online_retail

October 11, 2025

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

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
df = pd.read_excel("Online Retail.xlsx")
```

[4]:
```
df.head()
```

[4]:
```
  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
```

[5]:
```
df.tail()
```

[5]:
```
        InvoiceNo StockCode                       Description  Quantity  \
541904    581587     22613      PACK OF 20 SPACEBOY NAPKINS        12
541905    581587     22899     CHILDREN'S APRON DOLLY GIRL          6
541906    581587     23254    CHILDRENS CUTLERY DOLLY GIRL          4
541907    581587     23255  CHILDRENS CUTLERY CIRCUS PARADE         4
541908    581587     22138    BAKING SET 9 PIECE RETROSPOT          3

                InvoiceDate  UnitPrice  CustomerID Country
541904 2011-12-09 12:50:00       0.85     12680.0  France
541905 2011-12-09 12:50:00       2.10     12680.0  France
541906 2011-12-09 12:50:00       4.15     12680.0  France
541907 2011-12-09 12:50:00       4.15     12680.0  France
541908 2011-12-09 12:50:00       4.95     12680.0  France
```

[6]:
```
df.shape
```

[6]:
```
(541909, 8)
```

[7]:
```
# Checking data types.
df.dtypes
```

[7]:
```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
dtype: object
```

[8]:
```
# Checking for duplicates.
duplicate_rows_df = df[df.duplicated()]
print("number of duplicate rows", duplicate_rows_df.shape)
```

```
number of duplicate rows (5268, 8)
```

[9]:
```
# Count the number of rows.
df.count()
```

[9]:
```
InvoiceNo      541909
StockCode      541909
Description    540455
Quantity       541909
InvoiceDate    541909
UnitPrice      541909
CustomerID     406829
Country        541909
dtype: int64
```

[10]:
```
# Show duplicated rows.
duplicate_rows_df = df[df.duplicated()]
print("number of duplicate rows: ", duplicate_rows_df.shape)
```

```
number of duplicate rows:  (5268, 8)
```

[11]:
```
df = df.drop_duplicates()
```

[12]:
```
# Ensure that the duplicated rows have been removed.
duplicate_rows_df = df[df.duplicated()]
print("number of duplicate rows: ", duplicate_rows_df.shape)
```

```
number of duplicate rows:  (0, 8)
```

[13]:
```
# Missing or Null values.
print(df.isnull().sum())
```

```
InvoiceNo          0
StockCode          0
```

```
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135037
Country             0
dtype: int64
```

[14]:
```
# Dropping missing or null values.
df = df.dropna()
```

[15]:
```
# Verifying that missing and null values are dropped.
print(df.isnull().sum())
```

```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

[16]:
```
df.describe()
```

[16]:

	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

[17]:
```
df.info()
```

```
<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
```

[18]:
```
import datetime as dt
```

[19]:
```
# Convert InvoiceDate to datetime.
df['InvoiceDate']=pd.to_datetime(df['InvoiceDate'])
```

[20]:
```
# Create a month column.
df['month'] = df['InvoiceDate'].dt.month_name()

# Create a day column
df['day'] = df['InvoiceDate'].dt.day_name()
```

[21]:
```
df.head()
```

[21]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	month	day
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	December	Wednesday
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	December	Wednesday
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	December	Wednesday
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	December	Wednesday
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	December	Wednesday

[64]:
```
# Measure of Central Tendency
# Calculate Mean Sales
mean_sales = df['Sales'].mean()
mean_sales = round(mean_sales, 2)
print(mean_sales)
```

```
20.61
```

[65]:
```
# Calculate Median Sales
median_sales = df['Sales'].median()
median_sales = round(median_sales, 2)
print(median_sales)
```

```
11.7
```

[67]:
```
# Calculate Mode
mode_sales = df['Sales'].mode()[0]
print(mode_sales)
```

```
15.0
```

[68]:
```
# Calculate Range
range_sales = df['Sales'].max() - df['Sales'].min()
print(range_sales)
```

```
336939.2
```

[70]:
```
# Calculate Variance
variance_sales = df['Sales'].var()
variance_sales = round(variance_sales, 2)
print(variance_sales)
```

```
185203.03
```

[72]:
```
# Calculate STD
std_sales = df['Sales'].std()
std_sales = round(std_sales, 2)
print(std_sales)
```

```
430.35
```

[22]:
```
# Get total number of transaction for each month.
monthly_transactions = df['month'].value_counts()
monthly_transactions
```

```
November     64232
October      49928
December     43736
September    40459
May          28661
June         27576
March        27516
August       27444
July         27256
April        22988
January      21670
```

```
February     20138
Name: month, dtype: int64
```

[23]:
```
# Reorder the monthly transactions list so months go in order
month_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July',␣
 ↪'August',
               'September', 'October', 'November', 'December']

monthly_transactions = monthly_transactions.reindex(index=month_order)
monthly_transactions
```

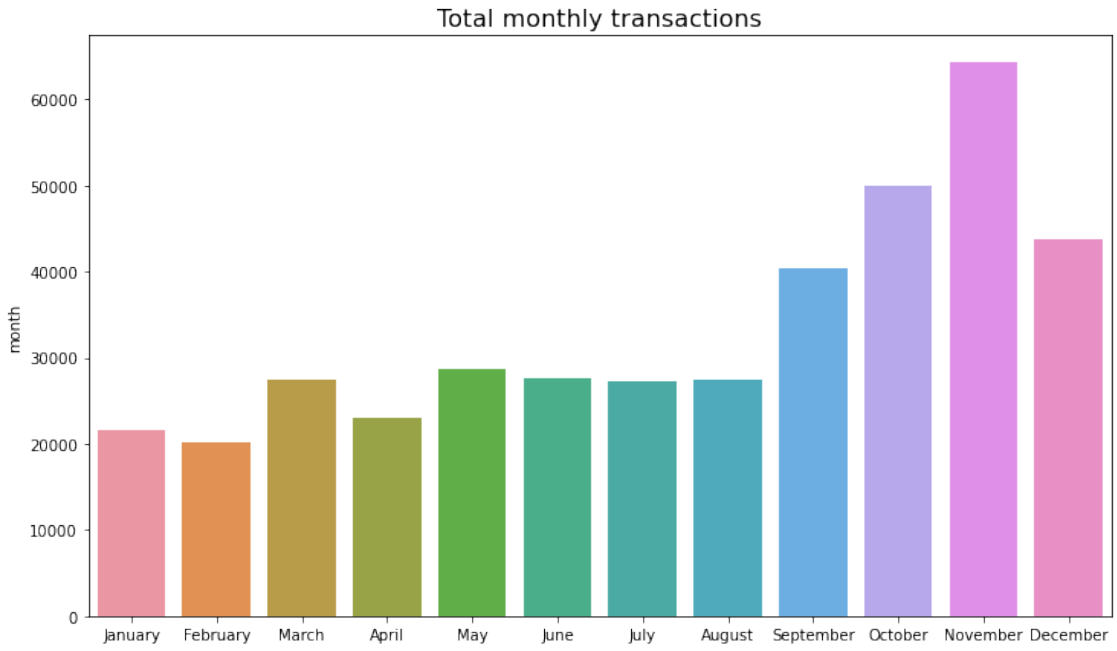
[23]:
```
January      21670
February     20138
March        27516
April        22988
May          28661
June         27576
July         27256
August       27444
September    40459
October      49928
November     64232
December     43736
Name: month, dtype: int64
```

[24]:
```
# Show the index.
monthly_transactions.index
```

[24]:
```
Index(['January', 'February', 'March', 'April', 'May', 'June', 'July',
       'August', 'September', 'October', 'November', 'December'],
      dtype='object')
```

[25]:
```
# Create a bar plot of total transactions per month.
plt.figure(figsize=(12,7))
ax = sns.barplot(x=monthly_transactions.index, y=monthly_transactions)
ax.set_xticklabels(month_order)
plt.title('Total monthly transactions', fontsize=16)
```

[25]:
```
Text(0.5, 1.0, 'Total monthly transactions')
```



[ ]:
```
The busiest month for the year is November.
```

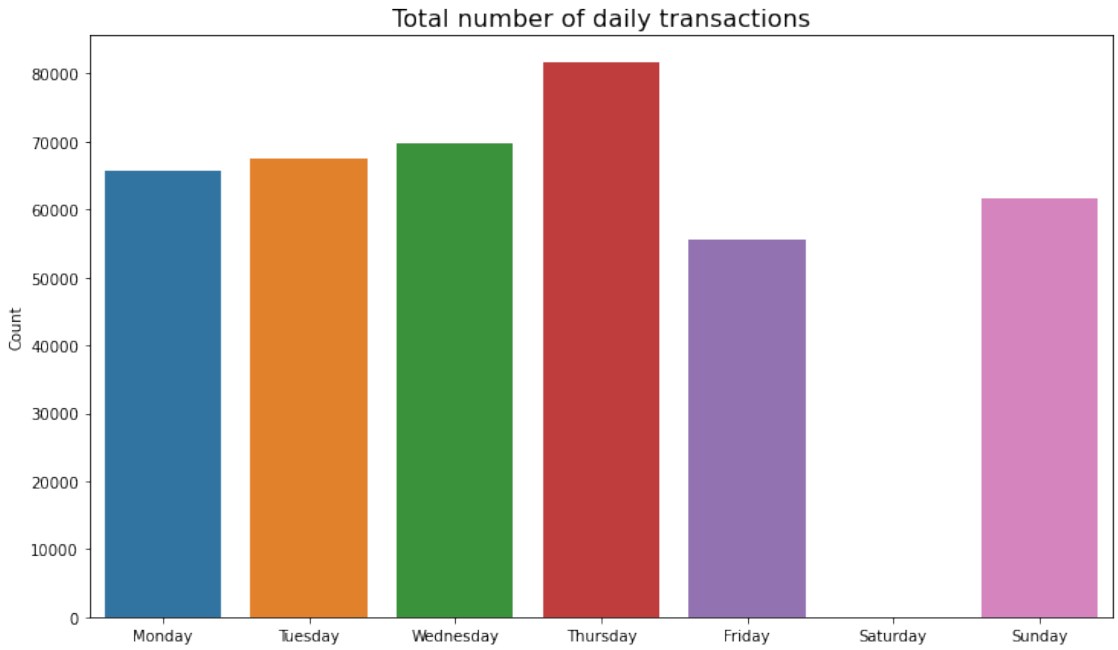
[26]:
```
# Daily transactions
daily_transactions = df['day'].value_counts()
day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',␣
 ↪'Saturday', 'Sunday']
daily_transactions = daily_transactions.reindex(index=day_order)
daily_transactions
```

[26]:
```
Monday       65715.0
Tuesday      67376.0
Wednesday    69753.0
Thursday     81575.0
Friday       55512.0
Saturday         NaN
Sunday       61673.0
Name: day, dtype: float64
```

[ ]:
```
The busiest day is Thursday.
```

[27]:
```
# Create a barplot for daily transactions.
plt.figure(figsize=(12,7))
ax = sns.barplot(x=daily_transactions.index, y=daily_transactions)
ax.set_xticklabels(day_order)
ax.set_ylabel('Count')
plt.title('Total number of daily transactions', fontsize=16)
```

[27]:
```
Text(0.5, 1.0, 'Total number of daily transactions')
```



[ ]:
```
The busiest day is Thursday. No transactions are completed on Saturdays.
```

[28]:
```
# Add a Sales column.
df['Sales'] = df['Quantity'] * df['UnitPrice']
```

[29]:
```
df.head()
```

[29]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	month	day	Sales
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	December	Wednesday	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	December	Wednesday	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	December		
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	December		
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	December		

```
2  Wednesday  22.00
3  Wednesday  20.34
4  Wednesday  20.34
```

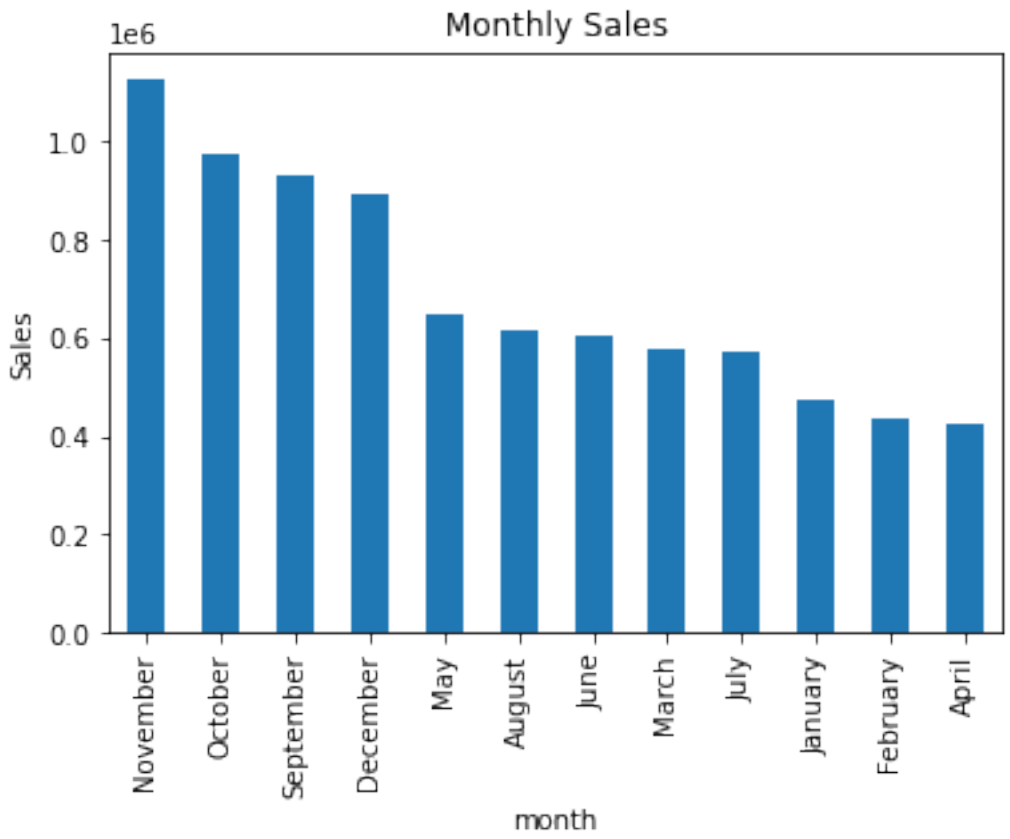
[30]:
```
# Sales by month

monthly_sales = df.groupby('month')['Sales'].sum().sort_values(ascending=False)
monthly_sales
```

[30]:
```
month
November     1126815.070
October       973306.380
September     929356.232
December      893912.290
May           647011.670
August        615078.090
June          606862.520
March         578576.210
July          573112.321
January       473731.900
February      435534.070
April         425222.671
Name: Sales, dtype: float64
```

[31]:
```
# Plot monthly sales

monthly_sales.plot(kind='bar')
plt.ylabel('Sales')
plt.title('Monthly Sales')
plt.show()
```



[ ]:
```
The busiest month is November.
```

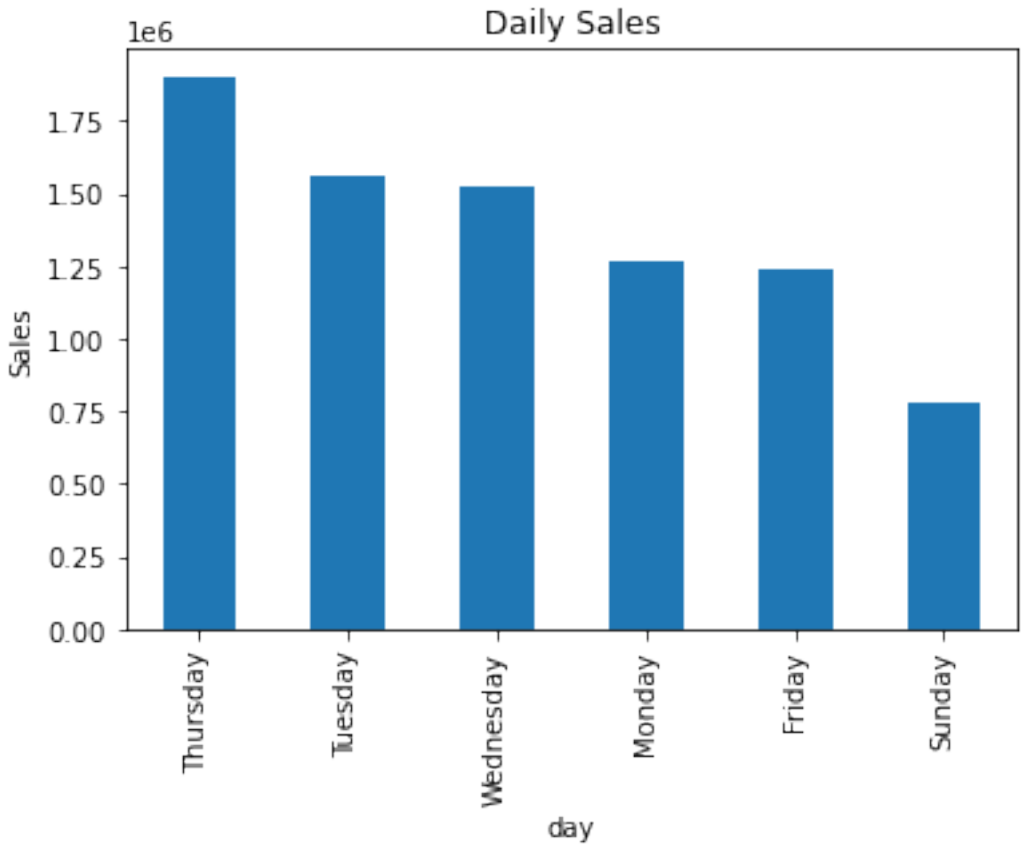
[32]:
```
# Daily sales.

daily_sales = df.groupby('day')['Sales'].sum().sort_values(ascending=False)
daily_sales
```

[32]:
```
day
Thursday     1902316.050
Tuesday      1562715.681
Wednesday    1526440.000
Monday       1271078.601
Friday       1238556.741
Sunday        777412.351
Name: Sales, dtype: float64
```

[32]:
```
# Plot daily sales.
daily_sales.plot(kind='bar')
plt.ylabel('Sales')
plt.title('Daily Sales')
```

```
plt.show()
```



[ ]:
```
Thursday is the busiest day.
```

[33]:
```
# Creating a list of countries and total sales.
country_sales = df.groupby('Country')['Sales'].sum().reset_index()
```

[42]:
```
# Sort country_sales in descending order
country_sales = country_sales.sort_values(by='Sales', ascending = False)
```

[57]:
```
print(country_sales.head(10))
```

```
           Country      Sales
0        Australia  137009.77
1          Austria   10154.32
2          Bahrain     548.40
3          Belgium   40910.96
4           Brazil    1143.60
5           Canada    3666.38
6  Channel Islands   20076.39
```

```
7           Cyprus   12858.76
8   Czech Republic     707.72
9          Denmark   18768.14
```

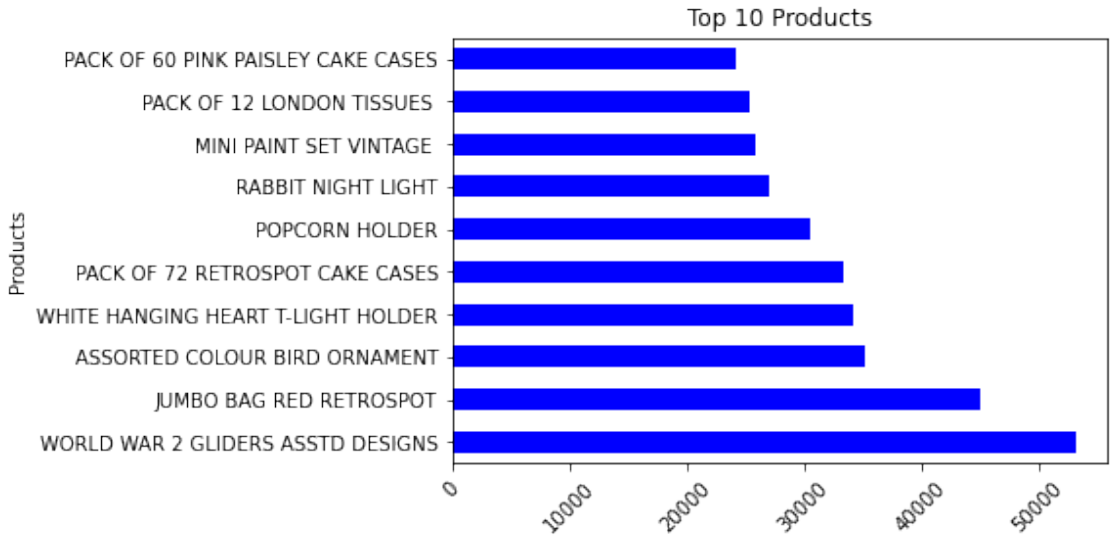
[58]:
```
print('Number of Countries', df['Country'].nunique())
```

```
Number of Countries 37
```

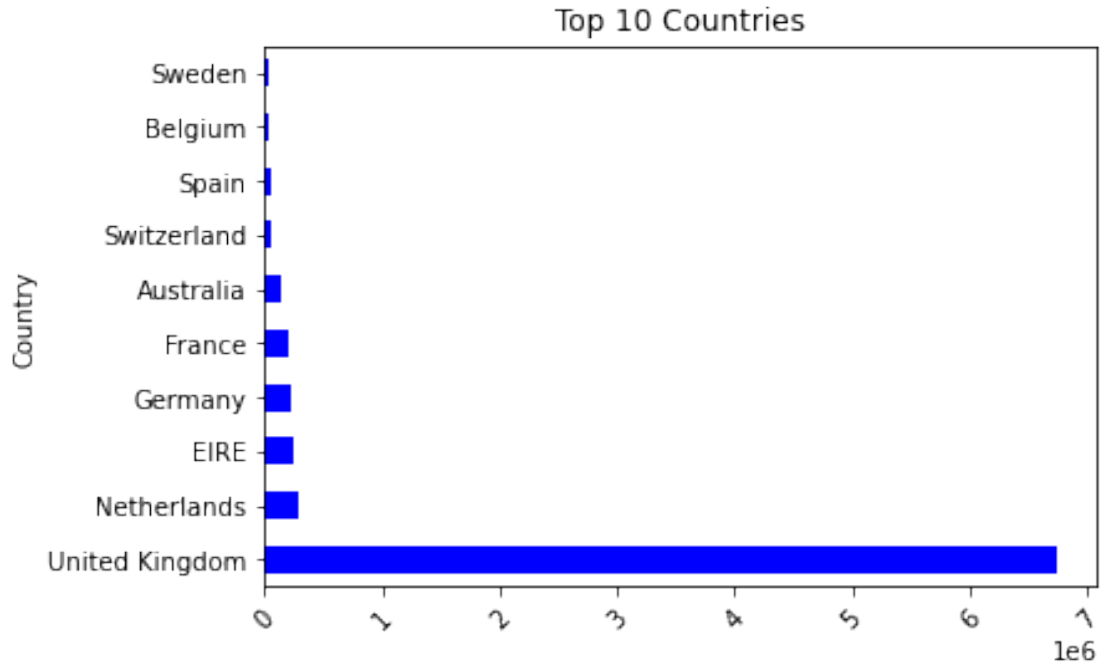
[59]:
```
# Total quantity of top selling products
top_selling_products = df.groupby('Description')['Quantity'].sum().
 ↪sort_values(ascending=False)
top_selling_products.head(10)
```

[59]:
```
Description
WORLD WAR 2 GLIDERS ASSTD DESIGNS     53119
JUMBO BAG RED RETROSPOT               44963
ASSORTED COLOUR BIRD ORNAMENT         35215
WHITE HANGING HEART T-LIGHT HOLDER    34128
PACK OF 72 RETROSPOT CAKE CASES       33386
POPCORN HOLDER                        30492
RABBIT NIGHT LIGHT                    27045
MINI PAINT SET VINTAGE                25880
PACK OF 12 LONDON TISSUES             25305
PACK OF 60 PINK PAISLEY CAKE CASES    24129
Name: Quantity, dtype: int64
```

[60]:
```
# Create a bar plot of the top selling products.
top_selling_products[0:10].plot(kind='barh',color='blue')
plt.ylabel('Products')
plt.xticks(rotation=45)
plt.title('Top 10 Products')
plt.show()
```



[61]:
```
# top_10_countries = df.groupby('Country')['Sales'].sum().
 ↪sort_values(ascending=False).head(10)
country_sales = df.groupby('Country')['Sales'].sum().reset_index()
# Visualize the top 10 countries.
top_10_countries[0:10].plot(kind='barh',color='blue')
plt.ylabel('Country')
plt.xticks(rotation=45)
plt.title('Top 10 Countries')
plt.show()
```



[ ]:
```
United Kindom is the largest market, now check for how many customers are from␣
↪there.
```

[48]:
```
# How many customers are in the United Kingdom.
df[df['Country']=='United Kingdom']['CustomerID'].nunique()
```

[48]:
```
3950
```

[ ]:
```
The United Kingdom is the largest market with 3950 customers.
```

[ ]:
```
Explore to know more about the customers in the United Kingdom.
```

[49]:
```
df_uk = df[df['Country']=='United Kingdom']
df_uk.describe()
```

[49]:

	Quantity	UnitPrice	CustomerID	Sales
count	356728.000000	356728.000000	356728.000000	356728.000000
mean	11.198644	3.268255	15543.795284	18.914008
std	264.998044	71.162330	1594.286219	455.157029
min	-80995.000000	0.000000	12346.000000	-168469.600000
25%	2.000000	1.250000	14191.000000	3.900000
50%	4.000000	1.950000	15513.000000	10.200000
75%	12.000000	3.750000	16931.000000	17.700000
max	80995.000000	38970.000000	18287.000000	168469.600000

[50]:
```
print("Number of Transactions: ", df_uk['InvoiceNo'].nunique())
print("Number of products bought: ", df_uk['StockCode'].nunique())
print("Number of Customers: ", df_uk['CustomerID'].nunique())
```

```
Number of Transactions:  19857
Number of products bought:  3661
Number of Customers:  3950
```

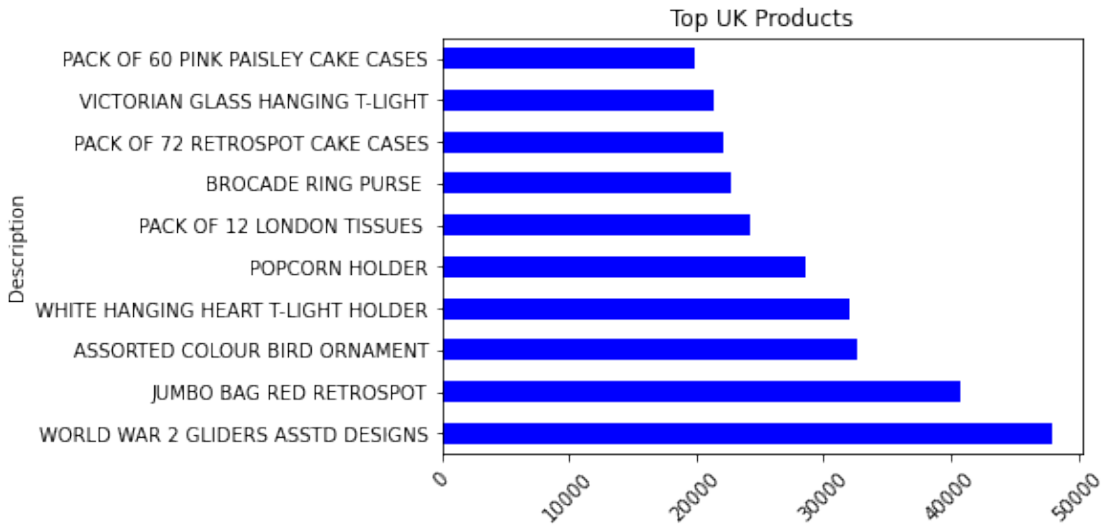
[ ]:
```
What are the most purchased products in the United Kingdom?
```

[51]:
```
uk_products = df_uk.groupby('Description')['Quantity'].sum().
↪sort_values(ascending=False)
uk_products.head(10)
```

[51]:
```
Description
WORLD WAR 2 GLIDERS ASSTD DESIGNS     47886
JUMBO BAG RED RETROSPOT               40777
ASSORTED COLOUR BIRD ORNAMENT         32580
WHITE HANGING HEART T-LIGHT HOLDER    32079
POPCORN HOLDER                        28550
```

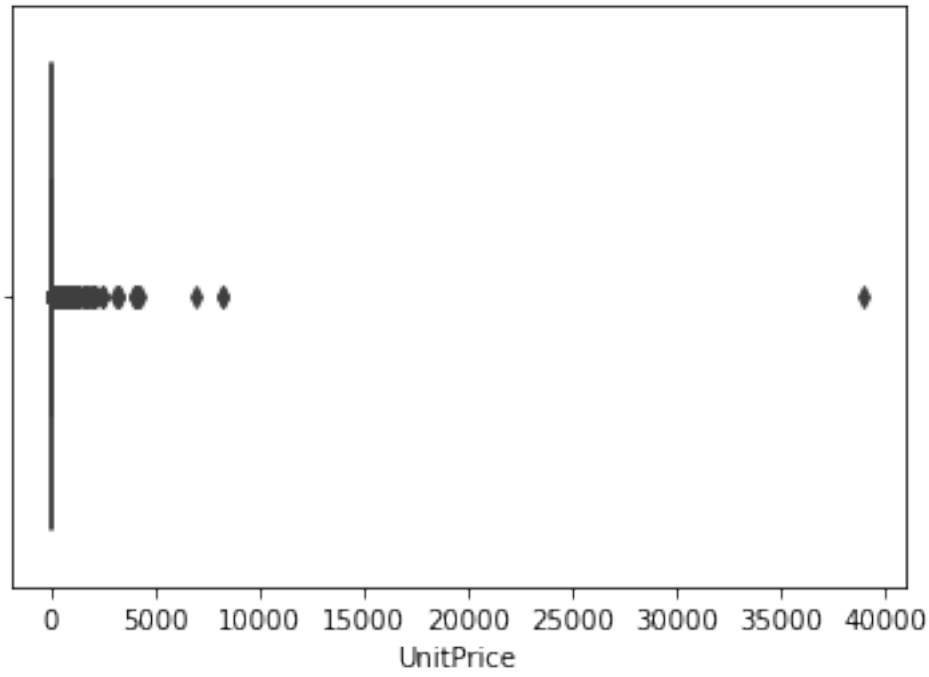
```
PACK OF 12 LONDON TISSUES             24297
BROCADE RING PURSE                    22672
PACK OF 72 RETROSPOT CAKE CASES       22182
VICTORIAN GLASS HANGING T-LIGHT       21427
PACK OF 60 PINK PAISLEY CAKE CASES    19882
Name: Quantity, dtype: int64
```

[64]:
```
uk_products [:10].plot(kind='barh', color='blue')
plt.ylabel('Description')
plt.xticks(rotation=45)
plt.title('Top UK Products')
plt.show()
```



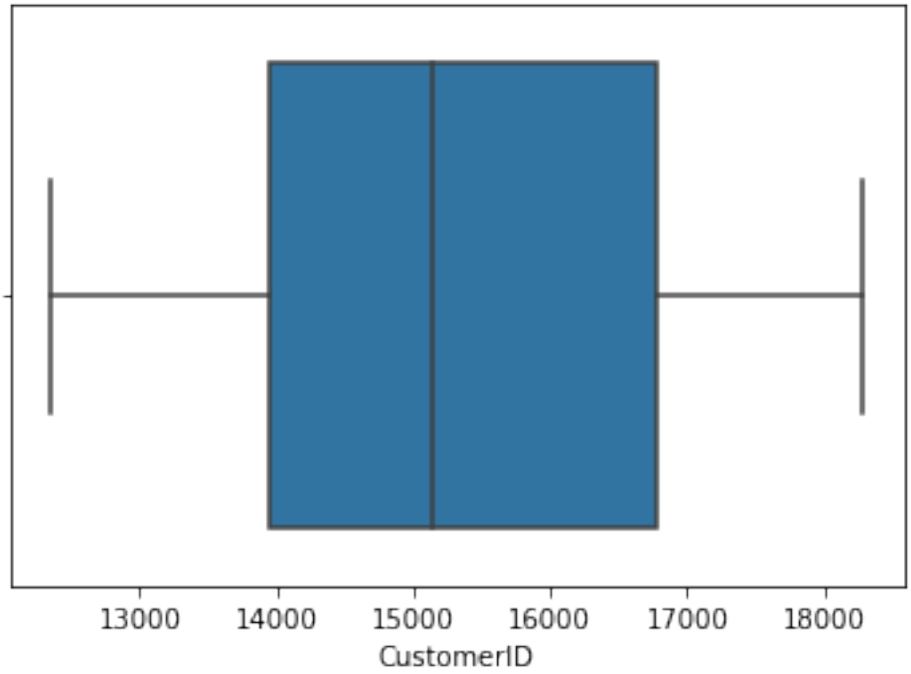
[52]:
```
sns.boxplot(x = df['UnitPrice'])
```

[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7103080b1b10>
```



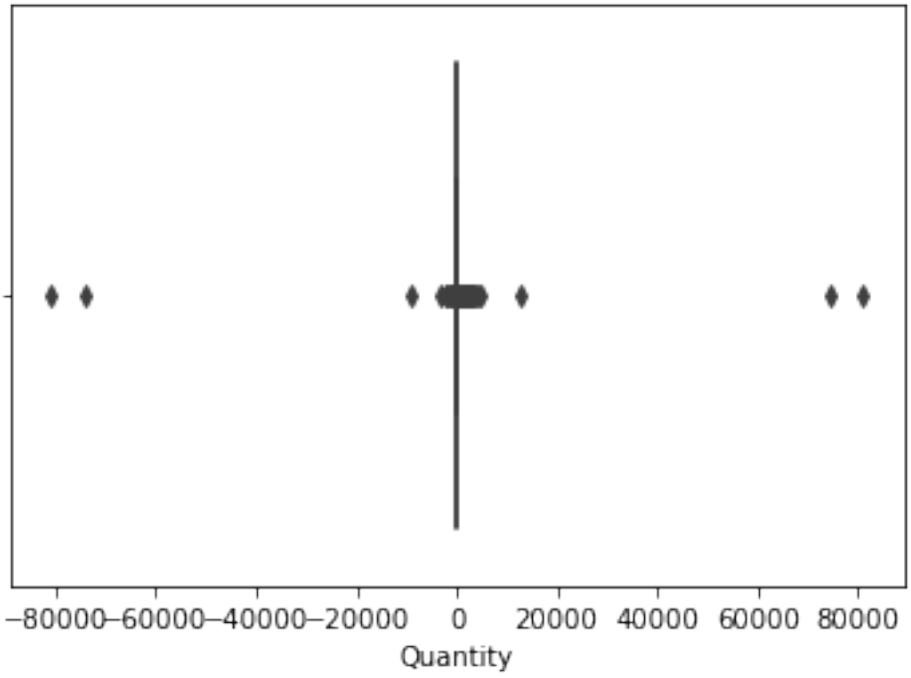
[53]:
```
sns.boxplot(x = df['CustomerID'])
```

[53]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x710309bf8d10>
```



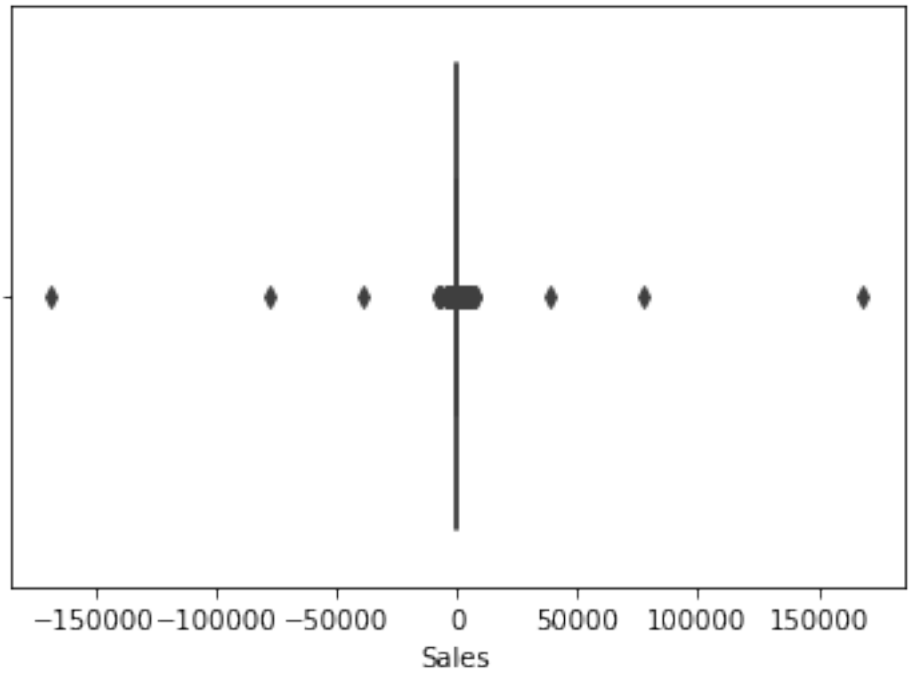
[54]:
```
sns.boxplot(x = df['Quantity'])
```

[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x71030a9d1190>
```

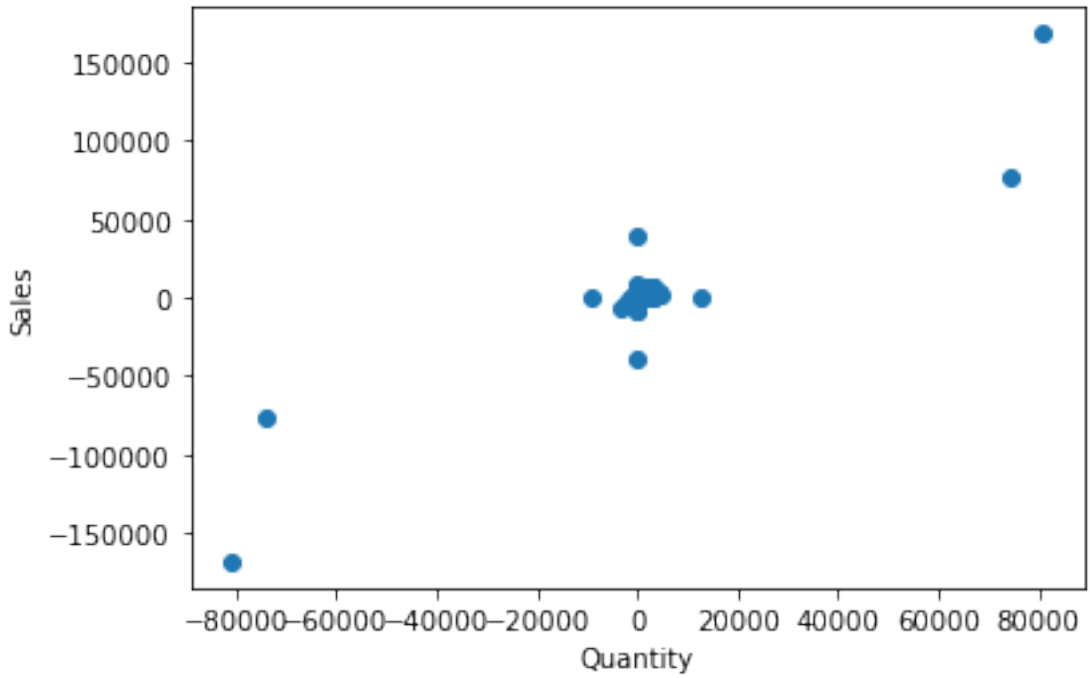


[55]:
```
sns.boxplot(x = df['Sales'])
```

[55]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x71030d2835d0>
```



[56]:
```
plt.scatter(df['Quantity'], df['Sales'])
plt.xlabel('Quantity')
plt.ylabel('Sales')
plt.show()
```



[ ]:
```
Summary
1. The busiest sales months are:
    November - $1126815.07
    October - $973306.38
    September - $929356.232


2. The busiest days of the week and sales:
    Thursday     $1902316.050
    Tuesday      $1562715.681
    Wednesday    $1526440.000

3. The top 10 selling products and quantity:
    WORLD WAR 2 GLIDERS ASSTD DESIGNS     47886
    JUMBO BAG RED RETROSPOT               40777
    ASSORTED COLOUR BIRD ORNAMENT         32580
    WHITE HANGING HEART T-LIGHT HOLDER    32079
    POPCORN HOLDER                        28550
    PACK OF 12 LONDON TISSUES             24297
    BROCADE RING PURSE                    22672
    PACK OF 72 RETROSPOT CAKE CASES       22182
    VICTORIAN GLASS HANGING T-LIGHT       21427
    PACK OF 60 PINK PAISLEY CAKE CASES    19882


4. The top customer - United Kingdom


6. Outliers were found in Unit Price: which will affect the Sales column
   Unit prices were extremely higher for some products.
   Negative values in the Sales.

7. There were large Quanties shown which is abnormal.

Recommendation

1. Increase the stocks in the top selling products

2. Create special offers for the top customers in United Kingdom to encourage␣
 ↪purchasing.

3. Open on Saturdays.

4. On the busy months have promotions to encourage more purchases.
```

```
5. Take a closer look at the abnormal figures because this can a deterring␣
↪customers from purchasing.
```