Objectives In this lab you will perform the following: Identify duplicate values in the dataset. Remove duplicate values from the dataset. Identify missing values in the dataset. Impute the missing values in the dataset. Normalize data in the dataset.

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

pd.set_option('display.max_columns', None)
#pd.set_option('display.max_rows', None)
```

In [2]:
```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.
```

In [3]:
```
df.describe()
```

Out[3]:

	Respondent	CompTotal	ConvertedComp	WorkWeekHrs	CodeRevHrs	
count	11552.000000	1.073700e+04	1.073000e+04	11427.000000	9083.000000	1
mean	12362.212517	7.499932e+05	1.313340e+05	42.051851	4.762829	
std	7271.939210	9.639522e+06	2.943245e+05	24.528561	4.548401	
min	4.000000	0.000000e+00	0.000000e+00	3.000000	0.000000	
25%	6011.500000	2.500000e+04	2.672700e+04	40.000000	2.000000	
50%	12323.500000	6.500000e+04	5.774400e+04	40.000000	4.000000	
75%	18686.500000	1.200000e+05	1.000000e+05	43.000000	5.000000	
max	25142.000000	7.000000e+08	2.000000e+06	1012.000000	99.000000	

In [4]:
```
df.Employment.value_counts()
```

Out[4]:
```
Employed full-time    11113
Employed part-time      439
Name: Employment, dtype: int64
```

In [5]:
```
df.duplicated().sum()
```

Out[5]:
```
154
```

In [6]:
```
df=df.drop_duplicates()
```

In [7]:
```
df.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
df.isna().sum()
```

Out[8]:
```
Respondent        0
MainBranch        0
Hobbyist          0
OpenSourcer       0
OpenSource       81
                ...
Sexuality       542
Ethnicity       675
Dependents      140
SurveyLength     19
SurveyEase       14
Length: 85, dtype: int64
```

In [30]:
```
df[(df['Employment']=='Employed full-time') & (df['WorkWeekHrs']<10)]
```

Out[30]:
```
239
```

CASE:1 Working hours: Understanding the distributions of working hours need to look into the statistics numbers taking into consideration the part time employee has diffenet working hours distriputio. After determing the data we will see the stratgy to be used t deall with the outliesrs under each category.

In [13]:
```
df[df['Employment']=='Employed part-time']['WorkWeekHrs'].sort_values
```

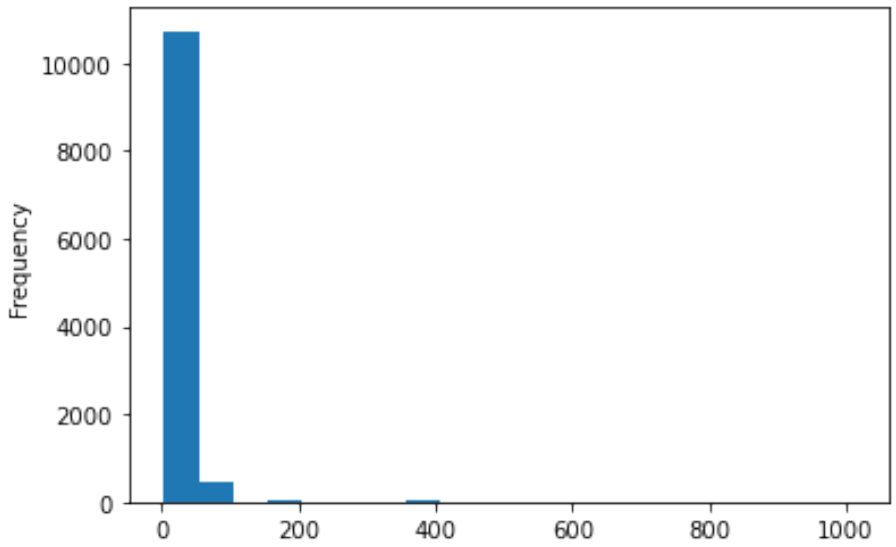
Out[13]:
```
8892    375.0
5396    145.0
7595     80.0
6462     75.0
Name: WorkWeekHrs, dtype: float64
```

In [14]:
```
df[df['Employment']=='Employed full-time']['WorkWeekHrs'].sort_values
```

Out[14]:
```
2753    1012.0
1001     475.0
9411     425.0
1923     385.0
Name: WorkWeekHrs, dtype: float64
```

In [15]:
```
df['WorkWeekHrs'].plot(kind='hist',bins=20)
```

Out[15]:
```
<AxesSubplot:ylabel='Frequency'>
```



In [16]:

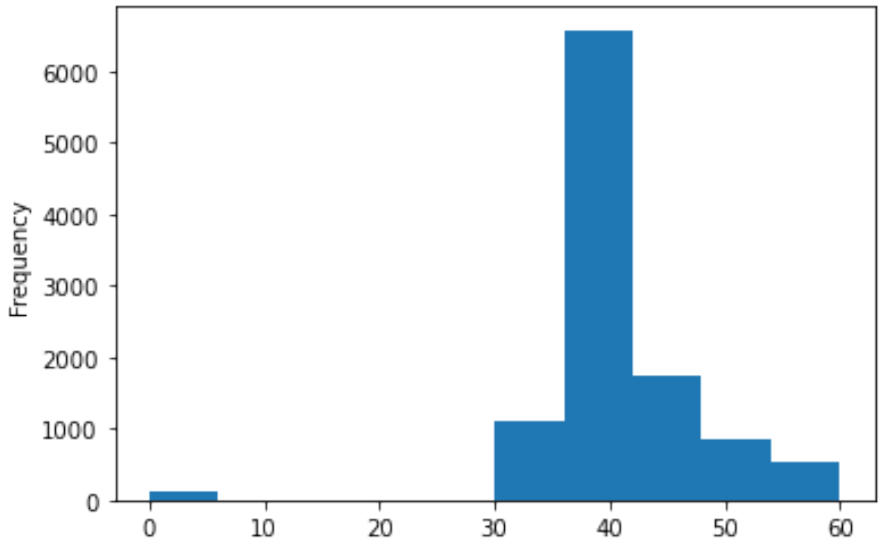
```
full_time_hours=[]
q_hi1  = 60
q_low1=30

q_hi2  = 30
q_low2=0
for x,y in zip(df['Employment'], df['WorkWeekHrs']):

        if (x=='Employed full-time') & (y<=q_low1) :
            full_time_hours.append(q_low1)
        elif (x=='Employed full-time') & (q_low1<y< q_hi1) :
            full_time_hours.append(y)
        elif (x=='Employed full-time') & ( y>= q_hi1) :
            full_time_hours.append(q_hi1)


        elif (x=='Employed part-time') & (y<= q_hi2) :
            full_time_hours.append(y)
        elif (x=='Employed part-time') & (y> q_hi2) :
            full_time_hours.append(q_hi2)

        else:
            full_time_hours.append(0)

df['Working_Hours']=full_time_hours
```

df[['Employment','Working_Hours','WorkWeekHrs']].sort_values(by='Employment',ascending=False)

In [17]:

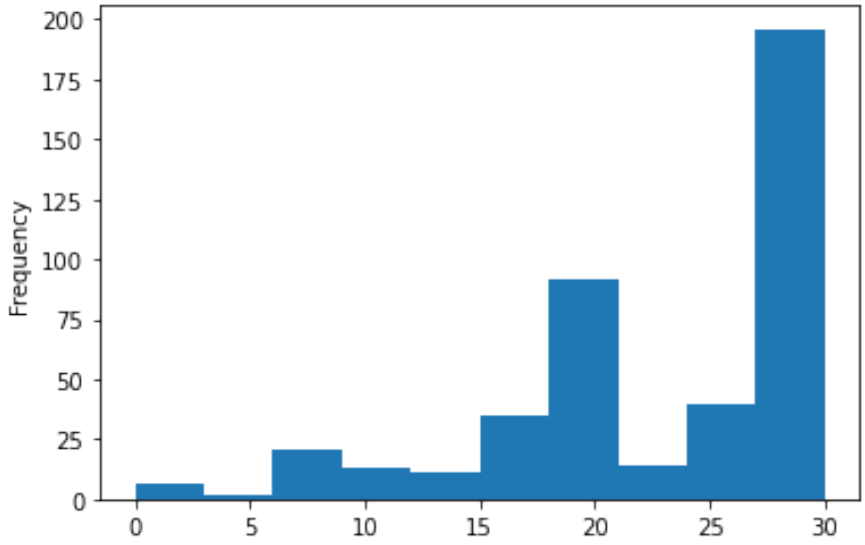
```
df[df['Employment']=='Employed full-time'].Working_Hours.plot(kind='h
```

Out[17]: <AxesSubplot:ylabel='Frequency'>

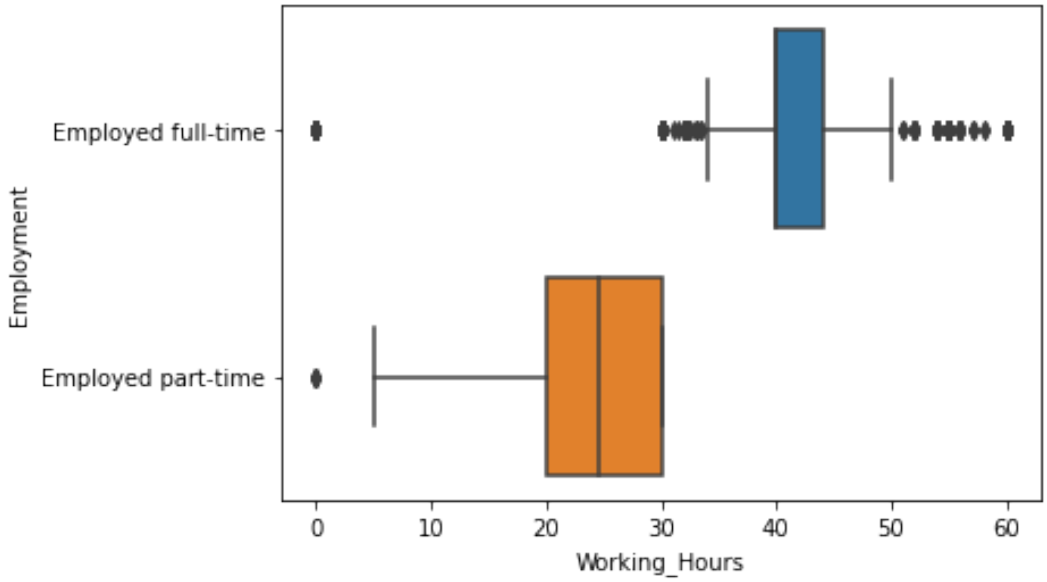


In [18]:
```
df[df['Employment']=='Employed part-time'].Working_Hours.plot(kind='h
```

Out[18]: <AxesSubplot:ylabel='Frequency'>

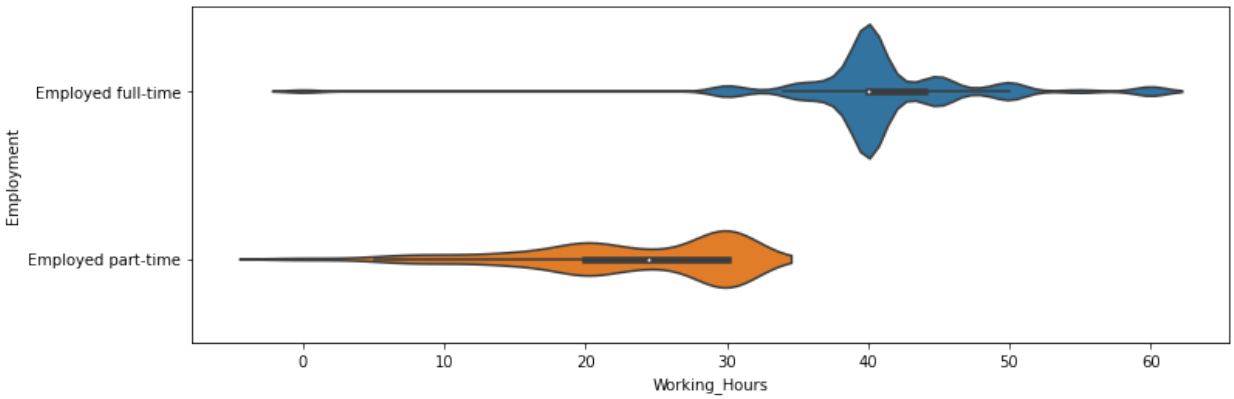


In [19]:
```
sns.boxplot(x='Working_Hours', y='Employment', data=df)
plt.show()
```



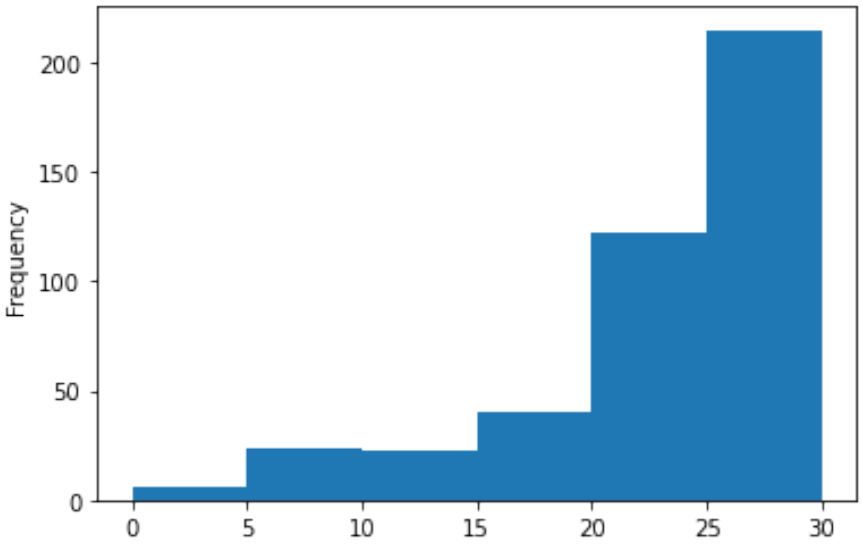
In [20]:
```
plt.figure(figsize=(12,4))
sns.violinplot(x=df['Working_Hours'],y= df['Employment'])
```

Out[20]: <AxesSubplot:xlabel='Working_Hours', ylabel='Employment'>



In [22]:
```
df[df['Employment']=='Employed part-time']['Working_Hours'].plot(kinc
```

Out[22]: <AxesSubplot:ylabel='Frequency'>


Frequency
200
150
100
50
0
0
5
10
15
20
25
30