Dataset Description

The MPG dataset is technical spec of cars originaly provided from UCI Machine Learning Repository. The data concerns city-cycle fuel consumption in miles per gallon to be analyzed in terms of 3 multivalued discrete and 5 continuous attributes. ___

Columns Description

1. `mpg` : miles per galon of fuel (continuous variable).
2. `cylinders` : number of engine cylinders (multi-valued discrete variable).
3. `displacement` : (continuous variable)
4. `horsepower` : the power produced by engine to move the car (continuous variable)
5. `weight` : car weight (continuous variable)
6. `acceleration` : the acceleration an engine can get per second (continuous variable)
7. `model year` : car release year from 1970 to 1982(multi-valued discrete variable)
8. `origin` : car manufacturing place (1 -> USA, 2 -> Europe, 3 -> Asia) (multi-valued discrete variable)
9. `car name` : car model name (unique for each instance)

Environment set-up

In [63]:
```
# import required libraries

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

Data Wrangling

We'd load our desired data from the flat csv file `auto-mpg.csv` to a dataframe using `pandas` , and display its first 20 records. here, we want to check for:

- Missingness in our dataframe.
- Inconsistent data types.
- Duplicated rows.
- columns to be droped or re-parsed.

In [64]:
```
#Load Data
mpg_df = pd.read_csv(r"D:\Projects\My portofolio - Completed projects\Python\2024 10 Auto MPG Dataset Analysis\a

#cheack top rows
mpg_df.head(20)
```

Out[64]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	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
5	15.0	8	429.0	198	4341	10.0	70	1	ford galaxie 500
6	14.0	8	454.0	220	4354	9.0	70	1	chevrolet impala
7	14.0	8	440.0	215	4312	8.5	70	1	plymouth fury iii
8	14.0	8	455.0	225	4425	10.0	70	1	pontiac catalina
9	15.0	8	390.0	190	3850	8.5	70	1	amc ambassador dpl
10	15.0	8	383.0	170	3563	10.0	70	1	dodge challenger se
11	14.0	8	340.0	160	3609	8.0	70	1	plymouth 'cuda 340
12	15.0	8	400.0	150	3761	9.5	70	1	chevrolet monte carlo
13	14.0	8	455.0	225	3086	10.0	70	1	buick estate wagon (sw)
14	24.0	4	113.0	95	2372	15.0	70	3	toyota corona mark ii
15	22.0	6	198.0	95	2833	15.5	70	1	plymouth duster
16	18.0	6	199.0	97	2774	15.5	70	1	amc hornet
17	21.0	6	200.0	85	2587	16.0	70	1	ford maverick
18	27.0	4	97.0	88	2130	14.5	70	3	datsun pl510
19	26.0	4	97.0	46	1835	20.5	70	2	volkswagen 1131 deluxe sedan

In [65]:

```
# display the number of rows and columns in the dataset

mpg_df.shape
```

Out[65]: (398, 9)

In [66]:

```
mpg_df.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   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    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
```

- `horsepower` column have inconsistant data type.

In [67]:

```
# check horsepower unique values
mpg_df.horsepower.unique()
```

Out[67]:

```
array(['130', '165', '150', '140', '198', '220', '215', '225', '190',
       '170', '160', '95', '97', '85', '88', '46', '87', '90', '113',
       '200', '210', '193', '?', '100', '105', '175', '153', '180', '110',
       '72', '86', '70', '76', '65', '69', '60', '80', '54', '208', '155',
       '112', '92', '145', '137', '158', '167', '94', '107', '230', '49',
       '75', '91', '122', '67', '83', '78', '52', '61', '93', '148',
       '129', '96', '71', '98', '115', '53', '81', '79', '120', '152',
       '102', '108', '68', '58', '149', '89', '63', '48', '66', '139',
       '103', '125', '133', '138', '135', '142', '77', '62', '132', '84',
       '64', '74', '116', '82'], dtype=object)
```

In [68]:

```
(mpg_df['horsepower'] == '?').sum()
```

Out[68]: 6

- It seems like we have `6` values in `horsepower` column containing `?`, and that is what is giving us the object data type instead of int.

In [69]:
```
#check data nulls
mpg_df.isnull().sum()
```

Out[69]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

In [70]:
```
#check for duplicated rows
mpg_df.duplicated().sum()
```

Out[70]:
```
0
```

Exploring Summary

- Our dataset has a total of 398 records and 9 columns.
- We have no NaNs in our dataset nor duplicated rows.
- `horsepower` column have inconsistant data type that needs to be handled and casted to `int`.
- `origin` would need to be parsed and casted into a categorical datatype.
- No columns would need to be dropped.

Data Cleaning

Here, we'd perform cleaning operations (dropping rows, mapping columns, converting data types). All of which would help us reach a more accurate result in creating meaningful and informative visualizations.

Cleaning `horsepower` column

We'd drop each row that contains a `?` on the `horsepower` column.

In [71]:
```
# droping '?' values
mpg_df = mpg_df[mpg_df.horsepower != '?']

# check change
(mpg_df.horsepower == '?').sum()
```

Out[71]:
```
0
```

In [72]:
```
#casting horsepower column to integer
mpg_df.horsepower = mpg_df.horsepower.astype(np.int64)

# check change
mpg_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 392 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    int64
 4   weight        392 non-null    int64
 5   acceleration  392 non-null    float64
 6   model year    392 non-null    int64
 7   origin        392 non-null    int64
 8   car name      392 non-null    object
dtypes: float64(3), int64(5), object(1)
memory usage: 30.6+ KB
```

Handling `origin` column

We'd change `origin` column according to the provided description (1 -> USA, 2 -> Europe, 3 -> Asia), and cast its datatype as category .

In [73]:
```
# changing origin column
mpg_df.loc[mpg_df.origin == 1, 'origin'] = 'USA'
mpg_df.loc[mpg_df.origin == 2, 'origin'] = 'Europe'
mpg_df.loc[mpg_df.origin == 3, 'origin'] = 'Asia'

# check data
mpg_df['origin'].unique()
```

```
C:\Users\Bassa\AppData\Local\Temp\ipykernel_22036\450501664.py:2: FutureWarning: Setting an item of incompatible
dtype is deprecated and will raise an error in a future version of pandas. Value 'USA' has dtype incompatible wi
th int64, please explicitly cast to a compatible dtype first.
  mpg_df.loc[mpg_df.origin == 1, 'origin'] = 'USA'
```

Out[73]:
```
array(['USA', 'Asia', 'Europe'], dtype=object)
```

In [74]:
```
# casting origin column into category

mpg_df.origin = mpg_df.origin.astype('category')

# validating changes
mpg_df.origin.dtype
```

Out[74]:
```
CategoricalDtype(categories=['Asia', 'Europe', 'USA'], ordered=False, categories_dtype=object)
```

Now, our dataframe is tidy and clean, and we are ready to move into visualizing it to get some meaningful insights! ___

Data Visualization

In this section, we'd use some informative visuals to help us draw insights and conclusions about our data and also help us in our EDA.

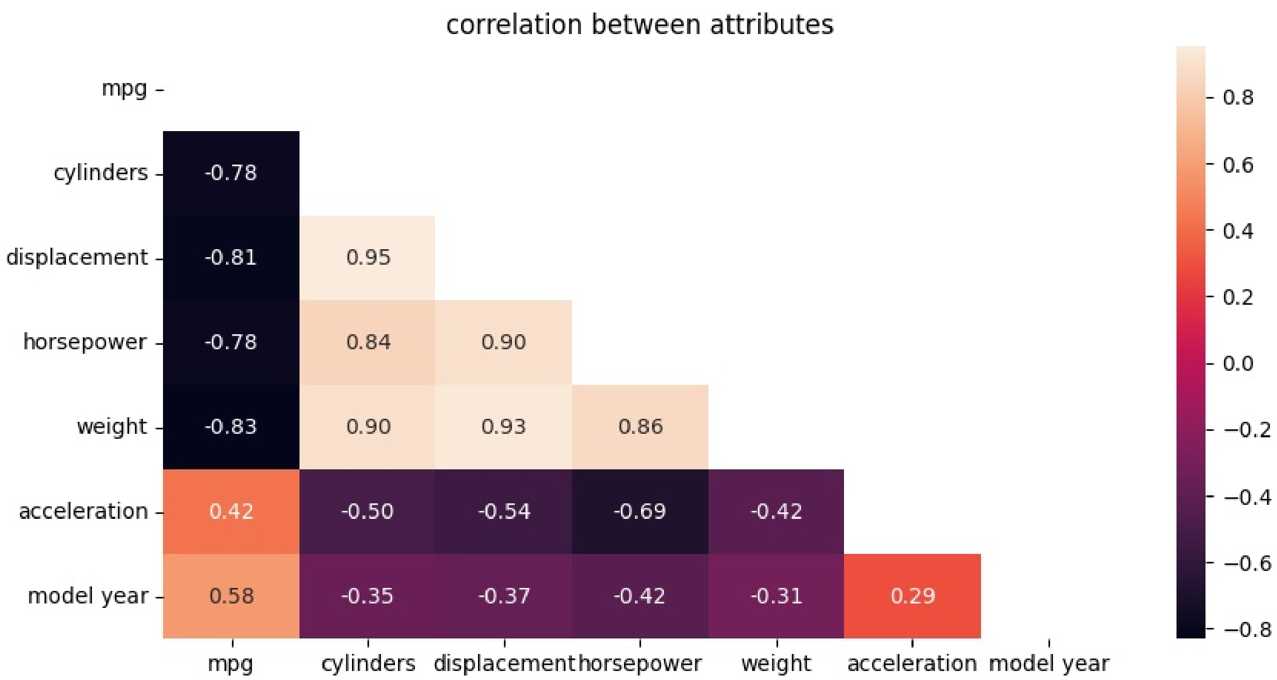
Let's first have a quick look at the summery statistics of our dataset.

In [75]:
```
# displaying summary statistics
mpg_df.describe()
```

Out[75]:

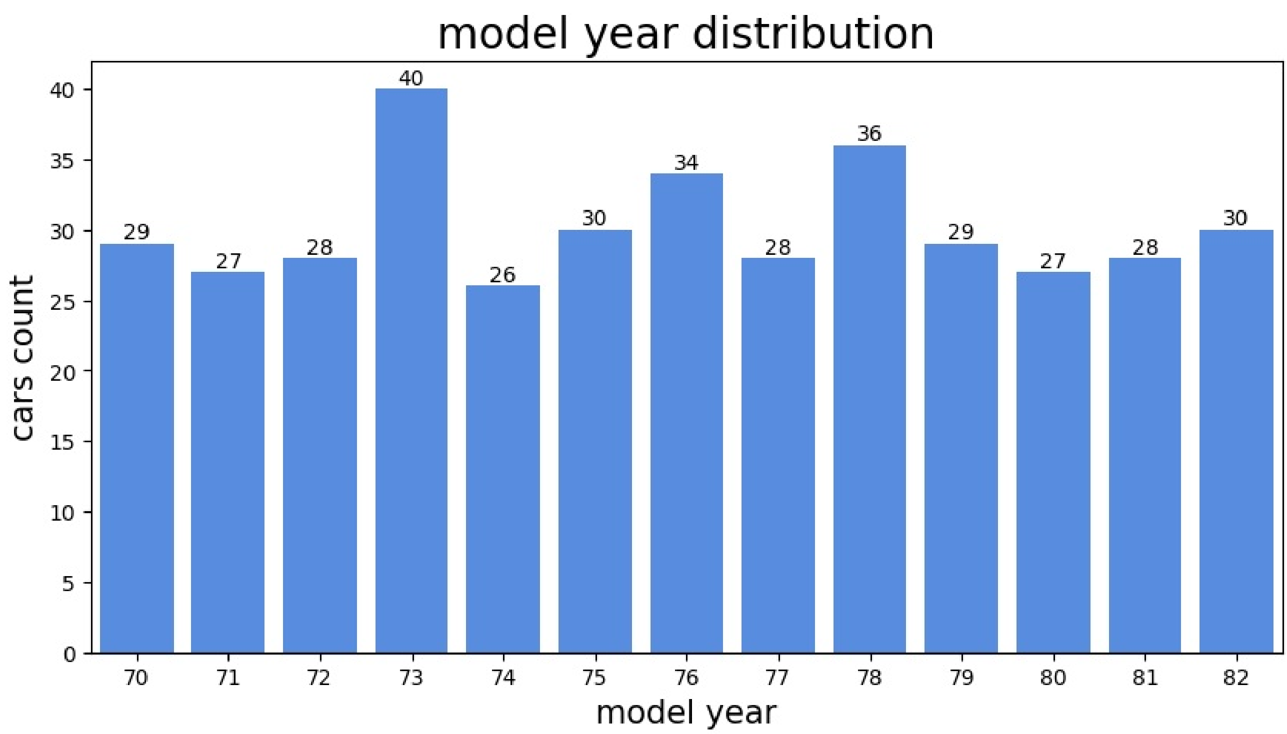
	mpg	cylinders	displacement	horsepower	weight	acceleration	model year
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	23.445918	5.471939	194.411990	104.469388	2977.584184	15.541327	75.979592
std	7.805007	1.705783	104.644004	38.491160	849.402560	2.758864	3.683737
min	9.000000	3.000000	68.000000	46.000000	1613.000000	8.000000	70.000000
25%	17.000000	4.000000	105.000000	75.000000	2225.250000	13.775000	73.000000
50%	22.750000	4.000000	151.000000	93.500000	2803.500000	15.500000	76.000000
75%	29.000000	8.000000	275.750000	126.000000	3614.750000	17.025000	79.000000
max	46.600000	8.000000	455.000000	230.000000	5140.000000	24.800000	82.000000

In [82]:
```
# plotting correlation heatmap
# Calculate correlation matrix using only numeric columns
numeric_cols = mpg_df.select_dtypes(include=[np.number]).columns
corr = mpg_df[numeric_cols].corr()
mask = np.triu(corr)

#heatmap
plt.figure(figsize=(10,5))
sns.heatmap(corr, annot= True, mask= mask, fmt = '.2f')
plt.title("correlation between attributes")
plt.show()
```

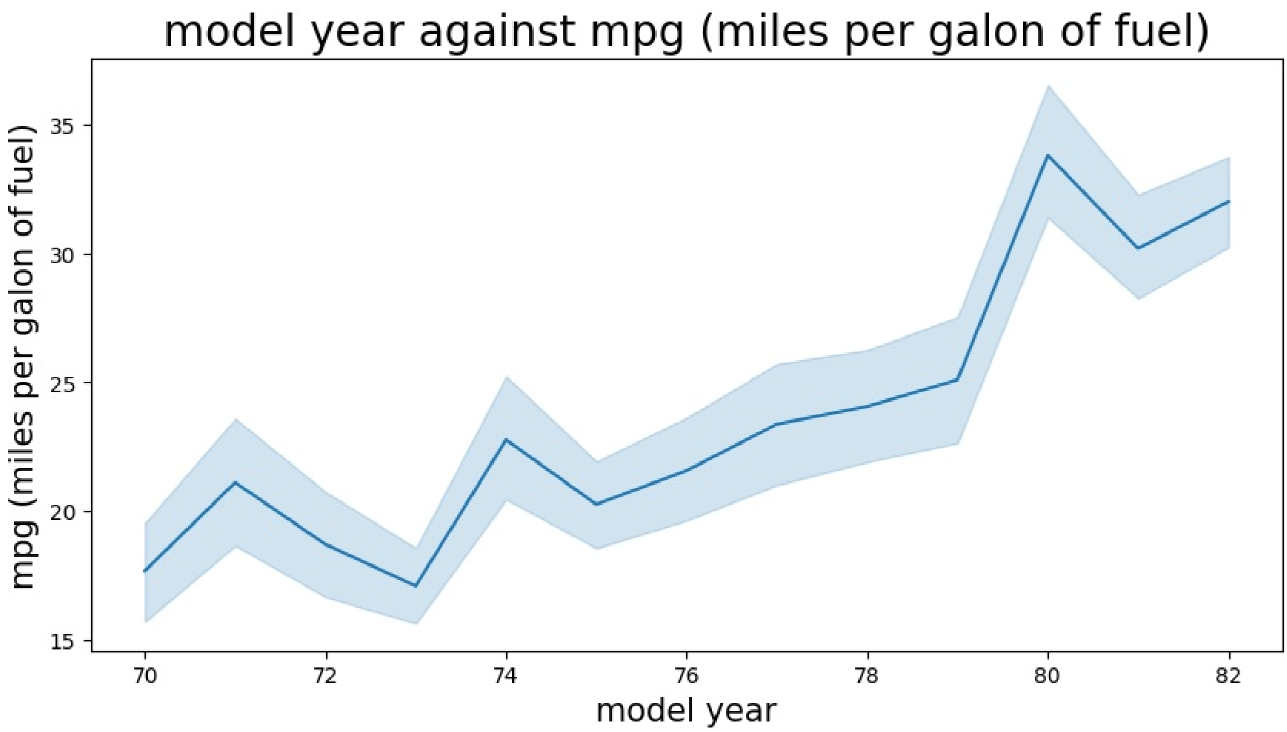


Taking a closer look into the `model year` distribution

In [89]:
```
# displaying histgram of model year column
plt.figure(figsize=(10,5))
ax = sns.countplot(x = 'model year', data = mpg_df, color = '#4287f5')
ax.bar_label(ax.containers[0], label_type='edge')
plt.title("model year distribution", fontsize = 20)
plt.xlabel("model year", fontsize = 15)
plt.ylabel("cars count", fontsize = 15)
plt.show()
```



In [98]:
```
# ploting model year against mpg
plt.figure(figsize=(10,5))
plt.title("model year against mpg (miles per galon of fuel)", fontsize = 20)
plt.xlabel("model year", fontsize = 15)
plt.ylabel("mpg (miles per galon of fuel)", fontsize = 15)
sns.lineplot(x = 'model year', y = 'mpg', data = mpg_df)
plt.show()
```



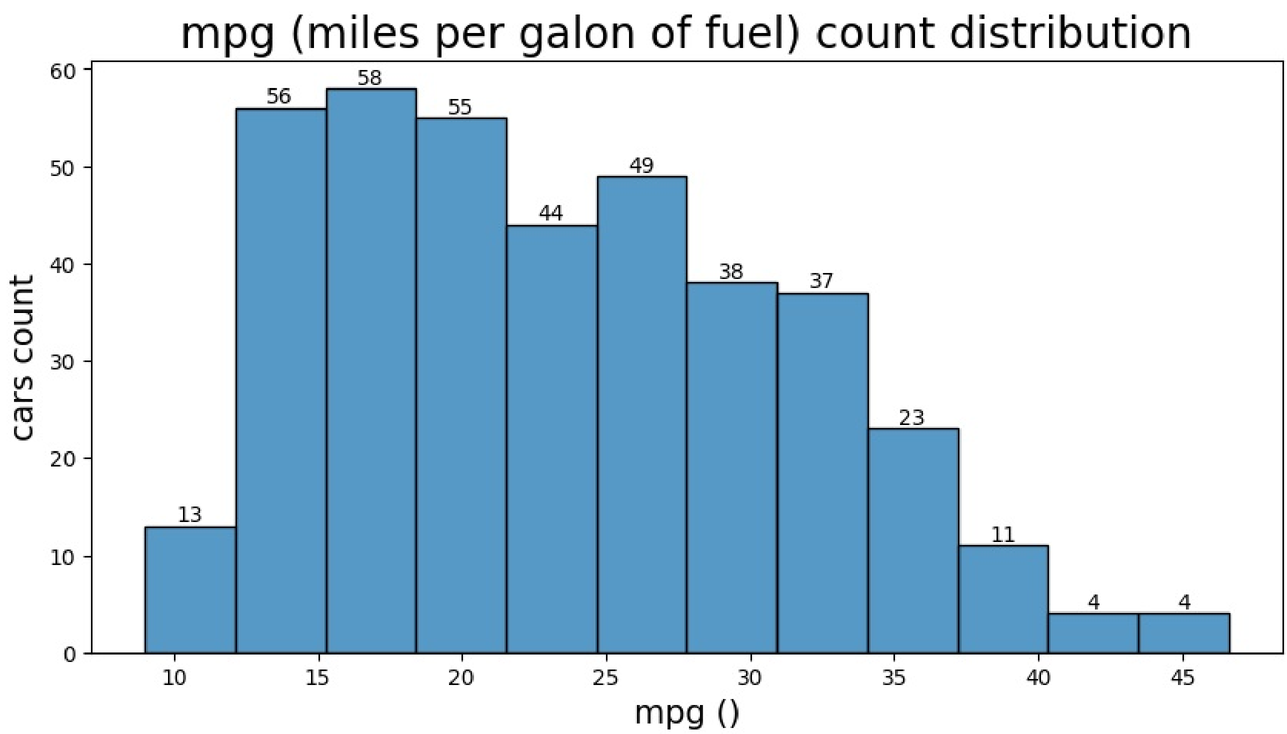
From the above visualizations, we we may totice the following:

- Our dataset contains info about cars from 1970 to 1982.
- Most of the cars are produced in `1973`.
- As years pass after `1973`, there has been a noticable increase in `mpg`.

Now, let's take a closer look at `mpg` coulmn.

In [109…

```
# ploting mpg distribution
plt.figure(figsize=(10,5))
ax = sns.histplot(x = 'mpg', data = mpg_df)
ax.bar_label(ax.containers[0], label_type='edge')
plt.title("mpg (miles per galon of fuel) count distribution", fontsize = 20)
plt.xlabel("mpg ()", fontsize = 15)
plt.ylabel("cars count", fontsize = 15)
plt.show()
```

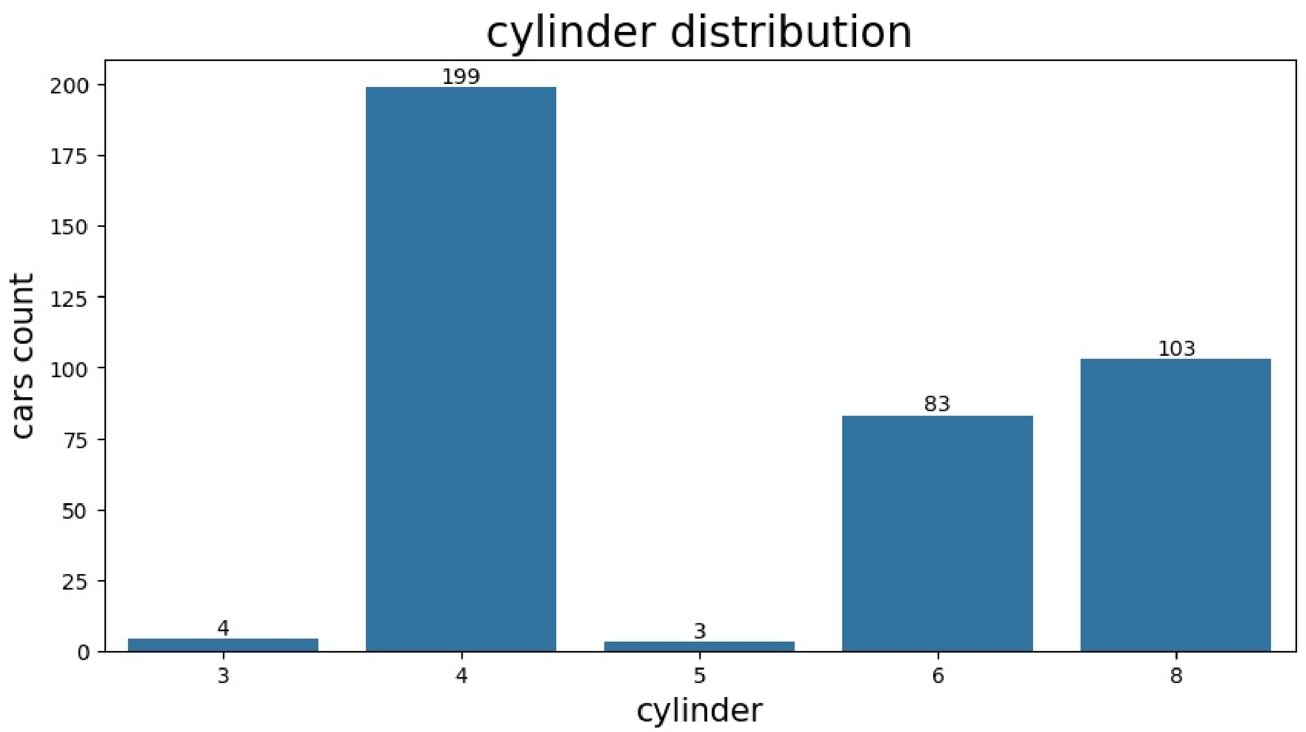


- Most of the cars in our dataset have mpg between 15 to 20. Our data is also skewed to the right.

let's compare these findings to `cylinders` and `horsepower` columns.

In [110…

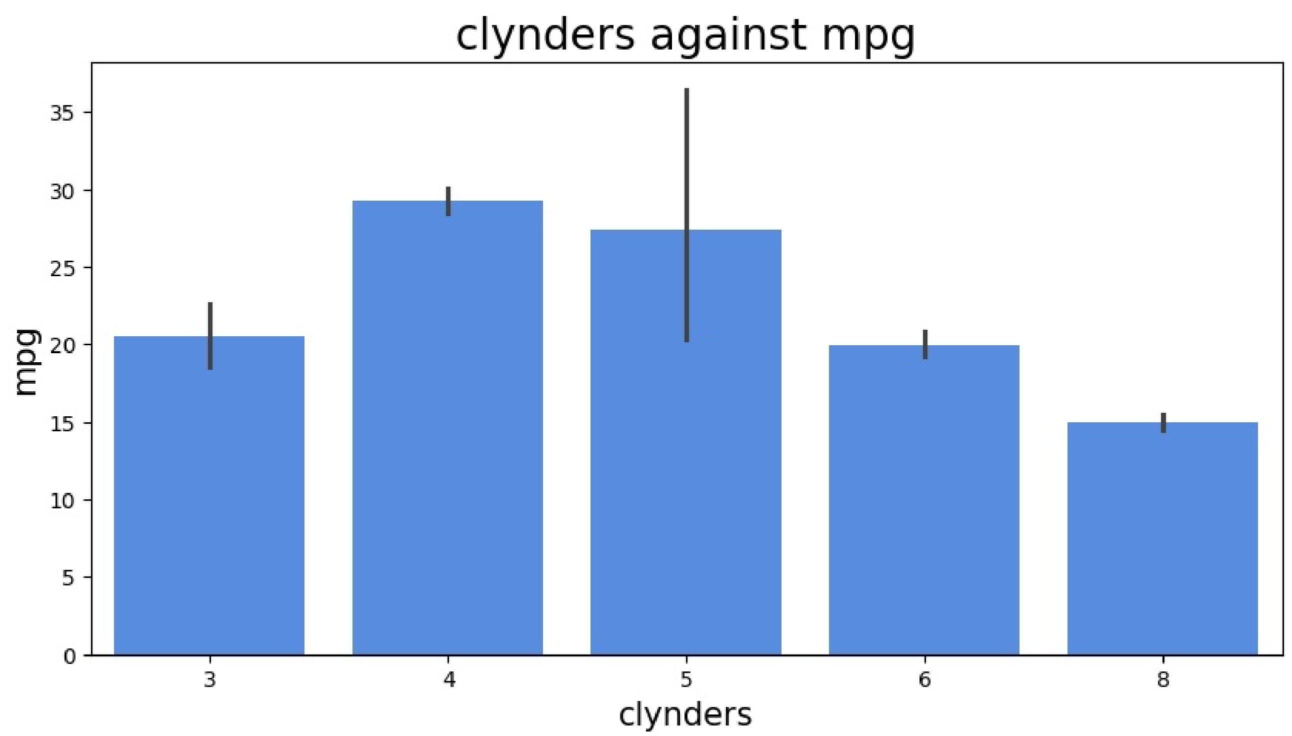
```
# displaying cylinder column distribution
plt.figure(figsize=(10,5))
ax = sns.countplot(x = 'cylinders', data = mpg_df)
ax.bar_label(ax.containers[0], label_type='edge')
plt.title("cylinder distribution", fontsize = 20)
plt.xlabel("cylinder", fontsize = 15)
plt.ylabel("cars count", fontsize = 15)
plt.show()
```



- The vast majority in cars have `4` cylinder engine.

In [111…

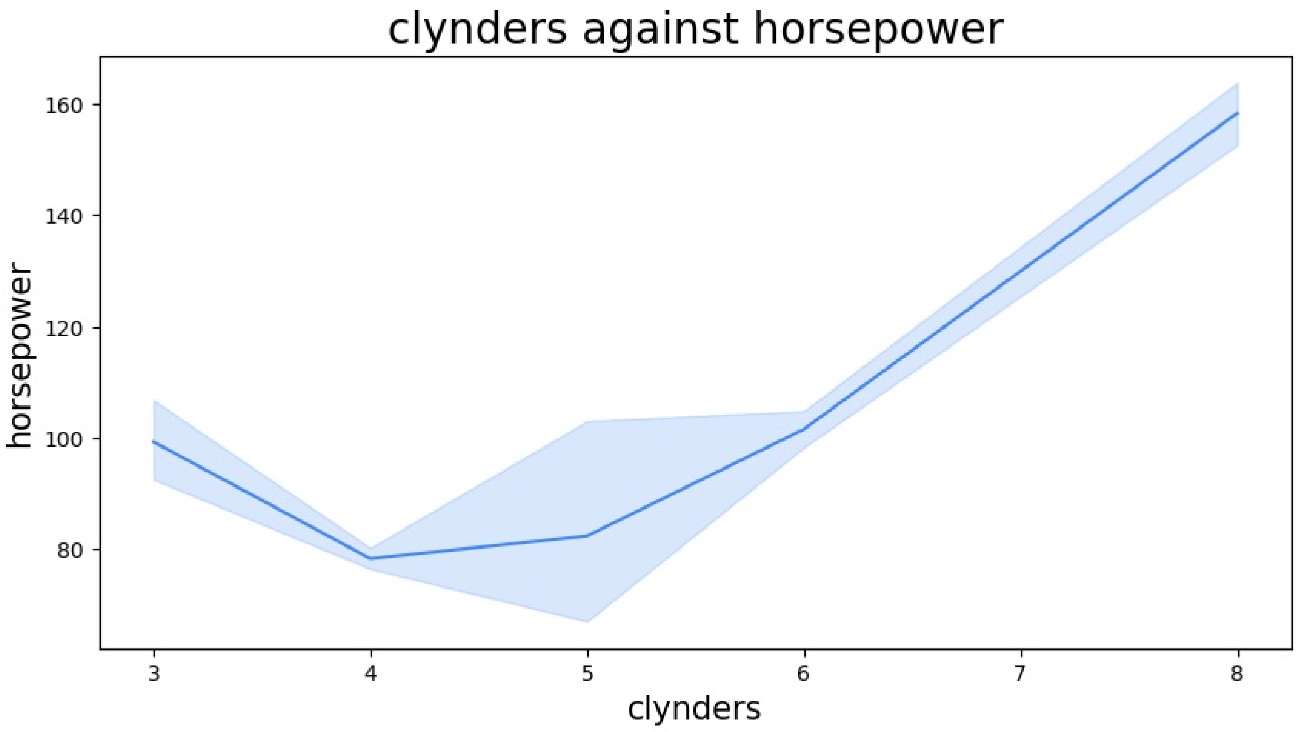
```
# ploting clynders against mpg
plt.figure(figsize=(10,5))
sns.barplot(x = 'cylinders', y = 'mpg', data = mpg_df, color = '#4287f5')
plt.title("clynders against mpg", fontsize = 20)
plt.xlabel("clynders", fontsize = 15)
plt.ylabel("mpg", fontsize = 15)
plt.show()
```



In [112…

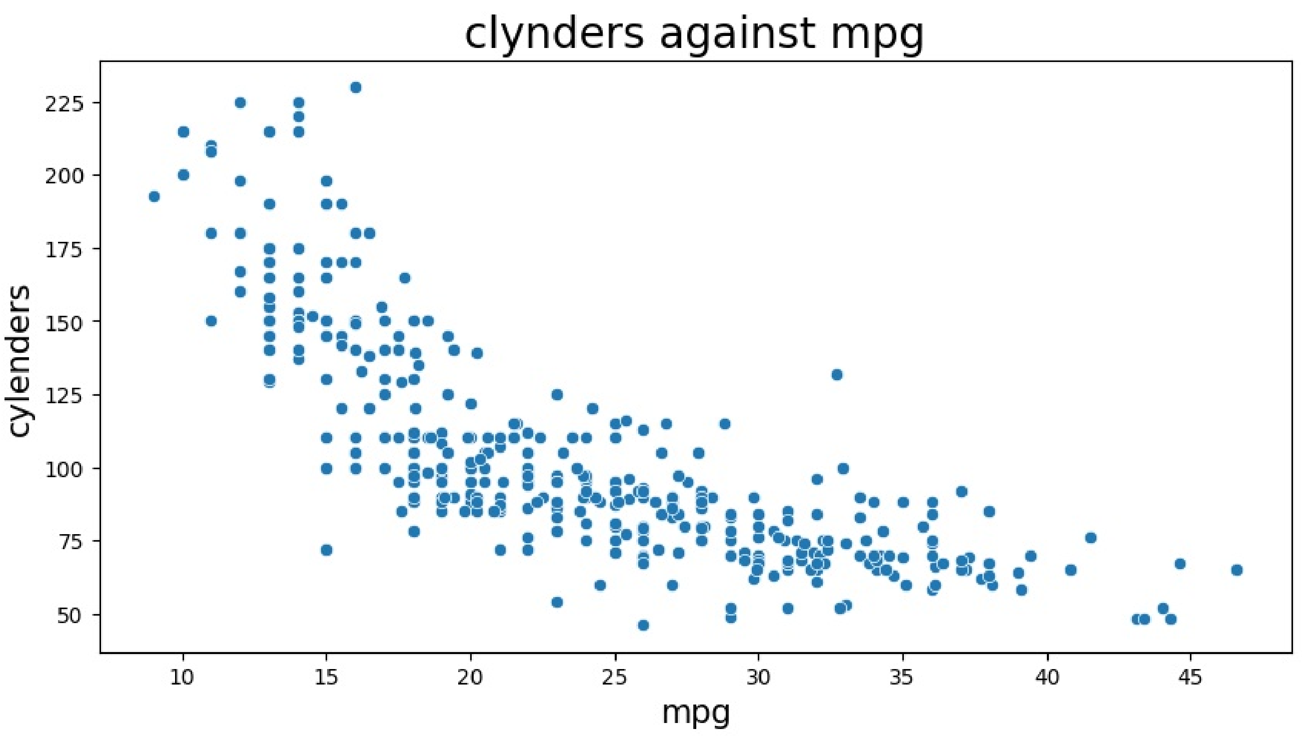
```
# ploting cylinders anainst horsepower
plt.figure(figsize=(10,5))
sns.lineplot(x = 'cylinders', y = 'horsepower', data = mpg_df, color = '#4287f5')
plt.title("clynders against horsepower", fontsize = 20)
```

```
plt.xlabel("clynders", fontsize = 15)
plt.ylabel("horsepower", fontsize = 15)
plt.show()
```



In [113…

```
# ploting mpg against horsepower
plt.figure(figsize=(10,5))
sns.scatterplot(x = 'mpg', y = 'horsepower', data = mpg_df)
plt.title("clynders against mpg", fontsize = 20)
plt.xlabel("mpg", fontsize = 15)
plt.ylabel("cylenders", fontsize = 15)
plt.show()
```



From the above visuals, we can notice that:

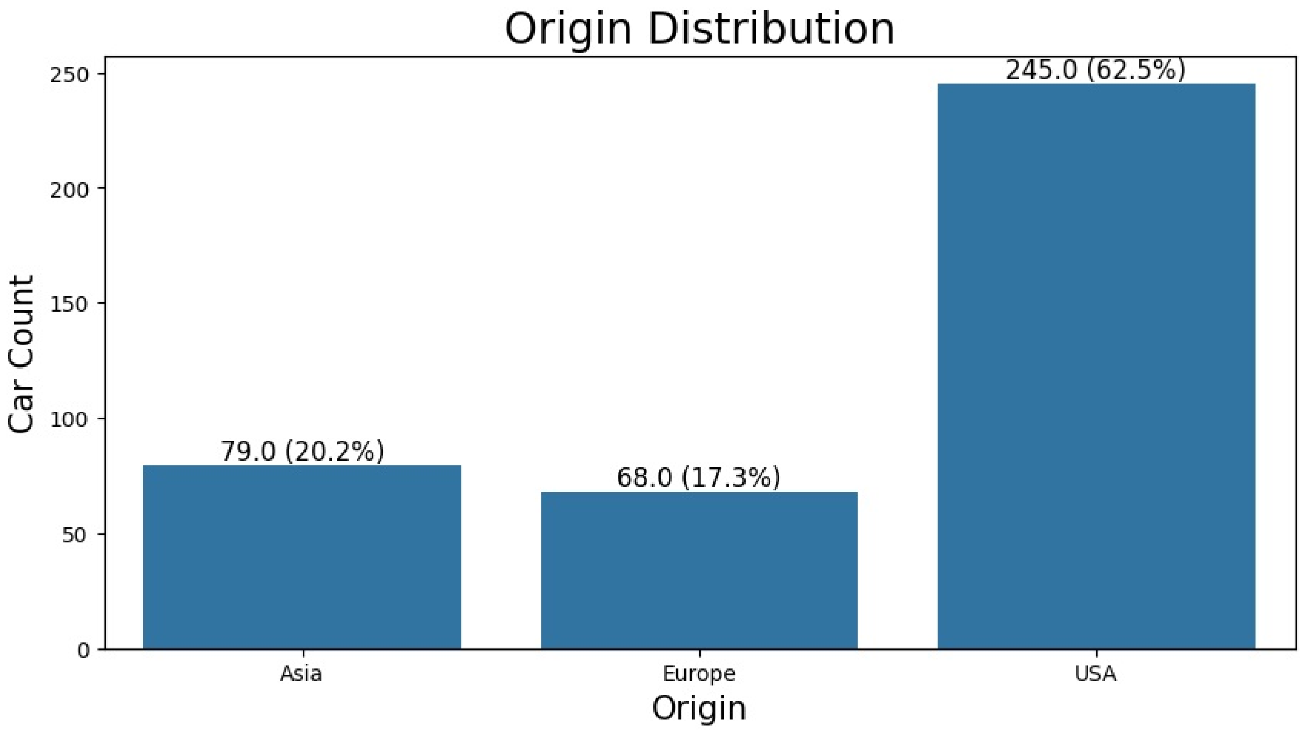
- As `cylinders` in the engine increases above 4, `MPG` decreases.
- As `cylinders` in the engine increases above 4, engine `horsepower` increases.
- there is negative correlation between `mpg` and `horsepower`.

Next, we'll move into exploring `origin` column and find insights about each manufacturing country.

In [117…

```
# Plotting cars origin distribution
plt.figure(figsize=(10, 5))
```

```python
ax = sns.countplot(x='origin', data=mpg_df)

# Calculate percentages and add labels
total_cars = len(mpg_df)
for p in ax.patches:
    count = p.get_height()
    percentage = (count / total_cars) * 100
    ax.annotate(f'{count} ({percentage:.1f}%)', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='bottom', fontsize=12)

# Title and labels
plt.title("Origin Distribution", fontsize=20)
plt.xlabel("Origin", fontsize=15)
plt.ylabel("Car Count", fontsize=15)

plt.show()
```

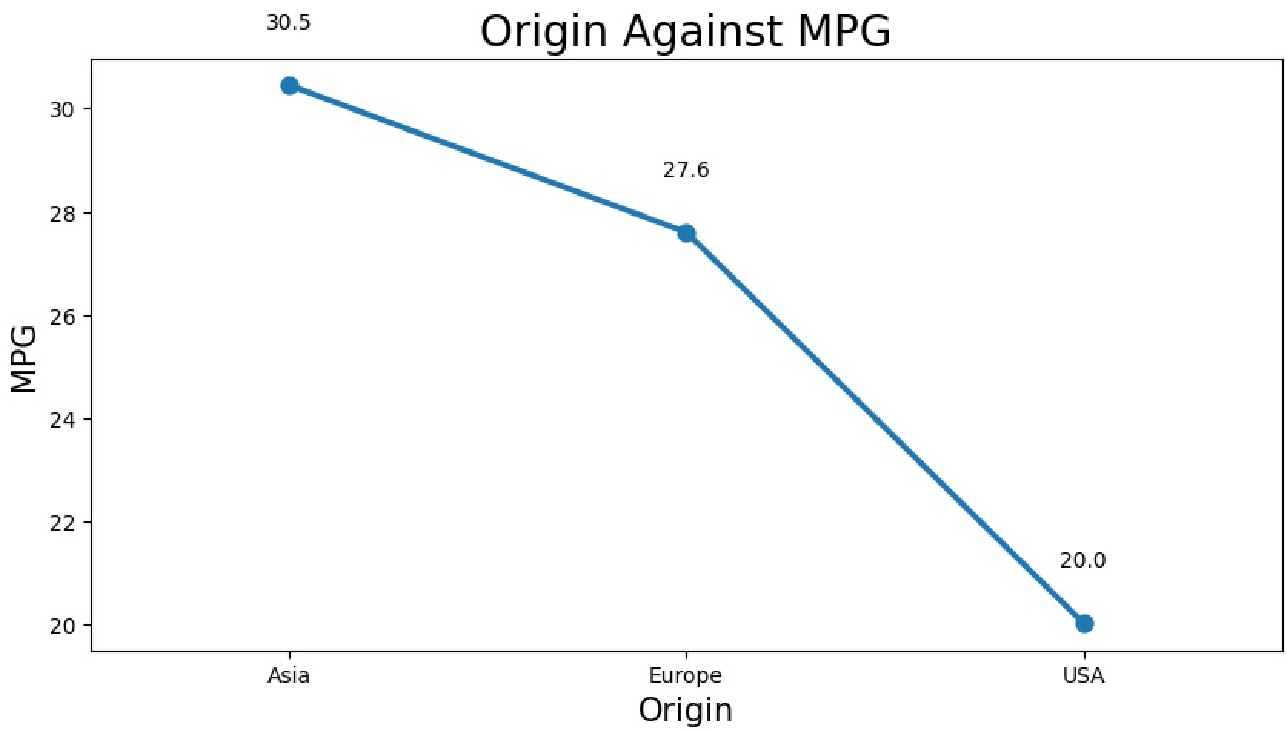


- We can see that `USA` alne produces more than 64% of the total cars in our dataset.

Let's explore its relations between other columns.

In [129…

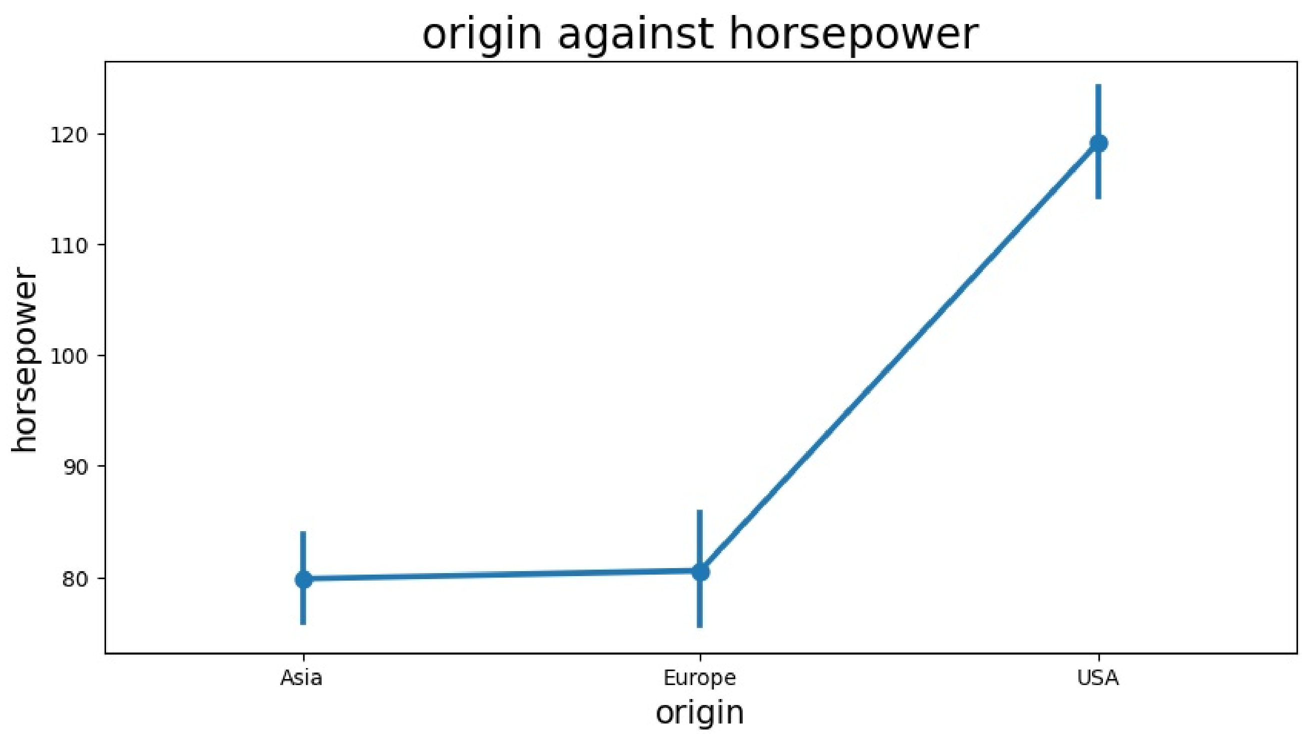
```python
# Plotting origin against mpg
plt.figure(figsize=(10, 5))
ax = sns.pointplot(x='origin', y='mpg', data=mpg_df, errorbar=None)

# Adding the numerical values to the points above
mean_mpg = mpg_df.groupby('origin', observed=False)['mpg'].mean()
for i, origin in enumerate(mean_mpg.index):
    ax.text(x=i, y=mean_mpg[origin] + 1,
            s=f"{mean_mpg[origin]:.1f}",
            ha='center', va='bottom', fontsize=10)

plt.title("Origin Against MPG", fontsize=20)
plt.xlabel("Origin", fontsize=15)
plt.ylabel("MPG", fontsize=15)
plt.show()
```



- Althogh `USA` has the biggest count of our dataset, it produces cars we relatively very low `mpg` compared to `Asia` and `Europe`
- `Asia` is the leading contry in producing cars with high `mpg` with a mean close to 30.

In [130…

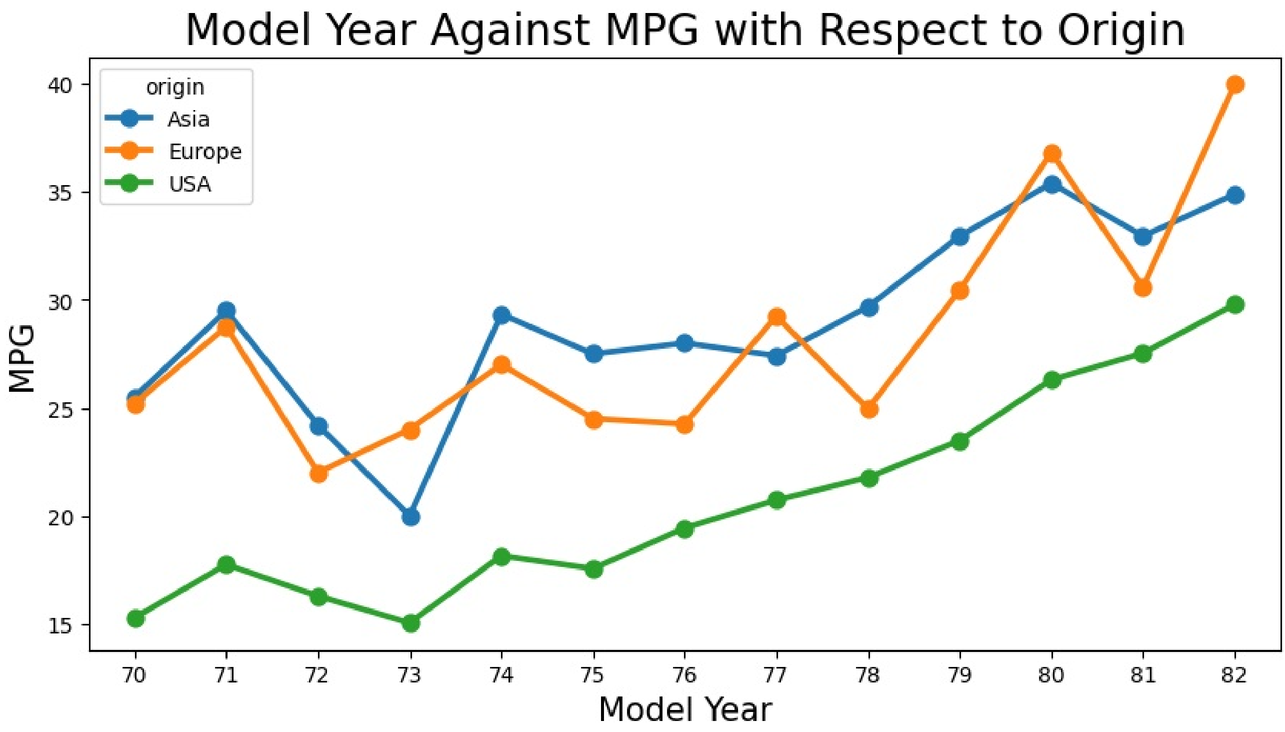
```
# ploting origin anainst horsepower
plt.figure(figsize=(10,5))
sns.pointplot(x = 'origin', y = 'horsepower', data = mpg_df)
plt.title("origin against horsepower", fontsize = 20)
plt.xlabel("origin", fontsize = 15)
plt.ylabel("horsepower", fontsize = 15)
plt.show()
```



- As expected, `USA` has the highest engine `horsepower`. This implies the previously observed conclusion that `hoursepower` and `mpg` has a negative correlation.

In [132…

```
# Display model year against mpg with respect to origin
plt.figure(figsize=(10, 5))
sns.pointplot(x='model year', y='mpg', hue='origin', data=mpg_df, errorbar=None)

plt.title("Model Year Against MPG with Respect to Origin", fontsize=20)
plt.xlabel("Model Year", fontsize=15)
plt.ylabel("MPG", fontsize=15)
plt.show()
```

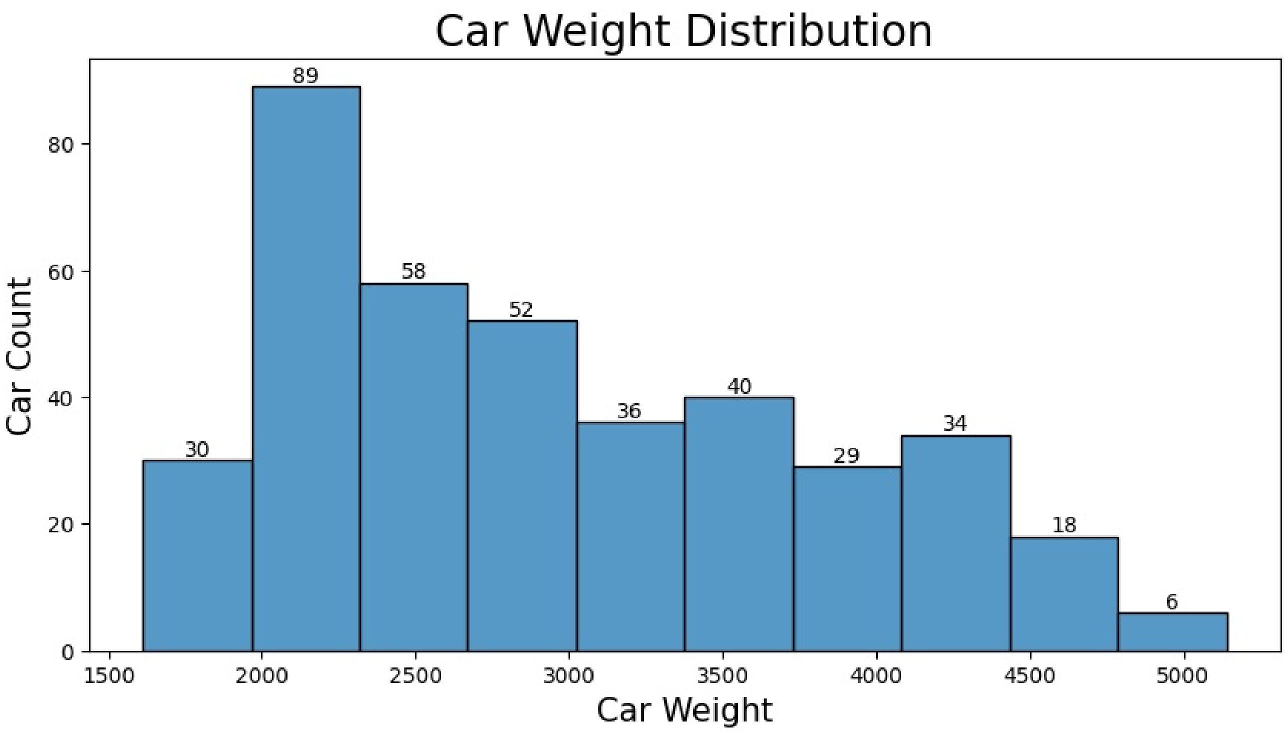


- This chart also implies the positive correlation between `model year` and `mpg`, and shows that `Asia` has been leading country in this industry, followed by `Europe` and `USA` comes in the last place.

Let's now explore how `weight` has been affecting our car specs over the years.

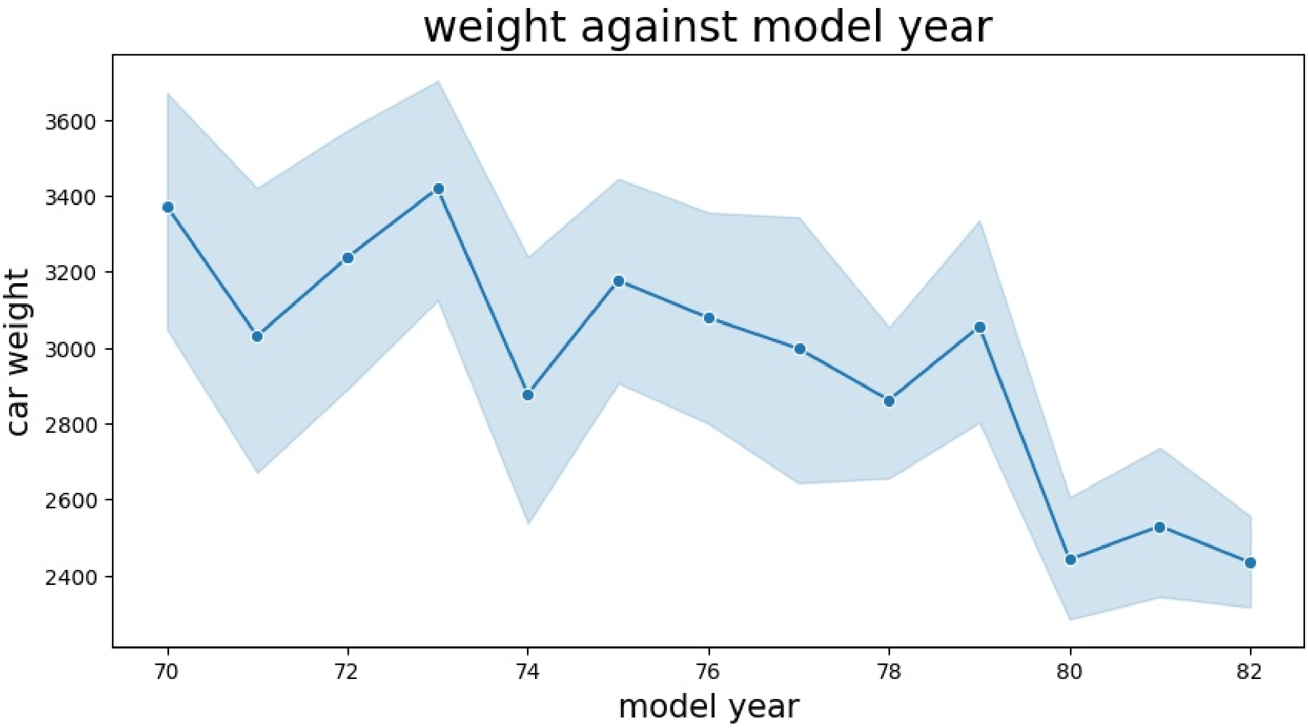
In [140…

```
# Plot a distribution of car weight
plt.figure(figsize=(10, 5))
ax = sns.histplot(x='weight', data=mpg_df, bins=10)

# Adding labels to each bin
for p in ax.patches:
    count = int(p.get_height())
    ax.text(p.get_x() + p.get_width() / 2, count,
            str(count),
            ha='center', va='bottom', fontsize=10)

plt.title("Car Weight Distribution", fontsize=20)
plt.xlabel("Car Weight", fontsize=15)
plt.ylabel("Car Count", fontsize=15)
plt.show()
```



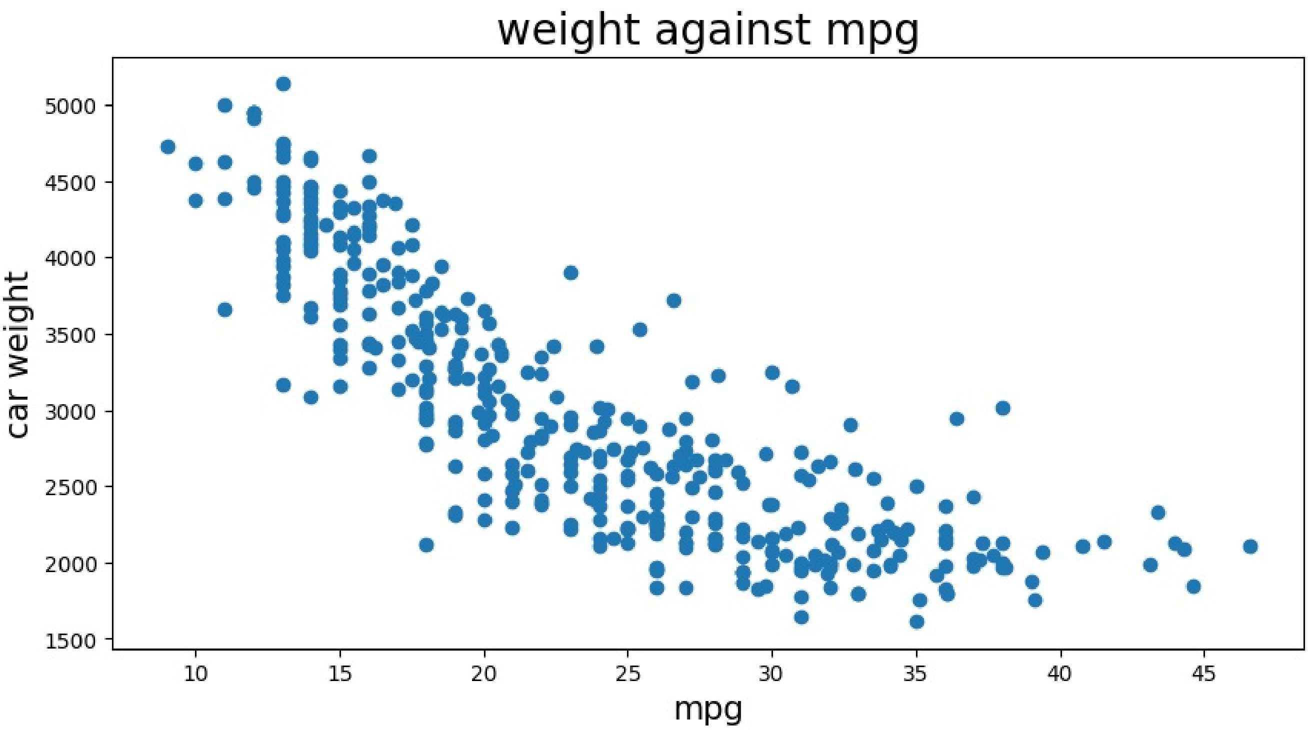
- car `weight` varies from 1500 to 5000, with the majority of cars at 2000.

In [145…
```
# plotting weight against model year
plt.figure(figsize=(10,5))
ax = sns.lineplot(x='model year', y='weight', data=mpg_df, marker='o')
plt.title("weight against model year", fontsize = 20)
plt.xlabel("model year", fontsize = 15)
plt.ylabel("car weight", fontsize = 15)
plt.show()
```



- Car `weight` has been decreasing over the years.

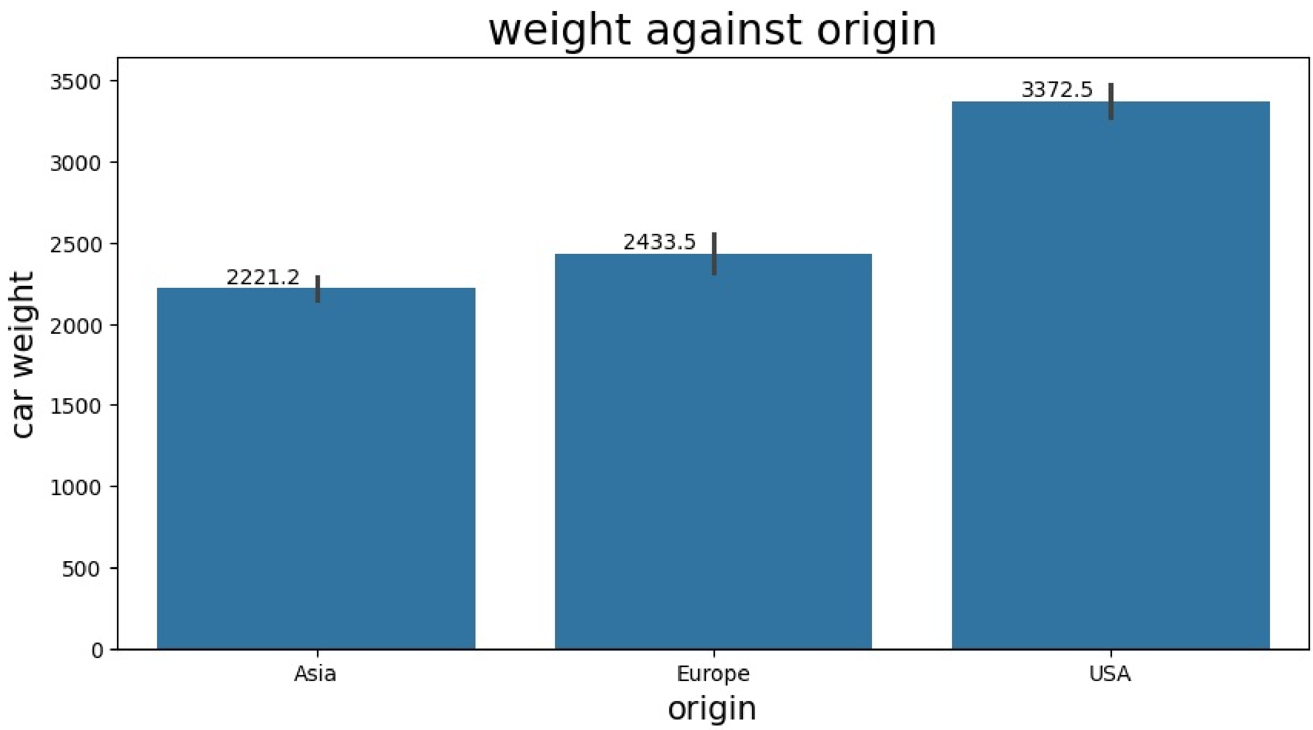
In [146…
```
# plotting weight against mpg
plt.figure(figsize=(10,5))
plt.scatter(x = 'mpg', y = 'weight', data = mpg_df)
plt.title("weight against mpg", fontsize = 20)
plt.xlabel("mpg", fontsize = 15)
plt.ylabel("car weight", fontsize = 15)
plt.show()
```



- `mpg` inceases as `weight` decreses over time, that indecates a strong correlation between them.

In [154…

```
# plotting weight against origin
plt.figure(figsize=(10,5))
ax = sns.barplot(x='origin', y='weight', data=mpg_df)

for p in ax.patches:
    ax.annotate(f'{p.get_height():.1f}',
                (p.get_x() + p.get_width() / 3., p.get_height()),
                ha='center', va='bottom', fontsize=10)

plt.title("weight against origin", fontsize = 20)
plt.xlabel("origin", fontsize = 15)
plt.ylabel("car weight", fontsize = 15)
plt.show()
```

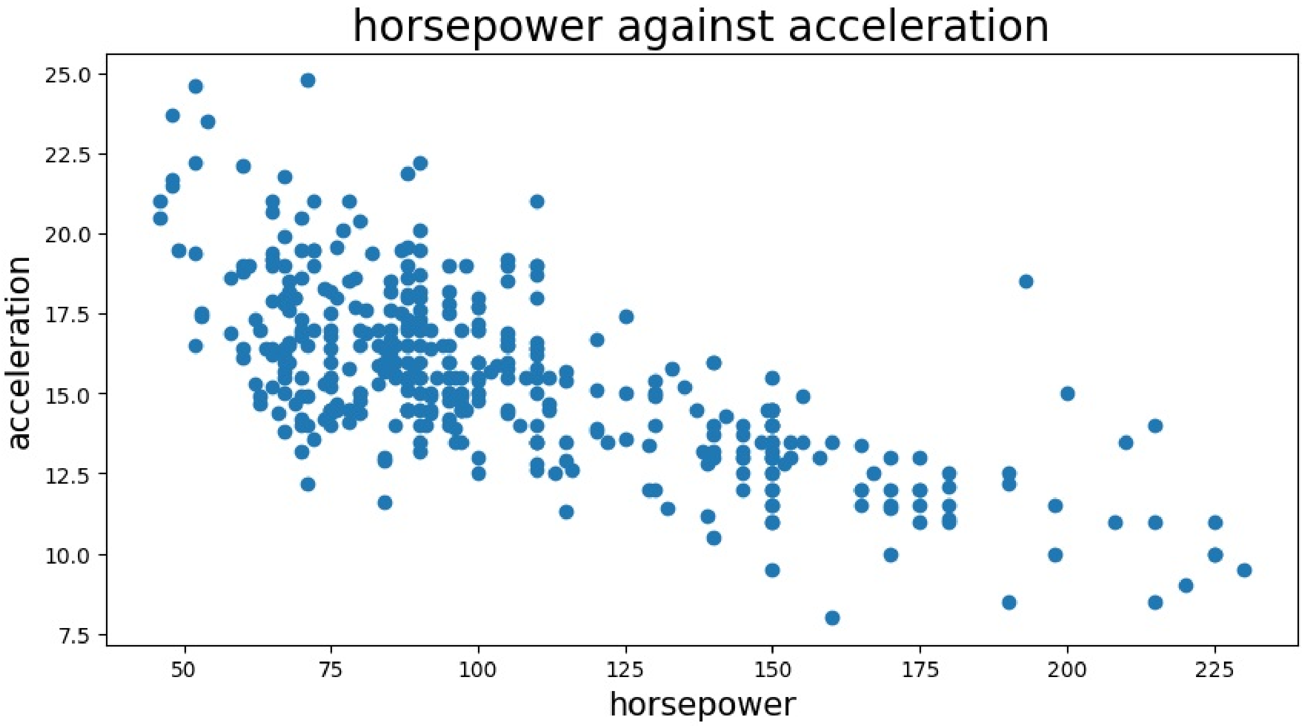


- As expected, `USA` gets the highest possible `weight` values, that explains alot about its low `mpg` and high `horsepower`.
- `Asia` produces the lightest cars campared to `USA` and `Europe`, this also explains itss high `mpg` and low `horsepower`.

Let's take a final look at `acceleration` column, and how it is related to `horsepower` and `mpg` to get a better understanding of our findings.

In [155…

```
# plotting horsepower against acceleration
plt.figure(figsize=(10,5))
plt.scatter(x = 'horsepower', y = 'acceleration', data = mpg_df)
plt.title("horsepower against acceleration", fontsize = 20)
plt.xlabel("horsepower", fontsize = 15)
plt.ylabel("acceleration", fontsize = 15)
plt.show()
```



- Wa can spot a negative correlation between `acceleration` and `horepower`, this means that it has a positive one with `mpg`.

Conclusion

In this section, we'd add the conclusions we draw from the previous visualisations.

- As years pass after `1973`, there has been a noticable increase in `mpg`.
- As `cylinders` in the engine increases above 4, `MPG` decreases and engine `horsepower` increases. That indicates negative correlation between `mpg` and `horsepower`.
- `mpg` increases as `weight` decreses over time, that also indecates a stron correlation between them.
- Althogh `USA` has the biggest count of produced cars, its cars has relatively very low `mpg`, thus the highest possible `weight` compared to `Asia` and `Europe`
- `Asia` is the leading contry in producing cars with high `mpg` with a mean close to 30, and it produces the lightest cars
- Wa can spot a negative correlation between `acceleration` and `horepower`, this means that it has a positive one with `mpg`.

> **"This project was entirely developed by** Bassam El-Shoraa**"**.