In [12]:
```
pip install numpy pandas matplotlib seaborn plotly
```

```
Requirement already satisfied: numpy in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (1.26.4)
Requirement already satisfied: pandas in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (2.2.2)
Requirement already satisfied: matplotlib in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (3.9.0)
Requirement already satisfied: seaborn in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (0.13.2)
Requirement already satisfied: plotly in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (5.22.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (4.53.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from plotly) (8.3.0)
Requirement already satisfied: six>=1.5 in c:\users\laasya popuri\appdata\local\programs\python\python311\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import warnings
warnings.filterwarnings("ignore")
```

In [15]:
```
df = pd.read_csv('car data.csv')
```

In [16]:
```
print(" First few rows of the dataset:")
df.head()
```

First few rows of the dataset:

Out[16]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

In [17]:
```
df.columns
```

Out[17]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Driven_kms',
       'Fuel_Type', 'Selling_type', 'Transmission', 'Owner'],
      dtype='object')
```

In [18]:
```
df.shape
```

Out[18]: (301, 9)

In [19]:
```
df.describe().style.format(precision=2).background_gradient(cmap='RdBu')
```

Out[19]:

	Year	Selling_Price	Present_Price	Driven_kms	Owner
count	301.00	301.00	301.00	301.00	301.00
mean	2013.63	4.66	7.63	36947.21	0.04
std	2.89	5.08	8.64	38886.88	0.25
min	2003.00	0.10	0.32	500.00	0.00
25%	2012.00	0.90	1.20	15000.00	0.00
50%	2014.00	3.60	6.40	32000.00	0.00
75%	2016.00	6.00	9.90	48767.00	0.00
max	2018.00	35.00	92.60	500000.00	3.00

In [20]:
```
df.info()
```

```
<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   Driven_kms     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Selling_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
```

In [21]:
```
# Check Missing Values
print('Missing Values:')
df.isnull().sum()
```

Missing Values:

Out[21]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Driven_kms       0
Fuel_Type        0
Selling_type     0
Transmission     0
Owner            0
dtype: int64
```

In [22]:
```
# Data types of columns
print(" Data types of columns:")
print(df.dtypes)
```

```
 Data types of columns:
Car_Name          object
Year               int64
Selling_Price    float64
Present_Price    float64
Driven_kms         int64
Fuel_Type         object
Selling_type      object
Transmission      object
Owner              int64
dtype: object
```

In [23]:
```
# check for duplicate values
print('Duplicate Values:')
df.duplicated().sum()
```

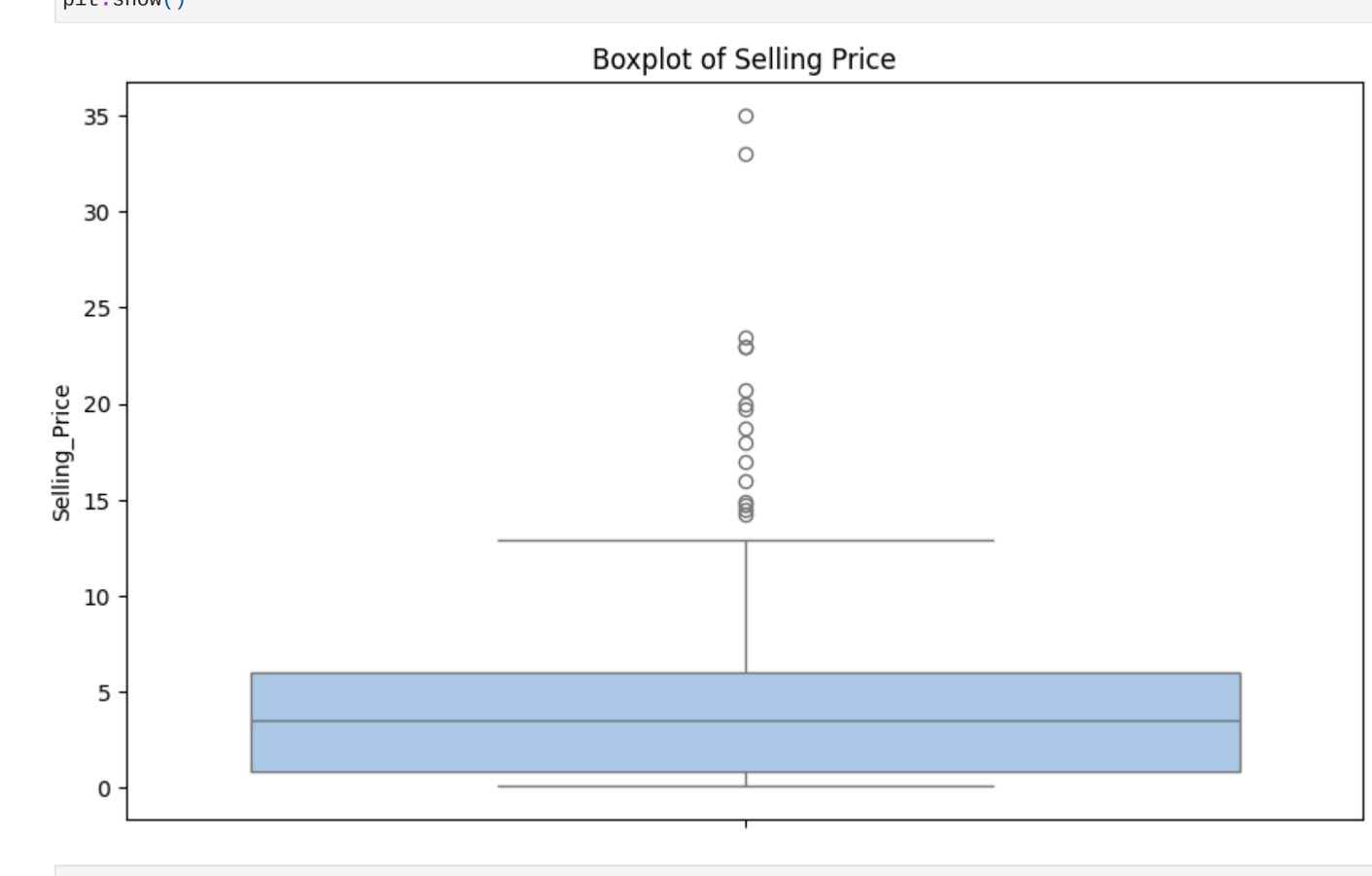
Duplicate Values:

Out[23]: 2

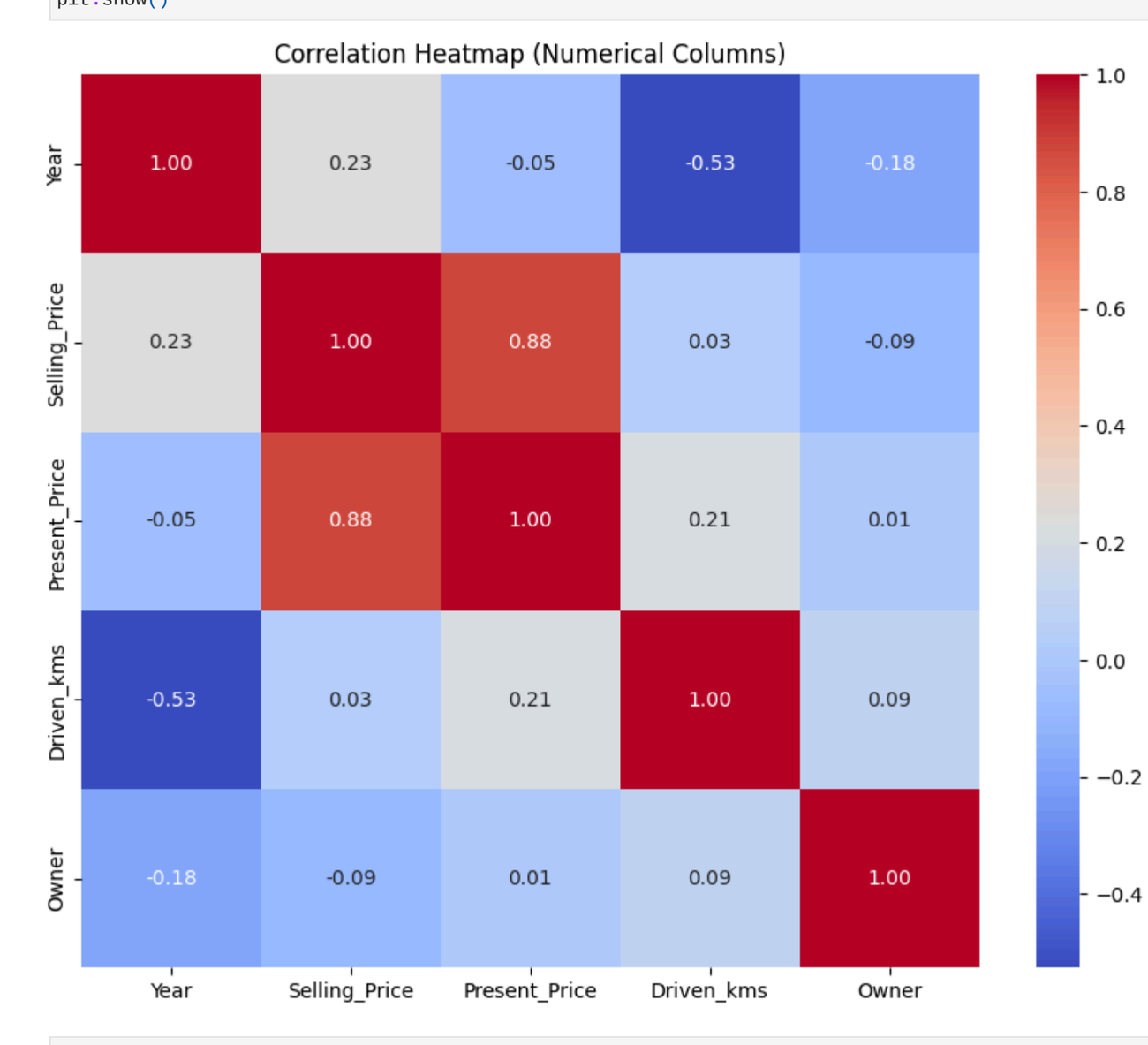
In [24]:
```
# drop duplicate values
df = df.drop_duplicates()
df.duplicated().sum()
```

Out[24]: 0

In [25]:
```
# Check for outliers using boxplots
plt.figure(figsize=(10, 6))
sns.boxplot(df['Selling_Price'],palette='pastel')
plt.title('Boxplot of Selling Price')
plt.show()
```

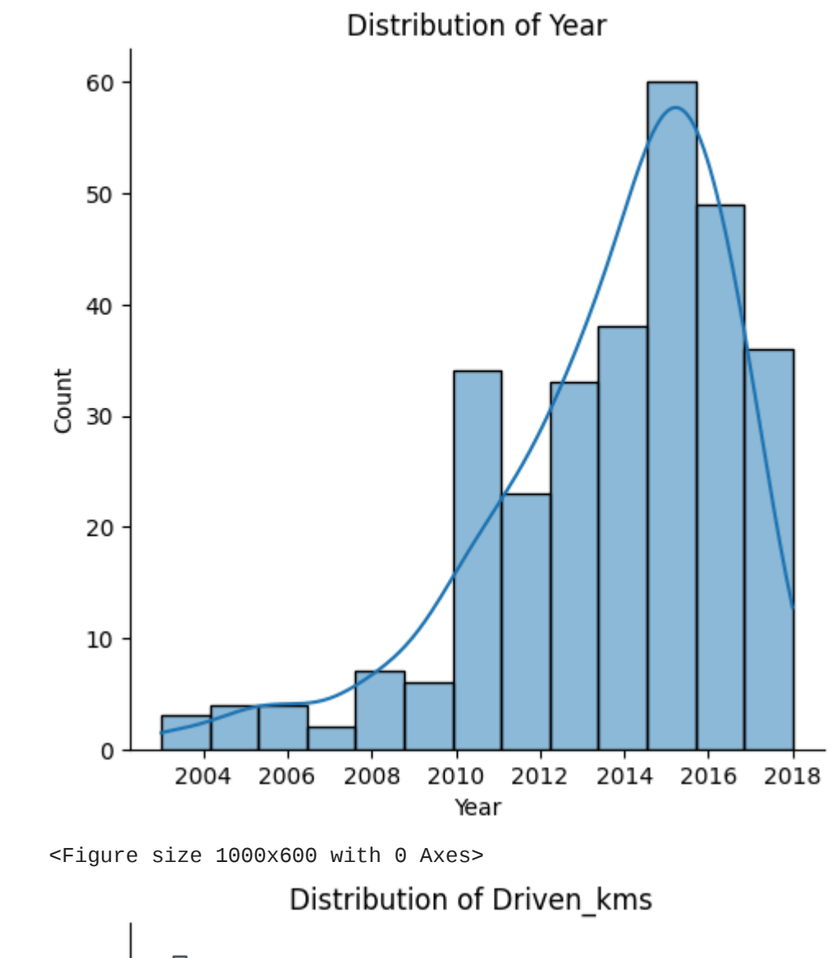


In [26]:
```
# Select numerical columns
numerical_columns = ['Year', 'Selling_Price', 'Present_Price', 'Driven_kms', 'Owner']

# Create a DataFrame containing only the numerical columns
numerical_df = df[numerical_columns]

# Calculate the correlation matrix for numerical columns
correlation_matrix = numerical_df.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap (Numerical Columns)')
plt.show()
```

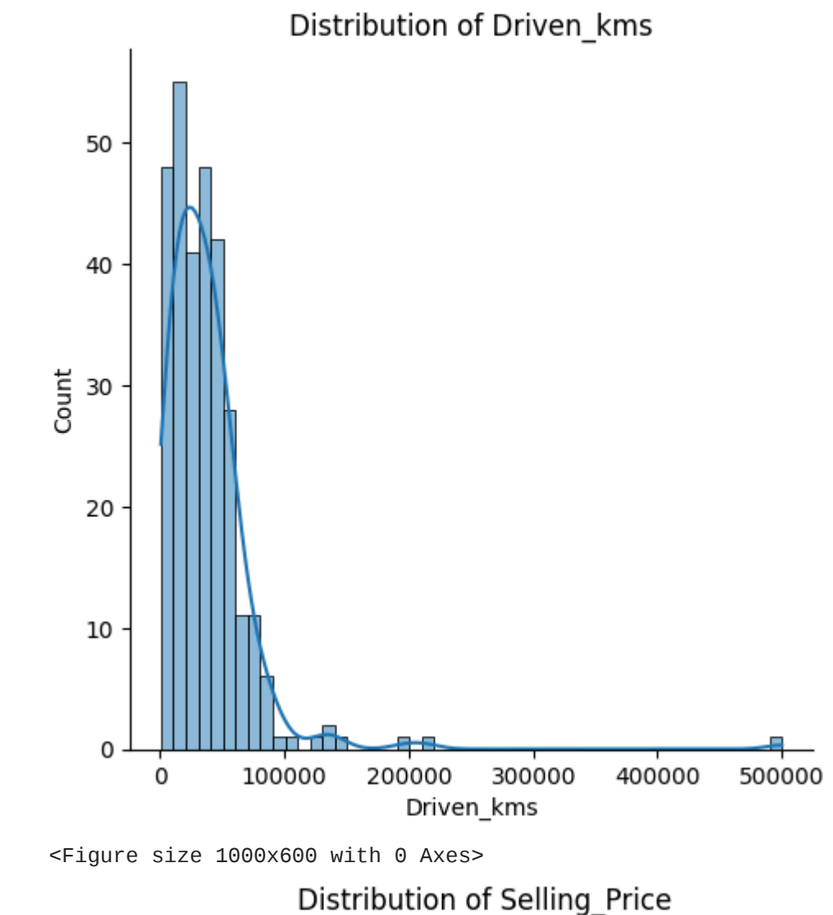


In [27]:
```
numerical_features = ['Year', 'Driven_kms', 'Selling_Price', 'Present_Price']
for feature in numerical_features:
    plt.figure(figsize=(10, 6))
    sns.displot(data=df, x=feature, kde=True)
    plt.title(f'Distribution of {feature}')
    plt.show()
```

<Figure size 1000x600 with 0 Axes>

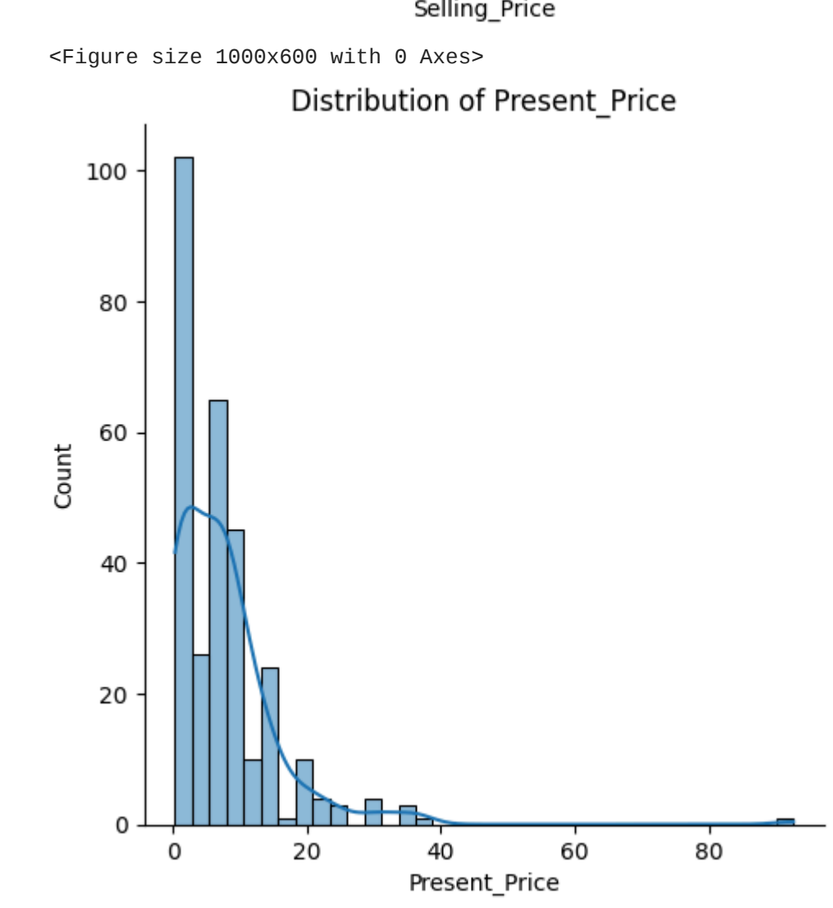


<Figure size 1000x600 with 0 Axes>

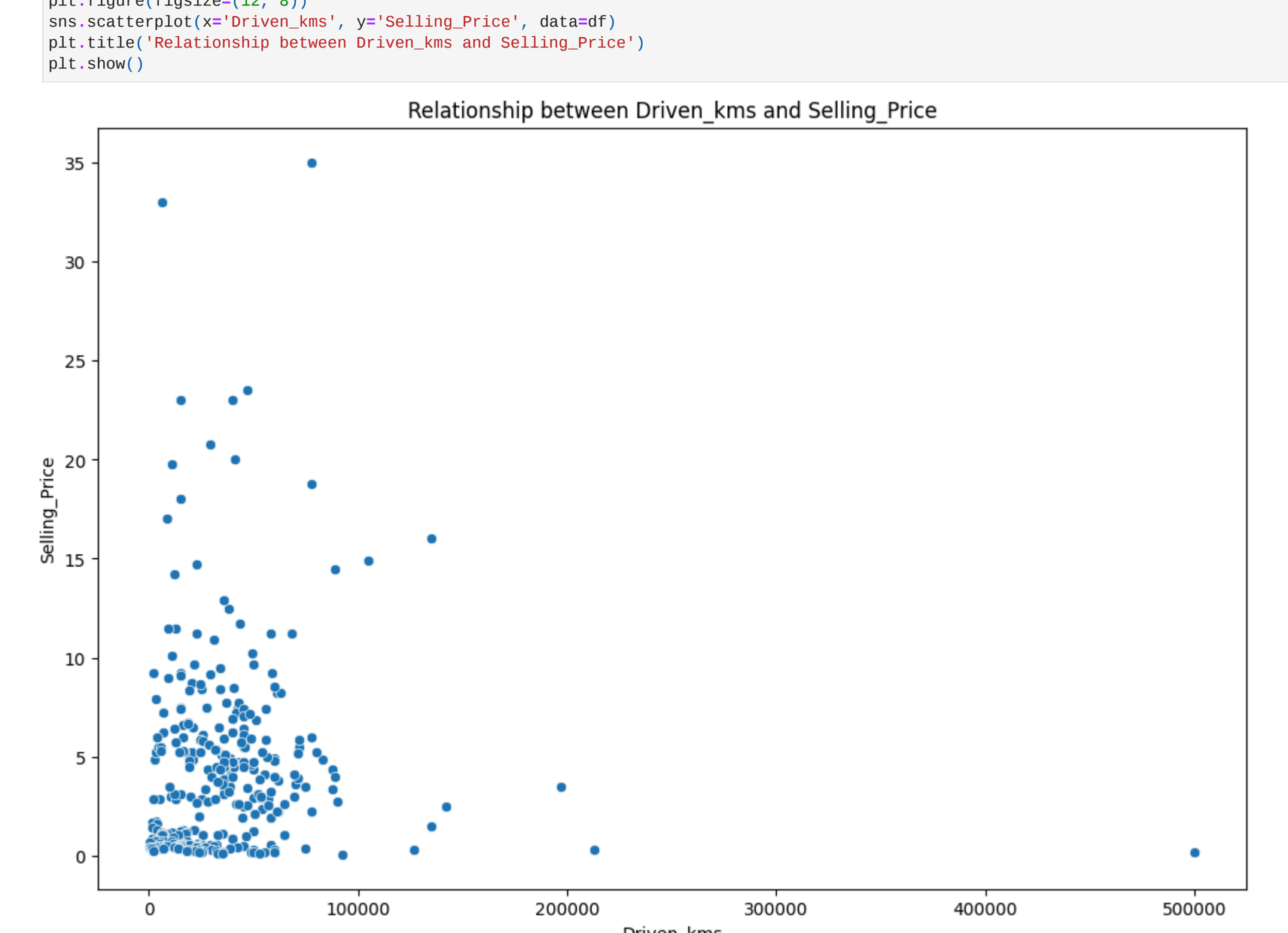


<Figure size 1000x600 with 0 Axes>

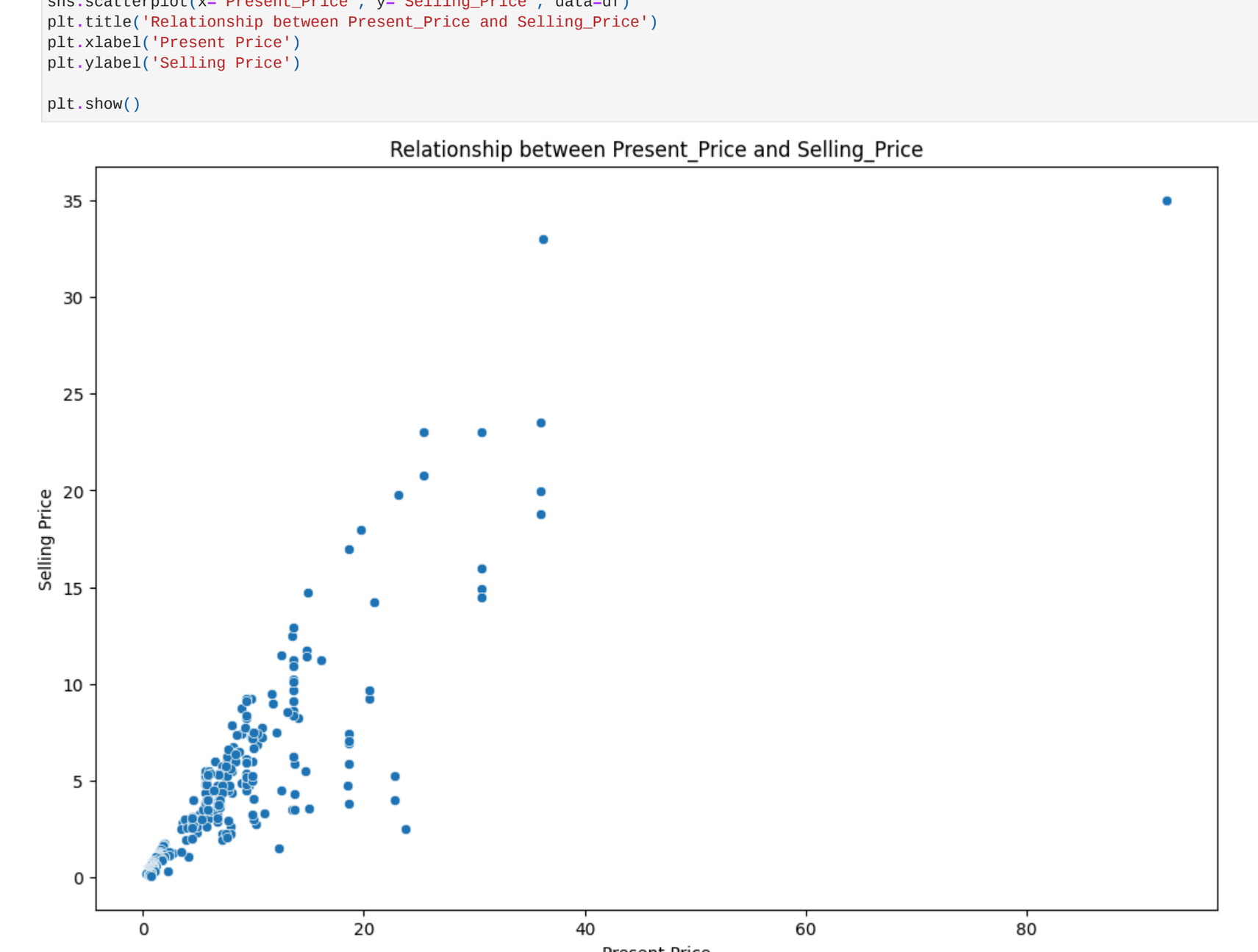
<Figure size 1000x600 with 0 Axes>



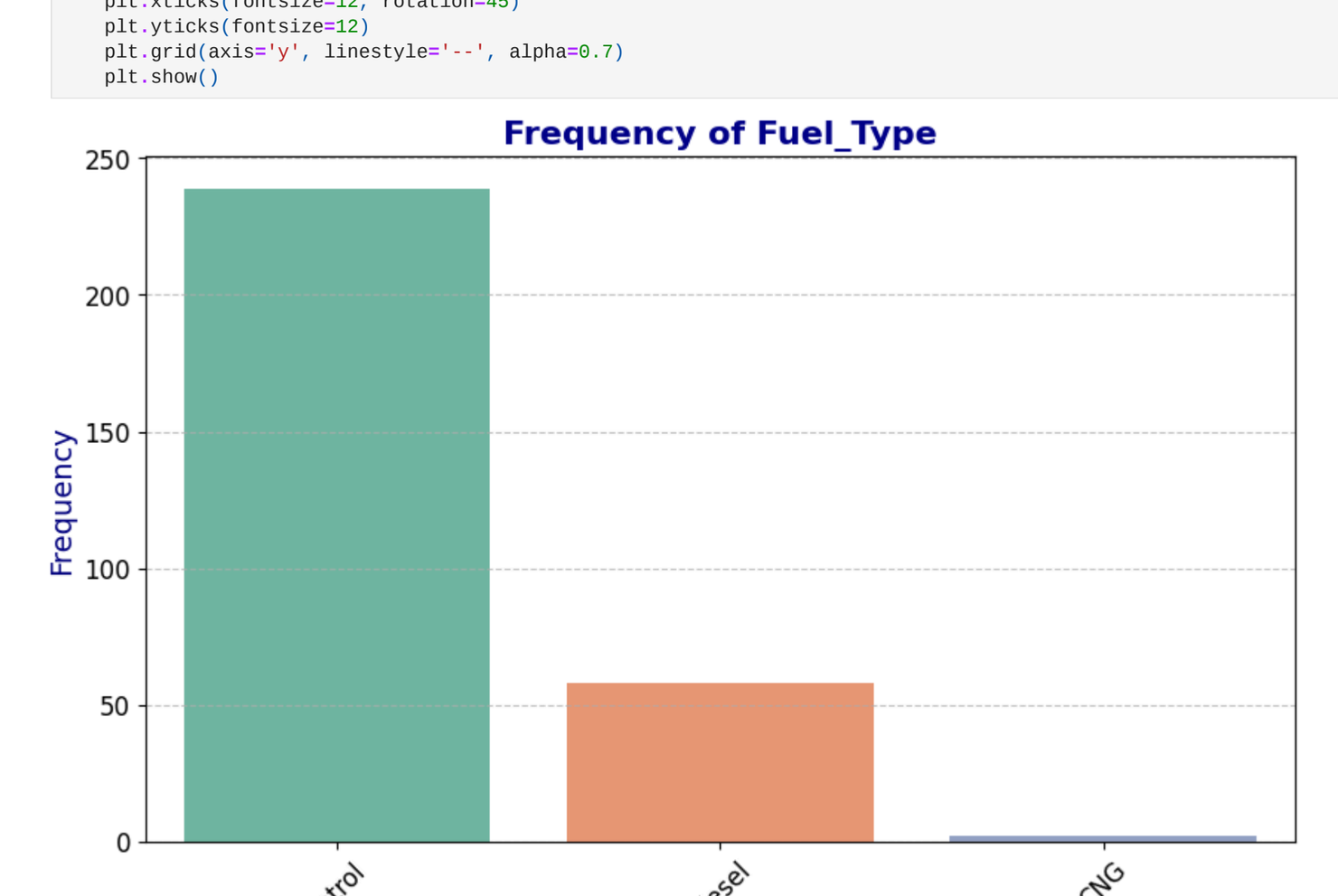
In [28]:
```
# Scatter plots
plt.figure(figsize=(12, 8))
sns.scatterplot(x='Driven_kms', y='Selling_Price', data=df)
plt.title('Relationship between Driven_kms and Selling_Price')
plt.show()
```

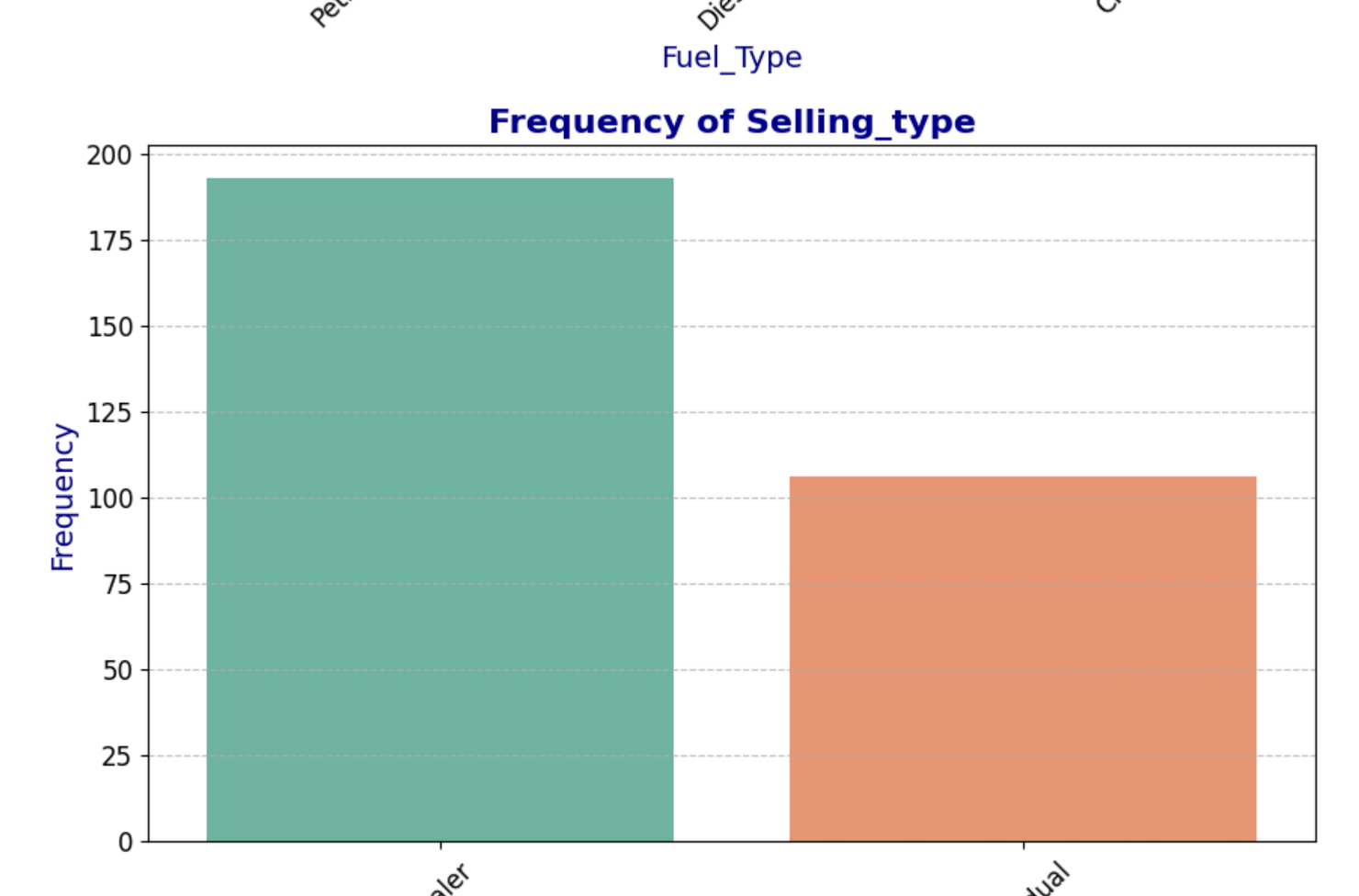


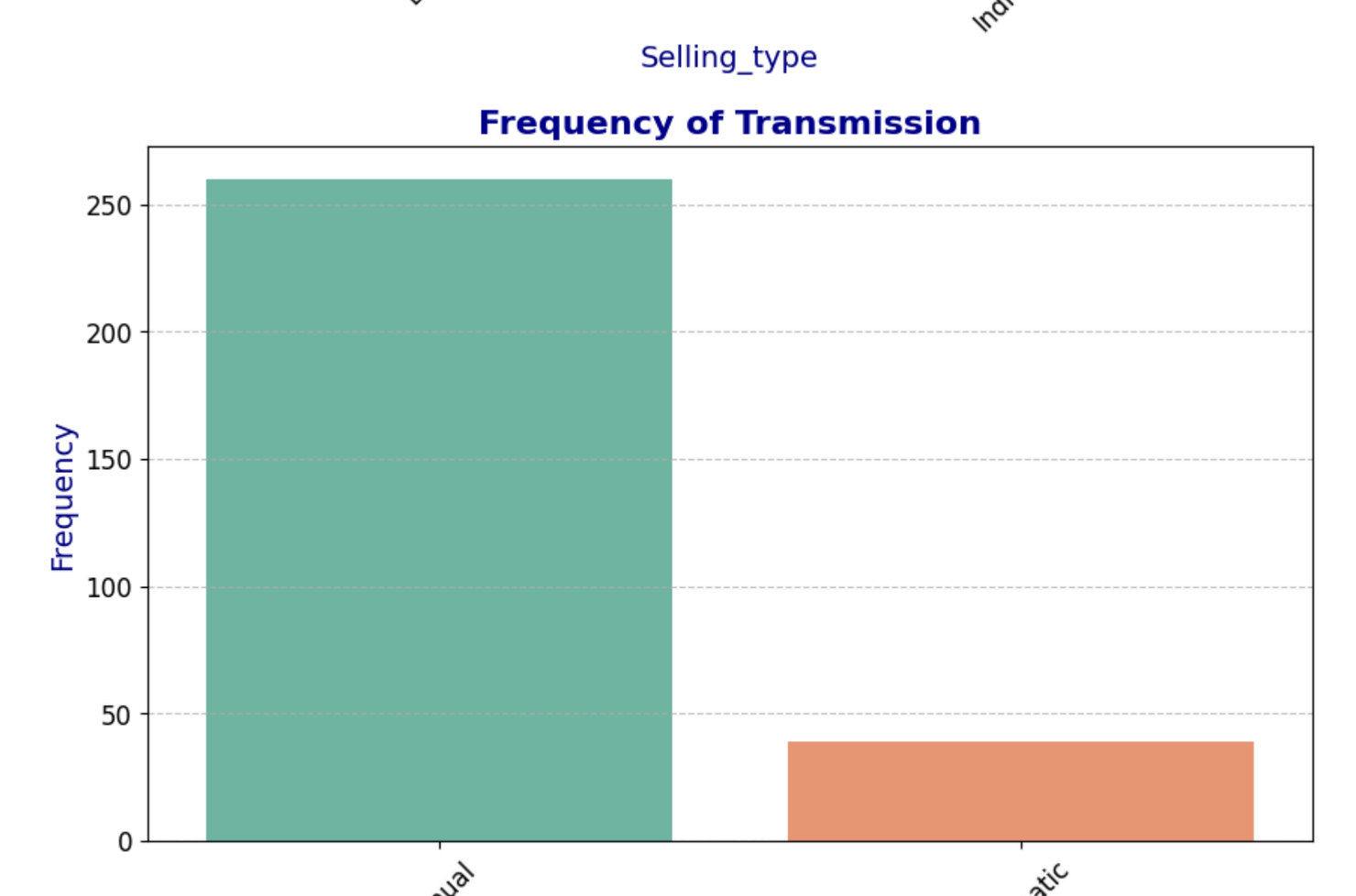
In [29]:
```
plt.figure(figsize=(12, 8))
sns.scatterplot(x='Present_Price', y='Selling_Price', data=df)
plt.title('Relationship between Present_Price and Selling_Price')
plt.xlabel('Present Price')
plt.ylabel('Selling Price')

plt.show()
```

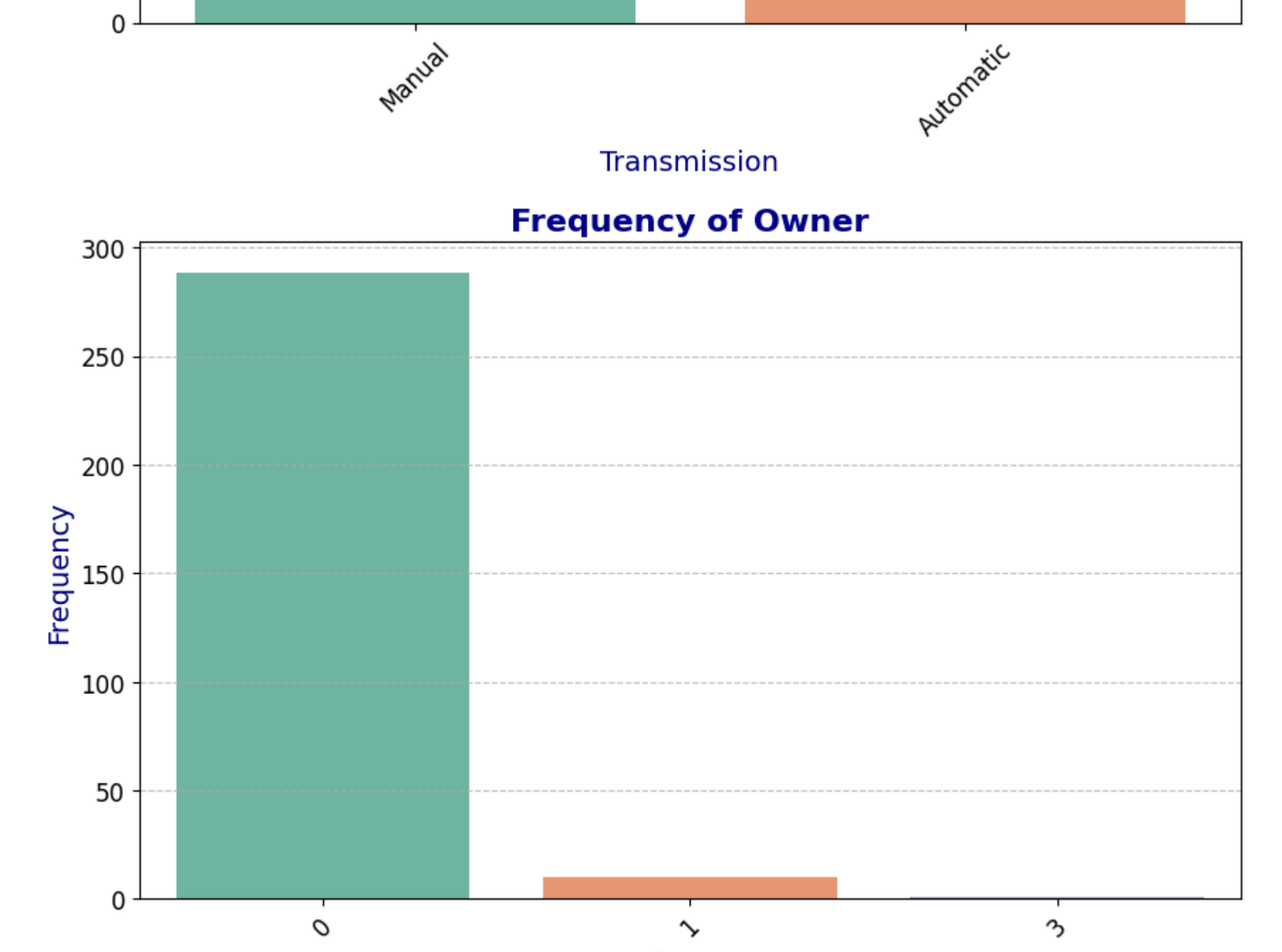


In [30]:
```
# Define a color palette for the plots
colors = sns.color_palette("Set2")
categorical_features = ['Fuel_Type', 'Selling_type', 'Transmission', 'Owner']
# Plot each categorical feature
for feature in categorical_features:
    plt.figure(figsize=(10, 6))
    sns.countplot(x=feature, data=df, palette=colors)
    plt.title(f'Frequency of {feature}', fontsize=16, color='darkblue', fontweight='bold')
    plt.xlabel(f'{feature}', fontsize=14, color='navy')
    plt.ylabel('Frequency', fontsize=14, color='navy')
    plt.xticks(fontsize=12, rotation=45)
    plt.yticks(fontsize=12)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```





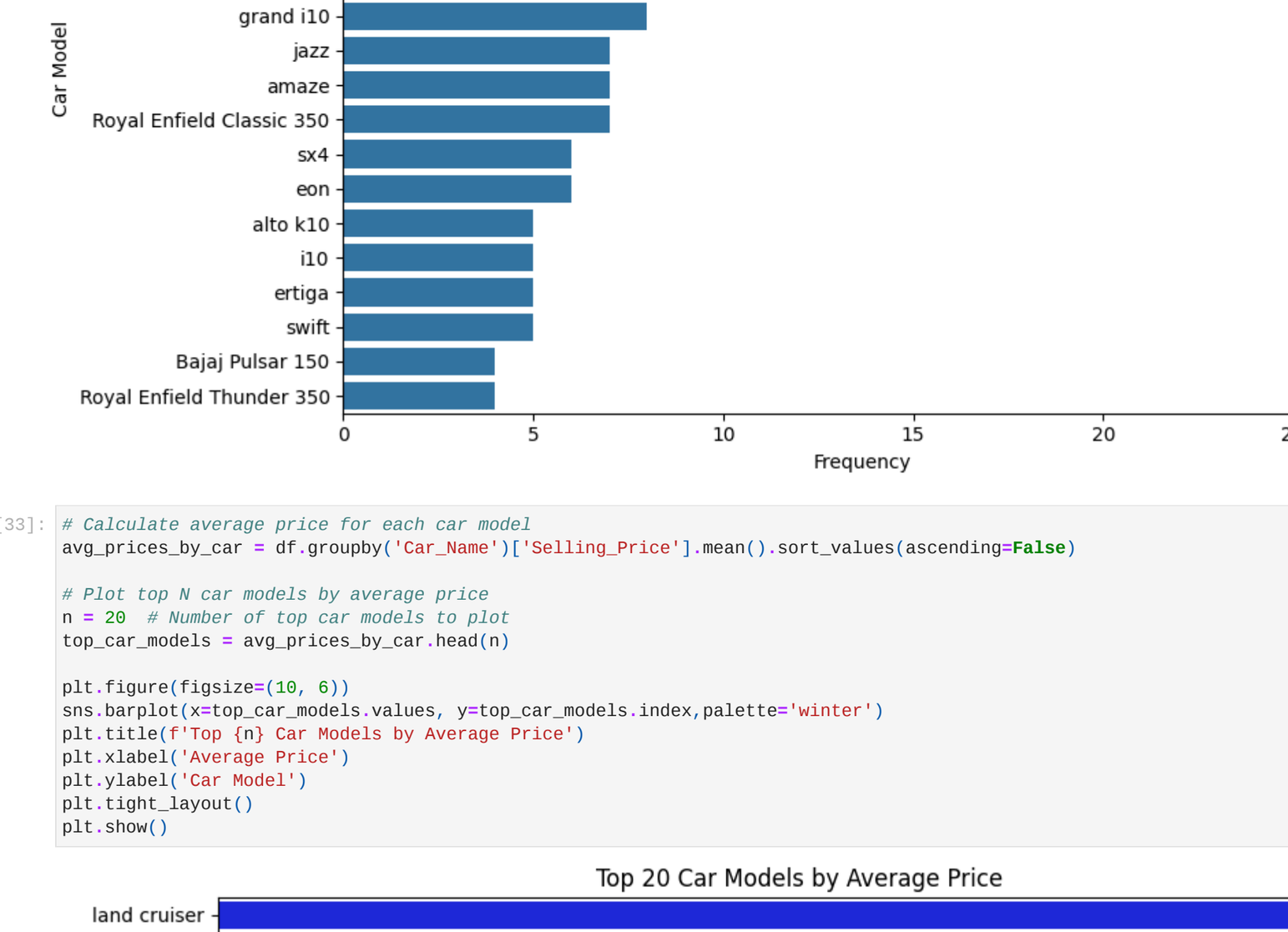




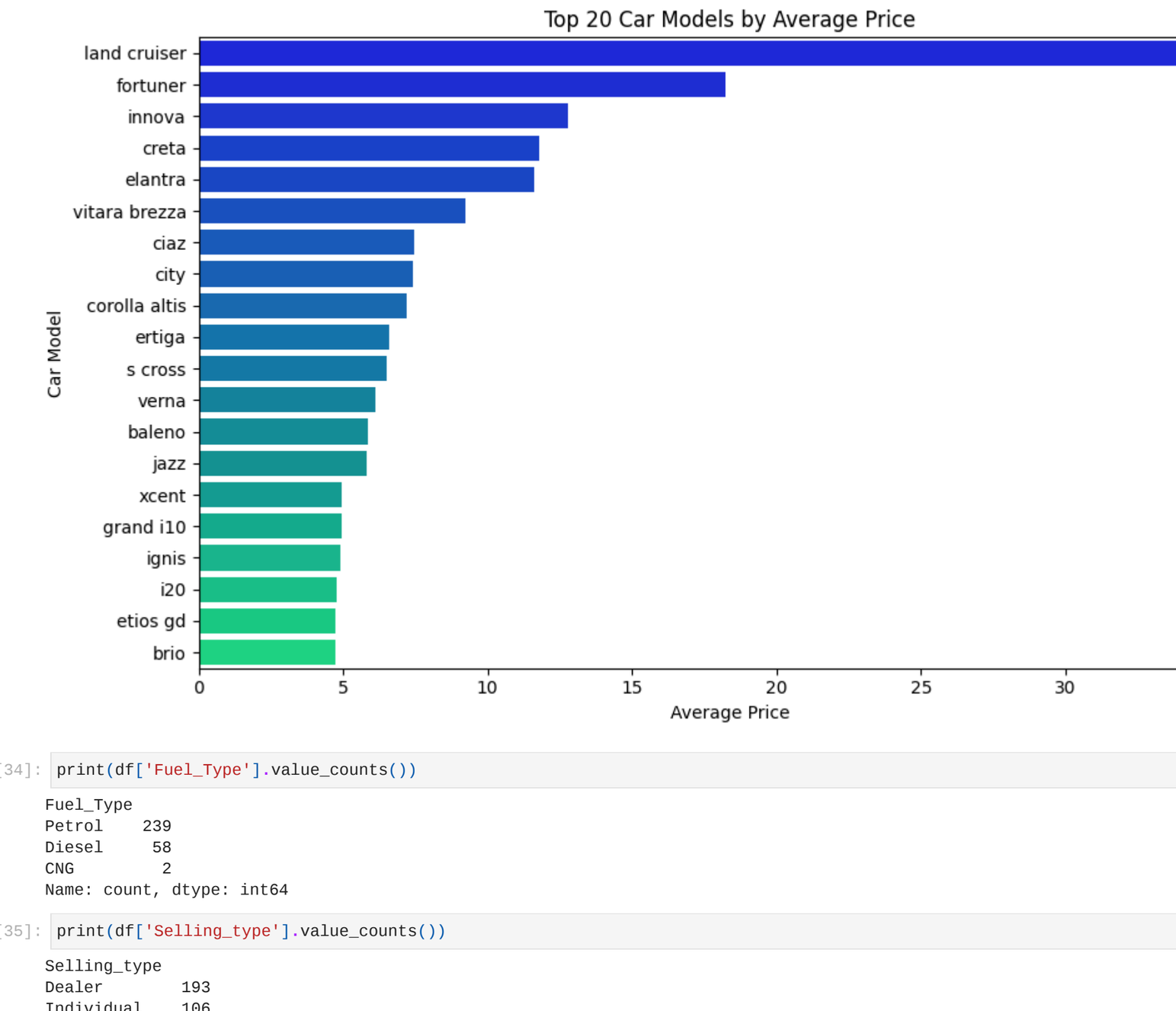
In [31]:
```
df.columns
```

Out[31]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Driven_kms',
       'Fuel_Type', 'Selling_type', 'Transmission', 'Owner'],
      dtype='object')
```

In [32]:
```
n = 20  # Number of top car models to plot
top_car_models = df['Car_Name'].value_counts().head(n)

plt.figure(figsize=(10, 6))
sns.barplot(x=top_car_models.values, y=top_car_models.index)
plt.title(f'Top {n} Car Models by Frequency')
plt.xlabel('Frequency')
plt.ylabel('Car Model')
plt.tight_layout()
plt.show()
```



In [33]:
```
# Calculate average price for each car model
avg_prices_by_car = df.groupby('Car_Name')['Selling_Price'].mean().sort_values(ascending=False)

# Plot top N car models by average price
n = 20  # Number of top car models to plot
top_car_models = avg_prices_by_car.head(n)

plt.figure(figsize=(10, 6))
sns.barplot(x=top_car_models.values, y=top_car_models.index,palette='winter')
plt.title(f'Top {n} Car Models by Average Price')
plt.xlabel('Average Price')
plt.ylabel('Car Model')
plt.tight_layout()
plt.show()
```



In [34]:
```
print(df['Fuel_Type'].value_counts())
```

```
Fuel_Type
Petrol    239
Diesel     58
CNG         2
Name: count, dtype: int64
```

In [35]:
```
print(df['Selling_type'].value_counts())
```

```
Selling_type
Dealer        193
Individual    106
Name: count, dtype: int64
```

In [36]:
```
print(df['Transmission'].value_counts())
```

```
Transmission
Manual       260
Automatic     39
Name: count, dtype: int64
```

In [37]:
```
# encoding "Fuel_Type" Column
df.replace({'Fuel_Type':{'Petrol':0,'Diesel':1,'CNG':2}},inplace=True)
# encoding "Seller_Type" Column
df.replace({'Selling_type':{'Dealer':0,'Individual':1}},inplace=True)
# encoding "Transmission" Column
df.replace({'Transmission':{'Manual':0,'Automatic':1}},inplace=True)
```

In [38]:
```
df.head()
```

Out[38]:

	Car_Name	Year	Selling_Price	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	0	0	0	0
1	sx4	2013	4.75	9.54	43000	1	0	0	0
2	ciaz	2017	7.25	9.85	6900	0	0	0	0
3	wagon r	2011	2.85	4.15	5200	0	0	0	0
4	swift	2014	4.60	6.87	42450	1	0	0	0

In [39]:
```
X = df.drop(['Car_Name','Selling_Price'],axis=1)
Y = df['Selling_Price']
X
```

Out[39]:

	Year	Present_Price	Driven_kms	Fuel_Type	Selling_type	Transmission	Owner
0	2014	5.59	27000	0	0	0	0
1	2013	9.54	43000	1	0	0	0
2	2017	9.85	6900	0	0	0	0
3	2011	4.15	5200	0	0	0	0
4	2014	6.87	42450	1	0	0	0
...	...	...	...	...	...	...	...
296	2016	11.60	33988	1	0	0	0
297	2015	5.90	60000	0	0	0	0
298	2009	11.00	87934	0	0	0	0
299	2017	12.50	9000	1	0	0	0
300	2016	5.90	5464	0	0	0	0

299 rows × 7 columns

In [41]:
```
Y
```

Out[41]:
```
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       ...  
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 299, dtype: float64
```

In [42]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [43]:
```
# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[43]:

LinearRegression

LinearRegression()

In [44]:
```
# Evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print('Mean Squared Error:', mse)
print('R-squared:', r2)
```

```
Mean Squared Error: 6.675444100493426
R-squared: 0.7409934356413088
```

In [45]:
```
# Make predictions
new_car = [[2022, 20000, 0, 1, 1, 0, 0]]  # Example new car data
predicted_price = model.predict(new_car)
print('Predicted Selling Price:', predicted_price[0])
```