St. Francis Institute of Technology, Mumbai-400 103

Department of Information Technology

A.Y. 2023-2024

Class: TE-ITA/B, Semester: VI

Subject: **Data Science Lab**

Experiment – 2: To implement Data Visualization/Data Exploratory Analysis using Matplolib and Seaborn.

1. **Aim:** To implement Data Visualization using Matplotlib and Seaborn.
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand Matplotlib Functions
   - Understand Seaborn Functions
3. **Outcomes:** After study of this experiment, the student will be able to
   - Understand data visualization /Exploratory Data Analysis using Matplolib and Seaborn.
4. **Prerequisite:** Fundamentals of Python Programming and Database Management System.
5. **Requirements:** Python Installation, Personal Computer, Windows operating system, Internet Connection, Microsoft Word.
6. **Pre-Experiment Exercise:**

   Brief Theory:

   Basic Concepts of Matplolib and Seaborn.
7. **Laboratory Exercise**

   A. Procedure:

```
import numpy as np
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
a

"""**matplotlib**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations
in Python. Matplotlib makes easy things easy and hard things possible.

1. Create publication quality plots.
2. Make interactive figures that can zoom, pan, update.
3. Customize visual style and layout.
4. Export to many file formats .
5. Embed in JupyterLab and Graphical User Interfaces.
6. Use a rich array of third-party packages built on Matplotlib.

**matplotlib.pyplot**

matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot
function makes some change to a figure: e.g., creates a figure, creates a plotting area in a
figure, plots some lines in a plotting area, decorates the plot with labels, etc.

**numpy.pi**

While both numpy.pi and math.pi provides us with the same value, the advantage of numpy.pi is that
it helps us by avoiding the dependency of obtaining the constant value of pi from a different
library.

**np.sin**

This mathematical function helps user to calculate trigonometric sine for all x(being the array
elements).
"""

# Sinusoidal plot
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(0.0,2.0,0.01)
y=np.sin(2*np.pi*x)
plt.plot(x,y)

#Barplot
import numpy as np
import matplotlib.pyplot as plt
```

```
fig=plt.figure()
x=[1,2,3,4,5,6,7]
y=[23,43,65,32,87,45,90]
plt.bar(x,y,color='orange')
plt.title('Average Score of Students')
plt.xlabel("Students")
plt.ylabel("Avg. Score")
fig.savefig("testing.jpg")

"""**Error Bars**

An error bar is a line through a point on a graph, parallel to one of the axes, which represents
the uncertainty or variation of the corresponding coordinate of the point.

Error bars can communicate the following information about your data:

1. How spread the data are around the mean value (small SD bar = low spread, data are clumped
around the mean; larger SD bar = larger spread, data are more variable from the mean).

2. The reliability of the mean value as a representative number for the data set.  In other words,
how accurately the mean value represents the data (small SD bar = more reliable, larger SD bar =
less reliable).  It's important to note that just because you have a larger SD, it does not
indicate your data is not valid.
3. The likelihood of there being a significant difference between data sets.

**Significant Difference**

A "significant difference" means that the results that are seen are most likely not due to chance
or sampling error.  In any experiment or observation that involves sampling from a population,
there is always the possibility that an observed effect would have occurred due to sampling error
alone.  But if result is "significant,"  then the investigator may conclude that the observed
effect actually reflects the characteristics of the population rather than just sampling error or
chance.

#single barplots with error bars
N=5
men=(20,35,30,36,27)
women=(25,32,34,20,25)
mstd=(2,2,2,2,2)
i=np.arange(N)
p1=plt.bar(i,men,width=0.35,yerr=mstd)
p2=plt.bar(i,women,width=0.35,bottom=men,yerr=mstd)
plt.xticks(i,('G1','G2','G3','G4','G5'))
plt.yticks(np.arange(0,90,10))
plt.legend((p1[0],p2[0]),('Men','Women'))

"""**Histogram**

A histogram is a graphical representation of a grouped frequency distribution with continuous
classes.

A histogram is a diagram involving rectangles whose area is proportional to the frequency of a
variable and width is equal to the class interval.

The histogram graph is used under certain conditions. They are:

1. The data should be numerical.
2. A histogram is used to check the shape of the data distribution.
3. Used to check whether the process changes from one period to another.
4. Used to determine whether the output is different when it involves two or more processes.
5. Used to analyse whether the given process meets the customer requirements.

Histograms and bar charts are different. In the bar chart, each column represents the group which
is defined by a categorical variable, whereas in the histogram each column is defined by the
continuous and quantitative variable.

**Difference Between Histogram and Bar Graph**

1. It is a two-dimensional figure
2. The frequency is shown by the area of each rectangle
3. It shows rectangles touching each other

**Bar Graph**

1. It is a one-dimensional figure
2. The height shows the frequency and the width has no significance.
3. It consists of rectangles separated from each other with equal spaces.

"""

#Histogram
m=100
std=15
x=m+std*np.random.randn(480)
y=50
plt.hist(x,y,color='blue',density=False)
plt.grid(color='gray',linestyle='--',linewidth=2,axis='x')
```

```
plt.grid(color='gray',linestyle='--',linewidth=2,axis='y')

#Pie Chart
labels='cricket','Hockey','Tennis','Football'
v=[40,20,10,30]
e=(0,0,0.1,0)
plt.pie(v,explode=e,labels=labels,autopct='%1.3f%%',shadow=True,startangle=0)

"""numpy.linspace(start, stop, num)
Return evenly spaced numbers over a specified interval.

Returns num evenly spaced samples, calculated over the interval [start, stop].

The endpoint of the interval can optionally be excluded.
"""

x=np.linspace(0,10,1000)
plt.plot(x,np.sin(x))
plt.plot(x,np.sin(x-1),color='g')
plt.plot(x,np.sin(x-2),color='0.25')
plt.plot(x,np.sin(x-3),color='red')

x=np.linspace(0,10,1000)
plt.plot(x,np.sin(x))
plt.plot(x,np.sin(x-1),linestyle='solid')
plt.plot(x,np.sin(x-2),linestyle='dashed')
plt.plot(x,np.sin(x-3),linestyle='dashdot')
plt.plot(x,np.sin(x-4),linestyle='dotted')

#plt.plot(x,np.sin(x))
plt.plot(x,np.sin(x-1),linestyle='-')
plt.plot(x,np.sin(x-2),linestyle='--')
plt.plot(x,np.sin(x-3),linestyle='-.')
plt.plot(x,np.sin(x-4),linestyle=':')

#Scatterplot
x=np.linspace(0,10,50)
y=np.sin(x)
plt.scatter(x,y)

x=np.linspace(0,10,50)
plt.plot(x,np.sin(x),'-o')


#Seaborn
import seaborn as sns
data=sns.load_dataset("iris")
sns.lineplot(x="sepal_length", y="sepal_width",data=data)

sns.scatterplot(x="sepal_length", y="sepal_width",data=data)

sns.stripplot(x="species", y="sepal_length",data=data)



sns.violinplot(x="species", y="sepal_length",data=data)

sns.swarmplot(x="species", y="sepal_length",data=data)
```

Paste Screenshots of above commands.

8. Post-Experiments Exercise

A. Extended Theory: (Soft Copy)

What is the difference between table and graph? (Ref Link 1)

Use Automobile Dataset and apply visualization methods on it.

B. Questions:

What is the difference between Matplotlib and Seaborn? (Refer Link 2)

C. Conclusion:

Write the significance of the topic studied in the experiment.

D. References:

1. https://www.wallstreetmojo.com/graphs-vs-charts/
2. https://sdsclub.com/seaborn-vs-matplotlib-python-visualization-tools-battle/

3. https://matplotlib.org/stable/tutorials/index
4. https://seaborn.pydata.org/

In [ ]:

```
import numpy as np
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
a
```

Out[ ]:

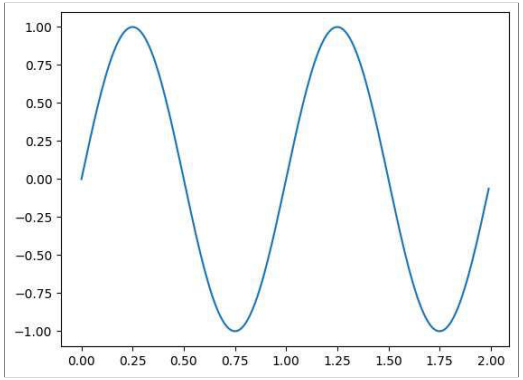
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [2]:

```
# Sinusoidal plot
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(0.0,2.0,0.01)
y=np.sin(2*np.pi*x)
plt.plot(x,y)
```

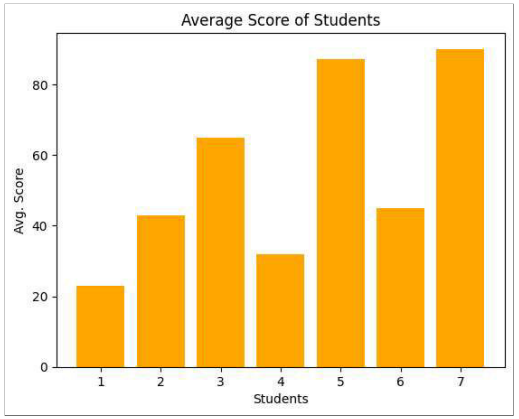
Out[2]:

```
[<matplotlib.lines.Line2D at 0x167c996e5a0>]
```



In [ ]:

```
#Barplot
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=[1,2,3,4,5,6,7]
y=[23,43,65,32,87,45,90]
plt.bar(x,y,color='orange')
plt.title('Average Score of Students')
plt.xlabel("Students")
plt.ylabel("Avg. Score")
fig.savefig("testing.jpg")
```

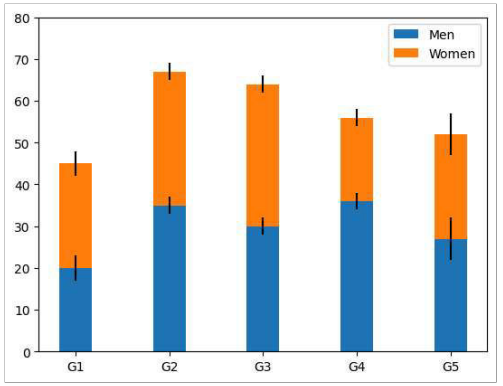


In [ ]:

```
#single barplots with error bars
N=5
men=(20,35,30,36,27)
women=(25,32,34,20,25)
mstd=(3,2,2,2,5)
i=np.arange(N)
p1=plt.bar(i,men,width=0.35,yerr=mstd)
p2=plt.bar(i,women,width=0.35,bottom=men,yerr=mstd)
plt.xticks(i,('G1','G2','G3','G4','G5'))
plt.yticks(np.arange(0,90,10))
plt.legend((p1[0],p2[0]),('Men','Women'))
```

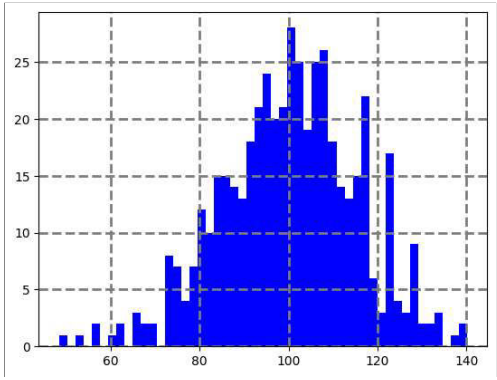
Out[ ]:

```
<matplotlib.legend.Legend at 0x7bf73fa270d0>
```



In [ ]:

```
#Histogram
m=100
std=15
x=m+std*np.random.randn(480)
y=50
plt.hist(x,y,color='blue',density=False)
plt.grid(color='gray',linestyle='--',linewidth=2,axis='x')
plt.grid(color='gray',linestyle='--',linewidth=2,axis='y')
```

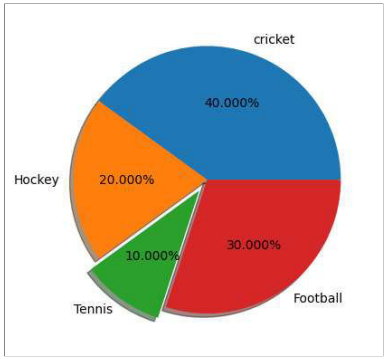


In [ ]:

```
#Pie Chart
labels='cricket','Hockey','Tennis','Football'
v=[40,20,10,30]
e=(0,0,0.1,0)
plt.pie(v,explode=e,labels=labels,autopct='%1.3f%%',shadow=True,startangle=0)
```

Out[ ]:

```
([<matplotlib.patches.Wedge at 0x7bf6fef5ffd0>,
  <matplotlib.patches.Wedge at 0x7bf6fc83d990>,
  <matplotlib.patches.Wedge at 0x7bf6fc83fe50>,
  <matplotlib.patches.Wedge at 0x7bf6fc652650>],
 [Text(0.33991867422268784, 1.0461621742897658, 'cricket'),
  Text(-1.0999999999999988, -5.149471622296949e-08, 'Hockey'),
  Text(-0.7053422436693694, -0.9708204361752281, 'Tennis'),
  Text(0.6465638691739385, -0.8899186272232008, 'Football')],
 [Text(0.1854101859396479, 0.5706339132489631, '40.000%'),
  Text(-0.5999999999999993, -2.808802703071063e-08, '20.000%'),
  Text(-0.4114496421404655, -0.5663119211022163, '10.000%'),
  Text(0.35267120136760277, -0.48541016030356404, '30.000%')])
```

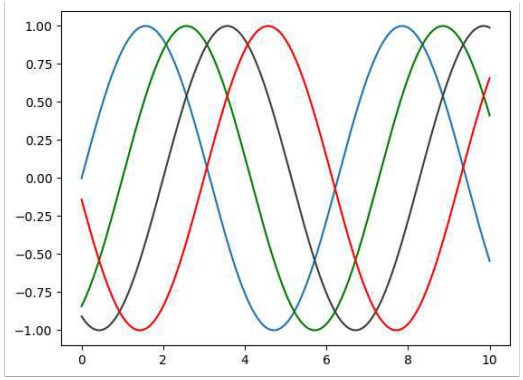


In [ ]:

```
x=np.linspace(0,10,1000) #generates linearly spaced vector with 1000 samples
plt.plot(x,np.sin(x))
plt.plot(x,np.sin(x-1),color='g')
plt.plot(x,np.sin(x-2),color='0.25')
plt.plot(x,np.sin(x-3),color='red')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7bf6fc6981d0>]
```

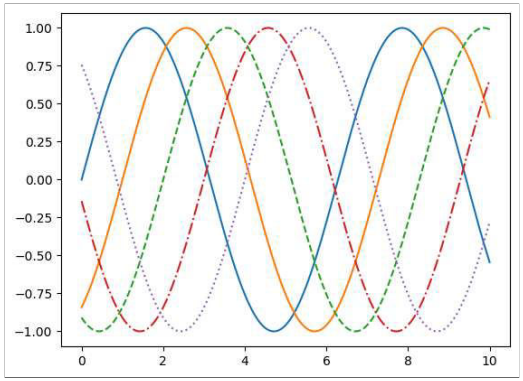


In [ ]:

```
x=np.linspace(0,10,1000)
plt.plot(x,np.sin(x))
plt.plot(x,np.sin(x-1),linestyle='solid')
plt.plot(x,np.sin(x-2),linestyle='dashed')
plt.plot(x,np.sin(x-3),linestyle='dashdot')
plt.plot(x,np.sin(x-4),linestyle='dotted')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7bf6fc715f10>]
```

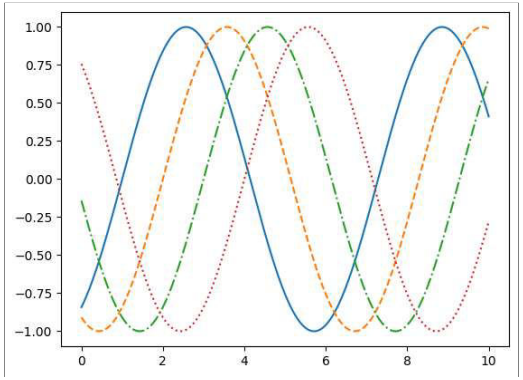


In [ ]:

```
#plt.plot(x,np.sin(x))
plt.plot(x,np.sin(x-1),linestyle='-')
plt.plot(x,np.sin(x-2),linestyle='--')
plt.plot(x,np.sin(x-3),linestyle='-.')
plt.plot(x,np.sin(x-4),linestyle=':')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7bf6fc58b210>]
```

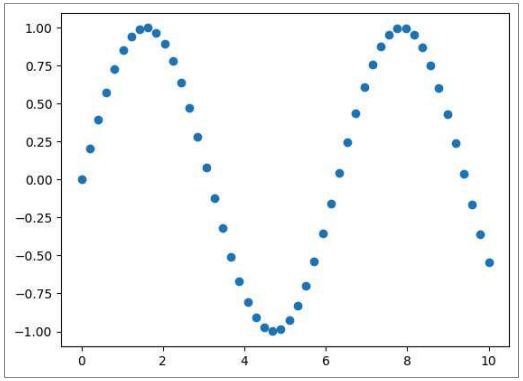


In [ ]:

```
#Scatterplot
x=np.linspace(0,10,50)
y=np.sin(x)
plt.scatter(x,y)
```

Out[ ]:

```
<matplotlib.collections.PathCollection at 0x7bf6fc7c2e50>
```

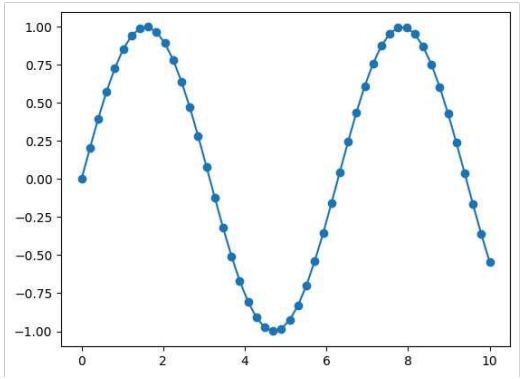


In [ ]:

```
x=np.linspace(0,10,50)
plt.plot(x,np.sin(x),'-o')
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7bf6fc46aed0>]
```

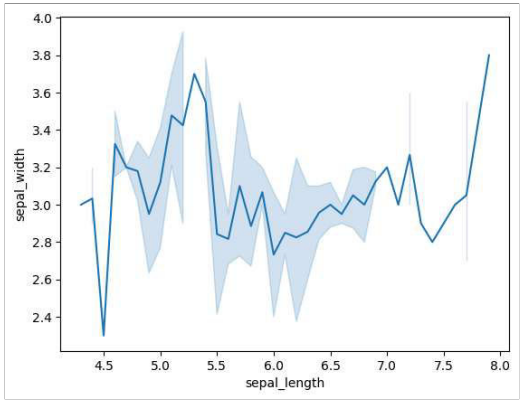


In [ ]:

```
#Seaborn
import seaborn as sns
data=sns.load_dataset("iris")
sns.lineplot(x="sepal_length", y="sepal_width",data=data)
```

Out[ ]:

```
<Axes: xlabel='sepal_length', ylabel='sepal_width'>
```

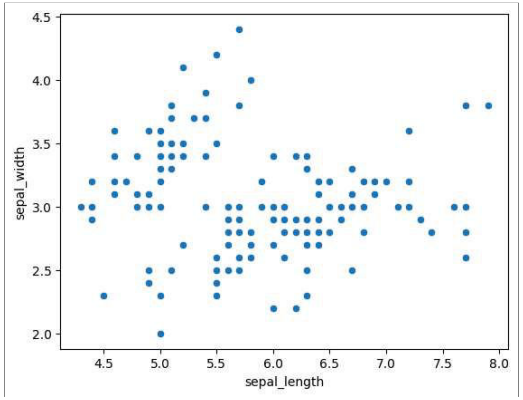


In [ ]:

```
sns.scatterplot(x="sepal_length", y="sepal_width",data=data)
```

Out[ ]:

```
<Axes: xlabel='sepal_length', ylabel='sepal_width'>
```

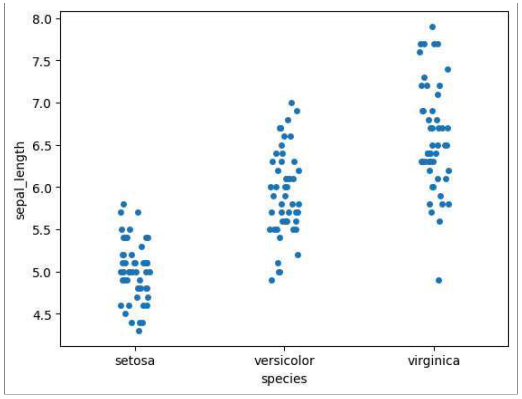


In [ ]:

```
sns.stripplot(x="species", y="sepal_length",data=data)
```

Out[ ]:

```
<Axes: xlabel='species', ylabel='sepal_length'>
```

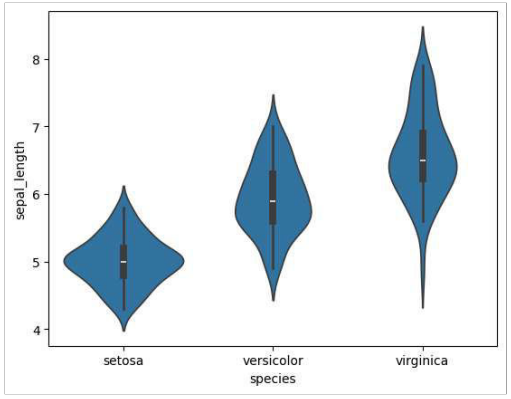


In [ ]:

```
sns.violinplot(x="species", y="sepal_length",data=data)
```

Out[ ]:

```
<Axes: xlabel='species', ylabel='sepal_length'>
```

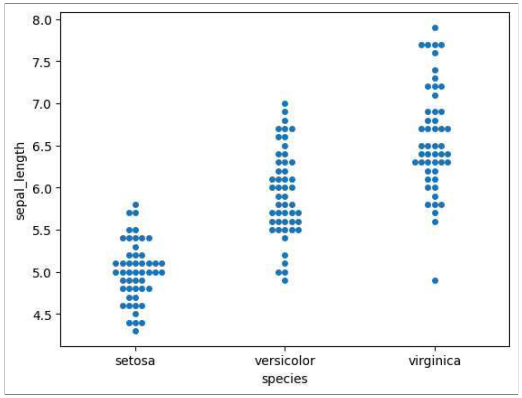


In [ ]:

```
sns.swarmplot(x="species", y="sepal_length",data=data)
```

Out[ ]:

```
<Axes: xlabel='species', ylabel='sepal_length'>
```



Importing Libraries and Data and Data cleaning

In [146]:

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

data = pd.read_csv("/content/Automobile.csv")
print(data.head())
```

```
   symboling normalized-losses         make fuel aspiration num-of-doors  \
0          3                 ?  alfa-romero  gas        std          two
1          3                 ?  alfa-romero  gas        std          two
2          1                 ?  alfa-romero  gas        std          two
3          2               164         audi  gas        std         four
4          2               164         audi  gas        std         four

    body-style drive-wheels engine-location  wheel-base  ...  engine-size  \
0  convertible          rwd           front        88.6  ...          130
1  convertible          rwd           front        88.6  ...          130
2    hatchback          rwd           front        94.5  ...          152
3        sedan          fwd           front        99.8  ...          109
4        sedan          4wd           front        99.4  ...          136

   fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city_mpg  \
0         mpfi  3.47    2.68               9.0        111      5000       21
1         mpfi  3.47    2.68               9.0        111      5000       21
2         mpfi  2.68    3.47               9.0        154      5000       19
3         mpfi  3.19     3.4              10.0        102      5500       24
4         mpfi  3.19     3.4               8.0        115      5500       18

  highway_mpg  price
0          27  13495
1          27  16500
2          26  16500
3          30  13950
4          22  17450

[5 rows x 26 columns]
```

Cleaning the data by replacing '?' with missing values (NaN), converting specific columns ('price', 'horsepower', 'engine-size', 'curb-weight') to numeric values, and dropping rows with missing values in these columns.

In [147]:

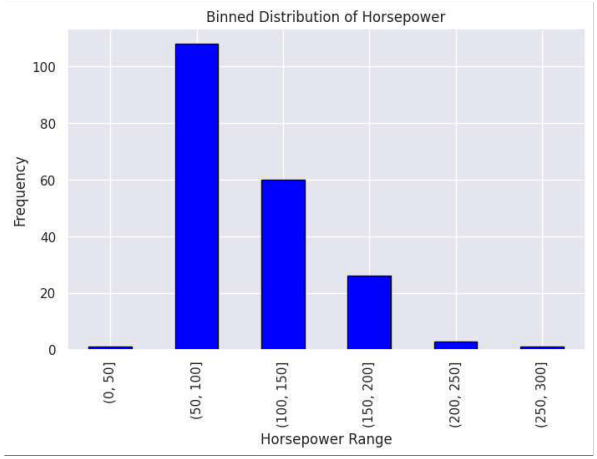
```
data.replace('?', pd.NA, inplace=True)

data['price'] = pd.to_numeric(data['price'], errors='coerce')
data['horsepower'] = pd.to_numeric(data['horsepower'], errors='coerce')
data['engine-size'] = pd.to_numeric(data['engine-size'], errors='coerce')
data['curb-weight'] = pd.to_numeric(data['curb-weight'], errors='coerce')

data = data.dropna(subset=['price', 'horsepower', 'engine-size', 'curb-weight'])
```

Visualization of Automobile data using Matplotlib

Plotting a bar chart to visualize the frequency distribution of horsepower values by grouping them into predefined bins (ranges).

In [148]:

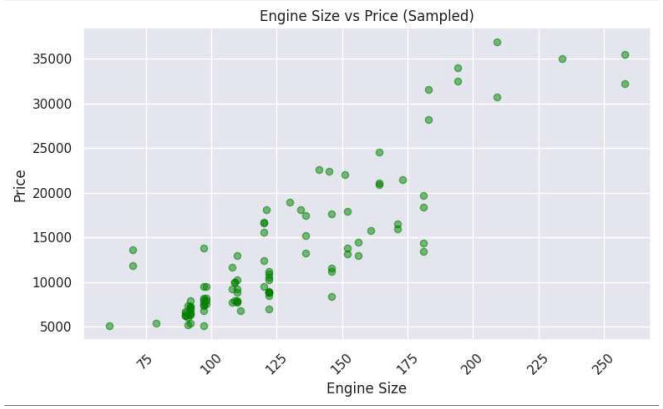
```
bins = [0, 50, 100, 150, 200, 250, 300]
data['horsepower_binned'] = pd.cut(data['horsepower'], bins=bins)
horsepower_counts = data['horsepower_binned'].value_counts()
plt.figure(figsize=(8, 5))
horsepower_counts.sort_index().plot(kind='bar', color='blue', edgecolor='black')
plt.title('Binned Distribution of Horsepower')
plt.xlabel('Horsepower Range')
plt.ylabel('Frequency')
plt.show()
```



Plotting a scatter plot of engine size vs. price

Tn [149]:

```
data_sampled = data.sample(100, random_state=1)
plt.figure(figsize=(8, 5))
plt.scatter(data_sampled['engine-size'], data_sampled['price'], color='green', alpha=0.5)
plt.title('Engine Size vs Price (Sampled)')
plt.xlabel('Engine Size')
plt.ylabel('Price')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Plotting a line chart showing the average price based on curb weight, with curb weight on the x-axis and average price on the y-axis.

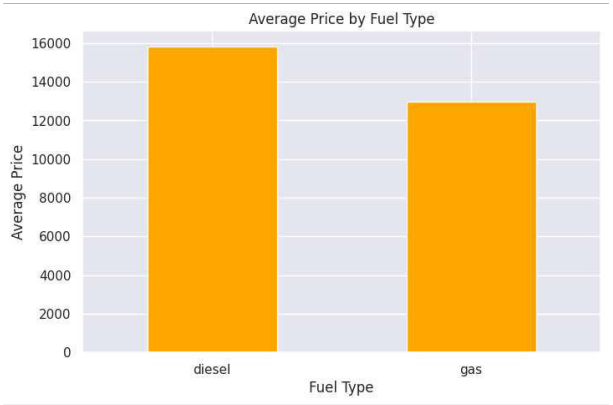
In [150]:

```
cw_price_avg = data.groupby('curb-weight')['price'].mean().reset_index()
plt.figure(figsize=(8, 5))
plt.plot(cw_price_avg['curb-weight'], cw_price_avg['price'], linestyle='-', marker='o', color='purple')
plt.title('Average Price by Curb Weight')
plt.xlabel('Curb Weight')
plt.ylabel('Average Price')
plt.show()
```

Plotting a bar chart to display the average price for each fuel type, with fuel type on the x-axis and average price on the y-axis.

In [151]:

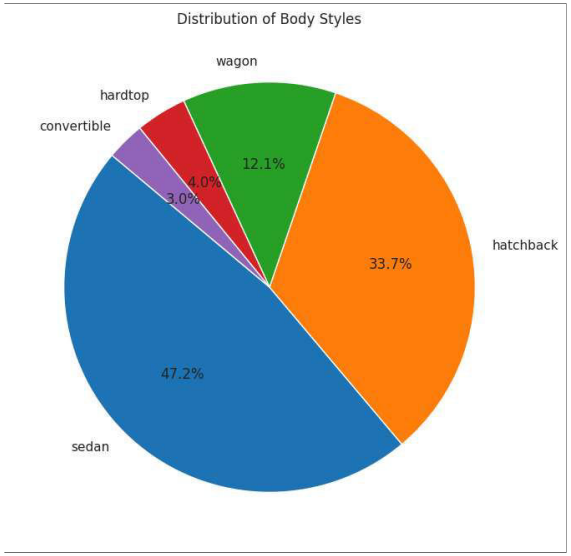
```
avg_price_by_fuel = data.groupby('fuel')['price'].mean()
plt.figure(figsize=(8, 5))
avg_price_by_fuel.plot(kind='bar', color='orange')
plt.title('Average Price by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Average Price')
plt.xticks(rotation=0)
plt.show()
```



Plotting a pie chart to show the distribution of different body styles, with percentage labels for each body style category.

In [152]:

```
body_style_counts = data['body-style'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(body_style_counts, labels=body_style_counts.index, autopct='%1.1f%%', startangle=140, colors=plt.
plt.title('Distribution of Body Styles')
plt.show()
```

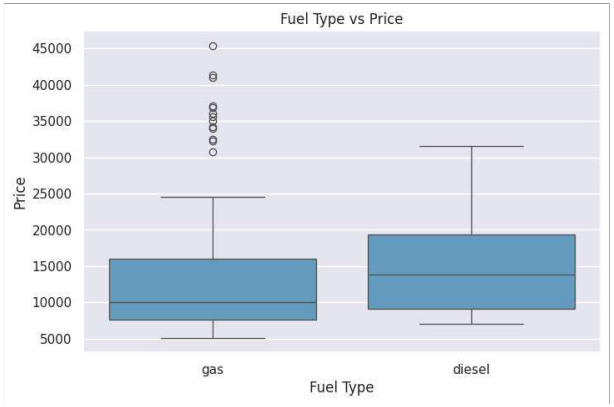


Visualization of Automobile data using Seaborn

Plotting a boxplot to visualize the distribution of price based on fuel type, with fuel type on the x-axis and price on the y-axis.

In [153]:

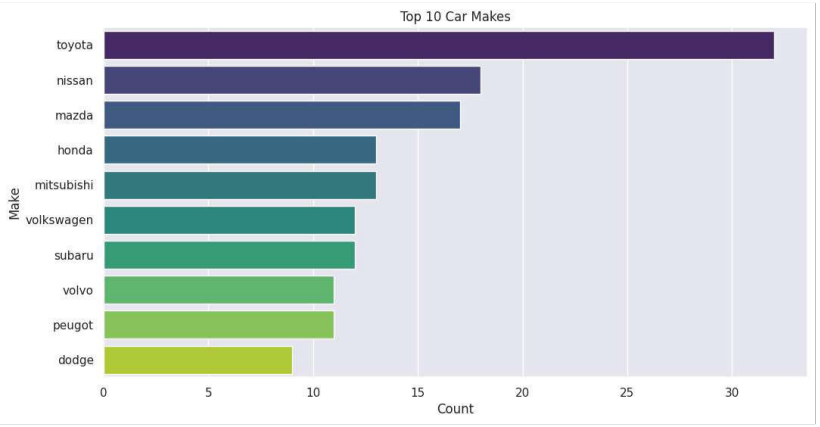
```
plt.figure(figsize=(8, 5))
sns.boxplot(x='fuel', y='price', data=data)
plt.title('Fuel Type vs Price')
plt.xlabel('Fuel Type')
plt.ylabel('Price')
plt.show()
```



Plotting a horizontal count plot to show the top 10 car makes based on frequency, with the count on the x-axis and car make on the y-axis.

In [155]:

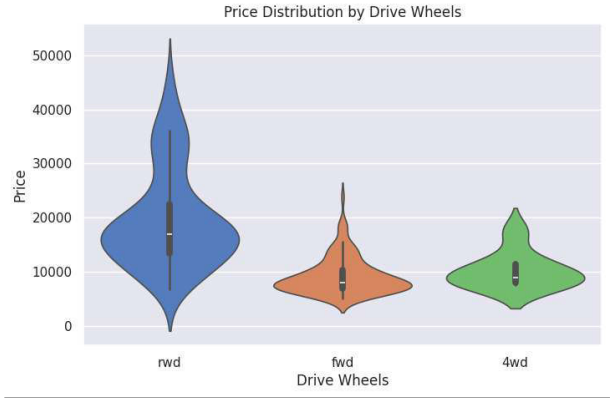
```
top_10_makes = data['make'].value_counts().nlargest(10).index
filtered_data = data[data['make'].isin(top_10_makes)]
plt.figure(figsize=(12, 6))
sns.countplot(y='make', data=filtered_data, order=top_10_makes, palette='viridis')
plt.title('Top 10 Car Makes')
plt.xlabel('Count')
plt.ylabel('Make')
plt.show()
```



Plotting a violin plot to visualize the price distribution across different drive wheels categories, with drive wheels on the x-axis and price on the y-axis.

In [156]:

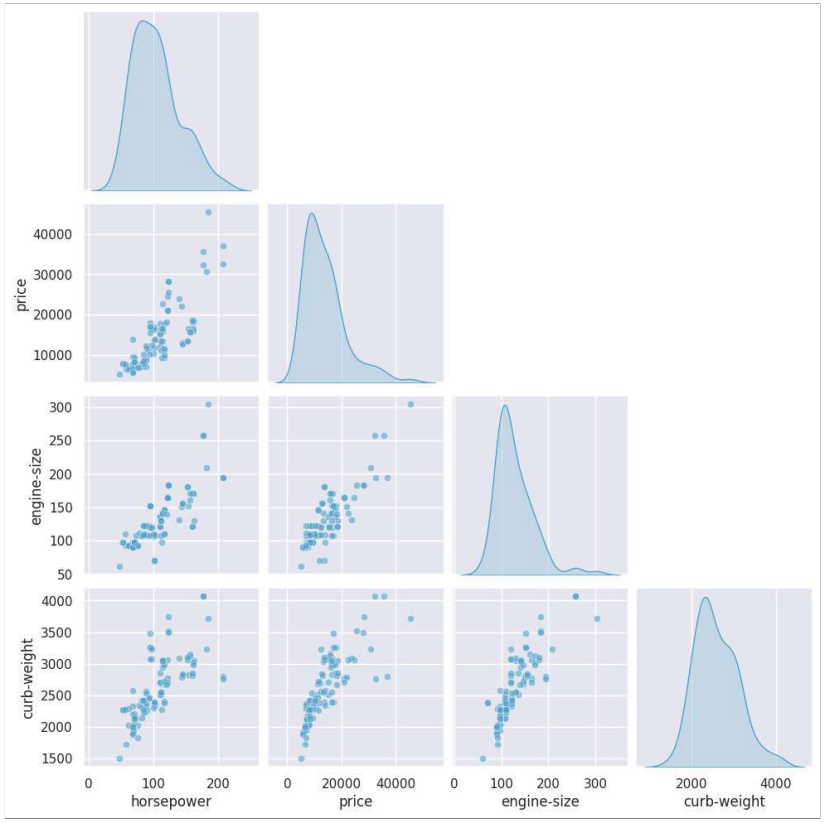
```
plt.figure(figsize=(8, 5))
sns.violinplot(x='drive-wheels', y='price', data=data, palette='muted')
plt.title('Price Distribution by Drive Wheels')
plt.xlabel('Drive Wheels')
plt.ylabel('Price')
plt.show()
```



Plotting a pairplot of sampled numerical columns (horsepower, price, engine size, and curb weight) to visualize the relationships between them, with kernel density estimates on the diagonal and transparency applied to the scatter plots.

Tn [157]:

```
numerical_columns = ['horsepower', 'price', 'engine-size', 'curb-weight']
data_sampled_pairplot = data[numerical_columns].sample(100, random_state=2)
sns.pairplot(data_sampled_pairplot, diag_kind='kde', corner=True, plot_kws={'alpha': 0.6})
plt.show()
```


price
engine-size
curb-weight
horsepower
price
engine-size
curb-weight