Import Libraries:

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
from itertools import combinations
from collections import Counter
```

In [3]:
```
os.listdir("./desktop/pythonfiles/data_science/sales_analysis/sales_data")
```

Out[3]:
```
['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']
```

Read the dataset:

In [4]:
```
# Read a sample dataset to examine the general structure of the sales datasets
sales = pd.read_csv("./desktop/pythonfiles/data_science/sales_analysis/sales_data/sales_march_2019.csv")
sales.head()
```

Out[4]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	162009	iPhone	1	700	03/28/19 20:59	942 Church St, Austin, TX 73301
1	162009	Lightning Charging Cable	1	14.95	03/28/19 20:59	942 Church St, Austin, TX 73301
2	162009	Wired Headphones	2	11.99	03/28/19 20:59	942 Church St, Austin, TX 73301
3	162010	Bose SoundSport Headphones	1	99.99	03/17/19 05:39	261 10th St, San Francisco, CA 94016
4	162011	34in Ultrawide Monitor	1	379.99	03/10/19 00:01	764 13th St, San Francisco, CA 94016

Join all the months data together:

In [5]:
```
path = "./desktop/pythonfiles/data_science/sales_analysis/sales_data/"
files = [file for file in os.listdir("./desktop/pythonfiles/data_science/sales_analysis/sales_data")]

all_months_data = pd.DataFrame()          #Creates an empty dataframe

for file in files:
    df = pd.read_csv(path + file)
    all_months_data = pd.concat([all_months_data, df])

# all_months_data.to_csv("./desktop/pythonfiles/data_science/sales_analysis/All_months_data.csv",index=False)
```

In [6]:
```
sales = pd.read_csv("./desktop/pythonfiles/data_science/sales_analysis/all_months_data.csv")
sales.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

In [7]:
```
sales.shape
```

Out[7]:
```
(186850, 6)
```

Data cleaning:

In [8]:
```
sales.isna().sum()
```

Out[8]:
```
Order ID            545
Product             545
Quantity Ordered    545
Price Each          545
Order Date          545
Purchase Address    545
dtype: int64
```

Drop all Null values:

In [9]:
```
sales = sales.dropna()
sales.isna().sum().sum()
```

Out[9]:
```
0
```

From Above, all Null values have been dropped

In [10]:
```
sales.sort_values("Order ID")
### Get rid of records with "Order date" as Order Date value:
```

Out[10]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
67547	141234	iPhone	1	700	01/22/19 21:25	944 Walnut St, Boston, MA 02215
67548	141235	Lightning Charging Cable	1	14.95	01/28/19 14:15	185 Maple St, Portland, OR 97035
67549	141236	Wired Headphones	2	11.99	01/17/19 13:33	538 Adams St, San Francisco, CA 94016
67550	141237	27in FHD Monitor	1	149.99	01/05/19 20:33	738 10th St, Los Angeles, CA 90001
67551	141238	Wired Headphones	1	11.99	01/25/19 11:59	387 10th St, Austin, TX 73301
...	...	...	...	...	...	...
47406	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
107991	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
47479	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
30648	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
72448	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address

186305 rows × 6 columns

In [11]:
```
sales = sales[sales["Order Date"] != "Order Date"]
sales
```

Out[11]:

	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
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
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001
...	...	...	...	...	...	...
186845	259353	AAA Batteries (4-pack)	3	2.99	09/17/19 20:56	840 Highland St, Los Angeles, CA 90001
186846	259354	iPhone	1	700	09/01/19 16:00	216 Dogwood St, San Francisco, CA 94016
186847	259355	iPhone	1	700	09/23/19 07:39	220 12th St, San Francisco, CA 94016
186848	259356	34in Ultrawide Monitor	1	379.99	09/19/19 17:30	511 Forest St, San Francisco, CA 94016
186849	259357	USB-C Charging Cable	1	11.95	09/30/19 00:18	250 Meadow St, San Francisco, CA 94016

185950 rows × 6 columns

Correct data Formats:

Change "Order Date" to datetime format

In [12]:
```
sales["Order Date"] = pd.to_datetime(sales["Order Date"])
sales["Order Date"].dtype
```

Out[12]:
```
dtype('<M8[ns]')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 6 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  datetime64[ns]
 5   Purchase Address  185950 non-null  object
dtypes: datetime64[ns](1), object(5)
memory usage: 9.9+ MB
```

Change the remaining important data fields from object to the appropriate numeric format:

In [14]:
```
sales["Quantity Ordered"] = pd.to_numeric(sales["Quantity Ordered"])
sales["Price Each"] = pd.to_numeric(sales["Price Each"])
```

In [15]:
```
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185950 entries, 0 to 186849
Data columns (total 6 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  int64
 3   Price Each        185950 non-null  float64
 4   Order Date        185950 non-null  datetime64[ns]
 5   Purchase Address  185950 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(3)
memory usage: 9.9+ MB
```

Add necessary additional columns:

Add month column:

In [16]:
```
sales["month"] = sales["Order Date"].dt.month
sales.head(2)
```

Out[16]:

	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
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4

Add city column:

In [17]:
```
def get_city(address):
    return address.split(",")[1].strip()

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

In [18]:
```
get_state("917 1st St, Dallas, TX 75001")
```

Out[18]:
```
'TX'
```

In [19]:
```
sales["city"] = sales["Purchase Address"].apply(lambda x: f"{get_city(x)} ({get_state(x)})")
```

In [20]:
```
sales
```

Out[20]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month	city
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	Boston (MA)
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	4	Los Angeles (CA)
...	...	...	...	...	...	...	...	...
186845	259353	AAA Batteries (4-pack)	3	2.99	2019-09-17 20:56:00	840 Highland St, Los Angeles, CA 90001	9	Los Angeles (CA)
186846	259354	iPhone	1	700.00	2019-09-01 16:00:00	216 Dogwood St, San Francisco, CA 94016	9	San Francisco (CA)
186847	259355	iPhone	1	700.00	2019-09-23 07:39:00	220 12th St, San Francisco, CA 94016	9	San Francisco (CA)
186848	259356	34in Ultrawide Monitor	1	379.99	2019-09-19 17:30:00	511 Forest St, San Francisco, CA 94016	9	San Francisco (CA)
186849	259357	USB-C Charging Cable	1	11.95	2019-09-30 00:18:00	250 Meadow St, San Francisco, CA 94016	9	San Francisco (CA)

185950 rows × 8 columns

Question 1: What was the best month for sales? And How much was earned that month?????

Firstly add an additional column "`sales`":

In [21]:
```
sales["sales"] = sales["Quantity Ordered"] * sales["Price Each"]
sales.head()
```

Out[21]:

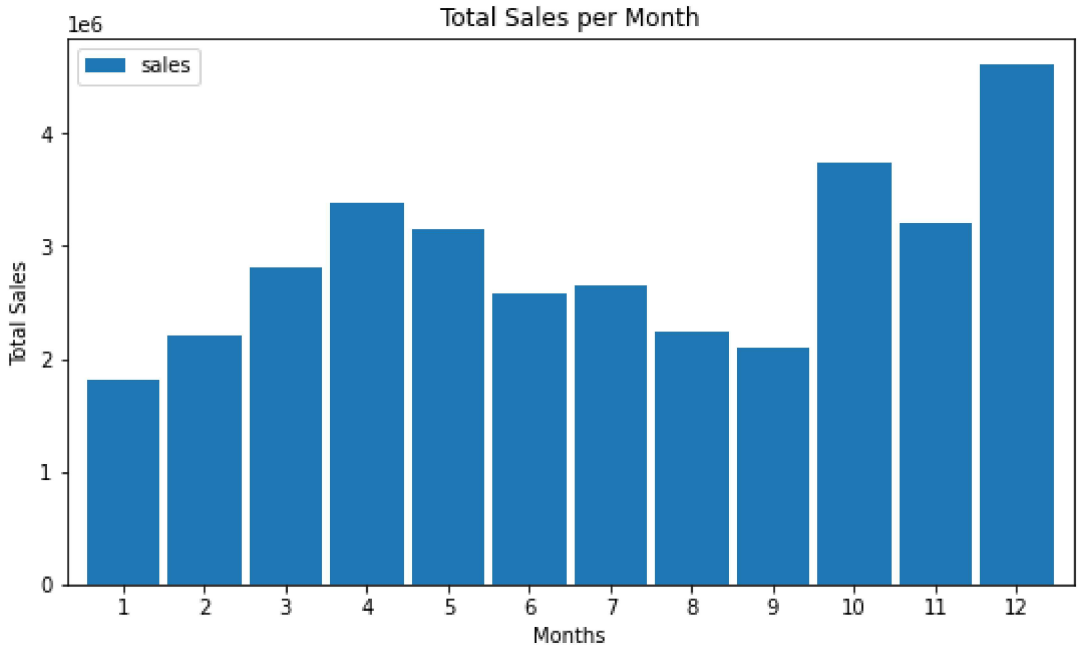
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	month	city	sales
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	Dallas (TX)	23.90
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	Boston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

In [22]:
```
sales_month = sales.groupby("month")["sales"].sum().reset_index()
sales_month
```

Out[22]:

	month	sales
0	1	1822256.73
1	2	2202022.42
2	3	2807100.38
3	4	3390670.24
4	5	3152606.75
5	6	2577802.26
6	7	2647775.76
7	8	2244467.88
8	9	2097560.13
9	10	3736726.88
10	11	3199603.20
11	12	4613443.34

In [23]:
```
# plt.style.use("bmh")

sales_month.plot.bar("month",rot=0,title="Total Sales per Month",ylabel="Total Sales",xlabel="Months",figsize=(9,5),wid
plt.show()
```



solution 1: We can see that *December* has the most sales of well over $4 million

Question2: What city sold the most products???

In [24]:
```
sales.groupby(["city"])["Quantity Ordered"].sum()
```

Out[24]:
```
city
Atlanta (GA)          16602
Austin (TX)           11153
Boston (MA)           22528
Dallas (TX)           16730
Los Angeles (CA)      33289
New York City (NY)    27932
Portland (ME)          2750
Portland (OR)         11303
San Francisco (CA)    50239
Seattle (WA)          16553
Name: Quantity Ordered, dtype: int64
```

solution 2: San Francisco(CA) sold the most Number of Products

Question 3: What time is best we display advertisements to boost likelihood of customers purchasing products:

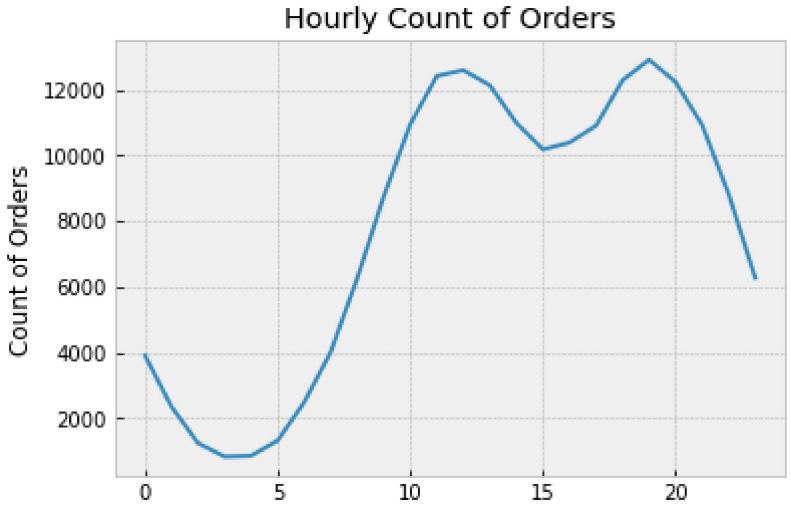
In [25]:
```
# we need to check for the hour of the day that has most purchases/counts:
sales["hour"]= sales["Order Date"].dt.hour
sales["minute"] = sales["Order Date"].dt.minute
```

In [26]:
```
# sales_data = sales.groupby("hour").count().sort_values("Order ID",ascending=False)
# sales_data

sales_data = sales.groupby("hour").count().reset_index()
sales_data = sales_data.loc[:,["hour","Order ID"]]
sales_data = sales_data.rename(columns={"Order ID":"order count"})
sales_data
```

Out[26]:

	hour	order count
0	0	3910
1	1	2350
2	2	1243
3	3	831
4	4	854
5	5	1321
6	6	2482
7	7	4011
8	8	6256
9	9	8748
10	10	10944
11	11	12411
12	12	12587
13	13	12129
14	14	10984
15	15	10175
16	16	10384
17	17	10899
18	18	12280
19	19	12905
20	20	12228
21	21	10921
22	22	8822
23	23	6275

In [27]:
```
plt.style.use("bmh")
sales_data.plot(y = "order count")
plt.ylabel("Count of Orders")
plt.title("Hourly Count of Orders")
plt.gca().get_legend().remove()

plt.show()
```



Solution 3: The best time for Advertisements to boost customer purchases is few minutes to 11am and 7pm

Question 4: What products are most often bought together???

In [28]:
```
sales["Product"].unique()
```

Out[28]:
```
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'],
      dtype=object)
```

In [29]:
```
df = sales[sales["Order ID"].duplicated(keep=False)]

df["Grouped"] = df.groupby("Order ID")["Product"].transform(lambda x: ",".join(x))
df2 = df[["Order ID","Grouped"]].drop_duplicates()
```

```
count = Counter()

for row in df2["Grouped"]:
    row_list = row.split(",")
    count.update(Counter(combinations(row_list, 2)))

for key, value in count.most_common(20):
    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
('Lightning Charging Cable', 'Apple Airpods Headphones') 81
('Vareebadd Phone', 'Bose SoundSport Headphones') 80
('USB-C Charging Cable', 'Bose SoundSport Headphones') 77
('Apple Airpods Headphones', 'Wired Headphones') 69
('Lightning Charging Cable', 'USB-C Charging Cable') 58
('Lightning Charging Cable', 'AA Batteries (4-pack)') 55
('Lightning Charging Cable', 'Lightning Charging Cable') 54
('Bose SoundSport Headphones', 'Wired Headphones') 53
('AA Batteries (4-pack)', 'Lightning Charging Cable') 51
('AAA Batteries (4-pack)', 'USB-C Charging Cable') 50
```

Solution 4: The `iPhone` & `Lightning Charging Cable` are the products most often bought together

Question 5: What product sold the most(What Product was ordered for the most)???

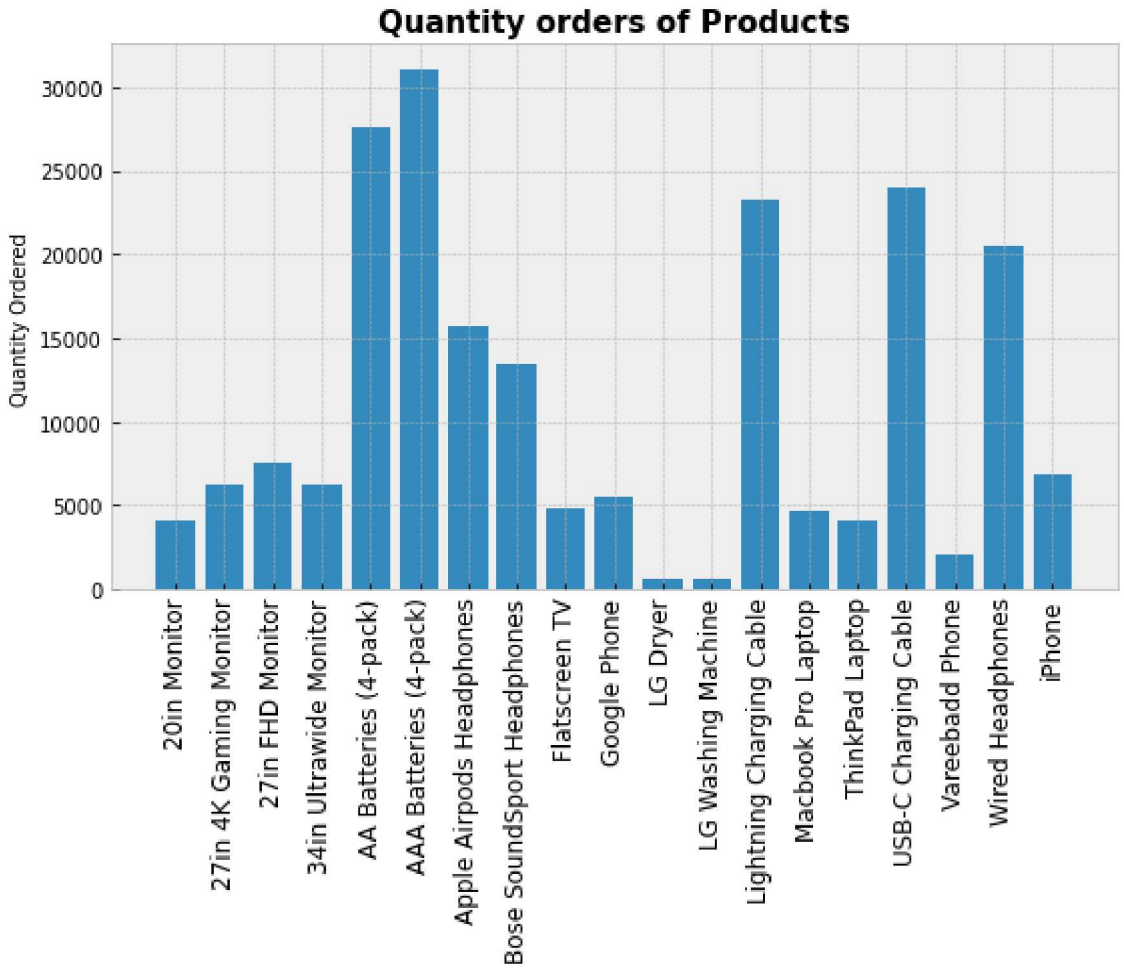
In [30]:
```
sales_count = sales.groupby(["Product"])["Quantity Ordered"].sum().reset_index()
sales_count
```

Out[30]:

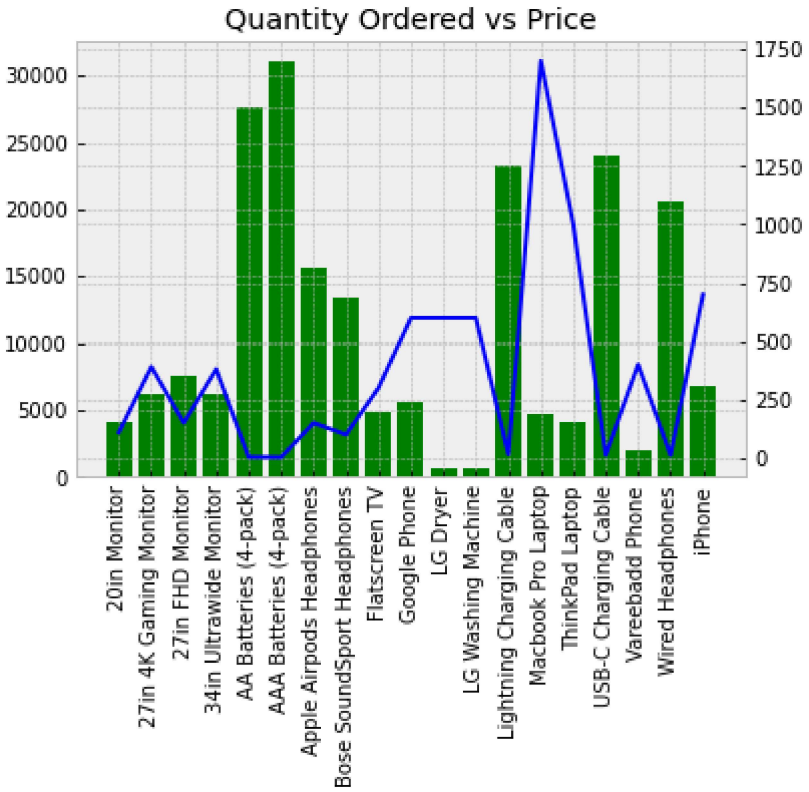
	Product	Quantity Ordered
0	20in Monitor	4129
1	27in 4K Gaming Monitor	6244
2	27in FHD Monitor	7550
3	34in Ultrawide Monitor	6199
4	AA Batteries (4-pack)	27635
5	AAA Batteries (4-pack)	31017
6	Apple Airpods Headphones	15661
7	Bose SoundSport Headphones	13457
8	Flatscreen TV	4819
9	Google Phone	5532
10	LG Dryer	646
11	LG Washing Machine	666
12	Lightning Charging Cable	23217
13	Macbook Pro Laptop	4728
14	ThinkPad Laptop	4130
15	USB-C Charging Cable	23975
16	Vareebadd Phone	2068
17	Wired Headphones	20557
18	iPhone	6849

In [31]:

```
plt.figure(figsize=(9,5))
plt.bar(data=sales_count, x="Product", height="Quantity Ordered")
plt.title("Quantity orders of Products",fontweight="bold",size=15)
plt.xlabel("")
plt.xticks(rotation=90, size=12)
plt.ylabel("Quantity Ordered",size=10)


plt.show()
```



In [72]:
```
prices = sales.groupby("Product").mean()["Price Each"].reset_index()
fig, ax1 = plt.subplots()

ax2= ax1.twinx()
ax1.bar(x= sales_count["Product"], height = sales_count["Quantity Ordered"],color="g")
ax1.set_xticklabels(sales_count["Product"],rotation="vertical")
ax2.plot(prices["Price Each"], color="b")
plt.title("Quantity Ordered vs Price")
plt.show()
```



From the figure above, we can see that when the price of a Product is low, the quantity ordered is High and vice-versa

Solution 5: The most sold product is `AAA batteries (4-pack)` while the least sold products are both the `LG Dryer` and `LG Washing Machine`

Question 6: What day was the highest total sales made???????????????

In [ ]:
```
#Get dataframe for total sales in a single day chronologically/ascending order:
sales_day = sales.groupby("Order Date")["sales"].sum().reset_index()
sales_day["Order Date"] = sales_day["Order Date"].dt.date
sales_day = sales_day.groupby("Order Date")["sales"].sum().sort_values(ascending=False).reset_index()
sales_day
```

Solution 6: The most sales was made on the **4th of December, 2019 (2019-12-04) with a sales total of $166,727** while the least sales was on the **1st of January, 2020 with a sales total of $8,670**. A possible reason to this might be because the vacations and holidays will have ended during that period of time!!!

Question 7: How much was made on Christmas day?????????????

In [ ]:
```
# Christmas was on the 25th of December, 2019
sales_day["Order Date"] = pd.to_datetime(sales_day["Order Date"])
sales_day[sales_day["Order Date"] == "2019-12-25"]
```

Solution 7: On Christmas Day, a sales total of **$152,268.42** was made

Question 7: What is the total sales made by the company in the just concluded year(2019)??????????????????

In [ ]:
```
#augment "year" column:
sales["year"] = sales["Order Date"].dt.year

#Get all sales for the year 2019:
sales_2019 = sales[sales["year"] == 2019]
print("The total sales made in the just concluding year(2019) is:","$",sales_2019["sales"].sum())
```

In [ ]:
```
# #Define the quarters:
# def quarter(month):
#     if 1 <= month <= 3:
#         return "Q1"
#     elif 4 <= month <= 6:
#         return "Q2"
#     elif 7 <= month <= 9:
#         return "Q3"
#     else: return "Q4"
```

In [ ]:
```
# Get the quarter for each record by applying the function on the month field:
# sales_2019["quarter"] = sales_2019["month"].apply(lambda x: f"{quarter(x)}")
```

In [ ]:
```

```

In [ ]:
```

```