Sales data Analysis

OBJECTIVES:

- importing zipdata and creating a dataframe
- figuring out optimal data visualization formats to visualize data
- Applying Data Visualization
- Creating interactive dashboard or tables and charts to better explain dataset

```
import pandas as pd
import matplotlib.pyplot as plt  # Corrected import statement
import numpy as np
import seaborn as sns

Sales_data = pd.read_csv('/content/Sales
data/retail_sales_dataset.csv')

Sales_data.head()
```


```
{"summary":"{\n  \"name\": \"Sales_data\",\n  \"rows\": 1000,\n
\"fields\": [\n    {\n      \"column\": \"Transaction ID\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
288,\n        \"min\": 1,\n        \"max\": 1000,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n          522,\n
738,\n          741\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Date\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"num_unique_values\": 345,\n        \"samples\": [\n          \"2023-
04-04\",\n          \"2023-04-13\",\n          \"2023-04-15\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Customer ID\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
\"CUST522\",\n          \"CUST738\",\n          \"CUST741\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Gender\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          \"Female\",\n          \"Male\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 13,\n        \"min\": 18,\n
\"max\": 64,\n        \"num_unique_values\": 47,\n        \"samples\":
[\n          48,\n          61\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Product Category\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"Beauty\",\n          \"Clothing\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Quantity\",\n
```

```
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1,\n        \"min\": 1,\n        \"max\": 4,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          2,\n
4\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Price per Unit\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 189,\n        \"min\": 25,\n
\"max\": 500,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          500,\n          300\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Total Amount\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
559,\n        \"min\": 25,\n        \"max\": 2000,\n
\"num_unique_values\": 18,\n        \"samples\": [\n          150,\n
1000\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"Sales_data"}
```


Checking for null values

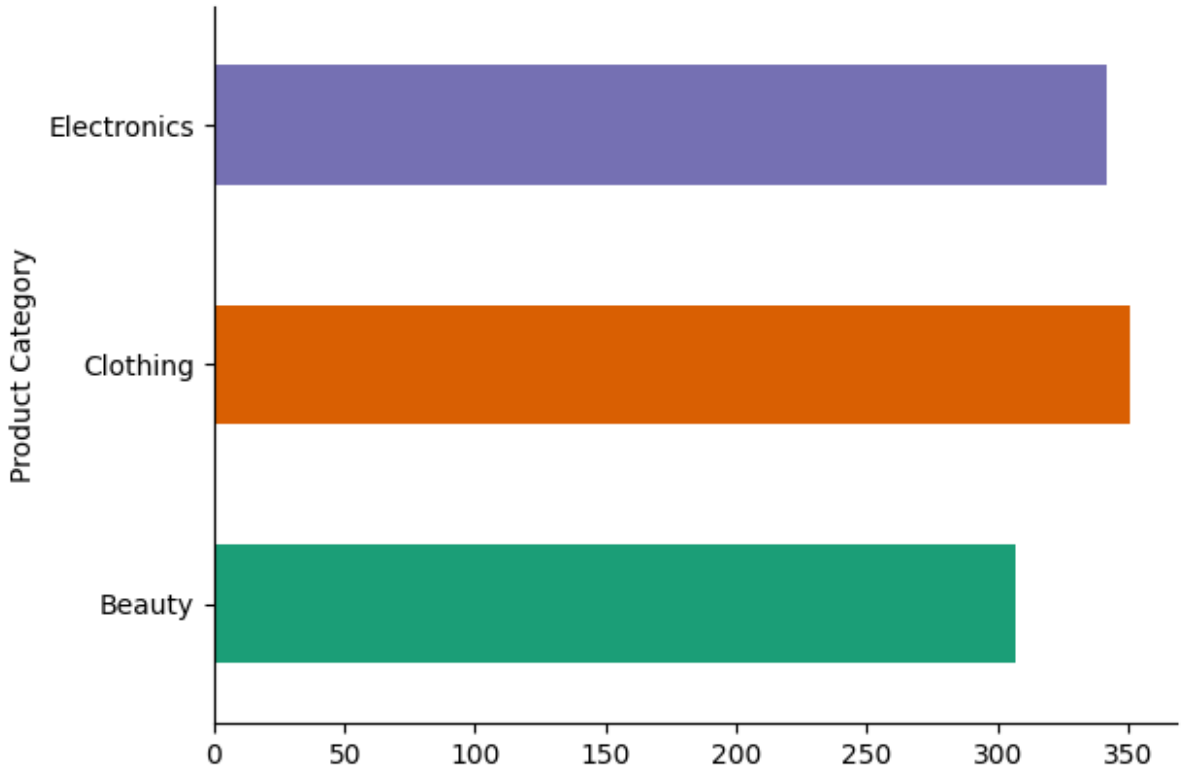
```
Sales_data.isna().sum()
```

```
Transaction ID      0
Date                0
Customer ID         0
Gender              0
Age                 0
Product Category    0
Quantity            0
Price per Unit      0
Total Amount        0
dtype: int64
```

Table has no null values so we can move ahead to data visualization

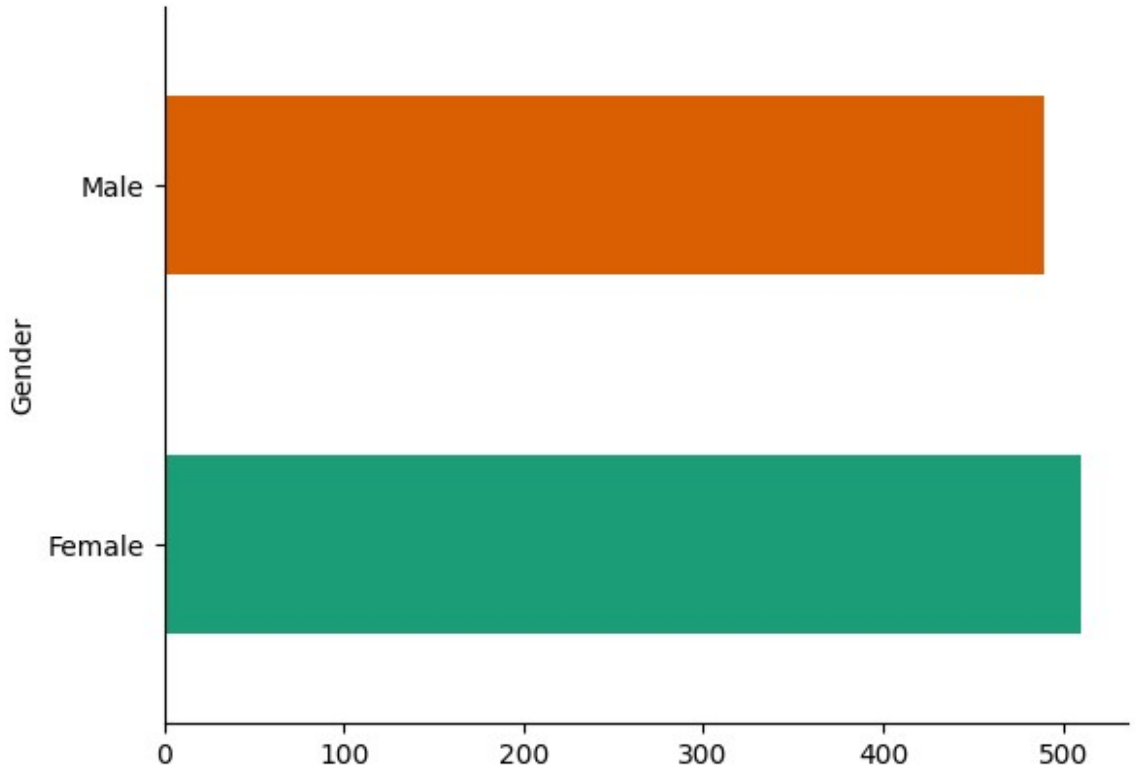
Checking what category of product sells the most

```
Sales_data.groupby('Product Category').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



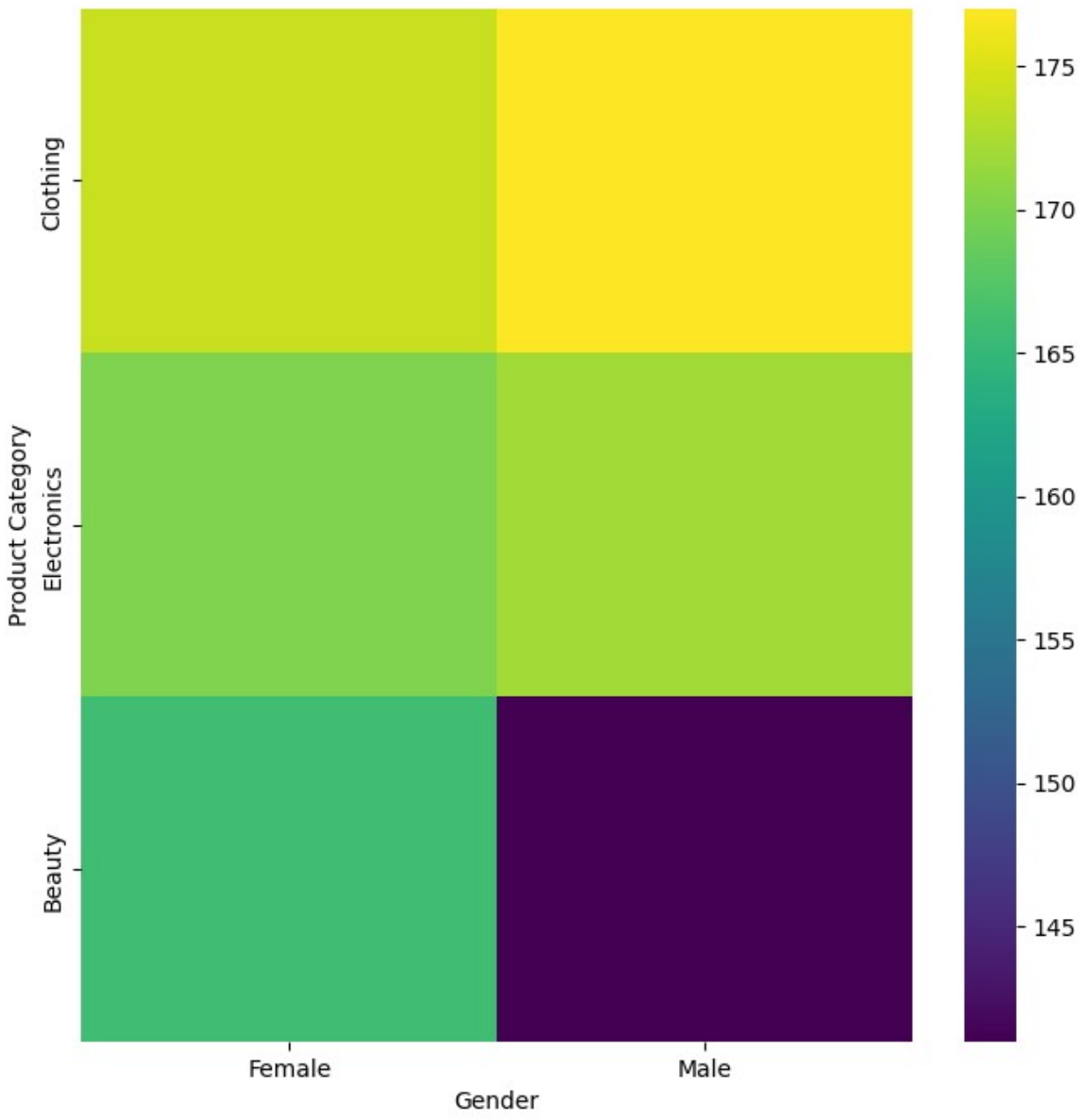
Checking Which Gender Buys the most products

```
Sales_data.groupby('Gender').size().plot(kind='barh',
color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



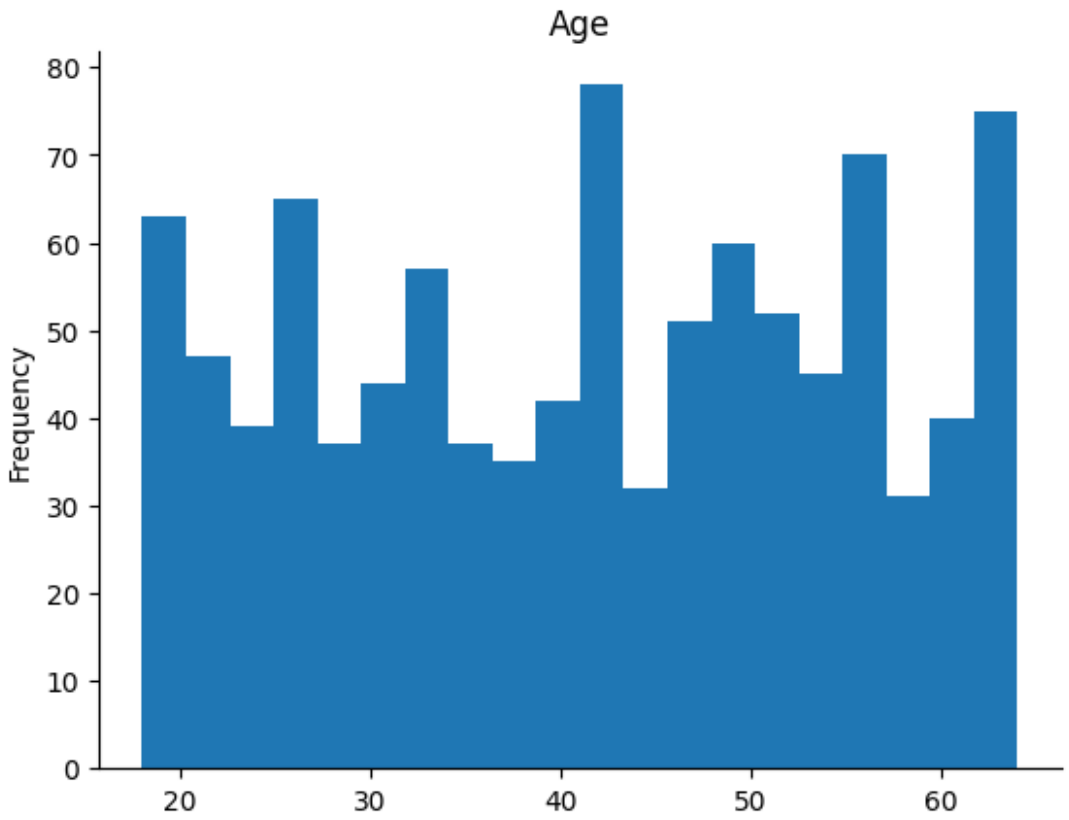
Mapping Sales product category for each Gender

```
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['Product Category'].value_counts()
    for x_label, grp in Sales_data.groupby('Gender')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('Gender')
_ = plt.ylabel('Product Category')
```

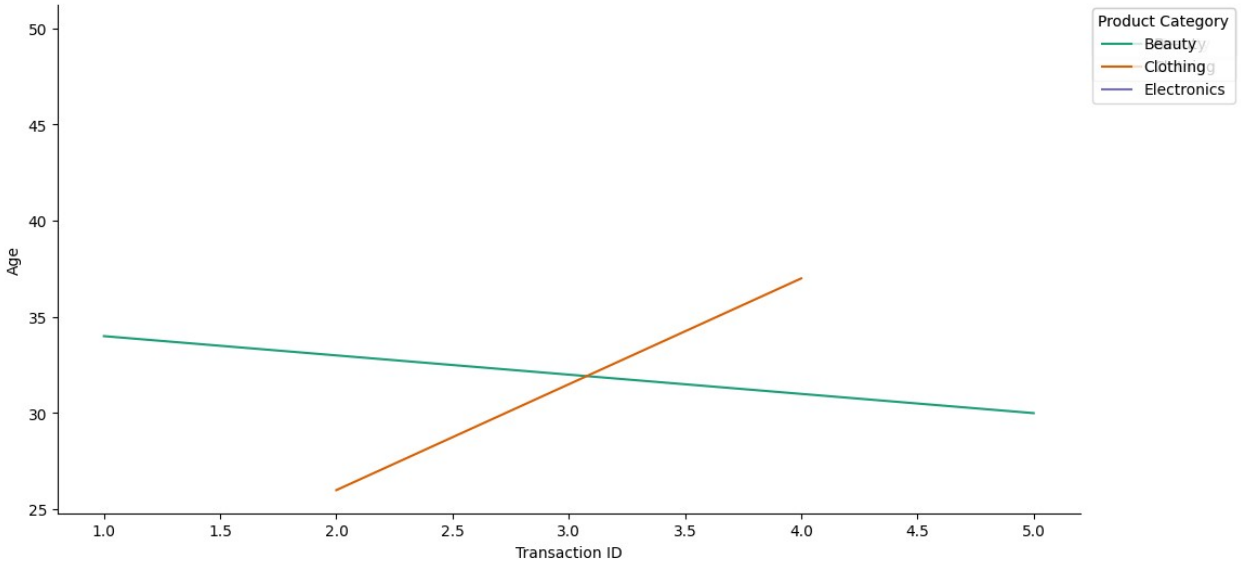


Age vs Buying Frequency

```
Sales_data['Age'].plot(kind='hist', bins=20, title='Age')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

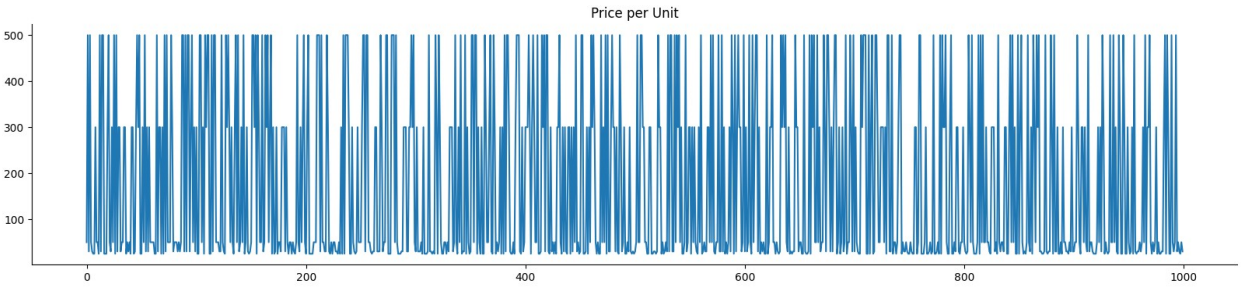



```
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['Transaction ID']
  ys = series['Age']

  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_42.sort_values('Transaction ID', ascending=True)
for i, (series_name, series) in enumerate(df_sorted.groupby('Product
Category')):
  _plot_series(series, series_name, i)
  fig.legend(title='Product Category', bbox_to_anchor=(1, 1),
loc='upper left')
sns.despine(fig=fig, ax=ax)
plt.xlabel('Transaction ID')
_ = plt.ylabel('Age')
```



Price per Unit of Products

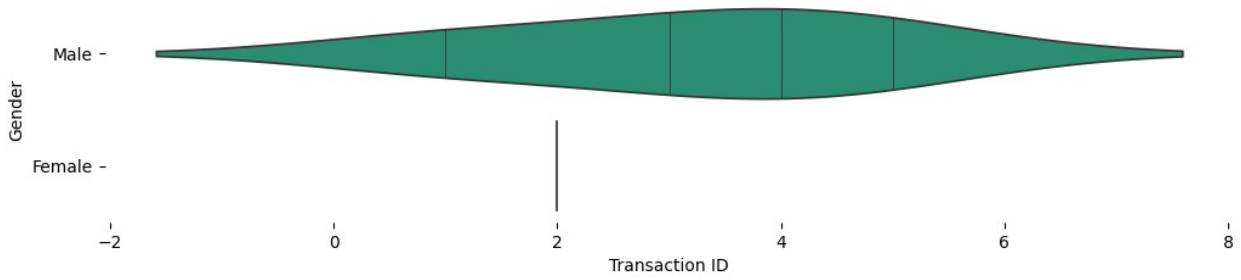
```
Sales_data['Price per Unit'].plot(kind='line', figsize=(20, 4),
title='Price per Unit')
plt.gca().spines[['top', 'right']].set_visible(False)
```



Gender Vs Transaction ID

```
figsize = (12, 1.2 * len(_df_52['Gender'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_52, x='Transaction ID', y='Gender', inner='stick',
palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

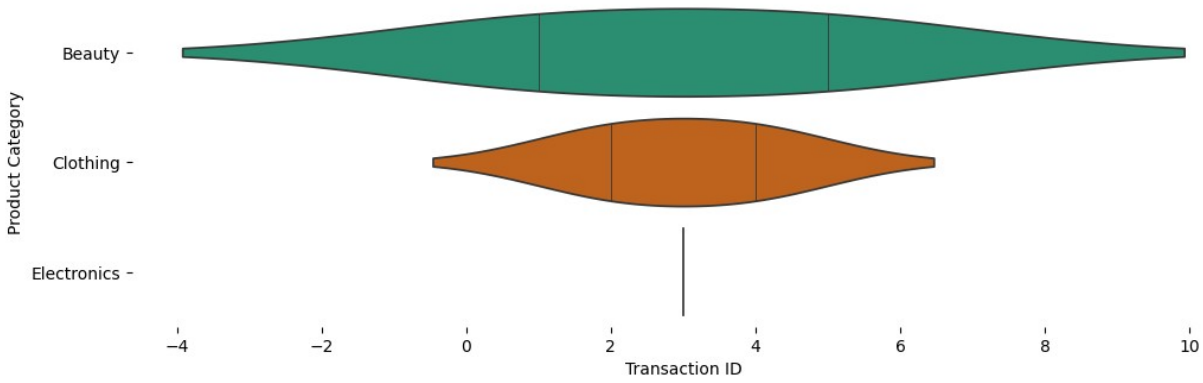
```
<ipython-input-155-25025927d693>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.violinplot(_df_52, x='Transaction ID', y='Gender',
inner='stick', palette='Dark2')
```



Transaction ID vs Product Category

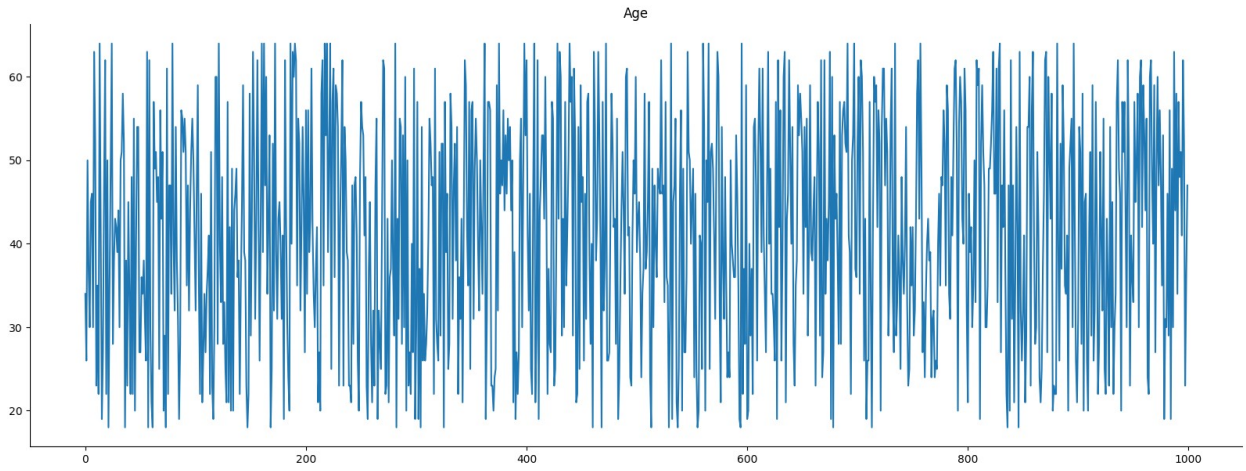
```
figsize = (12, 1.2 * len(_df_53['Product Category'].unique()))
plt.figure(figsize=figsize)
sns.violinplot(_df_53, x='Transaction ID', y='Product Category',
inner='stick', palette='Dark2')
sns.despine(top=True, right=True, bottom=True, left=True)
```

```
<ipython-input-156-89f27e2f743b>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.violinplot(_df_53, x='Transaction ID', y='Product Category',
inner='stick', palette='Dark2')
```



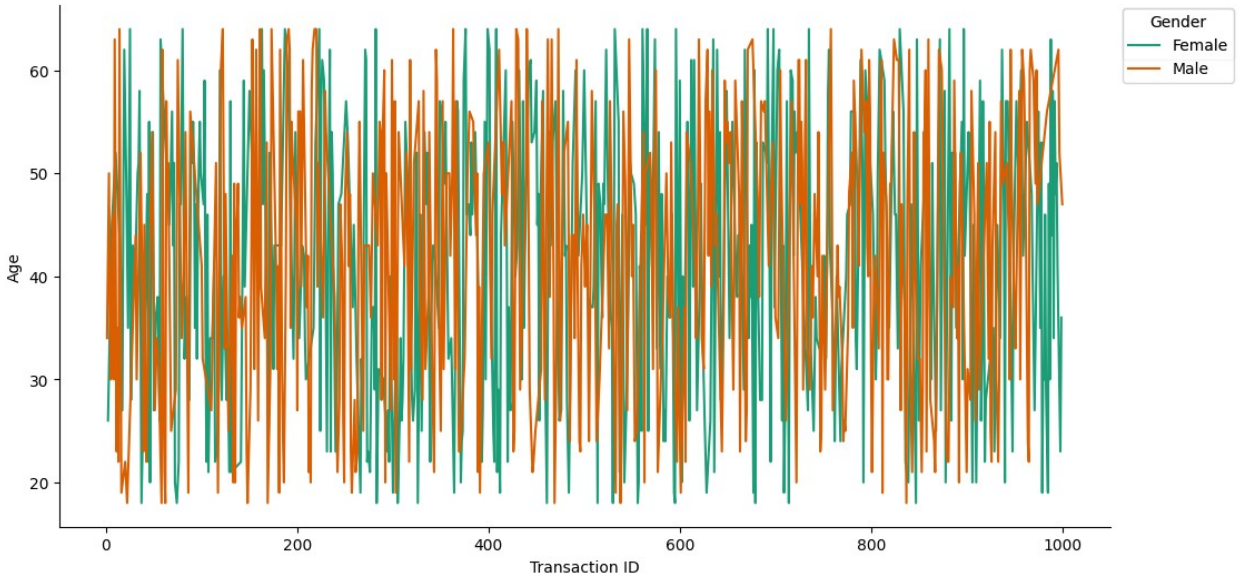
Age vs Quantity Plot

```
from matplotlib import pyplot as plt
Sales_data['Age'].plot(kind='line', figsize=(20, 7), title='Age')
plt.gca().spines[['top', 'right']].set_visible(False)
```



TRANSACTION ID VS AGE

```
# @title Transaction ID vs Age

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

  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 = Sales_data.sort_values('Transaction ID', ascending=True)
for i, (series_name, series) in
enumerate(df_sorted.groupby('Gender')):
  _plot_series(series, series_name, i)
  fig.legend(title='Gender', bbox_to_anchor=(1, 1), loc='upper left')
sns.despine(fig=fig, ax=ax)
plt.xlabel('Transaction ID')
_ = plt.ylabel('Age')
```


Gender
Female
Male
Age
60
50
40
30
20
0
200
400
600
800
1000
Transaction ID