Fuel Consumption

About Data

Datasets provide model-specific fuel consumption ratings and estimated carbon dioxide emissions for new light-duty vehicles for retail sale in Canada.

Model:

- 4WD/4X4 = Four-wheel drive
- AWD = All-wheel drive
- FFV = Flexible-fuel vehicle
- SWB = Short wheelbase
- LWB = Long wheelbase
- EWB = Extended wheelbase

Transmission:

- A = automatic
- AM = automated manual
- AS = automatic with select shift
- AV = continuously variable
- M = manual

- 3 – 10 = Number of gears

Fuel type:

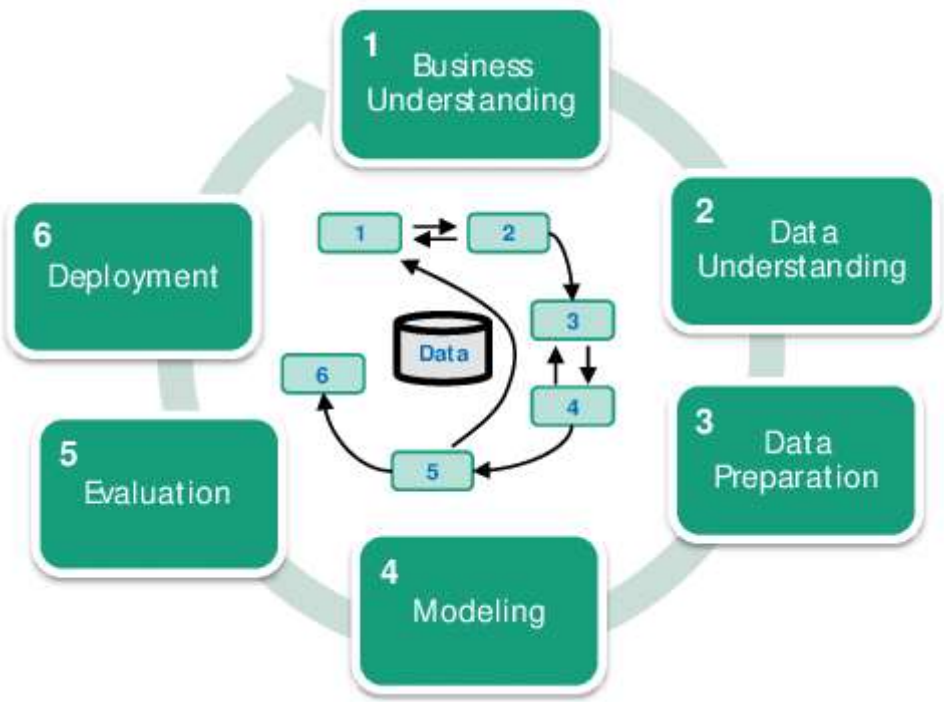
- X = regular gasoline
- Z = premium gasoline
- D = diesel
- E = ethanol (E85)
- N = natural gas

Fuel consumption: City and highway fuel consumption ratings are shown in litres per 100 kilometres (L/100 km) - the combined rating (55% city, 45% hwy) is shown in L/100 km and in miles per imperial gallon (mpg)

CO2 emissions: the tailpipe emissions of carbon dioxide (in grams per kilometre) for combined city and highway driving

CO2 rating: the tailpipe emissions of carbon dioxide rated on a scale from 1 (worst) to 10 (best)

Smog rating: the tailpipe emissions of smog-forming pollutants rated on a scale from 1 (worst) to 10 (best)



Content :

1.Busniess Understanding

2.Data Understanding

3.Data Prepartion

4.Modelling

5.Evaluation

Busniess Understanding :

We have to find the fuel consumption of the vehciles based on the features

Data Understanding :

2.1 Importing Libiraries

In [1]:
```
# To satitical operation
import pandas as pd
import numpy as np

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
```

2.2 Load Data

In [2]:
```
df=pd.read_csv("MY2022 Fuel Consumption Ratings.csv")
df.head(3)
```

Out[2]:

	Model Year	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fuel Consumption (City (L/100 km)	Consump (L/
0	2022	Acura	ILX	Compact	2.4	4	AM8	Z	9.9	
1	2022	Acura	MDX SH-AWD	SUV: Small	3.5	6	AS10	Z	12.6	
2	2022	Acura	RDX SH-AWD	SUV: Small	2.0	4	AS10	Z	11.0	

2.3 EDA

In [3]:
```
df.columns
```

Out[3]:
```
Index(['Model Year', 'Make', 'Model', 'Vehicle Class', 'Engine Size(L)',
       'Cylinders', 'Transmission', 'Fuel Type',
       'Fuel Consumption (City (L/100 km)', 'Fuel Consumption(Hwy (L/100 km))',
       'Fuel Consumption(Comb (L/100 km))', 'Fuel Consumption(Comb (mpg))',
       'CO2 Emissions(g/km)', 'CO2 Rating', 'Smog Rating'],
      dtype='object')
```

In [4]:
```
df.rename(columns={"Fuel Consumption (City (L/100 km)":"city"},inplace=True)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Model Year', 'Make', 'Model', 'Vehicle Class', 'Engine Size(L)',
       'Cylinders', 'Transmission', 'Fuel Type', 'city',
       'Fuel Consumption(Hwy (L/100 km))', 'Fuel Consumption(Comb (L/100 km))',
       'Fuel Consumption(Comb (mpg))', 'CO2 Emissions(g/km)', 'CO2 Rating',
       'Smog Rating'],
      dtype='object')
```

In [6]:
```
# Checking the dimension
df.shape
```

Out[6]:
```
(946, 15)
```

946 Records and 15 Features

In [7]:
```
# Listing the Total features
df.columns
```

Out[7]:
```
Index(['Model Year', 'Make', 'Model', 'Vehicle Class', 'Engine Size(L)',
       'Cylinders', 'Transmission', 'Fuel Type', 'city',
       'Fuel Consumption(Hwy (L/100 km))', 'Fuel Consumption(Comb (L/100 km))',
       'Fuel Consumption(Comb (mpg))', 'CO2 Emissions(g/km)', 'CO2 Rating',
       'Smog Rating'],
      dtype='object')
```

In [8]:
```
df.nunique()
```

Out[8]:
```
Model Year                          1
Make                               39
Model                             715
Vehicle Class                      14
Engine Size(L)                     36
Cylinders                           8
Transmission                       23
Fuel Type                           4
city                              149
Fuel Consumption(Hwy (L/100 km))  107
Fuel Consumption(Comb (L/100 km)) 131
Fuel Consumption(Comb (mpg))       49
CO2 Emissions(g/km)               242
CO2 Rating                         10
Smog Rating                         5
dtype: int64
```

No of Unique Values in the each feature are listed above

In [9]:
```
# For Unique Values in the Features
print("The Total Unique values in fetaures")
print(df.nunique())
for i in df.columns:
    x=df[i].unique()
    print("The Title :{}".format(i))
    print("Unique values :{}",format(x))
    print()
    print("*"*90)
    print()
```

```
 'X7 M50i' 'Z4 sDrive30i' 'Z4 M40i' 'Chiron' 'Chiron Pur Sport'
 'Chiron Super Sport' 'Enclave' 'Enclave AWD' 'Encore' 'Encore AWD'
 'Encore GX' 'Encore GX AWD' 'Envision' 'Envision AWD' 'CT4' 'CT4 AWD'

 'CT4-V' 'CT4-V AWD' 'CT4-V Blackwing' 'CT5' 'CT5 AWD' 'CT5-V' 'CT5-V AWD'
 'CT5-V Blackwing' 'Escalade 4WD' 'Escalade 4WD (No Stop-Start)' 'XT4'
 'XT4 AWD' 'XT5' 'XT5 AWD' 'XT6 AWD' 'Blazer' 'Blazer AWD' 'Camaro'
 'Camaro SS' 'Camaro ZL1' 'Colorado' 'Colorado 4WD' 'Colorado ZR2 4WD'
 'Corvette' 'Equinox' 'Equinox AWD' 'Malibu' 'Silverado' 'Silverado FFV'
 'Silverado 4WD' 'Silverado 4WD Mud Terrain Tire'
 'Silverado 4WD (With Sport Mode)' 'Silverado 4WD FFV'
 'Silverado 4WD Mud Terrain Tire FFV' 'Silverado 4WD (No DFM)'
 'Silverado 4WD (No Stop-Start)'
 'Silverado 4WD Mud Terrain Tire (No Stop-Start)'
 'Silverado 4WD Mud Terrain Tire (No DFM)'
 'Silverado 4WD Custom Trail Boss' 'Silverado 4WD ZR2' 'Spark' 'Suburban'
 'Suburban (No Stop-Start)' 'Suburban 4WD' 'Suburban 4WD (No Stop-Start)'
 'Tahoe' 'Tahoe (No Stop-Start)' 'Tahoe 4WD' 'Tahoe 4WD (No Stop-Start)'
 'Trailblazer' 'Trailblazer AWD' 'Traverse' 'Traverse AWD' 'Trax'
 'Trax AWD' '300' '300 AWD' 'Grand Caravan' 'Pacifica' 'Pacifica AWD'
```

In [10]:
```
# Checking data types of features
df.dtypes
```

Out[10]:
```
Model Year                          int64
Make                               object
Model                              object
Vehicle Class                      object
Engine Size(L)                    float64
Cylinders                           int64
Transmission                       object
Fuel Type                          object
city                              float64
Fuel Consumption(Hwy (L/100 km))  float64
Fuel Consumption(Comb (L/100 km))  float64
Fuel Consumption(Comb (mpg))        int64
CO2 Emissions(g/km)                 int64
CO2 Rating                          int64
Smog Rating                         int64
dtype: object
```

We have 5 Categorial Features in our dataset

In [11]:
```
# Check For Null values
df.isnull().sum()
```

Out[11]:
```
Model Year                          0
Make                                0
Model                               0
Vehicle Class                       0
Engine Size(L)                      0
Cylinders                           0
Transmission                        0
Fuel Type                           0
city                                0
Fuel Consumption(Hwy (L/100 km))    0
Fuel Consumption(Comb (L/100 km))   0
Fuel Consumption(Comb (mpg))        0
CO2 Emissions(g/km)                 0
CO2 Rating                          0
Smog Rating                         0
dtype: int64
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 946 entries, 0 to 945
Data columns (total 15 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   Model Year                         946 non-null    int64  
 1   Make                               946 non-null    object 
 2   Model                              946 non-null    object 
 3   Vehicle Class                      946 non-null    object 
 4   Engine Size(L)                     946 non-null    float64
 5   Cylinders                          946 non-null    int64  
 6   Transmission                       946 non-null    object 
 7   Fuel Type                          946 non-null    object 
 8   city                               946 non-null    float64
 9   Fuel Consumption(Hwy (L/100 km))   946 non-null    float64
 10  Fuel Consumption(Comb (L/100 km))  946 non-null    float64
 11  Fuel Consumption(Comb (mpg))       946 non-null    int64  
 12  CO2 Emissions(g/km)                946 non-null    int64  
 13  CO2 Rating                         946 non-null    int64  
 14  Smog Rating                        946 non-null    int64  
dtypes: float64(4), int64(6), object(5)
memory usage: 111.0+ KB
```

In [13]:
```
df.describe()
```

Out[13]:

	Model Year	Engine Size(L)	Cylinders	city	Fuel Consumption(Hwy (L/100 km))	Fuel Consumption(Comb (L/100 km))	Consum
count	946.0	946.000000	946.000000	946.000000	946.000000	946.000000	
mean	2022.0	3.198732	5.668076	12.506448	9.363319	11.092072	
std	0.0	1.374814	1.932670	3.452043	2.285125	2.876276	
min	2022.0	1.200000	3.000000	4.000000	3.900000	4.000000	
25%	2022.0	2.000000	4.000000	10.200000	7.700000	9.100000	
50%	2022.0	3.000000	6.000000	12.200000	9.200000	10.800000	
75%	2022.0	3.800000	6.000000	14.700000	10.700000	12.900000	
max	2022.0	8.000000	16.000000	30.300000	20.900000	26.100000	

In [14]:
```
# Segerating the Object type and Numerical type columns
obj=[]
num=[]
for i in df.columns:
    if type(df[i][0])==str:
        obj.append(i)
    else:
        num.append(i)
```

In [15]:
```
print("_________________The Object Type Columns_________________")
obj
```

```
_________________The Object Type Columns_________________
```

Out[15]:
```
['Make', 'Model', 'Vehicle Class', 'Transmission', 'Fuel Type']
```

In [16]:
```
print("_________________The Int Type Columns_____________________")
num
```

```
_________________The Int Type Columns_____________________
```

Out[16]:
```
['Model Year',
 'Engine Size(L)',
 'Cylinders',
 'city',
 'Fuel Consumption(Hwy (L/100 km))',
 'Fuel Consumption(Comb (L/100 km))',
 'Fuel Consumption(Comb (mpg))',
 'CO2 Emissions(g/km)',
 'CO2 Rating',
 'Smog Rating']
```

Let Understand by visualization

Make

In [17]:
```
df["Make"].unique()
```

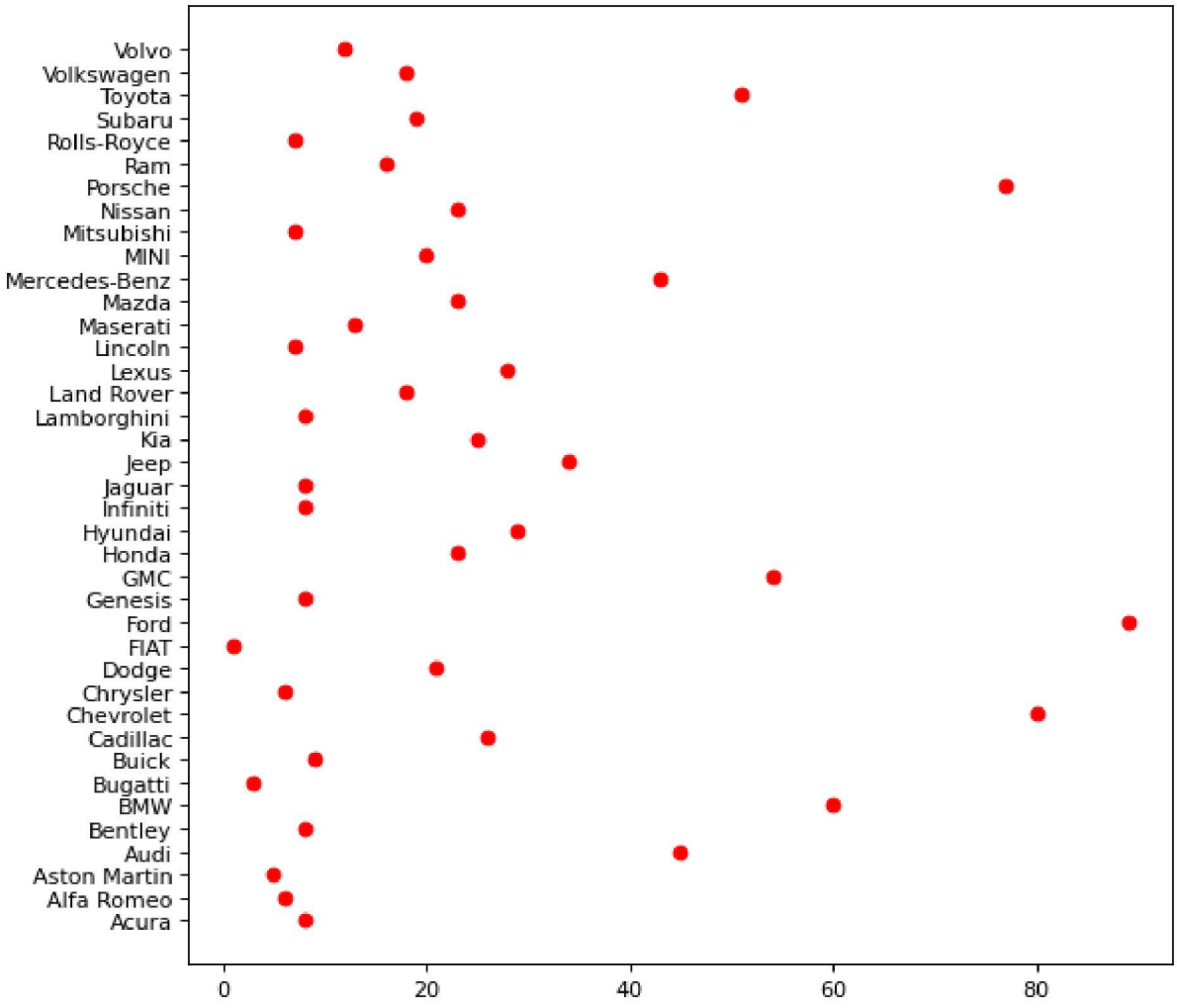
Out[17]:
```
array(['Acura', 'Alfa Romeo', 'Aston Martin', 'Audi', 'Bentley', 'BMW',
       'Bugatti', 'Buick', 'Cadillac', 'Chevrolet', 'Chrysler', 'Dodge',
       'FIAT', 'Ford', 'Genesis', 'GMC', 'Honda', 'Hyundai', 'Infiniti',
       'Jaguar', 'Jeep', 'Kia', 'Lamborghini', 'Land Rover', 'Lexus',
       'Lincoln', 'Maserati', 'Mazda', 'Mercedes-Benz', 'MINI',
       'Mitsubishi', 'Nissan', 'Porsche', 'Ram', 'Rolls-Royce', 'Subaru',
       'Toyota', 'Volkswagen', 'Volvo'], dtype=object)
```

In [18]:
```
len(df[df["Make"]=="Acura"])
```

Out[18]:
```
8
```

In [19]:
```
Makelen=[]
for i in df["Make"].unique():
    x=len(df[df["Make"]==i])
    Makelen.append(x)
```

In [20]:
```
plt.figure(figsize=(8, 8), dpi=80)
plt.scatter(Makelen,df["Make"].unique(),c="red")
plt.show()
```



Observation : In our Dataset More vehicles are ford and Less Vehicles are Flat

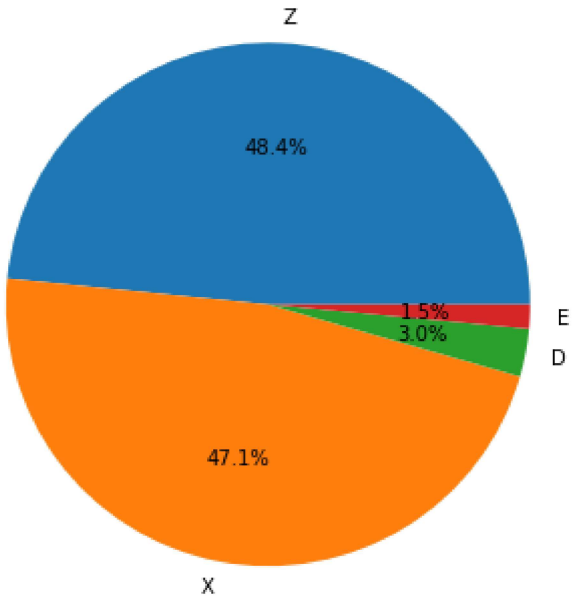
Fuel Type

In [21]:
```
df["Fuel Type"].unique()
```

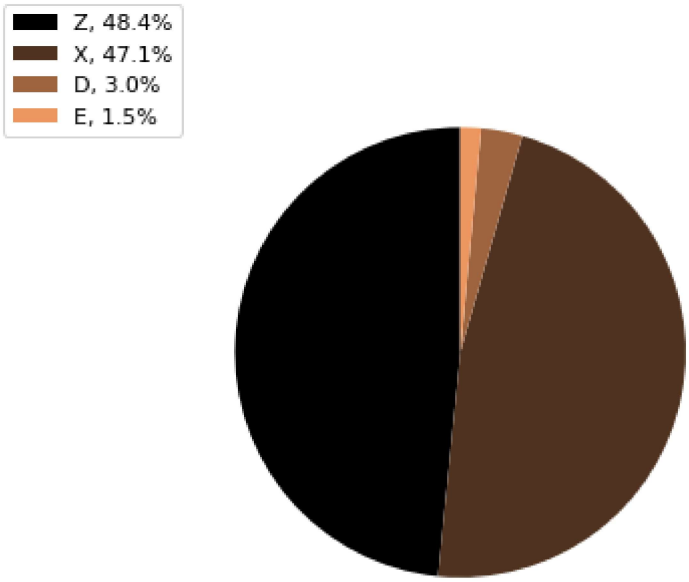
Out[21]:
```
array(['Z', 'X', 'D', 'E'], dtype=object)
```

In [22]:
```
Fuel=[]
for i in df["Fuel Type"].unique():
    x=len(df[df["Fuel Type"]==i])
    Fuel.append(x)
```

In [23]:
```
plt.figure(figsize=(6, 6))
plt.pie(x=Fuel,labels=df["Fuel Type"].unique(),autopct='%1.1f%%')
plt.show()
```



In [24]:
```
sizes = Fuel
labels = df["Fuel Type"].unique()

fig1, ax1 = plt.subplots(figsize=(5, 5))
fig1.subplots_adjust(0.3, 0, 1, 1)

theme = plt.get_cmap('copper')
ax1.set_prop_cycle("color", [theme(1. * i / len(sizes))
                             for i in range(len(sizes))])

_, _ = ax1.pie(sizes, startangle=90, radius=1800)

ax1.axis('equal')

total = sum(sizes)
plt.legend(
    loc='upper left',
    labels=['%s, %1.1f%%' % (
        l, (float(s) / total) * 100)
            for l, s in zip(labels, sizes)],
    prop={'size': 11},
    bbox_to_anchor=(0.0, 1),
    bbox_transform=fig1.transFigure
)

plt.show()
```



Observation Z Type fuel is higher in dataset

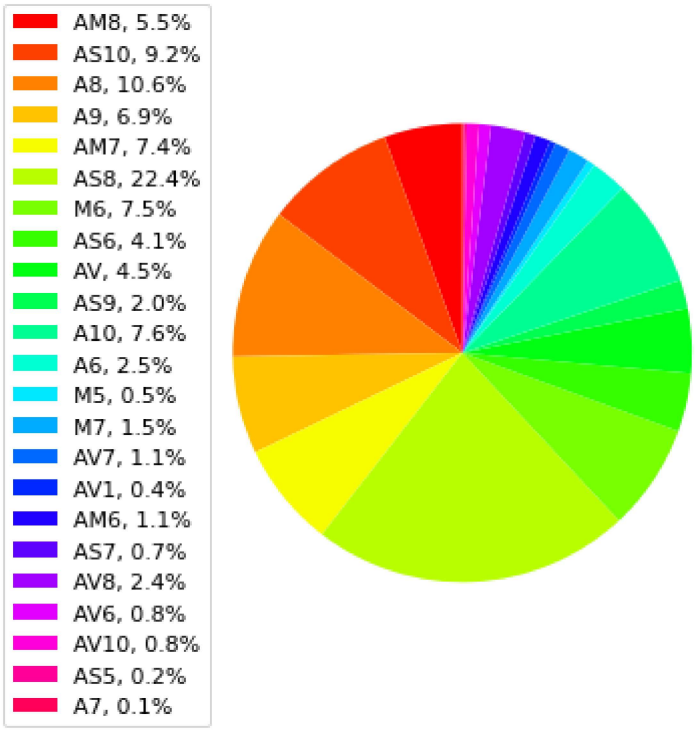
Transmission

In [25]:
```
df["Transmission"].unique()
```

Out[25]:
```
array(['AM8', 'AS10', 'A8', 'A9', 'AM7', 'AS8', 'M6', 'AS6', 'AV', 'AS9',
       'A10', 'A6', 'M5', 'M7', 'AV7', 'AV1', 'AM6', 'AS7', 'AV8', 'AV6',
       'AV10', 'AS5', 'A7'], dtype=object)
```

In [26]:
```
Trans=[]
for i in df["Transmission"].unique():
    x=len(df[df["Transmission"]==i])
    Trans.append(x)
```

In [27]:
```
sizes = Trans
labels = df["Transmission"].unique()

fig1, ax1 = plt.subplots(figsize=(5, 5))
fig1.subplots_adjust(0.3, 0, 1, 1)

theme = plt.get_cmap('hsv')
ax1.set_prop_cycle("color", [theme(1. * i / len(sizes))
                             for i in range(len(sizes))])

_, _ = ax1.pie(sizes, startangle=90, radius=1800)

ax1.axis('equal')

total = sum(sizes)
plt.legend(
    loc='upper left',
    labels=['%s, %1.1f%%' % (
        l, (float(s) / total) * 100)
            for l, s in zip(labels, sizes)],
    prop={'size': 11},
    bbox_to_anchor=(0.0, 1),
    bbox_transform=fig1.transFigure
)

plt.show()
```



Observation

- A8 Transmission Vehicles are 10.6 and AS10 are 9.2

Vehicle Class

In [28]:
```
df["Vehicle Class"].unique()
```

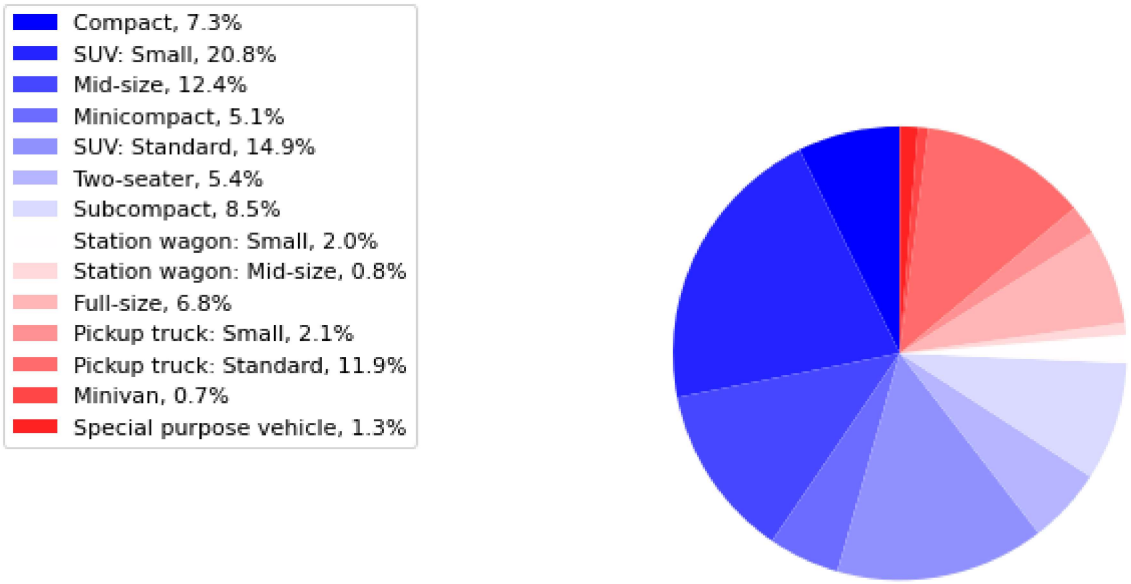
Out[28]:
```
array(['Compact', 'SUV: Small', 'Mid-size', 'Minicompact',
       'SUV: Standard', 'Two-seater', 'Subcompact',
       'Station wagon: Small', 'Station wagon: Mid-size', 'Full-size',
       'Pickup truck: Small', 'Pickup truck: Standard', 'Minivan',
       'Special purpose vehicle'], dtype=object)
```

In [29]:
```
Vehclass=[]
for i in df["Vehicle Class"].unique():
    x=len(df[df["Vehicle Class"]==i])
    Vehclass.append(x)
```

In [30]:
```
Vehclass
```

Out[30]:
```
[69, 197, 117, 48, 141, 51, 80, 19, 8, 64, 20, 113, 7, 12]
```

In [31]:
```
sizes = Vehclass
labels = df["Vehicle Class"].unique()

fig1, ax1 = plt.subplots(figsize=(5, 5))
fig1.subplots_adjust(0.3, 0, 1, 1)

theme = plt.get_cmap('bwr')
ax1.set_prop_cycle("color", [theme(1. * i / len(sizes))
                             for i in range(len(sizes))])

_, _ = ax1.pie(sizes, startangle=90, radius=1800)

ax1.axis('equal')

total = sum(sizes)
plt.legend(
    loc='upper right',
    labels=['%s, %1.1f%%' % (
        l, (float(s) / total) * 100)
            for l, s in zip(labels, sizes)],
    prop={'size': 11},
    bbox_to_anchor=(0.0, 1),
    bbox_transform=fig1.transFigure
)

plt.show()
```



Observation

- SUV small are 20.8
- SUV standard are 14.9
- pickup truck Standard 11.9

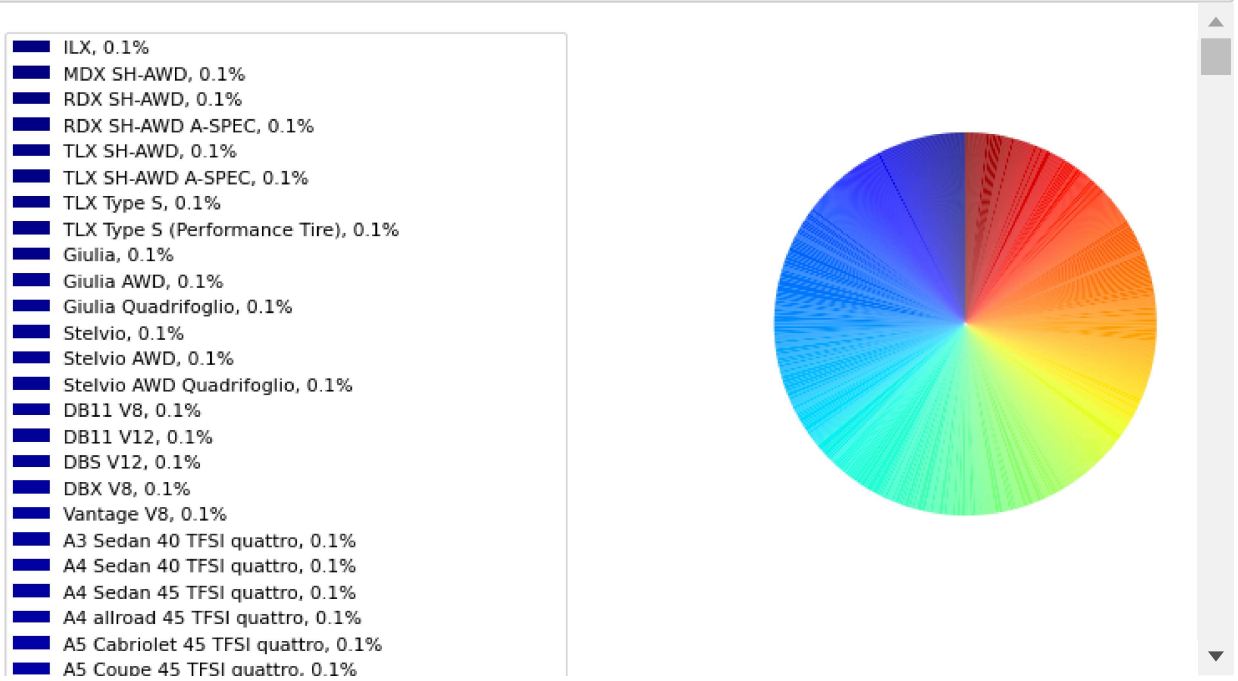
Model

In [32]:
```
df["Model"].unique()
```

Out[32]:
```
array(['ILX', 'MDX SH-AWD', 'RDX SH-AWD', 'RDX SH-AWD A-SPEC',
       'TLX SH-AWD', 'TLX SH-AWD A-SPEC', 'TLX Type S',
       'TLX Type S (Performance Tire)', 'Giulia', 'Giulia AWD',
       'Giulia Quadrifoglio', 'Stelvio', 'Stelvio AWD',
       'Stelvio AWD Quadrifoglio', 'DB11 V8', 'DB11 V12', 'DBS V12',
       'DBX V8', 'Vantage V8', 'A3 Sedan 40 TFSI quattro',
       'A4 Sedan 40 TFSI quattro', 'A4 Sedan 45 TFSI quattro',
       'A4 allroad 45 TFSI quattro', 'A5 Cabriolet 45 TFSI quattro',
       'A5 Coupe 45 TFSI quattro', 'A5 Sportback 45 TFSI quattro',
       'A6 Sedan 45 TFSI quattro', 'A6 Sedan 55 TFSI quattro',
       'A6 allroad 55 TFSI quattro', 'A7 Sportback 55 TFSI quattro',
       'A8 L Sedan 55 TFSI quattro', 'Q3 40 TFSI quattro',
       'Q3 45 TFSI quattro', 'Q5 40 TFSI quattro', 'Q5 45 TFSI quattro',
       'Q5 Sportback 45 TFSI quattro', 'Q7 45 TFSI quattro',
       'Q7 55 TFSI quattro', 'Q8 55 TFSI quattro', 'R8 Coupe Performance',
       'R8 Coupe Performance quattro', 'R8 Spyder Performance',
       'R8 Spyder Performance quattro', 'RS 5 Coupe quattro',
       'RS 5 Sportback quattro', 'RS 6 Avant quattro',
       'RS 7 Sportback quattro', 'RS Q8 quattro', 'S3 Sedan quattro',
```

In [33]:
```
Modells=[]
for i in df["Model"].unique():
    x=len(df[df["Model"]==i])
    Modells.append(x)
```

In [34]:
```
sizes = Modells
labels = df["Model"].unique()

fig1, ax1 = plt.subplots(figsize=(5, 5))
fig1.subplots_adjust(0.3, 0, 1, 1)

theme = plt.get_cmap('jet')
ax1.set_prop_cycle("color", [theme(1. * i / len(sizes))
                             for i in range(len(sizes))])

_, _ = ax1.pie(sizes, startangle=90, radius=1800)

ax1.axis('equal')

total = sum(sizes)
plt.legend(
    loc='upper right',
    labels=['%s, %1.1f%%' % (
        l, (float(s) / total) * 100)
            for l, s in zip(labels, sizes)],
    prop={'size': 11},
    bbox_to_anchor=(0.0, 1),
    bbox_transform=fig1.transFigure
)

plt.show()
```



vehicle class with fuel consumption

In [35]:
```
df["Vehicle Class"].unique()
```

Out[35]:
```
array(['Compact', 'SUV: Small', 'Mid-size', 'Minicompact',
       'SUV: Standard', 'Two-seater', 'Subcompact',
       'Station wagon: Small', 'Station wagon: Mid-size', 'Full-size',
       'Pickup truck: Small', 'Pickup truck: Standard', 'Minivan',
       'Special purpose vehicle'], dtype=object)
```

In [36]:
```
df[df["Vehicle Class"]=="Compact"]['Fuel Consumption(Comb (L/100 km))'].mean()
```

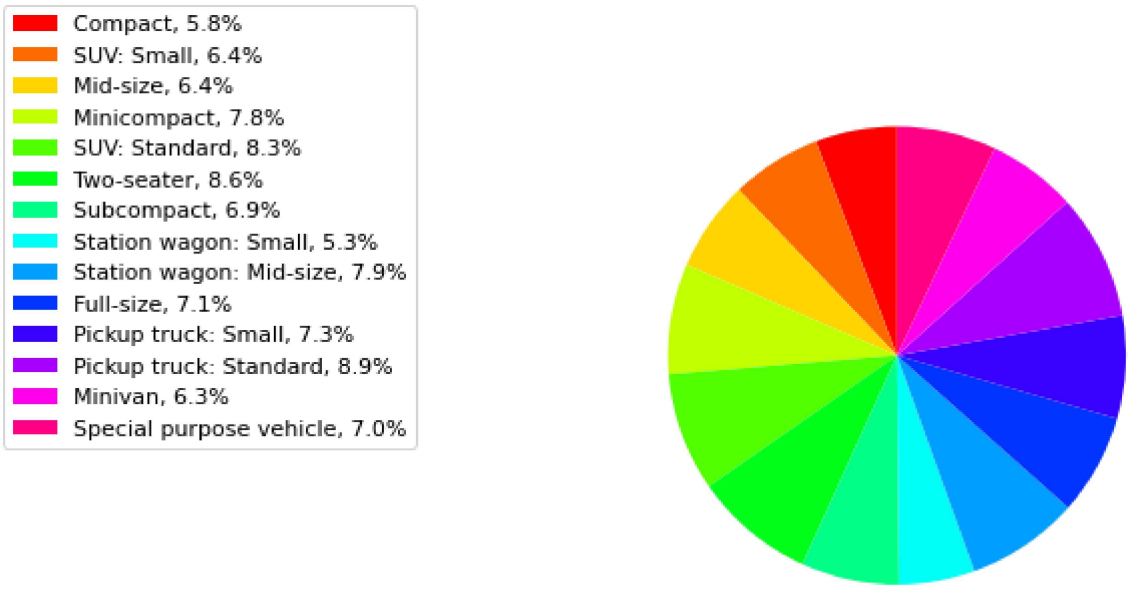
Out[36]:
```
8.840579710144926
```

In [37]:
```
vcm=[]
for i in df["Vehicle Class"].unique():
    meen = df[df["Vehicle Class"]==i]['Fuel Consumption(Comb (L/100 km))'].mean()
    vcm.append(meen)
```

In [38]:
```
vcm
```

Out[38]:
```
[8.840579710144926,
 9.780710659898483,
 9.805128205128208,
 11.960416666666665,
 12.8113475177305,
 13.139215686274513,
 10.6175,
 8.205263157894736,
 12.2,
 10.920312500000001,
 11.135,
 13.601769911504425,
 9.671428571428573,
 10.774999999999999]
```

In [39]:
```
sizes = vcm
labels = df["Vehicle Class"].unique()

fig1, ax1 = plt.subplots(figsize=(5, 5))
fig1.subplots_adjust(0.3, 0, 1, 1)

theme = plt.get_cmap('hsv')
ax1.set_prop_cycle("color", [theme(1. * i / len(sizes))
                             for i in range(len(sizes))])

_, _ = ax1.pie(sizes, startangle=90, radius=1800)

ax1.axis('equal')

total = sum(sizes)
plt.legend(
    loc='upper right',
    labels=['%s, %1.1f%%' % (
        l, (float(s) / total) * 100)
            for l, s in zip(labels, sizes)],
    prop={'size': 11},
    bbox_to_anchor=(0.0, 1),
    bbox_transform=fig1.transFigure
)

plt.show()
```



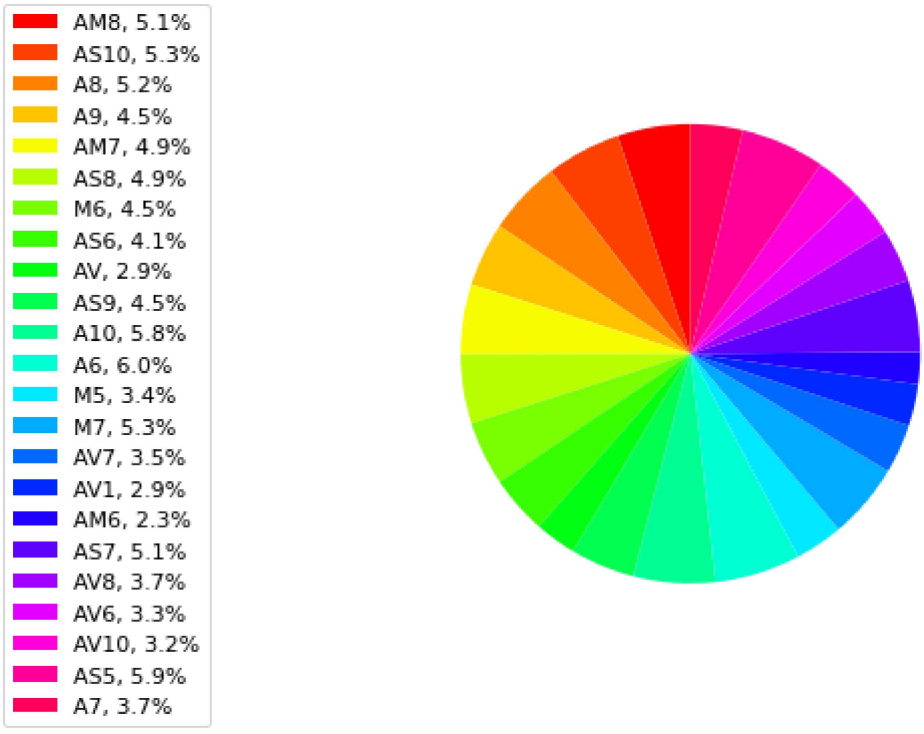
Observation:

- PickUp truck Standard consuming more Fuel 8.9
- SUV 8.3
- Two setter 8.6

Transmission

In [40]:
```
tsm=[]
for i in df["Transmission"].unique():
    meen = df[df["Transmission"]==i]['Fuel Consumption(Comb (L/100 km))'].mean()
    tsm.append(meen)
```

In [41]:
```
sizes = tsm
labels = df["Transmission"].unique()

fig1, ax1 = plt.subplots(figsize=(5, 5))
fig1.subplots_adjust(0.3, 0, 1, 1)

theme = plt.get_cmap('hsv')
ax1.set_prop_cycle("color", [theme(1. * i / len(sizes))
                             for i in range(len(sizes))])

_, _ = ax1.pie(sizes, startangle=90, radius=1800)

ax1.axis('equal')

total = sum(sizes)
plt.legend(
    loc='upper right',
    labels=['%s, %1.1f%%' % (
        l, (float(s) / total) * 100)
            for l, s in zip(labels, sizes)],
    prop={'size': 11},
    bbox_to_anchor=(0.0, 1),
    bbox_transform=fig1.transFigure
)

plt.show()
```



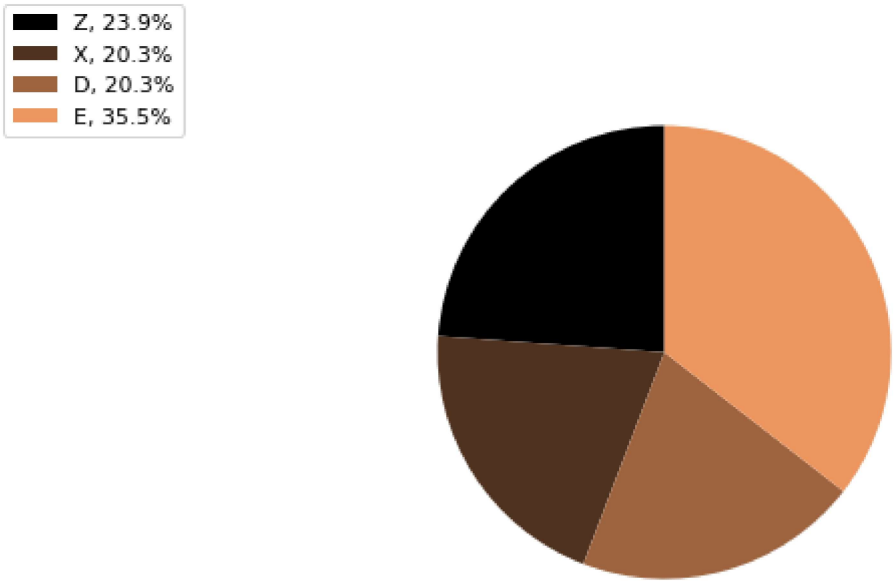
Observation

- A6 Transmission consumnes high 6.0%

Fuel Type

In [42]:
```
ft=[]
for i in df["Fuel Type"].unique():
    meen = df[df["Fuel Type"]==i]['Fuel Consumption(Comb (L/100 km))'].mean()
    ft.append(meen)
```

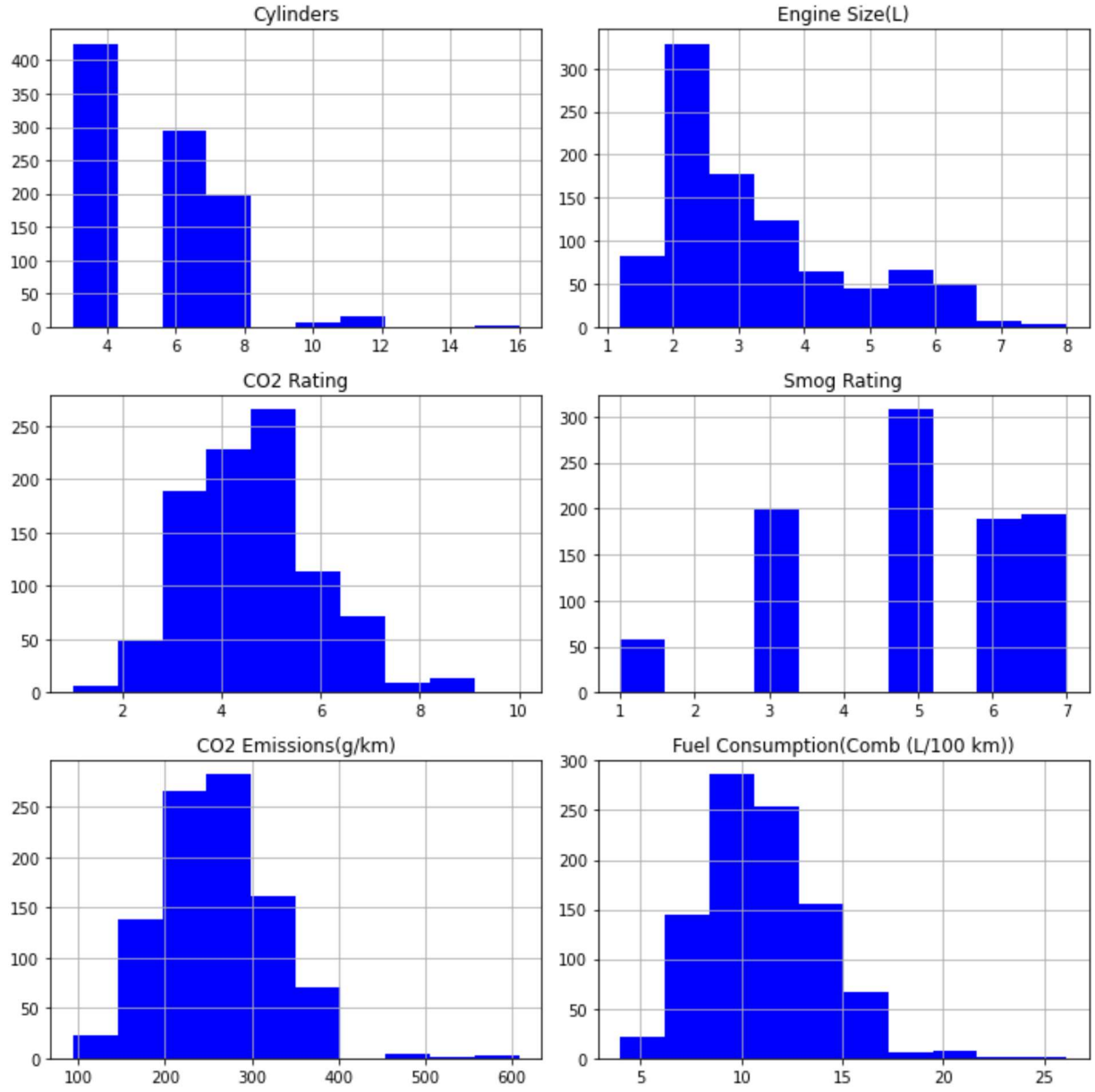
In [43]:
```
sizes = ft
labels = df["Fuel Type"].unique()

fig1, ax1 = plt.subplots(figsize=(5, 5))
fig1.subplots_adjust(0.3, 0, 1, 1)

theme = plt.get_cmap('copper')
ax1.set_prop_cycle("color", [theme(1. * i / len(sizes))
                             for i in range(len(sizes))])

_, _ = ax1.pie(sizes, startangle=90, radius=1800)

ax1.axis('equal')

total = sum(sizes)
plt.legend(
    loc='upper right',
    labels=['%s, %1.1f%%' % (
        l, (float(s) / total) * 100)
            for l, s in zip(labels, sizes)],
    prop={'size': 11},
    bbox_to_anchor=(0.0, 1),
    bbox_transform=fig1.transFigure
)

plt.show()
```



Observaion

- 35.5 percentage of Fuel is E

In [44]:
```
vizual = df[['Cylinders','Engine Size(L)','CO2 Rating', 'Smog Rating','CO2 Emissi
vizual.hist(color = 'blue', figsize = (10, 10))
plt.tight_layout();
plt.show();
```



In [45]:
```
# Note :
'''The first (and most common) reference is litres per 100km (litres/100km).
This is how many litres of fuel the car needs in order to travel 100km.'''
```

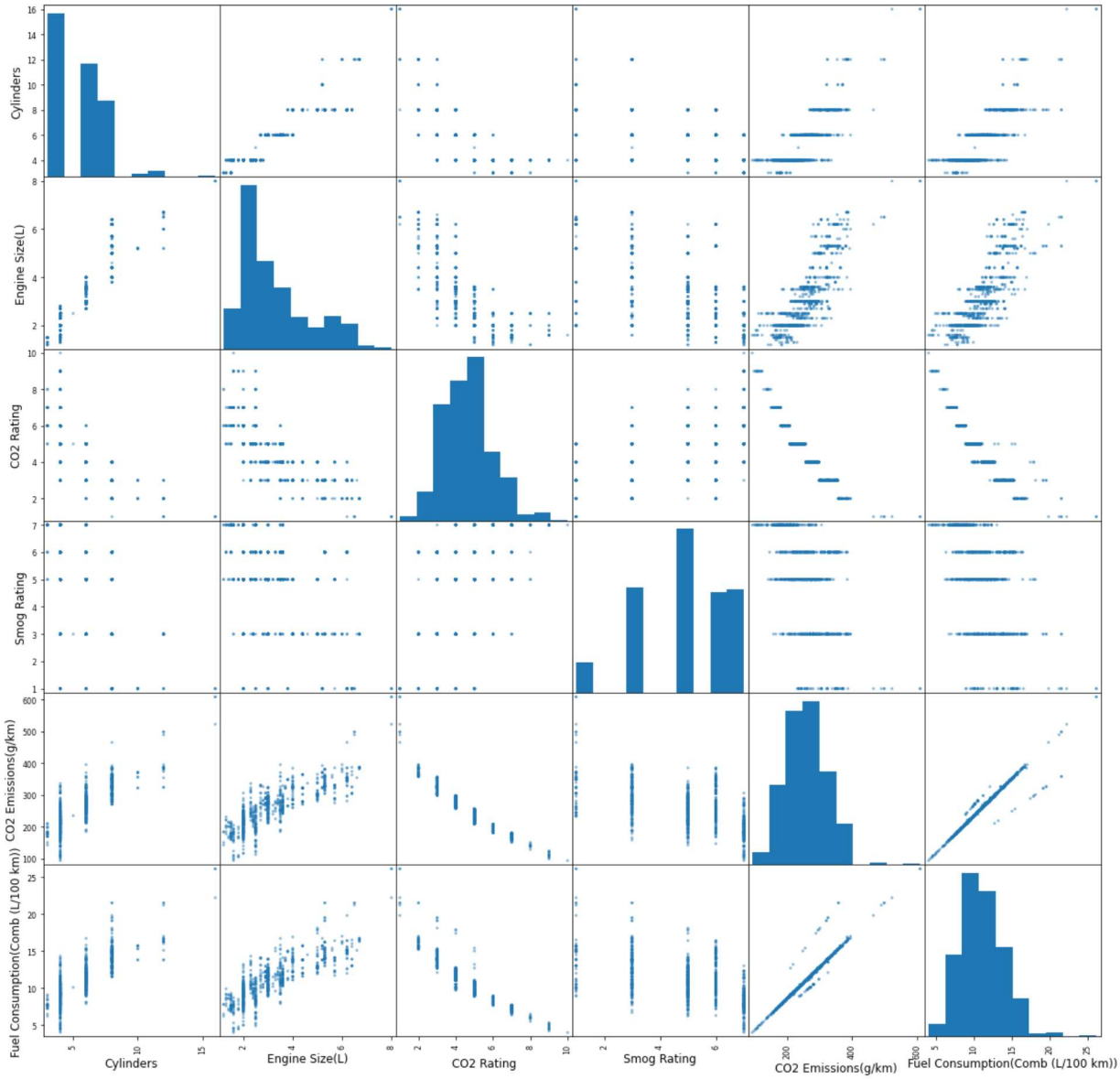
Out[45]:
```
'The first (and most common) reference is litres per 100km (litres/100km). \nTh
is is how many litres of fuel the car needs in order to travel 100km.'
```

In [46]:
```
df.head(1)
```

Out[46]:

	Model Year	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	city	Fuel Consumption(Hwy (L/100 km))
0	2022	Acura	ILX	Compact	2.4	4	AM8	Z	9.9	7.0

In [47]:
```
from pandas.plotting import scatter_matrix
#set font of all elements to size 15
plt.rc('font', size=12)
scatter_matrix(df[['Cylinders','Engine Size(L)','CO2 Rating', 'Smog Rating','CO2
plt.tight_layout;
plt.show()
```



Observations :

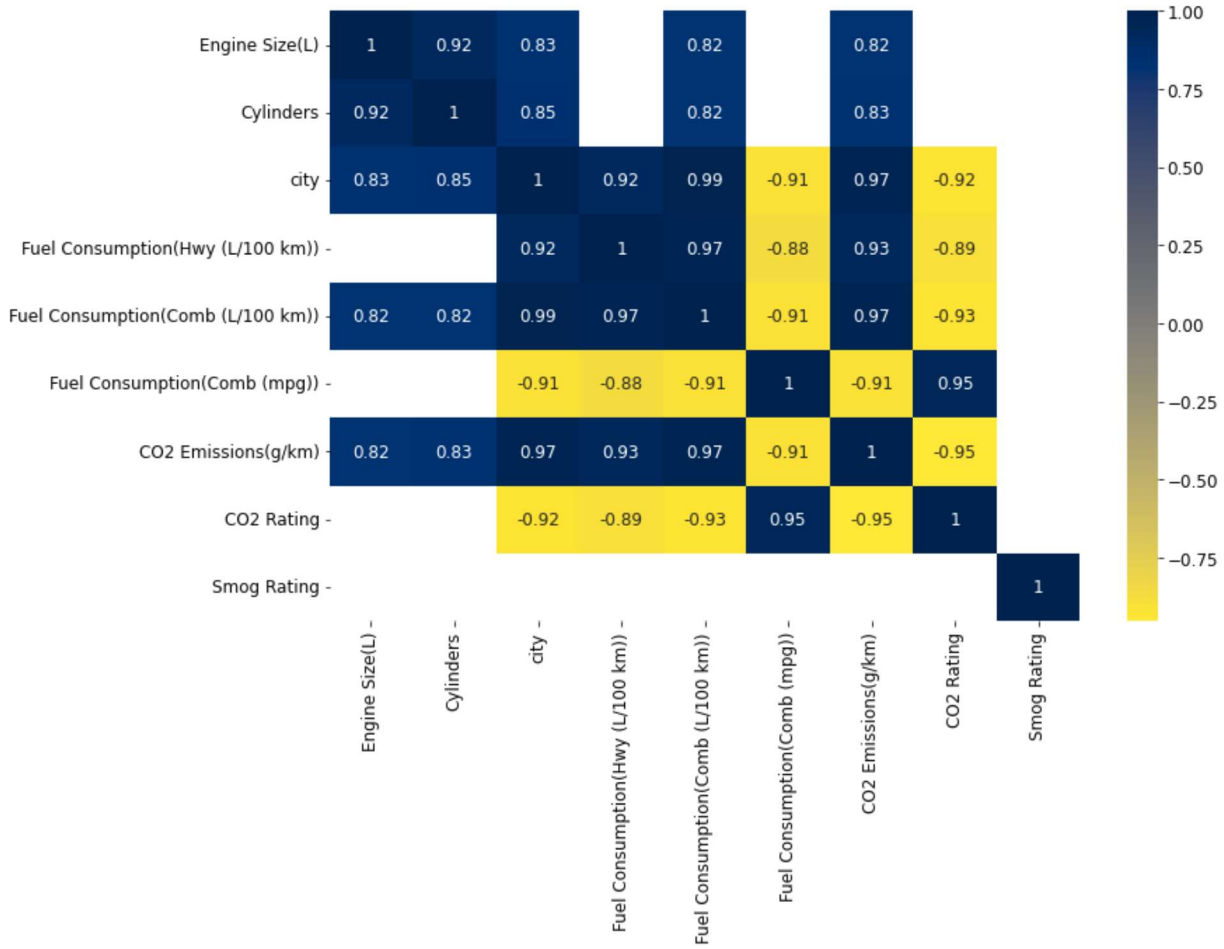
- No of Cylinders increases Fuel consumption also increases, if 15 cylinders it take 25 liters for 100km and 5 to 10 cylinders means it consume near 5 to 17 liters.
- Engine Size increases Fuel consumption also increases
- Full consumption is less means Co2 Rating is Good
- Full consumption is less then Smog Rating is Good
- Full consumption increases Co2 emission in also Increases in Gram/km

In [48]:
```
correc=df.corr()
correc
```

Out[48]:

	Model Year	Engine Size(L)	Cylinders	city	Fuel Consumption(Hwy (L/100 km))	Fuel Consumption(Comb (L/100 km))
Model Year	NaN	NaN	NaN	NaN	NaN	NaN
Engine Size(L)	NaN	1.000000	0.920698	0.834925	0.749374	0.818694
Cylinders	NaN	0.920698	1.000000	0.845688	0.737652	0.821718
city	NaN	0.834925	0.845688	1.000000	0.922850	0.990321
Fuel Consumption(Hwy (L/100 km))	NaN	0.749374	0.737652	0.922850	1.000000	0.967138
Fuel Consumption(Comb (L/100 km))	NaN	0.818694	0.821718	0.990321	0.967138	1.000000
Fuel Consumption(Comb (mpg))	NaN	-0.704163	-0.693594	-0.909477	-0.877531	-0.914305
CO2 Emissions(g/km)	NaN	0.824188	0.833241	0.965632	0.933991	0.971671
CO2 Rating	NaN	-0.766333	-0.762157	-0.920524	-0.894668	-0.927705
Smog Rating	NaN	-0.448239	-0.502149	-0.523928	-0.402099	-0.490473

In [49]:
```
corr = df.drop(['Model Year'], axis = 1).corr()
corr_top = corr[abs(corr)>=.8]
plt.figure(figsize=(12,8))
sns.heatmap(corr_top, cmap="cividis_r", annot = True);
```



Observation :

- Co2 Emission and Cylinders have High correlation to FuelConsumption
- Engine Size and Co2 has same coorelation 0.82 so I negleted Engine size

Data Processing

In [50]:
```
df_copy=df.copy()
```

In [51]:
```
df_copy
```

Out[51]:

	Model Year	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	city	Fu Consumption(Hw (L/100 km
0	2022	Acura	ILX	Compact	2.4	4	AM8	Z	9.9	7
1	2022	Acura	MDX SH-AWD	SUV: Small	3.5	6	AS10	Z	12.6	9
2	2022	Acura	RDX SH-AWD	SUV: Small	2.0	4	AS10	Z	11.0	8
3	2022	Acura	RDX SH-AWD A-SPEC	SUV: Small	2.0	4	AS10	Z	11.3	9
4	2022	Acura	TLX SH-AWD	Compact	2.0	4	AS10	Z	11.2	8
...	...	...	...	...	...	...	...	...	...	
941	2022	Volvo	XC40 T5 AWD	SUV: Small	2.0	4	AS8	Z	10.7	7
942	2022	Volvo	XC60 B5 AWD	SUV: Small	2.0	4	AS8	Z	10.5	8
943	2022	Volvo	XC60 B6 AWD	SUV: Small	2.0	4	AS8	Z	11.0	8
944	2022	Volvo	XC90 T5 AWD	SUV: Standard	2.0	4	AS8	Z	11.5	8
945	2022	Volvo	XC90 T6 AWD	SUV: Standard	2.0	4	AS8	Z	12.4	8

946 rows × 15 columns

In [52]:
```
# checking for Null Values
df_copy.isnull().sum()
```

Out[52]:
```
Model Year                          0
Make                                0
Model                               0
Vehicle Class                       0
Engine Size(L)                      0
Cylinders                           0
Transmission                        0
Fuel Type                           0
city                                0
Fuel Consumption(Hwy (L/100 km))    0
Fuel Consumption(Comb (L/100 km))   0
Fuel Consumption(Comb (mpg))        0
CO2 Emissions(g/km)                 0
CO2 Rating                          0
Smog Rating                         0
dtype: int64
```

Note

- No Null Values in our dataset

Label Encoding

Some features in this dataset are categorical such as Vehicle Class,Transmission, and Fuel Type. Unfortunately, Sklearn models do not handle categorical variables. To convert these features to numerical values, LabelEncoder from sklearn.preprocessing can be used to convert categorical variable into dummy/indicator variables.

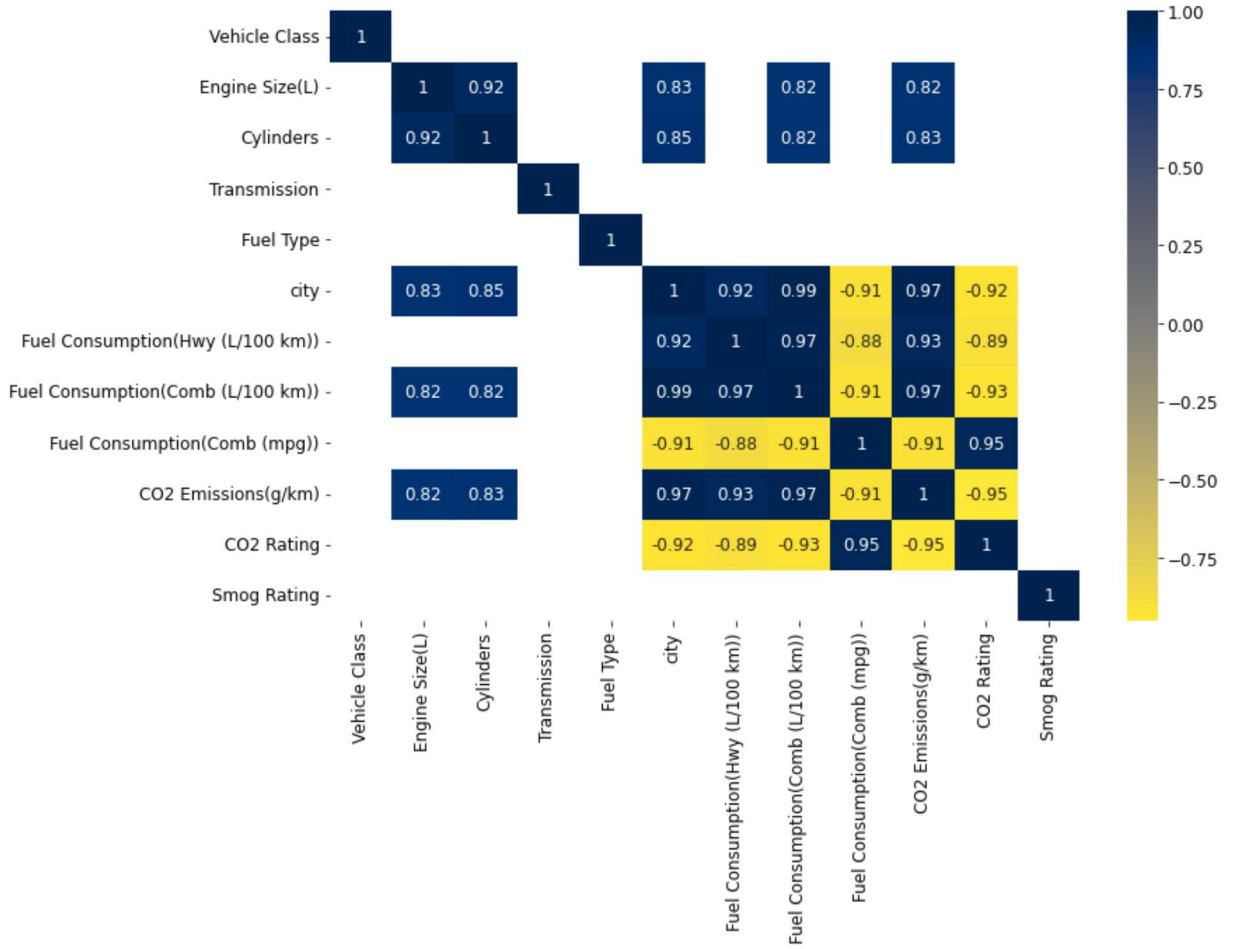
In [53]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df_copy['Vehicle Class'] = le.fit_transform(df_copy['Vehicle Class'])
df_copy['Transmission'] = le.fit_transform(df_copy['Transmission'])
df_copy['Fuel Type'] = le.fit_transform(df_copy['Fuel Type'])
```

In [54]:
```
corr = df_copy.drop(['Model Year'], axis = 1).corr()
corr
```

Out[54]:

	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	city	Consumpti (L/1
Vehicle Class	1.000000	0.110669	0.110852	-0.067992	0.036377	0.178521	0
Engine Size(L)	0.110669	1.000000	0.920698	-0.320581	0.146582	0.834925	0
Cylinders	0.110852	0.920698	1.000000	-0.287515	0.196623	0.845688	0
Transmission	-0.067992	-0.320581	-0.287515	1.000000	0.191615	-0.298531	-0
Fuel Type	0.036377	0.146582	0.196623	0.191615	1.000000	0.214049	0
city	0.178521	0.834925	0.845688	-0.298531	0.214049	1.000000	0
Fuel Consumption(Hwy (L/100 km))	0.210132	0.749374	0.737652	-0.333296	0.111284	0.922850	1
Fuel Consumption(Comb (L/100 km))	0.192646	0.818694	0.821718	-0.316395	0.180178	0.990321	0
Fuel Consumption(Comb (mpg))	-0.212348	-0.704163	-0.693594	0.300221	-0.207403	-0.909477	-0
CO2 Emissions(g/km)	0.199594	0.824188	0.833241	-0.336724	0.185856	0.965632	0
CO2 Rating	-0.200138	-0.766333	-0.762157	0.335756	-0.178188	-0.920524	-0
Smog Rating	-0.105389	-0.448239	-0.502149	0.072109	-0.116236	-0.523928	-0

In [55]:
```
corr_top = corr[abs(corr)>=.8]
plt.figure(figsize=(12,8))
sns.heatmap(corr_top, cmap="cividis_r", annot = True);
```



After Data Preprocessing

- No Change on it

Let start story telling process

First Collect all Observation to create story

Categorial

'Make', 'Model', 'Vehicle Class', 'Transmission', 'Fuel Type'

Numerical

'Model Year', 'Engine Size(L)', 'Cylinders', 'Fuel Consumption (City (L/100 km)', 'Fuel Consumption(Hwy (L/100 km))', 'Fuel Consumption(Comb (L/100 km))', 'Fuel Consumption(Comb (mpg))', 'CO2 Emissions(g/km)', 'CO2 Rating', 'Smog Rating'

No of Items and Height

- In our Dataset More vehicles are ford and Less Vehicles are Flat
- Z Type fuel using is higher in dataset
- A8 Transmission Vehicles are 10.6% and AS10 are 9.2%
- SUV small are 20.8%
- SUV standard are 14.9%
- pickup truck Standard 11.9%

vehicle class with fuel consumption

- PickUp truck Standard consuming more Fuel 8.9
- SUV 8.3
- Two setter 8.6

Transmission with fuel consumption

- A6 Transmission consumnes high 6.0%

Fuel Type with fuel consumption

- 35.5 percentage of Fuel is E

Observed in Paitplot

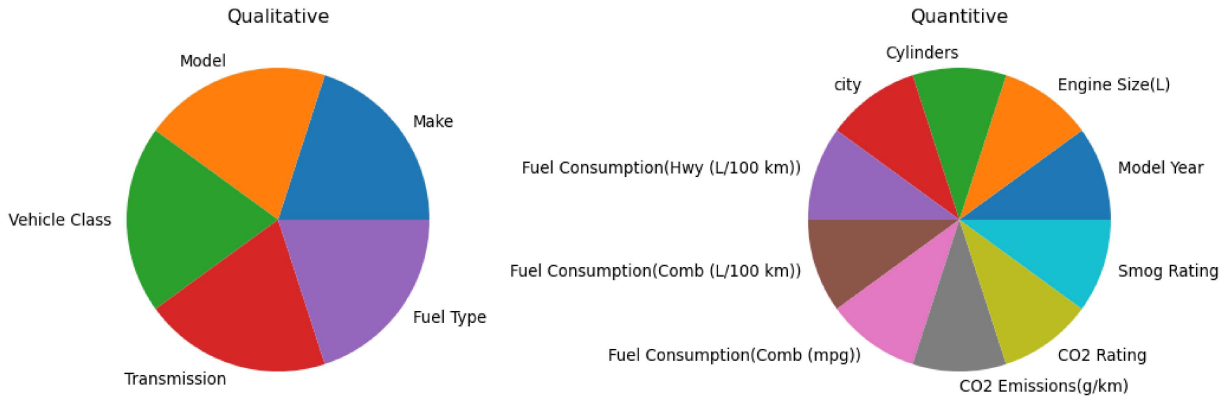
- No of Cylinders increases Fuel consumption also increases, if 15 cylinders it take 25 liters for 100km and 5 to 10 cylinders means it consume near 5 to 17 liters.
  - Engine Size increases Fuel consumption also increases
  - Full consumption is less means Co2 Rating is Good
  - Full consumption is less then Smog Rating is Good
  - Full consumption increases Co2 emission in also Increases in Gram/km

High co-orelation with target feature

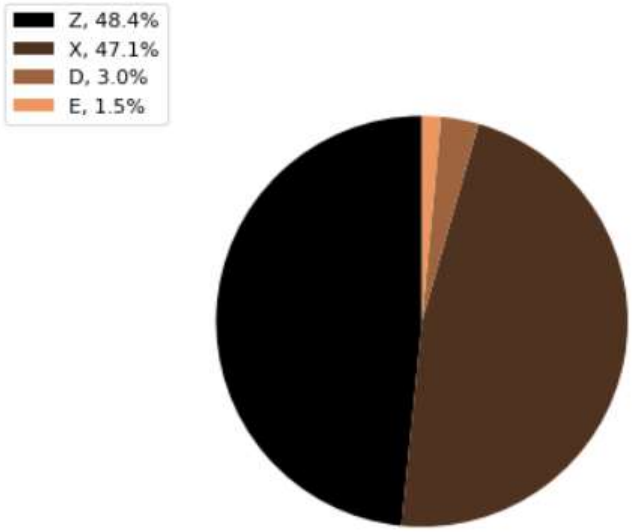
- Co2 Emission and Cylinders have High correlation to FuelConsumption
- Engine Size and Co2 has same coorelation 0.82 so I negleted Engine size

Visualization Story

In [56]:
```
plt.figure(figsize=(4, 4), dpi=80)
plt.subplots_adjust(left=0.5,bottom=1,right=3.5,top=2,wspace=0.5,hspace=None)

plt.subplot(1,2,1)     # It is Refering Postion
plt.pie(x=[1,1,1,1,1],labels=obj)
plt.title("Qualitative")

plt.subplot(1,2,2)    # It is Refering Postion
plt.pie(x=[1,1,1,1,1,1,1,1,1,1],labels=num)
plt.title("Quantitive")
plt.show()
```

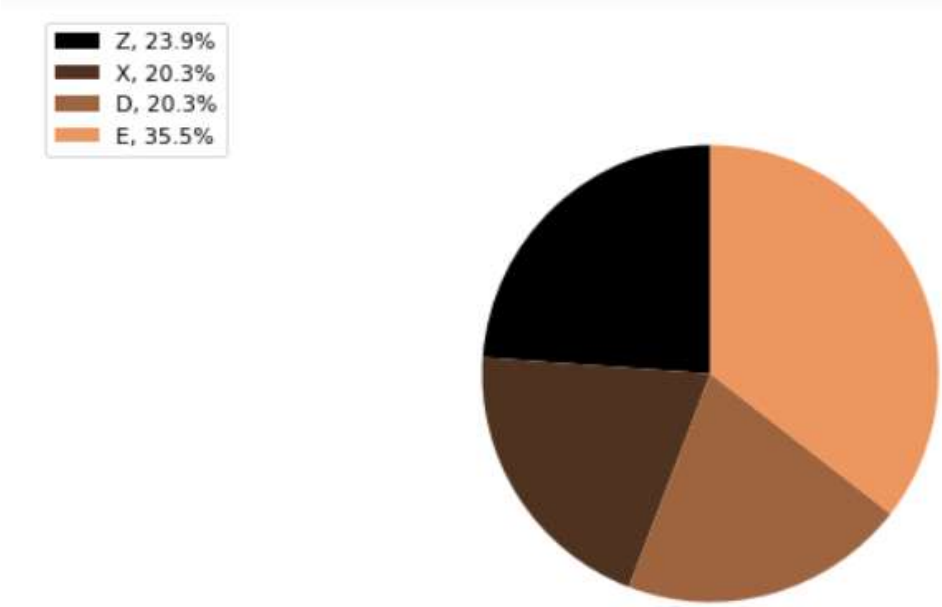


Our Promblem is to find out the which feature consuming the more fuel so let check which fuel type is majority in our data set

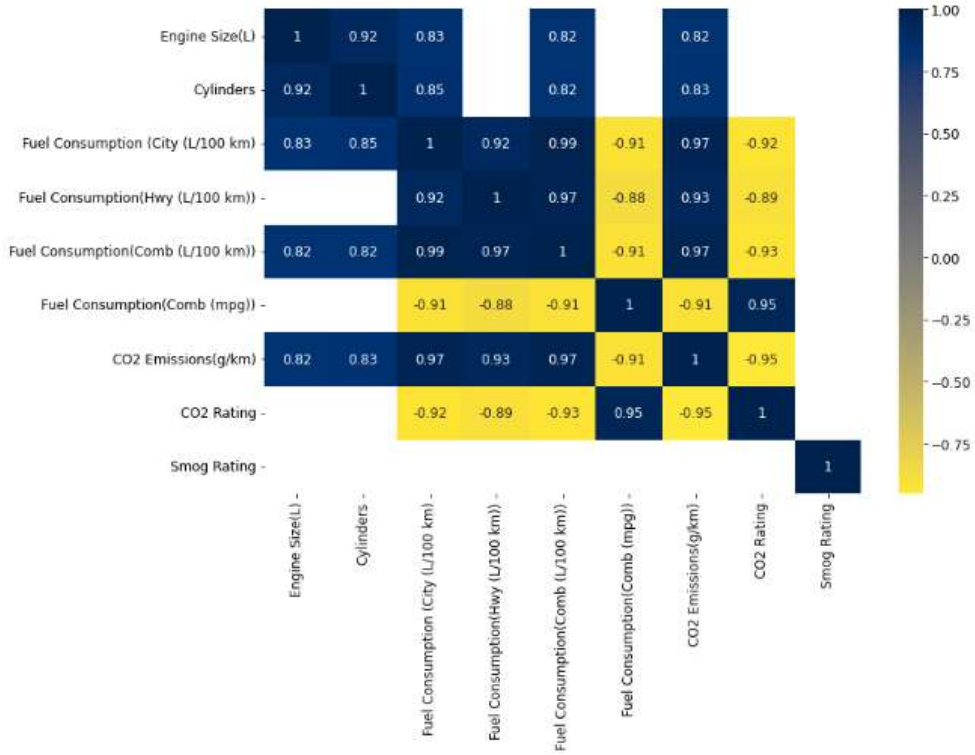


Perium Gasoline["Z"] type is highest in our dataset

Did Gasoline using vehicles using more fuel consumption No Etahol Using Vehicles are Consumning more fuel

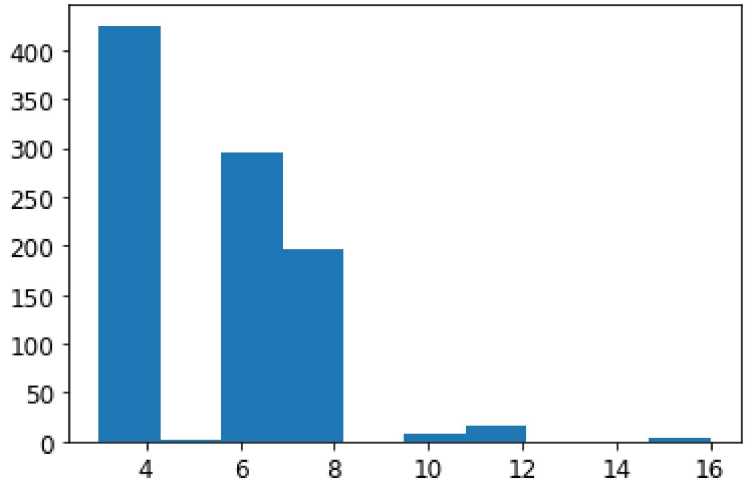


We can Observe Fuel Type X,D and Z are Less Consume compare to E !Ok We know that Z Fuel Using Vehicles are Highes in our dataset but more amount Fuel consumption tye is E



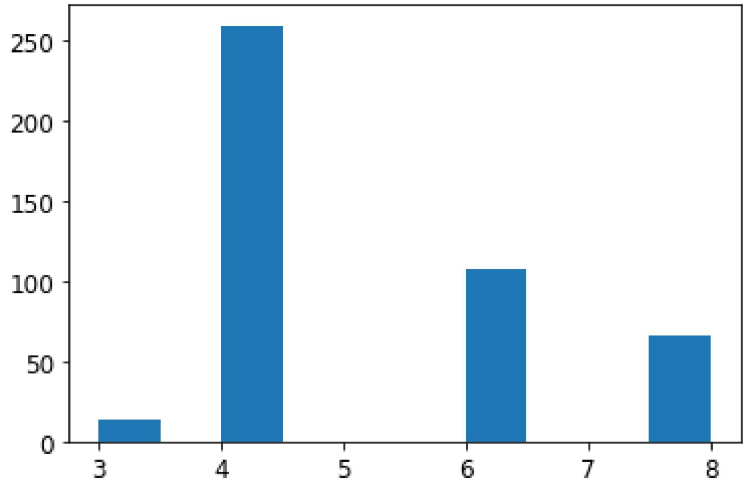
As we seen Cylinders and Co2 Emission have high correlation with the target variable and Cylinder and Co2 have same co-orelation so we negelted it

In [57]:
```
x=df["Cylinders"]
plt.hist(x)
plt.show()
```



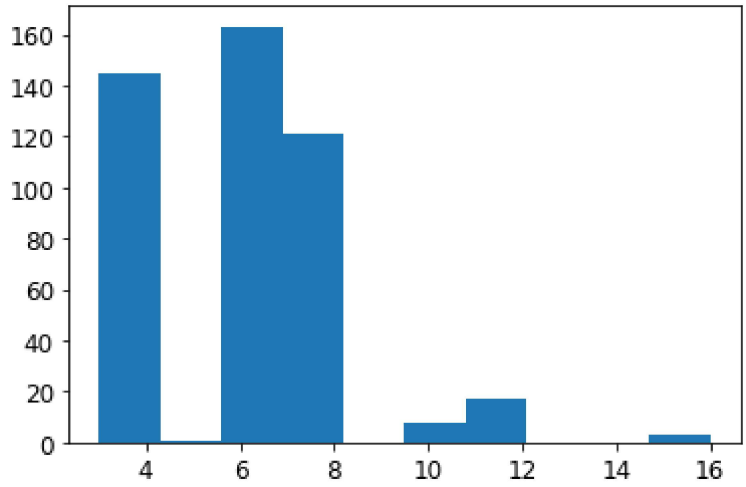
In 4 cylinder vehicles high in our dataset

In [58]:
```
x=df[df["Fuel Type"]=="X"]["Cylinders"]
plt.hist(x)
plt.show()
```



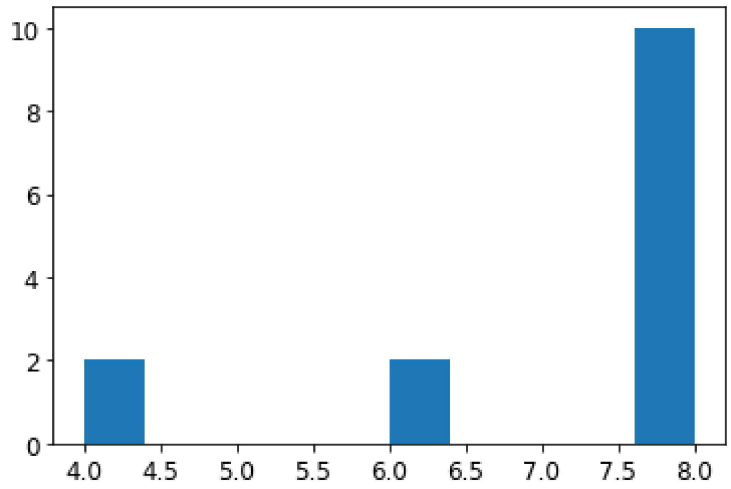
4 cylinder vehicles are most using X Fuel

In [59]:
```
x=df[df["Fuel Type"]=="Z"]["Cylinders"]
plt.hist(x)
plt.show()
```

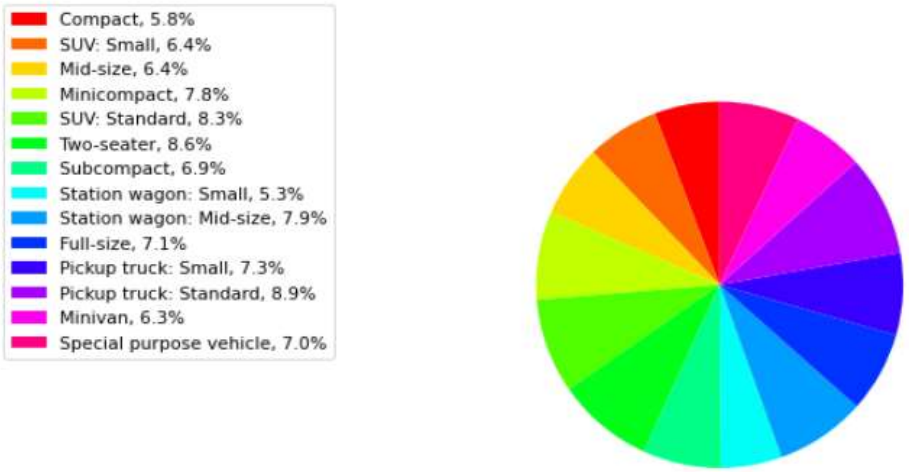


z type fuel balancing

In [60]:
```
x=df[df["Fuel Type"]=="E"]["Cylinders"]
plt.hist(x)
plt.show()
```



E-type vehicles more using the 8 Cylinders



Highest Fuel Consuming Vehicels class are "PickUp standard" and SUV, Two Seater

In [61]:
```
x=df[df["Cylinders"]>=8]["Vehicle Class"].unique()
x
```

Out[61]:
```
array(['Minicompact', 'SUV: Standard', 'Two-seater',
       'Station wagon: Mid-size', 'Mid-size', 'Full-size', 'Subcompact',
       'Pickup truck: Standard', 'SUV: Small', 'Compact'], dtype=object)
```

Data Prepartion

In [62]:
```
X = df_copy[['Vehicle Class', 'Engine Size(L)', 'Cylinders', 'Transmission', 'Fue
y = df_copy[['Fuel Consumption(Comb (L/100 km))']]
X.shape,y.shape
```

Out[62]:
```
((946, 8), (946, 1))
```

In [63]:
```
X
```

Out[63]:

	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	CO2 Emissions(g/km)	CO2 Rating	Smog Rating
0	0	2.4	4	7	3	200	6	3
1	7	3.5	6	8	3	263	4	5
2	7	2.0	4	8	3	232	5	6
3	7	2.0	4	8	3	242	5	6
4	0	2.0	4	8	3	230	5	7
...	...	...	...	...	...	...	...	...
941	7	2.0	4	12	3	219	5	5
942	7	2.0	4	12	3	219	5	5
943	7	2.0	4	12	3	232	5	7
944	8	2.0	4	12	3	236	5	5
945	8	2.0	4	12	3	252	5	7

946 rows × 8 columns

Model :

In [64]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (756, 8) (756, 1)
Test set: (190, 8) (190, 1)
```

In [65]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[65]:
```
LinearRegression()
```

In [66]:
```
model.intercept_
```

Out[66]:
```
array([0.06231241])
```

In [67]:
```
model.coef_
```

Out[67]:
```
array([[ 0.00066058,  0.13549093, -0.01479244,  0.00789365, -0.03712664,
         0.04138051, -0.0418067 ,  0.03372079]])
```

Prediction

In [68]:
```
train_pred = model.predict(X_train)
test_pred = model.predict(X_test)
```

Evaluation :

In [69]:
```
from sklearn.metrics import mean_squared_error
test_RMSE =np.sqrt(mean_squared_error(y_test,test_pred))
train_RMSE = np.sqrt(mean_squared_error(y_train,train_pred))
print(test_RMSE,train_RMSE)
```

```
0.5938690166819434 0.6873668226930023
```

In [70]:
```
model.score(X_train,y_train)
```

Out[70]:
```
0.9402143178088415
```

In [71]:
```
model.score(X_test,y_test)
```

Out[71]:
```
0.9636159473257309
```