_notebook_mode(connected=True)
```

STEP 2 - Loading Dataset and making single Dataset/DataFrame

- Reading all the 12 file and merging all file in single data frame.

In [3]:

```python
jan=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_January_2019.csv")
feb=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_February_2019.csv")
mar=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_March_2019.csv")
apr=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_April_2019.csv")
may=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_May_2019.csv")
june=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_June_2019.csv")
july=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_July_2019.csv")
aug=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_August_2019.csv")
sep=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_September_2019.csv")
octo=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_October_2019.csv")
nov=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_November_2019.csv")
dec=pd.read_csv("C:/Users/Lenovo/OneDrive/Desktop/Choice-Data/Sales_Data/Sales_December_2019.csv")
```

In [4]:

```python
jan["Month"]="Jan"
feb["Month"]="Feb"
mar["Month"]="March"
apr["Month"]="Apr"
may["Month"]="May"
june["Month"]="June"
july["Month"]="July"
aug["Month"]="Aug"
sep["Month"]="Sep"
octo["Month"]="Oct"
nov["Month"]="Nov"
dec["Month"]="Dec"
```

In [5]:

```python
#Now concatenating the all  data in one DataFrame called as df
df = pd.concat([jan,feb,mar,apr,may,june,july,aug,sep,octo,nov,dec],axis=0)
df.head()
```

Out[5]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	141234	iPhone	1	700	01/22/19 21:25	944 Walnut St, Boston, MA 02215	Jan
1	141235	Lightning Charging Cable	1	14.95	01/28/19 14:15	185 Maple St, Portland, OR 97035	Jan
2	141236	Wired Headphones	2	11.99	01/17/19 13:33	538 Adams St, San Francisco, CA 94016	Jan
3	141237	27in FHD Monitor	1	149.99	01/05/19 20:33	738 10th St, Los Angeles, CA 90001	Jan
4	141238	Wired Headphones	1	11.99	01/25/19 11:59	387 10th St, Austin, TX 73301	Jan

Following are the meaning of column which shown in above Dataset

- **Order ID** - This is the unique number identifier that gets generated when we place an order. which is use to keep track of orders. This number can be useful to the seller when attempting to find out certain details about an order such as shipment date or status.

- **Product** - The product that have been sold.
- **Quantity Ordered** - Ordered Quantity is the total item quantity ordered.
- **Price Each** - The price of each products.
- **Order Date** - This is the date on which the customer is requesting the order to be shipped.
- **Purchase Address** -This is the location where the customer would like his purchased items delivered.

STEP 3 - Handing Missing Value

```
In [6]: df.isnull().sum()
```

```
Out[6]: Order ID            545
        Product             545
        Quantity Ordered    545
        Price Each          545
        Order Date          545
        Purchase Address    545
        Month                 0
        dtype: int64
```

```
In [7]: percent_missing = df.isnull().sum() * 100 / len(df)
        percent_missing
```

```
Out[7]: Order ID            0.291678
        Product             0.291678
        Quantity Ordered    0.291678
        Price Each          0.291678
        Order Date          0.291678
        Purchase Address    0.291678
        Month               0.000000
        dtype: float64
```

```
In [8]: # by checking the above details Looks like the percent missing of the data is not too big.
        df = df.dropna()
        df.isnull().sum()
```

```
Out[8]: Order ID            0
        Product             0
        Quantity Ordered    0
        Price Each          0
        Order Date          0
        Purchase Address    0
        Month               0
        dtype: int64
```

- in the data set there are columns in which values in rows are the same as the header. so now droping this row from dataset

```
In [9]: df['Quantity Ordered'].unique()
```

```
Out[9]: array(['1', '2', '3', '5', '4', '7', 'Quantity Ordered', '6', '9', '8'],
              dtype=object)
```

```
In [10]: #create filter to drop text values
         filter_repeated_values = df['Quantity Ordered'] != 'Quantity Ordered'

         #replace data without text values in quality ordered
         df = df[filter_repeated_values]
```

```
In [11]: df['Quantity Ordered'].unique()
```

```
Out[11]: array(['1', '2', '3', '5', '4', '7', '6', '9', '8'], dtype=object)
```

STEP 4 - Basic Descriptions of the Data

```
In [12]: df.shape
```

```
Out[12]: (185950, 7)
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 25116
Data columns (total 7 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  object
 3   Price Each        185950 non-null  object
 4   Order Date        185950 non-null  object
 5   Purchase Address  185950 non-null  object
 6   Month             185950 non-null  object
dtypes: object(7)
memory usage: 11.3+ MB
```

- **As we see from the info the data type of column are object so converting data type of column**

In [14]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'], errors='coerce')
df['Purchase Address'] = df['Purchase Address'].astype('str')
df['Quantity Ordered'] = df['Quantity Ordered'].astype('int64')
df['Order ID'] = df['Order ID'].astype('int64')
df['Price Each'] = df['Price Each'].astype('float')
```

In [15]:
```
# after converting data type now checking info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 25116
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   Order ID          185950 non-null  int64         
 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  datetime64[ns]
 5   Purchase Address  185950 non-null  object        
 6   Month             185950 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(2), object(3)
memory usage: 11.3+ MB
```

In [16]:
```
df.describe().T
```

Out[16]:

	count	mean	std	min	25%	50%	75%	max
Order ID	185950.0	230417.569379	51512.737110	141234.00	185831.25	230367.50	275035.75	319670.0
Quantity Ordered	185950.0	1.124383	0.442793	1.00	1.00	1.00	1.00	9.0
Price Each	185950.0	184.399735	332.731330	2.99	11.95	14.95	150.00	1700.0

STEP 5 - Data preparation for the analysis work

- **Adding Month, City, Total Sale and Time Column**

In [17]:
```
df['City'] = df['Purchase Address'].str.split(',').str[1].astype(str)
df['Total Sale'] = df['Quantity Ordered']* df['Price Each']
# Also creating the new column of time for the analysis part of which was time is good for the advertisement
df['Time'] = df['Order Date'].dt.hour
```

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

Out[18]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City	Total Sale	Time
0	141234	iPhone	1	700.00	2019-01-22 21:25:00	944 Walnut St, Boston, MA 02215	Jan	Boston	700.00	21
1	141235	Lightning Charging Cable	1	14.95	2019-01-28 14:15:00	185 Maple St, Portland, OR 97035	Jan	Portland	14.95	14
2	141236	Wired Headphones	2	11.99	2019-01-17 13:33:00	538 Adams St, San Francisco, CA 94016	Jan	San Francisco	23.98	13
3	141237	27in FHD Monitor	1	149.99	2019-01-05 20:33:00	738 10th St, Los Angeles, CA 90001	Jan	Los Angeles	149.99	20
4	141238	Wired Headphones	1	11.99	2019-01-25 11:59:00	387 10th St, Austin, TX 73301	Jan	Austin	11.99	11

Sales Data Analysis

Q.1:- What was the best month for sales and why?

In [19]:

```
monthly_Product_sales = df[['Total Sale', 'Month', 'Product']].groupby(['Month', 'Product']).sum().reset_index()
monthly_Product_sales
```

Out[19]:

	Month	Product	Total Sale
0	Apr	20in Monitor	43446.05
1	Apr	27in 4K Gaming Monitor	220344.35
2	Apr	27in FHD Monitor	110542.63
3	Apr	34in Ultrawide Monitor	248133.47
4	Apr	AA Batteries (4-pack)	10836.48
...	...	...	...
223	Sep	ThinkPad Laptop	248997.51
224	Sep	USB-C Charging Cable	19048.30
225	Sep	Vareebadd Phone	50400.00
226	Sep	Wired Headphones	15610.98
227	Sep	iPhone	278600.00

228 rows × 3 columns

In [20]:

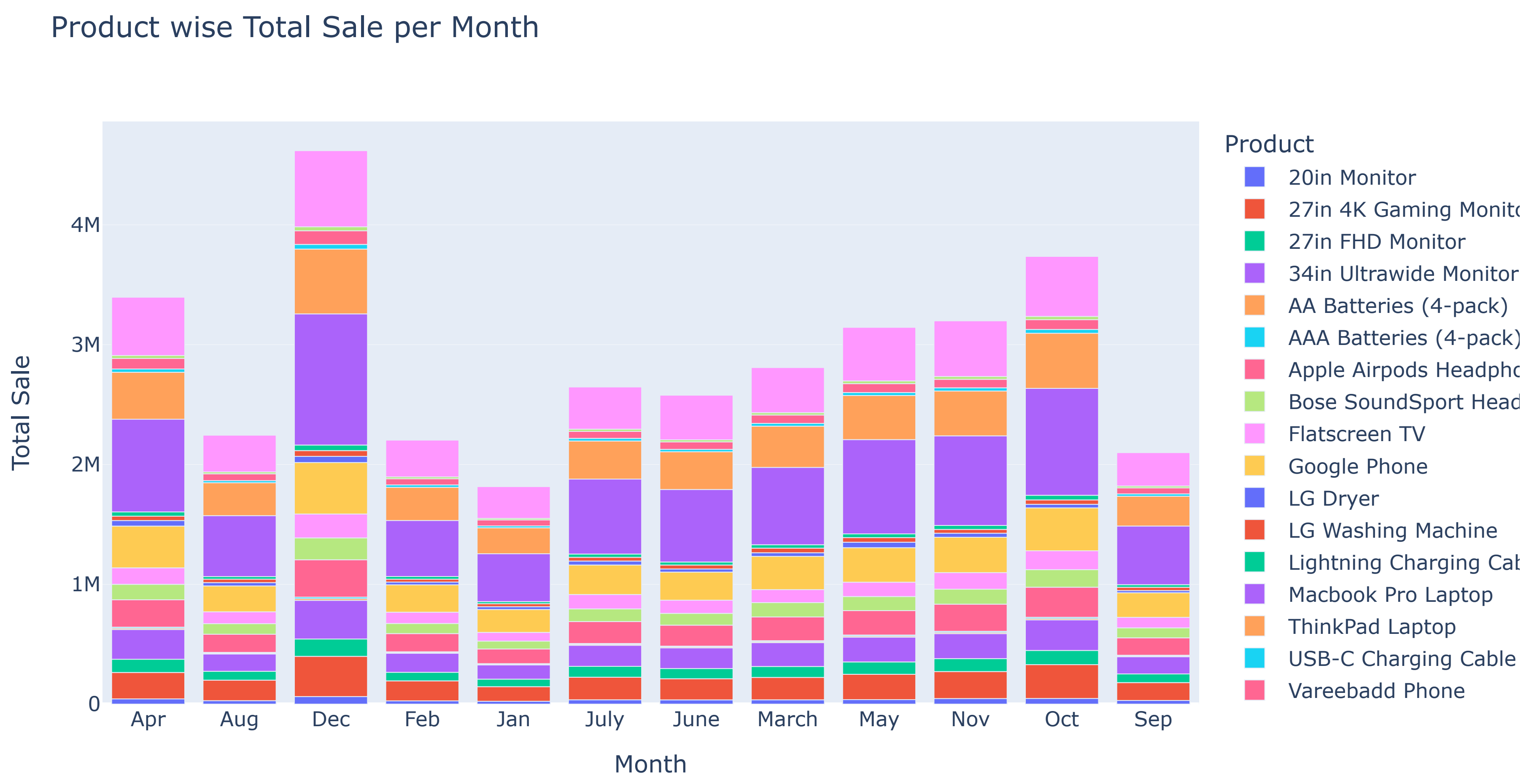
```
monthly_sales = df[['Month','Total Sale']].groupby('Month').sum().sort_values(by=['Total Sale'],ascending=False).reset_i
monthly_sales
```

Out[20]:

	Month	Total Sale
0	Dec	4619297.12
1	Oct	3736884.05
2	Apr	3396059.11
3	Nov	3198909.23
4	May	3144584.80
5	March	2809063.30
6	July	2646899.69
7	June	2578293.30
8	Aug	2244412.31
9	Feb	2203481.24
10	Sep	2098816.70
11	Jan	1815335.12

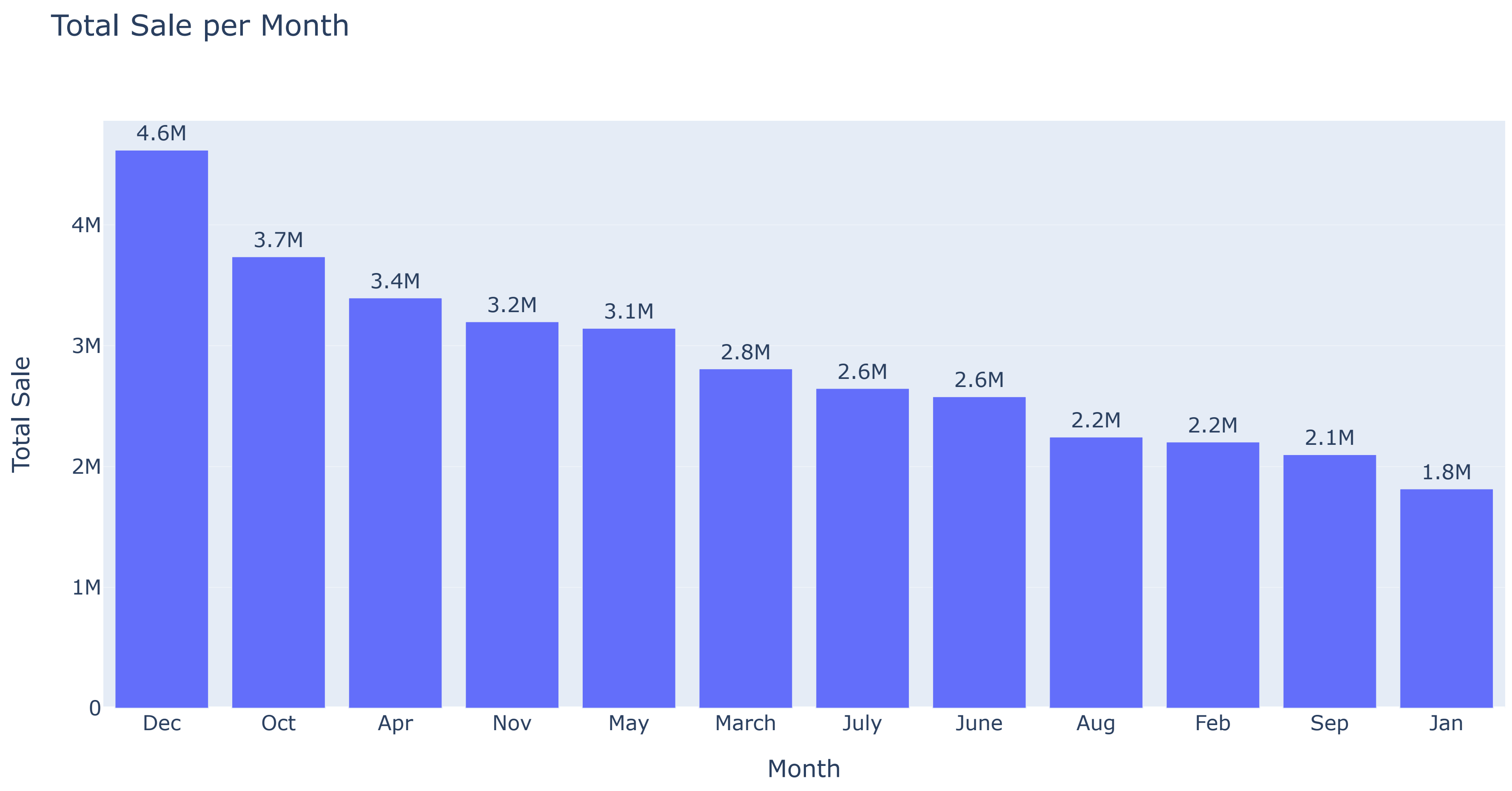
In [21]:

```
px.bar(monthly_Product_sales, x='Month', y='Total Sale', color='Product',title="Product wise Total Sale per Month")
```



In [22]:

```
fig = px.bar(monthly_sales, x='Month',y='Total Sale',text_auto='.2s',title="Total Sale per Month")
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.show()
```



Conclusion for question-1

1) The **best month to sell** is shown in the visualization above is **December** which has a record number of sales approximate to **4.62 Million Dollars.**
2) This **may be because in December there is Christmas,New Year Celebration and Holidays** where many people buy gifts for loved ones.

Q.2:- Which city sold the most product?

In [23]:

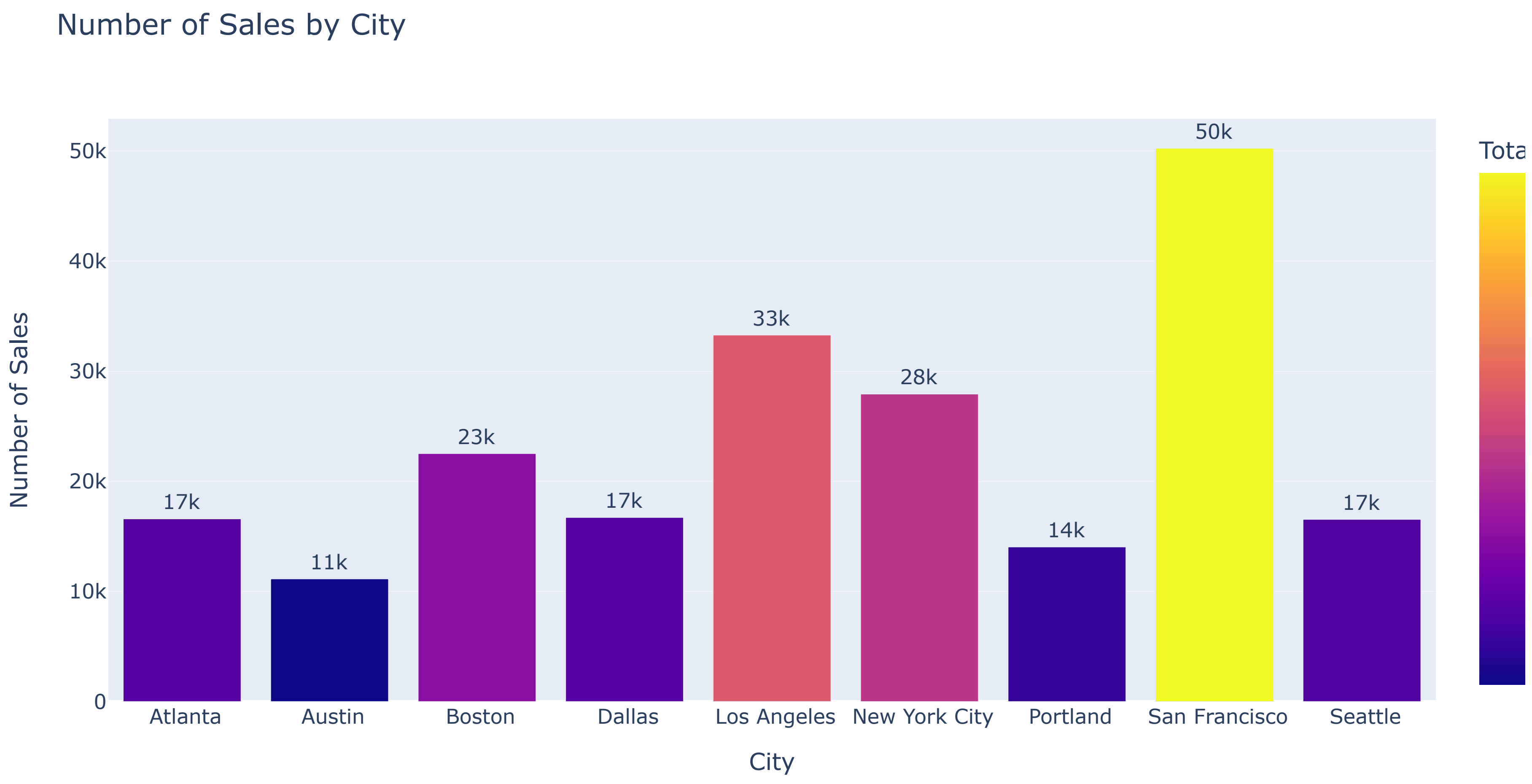
```
City_sales = df[['City','Quantity Ordered','Total Sale', ]].groupby(['City']).sum().reset_index()
City_sales.rename(columns={'Quantity Ordered': 'Number of Sales'}, inplace=True)
City_sales
```

Out[23]:

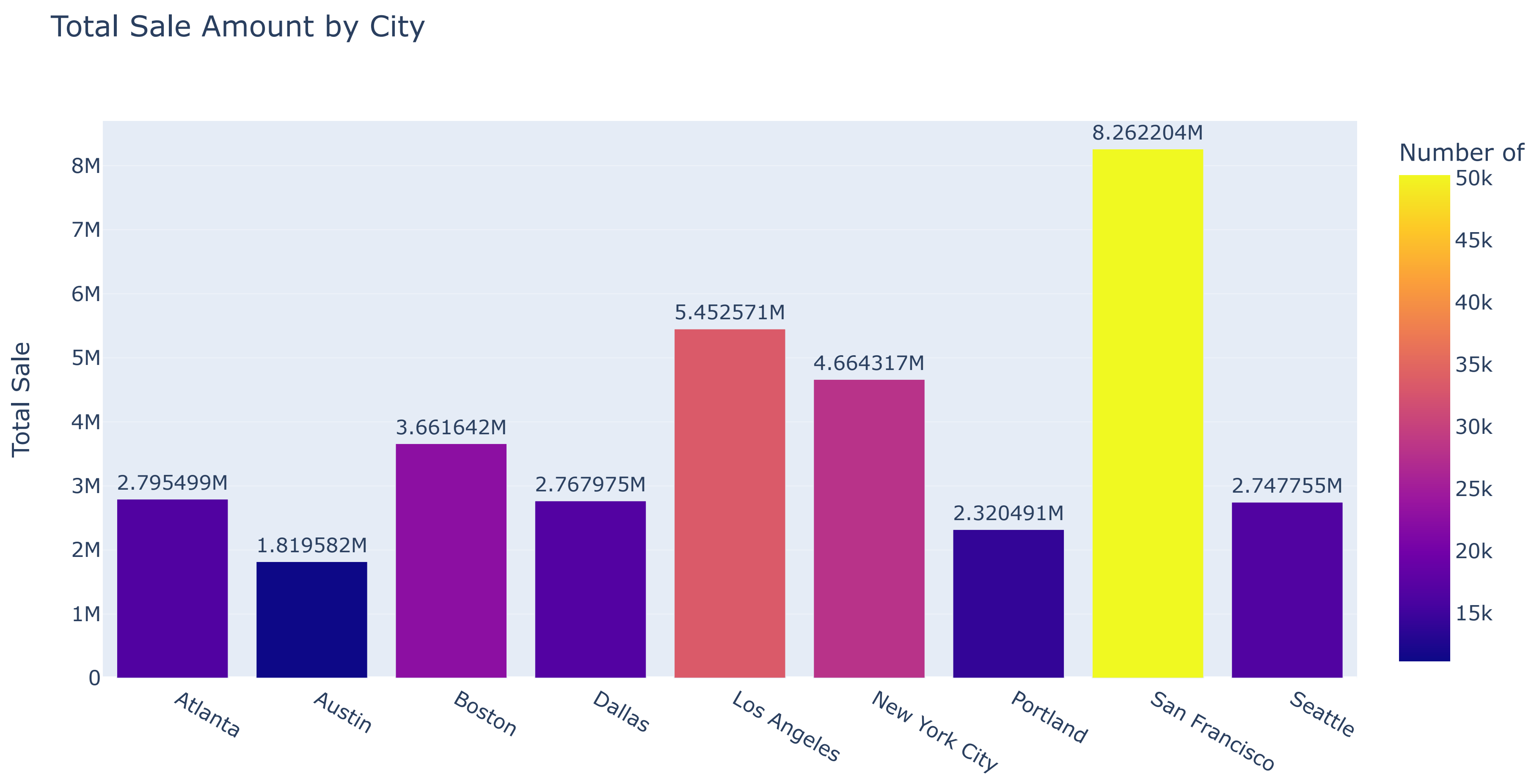
	City	Number of Sales	Total Sale
0	Atlanta	16602	2795498.58
1	Austin	11153	1819581.75
2	Boston	22528	3661642.01
3	Dallas	16730	2767975.40
4	Los Angeles	33289	5452570.80
5	New York City	27932	4664317.43
6	Portland	14053	2320490.61
7	San Francisco	50239	8262203.91
8	Seattle	16553	2747755.48

In [24]:

```
fig = px.bar(City_sales, 'City', 'Number of Sales', color='Total Sale',text_auto='.2s',title="Number of Sales by City")
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.show()
```



```
In [25]: fig = px.bar(City_sales, 'City', 'Total Sale', color='Number of Sales',text_auto=True,title="Total Sale Amount by City",
         fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
         fig.show()
```



Conclusion for question -2

From the above visulaization here we can conclude that **San Francisco** had the highest **Number of sales quantity** which is **50,239** and **Total Sale** value is approximately **8.26 Million Dollars.**

Q.3:-What time should we display advertisements to maximize likelihood of customer’s buying products and why?

In [26]:

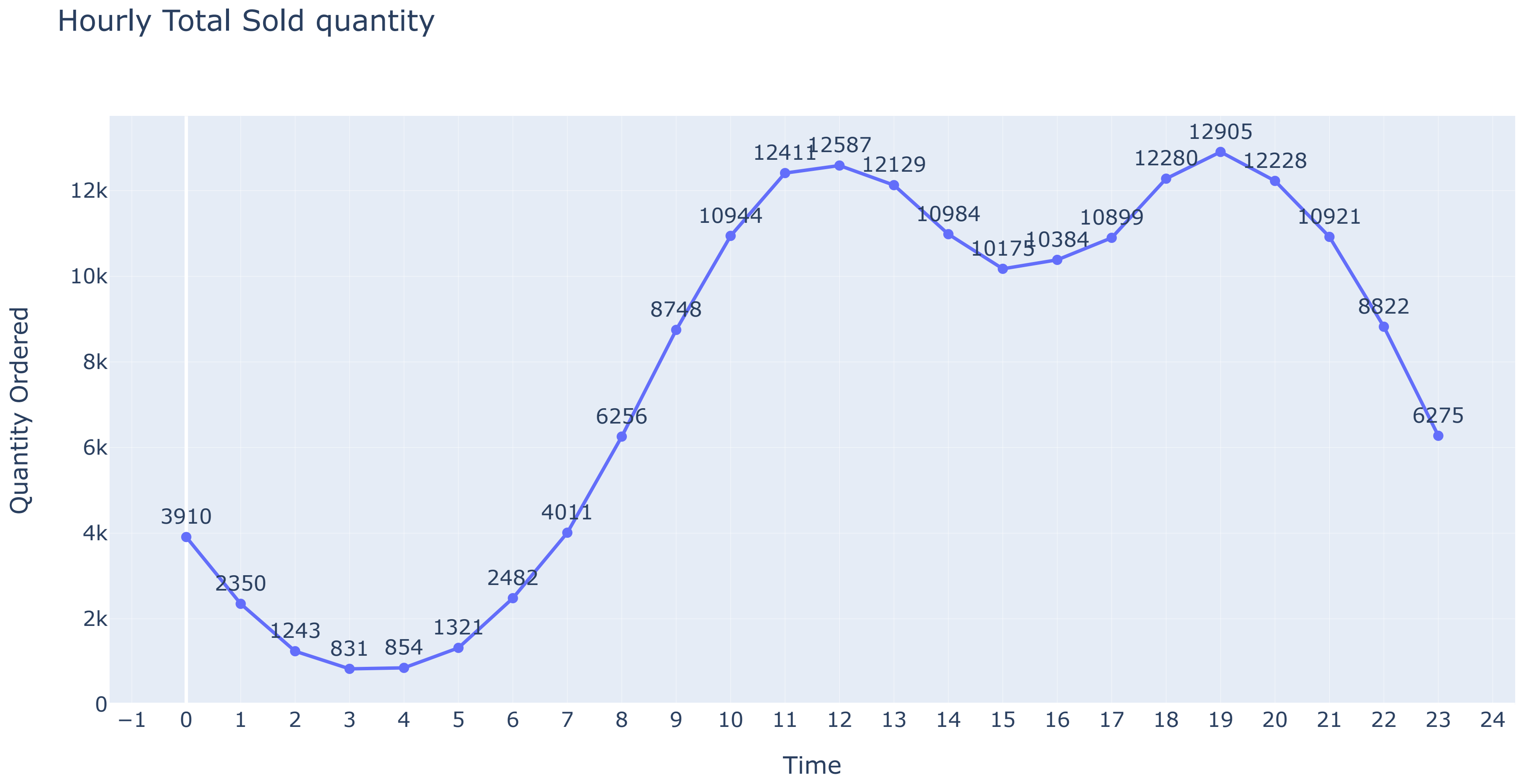
```
Hourly_sales = pd.concat([df.groupby(['Time']).count()['Quantity Ordered'],
                          df.groupby(['Time']).sum()[['Total Sale']]], axis=1).reset_index()
Hourly_sales
```

Out[26]:

	Time	Quantity Ordered	Total Sale
0	0	3910	713721.27
1	1	2350	460866.88
2	2	1243	234851.44
3	3	831	145757.89
4	4	854	162661.01
5	5	1321	230679.82
6	6	2482	448113.00
7	7	4011	744854.12
8	8	6256	1192348.97
9	9	8748	1639030.58
10	10	10944	1944286.77
11	11	12411	2300610.24
12	12	12587	2316821.34
13	13	12129	2155389.80
14	14	10984	2083672.73
15	15	10175	1941549.60
16	16	10384	1904601.31
17	17	10899	2129361.61
18	18	12280	2219348.30
19	19	12905	2412938.54
20	20	12228	2281716.24
21	21	10921	2042000.86
22	22	8822	1607549.21
23	23	6275	1179304.44

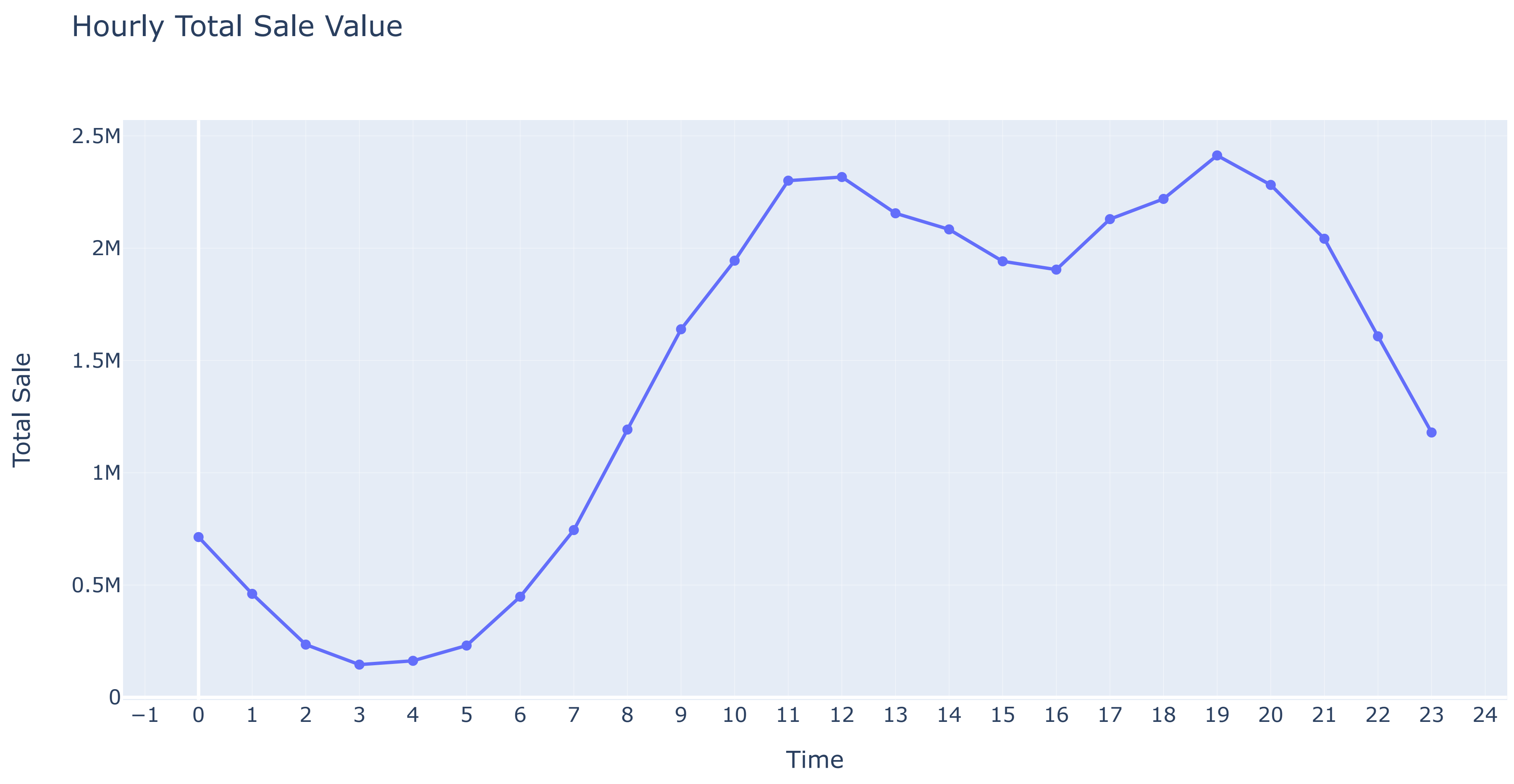
In [27]:

```
fig = px.line(Hourly_sales, x = 'Time', y ='Quantity Ordered', title="Hourly Total Sold quantity",markers=True,text='Qua
fig.update_layout(xaxis = dict(tickmode = 'linear',tick0 = 0,dtick = 1))
fig.update_traces(textposition = "top center")
fig.show()
```



In [28]:

```
fig = px.line(Hourly_sales, x = 'Time', y ='Total Sale', title="Hourly Total Sale Value",markers=True)
fig.update_layout(xaxis = dict(tickmode = 'linear',tick0 = 0,dtick = 1))
fig.show()
```



Conclusion for question -3

From the above visulaization it can be **concluded** that during **10 AM to 1 PM and 5 PM to 9 PM**, received a maximum number of orders and it is **probably the best time to show advertisements to maximize the product selling.**

Q.4:-Which product sold the most? Why do you think it did?

In [29]:

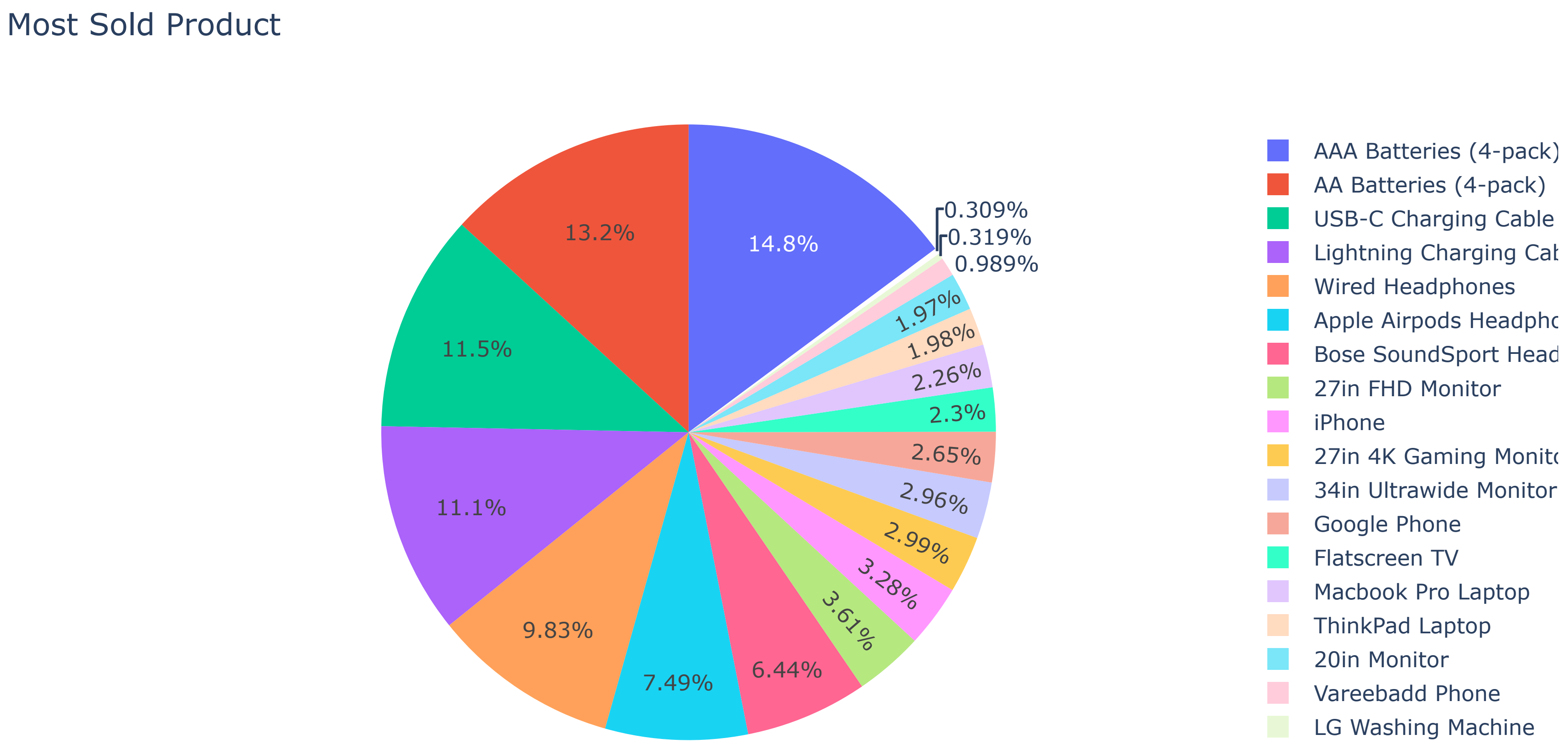
```
product_sold = pd.merge(df.groupby('Product')['Price Each'].mean(),
                    df.groupby('Product')['Quantity Ordered'].sum(),
                    left_index=True,
                    right_index=True).reset_index()

product_sold['Total Revenue'] = product_sold['Price Each'] * product_sold['Quantity Ordered']
product_sold.sort_values(by='Quantity Ordered', ascending=False)
```

Out[29]:

	Product	Price Each	Quantity Ordered	Total Revenue
5	AAA Batteries (4-pack)	2.99	31017	92740.83
4	AA Batteries (4-pack)	3.84	27635	106118.40
15	USB-C Charging Cable	11.95	23975	286501.25
12	Lightning Charging Cable	14.95	23217	347094.15
17	Wired Headphones	11.99	20557	246478.43
6	Apple Airpods Headphones	150.00	15661	2349150.00
7	Bose SoundSport Headphones	99.99	13457	1345565.43
2	27in FHD Monitor	149.99	7550	1132424.50
18	iPhone	700.00	6849	4794300.00
1	27in 4K Gaming Monitor	389.99	6244	2435097.56
3	34in Ultrawide Monitor	379.99	6199	2355558.01
9	Google Phone	600.00	5532	3319200.00
8	Flatscreen TV	300.00	4819	1445700.00
13	Macbook Pro Laptop	1700.00	4728	8037600.00
14	ThinkPad Laptop	999.99	4130	4129958.70
0	20in Monitor	109.99	4129	454148.71
16	Vareebadd Phone	400.00	2068	827200.00
11	LG Washing Machine	600.00	666	399600.00
10	LG Dryer	600.00	646	387600.00

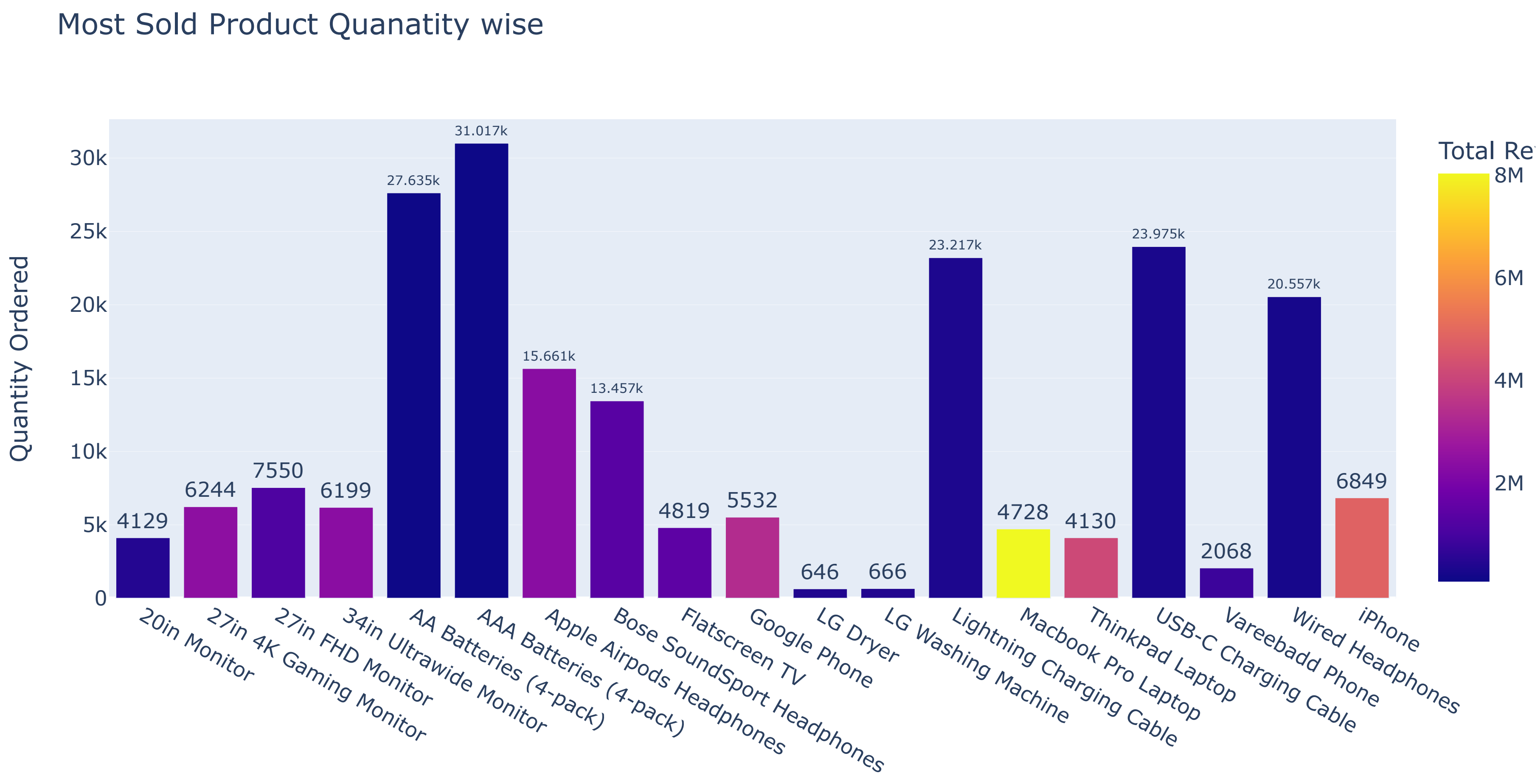
In [30]:

```
fig = px.pie(product_sold, values='Quantity Ordered', names='Product', title='Most Sold Product')
fig.show()
```



In [31]:

```
fig = px.bar(product_sold, 'Product', 'Quantity Ordered', color='Total Revenue',text_auto=True,
             title="Most Sold Product Quanatity wise",)
fig.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
fig.show()
```



Conclusion for question -4

1) The **most sold products** are **AA Batteries (4-pack), AA Batteries (4-pack), Lightning Charging Cable, USB-C Charging Cable, and Wired Headphones.**
2) This is because the **prices** of the most **ordered products** have a **low price** compared to other products.
3) Also it can be concluded that the selling of a product depends on its price. The more expensive the product, the lower will be the quantity ordered and vice versa.