In [1]:
```
import pandas as pd

# Loading the dataset
file_path = 'Used Car Dataset.csv'
data = pd.read_csv(file_path)

# Displaying basic information about the dataset
data_info = data.info()

# Displaying the first few rows of the dataset to understand its structure
data_head = data.head()

data_info, data_head
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1553 entries, 0 to 1552
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Unnamed: 0          1553 non-null   int64  
 1   car_name            1553 non-null   object 
 2   registration_year   1553 non-null   object 
 3   insurance_validity  1553 non-null   object 
 4   fuel_type           1553 non-null   object 
 5   seats               1553 non-null   int64  
 6   kms_driven          1553 non-null   int64  
 7   ownsership          1553 non-null   object 
 8   transmission        1553 non-null   object 
 9   manufacturing_year  1553 non-null   object 
 10  mileage(kmpl)       1550 non-null   float64
 11  engine(cc)          1550 non-null   float64
 12  max_power(bhp)      1550 non-null   float64
 13  torque(Nm)          1549 non-null   float64
 14  price(in lakhs)     1553 non-null   float64
dtypes: float64(5), int64(3), object(7)
memory usage: 182.1+ KB
```

Out[1]:

```
(None,
    Unnamed: 0                                        car_name  \
 0           0                     2017 Mercedes-Benz S-Class S400   
 1           1  2020 Nissan Magnite Turbo CVT XV Premium Opt BSVI   
 2           2                          2018 BMW X1 sDrive 20d xLine   
 3           3                              2019 Kia Seltos GTX Plus   
 4           4                     2019 Skoda Superb LK 1.8 TSI AT   

   registration_year insurance_validity fuel_type  seats  kms_driven  \
 0            Jul-17      Comprehensive    Petrol      5       56000   
 1            Jan-21      Comprehensive    Petrol      5       30615   
 2            Sep-18      Comprehensive    Diesel      5       24000   
 3            Dec-19      Comprehensive    Petrol      5       18378   
 4            Aug-19      Comprehensive    Petrol      5       44900   

     ownsership transmission manufacturing_year  mileage(kmpl)  engine(cc)  \
 0  First Owner    Automatic               2017           7.81      2996.0   
 1  First Owner    Automatic               2020          17.40       999.0   
 2  First Owner    Automatic               2018          20.68      1995.0   
 3  First Owner       Manual               2019          16.50      1353.0   
 4  First Owner    Automatic               2019          14.67      1798.0   

    max_power(bhp)  torque(Nm)  price(in lakhs)  
 0          2996.0       333.0            63.75  
 1           999.0      9863.0             8.99  
 2          1995.0       188.0            23.75  
 3          1353.0     13808.0            13.56  
 4          1798.0     17746.0            24.00  )
```

In [2]:
```
# Checking for missing values
missing_values = data.isnull().sum()
```

```
# Checking for duplicate entries
duplicate_entries = data.duplicated().sum()

# Summary of numerical columns for outlier detection
numerical_summary = data.describe()

missing_values, duplicate_entries, numerical_summary
```

Out[2]:

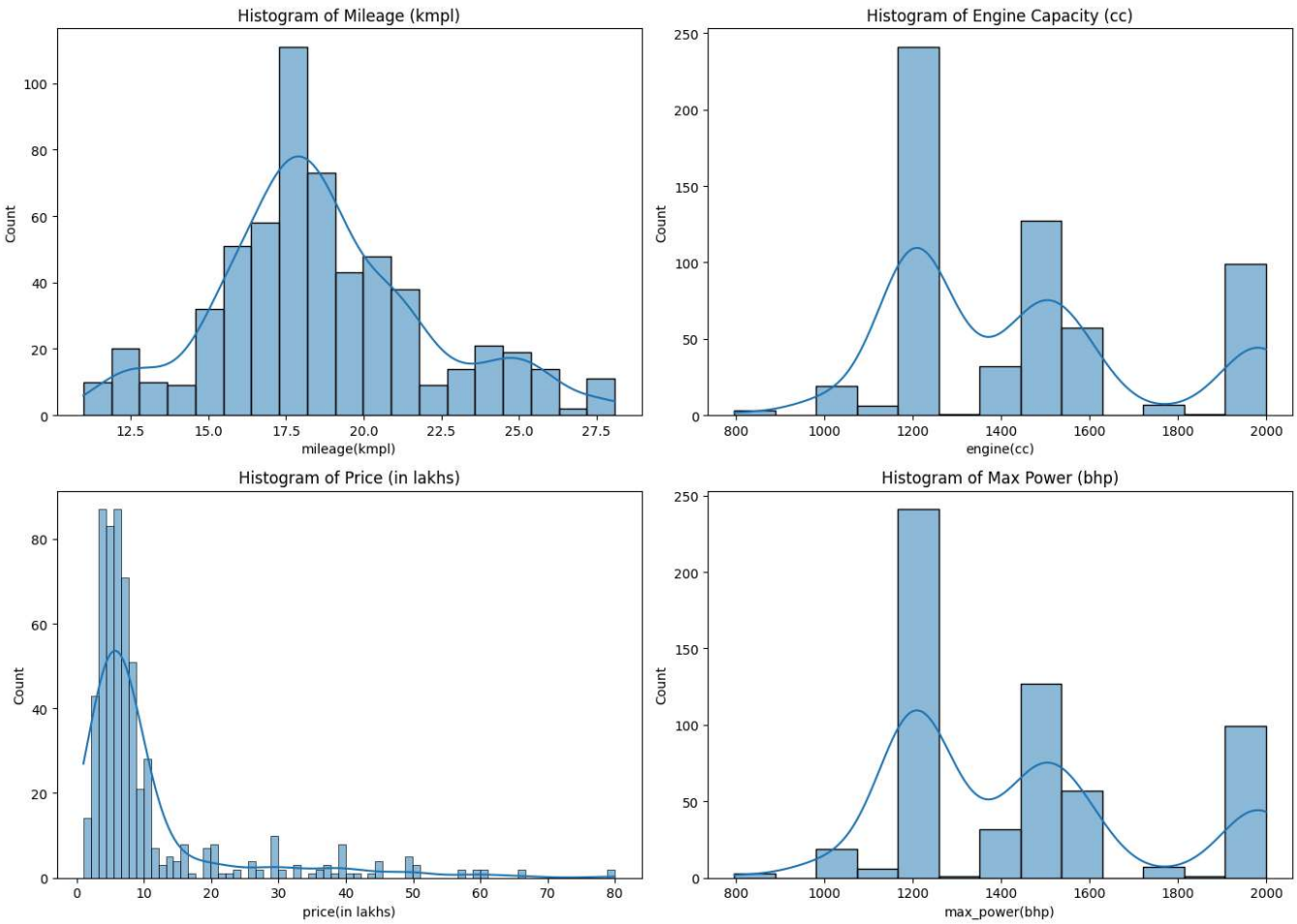
```
(Unnamed: 0            0
 car_name              0
 registration_year     0
 insurance_validity    0
 fuel_type             0
 seats                 0
 kms_driven            0
 ownsership            0
 transmission          0
 manufacturing_year    0
 mileage(kmpl)         3
 engine(cc)            3
 max_power(bhp)        3
 torque(Nm)            4
 price(in lakhs)       0
 dtype: int64,
 0,
         Unnamed: 0         seats     kms_driven  mileage(kmpl)    engine(cc)  \
 count  1553.000000   1553.000000    1553.000000    1550.000000  1.550000e+03
 mean    776.000000     91.480361   52841.931101     236.927277  1.471857e+10
 std     448.456798   2403.424060   40067.800347     585.964295  2.185629e+11
 min       0.000000      4.000000     620.000000       7.810000  5.000000e+00
 25%     388.000000      5.000000   30000.000000      16.342500  1.197000e+03
 50%     776.000000      5.000000   49134.000000      18.900000  1.462000e+03
 75%    1164.000000      5.000000   70000.000000      22.000000  1.995000e+03
 max    1552.000000  67000.000000  810000.000000    3996.000000  3.258640e+12

        max_power(bhp)    torque(Nm)  price(in lakhs)
 count    1.550000e+03  1.549000e+03      1553.000000
 mean     1.471857e+10  1.423989e+04       166.141494
 std      2.185629e+11  9.666241e+04      3478.855090
 min      5.000000e+00  5.000000e+00         1.000000
 25%      1.197000e+03  4.000000e+02         4.660000
 50%      1.462000e+03  1.173000e+03         7.140000
 75%      1.995000e+03  8.850000e+03        17.000000
 max      3.258640e+12  1.464800e+06     95000.000000  )
```

In [3]:

```
# Removing rows with missing values
cleaned_data = data.dropna()

# Removing rows with outliers in the 'seats' column (assuming a car can have a maximum of 10
cleaned_data = cleaned_data[cleaned_data['seats'] <= 10]

# Removing rows with outliers in 'engine(cc)', 'max_power(bhp)', and 'torque(Nm)' columns
# Assuming reasonable thresholds for these columns (these thresholds can be adjusted as neede
cleaned_data = cleaned_data[(cleaned_data['engine(cc)'] <= 10000) &
                            (cleaned_data['max_power(bhp)'] <= 2000) &
                            (cleaned_data['torque(Nm)'] <= 2000)]

# Summary of the cleaned data
cleaned_data_summary = cleaned_data.describe()

# Number of rows removed
rows_removed = data.shape[0] - cleaned_data.shape[0]

cleaned_data_summary, rows_removed
```

Out[3]:
```
(       Unnamed: 0       seats     kms_driven  mileage(kmpl)   engine(cc)  \
 count   703.000000  703.000000     703.000000     703.000000   703.000000
 mean    771.661451    5.068279   57174.504979     235.783158  1274.859175
 std     444.258545    0.389906   27993.657300     546.803374   488.649204
 min       2.000000    4.000000    1800.000000      10.980000    67.000000
 25%     382.000000    5.000000   37163.000000      17.005000  1197.000000
 50%     763.000000    5.000000   54000.000000      18.900000  1248.000000
 75%    1155.500000    5.000000   72503.000000      23.525000  1498.000000
 max    1551.000000    8.000000  150000.000000    3996.000000  1999.000000

        max_power(bhp)   torque(Nm)  price(in lakhs)
 count      703.000000   703.000000       703.000000
 mean      1274.859175   683.597440       344.472077
 std        488.649204   461.633574      5166.976583
 min         67.000000    17.000000         1.000000
 25%       1197.000000   184.000000         4.240000
 50%       1248.000000   831.000000         6.250000
 75%       1498.000000   996.000000         9.950000
 max       1999.000000  1944.000000     95000.000000  ,
 850)
```

In [4]:
```python
# Further clean the dataset by removing extreme outliers in 'mileage(kmpl)' and 'price(in lak
cleaned_data_further = cleaned_data[(cleaned_data['mileage(kmpl)'] <= 50) & (cleaned_data['pr

# Summary of the further cleaned data
cleaned_further_summary = cleaned_data_further.describe()

# Number of rows now removed
rows_now_removed = cleaned_data.shape[0] - cleaned_data_further.shape[0]

import matplotlib.pyplot as plt
import seaborn as sns

# Plotting histograms for key columns to visualize distributions
fig, axs = plt.subplots(2, 2, figsize=(14, 10))

sns.histplot(cleaned_data_further['mileage(kmpl)'], kde=True, ax=axs[0, 0])
axs[0, 0].set_title('Histogram of Mileage (kmpl)')

sns.histplot(cleaned_data_further['engine(cc)'], kde=True, ax=axs[0, 1])
axs[0, 1].set_title('Histogram of Engine Capacity (cc)')

sns.histplot(cleaned_data_further['price(in lakhs)'], kde=True, ax=axs[1, 0])
axs[1, 0].set_title('Histogram of Price (in lakhs)')

sns.histplot(cleaned_data_further['max_power(bhp)'], kde=True, ax=axs[1, 1])
axs[1, 1].set_title('Histogram of Max Power (bhp)')

plt.tight_layout()
plt.show()

cleaned_further_summary, rows_now_removed
```

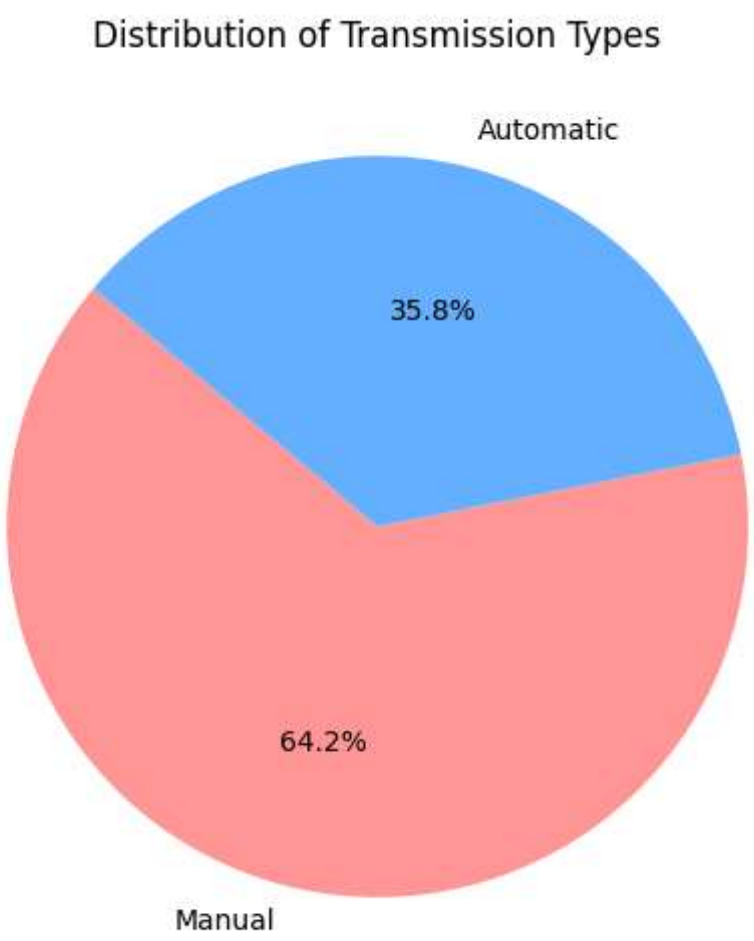


Out[4]:

```
(        Unnamed: 0        seats      kms_driven  mileage(kmpl)    engine(cc)  \
 count   593.000000   593.000000      593.000000     593.000000   593.000000
 mean    772.728499     5.064081    58785.291737      18.690995  1441.839798
 std     439.844055     0.384596    28393.313450       3.407763   292.668852
 min       2.000000     4.000000     1800.000000      10.980000   796.000000
 25%     405.000000     5.000000    40000.000000      16.800000  1198.000000
 50%     763.000000     5.000000    55861.000000      18.000000  1373.000000
 75%    1145.000000     5.000000    73937.000000      20.680000  1591.000000
 max    1551.000000     7.000000   150000.000000      28.090000  1999.000000

        max_power(bhp)   torque(Nm)  price(in lakhs)
 count      593.000000   593.000000       593.000000
 mean      1441.839798   757.527825        10.346762
 std        292.668852   453.608028        11.952074
 min        796.000000    67.000000         1.000000
 25%       1198.000000   190.000000         4.500000
 50%       1373.000000   868.000000         6.340000
 75%       1591.000000  1085.000000         9.220000
 max       1999.000000  1944.000000        80.000000  ,
 110)
```

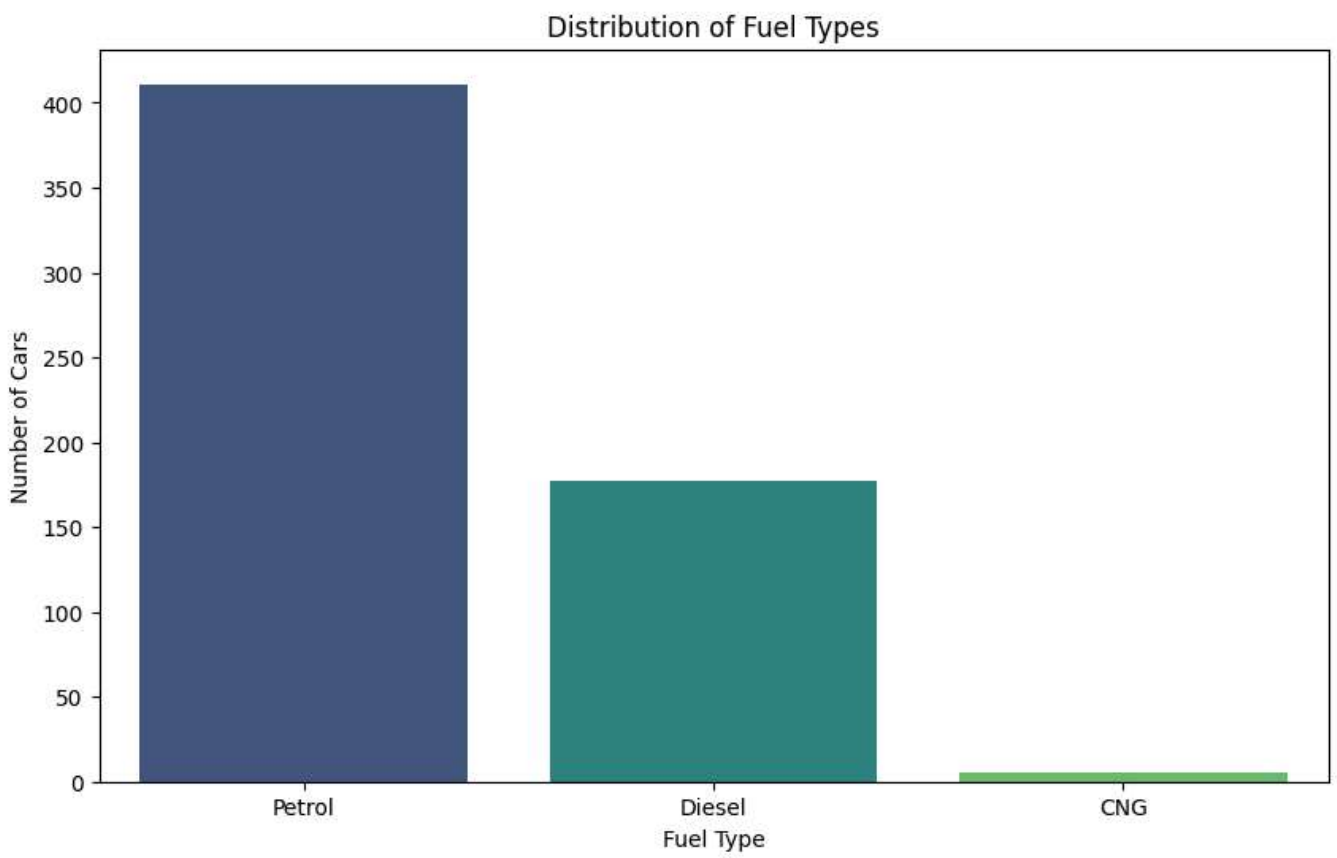
In [5]:

```
# Pie chart for transmission types
transmission_counts = cleaned_data_further['transmission'].value_counts()
plt.figure(figsize=(8, 6))
plt.pie(transmission_counts, labels=transmission_counts.index, autopct='%1.1f%%', startangle=
plt.title('Distribution of Transmission Types')
plt.show()
```



In [6]:
```
# Bar graph for fuel types
fuel_type_counts = cleaned_data_further['fuel_type'].value_counts()
plt.figure(figsize=(10, 6))
sns.barplot(x=fuel_type_counts.index, y=fuel_type_counts.values, palette='viridis')
plt.title('Distribution of Fuel Types')
plt.xlabel('Fuel Type')
plt.ylabel('Number of Cars')
plt.show()
```

```
C:\Users\Bunty\AppData\Local\Temp\ipykernel_7560\385298104.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assig
n the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=fuel_type_counts.index, y=fuel_type_counts.values, palette='viridis')
```



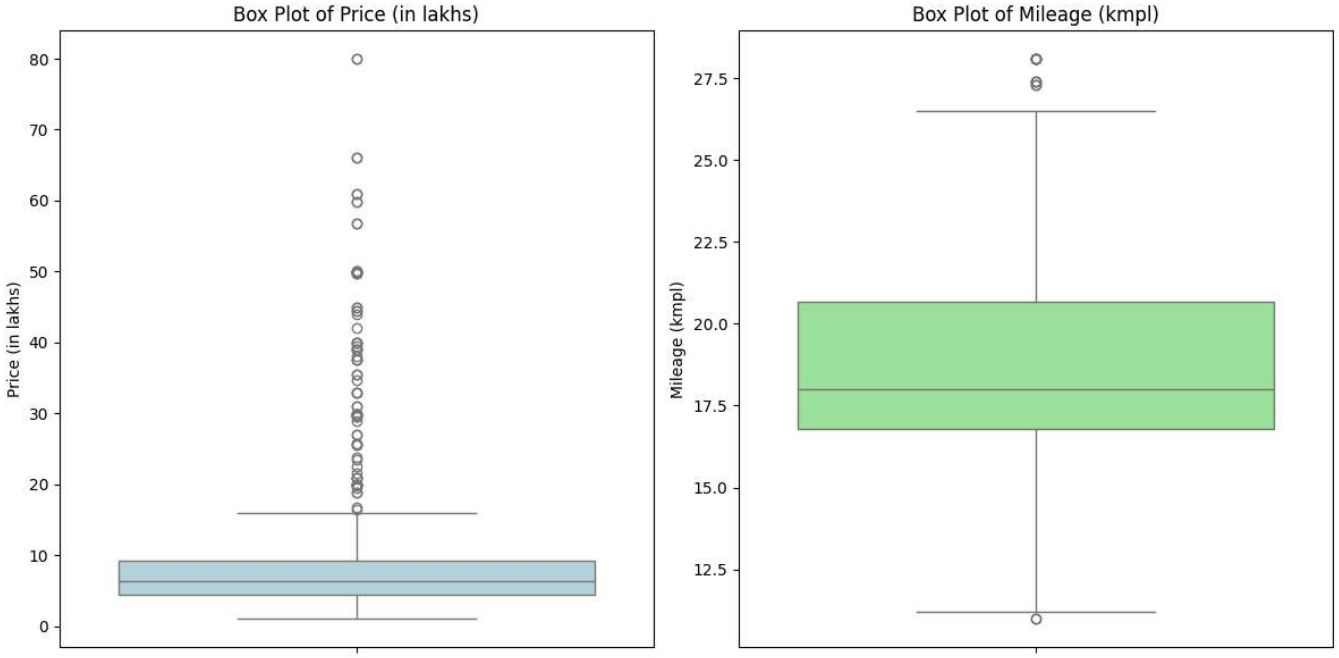
In [7]:

```
import numpy as np

# Measures of Center
mean_price = cleaned_data_further['price(in lakhs)'].mean()
median_price = cleaned_data_further['price(in lakhs)'].median()
mode_price = cleaned_data_further['price(in lakhs)'].mode()[0]

mean_mileage = cleaned_data_further['mileage(kmpl)'].mean()
median_mileage = cleaned_data_further['mileage(kmpl)'].median()
mode_mileage = cleaned_data_further['mileage(kmpl)'].mode()[0]

# Measures of Dispersion
range_price = cleaned_data_further['price(in lakhs)'].max() - cleaned_data_further['price(in
variance_price = cleaned_data_further['price(in lakhs)'].var()
std_dev_price = cleaned_data_further['price(in lakhs)'].std()

range_mileage = cleaned_data_further['mileage(kmpl)'].max() - cleaned_data_further['mileage(k
variance_mileage = cleaned_data_further['mileage(kmpl)'].var()
std_dev_mileage = cleaned_data_further['mileage(kmpl)'].std()

# Measures of Location (Percentiles and 5-number summary)
percentiles_price = np.percentile(cleaned_data_further['price(in lakhs)'], [25, 50, 75])
five_num_summary_price = np.percentile(cleaned_data_further['price(in lakhs)'], [0, 25, 50, 7

percentiles_mileage = np.percentile(cleaned_data_further['mileage(kmpl)'], [25, 50, 75])
five_num_summary_mileage = np.percentile(cleaned_data_further['mileage(kmpl)'], [0, 25, 50, 7

# Output the results
summary_stats = {
    "Price in Lakhs": {
        "Mean": mean_price,
        "Median": median_price,
        "Mode": mode_price,
        "Range": range_price,
        "Variance": variance_price,
        "Standard Deviation": std_dev_price,
        "25th, 50th, 75th Percentiles": percentiles_price,
        "5-number Summary": five_num_summary_price
    },
    "Mileage (kmpl)": {
        "Mean": mean_mileage,
```

```
        "Median": median_mileage,
        "Mode": mode_mileage,
        "Range": range_mileage,
        "Variance": variance_mileage,
        "Standard Deviation": std_dev_mileage,
        "25th, 50th, 75th Percentiles": percentiles_mileage,
        "5-number Summary": five_num_summary_mileage
    }
}

summary_stats
```

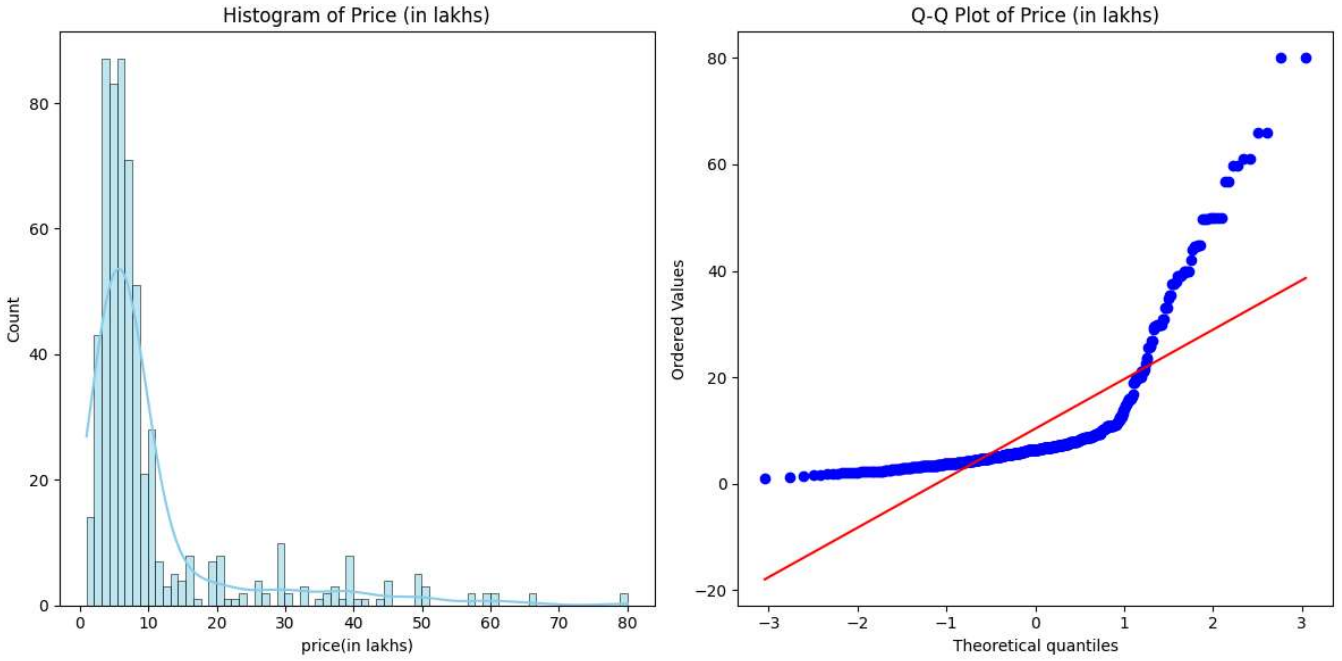
Out[7]:
```
{'Price in Lakhs': {'Mean': 10.346762225969647,
  'Median': 6.34,
  'Mode': 6.25,
  'Range': 79.0,
  'Variance': 142.8520820666788,
  'Standard Deviation': 11.952074383414738,
  '25th, 50th, 75th Percentiles': array([4.5 , 6.34, 9.22]),
  '5-number Summary': array([ 1.  ,  4.5 ,  6.34,  9.22, 80.  ])},
 'Mileage (kmpl)': {'Mean': 18.69099494097808,
  'Median': 18.0,
  'Mode': 21.4,
  'Range': 17.11,
  'Variance': 11.612851542204094,
  'Standard Deviation': 3.4077634222762727,
  '25th, 50th, 75th Percentiles': array([16.8 , 18.  , 20.68]),
  '5-number Summary': array([10.98, 16.8 , 18.  , 20.68, 28.09])}}
```

In [8]:
```
# Creating box plots for Price and Mileage
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.boxplot(y=cleaned_data_further['price(in lakhs)'], color='lightblue')
plt.title('Box Plot of Price (in lakhs)')
plt.ylabel('Price (in lakhs)')

plt.subplot(1, 2, 2)
sns.boxplot(y=cleaned_data_further['mileage(kmpl)'], color='lightgreen')
plt.title('Box Plot of Mileage (kmpl)')
plt.ylabel('Mileage (kmpl)')

plt.tight_layout()
plt.show()
```



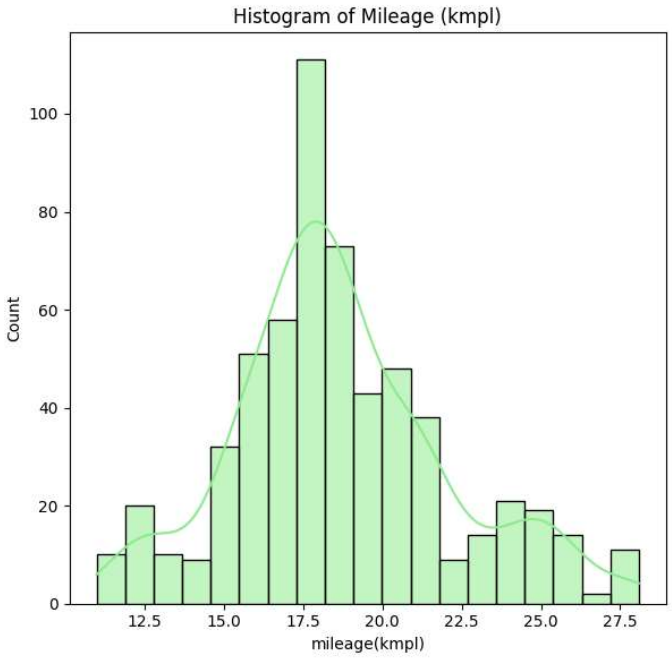
In [9]:
```
import scipy.stats as stats
```

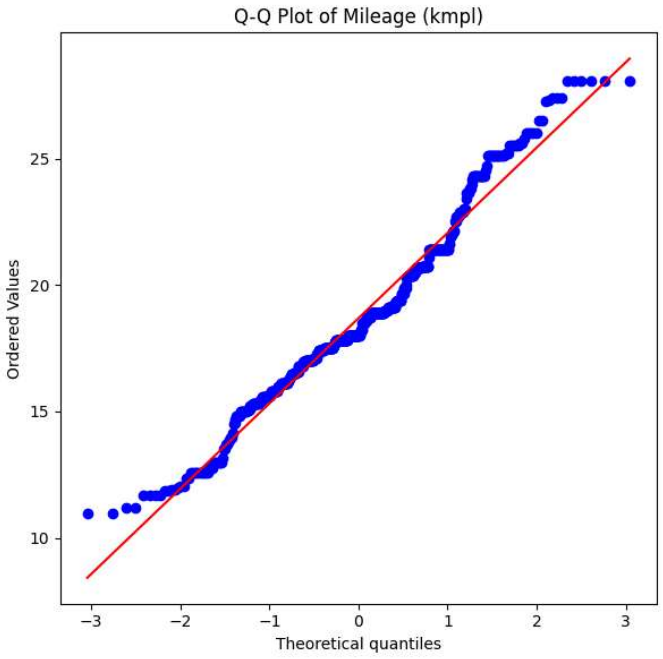
```
# Histogram and Q-Q plot for 'price(in lakhs)'
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.histplot(cleaned_data_further['price(in lakhs)'], kde=True, color='skyblue')
plt.title('Histogram of Price (in lakhs)')

plt.subplot(1, 2, 2)
stats.probplot(cleaned_data_further['price(in lakhs)'], dist="norm", plot=plt)
plt.title('Q-Q Plot of Price (in lakhs)')

plt.tight_layout()
plt.show()
```



In [11]:

```
# Histogram and Q-Q plot for 'mileage(kmpl)'
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.histplot(cleaned_data_further['mileage(kmpl)'], kde=True, color='lightgreen')
plt.title('Histogram of Mileage (kmpl)')

plt.subplot(1, 2, 2)
stats.probplot(cleaned_data_further['mileage(kmpl)'], dist="norm", plot=plt)
plt.title('Q-Q Plot of Mileage (kmpl)')

plt.tight_layout()
plt.show()
```





In [12]:
```
# One-sample t-test for 'price(in lakhs)'
mu = 5  # Hypothesized mean
t_stat, p_val = stats.ttest_1samp(cleaned_data_further['price(in lakhs)'], mu)

# Since it's a one-tailed test, we need to halve the p-value
p_val /= 2

t_stat, p_val
```

Out[12]:
```
(10.893687945321702, 1.2963227082855062e-25)
```

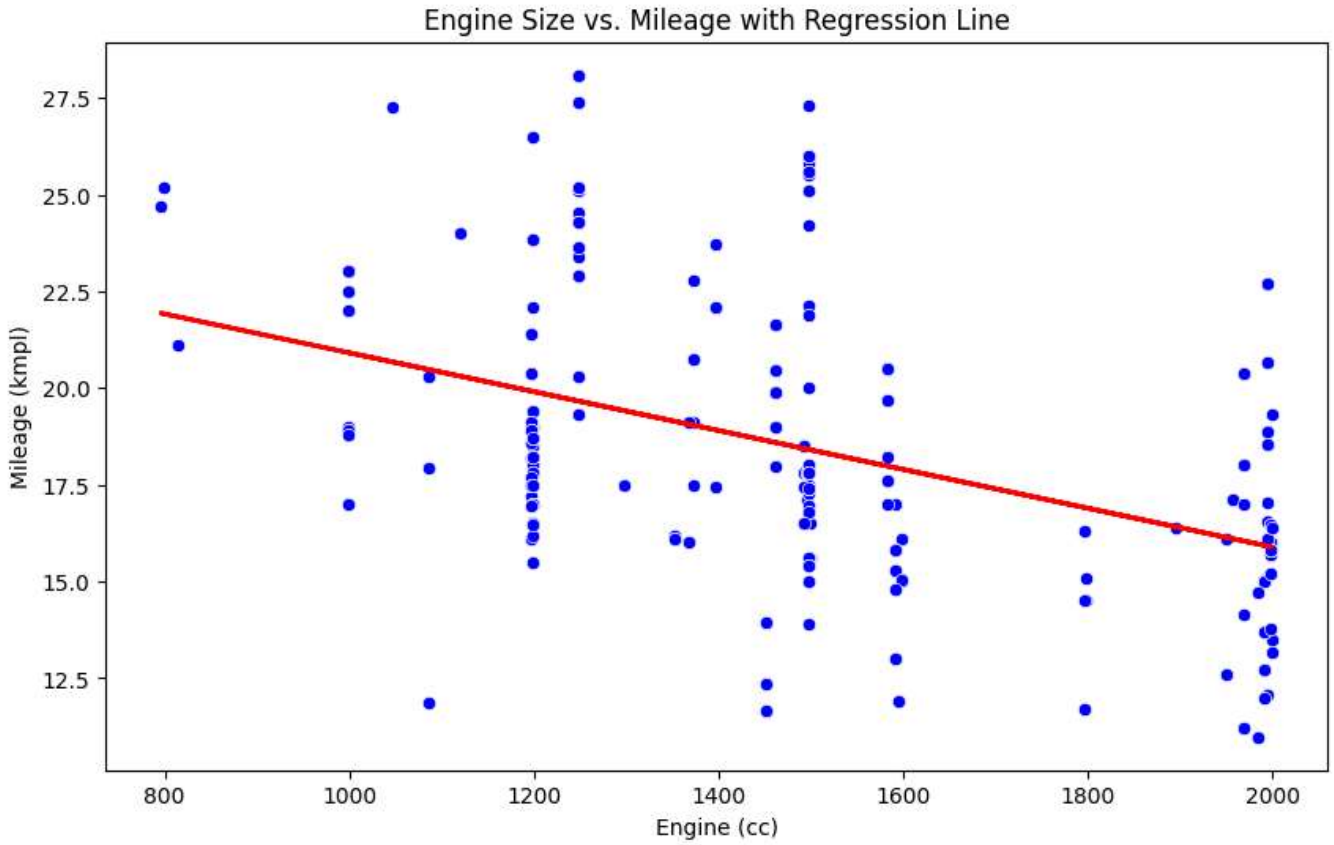
In [13]:
```
# Calculating the correlation coefficient between 'engine(cc)' and 'mileage(kmpl)'
correlation_coefficient = cleaned_data_further[['engine(cc)', 'mileage(kmpl)']].corr().iloc[0
correlation_coefficient
```

Out[13]:
```
-0.43138686355922556
```

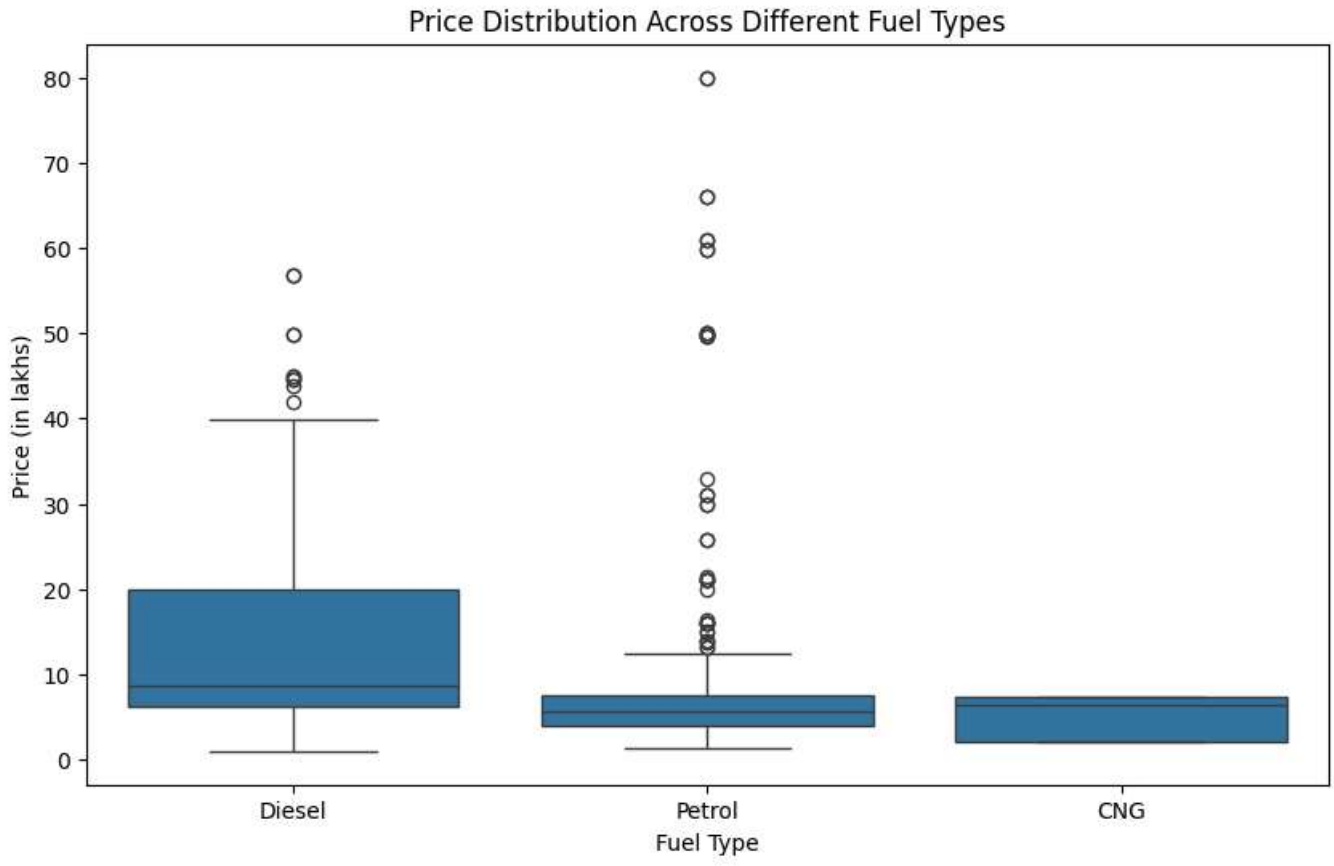
In [14]:
```
from sklearn.linear_model import LinearRegression

# Setting up the linear regression model
X = cleaned_data_further[['engine(cc)']]  # Independent variable
y = cleaned_data_further['mileage(kmpl)']  # Dependent variable

# Fitting the linear regression model
model = LinearRegression()
model.fit(X, y)

# Coefficients
slope = model.coef_[0]
intercept = model.intercept_

slope, intercept
```

Out[14]:
```
(-0.0050229614915167165, 25.93330072145565)
```

In [15]:
```
# Scatter plot with the regression line
plt.figure(figsize=(10, 6))
sns.scatterplot(x=cleaned_data_further['engine(cc)'], y=cleaned_data_further['mileage(kmpl)']
plt.plot(X, model.predict(X), color='red', linewidth=2)  # Regression line
plt.title('Engine Size vs. Mileage with Regression Line')
plt.xlabel('Engine (cc)')
plt.ylabel('Mileage (kmpl)')
plt.show()
```

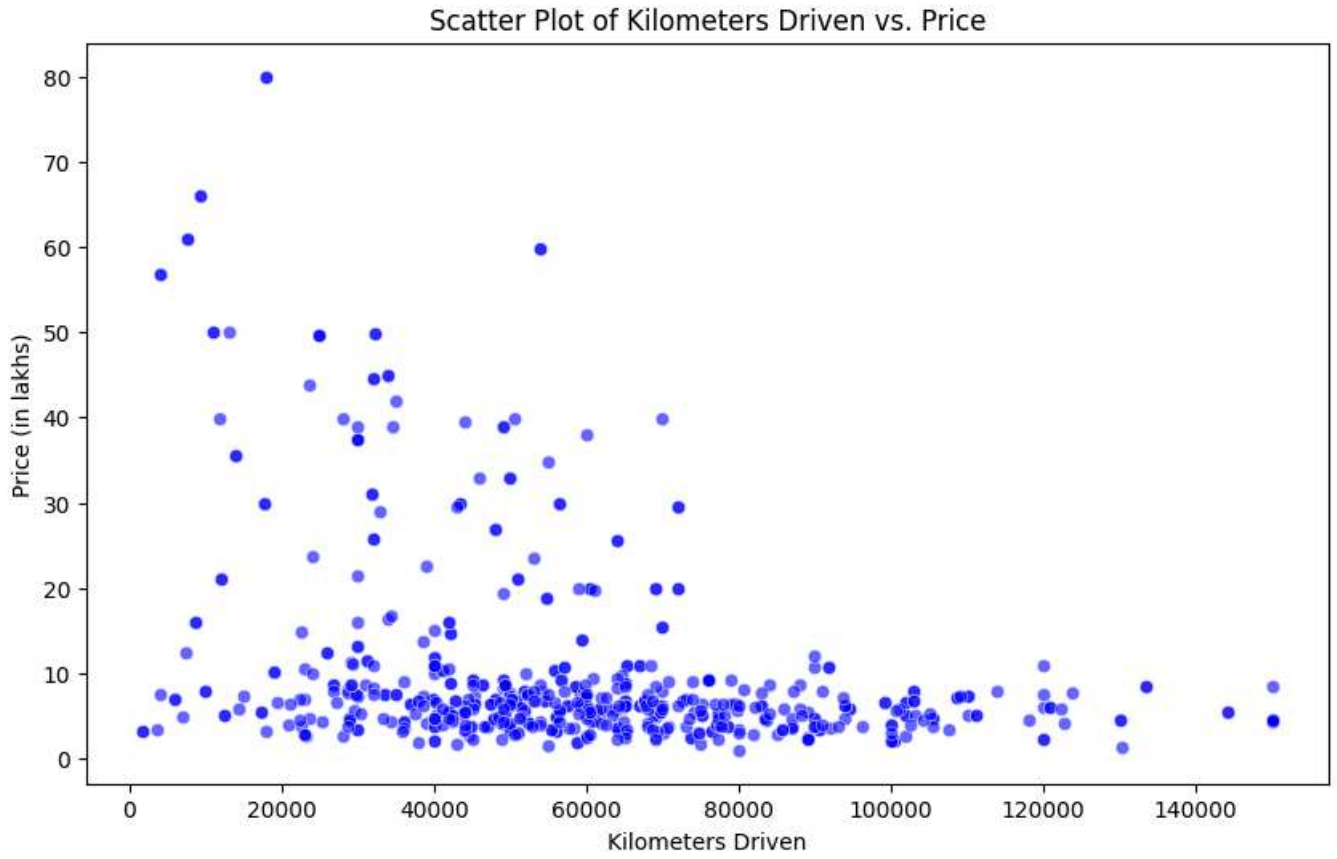


In [16]:
```
# Box plot for price distributions across different fuel types
plt.figure(figsize=(10, 6))
sns.boxplot(x='fuel_type', y='price(in lakhs)', data=cleaned_data_further)
plt.title('Price Distribution Across Different Fuel Types')
plt.xlabel('Fuel Type')
plt.ylabel('Price (in lakhs)')
plt.show()
```



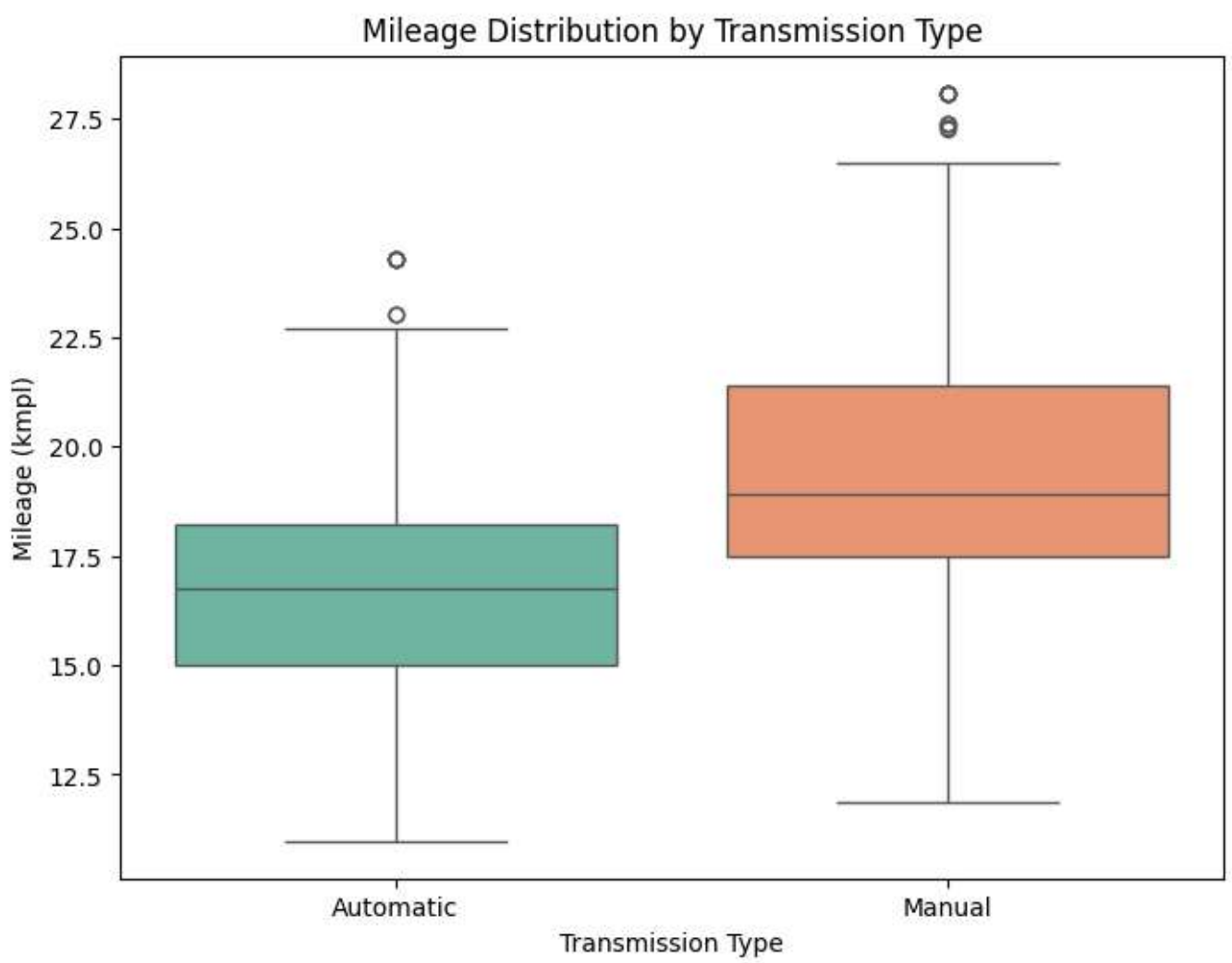
In [20]:
```
# Scatter plot for 'kms_driven' vs 'price_in_lakhs'
plt.figure(figsize=(10, 6))
sns.scatterplot(x='kms_driven', y='price_in_lakhs', data=cleaned_data_further_renamed, color=
plt.title('Scatter Plot of Kilometers Driven vs. Price')
plt.xlabel('Kilometers Driven')
```

```
plt.ylabel('Price (in lakhs)')
plt.show()
```



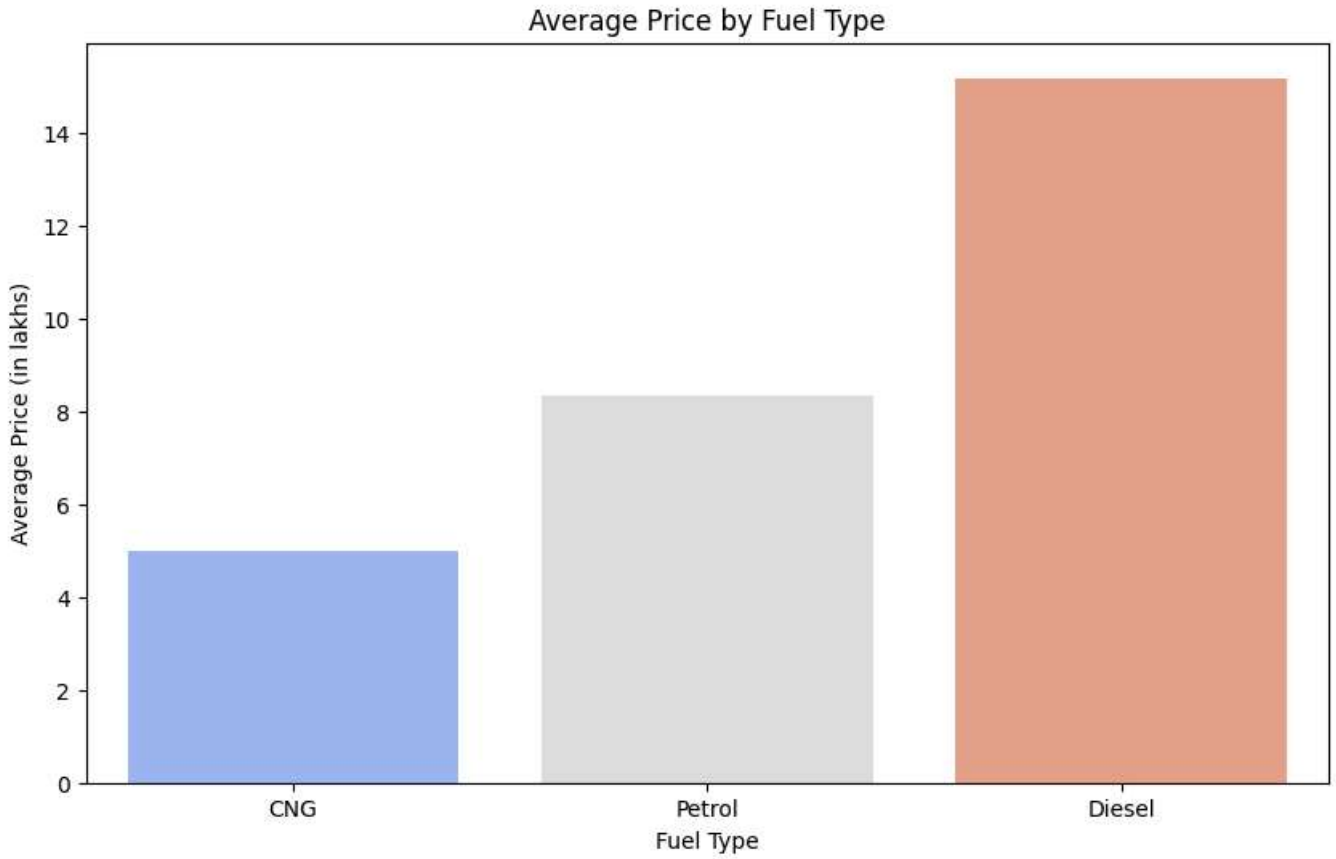
In [23]:
```
# Box plot for mileage distributions by transmission type
plt.figure(figsize=(8, 6))
sns.boxplot(x='transmission', y='mileage(kmpl)', data=cleaned_data_further_renamed, palette='
plt.title('Mileage Distribution by Transmission Type')
plt.xlabel('Transmission Type')
plt.ylabel('Mileage (kmpl)')
plt.show()
```

```
C:\Users\Bunty\AppData\Local\Temp\ipykernel_7560\3440263294.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assig
n the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='transmission', y='mileage(kmpl)', data=cleaned_data_further_renamed, palette
='Set2')
```



In [24]:
```
# Calculating average price for each fuel type
average_price_by_fuel_type = cleaned_data_further_renamed.groupby('fuel_type')['price_in_lakh

# Bar graph for average prices by fuel type
plt.figure(figsize=(10, 6))
sns.barplot(x=average_price_by_fuel_type.index, y=average_price_by_fuel_type.values, palette=
plt.title('Average Price by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Average Price (in lakhs)')
plt.show()
```

```
C:\Users\Bunty\AppData\Local\Temp\ipykernel_7560\3887504638.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assig
n the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=average_price_by_fuel_type.index, y=average_price_by_fuel_type.values, palett
e='coolwarm')
```



In [17]:
```
import statsmodels.api as sm
from statsmodels.formula.api import ols

cleaned_data_further_renamed = cleaned_data_further.rename(columns={'price(in lakhs)': 'price

# ANOVA test to see if there are significant price differences across fuel types
model = ols('price_in_lakhs ~ C(fuel_type)', data=cleaned_data_further_renamed).fit()
anova_results = sm.stats.anova_lm(model, typ=2)

anova_results
```

Out[17]:

	sum_sq	df	F	PR(>F)
C(fuel_type)	5945.383867	2.0	22.307558	4.579011e-10
Residual	78623.048716	590.0	NaN	NaN

In [18]:
```
# Extracting mileage data for manual and automatic transmission types
mileage_manual = cleaned_data_further_renamed[cleaned_data_further_renamed['transmission'] ==
mileage_automatic = cleaned_data_further_renamed[cleaned_data_further_renamed['transmission']

# Performing t-test between the two groups
t_stat, p_val = stats.ttest_ind(mileage_manual, mileage_automatic, equal_var=False)  # Welch'

t_stat, p_val
```

Out[18]:
```
(11.660754431320399, 1.0979216700538388e-27)
```

In [19]:
```
# Calculating the correlation coefficient between 'kms_driven' and 'price_in_lakhs'
correlation_kms_price = cleaned_data_further_renamed[['kms_driven', 'price_in_lakhs']].corr()
correlation_kms_price
```

Out[19]:
```
-0.35811355492047314
```

In [ ]: