```
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns

# Load datasets
customer_data = pd.read_csv('Customers.csv')
product_data = pd.read_csv('Products.csv')
transaction_data = pd.read_csv('Transactions.csv')
# Task 1: Exploratory Data Analysis (EDA)
# Merge datasets for comprehensive analysis
data = transactions.merge(customers, on='CustomerID').merge(products, or

# 1. Basic statistics and missing values
print(data.info())
print(data.describe())
print(data.isnull().sum())

# 2. Top products by sales volume
top_products = data.groupby('ProductName')['Quantity'].sum().sort_values
print("Top 10 Products by Quantity Sold:\n", top_products)

# 3. Revenue by region
revenue_region = data.groupby('Region')['TotalValue'].sum().sort_values(
print("Revenue by Region:\n", revenue_region)

# 4. Customer signups over time
data['SignupDate'] = pd.to_datetime(data['SignupDate'])
monthly_signups = data.groupby(data['SignupDate'].dt.to_period('M'))['Cu

# 5. Correlation between price and quantity
# Check if Price_x and Price_y exist
if 'Price_x' in data.columns and 'Price_y' in data.columns:
    # Check if the columns are identical
    if data['Price_x'].equals(data['Price_y']):
        print("Price_x and Price_y are identical.")
        data.rename(columns={'Price_x': 'Price'}, inplace=True)  # Use c
    else:
        print("Price_x and Price_y are different.")
        # Option 1: Take the average of the two columns
        data['Price'] = data[['Price_x', 'Price_y']].mean(axis=1)
        # Option 2: Choose one based on context (e.g., 'Price_x')
        # data.rename(columns={'Price_x': 'Price'}, inplace=True)
elif 'Price_x' in data.columns:
    data.rename(columns={'Price_x': 'Price'}, inplace=True)
elif 'Price_y' in data.columns:
    data.rename(columns={'Price_y': 'Price'}, inplace=True)
else:
    raise KeyError("No Price column found in the dataset.")

# Correlation analysis between Price and Quantity
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 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
```

FirstName_LastName_Lookalike.csv

1 to 10 of 199 entries Filter

| CustomerID | TotalValue | Quantity | Reg |
|---|---|---|---|
| C0001 | 3354.5200000000004 | 12 | 3 |
| C0002 | 1862.74 | 10 | 0 |
| C0003 | 2725.38 | 14 | 3 |
| C0004 | 5354.88 | 23 | 3 |
| C0005 | 2034.24 | 7 | 0 |
| C0006 | 4227.57 | 12 | 3 |
| C0007 | 2579.8199999999997 | 8 | 0 |
| C0008 | 4271.61 | 20 | 2 |
| C0009 | 896.5 | 3 | 1 |
| C0010 | 1717.5500000000002 | 12 | 1 |

Show 10 per page 1 2 10 20

```
 6   Price_x          1000 non-null   float64
 7   CustomerName     1000 non-null   object
 8   Region           1000 non-null   object
 9   SignupDate       1000 non-null   object
 10  ProductName      1000 non-null   object
 11  Category         1000 non-null   object
 12  Price_y          1000 non-null   float64
dtypes: float64(3), int64(1), object(9)
memory usage: 101.7+ KB
None
          Quantity   TotalValue     Price_x     Price_y
count  1000.000000  1000.000000  1000.00000  1000.00000
mean      2.537000   689.995560   272.55407   272.55407
std       1.117981   493.144478   140.73639   140.73639
min       1.000000    16.080000    16.08000    16.08000
25%       2.000000   295.295000   147.95000   147.95000
50%       3.000000   588.880000   299.93000   299.93000
75%       4.000000  1011.660000   404.40000   404.40000
max       4.000000  1991.040000   497.76000   497.76000
TransactionID      0
CustomerID         0
ProductID          0
TransactionDate    0
Quantity           0
TotalValue         0
Price_x            0
CustomerName       0
Region             0
SignupDate         0
ProductName        0
Category           0
Price_y            0
dtype: int64
Top 10 Products by Quantity Sold:
 ProductName
ActiveWear Smartwatch    100
SoundWave Headphones      97
HomeSense Desk Lamp       81
ActiveWear Rug            79
SoundWave Cookbook        78
ActiveWear Jacket         76
BookWorld Biography       71
TechPro T-Shirt           66
SoundWave Desk Lamp       64
TechPro Textbook          62
Name: Quantity, dtype: int64
Revenue by Region:
```

```
correlation_matrix = data[['Price', 'Quantity']].corr()
print("\nCorrelation between Price and Quantity:\n", correlation_matrix)
# Visualizations
plt.figure(figsize=(10, 6))
sns.barplot(x=top_products.index, y=top_products.values, palette="Set2")
plt.title('Top 10 Products by Quantity Sold')
plt.xticks(rotation=45)
plt.show()

# Line graph with measurements (showing values at each data point)
plt.figure(figsize=(10, 6))
ax = monthly_signups.plot(kind='line', marker='o')
plt.title('Monthly Customer Signups (Number of Signups)')
plt.xlabel('Month')
plt.ylabel('Number of Signups')

# Adding the data points with the unit in the annotations

plt.show()
```

```
Correlation between Price and Quantity:
              Price  Quantity
Price     1.000000 -0.009378
Quantity -0.009378  1.000000
<ipython-input-43-e4ef799c1a26>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will b

  sns.barplot(x=top_products.index, y=top_products.values, palette
```