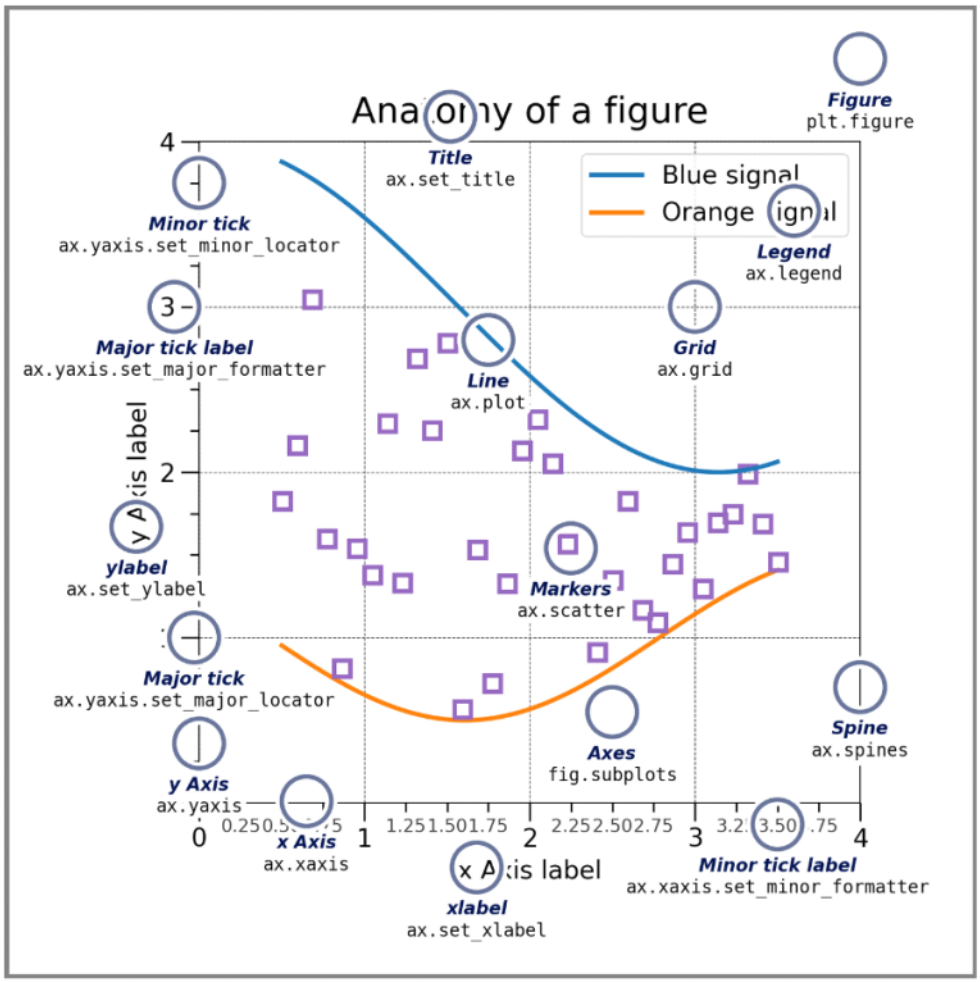
Matplotlib & Seaborn-2

🎯 Session Objectives:

- ✅ Understand what data visualization is and why it matters
- ✅ Use Matplotlib to plot different types of charts
- ✅ Customize plots with markers, colors, linewidth, and line styles
- ✅ Integrate Matplotlib with NumPy and Pandas
- ✅ Understand why Seaborn is important in visualization
- ✅ Recognize common Seaborn plot types



Bivariate Analysis

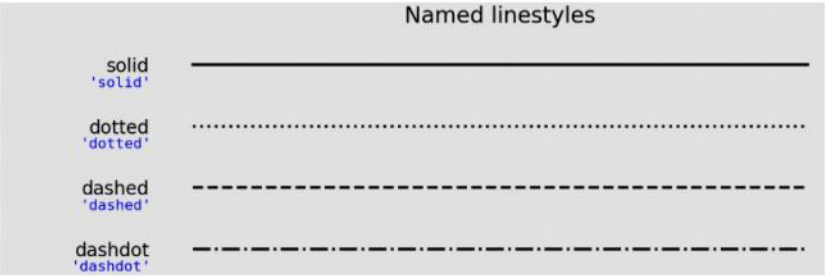
[Continuous VS Categorical]
[Continuous VS Continuous]
[Categorical Vs Categorical]

matplotlib.markers

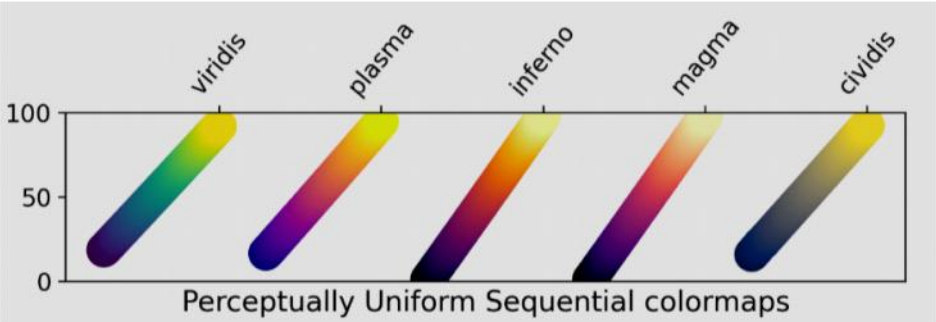
Functions to handle markers; used by the marker functionali

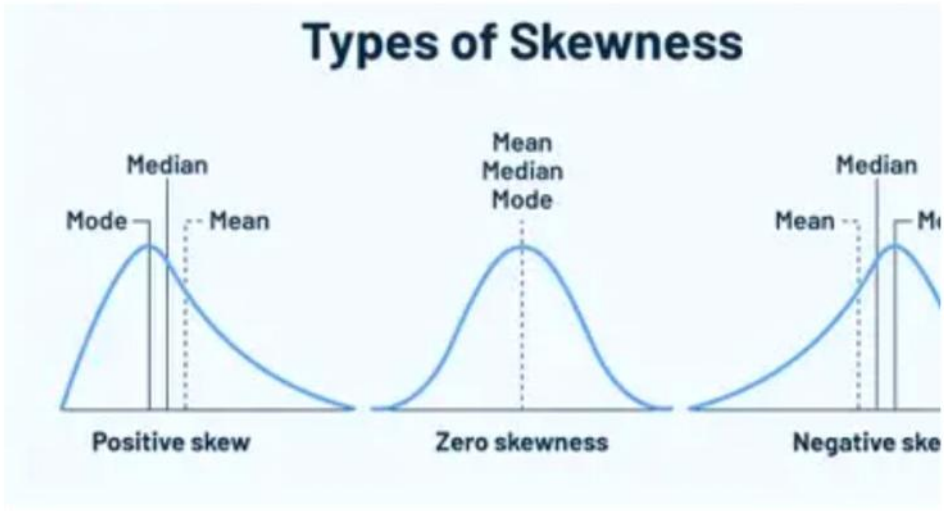
All possible markers are defined here:

marker	symbol	description
"."		point
","		pixel
"o"		circle
"v"		triangle_down
"^"		triangle_up
"<"		triangle_left
">"		triangle_right
"1"		tri_down



https://matplotlib.org/stable/users/explain/colors/colormaps.html



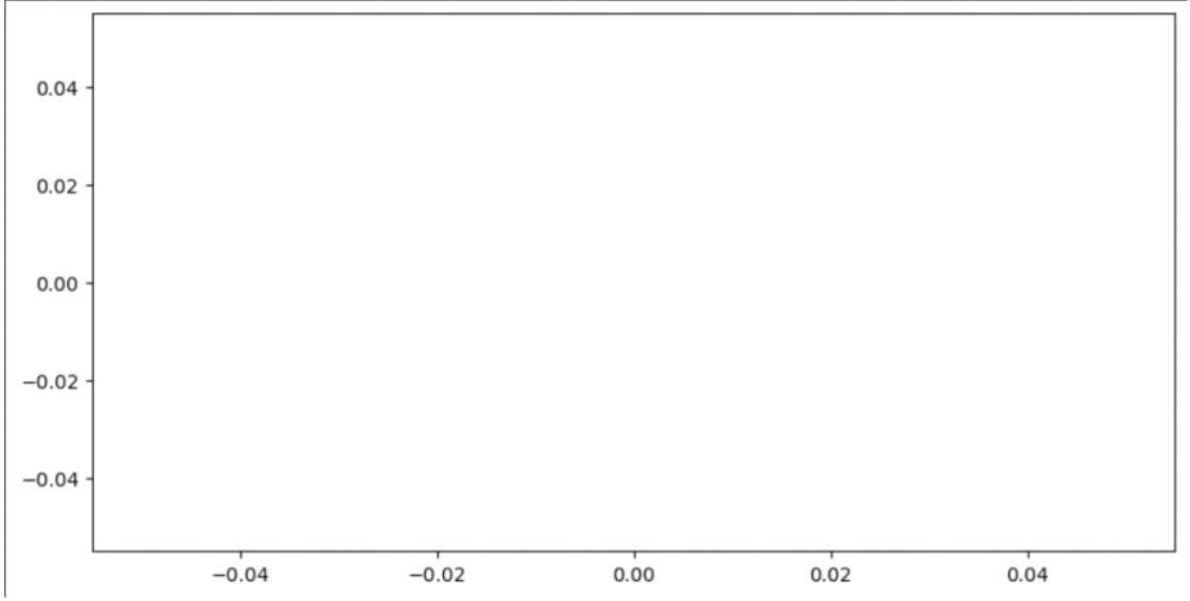


Note: Please complete all data cleaning steps before proceeding to the visualization stage.

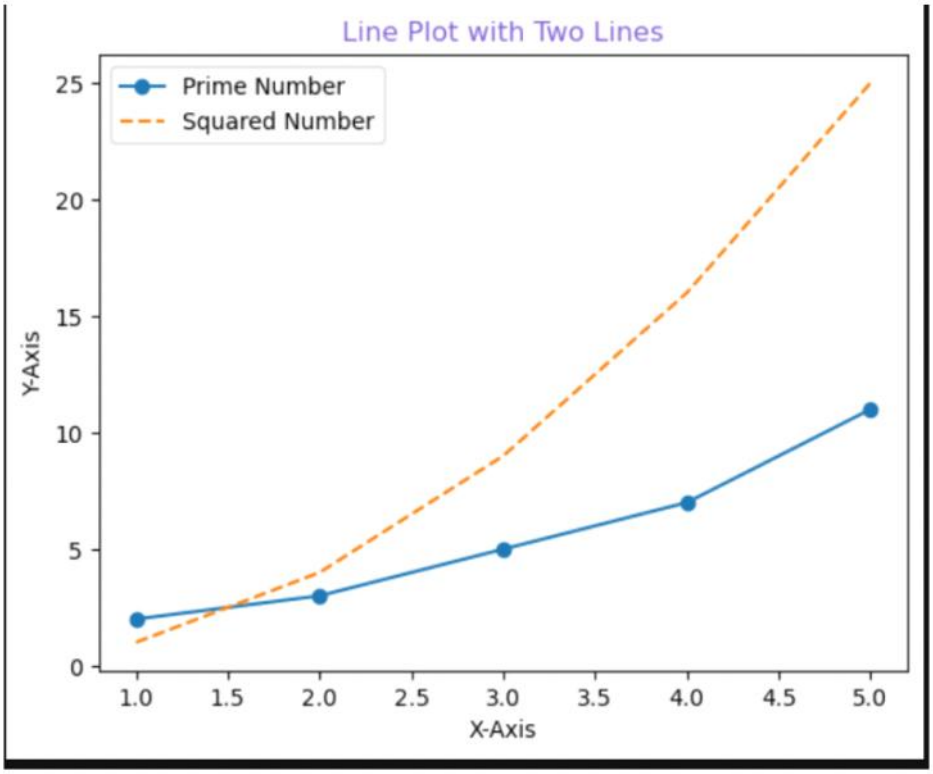
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```
plt.figure(figsize = (10,5)) # (width, height)
plt.plot()
```

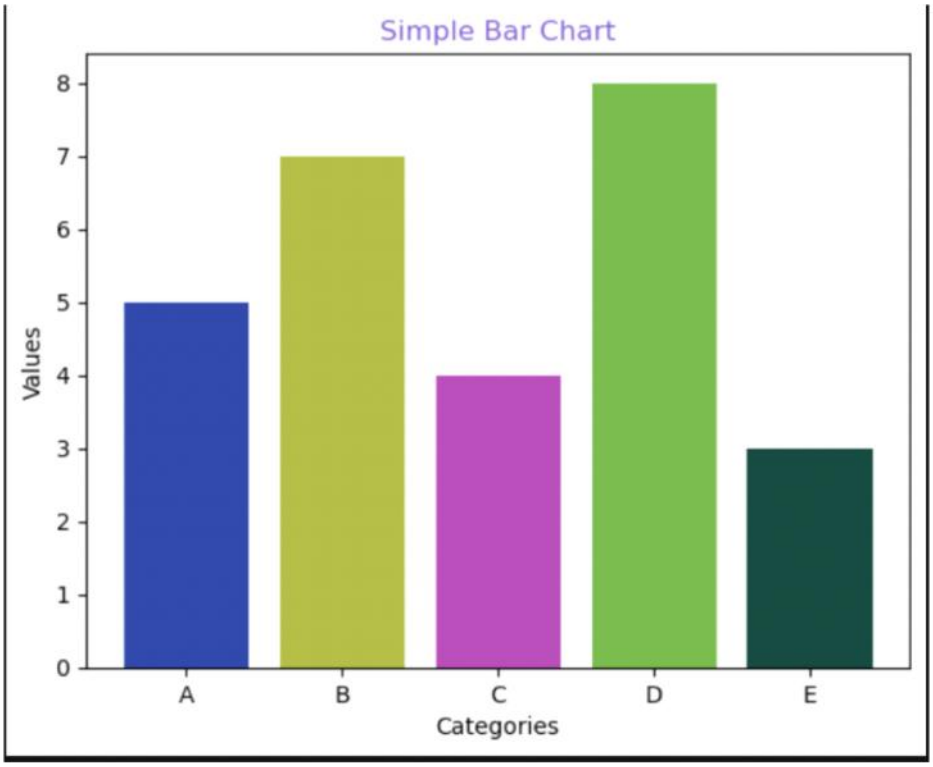
```
[]
```



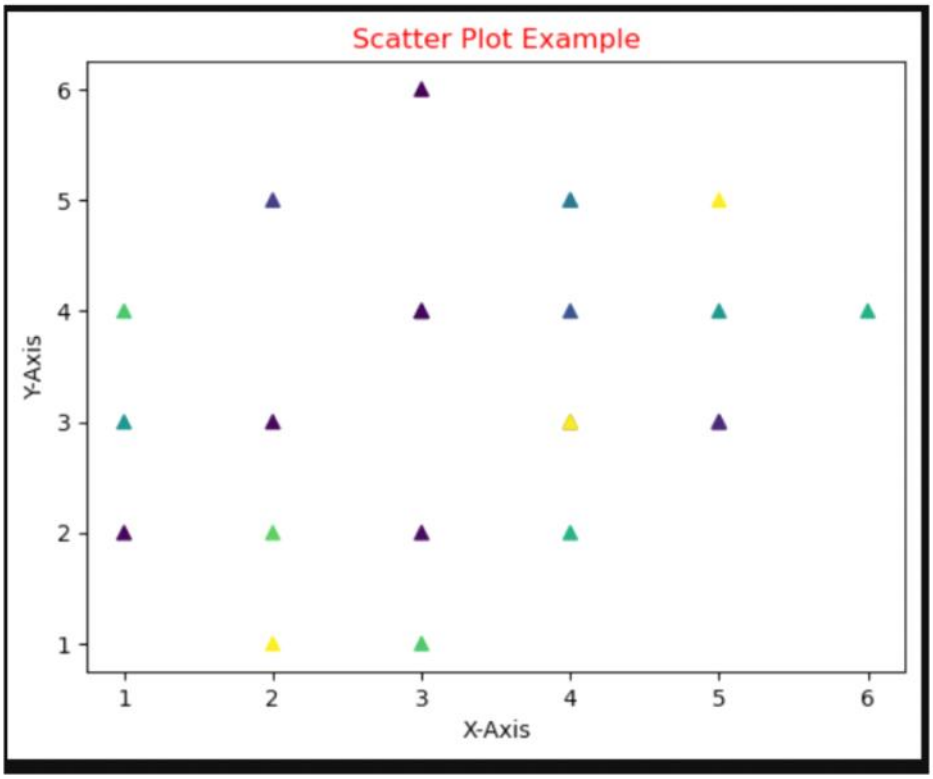
```python
# Line Plot with Multiple Lines
x = [1,2,3,4,5]
y1 = [2,3,5,7,11] # Prime Numbers
y2 = [1,4,9,16,25] # Squared Number

plt.plot(x , y1, label = "Prime Number" , marker = 'o')
plt.plot(x , y2, label = "Squared Number" , linestyle = '--')

plt.title("Line Plot with Two Lines" , color = '#7E5ED1')
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.legend()
plt.show()
```




```python
# Bar Chart [Categorical VS Continuous]
categories = ['A','B','C','D','E']
values = [5,7,4,8,3]
# colors = ['col1','col2',.....]
colors = ['#3147A8','#B0BA45','#B64DB8','#78B84D','#174A40']
plt.bar(categories, values, color = colors)

plt.title("Simple Bar Chart" , color = '#7E5ED1')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()
```



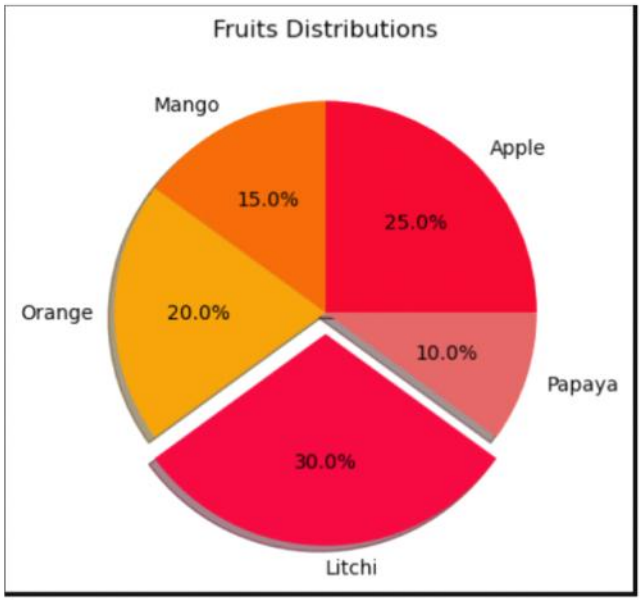

```
# Scatter Plot [Continuous VS Continuous]
x = [5,4,3,3,4,5,3,2,1,2,3,4,5,6,4,4,3,2,1,2,3,4,5,1]
y = [4,5,6,4,4,3,2,1,2,3,4,5,5,4,3,3,4,5,3,2,1,2,3,4]
colors = np.random.rand(24)
plt.scatter(x,y,c=colors,cmap = 'viridis', marker = '^')
plt.title("Scatter Plot Example" , color = 'red')
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.show()
```



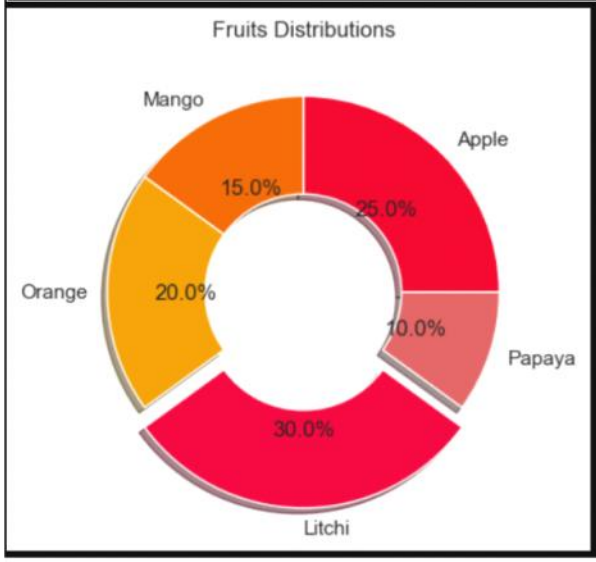
```
colors
```

```
array([0.50684192, 0.4916271 , 0.00892937, 0.4932071 , 0.25073179,
       0.04612673, 0.03462882, 0.97276215, 0.02366544, 0.0296304 ,
       0.10553535, 0.38785453, 0.95942208, 0.59700418, 0.05799306,
       0.95432777, 0.02186043, 0.1820911 , 0.50011489, 0.73553034,
       0.70239074, 0.61419957, 0.11905454, 0.69718322])
```

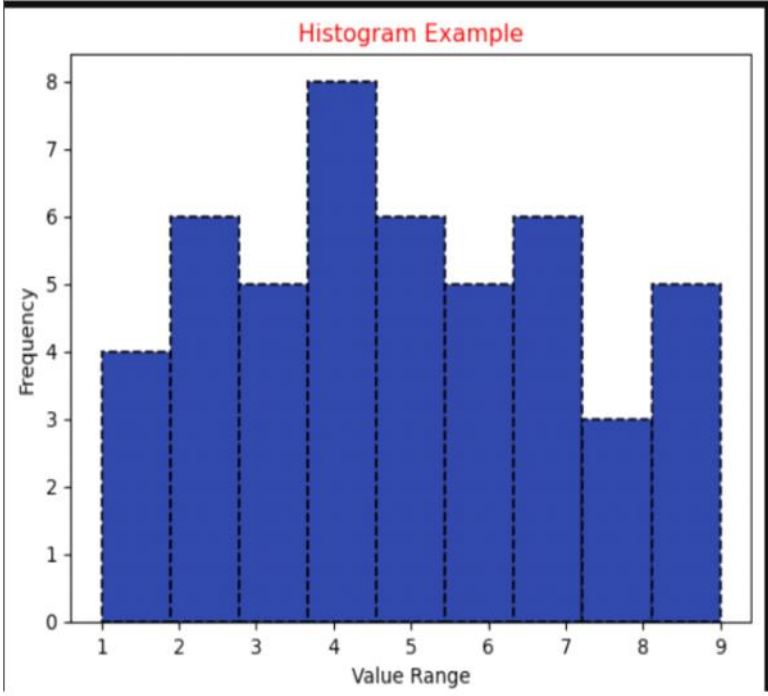
```
# Pie Chart [Composition Visuals]
sizes = [25,15,20,30,10]
labels = ['Apple','Mango','Orange','Litchi','Papaya']
colors = ['#F00A30', '#F06A0A', '#F0A00A', '#F00A40', '#E06565']
plt.pie(sizes, labels = labels, colors = colors , explode = [0,0,0,0.1,0], autopct = '%1.1f%%',shadow=True)
plt.title("Fruits Distributions")
plt.show()
```



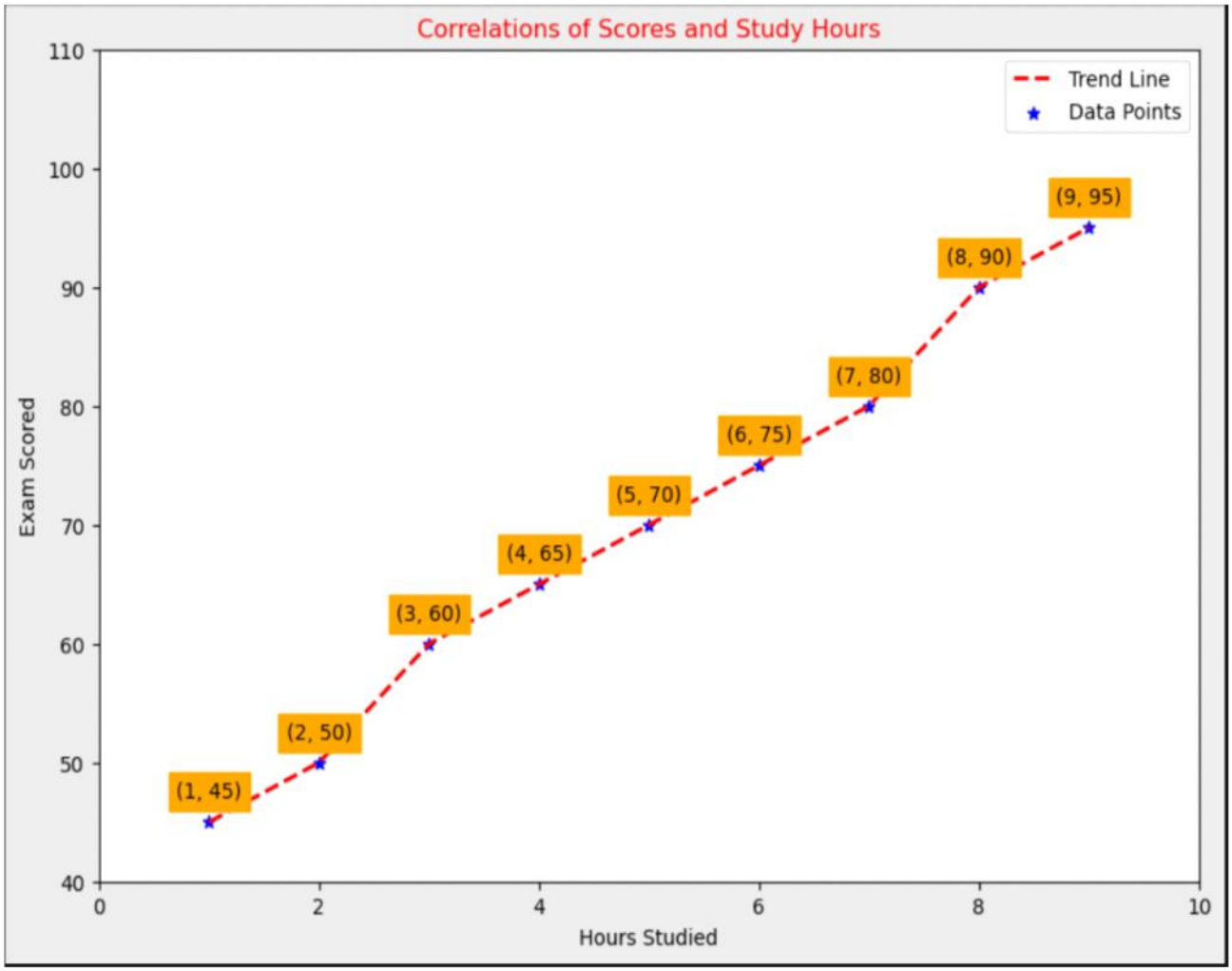

```
# Pie Chart [Composition Visuals]
sizes = [25,15,20,30,10]
labels = ['Apple','Mango','Orange','Litchi','Papaya']
colors = ['#F00A30', '#F06A0A', '#F0A00A', '#F00A40', '#E06565']
plt.pie(sizes, labels = labels, colors = colors , explode = [0,0,0,0.1,0], autopct = '%1.1f%%',shadow=True,
        wedgeprops=dict(width=0.5))
plt.title("Fruits Distributions")
plt.show()
```



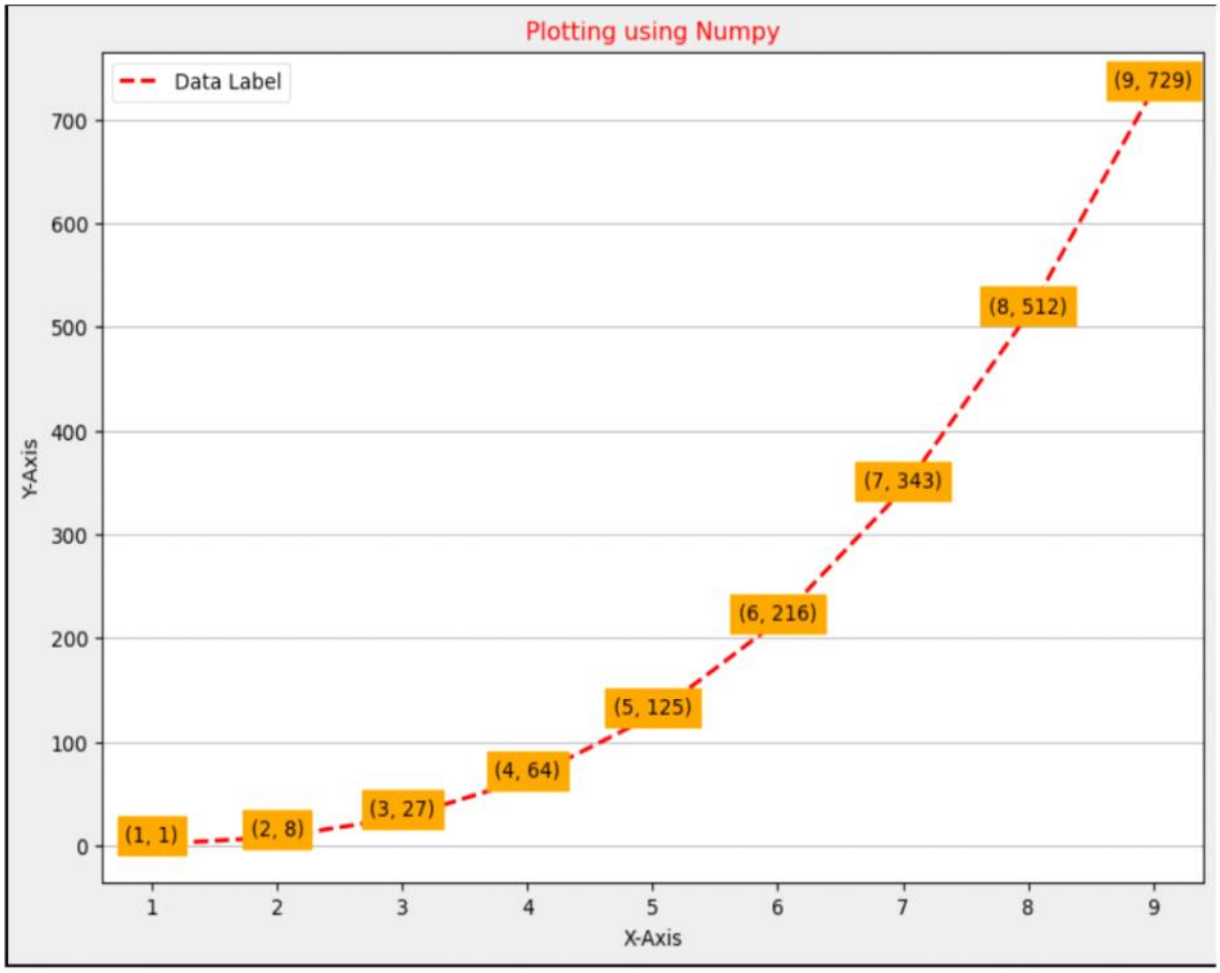
```
# Histogram
data = [1,2,2,4,4,5,6,5,4,3,4,5,6,7,8,7,5,7,8,9,9,9,9,1,2,3,4,3,2,4,5,6,7,8,9,7,6,4,3,3,2,1,1,2,4,5,6,7]
plt.hist(data, bins=9, color = '#3147A8' , edgecolor = 'black', linewidth = 1.2 , linestyle = '--')
plt.title("Histogram Example" , color = 'red')
plt.xlabel('Value Range')
plt.ylabel('Frequency')
plt.show()
```




```
# Continuous Vs Continuous
hours = [1,2,3,4,5,6,7,8,9]
scores = [45,50,60,65,70,75,80,90,95]
plt.figure(figsize = (10,7) , facecolor = '#E8E8E8')

# Line + Scatter
plt.plot(hours , scores , color = 'red', linestyle = '--', linewidth = 2, label = 'Trend Line')
plt.scatter(hours, scores, color = 'blue' , marker = '*', label = 'Data Points')

# Add a Data Label
for x,y in zip(hours,scores):
    plt.text(x,y+2,f"{x,y}", ha = 'center', fontsize=10, color = 'black', backgroundcolor = 'orange')

plt.title('Correlations of Scores and Study Hours' , color = 'red')
plt.xlabel('Hours Studied')
plt.ylabel('Exam Scored')
plt.ylim(40,110)
plt.xlim(0,10)
plt.legend()
plt.show()
```




```python
# Continuous Vs Continuous
import numpy as np
x = np.array([1,2,3,4,5,6,7,8,9])
y = x ** 3 # [1,8,27,64.........]
plt.figure(figsize = (10,7) , facecolor = '#E8E8E8')

# Line + Scatter
plt.plot(x , y , color = 'red', linestyle = '--', linewidth = 2, label = 'Data Label')

# Add a Data Label
for x,y in zip(x,y):
    plt.text(x,y+2,f"{x,y}", ha = 'center', fontsize=10, color = 'black', backgroundcolor = 'orange')

plt.title('Plotting using Numpy' , color = 'red')
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.grid(axis = 'y')
plt.legend()
plt.show()
```




```
# Customers Distribution ['genders']
gender_counts = customers['gender'].value_counts() #series
gender_counts
```

```
gender
Male      528
Female    472
Name: count, dtype: int64
```

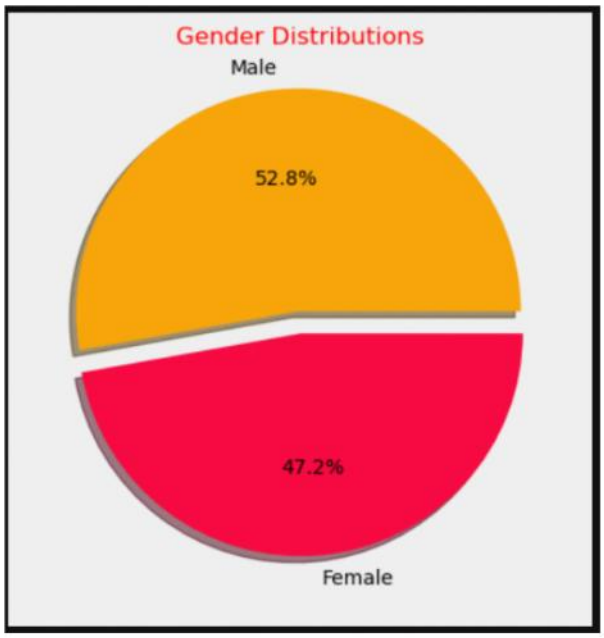
```
gender_counts.index
```

```
Index(['Male', 'Female'], dtype='object', name='gender')
```

```
gender_counts.values
```

```
array([528, 472], dtype=int64)
```

```
# Pie Chart [Composition Visuals]
plt.figure(figsize = (5,7) , facecolor = '#E8E8E8')
plt.pie(gender_counts, labels = gender_counts.index, colors = ['#F0A00A', '#F00A40'],
        explode = (0.1,0) , autopct = '%1.1f%%', shadow=True)
plt.title("Gender Distributions", color = 'red')
plt.show()
```




```
# Top 10 States based on count of customers.
top_states = customers['state'].value_counts().nlargest(10)
top_states # [Bar Chart]
```

```
state
Texas                   118
California              108
Florida                  85
New York                 48
Ohio                     45
Pennsylvania             36
District of Columbia     35
Georgia                  29
Tennessee                27
Virginia                 27
Name: count, dtype: int64
```

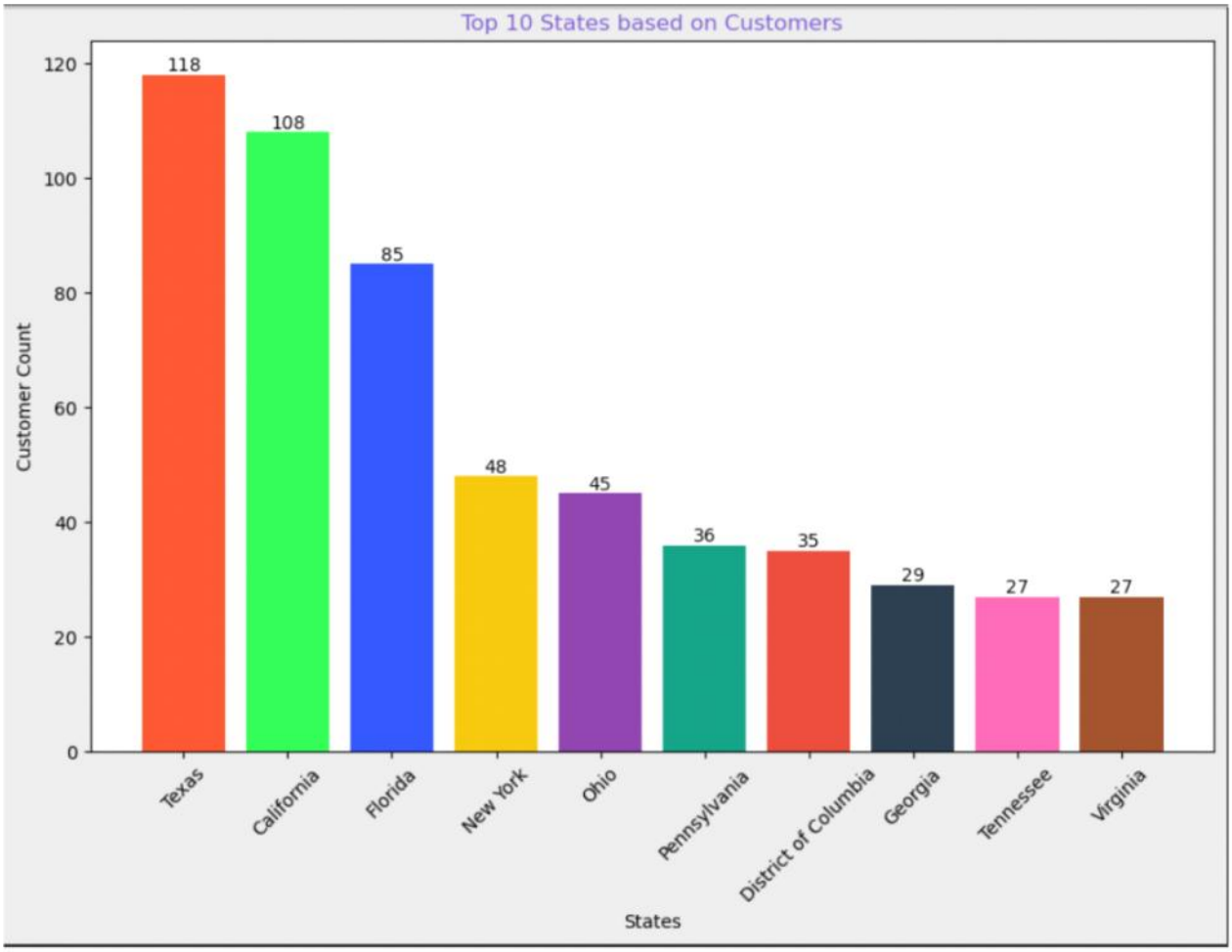
```
top_states.index
```

```
Index(['Texas', 'California', 'Florida', 'New York', 'Ohio', 'Pennsylvania',
       'District of Columbia', 'Georgia', 'Tennessee', 'Virginia'],
      dtype='object', name='state')
```

```
top_states.values
```

```
array([118, 108,  85,  48,  45,  36,  35,  29,  27,  27], dtype=int64)
```

```
# Plotting the Bar Plot
plt.figure(figsize = (11,7) , facecolor = '#E8E8E8')
colors = ["#FF5733", "#33FF57", "#3357FF", "#F1C40F", "#8E44AD",
          "#16A085", "#E74C3C", "#2C3E50", "#FF69B4", "#A0522D" ]
bar_state = plt.bar(top_states.index , top_states.values ,  color = colors)
plt.title("Top 10 States based on Customers" , color = '#7E5ED1')
plt.xlabel('States')
plt.ylabel('Customer Count')
plt.xticks(rotation=45)
plt.bar_label(bar_state)
plt.show()
```




```
# Calculated Columns [DAX] (row context , filtering)
# Add Column [Power Query Editor] [Date column -> Extract [Year,month,days]]
purchases['year'] = purchases['purch_date'].dt.year
purchases['month'] = purchases['purch_date'].dt.month
purchases['item_price'] = purchases['paid'] / purchases['amount']
purchases
```

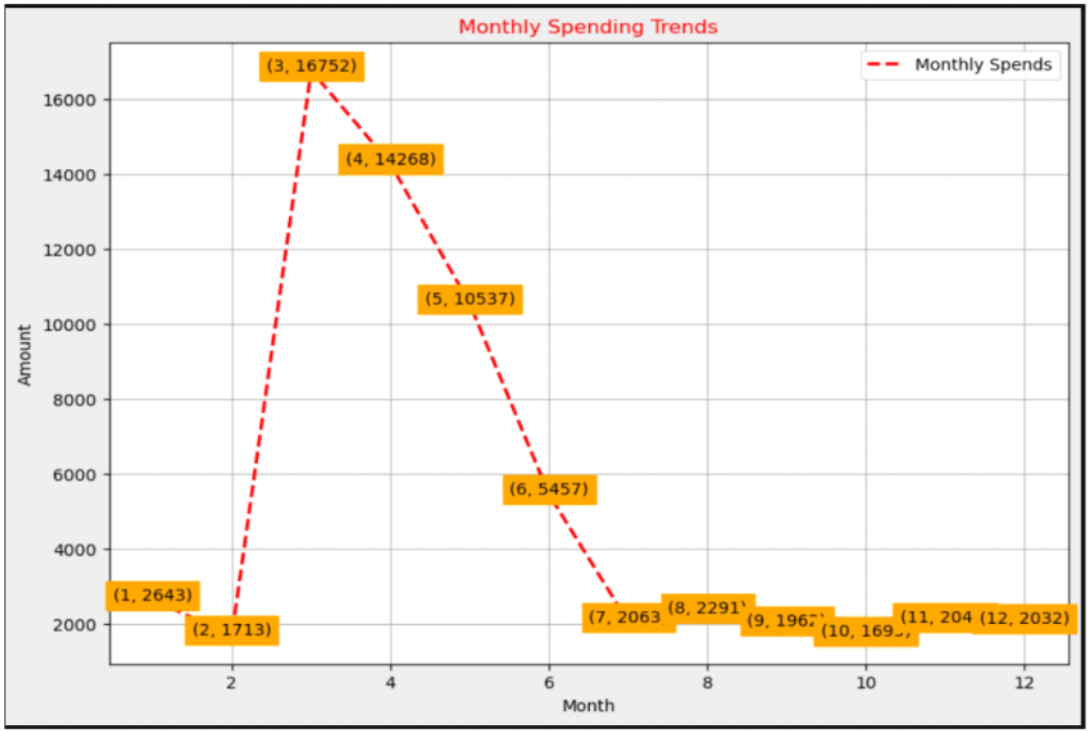
	id	purch_date	customer_num	product_num	amount	paid	year	month	item_price
0	1	2019-01-03	823	27	12	568.92	2019	1	47.41
1	2	2019-01-03	606	28	14	395.36	2019	1	28.24
2	3	2019-01-03	955	9	17	510.17	2019	1	30.01
3	4	2019-01-03	577	19	3	68.49	2019	1	22.83
4	5	2019-01-03	429	8	18	759.42	2019	1	42.19
...	...	...	...	...	...	...	...	...	...
5995	5996	2019-06-20	893	33	5	411.10	2019	6	82.22
5996	5997	2019-06-20	566	23	11	178.97	2019	6	16.27
5997	5998	2019-06-20	114	19	9	205.47	2019	6	22.83
5998	5999	2019-06-20	404	11	20	429.40	2019	6	21.47
5999	6000	2019-06-20	88	57	4	274.52	2019	6	68.63

6000 rows × 9 columns

```python
# Trend Axis [Time Intelligence] Over a Period of Time [Line Plot]
monthly_spendings = purchases.groupby('month')['amount'].sum().reset_index()
monthly_spendings
```

	month	amount
0	1	2643
1	2	1713
2	3	16752
3	4	14268
4	5	10537
5	6	5457
6	7	2063
7	8	2291
8	9	1962
9	10	1695
10	11	2044
11	12	2032

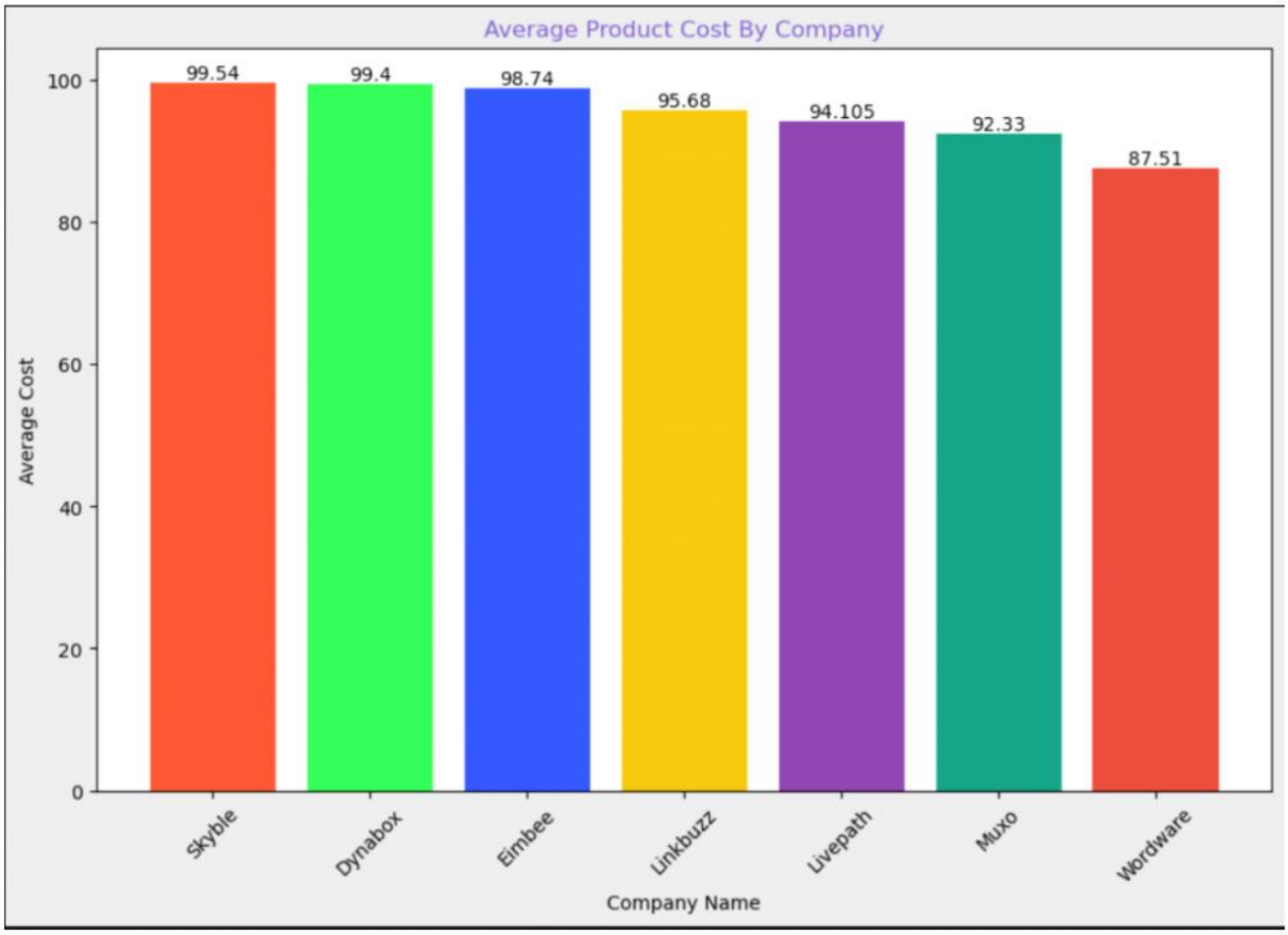
```python
plt.figure(figsize = (10,7) , facecolor = '#E8E8E8')

# Line
plt.plot(monthly_spendings['month'] , monthly_spendings['amount'] ,
         color = 'red', linestyle = '--', linewidth = 2, label = 'Monthly Spends')

# Add a Data Label
for x,y in zip(monthly_spendings['month'] , monthly_spendings['amount']):
    plt.text(x,y+2,f"{x,y}", ha = 'center', fontsize=10, color = 'black', backgroundcolor = 'orange')

plt.title('Monthly Spending Trends' , color = 'red')
plt.xlabel('Month')
plt.ylabel('Amount')
plt.grid()
plt.legend()
plt.show()
```



```python
# Average Product cost by company
avg_product_by_comp = products.groupby('company')['cost'].mean().nlargest(7)
avg_product_by_comp
```

```
company
Skyble      99.540
Dynabox     99.400
Eimbee      98.740
Linkbuzz    95.680
Livepath    94.105
Muxo        92.330
Wordware    87.510
Name: cost, dtype: float64
```

```python
# Plotting the Bar Plot
plt.figure(figsize = (11,7) , facecolor = '#E8E8E8')
colors = ["#FF5733", "#33FF57", "#3357FF", "#F1C40F", "#8E44AD","#16A085", "#E74C3C" ]
company_bar = plt.bar(avg_product_by_comp.index , avg_product_by_comp.values ,  color = colors)
plt.title("Average Product Cost By Company" , color = '#7E5ED1')
plt.xlabel('Company Name')
plt.ylabel('Average Cost')
plt.xticks(rotation=45)
plt.bar_label(company_bar)
# Save the figure
plt.savefig('avg_cost_by_company.png')
plt.show()
```



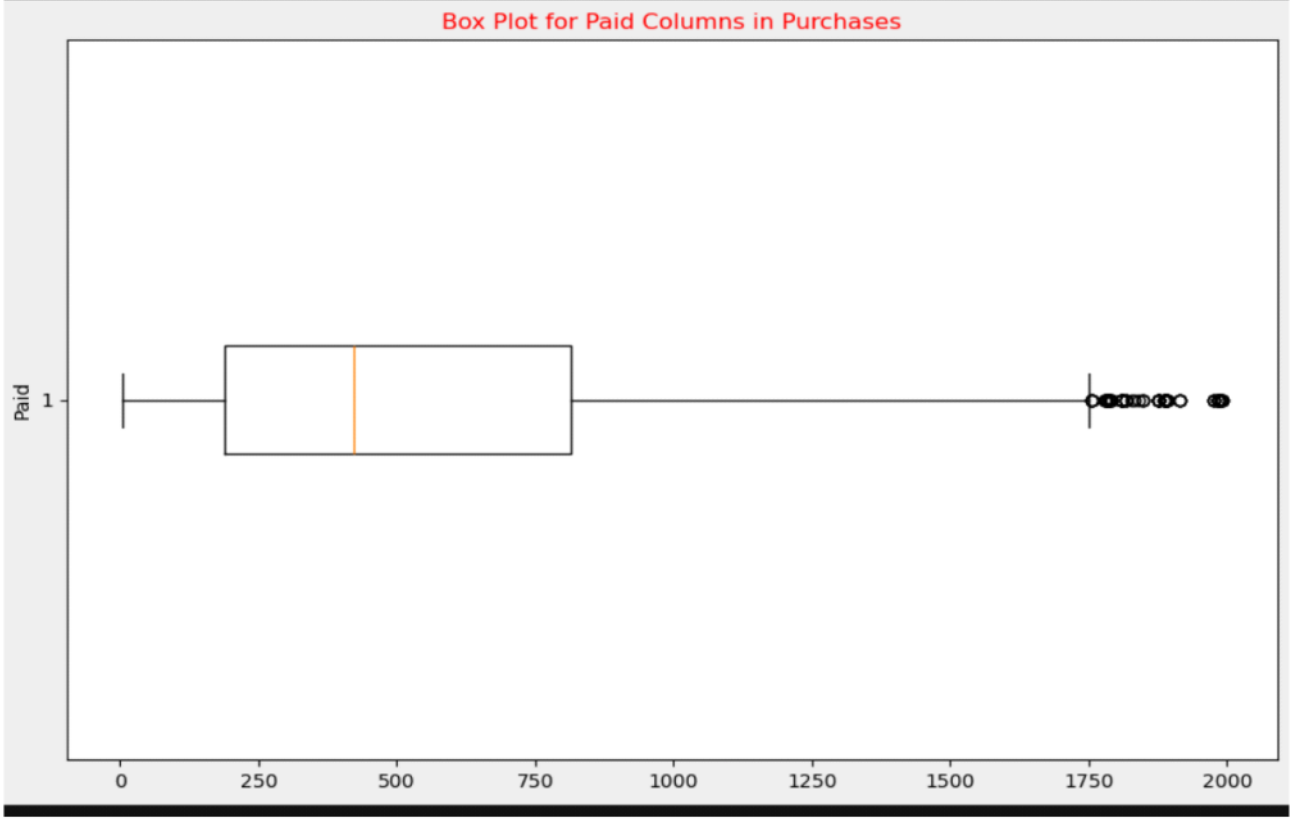
```python
# Univariate Analysis ['Single Column'] [Continuous Columns]
# Outlier Detection -> [Box-Wisker Plot]
purchases['paid'].describe() # +ve Skewness
```

```
count    6000.000000
mean      548.871353
std       444.237192
min         3.630000
25%       187.560000
50%       422.700000
75%       813.800000
max      1990.800000
Name: paid, dtype: float64
```

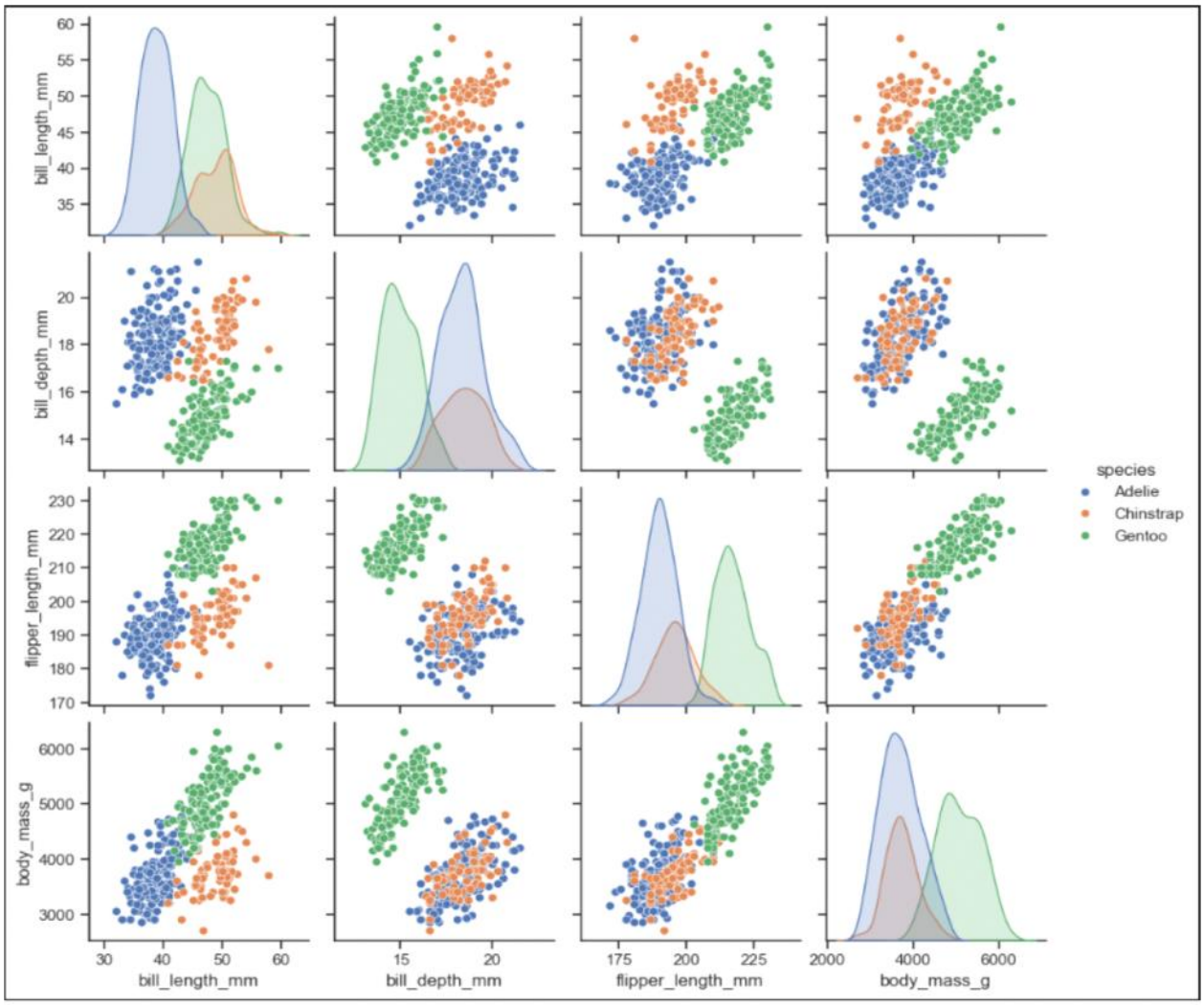
```python
purchases['paid']
```

```
0       568.92
1       395.36
2       510.17
3        68.49
4       759.42
         ...
5995    411.10
5996    178.97
5997    205.47
5998    429.40
5999    274.52
Name: paid, Length: 6000, dtype: float64
```

```python
# Plotting the Box Plot
plt.figure(figsize = (11,7) , facecolor = '#E8E8E8')
plt.boxplot(purchases['paid'] , vert = False)
plt.title('Box Plot for Paid Columns in Purchases', color ='red')
plt.ylabel('Paid')
plt.show()
```



```
import seaborn as sns
sns.set_theme(style="ticks")

df = sns.load_dataset("penguins")
sns.pairplot(df, hue="species")
```



df

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

344 rows × 7 columns

```
df['species'].unique()
```

```
array(['Adelie', 'Chinstrap', 'Gentoo'], dtype=object)
```

```
df['island'].unique()
```

```
array(['Torgersen', 'Biscoe', 'Dream'], dtype=object)
```

```
df['sex'].unique()
```

```
array(['Male', 'Female', nan], dtype=object)
```

```
import seaborn as sns
sns.get_dataset_names()
```

```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

```
import seaborn as sns
sns.set_theme(style="ticks")

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

	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

244 rows × 7 columns

```
tips['time'].unique()
```

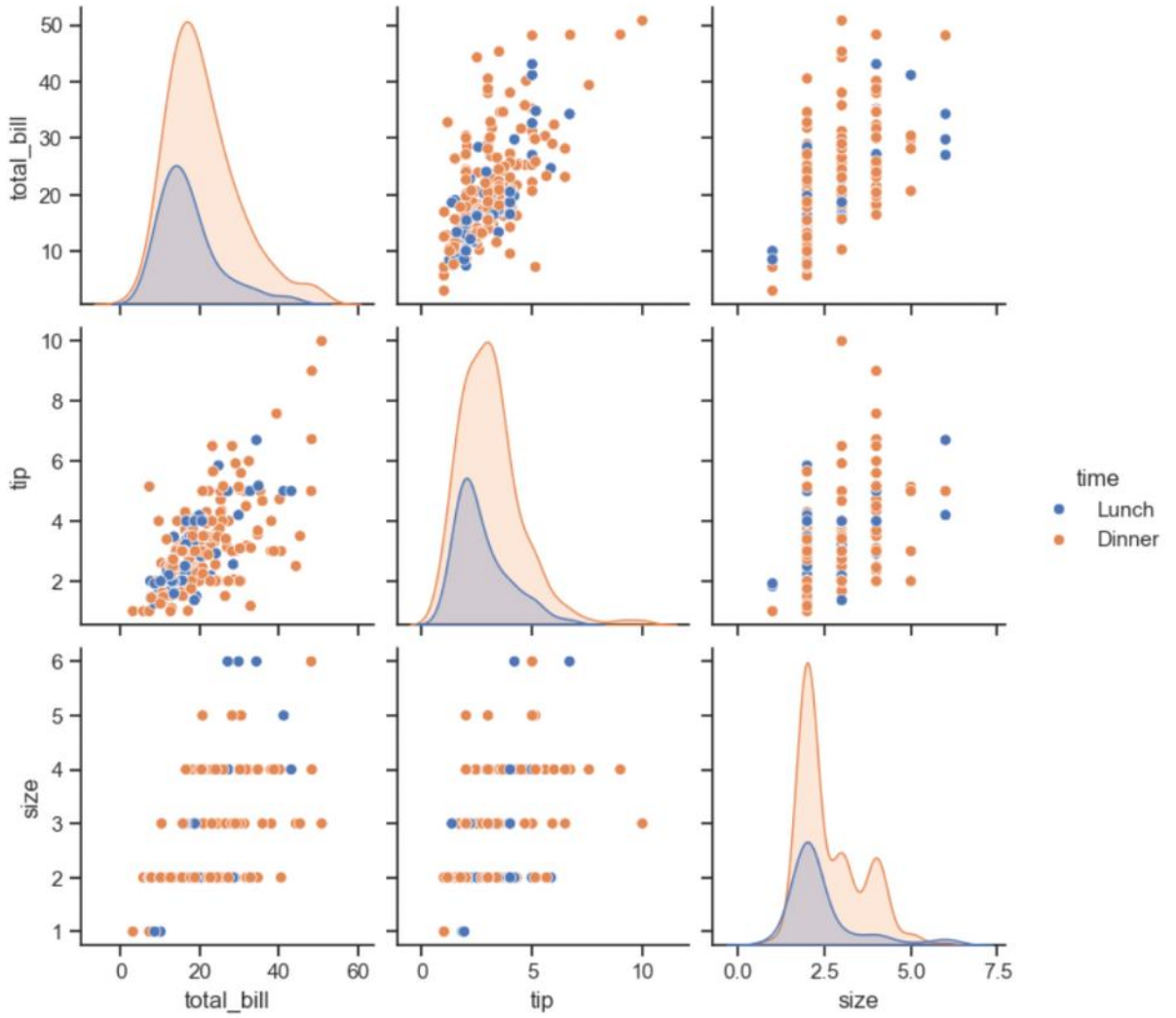
```
['Dinner', 'Lunch']
Categories (2, object): ['Lunch', 'Dinner']
```

```
tips['day'].unique()
```

```
['Sun', 'Sat', 'Thur', 'Fri']
Categories (4, object): ['Thur', 'Fri', 'Sat', 'Sun']
```

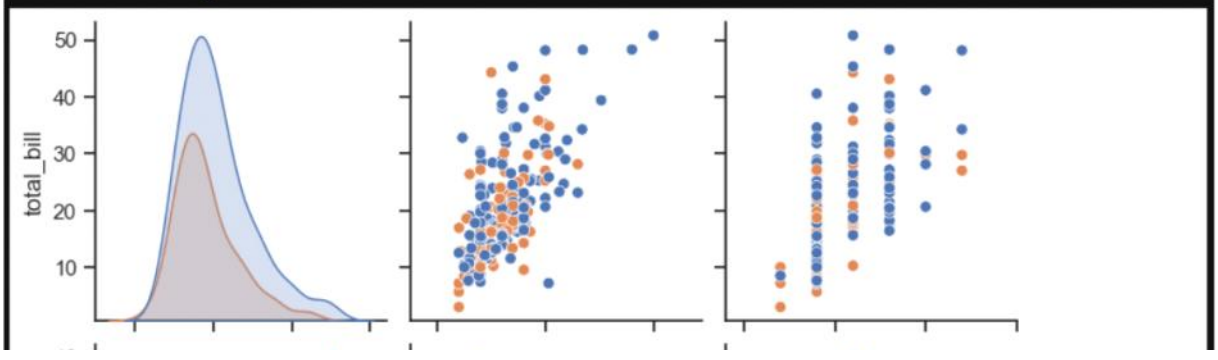
```
sns.pairplot(tips , hue = 'time')
```

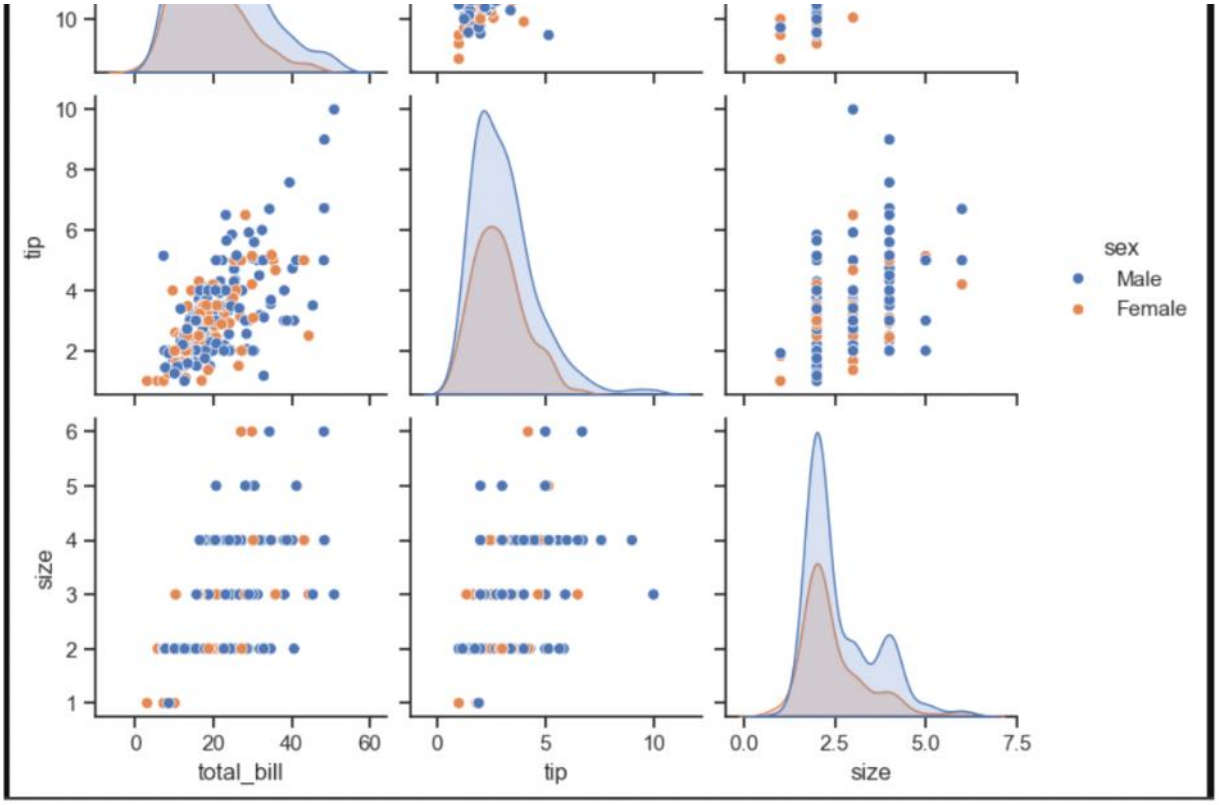
```
<seaborn.axisgrid.PairGrid at 0x2502a9af1a0>
```




```
sns.pairplot(tips , hue = 'sex')
```

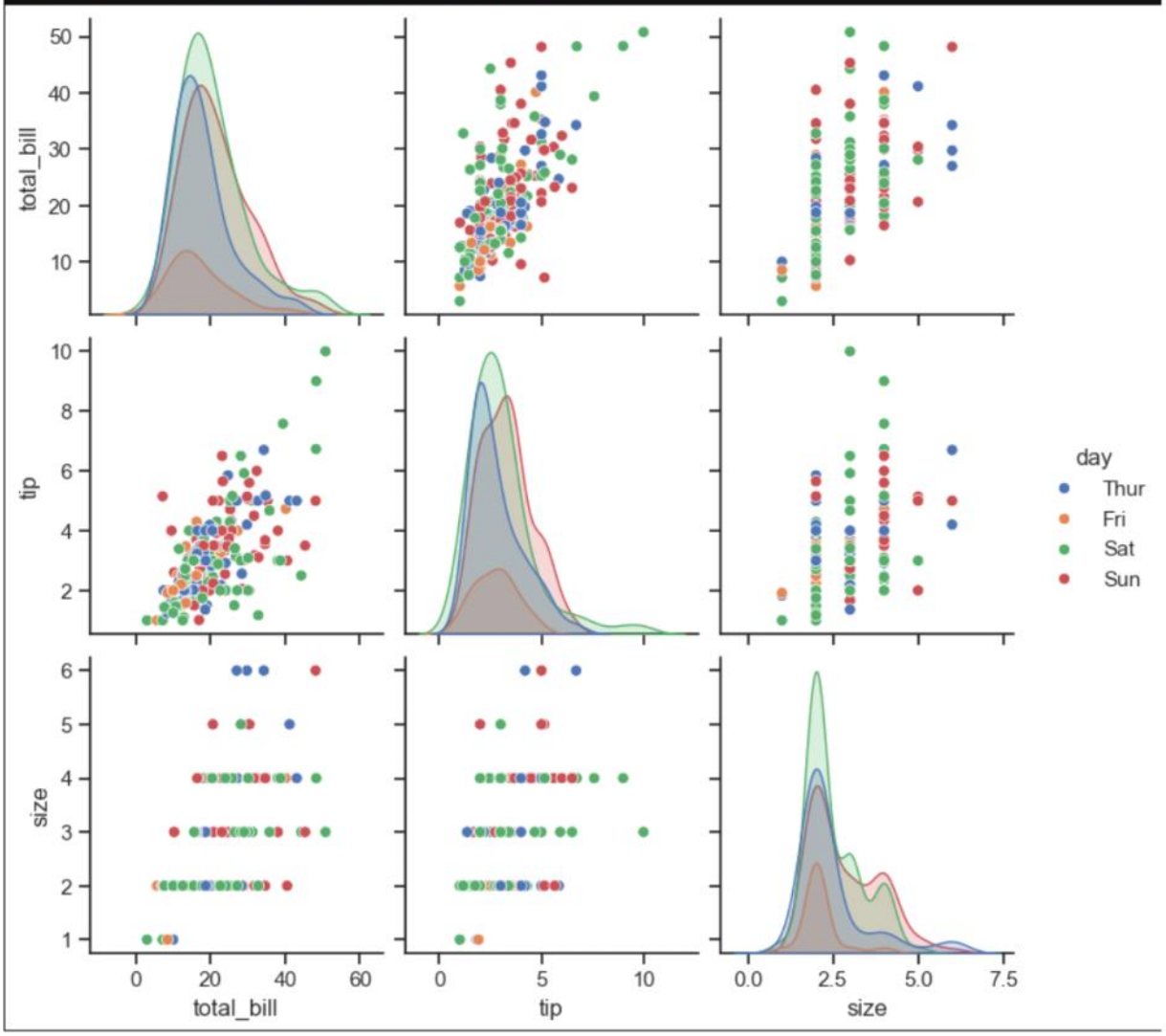
```
<seaborn.axisgrid.PairGrid at 0x2502aa24380>
```





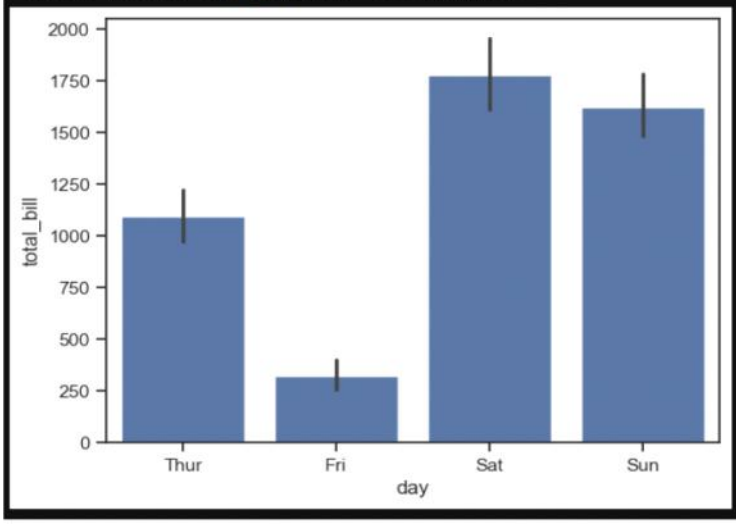

```
sns.pairplot(tips , hue = 'day')
```

```
<seaborn.axisgrid.PairGrid at 0x2502aaca840>
```



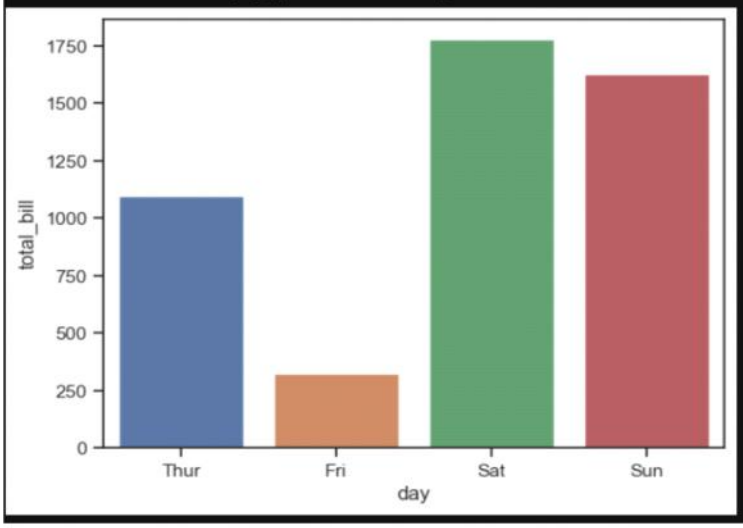
```
sns.barplot(tips, x="day", y="total_bill" , estimator='sum')
```

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

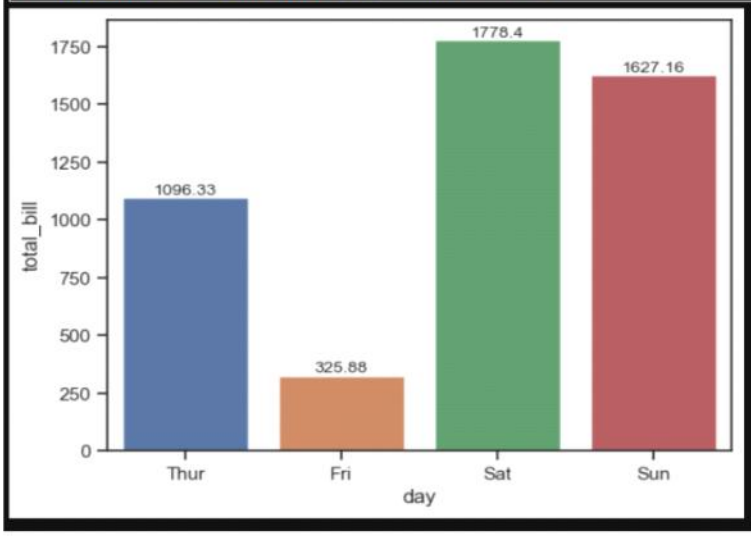



```
sns.barplot(tips, x="day", y="total_bill" , estimator='sum', errorbar=None, hue = 'day')
```

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



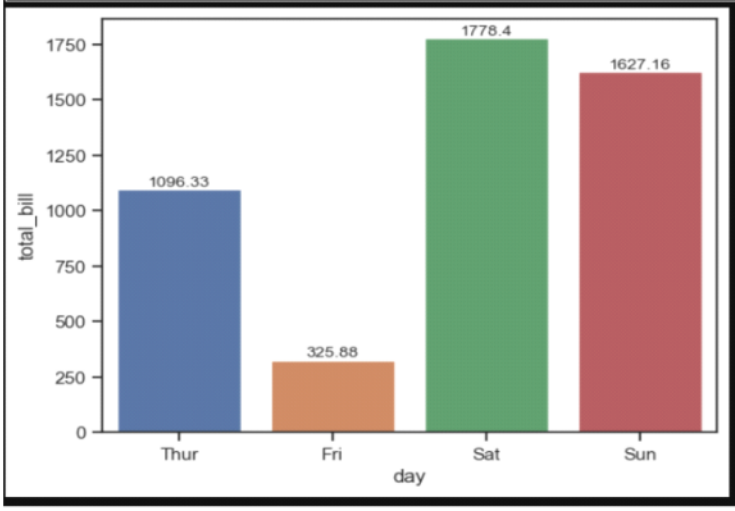

```
ax = sns.barplot(tips, x="day", y="total_bill" , estimator='sum', errorbar=None, hue = 'day')
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
ax.bar_label(ax.containers[2], fontsize=10)
ax.bar_label(ax.containers[3], fontsize=10);
```



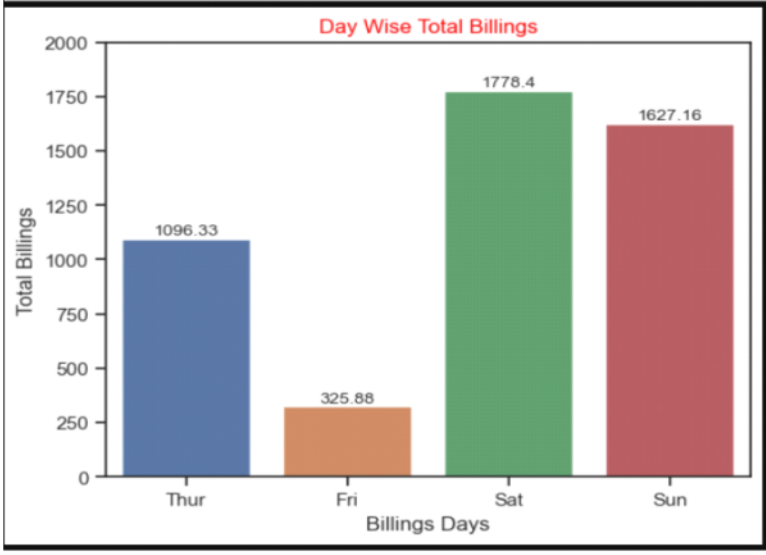
```
days_length = tips['day'].nunique()
days_length
```

```
4
```

```
ax = sns.barplot(tips, x="day", y="total_bill" , estimator='sum', errorbar=None, hue = 'day')
for i in range(0,days_length): # [0,1,2,3]
    ax.bar_label(ax.containers[i], fontsize=10)
```




```
ax = sns.barplot(tips, x="day", y="total_bill" , estimator='sum', errorbar=None, hue = 'day')
for i in range(0,days_length): # [0,1,2,3]
    ax.bar_label(ax.containers[i], fontsize=10)

plt.title('Day Wise Total Billings' , color = 'red')
plt.ylim(0,2000)
plt.xlabel('Billings Days')
plt.ylabel('Total Billings')
plt.show()
```




```
containers = ax.containers # [<BarContainer object of 1 artists>, ...]
heights = []
if containers:
    for i in range(0,4):
        nth_container = containers[i]  # First BarContainer
        for patch in nth_container.patches:
            heights.append(patch.get_height())
    print(heights)
```

```
[1096.33, 325.88, 1778.3999999999999, 1627.16]
```