For Credit card dataset perform the following

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

In [2]:
```
credit_df = pd.read_csv("credit_dataset.csv")
credit_df
```

Out[2]:

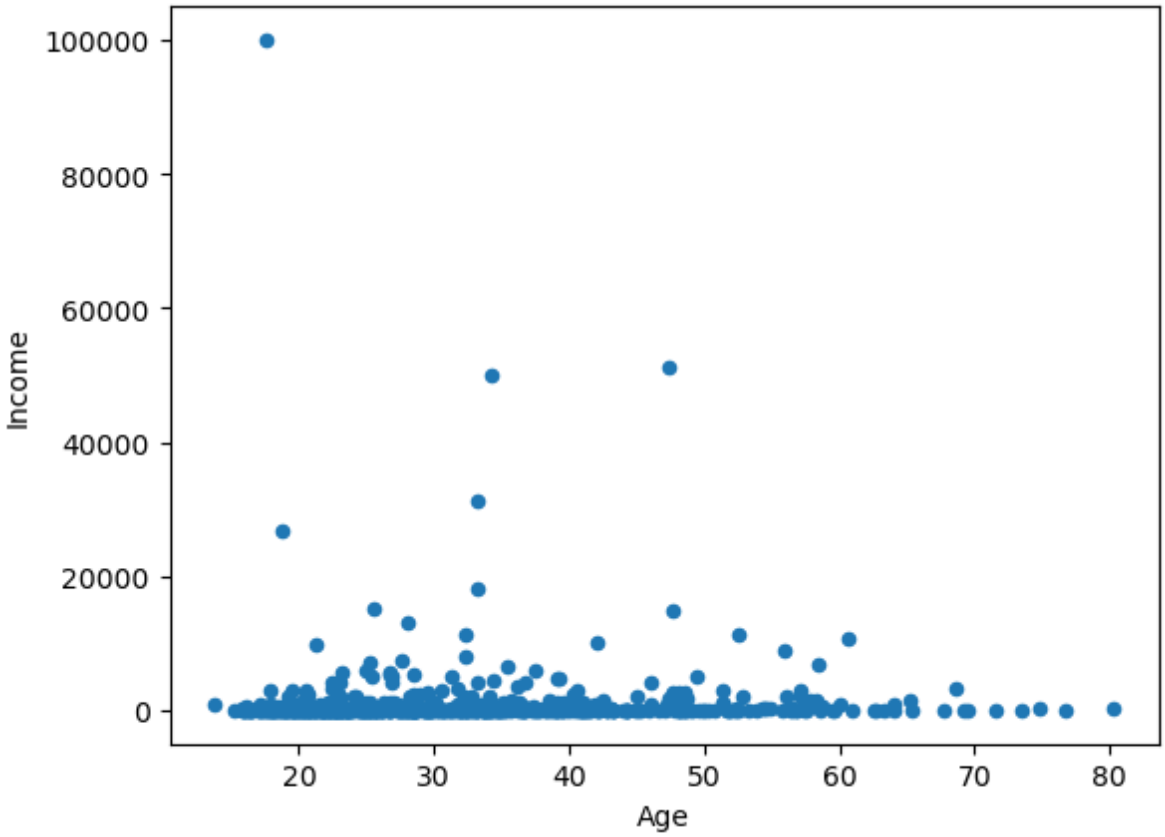
	Gender	Age	Debt	Married	BankCustomer	Industry	Ethnicity	YearsEmployed	I
0	1	30.83	0.000	1	1	Industrials	White	1.25	
1	0	58.67	4.460	1	1	Materials	Black	3.04	
2	0	24.50	0.500	1	1	Materials	Black	1.50	
3	1	27.83	1.540	1	1	Industrials	White	3.75	
4	1	20.17	5.625	1	1	Industrials	White	1.71	
...	...	...	...	...	...	...	...	...	
685	1	21.08	10.085	0	0	Education	Black	1.25	
686	0	22.67	0.750	1	1	Energy	White	2.00	
687	0	25.25	13.500	0	0	Healthcare	Latino	2.00	
688	1	17.92	0.205	1	1	ConsumerStaples	White	0.04	
689	1	35.00	3.375	1	1	Energy	Black	8.29	

690 rows × 16 columns

1.spot outliers in Income using bivariate plot

In [7]:
```
credit_df.plot('Age','Income',kind='scatter',marker='o')
```

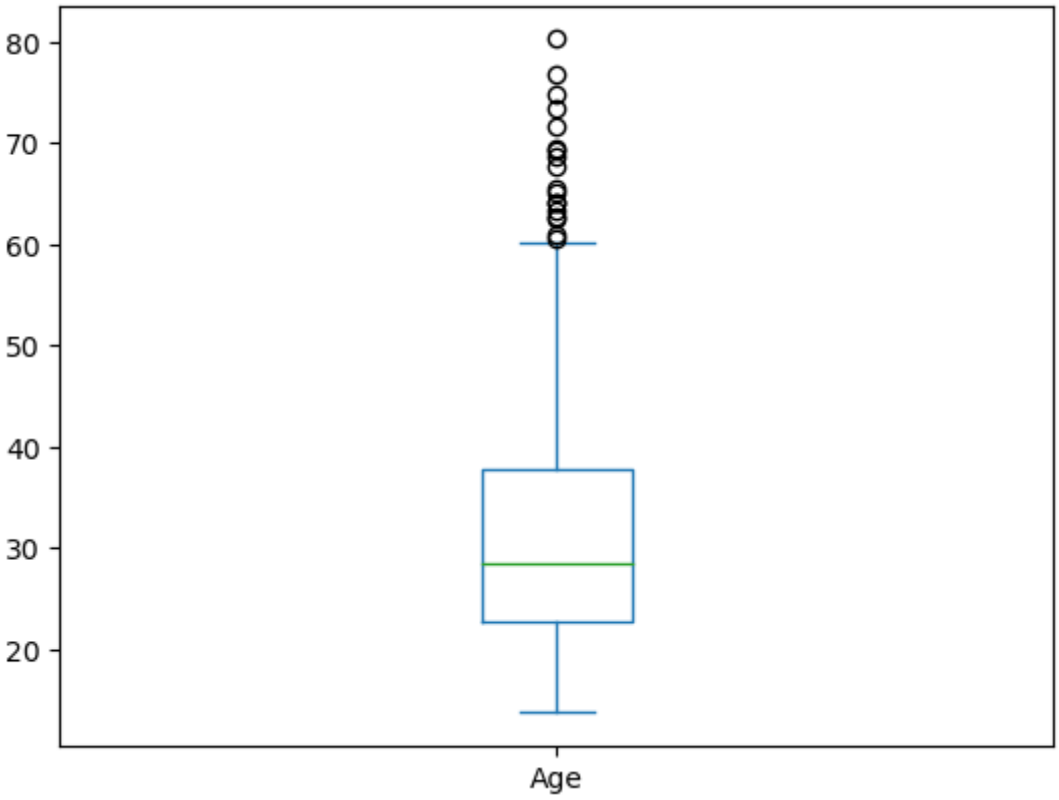
Out[7]:
```
<AxesSubplot:xlabel='Age', ylabel='Income'>
```



2.Spot outliers in any one feature using box plot

In [8]:
```
credit_df['Age'].plot(kind='box')
```

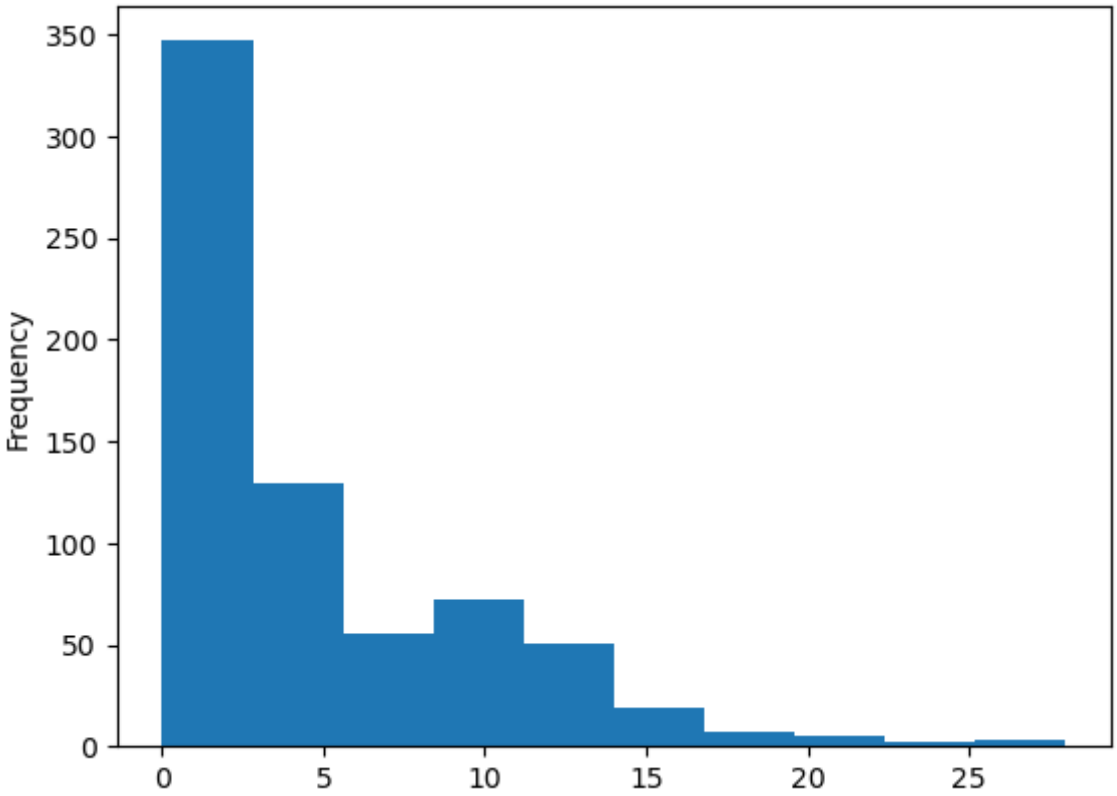
Out[8]:
```
<AxesSubplot:>
```



3.spot outliers using histogram plot

In [9]:
```
credit_df['Debt'].plot(kind='hist')
```

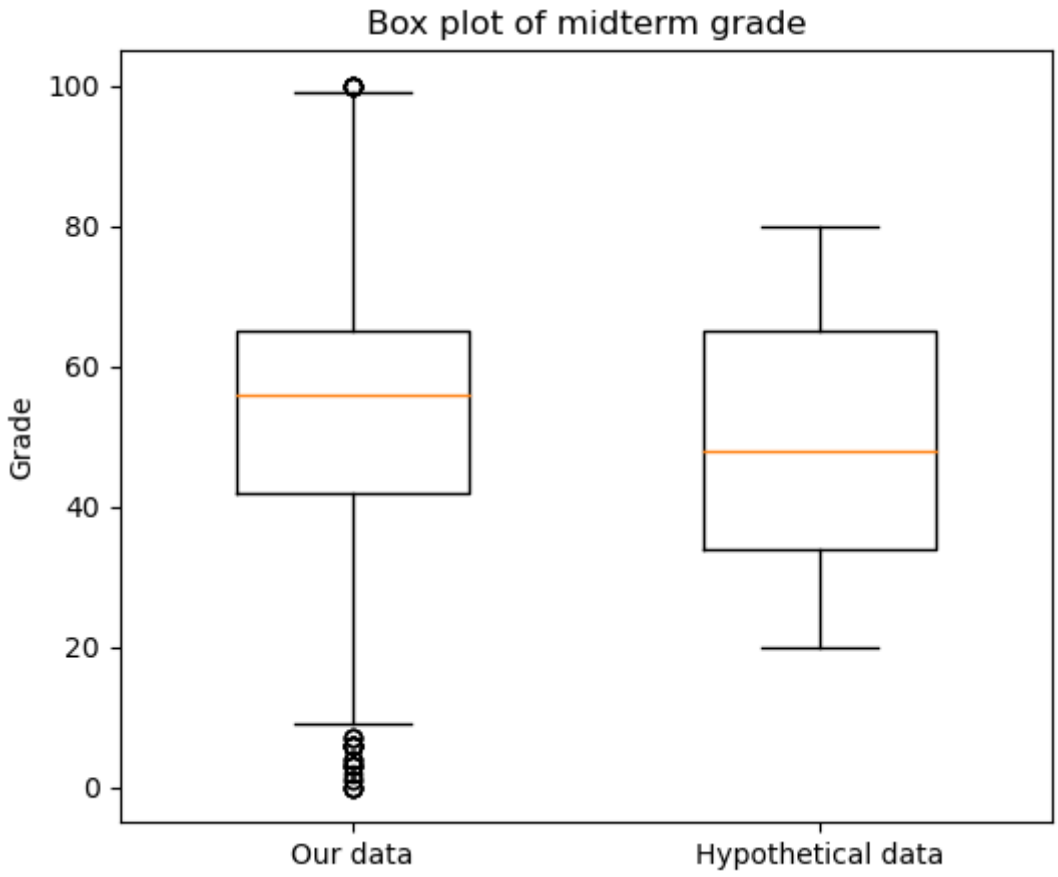
Out[9]:
```
<AxesSubplot:ylabel='Frequency'>
```



4.Detect outliers in any one feature using IQR method

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(102)
grades = np.concatenate([[50,52,53,55,56,60,61,62,65,67]*20,
np.random.randint(0, 101, size=300)])
Q1 = np.percentile(grades , 25)
Q3 = np.percentile(grades , 75)
Q1,Q3 = np.percentile(grades , [25,75])
IQR = Q3 - Q1
ul = Q3+1.5*IQR
ll = Q1-1.5*IQR
outliers = grades[(grades > ul) | (grades < ll)]
print(outliers)
fig = plt.figure(figsize=(6,5))
hypo = np.random.randint(20, 81, size=500)
plt.boxplot([grades, hypo], widths=0.5)
plt.xticks([1,2],['Our data', 'Hypothetical data'])
plt.ylabel('Grade')
plt.title('Box plot of midterm grade')
plt.show()
```

```
[  0   7   4   3   0   4   2   7   6 100   1   3   0   3 100 100 100 100
   4   0   3   6   6   6 100   7   6 100 100   6   3   6   1   6   0]
```



5.Detect outliers using z-score method

In [12]:
```
import numpy as np
data = [1, 2, 2, 2, 3, 1, 1, 15, 2, 2, 2, 3, 1, 1, 2]
mean = np.mean(data)
std = np.std(data)
print('mean of the dataset is', mean)
print('std. deviation is', std)
threshold = 3
outlier = []
for i in data:
    z = (i-mean)/std
    if z > threshold:
        outlier.append(i)
print('outlier in dataset of Z score is', outlier)
```

```
mean of the dataset is 2.6666666666666665
std. deviation is 3.3598941782277745
outlier in dataset of Z score is [15]
```

6.Treat outliers by Deleting observations

In [20]:
```
q1 = credit_df["Age"].quantile(0.25)
q3 = credit_df['Age'].quantile(0.75)
iqr = q3-q1
upper_bound = q3+(1.5*iqr)
lower_bound = q1-(1.5*iqr)
```

In [21]:
```
upperIndex = credit_df[credit_df['Age']>upper_bound].index
credit_df.drop(upperIndex,inplace=True)
lowerIndex = credit_df[credit_df['Age']<lower_bound].index
```

```
credit_df.drop(lowerIndex,inplace=True)
credit_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 553 entries, 0 to 689
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Gender          553 non-null    int64  
 1   Age             553 non-null    float64
 2   Debt            553 non-null    float64
 3   Married         553 non-null    int64  
 4   BankCustomer    553 non-null    int64  
 5   Industry        553 non-null    object 
 6   Ethnicity       553 non-null    object 
 7   YearsEmployed   553 non-null    float64
 8   PriorDefault    553 non-null    int64  
 9   Employed        553 non-null    int64  
 10  CreditScore     553 non-null    int64  
 11  DriversLicense  553 non-null    int64  
 12  Citizen         553 non-null    object 
 13  ZipCode         553 non-null    int64  
 14  Income          553 non-null    int64  
 15  Approved        553 non-null    int64  
dtypes: float64(3), int64(10), object(3)
memory usage: 73.4+ KB
```

7.Treat outliers using imputations

imputations using mean

In [22]:
```
m = np.mean(credit_df['Age'])
print('mean:',m)
for i in credit_df['Age']:
    if i<lower_bound or i>upper_bound :
        titanic_df['Age'] = titanic_df['Age'].replace(i,m)
```

```
mean: 29.347486437613018
```

imputations using median

In [24]:
```
m = credit_df['Age'].median()
print("median",m)
for i in credit_df['Age']:
    if i<lower_bound or i>upper_bound :
        credit_df['Age'] = credit_df['Age'].replace(i,m)
```

```
median 27.58
```

imputations using zero

In [25]:
```
for i in credit_df['Age']:
    if i<lower_bound or i>upper_bound :
        credit_df['Age'] = credit_df['Age'].replace(i,0)
```