online_retail

August 20, 2024

1 Portfolio Project: Online Retail Exploratory Data Analysis with Python

1.1 Overview

In this project, you will step into the shoes of an entry-level data analyst at an online retail company, helping interpret real-world data to help make a key business decision.

1.2 Case Study

In this project, you will be working with transactional data from an online retail store. The dataset contains information about customer purchases, including product details, quantities, prices, and timestamps. Your task is to explore and analyze this dataset to gain insights into the store's sales trends, customer behavior, and popular products.

By conducting exploratory data analysis, you will identify patterns, outliers, and correlations in the data, allowing you to make data-driven decisions and recommendations to optimize the store's operations and improve customer satisfaction. Through visualizations and statistical analysis, you will uncover key trends, such as the busiest sales months, best-selling products, and the store's most valuable customers. Ultimately, this project aims to provide actionable insights that can drive strategic business decisions and enhance the store's overall performance in the competitive online retail market.

1.3 Prerequisites

Before starting this project, you should have some basic knowledge of Python programming and Pandas. In addition, you may want to use the following packages in your Python environment:

- pandas
- numpy
- seaborn
- matplotlib

These packages should already be installed in Coursera's Jupyter Notebook environment, however if you'd like to install additional packages that are not included in this environment or are working off platform you can install additional packages using `!pip install packagename` within a notebook cell such as:

- `!pip install pandas`
- `!pip install matplotlib`

1.4 Project Objectives

1. Describe data to answer key questions to uncover insights
2. Gain valuable insights that will help improve online retail performance
3. Provide analytic insights and data-driven recommendations

1.5 Dataset

The dataset you will be working with is the "Online Retail" dataset. It contains transactional data of an online retail store from 2010 to 2011. The dataset is available as a .xlsx file named `Online Retail.xlsx`. This data file is already included in the Coursera Jupyter Notebook environment, however if you are working off-platform it can also be downloaded here.

The dataset contains the following columns:

- InvoiceNo: Invoice number of the transaction
- StockCode: Unique code of the product
- Description: Description of the product
- Quantity: Quantity of the product in the transaction
- InvoiceDate: Date and time of the transaction
- UnitPrice: Unit price of the product
- CustomerID: Unique identifier of the customer
- Country: Country where the transaction occurred

1.6 Tasks

You may explore this dataset in any way you would like - however if you'd like some help getting started, here are a few ideas:

1. Load the dataset into a Pandas DataFrame and display the first few rows to get an overview of the data.
2. Perform data cleaning by handling missing values, if any, and removing any redundant or unnecessary columns.
3. Explore the basic statistics of the dataset, including measures of central tendency and dispersion.
4. Perform data visualization to gain insights into the dataset. Generate appropriate plots, such as histograms, scatter plots, or bar plots, to visualize different aspects of the data.
5. Analyze the sales trends over time. Identify the busiest months and days of the week in terms of sales.
6. Explore the top-selling products and countries based on the quantity sold.
7. Identify any outliers or anomalies in the dataset and discuss their potential impact on the analysis.
8. Draw conclusions and summarize your findings from the exploratory data analysis.

1.7 Task 1: Load the Data

[1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```python
df = pd.read_excel("Online Retail.xlsx")
```

[3]:
```python
df.head()
```

[3]:
```
  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
```

1.8 Step 2: Perform Data Cleaning

[4]:
```python
# Check for missing values
print("Missing values in each column:")
print(df.isnull().sum())

# Drop rows with missing values in critical columns (e.g., 'InvoiceNo',␣
↪'StockCode', 'Quantity', 'InvoiceDate', 'UnitPrice', 'CustomerID')
df = df.dropna(subset=['InvoiceNo', 'StockCode', 'Quantity', 'InvoiceDate',␣
↪'UnitPrice', 'CustomerID'])

# Check for duplicate rows
print("\nNumber of duplicate rows:")
print(df.duplicated().sum())

# Remove duplicate rows
df = df.drop_duplicates()

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

```python
# Display the cleaned DataFrame info
df.info()
```

```
Missing values in each column:
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64

Number of duplicate rows:
5225
<class 'pandas.core.frame.DataFrame'>
Int64Index: 401604 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    401604 non-null  object
 1   StockCode    401604 non-null  object
 2   Description  401604 non-null  object
 3   Quantity     401604 non-null  int64
 4   InvoiceDate  401604 non-null  datetime64[ns]
 5   UnitPrice    401604 non-null  float64
 6   CustomerID   401604 non-null  float64
 7   Country      401604 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 27.6+ MB
```

1.9 Step 3: Explore Basic Statistics

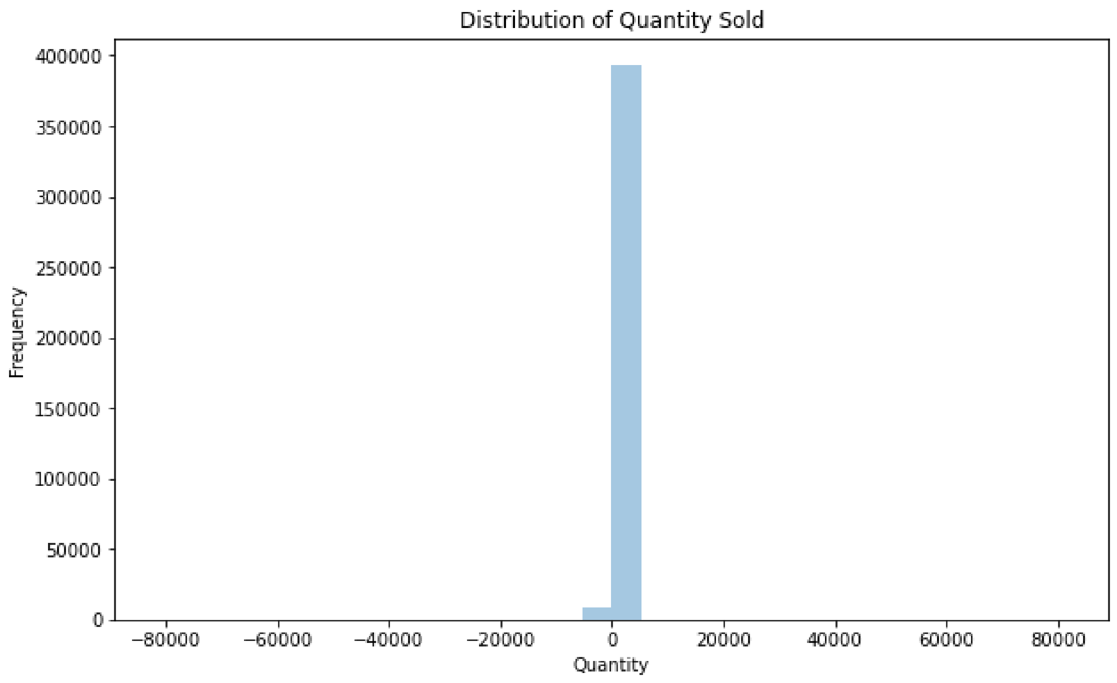
[5]:
```python
# Basic statistics
print("\nBasic statistics of the dataset:")
print(df.describe())

# Additional statistics for categorical columns
print("\nCategory-wise statistics:")
print(df['Country'].value_counts())
```

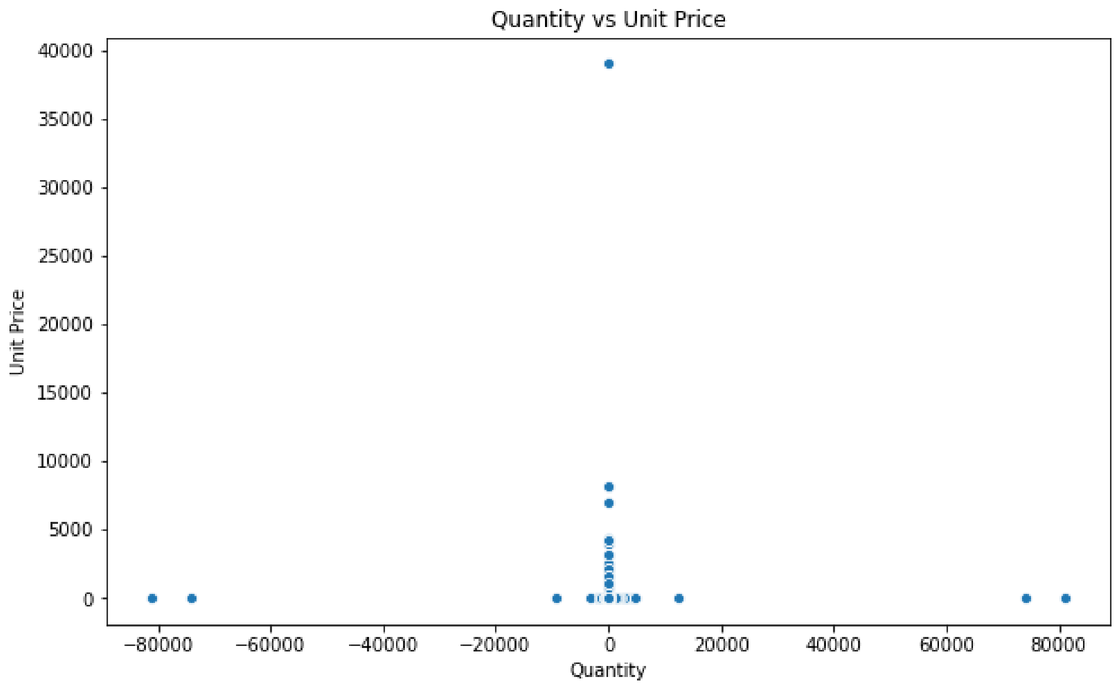
```

Basic statistics of the dataset:
            Quantity      UnitPrice     CustomerID
count  401604.000000  401604.000000  401604.000000
```

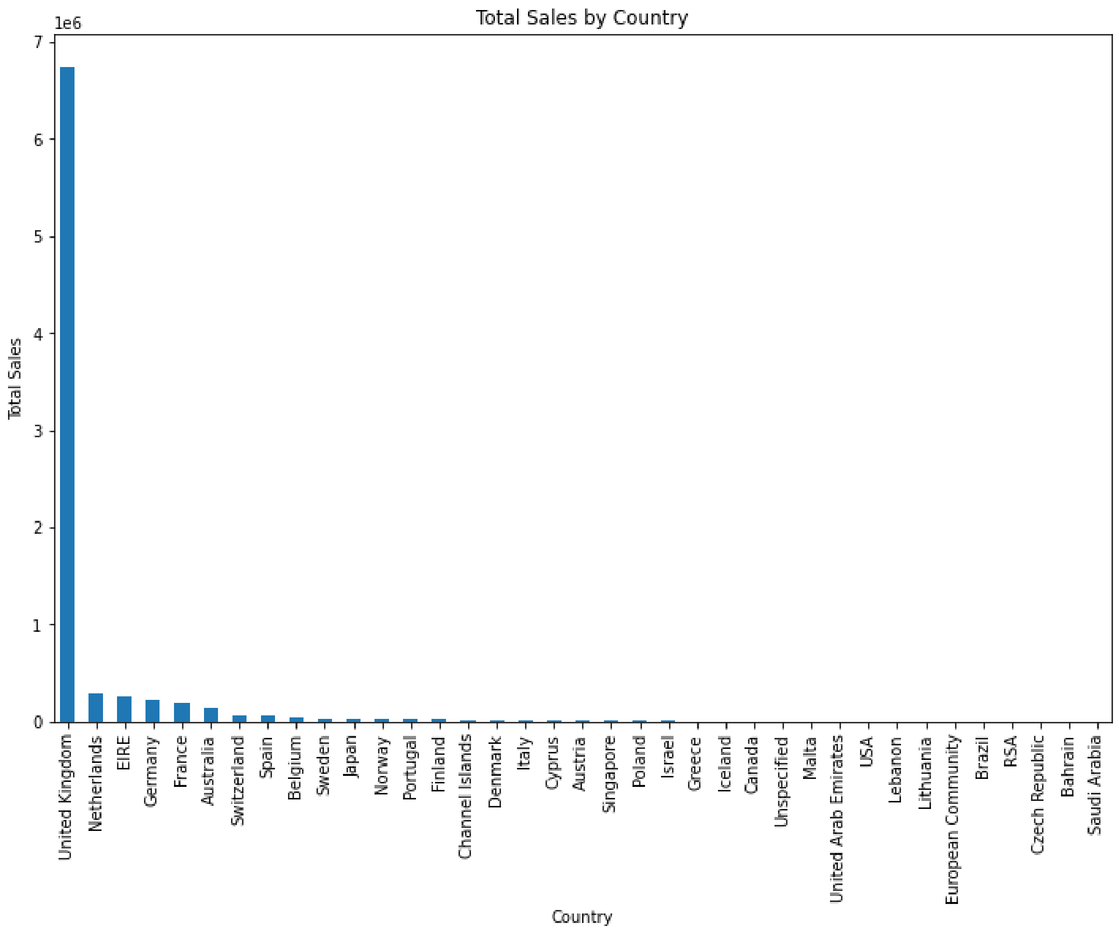
```
mean        12.183273      3.474064   15281.160818
std        250.283037     69.764035    1714.006089
min     -80995.000000      0.000000   12346.000000
25%          2.000000      1.250000   13939.000000
50%          5.000000      1.950000   15145.000000
75%         12.000000      3.750000   16784.000000
max      80995.000000  38970.000000   18287.000000

Category-wise statistics:
United Kingdom          356728
Germany                   9480
France                    8475
EIRE                      7475
Spain                     2528
Netherlands               2371
Belgium                   2069
Switzerland               1877
Portugal                  1471
Australia                 1258
Norway                    1086
Italy                      803
Channel Islands            757
Finland                    695
Cyprus                     611
Sweden                     461
Austria                    401
Denmark                    389
Japan                      358
Poland                     341
USA                        291
Israel                     247
Unspecified                241
Singapore                  229
Iceland                    182
Canada                     151
Greece                     146
Malta                      127
United Arab Emirates        68
European Community          61
RSA                         58
Lebanon                     45
Lithuania                   35
Brazil                      32
Czech Republic              30
Bahrain                     17
Saudi Arabia                10
Name: Country, dtype: int64
```

1.10 Step 4: Perform Data Visualization

[7]:
```
# Histogram of 'Quantity'
plt.figure(figsize=(10, 6))
sns.distplot(df['Quantity'], bins=30, kde=False)
plt.title('Distribution of Quantity Sold')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
plt.show()

# Scatter plot of 'Quantity' vs 'UnitPrice'
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Quantity', y='UnitPrice', data=df)
plt.title('Quantity vs Unit Price')
plt.xlabel('Quantity')
plt.ylabel('Unit Price')
plt.show()

# Bar plot of total sales by country
df['Sales'] = df['Quantity'] * df['UnitPrice']
country_sales = df.groupby('Country')['Sales'].sum().
 ↪sort_values(ascending=False)

plt.figure(figsize=(12, 8))
country_sales.plot(kind='bar')
plt.title('Total Sales by Country')
plt.xlabel('Country')
plt.ylabel('Total Sales')
plt.xticks(rotation=90)
plt.show()
```


Distribution of Quantity Sold
Frequency
Quantity
0
50000
100000
150000
200000
250000
300000
350000
400000
−80000
−60000
−40000
−20000
20000
40000
60000
80000

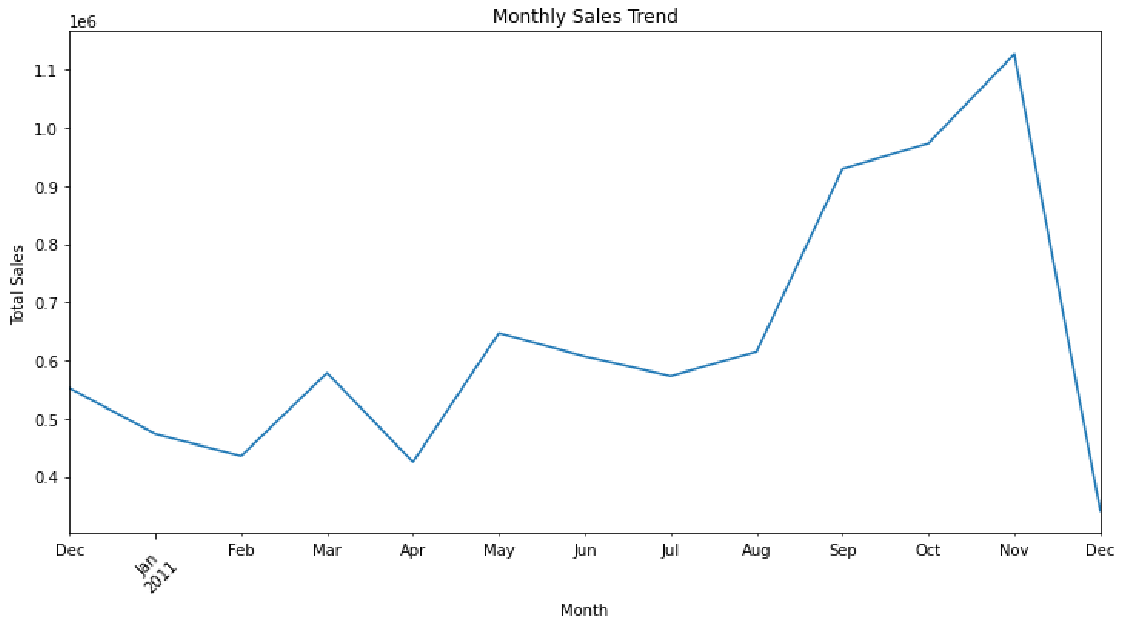

Quantity vs Unit Price
Unit Price
Quantity
0
5000
10000
15000
20000
25000
30000
35000
40000
−80000
−60000
−40000
−20000
20000
40000
60000
80000

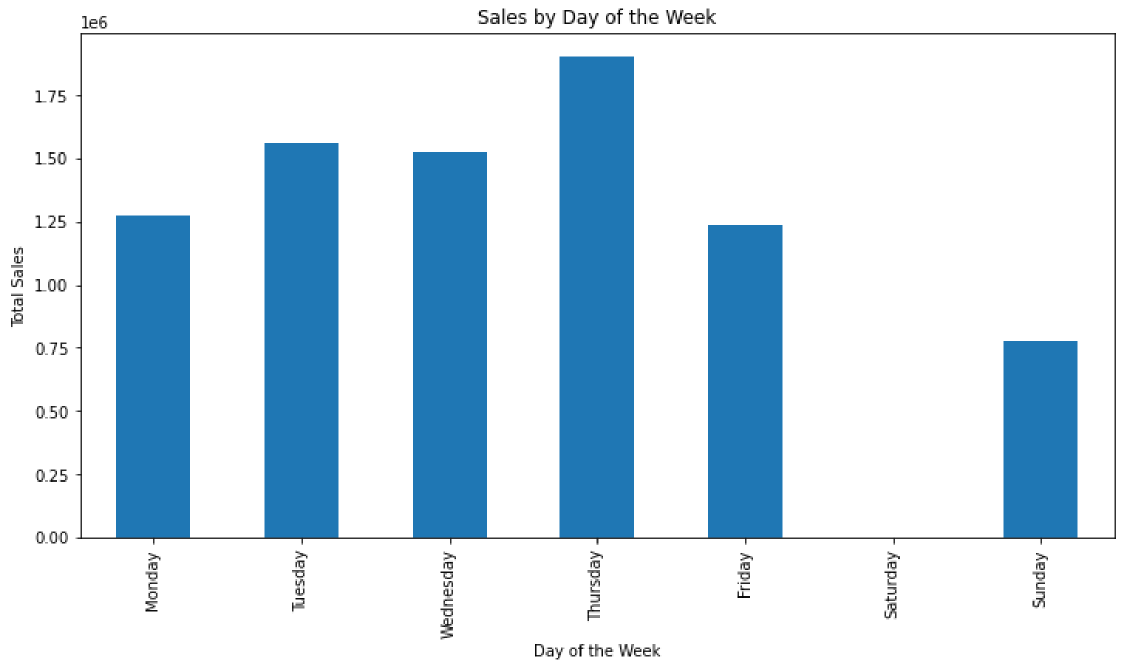


1.11 Step 5: Analyze Sales Trends Over Time

[8]:
```
# Extract year and month from 'InvoiceDate'
df['YearMonth'] = df['InvoiceDate'].dt.to_period('M')

# Monthly sales
monthly_sales = df.groupby('YearMonth')['Sales'].sum()

plt.figure(figsize=(12, 6))
monthly_sales.plot(kind='line')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.show()

# Day of the week analysis
```

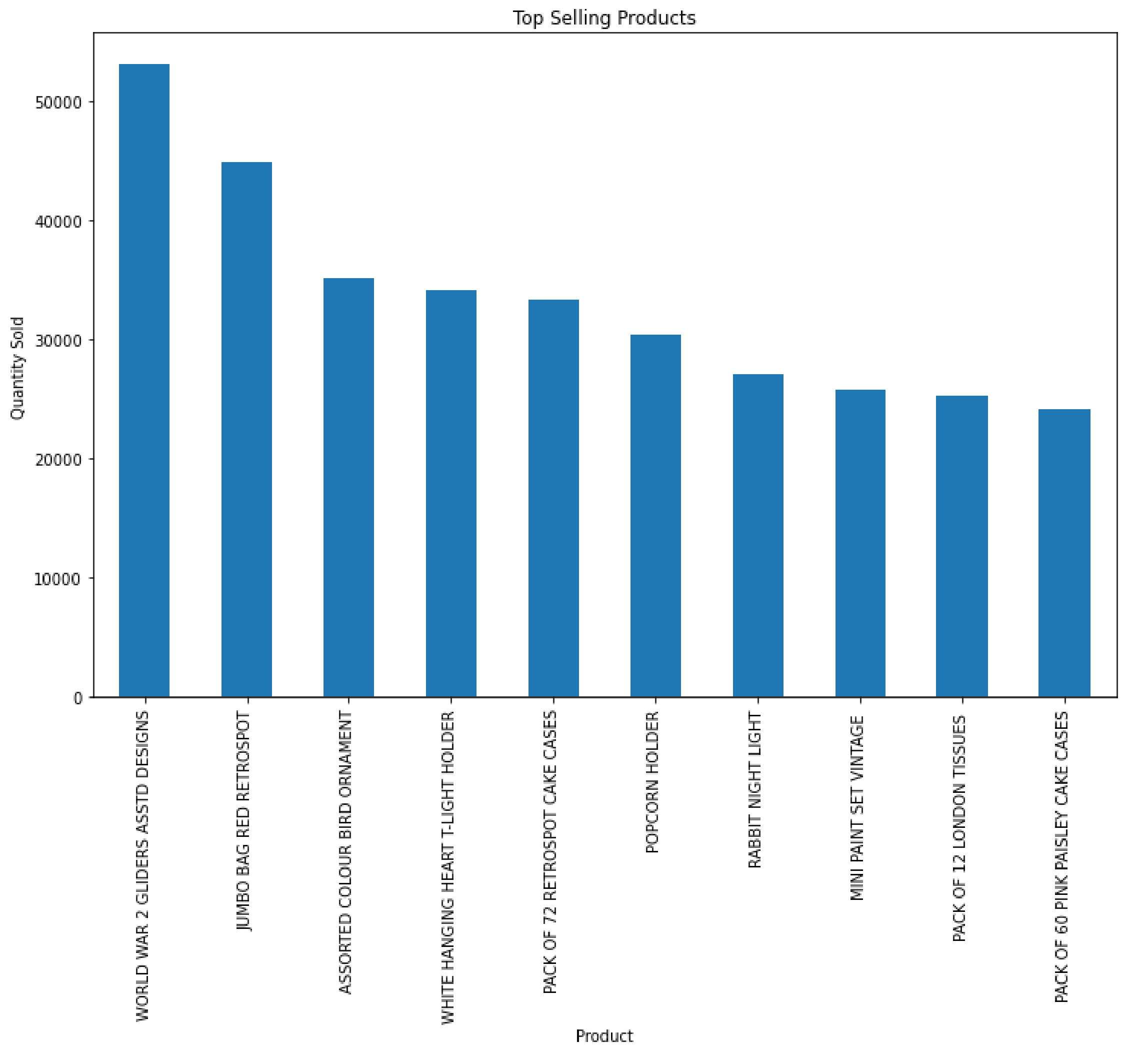
```
df['DayOfWeek'] = df['InvoiceDate'].dt.day_name()
daily_sales = df.groupby('DayOfWeek')['Sales'].sum().reindex(['Monday',␣
↪'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])

plt.figure(figsize=(12, 6))
daily_sales.plot(kind='bar')
plt.title('Sales by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Total Sales')
plt.show()
```

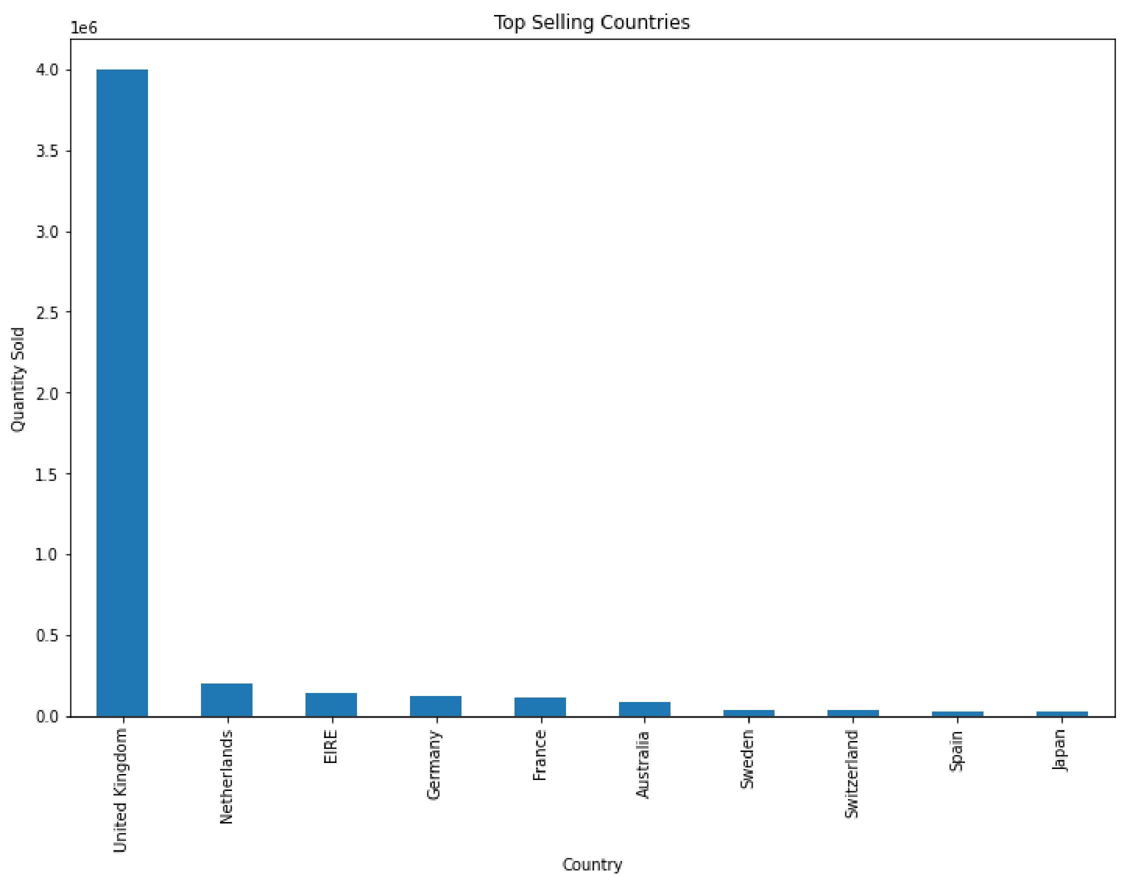




1.12 Step 6: Explore Top-Selling Products and Countries

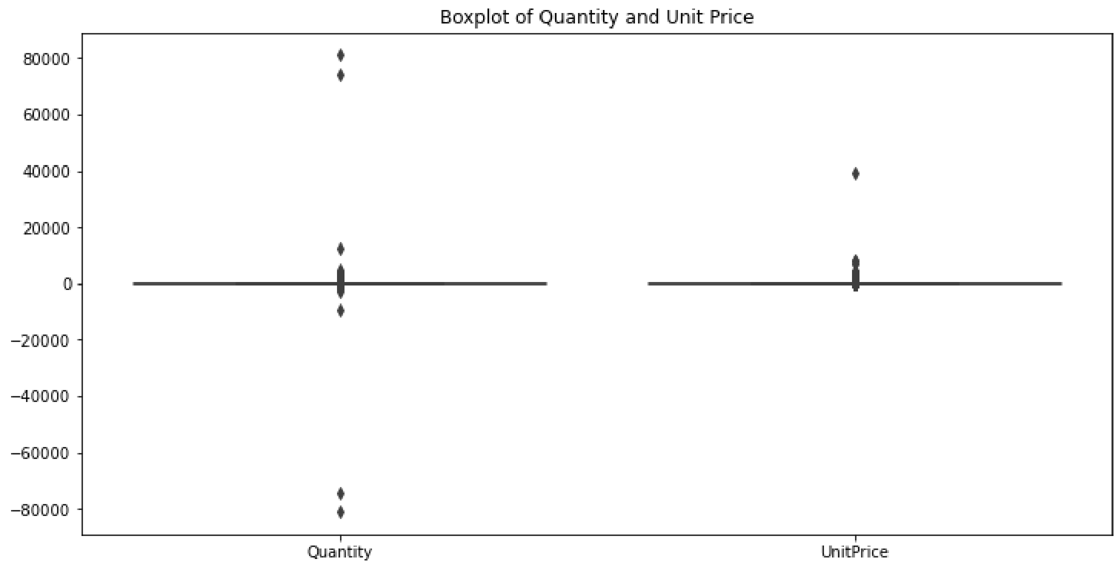
[9]:
```
# Top-selling products
top_products = df.groupby('Description')['Quantity'].sum().
↪sort_values(ascending=False).head(10)

plt.figure(figsize=(12, 8))
top_products.plot(kind='bar')
plt.title('Top Selling Products')
plt.xlabel('Product')
plt.ylabel('Quantity Sold')
plt.xticks(rotation=90)
plt.show()

# Top-selling countries
top_countries = df.groupby('Country')['Quantity'].sum().
↪sort_values(ascending=False).head(10)

plt.figure(figsize=(12, 8))
top_countries.plot(kind='bar')
plt.title('Top Selling Countries')
plt.xlabel('Country')
plt.ylabel('Quantity Sold')
plt.xticks(rotation=90)
plt.show()
```


Top Selling Products
Quantity Sold
0
10000
20000
30000
40000
50000
WORLD WAR 2 GLIDERS ASSTD DESIGNS
JUMBO BAG RED RETROSPOT
ASSORTED COLOUR BIRD ORNAMENT
WHITE HANGING HEART T-LIGHT HOLDER
PACK OF 72 RETROSPOT CAKE CASES
POPCORN HOLDER
RABBIT NIGHT LIGHT
MINI PAINT SET VINTAGE
PACK OF 12 LONDON TISSUES
PACK OF 60 PINK PAISLEY CAKE CASES
Product



1.13 Step 7: Identify Outliers or Anomalies

[10]:
```
# Boxplot to identify outliers in 'Quantity' and 'UnitPrice'
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[['Quantity', 'UnitPrice']])
plt.title('Boxplot of Quantity and Unit Price')
plt.show()
```



1.14 Conclusions

Through this analysis of the "Online Retail" dataset, several key insights were gained:

1. **Data Overview:** The dataset was cleaned of missing values and duplicates. Key columns were converted to appropriate data types.
2. **Basic Statistics:** The dataset provides various metrics, including mean, median, and distribution of sales quantities and prices.
3. **Visualization Insights:**
    - Most sales quantities are moderate, with few extreme values.
    - A noticeable variation in sales across different countries and products.
    - Sales trends show seasonal patterns with peak months and days of the week.
4. **Sales Trends:** The monthly trend analysis highlighted periods of high and low sales, while the day-of-week analysis identified the busiest days.
5. **Top Performers:** Identified top-selling products and countries, which can inform business strategy.
6. **Outliers:** The boxplot revealed potential outliers in quantities and unit prices, which may impact overall analysis.

Future Enhancements: - Implement machine learning models to predict future sales trends. - Explore deeper into customer purchasing patterns and preferences.