In [2]:
```
import pandas as pd
df = pd.read_csv('C:/Users/Dell/OneDrive/Desktop/python files/online retail.csv', e
```

In [3]:
```
df.describe()
```

Out[3]:

	Quantity	Price	Customer ID
count	525461.000000	525461.000000	417534.000000
mean	10.337667	4.688834	15360.645478
std	107.424110	146.126914	1680.811316
min	-9600.000000	-53594.360000	12346.000000
25%	1.000000	1.250000	13983.000000
50%	3.000000	2.100000	15311.000000
75%	10.000000	4.210000	16799.000000
max	19152.000000	25111.090000	18287.000000

In [4]:
```
print(df.isnull().sum())
```

```
Invoice             0
StockCode           0
Description      2928
Quantity            0
InvoiceDate         0
Price               0
Customer ID    107927
Country             0
dtype: int64
```

In [5]:
```
df['Description'] = df['Description'].fillna('return')  #replced null places in des
```

In [ ]:
```
duplicates = df[df.duplicated()]
print(f"Duplicate rows: {len(duplicates)}")
```

In [ ]:
```
ITEMS WHERE THE QUANTITY IS NEGATIVE THEY ARE THE ITEMS THAT ARE RETURNED
```

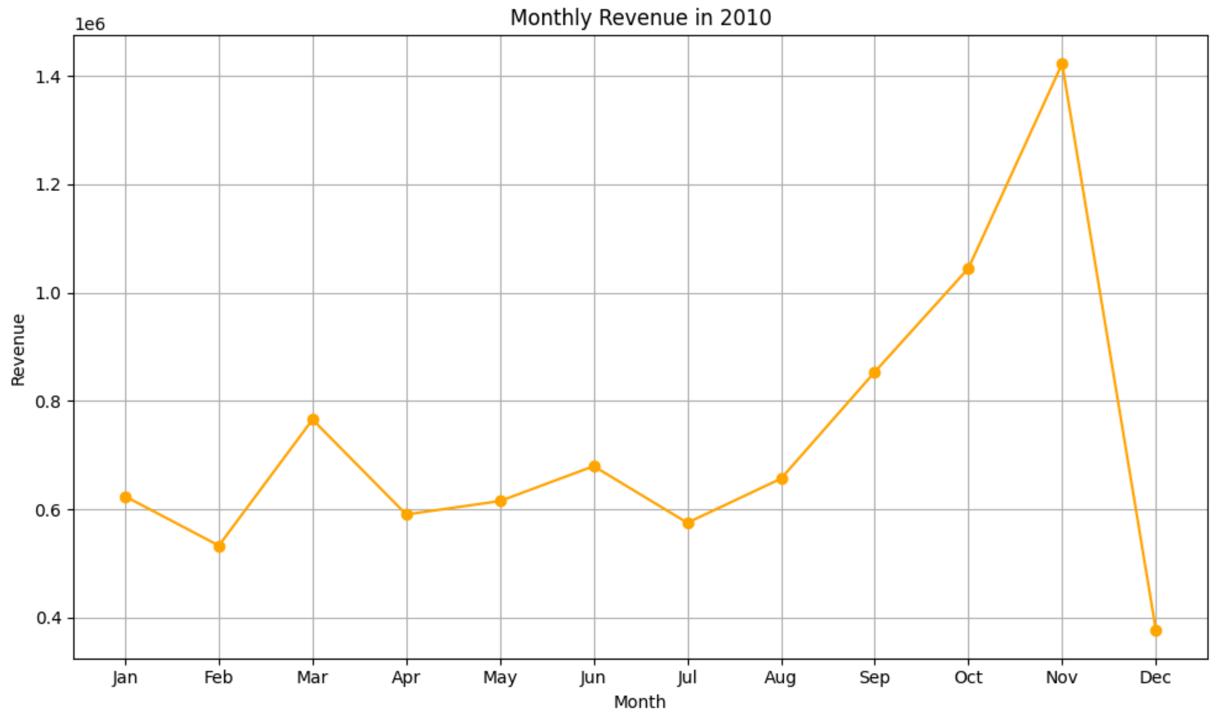
In [6]:
```
df = df[~((df['Quantity'] < 0) & (df['Price'] == 0))]
```

In [7]:
```
print(df[df['Quantity'] < 0].head())  # Returns
print(df[df['Price'] < 0].head())     # May be data errors
print(df[df['Price'] == 0].head())
```

```
     Invoice StockCode                     Description  Quantity  \
178  C489449     22087       PAPER BUNTING WHITE LACE       -12
179  C489449    85206A   CREAM FELT EASTER EGG BASKET        -6
180  C489449     21895  POTTING SHED SOW 'N' GROW SET        -4
181  C489449     21896              POTTING SHED TWINE        -6
182  C489449     22083     PAPER CHAIN KIT RETRO SPOT       -12

         InvoiceDate  Price  Customer ID    Country
178  12/1/2009 10:33   2.95      16321.0  Australia
179  12/1/2009 10:33   1.65      16321.0  Australia
180  12/1/2009 10:33   4.25      16321.0  Australia
181  12/1/2009 10:33   2.10      16321.0  Australia
182  12/1/2009 10:33   2.95      16321.0  Australia
        Invoice StockCode      Description  Quantity       InvoiceDate  \
179403  A506401         B  Adjust bad debt         1   4/29/2010 13:36
276274  A516228         B  Adjust bad debt         1   7/19/2010 11:24
403472  A528059         B  Adjust bad debt         1  10/20/2010 12:04

           Price  Customer ID         Country
179403 -53594.36          NaN  United Kingdom
276274 -44031.79          NaN  United Kingdom
403472 -38925.87          NaN  United Kingdom
     Invoice StockCode      Description  Quantity      InvoiceDate  Price  \
3161  489659     21350              NaN       230  12/1/2009 17:39    0.0
3731  489781     84292              NaN        17  12/2/2009 11:45    0.0
4674  489825     22076  6 RIBBONS EMPIRE        12  12/2/2009 13:34    0.0
5904  489861       DOT    DOTCOM POSTAGE         1  12/2/2009 14:50    0.0
6378  489882    35751C              NaN        12  12/2/2009 16:22    0.0

      Customer ID         Country
3161          NaN  United Kingdom
3731          NaN  United Kingdom
4674      16126.0  United Kingdom
5904          NaN  United Kingdom
6378          NaN  United Kingdom
```

In [8]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Prepare data
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])  # ensure datetime
df['TotalPrice'] = df['Quantity'] * df['Price']        # calculate revenue
df['Year'] = df['InvoiceDate'].dt.year                 # extract year

# Step 2: Filter for 2010 only
df_2010 = df[df['Year'] == 2010].copy()

# Step 3: Extract month info
df_2010['MonthNum'] = df_2010['InvoiceDate'].dt.month
df_2010['MonthName'] = df_2010['InvoiceDate'].dt.strftime('%b')

# Step 4: Group by month and sort
monthly_2010 = df_2010.groupby(['MonthNum', 'MonthName'])['TotalPrice'].sum().reset
monthly_2010 = monthly_2010.sort_values('MonthNum')

# Step 5: Plot
```

```
plt.figure(figsize=(10,6))
plt.plot(monthly_2010['MonthName'], monthly_2010['TotalPrice'], marker='o', color='
plt.title('Monthly Revenue in 2010')
plt.xlabel('Month')
plt.ylabel('Revenue')
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [9]:
```
# Step 1: Compute Total Revenue per row
df['TotalPrice'] = df['Quantity'] * df['Price']

# Step 2: Group by Product
product_perf = df.groupby(['StockCode', 'Description']).agg(
    TotalQuantity=('Quantity', 'sum'),
    TotalRevenue=('TotalPrice', 'sum')
).reset_index()

# Step 3: Sort by performance
top_products_qty = product_perf.sort_values(by='TotalQuantity', ascending=False).he
top_products_revenue = product_perf.sort_values(by='TotalRevenue', ascending=False)
worst_products_qty = product_perf.sort_values(by='TotalQuantity', ascending=True).h
worst_products_revenue = product_perf.sort_values(by='TotalRevenue', ascending=True

# Step 4: Display results
print("🔥 Top 10 Products by Quantity Sold:")
print(top_products_qty)

print("\n💰 Top 10 Products by Revenue:")
print(top_products_revenue)

print("\n📉 Bottom 10 Products by Quantity:")
print(worst_products_qty)
print("\n🪫 Bottom 10 Products by Revenue:")
print(worst_products_revenue)
```

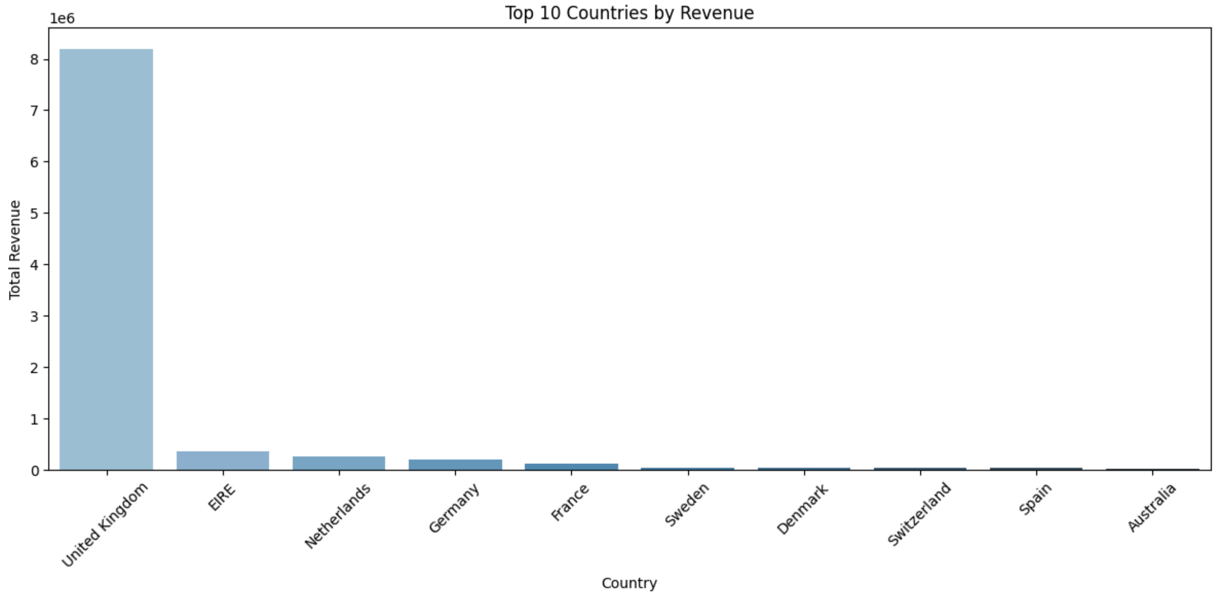
```
🔥 Top 10 Products by Quantity Sold:
     StockCode                         Description  TotalQuantity  \
4167    85123A  WHITE HANGING HEART T-LIGHT HOLDER          57428   
3364     84077   WORLD WAR 2 GLIDERS ASSTD DESIGNS          54698   
118      17003                 BROCADE RING PURSE           47647   
668      21212     PACK OF 72 RETRO SPOT CAKE CASES          46106   
3823     84879       ASSORTED COLOUR BIRD ORNAMENT          44925   
3983     84991         60 TEATIME FAIRY CAKE CASES          36326   
1382     21977  PACK OF 60 PINK PAISLEY CAKE CASES          31822   
4136    85099B             JUMBO BAG RED RETROSPOT          30308   
1603     22197                SMALL POPCORN HOLDER          29500   
689      21232      STRAWBERRY CERAMIC TRINKET BOX          26563   

      TotalRevenue  
4167     155825.52  
3364      11310.29  
118        8879.82  
668       23759.26  
3823      72454.12  
3983      18128.25  
1382      16184.21  
4136      54332.97  
1603      26791.95  
689       33834.70  

💰 Top 10 Products by Revenue:
     StockCode                          Description  TotalQuantity  \
1884     22423             REGENCY CAKESTAND 3 TIER          13093   
4167    85123A   WHITE HANGING HEART T-LIGHT HOLDER          57428   
4774       DOT                       DOTCOM POSTAGE            731   
3823     84879        ASSORTED COLOUR BIRD ORNAMENT          44925   
1477     22086      PAPER CHAIN KIT 50'S CHRISTMAS           17083   
4136    85099B              JUMBO BAG RED RETROSPOT          30308   
2927     47566                        PARTY BUNTING          10088   
3417     84347  ROTATING SILVER ANGELS T-LIGHT HLDR          13675   
4777      POST                              POSTAGE           2154   
4140    85099F               JUMBO BAG STRAWBERRY          19818   

      TotalRevenue  
1884     163051.46  
4167     155825.52  
4774     116401.99  
3823      72454.12  
1477      57870.20  
4136      54332.97  
2927      49645.52  
3417      47672.49  
4777      46092.36  
4140      35854.59  

📉 Bottom 10 Products by Quantity:
         StockCode                         Description  TotalQuantity  \
4755             D                            Discount          -1678   
3267        79323S                 SILVER CHERRY LIGHTS            -96   
3269        79323W                  WHITE CHERRY LIGHTS            -86   
4778             S                             SAMPLES            -39   
```

```
403          20879  TREE OF NOAH FESTIVE SCENTED CANDLE            -27
4753  BANK CHARGES                         Bank Charges            -27
1140         21701  SET 6 MINI SUSHI SET FRIDGE MAGNETS            -12
2730        35976B   WHITE SCANDINAVIAN HEART CHRISTMAS            -11
1399         22003      VINTAGE BLUE VACUUM FLASK 0.5L             -10
3250         79301                  FEATHER HEART LIGHTS            -8

      TotalRevenue
4755      -7788.32
3267       -611.28
3269       -427.24
4778      -3016.41
403        -126.00
4753     -26318.03
1140        -20.28
2730        -13.75
1399        -67.50
3250        -20.00

🪫 Bottom 10 Products by Revenue:
         StockCode                          Description  TotalQuantity  \
4751             B                      Adjust bad debt              3
4750     AMAZONFEE                           AMAZON FEE             -5
4753  BANK CHARGES                         Bank Charges            -27
4775             M                               Manual           1443
4755             D                             Discount          -1678
4778             S                              SAMPLES            -39
4752  BANK CHARGES                         Bank Charges             -2
4746        ADJUST  Adjustment by Peter on 24/05/2010 1             -3
3267        79323S                  SILVER CHERRY LIGHTS            -96
3269        79323W                   WHITE CHERRY LIGHTS            -86

      TotalRevenue
4751    -136552.02
4750     -39243.08
4753     -26318.03
4775     -14122.13
4755      -7788.32
4778      -3016.41
4752      -2068.96
4746       -731.05
3267       -611.28
3269       -427.24
```

In [10]:
```
# Create a new column for total sales
df['TotalSales'] = df['Quantity'] * df['Price']

# Group by Customer ID and sum the sales
customer_sales = df.groupby('Customer ID')['TotalSales'].sum().sort_values(ascendin

# Display the top customer
top_customer = customer_sales.head(1)
print(top_customer)
```

```
Customer ID
18102.0    341776.73
Name: TotalSales, dtype: float64
```

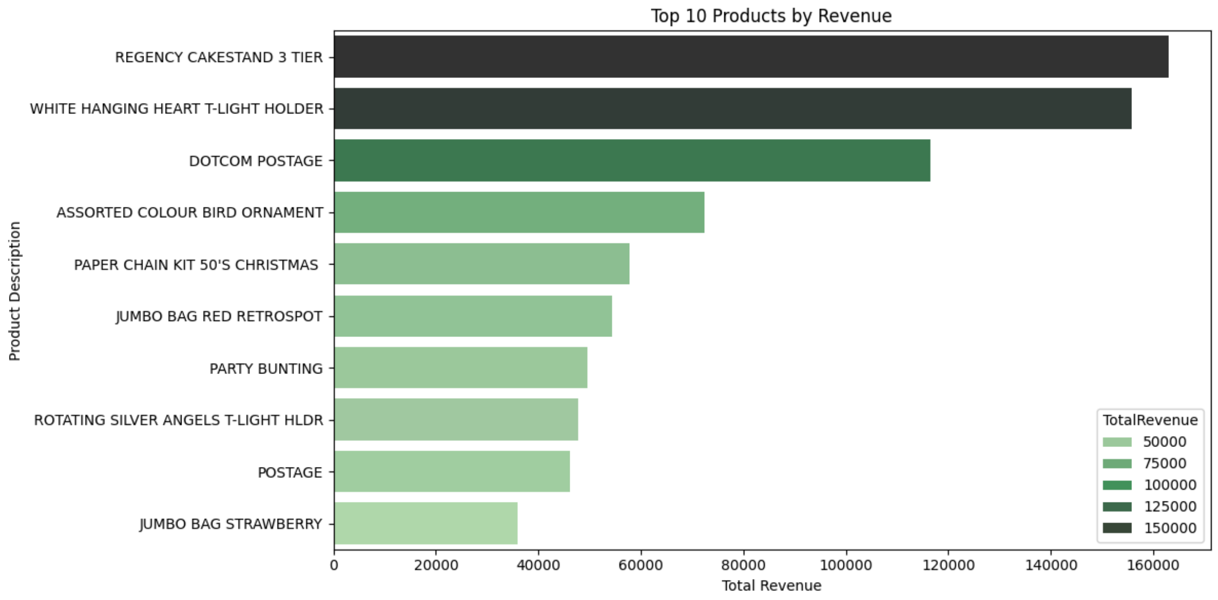
In [12]:
```
import seaborn as sns
import matplotlib.pyplot as plt
geo_sales = df.groupby('Country').agg({
    'Quantity': 'sum',
    'TotalSales': 'sum'
}).sort_values(by='TotalSales', ascending=False)
top_countries = geo_sales.head(10).reset_index()

plt.figure(figsize=(12,6))
sns.barplot(data=top_countries, x='Country', y='TotalSales', hue='Country', palette
plt.title('Top 10 Countries by Revenue')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [13]:
```
# First, calculate total revenue per order (Invoice)
df['TotalPrice'] = df['Quantity'] * df['Price']

# Group by Invoice to get total revenue per order
order_totals = df.groupby('Invoice')['TotalPrice'].sum()

# Now calculate AOV
average_order_value = order_totals.mean()

print(f'Average Order Value (AOV): £{average_order_value:.2f}')
```

```
Average Order Value (AOV): £357.35
```

In [15]:
```
# Step 1: Calculate total revenue per row
df['TotalPrice'] = df['Quantity'] * df['Price']

# Step 2: Group by product
product_perf = df.groupby(['StockCode', 'Description']).agg(
    TotalQuantity=('Quantity', 'sum'),
    TotalRevenue=('TotalPrice', 'sum')
).reset_index()

# Step 3: Top 10 products by quantity sold
top_by_quantity = product_perf.sort_values(by='TotalQuantity', ascending=False).hea

# Step 4: Top 10 products by revenue
top_by_revenue = product_perf.sort_values(by='TotalRevenue', ascending=False).head(

# Step 5: Bottom 10 products by quantity
bottom_by_quantity = product_perf.sort_values(by='TotalQuantity').head(10)

# Step 6: Bottom 10 products by revenue
bottom_by_revenue = product_perf.sort_values(by='TotalRevenue').head(10)


plt.figure(figsize=(12, 6))
sns.barplot(data=top_by_revenue, x='TotalRevenue',  hue='TotalRevenue',y='Descripti
plt.title('Top 10 Products by Revenue')
plt.xlabel('Total Revenue')
plt.ylabel('Product Description')
plt.tight_layout()
plt.show()
```



```
In [16]: returns = df[df['Quantity'] < 0]
         returns_summary = returns.groupby(['StockCode', 'Description'])['Quantity'].sum().r
```

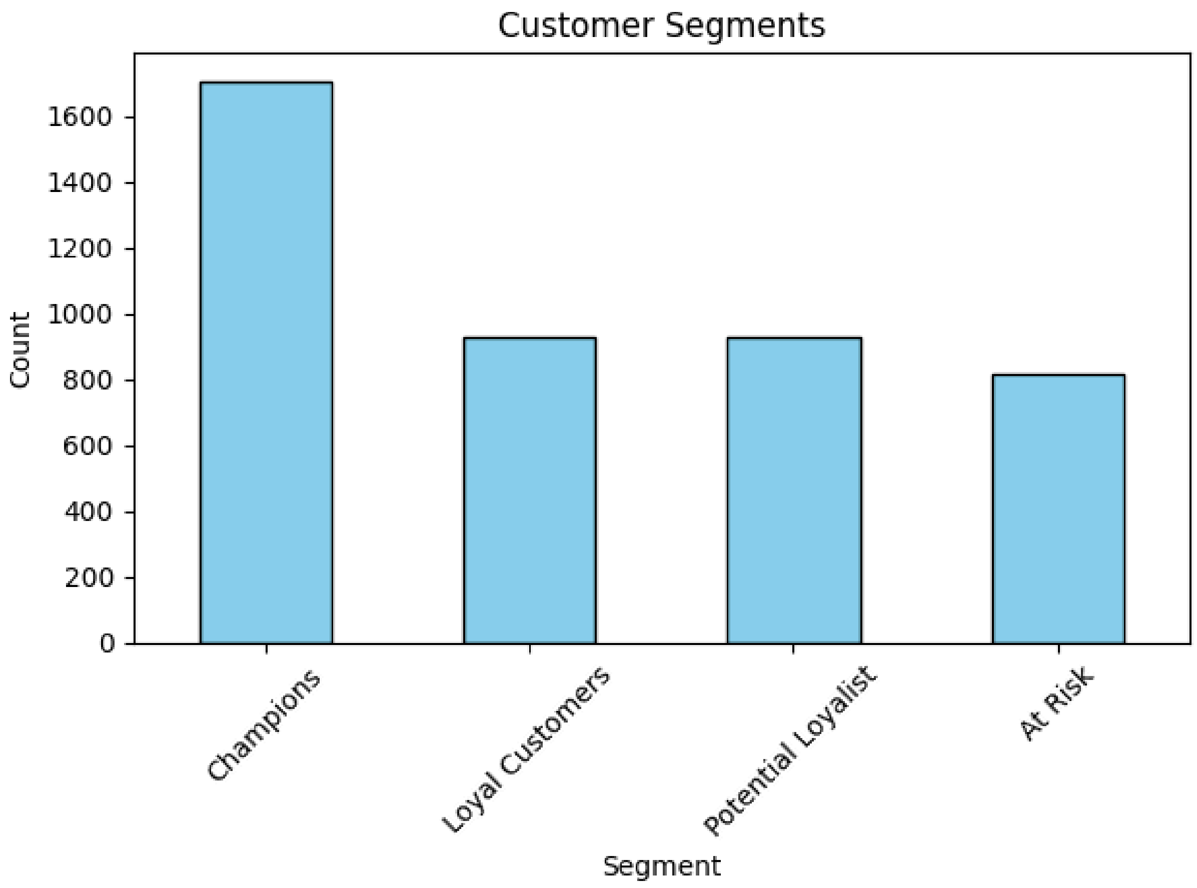
```
In [17]: print(returns_summary)
```

```
         StockCode                         Description  Quantity
1798         84347  ROTATING SILVER ANGELS T-LIGHT HLDR     -9374
266          21088         SET/6 FRUIT SALAD PAPER CUPS     -7128
269          21096       SET/6 FRUIT SALAD  PAPER PLATES     -7008
26           16047             POP ART PEN CASE & PENS     -5184
2077         85110   BLACK SILVER FLOWER T-LIGHT HOLDER     -5040
...            ...                                  ...       ...
1762        82607E           GLASS BONBON JAR. D'AMANDES        -1
2222         90203           SILVER CHARM NECKLACE 70CM        -1
37          16202E                    BLACK PHOTO ALBUM         -1
2245          PADS           PADS TO MATCH ALL CUSHIONS        -1
2249   gift_0001_80   Dotcomgiftshop Gift Voucher £80.00        -1

[2250 rows x 3 columns]
```

RFM Analysis

In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt

# 1. Prepare the data
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
rfm_df = df[df['Customer ID'].notnull()].copy()  # Avoid SettingWithCopyWarning
rfm_df['TotalPrice'] = rfm_df['Quantity'] * rfm_df['Price']

# 2. Create the RFM Table
snapshot_date = rfm_df['InvoiceDate'].max() + pd.Timedelta(days=1)
rfm = rfm_df.groupby('Customer ID').agg({
    'InvoiceDate': lambda x: (snapshot_date - x.max()).days,  # Recency
    'Invoice': 'nunique',                                      # Frequency
    'TotalPrice': 'sum'                                        # Monetary
}).reset_index()

rfm.rename(columns={
    'InvoiceDate': 'Recency',
    'Invoice': 'Frequency',
    'TotalPrice': 'Monetary'
}, inplace=True)

# 3. Score RFM values
rfm['R'] = pd.qcut(rfm['Recency'], 4, labels=[4, 3, 2, 1])
rfm['F'] = pd.qcut(rfm['Frequency'].rank(method='first'), 4, labels=[1, 2, 3, 4])
rfm['M'] = pd.qcut(rfm['Monetary'], 4, labels=[1, 2, 3, 4])

# 4. Combine RFM score
rfm['RFM_Segment'] = rfm['R'].astype(str) + rfm['F'].astype(str) + rfm['M'].astype(
rfm['RFM_Score'] = rfm[['R', 'F', 'M']].astype(int).sum(axis=1)

# 5. Customer segmentation
def segment_customer(score):
    if score >= 9:
        return 'Champions'
    elif score >= 7:
        return 'Loyal Customers'
    elif score >= 5:
        return 'Potential Loyalist'
    else:
        return 'At Risk'

rfm['Segment'] = rfm['RFM_Score'].apply(segment_customer)

# 6. Show results
print(rfm.head())

# 7. Plot segment distribution
rfm['Segment'].value_counts().plot(kind='bar', title='Customer Segments', ylabel='C
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
   Customer ID  Recency  Frequency  Monetary  R  F  M RFM_Segment  RFM_Score  \
0      12346.0       67         15    -64.68  2  4  1         241          7
1      12347.0        3          2   1323.32  4  2  3         423          9
2      12348.0       74          1    222.16  2  1  1         211          4
3      12349.0       43          4   2646.99  3  3  4         334         10
4      12351.0       11          1    300.93  4  1  2         412          7

           Segment
0  Loyal Customers
1        Champions
2          At Risk
3        Champions
4  Loyal Customers
```



```
In [ ]:
```