

Importing necessary libraries

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import mysql.connector
import warnings

# Ignoring warnings for clean output
warnings.filterwarnings('ignore')
```

Loading Data

```
In [2]: # Loading the data from CSV files
customers = pd.read_csv('Customers.csv', encoding='latin-1')
exchange_rate = pd.read_csv('Exchange_Rates.csv')
products = pd.read_csv('Products.csv')
sales = pd.read_csv('Sales.csv')
stores = pd.read_csv('Stores.csv')

# Display the first few rows of each dataframe to inspect them
print("Customers DataFrame:")
print(customers.head())
print("\nExchange Rate DataFrame:")
print(exchange_rate.head())
print("\nProducts DataFrame:")
print(products.head())
print("\nSales DataFrame:")
print(sales.head())
print("\nStores DataFrame:")
print(stores.head())
```

Customers DataFrame:

	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

Exchange Rate DataFrame:

	Date	Currency	Exchange
0	1/1/2015	USD	1.0000
1	1/1/2015	CAD	1.1583
2	1/1/2015	AUD	1.2214
3	1/1/2015	EUR	0.8237
4	1/1/2015	GBP	0.6415

Products DataFrame:

	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

Sales DataFrame:

	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	

Stores DataFrame:

	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

1.Customer Table

```
In [3]: # Load customer table
path="Customers.csv"
df_customers=pd.read_csv(path,encoding='latin-1')
df_customers
```

```
Out[3]:
```

	CustomerKey	Gender	Name	City	State Code	State	Zip Code	Country
0	301	Female	Lilly Harding	WANDEARAH EAST	SA	South Australia	5523	Australia
1	325	Female	Madison Hull	MOUNT BUDD	WA	Western Australia	6522	Australia
2	554	Female	Claire Ferres	WINJALLOK	VIC	Victoria	3380	Australia
3	786	Male	Jai Poltpalingada	MIDDLE RIVER	SA	South Australia	5223	Australia
4	1042	Male	Aidan Pankhurst	TAWONGA SOUTH	VIC	Victoria	3698	Australia
...
15261	2099600	Female	Denisa Duřková	Houston	TX	Texas	77017	United States
15262	2099618	Male	Justin Solórzano	Mclean	VA	Virginia	22101	United States
15263	2099758	Male	Svend Petrussen	Wilmington	NC	North Carolina	28405	United States
15264	2099862	Female	Lorenza Rush	Riverside	CA	California	92501	United States
15265	2099937	Male	Zygmunt Kaminski	Bloomfield Township	MI	Michigan	48302	United States

15266 rows × 10 columns



Why Use `latin-1` Encoding?

1. Non-UTF-8 Characters:

- Some CSV files may contain special characters (e.g., accented letters like `é`, `ç`) or symbols not supported by UTF-8.
- Using `latin-1` ensures these characters are interpreted correctly during file reading.

2. Compatibility:

- `latin-1` (ISO-8859-1) is a single-byte encoding widely used for Western languages and is often the fallback for files with non-UTF-8 characters.

3. Avoids Errors:

- Without specifying the encoding, pandas may raise a `UnicodeDecodeError` if the file contains incompatible characters.

```
In [4]: # Summarize the DataFrame
print("\nInfo:")
print(customers.info())
print("\nDescribe:")
print(customers.describe(include='all'))
```

```
Info:
<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
None
```

```
Describe:
```

	CustomerKey	Gender	Name	City	State Code	State \
count	1.526600e+04	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	1.060508e+06	NaN	NaN	NaN	NaN	NaN
std	6.127097e+05	NaN	NaN	NaN	NaN	NaN
min	3.010000e+02	NaN	NaN	NaN	NaN	NaN
25%	5.140335e+05	NaN	NaN	NaN	NaN	NaN
50%	1.079244e+06	NaN	NaN	NaN	NaN	NaN
75%	1.593980e+06	NaN	NaN	NaN	NaN	NaN
max	2.099937e+06	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

```
In [5]: # Check for Duplicates
duplicate_rows = customers.duplicated().sum()
print(f"\nNumber of duplicate rows: {duplicate_rows}")
```

Number of duplicate rows: 0

```
In [6]: # Statistical Summary (for numerical columns)
if customers.select_dtypes(include=[float, int]).columns.size > 0:
    print("\nStatistical summary for numerical columns:")
    print(customers.describe())
else:
    print("\nNo numerical columns for statistical summary.")
```

Statistical summary for numerical columns:

	CustomerKey
count	1.526600e+04
mean	1.060508e+06
std	6.127097e+05
min	3.010000e+02
25%	5.140335e+05
50%	1.079244e+06
75%	1.593980e+06
max	2.099937e+06

```
In [7]: # Data Types Validation
print("\nData types of each column:")
print(customers.dtypes)
```

Data types of each column:

CustomerKey	int64
Gender	object
Name	object
City	object
State Code	object
State	object
Zip Code	object
Country	object
Continent	object
Birthday	object
dtype:	object

```
In [8]: # Identify Missing Values
missing_values = customers.isnull().sum()
print("\nMissing values in each column:")
print(missing_values)
```

```
Missing values in each column:
CustomerKey      0
Gender           0
Name             0
City             0
State Code       10
State            0
Zip Code         0
Country          0
Continent        0
Birthday         0
dtype: int64
```

```
In [9]: # drop the missing values
df_customers.dropna(inplace=True)
df_customers.isnull().sum()
```

```
Out[9]: CustomerKey      0
Gender                 0
Name                  0
City                  0
State Code             0
State                 0
Zip Code               0
Country               0
Continent              0
Birthday              0
dtype: int64
```

Calculating the age of the customer with DOB

```
In [10]: from datetime import datetime
datetime.today()
```

```
Out[10]: datetime.datetime(2024, 12, 24, 15, 53, 31, 844207)
```

```
In [11]: # Convert 'Birthday' column to datetime
customers['Birthday'] = pd.to_datetime(customers['Birthday'], format='%m/%d/%Y')

# Function to calculate age
def calculate_age(born):
    today = datetime.today()
    age = today.year - born.year - ((today.month, today.day) < (born.month, born.day))
    return age

# Apply the function to the 'Birthday' column
customers['Age'] = customers['Birthday'].apply(calculate_age)
```


In [12]: customers

Out[12]:

	CustomerKey	Gender	Name	City	State Code	State	Zip Code	Country
0	301	Female	Lilly Harding	WANDEARAH EAST	SA	South Australia	5523	Australia
1	325	Female	Madison Hull	MOUNT BUDD	WA	Western Australia	6522	Australia
2	554	Female	Claire Ferres	WINJALLOK	VIC	Victoria	3380	Australia
3	786	Male	Jai Poltpalingada	MIDDLE RIVER	SA	South Australia	5223	Australia
4	1042	Male	Aidan Pankhurst	TAWONGA SOUTH	VIC	Victoria	3698	Australia
...
15261	2099600	Female	Denisa Duřková	Houston	TX	Texas	77017	United States
15262	2099618	Male	Justin Solórzano	Mclean	VA	Virginia	22101	United States
15263	2099758	Male	Svend Petrussen	Wilmington	NC	North Carolina	28405	United States
15264	2099862	Female	Lorenza Rush	Riverside	CA	California	92501	United States
15265	2099937	Male	Zygmunt Kaminski	Bloomfield Township	MI	Michigan	48302	United States

15266 rows × 11 columns

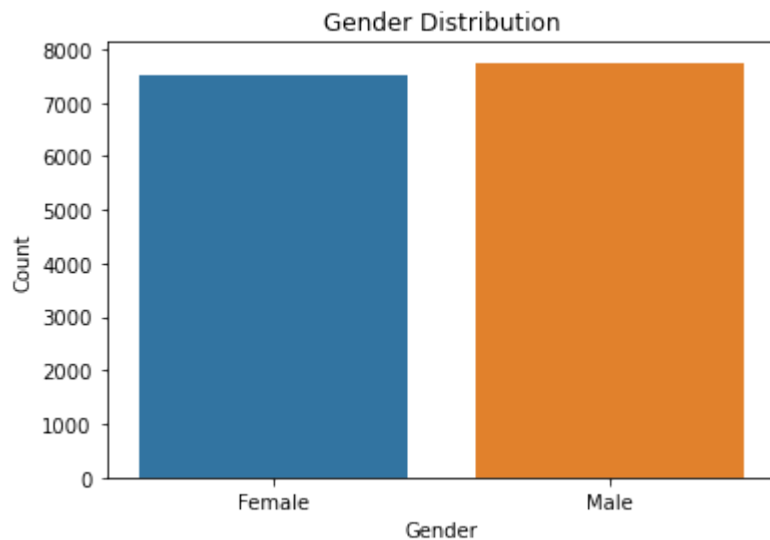


In [13]: customers['Gender'].value_counts()

Out[13]: Male 7748
Female 7518
Name: Gender, dtype: int64

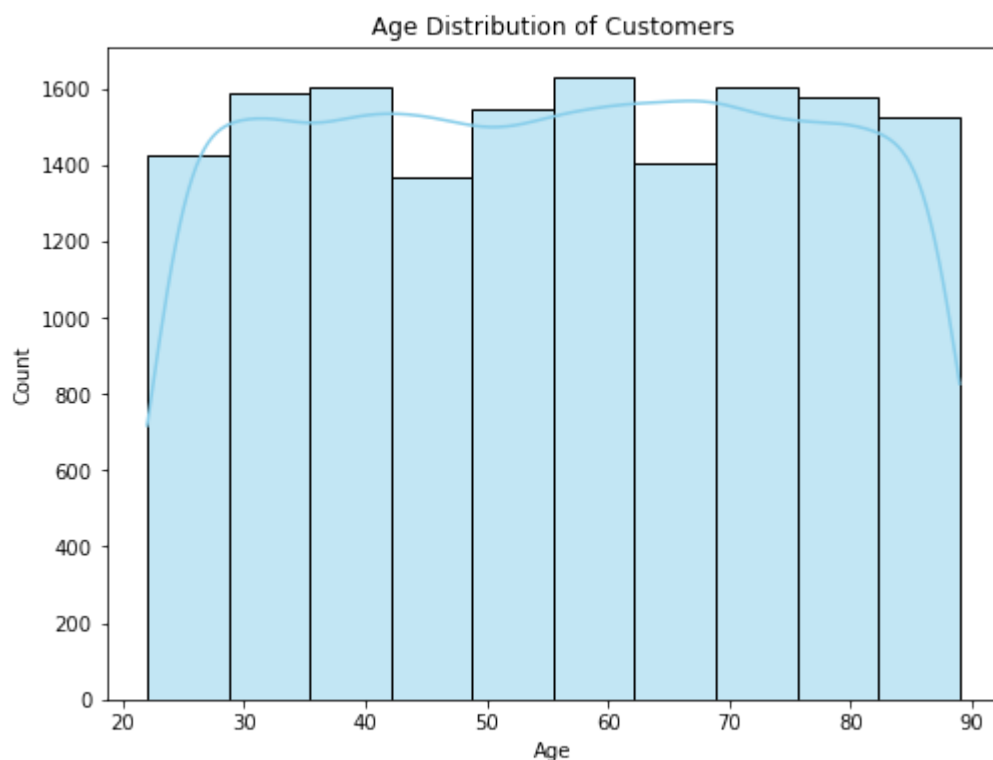
In [14]: # Gender Distribution:

```
plt.figure(figsize=(6, 4))
sns.countplot(data=customers, x='Gender')
plt.title('Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```



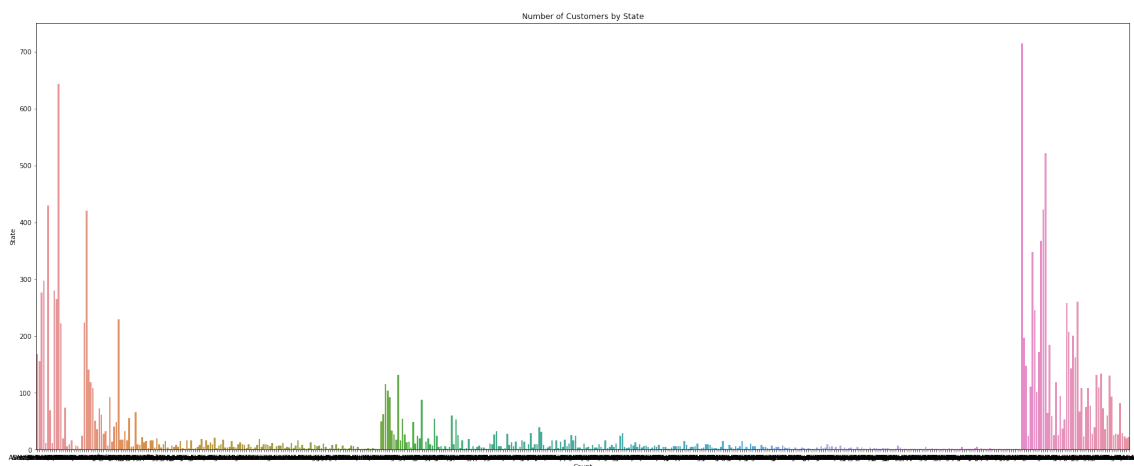
In [15]: # Age Distribution:

```
plt.figure(figsize=(8, 6))
sns.histplot(customers['Age'], bins=10, kde=True, color='skyblue')
plt.title('Age Distribution of Customers')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



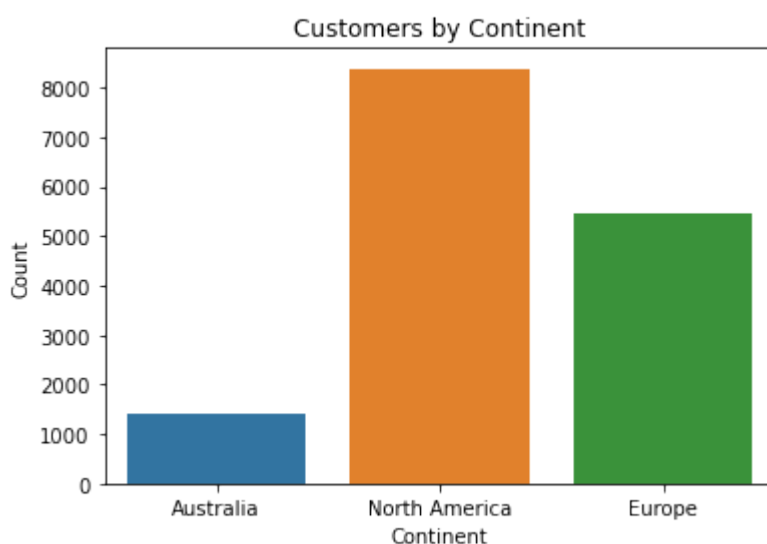
In [16]: *# Customers by State:*

```
plt.figure(figsize=(30, 12))
sns.countplot(data=customers, x='State')
plt.title('Number of Customers by State')
plt.xlabel('Count')
plt.ylabel('State')
plt.show()
```



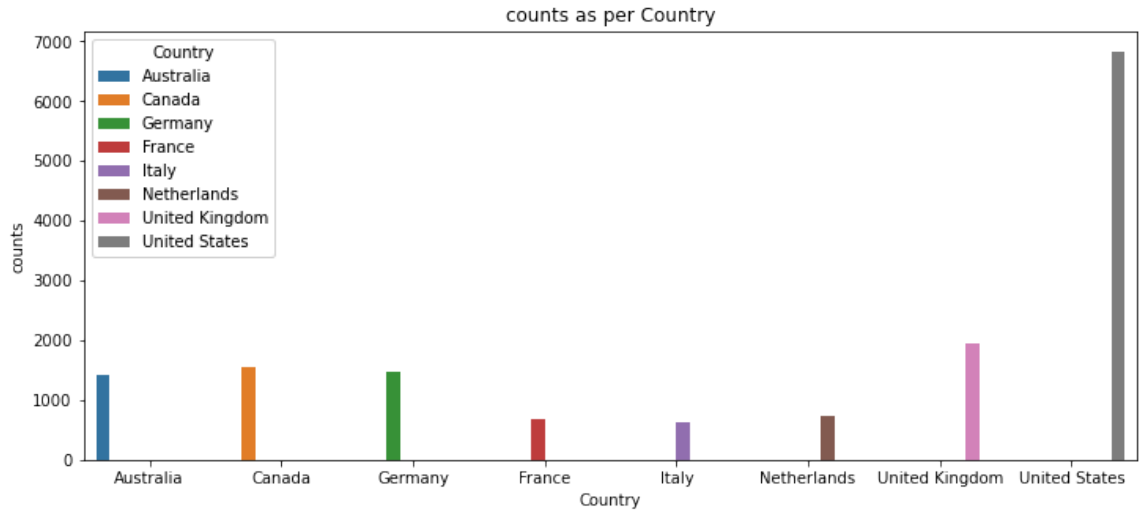
In [17]: *#Customers by Continent:*

```
plt.figure(figsize=(6, 4))
sns.countplot(data=customers, x='Continent')
plt.title('Customers by Continent')
plt.xlabel('Continent')
plt.ylabel('Count')
plt.show()
```

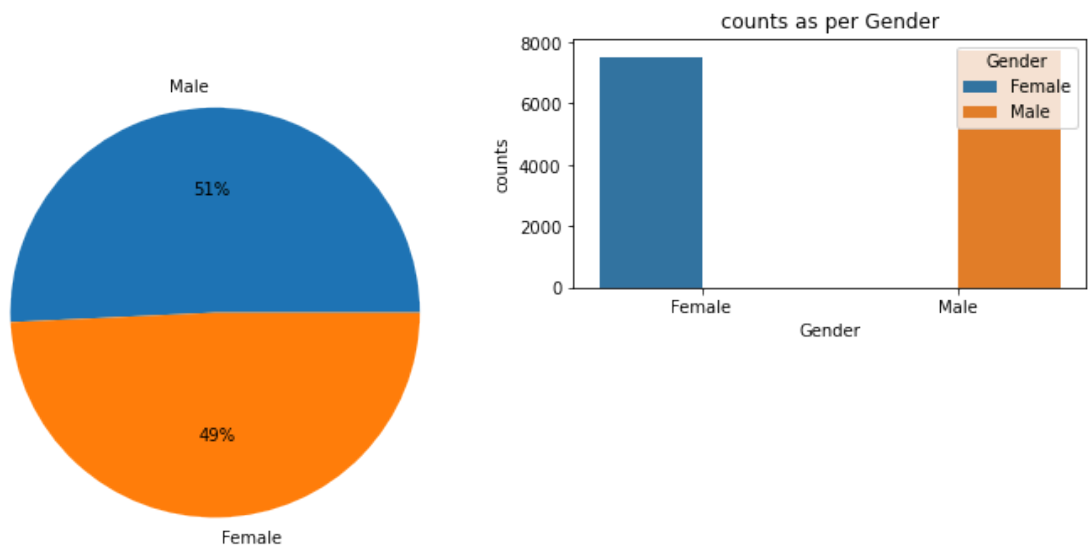


```
In [18]: # visualize country counts
plt.figure(figsize=(12,5))
sns.countplot(x=df_customers["Country"],data=customers,hue="Country")
plt.xlabel("Country")
plt.ylabel("counts")
plt.title("counts as per Country")
```

Out[18]: Text(0.5, 1.0, 'counts as per Country')



```
In [19]: # visualize gender details
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
plt.pie(df_customers["Gender"].value_counts(),
        autopct='%1.0f%%',
        labels=['Male', 'Female'])
plt.subplot(2,2,2)
sns.countplot(x=df_customers["Gender"],data=customers,hue="Gender")
plt.xlabel("Gender")
plt.ylabel("counts")
plt.title("counts as per Gender")
plt.show()
```



2. Exchange Rates

In [20]:

```
# Load exchange rates table
path="Exchange_Rates.csv"
df_exchange=pd.read_csv(path)
df_exchange
```

Out[20]:

	Date	Currency	Exchange
0	1/1/2015	USD	1.0000
1	1/1/2015	CAD	1.1583
2	1/1/2015	AUD	1.2214
3	1/1/2015	EUR	0.8237
4	1/1/2015	GBP	0.6415
...
11210	2/20/2021	USD	1.0000
11211	2/20/2021	CAD	1.2610
11212	2/20/2021	AUD	1.2723
11213	2/20/2021	EUR	0.8238
11214	2/20/2021	GBP	0.7126

11215 rows × 3 columns

In [21]:

```
exchange_rate.isna().sum()
```

Out[21]:

```
Date      0
Currency  0
Exchange  0
dtype: int64
```

In [22]:

```
exchange_rate.duplicated().sum()
0
```

Out[22]:

0

```
In [23]: exchange_rate.describe()
```

```
Out[23]:
```

	Exchange
count	11215.000000
mean	1.061682
std	0.245519
min	0.628500
25%	0.857800
50%	1.000000
75%	1.311900
max	1.725300

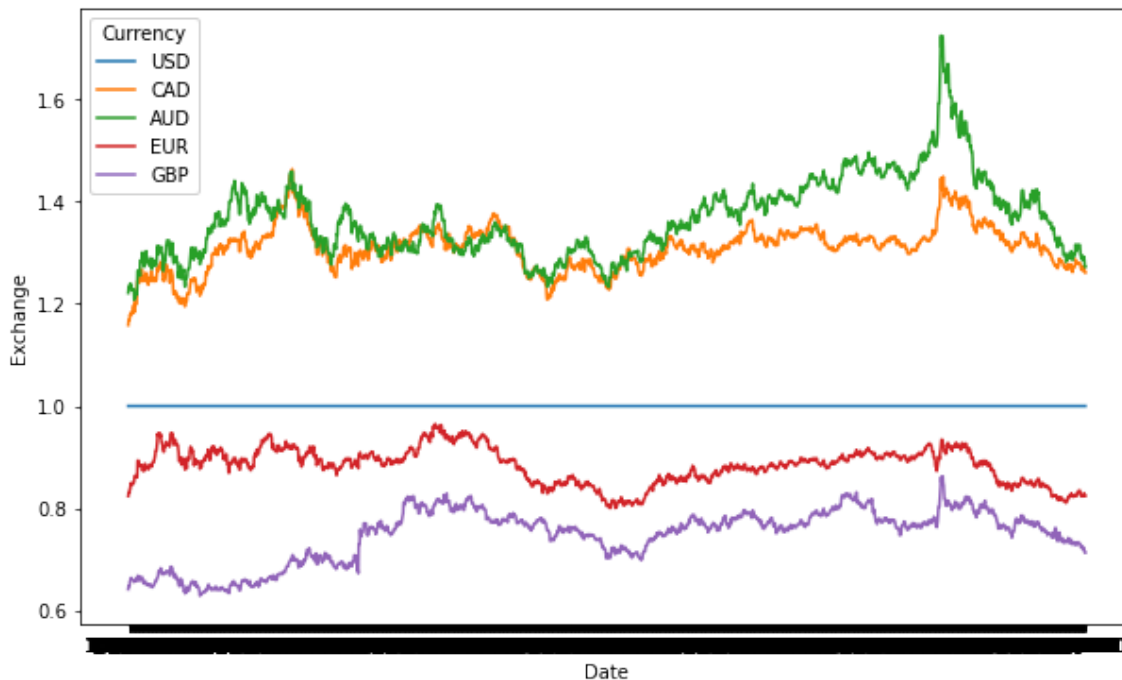
```
In [24]: exchange_rate.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 11215 entries, 0 to 11214  
Data columns (total 3 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   Date        11215 non-null  object  
1   Currency    11215 non-null  object  
2   Exchange    11215 non-null  float64  
dtypes: float64(1), object(2)  
memory usage: 263.0+ KB
```

In [25]: *#plot exchange rate, currency wise over the date*

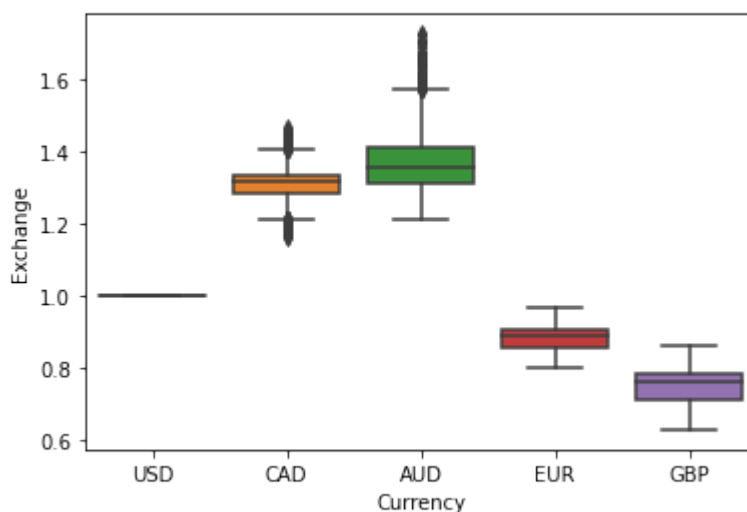
```
plt.figure(figsize=(10,6))  
sns.lineplot(data=exchange_rate,x='Date',y='Exchange',hue='Currency')
```

Out[25]: <AxesSubplot:xlabel='Date', ylabel='Exchange'>

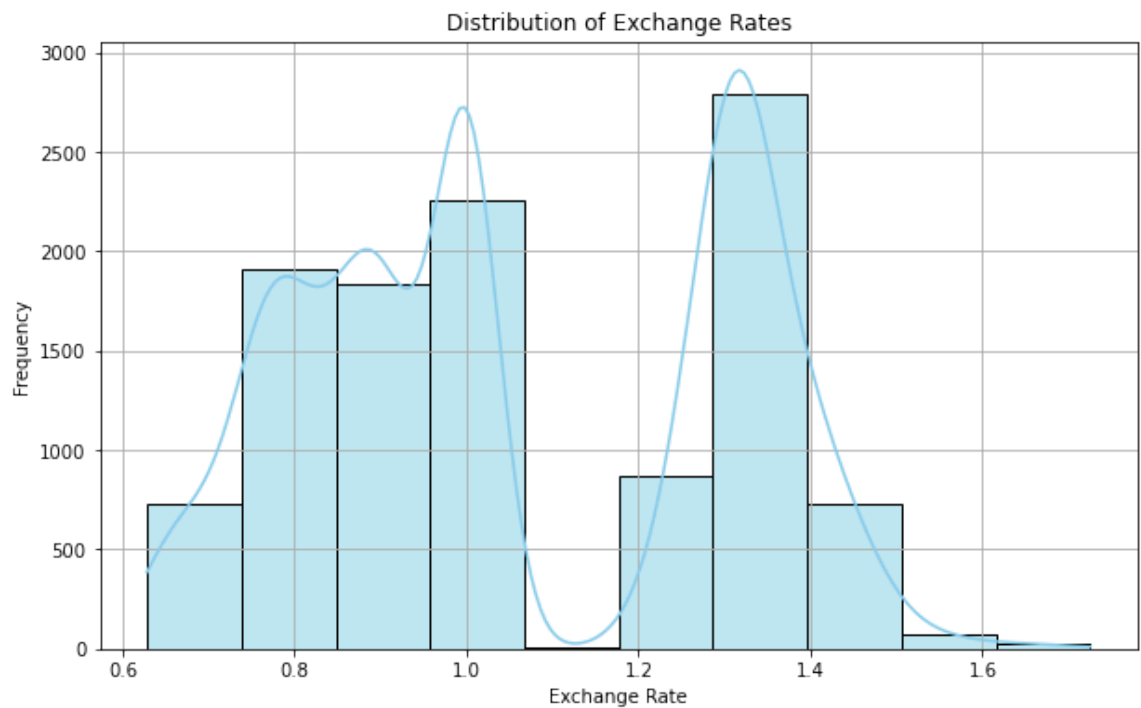


In [26]: `sns.boxplot(x='Currency', y = 'Exchange', data = exchange_rate)`

Out[26]: <AxesSubplot:xlabel='Currency', ylabel='Exchange'>



```
In [27]: plt.figure(figsize=(10, 6))
sns.histplot(exchange_rate['Exchange'], bins=10, kde=True, color='skyblue')
plt.title('Distribution of Exchange Rates')
plt.xlabel('Exchange Rate')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



3. Products

```
In [28]: # Load product table
path="Products.csv"
products=pd.read_csv(path)
products
```

Out[28]:

	ProductKey	Product Name	Brand	Color	Unit Cost USD	Unit Price USD	SubcategoryKey	Subcategory
0	1	Contoso 512MB MP3 Player E51 Silver	Contoso	Silver	\$6.62	\$12.99	101	MP4&MP
1	2	Contoso 512MB MP3 Player E51 Blue	Contoso	Blue	\$6.62	\$12.99	101	MP4&MP
2	3	Contoso 1G MP3 Player E100 White	Contoso	White	\$7.40	\$14.52	101	MP4&MP
3	4	Contoso 2G MP3 Player E200 Silver	Contoso	Silver	\$11.00	\$21.57	101	MP4&MP
4	5	Contoso 2G MP3 Player E200 Red	Contoso	Red	\$11.00	\$21.57	101	MP4&MP
...
2512	2513	Contoso Bluetooth Active Headphones L15 Red	Contoso	Red	\$43.07	\$129.99	505	Cell phone Accessorie
2513	2514	Contoso Bluetooth Active Headphones L15 White	Contoso	White	\$43.07	\$129.99	505	Cell phone Accessorie
2514	2515	Contoso In-Line Coupler E180 White	Contoso	White	\$1.71	\$3.35	505	Cell phone Accessorie
2515	2516	Contoso In-Line Coupler E180 Black	Contoso	Black	\$1.71	\$3.35	505	Cell phone Accessorie
2516	2517	Contoso In-Line Coupler E180 Silver	Contoso	Silver	\$1.71	\$3.35	505	Cell phone Accessorie

2517 rows × 10 columns



In [29]: *#check data types*
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
```

In [30]: *# Check for duplicate records*
products.duplicated().sum()

Out[30]: 0

In [31]: *# Checking for missing values*
products.isnull().sum()

Out[31]: 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

Units cost & price is prefixed with \$ symbol and keep the cost & price as float for easy calculations

In [32]: *# Remove \$ & spaces from Products Unit Cost & Price, covert as float and cr*
products['Unit Cost'] = products['Unit Cost USD'].str.replace('[\$,]', '',
products['Unit Price'] = products['Unit Price USD'].str.replace('[\$,]', ''

In [33]: products

Out[33]:

	ProductKey	Product Name	Brand	Color	Unit Cost USD	Unit Price USD	SubcategoryKey	Subcategor
0	1	Contoso 512MB MP3 Player E51 Silver	Contoso	Silver	\$6.62	\$12.99	101	MP4&MP
1	2	Contoso 512MB MP3 Player E51 Blue	Contoso	Blue	\$6.62	\$12.99	101	MP4&MP
2	3	Contoso 1G MP3 Player E100 White	Contoso	White	\$7.40	\$14.52	101	MP4&MP
3	4	Contoso 2G MP3 Player E200 Silver	Contoso	Silver	\$11.00	\$21.57	101	MP4&MP
4	5	Contoso 2G MP3 Player E200 Red	Contoso	Red	\$11.00	\$21.57	101	MP4&MP
...
2512	2513	Contoso Bluetooth Active Headphones L15 Red	Contoso	Red	\$43.07	\$129.99	505	Cell phone Accessorie
2513	2514	Contoso Bluetooth Active Headphones L15 White	Contoso	White	\$43.07	\$129.99	505	Cell phone Accessorie
2514	2515	Contoso In-Line Coupler E180 White	Contoso	White	\$1.71	\$3.35	505	Cell phone Accessorie
2515	2516	Contoso In-Line Coupler E180 Black	Contoso	Black	\$1.71	\$3.35	505	Cell phone Accessorie
2516	2517	Contoso In-Line Coupler E180 Silver	Contoso	Silver	\$1.71	\$3.35	505	Cell phone Accessorie

2517 rows × 12 columns

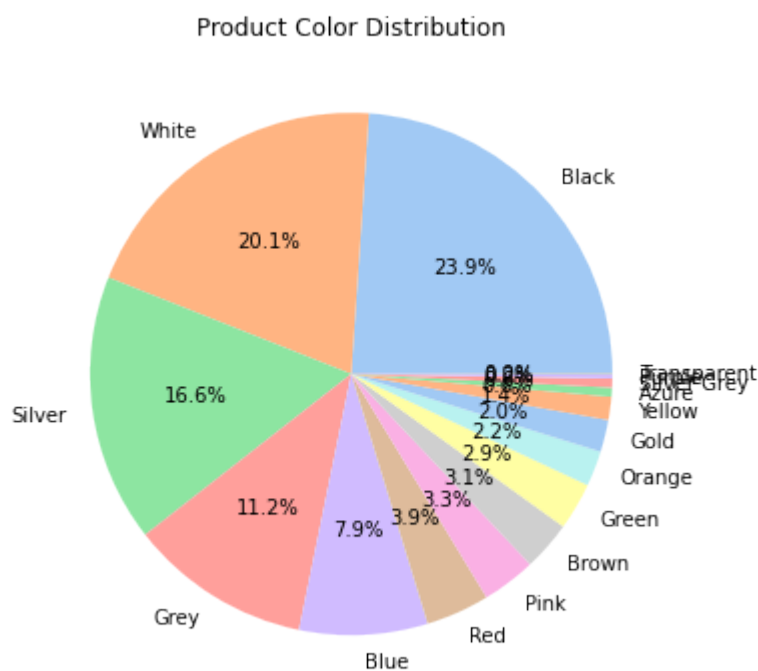


In [34]: `products.describe()`

Out[34]:

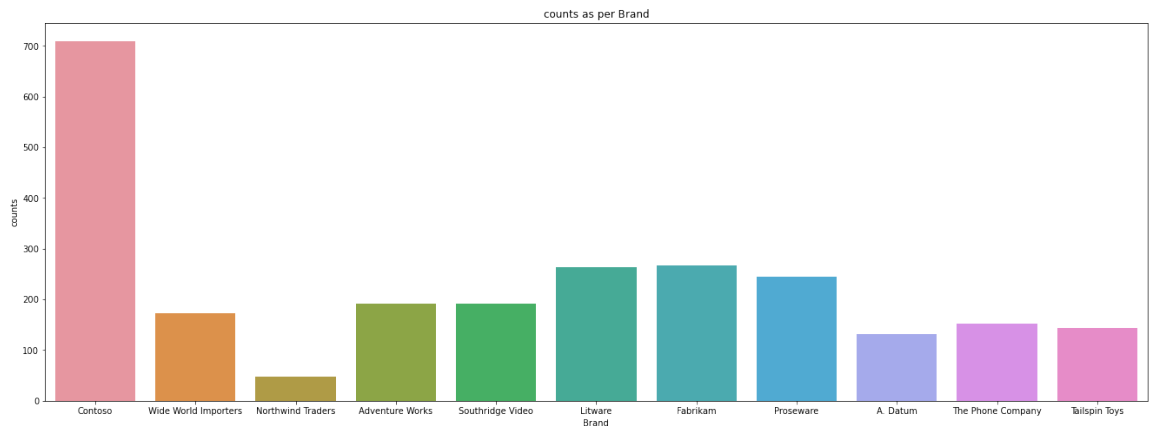
	ProductKey	SubcategoryKey	CategoryKey	Unit Cost	Unit Price
count	2517.000000	2517.000000	2517.000000	2517.000000	2517.000000
mean	1259.000000	491.810091	4.878824	147.655562	356.830131
std	726.739637	229.887134	2.299170	184.373683	494.054962
min	1.000000	101.000000	1.000000	0.480000	0.950000
25%	630.000000	305.000000	3.000000	32.250000	69.000000
50%	1259.000000	406.000000	4.000000	86.670000	199.990000
75%	1888.000000	801.000000	8.000000	183.950000	410.000000
max	2517.000000	808.000000	8.000000	1060.220000	3199.990000

In [35]: `plt.figure(figsize=(20, 6))`
`color_counts = products['Color'].value_counts()`
`plt.pie(color_counts, labels=color_counts.index, autopct='%1.1f%%', colors=)`
`plt.title('Product Color Distribution')`
`plt.show()`



Visualization of brand and category

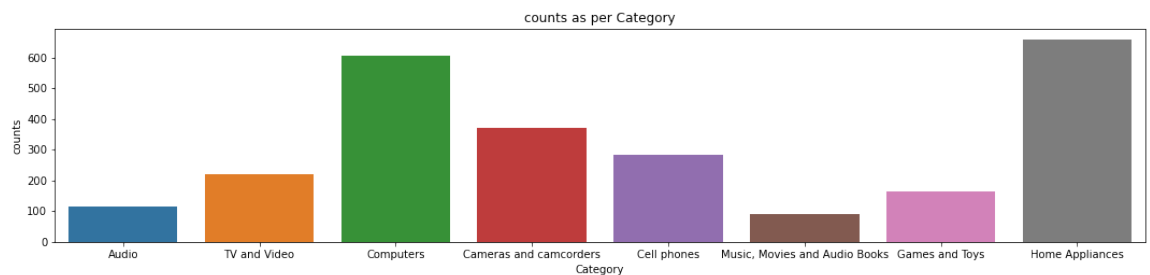
```
In [36]: plt.figure(figsize=(50,8))
plt.subplot(1,2,1)
sns.countplot(x=products["Brand"],data=products)
plt.xlabel("Brand")
plt.ylabel("counts")
plt.title("counts as per Brand")
plt.show()
```



Visualization for Category Distribution

```
In [37]: # Visualization for Category Distribution

plt.figure(figsize=(40,8))
plt.subplot(2,2,2)
sns.countplot(x=products["Category"],data=products)
plt.xlabel("Category")
plt.ylabel("counts")
plt.title("counts as per Category")
plt.show()
```



4. Sales

```
In [38]: # load sales table
path="Sales.csv"
df_sales=pd.read_csv(path)
df_sales
```

```
Out[38]:
```

	Order Number	Line Item	Order Date	Delivery Date	CustomerKey	StoreKey	ProductKey	Quantity	C
	0	366000	1	1/1/2016	NaN	265598	10	1304	1
	1	366001	1	1/1/2016	1/13/2016	1269051	0	1048	2
	2	366001	2	1/1/2016	1/13/2016	1269051	0	2007	1
	3	366002	1	1/1/2016	1/12/2016	266019	0	1106	7
	4	366002	2	1/1/2016	1/12/2016	266019	0	373	1

	62879	2243030	1	2/20/2021	NaN	1216913	43	632	3
	62880	2243031	1	2/20/2021	2/24/2021	511229	0	98	4
	62881	2243032	1	2/20/2021	2/23/2021	331277	0	1613	2
	62882	2243032	2	2/20/2021	2/23/2021	331277	0	1717	2
	62883	2243032	3	2/20/2021	2/23/2021	331277	0	464	7

62884 rows × 9 columns



```
In [39]: 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
```

```
In [40]: sales.isnull().sum()
```

```
Out[40]: 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
```

Delivery date for many of the sales records is missing.SO we can drop it

```
In [41]: sales.drop(columns="Delivery Date",inplace= True)
```

```
In [42]: sales.isnull().sum()
```

```
Out[42]: Order Number      0
Line Item      0
Order Date     0
CustomerKey    0
StoreKey       0
ProductKey     0
Quantity      0
Currency Code  0
dtype: int64
```

```
In [43]: #count 0 in StoreKey
(sales['StoreKey'] == 0).sum()
```

```
Out[43]: 13165
```

StoreKey is 0 for many rows.. There is a Store in Stores with 0. Make sure 0 is a valid store

In [44]: stores

Out[44]:

	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
...
62	63	United States	Utah	2000.0	3/6/2008
63	64	United States	Washington DC	1330.0	1/1/2010
64	65	United States	West Virginia	1785.0	1/1/2012
65	66	United States	Wyoming	840.0	1/1/2014
66	0	Online	Online	NaN	1/1/2010

67 rows × 5 columns

In [45]: stores[stores['StoreKey']!=0]

Out[45]:

	StoreKey	Country	State	Square Meters	Open Date
66	0	Online	Online	NaN	1/1/2010

In [46]: sales[sales['StoreKey'] == 0]

Out[46]:

	Order Number	Line Item	Order Date	CustomerKey	StoreKey	ProductKey	Quantity	Currency Code
1	366001	1	1/1/2016	1269051	0	1048	2	USD
2	366001	2	1/1/2016	1269051	0	2007	1	USD
3	366002	1	1/1/2016	266019	0	1106	7	CAD
4	366002	2	1/1/2016	266019	0	373	1	CAD
5	366002	3	1/1/2016	266019	0	1080	4	CAD
...
62878	2243029	6	2/20/2021	887764	0	1456	2	EUR
62880	2243031	1	2/20/2021	511229	0	98	4	EUR
62881	2243032	1	2/20/2021	331277	0	1613	2	CAD
62882	2243032	2	2/20/2021	331277	0	1717	2	CAD
62883	2243032	3	2/20/2021	331277	0	464	7	CAD

13165 rows × 8 columns

StoreKey 0 is for Online purchases, we should keep it

5. Stores

```
In [47]: # load sales table
path="Stores.csv"
df_stores=pd.read_csv(path)
df_stores
```

```
Out[47]:
```

	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
...
62	63	United States	Utah	2000.0	3/6/2008
63	64	United States	Washington DC	1330.0	1/1/2010
64	65	United States	West Virginia	1785.0	1/1/2012
65	66	United States	Wyoming	840.0	1/1/2014
66	0	Online	Online	NaN	1/1/2010

67 rows × 5 columns

```
In [48]: # check missing values
df_stores.isnull().sum()
```

```
Out[48]: StoreKey      0
Country      0
State        0
Square Meters  1
Open Date    0
dtype: int64
```

```
In [49]: #check data types
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
```

In [50]: `stores.fillna(0)`

Out[50]:

	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
...
62	63	United States	Utah	2000.0	3/6/2008
63	64	United States	Washington DC	1330.0	1/1/2010
64	65	United States	West Virginia	1785.0	1/1/2012
65	66	United States	Wyoming	840.0	1/1/2014
66	0	Online	Online	0.0	1/1/2010

67 rows × 5 columns

In [51]: `stores['Open Date'] = pd.to_datetime(df_stores['Open Date'], format="%m/%d/%Y")`

In [52]: `# replacing the nan value`
`stores = df_stores.replace({np.nan: None})`

In [53]: `# check missing values`
`df_stores.isnull().sum()`

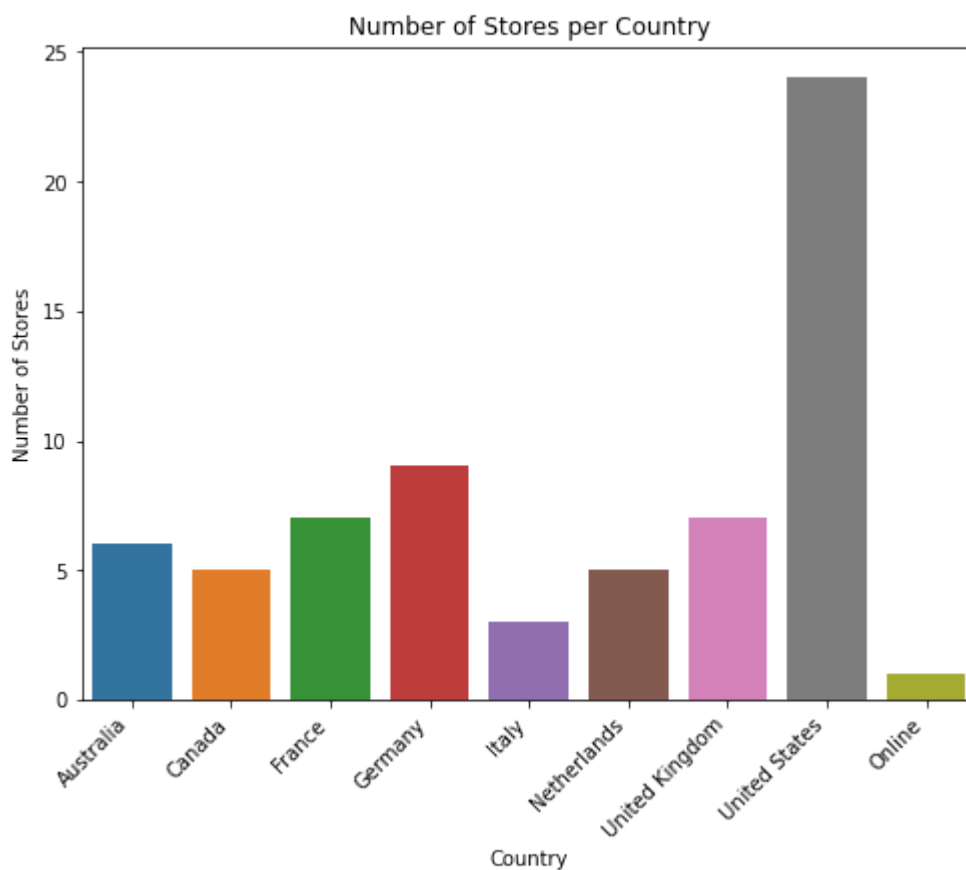
Out[53]:

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

Visualization for Number of Stores per Country

```
In [54]: plt.figure(figsize=(8, 6))
sns.countplot(x='Country', data=stores)
plt.title('Number of Stores per Country')
plt.xlabel('Country')
plt.ylabel('Number of Stores')

# Rotate the x-axis labels
plt.xticks(rotation=45, ha='right')
plt.show()
```



Create the sql database

```
In [55]: import mysql.connector

# Connect to MySQL Server without specifying a database
conn = mysql.connector.connect(host="localhost", user="root", password="1234")
my_cursor = conn.cursor()

# Execute SQL command to create the database if it does not exist
my_cursor.execute("CREATE DATABASE IF NOT EXISTS customer_sales")
print("Database 'customer_sales' created successfully")

# Close the cursor and connection
my_cursor.close()
conn.close()
```

Database 'customer_sales' created successfully

Customer Details

```
In [56]: import mysql.connector

# Connect to MySQL Server without specifying a database
conn = mysql.connector.connect(host="localhost", user="root", password="1234")
my_cursor = conn.cursor()

# Execute SQL command to create the database if it does not exist
my_cursor.execute("CREATE DATABASE IF NOT EXISTS customer_sales")
print("Database 'customer_sales' created successfully")

# Close the cursor and initial connection
my_cursor.close()
conn.close()

# Reconnect to the specific database
conn = mysql.connector.connect(host="localhost", user="root", password="1234")
my_cursor = conn.cursor() # Reinitialize the cursor on the new connection

# Create the customer_details table
my_cursor.execute('''
CREATE TABLE IF NOT EXISTS customer_details(
    CustomerKey INT,
    Gender VARCHAR(255),
    Name VARCHAR(255),
    City VARCHAR(255),
    State VARCHAR(255),
    Country VARCHAR(255),
    Continent VARCHAR(255),
    Birthday VARCHAR(255)
)
''')
print("Table 'customer_details' created successfully")

# Close the cursor and connection again
```

Database 'customer_sales' created successfully
Table 'customer_details' created successfully

```
In [57]: stores[stores['Square Meters'].isnull()]

# Hence it is a Online Store Square Meters is "0". So no need any changes

stores.describe()

# Insert cleaned data into DB without merging as a backup

from sqlalchemy import create_engine
import pandas as pd

database_url = 'mysql+mysqlconnector://root:12345@localhost/DataSpark'
engine = create_engine(database_url)

try:
    customers.to_sql(name="customers", con=engine, if_exists="replace", index=False)
    exchange_rate.to_sql(name="exchange_rate", con=engine, if_exists="replace", index=False)
    products.to_sql(name="products", con=engine, if_exists="replace", index=False)
    sales.to_sql(name="sales", con=engine, if_exists="replace", index=False)
    stores.to_sql(name="stores", con=engine, if_exists="replace", index=False)
    print("Data successfully uploaded.")
except Exception as e:
    print(f"An error occurred: {e}")
finally:
    engine.dispose()
```

Data successfully uploaded.

Converting the all dates to date type

```
In [58]: #Birthday in customers to date format
customers['Birthday'] = pd.to_datetime(customers['Birthday'])

#convert Date in exchange to date format
exchange_rate['Date'] = pd.to_datetime(exchange_rate['Date'])

#Order date in Sales to date format
sales['Order Date'] = pd.to_datetime(sales['Order Date'])

#convert Open Date in stores to Date format
stores['Open Date'] = pd.to_datetime(stores['Open Date'])
```

Dropping Unwanted columns

```
In [59]: #Drop Unit Cost USD, Unit Price USD from products
products.drop(columns=['Unit Cost USD', 'Unit Price USD'], inplace=True)

#Drop Name, State Code from customers
customers.drop(columns=['Name', 'State Code'], inplace=True)
```

Merging all dataframes to a single dataframe with required

columns

- customers
- exchange_rate
- products

1. Merging Sales & Products based on product key

```
In [60]: # Merge the two DataFrames on 'ProductKey'

merged_df = pd.merge(sales, products, on='ProductKey', how='inner')
```

```
In [61]: merged_df.isnull().sum()
```

```
Out[61]: Order Number      0
Line Item                 0
Order Date                0
CustomerKey               0
StoreKey                  0
ProductKey                0
Quantity                 0
Currency Code             0
Product Name              0
Brand                     0
Color                     0
SubcategoryKey            0
Subcategory                0
CategoryKey               0
Category                  0
Unit Cost                 0
Unit Price                0
dtype: int64
```

```
In [62]: merged_df.duplicated().sum()
```

```
Out[62]: 0
```

```
In [63]: merged_df.shape

print("size of merged_df after merging 'Sales' & 'Products' with 'product key'")

size of merged_df after merging 'Sales' & 'Products' with 'product key':
(62884, 17)
```

2. Merging customers with CustomerKey

```
In [64]: # Merge the DataFrame on 'CustomerKey'

merged_df = pd.merge(merged_df, customers, on='CustomerKey', how='inner')
```

```
In [65]: merged_df.isna().sum()
```

```
Out[65]: Order Number      0
Line Item      0
Order Date     0
CustomerKey    0
StoreKey       0
ProductKey     0
Quantity      0
Currency Code  0
Product Name   0
Brand          0
Color          0
SubcategoryKey 0
Subcategory    0
CategoryKey    0
Category       0
Unit Cost      0
Unit Price     0
Gender         0
City           0
State          0
Zip Code       0
Country        0
Continent      0
Birthday       0
Age            0
dtype: int64
```


In [66]: merged_df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62884 entries, 0 to 62883
Data columns (total 25 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  datetime64[ns]
3   CustomerKey            62884 non-null  int64
4   StoreKey               62884 non-null  int64
5   ProductKey             62884 non-null  int64
6   Quantity              62884 non-null  int64
7   Currency Code          62884 non-null  object
8   Product Name           62884 non-null  object
9   Brand                  62884 non-null  object
10  Color                  62884 non-null  object
11  SubcategoryKey         62884 non-null  int64
12  Subcategory            62884 non-null  object
13  CategoryKey            62884 non-null  int64
14  Category               62884 non-null  object
15  Unit Cost              62884 non-null  float64
16  Unit Price             62884 non-null  float64
17  Gender                 62884 non-null  object
18  City                   62884 non-null  object
19  State                  62884 non-null  object
20  Zip Code               62884 non-null  object
21  Country                62884 non-null  object
22  Continent              62884 non-null  object
23  Birthday               62884 non-null  datetime64[ns]
24  Age                   62884 non-null  int64
dtypes: datetime64[ns](2), float64(2), int64(9), object(12)
memory usage: 12.5+ MB
```

In [67]: merged_df.shape

```
print("size of merged_df after merging 'Sales,Products' & 'customers' with
size of merged_df after merging 'Sales,Products' & 'customers' with 'Custo
merKey': (62884, 25)
```

3. Merging exchange rates Date & Currency

```
In [68]: # prefix ex_ to exchange_rate
exchange_rate = exchange_rate.add_prefix('ex_')

# Merge the DataFrames on 'Date' & 'Currency'
merged_df = pd.merge(
    merged_df,
    exchange_rate,
    left_on=['Order Date', 'Currency Code'],
    right_on=['ex_Date', 'ex_Currency'],
    how='left'
)
```

```
In [69]: merged_df.isna().sum()
```

```
Out[69]: Order Number      0
Line Item      0
Order Date      0
CustomerKey     0
StoreKey        0
ProductKey      0
Quantity        0
Currency Code    0
Product Name     0
Brand           0
Color           0
SubcategoryKey   0
Subcategory      0
CategoryKey      0
Category         0
Unit Cost        0
Unit Price       0
Gender           0
City            0
State           0
Zip Code        0
Country         0
Continent       0
Birthday        0
Age             0
ex_Date         0
ex_Currency     0
ex_Exchange     0
dtype: int64
```

```
In [70]: merged_df.duplicated().sum()
```

```
Out[70]: 0
```

```
In [71]: print("size of merged_df after merging exchange rates Date & Currency: ", merged_df.size)

size of merged_df after merging exchange rates Date & Currency: (62884, 28)
```

4. Merging stores with StoreKey

In [72]: `merged_df.columns`

Out[72]: Index(['Order Number', 'Line Item', 'Order Date', 'CustomerKey', 'StoreKey',
 'ProductKey', 'Quantity', 'Currency Code', 'Product Name', 'Brand',
 'Color', 'SubcategoryKey', 'Subcategory', 'CategoryKey', 'Category',
 'Unit Cost', 'Unit Price', 'Gender', 'City', 'State', 'Zip Code',
 'Country', 'Continent', 'Birthday', 'Age', 'ex_Date', 'ex_Currency',
 'ex_Exchange'],
 dtype='object')

In [73]: `stores.columns`

Out[73]: Index(['StoreKey', 'Country', 'State', 'Square Meters', 'Open Date'], dtype='object')

In [74]: `# prefix store_ to stores`
`stores = stores.add_prefix('store_')`

`final_df = pd.merge(merged_df, stores, left_on='StoreKey', right_on='store_')`

In [75]: `final_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62884 entries, 0 to 62883
Data columns (total 33 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  datetime64[ns]
3   CustomerKey                       62884 non-null  int64
4   StoreKey                          62884 non-null  int64
5   ProductKey                       62884 non-null  int64
6   Quantity                          62884 non-null  int64
7   Currency Code                     62884 non-null  object
8   Product Name                      62884 non-null  object
9   Brand                            62884 non-null  object
10  Color                             62884 non-null  object
11  SubcategoryKey                   62884 non-null  int64
12  Subcategory                     62884 non-null  object
13  CategoryKey                     62884 non-null  int64
14  Category                       62884 non-null  object
15  Unit Cost                       62884 non-null  float64
16  Unit Price                      62884 non-null  float64
17  Gender                          62884 non-null  object
18  City                            62884 non-null  object
19  State                           62884 non-null  object
20  Zip Code                        62884 non-null  object
21  Country                         62884 non-null  object
22  Continent                       62884 non-null  object
23  Birthday                        62884 non-null  datetime64[ns]
24  Age                             62884 non-null  int64
25  ex_Date                         62884 non-null  datetime64[ns]
26  ex_Currency                     62884 non-null  object
27  ex_Exchange                     62884 non-null  float64
28  store_StoreKey                  62884 non-null  int64
29  store_Country                   62884 non-null  object
30  store_State                     62884 non-null  object
31  store_Square Meters             49719 non-null  object
32  store_Open Date                 62884 non-null  datetime64[ns]
dtypes: datetime64[ns](4), float64(3), int64(10), object(16)
memory usage: 16.3+ MB
```

```
In [76]: final_df.isnull().sum()
```

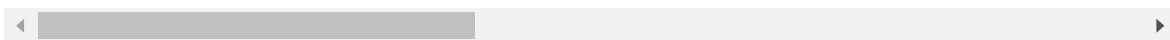
```
Out[76]: Order Number          0
Line Item                    0
Order Date                   0
CustomerKey                  0
StoreKey                     0
ProductKey                   0
Quantity                     0
Currency Code                0
Product Name                 0
Brand                       0
Color                       0
SubcategoryKey               0
Subcategory                  0
CategoryKey                  0
Category                     0
Unit Cost                    0
Unit Price                   0
Gender                       0
City                         0
State                        0
Zip Code                     0
Country                      0
Continent                    0
Birthday                     0
Age                           0
ex_Date                      0
ex_Currency                  0
ex_Exchange                  0
store_StoreKey               0
store_Country                0
store_State                  0
store_Square Meters          13165
store_Open Date              0
dtype: int64
```

```
In [77]: final_df[final_df["store_Square Meters"].isnull()]
```

```
Out[77]:
```

	Order Number	Line Item	Order Date	CustomerKey	StoreKey	ProductKey	Quantity	Currency Code	P
1	1749017	1	2019-10-15	265598	0	1619	1	CAD	C DVD M13
5	1749017	3	2019-10-15	265598	0	52	1	CAD	WV Pulse per
6	1749017	2	2019-10-15	265598	0	1764	2	CAD	MGS Emp The King
10	1848016	1	2020-01-22	1599716	0	1679	3	USD	MGS Gar kid
12	1848016	2	2020-01-22	1599716	0	23	1	USD	C 8G Play mode
...	
62844	1446001	1	2018-12-16	1395823	0	720	5	USD	Pro: Slim- M
62845	1105002	1	2018-01-09	404256	0	275	2	EUR	An: C T Syst C E1
62855	1837031	1	2020-01-11	215955	0	881	2	CAD	C Whe PS/2 E6C
62875	1723027	1	2019-09-19	1609370	0	2268	2	USD	WW Lam
62876	1710021	1	2019-09-06	1067403	0	820	1	GBP	C Home A E

13165 rows × 33 columns



Square Meters is null for all StoreKey == 0

- So we can fill those Square Meters null values with "0", Hence it is an Online Store

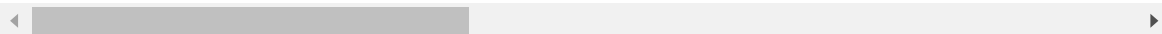
In [78]: `final_df['store_Square Meters'].fillna(0, inplace=True)`

`final_df[final_df['store_Square Meters'] == 0]`

Out[78]:

	Order Number	Line Item	Order Date	CustomerKey	StoreKey	ProductKey	Quantity	Currency Code	P
1	1749017	1	2019-10-15	265598	0	1619	1	CAD	C DVD M13
5	1749017	3	2019-10-15	265598	0	52	1	CAD	WV Pulse per
6	1749017	2	2019-10-15	265598	0	1764	2	CAD	MGS Emp The King
10	1848016	1	2020-01-22	1599716	0	1679	3	USD	MGS Gar kid
12	1848016	2	2020-01-22	1599716	0	23	1	USD	C 8G Play mode
...	
62844	1446001	1	2018-12-16	1395823	0	720	5	USD	Pro: Slim- M
62845	1105002	1	2018-01-09	404256	0	275	2	EUR	An: C T Syst C E1
62855	1837031	1	2020-01-11	215955	0	881	2	CAD	C Whee PS/2 E6C
62875	1723027	1	2019-09-19	1609370	0	2268	2	USD	WW Lamp
62876	1710021	1	2019-09-06	1067403	0	820	1	GBP	C Home A E

13165 rows × 33 columns




```
In [79]: final_df.duplicated().sum()
```

```
Out[79]: 0
```

```
In [80]: final_df.isna().sum()
```

```
Out[80]: Order Number      0
Line Item      0
Order Date     0
CustomerKey    0
StoreKey       0
ProductKey     0
Quantity       0
Currency Code  0
Product Name   0
Brand          0
Color          0
SubcategoryKey 0
Subcategory    0
CategoryKey    0
Category       0
Unit Cost      0
Unit Price     0
Gender         0
City           0
State          0
Zip Code       0
Country        0
Continent      0
Birthday       0
Age            0
ex_Date        0
ex_Currency    0
ex_Exchange    0
store_StoreKey 0
store_Country  0
store_State    0
store_Square Meters 0
store_Open Date 0
dtype: int64
```

Drop duplicate key columns

```
In [81]: final_df.drop(columns=['ex_Date', 'ex_Currency', 'store_StoreKey'], inplace=True)
```

```
In [82]: final_df.shape
```

```
Out[82]: (62884, 30)
```

In [83]: `final_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62884 entries, 0 to 62883
Data columns (total 30 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  datetime64[ns]
3   CustomerKey                           62884 non-null  int64
4   StoreKey                              62884 non-null  int64
5   ProductKey                            62884 non-null  int64
6   Quantity                              62884 non-null  int64
7   Currency Code                         62884 non-null  object
8   Product Name                          62884 non-null  object
9   Brand                                 62884 non-null  object
10  Color                                 62884 non-null  object
11  SubcategoryKey                        62884 non-null  int64
12  Subcategory                           62884 non-null  object
13  CategoryKey                           62884 non-null  int64
14  Category                              62884 non-null  object
15  Unit Cost                             62884 non-null  float64
16  Unit Price                            62884 non-null  float64
17  Gender                                62884 non-null  object
18  City                                  62884 non-null  object
19  State                                 62884 non-null  object
20  Zip Code                             62884 non-null  object
21  Country                              62884 non-null  object
22  Continent                             62884 non-null  object
23  Birthday                             62884 non-null  datetime64[ns]
24  Age                                   62884 non-null  int64
25  ex_Exchange                           62884 non-null  float64
26  store_Country                         62884 non-null  object
27  store_State                           62884 non-null  object
28  store_Square Meters                   62884 non-null  float64
29  store_Open Date                       62884 non-null  datetime64[ns]
dtypes: datetime64[ns](3), float64(4), int64(9), object(14)
memory usage: 14.9+ MB
```

Making required changes in column names (lower case ,renaming ,removing spaces and replace with "_") for easy handling in DB.

In [84]: `final_df.columns = final_df.columns.str.lower().str.replace(' ', '_')`

```
In [85]: final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62884 entries, 0 to 62883
Data columns (total 30 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  datetime64[ns]
3   customerkey                       62884 non-null  int64
4   storekey                          62884 non-null  int64
5   productkey                        62884 non-null  int64
6   quantity                          62884 non-null  int64
7   currency_code                     62884 non-null  object
8   product_name                      62884 non-null  object
9   brand                             62884 non-null  object
10  color                             62884 non-null  object
11  subcategorykey                    62884 non-null  int64
12  subcategory                       62884 non-null  object
13  categorykey                      62884 non-null  int64
14  category                          62884 non-null  object
15  unit_cost                        62884 non-null  float64
16  unit_price                       62884 non-null  float64
17  gender                           62884 non-null  object
18  city                             62884 non-null  object
19  state                            62884 non-null  object
20  zip_code                         62884 non-null  object
21  country                          62884 non-null  object
22  continent                        62884 non-null  object
23  birthday                         62884 non-null  datetime64[ns]
24  age                              62884 non-null  int64
25  ex_exchange                      62884 non-null  float64
26  store_country                    62884 non-null  object
27  store_state                      62884 non-null  object
28  store_square_meters              62884 non-null  float64
29  store_open_date                  62884 non-null  datetime64[ns]
dtypes: datetime64[ns](3), float64(4), int64(9), object(14)
memory usage: 14.9+ MB
```

```
In [86]: final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62884 entries, 0 to 62883
Data columns (total 30 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  datetime64[ns]
3   customerkey                           62884 non-null  int64
4   storekey                              62884 non-null  int64
5   productkey                            62884 non-null  int64
6   quantity                              62884 non-null  int64
7   currency_code                         62884 non-null  object
8   product_name                          62884 non-null  object
9   brand                                 62884 non-null  object
10  color                                 62884 non-null  object
11  subcategorykey                        62884 non-null  int64
12  subcategory                           62884 non-null  object
13  categorykey                           62884 non-null  int64
14  category                              62884 non-null  object
15  unit_cost                             62884 non-null  float64
16  unit_price                            62884 non-null  float64
17  gender                                 62884 non-null  object
18  city                                  62884 non-null  object
19  state                                 62884 non-null  object
20  zip_code                              62884 non-null  object
21  country                              62884 non-null  object
22  continent                             62884 non-null  object
23  birthday                             62884 non-null  datetime64[ns]
24  age                                   62884 non-null  int64
25  ex_exchange                           62884 non-null  float64
26  store_country                         62884 non-null  object
27  store_state                           62884 non-null  object
28  store_square_meters                   62884 non-null  float64
29  store_open_date                       62884 non-null  datetime64[ns]
dtypes: datetime64[ns](3), float64(4), int64(9), object(14)
memory usage: 14.9+ MB
```

In [87]: `final_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62884 entries, 0 to 62883
Data columns (total 30 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  datetime64[ns]
3   customerkey            62884 non-null  int64
4   storekey               62884 non-null  int64
5   productkey            62884 non-null  int64
6   quantity              62884 non-null  int64
7   currency_code         62884 non-null  object
8   product_name          62884 non-null  object
9   brand                 62884 non-null  object
10  color                 62884 non-null  object
11  subcategorykey        62884 non-null  int64
12  subcategory           62884 non-null  object
13  categorykey           62884 non-null  int64
14  category              62884 non-null  object
15  unit_cost             62884 non-null  float64
16  unit_price            62884 non-null  float64
17  gender                62884 non-null  object
18  city                  62884 non-null  object
19  state                 62884 non-null  object
20  zip_code              62884 non-null  object
21  country               62884 non-null  object
22  continent             62884 non-null  object
23  birthday              62884 non-null  datetime64[ns]
24  age                   62884 non-null  int64
25  ex_exchange           62884 non-null  float64
26  store_country         62884 non-null  object
27  store_state           62884 non-null  object
28  store_square_meters   62884 non-null  float64
29  store_open_date       62884 non-null  datetime64[ns]
dtypes: datetime64[ns](3), float64(4), int64(9), object(14)
memory usage: 14.9+ MB
```

5. Calculation Part

- Calculating cost of the products,
- Calculating price of the products,
- Profit/Loss Calculated in USD
- Calculations after applying exchange rates

In [88]: `print(final_df.columns)`

```
Index(['order_number', 'line_item', 'order_date', 'customerkey', 'storekey',
      'productkey', 'quantity', 'currency_code', 'product_name', 'brand',
      'color', 'subcategorykey', 'subcategory', 'categorykey', 'category',
      'unit_cost', 'unit_price', 'gender', 'city', 'state', 'zip_code',
      'country', 'continent', 'birthday', 'age', 'ex_exchange',
      'store_country', 'store_state', 'store_square_meters',
      'store_open_date'],
      dtype='object')
```

In [89]: `import pandas as pd`

```
# Assuming 'final_df' is your DataFrame and it already exists in your environment

# Renaming the column 'ex_exchange' to 'exchange_value'
final_df.rename(columns={'ex_exchange': 'exchange_value'}, inplace=True)

# Optionally, you can print the columns to confirm the change
print(final_df.columns)
```

```
Index(['order_number', 'line_item', 'order_date', 'customerkey', 'storekey',
      'productkey', 'quantity', 'currency_code', 'product_name', 'brand',
      'color', 'subcategorykey', 'subcategory', 'categorykey', 'category',
      'unit_cost', 'unit_price', 'gender', 'city', 'state', 'zip_code',
      'country', 'continent', 'birthday', 'age', 'exchange_value',
      'store_country', 'store_state', 'store_square_meters',
      'store_open_date'],
      dtype='object')
```

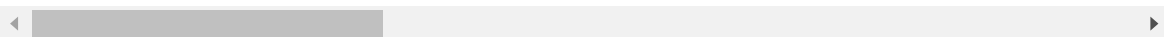
In [90]: `# Calculating line wise item profit/loss`
`# Calculating product cost in usd`
`final_df['total_cost_usd'] = final_df['quantity'] * final_df['unit_cost']`
`# Calculating total_price_usd`
`final_df['total_price_usd'] = final_df['quantity'] * final_df['unit_price']`
`# Calculating profit/loss`
`final_df['profit_loss_usd'] = final_df['total_price_usd'] - final_df['total_cost_usd']`
`# Calculating total_cost_org by applying exchange rate`
`final_df['total_cost_org'] = final_df['total_cost_usd'] * final_df['exchange_rate']`
`# Calculating total_price_org by applying exchange rate`
`final_df['total_price_org'] = final_df['total_price_usd'] * final_df['exchange_rate']`
`# Calculating profit/loss`
`final_df['profit_loss_org'] = final_df['total_price_org'] - final_df['total_cost_org']`

```
In [91]: final_df
```

```
Out[91]:
```

	order_number	line_item	order_date	customerkey	storekey	productkey	quantity	cu
0	366000	1	2016-01-01	265598	10	1304	1	
1	1749017	1	2019-10-15	265598	0	1619	1	
2	891000	2	2017-06-09	265598	9	174	1	
3	891000	1	2017-06-09	265598	9	385	2	
4	891000	4	2017-06-09	265598	9	87	1	
...
62879	1511040	1	2019-02-19	957765	40	757	6	
62880	1466016	1	2019-01-05	347907	9	2232	2	
62881	1625031	1	2019-06-13	725929	30	2308	2	
62882	1632012	1	2019-06-20	1292476	65	2331	2	
62883	1825024	1	2019-12-30	1546902	50	2479	5	

62884 rows × 36 columns

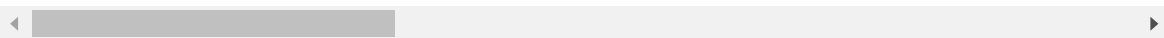


```
In [92]: final_df[final_df['profit_loss_org'] < 0]
```

```
Out[92]:
```

order_number	line_item	order_date	customerkey	storekey	productkey	quantity	currency
--------------	-----------	------------	-------------	----------	------------	----------	----------

0 rows × 36 columns



In [93]: final_df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 62884 entries, 0 to 62883
Data columns (total 36 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  datetime64[ns]
3   customerkey                       62884 non-null  int64
4   storekey                          62884 non-null  int64
5   productkey                        62884 non-null  int64
6   quantity                          62884 non-null  int64
7   currency_code                     62884 non-null  object
8   product_name                      62884 non-null  object
9   brand                             62884 non-null  object
10  color                             62884 non-null  object
11  subcategorykey                    62884 non-null  int64
12  subcategory                       62884 non-null  object
13  categorykey                      62884 non-null  int64
14  category                          62884 non-null  object
15  unit_cost                        62884 non-null  float64
16  unit_price                       62884 non-null  float64
17  gender                           62884 non-null  object
18  city                             62884 non-null  object
19  state                            62884 non-null  object
20  zip_code                         62884 non-null  object
21  country                          62884 non-null  object
22  continent                        62884 non-null  object
23  birthday                         62884 non-null  datetime64[ns]
24  age                             62884 non-null  int64
25  exchange_value                   62884 non-null  float64
26  store_country                    62884 non-null  object
27  store_state                      62884 non-null  object
28  store_square_meters              62884 non-null  float64
29  store_open_date                  62884 non-null  datetime64[ns]
30  total_cost_usd                   62884 non-null  float64
31  total_price_usd                  62884 non-null  float64
32  profit_loss_usd                  62884 non-null  float64
33  total_cost_org                   62884 non-null  float64
34  total_price_org                  62884 non-null  float64
35  profit_loss_org                  62884 non-null  float64
dtypes: datetime64[ns](3), float64(10), int64(9), object(14)
memory usage: 17.8+ MB
```


In [94]: `final_df.describe()`

Out[94]:

	order_number	line_item	customerkey	storekey	productkey	quantity
count	6.288400e+04	62884.000000	6.288400e+04	62884.000000	62884.000000	62884.000000
mean	1.430905e+06	2.164207	1.180797e+06	31.802144	1125.859344	3.144790
std	4.532963e+05	1.365170	5.859634e+05	22.978188	709.244010	2.256371
min	3.660000e+05	1.000000	3.010000e+02	0.000000	1.000000	1.000000
25%	1.121017e+06	1.000000	6.808580e+05	8.000000	437.000000	1.000000
50%	1.498016e+06	2.000000	1.261200e+06	37.000000	1358.000000	2.000000
75%	1.788010e+06	3.000000	1.686496e+06	53.000000	1650.000000	4.000000
max	2.243032e+06	7.000000	2.099937e+06	66.000000	2517.000000	10.000000

In [95]: `final_df.isnull().sum()`

Out[95]:

order_number	0
line_item	0
order_date	0
customerkey	0
storekey	0
productkey	0
quantity	0
currency_code	0
product_name	0
brand	0
color	0
subcategorykey	0
subcategory	0
categorykey	0
category	0
unit_cost	0
unit_price	0
gender	0
city	0
state	0
zip_code	0
country	0
continent	0
birthday	0
age	0
exchange_value	0
store_country	0
store_state	0
store_square_meters	0
store_open_date	0
total_cost_usd	0
total_price_usd	0
profit_loss_usd	0
total_cost_org	0
total_price_org	0
profit_loss_org	0
dtype: int64	

```
In [96]: final_df.duplicated().sum()
```

```
Out[96]: 0
```

6. Calculate age group of customers

```
In [97]: #Calculate age group of the customers based on their age  
final_df['age_group'] = pd.cut(final_df['age'], bins=[0, 20, 40, 60, 80, 100])  
  
#print cust_age and cust_age_group  
final_df[['age', 'age_group']].head()
```

```
Out[97]:
```

	age	age_group
0	53	40-60
1	53	40-60
2	53	40-60
3	53	40-60
4	53	40-60

In [98]: final_df

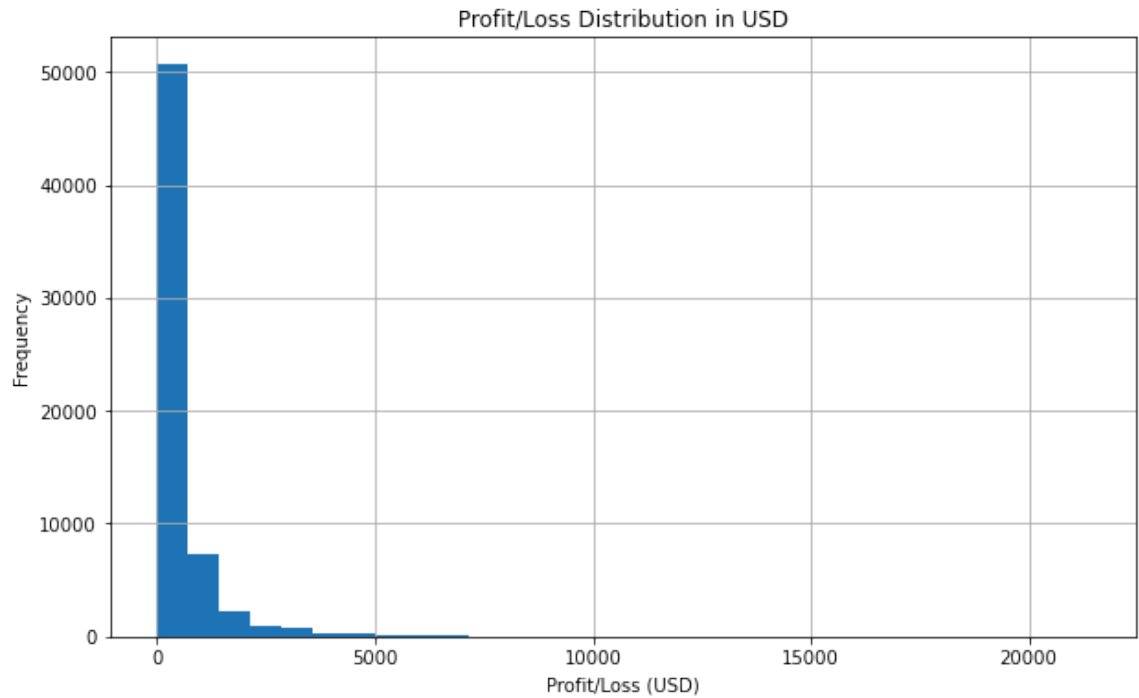
Out[98]:

	order_number	line_item	order_date	customerkey	storekey	productkey	quantity	cu
0	366000	1	2016-01-01	265598	10	1304	1	
1	1749017	1	2019-10-15	265598	0	1619	1	
2	891000	2	2017-06-09	265598	9	174	1	
3	891000	1	2017-06-09	265598	9	385	2	
4	891000	4	2017-06-09	265598	9	87	1	
...
62879	1511040	1	2019-02-19	957765	40	757	6	
62880	1466016	1	2019-01-05	347907	9	2232	2	
62881	1625031	1	2019-06-13	725929	30	2308	2	
62882	1632012	1	2019-06-20	1292476	65	2331	2	
62883	1825024	1	2019-12-30	1546902	50	2479	5	

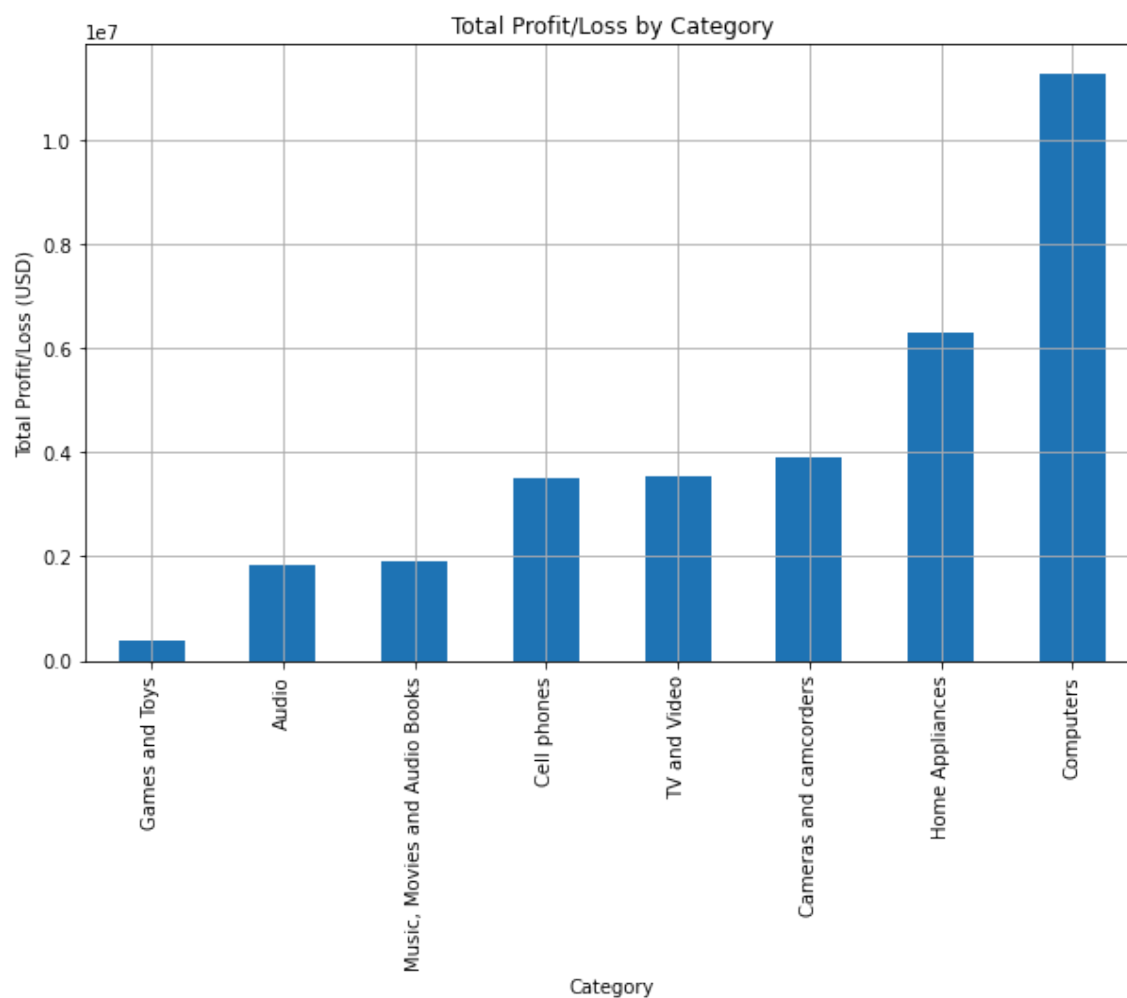
62884 rows × 37 columns



```
In [99]: # Profit/Loss Distribution
plt.figure(figsize=(10, 6))
plt.hist(final_df['profit_loss_usd'], bins=30)
plt.title('Profit/Loss Distribution in USD')
plt.xlabel('Profit/Loss (USD)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



```
In [100]: # Profit/Loss by Category
profit_loss_by_category = final_df.groupby('category')['profit_loss_usd'].sum()
plt.figure(figsize=(10, 6))
profit_loss_by_category.plot(kind='bar')
plt.title('Total Profit/Loss by Category')
plt.xlabel('Category')
plt.ylabel('Total Profit/Loss (USD)')
plt.grid(True)
plt.show()
```



```
In [101]: import pandas as pd
import matplotlib.pyplot as plt

# Convert 'order_date' to datetime type if it's not already
final_df['order_date'] = pd.to_datetime(final_df['order_date'])

# Set 'order_date' as the index of the DataFrame without dropping it from the DataFrame
final_df.set_index('order_date', inplace=True, drop=False)

# Now 'order_date' is both an index and a column in the DataFrame
# Resample the 'profit_loss_usd' by month and sum it
profit_loss_over_time = final_df['profit_loss_usd'].resample('M').sum()

# Plotting
plt.figure(figsize=(10, 6))
profit_loss_over_time.plot(color='green')
plt.title('Profit/Loss Over Time')
plt.xlabel('Date')
plt.ylabel('Profit/Loss (USD)')
plt.grid(True)
plt.show()
```



7. Loading the final_df into SQL DB

In [102]: `final_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 62884 entries, 2016-01-01 to 2019-12-30
Data columns (total 37 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  datetime64[ns]
3   customerkey                           62884 non-null  int64
4   storekey                              62884 non-null  int64
5   productkey                            62884 non-null  int64
6   quantity                              62884 non-null  int64
7   currency_code                         62884 non-null  object
8   product_name                          62884 non-null  object
9   brand                                 62884 non-null  object
10  color                                 62884 non-null  object
11  subcategorykey                        62884 non-null  int64
12  subcategory                           62884 non-null  object
13  categorykey                           62884 non-null  int64
14  category                              62884 non-null  object
15  unit_cost                             62884 non-null  float64
16  unit_price                            62884 non-null  float64
17  gender                                62884 non-null  object
18  city                                  62884 non-null  object
19  state                                 62884 non-null  object
20  zip_code                              62884 non-null  object
21  country                               62884 non-null  object
22  continent                             62884 non-null  object
23  birthday                              62884 non-null  datetime64[ns]
24  age                                    62884 non-null  int64
25  exchange_value                        62884 non-null  float64
26  store_country                         62884 non-null  object
27  store_state                           62884 non-null  object
28  store_square_meters                   62884 non-null  float64
29  store_open_date                       62884 non-null  datetime64[ns]
30  total_cost_usd                        62884 non-null  float64
31  total_price_usd                       62884 non-null  float64
32  profit_loss_usd                       62884 non-null  float64
33  total_cost_org                         62884 non-null  float64
34  total_price_org                       62884 non-null  float64
35  profit_loss_org                       62884 non-null  float64
36  age_group                             62884 non-null  category
dtypes: category(1), datetime64[ns](3), float64(10), int64(9), object(14)
memory usage: 17.8+ MB
```

In [103]: `final_df.to_sql(name= "dataspark_final_df",con= engine,if_exists="replace",`

Out[103]: -1

```
In [104]: sales.shape
```

```
Out[104]: (62884, 8)
```

```
In [105]: final_df.shape
```

```
Out[105]: (62884, 37)
```



```
In [106]: final_df.hist(figsize=(15,15))
```

```
Out[106]: array([[<AxesSubplot:title={'center':'order_number'}>,
<AxesSubplot:title={'center':'line_item'}>,
<AxesSubplot:title={'center':'order_date'}>,
<AxesSubplot:title={'center':'customerkey'}>,
<AxesSubplot:title={'center':'storekey'}>],
[<AxesSubplot:title={'center':'productkey'}>,
<AxesSubplot:title={'center':'quantity'}>,
<AxesSubplot:title={'center':'subcategorykey'}>,
<AxesSubplot:title={'center':'categorykey'}>,
<AxesSubplot:title={'center':'unit_cost'}>],
[<AxesSubplot:title={'center':'unit_price'}>,
<AxesSubplot:title={'center':'birthday'}>,
<AxesSubplot:title={'center':'age'}>,
<AxesSubplot:title={'center':'exchange_value'}>,
<AxesSubplot:title={'center':'store_square_meters'}>],
[<AxesSubplot:title={'center':'store_open_date'}>,
<AxesSubplot:title={'center':'total_cost_usd'}>,
<AxesSubplot:title={'center':'total_price_usd'}>,
<AxesSubplot:title={'center':'profit_loss_usd'}>,
<AxesSubplot:title={'center':'total_cost_org'}>],
[<AxesSubplot:title={'center':'total_price_org'}>,
<AxesSubplot:title={'center':'profit_loss_org'}>], <AxesSubplot:>,
<AxesSubplot:>], dtype=object)
```



```
In [107]: final_df.describe()
```

```
Out[107]:
```

	order_number	line_item	customerkey	storekey	productkey	quantity
count	6.288400e+04	62884.000000	6.288400e+04	62884.000000	62884.000000	62884.000000
mean	1.430905e+06	2.164207	1.180797e+06	31.802144	1125.859344	3.144790
std	4.532963e+05	1.365170	5.859634e+05	22.978188	709.244010	2.256371
min	3.660000e+05	1.000000	3.010000e+02	0.000000	1.000000	1.000000
25%	1.121017e+06	1.000000	6.808580e+05	8.000000	437.000000	1.000000
50%	1.498016e+06	2.000000	1.261200e+06	37.000000	1358.000000	2.000000
75%	1.788010e+06	3.000000	1.686496e+06	53.000000	1650.000000	4.000000
max	2.243032e+06	7.000000	2.099937e+06	66.000000	2517.000000	10.000000

```
In [108]: final_df.columns
```

```
Out[108]: Index(['order_number', 'line_item', 'order_date', 'customerkey', 'storekey',
                'productkey', 'quantity', 'currency_code', 'product_name', 'brand',
                'color', 'subcategorykey', 'subcategory', 'categorykey', 'category',
                'unit_cost', 'unit_price', 'gender', 'city', 'state', 'zip_code',
                'country', 'continent', 'birthday', 'age', 'exchange_value',
                'store_country', 'store_state', 'store_square_meters',
                'store_open_date', 'total_cost_usd', 'total_price_usd',
                'profit_loss_usd', 'total_cost_org', 'total_price_org',
                'profit_loss_org', 'age_group'],
                dtype='object')
```

```
In [109]: final_df.to_csv("dataspark.csv", index=False)
```

```
In [110]: data = pd.read_csv("dataspark.csv")
```

```
In [113]: data.columns
```

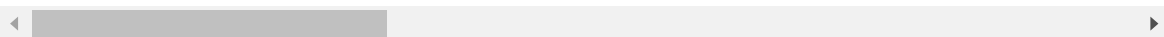
```
Out[113]: Index(['order_number', 'line_item', 'order_date', 'customerkey', 'storekey',
                'productkey', 'quantity', 'currency_code', 'product_name', 'brand',
                'color', 'subcategorykey', 'subcategory', 'categorykey', 'category',
                'unit_cost', 'unit_price', 'gender', 'city', 'state', 'zip_code',
                'country', 'continent', 'birthday', 'age', 'exchange_value',
                'store_country', 'store_state', 'store_square_meters',
                'store_open_date', 'total_cost_usd', 'total_price_usd',
                'profit_loss_usd', 'total_cost_org', 'total_price_org',
                'profit_loss_org', 'age_group'],
                dtype='object')
```

In [114]: data

Out[114]:

	order_number	line_item	order_date	customerkey	storekey	productkey	quantity	cu
0	366000	1	2016-01-01	265598	10	1304	1	
1	1749017	1	2019-10-15	265598	0	1619	1	
2	891000	2	2017-06-09	265598	9	174	1	
3	891000	1	2017-06-09	265598	9	385	2	
4	891000	4	2017-06-09	265598	9	87	1	
...
62879	1511040	1	2019-02-19	957765	40	757	6	
62880	1466016	1	2019-01-05	347907	9	2232	2	
62881	1625031	1	2019-06-13	725929	30	2308	2	
62882	1632012	1	2019-06-20	1292476	65	2331	2	
62883	1825024	1	2019-12-30	1546902	50	2479	5	

62884 rows × 37 columns



In []: