EDA of a used car dataset from Kaggle

link for the dataset= https://www.kaggle.com/datasets/lepchenkov/usedcarscatalog

In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# For example, here's several helpful packages to load in


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

In [2]:
```
cars= pd.read_csv('cars.csv')

cars.head(10)
```

Out[2]:

	manufacturer_name	model_name	transmission	color	odometer_value	year_produced	engine_fuel	engine_has
0	Subaru	Outback	automatic	silver	190000	2010	gasoline	
1	Subaru	Outback	automatic	blue	290000	2002	gasoline	
2	Subaru	Forester	automatic	red	402000	2001	gasoline	
3	Subaru	Impreza	mechanical	blue	10000	1999	gasoline	
4	Subaru	Legacy	automatic	black	280000	2001	gasoline	
5	Subaru	Outback	automatic	silver	132449	2011	gasoline	
6	Subaru	Forester	automatic	black	318280	1998	gasoline	
7	Subaru	Legacy	automatic	silver	350000	2004	gasoline	
8	Subaru	Outback	automatic	grey	179000	2010	gasoline	
9	Subaru	Forester	automatic	silver	571317	1999	gasoline	

10 rows × 30 columns

In [3]:
```
cars.shape
```

Out[3]:
```
(38531, 30)
```

In [4]:
```
cars.columns
```

Out[4]:
```
Index(['manufacturer_name', 'model_name', 'transmission', 'color',
       'odometer_value', 'year_produced', 'engine_fuel', 'engine_has_gas',
       'engine_type', 'engine_capacity', 'body_type', 'has_warranty', 'state',
       'drivetrain', 'price_usd', 'is_exchangeable', 'location_region',
       'number_of_photos', 'up_counter', 'feature_0', 'feature_1', 'feature_2',
       'feature_3', 'feature_4', 'feature_5', 'feature_6', 'feature_7',
       'feature_8', 'feature_9', 'duration_listed'],
      dtype='object')
```

In [5]:
```
cars.dtypes
```

Out[5]:
```
manufacturer_name     object
model_name            object
transmission          object
color                 object
```

```
odometer_value         int64
year_produced          int64
engine_fuel           object
engine_has_gas          bool
engine_type           object
engine_capacity      float64
body_type             object
has_warranty            bool
state                 object
drivetrain            object
price_usd            float64
is_exchangeable         bool
location_region       object
number_of_photos       int64
up_counter             int64
feature_0               bool
feature_1               bool
feature_2               bool
feature_3               bool
feature_4               bool
feature_5               bool
feature_6               bool
feature_7               bool
feature_8               bool
feature_9               bool
duration_listed        int64
dtype: object
```

In [6]:
```
cars.info()

#this shows that our data is pretty clean as dtype of each column corresponds well to it
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38531 entries, 0 to 38530
Data columns (total 30 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   manufacturer_name   38531 non-null  object 
 1   model_name          38531 non-null  object 
 2   transmission        38531 non-null  object 
 3   color               38531 non-null  object 
 4   odometer_value      38531 non-null  int64  
 5   year_produced       38531 non-null  int64  
 6   engine_fuel         38531 non-null  object 
 7   engine_has_gas      38531 non-null  bool   
 8   engine_type         38531 non-null  object 
 9   engine_capacity     38521 non-null  float64
 10  body_type           38531 non-null  object 
 11  has_warranty        38531 non-null  bool   
 12  state               38531 non-null  object 
 13  drivetrain          38531 non-null  object 
 14  price_usd           38531 non-null  float64
 15  is_exchangeable     38531 non-null  bool   
 16  location_region     38531 non-null  object 
 17  number_of_photos    38531 non-null  int64  
 18  up_counter          38531 non-null  int64  
 19  feature_0           38531 non-null  bool   
 20  feature_1           38531 non-null  bool   
 21  feature_2           38531 non-null  bool   
 22  feature_3           38531 non-null  bool   
 23  feature_4           38531 non-null  bool   
 24  feature_5           38531 non-null  bool   
 25  feature_6           38531 non-null  bool   
 26  feature_7           38531 non-null  bool   
 27  feature_8           38531 non-null  bool   
 28  feature_9           38531 non-null  bool
```

```
 29  duration_listed    38531 non-null  int64
dtypes: bool(13), float64(2), int64(5), object(10)
memory usage: 5.5+ MB
```

In [7]:
```
cars.isnull().sum()

#only 10 null values in engine capacity
#we will try to get some insights from our data and along the way we will perform more E
```

Out[7]:
```
manufacturer_name     0
model_name            0
transmission          0
color                 0
odometer_value        0
year_produced         0
engine_fuel           0
engine_has_gas        0
engine_type           0
engine_capacity      10
body_type             0
has_warranty          0
state                 0
drivetrain            0
price_usd             0
is_exchangeable       0
location_region       0
number_of_photos      0
up_counter            0
feature_0             0
feature_1             0
feature_2             0
feature_3             0
feature_4             0
feature_5             0
feature_6             0
feature_7             0
feature_8             0
feature_9             0
duration_listed       0
dtype: int64
```

Q1. Which location has the cars with the highest feature count?

In [8]:
```
#using np.where function to convert boolean into binary true=1 and false=0 to get count

cars['Feature_count_0']=np.where(cars['feature_0']==True, 1,0)
cars['Feature_count_1']=np.where(cars['feature_1']==True, 1,0)
cars['Feature_count_2']=np.where(cars['feature_2']==True, 1,0)
cars['Feature_count_3']=np.where(cars['feature_3']==True, 1,0)
cars['Feature_count_4']=np.where(cars['feature_4']==True, 1,0)
cars['Feature_count_5']=np.where(cars['feature_5']==True, 1,0)
cars['Feature_count_6']=np.where(cars['feature_6']==True, 1,0)
cars['Feature_count_7']=np.where(cars['feature_7']==True, 1,0)
cars['Feature_count_8']=np.where(cars['feature_8']==True, 1,0)
cars['Feature_count_9']=np.where(cars['feature_9']==True, 1,0)

cars.head()
```

Out[8]:

	manufacturer_name	model_name	transmission	color	odometer_value	year_produced	engine_fuel	engine_has
0	Subaru	Outback	automatic	silver	190000	2010	gasoline	

1	Subaru	Outback	automatic	blue	290000	2002	gasoline
2	Subaru	Forester	automatic	red	402000	2001	gasoline
3	Subaru	Impreza	mechanical	blue	10000	1999	gasoline
4	Subaru	Legacy	automatic	black	280000	2001	gasoline

5 rows × 40 columns

In [9]:
```
cars['Total_features']= cars['Feature_count_0']+cars['Feature_count_1']+cars['Feature_co

cars['Total_features'] #getting total features
```

Out[9]:
```
0         7
1         4
2         3
3         1
4         4
         ..
38526     5
38527     5
38528     5
38529     2
38530     1
Name: Total_features, Length: 38531, dtype: int32
```

In [10]:
```
df=cars

df.Total_features.value_counts() #getting value counts of each feature type
```

Out[10]:
```
1    14497
2     4973
3     4087
4     3305
5     2972
6     2493
9     2309
7     2128
8     1767
Name: Total_features, dtype: int64
```

In [11]:
```
df.Total_features.unique() #checking for any null values
```

Out[11]:
```
array([7, 4, 3, 1, 5, 9, 6, 2, 8])
```

In [12]:
```
loc_feat=df[['location_region', 'Total_features']] #making new dataframe with only desir

loc_feat
```

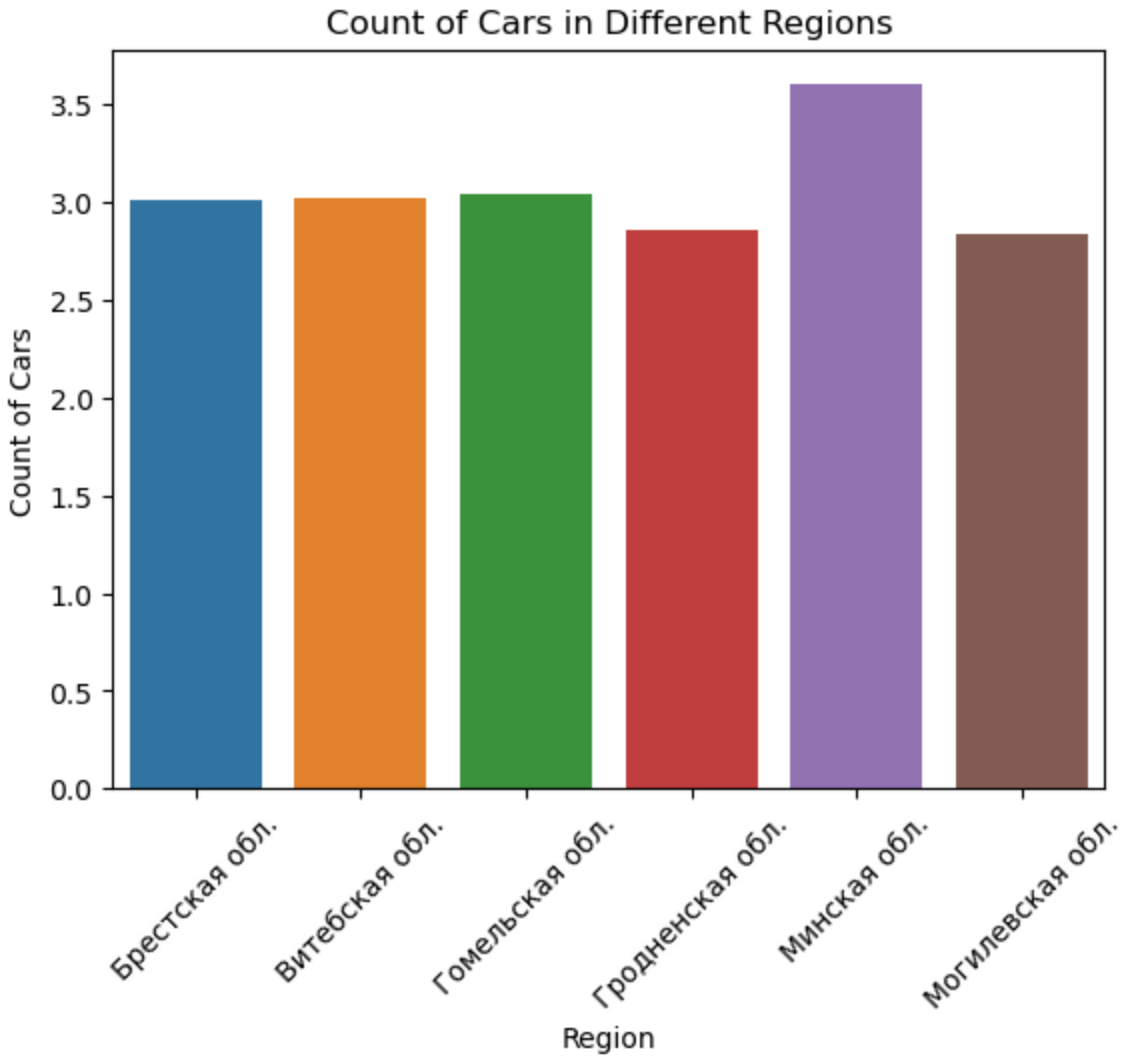
Out[12]:

	location_region	Total_features
0	Минская обл.	7
1	Минская обл.	4
2	Минская обл.	3
3	Минская обл.	1
4	Гомельская обл.	4
...	...	...
38526	Минская обл.	5
38527	Брестская обл.	5

38528	Минская обл.	5
38529	Брестская обл.	2
38530	Минская обл.	1

38531 rows × 2 columns

In [13]:
```
# Group data by 'region' and sum the 'count' within each region
var=loc_feat.groupby('location_region').mean().reset_index()

# Create a bar plot using Seaborn
sns.barplot(data=var, x='location_region', y='Total_features')

# Add labels and a title
plt.xlabel('Region')
plt.ylabel('Count of Cars')
plt.title('Count of Cars in Different Regions')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```



Q2. Regions with highest Asian car manufacturers?

In [14]:
```
#Getting all manufacturer names

df.manufacturer_name.unique()
```

Out[14]:
```
array(['Subaru', 'LADA', 'Dodge', 'УАЗ', 'Kia', 'Opel', 'Москвич',
       'Alfa Romeo', 'Acura', 'Dacia', 'Lexus', 'Mitsubishi', 'Lancia',
       'Citroen', 'Mini', 'Jaguar', 'Porsche', 'SsangYong', 'Daewoo',
       'Geely', 'ВАЗ', 'Fiat', 'Ford', 'Renault', 'Seat', 'Rover',
       'Volkswagen', 'Lifan', 'Jeep', 'Cadillac', 'Audi', 'ЗАЗ', 'Toyota',
       'ГАЗ', 'Volvo', 'Chevrolet', 'Great Wall', 'Buick', 'Pontiac',
       'Lincoln', 'Hyundai', 'Nissan', 'Suzuki', 'BMW', 'Mazda',
       'Land Rover', 'Iveco', 'Skoda', 'Saab', 'Infiniti', 'Chery',
       'Honda', 'Mercedes-Benz', 'Peugeot', 'Chrysler'], dtype=object)
```

In [15]:
```
#list of asian manufacturers
asian=['Kia', 'Mitsubishi', 'Daewoo', 'Toyota', 'Great Wall', 'Honda']

#creating a fiter variable
filt= df['manufacturer_name'].isin(asian)

df1=df[filt]

df1
```

Out[15]:

	manufacturer_name	model_name	transmission	color	odometer_value	year_produced	engine_fuel	engin
808	Kia	Cerato	mechanical	blue	165000	2009	gasoline	
809	Kia	Cee'd	mechanical	black	225275	2007	diesel	
810	Kia	Sportage	automatic	black	104000	2012	gasoline	
811	Kia	Sportage	automatic	black	50000	2017	gasoline	
812	Kia	Sportage	automatic	black	61000	2015	gasoline	
...	...	...	...	...	...	...	...	
33970	Honda	Civic	automatic	white	75000	2009	hybrid-petrol	
33971	Honda	Civic	automatic	grey	155000	2010	hybrid-petrol	
33972	Honda	Accord	mechanical	silver	201000	2001	gasoline	
33973	Honda	Civic	mechanical	grey	156000	2010	gasoline	
33974	Honda	Civic	mechanical	red	9	1993	gasoline	

4099 rows × 41 columns

In [16]:
```
df1.dropna()
```

Out[16]:

	manufacturer_name	model_name	transmission	color	odometer_value	year_produced	engine_fuel	engin
808	Kia	Cerato	mechanical	blue	165000	2009	gasoline	
809	Kia	Cee'd	mechanical	black	225275	2007	diesel	
810	Kia	Sportage	automatic	black	104000	2012	gasoline	
811	Kia	Sportage	automatic	black	50000	2017	gasoline	
812	Kia	Sportage	automatic	black	61000	2015	gasoline	
...	...	...	...	...	...	...	...	
33970	Honda	Civic	automatic	white	75000	2009	hybrid-	

							petrol
33971	Honda	Civic	automatic	grey	155000	2010	hybrid-petrol
33972	Honda	Accord	mechanical	silver	201000	2001	gasoline
33973	Honda	Civic	mechanical	grey	156000	2010	gasoline
33974	Honda	Civic	mechanical	red	9	1993	gasoline

4099 rows × 41 columns

In [17]:
```
#whenever dealing with string columns then it is best to keep data in one case
df1['location_region']=df1['location_region'].str.upper()

df1['location_region']
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_16324\1692310947.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['location_region']=df1['location_region'].str.upper()
```

Out[17]:
```
808           МИНСКАЯ ОБЛ.
809           МИНСКАЯ ОБЛ.
810           МИНСКАЯ ОБЛ.
811           МИНСКАЯ ОБЛ.
812           МИНСКАЯ ОБЛ.
               ...
33970         МИНСКАЯ ОБЛ.
33971      ГОМЕЛЬСКАЯ ОБЛ.
33972         МИНСКАЯ ОБЛ.
33973     ГРОДНЕНСКАЯ ОБЛ.
33974         МИНСКАЯ ОБЛ.
Name: location_region, Length: 4099, dtype: object
```

In [18]:
```
#creating a new dataset with loation and counts for display
df2= pd.DataFrame(df1.location_region.value_counts())

df2.reset_index(inplace=True)

df2
```

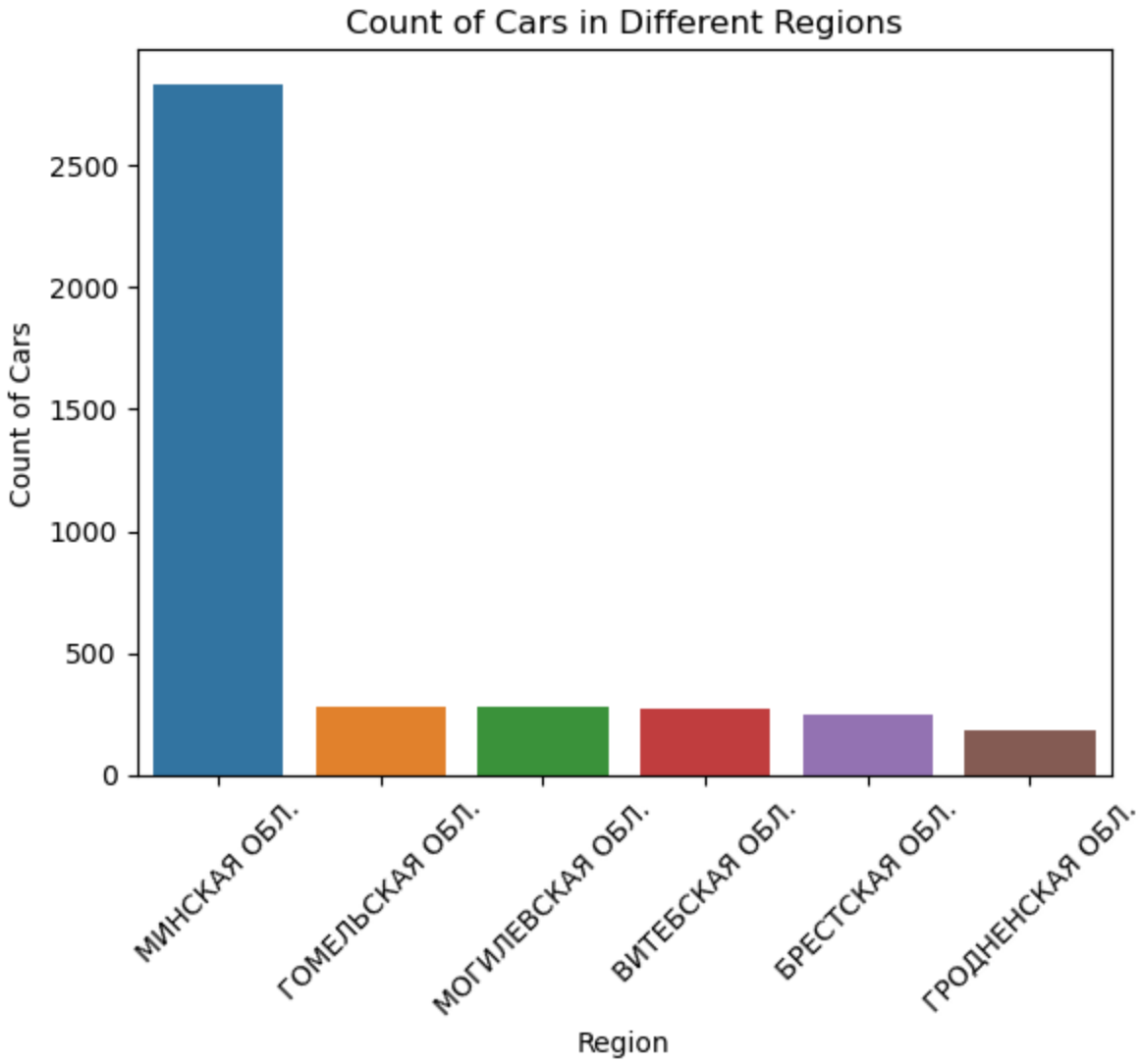
Out[18]:

	index	location_region
0	МИНСКАЯ ОБЛ.	2836
1	ГОМЕЛЬСКАЯ ОБЛ.	283
2	МОГИЛЕВСКАЯ ОБЛ.	282
3	ВИТЕБСКАЯ ОБЛ.	268
4	БРЕСТСКАЯ ОБЛ.	245
5	ГРОДНЕНСКАЯ ОБЛ.	185

In [19]:
```
sns.barplot(df2['index'], df2['location_region'])

# Add labels and a title
plt.xlabel('Region')
plt.ylabel('Count of Cars')
```

```
plt.title('Count of Cars in Different Regions')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```



Q3. Relationship b/w the price and miles driven with respect to manufacturers

In [20]:
```
#we will shortlist top 5 manufacturers
df.manufacturer_name.value_counts()
```

Out[20]:
```
Volkswagen       4243
Opel             2759
BMW              2610
Ford             2566
Renault          2493
Audi             2468
Mercedes-Benz    2237
Peugeot          1909
```

```
Citroen          1562
Nissan           1361
Mazda            1328
Toyota           1246
Hyundai          1116
Skoda            1089
Kia               912
Mitsubishi        887
Fiat              824
Honda             797
Volvo             721
ВАЗ               481
Chevrolet         436
Chrysler          410
Seat              303
Dodge             297
Subaru            291
Rover             235
Suzuki            234
Daewoo            221
Lexus             213
Alfa Romeo        207
ГАЗ               200
Land Rover        184
Infiniti          162
LADA              146
Iveco             139
Saab              108
Jeep              107
Lancia             92
SsangYong          79
УАЗ                74
Geely              71
Mini               68
Acura              66
Porsche            61
Dacia              59
Chery              58
Москвич            55
Jaguar             53
Buick              47
Lifan              47
Cadillac           43
Pontiac            42
ЗАЗ                42
Lincoln            36
Great Wall         36
Name: manufacturer_name, dtype: int64
```

In [21]:

```
#creating a list of the top manufacturers
top_man=['Volkswagen', 'Opel', 'BMW', 'Ford', 'Renault']

#applying filter to get results
df_top= df[df['manufacturer_name'].isin(top_man)]

df_top
```

Out[21]:

	manufacturer_name	model_name	transmission	color	odometer_value	year_produced	engine_fuel	engi
1720	Opel	Corsa	mechanical	red	132000	2014	diesel	
1721	Opel	Insignia	mechanical	other	200100	2008	gasoline	
1722	Opel	Omega	mechanical	brown	320000	2000	gasoline	
1723	Opel	Zafira	mechanical	blue	285000	2001	diesel	

1724	Opel	Vectra	mechanical	violet	298000	1997	gasoline
...	...	...	...	...	...	...	...
30105	BMW	X6	automatic	red	88900	2009	gasoline
30106	BMW	530	automatic	silver	240000	2001	gasoline
30107	BMW	520	mechanical	black	320000	1991	gasoline
30108	BMW	520	mechanical	silver	1000000	1982	gasoline
30109	BMW	318	mechanical	black	500000	1997	gasoline

14671 rows × 41 columns

In [22]:
```
#data cleaning

df_top.dropna(subset=['manufacturer_name','odometer_value', 'price_usd','year_produced'])

#no null values as the row count is unaffected
```

Out[22]:

	manufacturer_name	model_name	transmission	color	odometer_value	year_produced	engine_fuel	engi
1720	Opel	Corsa	mechanical	red	132000	2014	diesel	
1721	Opel	Insignia	mechanical	other	200100	2008	gasoline	
1722	Opel	Omega	mechanical	brown	320000	2000	gasoline	
1723	Opel	Zafira	mechanical	blue	285000	2001	diesel	
1724	Opel	Vectra	mechanical	violet	298000	1997	gasoline	
...	...	...	...	...	...	...	...	
30105	BMW	X6	automatic	red	88900	2009	gasoline	
30106	BMW	530	automatic	silver	240000	2001	gasoline	
30107	BMW	520	mechanical	black	320000	1991	gasoline	
30108	BMW	520	mechanical	silver	1000000	1982	gasoline	
30109	BMW	318	mechanical	black	500000	1997	gasoline	

14671 rows × 41 columns

In [23]:
```
#Filtering out incorrect values from year_produced lower limit 1950 and upper 2023

year_filt= (df_top['year_produced']>=1950) & (df_top['year_produced']<2023)

df_year=df_top[year_filt]

df_year
```

Out[23]:

	manufacturer_name	model_name	transmission	color	odometer_value	year_produced	engine_fuel	engi
1720	Opel	Corsa	mechanical	red	132000	2014	diesel	
1721	Opel	Insignia	mechanical	other	200100	2008	gasoline	
1722	Opel	Omega	mechanical	brown	320000	2000	gasoline	
1723	Opel	Zafira	mechanical	blue	285000	2001	diesel	
1724	Opel	Vectra	mechanical	violet	298000	1997	gasoline	

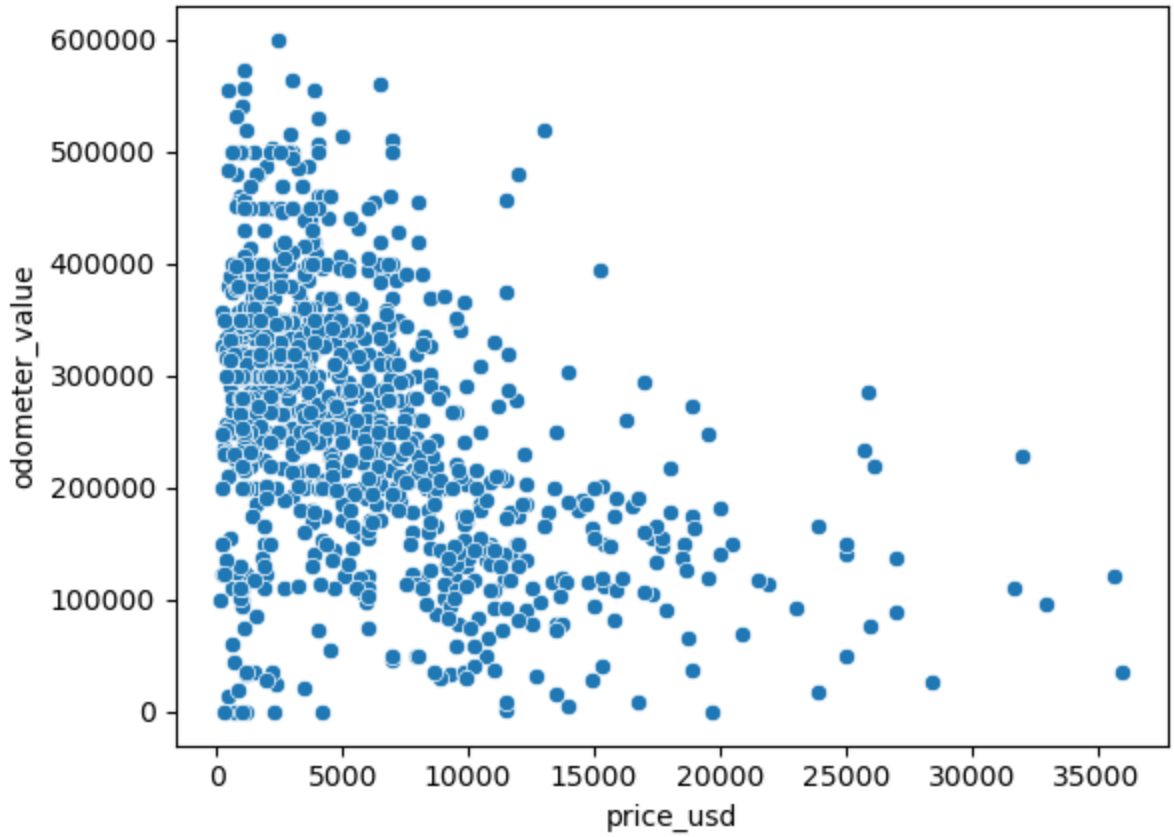
...	...	...	...	...	...	...	...
30105	BMW	X6	automatic	red	88900	2009	gasoline
30106	BMW	530	automatic	silver	240000	2001	gasoline
30107	BMW	520	mechanical	black	320000	1991	gasoline
30108	BMW	520	mechanical	silver	1000000	1982	gasoline
30109	BMW	318	mechanical	black	500000	1997	gasoline

14671 rows × 41 columns

In [24]:
```
#filtering out incorrect values
#filter for price lower= 10 upper= 1000000 and odo lower= 100 and upper=600000

price_odo_filt= (df_year['price_usd']>10) & (df_year['price_usd']<1000000) & (df_year['o

df_final=df_year[price_odo_filt]

#we have some values that were out of our range rows down from 14671 to 14406
```

In [25]:
```
df_sample=df_final.sample(1000)
#getting a sample of 1000 rows so that we get a better visualization
```

In [26]:
```
sns.scatterplot(x='price_usd', y='odometer_value', data=df_sample)

plt.show()
```



Q4. Relationship b/w price and miles top 3 models with most features?

In [27]:
```
df.Total_features.unique()
```

Out[27]:
```
array([7, 4, 3, 1, 5, 9, 6, 2, 8])
```

In [28]:
```
#making cut for the most amount of functions in the models
#bins 1-2 = low, 3-5=mid, 6-9=high
df['feature_category']=pd.cut(df['Total_features'], bins=[1,3,6,9], labels=['low', 'mid'
```

In [29]:
```
#shortlisting top 3 models
df.model_name.value_counts()
```

Out[29]:
```
Passat    1423
Astra      751
Golf       707
A6         687
Mondeo     637
          ...
C1500        1
Alero        1
Impala       1
360          1
Aspen        1
Name: model_name, Length: 1118, dtype: int64
```

In [30]:
```
top_models=['Passat','Astra','Golf']

model_filt= (df['model_name'].isin(top_models)) & (df['feature_category']=='high')

df_3=df[model_filt]

df_3
#this dataframe has top models with the best features
```

Out[30]:

	manufacturer_name	model_name	transmission	color	odometer_value	year_produced	engine_fuel	engin
1735	Opel	Astra	automatic	silver	102455	2012	gasoline	
1799	Opel	Astra	mechanical	black	215000	2010	gasoline	
1850	Opel	Astra	automatic	silver	165000	2005	gasoline	
1928	Opel	Astra	automatic	black	170000	2010	gasoline	
1932	Opel	Astra	mechanical	grey	200930	2011	diesel	
...	...	...	...	...	...	...	...	
19247	Volkswagen	Passat	automatic	blue	180000	2006	gasoline	
19253	Volkswagen	Passat	mechanical	black	185000	2010	diesel	
19254	Volkswagen	Passat	automatic	grey	270000	2007	diesel	
19284	Volkswagen	Golf	mechanical	silver	205000	2009	gasoline	
19292	Volkswagen	Passat	automatic	silver	210000	2007	gasoline	

253 rows × 42 columns

In [31]:
```
#removing Incorrect values from data
#applying limits to miles, price and year_produced

df_3=df_3[df_3['year_produced']>=1953]
df_3=df_3[df_3.year_produced<2023]

df_3=df_3[df_3.price_usd>=1000]
```

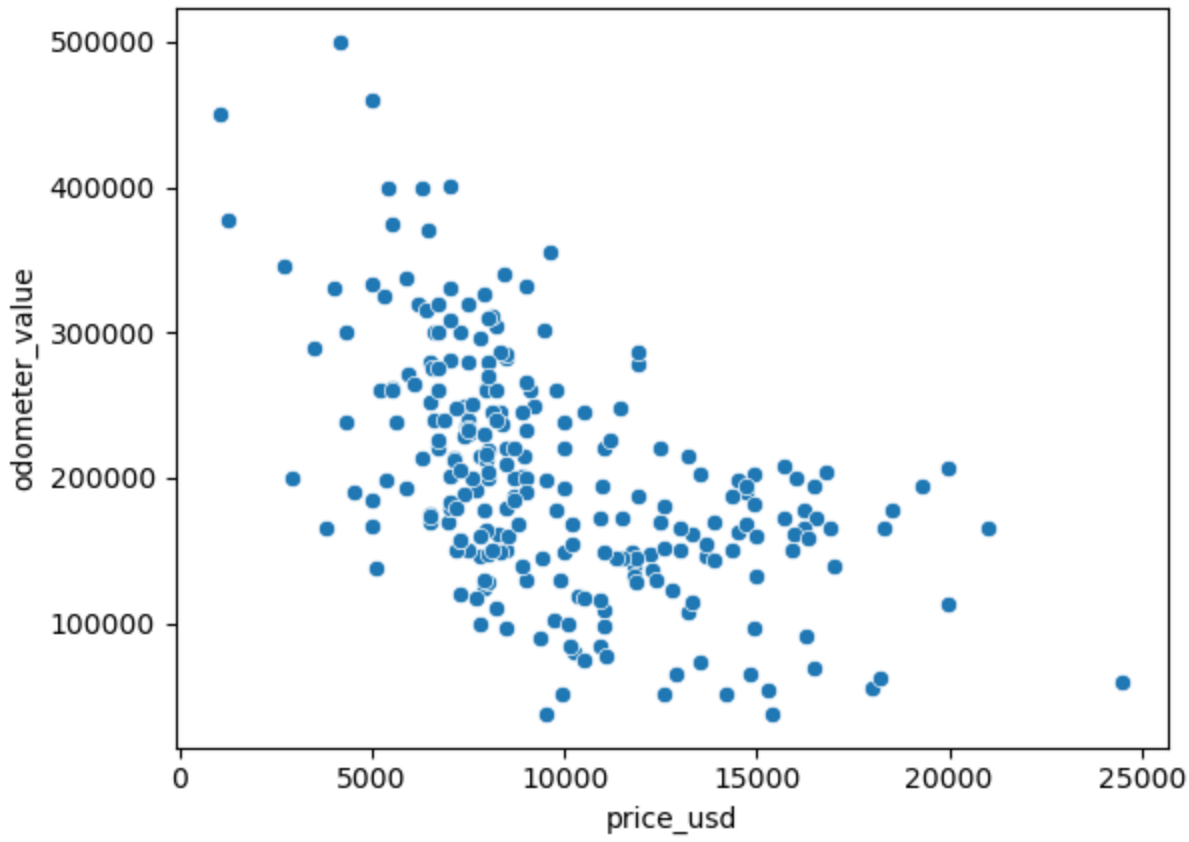
```
df_3=df_3[df_3.price_usd<1000000]

df_3=df_3[df_3.odometer_value>=1000]
df_3=df_3[df_3.odometer_value<6000000]

df_3
```

Out[31]:

	manufacturer_name	model_name	transmission	color	odometer_value	year_produced	engine_fuel	engin
1735	Opel	Astra	automatic	silver	102455	2012	gasoline	
1799	Opel	Astra	mechanical	black	215000	2010	gasoline	
1850	Opel	Astra	automatic	silver	165000	2005	gasoline	
1928	Opel	Astra	automatic	black	170000	2010	gasoline	
1932	Opel	Astra	mechanical	grey	200930	2011	diesel	
...	...	...	...	...	...	...	...	
19247	Volkswagen	Passat	automatic	blue	180000	2006	gasoline	
19253	Volkswagen	Passat	mechanical	black	185000	2010	diesel	
19254	Volkswagen	Passat	automatic	grey	270000	2007	diesel	
19284	Volkswagen	Golf	mechanical	silver	205000	2009	gasoline	
19292	Volkswagen	Passat	automatic	silver	210000	2007	gasoline	

252 rows × 42 columns

In [32]:
```
sns.scatterplot(x='price_usd', y='odometer_value', data=df_3)

plt.show()
```



Q5: Relationship between the kind of fuel and the number of different types of cars that run on it

In [33]:
```
df.engine_fuel.unique()
```

Out[33]:
```
array(['gasoline', 'gas', 'diesel', 'hybrid-petrol', 'hybrid-diesel',
       'electric'], dtype=object)
```

In [34]:
```
df.engine_fuel.value_counts()
```

Out[34]:
```
gasoline         24065
diesel           12872
gas               1347
hybrid-petrol      235
electric            10
hybrid-diesel        2
Name: engine_fuel, dtype: int64
```

In [35]:
```
df.body_type.unique()
```

Out[35]:
```
array(['universal', 'suv', 'sedan', 'hatchback', 'liftback', 'minivan',
       'minibus', 'van', 'pickup', 'coupe', 'cabriolet', 'limousine'],
      dtype=object)
```

In [36]:
```
df.body_type.value_counts()
```

Out[36]:
```
sedan        13011
hatchback     7644
universal     5507
suv           5164
minivan       3608
minibus       1369
van            808
coupe          652
liftback       552
pickup         129
cabriolet       75
limousine       12
Name: body_type, dtype: int64
```

In [37]:
```
#creating a one hot dataframe on the fuel column

one_hot=pd.get_dummies(df.engine_fuel)
one_hot['type']=df.body_type
one_hot
```

Out[37]:

	diesel	electric	gas	gasoline	hybrid-diesel	hybrid-petrol	type
0	0	0	0	1	0	0	universal
1	0	0	0	1	0	0	universal
2	0	0	0	1	0	0	suv
3	0	0	0	1	0	0	sedan
4	0	0	0	1	0	0	universal
...	...	...	...	...	...	...	...
38526	0	0	0	1	0	0	sedan
38527	1	0	0	0	0	0	hatchback
38528	0	0	0	1	0	0	sedan
38529	0	0	0	1	0	0	minivan

38530	0	0	0	1	0	0	minivan

38531 rows × 7 columns

In [38]:
```
sum_df= one_hot.groupby('type').sum()

sum_df
```

Out[38]:

type	diesel	electric	gas	gasoline	hybrid-diesel	hybrid-petrol
cabriolet	4.0	0.0	0.0	71.0	0.0	0.0
coupe	30.0	0.0	9.0	605.0	1.0	7.0
hatchback	1571.0	8.0	125.0	5872.0	0.0	68.0
liftback	96.0	2.0	14.0	428.0	0.0	12.0
limousine	0.0	0.0	1.0	11.0	0.0	0.0
minibus	1261.0	0.0	39.0	68.0	0.0	1.0
minivan	1956.0	0.0	268.0	1380.0	0.0	4.0
pickup	74.0	0.0	12.0	42.0	0.0	1.0
sedan	2553.0	0.0	447.0	9905.0	1.0	105.0
suv	1679.0	0.0	215.0	3246.0	0.0	24.0
universal	2937.0	0.0	204.0	2353.0	0.0	13.0
van	711.0	0.0	13.0	84.0	0.0	0.0

In [39]:
```
#we need to further clean the data to get better results
#filter out the car types with low car count
#limousine, cabriolet and pickup taken out of dataset

high_count=['coupe', 'hatchback', 'liftback', 'minibus', 'minivan', 'sedan', 'suv', 'uni

filt= one_hot['type'].isin(high_count)

fit=one_hot[filt]

fit.type.unique()
```

Out[39]:
```
array(['universal', 'suv', 'sedan', 'hatchback', 'liftback', 'minivan',
       'minibus', 'van', 'coupe'], dtype=object)
```

In [40]:
```
fit_sum= fit.groupby('type')[['diesel', 'gas', 'gasoline', 'hybrid-petrol','electric', ]

fit_sum.reset_index(inplace=True)

fit_sum
```

Out[40]:

	type	diesel	gas	gasoline	hybrid-petrol	electric
0	coupe	30.0	9.0	605.0	7.0	0.0
1	hatchback	1571.0	125.0	5872.0	68.0	8.0
2	liftback	96.0	14.0	428.0	12.0	2.0
3	minibus	1261.0	39.0	68.0	1.0	0.0

4	minivan	1956.0	268.0	1380.0	4.0	0.0
5	sedan	2553.0	447.0	9905.0	105.0	0.0
6	suv	1679.0	215.0	3246.0	24.0	0.0
7	universal	2937.0	204.0	2353.0	13.0	0.0
8	van	711.0	13.0	84.0	0.0	0.0

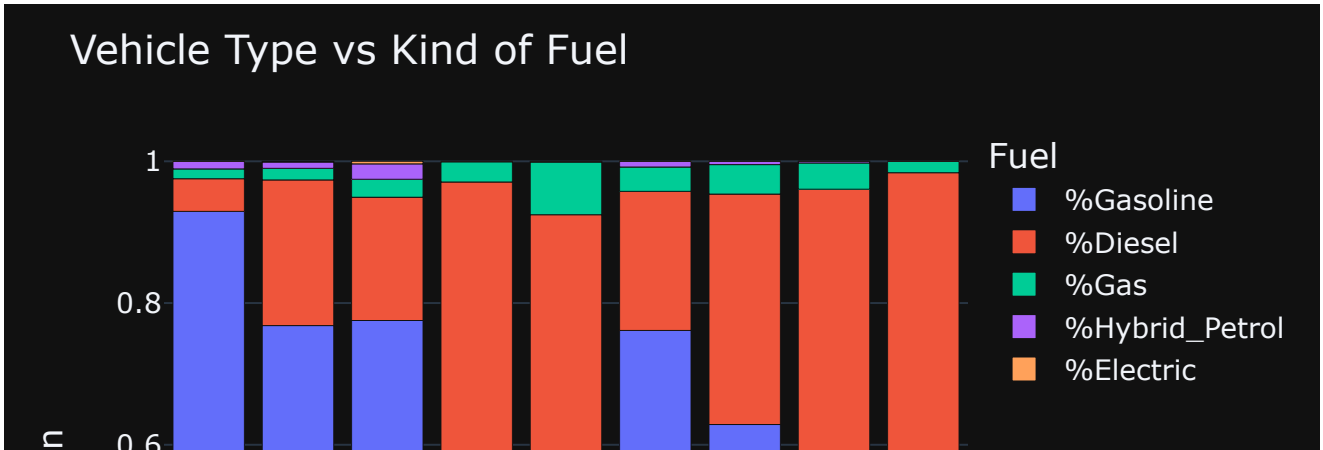
In [41]:
```
#now lets create a percentage dataframe for data

df_pct= pd.DataFrame()
df_pct['type']=fit_sum['type']
df_pct['%Gasoline']= fit_sum['gasoline']/(fit_sum['diesel']+fit_sum['gas']+fit_sum['gaso
df_pct['%Diesel']=fit_sum['diesel']/(fit_sum['diesel']+fit_sum['gas']+fit_sum['gasoline'
df_pct['%Gas']=fit_sum['gas']/(fit_sum['diesel']+fit_sum['gas']+fit_sum['gasoline']+fit_
df_pct['%Hybrid_Petrol']=fit_sum['hybrid-petrol']/(fit_sum['diesel']+fit_sum['gas']+fit_
df_pct['%Electric']=fit_sum['electric']/(fit_sum['diesel']+fit_sum['gas']+fit_sum['gasol
```

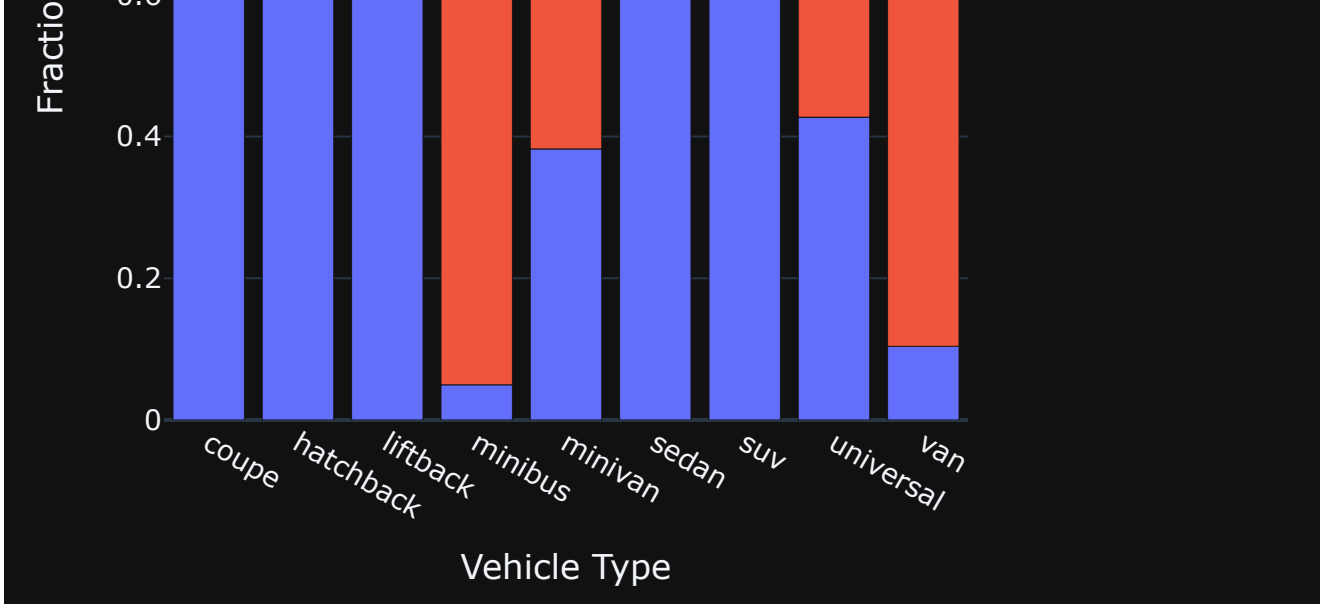
In [42]:
```
df_pct
```

Out[42]:

	type	%Gasoline	%Diesel	%Gas	%Hybrid_Petrol	%Electric
0	coupe	0.929339	0.046083	0.013825	0.010753	0.000000
1	hatchback	0.768184	0.205521	0.016353	0.008896	0.001047
2	liftback	0.775362	0.173913	0.025362	0.021739	0.003623
3	minibus	0.049671	0.921110	0.028488	0.000730	0.000000
4	minivan	0.382483	0.542129	0.074279	0.001109	0.000000
5	sedan	0.761337	0.196234	0.034358	0.008071	0.000000
6	suv	0.628582	0.325136	0.041634	0.004648	0.000000
7	universal	0.427274	0.533321	0.037044	0.002361	0.000000
8	van	0.103960	0.879950	0.016089	0.000000	0.000000

In [43]:
```
import plotly.express as px

stacked_bar = px.bar(df_pct, x="type", y=["%Gasoline","%Diesel","%Gas", "%Hybrid_Petrol"
stacked_bar.update_layout(xaxis_title='Vehicle Type',
            yaxis_title='Fraction',
            font_size=14,
            title="Vehicle Type vs Kind of Fuel"
            )
stacked_bar.show()
```





Q6. Comparison of the type of vehicles listed of the various manufacturees?

In [44]:
```
df.columns
```

Out[44]:
```
Index(['manufacturer_name', 'model_name', 'transmission', 'color',
       'odometer_value', 'year_produced', 'engine_fuel', 'engine_has_gas',
       'engine_type', 'engine_capacity', 'body_type', 'has_warranty', 'state',
       'drivetrain', 'price_usd', 'is_exchangeable', 'location_region',
       'number_of_photos', 'up_counter', 'feature_0', 'feature_1', 'feature_2',
       'feature_3', 'feature_4', 'feature_5', 'feature_6', 'feature_7',
       'feature_8', 'feature_9', 'duration_listed', 'Feature_count_0',
       'Feature_count_1', 'Feature_count_2', 'Feature_count_3',
       'Feature_count_4', 'Feature_count_5', 'Feature_count_6',
       'Feature_count_7', 'Feature_count_8', 'Feature_count_9',
       'Total_features', 'feature_category'],
      dtype='object')
```

In [45]:
```
df_tree=df[['manufacturer_name', 'body_type']]

df_tree
```

Out[45]:

	manufacturer_name	body_type
0	Subaru	universal
1	Subaru	universal
2	Subaru	suv
3	Subaru	sedan
4	Subaru	universal
...	...	...
38526	Chrysler	sedan
38527	Chrysler	hatchback
38528	Chrysler	sedan
38529	Chrysler	minivan
38530	Chrysler	minivan

38531 rows × 2 columns

In [46]:
```
count= pd.get_dummies(df_tree['body_type'])
count['manufacturer']=df_tree['manufacturer_name']
count['type']=df_tree['body_type']

count
```

Out[46]:

	cabriolet	coupe	hatchback	liftback	limousine	minibus	minivan	pickup	sedan	suv	universal	van
0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
38526	0	0	0	0	0	0	0	0	1	0	0	0
38527	0	0	1	0	0	0	0	0	0	0	0	0
38528	0	0	0	0	0	0	0	0	1	0	0	0
38529	0	0	0	0	0	0	1	0	0	0	0	0
38530	0	0	0	0	0	0	1	0	0	0	0	0

38531 rows × 14 columns

In [47]:
```
treemap= df_tree.groupby(['manufacturer_name', 'body_type']).value_counts()
#treemap.reset_index(inplace=True)

tf= pd.DataFrame(treemap)
tf.reset_index(inplace=True)
tf
```

Out[47]:

	manufacturer_name	body_type	0
0	Acura	coupe	3
1	Acura	hatchback	2
2	Acura	liftback	1
3	Acura	sedan	28
4	Acura	suv	32
...	...	...	...
359	Москвич	universal	2
360	УАЗ	minibus	7
361	УАЗ	pickup	6
362	УАЗ	suv	54
363	УАЗ	van	7

364 rows × 3 columns

In [48]:
```
fig = px.treemap(tf, path=[px.Constant("Manufacturer"), 'manufacturer_name', 'body_type'
                  color_continuous_scale='RdBu')
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```