In [31]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [42]:

```
df = pd.read_csv("acko_cars.csv")
```

In [43]:

```
df
```

Out[43]:

	Unnamed: 0	Brand	Model	Body_Type	variant	No.of Seats	Price	Fuel_Type	location	gearbo
0	0	Hyundai	Creta	SUV	1.5 E	5 Seater	₹13.01 L	Petrol	Delhi	
1	1	Hyundai	Venue	SUV	1.2 E	5 Seater	₹9.06 L	Petrol	Delhi	
2	2	Kia	Sonet	SUV	1.2 HTE(O)	5 Seater	₹9.50 L	Petrol	Delhi	
3	3	Kia	Syros	SUV	1.0 HTK Turbo	5 Seater	₹10.60 L	Petrol	Delhi	
4	4	Maruti Suzuki	Suzuki Baleno (2022-2025)	Hatchback	1.2 Sigma	5 Seater	₹7.58 L	Petrol	Delhi	
...	...	...	...	...	...	...	...	...	...	
291	291	Kia	EV6 (2022-2025)	SUV	GT Line AWD	5 Seater	₹68.05 L	Electric	Delhi	Au
292	292	BYD	e6 (2022-2024)	SUV	GL	5 Seater	₹30.31 L	Electric	Delhi	Au
293	293	MINI	Countryman	SUV	Cooper S JCW Inspired	5 Seater	₹54.73 L	Petrol	Delhi	Au
294	294	Audi	RS Q8 (2020-2025)	SUV	4.0L TFSI	5 Seater	₹2.51 Cr	Petrol	Delhi	Au
295	295	Land Rover	Rover Range Rover Evoque (2024-2025)	SUV	SE R-Dynamic Diesel	5 Seater	₹78.73 L	Diesel	Delhi	Au

296 rows × 10 columns

Data Exploration

Previewing Data:

Using head() and tail() functions

In [3]:

```
df.head()
```

Out[3]:

	Unnamed: 0	Brand	Model	Body_Type	variant	No.of Seats	Price	Fuel_Type	location	gearbox_Type
0	0	Hyundai	Creta	SUV	1.5 E	5 Seater	₹13.01 L	Petrol	Delhi	Manual
1	1	Hyundai	Venue	SUV	1.2 E	5 Seater	₹9.06 L	Petrol	Delhi	Manual
2	2	Kia	Sonet	SUV	1.2 HTE(O)	5 Seater	₹9.50 L	Petrol	Delhi	Manual
3	3	Kia	Syros	SUV	1.0 HTK Turbo	5 Seater	₹10.60 L	Petrol	Delhi	Manual
4	4	Maruti Suzuki	Suzuki Baleno (2022-2025)	Hatchback	1.2 Sigma	5 Seater	₹7.58 L	Petrol	Delhi	Manual

In [4]:

```
df.tail()
```

Out[4]:

	Unnamed: 0	Brand	Model	Body_Type	variant	No.of Seats	Price	Fuel_Type	location	gearbox
291	291	Kia	EV6 (2022-2025)	SUV	GT Line AWD	5 Seater	₹68.05 L	Electric	Delhi	Aut
292	292	BYD	e6 (2022-2024)	SUV	GL	5 Seater	₹30.31 L	Electric	Delhi	Aut
293	293	MINI	Countryman	SUV	Cooper S JCW Inspired	5 Seater	₹54.73 L	Petrol	Delhi	Aut
294	294	Audi	RS Q8 (2020-2025)	SUV	4.0L TFSI	5 Seater	₹2.51 Cr	Petrol	Delhi	Aut
295	295	Land Rover	Rover Range Rover Evoque (2024-2025)	SUV	SE R-Dynamic Diesel	5 Seater	₹78.73 L	Diesel	Delhi	Aut

Structure of the data

Using shape and ndim attributes

In [5]:

```
print("The shape of ----->" ,df.shape)
print("The No of Rows :" , df.shape[0])
print("The No of Columns :",df.shape[1])
```

```
The shape of -----> (296, 10)
The No of Rows : 296
```

```
The No of Columns : 10
```

In [6]:

```
print("The Dimension of the data Frame :",df.ndim)
```

```
The Dimension of the data Frame : 2
```

Check the random Samples of the Data

In [46]:

```
df.sample()
```

Out[46]:

	Unnamed: 0	Brand	Model	Body_Type	variant	No.of Seats	Price	Fuel_Type	location	gearbox_Type
123	123	BMW	i5	Sedan	M60 xDrive	5 Seater	₹1.25 Cr	Electric	Delhi	Automatic

Concise Summary:

info() : This is one of the most useful functions for a quick look at the data types, non-null values, and memory usage.

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 296 entries, 0 to 295
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Unnamed: 0    296 non-null    int64 
 1   Brand         296 non-null    object
 2   Model         296 non-null    object
 3   Body_Type     296 non-null    object
 4   variant       296 non-null    object
 5   No.of Seats   296 non-null    object
 6   Price         296 non-null    object
 7   Fuel_Type     296 non-null    object
 8   location      296 non-null    object
 9   gearbox_Type  296 non-null    object
dtypes: int64(1), object(9)
memory usage: 23.3+ KB
```

Get basic descriptive statistics for numerical columns:

This provides measures of central tendency, dispersion, and shape of the distribution.

In [8]:

```
df.describe(include = 'all')
```

Out[8]:

	Unnamed: 0	Brand	Model	Body_Type	variant	No.of Seats	Price	Fuel_Type	location	gearbox
count	296.000000	296	296	296	296	296	296	296	296	

	Unnamed: 0	Brand	Model	Body_Type	variant	No.of Seats	Price	Fuel_Type	location	gearbox
unique	NaN	35	295	8	254	7	278	4	1	
top	NaN	Mercedes-Benz	EQS	SUV	Coupe	5 Seater	₹1.70 Cr	Petrol	Delhi	Aut
freq	NaN	31	2	176	5	212	3	187	296	
mean	147.500000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	85.592056	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	73.750000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	147.500000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	221.250000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	295.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

To find the name of all columns

In [9]:
```
df.columns
```

Out[9]:
```
Index(['Unnamed: 0', 'Brand', 'Model', 'Body_Type', 'variant', 'No.of Seats',
       'Price', 'Fuel_Type', 'location', 'gearbox_Type'],
      dtype='object')
```

To find the datatypes of columns

In [10]:
```
df.dtypes
```

Out[10]:
```
Unnamed: 0       int64
Brand           object
Model           object
Body_Type       object
variant         object
No.of Seats     object
Price           object
Fuel_Type       object
location        object
gearbox_Type    object
dtype: object
```

Missing Values Calculation

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Unnamed: 0      0
Brand           0
Model           0
Body_Type       0
```

```
variant         0
No.of Seats     0
Price           0
Fuel_Type       0
location        0
gearbox_Type    0
dtype: int64
```

Data Cleaning

Drop the Unamed column from the data frame

In [12]:

```
df.drop(columns=['Unnamed: 0'],inplace=True)
```

Standardizing Column names

In [13]:

```
df.columns = df.columns.str.title()
df.columns
```

Out[13]:

```
Index(['Brand', 'Model', 'Body_Type', 'Variant', 'No.Of Seats', 'Price',
       'Fuel_Type', 'Location', 'Gearbox_Type'],
      dtype='object')
```

Change the column name and Standardizing Data Types

The No.of Seats column's datatype is object. Converting the object datatypes into integer

In [14]:

```
df['No.Of Seats']
```

Out[14]:

```
0      5 Seater
1      5 Seater
2      5 Seater
3      5 Seater
4      5 Seater
         ...
291    5 Seater
292    5 Seater
293    5 Seater
294    5 Seater
295    5 Seater
Name: No.Of Seats, Length: 296, dtype: object
```

In [15]:

```
df['No.Of Seats'] = df['No.Of Seats'].str.split(' ').str[0]
```

In [16]:

```
df['No.Of Seats']
```

Out[16]:

```
0      5
1      5
2      5
3      5
```

```
4      5
      ..
291    5
292    5
293    5
294    5
295    5
Name: No.Of Seats, Length: 296, dtype: object
```

In [18]:

```
df['No.Of Seats'] = df['No.Of Seats'].astype(int)
```

In [19]:

```
df.rename(columns = {'No.Of Seats' : 'Seater'},inplace = True)
```

In [20]:

```
df['Seater']
```

Out[20]:

```
0      5
1      5
2      5
3      5
4      5
      ..
291    5
292    5
293    5
294    5
295    5
Name: Seater, Length: 296, dtype: int32
```

Converting the Datatype of Price column into Float datatype

Remove ₹ symbol and replacing "L" and "cr" and normalizing the data

In [21]:

```
df['Price']
```

Out[21]:

```
0      ₹13.01 L
1       ₹9.06 L
2       ₹9.50 L
3      ₹10.60 L
4       ₹7.58 L
         ...
291    ₹68.05 L
292    ₹30.31 L
293    ₹54.73 L
294    ₹2.51 Cr
295    ₹78.73 L
Name: Price, Length: 296, dtype: object
```

In [51]:

```
df['Price'] =df['Price'].str.replace("₹",'')
```

In [52]:

```
def convert_price(price_str):
    price_str = str(price_str).strip()
```

```
    if 'L' in price_str:
        return float(price_str.replace('L', '').strip()) * 1e5
    elif 'Cr' in price_str:
        return float(price_str.replace('Cr', '').strip()) * 1e7
    else:

        try:
            return float(price_str)
        except ValueError:
            return np.nan
```

In [53]:

```
df['Price'] = df['Price'].apply(convert_price)
```

In [54]:

```
df['Price'] = df['Price']/100000
df['Price']
```

Out[54]:

```
0       13.01
1        9.06
2        9.50
3       10.60
4        7.58
        ...
291     68.05
292     30.31
293     54.73
294    251.00
295     78.73
Name: Price, Length: 296, dtype: float64
```

In [26]:

```
df.dtypes
```

Out[26]:

```
Brand            object
Model            object
Body_Type        object
Variant          object
Seater            int32
Price           float64
Fuel_Type        object
Location         object
Gearbox_Type     object
dtype: object
```

Capitalizing the first letter of Brand name using title()

In [27]:

```
df['Brand'] = df['Brand'].str.title()
df['Brand']
```

Out[27]:

```
0            Hyundai
1            Hyundai
2                Kia
3                Kia
4      Maruti Suzuki
```

```
          ...
291              Kia
292              Byd
293             Mini
294             Audi
295       Land Rover
Name: Brand, Length: 296, dtype: object
```

Removing year from model column

In [37]:

```
df['Model']
```

Out[37]:

```
0                                    Creta
1                                    Venue
2                                    Sonet
3                                    Syros
4                  Suzuki Baleno (2022-2025)
                        ...
291                          EV6 (2022-2025)
292                           e6 (2022-2024)
293                               Countryman
294                         RS Q8 (2020-2025)
295    Rover Range Rover Evoque (2024-2025)
Name: Model, Length: 296, dtype: object
```

Using Regex extracting only model names from model column

In [38]:

```
import re
df['Model'] = df['Model'].str.replace(r'\s\(\d{4}-\d{4}\)','',regex=True)
```

In [39]:

```
df['Model']
```

Out[39]:

```
0                          Creta
1                          Venue
2                          Sonet
3                          Syros
4                  Suzuki Baleno
                  ...
291                          EV6
292                           e6
293                   Countryman
294                         RS Q8
295    Rover Range Rover Evoque
Name: Model, Length: 296, dtype: object
```

In [40]:

```
models = df['Model'].unique()
index = np.arange(1,len(models)+1)
```

In [41]:

```
pd.Series(models,index = index)
```

Out[41]:

```
1                  Creta
2                  Venue
3                  Sonet
4                  Syros
5           Suzuki Baleno
              ...
247    Martin DB11Coupe4
248       M4 Competition
249                   e6
250           Countryman
251                RS Q8
Length: 251, dtype: object
```

In [42]:

```
body_type = df['Body_Type'].unique()
```

In [43]:

```
pd.Series(body_type)
```

Out[43]:

```
0            SUV
1      Hatchback
2          Sedan
3        MPV/MUV
4    Minivan/Van
5          Coupe
6    Convertible
7        Pick-Up
dtype: object
```

In [45]:

```
df['variant']
```

Out[45]:

```
0                      1.5 E
1                      1.2 E
2                  1.2 HTE(O)
3               1.0 HTK Turbo
4                   1.2 Sigma
               ...
291               GT Line AWD
292                        GL
293    Cooper S JCW Inspired
294                4.0L TFSI
295      SE R-Dynamic Diesel
Name: variant, Length: 296, dtype: object
```

In [46]:

```
df['Seater'].unique()
```

Out[46]:

```
array([5, 7, 6, 8, 4, 2, 9])
```

In [47]:

```
Fuel_type = df['Fuel_Type'].unique()
```

In [48]:

```
pd.Series(Fuel_type)
```

Out[48]:

```
0      Petrol
1      Diesel
2    Electric
3         CNG
dtype: object
```

In [49]:

```
Gear_type = df['gearbox_Type'].unique()
```

In [50]:

```
pd.Series(Gear_type)
```

Out[50]:

```
0       Manual
1    Automatic
dtype: object
```

In [54]:

```
df
```

Out[54]:

	Brand	Model	Body_Type	Variant	No.Of Seats	Price	Fuel_Type	Location	Gearbox_Type	Se
0	Hyundai	Creta	SUV	1.5 E	5	13.01	Petrol	Delhi	Manual	
1	Hyundai	Venue	SUV	1.2 E	5	9.06	Petrol	Delhi	Manual	
2	Kia	Sonet	SUV	1.2 HTE(O)	5	9.50	Petrol	Delhi	Manual	
3	Kia	Syros	SUV	1.0 HTK Turbo	5	10.60	Petrol	Delhi	Manual	
4	Maruti Suzuki	Suzuki Baleno	Hatchback	1.2 Sigma	5	7.58	Petrol	Delhi	Manual	
...	...	...	...	...	...	...	...	...	...	
291	Kia	EV6	SUV	GT Line AWD	5	68.05	Electric	Delhi	Automatic	
292	Byd	e6	SUV	GL	5	30.31	Electric	Delhi	Automatic	
293	Mini	Countryman	SUV	Cooper S JCW Inspired	5	54.73	Petrol	Delhi	Automatic	
294	Audi	RS Q8	SUV	4.0L TFSI	5	251.00	Petrol	Delhi	Automatic	
295	Land Rover	Rover Range Rover Evoque	SUV	SE R-Dynamic Diesel	5	78.73	Diesel	Delhi	Automatic	

296 rows × 10 columns

Data Visualization

Univariate Analysis

Categorial Columns Analysis

Gearbox_Type Distribution.
Chart Type: Bar Chart.
Purpose: To compare the number of cars with manual versus automatic transmissions.
Insights: This chart would quickly reveal the preference for a specific type of transmission among the cars in the dataset.

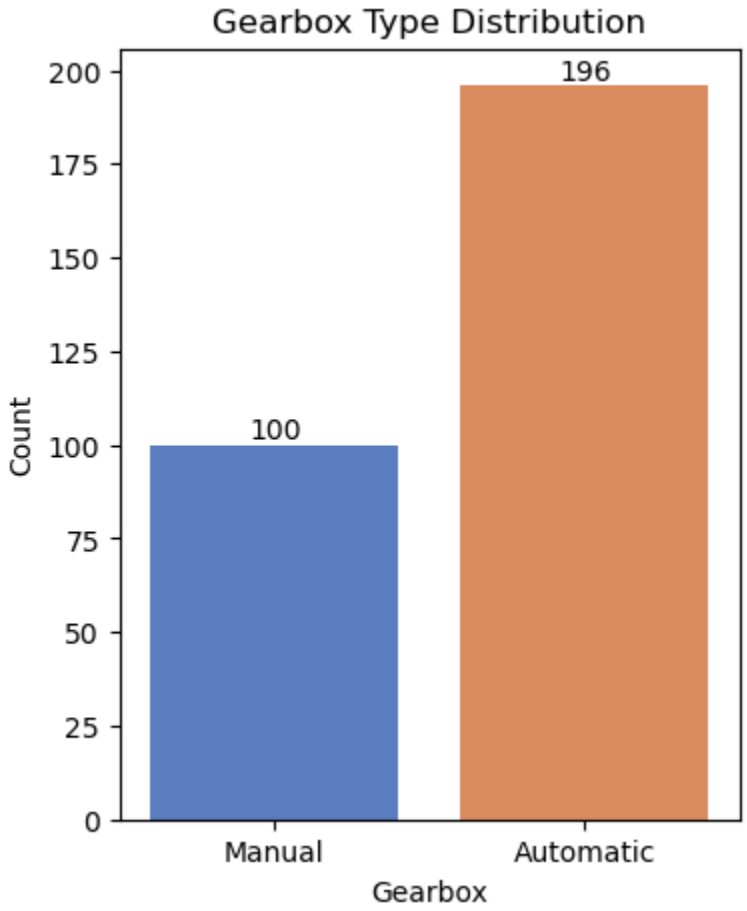
Gear Box Distribution

In [55]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [61]:

```
plt.figure(figsize=(4, 5))
ax= sns.countplot(x='Gearbox_Type', data=df, palette='muted')
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height(
                ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
                textcoords='offset points')
plt.title("Gearbox Type Distribution")
plt.xlabel("Gearbox")
plt.ylabel("Count")
plt.show()
```



Fuel_Type Distribution

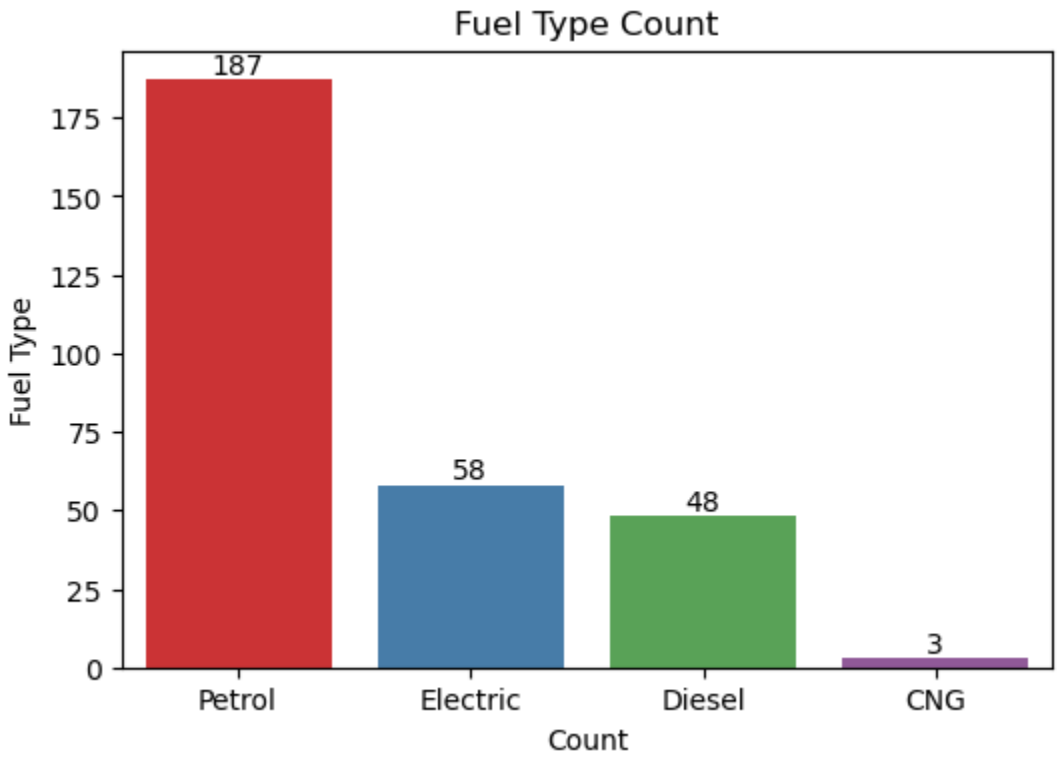
Chart Type: Bar Chart

Purpose: To display the proportion of each fuel type (Petrol, Diesel, Electric, CNG).

Insights: This will enable us find the distribution of cars over fuel types

In [68]:

```
plt.figure(figsize=(6, 4))
Fuel = sns.countplot(x='Fuel_Type', data=df, order=df['Fuel_Type'].value_counts().index,
for p in Fuel.patches:
    Fuel.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_heigh
                ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
                textcoords='offset points')
plt.title("Fuel Type Count")
plt.xlabel("Count")
plt.ylabel("Fuel Type")
plt.show()
```



Brand Distribution

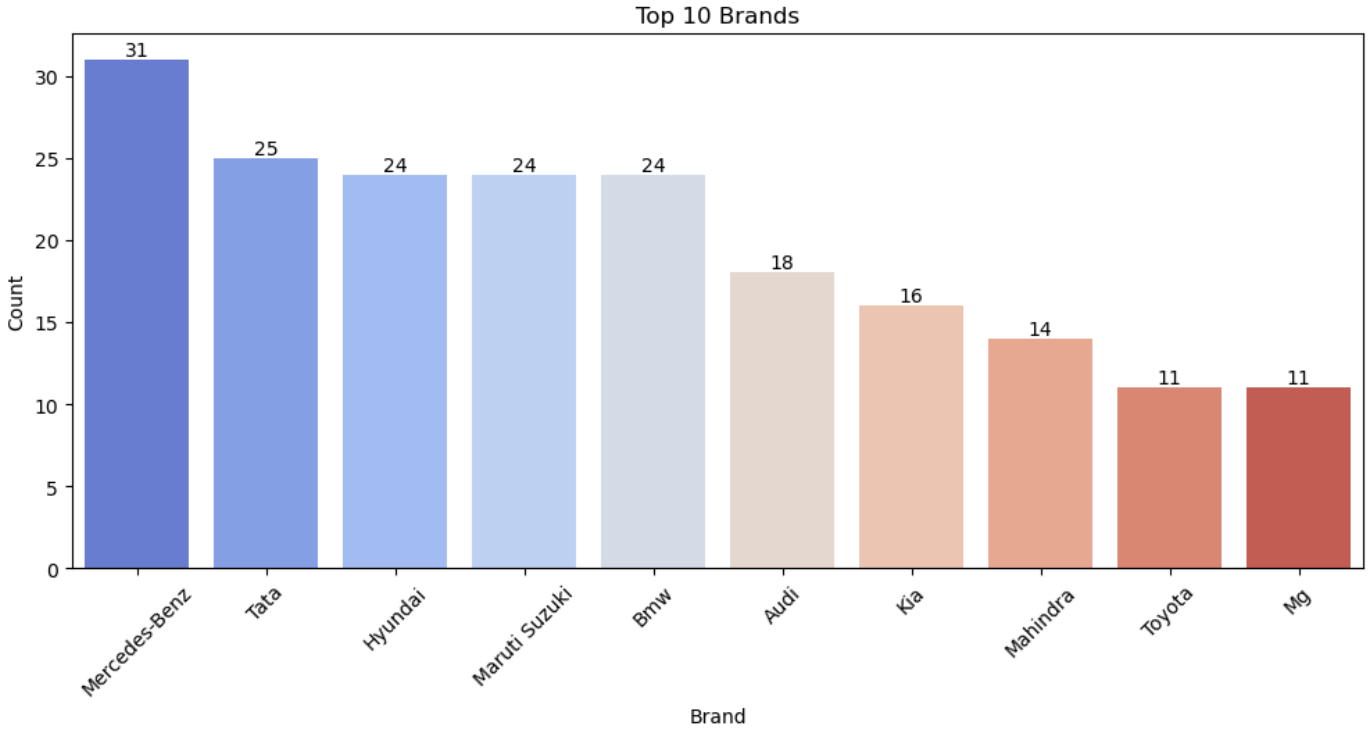
Chart Type: Bar Chart

Purpose: To identify the top car brands listed.

Insights: This visualization would show which brands (e.g., Maruti Suzuki, Hyundai, Tata) have the most models available in the Acko dataset, highlighting key partnerships or inventory focus.

In [71]:

```
top_brands = df['Brand'].value_counts().head(10)
plt.figure(figsize=(12, 5))
bx = sns.barplot(x=top_brands.index, y=top_brands.values, palette='coolwarm')
for p in bx.patches:
    bx.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height(
                ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
                textcoords='offset points')
plt.title("Top 10 Brands")
plt.xlabel("Brand")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```



Body_Type Distribution

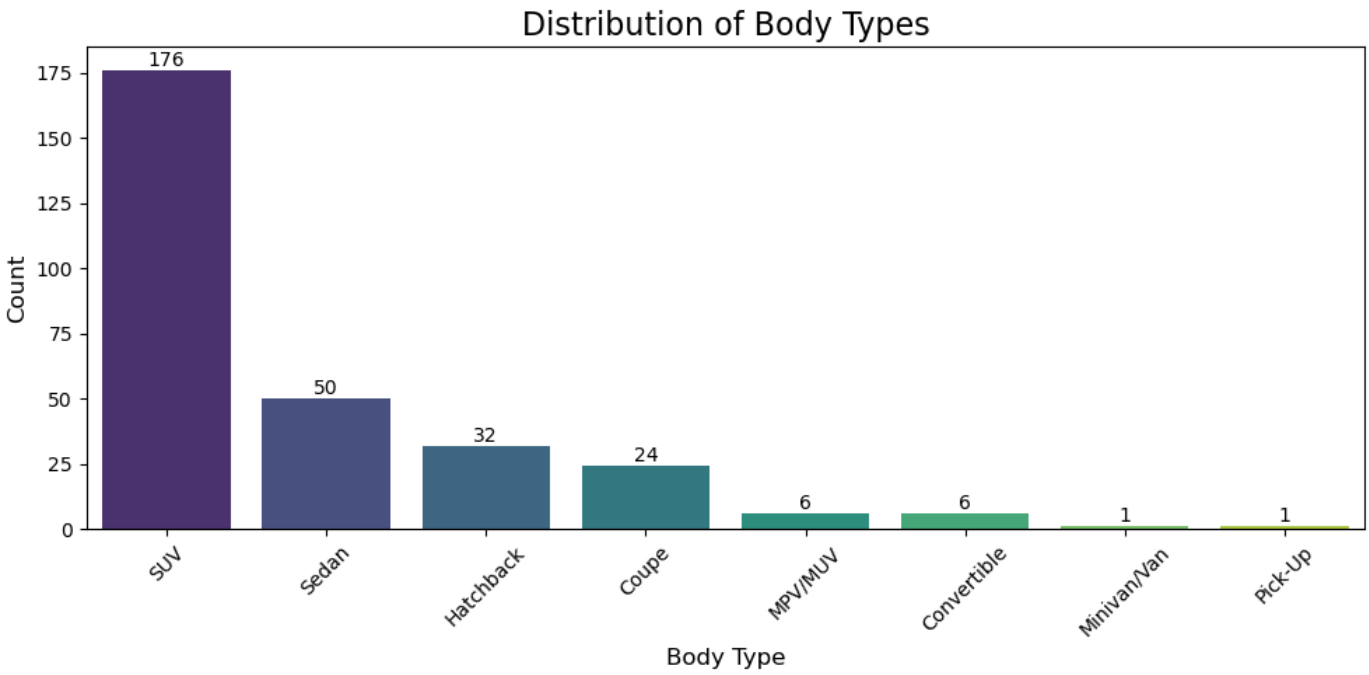
Chart Type: Bar Chart

Purpose: To show the frequency of each car body type in the dataset.

Insights: This visualization would clearly show which body types (e.g., SUV, Hatchback, Sedan) are most common in the Acko car listings, providing an overview of the most popular segments.

In [83]:

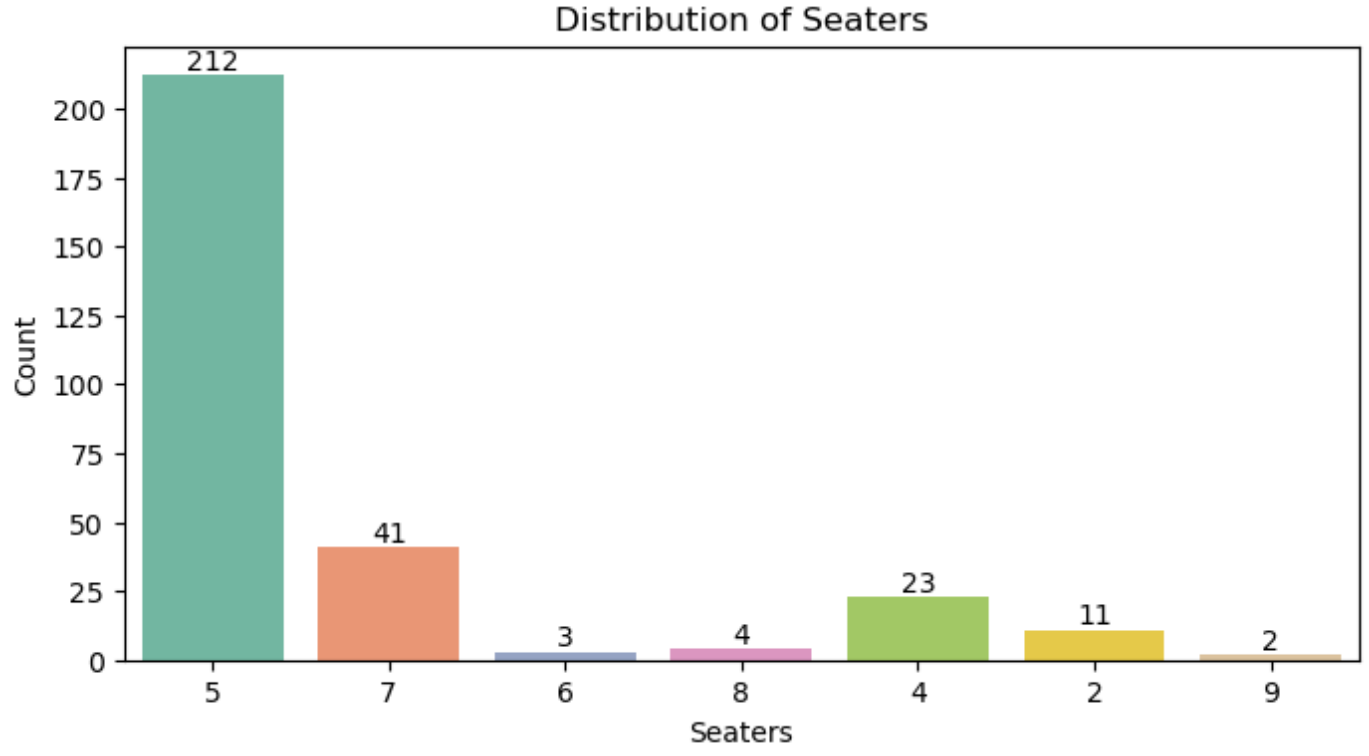
```
plt.figure(figsize=(10, 5))
Body_Type = df['Body_Type'].value_counts()
bt = sns.barplot(x=Body_Type.index, y=Body_Type.values, palette='viridis')
for p in bt.patches:
    bt.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height(
                ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
                textcoords='offset points')
plt.title('Distribution of Body Types', fontsize=16)
plt.xlabel('Body Type', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Numerical Columns

In [86]:

```
plt.figure(figsize=(8, 4))
st = sns.countplot(x='No.Of Seats', data=df, palette='Set2')
for p in st.patches:
    st.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height(
                ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
                textcoords='offset points')
plt.title("Distribution of Seaters")
plt.xlabel("Seaters")
plt.ylabel("Count")
plt.show()
```

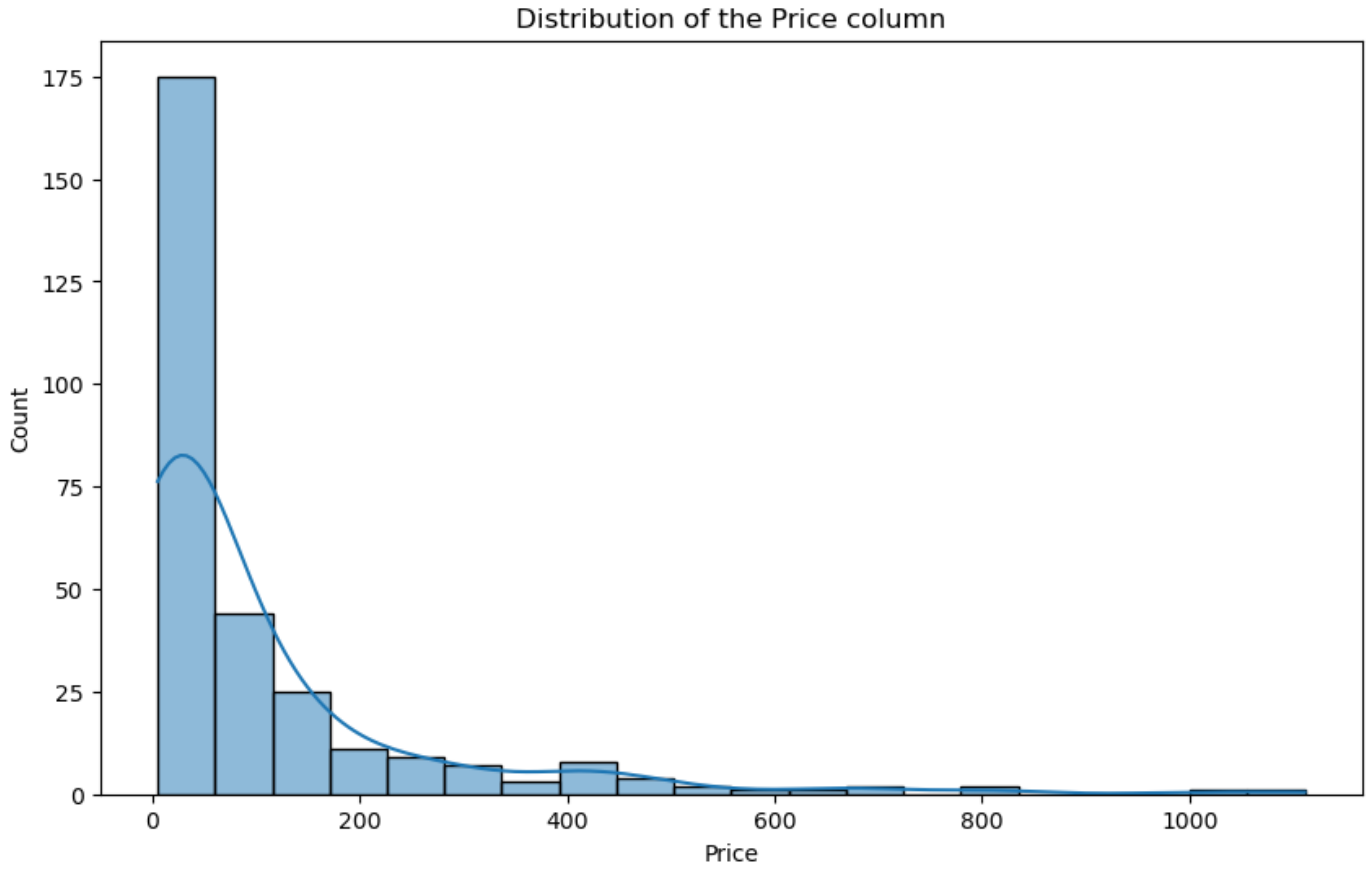


In [91]:

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Price'], kde=True, bins=20)
plt.title("Distribution of the Price column")
```

Out[91]:

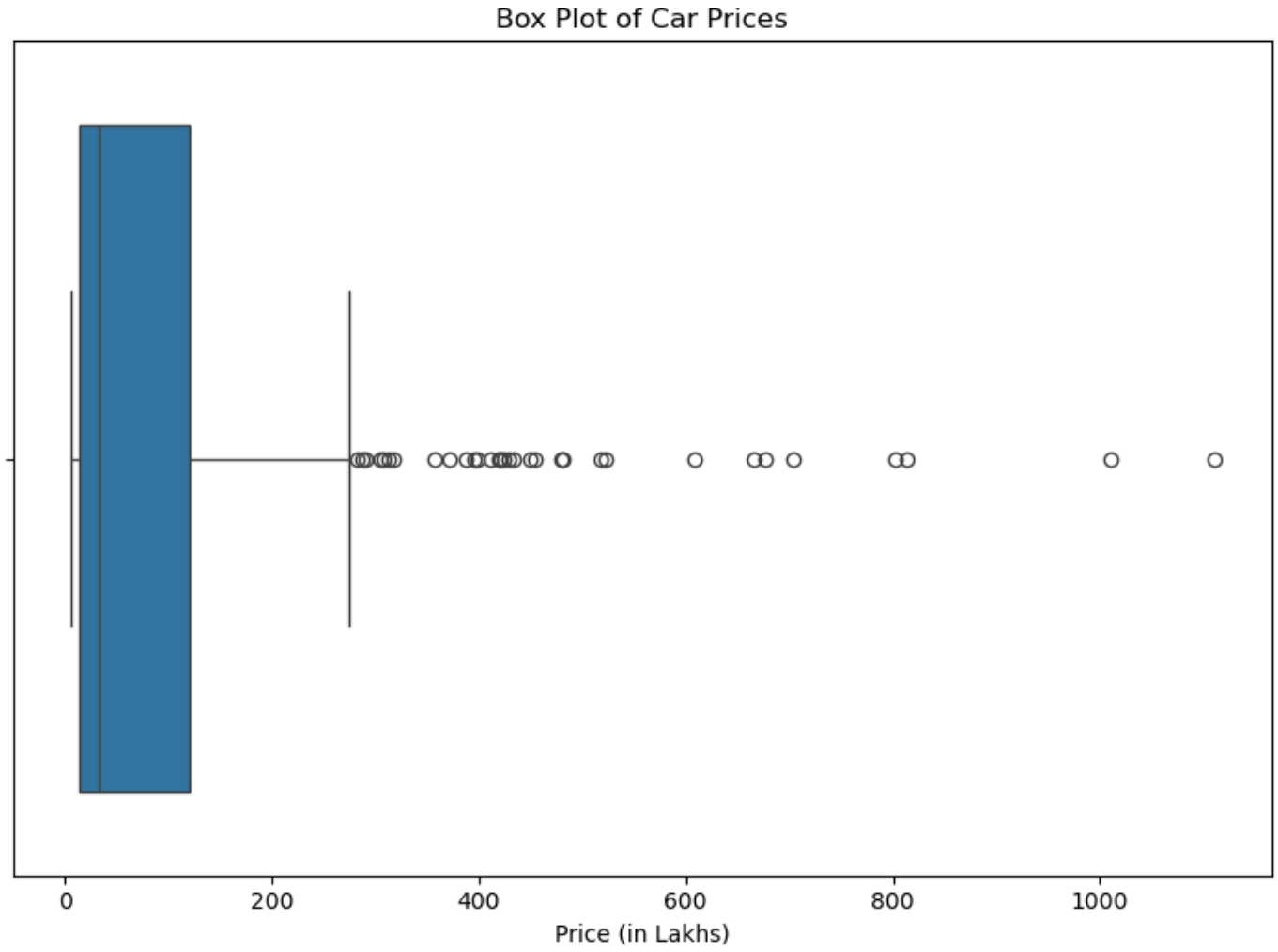
```
Text(0.5, 1.0, 'Distribution of the Price column')
```



Box Plot analysis of price column

In [58]:

```
plt.figure(figsize=(8, 6))
sns.boxplot(x=df['Price'])
plt.title('Box Plot of Car Prices')
plt.xlabel('Price (in Lakhs)')
plt.tight_layout()
plt.show()
Q1 = df['Price'].quantile(0.25)
Q3 = df['Price'].quantile(0.75)
IQR = Q3 - Q1
print(f"First Quartile (Q1): {Q1:.2f}")
print(f"Third Quartile (Q3): {Q3:.2f}")
print(f"Interquartile Range (IQR): {IQR:.2f}")
```




```
First Quartile (Q1): 13.54
Third Quartile (Q3): 120.25
Interquartile Range (IQR): 106.71
```

In [37]:

```
df.dtypes
```

Out[37]:

```
Brand            object
Model            object
Body_Type        object
Variant          object
Seater            int32
Price           float64
Fuel_Type        object
Location         object
Gearbox_Type     object
dtype: object
```

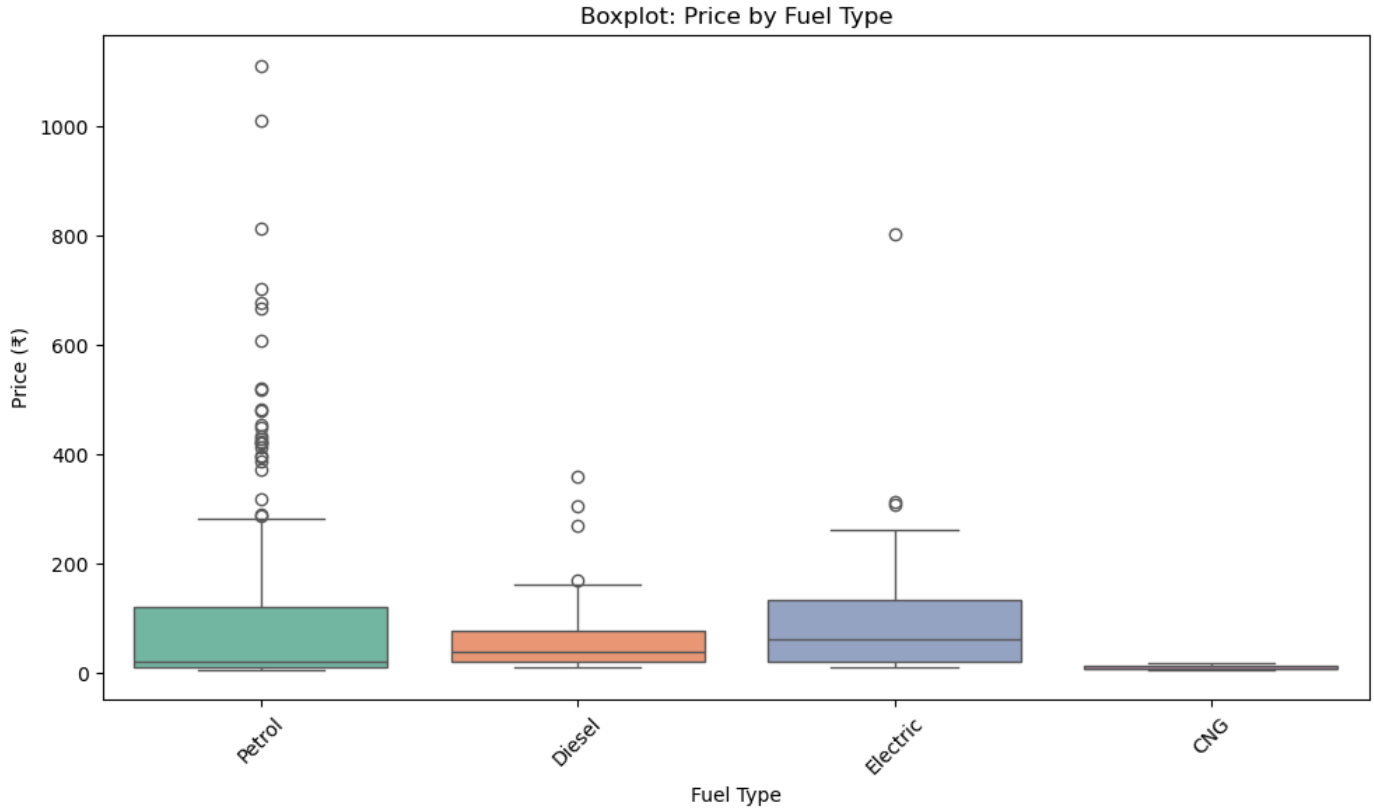
Bivariate Analysis

In [40]:

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='Fuel_Type', y='Price', palette='Set2',legend=False)
plt.title('Boxplot: Price by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Price (₹)')
plt.xticks(rotation=45)
```

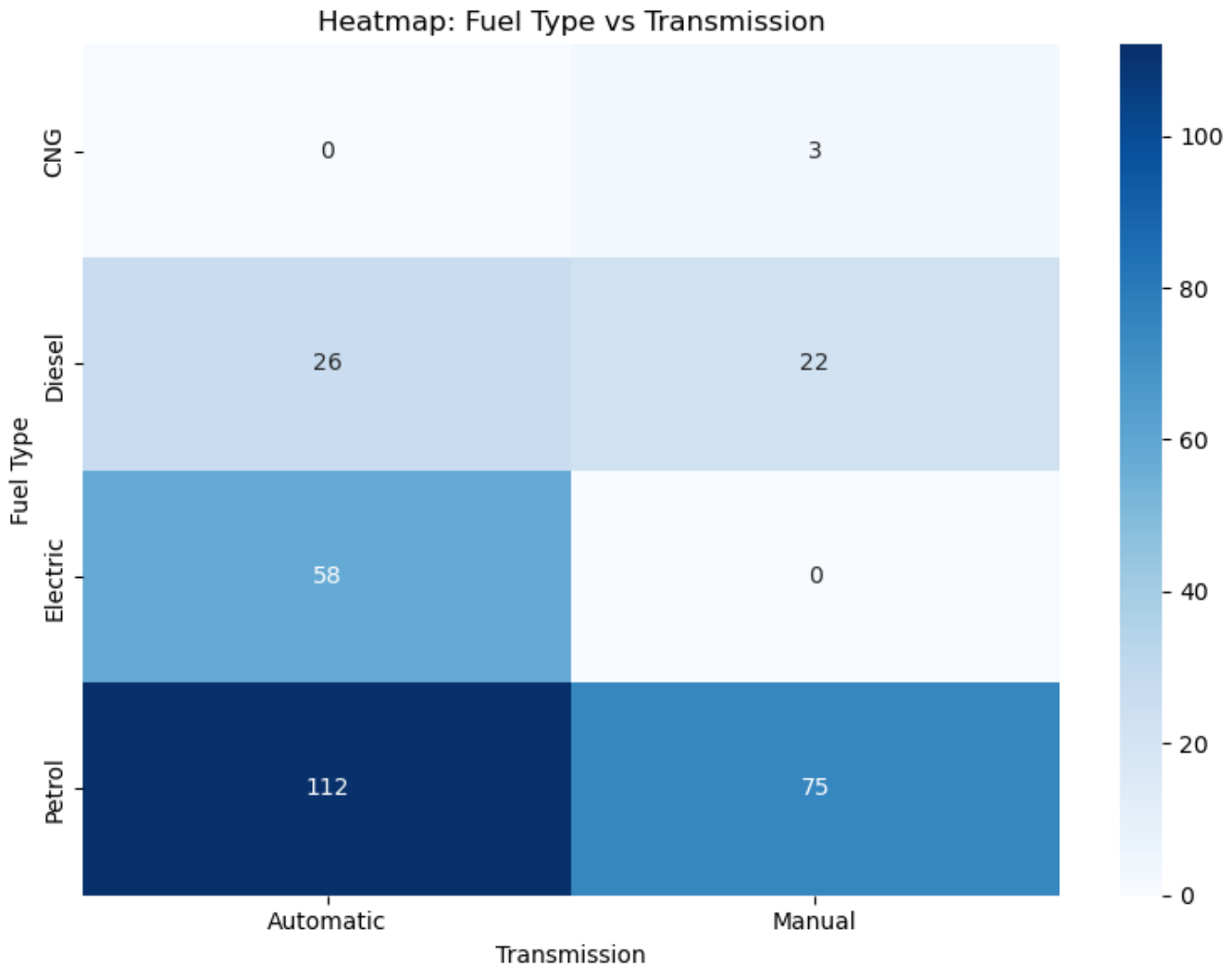
```
plt.tight_layout()
plt.show()
```

```
C:\Users\smomp\AppData\Local\Temp\ipykernel_33116\3171222078.py:2: FutureWarning:

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

  sns.boxplot(data=df, x='Fuel_Type', y='Price', palette='Set2',legend=False)
```



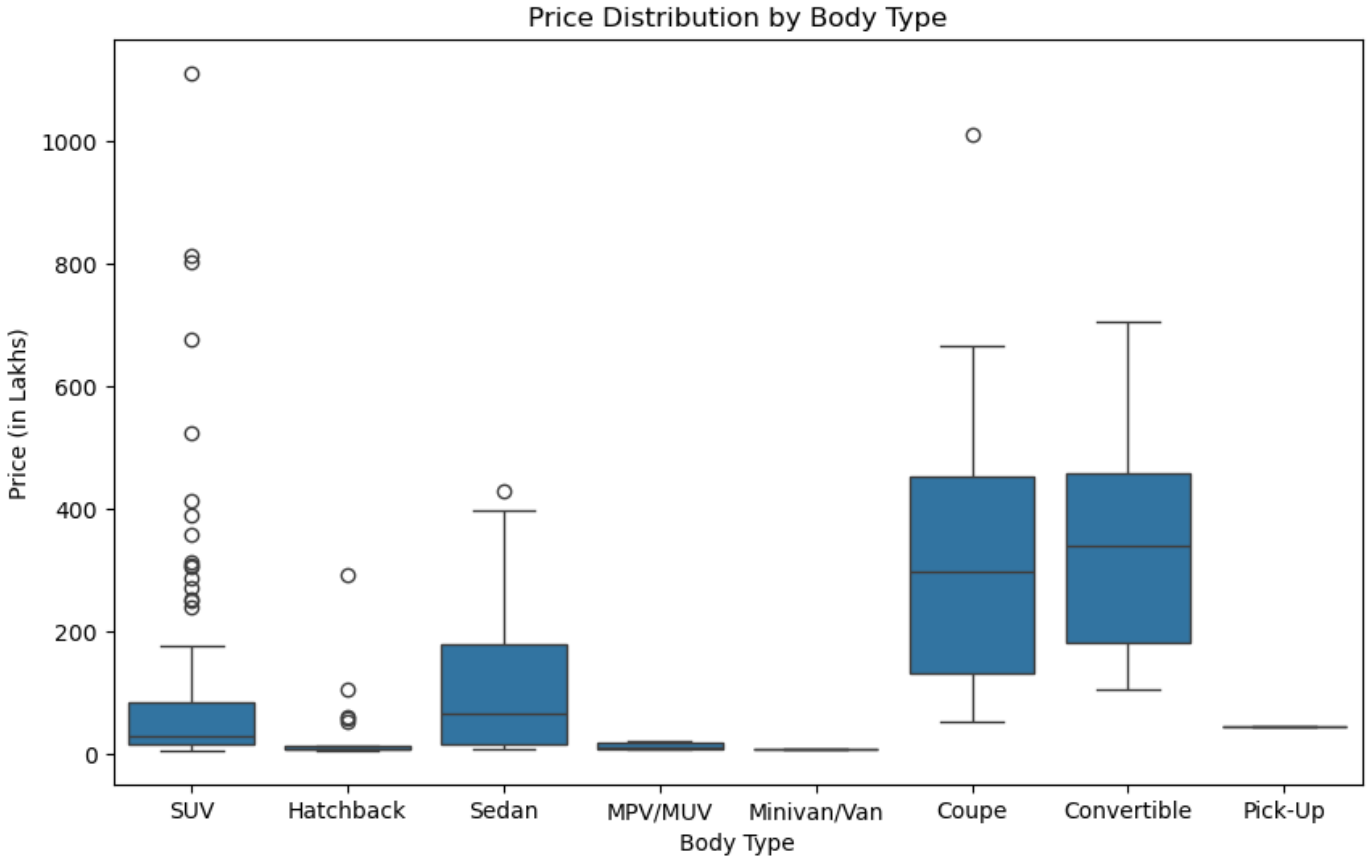
In [56]:

```
crosstab = pd.crosstab(df['Fuel_Type'], df['gearbox_Type'])
plt.figure(figsize=(8, 6))
sns.heatmap(crosstab, annot=True, cmap='Blues', fmt='d')
plt.title('Heatmap: Fuel Type vs Transmission')
plt.xlabel('Transmission')
plt.ylabel('Fuel Type')
plt.tight_layout()
plt.show()
```



In [95]:

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Body_Type', y='Price', data=df)
plt.title('Price Distribution by Body Type')
plt.xlabel('Body Type')
plt.ylabel('Price (in Lakhs)')
plt.show()
```



In [ ]: