In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
```

In [4]:
```
credit_df = pd.read_csv("C:/Users/ankus/OneDrive/Desktop/credit_card.csv")
credit_df
```

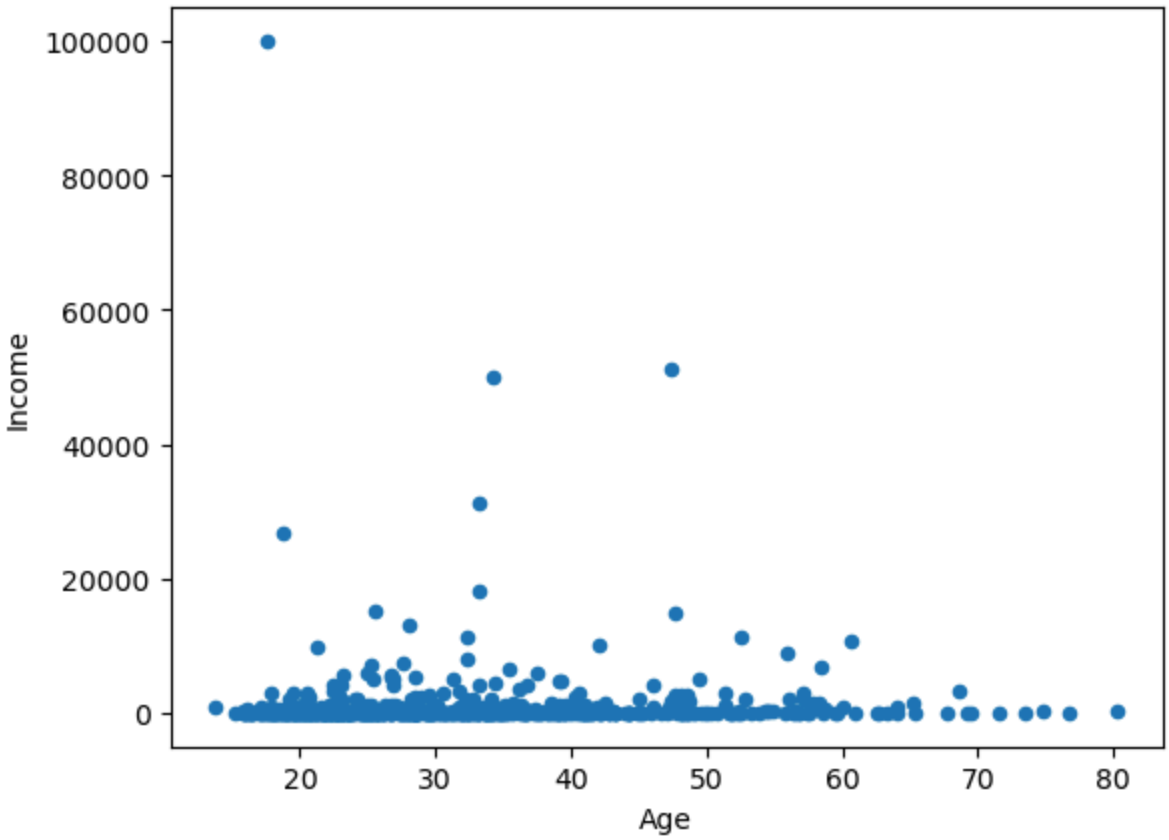
Out[4]:

	Gender	Age	Debt	Married	BankCustomer	Industry	Ethnicity	YearsEmpl
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	
...	...	...	...	...	...	...	...	
685	1	21.08	10.085	0	0	Education	Black	
686	0	22.67	0.750	1	1	Energy	White	
687	0	25.25	13.500	0	0	Healthcare	Latino	
688	1	17.92	0.205	1	1	ConsumerStaples	White	
689	1	35.00	3.375	1	1	Energy	Black	

690 rows × 16 columns

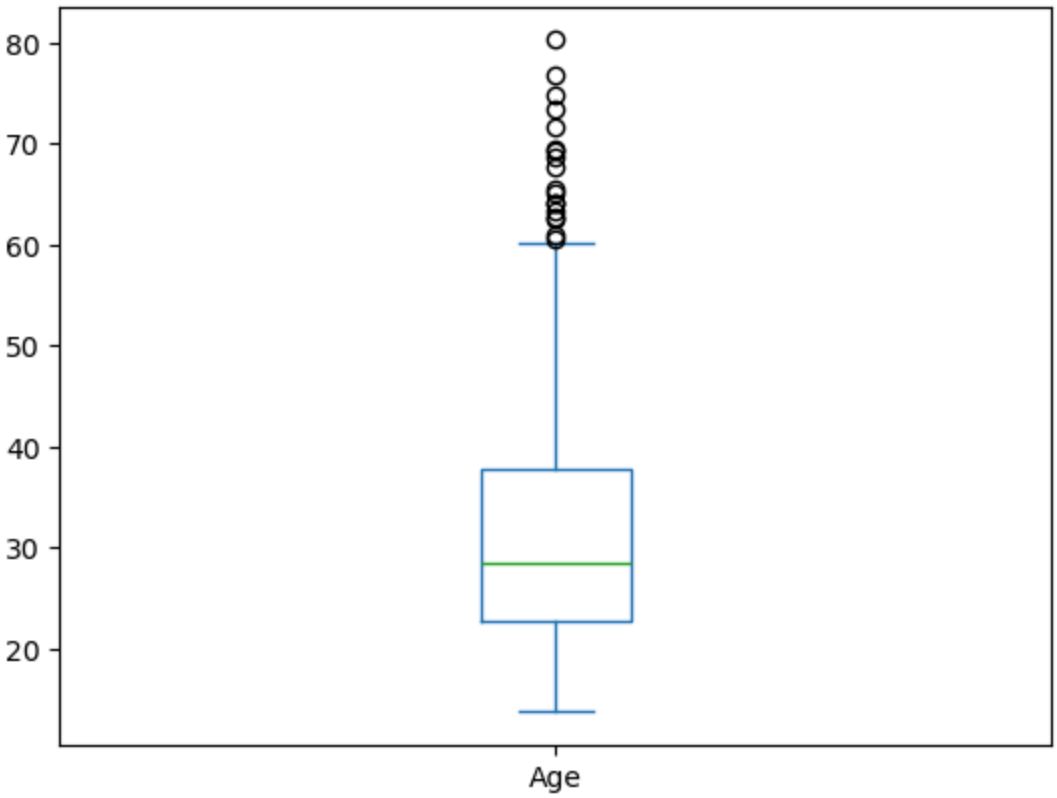
In [5]:
```
credit_df.plot('Age','Income',kind='scatter',marker='o')
```

Out[5]:
```
<Axes: xlabel='Age', ylabel='Income'>
```



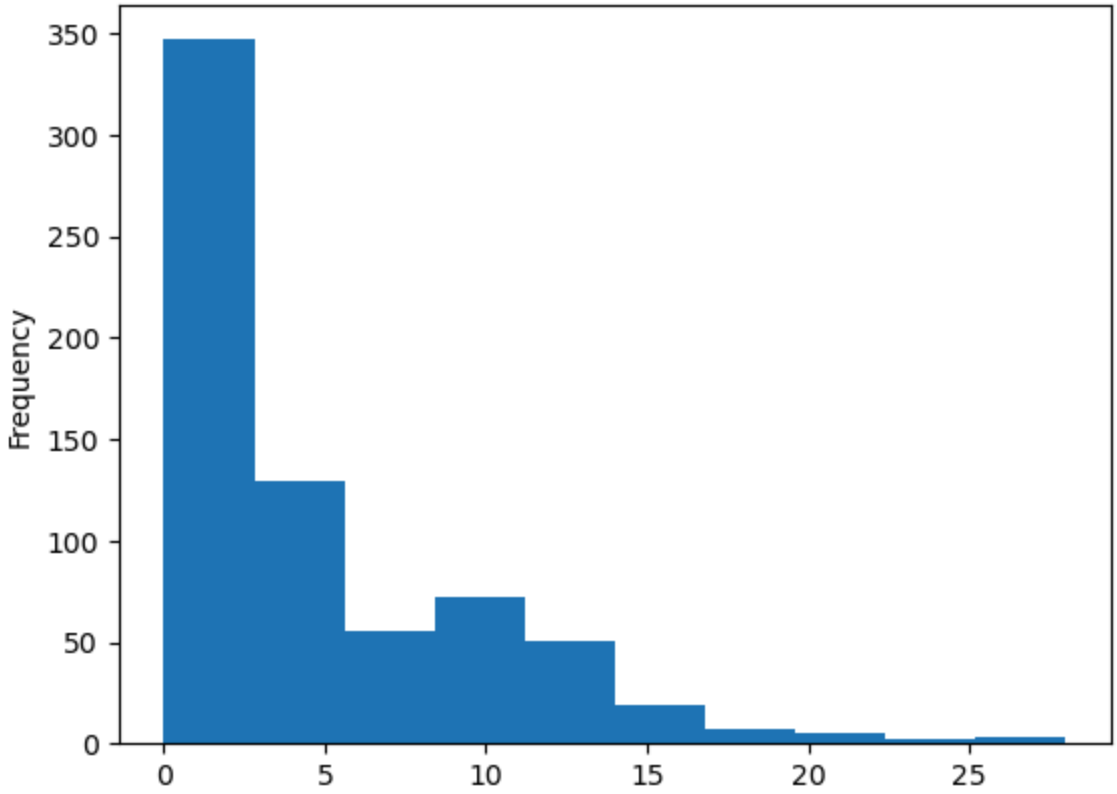
In [6]:
```
credit_df['Age'].plot(kind='box')
```

Out[6]: <Axes: >



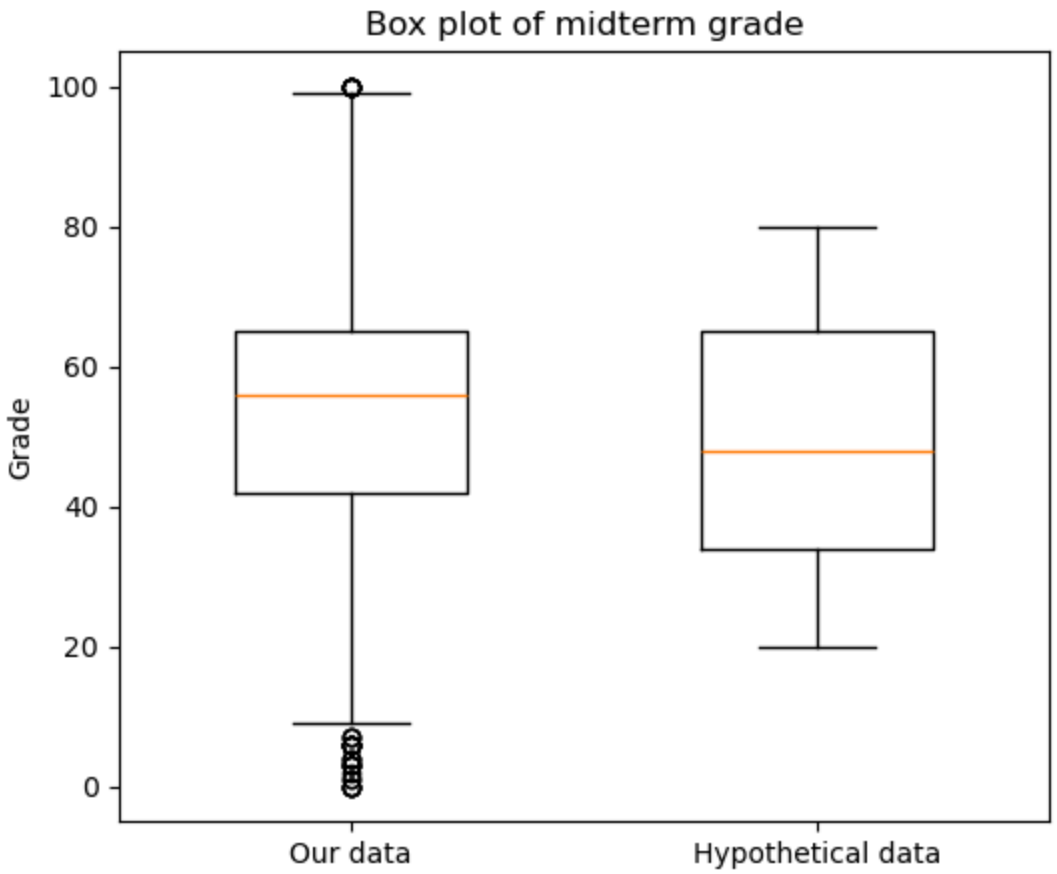
In [7]:
```
credit_df['Debt'].plot(kind='hist')
```

Out[7]: <Axes: ylabel='Frequency'>



In [8]:
```
C
```

```
[  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]
```



In [9]:
```
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]
```

In [10]:
```
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 [11]:
```
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'>
Index: 672 entries, 0 to 689
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Gender          672 non-null    int64  
 1   Age             672 non-null    float64
 2   Debt            672 non-null    float64
 3   Married         672 non-null    int64  
 4   BankCustomer    672 non-null    int64  
 5   Industry        672 non-null    object 
 6   Ethnicity       672 non-null    object 
 7   YearsEmployed   672 non-null    float64
 8   PriorDefault    672 non-null    int64  
 9   Employed        672 non-null    int64  
 10  CreditScore     672 non-null    int64  
 11  DriversLicense  672 non-null    int64  
 12  Citizen         672 non-null    object 
 13  ZipCode         672 non-null    int64  
 14  Income          672 non-null    int64  
 15  Approved        672 non-null    int64  
dtypes: float64(3), int64(10), object(3)
memory usage: 89.2+ KB
```

In [12]:
```
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: 30.541651785714286
```

In [13]:
```
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 28.25
```

In [14]:
```
for i in credit_df['Age']:
    if i<lower_bound or i>upper_bound :
       credit_df['Age'] = credit_df['Age'].replace(i,0)
```

In [15]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import math
```

In [16]:
```
card_approval_df=pd.read_csv("C:/Users/ankus/OneDrive/Desktop/credit_card.csv")
credit_df
```

```
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 [17]:
```
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 [18]:
```
card_approval_df.duplicated().sum()
```

Out[18]: 0

In [19]:
```
card_approval_df[['Age','Debt','YearsEmployed','CreditScore','Income']].describe()
```

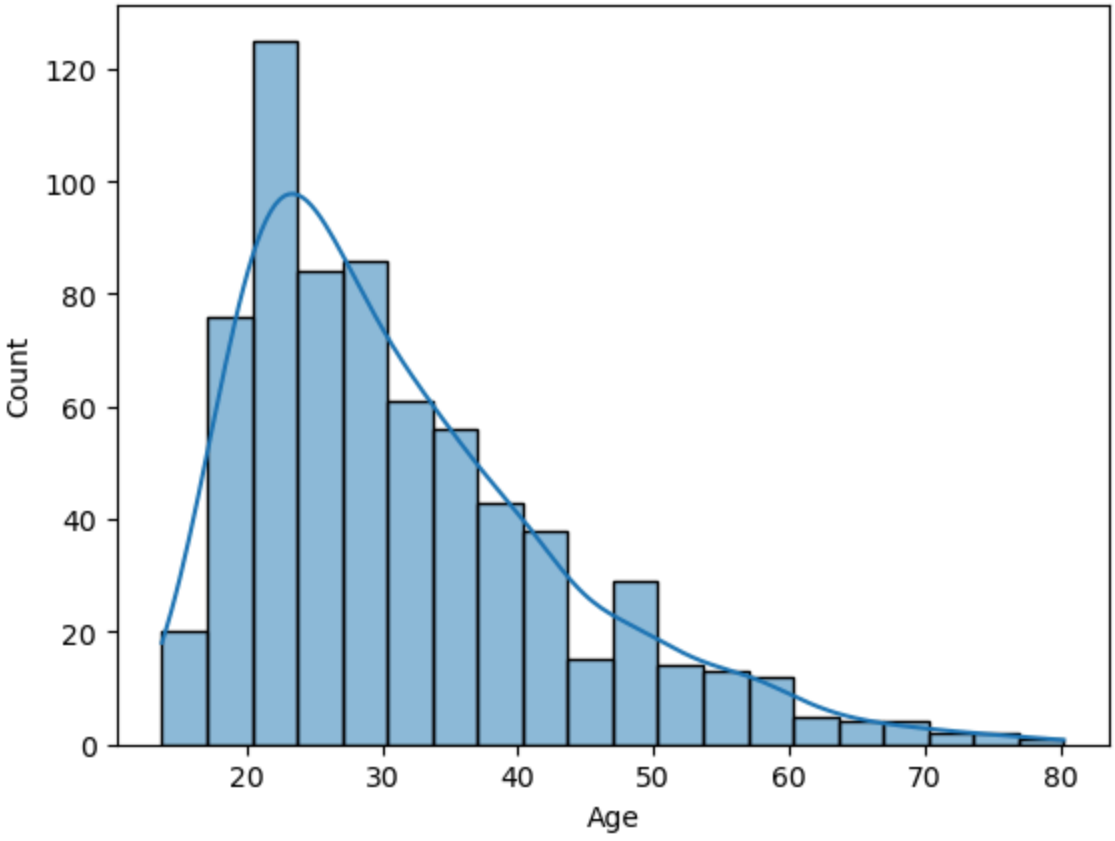
Out[19]:

	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 [20]:
```
sns.histplot(card_approval_df.Age,kde=True)
```

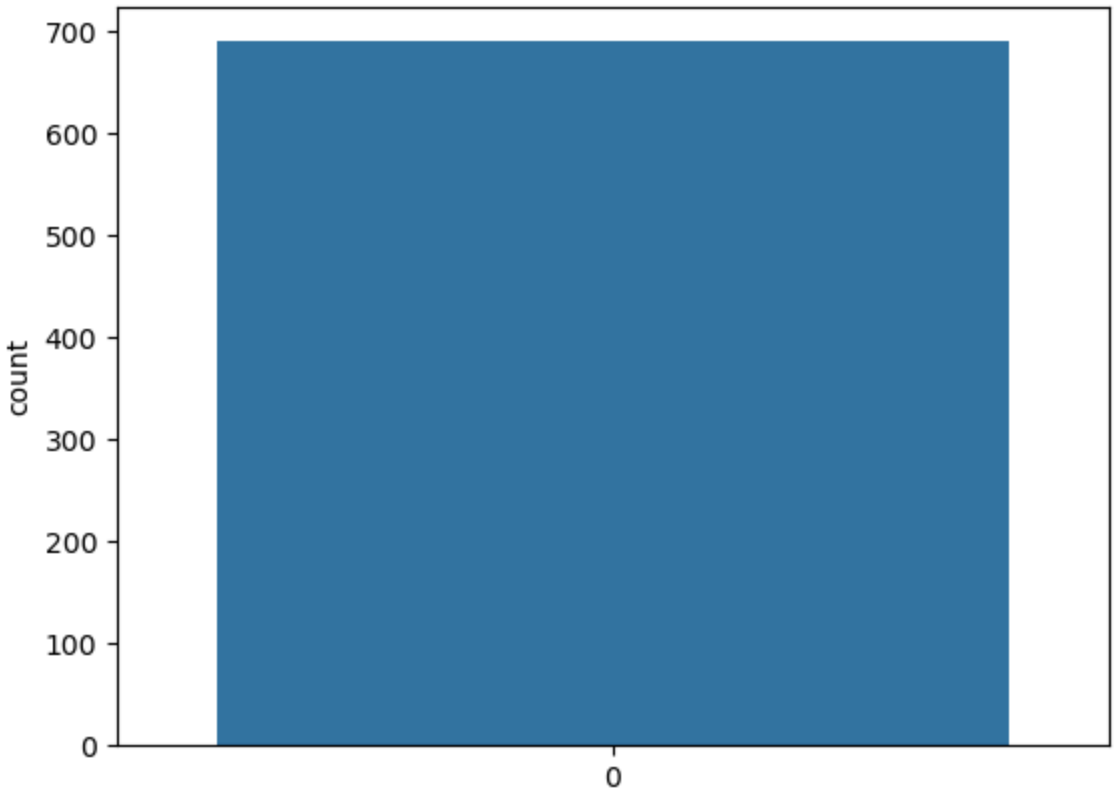
```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[20]: <Axes: xlabel='Age', ylabel='Count'>



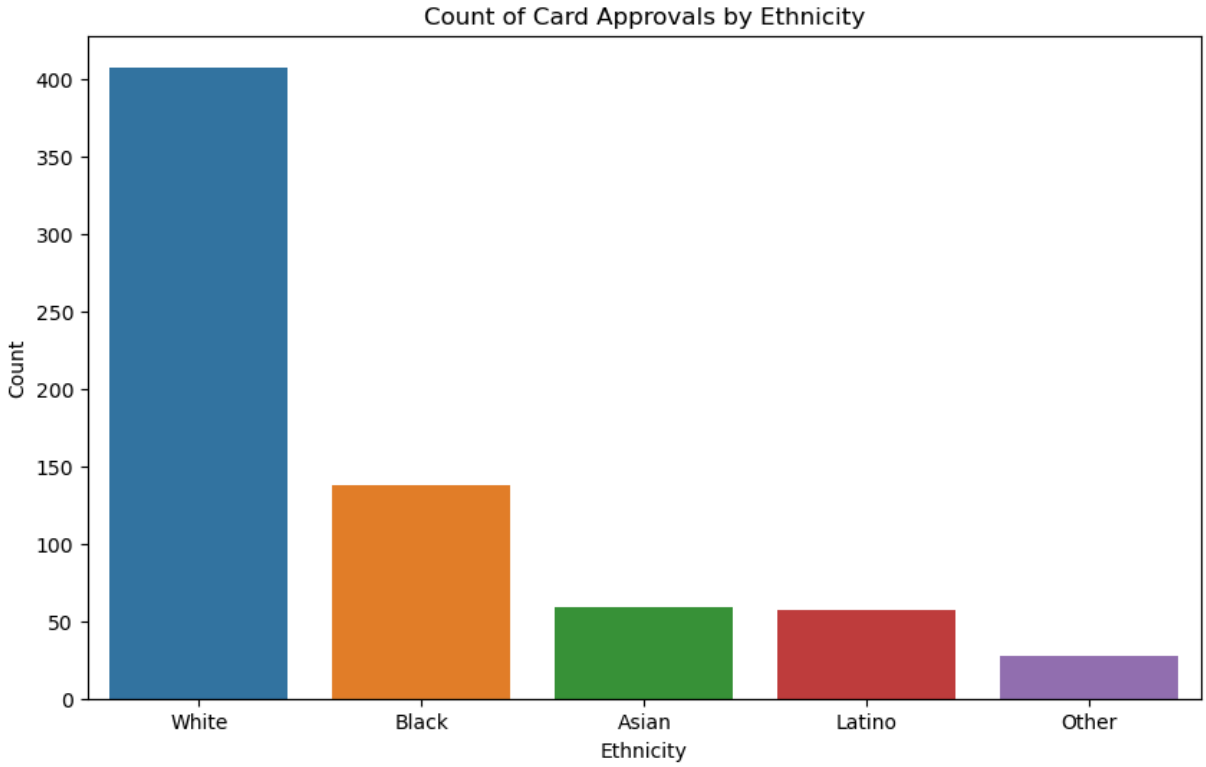
In [21]:
```
sns.countplot(card_approval_df.Gender)
```

Out[21]: <Axes: ylabel='count'>



In [24]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the figure size for better visibility
plt.figure(figsize=(10, 6))

# Create the count plot
sns.countplot(data=card_approval_df, x='Ethnicity')

# Adding labels and title for clarity
plt.xlabel('Ethnicity')
plt.ylabel('Count')
plt.title('Count of Card Approvals by Ethnicity')

# Show the plot
plt.show()
```



In [23]:
```
card_approval_df[['Age','Debt','YearsEmployed','CreditScore','Income']].corr()
```

Out[23]:

	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 [25]:
```
sns.scatterplot(card_approval_df.YearsEmployed,card_approval_df.Income)
plt.ylim(0,20000)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 sns.scatterplot(card_approval_df.YearsEmployed,card_approval_df.Income)
      2 plt.ylim(0,20000)

TypeError: scatterplot() takes from 0 to 1 positional arguments but 2 were given
```

In [30]:
```
card_approval_df.groupby(by='Approved').agg('mean')[['Age','Debt','YearsEmployed']]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\groupby\groupby.py:1874,
in GroupBy._agg_py_fallback(self, how, values, ndim, alt)
   1873 try:
-> 1874     res_values = self.grouper.agg_series(ser, alt, preserve_dtype=True)
   1875 except Exception as err:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\groupby\ops.py:849, in B
aseGrouper.agg_series(self, obj, func, preserve_dtype)
    847     preserve_dtype = True
--> 849 result = self._aggregate_series_pure_python(obj, func)
    851 if len(obj) == 0 and len(result) == 0 and isinstance(obj.dtype, ExtensionDty
pe):

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\groupby\ops.py:877, in B
aseGrouper._aggregate_series_pure_python(self, obj, func)
    876 for i, group in enumerate(splitter):
--> 877     res = func(group)
    878     res = extract_result(res)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\groupby\groupby.py:2380,
in GroupBy.mean.<locals>.<lambda>(x)
   2377 else:
   2378     result = self._cython_agg_general(
   2379         "mean",
-> 2380         alt=lambda x: Series(x).mean(numeric_only=numeric_only),
   2381         numeric_only=numeric_only,
   2382     )
   2383     return result.__finalize__(self.obj, method="groupby")

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\series.py:6225, in Serie
s.mean(self, axis, skipna, numeric_only, **kwargs)
   6217 @doc(make_doc("mean", ndim=1))
   6218 def mean(
   6219     self,
   (...)
   6223     **kwargs,
   6224 ):
-> 6225     return NDFrame.mean(self, axis, skipna, numeric_only, **kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\generic.py:11992, in NDF
rame.mean(self, axis, skipna, numeric_only, **kwargs)
  11985 def mean(
  11986     self,
  11987     axis: Axis | None = 0,
   (...)
  11990     **kwargs,
  11991 ) -> Series | float:
> 11992     return self._stat_function(
  11993         "mean", nanops.nanmean, axis, skipna, numeric_only, **kwargs
  11994     )

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\generic.py:11949, in NDF
rame._stat_function(self, name, func, axis, skipna, numeric_only, **kwargs)
  11947 validate_bool_kwarg(skipna, "skipna", none_allowed=False)
```

```
> 11949 return self._reduce(
  11950     func, name=name, axis=axis, skipna=skipna, numeric_only=numeric_only
  11951 )

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\series.py:6133, in Series._reduce(self, op, name, axis, skipna, numeric_only, filter_type, **kwds)
   6129     raise TypeError(
   6130         f"Series.{name} does not allow {kwd_name}={numeric_only} "
   6131         "with non-numeric dtypes."
   6132     )
-> 6133 return op(delegate, skipna=skipna, **kwds)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\nanops.py:147, in bottleneck_switch.__call__.<locals>.f(values, axis, skipna, **kwds)
    146 else:
--> 147     result = alt(values, axis=axis, skipna=skipna, **kwds)
    149 return result

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\nanops.py:404, in _datetimelike_compat.<locals>.new_func(values, axis, skipna, mask, **kwargs)
    402     mask = isna(values)
--> 404 result = func(values, axis=axis, skipna=skipna, mask=mask, **kwargs)
    406 if datetimelike:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\nanops.py:720, in nanmean(values, axis, skipna, mask)
    719 the_sum = values.sum(axis, dtype=dtype_sum)
--> 720 the_sum = _ensure_numeric(the_sum)
    722 if axis is not None and getattr(the_sum, "ndim", False):

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\nanops.py:1693, in _ensure_numeric(x)
   1691 if isinstance(x, str):
   1692     # GH#44008, GH#36703 avoid casting e.g. strings to numeric
-> 1693     raise TypeError(f"Could not convert string '{x}' to numeric")
   1694 try:

TypeError: Could not convert string 'EducationReal EstateInformationTechnologyCommunicationServicesEnergyMaterialsConsumerStaplesConsumerStaplesFinancialsEnergyHealthcareCommunicationServicesCommunicationServicesReal EstateConsumerStaplesEnergyReal EstateConsumerStaplesConsumerStaplesResearchEnergyEnergyConsumerStaplesEnergyEnergyConsumerDiscretionaryConsumerStaplesEnergyMaterialsMaterialsEnergyEnergyMaterialsFinancialsConsumerDiscretionaryEducationFinancialsConsumerStaplesUtilitiesEnergyIndustrialsHealthcareIndustrialsFinancialsEnergyMaterialsMaterialsFinancialsEnergyEnergyReal EstateConsumerDiscretionaryEnergyReal EstateHealthcareConsumerDiscretionaryUtilitiesHealthcareEnergyEducationReal EstateEnergyCommunicationServicesFinancialsInformationTechnologyReal EstateConsumerDiscretionaryMaterialsHealthcareCommunicationServicesEnergyInformationTechnologyEnergyMaterialsEnergyIndustrialsHealthcareIndustrialsEnergyResearchCommunicationServicesIndustrialsIndustrialsIndustrialsHealthcareHealthcareHealthcareConsumerStaplesConsumerStaplesEnergyHealthcareReal EstateEnergyMaterialsMaterialsHealthcareHealthcareHealthcareFinancialsResearchIndustrialsEnergyConsumerStaplesReal EstateConsumerStaplesCommunicationServicesEnergyIndustrialsUtilitiesEnergyConsumerDiscretionaryIndustrialsConsumerDiscretionaryEnergyEnergyEnergyReal EstateInformationTechnologyMaterialsEnergyConsumerDiscretionaryConsumerStaplesIndustrialsEnergyConsumerStaplesConsumerDiscretionaryConsumerDiscretionaryFinancialsEducationEnergyEnergyEnergyIndustrialsResearchHealthcareCommunicationServicesReal EstateConsumerDiscreti
```

```
onaryConsumerStaplesReal EstateMaterialsReal EstateHealthcareFinancialsHealthcareFin
ancialsFinancialsEducationCommunicationServicesConsumerStaplesCommunicationServicesF
inancialsTransportIndustrialsEducationFinancialsIndustrialsConsumerStaplesEducationR
eal EstateEnergyConsumerDiscretionaryConsumerDiscretionaryConsumerDiscretionaryEnerg
yConsumerDiscretionaryHealthcareFinancialsEnergyConsumerStaplesEnergyConsumerDiscret
ionaryCommunicationServicesEducationConsumerDiscretionaryConsumerStaplesCommunicatio
nServicesIndustrialsReal EstateUtilitiesHealthcareFinancialsEnergyResearchEnergyCons
umerDiscretionaryMaterialsConsumerDiscretionaryHealthcareFinancialsEnergyIndustrials
MaterialsHealthcareEnergyConsumerDiscretionaryConsumerDiscretionaryFinancialsHealthc
areFinancialsEnergyEnergyHealthcareConsumerDiscretionaryMaterialsIndustrialsInformat
ionTechnologyEnergyEnergyIndustrialsEducationHealthcareInformationTechnologyHealthca
reEnergyFinancialsConsumerStaplesEnergyHealthcareHealthcareHealthcareHealthcareFinan
cialsFinancialsHealthcareCommunicationServicesFinancialsResearchHealthcareEnergyHeal
thcareHealthcareEnergyConsumerDiscretionaryEnergyMaterialsIndustrialsReal EstateIndu
strialsEnergyEnergyConsumerDiscretionaryEnergyEnergyCommunicationServicesEnergyHealt
hcareResearchCommunicationServicesEnergyIndustrialsFinancialsConsumerDiscretionaryFi
nancialsConsumerDiscretionaryConsumerDiscretionaryConsumerDiscretionaryEnergyReal Es
tateConsumerDiscretionaryConsumerDiscretionaryConsumerDiscretionaryEnergyConsumerDis
cretionaryFinancialsConsumerStaplesEnergyCommunicationServicesHealthcareFinancialsCo
nsumerStaplesHealthcareMaterialsMaterialsMaterialsEnergyFinancialsConsumerDiscretion
aryFinancialsMaterialsEnergyConsumerStaplesConsumerDiscretionaryHealthcareConsumerSt
aplesHealthcareHealthcareUtilitiesIndustrialsEnergyInformationTechnologyConsumerDisc
retionaryFinancialsIndustrialsConsumerDiscretionaryReal EstateIndustrialsEnergyFinan
cialsMaterialsEnergyFinancialsFinancialsReal EstateUtilitiesFinancialsMaterialsIndus
trialsEnergyEnergyEnergyConsumerDiscretionaryHealthcareCommunicationServicesConsumer
StaplesConsumerDiscretionaryMaterialsInformationTechnologyEnergyConsumerDiscretionar
yHealthcareReal EstateIndustrialsMaterialsCommunicationServicesHealthcareFinancialsI
ndustrialsEnergyCommunicationServicesConsumerStaplesConsumerStaplesFinancialsConsume
rDiscretionaryInformationTechnologyConsumerStaplesConsumerStaplesIndustrialsInformat
ionTechnologyEnergyConsumerStaplesFinancialsHealthcareConsumerDiscretionaryMaterials
ConsumerDiscretionaryConsumerDiscretionaryMaterialsHealthcareEnergyConsumerStaplesEn
ergyEnergyConsumerStaplesEnergyIndustrialsInformationTechnologyIndustrialsCommunicat
ionServicesEnergyEnergyReal EstateHealthcareIndustrialsInformationTechnologyConsumer
StaplesEducationConsumerDiscretionaryConsumerStaplesMaterialsCommunicationServicesHe
althcareEnergyFinancialsReal EstateConsumerDiscretionaryReal EstateCommunicationServ
icesEducationEnergyHealthcareConsumerStaplesEnergy' to numeric

The above exception was the direct cause of the following exception:

TypeError                                 Traceback (most recent call last)
Cell In[30], line 1
----> 1 card_approval_df.groupby(by='Approved').agg('mean')[['Age','Debt','YearsEmpl
oyed']]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\groupby\generic.py:1445,
in DataFrameGroupBy.aggregate(self, func, engine, engine_kwargs, *args, **kwargs)
   1442     kwargs["engine_kwargs"] = engine_kwargs
   1444 op = GroupByApply(self, func, args=args, kwargs=kwargs)
-> 1445 result = op.agg()
   1446 if not is_dict_like(func) and result is not None:
   1447     # GH #52849
   1448     if not self.as_index and is_list_like(func):

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\apply.py:172, in Apply.a
gg(self)
    169 kwargs = self.kwargs
```

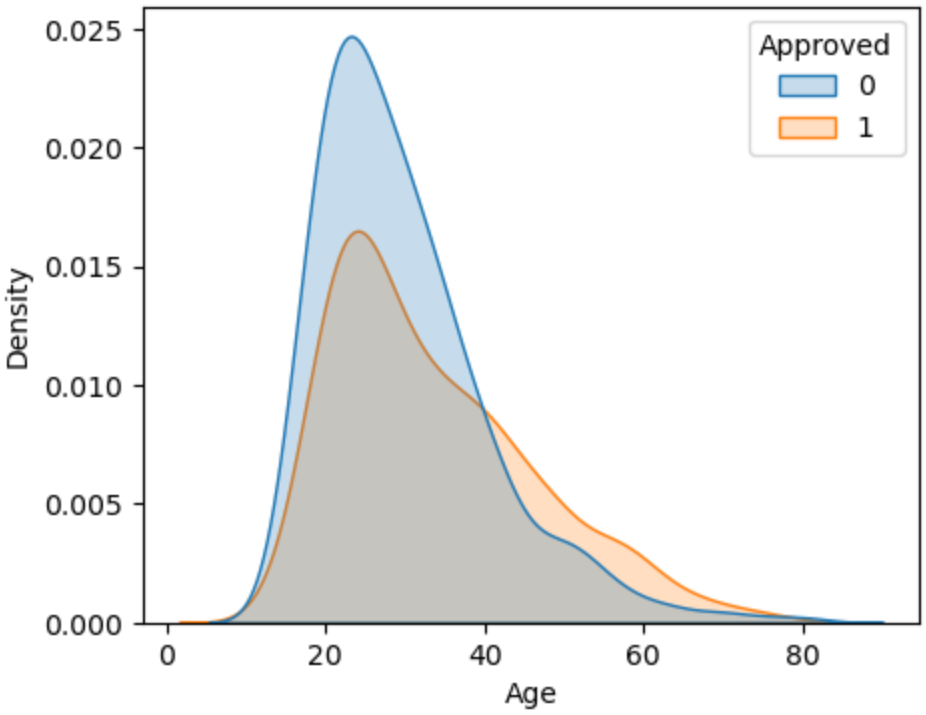
```
    171 if isinstance(func, str):
--> 172     return self.apply_str()
    174 if is_dict_like(func):
    175     return self.agg_dict_like()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\apply.py:586, in Apply.a
pply_str(self)
    584         else:
    585             self.kwargs["axis"] = self.axis
--> 586 return self._apply_str(obj, func, *self.args, **self.kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\apply.py:669, in Apply._
apply_str(self, obj, func, *args, **kwargs)
    667 f = getattr(obj, func)
    668 if callable(f):
--> 669     return f(*args, **kwargs)
    671 # people may aggregate on a non-callable attribute
    672 # but don't let them think they can pass args to it
    673 assert len(args) == 0

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\groupby\groupby.py:2378,
in GroupBy.mean(self, numeric_only, engine, engine_kwargs)
   2371     return self._numba_agg_general(
   2372         grouped_mean,
   2373         executor.float_dtype_mapping,
   2374         engine_kwargs,
   2375         min_periods=0,
   2376     )
   2377 else:
-> 2378     result = self._cython_agg_general(
   2379         "mean",
   2380         alt=lambda x: Series(x).mean(numeric_only=numeric_only),
   2381         numeric_only=numeric_only,
   2382     )
   2383     return result.__finalize__(self.obj, method="groupby")

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\groupby\groupby.py:1929,
in GroupBy._cython_agg_general(self, how, alt, numeric_only, min_count, **kwargs)
   1926     result = self._agg_py_fallback(how, values, ndim=data.ndim, alt=alt)
   1927     return result
-> 1929 new_mgr = data.grouped_reduce(array_func)
   1930 res = self._wrap_agged_manager(new_mgr)
   1931 out = self._wrap_aggregated_output(res)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:14
28, in BlockManager.grouped_reduce(self, func)
   1424 if blk.is_object:
   1425     # split on object-dtype blocks bc some columns may raise
   1426     #  while others do not.
   1427     for sb in blk._split():
-> 1428         applied = sb.apply(func)
   1429         result_blocks = extend_blocks(applied, result_blocks)
   1430 else:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\blocks.py:366,
in Block.apply(self, func, **kwargs)
```

```
    360 @final
    361 def apply(self, func, **kwargs) -> list[Block]:
    362     """
    363     apply the function to my values; return a block if we are not
    364     one
    365     """
--> 366     result = func(self.values, **kwargs)
    368     result = maybe_coerce_values(result)
    369     return self._split_op_result(result)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\groupby\groupby.py:1926,
in GroupBy._cython_agg_general.<locals>.array_func(values)
   1923 else:
   1924     return result
-> 1926 result = self._agg_py_fallback(how, values, ndim=data.ndim, alt=alt)
   1927 return result

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\groupby\groupby.py:1878,
in GroupBy._agg_py_fallback(self, how, values, ndim, alt)
   1876     msg = f"agg function failed [how->{how},dtype->{ser.dtype}]"
   1877     # preserve the kind of exception that raised
-> 1878     raise type(err)(msg) from err
   1880 if ser.dtype == object:
   1881     res_values = res_values.astype(object, copy=False)

TypeError: agg function failed [how->mean,dtype->object]
```

In [27]:
```
plt.figure(figsize=(5,4))
sns.kdeplot(data=card_approval_df,x='Age',hue='Approved',fill=True)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert
inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

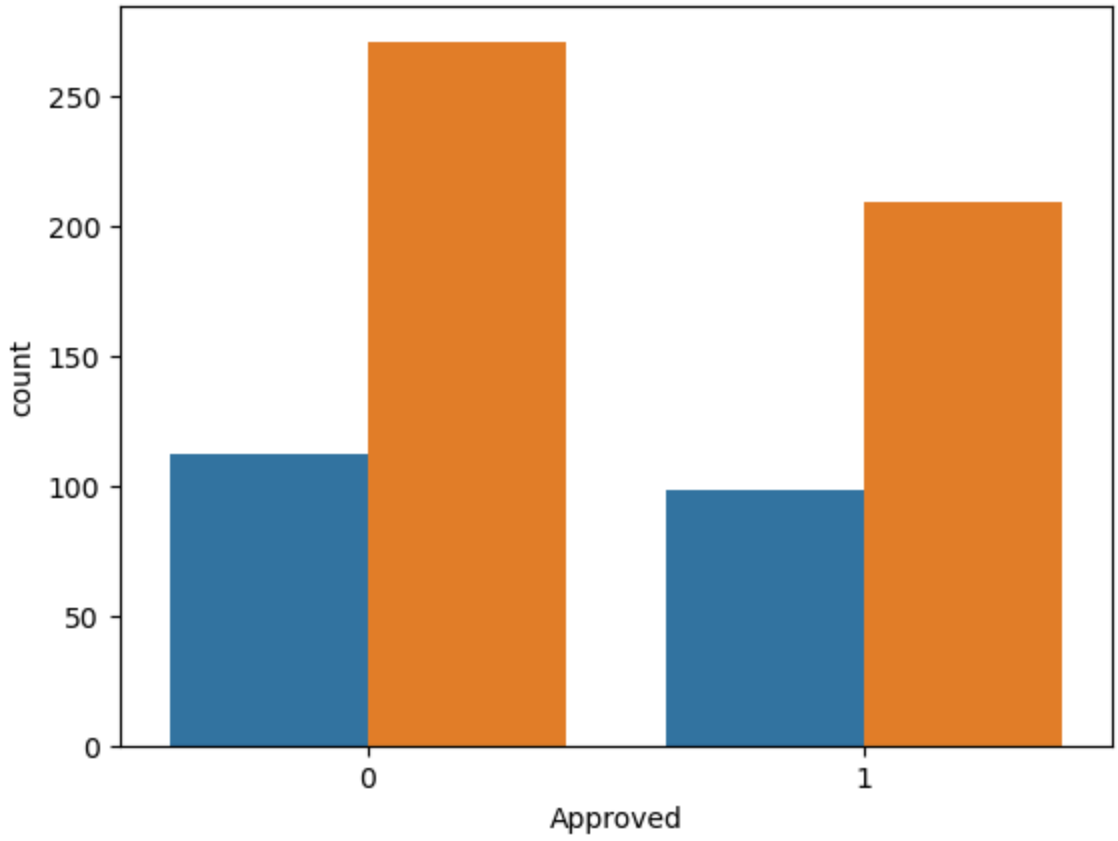
Out[27]:
```
<Axes: xlabel='Age', ylabel='Density'>
```



In [28]:
```
sns.countplot(data=card_approval_df,x='Approved',hue='Gender')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[28], line 1
----> 1 sns.countplot(data=card_approval_df,x='Approved',hue='Gender')

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\categorical.py:2955, in countplot(data, x, y, hue, order, hue_order, orient, color, palette, saturation, width, dodge, ax, **kwargs)
   2952 if ax is None:
   2953     ax = plt.gca()
-> 2955 plotter.plot(ax, kwargs)
   2956 return ax

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\categorical.py:1587, in _BarPlotter.plot(self, ax, bar_kws)
   1585 """Make the plot."""
   1586 self.draw_bars(ax, bar_kws)
-> 1587 self.annotate_axes(ax)
   1588 if self.orient == "h":
   1589     ax.invert_yaxis()

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\categorical.py:767, in _CategoricalPlotter.annotate_axes(self, ax)
    764     ax.set_ylim(-.5, len(self.plot_data) - .5, auto=None)
    766 if self.hue_names is not None:
--> 767     ax.legend(loc="best", title=self.hue_title)

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:322, in Axes.legend(self, *args, **kwargs)
    204 @_docstring.dedent_interpd
    205 def legend(self, *args, **kwargs):
    206     """
    207     Place a legend on the Axes.
    208
   (...)
    320     .. plot:: gallery/text_labels_and_annotations/legend.py
    321     """
--> 322     handles, labels, kwargs = mlegend._parse_legend_args([self], *args, **kwargs)
    323     self.legend_ = mlegend.Legend(self, handles, labels, **kwargs)
    324     self.legend_._remove_method = self._remove_legend

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\legend.py:1361, in _parse_legend_args(axs, handles, labels, *args, **kwargs)
   1357     handles = [handle for handle, label
   1358                in zip(_get_legend_handles(axs, handlers), labels)]
   1360 elif len(args) == 0:  # 0 args: automatically detect labels and handles.
-> 1361     handles, labels = _get_legend_handles_labels(axs, handlers)
   1362     if not handles:
   1363         log.warning(
   1364             "No artists with labels found to put in legend.  Note that "
   1365             "artists whose label start with an underscore are ignored "
   1366             "when legend() is called with no argument.")

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\legend.py:1291, in _get_legend_handles_labels(axs, legend_handler_map)
```

```
   1289 for handle in _get_legend_handles(axs, legend_handler_map):
   1290     label = handle.get_label()
-> 1291     if label and not label.startswith('_'):
   1292         handles.append(handle)
   1293         labels.append(label)

AttributeError: 'numpy.int64' object has no attribute 'startswith'
```



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

In [ ]:
```
497291
```

In [ ]:

In [ ]:

In [ ]: