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]:

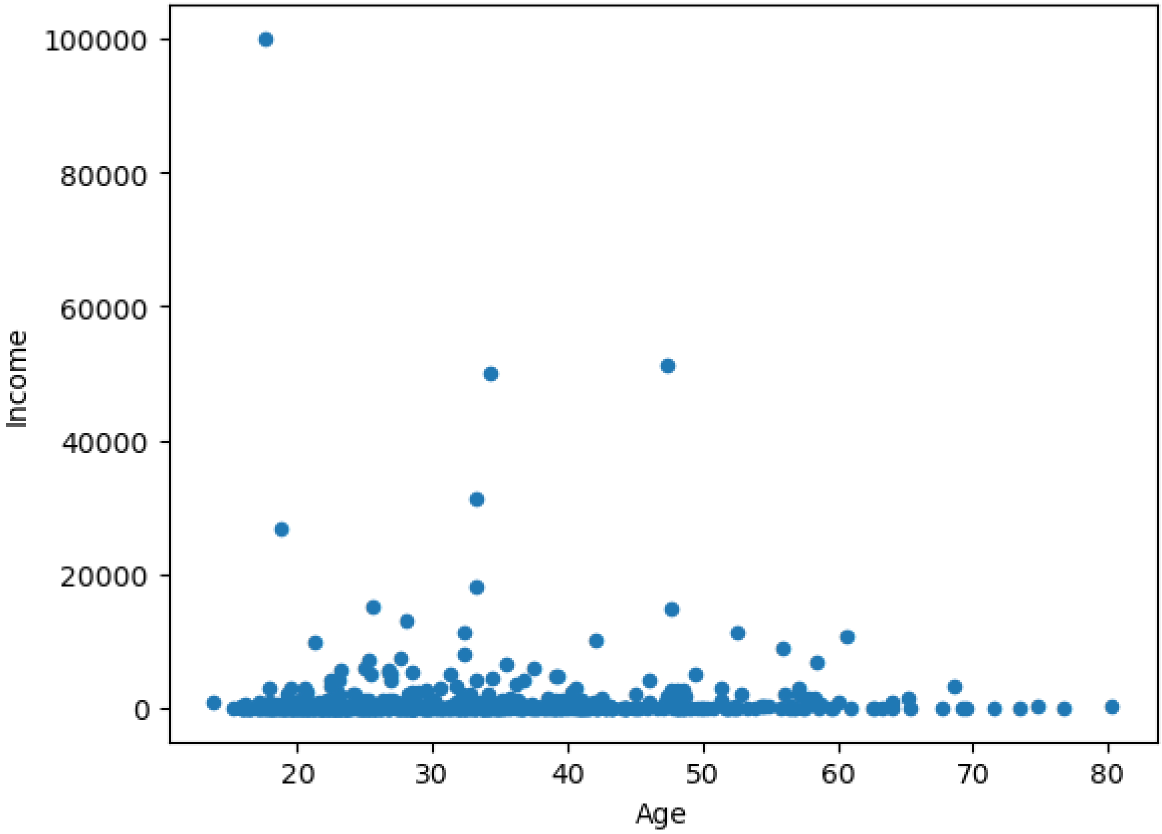
	Gender	Age	Debt	Married	BankCustomer	Industry	Ethnicity	YearsEmployed	I
0	1	30.83	0.000	1	1	Industrials	White	1.25	
1	0	58.67	4.460	1	1	Materials	Black	3.04	
2	0	24.50	0.500	1	1	Materials	Black	1.50	
3	1	27.83	1.540	1	1	Industrials	White	3.75	
4	1	20.17	5.625	1	1	Industrials	White	1.71	
...	...	...	...	...	...	...	...	...	
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')
```

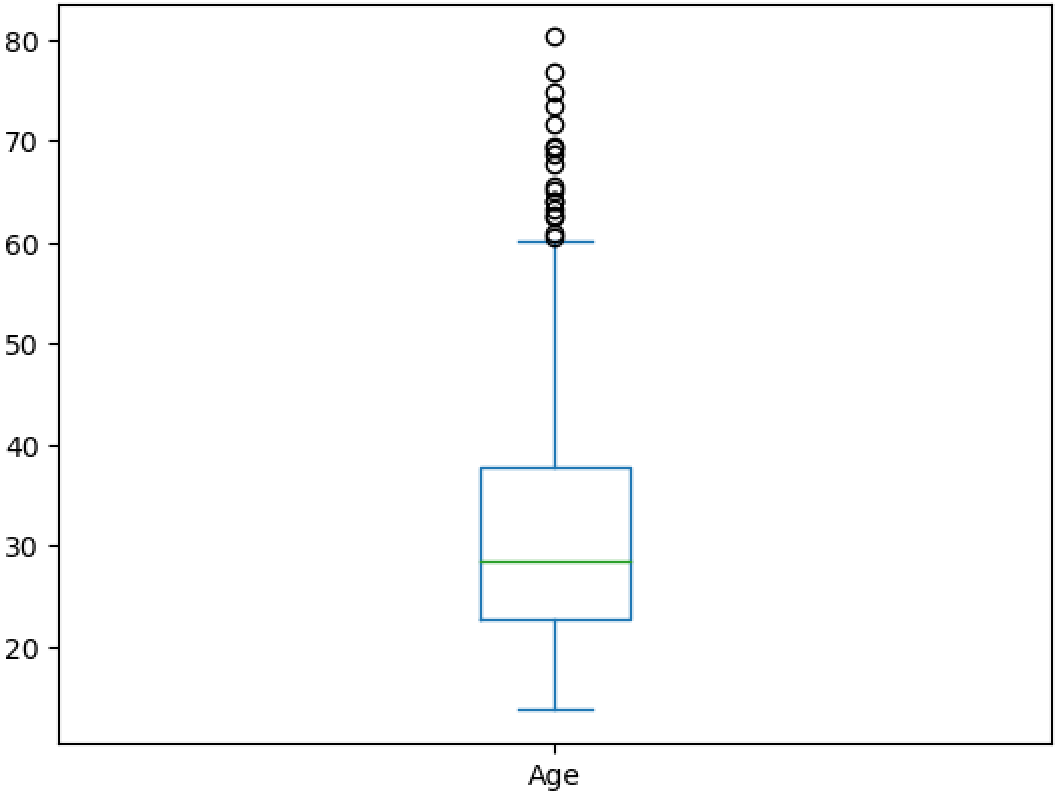
Out[7]:
```
<AxesSubplot:xlabel='Age', ylabel='Income'>
```



2.Spot outliers in any one feature using box plot

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

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



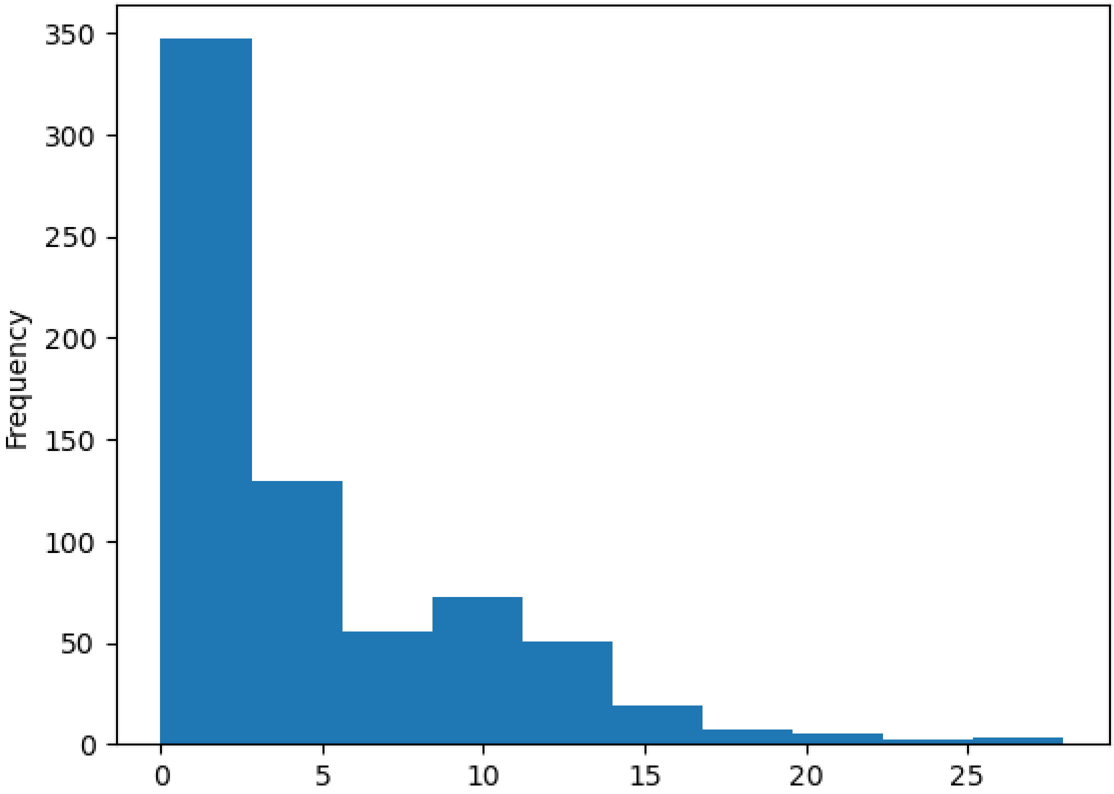
3.spot outliers using histogram plot

In [9]:

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

Out[9]:

```
<AxesSubplot:ylabel='Frequency'>
```

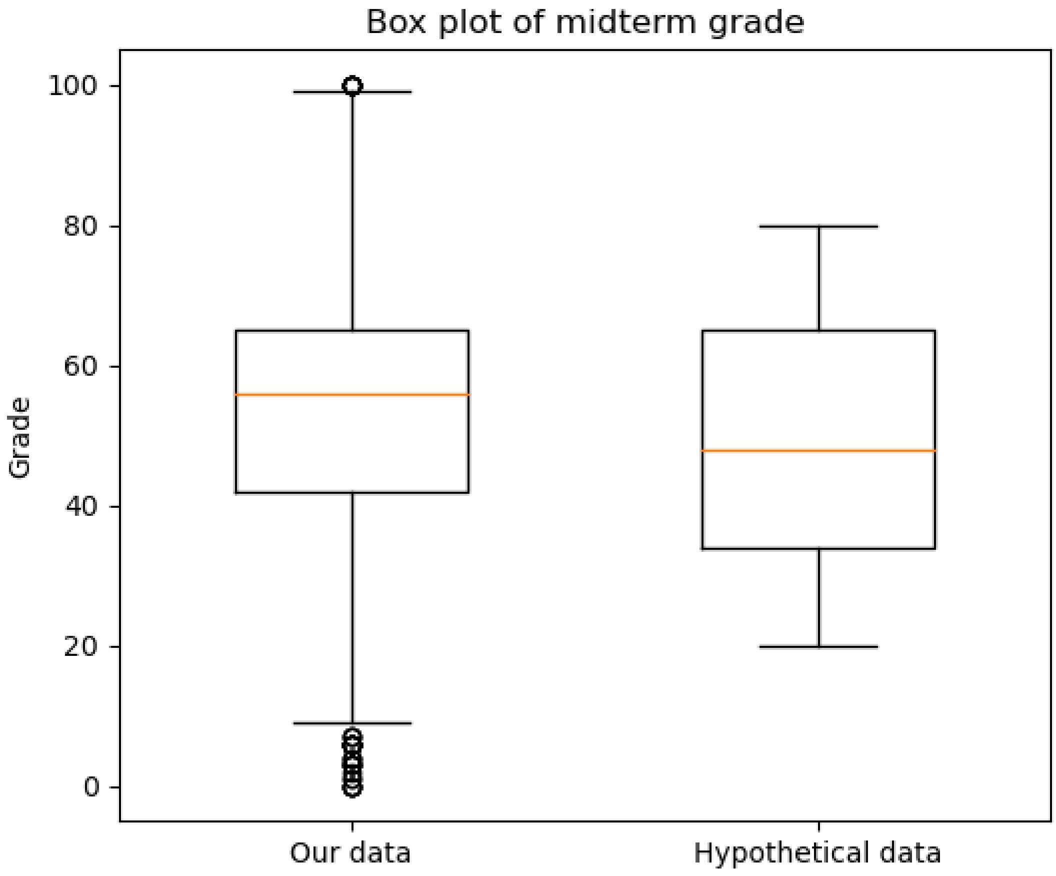


4.Detect outliers in any one feature using IQR method

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)
```

Univariate,Bivariate and Multivariate Analysis

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import math
```

In [3]:

```
card_approval_df=pd.read_csv('clean_dataset.csv')
print(card_approval_df.head())
```

```
   Gender    Age   Debt  Married  BankCustomer     Industry Ethnicity  \
0       1  30.83  0.000        1             1  Industrials     White   
1       0  58.67  4.460        1             1    Materials     Black   
2       0  24.50  0.500        1             1    Materials     Black   
3       1  27.83  1.540        1             1  Industrials     White   
4       1  20.17  5.625        1             1  Industrials     White   

   YearsEmployed  PriorDefault  Employed  CreditScore  DriversLicense  \
0           1.25             1         1            1               0   
1           3.04             1         1            6               0   
2           1.50             1         0            0               0   
3           3.75             1         1            5               1   
4           1.71             1         0            0               0   

        Citizen  ZipCode  Income  Approved  
0       ByBirth      202       0         1  
1       ByBirth       43     560         1  
2       ByBirth      280     824         1  
3       ByBirth      100       3         1  
4  ByOtherMeans      120       0         1  
```

In [4]:

```
print(card_approval_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 690 entries, 0 to 689
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Gender          690 non-null    int64  
 1   Age             690 non-null    float64
 2   Debt            690 non-null    float64
 3   Married         690 non-null    int64  
 4   BankCustomer    690 non-null    int64  
 5   Industry        690 non-null    object 
 6   Ethnicity       690 non-null    object 
 7   YearsEmployed   690 non-null    float64
 8   PriorDefault    690 non-null    int64  
 9   Employed        690 non-null    int64  
 10  CreditScore     690 non-null    int64  
 11  DriversLicense  690 non-null    int64  
 12  Citizen         690 non-null    object 
 13  ZipCode         690 non-null    int64  
 14  Income          690 non-null    int64  
 15  Approved        690 non-null    int64  
dtypes: float64(3), int64(10), object(3)
memory usage: 86.4+ KB
None
```

In [5]:

```
card_approval_df.duplicated().sum()
```

Out[5]:

```
0
```

Univariate Analysis of continuous Variables

In [6]:

```
card_approval_df[['Age','Debt','YearsEmployed','CreditScore','Income']].describe()
```

Out[6]:

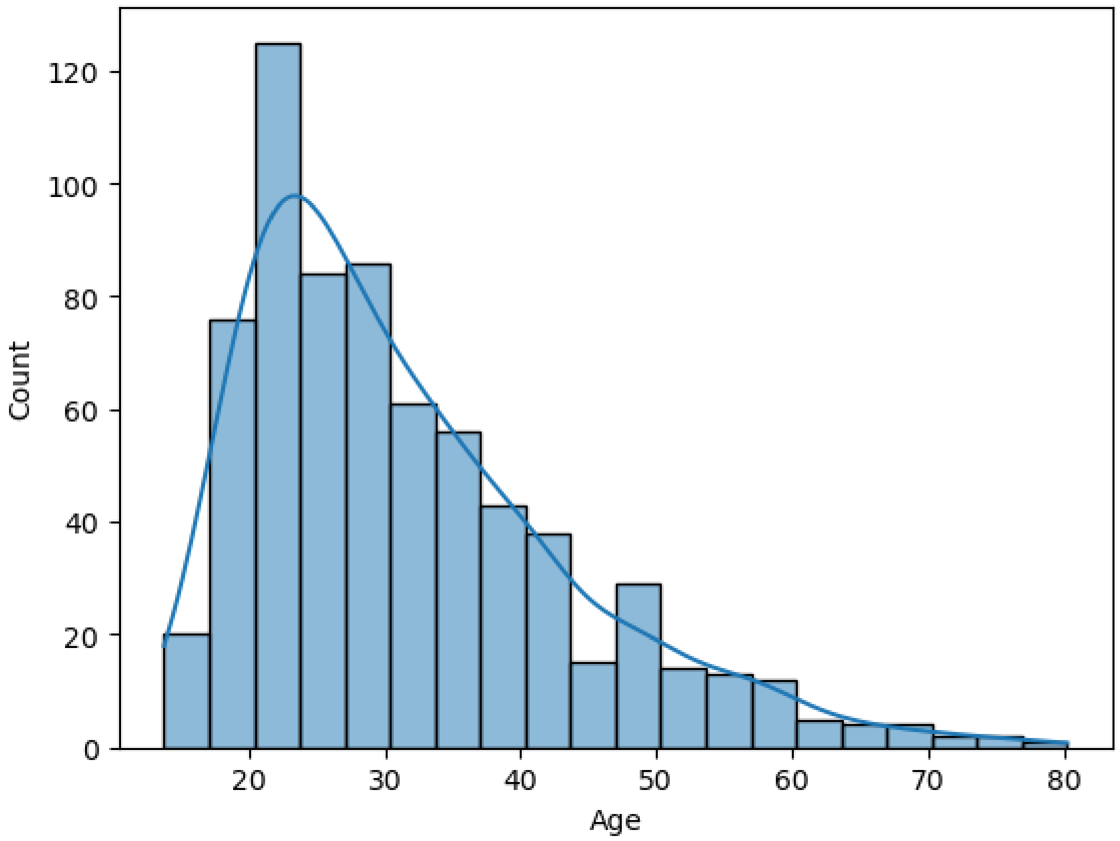
	Age	Debt	YearsEmployed	CreditScore	Income
count	690.000000	690.000000	690.000000	690.00000	690.000000
mean	31.514116	4.758725	2.223406	2.40000	1017.385507
std	11.860245	4.978163	3.346513	4.86294	5210.102598
min	13.750000	0.000000	0.000000	0.00000	0.000000
25%	22.670000	1.000000	0.165000	0.00000	0.000000
50%	28.460000	2.750000	1.000000	0.00000	5.000000
75%	37.707500	7.207500	2.625000	3.00000	395.500000
max	80.250000	28.000000	28.500000	67.00000	100000.000000

In [7]:

```
sns.histplot(card_approval_df.Age,kde=True)
```

Out[7]:

```
<AxesSubplot:xlabel='Age', ylabel='Count'>
```



Univariate Analysis of categorical Variables

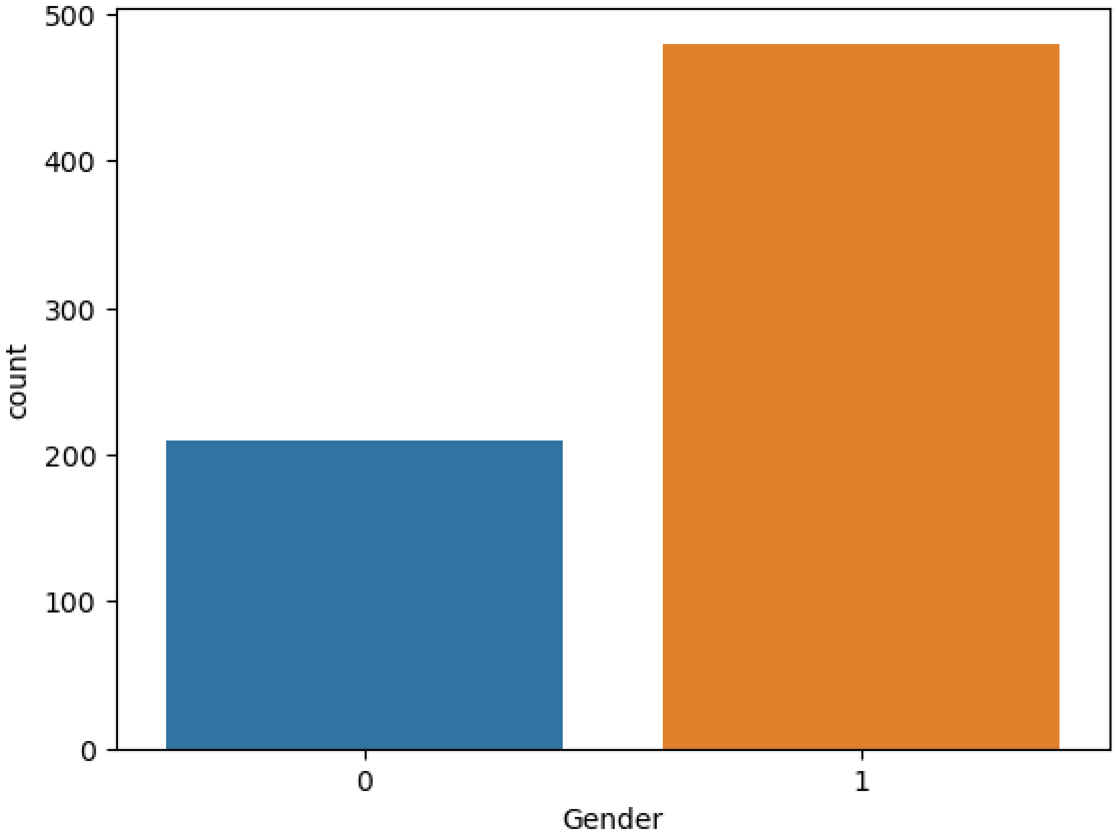
In [8]:

```
sns.countplot(card_approval_df.Gender)
```

```
C:\Users\gptkgf\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only v
alid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[8]:

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



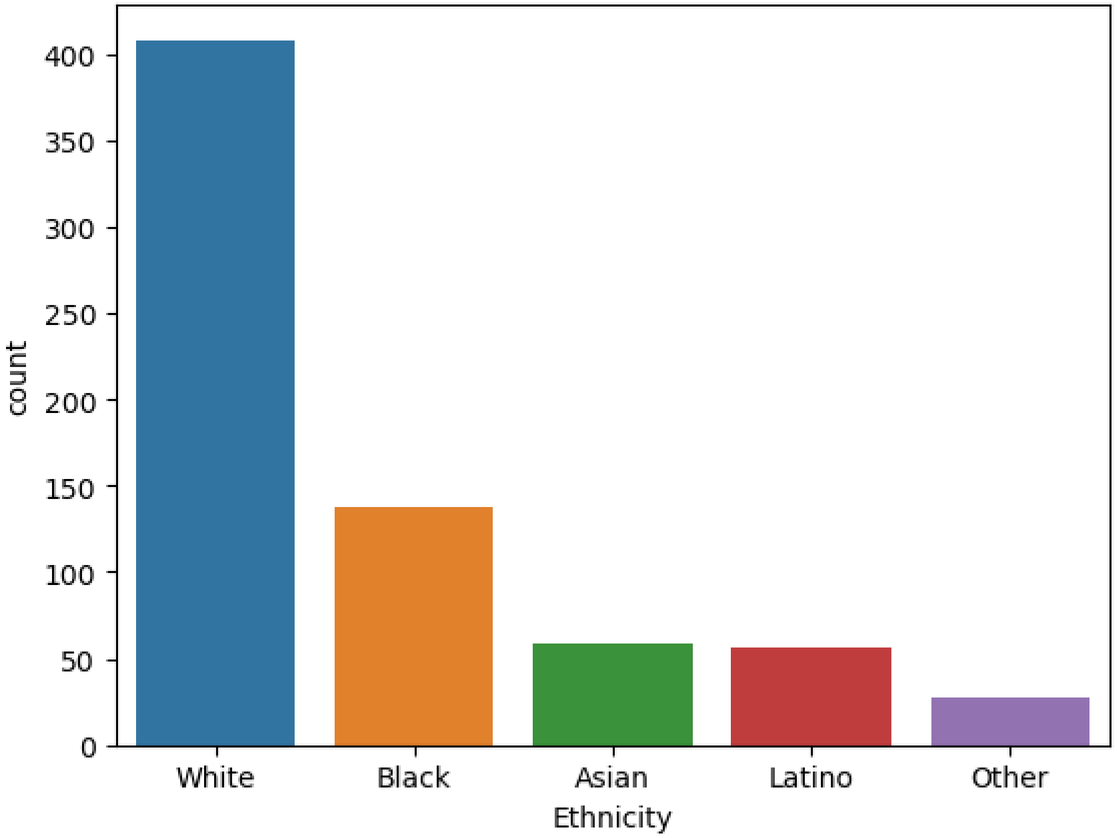
In [9]:

```
sns.countplot(card_approval_df.Ethnicity)
```

```
C:\Users\gptkgf\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only v
alid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]:

```
<AxesSubplot:xlabel='Ethnicity', ylabel='count'>
```



Bivariate analysisof continuous variable

In [10]:

```
card_approval_df[['Age','Debt','YearsEmployed','CreditScore','Income']].corr()
```

Out[10]:

	Age	Debt	YearsEmployed	CreditScore	Income
Age	1.000000	0.202177	0.391464	0.187327	0.018719
Debt	0.202177	1.000000	0.298902	0.271207	0.123121
YearsEmployed	0.391464	0.298902	1.000000	0.322330	0.051345
CreditScore	0.187327	0.271207	0.322330	1.000000	0.063692
Income	0.018719	0.123121	0.051345	0.063692	1.000000

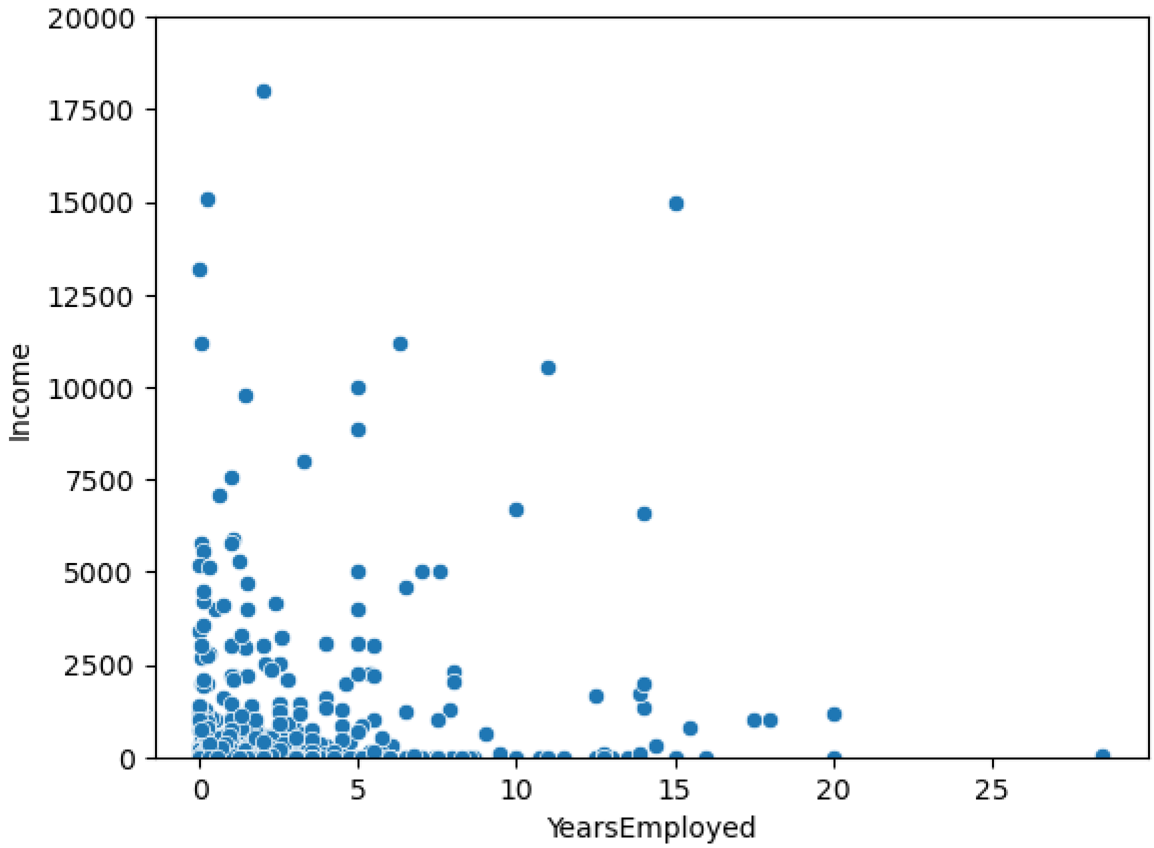
In [12]:

```
sns.scatterplot(card_approval_df.YearsEmployed,card_approval_df.Income)
plt.ylim(0,20000)
```

```
C:\Users\gptkgf\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variables as keyword args: x, y. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[12]:

```
(0.0, 20000.0)
```



Bivariate Analysis of Categorical Variables vs Continuous Variables

In [13]:

```
card_approval_df.groupby(by='Approved').agg('mean')[['Age','Debt','YearsEmployed',
```

Out[13]:

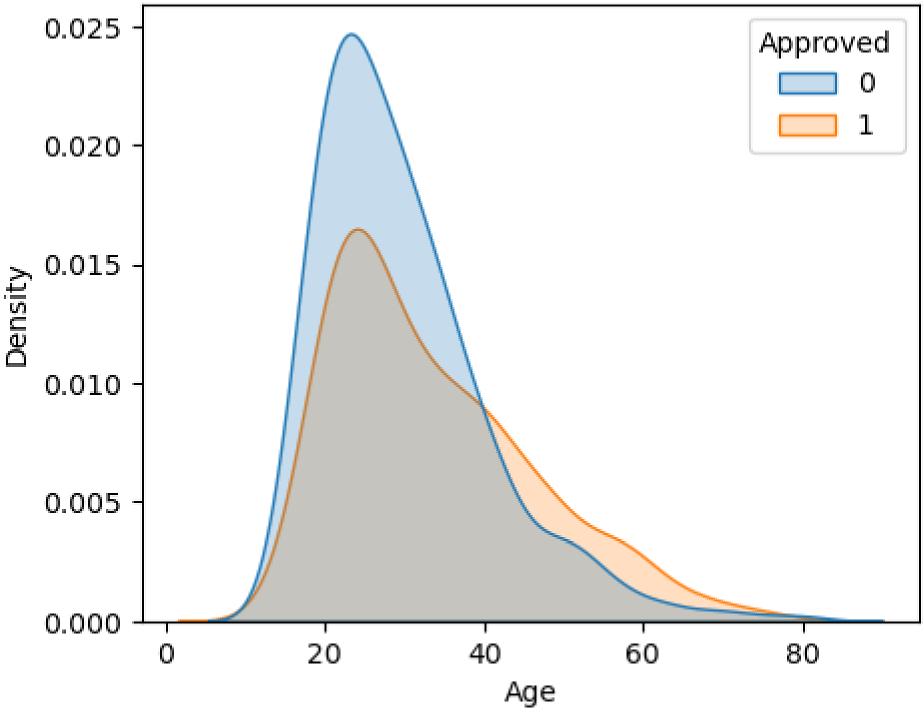
	Age	Debt	YearsEmployed	CreditScore	Income
Approved					
0	29.773029	3.839948	1.257924	0.631854	198.605744
1	33.686221	5.904951	3.427899	4.605863	2038.859935

In [19]:

```
plt.figure(figsize=(5,4))
sns.kdeplot(data=card_approval_df,x='Age',hue='Approved',fill=True)
```

Out[19]:

```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



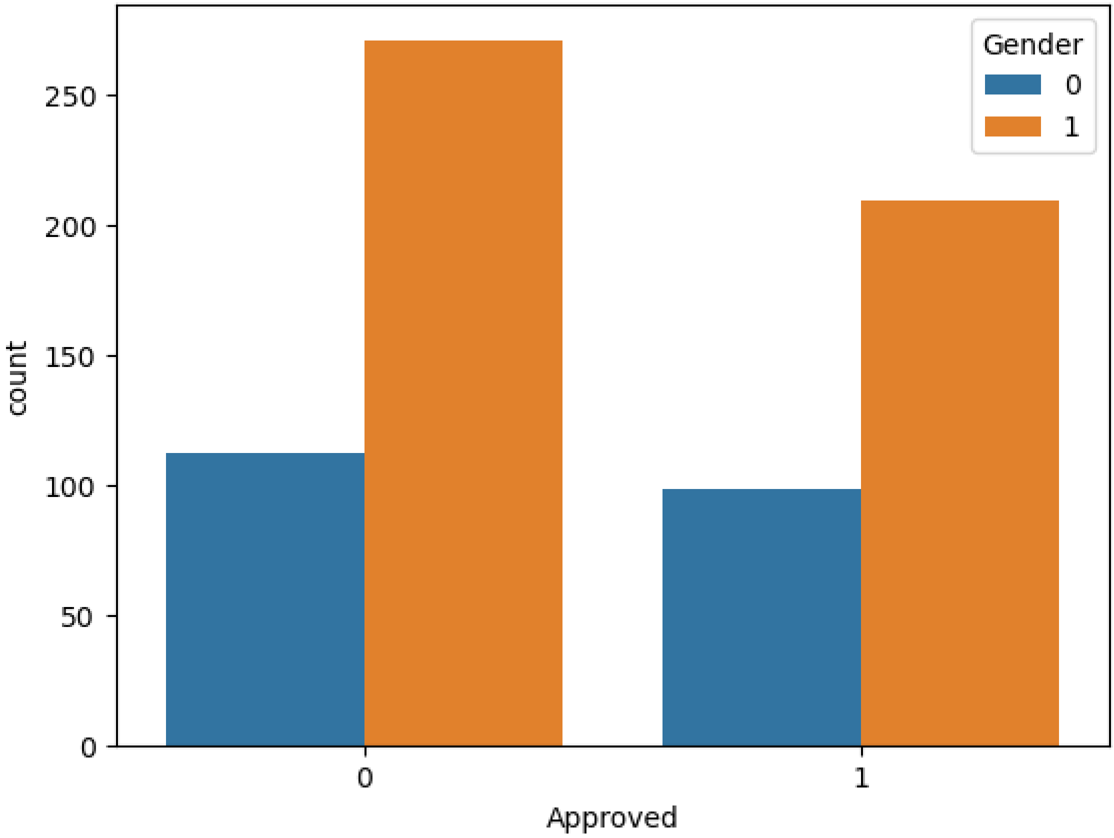
Bivariate Analysis of Categorical Variables vs Categorical Variables

In [20]:

```
sns.countplot(data=card_approval_df,x='Approved',hue='Gender')
```

Out[20]:

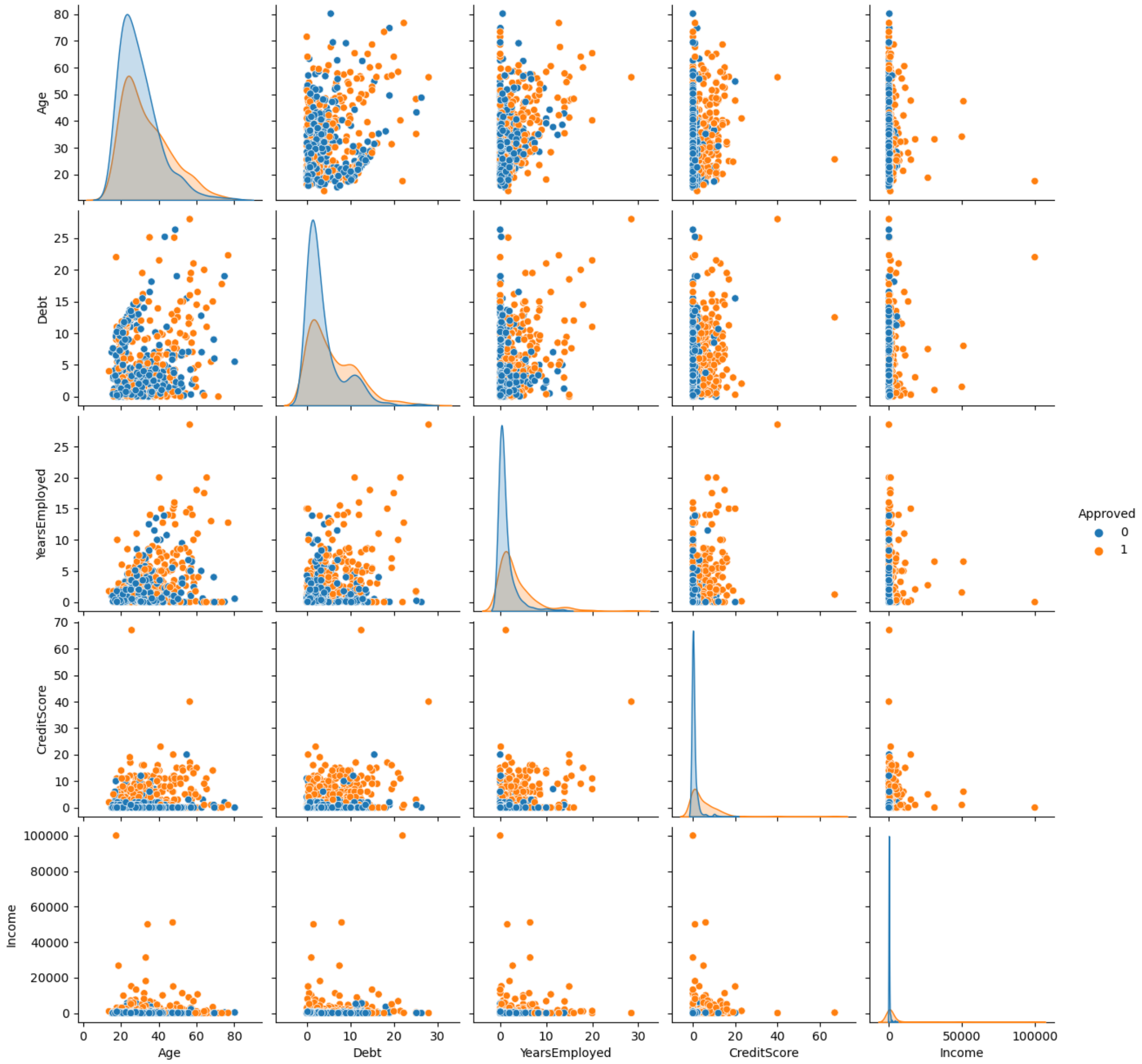
```
<AxesSubplot:xlabel='Approved', ylabel='count'>
```



Multivariate Analysis

In [21]:
```
sns.pairplot(data=card_approval_df[['Age','Debt','YearsEmployed','CreditScore','In
```

Out[21]:
```
<seaborn.axisgrid.PairGrid at 0x23b342f37c0>
```


Age
Debt
YearsEmployed
CreditScore
Income
Approved
● 0
● 1
80
70
60
50
40
30
20
25
15
10
5
0
100000
80000
60000
40000
20000
50000

```
In [ ]:
```

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

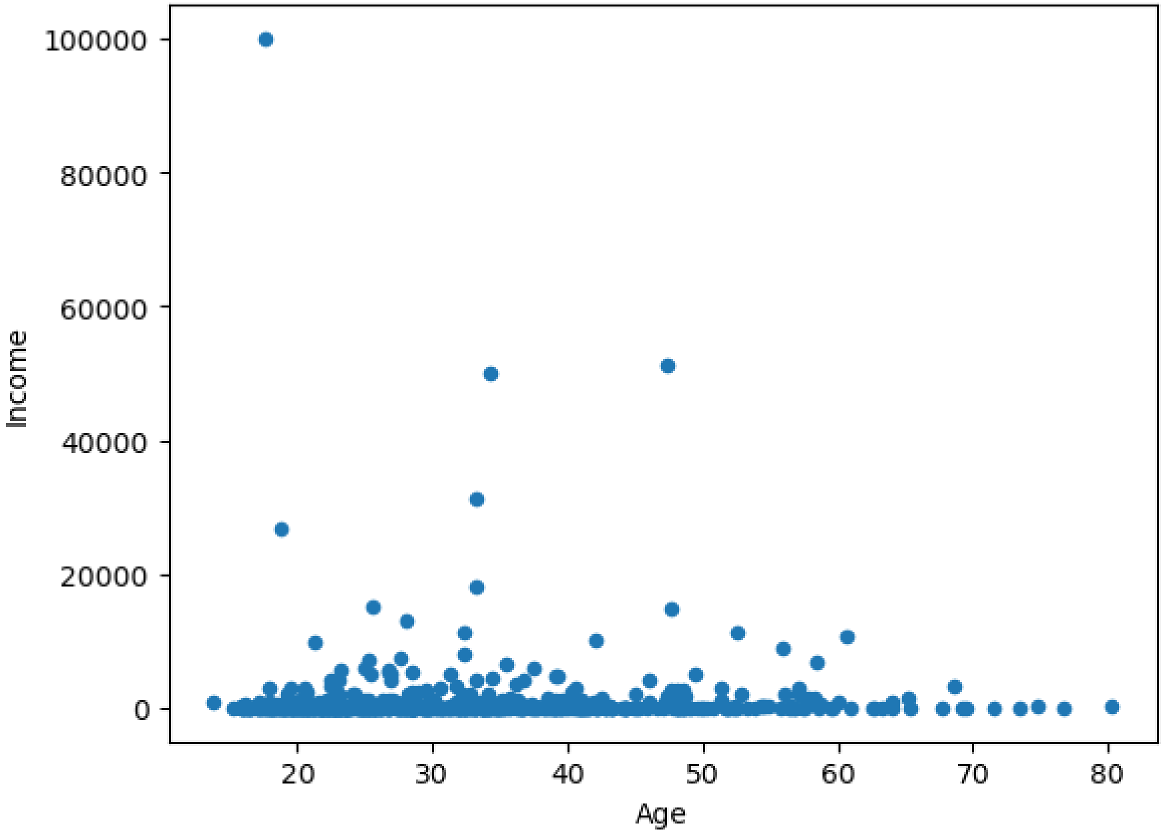
1.spot outliers in Income using bivariate plot

In [7]:

```
credit_df.plot('Age','Income',kind='scatter',marker='o')
```

Out[7]:

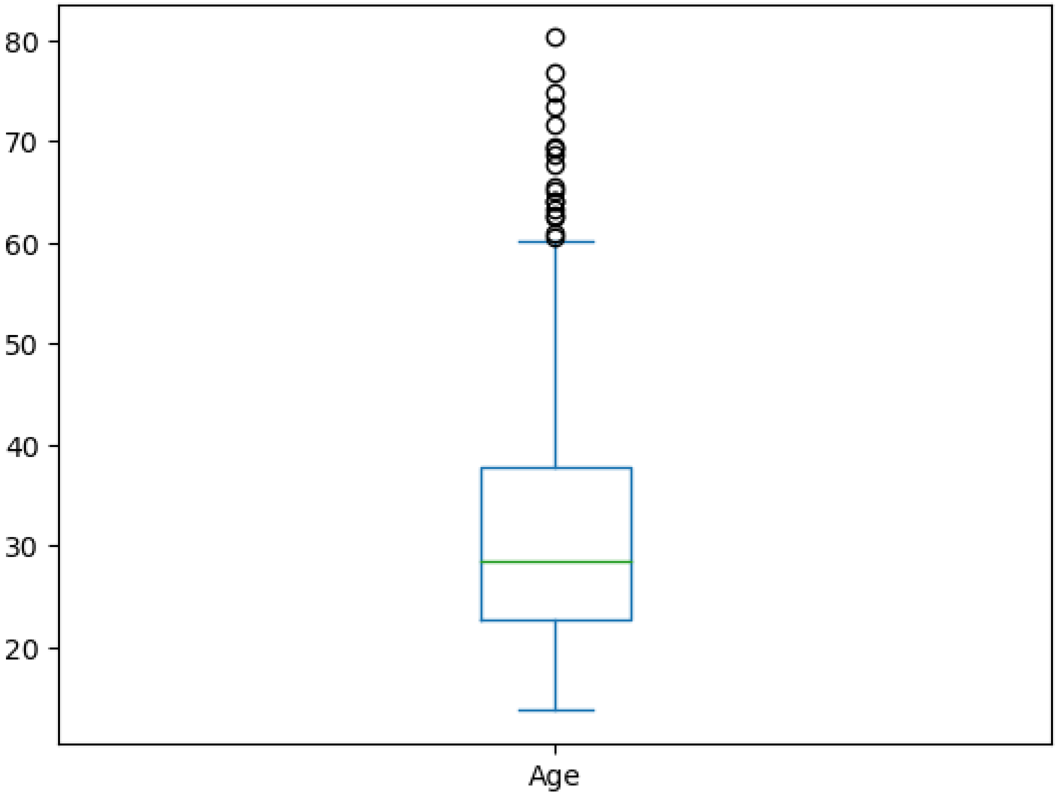
```
<AxesSubplot:xlabel='Age', ylabel='Income'>
```



2.Spot outliers in any one feature using box plot

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

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



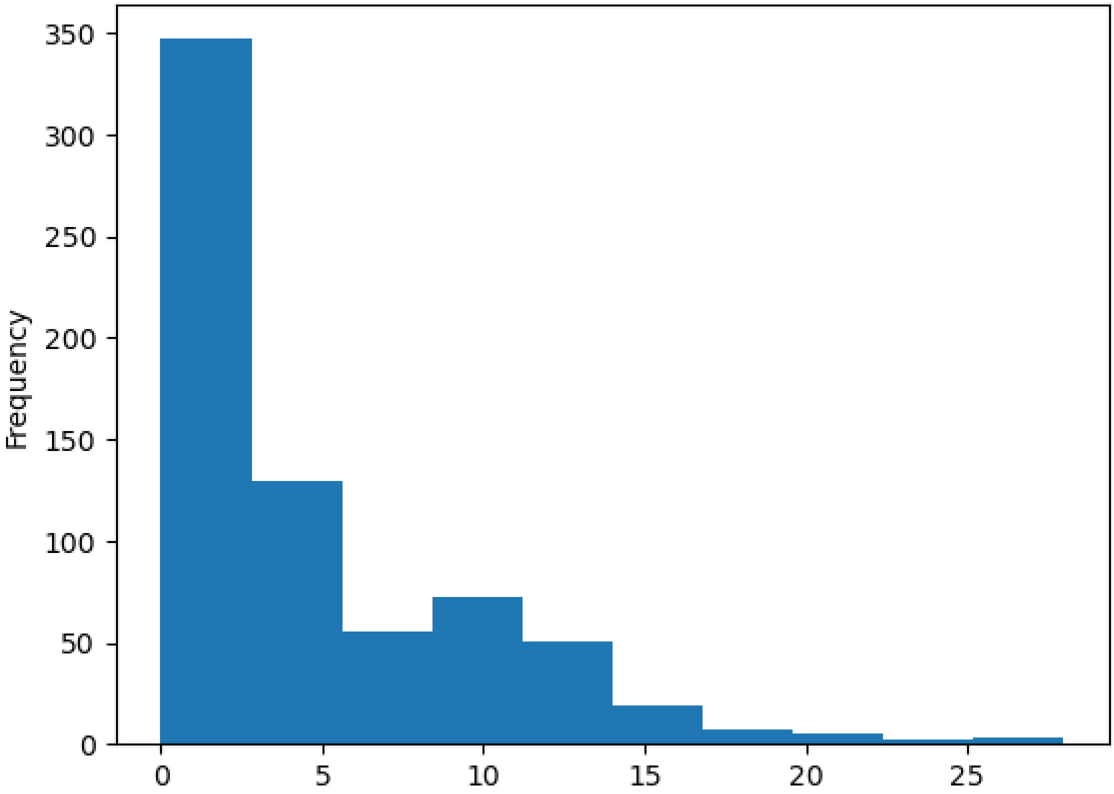
3.spot outliers using histogram plot

In [9]:

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

Out[9]:

```
<AxesSubplot:ylabel='Frequency'>
```

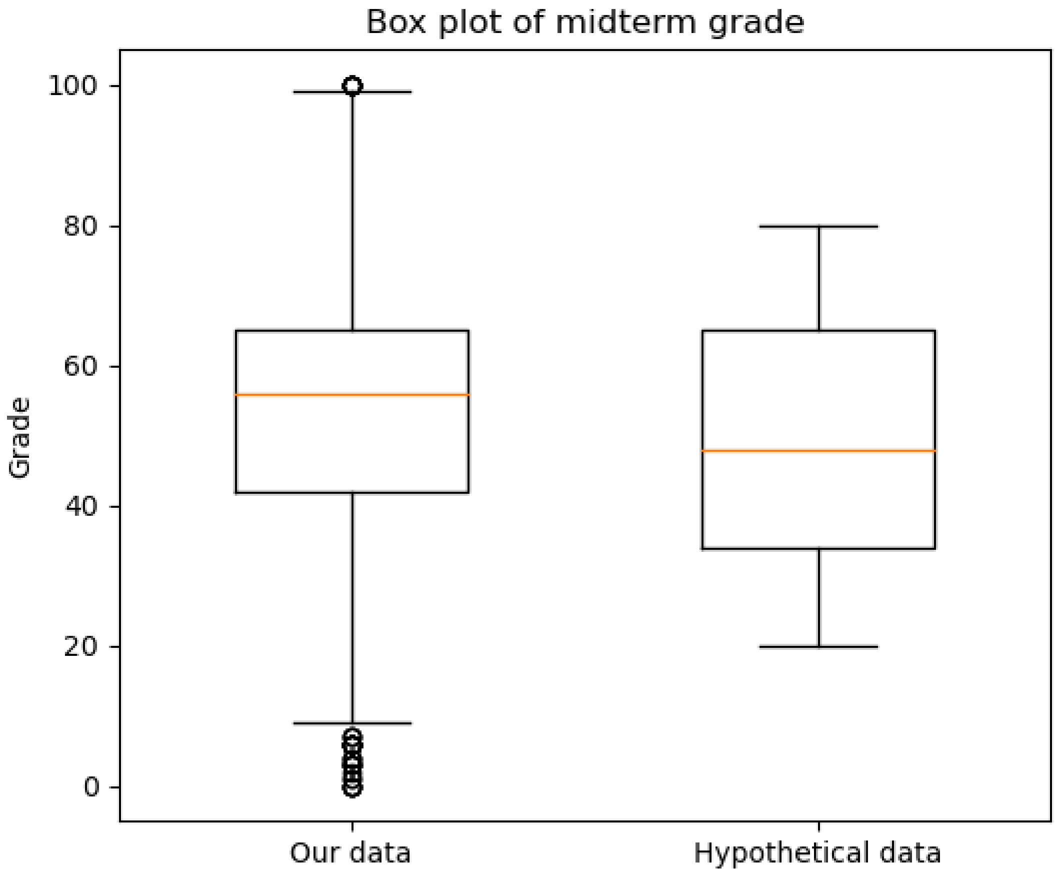


4.Detect outliers in any one feature using IQR method

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)
```

Univariate,Bivariate and Multivariate Analysis

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import math
```

In [3]:

```
card_approval_df=pd.read_csv('clean_dataset.csv')
print(card_approval_df.head())
```

```
   Gender    Age   Debt  Married  BankCustomer     Industry Ethnicity  \
0       1  30.83  0.000        1             1  Industrials     White   
1       0  58.67  4.460        1             1    Materials     Black   
2       0  24.50  0.500        1             1    Materials     Black   
3       1  27.83  1.540        1             1  Industrials     White   
4       1  20.17  5.625        1             1  Industrials     White   

   YearsEmployed  PriorDefault  Employed  CreditScore  DriversLicense  \
0           1.25             1         1            1               0   
1           3.04             1         1            6               0   
2           1.50             1         0            0               0   
3           3.75             1         1            5               1   
4           1.71             1         0            0               0   

        Citizen  ZipCode  Income  Approved  
0       ByBirth      202       0         1  
1       ByBirth       43     560         1  
2       ByBirth      280     824         1  
3       ByBirth      100       3         1  
4  ByOtherMeans      120       0         1  
```

In [4]:

```
print(card_approval_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 690 entries, 0 to 689
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Gender          690 non-null    int64  
 1   Age             690 non-null    float64
 2   Debt            690 non-null    float64
 3   Married         690 non-null    int64  
 4   BankCustomer    690 non-null    int64  
 5   Industry        690 non-null    object 
 6   Ethnicity       690 non-null    object 
 7   YearsEmployed   690 non-null    float64
 8   PriorDefault    690 non-null    int64  
 9   Employed        690 non-null    int64  
 10  CreditScore     690 non-null    int64  
 11  DriversLicense  690 non-null    int64  
 12  Citizen         690 non-null    object 
 13  ZipCode         690 non-null    int64  
 14  Income          690 non-null    int64  
 15  Approved        690 non-null    int64  
dtypes: float64(3), int64(10), object(3)
memory usage: 86.4+ KB
None
```

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

Out[5]:
```
0
```

Univariate Analysis of continuous Variables

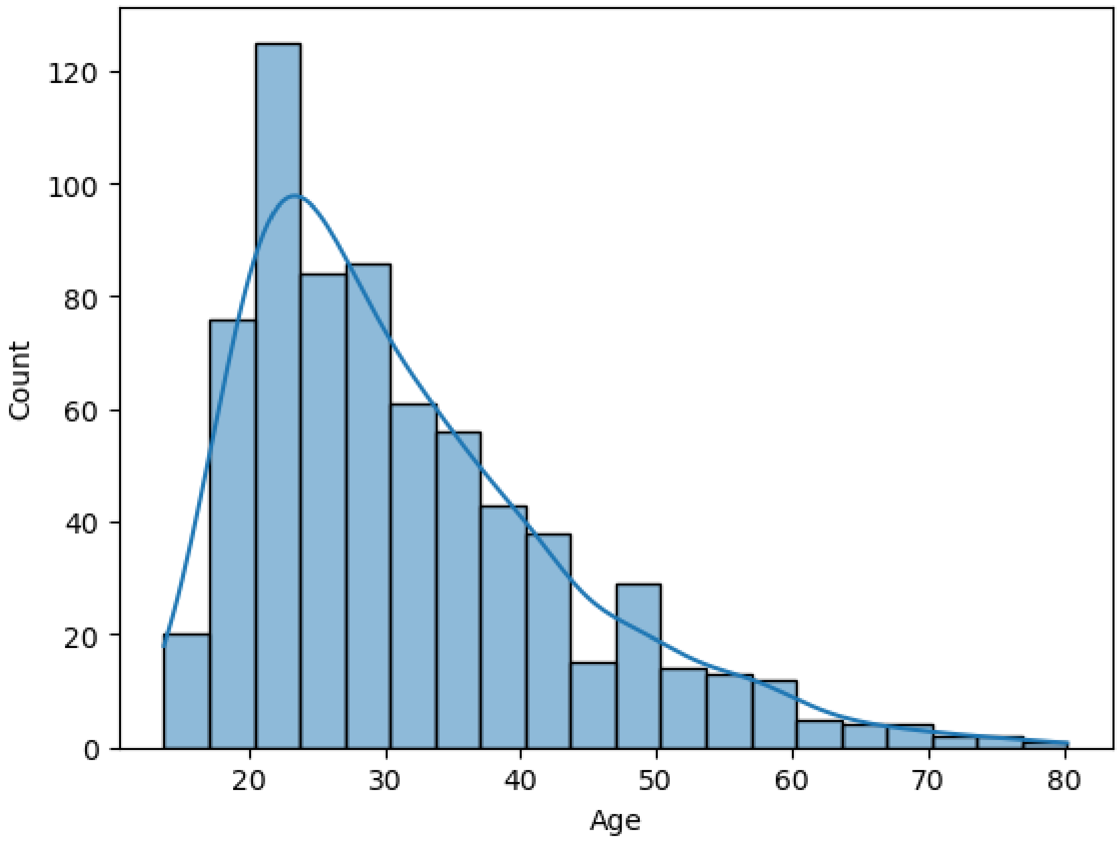
In [6]:
```
card_approval_df[['Age','Debt','YearsEmployed','CreditScore','Income']].describe()
```

Out[6]:

	Age	Debt	YearsEmployed	CreditScore	Income
count	690.000000	690.000000	690.000000	690.00000	690.000000
mean	31.514116	4.758725	2.223406	2.40000	1017.385507
std	11.860245	4.978163	3.346513	4.86294	5210.102598
min	13.750000	0.000000	0.000000	0.00000	0.000000
25%	22.670000	1.000000	0.165000	0.00000	0.000000
50%	28.460000	2.750000	1.000000	0.00000	5.000000
75%	37.707500	7.207500	2.625000	3.00000	395.500000
max	80.250000	28.000000	28.500000	67.00000	100000.000000

In [7]:
```
sns.histplot(card_approval_df.Age,kde=True)
```

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

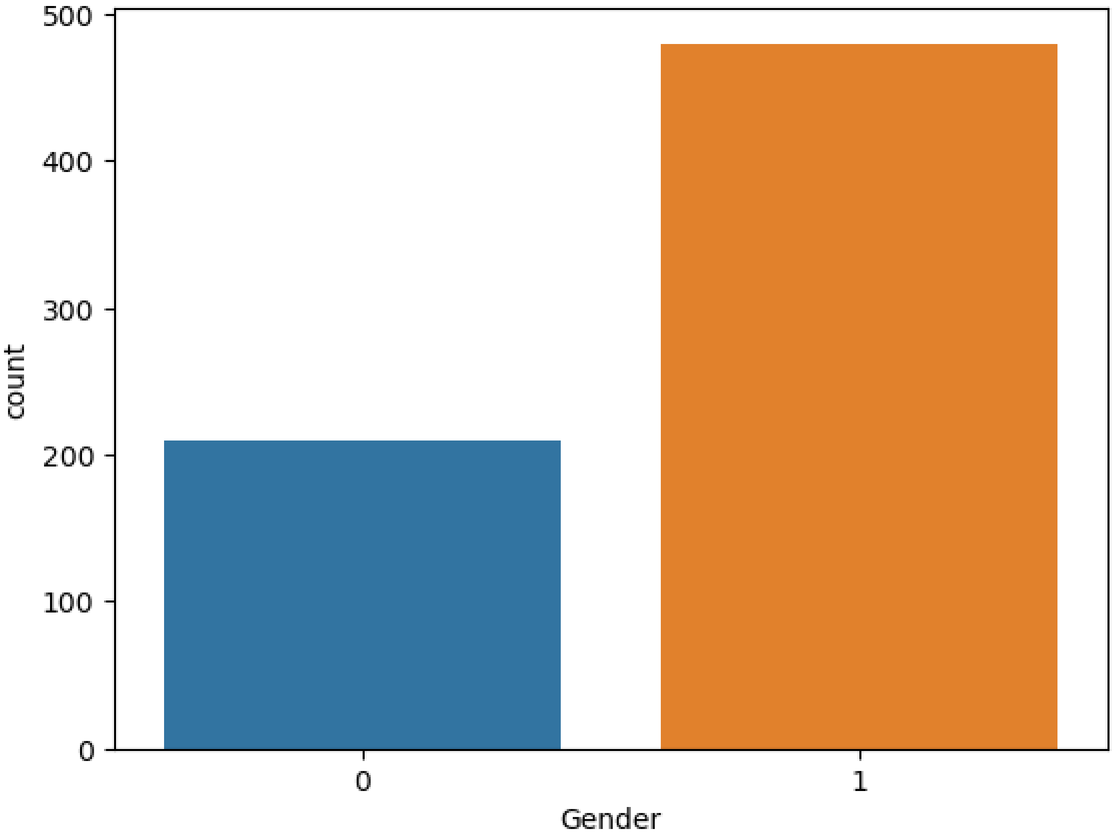


Univariate Analysis of categorical Variables

In [8]:
```
sns.countplot(card_approval_df.Gender)
```

```
C:\Users\gptkgf\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only v
alid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

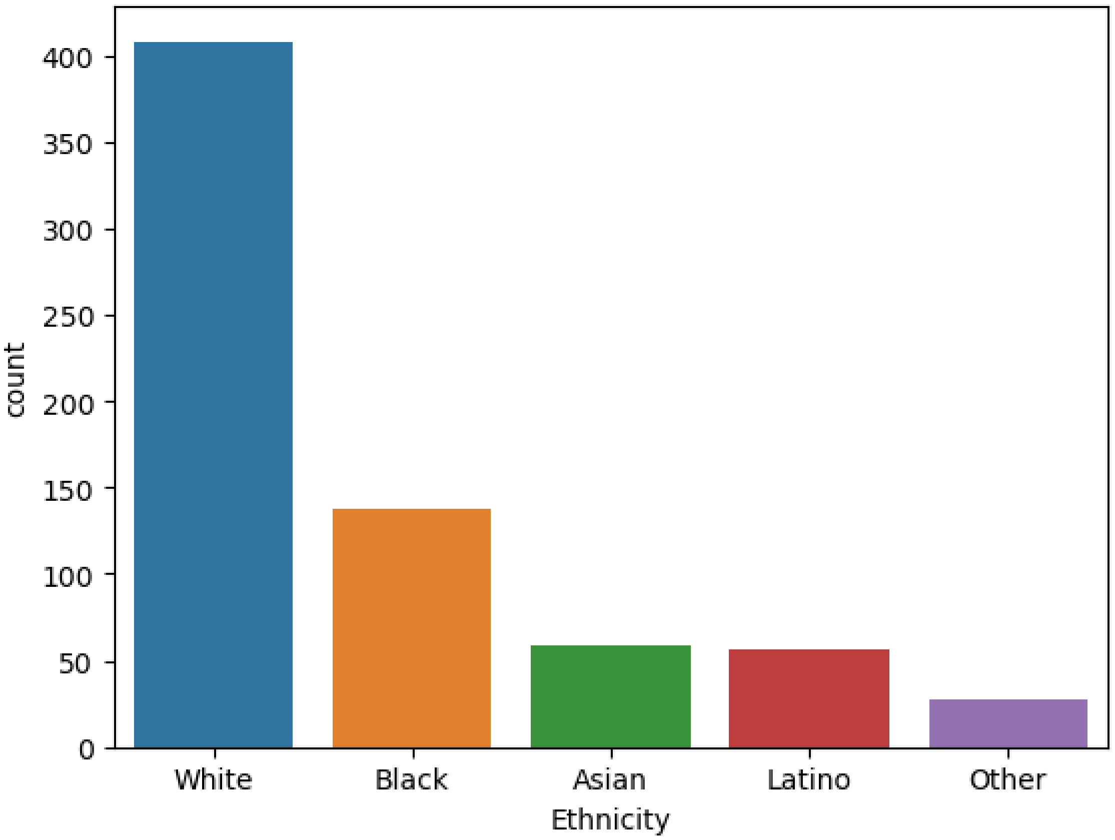
Out[8]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



In [9]:
```
sns.countplot(card_approval_df.Ethnicity)
```

```
C:\Users\gptkgf\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the only v
alid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[9]:
```
<AxesSubplot:xlabel='Ethnicity', ylabel='count'>
```



Bivariate analysisof continuous variable

In [10]:
```
card_approval_df[['Age','Debt','YearsEmployed','CreditScore','Income']].corr()
```

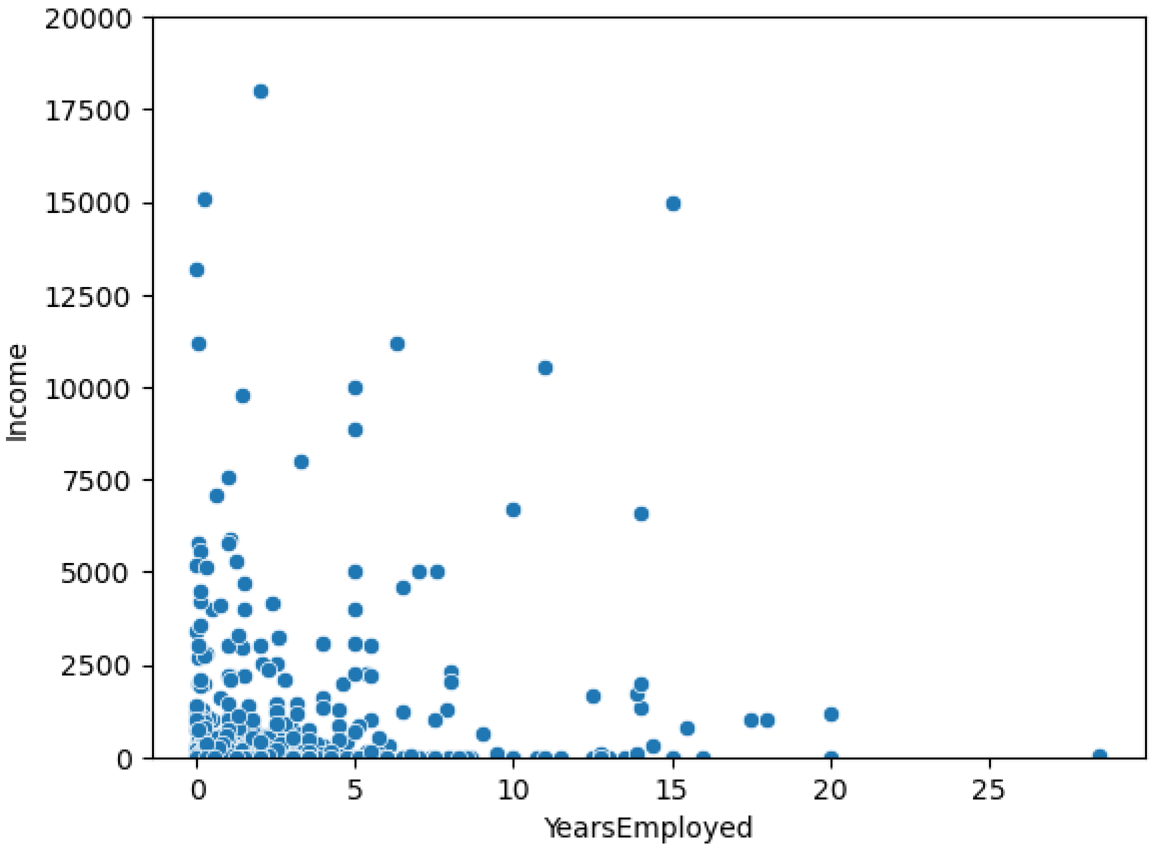
Out[10]:

	Age	Debt	YearsEmployed	CreditScore	Income
Age	1.000000	0.202177	0.391464	0.187327	0.018719
Debt	0.202177	1.000000	0.298902	0.271207	0.123121
YearsEmployed	0.391464	0.298902	1.000000	0.322330	0.051345
CreditScore	0.187327	0.271207	0.322330	1.000000	0.063692
Income	0.018719	0.123121	0.051345	0.063692	1.000000

In [12]:
```
sns.scatterplot(card_approval_df.YearsEmployed,card_approval_df.Income)
plt.ylim(0,20000)
```

```
C:\Users\gptkgf\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variables as keyword args: x, y. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[12]:
```
(0.0, 20000.0)
```



Bivariate Analysis of Categorical Variables vs Continuous Variables

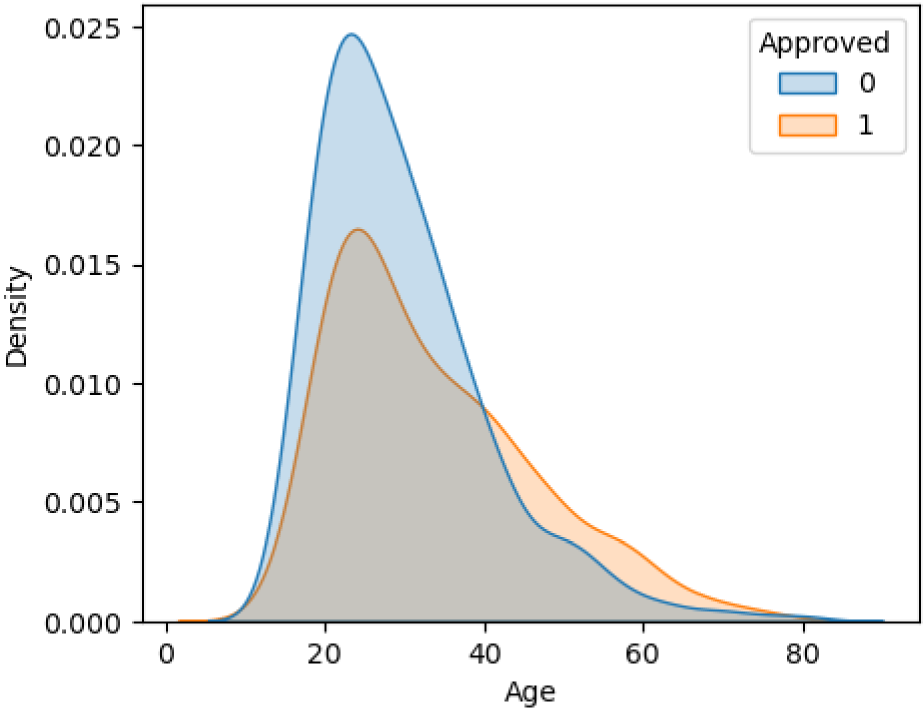
In [13]:
```
card_approval_df.groupby(by='Approved').agg('mean')[['Age','Debt','YearsEmployed',
```

Out[13]:

	Age	Debt	YearsEmployed	CreditScore	Income
Approved					
0	29.773029	3.839948	1.257924	0.631854	198.605744
1	33.686221	5.904951	3.427899	4.605863	2038.859935

In [19]:
```
plt.figure(figsize=(5,4))
sns.kdeplot(data=card_approval_df,x='Age',hue='Approved',fill=True)
```

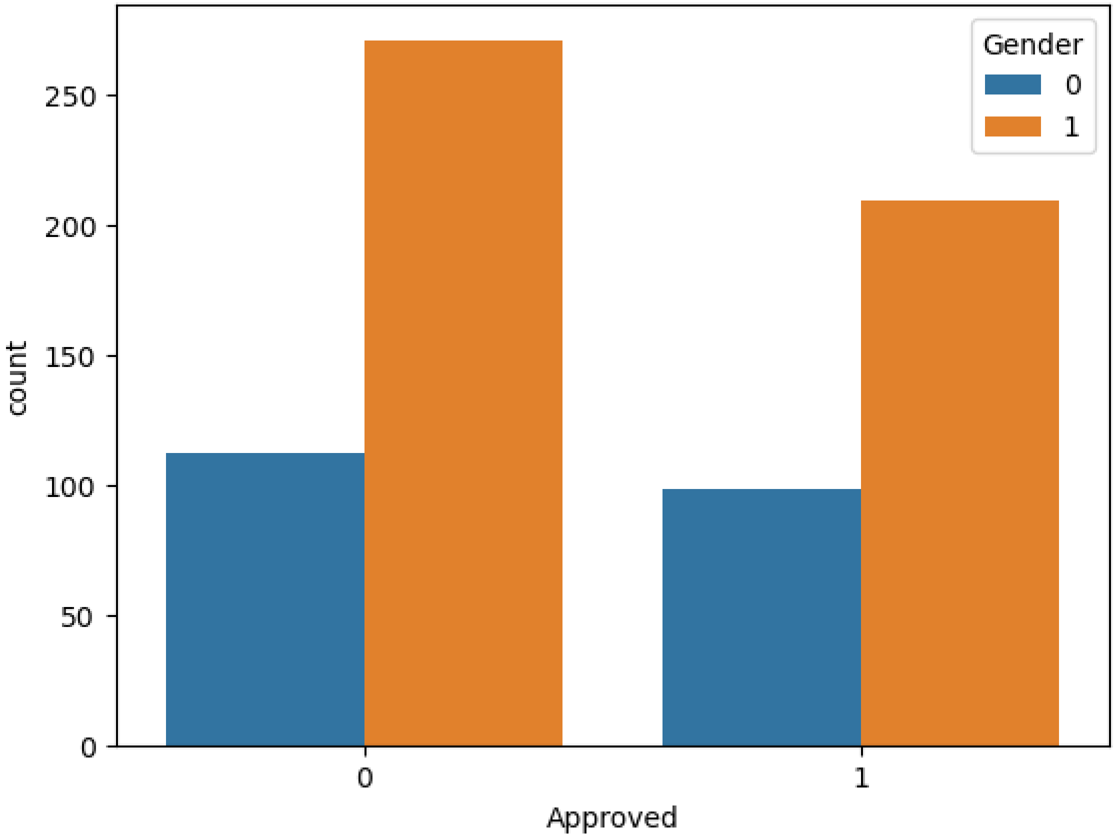
Out[19]:
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



Bivariate Analysis of Categorical Variables vs Categorical Variables

In [20]:
```
sns.countplot(data=card_approval_df,x='Approved',hue='Gender')
```

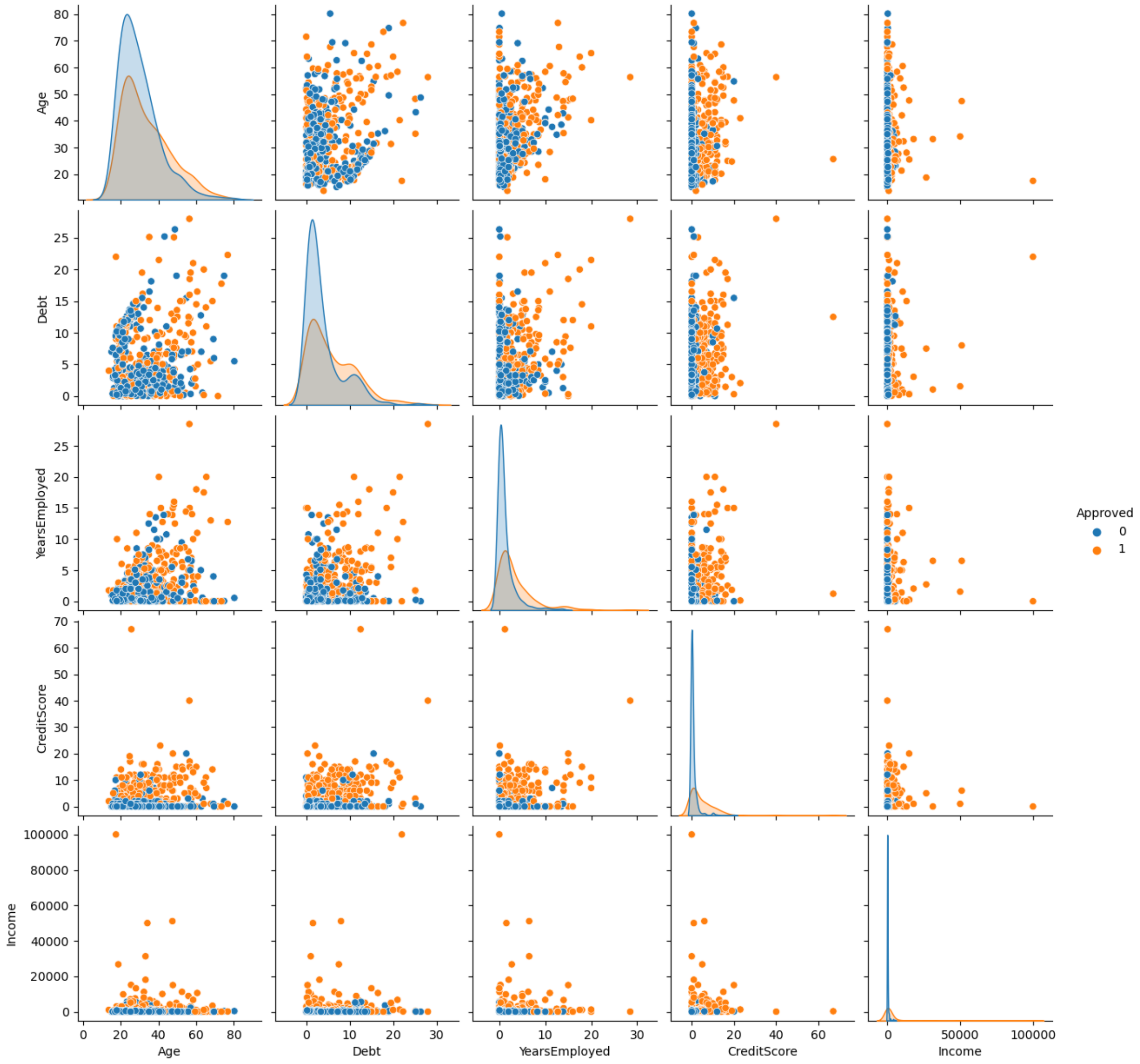
Out[20]:
```
<AxesSubplot:xlabel='Approved', ylabel='count'>
```



Multivariate Analysis

In [21]:
```
sns.pairplot(data=card_approval_df[['Age','Debt','YearsEmployed','CreditScore','In
```

Out[21]:
```
<seaborn.axisgrid.PairGrid at 0x23b342f37c0>
```


Age
Debt
YearsEmployed
CreditScore
Income
80
70
60
50
40
30
20
25
15
10
5
0
100000
80000
60000
40000
20000
0
20
40
60
80
Age
Debt
YearsEmployed
CreditScore
Income
50000
100000
Approved
0
1

```
In [ ]:
```