2024 09 30 Analysis_Online Retail Data_eng

October 4, 2024

0.0.1 Overview of the Online Retail Dataset

The Online Retail dataset is a transaction dataset for a British online retailer. It contains all transactions between 01.12.2010 and 09.12.2011. The dataset includes the following information:

InvoiceNo Invoice number. A unique six-digit number assigned to each transaction. If the number starts with 'C', it indicates a cancellation.

StockCode Product code. A unique five-digit number for each product.

Description Product description.

Quantity Number of products sold per transaction.

InvoiceDate Date and time of the transaction.

UnitPrice Unit price of the product.

CustomerID Customer number. A unique five-digit number for each customer.

Country The country where the customer is located.

0.0.2 Possible Analyses

With this dataset, we can conduct various analyses, such as:

Sales Analysis Revenue over time, top products, sales trends.

Customer Analysis Buying behavior, customer retention, geographic distribution of customers.

Product Analysis Popular products, cancellations, stock levels.

0.0.3 Steps for an Exploratory Data Analysis (EDA)

Data Import and Initial Inspection Load the dataset into a pandas DataFrame. Overview of the data structure (data types, number of rows/columns). Display the first and last rows.

Data Cleaning Check for missing values and decide how to handle them. Remove duplicates. Convert data types (e.g., InvoiceDate to datetime). Handle negative or unusual values (e.g., negative Quantity).

Descriptive Statistics Calculate metrics such as mean, median, standard deviation. Examine the distribution of numerical variables.

Data Visualization Create charts like histograms, box plots, scatter plots. Visualize trends over time.

Feature Engineering Create new variables, e.g., total price (TotalPrice = Quantity * UnitPrice). Extract time information from InvoiceDate (month, weekday, hour).

Gaining Insights Summarize the key findings from the analysis. Identify patterns or anomalies.

1. Importing the necessary libraries

[84]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Explanation: We import the necessary libraries for data analysis and visualization. %matplotlib inline ensures that the plots are displayed directly in the notebook.

2. Loading the Dataset

[80]:
```python
# Assuming the file is named 'OnlineRetail.xlsx' and is located in the current␣
↪directory
df = pd.read_excel('OnlineRetail.xlsx')
```

3. Initial Data Inspection

[86]:
```python
# Overview of the data
df.head()
```

[86]:
```
  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
```

[88]:
```python
# Information about the DataFrame
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
```

[90]:
```python
# Statistical metrics
df.describe()
```

[90]:
```
            Quantity                    InvoiceDate      UnitPrice  \
count  541909.000000                         541909  541909.000000
mean        9.552250  2011-07-04 13:34:57.156386048       4.611114
min    -80995.000000            2010-12-01 08:26:00  -11062.060000
25%         1.000000            2011-03-28 11:34:00       1.250000
50%         3.000000            2011-07-19 17:17:00       2.080000
75%        10.000000            2011-10-19 11:27:00       4.130000
max     80995.000000            2011-12-09 12:50:00   38970.000000
std       218.081158                            NaN      96.759853

          CustomerID
count  406829.000000
mean    15287.690570
min     12346.000000
25%     13953.000000
50%     15152.000000
75%     16791.000000
max     18287.000000
std      1713.600303
```

Explanation: head() displays the first five rows. info() provides information about the columns and their data types. describe() returns statistical metrics for numerical columns.

4. Checking for Missing Values

[92]:
```python
# Number of missing values per column
df.isnull().sum()
```

[92]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

Explanation: Using isnull().sum(), we get the number of missing values in each column.

5. Handling Missing Values

[94]:
```python
# Since 'CustomerID' has missing values, we could remove those rows (not␣
↪recommended though, you might end up missing a lot of important data)
df = df.dropna(subset=['CustomerID'])
```

Explanation: We remove rows where 'CustomerID' is missing, as this is important for customer analyses.

6. Converting Data Types

[100]:
```python
# Converting 'InvoiceDate' to datetime
df.loc[:, 'InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
```

Explanation: We convert the 'InvoiceDate' column to the datetime format for time series analysis.

7. Handling Negative Values

[102]:
```python
# Checking for negative quantities
df[df['Quantity'] < 0]
```

[102]:
```
       InvoiceNo StockCode                       Description  Quantity  \
141      C536379         D                          Discount        -1
154      C536383    35004C   SET OF 3 COLOURED  FLYING DUCKS        -1
235      C536391     22556    PLASTERS IN TIN CIRCUS PARADE        -12
236      C536391     21984  PACK OF 12 PINK PAISLEY TISSUES        -24
237      C536391     21983  PACK OF 12 BLUE PAISLEY TISSUES        -24
…            …         …                               …         …
540449   C581490     23144   ZINC T-LIGHT HOLDER STARS SMALL       -11
541541   C581499         M                            Manual        -1
541715   C581568     21258        VICTORIAN SEWING BOX LARGE        -5
```

```
541716    C581569      84978  HANGING HEART JAR T-LIGHT HOLDER        -1
541717    C581569      20979     36 PENCILS TUBE RED RETROSPOT        -5

               InvoiceDate  UnitPrice  CustomerID         Country
141     2010-12-01 09:41:00      27.50     14527.0  United Kingdom
154     2010-12-01 09:49:00       4.65     15311.0  United Kingdom
235     2010-12-01 10:24:00       1.65     17548.0  United Kingdom
236     2010-12-01 10:24:00       0.29     17548.0  United Kingdom
237     2010-12-01 10:24:00       0.29     17548.0  United Kingdom
…                       …         …          …             …
540449  2011-12-09 09:57:00       0.83     14397.0  United Kingdom
541541  2011-12-09 10:28:00     224.69     15498.0  United Kingdom
541715  2011-12-09 11:57:00      10.95     15311.0  United Kingdom
541716  2011-12-09 11:58:00       1.25     17315.0  United Kingdom
541717  2011-12-09 11:58:00       1.25     17315.0  United Kingdom

[8905 rows x 8 columns]
```

Explanation: Negative values in 'Quantity' may indicate cancellations. We can consider them separately or exclude them from specific analyses.

8. Calculating Total Price

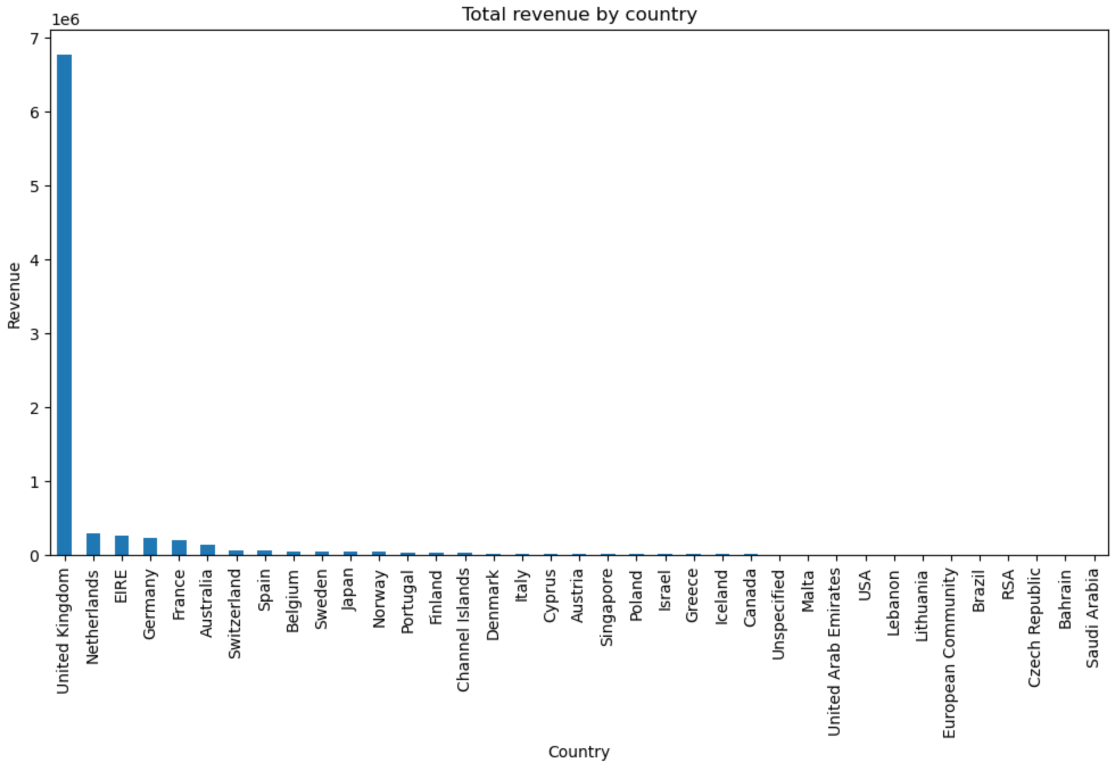
[106]:
```python
# Create a new column 'TotalPrice'
df.loc[:, 'TotalPrice'] = df['Quantity'] * df['UnitPrice']
```

Explanation: We create a new column that indicates the total price for each row.

9. Data Visualization

9.1. Revenue by Country

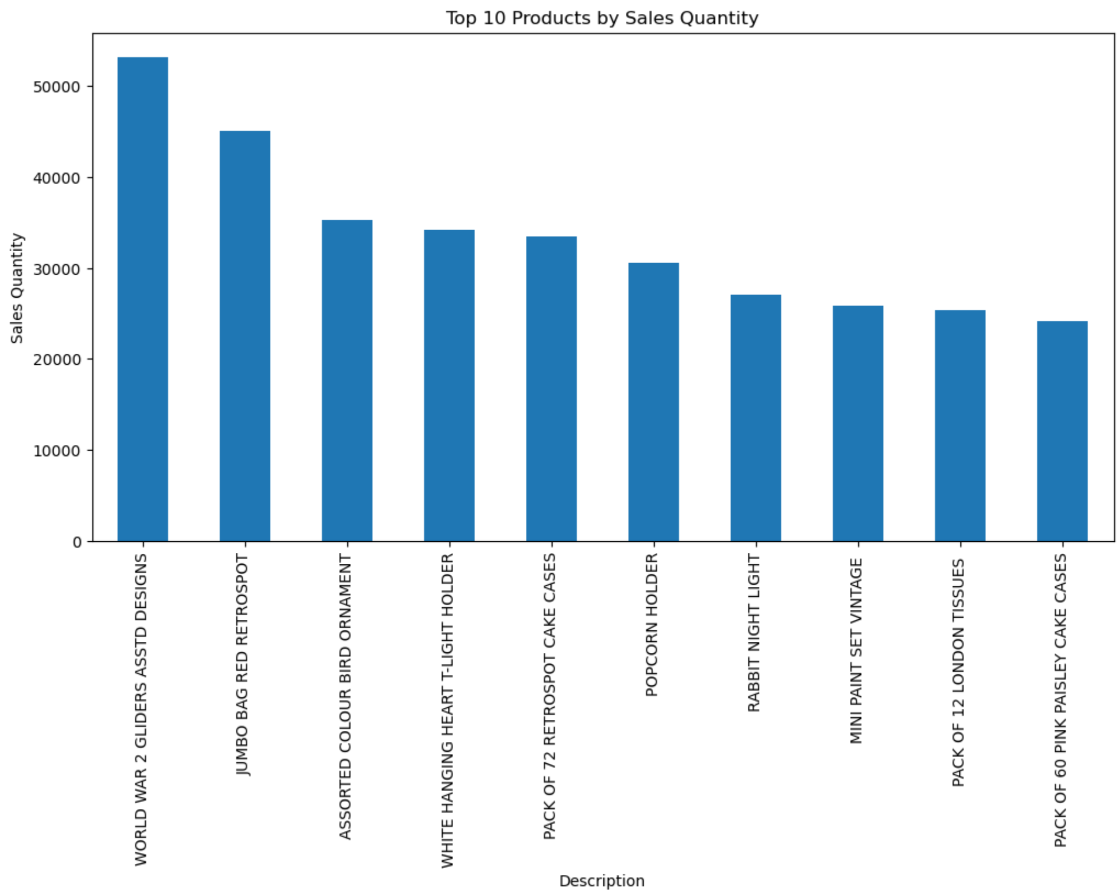
[108]:
```python
# Total revenue by country
sales_country = df.groupby('Country')['TotalPrice'].sum().
↪sort_values(ascending=False)

# Visualization
plt.figure(figsize=(12,6))
sales_country.plot(kind='bar')
plt.title('Total revenue by country')
plt.ylabel('Revenue')
plt.show()
```



Explanation: We group the revenue by country and visualize it in a bar chart.

9.2. Top 10 Products by Sales Quantity

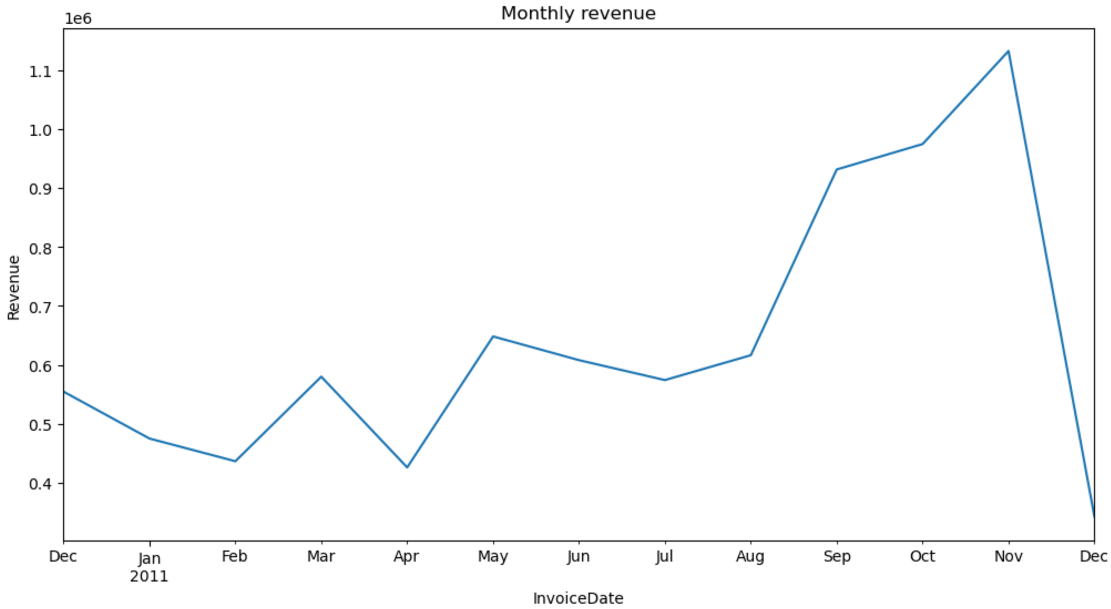
[110]:
```
# Top 10 products
top_products = df.groupby('Description')['Quantity'].sum().
 ↪sort_values(ascending=False).head(10)

# Visualization
plt.figure(figsize=(12,6))
top_products.plot(kind='bar')
plt.title('Top 10 Products by Sales Quantity')
plt.ylabel('Sales Quantity')
plt.show()
```



Explanation: We identify the top-selling products and visualize them.

9.3. Revenue Over Time

[122]:
```python
# Revenue per month
df.set_index('InvoiceDate', inplace=True)
monthly_sales = df['TotalPrice'].resample('M').sum()

# Visualization
plt.figure(figsize=(12,6))
monthly_sales.plot()
plt.title('Monthly revenue')
plt.ylabel('Revenue')
plt.show()
```

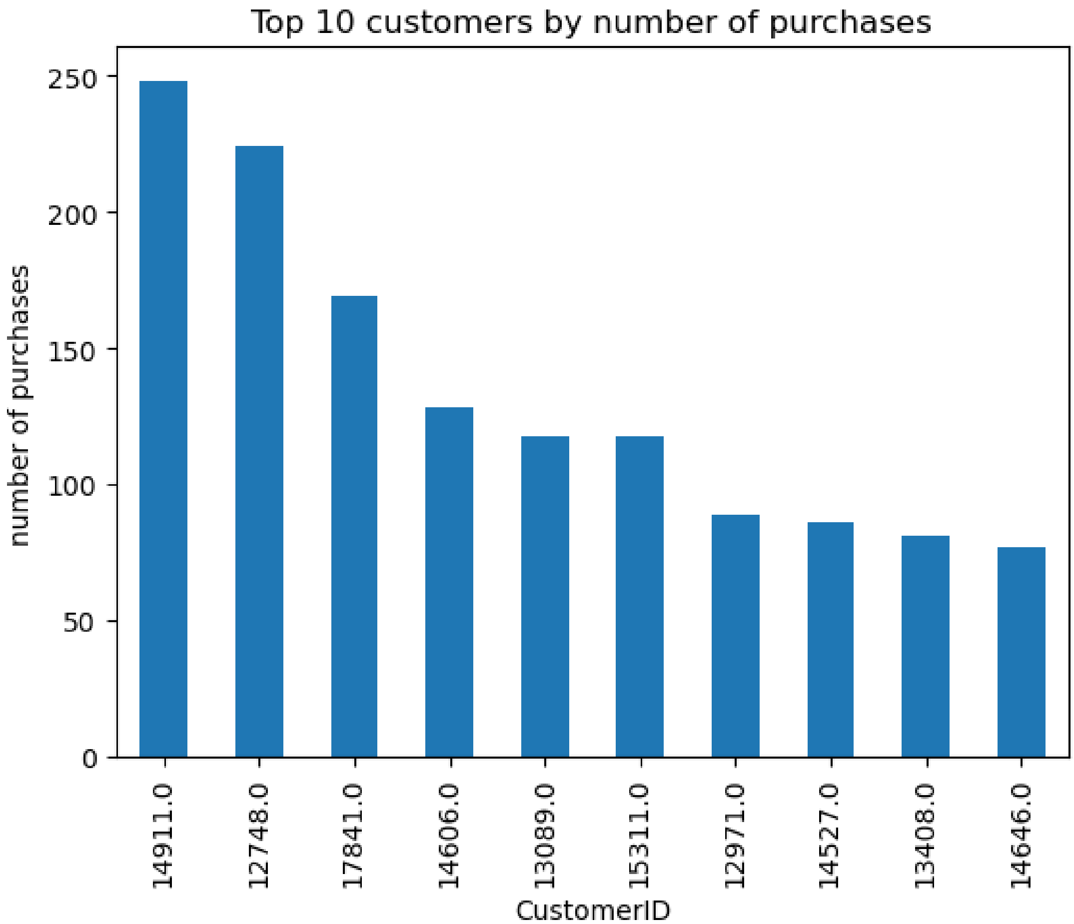


Explanation: We resample the data on a monthly basis to observe the revenue trend over time.

10. Customer Analysis

10.1. Number of Purchases per Customer

[124]:
```python
# Purchases per customer
purchases_per_customer = df.groupby('CustomerID')['InvoiceNo'].nunique().
 ↪sort_values(ascending=False)

# Visualization of the top 10 customers
purchases_per_customer.head(10).plot(kind='bar')
plt.title('Top 10 customers by number of purchases')
plt.ylabel('number of purchases')
plt.show()
```



Explanation: We analyze which customers made the most purchases.

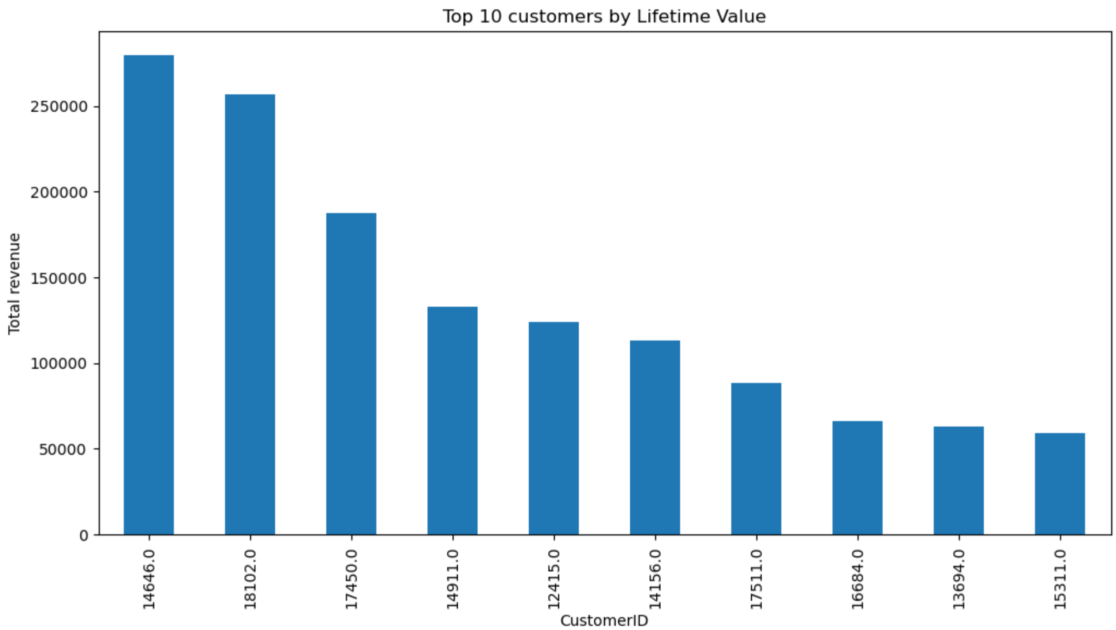
11. Summary of Findings After completing the EDA, you should summarize the key findings. For example, we might find:

Main Markets: Most sales occur in the United Kingdom. ##### Sales Trends: There are seasonal peaks in November and December. ##### Top Products:
Certain products sell significantly better than others. ##### Customer Behavior: Some customers make frequent purchases, indicating customer loyalty.

12. Customer Lifetime Value (CLV)

[126]:
```
# Calculating revenue per customer
clv = df.groupby('CustomerID')['TotalPrice'].sum().sort_values(ascending=False)

# Visualization of the top 10 customers with the highest customer value
clv.head(10).plot(kind='bar', figsize=(12,6))
plt.title('Top 10 customers by Lifetime Value')
```

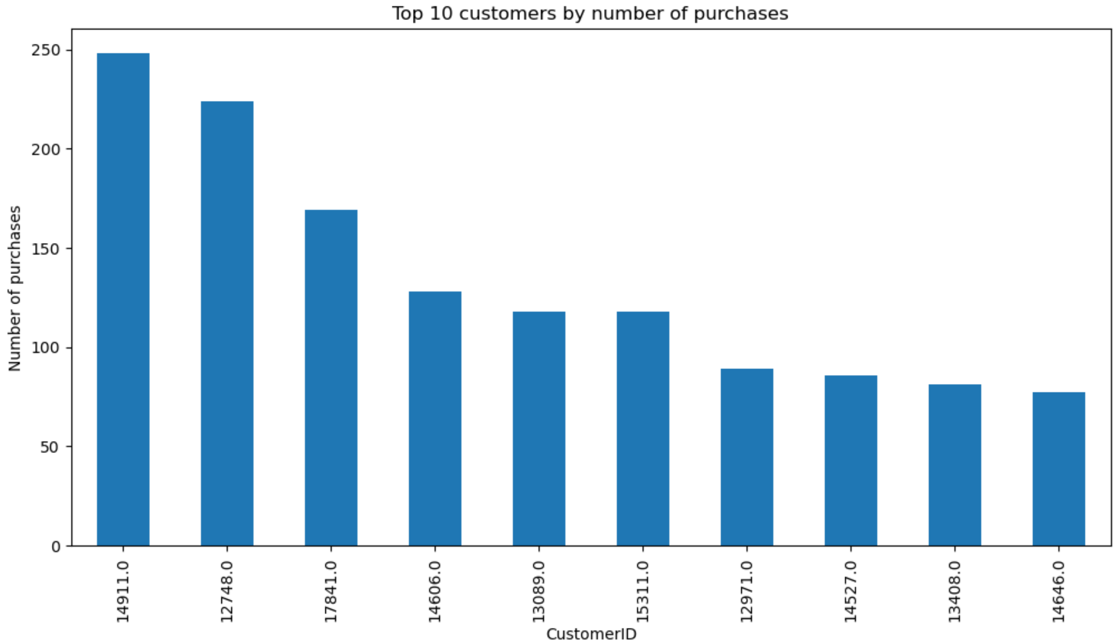
```
plt.ylabel('Total revenue')
plt.show()
```



13. Customer Retention and Repeat Purchases

[132]:

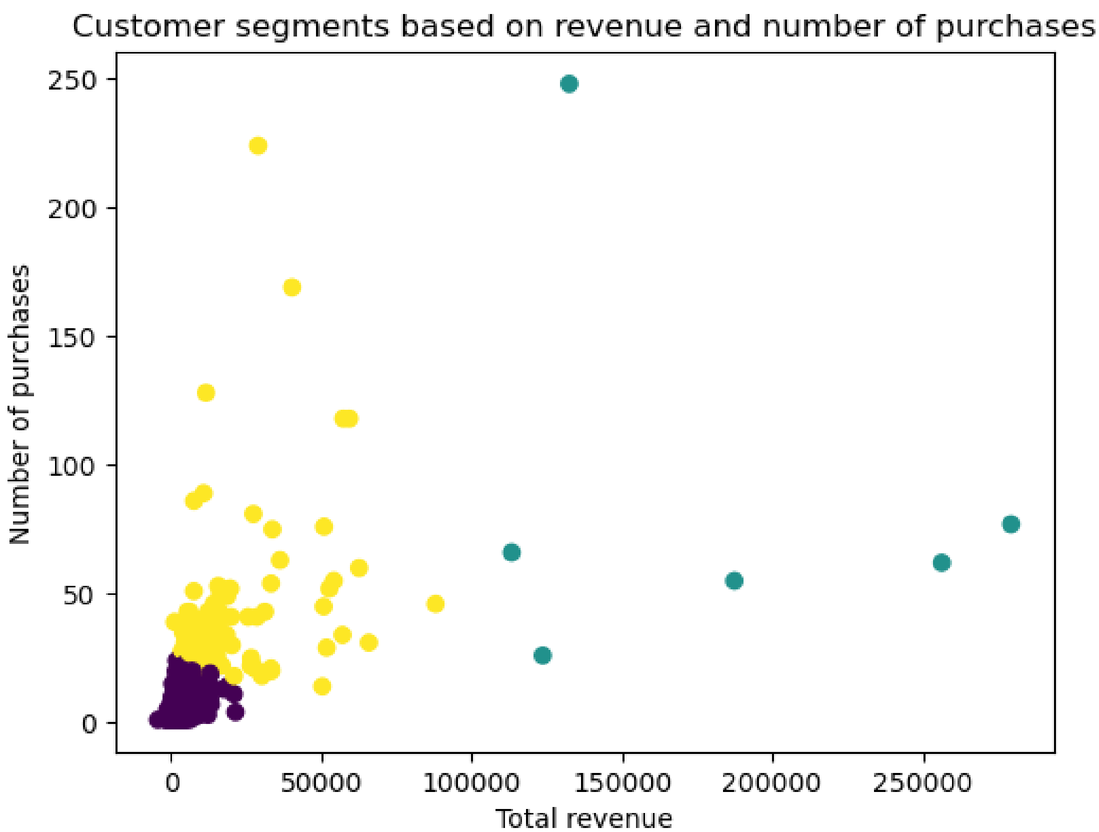
```
# Check if 'InvoiceDate' is the index
if df.index.name == 'InvoiceDate':
    df.reset_index(inplace=True)  # Reset of the Indexes, so InvoiceDate is␣
↪considered as a column again

# Number of purchases per customer
repeat_customers = df.groupby('CustomerID')['InvoiceNo'].nunique()

# Calculate average repurchase time per customer
df.loc[:, 'days_between_purchases'] = df.groupby('CustomerID')['InvoiceDate'].
↪diff().dt.days
avg_days_between_purchases = df.groupby('CustomerID')['days_between_purchases'].
↪mean()

# Visualization of the top 10 customers with the most purchases
repeat_customers.sort_values(ascending=False).head(10).plot(kind='bar',␣
↪figsize=(12,6))
plt.title('Top 10 customers by number of purchases')
plt.ylabel('Number of purchases')
plt.show()
```



14. Customer Segmentation (Clustering)

[130]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Data for cluster analysis (e.g., total revenue and number of purchases per␣
↪customer)
customer_data = df.groupby('CustomerID').agg({
    'TotalPrice': 'sum',
    'InvoiceNo': 'nunique'
}).rename(columns={'InvoiceNo': 'NumberOfPurchases'})

# Normalize data
scaler = StandardScaler()
customer_data_scaled = scaler.fit_transform(customer_data)

# K-Means Clustering
import os
os.environ["LOKY_MAX_CPU_COUNT"] = "2"  # Example: Set the number of logical␣
↪cores to 2

kmeans = KMeans(n_clusters=3, random_state=0, n_init=10)
customer_data['Cluster'] = kmeans.fit_predict(customer_data_scaled)

# Visualization of the clusters
```

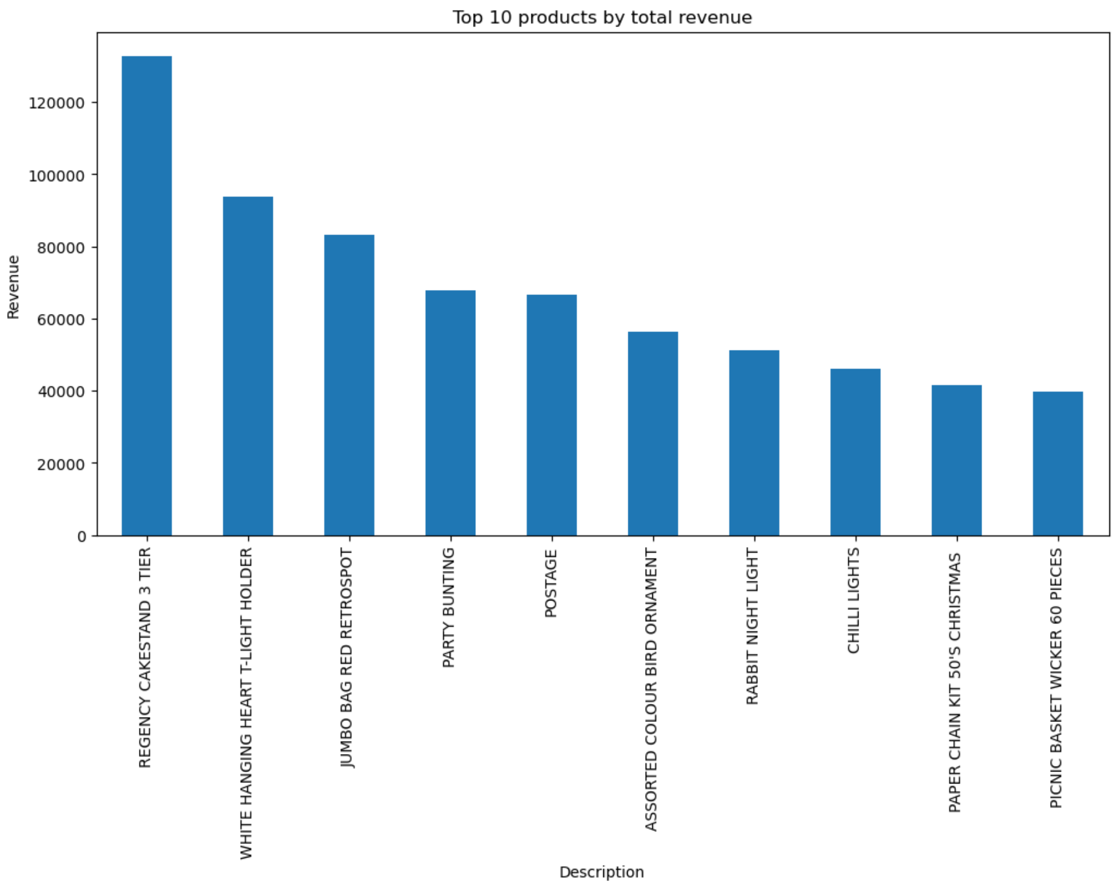
```
plt.scatter(customer_data['TotalPrice'], customer_data['NumberOfPurchases'],␣
 ↪c=customer_data['Cluster'], cmap='viridis')
plt.title('Customer segments based on revenue and number of purchases')
plt.xlabel('Total revenue')
plt.ylabel('Number of purchases')
plt.show()
```

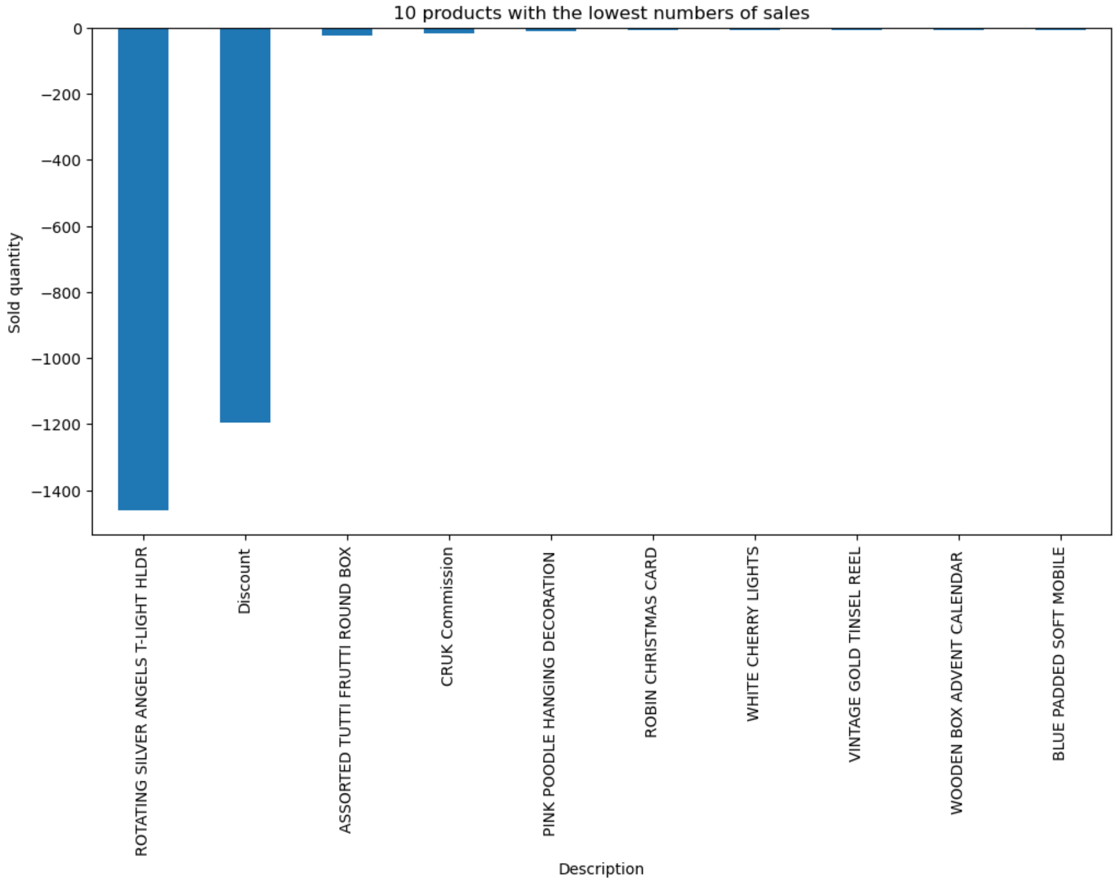


15. Product and Assortment Optimization

[134]:
```
# Products with the highest revenue
product_sales = df.groupby('Description')['TotalPrice'].sum().
 ↪sort_values(ascending=False)

# Products with the lowest sales
low_sales_products = df.groupby('Description')['Quantity'].sum().sort_values()

# Visualization of the top 10 highest-revenue products
product_sales.head(10).plot(kind='bar', figsize=(12,6))
plt.title('Top 10 products by total revenue')
plt.ylabel('Revenue')
```

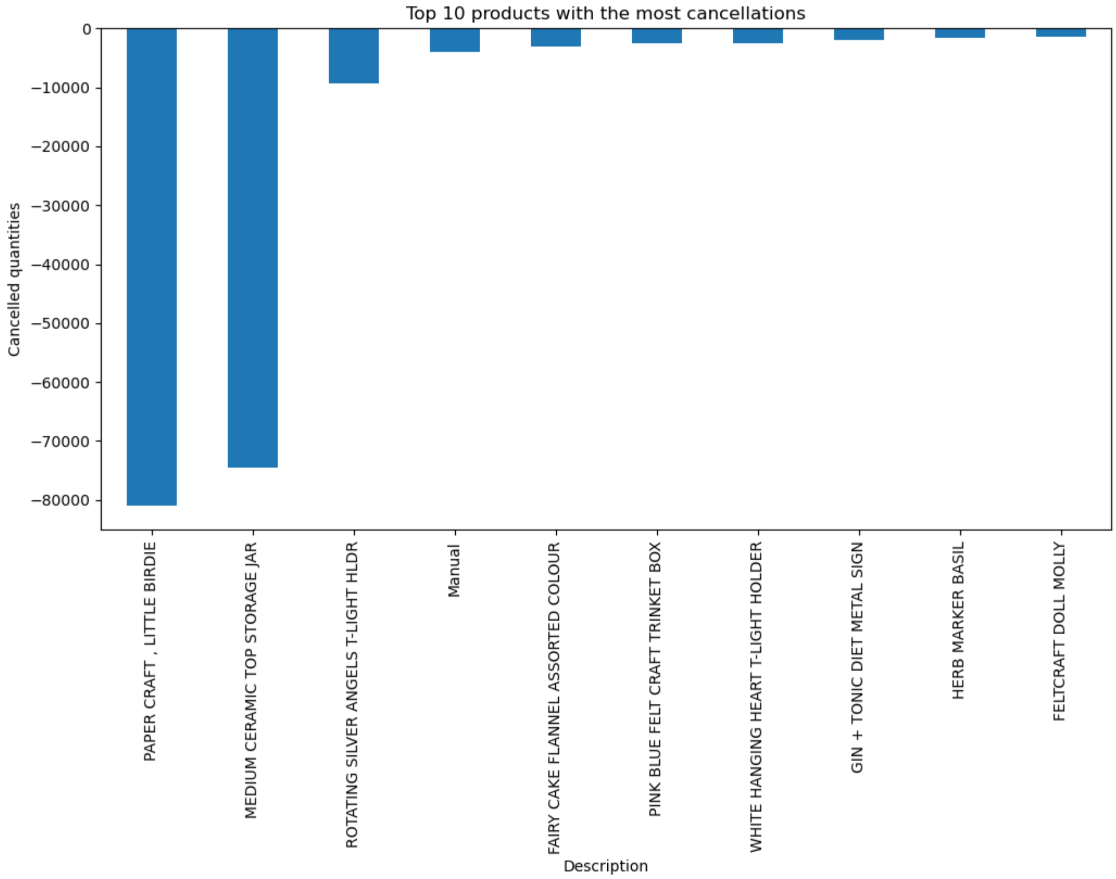
```
plt.show()

# Visualization of the 10 lowest-selling products
low_sales_products.head(10).plot(kind='bar', figsize=(12,6))
plt.title('10 products with the lowest numbers of sales')
plt.ylabel('Sold quantity')
plt.show()
```





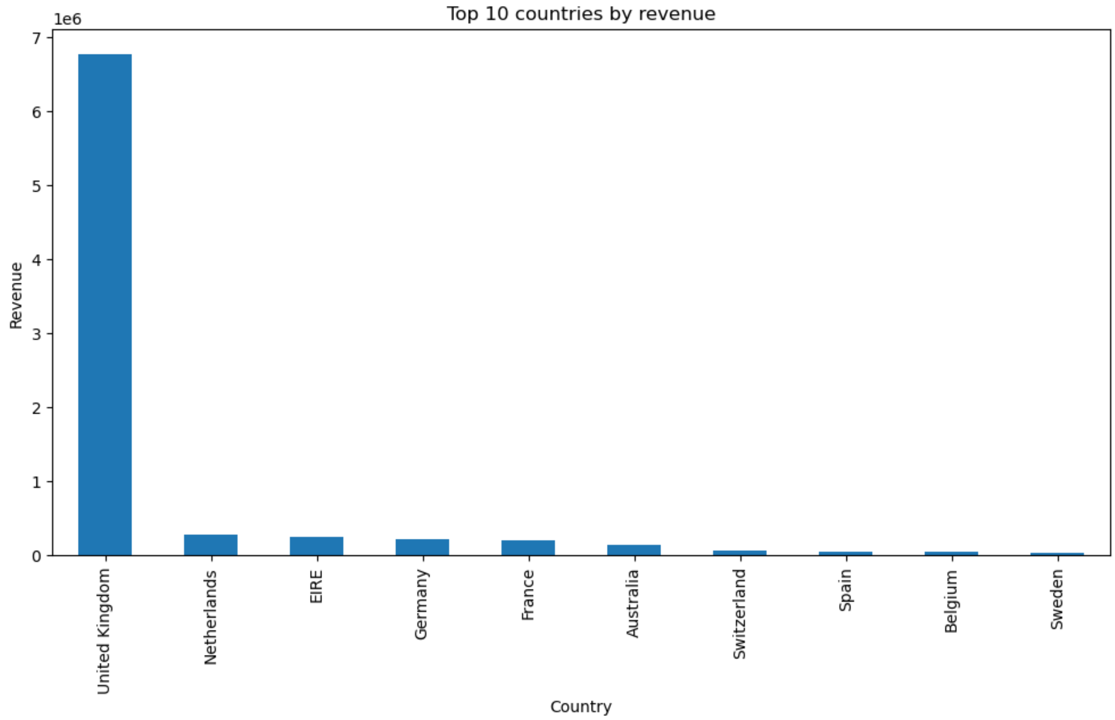
16. Cancellation Analysis

[136]:
```python
# Analysis of cancellations (negative quantities)
cancellations = df[df['Quantity'] < 0]

# Top products with the most cancellations
top_cancellations = cancellations.groupby('Description')['Quantity'].sum().
↪sort_values()

# Visualization of the top 10 products with the most cancellations
top_cancellations.head(10).plot(kind='bar', figsize=(12,6))
plt.title('Top 10 products with the most cancellations')
plt.ylabel('Cancelled quantities')
plt.show()
```



17. International Expansion

[138]:
```
# Revenue by country
sales_country = df.groupby('Country')['TotalPrice'].sum().
 ↪sort_values(ascending=False)

# Visualization of the top 10 countries by revenue
sales_country.head(10).plot(kind='bar', figsize=(12,6))
plt.title('Top 10 countries by revenue')
plt.ylabel('Revenue')
plt.show()
```



Top 10 countries by revenue
1e6
7
6
5
4
3
2
1
0
Revenue
United Kingdom
Netherlands
EIRE
Germany
France
Australia
Switzerland
Spain
Belgium
Sweden
Country