Imported the CSV File

```
import pandas as pan
file=pan.read_csv('/content/Vehicle.csv')
```

The info() and describe() gives the dimension and structure of the data

```
print("First few rows of the CSV file")
print(file.head())
```

```
First few rows
  Car_Name  Year  Selling_Price  Present_Price  Kms_Driven Fuel_Type  \
0     ritz  2014           3.35           5.59       27000    Petrol
1      sx4  2013           4.75           9.54       43000    Diesel
2     ciaz  2017           7.25           9.85        6900    Petrol
3  wagon r  2011           2.85           4.15        5200    Petrol
4    swift  2014           4.60           6.87       42450    Diesel

  Seller_Type Transmission  Owner
0      Dealer       Manual      0
1      Dealer       Manual      0
2      Dealer       Manual      0
3      Dealer       Manual      0
4      Dealer       Manual      0
```

```
print("Summary stats of the CSV data")
print(file.describe())
```

```
Summary stats of the CSV data
              Year  Selling_Price  Present_Price     Kms_Driven       Owner
count   301.000000     301.000000     301.000000     301.000000  301.000000
mean   2013.627907       4.661296       7.628472   36947.205980    0.043189
std       2.891554       5.082812       8.644115   38886.883882    0.247915
min    2003.000000       0.100000       0.320000     500.000000    0.000000
25%    2012.000000       0.900000       1.200000   15000.000000    0.000000
50%    2014.000000       3.600000       6.400000   32000.000000    0.000000
75%    2016.000000       6.000000       9.900000   48767.000000    0.000000
max    2018.000000      35.000000      92.600000  500000.000000    3.000000
```

```
print("Information about the CSV data")
print(file.info())
```

```
Information about the CSV data
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Seller_Type    301 non-null    object
 7   Transmission   301 non-null    object
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
None
```

Q3. Printing the column names from the CSV, first 3 rows and last 6 rows

```
print("Printing the column names")
print(file.columns)
```

```
Printing the column names
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

```
print("Printing the first 3 rows")
print(file.head(3))
```

```
Printing the first 3 rows
  Car_Name  Year  Selling_Price  Present_Price  Kms_Driven Fuel_Type  \
0     ritz  2014           3.35           5.59       27000    Petrol
1      sx4  2013           4.75           9.54       43000    Diesel
2     ciaz  2017           7.25           9.85        6900    Petrol

  Seller_Type Transmission  Owner
0      Dealer       Manual      0
1      Dealer       Manual      0
2      Dealer       Manual      0
```

```
print("Printing the last 6 rows")
print(file.tail(6))
```

```
Printing the last 6 rows
    Car_Name  Year  Selling_Price  Present_Price  Kms_Driven Fuel_Type  \
295     city  2015           8.55          13.09       60076    Diesel
296     city  2016           9.50          11.60       33988    Diesel
297     brio  2015           4.00           5.90       60000    Petrol
298     city  2009           3.35          11.00       87934    Petrol
299     city  2017          11.50          12.50        9000    Diesel
300     brio  2016           5.30           5.90        5464    Petrol

    Seller_Type Transmission  Owner
295      Dealer       Manual      0
296      Dealer       Manual      0
297      Dealer       Manual      0
298      Dealer       Manual      0
299      Dealer       Manual      0
300      Dealer       Manual      0
```

Q4. Show the average Kms_Driven for each type of car (Car_Name) in the dataset

```
print(file.groupby('Car_Name')['Kms_Driven'].mean())
```

```
Printing the average KMs driven by each car
Car_Name
800                  127000.000000
Activa 3g            250250.000000
Activa 4g              1300.000000
Bajaj  ct 100         35000.000000
Bajaj Avenger 150      7000.000000
                           ...
sx4                   50740.000000
verna                 42747.285714
vitara brezza          2071.000000
wagon r               40644.750000
xcent                 27448.333333
Name: Kms_Driven, Length: 98, dtype: float64
```

Q5. What is the average Selling_Price of the cars in each year?

```
print(file.groupby('Year')['Selling_Price'].mean())
```

```
Year
2003    1.300000
2004    1.500000
2005    2.487500
2006    1.437500
2007    0.160000
2008    1.002857
2009    2.816667
2010    5.262667
2011    2.375263
2012    3.841304
2013    3.540909
2014    4.762105
2015    5.927049
2016    5.213200
2017    6.209143
2018    9.250000
Name: Selling_Price, dtype: float64
```

Q6. Show the unique combinations of Car_Name, Fuel_Type, Seller_Type, and Transmission in the Vehicle dataset.

```
print(file[['Car_Name', 'Fuel_Type', 'Seller_Type', 'Transmission']].drop_duplicates())
```

```
    Car_Name Fuel_Type Seller_Type Transmission
0       ritz    Petrol      Dealer       Manual
1        sx4    Diesel      Dealer       Manual
2       ciaz    Petrol      Dealer       Manual
3    wagon r    Petrol      Dealer       Manual
4      swift    Diesel      Dealer       Manual
..       ...       ...         ...          ...
259    amaze    Petrol      Dealer       Manual
263     jazz    Petrol      Dealer       Manual
275     city    Petrol      Dealer    Automatic
285     jazz    Petrol      Dealer    Automatic
287    amaze    Petrol      Dealer    Automatic

[135 rows x 4 columns]
```

Q7. What are the different combinations of Car_Name, Fuel_Type, Seller_Type, and Transmission in the Vehicle dataset, and how many times does it occur? (Display all such in both ascending and descending orders)

```
combinations = file.groupby(['Car_Name', 'Fuel_Type', 'Seller_Type', 'Transmission']).size().reset_index(name='Count')
```

```
print("Combinations in Ascending Order:")
print(combinations.sort_values(by='Count', ascending=True))
```

```
Combinations in Ascending Order:
                     Car_Name Fuel_Type Seller_Type Transmission  Count
0                         800    Petrol  Individual       Manual      1
86                    elantra    Petrol      Dealer    Automatic      1
85                    elantra    Diesel      Dealer       Manual      1
82                      creta    Petrol      Dealer       Manual      1
77                    corolla    Petrol      Dealer    Automatic      1
..                        ...       ...         ...          ...    ...
46  Royal Enfield Classic 350    Petrol  Individual       Manual      7
97                   fortuner    Diesel      Dealer    Automatic      8
68                       brio    Petrol      Dealer       Manual      9
80              corolla altis    Petrol      Dealer       Manual     11
76                       city    Petrol      Dealer       Manual     19

[135 rows x 5 columns]
```

```
print("Combinations in Descending Order:")
print(combinations.sort_values(by='Count', ascending=False))
```

```
Combinations in Descending Order:
                     Car_Name Fuel_Type Seller_Type Transmission  Count
76                       city    Petrol      Dealer       Manual     19
80              corolla altis    Petrol      Dealer       Manual     11
68                       brio    Petrol      Dealer       Manual      9
97                   fortuner    Diesel      Dealer    Automatic      8
130                     verna    Petrol      Dealer       Manual      7
..                        ...       ...         ...          ...    ...
45   Royal Enfield Bullet 350    Petrol  Individual       Manual      1
44        Mahindra Mojo XT300    Petrol  Individual       Manual      1
43                  KTM RC390    Petrol  Individual       Manual      1
41              KTM 390 Duke     Petrol  Individual       Manual      1
67                       brio    Petrol      Dealer    Automatic      1

[135 rows x 5 columns]
```

Q8. Find if there are any missing values in the Vehicle dataset

```
print("Missing Values in the Vehicle dataset:")
print(file.isnull().sum())
```

```
Missing Values in the Vehicle dataset:
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

Q9. Find which columns contain missing values in the vehicles dataset. What are the total missing values for each column?

```
missingValues = file.isnull().sum()
print("Columns with Missing Values:")
print(missingValues[missingValues > 0])
```

```
Columns with Missing Values:
Series([], dtype: int64)
```

Q10. Replace the missing values in the dataset with the most repeated value of that field. Check if the missing values were replaced successfully

```
fileFilled = file.fillna(file.mode().iloc[0])
print("Columns with Missing Values After Replacement:")
print(fileFilled.isnull().sum())
```

```
Columns with Missing Values After Replacement:
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

Q11. Find if the dataset has duplicate rows. Remove them, if exist

```
print("Number of Duplicate Rows : ", file.duplicated().sum())
print("Dataset shape after removing duplicates:", file.drop_duplicates().shape)
```

```
Number of Duplicate Rows :  2
Dataset shape after removing duplicates: (299, 9)
```

Q12. Replace the values of the following attributes: a Fuel_Type: “Petrol”: 0, “Diesel”: 1, “CNG”: 2 b Seller_Type: “Dealer”: 0, “Individual”: 1 c Transmission: “Manual”: 0, “Automatic”: 1 Show the conversion output of the specific attribute

```
file['Fuel_Type'].replace({"Petrol": 0, "Diesel": 1, "CNG": 2}, inplace=True)
file['Seller_Type'].replace({"Dealer": 0, "Individual": 1}, inplace=True)
file['Transmission'].replace({"Manual": 0, "Automatic": 1}, inplace=True)
```

```
print("Conversion for Fuel_Type:")
print(file['Fuel_Type'].value_counts())

print("\nConversion for Seller_Type:")
print(file['Seller_Type'].value_counts())

print("\nConversion for Transmission:")
print(file['Transmission'].value_counts())
```

```
Conversion for Fuel_Type:
0    237
1     58
2      6
Name: Fuel_Type, dtype: int64

Conversion for Seller_Type:
0    195
1    106
Name: Seller_Type, dtype: int64

Conversion for Transmission:
0    261
1     40
Name: Transmission, dtype: int64
```

Q13. Add a new field called ‘Age’, and input the values by using the field Year. Show the output

```
file['Age'] = 2023 - file['Year']

print(file[['Car_Name', 'Year', 'Age']].head())
```

```
   Car_Name  Year  Age
0     ritz  2014    9
1      sx4  2013   10
2     ciaz  2017    6
3  wagon r  2011   12
4    swift  2014    9
```

Q14. Create a new dataset by selecting only the columns “Car_name”, “Selling_Price”, “Present_Price”, and “Kms_Drive”. Show the output of the new dataset

```
columns = file[['Car_Name', 'Selling_Price', 'Present_Price', 'Kms_Driven']]
print(columns.head())
```

```
   Car_Name  Selling_Price  Present_Price  Kms_Driven
0     ritz           3.35           5.59       27000
1      sx4           4.75           9.54       43000
2     ciaz           7.25           9.85        6900
3  wagon r           2.85           4.15        5200
4    swift           4.60           6.87       42450
```

Q15. Shuffle the rows of the Vehicle dataset randomly and show the output

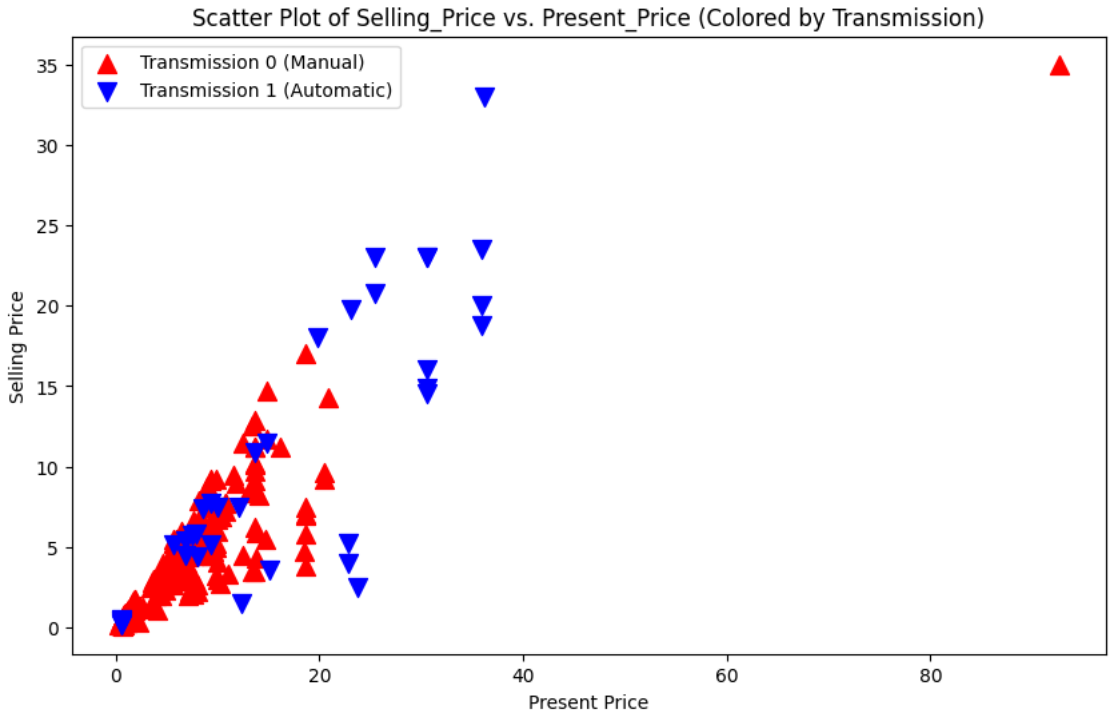
```
print(file.sample(frac=1.0, random_state=42).head())
```

```
                Car_Name  Year  Selling_Price  Present_Price  Kms_Driven  \
177    Honda Activa 125  2016           0.35           0.57       24000
289                city  2016          10.11          13.60       10980
228               verna  2012           4.95           9.40       60000
198  Bajaj Discover 125  2011           0.15           0.57       35000
60        corolla altis  2013           6.95          18.61       40001

     Fuel_Type  Seller_Type  Transmission  Owner  Age
177          0            1             1      0    7
289          0            0             0      0    7
228          1            0             0      0   11
198          0            1             0      1   12
60           0            0             0      0   10
```

Q16. Import the Vehicle dataset. Create a scatter plot of the Selling_Price Vs Present_Price. Colour code the points based on the Transmission (5 marks). a. Add labels, title and colour to the plot. The colour should be red for Transmission type ‘0’ and blue for ‘1’. b. Add open triangles to the plot. c. What do you understand from the output

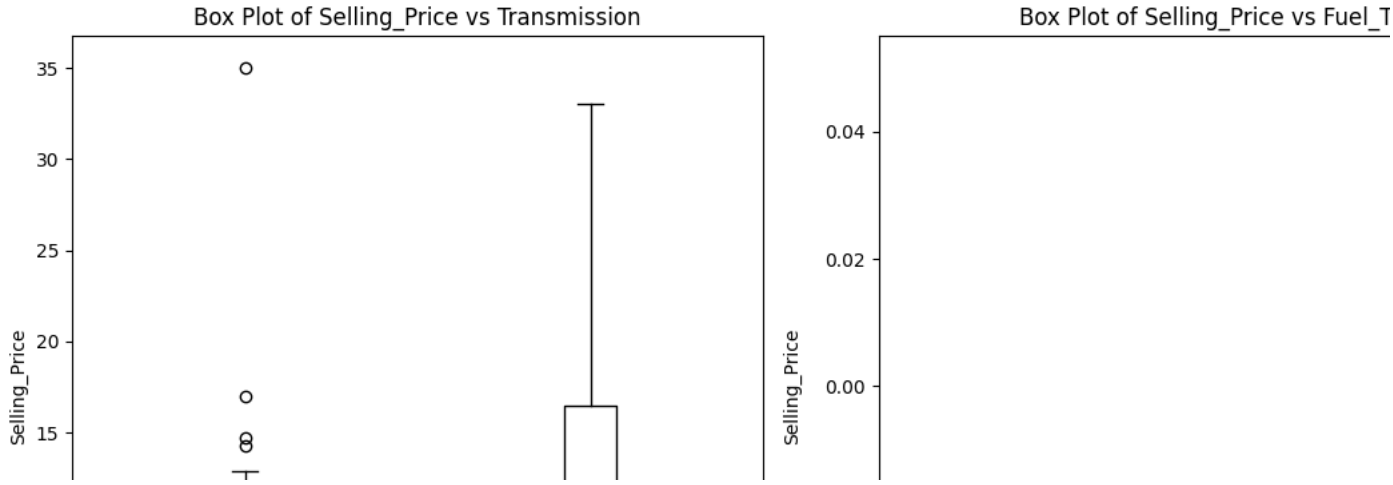
```
import matplotlib.pyplot as plt



plt.figure(figsize=(10, 6))


plt.scatter(
    file[file['Transmission'] == 0]['Present_Price'],
    file[file['Transmission'] == 0]['Selling_Price'],
    c='red',
    label='Transmission 0 (Manual)',
    marker='^',
    s=100,
)


plt.scatter(
    file[file['Transmission'] == 1]['Present_Price'],
    file[file['Transmission'] == 1]['Selling_Price'],
    c='blue',
    label='Transmission 1 (Automatic)',
    marker='v',
    s=100,
)
```

```
plt.xlabel('Present Price')
plt.ylabel('Selling Price')
plt.title('Scatter Plot of Selling_Price vs. Present_Price (Colored by Transmission)')
plt.legend()


plt.show()
```



Q17. Create a box plot of the Selling_Price Vs Transmission and Fuel_Type

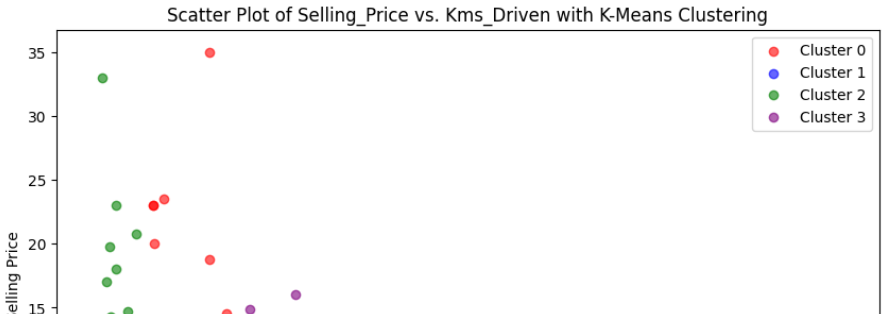
```
plt.figure(figsize=(12, 6))


plt.subplot(1, 2, 1)
plt.boxplot(
    [file[file['Transmission'] == 0]['Selling_Price'], file[file['Transmission'] == 1]['Selling_Price']],
    labels=['Transmission 0 (Manual)', 'Transmission 1 (Automatic)'],
)
plt.title('Box Plot of Selling_Price vs Transmission')
plt.ylabel('Selling_Price')

plt.subplot(1, 2, 2)
plt.boxplot(
    [file[file['Fuel_Type'] == 'Petrol']['Selling_Price'],
     file[file['Fuel_Type'] == 'Diesel']['Selling_Price'],
     file[file['Fuel_Type'] == 'CNG']['Selling_Price']],
    labels=['Petrol', 'Diesel', 'CNG'],
)
plt.title('Box Plot of Selling_Price vs Fuel_Type')
plt.ylabel('Selling_Price')


plt.tight_layout()


plt.show()
```



Q18. Create a scatter plot of the Selling_Price Vs Kms_Driven, and use k-means clustering to cluster the points into 4 clusters. Colour-code based on the cluster they belong to

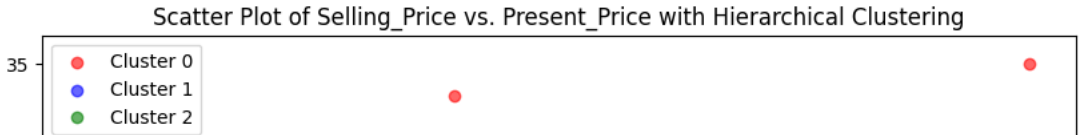
```
from sklearn.cluster import KMeans

X = file[['Selling_Price', 'Kms_Driven']]


kmeans = KMeans(n_clusters=4, random_state=0)

file['Cluster'] = kmeans.fit_predict(X)

plt.figure(figsize=(10, 6))


colors = ['red', 'blue', 'green', 'purple']


for i in range(4):
    cluster_points = file[file['Cluster'] == i]
    plt.scatter(
        cluster_points['Kms_Driven'],
        cluster_points['Selling_Price'],
        label=f'Cluster {i}',
        c=colors[i],
        alpha=0.6,
    )


plt.xlabel('Kms Driven')
plt.ylabel('Selling Price')
plt.title('Scatter Plot of Selling_Price vs. Kms_Driven with K-Means Clustering')
plt.legend()


plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWa
  warnings.warn(
```



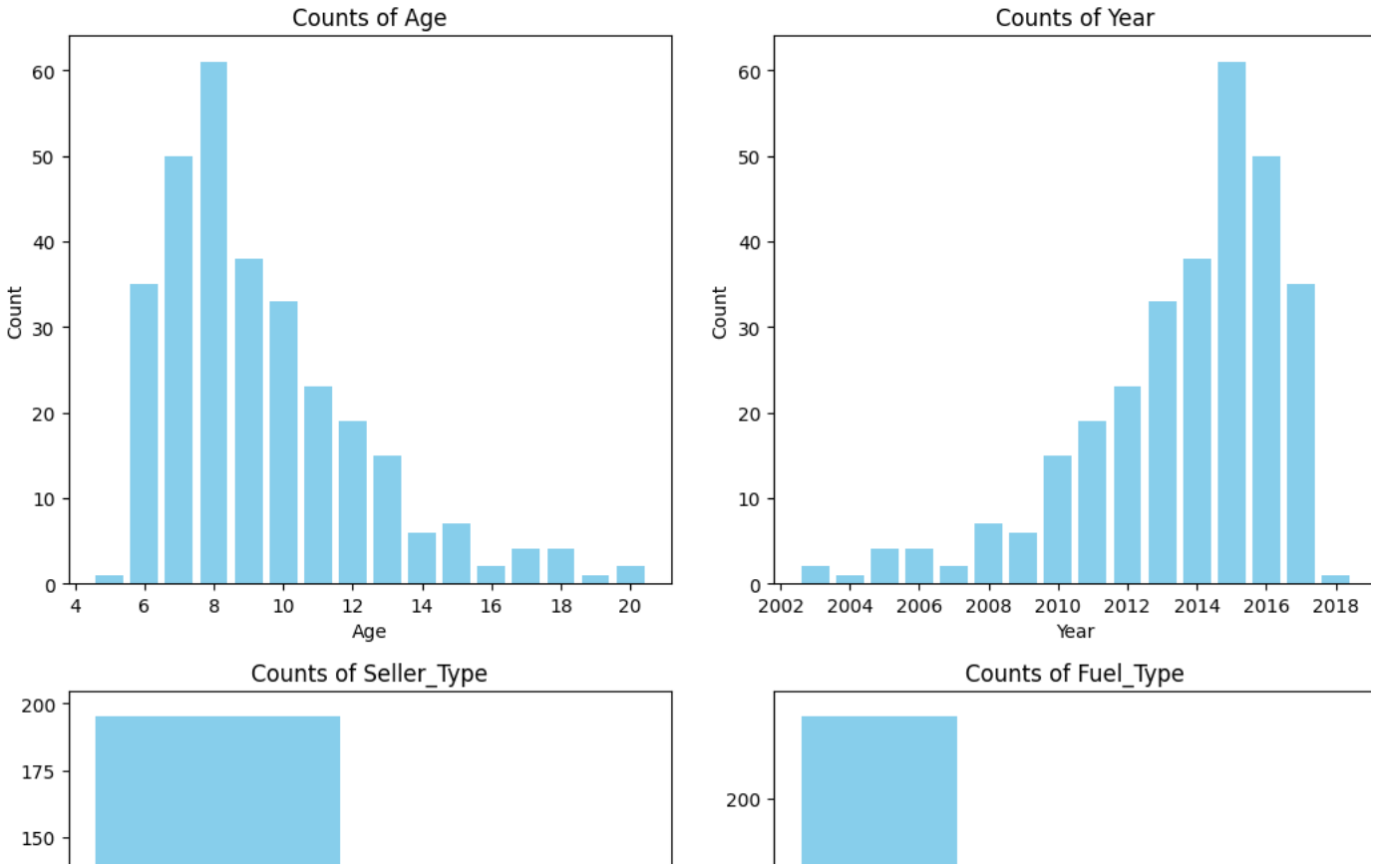
Q19. Create a scatter plot of the Selling_Price Vs Present_Price, and use hierarchical clustering to cluster the points into 3 clusters? Colour-code the points based on the cluster they belong to

```
from sklearn.cluster import AgglomerativeClustering


X = file[['Selling_Price', 'Present_Price']]

hierarchical = AgglomerativeClustering(n_clusters=3)


file['Cluster'] = hierarchical.fit_predict(X)

plt.figure(figsize=(10, 6))


colors = ['red', 'blue', 'green']


for i in range(3):
    cluster_points = file[file['Cluster'] == i]
    plt.scatter(
        cluster_points['Present_Price'],
        cluster_points['Selling_Price'],
        label=f'Cluster {i}',
        c=colors[i],
        alpha=0.6,
    )


plt.xlabel('Present Price')
plt.ylabel('Selling Price')
plt.title('Scatter Plot of Selling_Price vs. Present_Price with Hierarchical Clustering')
plt.legend()

plt.show()
```



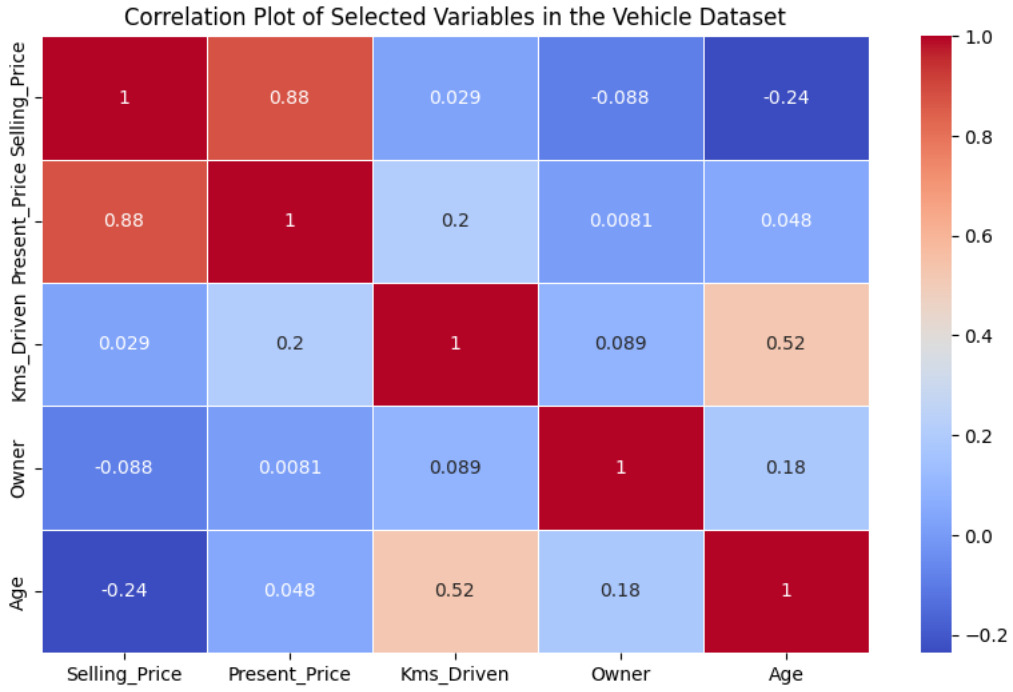
Q20. Add a new field called 'Age', and calculate it using the field 'Year'. Create a barplot for the following fields of the dataset: (10 marks) a. 'Age', 'Year', 'Transmission', 'Seller_Type', 'Fuel_Type' and 'Owner' b. Add labels, titles, and colours to the plot.

```
file['Age'] = 2023 - file['Year']


fields_to_plot = ['Age', 'Year', 'Transmission', 'Seller_Type', 'Fuel_Type', 'Owner']


fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(16, 10))

for i, field in enumerate(fields_to_plot):
    row = i // 3
    col = i % 3
    ax = axes[row, col]


    value_counts = file[field].value_counts()
    ax.bar(value_counts.index, value_counts.values, color='skyblue')
    ax.set_title(f'Counts of {field}')
    ax.set_xlabel(field)
    ax.set_ylabel('Count')


plt.tight_layout()


plt.show()
```



Q21. Create a correlation plot of the whole dataset variables and explain the output. Do not forget to convert some of the variable's datatype if required and possible

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


file = pd.read_csv('/content/Vehicle.csv')

file = pd.get_dummies(file, columns=['Fuel_Type', 'Seller_Type', 'Transmission'], drop_first=True)

file['Age'] = 2023 - file['Year']

columns_to_correlate = ['Selling_Price', 'Present_Price', 'Kms_Driven', 'Owner', 'Age']


correlation_matrix = file[columns_to_correlate].corr()


plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Plot of Selected Variables in the Vehicle Dataset')
plt.show()
```



Q22. Create a scatter plot of the Selling_Price Vs Kms_Driven, and use DBSCAN clustering to cluster the points into 3 clusters. Color-code based on the cluster they belong to. Add a legend to the plot

```
from sklearn.cluster import DBSCAN
file = pd.read_csv('/content/Vehicle.csv')


X = file[['Selling_Price', 'Kms_Driven']]
dbscan = DBSCAN(eps=0.5, min_samples=10)
cluster_labels = dbscan.fit_predict(X)


file['Cluster'] = cluster_labels


plt.figure(figsize=(10, 6))

colors = ['red', 'green', 'blue']

for i in range(-1, max(cluster_labels) + 1):
    if i == -1:
        cluster_points = file[cluster_labels == i]
        plt.scatter(
            cluster_points['Kms_Driven'],
            cluster_points['Selling_Price'],
```

```
                c='gray',
                label=f'Noise Points (Cluster {i})'
            )
        else:
            cluster_points = file[cluster_labels == i]
            plt.scatter(
                cluster_points['Kms_Driven'],
                cluster_points['Selling_Price'],
                c=colors[i],
                label=f'Cluster {i}'
            )

    plt.xlabel('Kms_Driven')
    plt.ylabel('Selling_Price')
    plt.title('Scatter Plot of Selling_Price vs. Kms_Driven (DBSCAN Clustering)')
    plt.legend()

    plt.show()
```