sales_analysis_project

March 16, 2023

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

```
[3]: df_jan =pd.read_csv("C:
      ↪\\Users\\USER\\Desktop\\workspace\\Sales_Data\\Sales_January_2019.csv")
     df_jan
```

```
[3]:      Order ID                     Product Quantity Ordered Price Each  \
     0      141234                      iPhone                1        700
     1      141235    Lightning Charging Cable                1      14.95
     2      141236            Wired Headphones                2      11.99
     3      141237            27in FHD Monitor                1     149.99
     4      141238            Wired Headphones                1      11.99
     …         …                           …               …          …
     9718   150497                20in Monitor                1     109.99
     9719   150498            27in FHD Monitor                1     149.99
     9720   150499             ThinkPad Laptop                1     999.99
     9721   150500      AAA Batteries (4-pack)                2       2.99
     9722   150501                Google Phone                1        600

               Order Date                       Purchase Address
     0     01/22/19 21:25        944 Walnut St, Boston, MA 02215
     1     01/28/19 14:15       185 Maple St, Portland, OR 97035
     2     01/17/19 13:33  538 Adams St, San Francisco, CA 94016
     3     01/05/19 20:33     738 10th St, Los Angeles, CA 90001
     4     01/25/19 11:59          387 10th St, Austin, TX 73301
     …                …                                   …
     9718  01/26/19 19:09            95 8th St, Dallas, TX 75001
     9719  01/10/19 22:58    403 7th St, San Francisco, CA 94016
     9720  01/21/19 14:31        214 Main St, Portland, OR 97035
     9721  01/15/19 14:21      810 2nd St, Los Angeles, CA 90001
     9722  01/13/19 16:43         428 Cedar St, Boston, MA 02215

     [9723 rows x 6 columns]
```

```
[4]: #Since we have sales data for a year stored on monthly basis, we'll iterate␣
      ↪through the directory to get alll the monthly data.
```

```
files = [file for file in os.listdir("C:
 ↪\\Users\\USER\\Desktop\\workspace\\Sales_Data")]
for file in files:
    print(file)
```

```
Sales_April_2019.csv
Sales_August_2019.csv
Sales_December_2019.csv
Sales_February_2019.csv
Sales_January_2019.csv
Sales_July_2019.csv
Sales_June_2019.csv
Sales_March_2019.csv
Sales_May_2019.csv
Sales_November_2019.csv
Sales_October_2019.csv
Sales_September_2019.csv
```

[5]:
```
#We're going to create an empty dataframe so that we can append the data from␣
 ↪all months to the dataframe.
#Our aim is to have the data for a year in a single dataframe.

all_mnths = pd.DataFrame()
for file in files:
    df = pd.read_csv("C:\\Users\\USER\\Desktop\\workspace\\Sales_Data\\"+file)
    all_mnths = pd.concat([all_mnths, df])
all_mnths.head()
```

[5]:
```
  Order ID                     Product Quantity Ordered Price Each  \
0   176558        USB-C Charging Cable                2      11.95
1      NaN                         NaN              NaN        NaN
2   176559 Bose SoundSport Headphones                1      99.99
3   176560               Google Phone                1        600
4   176560           Wired Headphones                1      11.99

       Order Date                      Purchase Address
0  04/19/19 08:46          917 1st St, Dallas, TX 75001
1             NaN                                   NaN
2  04/07/19 22:30     682 Chestnut St, Boston, MA 02215
3  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
4  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
```

[6]:
```
#Let's check the number of rows to confirm if the data has been appended on the␣
 ↪df.
all_mnths.shape
```

[6]:
```
(186850, 6)
```

[8]:

```
#Now, we'll save it into a csv file
all_mnths.to_csv('yearly_sales', index = False)
```

[9]:

```
#Loading the dataframe from the system
df_ = pd.read_csv('yearly_sales')
df_.head()
```

[9]:

```
  Order ID                     Product Quantity Ordered Price Each  \
0   176558        USB-C Charging Cable                2      11.95
1      NaN                         NaN              NaN        NaN
2   176559  Bose SoundSport Headphones                1      99.99
3   176560                Google Phone                1        600
4   176560            Wired Headphones                1      11.99

       Order Date                      Purchase Address
0  04/19/19 08:46          917 1st St, Dallas, TX 75001
1             NaN                                   NaN
2  04/07/19 22:30     682 Chestnut St, Boston, MA 02215
3  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
4  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
```

[10]:

```
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
```

[11]:

```
#Lets confirm the rows that have NaN values
df_nan = df_[df_.isna().any(axis=1)]
df_nan
#From the output, we can see that we have 545 blank rows and we have to remove␣
 ↪them.
```

[11]:

```
       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
```

```
1553        NaN      NaN                NaN         NaN         NaN
…         …     …              …        …        …
185176      NaN      NaN                NaN         NaN         NaN
185438      NaN      NaN                NaN         NaN         NaN
186042      NaN      NaN                NaN         NaN         NaN
186548      NaN      NaN                NaN         NaN         NaN
186826      NaN      NaN                NaN         NaN         NaN

       Purchase Address
1                   NaN
356                 NaN
735                 NaN
1433                NaN
1553                NaN
…                 …
185176              NaN
185438              NaN
186042              NaN
186548              NaN
186826              NaN

[545 rows x 6 columns]
```

[12]:

```
df_ = df_.dropna(how ='all')
#This line of code drops all rows where all the columns are NaN values.
df_.isna().sum()
```

[12]:

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

[ ]:

```
#Before we begin our analysis, we will add more columns to give more details to␣
 ↪our data.
```

1. Which month had the best sales and how much was earned?

[13]:

```
df_['Month'] = df_['Order Date'].str[:2]
df_['Month'] = df_['Month'].astype('int32')
df_.head()
#This is not running due to some rows on the date column containing 'Or' as the␣
 ↪first 2 characters. let's check for them
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

```
Input In [13], in <cell line: 2>()
      1 df_['Month'] = df_['Order Date'].str[:2]
----> 2 df_['Month'] = df_['Month'].astype('int32')
      3 df_.head()

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:5912, in NDFrame.
 ↪astype(self, dtype, copy, errors)
   5905     results = [
   5906         self.iloc[:, i].astype(dtype, copy=copy)
   5907         for i in range(len(self.columns))
   5908     ]
   5910 else:
   5911     # else, only a single dtype is given
-> 5912     new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   5913     return self._constructor(new_data).__finalize__(self,␣
 ↪method="astype")
   5915 # GH 33113: handle empty frame or series

File ~\anaconda3\lib\site-packages\pandas\core\internals\managers.py:419, in␣
 ↪BaseBlockManager.astype(self, dtype, copy, errors)
    418 def astype(self: T, dtype, copy: bool = False, errors: str = "raise") ->␣
 ↪T:
--> 419     return self.apply("astype", dtype=dtype, copy=copy, errors=errors)

File ~\anaconda3\lib\site-packages\pandas\core\internals\managers.py:304, in␣
 ↪BaseBlockManager.apply(self, f, align_keys, ignore_failures, **kwargs)
    302         applied = b.apply(f, **kwargs)
    303     else:
--> 304         applied = getattr(b, f)(**kwargs)
    305 except (TypeError, NotImplementedError):
    306     if not ignore_failures:

File ~\anaconda3\lib\site-packages\pandas\core\internals\blocks.py:580, in Block.
 ↪astype(self, dtype, copy, errors)
    562 """
    563 Coerce to the new dtype.
    564
   (…)
    576 Block
    577 """
    578 values = self.values
--> 580 new_values = astype_array_safe(values, dtype, copy=copy, errors=errors)
    582 new_values = maybe_coerce_values(new_values)
    583 newb = self.make_block(new_values)

File ~\anaconda3\lib\site-packages\pandas\core\dtypes\cast.py:1292, in␣
 ↪astype_array_safe(values, dtype, copy, errors)
   1289     dtype = dtype.numpy_dtype
   1291 try:
-> 1292     new_values = astype_array(values, dtype, copy=copy)
   1293 except (ValueError, TypeError):
   1294     # e.g. astype_nansafe can fail on object-dtype of strings
   1295     #  trying to convert to float
   1296     if errors == "ignore":

File ~\anaconda3\lib\site-packages\pandas\core\dtypes\cast.py:1237, in␣
 ↪astype_array(values, dtype, copy)
   1234     values = values.astype(dtype, copy=copy)
   1236 else:
-> 1237     values = astype_nansafe(values, dtype, copy=copy)
   1239 # in pandas we don't store numpy str dtypes, so convert to object
   1240 if isinstance(dtype, np.dtype) and issubclass(values.dtype.type, str):

File ~\anaconda3\lib\site-packages\pandas\core\dtypes\cast.py:1154, in␣
 ↪astype_nansafe(arr, dtype, copy, skipna)
   1150 elif is_object_dtype(arr.dtype):
   1151
   1152     # work around NumPy brokenness, #1987
   1153     if np.issubdtype(dtype.type, np.integer):
-> 1154         return lib.astype_intsafe(arr, dtype)
   1156     # if we have a datetime/timedelta array of objects
   1157     # then coerce to a proper dtype and recall astype_nansafe
   1159     elif is_datetime64_dtype(dtype):

File ~\anaconda3\lib\site-packages\pandas\_libs\lib.pyx:668, in pandas._libs.li
 ↪astype_intsafe()

ValueError: invalid literal for int() with base 10: 'Or'
```

[14]:

```
or_df =df_[df_['Order Date'].str[:2] == 'Or']
or_df
```

[14]:

```
        Order ID  Product  Quantity Ordered  Price Each  Order Date  \
519     Order ID  Product  Quantity Ordered  Price Each  Order Date
1149    Order ID  Product  Quantity Ordered  Price Each  Order Date
1155    Order ID  Product  Quantity Ordered  Price Each  Order Date
2878    Order ID  Product  Quantity Ordered  Price Each  Order Date
2893    Order ID  Product  Quantity Ordered  Price Each  Order Date
…            …        …                 …           …           …
185164  Order ID  Product  Quantity Ordered  Price Each  Order Date
185551  Order ID  Product  Quantity Ordered  Price Each  Order Date
186563  Order ID  Product  Quantity Ordered  Price Each  Order Date
186632  Order ID  Product  Quantity Ordered  Price Each  Order Date
186738  Order ID  Product  Quantity Ordered  Price Each  Order Date
```

```
        Purchase Address Month
519     Purchase Address    Or
1149    Purchase Address    Or
1155    Purchase Address    Or
2878    Purchase Address    Or
2893    Purchase Address    Or
…                      …   …
185164  Purchase Address    Or
185551  Purchase Address    Or
186563  Purchase Address    Or
186632  Purchase Address    Or
186738  Purchase Address    Or

[355 rows x 7 columns]
```

[15]:
```
#we'll drop those rows with 'Or' in the date column. waste of data...
df_ =df_[df_['Order Date'].str[:2] != 'Or']
df_
```

[15]:
```
       Order ID                     Product Quantity Ordered Price Each  \
0        176558        USB-C Charging Cable                2      11.95
2        176559  Bose SoundSport Headphones                1      99.99
3        176560                Google Phone                1        600
4        176560            Wired Headphones                1      11.99
5        176561            Wired Headphones                1      11.99
…           …                          …               …         …
186845   259353      AAA Batteries (4-pack)                3       2.99
186846   259354                      iPhone                1        700
186847   259355                      iPhone                1        700
186848   259356      34in Ultrawide Monitor                1     379.99
186849   259357        USB-C Charging Cable                1      11.95

            Order Date                        Purchase Address Month
0       04/19/19 08:46            917 1st St, Dallas, TX 75001    04
2       04/07/19 22:30       682 Chestnut St, Boston, MA 02215    04
3       04/12/19 14:38    669 Spruce St, Los Angeles, CA 90001    04
4       04/12/19 14:38    669 Spruce St, Los Angeles, CA 90001    04
5       04/30/19 09:27       333 8th St, Los Angeles, CA 90001    04
…                  …                                     …   …
186845  09/17/19 20:56  840 Highland St, Los Angeles, CA 90001    09
186846  09/01/19 16:00  216 Dogwood St, San Francisco, CA 94016    09
186847  09/23/19 07:39     220 12th St, San Francisco, CA 94016    09
186848  09/19/19 17:30   511 Forest St, San Francisco, CA 94016    09
186849  09/30/19 00:18   250 Meadow St, San Francisco, CA 94016    09

[185950 rows x 7 columns]
```

[16]:
```
#Some codes are being duplicated to emphasize the steps in the data cleaning␣
 ↪process.
df_['Month'] = df_['Order Date'].str[:2]
df_['Month'] = df_['Month'].astype('int32')
df_.head()
```

[16]:
```
  Order ID                     Product Quantity Ordered Price Each  \
0   176558        USB-C Charging Cable                2      11.95
2   176559  Bose SoundSport Headphones                1      99.99
3   176560                Google Phone                1        600
4   176560            Wired Headphones                1      11.99
5   176561            Wired Headphones                1      11.99

       Order Date                      Purchase Address  Month
0  04/19/19 08:46          917 1st St, Dallas, TX 75001      4
2  04/07/19 22:30     682 Chestnut St, Boston, MA 02215      4
3  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001      4
4  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001      4
5  04/30/19 09:27     333 8th St, Los Angeles, CA 90001      4
```

[17]:
```
#Let's convert all columns to the appropriate types.
df_['Quantity Ordered'] = pd.to_numeric(df_['Quantity Ordered'])
df_['Price Each'] = df_['Price Each'].astype(float)
```

[18]:
```
df_['Sales'] = df_['Quantity Ordered'] * df_['Price Each']
df_
```

[18]:
```
       Order ID                     Product  Quantity Ordered  Price Each  \
0        176558        USB-C Charging Cable                 2       11.95
2        176559  Bose SoundSport Headphones                 1       99.99
3        176560                Google Phone                 1      600.00
4        176560            Wired Headphones                 1       11.99
5        176561            Wired Headphones                 1       11.99
…         …                         …               …         …
186845   259353      AAA Batteries (4-pack)                 3        2.99
186846   259354                      iPhone                 1      700.00
186847   259355                      iPhone                 1      700.00
186848   259356      34in Ultrawide Monitor                 1      379.99
186849   259357        USB-C Charging Cable                 1       11.95

            Order Date                  Purchase Address  Month   Sales
0       04/19/19 08:46          917 1st St, Dallas, TX 75001      4   23.90
2       04/07/19 22:30     682 Chestnut St, Boston, MA 02215      4   99.99
3       04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001      4  600.00
4       04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001      4   11.99
5       04/30/19 09:27     333 8th St, Los Angeles, CA 90001      4   11.99
…                …                                      …    …     …
```

```
186845  09/17/19 20:56   840 Highland St, Los Angeles, CA 90001      9    8.97
186846  09/01/19 16:00  216 Dogwood St, San Francisco, CA 94016      9  700.00
186847  09/23/19 07:39     220 12th St, San Francisco, CA 94016      9  700.00
186848  09/19/19 17:30   511 Forest St, San Francisco, CA 94016      9  379.99
186849  09/30/19 00:18   250 Meadow St, San Francisco, CA 94016      9   11.95

[185950 rows x 8 columns]
```

[19]:

```
#Let's reorder the columns
df_ = df_[['Order ID', 'Product', 'Quantity Ordered', 'Price Each', 'Sales',␣
 ↪'Order Date', 'Month',
       'Purchase Address' ]]
df_
```

[19]:

```
       Order ID                     Product  Quantity Ordered  Price Each  \
0        176558        USB-C Charging Cable                 2       11.95
2        176559  Bose SoundSport Headphones                 1       99.99
3        176560                Google Phone                 1      600.00
4        176560            Wired Headphones                 1       11.99
5        176561            Wired Headphones                 1       11.99
…           …                         …                …          …
186845   259353      AAA Batteries (4-pack)                 3        2.99
186846   259354                      iPhone                 1      700.00
186847   259355                      iPhone                 1      700.00
186848   259356      34in Ultrawide Monitor                 1      379.99
186849   259357        USB-C Charging Cable                 1       11.95

         Sales      Order Date  Month                       Purchase Address
0        23.90  04/19/19 08:46      4           917 1st St, Dallas, TX 75001
2        99.99  04/07/19 22:30      4      682 Chestnut St, Boston, MA 02215
3       600.00  04/12/19 14:38      4   669 Spruce St, Los Angeles, CA 90001
4        11.99  04/12/19 14:38      4   669 Spruce St, Los Angeles, CA 90001
5        11.99  04/30/19 09:27      4      333 8th St, Los Angeles, CA 90001
…         …             …    …                                    …
186845    8.97  09/17/19 20:56      9    840 Highland St, Los Angeles, CA 90001
186846  700.00  09/01/19 16:00      9  216 Dogwood St, San Francisco, CA 94016
186847  700.00  09/23/19 07:39      9     220 12th St, San Francisco, CA 94016
186848  379.99  09/19/19 17:30      9   511 Forest St, San Francisco, CA 94016
186849   11.95  09/30/19 00:18      9   250 Meadow St, San Francisco, CA 94016

[185950 rows x 8 columns]
```

[20]:

```
#BAck to the Question - MOnth with the highest sales and amount earned.
results = df_.groupby('Month').sum()
results
```

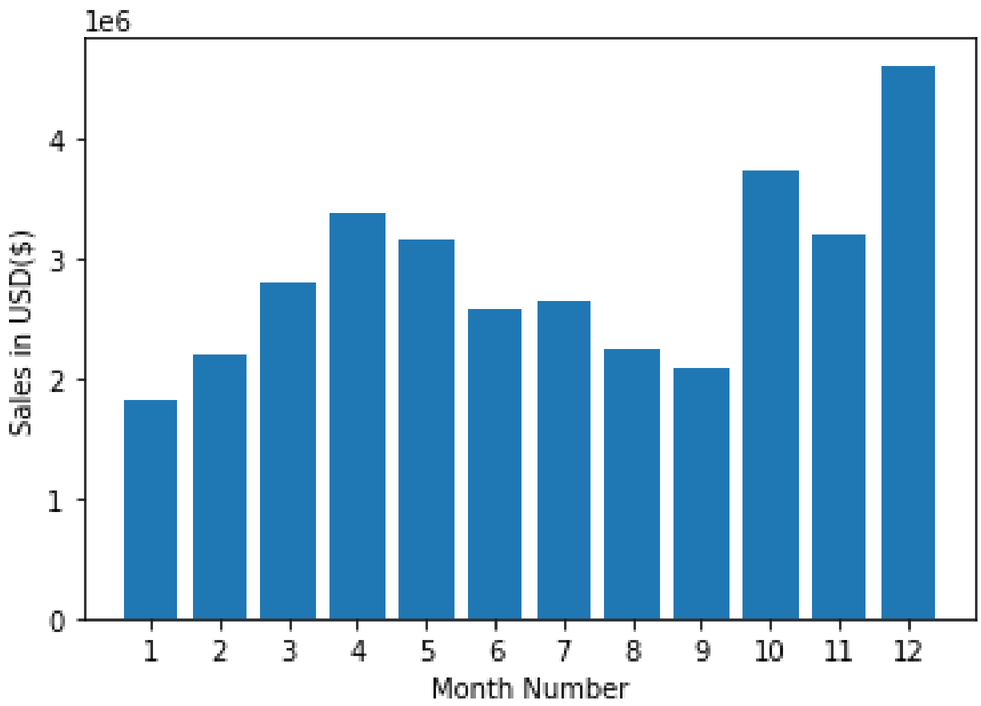
[20]:

```
       Quantity Ordered  Price Each       Sales
Month
1                 10903  1811768.38  1822256.73
2                 13449  2188884.72  2202022.42
3                 17005  2791207.83  2807100.38
4                 20558  3367671.02  3390670.24
5                 18667  3135125.13  3152606.75
6                 15253  2562025.61  2577802.26
7                 16072  2632539.56  2647775.76
8                 13448  2230345.42  2244467.88
9                 13109  2084992.09  2097560.13
10                22703  3715554.83  3736726.88
11                19798  3180600.68  3199603.20
12                28114  4588415.41  4613443.34
```

[21]:

```python
import matplotlib.pyplot as plt

months = range(1,13)
plt.bar(months, results['Sales'])
plt.xticks(months)
plt.ylabel('Sales in USD($)')
plt.xlabel('Month Number')
plt.show()
```



2. What city had the highest number of sales?

```
[24]: #We're going to change the data type of the purchase address column to string␣
       ↪so that we can apply .split method on it.
      df_['Purchase Address'] = df_['Purchase Address'].astype(str)
```

```
[25]: #Now we need to add a city column

      #We'll use the .apply function since it allows us to use any function on our␣
       ↪DataFrame.
      def get_city(address):
          return address.split(',')[1]

      def get_state(address):
          return address.split(',')[2].split(' ')[1]
      #The double splits helps us to get the state where the city is located to␣
       ↪prevent confusion.

      df_['City'] = df_['Purchase Address'].apply(lambda x: get_city(x) + ' ' +␣
       ↪get_state(x) )
      #We could also use an f string to reformat the code
      # df_['City'] = df_['Purchase Address'].apply(lambda x: f"{get_city(x)}␣
       ↪({get_state(x)})")
      df_
```

```
[25]:        Order ID                     Product  Quantity Ordered  Price Each  \
      0        176558        USB-C Charging Cable                 2       11.95
      2        176559  Bose SoundSport Headphones                 1       99.99
      3        176560                Google Phone                 1      600.00
      4        176560            Wired Headphones                 1       11.99
      5        176561            Wired Headphones                 1       11.99
      …           …                         …                …          …
      186845   259353      AAA Batteries (4-pack)                 3        2.99
      186846   259354                      iPhone                 1      700.00
      186847   259355                      iPhone                 1      700.00
      186848   259356      34in Ultrawide Monitor                 1      379.99
      186849   259357        USB-C Charging Cable                 1       11.95

               Sales      Order Date  Month  \
      0        23.90  04/19/19 08:46      4
      2        99.99  04/07/19 22:30      4
      3       600.00  04/12/19 14:38      4
      4        11.99  04/12/19 14:38      4
      5        11.99  04/30/19 09:27      4
      …          …             …     …
      186845    8.97  09/17/19 20:56      9
      186846  700.00  09/01/19 16:00      9
      186847  700.00  09/23/19 07:39      9
```

```
186848  379.99  09/19/19 17:30      9
186849   11.95  09/30/19 00:18      9

                               Purchase Address              City
0                  917 1st St, Dallas, TX 75001         Dallas TX
2             682 Chestnut St, Boston, MA 02215         Boston MA
3          669 Spruce St, Los Angeles, CA 90001    Los Angeles CA
4          669 Spruce St, Los Angeles, CA 90001    Los Angeles CA
5             333 8th St, Los Angeles, CA 90001    Los Angeles CA
…                                             …                 …
186845   840 Highland St, Los Angeles, CA 90001    Los Angeles CA
186846  216 Dogwood St, San Francisco, CA 94016  San Francisco CA
186847     220 12th St, San Francisco, CA 94016  San Francisco CA
186848   511 Forest St, San Francisco, CA 94016  San Francisco CA
186849   250 Meadow St, San Francisco, CA 94016  San Francisco CA

[185950 rows x 9 columns]
```

[39]:

```
city_results = df_.groupby('City').sum()
city_results
```

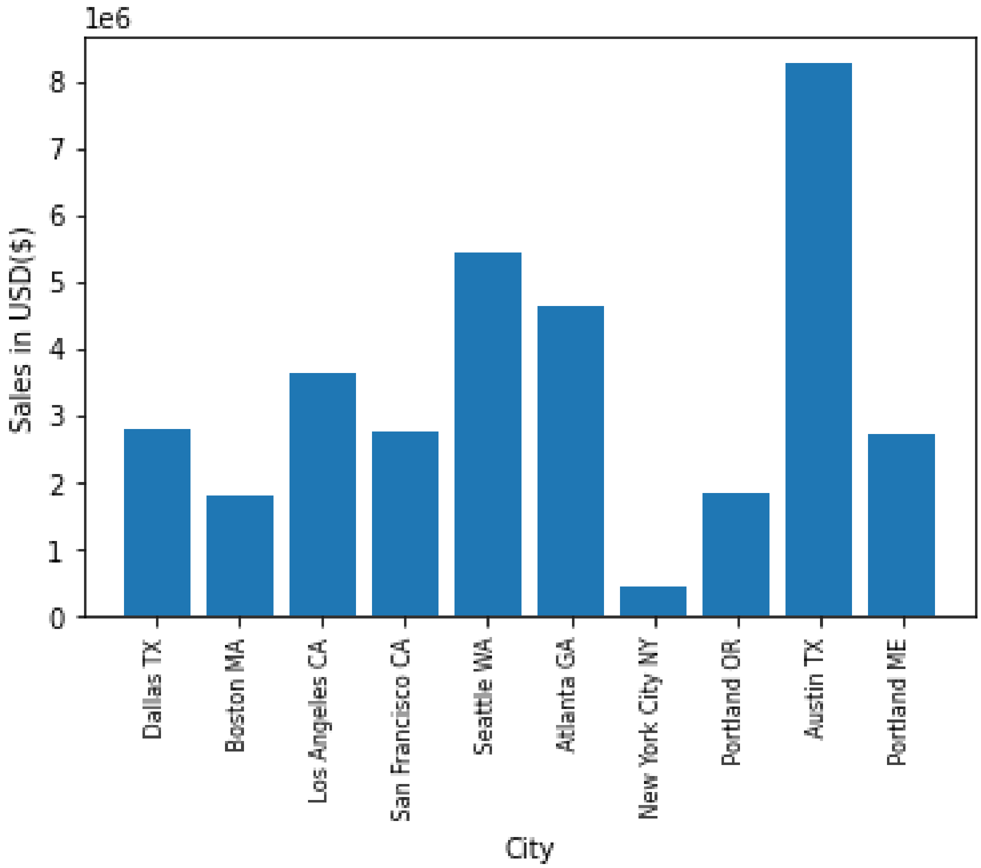
[39]:

```
                   Quantity Ordered  Price Each       Sales   Month
City
 Atlanta GA                   16602  2779908.20  2795498.58  104794
 Austin TX                    11153  1809873.61  1819581.75   69829
 Boston MA                    22528  3637409.77  3661642.01  141112
 Dallas TX                    16730  2752627.82  2767975.40  104620
 Los Angeles CA               33289  5421435.23  5452570.80  208325
 New York City NY             27932  4635370.83  4664317.43  175741
 Portland ME                   2750   447189.25   449758.27   17144
 Portland OR                  11303  1860558.22  1870732.34   70621
 San Francisco CA             50239  8211461.74  8262203.91  315520
 Seattle WA                   16553  2733296.01  2747755.48  104941
```

[40]:

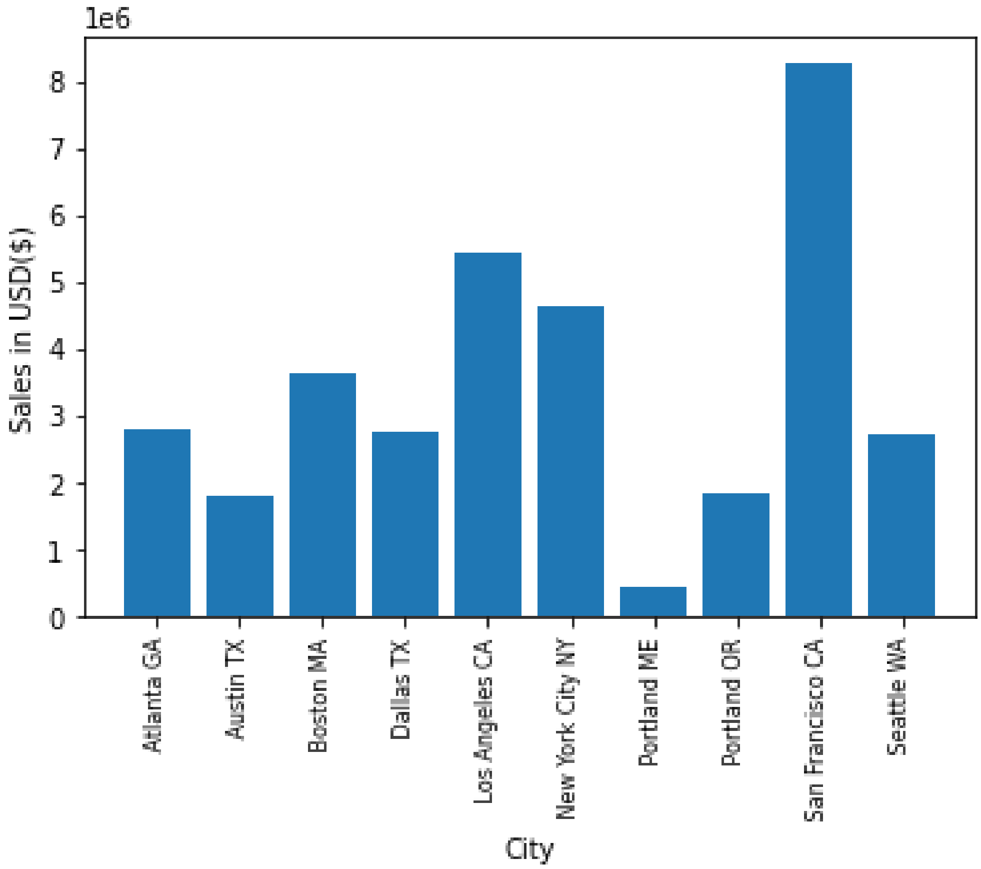
```
import matplotlib.pyplot as plt

cities = df_['City'].unique()
plt.bar(cities, city_results['Sales'])
plt.xticks(cities, rotation= 'vertical', size = 8)
plt.ylabel('Sales in USD($)')
plt.xlabel('City')
plt.show()
#From the plot below, we can see that Austin TX has the highest sales which is␣
 ↪wrong based on the dataframe we have.
#This is because when we use .unique(), the order in the bar chart becomes␣
 ↪distorted.
```



```
[41]: #We'll use list comprehension to achieve cohesion between city and sales figure.
      import matplotlib.pyplot as plt

      cities = [city for city, df in df_.groupby('City')]
      plt.bar(cities, city_results['Sales'])
      plt.xticks(cities, rotation= 'vertical', size = 8)
      plt.ylabel('Sales in USD($)')
      plt.xlabel('City')
      plt.show()
```



3. What time should we display advertisement to improve the likelyhood of a customer buying a product?

```
[42]: #To do this we have to convert the order date type to date time.
      df_['Order Date'] = pd.to_datetime(df_['Order Date'])
```

```
[43]: #Let's create columns for Hour, minute and count
      df_['Hour'] = df_['Order Date'].dt.hour
      df_['Minute'] = df_['Order Date'].dt.minute
      df_['Count'] = 1
```

```
[44]: df_.head()
```

```
[44]:   Order ID                     Product  Quantity Ordered  Price Each   Sales  \
      0   176558        USB-C Charging Cable                 2       11.95   23.90
      2   176559  Bose SoundSport Headphones                 1       99.99   99.99
      3   176560                Google Phone                 1      600.00  600.00
      4   176560            Wired Headphones                 1       11.99   11.99
      5   176561            Wired Headphones                 1       11.99   11.99
```

```
           Order Date  Month                    Purchase Address  \
0 2019-04-19 08:46:00      4          917 1st St, Dallas, TX 75001
2 2019-04-07 22:30:00      4     682 Chestnut St, Boston, MA 02215
3 2019-04-12 14:38:00      4  669 Spruce St, Los Angeles, CA 90001
4 2019-04-12 14:38:00      4  669 Spruce St, Los Angeles, CA 90001
5 2019-04-30 09:27:00      4     333 8th St, Los Angeles, CA 90001

              City  Hour  Minute  Count
0        Dallas TX     8      46      1
2        Boston MA    22      30      1
3   Los Angeles CA    14      38      1
4   Los Angeles CA    14      38      1
5   Los Angeles CA     9      27      1
```

[45]:
```
df_.groupby(['Hour']).count()
```

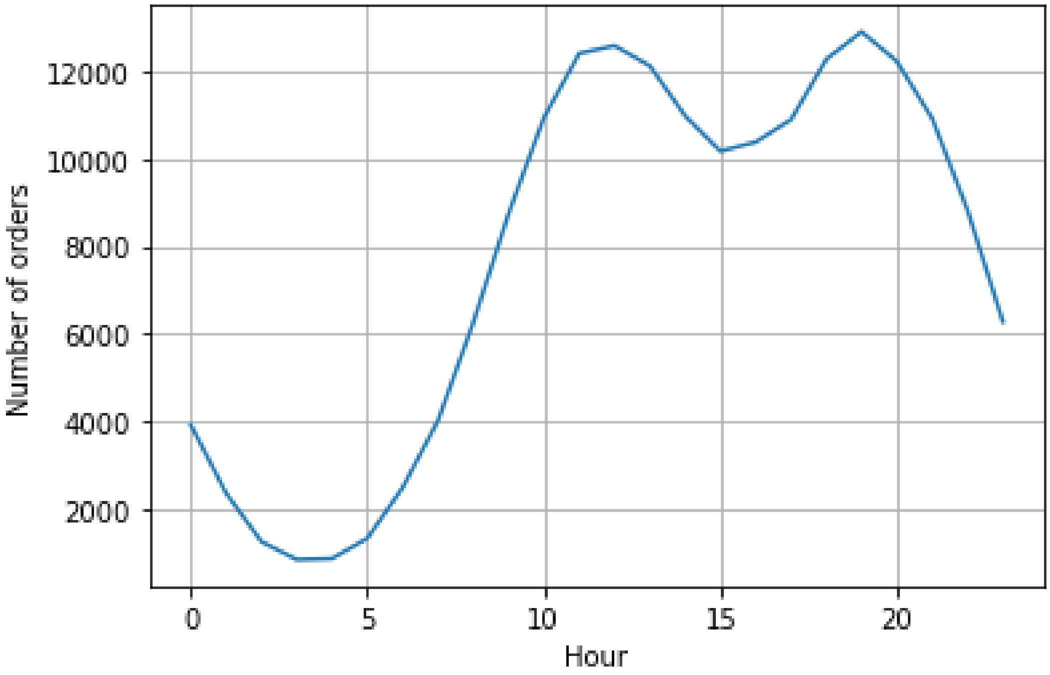
[45]:
```
      Order ID  Product  Quantity Ordered  Price Each  Sales  Order Date  \
Hour
0         3910     3910              3910        3910   3910        3910
1         2350     2350              2350        2350   2350        2350
2         1243     1243              1243        1243   1243        1243
3          831      831               831         831    831         831
4          854      854               854         854    854         854
5         1321     1321              1321        1321   1321        1321
6         2482     2482              2482        2482   2482        2482
7         4011     4011              4011        4011   4011        4011
8         6256     6256              6256        6256   6256        6256
9         8748     8748              8748        8748   8748        8748
10       10944    10944             10944       10944  10944       10944
11       12411    12411             12411       12411  12411       12411
12       12587    12587             12587       12587  12587       12587
13       12129    12129             12129       12129  12129       12129
14       10984    10984             10984       10984  10984       10984
15       10175    10175             10175       10175  10175       10175
16       10384    10384             10384       10384  10384       10384
17       10899    10899             10899       10899  10899       10899
18       12280    12280             12280       12280  12280       12280
19       12905    12905             12905       12905  12905       12905
20       12228    12228             12228       12228  12228       12228
21       10921    10921             10921       10921  10921       10921
22        8822     8822              8822        8822   8822        8822
23        6275     6275              6275        6275   6275        6275

      Month  Purchase Address   City  Minute  Count
Hour
0       3910              3910  3910    3910   3910
1       2350              2350  2350    2350   2350
```

```
2      1243                  1243   1243    1243   1243
3       831                   831    831     831    831
4       854                   854    854     854    854
5      1321                  1321   1321    1321   1321
6      2482                  2482   2482    2482   2482
7      4011                  4011   4011    4011   4011
8      6256                  6256   6256    6256   6256
9      8748                  8748   8748    8748   8748
10    10944                 10944  10944   10944  10944
11    12411                 12411  12411   12411  12411
12    12587                 12587  12587   12587  12587
13    12129                 12129  12129   12129  12129
14    10984                 10984  10984   10984  10984
15    10175                 10175  10175   10175  10175
16    10384                 10384  10384   10384  10384
17    10899                 10899  10899   10899  10899
18    12280                 12280  12280   12280  12280
19    12905                 12905  12905   12905  12905
20    12228                 12228  12228   12228  12228
21    10921                 10921  10921   10921  10921
22     8822                  8822   8822    8822   8822
23     6275                  6275   6275    6275   6275
```

[46]:

```
hours = df_.groupby(['Hour']).count()
y = hours['Count']
y
plt.plot(y)
plt.grid()
plt.xlabel('Hour')
plt.ylabel('Number of orders')
plt.show()
```



[ ]:
```
# hours = [hour for hour, df in df_.groupby('Hour')]
# plt.plot(hours, df_.groupby(['Hour']).count())
# plt.xticks(hours)
# plt.grid()
# plt.xlabel('Hour')
# plt.ylabel('Number of orders')
# plt.show
```

[ ]:
```
#From the chart above, we can see that the most orders came in at the 19th hour␣
 ↪followed by the 12th hour.
#Therefore, the advertisement can be placed on those hours.
#We can as well check the time when most orders were placed per city so that we␣
 ↪can time the adverts appropriately in each city.
```

4. What 2 products were ordered together?

[47]:
```
#To approach this question, we'll look out for order IDs that are same and have␣
 ↪the same order address.
df_.head()
```

[47]:
```
  Order ID                     Product  Quantity Ordered  Price Each   Sales  \
0   176558        USB-C Charging Cable                 2       11.95   23.90
2   176559  Bose SoundSport Headphones                 1       99.99   99.99
3   176560                Google Phone                 1      600.00  600.00
4   176560            Wired Headphones                 1       11.99   11.99
```

```
5   176561            Wired Headphones                 1       11.99   11.99

           Order Date  Month                     Purchase Address  \
0 2019-04-19 08:46:00      4          917 1st St, Dallas, TX 75001
2 2019-04-07 22:30:00      4     682 Chestnut St, Boston, MA 02215
3 2019-04-12 14:38:00      4  669 Spruce St, Los Angeles, CA 90001
4 2019-04-12 14:38:00      4  669 Spruce St, Los Angeles, CA 90001
5 2019-04-30 09:27:00      4     333 8th St, Los Angeles, CA 90001

              City  Hour  Minute  Count
0        Dallas TX     8      46      1
2        Boston MA    22      30      1
3   Los Angeles CA    14      38      1
4   Los Angeles CA    14      38      1
5   Los Angeles CA     9      27      1
```

[48]:

```
dup = df_[df_['Order ID'].duplicated(keep = False)]

dup.head(30)
```

[48]:

```
    Order ID                     Product  Quantity Ordered  Price Each  \
3     176560                Google Phone                 1      600.00
4     176560            Wired Headphones                 1       11.99
18    176574                Google Phone                 1      600.00
19    176574        USB-C Charging Cable                 1       11.95
30    176585  Bose SoundSport Headphones                 1       99.99
31    176585  Bose SoundSport Headphones                 1       99.99
32    176586      AAA Batteries (4-pack)                 2        2.99
33    176586                Google Phone                 1      600.00
119   176672    Lightning Charging Cable                 1       14.95
120   176672        USB-C Charging Cable                 1       11.95
129   176681    Apple Airpods Headphones                 1      150.00
130   176681             ThinkPad Laptop                 1      999.99
138   176689  Bose SoundSport Headphones                 1       99.99
139   176689      AAA Batteries (4-pack)                 2        2.99
189   176739      34in Ultrawide Monitor                 1      379.99
190   176739                Google Phone                 1      600.00
225   176774    Lightning Charging Cable                 1       14.95
226   176774        USB-C Charging Cable                 1       11.95
233   176781                      iPhone                 1      700.00
234   176781    Lightning Charging Cable                 1       14.95
250   176797                Google Phone                 1      600.00
251   176797  Bose SoundSport Headphones                 1       99.99
252   176797            Wired Headphones                 1       11.99
260   176805                Google Phone                 1      600.00
261   176805        USB-C Charging Cable                 1       11.95
264   176808                Google Phone                 1      600.00
```

```
265   176808            Wired Headphones                 1        11.99
270   176813                Google Phone                 1       600.00
271   176813            Wired Headphones                 1        11.99
394   176935      AAA Batteries (4-pack)                 1         2.99

      Sales          Order Date  Month  \
3    600.00 2019-04-12 14:38:00      4
4     11.99 2019-04-12 14:38:00      4
18   600.00 2019-04-03 19:42:00      4
19    11.95 2019-04-03 19:42:00      4
30    99.99 2019-04-07 11:31:00      4
31    99.99 2019-04-07 11:31:00      4
32     5.98 2019-04-10 17:00:00      4
33   600.00 2019-04-10 17:00:00      4
119   14.95 2019-04-12 11:07:00      4
120   11.95 2019-04-12 11:07:00      4
129  150.00 2019-04-20 10:39:00      4
130  999.99 2019-04-20 10:39:00      4
138   99.99 2019-04-24 17:15:00      4
139    5.98 2019-04-24 17:15:00      4
189  379.99 2019-04-05 17:38:00      4
190  600.00 2019-04-05 17:38:00      4
225   14.95 2019-04-25 15:06:00      4
226   11.95 2019-04-25 15:06:00      4
233  700.00 2019-04-03 07:37:00      4
234   14.95 2019-04-03 07:37:00      4
250  600.00 2019-04-21 08:54:00      4
251   99.99 2019-04-21 08:54:00      4
252   11.99 2019-04-21 08:54:00      4
260  600.00 2019-04-01 15:50:00      4
261   11.95 2019-04-01 15:50:00      4
264  600.00 2019-04-28 18:03:00      4
265   11.99 2019-04-28 18:03:00      4
270  600.00 2019-04-28 18:01:00      4
271   11.99 2019-04-28 18:01:00      4
394    2.99 2019-04-03 21:31:00      4

                            Purchase Address               City  Hour  Minute  \
3       669 Spruce St, Los Angeles, CA 90001     Los Angeles CA    14      38
4       669 Spruce St, Los Angeles, CA 90001     Los Angeles CA    14      38
18         20 Hill St, Los Angeles, CA 90001     Los Angeles CA    19      42
19         20 Hill St, Los Angeles, CA 90001     Los Angeles CA    19      42
30         823 Highland St, Boston, MA 02215          Boston MA    11      31
31         823 Highland St, Boston, MA 02215          Boston MA    11      31
32    365 Center St, San Francisco, CA 94016   San Francisco CA    17       0
33    365 Center St, San Francisco, CA 94016   San Francisco CA    17       0
119    778 Maple St, New York City, NY 10001   New York City NY    11       7
```

```
120    778 Maple St, New York City, NY 10001   New York City NY    11       7
129         331 Cherry St, Seattle, WA 98101         Seattle WA    10      39
130         331 Cherry St, Seattle, WA 98101         Seattle WA    10      39
138  659 Lincoln St, New York City, NY 10001   New York City NY    17      15
139  659 Lincoln St, New York City, NY 10001   New York City NY    17      15
189             730 6th St, Austin, TX 73301          Austin TX    17      38
190             730 6th St, Austin, TX 73301          Austin TX    17      38
225     372 Church St, Los Angeles, CA 90001     Los Angeles CA    15       6
226     372 Church St, Los Angeles, CA 90001     Los Angeles CA    15       6
233         976 Hickory St, Dallas, TX 75001          Dallas TX     7      37
234         976 Hickory St, Dallas, TX 75001          Dallas TX     7      37
250        923 Elm St, Los Angeles, CA 90001     Los Angeles CA     8      54
251        923 Elm St, Los Angeles, CA 90001     Los Angeles CA     8      54
252        923 Elm St, Los Angeles, CA 90001     Los Angeles CA     8      54
260        91 Lincoln St, Portland, OR 97035        Portland OR    15      50
261        91 Lincoln St, Portland, OR 97035        Portland OR    15      50
264   933 Meadow St, San Francisco, CA 94016   San Francisco CA    18       3
265   933 Meadow St, San Francisco, CA 94016   San Francisco CA    18       3
270           269 Hill St, Atlanta, GA 30301         Atlanta GA    18       1
271           269 Hill St, Atlanta, GA 30301         Atlanta GA    18       1
394             315 1st St, Dallas, TX 75001          Dallas TX    21      31

     Count
3        1
4        1
18       1
19       1
30       1
31       1
32       1
33       1
119      1
120      1
129      1
130      1
138      1
139      1
189      1
190      1
225      1
226      1
233      1
234      1
250      1
251      1
252      1
260      1
```

```
261      1
264      1
265      1
270      1
271      1
394      1
```

[49]:
```
#We'll group the products based on the order ID and join the products found on␣
 ↪the duplicate order IDs together

dup['Grouped'] = dup.groupby('Order ID')['Product'].transform(lambda x: ', '.
 ↪join(x))
dup.head()
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_23008\2868453735.py:3:
SettingWithCopyWarning:
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/user_guide/indexing.html#returning-a-view-versus-a-copy
  dup['Grouped'] = dup.groupby('Order ID')['Product'].transform(lambda x: ',
'.join(x))
```

[49]:
```
   Order ID                     Product  Quantity Ordered  Price Each   Sales  \
3    176560                Google Phone                 1      600.00  600.00
4    176560            Wired Headphones                 1       11.99   11.99
18   176574                Google Phone                 1      600.00  600.00
19   176574        USB-C Charging Cable                 1       11.95   11.95
30   176585  Bose SoundSport Headphones                 1       99.99   99.99

            Order Date  Month                       Purchase Address  \
3  2019-04-12 14:38:00      4  669 Spruce St, Los Angeles, CA 90001
4  2019-04-12 14:38:00      4  669 Spruce St, Los Angeles, CA 90001
18 2019-04-03 19:42:00      4     20 Hill St, Los Angeles, CA 90001
19 2019-04-03 19:42:00      4     20 Hill St, Los Angeles, CA 90001
30 2019-04-07 11:31:00      4     823 Highland St, Boston, MA 02215

               City  Hour  Minute  Count  \
3    Los Angeles CA    14      38      1
4    Los Angeles CA    14      38      1
18   Los Angeles CA    19      42      1
19   Los Angeles CA    19      42      1
30        Boston MA    11      31      1

                                              Grouped
3                       Google Phone, Wired Headphones
```

```
4                        Google Phone, Wired Headphones
18                   Google Phone, USB-C Charging Cable
19                   Google Phone, USB-C Charging Cable
30   Bose SoundSport Headphones, Bose SoundSport He…
```

[50]:

```
#drop duplicates
dup = dup[['Order ID', 'Grouped']].drop_duplicates()
dup.head(100)
```

[50]:

```
     Order ID                                           Grouped
3      176560                    Google Phone, Wired Headphones
18     176574                Google Phone, USB-C Charging Cable
30     176585  Bose SoundSport Headphones, Bose SoundSport He…
32     176586              AAA Batteries (4-pack), Google Phone
119    176672     Lightning Charging Cable, USB-C Charging Cable
…         …                                                 …
2662   179108   Lightning Charging Cable, AAA Batteries (4-pack)
2683   179128                    iPhone, Apple Airpods Headphones
2718   179162                Google Phone, USB-C Charging Cable
2783   179226         34in Ultrawide Monitor, Macbook Pro Laptop
2829   179270                    iPhone, Lightning Charging Cable

[100 rows x 2 columns]
```

[51]:

```
from itertools import combinations
from collections import Counter
count = Counter()
for row in dup['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
('Vareebadd Phone', ' Wired Headphones') 143
(' USB-C Charging Cable', ' Wired Headphones') 120
('Vareebadd Phone', ' Bose SoundSport Headphones') 80
```

5. What product sold the most and why?

[ ]:

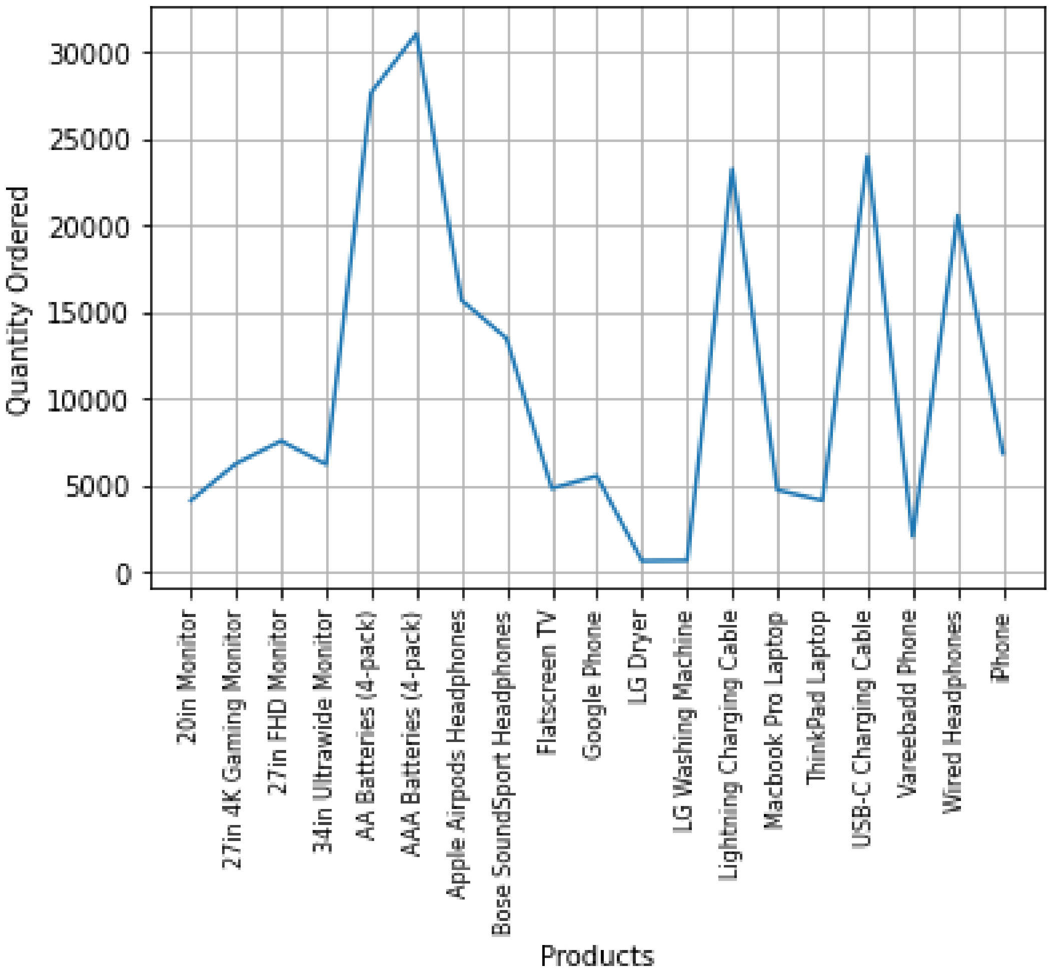
```
#To approach this, we use the sum of the Quantity ordered based on the products
```

[65]:

```
#We can plot a line graph

prod_ = df_.groupby('Product').sum()
prod_
quantity = prod_['Quantity Ordered']
quantity
plt.plot(quantity)
plt.grid()
plt.xlabel('Products')
plt.ylabel('Quantity Ordered')
plt.xticks(products, rotation = 'vertical', size=8)
plt.show()

#products = [product for product, df in prod_]
```

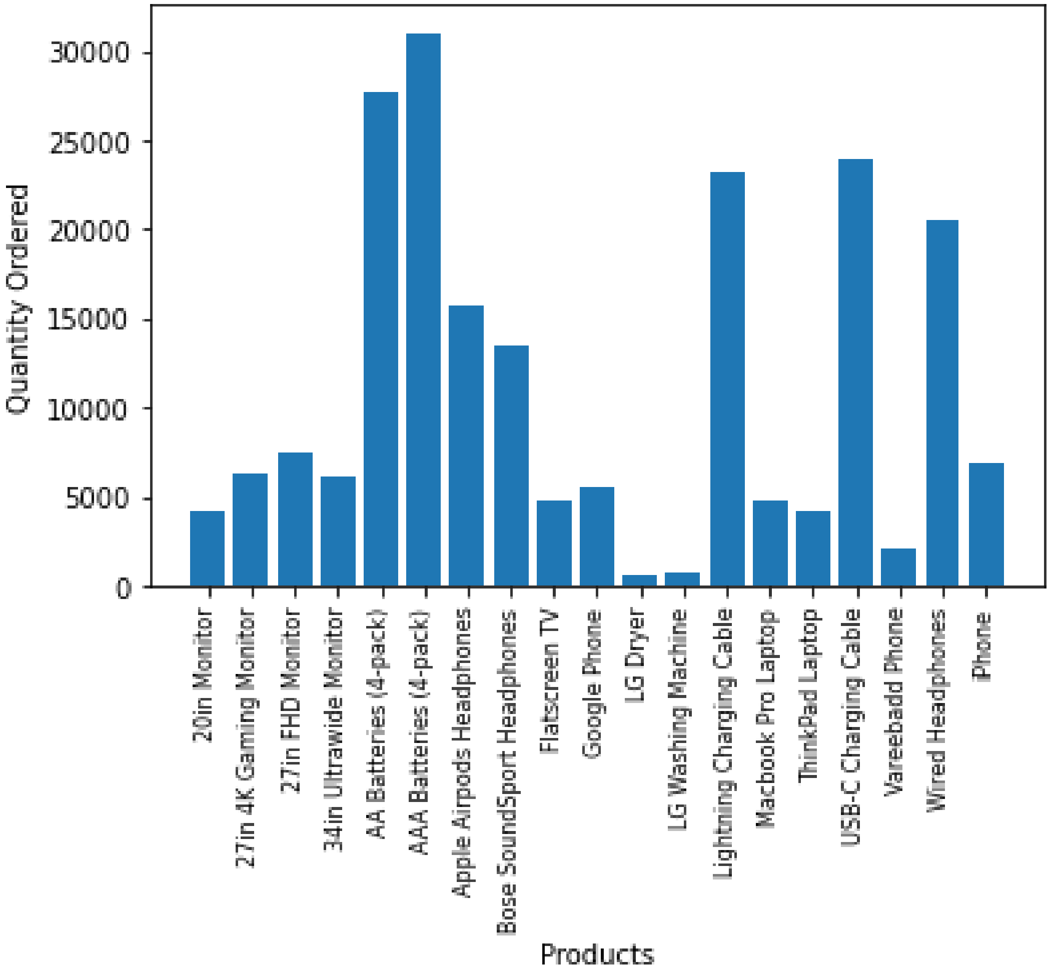


[67]:

```
#We can as well plot a bar chart

prod_ = df_.groupby('Product')
quantity = prod_.sum()['Quantity Ordered']

products = [product for product, df in prod_]
plt.bar(products, quantity)
plt.xticks(products, rotation = 'vertical', size=8)
plt.xlabel('Products')
plt.ylabel('Quantity Ordered')
plt.show()
#From the plot below, we can see that the AAA Batteries(4-pack) sold the most.
```



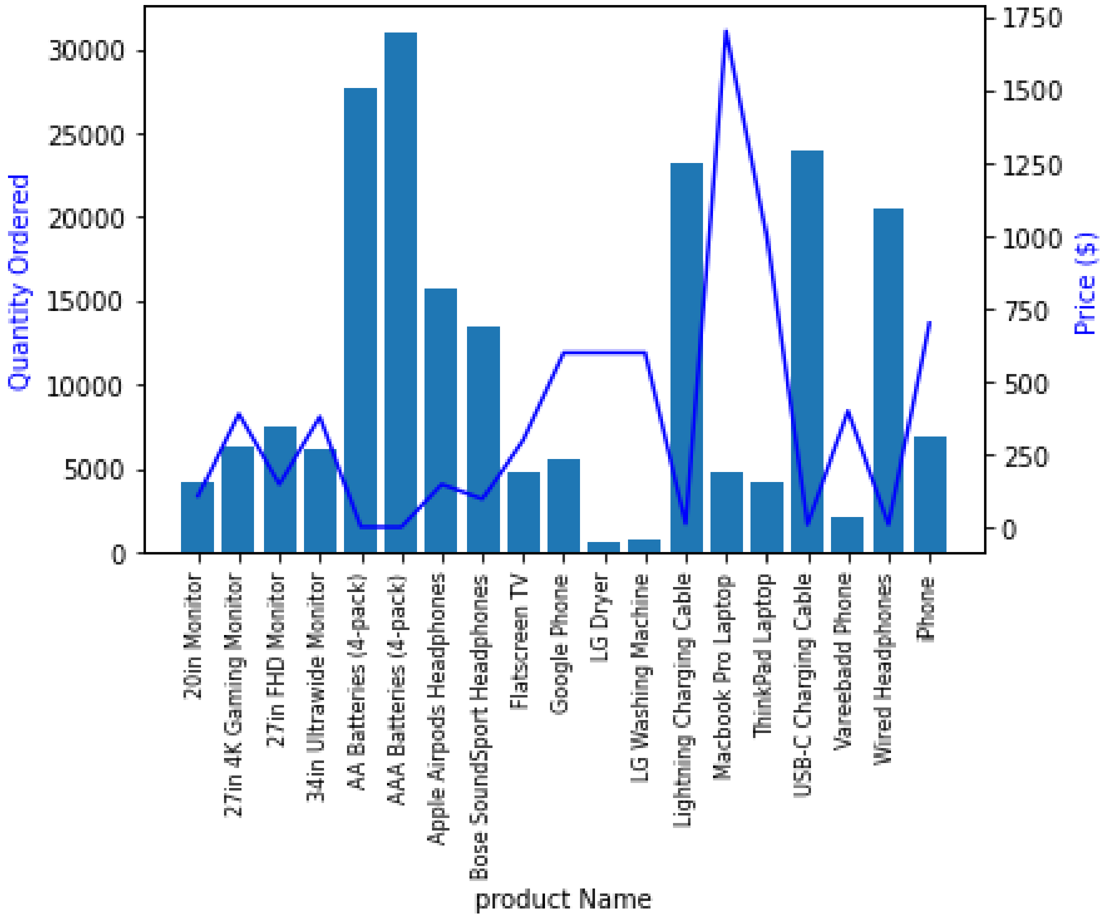
[110]:

```
#To see why it sold the most, we check for correlation between the price and␣
  ↪the products.
prices = df_.groupby('Product').mean()['Price Each']
```

```
fig, ax1 = plt.subplots()
ax1.bar(products, quantity)
ax2 = ax1.twinx()
ax2.plot(products, prices, 'b-')

ax1.set_xlabel('product Name')
ax1.set_ylabel('Quantity Ordered', color = 'b')
ax2.set_ylabel('Price ($)', color ='b')
ax1.set_xticklabels(products, rotation = 'vertical', size =8)
plt.show()
#From the plot, we can see that the most ordered item was one of the cheapest␣
  ↪which is the reason is was ordered the most.
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_23008\17279656.py:11: UserWarning:
FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(products, rotation = 'vertical', size =8)
```



```
[196]:
```

[ ]: