```
import pandas as pd


df = pd.read_excel('Online Retail.xlsx')


print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
None
```

```
print(df.head())
```

```
  InvoiceNo StockCode                      Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
```

```
df = df.dropna(subset=['CustomerID'])


df = df.drop_duplicates()


print(f"Dataset shape after cleaning: {df.shape}")
```

```
Dataset shape after cleaning: (401604, 8)
```

```
df = df[~df['InvoiceNo'].astype(str).str.startswith('C')]


print(f"Dataset shape after removing canceled invoices: {df.shape}")
```

```
Dataset shape after removing canceled invoices: (392732, 8)
```

```
df['TotalPrice'] = df['Quantity'] * df['UnitPrice']

print(df[['Quantity', 'UnitPrice', 'TotalPrice']].head())
```

```
   Quantity  UnitPrice  TotalPrice
0         6       2.55       15.30
1         6       3.39       20.34
2         8       2.75       22.00
3         6       3.39       20.34
4         6       3.39       20.34
```

```
# See quick summary
print(df.describe())

# Count unique values
print("Unique Customers:", df['CustomerID'].nunique())
```

```
print("Unique Products:", df['StockCode'].nunique())
```

```
          Quantity                    InvoiceDate      UnitPrice  \
count  392732.000000                         392732  392732.000000
mean       13.153718  2011-07-10 19:15:24.576301568       3.125596
min         1.000000            2010-12-01 08:26:00       0.000000
25%         2.000000            2011-04-07 11:12:00       1.250000
50%         6.000000            2011-07-31 12:02:00       1.950000
75%        12.000000            2011-10-20 12:53:00       3.750000
max     80995.000000            2011-12-09 12:50:00    8142.750000
std       181.588420                            NaN      22.240725

         CustomerID     TotalPrice
count  392732.000000  392732.000000
mean    15287.734822      22.629195
min     12346.000000       0.000000
25%     13955.000000       4.950000
50%     15150.000000      12.390000
75%     16791.000000      19.800000
max     18287.000000  168469.600000
std      1713.567773     311.083465
Unique Customers: 4339
Unique Products: 3665
```

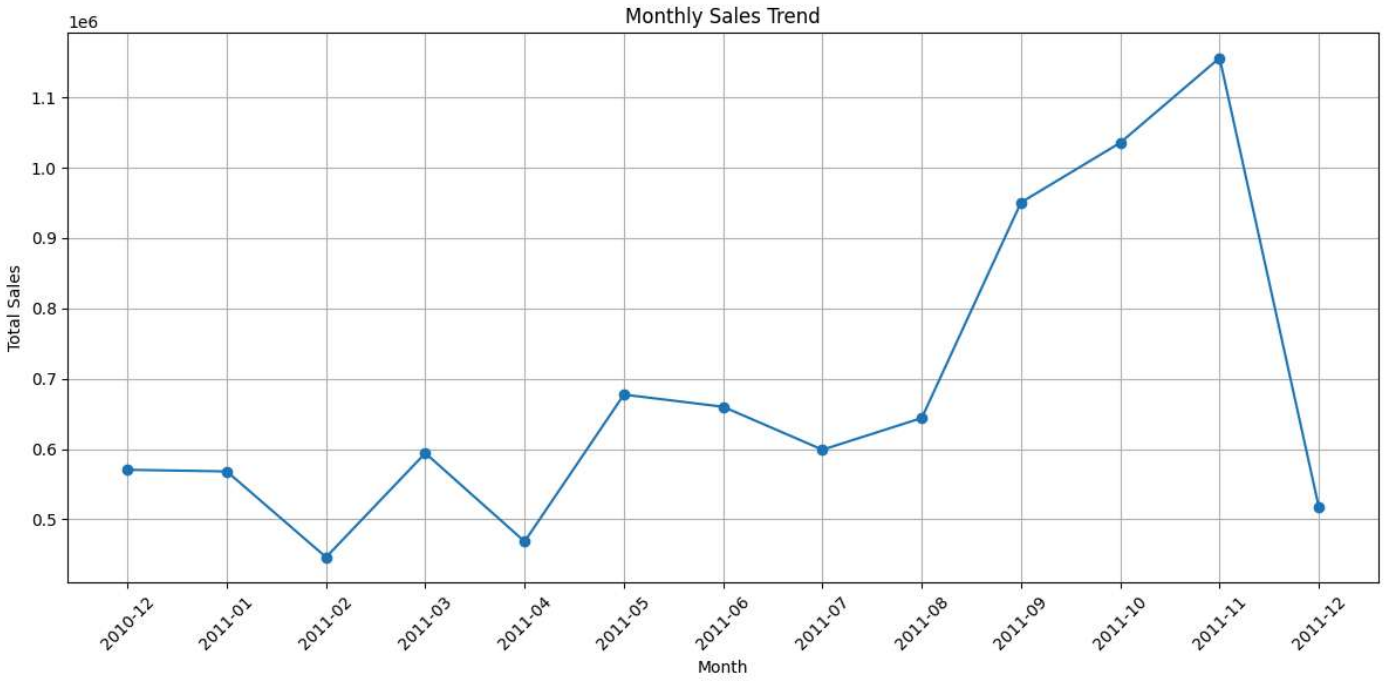
```
top_countries = df.groupby('Country')['TotalPrice'].sum().sort_values(ascending=False).head(10)

# Show top 10 countries by revenue
print(top_countries)
```

```
Country
United Kingdom    7285024.644
Netherlands        285446.340
EIRE               265262.460
Germany            228678.400
France             208934.310
Australia          138453.810
Spain               61558.560
Switzerland         56443.950
Belgium             41196.340
Sweden              38367.830
Name: TotalPrice, dtype: float64
```

```
# Convert InvoiceDate to datetime
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

# Extract year-month
df['InvoiceMonth'] = df['InvoiceDate'].dt.to_period('M')

monthly_sales = df.groupby('InvoiceMonth')['TotalPrice'].sum()

# Convert to DataFrame for display
monthly_sales_df = monthly_sales.to_frame().reset_index()
monthly_sales_df.columns = ['Month', 'Total Sales']
print(monthly_sales_df.head())
```

```
     Month  Total Sales
0  2010-12   570422.730
1  2011-01   568101.310
2  2011-02   446084.920
3  2011-03   594081.760
4  2011-04   468374.331
```

```
import matplotlib.pyplot as plt

# Plot line chart
plt.figure(figsize=(12,6))
plt.plot(monthly_sales.index.astype(str), monthly_sales.values, marker='o')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
import datetime as dt

# Use the latest invoice date for Recency calculation
latest_date = df['InvoiceDate'].max()

# RFM Table
rfm = df.groupby('CustomerID').agg({
    'InvoiceDate': lambda x: (latest_date - x.max()).days,  # Recency
    'InvoiceNo': 'nunique',                                  # Frequency
    'TotalPrice': 'sum'                                      # Monetary
}).reset_index()

rfm.columns = ['CustomerID', 'Recency', 'Frequency', 'Monetary']

print(rfm.head())
```

```
   CustomerID  Recency  Frequency  Monetary
0     12346.0      325          1  77183.60
1     12347.0        1          7   4310.00
2     12348.0       74          4   1797.24
3     12349.0       18          1   1757.55
4     12350.0      309          1    334.40
```

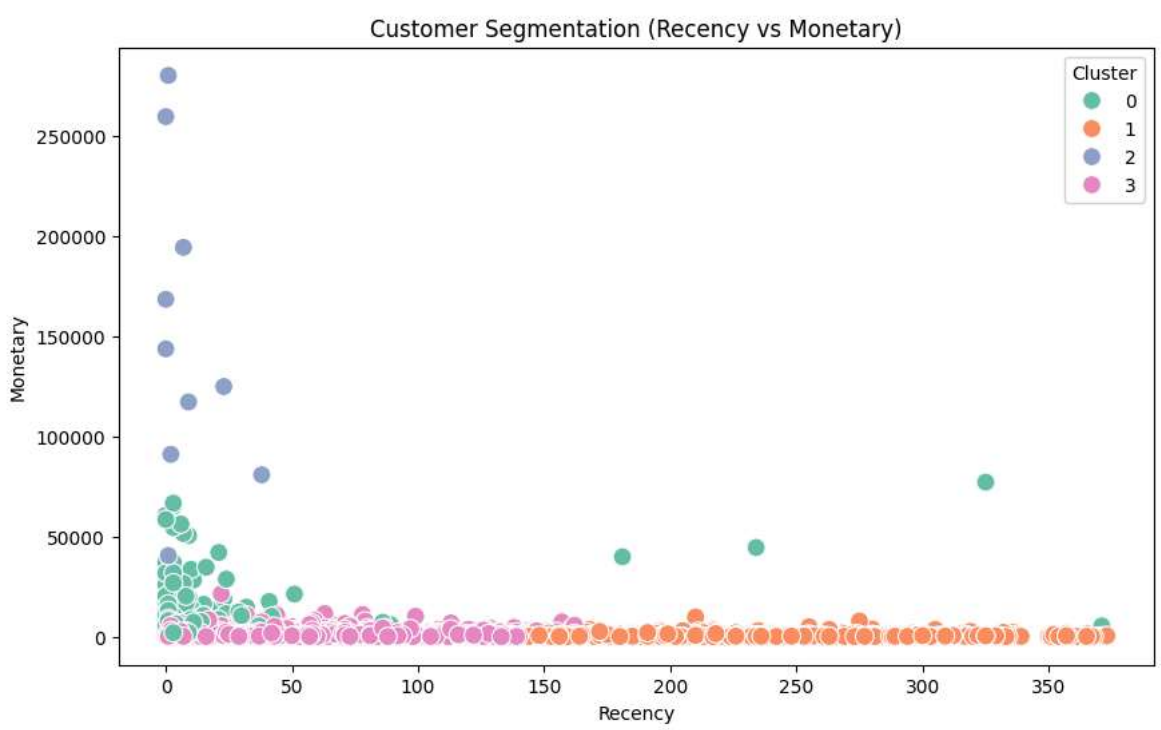
```
from sklearn.preprocessing import StandardScaler

# Scaling
scaler = StandardScaler()
rfm_scaled = scaler.fit_transform(rfm[['Recency', 'Frequency', 'Monetary']])
```

```
from sklearn.cluster import KMeans

# Let's assume 4 customer segments
kmeans = KMeans(n_clusters=4, random_state=42)
rfm['Cluster'] = kmeans.fit_predict(rfm_scaled)

# Show segmented customers
print(rfm.head())
```

```
   CustomerID  Recency  Frequency  Monetary  Cluster
0     12346.0      325          1  77183.60        0
```

```
1    12347.0      1        7  4310.00      3
2    12348.0     74        4  1797.24      3
3    12349.0     18        1  1757.55      3
4    12350.0    309        1   334.40      1
```

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

plt.figure(figsize=(10,6))
sns.scatterplot(data=rfm, x='Recency', y='Monetary', hue='Cluster', palette='Set2', s=100)
plt.title('Customer Segmentation (Recency vs Monetary)')
plt.show()
```




```
cluster_profile = rfm.groupby('Cluster').agg({
    'Recency': 'mean',
    'Frequency': 'mean',
    'Monetary': 'mean',
    'CustomerID': 'count'
}).round(1)

cluster_profile.rename(columns={'CustomerID': 'CustomerCount'}, inplace=True)
print(cluster_profile)
```

```
         Recency  Frequency  Monetary  CustomerCount
Cluster
0           14.7       22.0   12435.1            211
1          247.6        1.6     476.3           1062
2            6.4       82.7  127188.0             13
3           42.9        3.7    1344.3           3053
```

📊 Business Insights:

- 🎯 Cluster 2 includes **high-value, recent customers**. These should be targeted for loyalty programs and premium offers.
- ⚠️ Cluster 1 includes customers who haven't purchased in a long time. Send **re-engagement emails or discounts**.
- 📈 Cluster 0 are frequent buyers. Consider **subscription models or upselling**.
- 🧐 Cluster 3 includes average spenders. Try converting them to loyal ones with **personalized offers**.

✅ Action Plan:

- Segment-specific marketing
- Email campaigns for reactivation

- Premium offers for top segments