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

```
In [7]: plt.rcParams['figure.figsize'] = [19, 8]
        warnings.filterwarnings('ignore')
```

```
In [8]: try:
          cars_df = pd.read_csv("C:\\Users\\megha\\Downloads\\cars_df.csv",encoding="utf-8",nrows=20)
          print("Dataset loaded successfully.")
        except FileNotFoundError:
          print("Error: 'cars.csv' not found. Please ensure the file is in the correct directory and update the path.")
        if not cars_df.empty:
          print("\n--- Initial Data Details ---")
          print("Dataset shape (rows, columns):", {cars_df.shape})
          print("\nColumn Info:")
          cars_df.info()
```

```
Dataset loaded successfully.

--- Initial Data Details ---
Dataset shape (rows, columns): {(20, 15)}

Column Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Make         20 non-null     object
 1   Model        20 non-null     object
 2   Type         20 non-null     object
 3   Origin       20 non-null     object
 4   DriveTrain   20 non-null     object
 5   MSRP         20 non-null     float64
 6   Invoice      20 non-null     float64
 7   EngineSize   20 non-null     float64
 8   Cylinders    20 non-null     float64
 9   Horsepower   20 non-null     float64
 10  MPG_City     20 non-null     float64
 11  MPG_Highway  20 non-null     float64
 12  Weight       20 non-null     float64
 13  Wheelbase    20 non-null     float64
 14  Length       20 non-null     float64
dtypes: float64(10), object(5)
memory usage: 2.5+ KB
```

```
In [6]: print("\nFirst 5 rows of the dataset:")
        print(cars_df.head())
```

```
First 5 rows of the dataset:
    Make           Model   Type Origin DriveTrain     MSRP  Invoice  \
0  Acura             MDX    SUV   Asia        All  36945.0  33337.0
1  Acura  RSX Type S 2dr  Sedan   Asia      Front  23820.0  21761.0
2  Acura         TSX 4dr  Sedan   Asia      Front  26990.0  24647.0
3  Acura          TL 4dr  Sedan   Asia      Front  33195.0  30299.0
4  Acura      3.5 RL 4dr  Sedan   Asia      Front  43755.0  39014.0

   EngineSize  Cylinders  Horsepower  MPG_City  MPG_Highway  Weight  \
0         3.5        6.0       265.0      17.0         23.0  4451.0
1         2.0        4.0       200.0      24.0         31.0  2778.0
2         2.4        4.0       200.0      22.0         29.0  3230.0
3         3.2        6.0       270.0      20.0         28.0  3575.0
4         3.5        6.0       225.0      18.0         24.0  3880.0

   Wheelbase  Length
0      106.0   189.0
1      101.0   172.0
2      105.0   183.0
3      108.0   186.0
4      115.0   197.0
```

```
In [9]: print("\nLast 5 rows of the dataset:")
        print(cars_df.tail())
```

```
Last 5 rows of the dataset:
    Make                          Model   Type  Origin DriveTrain     MSRP  \
15  Audi  A4 3.0 Quattro convertible 2dr  Sedan  Europe        All  44240.0
16  Audi        A6 2.7 Turbo Quattro 4dr  Sedan  Europe        All  42840.0
17  Audi              A6 4.2 Quattro 4dr  Sedan  Europe        All  49690.0
18  Audi                A8 L Quattro 4dr  Sedan  Europe        All  69190.0
19  Audi                  S4 Quattro 4dr  Sedan  Europe        All  48040.0

    Invoice  EngineSize  Cylinders  Horsepower  MPG_City  MPG_Highway  Weight  \
15  40075.0         3.0        6.0       220.0      18.0         25.0  4013.0
16  38840.0         2.7        6.0       250.0      18.0         25.0  3836.0
17  44936.0         4.2        8.0       300.0      17.0         24.0  4024.0
18  64740.0         4.2        8.0       330.0      17.0         24.0  4399.0
19  43556.0         4.2        8.0       340.0      14.0         20.0  3825.0

    Wheelbase  Length
15      105.0   180.0
16      109.0   192.0
17      109.0   193.0
18      121.0   204.0
19      104.0   179.0
```

```
In [10]: print("\n--- Descriptive Statistics for Numerical Columns ---")
         print(cars_df.describe())
```

```
--- Descriptive Statistics for Numerical Columns ---
               MSRP       Invoice  EngineSize  Cylinders  Horsepower  \
count     20.000000     20.000000   20.000000  20.000000   20.000000
mean   41748.500000  37817.150000    3.060000   5.900000  238.750000
std    15151.963838  13665.271774    0.703675   1.209611   47.040716
min    23820.000000  21761.000000    1.800000   4.000000  170.000000
25%    33371.250000  30349.250000    2.925000   6.000000  220.000000
50%    38292.500000  34664.500000    3.000000   6.000000  220.000000
75%    44705.000000  40331.250000    3.500000   6.000000  266.250000
max    89765.000000  79978.000000    4.200000   8.000000  340.000000

       MPG_City  MPG_Highway       Weight   Wheelbase      Length
count  20.00000    20.000000    20.000000   20.000000   20.000000
mean   18.90000    26.000000  3693.700000  107.100000  185.300000
std     2.44734     2.846974   403.228302    5.066921    8.657823
min    14.00000    20.000000  2778.000000  100.000000  172.000000
25%    17.00000    24.000000  3536.250000  104.000000  179.000000
50%    18.00000    25.000000  3726.000000  105.000000  181.500000
75%    20.00000    28.000000  3883.250000  109.000000  192.000000
max    24.00000    31.000000  4451.000000  121.000000  204.000000
```

```
In [13]: print("\nNull values before handling:")
         cars_df = pd.read_csv("C:\\Users\\megha\\Downloads\\cars_df.csv", na_values=["NA", "Na", "?", "null", " "])
         print(cars_df.isnull().sum())
         cars_df.rename(columns={'MSRP': 'MRP', 'MPG_City': 'Mileage_City',
         'MPG_Highway': 'Mileage_Highway'}, inplace=True)
         print("\nColumns renamed. New column names:")
         print(cars_df.columns)
         print("\n--- Data Cleaning ---")
         initial_rows = len(cars_df)
         print(f"Number of duplicate rows found: {cars_df.duplicated().sum()}")
         cars_df.drop_duplicates(inplace=True)
         print(f"Duplicate rows dropped. Dataset shape is now: {cars_df.shape}")
```

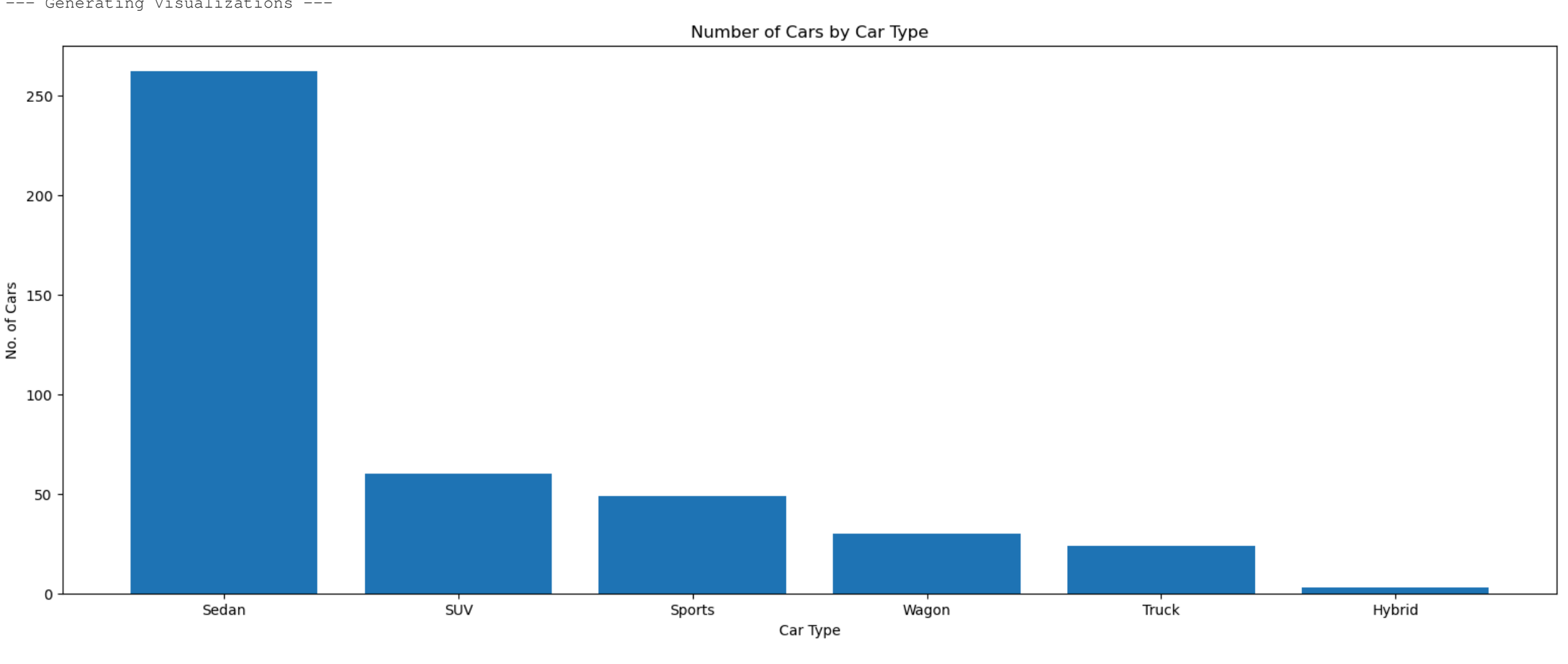
```
Null values before handling:
Make           0
Model          0
Type           0
Origin         0
DriveTrain     0
MSRP           0
Invoice        0
EngineSize     0
Cylinders      2
Horsepower     0
MPG_City       0
MPG_Highway    0
Weight         0
Wheelbase      0
Length         0
dtype: int64

Columns renamed. New column names:
Index(['Make', 'Model', 'Type', 'Origin', 'DriveTrain', 'MRP', 'Invoice',
       'EngineSize', 'Cylinders', 'Horsepower', 'Mileage_City',
       'Mileage_Highway', 'Weight', 'Wheelbase', 'Length'],
      dtype='object')

--- Data Cleaning ---
Number of duplicate rows found: 0
Duplicate rows dropped. Dataset shape is now: (428, 15)
```

```
In [14]: if 'Cylinders' in cars_df.columns and cars_df['Cylinders'].isnull().any():
           median_cylinders = cars_df['Cylinders'].median()
           cars_df['Cylinders'].fillna(median_cylinders, inplace=True)
           print(f"\nMissing 'Cylinders' values filled with median value: {median_cylinders}")
         print("\nNull values after handling:")
         print(cars_df.isnull().sum())
         if 'Cylinders' in cars_df.columns:
           cars_df['Cylinders'] = cars_df['Cylinders'].astype(np.int64)
           print("\n'Cylinders' column data type converted to int64.")
           print(cars_df.dtypes)
```

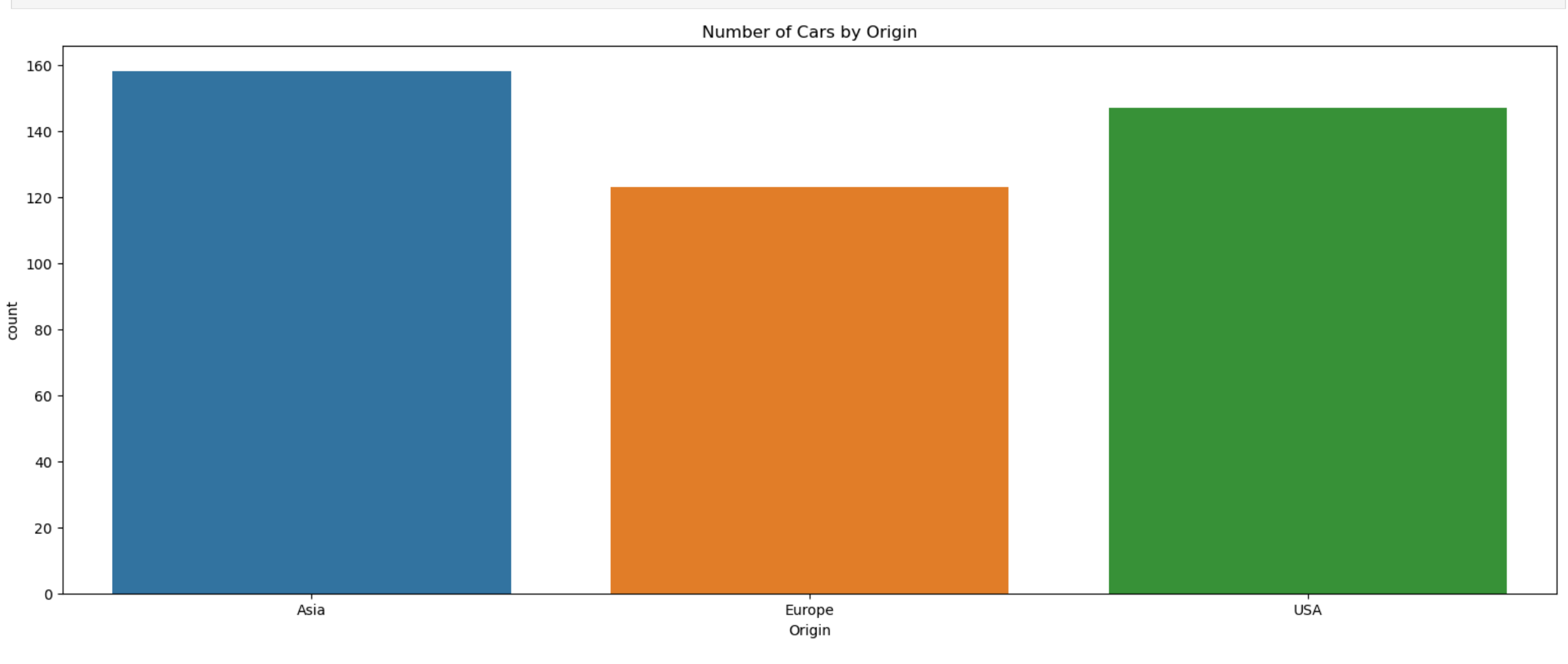
```
Missing 'Cylinders' values filled with median value: 6.0

Null values after handling:
Make               0
Model              0
Type               0
Origin             0
DriveTrain         0
MRP                0
Invoice            0
EngineSize         0
Cylinders          0
Horsepower         0
Mileage_City       0
Mileage_Highway    0
Weight             0
Wheelbase          0
Length             0
dtype: int64

'Cylinders' column data type converted to int64.
Make                object
Model               object
Type                object
Origin              object
DriveTrain          object
MRP                float64
Invoice            float64
EngineSize         float64
Cylinders            int64
Horsepower         float64
Mileage_City       float64
Mileage_Highway    float64
Weight             float64
Wheelbase          float64
Length             float64
dtype: object
```

```
In [15]: print("\n--- Generating Visualizations ---")
         plt.figure(figsize=(19, 7))
         type_counts = cars_df['Type'].value_counts()
         plt.title("Number of Cars by Car Type")
         plt.bar(x=type_counts.index, height=type_counts.values)
         plt.xlabel("Car Type")
         plt.ylabel("No. of Cars")
         plt.show()
```

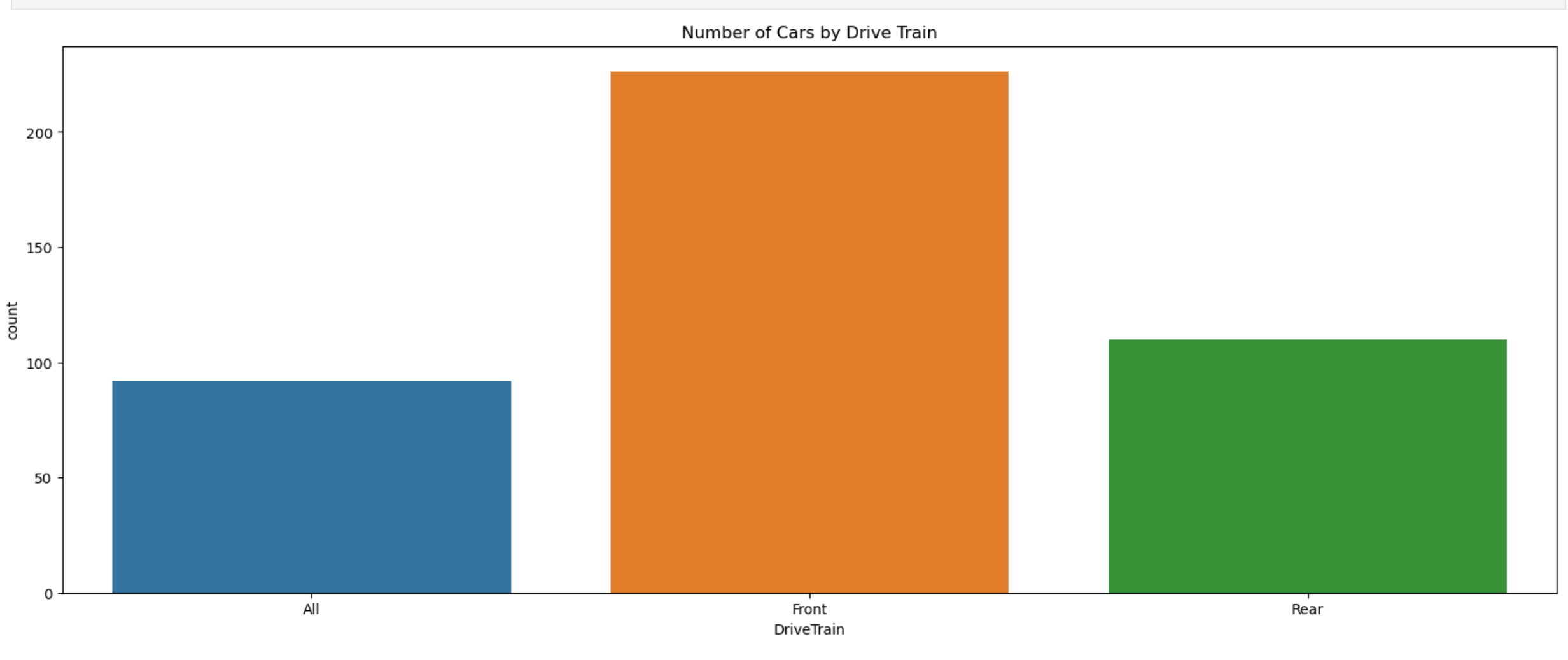
```
--- Generating Visualizations ---
```



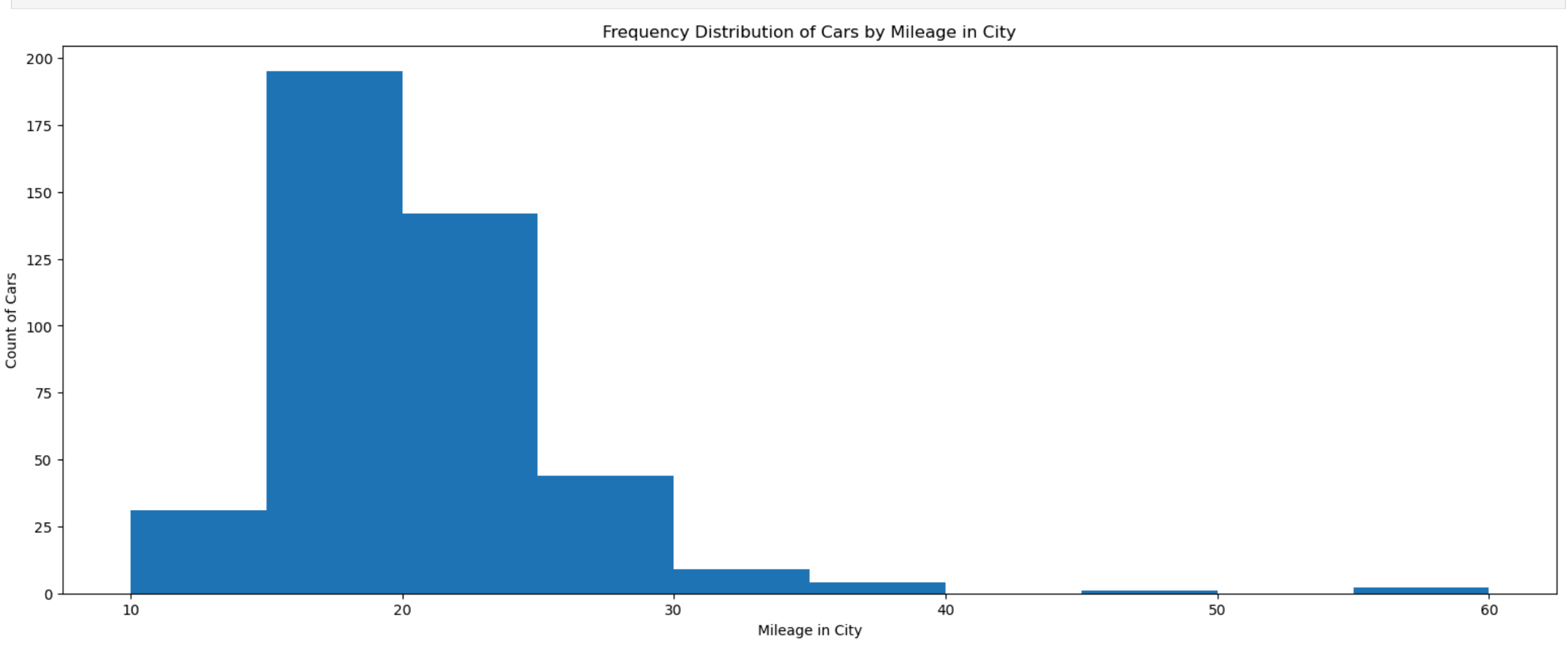

```
In [16]: plt.figure(figsize=(19, 7))
         plt.title("Number of Cars by Origin")
         sns.countplot(data=cars_df, x="Origin")
         plt.show()
```



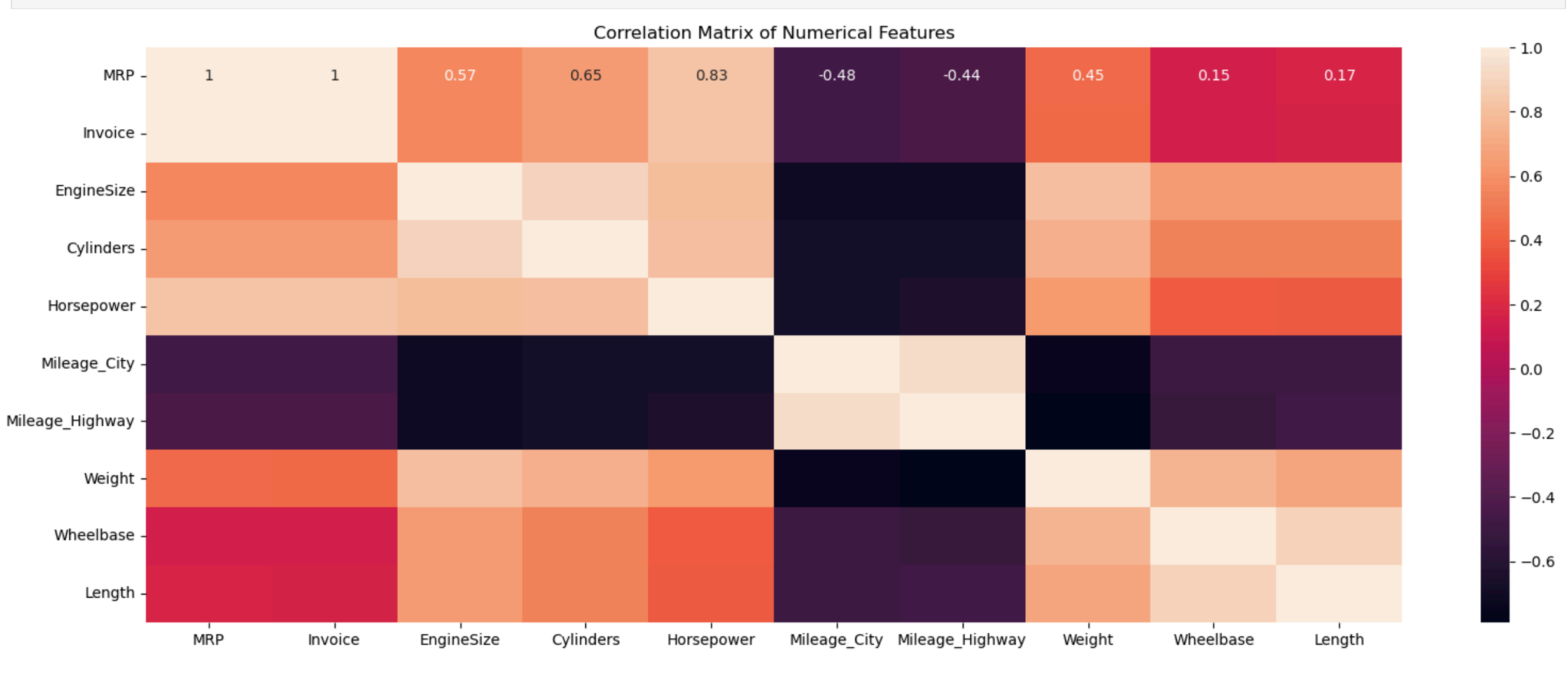

```
In [17]: plt.figure(figsize=(19, 7))
         plt.title("Number of Cars by Drive Train")
         sns.countplot(data=cars_df, x="DriveTrain")
         plt.show()
```



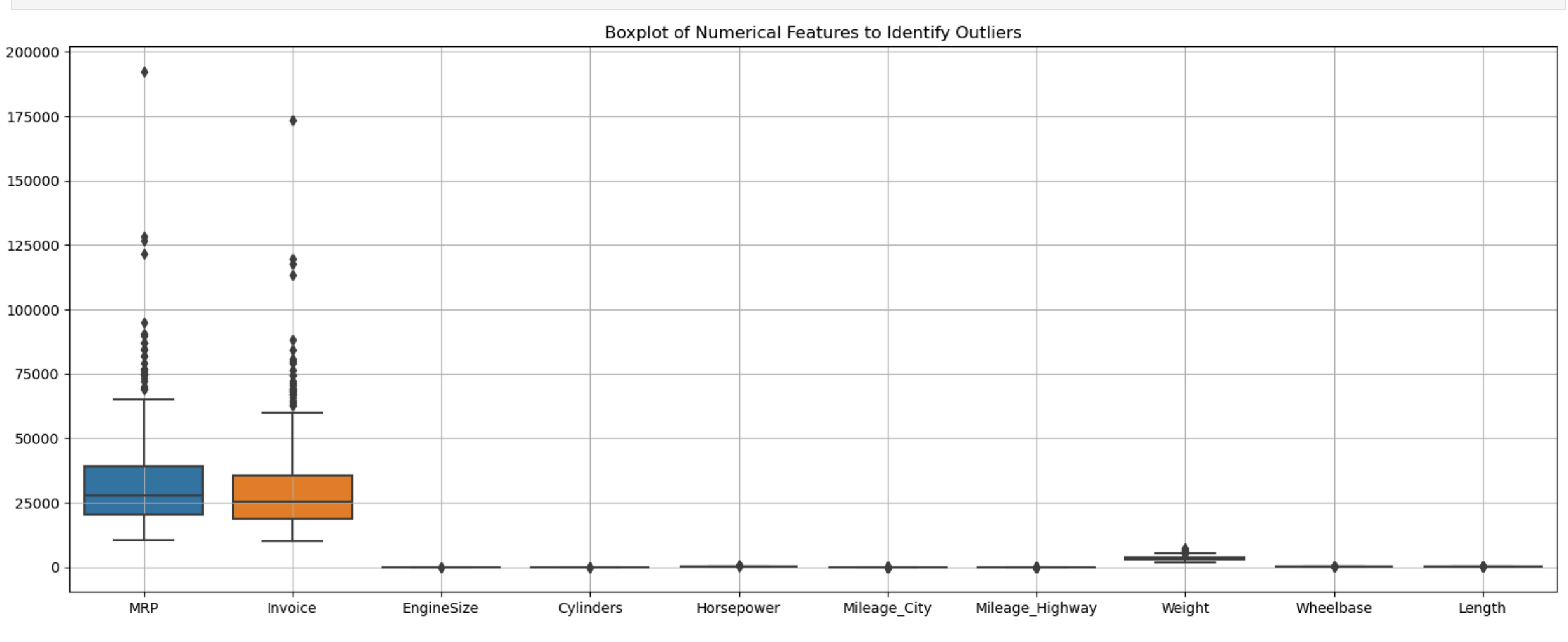

```
In [18]: plt.figure(figsize=(19, 7))
         plt.title("Frequency Distribution of Cars by Mileage in City")
         plt.hist(x=cars_df['Mileage_City'])
         plt.xlabel("Mileage in City")
         plt.ylabel("Count of Cars")
         plt.show()
```




```
In [19]: plt.figure(figsize=(19, 7))
         plt.title("Correlation Matrix of Numerical Features")
         numerical_df = cars_df.select_dtypes(include=np.number)
         sns.heatmap(numerical_df.corr(), annot=True)
         plt.show()
```




```
In [20]: plt.figure(figsize=(19, 7))
         plt.title("Boxplot of Numerical Features to Identify Outliers")
         sns.boxplot(data=numerical_df)
         plt.grid()
         plt.show()
```




```
In [21]: print("\n--- Feature Engineering ---")
         cars_cat_df = cars_df.select_dtypes(exclude=np.number)
         cars_num_df = cars_df.select_dtypes(include=np.number)
         print("Performing One-Hot Encoding on 'Type' column...")
         from sklearn.preprocessing import OneHotEncoder
         encoder = OneHotEncoder(sparse_output=False)
         encoded_type = encoder.fit_transform(cars_cat_df[['Type']])
         cars_encode_df = pd.DataFrame(encoded_type, columns=encoder.get_feature_names_out(['Type']))
         print("One-Hot Encoded DataFrame head:")
         print(cars_encode_df.head())
```

```
--- Feature Engineering ---
Performing One-Hot Encoding on 'Type' column...
One-Hot Encoded DataFrame head:
   Type_Hybrid  Type_SUV  Type_Sedan  Type_Sports  Type_Truck  Type_Wagon
0          0.0       1.0         0.0          0.0         0.0         0.0
1          0.0       0.0         1.0          0.0         0.0         0.0
2          0.0       0.0         1.0          0.0         0.0         0.0
3          0.0       0.0         1.0          0.0         0.0         0.0
4          0.0       0.0         1.0          0.0         0.0         0.0
```

```
In [22]: print("\n--- Data Scaling ---")
         print("Applying StandardScaler to numerical data...")
         from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         cars_num_scaled_df = pd.DataFrame(scaler.fit_transform(cars_num_df), columns=cars_num_df.columns)
         print("Scaled numerical data description:")
         print(cars_num_scaled_df.describe())
```

```
--- Data Scaling ---
Applying StandardScaler to numerical data...
Scaled numerical data description:
                MRP     Invoice    EngineSize     Cylinders    Horsepower  \
count  4.280000e+02  428.000000  4.280000e+02  4.280000e+02  4.280000e+02
mean  -1.577139e-16    0.000000  3.652322e-16  2.158191e-16 -7.470660e-17
std    1.001170e+00    1.001170  1.001170e+00  1.001170e+00  1.001170e+00
min   -1.158991e+00   -1.142905 -1.712933e+00 -1.808347e+00 -1.991379e+00
25%   -6.409709e-01   -0.632676 -7.421022e-01 -1.164443e+00 -7.091854e-01
50%   -2.648181e-01   -0.267866 -1.776656e-01  1.233648e-01 -8.202571e-02
75%    3.312970e-01    0.323216  6.351230e-01  1.233648e-01  5.451340e-01
max    8.227633e+00    8.146034  4.608757e+00  3.986788e+00  3.959670e+00

       Mileage_City  Mileage_Highway        Weight     Wheelbase        Length
count  4.280000e+02     4.280000e+02  4.280000e+02  4.280000e+02  4.280000e+02
mean  -2.448716e-16    -3.527811e-17  9.130806e-17 -6.474572e-16  2.573227e-16
std    1.001170e+00     1.001170e+00  1.001170e+00  1.001170e+00  1.001170e+00
min   -1.922891e+00    -2.588453e+00 -2.279333e+00 -2.307153e+00 -3.023605e+00
25%   -5.849947e-01    -4.958520e-01 -6.251890e-01 -6.208318e-01 -5.830854e-01
50%   -2.027386e-01    -1.470851e-01 -1.364646e-01 -1.390258e-01  4.447676e-02
75%    2.272995e-01     3.760652e-01  5.273695e-01  4.632317e-01  5.325807e-01
max    7.633511e+00     6.828252e+00  4.764630e+00  4.317679e+00  3.600663e+00
```

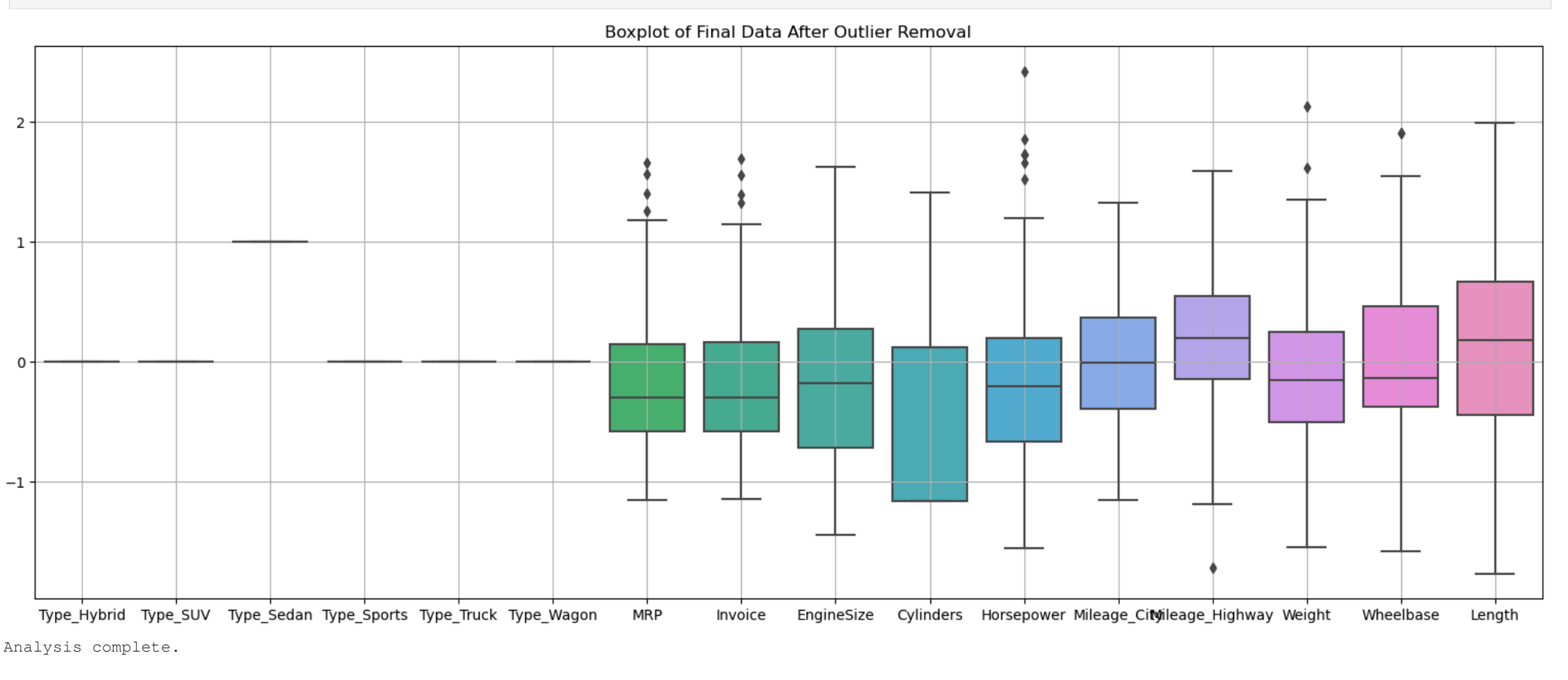
```
In [23]: cars_encode_df.reset_index(drop=True, inplace=True)
         cars_num_scaled_df.reset_index(drop=True, inplace=True)
         final_df = pd.concat([cars_encode_df, cars_num_scaled_df], axis=1)
         print("\n--- Final Processed DataFrame ---")
         print(f"Final shape: {final_df.shape}")
         print(final_df.head())
```

```
--- Final Processed DataFrame ---
Final shape: (428, 16)
   Type_Hybrid  Type_SUV  Type_Sedan  Type_Sports  Type_Truck  Type_Wagon  \
0          0.0       1.0         0.0          0.0         0.0         0.0
1          0.0       0.0         1.0          0.0         0.0         0.0
2          0.0       0.0         1.0          0.0         0.0         0.0
3          0.0       0.0         1.0          0.0         0.0         0.0
4          0.0       0.0         1.0          0.0         0.0         0.0

        MRP   Invoice  EngineSize  Cylinders  Horsepower  Mileage_City  \
0  0.214856  0.188537    0.273884   0.123365    0.684503     -0.584995
1 -0.461376 -0.468388   -1.080764  -1.164443   -0.221395      0.752902
2 -0.298050 -0.304611   -0.719525  -1.164443   -0.221395      0.370645
3  0.021647  0.016134    0.002954   0.123365    0.754187     -0.011611
4  0.565724  0.510700    0.273884   0.123365    0.127028     -0.393867

   Mileage_Highway    Weight  Wheelbase    Length
0        -0.670235  1.151631  -0.259477  0.183935
1         0.724832 -1.055214  -0.861735 -1.001460
2         0.376065 -0.458983  -0.379929 -0.234440
3         0.201682 -0.003896  -0.018574 -0.025252
4        -0.495852  0.398428   0.824586  0.741768
```

```
In [24]: print("\n--- Outlier Treatment ---")
         Q1 = final_df.quantile(0.25)
         Q3 = final_df.quantile(0.75)
         IQR = Q3 - Q1
         lower_limit = Q1 - 1.5 * IQR
         upper_limit = Q3 + 1.5 * IQR
         rows_before = final_df.shape[0]
         outliers_mask = ((final_df < lower_limit) | (final_df > upper_limit)).any(axis=1)
         cars_df_no_outliers = final_df[~outliers_mask]
         rows_after = cars_df_no_outliers.shape[0]
         print(f"\nNumber of outliers detected and removed: {rows_before - rows_after}")
         print(f"Shape of DataFrame after removing outliers: {cars_df_no_outliers.shape}")
```

```
--- Outlier Treatment ---

Number of outliers detected and removed: 203
Shape of DataFrame after removing outliers: (225, 16)
```

```
In [25]: plt.figure(figsize=(19, 7))
         sns.boxplot(data=cars_df_no_outliers)
         plt.title("Boxplot of Final Data After Outlier Removal")
         plt.grid()
         plt.show()
         print("\nAnalysis complete.")
```




```
Analysis complete.
```