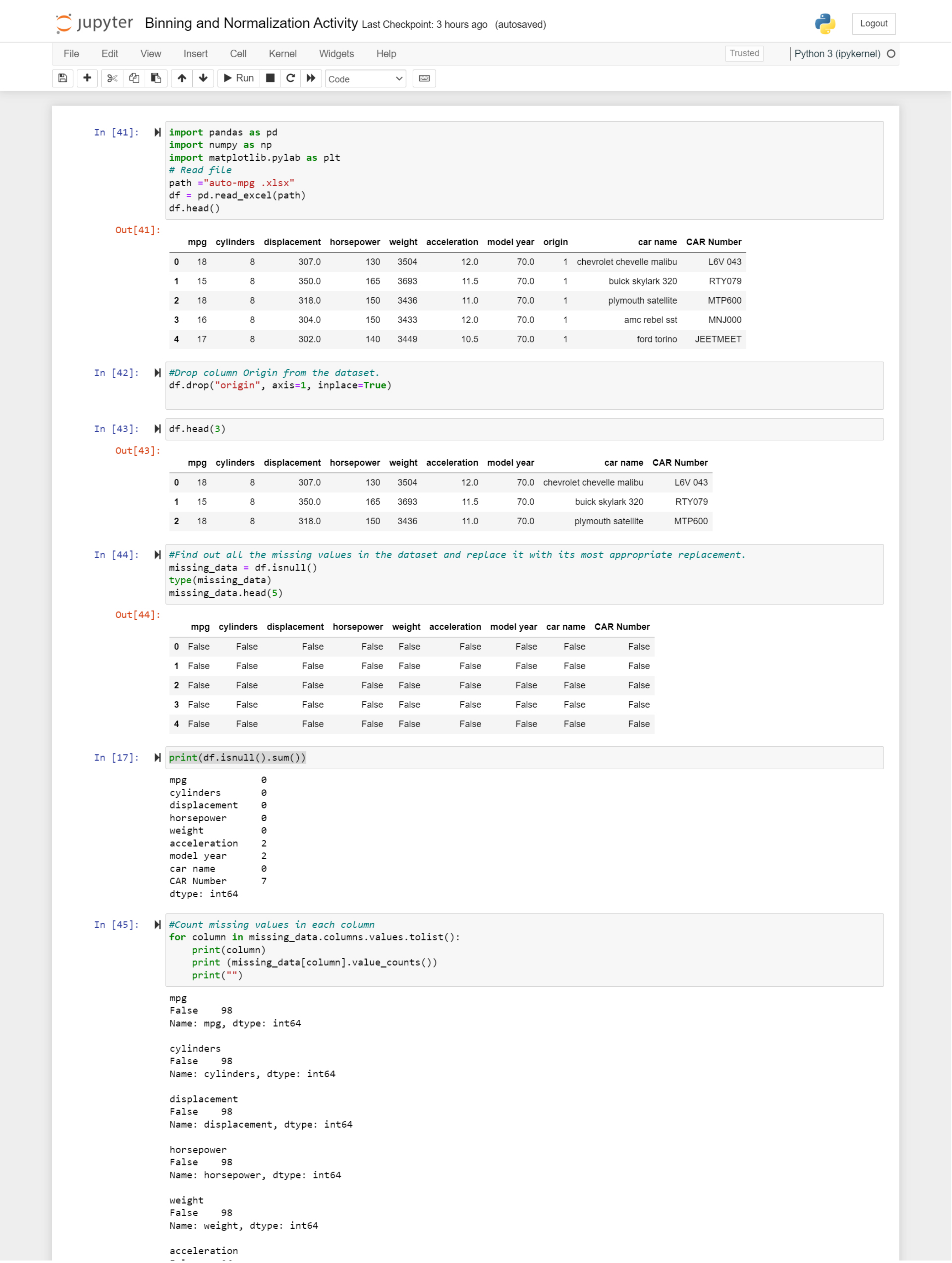

jupyter
Binning and Normalization Activity Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

In [41]:

```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
# Read file
path ="auto-mpg .xlsx"
df = pd.read_excel(path)
df.head()
```

Out[41]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name	CAR Number
0	18	8	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu	L6V 043
1	15	8	350.0	165	3693	11.5	70.0	1	buick skylark 320	RTY079
2	18	8	318.0	150	3436	11.0	70.0	1	plymouth satellite	MTP600
3	16	8	304.0	150	3433	12.0	70.0	1	amc rebel sst	MNJ000
4	17	8	302.0	140	3449	10.5	70.0	1	ford torino	JEETMEET

In [42]:

```
#Drop column Origin from the dataset.
df.drop("origin", axis=1, inplace=True)
```

In [43]:

```
df.head(3)
```

Out[43]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number
0	18	8	307.0	130	3504	12.0	70.0	chevrolet chevelle malibu	L6V 043
1	15	8	350.0	165	3693	11.5	70.0	buick skylark 320	RTY079
2	18	8	318.0	150	3436	11.0	70.0	plymouth satellite	MTP600

In [44]:

```
#Find out all the missing values in the dataset and replace it with its most appropriate replacement.
missing_data = df.isnull()
type(missing_data)
missing_data.head(5)
```

Out[44]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False

In [17]:

```
print(df.isnull().sum())
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    2
model year      2
car name        0
CAR Number      7
dtype: int64
```

In [45]:

```
#Count missing values in each column
for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")
```

```
mpg
False    98
Name: mpg, dtype: int64

cylinders
False    98
Name: cylinders, dtype: int64

displacement
False    98
Name: displacement, dtype: int64

horsepower
False    98
Name: horsepower, dtype: int64

weight
False    98
Name: weight, dtype: int64

acceleration
```

```
False    96
True      2
Name: acceleration, dtype: int64

model year
False    96
True      2
Name: model year, dtype: int64

car name
False    98
Name: car name, dtype: int64

CAR Number
False    91
True      7
Name: CAR Number, dtype: int64
```

In [46]:

```
#Calculate the mean value for the "acceleration" column
avg_acceleration = df["acceleration"].astype("float").mean(axis=0)
print("Average of acceleration:", avg_acceleration)
```

```
Average of acceleration: 14.046875
```

In [47]:

```
#Replace "NaN" with the mean value in the "acceleration" column
df["acceleration"].replace(np.nan, avg_acceleration, inplace=True)
```

In [48]:

```
print(df.isnull().sum())
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      2
car name        0
CAR Number      7
dtype: int64
```

In [49]:

```
#Calculate the median value for the "model year" column
median_model_year = df["model year"].median()
print("median for model year:", median_model_year)
```

```
median for model year: 71.0
```

In [50]:

```
#Replace "NaN" with the median value in the "model year" column
df["model year"].replace(np.nan,median_model_year, inplace=True)
```

In [51]:

```
print(df.isnull().sum())
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
car name        0
CAR Number      7
dtype: int64
```

In [52]:

```
# simply drop whole row with NaN in "CAR Number" column
df.dropna(subset=["CAR Number"], axis=0, inplace=True)

# reset index, because we droped 7 rows *****
df.reset_index(drop=True, inplace=True)
df.head()
```

Out[52]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number
0	18	8	307.0	130	3504	12.0	70.0	chevrolet chevelle malibu	L6V 043
1	15	8	350.0	165	3693	11.5	70.0	buick skylark 320	RTY079
2	18	8	318.0	150	3436	11.0	70.0	plymouth satellite	MTP600
3	16	8	304.0	150	3433	12.0	70.0	amc rebel sst	MNJ000
4	17	8	302.0	140	3449	10.5	70.0	ford torino	JEETMEET

In [53]:

```
print(df.isnull().sum())
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
car name        0
CAR Number      0
dtype: int64
```

In [54]:

```
df.shape[0]
```

Out[54]: 91

In [55]:

```python
#Find and remove duplicate entries for the column ‘ CAR Number’
df.drop_duplicates(inplace=True)
```

In [57]:

```python
df.head()
```

Out[57]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number
0	18	8	307.0	130	3504	12.0	70.0	chevrolet chevelle malibu	L6V 043
1	15	8	350.0	165	3693	11.5	70.0	buick skylark 320	RTY079
2	18	8	318.0	150	3436	11.0	70.0	plymouth satellite	MTP600
3	16	8	304.0	150	3433	12.0	70.0	amc rebel sst	MNJ000
4	17	8	302.0	140	3449	10.5	70.0	ford torino	JEETMEET

In [58]:

```python
df.shape[0]
```

Out[58]:

```
90
```

In [59]:

```python
#Get the basic insights:
#Display the first five and last five rows
df.head()
```

Out[59]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number
0	18	8	307.0	130	3504	12.0	70.0	chevrolet chevelle malibu	L6V 043
1	15	8	350.0	165	3693	11.5	70.0	buick skylark 320	RTY079
2	18	8	318.0	150	3436	11.0	70.0	plymouth satellite	MTP600
3	16	8	304.0	150	3433	12.0	70.0	amc rebel sst	MNJ000
4	17	8	302.0	140	3449	10.5	70.0	ford torino	JEETMEET

In [60]:

```python
df.tail()
```

Out[60]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number
86	14	8	318.0	150	4237	14.5	73.0	plymouth fury gran sedan	FGTEE4
87	13	8	440.0	215	4735	11.0	73.0	chrysler new yorker brougham	GTTYR6
88	12	8	455.0	225	4951	11.0	73.0	buick electra 225 custom	TTGGT53
89	13	8	360.0	175	3821	11.0	73.0	amc ambassador brougham	TGTYY67
90	18	6	225.0	105	3121	16.5	73.0	plymouth valiant	HUYT76

In [61]:

```python
#Display all the column names in the dataset
df.columns
```

Out[61]:

```
Index(['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model year', 'car name', 'CAR Number'],
      dtype='object')
```

In [62]:

```python
#Display the concise summary of your dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 90 entries, 0 to 90
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           90 non-null     int64  
 1   cylinders     90 non-null     int64  
 2   displacement  90 non-null     float64
 3   horsepower    90 non-null     int64  
 4   weight        90 non-null     int64  
 5   acceleration  90 non-null     float64
 6   model year    90 non-null     float64
 7   car name      90 non-null     object 
 8   CAR Number    90 non-null     object 
dtypes: float64(3), int64(4), object(2)
memory usage: 7.0+ KB
```

In [69]:

```python
#Display the name of the car with maximum number of horsepower.
max_horsehour=df["horsepower"].max()
df
```

Out[69]:

```
225
```

In [76]:

```python
#Display the name of the car with maximum number of horsepower.*****

df [['car name','horsepower']][df.horsepower==df['horsepower'].max()]
```

Out[76]:

	car name	horsepower
8	pontiac catalina	225
12	buick estate wagon (sw)	225
88	buick electra 225 custom	225

In [77]:

```python
#In our dataset, the fuel consumption column is "mpg" and is represented by mpg (miles per gallon) unit.
#Assume we are developing an application in a country that accepts fuel consumption with the L/100km standard.
#change the name of the column to "L/100km".
```

```
df['city-L/100km'] = 235/df["mpg"]

# check your transformed data
df.head()
```

Out[77]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number	city-L/100km
0	18	8	307.0	130	3504	12.0	70.0	chevrolet chevelle malibu	L6V 043	13.055556
1	15	8	350.0	165	3693	11.5	70.0	buick skylark 320	RTY079	15.666667
2	18	8	318.0	150	3436	11.0	70.0	plymouth satellite	MTP600	13.055556
3	16	8	304.0	150	3433	12.0	70.0	amc rebel sst	MNJ000	14.687500
4	17	8	302.0	140	3449	10.5	70.0	ford torino	JEETMEET	13.823529

In [80]:

```
df.drop('mpg', axis=1, inplace=True)
df.head()
```

Out[80]:

	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number	city-L/100km
0	8	307.0	130	3504	12.0	70.0	chevrolet chevelle malibu	L6V 043	13.055556
1	8	350.0	165	3693	11.5	70.0	buick skylark 320	RTY079	15.666667
2	8	318.0	150	3436	11.0	70.0	plymouth satellite	MTP600	13.055556
3	8	304.0	150	3433	12.0	70.0	amc rebel sst	MNJ000	14.687500
4	8	302.0	140	3449	10.5	70.0	ford torino	JEETMEET	13.823529

In [81]:

```
#6-Normalize the column “Weight” so that the values range from 0 to 1
df['weight'] = df['weight']/df['weight'].max()
df.head()
```

Out[81]:

	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number	city-L/100km
0	8	307.0	130	0.681712	12.0	70.0	chevrolet chevelle malibu	L6V 043	13.055556
1	8	350.0	165	0.718482	11.5	70.0	buick skylark 320	RTY079	15.666667
2	8	318.0	150	0.668482	11.0	70.0	plymouth satellite	MTP600	13.055556
3	8	304.0	150	0.667899	12.0	70.0	amc rebel sst	MNJ000	14.687500
4	8	302.0	140	0.671012	10.5	70.0	ford torino	JEETMEET	13.823529

In [ ]:

```
#Normalization is the process of transforming values of several variables into a similar range.
#Typical normalizations include scaling the variable so the variable average is 0,
#scaling the variable so the variance is 1, or scaling variable so the variable values range from 0 to 1
```

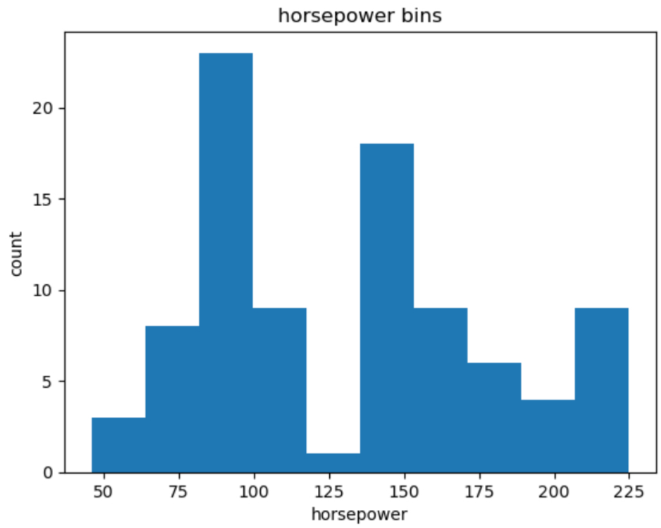
In [89]:

```
#histogram
%matplotlib inline
import matplotlib as plt
from matplotlib import pyplot
plt.pyplot.hist(df["horsepower"])

# set x/y labels and plot title
plt.pyplot.xlabel("horsepower")
plt.pyplot.ylabel("count")
plt.pyplot.title("horsepower bins")
```

Out[89]:

```
Text(0.5, 1.0, 'horsepower bins')
```



In [82]:

```
#11
#In our dataset, "horsepower" is a real valued variable ranging from 48 to 288, it has 57 unique values.
#What if we only care about the price difference between cars with high horsepower, medium horsepower,
#and little horsepower (3 types)? Can we rearrange them into three 'bins' to simplify analysis?
bins = np.linspace(min(df["horsepower"]), max(df["horsepower"]), 4)
bins
```

Out[82]:

```
array([ 46.        , 105.66666667, 165.33333333, 225.        ])
```

In [83]:

```
group_names = ['Low', 'Medium', 'High']
```

In [84]:

```
#We apply the function "cut" to determine what each value of df['horsepower'] belongs to.
df['horsepower-binned'] = pd.cut(df['horsepower'], bins, labels=group_names, include_lowest=True )
df[['horsepower','horsepower-binned']].head(20)
```

Out[84]:

	horsepower	horsepower-binned
0	130	Medium
1	165	Medium
2	150	Medium
3	150	Medium
4	140	Medium
5	198	High
6	220	High
7	215	High
8	225	High
9	170	High
10	160	Medium
11	150	Medium
12	225	High
13	95	Low
14	95	Low
15	97	Low
17	88	Low
18	46	Low
19	87	Low
20	90	Low

In [85]:

```
#Let's see the number of vehicles in each bin:
df["horsepower-binned"].value_counts()
```

Out[85]:

```
Low       40
Medium    29
High      21
Name: horsepower-binned, dtype: int64
```

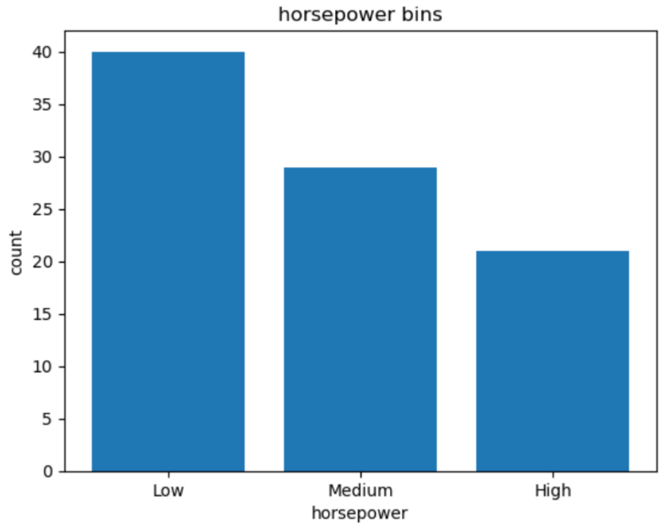
In [87]:

```
#Let's plot the distribution of each bin:
%matplotlib inline
import matplotlib as plt
from matplotlib import pyplot
pyplot.bar(group_names, df["horsepower-binned"].value_counts())

# set x/y labels and plot title
plt.pyplot.xlabel("horsepower")
plt.pyplot.ylabel("count")
plt.pyplot.title("horsepower bins")
```

Out[87]:

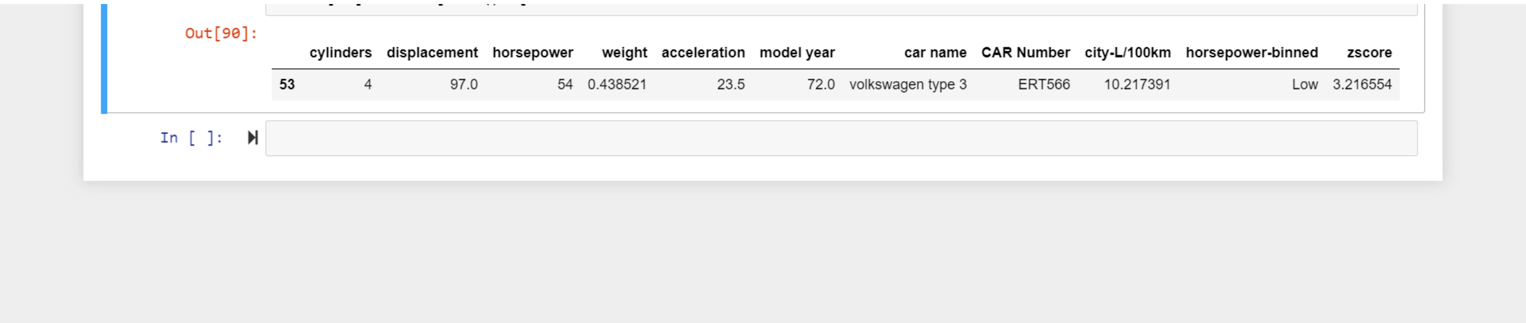
```
Text(0.5, 1.0, 'horsepower bins')
```



In [90]:

```
#Detect outliers using Z-score and remove them
#Z Score
from scipy import stats

df['zscore']=stats.zscore(df["acceleration"])
#outlier=np.where(score>3)
df.loc[df['zscore'].abs()>3]
```



Out[90]:

	cylinders	displacement	horsepower	weight	acceleration	model year	car name	CAR Number	city-L/100km	horsepower-binned	zscore
53	4	97.0	54	0.438521	23.5	72.0	volkswagen type 3	ERT566	10.217391	Low	3.216554

In [ ]: