# black-friday-sales

October 27, 2023

[9]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[10]:
```
import pandas as pd
import numpy as np
```

[11]:
```
df= pd.read_csv('/content/drive/MyDrive/100 Data Science/Day 3 - Black Friday␣
↪Sale/train.csv')
```

[12]:
```
df.head(5)
```

[12]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | Stay_In_Current_City_Years | Marital_Status | Product_Category_1 | Product_Category_2 | Product_Category_3 | Purchase |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1000001 | P00069042 | F | 0-17 | 10 | A | 2 | 0 | 3 | NaN | NaN | 8370 |
| 1 | 1000001 | P00248942 | F | 0-17 | 10 | A | 2 | 0 | 1 | 6.0 | 14.0 | 15200 |
| 2 | 1000001 | P00087842 | F | 0-17 | 10 | A | 2 | 0 | 12 | NaN | NaN | 1422 |
| 3 | 1000001 | P00085442 | F | 0-17 | 10 | A | 2 | 0 | 12 | 14.0 | NaN | 1057 |
| 4 | 1000002 | P00285442 | M | 55+ | 16 | C | 4+ | 0 | 8 | NaN | NaN | 7969 |

[13]:
```
df.isnull().sum()
```

[13]:
```
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          173638
Product_Category_3          383247
Purchase                         0
dtype: int64
```

[14]:
```
df.duplicated().sum()
```

[14]:
```
0
```

[15]:
```
df.nunique()
```

[15]:
```
User_ID                      5891
Product_ID                   3631
Gender                          2
Age                             7
Occupation                     21
City_Category                   3
Stay_In_Current_City_Years      5
Marital_Status                  2
Product_Category_1             20
Product_Category_2             17
Product_Category_3             15
Purchase                    18105
dtype: int64
```

[16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  object
```

```
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64
 8   Product_Category_1          550068 non-null  int64
 9   Product_Category_2          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

[55]:
```
df.fillna(0)
```

[55]:

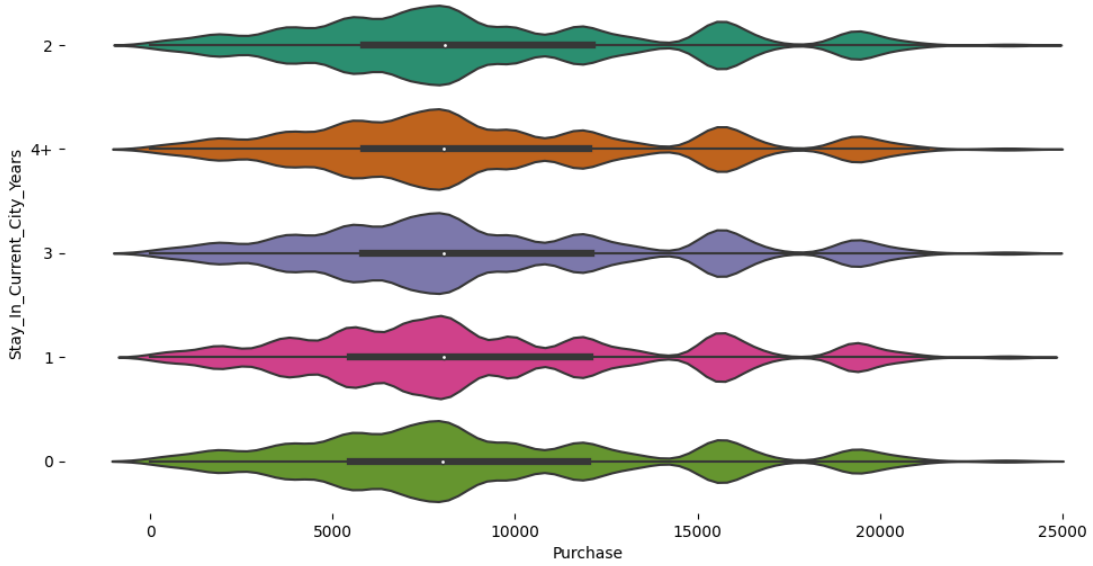
| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | \ |
|---|---|---|---|---|---|---|---|
| 0 | 1000001 | P00069042 | F | 0-17 | 10 | A | |
| 1 | 1000001 | P00248942 | F | 0-17 | 10 | A | |
| 2 | 1000001 | P00087842 | F | 0-17 | 10 | A | |
| 3 | 1000001 | P00085442 | F | 0-17 | 10 | A | |
| 4 | 1000002 | P00285442 | M | 55+ | 16 | C | |
| … | … | … | … | … | … | … | |
| 550063 | 1006033 | P00372445 | M | 51-55 | 13 | B | |
| 550064 | 1006035 | P00375436 | F | 26-35 | 1 | C | |
| 550065 | 1006036 | P00375436 | F | 26-35 | 15 | B | |
| 550066 | 1006038 | P00375436 | F | 55+ | 1 | C | |
| 550067 | 1006039 | P00371644 | F | 46-50 | 0 | B | |

| | 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 | |
| … | … | … | … | |
| 550063 | 1 | 1 | 20 | |
| 550064 | 3 | 0 | 20 | |
| 550065 | 4+ | 1 | 20 | |
| 550066 | 2 | 0 | 20 | |
| 550067 | 4+ | 1 | 20 | |

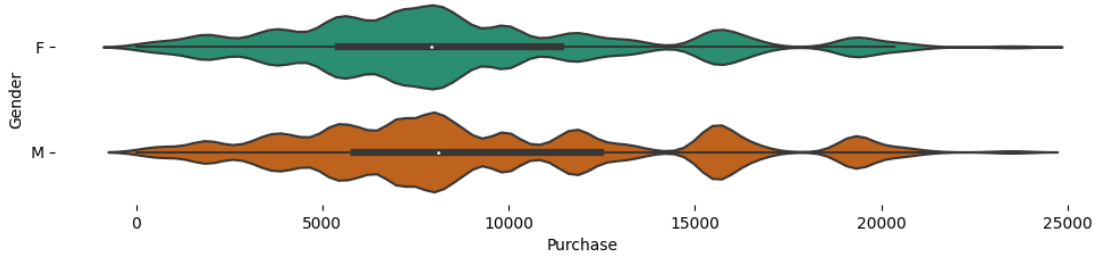
| | Product_Category_2 | Product_Category_3 | Purchase |
|---|---|---|---|
| 0 | 0.0 | 0.0 | 8370 |
| 1 | 6.0 | 14.0 | 15200 |
| 2 | 0.0 | 0.0 | 1422 |
| 3 | 14.0 | 0.0 | 1057 |
| 4 | 0.0 | 0.0 | 7969 |
| … | … | … | … |
| 550063 | 0.0 | 0.0 | 368 |
| 550064 | 0.0 | 0.0 | 371 |
| 550065 | 0.0 | 0.0 | 137 |

```
550066                0.0                0.0       365
550067                0.0                0.0       490

[550068 rows x 12 columns]
```

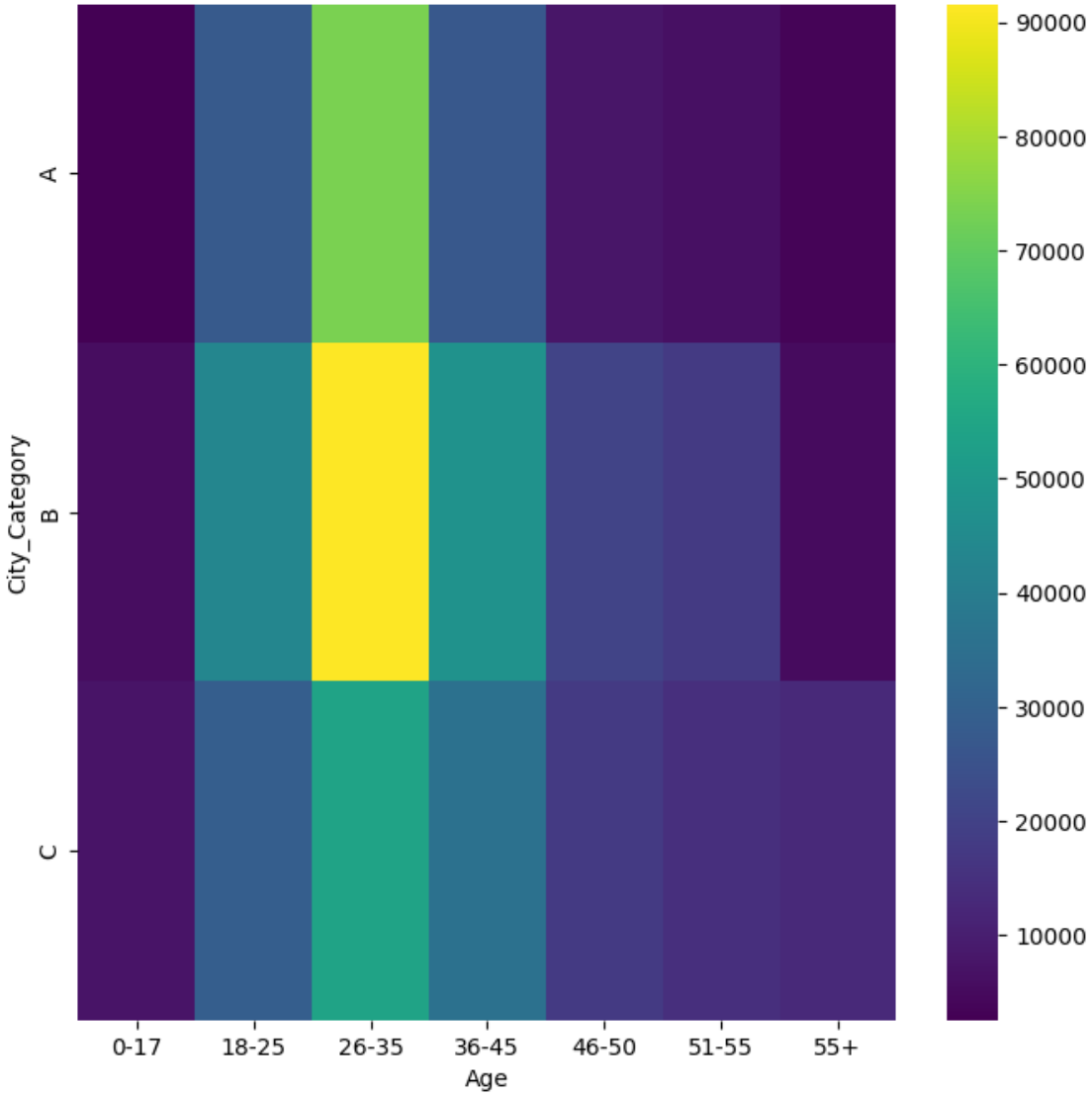
[38]:
```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_22['Stay_In_Current_City_Years'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_22, x='Purchase', y='Stay_In_Current_City_Years',␣
 ↪inner='box', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```



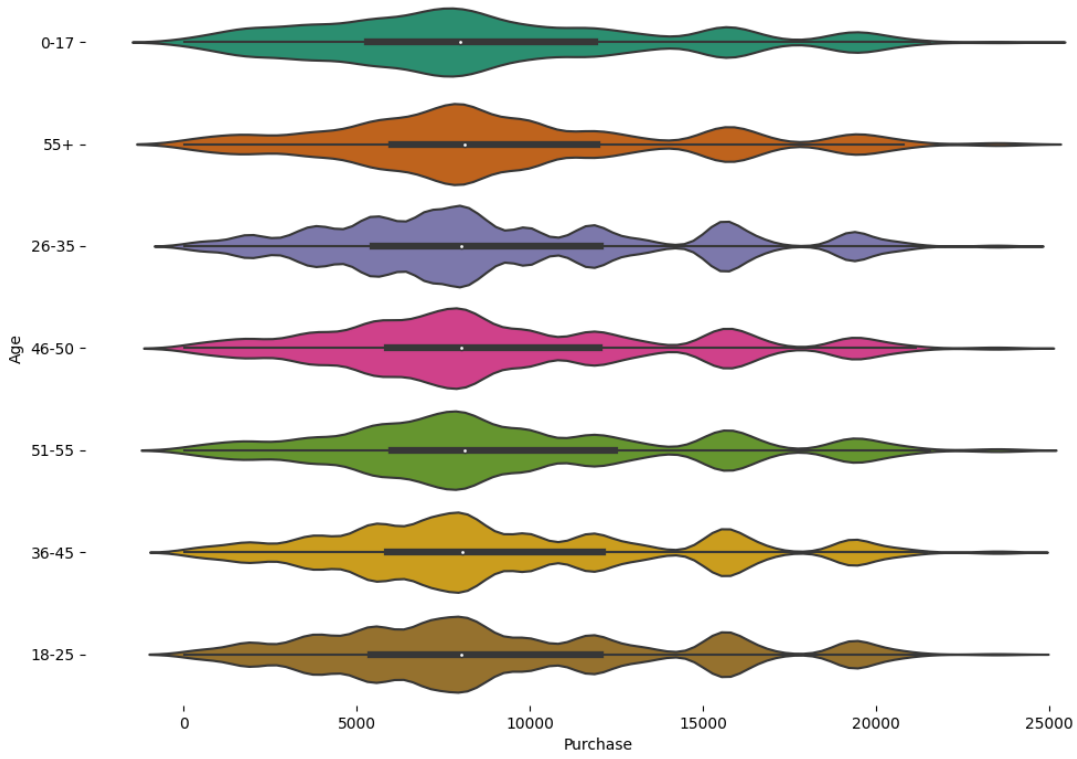

[37]:
```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_19['Gender'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_19, x='Purchase', y='Gender', inner='box', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```



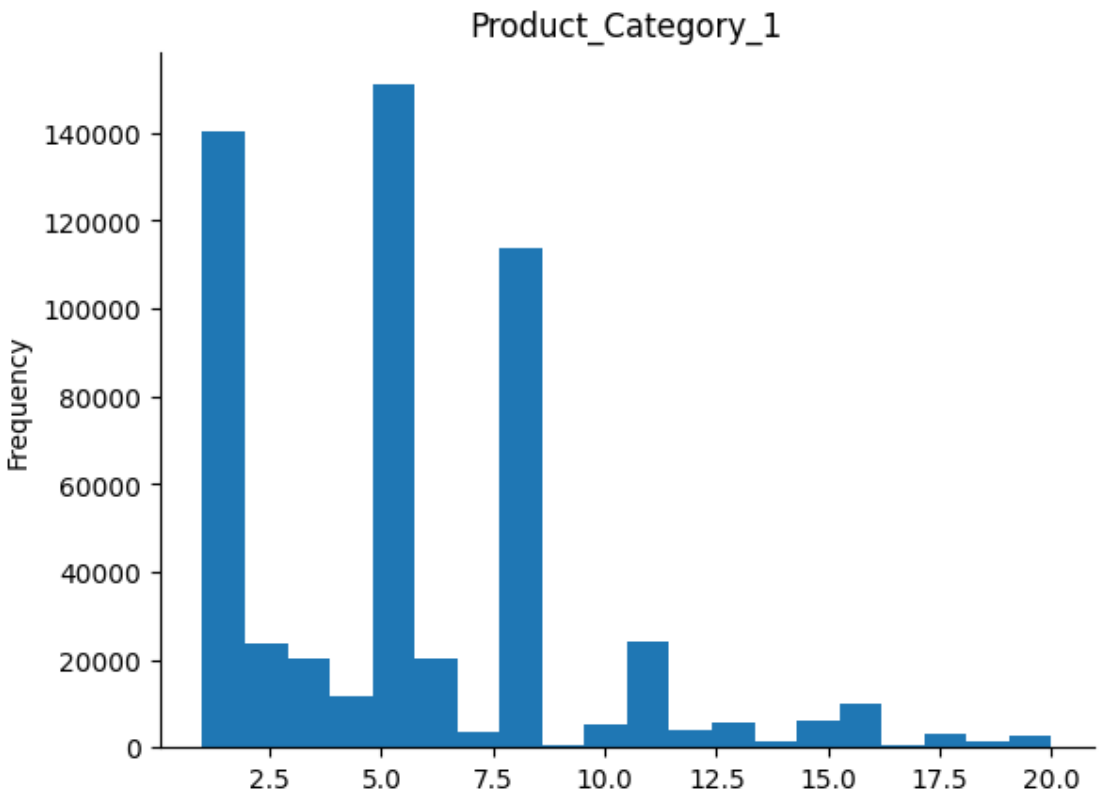
[36]:
```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['City_Category'].value_counts()
    for x_label, grp in _df_17.groupby('Age')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('Age')
_ = plt.ylabel('City_Category')
```



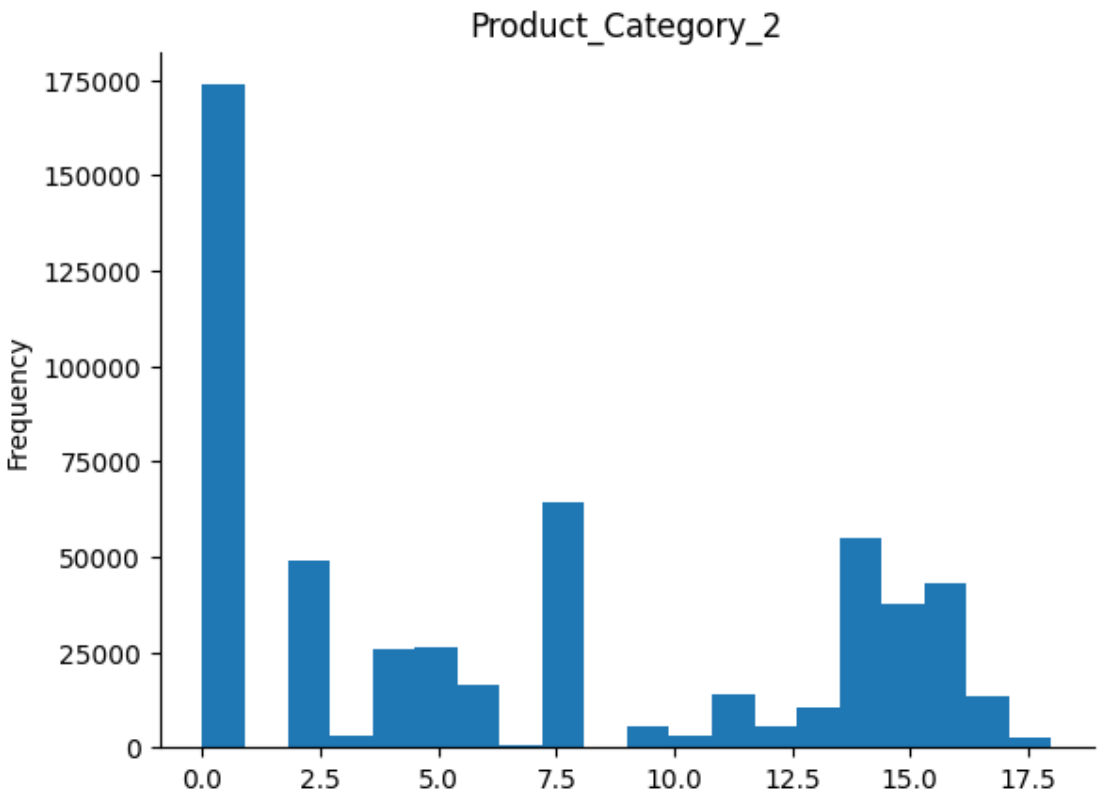
[35]:
```
from matplotlib import pyplot as plt
import seaborn as sns
figsize = (12, 1.2 * len(_df_20['Age'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_20, x='Purchase', y='Age', inner='box', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```



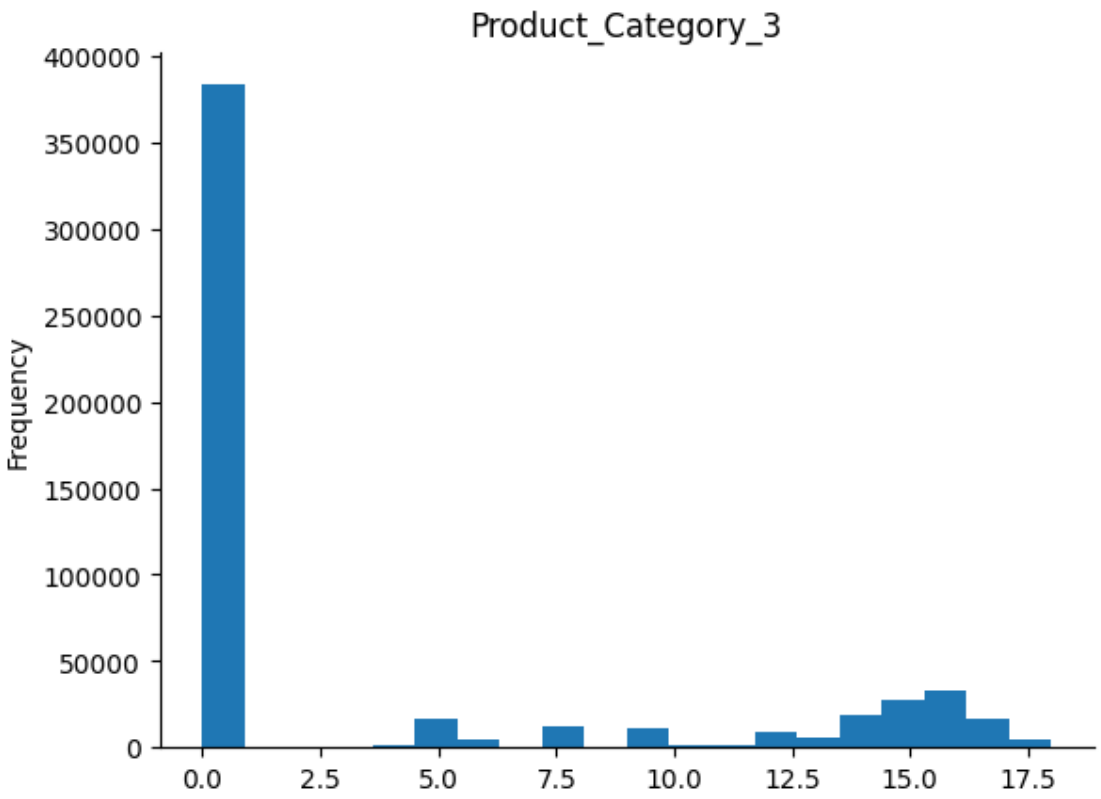

[32]:
```
from matplotlib import pyplot as plt
_df_3['Product_Category_1'].plot(kind='hist', bins=20,␣
  ↪title='Product_Category_1')
plt.gca().spines[['top', 'right',]].set_visible(False)
```



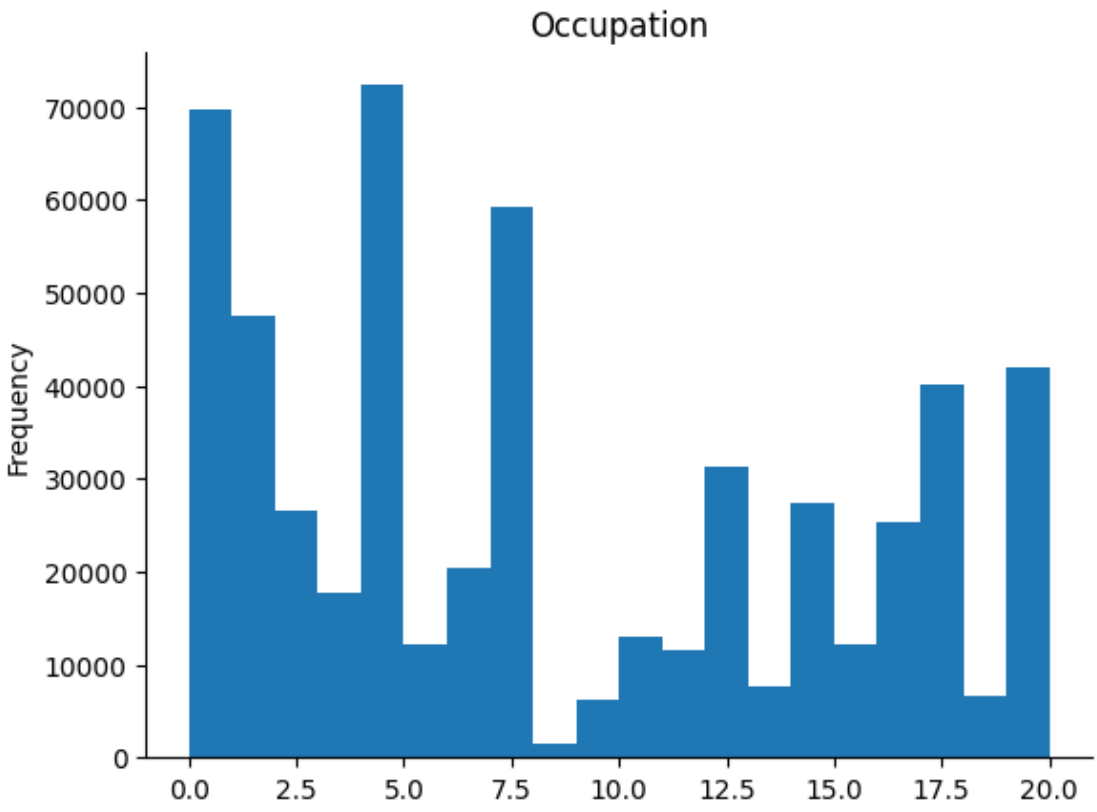

[33]:
```
from matplotlib import pyplot as plt
_df_3['Product_Category_2'].plot(kind='hist', bins=20,␣
  ↪title='Product_Category_2')
plt.gca().spines[['top', 'right',]].set_visible(False)
```




[34]:
```
from matplotlib import pyplot as plt
_df_3['Product_Category_3'].plot(kind='hist', bins=20,␣
  ↪title='Product_Category_3')
plt.gca().spines[['top', 'right',]].set_visible(False)
```




[31]:
```
from matplotlib import pyplot as plt
_df_1['Occupation'].plot(kind='hist', bins=20, title='Occupation')
plt.gca().spines[['top', 'right',]].set_visible(False)
```




[19]:
```
df.describe()
```

[19]:

| | User_ID | Occupation | Marital_Status | Product_Category_1 | \ |
|---|---|---|---|---|---|
| count | 5.500680e+05 | 550068.000000 | 550068.000000 | 550068.000000 | |
| mean | 1.003029e+06 | 8.076707 | 0.409653 | 5.404270 | |
| std | 1.727592e+03 | 6.522660 | 0.491770 | 3.936211 | |
| min | 1.000001e+06 | 0.000000 | 0.000000 | 1.000000 | |
| 25% | 1.001516e+06 | 2.000000 | 0.000000 | 1.000000 | |
| 50% | 1.003077e+06 | 7.000000 | 0.000000 | 5.000000 | |
| 75% | 1.004478e+06 | 14.000000 | 1.000000 | 8.000000 | |
| max | 1.006040e+06 | 20.000000 | 1.000000 | 20.000000 | |

| | Product_Category_2 | Product_Category_3 | Purchase |
|---|---|---|---|
| count | 376430.000000 | 166821.000000 | 550068.000000 |
| mean | 9.842329 | 12.668243 | 9263.968713 |
| std | 5.086590 | 4.125338 | 5023.065394 |
| min | 2.000000 | 3.000000 | 12.000000 |
| 25% | 5.000000 | 9.000000 | 5823.000000 |
| 50% | 9.000000 | 14.000000 | 8047.000000 |
| 75% | 15.000000 | 16.000000 | 12054.000000 |
| max | 18.000000 | 18.000000 | 23961.000000 |

[20]:
```
cat_features = [feature for feature in df.columns if df[feature].dtype == 'O'␣
 ↪and feature != 'Product_ID']
```

[21]:
```
for cat_feature in cat_features:
    print(f'Categories in {cat_feature} variable: ', end="")
    print(df[cat_feature].unique())
```

```
Categories in Gender variable: ['F' 'M']
Categories in Age variable: ['0-17' '55+' '26-35' '46-50' '51-55' '36-45'
'18-25']
Categories in City_Category variable: ['A' 'C' 'B']
Categories in Stay_In_Current_City_Years variable: ['2' '4+' '3' '1' '0']
```

[22]:
```
num_features = [feature for feature in df.columns if df[feature].dtype != 'O'␣
 ↪and feature != 'User_ID']
```

[23]:
```
for num_features in num_features:
    print(f'Categories in {num_features} variable: ', end="")
    print(df[num_features].unique())
```

```
Categories in Occupation variable: [10 16 15  7 20  9  1 12 17  0  3  4 11  8 19
2 18  5 14 13  6]
Categories in Marital_Status variable: [0 1]
Categories in Product_Category_1 variable: [ 3  1 12  8  5  4  2  6 14 11 13 15
7 16 18 10 17  9 20 19]
Categories in Product_Category_2 variable: [nan  6. 14.  2.  8. 15. 16. 11.  5.
3.  4. 12.  9. 10. 17. 13.  7. 18.]
Categories in Product_Category_3 variable: [nan 14. 17.  5.  4. 16. 15.  8.  9.
13.  6. 12.  3. 18. 11. 10.]
Categories in Purchase variable: [ 8370 15200  1422 …   135   123   613]
```

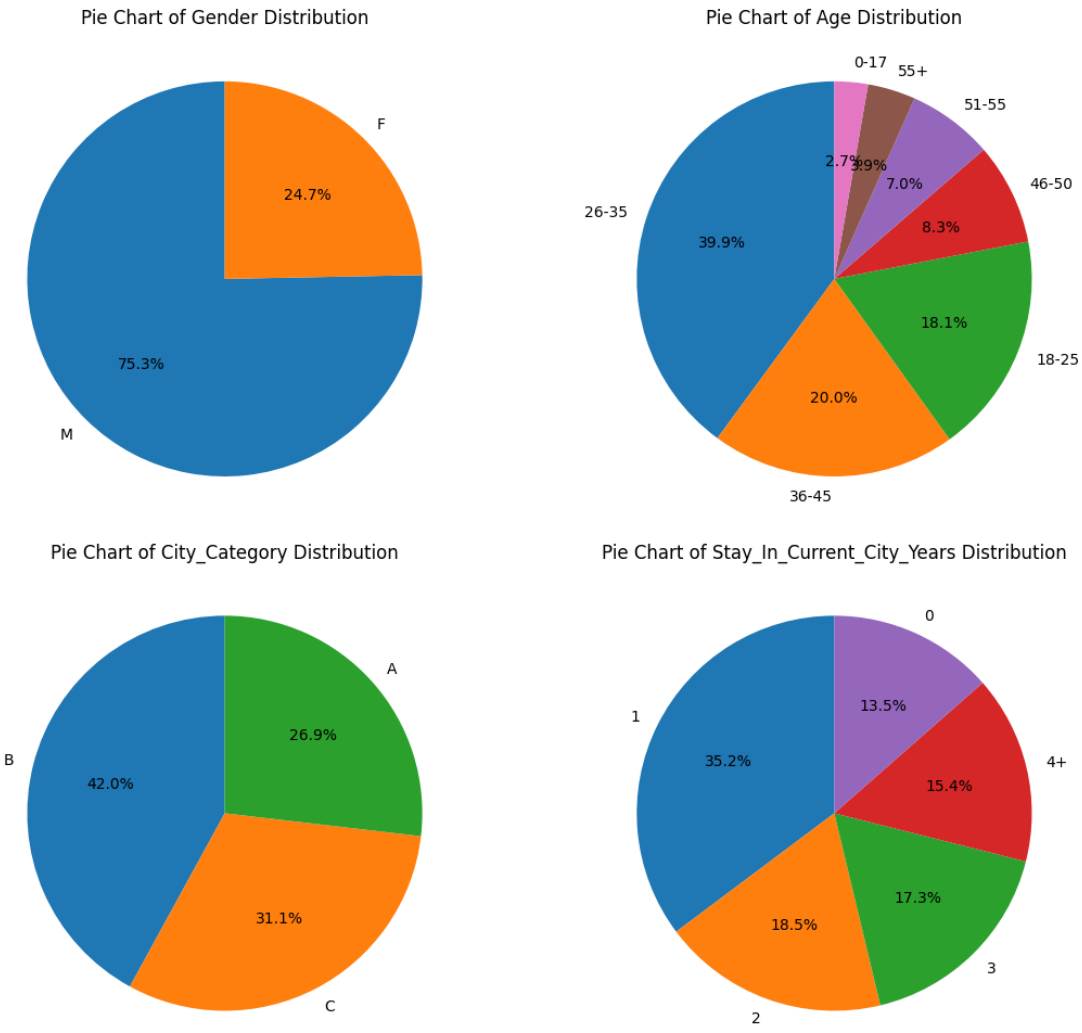
[24]:
```
print(f'We have {len(num_features)} numerical features: {num_features}\n')
print(f'We have {len(cat_features)} categorical features: {cat_features}')
```

```
We have 8 numerical features: Purchase

We have 4 categorical features: ['Gender', 'Age', 'City_Category',
'Stay_In_Current_City_Years']
```

[25]:
```
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
```

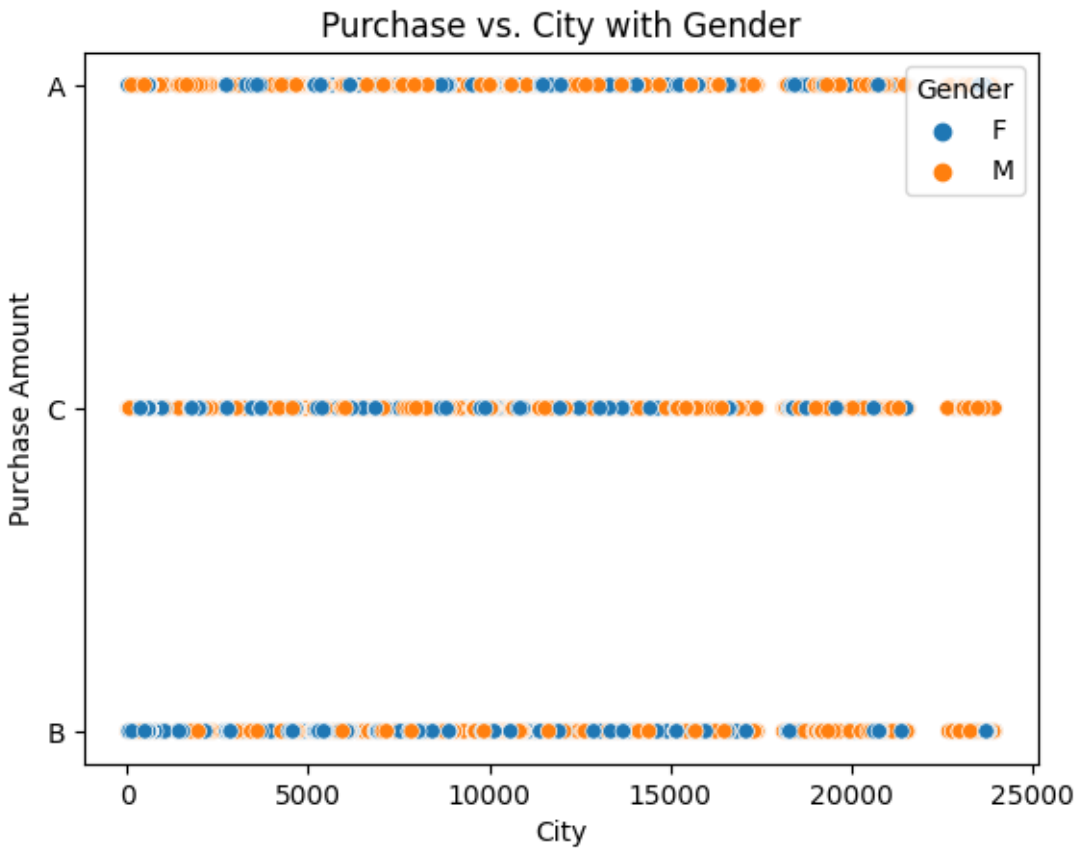
[26]:
```
# Set the number of columns and rows for subplots
n_columns = len(cat_features)
n_rows = (n_columns + 1) // 2
```

```
# Create subplots
fig, axes = plt.subplots(n_rows, 2, figsize=(12, 10))

# Flatten the axes array for easier indexing
axes = axes.flatten()

# Iterate through categorical columns and plot pie charts
for i, column in enumerate(cat_features):
    if i < n_columns:
        # Calculate value counts for the column
        value_counts = df[column].value_counts()

        # Plot a pie chart
        axes[i].pie(value_counts, labels=value_counts.index, autopct='%1.1f%%',␣
 ↪startangle=90)
        axes[i].set_title(f'Pie Chart of {column} Distribution')

# Remove any empty subplots
if n_columns < n_rows * 2:
    for j in range(n_columns, n_rows * 2):
        fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



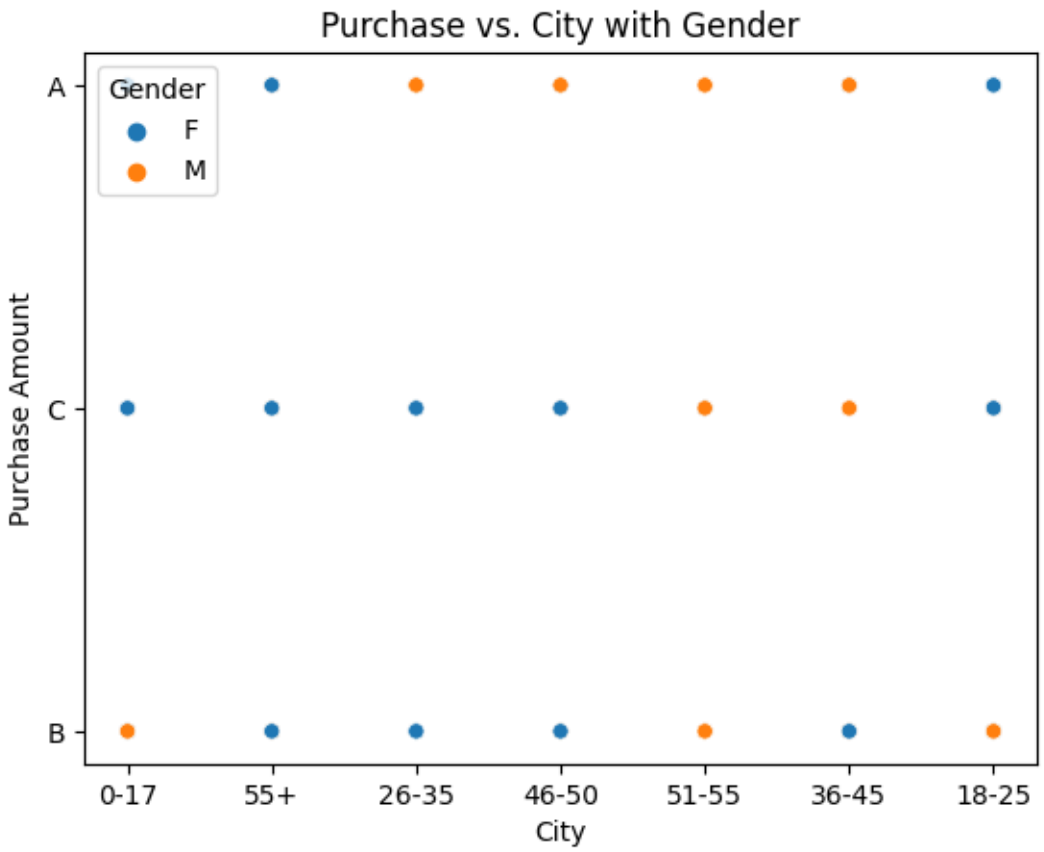

[27]:
```
sns.scatterplot(data=df, x="Purchase", y="City_Category", hue="Gender")

# Customize the plot (optional)
plt.title("Purchase vs. City with Gender")
plt.xlabel("City")
plt.ylabel("Purchase Amount")
plt.legend(title="Gender")
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Creating legend with loc="best" can be slow with large amounts of
data.
  fig.canvas.print_figure(bytes_io, **kw)
```



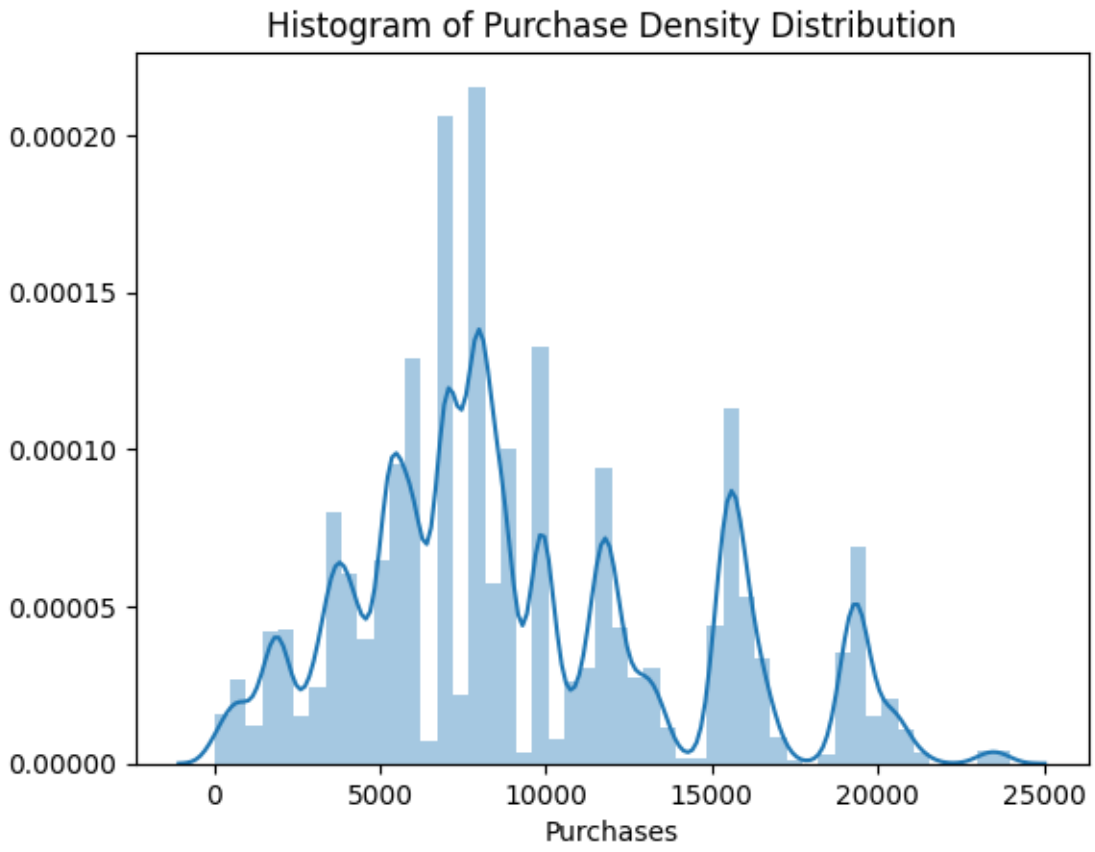

[28]:
```
sns.scatterplot(data=df, x="Age", y="City_Category", hue="Gender")

# Customize the plot (optional)
plt.title("Purchase vs. City with Gender")
plt.xlabel("City")
plt.ylabel("Purchase Amount")
plt.legend(title="Gender")
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151:
UserWarning: Creating legend with loc="best" can be slow with large amounts of
data.
  fig.canvas.print_figure(bytes_io, **kw)
```



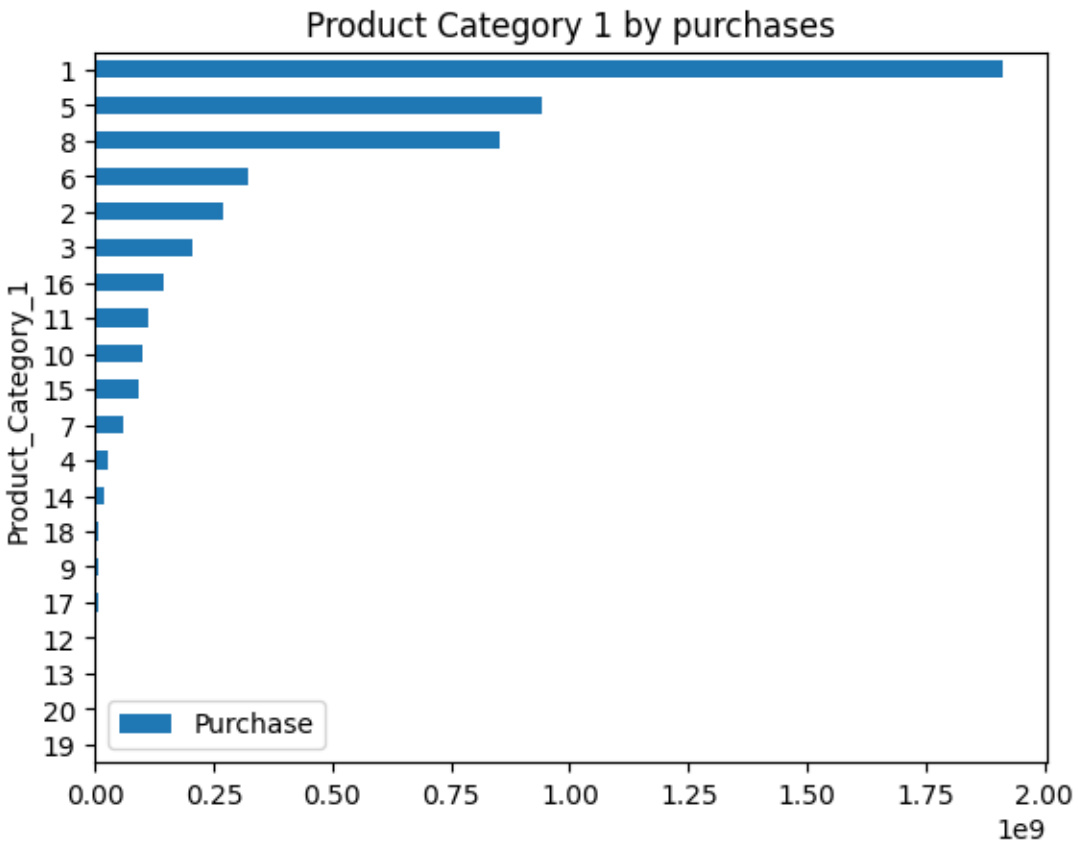

[29]:
```
sns.distplot(df['Purchase'], hist=True, kde=True)

# Customize the plot (optional)
plt.title("Histogram of Purchase Density Distribution")
plt.xlabel("Purchases")
plt.ylabel("")
plt.show()
```

```
<ipython-input-29-501715cf7912>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Purchase'], hist=True, kde=True)
```



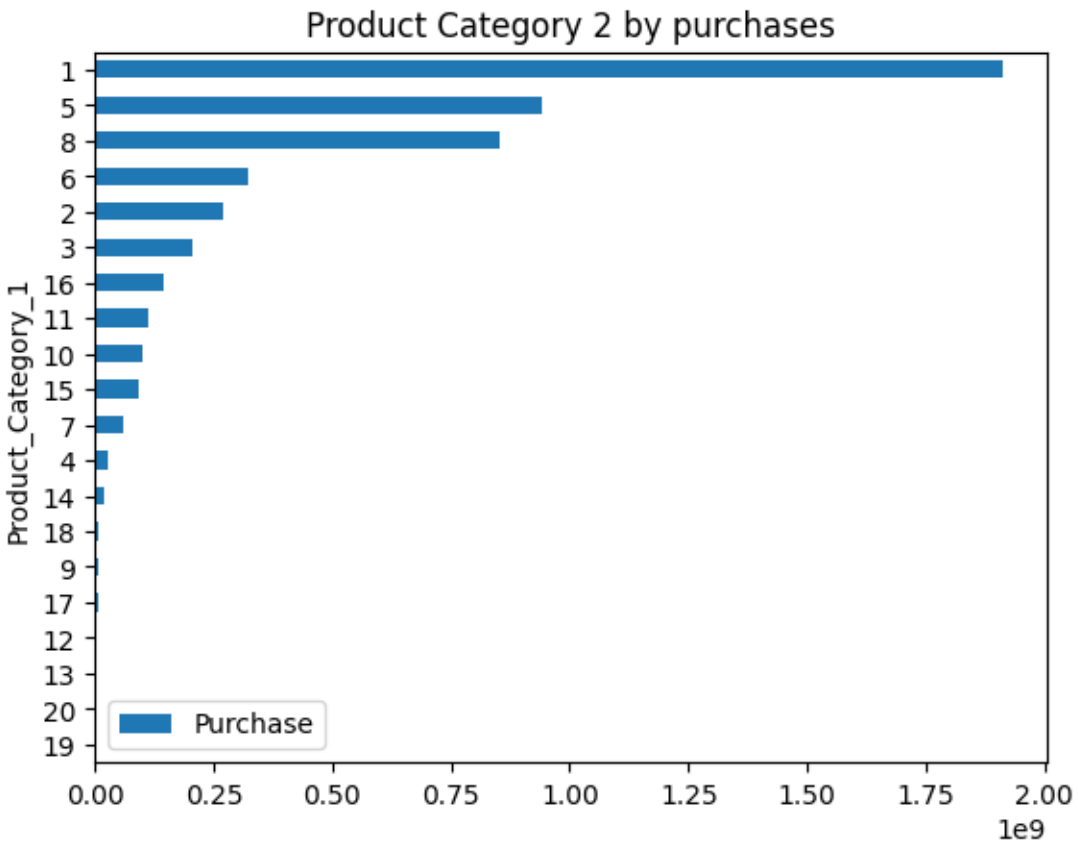

[39]:
```
prod_pu1=df[['Product_Category_1','Purchase']].groupby('Product_Category_1').
  ↪sum()
prod_pu1=prod_pu1.sort_values(by='Purchase')
prod_pu1.plot(kind='barh',title='Product Category 1 by purchases')
```

[39]:
```
<Axes: title={'center': 'Product Category 1 by purchases'},
ylabel='Product_Category_1'>
```



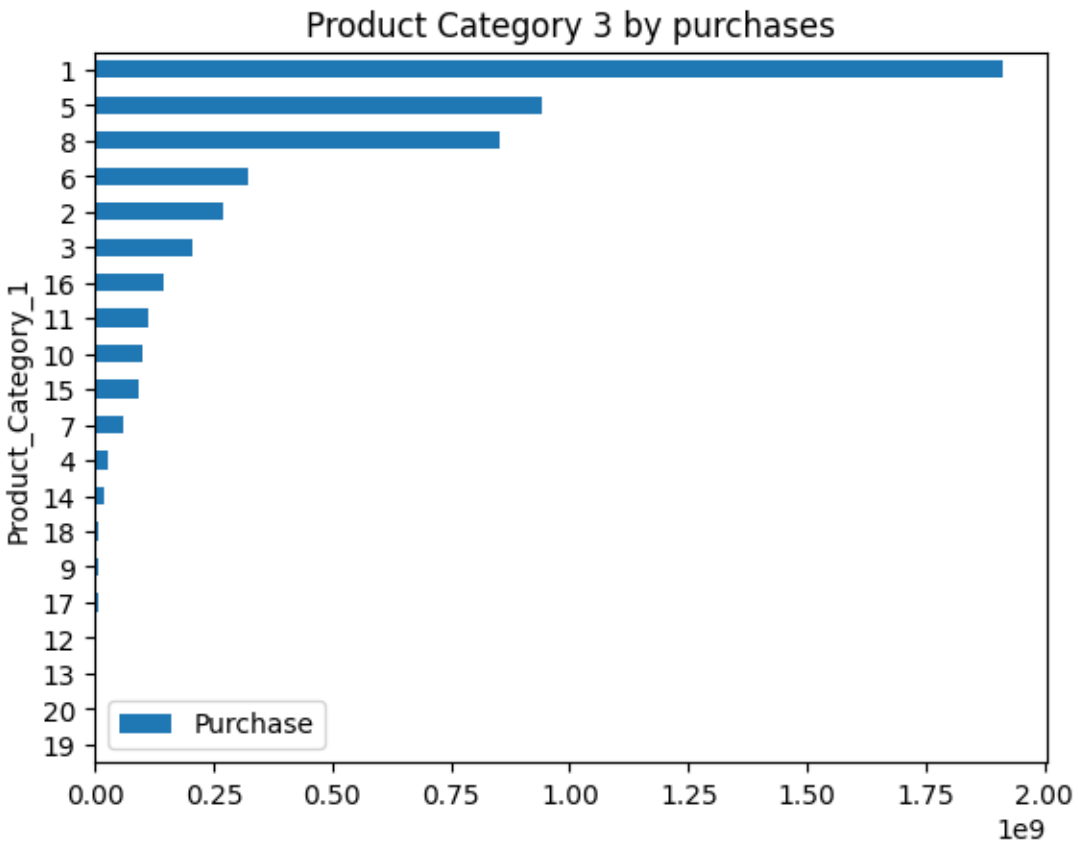

[40]:
```
prod_pu2=df[['Product_Category_2','Purchase']].groupby('Product_Category_2').
  ↪sum()
prod_pu2=prod_pu1.sort_values(by='Purchase')
prod_pu2.plot(kind='barh',title='Product Category 2 by purchases')
```

[40]:
```
<Axes: title={'center': 'Product Category 2 by purchases'},
ylabel='Product_Category_1'>
```



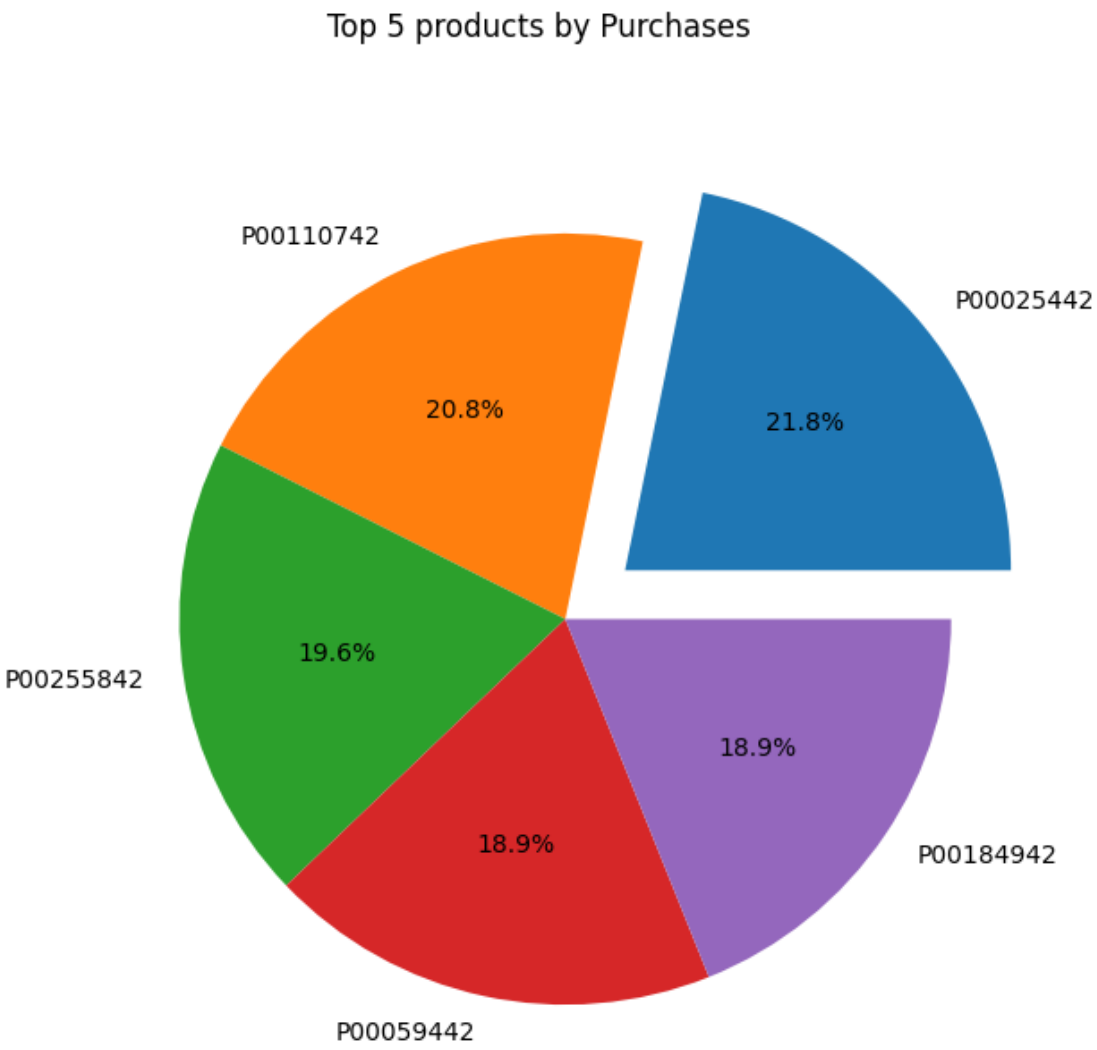

[41]:
```
prod_pu3=df[['Product_Category_3','Purchase']].groupby('Product_Category_3').
  ↪sum()
prod_pu3=prod_pu1.sort_values(by='Purchase')
prod_pu3.plot(kind='barh',title='Product Category 3 by purchases')
```

[41]:
```
<Axes: title={'center': 'Product Category 3 by purchases'},
ylabel='Product_Category_1'>
```




[47]:
```
top_prod=df[['Product_ID','Purchase']].groupby('Product_ID').sum()
top_prod=top_prod.sort_values(by='Purchase',ascending=False)
top_prod=top_prod.head(5)
top_prod.plot(kind='pie',title='Top 5 products by␣
 ↪Purchases',subplots=True,autopct='%1.
 ↪1f%%',legend=False,ylabel='',figsize=(12,7), explode=[0.2,0,0,0,0])
```

[47]:
```
array([<Axes: >], dtype=object)
```




[50]:
```
df.corr()
```

```
<ipython-input-50-2f6f6606aa2c>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df.corr()
```

[50]:

| | User_ID | Occupation | Marital_Status | Product_Category_1 | \ |
|---|---|---|---|---|---|
| User_ID | 1.000000 | -0.023971 | 0.020443 | 0.003825 | |
| Occupation | -0.023971 | 1.000000 | 0.024280 | -0.007618 | |
| Marital_Status | 0.020443 | 0.024280 | 1.000000 | 0.019888 | |
| Product_Category_1 | 0.003825 | -0.007618 | 0.019888 | 1.000000 | |

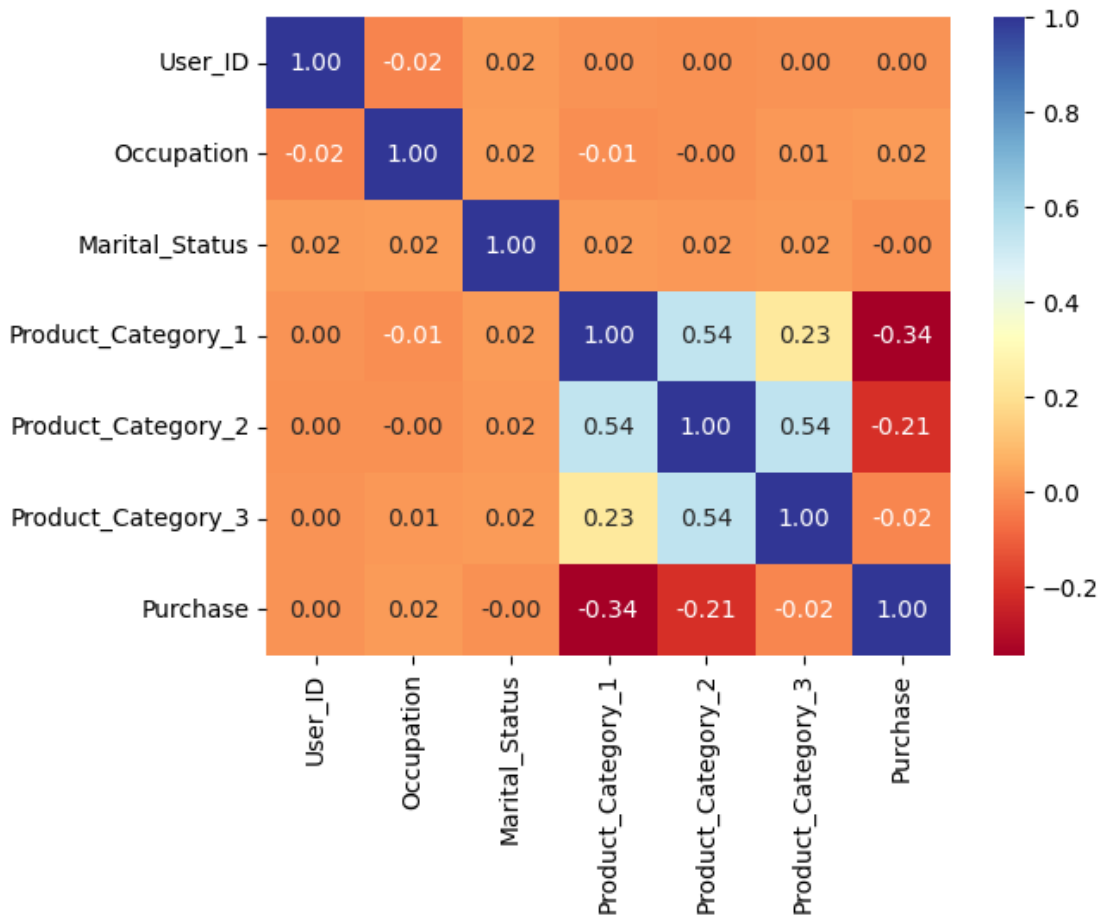
```
Product_Category_2  0.001529   -0.000384        0.015138            0.540583
Product_Category_3  0.003419    0.013263        0.019473            0.229678
Purchase            0.004716    0.020833       -0.000463           -0.343703
```

```
                   Product_Category_2  Product_Category_3  Purchase
User_ID                      0.001529            0.003419  0.004716
Occupation                  -0.000384            0.013263  0.020833
Marital_Status               0.015138            0.019473 -0.000463
Product_Category_1           0.540583            0.229678 -0.343703
Product_Category_2           1.000000            0.543649 -0.209918
Product_Category_3           0.543649            1.000000 -0.022006
Purchase                    -0.209918           -0.022006  1.000000
```

[53]:
```
sns.heatmap(df.corr(), annot = True, cmap= 'RdYlBu', fmt= '.2f')
```

```
<ipython-input-53-3048f36ecc00>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df.corr(), annot = True, cmap= 'RdYlBu', fmt= '.2f')
```

[53]:
```
<Axes: >
```




[67]:
```
df2=df.copy()
df2['Gender']=pd.factorize(df2.Gender)[0]
df2['Age']=pd.factorize(df2.Age)[0]
df2['City_Category']=pd.factorize(df2.City_Category)[0]
df2['Stay_In_Current_City_Years']=pd.factorize(df2.
  ↪Stay_In_Current_City_Years)[0]
df2['Product_ID']=pd.factorize(df2.Product_ID)[0]
df2['User_ID']=pd.factorize(df2.User_ID)[0]
df2['Product_Category_1']=pd.factorize(df2.Product_Category_1)[0]
df2['Product_Category_2']=pd.factorize(df2.Product_Category_2)[0]
df2['Product_Category_3']=pd.factorize(df2.Product_Category_3)[0]
```

[68]:
```
df2.corr()
```

[68]:

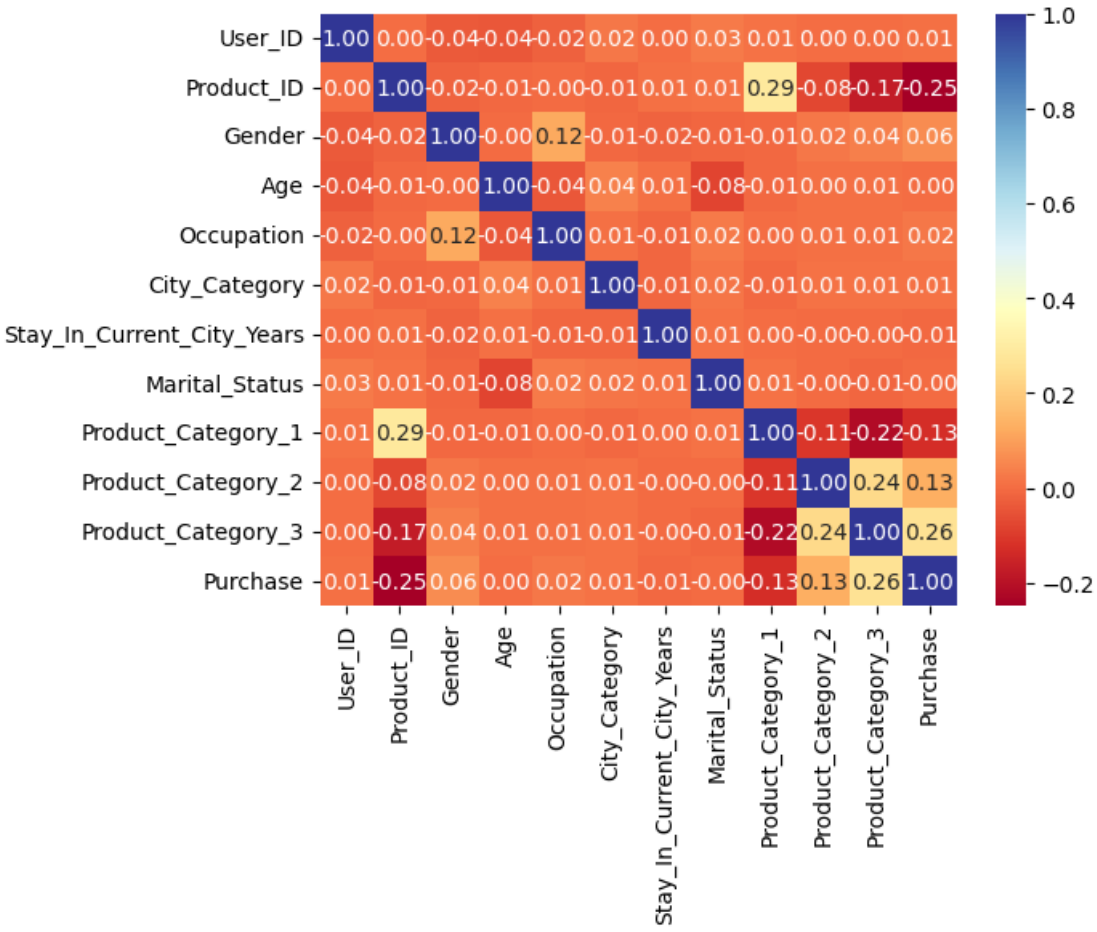
| | User_ID | Product_ID | Gender | Age | \ |
|---|---|---|---|---|---|
| User_ID | 1.000000 | 0.004389 | -0.038275 | -0.039948 | |
| Product_ID | 0.004389 | 1.000000 | -0.016938 | -0.006908 | |
| Gender | -0.038275 | -0.016938 | 1.000000 | -0.000777 | |

```
Age                         -0.039948   -0.006908 -0.000777  1.000000
Occupation                  -0.023348   -0.004188  0.117291 -0.040858
City_Category                0.023497   -0.013383 -0.007688  0.042006
Stay_In_Current_City_Years   0.003509    0.007775 -0.015108  0.007013
Marital_Status               0.025181    0.013193 -0.011603 -0.078776
Product_Category_1           0.010296    0.286762 -0.006406 -0.006687
Product_Category_2           0.002447   -0.077515  0.021938  0.004906
Product_Category_3           0.000739   -0.173713  0.037288  0.008081
Purchase                     0.006735   -0.249081  0.060346  0.002344

                            Occupation  City_Category  \
User_ID                      -0.023348       0.023497
Product_ID                   -0.004188      -0.013383
Gender                        0.117291      -0.007688
Age                          -0.040858       0.042006
Occupation                    1.000000       0.005470
City_Category                 0.005470       1.000000
Stay_In_Current_City_Years   -0.010470      -0.011205
Marital_Status                0.024280       0.016846
Product_Category_1            0.003964      -0.006594
Product_Category_2            0.008461       0.011255
Product_Category_3            0.008824       0.011226
Purchase                      0.020833       0.011513

                            Stay_In_Current_City_Years  Marital_Status  \
User_ID                                       0.003509        0.025181
Product_ID                                    0.007775        0.013193
Gender                                       -0.015108       -0.011603
Age                                           0.007013       -0.078776
Occupation                                   -0.010470        0.024280
City_Category                                -0.011205        0.016846
Stay_In_Current_City_Years                    1.000000        0.010461
Marital_Status                                0.010461        1.000000
Product_Category_1                            0.003440        0.011935
Product_Category_2                           -0.001197       -0.002574
Product_Category_3                           -0.004844       -0.010064
Purchase                                     -0.007504       -0.000463

                            Product_Category_1  Product_Category_2  \
User_ID                               0.010296            0.002447
Product_ID                            0.286762           -0.077515
Gender                               -0.006406            0.021938
Age                                  -0.006687            0.004906
Occupation                            0.003964            0.008461
City_Category                        -0.006594            0.011255
Stay_In_Current_City_Years            0.003440           -0.001197
Marital_Status                        0.011935           -0.002574
```

```
Product_Category_1                   1.000000           -0.108763
Product_Category_2                  -0.108763            1.000000
Product_Category_3                  -0.217742            0.238617
Purchase                            -0.130190            0.129176

                            Product_Category_3  Purchase
User_ID                               0.000739  0.006735
Product_ID                           -0.173713 -0.249081
Gender                                0.037288  0.060346
Age                                   0.008081  0.002344
Occupation                            0.008824  0.020833
City_Category                         0.011226  0.011513
Stay_In_Current_City_Years           -0.004844 -0.007504
Marital_Status                       -0.010064 -0.000463
Product_Category_1                   -0.217742 -0.130190
Product_Category_2                    0.238617  0.129176
Product_Category_3                    1.000000  0.262353
Purchase                              0.262353  1.000000
```

[69]:
```
sns.heatmap(df2.corr(), annot = True, cmap= 'RdYlBu', fmt= '.2f')
```

[69]:
```
<Axes: >
```




[61]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.metrics import r2_score
```

[70]:
```
X = df2.drop("Purchase", axis=1)
y = df2["Purchase"]
```

[71]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪random_state=42)
```

[72]:
```
rfr = RandomForestRegressor()
```

[74]:
```
rfr.fit(X_train, y_train)
```

[74]:
```
RandomForestRegressor()
```

[75]:
```
score = rfr.score(X_train,y_train)
score
```

[75]:
```
0.9583131091428291
```

[76]:
```
y_pred5 = rfr.predict(X_test)
rscore=r2_score(y_test, y_pred5)
rscore
```

[76]:
```
0.6968272172364111
```

[77]:
```
df2['Prediction']=rfr.predict(X)
```

[78]:
```
df2.head(5)
```

[78]:

| | User_ID | Product_ID | Gender | Age | Occupation | City_Category | \ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 10 | 0 | |
| 1 | 0 | 1 | 0 | 0 | 10 | 0 | |
| 2 | 0 | 2 | 0 | 0 | 10 | 0 | |
| 3 | 0 | 3 | 0 | 0 | 10 | 0 | |
| 4 | 1 | 4 | 1 | 1 | 16 | 1 | |

| | Stay_In_Current_City_Years | Marital_Status | Product_Category_1 | \ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 1 | |
| 2 | 0 | 0 | 2 | |
| 3 | 0 | 0 | 2 | |
| 4 | 1 | 0 | 3 | |

| | Product_Category_2 | Product_Category_3 | Purchase | Prediction |
|---|---|---|---|---|
| 0 | -1 | -1 | 8370 | 9935.52 |
| 1 | 0 | 0 | 15200 | 15817.66 |
| 2 | -1 | -1 | 1422 | 1326.38 |
| 3 | 1 | -1 | 1057 | 1310.33 |
| 4 | -1 | -1 | 7969 | 7715.25 |

```
<google.colab._quickchart_helpers.SectionTitle at 0x79773d56f280>
```

```
from matplotlib import pyplot as plt
_df_35['User_ID'].plot(kind='hist', bins=20, title='User_ID')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_36['Product_ID'].plot(kind='hist', bins=20, title='Product_ID')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_37['Gender'].plot(kind='hist', bins=20, title='Gender')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_38['Age'].plot(kind='hist', bins=20, title='Age')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x79773dd86bc0>
```

```
from matplotlib import pyplot as plt
_df_39.plot(kind='scatter', x='User_ID', y='Product_ID', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_40.plot(kind='scatter', x='Product_ID', y='Gender', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_41.plot(kind='scatter', x='Gender', y='Age', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_42.plot(kind='scatter', x='Age', y='Occupation', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x79773d14caf0>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['User_ID']
  ys = series['Product_Category_2']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_43.sort_values('User_ID', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('User_ID')
_ = plt.ylabel('Product_Category_2')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['User_ID']
  ys = series['Product_Category_3']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])
```

```
fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_44.sort_values('User_ID', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('User_ID')
_ = plt.ylabel('Product_Category_3')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['User_ID']
  ys = series['Purchase']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_45.sort_values('User_ID', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('User_ID')
_ = plt.ylabel('Purchase')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  from matplotlib import pyplot as plt
  import seaborn as sns
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['User_ID']
  ys = series['Prediction']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_46.sort_values('User_ID', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('User_ID')
_ = plt.ylabel('Prediction')
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x79773d14d7b0>
```

```
from matplotlib import pyplot as plt
_df_47['User_ID'].plot(kind='line', figsize=(8, 4), title='User_ID')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_48['Product_ID'].plot(kind='line', figsize=(8, 4), title='Product_ID')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_49['Gender'].plot(kind='line', figsize=(8, 4), title='Gender')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_50['Age'].plot(kind='line', figsize=(8, 4), title='Age')
plt.gca().spines[['top', 'right']].set_visible(False)
```