# DATA ANALYSIS
# ~Python Libraries

# GETTING STARTED WITH NUMPY

To import NumPy in your Python script or notebook:

python Copy

```python
import numpy as np
```

Copy

```
---

#### **3. Creating NumPy Arrays**
```markdown
# Creating NumPy Arrays

NumPy arrays are the core data structure in NumPy. They are more efficient than Python lists fo
r numerical computations.

### Creating a 1D Array
```python
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

# GETTING STARTED WITH NUMPY

## Creating a 2D Array

python Copy

```python
arr_2d = np.array([[1, 2, 3], [4, 5, 6]])
print(arr_2d)
```

## Array Properties

python Copy Edit

```python
arr.shape  # Returns (rows, columns)
arr.size   # Total number of elements
arr.dtype  # Data type of elements
arr.ndim   # Number of dimensions
```

## Indexing

python Copy Edit

```python
arr = np.array([10, 20, 30, 40, 50])
print(arr[2])  # Output: 30
```

## Slicing

python Copy Edit

```python
arr = np.array([1, 2, 3, 4, 5, 6])
print(arr[1:4])   # Output: [2 3 4]
print(arr[:3])    # First 3 elements
print(arr[-2:])   # Last 2 elements
```

## 2D Array Indexing

python Copy Edit

```python
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr[1, 2])   # Row 1, Column 2 → Output: 6
print(arr[:, 1])   # Second column
```

## Mathematical Operations

python Copy Edit

```python
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print(a + b)  # Element-wise addition
print(a * b)  # Element-wise multiplication
print(a / b)  # Element-wise division
print(a ** 2) # Squaring elements
```

## Aggregate Functions

python Copy Edit

```python
arr = np.array([10, 20, 30])

print(np.sum(arr))   # Sum of elements
print(np.mean(arr))  # Mean
print(np.median(arr))# Median
print(np.std(arr))   # Standard deviation
print(np.min(arr))   # Minimum value
print(np.max(arr))   # Maximum value
```

## Reshaping

python Copy Edit

```python
arr = np.array([1, 2, 3, 4, 5, 6])
arr.reshape(2, 3)  # Converts into 2x3 matrix
```

## Flattening

python Copy Edit

```python
arr.flatten()  # Converts into 1D array
```

## Transpose

python Copy Edit

```python
arr.T  # Swaps rows and columns
```

# Data Manipulation Using Pandas

## 1. Installing and Importing Pandas

You can install Pandas using pip:

bash Copy

```bash
pip install pandas
```

Import Pandas in your script or notebook:

python Copy

```python
import pandas as pd
```

# KEY DATA STRUCTURES IN PANDAS

1. Series
2. DataFrames

### a. Series

A **Series** is a one-dimensional array-like object that can hold any data type. It has an index that labels each element.

python Copy

```python
# Creating a Series
s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e'])
print(s)
```

### b. DataFrame

A **DataFrame** is a two-dimensional table-like structure with rows and columns. It is similar to a spreadsheet or SQL table.

python Copy

```python
# Creating a DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
print(df)
```

## 3. Loading Data

Pandas can read data from various file formats like CSV, Excel, SQL, and more.

### Reading a CSV File

python Copy

```python
df = pd.read_csv('data.csv')
print(df.head())  # Display the first 5 rows
```

### Reading an Excel File

python Copy

```python
df = pd.read_excel('data.xlsx')
print(df.head())
```

# Exploring Data

## Viewing Data

- `head(n)` : Displays the first `n` rows.
- `tail(n)` : Displays the last `n` rows.
- `info()` : Provides a summary of the DataFrame.
- `describe()` : Generates descriptive statistics.

python Copy

```python
print(df.head())
print(df.info())
print(df.describe())
```

## Accessing Columns and Rows

- Access a column: `df['ColumnName']` or `df.ColumnName`
- Access a row: `df.loc[index]` or `df.iloc[index]`

python Copy

```python
print(df['Name'])  # Access the 'Name' column
print(df.loc[0])   # Access the first row
```

# Data manipulation

### Adding a New Column

```python
df['Salary'] = [70000, 80000, 90000]
print(df)
```

### Dropping Columns or Rows

- Drop columns: `df.drop(columns=['ColumnName'])`
- Drop rows: `df.drop(index=[0, 1])`

```python
df = df.drop(columns=['Salary'])
print(df)
```

### Sorting Data

- Sort by column: `df.sort_values(by='ColumnName')`

```python
sorted_df = df.sort_values(by='Age', ascending=False)
print(sorted_df)
```

# Data manipulation

## Detecting Missing Data

python Copy

```python
print(df.isnull())
```

## Dropping Missing Data

python Copy

```python
df = df.dropna()
```

## Filling Missing Data

python Copy

```python
df = df.fillna(value=0)  # Fill NaN with 0
```

# Data manipulation

## Grouping Data

python Copy

```python
grouped_df = df.groupby('City').mean()
print(grouped_df)
```

## Aggregation Functions

- `sum()`, `mean()`, `count()`, `min()`, `max()`

python Copy

```python
print(df['Age'].mean())
```

# Data manipulation

## Merging DataFrames

python Copy

```python
df1 = pd.DataFrame({'Key': ['A', 'B', 'C'], 'Value': [1, 2, 3]})
df2 = pd.DataFrame({'Key': ['B', 'C', 'D'], 'Value': [4, 5, 6]})

merged_df = pd.merge(df1, df2, on='Key', how='inner')
print(merged_df)
```

## Concatenating DataFrames

python Copy

```python
concatenated_df = pd.concat([df1, df2], axis=0)
print(concatenated_df)
```

## 9. Applying Functions

You can apply functions to columns or rows using `apply()`.

python Copy

```python
df['Age'] = df['Age'].apply(lambda x: x + 1)
print(df)
```

# Reading Data From Different sources using pandas

python Udemy

EXPLORER

OPEN EDITORS
- pandas.ipynb
- pandasasssign...

PYTHON UDEMY
- numpypandasassign...
- numpyassignment...
- numpysolutions.ip...
- pandasasssignmen...
- pandassolution.ipy...
- venv
- Basics.ipynb
- customers.csv
- marks.csv
- pandas.ipynb
- requirements.txt

OUTLINE

TIMELINE

pandas.ipynb | pandasasssignments.ipynb

pandas.ipynb > df[1]

+ Code + Markdown | Run All | Restart | Clear All Outputs | Variables | Outline ... venv (Python 3.12.0)

# READING DATA FROM DIFFERENT SOURCES USING PANDAS

```
%pip install lxml
```

[11] ✓ 1.4s Python

```
Collecting lxml
  Downloading lxml-5.3.1-cp312-cp312-macosx_10_9_universal2.whl.metadata (3.7 kB)
Downloading lxml-5.3.1-cp312-cp312-macosx_10_9_universal2.whl (8.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.2/8.2 MB 30.8 MB/s eta 0:00:00a 0:00:01
Installing collected packages: lxml
Successfully installed lxml-5.3.1
Note: you may need to restart the kernel to use updated packages.
```

```
url = "https://www.fdic.gov/bank-failures/failed-bank-list"
df = pd.read_html(url)
```

[16] ✓ 0.0s Python

```
df[0]
```

[18] ✓ 0.0s Python

| | Bank Name | City | State | Cert | Acquiring Institution | Closing Date | Fund Sort ascending |
|---|---|---|---|---|---|---|---|
| 0 | Pulaski Savings Bank | Chicago | Illinois | 28611 | Millennium Bank | January 17, 2025 | 10548 |
| 1 | The First National Bank of Lindsay | Lindsay | Oklahoma | 4134 | First Bank & Trust Co., Duncan, OK | October 18, 2024 | 10547 |
| 2 | Republic First Bank dba Republic Bank | Philadelphia | Pennsylvania | 27332 | Fulton Bank, National Association | April 26, 2024 | 10546 |
| 3 | Citizens Bank | Sac City | Iowa | 8758 | Iowa Trust & Savings Bank | November 3, 2023 | 10545 |

# EXAMPLE : importing mobile country codes from wikipedia page using pandas

```
url = "https://en.wikipedia.org/wiki/Mobile_country_code"
df = pd.read_html(url)
```

[19] ✓ 0.8s Python

```
df[1]
```

[24] ✓ 0.0s Python

| | Mobile country code | Country | ISO 3166 | Mobile network codes | National MNC authority | Remarks |
|---|---|---|---|---|---|---|
| 0 | 289 | A Abkhazia | GE-AB | List of mobile network codes in Abkhazia | NaN | MCC is not listed by ITU |
| 1 | 412 | Afghanistan | AF | List of mobile network codes in Afghanistan | NaN | NaN |
| 2 | 276 | Albania | AL | List of mobile network codes in Albania | NaN | NaN |
| 3 | 603 | Algeria | DZ | List of mobile network codes in Algeria | NaN | NaN |
| 4 | 544 | American Samoa (United States of America) | AS | List of mobile network codes in American Samoa | NaN | NaN |
| ... | ... | ... | ... | ... | ... | ... |
| 247 | 452 | Vietnam | VN | List of mobile network codes in the Vietnam | NaN | NaN |
| 248 | 543 | W Wallis and Futuna | WF | List of mobile network codes in Wallis and Futuna | NaN | NaN |
| 249 | 421 | Y Yemen | YE | List of mobile network codes in the Yemen | NaN | NaN |
| 250 | 645 | Z Zambia | ZM | List of mobile network codes in Zambia | NaN | NaN |

## Creating and Converting Excel/ csv file into PICKLE file

```
df_excel.to_pickle('df_excel')
```

[46] ✓ 0.0s Python

```
pd.read_pickle('df_excel')
```

[47] ✓ 0.0s Python

| | Index | Customer Id | First Name | Last Name | Company | City | Country | Phone 1 | Phone 2 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | DD37Cf93aecA6Dc | Sheryl | Baxter | Rasmussen Group | East Leonard | Chile | 229.077.5154 | 397.884.0519x718 | zunigavanessa |
| 1 | 2 | 1Ef7b82A4CAAD10 | Preston | Lozano | Vega-Gentry | East Jimmychester | Djibouti | 5153435776 | 686-620-1820x944 | vmata |
| 2 | 3 | 6F94879bDAfE5a6 | Roy | Berry | Murillo-Perry | Isabelborough | Antigua and Barbuda | +1-539-402-0259 | (496)978-3969x58947 | beckycarr@ |
| 3 | 4 | 5Cef8BFA16c5e3c | Linda | Olsen | Dominguez, Mcmillan and Donovan | Bensonview | Dominican Republic | 001-808-617-6467x12895 | +1-813-324-8756 | stanleyblackwell@ |
| 4 | 5 | 053d585Ab6b3159 | Joanna | Bender | Martin, Lang and Andrade | West Priscilla | Slovakia (Slovak Republic) | 001-234-203-0635x76146 | 001-199-446-3860x3486 | colinalvarad |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 95 | 96 | cb8E23e48d22Eae | Karl | Greer | Carey LLC | East Richard | Guyana | (188)169-1674x58692 | 001-841-293-3519x614 | hhart@ |
| 96 | 97 | CeD220bdAaCfaDf | Lynn | Atkinson | Ware, Burns and Oneal | New Bradview | Sri Lanka | +1-846-706-2218 | 605.413.3198 | vkemp |
| | | | | | Schmitt- | | Solomon | +1-753-067- | +1-632-666- | |

# Matplotlib

# Matplotlib

```
%pip install matplotlib
```

[3] ✓ 3.5s Python

```
Collecting matplotlib
  Downloading matplotlib-3.10.1-cp312-cp312-macosx_11_0_arm64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.1-cp312-cp312-macosx_11_0_arm64.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from matplotlib)
```

```
import matplotlib.pyplot as plt
```

[4] ✓ 13.8s

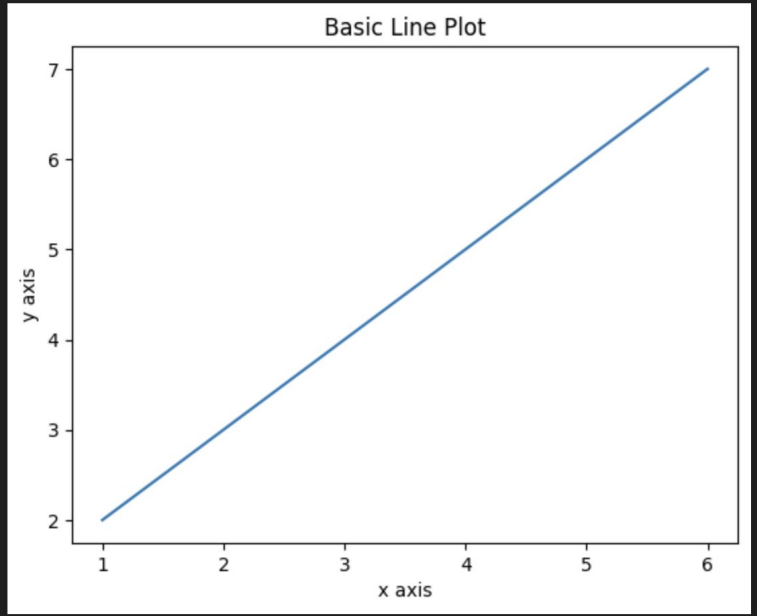
```
Matplotlib is building the font cache; this may take a moment.
```

# Line Plot

```
x = [1,2,3,4,5,6]
y = [2,3,4,5,6,7]

##create a line plot
plt.plot(x,y)
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.title("Basic Line Plot")
```

[6] ✓ 0.0s

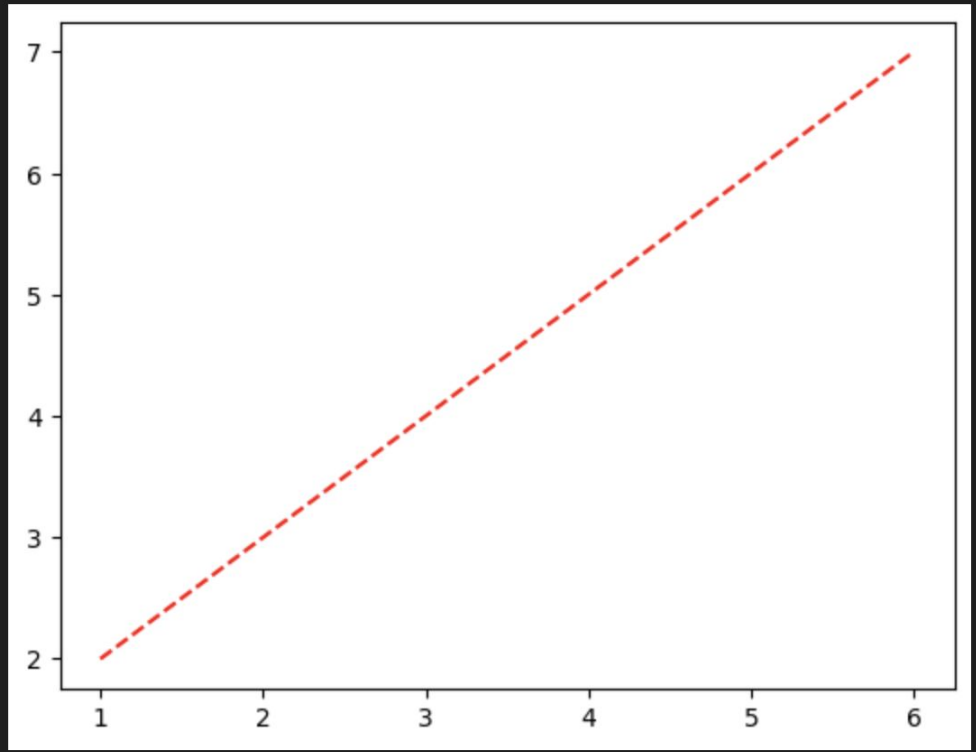
```
Text(0.5, 1.0, 'Basic Line Plot')
```



# Line Plot

```
[7]  plt.plot(x,y,color='red',linestyle='--')
     ✓ 0.0s
```

```
[<matplotlib.lines.Line2D at 0x10e4d7890>]
```



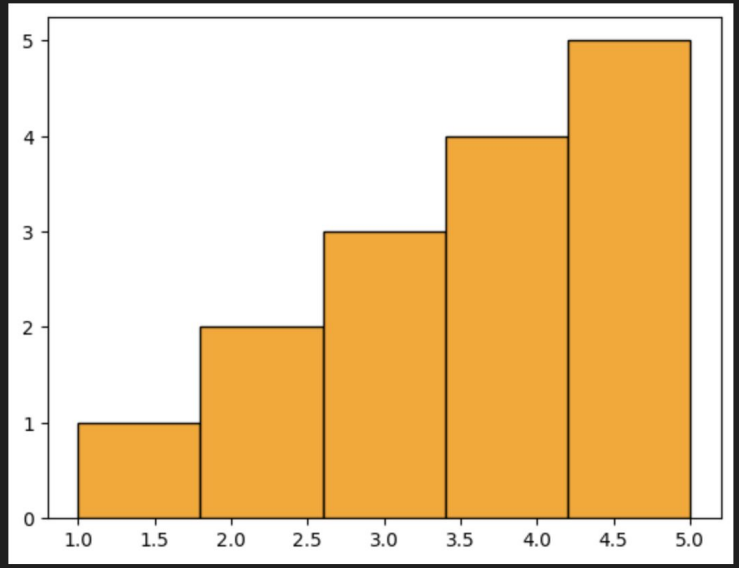
# Histogram:

Histogram

```
data = [1,2,2,3,3,3,4,4,4,4,5,5,5,5,5]

##create a Histogram

plt.hist(data,bins=5,color='orange',edgecolor='black')
```

[16] ✓ 0.0s

```
(array([1., 2., 3., 4., 5.]),
 array([1. , 1.8, 2.6, 3.4, 4.2, 5. ]),
 <BarContainer object of 5 artists>)
```

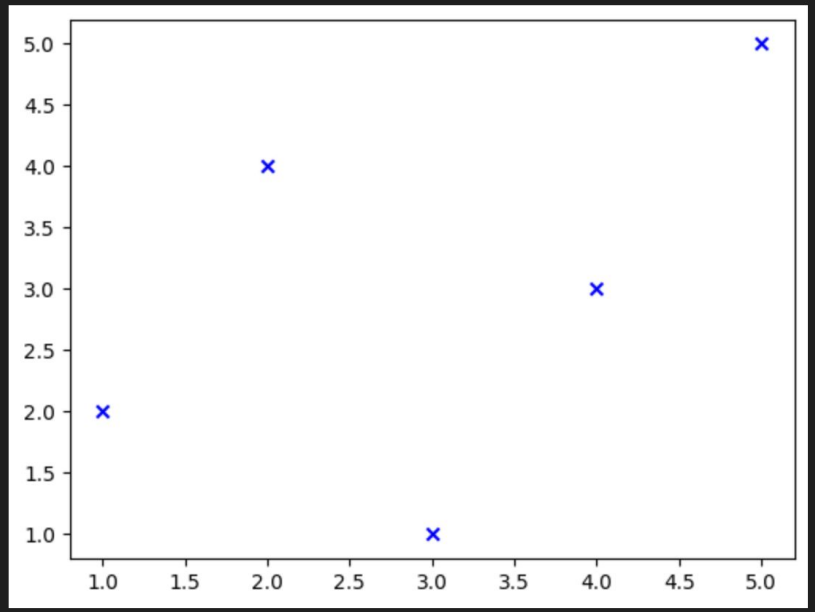


## Scatter Plot :

```
x = [1,2,3,4,5]
y = [2,4,1,3,5]

plt.scatter(x,y,color='blue',marker='x')
```

[17] ✓ 0.0s

```
<matplotlib.collections.PathCollection at 0x10ec68a40>
```



# Pie Chart:

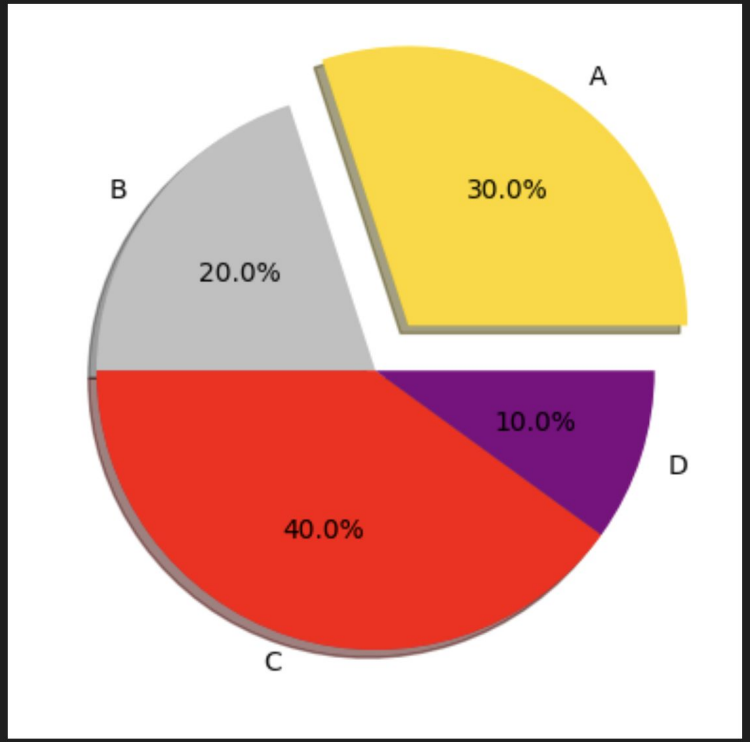
Pie Chart:

```
labels = ['A','B','C','D']
sizes = [30,20,40,10]
colors = ['gold','silver','red','purple']
explode = [0.2,0,0,0] ## move out the first slice

plt.pie(sizes,explode=explode,labels=labels,colors=colors,autopct="%1.1f%%",shadow='true')
```

[28] ✓ 0.0s Python

```
([<matplotlib.patches.Wedge at 0x10f189e80>,
  <matplotlib.patches.Wedge at 0x10f189dc0>,
  <matplotlib.patches.Wedge at 0x10f18adb0>,
  <matplotlib.patches.Wedge at 0x10f18b740>],
 [Text(0.7641207377093499, 1.0517221582730485, 'A'),
  Text(-0.8899187390319623, 0.6465637152823859, 'B'),
  Text(-0.3399188151951704, -1.046162128485022, 'C'),
  Text(1.0461621838648125, -0.339918644753721, 'D')],
 [Text(0.47022814628267684, 0.6472136358603374, '30.0%'),
  Text(-0.4854102212901612, 0.3526711174267559, '20.0%'),
  Text(-0.1854102628337293, -0.5706338882645574, '40.0%'),
  Text(0.5706339184717159, -0.185410169865666, '10.0%')])
```

# Pie Chart:



# Q. Example to read and visualize Sales Data :

```
import pandas as pd

df = pd.read_csv('Sales Data.csv')
```

[3] ✓ 0.2s

```
df.info()
```

[4] ✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 185950 entries, 0 to 185949
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Unnamed: 0        185950 non-null  int64
 1   Order ID          185950 non-null  int64
 2   Product           185950 non-null  object
 3   Quantity Ordered  185950 non-null  int64
 4   Price Each        185950 non-null  float64
 5   Order Date        185950 non-null  object
 6   Purchase Address  185950 non-null  object
 7   Month             185950 non-null  int64
 8   Sales             185950 non-null  float64
 9   City              185950 non-null  object
 10  Hour              185950 non-null  int64
dtypes: float64(2), int64(5), object(4)
```

```
total_sales = df.groupby('Product')['Sales'].sum()
print(total_sales)
```

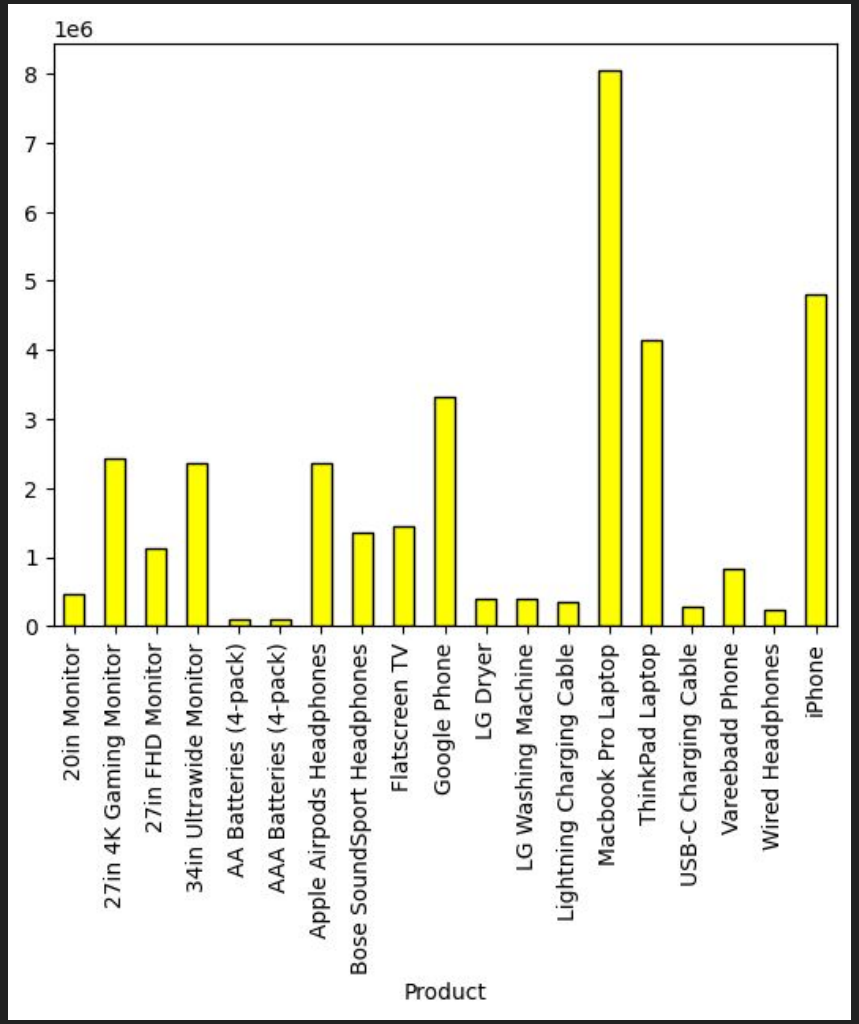
[36] ✓ 0.0s

```
Product
20in Monitor                454148.71
27in 4K Gaming Monitor     2435097.56
27in FHD Monitor           1132424.50
34in Ultrawide Monitor     2355558.01
AA Batteries (4-pack)       106118.40
AAA Batteries (4-pack)       92740.83
Apple Airpods Headphones   2349150.00
Bose SoundSport Headphones 1345565.43
Flatscreen TV              1445700.00
Google Phone               3319200.00
LG Dryer                    387600.00
LG Washing Machine          399600.00
Lightning Charging Cable    347094.15
Macbook Pro Laptop         8037600.00
ThinkPad Laptop            4129958.70
USB-C Charging Cable        286501.25
Vareebadd Phone             827200.00
Wired Headphones            246478.43
iPhone                     4794300.00
Name: Sales, dtype: float64
```

```
total_sales.plot(kind='bar',color='yellow',edgecolor='black')
```

[39] ✓ 0.0s

```
<Axes: xlabel='Product'>
```

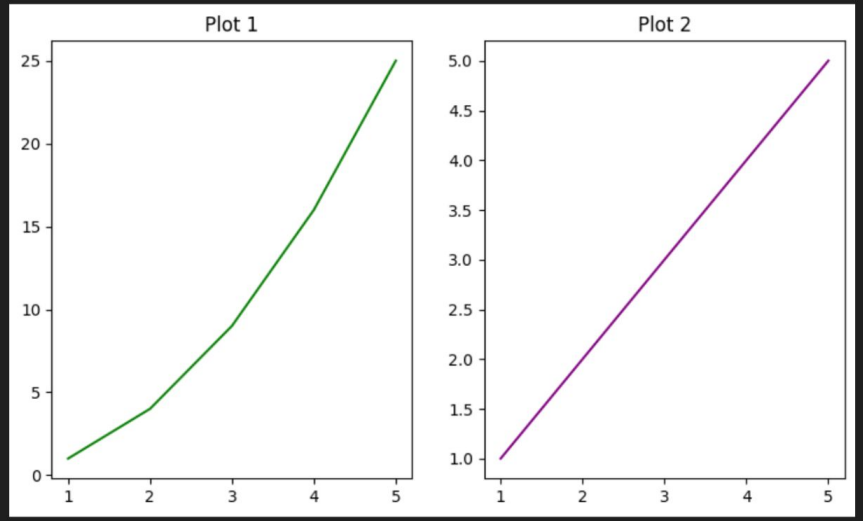

1e6
8
7
6
5
4
3
2
1
0
20in Monitor
27in 4K Gaming Monitor
27in FHD Monitor
34in Ultrawide Monitor
AA Batteries (4-pack)
AAA Batteries (4-pack)
Apple Airpods Headphones
Bose SoundSport Headphones
Flatscreen TV
Google Phone
LG Dryer
LG Washing Machine
Lightning Charging Cable
Macbook Pro Laptop
ThinkPad Laptop
USB-C Charging Cable
Vareebadd Phone
Wired Headphones
iPhone
Product

# Subplot :

```
x = [1,2,3,4,5]
y1 = [1,4,9,16,25]
y2 = [1,2,3,4,5]

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

plt.subplot(1,2,1)
plt.plot(x,y1,color='green')
plt.title("Plot 1")

plt.subplot(1,2,2)
plt.plot(x,y2,color='purple')
plt.title("Plot 2")
```

[48] ✓ 0.0s

```
Text(0.5, 1.0, 'Plot 2')
```



# SeaBorn

Seaborn is a powerful **Python data visualization library** built on top of Matplotlib. It provides a **high-level interface** for creating visually appealing and informative statistical graphics. Seaborn simplifies complex visualizations by integrating with Pandas and providing built-in themes, color palettes, and statistical functions.

## 🔹 Key Features of Seaborn

1. **Built-in themes and styles** → Makes graphs look attractive by default.
2. **Works seamlessly with Pandas** → Accepts Pandas DataFrames and Series as inputs.
3. **Advanced statistical plotting** → Includes visualizations like violin plots, box plots, and pair plots.
4. **Automatic handling of data relationships** → Makes it easier to plot relationships between multiple variables.
5. **Integration with Matplotlib** → Can be used alongside Matplotlib for customization.

# IMPORTING SEABORN

```
%pip install seaborn
```

[1] ✓ 1.9s Python

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

[6] ✓ 0.0s

```
## basic plotting with seaborn

tips = sns.load_dataset('tips')
tips
```

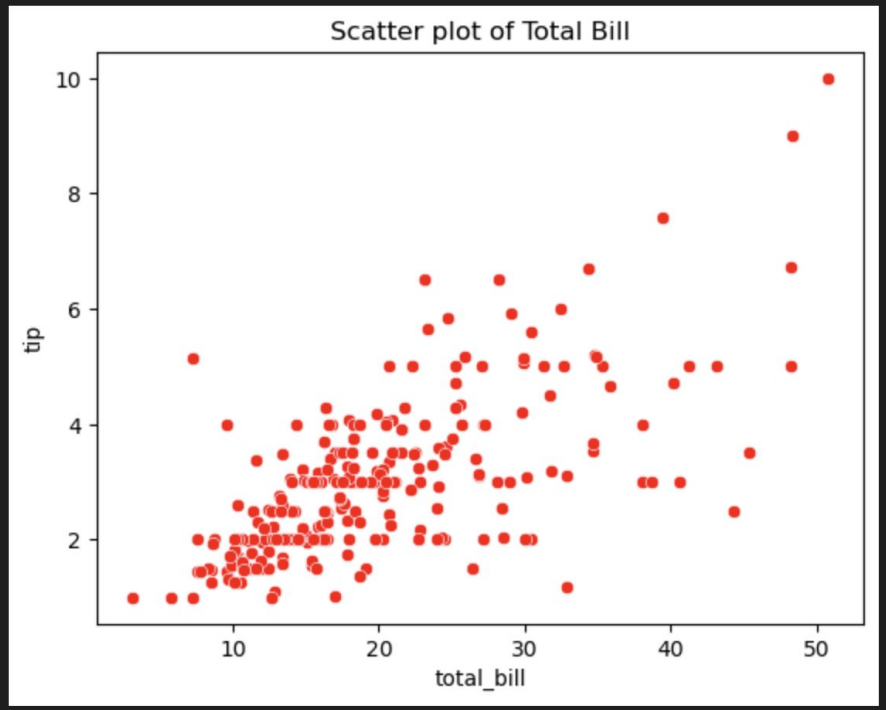
[4] ✓ 0.0s

| | total_bill | tip | sex | smoker | day | time | size |
|---|---|---|---|---|---|---|---|
| 0 | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| 1 | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| 2 | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| 3 | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| 4 | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 239 | 29.03 | 5.92 | Male | No | Sat | Dinner | 3 |
| 240 | 27.18 | 2.00 | Female | Yes | Sat | Dinner | 2 |
| 241 | 22.67 | 2.00 | Male | Yes | Sat | Dinner | 2 |
| 242 | 17.82 | 1.75 | Male | No | Sat | Dinner | 2 |
| 243 | 18.78 | 3.00 | Female | No | Thur | Dinner | 2 |

```
## creating a scatterplot

sns.scatterplot(x='total_bill',y='tip',data=tips,color='red')
plt.title("Scatter plot of Total Bill")
plt.show
```

[24] ✓ 0.0s

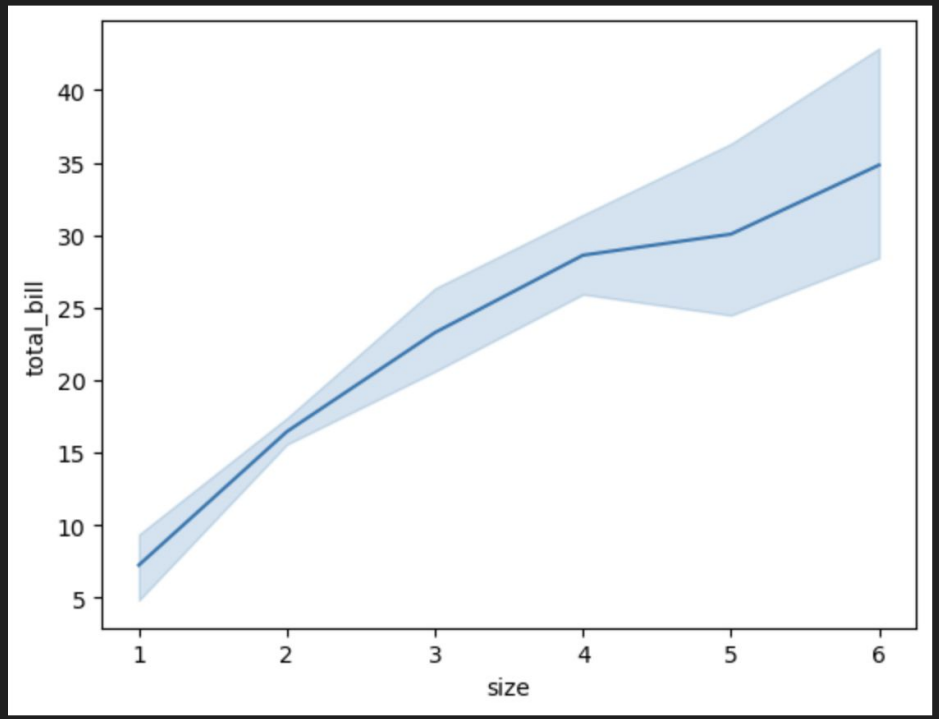
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



```
## LINE PLOT FOR THE SAME DATA

sns.lineplot(x='size',y='total_bill',data=tips)
```

[28] ✓ 0.1s

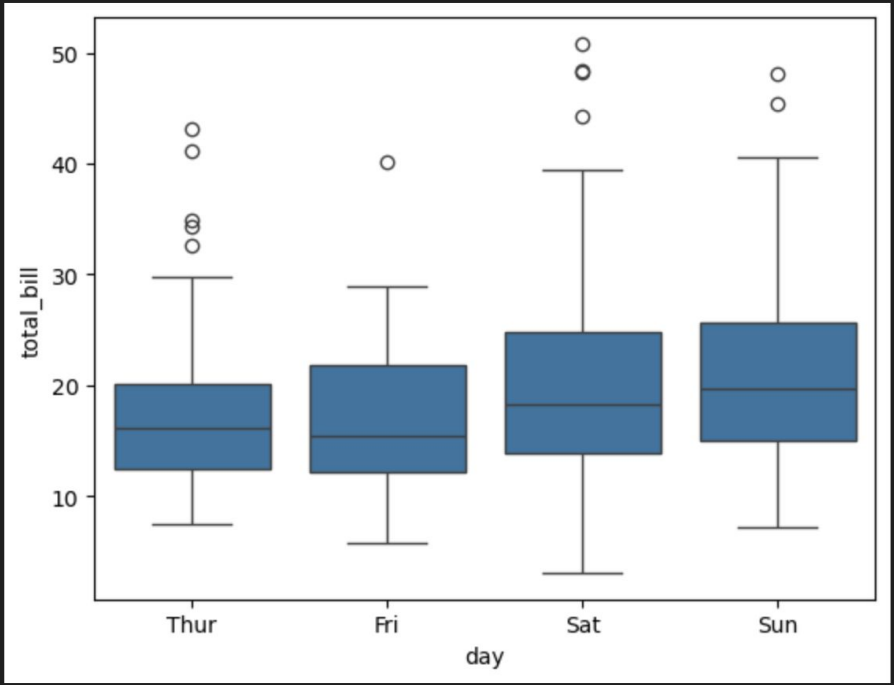
```
<Axes: xlabel='size', ylabel='total_bill'>
```



```
## CATEGORICAL PLOTS

## BOX PLOT

sns.boxplot(x='day',y='total_bill',data=tips)
```

[33] ✓ 0.0s

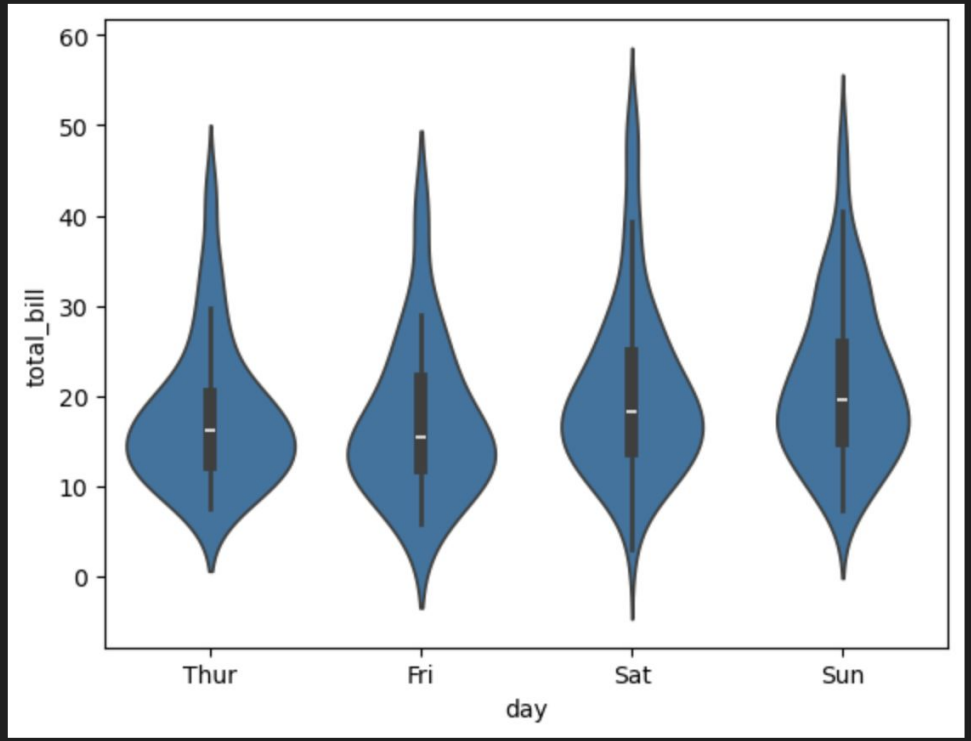
```
<Axes: xlabel='day', ylabel='total_bill'>
```



```
## VIOLIN PLOT

sns.violinplot(x='day',y='total_bill',data=tips)
```

[35] ✓ 0.1s

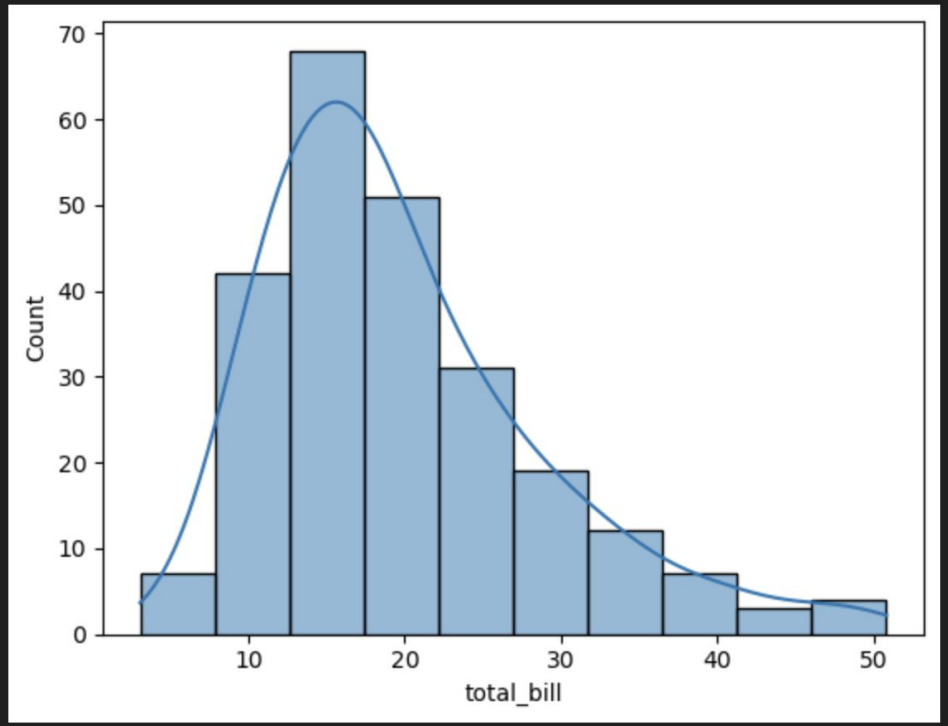
```
<Axes: xlabel='day', ylabel='total_bill'>
```



```
## HISTOGRAM

sns.histplot(tips['total_bill'],bins=10,kde=True)
```

✓ 0.2s

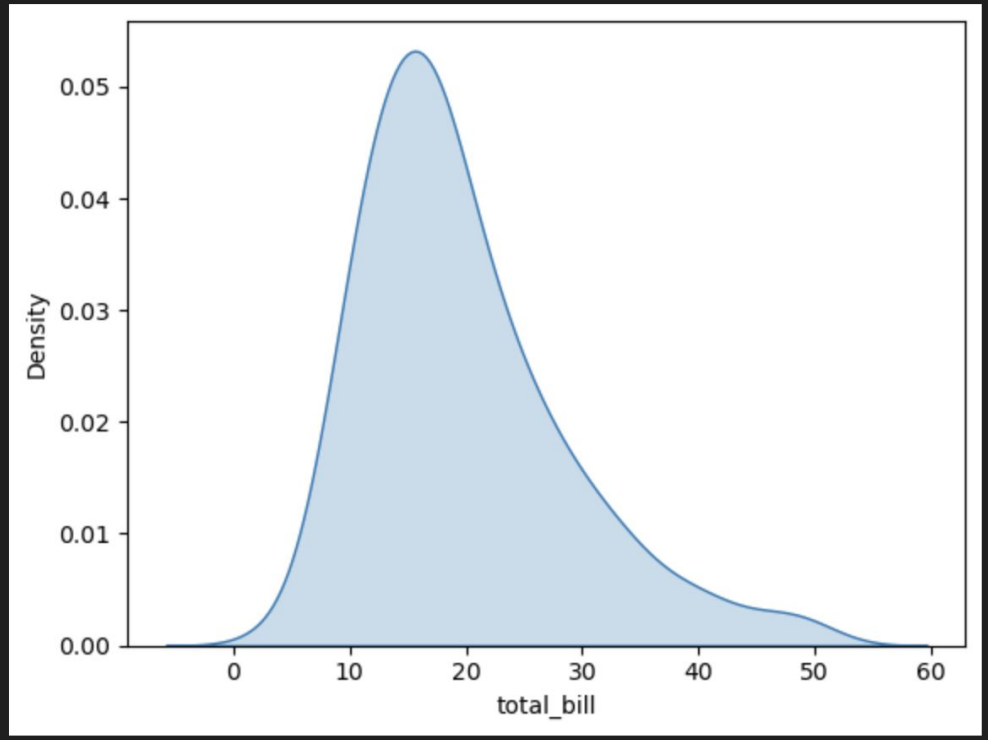
```
<Axes: xlabel='total_bill', ylabel='Count'>
```



```
## KDE PLOT

sns.kdeplot(tips['total_bill'],fill=True)
```

✓ 0.1s

```
<Axes: xlabel='total_bill', ylabel='Density'>
```



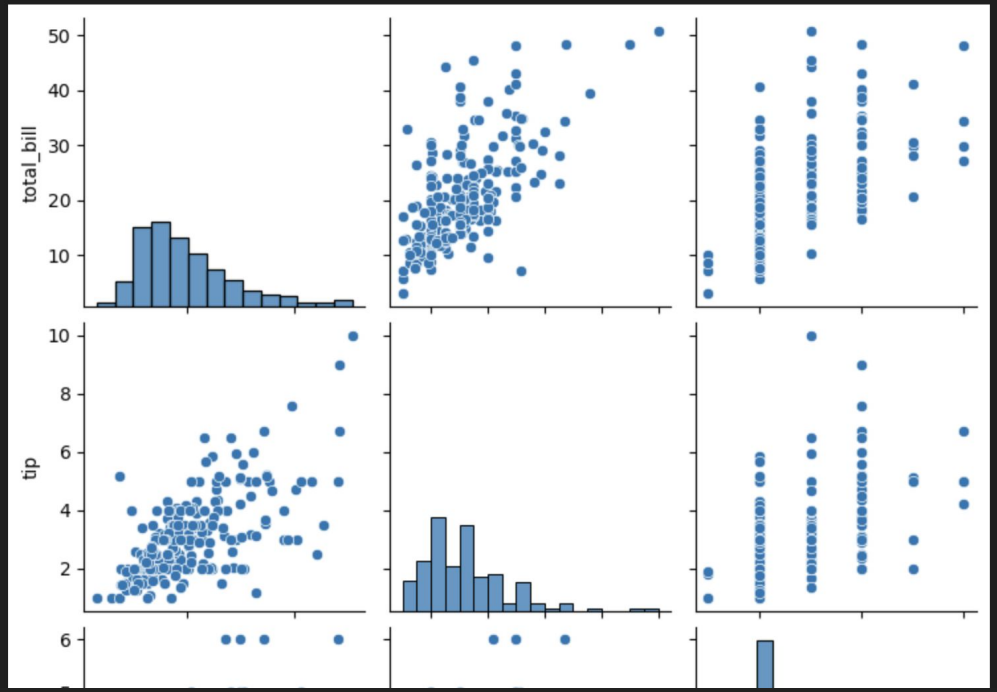
## PAIR PLOT -

Shows relation between each and every variable:

```
## PAIR PLOT

sns.pairplot(tips)
```

✓ 0.5s

```
<seaborn.axisgrid.PairGrid at 0x1408b40e0>
```



```
## HEAT MAP:
corr = tips[['total_bill','tip','size']].corr()
corr

## because data table conists of Gender variable which is string type
```

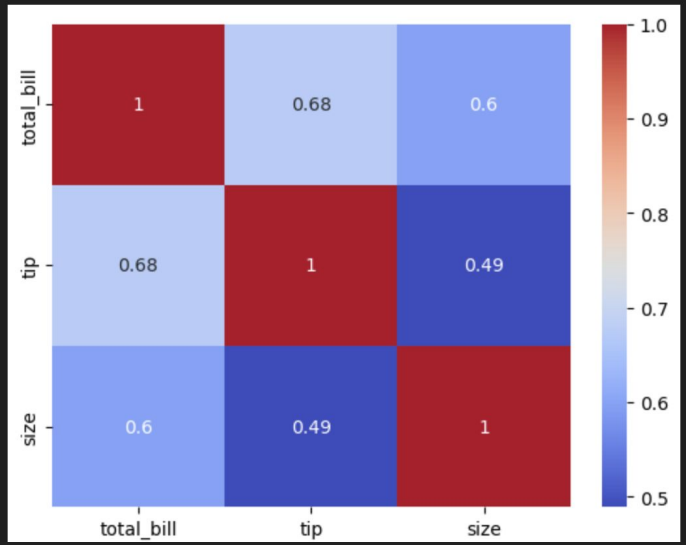
54] ✓ 0.0s

| | total_bill | tip | size |
|---|---|---|---|
| total_bill | 1.000000 | 0.675734 | 0.598315 |
| tip | 0.675734 | 1.000000 | 0.489299 |
| size | 0.598315 | 0.489299 | 1.000000 |

```
sns.heatmap(corr, cmap="coolwarm", annot=True)
```

53] ✓ 0.0s

<Axes: >



# Q. USING MATPLOTLIB TO READ DATA AND USING SEABORN TO PLOT BAR BLOT:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[72] ✓ 0.0s Python

```
df = pd.read_csv('Sales Data.csv')
df.head(5)
```

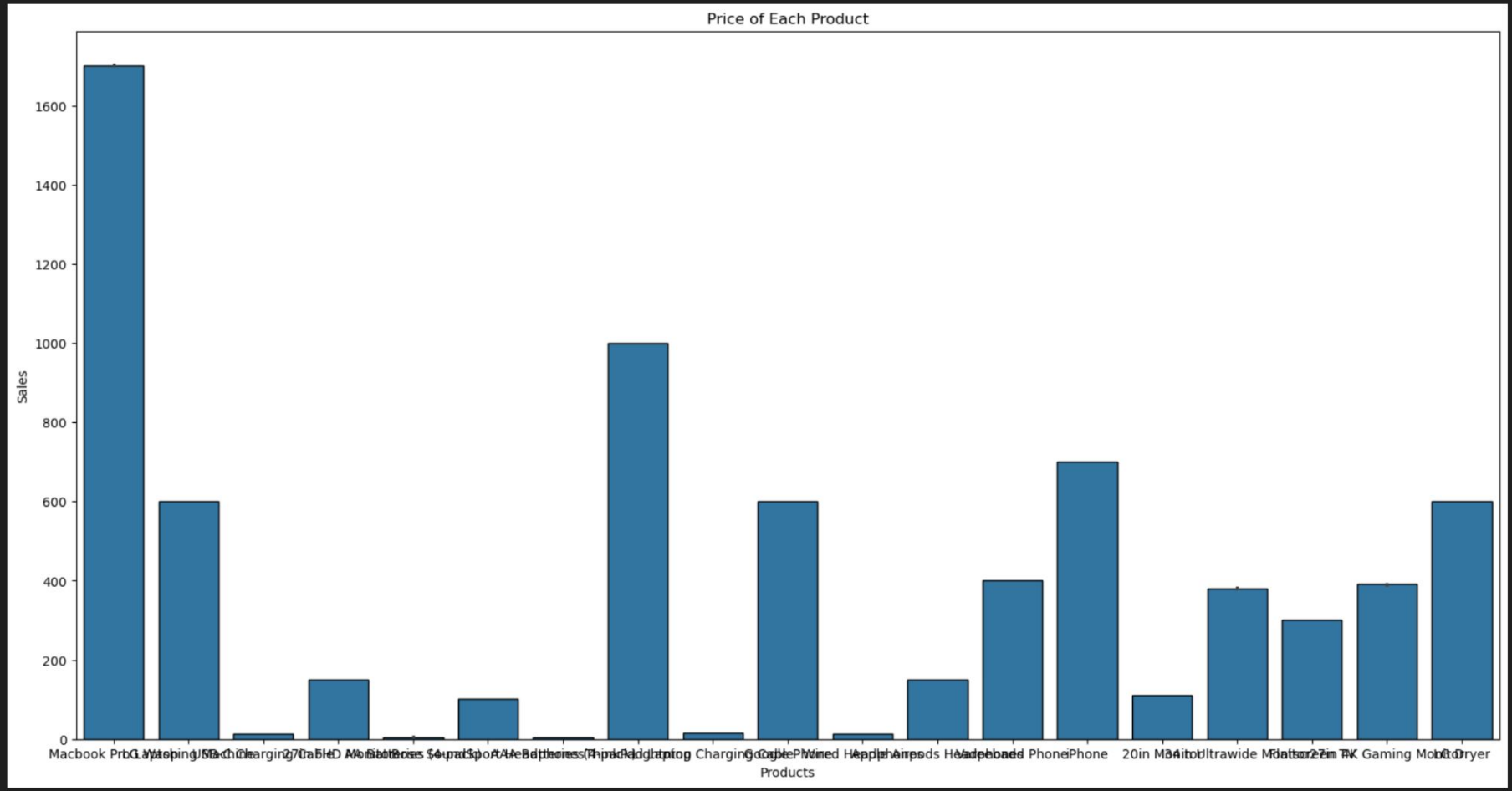
[61] ✓ 0.2s Python

| | Unnamed: 0 | Order ID | Product | Quantity Ordered | Price Each | Order Date | Purchase Address | Month | Sales | City | Hour |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 295665 | Macbook Pro Laptop | 1 | 1700.00 | 2019-12-30 00:01:00 | 136 Church St, New York City, NY 10001 | 12 | 1700.00 | New York City | 0 |
| 1 | 1 | 295666 | LG Washing Machine | 1 | 600.00 | 2019-12-29 07:03:00 | 562 2nd St, New York City, NY 10001 | 12 | 600.00 | New York City | 7 |
| 2 | 2 | 295667 | USB-C Charging Cable | 1 | 11.95 | 2019-12-12 18:21:00 | 277 Main St, New York City, NY 10001 | 12 | 11.95 | New York City | 18 |
| 3 | 3 | 295668 | 27in FHD Monitor | 1 | 149.99 | 2019-12-22 15:13:00 | 410 6th St, San Francisco, CA 94016 | 12 | 149.99 | San Francisco | 15 |
| 4 | 4 | 295669 | USB-C Charging Cable | 1 | 11.95 | 2019-12-18 12:38:00 | 43 Hill St, Atlanta, GA 30301 | 12 | 11.95 | Atlanta | 12 |

```
plt.figure(figsize=(20,10))
sns.barplot(data=df,x='Product',y='Sales',estimator=np.mean,edgecolor='black')
plt.title('Price of Each Product')
plt.xlabel('Products')
plt.ylabel('Sales')
```

[81] ✓ 1.3s Python

```
Text(0, 0.5, 'Sales')
```


Price of Each Product
Sales
0
200
400
600
800
1000
1200
1400
1600
Products
Macbook Pro Laptop
Lightning Charging Cable
USB-C Charging Cable
27in FHD Monitor
AAA Batteries (4-pack)
AA Batteries (4-pack)
Wired Headphones
Flatscreen TV
Bose SoundSport Headphones
iPhone Lightning Charging Cable
Google Phone
Wired Headphones
Apple Airpods Headphones
Vareebadd Phone
iPhone
20in Monitor
34in Ultrawide Monitor
27in 4K Gaming Monitor
Dryer