In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```python
dataset=pd.read_csv("iris.csv")
```

In [3]:
```python
dataset.head()
```

Out[3]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [4]:
```python
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Sepal_Length  150 non-null    float64
 1   Sepal_Width   150 non-null    float64
 2   Petal_Length  150 non-null    float64
 3   Petal_Width   150 non-null    float64
 4   Species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:
```python
dataset.describe()
```

Out[5]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [6]:
```python
dataset.isna().sum()
```

Out[6]:
```
Sepal_Length    0
Sepal_Width     0
Petal_Length    0
Petal_Width     0
Species         0
dtype: int64
```

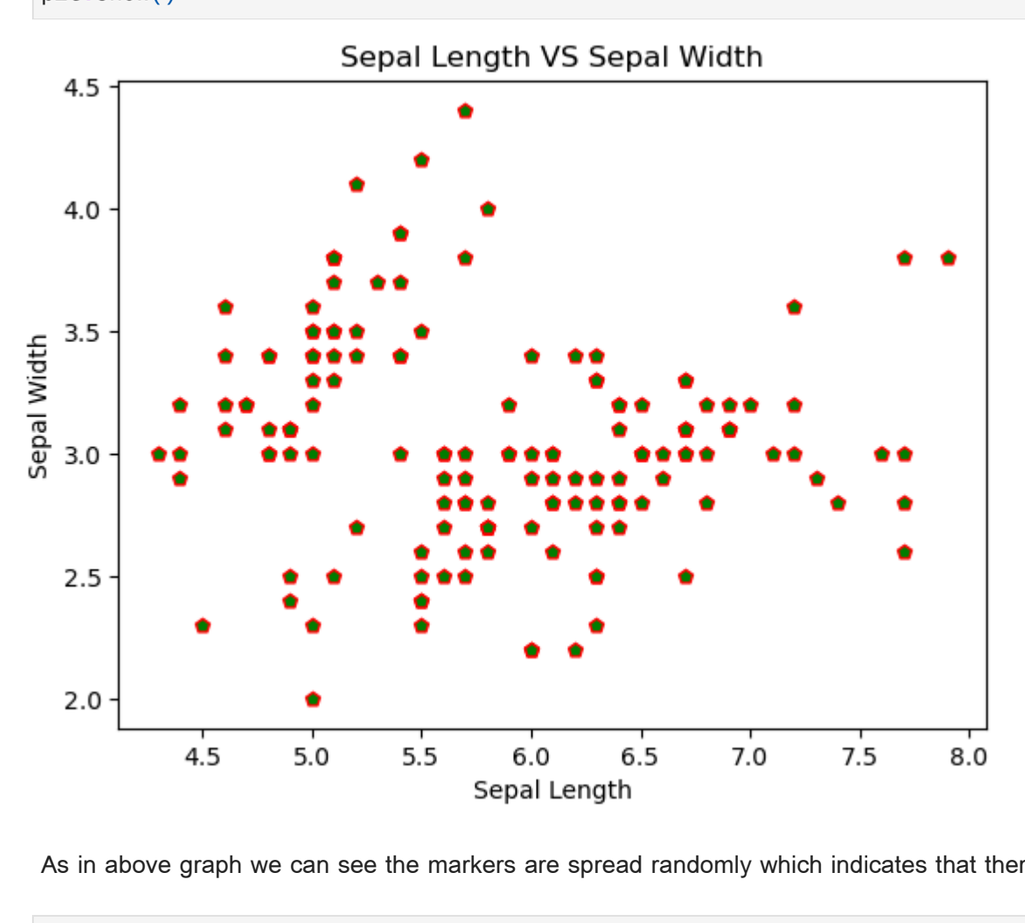
Scatter plot visualization

It is used to find the relationship between two numerical columns

types of relationship-

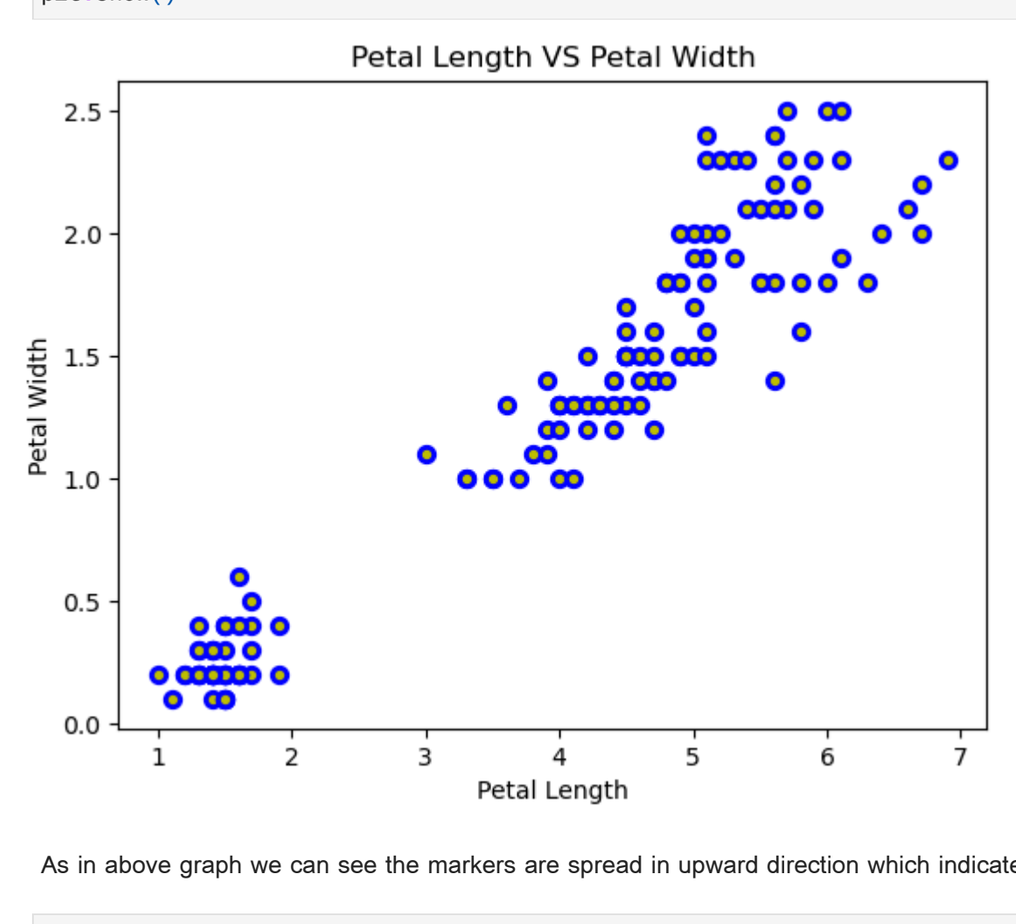
1. Positive relationship: Means Direct relationship between variables
2. Negative Reationship: Means inverse relationship between variables
3. Zero Relationship: There is no relation between those variables

In [7]:
```python
plt.scatter(dataset["Sepal_Length"],dataset["Sepal_Width"],marker="p" ,facecolor="g",edgecolor="r")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.title("Sepal Length VS Sepal Width")
plt.show()
```



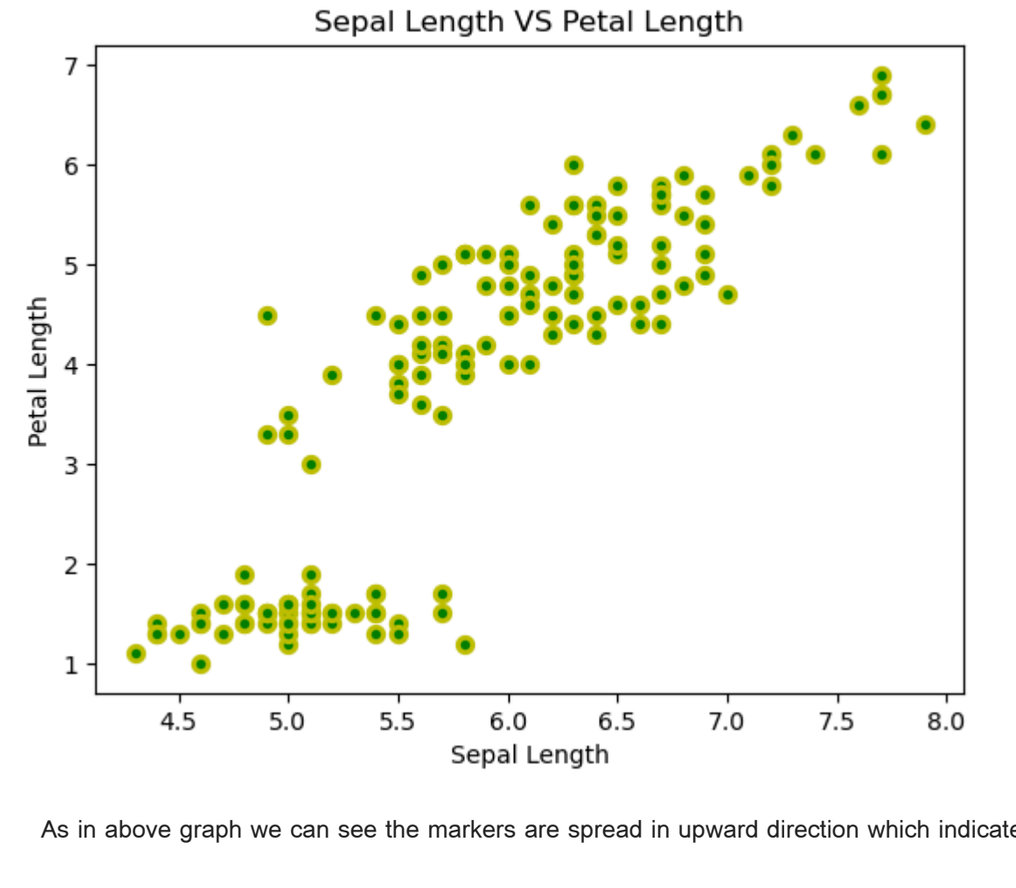
As in above graph we can see the markers are spread randomly which indicates that there is Zero relationship between Sepal_Length and Sepal_Width

In [8]:
```python
plt.scatter(dataset["Petal_Length"],dataset["Petal_Width"],facecolor="y",linewidths=2.1,edgecolor="b")
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.title("Petal Length VS Petal Width")
plt.show()
```



As in above graph we can see the markers are spread in upward direction which indicates that there is positive relationship between Petal_Length and Petal_Width. Which Means as the petal length increases petal width will also increases

In [9]:
```python
plt.scatter(dataset["Sepal_Length"],dataset["Petal_Length"],facecolor="g",linewidths=2.1,edgecolor="y")
plt.xlabel("Sepal Length")
plt.ylabel("Petal Length")
plt.title("Sepal Length VS Petal Length")
plt.show()
```



As in above graph we can see the markers are spread in upward direction which indicates that there is positive relationship between Sepal_Length and Petal Length. Which Means as the petal length increases Petal Length will also increases

Histogram Visualization

In [10]:
```python
carsdata=pd.read_csv("cars93.csv")
```

In [11]:
```python
carsdata.head()
```

Out[11]:

	Manufacturer	Model	Type	Price	MPG.city	AirBags	Horsepower	Passengers	Rear.seat.room	Luggage.room
0	Acura	Integra	Small	15.9	25	Driver only	140	5	26.5	11.0
1	Acura	Legend	Midsize	33.9	18	Driver & Passenger	200	5	30.0	15.0
2	Audi	90	Compact	29.1	20	Driver only	172	5	28.0	14.0
3	Audi	100	Midsize	37.7	19	Driver & Passenger	172	6	31.0	17.0
4	BMW	535i	Midsize	30.0	22	Driver only	208	4	27.0	13.0

In [12]:
```python
carsdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 93 entries, 0 to 92
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Manufacturer    93 non-null     object
 1   Model           93 non-null     object
 2   Type            93 non-null     object
 3   Price           93 non-null     float64
 4   MPG.city        93 non-null     int64
 5   AirBags         79 non-null     object
 6   Horsepower      93 non-null     int64
 7   Passengers      93 non-null     int64
 8   Rear.seat.room  91 non-null     float64
 9   Luggage.room    82 non-null     float64
dtypes: float64(3), int64(3), object(4)
memory usage: 7.4+ KB
```

As in above info we can see that- Out of all the columns some features are categorical in nature while some are numeric

In [13]:
```python
carsdata.isna().sum()
```

Out[13]:
```
Manufacturer       0
Model              0
Type               0
Price              0
MPG.city           0
AirBags           14
Horsepower         0
Passengers         0
Rear.seat.room     2
Luggage.room      11
dtype: int64
```

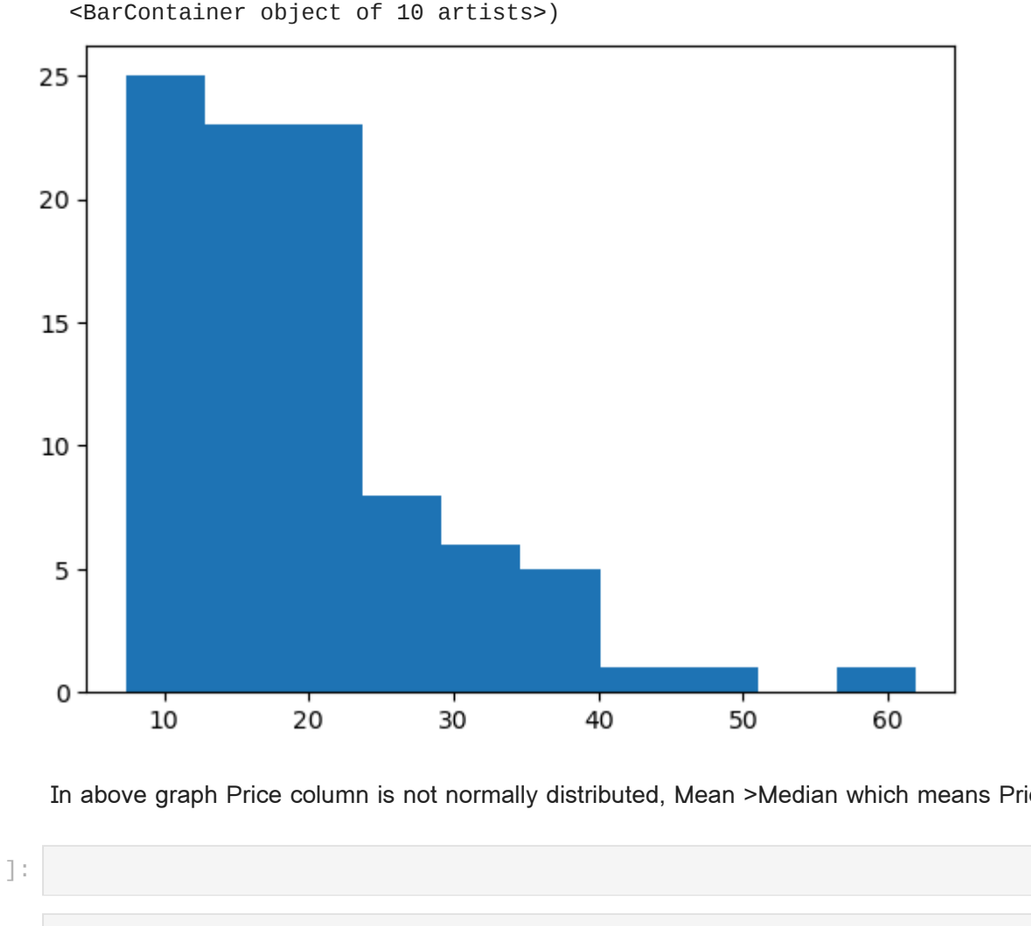
As we can see AirBags column is having 14 null values,Rear.seat.room is 2, and Luggage.room has 11 null values

In [14]:
```python
print("Size of the data is: ",carsdata.shape)
```

```
Size of the data is:  (93, 10)
```

In [15]:
```python
plt.hist(carsdata["Price"],histtype="barstacked")
```

Out[15]:
```
(array([25., 23., 23.,  8.,  6.,  5.,  1.,  1.,  0.,  1.]),
 array([ 7.4 , 12.85, 18.3 , 23.75, 29.2 , 34.65, 40.1 , 45.55, 51.  ,
        56.45, 61.9 ]),
 <BarContainer object of 10 artists>)
```

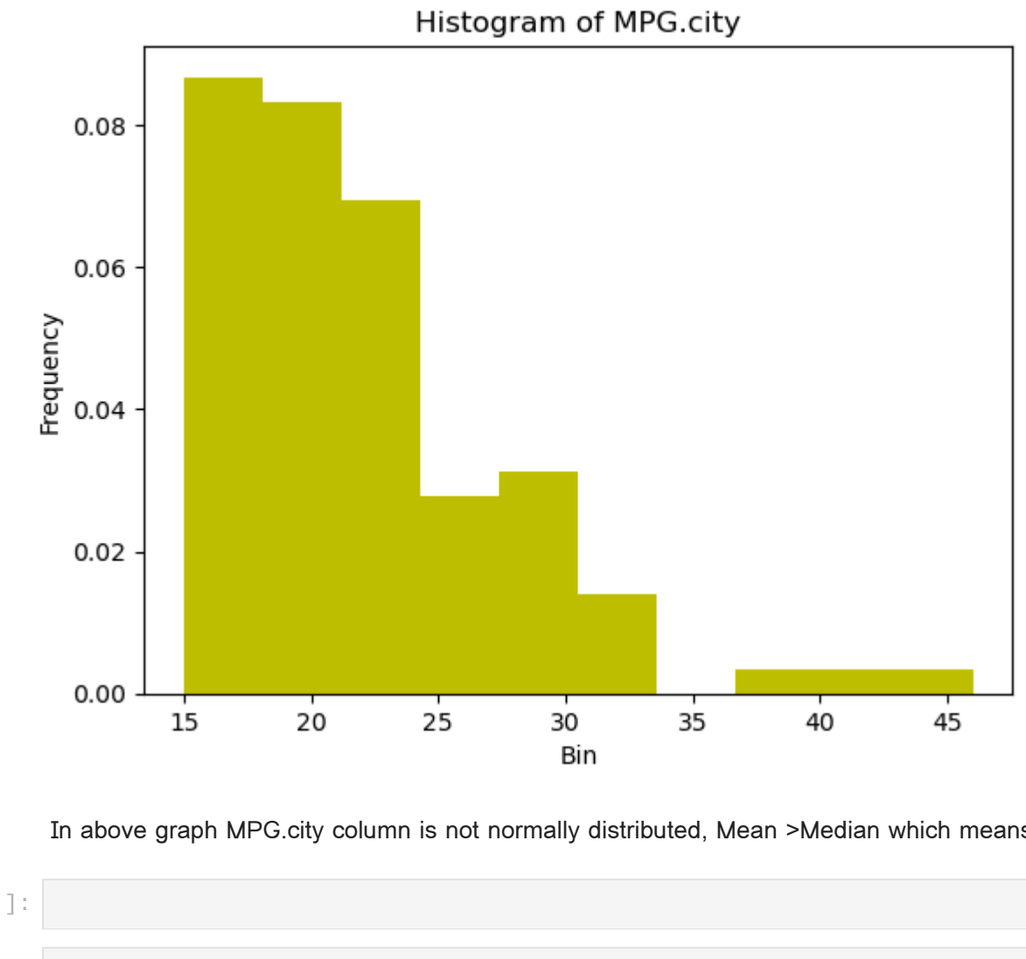


In above graph Price column is not normally distributed, Mean >Median which means Price column is right Skewed

In [ ]:

In [16]:
```python
plt.hist(carsdata["MPG.city"],density=True,color="y")
plt.xlabel("Bin")
plt.ylabel("Frequency")
plt.title("Histogram of MPG.city")
```

Out[16]:
```
Text(0.5, 1.0, 'Histogram of MPG.city')
```

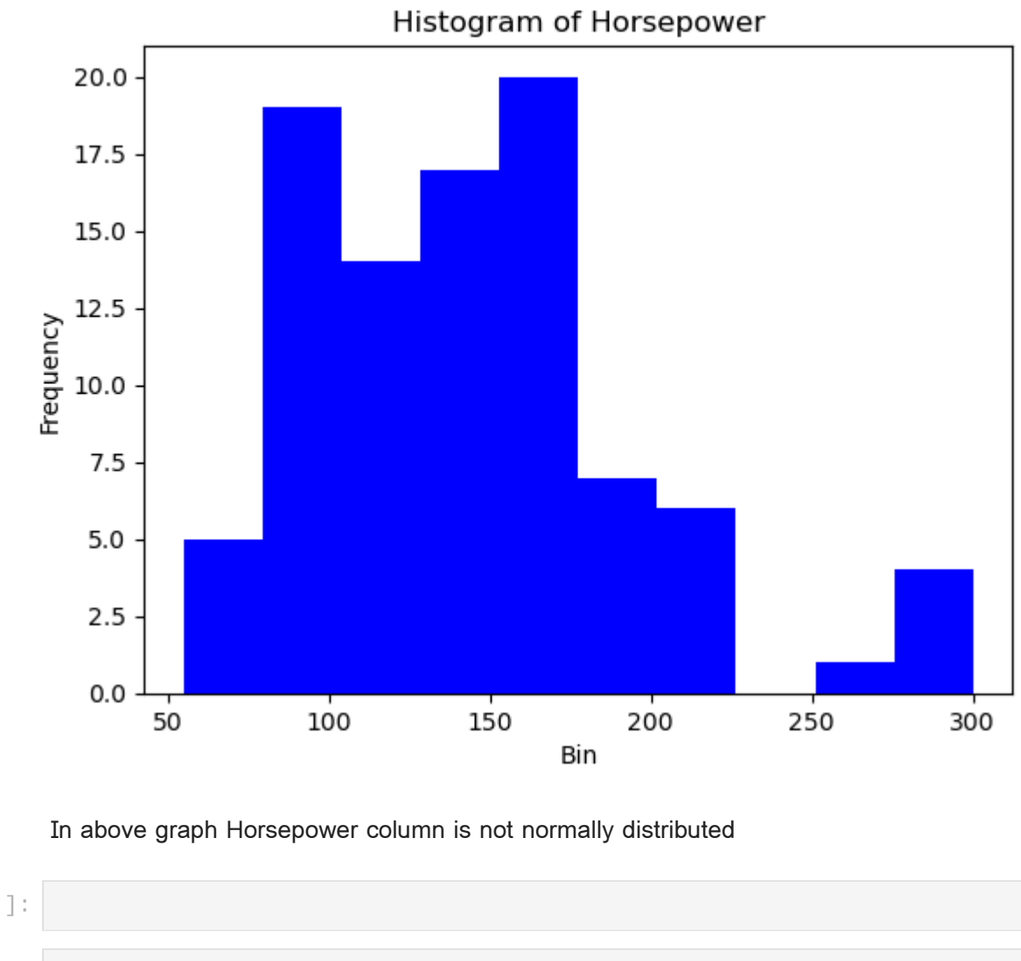


In above graph MPG.city column is not normally distributed, Mean >Median which means Price column is right Skewed

In [ ]:

In [17]:
```python
plt.hist(carsdata["Horsepower"],color="b")
plt.xlabel("Bin")
plt.ylabel("Frequency")
plt.title("Histogram of Horsepower")
```

Out[17]:
```
Text(0.5, 1.0, 'Histogram of Horsepower')
```

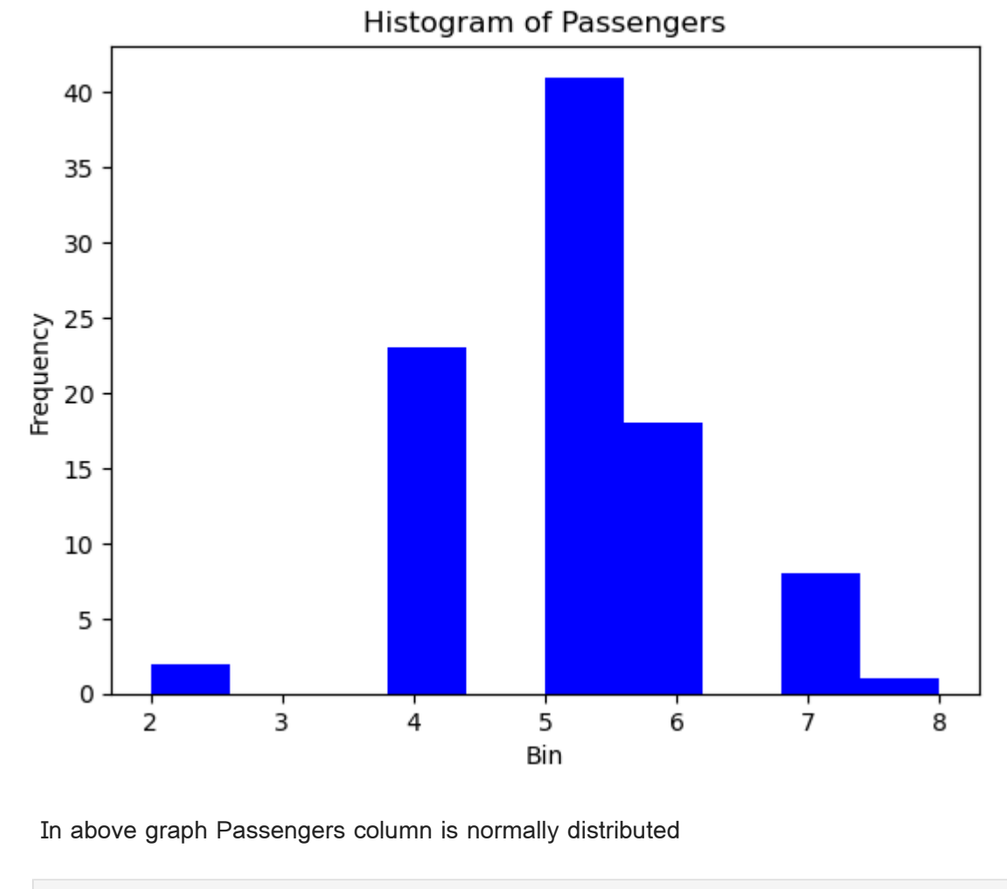


In above graph Horsepower column is not normally distributed

In [ ]:

In [18]:
```python
plt.hist(carsdata["Passengers"],color="b")
plt.xlabel("Bin")
plt.ylabel("Frequency")
plt.title("Histogram of Passengers")
```

Out[18]:
```
Text(0.5, 1.0, 'Histogram of Passengers')
```



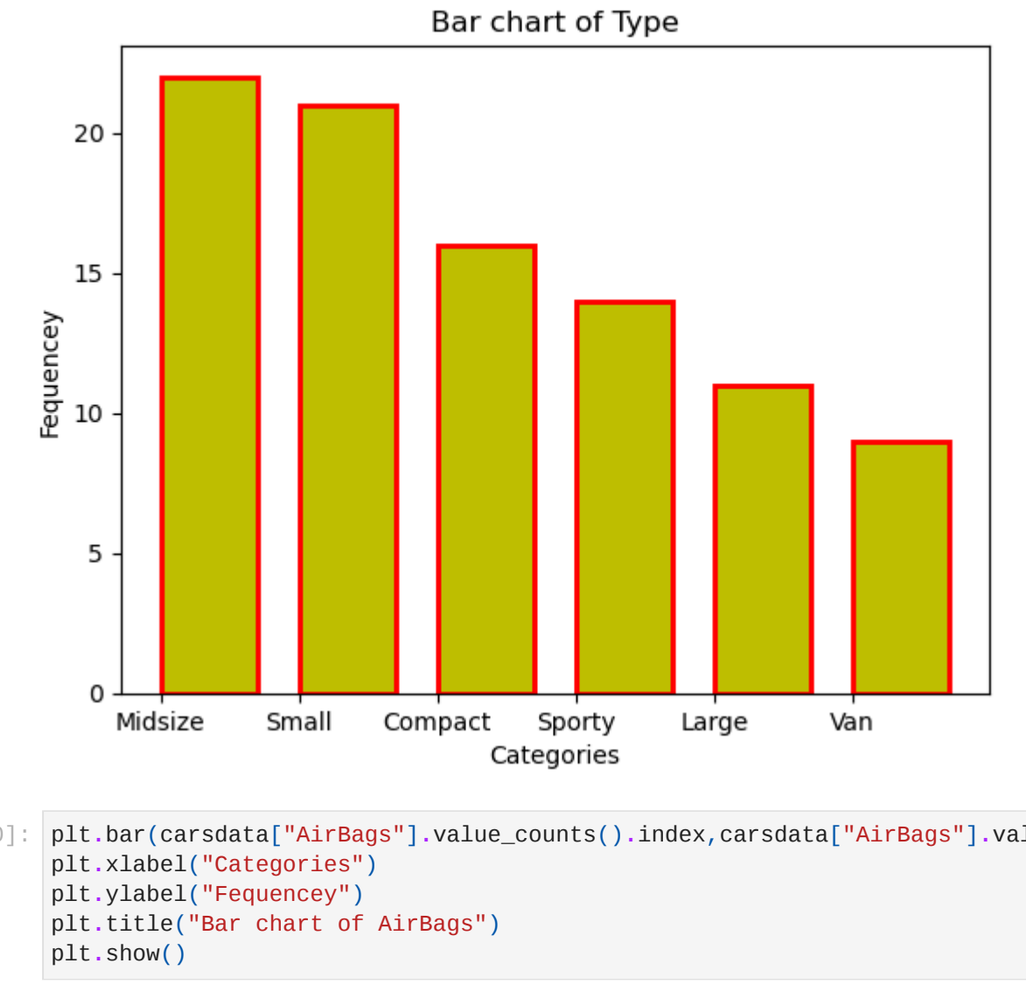
In above graph Passengers column is normally distributed

In [ ]:

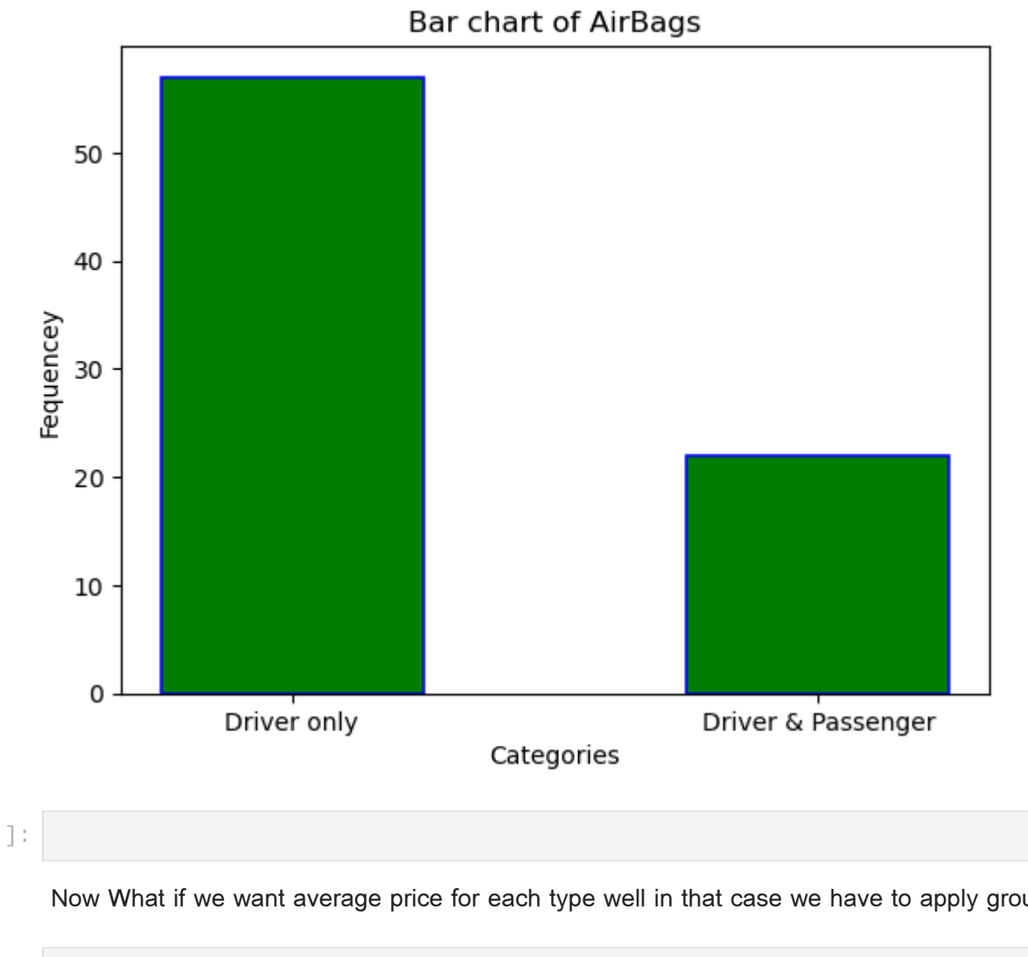
In [ ]:

Bar Chart Visualization (For Categorical Columns)

In [19]:
```python
plt.bar(carsdata["Type"].value_counts().index,carsdata["Type"].value_counts().values,width=0.7,color="y",align="edge",edgecolor="r",
        ecolor="g",linewidth=2.1)
plt.xlabel("Categories")
plt.ylabel("Fequency")
plt.title("Bar chart of Type")
plt.show()
```



In [20]:
```python
plt.bar(carsdata["AirBags"].value_counts().index,carsdata["AirBags"].value_counts().values,width=0.5,color="g",align="center",edgecolor="b",linewidth=1.1)
plt.xlabel("Categories")
plt.ylabel("Fequency")
plt.title("Bar chart of AirBags")
plt.show()
```



In [ ]:

Now What if we want average price for each type well in that case we have to apply groupby

In [21]:
```python
df=carsdata[["Price","Type"]]
df
```

Out[21]:

	Price	Type
0	15.9	Small
1	33.9	Midsize
2	29.1	Compact
3	37.7	Midsize
4	30.0	Midsize
...	...	...
88	19.7	Van
89	20.0	Compact
90	23.3	Sporty
91	22.7	Compact
92	26.7	Midsize

93 rows × 2 columns

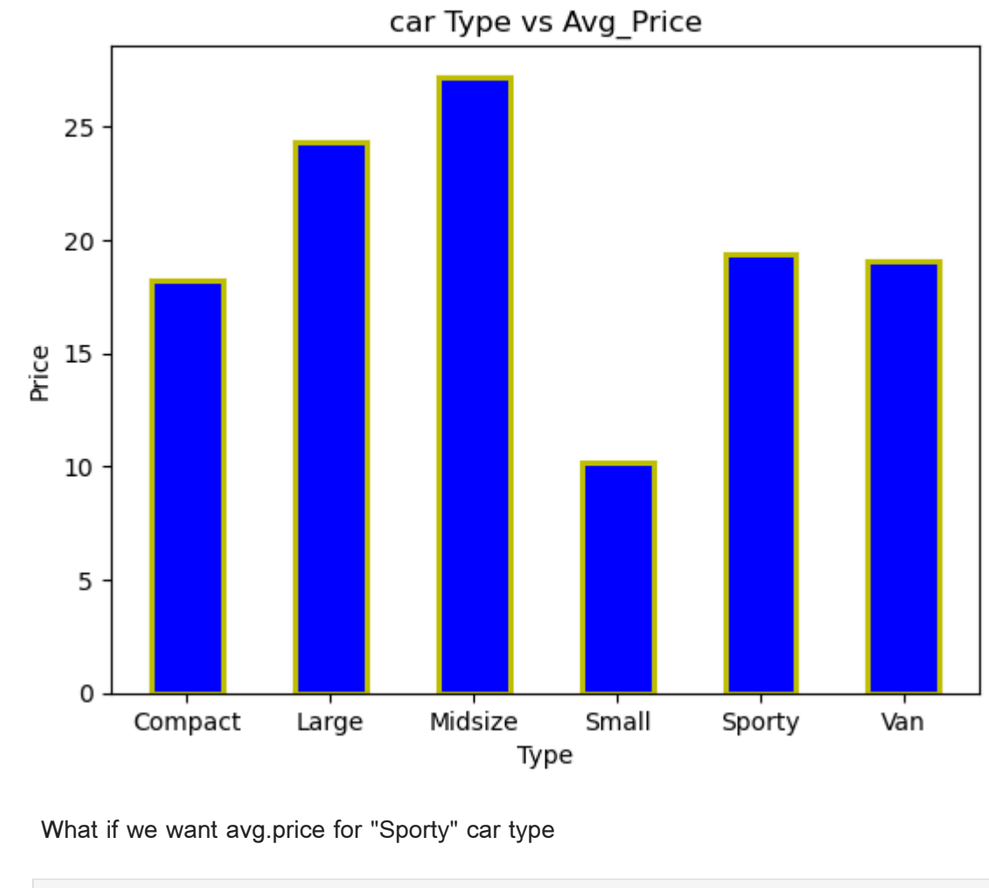
In [22]:
```python
grouped_df=df.groupby("Type").mean()
```

In [41]:
```python
grouped_df
```

Out[41]:

Type	Price
Compact	18.212500
Large	24.300000
Midsize	27.218182
Small	10.166667
Sporty	19.392857
Van	19.100000

In [42]:
```python
plt.bar(grouped_df.index,grouped_df["Price"],color="b",edgecolor="y",width=0.5,linewidth=2.2)
plt.xlabel("Type")
plt.ylabel("Price")
plt.title("car Type vs Avg_Price")
plt.show()
```

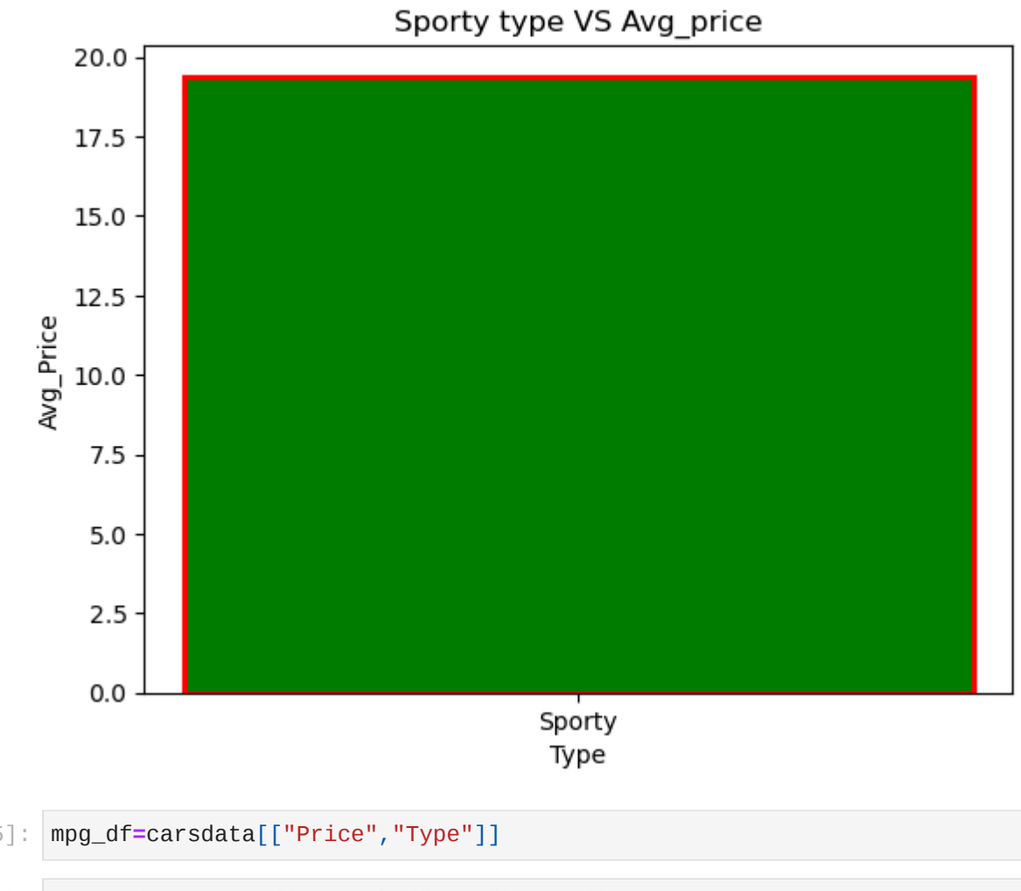


What if we want avg.price for "Sporty" car type

In [46]:
```python
sporty=grouped_df[4:5]
```

In [64]:
```python
plt.bar(sporty.index,sporty["Price"],width=0.1,color="g",edgecolor="r",linewidth=2.2)
plt.xlabel("Type")
plt.ylabel("Avg_Price")
plt.title("Sporty type VS Avg_price")
```

Out[64]:
```
Text(0.5, 1.0, 'Sporty type VS Avg_price')
```



In [65]:
```python
mpg_df=carsdata[["Price","Type"]]
```

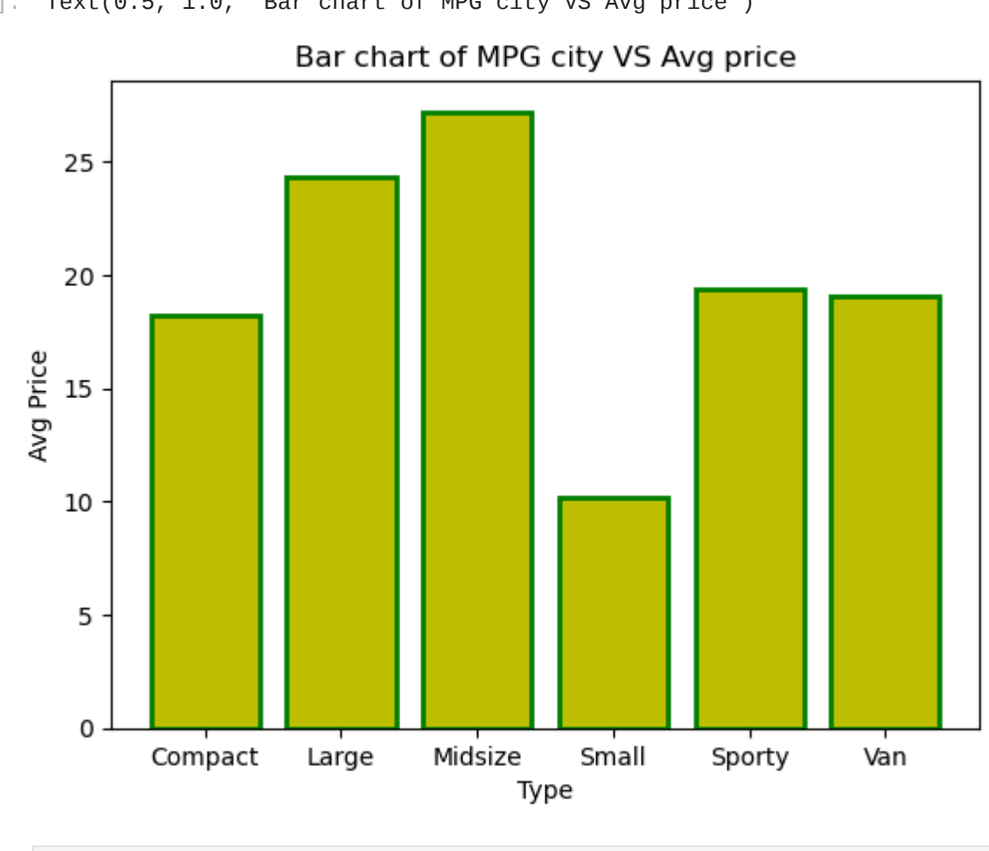
In [75]:
```python
grouped_mpg=mpg_df.groupby("Type").mean()
grouped_mpg
```

Out[75]:

Type	Price
Compact	18.212500
Large	24.300000
Midsize	27.218182
Small	10.166667
Sporty	19.392857
Van	19.100000

In [74]:
```python
plt.bar(grouped_mpg.index,grouped_mpg["Price"],color="y",width=0.8,edgecolor="g",linewidth=2)
plt.xlabel("Type")
plt.ylabel("Avg Price")
plt.title("Bar chart of MPG city VS Avg price")
```

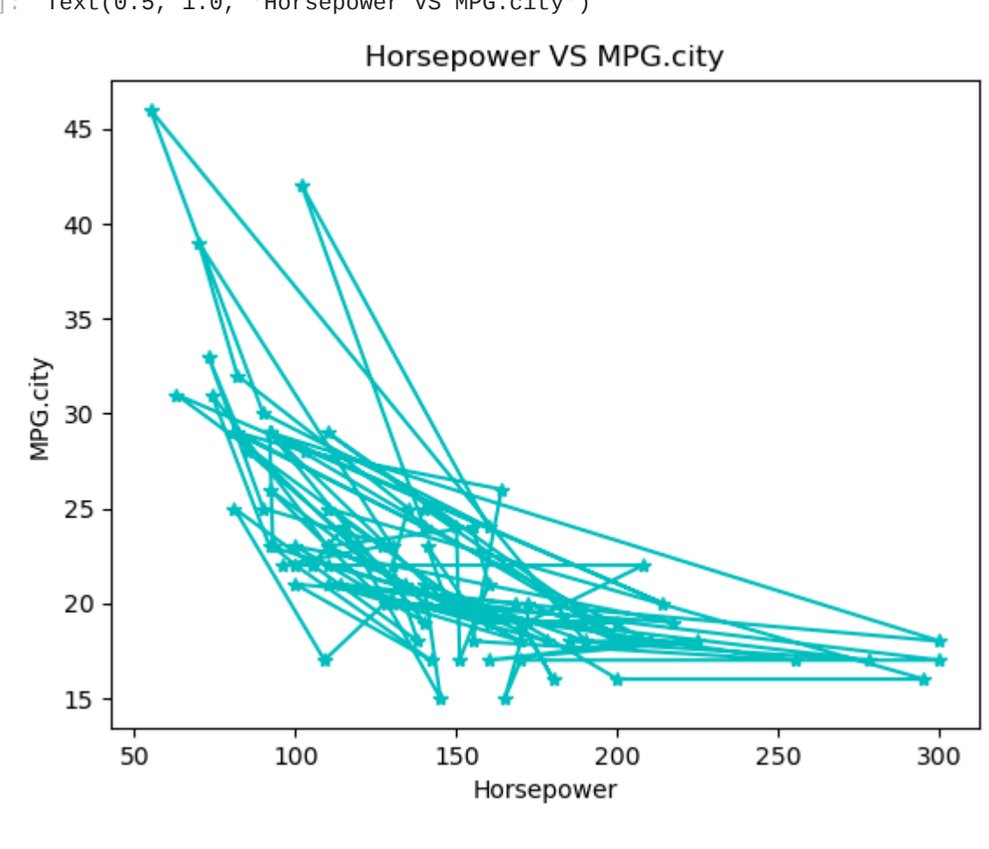
Out[74]:
```
Text(0.5, 1.0, 'Bar chart of MPG city VS Avg price')
```



In [ ]:

In [81]:
```python
plt.plot(carsdata["Horsepower"],carsdata["MPG.city"],color="c",marker="*")
plt.xlabel("Horsepower")
plt.ylabel("MPG.city")
plt.title("Horsepower VS MPG.city")
```

Out[81]:
```
Text(0.5, 1.0, 'Horsepower VS MPG.city')
```



As We can see in above graph,there is no clear picture of pattern this is because MPG.cty are not sorted for Horsepower so we have to sort them as follows-

In [85]:
```python
df3=carsdata[["Horsepower","MPG.city"]]
```

In [95]:
```python
sorted_df=df3.sort_values(by="Horsepower",ascending=True)
```

In [96]:
```python
plt.plot(sorted_df,sorted_df["MPG.city"],color="c",marker="*")
plt.xlabel("Sorted Horsepower")
plt.ylabel("MPG.city")
plt.title("Sorted Horsepower VS MPG.city")
```

Out[96]:
```
Text(0.5, 1.0, 'Sorted Horsepower VS MPG.city')
```