```
import pandas as pd

df = pd.read_csv('/content/car data.csv')

print("First 5 rows of the DataFrame:")
print(df.head())

print("\nDataFrame Info:")
df.info()

print("\nDescriptive Statistics:")
print(df.describe())

print("\nMissing values per column:")
print(df.isnull().sum())
```

```
First 5 rows of the DataFrame:
  Car_Name  Year  Selling_Price  Present_Price  Driven_kms 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

  Selling_type Transmission  Owner
0       Dealer       Manual      0
1       Dealer       Manual      0
2       Dealer       Manual      0
3       Dealer       Manual      0
4       Dealer       Manual      0

DataFrame 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

Descriptive Statistics:
              Year  Selling_Price  Present_Price     Driven_kms       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       8.642584   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

Missing values per column:
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
```

```
print("Missing values per column (re-verification):")
print(df.isnull().sum())

print("\nUnique values and counts for 'Fuel_Type':")
print(df['Fuel_Type'].value_counts())
print("\nUnique values and counts for 'Selling_type':")
print(df['Selling_type'].value_counts())

print("\nUnique values and counts for 'Transmission':")
print(df['Transmission'].value_counts())

print("\nDescriptive Statistics for numerical columns (re-verification):")
print(df[['Driven_kms', 'Year', 'Selling_Price', 'Present_Price', 'Owner']].describe())
```

```
Missing values per column (re-verification):
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

Unique values and counts for 'Fuel_Type':
Fuel_Type
Petrol    239
Diesel     60
CNG         2
Name: count, dtype: int64

Unique values and counts for 'Selling_type':
Selling_type
Dealer        195
Individual    106
Name: count, dtype: int64

Unique values and counts for 'Transmission':
Transmission
Manual       261
Automatic     40
Name: count, dtype: int64

Descriptive Statistics for numerical columns (re-verification):
          Driven_kms         Year  Selling_Price  Present_Price       Owner
count     301.000000   301.000000     301.000000     301.000000  301.000000
mean    36947.205980  2013.627907       4.661296       7.628472    0.043189
std     38886.883882     2.891554       5.082812       8.642584    0.247915
min       500.000000  2003.000000       0.100000       0.320000    0.000000
25%     15000.000000  2012.000000       0.900000       1.200000    0.000000
50%     32000.000000  2014.000000       3.600000       6.400000    0.000000
75%     48767.000000  2016.000000       6.000000       9.900000    0.000000
max    500000.000000  2018.000000      35.000000      92.600000    3.000000
```

```
df = pd.get_dummies(df, columns=['Fuel_Type', 'Selling_type', 'Transmission'], drop_first=True)

df.drop('Car_Name', axis=1, inplace=True)

print("DataFrame after one-hot encoding and dropping 'Car_Name':")
print(df.head())
print("\nDataFrame Info after preprocessing:")
df.info()
```

```
DataFrame after one-hot encoding and dropping 'Car_Name':
   Year  Selling_Price  Present_Price  Driven_kms  Owner  Fuel_Type_Diesel  \
0  2014           3.35           5.59       27000      0             False
1  2013           4.75           9.54       43000      0              True
2  2017           7.25           9.85        6900      0             False
3  2011           2.85           4.15        5200      0             False
4  2014           4.60           6.87       42450      0              True

   Fuel_Type_Petrol  Selling_type_Individual  Transmission_Manual
0              True                    False                 True
1             False                    False                 True
2              True                    False                 True
3              True                    False                 True
4             False                    False                 True

DataFrame Info after preprocessing:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
 ---  ------                  --------------  -----
  0   Year                    301 non-null    int64
  1   Selling_Price           301 non-null    float64
  2   Present_Price           301 non-null    float64
  3   Driven_kms              301 non-null    int64
  4   Owner                   301 non-null    int64
  5   Fuel_Type_Diesel        301 non-null    bool
  6   Fuel_Type_Petrol        301 non-null    bool
  7   Selling_type_Individual  301 non-null    bool
  8   Transmission_Manual     301 non-null    bool
 dtypes: bool(4), float64(2), int64(3)
 memory usage: 13.1 KB
```

```
df['Age_of_Car'] = 2024 - df['Year']

print("DataFrame with new 'Age_of_Car' feature:")
print(df.head())
```

```
DataFrame with new 'Age_of_Car' feature:
   Year  Selling_Price  Present_Price  Driven_kms  Owner  Fuel_Type_Diesel  \
0  2014           3.35           5.59       27000      0             False
1  2013           4.75           9.54       43000      0              True
2  2017           7.25           9.85        6900      0             False
3  2011           2.85           4.15        5200      0             False
4  2014           4.60           6.87       42450      0              True

   Fuel_Type_Petrol  Selling_type_Individual  Transmission_Manual  Age_of_Car
0              True                    False                 True          10
1             False                    False                 True          11
2              True                    False                 True           7
3              True                    False                 True          13
4             False                    False                 True          10
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# 1. Separate features (X) from the target variable (y)
X = df.drop(['Selling_Price', 'Year'], axis=1)
y = df['Selling_Price']

print("Features (X) shape:", X.shape)
print("Target (y) shape:", y.shape)

# 2. 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)

print("\nTraining set shape (X_train, y_train):", X_train.shape, y_train.shape)
print("Testing set shape (X_test, y_test):", X_test.shape, y_test.shape)

# 3. Import LinearRegression (already imported above)

# 4. Instantiate a LinearRegression model
model = LinearRegression()

# 5. Train the model using the training data
model.fit(X_train, y_train)

print("\nLinear Regression model trained successfully.")
```

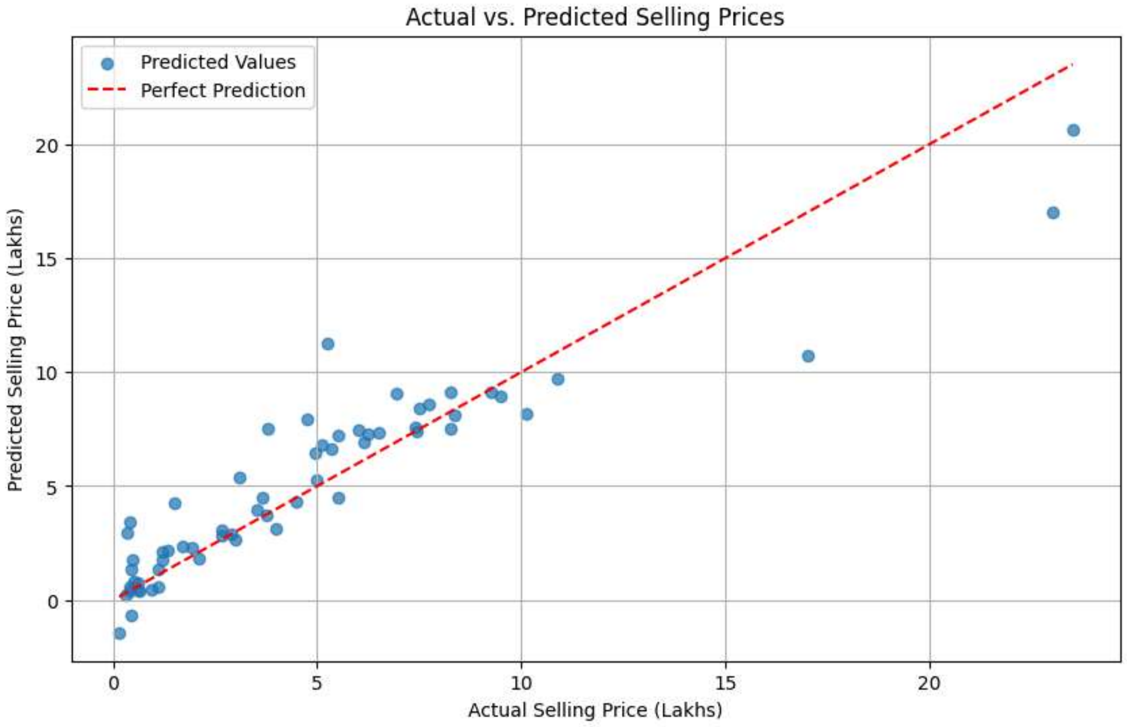
```
Features (X) shape: (301, 8)
Target (y) shape: (301,)

Training set shape (X_train, y_train): (240, 8) (240,)
Testing set shape (X_test, y_test): (61, 8) (61,)

Linear Regression model trained successfully.
```

```
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mse**0.5 # Calculate RMSE
```

```
print("\nModel Evaluation:")
print(f"R-squared (R2): {r2:.4f}")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
```

```
Model Evaluation:
R-squared (R2): 0.8489
Mean Absolute Error (MAE): 1.2164
Mean Squared Error (MSE): 3.4813
Root Mean Squared Error (RMSE): 1.8658
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.xlabel('Actual Selling Price (Lakhs)')
plt.ylabel('Predicted Selling Price (Lakhs)')
plt.title('Actual vs. Predicted Selling Prices')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--') # Add a diagonal line for perfect
plt.legend(['Predicted Values', 'Perfect Prediction'])
plt.grid(True)
plt.show()
```


Actual vs. Predicted Selling Prices
● Predicted Values
--- Perfect Prediction
Predicted Selling Price (Lakhs)
Actual Selling Price (Lakhs)