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

1. Read Dataset

In [2]:
```
A=pd.read_csv("C:/Users/ASUS/OneDrive/Desktop/Datasets in python/Cars93.csv")
```

In [3]:
```
A
```

Out[3]:

	id	Manufacturer	Model	Type	MinPrice	Price	MaxPrice	MPGcity	MPGhighway	AirBags	...	Pass
0	1	Acura	Integra	Small	12.9	15.9	18.8	25	31	NaN	...	
1	2	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger	...	
2	3	Audi	90	Compact	25.9	29.1	32.3	20	26	Driver only	...	
3	4	Audi	100	Midsize	30.8	37.7	44.6	19	26	NaN	...	
4	5	BMW	535i	Midsize	23.7	30.0	36.2	22	30	Driver only	...	
...	...	...	...	...	...	...	...	...	...	...	...	
88	89	Volkswagen	Eurovan	Van	16.6	19.7	22.7	17	21	NaN	...	
89	90	Volkswagen	Passat	Compact	17.6	20.0	22.4	21	30	NaN	...	
90	91	Volkswagen	Corrado	Sporty	22.9	23.3	23.7	18	25	NaN	...	
91	92	Volvo	240	Compact	21.8	22.7	23.5	21	28	Driver only	...	
92	93	Volvo	850	Midsize	24.8	26.7	28.5	20	28	Driver & Passenger	...	

93 rows × 28 columns

1. Describe DataSet

In [4]:
```
A.describe()
```

Out[4]:

	id	MinPrice	Price	MaxPrice	MPGcity	MPGhighway	EngineSize	Horsepower	RI
count	93.000000	93.000000	93.000000	93.000000	93.000000	93.000000	93.000000	93.000000	93.0000
mean	47.000000	17.125806	19.509677	21.898925	22.365591	29.086022	2.667742	143.827957	5280.6451
std	26.990739	8.746029	9.659430	11.030457	5.619812	5.331726	1.037363	52.374410	596.7316
min	1.000000	6.700000	7.400000	7.900000	15.000000	20.000000	1.000000	55.000000	3800.0000
25%	24.000000	10.800000	12.200000	14.700000	18.000000	26.000000	1.800000	103.000000	4800.0000
50%	47.000000	14.700000	17.700000	19.600000	21.000000	28.000000	2.400000	140.000000	5200.0000
75%	70.000000	20.300000	23.300000	25.300000	25.000000	31.000000	3.300000	170.000000	5750.0000
max	93.000000	45.400000	61.900000	80.000000	46.000000	50.000000	5.700000	300.000000	6500.0000

1. Structure of DataSet

In [7]:
```
A.shape
```

Out[7]:
```
(93, 28)
```

In [8]:
```
A.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 93 entries, 0 to 92
Data columns (total 28 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   id                93 non-null     int64  
 1   Manufacturer      93 non-null     object 
 2   Model             93 non-null     object 
 3   Type              93 non-null     object 
 4   MinPrice          93 non-null     float64
 5   Price             93 non-null     float64
 6   MaxPrice          93 non-null     float64
 7   MPGcity           93 non-null     int64  
 8   MPGhighway        93 non-null     int64  
 9   AirBags           56 non-null     object 
 10  DriveTrain        93 non-null     object 
 11  Cylinders         93 non-null     object 
 12  EngineSize        93 non-null     float64
 13  Horsepower        93 non-null     int64  
 14  RPM               93 non-null     int64  
 15  Rev.per.mile      93 non-null     int64  
 16  Man.trans.avail   93 non-null     object 
 17  Fueltankcapacity  93 non-null     float64
 18  Passengers        93 non-null     int64  
 19  Length            93 non-null     int64  
 20  Wheelbase         93 non-null     int64  
 21  Width             93 non-null     int64  
 22  Turn.circle       93 non-null     int64  
 23  Rear.seat.room    91 non-null     float64
 24  Luggage.room      82 non-null     float64
 25  Weight            93 non-null     int64  
 26  Origin            93 non-null     object 
 27  Make              93 non-null     object 
dtypes: float64(7), int64(12), object(9)
memory usage: 20.5+ KB
```

In [6]:
```
A.head()
```

Out[6]:

	id	Manufacturer	Model	Type	MinPrice	Price	MaxPrice	MPGcity	MPGhighway	AirBags	...	Passer
0	1	Acura	Integra	Small	12.9	15.9	18.8	25	31	NaN	...	
1	2	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger	...	
2	3	Audi	90	Compact	25.9	29.1	32.3	20	26	Driver only	...	
3	4	Audi	100	Midsize	30.8	37.7	44.6	19	26	NaN	...	
4	5	BMW	535i	Midsize	23.7	30.0	36.2	22	30	Driver only	...	

5 rows × 28 columns

In [6]:
```
from preprocessing import replacer
replacer(A)
```

1. Missing value Treatment

In [7]:
```
A.isnull().sum()
```

Out[7]:
```
id                 0
Manufacturer       0
Model              0
Type               0
MinPrice           0
Price              0
MaxPrice           0
MPGcity            0
MPGhighway         0
AirBags            0
DriveTrain         0
Cylinders          0
EngineSize         0
Horsepower         0
RPM                0
Rev.per.mile       0
Man.trans.avail    0
Fueltankcapacity   0
Passengers         0
Length             0
Wheelbase          0
Width              0
Turn.circle        0
Rear.seat.room     0
Luggage.room       0
Weight             0
Origin             0
Make               0
dtype: int64
```

8.Correlation Martics using HeatMap

In [8]:
```
numeric_columns = A.select_dtypes(include=['int', 'float']).columns
numeric_df = A[numeric_columns]
numeric_df
```

Out[8]:

	id	MinPrice	Price	MaxPrice	MPGcity	MPGhighway	EngineSize	Horsepower	RPM	Rev.per.mile	Fueltanl
0	1	12.9	15.9	18.8	25	31	1.8	140	6300	2890	
1	2	29.2	33.9	38.7	18	25	3.2	200	5500	2335	
2	3	25.9	29.1	32.3	20	26	2.8	172	5500	2280	
3	4	30.8	37.7	44.6	19	26	2.8	172	5500	2535	
4	5	23.7	30.0	36.2	22	30	3.5	208	5700	2545	
...	...	...	...	...	...	...	...	...	...	...	
88	89	16.6	19.7	22.7	17	21	2.5	109	4500	2915	
89	90	17.6	20.0	22.4	21	30	2.0	134	5800	2685	
90	91	22.9	23.3	23.7	18	25	2.8	178	5800	2385	
91	92	21.8	22.7	23.5	21	28	2.3	114	5400	2215	
92	93	24.8	26.7	28.5	20	28	2.4	168	6200	2310	

93 rows × 19 columns

In [9]:
```
x=A.drop(["RPM","Cylinders","DriveTrain","EngineSize","Luggage.room","Make","Wheelbase",
```

In [10]:
```
x
```

Out[10]:

	id	Manufacturer	Model	Type	MinPrice	Price	MaxPrice	MPGcity	MPGhighway	AirBags	Horsepo
0	1	Acura	Integra	Small	12.9	15.9	18.8	25	31	Driver only	
1	2	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger	
2	3	Audi	90	Compact	25.9	29.1	32.3	20	26	Driver only	
3	4	Audi	100	Midsize	30.8	37.7	44.6	19	26	Driver only	
4	5	BMW	535i	Midsize	23.7	30.0	36.2	22	30	Driver only	
...	...	...	...	...	...	...	...	...	...	...	
88	89	Volkswagen	Eurovan	Van	16.6	19.7	22.7	17	21	Driver only	
89	90	Volkswagen	Passat	Compact	17.6	20.0	22.4	21	30	Driver only	
90	91	Volkswagen	Corrado	Sporty	22.9	23.3	23.7	18	25	Driver only	
91	92	Volvo	240	Compact	21.8	22.7	23.5	21	28	Driver only	
92	93	Volvo	850	Midsize	24.8	26.7	28.5	20	28	Driver & Passenger	

93 rows × 13 columns

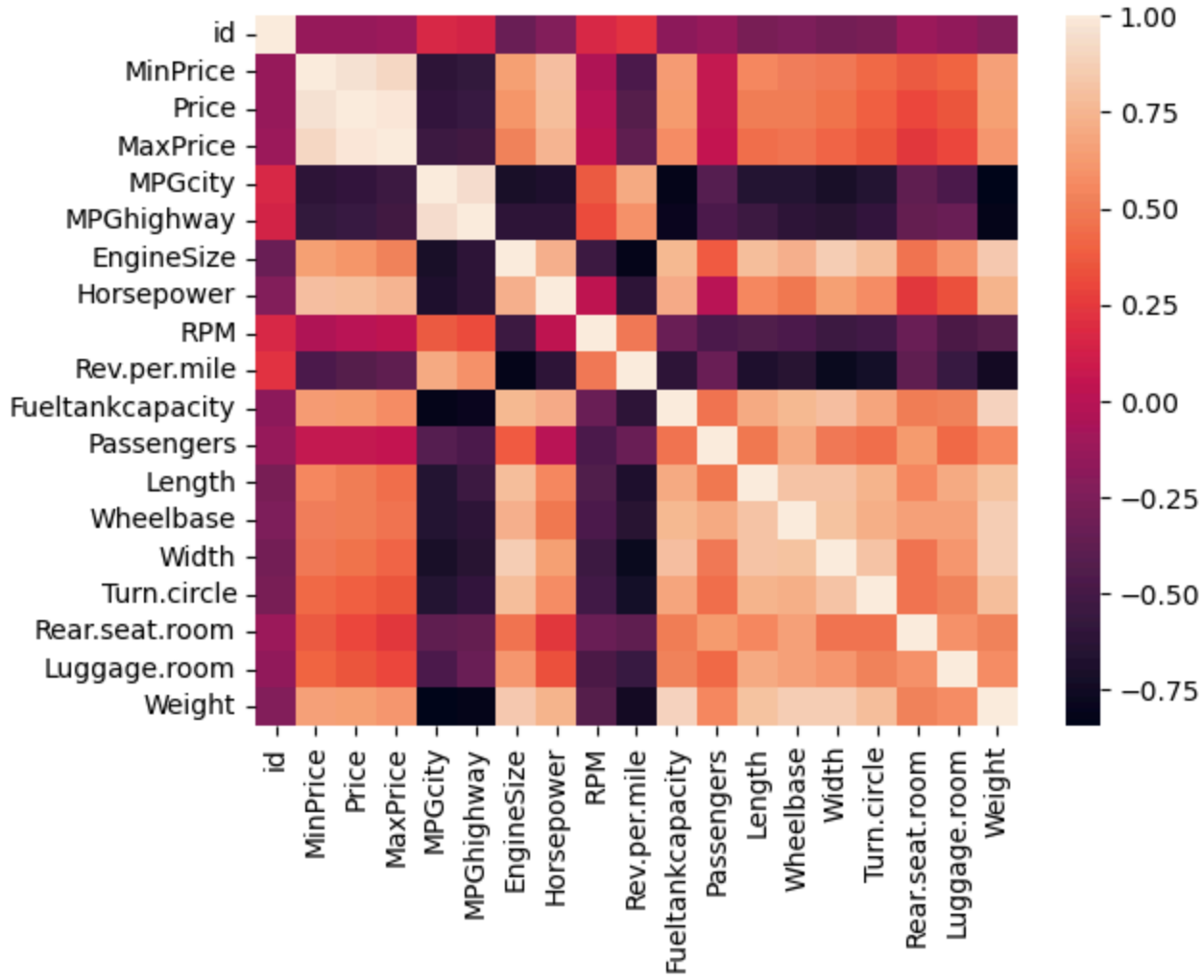
In [11]:
```
correlation_matrix = numeric_df.corr()
correlation_matrix
```

Out[11]:

	id	MinPrice	Price	MaxPrice	MPGcity	MPGhighway	EngineSize	Horsepower	
id	1.000000	-0.144463	-0.137277	-0.126567	0.161808	0.137317	-0.338286	-0.233012	
MinPrice	-0.144463	1.000000	0.970601	0.906756	-0.622875	-0.579966	0.645488	0.802444	-
Price	-0.137277	0.970601	1.000000	0.981580	-0.594562	-0.560680	0.597425	0.788218	-
MaxPrice	-0.126567	0.906756	0.981580	1.000000	-0.547811	-0.522561	0.535012	0.744445	
MPGcity	0.161808	-0.622875	-0.594562	-0.547811	1.000000	0.943936	-0.710003	-0.672636	
MPGhighway	0.137317	-0.579966	-0.560680	-0.522561	0.943936	1.000000	-0.626795	-0.619044	
EngineSize	-0.338286	0.645488	0.597425	0.535012	-0.710003	-0.626795	1.000000	0.732120	-
Horsepower	-0.233012	0.802444	0.788218	0.744445	-0.672636	-0.619044	0.732120	1.000000	
RPM	0.167772	-0.042598	-0.004955	0.025015	0.363045	0.313469	-0.547898	0.036688	
Rev.per.mile	0.226389	-0.470395	-0.426395	-0.374024	0.695857	0.587497	-0.824009	-0.600314	
Fueltankcapacity	-0.179034	0.635369	0.619480	0.581294	-0.813144	-0.786039	0.759306	0.711790	-
Passengers	-0.142252	0.061236	0.057860	0.053216	-0.416856	-0.466386	0.372721	0.009264	-
Length	-0.277911	0.553859	0.503628	0.442933	-0.666239	-0.542897	0.780283	0.550865	-
Wheelbase	-0.244888	0.516758	0.500864	0.467501	-0.667108	-0.615384	0.732484	0.486854	-
Width	-0.293592	0.492878	0.456028	0.408414	-0.720534	-0.640359	0.867110	0.644413	-
Turn.circle	-0.275867	0.428603	0.392590	0.347785	-0.666389	-0.593683	0.778464	0.561216	-
Rear.seat.room	-0.126129	0.361525	0.301888	0.241600	-0.380435	-0.364284	0.473740	0.236871	-
Luggage.room	-0.161114	0.395783	0.354635	0.307020	-0.462204	-0.327732	0.618326	0.328568	-
Weight	-0.220525	0.666554	0.647179	0.605142	-0.843139	-0.810658	0.845075	0.738798	-

In [12]:
```
sb.heatmap(correlation_matrix)
```

Out[12]:
```
<Axes: >
```



1. Y axis=Origin

In [13]:
```
y=A['Origin']
y
```

Out[13]:
```
0     non-USA
1     non-USA
2     non-USA
3     non-USA
4     non-USA
       ...
88    non-USA
89    non-USA
90    non-USA
91    non-USA
92    non-USA
Name: Origin, Length: 93, dtype: object
```

1. Divide data set into cat and con

In [14]:
```
cat=[]
con=[]
for i in x.columns:
    if(x[i].dtypes=="object"):
        cat.append(i)
    else:
        con.append(i)
```

In [15]:
```
cat
```

Out[15]:
```
['Manufacturer', 'Model', 'Type', 'AirBags', 'Origin']
```

In [16]:
```
con
```

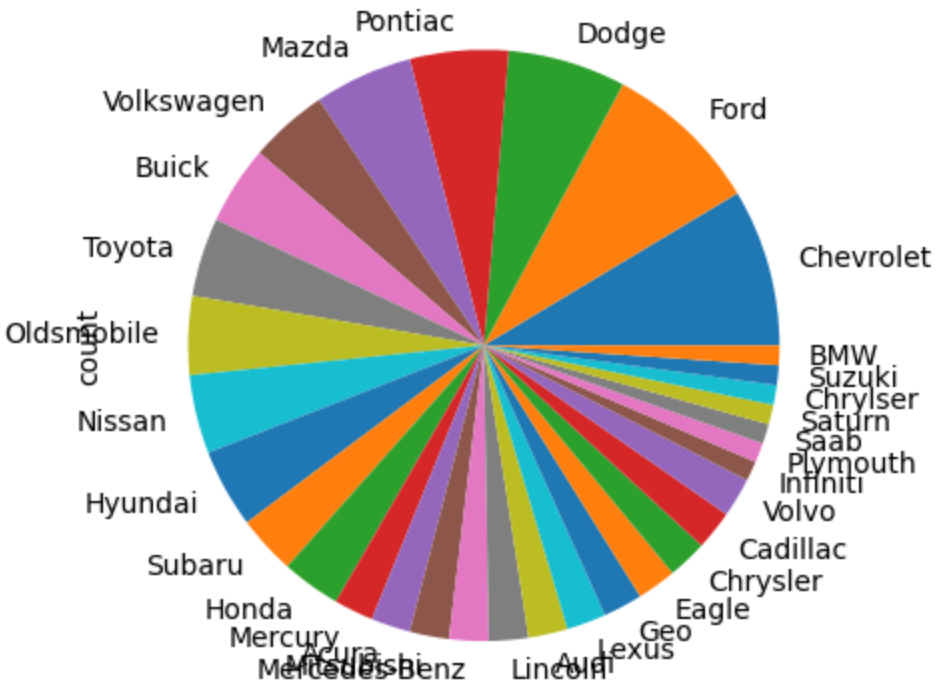
Out[16]:
```
['id',
 'MinPrice',
 'Price',
 'MaxPrice',
 'MPGcity',
 'MPGhighway',
 'Horsepower',
 'Fueltankcapacity']
```

1. Prepare Charts according to the Data (cat) or (con)

catogerical charts:

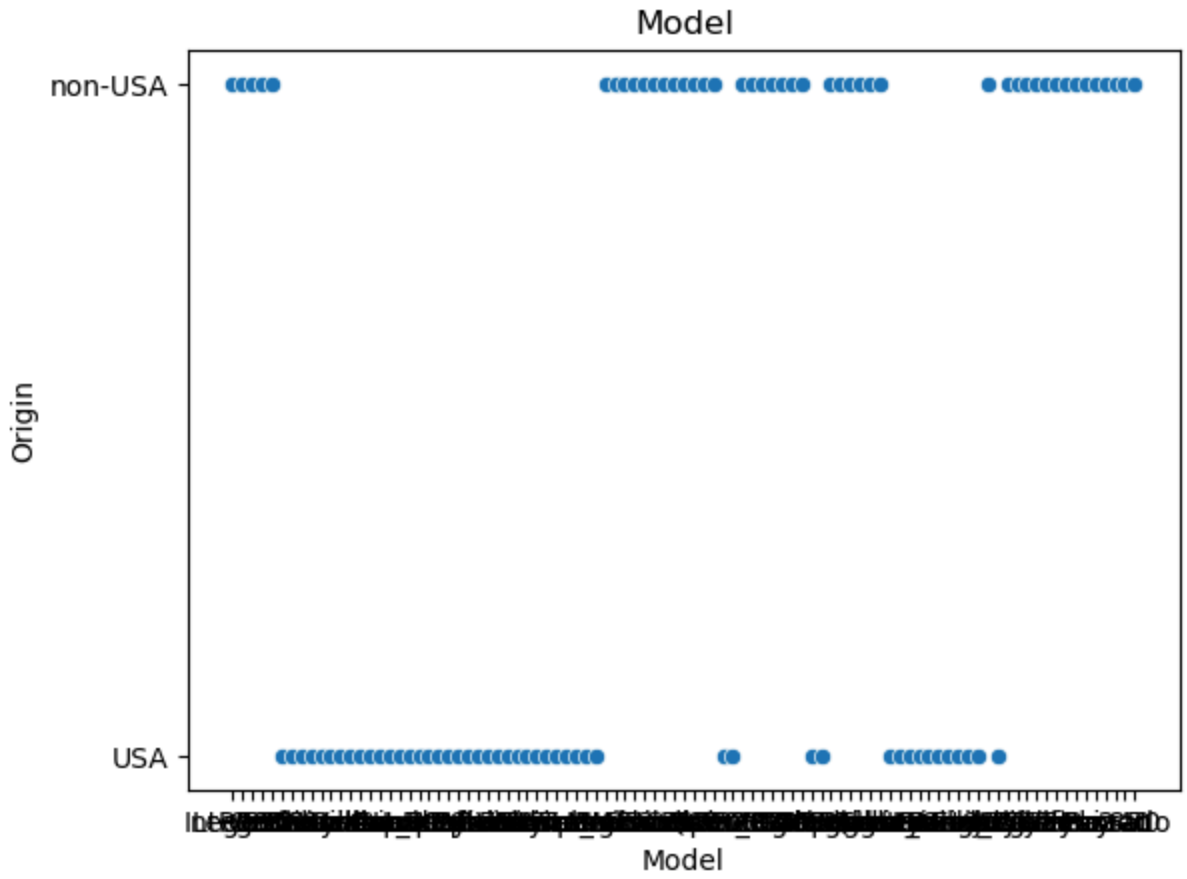
In [17]:
```
A.Manufacturer.value_counts().plot(kind="pie")
```

Out[17]:
```
<Axes: ylabel='count'>
```



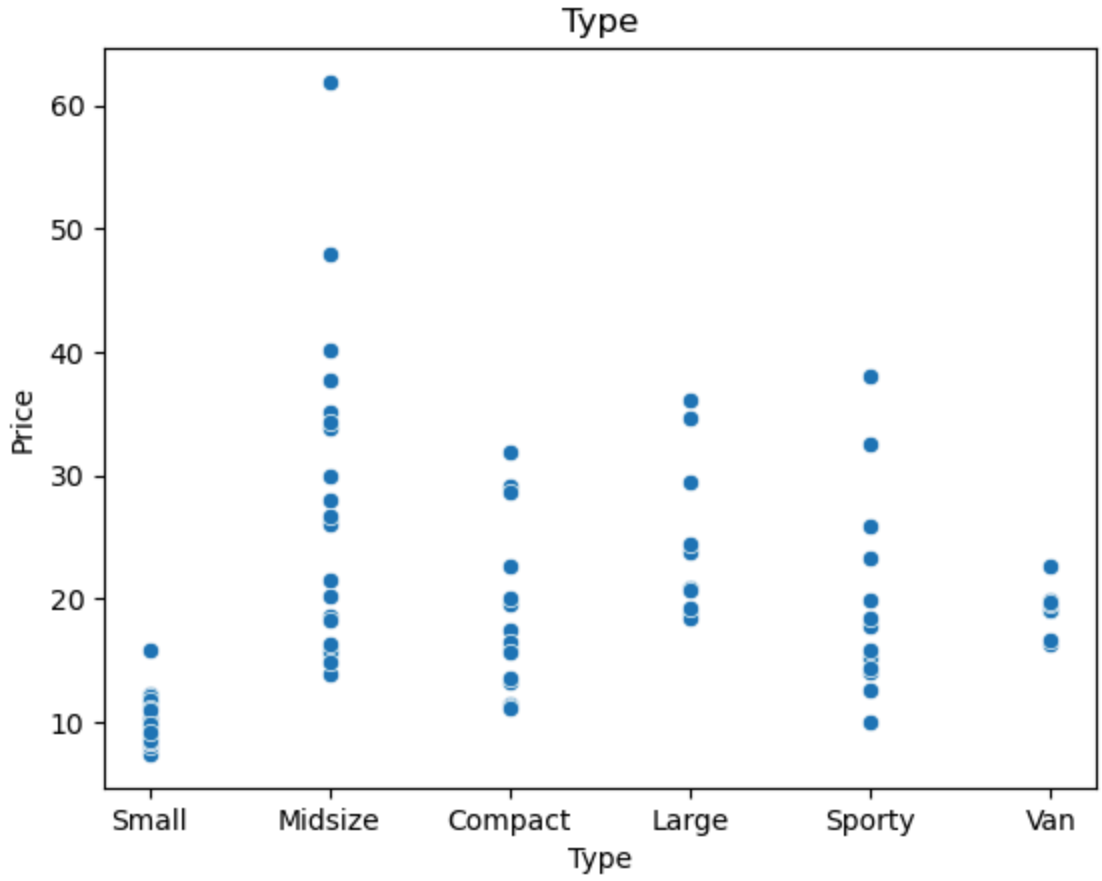
In [18]:
```
sb.scatterplot(data=A,x="Model",y="Origin")
plt.title("Model")
```

Out[18]:
```
Text(0.5, 1.0, 'Model')
```



In [21]:
```
sb.scatterplot(data=A,x="Type",y="Price")
plt.title("Type")
```

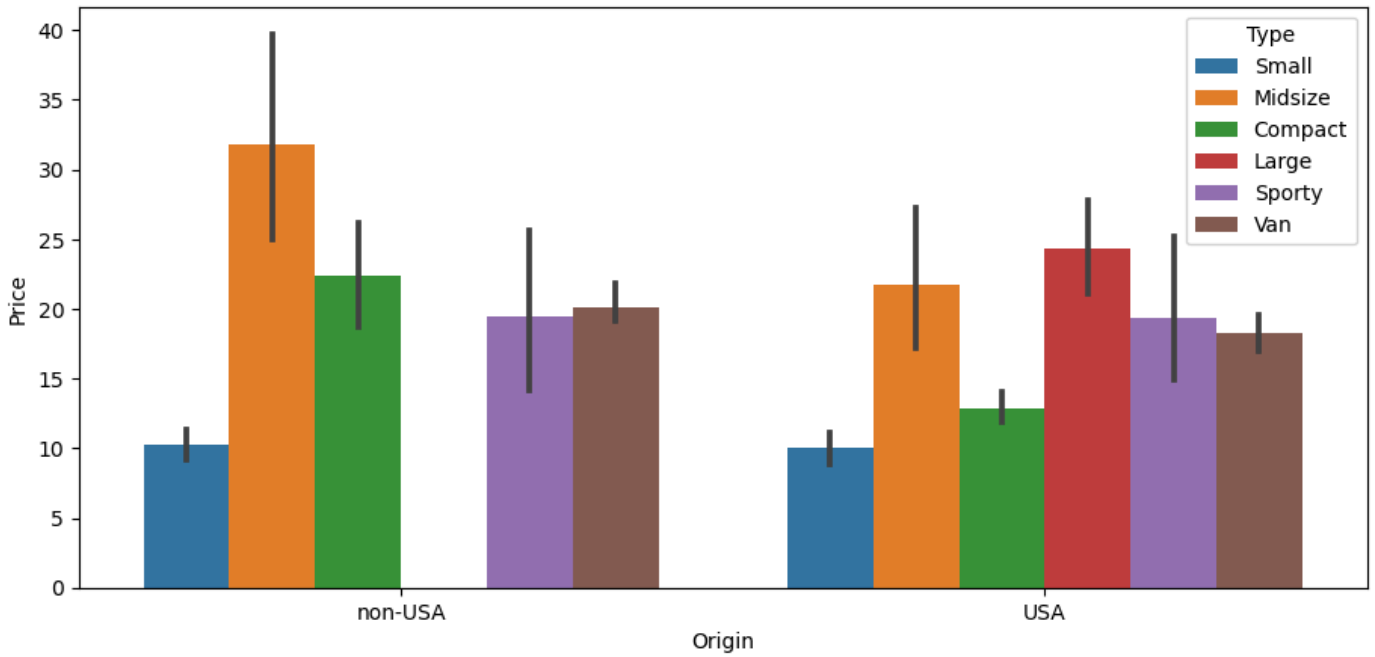
Out[21]:
```
Text(0.5, 1.0, 'Type')
```



In [20]:
```
import warnings
warnings.filterwarnings('ignore')
```

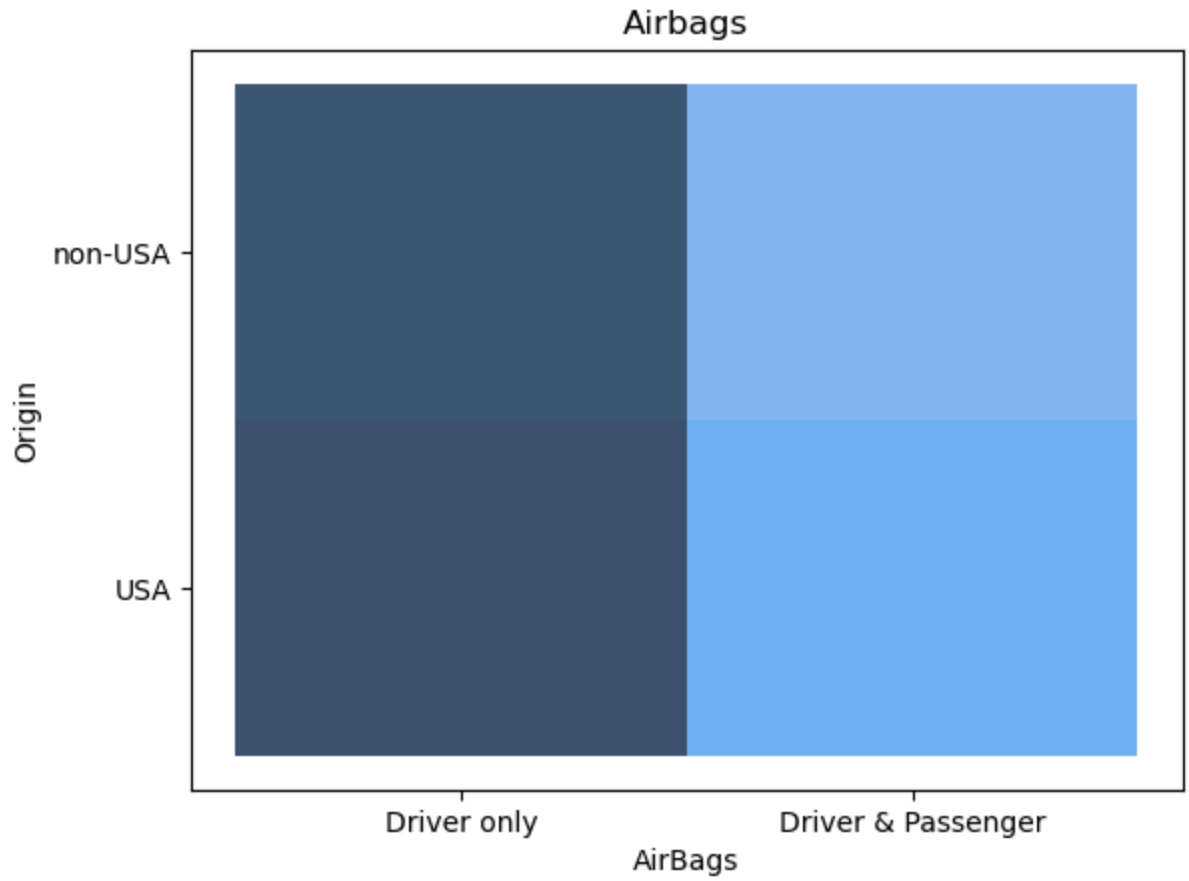
In [74]:
```
plt.figure(figsize=(11,5))
sb.barplot(data=A,x="Origin", y="Price", hue=A['Type'])
```

Out[74]:
```
<Axes: xlabel='Origin', ylabel='Price'>
```



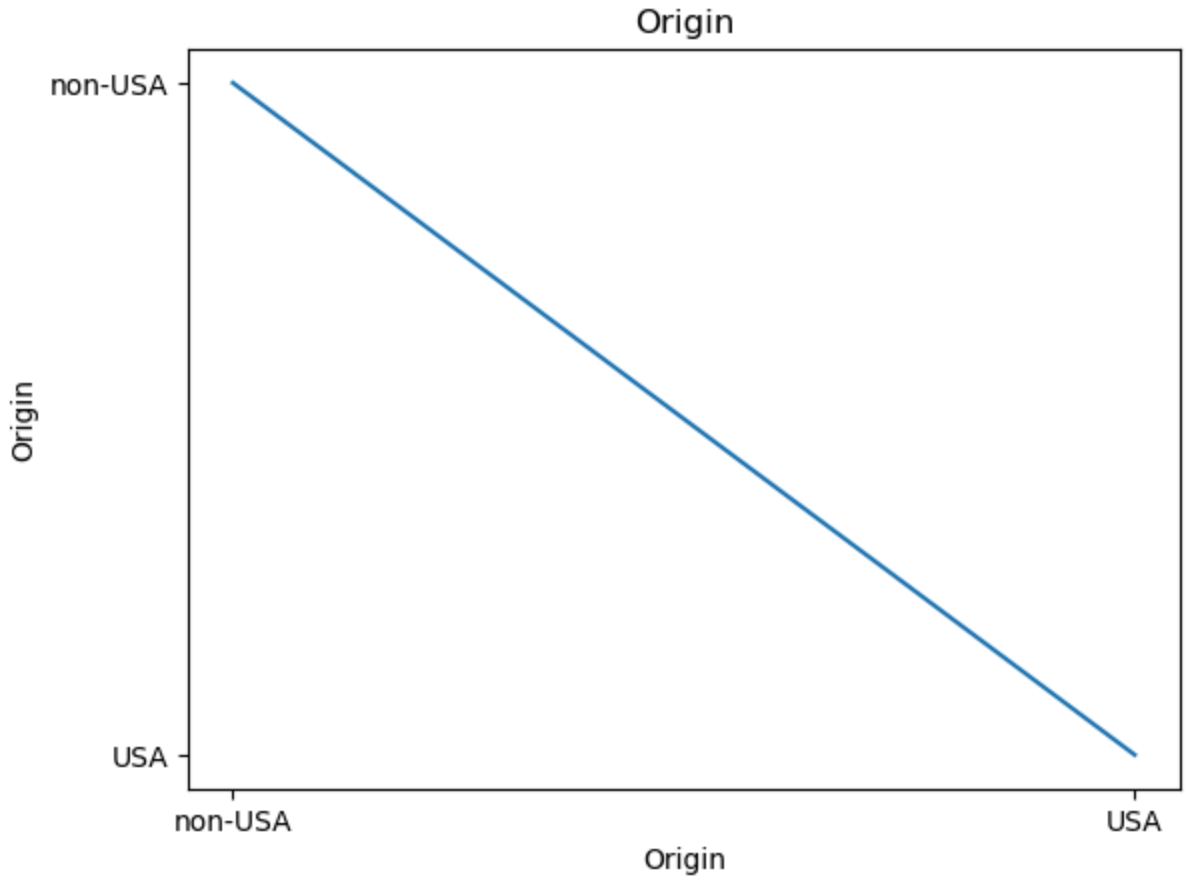
In [22]:
```
sb.histplot(data=A,x="AirBags",y="Origin")
plt.title("Airbags")
```

Out[22]:
```
Text(0.5, 1.0, 'Airbags')
```



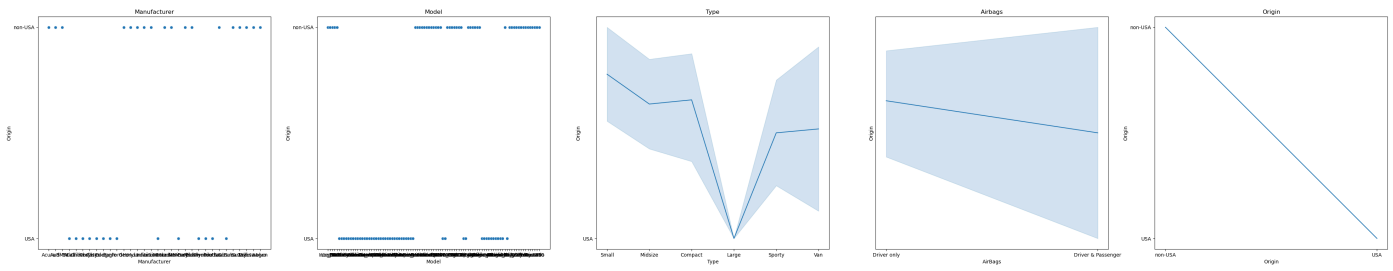
In [23]:
```
sb.lineplot(data=A,x="Origin",y="Origin")
plt.title("Origin")
```

Out[23]: Text(0.5, 1.0, 'Origin')



In [24]:
```
plt.figure(figsize=(100,100))
plt.subplot(10,10,1)
sb.scatterplot(data=A,x="Manufacturer",y="Origin")
plt.title("Manufacturer")
plt.subplot(10,10,2)
sb.scatterplot(data=A,x="Model",y="Origin")
plt.title("Model")
plt.subplot(10,10,3)
sb.lineplot(data=A,x="Type",y="Origin")
plt.title("Type")
plt.subplot(10,10,4)
sb.lineplot(data=A,x="AirBags",y="Origin")
plt.title("Airbags")
plt.subplot(10,10,5)
sb.lineplot(data=A,x="Origin",y="Origin")
plt.title("Origin")
```

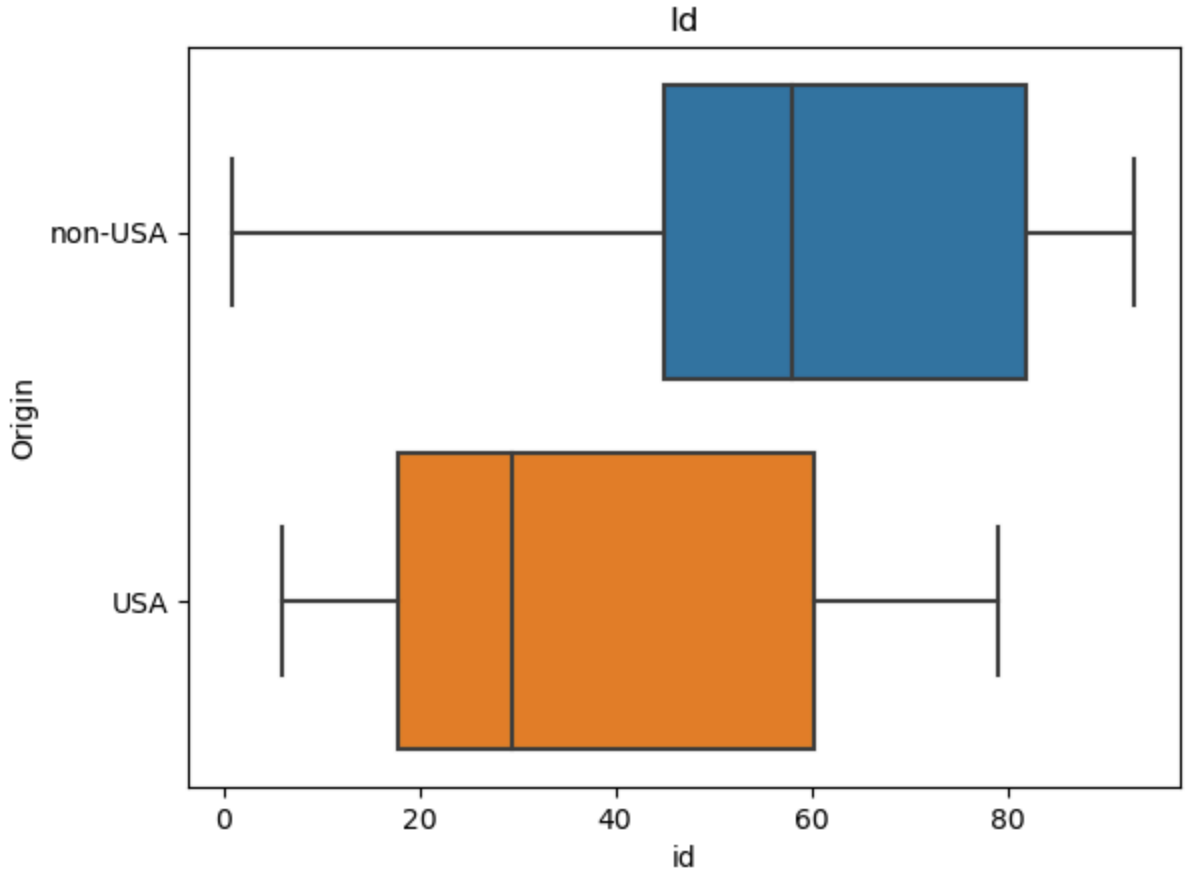
Out[24]: Text(0.5, 1.0, 'Origin')



Continous data on charts AS:-

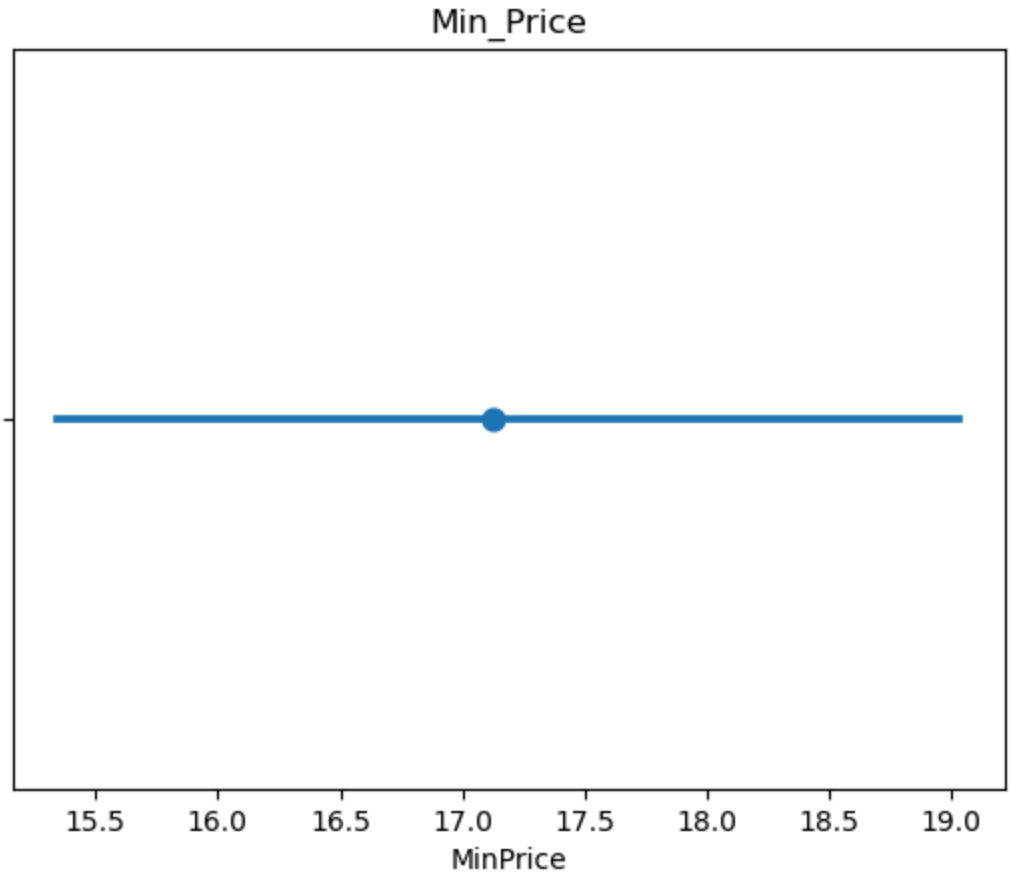
In [25]:
```
sb.boxplot(data=A,x="id",y="Origin")
plt.title("Id")
```

Out[25]: Text(0.5, 1.0, 'Id')



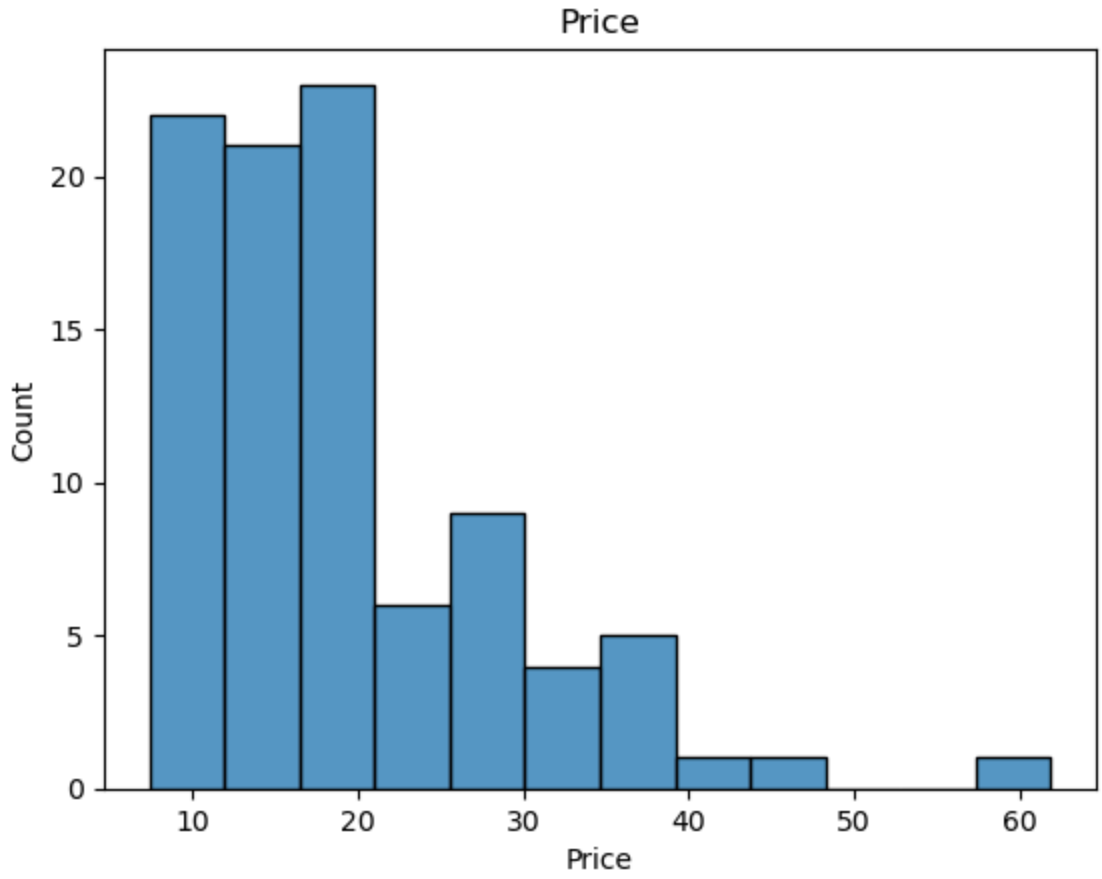
In [10]:
```
sb.pointplot(data=A,x="MinPrice")
plt.title("Min_Price")
```

Out[10]: Text(0.5, 1.0, 'Min_Price')



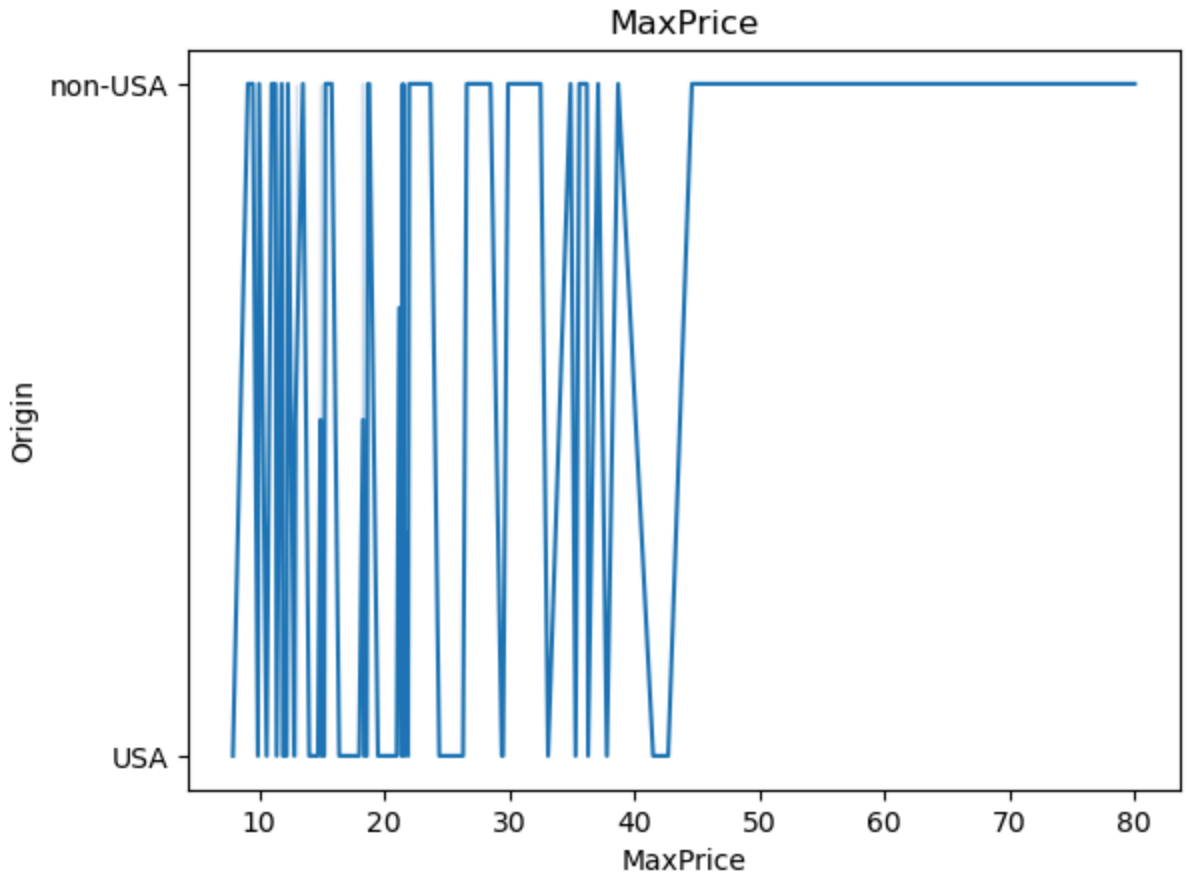
In [11]:
```
sb.histplot(data=A,x="Price")
plt.title("Price")
```

Out[11]:
```
Text(0.5, 1.0, 'Price')
```



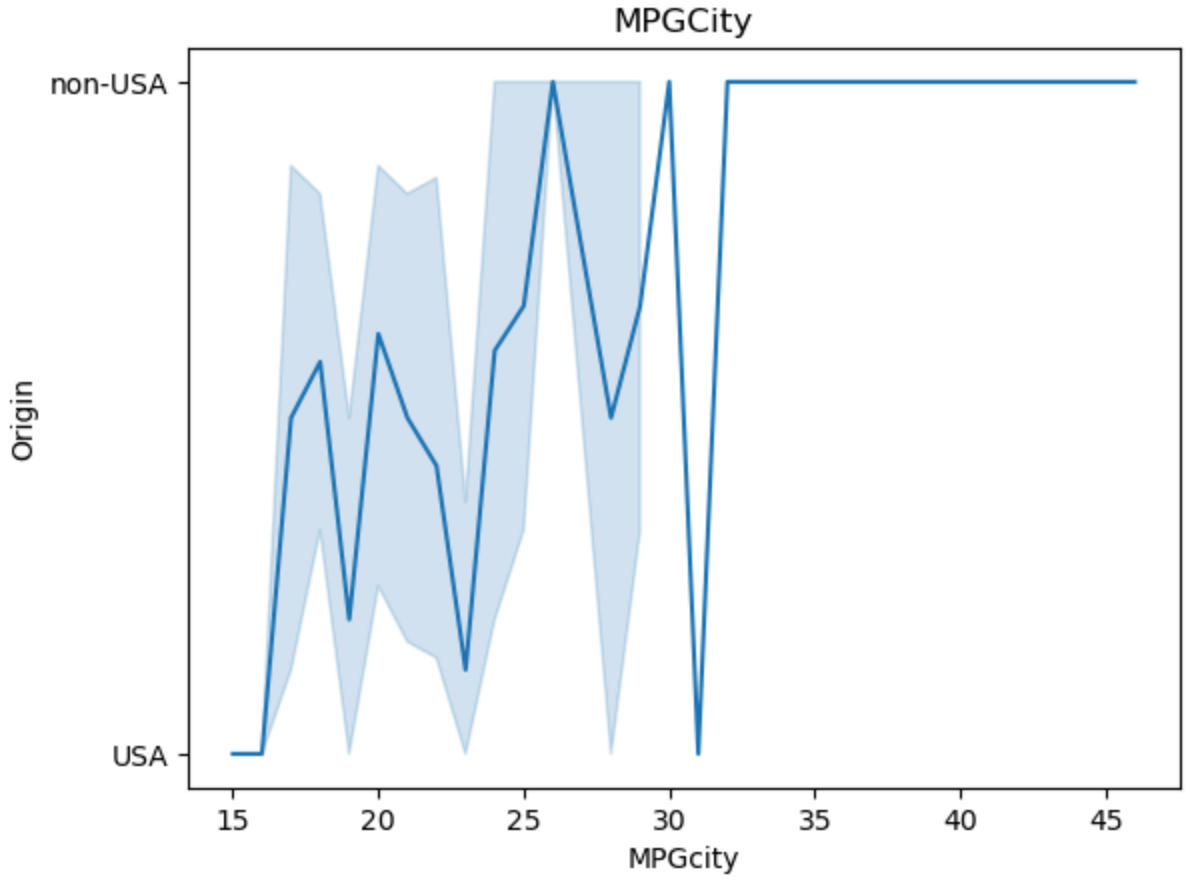
In [26]:
```
sb.lineplot(data=A,x="MaxPrice",y="Origin")
plt.title("MaxPrice")
```

Out[26]:
```
Text(0.5, 1.0, 'MaxPrice')
```



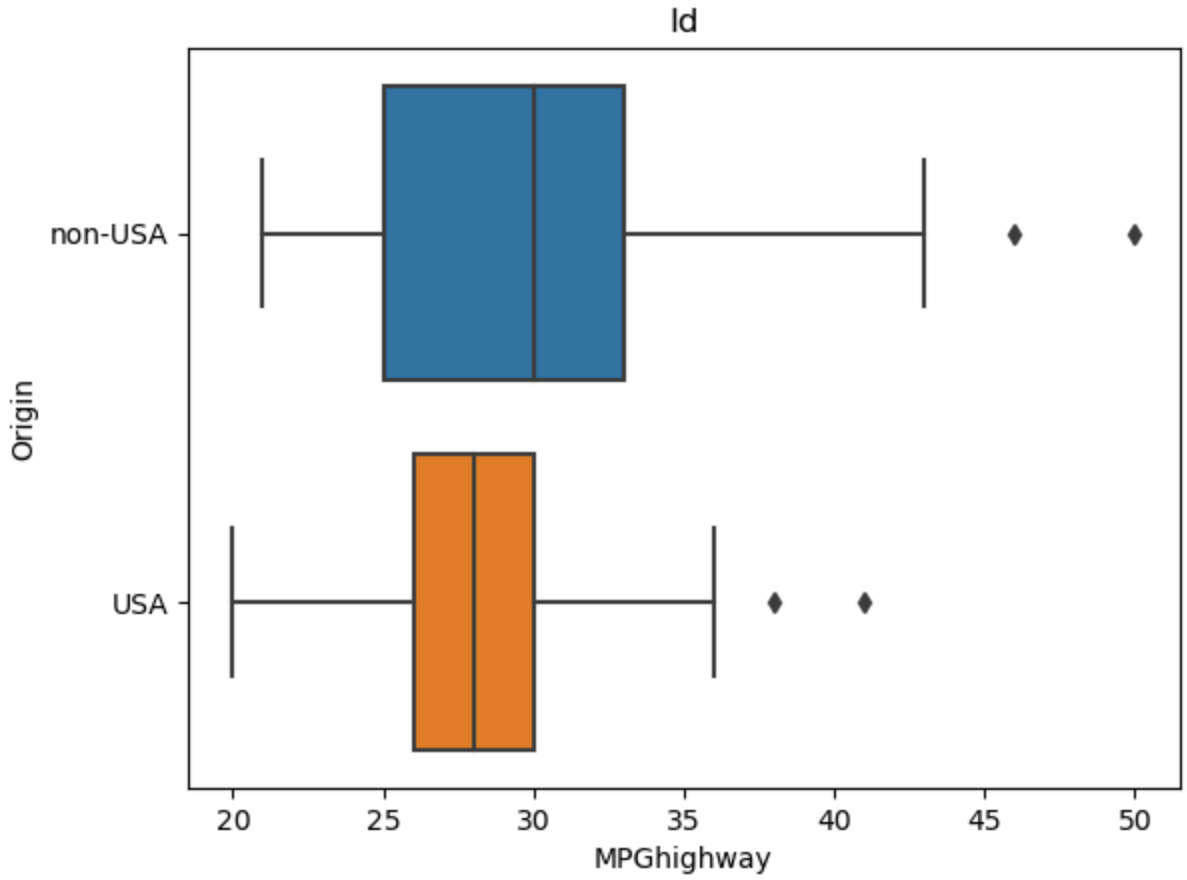
In [49]:
```
sb.lineplot(data=A,x="MPGcity",y="Origin")
plt.title("MPGCity")
```

Out[49]:
```
Text(0.5, 1.0, 'MPGCity')
```



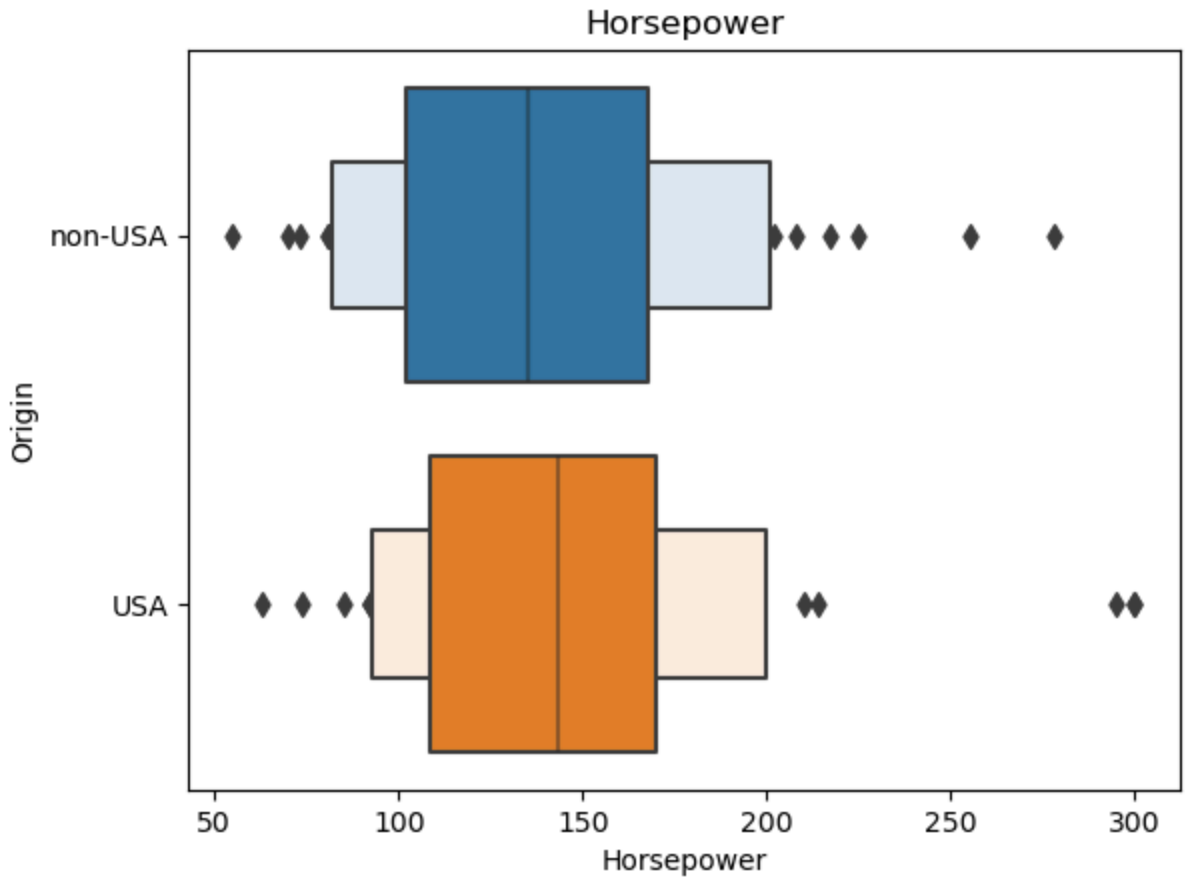
In [28]:
```
sb.boxplot(data=A,x="MPGhighway",y="Origin")
l")
```

Out[28]:
```
Text(0.5, 1.0, 'Id')
```



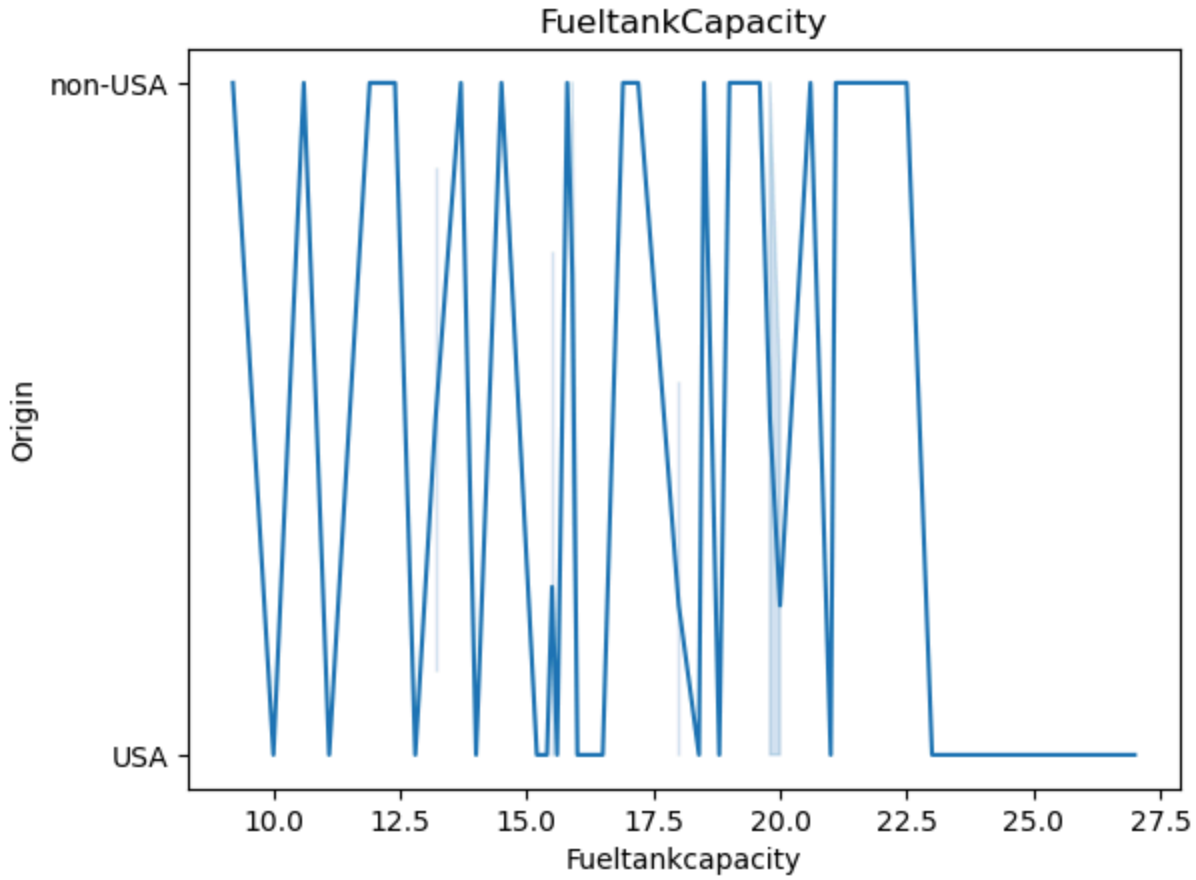
In [29]:
```
sb.boxenplot(data=A,x="Horsepower",y="Origin")
plt.title("Horsepower")
```

Out[29]:
```
Text(0.5, 1.0, 'Horsepower')
```



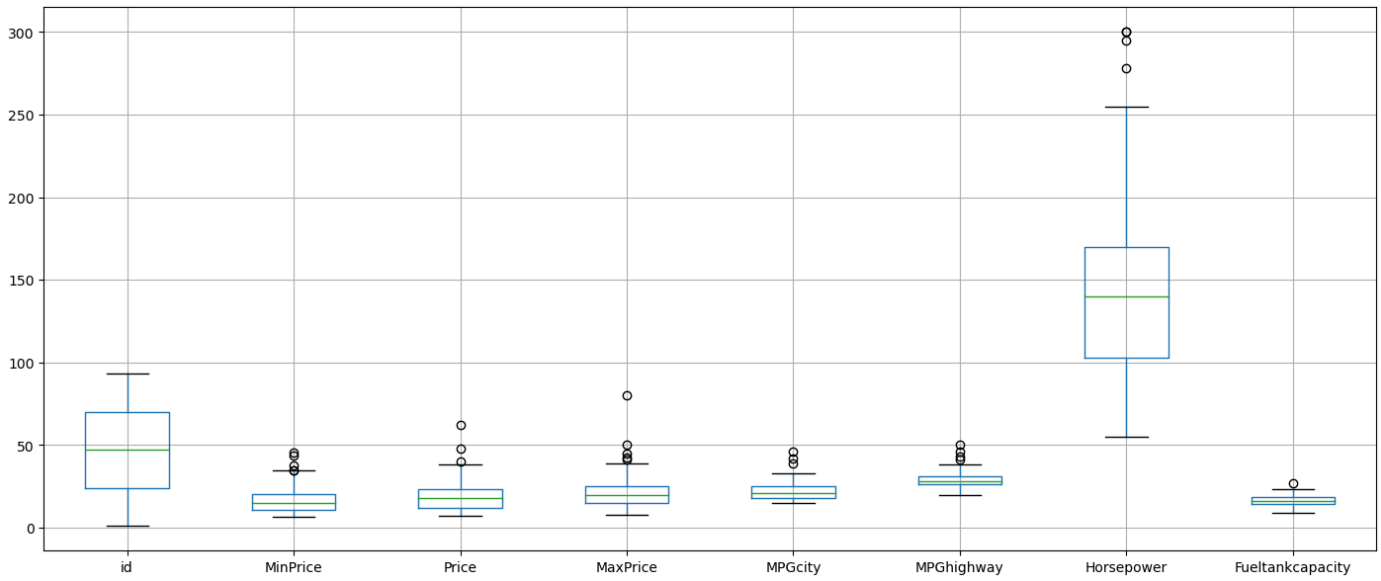
In [30]:
```
sb.lineplot(data=A,x="Fueltankcapacity",y="Origin")
plt.title("FueltankCapacity")
```

Out[30]:
```
Text(0.5, 1.0, 'FueltankCapacity')
```



In [31]:
```
plt.figure(figsize=(80,50))
plt.subplot(6,4,1)
x.boxplot()
```

Out[31]:
```
<Axes: >
```



Queries about SQL:-

In [32]:
```
#pip install sqldf
import pandas as pd
```

In [33]:
```
C=pd.read_csv("C:/Users/ASUS/OneDrive/Desktop/Datasets in python/Cars93.csv")
C
```

Out[33]:

	id	Manufacturer	Model	Type	MinPrice	Price	MaxPrice	MPGcity	MPGhighway	AirBags	...	Pass
0	1	Acura	Integra	Small	12.9	15.9	18.8	25	31	NaN	...	
1	2	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger	...	
2	3	Audi	90	Compact	25.9	29.1	32.3	20	26	Driver only	...	
3	4	Audi	100	Midsize	30.8	37.7	44.6	19	26	NaN	...	
4	5	BMW	535i	Midsize	23.7	30.0	36.2	22	30	Driver only	...	
...	...	...	...	...	...	...	...	...	...	...	...	
88	89	Volkswagen	Eurovan	Van	16.6	19.7	22.7	17	21	NaN	...	
89	90	Volkswagen	Passat	Compact	17.6	20.0	22.4	21	30	NaN	...	
90	91	Volkswagen	Corrado	Sporty	22.9	23.3	23.7	18	25	NaN	...	
91	92	Volvo	240	Compact	21.8	22.7	23.5	21	28	Driver only	...	
92	93	Volvo	850	Midsize	24.8	26.7	28.5	20	28	Driver & Passenger	...	

93 rows × 28 columns

In [34]:
```
from pandasql import sqldf
```

1.Find out car having highest Mileage in city

In [35]:
```
sqldf("select Model,max(MPGcity) from C")
```

Out[35]:

	Model	max(MPGcity)
0	Metro	46

1. Find out Models of cars which are having Price between 17 and 25

In [36]:
```
sqldf("select Price,Model from C where Price between 17 and 25 order by Price")
```

Out[36]:

	Price	Model
0	17.5	Accord
1	17.7	Firebird
2	18.2	Camry
3	18.4	Concorde
4	18.4	Celica
5	18.5	Grand_Prix
6	18.8	Caprice
7	19.0	Caravan
8	19.1	MPV
9	19.1	Quest
10	19.3	Vision
11	19.5	Silhouette
12	19.5	Legacy
13	19.7	Eurovan
14	19.8	Prelude
15	19.9	Aerostar
16	20.0	Passat
17	20.2	Taurus
18	20.7	Eighty-Eight
19	20.8	LeSabre
20	20.9	Crown_Victoria
21	21.5	Maxima
22	22.7	Previa
23	22.7	240
24	23.3	Corrado
25	23.7	Roadmaster
26	24.4	Bonneville

1. Find out top 7 most fuel economic cars for highway

In [37]:
```
sqldf("select MPGhighway,Model from C Order By MPGhighway desc limit 7")
```

Out[37]:

	MPGhighway	Model
0	50	Metro
1	46	Civic
2	43	Swift
3	41	LeMans
4	38	SL
5	37	323
6	37	Justy

1. Find out Models of cars having no AirBags and which are of compact type whose price is below 20.

In [42]:
```
sqldf("select Type,AirBags,Price from C where Type='Compact' and AirBags='None' Order By
```

Out[42]:

	Type	AirBags	Price

1. Find out top 7 most fuel economic cars for city, show results in sorted manner based on Horsepower.

In [39]:
```
sqldf("select MPGcity,Horsepower from C order by MPGcity desc limit 7")
```

Out[39]:

	MPGcity	Horsepower
0	46	55
1	42	102
2	39	70
3	33	73
4	32	82
5	31	63
6	31	74