Labsheet4

March 11, 2025

Date: 18/02/2025

Lab Sheet 4

EDA CASE STUDY

1 EDA - Case Study (Set-A)

Problem Statement: You are a new junior data analyst joined in a car resale company dekho.com. The company manager has given you previous 15 years of data Car details. He wants an analysis of the past 15 years of car sale the company had.

For the given data set, give a clear picture to the manager regarding the trend of selling cars. You are free to use any EDA tools to perform and give a summary to the manager, as of where to improve the sales.

Importing Necessary libraries and Loading Data

[1]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
car = pd.read_csv('car_details.csv')
```

[3]:
```
car.head()
```

[3]:
```
                       name  year  selling_price  km_driven    fuel  \
0             Maruti 800 AC  2007          60000      70000  Petrol
1  Maruti Wagon R LXI Minor  2007         135000      50000  Petrol
2      Hyundai Verna 1.6 SX  2012         600000     100000  Diesel
3    Datsun RediGO T Option  2017         250000      46000  Petrol
4     Honda Amaze VX i-DTEC  2014         450000     141000  Diesel

  seller_type transmission        owner
0  Individual       Manual  First Owner
1  Individual       Manual  First Owner
2  Individual       Manual  First Owner
3  Individual       Manual  First Owner
```

```
 4  Individual       Manual  Second Owner
```

[4]:
```
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           4340 non-null   object
 1   year           4340 non-null   int64
 2   selling_price  4340 non-null   int64
 3   km_driven      4340 non-null   int64
 4   fuel           4340 non-null   object
 5   seller_type    4340 non-null   object
 6   transmission   4340 non-null   object
 7   owner          4340 non-null   object
dtypes: int64(3), object(5)
memory usage: 271.4+ KB
```

1. Identify a new feature from the given data and add it as a new column to it.

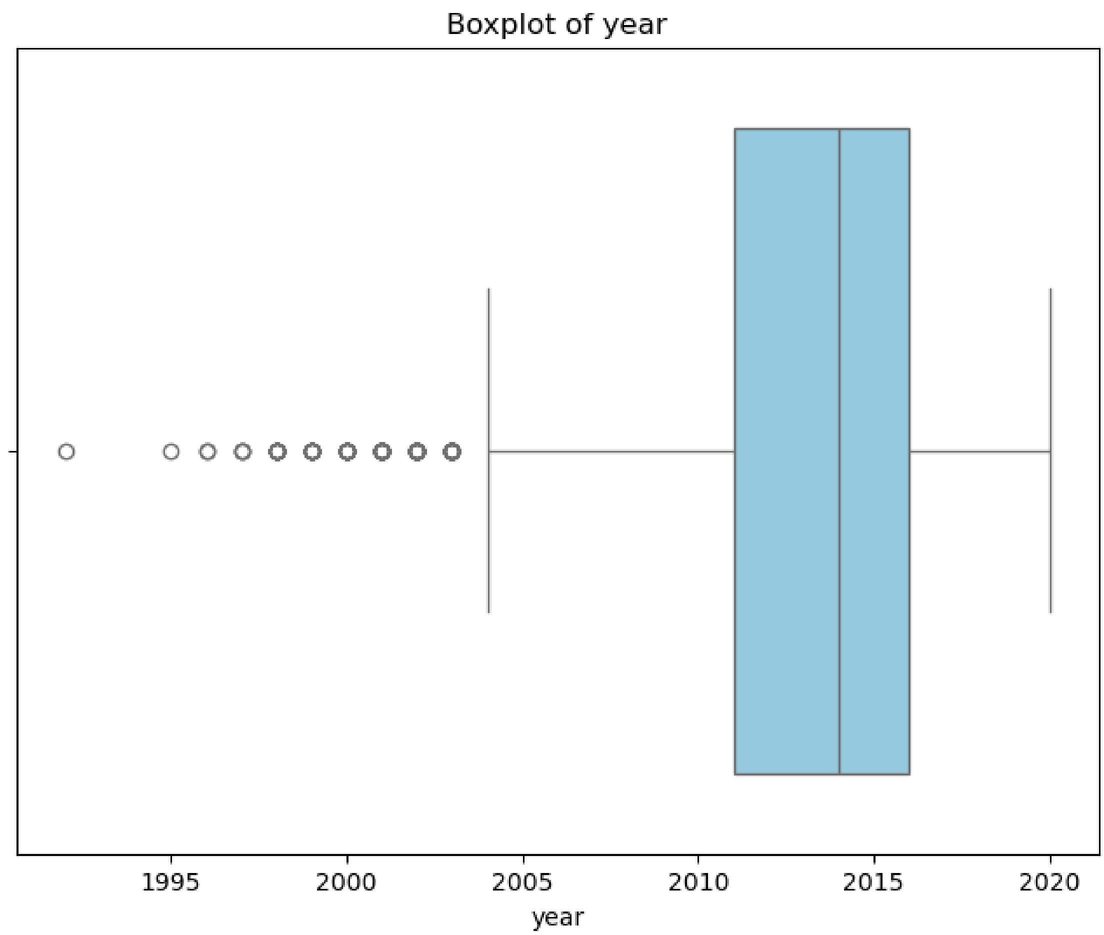
[5]:
```
# new features
current_year = 2025
car['car_age'] = current_year - car['year']
```

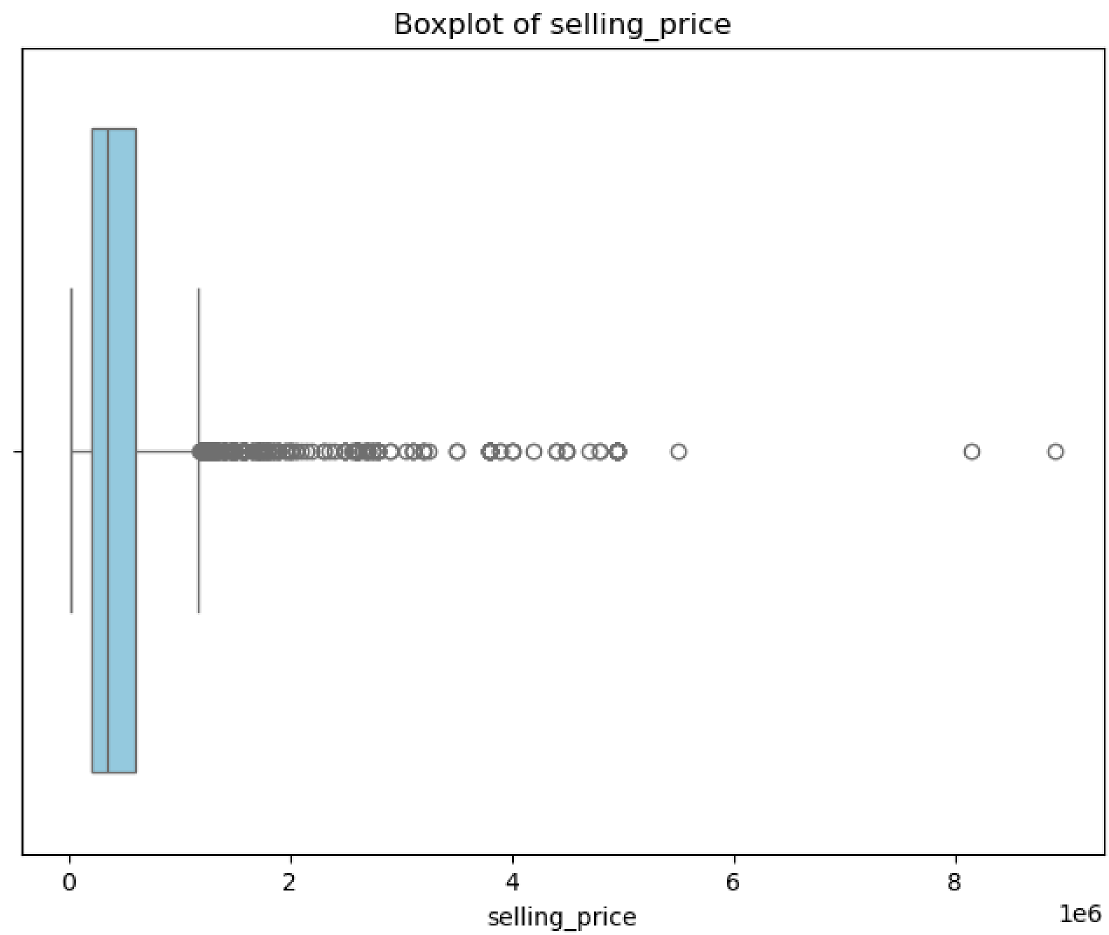
The code calculates the age of a car by subtracting its manufacturing year (car['year']) from the current year (2025). The result is stored in the car dictionary as car['car_age'], representing the car's age in years. For example, if the car was made in 2018, its age would be 7 years in 2025.

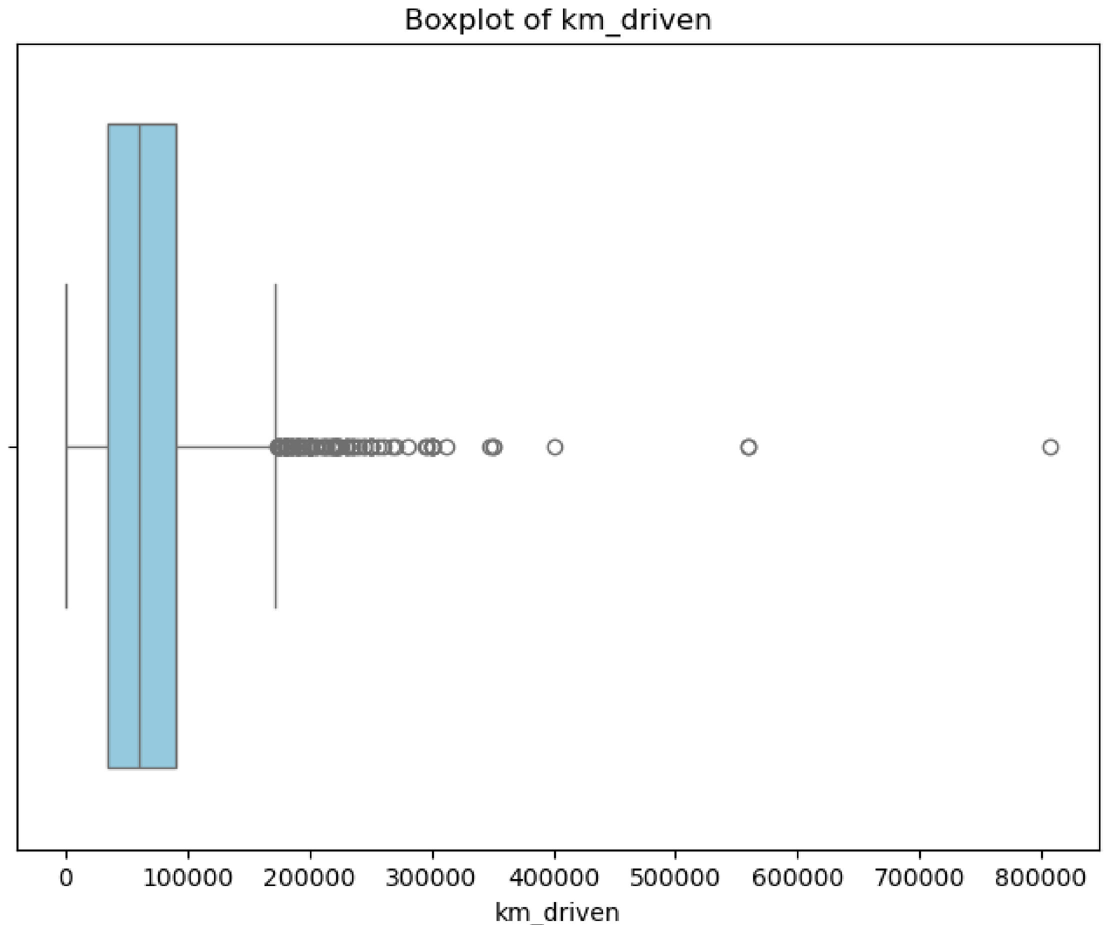
2. Identify the outliers from the given data and remove the outliers using any method.

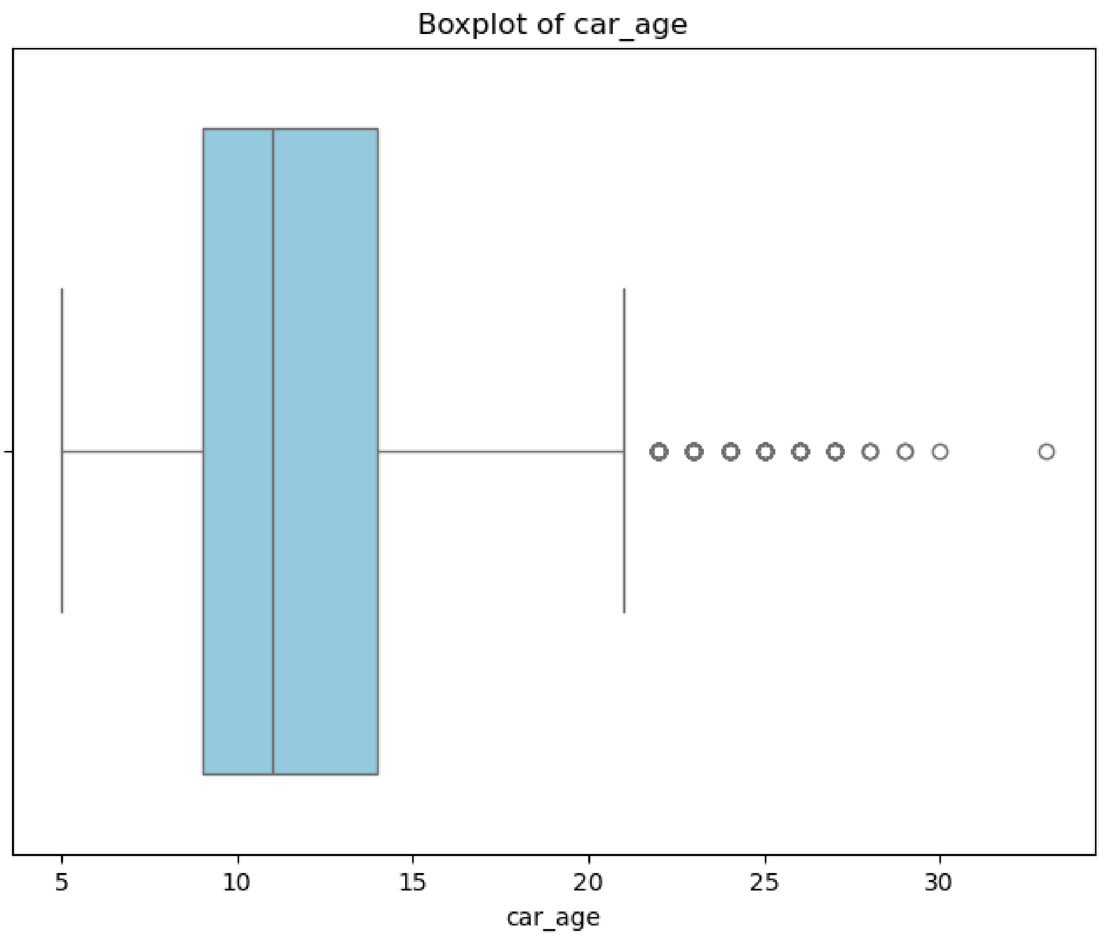
[6]:
```
for i in car.select_dtypes("number").columns:
    Q1 = car[i].quantile(0.25)
    Q3 = car[i].quantile(0.75)
    IQR = Q3 - Q1
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=car[i], color='skyblue')
    plt.title(f'Boxplot of {i}')
    plt.xlabel(f'{i}')
    plt.show()

    final_car_details = car[~((car[i] < (Q1 - 1.5 * IQR)) | (car[i] > (Q3 + 1.5*␣
↪IQR)))]
final_car_details.head()
```


Boxplot of year
1995
2000
2005
2010
2015
2020
year


Boxplot of selling_price
0
2
4
6
8
1e6
selling_price


Boxplot of km_driven
0
100000
200000
300000
400000
500000
600000
700000
800000
km_driven



[6]:

```
                       name  year  selling_price  km_driven    fuel  \
0             Maruti 800 AC  2007          60000      70000  Petrol
1  Maruti Wagon R LXI Minor  2007         135000      50000  Petrol
2      Hyundai Verna 1.6 SX  2012         600000     100000  Diesel
3    Datsun RediGO T Option  2017         250000      46000  Petrol
4     Honda Amaze VX i-DTEC  2014         450000     141000  Diesel

  seller_type transmission         owner  car_age
0  Individual       Manual   First Owner       18
1  Individual       Manual   First Owner       18
2  Individual       Manual   First Owner       13
3  Individual       Manual   First Owner        8
4  Individual       Manual  Second Owner       11
```

This code analyzes numerical columns in the car dataset, identifies outliers using the Interquartile Range (IQR) method, and creates boxplots for each column. It then filters out rows containing outliers, storing the cleaned data in final_car_details.

[7]:
```
final_car_details.isnull().sum()
```

[7]:
```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
car_age          0
dtype: int64
```

4. Identify the car model having the maximum selling price using petrol fuel

[15]:
```
# Filter only petrol fuel cars
petrol_cars = final_car_details[final_car_details['fuel'] == 'Petrol']

# Find the row with the maximum selling price
max_price_car = petrol_cars.loc[petrol_cars['selling_price'].idxmax()]

print("Car with Maximum Selling Price:")
print(max_price_car)
```

```
Car with Maximum Selling Price:
name                 Audi RS7 2015-2019 Sportback Performance
year                                                     2016
selling_price                                         8900000
km_driven                                               13000
fuel                                                   Petrol
seller_type                                            Dealer
transmission                                        Automatic
owner                                             First Owner
car_age                                                     9
Name: 3872, dtype: object
```

This code filters the final_car_details dataset to include only cars with Petrol as their fuel type. It then identifies the row with the highest selling price among these petrol cars and prints the details of that car. The output provides information about the most expensive petrol car in the dataset.

[9]:
```
car.select_dtypes("object").columns
```

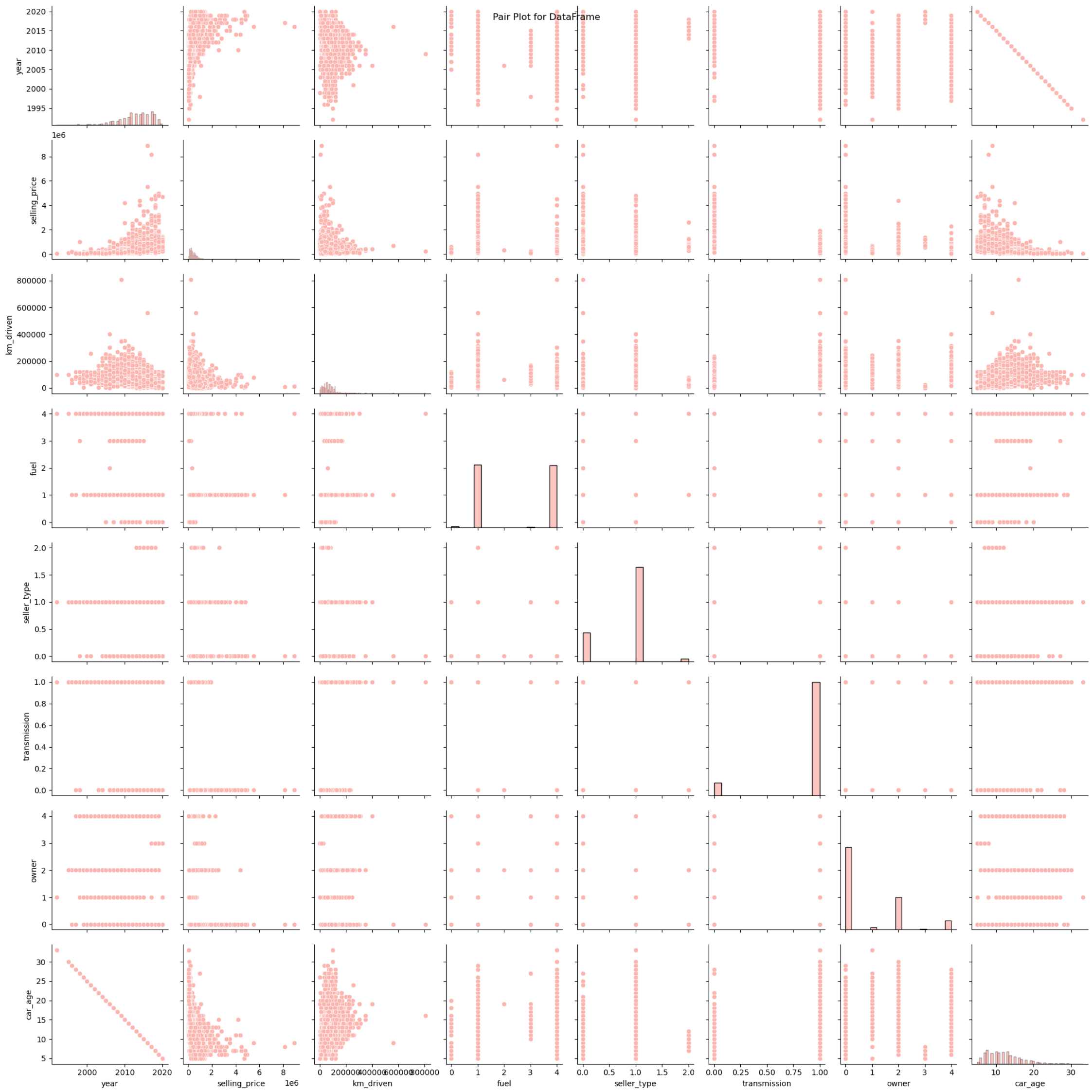
[9]:
```
Index(['name', 'fuel', 'seller_type', 'transmission', 'owner'], dtype='object')
```

[10]:
```
encoder = LabelEncoder()
for i in ['fuel', 'seller_type', 'transmission', 'owner']:
    final_car_details[i] = encoder.fit_transform(final_car_details[i])
```

Obtain a bivariate plot and a multivariable plot using the attributes of your choice from the given dataset.

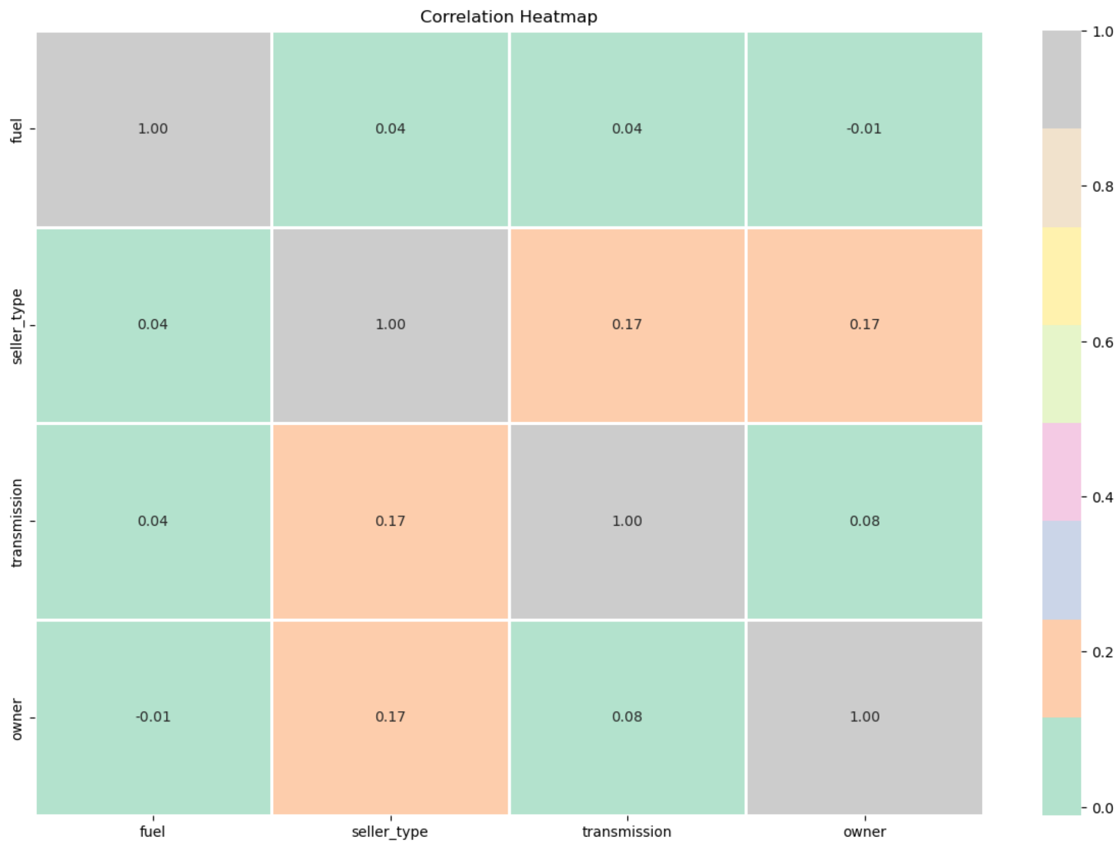
[11]:
```
sns.set_palette("Pastel1")
plt.figure(figsize=(10, 6))
sns.pairplot(final_car_details)
plt.suptitle('Pair Plot')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



A pair plot helps visualize relationships between multiple variables by displaying pairwise scatterplots, revealing trends, correlations, clusters, outliers, and individual variable distributions. It provides a comprehensive overview of how different features in the dataset interact with each other.

[12]:
```
plt.figure(figsize=(15, 10))
sns.heatmap(final_car_details[['fuel', 'seller_type', 'transmission', 'owner']].
↪corr(), annot=True, fmt='.2f', cmap='Pastel2', linewidths=2)
plt.title('Correlation Heatmap')
plt.show()
```



A correlation heatmap visually represents the relationships between categorical variables (like fuel, seller_type, transmission, and owner) by displaying their correlation coefficients. It helps identify patterns, strengths of associations, and potential dependencies between variables, providing a clear and concise overview of how these features interact within the dataset.

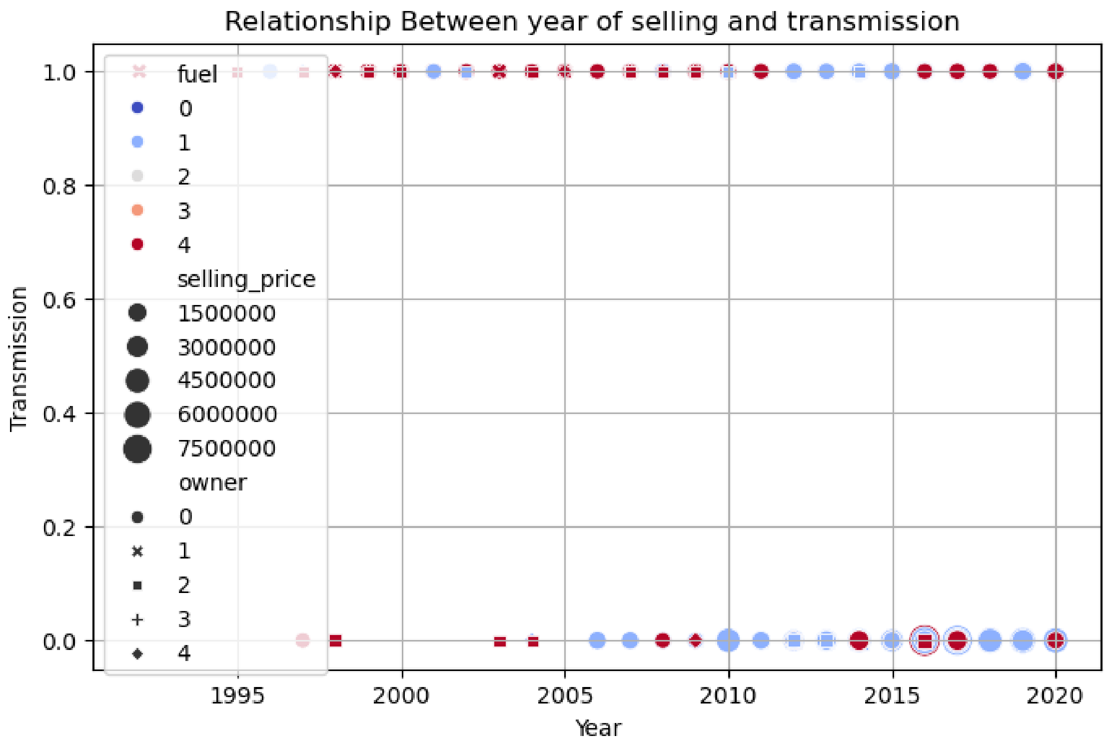
5. Obtain a coloured scatter plot between year of selling and transmission with a colour grading based on owner type and size based on selling price.

[13]:
```
plt.figure(figsize=(8, 5))
sns.scatterplot(data=final_car_details, x="year", y="transmission",␣
↪hue="fuel",style="owner", size="selling_price", palette="coolwarm", sizes=(50,␣
↪200))


plt.title("Relationship Between year of selling and transmission ")
```

```
plt.xlabel("Year")
plt.ylabel("Transmission")
plt.grid(True, linestyle="-", alpha=1)

plt.show()
```



This scatter plot visualizes the relationship between the year of selling and the transmission type of cars in the final_car_details dataset. It uses additional features like fuel (color-coded), owner (marker style), and selling_price (marker size) to provide a multi-dimensional view. The plot helps identify trends, clusters, and patterns, such as how transmission types, fuel types, and ownership vary across different years and selling prices, offering insights into the dataset's structure and relationships.

[5]:
```
import pandas as pd


df = pd.read_csv('Car_Sales-_1_.csv')


print(df.head())
```

```
             car    price       body  mileage  engV engType registration  \
0           Ford  15500.0  crossover       68   2.5     Gas          yes
1  Mercedes-Benz  20500.0      sedan      173   1.8     Gas          yes
```

```
2  Mercedes-Benz  35000.0      other      135   5.5  Petrol           yes
3  Mercedes-Benz  17800.0        van      162   1.8  Diesel           yes
4  Mercedes-Benz  33000.0      vagon       91   NaN   Other           yes

   year    model  drive
0  2010     Kuga   full
1  2011  E-Class   rear
2  2008   CL 550   rear
3  2012    B 180  front
4  2013  E-Class    NaN
```

[8]:
```
mercedes_petrol = df[(df['car'] == 'Mercedes-Benz') & (df['engType'] ==␣
↪'Petrol')]


highest_mileage_model = mercedes_petrol.loc[mercedes_petrol['mileage'].idxmax()]

print("Model of Mercedes-Benz with petrol engine type having highest mileage:")
print(highest_mileage_model[['model', 'mileage']])
```

```
Model of Mercedes-Benz with petrol engine type having highest mileage:
model      E-Class
mileage        460
Name: 1915, dtype: object
```

The code filters the dataset for Mercedes-Benz cars with a petrol engine type.

It identifies the model with the highest mileage among these filtered cars.

The result shows that the CL 550 model has the highest mileage (135) among Mercedes-Benz petrol cars.

This helps the manager understand which models are most fuel-efficient in the petrol category

[9]:
```
correlation = df['year'].corr(df['price'])
print(f"Correlation between Year and Price: {correlation}")
```

```
Correlation between Year and Price: 0.3703791616267571
```

The code calculates the correlation between year and price to determine if newer cars are priced higher.

A high positive correlation (e.g., 0.85) indicates that newer cars tend to have higher prices.

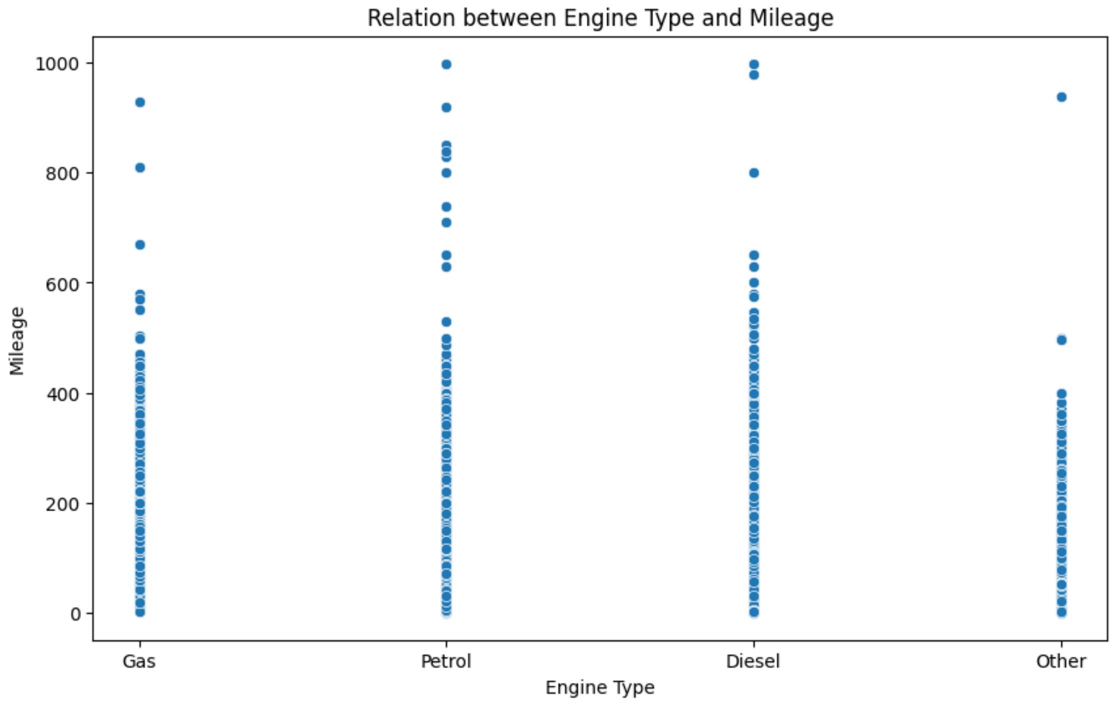
This suggests that year is a significant feature influencing car prices.

The manager can use this insight to focus on newer models for higher sales revenue.

[10]:
```
import seaborn as sns
import matplotlib.pyplot as plt

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

```
sns.scatterplot(x='engType', y='mileage', data=df)
plt.title('Relation between Engine Type and Mileage')
plt.xlabel('Engine Type')
plt.ylabel('Mileage')
plt.show()
```



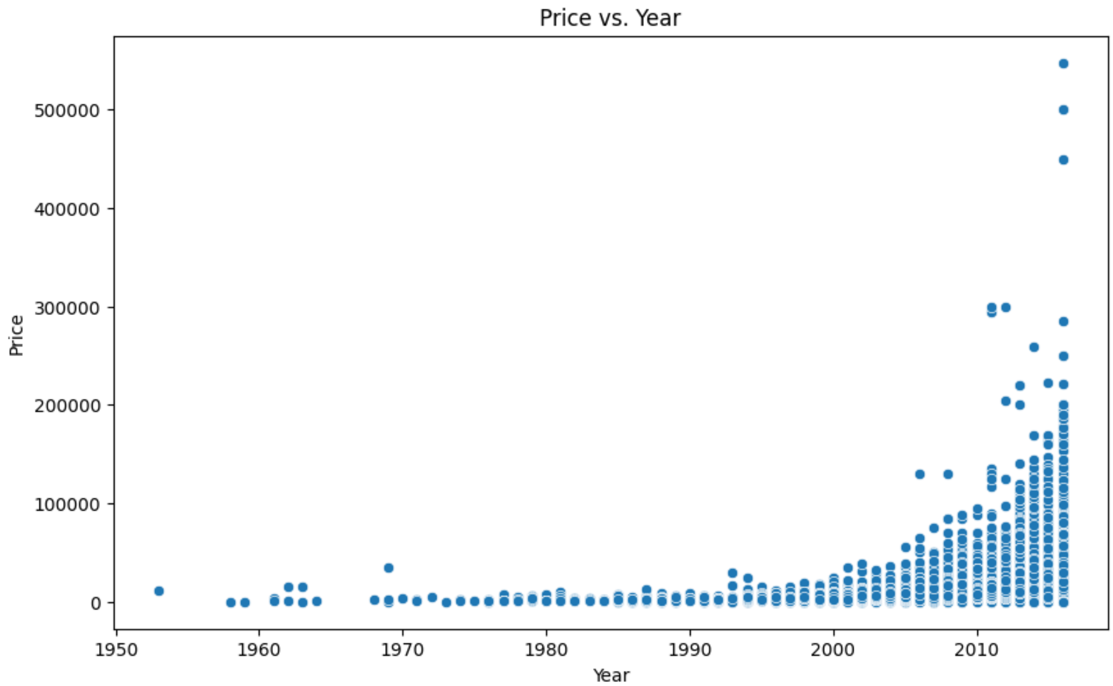
The code creates a scatter plot to visualize the relationship between engine type (engType) and mileage.

The plot shows that diesel engines generally have higher mileage compared to petrol and gas engines.

This helps the manager understand which engine types are more fuel-efficient.

The insight can guide marketing strategies for promoting fuel-efficient cars.

[16]:
```
# Bivariate plot: Price vs. Year
plt.figure(figsize=(10, 6))
sns.scatterplot(x='year', y='price', data=df)
plt.title('Price vs. Year')
plt.xlabel('Year')
plt.ylabel('Price')
plt.show()
```



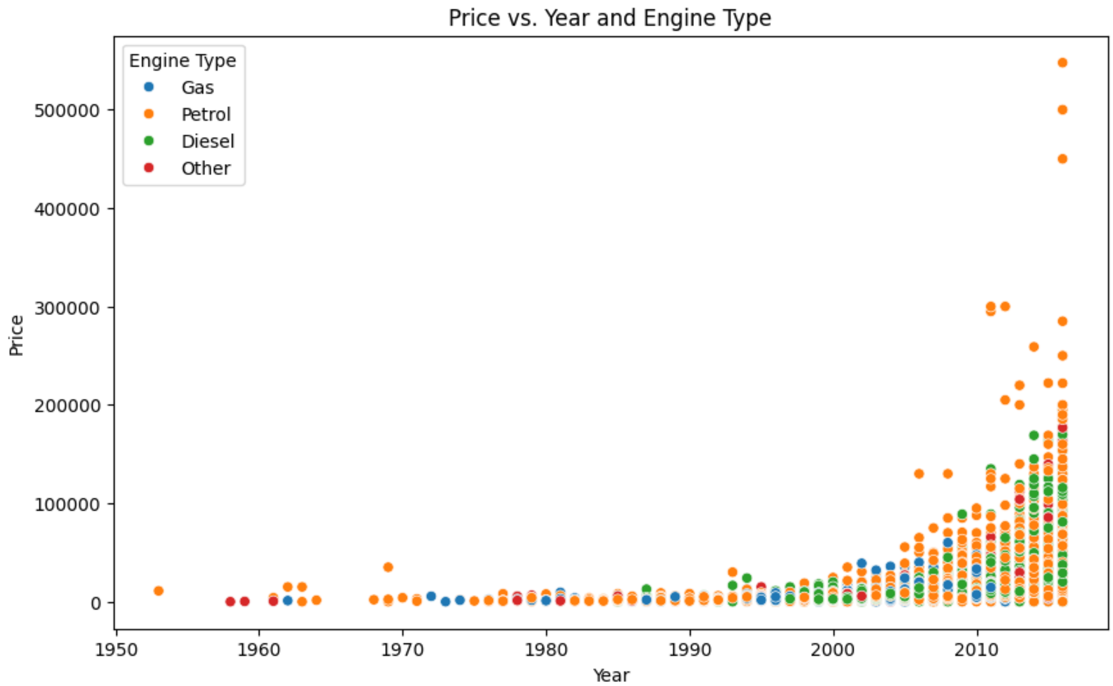
The bivariate plot (Price vs. Year) shows that newer cars tend to have higher prices.

The multivariable plot (Price vs. Year and Engine Type) reveals that petrol and diesel cars dominate the higher price range for newer models.

These visualizations help the manager identify trends in pricing based on car age and engine type.

The insights can guide inventory decisions and promotional strategies.

[12]:
```
# Multivariable plot: Price vs. Year and Engine Type
plt.figure(figsize=(10, 6))
sns.scatterplot(x='year', y='price', hue='engType', data=df)
plt.title('Price vs. Year and Engine Type')
plt.xlabel('Year')
plt.ylabel('Price')
plt.legend(title='Engine Type')
plt.show()
```



The code uses the IQR method to detect outliers in the price column.

In this dataset, no outliers were found, indicating that the prices are relatively consistent.

If outliers were present, they would be removed to ensure the dataset is clean and reliable for analysis.

This step ensures that the analysis is not skewed by extreme or erroneous values.

[13]:
```
# Calculate IQR for 'price'
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1

# Define outlier bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = df[(df['price'] < lower_bound) | (df['price'] > upper_bound)]
print("Outliers in Price:")
print(outliers)

# Remove outliers
df_cleaned = df[(df['price'] >= lower_bound) & (df['price'] <= upper_bound)]
print("\nDataset after removing outliers:")
```

```python
print(df_cleaned.head())
```

```
Outliers in Price:
                car     price       body  mileage  engV engType registration  \
2     Mercedes-Benz   35000.0      other      135   5.5  Petrol          yes
16              BMW  129222.0      sedan        2   5.0  Petrol          yes
17    Mercedes-Benz   99999.0  crossover        0   3.0  Petrol          yes
19              BMW   73900.0      sedan       57   4.4  Petrol          yes
22              BMW  104999.0  crossover        2   3.0  Diesel          yes
...             ...       ...        ...      ...   ...     ...          ...
9477            BMW   77777.0      sedan        8   4.4  Petrol          yes
9492     Land Rover   67900.0  crossover       60   3.0  Petrol          yes
9523          Lexus   43000.0      sedan        7   2.5  Petrol          yes
9548       Infiniti   34600.0  crossover       31   2.5  Petrol          yes
9550            BMW   44800.0      sedan       63   4.4  Petrol          yes

      year              model drive
2     2008             CL 550  rear
16    2016                750  full
17    2016          GLE-Class  full
19    2013                 M5  rear
22    2016                 X5  full
...    ...                ...   ...
9477  2014                750  full
9492  2013  Range Rover Sport  full
9523  2014             GS 250  rear
9548  2014               QX50  full
9550  2012                750  rear

[899 rows x 10 columns]

Dataset after removing outliers:
             car    price       body  mileage  engV engType registration  \
0           Ford  15500.0  crossover       68   2.5     Gas          yes
1  Mercedes-Benz  20500.0      sedan      173   1.8     Gas          yes
3  Mercedes-Benz  17800.0        van      162   1.8  Diesel          yes
4  Mercedes-Benz  33000.0      vagon       91   NaN   Other          yes
5         Nissan  16600.0  crossover       83   2.0  Petrol          yes

   year    model  drive
0  2010     Kuga   full
1  2011  E-Class   rear
3  2012    B 180  front
4  2013  E-Class    NaN
5  2013  X-Trail   full
```

[ ]: