In [1]:

```
import pandas as pd
import os
```

In [2]:

```
from openpyxl import load_workbook
```

Select Data Source

In [3]:

```
files= os.listdir('./SalesAnalysis/Sales_Data')
```

Remove Unwanted Files

In [4]:

```
files=files[1:]
```

Merge Data from all files

In [5]:

```
all_data=pd.DataFrame()
for file in files:
    data=pd.read_csv('./SalesAnalysis/Sales_Data/'+file)
    all_data=pd.concat([all_data,data])
```

In [6]:

```
all_data.head()
```

Out[6]:

	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

Remove rows containing headings

In [7]:

```
all_data=all_data.loc[all_data['Quantity Ordered']!='Quantity Ordered']
```

Question 1: What month has maximum Sales?

Extract Day and Month from Datetime column

In [8]:

```
all_data['Order Day']=pd.DatetimeIndex(all_data['Order Date']).day

all_data['Order Month']=pd.DatetimeIndex(all_data['Order Date']).month

all_data.dropna(how='all',inplace=True)
```

Create a Calculated Column of Sales

In [9]:

```
all_data['Quantity Ordered']=pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each']=pd.to_numeric(all_data['Price Each'])
all_data['Sales']=all_data['Quantity Ordered']*all_data['Price Each']
```

Best Sales Month

In [10]:

```
bsm=all_data.groupby('Order Month').sum()
bsm
```

Out[10]:

	Quantity Ordered	Price Each	Order Day	Sales
Order Month				
1.0	10903	1.811768e+06	155814.0	1.822257e+06
2.0	13449	2.188885e+06	174408.0	2.202022e+06
3.0	17005	2.791208e+06	241774.0	2.807100e+06
4.0	20558	3.367671e+06	282960.0	3.390670e+06
5.0	18667	3.135125e+06	264875.0	3.152607e+06
6.0	15253	2.562026e+06	209880.0	2.577802e+06
7.0	16072	2.632540e+06	227910.0	2.647776e+06
8.0	13448	2.230345e+06	192315.0	2.244468e+06
9.0	13109	2.084992e+06	180101.0	2.097560e+06
10.0	22703	3.715555e+06	326141.0	3.736727e+06
11.0	19798	3.180601e+06	272854.0	3.199603e+06
12.0	28114	4.588415e+06	401453.0	4.613443e+06

In [11]:

```
bsm['Sales'].plot.bar()
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xee9610>
```

Question 2: What City has maximum Sales?

Extract City from Address

Lets use apply function

In [12]:

```
all_data['City']=all_data['Purchase Address'].apply(lambda x: x.split(',')[1])
```

In [13]:

```
all_data['LPin']=all_data['Purchase Address'].apply(lambda x: x.split(',')[2]).str[-5:]
```

In [14]:

```
all_data.head()
```

Out[14]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Order Day	Order Month	Sales	City	
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	19.0	4.0	23.90	Dallas	7
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	7.0	4.0	99.99	Boston	(
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	600.00	Los Angeles	9
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	11.99	Los Angeles	9
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	30.0	4.0	11.99	Los Angeles	9

In [15]:

```
all_data['Location']=all_data[['City','LPin']].apply(lambda x: " ".join(x),axis=1)
```

In [16]:

```
result=all_data.groupby('Location').sum().sort_values('Sales',ascending=False)
```

In [17]:

```
result
```

Out[17]:

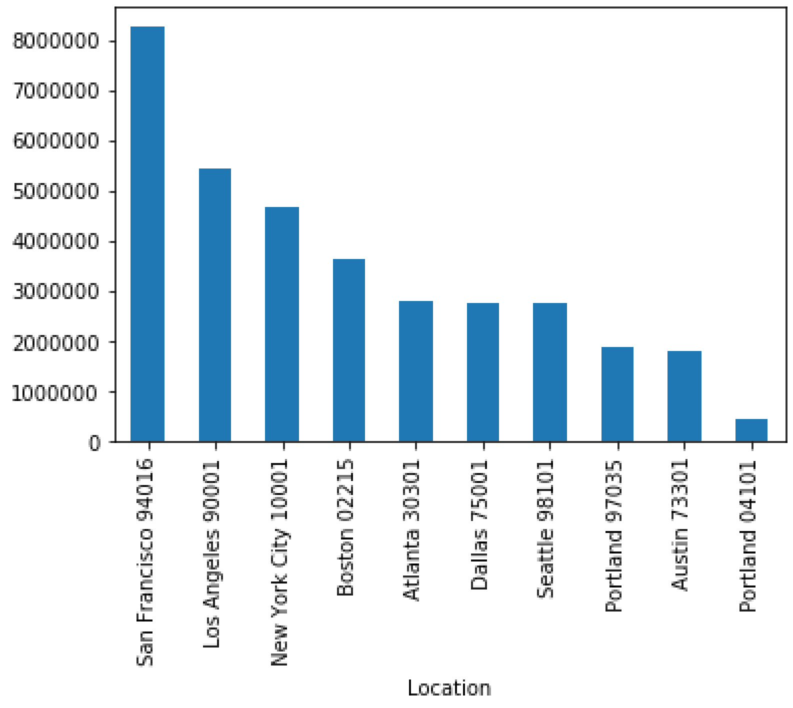
	Quantity Ordered	Price Each	Order Day	Order Month	Sales
Location					
San Francisco 94016	50239	8.211462e+06	702669.0	315520.0	8.262204e+06
Los Angeles 90001	33289	5.421435e+06	469607.0	208325.0	5.452571e+06
New York City 10001	27932	4.635371e+06	392706.0	175741.0	4.664317e+06
Boston 02215	22528	3.637410e+06	312376.0	141112.0	3.661642e+06
Atlanta 30301	16602	2.779908e+06	234837.0	104794.0	2.795499e+06
Dallas 75001	16730	2.752628e+06	234435.0	104620.0	2.767975e+06
Seattle 98101	16553	2.733296e+06	229552.0	104941.0	2.747755e+06
Portland 97035	11303	1.860558e+06	159233.0	70621.0	1.870732e+06
Austin 73301	11153	1.809874e+06	156782.0	69829.0	1.819582e+06
Portland 04101	2750	4.471893e+05	38288.0	17144.0	4.497583e+05

In [18]:

```
result['Sales'].plot.bar()
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xdcf8c30>
```



Question 3: What time advertisements should display

to maximize sales?

In [19]:

```
all_data.head()
```

Out[19]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Order Day	Order Month	Sales	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	19.0	4.0	23.90	Dallas
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	7.0	4.0	99.99	Boston
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	600.00	Los Angeles
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	11.99	Los Angeles
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	30.0	4.0	11.99	Los Angeles

In [20]:

```
all_data['Order Date']=pd.to_datetime(all_data['Order Date'])
```

In [21]:

```
all_data['Order Time']=all_data['Order Date'].apply(lambda x: x.time().hour)
```

In [22]:

```
adtime=all_data.groupby('Order Time').sum()
adtime
```

Out[22]:

	Quantity Ordered	Price Each	Order Day	Order Month	Sales
Order Time					
0	4428	709296.70	62015.0	27554.0	713721.27
1	2619	458490.00	37333.0	16657.0	460866.88
2	1398	233833.64	19631.0	8507.0	234851.44
3	928	144726.42	13430.0	5904.0	145757.89
4	937	162058.18	13756.0	6148.0	162661.01
5	1493	229621.21	21347.0	9301.0	230679.82
6	2810	445000.11	39824.0	17539.0	448113.00
7	4556	740568.11	63111.0	28850.0	744854.12
8	7002	1185970.62	98078.0	43626.0	1192348.97
9	9816	1628498.49	137512.0	60981.0	1639030.58
10	12308	1932665.62	173128.0	76928.0	1944286.77
11	14005	2288855.18	195901.0	87654.0	2300610.24
12	14202	2299876.68	197231.0	89161.0	2316821.34
13	13685	2139743.86	190415.0	85808.0	2155389.80
14	12362	2072194.77	173264.0	77836.0	2083672.73
15	11391	1931174.99	161441.0	72060.0	1941549.60
16	11662	1892454.54	162579.0	72939.0	1904601.31
17	12229	2116777.02	169311.0	77454.0	2129361.61
18	13802	2207696.93	194087.0	86421.0	2219348.30
19	14470	2398588.31	205868.0	91389.0	2412938.54
20	13768	2268185.16	192147.0	86375.0	2281716.24
21	12244	2030763.83	171214.0	77103.0	2042000.86
22	9899	1599464.44	138981.0	62088.0	1607549.21
23	7065	1172625.87	98881.0	44364.0	1179304.44

In [23]:

```
z=all_data['Order Time'].unique()
z
```

Out[23]:

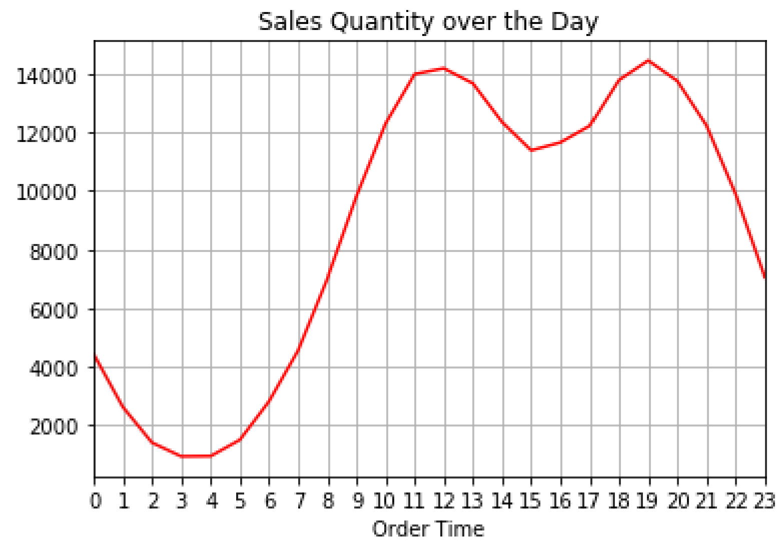
```
array([ 8, 22, 14,  9, 13,  7, 10, 17, 12, 19, 15, 20, 18,  0, 11, 23, 21,
        4, 16,  5,  2,  1,  6,  3], dtype=int64)
```

In [24]:

```
adtime['Quantity Ordered'].plot(xticks=z,color='r',title='Sales Quantity over the Day',gric
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xb2de690>
```

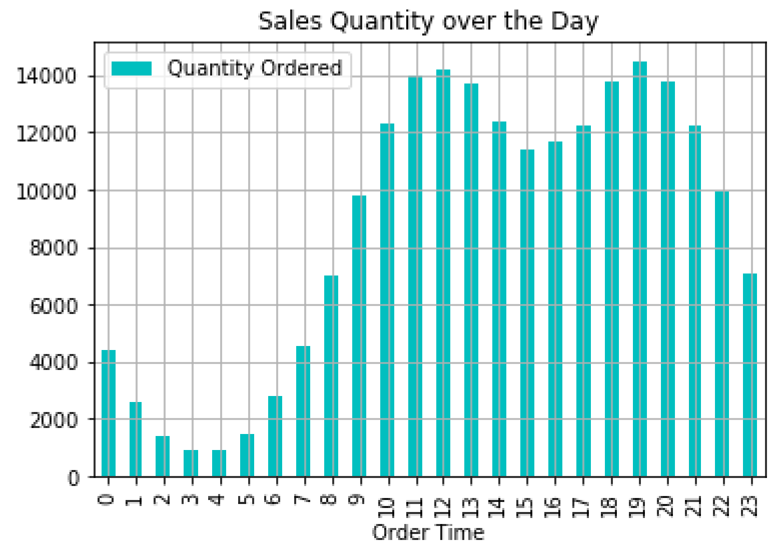


In [25]:

```
adtime['Quantity Ordered'].plot.bar(xticks=z,color='c',title='Sales Quantity over the Day',
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xdcc6610>
```



Question 4: Which are the most sold together items?

In [26]:

```
all_data.head()
```

Out[26]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Order Day	Order Month	Sales	City
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	19.0	4.0	23.90	Dallas
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	7.0	4.0	99.99	Boston
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	600.00	Los Angeles
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	11.99	Los Angeles
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	30.0	4.0	11.99	Los Angeles

In [27]:

```
dup=all_data[all_data.duplicated(subset=['Order ID'],keep=False)]
```

In [28]:

```
all_data.head()
```

Out[28]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Order Day	Order Month	Sales	City
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	19.0	4.0	23.90	Dallas
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	7.0	4.0	99.99	Boston
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	600.00	Los Angeles
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	11.99	Los Angeles
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	30.0	4.0	11.99	Los Angeles

In [29]:

```
dup.head(3)
```

Out[29]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Order Day	Order Month	Sales	City
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	600.00	Los Angeles
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	12.0	4.0	11.99	Los Angeles
18	176574	Google Phone	1	600.00	2019-04-03 19:42:00	20 Hill St, Los Angeles, CA 90001	3.0	4.0	600.00	Los Angeles

In [30]:

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

```
c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\i
pykernel_launcher.py:1: 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: http://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (http://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy)
  """Entry point for launching an IPython kernel.
```

In [31]:

```
dup[['Grouped','Order ID']].drop_duplicates()
```

Out[31]:

	Grouped	Order ID
3	Google Phone,Wired Headphones	176560
18	Google Phone,USB-C Charging Cable	176574
30	Bose SoundSport Headphones,Bose SoundSport Hea...	176585
32	AAA Batteries (4-pack),Google Phone	176586
119	Lightning Charging Cable,USB-C Charging Cable	176672
...	...	...
11617	Apple Airpods Headphones,Apple Airpods Headphones	259296
11619	iPhone,Lightning Charging Cable,Lightning Char...	259297
11627	34in Ultrawide Monitor,AA Batteries (4-pack)	259303
11639	Wired Headphones,AAA Batteries (4-pack)	259314
11677	Google Phone,USB-C Charging Cable	259350

7136 rows × 2 columns

In [32]:

```
from itertools import combinations
from collections import Counter
```

In [33]:

```
count=Counter()
for items in dup['Grouped']:
    items_list = items.split(',')
    count.update(Counter(combinations(items_list,2)))


print(count)
```

```
Counter({('iPhone', 'Lightning Charging Cable'): 2140, ('Google Phone', 'U
SB-C Charging Cable'): 2116, ('iPhone', 'Wired Headphones'): 987, ('Google
Phone', 'Wired Headphones'): 949, ('iPhone', 'Apple Airpods Headphones'):
799, ('Vareebadd Phone', 'USB-C Charging Cable'): 773, ('Google Phone', 'B
ose SoundSport Headphones'): 503, ('USB-C Charging Cable', 'Wired Headphon
es'): 452, ('Vareebadd Phone', 'Wired Headphones'): 327, ('Lightning Charg
ing Cable', 'Wired Headphones'): 253, ('Lightning Charging Cable', 'Apple
Airpods Headphones'): 214, ('USB-C Charging Cable', 'Bose SoundSport Headp
hones'): 211, ('Vareebadd Phone', 'Bose SoundSport Headphones'): 182, ('Ap
ple Airpods Headphones', 'Wired Headphones'): 170, ('Bose SoundSport Headp
hones', 'Wired Headphones'): 140, ('Lightning Charging Cable', 'USB-C Char
ging Cable'): 120, ('Lightning Charging Cable', 'AA Batteries (4-pack)'):
114, ('Lightning Charging Cable', 'Lightning Charging Cable'): 111, ('AA B
atteries (4-pack)', 'Lightning Charging Cable'): 102, ('AAA Batteries (4-p
ack)', 'USB-C Charging Cable'): 100, ('Apple Airpods Headphones', 'AAA Bat
teries (4-pack)'): 99, ('USB-C Charging Cable', 'USB-C Charging Cable'): 9
9, ('AA Batteries (4-pack)', 'AAA Batteries (4-pack)'): 96, ('AAA Batterie
s (4-pack)', 'AAA Batteries (4-pack)'): 96, ('Wired Headphones', 'USB-C Ch
arging Cable'): 95, ('USB-C Charging Cable', 'AAA Batteries (4-pack)'): 9
```

In [34]:

```
for x,y in count.most_common():
    print(x,'-----',y)
    print('_________________________________________')
```

```
('iPhone', 'Lightning Charging Cable') ----- 2140
_________________________________________
('Google Phone', 'USB-C Charging Cable') ----- 2116
_________________________________________
('iPhone', 'Wired Headphones') ----- 987
_________________________________________
('Google Phone', 'Wired Headphones') ----- 949
_________________________________________
('iPhone', 'Apple Airpods Headphones') ----- 799
_________________________________________
('Vareebadd Phone', 'USB-C Charging Cable') ----- 773
_________________________________________
('Google Phone', 'Bose SoundSport Headphones') ----- 503
_________________________________________
('USB-C Charging Cable', 'Wired Headphones') ----- 452
_________________________________________
('Vareebadd Phone', 'Wired Headphones') ----- 327
_________________________________________
('Lightning Charging Cable', 'Wired Headphones') ----- 253
```

Question 5: What product sold the most? And Why do you think it sold the most?

In [35]:

```
Psales=all_data[['Product','Quantity Ordered','Price Each']].groupby('Product').sum(ascendi
```

In [36]:

```
Psales=all_data[['Product','Quantity Ordered','Price Each']].groupby('Product').agg({'Quant
```

In [40]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [41]:

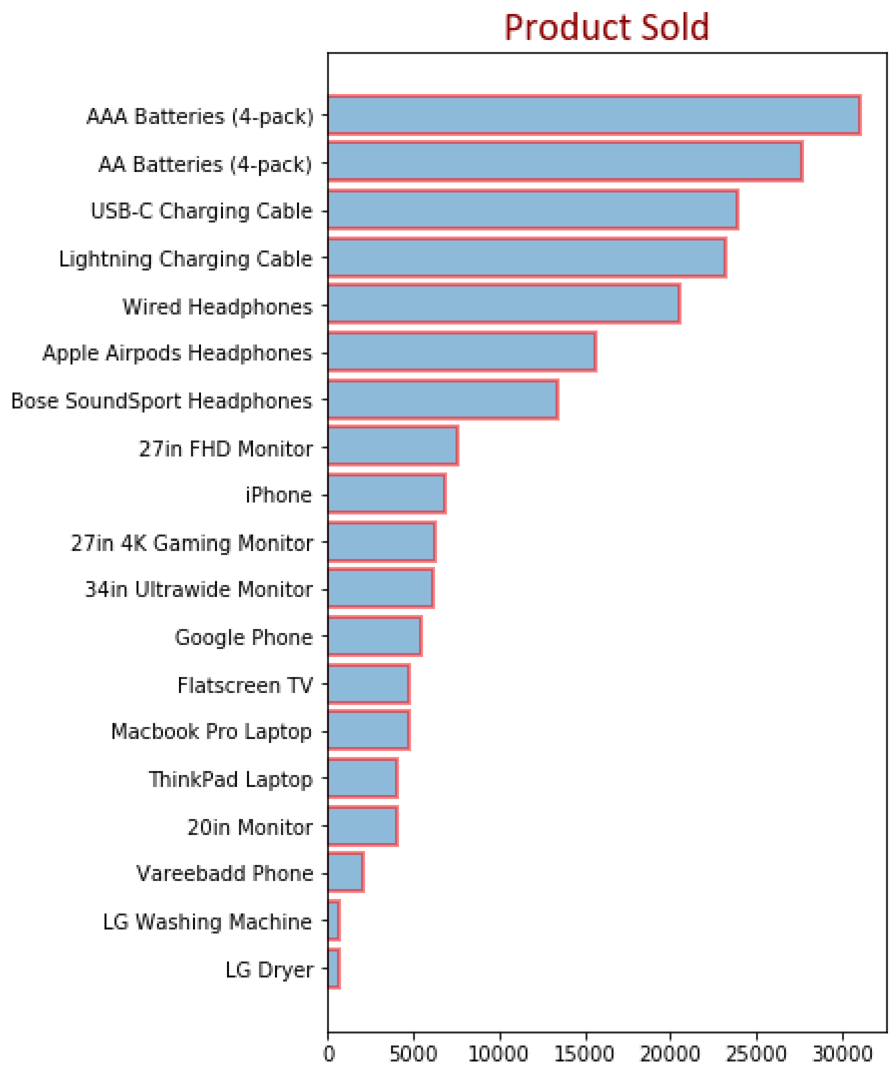
```
x=Psales.index
```

In [55]:

```
y=Psales['Quantity Ordered']
z=Psales['Price Each'].mean()
```

In [89]:

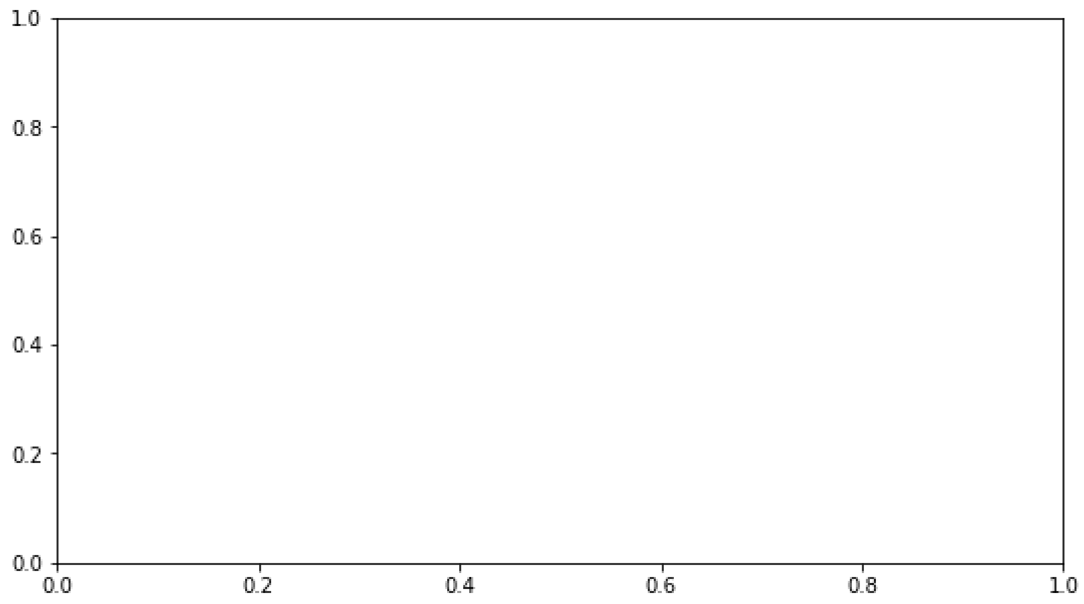
```
plt.figure(figsize=(5,9))
plt.barh(x,y,alpha=0.5,label=Psales.index,edgecolor='r',linewidth=2)
plt.title('Product Sold',fontdict={'family':'calibri','fontsize':20,'color':'darkred'})
plt.show()
```



In [110]:

```
plt.figure(figsize=(9,5))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\m
atplotlib\markers.py in set_marker(self, marker)
    308             try:
--> 309                 Path(marker)
    310                 self._marker_function = self._set_vertices

c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\m
atplotlib\path.py in __init__(self, vertices, codes, _interpolation_steps, c
losed, readonly)
    126         """
--> 127         vertices = _to_unmasked_float_array(vertices)
    128         if vertices.ndim != 2 or vertices.shape[1] != 2:

c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\m
atplotlib\cbook\__init__.py in _to_unmasked_float_array(x)
   1389     else:
-> 1390         return np.asarray(x, float)
   1391 

c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\n
umpy\core\_asarray.py in asarray(a, dtype, order)
     84     """
---> 85     return array(a, dtype, copy=False, order=order)
     86 

ValueError: could not convert string to float: '*'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-110-291d89f88eb0> in <module>
      1 plt.figure(figsize=(9,5))
----> 2 plt.scatter([all_data['Order Day'],all_data['Order Day']],[all_data[
'Quantity Ordered'],all_data['Price Each']],marker=['*','o'])

c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\m
atplotlib\pyplot.py in scatter(x, y, s, c, marker, cmap, norm, vmin, vmax, a
lpha, linewidths, verts, edgecolors, plotnonfinite, data, **kwargs)
   2846         verts=verts, edgecolors=edgecolors,
   2847         plotnonfinite=plotnonfinite, **({"data": data} if data is no
t
-> 2848         None else {}), **kwargs)
   2849     sci(__ret)
   2850     return __ret

c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\m
atplotlib\__init__.py in inner(ax, data, *args, **kwargs)
   1597     def inner(ax, *args, data=None, **kwargs):
   1598         if data is None:
-> 1599             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1600 
   1601         bound = new_sig.bind(ax, *args, **kwargs)

c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\m
atplotlib\axes\_axes.py in scatter(self, x, y, s, c, marker, cmap, norm, vmi
n, vmax, alpha, linewidths, verts, edgecolors, plotnonfinite, **kwargs)
   4478                 marker_obj = marker
   4479             else:
-> 4480                 marker_obj = mmarkers.MarkerStyle(marker)
   4481
   4482             path = marker_obj.get_path().transformed(

c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\m
atplotlib\markers.py in __init__(self, marker, fillstyle)
    241         self._marker_function = None
    242         self.set_fillstyle(fillstyle)
--> 243         self.set_marker(marker)
    244
    245     def _recache(self):

c:\users\bipin\appdata\local\programs\python\python37-32\lib\site-packages\m
atplotlib\markers.py in set_marker(self, marker)
    311             except ValueError:
    312                 raise ValueError('Unrecognized marker style {!r}'
--> 313                                  .format(marker))
    314
    315         self._marker = marker

ValueError: Unrecognized marker style ['*', 'o']
```



In [ ]: