```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

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

```
customers_df = pd.read_csv("/content/drive/MyDrive/Zeotap_Assignment_Ansh/Customers.csv")
customers_df.head()
```

	CustomerID	CustomerName	Region	SignupDate
0	C0001	Lawrence Carroll	South America	2022-07-10
1	C0002	Elizabeth Lutz	Asia	2022-02-13
2	C0003	Michael Rivera	South America	2024-03-07
3	C0004	Kathleen Rodriguez	South America	2022-10-09
4	C0005	Laura Weber	Asia	2022-08-15

Next steps: Generate code with customers_df View recommended plots New interactive sheet

```
products_df = pd.read_csv("/content/drive/MyDrive/Zeotap_Assignment_Ansh/Products.csv")
products_df.head()
```

	ProductID	ProductName	Category	Price
0	P001	ActiveWear Biography	Books	169.30
1	P002	ActiveWear Smartwatch	Electronics	346.30
2	P003	ComfortLiving Biography	Books	44.12
3	P004	BookWorld Rug	Home Decor	95.69
4	P005	TechPro T-Shirt	Clothing	429.31

Next steps: Generate code with products_df View recommended plots New interactive sheet

```
transactions_df = pd.read_csv("/content/drive/MyDrive/Zeotap_Assignment_Ansh/Transactions.csv")
transactions_df.head()
```

	TransactionID	CustomerID	ProductID	TransactionDate	Quantity	TotalValue	Price
0	T00001	C0199	P067	2024-08-25 12:38:23	1	300.68	300.68
1	T00112	C0146	P067	2024-05-27 22:23:54	1	300.68	300.68
2	T00166	C0127	P067	2024-04-25 07:38:55	1	300.68	300.68
3	T00272	C0087	P067	2024-03-26 22:55:37	2	601.36	300.68
4	T00363	C0070	P067	2024-03-21 15:10:10	3	902.04	300.68

Next steps: Generate code with transactions_df View recommended plots New interactive sheet

Summaries of the given datasets:

1. Customers

```
customers_df.describe(include='all')
```

	CustomerID	CustomerName	Region	SignupDate
count	200	200	200	200
unique	200	200	4	179
top	C0185	Kathleen Logan	South America	2022-04-16
freq	1	1	59	3

```
customers_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   CustomerID    200 non-null    object
 1   CustomerName  200 non-null    object
 2   Region        200 non-null    object
 3   SignupDate    200 non-null    object
dtypes: object(4)
memory usage: 6.4+ KB
```

2. Products

```
products_df.describe(include='all')
```

	ProductID	ProductName	Category	Price
count	100	100	100	100.000000
unique	100	66	4	NaN
top	P100	SoundWave Headphones	Books	NaN
freq	1	4	26	NaN
mean	NaN	NaN	NaN	267.551700
std	NaN	NaN	NaN	143.219383
min	NaN	NaN	NaN	16.080000
25%	NaN	NaN	NaN	147.767500
50%	NaN	NaN	NaN	292.875000
75%	NaN	NaN	NaN	397.090000
max	NaN	NaN	NaN	497.760000

```
products_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   ProductID    100 non-null    object 
 1   ProductName  100 non-null    object 
 2   Category     100 non-null    object 
 3   Price        100 non-null    float64
dtypes: float64(1), object(3)
memory usage: 3.3+ KB
```

3. Transactions

```
transactions_df.describe(include='all')
```

	TransactionID	CustomerID	ProductID	TransactionDate	Quantity	TotalValue	Price
count	1000	1000	1000	1000	1000.000000	1000.000000	1000.00000
unique	1000	199	100	1000	NaN	NaN	NaN
top	T00992	C0109	P059	2024-04-21 10:52:24	NaN	NaN	NaN
freq	1	11	19	1	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	2.537000	689.995560	272.55407
std	NaN	NaN	NaN	NaN	1.117981	493.144478	140.73639
min	NaN	NaN	NaN	NaN	1.000000	16.080000	16.08000
25%	NaN	NaN	NaN	NaN	2.000000	295.295000	147.95000
50%	NaN	NaN	NaN	NaN	3.000000	588.880000	299.93000
75%	NaN	NaN	NaN	NaN	4.000000	1011.660000	404.40000
max	NaN	NaN	NaN	NaN	4.000000	1991.040000	497.76000

```
transactions_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   TransactionID    1000 non-null   object
 1   CustomerID       1000 non-null   object
 2   ProductID        1000 non-null   object
 3   TransactionDate  1000 non-null   object
 4   Quantity         1000 non-null   int64
 5   TotalValue       1000 non-null   float64
 6   Price            1000 non-null   float64
dtypes: float64(2), int64(1), object(4)
memory usage: 54.8+ KB
```

Key Observations:

1. In the given dataset there are **no missing values** in any of the features/input vector.

2. Except **SignupDate feature in customers dataset and TransactionDate feature in transactions dataset**, every feature is in desired and correct datatype. Changing them into pandas's datetime format to allow date-based operations.

```
customers_df['SignupDate'] = pd.to_datetime(customers_df['SignupDate'])
transactions_df['TransactionDate'] = pd.to_datetime(transactions_df['TransactionDate'])
```

```
print(customers_df.dtypes)
print(products_df.dtypes)
print(transactions_df.dtypes)
```

```
CustomerID              object
CustomerName            object
Region                  object
SignupDate      datetime64[ns]
dtype: object
ProductID       object
ProductName     object
Category        object
Price          float64
dtype: object
TransactionID              object
CustomerID                 object
ProductID                  object
TransactionDate    datetime64[ns]
Quantity                    int64
TotalValue                float64
Price                     float64
dtype: object
```

3. No important feature's values are duplicated in the given dataset. The Customer_ID's values in customers dataset are all unique similarly ProducId in products dataset and TransactionsID in transaction's dataset. There are some CustomerID which are not unique in transaction's dataset but It is valid because same person can but multiple products and multiple times.

```
merged_data = pd.merge(transactions_df, customers_df, on='CustomerID')
merged_data = pd.merge(merged_data, products_df, on='ProductID')

# Count the number of unique customers per region
region_customer_counts = merged_data.groupby('Region')['CustomerID'].nunique().sort_values(ascending=False)
print("Number of active customers per region:")
print(region_customer_counts, "\n")

# Region with the highest number of customers
top_region = region_customer_counts.idxmax()
print(f"Region with the highest number of active customers: {top_region}\n")

# Analyze purchasing habits in the top region
top_region_data = merged_data[merged_data['Region'] == top_region]

# Aggregating purchasing habits
top_region_habits = top_region_data.groupby('Category').agg(
    TotalSales=('TotalValue', 'sum'),
    AverageQuantity=('Quantity', 'mean'),
    TotalTransactions=('TransactionID', 'count')
).sort_values(by='TotalSales', ascending=False)
```

```
print(f"Purchasing habits in the top region ({top_region}):")
print(top_region_habits)
```

```
Number of active customers per region:
Region
South America    59
Europe           50
North America    46
Asia             44
Name: CustomerID, dtype: int64

Region with the highest number of active customers: South America

Purchasing habits in the top region (South America):
             TotalSales  AverageQuantity  TotalTransactions
Category
Books          69752.03         2.677778                 90
Electronics    58846.32         2.506329                 79
Home Decor     48310.72         2.666667                 72
Clothing       42443.49         2.507937                 63
```

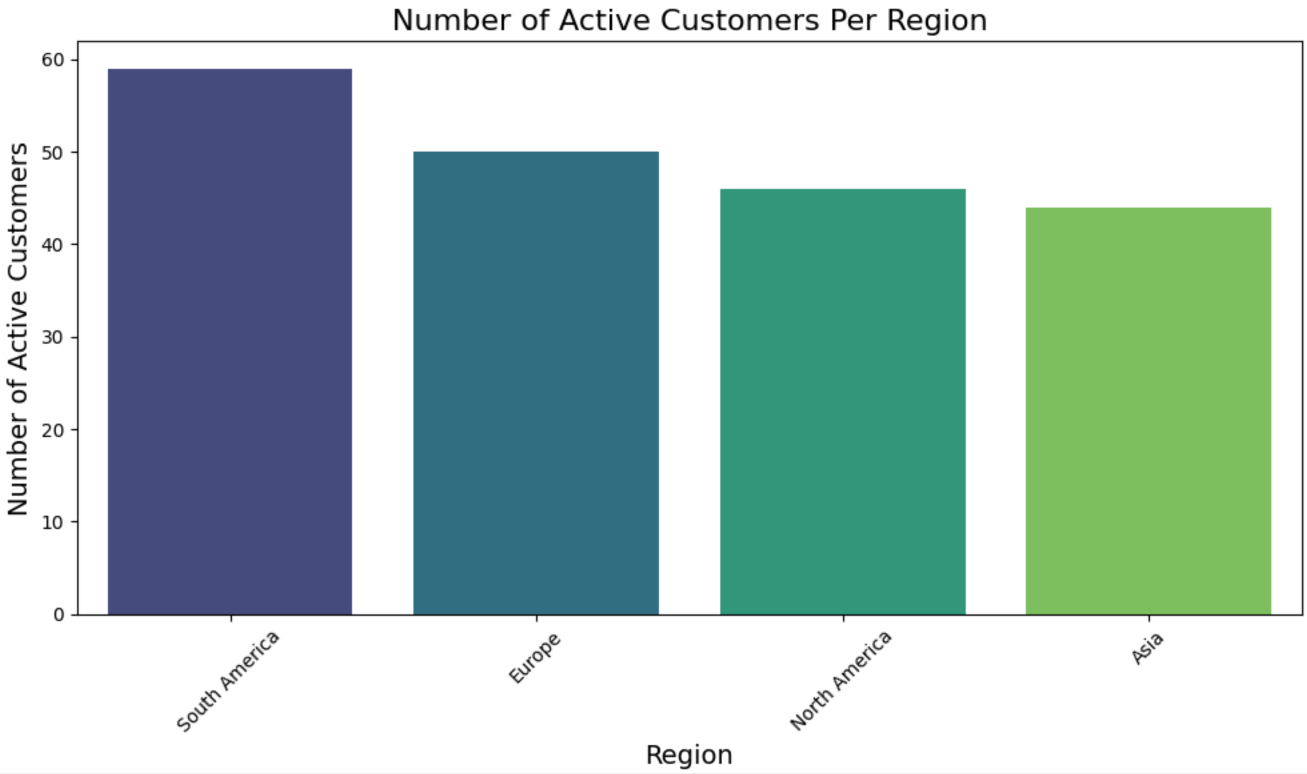
```
import matplotlib.pyplot as plt
import seaborn as sns

# Bar chart for active customers per region
plt.figure(figsize=(10, 6))
sns.barplot(x=region_customer_counts.index, y=region_customer_counts.values, palette="viridis")
plt.title("Number of Active Customers Per Region", fontsize=16)
plt.xlabel("Region", fontsize=14)
plt.ylabel("Number of Active Customers", fontsize=14)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-37-2298b66a0bc2>: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 `le

  sns.barplot(x=region_customer_counts.index, y=region_customer_counts.values, palette="viridis")
```




```
# Best-performing product categories and revenue contribution
category_revenue = merged_data.groupby('Category').agg(
    TotalRevenue=('TotalValue', 'sum'),
    RevenueContribution=('TotalValue', lambda x: (x.sum() / merged_data['TotalValue'].sum()) * 100)
).sort_values(by='TotalRevenue', ascending=False)

print("Best-performing product categories and their contribution to revenue:")
print(category_revenue)
```

```
Best-performing product categories and their contribution to revenue:
             TotalRevenue  RevenueContribution
Category
Books           192147.47            27.847639
Electronics     180783.50            26.200676
Clothing        166170.66            24.082859
Home Decor      150893.93            21.868826
```

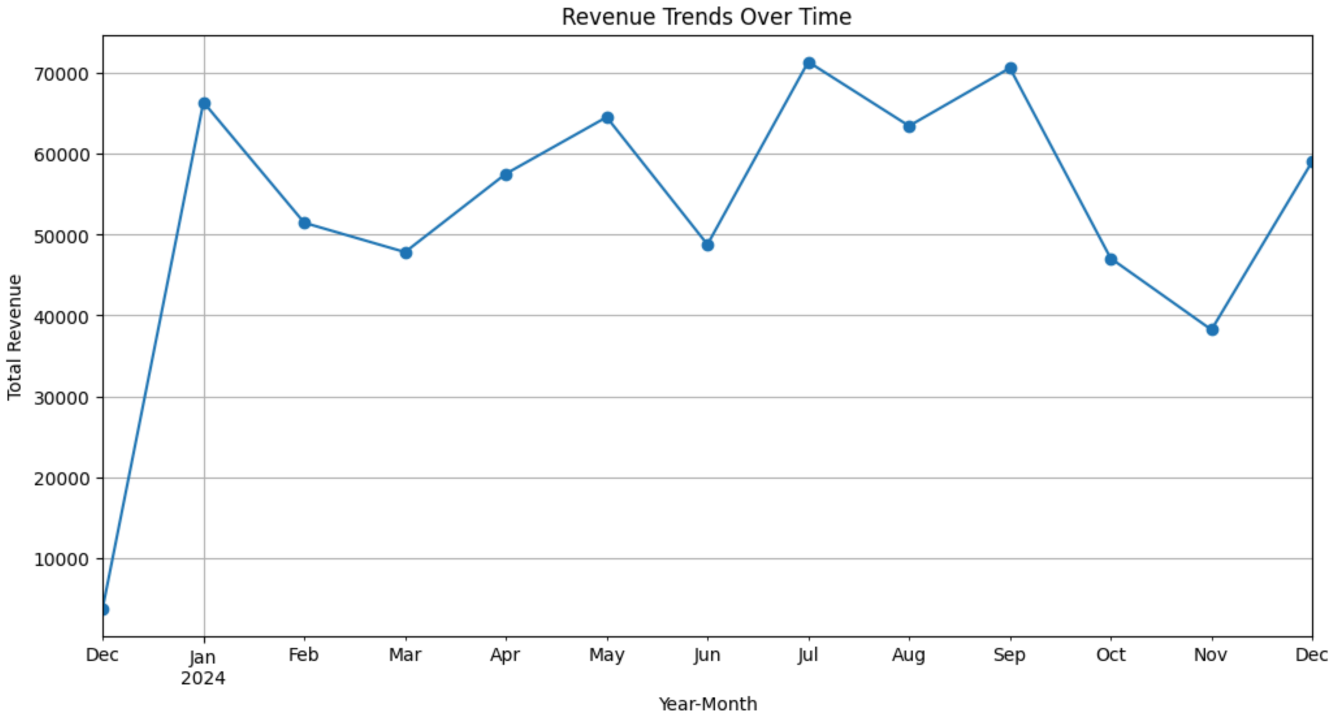
```
import matplotlib.pyplot as plt

# Extract year and month for trend analysis
merged_data['YearMonth'] = merged_data['TransactionDate'].dt.to_period('M')

# Revenue trends over time
revenue_trends = merged_data.groupby('YearMonth')['TotalValue'].sum()

# Plot revenue trends
plt.figure(figsize=(12, 6))
revenue_trends.plot(kind='line', marker='o', title='Revenue Trends Over Time')
plt.xlabel('Year-Month')
plt.ylabel('Total Revenue')
plt.grid()
plt.show()

# Identify peak sales periods
peak_sales = revenue_trends.sort_values(ascending=False).head(5)
print("Top 5 peak sales periods:")
print(peak_sales)
```




```
Top 5 peak sales periods:
YearMonth
2024-07    71366.39
2024-09    70603.75
2024-01    66376.39
2024-05    64527.74
2024-08    63436.74
Freq: M, Name: TotalValue, dtype: float64
```

```
# Count transactions per customer
customer_transactions = merged_data.groupby('CustomerID').size()

# Categorize customers
merged_data['CustomerType'] = merged_data['CustomerID'].map(
    lambda x: 'Repeat' if customer_transactions[x] > 1 else 'One-Time'
)

# Calculate contributions to total sales
customer_type_revenue = merged_data.groupby('CustomerType').agg(
    TotalRevenue=('TotalValue', 'sum'),
    RevenueContribution=('TotalValue', lambda x: (x.sum() / merged_data['TotalValue'].sum()) * 100),
    CustomerCount=('CustomerID', 'nunique')
```

```
)

print("Customer retention analysis (repeat vs. one-time buyers):")
print(customer_type_revenue)
```

```
Customer retention analysis (repeat vs. one-time buyers):
              TotalRevenue  RevenueContribution  CustomerCount
CustomerType
One-Time           6340.97             0.918987             12
Repeat           683654.59            99.081013            187
```

Business Insights:

1. Regional Customer Activity:

South America leads with **59 active customer**s, followed by Europe (50) and North America (46). The top region's dominance reflects its strong customer base, providing an opportunity to expand marketing efforts in nearby regions like North America and Asia.

2. Purchasing Trends in South America:

In South America,**"Books"** generate the highest sales**(69,752 USD)**, then "Electronics" (58846 USD). Customers in this region purchase an average of 2.5–2.7 units per transaction, indicating a preference for moderate quantities across all categories.

3. Top-Performing Categories:

"Books" contribute the most to revenue (27.85%), followed by "Electronics" (26.20%). Combined with "Clothing" (24.08%) and "Home Decor" (21.87%), these categories dominate sales, suggesting potential in expanding these product lines for further growth.

4. Seasonal Sales Peaks:

The months of **July, September, and January witness the highest sales**, with **July 2024 leading at $71,366**. This pattern highlights seasonal opportunities for promotional campaigns and inventory planning during these peak periods.

5. Customer Retention Insights:

Repeat customers dominate revenue generation, contributing 99.08% of total sales, while **one-time buyers contribute less than 1%.** This underscores the importance of loyalty programs and retention strategies to sustain and grow the existing customer base.