In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df=pd.read_csv("used_car_dataset.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Brand	model	Year	Age	kmDriven	Transmission	Owner	FuelType	PostedDate	Additi
0	Honda	City	2001	23	98,000 km	Manual	second	Petrol	Nov-24	Hondateck co vali
1	Toyota	Innova	2009	15	190000.0 km	Manual	second	Diesel	Jul-24	Innov (D 20
2	Volkswagen	VentoTest	2010	14	77,246 km	Manual	first	Diesel	Nov-24	Volks Vento 2013
3	Maruti Suzuki	Swift	2017	7	83,500 km	Manual	second	Diesel	Nov-24	Suzu 2017 Co
4	Maruti Suzuki	Baleno	2019	5	45,000 km	Automatic	first	Petrol	Nov-24	Alph 2019

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9582 entries, 0 to 9581
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Brand         9582 non-null   object
 1   model         9582 non-null   object
 2   Year          9582 non-null   int64 
 3   Age           9582 non-null   int64 
 4   kmDriven      9535 non-null   object
 5   Transmission  9582 non-null   object
 6   Owner         9582 non-null   object
 7   FuelType      9582 non-null   object
 8   PostedDate    9582 non-null   object
 9   AdditionInfo  9582 non-null   object
 10  AskPrice      9582 non-null   object
dtypes: int64(2), object(9)
memory usage: 823.6+ KB
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Brand           0
model           0
Year            0
Age             0
kmDriven       47
Transmission    0
Owner           0
FuelType        0
PostedDate      0
AdditionInfo    0
AskPrice        0
dtype: int64
```

In [7]:
```
df=df.dropna(subset=['kmDriven'])
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Brand           0
model           0
Year            0
Age             0
kmDriven        0
Transmission    0
Owner           0
FuelType        0
PostedDate      0
AdditionInfo    0
AskPrice        0
dtype: int64
```

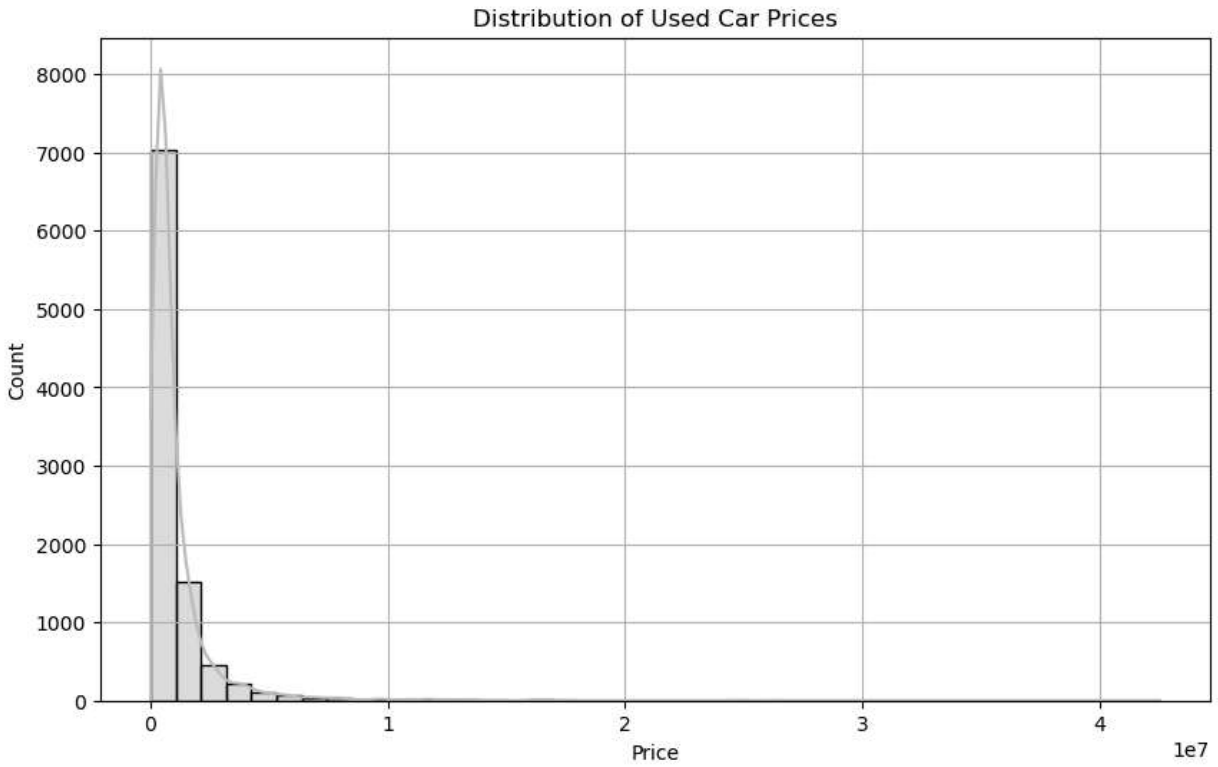
changing data type of ask price as float to plot the graph

In [9]:
```
# Remove ₹ symbol and commas if present, then convert to float
df['AskPrice'] = df['AskPrice'].replace('[₹,]', '', regex=True).astype(float)
print(df['AskPrice'].dtype)
```

```
float64
```

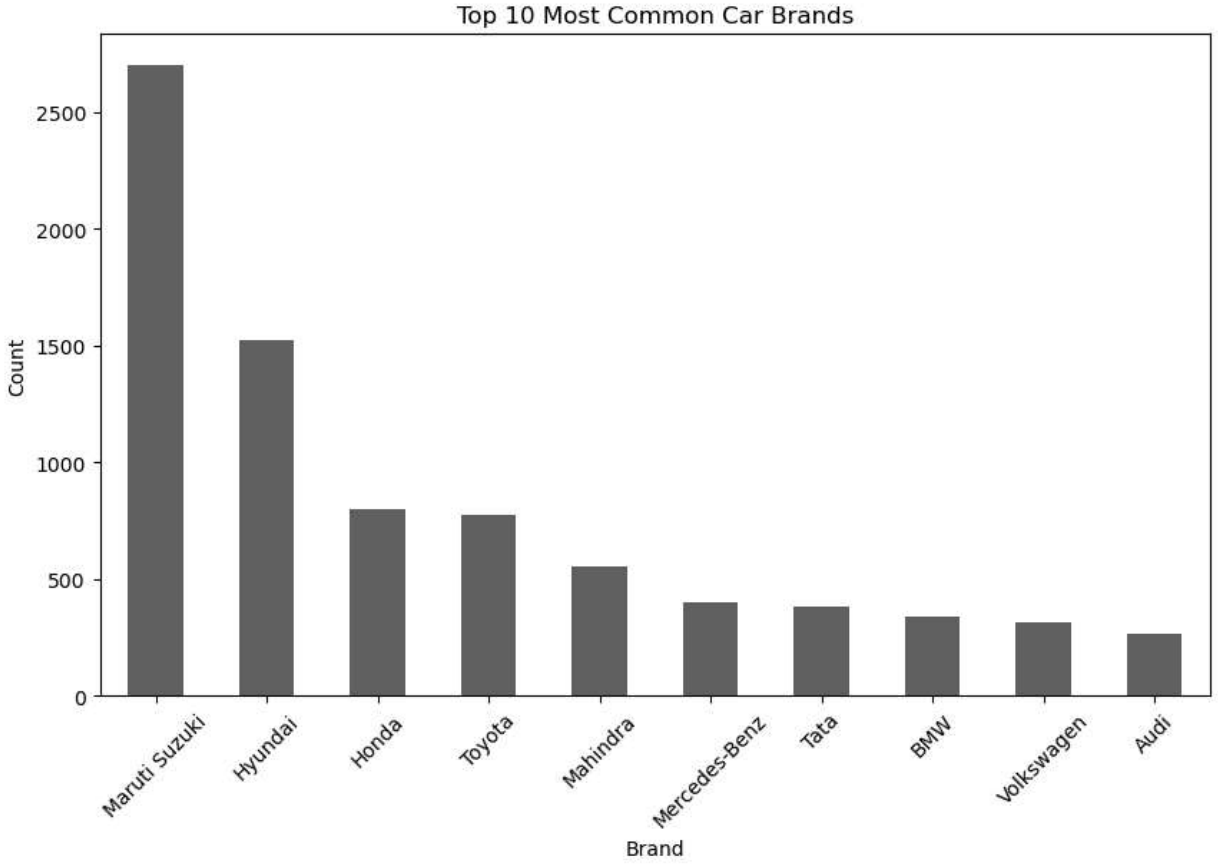
1.DISTRIBUTION OF CAR PRICES

In [11]:
```
plt.figure(figsize=(10,6))

# Use fewer bins (e.g., 40 instead of auto)
sns.histplot(df['AskPrice'], kde=True, bins=40, color='skyblue')

plt.title('Distribution of Used Car Prices')
plt.xlabel('Price')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



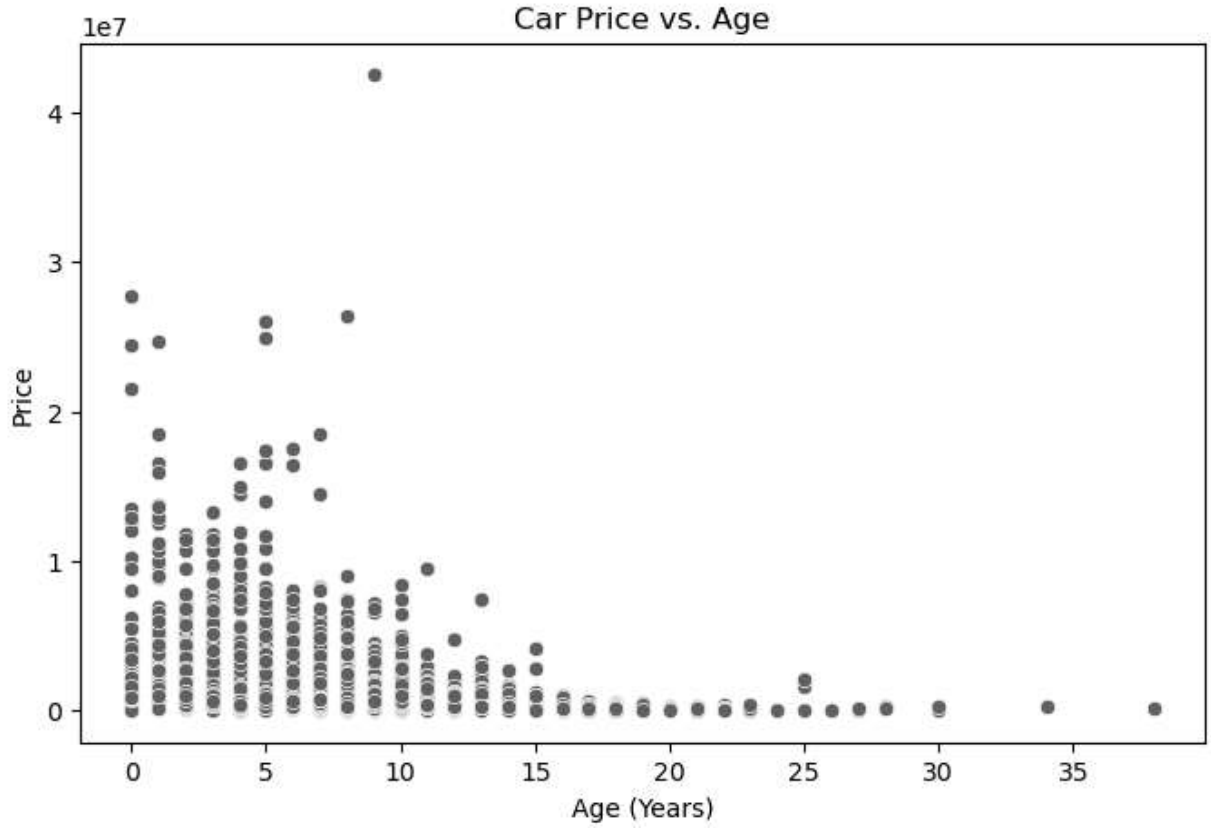
2. Top 10 Most Common Car Brands

In [12]:
```
plt.figure(figsize=(10,6))
df['Brand'].value_counts().head(10).plot(kind='bar')
plt.title('Top 10 Most Common Car Brands')
plt.xlabel('Brand')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



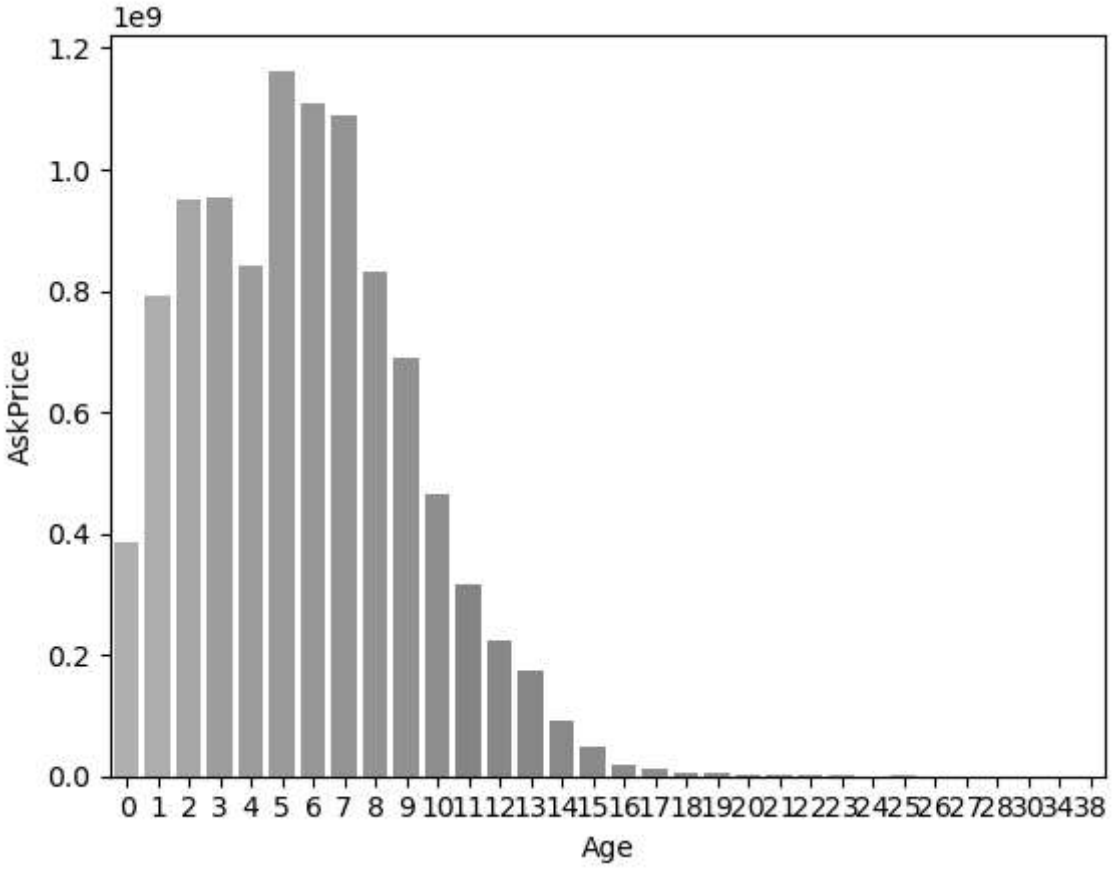
3. Car Price vs. Age

In [13]:
```
plt.figure(figsize=(8,5))
sns.scatterplot(x='Age', y='AskPrice', data=df)
plt.title('Car Price vs. Age')
plt.xlabel('Age (Years)')
plt.ylabel('Price')
plt.show()
```



In [14]:
```
price_age = df.groupby(['Age'],as_index=False)['AskPrice'].sum().sort_values(by=
sns.barplot(x='Age',y='AskPrice',data=price_age)
```

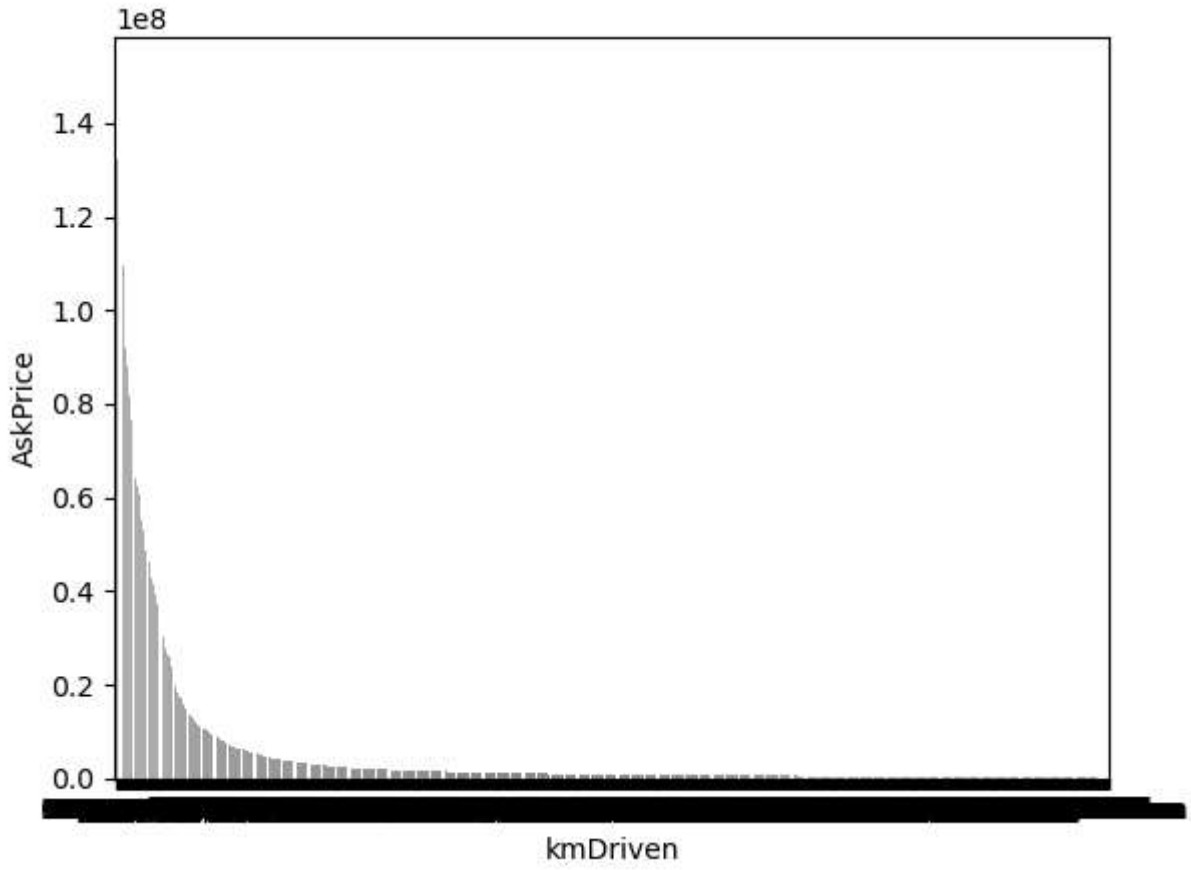
Out[14]: <Axes: xlabel='Age', ylabel='AskPrice'>



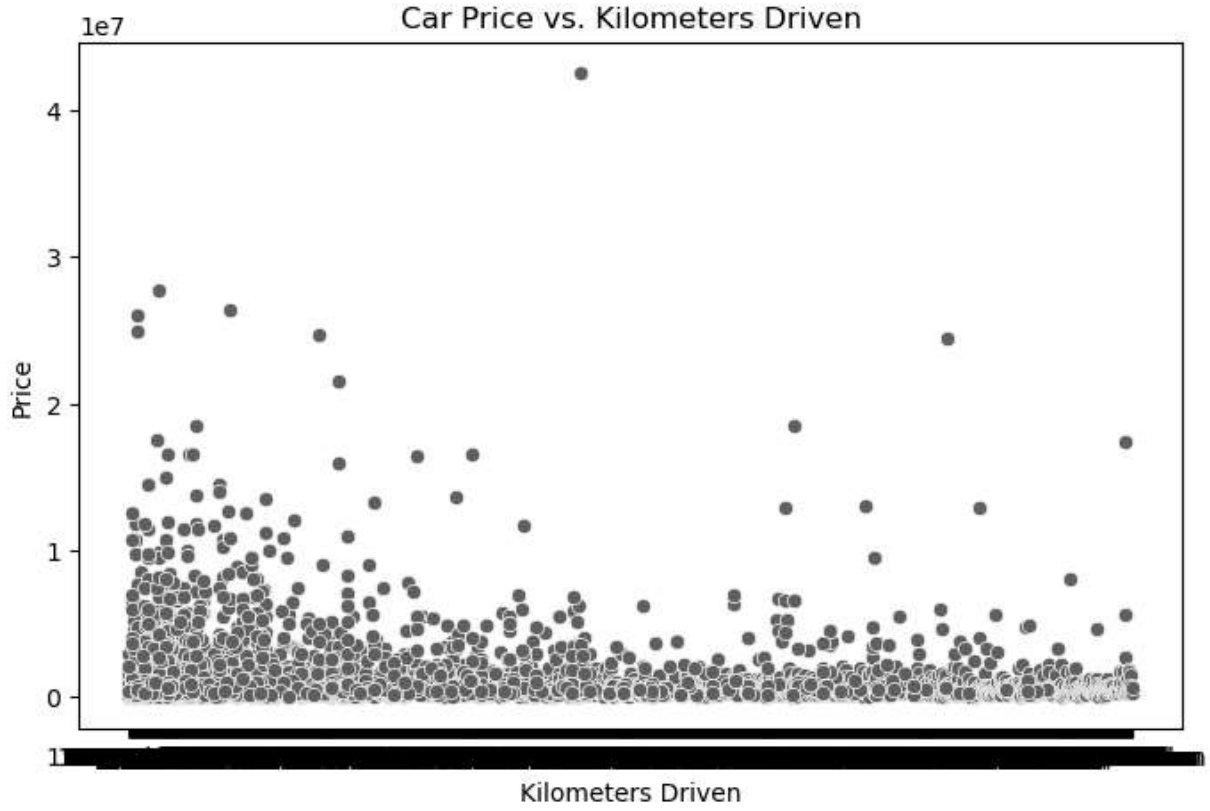
4. Car Price vs. Kilometers Driven

In [15]:
```
price_km= df.groupby(['kmDriven'],as_index=False)['AskPrice'].sum().sort_values(b
sns.barplot(x='kmDriven',y='AskPrice',data=price_km)
```

Out[15]: <Axes: xlabel='kmDriven', ylabel='AskPrice'>



In [16]:
```
plt.figure(figsize=(8,5))
sns.scatterplot(x='kmDriven', y='AskPrice', data=df)
plt.title('Car Price vs. Kilometers Driven')
plt.xlabel('Kilometers Driven')
plt.ylabel('Price')
plt.show()
```



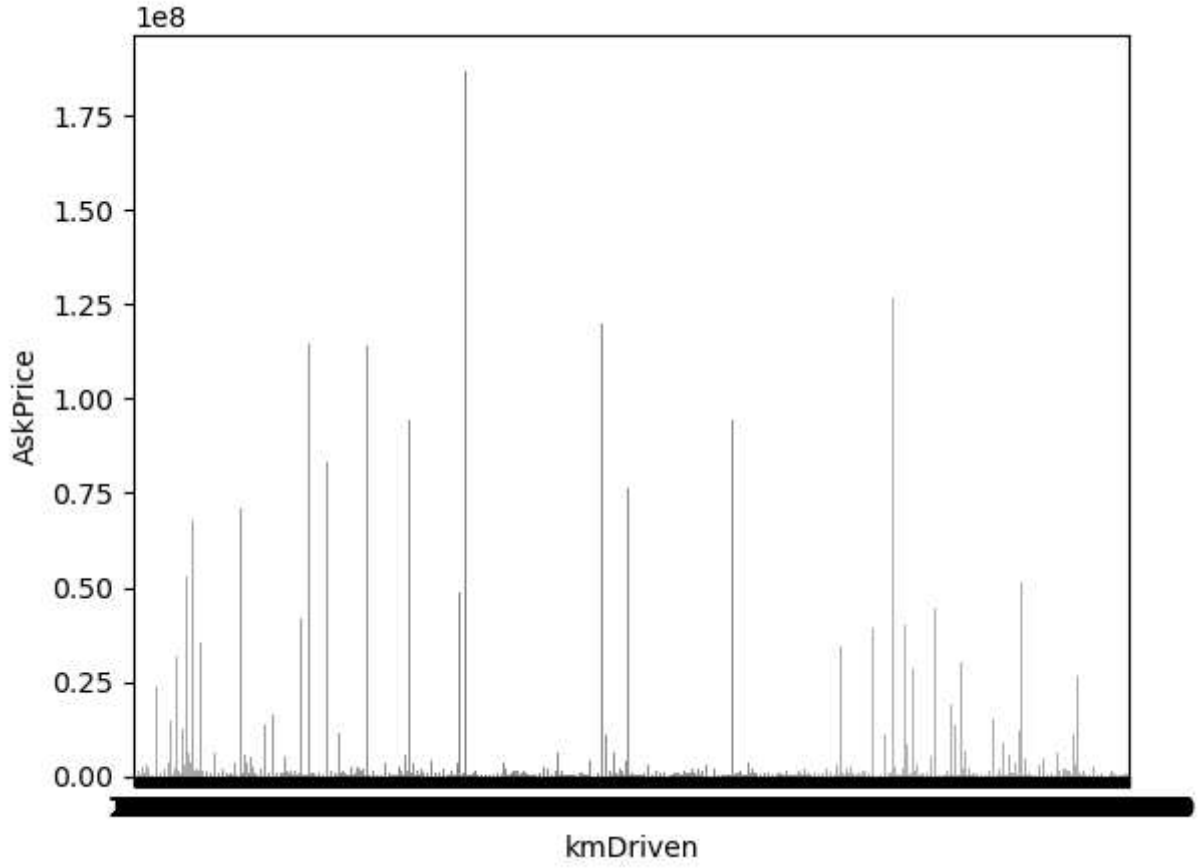
In [17]:
```
# Remove " km" and commas, then convert to float
df['kmDriven'] = df['kmDriven'].str.replace(' km', '', regex=False)
df['kmDriven'] = df['kmDriven'].str.replace(',', '', regex=False)
df['kmDriven'] = df['kmDriven'].astype(float)
```

In [18]:
```
print(df['kmDriven'].dtype)     # should be float64
print(df['kmDriven'].head())    # should show numeric values like 98000.0
```

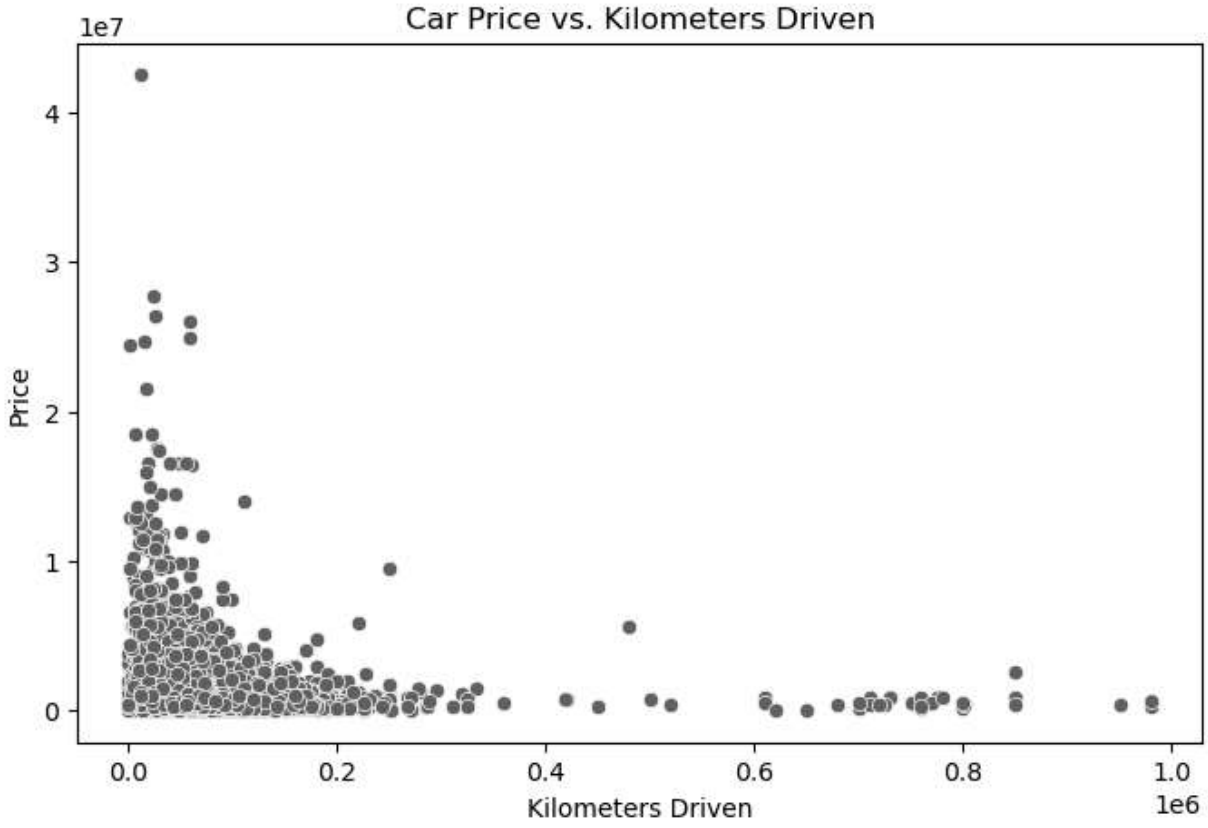
```
float64
0     98000.0
1    190000.0
2     77246.0
3     83500.0
4     45000.0
Name: kmDriven, dtype: float64
```

In [19]:
```
price_km= df.groupby(['kmDriven'],as_index=False)['AskPrice'].sum().sort_values(b
sns.barplot(x='kmDriven',y='AskPrice',data=price_km)
```

Out[19]: <Axes: xlabel='kmDriven', ylabel='AskPrice'>

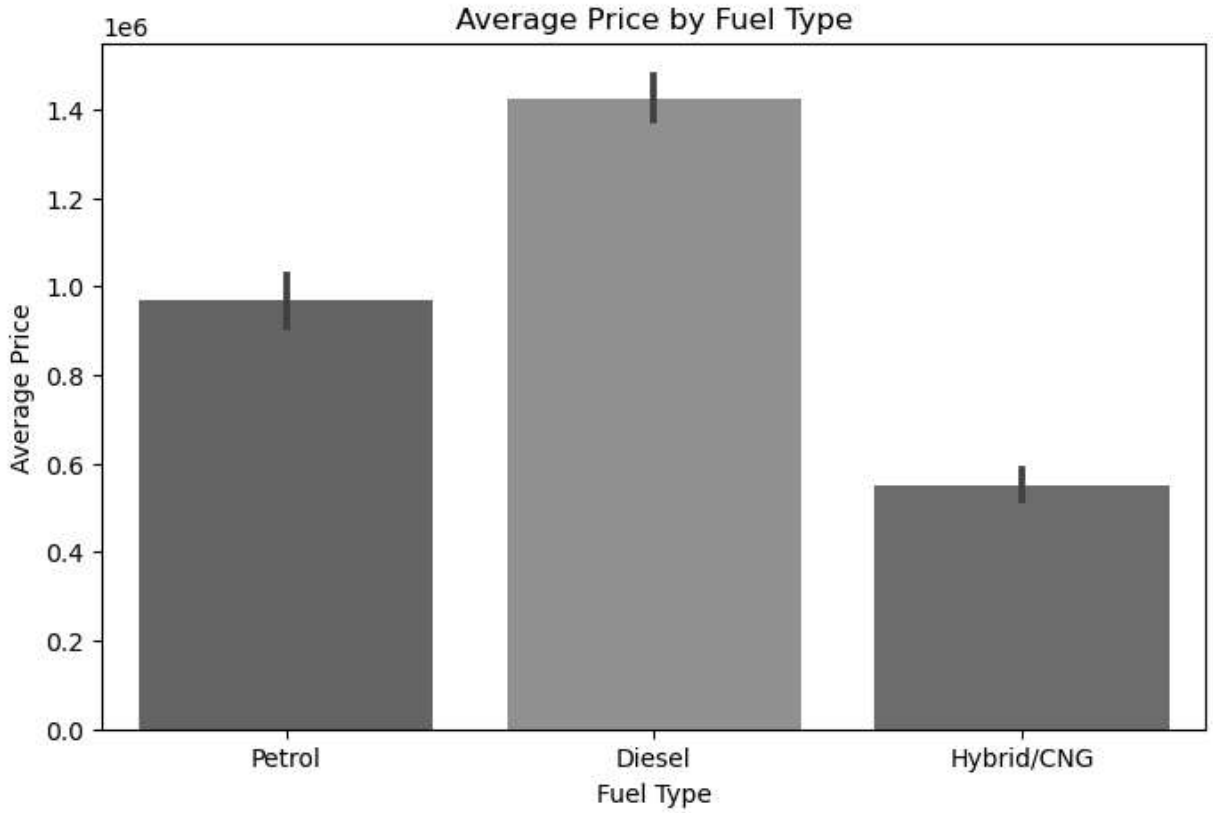


In [20]:
```
plt.figure(figsize=(8,5))
sns.scatterplot(x='kmDriven', y='AskPrice', data=df)
plt.title('Car Price vs. Kilometers Driven')
plt.xlabel('Kilometers Driven')
plt.ylabel('Price')
plt.show()
```



5. Average Price by Fuel Type

In [21]:
```
plt.figure(figsize=(8,5))
sns.barplot(x='FuelType', y='AskPrice', data=df)
plt.title('Average Price by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Average Price')
plt.show()
```



Code to Flag Good-Value Cars:

In [22]:
```
# 1. Model-wise average price and kmDriven
model_avg = df.groupby('model')[['AskPrice', 'kmDriven']].mean().reset_index()
model_avg.columns = ['model', 'avg_price', 'avg_km']
```

In [23]:
```
# 2. Merge with original DataFrame
df = df.merge(model_avg, on='model', how='left')
```

In [24]:
```
# 3. Calculate median age
median_age = df['Age'].median()
```

In [25]:
```
# 4. Flag good-value cars
df['GoodValue'] = (
    (df['AskPrice'] < df['avg_price']) &
    (df['kmDriven'] < df['avg_km']) &
    (df['Age'] <= median_age)
)
```

View Good-Value Cars

In [26]:
```
# Show top good-value cars sorted by price
good_cars = df[df['GoodValue']].sort_values(by='AskPrice')
good_cars[['Brand', 'model', 'Year', 'Age', 'kmDriven', 'AskPrice']]
```

Out[26]:

	Brand	model	Year	Age	kmDriven	AskPrice
3020	Maruti Suzuki	Baleno	2024	0	9000.0	18500.0
5638	Maruti Suzuki	Wagon-R	2020	4	68000.0	49999.0
6694	Volkswagen	Ameo	2017	7	83000.0	50000.0
7344	Maruti Suzuki	Fronx	2024	0	8000.0	100000.0
9430	Maruti Suzuki	Celerio	2023	1	1234.0	130000.0
...	...	...	...	...	...	...
3303	Mercedes-Benz	GLS	2018	6	59950.0	6211000.0
5442	Mercedes-Benz	GLS	2018	6	54855.0	6750000.0
5117	Porsche	Panamera	2017	7	16800.0	6900000.0
8484	Land Rover	Range Rover Sport	2019	5	64000.0	7900000.0
3536	Mercedes-Benz	GLE COUPE	2019	5	30000.0	8000000.0

812 rows × 6 columns

Only diesel cars

In [27]:
```
diesel_cars = good_cars[good_cars['FuelType'].str.lower() == 'diesel']
diesel_cars
```

Out[27]:

	Brand	model	Year	Age	kmDriven	Transmission	Owner	FuelType	PostedDate	A
6694	Volkswagen	Ameo	2017	7	83000.0	Automatic	first	Diesel	Nov-24	
1903	Mahindra	Bolero	2017	7	3.0	Manual	first	Diesel	Nov-24	
8107	Honda	Amaze	2018	6	42000.0	Manual	second	Diesel	Nov-24	
7080	Ford	Free Style	2019	5	51000.0	Manual	first	Diesel	Nov-24	
5453	Maruti Suzuki	Vitara-Brezza	2019	5	65000.0	Manual	first	Diesel	Nov-24	
...	...	...	...	...	...	...	...	...	...	
2122	Mercedes-Benz	GLS	2017	7	53000.0	Automatic	first	Diesel	Nov-24	
3303	Mercedes-Benz	GLS	2018	6	59950.0	Automatic	first	Diesel	Nov-24	
5442	Mercedes-Benz	GLS	2018	6	54855.0	Automatic	first	Diesel	Nov-24	
5117	Porsche	Panamera	2017	7	16800.0	Automatic	second	Diesel	Nov-24	
8484	Land Rover	Range Rover Sport	2019	5	64000.0	Automatic	first	Diesel	Oct-24	

266 rows × 14 columns

only automatic cars

In [28]:
```
automatic_cars = good_cars[good_cars['Transmission'].str.lower() == 'automatic']
automatic_cars
```

Out[28]:

	Brand	model	Year	Age	kmDriven	Transmission	Owner	FuelType	PostedDate
6694	Volkswagen	Ameo	2017	7	83000.0	Automatic	first	Diesel	Nov-24
6275	Maruti Suzuki	Swift-Dzire	2024	0	1234.0	Automatic	first	Hybrid/CNG	Nov-24
5919	Maruti Suzuki	Swift-Dzire-Tour	2020	4	66000.0	Automatic	second	Hybrid/CNG	Nov-24
8997	Maruti Suzuki	Swift Dzire	2024	0	36.0	Automatic	first	Hybrid/CNG	Nov-24
3393	Datsun	GO	2019	5	70000.0	Automatic	second	Petrol	Nov-24
...	...	...	...	...	...	...	...	...	...
3303	Mercedes-Benz	GLS	2018	6	59950.0	Automatic	first	Diesel	Nov-24
5442	Mercedes-Benz	GLS	2018	6	54855.0	Automatic	first	Diesel	Nov-24
5117	Porsche	Panamera	2017	7	16800.0	Automatic	second	Diesel	Nov-24
8484	Land Rover	Range Rover Sport	2019	5	64000.0	Automatic	first	Diesel	Oct-24
3536	Mercedes-Benz	GLE COUPE	2019	5	30000.0	Automatic	first	Petrol	Nov-24

400 rows × 14 columns

Both automatic and diesel cars

In [29]:
```
disel_auto_cars=good_cars[(good_cars['FuelType'].str.lower()=='diesel')
                                        &
                          (good_cars['Transmission'].str.lower()=='automatic')

                          ]
```

In [30]:
```
disel_auto_cars
```

Out[30]:

	Brand	model	Year	Age	kmDriven	Transmission	Owner	FuelType	PostedDate
6694	Volkswagen	Ameo	2017	7	83000.0	Automatic	first	Diesel	Nov-24
3609	Mahindra	Bolero	2022	2	55800.0	Automatic	second	Diesel	Nov-24
3254	Hyundai	Verna	2017	7	70000.0	Automatic	first	Diesel	Nov-24
2193	Maruti Suzuki	Vitara Brezza	2019	5	35000.0	Automatic	second	Diesel	Nov-24

Count of Good-Value Cars per Brand

In [32]:
```
plt.figure(figsize=(10,6))
ax = sns.countplot(data=good_cars, x='Brand', order=good_cars['Brand'].value_cour
plt.title('Good-Value Cars per Brand')
plt.xlabel('Brand')
plt.ylabel('Number of Good-Value Cars')
plt.xticks(rotation=45)
for container in ax.containers:
    ax.bar_label(container)
plt.tight_layout()
plt.show()
```