black-friday-sales-basic-eda

June 5, 2024

1 Analyzing the Raw data given wrt Black Friday Sales

2 Installing necessary libraries

[1]:
```
pip install pandas numpy matplotlib seaborn
```

```
Requirement already satisfied: pandas in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (2.2.2)
Requirement already satisfied: numpy in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages
(1.26.4)
Requirement already satisfied: matplotlib in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (3.9.0)
Requirement already satisfied: seaborn in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages
(0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (4.53.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (1.4.5)
```

```
Requirement already satisfied: packaging>=20.0 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (3.1.2)
Requirement already satisfied: six>=1.5 in
c:\users\shubh\appdata\local\programs\python\python312\lib\site-packages (from
python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

3 Reading the csv file

[3]:
```
df = pd.read_csv(r'C:\Users\Shubh\Desktop\PROJECTS\PYTHON\EDA␣
↪_BlackFridaySales\Data\BlackFriday.csv')
```

4 Total number of rows and column

To find total number of rows and column, here we have stored csv file in "df" variable. If we execute this df we will get total rows and total columns

Here total rows : 537577 rows total colums : 12 columns

[4]:
```
df
```

[4]:
```
        User_ID Product_ID Gender   Age  Occupation City_Category  \
0       1000001  P00069042      F  0-17          10             A
1       1000001  P00248942      F  0-17          10             A
2       1000001  P00087842      F  0-17          10             A
3       1000001  P00085442      F  0-17          10             A
4       1000002  P00285442      M   55+          16             C
…           …         …    …    …     …           …
537572  1004737  P00193542      M 36-45          16             C
537573  1004737  P00111142      M 36-45          16             C
537574  1004737  P00345942      M 36-45          16             C
537575  1004737  P00285842      M 36-45          16             C
537576  1004737  P00118242      M 36-45          16             C
```

```
       Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                               2               0                   3
1                               2               0                   1
2                               2               0                  12
3                               2               0                  12
4                              4+               0                   8
…                               …               …                   …
537572                          1               0                   1
537573                          1               0                   1
537574                          1               0                   8
537575                          1               0                   5
537576                          1               0                   5

        Product_Category_2  Product_Category_3  Purchase
0                      NaN                 NaN      8370
1                      6.0                14.0     15200
2                      NaN                 NaN      1422
3                     14.0                 NaN      1057
4                      NaN                 NaN      7969
…                        …                   …         …
537572                 2.0                 NaN     11664
537573                15.0                16.0     19196
537574                15.0                 NaN      8043
537575                 NaN                 NaN      7172
537576                 8.0                 NaN      6875

[537577 rows x 12 columns]
```

5 Analyzing Dataframe (initial level)

[5]:
```python
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
```

Observation 1: Now here we can notice age datatype is object which should be int ;
In Product_Category_2 and Product_Category_3 there seems to be some null values
rest all seems ok.

[6]:
```
df
```

[6]:
```
        User_ID Product_ID Gender   Age  Occupation City_Category  \
0       1000001  P00069042      F  0-17          10             A
1       1000001  P00248942      F  0-17          10             A
2       1000001  P00087842      F  0-17          10             A
3       1000001  P00085442      F  0-17          10             A
4       1000002  P00285442      M   55+          16             C
…           …        …      …    …          …
537572  1004737  P00193542      M 36-45          16             C
537573  1004737  P00111142      M 36-45          16             C
537574  1004737  P00345942      M 36-45          16             C
537575  1004737  P00285842      M 36-45          16             C
537576  1004737  P00118242      M 36-45          16             C

       Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                               2               0                   3
1                               2               0                   1
2                               2               0                  12
3                               2               0                  12
4                              4+               0                   8
…                            …             …                  …
537572                          1               0                   1
537573                          1               0                   1
537574                          1               0                   8
537575                          1               0                   5
537576                          1               0                   5

        Product_Category_2  Product_Category_3  Purchase
0                      NaN                 NaN      8370
1                      6.0                14.0     15200
2                      NaN                 NaN      1422
3                     14.0                 NaN      1057
4                      NaN                 NaN      7969
…                …                 …       …
537572                 2.0                 NaN     11664
537573                15.0                16.0     19196
537574                15.0                 NaN      8043
```

```
537575                  NaN                NaN      7172
537576                  8.0                NaN      6875

[537577 rows x 12 columns]
```

```
Identifying columns with missing values and decide how to handle them, such as imputing missin
```

6 Handling Missing Value

The df.isnull().sum() method in pandas is used to count the number of missing values (NaN) in each column of a DataFrame.

Here's a breakdown of how it works:

1) df.isnull() returns a DataFrame of the same shape as the original DataFrame (df), where each cell contains a boolean value indicating whether the corresponding cell in the original DataFrame is missing (True) or not (False).
2) sum() is then applied to this DataFrame, which sums up the boolean values along each column. Since in Python, True is treated as 1 and False as 0 when summed, this effectively counts the number of missing values in each column.

When we run df.isnull().sum(), we will get a Series object where each index corresponds to a column name in df, and each value indicates the number of missing values in that column.

[7]:
```
df.isnull().sum()
```

[7]:
```
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
```

```
Observation2 : Product_Category_2 and Product_Category_3 have 166986 and 373299 missing values
```

The dropna() method is commonly used in data preprocessing to remove rows or columns with missing data before analysis. The df.dropna() method in pandas is used to remove rows or columns from a DataFrame that contain any missing values (NaN).

#Remove rows with any missing values df_cleaned = df.dropna()

#Remove columns with any missing values df_cleaned = df.dropna(axis=1)

#Remove rows with missing values in specific columns df_cleaned = df.dropna(subset=['column1', 'column2'])

#Remove rows with at least 3 non-missing values df_cleaned = df.dropna(thresh=3)

7 Removing Missing Values in Rows and Column

[8]:
```
df.dropna()
```

[8]:
```
        User_ID Product_ID Gender   Age  Occupation City_Category  \
1       1000001  P00248942      F  0-17          10             A
6       1000004  P00184942      M 46-50           7             B
13      1000005  P00145042      M 26-35          20             A
14      1000006  P00231342      F 51-55           9             A
16      1000006   P0096642      F 51-55           9             A
…          …        …     …     …         …
537549  1004734  P00345842      M 51-55           1             B
537551  1004735  P00313442      M 46-50           3             C
537562  1004736  P00146742      M 18-25          20             A
537571  1004737  P00221442      M 36-45          16             C
537573  1004737  P00111142      M 36-45          16             C

       Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
1                               2               0                   1
6                               2               1                   1
13                              1               1                   1
14                              1               0                   5
16                              1               0                   2
…                            …             …                   …
537549                          1               1                   2
537551                          3               0                   5
537562                          1               1                   1
537571                          1               0                   1
537573                          1               0                   1

        Product_Category_2  Product_Category_3  Purchase
1                      6.0                14.0     15200
6                      8.0                17.0     19215
13                     2.0                 5.0     15665
14                     8.0                14.0      5378
16                     3.0                 4.0     13055
…                    …                 …       …
537549                 8.0                14.0     13082
537551                 6.0                 8.0      6863
537562                13.0                14.0     11508
537571                 2.0                 5.0     11852
537573                15.0                16.0     19196
```

```
[164278 rows x 12 columns]
```

Writing below

del df['Product_Category_2'] del df['Product_Category_3']

or df.dropna(axis=1) will do one and the same thing as here we need to delete the entire column.

[9]:
```
df.dropna(axis=1)
```

[9]:
```
        User_ID Product_ID Gender   Age  Occupation City_Category  \
0       1000001  P00069042      F  0-17          10             A
1       1000001  P00248942      F  0-17          10             A
2       1000001  P00087842      F  0-17          10             A
3       1000001  P00085442      F  0-17          10             A
4       1000002  P00285442      M   55+          16             C
…           …         …    …     …         …
537572  1004737  P00193542      M 36-45          16             C
537573  1004737  P00111142      M 36-45          16             C
537574  1004737  P00345942      M 36-45          16             C
537575  1004737  P00285842      M 36-45          16             C
537576  1004737  P00118242      M 36-45          16             C

       Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                               2               0                   3
1                               2               0                   1
2                               2               0                  12
3                               2               0                  12
4                              4+               0                   8
…                           …              …                  …
537572                          1               0                   1
537573                          1               0                   1
537574                          1               0                   8
537575                          1               0                   5
537576                          1               0                   5

        Purchase
0           8370
1          15200
2           1422
3           1057
4           7969
…          …
537572     11664
537573     19196
537574      8043
537575      7172
```

```
537576      6875

[537577 rows x 10 columns]
```

[10]:
```
df
```

[10]:
```
        User_ID Product_ID Gender    Age  Occupation City_Category  \
0       1000001  P00069042      F   0-17          10             A
1       1000001  P00248942      F   0-17          10             A
2       1000001  P00087842      F   0-17          10             A
3       1000001  P00085442      F   0-17          10             A
4       1000002  P00285442      M    55+          16             C
…           …         …    …      …        …           …
537572  1004737  P00193542      M  36-45          16             C
537573  1004737  P00111142      M  36-45          16             C
537574  1004737  P00345942      M  36-45          16             C
537575  1004737  P00285842      M  36-45          16             C
537576  1004737  P00118242      M  36-45          16             C

       Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                               2               0                   3
1                               2               0                   1
2                               2               0                  12
3                               2               0                  12
4                              4+               0                   8
…                             …             …                 …
537572                          1               0                   1
537573                          1               0                   1
537574                          1               0                   8
537575                          1               0                   5
537576                          1               0                   5

        Product_Category_2  Product_Category_3  Purchase
0                      NaN                 NaN      8370
1                      6.0                14.0     15200
2                      NaN                 NaN      1422
3                     14.0                 NaN      1057
4                      NaN                 NaN      7969
…                    …                 …       …
537572                 2.0                 NaN     11664
537573                15.0                16.0     19196
537574                15.0                 NaN      8043
537575                 NaN                 NaN      7172
537576                 8.0                 NaN      6875

[537577 rows x 12 columns]
```

8 Step 2 : Now Lets Analyze the columns of the data

[11]:
```
df.head()
```

[11]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17          10             A
1  1000001  P00248942      F  0-17          10             A
2  1000001  P00087842      F  0-17          10             A
3  1000001  P00085442      F  0-17          10             A
4  1000002  P00285442      M   55+          16             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                          2               0                   3
1                          2               0                   1
2                          2               0                  12
3                          2               0                  12
4                         4+               0                   8

   Product_Category_2  Product_Category_3  Purchase
0                 NaN                 NaN      8370
1                 6.0                14.0     15200
2                 NaN                 NaN      1422
3                14.0                 NaN      1057
4                 NaN                 NaN      7969
```

[12]:
```
df['User_ID'].nunique()
```

[12]:
```
5891
```

[13]:
```
df['Product_ID'].nunique()
```

[13]:
```
3623
```

[14]:
```
df['Gender'].unique()
```

[14]:
```
array(['F', 'M'], dtype=object)
```

[15]:
```
df['Age'].unique()
```

[15]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

[16]:
```
df['Occupation'].unique()
```

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

[17]:
```python
df['City_Category'].unique()
```

[17]:
```
array(['A', 'C', 'B'], dtype=object)
```

[18]:
```python
df['Stay_In_Current_City_Years'].unique()
```

[18]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

[19]:
```python
df['Marital_Status'].unique()
```

[19]:
```
array([0, 1], dtype=int64)
```

[20]:
```python
df['Product_Category_1'].unique()
```

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

[21]:
```python
df['Purchase'].sum()/len(df['Purchase'])
```

[21]:
```
9333.859852635065
```

[22]:
```python
for column in df.columns:
    print(column, ":" ,df[column].nunique())
```

```
User_ID : 5891
Product_ID : 3623
Gender : 2
Age : 7
Occupation : 21
City_Category : 3
Stay_In_Current_City_Years : 5
Marital_Status : 2
Product_Category_1 : 18
Product_Category_2 : 17
Product_Category_3 : 15
Purchase : 17959
```

9 Step3: Analyzing Gender

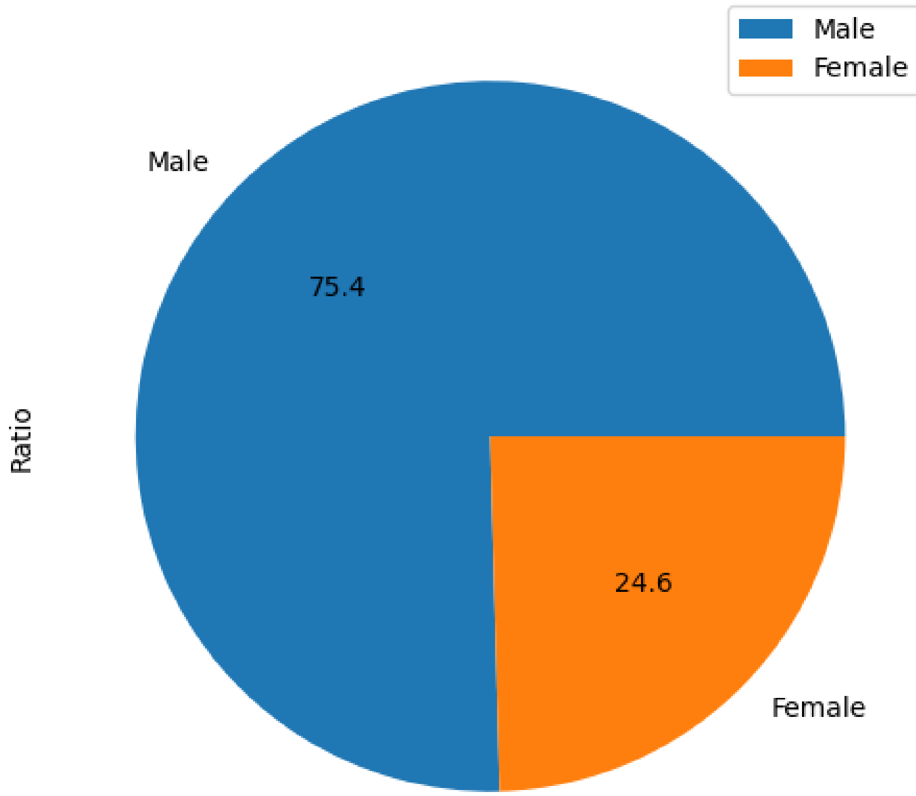
[23]:
```python
# Create a DataFrame to hold unique counts of each column
unique_counts_df = pd.DataFrame({
    'Column': df.columns,
    'Unique Counts': [df[column].nunique() for column in df.columns]
})

# Display the unique counts DataFrame in tabular form
display(unique_counts_df)
```

	Column	Unique Counts
0	User_ID	5891
1	Product_ID	3623
2	Gender	2
3	Age	7
4	Occupation	21
5	City_Category	3
6	Stay_In_Current_City_Years	5
7	Marital_Status	2
8	Product_Category_1	18
9	Product_Category_2	17
10	Product_Category_3	15
11	Purchase	17959

Creating a pie chart to visualize the gender distribution in the DataFrame df.

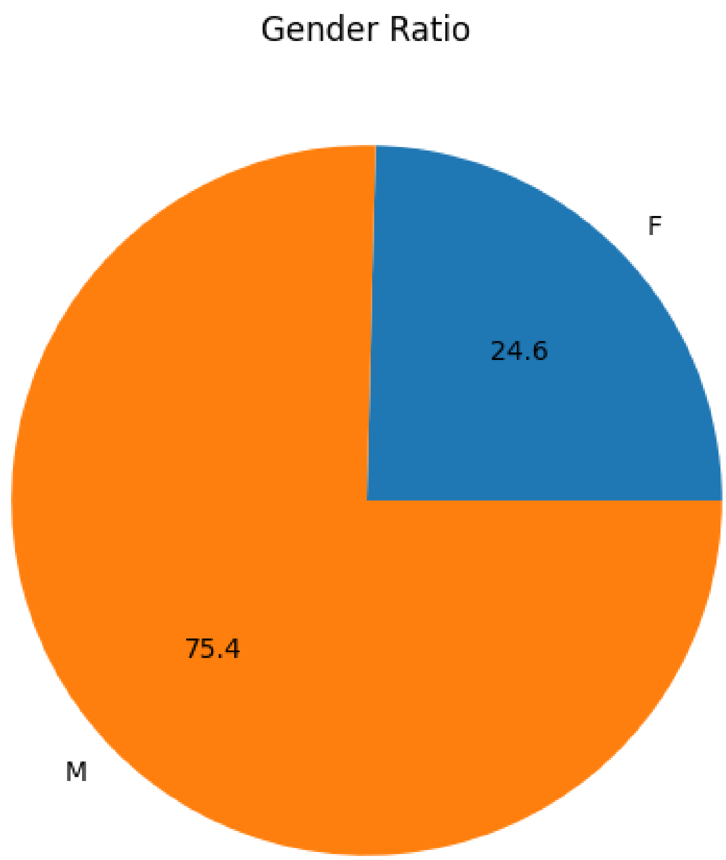
[24]:
```python
data = pd.DataFrame({'Ratio' : [len(df[df['Gender'] == 'M']),␣
↪len(df[df['Gender'] == 'F'])]},
                    index = ['Male', 'Female'])

data.plot.pie(y = 'Ratio', figsize = (6,6), autopct = "%.1f")
```

[24]: <Axes: ylabel='Ratio'>



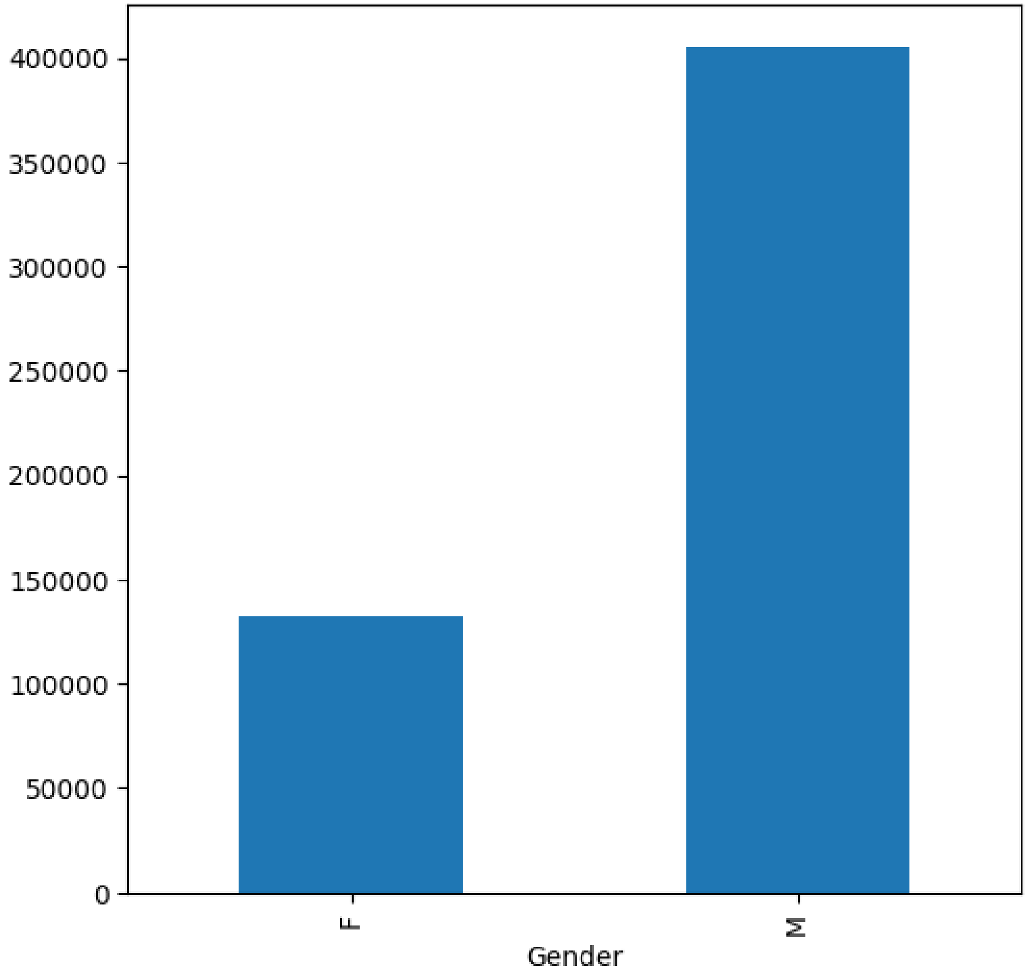

```
[25]: df.groupby('Gender').size().plot(kind = 'pie',
                                        autopct = "%.1f",
                                        title = 'Gender Ratio',
                                        figsize = (6,6))
```

```
[25]: <Axes: title={'center': 'Gender Ratio'}>
```



[26]:
```
df.groupby('Gender').size().plot(kind = 'bar',
                                 figsize = (6,6))
```

[26]: <Axes: xlabel='Gender'>



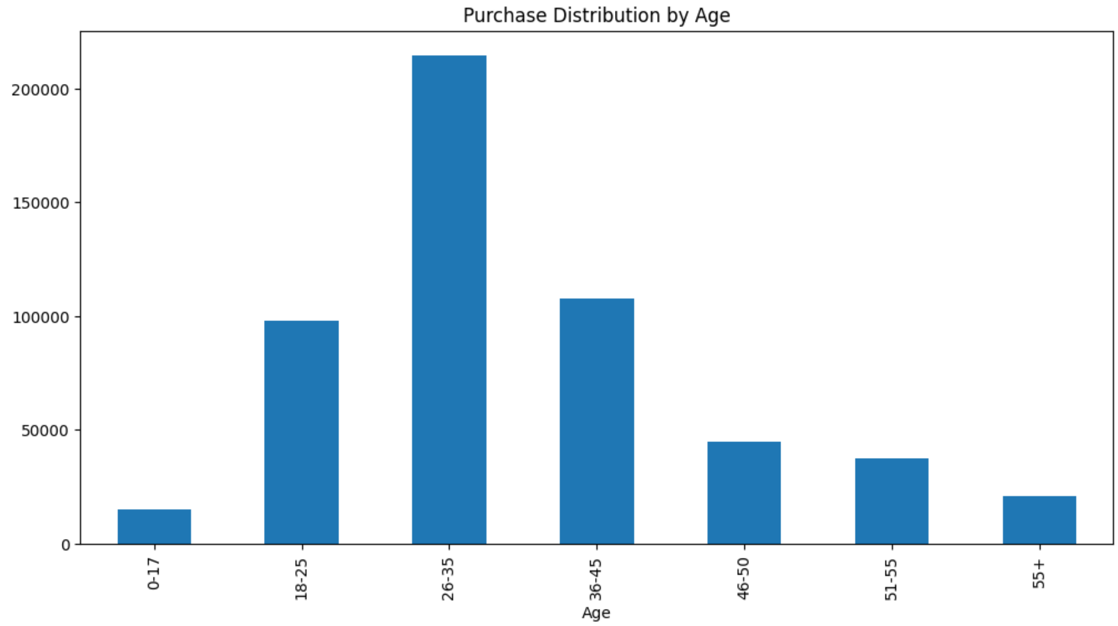
[27]:
```
df.groupby('Gender').size()
```

[27]:
```
Gender
F    132197
M    405380
dtype: int64
```

10 Analyzing Age and Marital Status

[28]:
```
df.groupby('Age').size().plot(kind = 'bar', figsize = (12, 6), title =␣
↪'Purchase Distribution by Age')
```

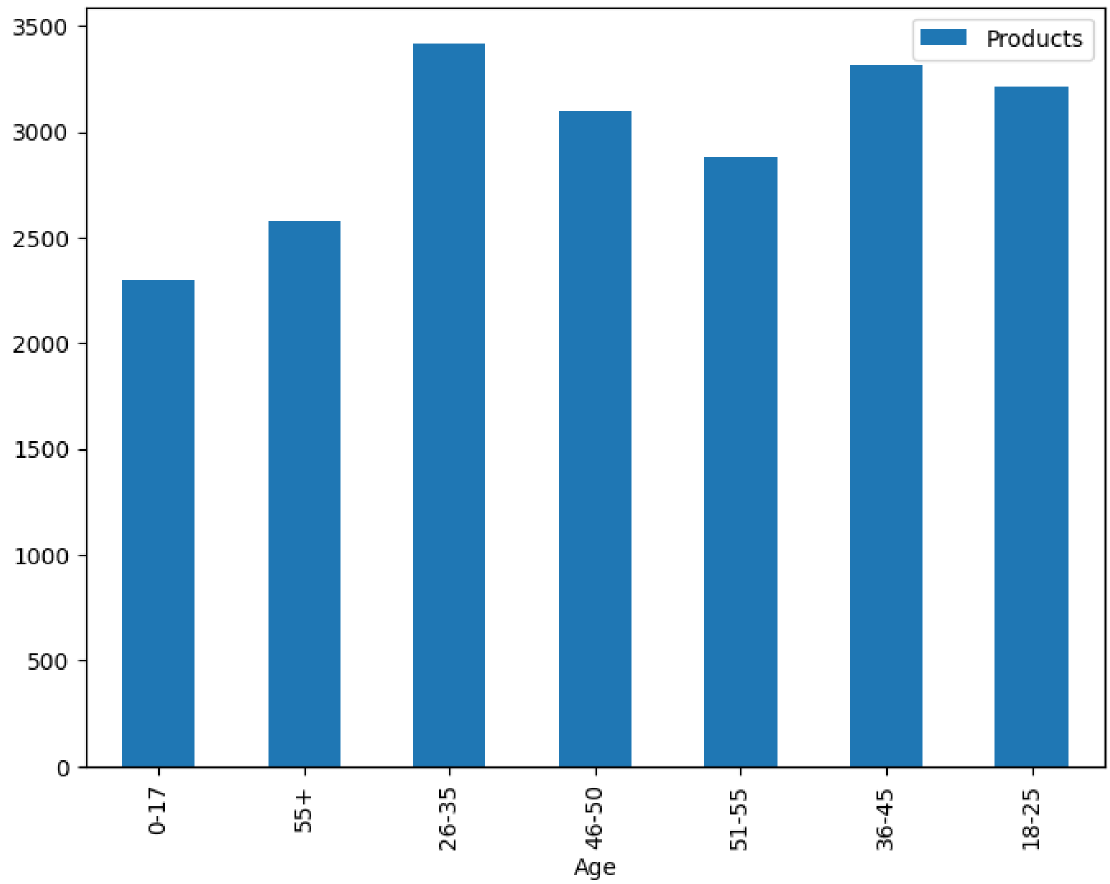
[28]:
```
<Axes: title={'center': 'Purchase Distribution by Age'}, xlabel='Age'>
```



[29]:
```
lst = []
for i in df['Age'].unique():
    lst.append([i, df[df['Age'] == i]['Product_ID'].nunique()])

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

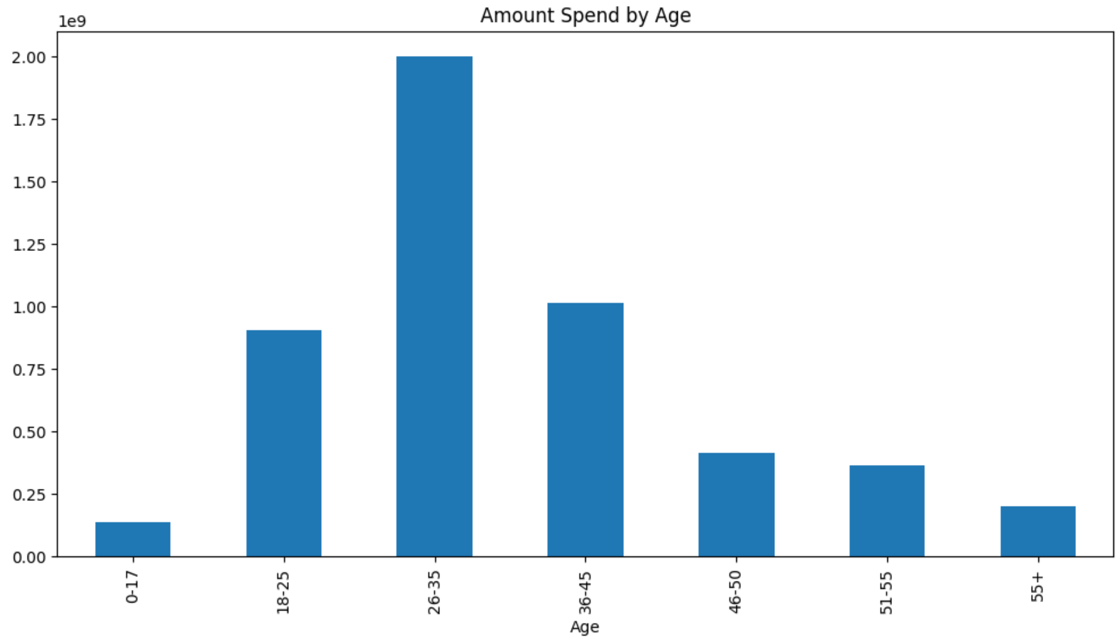
[30]:
```
data.plot.bar(x = 'Age', figsize = (8,6))
```

[30]: <Axes: xlabel='Age'>



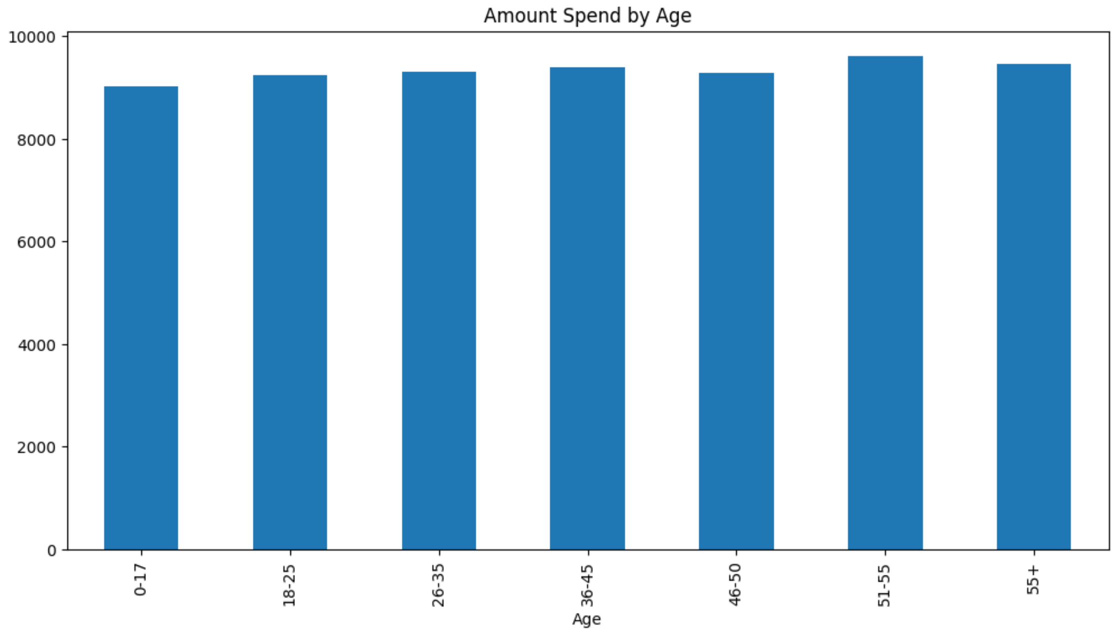
[31]:
```
df.groupby('Age').sum()['Purchase'].plot(kind = 'bar', figsize = (12, 6), title␣
↪= 'Amount Spend by Age')
```

[31]:
```
<Axes: title={'center': 'Amount Spend by Age'}, xlabel='Age'>
```

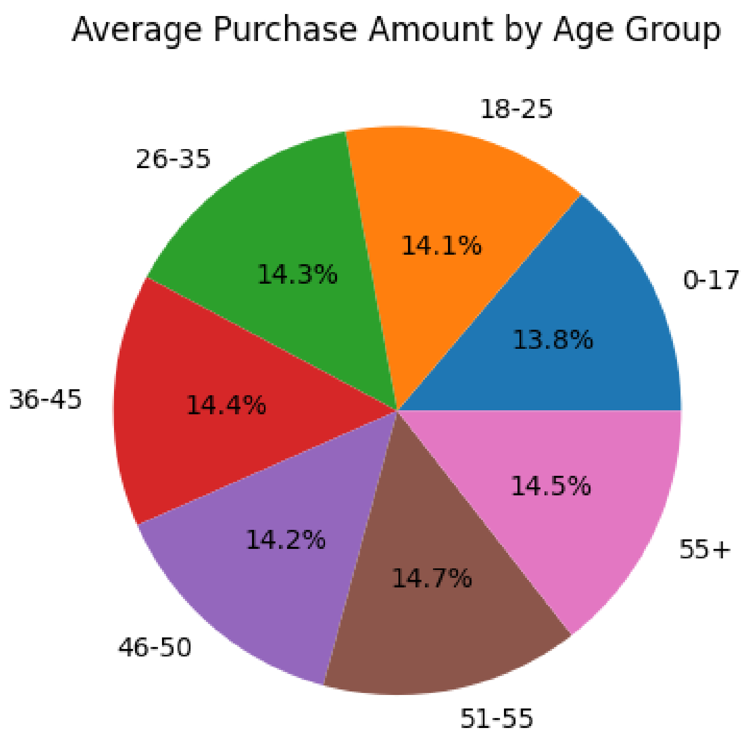


[37]:
```
##df.groupby('Age').mean()['Purchase'].plot(kind = 'bar', figsize = (12, 6),␣
↪title = 'Amount Spend by Age')

# Convert 'Purchase' column to numeric, coercing errors to NaN
df['Purchase'] = pd.to_numeric(df['Purchase'], errors='coerce')

# Drop rows with NaN values in 'Purchase' column
df.dropna(subset=['Purchase'], inplace=True)

# Now, try plotting again
df.groupby('Age')['Purchase'].mean().plot(kind='bar', figsize=(12, 6),␣
↪title='Amount Spend by Age')
```

[37]:
```
<Axes: title={'center': 'Amount Spend by Age'}, xlabel='Age'>
```

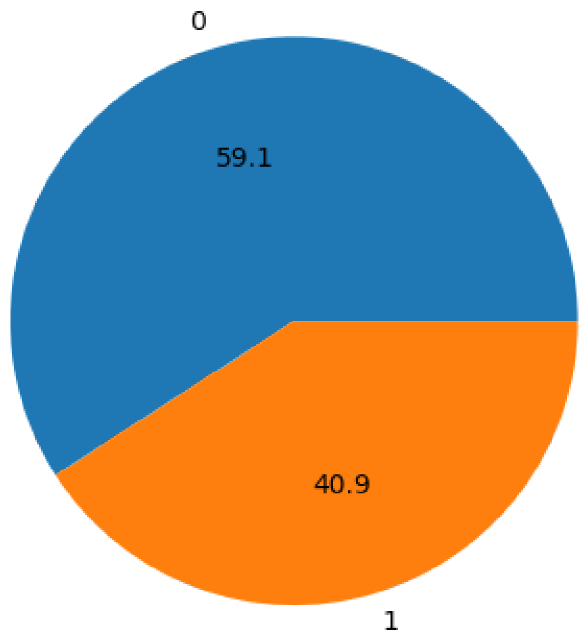


[39]:
```
#df.groupby('Age').mean()['Purchase'].plot(kind = 'pie', autopct = '%0.1f')


# Group by 'Age' and calculate the mean purchase amount
mean_purchase_by_age = df.groupby('Age')['Purchase'].mean()

# Plot the pie chart
mean_purchase_by_age.plot(kind='pie', autopct='%0.1f%%')
plt.ylabel('')  # Remove the y-label
plt.title('Average Purchase Amount by Age Group')
plt.show()
```



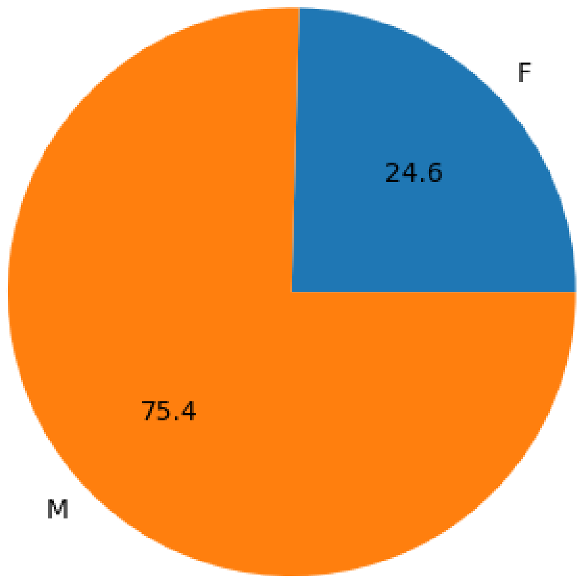
[40]:
```
df.groupby('Marital_Status').size().plot(kind = 'pie', autopct = '%0.1f')
```

[40]: <Axes: >



[41]:
```
df.groupby('Gender').size().plot(kind = 'pie', autopct = '%0.1f')
```

[41]: <Axes: >

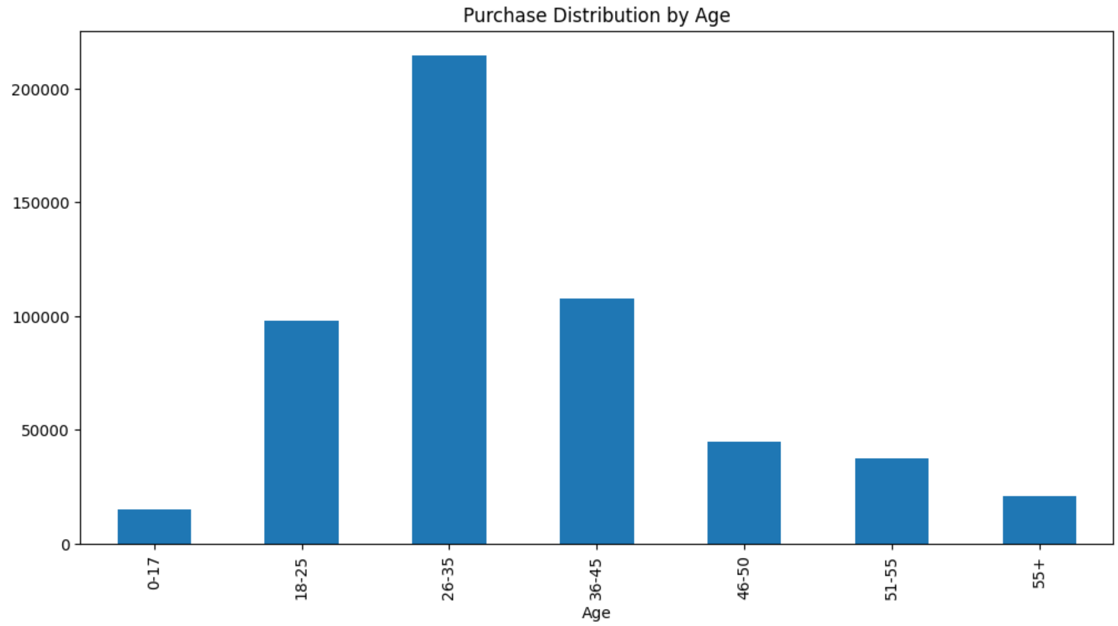


11 MultiColumn Analysis

[42]:
```
import seaborn as sns
```

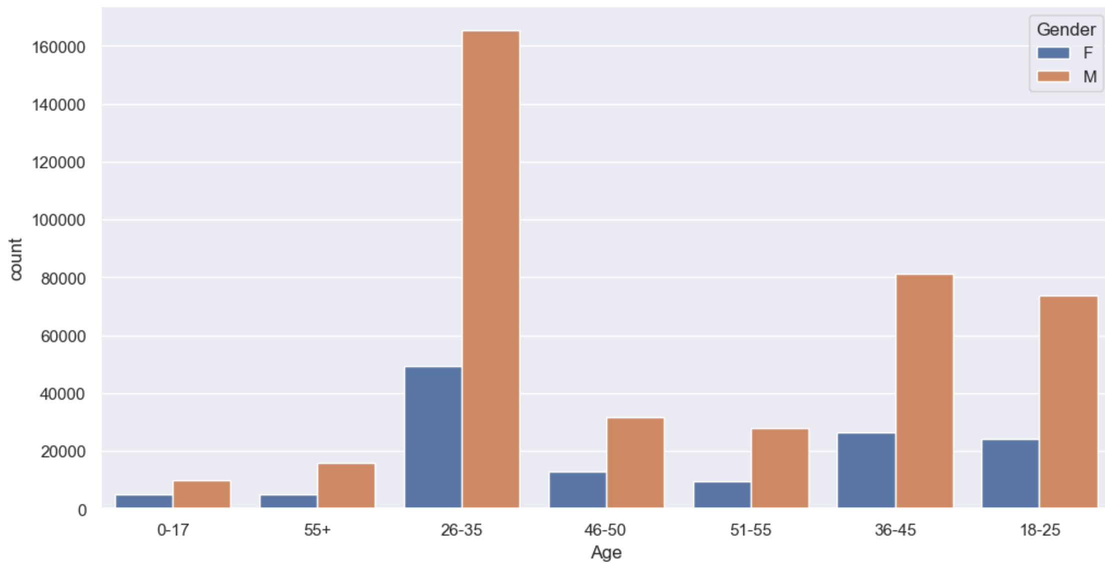
[43]:
```
df.groupby('Age').size().plot(kind = 'bar', figsize = (12, 6), title =␣
↪'Purchase Distribution by Age')
```

[43]:
```
<Axes: title={'center': 'Purchase Distribution by Age'}, xlabel='Age'>
```



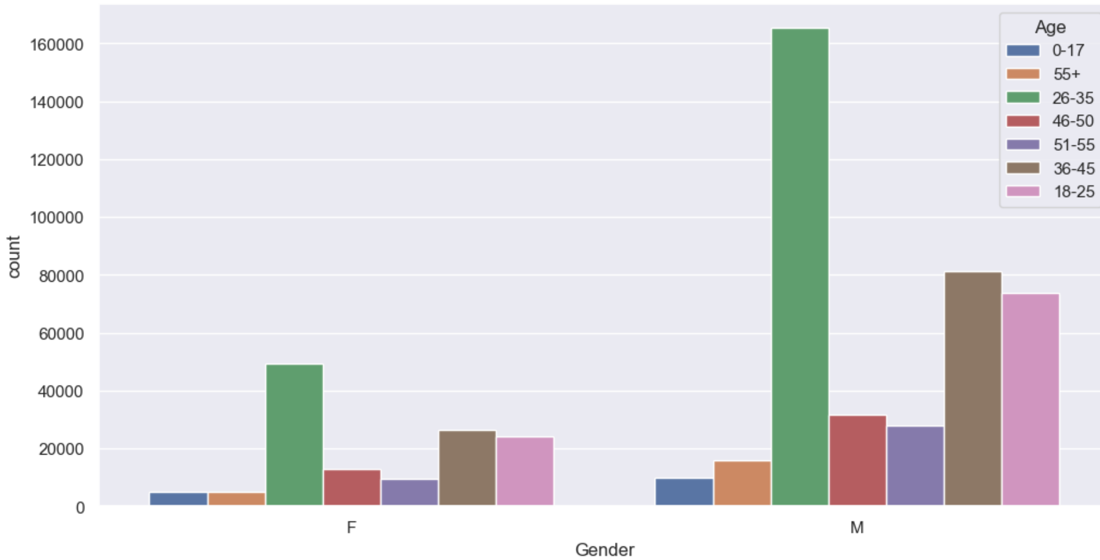
[44]:
```
sns.set(rc = {'figure.figsize' : (12,6)})
sns.countplot(x = "Age", hue = 'Gender', data = df)
```

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



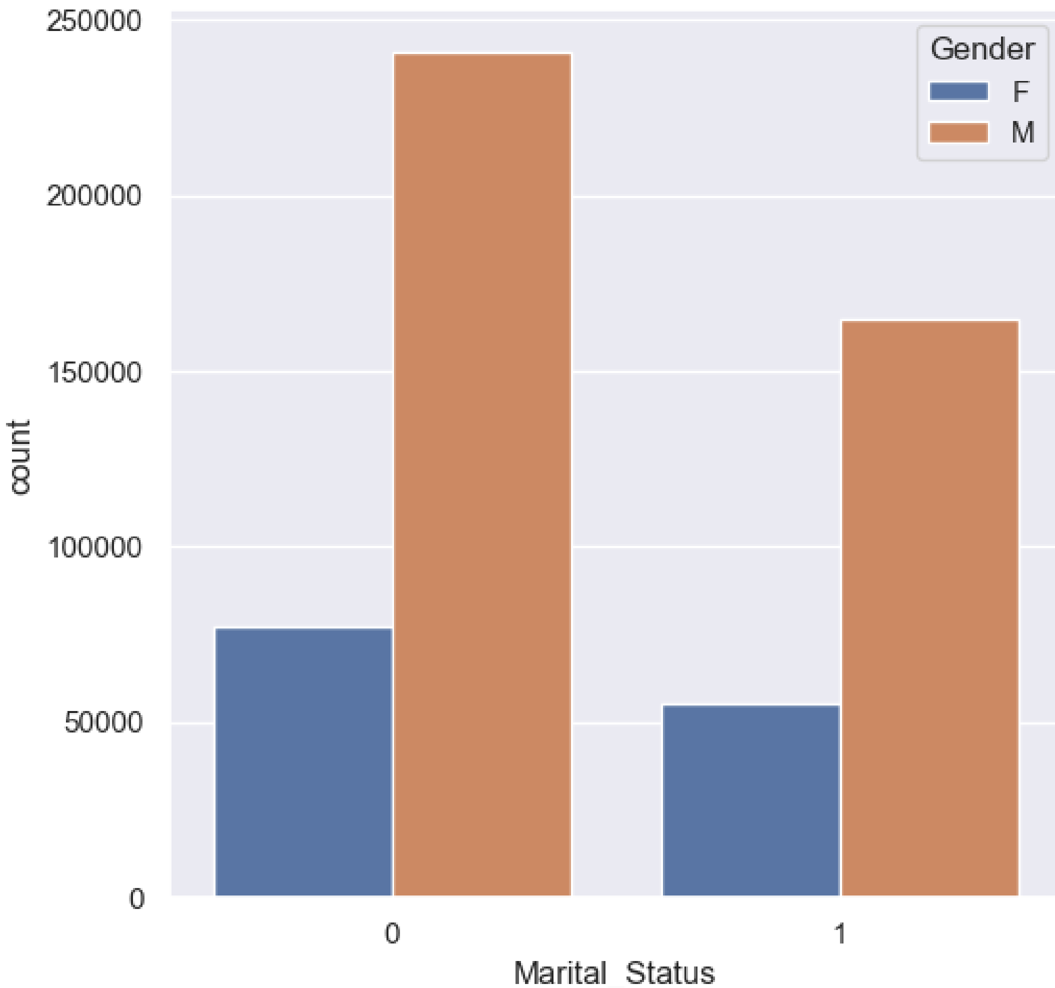
[45]:
```
sns.set(rc = {'figure.figsize' : (12,6)})
sns.countplot(x = "Gender", hue = 'Age', data = df)
```

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



[46]:
```
sns.set(rc = {'figure.figsize' : (6,6)})
sns.countplot(x = "Marital_Status", hue = 'Gender', data = df)
```

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

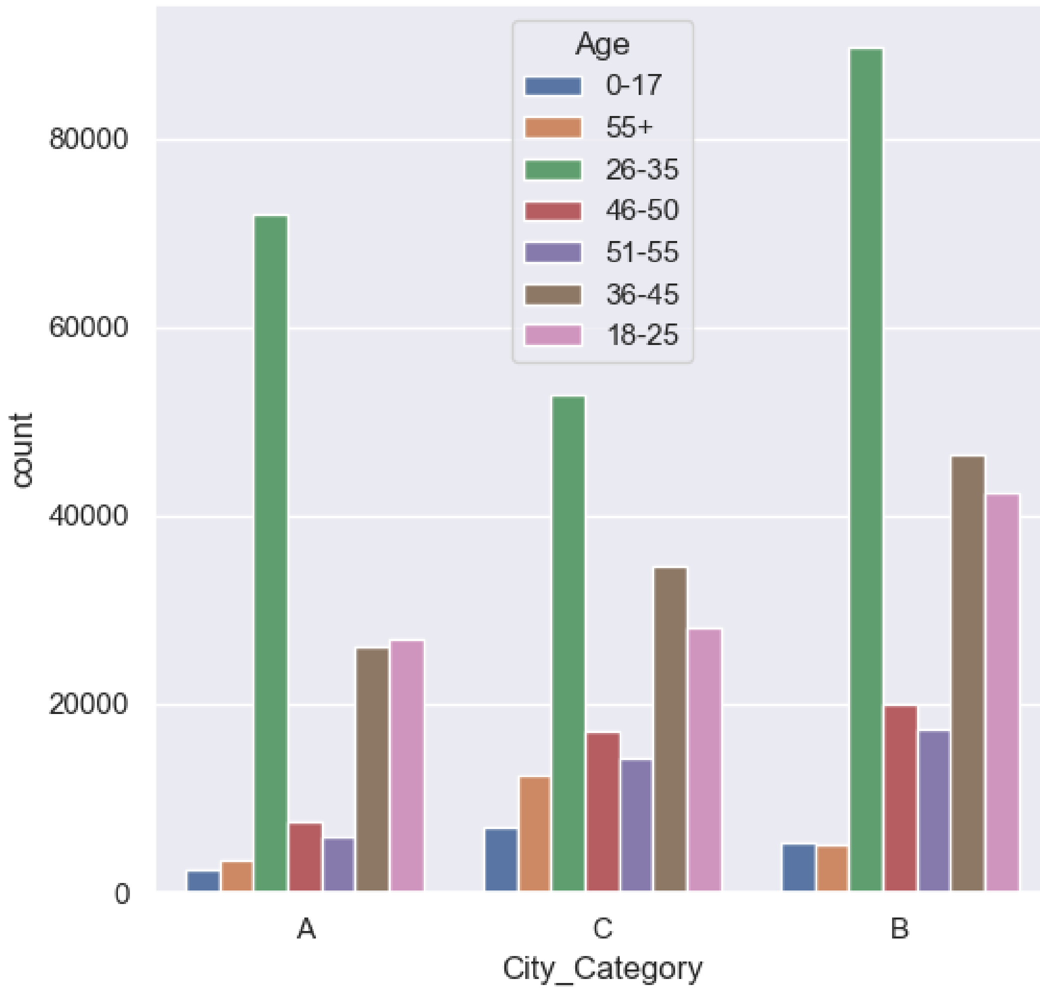



```
[47]: sns.countplot(x = df['City_Category'])
```

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

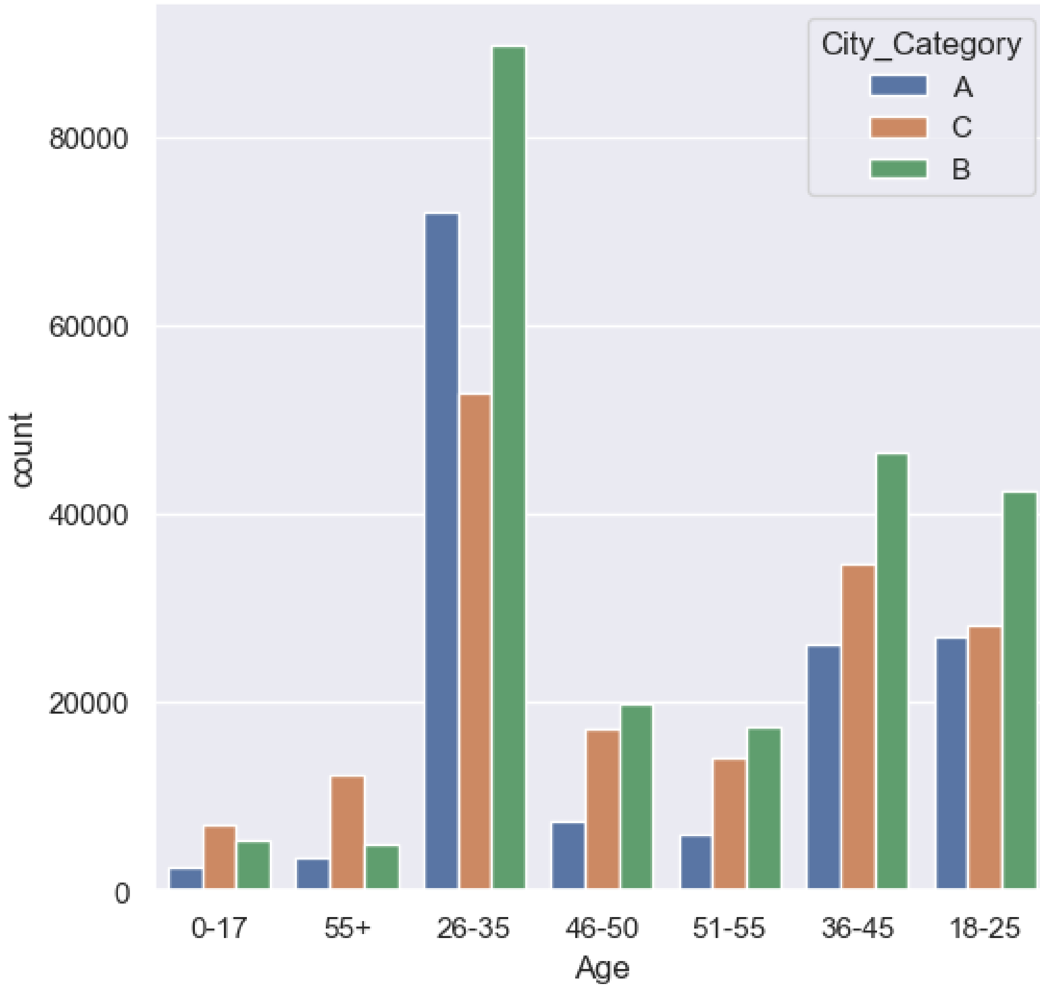
[48]:
```
sns.countplot(x = 'City_Category', hue = 'Age', data = df)
```

[48]: <Axes: xlabel='City_Category', ylabel='count'>



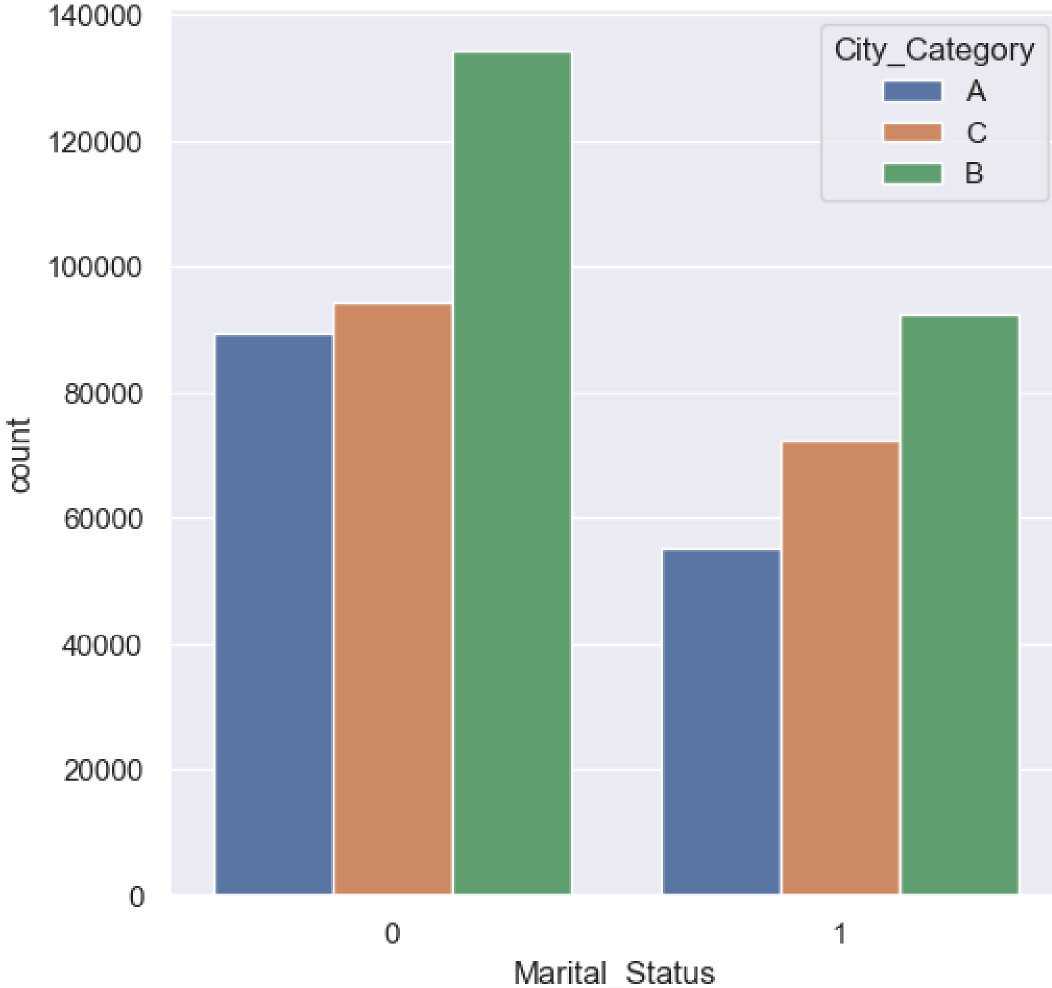
[49]:
```python
sns.countplot(x = 'Age', hue = 'City_Category', data = df)
```

[49]: <Axes: xlabel='Age', ylabel='count'>



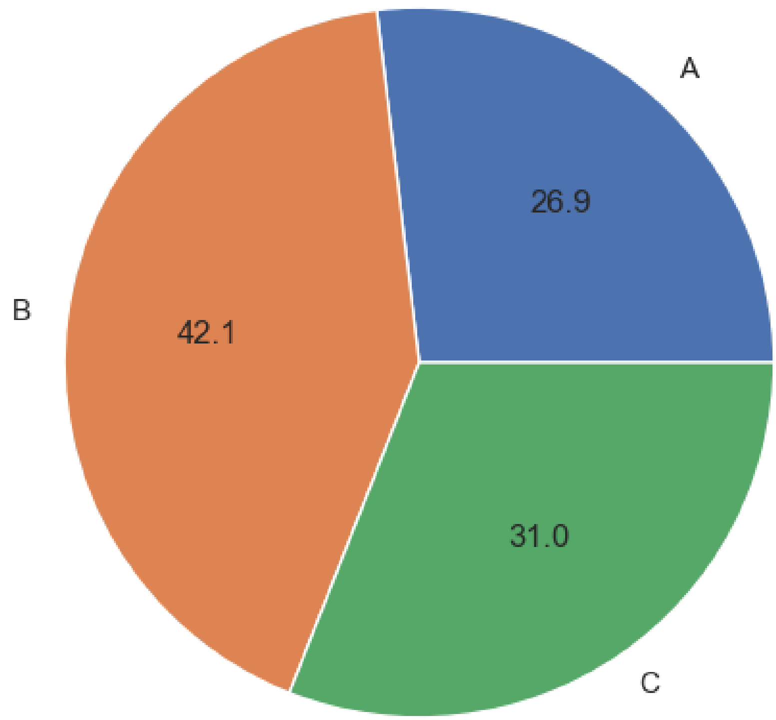
[50]:
```
sns.countplot(x = 'Marital_Status', hue = 'City_Category', data = df)
```

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



[51]:
```
df.groupby('City_Category').size().plot(kind = 'pie', autopct = '%0.1f')
```

[51]: <Axes: >

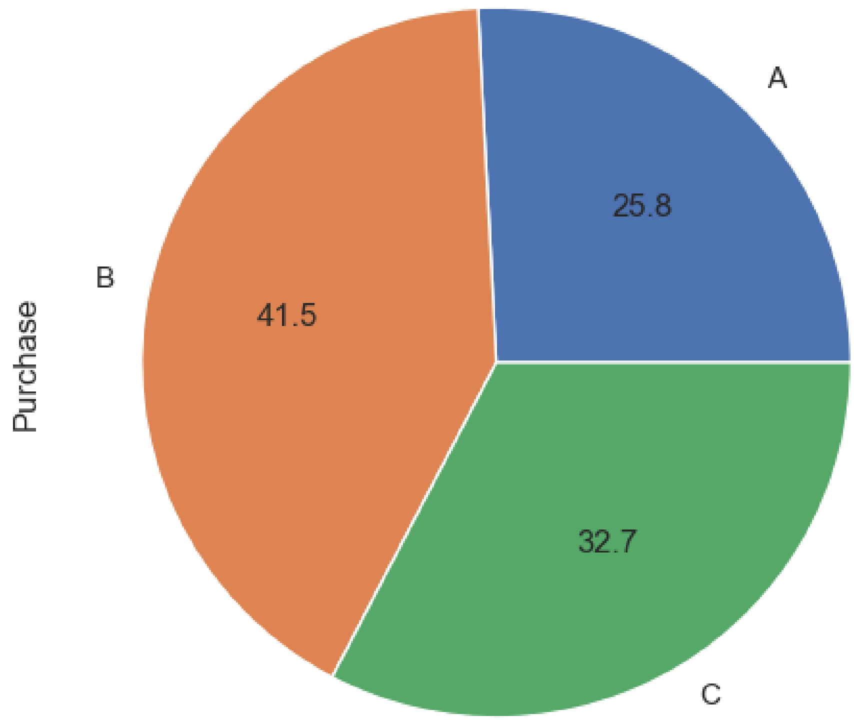



```
[52]: sns.countplot(x = 'City_Category', hue = 'Gender', data = df)
```

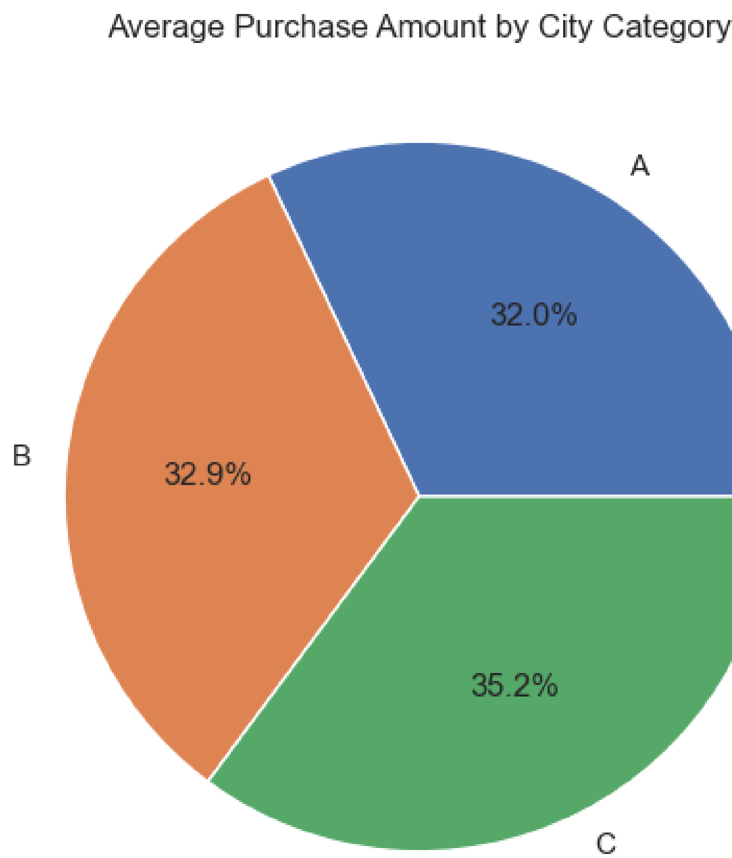
```
[52]: <Axes: xlabel='City_Category', ylabel='count'>
```

[53]:
```
df.groupby('City_Category').sum()['Purchase'].plot(kind = 'pie', autopct = "%0.
↪1f")
```

[53]: <Axes: ylabel='Purchase'>



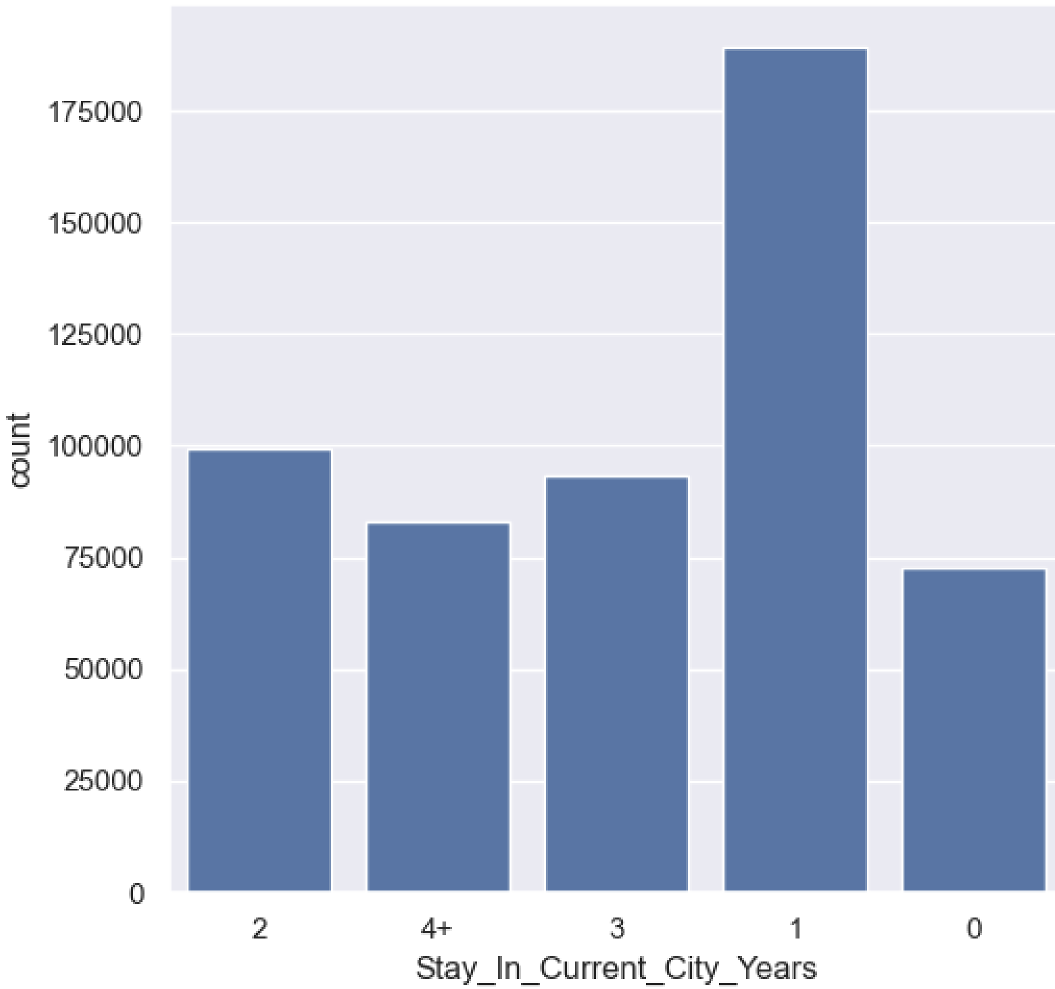
[55]:
```
# Group by 'City_Category' and calculate the mean purchase amount
mean_purchase_by_city = df.groupby('City_Category')['Purchase'].mean()

# Plot the pie chart
mean_purchase_by_city.plot(kind='pie', autopct='%0.1f%%')
plt.ylabel('')  # Remove the y-label
plt.title('Average Purchase Amount by City Category')
plt.show()
```



12 Occupation and Products Analysis

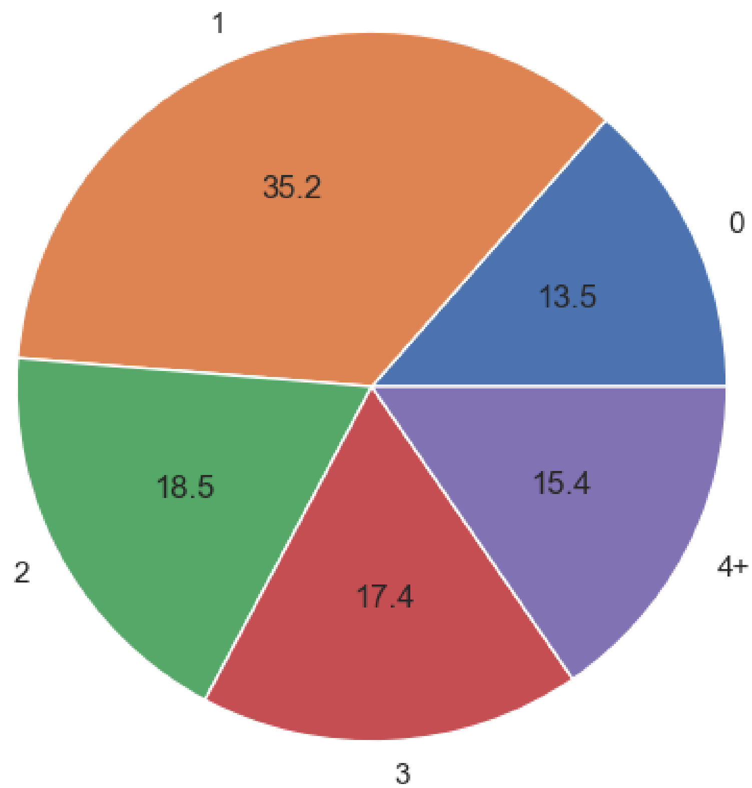
[56]:
```
sns.countplot(x = df['Stay_In_Current_City_Years'])
```

[56]: <Axes: xlabel='Stay_In_Current_City_Years', ylabel='count'>



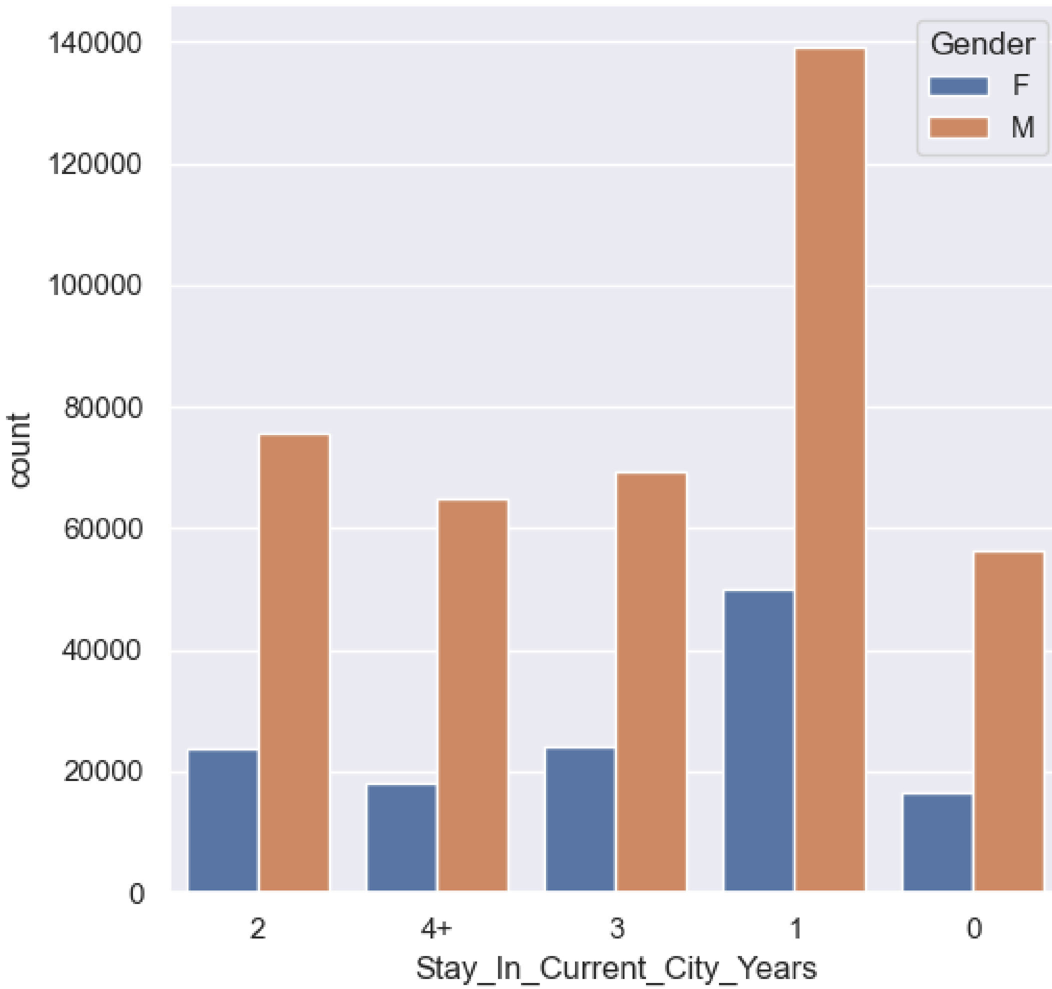
[57]:
```
df.groupby('Stay_In_Current_City_Years').size().plot(kind = 'pie', autopct = "%.
↪1f")
```

[57]: <Axes: >



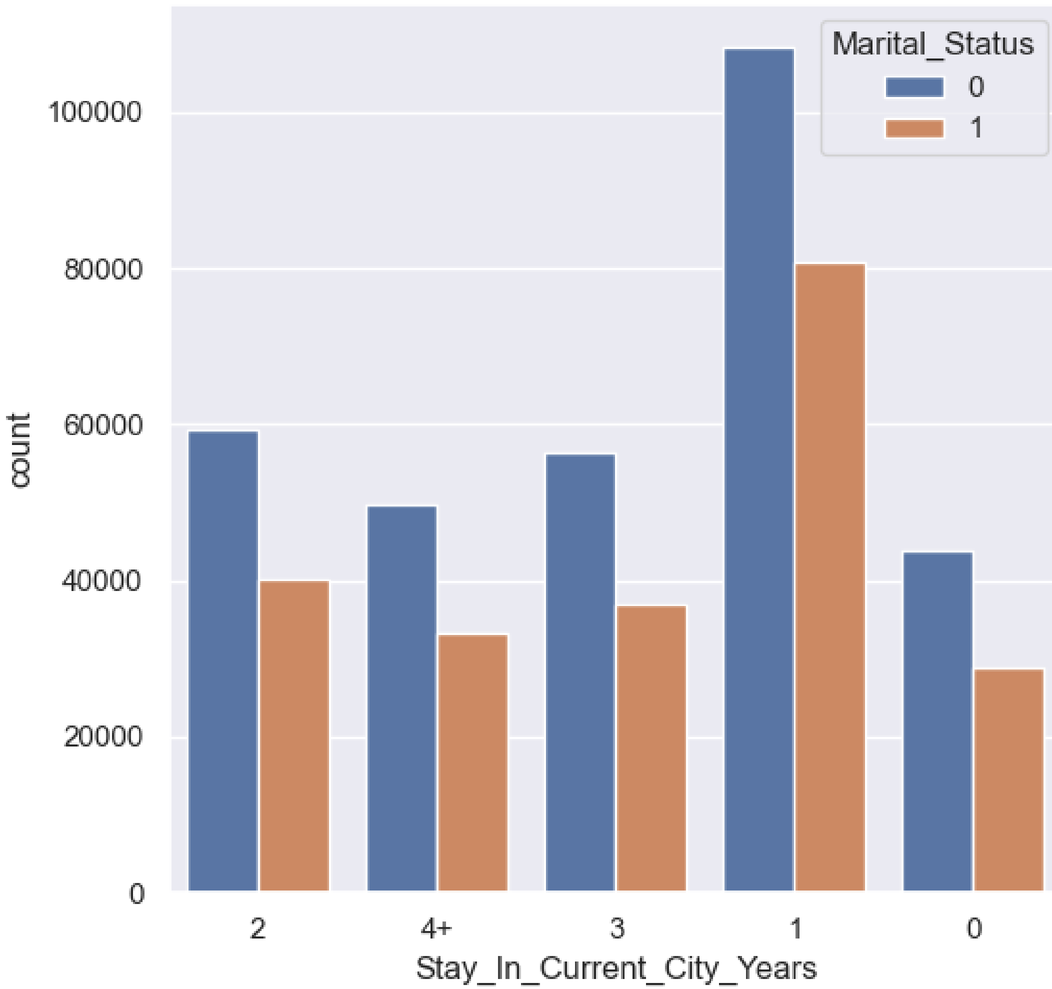
[58]:
```
sns.countplot(x = 'Stay_In_Current_City_Years', hue = 'Gender', data = df)
```

[58]: <Axes: xlabel='Stay_In_Current_City_Years', ylabel='count'>



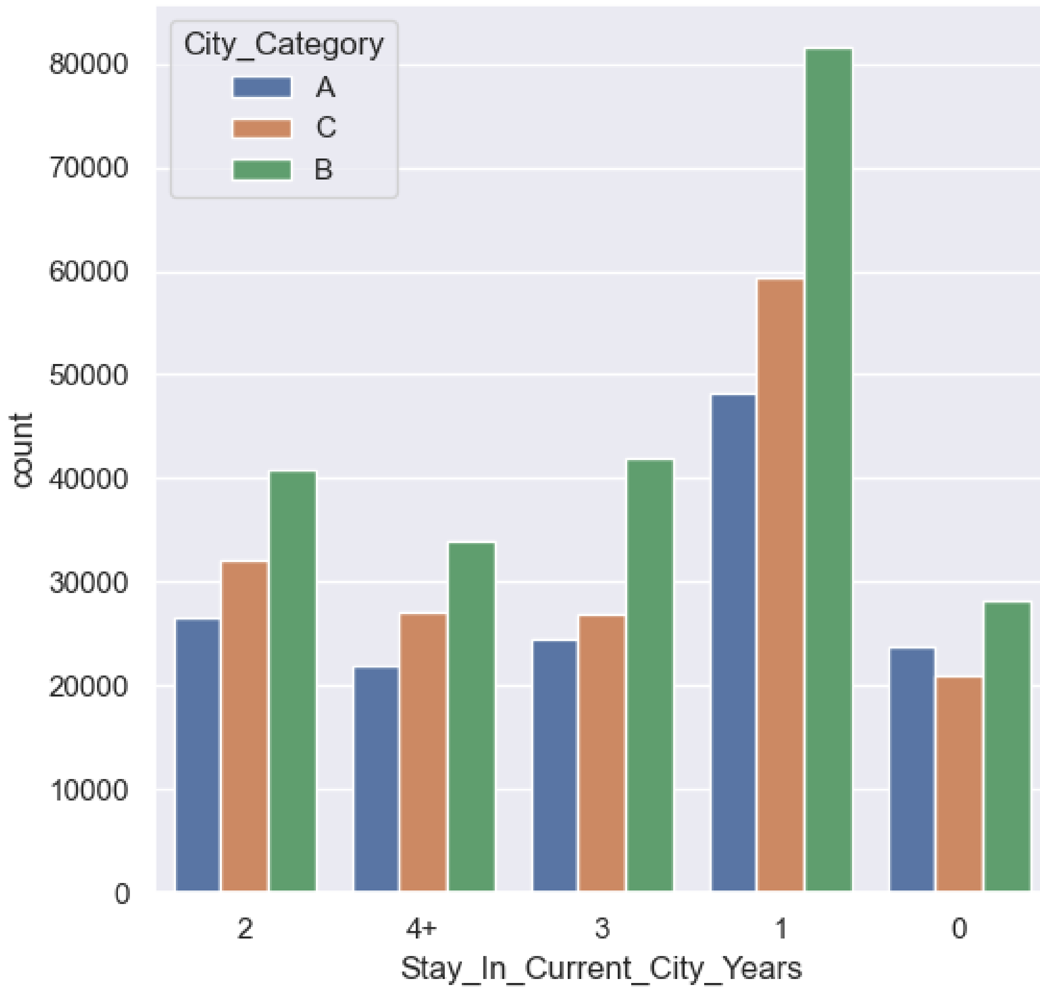

```
[59]: sns.countplot(x = 'Stay_In_Current_City_Years', hue = 'Marital_Status', data =␣
      ↪df)
```

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



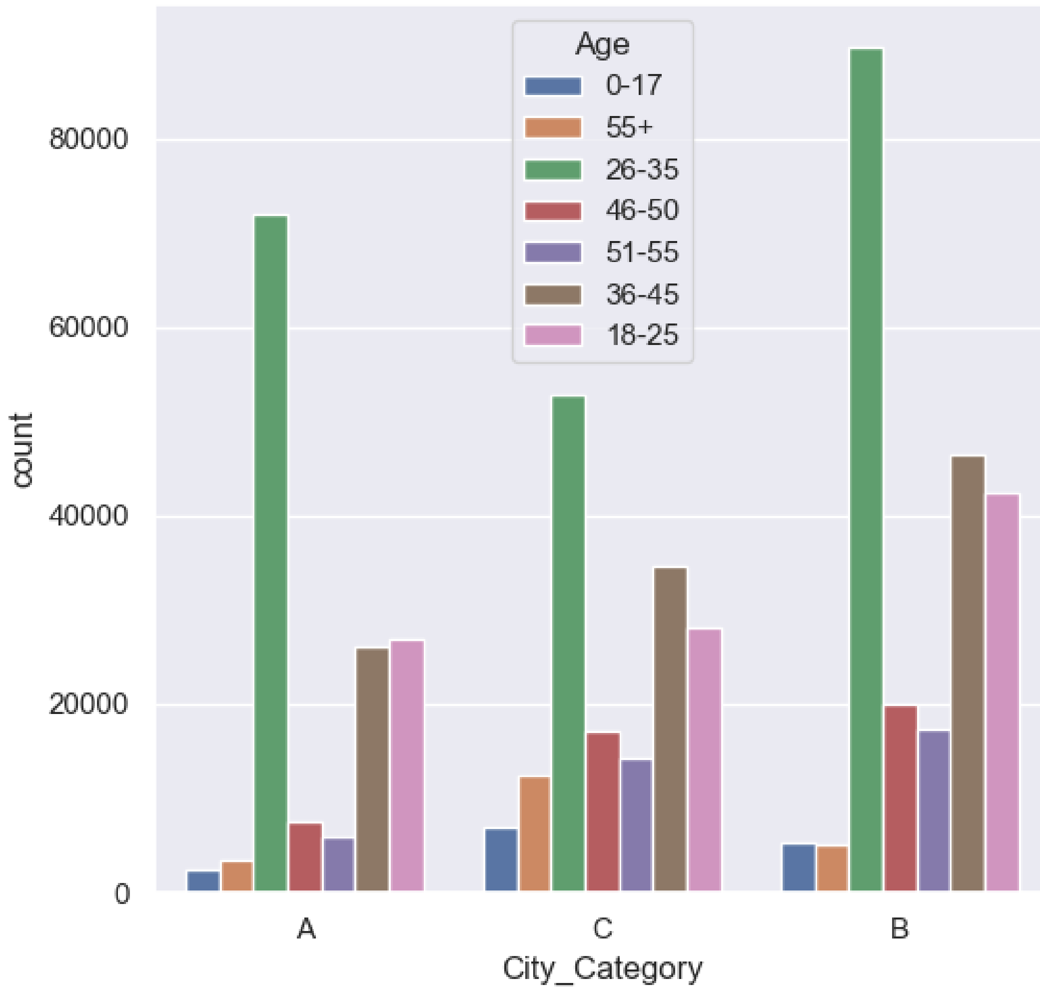
[60]:
```
sns.countplot(x = 'Stay_In_Current_City_Years', hue = 'City_Category', data =␣
↪df)
```

[60]: <Axes: xlabel='Stay_In_Current_City_Years', ylabel='count'>



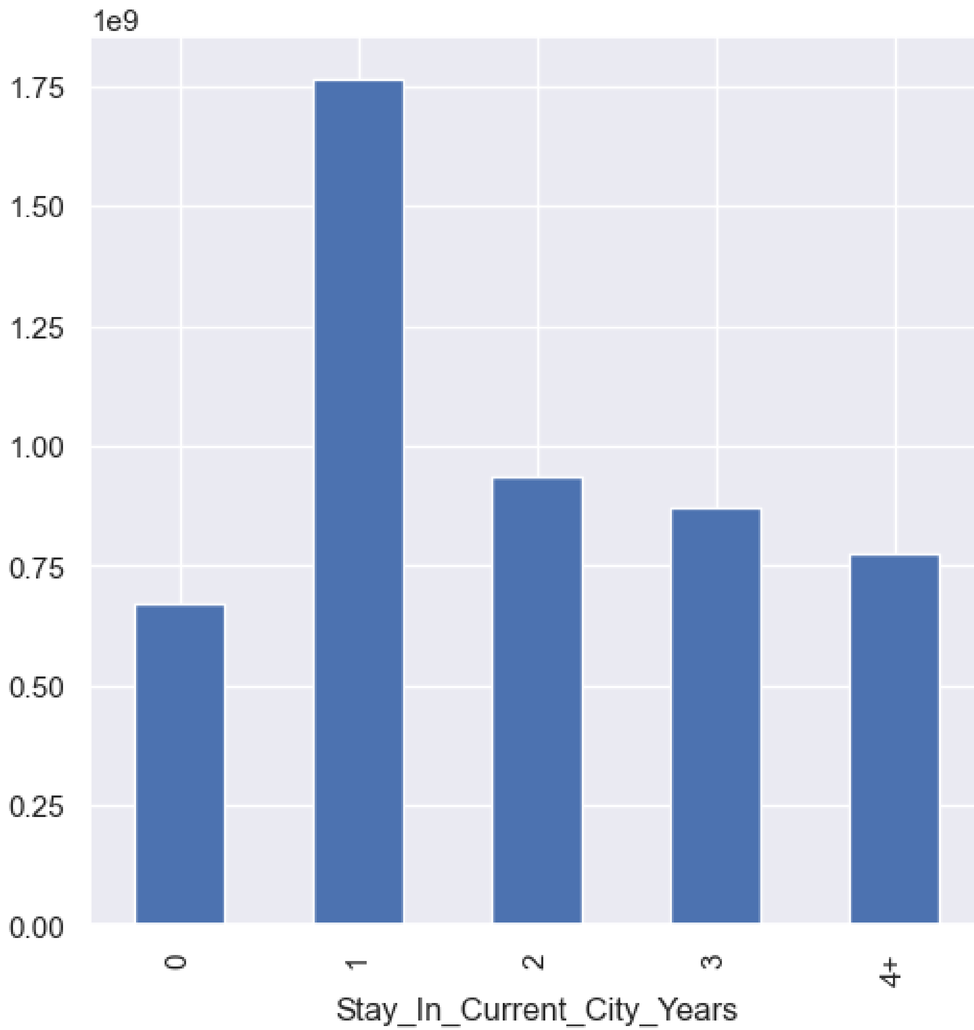
[61]:
```
sns.countplot(x = 'City_Category', hue = 'Age',data = df)
```

[61]: <Axes: xlabel='City_Category', ylabel='count'>



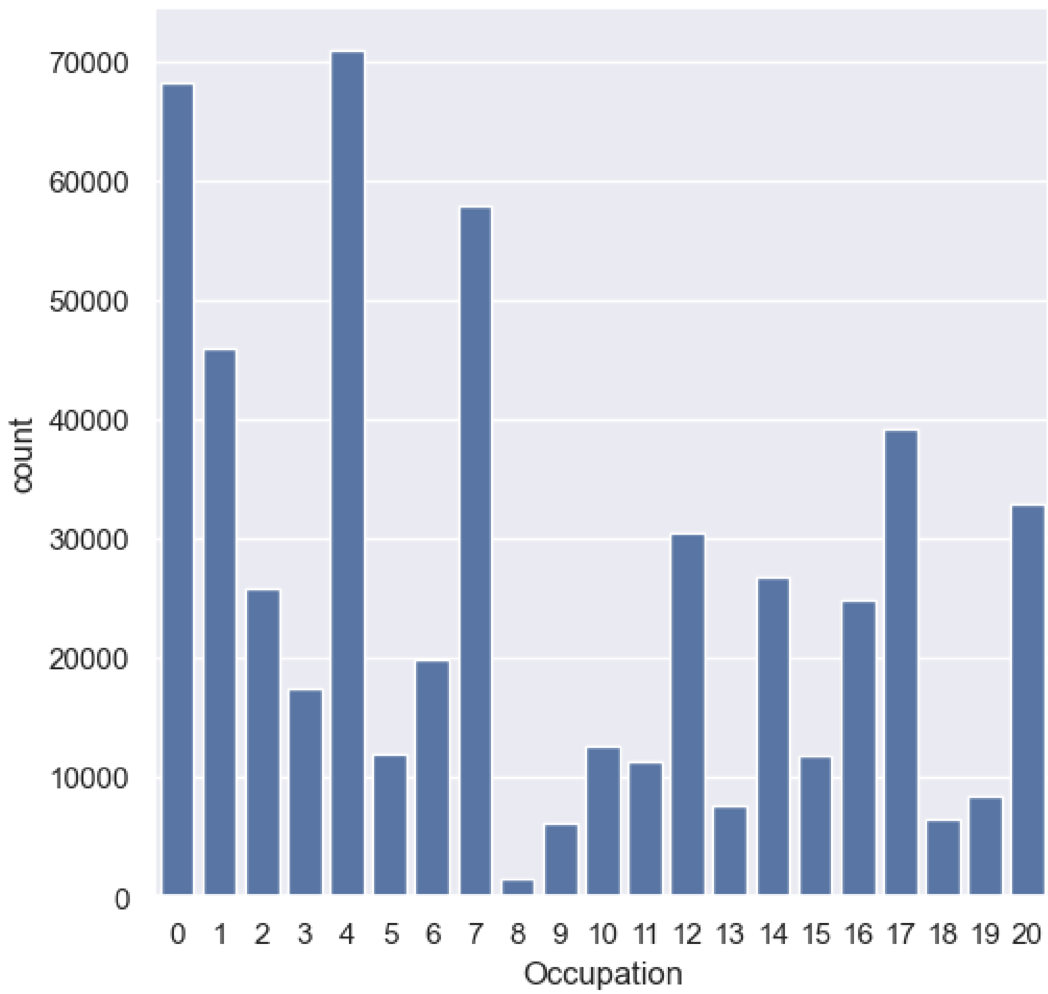
[63]:
```
df.groupby('Stay_In_Current_City_Years').sum()['Purchase'].plot(kind = 'bar')
```

[63]: <Axes: xlabel='Stay_In_Current_City_Years'>



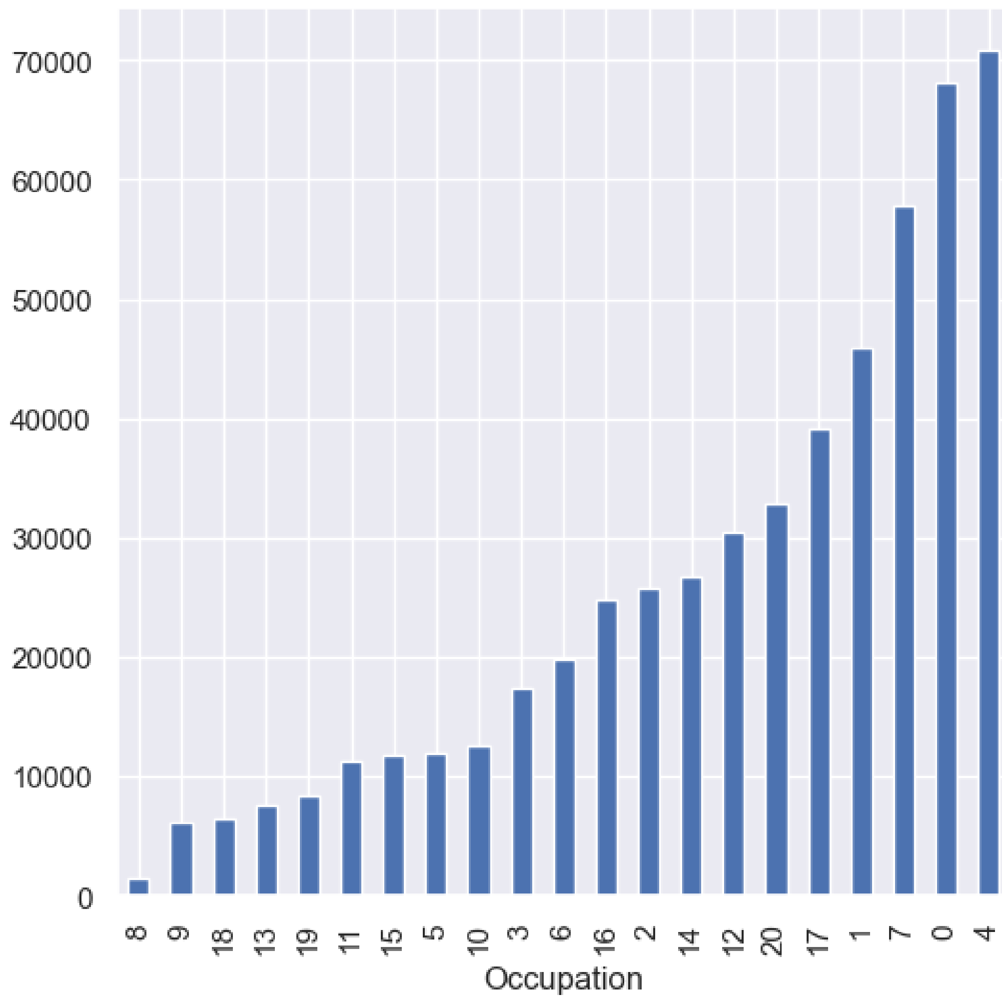
[67]:
```
sns.countplot(x = df['Occupation'])
```

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



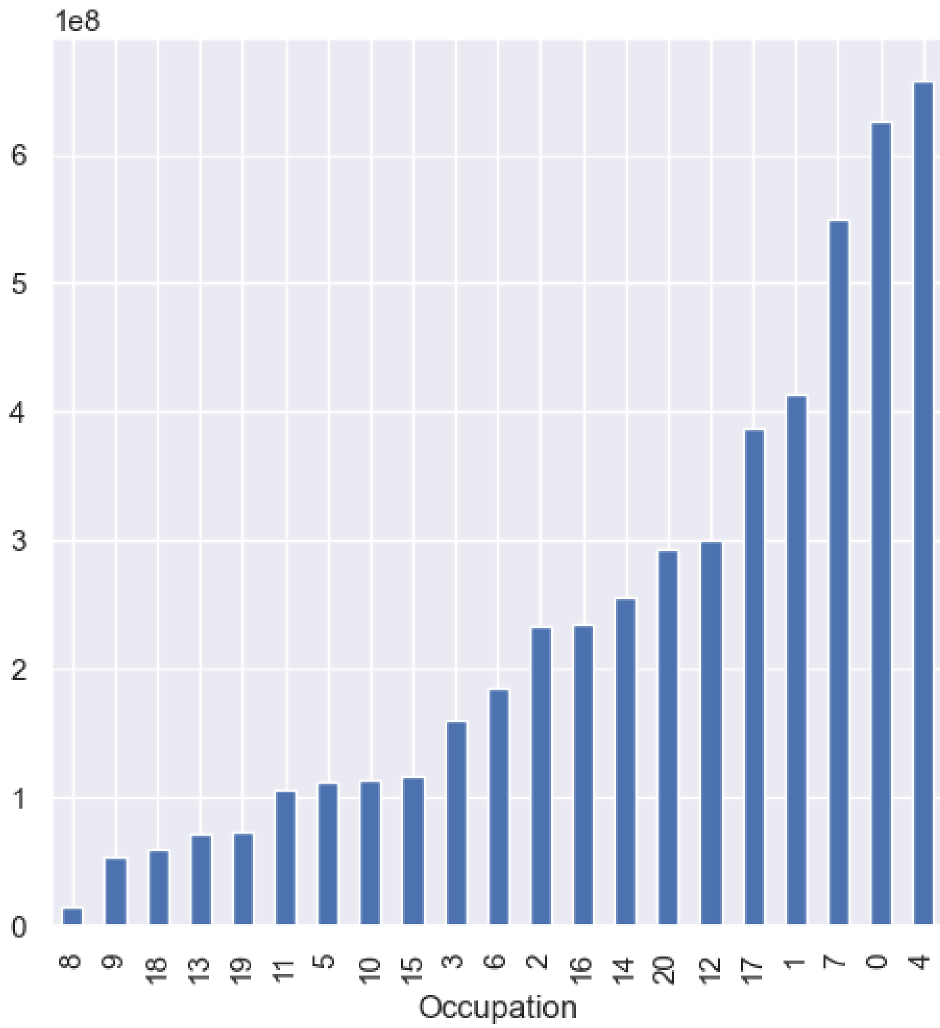
[68]:
```
df.groupby('Occupation').size().sort_values().plot(kind = 'bar')
```

[68]: <Axes: xlabel='Occupation'>



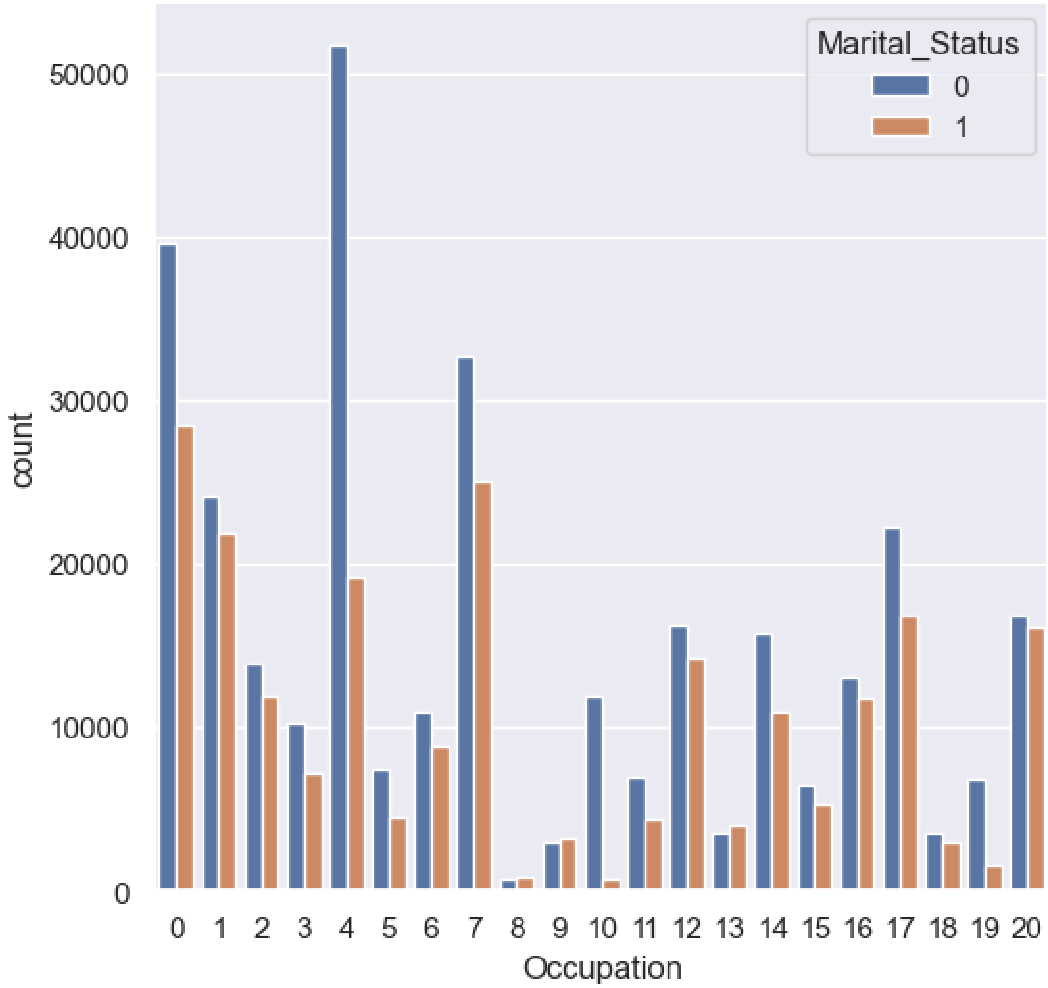
[69]:
```
df.groupby('Occupation').sum()['Purchase'].sort_values().plot(kind = 'bar')
```

[69]: `<Axes: xlabel='Occupation'>`



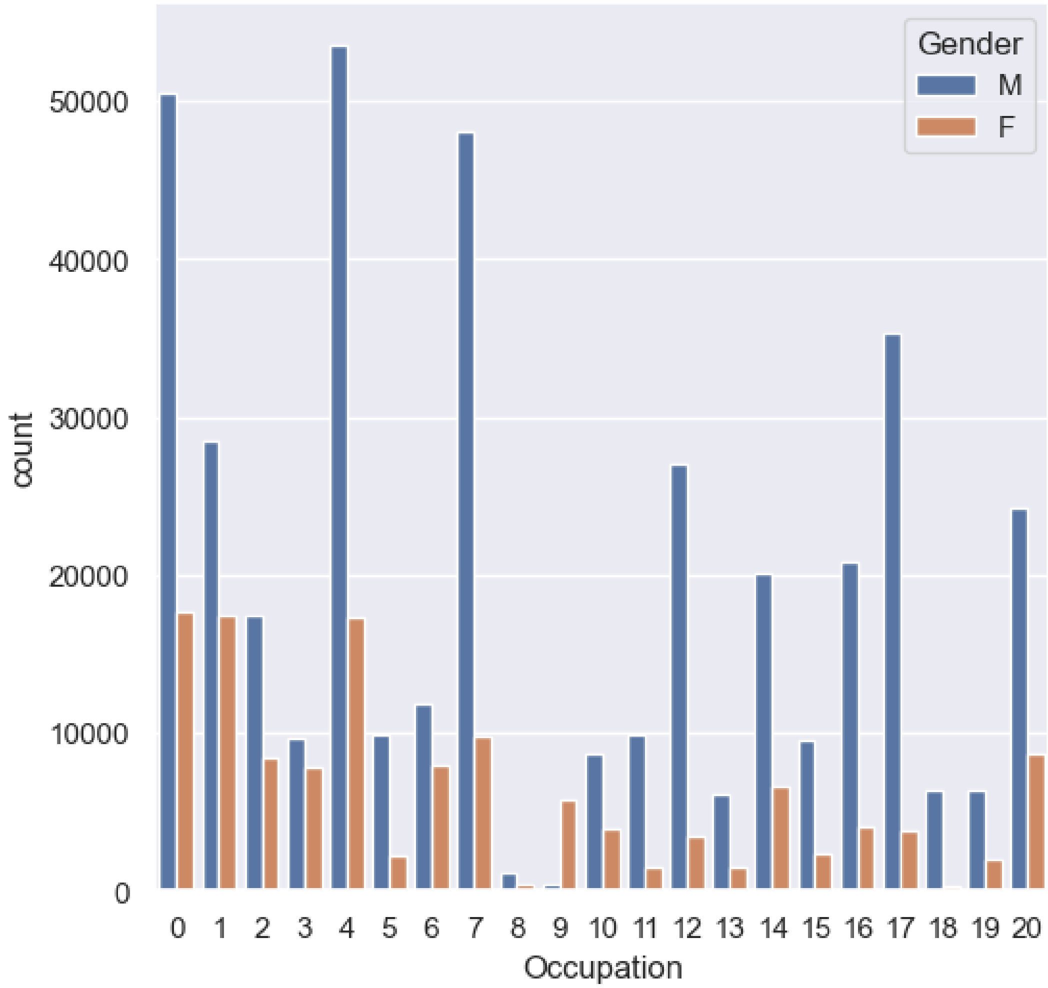

```
[74]: sns.countplot(x = 'Occupation', hue = 'Marital_Status', data = df)
```

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



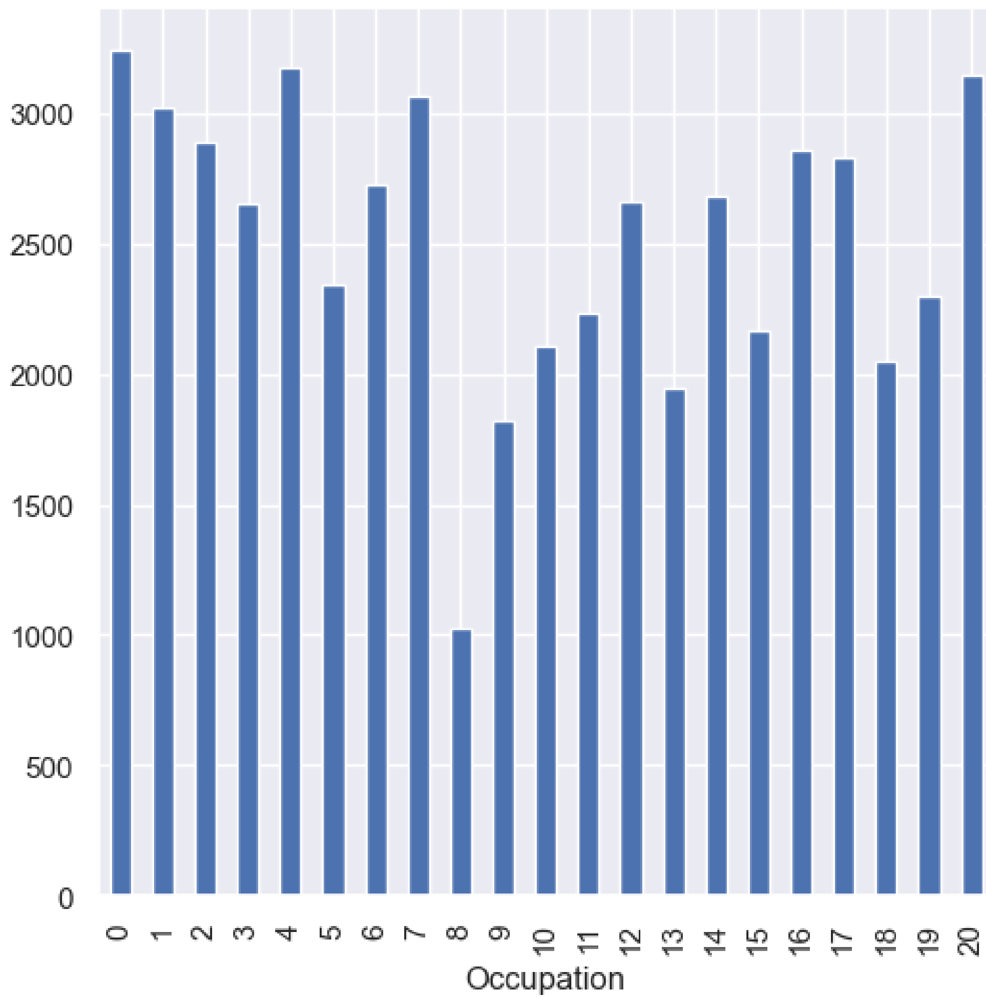
[75]:
```
sns.countplot(x = 'Occupation', hue = 'Gender', data = df)
```

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



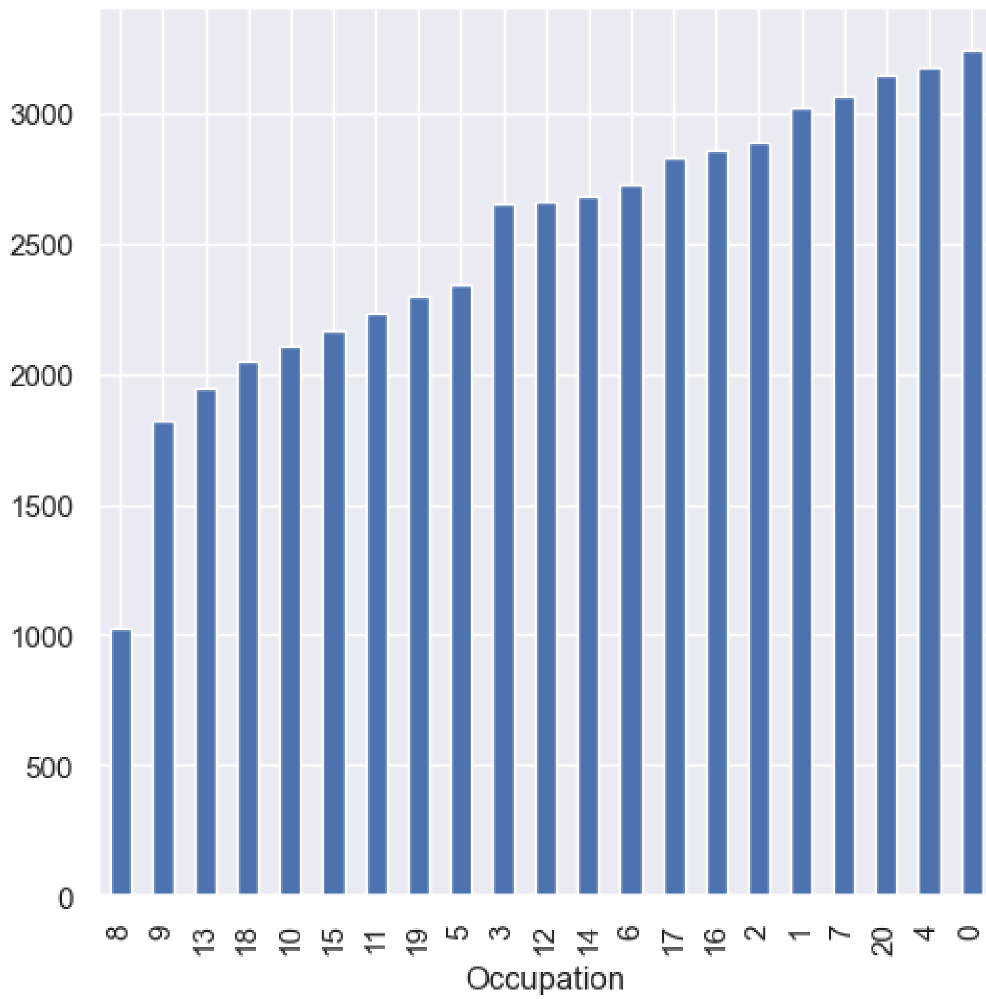
[76]:
```
df.groupby('Occupation').nunique()['Product_ID'].plot(kind = 'bar')
```

[76]: <Axes: xlabel='Occupation'>



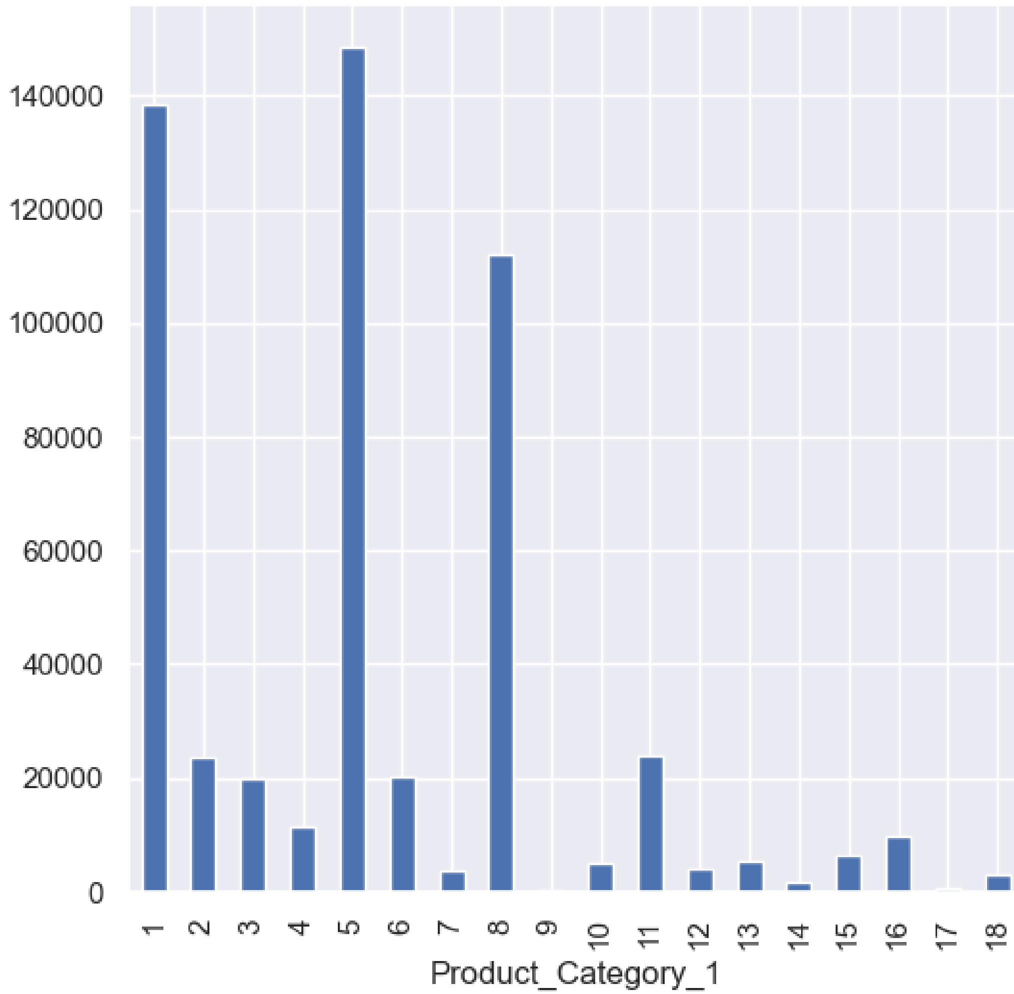

```
[77]: df.groupby('Occupation').nunique()['Product_ID'].sort_values().plot(kind =␣
      ↪'bar')
```

```
[77]: <Axes: xlabel='Occupation'>
```



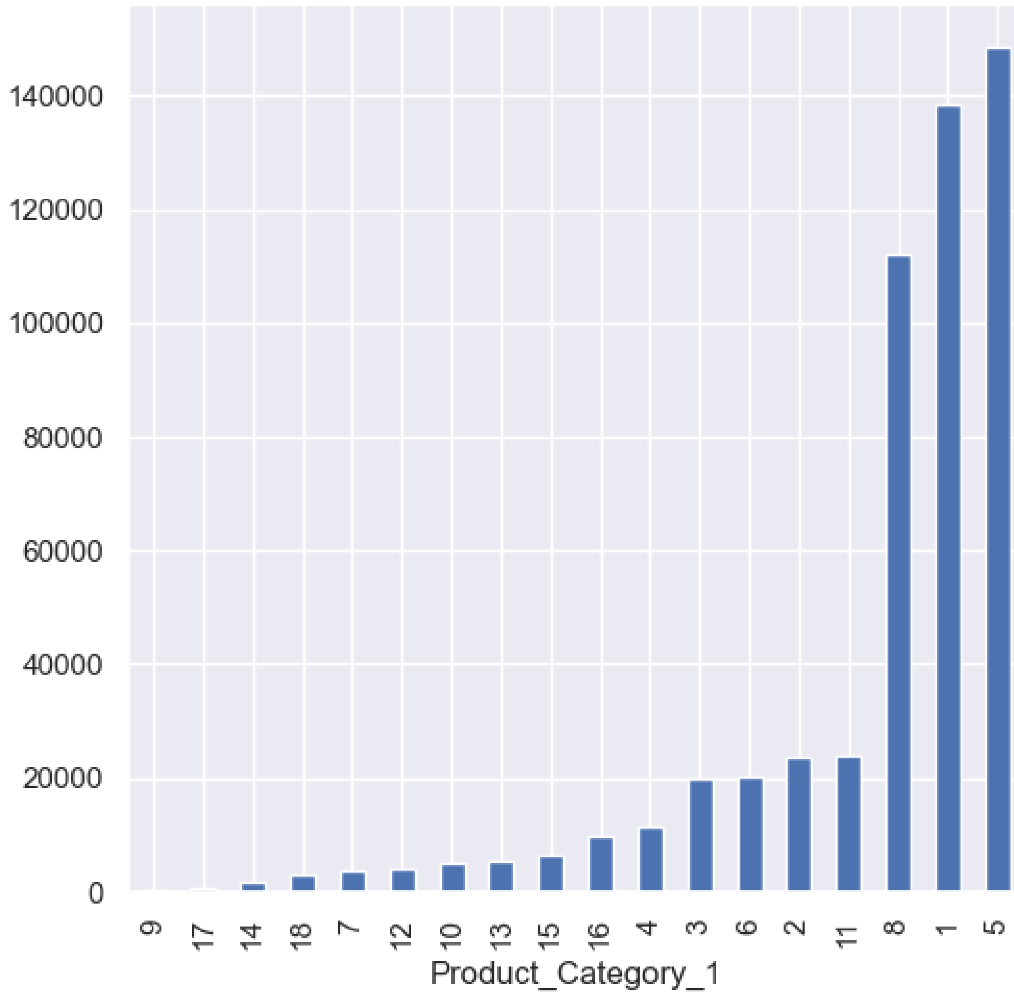
[78]:
```
df.groupby('Product_Category_1').size().plot(kind = 'bar')
```

[78]: <Axes: xlabel='Product_Category_1'>



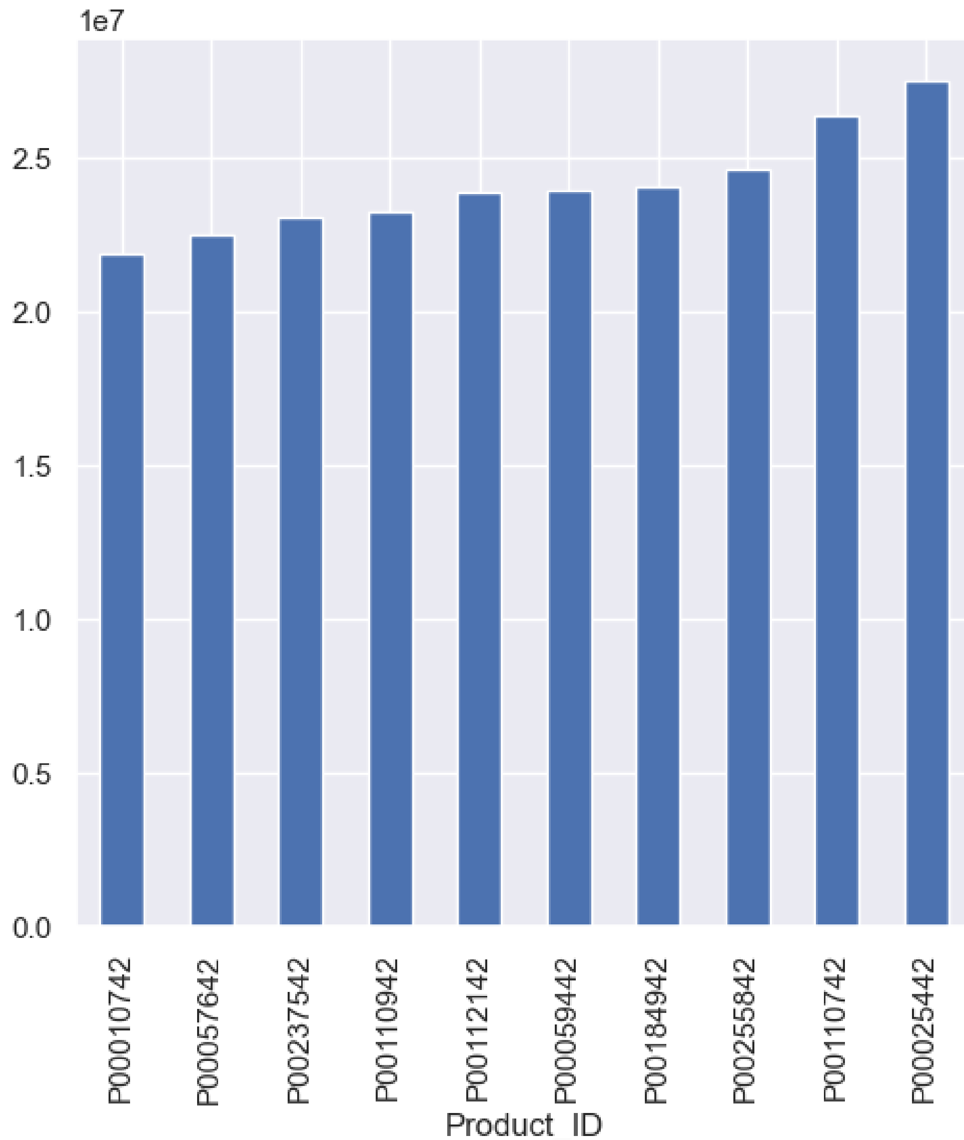

```
[79]: df.groupby('Product_Category_1').size().sort_values().plot(kind = 'bar')
```

```
[79]: <Axes: xlabel='Product_Category_1'>
```



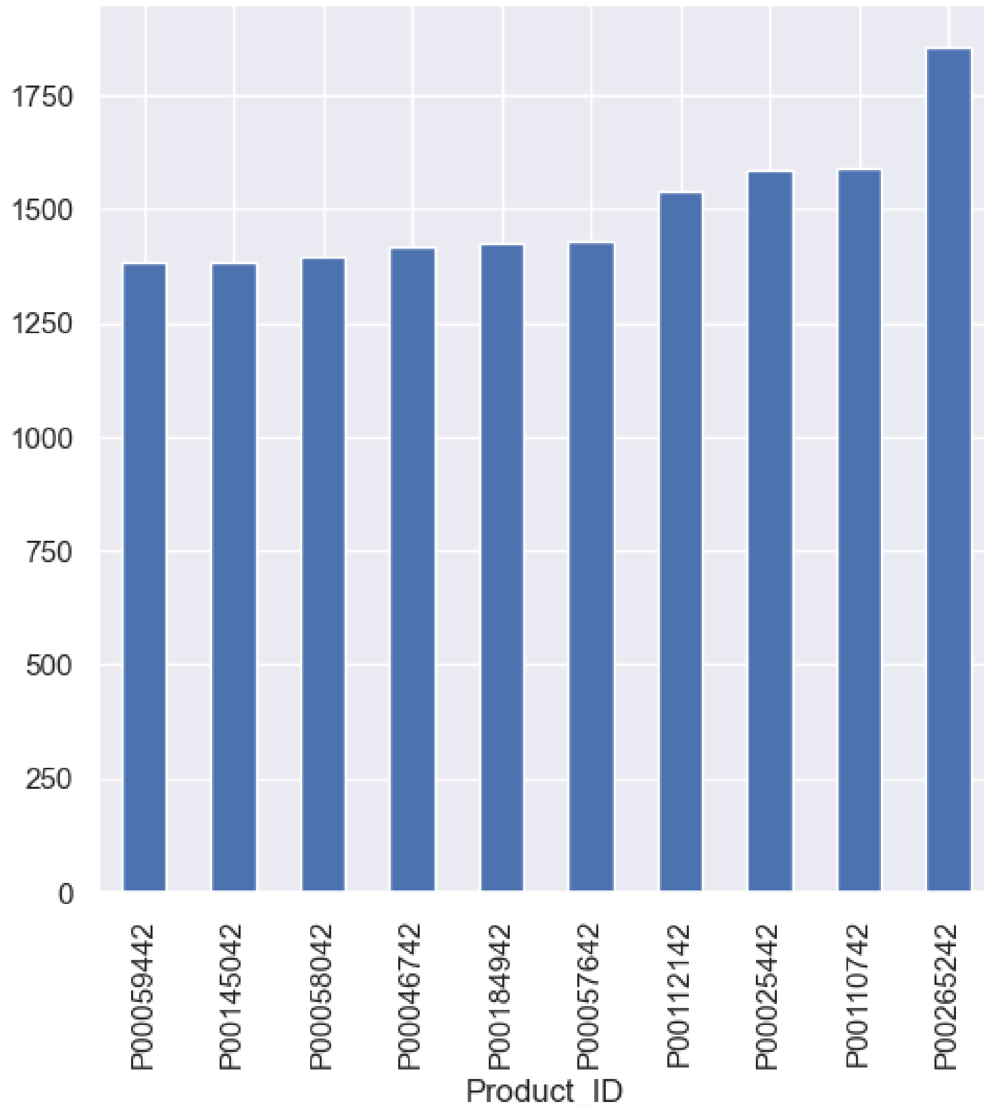

```
[80]: df.groupby('Product_ID').sum()['Purchase'].nlargest(10).sort_values().plot(kind␣
      ↪= 'bar')
```

```
[80]: <Axes: xlabel='Product_ID'>
```



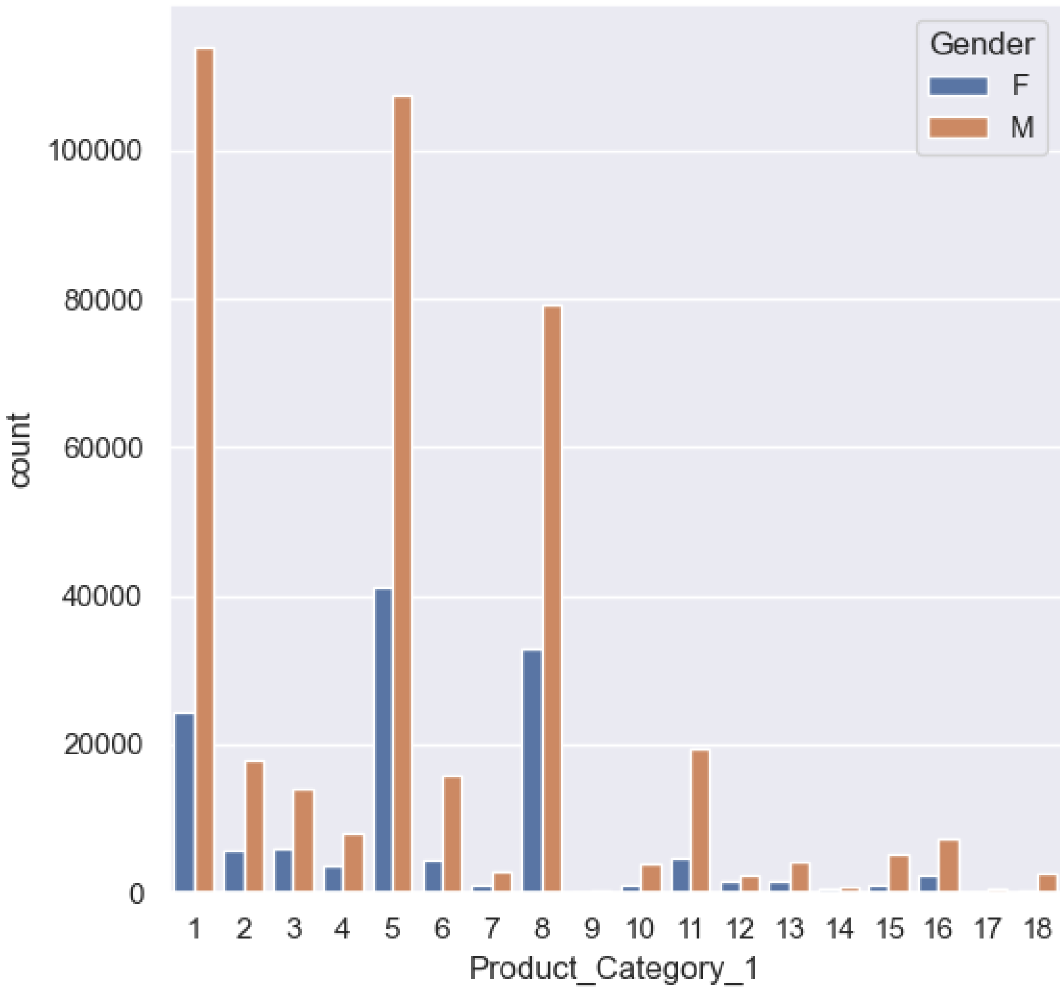
[81]:
```
df.groupby('Product_ID').size().nlargest(10).sort_values().plot(kind = 'bar')
```

[81]: <Axes: xlabel='Product_ID'>



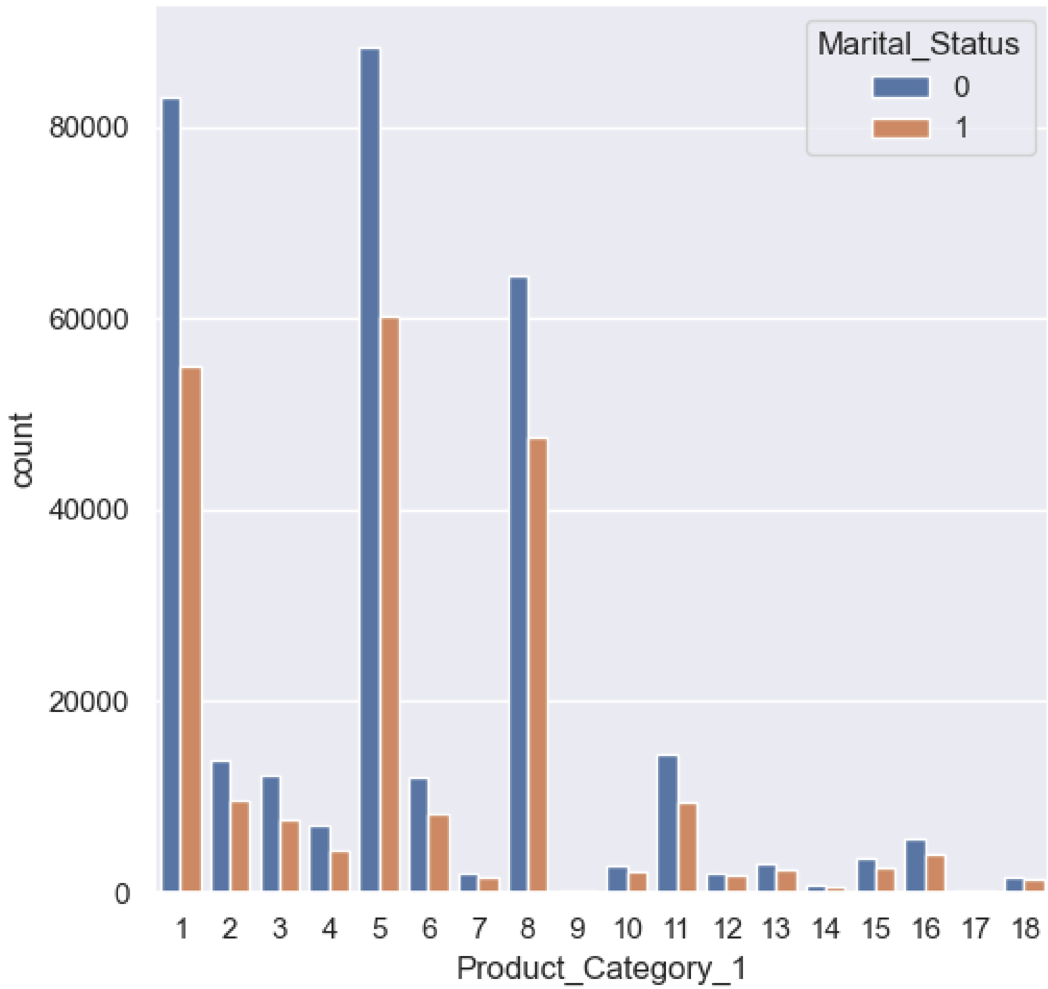
[83]:
```
sns.countplot(x = 'Product_Category_1', hue = 'Gender', data = df)
```

[83]: <Axes: xlabel='Product_Category_1', ylabel='count'>



[84]:
```
sns.countplot(x = 'Product_Category_1', hue = 'Marital_Status', data = df)
```

[84]: <Axes: xlabel='Product_Category_1', ylabel='count'>

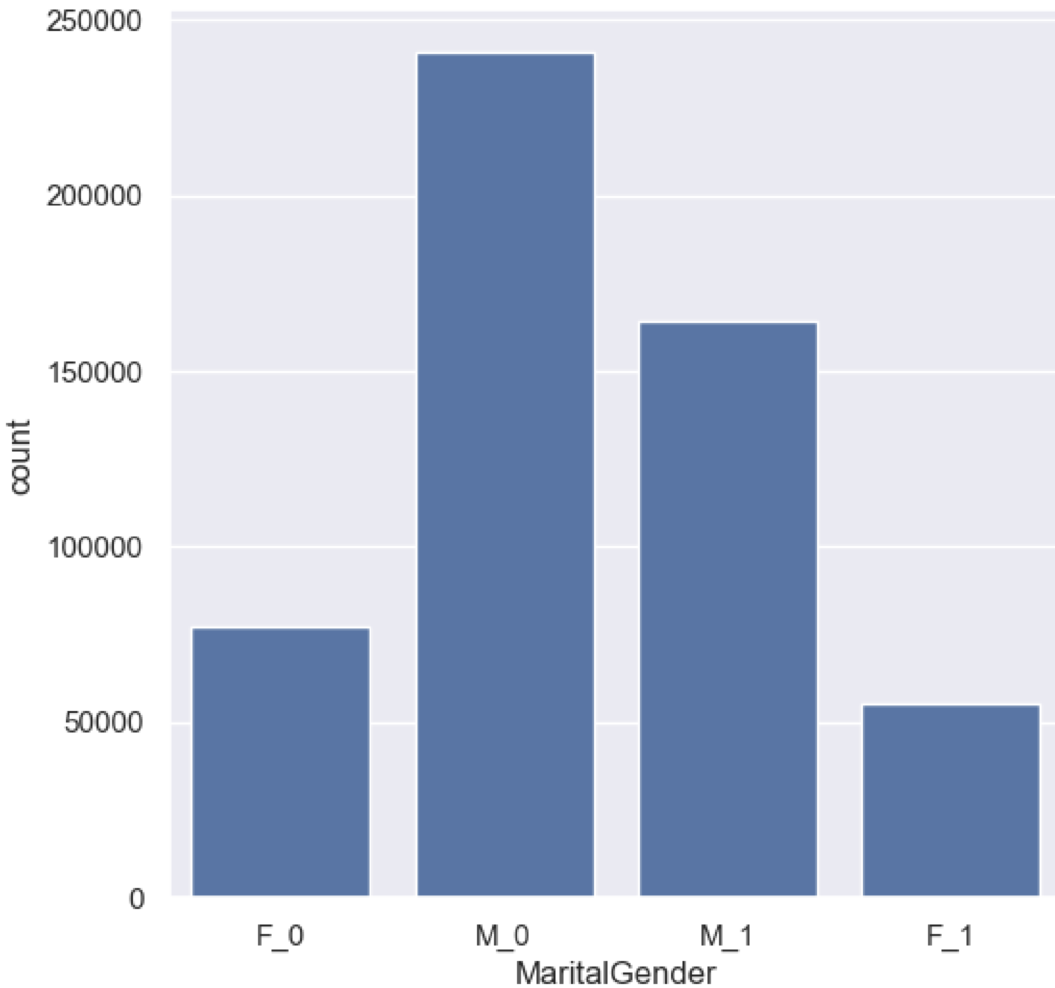


13 Combining Age & Marital Status

[85]:
```
l = []
for i in range(len(df)):
    l.append(df['Gender'][i] +"_"+ str(df['Marital_Status'][i]))

df['MaritalGender'] = l
```

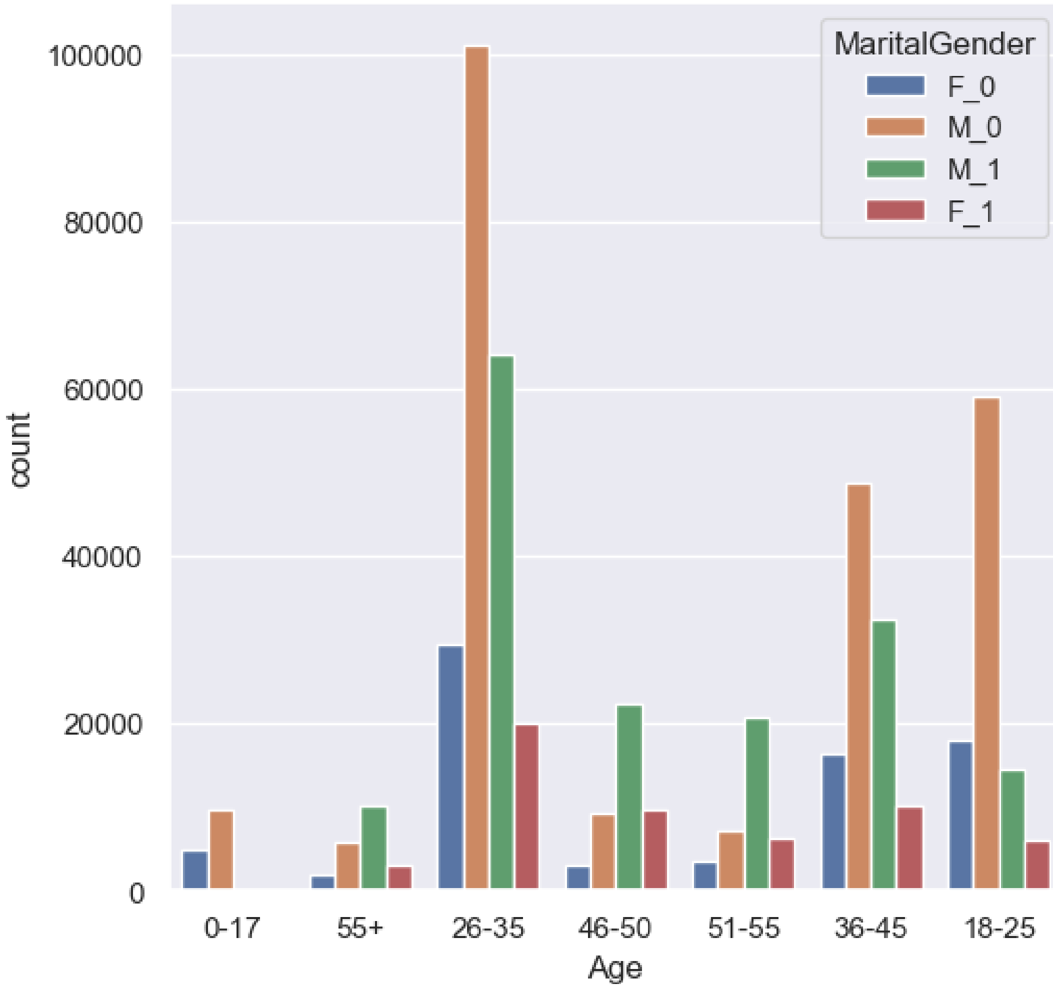
[86]:
```
sns.countplot(x = df['MaritalGender'])
```

[86]: <Axes: xlabel='MaritalGender', ylabel='count'>



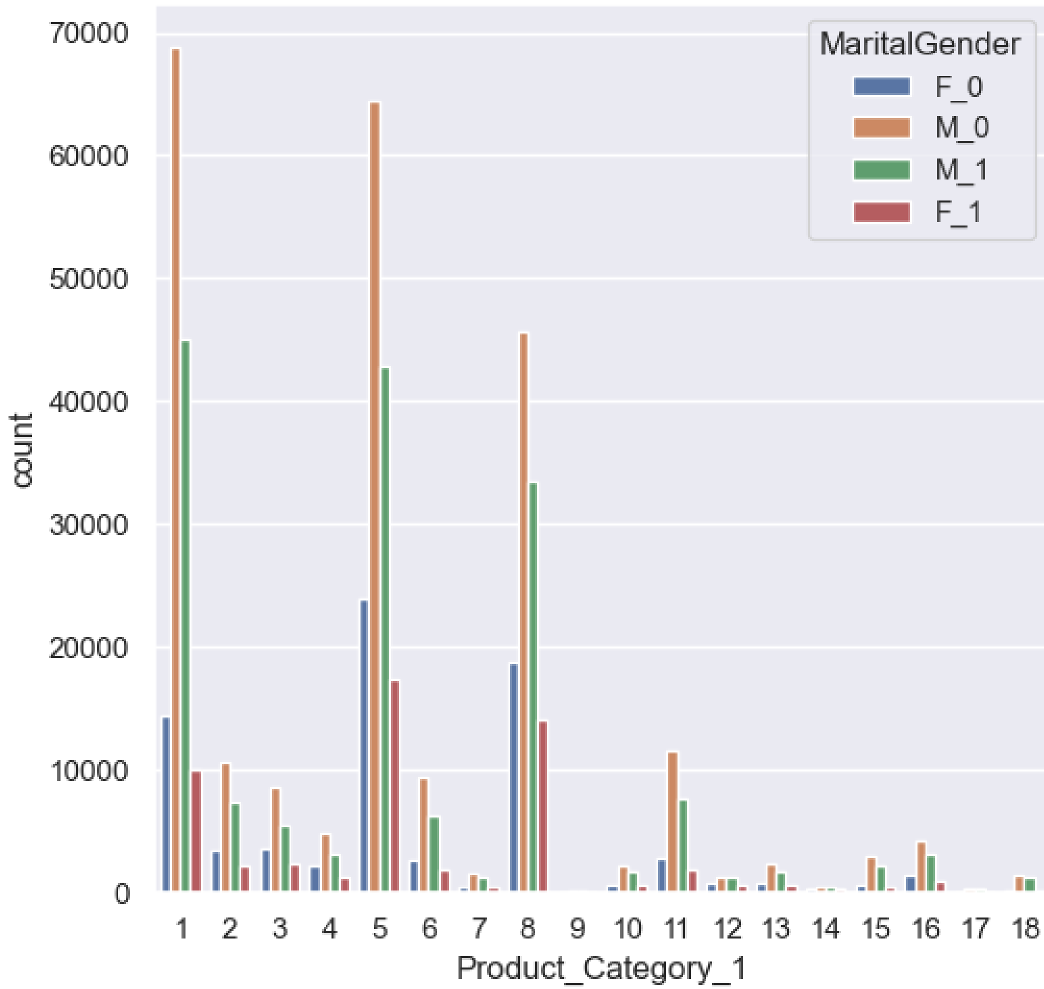

```
[87]: sns.countplot(x = df['Age'], hue = df['MaritalGender'])
```

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



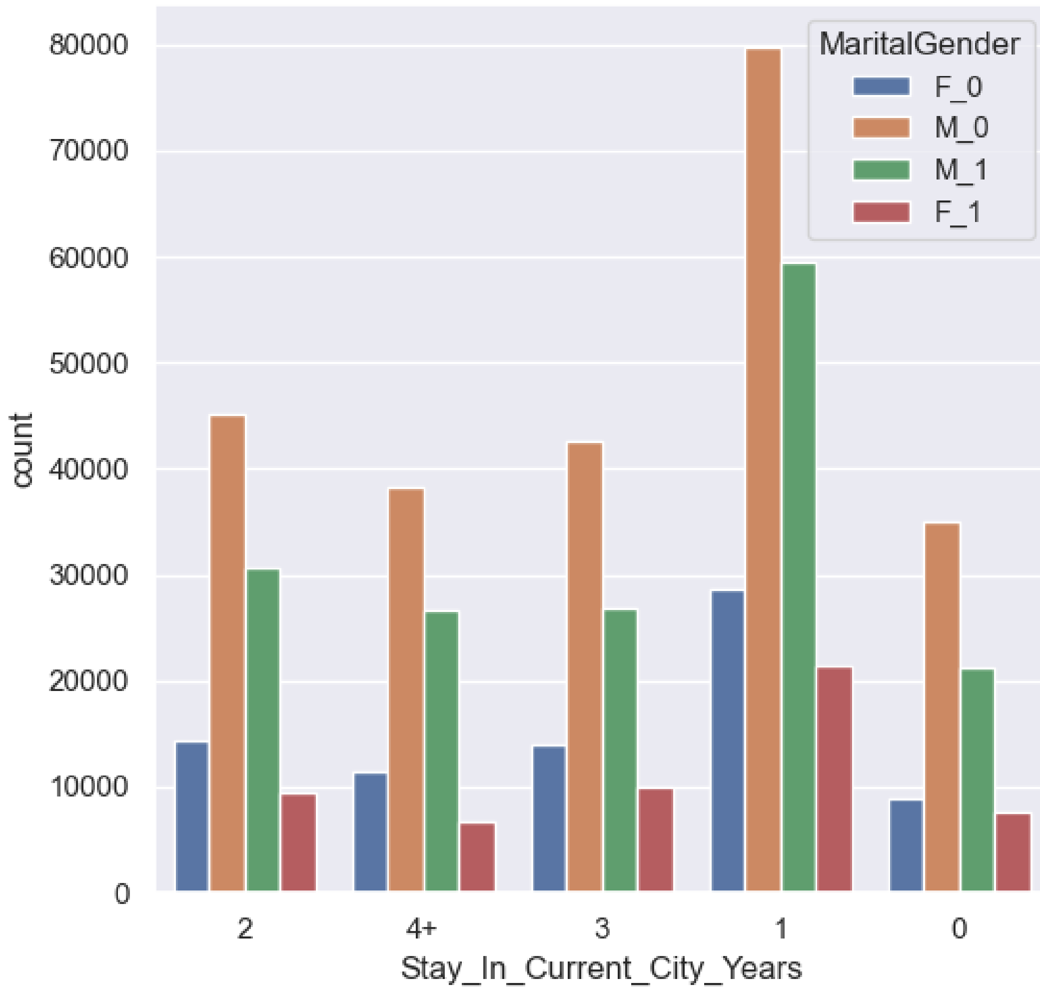
[88]:
```
sns.countplot(x = df['Product_Category_1'], hue = df['MaritalGender'])
```

[88]: <Axes: xlabel='Product_Category_1', ylabel='count'>



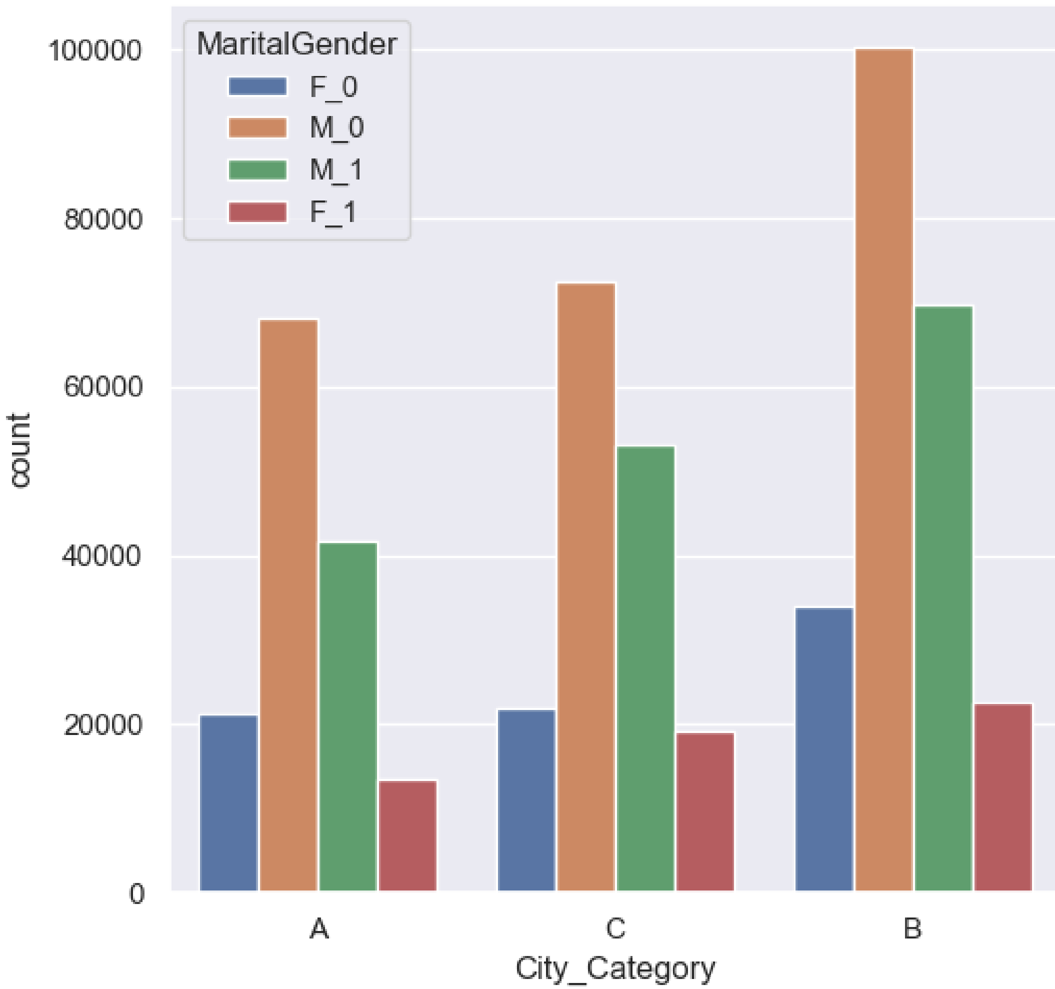

```
[89]: sns.countplot(x = df['Stay_In_Current_City_Years'], hue = df['MaritalGender'])
```

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




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

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


MaritalGender
F_0
M_0
M_1
F_1
100000
80000
60000
40000
20000
0
count
A
C
B
City_Category

```
[ ]:
```