DATA ANALYSIS AND VISUALIZATION PROJECT

CAR SALE ANALYSIS

Submitted by:

- **Komal Chamyal 13515**
- **Pulkit Vashisht 13560**
- **Yuvraj Singh 13508**
- **Abhishek Jeph 13565**

Dataset Description

The Car Sales Information is a dataset with data on car sales ads in Russia by region. The data is taken from a popular website in Russia with ads for the sale of cars. Data is collected hourly from the first hundred pages. The only filter is the region to search for. The date and time of selection is indicated in the "parse_date" column.

It includes various brands of cars like Toyota, Honda, Ford, Porsche and so on. The dataset has numerous information about the engine type, transmission type, mileage and configuration.

The Columns precisely include:

- `Brand` **Brand of Vehicle**
- `Name` **Model Name of Car**
- `BodyType` **Car Body type**
- `Color` **Car Color**
- `FuelType` **Fuel Type used in car**
- `Year` **Year of manufacture of car**
- `Mileage` **Car mileage**
- `Transmission` **Type of transmission of the machine**
- `Power` **Horsepower**
- `Price` **Price in Russian Rubies**
- `VehicleConfiguration`, `EngineName`, `EngineDisplacement` **Technical Details about the car**
- `date`, `parse_date` **Date of scrapping of data**
- `location` **Location of sale of car in Russia**

Download link

> https://www.kaggle.com/datasets/ekibee/car-sales-information

1. Importing Libraries

In [1]:

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

Reading dataset CSV file

In [2]:

```
df1 = pd.read_csv("region41_en.csv")
```

Copy data to manipulate

In [3]:

```
cars = df1.copy()
cars
```

Out[3]:

	brand	name	bodyType	color	fuelType	year	mileage	transmission	power	price	vehicleConfigu
0	Toyota	Land Cruiser Prado	jeep 5 doors	blue	Diesel	1995.0	168000.0	AT	130.0	1860000	3.0 SX Wide I diesel
1	Toyota	Land Cruiser	jeep 5 doors	black	Diesel	NaN	260000.0	Automatic	286.0	2300000	
2	Toyota	Vitz	hatchback 5 doors	blue	Gasoline	2019.0	100000.0	CVT	95.0	1075000	1.3 F Safety E II
3	Toyota	Mark II	sedan	grey	Gasoline	2002.0	239000.0	AT	160.0	480000	2.0 Grand
4	Toyota	RAV4	jeep 5 doors	golden	Gasoline	2010.0	101000.0	AT	170.0	1450000	2.4 AT Престиж
...	...	...	...	...	...	...	...	...	...	...	
1498735	Toyota	Caldina	station wagon	white	Gasoline	NaN	250000.0	AT	260.0	390000	
1498736	Honda	HR-V	jeep 3 doors	silver	Gasoline	1998.0	250000.0	CVT	105.0	370000	
1498737	Mazda	CX-7	jeep 5 doors	black	Gasoline	2006.0	108000.0	AT	244.0	500000	2.3 AT T
1498738	Mitsubishi	RVR	jeep 5 doors	burgundy	Gasoline	2012.0	112000.0	CVT	139.0	1100000	1.8 Roadest (
1498739	Nissan	Elgrand	minivan	grey	Gasoline	2002.0	111000.0	AT	240.0	1599999	3.5 VIP specifi

1498740 rows × 17 columns

2. Exploring the dataset

Checking the rows and columns of the data

In [4]:

```
cars.head()
```

Out[4]:

	brand	name	bodyType	color	fuelType	year	mileage	transmission	power	price	vehicleConfiguration	engin
0	Toyota	Land Cruiser Prado	jeep 5 doors	blue	Diesel	1995.0	168000.0	AT	130.0	1860000	3.0 SX Wide limited diesel turbo	1
1	Toyota	Land Cruiser	jeep 5 doors	black	Diesel	NaN	260000.0	Automatic	286.0	2300000	NaN	
2	Toyota	Vitz	hatchback 5 doors	blue	Gasoline	2019.0	100000.0	CVT	95.0	1075000	1.3 F Safety Edition III 4WD	1
3	Toyota	Mark II	sedan	grey	Gasoline	2002.0	239000.0	AT	160.0	480000	2.0 Grande Four	
4	Toyota	RAV4	jeep 5 doors	golden	Gasoline	2010.0	101000.0	AT	170.0	1450000	2.4 AT Long Престиж Плюс	2

In [5]:

```
cars.shape
```

Out[5]:

```
(1498740, 17)
```

In [6]:

```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1498740 entries, 0 to 1498739
Data columns (total 17 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   brand                 1498740 non-null  object
 1   name                  1498740 non-null  object
 2   bodyType              1498740 non-null  object
 3   color                 1448934 non-null  object
 4   fuelType              1492145 non-null  object
 5   year                  915699 non-null   float64
 6   mileage               1491876 non-null  float64
 7   transmission          1491339 non-null  object
 8   power                 1484500 non-null  float64
 9   price                 1498740 non-null  int64
 10  vehicleConfiguration  915699 non-null   object
 11  engineName            915506 non-null   object
 12  engineDisplacement    914866 non-null   object
 13  date                  1498740 non-null  object
 14  location              1498740 non-null  object
 15  link                  1498740 non-null  object
 16  parse_date            1498740 non-null  object
dtypes: float64(3), int64(1), object(13)
memory usage: 194.4+ MB
```

In [7]:

```
cars.columns
```

Out[7]:

```
Index(['brand', 'name', 'bodyType', 'color', 'fuelType', 'year', 'mileage',
       'transmission', 'power', 'price', 'vehicleConfiguration', 'engineName',
       'engineDisplacement', 'date', 'location', 'link', 'parse_date'],
      dtype='object')
```

In [8]:

```
cars.describe()
```

Out[8]:

	year	mileage	power	price
count	915699.000000	1.491876e+06	1.484500e+06	1.498740e+06
mean	2005.327732	1.823240e+05	1.588104e+02	1.147137e+06
std	8.206993	1.013326e+05	7.169883e+01	1.370128e+06
min	1953.000000	1.000000e+03	4.500000e+01	6.000000e+03
25%	1999.000000	1.100000e+05	1.070000e+02	3.800000e+05
50%	2006.000000	1.770000e+05	1.400000e+02	7.500000e+05
75%	2012.000000	2.500000e+05	1.850000e+02	1.400000e+06
max	2021.000000	1.000000e+06	6.500000e+02	3.300000e+07

Unique values in the `color` and `fuelType` columns

In [9]:

```
list(cars.color.unique())
```

Out[9]:

```
['blue',
 'black',
 'grey',
 'golden',
 'silver',
 'white',
 'beige',
 'red',
 'burgundy',
 nan,
 'green',
 'brown',
 'yellow',
 'pink',
 'violet',
 'orange']
```

In [10]:

```
list(cars.fuelType.unique())
```

Out[10]:

```
['Diesel', 'Gasoline', nan, 'Electro']
```

---

3. Data Cleaning

3.1 Dropping irrelevant columns

The dataset has a column `Link` which is not relevant to our analysis.

In [11]:

```
cars.head(1)
```

Out[11]:

	brand	name	bodyType	color	fuelType	year	mileage	transmission	power	price	vehicleConfiguration	engineN
0	Toyota	Land Cruiser Prado	jeep 5 doors	blue	Diesel	1995.0	168000.0	AT	130.0	1860000	3.0 SX Wide limited diesel turbo	1K

The dataset also has two similar columns `date` and `parse_date` which refer to the date and date+hours of the running sale ad in website respectively. Scraping `date` column as `parse_date` also provides same information.

In [12]:

```
cars=cars.drop(columns=['link', 'date'])
cars
```

Out[12]:

	brand	name	bodyType	color	fuelType	year	mileage	transmission	power	price	vehicleConfigu
0	Toyota	Land Cruiser Prado	jeep 5 doors	blue	Diesel	1995.0	168000.0	AT	130.0	1860000	3.0 SX Wide l diesel
1	Toyota	Land Cruiser	jeep 5 doors	black	Diesel	NaN	260000.0	Automatic	286.0	2300000	
2	Toyota	Vitz	hatchback 5 doors	blue	Gasoline	2019.0	100000.0	CVT	95.0	1075000	1.3 F Safety E II
3	Toyota	Mark II	sedan	grey	Gasoline	2002.0	239000.0	AT	160.0	480000	2.0 Grand
4	Toyota	RAV4	jeep 5 doors	golden	Gasoline	2010.0	101000.0	AT	170.0	1450000	2.4 AT Престиж
...	...	...	...	...	...	...	...	...	...	...	
1498735	Toyota	Caldina	station wagon	white	Gasoline	NaN	250000.0	AT	260.0	390000	
1498736	Honda	HR-V	jeep 3 doors	silver	Gasoline	1998.0	250000.0	CVT	105.0	370000	
1498737	Mazda	CX-7	jeep 5 doors	black	Gasoline	2006.0	108000.0	AT	244.0	500000	2.3 AT T
1498738	Mitsubishi	RVR	jeep 5 doors	burgundy	Gasoline	2012.0	112000.0	CVT	139.0	1100000	1.8 Roadest (
1498739	Nissan	Elgrand	minivan	grey	Gasoline	2002.0	111000.0	AT	240.0	1599999	3.5 VIP specifi

1498740 rows × 15 columns

Renaming `parse_date` as `date`

In [13]:

```
cars.rename(columns = {'parse_date':'date'}, inplace = True)
```

3.2 Handling Missing Values

Checking for the null values in each column

In [14]:

```
cars.isnull().sum()
```

Out[14]:

```
brand                         0
name                          0
bodyType                      0
color                     49806
fuelType                   6595
year                     583041
mileage                    6864
transmission               7401
power                     14240
price                         0
vehicleConfiguration     583041
engineName               583234
engineDisplacement       583874
location                      0
date                          0
dtype: int64
```

In [15]:

```
#Percentage of Null values
for col in cars.columns:
    null = cars[col].isnull().sum()
    percentage = (null/len(cars))*100
    print(col,":", round(percentage),"%")
```

```
brand : 0 %
name : 0 %
bodyType : 0 %
color : 3 %
fuelType : 0 %
year : 39 %
mileage : 0 %
transmission : 0 %
power : 1 %
price : 0 %
vehicleConfiguration : 39 %
engineName : 39 %
engineDisplacement : 39 %
location : 0 %
date : 0 %
```

Filling missing values

To fill the nan values we find the which entity have maximum frequecy in each column to find frequency we calculate the mode

In [16]:

```
a = cars.color.mode()
cars.color.fillna(a[0],inplace = True)
ftm = cars.fuelType.mode()
cars.fuelType.fillna(ftm[0],inplace = True)
cym = cars.year.mode()
cars.year.fillna(cym[0],inplace = True)
cmm = cars.mileage.mean()
cars.mileage.fillna(cmm,inplace = True)
ctm = cars.transmission.mode()
cars.transmission.fillna(ctm[0],inplace = True)
```

```python
cpm = cars.power.mean()
cars.power.fillna(cpm,inplace = True)
cvcm = cars.vehicleConfiguration.mode()
cars.vehicleConfiguration.fillna(cvcm[0],inplace = True)
cen = cars.engineName.mode()
cars.engineName.fillna(cen[0],inplace = True)
ced = cars.engineDisplacement.mode()
cars.engineDisplacement.fillna(ced[0],inplace = True)
```

Checking Null Values Now

In [17]:

```python
cars.isnull().sum()
```

Out[17]:

```
brand                   0
name                    0
bodyType                0
color                   0
fuelType                0
year                    0
mileage                 0
transmission            0
power                   0
price                   0
vehicleConfiguration    0
engineName              0
engineDisplacement      0
location                0
date                    0
dtype: int64
```

Changing the datatype of `date` from `OBJECT` TO `DATETIME64` TYPE

In [18]:

```python
#Before parsing
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1498740 entries, 0 to 1498739
Data columns (total 15 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   brand                 1498740 non-null  object
 1   name                  1498740 non-null  object
 2   bodyType              1498740 non-null  object
 3   color                 1498740 non-null  object
 4   fuelType              1498740 non-null  object
 5   year                  1498740 non-null  float64
 6   mileage               1498740 non-null  float64
 7   transmission          1498740 non-null  object
 8   power                 1498740 non-null  float64
 9   price                 1498740 non-null  int64
 10  vehicleConfiguration  1498740 non-null  object
 11  engineName            1498740 non-null  object
 12  engineDisplacement    1498740 non-null  object
 13  location              1498740 non-null  object
 14  date                  1498740 non-null  object
dtypes: float64(3), int64(1), object(11)
memory usage: 171.5+ MB
```

In [19]:

```python
cars['date'] = pd.to_datetime(cars.date)
```

In [20]:

```python
#After parsing
```

```
#After parsing
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1498740 entries, 0 to 1498739
Data columns (total 15 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   brand                 1498740 non-null  object
 1   name                  1498740 non-null  object
 2   bodyType              1498740 non-null  object
 3   color                 1498740 non-null  object
 4   fuelType              1498740 non-null  object
 5   year                  1498740 non-null  float64
 6   mileage               1498740 non-null  float64
 7   transmission          1498740 non-null  object
 8   power                 1498740 non-null  float64
 9   price                 1498740 non-null  int64
 10  vehicleConfiguration  1498740 non-null  object
 11  engineName            1498740 non-null  object
 12  engineDisplacement    1498740 non-null  object
 13  location              1498740 non-null  object
 14  date                  1498740 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(3), int64(1), object(10)
memory usage: 171.5+ MB
```

3.3 Handling Duplicate Values

Checking for duplicate values

In [21]:

```
cars.duplicated().sum()
```

Out[21]:

8048

In [22]:

```
cars[cars.duplicated()]
```

Out[22]:

	brand	name	bodyType	color	fuelType	year	mileage	transmission	power	price	vehicleConfiguration	e
656	Nissan	Bluebird Sylphy	sedan	grey	Gasoline	2005.0	247000.0	AT	109.0	650000	1.5 15M FOUR 4WD	
2658	Nissan	Bluebird Sylphy	sedan	grey	Gasoline	2005.0	247000.0	AT	109.0	650000	1.5 15M FOUR 4WD	
4659	Nissan	Bluebird Sylphy	sedan	grey	Gasoline	2005.0	247000.0	AT	109.0	650000	1.5 15M FOUR 4WD	
6659	Nissan	Bluebird Sylphy	sedan	grey	Gasoline	2005.0	247000.0	AT	109.0	650000	1.5 15M FOUR 4WD	
8659	Nissan	Bluebird Sylphy	sedan	grey	Gasoline	2005.0	247000.0	AT	109.0	650000	1.5 15M FOUR 4WD	
...	...	...	...	...	...	...	...	...	...	...	...	
1491234	Suzuki	Jimny Wide	jeep 3 doors	silver	Gasoline	1998.0	229000.0	AT	85.0	350000	1.5 G 4WD	
1492843	Suzuki	Jimny Wide	jeep 3 doors	silver	Gasoline	1998.0	229000.0	AT	85.0	350000	1.5 G 4WD	

	brand	name	bodyType	color	fuelType	year	mileage	transmission	power	price	vehicleConfiguration	e
1494449	Suzuki	Jimny Wide	jeep 3 doors	silver	Gasoline	1998.0	229000.0	AT	85.0	350000	1.5 G 4WD	
1496052	Suzuki	Jimny Wide	jeep 3 doors	silver	Gasoline	1998.0	229000.0	AT	85.0	350000	1.5 G 4WD	
1497637	Suzuki	Jimny Wide	jeep 3 doors	silver	Gasoline	1998.0	229000.0	AT	85.0	350000	1.5 G 4WD	

8048 rows × 15 columns

Dropping the Duplicates

In [23]:

```
cars.drop_duplicates(inplace=True)
```

In [24]:

```
#After dropping
cars.duplicated().sum()
```

Out[24]:

0

3.4 Handling Outliers

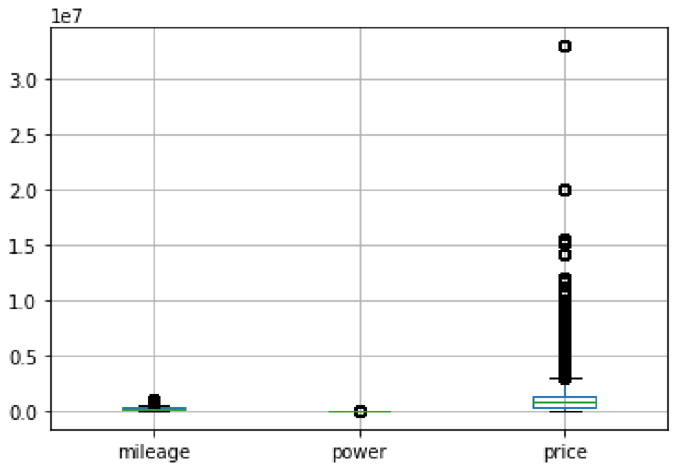
Checking outliers through Boxplot

In [25]:

```
cars.boxplot(column=['mileage','power','price'])
```

Out[25]:

<AxesSubplot:>



We can clearly see here that there are many outlier values in 'price' column but we can't manipulate or remove these outlier values because these values are helpful in our analysis. Hence we will not operate on these outlier value.

QUERY 1

QUERY 1

Minimum, Maximum, Count and Mean `Mileage` of each `bodyType` of car

In [26]:

```
cars.groupby(by=['bodyType']).mileage.agg(["min",
                                          "max",
                                          "count",
                                          "mean"]).round(2)
```

Out[26]:

bodyType	min	max	count	mean
coupe	1000.0	384000.0	6038	154054.82
hatchback 3 door	1000.0	500000.0	8249	177895.38
hatchback 5 doors	1000.0	1000000.0	131713	142062.65
jeep 3 doors	1000.0	556000.0	113256	197989.40
jeep 5 doors	1000.0	1000000.0	657612	165320.42
liftback	91000.0	300000.0	4059	174588.81
minivan	1000.0	1000000.0	119929	209302.62
open	120000.0	160000.0	832	141875.00
pickup	1000.0	1000000.0	35655	172418.50
sedan	1000.0	1000000.0	283360	218663.81
station wagon	1000.0	900000.0	129989	195847.76

QUERY 2

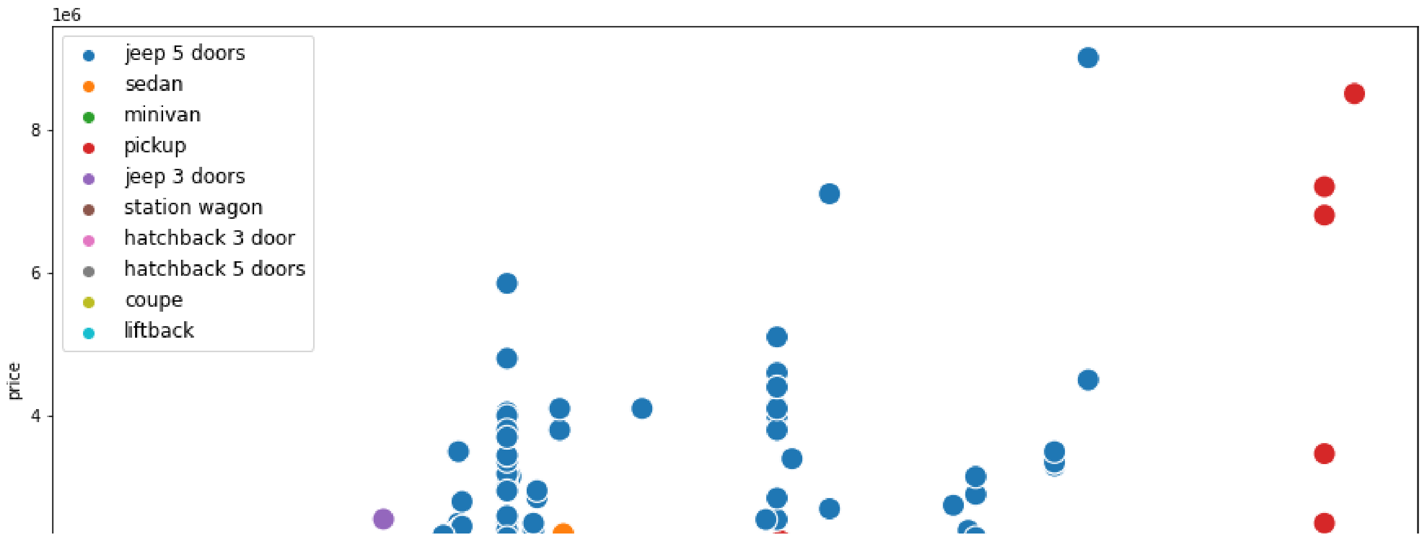
The scatter plot shows a slight positive correlation between `price` and `power`.

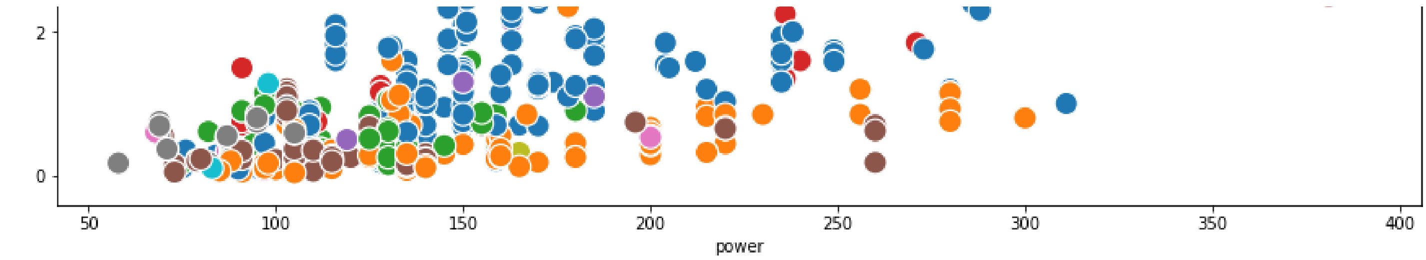
In [27]:

```
df = cars.sort_values(by=['brand'], ascending=False).head(15000)
plt.rcParams['figure.figsize']=(15,8)
sns.scatterplot(x=df.power,y=df.price,hue=df.bodyType,s=200)
plt.legend(loc='upper left',fontsize='12')
plt.xlabel('power')
plt.ylabel('price')
```

Out[27]:

```
Text(0, 0.5, 'price')
```



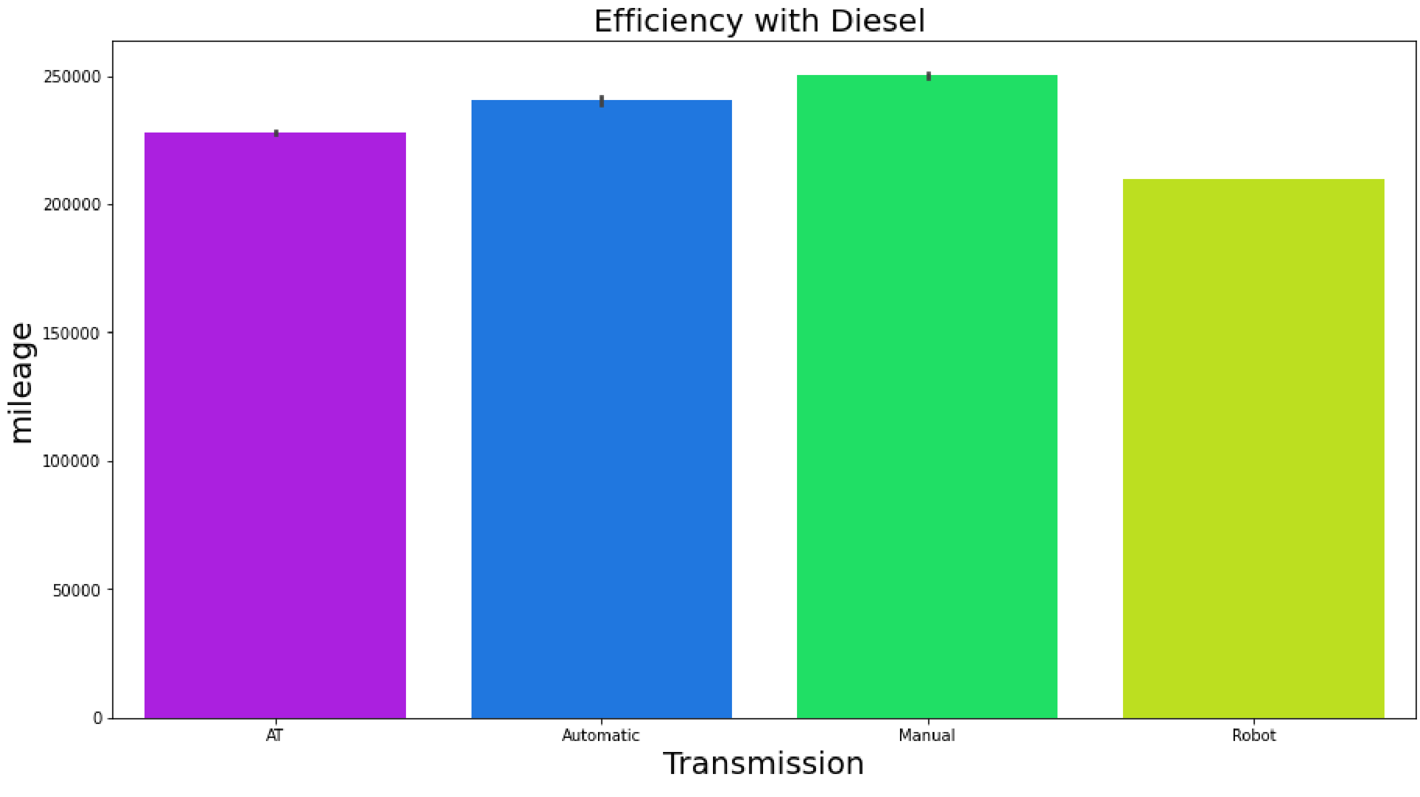


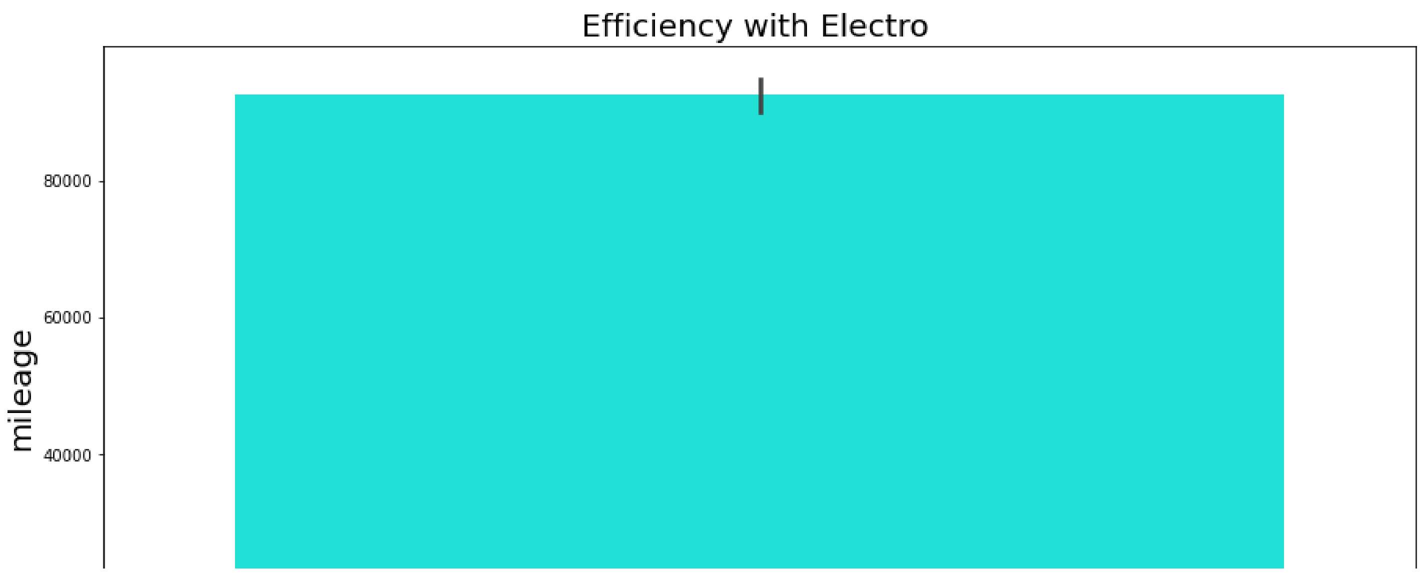
QUERY 3

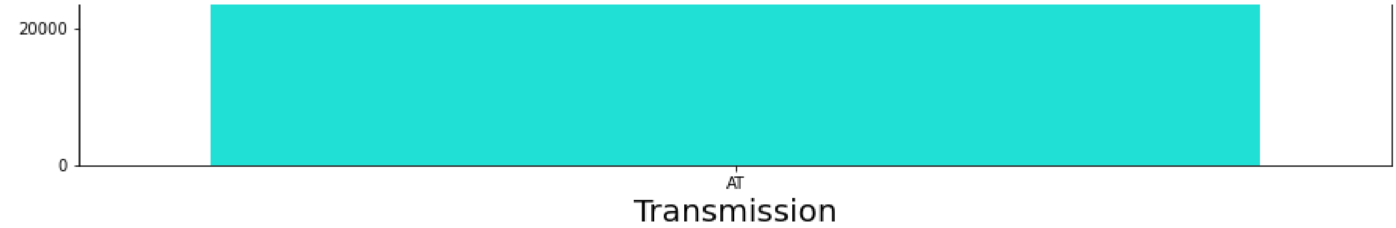
mileages with each fueltype on the basis of Transmisson

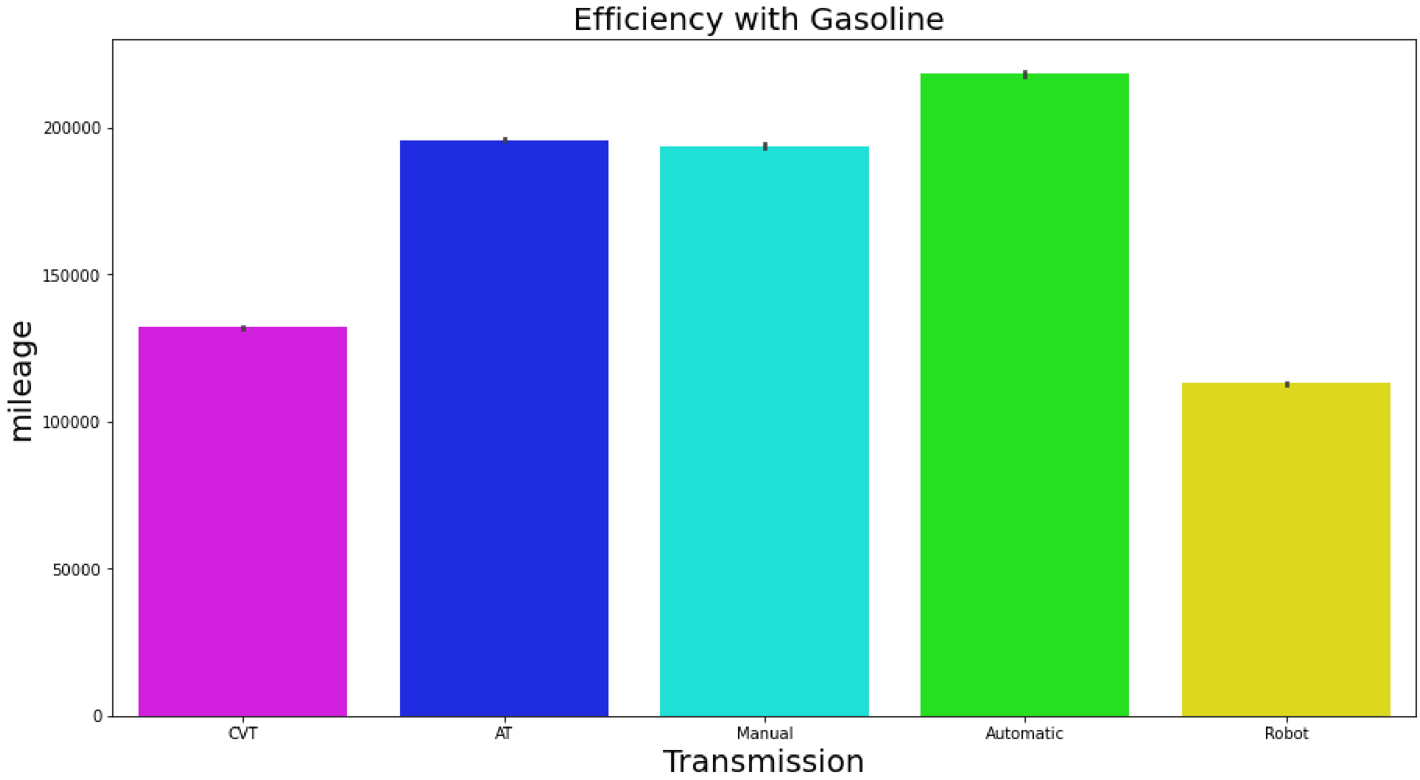
In [28]:

```
top10data = cars.query("mileage>= 10000")
for i,j in top10data.groupby('fuelType'):
    sns.barplot(x=j['transmission'],y=j['mileage'],label=f"{i}",palette='hsv_r')
    plt.title(f"Efficiency with {i} ",fontsize=20)
    plt.ylabel('mileage',fontsize=20)
    plt.xlabel("Transmission",fontsize=20)
    plt.xticks(fontsize = 10)
    plt.yticks(fontsize = 10)
    plt.show()
```







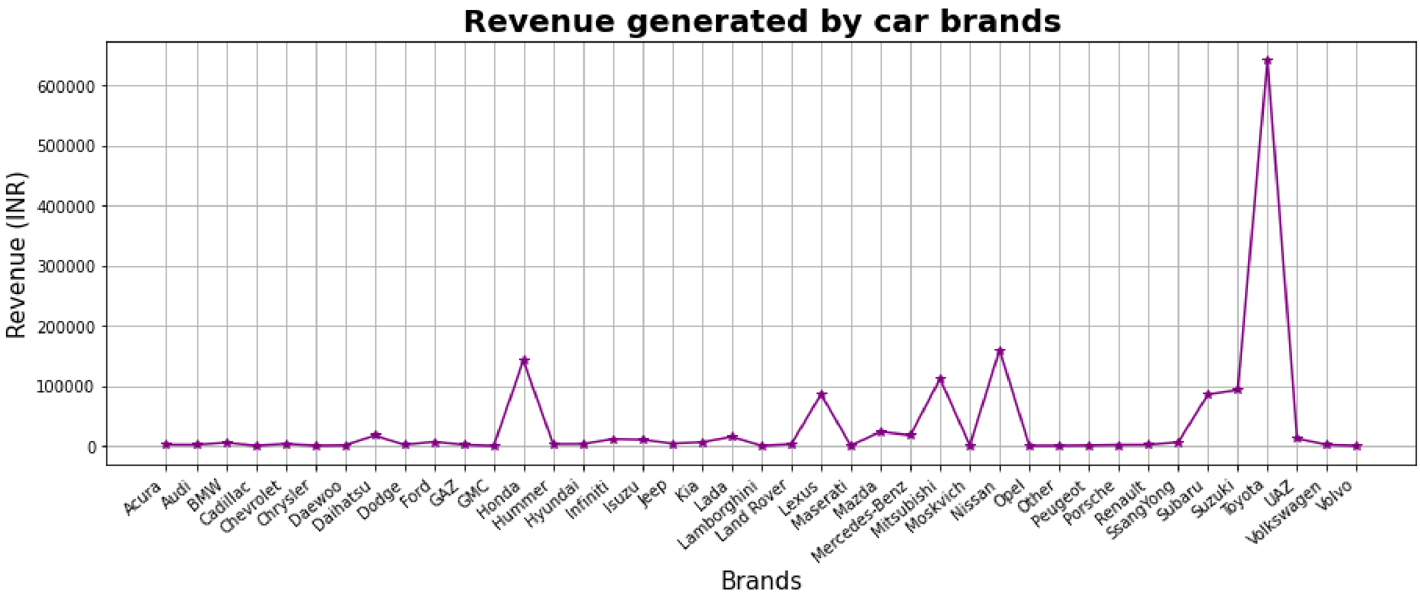


QUERY 4

query for revenue generated by different car brands

In [29]:

```
df= cars['price'].groupby(cars['brand']).value_counts().unstack('brand').sum(axis=0)
plt.figure(figsize = (15,5))
plt.plot(df, marker='*',color='purple')
plt.xlabel('Brands',fontsize=15)
plt.ylabel('Revenue (INR)',fontsize=15)
plt.xticks(size = 10, rotation=40, ha="right")
plt.yticks(size = 10)
plt.title('Revenue generated by car brands', fontsize=20, fontweight="bold")
plt.grid()
```



It can be concluded that Toyota brands generates the maximum revenue

QUERY 5

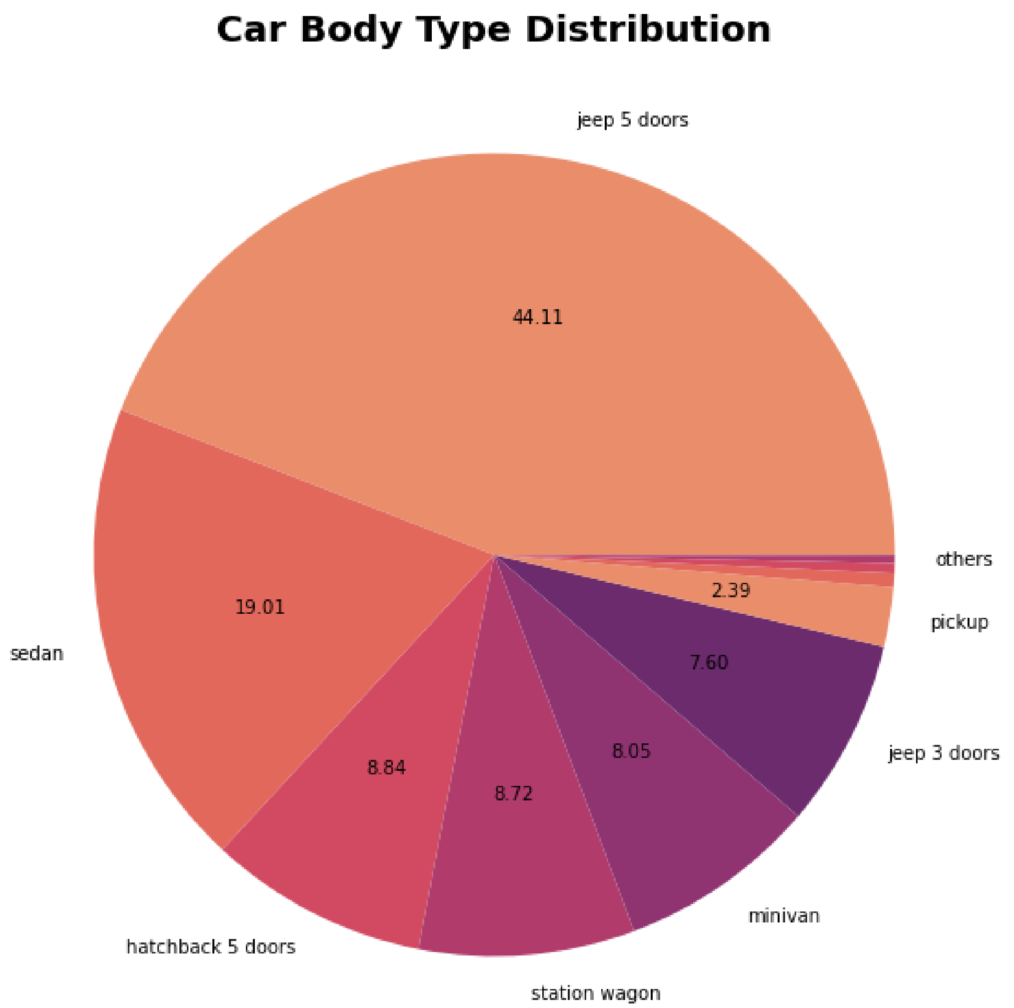
query for car body type distribution

In [30]:

```
def my_autopct(pct):
    return ('%.2f' % pct) if pct > 1 else ''
mylabels=['jeep 5 doors', 'sedan', 'hatchback 5 doors', 'station wagon',
       'minivan', 'jeep 3 doors', 'pickup', '', '', 'others', '']
df = pd.DataFrame(cars.groupby(cars['bodyType'])
                  .value_counts()
                  .unstack('bodyType')
                  .sum(axis=0,skipna=True)
                  .sort_values(ascending=False)).rename(columns={0:'count'})
plt.figure(figsize = (10,10))
plt.pie(df['count'],labels=mylabels,colors=sns.color_palette('flare'),autopct=my_autopct
)
plt.title('Car Body Type Distribution', fontsize=20, fontweight="bold")
plt.show
```

Out[30]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



It can be seen that jeep 5 doors dominates the car body type

QUERY 6

query for car color wise distribution

In [31]:

```
df = pd.DataFrame(cars['name'].groupby(cars['color'])
                  .value_counts()
                  .unstack('color')
                  .sum(axis=0,skipna=True)
                  .sort_values(ascending=False)).rename(columns={0:'count'})
df.head(10)
```

Out[31]:

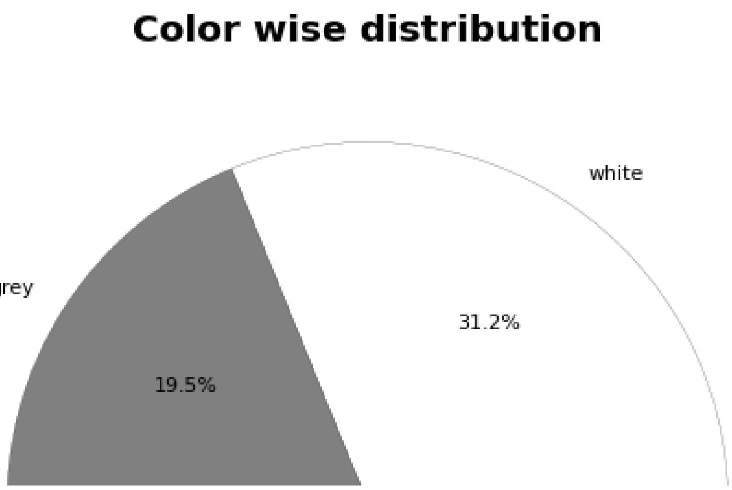
color	count
white	453154.0
grey	283428.0
black	265179.0
silver	163156.0
blue	118403.0
green	62443.0
red	42679.0
brown	25099.0
burgundy	24042.0
beige	16583.0

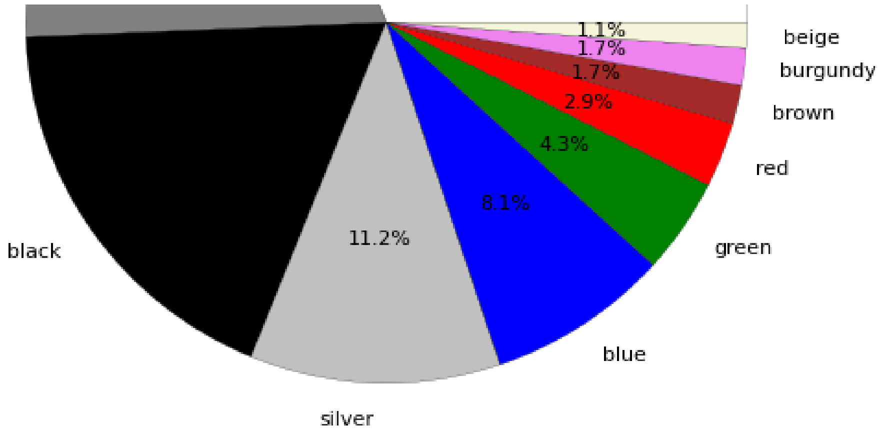
In [32]:

```
df = pd.DataFrame(cars['name'].groupby(cars['color'])
                  .value_counts()
                  .unstack('color')
                  .sum(axis=0,skipna=True)
                  .sort_values(ascending=False)).rename(columns={0:'count'})
df=df.head(10)
mycolors= ["white", "gray", "black", "silver", "blue", "green", "red", "brown", "violet"
, "beige"]
plt.figure(figsize = (9,9))
plt.pie(df['count'],labels=df.index,colors=mycolors,autopct='%.1f%%',
        wedgeprops= {"edgecolor":"black",
                     'linewidth': 0.2,
                     'antialiased': True}, textprops={'fontsize': 11})
plt.title("Color wise distribution", fontsize=20, fontweight="bold")
# plt.xlabel(df,fontsize=15)
plt.show
```

Out[32]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```





This bar chart shows the sale of cars having white color is more than any other

QUERY 7

query for fueltype categorisation according to brand

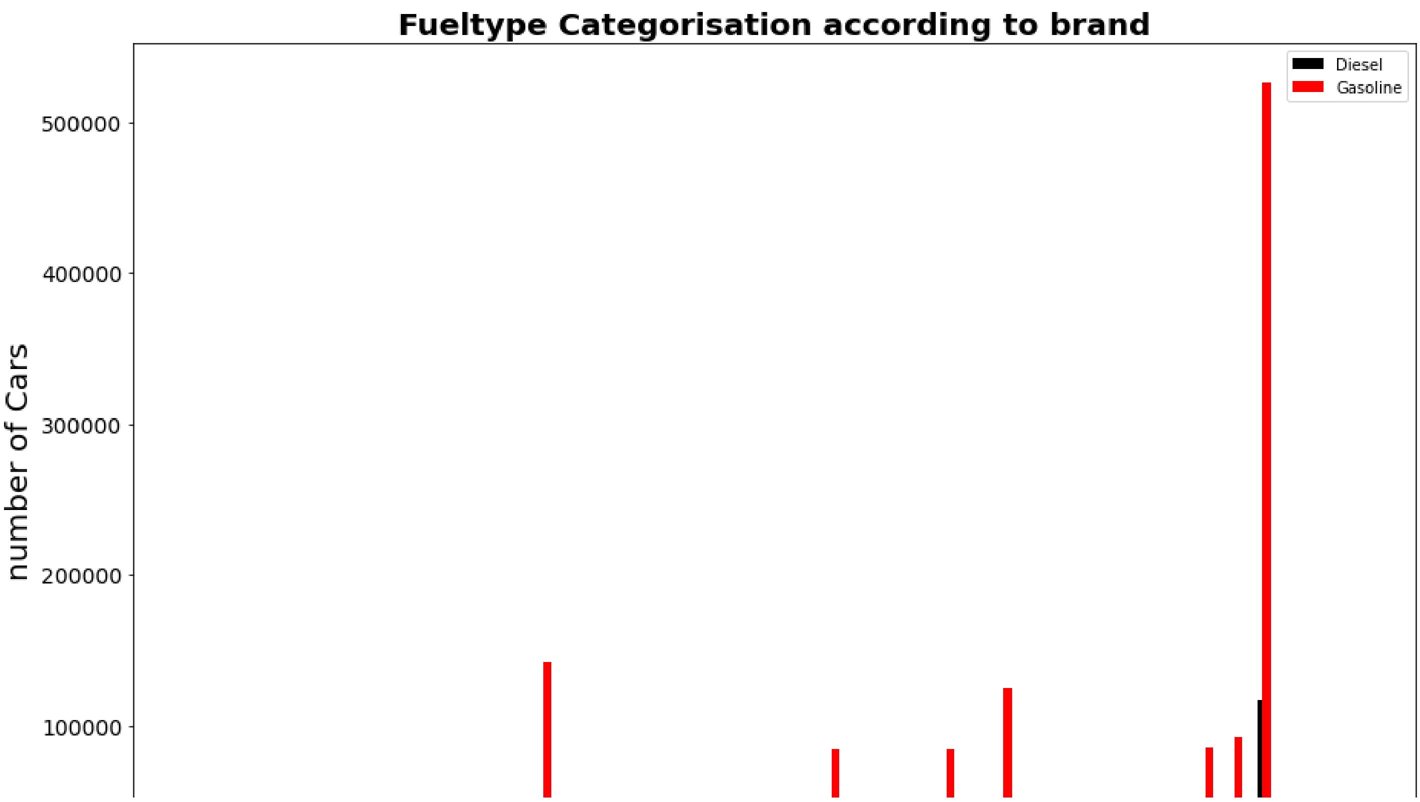
In [33]:

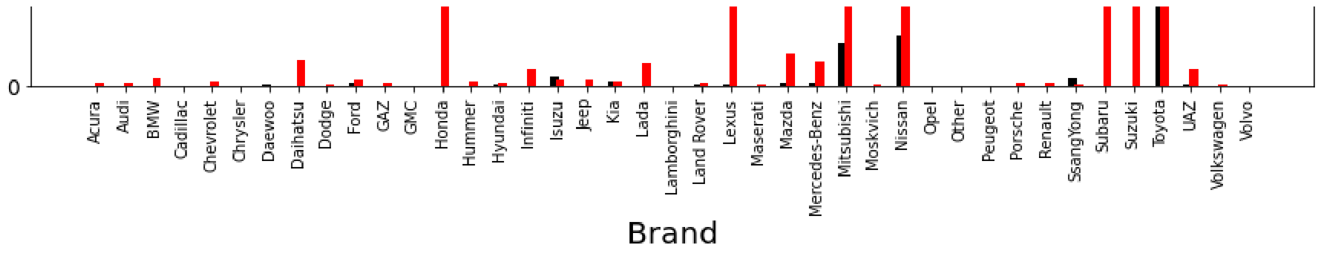
```
cdf = cars['brand'].groupby(cars['fuelType']).value_counts().unstack('fuelType')
cdf=cdf.fillna(0)
axis = np.arange(len(cdf))
plt.figure(figsize = (15,10))
plt.bar(axis-0.1, cdf['Diesel'], 0.3, label = 'Diesel', color='black')
plt.bar(axis+0.1, cdf['Gasoline'], 0.3, label = 'Gasoline', color='red')

plt.title("Fueltype Categorisation according to brand", fontsize = 20, fontweight = 'bold
')
plt.xlabel("Brand", fontsize = 20)
plt.ylabel("number of Cars",fontsize=20)
plt.xticks(axis, cdf.index,rotation=90, size = 10)
plt.yticks(size = 14)
plt.legend()
```

Out[33]:

```
<matplotlib.legend.Legend at 0x2bd355ac670>
```





it is evident that companies manufacture gasoline fueltype cars more than diesel

QUERY 8

This pie chart shows the transmission of the cars

In [34]:

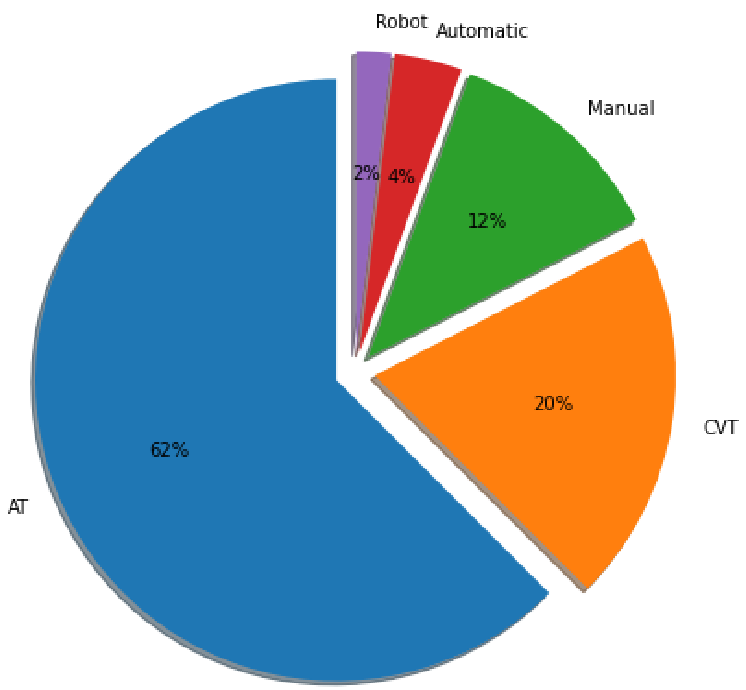
```
b = cars.transmission.value_counts()
b
```

Out[34]:

```
AT           931439
CVT          298975
Manual       178319
Automatic     54145
Robot         27814
Name: transmission, dtype: int64
```

In [35]:

```
plt.figure(figsize=(7,5))
plt.pie(b, labels = b.index,
        radius= 1.5,
        explode=[0.1,0.1,0.1, 0.1,0.1],
        startangle=90 ,shadow = True,autopct='%.0f%%')
plt.show()
```



AT transmission has the greatest share in transmissioin types followed by CVT and manual

QUERY 9

In [36]:

In [36]:

```
c = cars.color.value_counts()
c
```

Out[36]:

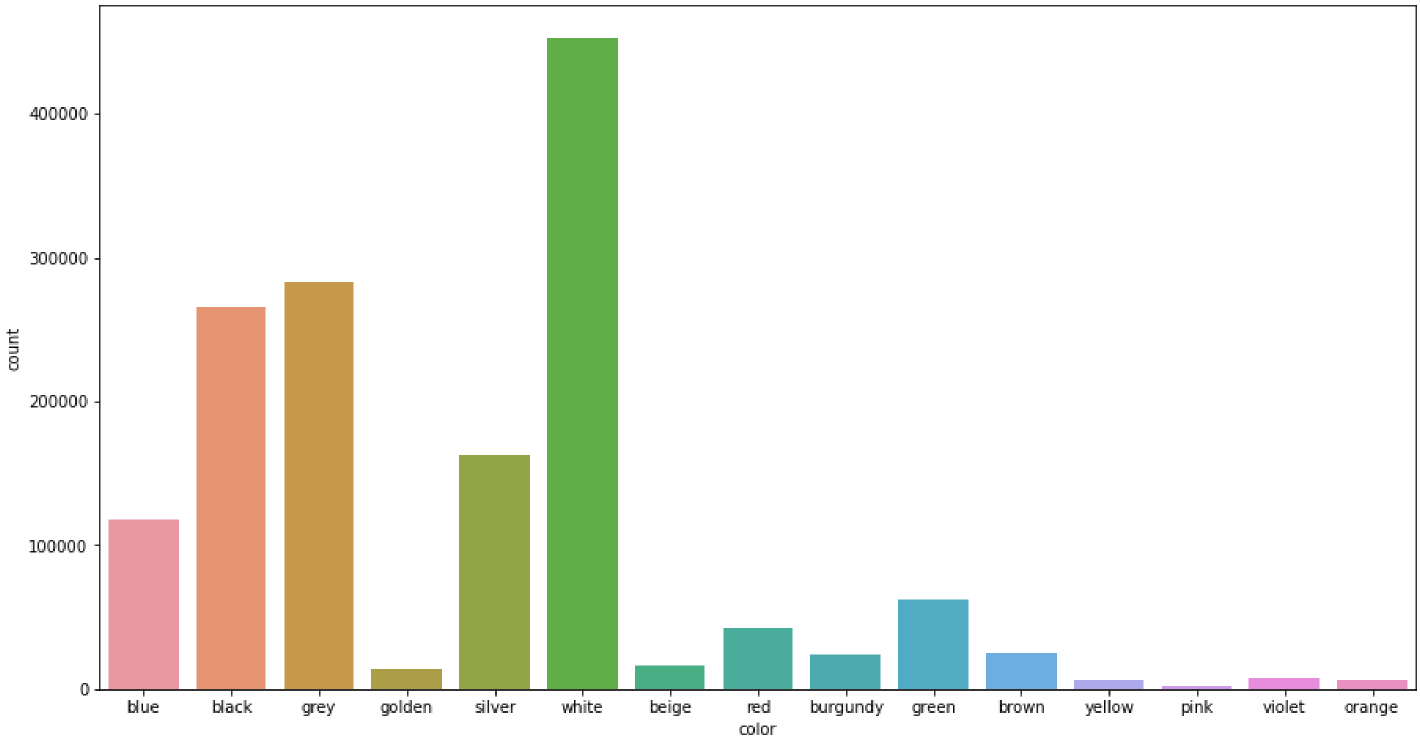
```
white       453154
grey        283428
black       265179
silver      163156
blue        118403
green        62443
red          42679
brown        25099
burgundy     24042
beige        16583
golden       14204
violet        7748
yellow        6535
orange        6232
pink          1807
Name: color, dtype: int64
```

In [37]:

```
plt.figure(figsize = (15,8))
sns.countplot(x = 'color',data = cars)
```

Out[37]:

```
<AxesSubplot:xlabel='color', ylabel='count'>
```



QUERY 10

query for car sales location

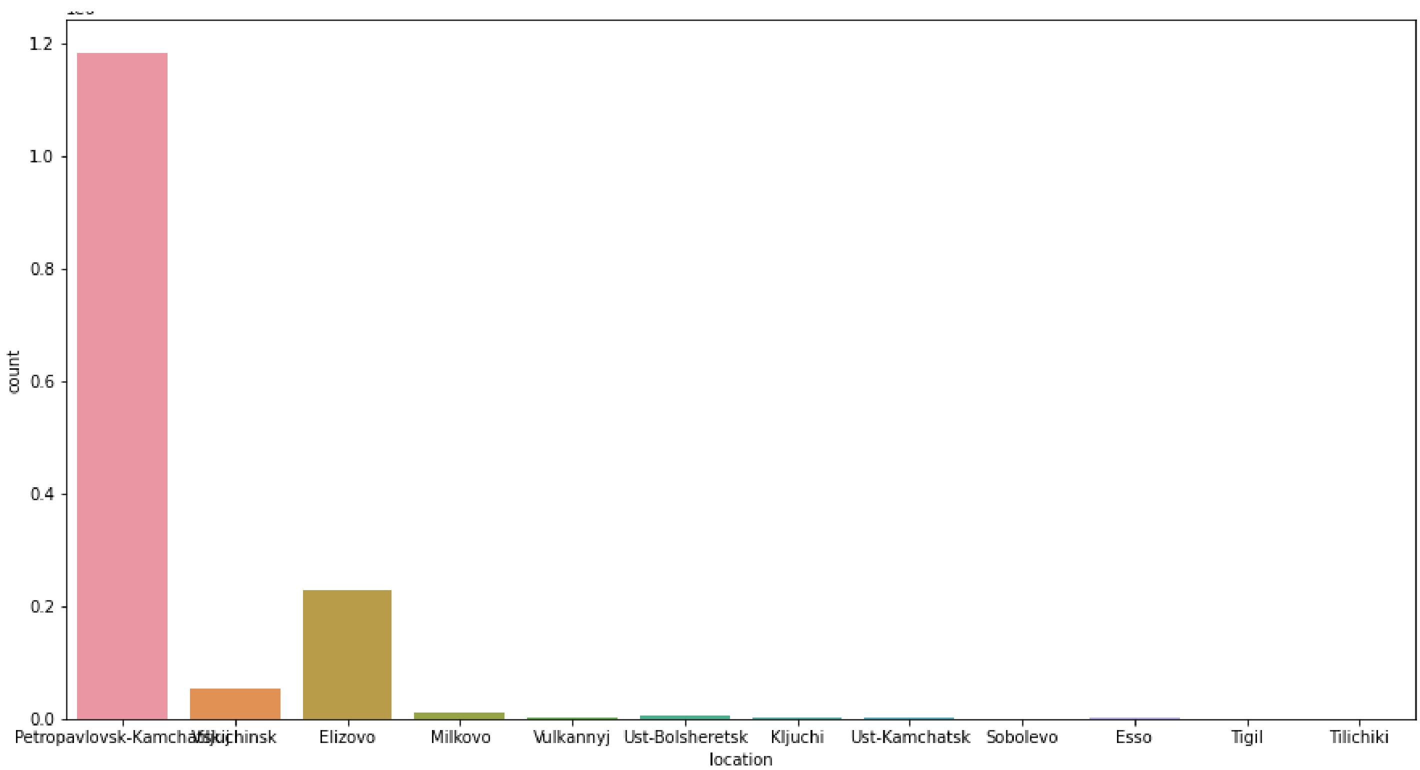
In [38]:

```
plt.figure(figsize=(15,8))
sns.countplot(x = 'location',data = cars)
```

Out[38]:

```
<AxesSubplot:xlabel='location', ylabel='count'>
```

1e6



it can be noticed that most cars are sold in 'Petropavlovsk-Kamchatskij' and 'Elizovo'

QUERY 11

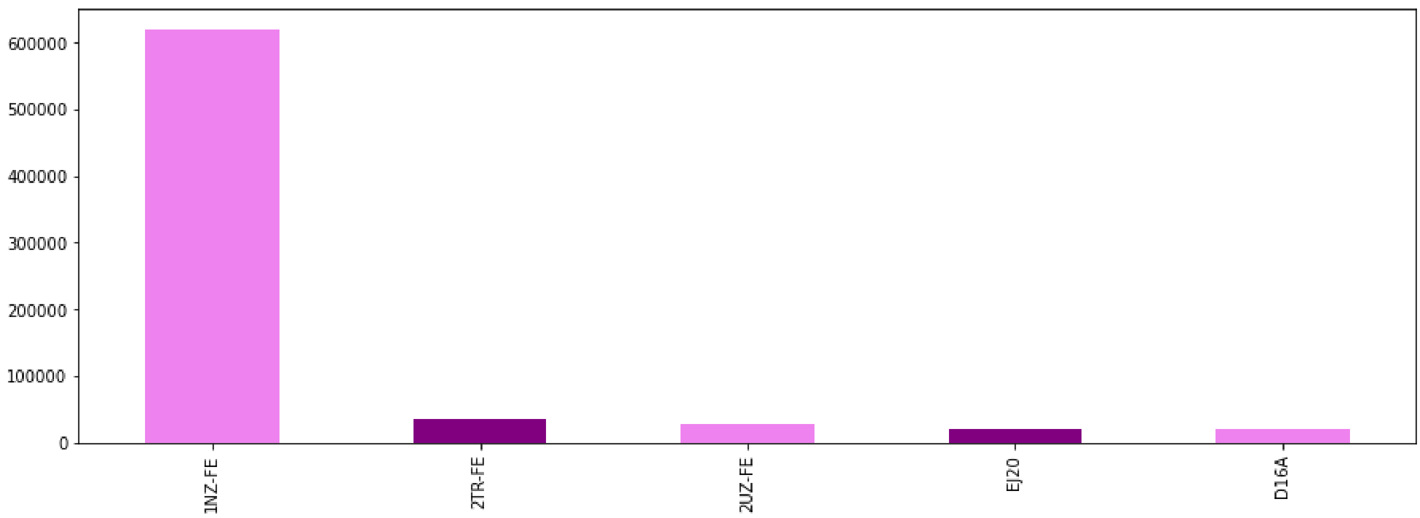
query for car sold with respect to Engune Name

In [39]:

```
cars.engineName.value_counts().sort_values(ascending = False).head().plot(kind = 'bar',f
igsize = (15,5), color=['violet','purple'])
```

Out[39]:

```
<AxesSubplot:>
```



This bar plot shows the car having engine name 1NZ-FE have highest sale

QUERY 12

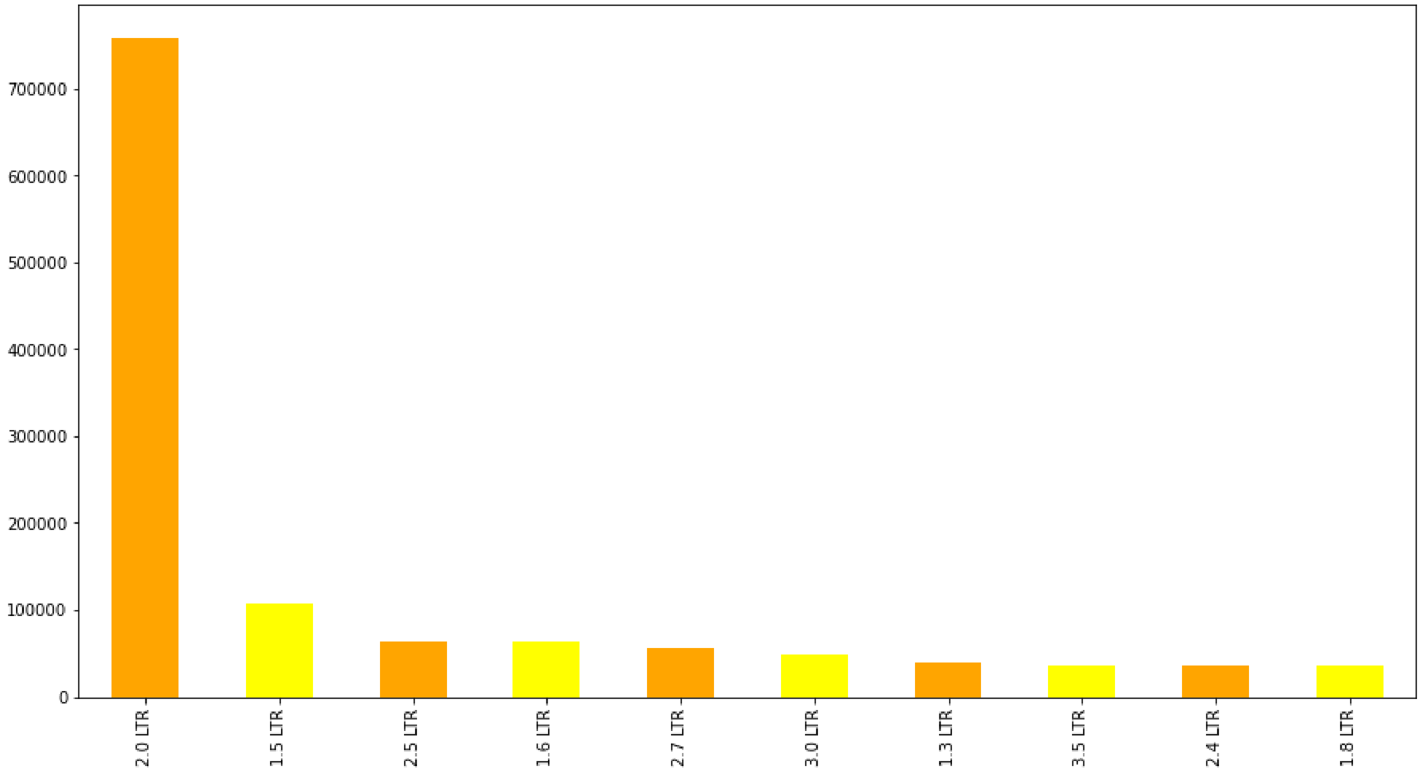
query for various engineDisplacement cars sold

In [40]:

```
cars.engineDisplacement.value_counts().sort_values(ascending = False).head(10).plot(kind
= 'bar',figsize = (15,8), color=['orange', 'yellow'])
```

Out[40]:

```
<AxesSubplot:>
```



Cars having 2.0LTR engine Displacement have more sale

QUERY 13

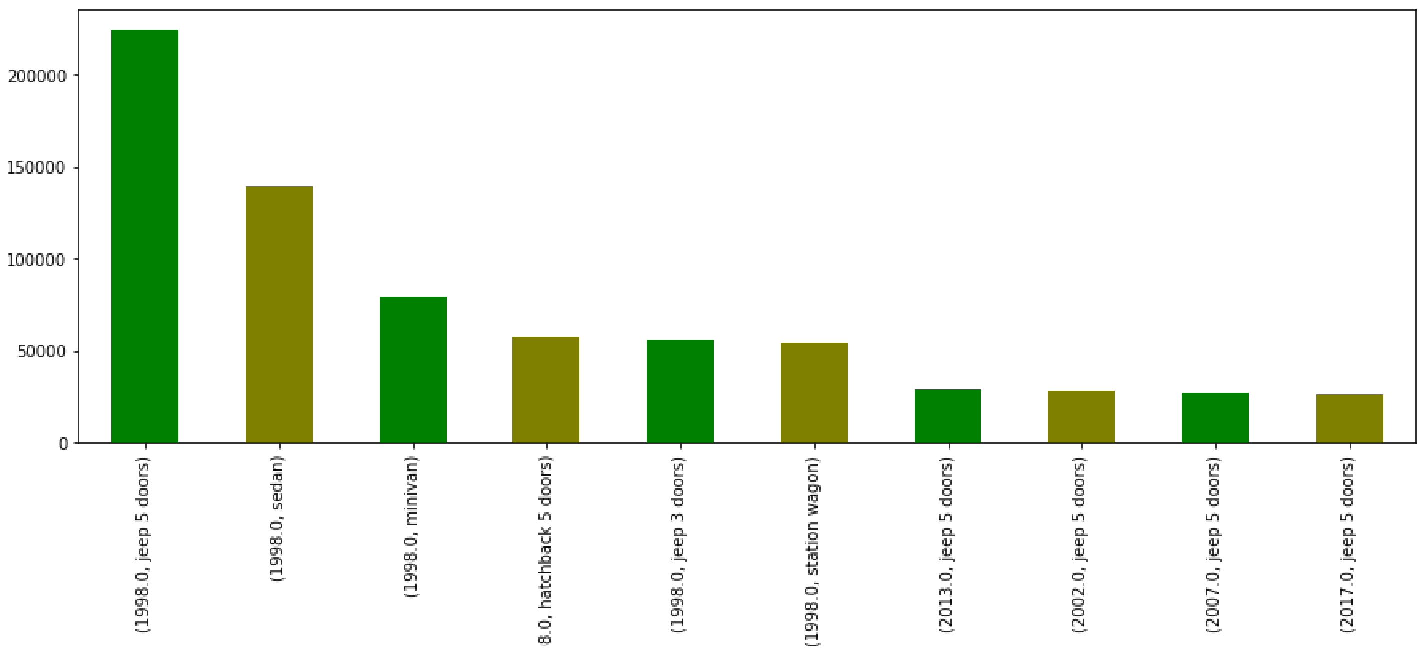
query for max cars sold and their years

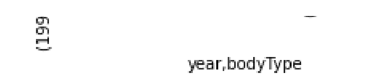
In [41]:

```
cars.groupby('year')['bodyType'].value_counts().sort_values(ascending = False).head(10).
plot(kind = 'bar',figsize = (15,5), color=['green', 'olive'])
```

Out[41]:

```
<AxesSubplot:xlabel='year,bodyType'>
```





Cars of jeep 5 doors bodytype have more sales

QUERY 14

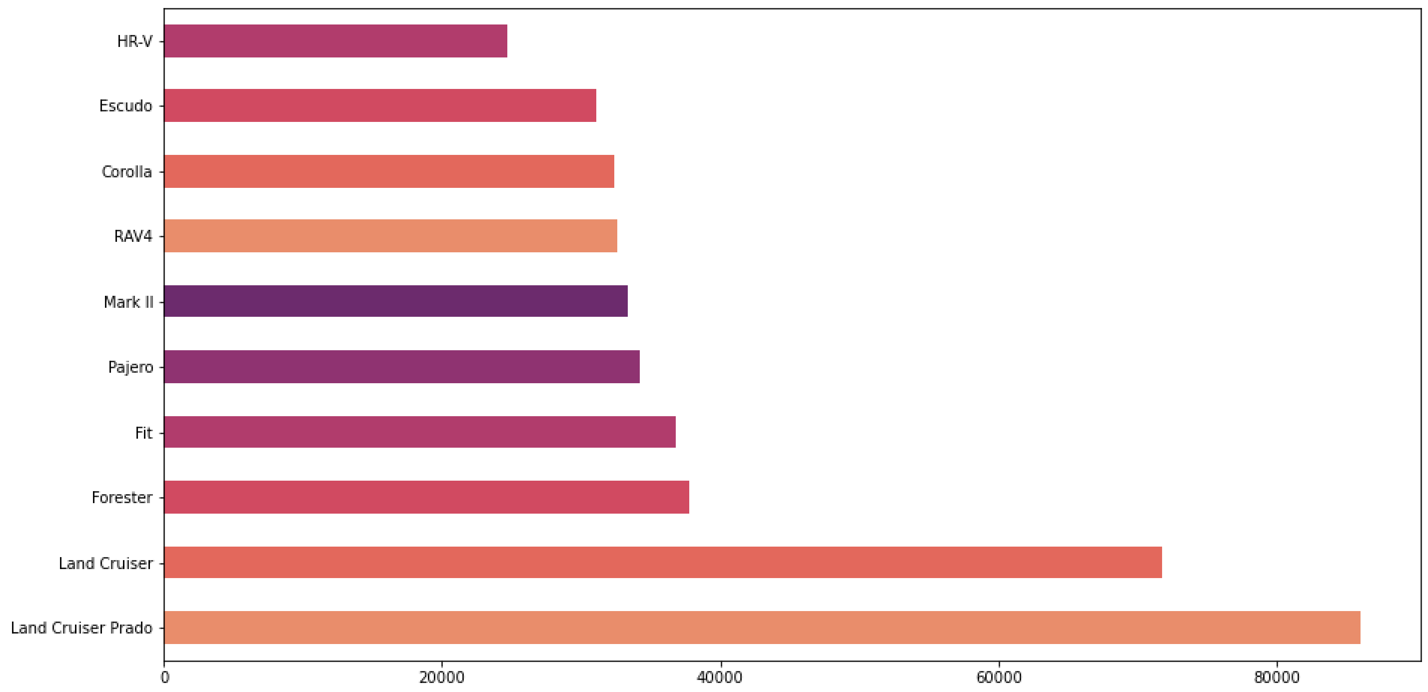
query for car sales

In [42]:

```
cars.name.value_counts().sort_values(ascending = False).head(10).plot(kind = 'barh',figs
ize = (15,8),color=sns.color_palette('flare'))
```

Out[42]:

```
<AxesSubplot:>
```



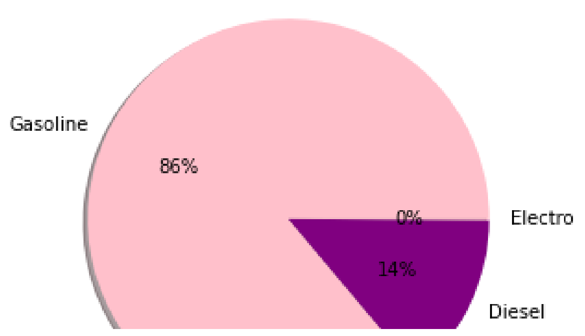
Land Cruiser Prado car have maximum Sale

QUERY 15

This pie chart shows the type of cars based on `fuelType`

In [43]:

```
a = cars.fuelType.value_counts()
plt.figure(figsize=(12,5))
plt.pie(a, labels = a.index, shadow = True,autopct='%.0f%%', colors= ['pink', 'purple'])
plt.show()
```



most cars have fueltype as Gasoline followed by diesel

QUERY 16

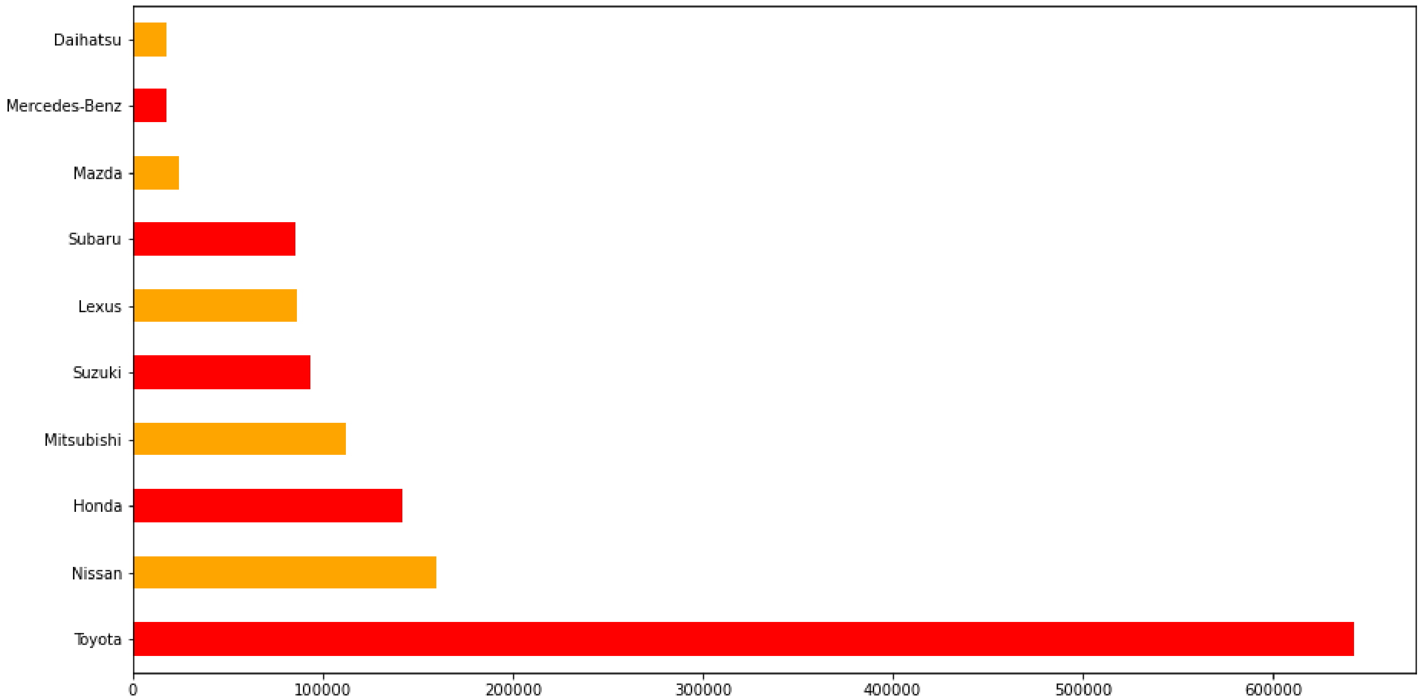
query for max cars sold with respect to brand

In [44]:

```
cars.brand.value_counts().sort_values(ascending = False).head(10).plot(kind = 'barh', co
lor=['red', 'orange'] ,figsize = (15,8))
```

Out[44]:

```
<AxesSubplot:>
```



Toyota Company Has highest sales

QUERY 17

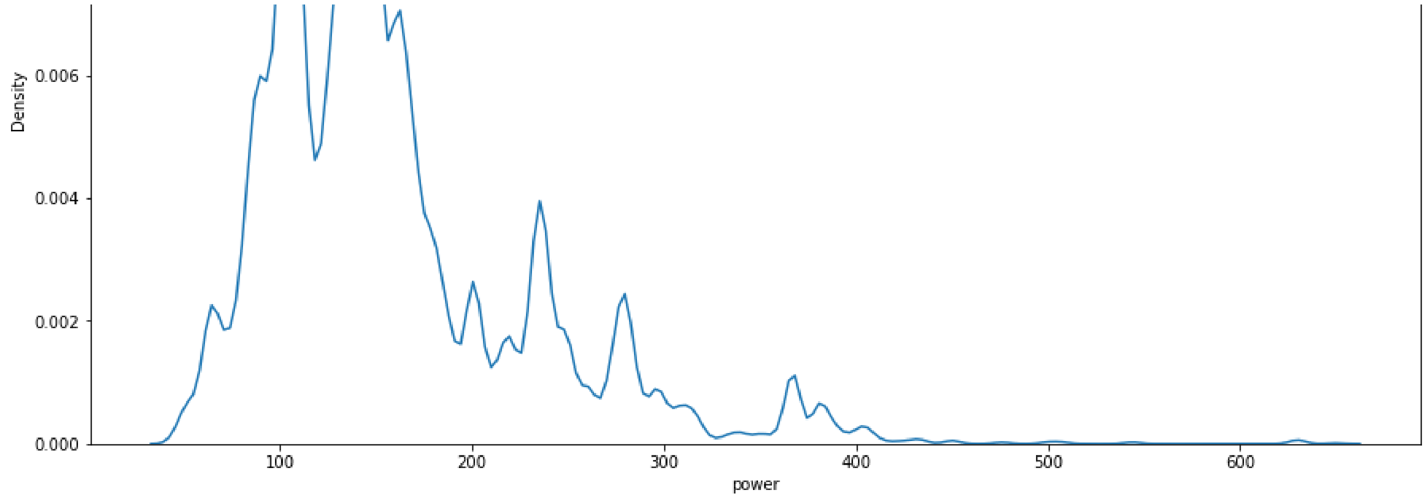
query for powers of car

In [45]:

```
sns.kdeplot(x = 'power',data = cars)
```

Out[45]:

```
<AxesSubplot:xlabel='power', ylabel='Density'>
```



Average powers of all cars are between 100 to 200

QUERY 18

Query for displaying no of cars manufactured per year(after 2010) of each brand

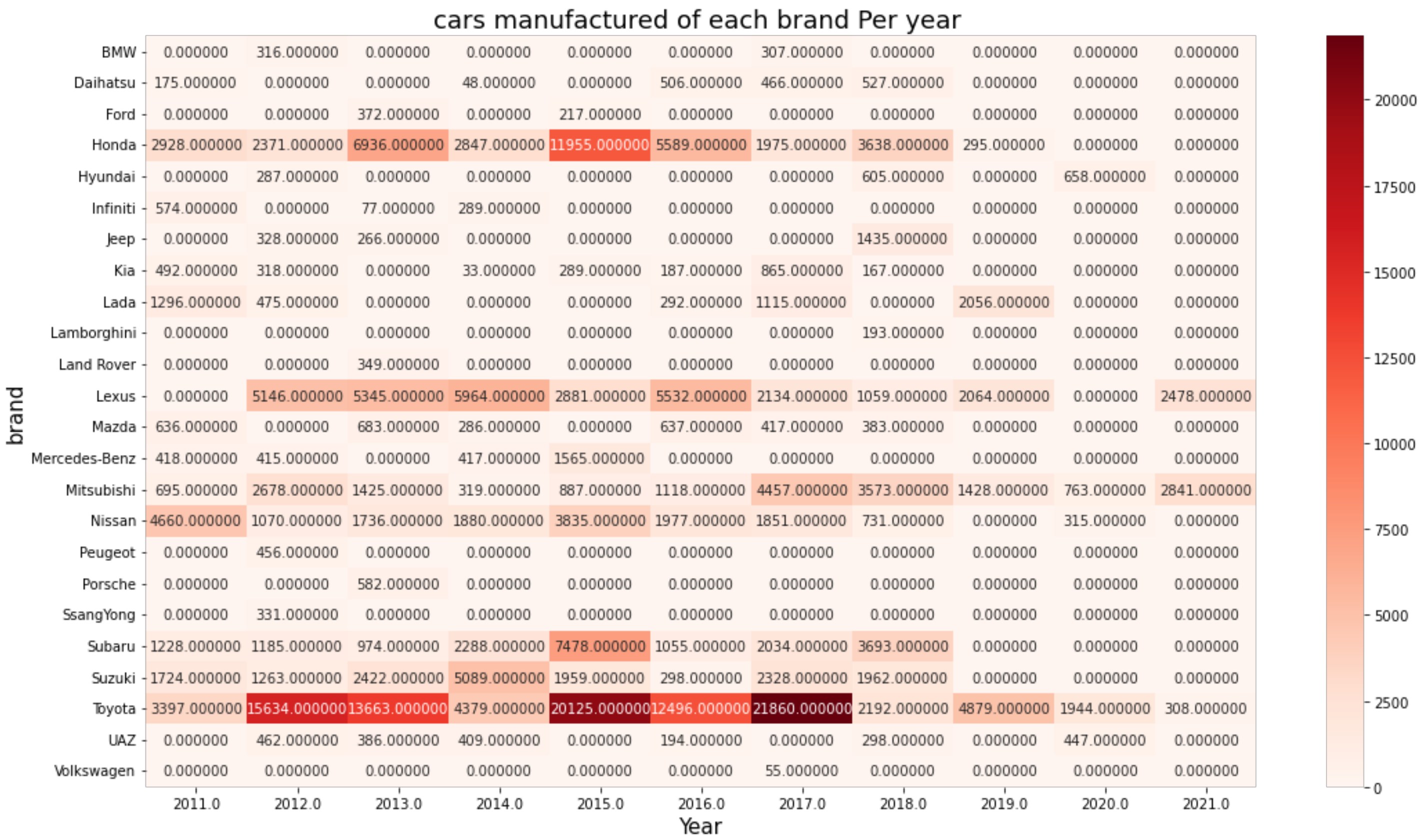
In [46]:

```
cdf = cars[cars['year']>2010]
cdf=cdf['brand'].groupby(cars['year']).value_counts().unstack('year').fillna(0)

cdf
plt.figure(figsize=(18,10))
ax = sns.heatmap(cdf, annot=True, fmt="f", cmap='Reds')
plt.ylabel('brand',fontsize = 15)
plt.xlabel('Year',fontsize = 15)
plt.title('cars manufactured of each brand Per year ',fontsize = 18)
plt.show
```

Out[46]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

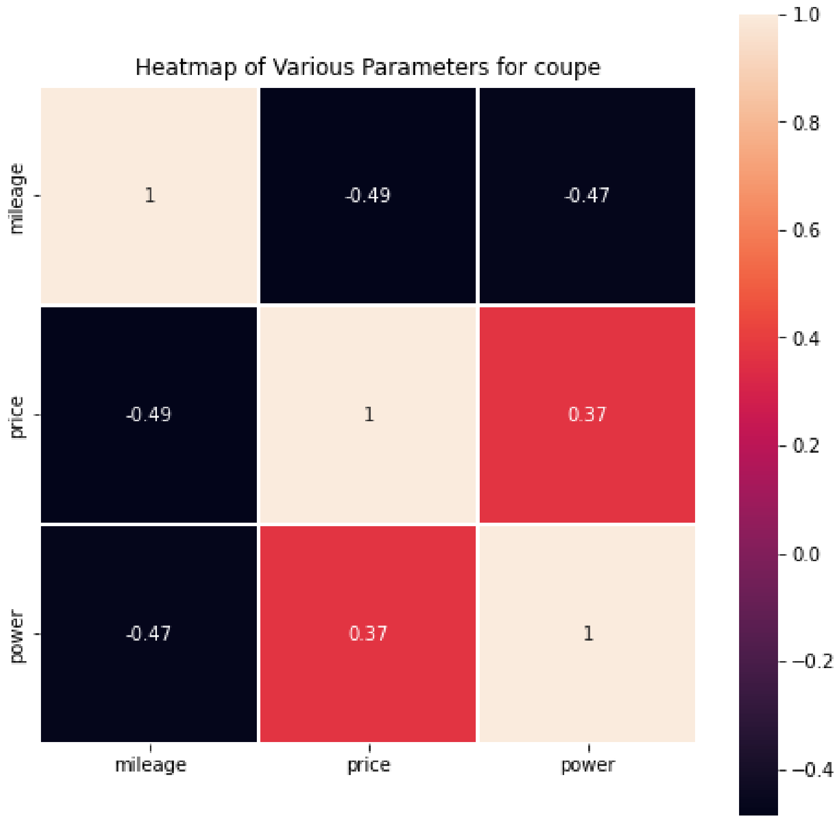


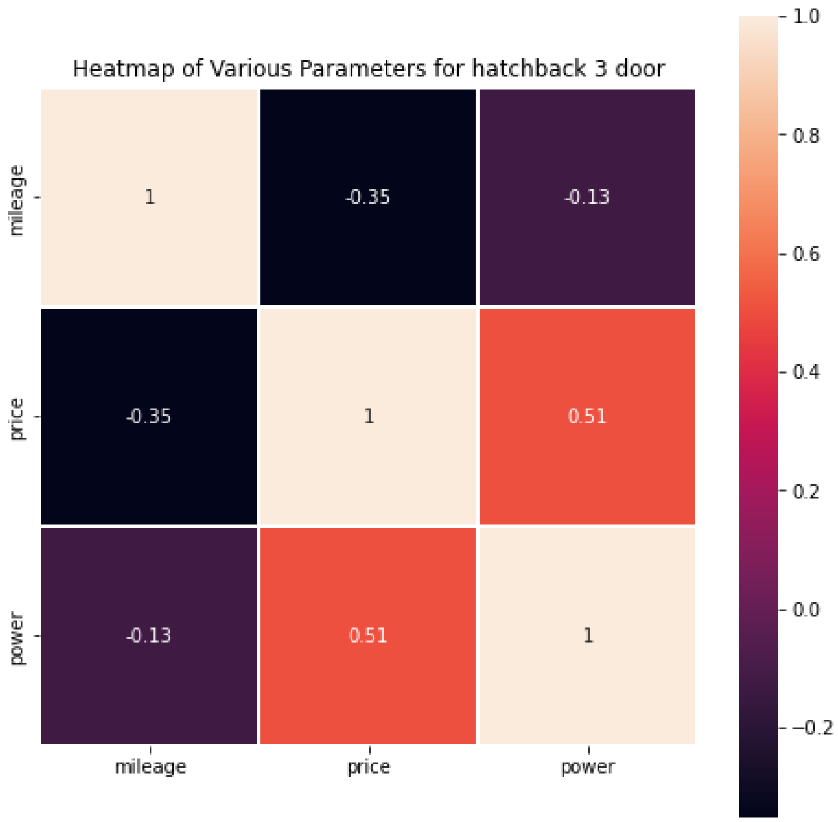
QUERY 19

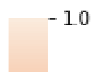
Heatmap of Various Parameters for each loaction

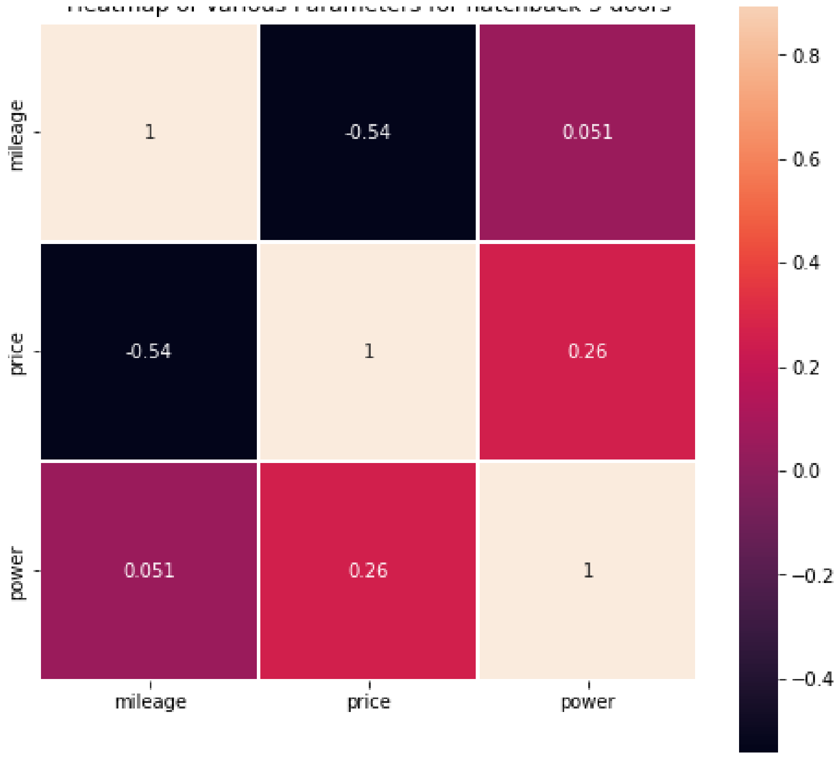
In [47]:

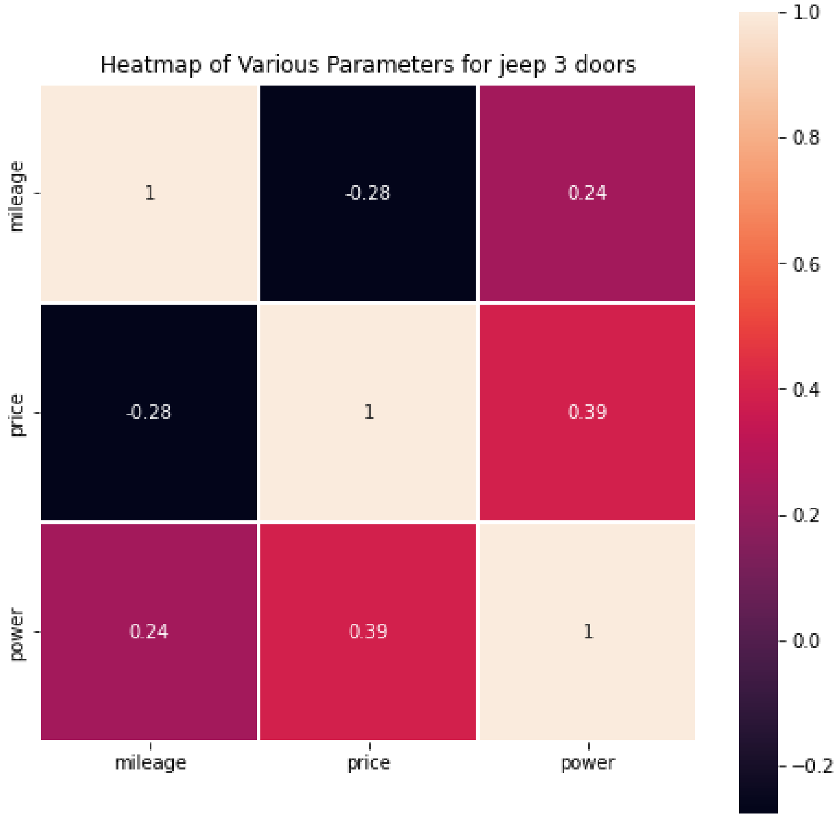
```
plt.rcParams['figure.figsize'] = [8,8]
k=['mileage','price','power','color','fuelType','bodyType']
for i,j in cars.groupby('bodyType'):
    plt.title(f"Heatmap of Various Parameters for {i}")
    sns.heatmap(j[k].corr(),annot=True,square=True,linewidth=0.2)
    plt.show()
```

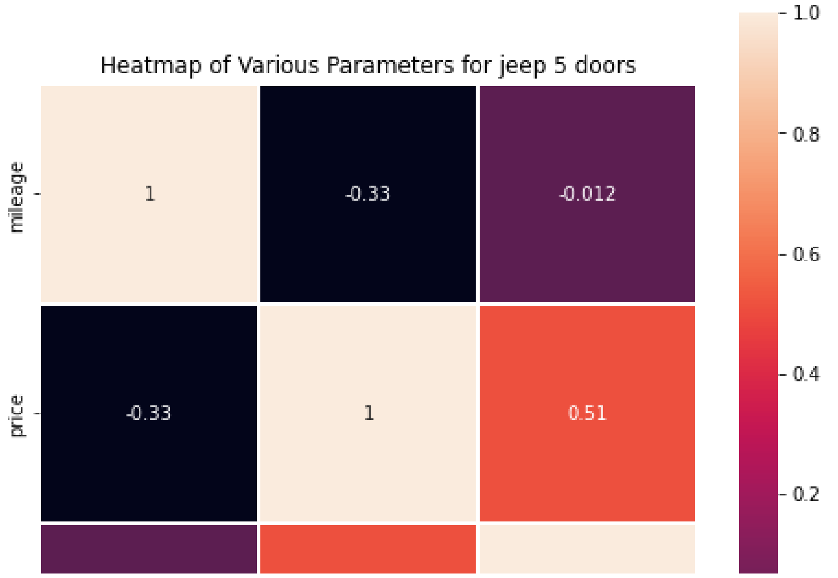




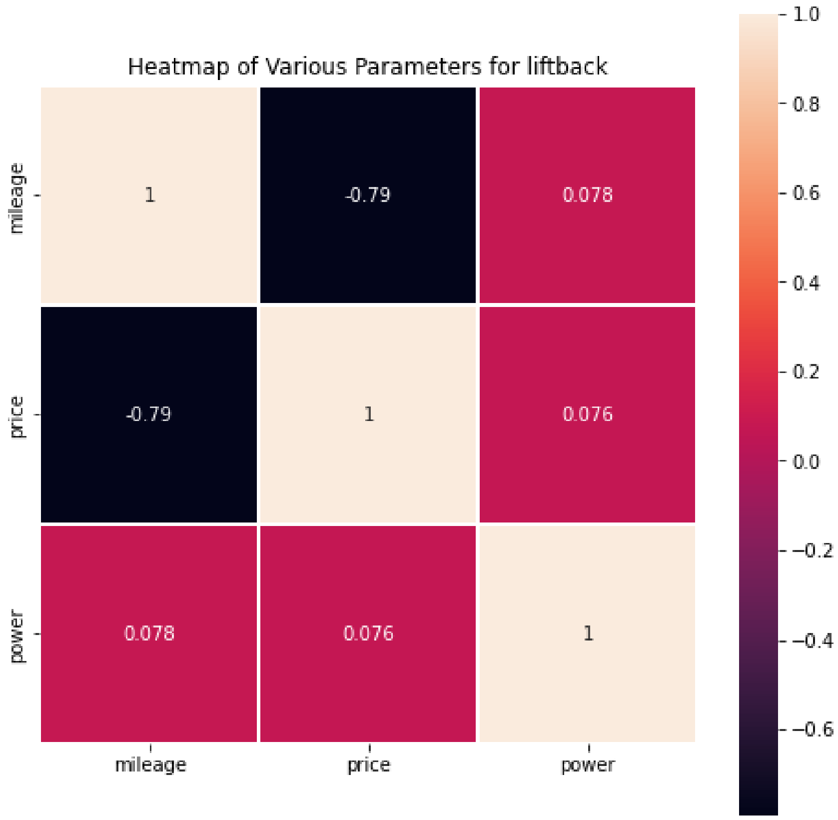



mileage
price
power
1
-0.54
0.051
-0.54
1
0.26
0.051
0.26
1
mileage
price
power
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

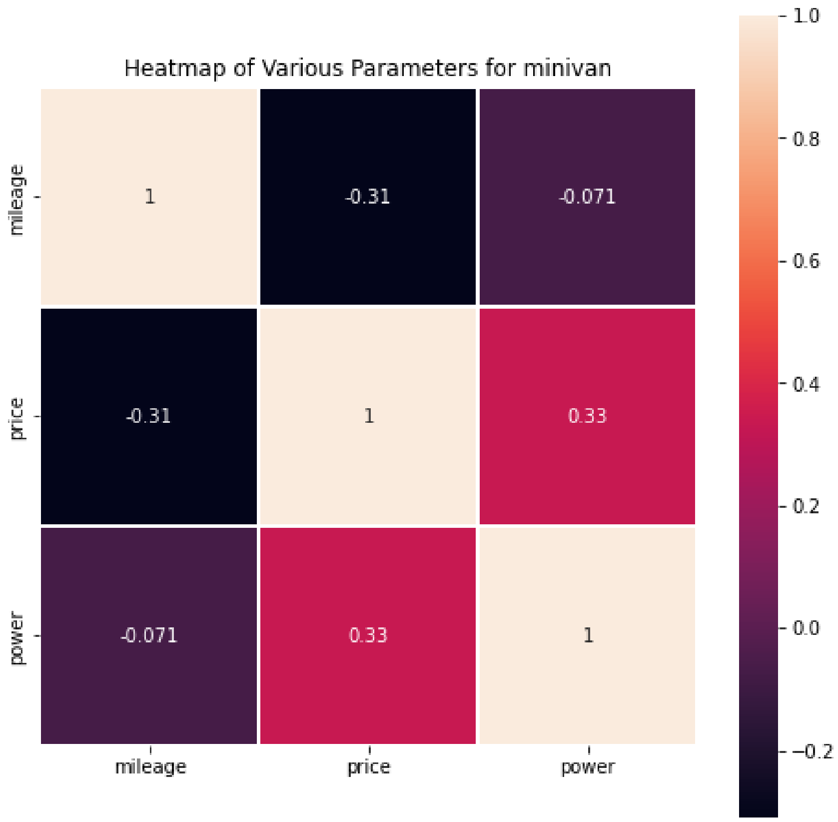

Heatmap of Various Parameters for jeep 3 doors
mileage
price
power
1
-0.28
0.24
-0.28
1
0.39
0.24
0.39
1
mileage
price
power
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


Heatmap of Various Parameters for jeep 5 doors
mileage
price
1
-0.33
-0.012
-0.33
1
0.51
1.0
0.8
0.6
0.4
0.2

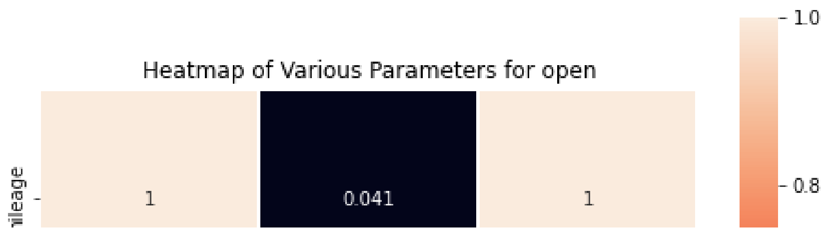
power
-0.012
0.51
1
mileage
price
power
0.0
−0.2



Heatmap of Various Parameters for liftback
mileage
price
power
1
-0.79
0.078
-0.79
1
0.076
0.078
0.076
1
mileage
price
power
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

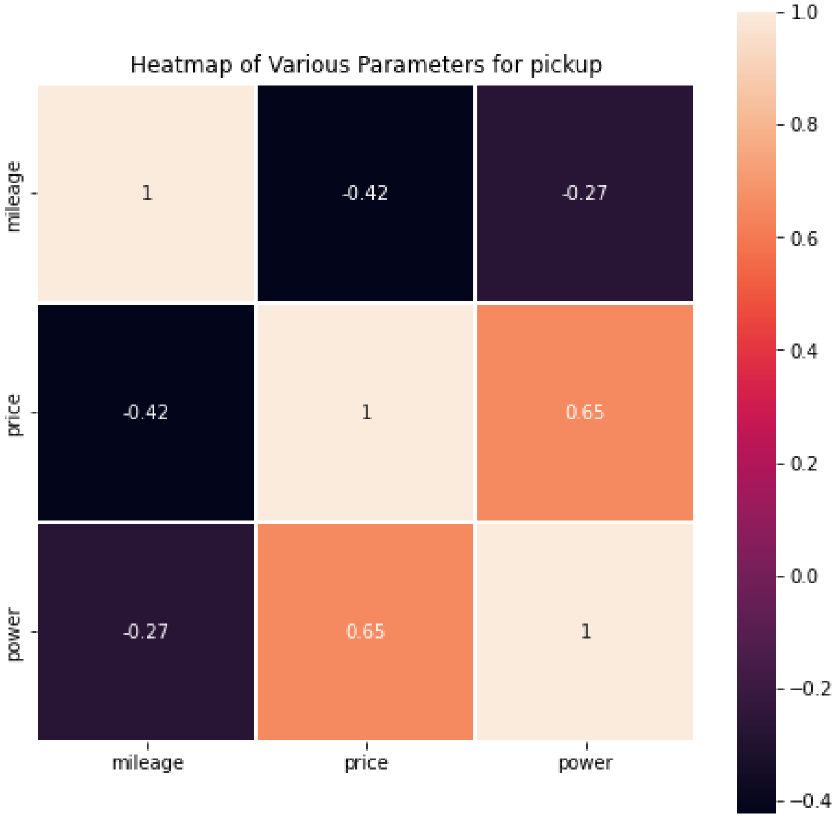


Heatmap of Various Parameters for minivan
mileage
price
power
1
-0.31
-0.071
-0.31
1
0.33
-0.071
0.33
1
mileage
price
power
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

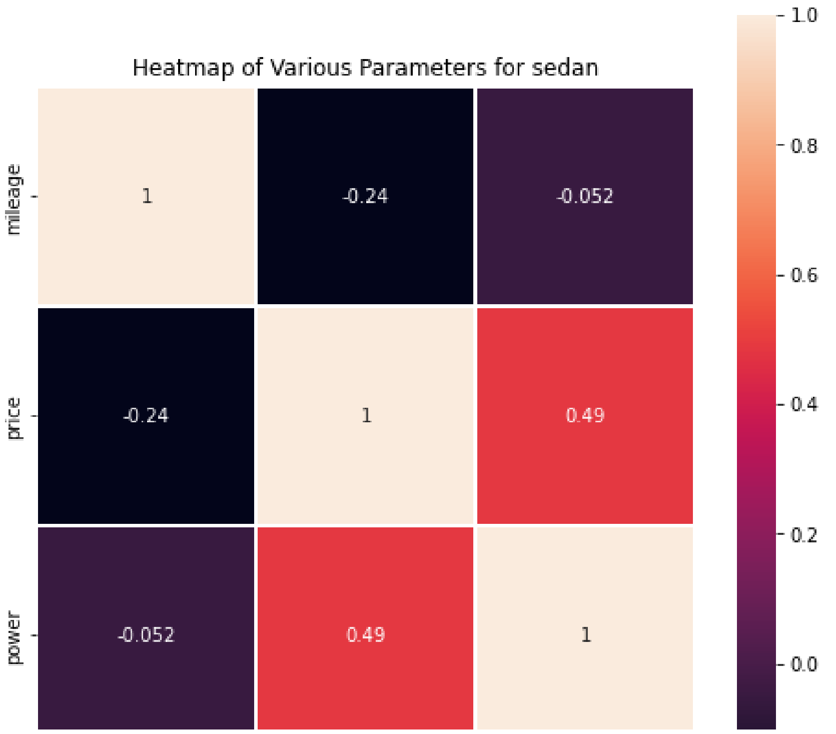


Heatmap of Various Parameters for open
1
0.041
1
1.0
0.8

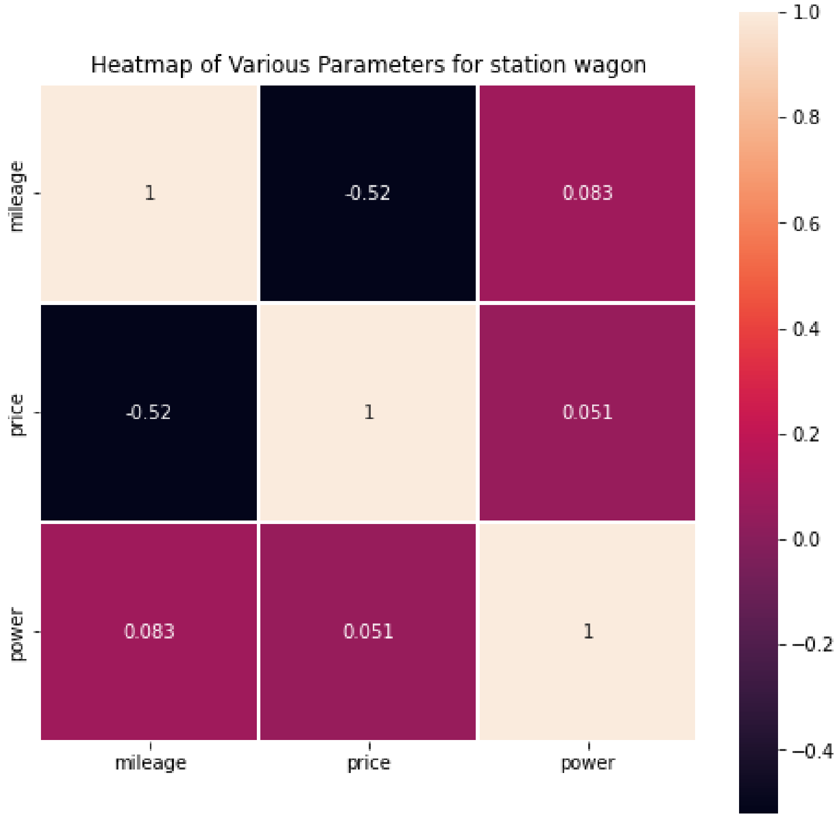
price
power
0.041
1
0.041
1
0.041
1
mileage
price
power
0.6
0.4
0.2



Heatmap of Various Parameters for pickup
mileage
price
power
1
-0.42
-0.27
-0.42
1
0.65
-0.27
0.65
1
mileage
price
power
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4



Heatmap of Various Parameters for sedan
mileage
price
power
1
-0.24
-0.052
-0.24
1
0.49
-0.052
0.49
1
1.0
0.8
0.6
0.4
0.2
0.0



QUERY 20

Correlation Matrix

In [48]:

```
corr = cars.loc[:,[ 'price','power', 'year']].corr()
corr
```

Out[48]:

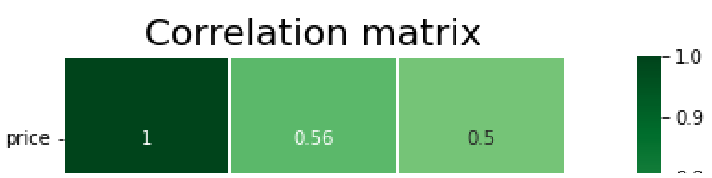
	price	power	year
price	1.000000	0.559953	0.498009
power	0.559953	1.000000	0.176061
year	0.498009	0.176061	1.000000

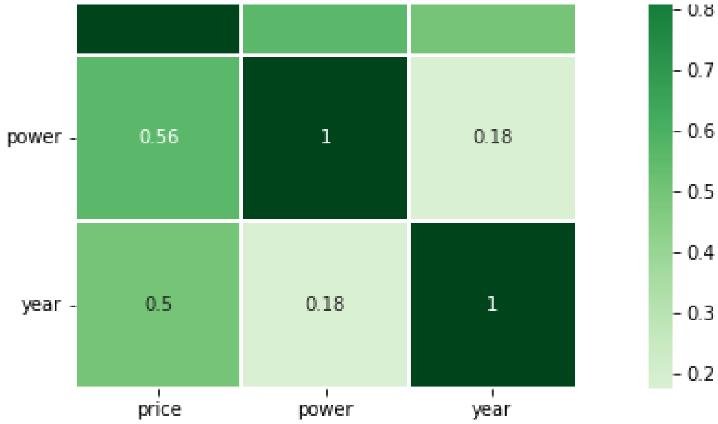
In [49]:

```
fig = plt.figure(figsize=(14, 5))
ax1 = plt.subplot()
ax1 = sns.heatmap(corr, annot=True , cmap='Greens', vmax=1, center=0.5, square=True, lin
ewidths=2)
ax1.set_title('Correlation matrix', fontsize=20)
plt.yticks(rotation=0)
```

Out[49]:

```
(array([0.5, 1.5, 2.5]),
 [Text(0, 0.5, 'price'), Text(0, 1.5, 'power'), Text(0, 2.5, 'year')])
```





`price` and `power` increase with increase in `year` of manufacture

Conclusion

In the Given dataset we explored and learnt many things.

- **Average powers of all cars are between 100 to 200**
- **Land Cruiser Prado car have maximum Sale**
- **Toyota Company has highest sales and revenue**
- **Cars of 2.0LTR engine type have more sales**
- **White color cars are most common**
- **Most cars have fueltype as Gasoline followed by diesel**
- `price` **and** `power` **increase with increase in** `year` **of manufacture**
- **AT transmission has the greatest share in transmissioin types followed by CVT and manual**
- **Cars of jeep 5 doors bodytype have more sales**
- **Cars having 2.0LTR engine Displacement have more sale**
- **car having engine name 1NZ-FE have highest sale**