

```
In [ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
sales_data = pd.read_excel("/content/drive/MyDrive/ECOMM DATA.xlsx")
```

```
In [ ]: sales_data.columns
```

```
Out[82]: Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
               'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',
               'Postal Code', 'Market', 'Region', 'Product ID', 'Category',
               'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
               'Profit', 'Shipping Cost', 'Order Priority'],
              dtype='object')
```

```
In [ ]: sales_data.shape
```

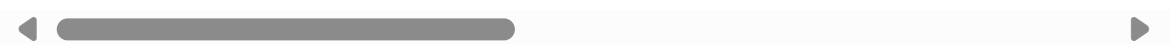
```
Out[83]: (51290, 24)
```

In []: sales_data

Out[84]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City
0	32298	CA-2012-124891	2012-07-31	2012-07-31	Same Day	RH-19495	Rick Hansen	Consumer	New York City
1	26341	IN-2013-77878	2013-02-05	2013-02-07	Second Class	JR-16210	Justin Ritter	Corporate	Wollongong
2	25330	IN-2013-71249	2013-10-17	2013-10-18	First Class	CR-12730	Craig Reiter	Consumer	Brisbane
3	13524	ES-2013-1579342	2013-01-28	2013-01-30	First Class	KM-16375	Katherine Murray	Home Office	Berlin
4	47221	SG-2013-4320	2013-11-05	2013-11-06	Same Day	RH-9495	Rick Hansen	Consumer	Dakar
...
51285	29002	IN-2014-62366	2014-06-19	2014-06-19	Same Day	KE-16420	Katrina Edelman	Corporate	Kure
51286	35398	US-2014-102288	2014-06-20	2014-06-24	Standard Class	ZC-21910	Zuschuss Carroll	Consumer	Houston
51287	40470	US-2013-155768	2013-12-02	2013-12-02	Same Day	LB-16795	Laurel Beltran	Home Office	Oxnard
51288	9596	MX-2012-140767	2012-02-18	2012-02-22	Standard Class	RB-19795	Ross Baird	Home Office	Valinhos
51289	6147	MX-2012-134460	2012-05-22	2012-05-26	Second Class	MC-18100	Mick Crebagga	Consumer	Tipitapa

51290 rows × 24 columns



Information about Dataset

```
In [ ]: sales_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Row ID                51290 non-null  int64
1   Order ID              51290 non-null  object
2   Order Date            51290 non-null  datetime64[ns]
3   Ship Date             51290 non-null  datetime64[ns]
4   Ship Mode              51290 non-null  object
5   Customer ID           51290 non-null  object
6   Customer Name         51290 non-null  object
7   Segment               51290 non-null  object
8   City                  51290 non-null  object
9   State                 51290 non-null  object
10  Country               51290 non-null  object
11  Postal Code           9994 non-null   float64
12  Market                51290 non-null  object
13  Region                51290 non-null  object
14  Product ID            51290 non-null  object
15  Category              51290 non-null  object
16  Sub-Category          51290 non-null  object
17  Product Name          51290 non-null  object
18  Sales                  51290 non-null  float64
19  Quantity              51290 non-null  int64
20  Discount              51290 non-null  float64
21  Profit                51290 non-null  float64
22  Shipping Cost         51290 non-null  float64
23  Order Priority         51290 non-null  object
dtypes: datetime64[ns](2), float64(5), int64(2), object(15)
memory usage: 9.4+ MB
```

```
In [ ]: sales_data.describe()
```

Out[86]:

	Row ID	Order Date	Ship Date	Postal Code	Sales	
count	51290.00000	51290	51290	9994.000000	51290.000000	5129
mean	25645.50000	2013-05-11 21:26:49.155781120	2013-05-15 20:42:42.745174528	55190.379428	246.490581	
min	1.00000	2011-01-01 00:00:00	2011-01-03 00:00:00	1040.000000	0.444000	
25%	12823.25000	2012-06-19 00:00:00	2012-06-23 00:00:00	23223.000000	30.758625	
50%	25645.50000	2013-07-08 00:00:00	2013-07-12 00:00:00	56430.500000	85.053000	
75%	38467.75000	2014-05-22 00:00:00	2014-05-26 00:00:00	90008.000000	251.053200	
max	51290.00000	2014-12-31 00:00:00	2015-01-07 00:00:00	99301.000000	22638.480000	1
std	14806.29199	NaN	NaN	32063.693350	487.565361	

Checking any missing values in Dataset

```
In [ ]: sales_data.isna().any()
```

```
Out[87]: Row ID           False
Order ID           False
Order Date         False
Ship Date          False
Ship Mode          False
Customer ID        False
Customer Name      False
Segment           False
City              False
State             False
Country           False
Postal Code        True
Market            False
Region            False
Product ID        False
Category           False
Sub-Category       False
Product Name       False
Sales             False
Quantity          False
Discount           False
Profit            False
Shipping Cost      False
Order Priority     False
dtype: bool
```

we have Postal Code column with 41296 missing values

```
In [ ]: sales_data.isna().sum()
```

```
Out[88]: Row ID           0
Order ID           0
Order Date         0
Ship Date          0
Ship Mode          0
Customer ID        0
Customer Name      0
Segment           0
City              0
State             0
Country           0
Postal Code        41296
Market            0
Region            0
Product ID        0
Category           0
Sub-Category       0
Product Name       0
Sales             0
Quantity          0
Discount           0
Profit            0
Shipping Cost      0
Order Priority     0
dtype: int64
```

```
In [ ]: sales_data['Postal Code']
```

```
Out[89]: 0          10024.0  
         1           NaN  
         2           NaN  
         3           NaN  
         4           NaN  
         ...  
        51285          NaN  
        51286       77095.0  
        51287       93030.0  
        51288          NaN  
        51289          NaN  
        Name: Postal Code, Length: 51290, dtype: float64
```

```
In [ ]: sales_data['Postal Code'].unique()
```

```
Out[90]: array([10024.,      nan, 95823., 28027., 22304., 42420., 60610., 90008.,
79109., 93727., 10009., 27217., 55407., 92646., 98115., 32303.,
23223., 30318., 49201., 19134., 89015., 98105., 8701., 90045.,
48205., 22801., 19120., 2149., 92037., 53711., 90805., 10035.,
77036., 10701., 73071., 43130., 28205., 89031., 90049., 19711.,
2908., 11561., 94122., 43229., 90032., 48227., 23464., 7960.,
94110., 43055., 41042., 65807., 47905., 35810., 31907., 78207.,
14701., 46203., 48234., 53132., 68104., 92704., 36608., 22153.,
18018., 98226., 2920., 70506., 60623., 44052., 84043., 98103.,
65203., 75007., 10011., 90004., 77095., 30328., 44105., 78664.,
76106., 3820., 32216., 94521., 30076., 45373., 19140., 85323.,
45014., 1852., 80219., 21044., 46060., 92804., 27604., 2886.,
60653., 48911., 92374., 94601., 5408., 74403., 23320., 98198.,
53209., 19143., 93101., 14609., 91776., 45231., 13601., 28314.,
85705., 33180., 27834., 81001., 40214., 43615., 89431., 22980.,
94109., 60505., 77070., 35630., 48104., 46142., 85023., 92105.,
33012., 30062., 80229., 40324., 77041., 40475., 74133., 33801.,
94513., 84062., 28806., 61107., 85301., 75081., 68025., 92683.,
28540., 99207., 21215., 58103., 80013., 19805., 6824., 90036.,
79907., 33311., 6040., 1841., 94526., 43302., 98026., 60201.,
71111., 33614., 53142., 47201., 92592., 92307., 75080., 36116.,
78745., 94533., 84057., 23602., 2038., 77581., 80501., 89115.,
7060., 1915., 14215., 27405., 97030., 89502., 37042., 29203.,
74012., 46226., 80906., 73120., 59405., 55044., 80027., 30080.,
56301., 75217., 22204., 54915., 97206., 6450., 37211., 52001.,
44107., 72209., 60540., 99301., 82001., 92024., 50315., 75220.,
94086., 44134., 87401., 85234., 11572., 77506., 23434., 80525.,
43017., 37918., 48640., 45503., 61604., 39212., 7109., 3301.,
64055., 78521., 78577., 91505., 32725., 38109., 27707., 8861.,
43402., 33319., 20735., 95123., 77340., 19013., 38301., 55113.,
98059., 10550., 90278., 72401., 80134., 83704., 76063., 37167.,
35244., 77705., 20016., 95928., 91104., 73034., 62521., 18103.,
85281., 72701., 34741., 84604., 36830., 90503., 33065., 78041.,
60035., 35601., 61761., 97756., 97477., 76017., 6708., 83201.,
47374., 38401., 84106., 85224., 60126., 80004., 29501., 33317.,
55369., 13021., 90301., 60174., 91767., 28052., 85345., 87105.,
75023., 54302., 6457., 91911., 92503., 12180., 92020., 78501.,
44256., 48126., 85254., 93030., 63122., 85204., 98031., 33142.,
46614., 48066., 48187., 42071., 66212., 73505., 7501., 63376.,
90712., 54601., 30188., 26003., 38671., 97301., 94591., 93010.,
92691., 55125., 60016., 95661., 7055., 48127., 83301., 85364.,
29464., 28403., 87124., 48183., 71203., 92630., 60090., 19601.,
92345., 92404., 32712., 2151., 67212., 97405., 93309., 44060.,
75051., 77642., 44312., 80020., 46350., 60089., 60098., 28110.,
55901., 13501., 32839., 90660., 98502., 95687., 97224., 95207.,
11550., 64118., 37604., 92054., 48237., 54703., 17403., 60076.,
60543., 93905., 92627., 54880., 90604., 95037., 60188., 79424.,
33021., 71603., 7011., 37064., 4401., 46544., 92563., 56560.,
93454., 88220., 77803., 30605., 33710., 2169., 17602., 7090.,
76706., 75061., 71854., 91761., 70601., 57103., 92530., 10801.,
72032., 39401., 11520., 59601., 32137., 95336., 78539., 92553.,
7601., 13440., 33445., 33024., 6010., 59801., 33023., 23666.,
96003., 75701., 1040., 95695., 84020., 37087., 7017., 8360.,
91730., 76903., 48180., 63116., 52402., 33178., 37421., 24153.,
76021., 93277., 37075., 77573., 57701., 60423., 60068., 6360.,
66502., 75056., 32771., 77301., 98042., 66062., 91941., 93534.,
32114., 61701., 20852., 61032., 14304., 67846., 60067., 29406.,
60025., 75019., 20877., 77590., 44240., 34952., 31088., 48185.,
94403., 19901., 6460., 29483., 77840., 42104., 27893., 78550.,
98661., 84107., 49505., 95610., 71901., 55433., 2138., 88001.,
63301., 78666., 83605., 43123., 48307., 27511., 76117., 2895.,
```

```

33437., 75002., 45011., 52302., 79762., 48146., 46514., 62301.,
60462., 83642., 31204., 1752., 49423., 80022., 37620., 92677.,
7036., 61821., 54401., 92236., 80122., 84084., 70065., 60440.,
33068., 48601., 78415., 32935., 88101., 2740., 95351., 75104.,
7050., 55106., 92253., 28601., 97123., 39503., 20707., 27534.,
33030., 47150., 47401., 95051., 35401., 30344., 55016., 55124.,
55122., 1453., 37130., 95240., 80112., 91360., 7002., 65109.,
45040., 50322., 27360., 2148., 98270., 1810., 47362., 33134.,
75150., 94061., 48310., 92672., 72756., 11757., 77571., 90640.,
33407., 29730., 22901., 6484., 57401., 76051., 21740., 85635.,
92399., 33433., 84321., 42301., 52601., 98632., 87505., 75043.,
75034., 94509., 53214., 92025., 61832., 53186., 38134., 3060.,
59715., 48858., 46368., 8302., 97504., 60477., 16602., 53081.,
98006., 6810., 95616., 98052., 4240., 80634., 48073., 59102.,
27514., 33063., 94568., 44221., 77489., 80538., 77536., 33161.,
32503., 77520., 50701., 86442., 32127., 60004., 8901., 68701.,
52240., 68801., 72762., 76248., 60302., 8401., 60441., 84041.,
44035., 33458., 98208., 98002., 32174., 93405., 79605., 83501.])

```

In []:

```

postal_codes = sales_data['Postal Code']

# Attempt to convert the "Postal Code" column to numeric format
try:
    postal_codes_numeric = pd.to_numeric(postal_codes)
    print("Conversion to numeric format successful.")
except ValueError as e:
    print("Error encountered during conversion to numeric format:")
    print(e)

```

Conversion to numeric format successful.

In []:

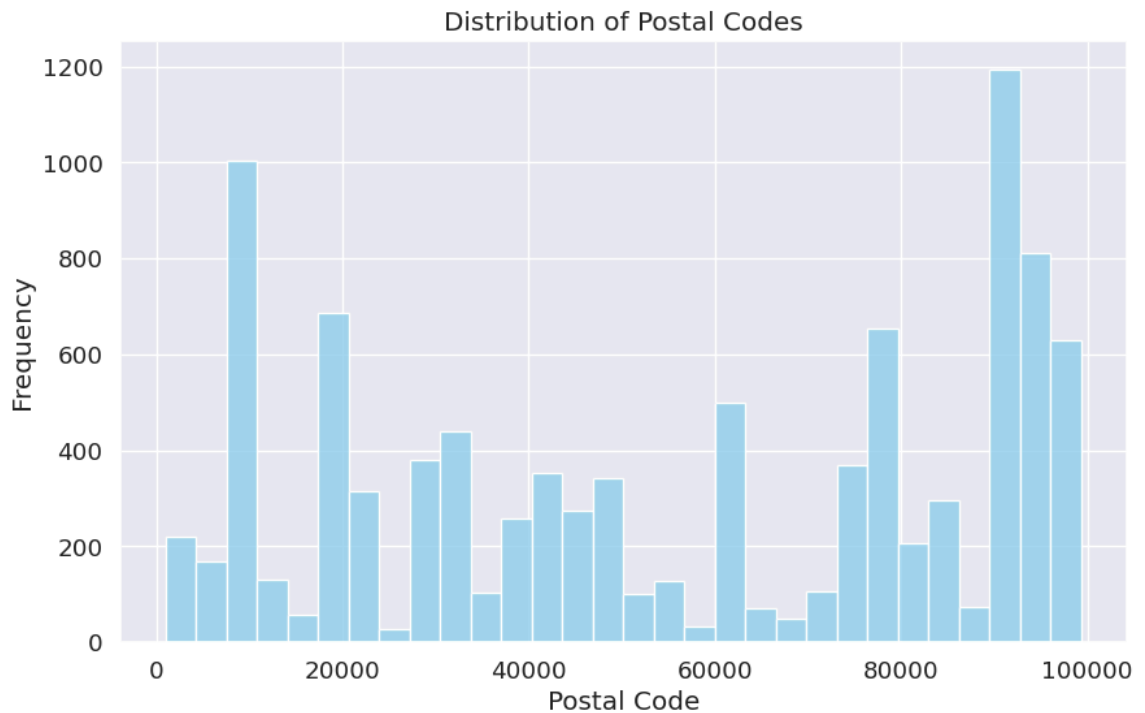
```

# Convert the values in the "Postal Code" column to numeric format
sales_data['Postal Code'] = pd.to_numeric(sales_data['Postal Code'])

```


In []:

```
# Plot distribution of postal codes
plt.figure(figsize=(10, 6))
sns.histplot(postal_codes, bins=30, color='skyblue')
plt.xlabel('Postal Code')
plt.ylabel('Frequency')
plt.title('Distribution of Postal Codes')
plt.show()
```



In []: sales_data['Postal Code']

```
Out[95]: 0      10024.0
         1         NaN
         2         NaN
         3         NaN
         4         NaN
         ...
        51285        NaN
        51286    77095.0
        51287    93030.0
        51288         NaN
        51289         NaN
        Name: Postal Code, Length: 51290, dtype: float64
```

```
In [ ]: # Fill missing values with the mode
mode_postal_code = sales_data['Postal Code'].mode()[0]
sales_data['Postal Code'].fillna(mode_postal_code, inplace=True)
```

In []: sales_data['Postal Code'].isna().sum() *#cleaned the column*

```
Out[99]: 0
```

Checking all columns in sales_Data

Row ID column

```
In [ ]: sales_data['Row ID'] #row_id
```

```
Out[100]: 0          32298
          1          26341
          2          25330
          3          13524
          4          47221
          ...
          51285       29002
          51286       35398
          51287       40470
          51288         9596
          51289         6147
          Name: Row ID, Length: 51290, dtype: int64
```

```
In [ ]: print(sales_data['Row ID'].unique())
        print(sales_data['Row ID'].nunique())
```

```
[32298 26341 25330 ... 40470 9596 6147]
51290
```

Order_ID column

```
In [ ]: sales_data['Order ID'].dtype
```

```
Out[13]: dtype('O')
```

```
In [ ]: sales_data['Order ID']
```

```
Out[23]: 0          CA-2012-124891
          1          IN-2013-77878
          2          IN-2013-71249
          3          ES-2013-1579342
          4          SG-2013-4320
          ...
          51285       IN-2014-62366
          51286       US-2014-102288
          51287       US-2013-155768
          51288       MX-2012-140767
          51289       MX-2012-134460
          Name: Order ID, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Order ID'].unique())
        print(sales_data['Order ID'].nunique())
```

```
['CA-2012-124891' 'IN-2013-77878' 'IN-2013-71249' ... 'IN-2014-72327'
 'IN-2014-57662' 'MX-2012-134460']
25035
```

Order_ID Column - (Graphical Representation)

In []:

```
# Get the unique values and their counts
unique_ids, counts = sales_data['Order ID'].value_counts().index, sales_data['Order ID'].value_counts().values

# Define colors for bars
colors = ['skyblue', 'salmon', 'lightgreen', 'lightcoral', 'lightblue',
          'orange', 'yellow', 'green', 'purple', 'pink']

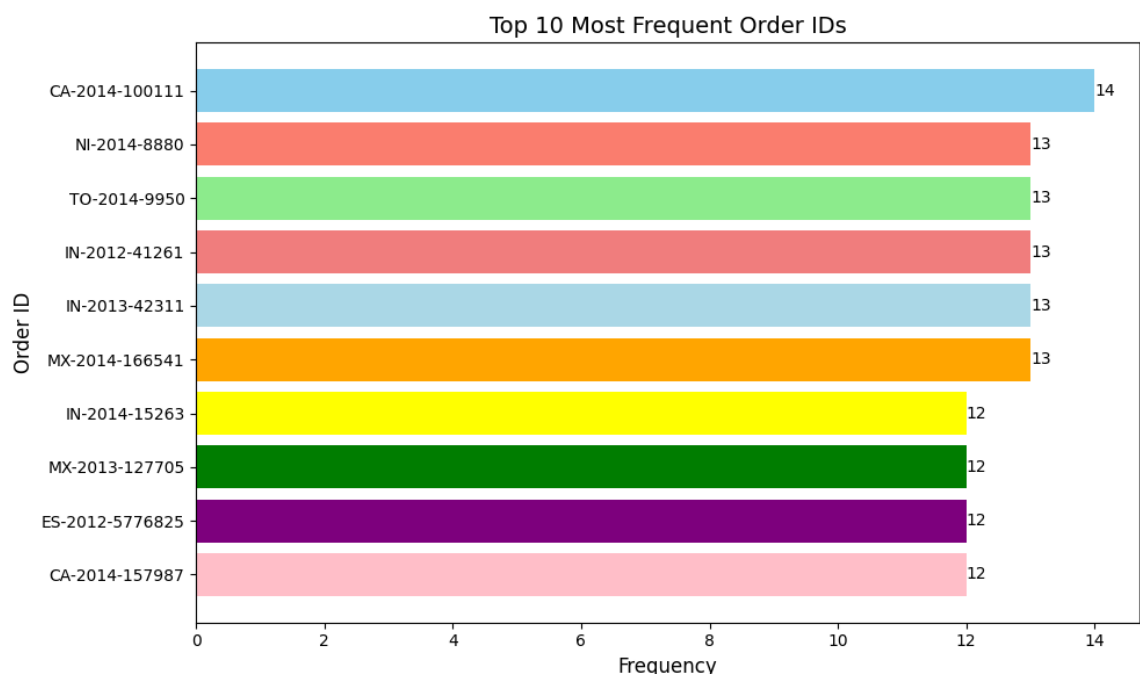
# Plot the bar plot
plt.figure(figsize=(10, 6))
bars = plt.barh(unique_ids[:10], counts[:10], color=colors)

# Add labels and title
plt.xlabel('Frequency', fontsize=12)
plt.ylabel('Order ID', fontsize=12)
plt.title('Top 10 Most Frequent Order IDs', fontsize=14)

# Add frequency labels on each bar
for bar, count in zip(bars, counts[:10]):
    plt.text(bar.get_width(), bar.get_y() + bar.get_height() / 2, f'{count}',
             va='center', ha='left', fontsize=10, color='black')

# Invert y-axis to display the highest frequency at the top
plt.gca().invert_yaxis()

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



Order Date Column

```
In [ ]: sales_data['Order Date']
```

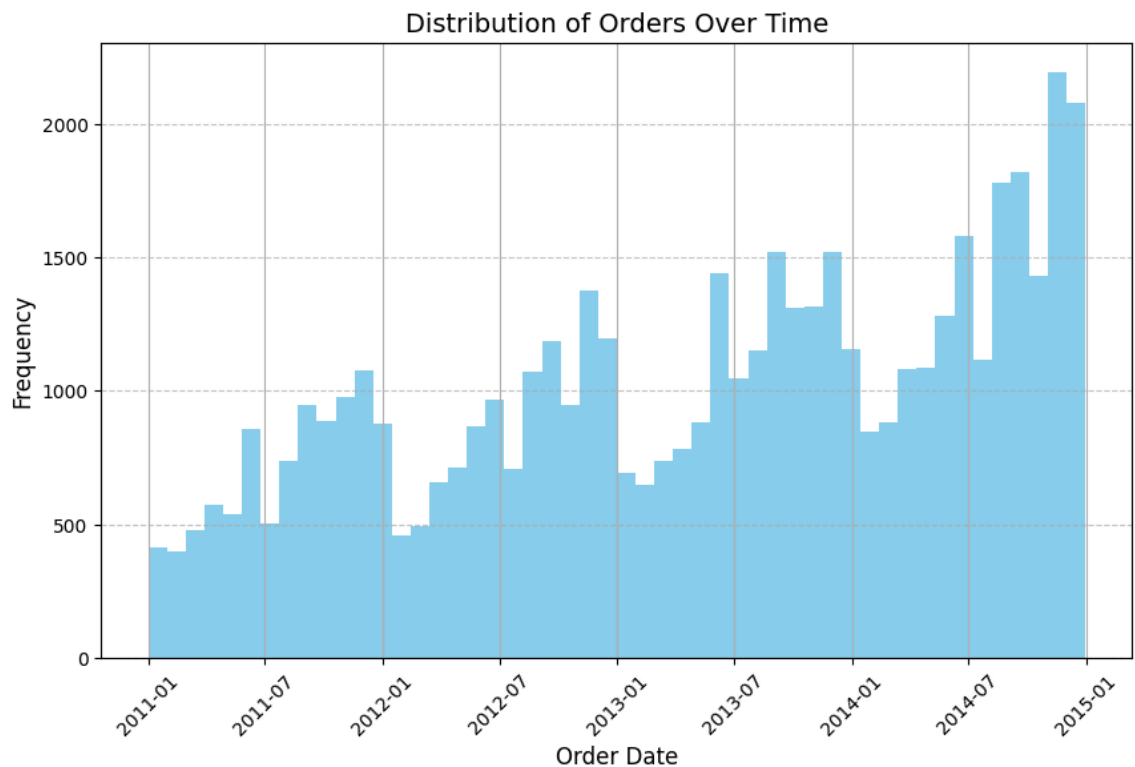
```
Out[16]: 0      2012-07-31
          1      2013-02-05
          2      2013-10-17
          3      2013-01-28
          4      2013-11-05
          ...
          51285   2014-06-19
          51286   2014-06-20
          51287   2013-12-02
          51288   2012-02-18
          51289   2012-05-22
          Name: Order Date, Length: 51290, dtype: datetime64[ns]
```

```
In [ ]: print(sales_data['Order Date'].unique())
         print(sales_data['Order Date'].nunique())
```

```
<DatetimeArray>
['2012-07-31 00:00:00', '2013-02-05 00:00:00', '2013-10-17 00:00:00',
 '2013-01-28 00:00:00', '2013-11-05 00:00:00', '2013-06-28 00:00:00',
 '2011-11-07 00:00:00', '2012-04-14 00:00:00', '2014-10-14 00:00:00',
 '2012-01-28 00:00:00',
 ...
 '2014-01-12 00:00:00', '2012-07-29 00:00:00', '2012-07-15 00:00:00',
 '2012-08-19 00:00:00', '2011-03-27 00:00:00', '2011-06-12 00:00:00',
 '2012-07-08 00:00:00', '2013-07-07 00:00:00', '2012-05-27 00:00:00',
 '2011-02-06 00:00:00']
Length: 1430, dtype: datetime64[ns]
1430
```

In []:

```
# Plot the histogram of order dates
plt.figure(figsize=(10, 6))
sales_data['Order Date'].hist(bins=50, color='skyblue')
plt.xlabel('Order Date', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.title('Distribution of Orders Over Time', fontsize=14)
plt.xticks(rotation=45) # Rotate x-axis labels for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7) # Add grid lines for better
plt.show()
```



Ship Date Column

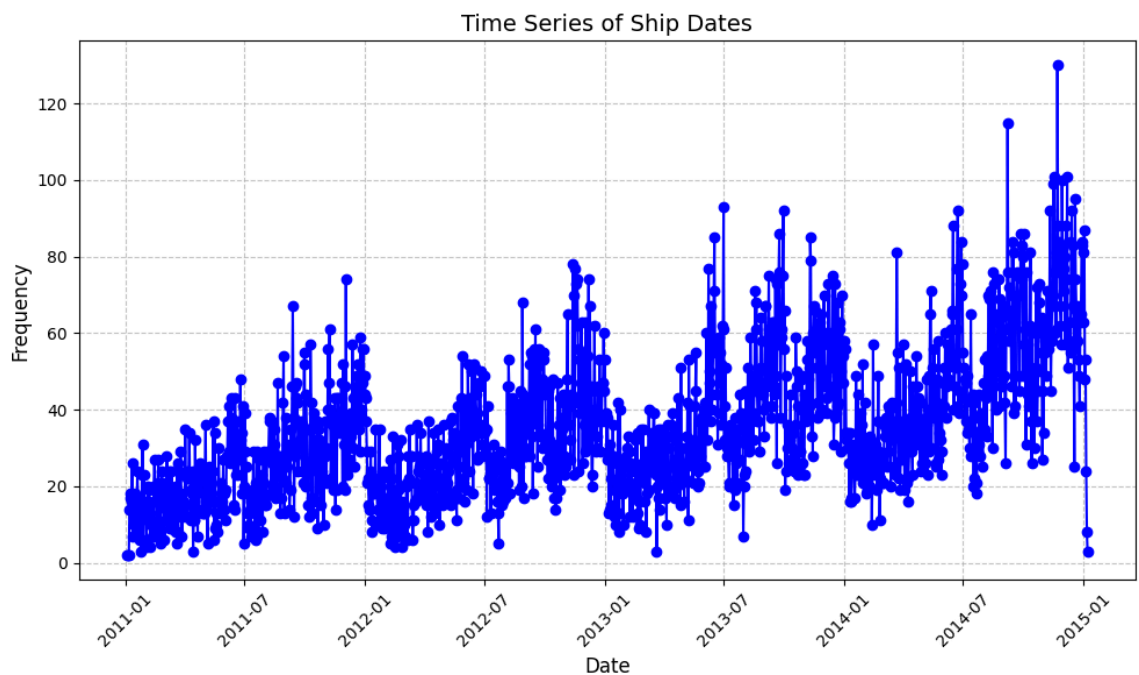
In []: sales_data['Ship Date']

```
Out[19]: 0      2012-07-31
          1      2013-02-07
          2      2013-10-18
          3      2013-01-30
          4      2013-11-06
          ...
          51285   2014-06-19
          51286   2014-06-24
          51287   2013-12-02
          51288   2012-02-22
          51289   2012-05-26
          Name: Ship Date, Length: 51290, dtype: datetime64[ns]
```

```
In [ ]: print(sales_data['Ship Date'].unique())
print(sales_data['Ship Date'].nunique())
```

```
<DatetimeArray>
['2012-07-31 00:00:00', '2013-02-07 00:00:00', '2013-10-18 00:00:00',
 '2013-01-30 00:00:00', '2013-11-06 00:00:00', '2013-07-01 00:00:00',
 '2011-11-09 00:00:00', '2012-04-18 00:00:00', '2014-10-21 00:00:00',
 '2012-01-31 00:00:00',
...
 '2011-07-11 00:00:00', '2011-02-23 00:00:00', '2011-01-25 00:00:00',
 '2015-01-07 00:00:00', '2011-02-08 00:00:00', '2012-01-24 00:00:00',
 '2012-02-15 00:00:00', '2012-07-23 00:00:00', '2012-04-08 00:00:00',
 '2011-01-05 00:00:00']
Length: 1464, dtype: datetime64[ns]
1464
```

```
In [ ]: # Plot the time series of ship dates as a line plot
plt.figure(figsize=(10, 6))
plt.plot(sales_data['Ship Date'].value_counts().sort_index(), marker='o', c
plt.xlabel('Date', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.title('Time Series of Ship Dates', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7) # Add grid lines for better visu
plt.xticks(rotation=45) # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



Ship Mode column

```
In [ ]: sales_data['Ship Mode']
```

```
Out[22]: 0          Same Day
          1      Second Class
          2      First Class
          3      First Class
          4          Same Day
          ...
          51285      Same Day
          51286      Standard Class
          51287      Same Day
          51288      Standard Class
          51289      Second Class
          Name: Ship Mode, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Ship Mode'].unique())
         print(sales_data['Ship Mode'].nunique())
```

```
['Same Day' 'Second Class' 'First Class' 'Standard Class']
4
```

In []:

```
# Set Seaborn style
sns.set(style="whitegrid")

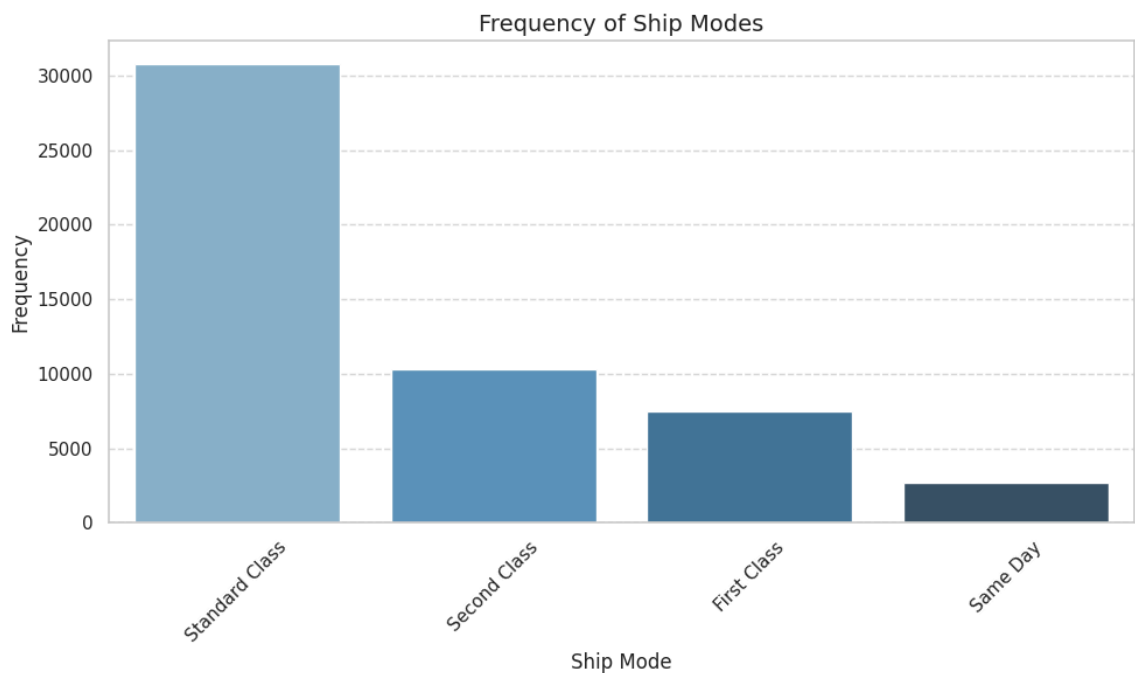
# Count the frequency of each ship mode
ship_mode_counts = sales_data['Ship Mode'].value_counts()

# Plot the bar chart using Seaborn
plt.figure(figsize=(10, 6))
sns.barplot(x=ship_mode_counts.index, y=ship_mode_counts.values, palette="Blues_d")
plt.title('Frequency of Ship Modes', fontsize=14)
plt.xlabel('Ship Mode', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

<ipython-input-24-3f784ab302dc>:9: FutureWarning:

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

```
sns.barplot(x=ship_mode_counts.index, y=ship_mode_counts.values, palette="Blues_d")
```



Customer ID column


```
In [ ]: sales_data['Customer ID']
```

```
Out[25]: 0      RH-19495
          1      JR-16210
          2      CR-12730
          3      KM-16375
          4      RH-9495
          ...
          51285   KE-16420
          51286   ZC-21910
          51287   LB-16795
          51288   RB-19795
          51289   MC-18100
          Name: Customer ID, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Customer ID'].unique())
        print(sales_data['Customer ID'].nunique())
```

```
['RH-19495' 'JR-16210' 'CR-12730' ... 'RC-9825' 'MG-7890' 'ZC-11910']
1590
```

In []:

```
# Set Seaborn style
sns.set(style="whitegrid")

# Count the frequency of each customer ID
customer_id_counts = sales_data['Customer ID'].value_counts()

# Plot the bar chart using Seaborn
plt.figure(figsize=(12, 6))
barplot = sns.barplot(x=customer_id_counts.index[:10], y=customer_id_counts)
plt.title('Top 10 Most Frequent Customer IDs', fontsize=14)
plt.xlabel('Customer ID', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.xticks(rotation=45)

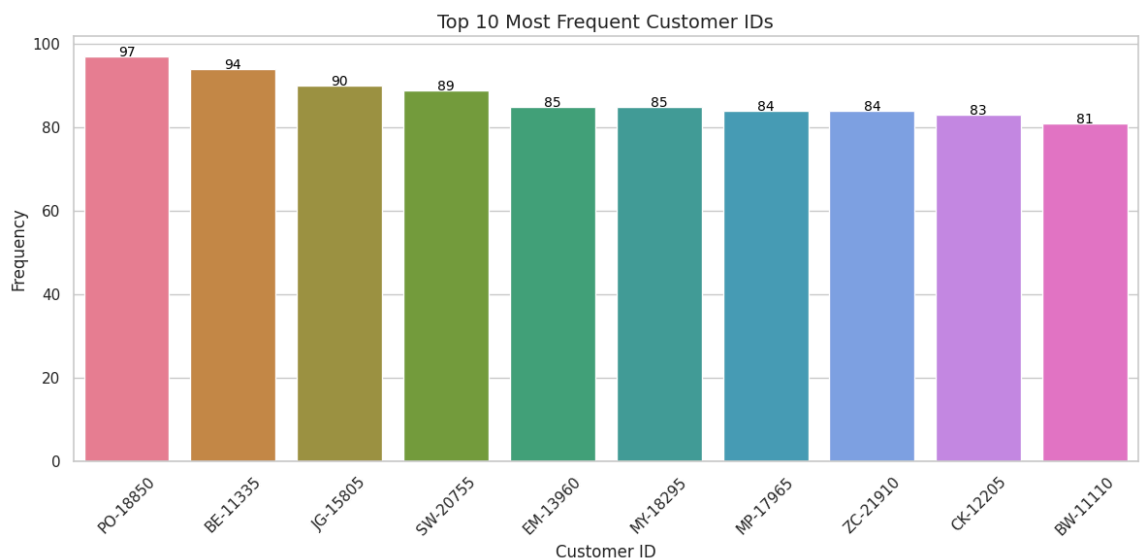
# Adding annotations
for index, value in enumerate(customer_id_counts.values[:10]):
    barplot.text(index, value, str(value), ha="center", fontsize=10, color=

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

<ipython-input-27-007cf4003afa>:9: FutureWarning:

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

```
barplot = sns.barplot(x=customer_id_counts.index[:10], y=customer_id_counts.values[:10], palette="husl")
```



Customer Name Column

```
In [ ]: sales_data['Customer Name']
```

```
Out[28]: 0          Rick Hansen
          1          Justin Ritter
          2          Craig Reiter
          3      Katherine Murray
          4          Rick Hansen
          ...
          51285      Katrina Edelman
          51286      Zuschuss Carroll
          51287      Laurel Beltran
          51288          Ross Baird
          51289      Mick Crebagga
          Name: Customer Name, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Customer Name'].unique())
        print(sales_data['Customer Name'].nunique()) #795 unique customers
```

```
['Rick Hansen' 'Justin Ritter' 'Craig Reiter' 'Katherine Murray'
 'Jim Mitchum' 'Toby Swindell' 'Mick Brown' 'Jane Waco' 'Joseph Holt'
 'Greg Maxwell' 'Anthony Jacobs' 'Magdelene Morse' 'Vicky Freymann'
 'Peter Fuller' 'Ben Peterman' 'Thomas Boland' 'Patrick Jones' 'Jim Sin
 k'
 'Ritsa Hightower' 'Ann Blume' 'Sue Ann Reed' 'Jason Klamczynski'
 'Laurel Beltran' 'Naresj Patel' 'Valerie Dominguez' 'Phillip Breyer'
 'Eugene Barchas' 'Karen Ferguson' 'Benjamin Patterson' 'Rick Reed'
 'Bill Shonely' 'Joel Eaton' 'Dave Poirier' 'Nora Preis' 'Aaron Hawkins'
 'Darrin Martin' 'Grant Thornton' 'Patrick O'Donnell' 'Dan Lawera'
 'Joy Bell-' 'Barry Franz' 'Vivek Grady' 'Greg Tran' 'Zuschuss Carroll'
 'Sanjit Chand' 'Ellis Ballard' 'Arthur Prichep' 'Scott Williamson'
 'John Huston' 'Lena Creighton' 'Trudy Glocke' 'Harold Ryan'
 'Deirdre Greer' 'Sheri Gordon' 'Fred Hopkins' 'Guy Phonely'
 'Mitch Webber' 'Patrick O'Brill' 'Chuck Sachs' 'Keith Dawkins'
 'Michael Stewart' 'Kimberly Carter' 'Denny Blanton' 'Jonathan Doherty'
 'Dave Kipp' 'Cari Sayre' 'Evan Minnotte' 'Dianna Wilson'
 'Alan Schoenberger' 'Shui Tom' 'Barry Weirich' 'Laura Armstrong'
 'Aimee Bixby' 'Christopher Martinez' 'Bobby Elias' 'Sam Zeldin'
 'Raymond Messel' 'Harry Green' 'Andy Reiter' 'Tom Prescott']
```

```
In [ ]: !pip install squarify
```

```
Collecting squarify
  Downloading squarify-0.4.3-py3-none-any.whl (4.3 kB)
Installing collected packages: squarify
Successfully installed squarify-0.4.3
```

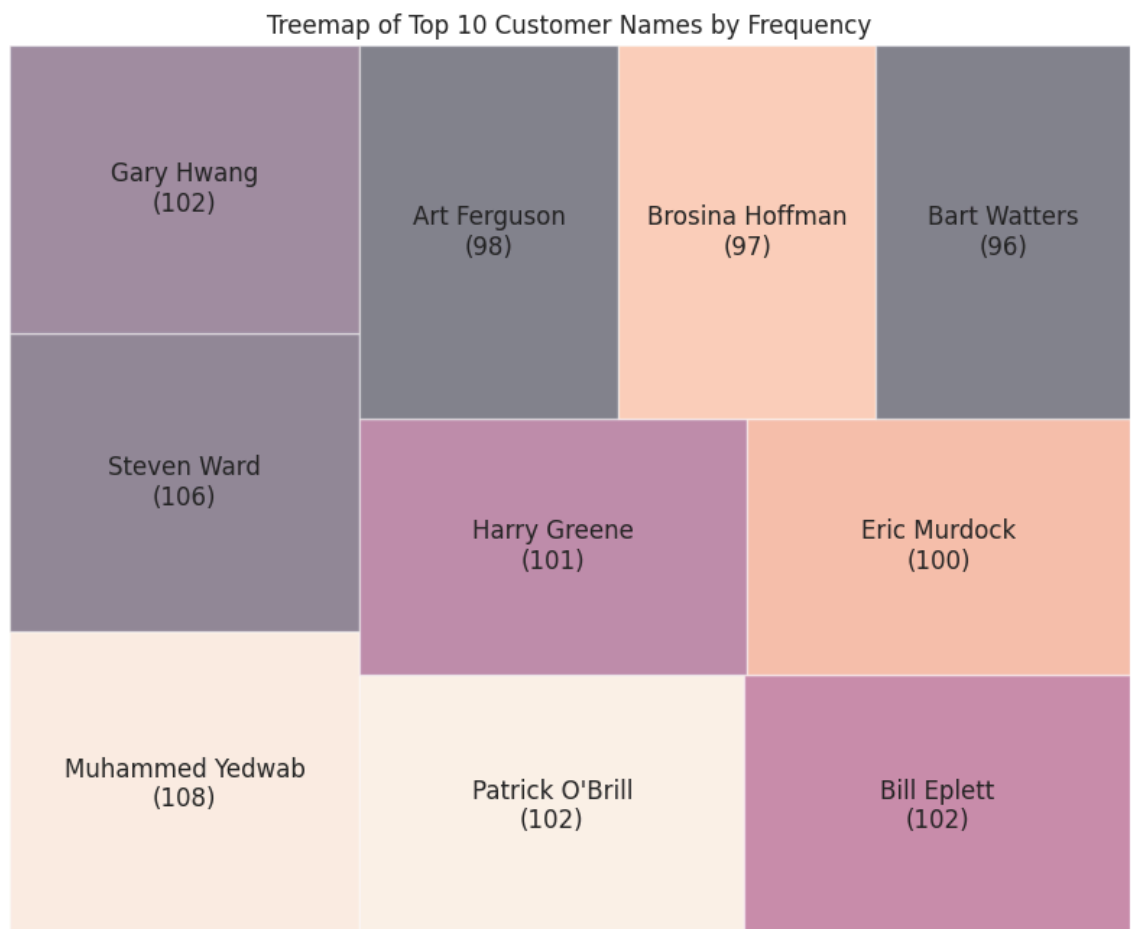
```
In [ ]: import squarify

# Get the frequency of each customer name
customer_name_counts = sales_data['Customer Name'].value_counts()

# Selecting only the top 10 customers
top_10_customers = customer_name_counts.head(10)

# Prepare the labels with frequencies
labels = [f'{name}\n({count})' for name, count in zip(top_10_customers.index, top_10_customers.values)]

# Plotting the treemap with labeled frequencies
plt.figure(figsize=(10, 8))
squarify.plot(sizes=top_10_customers.values, label=labels, alpha=0.5)
plt.axis('off')
plt.title('Treemap of Top 10 Customer Names by Frequency')
plt.show()
```



Segment column

```
In [ ]: sales_data['Segment']
```

```
Out[32]: 0          Consumer
          1          Corporate
          2          Consumer
          3      Home Office
          4          Consumer
          ...
          51285      Corporate
          51286      Consumer
          51287      Home Office
          51288      Home Office
          51289      Consumer
          Name: Segment, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Segment'].unique())
          print(sales_data['Segment'].nunique())

['Consumer' 'Corporate' 'Home Office']
3
```

In []:

```
# Set the font scale for better readability
sns.set(font_scale=1.2)

# Get the unique values and their counts for the Segment column
segment_counts = sales_data['Segment'].value_counts()

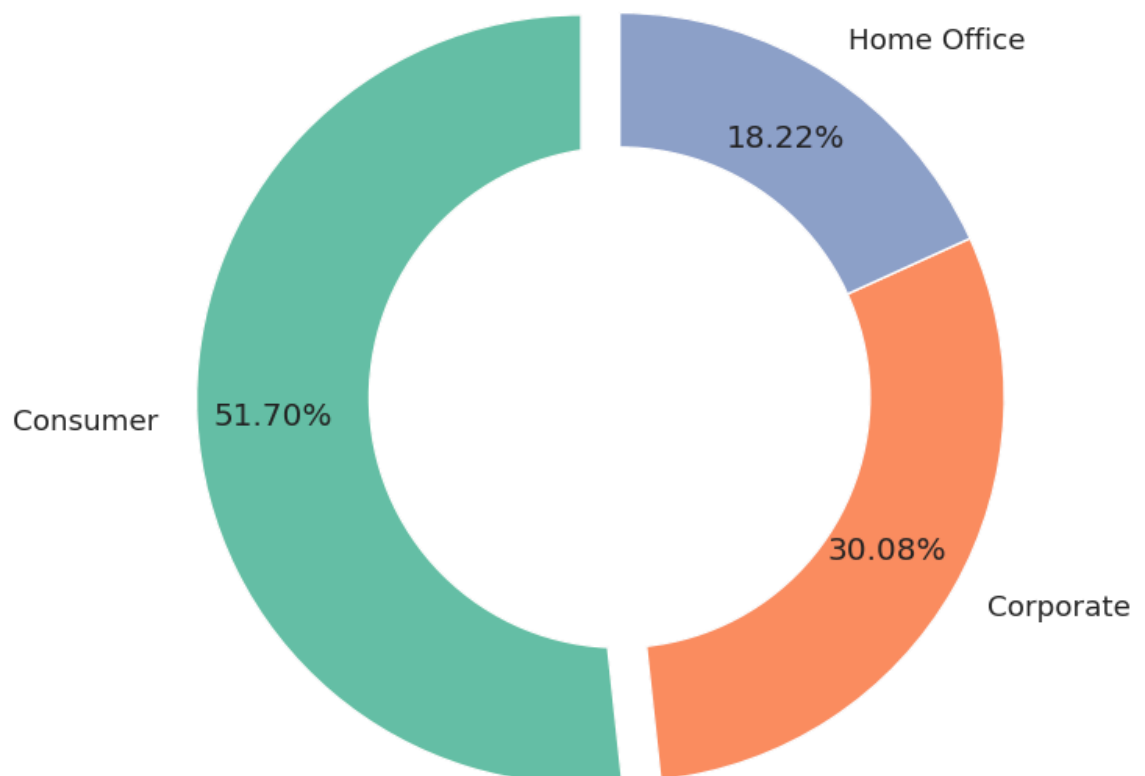
# Define explode values
explode = [0.1 if seg == 'Consumer' else 0 for seg in segment_counts.index]

# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(x=segment_counts, labels=segment_counts.index, colors=sns.color_palette('magma', 3))

# Add a hole in the pie
hole = plt.Circle((0, 0), 0.65, facecolor='white')
plt.gcf().gca().add_artist(hole)

plt.title('Distribution of Segments')
plt.show()
```

Distribution of Segments



***City column ***

```
In [ ]: sales_data['City']
```

```
Out[35]: 0      New York City
          1      Wollongong
          2      Brisbane
          3      Berlin
          4      Dakar
          ...
          51285     Kure
          51286     Houston
          51287     Oxnard
          51288     Valinhos
          51289     Tipitapa
          Name: City, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['City'].unique())
         print(sales_data['City'].nunique())
```

```
['New York City' 'Wollongong' 'Brisbane' ... 'Abilene' 'Felahiye'
 'Victoria Falls']
3636
```

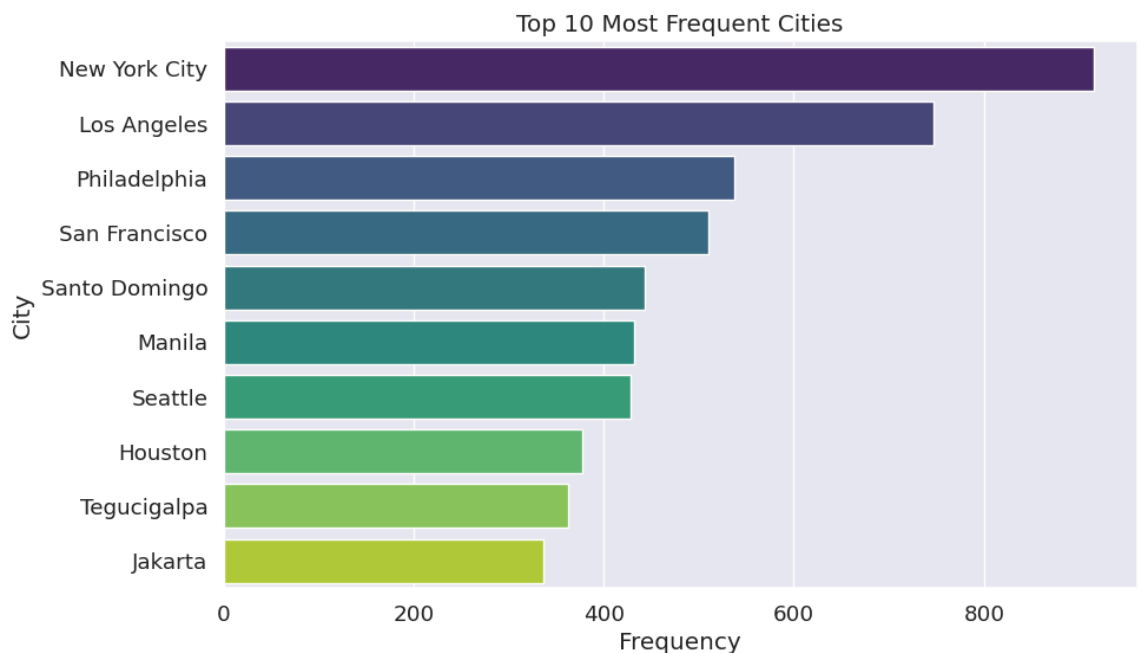
```
In [ ]: # Get the unique values and their counts for the City column
city_counts = sales_data['City'].value_counts()

# Plotting the count plot
plt.figure(figsize=(10, 6))
sns.countplot(y='City', data=sales_data, order=city_counts.index[:10], palette=
plt.title('Top 10 Most Frequent Cities')
plt.xlabel('Frequency')
plt.ylabel('City')
plt.show()
```

<ipython-input-37-881d807890c8>:6: FutureWarning:

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

```
sns.countplot(y='City', data=sales_data, order=city_counts.index[:10], palette='viridis')
```



State column

```
In [ ]: sales_data['State']
```

```
Out[38]: 0          New York
1      New South Wales
2      Queensland
3          Berlin
4          Dakar
...
51285      Hiroshima
51286          Texas
51287      California
51288      São Paulo
51289      Managua
Name: State, Length: 51290, dtype: object
```



```
In [ ]: print(sales_data['State'].unique())
print(sales_data['State'].nunique())
```

```
['New York' 'New South Wales' 'Queensland' ... 'Manicaland' 'Kabarole'
'Matabeleland North']
1094
```

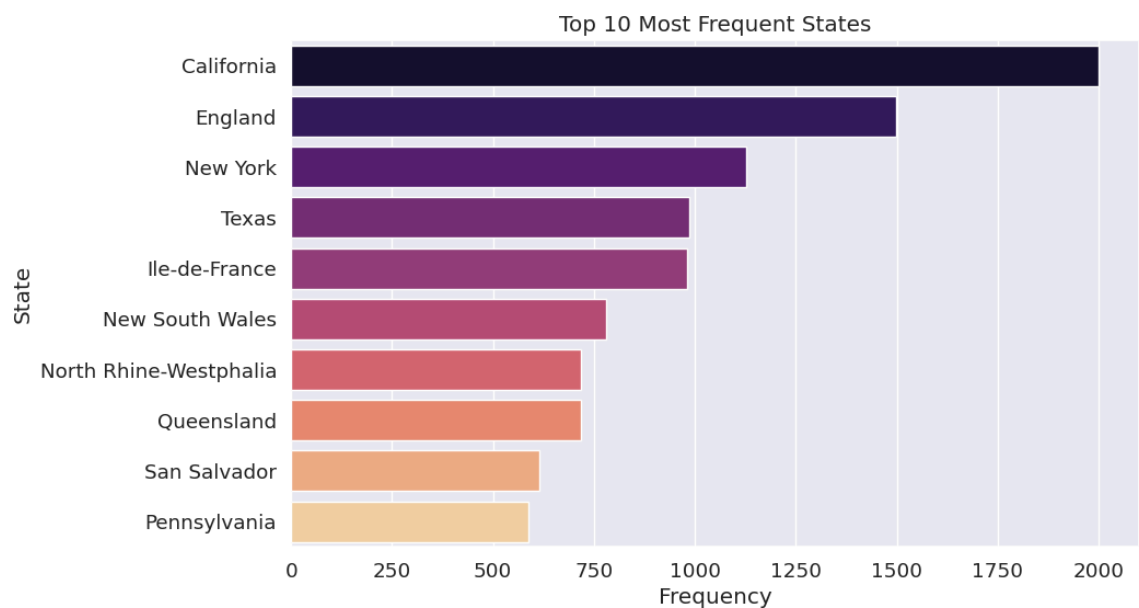
```
In [ ]: state_counts = sales_data['State'].value_counts()
```

```
# Plotting the count plot
plt.figure(figsize=(10, 6))
sns.countplot(y='State', data=sales_data, order=state_counts.index[:10], pa
plt.title('Top 10 Most Frequent States')
plt.xlabel('Frequency')
plt.ylabel('State')
plt.show()
```

<ipython-input-40-88c50122b8d7>:5: FutureWarning:

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

```
sns.countplot(y='State', data=sales_data, order=state_counts.index[:10],
palette='magma')
```



Country column

```
In [ ]: sales_data['Country']
```

```
Out[41]: 0      United States
          1      Australia
          2      Australia
          3      Germany
          4      Senegal
          ...
          51285    Japan
          51286    United States
          51287    United States
          51288      Brazil
          51289    Nicaragua
          Name: Country, Length: 51290, dtype: object
```

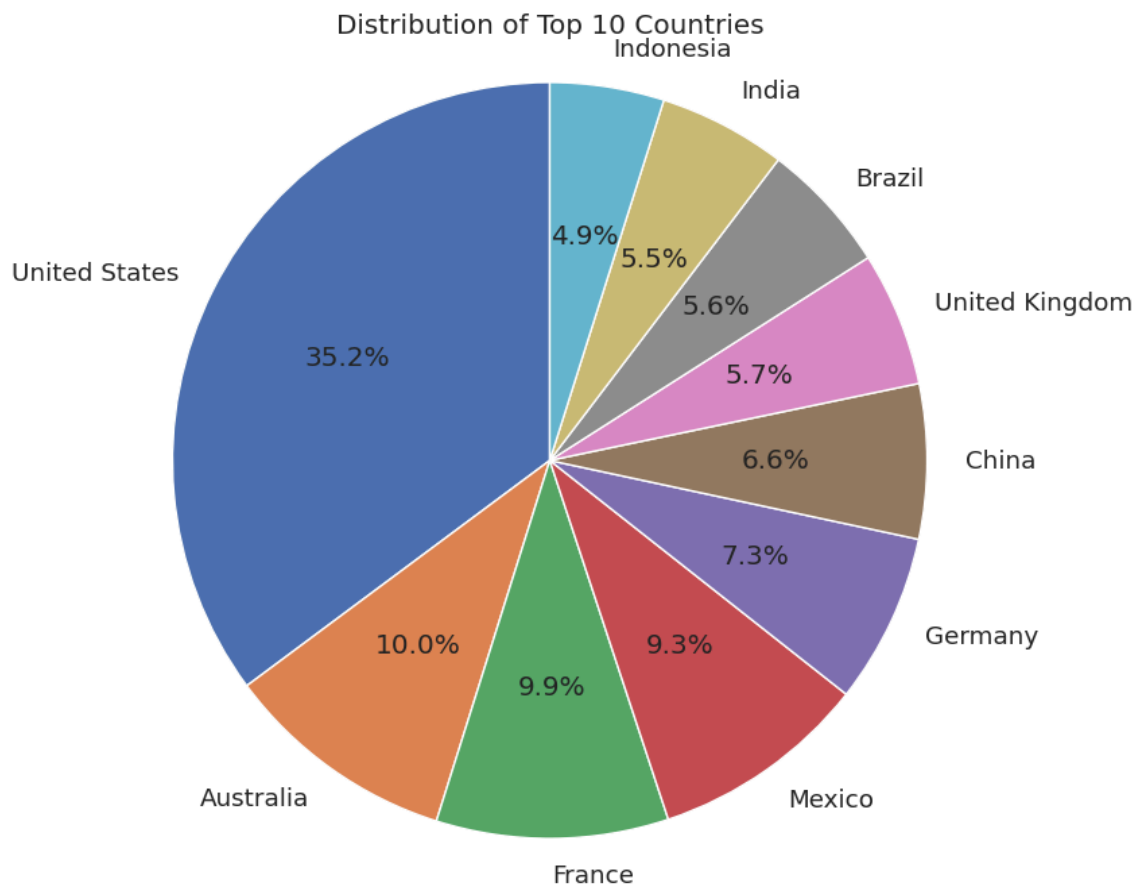
```
In [ ]: print(sales_data['Country'].unique())
        print(sales_data['Country'].nunique())
```

```
['United States' 'Australia' 'Germany' 'Senegal' 'New Zealand'
 'Afghanistan' 'Saudi Arabia' 'Brazil' 'China' 'France' 'Italy' 'Tanzania'
 'Poland' 'United Kingdom' 'Mexico' 'El Salvador' 'Taiwan' 'India'
 'Dominican Republic' 'Democratic Republic of the Congo' 'Indonesia'
 'Uruguay' 'Iran' 'Mozambique' 'Bangladesh' 'Spain' 'Ukraine' 'Nicaragua'
 'Morocco' 'Canada' 'Philippines' 'Austria' 'Colombia' 'Netherlands'
 'Malaysia' 'Ecuador' 'Thailand' 'Somalia' 'Guatemala' 'Belarus'
 'Cambodia' 'South Africa' 'Japan' 'Russia' 'Egypt' 'Azerbaijan'
 'Lithuania' 'Argentina' 'Lesotho' 'Vietnam' 'Cuba' 'Romania' 'Turkey'
 'Cameroon' 'Hungary' 'Singapore' 'Angola' 'Belgium' 'Pakistan' 'Finland'
 'Ghana' 'Zambia' 'Iraq' 'Liberia' 'Georgia' 'Switzerland' 'Albania'
 'Chad' 'Montenegro' 'Namibia' 'Portugal' 'Madagascar' 'Sweden'
 'Myanmar (Burma)' 'Jamaica' 'Qatar' 'Republic of the Congo' 'Norway'
 'Algeria' 'South Korea' 'Nigeria' 'Estonia' 'Cote d'Ivoire' 'Honduras'
 'Paraguay' 'Czech Republic' 'Central African Republic' 'Benin' 'Bolivia'
 'Chile' 'Martinique' 'Syria' 'Lebanon' 'Kenya' 'Mali' 'Libya' 'Venezuela'
 'Trinidad and Tobago' 'Ireland' 'Bulgaria' 'Panama' 'Israel' 'Haiti'
 'Barbados' 'Slovenia' 'Togo' 'Mauritania' 'Guinea' 'Rwanda' 'Denmark'
 'Niger' 'Papua New Guinea' 'Mongolia' 'Sudan' 'Peru' 'Sierra Leone'
 'Bosnia and Herzegovina' 'Guinea-Bissau' 'Djibouti' 'Tunisia' 'Croatia'
 'Hong Kong' 'Nepal' 'Guadeloupe' 'Kyrgyzstan' 'Zimbabwe' 'Uzbekistan'
 'South Sudan' 'Gabon' 'Bahrain' 'Yemen' 'Jordan' 'United Arab Emirates'
 'Moldova' 'Swaziland' 'Turkmenistan' 'Kazakhstan' 'Ethiopia' 'Uganda'
 'Slovakia' 'Sri Lanka' 'Tajikistan' 'Burundi' 'Macedonia' 'Eritrea'
 'Equatorial Guinea' 'Armenia']
```

147

```
In [ ]: country_counts = sales_data['Country'].value_counts().head(10)

# Plotting the pie chart
plt.figure(figsize=(10, 8))
sns.set(font_scale=1.2)
sns.color_palette("tab10")
plt.pie(country_counts, labels=country_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Top 10 Countries')
plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()
```



Market Column

```
In [ ]: sales_data['Market']
```

```
Out[44]: 0      US
1      APAC
2      APAC
3      EU
4      Africa
...
51285   APAC
51286   US
51287   US
51288   LATAM
51289   LATAM
Name: Market, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Market'].unique())
print(sales_data['Market'].nunique())
```

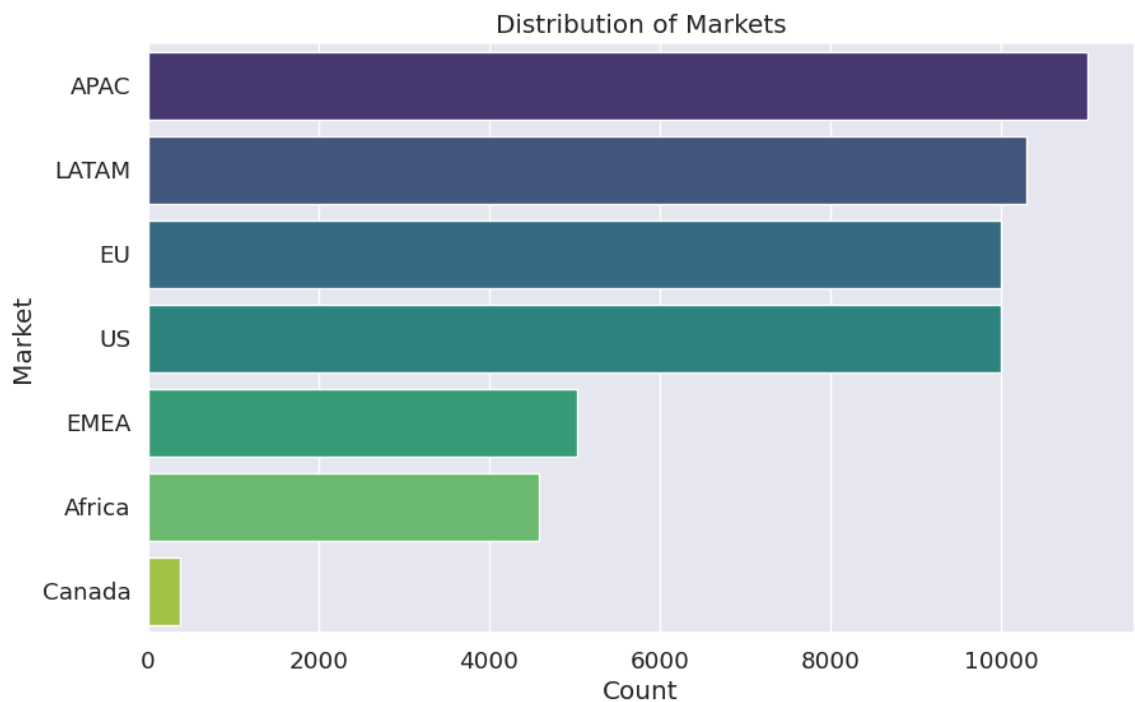
```
['US' 'APAC' 'EU' 'Africa' 'EMEA' 'LATAM' 'Canada']
7
```

```
In [ ]: plt.figure(figsize=(10, 6))
sns.countplot(data=sales_data, y='Market', order=sales_data['Market'].value_counts().index)
plt.title('Distribution of Markets')
plt.xlabel('Count')
plt.ylabel('Market')
plt.show()
```

<ipython-input-46-9b1a6bffcbb0>:2: FutureWarning:

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

```
sns.countplot(data=sales_data, y='Market', order=sales_data['Market'].value_counts().index, palette='viridis')
```



Region Column

```
In [ ]: sales_data['Region']
```

```
Out[47]: 0          East
          1      Oceania
          2      Oceania
          3      Central
          4      Africa
          ...
          51285  North Asia
          51286      Central
          51287          West
          51288          South
          51289      Central
          Name: Region, Length: 51290, dtype: object
```

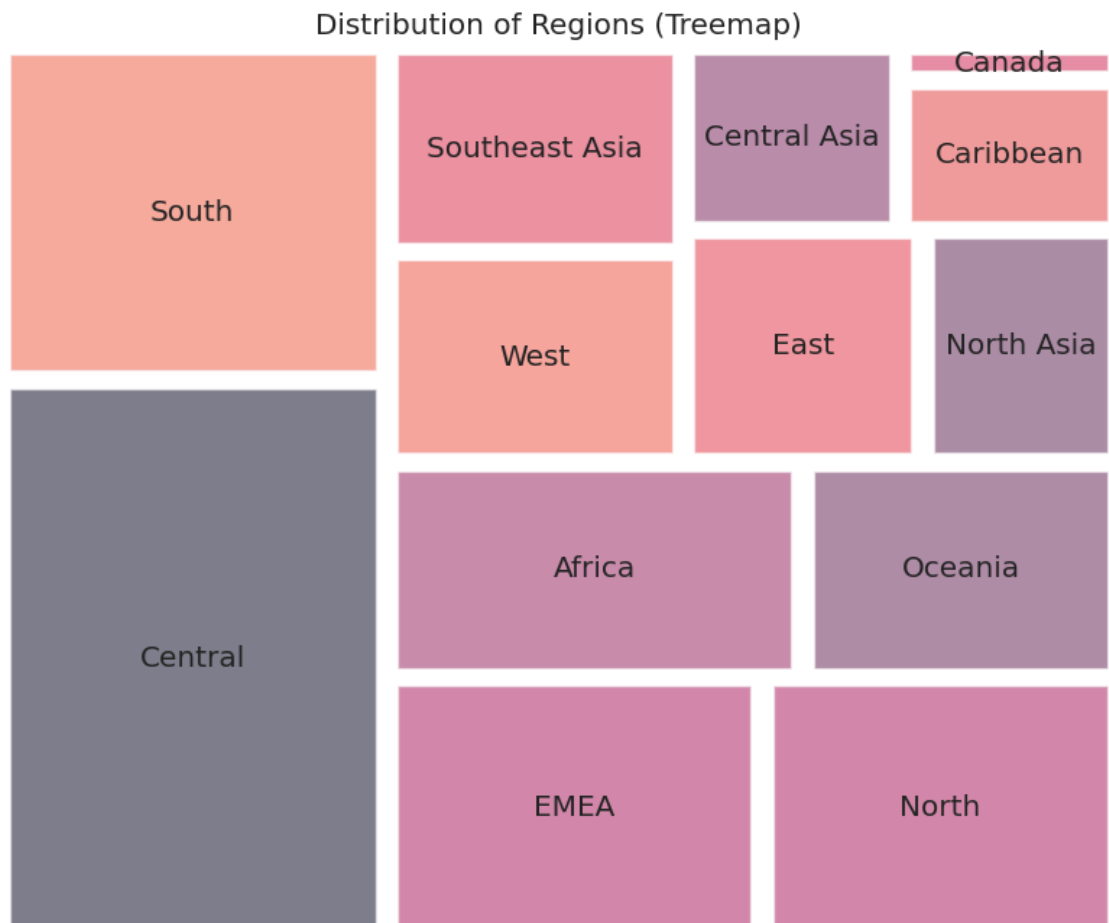
```
In [ ]: print(sales_data['Region'].unique())
         print(sales_data['Region'].nunique())
```

```
['East' 'Oceania' 'Central' 'Africa' 'West' 'South' 'Central Asia' 'EMEA'
 'North Asia' 'North' 'Caribbean' 'Southeast Asia' 'Canada']
```

13

```
In [ ]: # Get the counts for each region
region_counts = sales_data['Region'].value_counts()

# Create a squarify plot
plt.figure(figsize=(10, 8))
squarify.plot(sizes=region_counts, label=region_counts.index, alpha=0.5, padding=5)
plt.title('Distribution of Regions (Treemap)')
plt.axis('off') # Turn off axis
plt.show()
```



***Product ID Column ***

```
In [ ]: sales_data['Product ID']
```

```
Out[50]: 0      TEC-AC-10003033
1      FUR-CH-10003950
2      TEC-PH-10004664
3      TEC-PH-10004583
4      TEC-SHA-10000501
...
51285  OFF-FA-10000746
51286  OFF-AP-10002906
51287  OFF-EN-10001219
51288  OFF-BI-10000806
51289  OFF-PA-10004155
Name: Product ID, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Product ID'].unique())
print(sales_data['Product ID'].nunique())
```

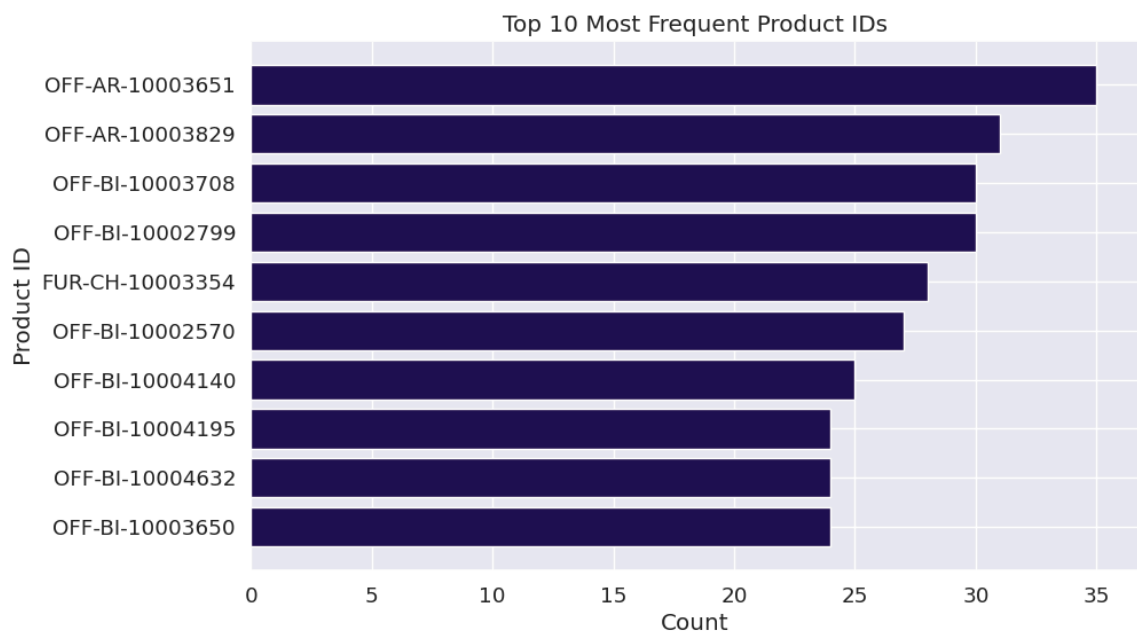
```
['TEC-AC-10003033' 'FUR-CH-10003950' 'TEC-PH-10004664' ...
 'OFF-BI-10002510' 'FUR-ADV-10002329' 'OFF-AP-10002203']
10292
```

```
In [ ]: # Define the number of top product IDs to consider
top_n = 10

# Get the top N most frequent product IDs and their counts
top_product_ids = sales_data['Product ID'].value_counts().head(top_n)
product_counts = top_product_ids.values
product_ids = top_product_ids.index

# Set Seaborn's color palette
sns.set_palette("magma")

# Create the bar plot
plt.figure(figsize=(10, 6))
plt.barh(product_ids, product_counts)
plt.xlabel('Count')
plt.ylabel('Product ID')
plt.title(f'Top {top_n} Most Frequent Product IDs')
plt.gca().invert_yaxis() # Invert y-axis to have the highest count on top
plt.show()
```



Category column

```
In [ ]: sales_data['Category']
```

```
Out[53]: 0          Technology
          1          Furniture
          2          Technology
          3          Technology
          4          Technology
          ...
          51285      Office Supplies
          51286      Office Supplies
          51287      Office Supplies
          51288      Office Supplies
          51289      Office Supplies
          Name: Category, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Category'].unique())
         print(sales_data['Category'].nunique())

['Technology' 'Furniture' 'Office Supplies']
3
```

```
In [ ]: from wordcloud import WordCloud

# Concatenate all categories into a single string
categories_text = ' '.join(sales_data['Category'])

# Generate word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(categories_text)

# Display the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud of Categories')
plt.axis('off')
plt.show()
```



Sub-Category Column


```
In [ ]: sales_data['Sub-Category']
```

```
Out[56]: 0      Accessories
          1      Chairs
          2      Phones
          3      Phones
          4      Copiers
          ...
          51285    Fasteners
          51286    Appliances
          51287    Envelopes
          51288    Binders
          51289    Paper
          Name: Sub-Category, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Sub-Category'].unique())
         print(sales_data['Sub-Category'].nunique())
```

```
['Accessories' 'Chairs' 'Phones' 'Copiers' 'Tables' 'Binders' 'Supplies'
 'Appliances' 'Machines' 'Bookcases' 'Storage' 'Furnishings' 'Art' 'Paper'
 'Envelopes' 'Fasteners' 'Labels']
17
```

In []:

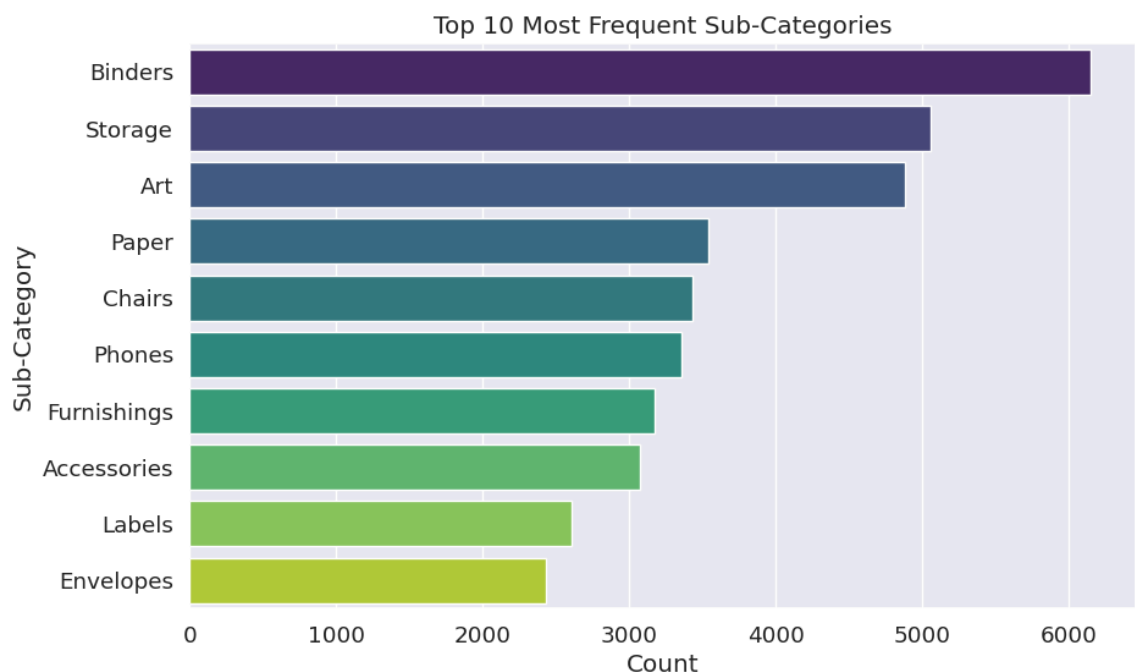
```
# Get the top 10 most frequent sub-categories and their counts
top_subcategories = sales_data['Sub-Category'].value_counts().head(10)

# Create the vertical bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x=top_subcategories.values, y=top_subcategories.index, palette=
plt.xlabel('Count')
plt.ylabel('Sub-Category')
plt.title('Top 10 Most Frequent Sub-Categories')
plt.show()
```

<ipython-input-58-321a8e793935>:6: FutureWarning:

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

```
sns.barplot(x=top_subcategories.values, y=top_subcategories.index, palette='viridis')
```



Product name

In []: sales_data['Product Name']

```
Out[59]: 0      Plantronics CS510 - Over-the-Head monaural Wir...
1      Novimex Executive Leather Armchair, Black
2      Nokia Smart Phone, with Caller ID
3      Motorola Smart Phone, Cordless
4      Sharp Wireless Fax, High-Speed
...
51285   Advantus Thumb Tacks, 12 Pack
51286   Hoover Replacement Belt for Commercial Guardsm...
51287   #10- 4 1/8" x 9 1/2" Security-Tint Envelopes
51288   Acco Index Tab, Economy
51289   Eaton Computer Printout Paper, 8.5 x 11
Name: Product Name, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Product Name'].unique())
print(sales_data['Product Name'].nunique())
```

```
['Plantronics CS510 - Over-the-Head monaural Wireless Headset System'
'Novimex Executive Leather Armchair, Black'
'Nokia Smart Phone, with Caller ID' ...
'Kleencut Forged Office Shears by Acme United Corporation'
'Holmes Visible Mist Ultrasonic Humidifier with 2.3-Gallon Output per Da
y, Replacement Filter'
'Eureka Disposable Bags for Sanitaire Vibra Groomer I Upright Vac']
3788
```

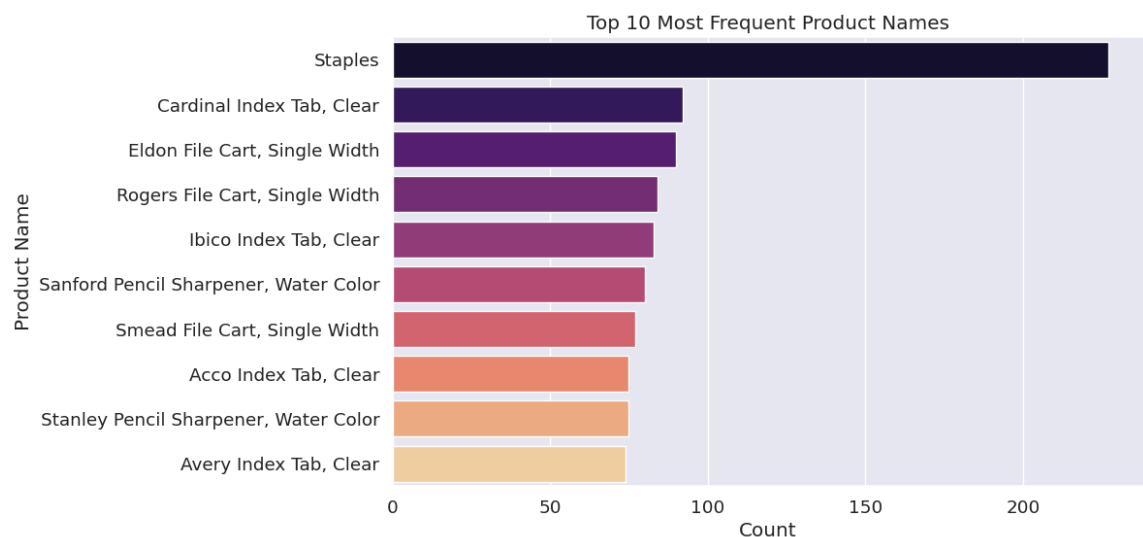
```
In [ ]: # Get the top 10 most frequent product names and their counts
top_product_names = sales_data['Product Name'].value_counts().head(10)

# Create the horizontal bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x=top_product_names.values, y=top_product_names.index, palette=
plt.xlabel('Count')
plt.ylabel('Product Name')
plt.title('Top 10 Most Frequent Product Names')
plt.show()
```

<ipython-input-61-f9b85084377e>:6: FutureWarning:

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

```
sns.barplot(x=top_product_names.values, y=top_product_names.index, palette='magma')
```



Sales Column

```
In [ ]: sales_data['Sales']
```

```
Out[62]: 0      2309.650
         1      3709.395
         2      5175.171
         3      2892.510
         4      2832.960
         ...
        51285     65.100
        51286      0.444
        51287     22.920
        51288     13.440
        51289     61.380
        Name: Sales, Length: 51290, dtype: float64
```

```
In [ ]: print(sales_data['Sales'].unique())
        print(sales_data['Sales'].nunique())
```

```
[2.309650e+03 3.709395e+03 5.175171e+03 ... 1.624000e+00 5.364000e+00
 4.440000e-01]
24988
```

Qunatity column

```
In [ ]: sales_data['Quantity']
```

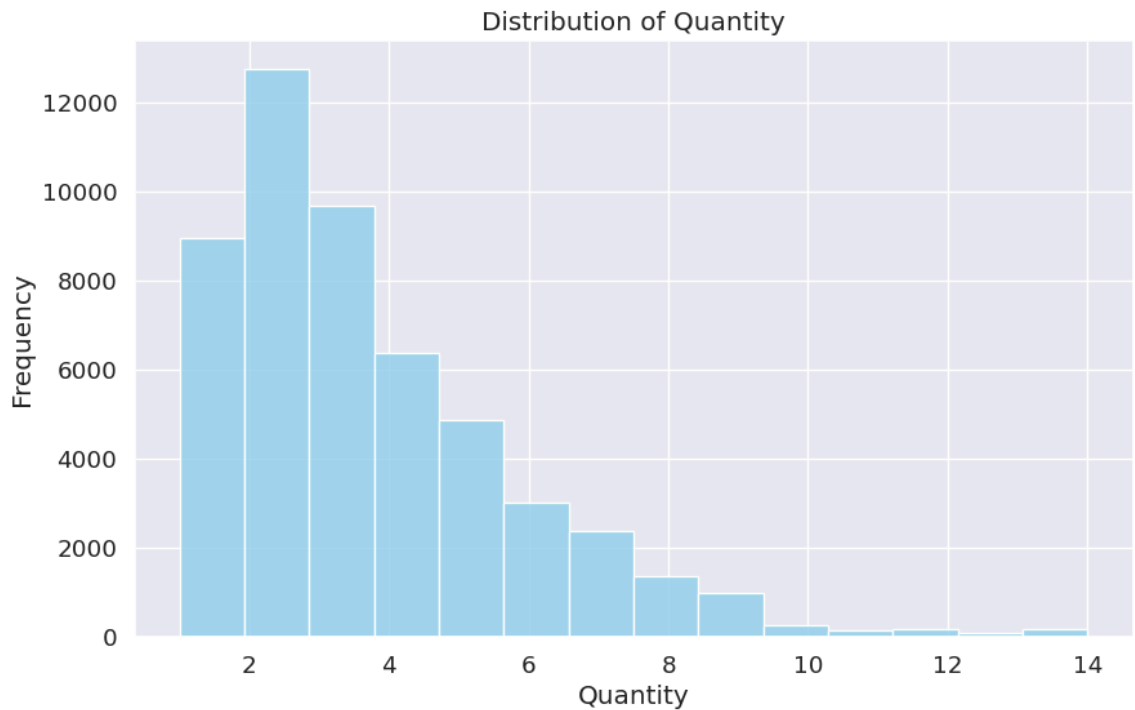
```
Out[64]: 0      7
         1      9
         2      9
         3      5
         4      8
         ..
        51285     5
        51286     1
        51287     3
        51288     2
        51289     3
        Name: Quantity, Length: 51290, dtype: int64
```

```
In [ ]: print(sales_data['Quantity'].unique())
        print(sales_data['Quantity'].nunique())
```

```
[ 7  9  5  8  4  6 13 12 14 10  2 11  3  1]
14
```

In []:

```
# Create the histogram
plt.figure(figsize=(10, 6))
sns.histplot(sales_data['Quantity'], bins=14, color='skyblue')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
plt.title('Distribution of Quantity')
plt.grid(True)
plt.show()
```



***Discount ***

In []: sales_data['Discount']

```
Out[67]: 0      0.0
1      0.1
2      0.1
3      0.1
4      0.0
...
51285  0.0
51286  0.8
51287  0.0
51288  0.0
51289  0.0
Name: Discount, Length: 51290, dtype: float64
```

In []: print(sales_data['Discount'].unique())
print(sales_data['Discount'].nunique())

```
[0.    0.1   0.2   0.4   0.15  0.3   0.5   0.17  0.47  0.25  0.002 0.07
 0.32  0.27  0.7   0.35  0.6   0.65  0.8   0.57  0.37  0.402 0.55  0.202
 0.45  0.602 0.85 ]
27
```

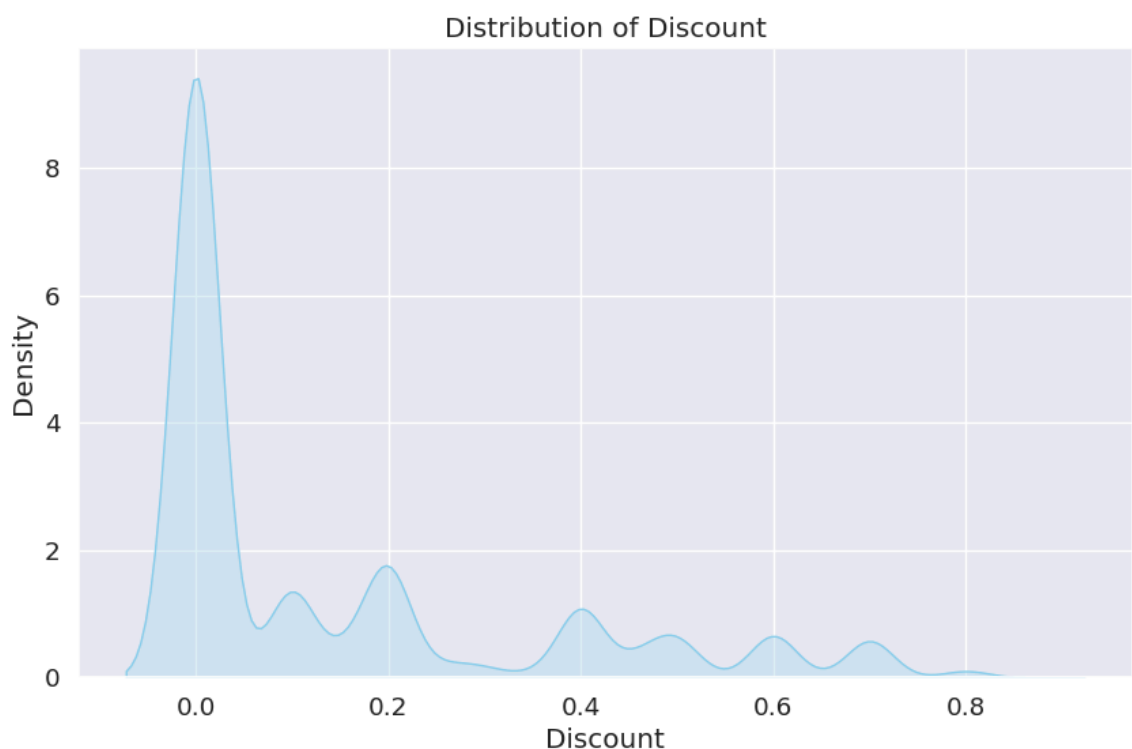
```
In [ ]: import seaborn as sns
import matplotlib.pyplot as plt

# Create the KDE plot
plt.figure(figsize=(10, 6))
sns.kdeplot(sales_data['Discount'], shade=True, color='skyblue')
plt.xlabel('Discount')
plt.ylabel('Density')
plt.title('Distribution of Discount')
plt.grid(True)
plt.show()
```

<ipython-input-76-4ef045299e43>:6: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

```
sns.kdeplot(sales_data['Discount'], shade=True, color='skyblue')
```



Profit column

```
In [ ]: sales_data['Profit']
```

```
Out[69]: 0      762.1845
1     -288.7650
2      919.9710
3     -96.5400
4      311.5200
...
51285     4.5000
51286    -1.1100
51287     11.2308
51288     2.4000
51289     1.8000
Name: Profit, Length: 51290, dtype: float64
```

```
In [ ]: print(sales_data['Profit'].unique())
print(sales_data['Profit'].nunique())
```

```
[ 762.1845 -288.765   919.971   ...   -4.466   -6.456   -49.572 ]
27085
```

Shipping Cost

```
In [ ]: sales_data['Shipping Cost']
```

```
Out[71]: 0          933.570
1          923.630
2          915.490
3          910.160
4          903.040
...
51285      0.010
51286      0.010
51287      0.010
51288      0.003
51289      0.002
Name: Shipping Cost, Length: 51290, dtype: float64
```

```
In [ ]: print(sales_data['Shipping Cost'].unique())
print(sales_data['Shipping Cost'].nunique())
```

```
[9.3357e+02 9.2363e+02 9.1549e+02 ... 1.0000e-02 3.0000e-03 2.0000e-03]
16936
```

Order Priority

```
In [ ]: sales_data['Order Priority']
```

```
Out[74]: 0          Critical
1          Critical
2          Medium
3          Medium
4          Critical
...
51285      Medium
51286      Medium
51287      High
51288      Medium
51289      High
Name: Order Priority, Length: 51290, dtype: object
```

```
In [ ]: print(sales_data['Order Priority'].unique())
print(sales_data['Order Priority'].nunique())
```

```
['Critical' 'Medium' 'High' 'Low']
4
```

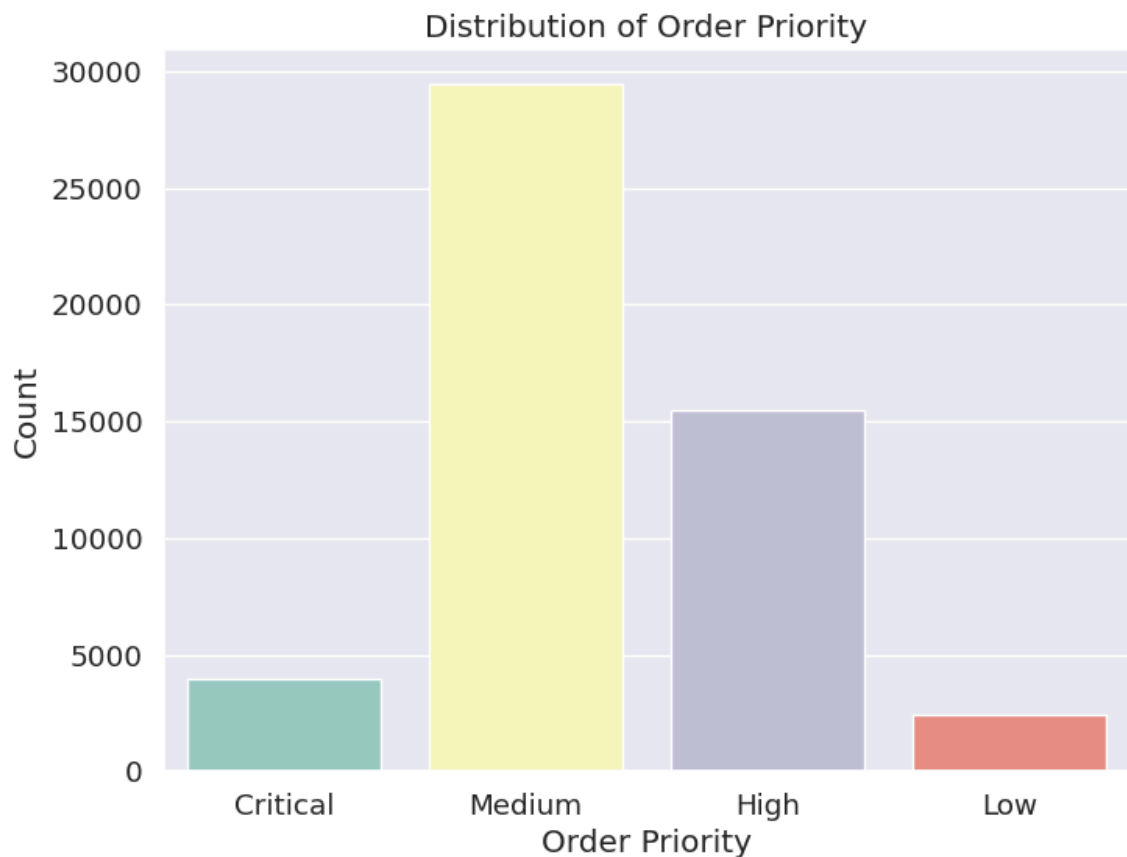
```
In [ ]: import seaborn as sns
import matplotlib.pyplot as plt

# Create the count plot
plt.figure(figsize=(8, 6))
sns.countplot(x='Order Priority', data=sales_data, palette='Set3')
plt.xlabel('Order Priority')
plt.ylabel('Count')
plt.title('Distribution of Order Priority')
plt.show()
```

<ipython-input-77-8a8241bbcc37>:6: FutureWarning:

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

```
sns.countplot(x='Order Priority', data=sales_data, palette='Set3')
```



***Now Lets derive some insights from this Dataset ***

```
In [ ]: sales_data.columns
```

```
Out[102]: Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
                  'Customer ID', 'Customer Name', 'Segment', 'City', 'State', 'Country',
                  'Postal Code', 'Market', 'Region', 'Product ID', 'Category',
                  'Sub-Category', 'Product Name', 'Sales', 'Quantity', 'Discount',
                  'Profit', 'Shipping Cost', 'Order Priority'],
                  dtype='object')
```


***Total Sales ***

```
In [ ]: total_sales = sales_data['Sales'].sum()  
print("Total Sales:", total_sales)
```

Total Sales: 12642501.909880001

```
In [ ]: # Calculate total sales and round to 2 decimal places  
total_sales = round(sales_data['Sales'].sum(), 2)  
print("Total Sales:", total_sales)
```

Total Sales: 12642501.91

Total sales by category and subcategory

In []:

```
total_sales_by_category = sales_data.groupby('Category')['Sales'].sum().reset_index()

# Group the data by 'Sub-Category' and calculate total sales for each sub-category
total_sales_by_subcategory = sales_data.groupby(['Category', 'Sub-Category'])['Sales'].sum().reset_index()

# Plotting
fig, axes = plt.subplots(2, 1, figsize=(12, 12))

# Plot for Total Sales by Category
sns.barplot(x='Category', y='Sales', data=total_sales_by_category, ax=axes[0])
axes[0].set_title('Total Sales by Category')
axes[0].set_xlabel('Category')
axes[0].set_ylabel('Total Sales')
axes[0].tick_params(axis='x', rotation=45)

# Plot for Total Sales by Sub-Category
sns.barplot(x='Sub-Category', y='Sales', data=total_sales_by_subcategory, ax=axes[1])
axes[1].set_title('Total Sales by Sub-Category')
axes[1].set_xlabel('Sub-Category')
axes[1].set_ylabel('Total Sales')

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

<ipython-input-109-00dbef1acce7>:10: FutureWarning:

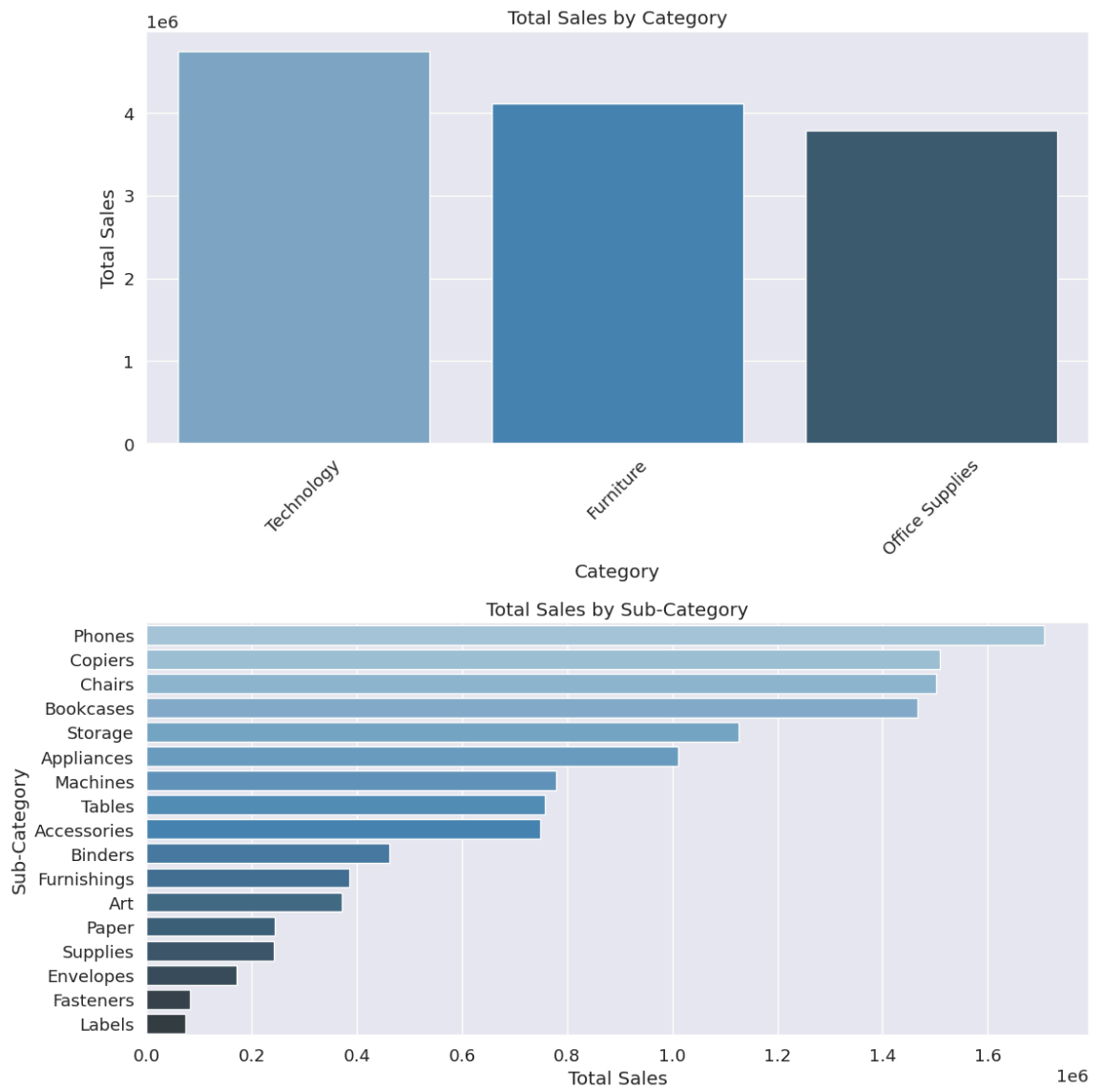
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.barplot(x='Category', y='Sales', data=total_sales_by_category, ax=axes[0], palette='Blues_d')
```

<ipython-input-109-00dbef1acce7>:17: FutureWarning:

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

```
sns.barplot(x='Sub-Category', y='Sales', data=total_sales_by_subcategory, ax=axes[1], palette='Blues_d')
```



Total sales over time and Total sales by Region

```

In [ ]: sns.set_style("whitegrid")

# Plotting
fig, axes = plt.subplots(2, 1, figsize=(12, 12))

# Total Sales Over Time
sales_data['Order Date'] = pd.to_datetime(sales_data['Order Date']) # Convert to datetime
total_sales_over_time = sales_data.groupby(sales_data['Order Date'].dt.to_period('M')).sum()
total_sales_over_time.plot(ax=axes[0], marker='o', color='skyblue')
axes[0].set_title('Total Sales Over Time')
axes[0].set_xlabel('Order Date')
axes[0].set_ylabel('Total Sales')

# Total Sales by Region
total_sales_by_region = sales_data.groupby('Region')['Sales'].sum().sort_values(ascending=False)
sns.barplot(x=total_sales_by_region.index, y=total_sales_by_region.values, palette='viridis')
axes[1].set_title('Total Sales by Region')
axes[1].set_xlabel('Region')
axes[1].set_ylabel('Total Sales')
axes[1].tick_params(axis='x', rotation=45) # Rotate x-axis labels

plt.tight_layout()
plt.show()

```

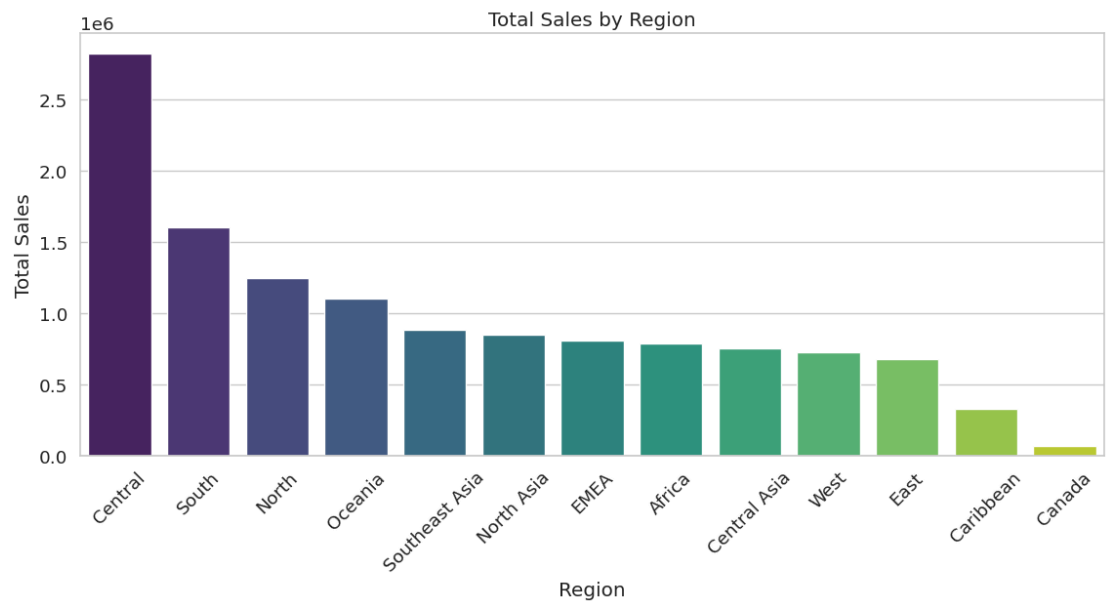
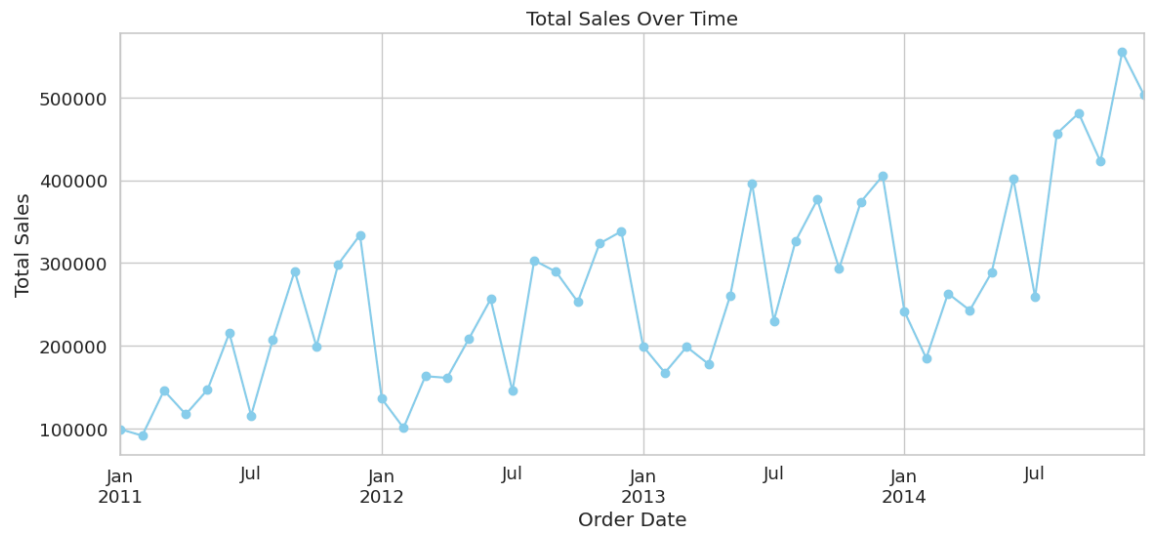
<ipython-input-111-d3b1712da16b>:16: FutureWarning:

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

```

sns.barplot(x=total_sales_by_region.index, y=total_sales_by_region.values,
            palette='viridis')

```



Total sales by Order Priority, Customer Segment and Market

```

In [ ]: sns.set_style("whitegrid")

# Plotting
fig, axes = plt.subplots(3, 1, figsize=(12, 18))

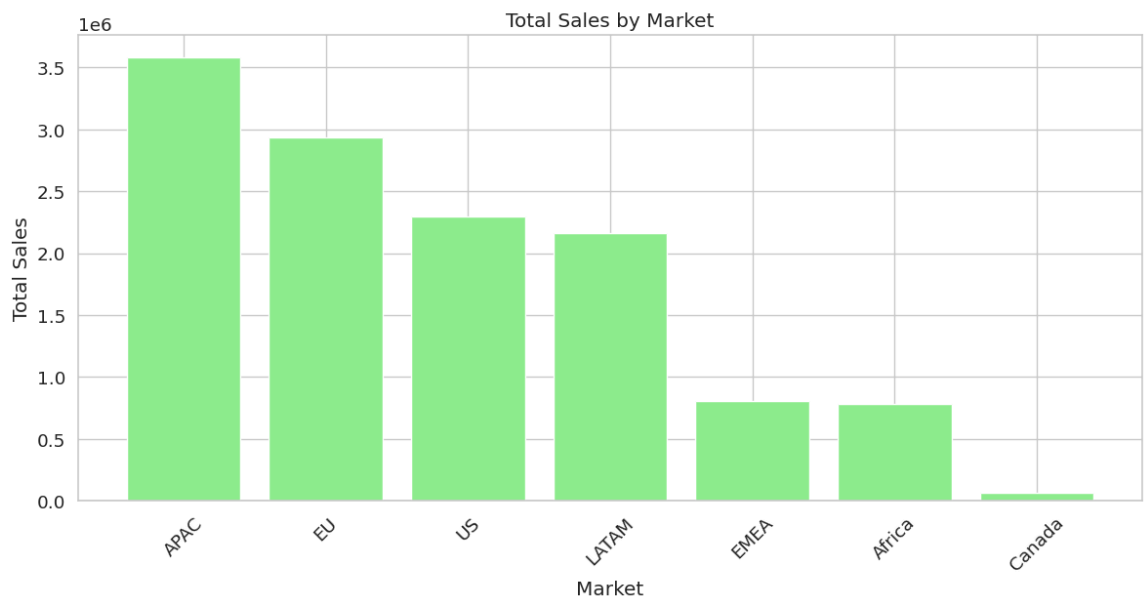
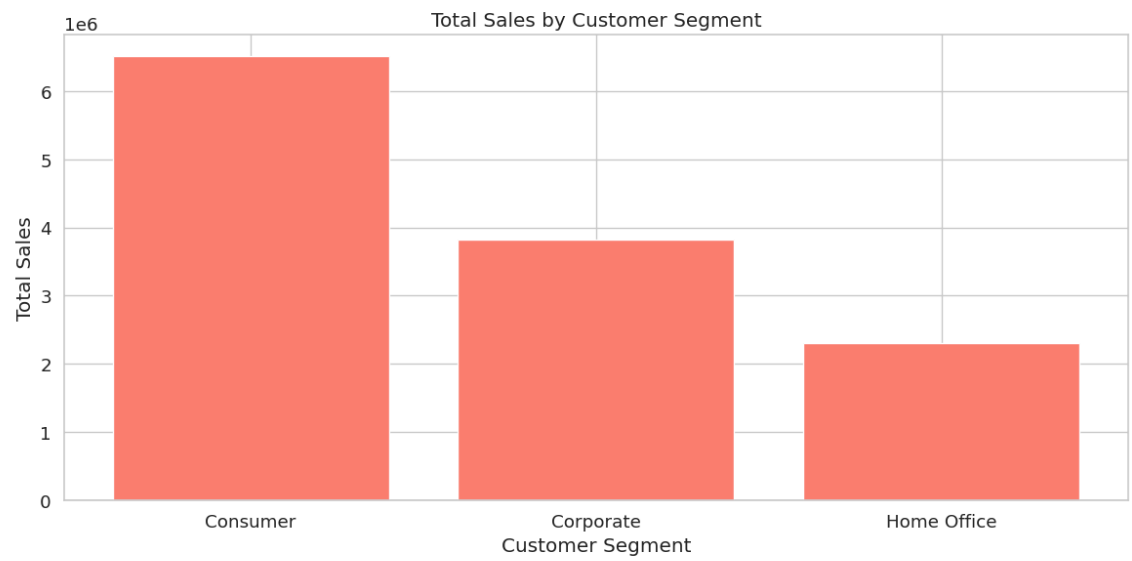
# Total Sales by Order Priority
total_sales_by_order_priority = sales_data.groupby('Order Priority')['Sales'].sum()
axes[0].bar(total_sales_by_order_priority.index, total_sales_by_order_priority.values, color='red')
axes[0].set_title('Total Sales by Order Priority')
axes[0].set_xlabel('Order Priority')
axes[0].set_ylabel('Total Sales')

# Total Sales by Customer Segment
total_sales_by_customer_segment = sales_data.groupby('Segment')['Sales'].sum()
axes[1].bar(total_sales_by_customer_segment.index, total_sales_by_customer_segment.values, color='blue')
axes[1].set_title('Total Sales by Customer Segment')
axes[1].set_xlabel('Customer Segment')
axes[1].set_ylabel('Total Sales')

# Total Sales by Market
total_sales_by_market = sales_data.groupby('Market')['Sales'].sum().sort_values()
axes[2].bar(total_sales_by_market.index, total_sales_by_market.values, color='green')
axes[2].set_title('Total Sales by Market')
axes[2].set_xlabel('Market')
axes[2].set_ylabel('Total Sales')
axes[2].tick_params(axis='x', rotation=45) # Rotate x-axis labels

plt.tight_layout()
plt.show()

```



best-selling products

```
In [ ]: total_sales_by_product = sales_data.groupby('Product Name')['Sales'].sum()..

# Identifying the best-selling products (top 10)
best_selling_products = total_sales_by_product.head(10)

# Displaying the best-selling products
print("Top 10 Best-Selling Products:")
print(best_selling_products)
```

Top 10 Best-Selling Products:

Product Name	
Apple Smart Phone, Full Size	86935.7786
Cisco Smart Phone, Full Size	76441.5306
Motorola Smart Phone, Full Size	73156.3030
Nokia Smart Phone, Full Size	71904.5555
Canon imageCLASS 2200 Advanced Copier	61599.8240
Hon Executive Leather Armchair, Adjustable	58193.4841
Office Star Executive Leather Armchair, Adjustable	50661.6840
Harbour Creations Executive Leather Armchair, Adjustable	50121.5160
Samsung Smart Phone, Cordless	48653.4600
Nokia Smart Phone, with Caller ID	47877.7857

Name: Sales, dtype: float64

In []:

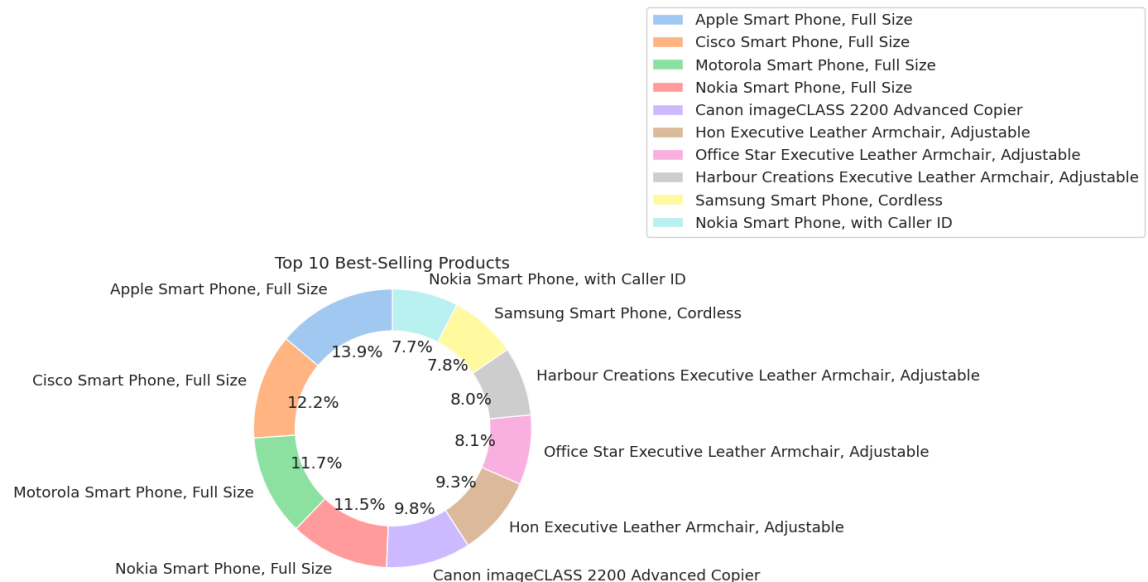
```
# Define colors for the pie chart
colors = sns.color_palette('pastel')[0:len(best_selling_products)]

# Plotting the donut chart
patches, texts, autotexts = plt.pie(best_selling_products, labels=best_selling_products, autopct='%1.1f%%')
plt.title('Top 10 Best-Selling Products')

# Draw a circle in the middle to create the donut shape
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Create legend based on sales
sorted_labels = [label for _, label in sorted(zip(best_selling_products, best_selling_products), key=lambda x: x[1], reverse=True)]
plt.legend(handles=patches, labels=sorted_labels, loc="center left", bbox_to_anchor=(1, 0.5))

# Equal aspect ratio ensures that pie is drawn as a circle
plt.axis('equal')
plt.show()
```



Order processing Analysis

In []:

```
# Replace non-numeric values with zeros
order_processing_efficiency = order_processing_efficiency.fillna(0)

# Convert values to integers
order_processing_efficiency = order_processing_efficiency.astype(int)

# Plotting
plt.figure(figsize=(10, 6))
sns.heatmap(order_processing_efficiency, annot=True, cmap='Blues', fmt='d')
plt.title('Order Processing Efficiency by Ship Mode and Order Priority')
plt.xlabel('Order Priority')
plt.ylabel('Ship Mode')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()
```

