Global_Electronic_Retailer_Data_analysis

May 16, 2024

[ ]:
```
#importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:
```
#importing customer dataset
customer = pd.read_csv("/content/drive/MyDrive/Computers/Python/Datasets for␣
 ↪data analysis/Global Electronic Sales/Customers.csv", encoding="latin-1")
#There is an encoding error, so we used a different encoding
```

[ ]:
```
customer.info()
#Getting the info about the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15266 entries, 0 to 15265
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   CustomerKey  15266 non-null  int64
 1   Gender       15266 non-null  object
 2   Name         15266 non-null  object
 3   City         15266 non-null  object
 4   State Code   15256 non-null  object
 5   State        15266 non-null  object
 6   Zip Code     15266 non-null  object
 7   Country      15266 non-null  object
 8   Continent    15266 non-null  object
 9   Birthday     15266 non-null  object
dtypes: int64(1), object(9)
memory usage: 1.2+ MB
```

[ ]:
```
customer.shape
#(rows,columns)
```

[ ]:
```
(15266, 10)
```

[ ]:
```
customer.head()
```

[ ]:
```
   CustomerKey  Gender               Name            City State Code  \
0          301  Female      Lilly Harding  WANDEARAH EAST         SA
1          325  Female       Madison Hull      MOUNT BUDD         WA
2          554  Female      Claire Ferres       WINJALLOK        VIC
3          786    Male  Jai Poltpalingada    MIDDLE RIVER         SA
4         1042    Male    Aidan Pankhurst   TAWONGA SOUTH        VIC

                State Zip Code    Country  Continent    Birthday
0    South Australia     5523  Australia  Australia    7/3/1939
1  Western Australia     6522  Australia  Australia   9/27/1979
2           Victoria     3380  Australia  Australia   5/26/1947
3    South Australia     5223  Australia  Australia   9/17/1957
4           Victoria     3698  Australia  Australia  11/19/1965
```

[ ]:
```
#Checking and removing duplicates
customer.duplicated().sum()
```

[ ]:
```
0
```

[ ]:
```
#Checking for null values
customer.isnull().sum()
#There are 10 null values in state code
```

[ ]:
```
CustomerKey     0
Gender          0
Name            0
City            0
State Code     10
State           0
Zip Code        0
Country         0
Continent       0
Birthday        0
dtype: int64
```

[ ]:
```
#Calculating the percentage of null values
(customer.isnull().sum()/len(customer))*100
#The null percentage is less that 1%, so we can leave it as it is.
```

[ ]:
```
CustomerKey    0.000000
Gender         0.000000
Name           0.000000
City           0.000000
State Code     0.065505
State          0.000000
Zip Code       0.000000
Country        0.000000
```

```
Continent      0.000000
Birthday       0.000000
dtype: float64
```

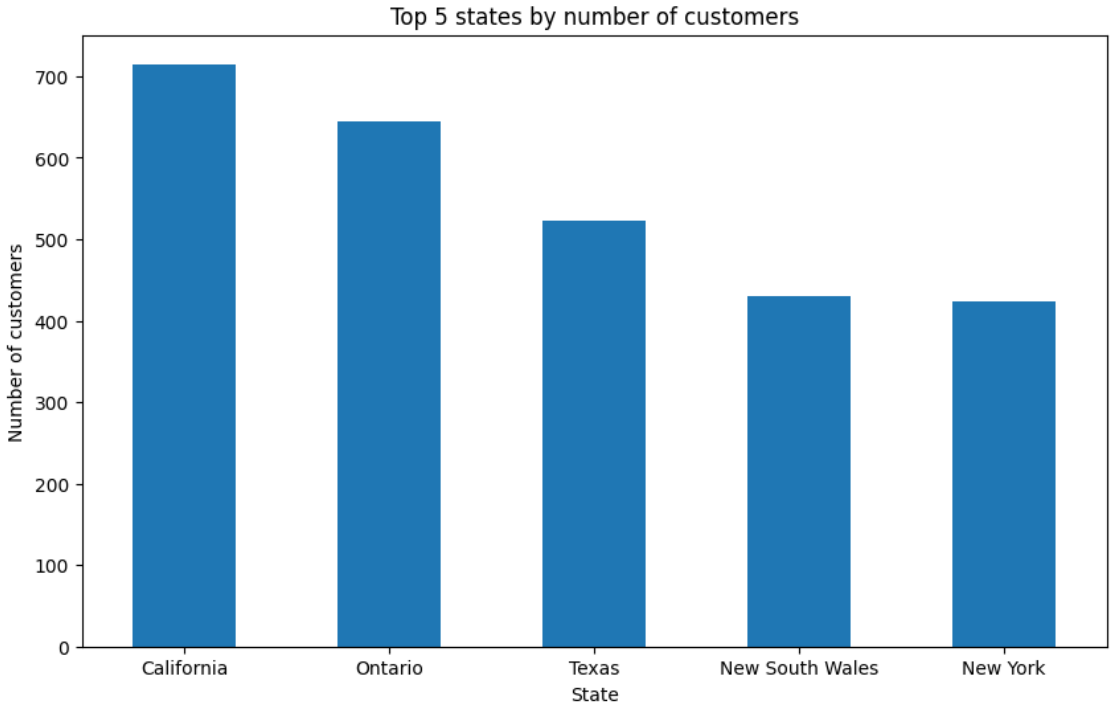
[ ]:
```
customer.describe(include = 'all').round(2)
```

[ ]:
```
        CustomerKey Gender        Name     City State Code       State  \
count      15266.00  15266       15266    15266      15256       15266
unique          NaN      2       15118     8258        467         512
top             NaN   Male  John Smith  Toronto         CA  California
freq            NaN   7748           4      204        740         715
mean     1060508.14    NaN         NaN      NaN        NaN         NaN
std       612709.69    NaN         NaN      NaN        NaN         NaN
min          301.00    NaN         NaN      NaN        NaN         NaN
25%       514033.50    NaN         NaN      NaN        NaN         NaN
50%      1079244.50    NaN         NaN      NaN        NaN         NaN
75%      1593979.50    NaN         NaN      NaN        NaN         NaN
max      2099937.00    NaN         NaN      NaN        NaN         NaN

       Zip Code        Country      Continent   Birthday
count     15266          15266          15266      15266
unique     9505              8              3      11270
top       90017  United States  North America  6/12/1989
freq         70           6828           8381          5
mean        NaN            NaN            NaN        NaN
std         NaN            NaN            NaN        NaN
min         NaN            NaN            NaN        NaN
25%         NaN            NaN            NaN        NaN
50%         NaN            NaN            NaN        NaN
75%         NaN            NaN            NaN        NaN
max         NaN            NaN            NaN        NaN
```

[ ]:
```
#Getting the count of the customers per State (top 5)
top_5_states = customer.groupby(by = 'State')['State'].count().
 ↪sort_values(ascending = False).head(5)
#California has the most number of customers, followed by Ontario and Texas
print(top_5_states)
```

```
State
California         715
Ontario            644
Texas              522
New South Wales    430
New York           423
Name: State, dtype: int64
```

[ ]:
```
#Plotting the top 5 states
top_5_states.plot(kind = 'bar',figsize = (10,6))

#Labels
plt.title('Top 5 states by number of customers')
plt.xlabel('State')
plt.ylabel('Number of customers')

plt.xticks(rotation = 0)

plt.show()
```

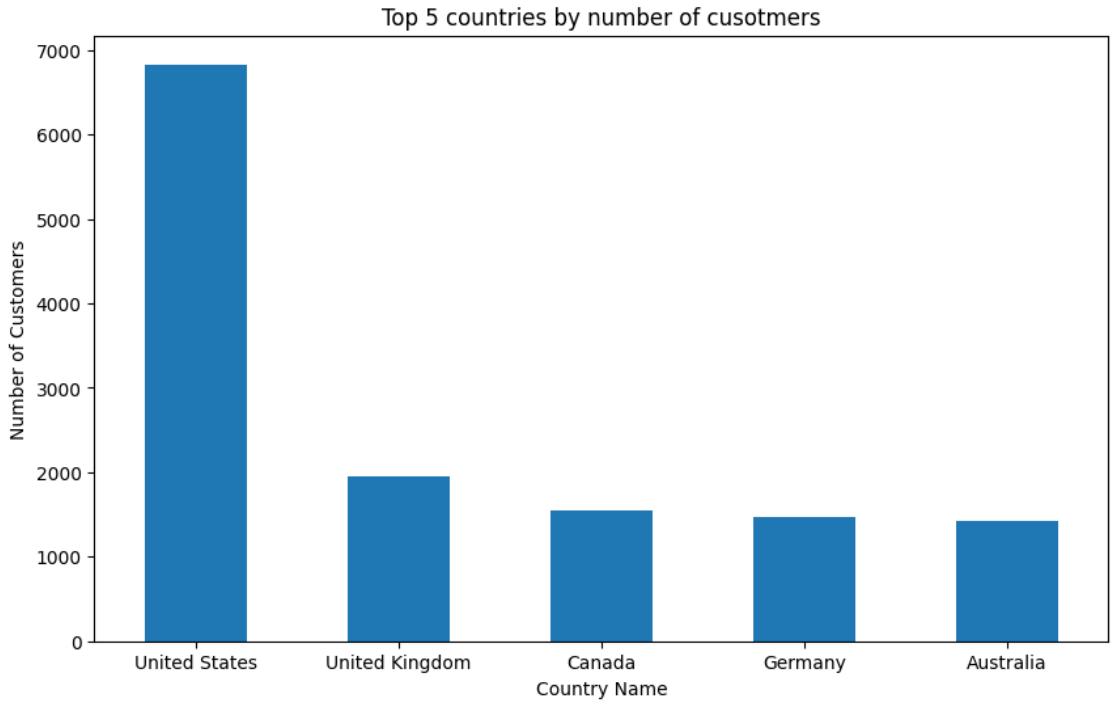


[ ]:
```
#Getting the top 5 countries by customers
top_5_countries = customer.groupby(by = 'Country')['Country'].count().
 ↪sort_values(ascending = False).head(5)
#USA has the most customers followed by UK and Canada
print(top_5_countries)
```

```
Country
United States     6828
United Kingdom    1944
Canada            1553
Germany           1473
Australia         1420
```

```
Name: Country, dtype: int64
```

[ ]:
```
#Plotting the top 5 countries
top_5_countries.plot(kind = 'bar', figsize = (10,6), rot = 0)

#lables
plt.title('Top 5 countries by number of cusotmers')
plt.xlabel('Country Name')
plt.ylabel('Number of Customers')

plt.show()
```

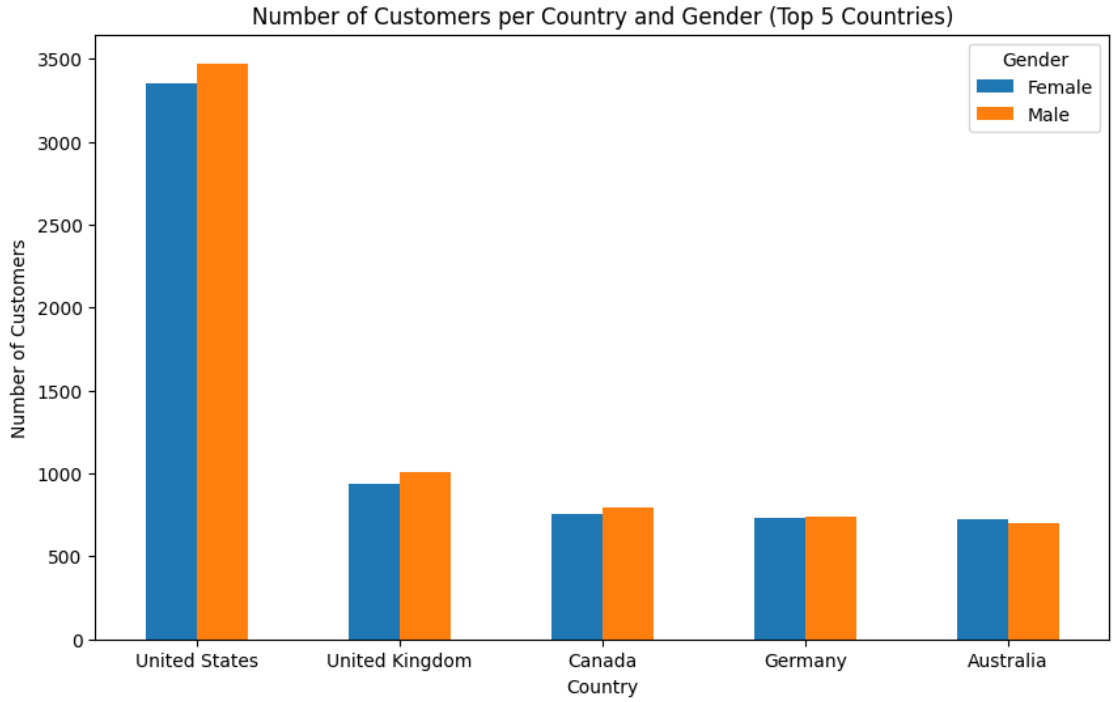


[ ]:
```
#Number of customers based on Gender
customer.groupby(by='Gender')['Gender'].count()
#There are more Male customers than female customers
```

[ ]:
```
Gender
Female    7518
Male      7748
Name: Gender, dtype: int64
```

[ ]:
```
#Number of customers based on country (Top 5 countrieds)
top_5_countries_gender = customer.groupby(['Country','Gender'])['Gender'].
↪count().sort_values(ascending = False).head(10)
```

[ ]:
```
top_5_countries = top_5_countries_gender.unstack()
# Create a bar chart
ax = top_5_countries.plot(kind='bar', figsize=(10, 6), rot=0)

# Add title and axis labels
plt.title("Number of Customers per Country and Gender (Top 5 Countries)")
plt.xlabel("Country")
plt.ylabel("Number of Customers")

# Add legend
plt.legend(title="Gender")

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

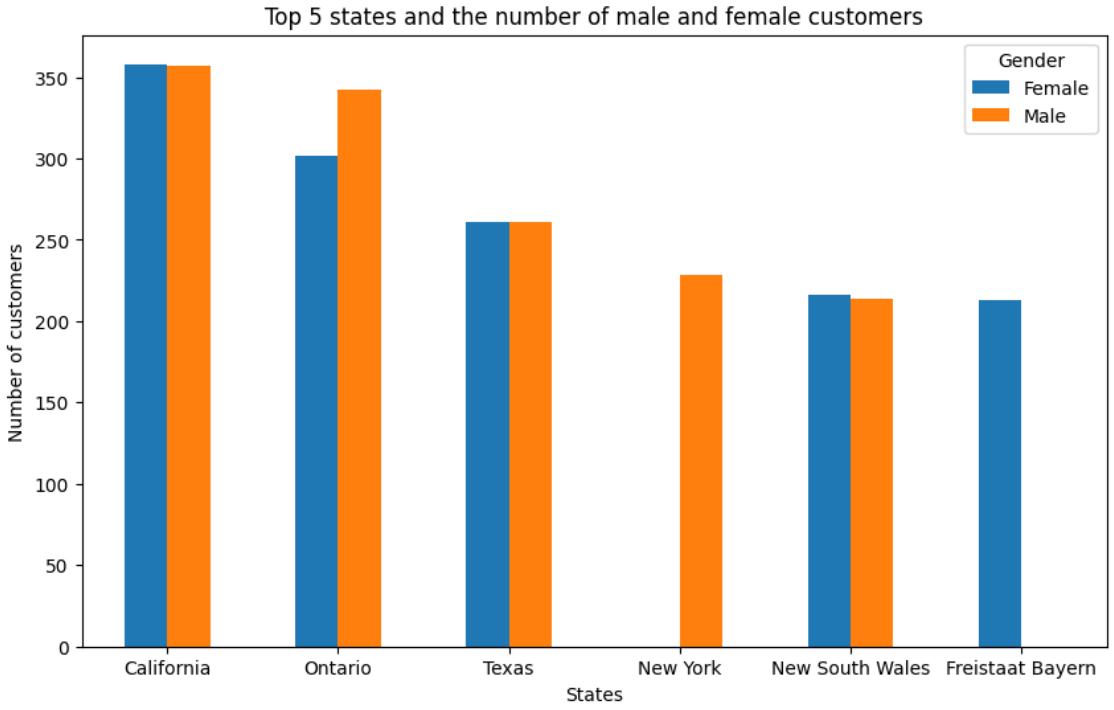


[ ]:
```
#Gender divide in the top 5 states
top_5_states_gender = customer.groupby(['State','Gender'])['Gender'].count().
  ↪sort_values(ascending = False).head(10)
print(top_5_states_gender)
```

```
State             Gender
California        Female    358
                  Male      357
Ontario           Male      342
```

```
                  Female    302
Texas             Male      261
                  Female    261
New York          Male      228
New South Wales   Female    216
                  Male      214
Freistaat Bayern  Female    213
Name: Gender, dtype: int64
```

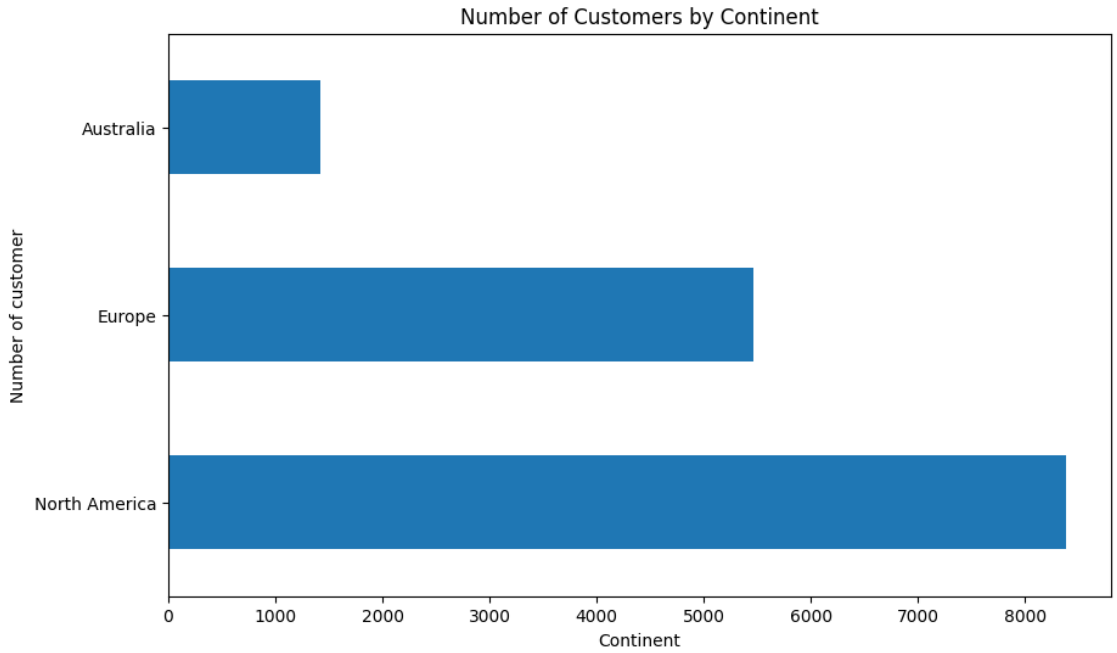
[ ]:
```
#Plotting the graph for above
top_5_states_by_gender = top_5_states_gender.unstack()
top_5_states_by_gender.plot(kind = 'bar',figsize=(10,6),rot = 0)

#Labels
plt.title("Top 5 states and the number of male and female customers")
plt.xlabel('States')
plt.ylabel('Number of customers')

plt.show()
```



[ ]:
```
#Continent wise customer data
customer_continent = customer.groupby(by = 'Continent')['Continent'].count().
  ↪sort_values(ascending = False)
print(customer_continent)
```

```
Continent
North America    8381
Europe           5465
Australia        1420
Name: Continent, dtype: int64
```

[ ]:
```
#Plotting the countries
customer_continent.plot(kind = 'barh',figsize=(10,6),rot = 0)

#Lables
plt.title('Number of Customers by Continent')
plt.xlabel('Continent')
plt.ylabel('Number of customer')

plt.show()
```

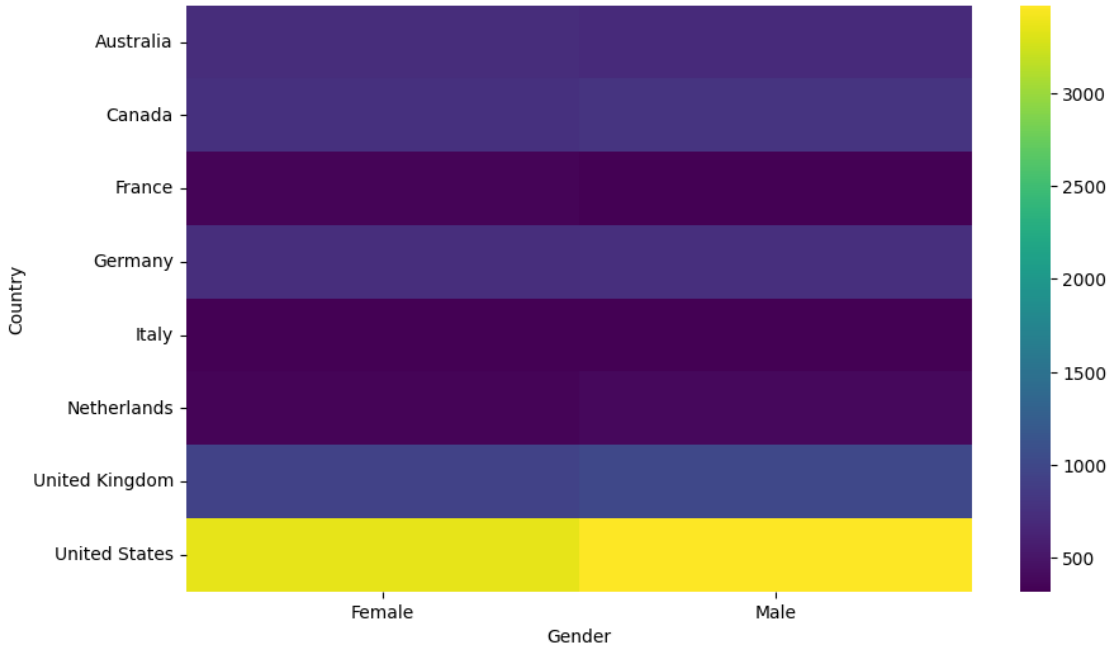


[ ]:
```
customer.head()
```

[ ]:
```
   CustomerKey  Gender               Name            City State Code  \
0          301  Female      Lilly Harding  WANDEARAH EAST         SA
1          325  Female       Madison Hull      MOUNT BUDD         WA
2          554  Female      Claire Ferres       WINJALLOK        VIC
3          786    Male  Jai Poltpalingada    MIDDLE RIVER         SA
4         1042    Male    Aidan Pankhurst   TAWONGA SOUTH        VIC

              State Zip Code    Country  Continent    Birthday
```

```
0    South Australia     5523  Australia  Australia    7/3/1939
1  Western Australia     6522  Australia  Australia   9/27/1979
2           Victoria     3380  Australia  Australia   5/26/1947
3    South Australia     5223  Australia  Australia   9/17/1957
4           Victoria     3698  Australia  Australia  11/19/1965
```

[ ]:
```
#Gender vs Country
import seaborn as sns
plt.subplots(figsize=(10, 6))
df_heatmap = pd.DataFrame({
    x_label: groupby['Country'].value_counts()
    for x_label, groupby in customer.groupby('Gender')
})
sns.heatmap(df_heatmap, cmap='viridis')
plt.xlabel('Gender')
plt.ylabel('Country')
plt.show()
```



[ ]:
```
#Importing the second dataset
products = pd.read_csv('/content/drive/MyDrive/Computers/Python/Datasets for␣
↪data analysis/Global Electronic Sales/Products.csv')
```

[ ]:
```
products.head()
```

[ ]:
```
   ProductKey                         Product Name    Brand   Color  \
0           1  Contoso 512MB MP3 Player E51 Silver  Contoso  Silver
1           2    Contoso 512MB MP3 Player E51 Blue  Contoso    Blue
2           3     Contoso 1G MP3 Player E100 White  Contoso   White
3           4    Contoso 2G MP3 Player E200 Silver  Contoso  Silver
4           5       Contoso 2G MP3 Player E200 Red  Contoso     Red

  Unit Cost USD Unit Price USD  SubcategoryKey Subcategory  CategoryKey  \
0        $6.62         $12.99              101     MP4&MP3            1
1        $6.62         $12.99              101     MP4&MP3            1
2        $7.40         $14.52              101     MP4&MP3            1
3       $11.00         $21.57              101     MP4&MP3            1
4       $11.00         $21.57              101     MP4&MP3            1

  Category
0    Audio
1    Audio
2    Audio
3    Audio
4    Audio
```

Data Cleaning

[ ]:
```
#Getting the info on the dataset
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2517 entries, 0 to 2516
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ProductKey      2517 non-null   int64
 1   Product Name    2517 non-null   object
 2   Brand           2517 non-null   object
 3   Color           2517 non-null   object
 4   Unit Cost USD   2517 non-null   object
 5   Unit Price USD  2517 non-null   object
 6   SubcategoryKey  2517 non-null   int64
 7   Subcategory     2517 non-null   object
 8   CategoryKey     2517 non-null   int64
 9   Category        2517 non-null   object
dtypes: int64(3), object(7)
memory usage: 196.8+ KB
```

[ ]:
```
#Checking if there are any duplicates
products.duplicated().sum()
#There are no duplicates
```

[ ]: 0

[ ]:
```
#Checking if there are any null values
products.isnull().sum()
#There are no null values
```

[ ]:
```
ProductKey        0
Product Name      0
Brand             0
Color             0
Unit Cost USD     0
Unit Price USD    0
SubcategoryKey    0
Subcategory       0
CategoryKey       0
Category          0
dtype: int64
```

[ ]:
```
#Getting the shape of the dataset
products.shape
#There are 2517 rows and 10 columns
```

[ ]:
```
(2517, 10)
```

[ ]:
```
#Gettings the statistical data about the dataset
products.describe(include = 'all')
#There are 8 categories and 32 sub categories
```

[ ]:
```
          ProductKey                        Product Name    Brand  Color  \
count   2517.000000                                2517     2517   2517
unique          NaN                                2517       11     16
top             NaN  Contoso 512MB MP3 Player E51 Silver  Contoso  Black
freq            NaN                                   1      710    602
mean    1259.000000                                 NaN      NaN    NaN
std      726.739637                                 NaN      NaN    NaN
min        1.000000                                 NaN      NaN    NaN
25%      630.000000                                 NaN      NaN    NaN
50%     1259.000000                                 NaN      NaN    NaN
75%     1888.000000                                 NaN      NaN    NaN
max     2517.000000                                 NaN      NaN    NaN

       Unit Cost USD Unit Price USD  SubcategoryKey            Subcategory  \
count           2517           2517     2517.000000                   2517
unique           480            426             NaN                     32
top          $15.29         $29.99              NaN  Computers Accessories
freq             34             30              NaN                    201
mean            NaN            NaN      491.810091                    NaN
```

```
std              NaN            NaN      229.887134                   NaN
min              NaN            NaN      101.000000                   NaN
25%              NaN            NaN      305.000000                   NaN
50%              NaN            NaN      406.000000                   NaN
75%              NaN            NaN      801.000000                   NaN
max              NaN            NaN      808.000000                   NaN

        CategoryKey         Category
count   2517.000000             2517
unique          NaN                8
top             NaN  Home Appliances
freq            NaN              661
mean       4.878824              NaN
std        2.299170              NaN
min        1.000000              NaN
25%        3.000000              NaN
50%        4.000000              NaN
75%        8.000000              NaN
max        8.000000              NaN
```

[ ]:
```
#Getting the columns
products.columns
```

[ ]:
```
Index(['ProductKey', 'Product Name', 'Brand', 'Color', 'Unit Cost USD',
       'Unit Price USD', 'SubcategoryKey', 'Subcategory', 'CategoryKey',
       'Category'],
      dtype='object')
```

[ ]:
```
#Importing the sales data
sales = pd.read_csv('/content/drive/MyDrive/Computers/Python/Datasets for data␣
 ↪analysis/Global Electronic Sales/Sales.csv')
```

[ ]:
```
#Getting the info about the dataset
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62884 entries, 0 to 62883
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Order Number   62884 non-null  int64
 1   Line Item      62884 non-null  int64
 2   Order Date     62884 non-null  object
 3   Delivery Date  13165 non-null  object
 4   CustomerKey    62884 non-null  int64
 5   StoreKey       62884 non-null  int64
 6   ProductKey     62884 non-null  int64
```

```
 7   Quantity       62884 non-null  int64
 8   Currency Code  62884 non-null  object
dtypes: int64(6), object(3)
memory usage: 4.3+ MB
```

[ ]:
```
sales.shape
```

[ ]:
```
(62884, 9)
```

[ ]:
```
sales.head()
```

[ ]:
```
   Order Number  Line Item Order Date Delivery Date  CustomerKey  StoreKey  \
0        366000          1   1/1/2016           NaN       265598        10
1        366001          1   1/1/2016     1/13/2016      1269051         0
2        366001          2   1/1/2016     1/13/2016      1269051         0
3        366002          1   1/1/2016     1/12/2016       266019         0
4        366002          2   1/1/2016     1/12/2016       266019         0

   ProductKey  Quantity Currency Code
0        1304         1           CAD
1        1048         2           USD
2        2007         1           USD
3        1106         7           CAD
4         373         1           CAD
```

Data Cleaning

[ ]:
```
#Checking if there are any duplicates
sales.duplicated().sum()
#There are no duplicates
```

[ ]:
```
0
```

[ ]:
```
#Checking if there are any null values
sales.isnull().sum()
#There are null values in teh delivery date
```

[ ]:
```
Order Number         0
Line Item            0
Order Date           0
Delivery Date    49719
CustomerKey          0
StoreKey             0
ProductKey           0
Quantity             0
Currency Code        0
dtype: int64
```

[ ]:
```
#Checking the percentage of null values
percentage_null_val_in_sales = ((sales.isnull().sum()/len(sales))*100)
print(percentage_null_val_in_sales )
#79% of the delivery dates are missing. So we need to replace them with␣
 ↪something
```

```
Order Number      0.000000
Line Item         0.000000
Order Date        0.000000
Delivery Date    79.064627
CustomerKey       0.000000
StoreKey          0.000000
ProductKey        0.000000
Quantity          0.000000
Currency Code     0.000000
dtype: float64
```

[ ]:
```
#Replacig the null date values with avg of delivery time.
#But to calculate the avg of delivery dates, we need a new columns.
#The columns are not in date format. So changing them to date format
sales['Delivery Date'] = pd.to_datetime(sales['Delivery Date'])
sales['Order Date'] = pd.to_datetime(sales['Order Date'])
sales['Delivery Days'] = (sales['Delivery Date'] - sales['Order Date']).dt.days

#Taking the average of the delivery days.
avg_delivery_days = round(sales['Delivery Days'].mean(),2)
print(avg_delivery_days)

#The average number of days for delivery is 4.53, so we can round it off to 5␣
 ↪days.
#Now let us replace the missing values in deliveery date with order date + 5␣
 ↪days(since 5 days is average)
sales['Delivery Date'] = sales['Delivery Date'].fillna(sales['Order Date'] + pd.
 ↪to_timedelta(5, unit='d'))

#Dropping the Delivery days columns
sales = sales.drop(columns = 'Delivery Days')
```

```
4.53
```

[ ]:
```
sales.shape
```

[ ]:
```
(62884, 9)
```

Joining the Sales and Products table for analysis

[ ]:
```
products.columns
```

[ ]:
```
Index(['ProductKey', 'Product Name', 'Brand', 'Color', 'Unit Cost USD',
       'Unit Price USD', 'SubcategoryKey', 'Subcategory', 'CategoryKey',
       'Category'],
      dtype='object')
```

[ ]:
```
sales.columns
```

[ ]:
```
Index(['Order Number', 'Line Item', 'Order Date', 'Delivery Date',
       'CustomerKey', 'StoreKey', 'ProductKey', 'Quantity', 'Currency Code'],
      dtype='object')
```

[ ]:
```
#Joining the sales and products table
sales_product_data = pd.merge(sales, products, on='ProductKey', how='inner')
sales_product_data.head()
```

[ ]:

	Order Number	Line Item	Order Date	Delivery Date	CustomerKey	StoreKey	\
0	366000	1	2016-01-01	2016-01-06	265598	10	
1	378002	2	2016-01-13	2016-01-18	1599716	45	
2	868008	2	2017-05-17	2017-05-22	1540067	51	
3	1078002	1	2017-12-13	2017-12-18	631631	18	
4	1371000	1	2018-10-02	2018-10-07	1257599	48	

	ProductKey	Quantity	Currency Code	Product Name	\
0	1304	1	CAD	Contoso Lens Adapter M450 White	
1	1304	1	USD	Contoso Lens Adapter M450 White	
2	1304	1	USD	Contoso Lens Adapter M450 White	
3	1304	1	EUR	Contoso Lens Adapter M450 White	
4	1304	8	USD	Contoso Lens Adapter M450 White	

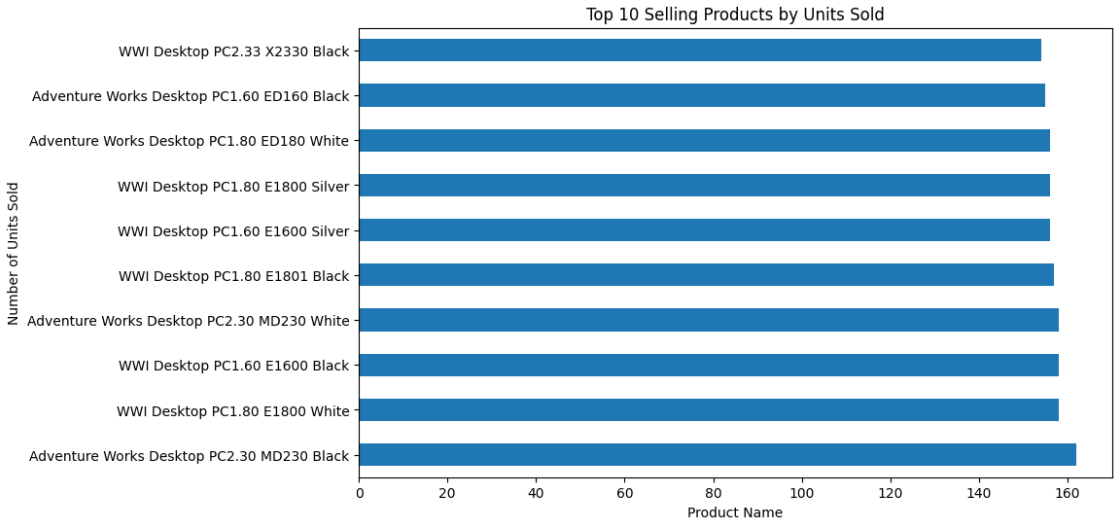
	Brand	Color	Unit Cost USD	Unit Price USD	SubcategoryKey	\
0	Contoso	White	$31.27	$68.00	406	
1	Contoso	White	$31.27	$68.00	406	
2	Contoso	White	$31.27	$68.00	406	
3	Contoso	White	$31.27	$68.00	406	
4	Contoso	White	$31.27	$68.00	406	

	Subcategory	CategoryKey	Category
0	Cameras & Camcorders	Accessories	4 Cameras and camcorders
1	Cameras & Camcorders	Accessories	4 Cameras and camcorders
2	Cameras & Camcorders	Accessories	4 Cameras and camcorders
3	Cameras & Camcorders	Accessories	4 Cameras and camcorders
4	Cameras & Camcorders	Accessories	4 Cameras and camcorders

[ ]:
```
#Top 5 selling products by number of units sold.
top_10_products = sales_product_data.groupby('Product Name')['Product Name'].
  ↪count().sort_values(ascending = False).head(10)
print(top_10_products)
```

```
Product Name
Adventure Works Desktop PC2.30 MD230 Black    162
WWI Desktop PC1.80 E1800 White                158
WWI Desktop PC1.60 E1600 Black                158
Adventure Works Desktop PC2.30 MD230 White    158
WWI Desktop PC1.80 E1801 Black                157
WWI Desktop PC1.60 E1600 Silver               156
WWI Desktop PC1.80 E1800 Silver               156
Adventure Works Desktop PC1.80 ED180 White    156
Adventure Works Desktop PC1.60 ED160 Black    155
WWI Desktop PC2.33 X2330 Black                154
Name: Product Name, dtype: int64
```

[ ]:
```
#Plotting
top_10_products.plot(kind='barh', figsize=(10, 6))

# Add title and axis labels
plt.title("Top 10 Selling Products by Units Sold")
plt.xlabel("Product Name")
plt.ylabel("Number of Units Sold")

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



[ ]:
```
sales_product_data.columns
```

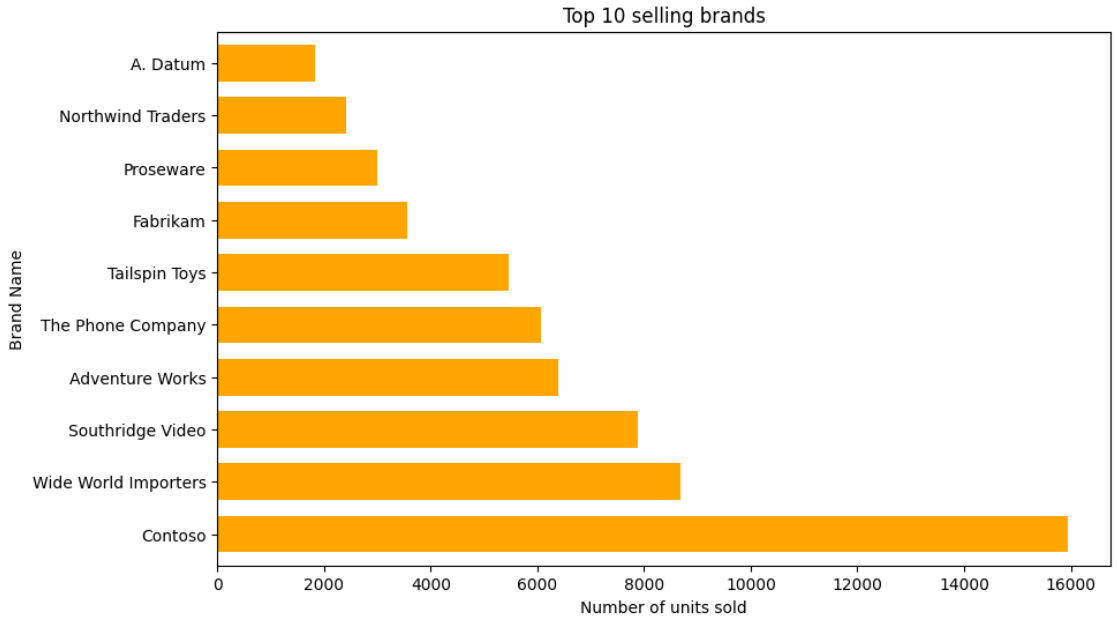
[ ]:
```
Index(['Order Number', 'Line Item', 'Order Date', 'Delivery Date',
       'CustomerKey', 'StoreKey', 'ProductKey', 'Quantity', 'Currency Code',
       'Product Name', 'Brand', 'Color', 'Unit Cost USD', 'Unit Price USD',
```

```
       'SubcategoryKey', 'Subcategory', 'CategoryKey', 'Category'],
      dtype='object')
```

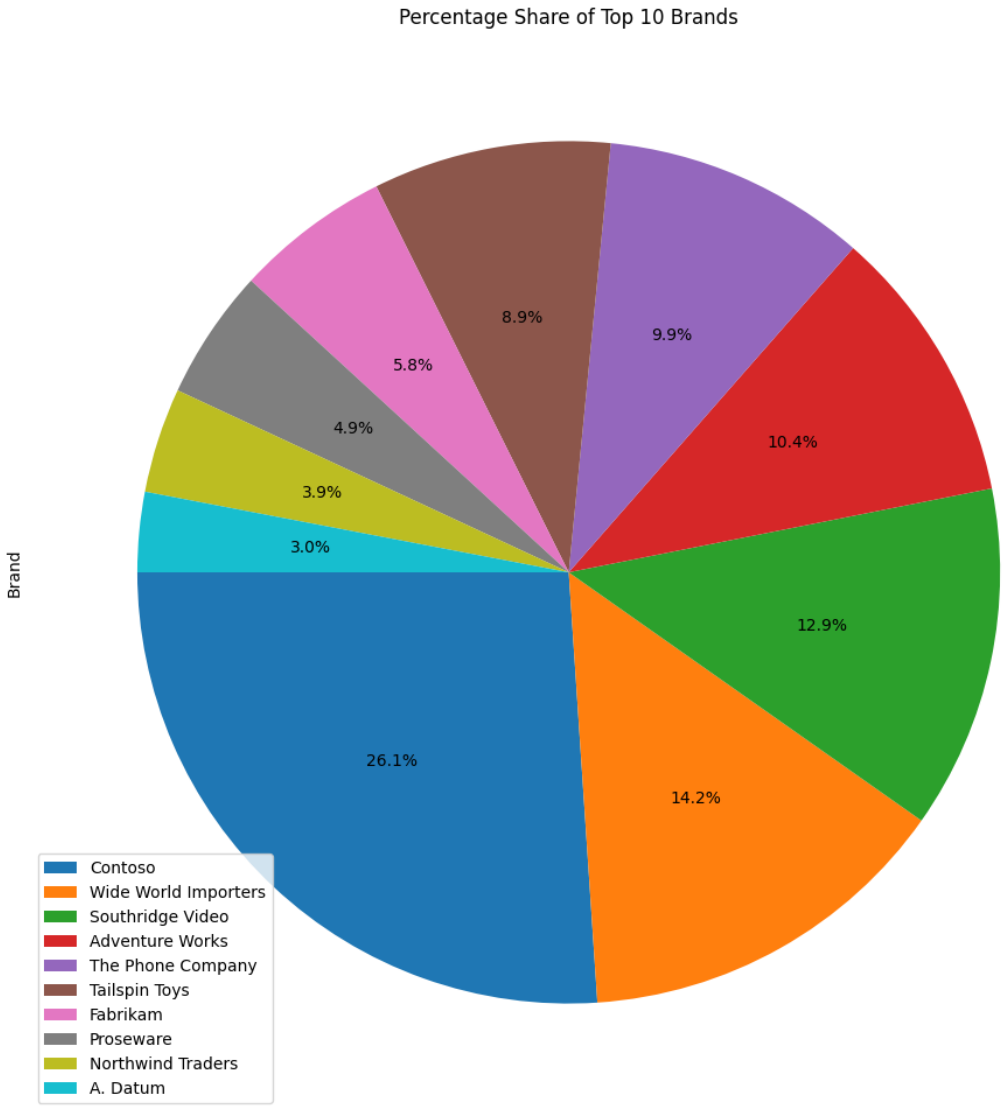
[ ]:
```
#Top 10 selling brands by number of units sold
top_10_brands = sales_product_data.groupby('Brand')['Brand'].count().
 ↪sort_values(ascending = False).head(10)
print(top_10_brands)
```

```
Brand
Contoso                 15953
Wide World Importers     8680
Southridge Video         7887
Adventure Works          6382
The Phone Company        6062
Tailspin Toys            5457
Fabrikam                 3560
Proseware                2995
Northwind Traders        2411
A. Datum                 1834
Name: Brand, dtype: int64
```

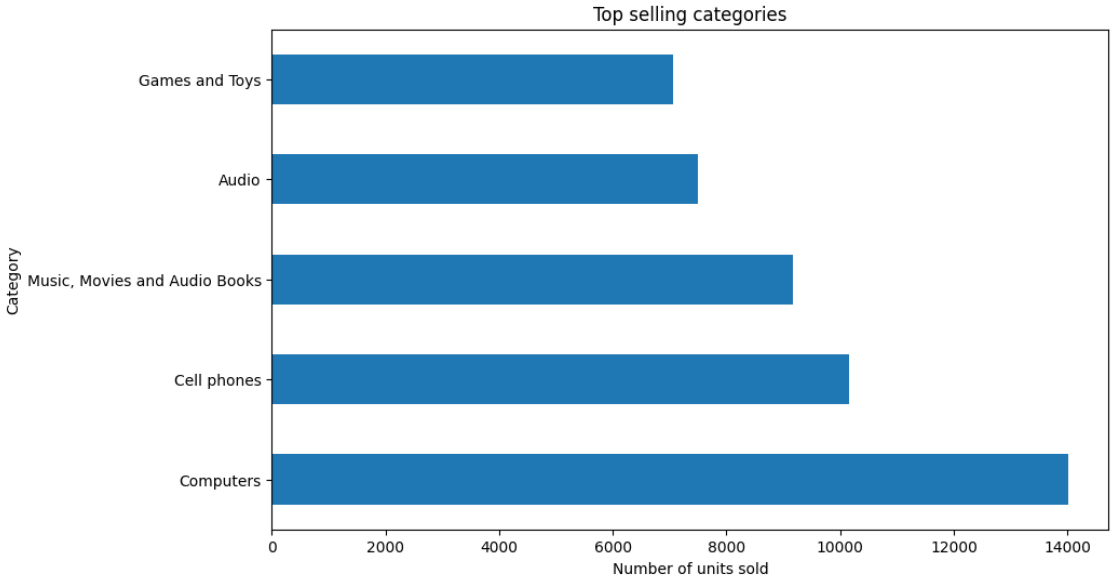
[ ]:
```
#Plotting
top_10_brands.plot(kind = 'barh',figsize = (10,6),width = 0.7, color = 'Orange')

#Labels
plt.title("Top 10 selling brands")
plt.xlabel('Number of units sold')
plt.ylabel('Brand Name')

plt.show()
```



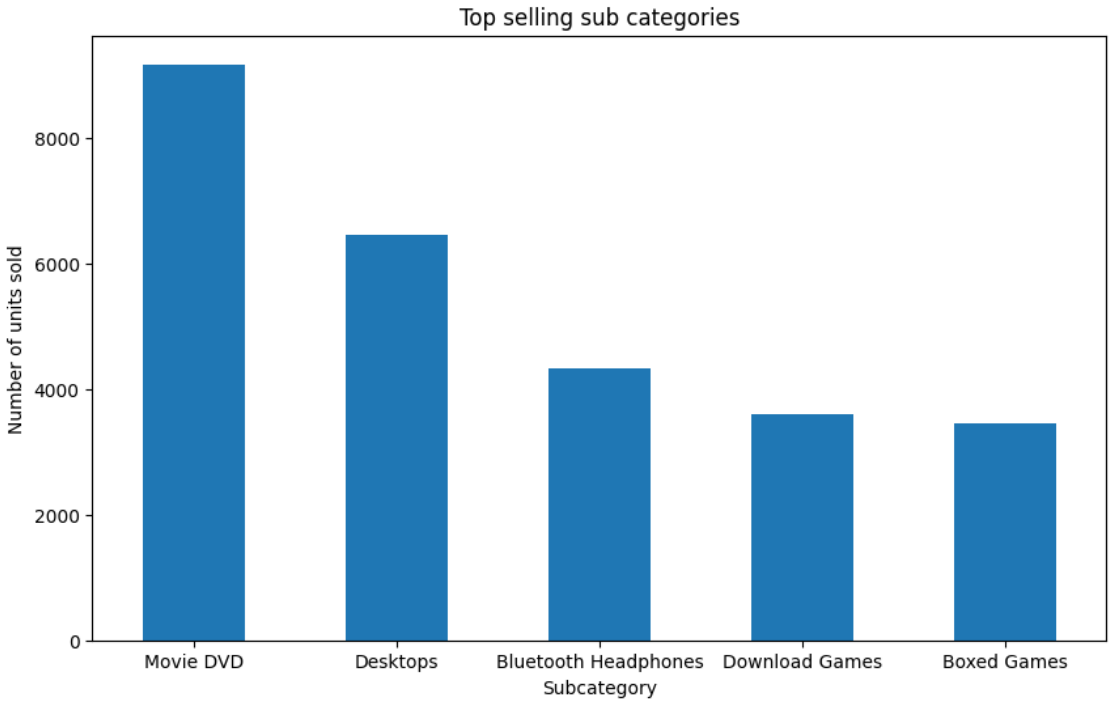
[ ]:
```
# Pie chart for the percentage share of the top 10 brands
top_10_brands_percentage = (top_10_brands / top_10_brands.sum()) * 100
top_10_brands_percentage.plot(kind='pie',
                              figsize=(12, 12),
                              autopct='%1.1f%%',
                              startangle = 180,
                              labels = None,
                              legend = True)

# Add title
plt.title("Percentage Share of Top 10 Brands")

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



[ ]:
```
#Top selling category
top_category = sales_product_data.groupby('Category')['Category'].count().
 ↪sort_values(ascending = False).head()

#Plotting
top_category.plot (kind = 'barh',figsize = (10,6))

#labels
plt.title('Top selling categories')
plt.xlabel('Number of units sold')
plt.ylabel('Category')

plt.show()
```



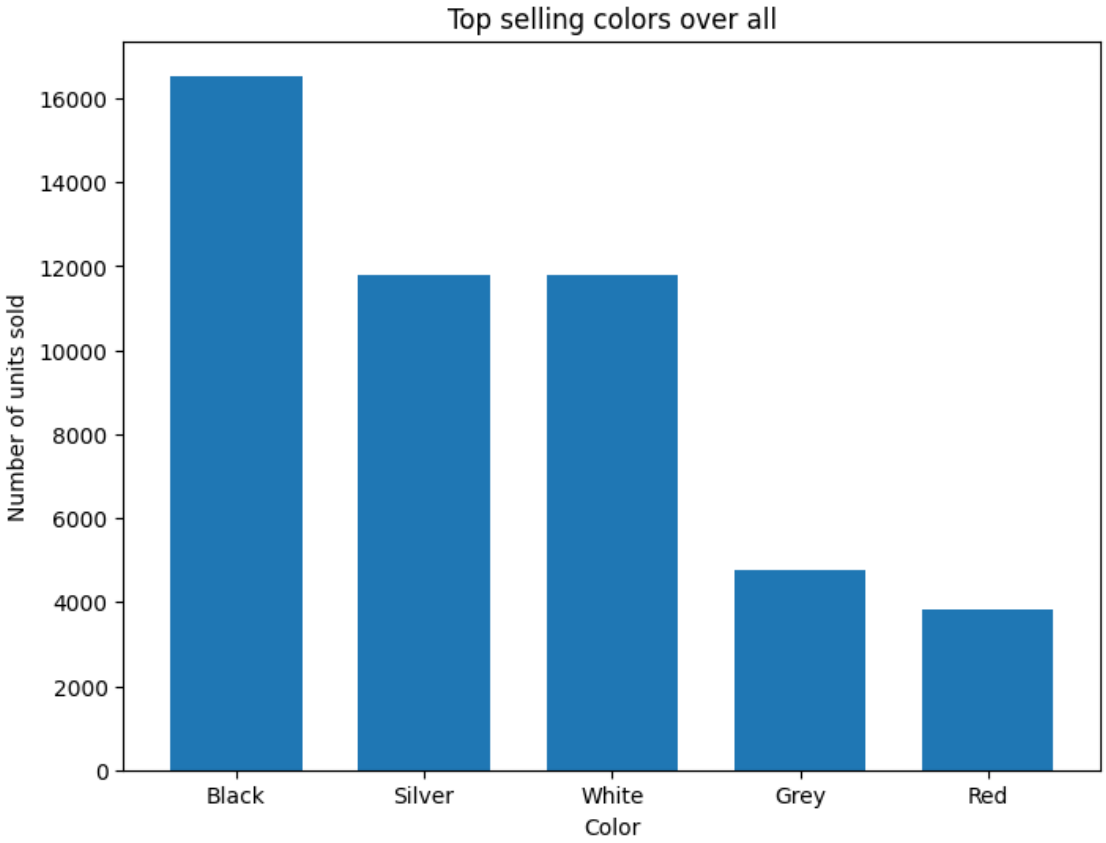
[ ]:
```
#Top 5 selling sub categories
top_subcategories = sales_product_data.groupby('Subcategory')['Subcategory'].
  ↪count().sort_values(ascending = False).head()

#Plotting
top_subcategories.plot (kind = 'bar',figsize = (10,6),rot = 0)

#labels
plt.title('Top selling sub categories')
plt.ylabel('Number of units sold')

plt.show()
```



[ ]:
```
sales_product_data.columns
```

[ ]:
```
Index(['Order Number', 'Line Item', 'Order Date', 'Delivery Date',
       'CustomerKey', 'StoreKey', 'ProductKey', 'Quantity', 'Currency Code',
       'Product Name', 'Brand', 'Color', 'Unit Cost USD', 'Unit Price USD',
       'SubcategoryKey', 'Subcategory', 'CategoryKey', 'Category'],
      dtype='object')
```

[ ]:
```
#Most selling colour
top_color = sales_product_data.groupby('Color')['Color'].count().
 ↪sort_values(ascending = False).head()

#Plotting
top_color.plot(kind = 'bar', figsize = (8,6),rot = 0,width = 0.7)

#Labels
plt.title("Top selling colors over all")
plt.ylabel("Number of units sold")

plt.show()
```



[ ]:
```
#Cheapest and costiest products and number of units sold
max_units_price = sales_product_data['Unit Price USD'].max()
count_of_max_unit_price = sales_product_data[sales_product_data['Unit Price␣
 ↪USD']== max_units_price].shape[0]
print(f'Maximum unit price USD: {max_units_price} \n')
print(f'Number of products sold at maximum unit price USD:␣
 ↪{count_of_max_unit_price} \n')

min_units_price = sales_product_data['Unit Price USD'].min()
count_of_min_unit_price = sales_product_data[sales_product_data['Unit Price␣
 ↪USD']== min_units_price].shape[0]
print(f'Minimum unit price USD: {min_units_price} \n')
print(f'Number of products sold at minimum unit price USD:␣
 ↪{count_of_min_unit_price} \n')
```

```
Maximum unit price USD: $999.90

Number of products sold at maximum unit price USD: 24

Minimum unit price USD: $0.95
```

```
Number of products sold at minimum unit price USD: 30
```

[ ]:
```
#Importing a new dataset
stores = pd.read_csv('/content/drive/MyDrive/Computers/Python/Datasets for data␣
↪analysis/Global Electronic Sales/Stores.csv')
```

[ ]:
```
stores.head()
```

[ ]:

	StoreKey	Country	State	Square Meters	Open Date
0	1	Australia	Australian Capital Territory	595.0	1/1/2008
1	2	Australia	Northern Territory	665.0	1/12/2008
2	3	Australia	South Australia	2000.0	1/7/2012
3	4	Australia	Tasmania	2000.0	1/1/2010
4	5	Australia	Victoria	2000.0	12/9/2015

Data Cleaning and Understanding

[ ]:
```
stores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67 entries, 0 to 66
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   StoreKey       67 non-null     int64
 1   Country        67 non-null     object
 2   State          67 non-null     object
 3   Square Meters  66 non-null     float64
 4   Open Date      67 non-null     object
dtypes: float64(1), int64(1), object(3)
memory usage: 2.7+ KB
```

[ ]:
```
#Covnerting opening date to date datatype
stores['Open Date'] = pd.to_datetime(stores['Open Date'])
```

[ ]:
```
stores.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67 entries, 0 to 66
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   StoreKey       67 non-null     int64
 1   Country        67 non-null     object
 2   State          67 non-null     object
 3   Square Meters  66 non-null     float64
```

```
 4   Open Date      67 non-null     datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(1), object(2)
memory usage: 2.7+ KB
```

[ ]:
```
#Checking for duplicated
stores.duplicated().sum()
#There are no duplicates
```

[ ]:
```
0
```

[ ]:
```
#Checking for null values
stores.isnull().sum()
#There is 1 null values
```

[ ]:
```
StoreKey         0
Country          0
State            0
Square Meters    1
Open Date        0
dtype: int64
```

[ ]:
```
#Checking the percentage of null values
(stores.isnull().sum()/len(stores))*100
#The error rate is less than 2%, so we can leave it as it is and work on the␣
 ↪insights.
```

[ ]:
```
StoreKey         0.000000
Country          0.000000
State            0.000000
Square Meters    1.492537
Open Date        0.000000
dtype: float64
```

[ ]:
```
#Checking the number of rows and col
stores.shape
```

[ ]:
```
(67, 5)
```

[ ]:
```
#Getting the statiscal data
stores.describe(include = 'all')
```

[ ]:
```
          StoreKey        Country                          State  Square Meters  \
count    67.000000             67                             67      66.000000
unique         NaN              9                             67            NaN
top            NaN  United States  Australian Capital Territory            NaN
freq           NaN             24                              1            NaN
mean     33.000000            NaN                            NaN    1402.196970
```

```
min       0.000000            NaN                          NaN     245.000000
25%      16.500000            NaN                          NaN    1108.750000
50%      33.000000            NaN                          NaN    1347.500000
75%      49.500000            NaN                          NaN    2000.000000
max      66.000000            NaN                          NaN    2105.000000
std      19.485037            NaN                          NaN     576.404058
```

```
                           Open Date
count                             67
unique                           NaN
top                              NaN
freq                             NaN
mean    2011-06-24 00:42:59.104477696
min               2005-03-04 00:00:00
25%               2009-06-03 00:00:00
50%               2010-06-03 00:00:00
75%               2013-06-07 00:00:00
max               2019-03-05 00:00:00
std                              NaN
```

Data Insights

[ ]:
```
stores.columns
```

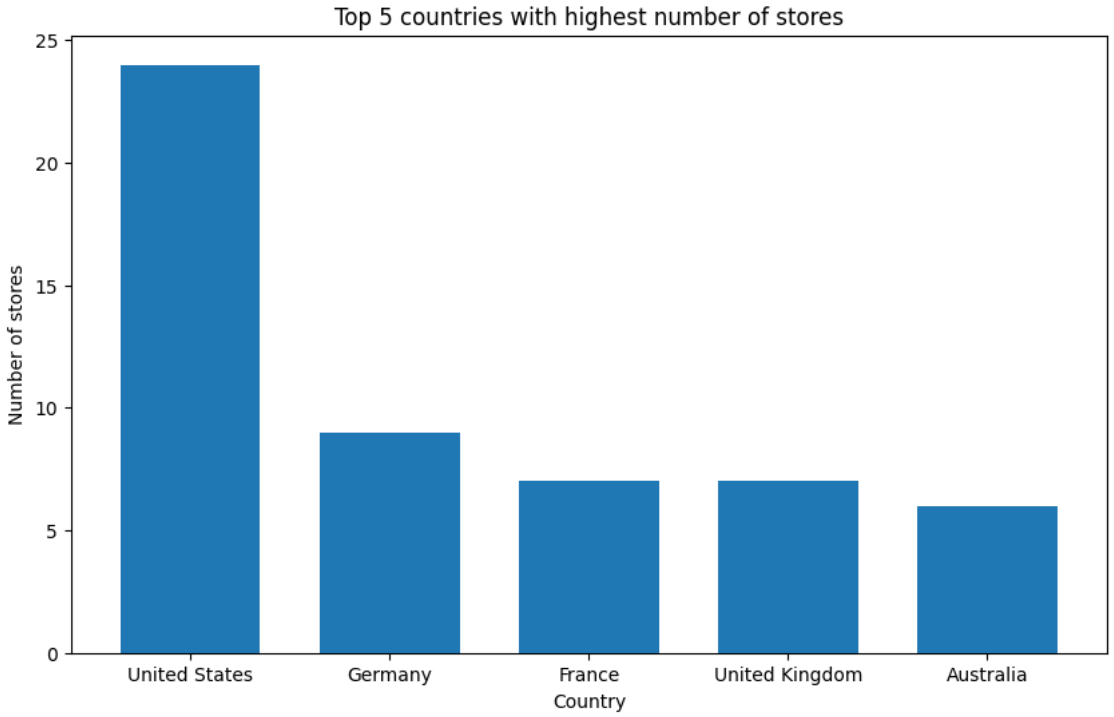
[ ]:
```
Index(['StoreKey', 'Country', 'State', 'Square Meters', 'Open Date'],
dtype='object')
```

[ ]:
```
#Countries with the highest number of stores
stores_by_country = stores.groupby('Country')['Country'].count().
 ↪sort_values(ascending = False).head()
print(stores_by_country)
```

```
Country
United States     24
Germany            9
France             7
United Kingdom     7
Australia          6
Name: Country, dtype: int64
```

[ ]:
```
#Plotting
stores_by_country.plot(kind = "bar", figsize = (10,6), rot = 0, width = 0.7)

#Lables
plt.title("Top 5 countries with highest number of stores")
plt.ylabel('Number of stores')
```

```
plt.show()
```



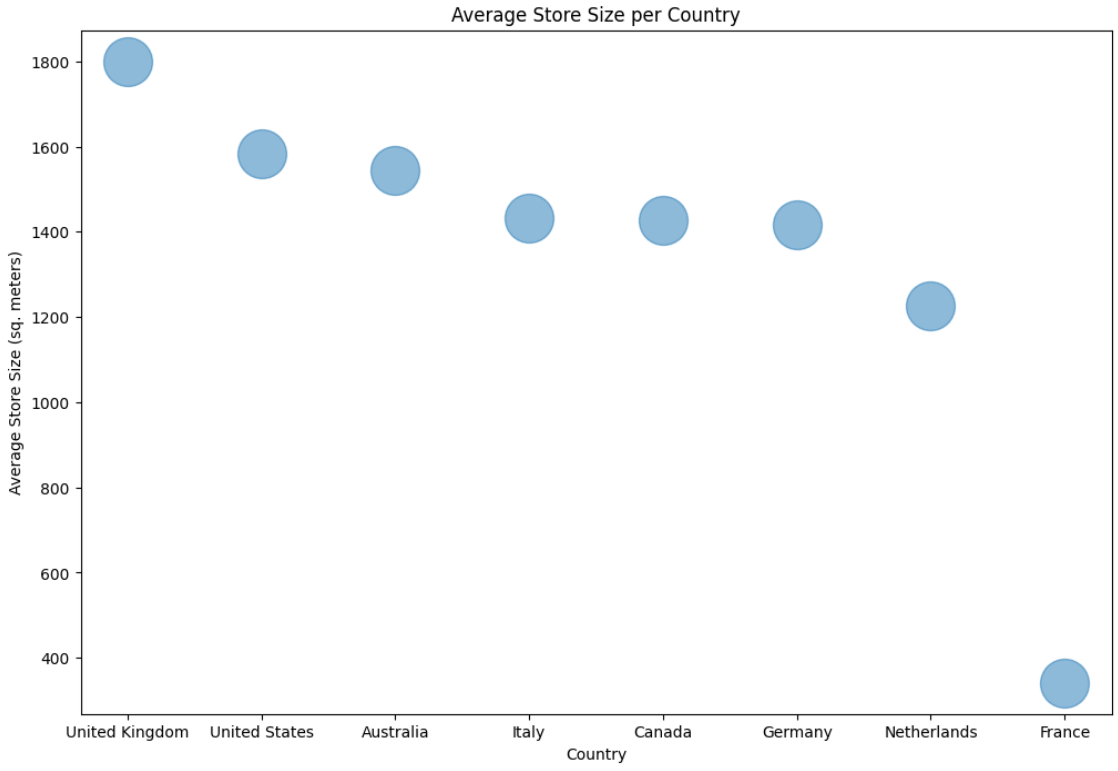
[ ]:
```
#Average size of the store
avg_size = round(stores['Square Meters'].mean(),2)
print(f'The Average size of the store is: {avg_size} sq.mts')
```

```
The Average size of the store is: 1402.2 sq.mts
```

[ ]:
```
#Average size of the store per country
avg_per_country = round(stores.groupby('Country')['Square Meters'].mean(),2).
  ↪sort_values(ascending = False).head(8)
print(avg_per_country)
```

```
Country
United Kingdom    1800.00
United States     1582.92
Australia         1543.33
Italy             1433.33
Canada            1426.00
Germany           1416.67
Netherlands       1225.00
France             341.43
Name: Square Meters, dtype: float64
```

[ ]:
```
# PLotting
x = avg_per_country.index
y = avg_per_country.values

# Create the bubble chart
plt.figure(figsize=(12, 8))
plt.scatter(x, y, s=1000, alpha=0.5)

# Add title and axis labels
plt.title("Average Store Size per Country")
plt.xlabel("Country")
plt.ylabel("Average Store Size (sq. meters)")

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



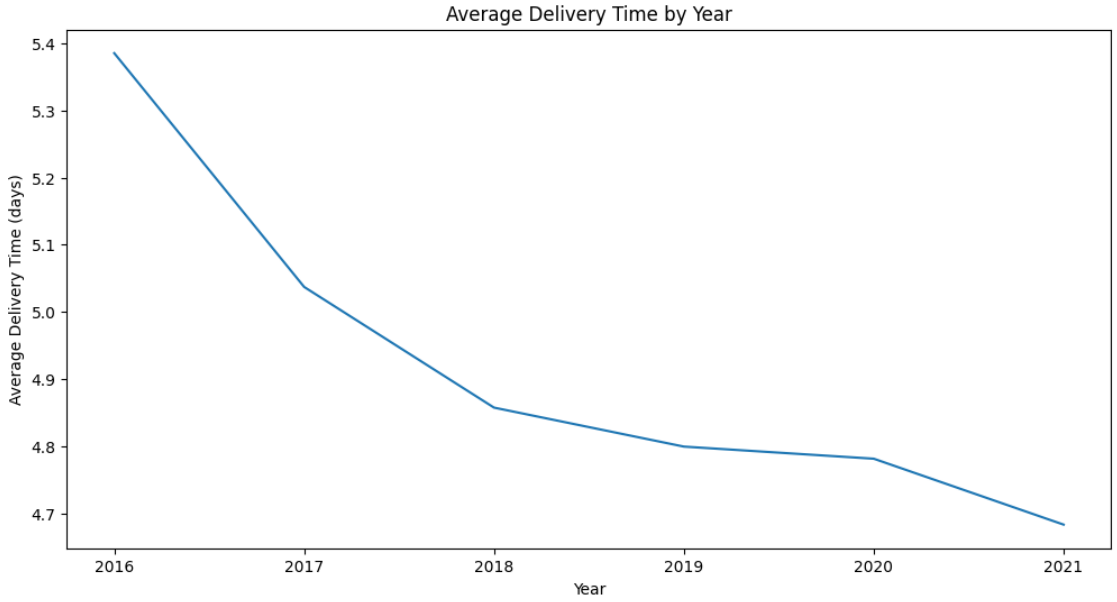
[ ]:
```
#Largest and smallest store
largest_store = stores.groupby('Country')['Square Meters'].max().
 ↪sort_values(ascending = False).head(1)
print('The largest store is in:',largest_store)
```

```
smallest_store = stores.groupby('Country')['Square Meters'].min().
 ↪sort_values(ascending = True).head(1)
print('The smallest store is in:',smallest_store)
```

```
The largest store is in: Country
Canada    2105.0
Name: Square Meters, dtype: float64
The smallest store is in: Country
France    245.0
Name: Square Meters, dtype: float64
```

1 Other Insigths

[ ]:
```
#Average delivery time over the years.
sales_product_data['Delivery Days'] = (sales_product_data['Delivery Date'] -␣
 ↪sales_product_data['Order Date']).dt.days

# Calculate the average delivery time for each year
avg_delivery_time_by_year = sales_product_data.
 ↪groupby(sales_product_data['Order Date'].dt.year)['Delivery Days'].mean()

# Create a line chart
plt.figure(figsize=(12, 6))
plt.plot(avg_delivery_time_by_year.index, avg_delivery_time_by_year.values)

# Add title and axis labels
plt.title("Average Delivery Time by Year")
plt.xlabel("Year")
plt.ylabel("Average Delivery Time (days)")

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


Average Delivery Time by Year
Average Delivery Time (days)
5.4
5.3
5.2
5.1
5.0
4.9
4.8
4.7
2016
2017
2018
2019
2020
2021
Year