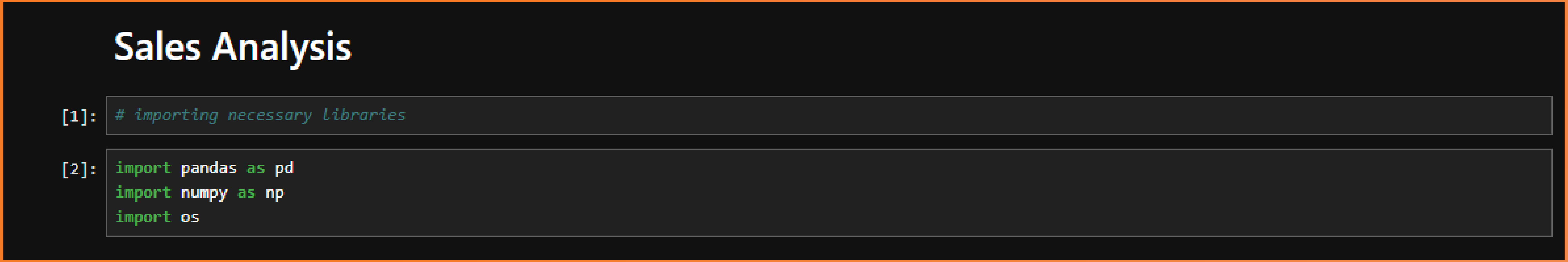
SALES ANALYSIS IN PYHTON

❑ Importing necessary libraries to analyse the data

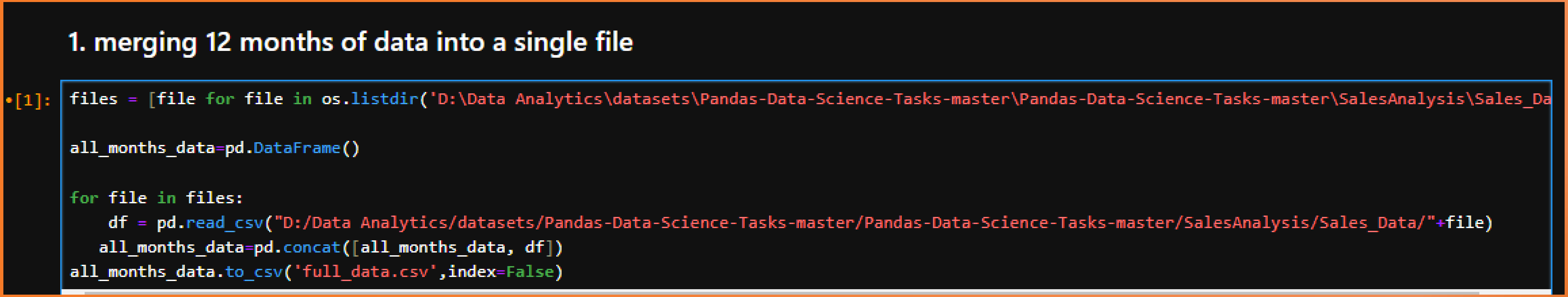


Sales Analysis

```
[1]: # importing necessary libraries
```

```
[2]: import pandas as pd
     import numpy as np
     import os
```

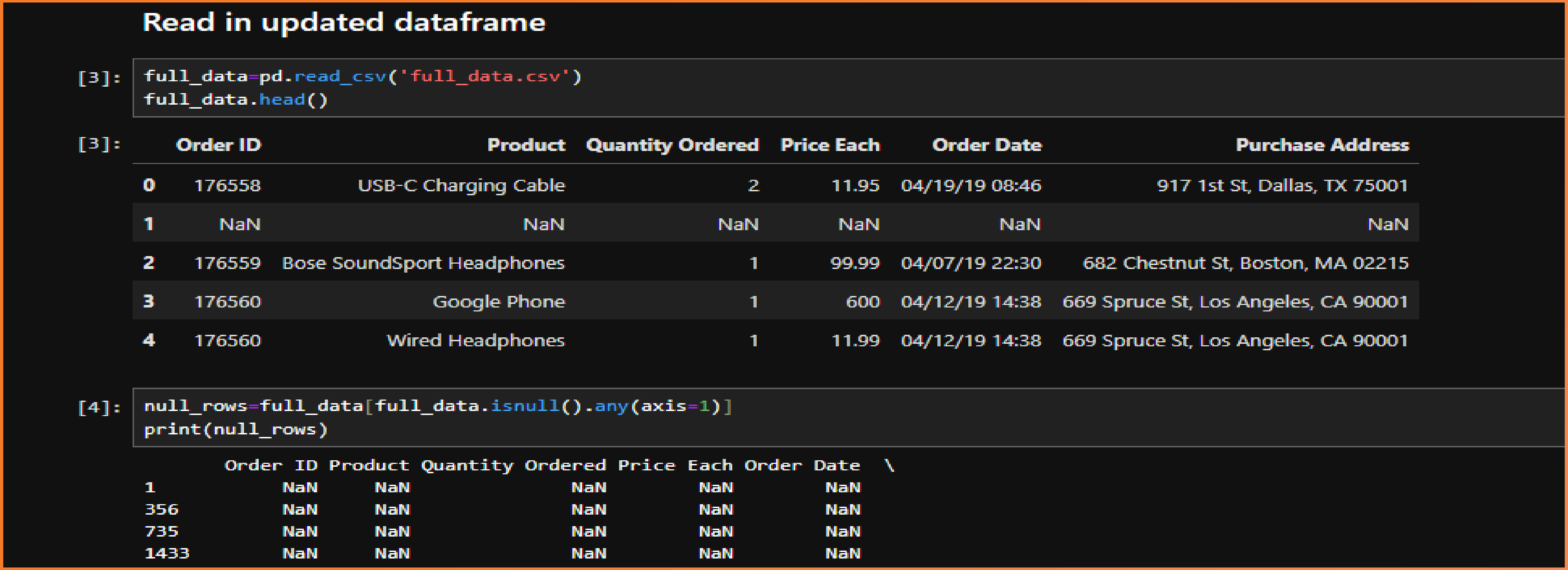
❑ Merging 12 csv files into one to create a single data frame



1. merging 12 months of data into a single file

```
•[1]: files = [file for file in os.listdir('D:\Data Analytics\datasets\Pandas-Data-Science-Tasks-master\Pandas-Data-Science-Tasks-master\SalesAnalysis\Sales_Da

      all_months_data=pd.DataFrame()

      for file in files:
          df = pd.read_csv("D:/Data Analytics/datasets/Pandas-Data-Science-Tasks-master/Pandas-Data-Science-Tasks-master/SalesAnalysis/Sales_Data/"+file)
         all_months_data=pd.concat([all_months_data, df])
      all_months_data.to_csv('full_data.csv',index=False)
```

- Reading the data in new data frame as to create a new combined file then identifying null rows.



Read in updated dataframe

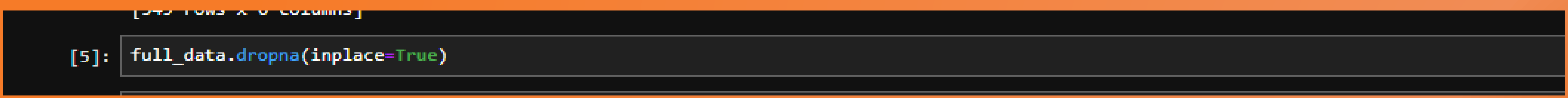
```
[3]: full_data=pd.read_csv('full_data.csv')
     full_data.head()
```

[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

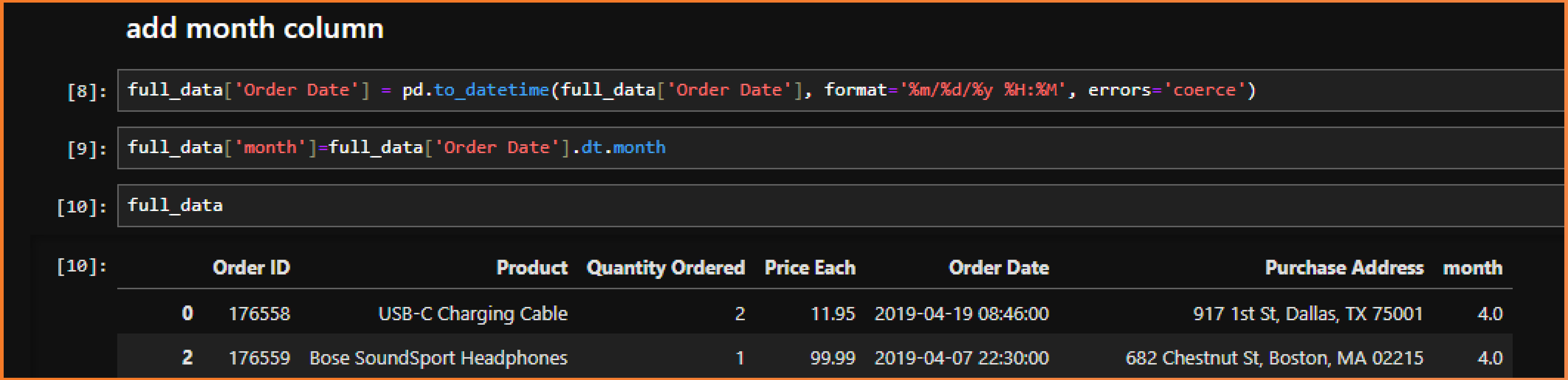
```
[4]: null_rows=full_data[full_data.isnull().any(axis=1)]
     print(null_rows)

              Order ID Product Quantity Ordered Price Each Order Date  \
     1             NaN     NaN              NaN        NaN        NaN
     356           NaN     NaN              NaN        NaN        NaN
     735           NaN     NaN              NaN        NaN        NaN
     1433          NaN     NaN              NaN        NaN        NaN
```

- Deleting the null rows as there are few number of them.



```
     [545 rows x 6 columns]

[5]: full_data.dropna(inplace=True)
```

❑ Adding ‘month’ and ‘Sales’ column



add month column

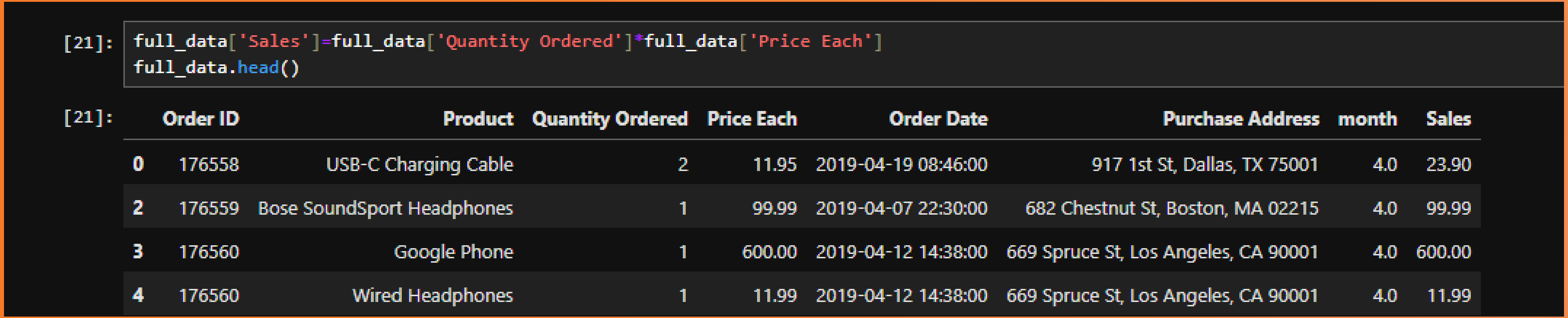
```
[8]: full_data['Order Date'] = pd.to_datetime(full_data['Order Date'], format='%m/%d/%y %H:%M', errors='coerce')
```

```
[9]: full_data['month']=full_data['Order Date'].dt.month
```

```
[10]: full_data
```

[10]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4.0
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4.0

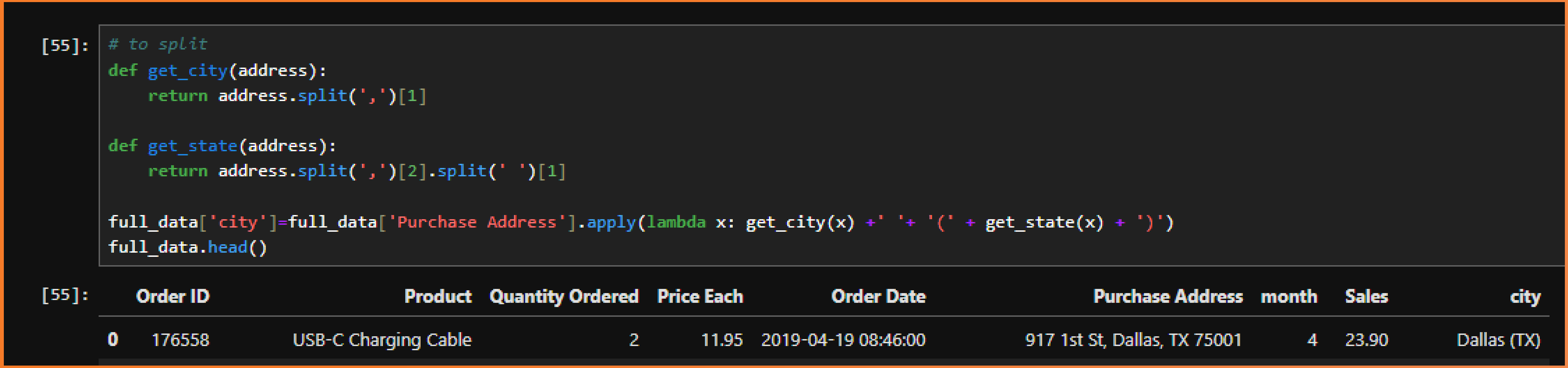


```
[21]: full_data['Sales']=full_data['Quantity Ordered']*full_data['Price Each']
      full_data.head()
```

[21]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month	Sales
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4.0	23.90
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4.0	99.99
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4.0	600.00
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4.0	11.99

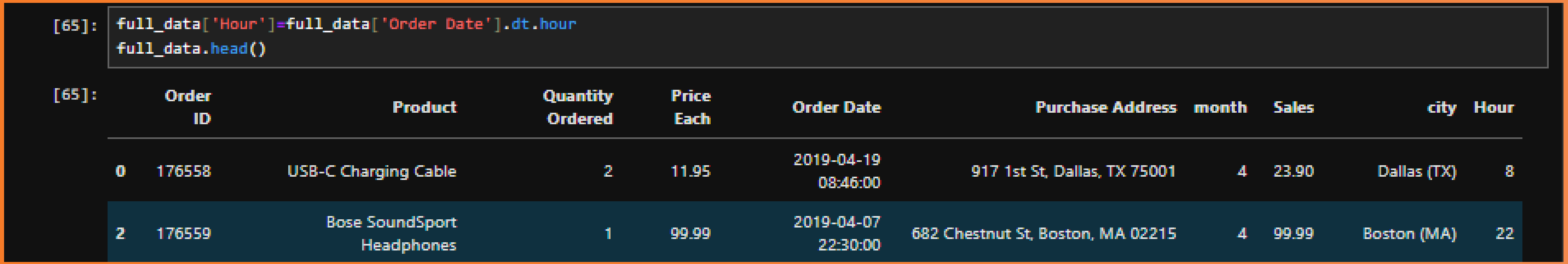
❑ Extracting ‘City’, ‘Hour’, ‘minute’ from their respective columns



[55]:
```
# to split
def get_city(address):
    return address.split(',')[1]

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

full_data['city']=full_data['Purchase Address'].apply(lambda x: get_city(x) +' '+ '(' + get_state(x) + ')')
full_data.head()
```

[55]:

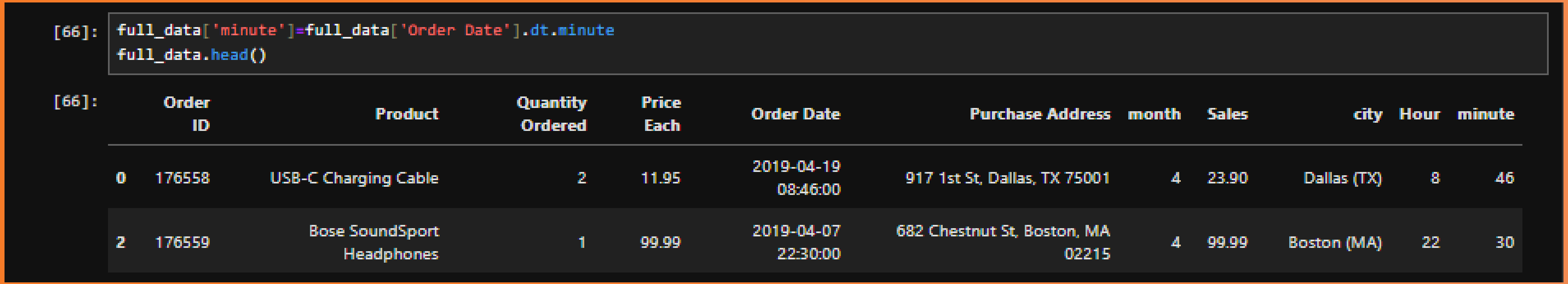
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month	Sales	city
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)



[65]:
```
full_data['Hour']=full_data['Order Date'].dt.hour
full_data.head()
```

[65]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month	Sales	city	Hour
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)	8
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)	22

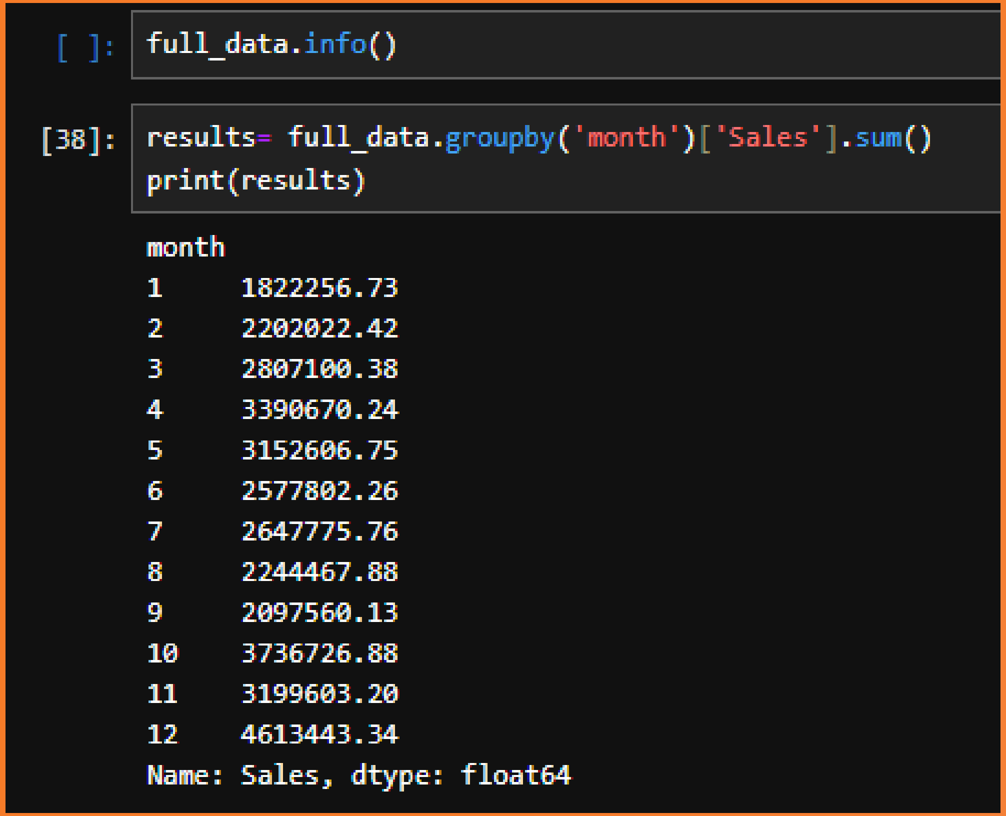


[66]:
```
full_data['minute']=full_data['Order Date'].dt.minute
full_data.head()
```

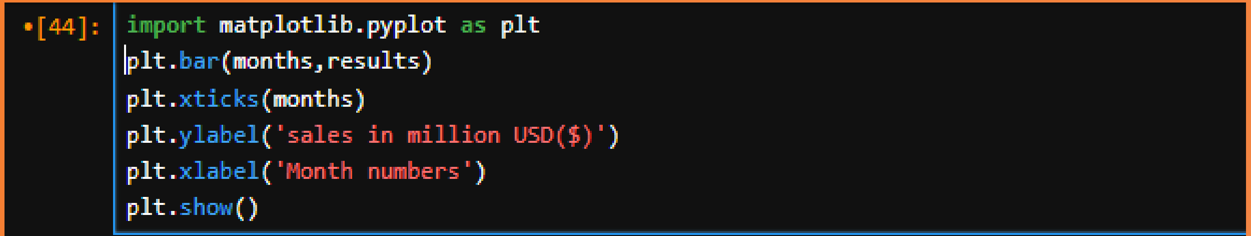
[66]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month	Sales	city	Hour	minute
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	23.90	Dallas (TX)	8	46
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	99.99	Boston (MA)	22	30

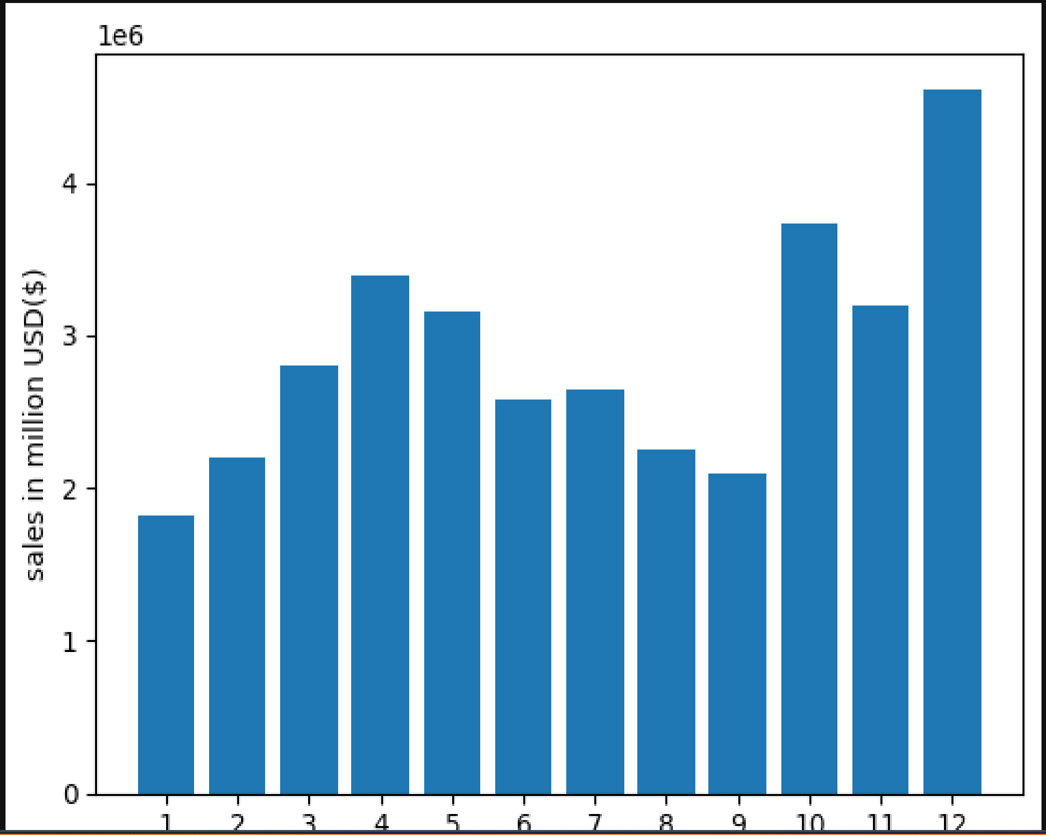
Q.1 What was the best month for sales? How much was earned that month?



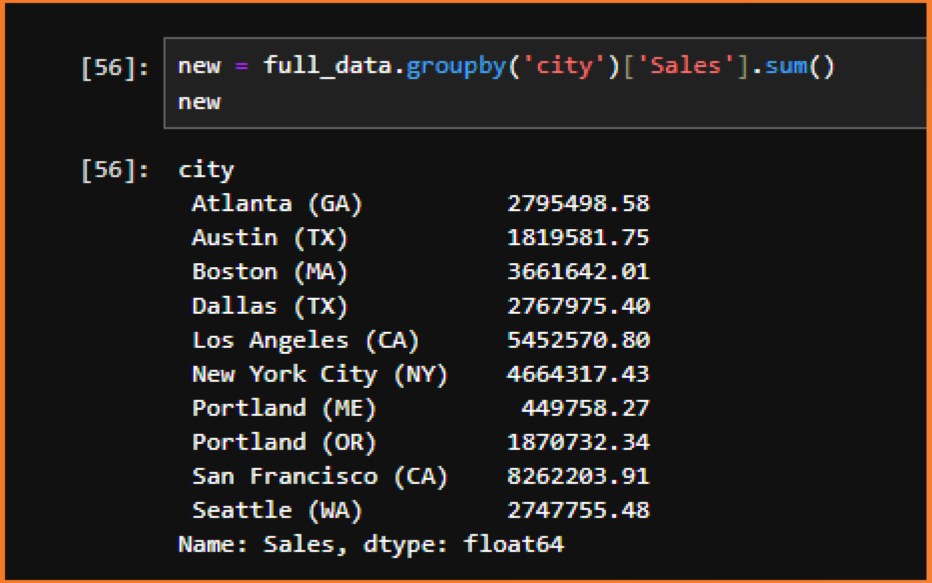
```
[ ]: full_data.info()

[38]: results= full_data.groupby('month')['Sales'].sum()
      print(results)

      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
      Name: Sales, dtype: float64
```



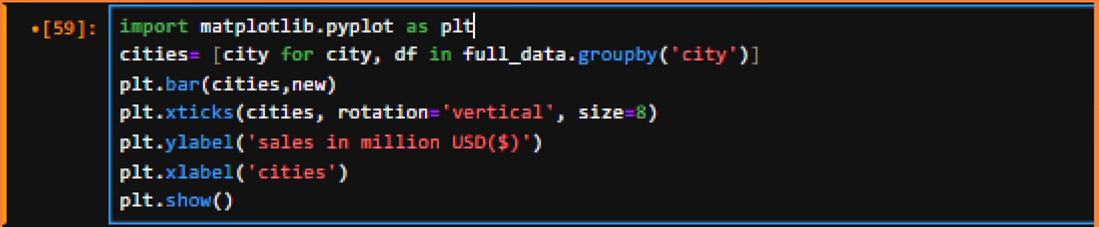
```
•[44]: import matplotlib.pyplot as plt
       plt.bar(months,results)
       plt.xticks(months)
       plt.ylabel('sales in million USD($)')
       plt.xlabel('Month numbers')
       plt.show()
```



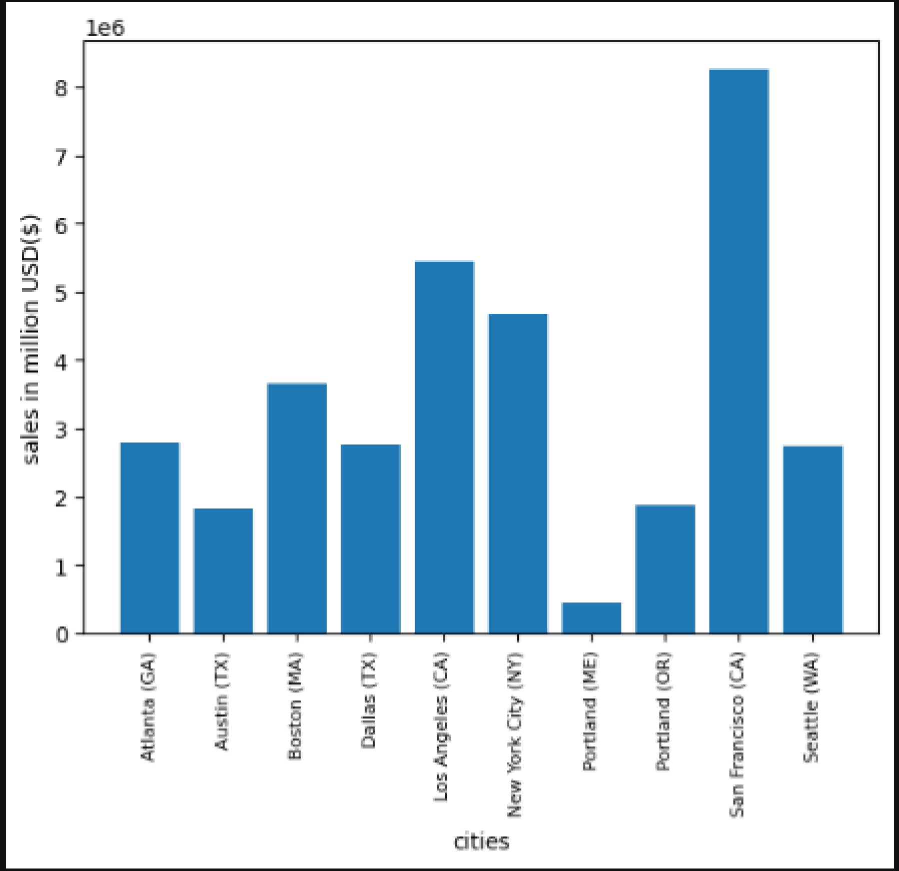
Q.2 What city have a highest number of sales?



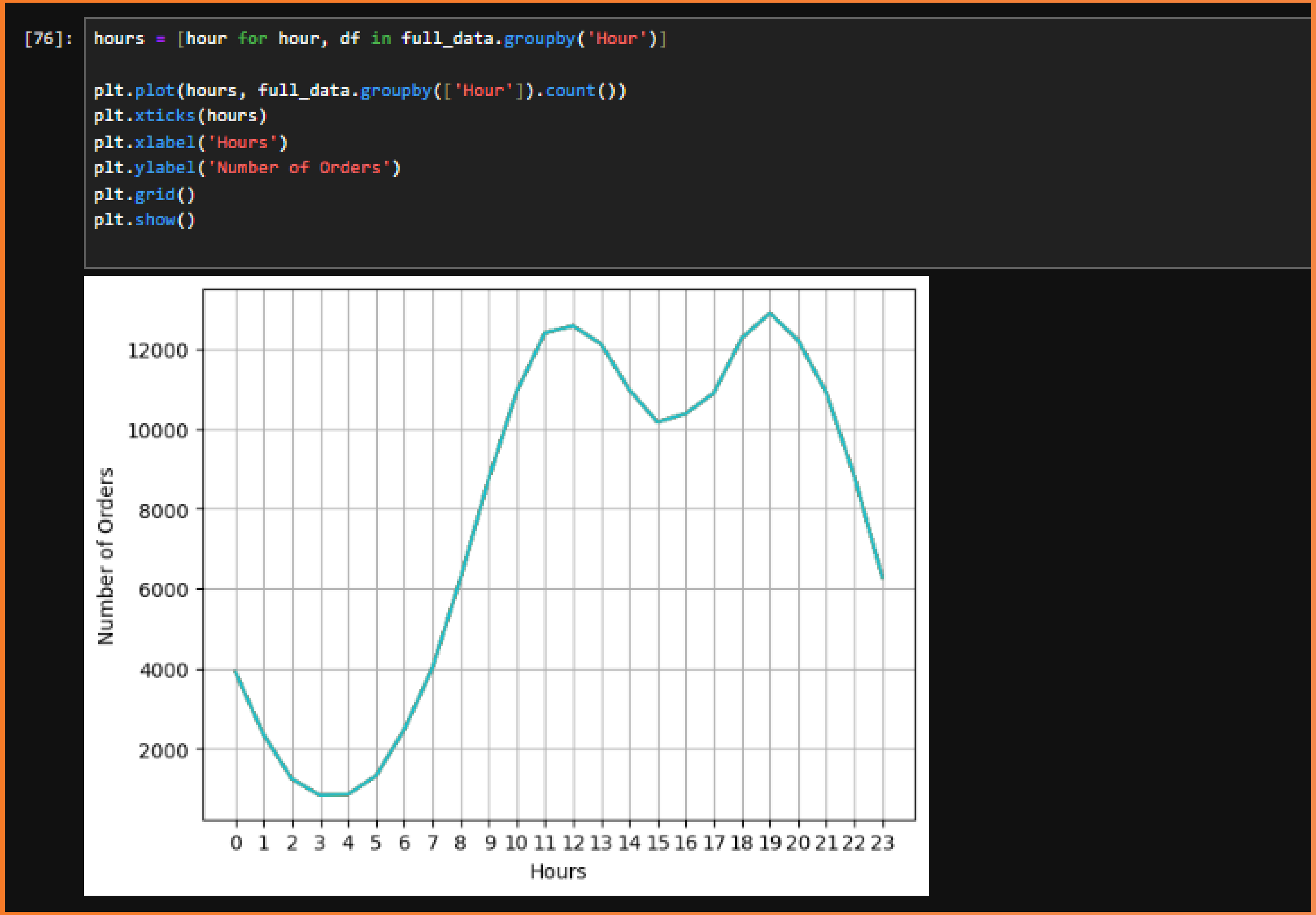
```
[56]: new = full_data.groupby('city')['Sales'].sum()
      new

[56]: city
       Atlanta (GA)          2795498.58
       Austin (TX)           1819581.75
       Boston (MA)           3661642.01
       Dallas (TX)           2767975.40
       Los Angeles (CA)      5452570.80
       New York City (NY)    4664317.43
       Portland (ME)          449758.27
       Portland (OR)         1870732.34
       San Francisco (CA)    8262203.91
       Seattle (WA)          2747755.48
      Name: Sales, dtype: float64
```



```
•[59]: import matplotlib.pyplot as plt
       cities= [city for city, df in full_data.groupby('city')]
       plt.bar(cities,new)
       plt.xticks(cities, rotation='vertical', size=8)
       plt.ylabel('sales in million USD($)')
       plt.xlabel('cities')
       plt.show()
```

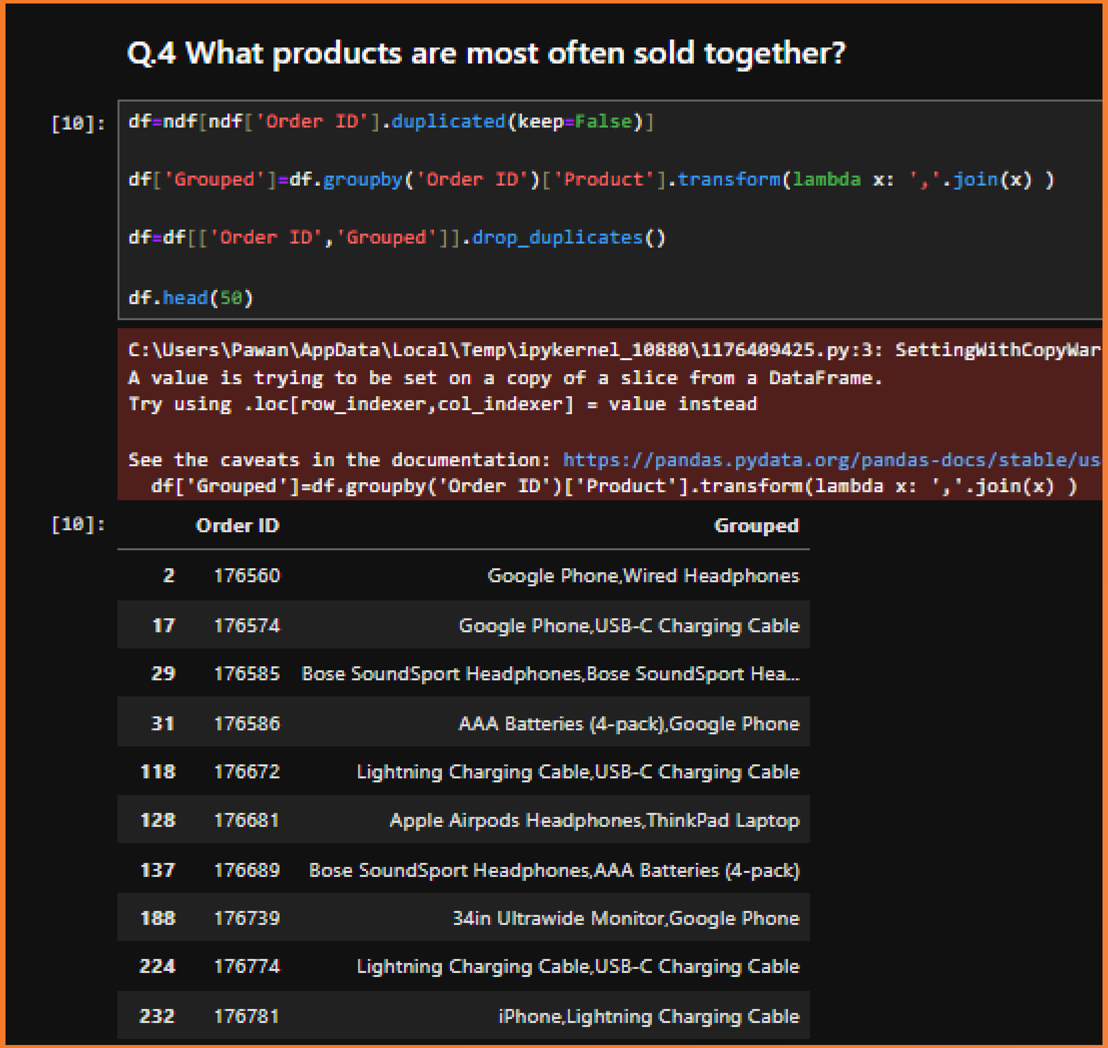


Q.3 What time should we display advertisements to maximaze the likelihood of the customer's buying product?



```
[76]: hours = [hour for hour, df in full_data.groupby('Hour')]

      plt.plot(hours, full_data.groupby(['Hour']).count())
      plt.xticks(hours)
      plt.xlabel('Hours')
      plt.ylabel('Number of Orders')
      plt.grid()
      plt.show()
```

Q.4 What products are most often sold together?



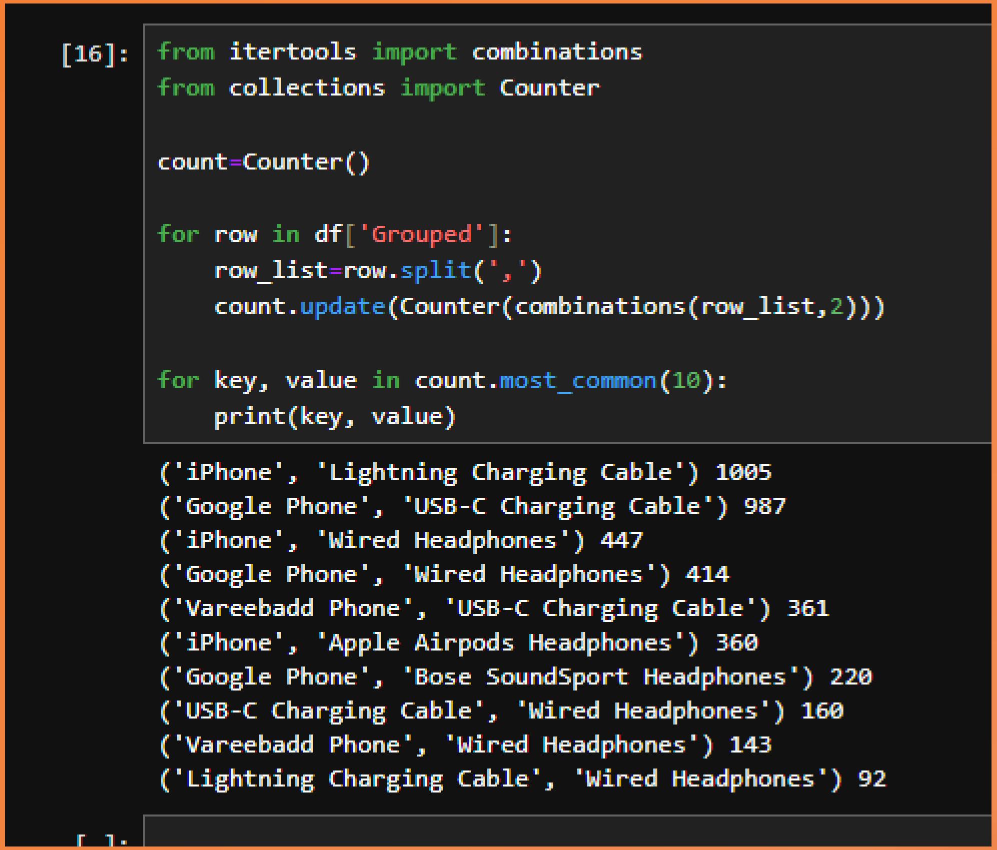
Q.4 What products are most often sold together?

```
[10]: df=ndf[ndf['Order ID'].duplicated(keep=False)]

      df['Grouped']=df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x) )

      df=df[['Order ID','Grouped']].drop_duplicates()

      df.head(50)
```

```
C:\Users\Pawan\AppData\Local\Temp\ipykernel_10880\1176409425.py:3: SettingWithCopyWar
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  df['Grouped']=df.groupby('Order ID')['Product'].transform(lambda x: ','.join(x) )
```

[10]:

	Order ID	Grouped
2	176560	Google Phone,Wired Headphones
17	176574	Google Phone,USB-C Charging Cable
29	176585	Bose SoundSport Headphones,Bose SoundSport Hea...
31	176586	AAA Batteries (4-pack),Google Phone
118	176672	Lightning Charging Cable,USB-C Charging Cable
128	176681	Apple Airpods Headphones,ThinkPad Laptop
137	176689	Bose SoundSport Headphones,AAA Batteries (4-pack)
188	176739	34in Ultrawide Monitor,Google Phone
224	176774	Lightning Charging Cable,USB-C Charging Cable
232	176781	iPhone,Lightning Charging Cable

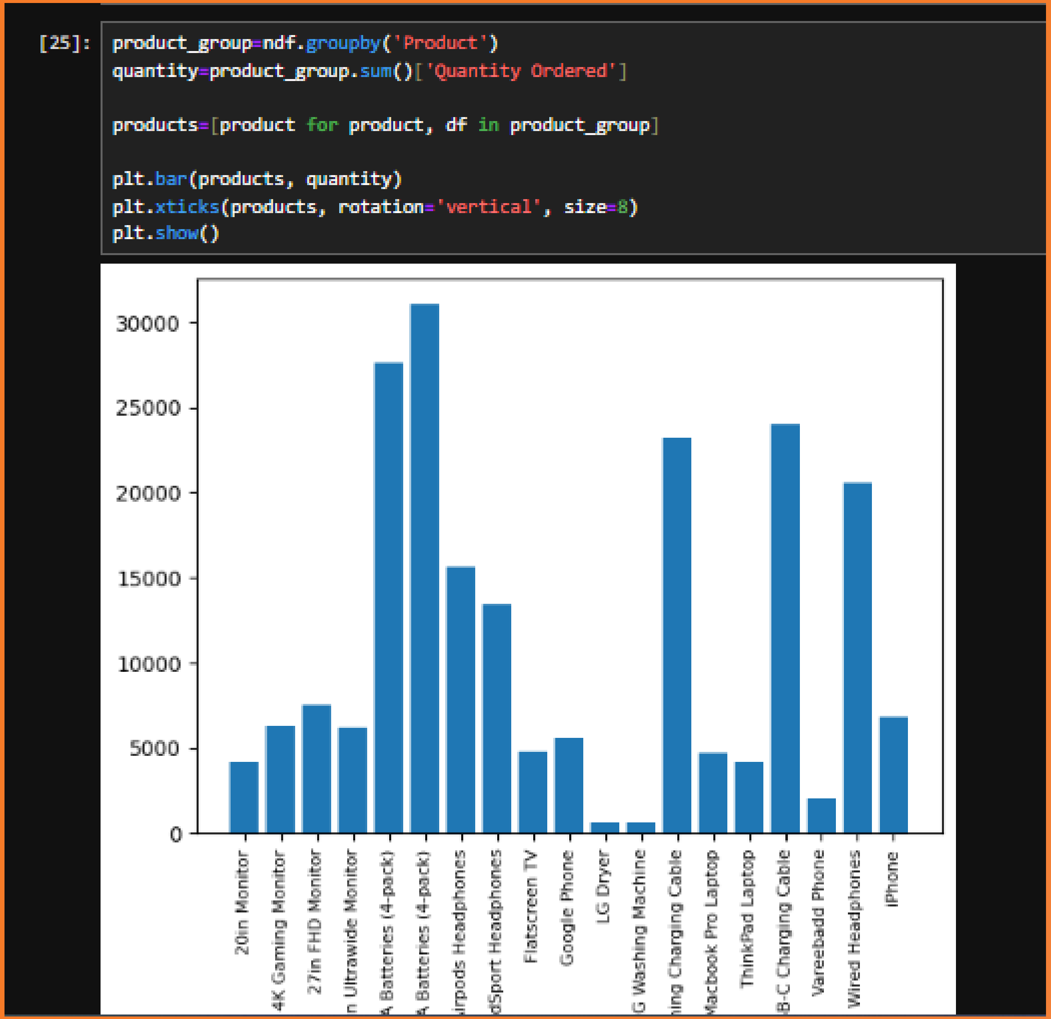


```
[16]: from itertools import combinations
      from collections import Counter

      count=Counter()

      for row in df['Grouped']:
          row_list=row.split(',')
          count.update(Counter(combinations(row_list,2)))

      for key, value in count.most_common(10):
          print(key, value)
```

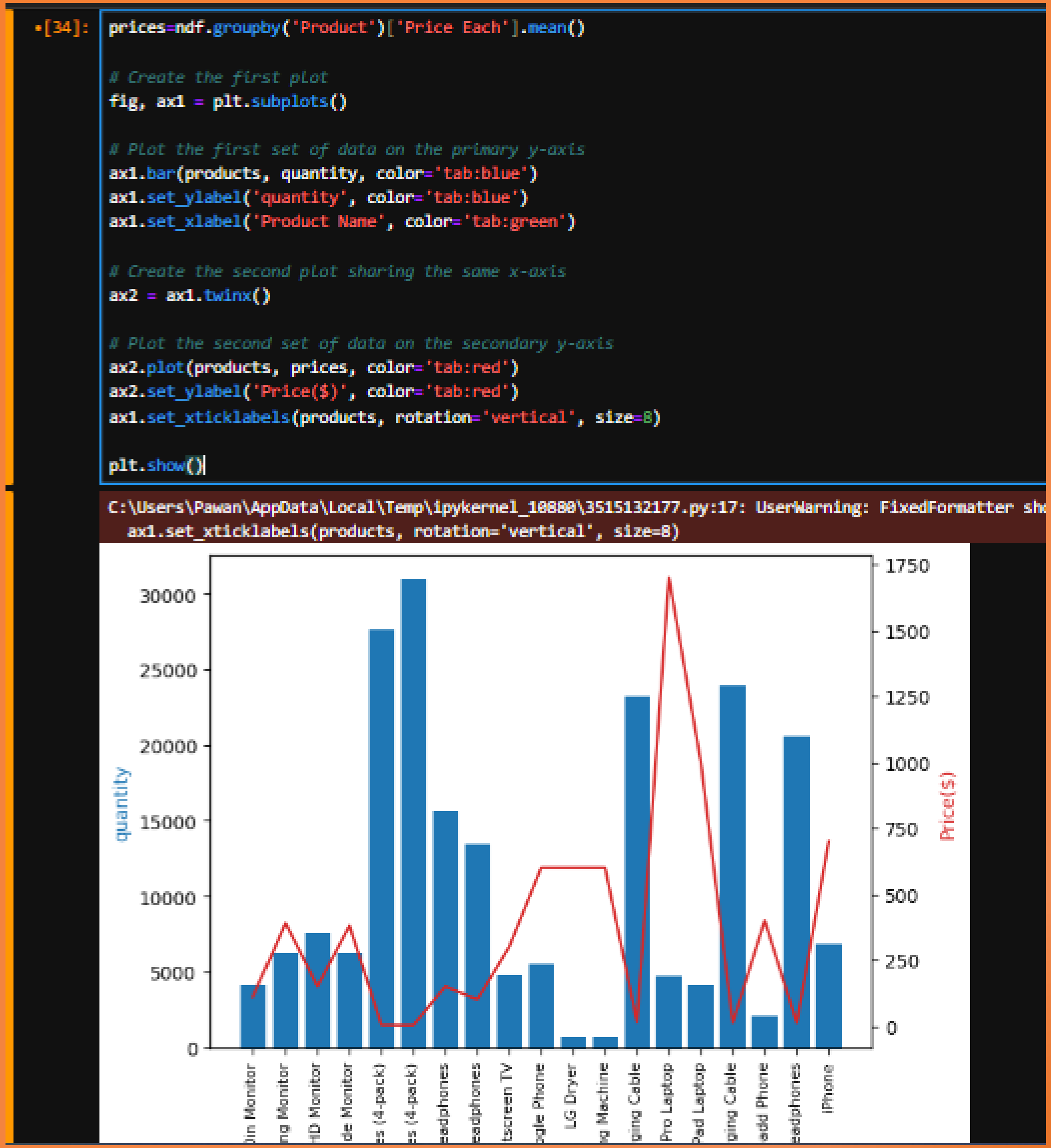
```
('iPhone', 'Lightning Charging Cable') 1005
('Google Phone', 'USB-C Charging Cable') 987
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 414
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple Airpods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 160
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92
```

[ ]:

Q.5 What products sold the most? Why do you think it sold the most?



```
[25]: product_group=ndf.groupby('Product')
      quantity=product_group.sum()['Quantity Ordered']

      products=[product for product, df in product_group]

      plt.bar(products, quantity)
      plt.xticks(products, rotation='vertical', size=8)
      plt.show()
```



```
[34]: prices=ndf.groupby('Product')['Price Each'].mean()

      # Create the first plot
      fig, ax1 = plt.subplots()

      # Plot the first set of data on the primary y-axis
      ax1.bar(products, quantity, color='tab:blue')
      ax1.set_ylabel('quantity', color='tab:blue')
      ax1.set_xlabel('Product Name', color='tab:green')

      # Create the second plot sharing the same x-axis
      ax2 = ax1.twinx()

      # Plot the second set of data on the secondary y-axis
      ax2.plot(products, prices, color='tab:red')
      ax2.set_ylabel('Price($)', color='tab:red')
      ax1.set_xticklabels(products, rotation='vertical', size=8)

      plt.show()
```

```
C:\Users\Pawan\AppData\Local\Temp\ipykernel_10880\3515132177.py:17: UserWarning: FixedFormatter sh
  ax1.set_xticklabels(products, rotation='vertical', size=8)
```

INSIGHTS

- Timing is crucial! Focus on peak buying hours to maximize customer engagement.
- Consider city-specific marketing strategies to capitalize on regional sales patterns.
- Bundle frequently sold products together to encourage cross-selling opportunities.
- Give special discounts and deals on pricy items if possible, to increase sales.
- Increase the stock of cheap items as they can run out of stock.