In [1]:
```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the
 input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/car-mpg/mpg_raw.csv
```

This is a 3 part series in which I will walk through a dataset analysing it and then at the end do predictive modelling. I recommend to follow the parts in sequential order but you can jump to any part by clicking on the heading links in blue.

Part 1, Exploratory Data Analysis(EDA):
This part consists of summary statistics of data but the major focus will be on EDA where we extract meaning/information from data using plots and report important insights about data. This part is more about **data analysis** and **business intelligence(BI)**.

Part 2, Statistical Analysis:
In this part we will do many statistical hypothesis testing, apply estimation statistics and interpret the results we get. We will also validate this with the findings from part one. We will apply both parametric and non-parametric tests. We will report all the important insights we get in this part. This part is all about **data science** requires statistical background.

Part 3, Predictive Modelling:
In this part we will predict some response using predictors. This part is all about **machine learning**.

If you like these notebooks then please upvote and also share with others.

Data Description

The data we are using for EDA is the auto mpg dataset taken from UCI repository.

Information regarding data
Title: Auto-Mpg Data
Number of Instances: 398
Number of Attributes: 9 including the class attribute
Attribute Information:

```
1. mpg:           continuous
2. cylinders:     multi-valued discrete
3. displacement:  continuous
4. horsepower:    continuous
5. weight:        continuous
6. acceleration:  continuous
7. model year:    multi-valued discrete
8. origin:        multi-valued discrete
9. car name:      string (unique for each instance)

All the attributes are self-explanatory.
```

This data is not complex and is good for analysis as it has a nice blend of both categorical and numerical attributes.

data source

This is **part 1** ie., EDA. We won't stretch this part long and do following things in sequential manner.

1. **Preprocess the data**, this includes dealing with missing values, duplicate data if any and then align the data.
2. **EDA on categorical attributes**, this includes analysing their distributions and relations with other cat. attributes.
3. **EDA on numerical attributes**, this includes analysing their distributions and relations with other num. attributes.
4. Then we will analyse the **relation b/w num. & cat. attributes**.

I make use of **seaborn** heavily throughout the notebook, so it is also a good goto notebook for those who are looking for EDA using seaborn.

In [2]:
```python
# first import all necessary libraries
import itertools
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```python
# set seaborn's default settings
sns.set()
```

```
We will first import the data into a pandas dataframe and inspect it's properties.
```

In [4]:
```python
df = pd.read_csv("../input/car-mpg/mpg_raw.csv")
df.head()
```

Out[4]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino

In [5]:
```python
# so now the data is in rectangular form with 398 entries each having 9 distinct properties
df.shape
```

Out[5]:
```
(398, 9)
```

In [6]:
```python
# let's list all the columns
columns = list(df.columns)
columns
```

Out[6]:
```
['mpg',
 'cylinders',
 'displacement',
 'horsepower',
 'weight',
 'acceleration',
 'model_year',
 'origin',
 'name']
```

In [7]:
```python
# we now describe the properties of this dataframe like column datatype etc.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
mpg             398 non-null float64
cylinders       398 non-null int64
displacement    398 non-null float64
horsepower      392 non-null float64
weight          398 non-null int64
acceleration    398 non-null float64
model_year      398 non-null int64
origin          398 non-null object
name            398 non-null object
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

We now make two distinct list for categorical and numerical column names as the analysis differ for both the types. For that we introspect the datatypes of each column and if it is of type `object` then it's categorical and else numerical.

We will use these two lists heavily throughout the analysis.

In [8]:
```python
cats = list(df.select_dtypes(include=['object']).columns)
nums = list(df.select_dtypes(exclude=['object']).columns)
print(f'categorical variables:  {cats}')
print(f'numerical variables:  {nums}')
```

```
categorical variables:  ['origin', 'name']
numerical variables:  ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'm
odel_year']
```

In [9]:
```python
# let's inspect how many unique values are there in each column.
df.nunique(axis=0)
```

Out[9]:
```
mpg             129
cylinders         5
displacement     82
horsepower       93
weight          351
acceleration     95
model_year       13
origin            3
name            305
dtype: int64
```

```
As there are very few unique values for cylinders and model_year, so it's safe to make them ca
tegorical instead of numeric.
```

In [10]:
```python
# cylinders and model_year also seems to be categorical so lets update the lists
cats.extend(['cylinders', 'model_year'])
nums.remove('cylinders')
nums.remove('model_year')

print(f'categorical variables:  {cats}')
print(f'numerical variables:  {nums}')
```

```
categorical variables:  ['origin', 'name', 'cylinders', 'model_year']
numerical variables:  ['mpg', 'displacement', 'horsepower', 'weight', 'acceleration']
```

```
Now inspect for nans in data.
```

In [11]:
```python
# check for `nans` in each column
df.isna().sum()
```

Out[11]:
```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
name            0
dtype: int64
```

In [12]:
```python
# let's print these 6 `nan` containing rows
df[df.isnull().any(axis=1)]
```

Out[12]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
32	25.0	4	98.0	NaN	2046	19.0	71	usa	ford pinto
126	21.0	6	200.0	NaN	2875	17.0	74	usa	ford maverick
330	40.9	4	85.0	NaN	1835	17.3	80	europe	renault lecar deluxe
336	23.6	4	140.0	NaN	2905	14.3	80	usa	ford mustang cobra
354	34.5	4	100.0	NaN	2320	15.8	81	europe	renault 18i
374	23.0	4	151.0	NaN	3035	20.5	82	usa	amc concord dl

In [13]:
```python
# nan rows proportion in data
6 / len(df)
```

Out[13]:
```
0.01507537688442211
```

```
So horsepower consists of total of 6 nan rows comprising of around 1.5% of data. As this fract
ion is very low so it's safe to drop these nan rows for now.
```

Note: If the nan-row proportion is large enough then we won't drop it but instead impute missing values.

In [14]:
```python
# for now remove all nan rows as they are just 1.5%
df = df[~df.isnull().any(axis=1)]
df.reset_index(inplace=True)
df.drop('index', inplace=True, axis=1)
df.shape
```

Out[14]:
```
(392, 9)
```

In [15]:
```python
# find total duplicate entries and drop them if any
print(f'total duplicate rows: {df.duplicated().sum()}')

# drop duplicate rows if any
df = df[~df.duplicated()]
df.shape
```

```
total duplicate rows: 0
```

Out[15]:
```
(392, 9)
```

In [16]:
```python
# before we move ahead it's a good practice to group all variables together having same type.
df = pd.concat((df[cats], df[nums]), axis=1)
df.head()
```

Out[16]:

	origin	name	cylinders	model_year	mpg	displacement	horsepower	weight	acceleration
0	usa	chevrolet chevelle malibu	8	70	18.0	307.0	130.0	3504	12.0
1	usa	buick skylark 320	8	70	15.0	350.0	165.0	3693	11.5
2	usa	plymouth satellite	8	70	18.0	318.0	150.0	3436	11.0
3	usa	amc rebel sst	8	70	16.0	304.0	150.0	3433	12.0
4	usa	ford torino	8	70	17.0	302.0	140.0	3449	10.5

In [17]:
```python
num_rows, num_cols = df.shape
```

In [18]:
```python
# save this cleaned df to csv
df.to_csv('mpg_cleaned.csv', index=False)
```

```
Now we are all good to go for some in-depth analysis
```

Analysis on Categorical Attributes

Our analysis includes both descriptive stats and EDA.

In [19]:
```python
# let's import the cleaned version of mpg although no need here because we already updated df
df = pd.read_csv("mpg_cleaned.csv")
```

In [20]:
```python
print(f'categorical variables:  {cats}')
```

```
categorical variables:  ['origin', 'name', 'cylinders', 'model_year']
```

```
We will first slice out the categorical columns from original dataframe and then do analysis o
n it keeping the original data untou-ched, and at the end incorporate needed changes in our or
iginal dataframe.
```

In [21]:
```python
df_cat = df.loc[:, 'origin':'model_year']
df_cat.head()
```

Out[21]:

	origin	name	cylinders	model_year
0	usa	chevrolet chevelle malibu	8	70
1	usa	buick skylark 320	8	70
2	usa	plymouth satellite	8	70
3	usa	amc rebel sst	8	70
4	usa	ford torino	8	70

As `origin` and `name` consists of text data so it needs some preprocessing. We will remove all extra spaces from each string, otherwise the same string with different spacings will be treated as different categories which should not be the case.

In [22]:
```python
# remove extra spaces if any
for col in ['origin', 'name']:
    df_cat[col] = df_cat[col].apply(lambda x: ' '.join(x.split()))
```

We will create an artificial categorical attribute named `mpg_level` which categorizes `mpg` into `low`, `medium` and `high`. This is done for two reasons, first it will help a lot in EDA i.e., we can bifurcate plots on the basis of mpg and secondly this is easy to understand as compared to numbers in `mpg`.

We are dividing `mpg` into three regions as,
[min, 17) -> low
[17, 29) -> medium
[29, max) -> high

Also the choice of the range is analytical and can be anything till it seems to be reasonable.

Note: This is feature-engineering and mostly done in predictive modelling but it makes sense to introduce it here.

In [23]:
```python
df_cat['mpg_level'] = df['mpg'].apply(lambda x: 'low' if x<17 else 'high' if x>29 else 'medium')
cats.append('mpg_level')
print(f'categorical variables:  {cats}')
```

```
categorical variables:  ['origin', 'name', 'cylinders', 'model_year', 'mpg_level']
```

In [24]:
```python
# let's look at the unique categories in `origin`, `cylinders` & `model_year`
# we are leaving `name` because it is almost unique for each entry (nothing interesting)
print(f"categories in origin: {pd.unique(df_cat['origin'])}")
print(f"categories in cylinders: {pd.unique(df_cat['cylinders'])}")
print(f"categories in model_year: {pd.unique(df_cat['model_year'])}")
```

```
categories in origin: ['usa' 'japan' 'europe']
categories in cylinders: [8 4 6 3 5]
categories in model_year: [70 71 72 73 74 75 76 77 78 79 80 81 82]
```

In [25]:
```python
# Although descriptive stats for categorical variables are not much informatic but still it's worth
 looking once.
# Also pandas describe function is only for numeric data and in df_cat `cylinders` & `model_year` ar
e the only numeric type.
df_cat.describe()
```

Out[25]:

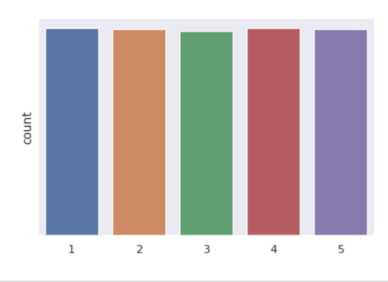
	cylinders	model_year
count	392.000000	392.000000
mean	5.471939	75.979592
std	1.705783	3.683737
min	3.000000	70.000000
25%	4.000000	73.000000
50%	4.000000	76.000000
75%	8.000000	79.000000
max	8.000000	82.000000

```
It seems that most of the values in `cylinders` are 4 with a min of 3 and max of 8.
```

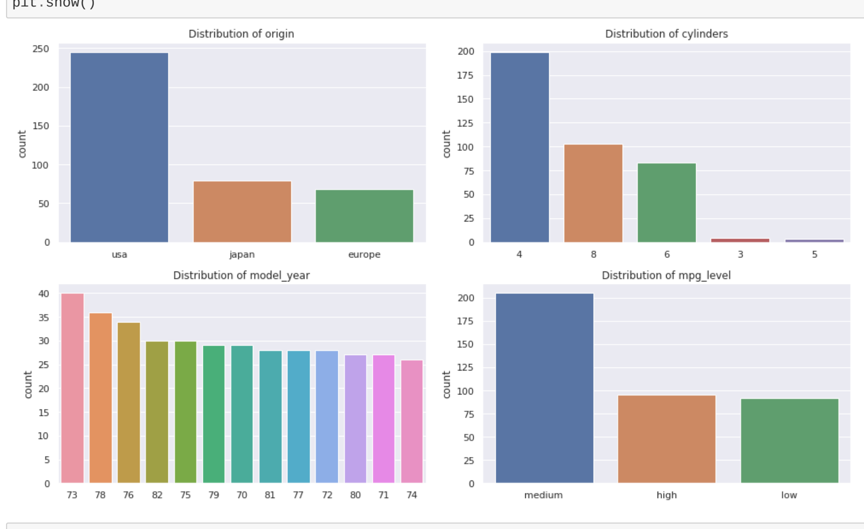
Analysis of Distribution

Now we analyse the distribution for each categorical feature and make some insights from the plots.

In case of categorical variables an ideal (or atleast loved) distribution is `uniform`,



In [26]:
```python
fig = plt.figure(1, (14, 8))

for i,cat in enumerate(df_cat.drop(['name'], axis=1).columns):
    ax = plt.subplot(2,2,i+1)
    sns.countplot(df_cat[cat], order=df_cat[cat].value_counts().index)
    ax.set_xlabel(None)
    ax.set_title(f'Distribution of {cat}')
    plt.tight_layout()

plt.show()
```



In [27]:
```python
# calculate proportion of dominant classes in each category
for i,cat in enumerate(df_cat.drop(['name'], axis=1).columns):
    val_counts = df_cat[cat].value_counts()
    dominant_frac = val_counts.iloc[0] / num_rows
    print(f'`{val_counts.index[0]}` alone contributes to {round(dominant_frac * 100, 2)}% of {cat}')
```

```
`usa` alone contributes to 62.5% of origin
`4` alone contributes to 50.77% of cylinders
`73` alone contributes to 10.2% of model_year
`medium` alone contributes to 52.3% of mpg_level
```

In [28]:
```python
# count of different cylinders
df_cat.cylinders.value_counts()
```

Out[28]:
```
4    199
8    103
6     83
3      4
5      3
Name: cylinders, dtype: int64
```

Insights

- `origin` is highly imbalanced, `usa` alone consists of 62.5% of data whereas `japan` & `europe` are having similar proportion. We will see this dominance in future analysis. `We will try to find the reason for this in our further analysis.`
- `cylinders` is highly imbalanced, `4` alone consists of 50.77% of data. Whereas `8` & `6` are nearly in same proportion but `3` & `5` collectively accounts for only 7 entries i.e., 1.8% of entire data. We will see this huge proportional imbalance in `cylinders` in future analysis.
- `mpg_level` is highly imbalanced, `medium` alone consists of 52.3% of data while `low` & `high` are in the same proportion. This dominance is due the fact of our thresholding while manufacturing this feature because the medium range is broader hence it consists of more data points. It won't be there in original `mpg` feature as it is continous.
- `model_year` is considerably balanced which is good.

```
Now we analyse car `name`.
```

In [29]:
```python
print(f'total unique categories in `name`: {df_cat.name.nunique()}')
print(f"\nunique categories in `name`:\n\n {df_cat.name.unique()}")
```

```
total unique categories in `name`: 301

unique categories in `name`:

 ['chevrolet chevelle malibu' 'buick skylark 320' 'plymouth satellite'
 'amc rebel sst' 'ford torino' 'ford galaxie 500' 'chevrolet impala'
 'plymouth fury iii' 'pontiac catalina' 'amc ambassador dpl'
 'dodge challenger se' "plymouth 'cuda 340" 'chevrolet monte carlo'
 'buick estate wagon (sw)' 'toyota corona mark ii' 'plymouth duster'
 'amc hornet' 'ford maverick' 'datsun pl510'
 'volkswagen 1131 deluxe sedan' 'peugeot 504' 'audi 100 ls' 'saab 99e'
 'bmw 2002' 'amc gremlin' 'ford f250' 'chevy c20' 'dodge d200' 'hi 1200d'
 'chevrolet vega 2300' 'toyota corona' 'plymouth satellite custom'
 'ford torino 500' 'amc matador' 'pontiac catalina brougham'
 'dodge monaco (sw)' 'ford country squire (sw)' 'pontiac safari (sw)'
 'amc hornet sportabout (sw)' 'chevrolet vega (sw)' 'pontiac firebird'
 'ford mustang' 'mercury capri 2000' 'opel 1900' 'peugeot 304' 'fiat 124b'
 'toyota corolla 1200' 'datsun 1200' 'volkswagen model 111'
 'plymouth cricket' 'toyota corona hardtop' 'dodge colt hardtop'
 'volkswagen type 3' 'chevrolet vega' 'ford pinto runabout'
 'amc ambassador sst' 'mercury marquis' 'buick lesabre custom'
 'oldsmobile delta 88 royale' 'chrysler newport royal' 'mazda rx2 coupe'
 'amc matador (sw)' 'chevrolet chevelle concours (sw)'
 'ford gran torino (sw)' 'plymouth satellite custom (sw)'
 'volvo 145e (sw)' 'volkswagen 411 (sw)' 'peugeot 504 (sw)'
 'renault 12 (sw)' 'ford pinto (sw)' 'datsun 510 (sw)'
 'toyouta corona mark ii (sw)' 'dodge colt (sw)'
 'toyota corolla 1600 (sw)' 'buick century 350' 'chevrolet malibu'
 'ford gran torino' 'dodge coronet custom' 'mercury marquis brougham'
 'chevrolet caprice classic' 'ford ltd' 'plymouth fury gran sedan'
 'chrysler new yorker brougham' 'buick electra 225 custom'
 'amc ambassador brougham' 'plymouth valiant' 'chevrolet nova custom'
 'volkswagen super beetle' 'ford country' 'plymouth custom suburb'
 'oldsmobile vista cruiser' 'toyota carina' 'datsun 610' 'maxda rx3'
 'ford pinto' 'mercury capri v6' 'fiat 124 sport coupe'
 'chevrolet monte carlo s' 'pontiac grand prix' 'fiat 128' 'opel manta'
 'audi 100ls' 'volvo 144ea' 'dodge dart custom' 'saab 99le'
 'toyota mark ii' 'oldsmobile omega' 'chevrolet nova' 'datsun b210'
 'chevrolet chevelle malibu classic' 'plymouth satellite sebring'
 'buick century luxus (sw)' 'dodge coronet custom (sw)' 'audi fox'
 'volkswagen dasher' 'datsun 710' 'dodge colt' 'fiat 124 tc' 'honda civic'
 'subaru' 'fiat x1.9' 'plymouth valiant custom' 'mercury monarch'
 'chevrolet bel air' 'plymouth grand fury' 'buick century'
 'chevroelt chevelle malibu' 'plymouth fury' 'buick skyhawk'
 'chevrolet monza 2+2' 'ford mustang ii' 'toyota corolla' 'pontiac astro'
 'volkswagen rabbit' 'amc pacer' 'volvo 244dl' 'honda civic cvcc'
 'fiat 131' 'capri ii' 'renault 12tl' 'dodge coronet brougham'
 'chevrolet chevette' 'chevrolet woody' 'vw rabbit' 'dodge aspen se'
 'ford granada ghia' 'pontiac ventura sj' 'amc pacer d/l' 'datsun b-210'
 'volvo 245' 'plymouth volare premier v8' 'mercedes-benz 280s'
 'cadillac seville' 'chevy c10' 'ford f108' 'dodge d100'
 'honda accord cvcc' 'buick opel isuzu deluxe' 'renault 5 gtl'
 'plymouth arrow gs' 'datsun f-10 hatchback' 'oldsmobile cutlass supreme'
 'dodge monaco brougham' 'mercury cougar brougham' 'chevrolet concours'
 'buick skylark' 'plymouth volare custom' 'ford granada'
 'pontiac grand prix lj' 'chevrolet monte carlo landau' 'chrysler cordoba'
 'ford thunderbird' 'volkswagen rabbit custom' 'pontiac sunbird coupe'
 'toyota corolla liftback' 'ford mustang ii 2+2' 'dodge colt m/m'
 'subaru dl' 'datsun 810' 'bmw 320i' 'mazda rx-4'
 'volkswagen rabbit custom diesel' 'ford fiesta' 'mazda glc deluxe'
 'datsun b210 gx' 'oldsmobile cutlass salon brougham' 'dodge diplomat'
 'mercury monarch ghia' 'pontiac phoenix lj' 'ford fairmont (auto)'
 'ford fairmont (man)' 'plymouth volare' 'amc concord'
 'buick century special' 'mercury zephyr' 'dodge aspen' 'amc concord d/l'
 'buick regal sport coupe (turbo)' 'ford futura' 'dodge magnum xe'
 'datsun 510' 'dodge omni' 'toyota celica gt liftback' 'plymouth sapporo'
 'oldsmobile starfire sx' 'datsun 200-sx' 'audi 5000' 'volvo 264gl'
 'saab 99gle' 'peugeot 604sl' 'volkswagen scirocco' 'honda accord lx'
 'pontiac lemans v6' 'mercury zephyr 6' 'ford fairmont 4'
 'amc concord dl 6' 'dodge aspen 6' 'ford ltd landau'
 'mercury grand marquis' 'dodge st. regis' 'chevrolet malibu classic (sw)'
 'chrysler lebaron town @ country (sw)' 'vw rabbit custom'
 'maxda glc deluxe' 'dodge colt hatchback custom' 'amc spirit dl'
 'mercedes benz 300d' 'cadillac eldorado' 'plymouth horizon'
 'plymouth horizon tc3' 'datsun 210' 'fiat strada custom'
 'buick skylark limited' 'chevrolet citation' 'oldsmobile omega brougham'
 'pontiac phoenix' 'toyota corolla tercel' 'datsun 310' 'ford fairmont'
 'audi 4000' 'toyota corona liftback' 'mazda 626' 'datsun 510 hatchback'
 'mazda glc' 'vw rabbit c (diesel)' 'vw dasher (diesel)'
 'audi 5000s (diesel)' 'mercedes-benz 240d' 'honda civic 1500 gl'
 'vokswagen rabbit' 'datsun 280-zx' 'mazda rx-7 gs' 'triumph tr7 coupe'
 'honda accord' 'plymouth reliant' 'dodge aries wagon (sw)'
 'toyota starlet' 'plymouth champ' 'honda civic 1300' 'datsun 210 mpg'
 'toyota tercel' 'mazda glc 4' 'plymouth horizon 4' 'ford escort 4w'
 'ford escort 2h' 'volkswagen jetta' 'honda prelude' 'datsun 200sx'
 'peugeot 505s turbo diesel' 'volvo diesel' 'toyota cressida'
 'datsun 810 maxima' 'oldsmobile cutlass ls' 'ford granada gl'
 'chrysler lebaron salon' 'chevrolet cavalier' 'chevrolet cavalier wagon'
 'chevrolet cavalier 2-door' 'pontiac j2000 se hatchback' 'dodge aries se'
 'ford fairmont futura' 'volkswagen rabbit l' 'mazda glc custom l'
 'mazda glc custom' 'plymouth horizon miser' 'mercury lynx l'
 'nissan stanza xe' 'honda civic (auto)' 'datsun 310 gx'
 'buick century limited' 'oldsmobile cutlass ciera (diesel)'
 'chrysler lebaron medallion' 'ford granada l' 'toyota celica gt'
 'dodge charger 2.2' 'chevrolet camaro' 'ford mustang gl' 'vw pickup'
 'dodge rampage' 'ford ranger' 'chevy s-10']
```

First thing even though name is categorical but it has a lot categories and this even makes sense because product names generally varies a lot in any domain. So it's not fruitfull to do analysis on car names as these are names just like product id and seems to hold no important insights.

But one thing to be noticed here is that each car name starts with a company name, so maybe the case that there are very few companies in the dataset and it will be fruitful to extract them as seperate feature and do analysis on that. So let's do it.

We will create a new attribute named as car_company by extracting the first word from all names. We also remove the car company from each car name because it is not needed now, and also we rename column `name` to `car_name`.

In [30]:
```python
# extract car company from `name`
df_cat['car_company'] = df_cat['name'].apply(lambda x: x.split()[0])

# remove car company from `name` and rename to `car_name`
df_cat['car_name'] = df_cat['name'].apply(lambda x: ' '.join(x.split()[1:]))
df_cat.drop('name', axis=1, inplace=True)

cats.extend(['car_company', 'car_name'])
cats.remove('name')

print(f'categorical variables:  {cats}')
df_cat.head()
```

```
categorical variables:  ['origin', 'cylinders', 'model_year', 'mpg_level', 'car_company', 'car_name']
```

Out[30]:

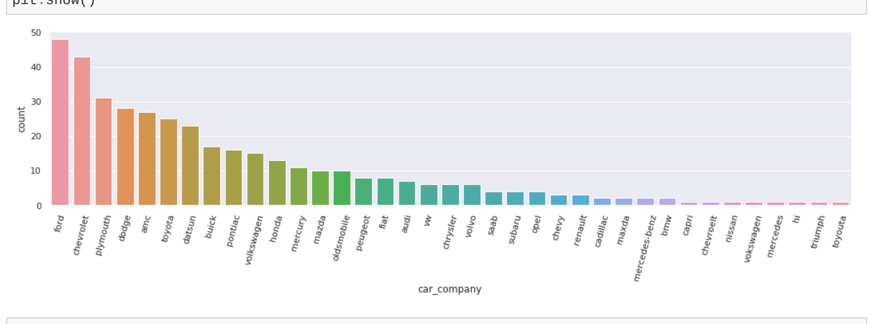
	origin	cylinders	model_year	mpg_level	car_company	car_name
0	usa	8	70	medium	chevrolet	chevelle malibu
1	usa	8	70	low	buick	skylark 320
2	usa	8	70	medium	plymouth	satellite
3	usa	8	70	low	amc	rebel sst
4	usa	8	70	medium	ford	torino

In [31]:
```python
# now check for total unique values in `car_company`
print(f'total unique categories in `car_company`: {df_cat.car_company.nunique()}')
print(f"\nunique categories in `car_company`:\n\n {df_cat.car_company.unique()}")
```

```
total unique categories in `car_company`: 37

unique categories in `car_company`:

 ['chevrolet' 'buick' 'plymouth' 'amc' 'ford' 'pontiac' 'dodge' 'toyota'
 'datsun' 'volkswagen' 'peugeot' 'audi' 'saab' 'bmw' 'chevy' 'hi'
 'mercury' 'opel' 'fiat' 'oldsmobile' 'chrysler' 'mazda' 'volvo' 'renault'
 'toyouta' 'maxda' 'honda' 'subaru' 'chevroelt' 'capri' 'vw'
 'mercedes-benz' 'cadillac' 'mercedes' 'vokswagen' 'triumph' 'nissan']
```

```
Great this is what we wished and indeed we get. Our idea that there will be few car companies
 involved in this data is indeed corr-ect. Because the number of categories are less we can no
w do analysis on that. So we took a step in right direction just by our in-tuition.

Now we analyse the distribution of `car_company`.
```

In [32]:
```python
fig = plt.figure(1, (18, 4))

ax1 = plt.subplot(1,1,1)
sns.countplot(df_cat['car_company'], order=df_cat['car_company'].value_counts().index)
ax1.set_xticklabels(ax1.get_xticklabels(), rotation=75)

plt.show()
```



In [33]:
```python
df_cat.car_company.value_counts()[:2]
```

Out[33]:
```
ford         48
chevrolet    43
Name: car_company, dtype: int64
```

Insights

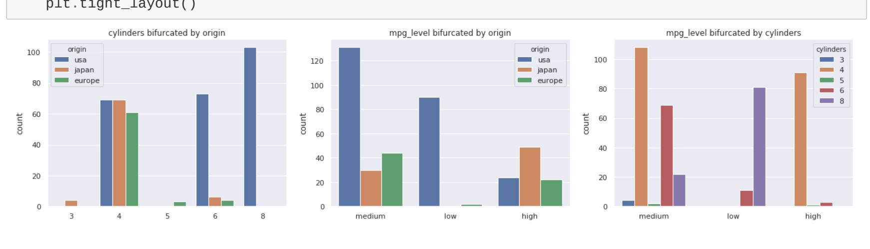
- We found that `car_name` has a lot categories, close to total data points. So it's not fruitfull to do analysis on that as it is unique for most of the points and also in most cases names are safe to be avoided as they doesn't have correlations with other.
- We then create an artificial attribute named `car_company` by extracting company names from car names. We find that there are much few car companies as compared to car names (around 8 times less).
- We then found that the distribution of `car_company` is not uniform and most of the proportion is covered by top 15 car companies. Whereas `ford` and `chevrolet` alone comprises of around 23% (almost a quater).

Conclusion

- Every categorical attribute except `model_year` is highly imbalanced and far from `uniform distribution`. In all cases most of the data is comprised of top few categories.
- Although `model_year` is not perfectly uniform but we can think it as `uniform-like distribution`. This is a digestible assumption for two reasons, first we can clearly see in plot that indeed the distribution is `uniform-like` and also this is not the entire population but a sample of it so may be in large run it will converge to `uniform` which may be the true population distribution (**Law of Large Number**).

```
Now we will analyze how different features behaves on changing other features.
```

In [34]:
```python
combos = itertools.combinations(['origin', 'cylinders', 'mpg_level'], 2)

fig = plt.figure(1, (18, 8))

i = 0
for pair in combos:
#     i+=1
#     ax = plt.subplot(2,3,i)
#     sns.countplot(x=pair[0], hue=pair[1], data=df_cat)
#     ax.set_xlabel(None)
#     ax.set_title(f'{pair[0]} bifurcated by {pair[1]}')
#     plt.tight_layout()

    i+=1
    ax = plt.subplot(2,3,i)
    sns.countplot(x=pair[1], hue=pair[0], data=df_cat)
    ax.set_xlabel(None)
    ax.set_title(f'{pair[1]} bifurcated by {pair[0]}')
    plt.tight_layout()
```



Insights

```
We can clearly see the inpact of imbalanced categories in our bifurcated plots.
```

cylinders bigurcated by origin

- Japan is the only origin with vehicles having 3 cylinders.
- Europe is the only origin with vehicles having 5 cylinders.
- USA is the only origin with vehicles having 8 cylinders.
- All origins has 4 cylinder vehicles and in almost equal proportion, also because 4 is dominating in cylinders.
- All origins has 6 cylinder vehicles but dominated by USA due the fact that it is dominating in origin.

mpg_level bigurcated by origin

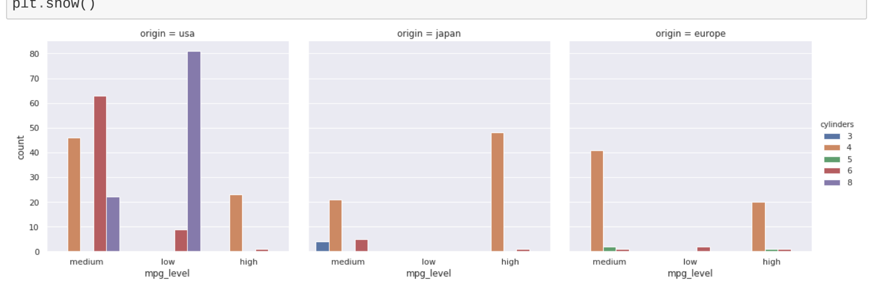
- Japan doesn't have any vehicle with low mpg_level whereas europe has negligible vehicles with low mpg_level and almost all vehicles that has low mpg_level are from usa.
- Japan has the most vehicles with high mpg_level.
- USA has the most vehicles with medium mpg_level (again due to the fact that most vehicles belongs to USA).

mpg_level bigurcated by cylinders

- Vehicles with low mpg_level has either 6 or 8 cylinders and most of them has 8 cylinders.
- Almost all vehicles with high mpg_level has 4 cylinders and with very few (less than 5) has 5-6 cylinders.
- Most vehicles with medium mpg_level has 4,6 & 8 cylinders. This is due to the fact that most of the vehicles has these number of cylinders as we inspected in cylinder distribution earlier.

```
Let's analyse mpg_level with cylinders bifurcated by origin.
```

In [35]:
```python
sns.catplot(x='mpg_level', hue='cylinders', col='origin', data=df_cat, kind='count')
plt.show()
```



Insights

- Japan doesn't have any cylinder with low mpg_level and most of it's vehicles have high mpg_level with mostly 4 cylinders(this we already inspected from earlier plots as well, nothing new).
- Almost every vehicle in europe has 4 cylinders and most of them are medium or high.
- USA has few vehicles with high mpg level as compared to other knowing the fact that most of the vehicles belongs to USA.

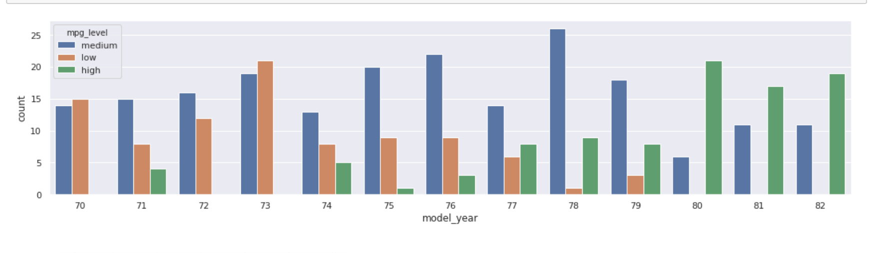
Note: Although these insights can be detected from earlier plots but it was bit hard and bifurcating helps us and we extracted even more meaning from data.

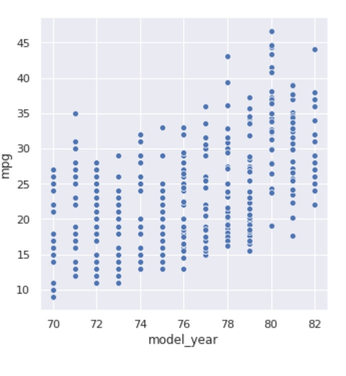
Conclusion

- Japan is leading in case of mpg level with most of it's vehicles having high mpg_level. It has more than twice vehicles with high mpg compared to other origins.
- **It seems like as number of cylinders increases mpg decreases generally.**

```
Let's analyse model_year
```

```
In [36]: fig = plt.figure(1, (18,4))
         sns.countplot(x='model_year', hue='mpg_level', data=df_cat)
         sns.relplot(x='model_year', y='mpg', data=df)
         plt.show()
```

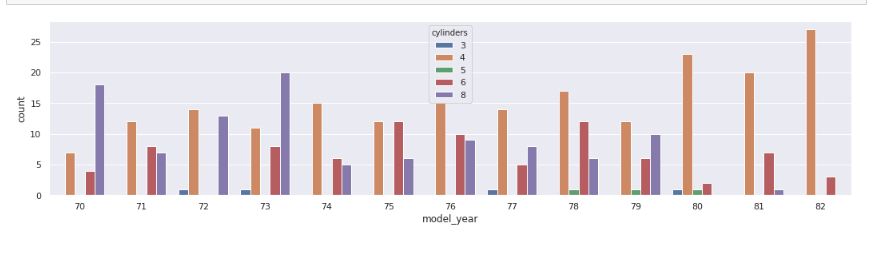




Insights

- As year progresses manufacturing of low mpg car decreases and eventually stopped after 79.
- As year progresses manufacturing of high mpg car increases and infact after 79 their manufacturing was 1.5-2 times as compared to vehicles with medium mpg.
- Throughout the years there is no significant change in manufacturing of vehicles with medium mpg and infact from 74-79 their manufacturing was more than the sum of remaining two.
- From the scatter plot we can clearly see an upward linear trend i.e., as year progress mpg increases.

```
In [37]: fig = plt.figure(1, (18,4))
         sns.countplot(x='model_year', hue='cylinders', data=df_cat)
         plt.show()
```

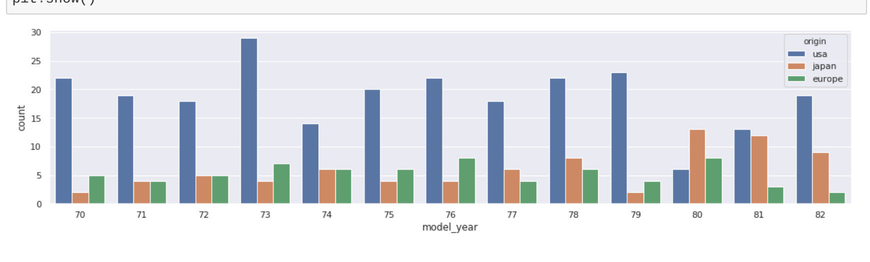


Insights

- As the year progresses vehicles with more cylinders (8 & 6) decreases significantly.
- As the year progresses vehicles with less cylinders increases.
- One important thing to be noticed that throughout the years vehicles with 4 cylinders have significant proportion and infact in the 80's most of the vehicles has 4 cylinders.

```
These results make sense because as year progresses technology advances vehicles with low mpg
 and more cylinders looses focus and vehicles with high mpg and less cylinders are the new sta
rs.
```

```
In [38]: fig = plt.figure(1, (18,4))
         sns.countplot(x='model_year', hue='origin', data=df_cat)
         plt.show()
```



Insights

- In the starting year manufacturing in USA is dominated compeletely.
- As the year progresses japan and europe started manufacturing more vehicles. Infact in the year 80 both japan and europe manufactured more than USA. This may be due to something because throught the years USA dominates and suddenly there is a considerable decrease in their manufacturing. This may have something to do in USA in year 80.
- Initially europe manufactures more vehicle then japan but then japan exceeds it after 76.

```
Let's analyse car_company
```

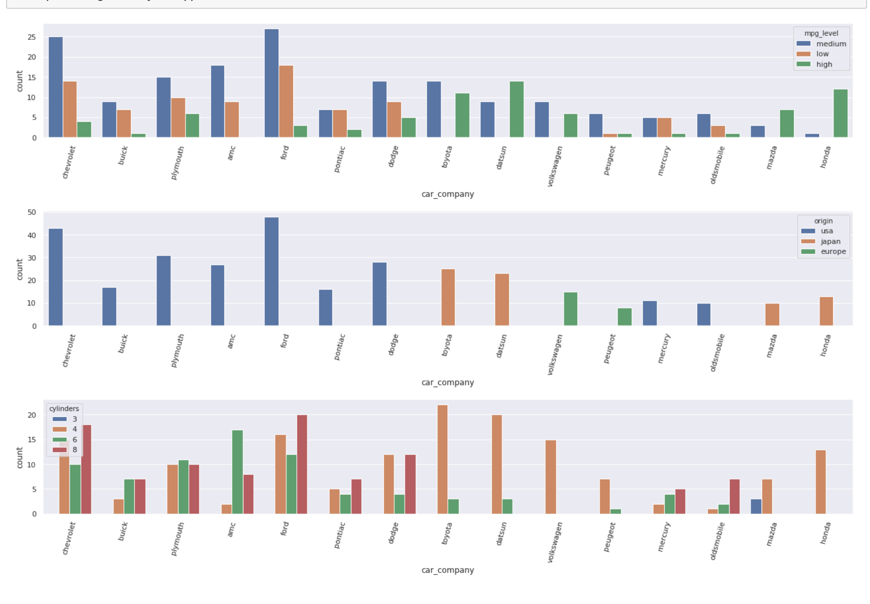
As `car_company` contains a lot categories and most of them has very low proportion so we will analyse only the top 15 car companies.

```
In [39]: top_car_companies = df_cat.car_company.value_counts()[:15].index
         top_car_companies

Out[39]: Index(['ford', 'chevrolet', 'plymouth', 'dodge', 'amc', 'toyota', 'datsun',
               'buick', 'pontiac', 'volkswagen', 'honda', 'mercury', 'mazda',
               'oldsmobile', 'peugeot'],
              dtype='object')
```

```
In [40]: df_cat_top_comp = df_cat[df_cat.car_company.isin(top_car_companies)]
         df_cat_top_comp.shape

Out[40]: (325, 6)
```

```
We can see that top 15 car companies alone manufactures 83% of vehicles.
```

```
In [41]: fig = plt.figure(1, (18,12))

         for i,cat in enumerate(['mpg_level', 'origin', 'cylinders']):
             ax = plt.subplot(3,1,i+1)
             sns.countplot(x='car_company', hue=cat, data=df_cat_top_comp)
             ax.set_xticklabels(ax.get_xticklabels(), rotation=75)
             plt.tight_layout()
```



Insights

- Top manufacturing companies have all mpg level vehicles but companies with less vehicles focuses more on high or medium mpg vehicle.
- All top manufacturing companies are from `usa` and that is the reason why usa has most of the vehicles in dataset (this is one of our key findings). We now answered this question asked earlier by us.
- All top manufacturing companies focuses on vehicles with cylinders 4, 6 & 8 equally, but the companies with less manufacturing generally uses less cylinders in their vehicles.

```
We are done with the analysis of categorical attributes and found lots of interesting things a
nd answered many unknown questions.  Now we will incorporate the required changes of df_cat in
to df.
```

Every attribute except `car_name` is of interest and participated in our analysis. So we will not add `car_name` to our dataframe as it is of no interest. This is `feature reduction` and is an integral part of `feature engineering`.

```
In [42]: df = pd.concat((df_cat.loc[:, 'origin':'car_company'], df.loc[:, 'mpg':'acceleration']), axis=1)
         df.head()
```

Out[42]:

	origin	cylinders	model_year	mpg_level	car_company	mpg	displacement	horsepower	weight	acceleration
0	usa	8	70	medium	chevrolet	18.0	307.0	130.0	3504	12.0
1	usa	8	70	low	buick	15.0	350.0	165.0	3693	11.5
2	usa	8	70	medium	plymouth	18.0	318.0	150.0	3436	11.0
3	usa	8	70	low	amc	16.0	304.0	150.0	3433	12.0
4	usa	8	70	medium	ford	17.0	302.0	140.0	3449	10.5

```
In [43]: # save these changes to new file
         df.to_csv("mpg_cated.csv", index=False)
```

Analysis on Numerical Attributes

Our analysis includes both descriptive stats and EDA.

```
In [44]: df = pd.read_csv("mpg_cated.csv")
         df.head()
```

Out[44]:

	origin	cylinders	model_year	mpg_level	car_company	mpg	displacement	horsepower	weight	acceleration
0	usa	8	70	medium	chevrolet	18.0	307.0	130.0	3504	12.0
1	usa	8	70	low	buick	15.0	350.0	165.0	3693	11.5
2	usa	8	70	medium	plymouth	18.0	318.0	150.0	3436	11.0
3	usa	8	70	low	amc	16.0	304.0	150.0	3433	12.0
4	usa	8	70	medium	ford	17.0	302.0	140.0	3449	10.5

```
In [45]: print(f'numerical variables:  {nums}')

numerical variables:  ['mpg', 'displacement', 'horsepower', 'weight', 'acceleration']
```

```
We will first slice out the numerical columns from original dataframe & then do analysis on it
 keeping the original data untouched,and at the end incorporate needed changes in our original
 dataframe.
```

```
In [46]: df_num = df.loc[:, 'mpg':]
```

```
Let's first look at some descriptive stats.
```

```
In [47]: df_num.describe()
```

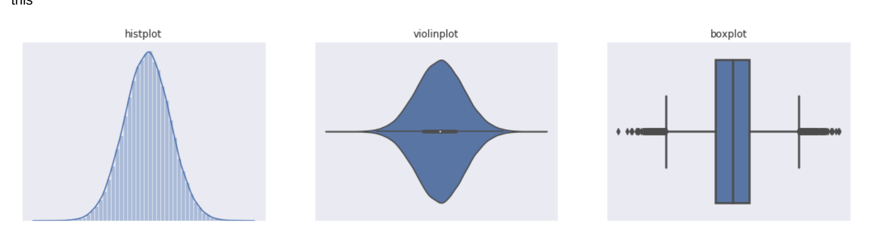
Out[47]:

	mpg	displacement	horsepower	weight	acceleration
count	392.000000	392.000000	392.000000	392.000000	392.000000
mean	23.445918	194.411990	104.469388	2977.584184	15.541327
std	7.805007	104.644004	38.491160	849.402560	2.758864
min	9.000000	68.000000	46.000000	1613.000000	8.000000
25%	17.000000	105.000000	75.000000	2225.250000	13.775000
50%	22.750000	151.000000	93.500000	2803.500000	15.500000
75%	29.000000	275.750000	126.000000	3614.750000	17.025000
max	46.600000	455.000000	230.000000	5140.000000	24.800000

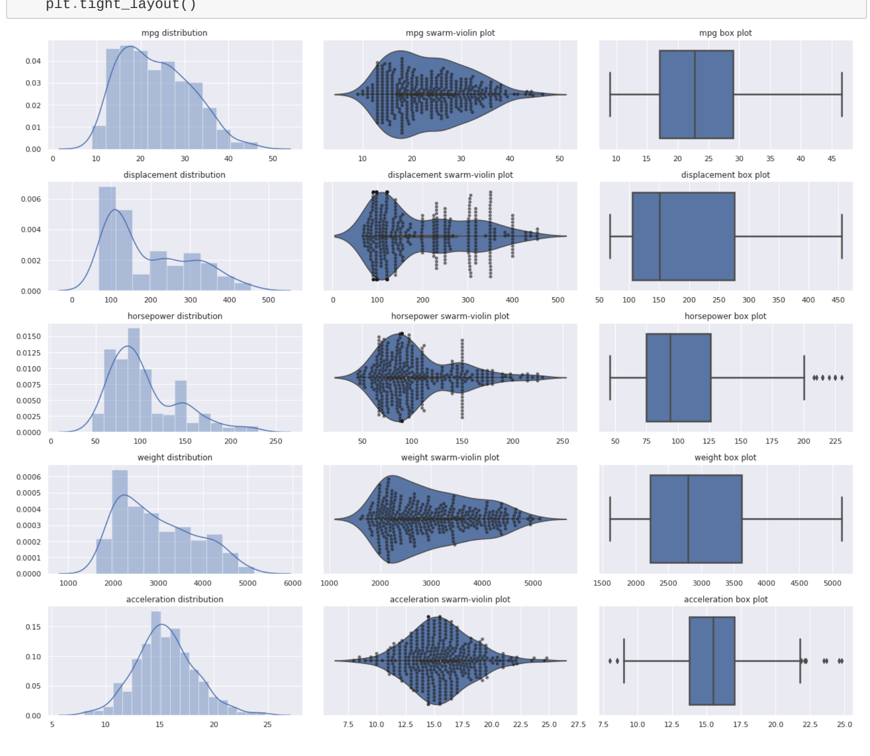
Analysis of Distribution

Now we analyse the distribution for each numerical attribute and make some insights from the plots.

In case of numerical variables an ideal (or atleast loved) distribution is `gaussian`, for a gaussian it's various distribution plots look like this




```
In [48]: rows = len(nums)
         cols = 3

         fig = plt.figure(1, (18, rows*3))

         i = 0
         for col in nums:

             i += 1
             ax1 = plt.subplot(rows, cols,i)
         #     ax1.hist(df[col], alpha=0.6)
             sns.distplot(df_num[col])
             ax1.set_xlabel(None)
             ax1.set_title(f'{col} distribution')
             plt.tight_layout()

             i += 1
             ax2 = plt.subplot(rows, cols,i)
             sns.violinplot(df_num[col])
             sns.swarmplot(df_num[col], alpha=0.6, color='k')
             ax2.set_xlabel(None)
             ax2.set_title(f'{col} swarm-violin plot')
             plt.tight_layout()

             i += 1
             ax3 = plt.subplot(rows, cols,i)
             sns.boxplot(df_num[col], orient='h', linewidth=2.5)
             ax3.set_xlabel(None)
             ax3.set_title(f'{col} box plot')
             plt.tight_layout()
```



Insights

- `acceleration` is the only distribution which is gaussian. There are few values in acceleration which lie outside the whiskers(the bars extending outwards from the box), these are fliers/outliers.
- distributions of `mpg` & `weight` seems to be `right-skewed gaussian`.
- distributions of `displacement` & `horsepower` seems to be far from gaussian.

Currently we are analysing the distributions just from plots, in the next phase(statistical analysis) we will do hypothesis testing for the normality of these distributions.

```
Let's analyse the outliers using tukey formula.
```

```
In [49]: def tukey_outliers(x):
             q1 = np.percentile(x,25)
             q3 = np.percentile(x,75)

             iqr = q3-q1

             min_range = q1 - iqr*1.5
             max_range = q3 + iqr*1.5

             outliers = x[(x<min_range) | (x>max_range)]
             return outliers
```

```
In [50]: for col in nums:
             outliers = tukey_outliers(df_num[col])
             if len(outliers):
                 print(f"* {col} has these tukey outliers,\n{outliers}\n")
             else:
                 print(f"* {col} doesn't have any tukey outliers.\n")

* mpg doesn't have any tukey outliers.

* displacement doesn't have any tukey outliers.

* horsepower has these tukey outliers,
6      220.0
7      215.0
8      225.0
13     225.0
25     215.0
27     210.0
66     208.0
93     215.0
94     225.0
115    230.0
Name: horsepower, dtype: float64

* weight doesn't have any tukey outliers.

* acceleration has these tukey outliers,
7       8.5
9       8.5
11      8.0
58     23.5
193    22.2
194    22.1
207    21.9
297    24.8
298    22.2
324    23.7
388    24.6
Name: acceleration, dtype: float64
```

`acceleration` and `horsepower` are the only attributes with tukey outliers and we can also notice this from the above boxplots.

```
In [51]: df.iloc[list(tukey_outliers(df_num.acceleration).index)]
```

Out[51]:

	origin	cylinders	model_year	mpg_level	car_company	mpg	displacement	horsepower	weight	acceleration
7	usa	8	70	low	plymouth	14.0	440.0	215.0	4312	8.5
9	usa	8	70	low	amc	15.0	390.0	190.0	3850	8.5
11	usa	8	70	low	plymouth	14.0	340.0	160.0	3609	8.0
58	europe	4	72	medium	volkswagen	23.0	97.0	54.0	2254	23.5
193	usa	4	76	medium	chevrolet	29.0	85.0	52.0	2035	22.2
194	usa	4	76	medium	chevrolet	24.5	98.0	60.0	2164	22.1
207	europe	4	76	medium	peugeot	19.0	120.0	88.0	3270	21.9
297	europe	4	79	medium	peugeot	27.2	141.0	71.0	3190	24.8
298	usa	8	79	medium	oldsmobile	23.9	260.0	90.0	3420	22.2
324	europe	4	80	high	vw	43.4	90.0	48.0	2335	23.7
388	europe	4	82	high	vw	44.0	97.0	52.0	2130	24.6

```
In [52]: df.iloc[list(tukey_outliers(df_num.horsepower).index)]
```

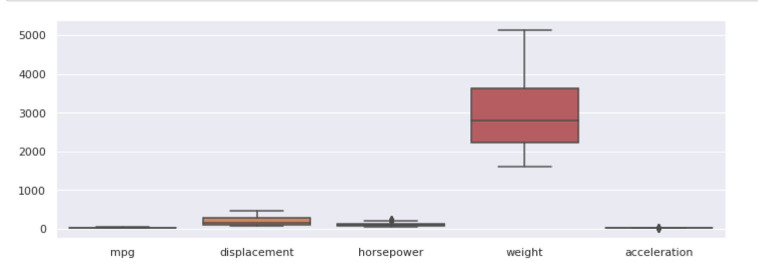
Out[52]:

	origin	cylinders	model_year	mpg_level	car_company	mpg	displacement	horsepower	weight	acceleration
6	usa	8	70	low	chevrolet	14.0	454.0	220.0	4354	9.0
7	usa	8	70	low	plymouth	14.0	440.0	215.0	4312	8.5
8	usa	8	70	low	pontiac	14.0	455.0	225.0	4425	10.0
13	usa	8	70	low	buick	14.0	455.0	225.0	3086	10.0
25	usa	8	70	low	ford	10.0	360.0	215.0	4615	14.0
27	usa	8	70	low	dodge	11.0	318.0	210.0	4382	13.5
66	usa	8	72	low	mercury	11.0	429.0	208.0	4633	11.0
93	usa	8	73	low	chrysler	13.0	440.0	215.0	4735	11.0
94	usa	8	73	low	buick	12.0	455.0	225.0	4951	11.0
115	usa	8	73	low	pontiac	16.0	400.0	230.0	4278	9.5

Insights

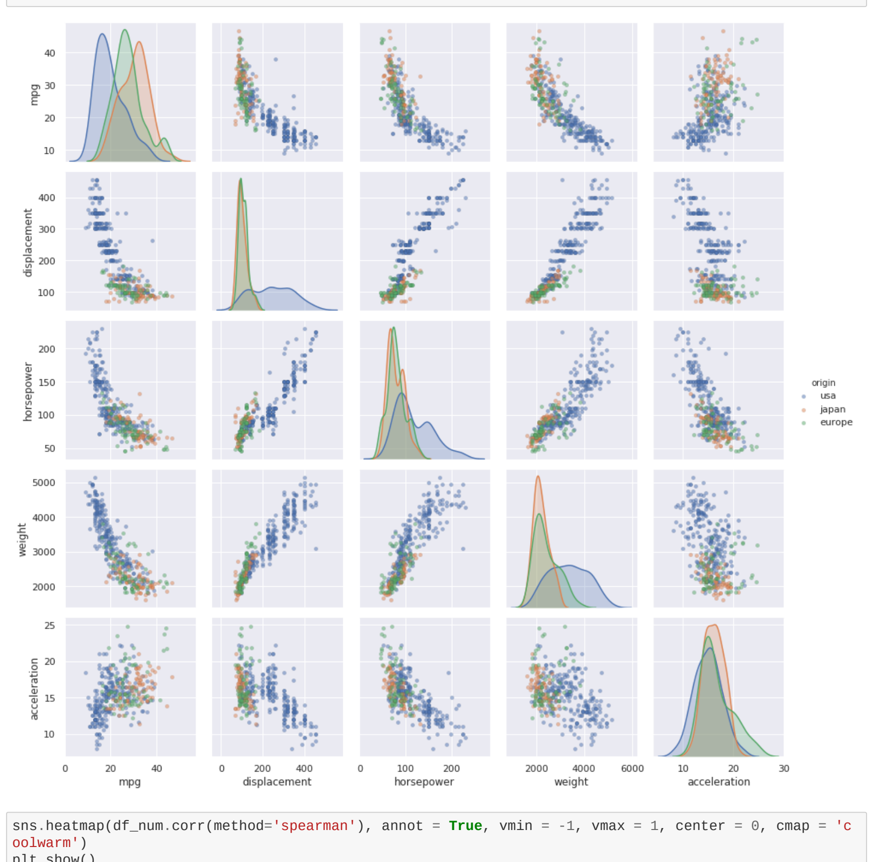
- Outliers in acceleration seems to be random nothing conclusive. One thing we can notice that none of them are from japan.
- Outliers in horsepower are not seeming random, there is a lot common in them
  - All of them are from usa (maybe because vehicles from usa are in majority).
  - All of them has 8 cylinders.
  - All of them has low mpg level.
  - All of them has weight in the range 4000.
  - Most of them has displacement in range 400.
  - Most of them were manufactured in early years(before 74).

```
In [53]: # see data is not scaled properly, we need to scale it for modelling but works fine for analysis.
         fig = plt.figure(1, (12, 4))
         ax = plt.subplot(1,1,1)
         sns.boxplot(x="variable", y="value", data=pd.melt(df_num))
         plt.xlabel(None)
         plt.ylabel(None)
         plt.show()
```

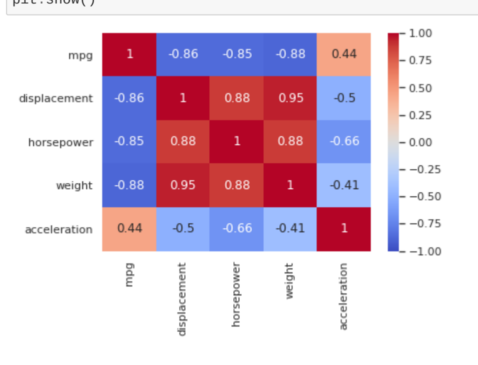


Now we analyze relationship between different numerical attributes

```
In [54]: sns.pairplot(data=df, vars=nums, diag_kind='kde', hue='origin',
                      plot_kws=dict(s=20, edgecolor="k", linewidth=0.1, alpha=0.5), diag_kws=dict(shade=True))
         plt.show()
```




```
In [55]: sns.heatmap(df_num.corr(method='spearman'), annot = True, vmin = -1, vmax = 1, center = 0, cmap = 'c
         oolwarm')
         plt.show()
```



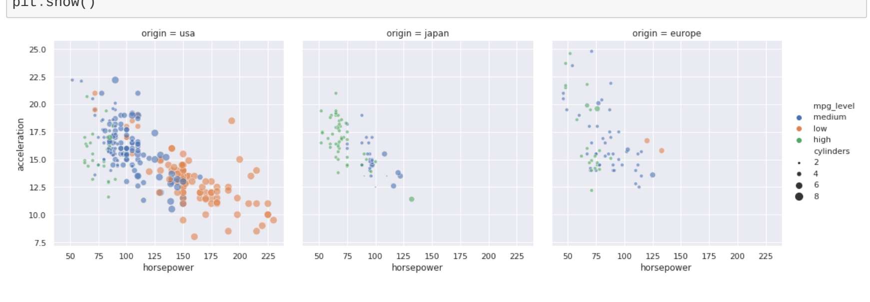
Insights

- as mpg increases displacement, horsepower & weight decreases but acceleration increases.
- as horsepower increases displacement & weight increases but acceleration decreases.
- as weight increases displacement increases but acceleration decreases.
- as acceleration increases displacement decreases.

So all numerical attributes are related with each other.

```
Now we bifurcate these relationships with different categories.
```

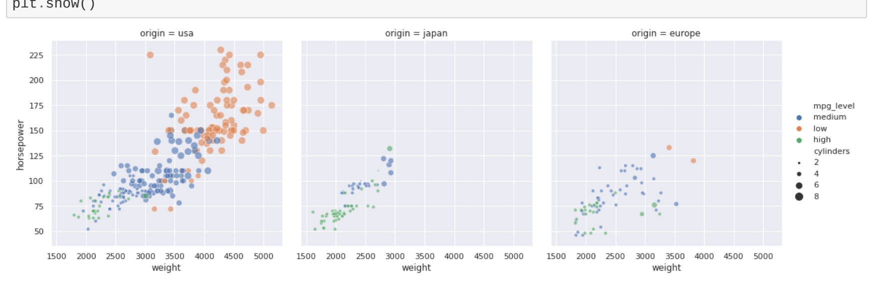
```
In [56]: '''In this plot we analyze the relationship of horsepower & acceleration
         bifurcated by origin, mpg_level & cylinders in a single plot.'''

         sns.relplot(x='horsepower', y='acceleration', hue='mpg_level', #style='mpg_level',
                     size='cylinders', col='origin', data=df, kind='scatter', sizes=(5, 100), alpha=0.6)
         plt.show()
```



Insights

- In every region there is a negative relation b/w horsepower & acceleration.
- vehicles with low mpg has low acceleration and high horsepower.
- vehicles with more cylinders has low acceleration and high horsepower.

```
In [57]: '''In this plot we analyze the relationship of weight & horsepower
         bifurcated by origin, mpg_level & cylinders in a single plot.'''

         sns.relplot(x='weight', y='horsepower', hue='mpg_level', #style='mpg_level',
                     size='cylinders', col='origin', data=df, kind='scatter', sizes=(5, 100), alpha=0.6)
         plt.show()
```



Insights

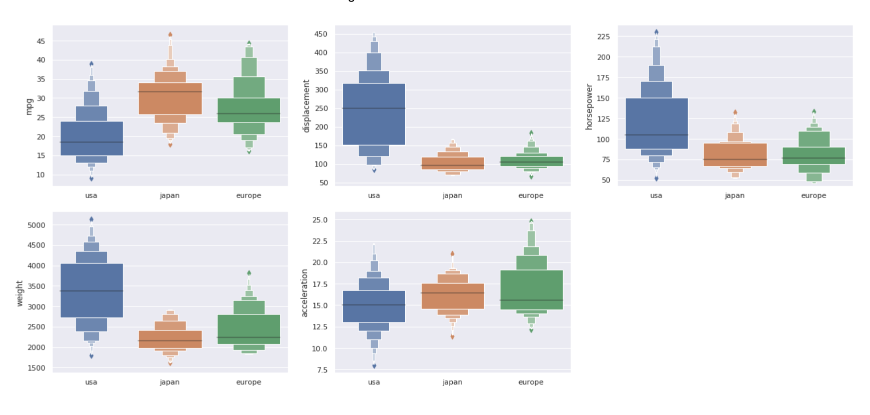
- In every region there is a positive relation b/w weight & horsepower.
- vehicles with low mpg has high weight & horsepower.
- vehicles with more cylinders has high weight & horsepower.

```
on bifurcating we didn't found anything new.
```

Now we analyze relationship between numerical and categorical attributes

Note: We are using boxen and violin plots for this but we can also use stripplot.

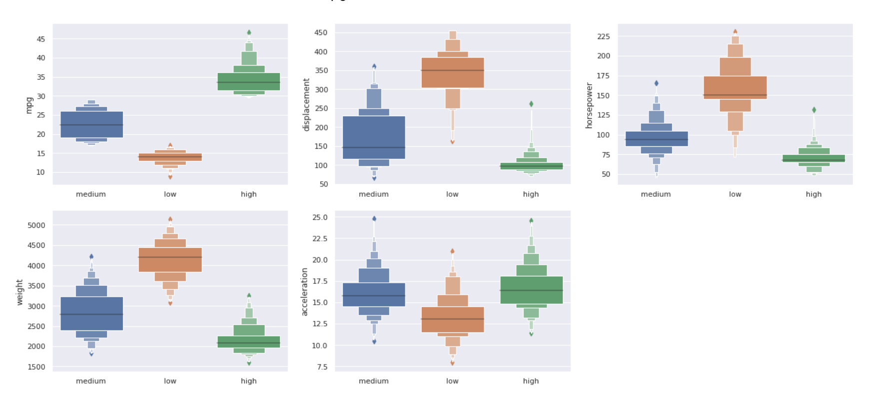
```
In [58]: print('variation of numerical features with origin')

         fig = plt.figure(1, (18, 8))

         for idx,col in enumerate(nums):
             ax = plt.subplot(2, 3, idx+1)
             sns.boxenplot(x='origin', y=col, data=df)
             ax.set_xlabel(None)
             plt.tight_layout()

variation of numerical features with origin
```



Insights

- vehicles of `usa` has less mpg on an average as compared to japan & europe.
- vehicles of `usa` has more displacement, horsepower and weight as compared to japan & europe.
- all vehicles has relatively same acceleration irrespective of the origin but distribution of acceleration from `usa` is more spreaded due to the fact that it comprises a lot vehicles compared to other.

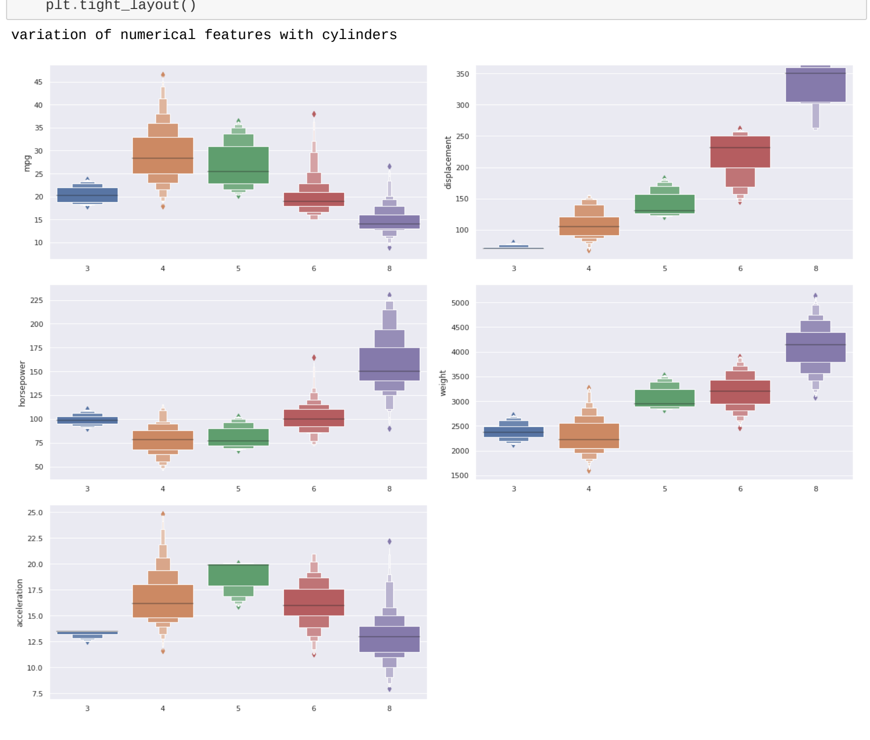
```
In [59]: print('variation of numerical features with mpg_level')

         fig = plt.figure(1, (18, 8))

         for idx,col in enumerate(nums):
             ax = plt.subplot(2, 3, idx+1)
             sns.boxenplot(x='mpg_level', y=col, data=df)
             ax.set_xlabel(None)
             plt.tight_layout()

variation of numerical features with mpg_level
```



Insights

- as mpg_level increases displacement decreases on average.
- as mpg_level increases horsepower decreases on average.
- as mpg_level increases weight decreases on average.
- vehicles with low mpg_level usually has less acceleration compared to other whereas vehicles with medium and high mpg_level has same acceleration.

```
In [60]: print('variation of numerical features with cylinders')

         fig = plt.figure(1, (18, 14))

         for idx,col in enumerate(nums):
             ax = plt.subplot(3, 2, idx+1)
             sns.boxenplot(x='cylinders', y=col, data=df)
             ax.set_xlabel(None)
             plt.tight_layout()

variation of numerical features with cylinders
```

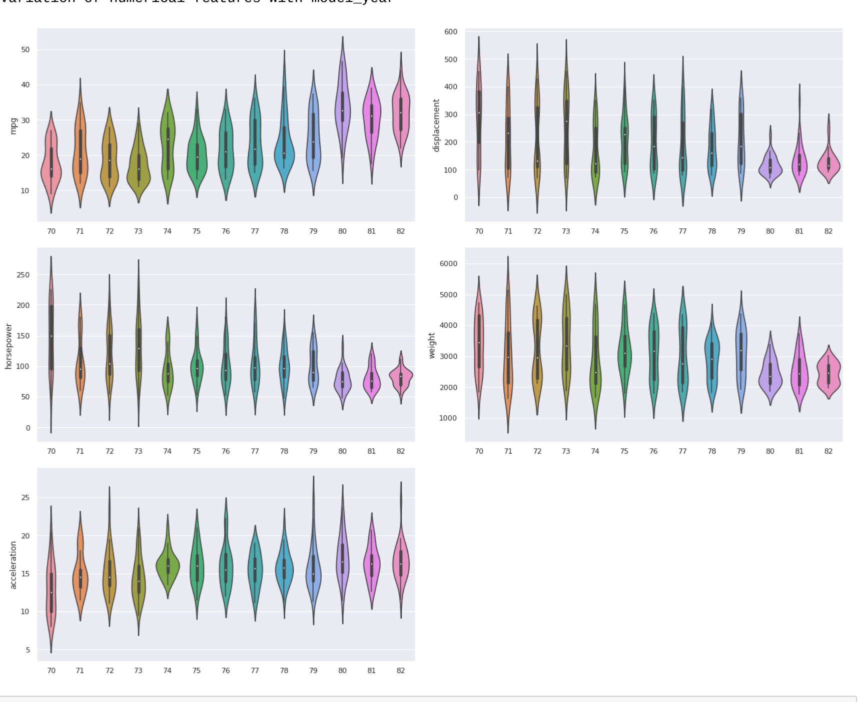


Insights

- as cylinders increases from 3 to 4 mpg also increases but on further increasing the cylinders mpg starts decreasing.
- displacement increases in polynomial order as cylinders increases.
- as cylinders increases from 3 to 5 horsepower decreases but on further increasing the cylinders it starts increasing.
- on increasing cylinders vehicle's weight increases on average (very obvious).
- as cylinders increases from 3 to 5 vehicle's acceleration also increases but on further increasing the cylinders it starts decreasing(maybe due to the fact that vehicles with more cylinders have more weight and hence less acceleration).

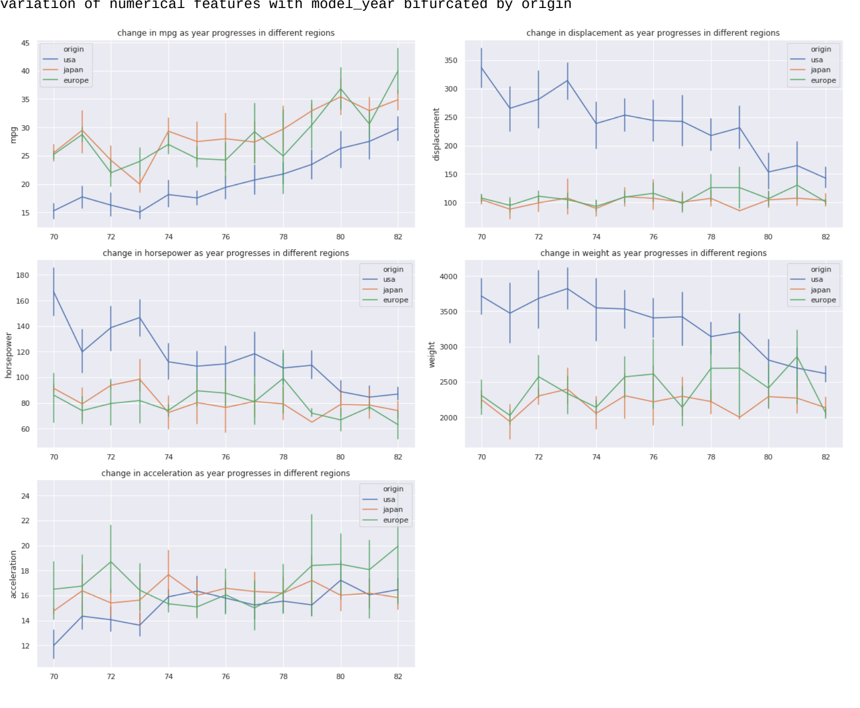
In [61]:
```python
print('variation of numerical features with model_year')

fig = plt.figure(1, (18, 14))
# fig.tight_layout()

for idx,col in enumerate(nums):
    ax = plt.subplot(3, 2, idx+1)
    sns.violinplot(x='model_year', y=col, data=df)
    ax.set_xlabel(None)
    plt.tight_layout()
```

```
variation of numerical features with model_year
```



In [62]:
```python
print('variation of numerical features with model_year bifurcated by origin')

fig = plt.figure(1, (18, 14))
# fig.tight_layout()

for idx,col in enumerate(nums):
    ax = plt.subplot(3, 2, idx+1)
    sns.lineplot(x="model_year", y=col, hue='origin', data=df, err_style='bars')
    ax.set_title(f'change in {col} as year progresses in different regions')
    ax.set_xlabel(None)
    plt.tight_layout()

plt.show()
```

```
variation of numerical features with model_year bifurcated by origin
```



Insights

- as year progresses there is an increase in mpg across all origins(this we already observed in analysis on categorical data).
- as year progresses there is a slight decrease in displacement, horsepower & weight of the vehicles belonging to `usa` but there is no significant change in `japan` & `europe`.One thing we can observe is that in the 80's all vehicles have similar displacement because unlike the 70's the distribution is not spread out (i.e., distribution is **short fatty** instead of **tall skiny**).
- throughout the years acceleration remains relatively the same across all regions.

So we are done for now. We did some good amount of EDA and also explored various plotting features provided to us by seaborn. I highly recommend you all to use seaborn because it's easy and simple. Also you can use plotly to contruct all the graphs in this notebook. Plotly plots are not only visually awesome but are also interactive and due to this they take much more memory specially scatter plots (as they need to store every data-point info. to enable intractivity). That's why I didn't included any plotly plots.

In the next part we will do some **Statistical Analysis**.

If you like this notebook then please upvote and also share with others.

In [ ]: