MAHARAJA SURAJMAL INSTITUTE

C-4, Janakpuri
New Delhi - 110058

DEPARTMENT OF COMPUTER SCIENCE

BCA-312
Data Visualization & Analytics (Practical)

Submitted to:
Dr. Tarunim Sharma
(Associate Professor)
Department of Computer Science

Submitted by:
Name: Hardik Sethi
Roll no. 07114902021
Class: BCA 6(A) - Morning

INDEX

S. No.	Practical Name	Signature
	Assignment 1:	
1.	Upload Toyota.csv in dataframe df.	
2.	What is the data type of MetColor?	
3.	How many null value are there in KM field?	
4.	Which column has 7 unique values.	
5.	How many records are there? What is mean, median of age grouped by FuelType?	
6.	Replace three, four, five value in Doors column to 3,4,5 respectively.	
7.	Change the datatype of Doors to int64.	
8.	Impute the value of Price with median	
9.	Replace ???? in HP field with mean.	
10.	Impute blank values in FuelType with Mode.	
11.	Delete the rows with MetColor and Age as blank.	
12.	Replace ?? value in KM with Mean	
13.	What is the mean, median and mode of KM field.	
14.	Categorise Age into AgeCat column with 0-10 NewCarCat, 11-20 MediumCarCat, 21- highest value – OldCarCat.	
15.	Create Dummy fields for FuelType.	
	Assignment 2 :	
1	Read Carfeatures.csv	
2	How many attributes are there?	
3	Fill the missing values, impute data, check and eliminate for outlier.	
4	How many unique values are there in popularity column?	
5	Using matplot/seaborn a) Draw lineplot using relplot function of seaborn between	

	highway MPG and city mpg b) Draw scatter plot between Engine HP and Popularity c) Draw barplot between Vehicle Style and highway MPG using seaborn function d) Draw piechart on the basis of Market Category, average of MSRP. e) Draw boxplot MSRP	
6	Using pandas draw the above four plots.	
	Assignment 3 :	
1	Perform one sample z-test in python for the given problem:	
2	Perform One Sample t-Test in Python for the given problem	
	Assignment 4 :	
1	Create TensileStrength.xlsx file with the given data	
2	Read the excel file in dataframe, use the melt command of pandas to pivot the table	
3	Run one way anova to check whether the null hypothesis is rejected or not rejected	
4	If null hypothesis is rejected test for Posthoc test (Tukey's) and discuss the result.	
	Assignment 5 :	
1	Create a database in SQLite "College" and create a table Student with five fields : Name, EnrolmentNo, Percentage, Course, Batch.	
2	Create an interface in tkinter to insert new data in Student table, through Entry text, radio buttons and on click of "'Add" button, the record should be added to the table.	
3	Create "Display" and "Display All"' buttons, and on click of button- display the current record in above text fields and on click of "display all" button , show all the records in Listbox.	

<table>
<tr><td></td><td>Syllabus Practicals :</td><td></td></tr>
<tr><td>1</td><td>Write a program to create a DataFrame have E-commerce data and perform selection of row/ column using loc() and iloc()</td><td></td></tr>
<tr><td>2</td><td>Create a Series object S5 containing numbers. Write a program to store the square of the series values in object S6. Display S6's values which are >15.</td><td></td></tr>
<tr><td>3</td><td>Write a program to fill all missing values in a DataFrame with zero.</td><td></td></tr>
<tr><td>4</td><td>Program for combining DataFrames using concat(), join(),merge()</td><td></td></tr>
<tr><td>5</td><td>Write a program to draw bar graph for the following data for the Medal tally of CWG-2018:-

<table><tr><td>Gold</td><td>Silver</td><td>Bronze</td><td>Total</td></tr><tr><td>26</td><td>20</td><td>20</td><td>66</td></tr></table></td><td></td></tr>
<tr><td>6</td><td>Implement Line lot, Dist lot, Lmplot, Count plot using Seaborn library</td><td></td></tr>
<tr><td>7</td><td>Create a DataFrame that stores aid (Toys,books,uniform,shoes) by NGOs for different states. Write a program to display the aid for:-
(a) Books and Uniforms only
(b) Shoes only</td><td></td></tr>
<tr><td>8</td><td>Create a DataFrame ndf have Name, Gender, Position, City, Age, Projects. Write a program to summarize how many projects are being handled by each position for each city? Use pivot()</td><td></td></tr>
<tr><td>9</td><td>Marks is a list that stores marks of a student in a 10 unit test. Write a program to plot Line charts for the student's performance in these 10 tests.</td><td></td></tr>
<tr><td>10</td><td>Write a program to plot a horizontal bar chart from the height of some students.</td><td></td></tr>
<tr><td>11</td><td>Write a program to implement ANNOVA.</td><td></td></tr>
</table>

12	Write a program to show correlation between two randomly generated numbers.	
13	Write a program to implement Covariance.	
14	Create a GUI based form for admission purposes for your college.	
15	The created GUI based application form is connected to a database and uses insert query to enter data.	

Practical Assignment 1

1) Upload Toyota.csv in dataframe df.

```
[1]: import os
     import warnings
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
[3]: #1
     filepath = r"C:\Users\PARKASH\Desktop\Toyota.csv";

     df = pd.read_csv(filepath, na_values=['??','????'])
     df
```

[3]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500.0	23.0	46986.0	Diesel	90.0	1.0	0	2000	three	1165
1	1	13750.0	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	2	13950.0	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	3	NaN	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	4	13750.0	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500.0	NaN	20544.0	Petrol	86.0	1.0	0	1300	3	1025
1432	1432	10845.0	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015
1433	1433	8500.0	NaN	17016.0	Petrol	86.0	0.0	0	1300	3	1015
1434	1434	7250.0	70.0	NaN	NaN	86.0	1.0	0	1300	3	1015
1435	1435	6950.0	76.0	1.0	Petrol	110.0	0.0	0	1600	5	1114

2) What is the data type of MetColor?

```
[4]: #2
     datatypes = df.dtypes['MetColor']
     datatypes
```

```
[4]: dtype('float64')
```

3) How many null value are there in KM field?

```
[5]: #3
     print(df['KM'].isnull().sum())
```

```
15
```

4) Which column has 7 unique values.

```
[6]: #4
     a = df['Doors'].unique()
     print(sorted(a))
```

```
['2', '3', '4', '5', 'five', 'four', 'three']
```

5) How many records are there? What is mean, median of age grouped by FuelType?

[7]:
```
#5
count_row = df.shape[0]
print(count_row)
print()

#5.1
print(df.groupby(['FuelType'])['Age'].mean())
print()

#5.2
print(df.groupby(['FuelType'])['Age'].median())
```

```
1436

FuelType
CNG       56.928571
Diesel    51.795620
Petrol    56.234432
Name: Age, dtype: float64

FuelType
CNG       57.0
Diesel    56.0
Petrol    61.0
Name: Age, dtype: float64
```

6) Replace three, four, five value in Doors column to 3,4,5 respectively.

[8]:
```
#6
df.replace({'Doors' : { 'five' : 5, 'four' : 4, 'three' : 3 }},inplace=True)
df
```

[8]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500.0	23.0	46986.0	Diesel	90.0	1.0	0	2000	3	1165
1	1	13750.0	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	2	13950.0	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	3	NaN	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	4	13750.0	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500.0	NaN	20544.0	Petrol	86.0	1.0	0	1300	3	1025
1432	1432	10845.0	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015
1433	1433	8500.0	NaN	17016.0	Petrol	86.0	0.0	0	1300	3	1015

7) Change the datatype of Doors to int64.

[9]:
```
#7
df['Doors'] = df['Doors'].astype('int64')
datatypes1 = df.dtypes['Doors']
datatypes1
```

[9]:
```
dtype('int64')
```

8) Impute the value of Price with median.

```
[10]: #8
df['Price'].fillna(df['Price'].median(), inplace = True)
df
```

[10]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500.0	23.0	46986.0	Diesel	90.0	1.0	0	2000	3	1165
1	1	13750.0	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	2	13950.0	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	3	9750.0	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	4	13750.0	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500.0	NaN	20544.0	Petrol	86.0	1.0	0	1300	3	1025
1432	1432	10845.0	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015
1433	1433	8500.0	NaN	17016.0	Petrol	86.0	0.0	0	1300	3	1015

9) Replace ???? in HP field with mean.

```
[11]: #9
df['HP'].fillna(df['HP'].mean(), inplace = True)
df
```

[11]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500.0	23.0	46986.0	Diesel	90.0	1.0	0	2000	3	1165
1	1	13750.0	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	2	13950.0	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	3	9750.0	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	4	13750.0	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500.0	NaN	20544.0	Petrol	86.0	1.0	0	1300	3	1025
1432	1432	10845.0	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015
1433	1433	8500.0	NaN	17016.0	Petrol	86.0	0.0	0	1300	3	1015

10) Impute blank values in FuelType with Mode.

```
[12]: #10
df['FuelType'].fillna(df['FuelType'].mode(), inplace = True)
df
```

[12]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500.0	23.0	46986.0	Diesel	90.0	1.0	0	2000	3	1165
1	1	13750.0	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	2	13950.0	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	3	9750.0	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	4	13750.0	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500.0	NaN	20544.0	Petrol	86.0	1.0	0	1300	3	1025
1432	1432	10845.0	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015
1433	1433	8500.0	NaN	17016.0	Petrol	86.0	0.0	0	1300	3	1015

11) Delete the rows with MetColor and Age as blank.

```
[13]: #11
df.dropna(subset=['MetColor' , 'Age'], how='all')
```

[13]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500.0	23.0	46986.0	Diesel	90.0	1.0	0	2000	3	1165
1	1	13750.0	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	2	13950.0	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	3	9750.0	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	4	13750.0	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500.0	NaN	20544.0	Petrol	86.0	1.0	0	1300	3	1025
1432	1432	10845.0	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015
1433	1433	8500.0	NaN	17016.0	Petrol	86.0	0.0	0	1300	3	1015

12) Replace ?? value in KM with Mean.

```
[14]: #12
df['HP'].fillna(df['HP'].mean(), inplace = True)
df
```

[14]:

	Unnamed: 0	Price	Age	KM	FuelType	HP	MetColor	Automatic	CC	Doors	Weight
0	0	13500.0	23.0	46986.0	Diesel	90.0	1.0	0	2000	3	1165
1	1	13750.0	23.0	72937.0	Diesel	90.0	1.0	0	2000	3	1165
2	2	13950.0	24.0	41711.0	Diesel	90.0	NaN	0	2000	3	1165
3	3	9750.0	26.0	48000.0	Diesel	90.0	0.0	0	2000	3	1165
4	4	13750.0	30.0	38500.0	Diesel	90.0	0.0	0	2000	3	1170
...	...	...	...	...	...	...	...	...	...	...	...
1431	1431	7500.0	NaN	20544.0	Petrol	86.0	1.0	0	1300	3	1025
1432	1432	10845.0	72.0	NaN	Petrol	86.0	0.0	0	1300	3	1015
1433	1433	8500.0	NaN	17016.0	Petrol	86.0	0.0	0	1300	3	1015

13) What is the mean, median and mode of KM field.

```
[15]: #13
print('Median =',df['KM'].median())
print('Mean =',df['KM'].mean())
print('Mode =',df['KM'].mode())
```

```
Median = 63634.0
Mean = 68647.23997185081
Mode = 0    1.0
Name: KM, dtype: float64
```

14) Categorise Age into AgeCat column with 0-10 NewCarCat, 11-20 MediumCarCat, 21- highest value – OldCarCat.

[27]:
```
#14
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

Age = df["Age"]
max_Age = df['Age'].max()

cond_list = [Age.between(0,10), Age.between(11,20) , Age.between(21,max_Age)]
AgeCat = ["NewCarCat", "MediumCarCat", "OldCarCat"]

df["AgeCat"] = np.select(cond_list, AgeCat)
df.AgeCat
```

```
162        NewCarCat
163     MediumCarCat
164     MediumCarCat
165     MediumCarCat
166     MediumCarCat
167     MediumCarCat
168     MediumCarCat
169        NewCarCat
170        NewCarCat
171        NewCarCat
172        NewCarCat
173        NewCarCat
174        NewCarCat
175        NewCarCat
176        NewCarCat
177        NewCarCat
178        NewCarCat
179        NewCarCat
180        NewCarCat
```

15) Create Dummy fields for FuelType.

[17]:
```
#15
pd.get_dummies(df)[['FuelType_CNG','FuelType_Diesel','FuelType_Petrol']]
```

[17]:

	FuelType_CNG	FuelType_Diesel	FuelType_Petrol
0	False	True	False
1	False	True	False
2	False	True	False
3	False	True	False
4	False	True	False
...	...	...	...
1431	False	False	True
1432	False	False	True
1433	False	False	True
1434	False	False	False
1435	False	False	True

1436 rows × 3 columns

Practical Assignment-2

1) Read Carfeatures.csv

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     filepath = r"C:\Users\maxim\Downloads\carfeatures.csv";
     df = pd.read_csv(filepath)
```

2) How many attributes are there?

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11914 entries, 0 to 11913
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Make               11914 non-null  object
 1   Model              11914 non-null  object
 2   Year               11914 non-null  int64
 3   Engine Fuel Type   11914 non-null  object
 4   Engine HP          11914 non-null  float64
 5   Engine Cylinders   11914 non-null  float64
 6   Transmission Type  11914 non-null  object
 7   Driven_Wheels      11914 non-null  object
 8   Number of Doors    11914 non-null  float64
 9   Market Category    11914 non-null  object
 10  Vehicle Size       11914 non-null  object
 11  Vehicle Style      11914 non-null  object
 12  highway MPG        11914 non-null  int64
 13  city mpg           11914 non-null  int64
 14  Popularity         11914 non-null  int64
 15  MSRP               11914 non-null  int64
dtypes: float64(3), int64(5), object(8)
```

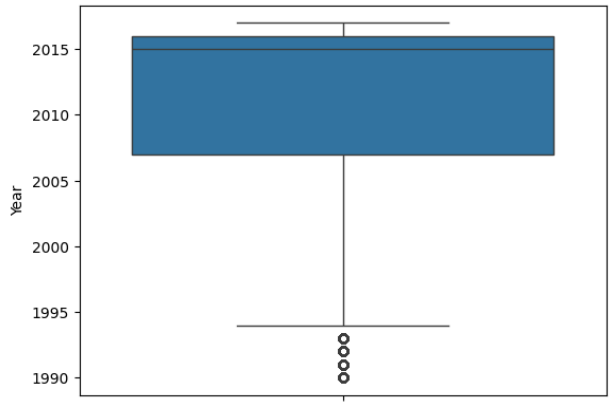
3) Fill the missing values, impute data, check and eliminate for outlier.

```
[2]: updated_df = df
     updated_df['Engine HP']=updated_df['Engine HP'].fillna(updated_df['Engine HP'].mean())
     updated_df['Engine Cylinders']=updated_df['Engine Cylinders'].fillna(updated_df['Engine Cylinders'].mean())
     updated_df['Number of Doors']=updated_df['Number of Doors'].fillna(updated_df['Number of Doors'].mean())
     updated_df["Market Category"].fillna("No Category", inplace = True)
     updated_df["Engine Fuel Type"].fillna("No type mentioned", inplace = True)
     updated_df.info()
```

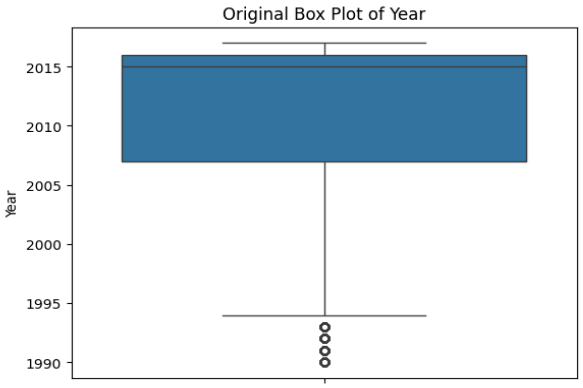
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11914 entries, 0 to 11913
Data columns (total 16 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Make               11914 non-null  object
 1   Model              11914 non-null  object
 2   Year               11914 non-null  int64
 3   Engine Fuel Type   11914 non-null  object
 4   Engine HP          11914 non-null  float64
 5   Engine Cylinders   11914 non-null  float64
 6   Transmission Type  11914 non-null  object
 7   Driven_Wheels      11914 non-null  object
 8   Number of Doors    11914 non-null  float64
 9   Market Category    11914 non-null  object
 10  Vehicle Size       11914 non-null  object
 11  Vehicle Style      11914 non-null  object
 12  highway MPG        11914 non-null  int64
 13  city mpg           11914 non-null  int64
 14  Popularity         11914 non-null  int64
 15  MSRP               11914 non-null  int64
dtypes: float64(3), int64(5), object(8)
memory usage: 1.5+ MB
```

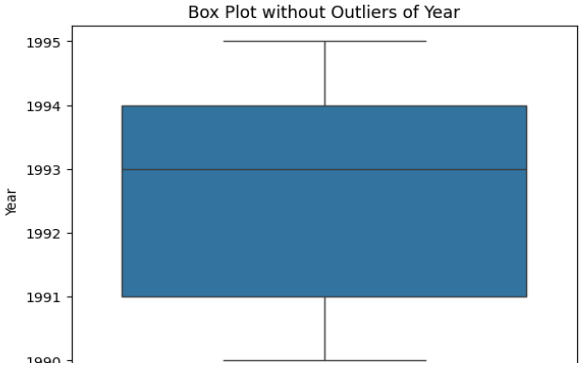
[3]:
```
#Checking for outliers
sns.boxplot(updated_df['Year'])
```

[3]:
```
<Axes: ylabel='Year'>
```



[4]:
```
def removal_box_plot(df, column, threshold):
    sns.boxplot(df[column])
    plt.title(f'Original Box Plot of {column}')
    plt.show()

    removed_outliers = df[df[column] <= threshold]

    sns.boxplot(removed_outliers[column])
    plt.title(f'Box Plot without Outliers of {column}')
    plt.show()
    return removed_outliers


threshold_value = 1995

no_outliers = removal_box_plot(updated_df, 'Year', threshold_value)
```





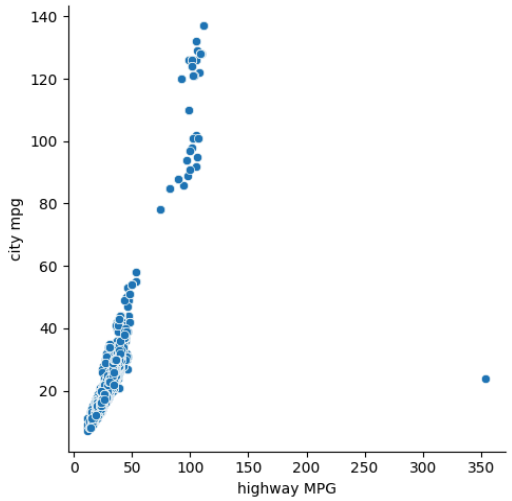
4) How many unique values are there in popularity column?

```
[11]: df.Popularity.nunique()

[11]: 48
```

5) Using matplot/seaborn

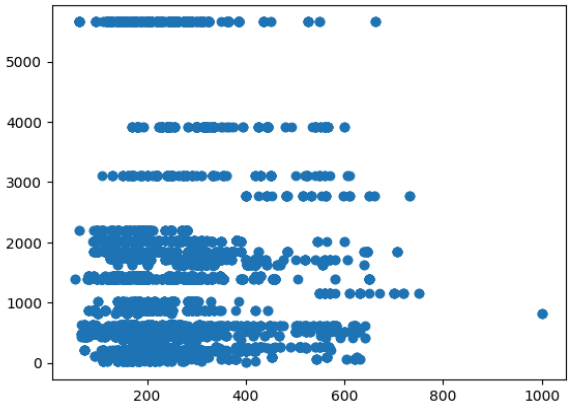
a) Draw lineplot using relplot function of seaborn between highway MPG and city mpg

```
[12]: sns.relplot(x='highway MPG', y='city mpg', data=updated_df)

[12]: <seaborn.axisgrid.FacetGrid at 0x1c2b5740170>
```



b) Draw scatter plot between Engine HP and Popularity

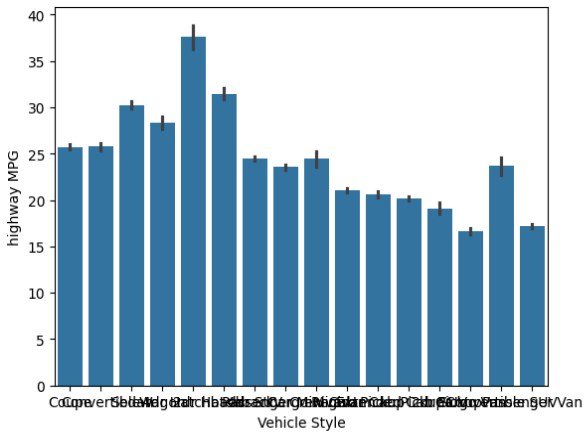
```
[13]: plt.scatter(x='Engine HP', y='Popularity', data=updated_df)

[13]: <matplotlib.collections.PathCollection at 0x1c2b5a986b0>
```



c) Draw barplot between Vehicle Style and highway MPG using seaborn function

```
[14]: sns.barplot(updated_df, x="Vehicle Style", y="highway MPG")
```

```
[14]: <Axes: xlabel='Vehicle Style', ylabel='highway MPG'>
```



d) Draw piechart on the basis of Market Category, average of MSRP.

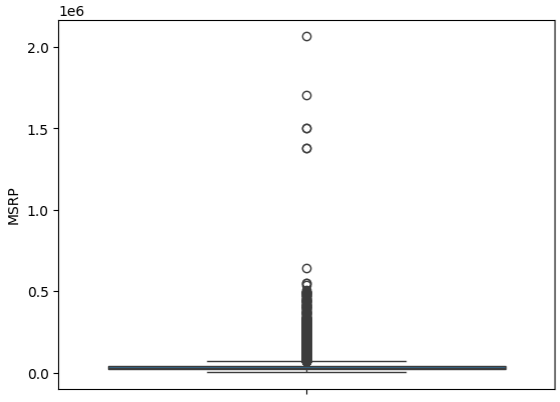
```
[26]: plt.pie(updated_df['MSRP'], labels = updated_df['Market Category'])
```

```
[26]: ([<matplotlib.patches.Wedge at 0x1c2ee06e9f0>,
  <matplotlib.patches.Wedge at 0x1c2ee0d4b00>,
  <matplotlib.patches.Wedge at 0x1c2eddcb530>,
  <matplotlib.patches.Wedge at 0x1c2e00ee030>,
  <matplotlib.patches.Wedge at 0x1c2e00ee240>,
  <matplotlib.patches.Wedge at 0x1c2e00efa70>,
  <matplotlib.patches.Wedge at 0x1c2edddac00>,
  <matplotlib.patches.Wedge at 0x1c2edd8170>,
  <matplotlib.patches.Wedge at 0x1c2edd85f0>,
  <matplotlib.patches.Wedge at 0x1c2edd9430>,
  <matplotlib.patches.Wedge at 0x1c2edd91c0>,
  <matplotlib.patches.Wedge at 0x1c2edd85c0>,
  <matplotlib.patches.Wedge at 0x1c2edd8c20>,
  <matplotlib.patches.Wedge at 0x1c2edd99a0>,
  <matplotlib.patches.Wedge at 0x1c2eddda030>,
  <matplotlib.patches.Wedge at 0x1c2eddda4b0>,
  <matplotlib.patches.Wedge at 0x1c2edddaf00>,
  <matplotlib.patches.Wedge at 0x1c2ededdb440>,
```

e) Draw boxplot MSRP

```
[16]: sns.boxplot(updated_df['MSRP'])
```

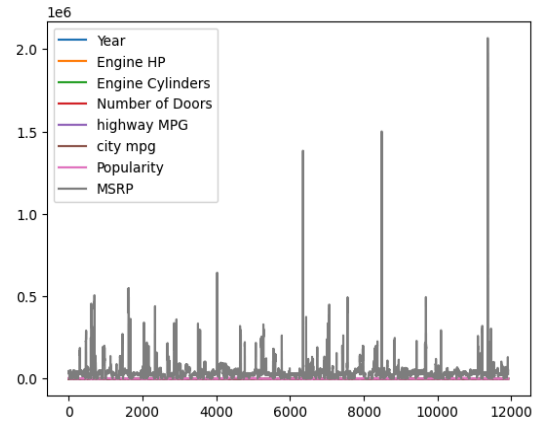
```
[16]: <Axes: ylabel='MSRP'>
```



5) Using pandas draw the above four plots.

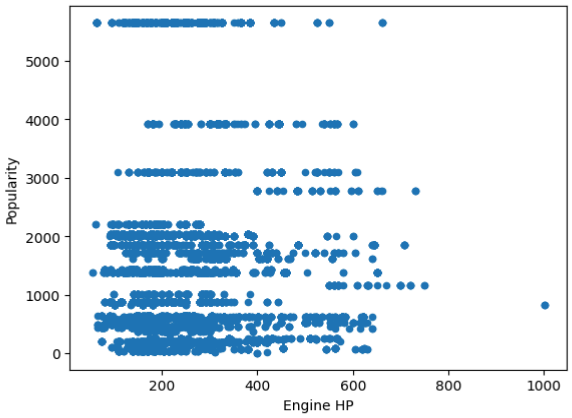
```
[17]: updated_df.plot.line()
```

```
[17]: <Axes: >
```



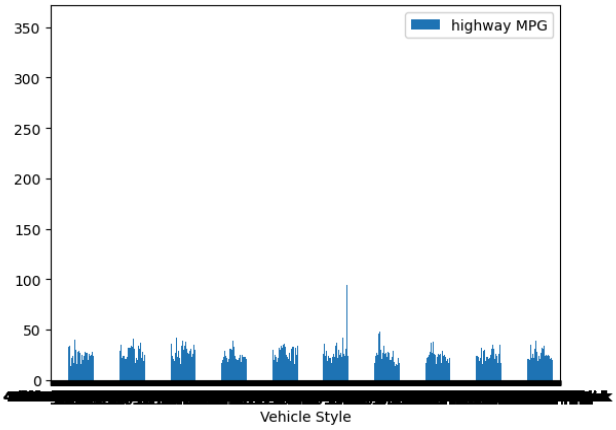

```
[18]: updated_df.plot.scatter(x='Engine HP',y='Popularity')
```

```
[18]: <Axes: xlabel='Engine HP', ylabel='Popularity'>
```



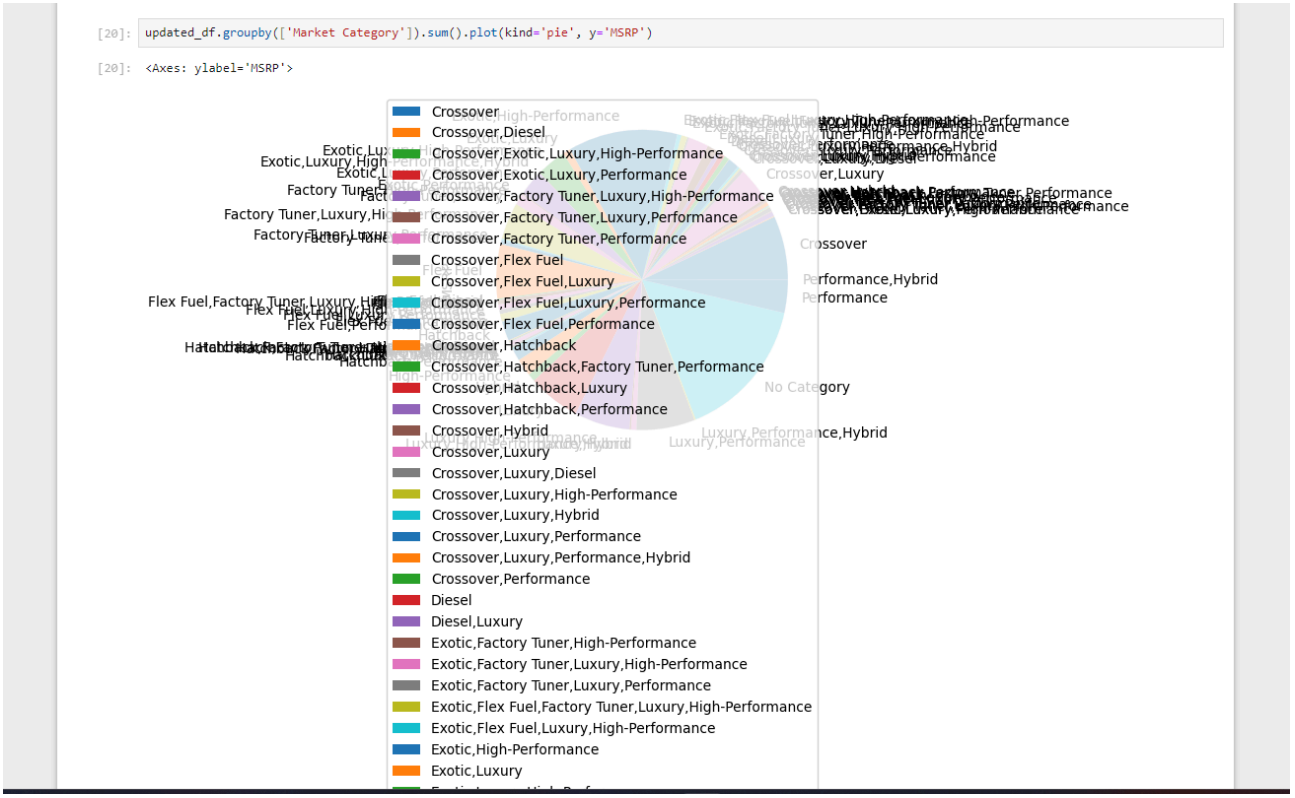

```
[19]: updated_df.plot.bar(x='Vehicle Style', y='highway MPG',rot=0)
```

```
[19]: <Axes: xlabel='Vehicle Style'>
```



```
[20]: updated_df.groupby(['Market Category']).sum().plot(kind='pie', y='MSRP')
```

```
[20]: <Axes: ylabel='MSRP'>
```



Practical Assignment 3

1) Perform one sample z-test in python for the given problem:

Suppose the IQ in a certain population is normally distributed with a mean of μ = 100 and standard deviation of σ = 15.

A researcher wants to know if a new drug affects IQ levels, so he recruits 20 patients to try it and records their IQ levels.

data = [88, 92, 94, 94, 96, 97, 97, 97, 99, 99,

105, 109, 109, 109, 110, 112, 112, 113, 114, 115]

```
import numpy as np
from statsmodels.stats.weightstats import ztest

data = [88, 92, 94, 94, 96, 97, 97, 97, 99, 99,
        105, 109, 109, 109, 110, 112, 112, 113, 114, 115]
population_mean = 100
population_std = 15
```

```
[3] z_score, p_value = ztest(x1=data, value=population_mean, alternative='two-sided')

    print("Z-score:", z_score)
    print("P-value:", p_value)
```

```
Z-score: 1.5976240527147705
P-value: 0.1101266701438426
```

2) Perform One Sample t-Test in Python for the given problem

Suppose a botanist wants to know if the mean height of a certain species of plant is equal to 15 inches. She collects a random sample of 12 plants and records each of their heights in inches.

data = [14, 14, 16, 13, 12, 17, 15, 14, 15, 13, 15, 14]

Explain the result of t-test on the above sample.

```
import numpy as np
from statsmodels.stats.weightstats import ztest

data = [14, 14, 16, 13, 12, 17, 15, 14, 15, 13, 15, 14]
population_mean = 15

# Perform one-sample z-test
z_score, p_value = ztest(x1=data, value=population_mean)

print("Z-score:", z_score)
print("P-value:", p_value)
```

```
Z-score: -1.6848470783484626
P-value: 0.09201807918461961
```

Practical Assignment 4

1) Create TensileStrength.xlsx file with the given data

	concentration5	concentration10	concentration15	concentration20
0	7	12	14	19
1	8	17	18	25
2	15	13	19	22
3	11	18	17	23
4	9	19	16	18
5	10	15	18	20

2) Read the excel file in dataframe, use the melt command of pandas to pivot the table
3) Run one way anova to check whether the null hypothesis is rejected or not rejected.
4) If null hypothesis is rejected test for Posthoc test (Tukey's) and discuss the result.

```python
import pandas as pd
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import MultiComparison
data = {
    'concentration5': [7, 8, 15, 11, 9, 10],
    'concentration10': [12, 17, 13, 18, 19, 15],
    'concentration15': [14, 18, 19, 17, 16, 18],
    'concentration20': [19, 25, 22, 23, 18, 20]
}
df = pd.DataFrame(data)
# Run one-way ANOVA
f_statistic, p_value = f_oneway(df['concentration5'], df['concentration10'], df['concentration15'], df['concentration20'])

print("One-way ANOVA results:")
print("F-statistic:", f_statistic)
print("P-value:", p_value)

# Tukey's post hoc test if null hypothesis is rejected
if p_value < 0.05:
    # Combine all data into one array
    data_array = df['concentration5'].to_numpy(), df['concentration10'].to_numpy(), df['concentration15'].to_numpy(), df['concentration20'].to_numpy()

    #  Tukey's HSD test
    mc = MultiComparison(df.melt(var_name='Concentration', value_name='TensileStrength')['TensileStrength'], df.melt(var_name='Concentration', value_name='TensileStrength')['Concentration'])
    tukey_results = mc.tukeyhsd()

    print("\nTukey's HSD test results:")
    print(tukey_results)
```

```
One-way ANOVA results:
F-statistic: 19.605206999573184
P-value: 3.5925782584743027e-06

Tukey's HSD test results:
          Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====================================================================
     group1          group2     meandiff p-adj   lower    upper  reject
---------------------------------------------------------------------
concentration10 concentration15   1.3333 0.8022  -2.7892  5.4559  False
concentration10 concentration20      5.5 0.0066   1.3774  9.6226   True
concentration10  concentration5  -5.6667 0.0051  -9.7892 -1.5441   True
concentration15 concentration20   4.1667  0.047   0.0441  8.2892   True
concentration15  concentration5     -7.0 0.0007 -11.1226 -2.8774   True
concentration20  concentration5 -11.1667    0.0 -15.2892 -7.0441   True
---------------------------------------------------------------------
```

Practical Assignment 5

1) Create a database in SQLite “College” and create a table Student with five fields : Name, EnrolmentNo, Percentage, Course, Batch.
2) Create an interface in tkinter to insert new data in Student table, through Entry text, radio buttons and on click of “’Add” button, the record should be added to the table.
3) Create “Display” and “Display All”’ buttons, and on click of button- display the current record in above text fields and on click of “display all” button , show all the records in Listbox.

```
import tkinter as tk
import sqlite3

def add_student():
    name = name_entry.get()
    enrolment_no = enrolment_entry.get()
    percentage = float(percentage_entry.get())
    course = course_var.get()
    batch = batch_var.get()

    conn = sqlite3.connect('College.db')
    cursor = conn.cursor()

    cursor.execute('''INSERT INTO Student (Name, EnrolmentNo, Percentage, Course, Batch)
                      VALUES (?, ?, ?, ?, ?)''', (name, enrolment_no, percentage, course, batch))

    conn.commit()
    conn.close()
    clear_entries()

def clear_entries():
    name_entry.delete(0, tk.END)
    enrolment_entry.delete(0, tk.END)
    percentage_entry.delete(0, tk.END)
    course_var.set("B.Tech")
    batch_var.set("2023")

# Tkinter window
window = tk.Tk()
window.title("Add Student - College Management System")

# Student Name
tk.Label(window, text="Name:").grid(row=0, column=0)
```

```
name_entry = tk.Entry(window)
name_entry.grid(row=0, column=1)

# Enrolment No
tk.Label(window, text="Enrolment No:").grid(row=1, column=0)
enrolment_entry = tk.Entry(window)
enrolment_entry.grid(row=1, column=1)

# Percentage
tk.Label(window, text="Percentage:").grid(row=2, column=0)
percentage_entry = tk.Entry(window)
percentage_entry.grid(row=2, column=1)

# Course
tk.Label(window, text="Course:").grid(row=3, column=0)
course_var = tk.StringVar()
course_var.set("B.Tech")
course_radio1 = tk.Radiobutton(window, text="B.Tech", variable=course_var,
value="B.Tech")
course_radio1.grid(row=3, column=1)
course_radio2 = tk.Radiobutton(window, text="M.Tech", variable=course_var,
value="M.Tech")
course_radio2.grid(row=3, column=2)

# Batch
tk.Label(window, text="Batch:").grid(row=4, column=0)
batch_var = tk.StringVar()
batch_var.set("2023")
batch_radio1 = tk.Radiobutton(window, text="2022", variable=batch_var, value="2022")
batch_radio1.grid(row=4, column=1)
batch_radio2 = tk.Radiobutton(window, text="2023", variable=batch_var, value="2023")
batch_radio2.grid(row=4, column=2)

# Add Button
tk.Button(window, text="Add", command=add_student).grid(row=5, column=0,
columnspan=3)

window.mainloop()
```

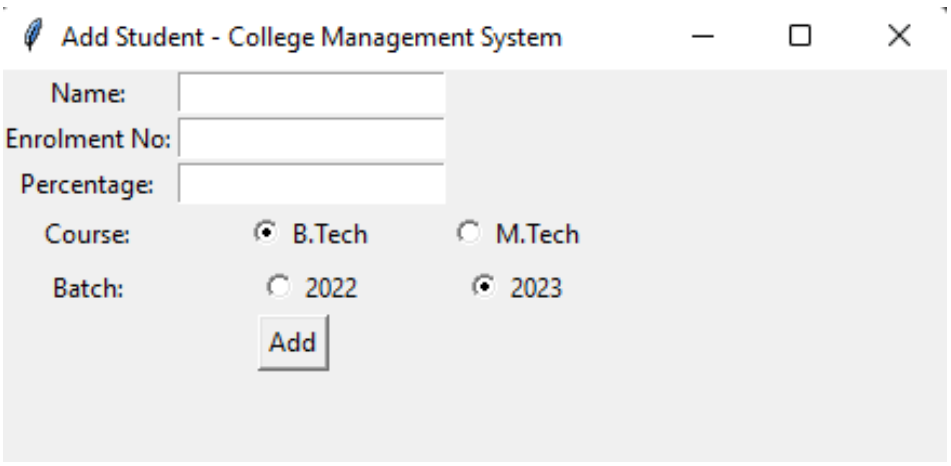
(3)

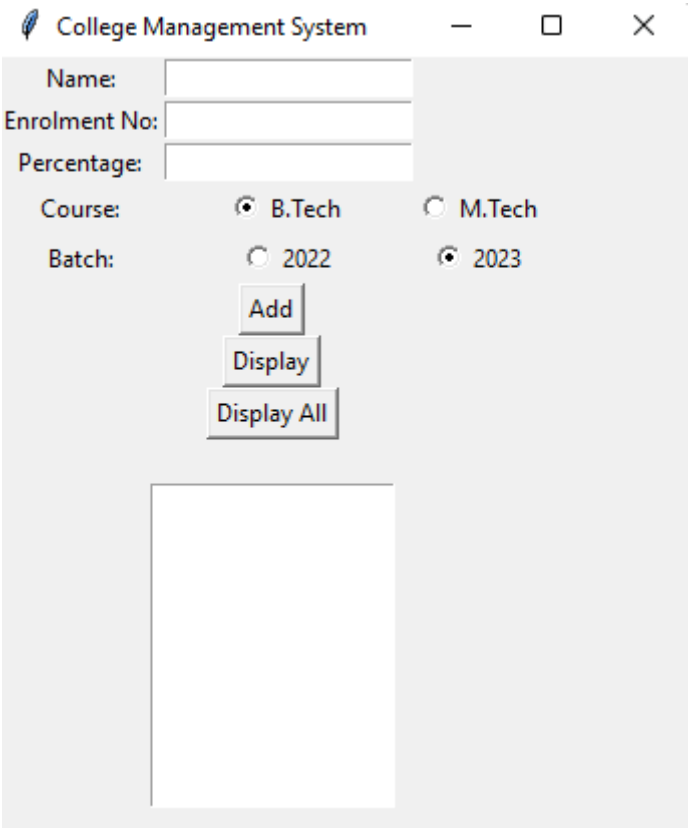
```
        import tkinter as tk
        import sqlite3

        def add_student():
```

```
    name = name_entry.get()
    enrolment_no = enrolment_entry.get()
    percentage = float(percentage_entry.get())
    course = course_var.get()
    batch = batch_var.get()

    conn = sqlite3.connect('College.db')
    cursor = conn.cursor()

    cursor.execute('''INSERT INTO Student (Name, EnrolmentNo, Percentage, Course,
Batch)
                      VALUES (?, ?, ?, ?, ?)''', (name, enrolment_no, percentage, course,
batch))

    conn.commit()
    conn.close()
    clear_entries()

def clear_entries():
    name_entry.delete(0, tk.END)
    enrolment_entry.delete(0, tk.END)
    percentage_entry.delete(0, tk.END)
    course_var.set("B.Tech")
    batch_var.set("2023")

def display_student():
    name = name_entry.get()
    conn = sqlite3.connect('College.db')
    cursor = conn.cursor()

    cursor.execute('''SELECT * FROM Student WHERE Name = ?''', (name,))
    student = cursor.fetchone()

    result_label.config(text=f"Student Details:\n\nName: {student[1]}\nEnrolment No:
{student[2]}\nPercentage: {student[3]}\nCourse: {student[4]}\nBatch: {student[5]}"
if student else "Student not found")

    conn.close()

def display_all_students():
    conn = sqlite3.connect('College.db')
    cursor = conn.cursor()
```

```
    cursor.execute('"SELECT * FROM Student"')
    students = cursor.fetchall()

    listbox.delete(0, tk.END)
    for student in students:
        listbox.insert(tk.END, f"Name: {student[1]}, Enrolment No: {student[2]},
Percentage: {student[3]}, Course: {student[4]}, Batch: {student[5]}")

    conn.close()

# Tkinter window
window = tk.Tk()
window.title("College Management System")

# Name
tk.Label(window, text="Name:").grid(row=0, column=0)
name_entry = tk.Entry(window)
name_entry.grid(row=0, column=1)

# Enrolment No
tk.Label(window, text="Enrolment No:").grid(row=1, column=0)
enrolment_entry = tk.Entry(window)
enrolment_entry.grid(row=1, column=1)

# Percentage
tk.Label(window, text="Percentage:").grid(row=2, column=0)
percentage_entry = tk.Entry(window)
percentage_entry.grid(row=2, column=1)

# Course
tk.Label(window, text="Course:").grid(row=3, column=0)
course_var = tk.StringVar()
course_var.set("B.Tech")
course_radio1 = tk.Radiobutton(window, text="B.Tech", variable=course_var,
value="B.Tech")
course_radio1.grid(row=3, column=1)
course_radio2 = tk.Radiobutton(window, text="M.Tech", variable=course_var,
value="M.Tech")
course_radio2.grid(row=3, column=2)

# Batch
tk.Label(window, text="Batch:").grid(row=4, column=0)
batch_var = tk.StringVar()
```

```
batch_var.set("2023")
batch_radio1 = tk.Radiobutton(window, text="2022", variable=batch_var,
value="2022")
batch_radio1.grid(row=4, column=1)
batch_radio2 = tk.Radiobutton(window, text="2023", variable=batch_var,
value="2023")
batch_radio2.grid(row=4, column=2)

# Add Button
tk.Button(window, text="Add", command=add_student).grid(row=5, column=0,
columnspan=3)

# Display Button
tk.Button(window, text="Display", command=display_student).grid(row=6,
column=0, columnspan=3)

# Display All Button
tk.Button(window, text="Display All", command=display_all_students).grid(row=7,
column=0, columnspan=3)

# Result Label
result_label = tk.Label(window, text="")
result_label.grid(row=8, column=0, columnspan=3)

# Listbox for Display All
listbox = tk.Listbox(window)
listbox.grid(row=9, column=0, columnspan=3)

window.mainloop()
```




College Management System
Name:
Enrolment No:
Percentage:
Course:
B.Tech
M.Tech
Batch:
2022
2023
Add
Display
Display All

Syllabus Practicals

1. Write a program to create a DataFrame have E-commerce data and perform selection of row/ column using loc() and iloc()

```
# Sample E-commerce data
data = {
    'Order_ID': [101, 102, 103, 104, 105],
    'Product': ['Shoes', 'Shirt', 'Pants', 'Hat', 'Socks'],
    'Price': [29.99, 14.99, 24.99, 9.99, 4.99],
    'Quantity': [2, 1, 3, 2, 4],
    'Total': [59.98, 14.99, 74.97, 19.98, 19.96]
}
df = pd.DataFrame(data)
# Selecting rows and columns using loc[]
print("Using loc[]:")
print(df.loc[1:3, ['Order_ID', 'Product', 'Total']])
# Selecting rows and columns using iloc[]
print("\nUsing iloc[]:")
print(df.iloc[1:4, [0, 1, 4]])
```

```
Using loc[]:
   Order_ID Product  Total
1       102   Shirt  14.99
2       103   Pants  74.97
3       104     Hat  19.98

Using iloc[]:
   Order_ID Product  Total
1       102   Shirt  14.99
2       103   Pants  74.97
3       104     Hat  19.98
```

2. Create a Series object S5 containing numbers. Write a program to store the square of the series values in object S6. Display S6's values which are >15.

```
# Create Series S5 containing numbers
S5 = pd.Series([1, 2, 3, 4, 5])
# Store the square of S5 values in Series S6
S6 = S5 ** 2
# Display S6's values that are greater than 15
result = S6[S6 > 15]
print(result)
```

```
3    16
4    25
dtype: int64
```

3. Write a program to fill all missing values in a DataFrame with zero.

```
import numpy as np
# Sample DataFrame with missing values
data = {'A': [1, np.nan, 3, 4, np.nan], 'B': [5, 6, np.nan, 8, 9]}
df = pd.DataFrame(data)

# Fill missing values with zero
df_filled = df.fillna(0)

print("Original DataFrame:")
print(df)
print("\nDataFrame with missing values filled with zero:")
print(df_filled)
```

```
Original DataFrame:
     A    B
0  1.0  5.0
1  NaN  6.0
2  3.0  NaN
3  4.0  8.0
4  NaN  9.0

DataFrame with missing values filled with zero:
     A    B
0  1.0  5.0
1  0.0  6.0
2  3.0  0.0
3  4.0  8.0
4  0.0  9.0
```

4. Program for combining DataFrames using concat(), join(),merge()

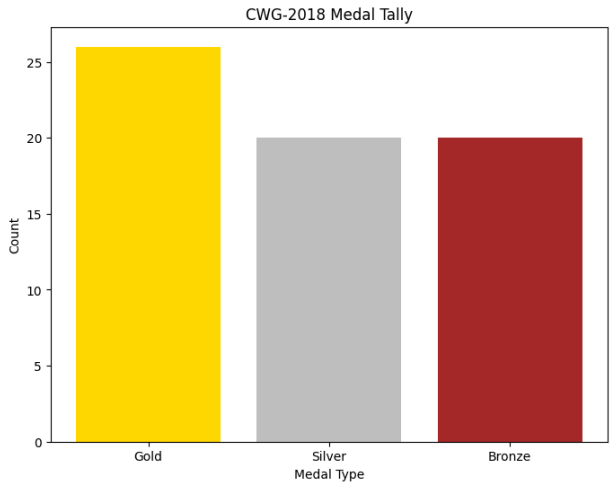
```
# Sample data
data1 = {'A': [1, 2, 3], 'B': [4, 5, 6]}
data2 = {'A': [7, 8, 9], 'B': [10, 11, 12]}

df1 = pd.DataFrame(data1)
df2 = pd.DataFrame(data2)

# Using concat
concatenated = pd.concat([df1, df2], ignore_index=True)
print("Concatenated DataFrame:")
print(concatenated)

# Using join (merge on index)
joined = df1.join(df2, lsuffix='_left', rsuffix='_right')
print("\nJoined DataFrame:")
print(joined)

# Using merge (merge on a common column)
merged = pd.merge(df1, df2, on='A')
print("\nMerged DataFrame:")
print(merged)
```

```
Concatenated DataFrame:
   A   B
0  1   4
1  2   5
2  3   6
3  7  10
4  8  11
5  9  12

Joined DataFrame:
   A_left  B_left  A_right  B_right
0       1       4        7       10
1       2       5        8       11
2       3       6        9       12

Merged DataFrame:
Empty DataFrame
Columns: [A, B_x, B_y]
Index: []
```

5. Write a program to draw bar graph for the following data for the Medal tally of CWG-2018:-

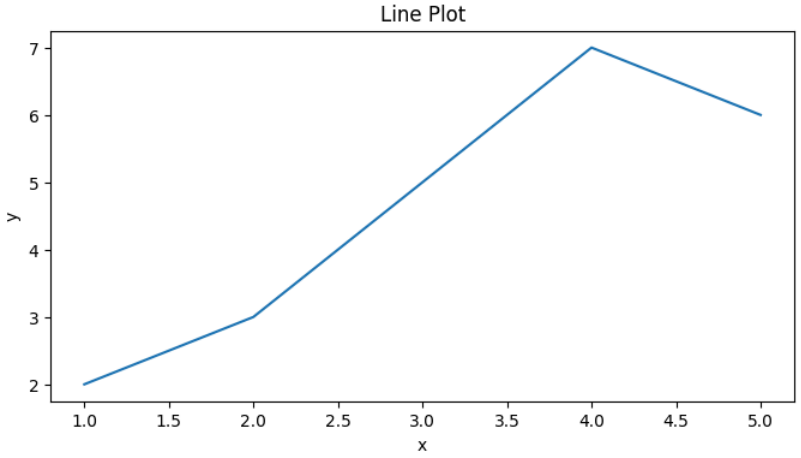
Gold	Silver	Bronze	Total
26	20	20	66

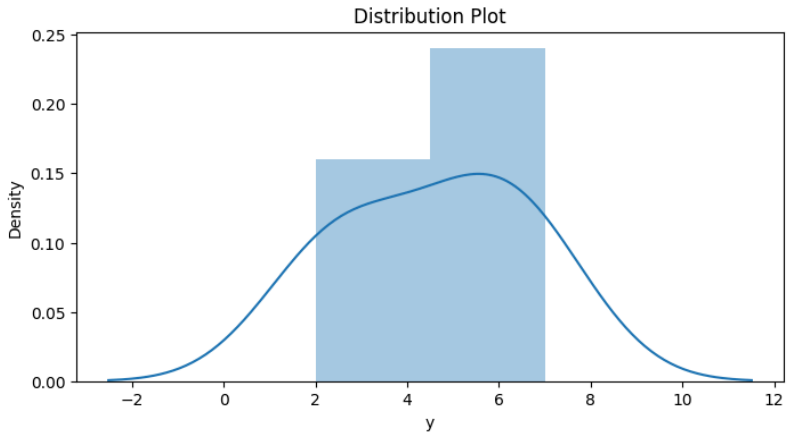
```python
# Data for the medal tally
medals = ['Gold', 'Silver', 'Bronze']
counts = [26, 20, 20]

# Creating the bar graph
plt.figure(figsize=(8, 6))
plt.bar(medals, counts, color=['gold', 'silver', 'brown'])
plt.xlabel('Medal Type')
plt.ylabel('Count')
plt.title('CWG-2018 Medal Tally')
plt.show()
```

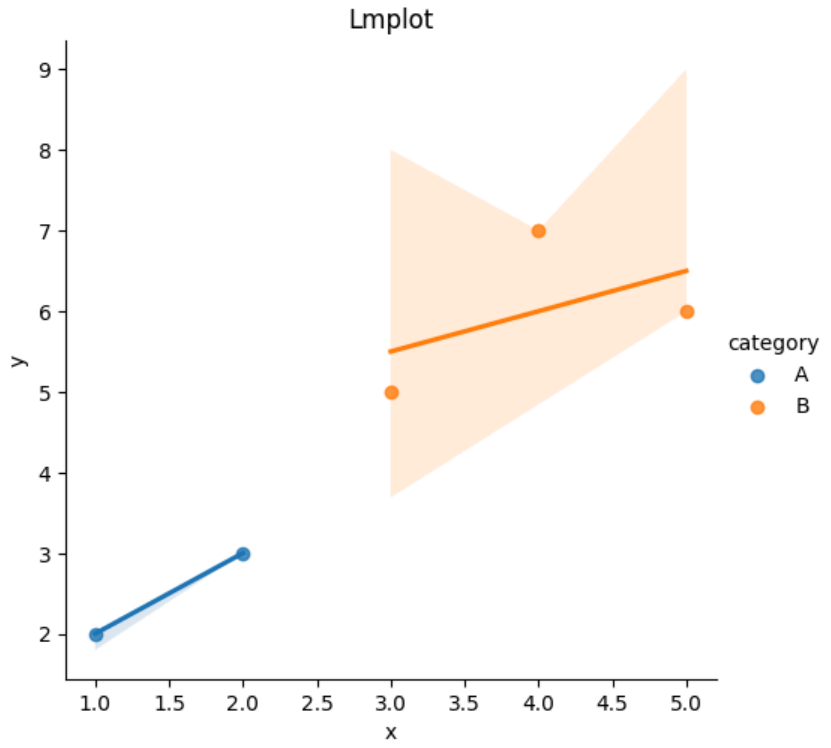


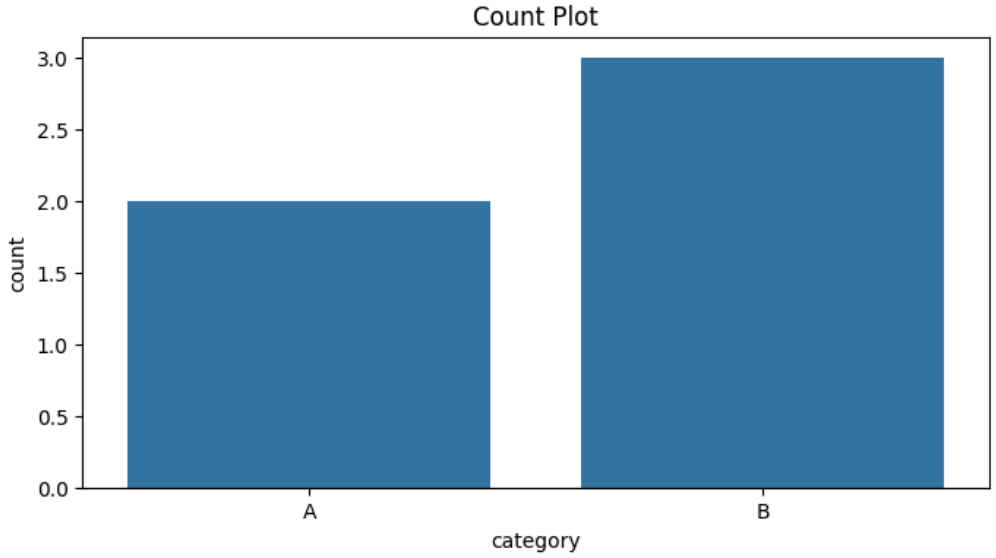
6. Implement Line lot, Dist lot, Lmplot, Count plot using Seaborn library

```
import seaborn as sns
# Sample data
data = pd.DataFrame({
    'x': [1, 2, 3, 4, 5],
    'y': [2, 3, 5, 7, 6],
    'category': ['A', 'A', 'B', 'B', 'B']
})
# Line plot
plt.figure(figsize=(8, 4))
sns.lineplot(x='x', y='y', data=data)
plt.title('Line Plot')
plt.show()
# Dist plot
plt.figure(figsize=(8, 4))
sns.distplot(data['y'])
plt.title('Distribution Plot')
plt.show()
# Lmplot
plt.figure(figsize=(8, 4))
sns.lmplot(x='x', y='y', data=data, hue='category')
plt.title('Lmplot')
plt.show()
# Count plot
plt.figure(figsize=(8, 4))
sns.countplot(x='category', data=data)
plt.title('Count Plot')
plt.show()
```









7. Create a DataFrame that stores aid (Toys,books,uniform,shoes) by NGOs for different states. Write a program to display the aid for:-

(a) Books and Uniforms only

(b) Shoes only

```
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import f_oneway
```

```python
# Create the DataFrame
data = {
    'NGO': ['NGO1', 'NGO2', 'NGO3', 'NGO4'],
    'State': ['State1', 'State2', 'State3', 'State4'],
    'Toys': [100, 150, 200, 120],
    'Books': [300, 250, 350, 200],
    'Uniform': [200, 180, 220, 150],
    'Shoes': [150, 120, 100, 180]
}

aid = pd.DataFrame(data)

# Display aid for Books and Uniforms only
print("Aid for Books and Uniforms only:")
print(aid[['NGO', 'State', 'Books', 'Uniform']])

# Display aid for Shoes only
print("\nAid for Shoes only:")
print(aid[['NGO', 'State', 'Shoes']])
```

```
Aid for Books and Uniforms only:
    NGO   State  Books  Uniform
0  NGO1  State1    300      200
1  NGO2  State2    250      180
2  NGO3  State3    350      220
3  NGO4  State4    200      150

Aid for Shoes only:
    NGO   State  Shoes
0  NGO1  State1    150
1  NGO2  State2    120
2  NGO3  State3    100
3  NGO4  State4    180
```

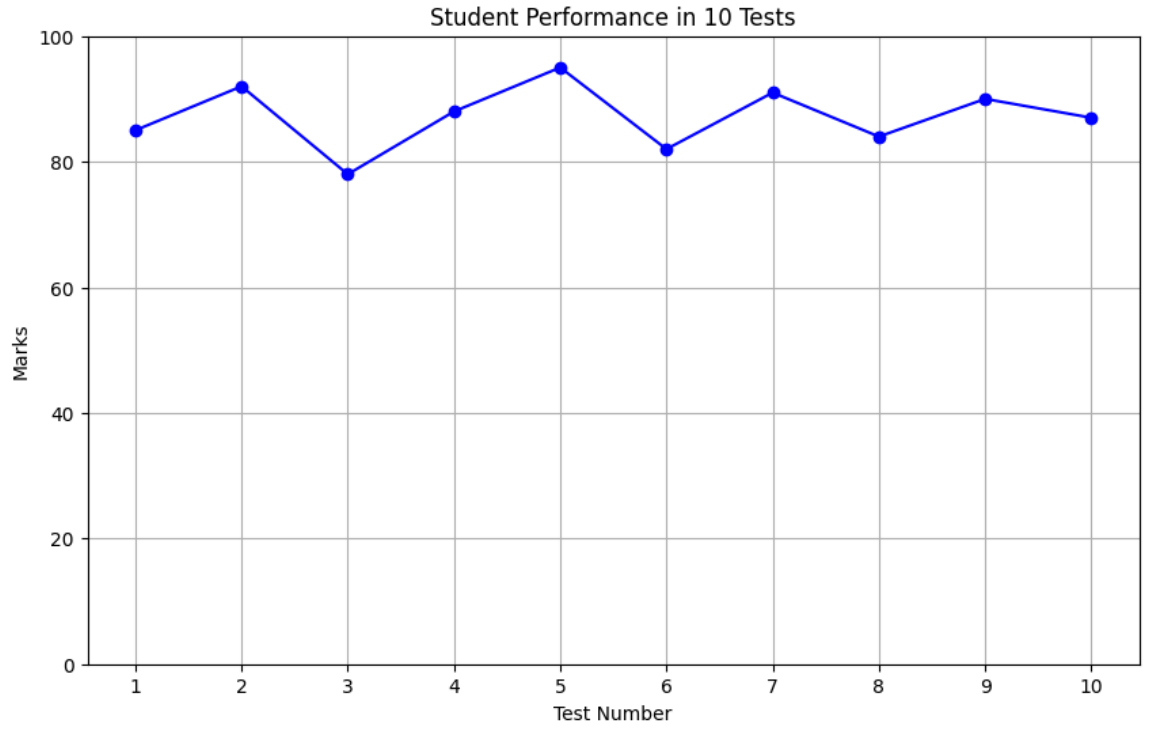
8. Create a DataFrame ndf have Name, Gender, Position, City, Age, Projects. Write a program to summarize how many projects are being handled by each position for each city? Use pivot()

```python
# Create the DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Gender': ['Female', 'Male', 'Male', 'Male', 'Female'],
    'Position': ['Manager', 'Developer', 'Manager', 'Developer', 'Manager'],
    'City': ['New York', 'San Francisco', 'New York', 'San Francisco', 'New York'],
    'Age': [30, 35, 40, 25, 45],
    'Projects': [3, 2, 4, 1, 5]
}

ndf = pd.DataFrame(data)

# Summarize the number of projects for each position in each city
summary = ndf.pivot_table(index='Position', columns='City', values='Projects', aggfunc='sum', fill_value=0)

print(summary)
```

```
City       New York  San Francisco
Position
Developer         0              3
Manager          12              0
```

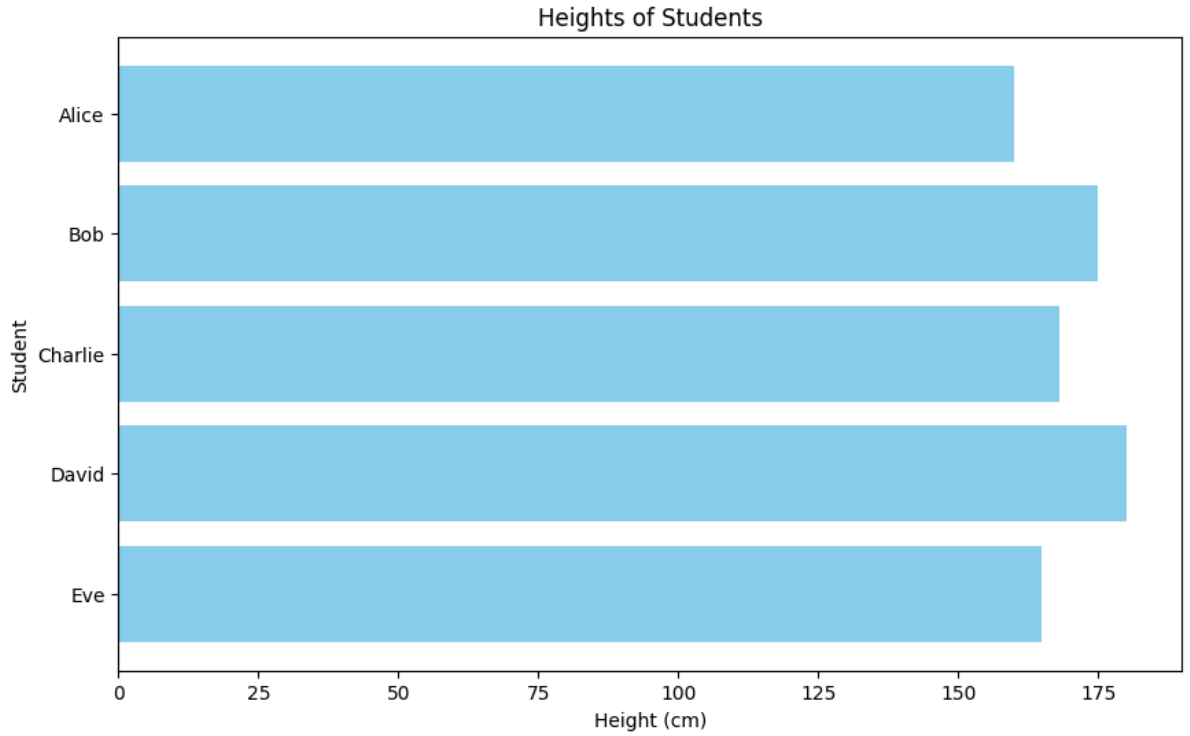
9. Marks is a list that stores marks of a student in a 10 unit test. Write a program to plot Line charts for the student's performance in these 10 tests.

```
marks = [85, 92, 78, 88, 95, 82, 91, 84, 90, 87]
# Test numbers
tests = range(1, 11)
# Plotting the line chart
plt.figure(figsize=(10, 6))
plt.plot(tests, marks, marker='o', color='b', linestyle='-')
plt.title('Student Performance in 10 Tests')
plt.xlabel('Test Number')
plt.ylabel('Marks')
plt.grid(True)
plt.xticks(tests)
plt.ylim(0, 100)
plt.show()
```



10. Write a program to plot a horizontal bar chart from the height of some students.

```
students = ['Alice', 'Bob', 'Charlie', 'David', 'Eve']
heights = [160, 175, 168, 180, 165]

# Plotting the horizontal bar chart
plt.figure(figsize=(10, 6))
plt.barh(students, heights, color='skyblue')
plt.xlabel('Height (cm)')
plt.ylabel('Student')
plt.title('Heights of Students')
plt.xlim(0, max(heights) + 10)  # Set the x-axis limit slightly larger than the maximum height
plt.gca().invert_yaxis()  # Invert y-axis to display the student with the highest height at the top
plt.show()
```



11. Write a program to implement ANNOVA.

```
group1 = [10, 15, 20, 25, 30]
group2 = [12, 18, 24, 30, 36]
group3 = [8, 12, 16, 20, 24]

# Perform one-way ANOVA
f_statistic, p_value = f_oneway(group1, group2, group3)

# Print the results
print("F statistic:", f_statistic)
print("P-value:", p_value)

# Interpret the results
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis, there is a significant difference between the group means.")
else:
    print("Fail to reject the null hypothesis, there is no significant difference between the group means.")
```

```
F statistic: 1.2467532467532467
P-value: 0.3221414874920264
Fail to reject the null hypothesis, there is no significant difference between the group means.
```

12. Write a program to show correlation between two randomly generated numbers.

```
import numpy as np

# Generating two random integers
np.random.seed(0)  # For reproducibility
x = np.random.randint(1, 100)  # Random integer for x-axis
y = np.random.randint(1, 100)  # Random integer for y-axis

# Calculating correlation coefficient
correlation = 1.0 if x == y else -1.0

print("Random Number (x):", x)
print("Random Number (y):", y)
print("Correlation coefficient:", correlation)
```

Output:

```
PS D:\Desktop> python -u "d:\Desktop\test.py"
Random Number (x): 45
Random Number (y): 48
Correlation coefficient: -1.0
```

13. Write a program to implement Covariance.

```
import numpy as np

def covariance(x, y):
    n = len(x)
    mean_x = np.mean(x)
    mean_y = np.mean(y)
    cov = sum((x[i] - mean_x) * (y[i] - mean_y) for i in range(n)) / (n - 1)
    return cov

# Example usage
x = [1, 2, 3, 4, 5]
y = [5, 4, 3, 2, 1]

cov = covariance(x, y)
print("Covariance:", cov)
```

Output:

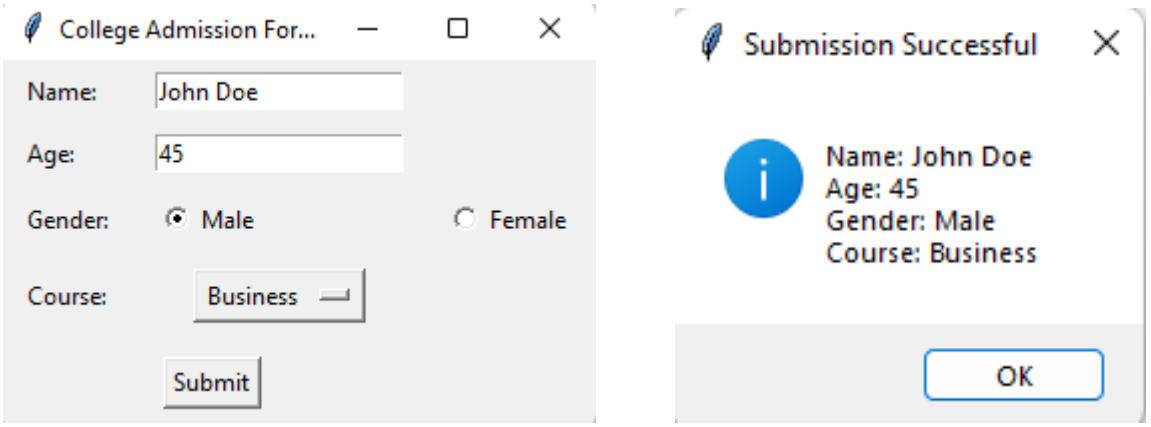
```
PS D:\Desktop> python -u "d:\Desktop\test.py"
Covariance: -2.5
```

14. Create a GUI based form for admission purposes for your college.

```
import tkinter as tk
from tkinter import messagebox

def submit_form():
    # Retrieve values from the entry fields
    name = entry_name.get()
    age = entry_age.get()
    gender = gender_var.get()
    course = course_var.get()

    # Display submitted information
    messagebox.showinfo("Submission Successful",
                        f"Name: {name}\nAge: {age}\nGender: {gender}\nCourse: {course}")

# Create main window
root = tk.Tk()
root.title("College Admission Form")

# Label and Entry for Name
label_name = tk.Label(root, text="Name:")
label_name.grid(row=0, column=0, padx=10, pady=5, sticky="w")
entry_name = tk.Entry(root)
entry_name.grid(row=0, column=1, padx=10, pady=5)

# Label and Entry for Age
label_age = tk.Label(root, text="Age:")
label_age.grid(row=1, column=0, padx=10, pady=5, sticky="w")
entry_age = tk.Entry(root)
entry_age.grid(row=1, column=1, padx=10, pady=5)

# Radio Buttons for Gender
label_gender = tk.Label(root, text="Gender:")
label_gender.grid(row=2, column=0, padx=10, pady=5, sticky="w")
gender_var = tk.StringVar()
gender_var.set("Male")
radio_male = tk.Radiobutton(root, text="Male", variable=gender_var, value="Male")
radio_male.grid(row=2, column=1, padx=10, pady=5, sticky="w")
radio_female = tk.Radiobutton(root, text="Female", variable=gender_var, value="Female")
radio_female.grid(row=2, column=2, padx=10, pady=5, sticky="w")

# Dropdown Menu for Course
```

```
label_course = tk.Label(root, text="Course:")
label_course.grid(row=3, column=0, padx=10, pady=5, sticky="w")
courses = ["Engineering", "Medical", "Business", "Arts"]
course_var = tk.StringVar()
course_var.set(courses[0])
dropdown_course = tk.OptionMenu(root, course_var, *courses)
dropdown_course.grid(row=3, column=1, padx=10, pady=5)

# Submit Button
submit_button = tk.Button(root, text="Submit", command=submit_form)
submit_button.grid(row=4, column=0, columnspan=2, pady=10)

# Run the application
root.mainloop()
```

Output:



15. The created GUI based application form is connected to a database and use insert query to enter data.

```
import tkinter as tk
from tkinter import messagebox
import sqlite3

# Function to submit form and insert data into the database
def submit_form():
    # Retrieve values from the entry fields
    name = entry_name.get()
    age = entry_age.get()
    gender = gender_var.get()
```

```
    course = course_var.get()

    # Connect to the database
    conn = sqlite3.connect('college.db')
    cursor = conn.cursor()

    # Create students table if it doesn't exist
    cursor.execute('''CREATE TABLE IF NOT EXISTS students
                      (id INTEGER PRIMARY KEY,
                       name TEXT,
                       age INTEGER,
                       gender TEXT,
                       course TEXT)''')

    # Execute the INSERT query
    cursor.execute("INSERT INTO students (name, age, gender, course) VALUES (?, ?, ?, ?)",
(name, age, gender, course))

    # Commit changes and close connection
    conn.commit()
    conn.close()

    # Display submitted information
    messagebox.showinfo("Submission Successful",
                        f"Name: {name}\nAge: {age}\nGender: {gender}\nCourse: {course}")

# Create main window
root = tk.Tk()
root.title("College Admission Form")

# Label and Entry for Name
label_name = tk.Label(root, text="Name:")
label_name.grid(row=0, column=0, padx=10, pady=5, sticky="w")
entry_name = tk.Entry(root)
entry_name.grid(row=0, column=1, padx=10, pady=5)

# Label and Entry for Age
label_age = tk.Label(root, text="Age:")
label_age.grid(row=1, column=0, padx=10, pady=5, sticky="w")
entry_age = tk.Entry(root)
entry_age.grid(row=1, column=1, padx=10, pady=5)

# Radio Buttons for Gender
```

```
label_gender = tk.Label(root, text="Gender:")
label_gender.grid(row=2, column=0, padx=10, pady=5, sticky="w")
gender_var = tk.StringVar()
gender_var.set("Male")
radio_male = tk.Radiobutton(root, text="Male", variable=gender_var, value="Male")
radio_male.grid(row=2, column=1, padx=10, pady=5, sticky="w")
radio_female = tk.Radiobutton(root, text="Female", variable=gender_var, value="Female")
radio_female.grid(row=2, column=2, padx=10, pady=5, sticky="w")

# Dropdown Menu for Course
label_course = tk.Label(root, text="Course:")
label_course.grid(row=3, column=0, padx=10, pady=5, sticky="w")
courses = ["Engineering", "Medical", "Business", "Arts"]
course_var = tk.StringVar()
course_var.set(courses[0])
dropdown_course = tk.OptionMenu(root, course_var, *courses)
dropdown_course.grid(row=3, column=1, padx=10, pady=5)

# Submit Button
submit_button = tk.Button(root, text="Submit", command=submit_form)
submit_button.grid(row=4, column=0, columnspan=2, pady=10)

# Run the application
root.mainloop()
```

Output: