Problem Statement

Car Data- Here, we will apply k-means clustering for grouping the similar cars in one cluster

In [16]:

```
#Import all the neccessary modules
#Import all the neccessary modules
import pandas as pd
import numpy as np
import os
import seaborn as sns
from sk.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
%matplotlib inline
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-16-87f0a5a98d83> in <module>
      5 import os
      6 import seaborn as sns
----> 7 from sk.cluster import KMeans
      8 from sklearn.preprocessing import LabelEncoder
      9 from sklearn.preprocessing import MinMaxScaler

ModuleNotFoundError: No module named 'sk'
```

1. Load the Cars Data file into Python DataFrame and view top 10 rows

In [20]:

```
data = pd.read_csv(r'C:\Users\Anas Khanooni\Documents\ANAS KHANOONI\ANAS POST-GRD IN AI\Ass
```

In [21]:

```
data.head()
```

Out[21]:

	mpg	cyl	disp	hp	wt	acc	yr	origin	car_name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

2. Printing the datatypes of each column and the shape of the dataset. Performing descriptive analysis

In [22]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   mpg       398 non-null    float64
 1   cyl       398 non-null    int64  
 2   disp      398 non-null    float64
 3   hp        398 non-null    object 
 4   wt        398 non-null    int64  
 5   acc       398 non-null    float64
 6   yr        398 non-null    int64  
 7   origin    398 non-null    int64  
 8   car_name  398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [23]:

```
data.describe
```

Out[23]:

```
<bound method NDFrame.describe of       mpg  cyl   disp   hp    wt   acc  yr  
origin                  car_name
0    18.0    8  307.0  130  3504  12.0  70       1  chevrolet chevelle malib
u
1    15.0    8  350.0  165  3693  11.5  70       1          buick skylark 32
0
2    18.0    8  318.0  150  3436  11.0  70       1         plymouth satellit
e
3    16.0    8  304.0  150  3433  12.0  70       1             amc rebel ss
t
4    17.0    8  302.0  140  3449  10.5  70       1               ford torin
o
..    ...  ...    ...  ...   ...   ...  ..     ...                       
...
393  27.0    4  140.0   86  2790  15.6  82       1           ford mustang g
l
394  44.0    4   97.0   52  2130  24.6  82       2                 vw picku
p
395  32.0    4  135.0   84  2295  11.6  82       1             dodge rampag
e
396  28.0    4  120.0   79  2625  18.6  82       1               ford range
r
397  31.0    4  119.0   82  2720  19.4  82       1                 chevy s-1
0

[398 rows x 9 columns]>
```

3. Checking for missing value check, incorrect data and perform imputation with mean, median and mode.

In [24]:

```
#Checking for the missing value
data.isna().sum()
```

Out[24]:

```
mpg         0
cyl         0
disp        0
hp          0
wt          0
acc         0
yr          0
origin      0
car_name    0
dtype: int64
```

In [25]:

```
# Na shows no missing value, but on careful data observation we could see "?" for hp values
data[data['hp']=="?"]
```

Out[25]:

	mpg	cyl	disp	hp	wt	acc	yr	origin	car_name
32	25.0	4	98.0	?	2046	19.0	71	1	ford pinto
126	21.0	6	200.0	?	2875	17.0	74	1	ford maverick
330	40.9	4	85.0	?	1835	17.3	80	2	renault lecar deluxe
336	23.6	4	140.0	?	2905	14.3	80	1	ford mustang cobra
354	34.5	4	100.0	?	2320	15.8	81	2	renault 18i
374	23.0	4	151.0	?	3035	20.5	82	1	amc concord dl

In [26]:

```
data['hp'].replace("?",np.nan, inplace=True)
```

In [28]:

```
# Now we try to impute with mean of respective cylinders, but before this we must see the d
# We would drop na values and check distribution before taking call on whether imputation w
import seaborn as sns
hp = data['hp'].dropna()
hp.count()
```

Out[28]:

392

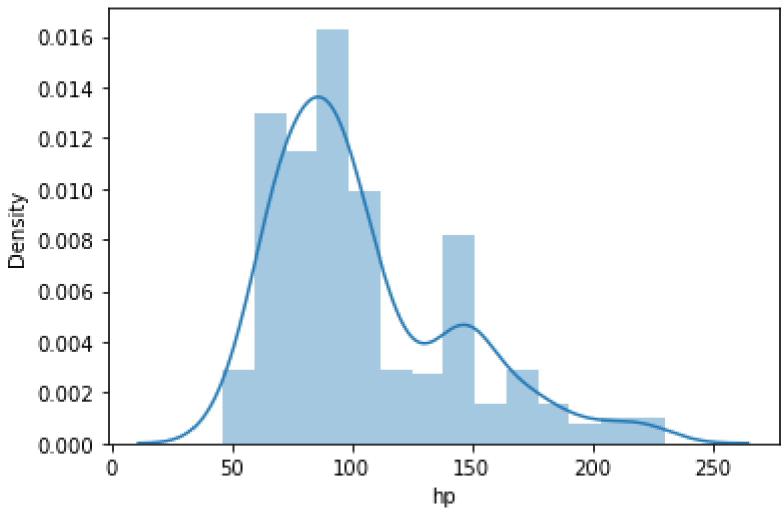
In [29]:

```
sns.distplot(pd.to_numeric(hp))
```

```
C:\Users\Anas Khanooni\anaconda3\lib\site-packages\seaborn\distributions.py:
2551: FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a figur
e-level function with similar flexibility) or `histplot` (an axes-level func
tion for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[29]:

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



In [36]:

```
# Since this does not look to be normally distributed, let us impute by using median
data['hp'].fillna((data['hp'].median()), inplace=True)
data['hp'] = data['hp'].astype('float')
```

In [37]:

```
data.dtypes
```

Out[37]:

```
mpg          float64
cyl            int64
disp         float64
hp           float64
wt             int64
acc          float64
yr             int64
origin         int64
car_name      object
dtype: object
```

4. Performing bi variate analysis incuding correalation, pairplots and state the inferences

In [38]:

```
data.corr(method='kendall')
```

Out[38]:

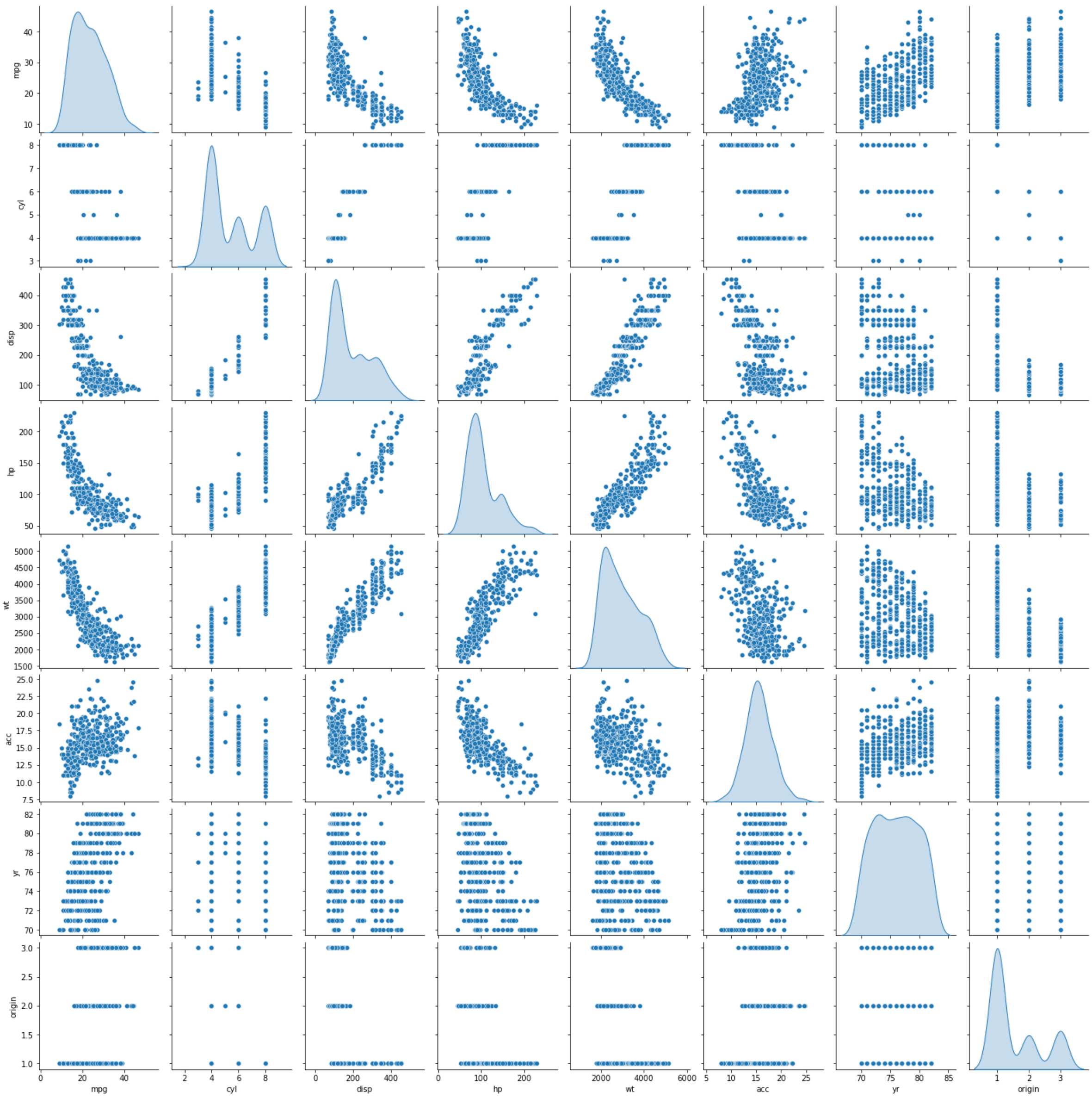
	mpg	cyl	disp	hp	wt	acc	yr	origin
mpg	1.000000	-0.686477	-0.679847	-0.673377	-0.694006	0.301096	0.413661	0.467249
cyl	-0.686477	1.000000	0.794854	0.682006	0.735481	-0.367194	-0.273742	-0.551610
disp	-0.679847	0.794854	1.000000	0.711556	0.800508	-0.352110	-0.218920	-0.570074
hp	-0.673377	0.682006	0.711556	1.000000	0.696368	-0.482267	-0.274888	-0.402494
wt	-0.694006	0.735481	0.800508	0.696368	1.000000	-0.268619	-0.196863	-0.496185
acc	0.301096	-0.367194	-0.352110	-0.482267	-0.268619	1.000000	0.196024	0.173055
yr	0.413661	-0.273742	-0.218920	-0.274888	-0.196863	0.196024	1.000000	0.136967
origin	0.467249	-0.551610	-0.570074	-0.402494	-0.496185	0.173055	0.136967	1.000000

In [39]:

```
sns.pairplot(data,diag_kind='kde')
```

Out[39]:

```
<seaborn.axisgrid.PairGrid at 0x1d2a493d9d0>
```



In [40]:

```
# Further dig into data shows max mpg is for 4 cylinders vehicles
# Origin as pointed earlier indicates production point so should be broken into dummy varia
# Year would be more effective if we can transorm this to calculate age of vehicle. This da
# subtract year from 83 to get the age
# Other continuous variables should be checked for outliers and should be normlized using z
```

In [43]:

```
# Calculate age of vechile
data['age'] = 83-data['yr']
data.head()
```

Out[43]:

	mpg	cyl	disp	hp	wt	acc	yr	origin	car_name	age
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu	13
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320	13
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite	13
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst	13
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino	13

In [44]:

```
#Convert origing into dummy variables (This again is subjected to business knowledge. We mi
# ... Inclusion is more to demonstrate on how to use categorical data)

one_hot = pd.get_dummies(data['origin'])
one_hot = one_hot.add_prefix('origin_')

# merge in main data frame
data = data.join(one_hot)
data.head()
```

Out[44]:

	mpg	cyl	disp	hp	wt	acc	yr	origin	car_name	age	origin_1	origin_2	origin_3
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu	13	1	0	0
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320	13	1	0	0
2	18.0	8	318.0	150.0	3436	11.0	70	1	plymouth satellite	13	1	0	0
3	16.0	8	304.0	150.0	3433	12.0	70	1	amc rebel sst	13	1	0	0
4	17.0	8	302.0	140.0	3449	10.5	70	1	ford torino	13	1	0	0

In [45]:

```
# Let us now remove duplicate/irrelevant columns

cars_new  = data.drop(['yr','origin','car_name'], axis =1)

cars_new.head()
```

Out[45]:

	mpg	cyl	disp	hp	wt	acc	age	origin_1	origin_2	origin_3
0	18.0	8	307.0	130.0	3504	12.0	13	1	0	0
1	15.0	8	350.0	165.0	3693	11.5	13	1	0	0
2	18.0	8	318.0	150.0	3436	11.0	13	1	0	0
3	16.0	8	304.0	150.0	3433	12.0	13	1	0	0
4	17.0	8	302.0	140.0	3449	10.5	13	1	0	0

6. Create new data frame with standardize variables and imputation for any missing/outliers

In [46]:

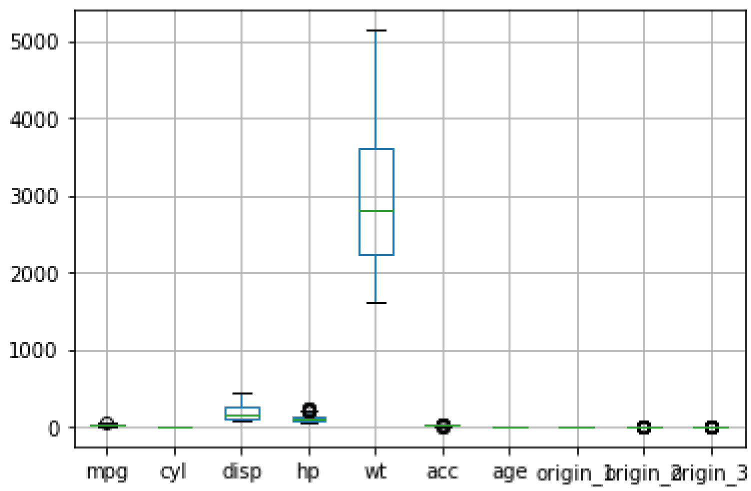
```
# Missing value check was done above and hp column was treated with median values
# Let us check for outliers
```

In [47]:

```
cars_new.boxplot()
```

Out[47]:

```
<AxesSubplot:>
```



In [52]:

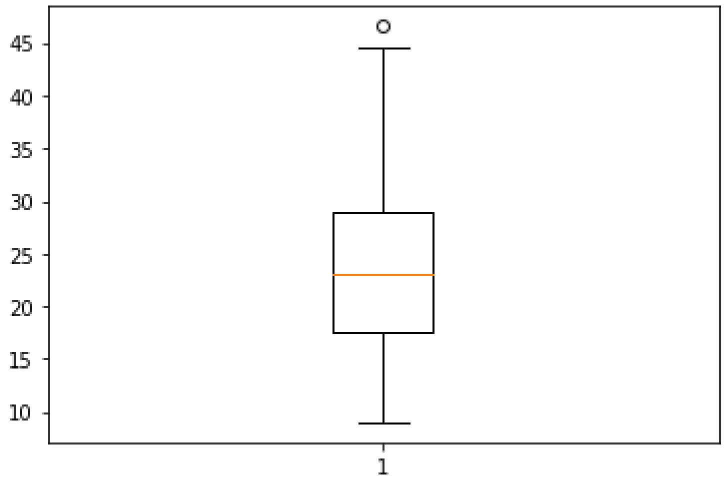
```
import matplotlib.pyplot as plt
```

In [53]:

```
# We could see some outliers for mpg, hp and acc
plt.boxplot(cars_new['mpg'])
```

Out[53]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1d2a74a0850>,
  <matplotlib.lines.Line2D at 0x1d2a852d040>],
 'caps': [<matplotlib.lines.Line2D at 0x1d2a852d370>,
  <matplotlib.lines.Line2D at 0x1d2a7f1aa60>],
 'boxes': [<matplotlib.lines.Line2D at 0x1d2a4e008e0>],
 'medians': [<matplotlib.lines.Line2D at 0x1d2a7f1a3a0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1d2a74172b0>],
 'means': []}
```

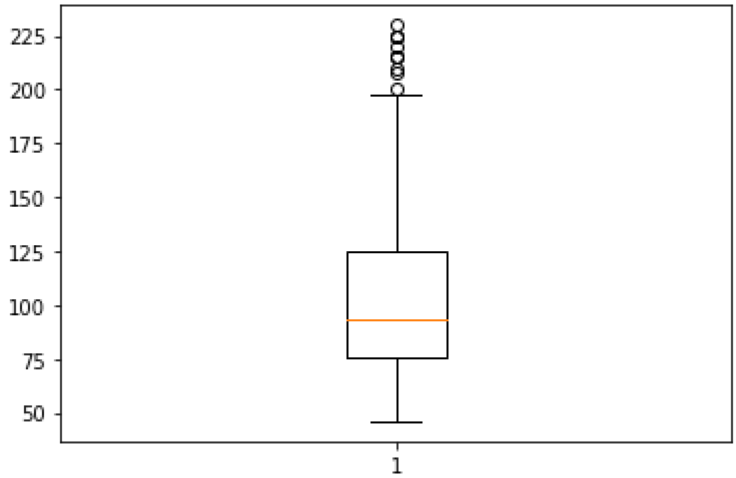


In [54]:

```
plt.boxplot(cars_new['hp'])
```

Out[54]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1d2a7de8d90>,
  <matplotlib.lines.Line2D at 0x1d2a7de8640>],
 'caps': [<matplotlib.lines.Line2D at 0x1d2a7de21c0>,
  <matplotlib.lines.Line2D at 0x1d2a7de2880>],
 'boxes': [<matplotlib.lines.Line2D at 0x1d2a7de8b80>],
 'medians': [<matplotlib.lines.Line2D at 0x1d2a7de26d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1d2a84db520>],
 'means': []}
```

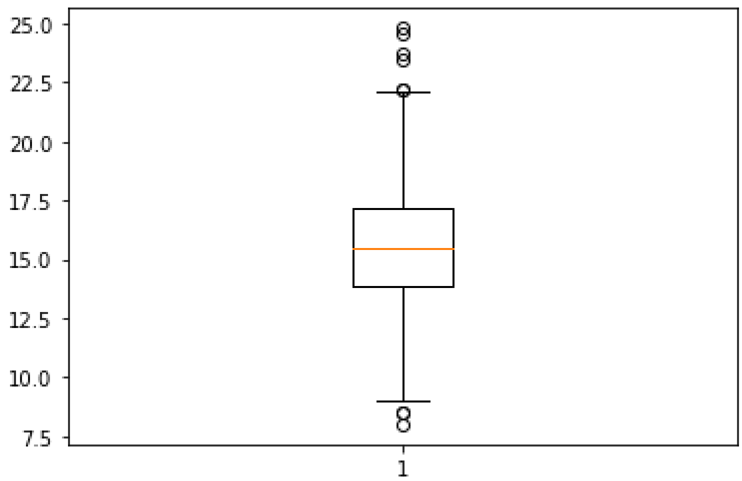


In [55]:

```
plt.boxplot(cars_new['acc'])
```

Out[55]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1d2a7ec9d00>,
  <matplotlib.lines.Line2D at 0x1d2a7de6400>],
 'caps': [<matplotlib.lines.Line2D at 0x1d2a7de66d0>,
  <matplotlib.lines.Line2D at 0x1d2a7de6d00>],
 'boxes': [<matplotlib.lines.Line2D at 0x1d2a7431ac0>],
 'medians': [<matplotlib.lines.Line2D at 0x1d2a7edfbe0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1d2a7eec640>],
 'means': []}
```



In [57]:

```
# Let us take logaritmic transform for hp, mpg and acc to remove outliers
cars_new['hp'] = np.log(cars_new['hp'])
cars_new["acc"] = np.log(cars_new['acc'])
cars_new['mpg'] = np.log(cars_new['mpg'])

cars_new.head()
```

Out[57]:

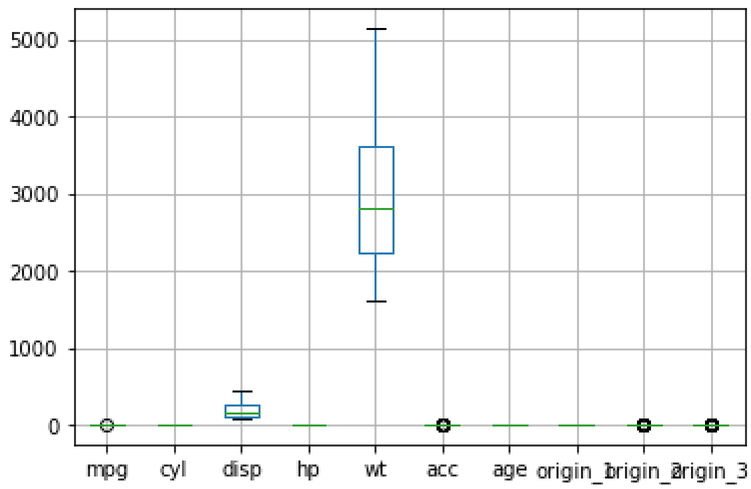
	mpg	cyl	disp	hp	wt	acc	age	origin_1	origin_2	origin_3
0	1.061385	8	307.0	1.582588	3504	0.910235	13	1	0	0
1	0.996229	8	350.0	1.630406	3693	0.892959	13	1	0	0
2	1.061385	8	318.0	1.611563	3436	0.874591	13	1	0	0
3	1.019781	8	304.0	1.611563	3433	0.910235	13	1	0	0
4	1.041412	8	302.0	1.597698	3449	0.855000	13	1	0	0

In [58]:

```
cars_new.boxplot()
```

Out[58]:

<AxesSubplot:>



In [59]:

```
# This looks better
# Now we will scale the variables
from scipy.stats import zscore

cars_new.dtypes
numeric_cols = cars_new.select_dtypes(include=[np.int64, np.float64]).columns
numeric_cols
cars_new[numeric_cols] = cars_new[numeric_cols].apply(zscore)
```

In [60]:

```
cars_new.head()
```

Out[60]:

	mpg	cyl	disp	hp	wt	acc	age	origin_1	origin_2	c
0	-0.577492	1.498191	1.090604	0.843143	0.630870	-1.358526	1.627426	1	0	
1	-1.162320	1.498191	1.503514	1.492969	0.854333	-1.614868	1.627426	1	0	
2	-0.577492	1.498191	1.196232	1.236902	0.550470	-1.887421	1.627426	1	0	
3	-0.950917	1.498191	1.061796	1.236902	0.546923	-1.358526	1.627426	1	0	
4	-0.756771	1.498191	1.042591	1.048484	0.565841	-2.178121	1.627426	1	0	

7. Creating appropriate clusters with the new data set

In [71]:

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
```

In [73]:

```
# Variables are now scaled. Let us now try to create clusters

cluster_range = range(1,15)
cluster_errors = []
for num_clusters in cluster_range:
    clusters = KMeans(num_clusters, n_init = 5)
    clusters.fit(cars_new)
    labels = clusters.labels_
    centroids = clusters.cluster_centers_
    cluster_errors.append(clusters.inertia_)

clusters_df = pd.DataFrame({"num_clusters": cluster_range, "cluster_errors": cluster_errors
clusters_df[0:15]
```

Out[73]:

	num_clusters	cluster_errors
0	1	3000.226131
1	2	1431.181714
2	3	1069.008744
3	4	875.277552
4	5	787.272965
5	6	719.139976
6	7	677.363200
7	8	631.724230
8	9	603.649034
9	10	566.338831
10	11	525.269291
11	12	495.488854
12	13	472.151350
13	14	463.138570

8. Identifying the appropriate clusters with result from the above question

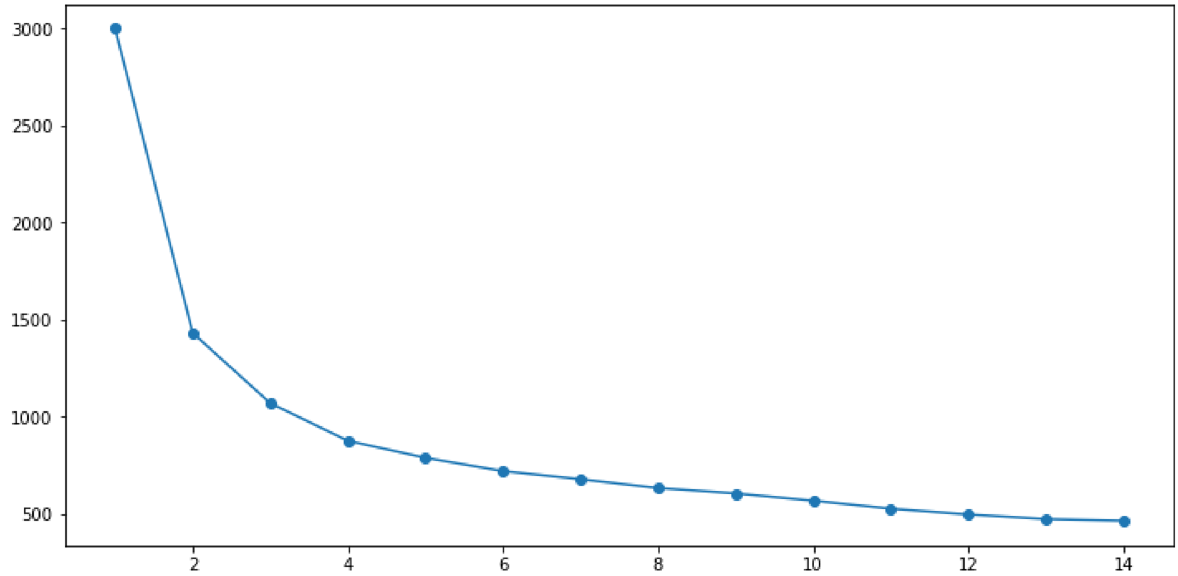
In [74]:

```
from matplotlib import cm

plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

Out[74]:

```
[<matplotlib.lines.Line2D at 0x1d2a95bb130>]
```



In [75]:

```
# We could see the bend at 4, so let us create 4 clusters

kmeans = KMeans(n_clusters=4, n_init = 5, random_state=12345)
kmeans.fit(cars_new)
```

Out[75]:

```
KMeans(n_clusters=4, n_init=5, random_state=12345)
```

9. Checking for no of values in each cluster and centres for each variables

In [76]:

```
# Check the number of data in each cluster

lables = kmeans.labels_
counts = np.bincount(labels[labels>=0])
print(counts)
```

```
[37 42 46 47 12 18 18 20 50 23 22 30 11 22]
```

In [78]:

```
# Distribution looks fine

# Let's check the centres in each group
centroids = kmeans.cluster_centers_
centroid_df = pd.DataFrame(centroids, columns = list(cars_new) )
centroid_df.transpose()
```

Out[78]:

	0	1	2	3
mpg	-1.343341e+00	-0.373931	1.005433	0.375068
cyl	1.498191e+00	0.441337	-0.804104	-0.869117
disp	1.503923e+00	0.332328	-0.755491	-0.836225
hp	1.413176e+00	0.133349	-0.754421	-0.554620
wt	1.404098e+00	0.339986	-0.724316	-0.783574
acc	-1.165944e+00	0.348468	0.396094	0.324108
age	6.883238e-01	0.030415	-1.062945	0.700285
origin_1	1.000000e+00	0.931818	0.419355	0.228261
origin_2	-2.498002e-16	0.034091	0.177419	0.489130
origin_3	8.326673e-17	0.034091	0.403226	0.282609

In [79]:

```
# Group 1 has highest values for mpg while 3rd has lowest
# Group 0 has max no of cylinders and 2 forms of lower cylinder values
# As seen in correlation and pairplot, Group 0 has highest values for hp,wt and displ
# Group 1 seems to be comprising of newest cars
# Group 3 and 0 seems to be originated at point 3, while 2 in 2nd point and 1 again at poin
```

10. Assigning the groups created above to data frame and study the characteristics for each group

In [80]:

```
# Add cluster number to original cars data

predictions = kmeans.predict(cars_new)
predictions
data["group"] = predictions
data["group"] = data['group'].astype('category')
data.dtypes
```

Out[80]:

```
mpg          float64
cyl            int64
disp         float64
hp           float64
wt             int64
acc          float64
yr             int64
origin         int64
car_name      object
age            int64
origin_1       uint8
origin_2       uint8
origin_3       uint8
group       category
dtype: object
```

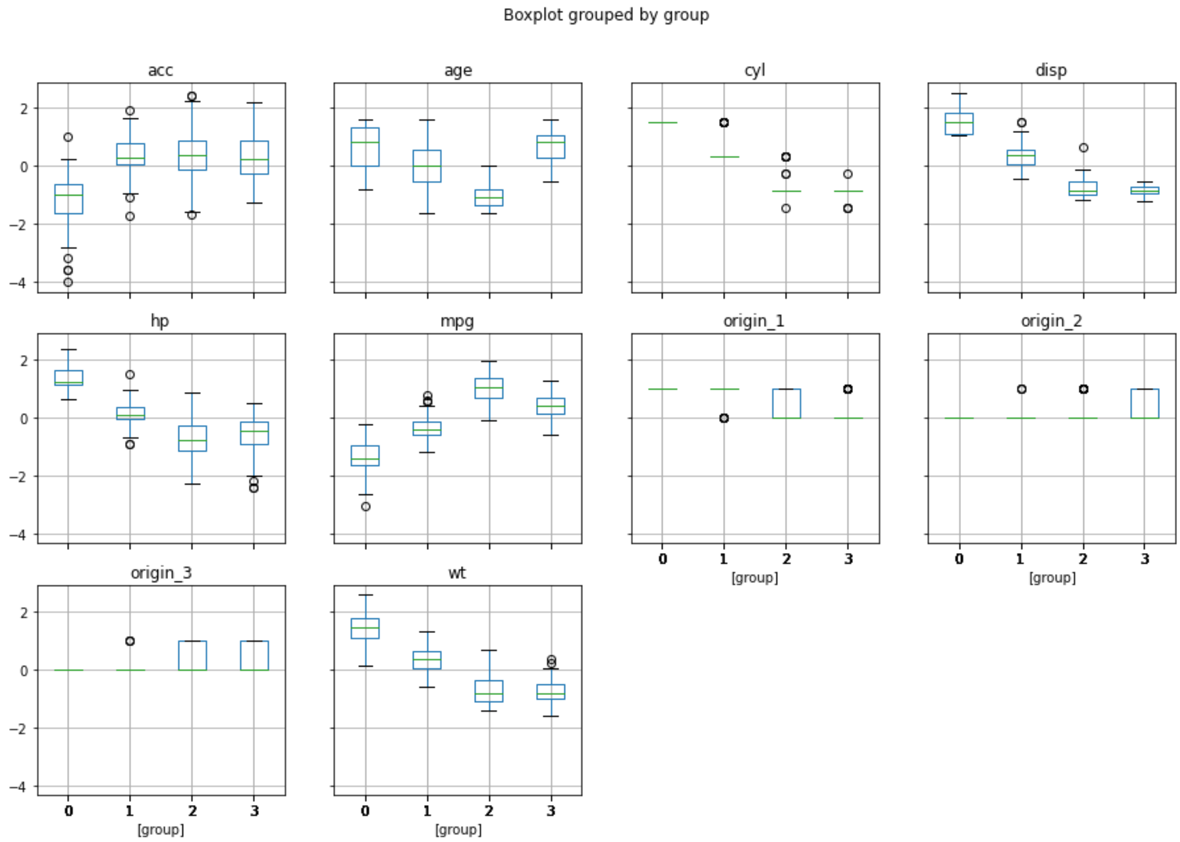
In [82]:

```
# Visualize the centres


cars_new["group"] = predictions
cars_new.boxplot(by = 'group',  layout=(3,4), figsize=(15, 10))
```

Out[82]:

```
array([[<AxesSubplot:title={'center':'acc'}, xlabel='[group]'>,
        <AxesSubplot:title={'center':'age'}, xlabel='[group]'>,
        <AxesSubplot:title={'center':'cyl'}, xlabel='[group]'>,
        <AxesSubplot:title={'center':'disp'}, xlabel='[group]'>],
       [<AxesSubplot:title={'center':'hp'}, xlabel='[group]'>,
        <AxesSubplot:title={'center':'mpg'}, xlabel='[group]'>,
        <AxesSubplot:title={'center':'origin_1'}, xlabel='[group]'>,
        <AxesSubplot:title={'center':'origin_2'}, xlabel='[group]'>],
       [<AxesSubplot:title={'center':'origin_3'}, xlabel='[group]'>,
        <AxesSubplot:title={'center':'wt'}, xlabel='[group]'>,
        <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



In [83]:

```
# Group 0 is characterised by lower acc, comparitely old models, higher wt, hp but lowest m
# Group 1 -Highest mpg, lower wt and hp. Lower age limits suggest comparitevly newer cars.
# Group 2 - Origin mostly in location 2, lower deviation in wts, and hp so medain mpg and a
# Group 3 - Again slighlty higher in wt origin code as 1. Better performance in terms of mp
```

In [85]:

```
# Export the data into csv for any further analysis

from pandas import ExcelWriter
writer = ExcelWriter('d:\groups.xls')
data.to_excel(writer,'Sheet1')
writer.save()
```

In [ ]:

```
# We can try similar analysis for 3 grps as well to check if we get more clear distinction
```