In [ ]:

In [ ]:

Frequency tables

1)to compute a simple cross-tabulation of one or two (or more) factors 2)by default computes frquency table of factors

In [ ]:

In [ ]:

Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

In [ ]:

In [ ]:

Data visulation

```
1)Data visualition allows us to quickly interpret the data and adjust
the different variable to see their effect 2)Technology is
increasingly making easier for us to do so Ques-Why visualise data?
Ans-Observe the pattern
2)Identify extreme values that could be anomalies
3)easy interpretion
```

In [2]:

```
#POPULAR plotting libraries in python
#Python ofers mutiple graphing libraries tht offer diverse feature
```

```
1)matplotlib-to create 2D graphs and plots
2)pandas visulisation-easy to use interface,built on matplotlib
3)seaborn-provides a high level interface for drawing attraction and
informative statiscal graphics
4)ggplot-based on R's ggpplot2,uses grammer and graphics
5)plotly -can create interactive plots
```

matplotlib

```
1)matplotlib is a 2d plotting library which produces a good  quality
figures
2)Although it has a its orgin, in emulating the matlab graphic
command ,it is independent of matlab
3)it makes heavy use of numpy and other extension code to provide
good perfrmance even for large arrays
```

In [3]:

```
#Scatter plot
```

```
Ques-What is scatter plot
Ans-The scatter plot is a set of points that represent the values
obtained for two diffrent variable plotted on a horizontol and
vertical axes
Ques-When to use Scatter plots?
Ans-1)Scatter plots used to convey the relationship between two
numwerical values
Ques2-When to use scatter plots?
```

```
Ans-Scatter plots are used to convey the relationship between two
numerical valus
2)Scatter plots are sometime called correlation plots because they
show how two variable arw correlated
```

In [4]:

```
#Import data into Spyder
```

In [5]:

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

In [6]:

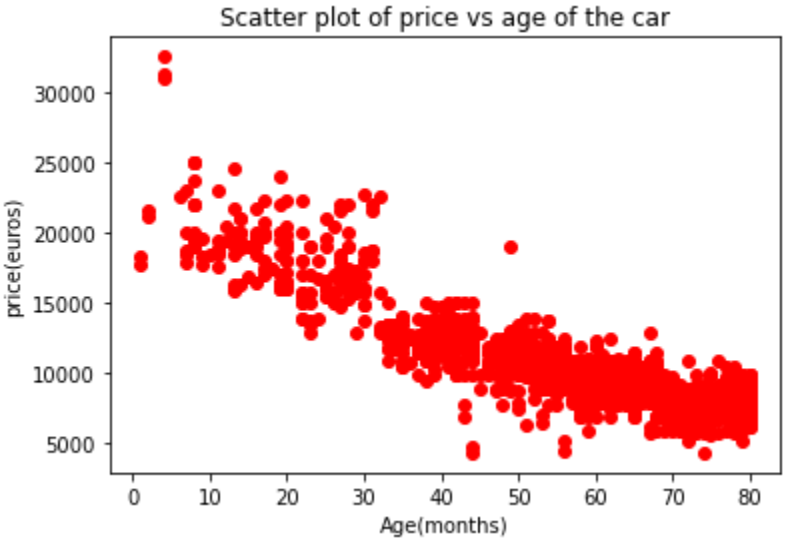
```
cars_data=pd.read_csv('ToyotaCorolla.csv',index_col=0,na_values=["??",'
```

In [7]:

```
#Removing the missing values from the dataframe
cars_data.dropna(axis=0,inplace=True)
#here we are using dropna to drop the nan values
#axis=0 means drop the values from all the rows
#inplace=True means we are droping the alues and it will show n the dat
```

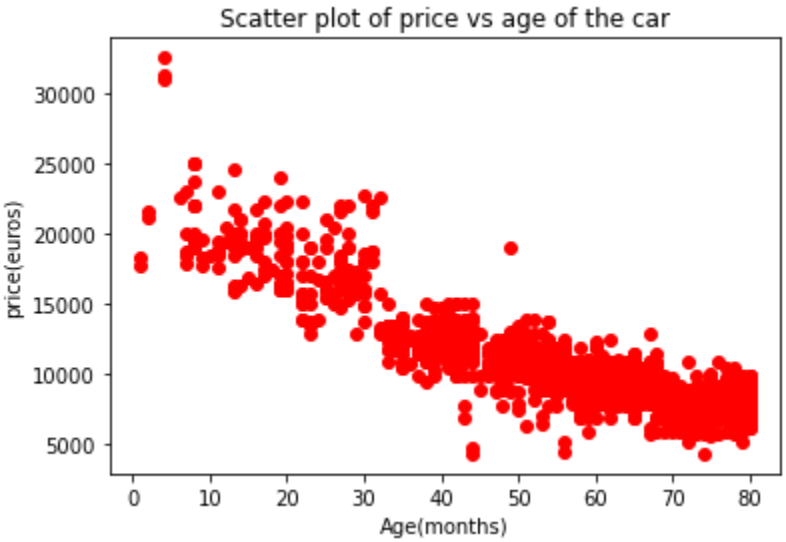
In [8]:

```
plt.scatter(cars_data['Age_08_04'],cars_data['Price'],c='red')
#here age_08_04 represent the x axis
#here price represent the y axis
#here the red represent the mark
plt.title("Scatter plot of price vs age of the car")#fiving the tittle
plt.xlabel('Age(months)')#giving the x axis name
plt.ylabel('price(euros)')#giving the y axis name
plt.show()#showing the plot
```



In [9]:

```
plt.scatter(cars_data['Age_08_04'],cars_data['Price'],c='red')
plt.title("Scatter plot of price vs age of the car")#fiving the tittle
plt.xlabel('Age(months)')#giving the x axis name
plt.ylabel('price(euros)')#giving the y axis name
plt.show()#showing the plot
```



In [10]:

```
#the price of the car increases as age of car decreases
```

Histogram

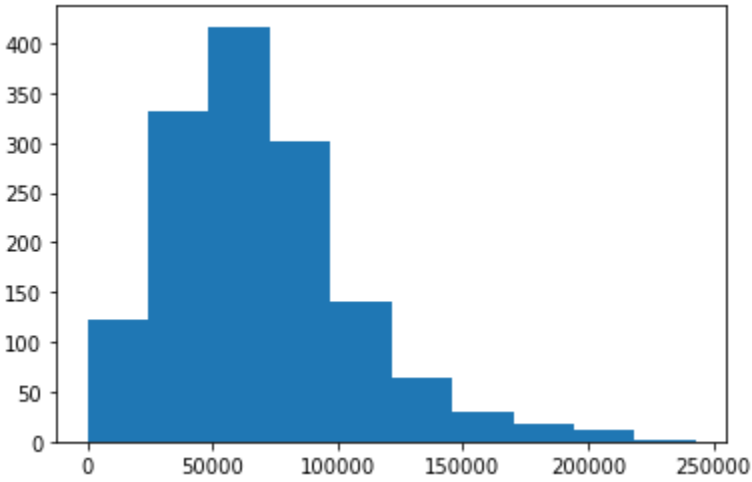
```
Ques-What is a histogram?
Ans-It is a graphical representation of data using bars of diffrent
height
2)histogram groups the number into ranges and the height of each bar
depicts the frequency of each range or bin
Ques-whhen to use histogram?
Ans-to represent the frequency distribution of numerical variable
```

In [11]:

```
plt.hist(cars_data['KM'])#plt is sub package of matplotlib hist is the
#KM is the x axis
```

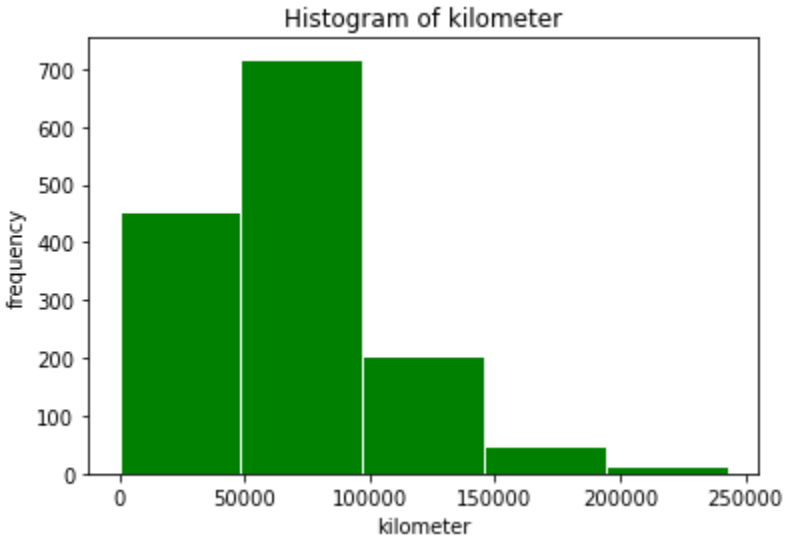
Out[11]:

```
(array([122., 331., 417., 301., 140.,  64.,  30.,  17.,
12.,   2.]),
 array([1.000000e+00, 2.430090e+04, 4.860080e+04, 7.2900
70e+04,
        9.720060e+04, 1.215005e+05, 1.458004e+05, 1.7010
03e+05,
        1.944002e+05, 2.187001e+05, 2.430000e+05]),
 <BarContainer object of 10 artists>)
```



In [12]:

```
plt.hist(cars_data['KM'],color='green',edgecolor='white',bins=5)
plt.title('Histogram of kilometer')
plt.xlabel("kilometer")
plt.ylabel("frequency")
plt.show()
```



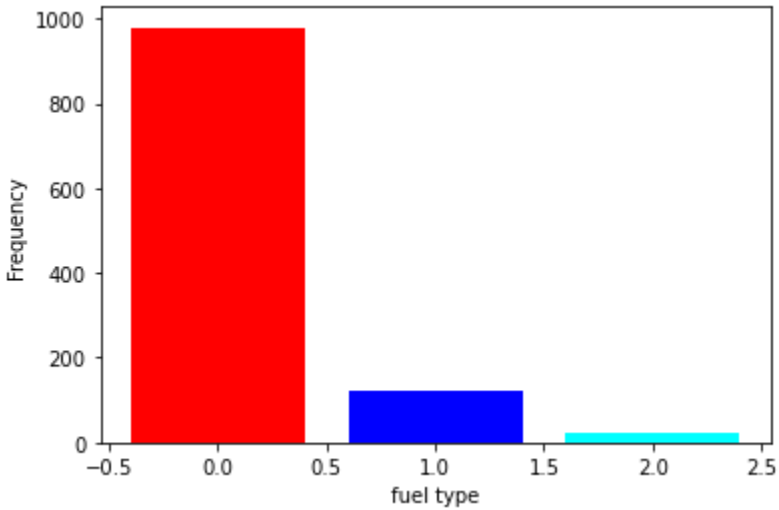
Bar plot

```
Ques-What is bar plot?
Ans-A bar plot is a plot that presents categorical data with
rectangular bars with lengths proportional to the counts that they
represent
Ques-When to use bar  plot?
```

```
Ans-1)to represent the frequency distribution of categorical
variables
2)A bar diagram makes it easy to compares sets of data betweeen
diffrent groups
3)the space between the graphs is present in bar graph but it is not
in histogram
4)use numerical values for histogram and categorical for bar plot
```

In [13]:

```
counts=[979,120,22]
fueltype=('petrol','diesel','cng')
index=np.arange(len(fueltype))
plt.bar(index,counts,color=['red','blue','cyan'])#by this code petrol r
plt.xlabel('fuel type')
plt.ylabel('Frequency')
plt.show()
```



seaborn

```
1)Seaborn is a python data visualisation library based on matplotlib
2)it provides high level interface for drawing attractive and
informative statiscal graph
```

In [14]:

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

In [15]:

```
cars_data=pd.read_csv('ToyotaCorolla.csv',index_col=0,na_values=["??",'
```

In [16]:

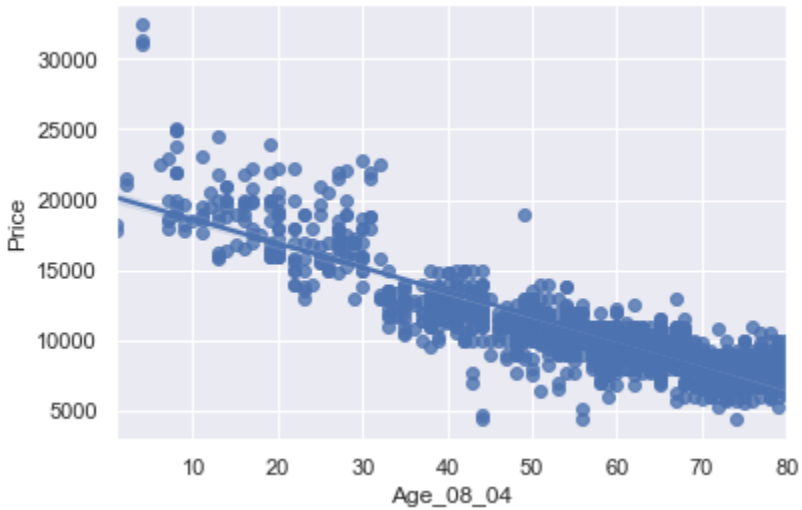
```
cars_data.dropna(axis=0,inplace=True)
```

In [17]:

```
#scatter plt price vs age with default arguments
sns.set(style="darkgrid")#it means in background we have black backgrou
sns.regplot(x=cars_data['Age_08_04'],y=cars_data['Price'])#regplot star
#if we write sns.set() we get default as background
```

Out[17]:

```
<AxesSubplot:xlabel='Age_08_04', ylabel='Price'>
```



1)By default fit_reg=True 2)It estimate and plots a regression model relating the x and

y variables

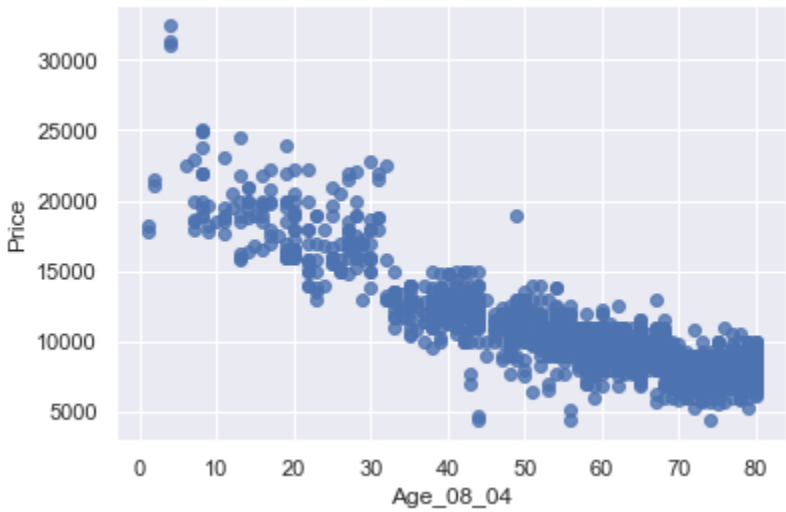
Scatter plot price vs age without the regression fit line

In [18]:

```
sns.regplot(x=cars_data['Age_08_04'],y=cars_data['Price'],fit_reg=False
#we did not metion the name of x axis and y axis it directly take it fr
#1,10.20..... are called astic labels
#500.1000...are white tick labels
```

Out[18]:

```
<AxesSubplot:xlabel='Age_08_04', ylabel='Price'>
```



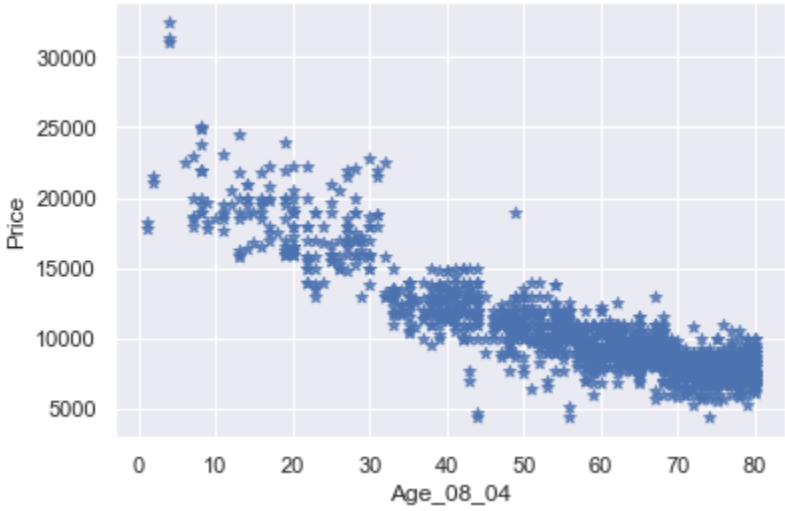
Customizing the appearance of marker

In [19]:

```
#here we are changing the shape of the marker
sns.regplot(x=cars_data['Age_08_04'],y=cars_data['Price'],marker="*",fi
```

Out[19]:

```
<AxesSubplot:xlabel='Age_08_04', ylabel='Price'>
```



Scatter plot of price vs age by fuel_type

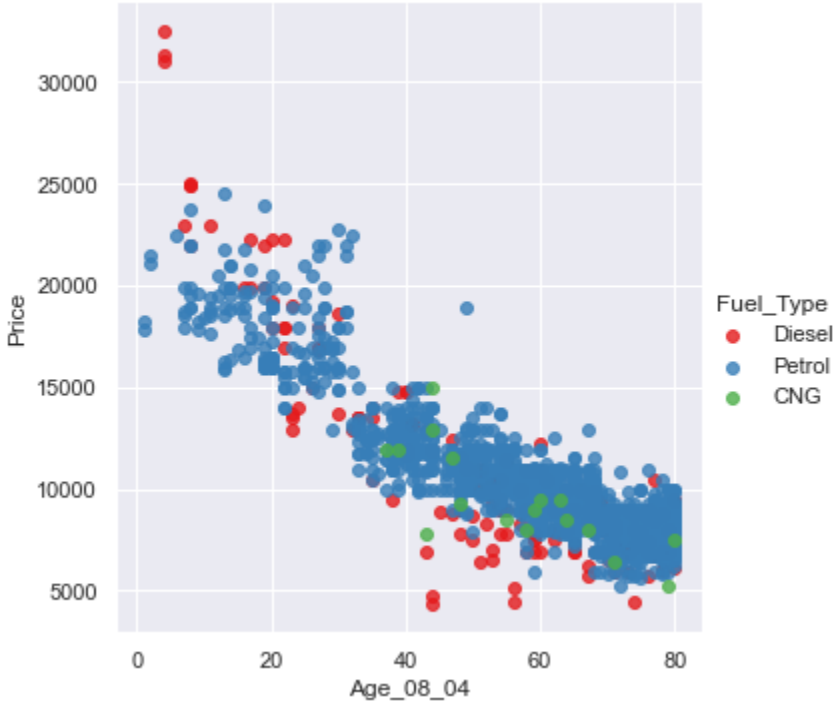
```
Using hue parameter including another variable to show the fuel types
categories with diffrent colors
```

In [20]:

```
sns.lmplot(x='Age_08_04',y='Price',data=cars_data,fit_reg=False,hue='Fu
#lmplot combine the regression plot and face grid it is very useful for
#hue points are different color on the basis of car
#legend is equal true means which color represent which category
#pelette it is color set here we are using set1
#similar custom the appearance of the marker using 1)transparency 2)sha
```

Out[20]:

```
<seaborn.axisgrid.FacetGrid at 0x772af58>
```



Histogram

```
1)Histogram with default kernel density estimate
```

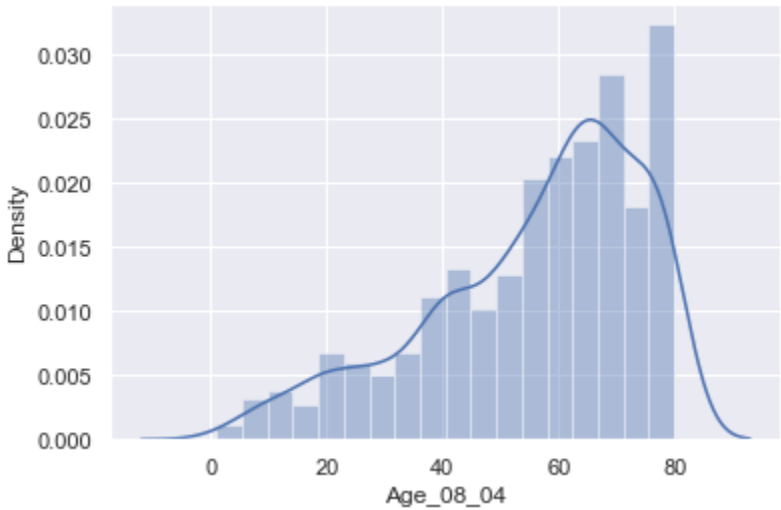
In [21]:

```
sns.distplot(cars_data['Age_08_04'])
#distplot represent the distribution plot
#input can be numerical or continus variable
#
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:2551: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[21]:

```
<AxesSubplot:xlabel='Age_08_04', ylabel='Density'>
```



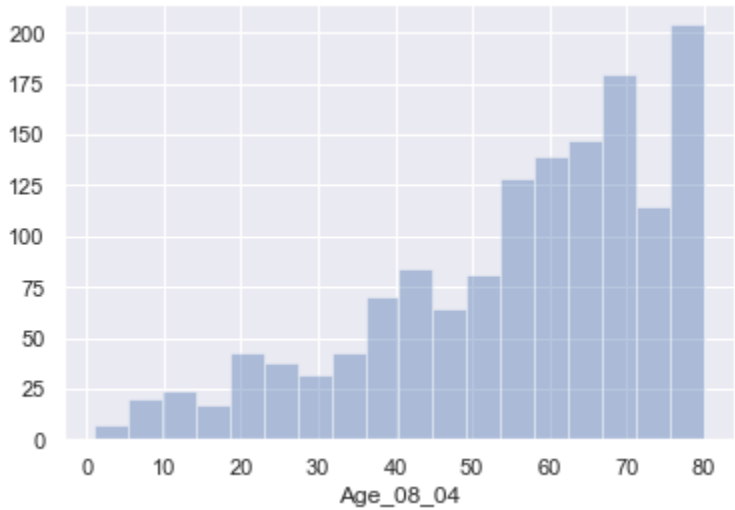
In [22]:

```
#plotting without the kernel density
sns.distplot(cars_data['Age_08_04'],kde=False)
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:2551: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[22]:

```
<AxesSubplot:xlabel='Age_08_04'>
```

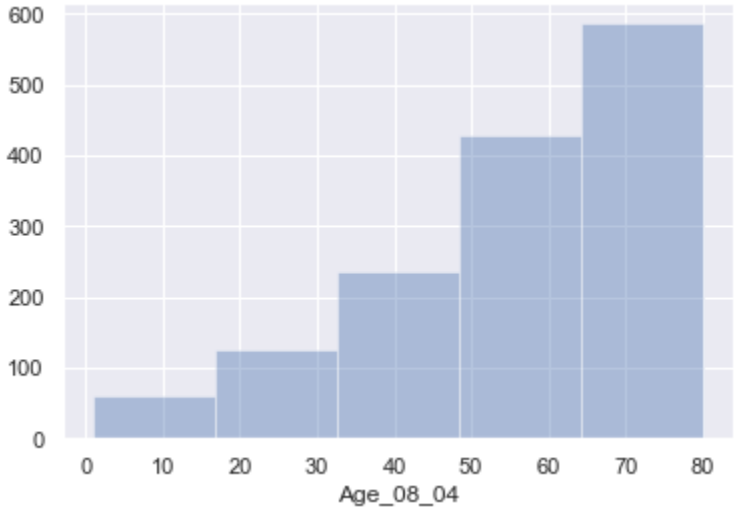


In [23]:

```
#Histogram with fixed no of bins
sns.distplot(cars_data['Age_08_04'],kde=False,bins=5)
#
```

Out[23]:

```
<AxesSubplot:xlabel='Age_08_04'>
```



Bar plot

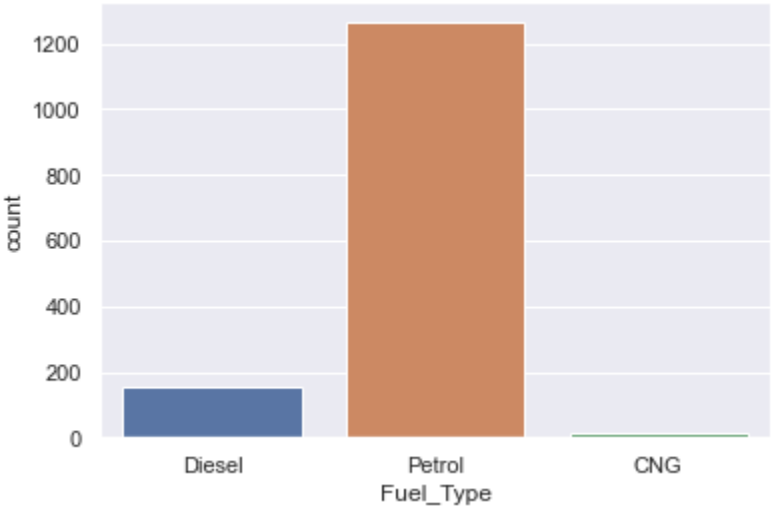
```
Frequency distribution of fuel type of cars because it is categorical
```

In [24]:

```
sns.countplot(x="Fuel_Type",data=cars_data)
#countplot is used in bar plot
```

Out[24]:

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



Groupoed bar plot

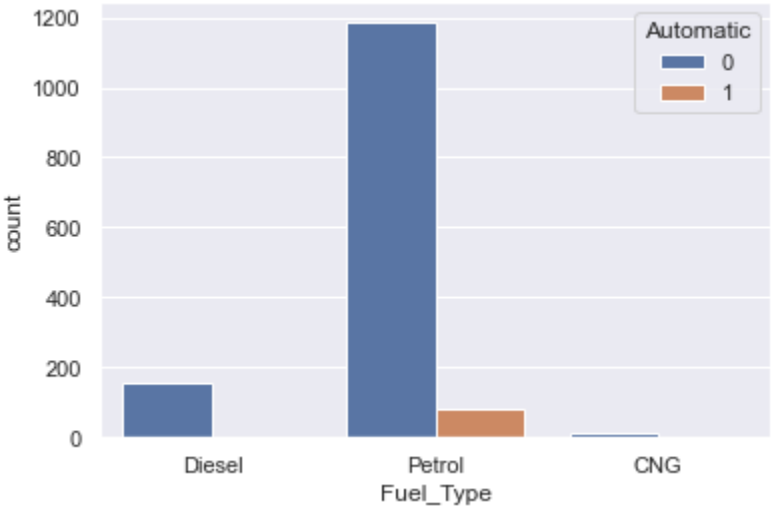
```
1)Grouped bar plot of fuel_Type and automatic
```

In [25]:

```
sns.countplot(x="Fuel_Type",data=cars_data,hue="Automatic")
#hue fives the relation between Fuel_type and automatic
```

Out[25]:

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



Box and whisker plot numerical variable

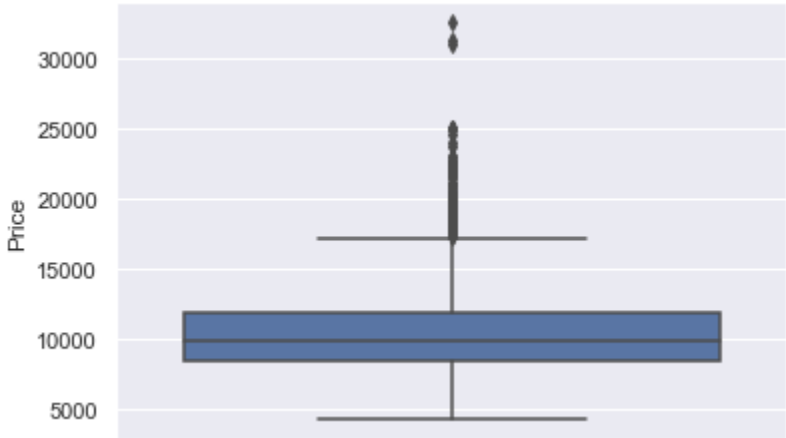
```
Box and whisker plot of price to visuaaly interpret the five number
summary
1)min
2)max
3) 1st quartile
4)2nd quaritile
5)3rd quaritile
```

In [26]:

```
sns.boxplot(y=cars_data['Price'])
#y will be repesentation of price
#it is called boz and whisker because it has box as well as whisker to
#line at 5000 is called as low whisker
#line at between 1500 and 20000 are called as high whisker it excludes
#25 % is less than 5000 1 st quartile
#50 % is equal to the beloww 8000 hence called 2nd quartile
#15,00 to 20,00 is called as third quartile
#the value above the third quaritile or 1st quartile  is called as outl
#
```

Out[26]:

```
<AxesSubplot:ylabel='Price'>
```



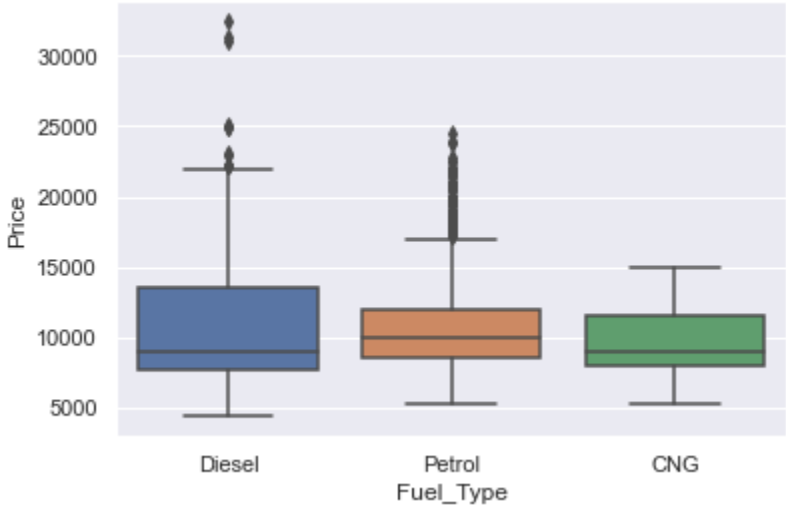
Box and whisker plot (numerical vs categorical variable)

In [27]:

```
sns.boxplot(x=cars_data['Fuel_Type'],y=cars_data['Price'])
#from the data we see that petrol has highest price because its median
#there are extreme values
```

Out[27]:

```
<AxesSubplot:xlabel='Fuel_Type', ylabel='Price'>
```



Grouped box and whisker plot

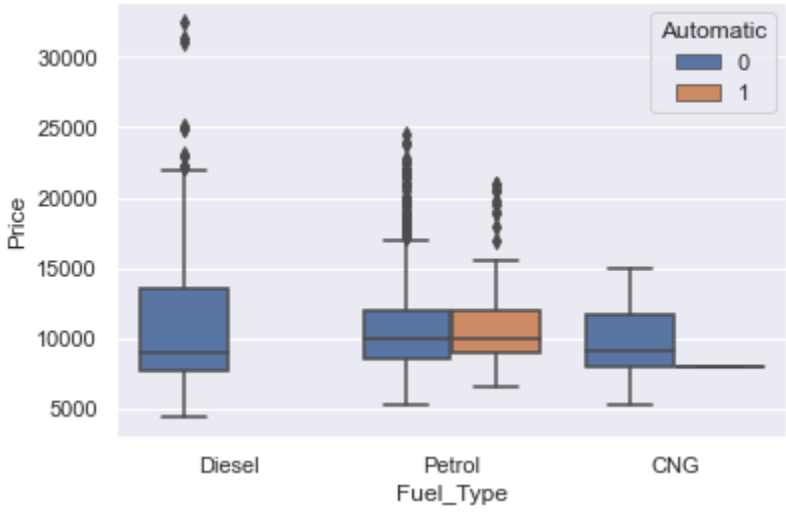
```
Grouped box and whisker plot of price vs fueltype and automatic
```

In [28]:

```
sns.boxplot(x=cars_data['Fuel_Type'],y=cars_data['Price'],hue='Automati
#the 1 type data is not present in diesel and cng hence we did not get
```

Out[28]:

```
<AxesSubplot:xlabel='Fuel_Type', ylabel='Price'>
```



Box-whisker plot and histogram

In [29]:

```
#lets plot box-whisker plot and histogram on the same window
#split the plotting window into 2 parts
```

In [30]:

```
f,(ax_box,ax_hist)=plt.subplots(2,gridspec_kw={"hieght_ratios":(.15,.85
#2 means we have 2 rows means 2 parts
#there is larger space for first wndow and small space for second windo
#gridspec you can set heights
#larger space for second window and less space for first window
#and sav it two variable called f and brackets
#f is figure and ax_box and ax_hist is the object
#now add create two plot
sns.boxplot(carsdata["Price"],ax-ax_box)
sns.distplot(cars_data["Price"],ax=ax_hist,kde=False)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-30-12d2791d3065> in <module>
----> 1 f,(ax_box,ax_hist)=plt.subplots(2,gridspec_kw={"hieght_ratios":(.15,.85)})
      2 #2 means we have 2 rows means 2 parts
      3 #there is larger space for first wndow and small space for second window
      4 #gridspec you can set heights
      5 #larger space for second window and less space for first window

~\anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py in wrapper(*args, **kwargs)
    449                 "parameter will become keyword-only %(removal)s.",
    450                 name=name, obj_type=f"parameter of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

~\anaconda3\lib\site-packages\matplotlib\pyplot.py in subplots(nrows, ncols, sharex, sharey, squeeze, subplot_kw, gridspec_kw, **fig_kw)
   1270     """
   1271     fig = figure(**fig_kw)
-> 1272     axs = fig.subplots(nrows=nrows, ncols=ncols, sharex=sharex, sharey=sharey,
   1273                        squeeze=squeeze, subplot_
```

```
kw=subplot_kw,
   1274                         gridspec_kw=gridspec_kw)

~\anaconda3\lib\site-packages\matplotlib\cbook\deprecati
on.py in wrapper(*args, **kwargs)
    449                 "parameter will become keyword-o
nly %(removal)s.",
    450                 name=name, obj_type=f"parameter
 of {func.__name__}()")
--> 451         return func(*args, **kwargs)
    452
    453     return wrapper

~\anaconda3\lib\site-packages\matplotlib\figure.py in su
bplots(self, nrows, ncols, sharex, sharey, squeeze, subp
lot_kw, gridspec_kw)
   1520         if gridspec_kw is None:
   1521             gridspec_kw = {}
-> 1522         return (self.add_gridspec(nrows, ncols,
 figure=self, **gridspec_kw)
   1523                 .subplots(sharex=sharex, sharey=
sharey, squeeze=squeeze,
   1524                           subplot_kw=subplot_k
w))

~\anaconda3\lib\site-packages\matplotlib\figure.py in ad
d_gridspec(self, nrows, ncols, **kwargs)
   2802
   2803         _ = kwargs.pop('figure', None)  # pop in
case user has added this...
-> 2804         gs = GridSpec(nrows=nrows, ncols=ncols,
 figure=self, **kwargs)
   2805         self._gridspecs.append(gs)
   2806         return gs

TypeError: __init__() got an unexpected keyword argument
'hieght_ratios'
```

```
<Figure size 432x288 with 0 Axes>
```

Pairwise plot

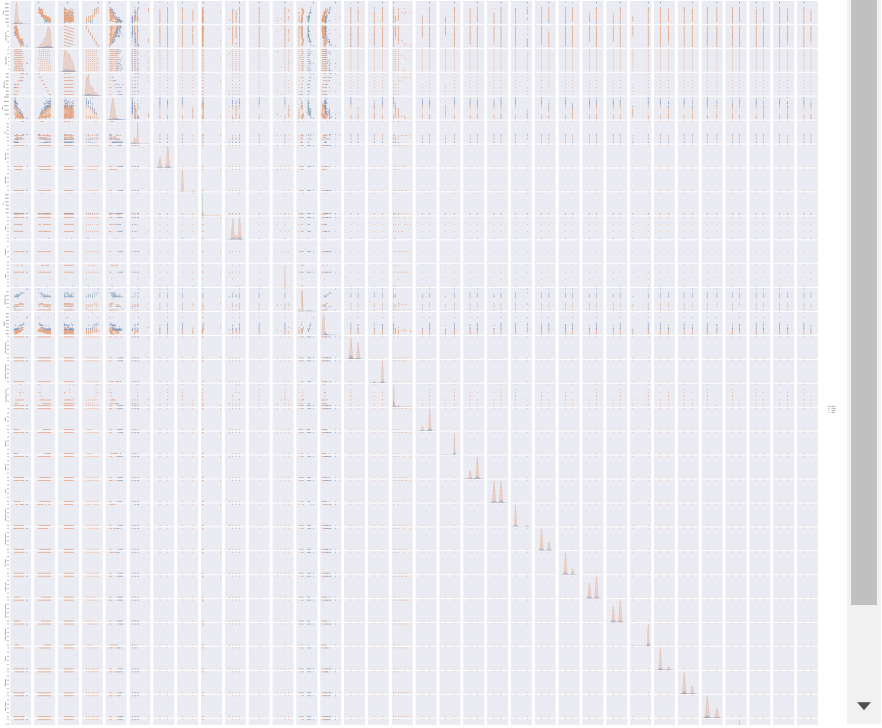
```
1)It is used to plot pairwise relationships in a dataset
```

2)creates scatter plots for joint relationship and histogram for univerate disributions

In [65]:

```
sns.pairplot(cars_data,kind="scatter",hue="Fuel_Type")
plt.show()
#pairplot is from sea born library
#cars_data is data frame because we will see all the variable
#kind is scatter
#
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:305: UserWarning: Dataset has 0 variance; ski
pping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:305: UserWarning: Dataset has 0 variance; ski
pping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:305: UserWarning: Dataset has 0 variance; ski
pping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:305: UserWarning: Dataset has 0 variance; ski
pping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:305: UserWarning: Dataset has 0 variance; ski
pping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:305: UserWarning: Dataset has 0 variance; ski
pping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:305: UserWarning: Dataset has 0 variance; ski
pping density estimate.
  warnings.warn(msg, UserWarning)
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:305: UserWarning: Dataset has 0 variance; ski
pping density estimate.
  warnings.warn(msg, UserWarning)
```



Identifying the missing values

In [66]:

```
import pandas as pd
cars_data=pd.read_csv('TT.CSV',index_col=0,na_values=["??"])
```

In [67]:

```
#creating copy of orginal data
cars_data2=cars_data.copy()
cars_data3=cars_data2.copy()
```

In [ ]:

```

```

```
1)In pandas dataframes, missing data is represented by NAN(an acronym
for not a number)
2)To check null values on pandas dataframes, isnull() and isna() are
used
```

```
3)these functions returns a dataframe of boolean values which are
True for NaN values
```

In [68]:

```
#code to identify null values
```

```
Dataframe.isna.sum().Dataframe.isnull.sum()
To check the count of sum of each null value in the column
```

In [69]:

```
cars_data2.isna().sum()
```

Out[69]:

```
Model                  0
Price                  0
Age_08_04              0
Mfg_Month              7
Mfg_Year               0
KM                     0
Fuel_Type              0
HP                     0
Met_Color              1
Automatic              2
cc                     4
Doors               1049
Cylinders           1051
Gears               1047
Quarterly_Tax       1047
Weight              1047
Mfr_Guarantee       1048
BOVAG_Guarantee     1048
Guarantee_Period    1050
ABS                 1047
Airbag_1            1046
Airbag_2            1046
Airco               1046
Automatic_airco     1046
Boardcomputer       1046
CD_Player           1046
Central_Lock        1046
Powered_Windows     1046
Power_Steering      1046
Radio               1046
Mistlamps           1046
Sport_Model         1046
Backseat_Divider    1046
Metallic_Rim        1046
Radio_cassette      1046
Tow_Bar             1046
dtype: int64
```

In [70]:

```
cars_data2.isnull().sum()
```

Out[70]:

```
Model                  0
Price                  0
Age_08_04              0
Mfg_Month              7
Mfg_Year               0
KM                     0
Fuel_Type              0
HP                     0
Met_Color              1
Automatic              2
cc                     4
Doors               1049
Cylinders           1051
Gears               1047
Quarterly_Tax       1047
Weight              1047
Mfr_Guarantee       1048
BOVAG_Guarantee     1048
Guarantee_Period    1050
ABS                 1047
Airbag_1            1046
Airbag_2            1046
Airco               1046
Automatic_airco     1046
Boardcomputer       1046
CD_Player           1046
Central_Lock        1046
Powered_Windows     1046
Power_Steering      1046
Radio               1046
Mistlamps           1046
Sport_Model         1046
Backseat_Divider    1046
Metallic_Rim        1046
Radio_cassette      1046
Tow_Bar             1046
dtype: int64
```

In [71]:

```
#to check the rows that have one column is missing or more than one col
missing=cars_data2[cars_data2.isnull().any(axis=1)]#the axis=1 is missi
print(missing)#we get amount of rows were single column is missing
```

```
                                                   Model
Price  Age_08_04  \
Id
12    TOYOTA Corolla 1.8 16V VVTLI 3DR T SPORT BNS 2...
19950         22
16    TOYOTA Corolla 1.8 16V VVTLI 3DR T SPORT 2/3-D...
22000         28
17    ?TOYOTA Corolla 1.8 16V VVTLI 3DR T SPORT 2/3-...
22750         30
18    ?TOYOTA Corolla 1.6 VVTI Linea Terra Comfort 2...
17950         24
19               TOYOTA Corolla 1.6 16v L.SOL 2/3-Doors
16750         24
...                                                  ...
...        ...
1128  TOYOTA Corolla 1.6 Linea Terra / Keuze uit 700...
7460         80
1129  TOYOTA Corolla 1.6 16V WAGON LINEA TERRA Stati...
9250         76
1130  TOYOTA Corolla 1.6 16V HATCHB LINEA LUNA 2/3-D...
7250         80
1131          TOYOTA Corolla 1.3 16V HATCHB S 2/3-Doors
6500         72
1132  TOYOTA Corolla 1.6 16V SEDAN LINEA TERRA 4/5-D...
6800         80

      Mfg_Month  Mfg_Year       KM Fuel_Type     HP  Met
_Color  Automatic  \
Id
12         11.0      2002  43610.0    Petrol  192.0
0.0        0.0
16          5.0      2002  18739.0    Petrol  192.0
0.0        0.0
17          NaN      2002  34000.0    Petrol  192.0
NaN        0.0
18          9.0      2002  21716.0    Petrol  110.0
1.0        0.0
19          9.0      2002  25563.0    Petrol  110.0
0.0        NaN
```

```
...          ...        ...       ...        ...    ...
...         ...
1128         1.0         0      0.0         0    0.0
1.0         0.0
1129         5.0         0      0.0         0    1.0
0.0         0.0
1130         1.0         0      1.0         0    1.0
0.0         0.0
1131         9.0         0      0.0         1    1.0
1.0         0.0
1132         1.0         1      0.0         0    0.0
0.0         1.0

      ...  Central_Lock  Powered_Windows  Power_Steering
Radio  Mistlamps  \
Id    ...
12    ...           1.0              1.0             1.0
0.0         1.0
16    ...           1.0              1.0             1.0
0.0         1.0
17    ...           1.0              1.0             1.0
0.0         1.0
18    ...           1.0              1.0             1.0
1.0         0.0
19    ...           1.0              1.0             1.0
0.0         1.0
...   ...           ...              ...             ...
...         ...
1128  ...           NaN              NaN             NaN
NaN         NaN
1129  ...           NaN              NaN             NaN
NaN         NaN
1130  ...           NaN              NaN             NaN
NaN         NaN
1131  ...           NaN              NaN             NaN
NaN         NaN
1132  ...           NaN              NaN             NaN
NaN         NaN

      Sport_Model  Backseat_Divider  Metallic_Rim  Radio
_cassette  Tow_Bar
Id
12            1.0               1.0           1.0
0.0      0.0
16            1.0               1.0           1.0
0.0      0.0
```

```
17            0.0                1.0           1.0
0.0      0.0
18            0.0                0.0           0.0
1.0      1.0
19            0.0                0.0           0.0
0.0      0.0
...           ...                ...           ...
...      ...
1128          NaN                NaN           NaN
NaN      NaN
1129          NaN                NaN           NaN
NaN      NaN
1130          NaN                NaN           NaN
NaN      NaN
1131          NaN                NaN           NaN
NaN      NaN
1132          NaN                NaN           NaN
NaN      NaN

[1068 rows x 36 columns]
```

approches to fill the missing values

```
Two ways of approch
1)Fill the missing values by mean/mesdian in case of numerical
variable
2)fill the missing values with the class which has maximum
count(mode=the maximmum number repeated),in case of categorical value
```

Imputing the numerical values

```
1)Look at the description to know whether numerical variables should
be inputed with mean or median because if we have extreme values in
the data then we cant fill mean values
Dtaframe.describe()
2)generte the descriptive statics that summarize the central tendency
,dispersion and shape of dataset's distribution excluding nan values
```

In [72]:

```
cars_data2.describe()
#count gives how maany no is under the price
#mean is the average of the price
#25%gives how many vars is under
#50%gives the median
#there is no diffrence between mean and median IN aGE SO WE CAN MEAN HE
#THERE IS BIF DIFFRENCE MEAN AND MEDIAN IN KM SO WE ARE USING MEDIAN  H
```

Out[72]:

	Price	Age_08_04	Mfg_Month	Mfg_Year	
count	1436.000000	1436.000000	1429.000000	1436.000000	1436.0
mean	10730.824513	55.947075	5.939686	542.972145	20092.3
std	3626.964585	18.599988	3.788111	889.277199	36126.0
min	4350.000000	1.000000	1.000000	0.000000	0.0
25%	8450.000000	44.000000	3.000000	0.000000	0.0
50%	9900.000000	61.000000	6.000000	0.000000	0.0
75%	11950.000000	70.000000	9.000000	1998.000000	34234.5
max	32500.000000	80.000000	19.183173	2002.000000	172950.9

8 rows × 34 columns

In [73]:

```
#CALCULATING THe mean value of age
cars_data2['Age_08_04'].mean()
```

Out[73]:

```
55.94707520891365
```

In [74]:

```
# to fill NA/Nan values using the specified value
#Dataframe.fillna()
cars_data2['Age_08_04'].fillna(cars_data2['Age_08_04'].mean(),inplace=T
#inplace =true fill alll the missing values in the data frame itself
```

In [75]:

```
#imputing values of 'KM'
#calculating the median value of KM variable
cars_data2['KM'].median()
```

Out[75]:

```
0.0
```

In [76]:

```
cars_data2['KM'].fillna(cars_data2['KM'].median(),inplace=True)
```

Imputing the missing values of hp

In [77]:

```
#there is no big diffrence between mean and median values so we can use
cars_data2['HP'].fillna(cars_data2['HP'].mean(),inplace=True)
```

In [78]:

```
cars_data2.isnull().sum()#there is no missing values in age km and hp
```

Out[78]:

```
Model                  0
Price                  0
Age_08_04              0
Mfg_Month              7
Mfg_Year               0
KM                     0
Fuel_Type              0
HP                     0
Met_Color              1
Automatic              2
cc                     4
Doors               1049
Cylinders           1051
Gears               1047
Quarterly_Tax       1047
Weight              1047
Mfr_Guarantee       1048
BOVAG_Guarantee     1048
Guarantee_Period    1050
ABS                 1047
Airbag_1            1046
Airbag_2            1046
Airco               1046
Automatic_airco     1046
Boardcomputer       1046
CD_Player           1046
Central_Lock        1046
Powered_Windows     1046
Power_Steering      1046
Radio               1046
Mistlamps           1046
Sport_Model         1046
Backseat_Divider    1046
Metallic_Rim        1046
Radio_cassette      1046
Tow_Bar             1046
dtype: int64
```

Imputing the missing values of categorical

```
1)for any categorical value fill the missing values with most
occuring categorical data
```

In [79]:

```
#series.value_count()
```

```
1)returns a series containing unique values
2)the output is in descending order so the firstelement is the most
frequently occuring element
3)exclude na values by default
```

In [87]:

```
cars_data2['Fuel_Type'].value_counts()
#here the maximum occuring value is 0
```

Out[87]:

```
0         766
Petrol    352
1         280
Diesel     38
Name: Fuel_Type, dtype: int64
```

In [81]:

```
cars_data2['Fuel_Type'].value_counts().index[0]#here we get the first l
```

Out[81]:

```
'0'
```

In [82]:

```
#to fill na values by the specified value
cars_data2['Fuel_Type'].fillna(cars_data2['Fuel_Type'].value_counts().i
#inplace=true because we are replacing existing dataframe itself
```

Imputing the missing values of

'Metcolor'

In [83]:

```
#ton get mode value of MetColOr
cars_data2['Met_Color'].mode()
#it give value along with the index
#it gives the value of index which is zero
#we did not get index in mean or meadian becuse they are scalar but mod
```

Out[83]:

```
0    0.0
dtype: float64
```

In [84]:

```
cars_data2['Met_Color'].fillna(cars_data2['Met_Color'].mode()[0],inplac
#we have given index because output has index
```

Imputing missing values using lambda function

In [85]:

```
cars_data3=cars_data3.apply(lambda x:x.fillna(x.mean()) if x.dtype=='fl
#we replace the both categorical and numerical value
```

Introduction to case study

```
1)classification problem above cerytain value and below certain value
```

```
1)Subsidy Inc. delivers subsidies to individual based on their income
2)Accurate income data is one of the hardest piece of data to obtain
across the world.
3)Subsidy Inc. has obtained a large data set of authenciated data on
individual income, demographic parameter and a few finicial parameter
4)Subsidy Inc. wishes us to.
Devlop an income classifier system for individual

The objective is to:
```

```
Simplify the data system by reducing the number of variables to be
studied,without scarificing too much of accuracy. Such a system would
help Subsidy Inc. in planning subsidy outlay,monitoring and
preventing missue
```

In [86]:

```
#problem conceptualization
```

```
1)Devlop an income classifier for individuals with reduced no of
variables
2)problem characterziation -classification
```

In [50]:

```
#Apriori known:
```

In [51]:

```
Dapendent variable
categoeical(binary)
Independent variable
numerical +categorical
```

```
  File "<ipython-input-51-dd9e8dc7c5e1>", line 1
    Dapendent variable
                      ^
SyntaxError: invalid syntax
```

Framework for solution conceptualization

```
1)Identify if data is clean-if there is categories for all the
variable all the data should be the same category means they should
be exactly one type of data type
```

```
2)look for missing values-so there are rows which are missing so we
can handle it in two ways1) remove all the samples that have
incomplete data but by this way we can lost most of the data 2)some
how we fill the missing values the problem is we introduce
artificality whether we do not know it is right or wrong
3)identify variables influening salary status and look for possible
relationship between variables and if they are correalated
1)correlation 2)chisquare 3)boxplot 4)scatter plot
4)identify catregories can be combined
5)Build models with reduced number of variables to classify the
individual's salary status to plan subsidy outlay,monitor and prevent
missue
```

In [52]:

```
import pandas as pd #to work with datafrrame
import numpy as np #TO PERFORM NUMERICAL operation
import seaborn as sns#to visualise data
from sklearn.model_selection import train_test_split#to participation t
#we are importing train_test_split from the package sklearn
#model selection is package under the sklearn
from sklearn.linear_model import LogisticRegression#importing library f
#We are importing logisticRegression from sklearn liniar model
from sklearn.metrics import accuracy_score,confusion_matrix
#importing performance matrics- accuracy_score,confusion_matrix
```

In [53]:

```
data_income=pd.read_csv('income(1).csv')
```

In [54]:

```
data=data_income.copy()
#ecplorattory data analysis:
#1)getting to know the data
#2)data processing(misssing calues how to find missing values and deal
#3)cross tables and visualisation
```

In [55]:

```
#getting to know the data
print(data.info())
#to know the information about datatypes
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31978 entries, 0 to 31977
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            31978 non-null  int64 
 1   JobType        31978 non-null  object
 2   EdType         31978 non-null  object
 3   maritalstatus  31978 non-null  object
 4   occupation     31978 non-null  object
 5   relationship   31978 non-null  object
 6   race           31978 non-null  object
 7   gender         31978 non-null  object
 8   capitalgain    31978 non-null  int64 
 9   capitalloss    31978 non-null  int64 
 10  hoursperweek   31978 non-null  int64 
 11  nativecountry  31978 non-null  object
 12  SalStat        31978 non-null  object
dtypes: int64(4), object(9)
memory usage: 2.1+ MB
None
```

In [56]:

```
#check the missing values
data.isnull().sum()
```

Out[56]:

```
age               0
JobType           0
EdType            0
maritalstatus     0
occupation        0
relationship      0
race              0
gender            0
capitalgain       0
capitalloss       0
hoursperweek      0
nativecountry     0
SalStat           0
dtype: int64
```

In [57]:

```
#summary of numerical variables
summary_num=data.describe()
print(summary_num)
```

```
                 age   capitalgain   capitalloss  hourspe
rweek
count  31978.000000  31978.000000  31978.000000  31978.0
00000
mean      38.579023   1064.360623     86.739352     40.4
17850
std       13.662085   7298.596271    401.594301     12.3
45285
min       17.000000      0.000000      0.000000      1.0
00000
25%       28.000000      0.000000      0.000000     40.0
00000
50%       37.000000      0.000000      0.000000     40.0
00000
75%       48.000000      0.000000      0.000000     45.0
00000
max       90.000000  99999.000000   4356.000000     99.0
00000
```

In [58]:

```
#summary of categorical values
summary_cate=data.describe(include="O")
print(summary_cate)
#it gives the summary of categorical values
#here O object
#how many unique calue present in job
#top gives most occuring values in column
#freq gives how many times top value get repeated
#
```

```
         JobType    EdType        maritalstatus       oc
cupation relationship  \
count      31978     31978                31978
31978        31978
unique         9        16                    7
15            6
top      Private   HS-grad   Married-civ-spouse   Prof-s
pecialty      Husband
freq       22286     10368                14692
4038        12947

          race gender   nativecountry
SalStat
count    31978  31978           31978
31978
unique       5      2              41
2
top      White   Male   United-States   less than or equ
al to 50,000
freq     27430  21370           29170
24283
```

In [59]:

```
#frequecy of each values in a column
data['JobType'].value_counts()#we have the some missing values in form
```

Out[59]:

```
 Private             22286
 Self-emp-not-inc     2499
 Local-gov            2067
 ?                    1809
 State-gov            1279
 Self-emp-inc         1074
 Federal-gov           943
 Without-pay            14
 Never-worked            7
Name: JobType, dtype: int64
```

In [60]:

```
data['occupation'].value_counts
```

Out[60]:

```
<bound method IndexOpsMixin.value_counts of 0
Adm-clerical
1            Armed-Forces
2          Prof-specialty
3            Craft-repair
4                   Sales
               ...
31973     Farming-fishing
31974     Protective-serv
31975        Adm-clerical
31976        Adm-clerical
31977      Prof-specialty
Name: occupation, Length: 31978, dtype: object>
```

In [61]:

```
print(np.unique(data['JobType']))#here ? is present instead of nan valu
```

```
[' ?' ' Federal-gov' ' Local-gov' ' Never-worked' ' Priv
ate'
 ' Self-emp-inc' ' Self-emp-not-inc' ' State-gov' ' With
out-pay']
```

In [62]:

```
print(np.unique(data['occupation']))#here ? is represented instead of n
```

```
[' ?' ' Adm-clerical' ' Armed-Forces' ' Craft-repair' '
Exec-managerial'
 ' Farming-fishing' ' Handlers-cleaners' ' Machine-op-in
spct'
 ' Other-service' ' Priv-house-serv' ' Prof-specialty' '
Protective-serv'
 ' Sales' ' Tech-support' ' Transport-moving']
```

In [63]:

```
import pandas as pd
#now we read the data and replace ? with nan values
data=pd.read_csv('income(1).csv',na_values=[" ?"])
```

In [64]:

```
#now finding the exact value of missing data
data.isnull().sum()
#two missing values job type and occupation
```

Out[64]:

```
age                 0
JobType          1809
EdType              0
maritalstatus       0
occupation       1816
relationship        0
race                0
gender              0
capitalgain         0
capitalloss         0
hoursperweek        0
nativecountry       0
SalStat             0
dtype: int64
```

In [44]:

```
#now we are lokking either in a row one column is missing or both colum
missing=data[data.isnull().any(axis=1)]
#AXIS=1 means it consider at least column value is missing
```

```
points to note
1)missing values in job type=1809
2)missing values in occupation =1816
3)there are 1809 rows were two speific columns i.e  occupation &
jobtype have missing values
4)1816-1809 you still have occupation unfilled for these 7 rows
because job type is never worked.
```

In [45]:

```
data2=data.dropna(axis=0)
#it means the value where the nan values is present delete that row
```

In [46]:

```
#relationship between independent variable
correlation =data2.corr()
print(correlation)
#here the most of the values are near to zero then they are less correl
```

```
                   age  capitalgain  capitalloss  hoursp
erweek
age           1.000000     0.080154     0.060165      0.
101599
capitalgain   0.080154     1.000000    -0.032229      0.
080432
capitalloss   0.060165    -0.032229     1.000000      0.
052417
hoursperweek  0.101599     0.080432     0.052417      1.
000000
```

In [47]:

```
#extracting the columns names
data2.columns
```

Out[47]:

```
Index(['age', 'JobType', 'EdType', 'maritalstatus', 'occ
upation',
       'relationship', 'race', 'gender', 'capitalgain',
'capitalloss',
       'hoursperweek', 'nativecountry', 'SalStat'],
      dtype='object')
```

In [48]:

```
#gender proportion table
gender=pd.crosstab(index=data2["gender"],columns='count',normalize=True
print(gender)
#male is high in frequency correspond to 67%
```

```
col_0       count
gender
 Female  0.324315
 Male    0.675685
```

In [49]:

```
#gender vs salary status
gender_salstat=pd.crosstab(index=data2["gender"],columns=["SalStat"],ma
```

In [50]:

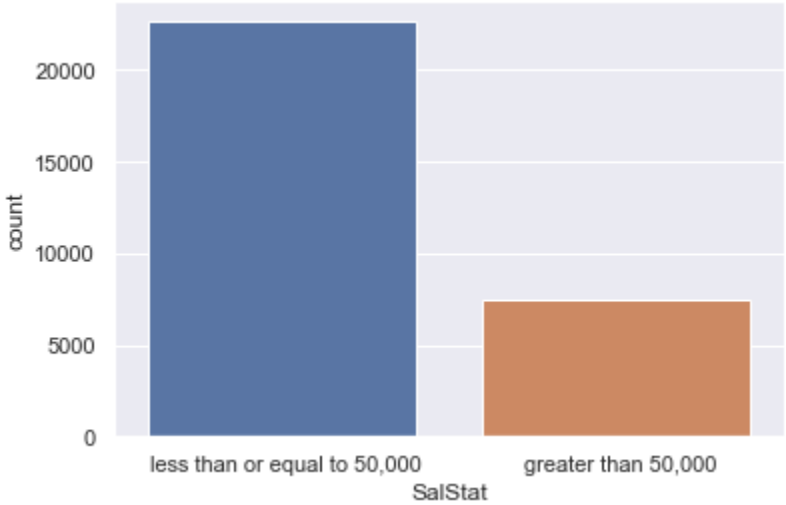
```
print(gender_salstat)
```

```
col_0    SalStat
gender
 Female      1.0
 Male        1.0
All          1.0
```

In [53]:

```
SalStat=sns.countplot(data2["SalStat"])
#on classification we have to know how balanced classification are 75%
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\_decor
ators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid p
ositional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```



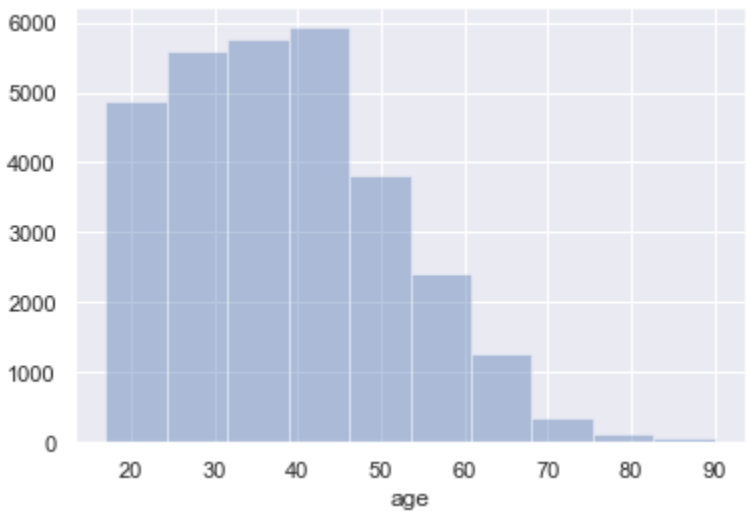
In [55]:

```
#histogram of age
sns.distplot(data2['age'],bins=10,kde=False)
#people with GE 20-40 AGE ARE HIGH IN FREQUENCY
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\distri
butions.py:2551: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Ple
ase adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an
axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[55]:

```
<AxesSubplot:xlabel='age'>
```



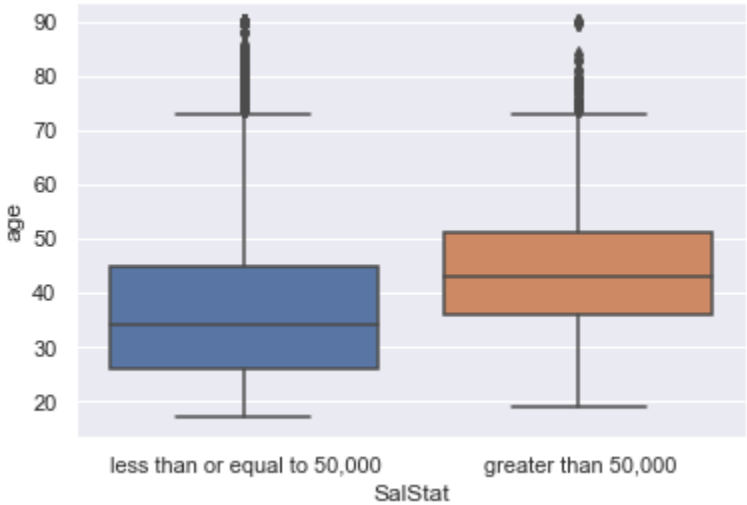
In [56]:

```
#BOX PLOT -aGE VS SALARY
sns.boxplot('SalStat','age',data=data2)
data2.groupby("SalStat")['age'].median()
#people with age 35-40are more likely to earn more than 50,000
#people with age 25-35 are less likely to earn 50000
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\_decor
ators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an er
ror or misinterpretation.
  warnings.warn(
```

Out[56]:

```
SalStat
 greater than 50,000             43
 less than or equal to 50,000    34
Name: age, dtype: int64
```



Exploratory Data analysis

In [ ]: