Vehicle users segmentation

This dataset has been obtained from a famous car dealer website.This dataset contains the details of around 200 indian cars and has features like the price of the car, the type of fuel it consumes, its torque etc. Our goal here is to make user segments according to the vehicle they use.

In [31]:
```python
#importing the required packages
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import OneHotEncoder
```

In [18]:
```python
# Load the datasets
icd = pd.read_csv('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Internships\\Feynn Labs\\Electric vehicle Ind
icd.head()
```

Out[18]:

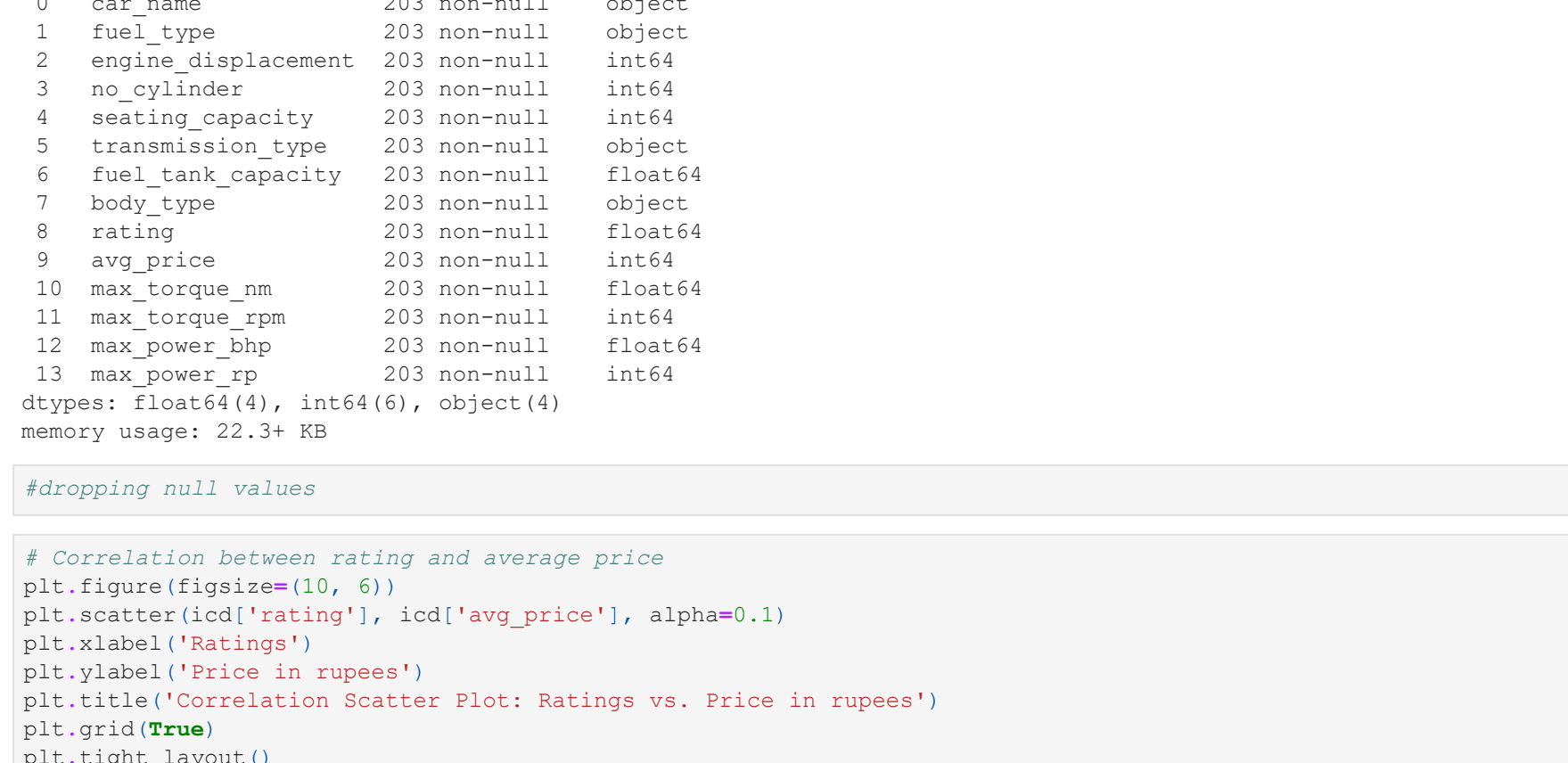
	car_name	fuel_type	engine_displacement	no_cylinder	seating_capacity	transmission_type	fuel_tank_capacity	body_type	rating	avg_price
0	Maruti Alto K10	Petrol	998	3	5	Automatic	27.0	Hatchback	4.5	491000
1	Maruti Brezza	Petrol	1462	4	5	Automatic	48.0	SUV	4.5	1097500
2	Mahindra Thar	Diesel	2184	4	4	Automatic	57.0	SUV	4.5	1478000
3	Mahindra XUV700	Diesel	2198	4	7	Automatic	60.0	SUV	4.5	1888000
4	Mahindra Scorpio-N	Diesel	2198	4	7	Automatic	57.0	SUV	4.5	1794500

In [19]:
```python
#examining the dataset
icd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 203 entries, 0 to 202
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   car_name              203 non-null    object 
 1   fuel_type             203 non-null    object 
 2   engine_displacement   203 non-null    int64  
 3   no_cylinder           203 non-null    int64  
 4   seating_capacity      203 non-null    int64  
 5   transmission_type     203 non-null    object 
 6   fuel_tank_capacity    203 non-null    float64
 7   body_type             203 non-null    object 
 8   rating                203 non-null    float64
 9   avg_price             203 non-null    int64  
 10  max_torque_nm         203 non-null    float64
 11  max_torque_rpm        203 non-null    int64  
 12  max_power_bhp         203 non-null    float64
 13  max_power_rp          203 non-null    int64  
dtypes: float64(4), int64(6), object(4)
memory usage: 22.3+ KB
```

In [20]:
```python
#dropping null values
```

In [21]:
```python
# Correlation between rating and average price
plt.figure(figsize=(10, 6))
plt.scatter(icd['rating'], icd['avg_price'], alpha=0.1)
plt.xlabel('Ratings')
plt.ylabel('Price in rupees')
plt.title('Correlation Scatter Plot: Ratings vs. Price in rupees')
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [22]:
```python
# Calculating the correlation coefficient
correlation_coefficient = icd['rating'].corr(icd['avg_price'])

print(f"Correlation Coefficient between AQI and % Renewable: {correlation_coefficient}")
```

```
Correlation Coefficient between AQI and % Renewable: 0.03350220450871702
```

Comment: Absolutely no correlation between rating and car price

In [23]:
```python
# correlation matrix
correlation_matrix = icd.corr()
correlation_matrix
```

Out[23]:

	engine_displacement	no_cylinder	seating_capacity	fuel_tank_capacity	rating	avg_price	max_torque_nm	max_torque_
engine_displacement	1.000000	0.948265	-0.277966	0.403951	-0.040186	0.632728	0.678812	0.399
no_cylinder	0.948265	1.000000	-0.289816	0.381672	-0.089962	0.590956	0.615500	0.474
seating_capacity	-0.277966	-0.289816	1.000000	-0.013077	0.201402	-0.296028	-0.226848	-0.187
fuel_tank_capacity	0.403951	0.381672	-0.013077	1.000000	0.023962	0.163430	0.094807	0.322
rating	-0.040186	-0.089962	0.201402	0.023962	1.000000	0.033502	-0.011984	-0.041
avg_price	0.632728	0.590956	-0.296028	0.163430	0.033502	1.000000	0.636429	0.107
max_torque_nm	0.678812	0.615500	-0.226848	0.094807	-0.011984	0.636429	1.000000	0.067
max_torque_rpm	0.399129	0.474884	-0.187166	0.322906	-0.041517	0.107133	0.067723	1.000
max_power_bhp	0.723541	0.686595	-0.389978	0.151109	-0.036662	0.675211	0.917634	0.247
max_power_rp	0.427978	0.523751	-0.209913	0.330376	-0.019821	0.090975	0.003525	0.865

In [24]:
```python
# correlation plot
plt.figure(figsize=(8, 6))
sb.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
```

Out[24]:
```
<AxesSubplot: >
```

In [25]:
```python
#Taking a subset
variables = ['engine_displacement', 'no_cylinder', 'seating_capacity', 'fuel_tank_capacity', 'rating', 'avg_pri
             'max_torque_nm', 'max_torque_rpm', 'max_power_bhp', 'max_power_rp']


subset_df = icd[variables]
subset_df
```

Out[25]:

	engine_displacement	no_cylinder	seating_capacity	fuel_tank_capacity	rating	avg_price	max_torque_nm	max_torque_rpm	max_power_bh
0	998	3	5	27.0	4.5	491000	89.0	3500	65.7
1	1462	4	5	48.0	4.5	1097500	136.8	4400	101.6
2	2184	4	4	57.0	4.5	1478000	300.0	2800	130.0
3	2198	4	7	60.0	4.5	1888000	450.0	2800	182.3
4	2198	4	7	57.0	4.5	1794500	400.0	2750	172.4
...	...	...	...	...	...	...	...	...	
198	1991	4	5	0.0	4.5	829000	500.0	5250	415.7
199	1998	4	5	59.0	4.5	1041000	400.0	4400	254.7
200	1956	4	7	60.0	4.5	1845000	350.0	2500	167.6
201	3998	8	5	85.0	3.5	21700000	800.0	4500	591.3
202	796	3	5	35.0	4.5	394000	69.0	3500	47.3

203 rows × 10 columns

In [27]:
```python
#top 10 costliest cars
car_price = icd.groupby('car_name')['avg_price'].sum()

top_ten_costliest_cars = car_price.sort_values(ascending=False).head(10)
top_ten_costliest_cars
```

Out[27]:
```
car_name
Lamborghini Aventador    76250000
Rolls-Royce Ghost        74500000
Jaguar I-Pace            73500000
Rolls-Royce Cullinan     69500000
Rolls Royce Wraith       67150000
Ferrari 812              57500000
Mclaren GT               45000000
BMW X5 M                 41000000
Bentley Bentayga         41000000
Maserati GranTurismo     40200000
Name: avg_price, dtype: int64
```

In [29]:
```python
# Creating a bar graph
plt.figure(figsize=(10, 6))
top_ten_costliest_cars.plot(kind='bar', color='lime')
plt.xlabel('Car model')
plt.ylabel('Car price in Rupees')
plt.title('Top 10 cars by Price')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

In [33]:
```python
# Creating an instance of OneHotEncoder
encoder = OneHotEncoder(sparse=False)

# Fitting and transforming the categorical variables
encoded_categories = encoder.fit_transform(icd[['car_name','fuel_type','transmission_type','body_type']])

# Creating a DataFrame with the one-hot encoded columns
encoded_icd = pd.DataFrame(encoded_categories, columns=encoder.get_feature_names(['car_name','fuel_type','trans
encoded_icd
```

```
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_fe
ature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_fe
ature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[33]:

	car_name_Aston Martin DBX	car_name_Aston Martin Vantage	car_name_Audi A4	car_name_Audi A6	car_name_Audi Q2	car_name_Audi Q5	car_name_Audi Q7	car_name_Audi RS Q8	car_nam
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	
198	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
199	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
200	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
201	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
202	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

203 rows × 217 columns

In [35]:
```python
# Standardizing the numerical features
scaler = StandardScaler()
numerical_cols = ['engine_displacement', 'no_cylinder', 'seating_capacity', 'fuel_tank_capacity', 'rating', 'av
             'max_torque_nm', 'max_torque_rpm', 'max_power_bhp', 'max_power_rp']

encoded_icd[numerical_cols] = scaler.fit_transform(icd[numerical_cols])

encoded_icd[numerical_cols]
```

Out[35]:

	engine_displacement	no_cylinder	seating_capacity	fuel_tank_capacity	rating	avg_price	max_torque_nm	max_torque_rpm	max_power
0	-0.877709	-0.674995	-0.012792	-0.663958	0.237980	-0.679006	-1.326691	0.096474	-1.04
1	-0.566332	-0.280113	-0.012792	0.064376	0.237980	-0.636945	-1.124605	0.662146	-0.85
2	-0.081818	-0.280113	-0.878355	0.376519	0.237980	-0.610557	-0.434634	-0.343494	-0.70
3	-0.072423	-0.280113	1.718335	0.480567	0.237980	-0.582123	0.199530	-0.343494	-0.43
4	-0.072423	-0.280113	1.718335	0.376519	0.237980	-0.588607	-0.011858	-0.374920	-0.48
...	...	...	...	...	...	...	...	...	
198	-0.211335	-0.280113	-0.012792	-1.600387	0.237980	-0.655565	0.410918	1.196393	0.77
199	-0.206637	-0.280113	-0.012792	0.445885	0.237980	-0.640863	-0.011858	0.662146	-0.06
200	-0.234822	-0.280113	1.718335	0.480567	0.237980	-0.585105	-0.223246	-0.532051	-0.51
201	1.135506	1.299414	-0.012792	1.347631	-3.340536	0.791862	1.679246	0.724999	1.68
202	-1.013265	-0.674995	-0.012792	-0.386497	0.237980	-0.685733	-1.411247	0.096474	-1.13

203 rows × 10 columns

In [37]:
```python
# Applying PCA to the combined numerical features
pca = PCA(n_components=2)
pca_result = pca.fit_transform(encoded_icd[numerical_cols])
pca_result
```

Out[37]:
```
array([[-1.83624445e+00, -1.00691774e+00],
       [-1.02044998e+00, -1.54559093e+00],
       [-7.84849524e-01, -1.01972080e-01],
       [-9.70322800e-01,  1.66729611e-01],
       [-1.09757320e+00,  1.17060572e-01],
       [-5.16947777e-01,  1.50518020e-01],
       [-1.30010135e+00, -2.29142486e-01],
       [-1.58120489e+00, -1.14353999e+00],
       [-2.04528000e+00, -1.95726397e-01],
       [-1.18386722e+00, -1.54069733e+00],
       [-1.34230719e+00, -1.66755039e-01],
       [-1.32573068e+00, -1.28079666e+00],
       [-1.29952483e+00, -2.28659495e-01],
       [-6.83567699e-01,  1.34547056e-01],
       [-9.57259026e-01,  1.76195766e-01],
       [-1.24266487e+00, -1.50035817e+00],
       [-1.36759893e+00,  9.53097446e-02],
       [-1.39302884e+00, -1.58083024e+00],
       [-1.28165803e+00, -2.25382465e-01],
       [-1.24359285e+00, -1.50112452e+00],
       [-9.14246302e-01, -1.67137437e+00],
       [ 2.11460283e+00,  8.16555316e-01],
       [-1.46877204e+00, -1.45580584e+00],
       [-1.26066605e+00, -1.28110621e+00],
       [-1.35025059e+00,  8.38675254e-02],
       [-8.33318064e-01,  3.48144282e-02],
       [-1.48881086e+00, -3.11154649e-01],
       [-1.29467810e+00, -2.30483708e-01],
       [-1.33785563e+00, -1.82903025e-01],
       [-1.78274340e+00, -2.15250084e-01],
       [-1.68284848e+00,  2.19676072e-01],
       [-1.67907460e+00, -1.13233230e+00],
       [-1.49640574e+00, -1.14423161e+00],
       [-1.70461006e+00, -1.26507705e+00],
       [ 9.19949631e-01,  5.98367841e-01],
       [-1.71336990e+00, -1.56451838e+00],
       [-3.40488560e+00,  2.71179394e+00],
       [-1.46629950e+00, -1.45390164e+00],
       [-1.49658420e+00, -1.14437898e+00],
       [-1.83574476e+00, -1.00650509e+00],
       [-1.21043989e+00, -9.15208855e-01],
       [ 3.59374777e+00,  6.04826142e-01],
       [-1.82735637e+00,  1.69914061e-01],
       [-1.98061109e+00, -1.39204848e+00],
       [-1.58236549e+00, -3.25133828e-01],
       [-1.42694104e+00, -1.23370227e+00],
       [-8.93324842e-01, -9.56267520e-01],
       [-8.93018242e-01, -9.56017766e-01],
       [-1.32520326e+00, -1.39509052e+00],
       [-1.08365994e+00,  7.92114419e-01],
       [-6.55728988e-01,  2.20456510e+00],
       [-8.58092781e-01, -1.00458988e+00],
       [-1.21161381e+00, -1.54401066e+00],
       [-6.30266967e-01,  3.77473551e-01],
       [-1.73408685e+00, -1.45833452e+00],
       [-9.16141057e-01, -3.51977555e-03],
       [-1.01534388e+00, -1.54149581e+00],
       [-1.04969555e+00, -1.49244641e+00],
       [-1.27838737e+00, -1.71292834e+00],
       [-8.59020767e-01, -1.00535623e+00],
       [ 1.11222609e+00,  6.50686174e-01],
       [ 3.94149649e-01, -2.55722768e-01],
       [-9.09119638e-01, -9.16430038e-01],
       [-1.78602281e-01,  2.03937368e-01],
       [-7.76734864e-01,  1.34781939e-01],
       [ 7.39510044e+00, -1.95702514e-01],
       [-4.82684301e-01, -3.18779926e-01],
       [ 7.99029105e-01, -5.32725187e-01],
       [-5.82225684e-01, -9.38231234e-01],
       [-3.21000570e-01, -1.10027114e+00],
       [ 4.81250338e+00,  3.16897109e+00],
       [-4.15419976e-01, -9.71870937e-01],
       [-2.60011681e-01, -1.18965948e+00],
       [ 1.51425112e-01, -1.29342871e+00],
       [-1.75802029e+00,  2.05188957e+00],
       [-2.60008387e+00,  3.38733202e+00],
       [-1.46621398e+00,  2.19747777e+00],
       [-1.16130978e+00, -1.66857957e+00],
       [ 7.91151690e-02, -8.76867433e-01],
       [-1.68959423e+00,  2.14105306e-01],
       [-1.12737236e-01,  6.07883429e-02],
       [-6.71813725e-01, -1.13301946e+00],
       [-3.08381924e-01, -1.04709914e+00],
       [-4.56359034e-01, -3.13880524e-01],
       [ 1.61578275e+00, -6.56429558e-01],
       [-7.75602305e-02, -6.33992501e-01],
       [-1.25630957e+00, -2.10391597e-01],
       [-1.86895294e+00,  4.30218598e-02],
       [-6.73519952e-01,  1.18111854e+00],
       [ 1.38978033e+00, -1.33000136e+00],
       [-9.33044373e-01,  2.02274615e-02],
       [-2.27397484e+00, -1.41753771e+00],
       [-1.33203277e+00, -1.24258740e+00],
       [-4.19587388e-01, -3.48411971e-01],
       [ 4.40796771e+00, -2.95793227e-01],
       [-3.49663475e+00,  2.58123302e+00],
       [ 1.24366148e+00, -6.84562046e-01],
       [ 3.14853976e-01,  8.31143057e-01],
       [-2.81838563e+00,  3.13942256e+00],
       [ 7.51500597e-01,  1.51493683e+00],
       [ 2.44419402e+00,  1.00293607e+00],
       [-1.57943407e+00, -1.35889589e+00],
       [-7.31075617e-01,  2.52263576e-01],
       [-1.32107476e+00,  7.27696818e-01],
       [ 1.23082830e-01,  1.23165802e+00],
       [-3.19953452e-01, -1.05947109e+00],
       [ 2.67376716e+00, -3.63439478e-01],
       [ 2.97771371e+00, -1.33445089e-01],
       [-8.21620538e-01,  4.00799297e-01],
       [-1.47420749e+00, -1.49295371e+00],
       [-2.10864591e+00,  2.59755221e+00],
       [ 1.70260537e+00, -9.93088385e-01],
       [-1.55561595e-01, -7.56352256e-01],
       [-3.80524477e+00,  2.31814070e+00],
       [ 3.13353809e+00, -5.68064526e-01],
       [-1.61215454e+00, -1.09127881e+00],
       [-1.80515188e+00, -1.08237548e+00],
       [-3.09965193e+00,  2.32148976e+00],
       [-2.44021486e+00,  3.54752334e+00],
       [-1.82499752e+00, -1.55664999e+00],
       [ 2.62162939e+00,  1.45727739e+00],
       [ 3.80277905e+00, -1.55274244e-01],
       [-5.65763791e-02, -1.52815732e+00],
       [-1.87789987e+00,  3.94398844e+00],
       [ 1.82153600e+00, -1.21298515e+00],
       [ 4.72396233e+00,  3.15674720e+00],
       [-1.26748939e+00,  1.46347132e-01],
       [-2.43180469e+00,  3.53479513e+00],
       [ 5.18678513e+00,  4.70215715e-01],
       [ 9.54702660e-01,  8.74527531e-02],
       [ 2.72360694e+00, -2.54289454e-01],
       [-8.91595948e-01,  8.17982250e-01],
       [-5.56677075e-01,  1.09533405e-01],
       [ 3.88104207e+00, -2.79381056e-01],
       [ 4.29780192e+00,  1.55859400e+00],
       [ 1.83279636e+00, -1.01623741e+00],
       [-1.04495600e+00,  3.50720739e-01],
       [ 3.48554006e+00, -8.39405295e-02],
       [ 4.37321086e+00,  1.09629668e+00],
       [-6.64945070e-01,  5.07455269e+00],
       [ 6.36952609e+00, -5.33344558e-01],
       [ 3.01058571e+00,  2.77417927e-01],
       [ 2.88071639e+00,  7.65166060e-01],
       [ 2.01121019e+00,  3.71323607e+00],
       [ 1.20889957e+00,  2.11096439e-02],
       [ 2.89348376e+00, -6.59046407e-01],
       [ 1.59214564e+00, -6.98421900e-01],
       [-3.13196488e+00,  7.54494013e-03],
       [ 6.00269320e+00,  1.30729062e+00],
       [-9.70561035e-01,  2.79251419e-01],
       [ 2.37870994e+00, -3.39652732e-01],
       [ 2.45722700e+00,  1.04502371e+00],
       [-7.08977721e-01,  3.13180691e-02],
       [ 1.03844533e+00,  1.31288053e+00],
       [-1.21834118e-01, -1.28210331e+00],
       [-3.26801922e-01, -1.03408976e+00],
       [ 3.74697048e+00,  3.89520970e-01],
       [ 3.91667070e+00, -7.04648310e-01],
       [-3.00010703e+00,  2.45946427e+00],
       [ 1.77224530e+00, -1.23339152e+00],
       [ 3.43964497e+00, -4.41443918e-01],
       [ 3.77929204e+00, -2.61562750e-02],
       [ 1.41733901e+00,  1.02585226e-01],
       [ 1.72076047e+00,  1.99920591e-01],
       [ 3.09636479e+00, -4.41525191e-01],
       [-8.42723175e-01, -1.21372730e-01],
       [-3.31337531e-01, -1.22568734e-01],
       [ 1.29362690e+00,  2.72096994e+00],
       [ 2.29075251e+00, -7.52348993e-01],
       [ 3.20400910e+00, -4.52612466e-01],
       [-2.14375695e+00,  3.79146136e+00],
       [ 9.83609658e-01, -1.22929255e-01],
       [ 6.21503278e-01, -1.24436993e+00],
       [-9.37998629e-01, -1.43078211e+00],
       [-7.05673547e-01,  4.95894145e+00],
       [ 4.36923213e+00,  7.05897178e-01],
       [ 3.86630189e+00,  1.58136252e-01],
       [ 4.27351083e+00, -1.68004339e+00],
       [-6.29979918e-01,  9.65559588e-02],
       [-4.50915990e-01,  2.48527396e-01],
       [-2.24830787e-01,  4.85931076e-01],
       [ 1.71148159e+00, -4.47055513e-01],
       [ 7.42130984e-01, -3.22346943e-01],
       [-1.24877832e+00,  2.57755144e-02],
       [-3.71649706e+00,  2.48912037e+00],
       [-1.95640809e+00, -3.87006867e-02],
       [-1.51473423e+00,  3.26041996e-01],
       [-2.14608339e+00,  3.70152050e+00],
       [-8.04184888e-01, -1.22975158e+00],
       [-8.79192602e-02, -6.56241715e-01],
       [-4.58101362e-01,  3.31461420e-01],
       [-1.18240525e+00, -1.45059463e+00],
       [-1.12892707e+00, -1.40643133e+00],
       [ 3.03847276e+00,  1.01997555e+00],
       [ 2.59813140e+00,  6.56333268e-01],
       [-7.77030047e-01, -1.66728017e+00],
       [-7.84769351e-01,  1.53112827e-01],
       [-1.24234384e+00,  1.19247765e-02],
       [ 2.52045059e-01, -5.43646961e-01],
       [-1.90560400e-01, -7.27310194e-01],
       [-1.24234384e+00,  1.19247765e-02],
       [ 3.33460923e+00,  5.69236871e-02],
       [-1.85200320e+00, -1.29109689e+00]])
```

In [38]:
```python
# seeing the weightage  of variables in each principal component
loadings = pca.components_
loadings_df = pd.DataFrame(loadings, columns=numerical_cols, index=['PC1', 'PC2'])
print("Variable Loadings in Principal Components:")
print(loadings_df)
```

```
Variable Loadings in Principal Components:
     engine_displacement  no_cylinder  seating_capacity  fuel_tank_capacity  \
PC1             0.439420     0.437180         -0.203132            0.191917   
PC2             0.011951    -0.061751         -0.037203           -0.287947   

       rating  avg_price  max_torque_nm  max_torque_rpm  max_power_bhp  \
PC1 -0.033812   0.343102       0.360873        0.249949       0.406556   
PC2  0.021760   0.283340       0.382589       -0.539030       0.269319   

     max_power_rp  
PC1      0.246648  
PC2     -0.567037  
```

Observation: The first Component can be called the Performance component and the second compnent could be called Efficiency component.

In [39]:
```python
# Creating a new DataFrame with PCA components
pca_icd = pd.DataFrame(data=pca_result, columns=['Performance_component', 'Efficiency_component'])
pca_icd
```

Out[39]:

	Performance_component	Efficiency_component
0	-1.836244	-1.006918
1	-1.020450	-1.545591
2	-0.784850	-0.101972
3	-0.970323	0.166730
4	-1.097573	0.117061
...	...	...
198	0.252045	-0.543647
199	-0.190560	-0.727310
200	-1.242344	0.011925
201	3.334609	0.056924
202	-1.852003	-1.291097

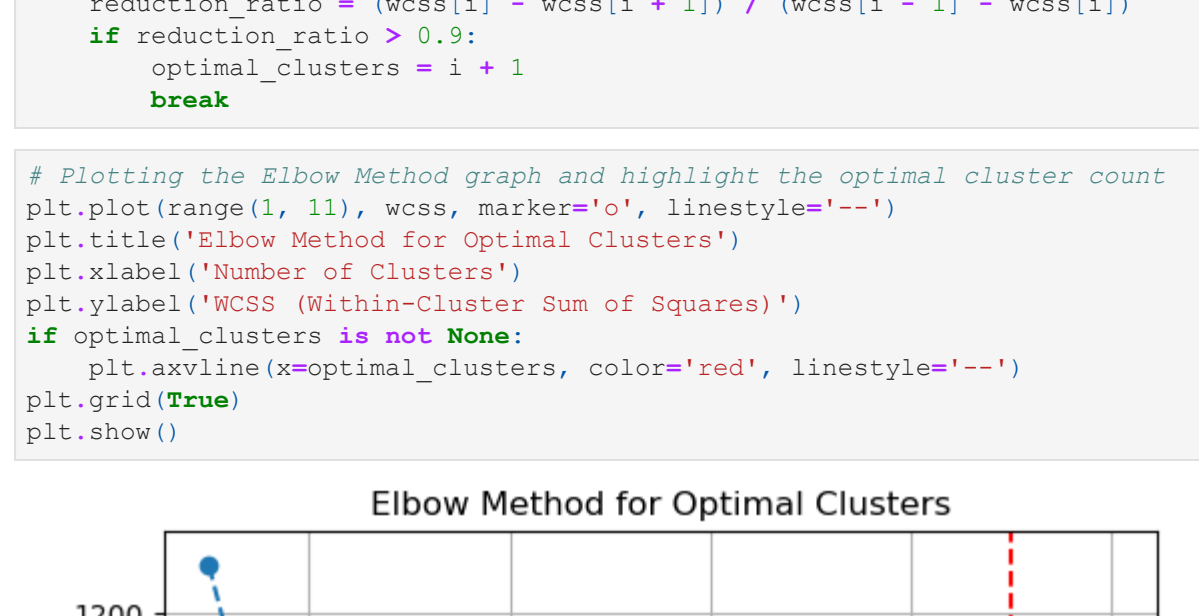
203 rows × 2 columns

In [45]:
```python
# Finding the optimal number of clusters using the Elbow Method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=42)
    kmeans.fit(pca_icd[['Performance_component', 'Efficiency_component']])
    wcss.append(kmeans.inertia_)
```

```
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [46]:
```python
# Determine the optimal number of clusters using the "Elbow Method"
optimal_clusters = None
for i in range(1, len(wcss) - 1):
    reduction_ratio = (wcss[i] - wcss[i + 1]) / (wcss[i - 1] - wcss[i])
    if reduction_ratio > 0.9:
        optimal_clusters = i + 1
        break
```

In [47]:
```python
# Plotting the Elbow Method graph and highlight the optimal cluster count
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS (Within-Cluster Sum of Squares)')
if optimal_clusters is not None:
    plt.axvline(x=optimal_clusters, color='red', linestyle='--')
plt.grid(True)
plt.show()
```



Conclusion:We need to make 9 number of clusters

In [48]:
```python
# Apply K-means clustering with the optimal number of clusters
kmeans = KMeans(n_clusters=optimal_clusters, init='k-means++', max_iter=300, n_init=10, random_state=42)
pca_icd['Cluster'] = kmeans.fit_predict(pca_icd[['Performance_component', 'Efficiency_component']])
pca_icd['Cluster']
```

```
C:\Users\sujoydutta\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known t
o have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it b
y setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[48]:
```
0      3
1      3
2      0
3      0
4      0
      ..
198    6
199    6
200    0
201    1
202    3
Name: Cluster, Length: 203, dtype: int32
```

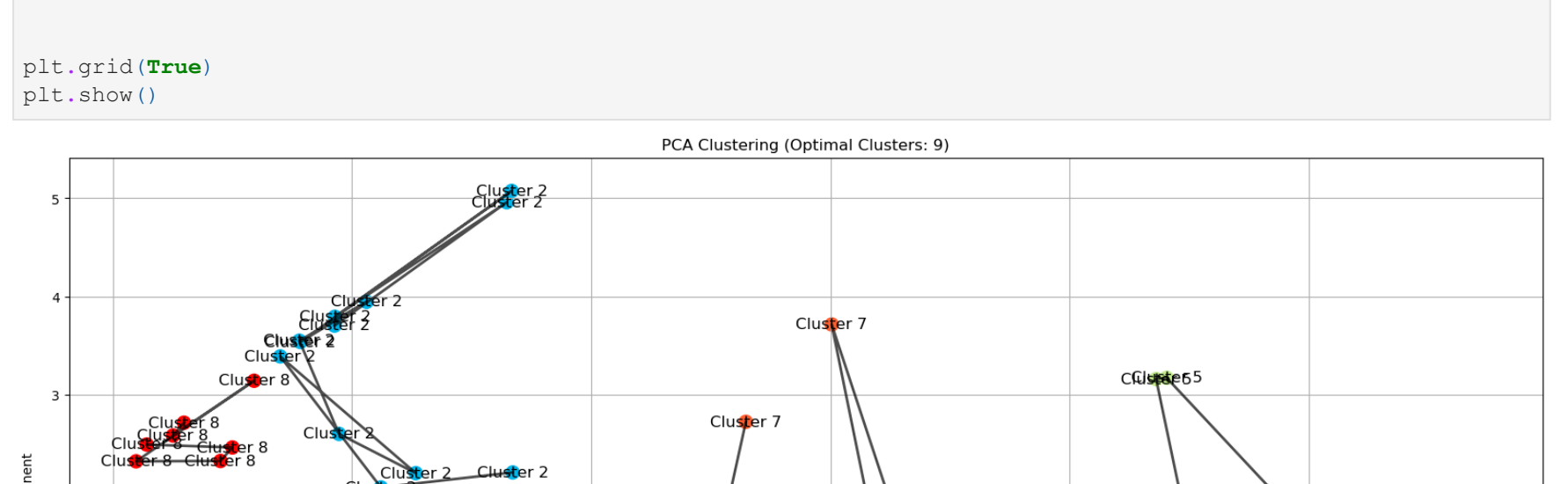
In [55]:
```python
# Plotting the clusters
plt.figure(figsize=(20, 10))
scatter = plt.scatter(pca_icd['Performance_component'], pca_icd['Efficiency_component'], c=pca_icd['Cluster'],


for i, txt in enumerate(pca_icd['Cluster']):
    plt.annotate(f'Cluster {txt}', (pca_icd['Performance_component'][i], pca_icd['Efficiency_component'][i]), f

for cluster_num in set(pca_icd['Cluster']):
    cluster_points = pca_icd[pca_icd['Cluster'] == cluster_num]
    plt.plot(cluster_points['Performance_component'], cluster_points['Efficiency_component'], linestyle='-', li


plt.xlabel('Performance component')
plt.ylabel('Efficiency component')
plt.title(f'PCA Clustering (Optimal Clusters: {optimal_clusters})')


plt.grid(True)
plt.show()
```



Comments:

We can see there is a trade off between Efficiency and Performance.Most Indians are in clusters 3,0 and 6 which means poor Performance and Efficiency. This should not come as a surprise because India is a lower per capita income country. The clusters 2 and 8 are the price sensitive ones who prefer having fuel efficiency. Cluster 4 reflect the Upper Middle class who prefer Performance over efficiency as a social status symbol while the ones in cluster 7 try to take the best from both worlds. Cluster 1 people appear to be the ones who totally disregard efficiency and appear to be young blooded and nouveau riche type of people to whom performance means everything and they dont care about the costs. Finally Cluster 5 appears to be the optimum as they can afford performance as well as efficieny. They appear to be people with generational wealth which is less per cent of the indian population that is India being a poor and backward country. All the cars in this list are fossil fuel vehicles which means Electric Vehicles are not there in the market. I think the company should keep a flexible price range according to the various segments. The clusters 8 and 2 appear to be ideal consumers as they seem to be concerned over fuel costs and hence ideal consumer for electric vehicles as electricity is cheaper than fossil fuels.

In [59]:
```python
# Concatenating the dataframes side by side
icd_consolidated = pd.concat([icd, pca_icd], axis=1)

icd_consolidated
```

Out[59]:

	car_name	fuel_type	engine_displacement	no_cylinder	seating_capacity	transmission_type	fuel_tank_capacity	body_type	rating	avg_pri
0	Maruti Alto K10	Petrol	998	3	5	Automatic	27.0	Hatchback	4.5	4910
1	Maruti Brezza	Petrol	1462	4	5	Automatic	48.0	SUV	4.5	10975
2	Mahindra Thar	Diesel	2184	4	4	Automatic	57.0	SUV	4.5	14780
3	Mahindra XUV700	Diesel	2198	4	7	Automatic	60.0	SUV	4.5	18880
4	Mahindra Scorpio-N	Diesel	2198	4	7	Automatic	57.0	SUV	4.5	17945
...	...	...	...	...	...	...	...	...	...	
198	Mercedes-Benz AMG A 45 S	Petrol	1991	4	5	Automatic	0.0	Hatchback	4.5	8290
199	BMW 3 Series Gran Limousine	Petrol	1998	4	5	Automatic	59.0	Sedan	4.5	10410
200	MG Hector Plus	Diesel	1956	4	7	Manual	60.0	SUV	4.5	18450
201	Audi RS Q8	Petrol	3998	8	5	Automatic	85.0	SUV	3.5	217000
202	Maruti Alto 800 tour	Petrol	796	3	5	Manual	35.0	Hatchback	4.5	3940

203 rows × 17 columns

In [60]:
```python
#seeing the cars by segment
icd_subset = icd_consolidated.iloc[:, [0, -1]]
icd_subset
```

Out[60]:

	car_name	Cluster
0	Maruti Alto K10	3
1	Maruti Brezza	3
2	Mahindra Thar	0
3	Mahindra XUV700	0
4	Mahindra Scorpio-N	0
...	...	...
198	Mercedes-Benz AMG A 45 S	6
199	BMW 3 Series Gran Limousine	6
200	MG Hector Plus	0
201	Audi RS Q8	1
202	Maruti Alto 800 tour	3

203 rows × 2 columns

In [61]:
```python
# Grouping the dataframe by cluster and displaying the respective cars by their cluster
result = icd_subset.groupby('Cluster')['car_name'].agg(', '.join).reset_index()
pd.set_option('display.max_colwidth', None)
result
```

Out[61]:

	Cluster	car_name
0	0	Mahindra Thar, Mahindra XUV700, Mahindra Scorpio-N, Toyota Fortuner, Hyundai Creta, Mahindra Bolero, Tata Nexon, Kia Seltos, Hyundai Tucson, Tata Harrier, Toyota Innova Crysta, Kia Sonet, Mahindra XUV300, MG Hector, Tata Altroz, Mahindra Scorpio Classic, Hyundai Verna, Citroen C3, Honda City, Honda Amaze, Hyundai Alcazar, Mercedes-Benz GLA, Jeep Wrangler, Force Gurkha, Toyota Hilux, MG Gloster, Honda WR-V, Nissan Kicks, Mahindra Marazzo, Mercedes-Benz C-Class, Mahindra Alturas G4, BMW 2 Series, Isuzu D-Max, Citroen C5 Aircross, Isuzu MU-X, Jaguar XF, Mercedes-Benz GLC, Tata Yodha Pickup, Mahindra E Verito, Mercedes-Benz V-Class, Mini Cooper Convertible, Mini Cooper Countryman, Mini Cooper 3 DOOR, Mercedes-Benz CLS, Mahindra Bolero PikUp ExtraLong, Mahindra Bolero Camper, Mahindra Bolero Neo, Mahindra Bolero Neo, Mercedes-Benz GLC Coupe, Compass Trailhawk, Jeep Meridian, MG Hector Plus
1	1	Lamborghini Urus, Ferrari Roma, Mercedes-Benz GLS, Porsche Cayenne, Jaguar F-Pace, Mercedes-Benz AMG GT, Maserati Quattroporte, Ferrari Portofino, Aston Martin DBX, Audi RS7, Mercedes-Benz AMG GLE 63 S, Nissan GT-R, Porsche 911, Bentley Bentayga, Ferrari SF90 Stradale, Maserati GranTurismo, Ferrari F8 Tributo, BMW 8 Series, Aston Martin Vantage, BMW X5 M, Lamborghini Huracan EVO, Porsche Cayenne Coupe, Porsche Cayenne Coupe, Audi RS Q8
2	2	Volvo XC90, Volvo XC40, Kia EV6, Volvo S90, Volvo S60, Audi e-tron, Porsche Taycan, BMW iX, Mercedes-Benz EQS, Mercedes-Benz EQC, Jaguar I-Pace, Audi e-tron GT
3	3	Maruti Alto K10, Maruti Brezza, Tata Punch, Maruti Swift, Hyundai Venue, Maruti Baleno, Maruti Ertiga, Maruti Dzire, Honda City 4th Generation, Tata Tiago, Hyundai i20, Maruti Celerio, Nissan Magnite, Renault KWID, Maruti Alto 800, Tata Tigor, Renault Kiger, Maruti S-Presso, MG Astor, Renault Triber, Hyundai Aura, Maruti Ignis, Maruti XL6, Maruti Eeco, Maruti S-Cross, Maruti Ciaz, Hyundai Santro, Honda Jazz, Bajaj Qute (RE60), Volkswagen Vento, Datsun redi-GO, Datsun GO, Tata Tiago NRG, Mahindra KUV 100 NXT, Datsun GO Plus, Hyundai Xcent Prime, Toyota Glanza, Toyota Glanza, Honda City Hybrid, Maruti Alto 800 tour
4	4	BMW X5, BMW Z4, BMW X6, BMW 7 Series, Porsche Macan, Lexus LS, Mercedes-Benz S-Class, Maserati Levante, Mercedes-Benz G-Class, BMW M2, Lexus LC 500h, Mercedes-Benz GLE, BMW M4 Competition, Mercedes-Benz AMG GLE 53, Mercedes-Benz AMG C 43, Lexus RX, Lexus LX, Mercedes-Benz AMG GLC 43
5	5	Lamborghini Aventador, Rolls-Royce Ghost, Rolls-Royce Cullinan, Bentley Flying Spur, Mercedes-Benz Maybach S-Class, Bentley Continental, Ferrari 812, Rolls Royce Wraith, Rolls Royce Dawn
6	6	Volkswagen Virtus, Skoda Slavia, Skoda Kushaq, Volkswagen Taigun, Audi Q7, Toyota Vellfire, BMW X1, Toyota Camry, Volkswagen Tiguan, Skoda Octavia, Skoda Superb, Audi A6, Land Rover Range Rover Velar, Volvo XC60, Skoda Kodiaq, Audi A4, BMW X3, BMW 3 Series Gran Limousine, Audi Q5, Audi Q2, Jaguar XE, Lexus ES, Lexus NX, Land Rover Discovery Sport, Mercedes-Benz A-Class Limousine, Maruti Swift Dzire Tour, Mercedes-Benz AMG A 35, Mercedes-Benz AMG A 45 S, BMW 3 Series Gran Limousine
7	7	Land Rover Range Rover, Land Rover Defender, BMW X7, Mercedes-Benz E-Class, Land Rover Range Rover Sport, Jaguar F-TYPE, BMW 5 Series, Mercedes-Benz AMG G 63, Mclaren GT, Porsche Panamera, Maserati Ghibli, Maserati GranCabrio
8	8	Hyundai Kona Electric, MG ZS EV, BMW i4, BYD E6, Mini Cooper SE, Strom Motors R3, Tata Nexon EV Max

In [63]:

```
# Seeing the percentage of cars and count of cars in each cluster
result['Count'] = result['car_name'].apply(lambda x: len(x.split(', ')))


total_cars = len(icd_subset)
result['Percentage'] = (result['Count'] / total_cars) * 100


pd.set_option('display.max_colwidth', None)
result
```

Out[63]:

	Cluster	car_name	Count	Percentage
0	0	Mahindra Thar, Mahindra XUV700, Mahindra Scorpio-N, Toyota Fortuner, Hyundai Creta, Mahindra Bolero, Tata Nexon, Kia Seltos, Hyundai Tucson, Tata Harrier, Toyota Innova Crysta, Kia Sonet, Mahindra XUV300, MG Hector, Tata Altroz, Mahindra Scorpio Classic, Hyundai Verna, Citroen C3, Honda City, Honda Amaze, Hyundai Alcazar, Mercedes-Benz GLA, Jeep Wrangler, Force Gurkha, Toyota Hilux, MG Gloster, Honda WR-V, Nissan Kicks, Mahindra Marazzo, Mercedes-Benz C-Class, Mahindra Alturas G4, BMW 2 Series, Isuzu D-Max, Citroen C5 Aircross, Isuzu MU-X, Jaguar XF, Mercedes-Benz GLC, Tata Yodha Pickup, Mahindra E Verito, Mercedes-Benz V-Class, Mini Cooper Convertible, Mini Cooper Countryman, Mini Cooper 3 DOOR, Mercedes-Benz CLS, Mahindra Bolero PikUp ExtraLong, Mahindra Bolero Camper, Mahindra Bolero Neo, Mahindra Bolero Neo, Mercedes-Benz GLC Coupe, Compass Trailhawk, Jeep Meridian, MG Hector Plus	52	25.615764
1	1	Lamborghini Urus, Ferrari Roma, Mercedes-Benz GLS, Porsche Cayenne, Jaguar F-Pace, Mercedes-Benz AMG GT, Maserati Quattroporte, Ferrari Portofino, Aston Martin DBX, Audi RS7, Mercedes-Benz AMG GLE 63 S, Nissan GT-R, Porsche 911, Bentley Bentayga, Ferrari SF90 Stradale, Maserati GranTurismo, Ferrari F8 Tributo, BMW 8 Series, Aston Martin Vantage, BMW X5 M, Lamborghini Huracan EVO, Porsche Cayenne Coupe, Porsche Cayenne Coupe, Audi RS Q8	24	11.822660
2	2	Volvo XC90, Volvo XC40, Kia EV6, Volvo S90, Volvo S60, Audi e-tron, Porsche Taycan, BMW iX, Mercedes-Benz EQS, Mercedes-Benz EQC, Jaguar I-Pace, Audi e-tron GT	12	5.911330
3	3	Maruti Alto K10, Maruti Brezza, Tata Punch, Maruti Swift, Hyundai Venue, Maruti Baleno, Maruti Ertiga, Maruti Dzire, Honda City 4th Generation, Tata Tiago, Hyundai i20, Maruti Celerio, Nissan Magnite, Renault KWID, Maruti Alto 800, Tata Tigor, Renault Kiger, Maruti S-Presso, MG Astor, Renault Triber, Hyundai Aura, Maruti Ignis, Maruti XL6, Maruti Eeco, Maruti S-Cross, Maruti Ciaz, Hyundai Santro, Honda Jazz, Bajaj Qute (RE60), Volkswagen Vento, Datsun redi-GO, Datsun GO, Tata Tiago NRG, Mahindra KUV 100 NXT, Datsun GO Plus, Hyundai Xcent Prime, Toyota Glanza, Toyota Glanza, Honda City Hybrid, Maruti Alto 800 tour	40	19.704433
4	4	BMW X5, BMW Z4, BMW X6, BMW 7 Series, Porsche Macan, Lexus LS, Mercedes-Benz S-Class, Maserati Levante, Mercedes-Benz G-Class, BMW M2, Lexus LC 500h, Mercedes-Benz GLE, BMW M4 Competition, Mercedes-Benz AMG GLE 53, Mercedes-Benz AMG C 43, Lexus RX, Lexus LX, Mercedes-Benz AMG GLC 43	18	8.866995
5	5	Lamborghini Aventador, Rolls-Royce Ghost, Rolls-Royce Cullinan, Bentley Flying Spur, Mercedes-Benz Maybach S-Class, Bentley Continental, Ferrari 812, Rolls Royce Wraith, Rolls Royce Dawn	9	4.433498
6	6	Volkswagen Virtus, Skoda Slavia, Skoda Kushaq, Volkswagen Taigun, Audi Q7, Toyota Vellfire, BMW X1, Toyota Camry, Volkswagen Tiguan, Skoda Octavia, Skoda Superb, Audi A6, Land Rover Range Rover Velar, Volvo XC60, Skoda Kodiaq, Audi A4, BMW X3, BMW 3 Series Gran Limousine, Audi Q5, Audi Q2, Jaguar XE, Lexus ES, Lexus NX, Land Rover Discovery Sport, Mercedes-Benz A-Class Limousine, Maruti Swift Dzire Tour, Mercedes-Benz AMG A 35, Mercedes-Benz AMG A 45 S, BMW 3 Series Gran Limousine	29	14.285714
7	7	Land Rover Range Rover, Land Rover Defender, BMW X7, Mercedes-Benz E-Class, Land Rover Range Rover Sport, Jaguar F-TYPE, BMW 5 Series, Mercedes-Benz AMG G 63, Mclaren GT, Porsche Panamera, Maserati Ghibli, Maserati GranCabrio	12	5.911330
8	8	Hyundai Kona Electric, MG ZS EV, BMW i4, BYD E6, Mini Cooper SE, Strom Motors R3, Tata Nexon EV Max	7	3.448276