CARS24_USEDCARS_EDA_PROJECT_VIVEK_CHAUHAN

This Analysis Is Devided Into Four Major Parts:

1. Data Understanding

2. Data Cleaning If Needed

3. Data Analysis

4. Recomendations

In [1]:
```
# upload the necessary libraries for work on the data

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# show 100 rows and 100 columns of our dataset

pd.options.display.max_rows = 100
pd.options.display.max_columns = 100
```

In [3]:
```
# load the dataset

data = pd.read_csv("cars24data..csv")
data
```

Out[3]:

	Model Year	Brand Name	Model Name	Engine capacity	Spare key	Transmission	KM driven	Ownership	Fuel type	Imperfections	Repainted Parts
0	2017	Maruti	Swift VXI	1197	No	Manual	25847	2	Petrol	6	2
1	2016	Maruti	Baleno DELTA PETROL 1.2	1197	Yes	Manual	55511	2	Petrol	12	1
2	2020	Maruti	Swift VXI	1197	No	Manual	47110	1	Petrol	4	2
3	2022	Maruti	Ertiga VXI AT SHVS	1462	Yes	Automatic	35378	1	Petrol	2	3
4	2019	Maruti	Dzire VXI	1197	Yes	Manual	91856	1	Petrol	3	2
...	...	...	...	...	...	...	...	...	...	...	...
1440	2021	Maruti	Ertiga VXI SHVS	1462	No	Manual	19901	1	Petrol	1	0
1441	2015	Maruti	Ciaz ZXI	1373	No	Manual	50022	1	Petrol	5	2
1442	2019	Maruti	Baleno DELTA PETROL 1.2	1197	Yes	Manual	58679	1	Petrol	24	4
1443	2017	Maruti	Ciaz S 1.4 MT PETROL	1373	Yes	Manual	73948	2	Petrol	4	5
1444	2012	Maruti	Wagon R 1.0 LXI	998	No	Manual	55994	1	Petrol	20	9

1445 rows × 11 columns

Data Understanding

In [4]:
```
# first 10 rows of the dataset

data.head(10)
```

Out[4]:

	Model Year	Brand Name	Model Name	Engine capacity	Spare key	Transmission	KM driven	Ownership	Fuel type	Imperfections	Repainted Parts
0	2017	Maruti	Swift VXI	1197	No	Manual	25847	2	Petrol	6	2
1	2016	Maruti	Baleno DELTA PETROL 1.2	1197	Yes	Manual	55511	2	Petrol	12	1
2	2020	Maruti	Swift VXI	1197	No	Manual	47110	1	Petrol	4	2
3	2022	Maruti	Ertiga VXI AT SHVS	1462	Yes	Automatic	35378	1	Petrol	2	3
4	2019	Maruti	Dzire VXI	1197	Yes	Manual	91856	1	Petrol	3	2
5	2014	Maruti	Alto 800 LXI	796	No	Manual	43780	1	Petrol	10	2
6	2020	Maruti	Swift VXI	1197	Yes	Manual	49583	1	Petrol	1	0
7	2018	Maruti	Dzire VXI AMT	1197	No	Automatic	86837	2	Petrol	4	6
8	2016	Maruti	Swift Dzire VXI	1197	Yes	Manual	58570	2	Petrol	10	3
9	2019	Maruti	S PRESSO VXI	998	Yes	Manual	50645	1	Petrol	0	0

In [5]:
```
# last 10 rows of the dataset

data.tail(10)
```

Out[5]:

	Model Year	Brand Name	Model Name	Engine capacity	Spare key	Transmission	KM driven	Ownership	Fuel type	Imperfections	Repainted Parts
1435	2018	Maruti	Ciaz ZETA 1.4 AT PETROL	1373	Yes	Automatic	64686	1	Petrol	14	1
1436	2017	Maruti	Baleno ALPHA PETROL 1.2	1197	Yes	Manual	75059	1	Petrol	5	6
1437	2020	Maruti	Baleno ZETA PETROL 1.2	1197	No	Manual	58346	1	Petrol	1	0
1438	2016	Maruti	Ciaz ZDI SHVS	1248	Yes	Manual	79630	1	Diesel	8	7
1439	2021	Maruti	Swift VXI	1197	No	Manual	44299	1	Petrol	1	0
1440	2021	Maruti	Ertiga VXI SHVS	1462	No	Manual	19901	1	Petrol	1	0
1441	2015	Maruti	Ciaz ZXI	1373	No	Manual	50022	1	Petrol	5	2
1442	2019	Maruti	Baleno DELTA PETROL 1.2	1197	Yes	Manual	58679	1	Petrol	24	4
1443	2017	Maruti	Ciaz S 1.4 MT PETROL	1373	Yes	Manual	73948	2	Petrol	4	5
1444	2012	Maruti	Wagon R 1.0 LXI	998	No	Manual	55994	1	Petrol	20	9

In [6]:
```
# print the shape of our dataset

data.shape
```

Out[6]:
```
(1445, 11)
```

In [7]:
```
# print the overall statistics of the dataset

data.describe(include = "all")
```

Out[7]:

	Model Year	Brand Name	Model Name	Engine capacity	Spare key	Transmission	KM driven	Ownership	Fuel type	Imperfections	Repainted Parts
count	1445.000000	1445	1445	1445.000000	1445	1445	1445.000000	1445.000000	1445	1445.000000	1445.000000
unique	NaN	1	212	NaN	2	2	NaN	NaN	3	NaN	NaN
top	NaN	Maruti	Swift VXI	NaN	Yes	Manual	NaN	NaN	Petrol	NaN	NaN
freq	NaN	1445	81	NaN	908	1073	NaN	NaN	1264	NaN	NaN
mean	2017.817301	NaN	NaN	1142.104498	NaN	NaN	50588.903114	1.285121	NaN	9.597232	3.228374
std	2.986554	NaN	NaN	169.020818	NaN	NaN	27339.562631	0.489877	NaN	8.398637	3.364578
min	2010.000000	NaN	NaN	796.000000	NaN	NaN	1207.000000	1.000000	NaN	0.000000	0.000000
25%	2016.000000	NaN	NaN	998.000000	NaN	NaN	28803.000000	1.000000	NaN	3.000000	0.000000
50%	2018.000000	NaN	NaN	1197.000000	NaN	NaN	47849.000000	1.000000	NaN	8.000000	2.000000
75%	2020.000000	NaN	NaN	1197.000000	NaN	NaN	70337.000000	2.000000	NaN	14.000000	5.000000
max	2023.000000	NaN	NaN	1462.000000	NaN	NaN	124716.000000	3.000000	NaN	43.000000	27.000000

In [8]:
```
# check the datatype of our dataset

data.dtypes
```

Out[8]:
```
Model Year          int64
Brand Name         object
Model Name         object
Engine capacity     int64
Spare key          object
Transmission       object
KM driven           int64
Ownership           int64
Fuel type          object
Imperfections       int64
Repainted Parts     int64
dtype: object
```

Data Cleaning

In [9]:
```
# print all the column names which is present in the dataset

data.columns
```

Out[9]:
```
Index(['Model Year', 'Brand Name', 'Model Name', 'Engine capacity',
       'Spare key', 'Transmission', 'KM driven', 'Ownership', 'Fuel type',
       'Imperfections', 'Repainted Parts'],
      dtype='object')
```

In [10]:
```
# check is there any null values is present in the dataset column wise

missing_info = pd.DataFrame(data.isnull().sum().sort_values(ascending = False))
missing_info
```

Out[10]:

	0
Model Year	0
Brand Name	0
Model Name	0
Engine capacity	0
Spare key	0
Transmission	0
KM driven	0
Ownership	0
Fuel type	0
Imperfections	0
Repainted Parts	0

Data Analysis

Uni-Variate-Analysis

In [11]:
```
# let's categorise the data of imperfections

def impns(n):
    if(n<=5):
        return "low_imperfections"
    elif(n>=6 & n<=10):
        return "medium_imperfections"
    elif(n>=11 & n<=15):
        return "high_imperfections"
    else:
```

```
        return "very_high_imperfections"

data["Imperfections_cat"] = data["Imperfections"].apply(lambda x : impns(x))
```

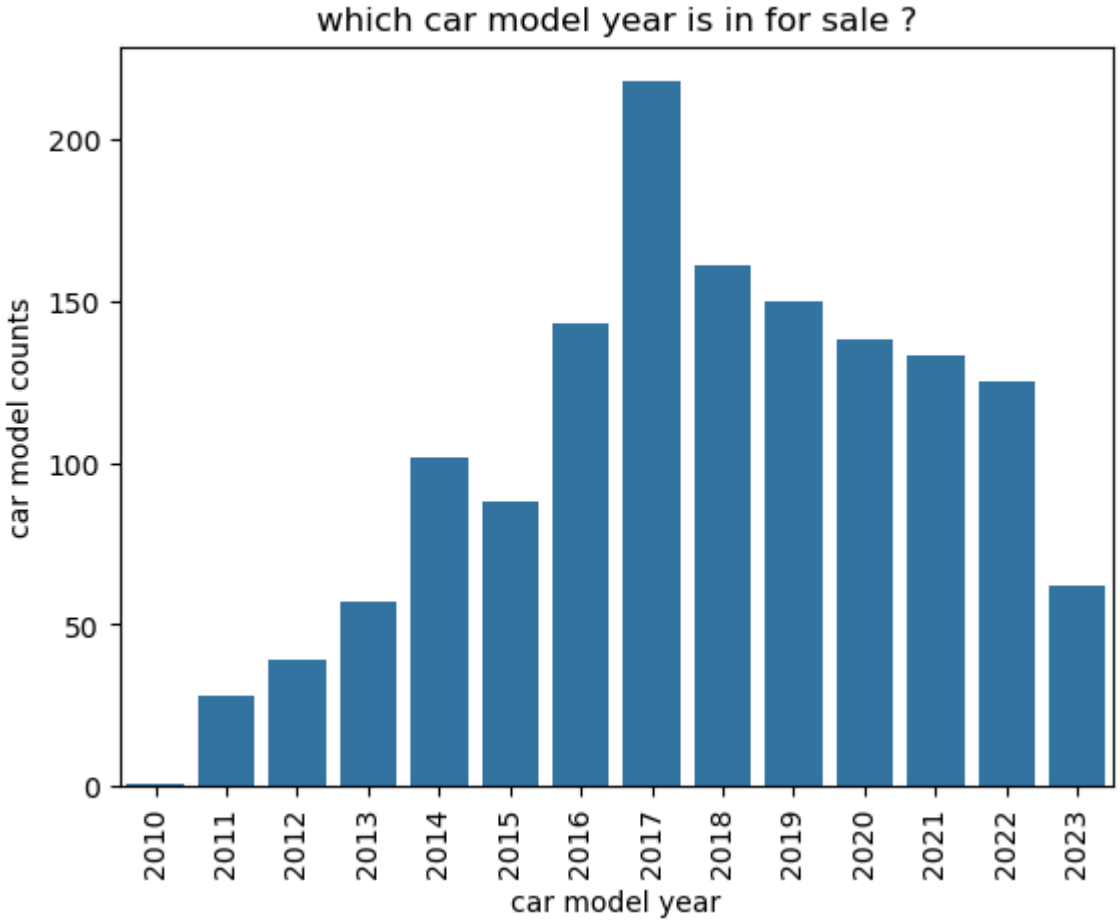
In [12]:
```
# check the datatype of the Imperfections column after the category

data["Imperfections"].dtypes
```

Out[12]:
```
dtype('int64')
```

In [13]:
```
# check the datatype of the Imperfections column after the category

data["Imperfections_cat"].dtypes
```

Out[13]:
```
dtype('O')
```

In [14]:
```
# which model year cars is in demands and people wants to purchase or like that model

sns.countplot(x = "Model Year",data = data)
plt.title("which car model year is in for sale ?")
plt.xlabel("car model year")
plt.ylabel("car model counts")
plt.xticks(rotation = "vertical")
plt.show()
```



You can see in the above chart 2017 year car models are very large amounts for sales in cars24 platform.

In [15]:
```
# check the unique values is present in brand name column in the dataset

data["Brand Name"].unique()
```

Out[15]:
```
array(['Maruti'], dtype=object)
```

In [16]:
```
# let's categorise the data of Repainted Parts

def impns(n):
    if(n<=5):
        return "low_Repainted_Parts"
    elif(n>=6 & n<=10):
        return "medium_Repainted_Parts"
    elif(n>=11 & n<=15):
        return "high_Repainted_Parts"
    else:
        return "very_Repainted_Parts"

data["Repainted Parts_cat"] = data["Repainted Parts"].apply(lambda x : impns(x))
```

In [17]:
```
# print the Repainted Parts column

data["Repainted Parts"]
```

Out[17]:
```
0       2
1       1
2       2
3       3
4       2
       ..
1440    0
1441    2
1442    4
1443    5
1444    9
Name: Repainted Parts, Length: 1445, dtype: int64
```

In [18]:
```
# now cross check the Repainted Parts datatypes

data["Repainted Parts"].dtypes
```

Out[18]:
```
dtype('int64')
```

In [19]:
```
# let's check car is in which paints conditions?
```

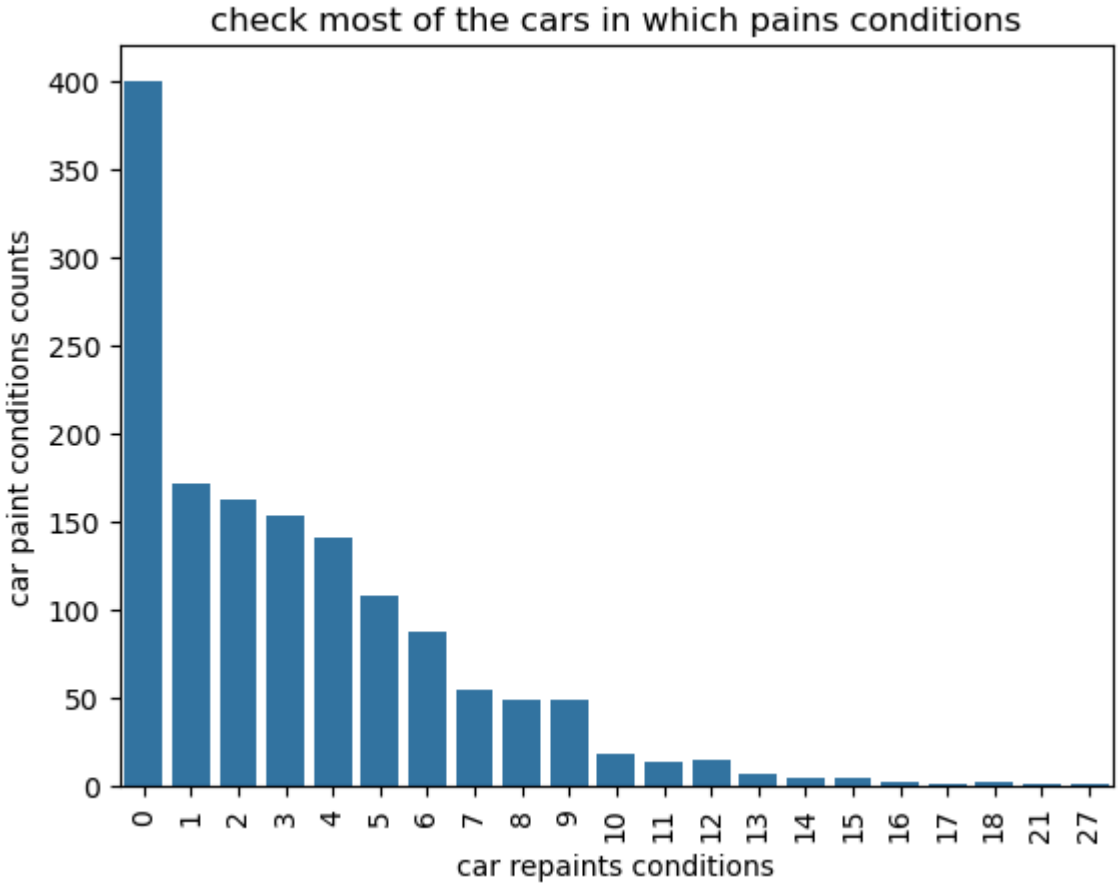
```
a = data["Repainted Parts"].mode()
a
```

Out[19]:
```
0    0
Name: Repainted Parts, dtype: int64
```

In [20]:
```
# print the Repainted Parts column

data["Repainted Parts_cat"]
```

Out[20]:
```
0          low_Repainted_Parts
1          low_Repainted_Parts
2          low_Repainted_Parts
3          low_Repainted_Parts
4          low_Repainted_Parts
                 ...
1440       low_Repainted_Parts
1441       low_Repainted_Parts
1442       low_Repainted_Parts
1443       low_Repainted_Parts
1444    medium_Repainted_Parts
Name: Repainted Parts_cat, Length: 1445, dtype: object
```

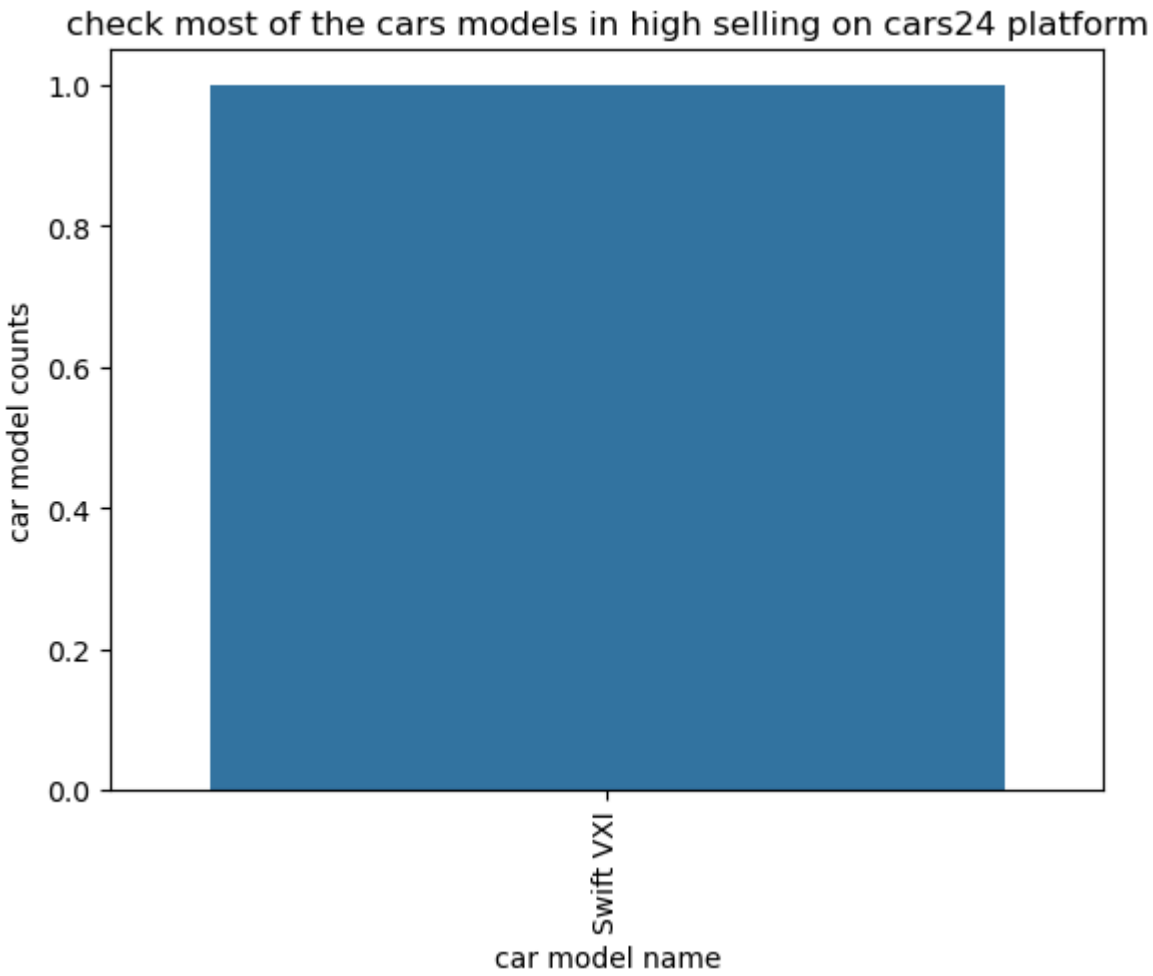
In [21]:
```
# let's visualize & check car is in which paints conditions?

sns.countplot(x = "Repainted Parts",data= data)
plt.title("check most of the cars in which pains conditions")
plt.xlabel("car repaints conditions")
plt.ylabel("car paint conditions counts")
plt.xticks(rotation = "vertical")
plt.show()
```



In above charts you can see the most of the used cars has low repainted parts conditions means most of the cars which is selling on cars24 platform has below 5 painted parts.

In [22]:
```
# let's check which car models Name is in highest selling on cars24 platform

sns.countplot(x = data["Model Name"].mode(),data= data)
plt.title("check most of the cars models in high selling on cars24 platform")
plt.xlabel("car model name")
plt.ylabel("car model counts")
```

```
plt.xticks(rotation = "vertical")
plt.show()
```

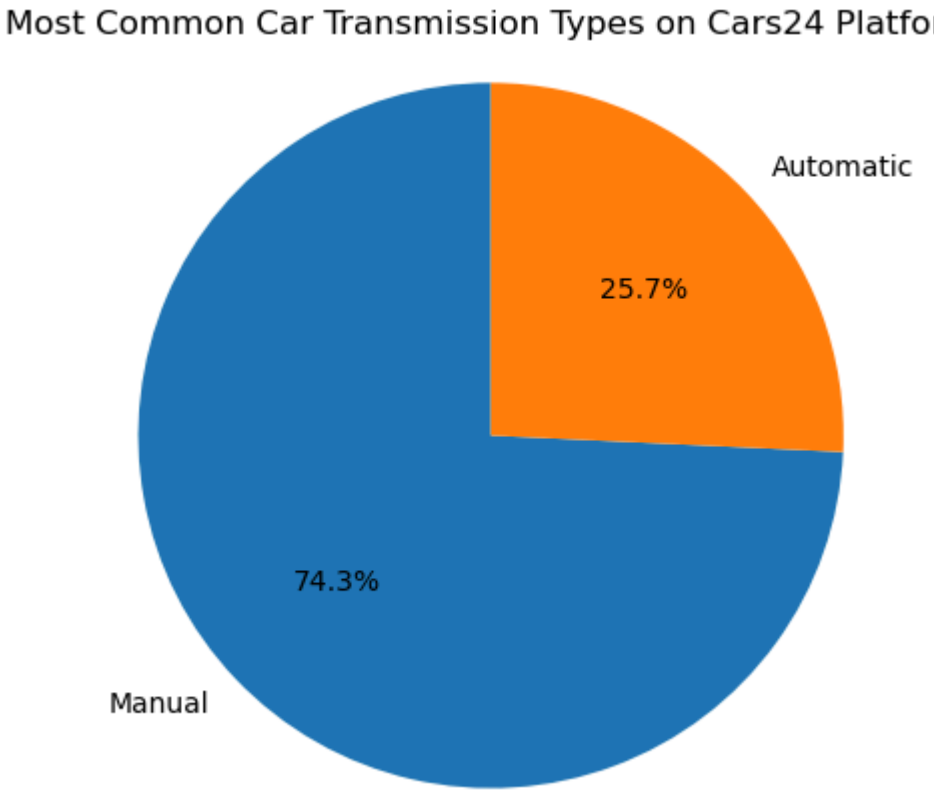


In above chart you can see the Swift VXI car model is highest selling on cars24 platform.

In [101…

```
# let's check which car transmission type is in highest selling on cars24 platform
```

```
# Count the transmission types
transmission_counts = data["Transmission"].value_counts()

# Plot pie chart
plt.figure(figsize=(5,5))
plt.pie(transmission_counts, labels=transmission_counts.index, autopct='%1.1f%%', startangle=90)
plt.title("Most Common Car Transmission Types on Cars24 Platform")
plt.axis('equal')  # Ensures pie is a circle
plt.show()
```

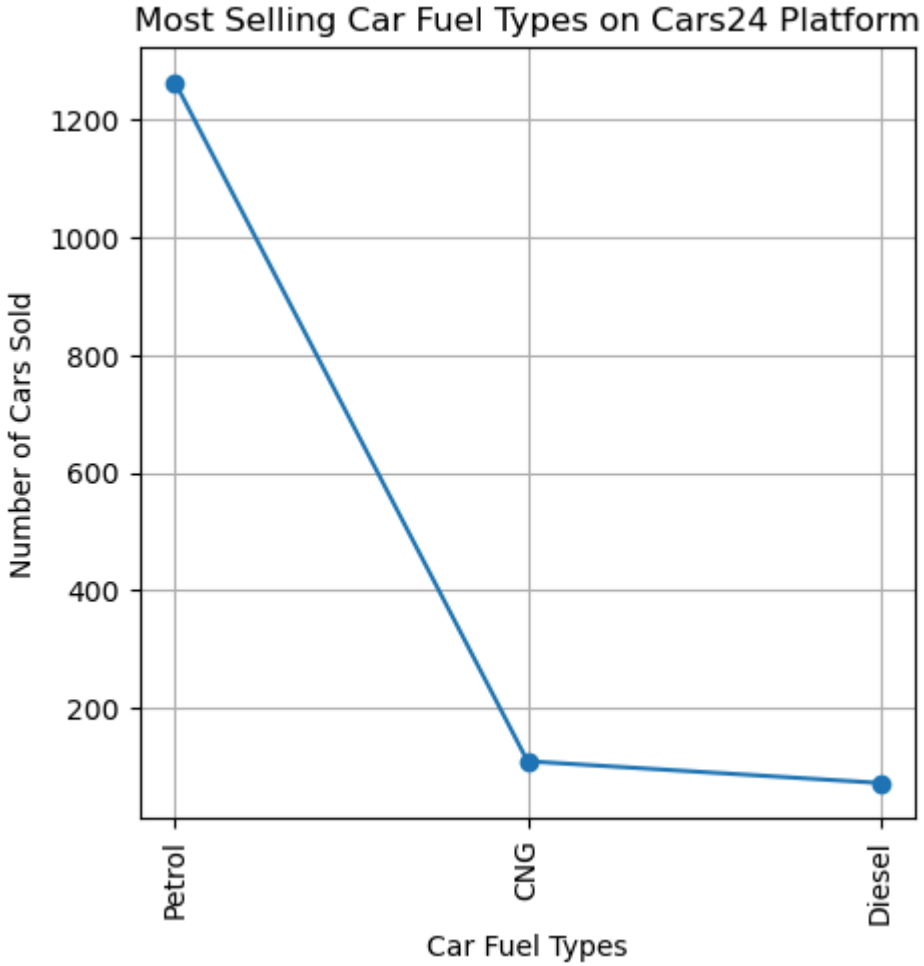


Manual Transmission type cars is very high for selling on cars24 platform.

In [24]:
```
# print all the column names

data.columns
```

Out[24]:
```
Index(['Model Year', 'Brand Name', 'Model Name', 'Engine capacity',
       'Spare key', 'Transmission', 'KM driven', 'Ownership', 'Fuel type',
       'Imperfections', 'Repainted Parts', 'Imperfections_cat',
       'Repainted Parts_cat'],
      dtype='object')
```

In [65]:
```
# let's see which fuel type cars is most selling on the cars24 platform

# Count the fuel types
fuel_counts = data['Fuel type'].value_counts()

# Plot line chart
plt.figure(figsize=(5,5))
plt.plot(fuel_counts.index, fuel_counts.values, marker='o')
plt.title("Most Selling Car Fuel Types on Cars24 Platform")
plt.xlabel("Car Fuel Types")
plt.ylabel("Number of Cars Sold")
plt.xticks(rotation="vertical")
plt.grid(True)
plt.show()
```



In above chart as you can see most of the petrol driven cars are high for selling on cars24 platform.

In [26]:
```
# let's check the datatype of Ownership column

data.Ownership.dtypes
```

Out[26]:
```
dtype('int64')
```

In [27]:
```
# let's convert it into categorical data for Ownership column

def ownership_cat(x):
    if(x==1):
        return "first_owner"
    elif(x==2):
        return "second_owner"
    elif(x==3):
        return "third_owner"
    else:
        return "fourth_and_above_owner"

data["Ownership_cat"] = data["Ownership"].apply(ownership_cat)
```

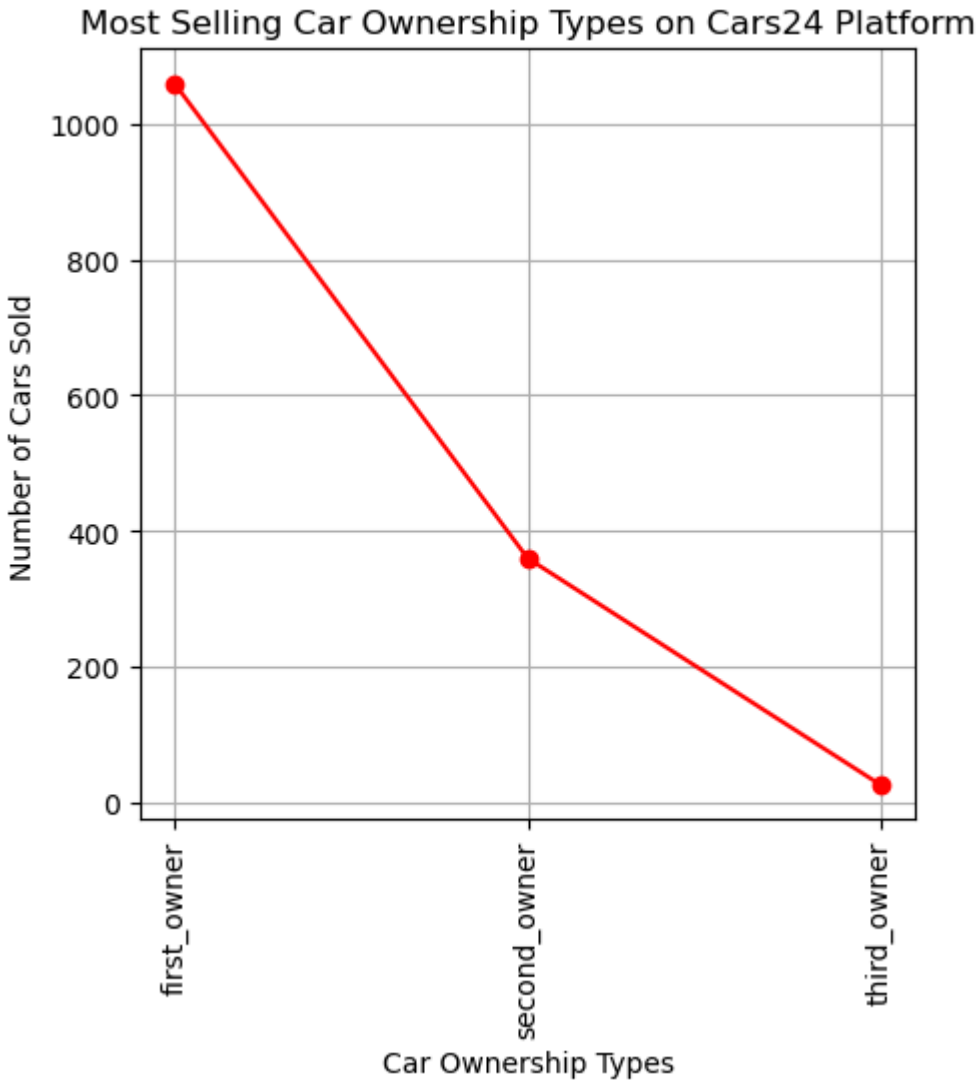
In [28]:
```
# let's print the ownership column

data["Ownership"]
```

Out[28]:
```
0       2
1       2
2       1
3       1
4       1
       ..
1440    1
1441    1
1442    1
1443    2
1444    1
Name: Ownership, Length: 1445, dtype: int64
```

In [29]:
```
# let's print the ownership column

data["Ownership_cat"]
```

Out[29]:
```
0       second_owner
1       second_owner
2        first_owner
3        first_owner
4        first_owner
           ...
1440     first_owner
1441     first_owner
1442     first_owner
1443    second_owner
1444     first_owner
Name: Ownership_cat, Length: 1445, dtype: object
```

In [72]:
```
# let's visualize the ownerships

# Count the ownership types
ownership_counts = data['Ownership_cat'].value_counts()

# Plot line chart
plt.figure(figsize=(5, 5))
plt.plot(ownership_counts.index, ownership_counts.values, marker='o', color="red")
plt.title("Most Selling Car Ownership Types on Cars24 Platform")
plt.xlabel("Car Ownership Types")
plt.ylabel("Number of Cars Sold")
plt.xticks(rotation="vertical")
plt.grid(True)
plt.show()
```



As you can see in above chart first_owner cars are majority for salling on the cars24 platform

In [31]:
```
# let's check the km driven datatypes
```

```
data['KM driven'].dtypes
```

Out[31]:
```
dtype('int64')
```

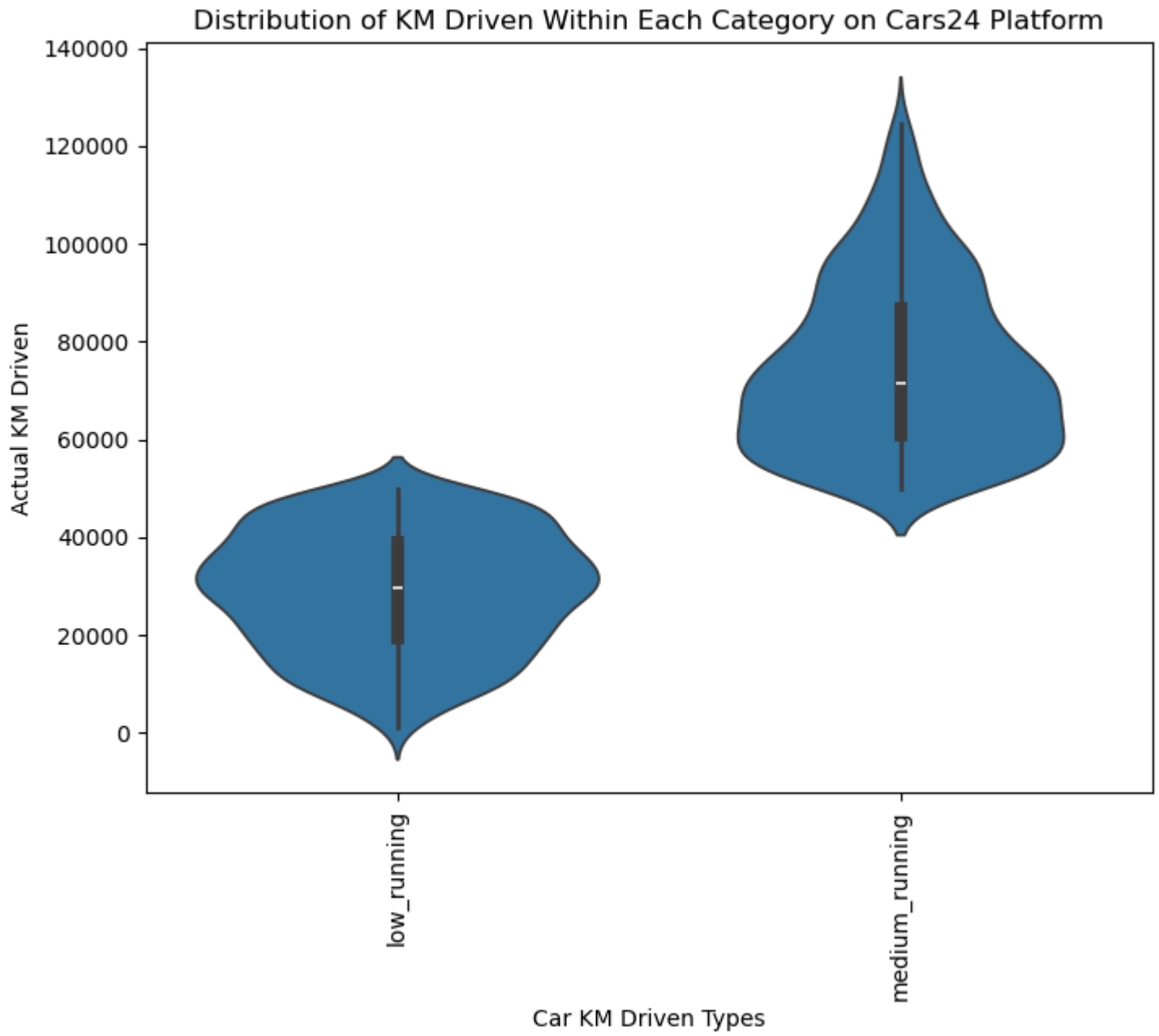
In [32]:
```
# make the category function for better understanding

def km_driven(x):
    if(x<=50000):
        return "low_running"
    elif(x>50000 & x<=100000):
        return "medium_running"
    else:
        return "veryhigh_running"


data['KM driven_cat'] = data['KM driven'].apply(km_driven)
```

In [33]:
```
# let's print the km_driven column

data['KM driven']
```

Out[33]:
```
0       25847
1       55511
2       47110
3       35378
4       91856
        ...
1440    19901
1441    50022
1442    58679
1443    73948
1444    55994
Name: KM driven, Length: 1445, dtype: int64
```

In [73]:
```
# let's visualize the 'KM driven' column for better understanding

# Violin plot
plt.figure(figsize=(8,6))
sns.violinplot(x=data['KM driven_cat'], y=data['KM driven'])
plt.title("Distribution of KM Driven Within Each Category on Cars24 Platform")
```

```
plt.xlabel("Car KM Driven Types")
plt.ylabel("Actual KM Driven")
plt.xticks(rotation="vertical")
plt.show()
```


Distribution of KM Driven Within Each Category on Cars24 Platform
Actual KM Driven
140000
120000
100000
80000
60000
40000
20000
0
low_running
medium_running
Car KM Driven Types

In the above chart low_running cars is highest for salling on cars24 platform.

In [35]:
```
# check the engine capacity datatype

data['Engine capacity'].dtypes
```

Out[35]:
```
dtype('int64')
```

In [36]:
```
# convert the engine capacity data type to categorical

def eng_cap(x):
    if(x<=1000):
        return  "low_cc_engine"
    elif(x>=1001 & x<=1500):
        return "medium_cc_engine"
    else:
        return "high_cc_engine"

data['Engine capacity_cat'] = data['Engine capacity'].apply(eng_cap)
```

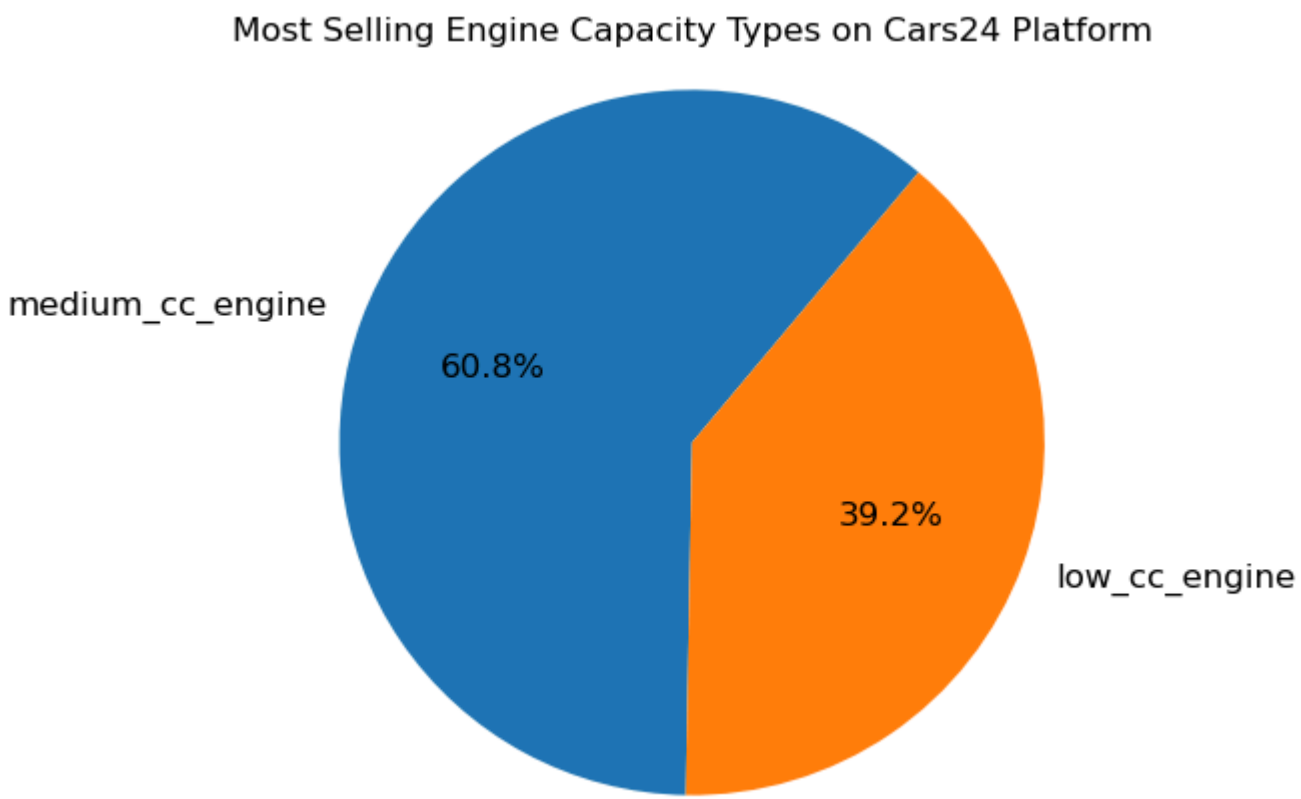
In [37]:
```
# print the engine capacity column

data['Engine capacity']
```

Out[37]:
```
0       1197
1       1197
2       1197
3       1462
4       1197
        ...
1440    1462
1441    1373
1442    1197
1443    1373
1444     998
Name: Engine capacity, Length: 1445, dtype: int64
```

In [38]:
```
# print the engine capacity column

data['Engine capacity_cat']
```

Out[38]:
```
0       medium_cc_engine
1       medium_cc_engine
2       medium_cc_engine
3       medium_cc_engine
4       medium_cc_engine
              ...
1440    medium_cc_engine
1441    medium_cc_engine
1442    medium_cc_engine
1443    medium_cc_engine
1444       low_cc_engine
Name: Engine capacity_cat, Length: 1445, dtype: object
```

In [78]:
```
# let's visualize which engine cc types is highes for sale on cars24 platform

# Count the engine capacity types
engine_capacity_counts = data['Engine capacity_cat'].value_counts()

# Plot pie chart
plt.figure(figsize=(5,5))
plt.pie(engine_capacity_counts, labels=engine_capacity_counts.index, autopct='%1.1f%%', startangle=50,textprops={'fontsize': 1
plt.title("Most Selling Engine Capacity Types on Cars24 Platform")
plt.axis('equal')  # Make it a circle
plt.show()
```



As you can see the medium_cc_engine is most of the salling on the cars24 platform.

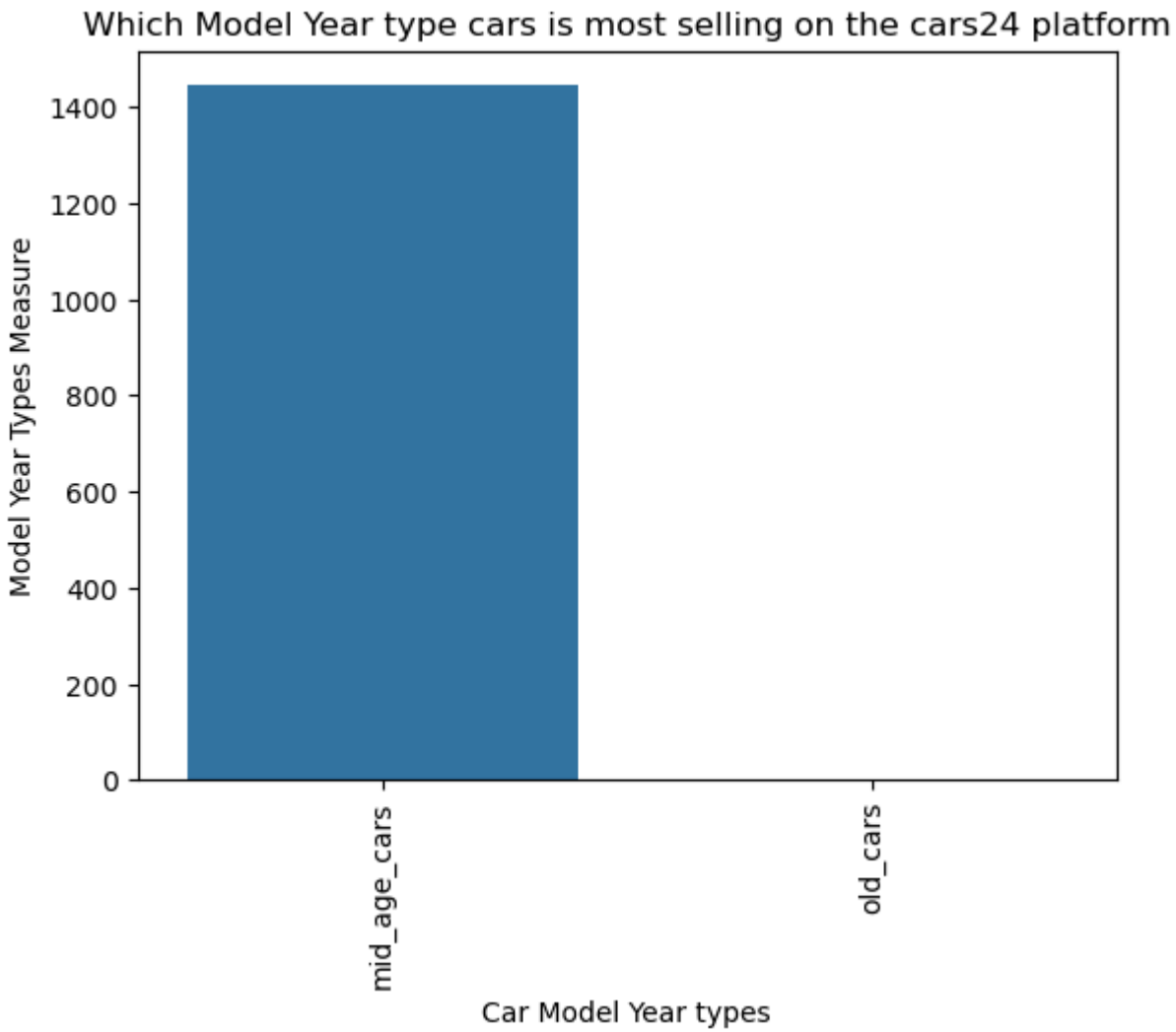
In [40]:
```
# check the datatype for the column of car model year

data["Model Year"].dtypes
```

Out[40]:
```
dtype('int64')
```

In [41]:
```
# categorize the Model Year column for better understanding

def model_cat(x):
```

```
    if(x<=2010):
        return "old_cars"
    elif(x>=2011 & x<=2015):
        return "mid_age_cars"
    elif(x>=2016 & x<=2020):
        return "fresh_age_cars"
    else:
        return "new_cars"

data["Model Year_cat"] = data["Model Year"].apply(model_cat)
```

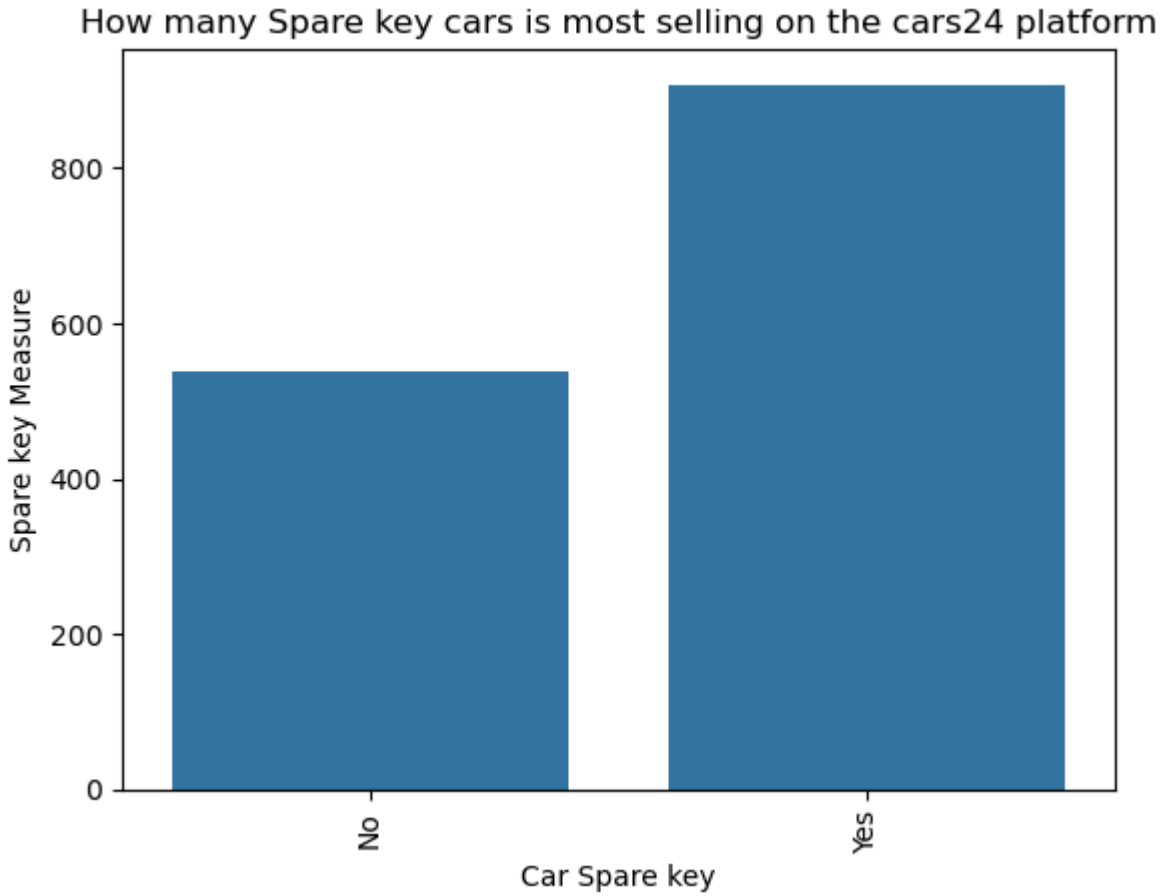
In [42]:
```
# let's visualize the which car model is highest for selling on the cars24 platform

sns.countplot(x = data['Model Year_cat'],data = data)
plt.title("Which Model Year type cars is most selling on the cars24 platform")
plt.xlabel("Car Model Year types")
plt.ylabel("Model Year Types Measure")
plt.xticks(rotation = "vertical")
plt.show()
```



In above chart you can see the mid_age cars are highest for salling on cars24 platform.

In [43]:
```
# let's check the cars owner has how many keys for one car?

sns.countplot(x = data['Spare key'],data = data)
```

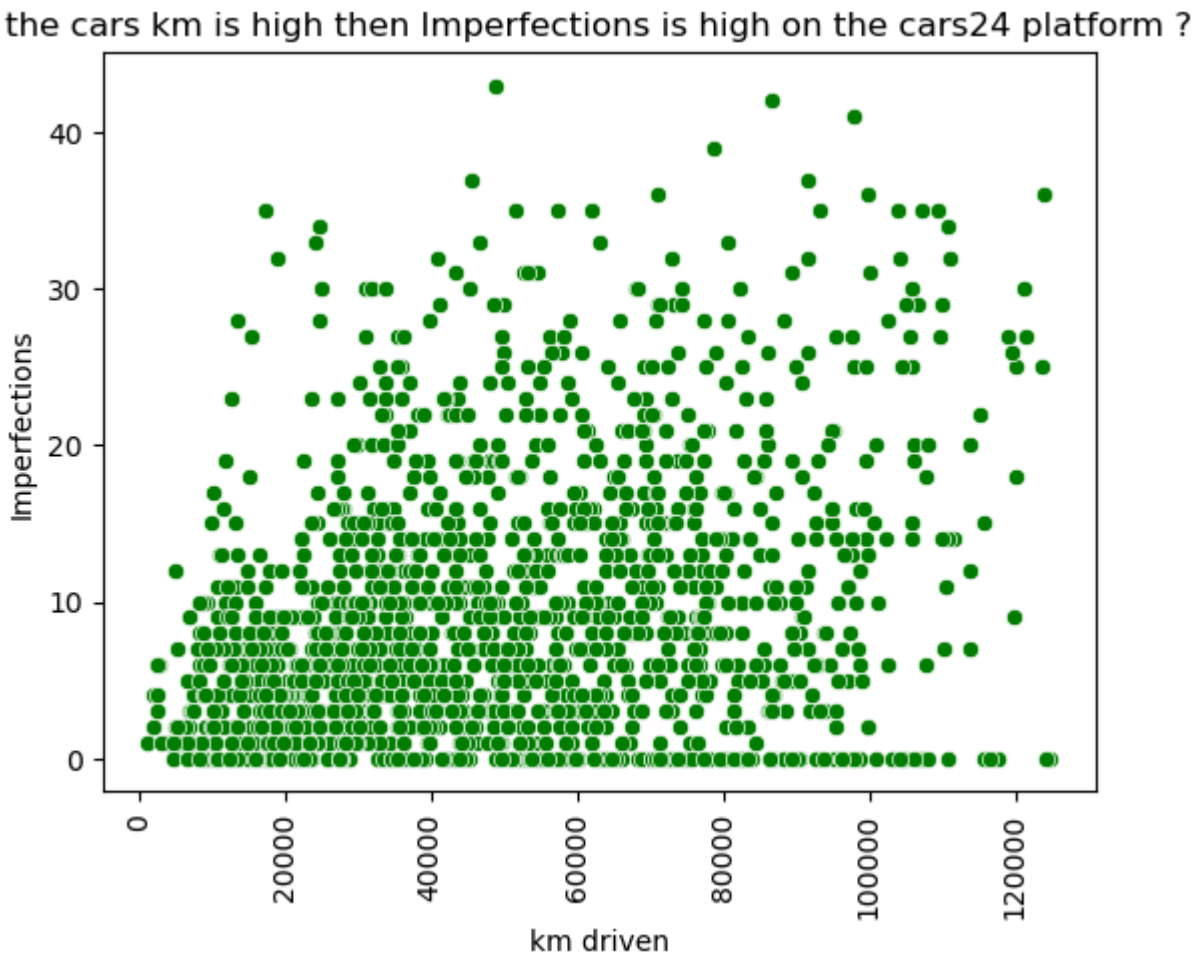
```
plt.title("How many Spare key cars is most selling on the cars24 platform")
plt.xlabel("Car Spare key")
plt.ylabel("Spare key Measure")
plt.xticks(rotation = "vertical")
plt.show()
```



As you can see in the above chart most of the cars has spare keys which is benifit for buyer to buy a car from cars24 platform.

Bi-Variate-Analysis

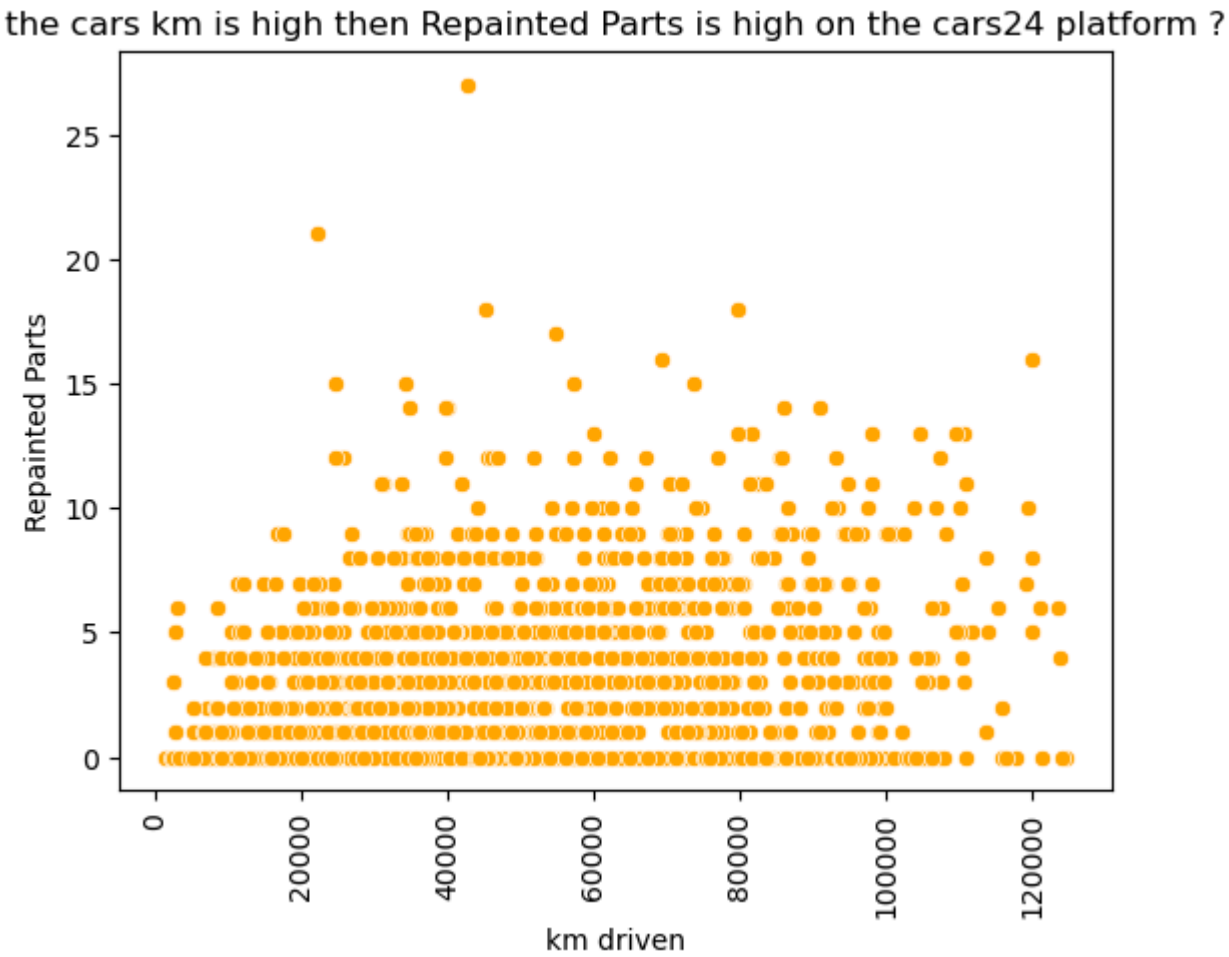
In [44]:
```
# let's see the cars km is high then Imperfections is high check the relationship between 2 variables

sns.scatterplot(x="KM driven",y="Imperfections",data=data,color = "green")
plt.title("the cars km is high then Imperfections is high on the cars24 platform ?")
plt.xlabel("km driven")
plt.ylabel("Imperfections")
plt.xticks(rotation = "vertical")
plt.show()
```



As we see in above scatterplot the car km driven is high then imperfections is high.

In [45]:
```
# let's see the cars km is high then Repainted Parts is high check the relationship between 2 variables

sns.scatterplot(x="KM driven",y="Repainted Parts",data=data,color="orange")
plt.title("the cars km is high then Repainted Parts is high on the cars24 platform ?")
plt.xlabel("km driven")
plt.ylabel("Repainted Parts")
```

```
plt.xticks(rotation = "vertical")
plt.show()
```

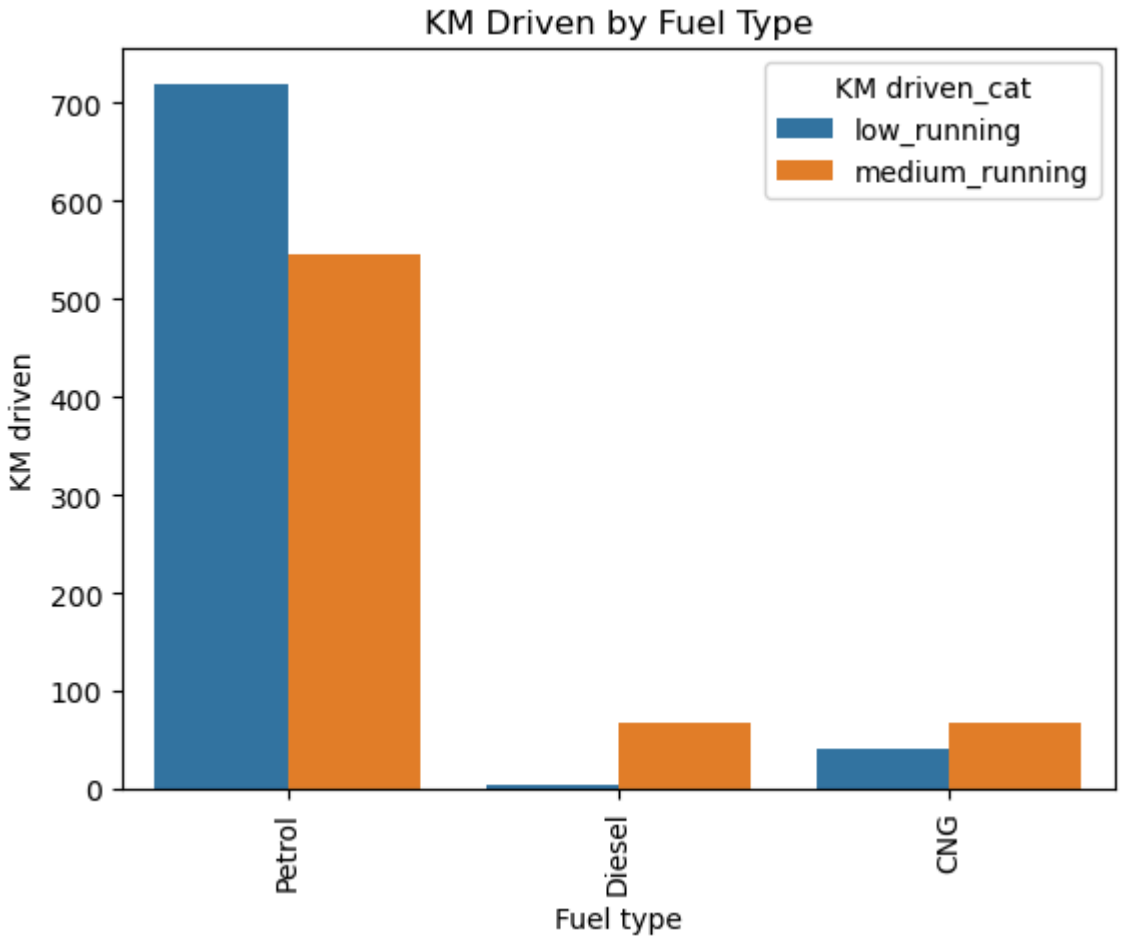


As we can see in the above chart the car km driven is high then the car repainted parts is not much high on cars24 platform.

In [46]:
```
# as we know we have all the column has categorical data

data.dtypes
```

Out[46]:
```
Model Year              int64
Brand Name             object
Model Name             object
Engine capacity         int64
Spare key              object
Transmission           object
KM driven               int64
Ownership               int64
Fuel type              object
Imperfections           int64
Repainted Parts         int64
Imperfections_cat      object
Repainted Parts_cat    object
Ownership_cat          object
KM driven_cat          object
Engine capacity_cat    object
Model Year_cat         object
dtype: object
```

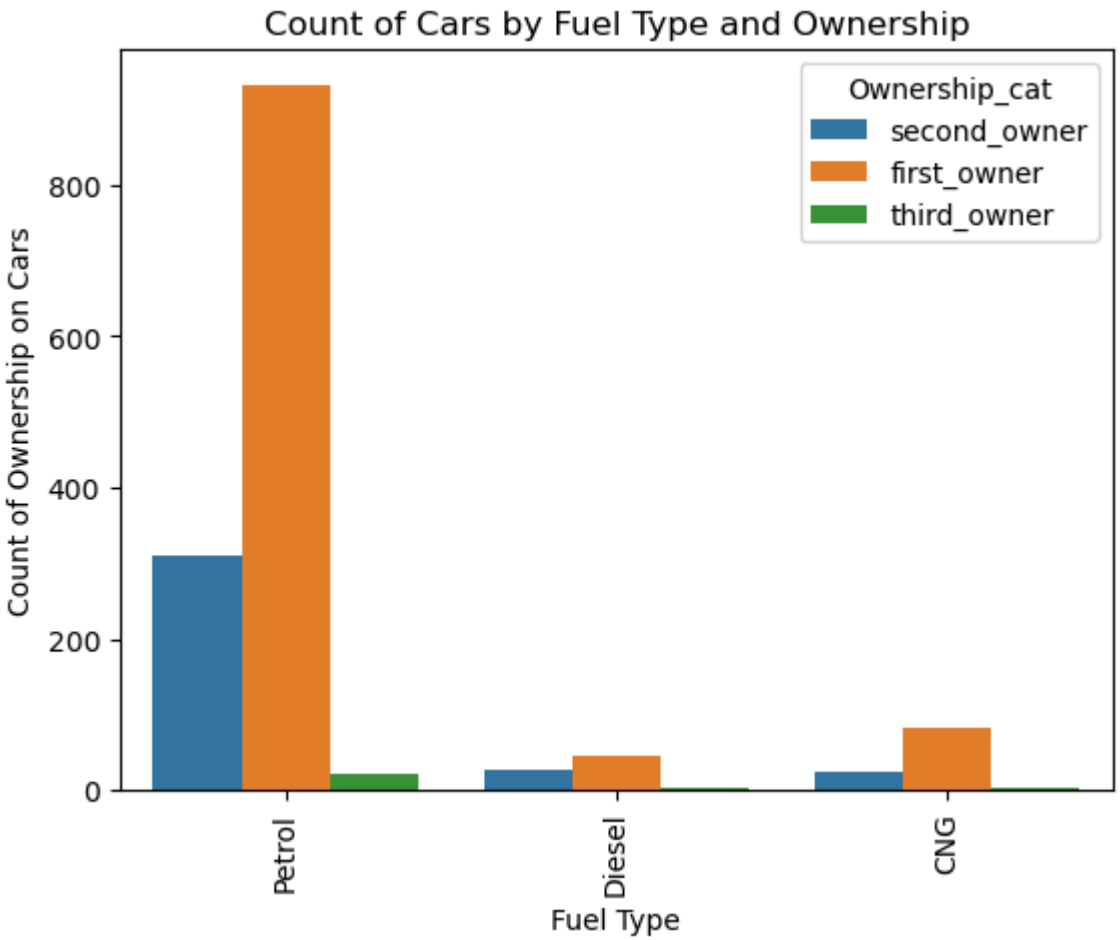
In [47]:
```
# let's see which fuel type cars is highest driven and sale on cars24 platform

sns.countplot(x='Fuel type', hue='KM driven_cat', data=data)
plt.title("KM Driven by Fuel Type")
plt.xlabel("Fuel type")
plt.ylabel("KM driven")
plt.xticks(rotation = "vertical")
plt.show()
```



As you can see most of the petrol driven cars is low running and diesel and cng cars are medium running.

In [48]:
```
# let's see which fuel type cars is highest Ownership and sale on cars24 platform

sns.countplot(x='Fuel type', hue='Ownership_cat', data=data)
plt.title("Count of Cars by Fuel Type and Ownership")
plt.xlabel("Fuel Type")
plt.ylabel("Count of Ownership on Cars")
```

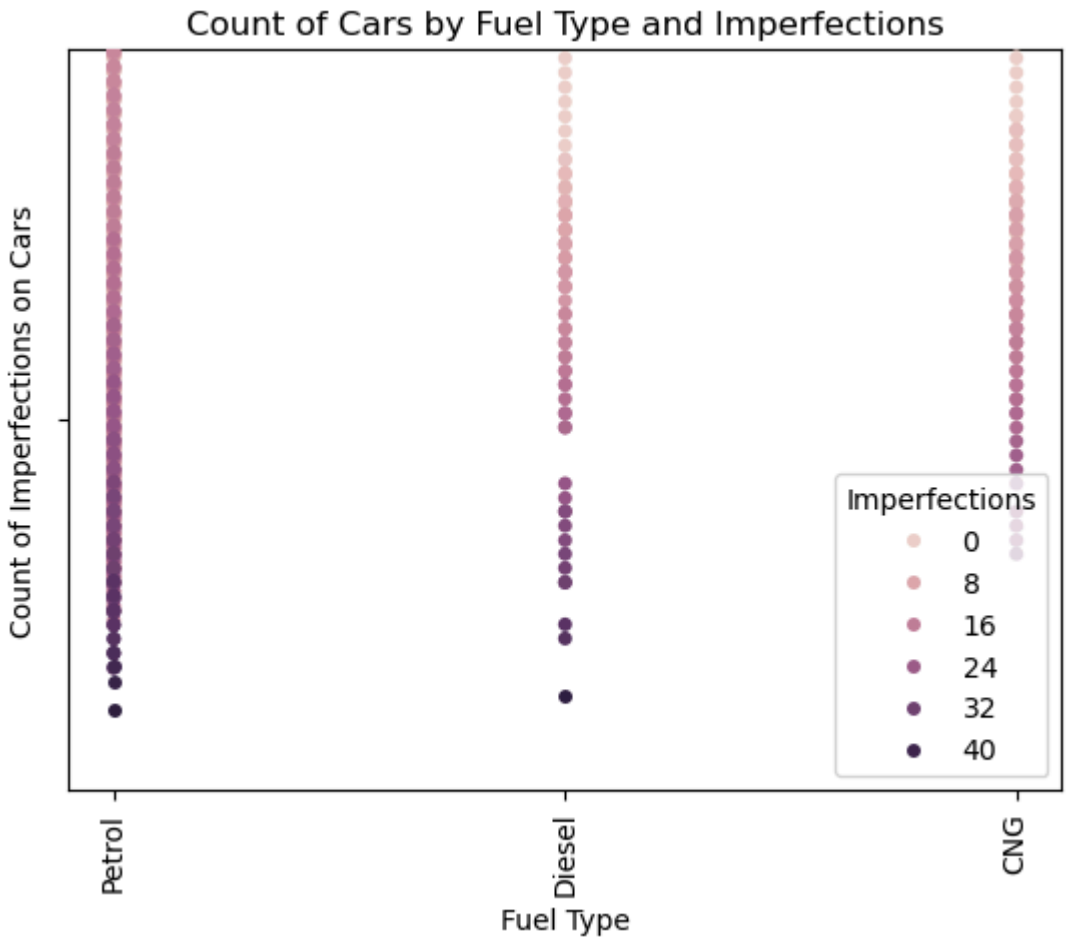
```
plt.xticks(rotation=90)
plt.show()
```



Most of the Petrol Driven Cars is Most for salling and interesting thing is first owner driven cars is high for on sale on cars24 platform.

In [49]:
```
sns.swarmplot(x='Fuel type', hue='Imperfections', data=data,dodge = True)
plt.title("Count of Cars by Fuel Type and Imperfections")
plt.xlabel("Fuel Type")
plt.ylabel("Count of Imperfections on Cars")
```

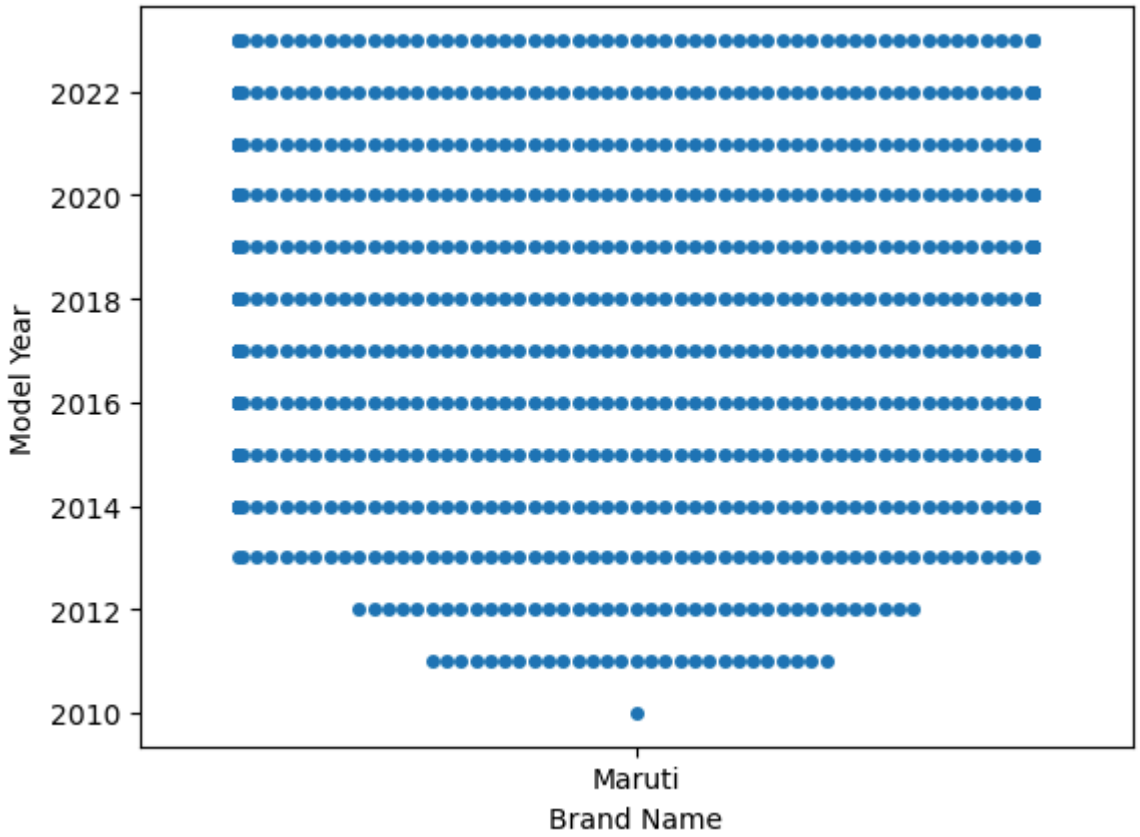
```
plt.xticks(rotation=90)
plt.show()
```



In the above chart as you can see the Imperfections is high in the petrol driven cars as compare to the other fuel type cars.

In [50]:
```
# let's see which car company's car & car model year is salling on cars24 platform.

sns.swarmplot(x = "Brand Name",y = "Model Year",data = data)
```

Out[50]: <Axes: xlabel='Brand Name', ylabel='Model Year'>

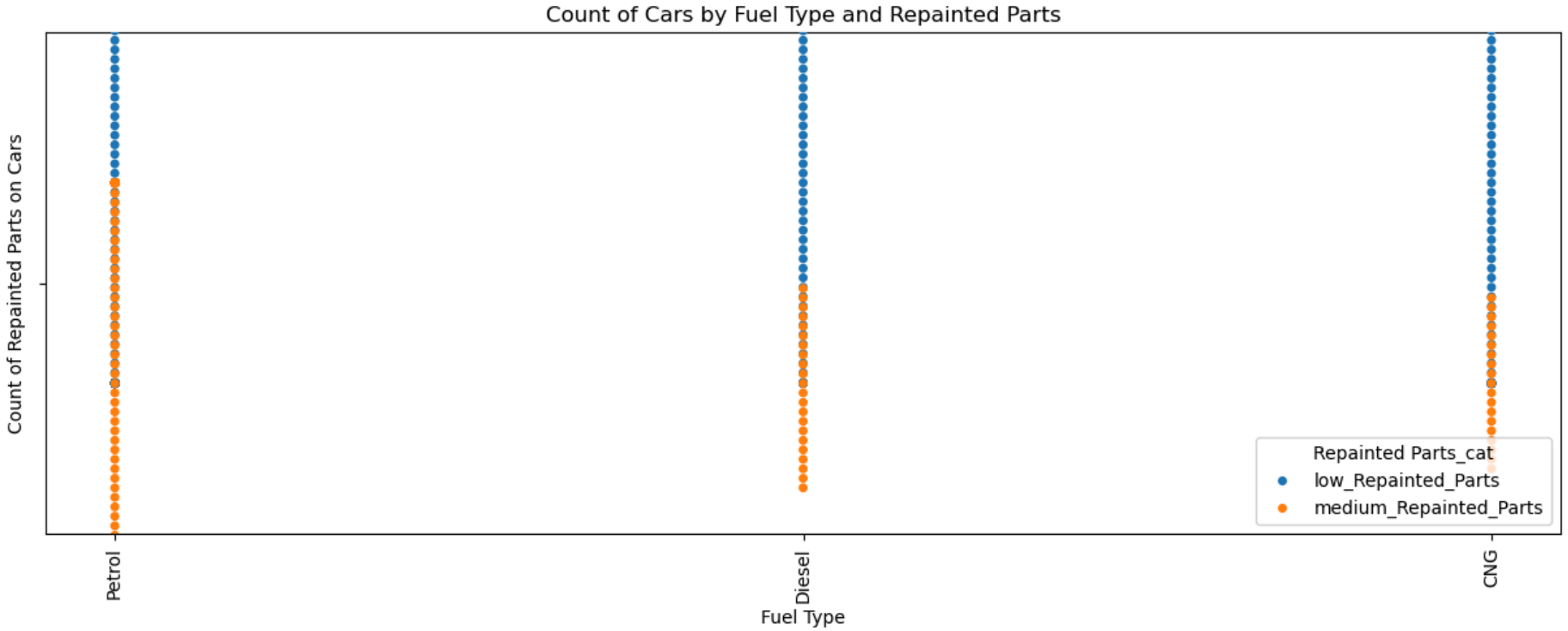


As you can see in the above chart the 2012 above Maruti Brand Cars is highest for on sallings on cars24 platform.

In [51]:
```
# print all the column names
data.columns
```

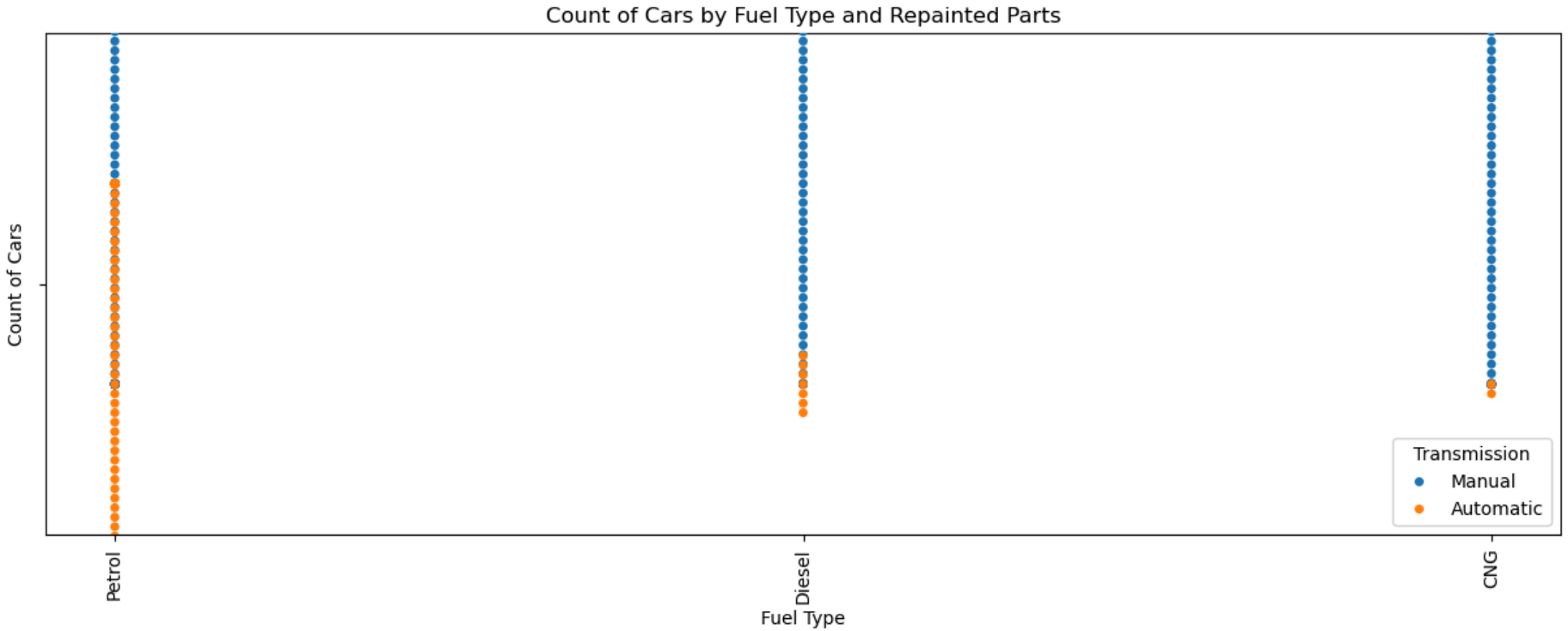
Out[51]:
```
Index(['Model Year', 'Brand Name', 'Model Name', 'Engine capacity',
       'Spare key', 'Transmission', 'KM driven', 'Ownership', 'Fuel type',
       'Imperfections', 'Repainted Parts', 'Imperfections_cat',
       'Repainted Parts_cat', 'Ownership_cat', 'KM driven_cat',
       'Engine capacity_cat', 'Model Year_cat'],
      dtype='object')
```

In [82]:
```
plt.figure(figsize=(15, 5))
sns.swarmplot(x='Fuel type', hue='Repainted Parts_cat', data=data,dodge = True)
plt.title("Count of Cars by Fuel Type and Repainted Parts")
plt.xlabel("Fuel Type")
plt.ylabel("Count of Repainted Parts on Cars")
plt.xticks(rotation=90)
plt.show()
```



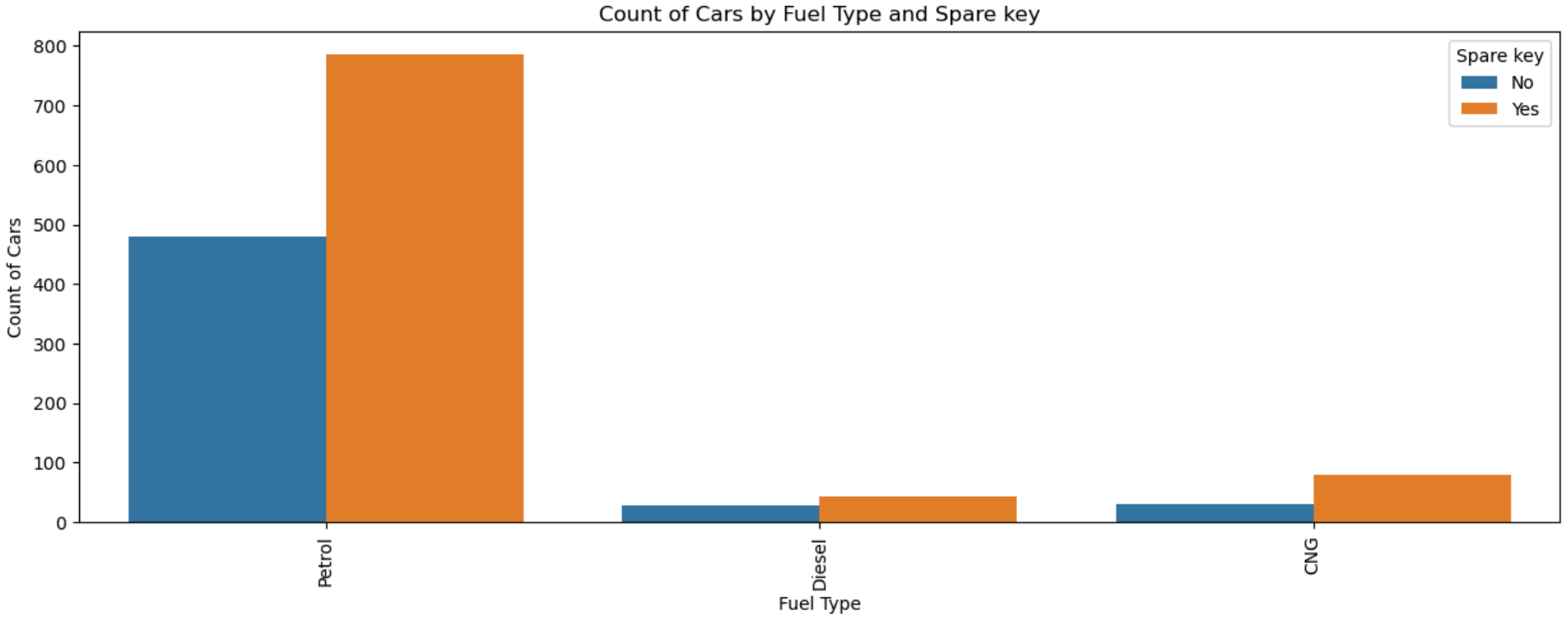
As you can see the Petrol driven cars has the major medium repainted parts as compare to the other fuel type driven cars and important to note that the Diesel & Cng driven cars has the almost similar repainted parts so don't confuse between.

In [83]:
```
plt.figure(figsize=(15, 5))
sns.swarmplot(x='Fuel type', hue='Transmission', data=data,dodge = True)
plt.title("Count of Cars by Fuel Type and Repainted Parts")
plt.xlabel("Fuel Type")
plt.ylabel("Count of Cars")
plt.xticks(rotation=90)
plt.show()
```



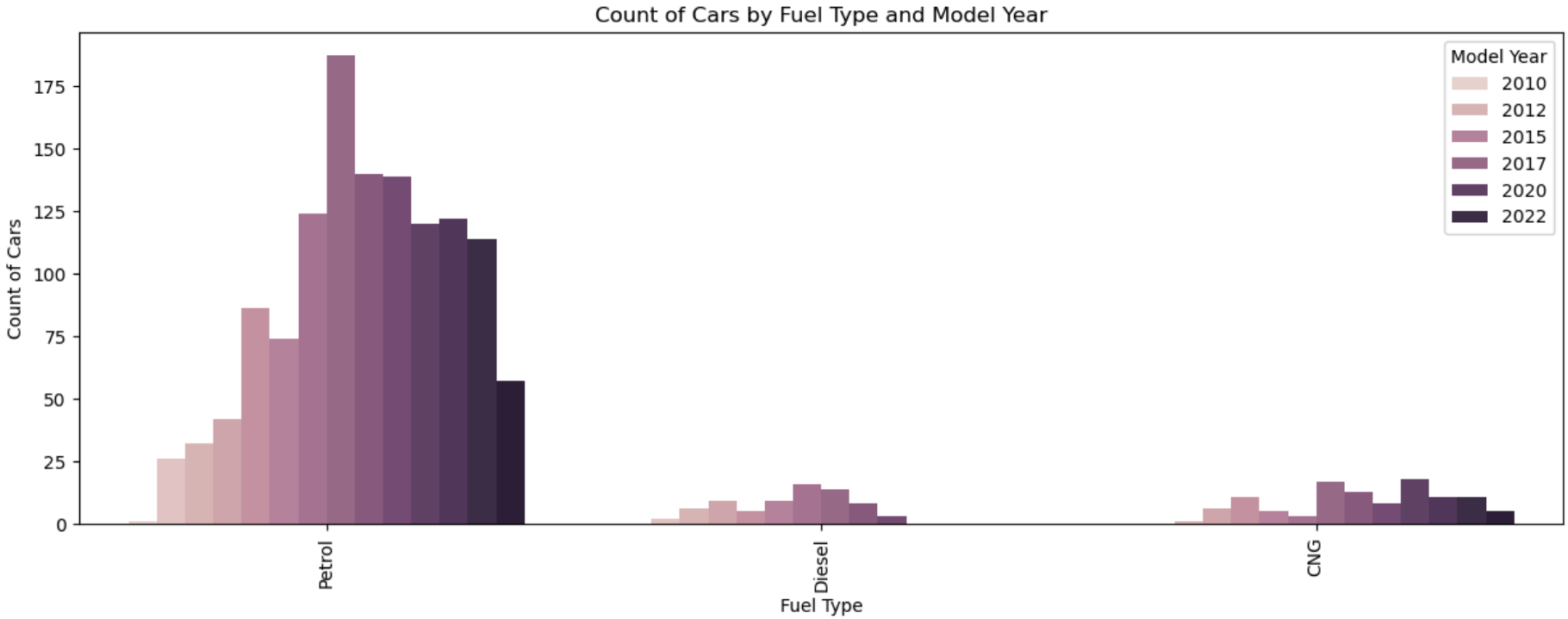
You can see the petrol driven cars is high in the transmission of manual and diesel and cng fuel type cars in the transmission of automatic.

In [54]:
```
plt.figure(figsize=(15, 5))
sns.countplot(x='Fuel type', hue='Spare key', data=data)
plt.title("Count of Cars by Fuel Type and Spare key")
plt.xlabel("Fuel Type")
plt.ylabel("Count of Cars")
plt.xticks(rotation=90)
plt.show()
```



Majority of all the fuel type driven cars has sparekey option is available.

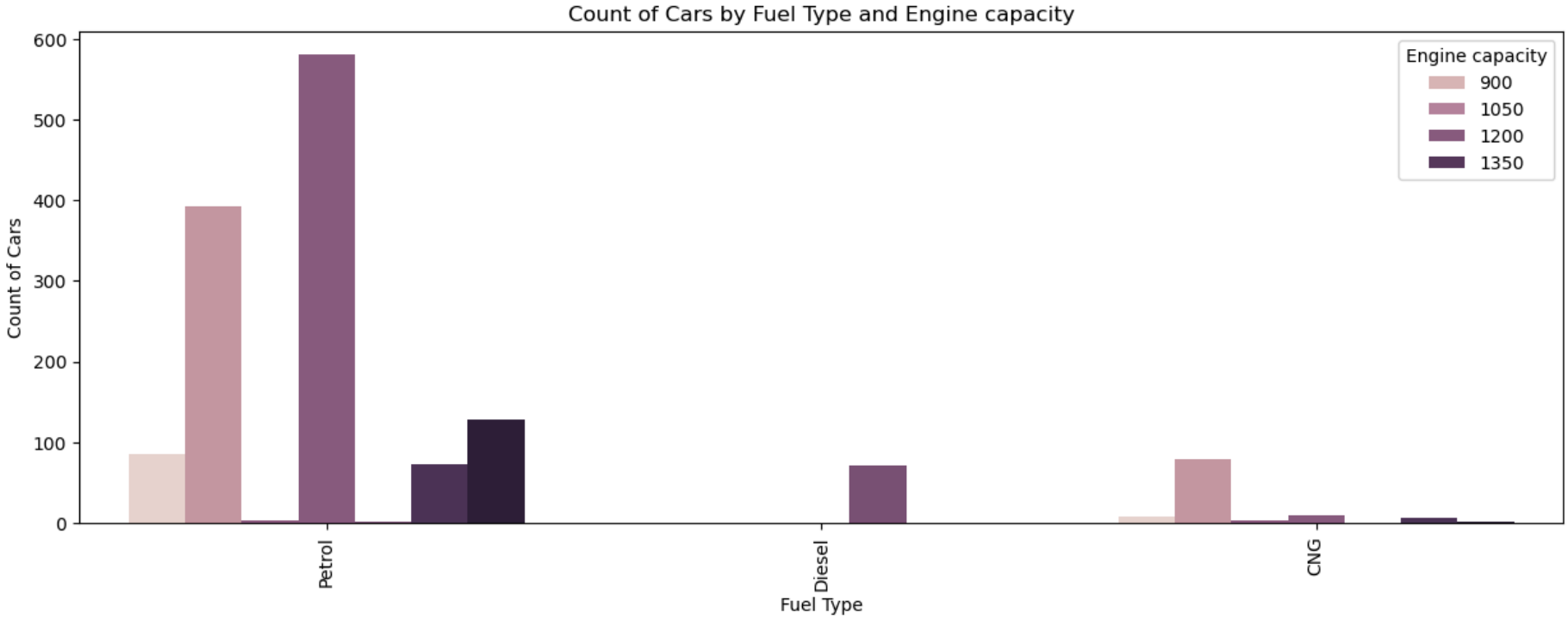
In [55]:
```
# let's check the model year wise fuel type is high

plt.figure(figsize=(15, 5))
sns.countplot(x='Fuel type', hue='Model Year', data=data)
plt.title("Count of Cars by Fuel Type and Model Year")
plt.xlabel("Fuel Type")
plt.ylabel("Count of Cars")
plt.xticks(rotation=90)
plt.show()
```



As you can see in the above charts petrol driven cars has the mid age cars old cars is not in the fuel type.

In [56]:
```
# let's check the which fuel type cars engine cc by category.

plt.figure(figsize=(15, 5))
```

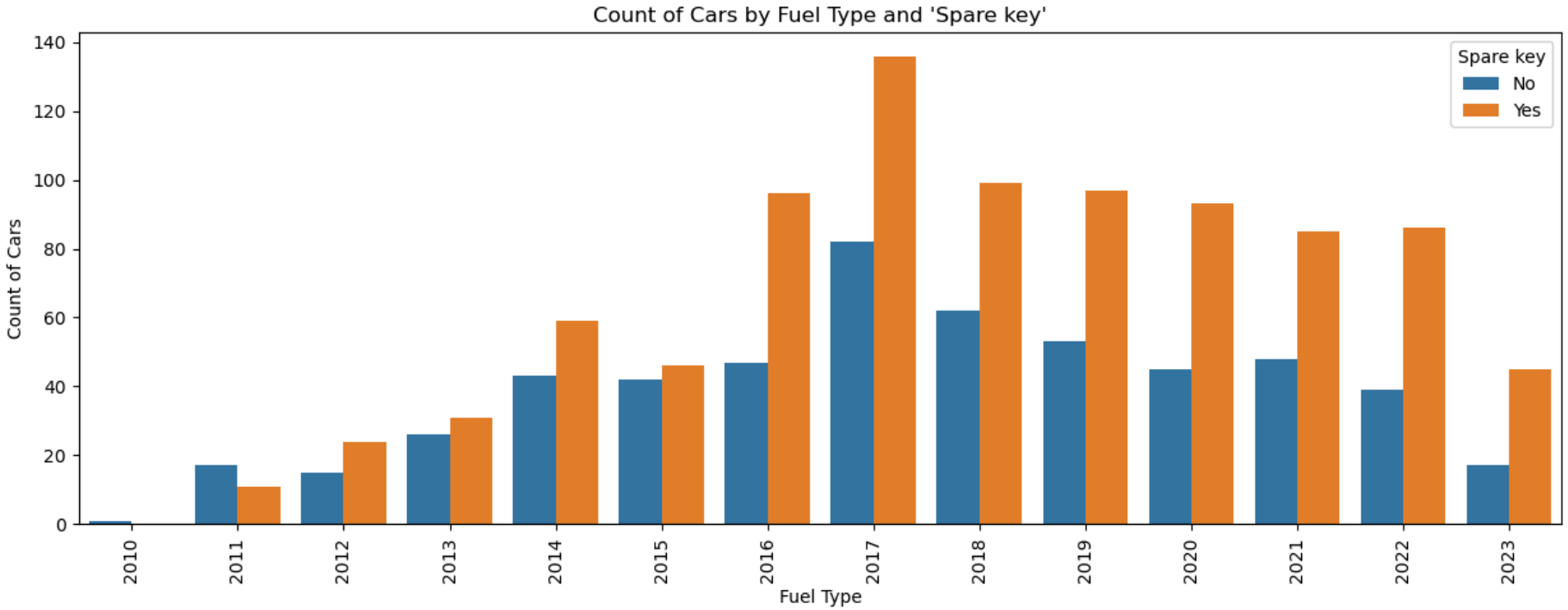
```
sns.countplot(x='Fuel type', hue="Engine capacity", data=data)
plt.title("Count of Cars by Fuel Type and Engine capacity")
plt.xlabel("Fuel Type")
plt.ylabel("Count of Cars")
plt.xticks(rotation=90)
plt.show()
```



As you can see in the above charts petrol driven cars has medium cc engines and diesel driven cars has only one option is available medium cc adn cng cars has the low cc engines.

In [57]:
```
# let's check the which model year has the spare key is available or not?

plt.figure(figsize=(15, 5))
sns.countplot(x='Model Year', hue='Spare key', data=data)
plt.title("Count of Cars by Fuel Type and 'Spare key'")
```

```
plt.xlabel("Fuel Type")
plt.ylabel("Count of Cars")
plt.xticks(rotation=90)
plt.show()
```

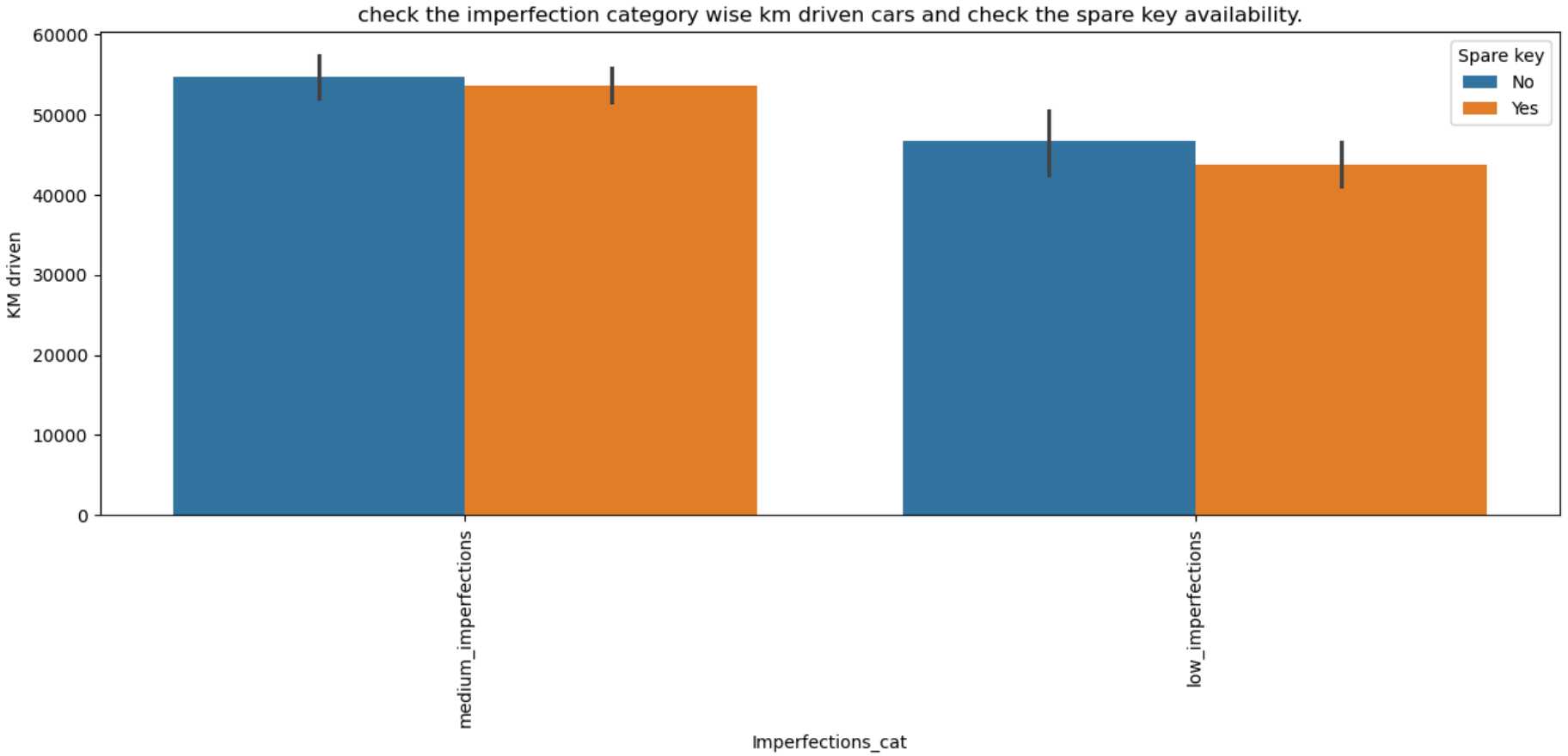


In above chart as you can see the mid age cars has the key is available and old cars has no spare key available so makesure before buy it.

Multi-Variate-Analysis

In [85]:
```
# let's check the Imperfections_cat wise km driven and check the spare key availability

plt.figure(figsize=(15, 5))
sns.barplot(x='Imperfections_cat',y="KM driven",hue = 'Spare key', data=data)
plt.title("check the imperfection category wise km driven cars and check the spare key availability.")
```

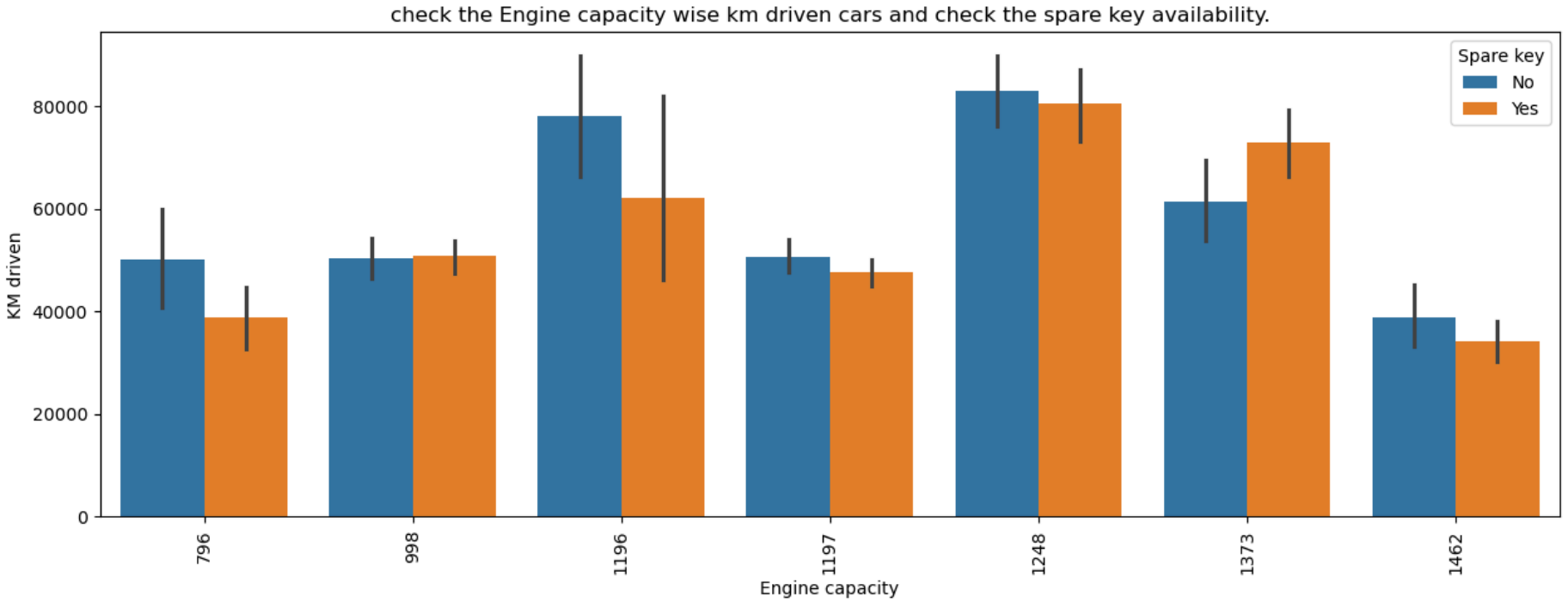
```
plt.xlabel("Imperfections_cat")
plt.ylabel("KM driven")
plt.xticks(rotation=90)
plt.show()
```



You can see the medium imperfections cars are highest runnings in km and interesting thing is that types of cars has no spare key makesure before buy it.

In [86]:
```
# let's check the Engine capacity wise km driven and check the spare key availability

plt.figure(figsize=(15, 5))
sns.barplot(x='Engine capacity',y="KM driven",hue = 'Spare key', data=data)
```

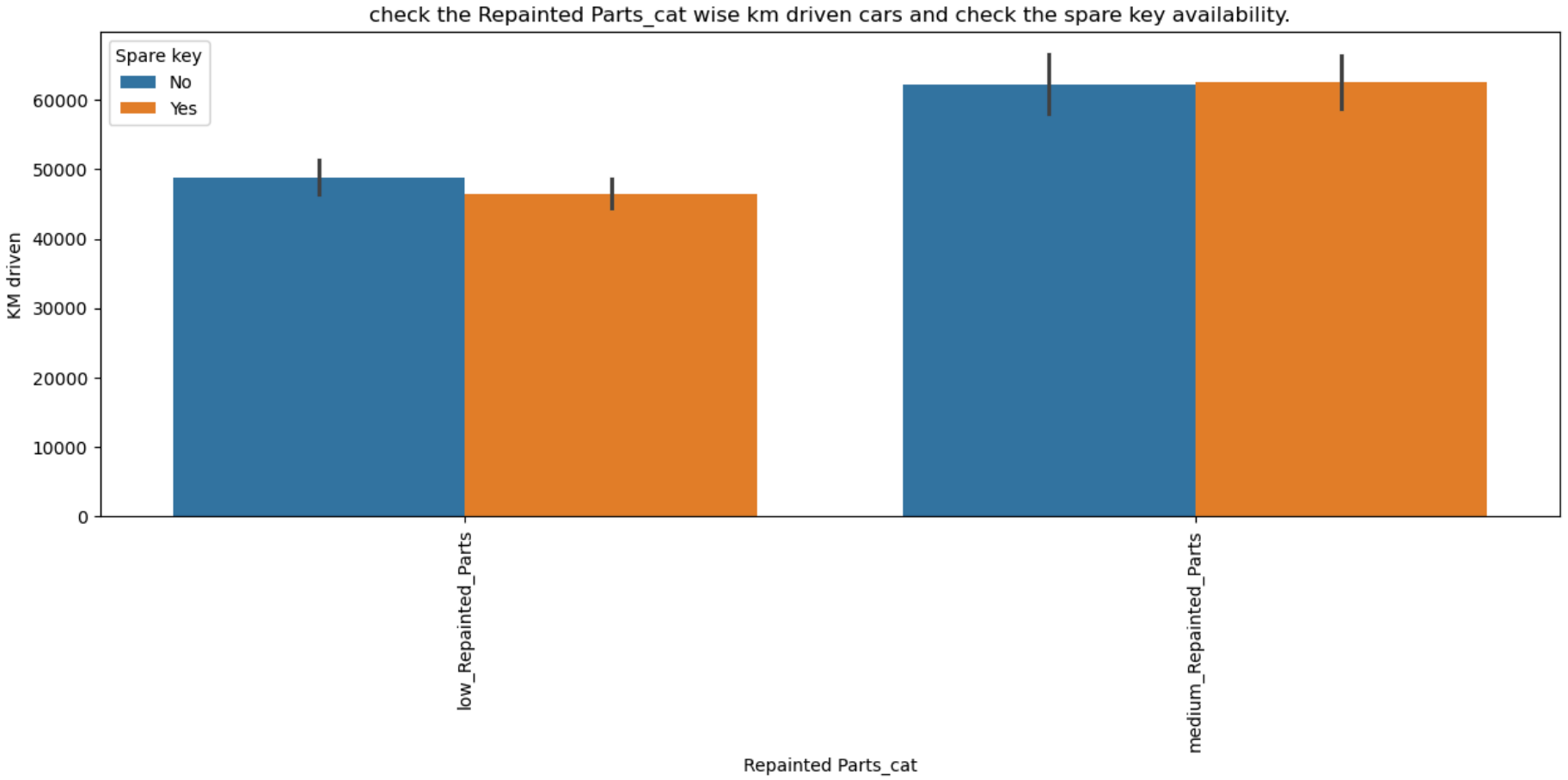
```
plt.title("check the Engine capacity wise km driven cars and check the spare key availability.")
plt.xlabel("Engine capacity")
plt.ylabel("KM driven")
plt.xticks(rotation=90)
plt.show()
```



Majority of all the cars has no spare key option available but the interesting thing is 998 & 1373 engine cc cars has spare key options available.

In [87]:
```
# let's check the Repainted Parts_cat wise km driven and check the spare key availability

plt.figure(figsize=(15, 5))
sns.barplot(x='Repainted Parts_cat',y="KM driven",hue = 'Spare key', data=data)
plt.title("check the Repainted Parts_cat wise km driven cars and check the spare key availability.")
plt.xlabel("Repainted Parts_cat")
plt.ylabel("KM driven")
```

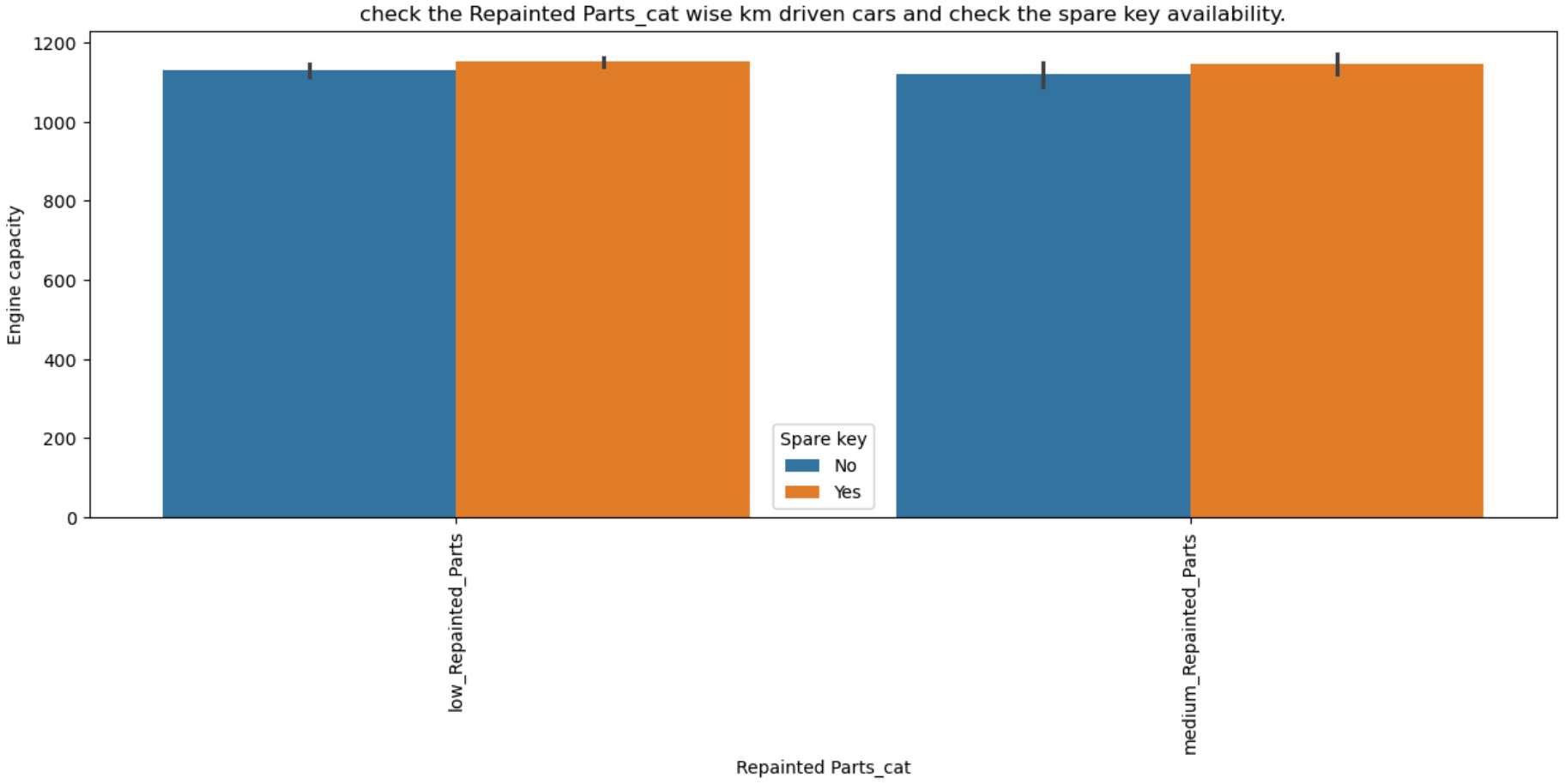
```
plt.xticks(rotation=90)
plt.show()
```



In the above chart the low repainted parts category types of cars has no spare key options available but the interesting thing is medium repainted parts category type of cars has spare key options is available.

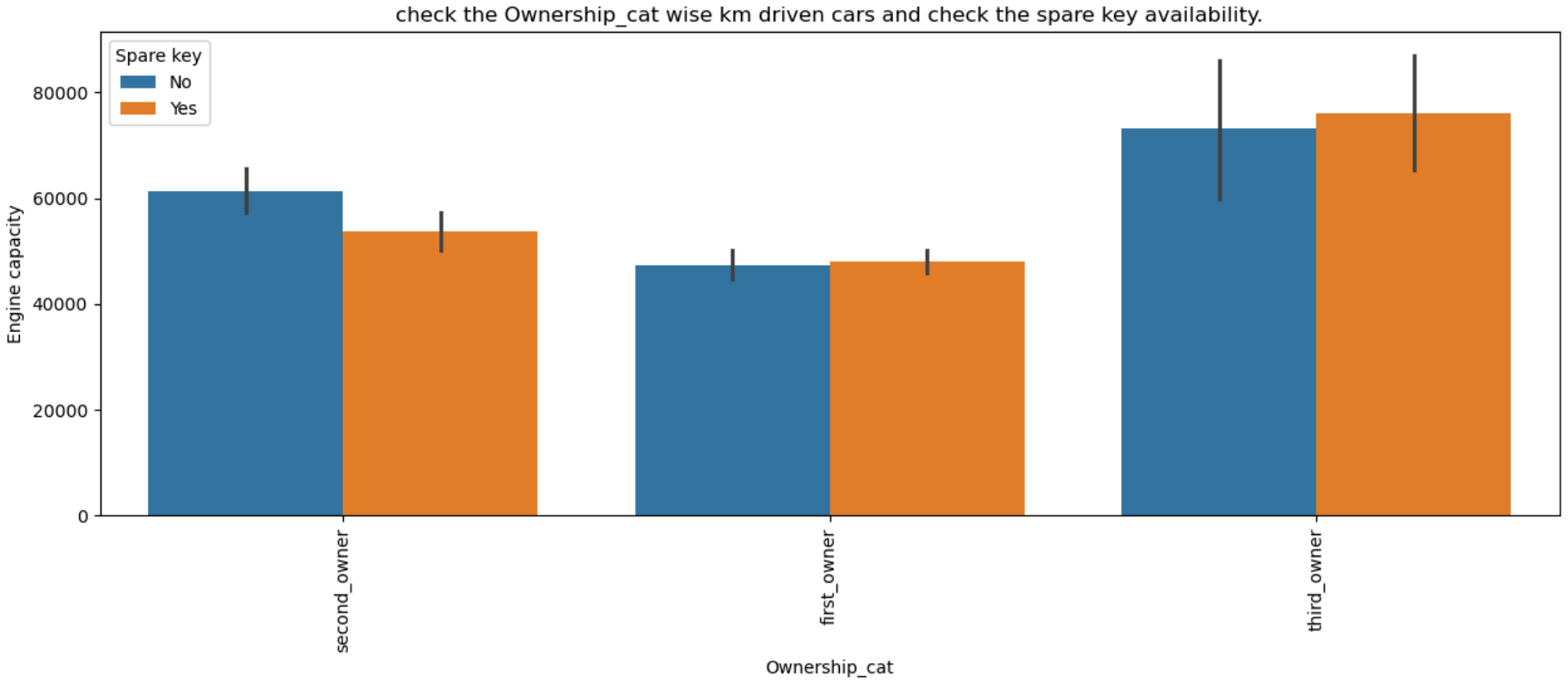
In [90]:
```
# let's check the Repainted Parts_cat wise Engine capacity and check the spare key availability

plt.figure(figsize=(15, 5))
```

```
sns.barplot(x='Repainted Parts_cat',y="Engine capacity",hue = 'Spare key', data=data)
plt.title("check the Repainted Parts_cat wise km driven cars and check the spare key availability.")
plt.xlabel("Repainted Parts_cat")
plt.ylabel("Engine capacity")
plt.xticks(rotation=90)
plt.show()
```



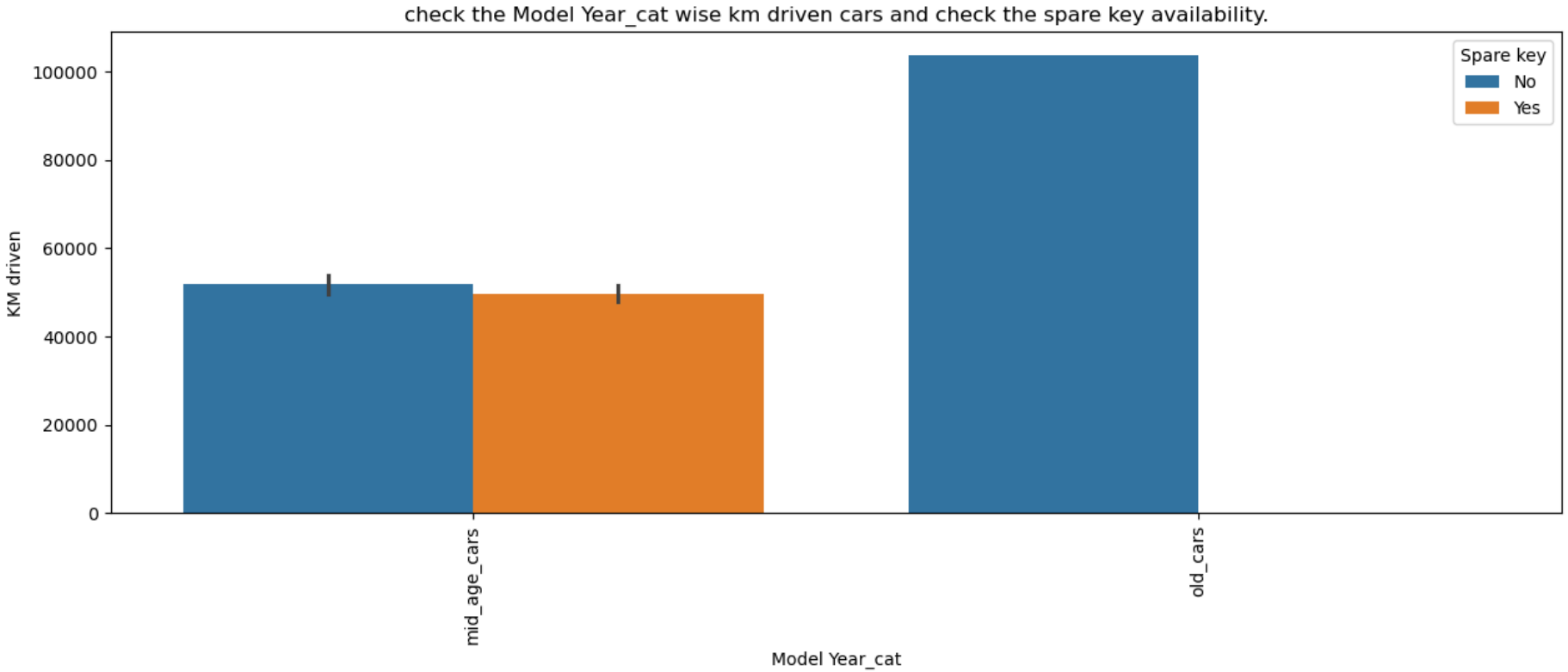
You can see clearly majority of the all the cars below 1200 cc has sparekey options is available.

In [92]:
```
# let's check the Ownership_cat wise Engine capacity and check the spare key availability

plt.figure(figsize=(15, 5))
sns.barplot(x='Ownership_cat',y="KM driven",hue = 'Spare key', data=data)
plt.title("check the Ownership_cat wise km driven cars and check the spare key availability.")
plt.xlabel("Ownership_cat")
plt.ylabel("Engine capacity")
plt.xticks(rotation=90)
plt.show()
```



First & Third owner type used cars has spare key opthions is available but the interesting thing is First owner type cars has no spare key options is availbable.

In [94]:
```
# let's check the Ownership_cat wise Engine capacity and check the spare key availability

plt.figure(figsize=(15, 5))
sns.barplot(x='Model Year_cat',y="KM driven",hue = 'Spare key', data=data)
plt.title("check the Model Year_cat wise km driven cars and check the spare key availability.")
plt.xlabel("Model Year_cat")
plt.ylabel("KM driven")
plt.xticks(rotation=90)
plt.show()
```



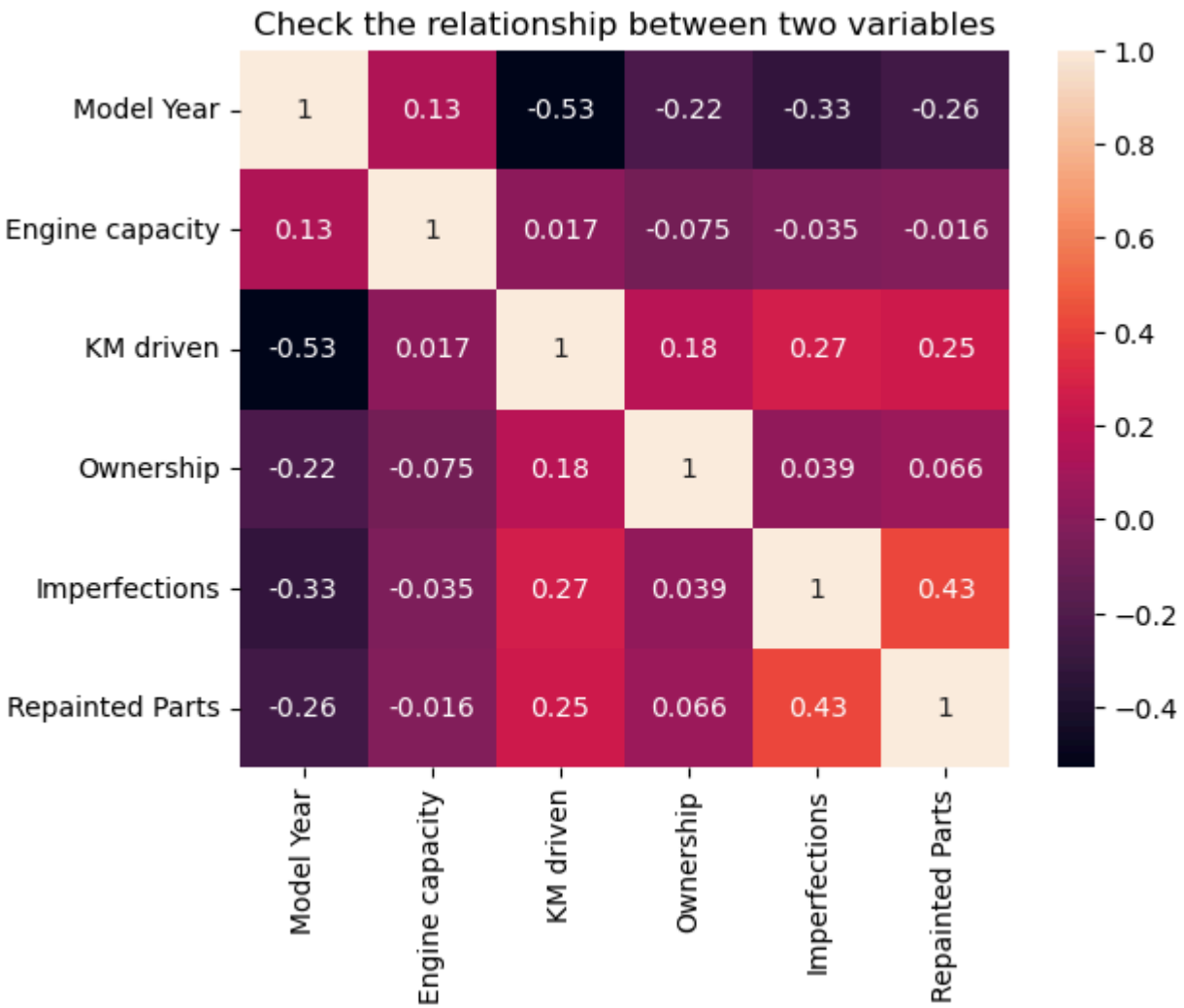
Majority of the all the cars has key options available but the interesting thing is those cars nearest or above 1 lakh km has spare key options is available.

In [95]:
```
# check the relationship between variables

relation_variables =  data.corr(numeric_only = True)
relation_variables
```

Out[95]:

	Model Year	Engine capacity	KM driven	Ownership	Imperfections	Repainted Parts
Model Year	1.000000	0.128797	-0.529817	-0.217609	-0.326156	-0.256837
Engine capacity	0.128797	1.000000	0.017436	-0.075150	-0.034947	-0.015830
KM driven	-0.529817	0.017436	1.000000	0.183771	0.266018	0.249148
Ownership	-0.217609	-0.075150	0.183771	1.000000	0.039040	0.066348
Imperfections	-0.326156	-0.034947	0.266018	0.039040	1.000000	0.426276
Repainted Parts	-0.256837	-0.015830	0.249148	0.066348	0.426276	1.000000

In [96]:
```
# plot the heatmap for relationship variables

sns.heatmap(relation_variables,annot = True)
plt.title("Check the relationship between two variables")
plt.show()
```



You can see the Imperfection wise km driven has 27% relationship

km driven wise repainted parts has 25% relationship

imperfections wise repainted parts has 43% relationship

📋 Final EDA Insights and Recommendations

1) Verify Ownership and Spare Key Availability:

Always ensure that the spare key is available, especially for mid-aged cars and first-owner cars. A significant percentage of older cars or those with imperfections may not come with a spare key, so check this before making a purchase.

2) Car Age Considerations:

Mid-aged cars tend to have higher resale value and maintain better condition. Consider mid-aged cars over very old cars as they are usually in better shape. Be cautious with older petrol cars as they might have more imperfections.

3) Fuel Type Considerations:

Petrol cars are the most popular on the platform and are often low-running. However, they tend to have more imperfections. Carefully inspect petrol cars for any flaws.

Diesel and CNG cars are typically medium-running and are more likely to have automatic transmission. Choose automatic transmission cars if that's your preference.

4) Engine Capacity and Transmission Type:

Most cars have medium engine capacity (cc). Petrol cars often fall into this category. If you want higher performance, look for medium cc engine cars.

Manual transmission is common in petrol cars, while diesel and CNG cars generally feature automatic transmission. Choose based on your driving style and preference.

5) Repainted Parts:

Check the repainted parts of the car, as petrol-driven cars often have medium repainted parts. Excessive repainting may indicate prior accidents or repairs. Inspect the car's condition thoroughly before buying.

6) Running Distance (KM):

Low-running (KM) cars are more desirable as they tend to have a longer lifespan. Look for low KM cars to ensure better condition and longevity.

Medium KM cars, especially diesel and CNG cars, are still reliable options, so don’t overlook them if you’re seeking a dependable vehicle.

7) Brand Preferences:

Maruti cars manufactured in 2012 or later are some of the highest-selling models on the platform. If you’re looking for a reliable car, Maruti is a solid choice.

8) Imperfections in Petrol Cars:

Petrol cars generally have higher imperfections compared to other fuel types. If you are considering a petrol car, ensure that the imperfections are not too severe.

9) Repainted Parts and Spare Key Availability:

Cars with medium-level repainted parts are more likely to have a spare key. These cars are often in better condition, so consider them if you prioritize having a spare key.

10) Mileage Considerations:

First-owner cars typically have better maintenance records and lower running KM. Look for first-owner cars as they tend to be well-maintained, offering better long-term value.