In [4]:
```
import pandas as pd
import seaborn as  sns
```

In [5]:
```
df=pd.read_csv("BlackFriday.csv")
```

In [8]:
```
df.head()
```

Out[8]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 537577 entries, 0 to 537576
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     537577 non-null  int64  
 1   Product_ID                  537577 non-null  object 
 2   Gender                      537577 non-null  object 
 3   Age                         537577 non-null  object 
 4   Occupation                  537577 non-null  int64  
 5   City_Category               537577 non-null  object 
 6   Stay_In_Current_City_Years  537577 non-null  object 
 7   Marital_Status              537577 non-null  int64  
 8   Product_Category_1          537577 non-null  int64  
 9   Product_Category_2          370591 non-null  float64
 10  Product_Category_3          164278 non-null  float64
 11  Purchase                    537577 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 49.2+ MB
```

In [11]:
```
df.isnull()
```

Out[11]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
537572	False	False	False	False	False	False	False
537573	False	False	False	False	False	False	False
537574	False	False	False	False	False	False	False
537575	False	False	False	False	False	False	False
537576	False	False	False	False	False	False	False

537577 rows × 12 columns

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
User_ID                        0
Product_ID                     0
Gender                         0
Age                            0
Occupation                     0
City_Category                  0
Stay_In_Current_City_Years     0
Marital_Status                 0
Product_Category_1             0
Product_Category_2        166986
Product_Category_3        373299
Purchase                       0
dtype: int64
```

In [13]:
```
del df["Product_Category_2"]
del df["Product_Category_3"]
```

In [14]:
```
df.head()
```

Out[14]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

In [17]:
```
df["User_ID"].nunique()#find the unique user id
```

Out[17]:
```
5891
```

In [18]:
```
df["Product_ID"].nunique()#find unique produc
```

Out[18]:
```
3623
```

In [24]:
```
df["Gender"].unique()
```

Out[24]:
```
array(['F', 'M'], dtype=object)
```

In [25]:
```
df['Age'].unique()
```

Out[25]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [26]:
```
df["Occupation"].unique()
```

Out[26]:
```
array([10, 16, 15,  7, 20,  9,  1, 12, 17,  0,  3,  4, 11,  8, 19,  2, 18,
        5, 14, 13,  6], dtype=int64)
```

In [27]:
```
df['City_Category'].unique()
```

Out[27]:
```
array(['A', 'C', 'B'], dtype=object)
```

In [28]:
```
df["Stay_In_Current_City_Years"].unique()
```

Out[28]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [29]:
```
df["Marital_Status"].unique()
```

Out[29]:
```
array([0, 1], dtype=int64)
```

In [30]:
```
df["Product_Category_1"].unique()
```

Out[30]:
```
array([ 3,  1, 12,  8,  5,  4,  2,  6, 14, 11, 13, 15,  7, 16, 18, 10, 17,
        9], dtype=int64)
```

In [31]:
```
df["Purchase"].sum()#purchase of total amount
```

Out[31]:
```
5017668378
```

In [32]:
```
#finding mean of this
df["Purchase"].sum()/len(df["Purchase"])
```

Out[32]:
```
9333.859852635065
```

In [39]:
```
for column in df.columns:
    print("Unique :",column," :  ",df[column].nunique())
```

```
Unique : User_ID  :   5891
Unique : Product_ID  :   3623
Unique : Gender  :   2
Unique : Age  :   7
Unique : Occupation  :   21
Unique : City_Category  :   3
Unique : Stay_In_Current_City_Years  :   5
Unique : Marital_Status  :   2
Unique : Product_Category_1  :   18
Unique : Purchase  :   17959
```

Analyze Gender

In [40]:
```
df["Gender"]
```

Out[40]:
```
0         F
1         F
2         F
3         F
4         M
         ..
537572    M
537573    M
537574    M
537575    M
537576    M
Name: Gender, Length: 537577, dtype: object
```

In [42]:
```
df[df["Gender"]=='M']
```

Out[42]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
4	1000002	P00285442	M	55+	16	C	4+
5	1000003	P00193542	M	26-35	15	A	3
6	1000004	P00184942	M	46-50	7	B	2
7	1000004	P00346142	M	46-50	7	B	2
8	1000004	P0097242	M	46-50	7	B	2
...	...	...	...	...	...	...	...
537572	1004737	P00193542	M	36-45	16	C	1
537573	1004737	P00111142	M	36-45	16	C	1
537574	1004737	P00345942	M	36-45	16	C	1
537575	1004737	P00285842	M	36-45	16	C	1
537576	1004737	P00118242	M	36-45	16	C	1

405380 rows × 10 columns

In [43]:
```
df[df["Gender"]=='F']
```

Out[43]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
14	1000006	P00231342	F	51-55	9	A	1
...	...	...	...	...	...	...	...
537472	1004726	P00037142	F	36-45	16	C	2
537473	1004726	P00058442	F	36-45	16	C	2
537474	1004726	P00303242	F	36-45	16	C	2
537475	1004727	P00295942	F	55+	0	C	3
537476	1004727	P00351142	F	55+	0	C	3

132197 rows × 10 columns

In [44]:
```
#chek how many amles and female
len(df[df["Gender"]=='M']),len(df[df["Gender"]=='F'])
```

Out[44]:
```
(405380, 132197)
```

In [48]:
```
data=pd.DataFrame({'Ratio': [len(df[df["Gender"]=='M']),len(df[df["Gender"]=='F'])]
```

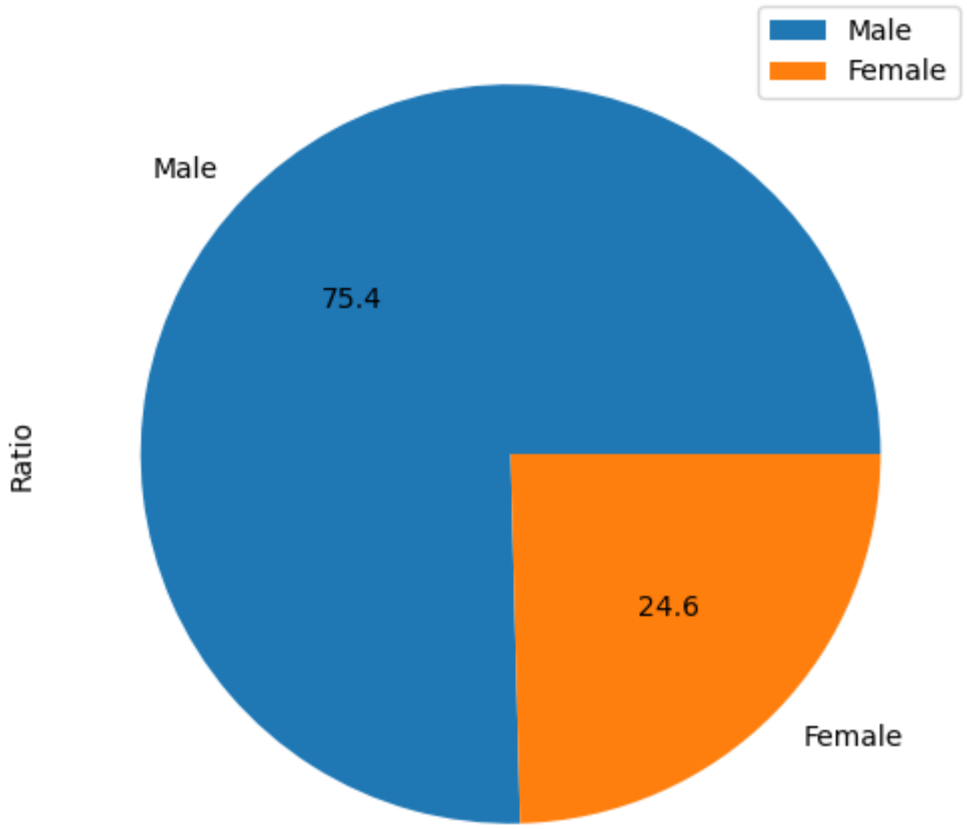
In [49]:
```
data
```

Out[49]:

	Ratio
Male	405380
Female	132197

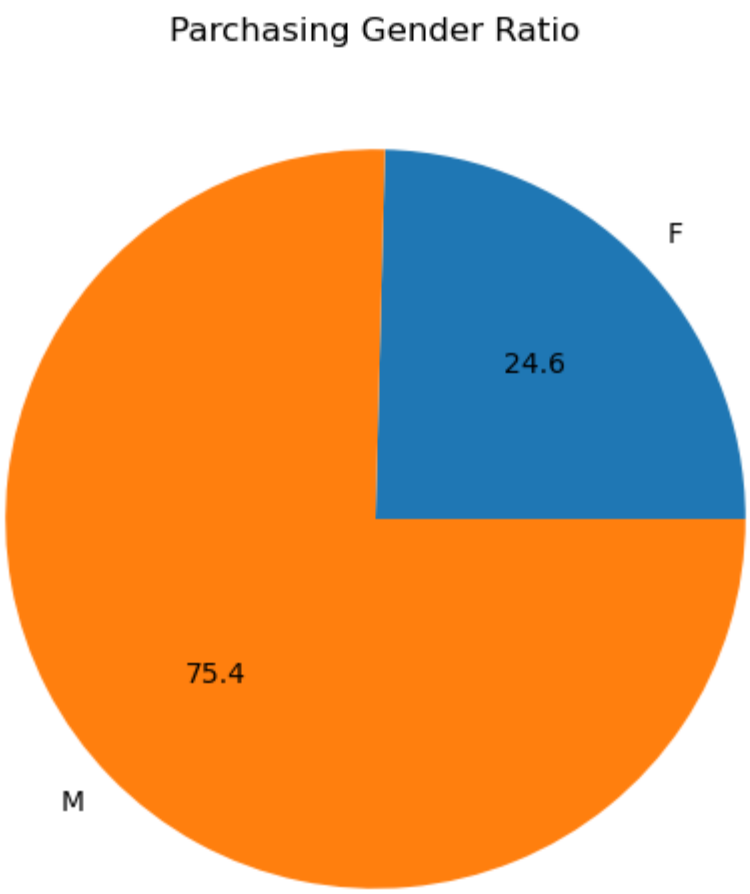
In [53]:
```
data.plot.pie(y="Ratio",figsize=(6,6),autopct="%.1f")
```

Out[53]:
```
<Axes: ylabel='Ratio'>
```



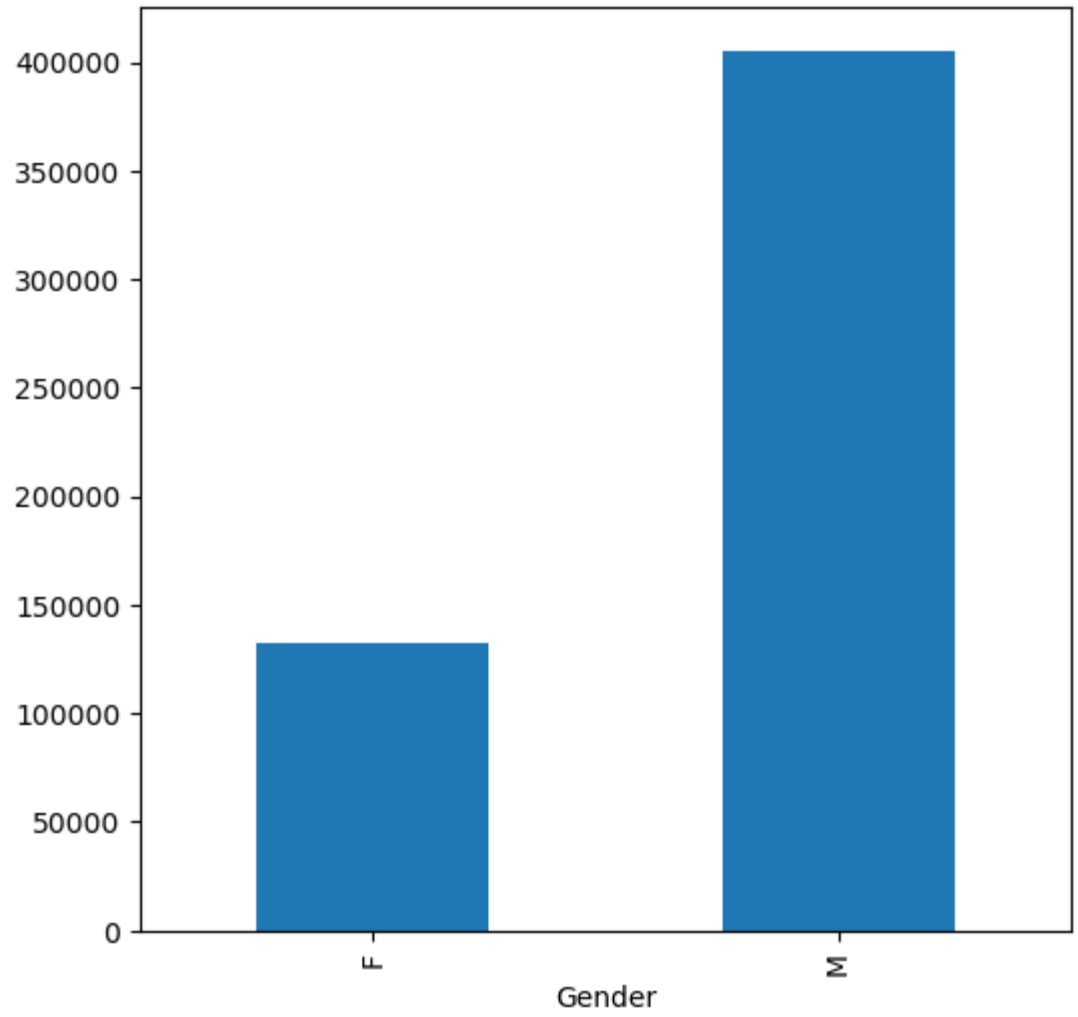
In [55]:
```
df.groupby('Gender').size().plot(kind='pie',figsize=(6,6),autopct="%.1f",title="Par
```

Out[55]:
```
<Axes: title={'center': 'Parchasing Gender Ratio'}>
```



In [62]:
```
df.groupby('Gender').size().plot(kind='bar',figsize=(6,6))
```

Out[62]:
```
<Axes: xlabel='Gender'>
```



In [58]:
```
df.groupby('Gender').size()
```

Out[58]:
```
Gender
F    132197
M    405380
dtype: int64
```

In [59]:
```
df.groupby('Gender').sum()["Purchase"]
```

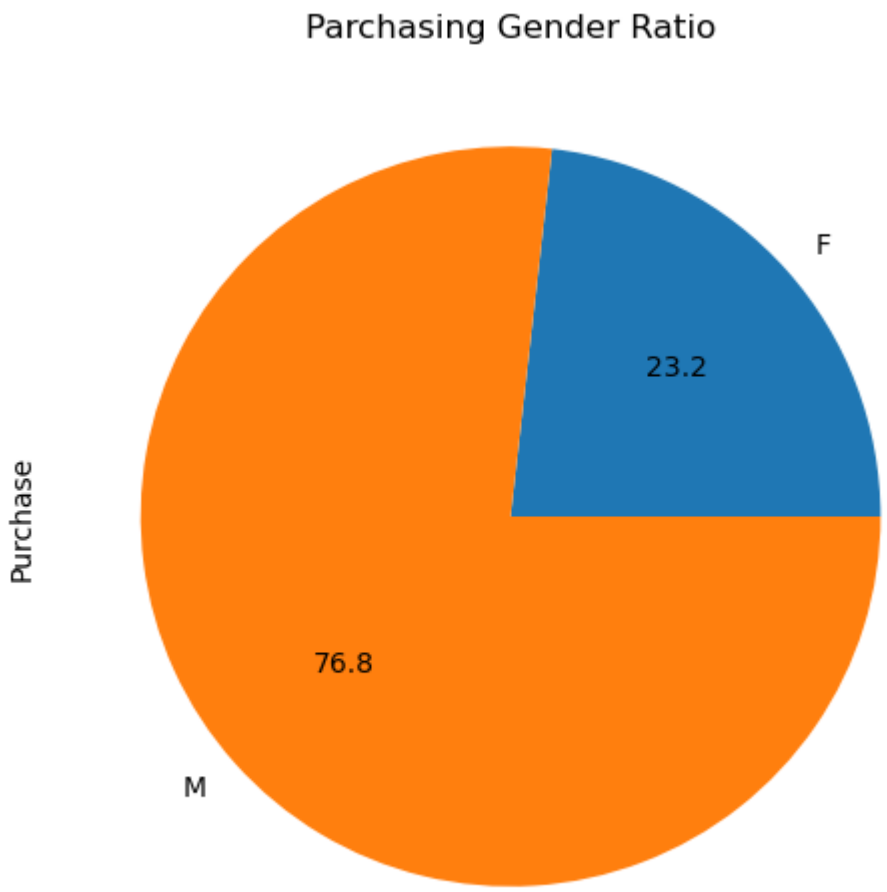
```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\1159811707.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby('Gender').sum()["Purchase"]
```

Out[59]:
```
Gender
F    1164624021
M    3853044357
Name: Purchase, dtype: int64
```

In [64]:
```
df.groupby('Gender').sum()["Purchase"].plot(kind='pie',figsize=(6,6),autopct="%.1f"
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\286340306.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select
only columns which should be valid for the function.
  df.groupby('Gender').sum()["Purchase"].plot(kind='pie',figsize=(6,6),autopct="%.1
f",title="Parchasing Gender Ratio")
```

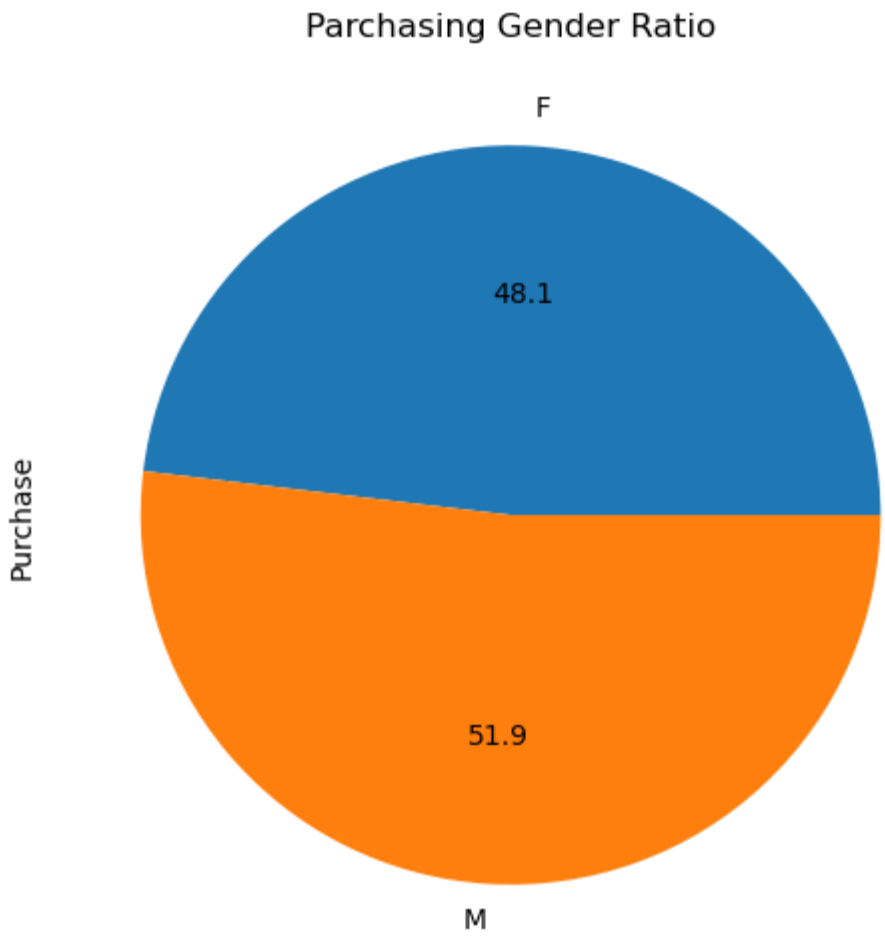
Out[64]:
```
<Axes: title={'center': 'Parchasing Gender Ratio'}, ylabel='Purchase'>
```



In [66]:
```
df.groupby('Gender').mean()["Purchase"].plot(kind='pie',figsize=(6,6),autopct="%.1f
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\2342984211.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futu
re version, numeric_only will default to False. Either specify numeric_only or sele
ct only columns which should be valid for the function.
  df.groupby('Gender').mean()["Purchase"].plot(kind='pie',figsize=(6,6),autopct="%.
1f",title="Parchasing Gender Ratio")
```

Out[66]:
```
<Axes: title={'center': 'Parchasing Gender Ratio'}, ylabel='Purchase'>
```



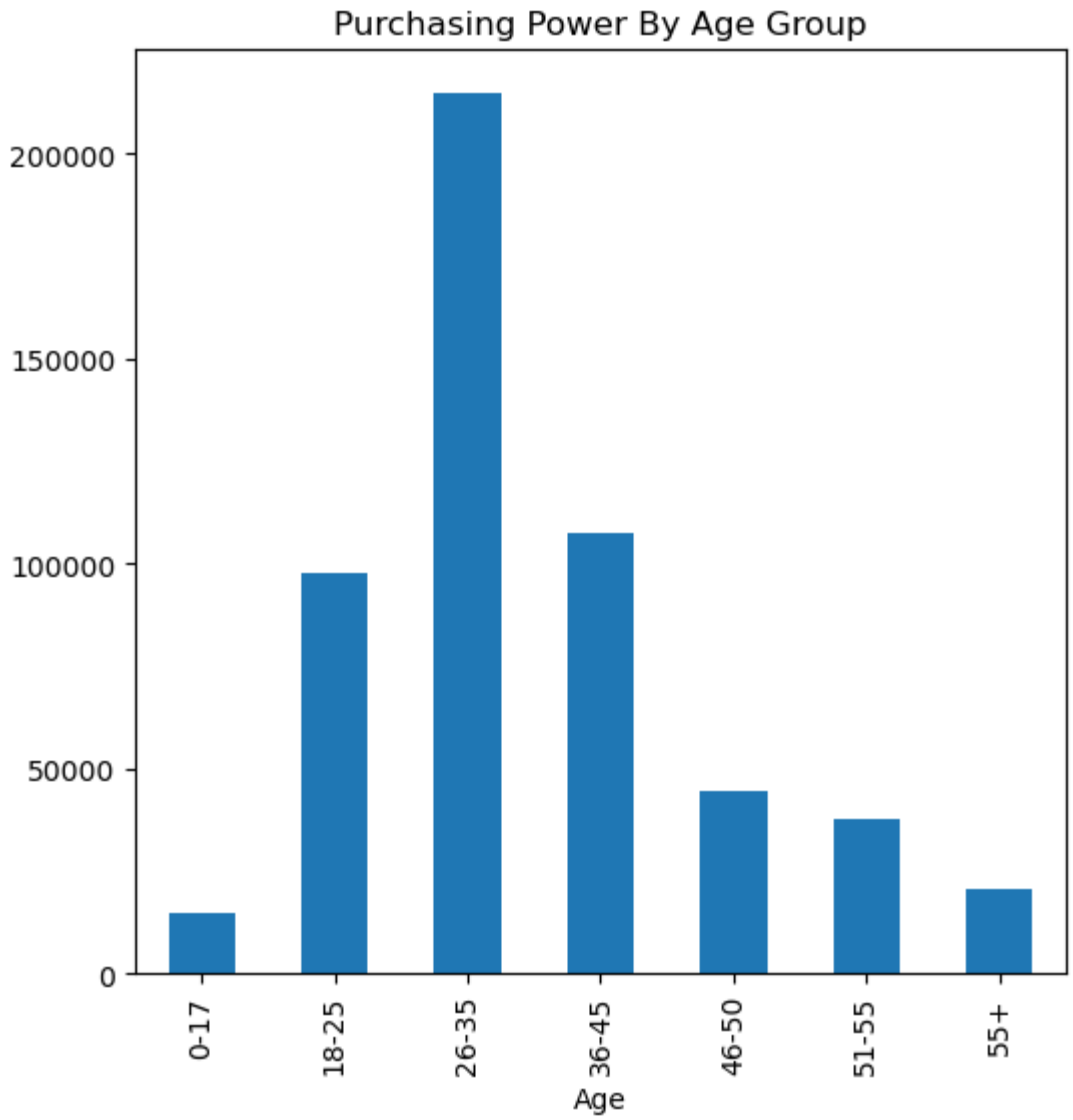
Analyzing Matrial Status and Age gruop By :

In [69]:
```
df.groupby('Age').size()
```

Out[69]:
```
Age
0-17      14707
18-25     97634
26-35    214690
36-45    107499
46-50     44526
51-55     37618
55+       20903
dtype: int64
```

In [71]:
```
df.groupby('Age').size().plot(kind='bar',figsize=(6,6),title=("Purchasing Power By
```

Out[71]:
```
<Axes: title={'center': 'Purchasing Power By Age Group'}, xlabel='Age'>
```



In [74]:
```
for i in df["Age"].unique():
    print(i)
```

```
0-17
55+
26-35
46-50
51-55
36-45
18-25
```

In [75]:
```
df[df["Age"]=='0-17']
```

Out[75]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
85	1000019	P00112542	M	0-17	10	A	3
...	...	...	...	...	...	...	...
537280	1004690	P00242742	F	0-17	10	C	3
537281	1004690	P00226442	F	0-17	10	C	3
537282	1004690	P00181442	F	0-17	10	C	3
537283	1004690	P00192042	F	0-17	10	C	3
537354	1004707	P00253042	M	0-17	0	C	4+

14707 rows × 10 columns

In [76]:
```
#now going to find mayn user havaing 0-17 Age
len(df[df["Age"]=='0-17'])
```

Out[76]: 14707

In [77]:
```
#now going to find mayn user havaing 18-25 Age
len(df[df["Age"]=='18-25'])
```

Out[77]: 97634

In [80]:
```
#now going to find mayn user havaing 36-455 Age
len(df[df["Age"]=='36-45'])
```

Out[80]: 107499

In [81]:
```
#now going to find mayn user havaing 46-50 Age
len(df[df["Age"]=='46-50'])
```

Out[81]: 44526

In [82]:
```
#now going to find mayn user havaing 51-55 Age
len(df[df["Age"]=='51-55'])
```

Out[82]: 37618

In [95]:
```
df[df['Age']==i]["Product_ID"]
```

Out[95]:
```
0         P00069042
1         P00248942
2         P00087842
3         P00085442
85        P00112542
             ...
537280    P00242742
537281    P00226442
537282    P00181442
537283    P00192042
537354    P00253042
Name: Product_ID, Length: 14707, dtype: object
```

In [86]:
```
i
```

Out[86]:
```
'18-25'
```

In [96]:
```
 for i in df['Age'].unique():
        print(i,":",df[df['Age']==i]["Product_ID"].nunique())
```

```
0-17 : 2300
55+ : 2573
26-35 : 3419
46-50 : 3099
51-55 : 2877
36-45 : 3318
18-25 : 3213
```

In [97]:
```
df[df['Age']==i]["Product_ID"].nunique()
```

Out[97]:
```
3213
```

In [104…
```
lst=[]
for i in df['Age'].unique():
    lst.append([i,":",df[df['Age']==i]["Product_ID"].nunique()])
data =pd.DataFrame(lst, columns = ['Age',"Products"])
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
File E:\Anaconda\lib\site-packages\pandas\core\internals\construction.py:969, in _f
inalize_columns_and_data(content, columns, dtype)
    968 try:
--> 969     columns = _validate_or_indexify_columns(contents, columns)
    970 except AssertionError as err:
    971     # GH#26429 do not raise user-facing AssertionError

File E:\Anaconda\lib\site-packages\pandas\core\internals\construction.py:1017, in _
validate_or_indexify_columns(content, columns)
   1015 if not is_mi_list and len(columns) != len(content):  # pragma: no cover
   1016     # caller's responsibility to check for this...
-> 1017     raise AssertionError(
   1018         f"{len(columns)} columns passed, passed data had "
   1019         f"{len(content)} columns"
   1020     )
   1021 elif is_mi_list:
   1022
   1023     # check if nested list column, length of each sub-list should be equal

AssertionError: 2 columns passed, passed data had 3 columns

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

ValueError                                Traceback (most recent call last)
Cell In[104], line 4
      2 for i in df['Age'].unique():
      3     lst.append([i,":",df[df['Age']==i]["Product_ID"].nunique()])
----> 4 data =pd.DataFrame(lst, columns = ['Age',"Products"])

File E:\Anaconda\lib\site-packages\pandas\core\frame.py:746, in DataFrame.__init__
(self, data, index, columns, dtype, copy)
    744     if columns is not None:
    745         columns = ensure_index(columns)
--> 746     arrays, columns, index = nested_data_to_arrays(
    747         # error: Argument 3 to "nested_data_to_arrays" has incompatible
    748         # type "Optional[Collection[Any]]"; expected "Optional[Index]"
    749         data,
    750         columns,
    751         index,  # type: ignore[arg-type]
    752         dtype,
    753     )
    754     mgr = arrays_to_mgr(
    755         arrays,
    756         columns,
   (...)
    759         typ=manager,
    760     )
    761 else:

File E:\Anaconda\lib\site-packages\pandas\core\internals\construction.py:510, in ne
sted_data_to_arrays(data, columns, index, dtype)
    507 if is_named_tuple(data[0]) and columns is None:
    508     columns = ensure_index(data[0]._fields)
--> 510 arrays, columns = to_arrays(data, columns, dtype=dtype)
    511 columns = ensure_index(columns)
    513 if index is None:
```

```
File E:\Anaconda\lib\site-packages\pandas\core\internals\construction.py:875, in to
_arrays(data, columns, dtype)
    872     data = [tuple(x) for x in data]
    873     arr = _list_to_arrays(data)
--> 875 content, columns = _finalize_columns_and_data(arr, columns, dtype)
    876 return content, columns

File E:\Anaconda\lib\site-packages\pandas\core\internals\construction.py:972, in _f
inalize_columns_and_data(content, columns, dtype)
    969     columns = _validate_or_indexify_columns(contents, columns)
    970 except AssertionError as err:
    971     # GH#26429 do not raise user-facing AssertionError
--> 972     raise ValueError(err) from err
    974 if len(contents) and contents[0].dtype == np.object_:
    975     contents = _convert_object_array(contents, dtype=dtype)

ValueError: 2 columns passed, passed data had 3 columns
```

In [105…

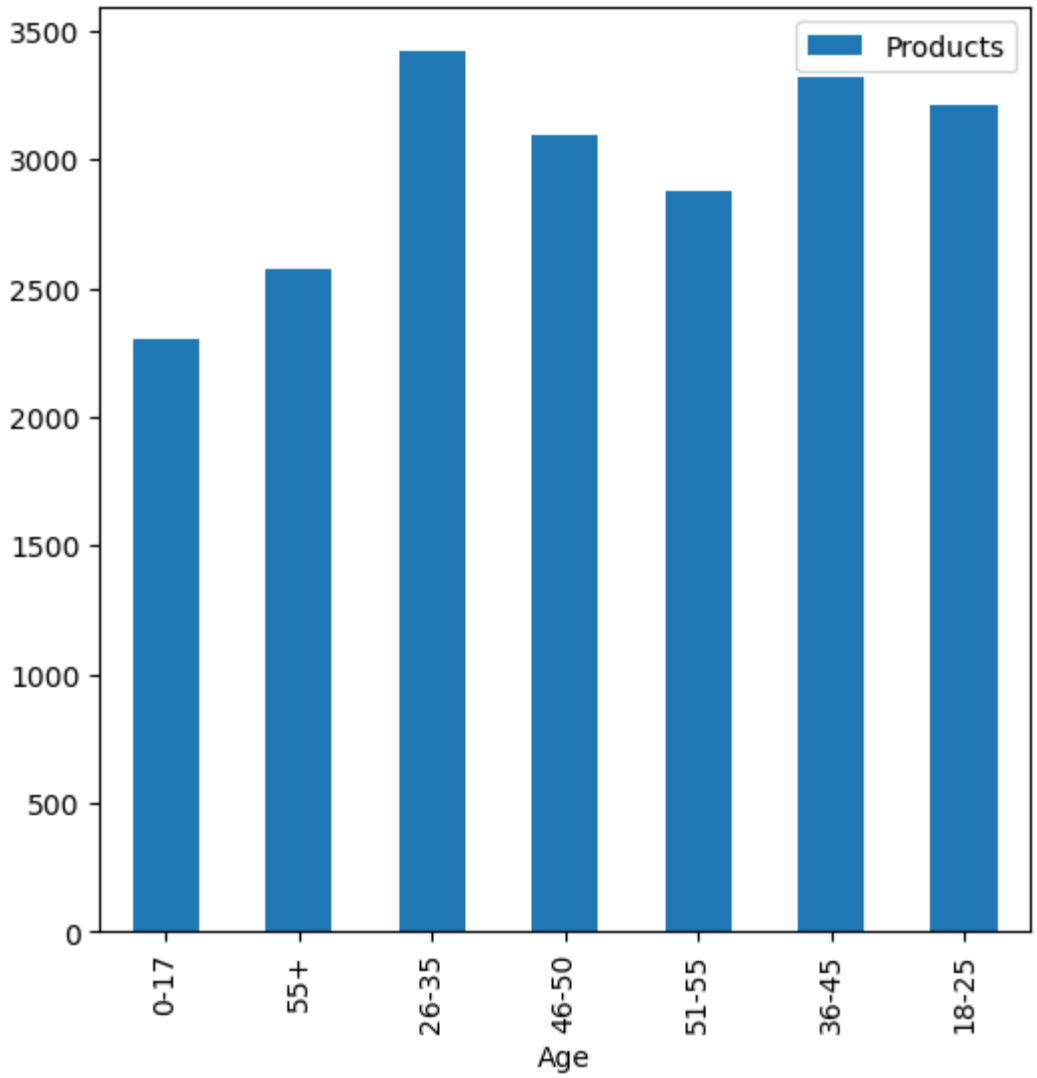
```
lst = []
for i in df["Age"].unique():
    lst.append( [i, df [df['Age']== i]['Product_ID'].nunique()])

data=pd.DataFrame(lst,columns=['Age','Products'])
```

In [108…

```
data.plot.bar(x='Age',figsize=(6,6))
```

Out[108]:

```
<Axes: xlabel='Age'>
```



In [107…
```
data
```

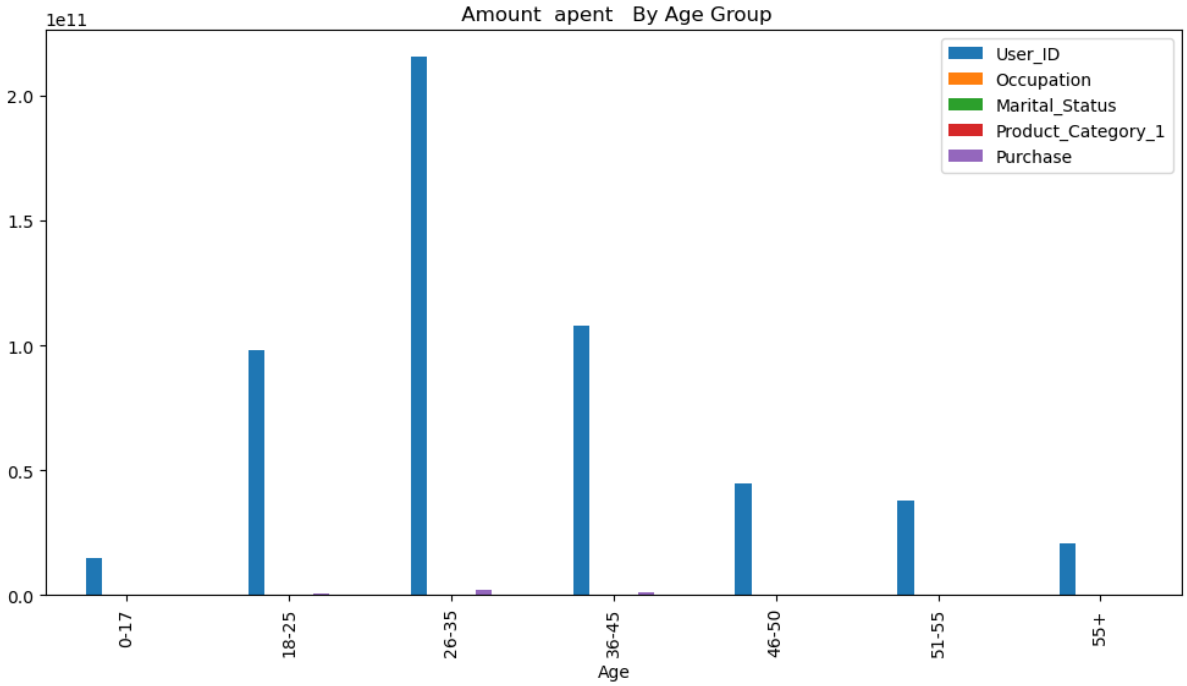
Out[107]:

	Age	Products
0	0-17	2300
1	55+	2573
2	26-35	3419
3	46-50	3099
4	51-55	2877
5	36-45	3318
6	18-25	3213

In [109…
```
df.groupby('Age').sum().plot(kind='bar',figsize=(12,6),title=(" Amount  apent   By
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\2576127938.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby('Age').sum().plot(kind='bar',figsize=(12,6),title=(" Amount  apent   B
y Age Group"))
```

Out[109]: <Axes: title={'center': ' Amount apent   By Age Group'}, xlabel='Age'>

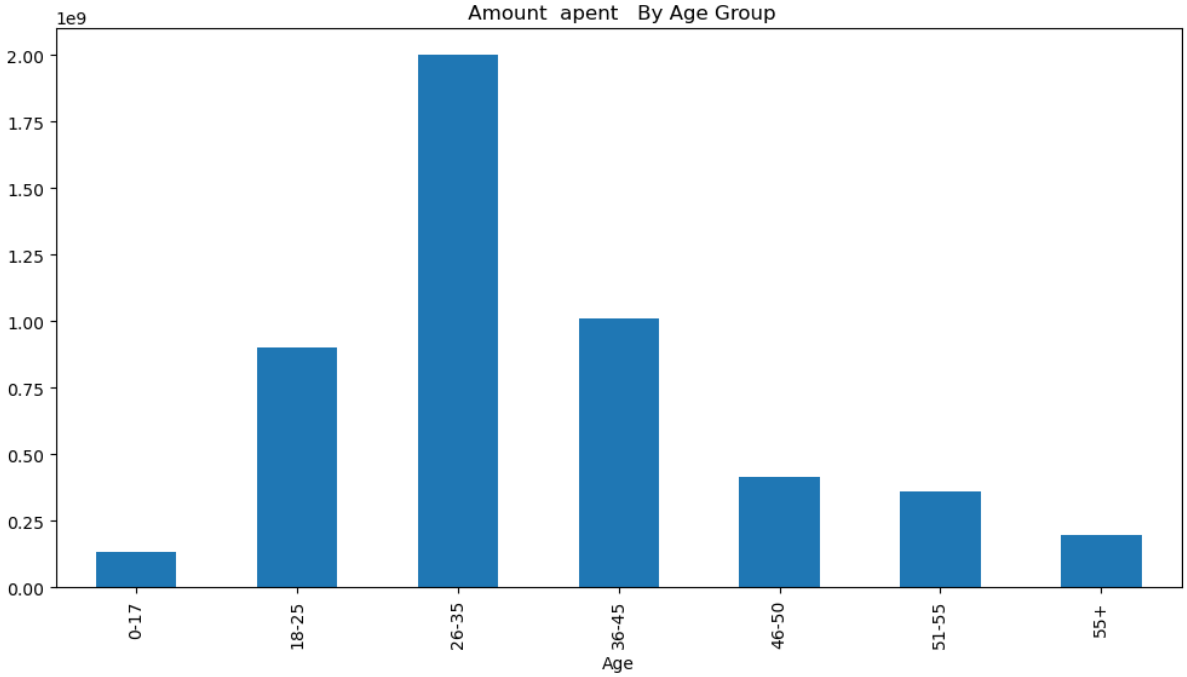


In [110…

```
df.groupby('Age').sum()["Purchase"].plot(kind='bar',figsize=(12,6),title=(" Amount
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\1403859092.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby('Age').sum()["Purchase"].plot(kind='bar',figsize=(12,6),title=(" Amoun
t  apent   By Age Group"))
```

Out[110]: <Axes: title={'center': ' Amount apent   By Age Group'}, xlabel='Age'>



In [113…
```
df.groupby('Age').mean()['Purchase']
```

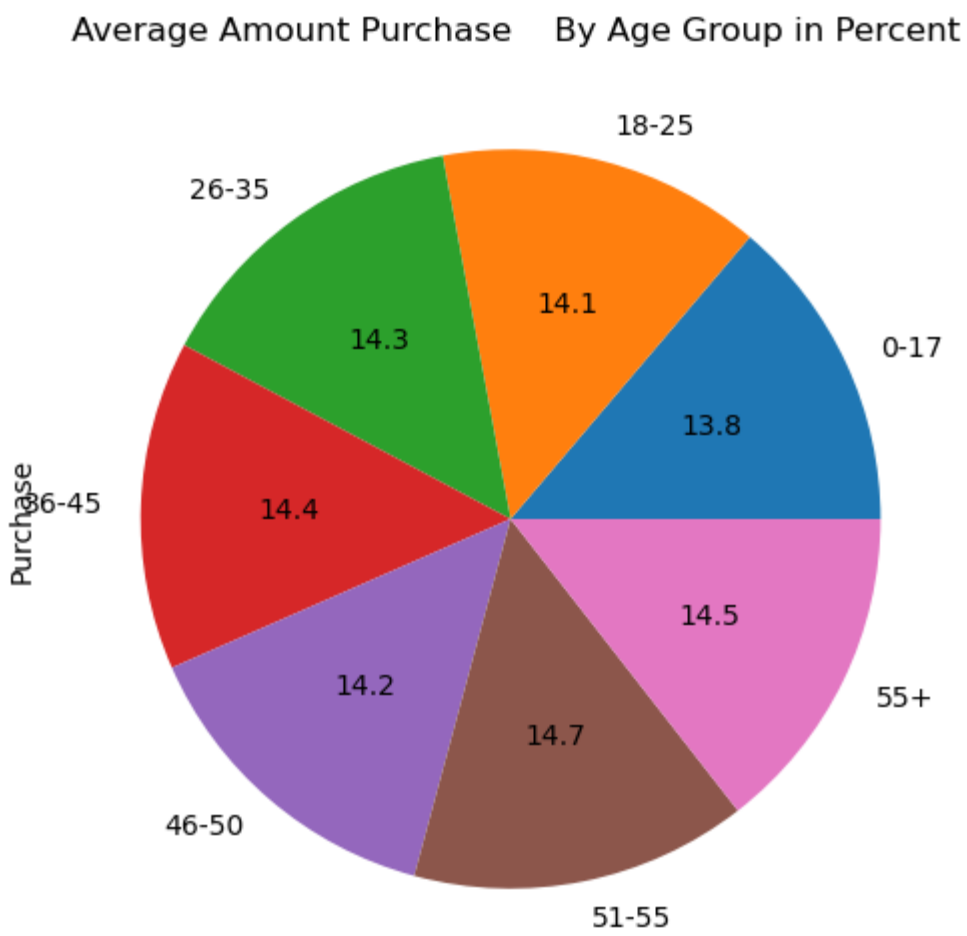
```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\3630277726.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futu
re version, numeric_only will default to False. Either specify numeric_only or sele
ct only columns which should be valid for the function.
  df.groupby('Age').mean()['Purchase']
```

Out[113]:
```
Age
0-17     9020.126878
18-25    9235.197575
26-35    9314.588970
36-45    9401.478758
46-50    9284.872277
51-55    9620.616620
55+      9453.898579
Name: Purchase, dtype: float64
```

In [117…
```
df.groupby('Age').mean()['Purchase'].plot(kind='pie',figsize=(12,6),autopct="%.1f",
```

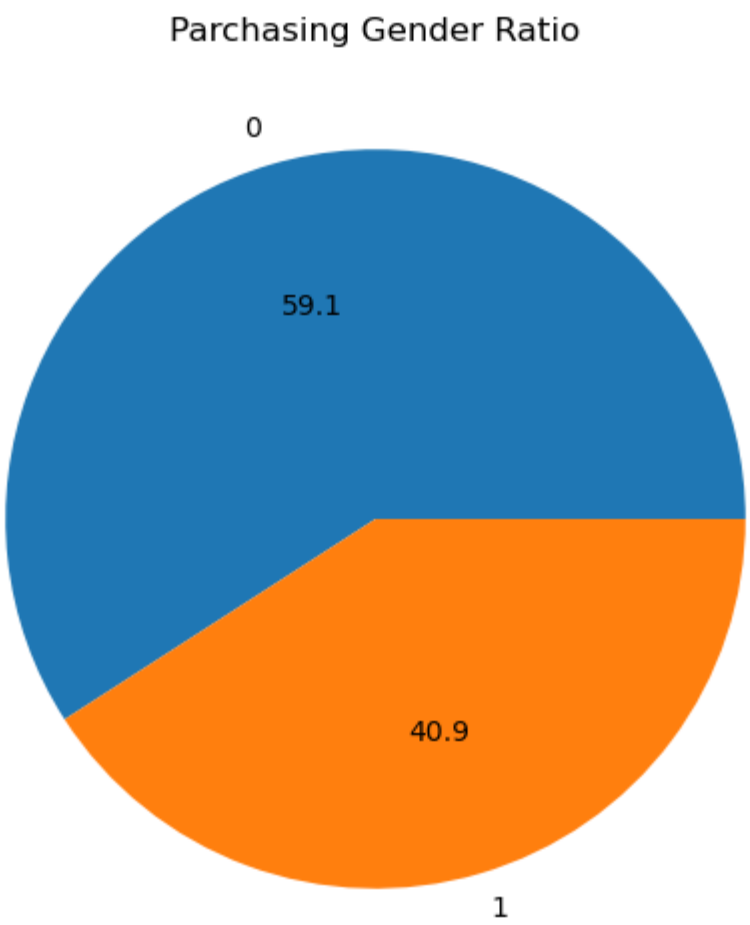
```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\383128545.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby('Age').mean()['Purchase'].plot(kind='pie',figsize=(12,6),autopct="%.1
f",title=(" Average Amount Purchase    By Age Group in Percent"))
```

Out[117]:
```
<Axes: title={'center': ' Average Amount Purchase    By Age Group in Percent'}, yl
abel='Purchase'>
```



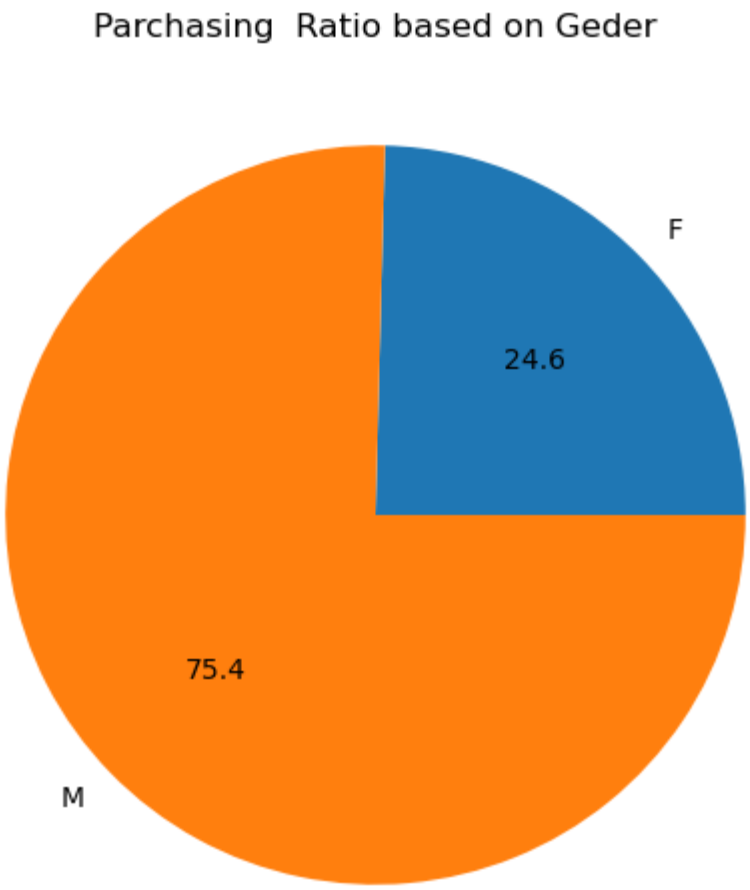
In [118…
```
df.groupby('Marital_Status').size().plot(kind='pie',figsize=(6,6),autopct="%.1f",ti
```

Out[118]:
```
<Axes: title={'center': 'Parchasing Gender Ratio'}>
```



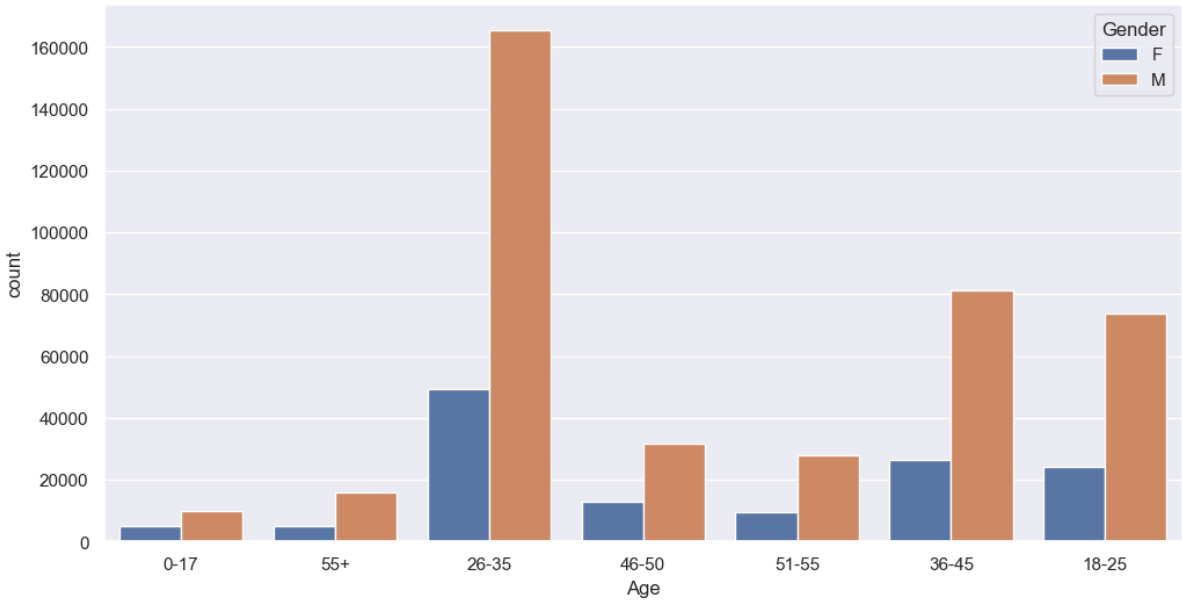
In [122…
```
df.groupby('Gender').size().plot(kind='pie',figsize=(6,6),autopct="%.1f",title="Par
```

Out[122]:
```
<Axes: title={'center': 'Parchasing  Ratio based on Geder'}>
```



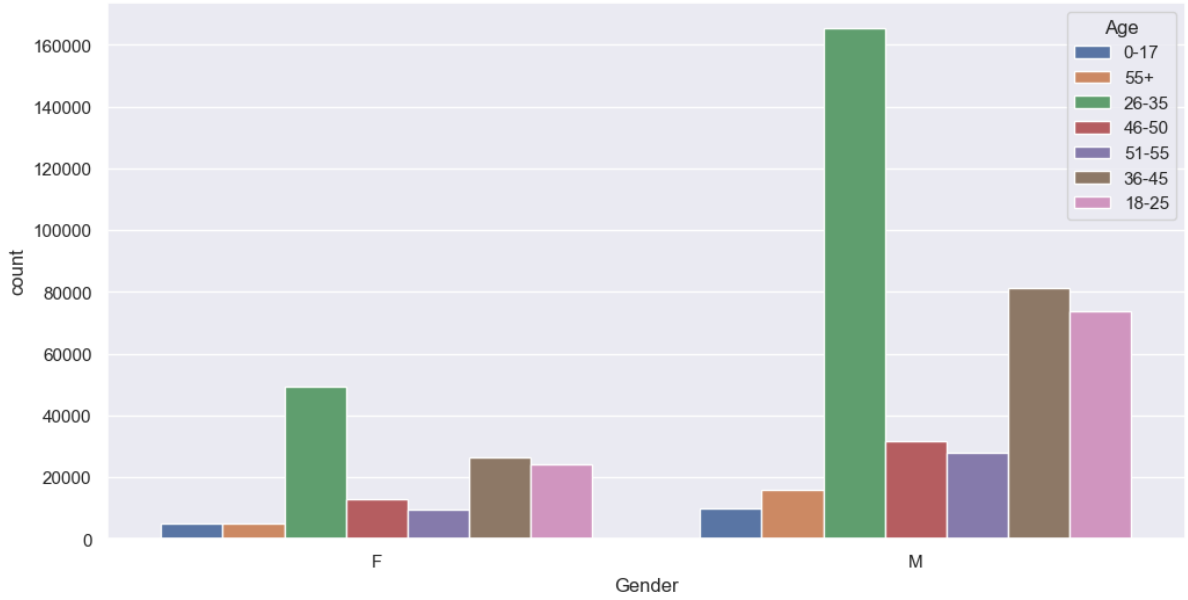
In [127…
```
sns.set(rc={"figure.figsize":(12,6)})
sns.countplot(x='Age',hue='Gender',data=df)
```

Out[127]:
```
<Axes: xlabel='Age', ylabel='count'>
```



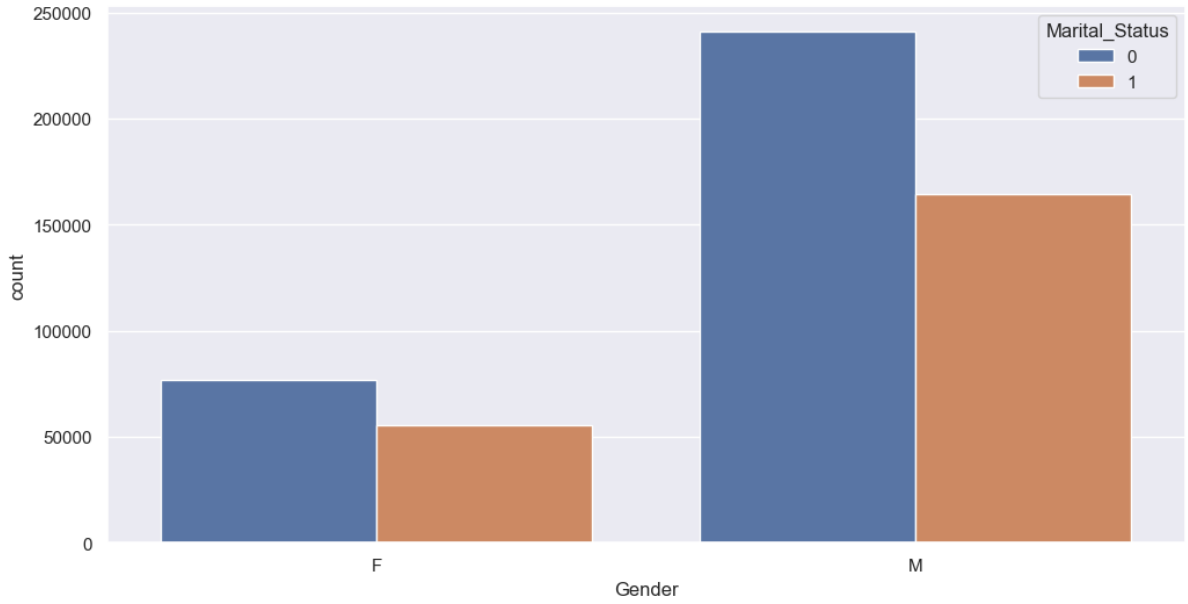
In [128…
```
sns.set(rc={"figure.figsize":(12,6)})
sns.countplot(x='Gender',hue='Age',data=df)
```

Out[128]: <Axes: xlabel='Gender', ylabel='count'>



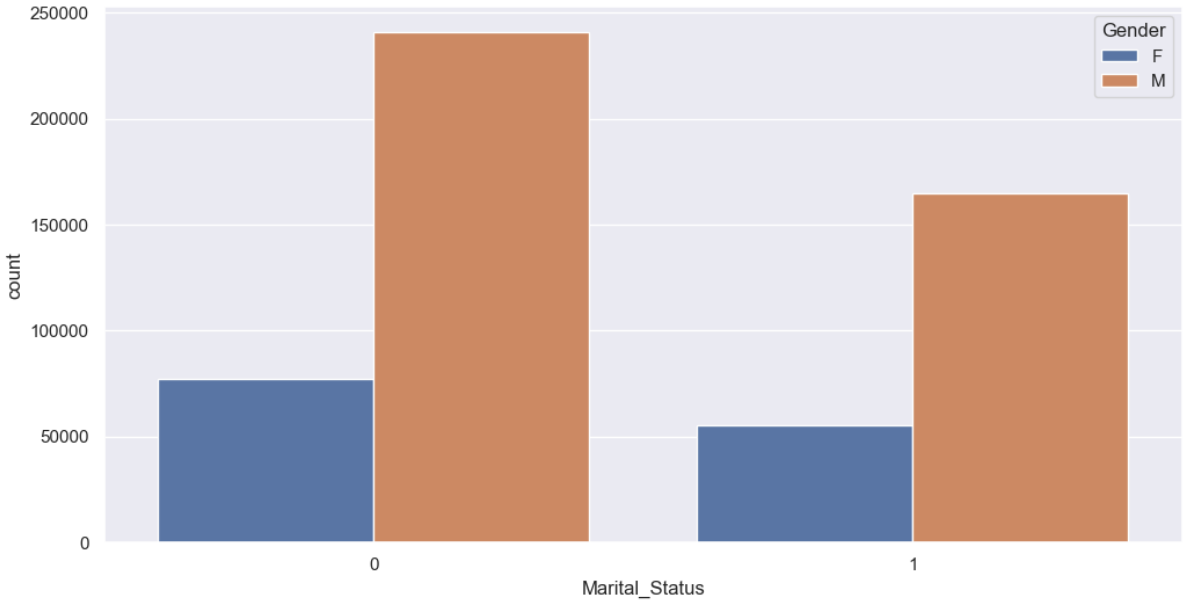
In [131…
```
sns.set(rc={"figure.figsize":(12,6)})
sns.countplot(x='Gender',hue='Marital_Status',data=df)
```

Out[131]: <Axes: xlabel='Gender', ylabel='count'>



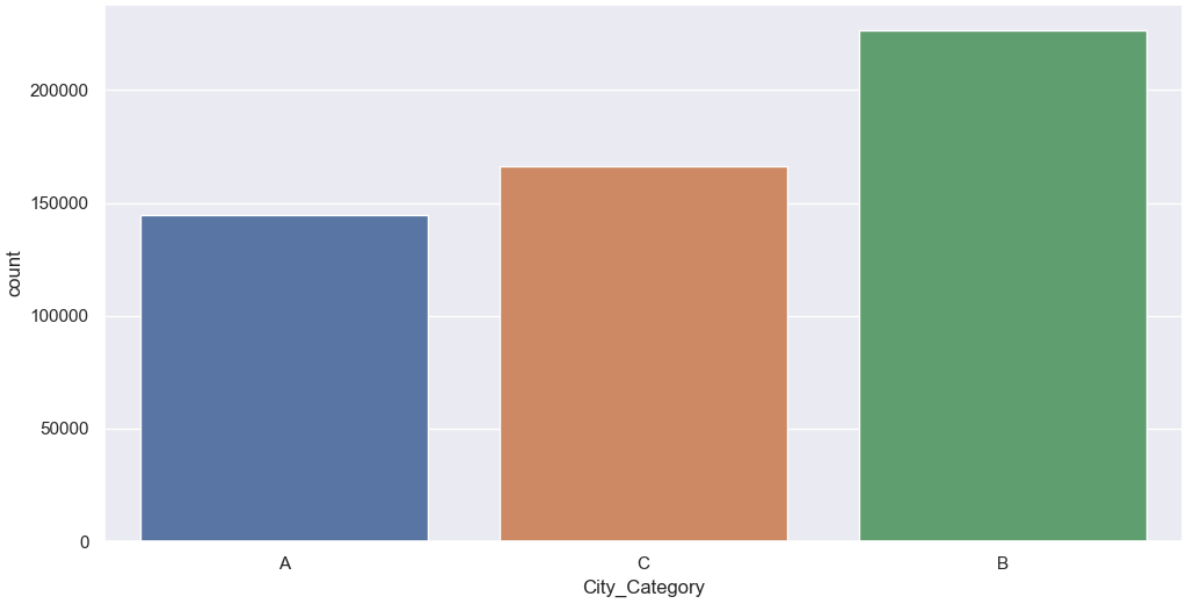
In [132…
```
sns.set(rc={"figure.figsize":(12,6)})
sns.countplot(x='Marital_Status',hue='Gender',data=df)
```

Out[132]: <Axes: xlabel='Marital_Status', ylabel='count'>



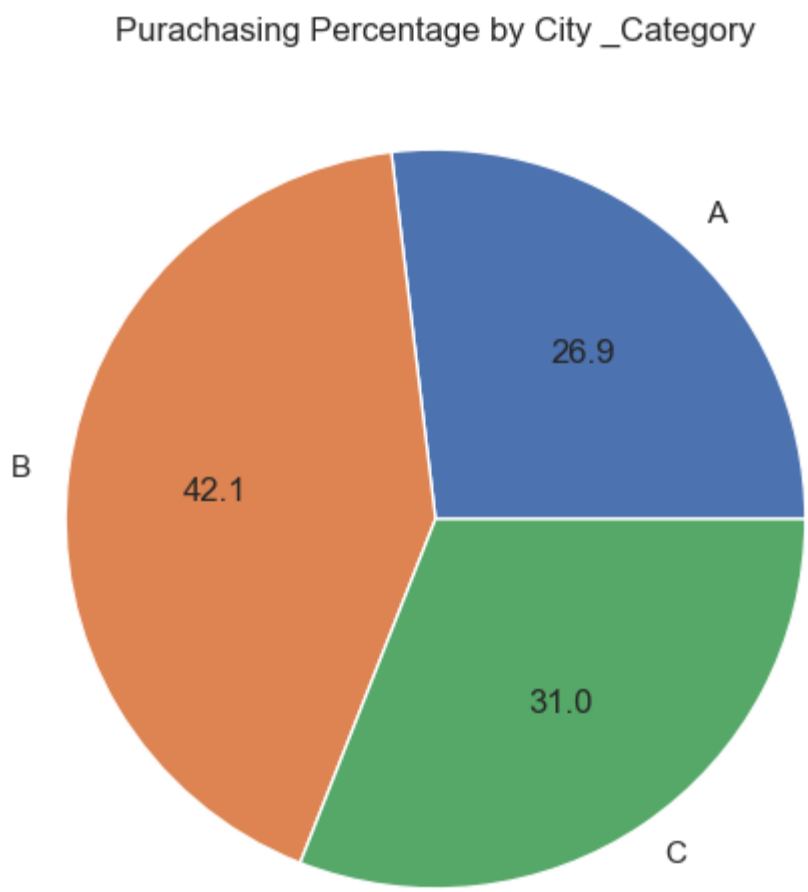
In [133…
```
sns.countplot(x=df["City_Category"])
```

Out[133]:
```
<Axes: xlabel='City_Category', ylabel='count'>
```



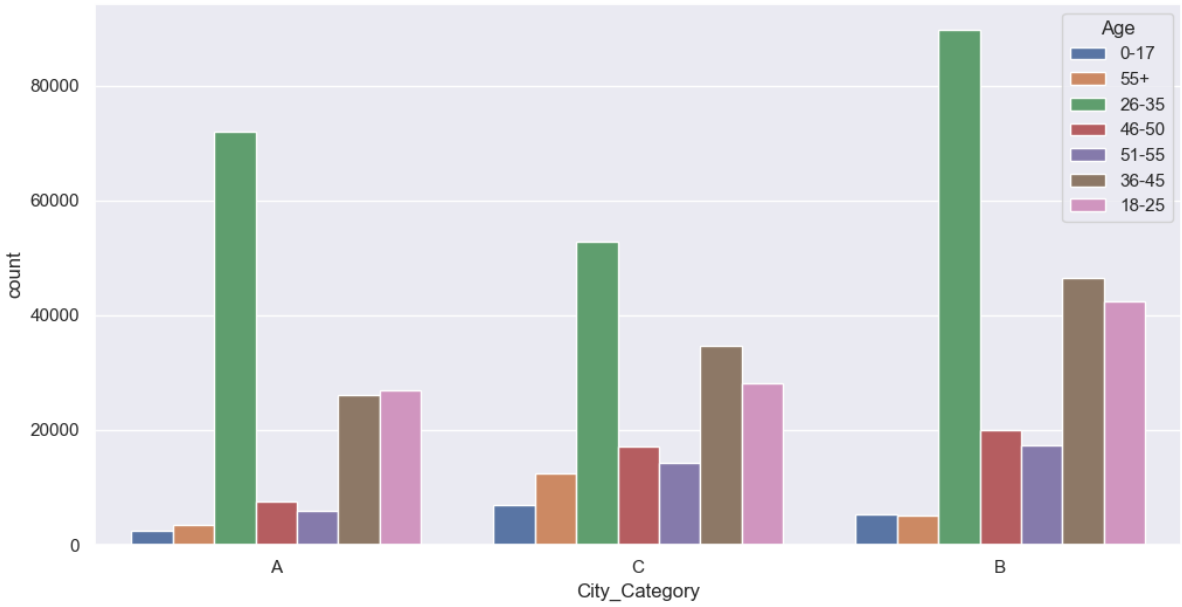
In [135…
```
df.groupby('City_Category').size().plot(kind='pie',figsize=(6,6),autopct='%0.1f',ti
```

Out[135]:
```
<Axes: title={'center': 'Purachasing Percentage by City _Category'}>
```



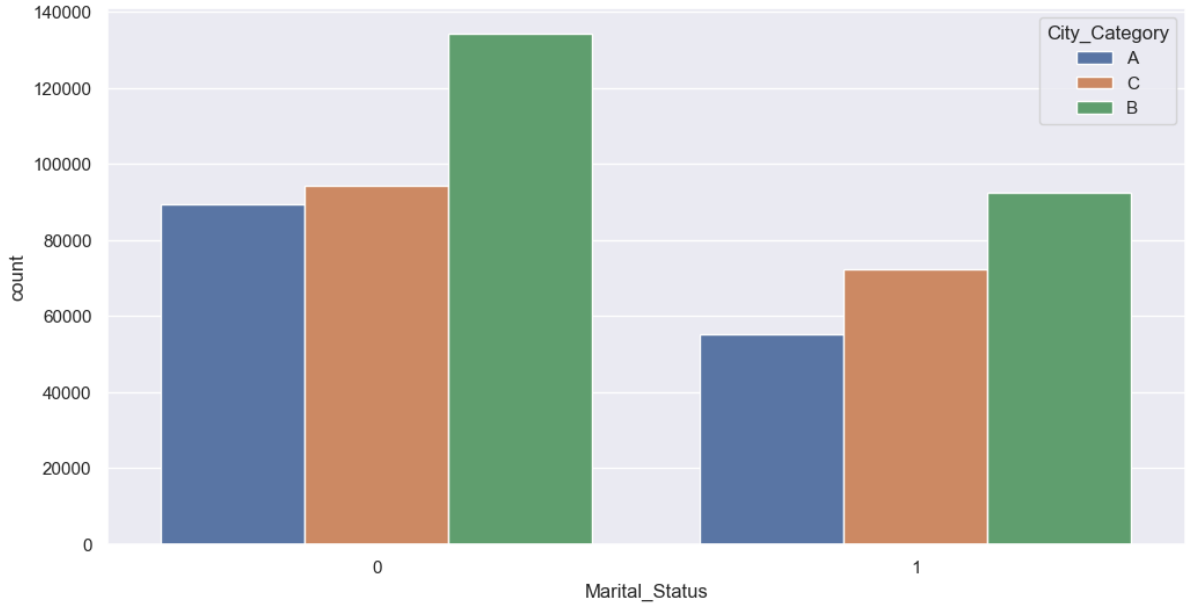
In [136…
```
sns.countplot(x='City_Category',hue='Age',data=df)
```

Out[136]:
```
<Axes: xlabel='City_Category', ylabel='count'>
```



In [138…
```
sns.countplot(x='Marital_Status',hue='City_Category',data=df)
```

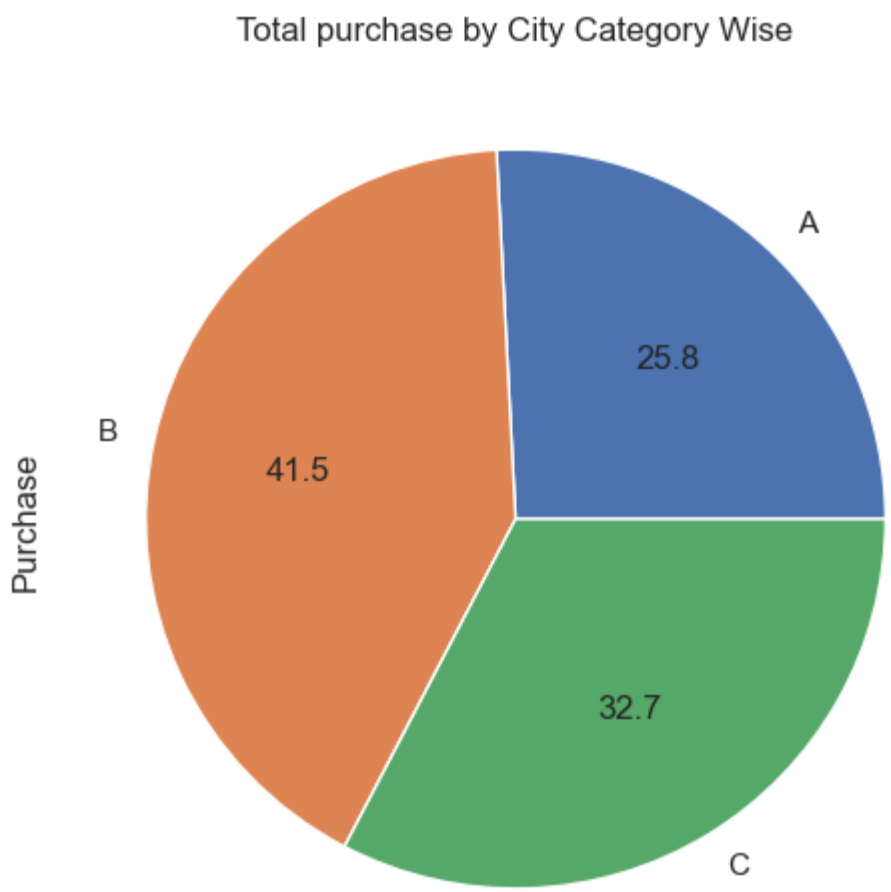
Out[138]:
```
<Axes: xlabel='Marital_Status', ylabel='count'>
```



In [149…
```
df.groupby("City_Category").sum()['Purchase'].plot(kind='pie',autopct='%0.1f',figsi
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\1638197614.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby("City_Category").sum()['Purchase'].plot(kind='pie',autopct='%0.1f',fig
size=(6,6),title="Total purchase by City Category Wise")
```

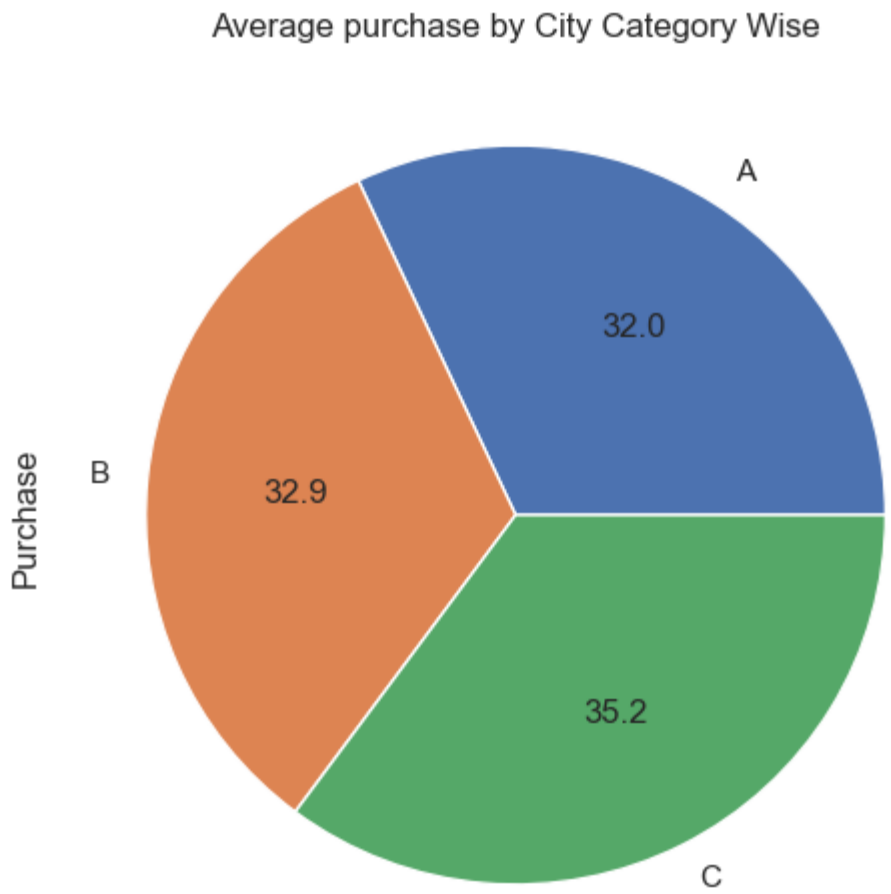
Out[149]:
```
<Axes: title={'center': 'Total purchase by City Category Wise'}, ylabel='Purchase'
>
```



In [150…
```
df.groupby("City_Category").mean()['Purchase'].plot(kind='pie',autopct='%0.1f',figs
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\3878864394.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futu
re version, numeric_only will default to False. Either specify numeric_only or sele
ct only columns which should be valid for the function.
  df.groupby("City_Category").mean()['Purchase'].plot(kind='pie',autopct='%0.1f',fi
gsize=(6,6),title="Average purchase by City Category Wise")
```

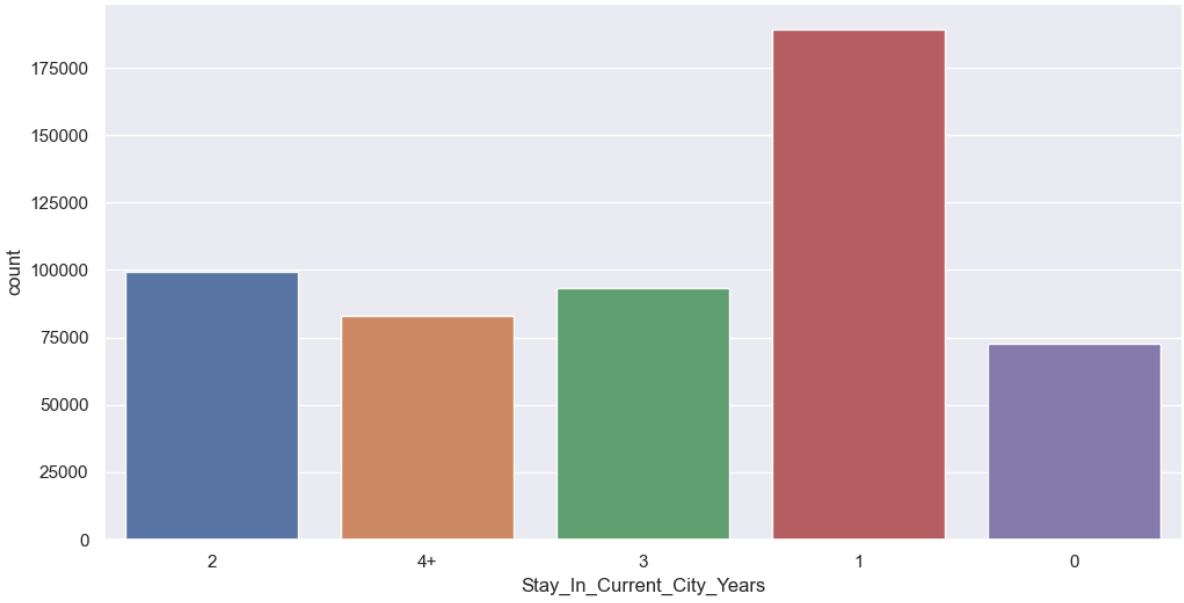
Out[150]:
```
<Axes: title={'center': 'Average purchase by City Category Wise'}, ylabel='Purchas
e'>
```



Ocupation and Product Analysis

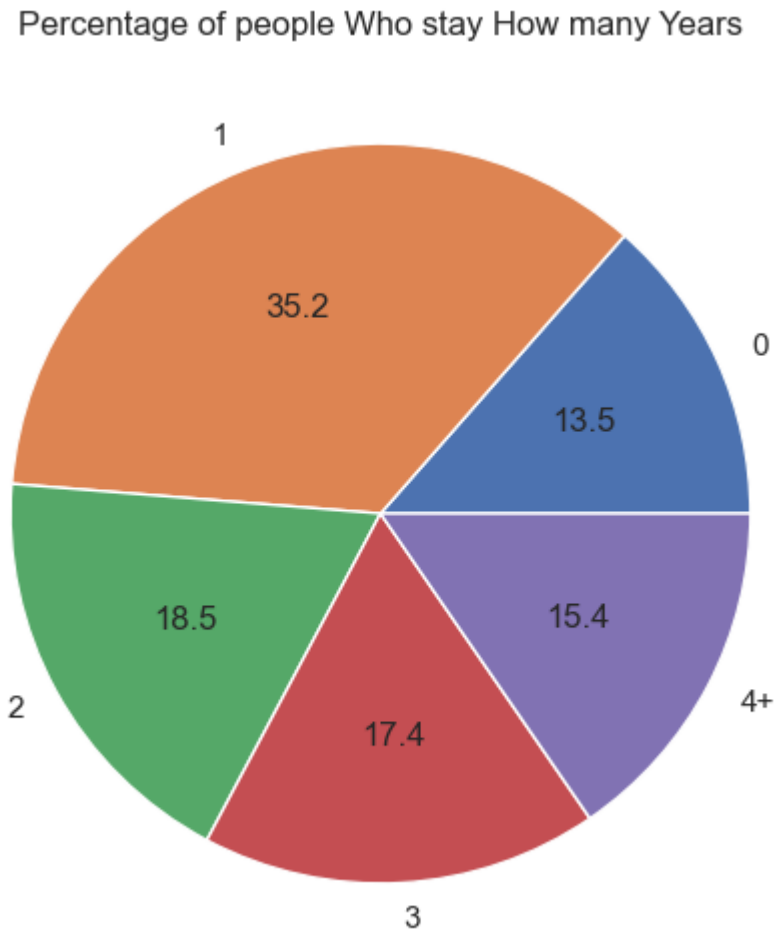
In [152…
```
sns.countplot(x=df['Stay_In_Current_City_Years'])
```

Out[152]:
```
<Axes: xlabel='Stay_In_Current_City_Years', ylabel='count'>
```



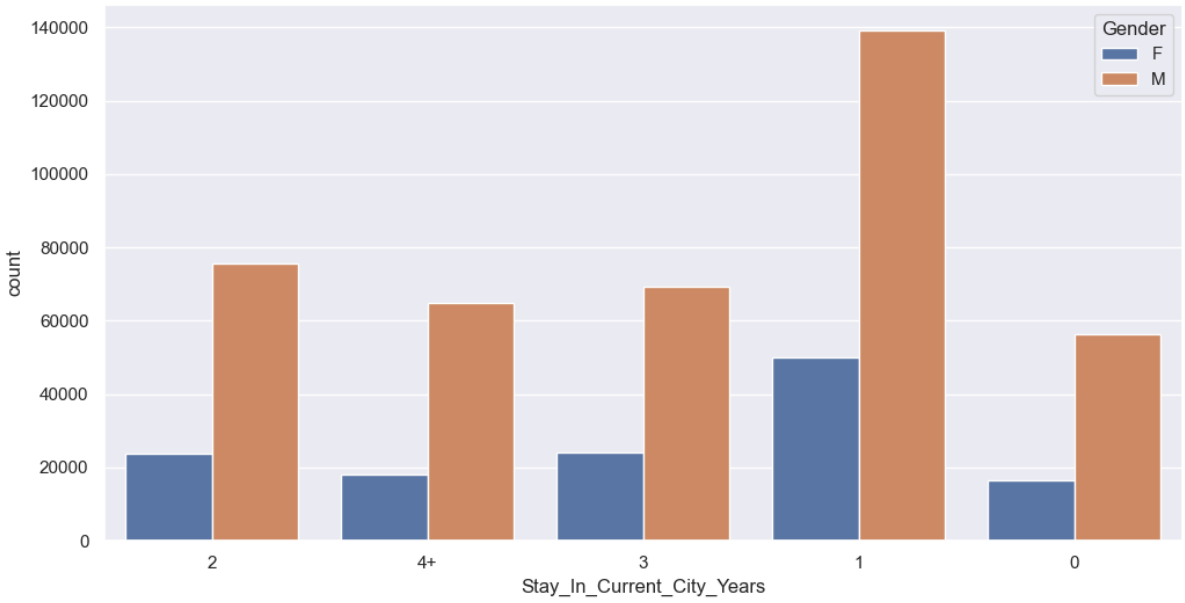
In [166…
```
df.groupby("Stay_In_Current_City_Years").size().plot(kind='pie',figsize=(6,6),autop
```

Out[166]:
```
<Axes: title={'center': 'Percentage of people Who stay How many Years'}>
```



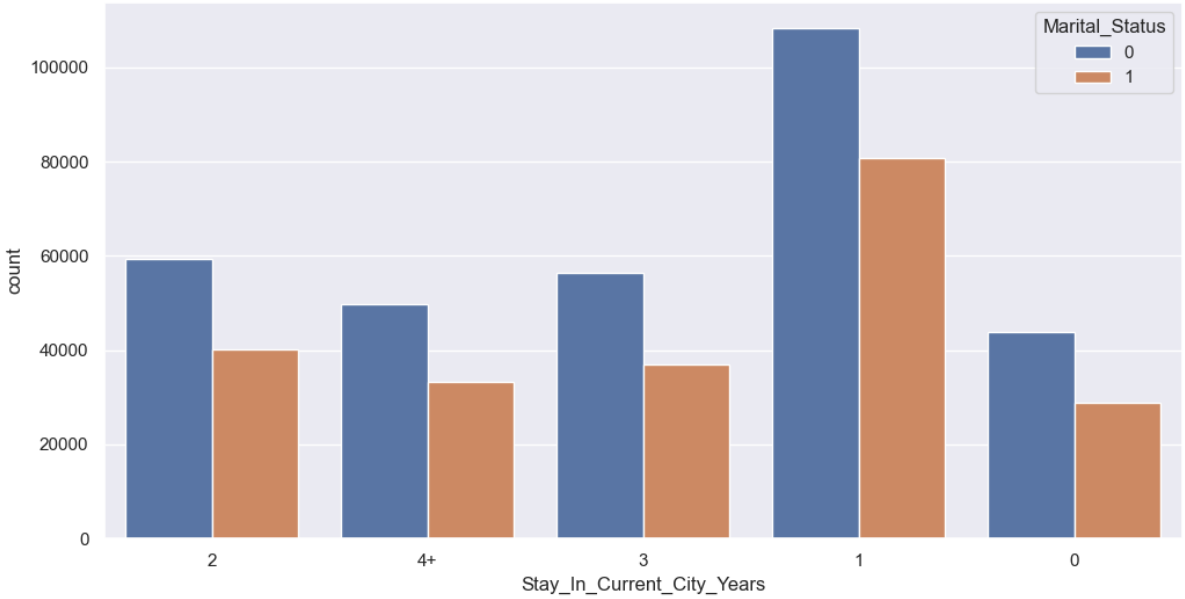
In [153…
```
sns.countplot(x='Stay_In_Current_City_Years',hue='Gender',data=df)
```

Out[153]:
```
<Axes: xlabel='Stay_In_Current_City_Years', ylabel='count'>
```



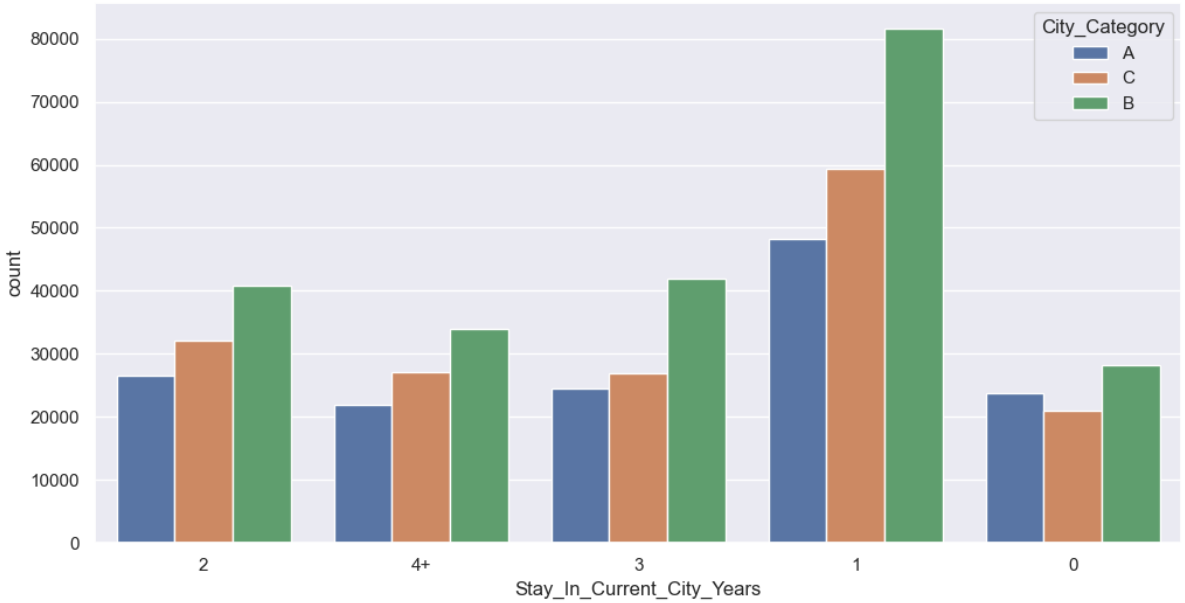
In [154…
```
sns.countplot(x='Stay_In_Current_City_Years',hue='Marital_Status',data=df)
```

Out[154]:
```
<Axes: xlabel='Stay_In_Current_City_Years', ylabel='count'>
```



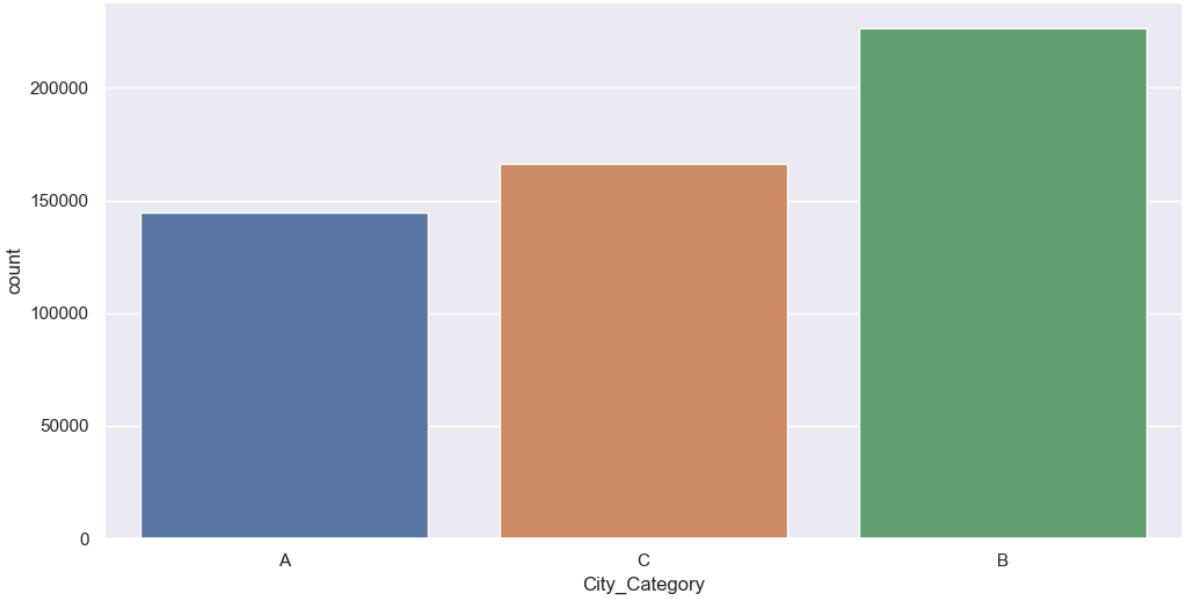
In [155…
```
sns.countplot(x='Stay_In_Current_City_Years',hue='City_Category',data=df)
```

Out[155]:
```
<Axes: xlabel='Stay_In_Current_City_Years', ylabel='count'>
```



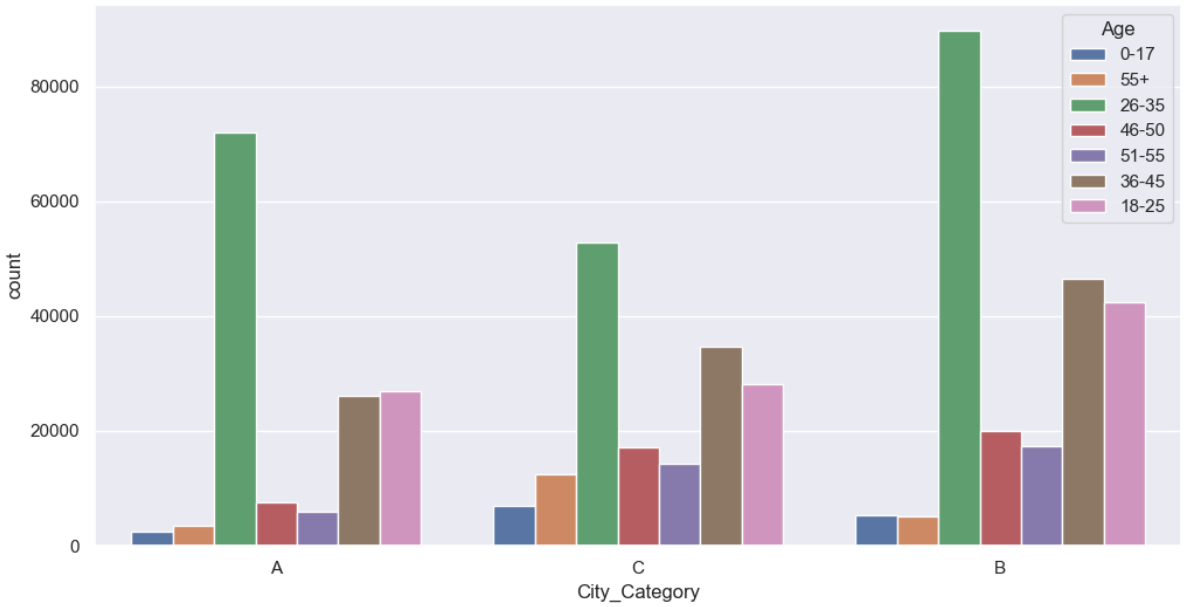
In [156…
```
sns.countplot(x='City_Category',data=df)
```

Out[156]:
```
<Axes: xlabel='City_Category', ylabel='count'>
```



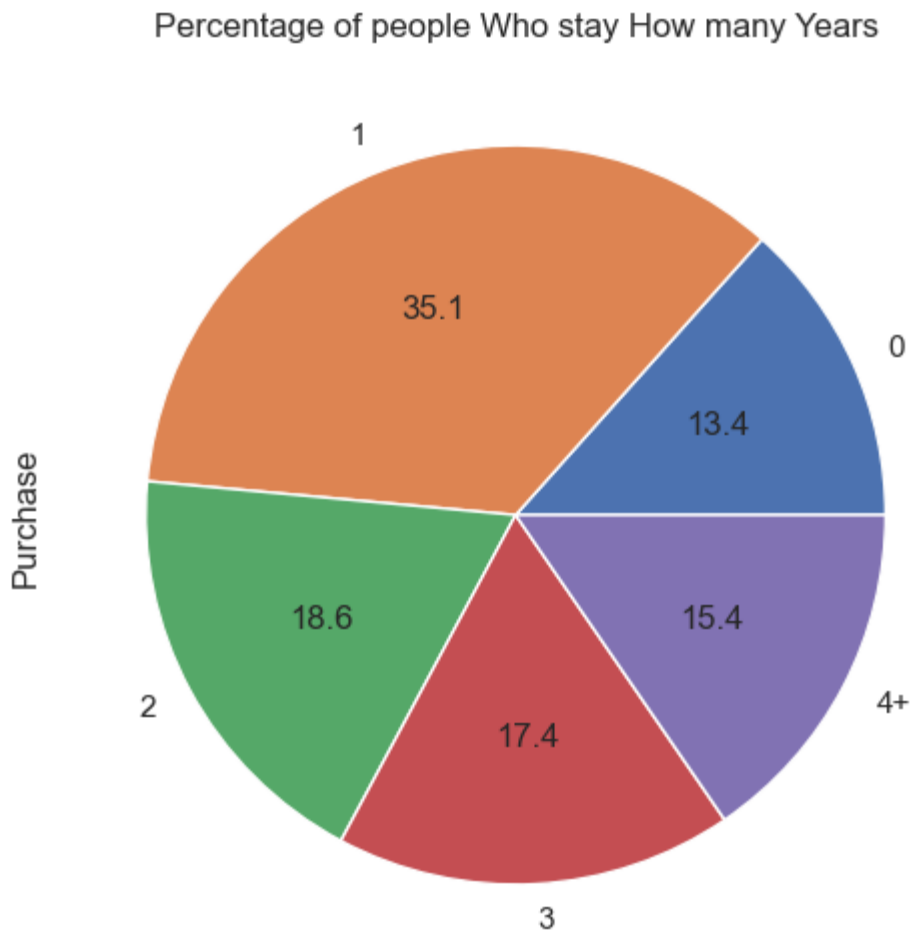
In [157…
```
sns.countplot(x='City_Category',hue='Age',data=df)
```

Out[157]:
```
<Axes: xlabel='City_Category', ylabel='count'>
```



In [167…
```
df.groupby("Stay_In_Current_City_Years").sum()["Purchase"].plot(kind='pie',figsize=
```

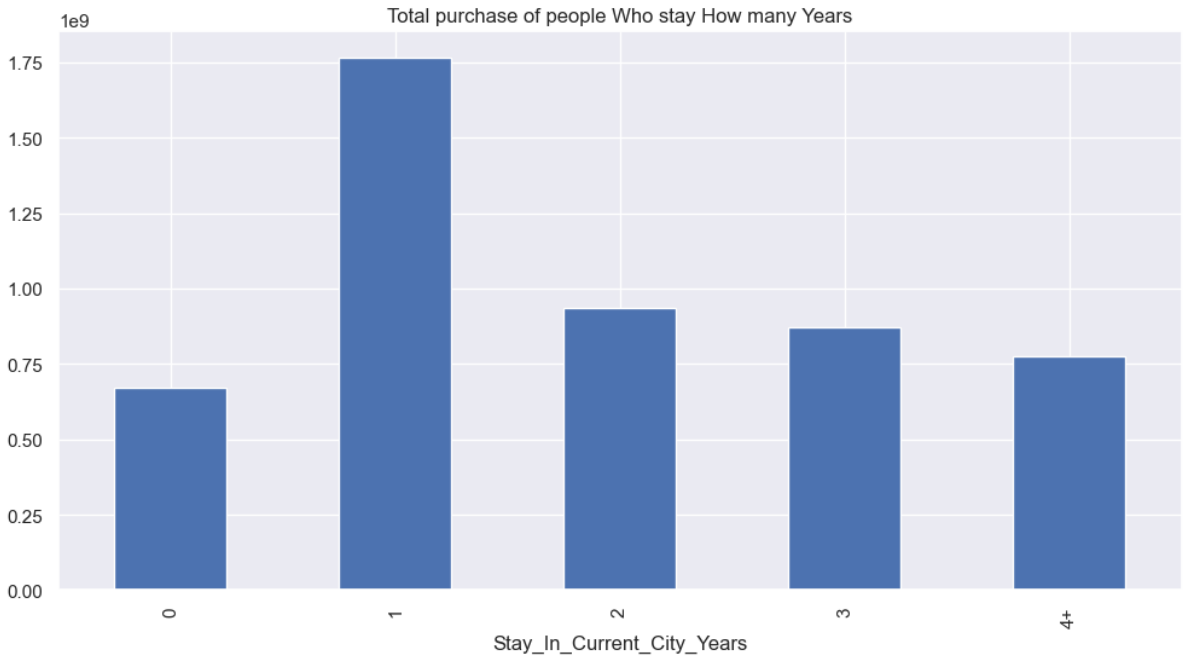
Out[167]:
```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\153402742.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select
only columns which should be valid for the function.
  df.groupby("Stay_In_Current_City_Years").sum()["Purchase"].plot(kind='pie',figsiz
e=(6,6),autopct='%0.1f',title="Percentage of people Who stay How many Years")
<Axes: title={'center': 'Percentage of people Who stay How many Years'}, ylabel='P
urchase'>
```



In [170…
```
df.groupby("Stay_In_Current_City_Years").sum()["Purchase"].plot(kind='bar',title="T
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\684859561.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select
only columns which should be valid for the function.
  df.groupby("Stay_In_Current_City_Years").sum()["Purchase"].plot(kind='bar',title
="Total purchase of people Who stay How many Years")
```

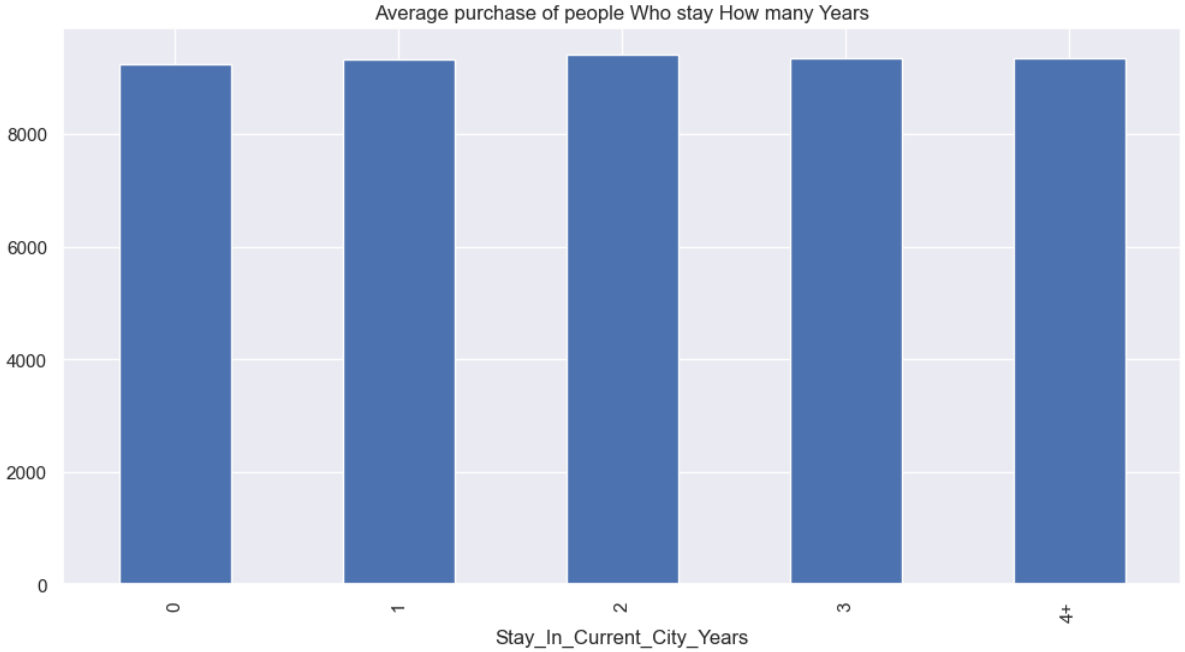
Out[170]:
```
<Axes: title={'center': 'Total purchase of people Who stay How many Years'}, xlabe
l='Stay_In_Current_City_Years'>
```



In [172…
```
df.groupby("Stay_In_Current_City_Years").mean()["Purchase"].plot(kind='bar',title="
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\3874541039.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futu
re version, numeric_only will default to False. Either specify numeric_only or sele
ct only columns which should be valid for the function.
  df.groupby("Stay_In_Current_City_Years").mean()["Purchase"].plot(kind='bar',title
="Average purchase of people Who stay How many Years")
```

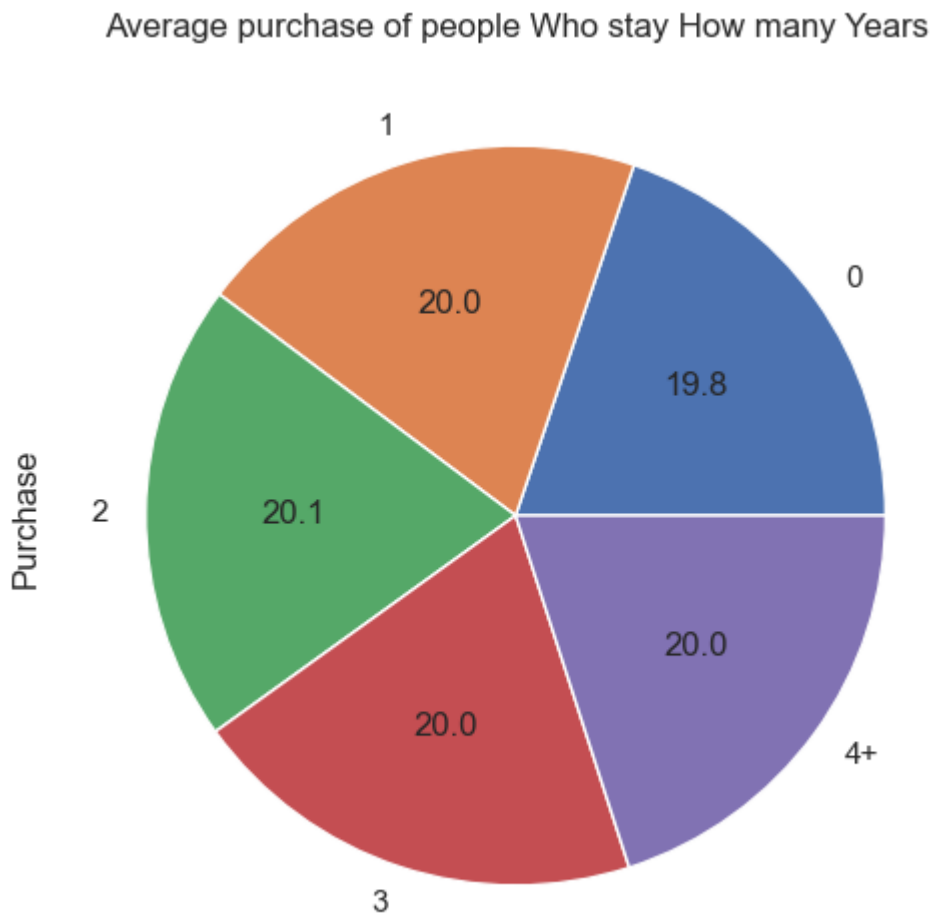
Out[172]:
```
<Axes: title={'center': 'Average purchase of people Who stay How many Years'}, xla
bel='Stay_In_Current_City_Years'>
```



In [173…
```
df.groupby("Stay_In_Current_City_Years").mean()["Purchase"].plot(kind='pie',figsize
```

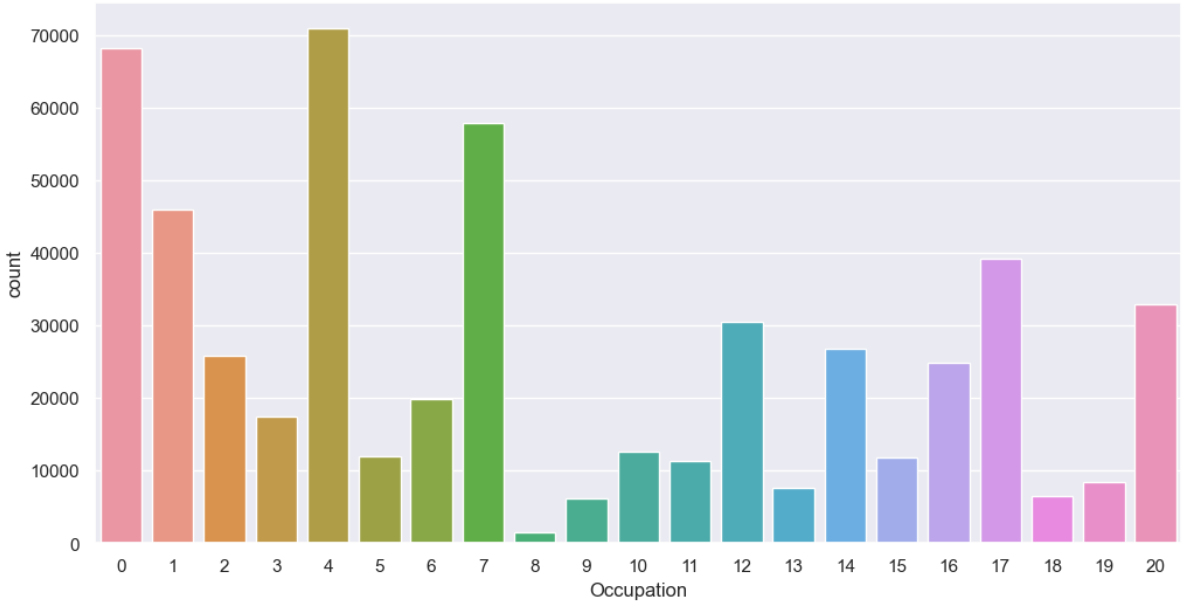
```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\594988940.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby("Stay_In_Current_City_Years").mean()["Purchase"].plot(kind='pie',figsi
ze=(6,6),autopct='%0.1f',title="Average purchase of people Who stay How many Year
s")
```

Out[173]: <Axes: title={'center': 'Average purchase of people Who stay How many Years'}, yla
bel='Purchase'>



In [176…
```
sns.countplot(x = df['Occupation'])
```

Out[176]: <Axes: xlabel='Occupation', ylabel='count'>

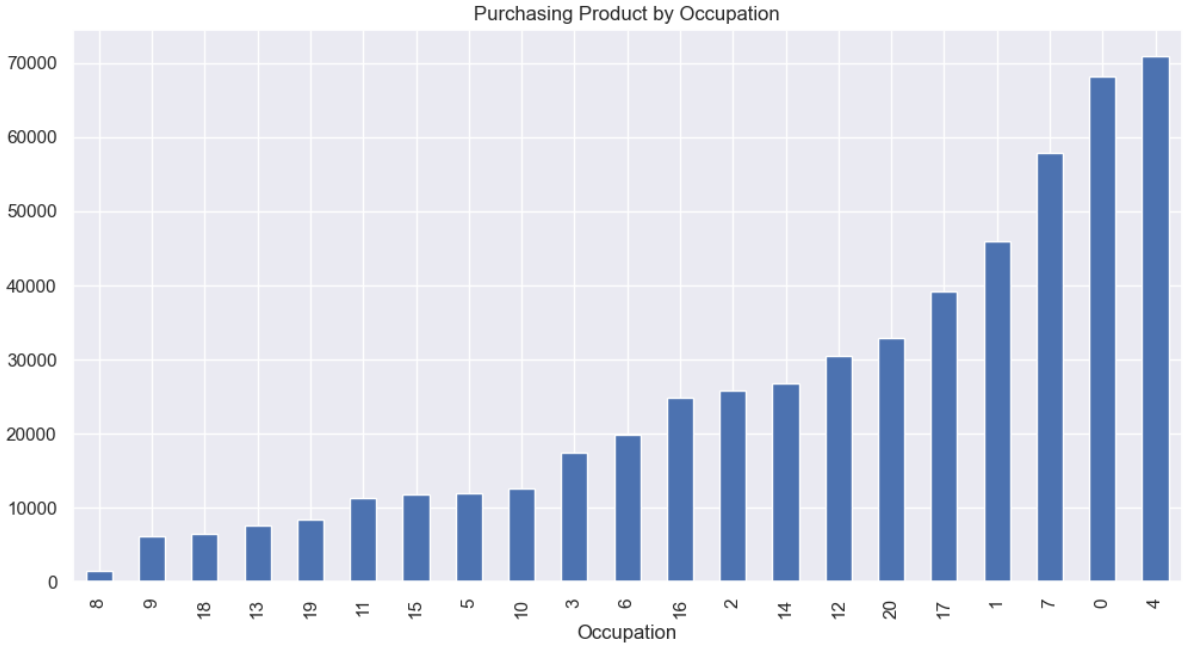


In [177…
```
df.groupby('Occupation').size()
```

Out[177]:
```
Occupation
0     68120
1     45971
2     25845
3     17366
4     70862
5     11985
6     19822
7     57806
8      1524
9      6153
10    12623
11    11338
12    30423
13     7548
14    26712
15    11812
16    24790
17    39090
18     6525
19     8352
20    32910
dtype: int64
```

In [181…
```
df.groupby('Occupation').size().sort_values().plot(kind='bar',title="Purchasing Pro
```

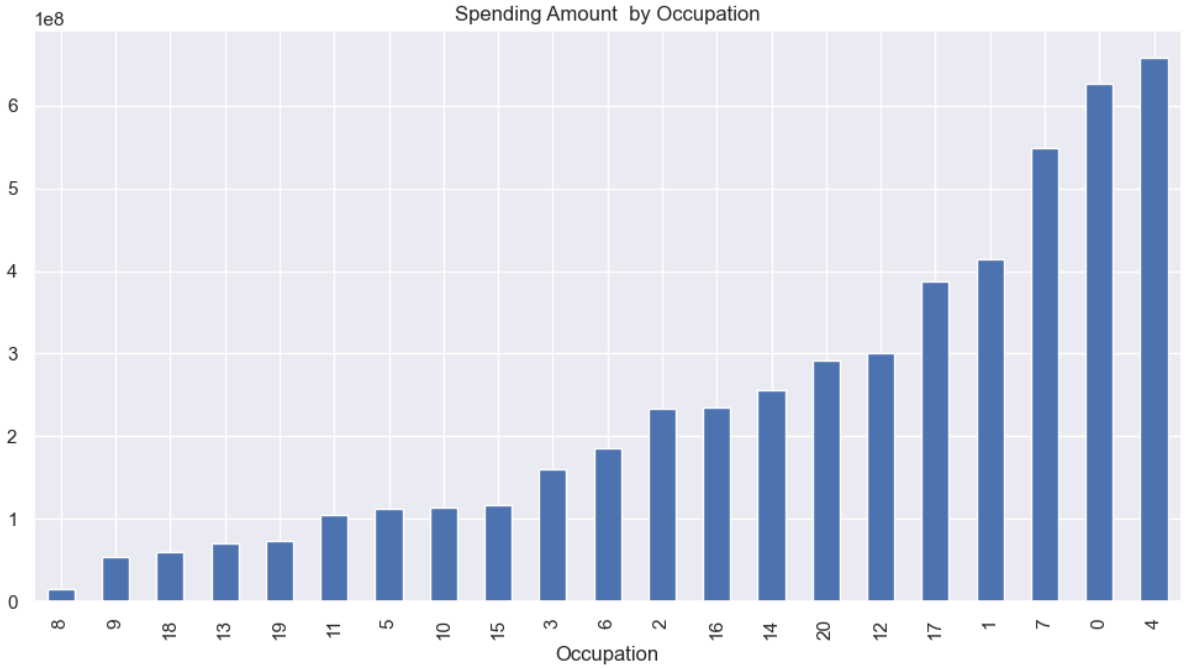
Out[181]:
```
<Axes: title={'center': 'Purchasing Product by Occupation'}, xlabel='Occupation'>
```



In [183…
```
df.groupby('Occupation').sum()['Purchase'].sort_values().plot(kind='bar',title="Spe
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\4230847904.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby('Occupation').sum()['Purchase'].sort_values().plot(kind='bar',title="S
pending Amount  by Occupation")
```

Out[183]:
```
<Axes: title={'center': 'Spending Amount  by Occupation'}, xlabel='Occupation'>
```

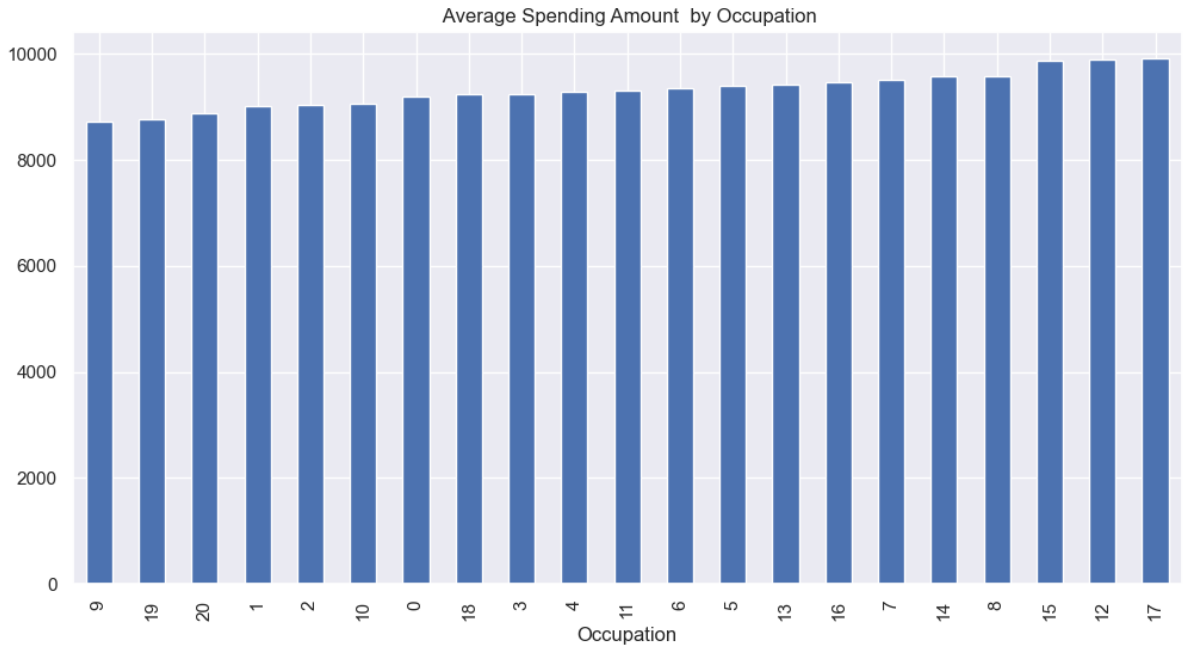


In [184…
```
df.groupby('Occupation').mean()['Purchase'].sort_values().plot(kind='bar',title=" A
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\656845213.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby('Occupation').mean()['Purchase'].sort_values().plot(kind='bar',title="
Average Spending Amount  by Occupation")
```

Out[184]:

```
<Axes: title={'center': ' Average Spending Amount  by Occupation'}, xlabel='Occupa
tion'>
```



In [185…

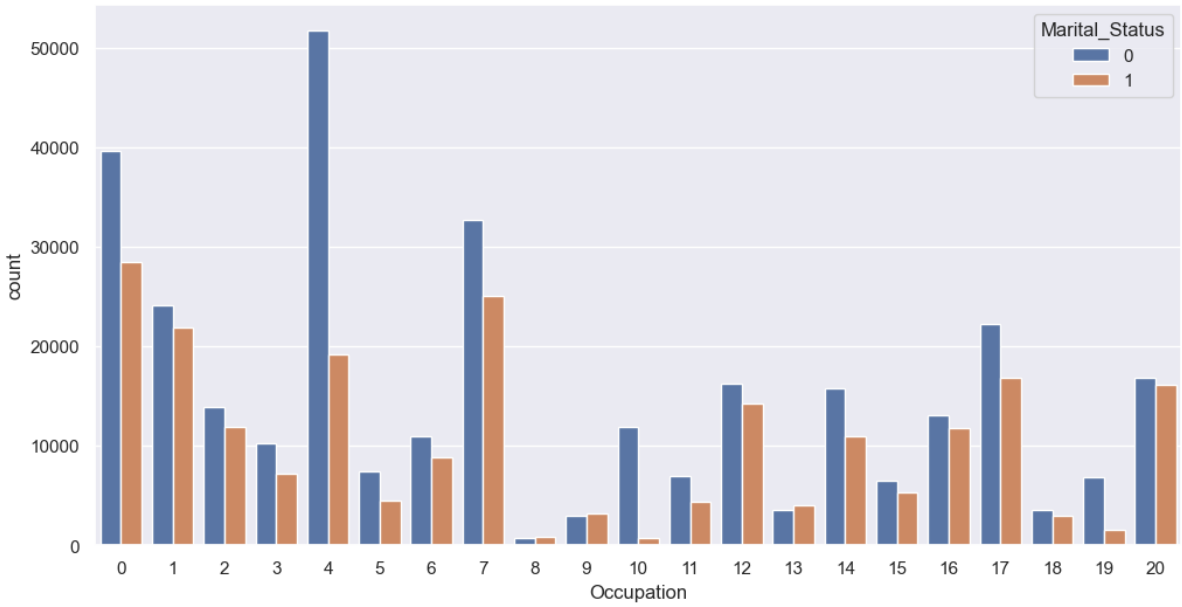
```
df.groupby('Occupation').mean()['Purchase'].sort_values()
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_13456\3915541376.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futu
re version, numeric_only will default to False. Either specify numeric_only or sele
ct only columns which should be valid for the function.
  df.groupby('Occupation').mean()['Purchase'].sort_values()
```

Out[185]:
```
Occupation
9     8714.335934
19    8754.249162
20    8881.099514
1     9017.703095
2     9025.938982
10    9052.836410
0     9186.946726
18    9233.671418
3     9238.077277
4     9279.026742
11    9299.467190
6     9336.378620
5     9388.848978
13    9424.449391
16    9457.133118
7     9502.175276
14    9568.536426
8     9576.508530
15    9866.239925
12    9883.052460
17    9906.378997
Name: Purchase, dtype: float64
```

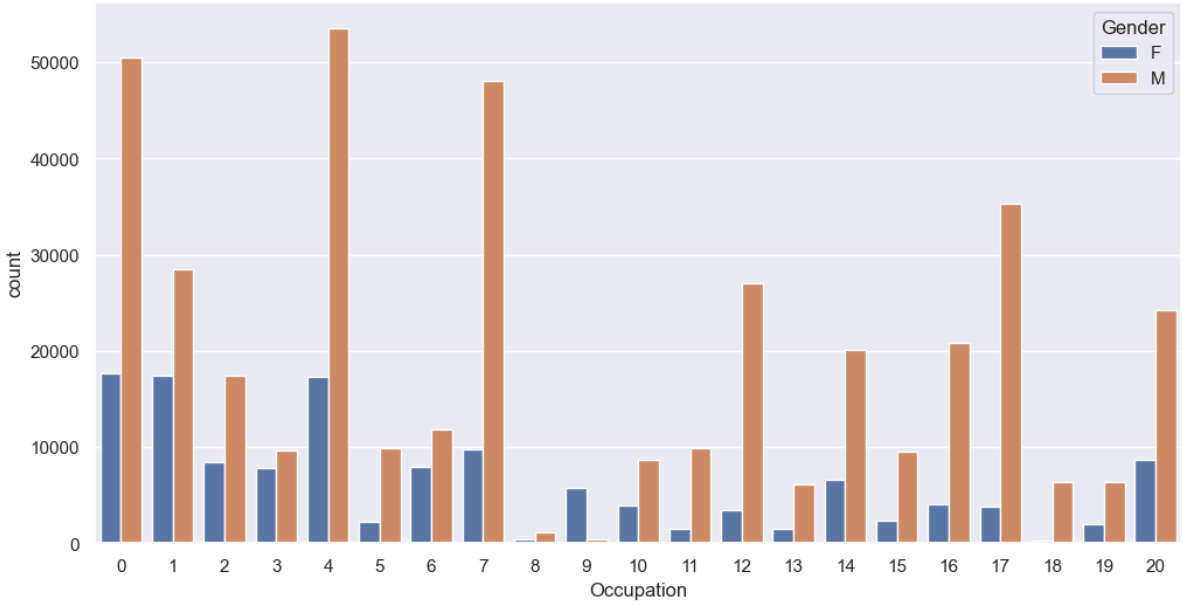
In [186…
```
sns.countplot(x='Occupation',hue='Marital_Status',data=df)
```

Out[186]:
```
<Axes: xlabel='Occupation', ylabel='count'>
```



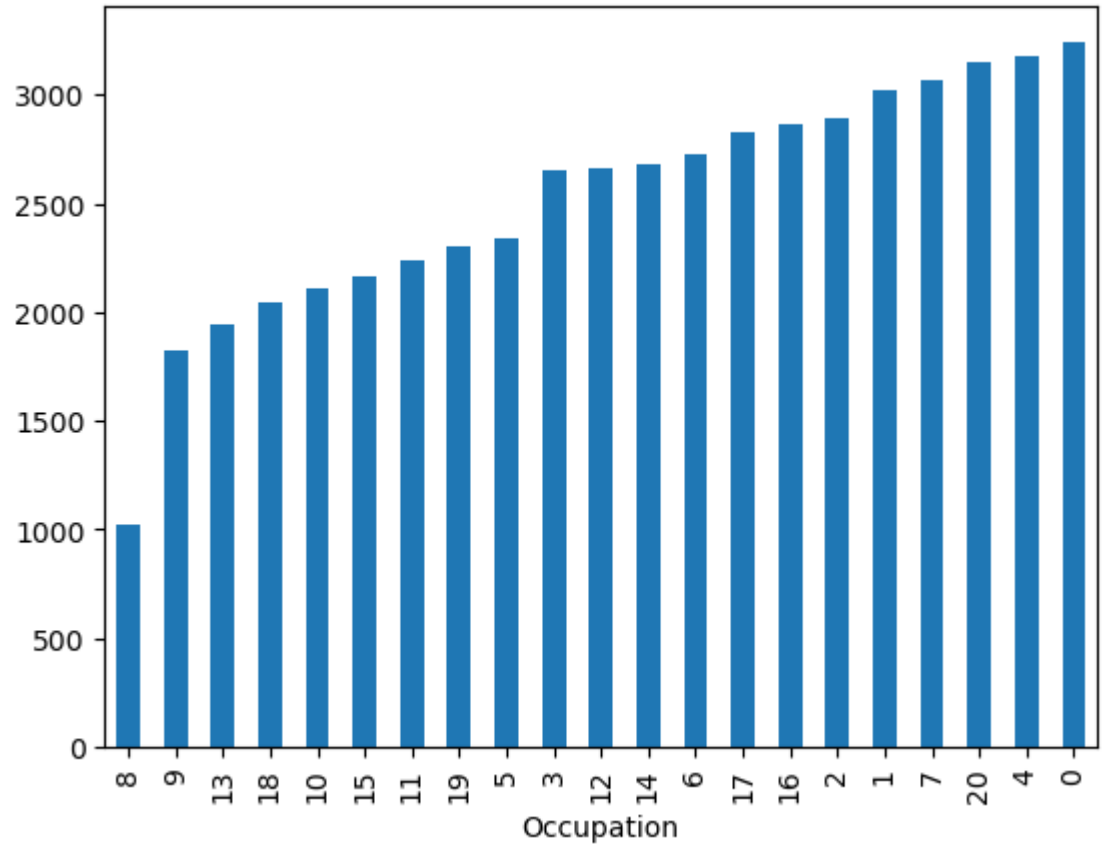
In [187…
```
sns.countplot(x='Occupation',hue='Gender',data=df)
```

Out[187]:
```
<Axes: xlabel='Occupation', ylabel='count'>
```



In [17]:
```
df.groupby('Occupation').nunique()['Product_ID'].sort_values().plot(kind='bar')
```

Out[17]:
```
<Axes: xlabel='Occupation'>
```

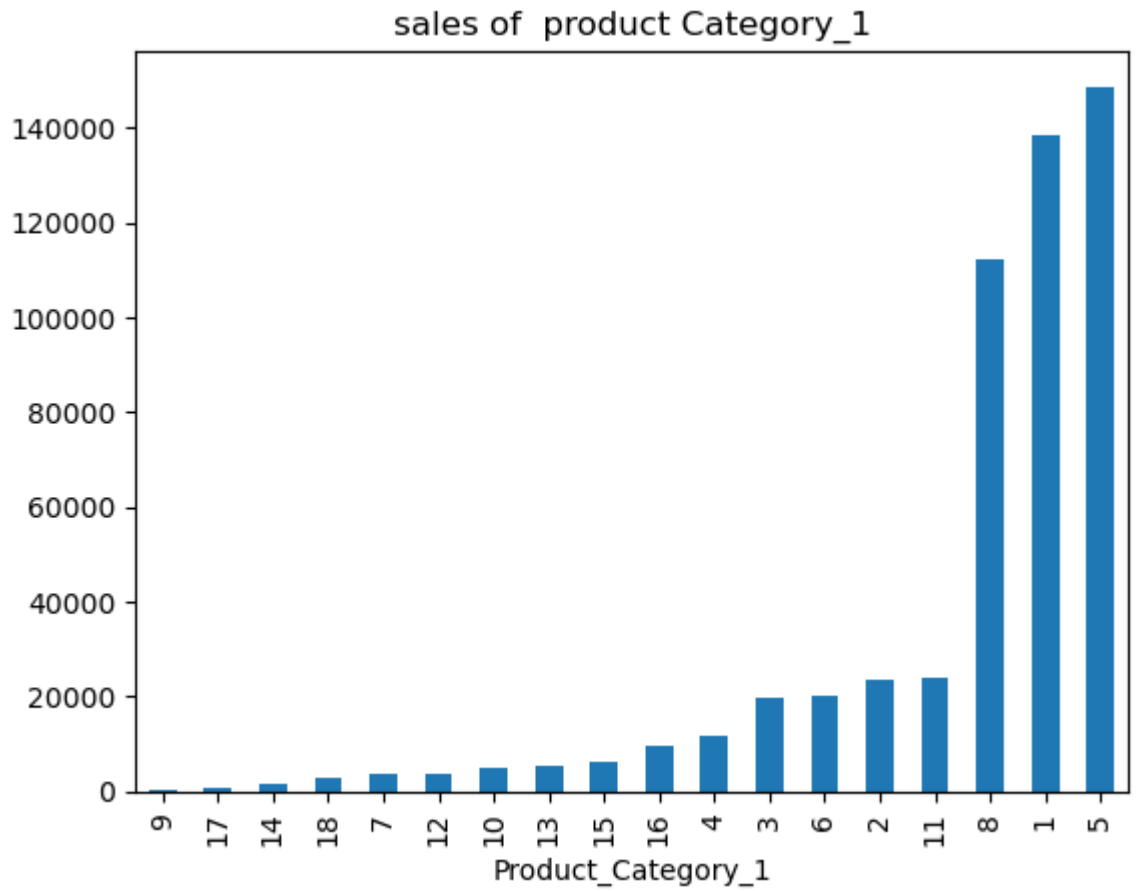


In [19]:
```
df.groupby('Product_Category_1').size()
```

Out[19]:
```
Product_Category_1
1     138353
2      23499
3      19849
4      11567
5     148592
6      20164
7       3668
8     112132
9        404
10      5032
11     23960
12      3875
13      5440
14      1500
15      6203
16      9697
17       567
18      3075
dtype: int64
```

In [21]:
```
df.groupby('Product_Category_1').size().sort_values().plot(kind='bar',title ='sales
```

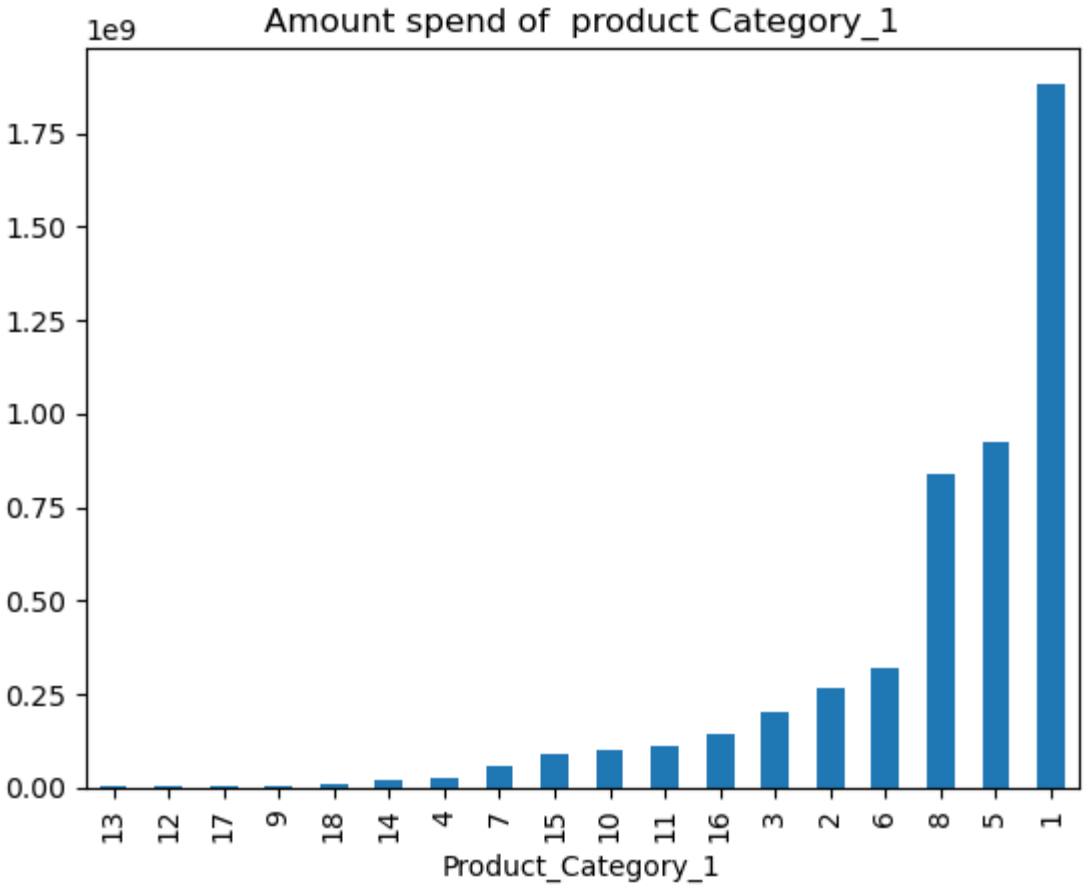
Out[21]:
```
<Axes: title={'center': 'sales of  product Category_1'}, xlabel='Product_Category_1
'>
```



In [26]:
```
df.groupby('Product_Category_1').sum()['Purchase'].sort_values().plot(kind='bar',ti
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_8424\1936599419.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select
only columns which should be valid for the function.
  df.groupby('Product_Category_1').sum()['Purchase'].sort_values().plot(kind='bar',
title ='Amount spend of  product Category_1')
```

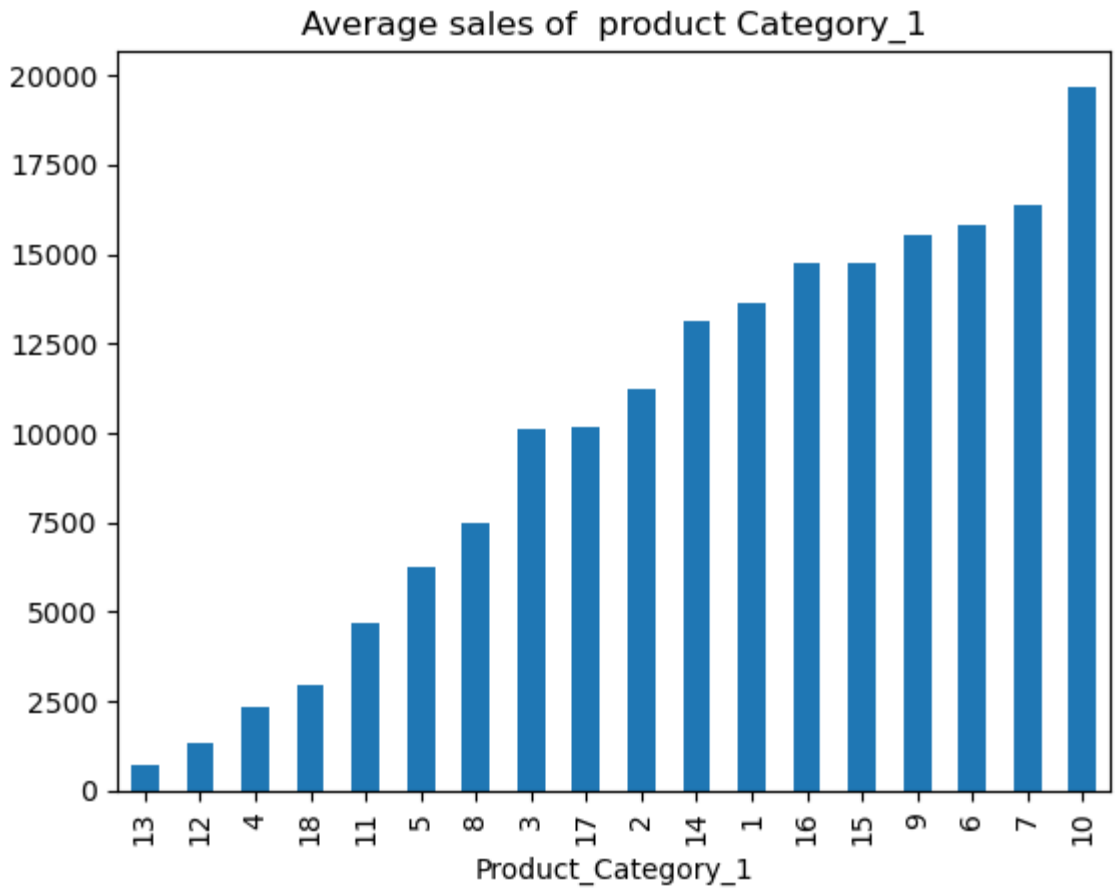
Out[26]:
```
<Axes: title={'center': 'Amount spend of  product Category_1'}, xlabel='Product_Cat
egory_1'>
```



In [25]:
```
df.groupby('Product_Category_1').mean()['Purchase'].sort_values().plot(kind='bar',t
```

```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_8424\3786040791.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby('Product_Category_1').mean()['Purchase'].sort_values().plot(kind='bar
',title ='Average sales of  product Category_1')
```

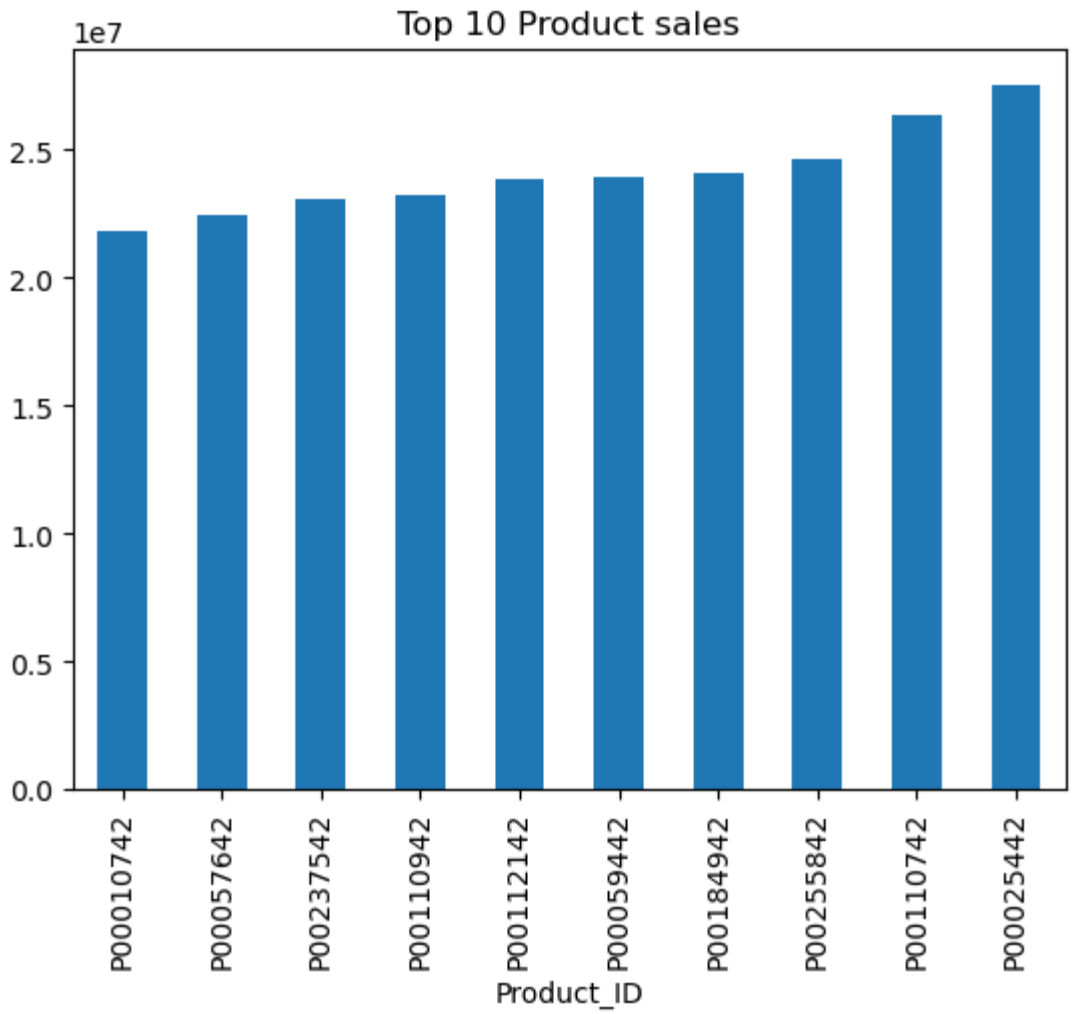
Out[25]:
```
<Axes: title={'center': 'Average sales of  product Category_1'}, xlabel='Product_Ca
tegory_1'>
```



In [29]:
```
df.groupby('Product_ID').sum()['Purchase'].nlargest(10).sort_values().plot(kind='ba
```

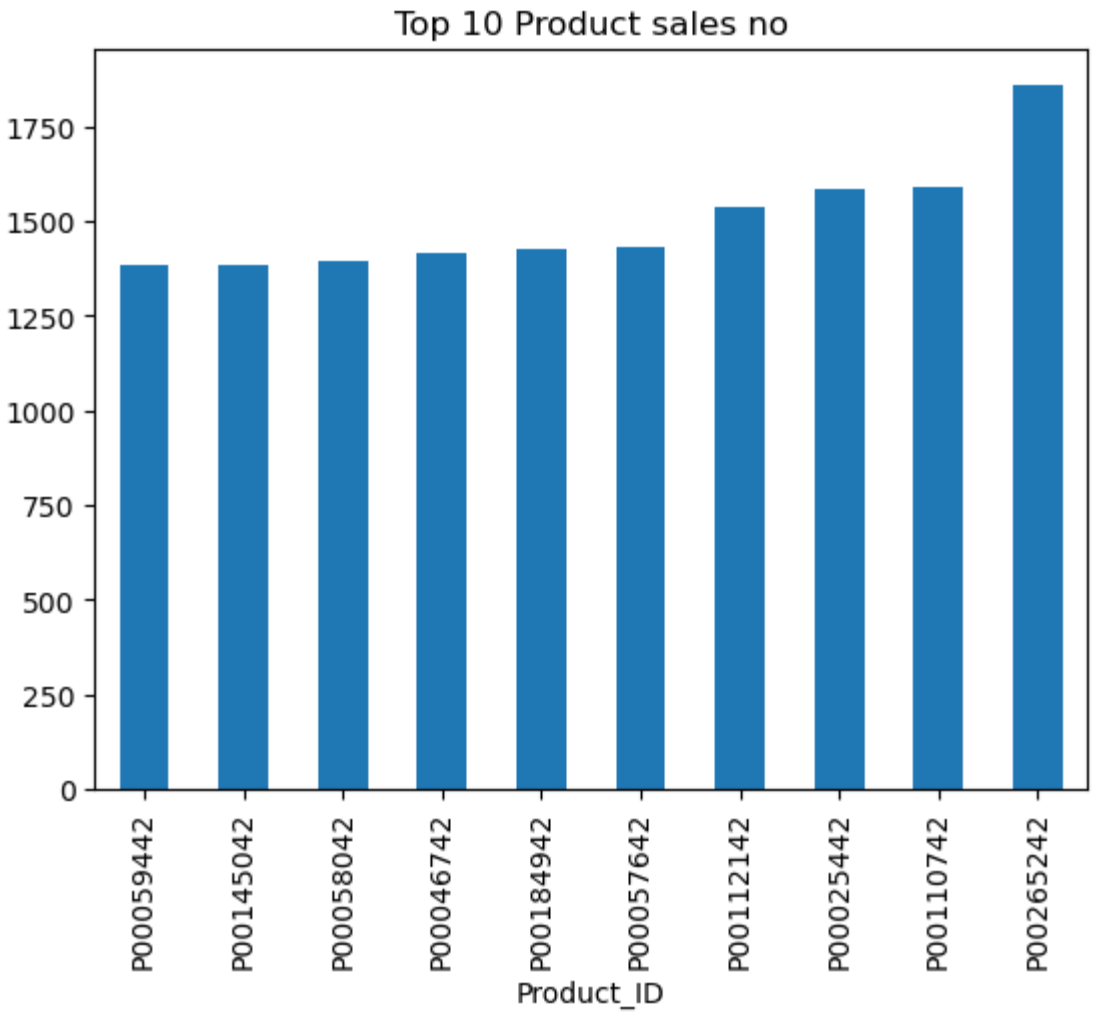
```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_8424\2871155693.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future
version, numeric_only will default to False. Either specify numeric_only or select
only columns which should be valid for the function.
  df.groupby('Product_ID').sum()['Purchase'].nlargest(10).sort_values().plot(kind='
bar',title ='Top 10 Product sales')
```

Out[29]:
```
<Axes: title={'center': 'Top 10 Product sales'}, xlabel='Product_ID'>
```



In [31]:
```
df.groupby('Product_ID').size().nlargest(10).sort_values().plot(kind='bar',title ='
```

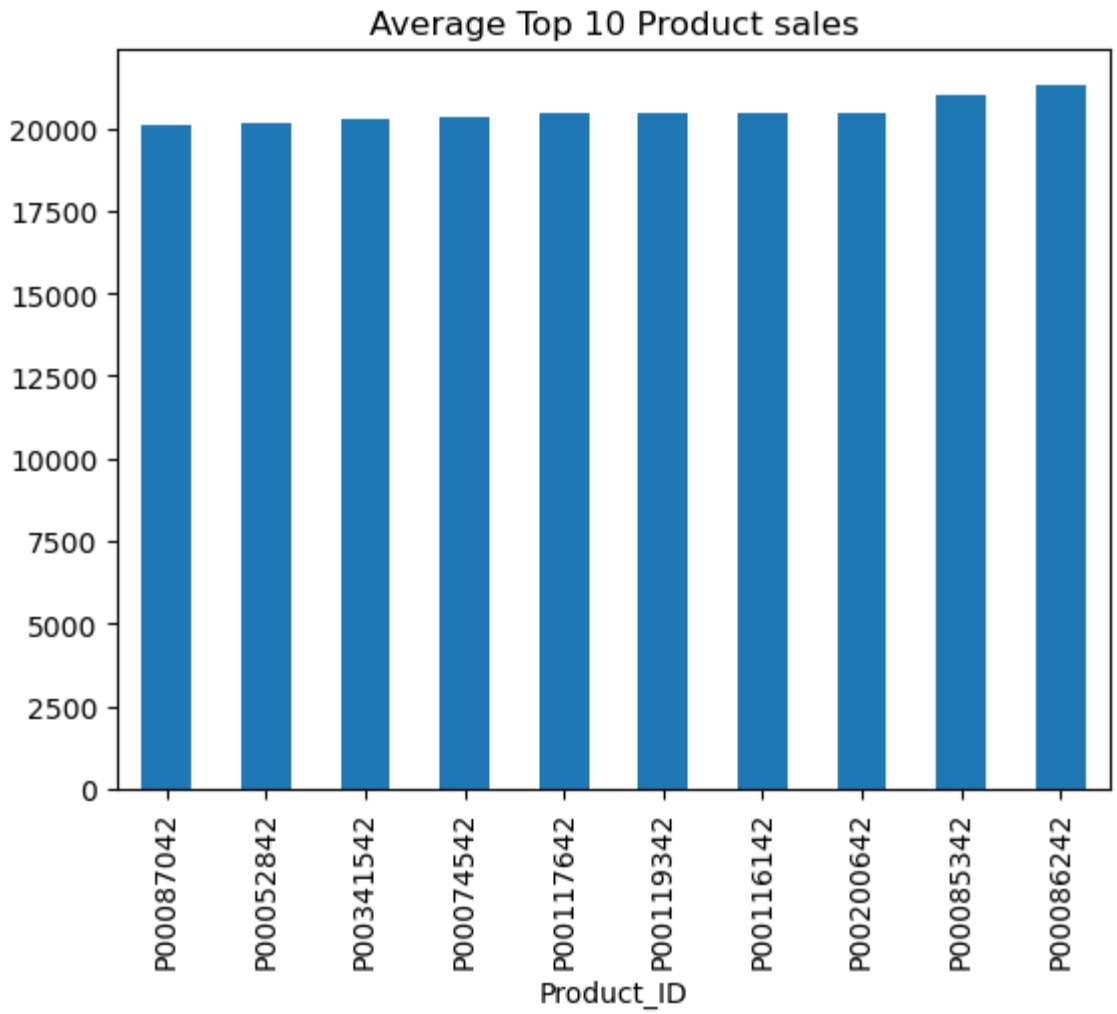
Out[31]:
```
<Axes: title={'center': 'Top 10 Product sales no'}, xlabel='Product_ID'>
```



In [32]:
```
df.groupby('Product_ID').mean()['Purchase'].nlargest(10).sort_values().plot(kind='b
```

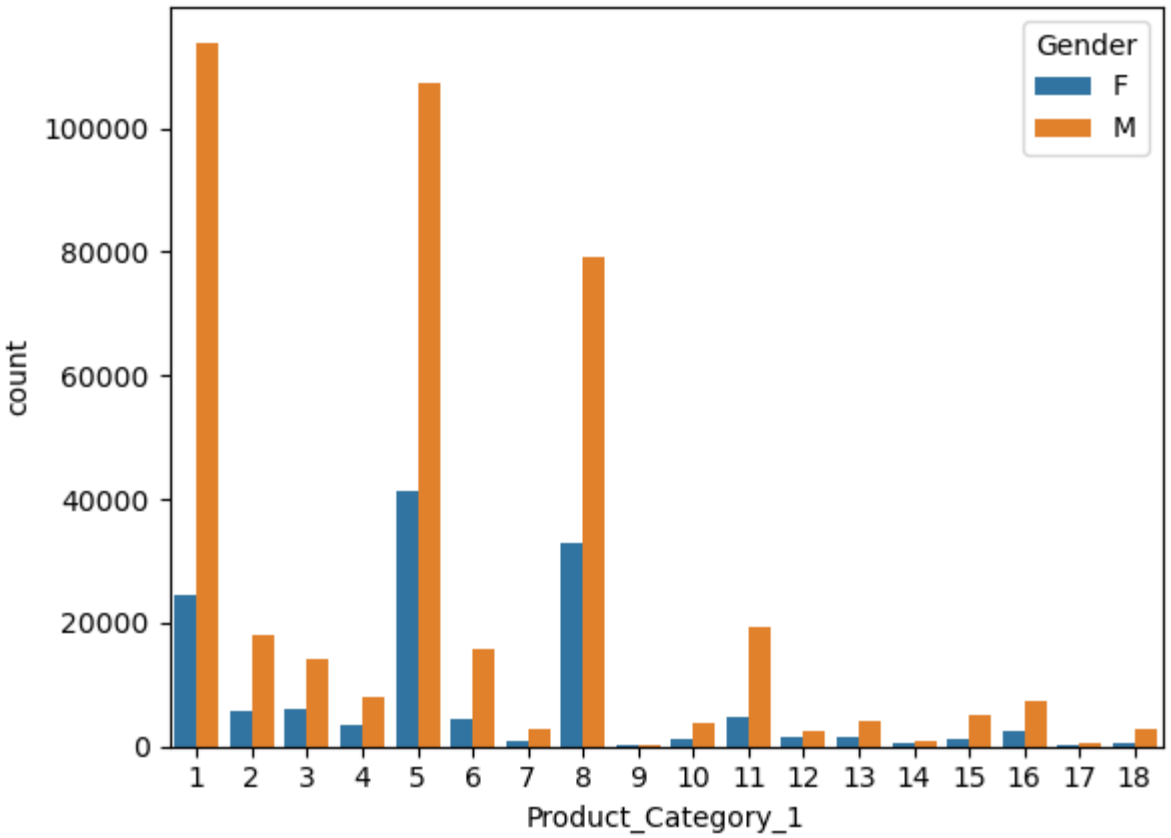
```
C:\Users\Shisht\AppData\Local\Temp\ipykernel_8424\2161939301.py:1: FutureWarning: T
he default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a futur
e version, numeric_only will default to False. Either specify numeric_only or selec
t only columns which should be valid for the function.
  df.groupby('Product_ID').mean()['Purchase'].nlargest(10).sort_values().plot(kind=
'bar',title ='Average Top 10 Product sales')
```

Out[32]:
```
<Axes: title={'center': 'Average Top 10 Product sales'}, xlabel='Product_ID'>
```



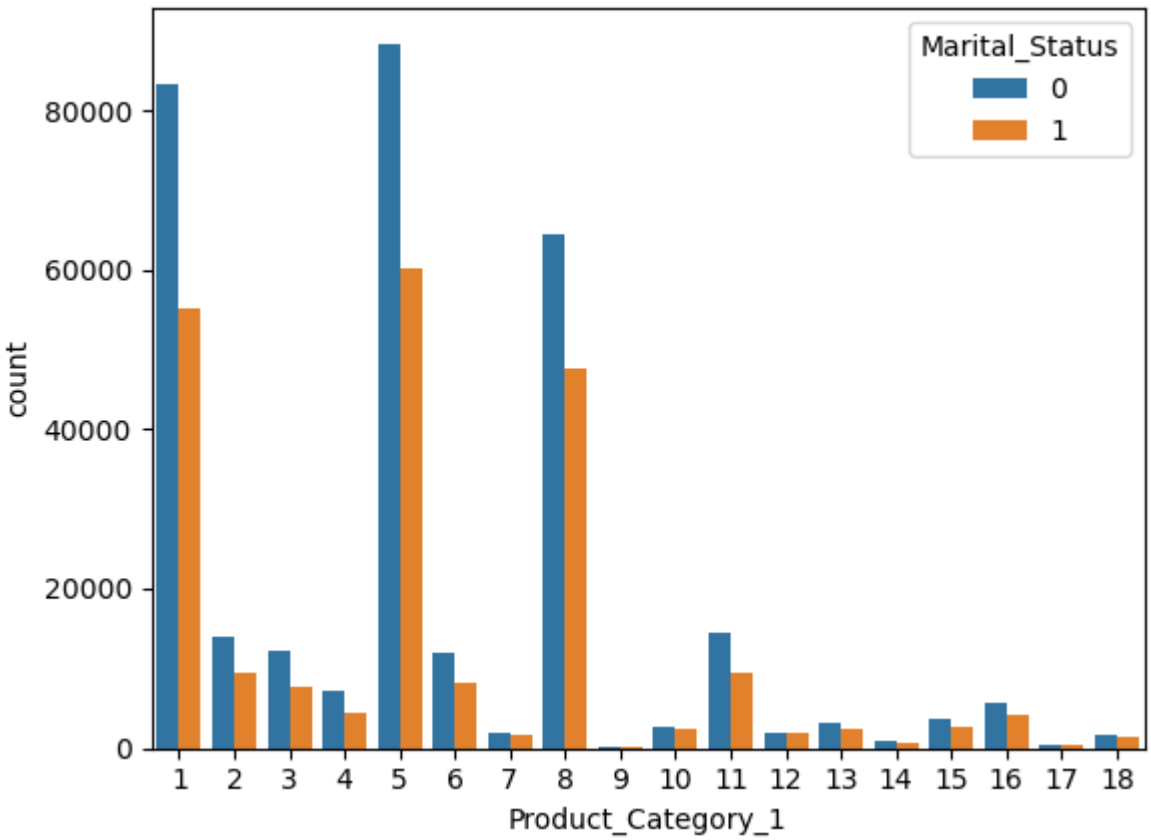
In [34]:
```
sns.countplot(x='Product_Category_1',hue='Gender',data=df)
```

Out[34]:
```
<Axes: xlabel='Product_Category_1', ylabel='count'>
```



In [35]:
```
sns.countplot(x='Product_Category_1',hue='Marital_Status',data=df)
```

Out[35]:
```
<Axes: xlabel='Product_Category_1', ylabel='count'>
```



Combinding Geneder and Marital Satus and prform Analysis

In [40]:
```
l=[]
for i in range(len(df)):
    l.append(df['Gender'][i]+"_"+str(df["Marital_Status"][i]))

df['MaritalGender'] = l
```

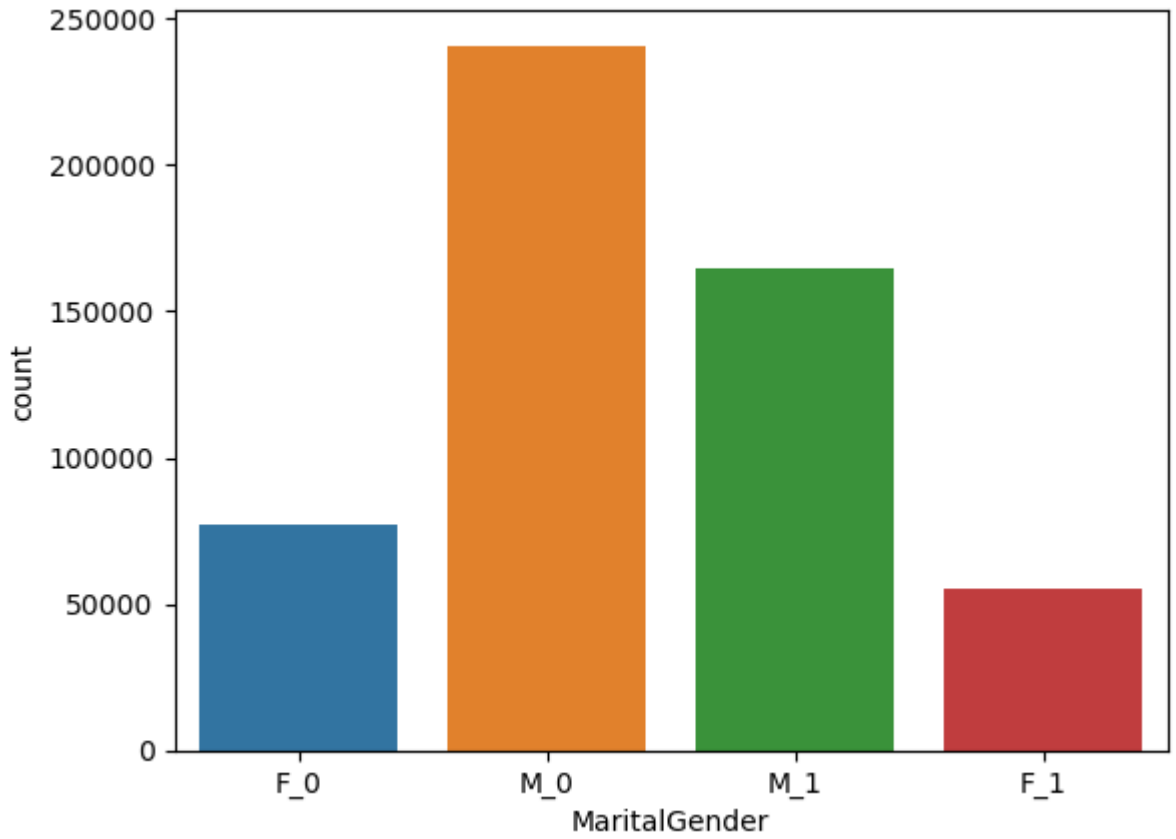
In [41]:
```
df.head()
```

Out[41]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

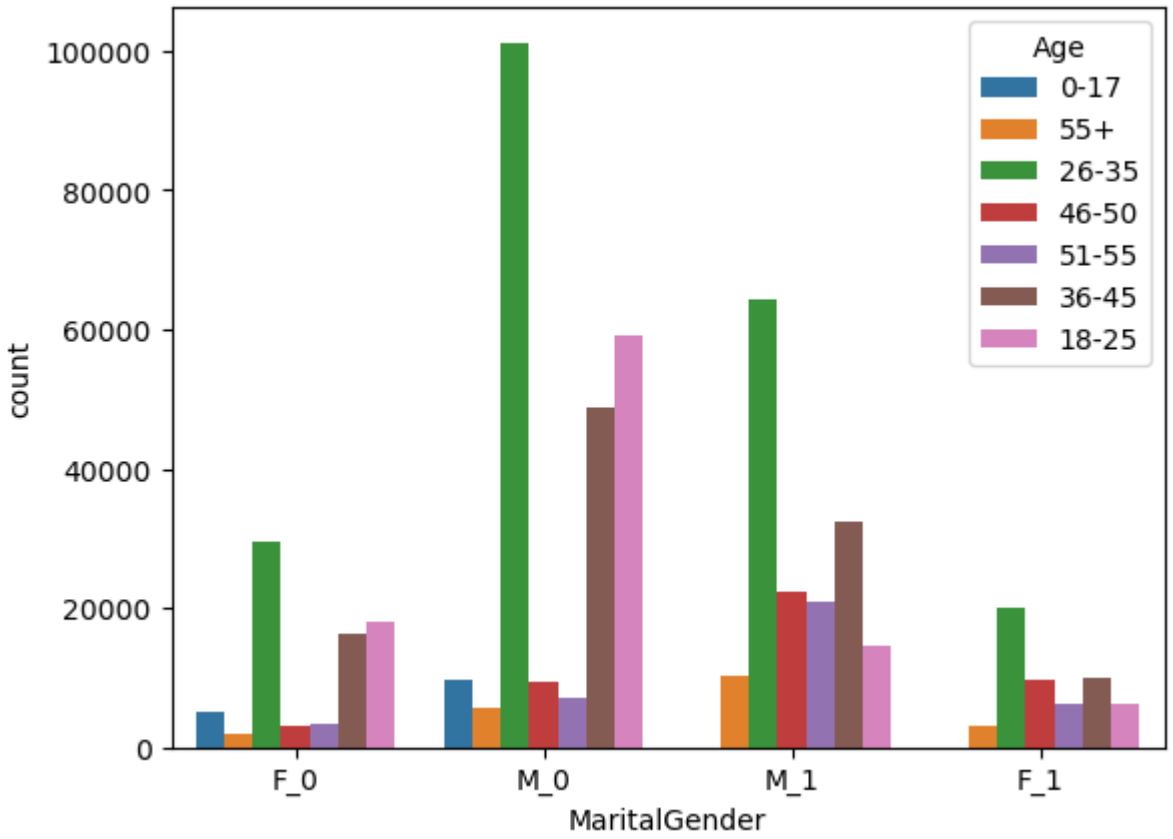
In [42]:
```
sns.countplot(x=df['MaritalGender'])
```

Out[42]:
```
<Axes: xlabel='MaritalGender', ylabel='count'>
```



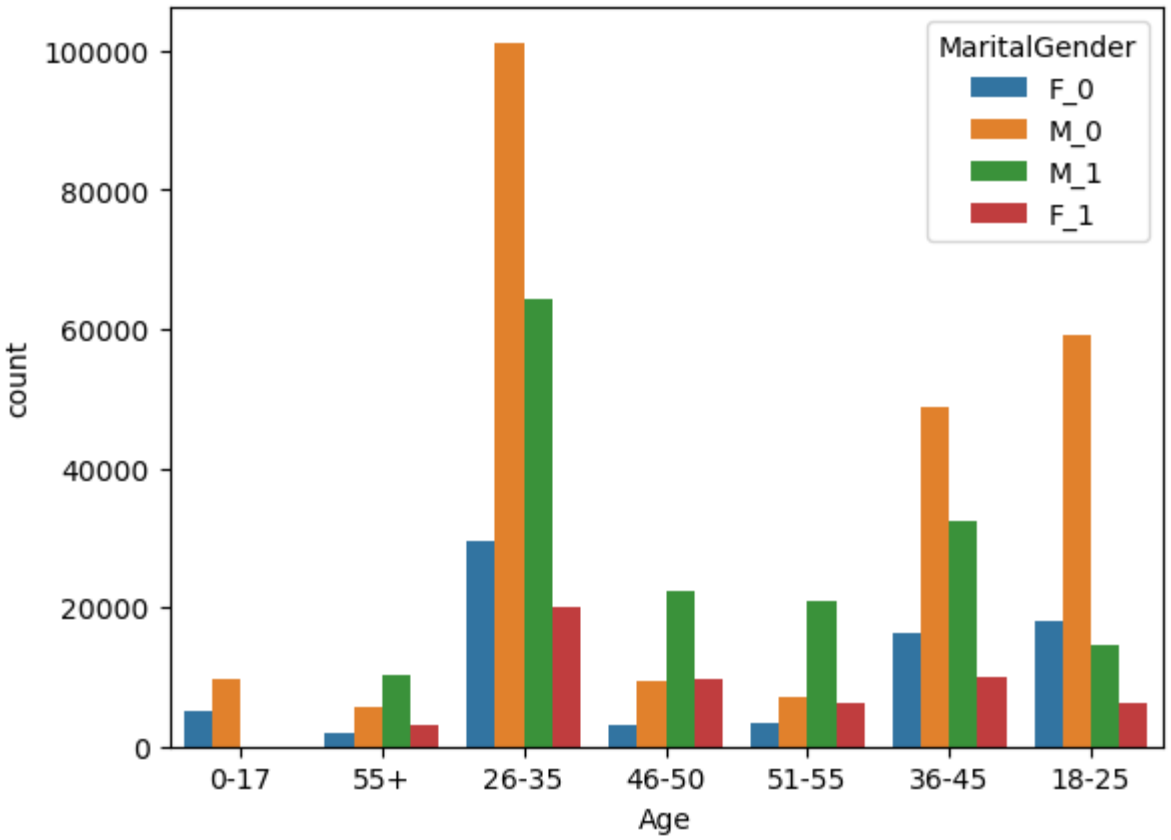
In [43]:
```
sns.countplot(x=df['MaritalGender'],hue=df['Age'])
```

Out[43]:
```
<Axes: xlabel='MaritalGender', ylabel='count'>
```



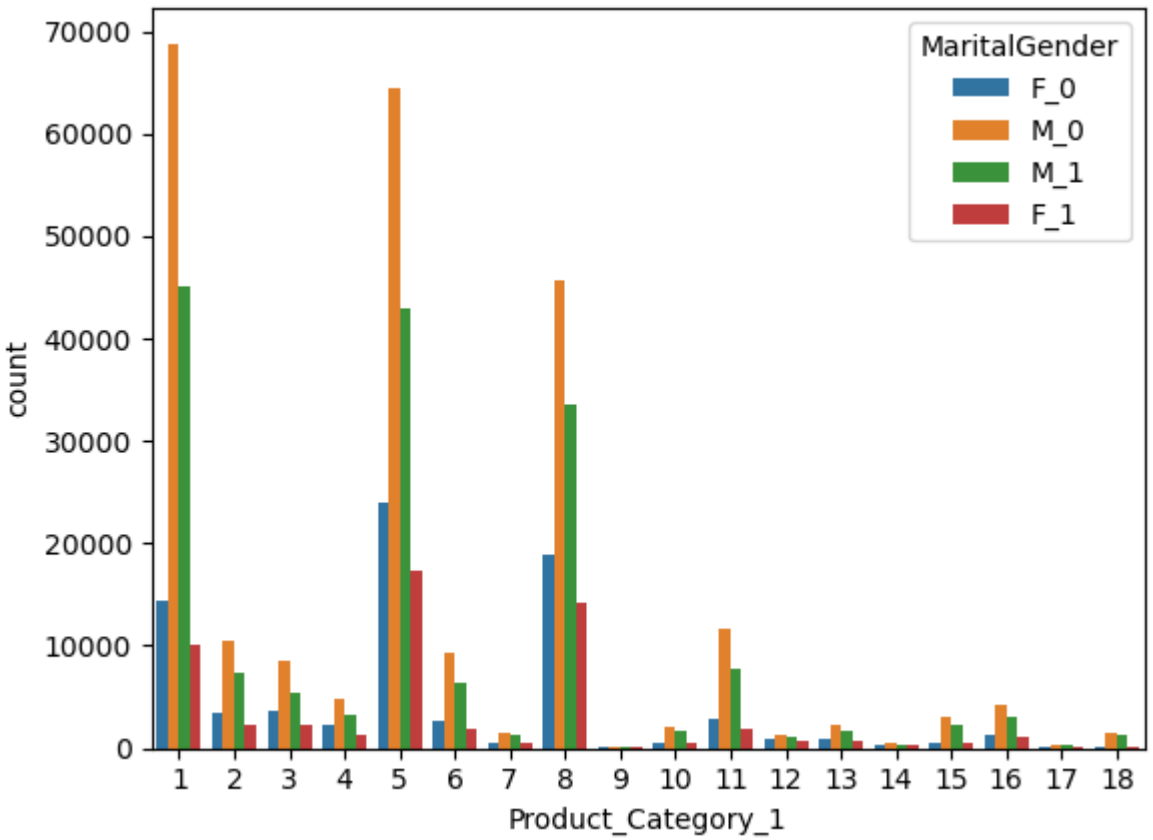
In [44]:
```
sns.countplot(hue=df['MaritalGender'],x=df['Age'])
```

Out[44]:
```
<Axes: xlabel='Age', ylabel='count'>
```



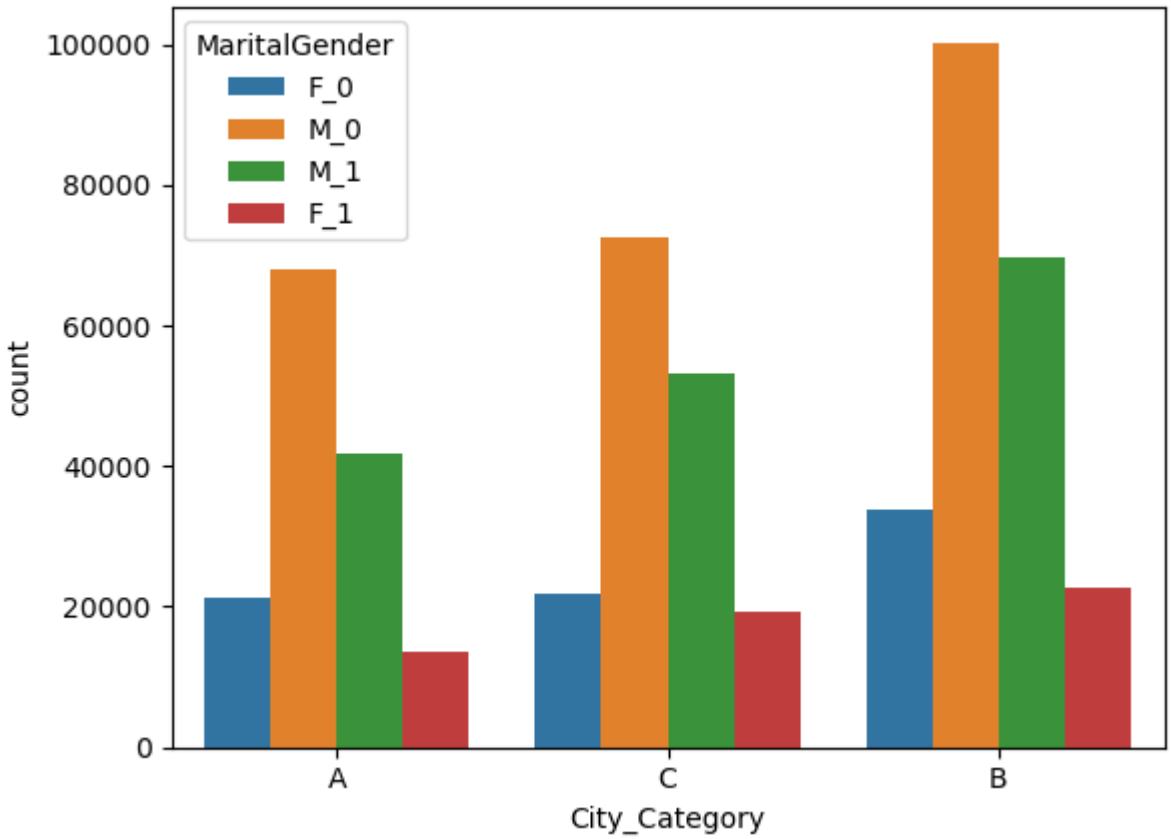
In [45]:
```
sns.countplot(x=df["Product_Category_1"],hue=df['MaritalGender'])
```

Out[45]:
```
<Axes: xlabel='Product_Category_1', ylabel='count'>
```



In [47]:
```
sns.countplot(x=df["City_Category"],hue=df['MaritalGender'])
```

Out[47]:
```
<Axes: xlabel='City_Category', ylabel='count'>
```



In [ ]: