In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Loading the data in the variable:

df = pd.read_csv("all_data.csv")
```

In [3]:
```
# Loading the top 5 Rows of the dataframe:

df.head()
```

Out[3]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001

In [4]:
```
# With the help of .shape we get total number of rows and columns in the data frame:

df.shape
```

Out[4]: (186850, 6)

In [5]:
```
# Gives all the information about the dataframe:

df.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
```

In [6]:
```
# Give to count of the null values column vise:

df.isnull().sum()
```

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

In [7]:
```
# Dropping the rows whose order id is null:

a = df[df['Order ID'].isnull()].index
df.drop(a , axis = 0 , inplace = True)
```

In [8]:
```
# Checking for the null values once again:

df.isnull().sum()
```

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

In [9]:
```
# Checking the datatypes of all the columns
df.dtypes
```

Out[9]:
```
Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
dtype: object
```

In [10]:
```
# Checking all the unique entries of the product column:

df['Product'].unique()
```

Out[10]:
```
array(['USB-C Charging Cable', 'Bose SoundSport Headphones',
       'Google Phone', 'Wired Headphones', 'Macbook Pro Laptop',
       'Lightning Charging Cable', '27in 4K Gaming Monitor',
       'AA Batteries (4-pack)', 'Apple Airpods Headphones',
       'AAA Batteries (4-pack)', 'iPhone', 'Flatscreen TV',
       '27in FHD Monitor', '20in Monitor', 'LG Dryer', 'ThinkPad Laptop',
       'Vareebadd Phone', 'LG Washing Machine', '34in Ultrawide Monitor',
       'Product'], dtype=object)
```

In [11]:
```
# Dropping the rows with the input 'Product' in it

a = df[df['Product'] == 'Product']
a
df.drop( a.index , axis = 0 , inplace = True)
```

In [12]:
```
#

df[df['Product'] == 'Product']
```

Out[12]:

Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address

In [ ]:
```

```

In [13]:
```
# Changing the datatype of "Quantity Ordered" column from 'object' to 'int'

df['Quantity Ordered'] = df['Quantity Ordered'].astype(int)
```

In [14]:
```
df['Price Each'].unique()
```

Out[14]:
```
array(['11.95', '99.99', '600', '11.99', '1700', '14.95', '389.99',
       '3.84', '150', '2.99', '700', '300', '149.99', '109.99', '600.0',
       '999.99', '400', '379.99', '700.0', '1700.0', '150.0', '300.0',
       '400.0'], dtype=object)
```

In [ ]:
```

```

In [15]:
```
# Changing the datatype of "Price Each" column from 'object' to 'float'

df['Price Each'] = df['Price Each'].astype(float)
```

In [16]:
```
# Trimming the address and keeping only the city name:

df['Purchase Address'] = df['Purchase Address'].map(lambda x : x.split(',')[-2].strip())
```

In [17]:
```
1 df.head()
```

Out[17]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	Dallas
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	Boston
3	176560	Google Phone	1	600.00	04/12/19 14:38	Los Angeles
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	Los Angeles
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	Los Angeles

In [18]:
```
1 # Changing the datatype of column 'Ordered Date' from 'object' to 'datetime'
2 # Datetime conversion using astype:
3 df['Order Date'] = df['Order Date'].astype('datetime64[ns]')
```

In [19]:
```
1 df.dtypes
```

Out[19]:
```
Order ID                    object
Product                     object
Quantity Ordered             int32
Price Each                 float64
Order Date          datetime64[ns]
Purchase Address            object
dtype: object
```

In [20]:
```
1 # Inserting a new column in the dataframe:
2
3 df.insert(2 , 'Prod_Category' , np.nan)
```

In [21]:
```
1 # Filling the Prod_Category column with Types of product from "PRODUCT" column
2
3 df['Prod_Category'] = df['Product'].map(lambda x : x.split(' ')[-1])
```

In [22]:
```
df['Prod_Category'].unique()
```

Out[22]:
```
array(['Cable', 'Headphones', 'Phone', 'Laptop', 'Monitor', '(4-pack)',
       'iPhone', 'TV', 'Dryer', 'Machine'], dtype=object)
```

In [23]:
```
# Replacing '(4-pack)' with Batteries:

df['Prod_Category'].replace('(4-pack)', 'Batteries' ,inplace = True)
```

In [24]:
```
df.head()
```

Out[24]:

	Order ID	Product	Prod_Category	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	Cable	2	11.95	2019-04-19 08:46:00	Dallas
2	176559	Bose SoundSport Headphones	Headphones	1	99.99	2019-04-07 22:30:00	Boston
3	176560	Google Phone	Phone	1	600.00	2019-04-12 14:38:00	Los Angeles
4	176560	Wired Headphones	Headphones	1	11.99	2019-04-12 14:38:00	Los Angeles
5	176561	Wired Headphones	Headphones	1	11.99	2019-04-30 09:27:00	Los Angeles

In [25]:
```
# Seperating the 'Order-Date' column into 'Year-Month' ; 'Date' ; 'Hour'

df.insert(6 ,'Year-Month', np.nan)
df.insert(7 , 'Date' , np.nan)
df.insert(8 , 'Hour' , np.nan)
```

In [26]:
```
df['Year-Month'] = df['Order Date'].dt.to_period('M')
df['Date'] = df['Order Date'].dt.day
df['Hour'] = df['Order Date'].dt.hour
```

In [27]:
```
# Dropping the order date from the data frame:
df.drop('Order Date',axis = 1 , inplace = True)

# Making a total bill Column
Bill_Amount = df['Quantity Ordered']*df['Price Each']
df.insert(5 , 'Bill_Amount' , Bill_Amount)
```

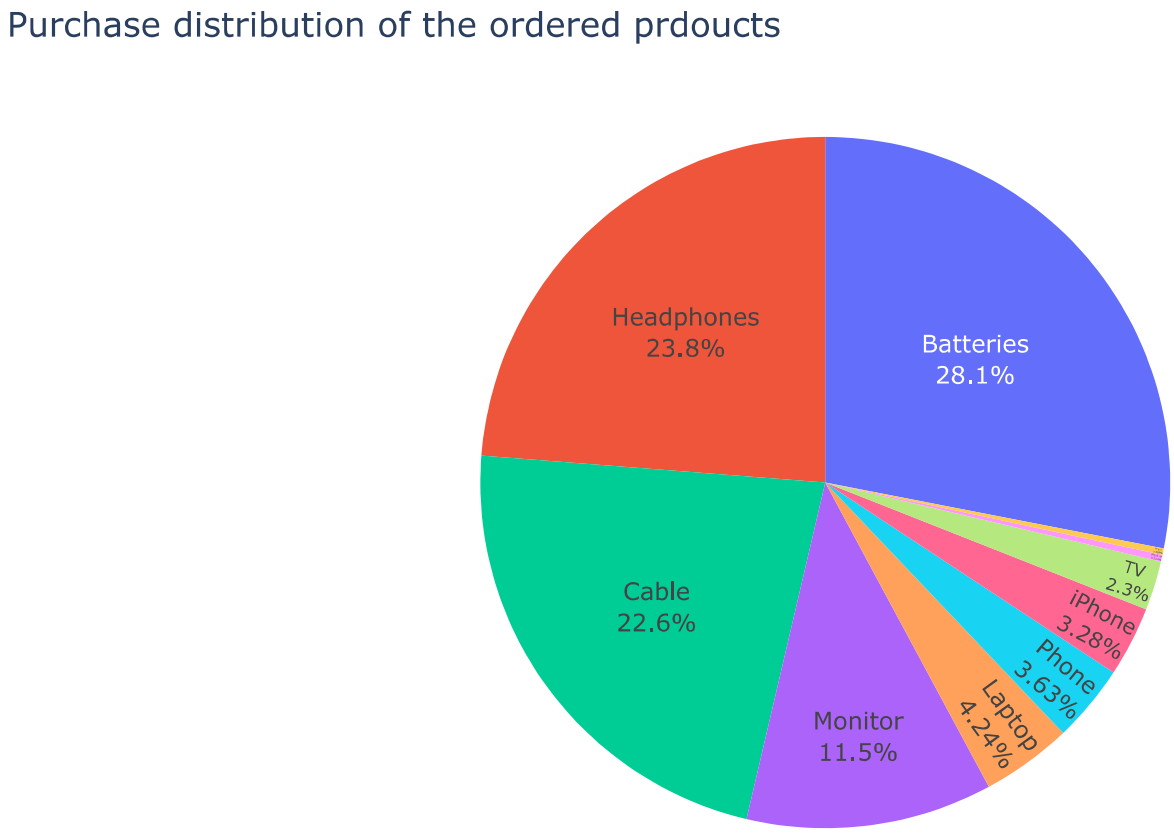
In [28]:
```
1 df.head()
```

Out[28]:

	Order ID	Product	Prod_Category	Quantity Ordered	Price Each	Bill_Amount	Year-Month	Date	Hour	Purchase Address
0	176558	USB-C Charging Cable	Cable	2	11.95	23.90	2019-04	19	8	Dallas
2	176559	Bose SoundSport Headphones	Headphones	1	99.99	99.99	2019-04	7	22	Boston
3	176560	Google Phone	Phone	1	600.00	600.00	2019-04	12	14	Los Angeles
4	176560	Wired Headphones	Headphones	1	11.99	11.99	2019-04	12	14	Los Angeles
5	176561	Wired Headphones	Headphones	1	11.99	11.99	2019-04	30	9	Los Angeles

In [29]:
```
1 # Importing a library:
2
3 import plotly.express as px
```

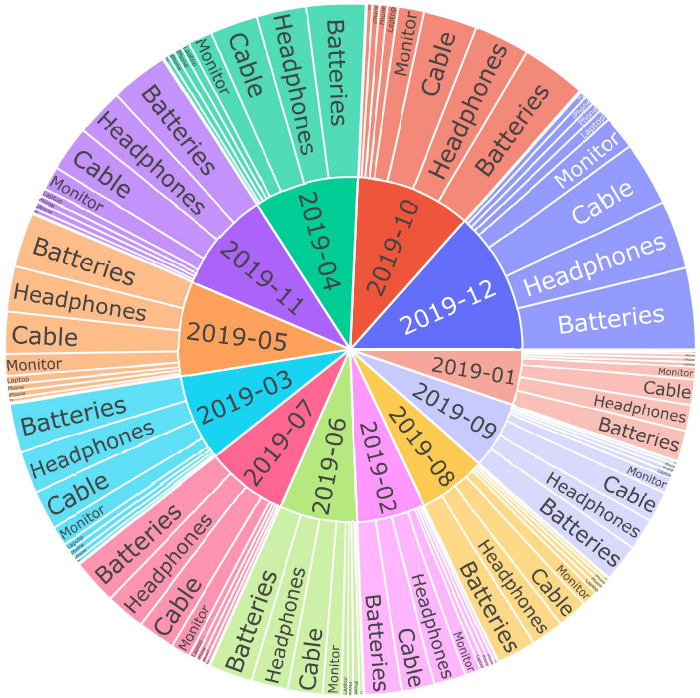
In [30]:
```
fig = px.pie(df , values = 'Quantity Ordered' , names = 'Prod_Category' , title = 'Purchase distribution of
fig.update_traces(textposition='inside', textinfo='percent+label' )
fig.show()
```



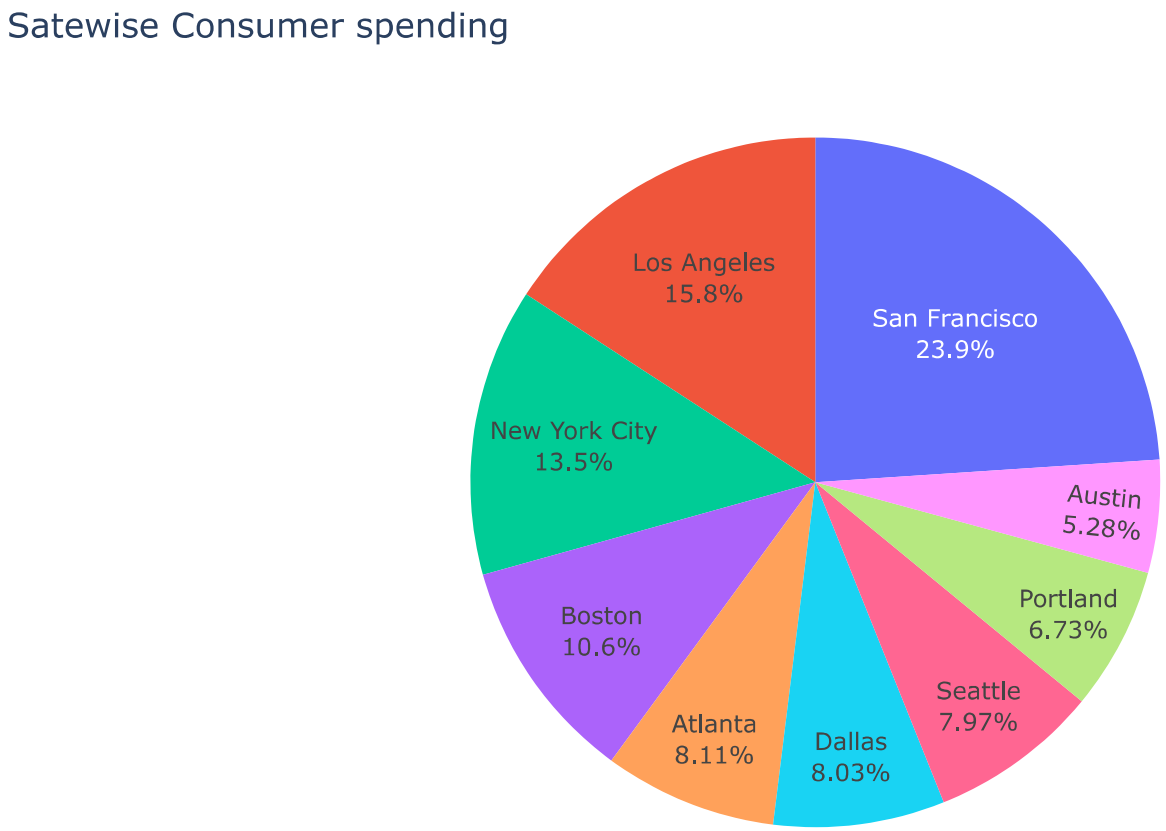
In [ ]:
```
1
```

In [31]:
```
fig = px.sunburst(df, path=['Year-Month', 'Prod_Category'], values='Quantity Ordered',
                  title = 'Monthly sales Distribution')
fig.show()
```

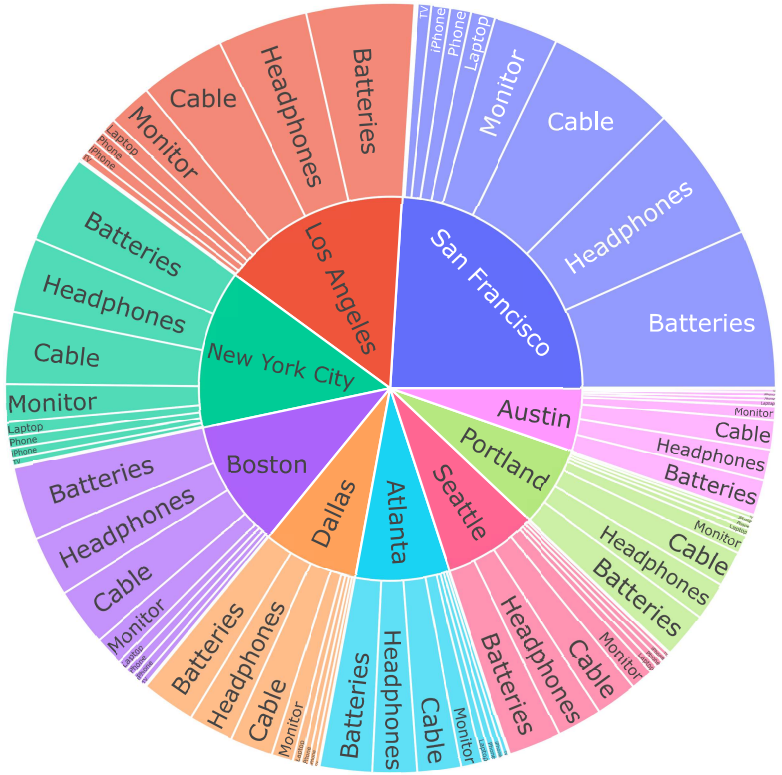
Monthly sales Distribution



In [32]:
```
fig = px.pie( df , values = 'Price Each' , names = 'Purchase Address' , title = 'Satewise Consumer spending
fig.update_traces(textposition = 'inside' , textinfo = 'percent+label' )
fig.show()
```



In [33]:
```
fig = px.sunburst( df , path = ['Purchase Address' , 'Prod_Category'] , values = 'Quantity Ordered')
fig.show()
```

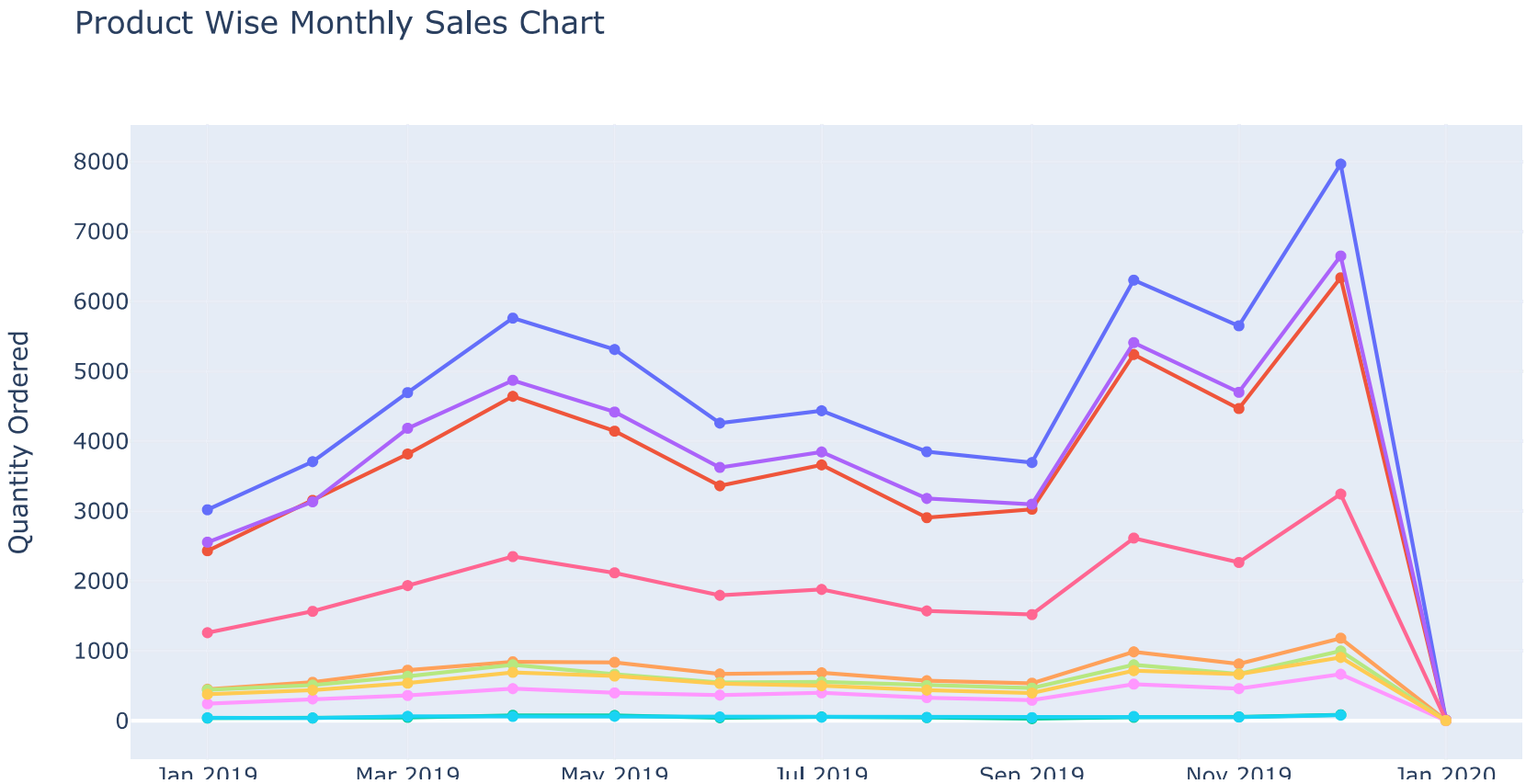


In [ ]:
```

```

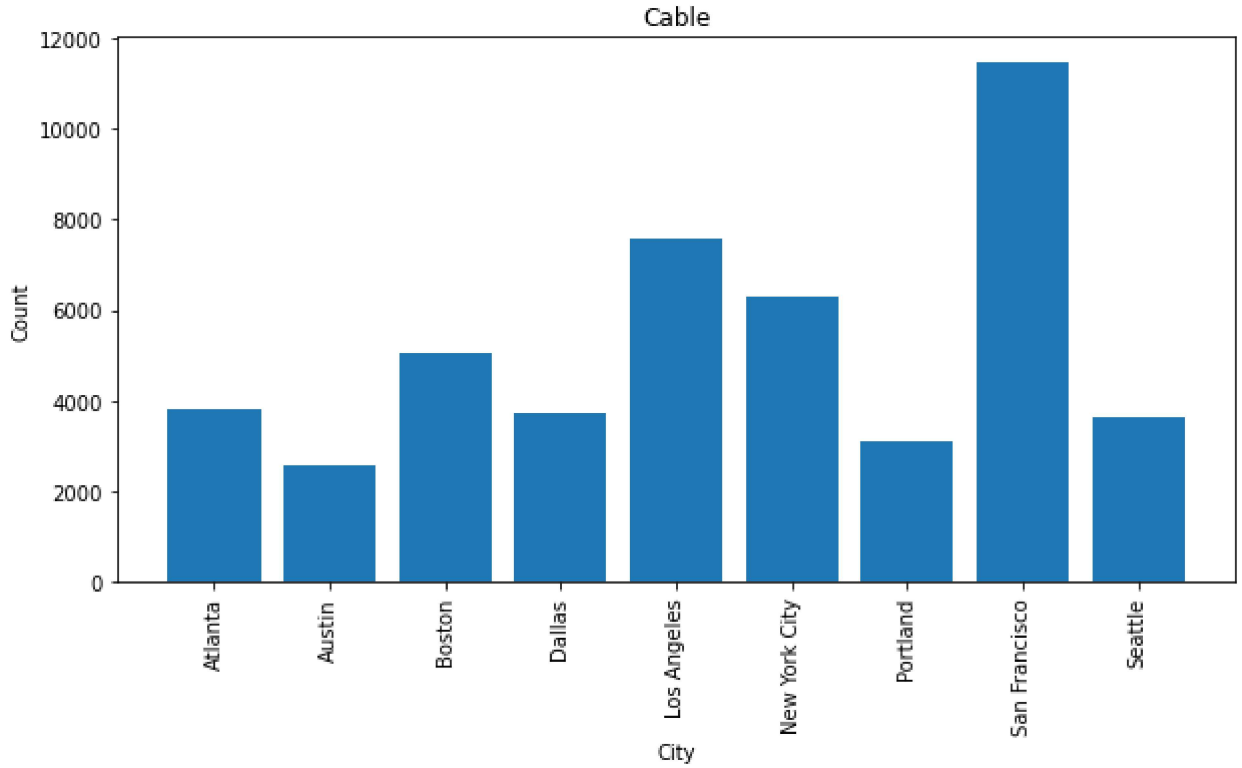
In [34]:
```
df1 = df.groupby(['Year-Month', 'Prod_Category'])['Quantity Ordered'].sum()
df1 = pd.DataFrame(df1)
df1.reset_index(inplace = True)
df1['Year-Month'] = df1['Year-Month'].astype(str)
```

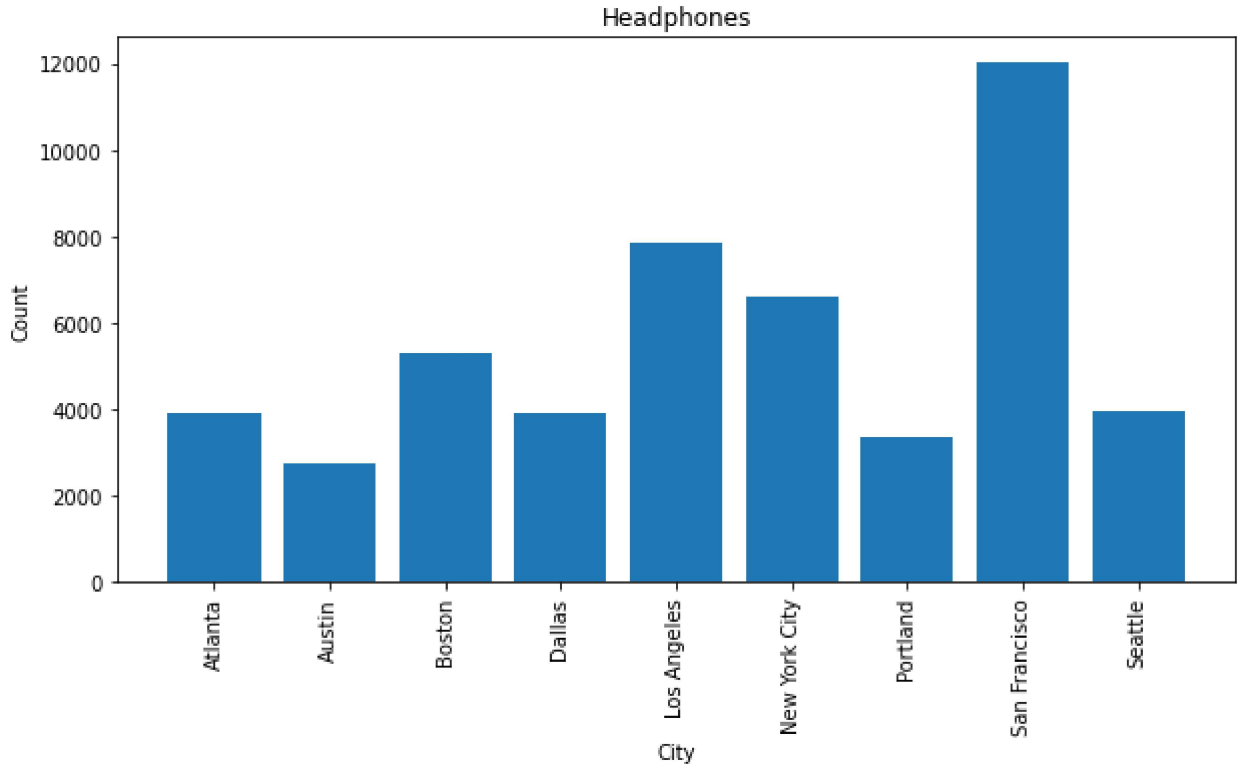
In [35]:
```
fig = px.line(df1, x = 'Year-Month' , y = 'Quantity Ordered' , color = 'Prod_Category', markers = True , ti
fig.show()
```

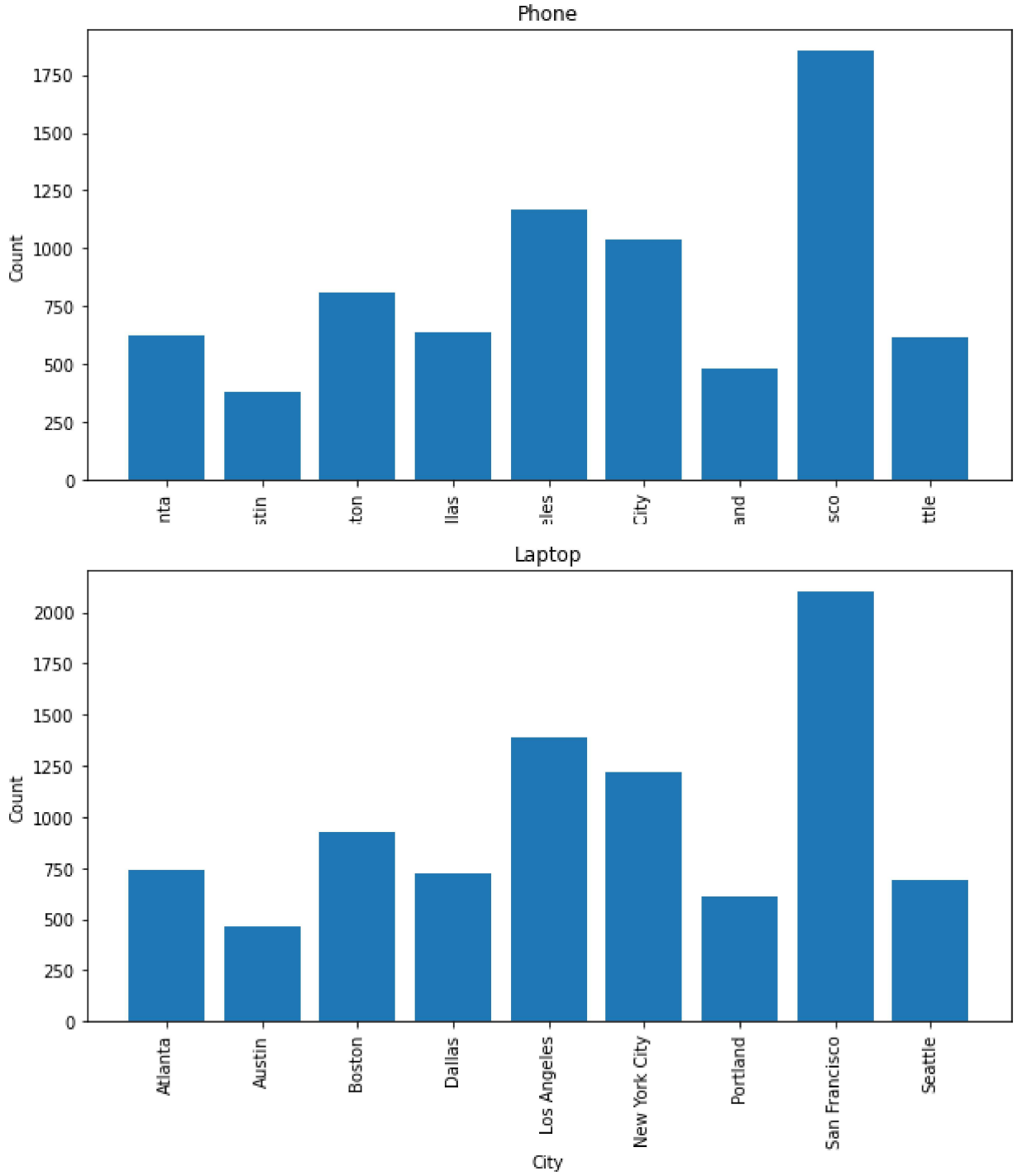


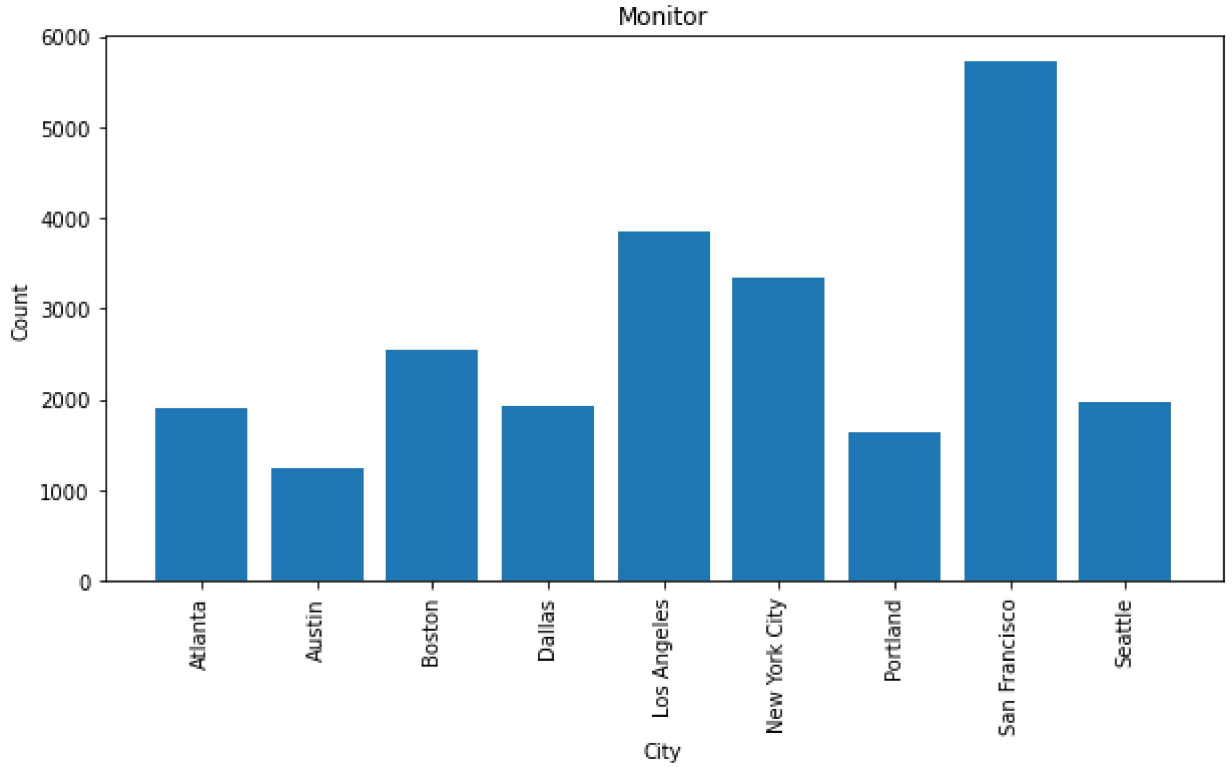
In [36]:

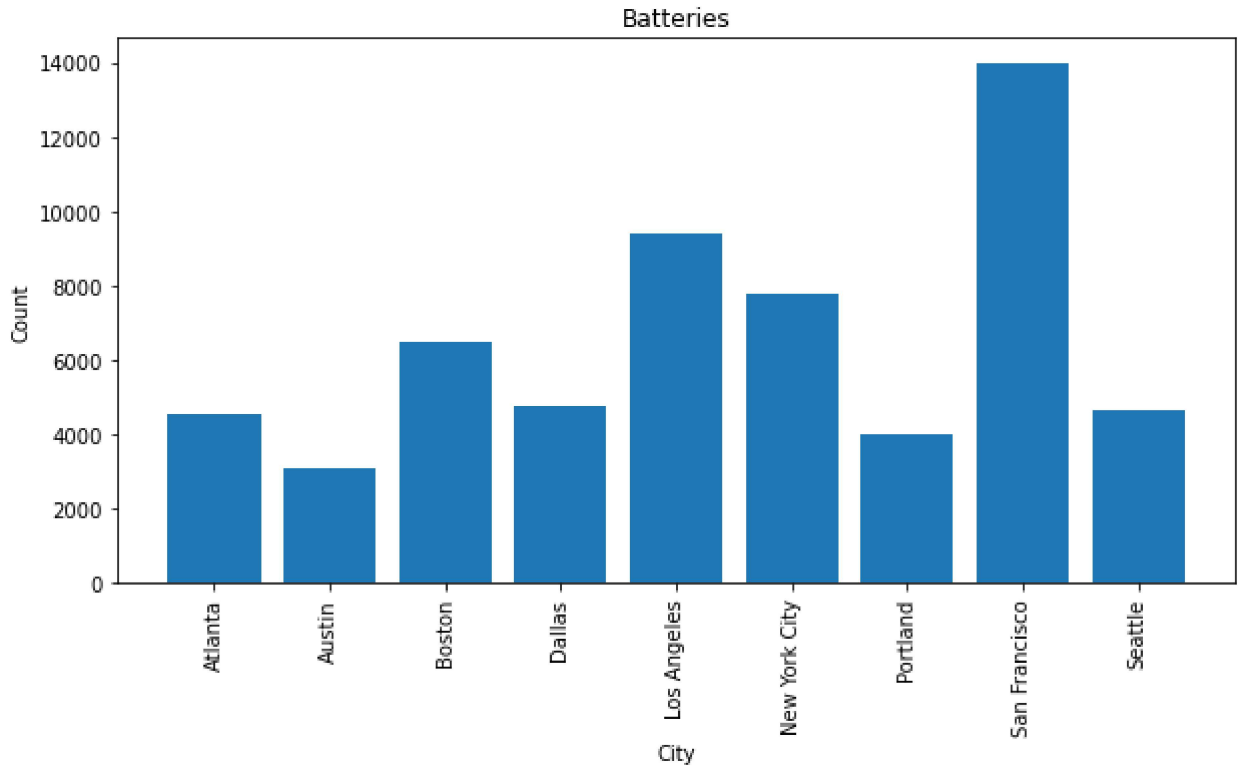
```
for i,v in enumerate(df['Prod_Category'].unique()):
    plt.figure(figsize = (10,65))
    plt.subplot(11,1,i+1)
    df1 = df.groupby(['Prod_Category' , 'Purchase Address'])['Quantity Ordered'].sum()
    df1 = pd.DataFrame(df1)
    df1.reset_index(inplace = True)
    df2 = df1[df1['Prod_Category'] == v]
    plt.bar(list(df2['Purchase Address']) , list(df2['Quantity Ordered']) )
    plt.xticks(rotation = 90 )
    plt.xlabel('City')
    plt.ylabel('Count')
    plt.title(v)
plt.show()
```

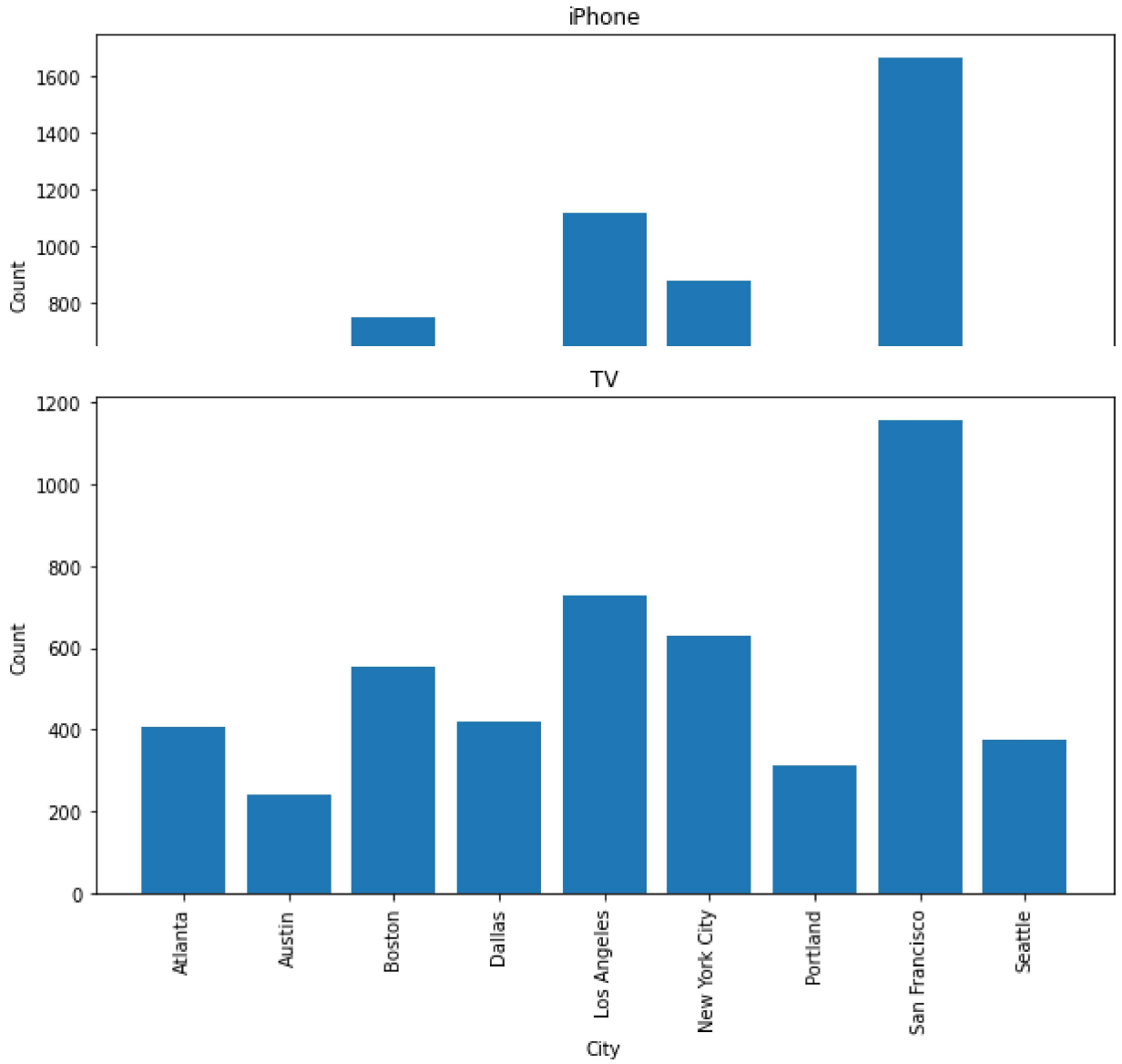


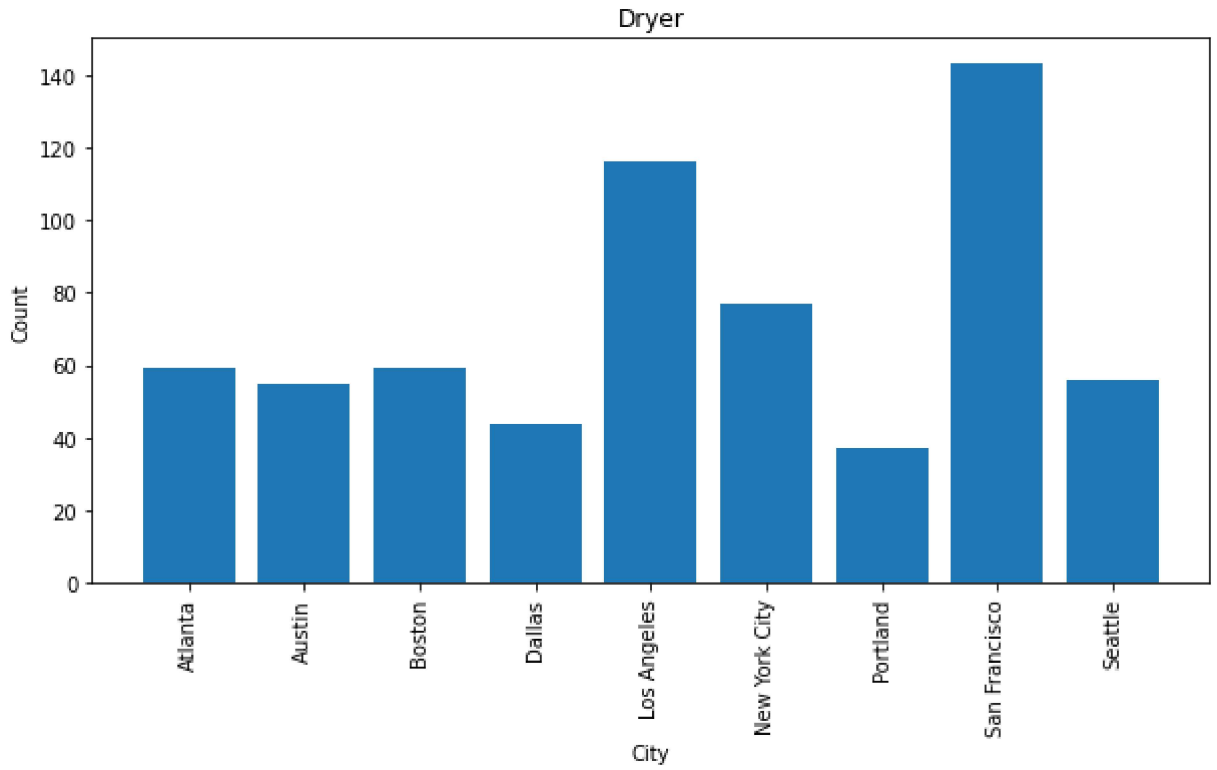

Headphones
Count
12000
10000
8000
6000
4000
2000
0
Atlanta
Austin
Boston
Dallas
Los Angeles
New York City
Portland
San Francisco
Seattle
City

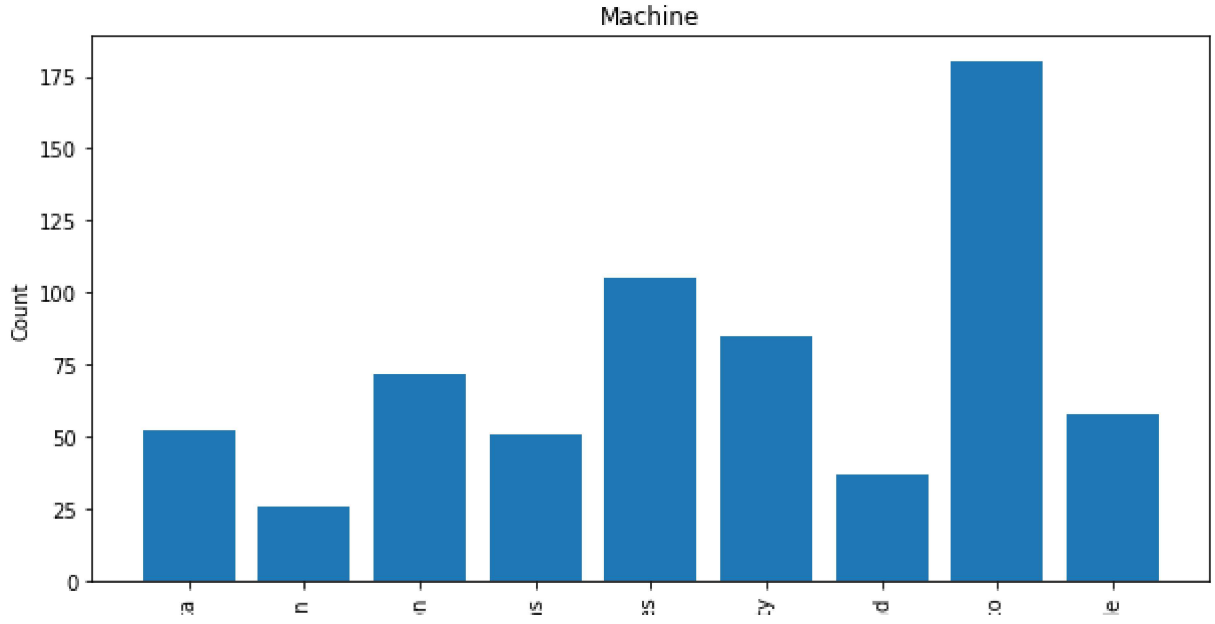

Phone
Count
1750
1500
1250
1000
750
500
250
0
nta
stin
ton
llas
les
City
and
sco
ttle
Laptop
Count
2000
1750
1500
1250
1000
750
500
250
0
Atlanta
Austin
Boston
Dallas
Los Angeles
New York City
Portland
San Francisco
Seattle
City


Monitor
Count
6000
5000
4000
3000
2000
1000
0
Atlanta
Austin
Boston
Dallas
Los Angeles
New York City
Portland
San Francisco
Seattle
City


Batteries
Count
14000
12000
10000
8000
6000
4000
2000
0
Atlanta
Austin
Boston
Dallas
Los Angeles
New York City
Portland
San Francisco
Seattle
City


iPhone
Count
1600
1400
1200
1000
800
TV
Count
1200
1000
800
600
400
200
0
Atlanta
Austin
Boston
Dallas
Los Angeles
New York City
Portland
San Francisco
Seattle
City


Dryer
Count
0
20
40
60
80
100
120
140
Atlanta
Austin
Boston
Dallas
Los Angeles
New York City
Portland
San Francisco
Seattle
City



In [ ]:
```
1
```