Sales Data Analysis

This huge data is about sales of different accessories throughout the whole year of 2019. In this project, We will know about differents trends and Visualisation data is showing and then on that basis we will make data driven decisions. Moreover we will use different functions to clean the data as far as possible because cleaned data will be more efficient for our analysis and moreover we will see that python or more precisely Jupyter Notebook is the only tool with which we can use in place of SQL or spreadsheets. So let's begin this adventure !!!!

How to run the code

This is an executable *Jupyter notebook* (https://jupyter.org) hosted on Jovian.ml (https://www.jovian.ml), a platform for sharing data science projects. You can run and experiment with the code in a couple of ways: *using free online resources* (recommended) or *on your own computer*.

Option 1: Running using free online resources (1-click, recommended)

The easiest way to start executing this notebook is to click the "Run" button at the top of this page, and select "Run on Binder". This will run the notebook on mybinder.org (https://mybinder.org), a free online service for running Jupyter notebooks. You can also select "Run on Colab" or "Run on Kaggle".

Option 2: Running on your computer locally

1. Install Conda by following these instructions (https://conda.io/projects/conda/en/latest/user-guide/install/index.html). Add Conda binaries to your system `PATH`, so you can use the `conda` command on your terminal.
2. Create a Conda environment and install the required libraries by running these commands on the terminal:

```
conda create -n zerotopandas -y python=3.8
conda activate zerotopandas
pip install jovian jupyter numpy pandas matplotlib seaborn opendatasets --upgrade
```

3. Press the "Clone" button above to copy the command for downloading the notebook, and run it on the terminal. This will create a new directory and download the notebook. The command will look something like this:

```
jovian clone notebook-owner/notebook-id
```

4. Enter the newly created directory using `cd directory-name` and start the Jupyter notebook.

```
jupyter notebook
```

You can now access Jupyter's web interface by clicking the link that shows up on the terminal or by visiting http://localhost:8888 (http://localhost:8888) on your browser. Click on the notebook file (it has a `.ipynb` extension) to open it.

Downloading the Dataset

TODO - add some explanation here

In [8]:
```
!pip install jovian opendatasets --upgrade --quiet
```

Let's begin by downloading the data, and listing the files within the dataset.

In [6]:
```
# Change this
dataset_url = 'https://www.kaggle.com/tunguz/us-elections-dataset'
```

In [15]:
```
import opendatasets as od
od.download(dataset_url)
```

```
Kaggle dataset ID:  tunguz/us-elections-dataset

0it [00:00, ?it/s]

Downloading https://www.kaggle.com/tunguz/us-elections-dataset/download?resource=download&downloadHash=7ba6986e70e4d0e9f17ef767bfef5a5fc294114a206832fd436474dafb200649 (https://www.kaggle.com/tunguz/us-elections-dataset/download?resource=download&downloadHash=7ba6986e70e4d0e9f17ef767bfef5a5fc294114a206832fd436474dafb200649) to ./us-elections-dataset.zip

37806080it [00:03, 12417556.13it/s]

Extracting archive ./us-elections-dataset.zip to ./us-elections-dataset
```

The dataset has been downloaded and extracted.

In [1]:
```
import os
os.listdir()
```

Out[1]:
```
['.bash_logout',
 '.profile',
 '.bashrc',
 '.ipynb_checkpoints',
 '.ipython',
 '.local',
 '.cache',
 'zerotopandas-course-project.ipynb',
 'sales_data.csv',
 '.jupyter',
 '.jovian',
 '.config',
 '.conda',
 '.wget-hsts',
 '.jovianrc',
 '.git',
 'work',
 '.npm']
```

Let us save and upload our work to Jovian before continuing.

In [2]:
```
project_name = "sales_data_analysis" # change this (use lowercase letters and hyphens only)
```

In [3]:
```
!pip install jovian --upgrade -q
```

In [4]:
```
import jovian
```

```
<IPython.core.display.Javascript object>
```

In [ ]:
```
jovian.commit(project=project_name)
```

```
<IPython.core.display.Javascript object>
```

Data Preparation and Cleaning

TODO - Here we will clean all the useless data that could create chaos and bias in our data driven decision

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

In [3]:
```
sales_data = pd.read_csv("sales_data.csv")
sales_data.head(7)
```

Out[3]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101
5	194100	iPhone	1	700.0	05/10/19 19:44	81 Jefferson St, San Francisco, CA 94016
6	194101	USB-C Charging Cable	1	11.95	05/11/19 22:44	354 Meadow St, Boston, MA 02215

In [123]:
```
#Clean up the data!
#Drop rows of NAN
# Find NAN
nan_df = sales_data[sales_data.isna().any(axis=1)]
display(nan_df.head())

sales_data = sales_data.dropna(how='all')
```

```
sales_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
58	NaN	NaN	NaN	NaN	NaN	NaN
111	NaN	NaN	NaN	NaN	NaN	NaN
522	NaN	NaN	NaN	NaN	NaN	NaN
839	NaN	NaN	NaN	NaN	NaN	NaN
1590	NaN	NaN	NaN	NaN	NaN	NaN

Out[123]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101

In [124]:

```
#Delete useless Text in Order Date Column
sales_data = sales_data[sales_data['Order Date'].str[0:2]!='Or']
```

In [125]:

```
#Set Correct Data Types
sales_data['Quantity Ordered'] = pd.to_numeric(sales_data['Quantity Ordered'])
sales_data['Price Each'] = pd.to_numeric(sales_data['Price Each'])
```

In [126]:

```
#Add a new Column named as Month
sales_data['Month'] = sales_data['Order Date'].str[0:2]
sales_data['Month'] = sales_data['Month'].astype('int32')
sales_data.head()
```

Out[126]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001	5
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001	5
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001	5
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001	5
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101	5

In [127]:

```
#Add a new Column named as City
def city(address):
    return address.split(",")[1].strip(" ")
```

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

sales_data['City'] = sales_data['Purchase Address'].apply(lambda x: f"{city(x)}  ({state(x)})")
sales_data.head()
```

Out[127]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001	5	New York City (NY)
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001	5	Dallas (TX)
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001	5	New York City (NY)
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001	5	Dallas (TX)
4	194099	AAA Batteries (4-pack)	2	2.99	05/11/19 22:55	17 Church St, Seattle, WA 98101	5	Seattle (WA)

In [128]:

```
import jovian
```

In [129]:

```
jovian.commit()
```

```
<IPython.core.display.Javascript object>

[jovian] Updating notebook "mohammadowaisprofessional/sales-data-analysis" on https://jovian.ai (https://jovian.ai)
[jovian] Committed successfully! https://jovian.ai/mohammadowaisprofessional/sales-data-analysis (https://jovian.ai/mohammadowaisprofessional/sales-data-analysis)
```

Out[129]:

```
'https://jovian.ai/mohammadowaisprofessional/sales-data-analysis'
```

Exploratory Analysis and Visualization

TODO - Here we will ask some questions about data and check wether data is giving us right answers or appropriate answers

Let's begin by importing `matplotlib.pyplot` and `seaborn`.

In [130]:

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

sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

TODO - Explore one or more columns by plotting a graph below, and add some explanation about it

In [131]:

```
#In which month sales was highest??
sales_data['Sales'] = sales_data['Quantity Ordered'].astype('int') * sales_data['Price Each'].astype('float')
sales_data.groupby(['Month']).sum()
```

Out[131]:

	Quantity Ordered	Price Each	Sales
Month			
1	21806	3623536.76	3644513.46
2	26898	4377769.44	4404044.84
3	34010	5582415.66	5614200.76
4	41116	6735342.04	6781340.48
5	37334	6270250.26	6305213.50
6	30506	5124051.22	5155604.52
7	32144	5265079.12	5295551.52
8	26896	4460690.84	4488935.76
9	26218	4169984.18	4195120.26
10	45406	7431109.66	7473453.76
11	39596	6361201.36	6399206.40
12	56228	9176830.82	9226886.68

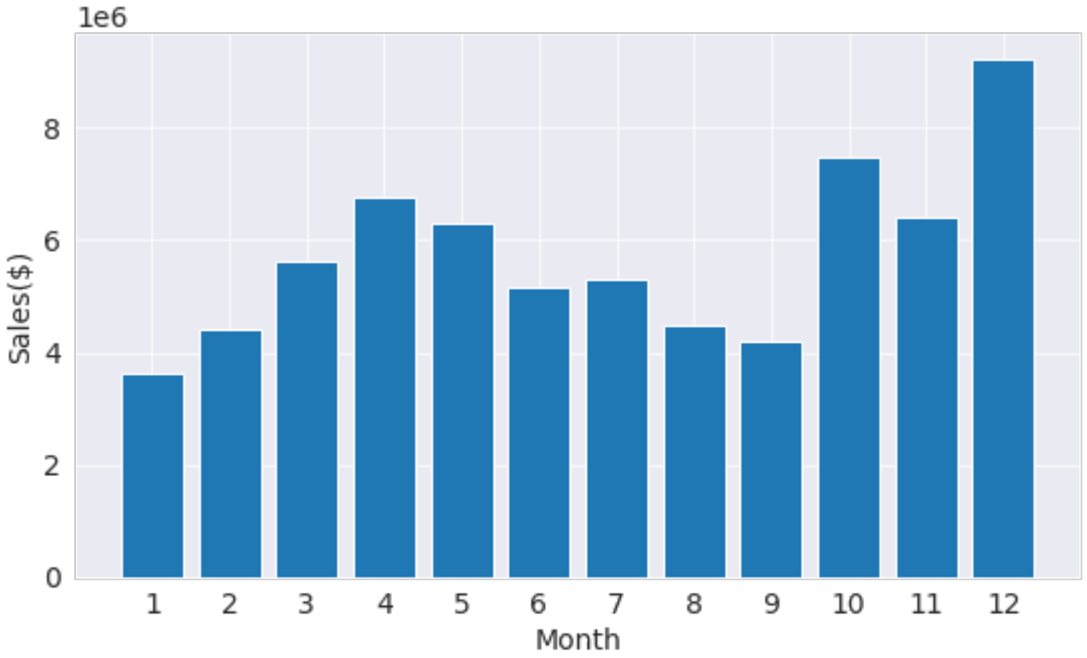
In [132]:

```python
import matplotlib.pyplot as plt

months = range(1,13)
print(months)

plt.bar(months,sales_data.groupby(['Month']).sum()['Sales'])
plt.xticks(months)
plt.ylabel('Sales($)')
plt.xlabel('Month')
plt.show()
```

```
range(1, 13)
```



TODO - Explore one or more columns by plotting a graph below, and add some explanation about it

In [133]:

```python
#In which city most product was sold???
sales_data.groupby(['City']).sum()
```

Out[133]:

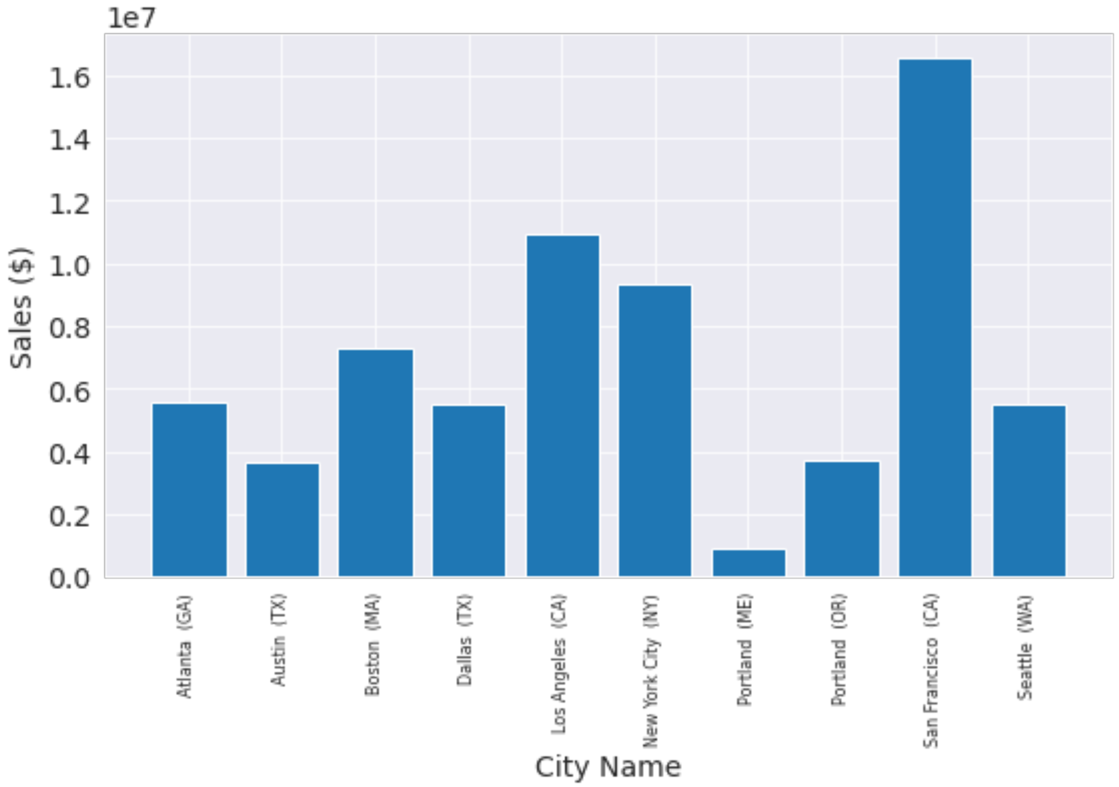
	Quantity Ordered	Price Each	Month	Sales
City				
Atlanta (GA)	33204	5559816.40	209588	5590997.16
Austin (TX)	22306	3619747.22	139658	3639163.50
Boston (MA)	45056	7274819.54	282224	7323284.02
Dallas (TX)	33460	5505255.64	209240	5535950.80

	Quantity Ordered	Price Each	Month	Sales
City				
Los Angeles (CA)	66578	10842870.46	416650	10905141.60
New York City (NY)	55864	9270741.66	351482	9328634.86
Portland (ME)	5500	894378.50	34288	899516.54
Portland (OR)	22606	3721116.44	141242	3741464.68

In [134]:
```
import matplotlib.pyplot as plt

keys = [city for city, df in sales_data.groupby(['City'])]

plt.bar(keys,sales_data.groupby(['City']).sum()['Sales'])
plt.ylabel('Sales ($)')
plt.xlabel('City Name')
plt.xticks(keys, rotation='vertical', size=8)
plt.show()
```



TODO - Explore one or more columns by plotting a graph below, and add some explanation about it

In [135]:

```
#List of product which was sold most??
sales_data.groupby(['Product']).sum()
```

Out[135]:

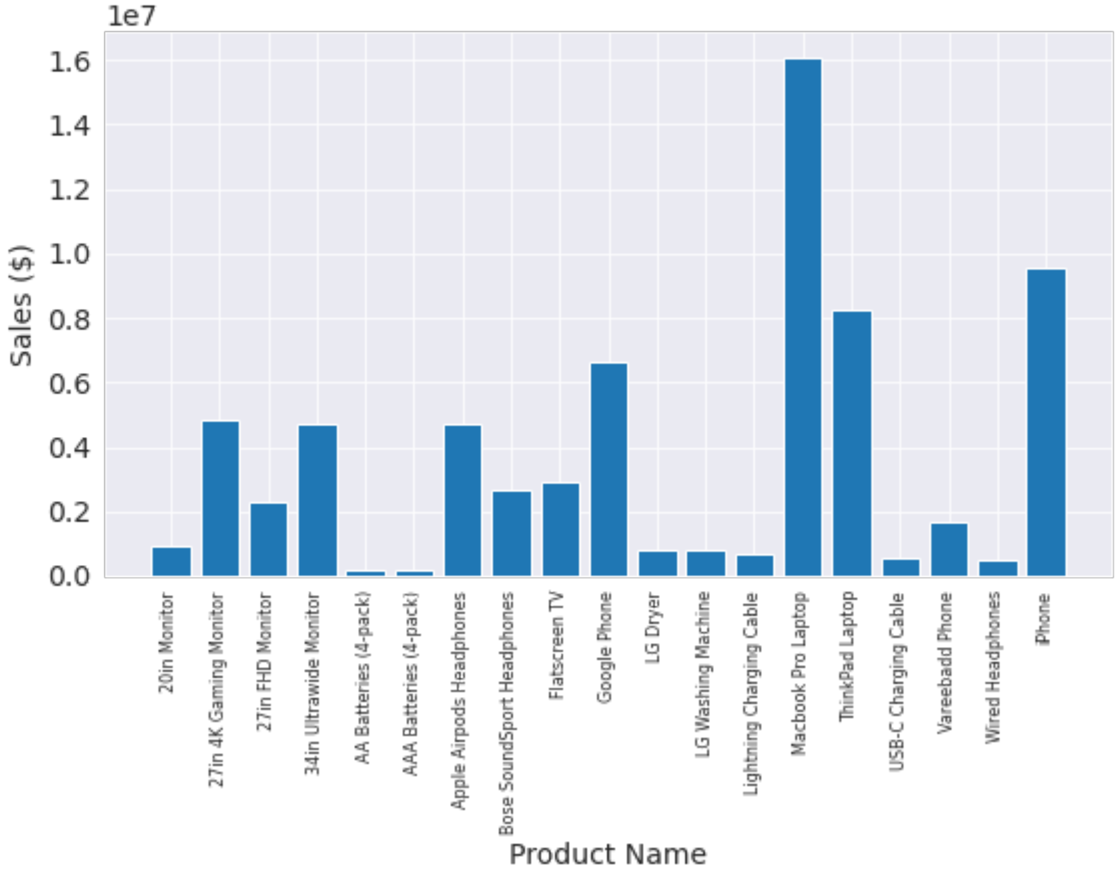
	Quantity Ordered	Price Each	Month	Sales
Product				
20in Monitor	8258	902137.98	58672	908297.42
27in 4K Gaming Monitor	12488	4859275.40	88880	4870195.12
27in FHD Monitor	15100	2251949.86	105116	2264849.00
34in Ultrawide Monitor	12398	4697436.38	86608	4711116.02
AA Batteries (4-pack)	55270	158031.36	291116	212236.80
AAA Batteries (4-pack)	62034	123433.18	292740	185481.66
Apple Airpods Headphones	31322	4664700.00	218954	4698300.00
Bose SoundSport Headphones	26914	2664733.50	188226	2691130.86
Flatscreen TV	9638	2880000.00	68448	2891400.00
Google Phone	11064	6630000.00	76610	6638400.00
LG Dryer	1292	775200.00	8766	775200.00
LG Washing Machine	1332	799200.00	9046	799200.00
Lightning Charging Cable	46434	647574.20	306184	694188.30
Macbook Pro Laptop	9456	16061600.00	67096	16075200.00
ThinkPad Laptop	8260	8255917.44	57900	8259917.40
USB-C Charging Cable	47950	523481.70	309638	573002.50
Vareebadd Phone	4136	1652000.00	28618	1654400.00
Wired Headphones	41114	452790.36	266794	492956.86
iPhone	13698	9578800.00	95882	9588600.00

In [136]:

```
import matplotlib.pyplot as plt

keys = [Product for Product, df in sales_data.groupby(['Product'])]

plt.bar(keys,sales_data.groupby(['Product']).sum()['Sales'])
plt.ylabel('Sales ($)')
plt.xlabel('Product Name')
plt.xticks(keys, rotation='vertical', size=8)
plt.show()
```



TODO - Explore one or more columns by plotting a graph below, and add some explanation about it

In [137]:

```
#what should be the appropriate time for displaying advertisement which would attract more customer???
sales_data['Hour'] = pd.to_datetime(sales_data['Order Date']).dt.hour
sales_data['Minute'] = pd.to_datetime(sales_data['Order Date']).dt.minute
sales_data['Count'] = 1
sales_data.head()
```

Out[137]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City	Sales	Hour	Minute	Count

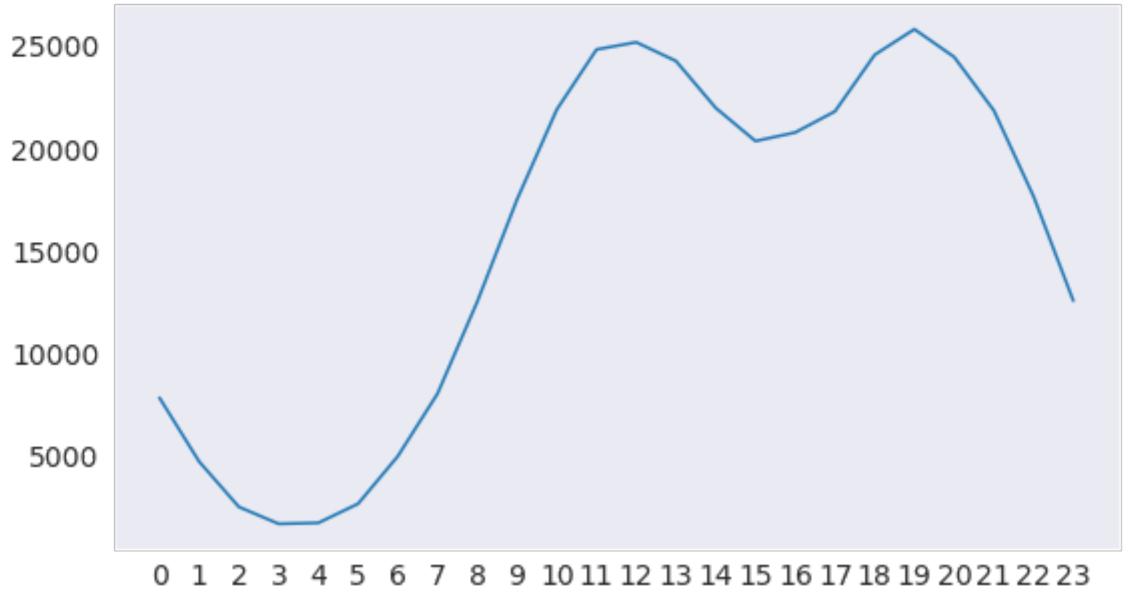
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City	Sales	Hour	Minute	Count
0	194095	Wired Headphones	1	11.99	05/16/19 17:14	669 2nd St, New York City, NY 10001	5	New York City (NY)	11.99	17	14	1
1	194096	AA Batteries (4-pack)	1	3.84	05/19/19 14:43	844 Walnut St, Dallas, TX 75001	5	Dallas (TX)	3.84	14	43	1
2	194097	27in FHD Monitor	1	149.99	05/24/19 11:36	164 Madison St, New York City, NY 10001	5	New York City (NY)	149.99	11	36	1
3	194098	Wired Headphones	1	11.99	05/02/19 20:40	622 Meadow St, Dallas, TX 75001	5	Dallas (TX)	11.99	20	40	1

In [138]:

```
keys = [pair for pair, df in sales_data.groupby(['Hour'])]

plt.plot(keys, sales_data.groupby(['Hour']).count()['Count'])
plt.xticks(keys)
plt.grid()
plt.show()

# I like to go for shopping at 6 pm  evening
```



TODO - Explore one or more columns by plotting a graph below, and add some explanation about it

In [139]:

```
#which products are sold in groups more oftenly??
df = sales_data[sales_data['Order ID'].duplicated(keep=False)]\

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

In [140]:

```
#This module implements a number of iterator building blocks inspired by constructs from APL, Haskell, and SML. Each has been recast

#The module standardizes a core set of fast, memory efficient tools that are useful by themselves or in combination. Together, they
```

```
from itertools import combinations
from collections import Counter

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(10):
    print(key, value)
```

```
('USB-C Charging Cable', 'USB-C Charging Cable') 22095
('Lightning Charging Cable', 'Lightning Charging Cable') 21874
('AAA Batteries (4-pack)', 'AAA Batteries (4-pack)') 20833
('AA Batteries (4-pack)', 'AA Batteries (4-pack)') 20717
('Wired Headphones', 'Wired Headphones') 19022
('Apple Airpods Headphones', 'Apple Airpods Headphones') 15645
('Bose SoundSport Headphones', 'Bose SoundSport Headphones') 13433
('27in FHD Monitor', '27in FHD Monitor') 7543
('iPhone', 'iPhone') 6850
('27in 4K Gaming Monitor', '27in 4K Gaming Monitor') 6250
```

Let us save and upload our work to Jovian before continuing

In [141]:

```
import jovian
```

In [142]:

```
jovian.commit()
```

```
<IPython.core.display.Javascript object>

[jovian] Updating notebook "mohammadowaisprofessional/sales-data-analysis" on https://jovian.ai (https://jovian.ai)
[jovian] Committed successfully! https://jovian.ai/mohammadowaisprofessional/sales-data-analysis (https://jovian.ai/mohammadowaispr
ofessional/sales-data-analysis)
```

Out[142]:

```
'https://jovian.ai/mohammadowaisprofessional/sales-data-analysis'
```

Asking and Answering Questions

TODO - write some explanation here.

Q1: In which month sales was highest ?

In [143]:

```
sales_data.groupby(['Month']).sum()
```

Out[143]:

	Quantity Ordered	Price Each	Sales	Hour	Minute	Count

Month						
1	21806	3623536.76	3644513.46	278970	564880	19418
2	26898	4377769.44	4404044.84	345338	709770	23950
3	34010	5582415.66	5614200.76	437938	895118	30306
4	41116	6735342.04	6781340.48	524518	1088372	36558
5	37334	6270250.26	6305213.50	477560	975798	33132
6	30506	5124051.22	5155604.52	391056	804872	27108
7	32144	5265079.12	5295551.52	412338	834698	28586
8	26896	4460690.84	4488935.76	344578	707714	23922
9	26218	4169984.18	4195120.26	337026	683396	23242
10	45406	7431109.66	7473453.76	581300	1196874	40564
11	39596	6361201.36	6399206.40	509730	1036462	35146
12	56228	9176830.82	9226886.68	719956	1466164	49968

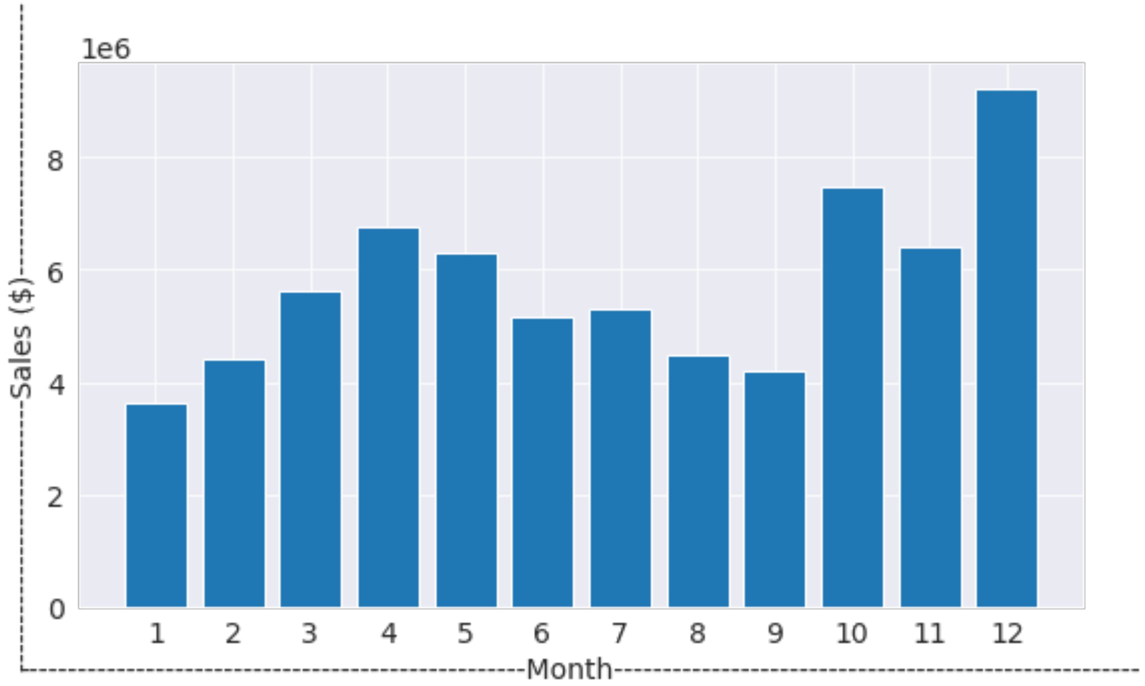
In [144]:

```
import matplotlib.pyplot as plt

months = range(1,13)
print(months)

plt.bar(months,sales_data.groupby(['Month']).sum()['Sales'])
plt.xticks(months)
plt.ylabel('----------------------------Sales ($)----------------------------')
plt.xlabel('----------------------------------------------------Month------------------------------------------------------')
plt.show()
```

```
range(1, 13)
```



In [145]:

```
## As we can see from graph in decembers customers have bought a lot of product!!
```

Q2: Name the city where greatest number of products were sold ?

In [146]:

```
sales_data.groupby(['City']).sum()
```

Out[146]:

	Quantity Ordered	Price Each	Month	Sales	Hour	Minute	Count
City							
Atlanta (GA)	33204	5559816.40	209588	5590997.16	428528	885864	29762
Austin (TX)	22306	3619747.22	139658	3639163.50	283892	578120	19810

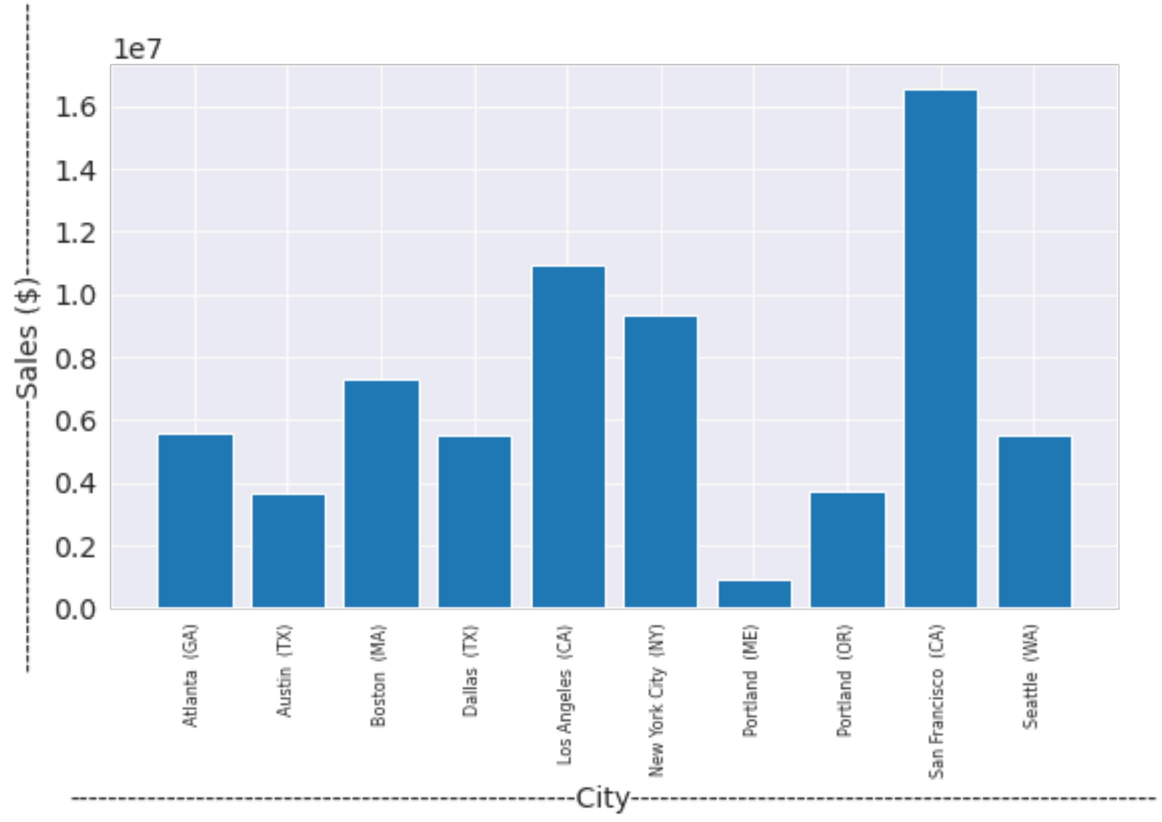
	Quantity Ordered	Price Each	Month	Sales	Hour	Minute	Count
City							
Boston (MA)	45056	7274819.54	282224	7323284.02	576450	1180884	39868
Dallas (TX)	33460	5505255.64	209240	5535950.80	428780	870310	29640
Los Angeles (CA)	66578	10842870.46	416650	10905141.60	854888	1733276	59210
New York City (NY)	55864	9270741.66	351482	9328634.86	715392	1467196	49752
Portland (ME)	5500	894378.50	34288	899516.54	70422	145712	4910
Portland (OR)	22606	3721116.44	141242	3741464.68	288842	591066	20020

In [147]:

```
import matplotlib.pyplot as plt

keys = [city for city, df in sales_data.groupby(['City'])]

plt.bar(keys,sales_data.groupby(['City']).sum()['Sales'])
plt.ylabel('----------------------------Sales ($)----------------------------')
plt.xlabel('----------------------------------------------------City------------------------------------------------------')
plt.xticks(keys, rotation='vertical', size=8)
plt.show()
```



In [148]:
```
## As we can see from graph San Francisco was the place where customers bought greatest
```

Q3: list the products that were sold most??

In [149]:
```
sales_data.groupby(['Product']).sum()
```

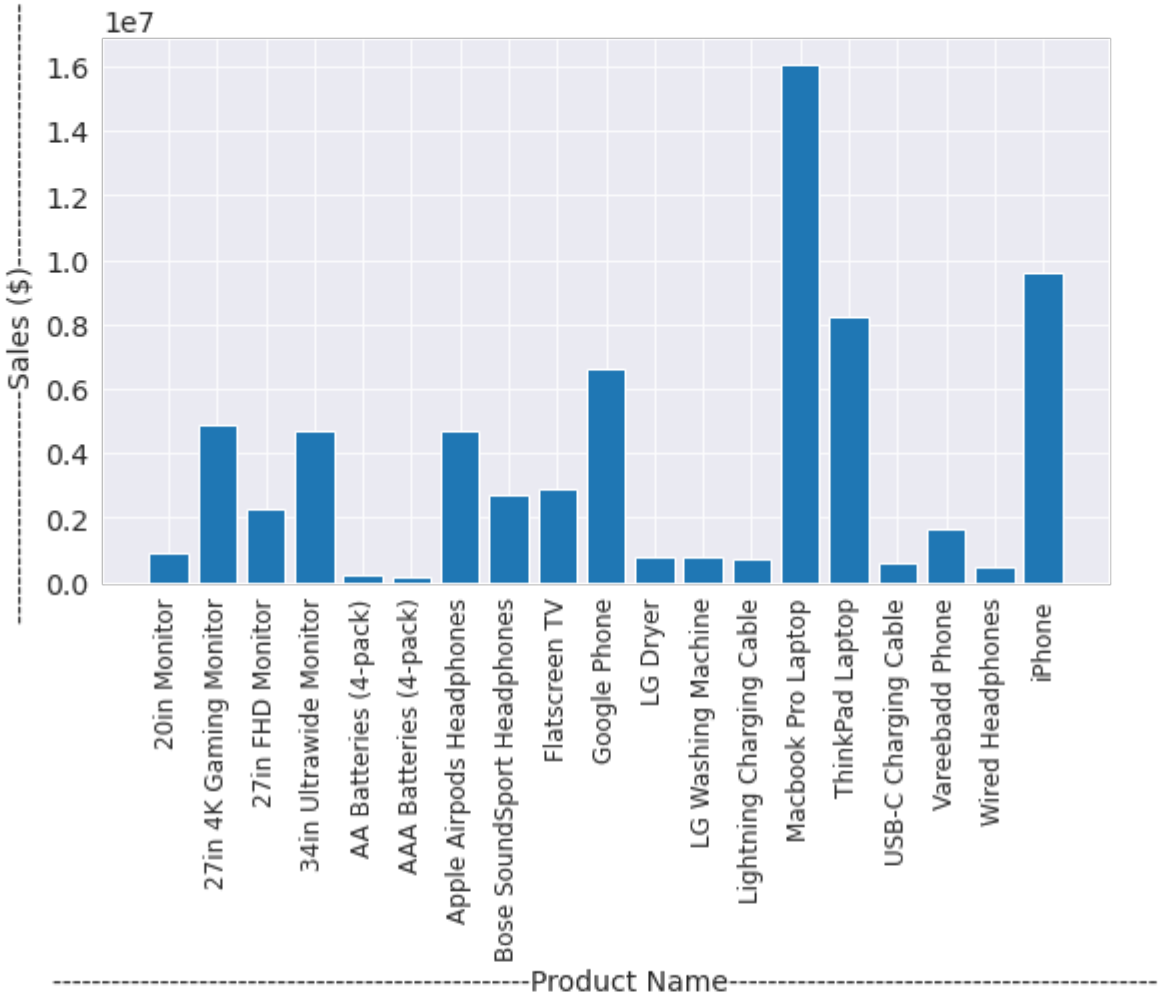
Out[149]:

	Quantity Ordered	Price Each	Month	Sales	Hour	Minute	Count
Product							
20in Monitor	8258	902137.98	58672	908297.42	117528	244504	8202
27in 4K Gaming Monitor	12488	4859275.40	88880	4870195.12	181832	368662	12460
27in FHD Monitor	15100	2251949.86	105116	2264849.00	215080	439896	15014
34in Ultrawide Monitor	12398	4697436.38	86608	4711116.02	178152	366960	12362
AA Batteries (4-pack)	55270	158031.36	291116	212236.80	596684	1218078	41154
AAA Batteries (4-pack)	62034	123433.18	292740	185481.66	594664	1224226	41282
Apple Airpods Headphones	31322	4664700.00	218954	4698300.00	446608	911140	31098
Bose SoundSport Headphones	26914	2664733.50	188226	2691130.86	384890	785206	26650
Flatscreen TV	9638	2880000.00	68448	2891400.00	137630	285578	9600
Google Phone	11064	6630000.00	76610	6638400.00	158958	325546	11050
LG Dryer	1292	775200.00	8766	775200.00	18652	38086	1292
LG Washing Machine	1332	799200.00	9046	799200.00	19570	38924	1332
Lightning Charging Cable	46434	647574.20	306184	694188.30	625058	1268884	43316
Macbook Pro Laptop	9456	16061600.00	67096	16075200.00	136522	275148	9448
ThinkPad Laptop	8260	8255917.44	57900	8259917.40	119492	243016	8256
USB-C Charging Cable	47950	523481.70	309638	573002.50	629290	1295172	43806
Vareebadd Phone	4136	1652000.00	28618	1654400.00	58944	123670	4130
Wired Headphones	41114	452790.36	266794	492956.86	543440	1108046	37764
iPhone	13698	9578800.00	95882	9588600.00	197314	403376	13684

In [150]:
```
import matplotlib.pyplot as plt

keys = [Product for Product, df in sales_data.groupby(['Product'])]

plt.bar(keys,sales_data.groupby(['Product']).sum()['Sales'])
plt.ylabel('------------------------Sales ($)---------------------------')
plt.xlabel('-------------------------------------------------Product Name--------------------------------------------')
plt.xticks(keys, rotation='vertical', size=12)
plt.show()
```

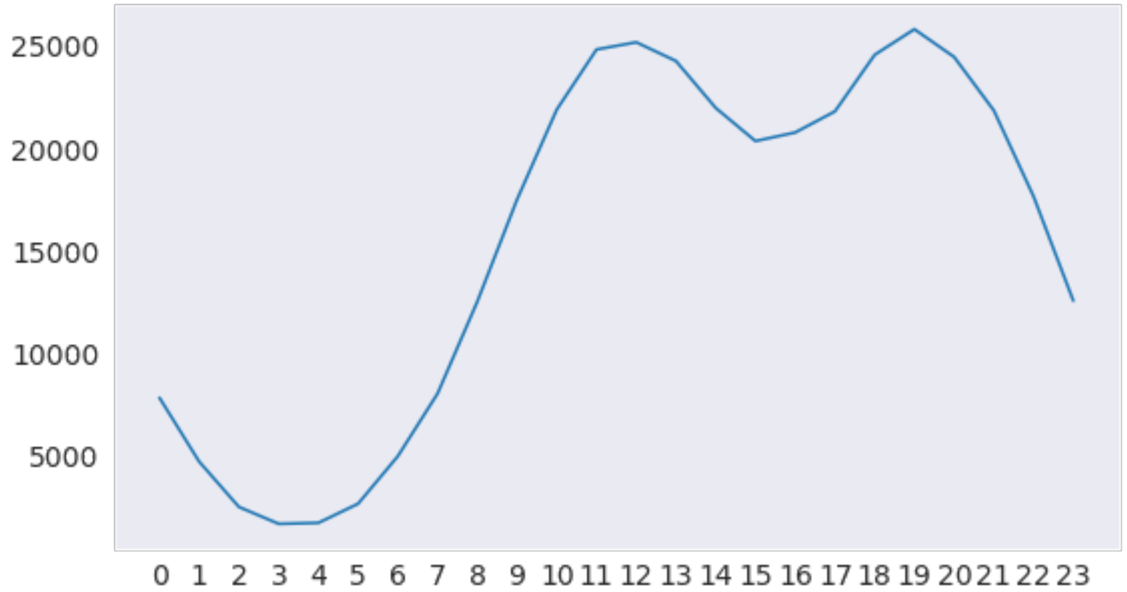


In [151]:
```
## As we can see from graph Mackbook Pro Laptop was most sold product of 2019
```

Q4: what should be the appropriate time for displaying advertisement which would attract more customer???

In [152]:
```
keys = [pair for pair, df in sales_data.groupby(['Hour'])]

plt.plot(keys, sales_data.groupby(['Hour']).count()['Count'])
plt.xticks(keys)
plt.grid()
plt.show()
```



In [153]:
```
#I like to go for shopping at 6pm evening
```

In [ ]:
```
```

Q5: which products are sold in groups more oftenly??

In [154]:
```
df = sales_data[sales_data['Order ID'].duplicated(keep=False)]\

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

In [155]:
```
from itertools import combinations
from collections import Counter

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(10):
    print(key, value)
```

```
('USB-C Charging Cable', 'USB-C Charging Cable') 22095
('Lightning Charging Cable', 'Lightning Charging Cable') 21874
('AAA Batteries (4-pack)', 'AAA Batteries (4-pack)') 20833
('AA Batteries (4-pack)', 'AA Batteries (4-pack)') 20717
('Wired Headphones', 'Wired Headphones') 19022
('Apple Airpods Headphones', 'Apple Airpods Headphones') 15645
('Bose SoundSport Headphones', 'Bose SoundSport Headphones') 13433
('27in FHD Monitor', '27in FHD Monitor') 7543
('iPhone', 'iPhone') 6850
('27in 4K Gaming Monitor', '27in 4K Gaming Monitor') 6250
```

In [156]:

```
## As we can see from graph USB-C Charging Cables are most oftenly sold in group
```

Let us save and upload our work to Jovian before continuing.

In [157]:

```
import jovian
```

In [158]:

```
jovian.commit()
```

```
<IPython.core.display.Javascript object>

[jovian] Updating notebook "mohammadowaisprofessional/sales-data-analysis" on https://jovian.ai (https://jovian.ai)
[jovian] Committed successfully! https://jovian.ai/mohammadowaisprofessional/sales-data-analysis (https://jovian.ai/mohammadowaispr
ofessional/sales-data-analysis)
```

Out[158]:

```
'https://jovian.ai/mohammadowaisprofessional/sales-data-analysis'
```

Inferences and Conclusion

TODO - Through all data analysis I did above I learned a first hand experience of dealing with data, How to make data driven decisions with the help of data and try to be unbiased as far as possible throughout data and that's it !!!

https://docs.python.org/3/library/itertools.html (https://docs.python.org/3/library/itertools.html)

In [159]:

```
import jovian
```

In [ ]:

```
jovian.commit()
```

```
<IPython.core.display.Javascript object>
```

References and Future Work

TODO - I found sales data from github which was really huge and dealing with huge data increase your experience with dealing problems

```
In [ ]: import jovian
```

```
In [ ]: jovian.commit()
```

```
In [ ]:
```