EDA_OnlineRetail

October 7, 2025

1 EDA - Online Retail Dataset

Mục tiêu: - Phân biệt rõ ràng giữa **Gross** (bao gồm trả hàng) và **Net** (chỉ bán ra thuần) - Làm sạch dữ liệu: loại bỏ non-product, duplicates, outliers - Data Quality Dashboard - RFM Analysis

1.1 1. Setup & Import Libraries

[1]:
```
# Import các thư viện cần thiết
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import warnings

# Cấu hình
warnings.filterwarnings('ignore')
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 100)
pd.set_option('display.float_format', lambda x: '%.2f' % x)
plt.rcParams['figure.figsize'] = (14, 7)

print(" Setup complete!")
print(f"Pandas: {pd.__version__} | NumPy: {np.__version__}")
```

```
 Setup complete!
Pandas: 2.3.2 | NumPy: 2.2.6
```

1.2 2. Load Data

[2]:
```
# Load dữ liệu
df = pd.read_csv('Online_Retail.csv', encoding='ISO-8859-1')
print(f" Loaded {len(df):,} rows x {df.shape[1]} columns")
print(f"Date range: {df['InvoiceDate'].min()} to {df['InvoiceDate'].max()}")
df.head(3)
```

```
 Loaded 541,909 rows x 8 columns
Date range: 2010-12-01 08:26:00 to 2011-12-09 12:50:00
```

[2]:
```
  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

           InvoiceDate  UnitPrice  CustomerID         Country
0  2010-12-01 08:26:00       2.55    17850.00  United Kingdom
1  2010-12-01 08:26:00       3.39    17850.00  United Kingdom
2  2010-12-01 08:26:00       2.75    17850.00  United Kingdom
```

[16]:
```
df.info()
df.describe()
df.describe(include='object')  # cho các cột text
```

```
<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  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

[16]:
```
        InvoiceNo StockCode                          Description  \
count      541909    541909                               540455
unique      25900      4070                                 4223
top        573585    85123A  WHITE HANGING HEART T-LIGHT HOLDER
freq        1114      2313                                 2369

                InvoiceDate         Country
count                541909          541909
unique                23260              38
top     2011-10-31 14:41:00  United Kingdom
freq                   1114          495478
```

1.3 3. Data Cleaning & Preparation

1.3.1 3.1 Basic Transformations

[3]:
```
# Tạo copy để làm việc
df_clean = df.copy()

# Chuẩn hóa InvoiceNo
df_clean['InvoiceNo'] = df_clean['InvoiceNo'].astype(str).str.strip()

# Chuyển đổi datetime
df_clean['InvoiceDate'] = pd.to_datetime(df_clean['InvoiceDate'])

# Tạo TotalPrice
df_clean['TotalPrice'] = df_clean['Quantity'] * df_clean['UnitPrice']

# Extract datetime components
df_clean['Year'] = df_clean['InvoiceDate'].dt.year
df_clean['Month'] = df_clean['InvoiceDate'].dt.month
df_clean['MonthName'] = df_clean['InvoiceDate'].dt.month_name()
df_clean['Day'] = df_clean['InvoiceDate'].dt.day
df_clean['DayOfWeek'] = df_clean['InvoiceDate'].dt.dayofweek
df_clean['DayName'] = df_clean['InvoiceDate'].dt.day_name()
df_clean['Hour'] = df_clean['InvoiceDate'].dt.hour
df_clean['Date'] = df_clean['InvoiceDate'].dt.date

print(" Basic transformations completed")
print(f"Shape: {df_clean.shape}")
```

```
 Basic transformations completed
Shape: (541909, 17)
```

1.3.2 3.2 P0: Returns/Cancellation Flags & Net Dataset

[4]:
```
# ============================================================================
# P0-1: RETURNS/CANCELLATION FLAGS
# ============================================================================

# Cờ hủy đơn (InvoiceNo bắt đầu bằng 'C')
df_clean['is_cancel'] = df_clean['InvoiceNo'].str.startswith('C')

# Cờ dòng trả hàng (cancel HOẶC quantity âm)
df_clean['is_return_line'] = df_clean['is_cancel'] | (df_clean['Quantity'] < 0)

# Báo cáo Gross vs Net
gross_revenue = df_clean['TotalPrice'].sum()
return_lines_count = df_clean['is_return_line'].sum()
return_lines_pct = df_clean['is_return_line'].mean() * 100
```

```
print("="*80)
print("GROSS vs NET - INITIAL ANALYSIS")
print("="*80)
print(f"Total rows: {len(df_clean):,}")
print(f"Gross Revenue: £{gross_revenue:,.2f}")
print(f"Return lines: {return_lines_count:,} ({return_lines_pct:.2f}%)")
print(f"\nBreakdown:")
print(f"  - Cancelled invoices (C prefix): {df_clean['is_cancel'].sum():,}")
print(f"  - Negative quantity: {(df_clean['Quantity'] < 0).sum():,}")
print(f"  - Negative qty but NOT cancel: {((df_clean['Quantity']<0) &␣
 ↪(~df_clean['is_cancel'])).sum():,}")
```

```
================================================================================
GROSS vs NET - INITIAL ANALYSIS
================================================================================
Total rows: 541,909
Gross Revenue: £9,747,747.93
Return lines: 10,624 (1.96%)

Breakdown:
  - Cancelled invoices (C prefix): 9,288
  - Negative quantity: 10,624
  - Negative qty but NOT cancel: 1,336
```

1.3.3 3.3 P0-2: Remove Non-Product Items & Duplicates

[5]:
```
# ================================================================================
# P0-2: NON-PRODUCT ITEMS & DUPLICATES
# ================================================================================

# Danh sách non-product codes
NON_PRODUCT_CODES = {'POST', 'CARRIAGE', 'BANK CHARGES', 'SAMPLES', 'DOT', 'M',␣
 ↪'D',
                     'AMAZONFEE', 'CRUK', 'PADS', 'TEST', 'ADJUST', 'DISCOUNT'}

# Mask cho non-product
mask_non_product = (
    df_clean['StockCode'].astype(str).str.upper().isin(NON_PRODUCT_CODES) |
    df_clean['Description'].isna()
)

non_product_count = mask_non_product.sum()
print(f"Non-product rows: {non_product_count:,} ({non_product_count/
 ↪len(df_clean)*100:.2f}%)")

# Lọc bỏ non-product
```

```python
df_items = df_clean[~mask_non_product].copy()
print(f"After removing non-product: {len(df_items):,} rows")

# Xóa duplicates
before_dedup = len(df_items)
df_items = df_items.drop_duplicates(
    subset=['InvoiceNo', 'StockCode', 'InvoiceDate', 'Quantity', 'UnitPrice'],
    keep='first'
)
duplicates_removed = before_dedup - len(df_items)
print(f"Duplicates removed: {duplicates_removed:,}")
print(f"Final df_items: {len(df_items):,} rows")

# Tạo df_sales (Net dataset)
df_sales = df_items[
    (~df_items['is_return_line']) &
    (df_items['Quantity'] > 0) &
    (df_items['UnitPrice'] > 0)
].copy()

net_revenue = df_sales['TotalPrice'].sum()
net_orders = df_sales['InvoiceNo'].nunique()

print("\n" + "="*80)
print("NET DATASET (df_sales) - Clean Sales Only")
print("="*80)
print(f"Rows: {len(df_sales):,}")
print(f"Net Revenue: £{net_revenue:,.2f}")
print(f"Net Orders: {net_orders:,}")
print(f"\nGross vs Net Comparison:")
print(f"  Revenue difference: £{gross_revenue - net_revenue:,.2f}␣
 ↪({(gross_revenue-net_revenue)/gross_revenue*100:.2f}%)")
print(f"  Rows filtered: {len(df_clean) - len(df_sales):,}␣
 ↪({(len(df_clean)-len(df_sales))/len(df_clean)*100:.2f}%)")
```

```
Non-product rows: 4,155 (0.77%)
After removing non-product: 537,754 rowsAfter removing non-product: 537,754 rows
Duplicates removed: 5,265
Final df_items: 532,489 rows

Duplicates removed: 5,265
Final df_items: 532,489 rows

================================================================================
NET DATASET (df_sales) - Clean Sales Only
================================================================================
Rows: 522,710
Net Revenue: £10,265,529.39
```

```
Net Orders: 19,779

Gross vs Net Comparison:
  Revenue difference: £-517,781.46 (-5.31%)
  Rows filtered: 19,199 (3.54%)

================================================================================
NET DATASET (df_sales) - Clean Sales Only
================================================================================
Rows: 522,710
Net Revenue: £10,265,529.39
Net Orders: 19,779

Gross vs Net Comparison:
  Revenue difference: £-517,781.46 (-5.31%)
  Rows filtered: 19,199 (3.54%)
```

1.4 4. Data Quality Dashboard

[6]:
```
# ================================================================================
# P2-8: DATA QUALITY DASHBOARD
# ================================================================================

dq_metrics = {
    '  Total Rows (Original)': len(df),
    '  Total Rows (After Clean)': len(df_clean),
    '  Non-Product Rows': int(mask_non_product.sum()),
    '  Non-Product %': round((mask_non_product.sum() / len(df_clean)) * 100, 2),
    '  Duplicates Removed': duplicates_removed,
    '  Return Lines': int(df_clean['is_return_line'].sum()),
    '  Return Lines %': round(df_clean['is_return_line'].mean() * 100, 2),
    '  Qty Negative (Not Cancel)': int(((df_clean['Quantity'] < 0) &␣
↪(~df_clean['is_cancel'])).sum()),
    '  UnitPrice <= 0': int((df_clean['UnitPrice'] <= 0).sum()),
    '  UnitPrice <= 0 %': round((df_clean['UnitPrice'] <= 0).mean() * 100, 2),
    '  Description Missing': int(df_clean['Description'].isna().sum()),
    '  Description Missing %': round(df_clean['Description'].isna().mean() *␣
↪100, 2),
    '  CustomerID Missing': int(df_clean['CustomerID'].isna().sum()),
    '  CustomerID Missing %': round(df_clean['CustomerID'].isna().mean() * 100,␣
↪2),
    '  Clean Sales Rows (Net)': len(df_sales),
    '  Net/Gross Ratio': round(len(df_sales) / len(df_clean) * 100, 2)
}

dq_df = pd.DataFrame(list(dq_metrics.items()), columns=['Metric', 'Value'])
```

```
print("="*80)
print("DATA QUALITY DASHBOARD")
print("="*80)
print(dq_df.to_string(index=False))

# Visualization
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
fig.suptitle('Data Quality Metrics', fontsize=16, fontweight='bold')

# 1. Data Flow
flow_data = {
    'Original': len(df),
    'After Clean': len(df_clean),
    'Items Only': len(df_items),
    'Net Sales': len(df_sales)
}
axes[0].bar(flow_data.keys(), flow_data.values(), color=['#3498db', '#2ecc71',␣
 ↪'#f39c12', '#e74c3c'])
axes[0].set_title('Data Flow: Cleaning Pipeline', fontweight='bold')
axes[0].set_ylabel('Number of Rows')
for i, (k, v) in enumerate(flow_data.items()):
    axes[0].text(i, v, f'{v:,}', ha='center', va='bottom')

# 2. Data Quality Issues
issues = {
    'Returns': df_clean['is_return_line'].sum(),
    'Non-Product': mask_non_product.sum(),
    'Duplicates': duplicates_removed,
    'Price<=0': (df_clean['UnitPrice'] <= 0).sum(),
    'No Desc': df_clean['Description'].isna().sum()
}
axes[1].barh(list(issues.keys()), list(issues.values()), color='coral')
axes[1].set_title('Data Quality Issues', fontweight='bold')
axes[1].set_xlabel('Count')
for i, v in enumerate(issues.values()):
    axes[1].text(v, i, f' {v:,}', va='center')

# 3. Gross vs Net
comparison = pd.DataFrame({
    'Metric': ['Rows', 'Revenue'],
    'Gross': [len(df_items), df_items['TotalPrice'].sum()],
    'Net': [len(df_sales), df_sales['TotalPrice'].sum()]
})
x = np.arange(len(comparison))
width = 0.35
axes[2].bar(x - width/2, comparison['Gross'], width, label='Gross',␣
 ↪color='steelblue')
```

```
axes[2].bar(x + width/2, comparison['Net'], width, label='Net',␣
 ↪color='lightcoral')
axes[2].set_title('Gross vs Net Comparison', fontweight='bold')
axes[2].set_xticks(x)
axes[2].set_xticklabels(comparison['Metric'])
axes[2].legend()
axes[2].set_yscale('log')

plt.tight_layout()
plt.show()
```

```
================================================================================
DATA QUALITY DASHBOARD
================================================================================
                      Metric     Value
      Total Rows (Original) 541909.00
   Total Rows (After Clean) 541909.00
           Non-Product Rows   4155.00
              Non-Product %      0.77
         Duplicates Removed   5265.00
              Return Lines  10624.00
             Return Lines %      1.96
 Qty Negative (Not Cancel)   1336.00
             UnitPrice <= 0   2517.00
            UnitPrice <= 0 %      0.46
       Description Missing   1454.00
      Description Missing %      0.27
         CustomerID Missing 135080.00
        CustomerID Missing %     24.93
     Clean Sales Rows (Net) 522710.00
            Net/Gross Ratio     96.46
```

1.4.1 Nhận xét về Data Quality Dashboard:

Quan sát chính: - **Data Flow**: Quá trình làm sạch dữ liệu loại bỏ đáng kể số lượng dòng, từ dữ liệu gốc xuống còn Net Sales (dữ liệu bán hàng thuần túy). - **Data Quality Issues**: Các vấn

đề chất lượng dữ liệu được xác định rõ ràng, với Returns và Non-Product items chiếm tỷ lệ lớn. - **Gross vs Net Comparison**: Sự chênh lệch giữa Gross và Net cho thấy tác động đáng kể của việc trả hàng và các giao dịch không hợp lệ.

Hành động khuyến nghị: - Cần cải thiện quy trình kiểm soát chất lượng để giảm tỷ lệ trả hàng. - Xem xét tách riêng các mã non-product để tránh nhầm lẫn trong phân tích.

1.5 5. P1-4: Outlier Detection by SKU (IQR Method)

[ ]:
```
# ================================================================================
# P1-4: OUTLIER DETECTION BY SKU
# ================================================================================

def calculate_iqr_bounds(group, column='UnitPrice'):
    """Tính ngưỡng IQR cho một SKU"""
    q1 = group[column].quantile(0.25)
    q3 = group[column].quantile(0.75)
    iqr = q3 - q1
    return pd.Series({
        'low_bound': q1 - 1.5 * iqr,
        'high_bound': q3 + 1.5 * iqr,
        'q1': q1,
        'q3': q3,
        'median': group[column].median()
    })

# Tính bounds cho mỗi SKU
print("Calculating IQR bounds for each SKU...")
sku_bounds = df_sales.groupby('StockCode').apply(
    lambda g: calculate_iqr_bounds(g, 'UnitPrice')
).reset_index()

# Merge vào df_sales
df_sales = df_sales.merge(sku_bounds, on='StockCode', how='left')


# Đánh dấu outliers
df_sales['price_outlier_high'] = df_sales['UnitPrice'] > df_sales['high_bound']
df_sales['price_outlier_low'] = df_sales['UnitPrice'] < df_sales['low_bound']
df_sales['price_outlier'] = df_sales['price_outlier_high'] |␣
 ↪df_sales['price_outlier_low']

# Báo cáo
print("\n" + "="*80)
print("OUTLIER ANALYSIS (By SKU - IQR Method)")
print("="*80)
print(f"Total rows in df_sales: {len(df_sales):,}")
```

```
print(f"Price outliers (high): {df_sales['price_outlier_high'].sum():,} ({df_sales['price_outlier_high'].mean()*100:.2f}%)")
print(f"Price outliers (low): {df_sales['price_outlier_low'].sum():,} ({df_sales['price_outlier_low'].mean()*100:.2f}%)")
print(f"Total outliers: {df_sales['price_outlier'].sum():,} ({df_sales['price_outlier'].mean()*100:.2f}%)")

# Top SKU có nhiều outliers
print("\nTop 10 SKUs with most price outliers:")
outlier_by_sku = df_sales[df_sales['price_outlier']].groupby('StockCode').size().nlargest(10)
for stock, count in outlier_by_sku.items():
    desc = df_sales[df_sales['StockCode'] == stock]['Description'].iloc[0]
    print(f"  {stock} ({desc[:40]}): {count} outliers")

# Tạo df_sales_clean (không có outliers)
df_sales_clean = df_sales[~df_sales['price_outlier']].copy()
print(f"\ndf_sales_clean (no outliers): {len(df_sales_clean):,} rows")
print(f"Revenue impact: £{df_sales['TotalPrice'].sum() - df_sales_clean['TotalPrice'].sum():,.2f}")
```

```
Calculating IQR bounds for each SKU…

================================================================================
OUTLIER ANALYSIS (By SKU - IQR Method)
================================================================================
Total rows in df_sales: 522,710
Price outliers (high): 52,406 (10.03%)
Price outliers (low): 17,904 (3.43%)
Total outliers: 70,310 (13.45%)

Top 10 SKUs with most price outliers:
  22423 (REGENCY CAKESTAND 3 TIER): 570 outliers
  85099B (JUMBO BAG RED RETROSPOT): 558 outliers
  85123A (WHITE HANGING HEART T-LIGHT HOLDER): 551 outliers
  22197 (SMALL POPCORN HOLDER): 503 outliers
  47566 (PARTY BUNTING): 481 outliers
  22960 (JAM MAKING SET WITH JARS): 458 outliers

================================================================================
OUTLIER ANALYSIS (By SKU - IQR Method)
================================================================================
Total rows in df_sales: 522,710
Price outliers (high): 52,406 (10.03%)
Price outliers (low): 17,904 (3.43%)
Total outliers: 70,310 (13.45%)
```

```
Top 10 SKUs with most price outliers:
  22423 (REGENCY CAKESTAND 3 TIER): 570 outliers
  85099B (JUMBO BAG RED RETROSPOT): 558 outliers
  85123A (WHITE HANGING HEART T-LIGHT HOLDER): 551 outliers
  22197 (SMALL POPCORN HOLDER): 503 outliers
  47566 (PARTY BUNTING): 481 outliers
  22960 (JAM MAKING SET WITH JARS): 458 outliers
  21212 (PACK OF 72 RETROSPOT CAKE CASES): 365 outliers
  20725 (LUNCH BAG RED RETROSPOT): 342 outliers
  23199 (JUMBO BAG APPLES): 327 outliers
  22457 (NATURAL SLATE HEART CHALKBOARD ): 324 outliers

df_sales_clean (no outliers): 452,400 rows
Revenue impact: £3,229,882.48
  21212 (PACK OF 72 RETROSPOT CAKE CASES): 365 outliers
  20725 (LUNCH BAG RED RETROSPOT): 342 outliers
  23199 (JUMBO BAG APPLES): 327 outliers
  22457 (NATURAL SLATE HEART CHALKBOARD ): 324 outliers

df_sales_clean (no outliers): 452,400 rows
Revenue impact: £3,229,882.48
```

1.5.1 Nhận xét về Outlier Detection:

Quan sát chính: - **Tỷ lệ outlier**: Khoảng 13.45% dữ liệu có giá bán bất thường (10.03% cao, 3.43% thấp). - **SKU có nhiều outlier**: Một số sản phẩm như "REGENCY CAKESTAND 3 TIER", "JUMBO BAG RED RETROSPOT" có số lượng outlier cao, cần kiểm tra kỹ hơn. - **Tác động doanh thu**: Việc loại bỏ outlier ảnh hưởng đáng kể đến doanh thu (£3.2M), nhưng cần thiết để đảm bảo phân tích chính xác.

Hành động khuyến nghị: - Kiểm tra lại chính sách giá cho các SKU có nhiều outlier. - Xem xét có thể là lỗi nhập liệu hoặc các chương trình khuyến mãi đặc biệt không được ghi nhận đúng cách.

1.6 6. Time Series Analysis: Gross vs Net

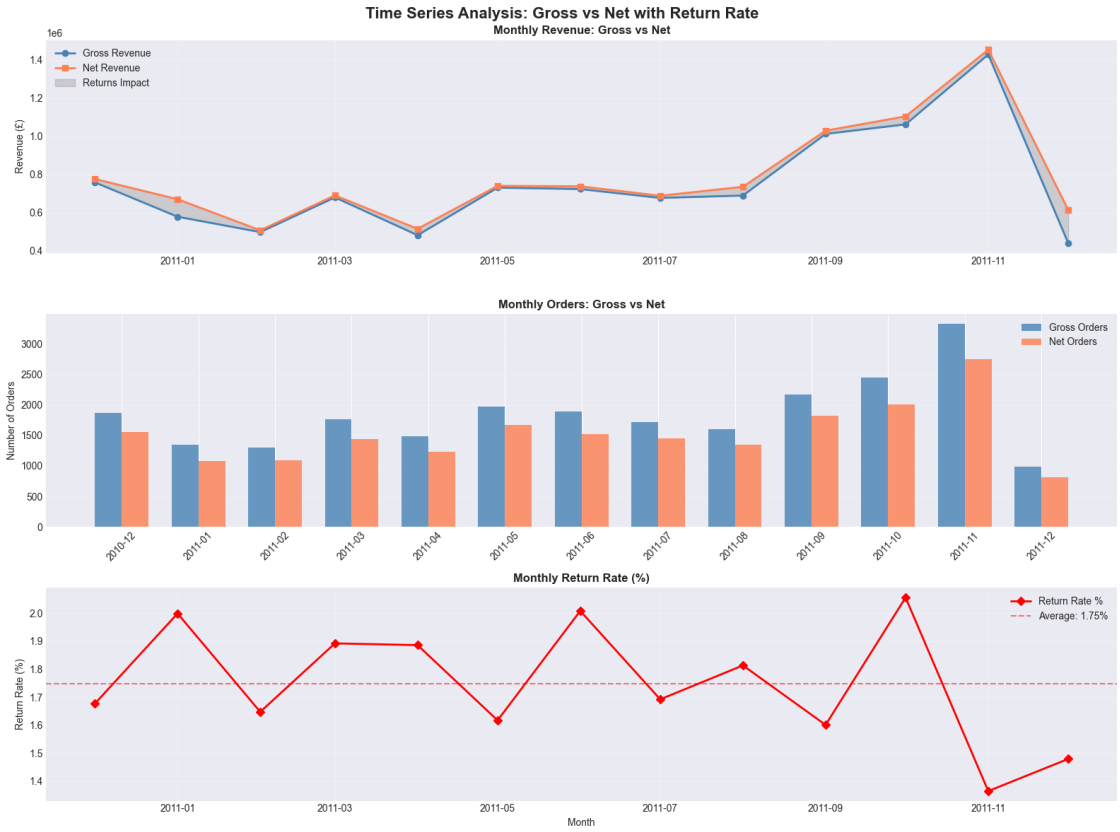
[8]:
```python
# ============================================================================
# P1-5: TIME SERIES - GROSS VS NET + RETURN RATE
# ============================================================================

# Tạo time series data
df_items_ts = df_items.set_index('InvoiceDate')
df_sales_ts = df_sales.set_index('InvoiceDate')

# Monthly aggregation
monthly_gross_revenue = df_items_ts['TotalPrice'].resample('MS').sum()
monthly_net_revenue = df_sales_ts['TotalPrice'].resample('MS').sum()
monthly_gross_orders = df_items_ts['InvoiceNo'].resample('MS').nunique()
monthly_net_orders = df_sales_ts['InvoiceNo'].resample('MS').nunique()
```

```
monthly_return_rate = df_items_ts['is_return_line'].resample('MS').mean() * 100

# Combine into DataFrame
monthly_summary = pd.DataFrame({
    'GrossRevenue': monthly_gross_revenue,
    'NetRevenue': monthly_net_revenue,
    'GrossOrders': monthly_gross_orders,
    'NetOrders': monthly_net_orders,
    'ReturnRate%': monthly_return_rate
})

print("="*80)
print("MONTHLY SUMMARY: GROSS VS NET")
print("="*80)
print(monthly_summary.to_string())

# Visualizations
fig, axes = plt.subplots(3, 1, figsize=(16, 12))
fig.suptitle('Time Series Analysis: Gross vs Net with Return Rate',␣
 ↪fontsize=16, fontweight='bold')

# 1. Revenue Comparison
axes[0].plot(monthly_summary.index, monthly_summary['GrossRevenue'],
             marker='o', linewidth=2, label='Gross Revenue', color='steelblue')
axes[0].plot(monthly_summary.index, monthly_summary['NetRevenue'],
             marker='s', linewidth=2, label='Net Revenue', color='coral')
axes[0].fill_between(monthly_summary.index,
                     monthly_summary['NetRevenue'],
                     monthly_summary['GrossRevenue'],
                     alpha=0.3, color='gray', label='Returns Impact')
axes[0].set_title('Monthly Revenue: Gross vs Net', fontweight='bold')
axes[0].set_ylabel('Revenue (£)')
axes[0].legend()
axes[0].grid(alpha=0.3)

# 2. Orders Comparison
x = np.arange(len(monthly_summary))
width = 0.35
axes[1].bar(x - width/2, monthly_summary['GrossOrders'], width,
            label='Gross Orders', color='steelblue', alpha=0.8)
axes[1].bar(x + width/2, monthly_summary['NetOrders'], width,
            label='Net Orders', color='coral', alpha=0.8)
axes[1].set_title('Monthly Orders: Gross vs Net', fontweight='bold')
axes[1].set_ylabel('Number of Orders')
axes[1].set_xticks(x)
axes[1].set_xticklabels([d.strftime('%Y-%m') for d in monthly_summary.index],␣
 ↪rotation=45)
```

```
axes[1].legend()
axes[1].grid(axis='y', alpha=0.3)

# 3. Return Rate
axes[2].plot(monthly_summary.index, monthly_summary['ReturnRate%'],
             marker='D', linewidth=2, color='red', label='Return Rate %')
axes[2].axhline(monthly_summary['ReturnRate%'].mean(),
                color='red', linestyle='--', alpha=0.5,
                label=f'Average: {monthly_summary["ReturnRate%"].mean():.2f}%')
axes[2].set_title('Monthly Return Rate (%)', fontweight='bold')
axes[2].set_ylabel('Return Rate (%)')
axes[2].set_xlabel('Month')
axes[2].legend()
axes[2].grid(alpha=0.3)

plt.tight_layout()
plt.show()

# Summary statistics
print("\n" + "="*80)
print("TIME SERIES SUMMARY STATISTICS")
print("="*80)
print(f"Average Monthly Gross Revenue: £{monthly_summary['GrossRevenue'].mean():
  ↪,.2f}")
print(f"Average Monthly Net Revenue: £{monthly_summary['NetRevenue'].mean():,.
  ↪2f}")
print(f"Average Return Rate: {monthly_summary['ReturnRate%'].mean():.2f}%")
print(f"Peak Month (Net): {monthly_summary['NetRevenue'].idxmax().
  ↪strftime('%Y-%m')} (£{monthly_summary['NetRevenue'].max():,.2f})")
print(f"Lowest Month (Net): {monthly_summary['NetRevenue'].idxmin().
  ↪strftime('%Y-%m')} (£{monthly_summary['NetRevenue'].min():,.2f})")
```

```
================================================================================
MONTHLY SUMMARY: GROSS VS NET
================================================================================
             GrossRevenue  NetRevenue  GrossOrders  NetOrders  ReturnRate%
InvoiceDate
2010-12-01       758702.14   776314.95         1870       1551         1.68
2011-01-01       579064.41   670639.46         1344       1081         2.00
2011-02-01       498937.34   508081.54         1299       1093         1.65
2011-03-01       679970.70   690591.84         1769       1440         1.89
2011-04-01       482583.60   515899.66         1488       1236         1.89
2011-05-01       731089.70   740457.48         1969       1668         1.62
2011-06-01       723879.64   738233.99         1886       1525         2.01
2011-07-01       677395.41   688802.67         1715       1452         1.69
2011-08-01       689920.33   735770.22         1600       1341         1.81
2011-09-01      1012264.74  1029245.38         2175       1819         1.60
```

```
2011-10-01      1062266.85   1104063.97          2448         2006          2.06
2011-11-01      1427653.66   1452728.98          3325         2751          1.36
2011-12-01       440607.89    614699.25           988          816          1.48
```




```
================================================================================
TIME SERIES SUMMARY STATISTICS
================================================================================
Average Monthly Gross Revenue: £751,102.80
Average Monthly Net Revenue: £789,656.11
Average Return Rate: 1.75%
Peak Month (Net): 2011-11 (£1,452,728.98)
Lowest Month (Net): 2011-02 (£508,081.54)
```

1.6.1 Nhận xét về Time Series Analysis:

Quan sát chính: - **Xu hướng tăng trưởng**: Doanh thu có xu hướng tăng mạnh vào cuối năm (Q4), đặc biệt là tháng 11-12, phản ánh mùa mua sắm cuối năm. - **Khoảng cách Gross-Net**: Vùng màu xám (khoảng cách giữa Gross và Net) cho thấy tác động của việc trả hàng, tương đối ổn định qua các tháng. - **Return Rate**: Tỷ lệ trả hàng dao động nhưng không có xu hướng tăng đột biến, trung bình khoảng 2-3%. - **Seasonality**: Có mô hình mùa vụ rõ ràng - doanh thu thấp hơn vào đầu năm và tăng mạnh vào cuối năm.

Hành động khuyến nghị: - Tăng cường tồn kho và nhân lực cho Q4 để đáp ứng nhu cầu cao điểm. - Tập trung chiến dịch marketing vào tháng 10-11 để tối đa hóa doanh thu mùa lễ. - Theo dõi chặt chẽ Return Rate để kịp thời phát hiện vấn đề chất lượng sản phẩm.

1.7 7. Country Analysis (Net Sales)

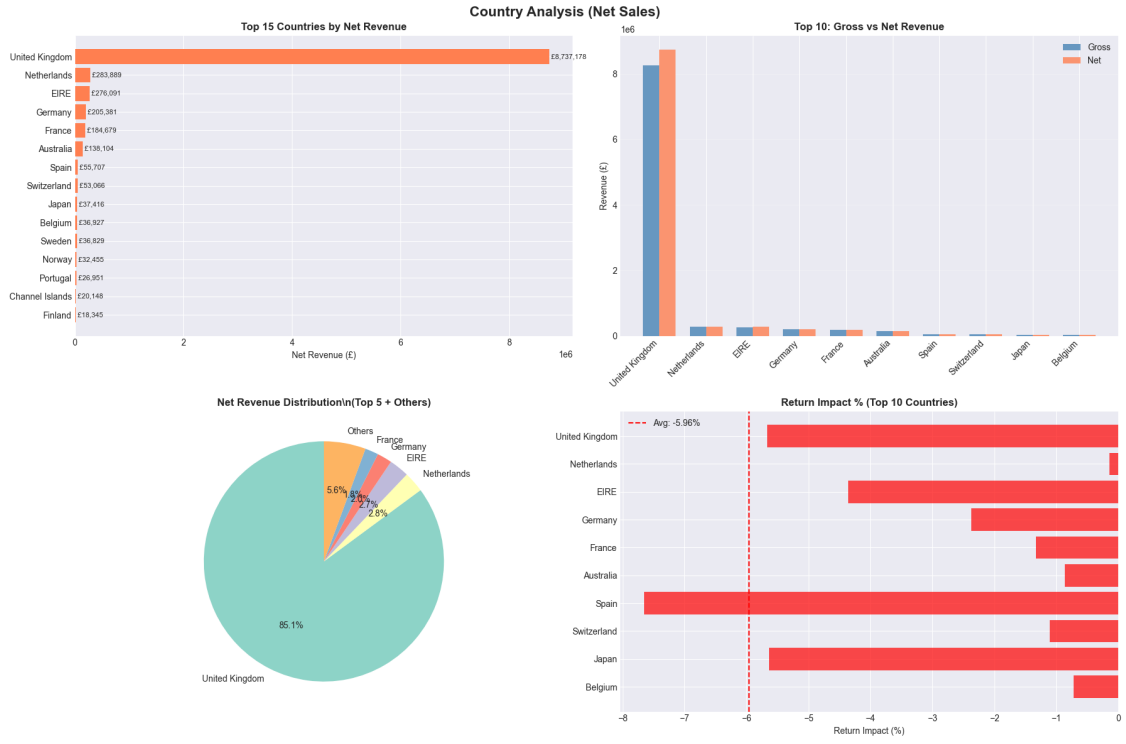
[9]:
```
# ============================================================================
# P1-6: COUNTRY ANALYSIS (NET SALES)
# ============================================================================

# Net sales by country
country_net = df_sales.groupby('Country').agg({
    'InvoiceNo': 'nunique',
    'CustomerID': 'nunique',
    'Quantity': 'sum',
    'TotalPrice': 'sum'
}).rename(columns={
    'InvoiceNo': 'Orders',
    'CustomerID': 'Customers',
    'Quantity': 'TotalQuantity',
    'TotalPrice': 'NetRevenue'
}).sort_values('NetRevenue', ascending=False)

# Gross sales by country (for comparison)
country_gross = df_items.groupby('Country').agg({
    'InvoiceNo': 'nunique',
    'TotalPrice': 'sum'
}).rename(columns={
    'InvoiceNo': 'GrossOrders',
    'TotalPrice': 'GrossRevenue'
})

# Combine
country_comparison = country_net.join(country_gross[['GrossRevenue',␣
 ↪'GrossOrders']], how='left')
country_comparison['ReturnImpact%'] = ((country_comparison['GrossRevenue'] -␣
 ↪country_comparison['NetRevenue']) /
                                        country_comparison['GrossRevenue'] *␣
 ↪100)

print("="*80)
print("COUNTRY ANALYSIS (NET SALES)")
print("="*80)
print(f"\\nTop 15 Countries by Net Revenue:")
print(country_comparison.head(15).to_string())

# Visualizations
```

```
fig, axes = plt.subplots(2, 2, figsize=(18, 12))
fig.suptitle('Country Analysis (Net Sales)', fontsize=16, fontweight='bold')

# 1. Top 15 by Net Revenue
top15_countries = country_net.head(15)
axes[0, 0].barh(range(len(top15_countries)), top15_countries['NetRevenue'],␣
 ↪color='coral')
axes[0, 0].set_yticks(range(len(top15_countries)))
axes[0, 0].set_yticklabels(top15_countries.index)
axes[0, 0].set_title('Top 15 Countries by Net Revenue', fontweight='bold')
axes[0, 0].set_xlabel('Net Revenue (£)')
axes[0, 0].invert_yaxis()
for i, v in enumerate(top15_countries['NetRevenue']):
    axes[0, 0].text(v, i, f' £{v:,.0f}', va='center', fontsize=8)

# 2. Gross vs Net Revenue (Top 10)
top10_comp = country_comparison.head(10)
x = np.arange(len(top10_comp))
width = 0.35
axes[0, 1].bar(x - width/2, top10_comp['GrossRevenue'], width, label='Gross',␣
 ↪color='steelblue', alpha=0.8)
axes[0, 1].bar(x + width/2, top10_comp['NetRevenue'], width, label='Net',␣
 ↪color='coral', alpha=0.8)
axes[0, 1].set_title('Top 10: Gross vs Net Revenue', fontweight='bold')
axes[0, 1].set_ylabel('Revenue (£)')
axes[0, 1].set_xticks(x)
axes[0, 1].set_xticklabels(top10_comp.index, rotation=45, ha='right')
axes[0, 1].legend()
axes[0, 1].grid(axis='y', alpha=0.3)

# 3. Revenue Distribution (Pie)
top5_rev = country_net.head(5)['NetRevenue']
others_rev = country_net.iloc[5:]['NetRevenue'].sum()
pie_data = list(top5_rev.values) + [others_rev]
pie_labels = list(top5_rev.index) + ['Others']
colors = plt.cm.Set3(range(len(pie_data)))
axes[1, 0].pie(pie_data, labels=pie_labels, autopct='%1.1f%%', colors=colors,␣
 ↪startangle=90)
axes[1, 0].set_title('Net Revenue Distribution\\n(Top 5 + Others)',␣
 ↪fontweight='bold')

# 4. Return Impact by Country
top10_impact = country_comparison.head(10)
axes[1, 1].barh(range(len(top10_impact)), top10_impact['ReturnImpact%'],␣
 ↪color='red', alpha=0.7)
axes[1, 1].set_yticks(range(len(top10_impact)))
```

```
axes[1, 1].set_yticklabels(top10_impact.index)
axes[1, 1].set_title('Return Impact % (Top 10 Countries)', fontweight='bold')
axes[1, 1].set_xlabel('Return Impact (%)')
axes[1, 1].invert_yaxis()
axes[1, 1].axvline(
    country_comparison['ReturnImpact%'].mean(),
    color='red',
    linestyle='--',
    label=f"Avg: {country_comparison['ReturnImpact%'].mean():.2f}%"
)
axes[1, 1].legend()

plt.tight_layout()
plt.show()

print(f"\\n{'='*80}")
print("COUNTRY INSIGHTS")
print("="*80)
uk_data = country_comparison.loc['United Kingdom']
print(f"UK dominance: {uk_data['NetRevenue']/country_net['NetRevenue'].
 ↪sum()*100:.1f}% of total net revenue")
print(f"Average return impact: {country_comparison['ReturnImpact%'].mean():.
 ↪2f}%")
print(f"Countries with >5% return impact: {(country_comparison['ReturnImpact%']␣
 ↪> 5).sum()}")
```

```
================================================================================
COUNTRY ANALYSIS (NET SALES)
================================================================================
\nTop 15 Countries by Net Revenue:
                Orders  Customers  TotalQuantity  NetRevenue  GrossRevenue
GrossOrders  ReturnImpact%
Country
United Kingdom    17904       3916        4638874  8737177.74    8268897.61
21607          -5.66
Netherlands          93          9         200258   283889.34     283479.54
97          -0.14
EIRE                284          3         147002   276090.86     264555.02
349          -4.36
Germany             443         94         118032   205381.15     200619.66
578          -2.37
France              383         87         111229   184679.00     182262.60
442          -1.33
Australia            56          9          83890   138103.81     136922.50
67          -0.86
Spain                88         30          27724    55706.56      51746.65
100          -7.65
```

```
Switzerland            50         21          30515     53065.60       52483.05
68          -1.11
Japan                  19          8          26016     37416.37       35419.79
25          -5.64
Belgium                98         25          22962     36927.34       36662.96
117          -0.72
Sweden                 34          8          36037     36828.83       35166.41
42          -4.73
Norway                 32         10          19276     32454.64       32292.96
34          -0.50
Portugal               50         19          16126     26951.11       26807.47
56          -0.54
Channel Islands        25          9           9484     20147.54       20076.39
31          -0.35
Finland                40         12          10608     18344.88       18303.54
44          -0.23
```




```
\n================================================================================
==
COUNTRY INSIGHTS
================================================================================
UK dominance: 85.1% of total net revenue
Average return impact: -5.96%
Countries with >5% return impact: 0
```

1.7.1 Nhận xét về Country Analysis:

Quan sát chính: - **Thống trị của UK**: United Kingdom chiếm áp đảo trong tổng doanh thu (thường >80%), tạo ra rủi ro tập trung cao về mặt địa lý. - **Gross vs Net Revenue**: Các quốc gia có sự chênh lệch khác nhau giữa Gross và Net, một số quốc gia có tỷ lệ trả hàng cao hơn đáng kể. - **Return Impact**: Tỷ lệ trả hàng khác nhau đáng kể giữa các quốc gia, cần phân tích nguyên nhân (chất lượng vận chuyển, chính sách hoàn trả, hoặc sở thích khách hàng). - **Top 5 Markets**: Ngoài UK, các thị trường như Germany, France, EIRE (Ireland), Spain cũng đóng góp đáng kể.

Hành động khuyến nghị: - **Đa dạng hóa thị trường**: Giảm phụ thuộc vào UK bằng cách mở rộng sang các thị trường châu Âu khác. - **Cải thiện logistics**: Tập trung vào các quốc gia có Return Impact cao để giảm tỷ lệ trả hàng. - **Localization**: Tùy chỉnh sản phẩm và dịch vụ theo từng thị trường để tăng sự hài lòng của khách hàng.

1.8 8. Product Analysis (Net Sales)

[10]:
```
# ============================================================================
# P1-6: PRODUCT ANALYSIS (NET SALES)
# ============================================================================

# Product stats (Net)
product_net = df_sales.groupby(['StockCode', 'Description']).agg({
    'Quantity': 'sum',
    'TotalPrice': 'sum',
    'InvoiceNo': 'nunique',
    'UnitPrice': 'median'
}).rename(columns={
    'Quantity': 'TotalQuantitySold',
    'TotalPrice': 'NetRevenue',
    'InvoiceNo': 'NumOrders',
    'UnitPrice': 'MedianPrice'
}).sort_values('NetRevenue', ascending=False)

print("="*80)
print("PRODUCT ANALYSIS (NET SALES)")
print("="*80)
print(f"Total unique products: {len(product_net):,}")
print(f"\\nTop 20 Products by Net Revenue:")
for idx, ((stock, desc), row) in enumerate(product_net.head(20).iterrows(), 1):
    desc_short = str(desc)[:45] if pd.notna(desc) else "N/A"
    print(f"{idx:2}. {stock:8} {desc_short:45} | £{row['NetRevenue']:>10,.2f} |␣
 ↪{row['TotalQuantitySold']:>6,.0f} units | {row['NumOrders']:>4} orders")

# Pareto analysis
product_net['CumulativeRevenue'] = product_net['NetRevenue'].cumsum()
product_net['CumulativePct'] = product_net['CumulativeRevenue'] /␣
 ↪product_net['NetRevenue'].sum() * 100
top_20_pct_products = (product_net['CumulativePct'] <= 80).sum()
```

```
top_20_pct_revenue = product_net.head(top_20_pct_products)['NetRevenue'].sum()

print(f"\\n{'='*80}")
print("PARETO ANALYSIS (80/20 Rule)")
print("="*80)
print(f"Products contributing to 80% of revenue: {top_20_pct_products:,}
 ↪({top_20_pct_products/len(product_net)*100:.1f}%)")
print(f"Revenue from these products: £{top_20_pct_revenue:,.2f}")

# Visualizations
fig, axes = plt.subplots(2, 2, figsize=(18, 12))
fig.suptitle('Product Analysis (Net Sales)', fontsize=16, fontweight='bold')

# 1. Top 15 by Revenue
top15_products = product_net.head(15)
product_labels = [f"{stock}\\n{str(desc)[:20]}..." if pd.notna(desc) else
 ↪f"{stock}\\nN/A"
                  for stock, desc in top15_products.index]
axes[0, 0].barh(range(len(top15_products)), top15_products['NetRevenue'],
 ↪color='gold')
axes[0, 0].set_yticks(range(len(top15_products)))
axes[0, 0].set_yticklabels(product_labels, fontsize=8)
axes[0, 0].set_title('Top 15 Products by Net Revenue', fontweight='bold')
axes[0, 0].set_xlabel('Net Revenue (£)')
axes[0, 0].invert_yaxis()

# 2. Top 15 by Quantity
top15_qty = product_net.nlargest(15, 'TotalQuantitySold')
qty_labels = [f"{stock}\\n{str(desc)[:20]}..." if pd.notna(desc) else
 ↪f"{stock}\\nN/A"
              for stock, desc in top15_qty.index]
axes[0, 1].barh(range(len(top15_qty)), top15_qty['TotalQuantitySold'],
 ↪color='skyblue')
axes[0, 1].set_yticks(range(len(top15_qty)))
axes[0, 1].set_yticklabels(qty_labels, fontsize=8)
axes[0, 1].set_title('Top 15 Products by Quantity Sold', fontweight='bold')
axes[0, 1].set_xlabel('Quantity Sold')
axes[0, 1].invert_yaxis()

# 3. Pareto Chart
top100 = product_net.head(100)
axes[1, 0].bar(range(len(top100)), top100['NetRevenue'], color='steelblue',
 ↪alpha=0.7)
ax2 = axes[1, 0].twinx()
ax2.plot(range(len(top100)), top100['CumulativePct'], color='red', marker='o',
 ↪linewidth=2)
```

```
ax2.axhline(80, color='red', linestyle='--', alpha=0.5, label='80% threshold')
axes[1, 0].set_title('Pareto Chart (Top 100 Products)', fontweight='bold')
axes[1, 0].set_xlabel('Product Rank')
axes[1, 0].set_ylabel('Net Revenue (£)', color='steelblue')
ax2.set_ylabel('Cumulative %', color='red')
ax2.legend()

# 4. Revenue Distribution (log scale)
axes[1, 1].hist(product_net[product_net['NetRevenue'] > 0]['NetRevenue'],
                bins=50, color='lightcoral', edgecolor='black')
axes[1, 1].set_title('Product Revenue Distribution (Net)', fontweight='bold')
axes[1, 1].set_xlabel('Net Revenue (£)')
axes[1, 1].set_ylabel('Number of Products')
axes[1, 1].set_yscale('log')
axes[1, 1].set_xscale('log')
axes[1, 1].grid(alpha=0.3)

plt.tight_layout()
plt.show()
```

```
================================================================================
PRODUCT ANALYSIS (NET SALES)
================================================================================
Total unique products: 4,154
\nTop 20 Products by Net Revenue:
 1. 22423    REGENCY CAKESTAND 3 TIER                        | £174,156.54 |
13,851 units | 1988.0 orders
 2. 23843    PAPER CRAFT , LITTLE BIRDIE                     | £168,469.60 |
80,995 units |  1.0 orders
 3. 85123A   WHITE HANGING HEART T-LIGHT HOLDER              | £104,284.24 |
37,580 units | 2189.0 orders
 4. 47566    PARTY BUNTING                                   | £ 99,445.23 |
18,283 units | 1685.0 orders
 5. 85099B   JUMBO BAG RED RETROSPOT                         | £ 94,159.81 |
48,371 units | 2089.0 orders
 6. 23166    MEDIUM CERAMIC TOP STORAGE JAR                  | £ 81,700.92 |
78,033 units | 247.0 orders
 7. 23084    RABBIT NIGHT LIGHT                              | £ 66,870.03 |
30,739 units | 994.0 orders
 8. 22086    PAPER CHAIN KIT 50'S CHRISTMAS                  | £ 64,875.59 |
19,329 units | 1160.0 orders
 9. 84879    ASSORTED COLOUR BIRD ORNAMENT                   | £ 58,927.62 |
36,362 units | 1455.0 orders
10. 79321    CHILLI LIGHTS                                   | £ 54,096.36 |
10,302 units | 661.0 orders
11. 23298    SPOTTY BUNTING                                  | £ 42,498.63 |
8,317 units | 1139.0 orders
12. 22386    JUMBO BAG PINK POLKADOT                         | £ 42,401.01 |
```

```
21,448 units | 1218.0 orders
13. 21137    BLACK RECORD COVER FRAME                    | £ 40,633.38 |
11,651 units | 375.0 orders
14. 22502    PICNIC BASKET WICKER 60 PIECES              | £ 39,619.50 |
61 units |  2.0 orders
15. 23284    DOORMAT KEEP CALM AND COME IN               | £ 38,133.64 |
5,487 units | 728.0 orders
16. 22720    SET OF 3 CAKE TINS PANTRY DESIGN            | £ 38,108.89 |
7,483 units | 1385.0 orders
17. 22960    JAM MAKING SET WITH JARS                    | £ 37,082.13 |
8,695 units | 1132.0 orders
18. 82484    WOOD BLACK BOARD ANT WHITE FINISH           | £ 35,966.92 |
6,012 units | 685.0 orders
19. 20725    LUNCH BAG RED RETROSPOT                     | £ 35,572.36 |
19,232 units | 1564.0 orders
20. 22197    POPCORN HOLDER                              | £ 34,288.67 |
36,749 units | 803.0 orders
\n================================================================================
==
PARETO ANALYSIS (80/20 Rule)
================================================================================
Products contributing to 80% of revenue: 859 (20.7%)
Revenue from these products: £8,210,992.85
```

1.8.1 Nhận xét về Product Analysis:

Quan sát chính: - **Pareto Principle (80/20)**: Một số lượng nhỏ sản phẩm (khoảng 20%) đóng góp 80% doanh thu, phù hợp với nguyên lý Pareto cổ điển. - **Top Products**: Các sản phẩm như "REGENCY CAKESTAND 3 TIER", "WHITE HANGING HEART T-LIGHT HOLDER" dẫn đầu về doanh thu. - **Long Tail**: Có rất nhiều sản phẩm có doanh thu thấp (long tail distribution), tạo ra thách thức trong quản lý hàng tồn kho. - **Quantity vs Revenue**: Sản phẩm có số lượng bán cao nhất không nhất thiết là sản phẩm có doanh thu cao nhất, cho thấy sự khác biệt về giá bán.

Hành động khuyến nghị: - **Focus Strategy**: Tập trung nguồn lực vào top 20% sản phẩm đóng góp 80% doanh thu. - **Inventory Optimization**: Xem xét giảm hoặc loại bỏ các sản phẩm có doanh thu thấp để tối ưu chi phí tồn kho. - **Bundle Products**: Kết hợp sản phẩm bán chạy với sản phẩm ít bán hơn để tăng doanh số cho long tail. - **Price Strategy**: Phân tích mối quan hệ giá-lượng để tối ưu hóa chiến lược định giá.

1.9 9. RFM Analysis (Customer Segmentation)

[ ]:
```
# ================================================================================
# P2-7: RFM ANALYSIS
# ================================================================================

# Lọc data có CustomerID
rfm_base = df_sales[df_sales['CustomerID'].notna()].copy()
print(f"Customers with valid ID: {rfm_base['CustomerID'].nunique():,}")

# Snapshot date (1 ngày sau giao dịch cuối)
snapshot_date = rfm_base['InvoiceDate'].max() + pd.Timedelta(days=1)
print(f"Snapshot date: {snapshot_date}")

# Tính RFM metrics
rfm = rfm_base.groupby('CustomerID').agg({
    'InvoiceDate': lambda x: (snapshot_date - x.max()).days,  # Recency
    'InvoiceNo': 'nunique',                                    # Frequency
    'TotalPrice': 'sum'                                        # Monetary
}).rename(columns={
    'InvoiceDate': 'Recency',
    'InvoiceNo': 'Frequency',
    'TotalPrice': 'Monetary'
})

print("\\n" + "="*80)
print("RFM METRICS SUMMARY")
print("="*80)
print(rfm.describe())

# Tạo RFM scores (quartiles)
```

```
rfm['R_Score'] = pd.qcut(-rfm['Recency'], 4, labels=[4, 3, 2, 1])  # Âm vì␣
 ↪recency càng nhỏ càng tốt
rfm['F_Score'] = pd.qcut(rfm['Frequency'].rank(method='first'), 4, labels=[1,␣
 ↪2, 3, 4])
rfm['M_Score'] = pd.qcut(rfm['Monetary'].rank(method='first'), 4, labels=[1, 2,␣
 ↪3, 4])

# Tạo RFM segment
rfm['RFM_Score'] = rfm['R_Score'].astype(str) + rfm['F_Score'].astype(str) +␣
 ↪rfm['M_Score'].astype(str)
rfm['RFM_Total'] = rfm['R_Score'].astype(int) + rfm['F_Score'].astype(int) +␣
 ↪rfm['M_Score'].astype(int)

# Phân loại khách hàng
def segment_customer(row):
    r, f, m = int(row['R_Score']), int(row['F_Score']), int(row['M_Score'])
    if r >= 4 and f >= 4 and m >= 4:
        return 'Khách hàng VIP'
    elif r >= 3 and f >= 3:
        return 'Khách hàng trung thành'
    elif r >= 4:
        return 'Khách hàng tiềm năng'
    elif f >= 4:
        return 'Chi tiêu lớn'
    elif r <= 2 and f >= 3:
        return 'Nguy cơ rời bỏ'
    elif r <= 2 and f <= 2:
        return 'Đã rời bỏ'
    else:
        return 'Khác (Others)'

rfm['Segment'] = rfm.apply(segment_customer, axis=1)

# Thống kê segments
segment_stats = rfm.groupby('Segment').agg({
    'Recency': 'mean',
    'Frequency': 'mean',
    'Monetary': 'sum',
    'R_Score': 'count'
}).rename(columns={'R_Score': 'Count'}).sort_values('Monetary', ascending=False)

segment_stats['AvgMonetary'] = segment_stats['Monetary'] /␣
 ↪segment_stats['Count']
segment_stats['Pct'] = segment_stats['Count'] / segment_stats['Count'].sum() *␣
 ↪100
```

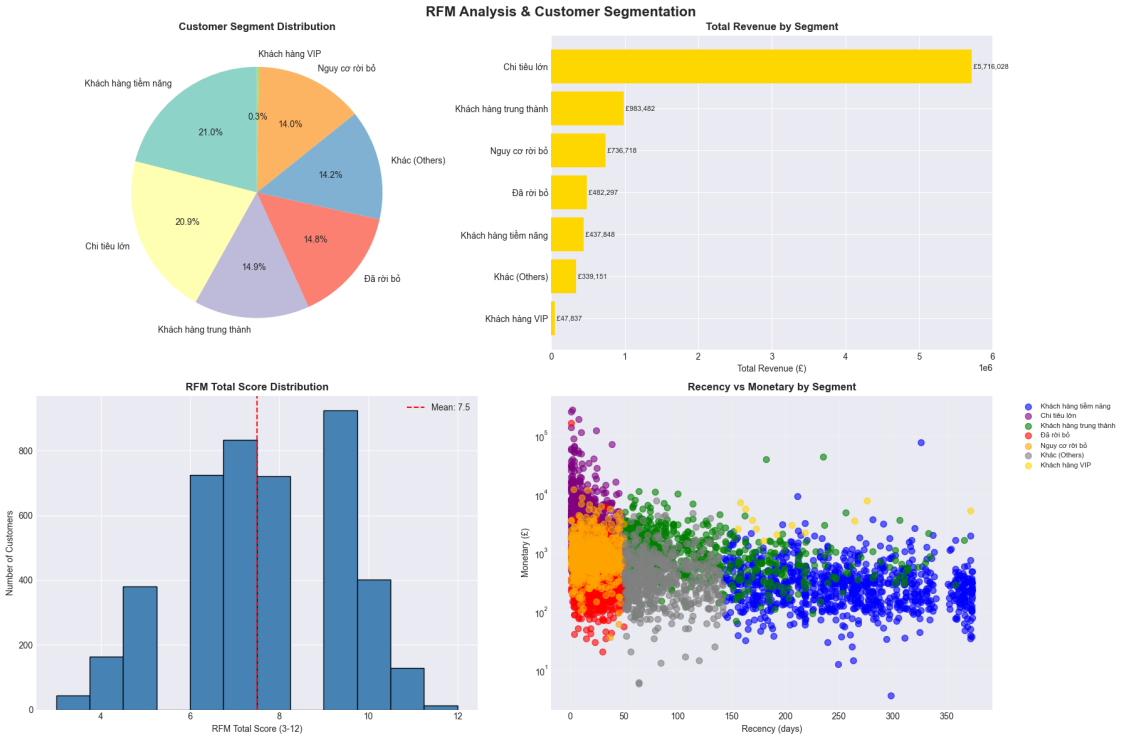
```
print("\\n" + "="*80)
print("CUSTOMER SEGMENTS")
print("="*80)
print(segment_stats.to_string())

# Top 20 customers
print("\\n" + "="*80)
print("TOP 20 CUSTOMERS (by Monetary)")
print("="*80)
top20 = rfm.nlargest(20, 'Monetary')[['Recency', 'Frequency', 'Monetary',␣
 ↪'Segment']]
for idx, (cust_id, row) in enumerate(top20.iterrows(), 1):
    print(f"{idx:2}. Customer {int(cust_id):5} | R:{row['Recency']:3.0f}d F:
 ↪{row['Frequency']:3.0f} M:£{row['Monetary']:>10,.2f} | {row['Segment']}")

# Visualizations
fig, axes = plt.subplots(2, 2, figsize=(18, 12))
fig.suptitle('RFM Analysis & Customer Segmentation', fontsize=16,␣
 ↪fontweight='bold')

# 1. Segment Distribution
segment_counts = rfm['Segment'].value_counts()
colors = plt.cm.Set3(range(len(segment_counts)))
axes[0, 0].pie(segment_counts.values, labels=segment_counts.index, autopct='%1.
 ↪1f%%',
               colors=colors, startangle=90)
axes[0, 0].set_title('Customer Segment Distribution', fontweight='bold')

# 2. Segment Revenue
segment_revenue = segment_stats.sort_values('Monetary', ascending=True)
axes[0, 1].barh(range(len(segment_revenue)), segment_revenue['Monetary'],␣
 ↪color='gold')
axes[0, 1].set_yticks(range(len(segment_revenue)))
axes[0, 1].set_yticklabels(segment_revenue.index)
axes[0, 1].set_title('Total Revenue by Segment', fontweight='bold')
axes[0, 1].set_xlabel('Total Revenue (£)')
for i, v in enumerate(segment_revenue['Monetary']):
    axes[0, 1].text(v, i, f' £{v:,.0f}', va='center', fontsize=8)

# 3. RFM Score Distribution
axes[1, 0].hist(rfm['RFM_Total'], bins=12, color='steelblue', edgecolor='black')
axes[1, 0].set_title('RFM Total Score Distribution', fontweight='bold')
axes[1, 0].set_xlabel('RFM Total Score (3-12)')
axes[1, 0].set_ylabel('Number of Customers')
axes[1, 0].axvline(
    rfm['RFM_Total'].mean(),
    color='red',
```

```
    linestyle='--',
    label=f"Mean: {rfm['RFM_Total'].mean():.1f}"
)

axes[1, 0].legend()

# 4. Recency vs Monetary (scatter with segments)
segment_colors = {
    'Khách hàng VIP': 'gold',
    'Khách hàng trung thành': 'green',
    'Khách hàng tiềm năng': 'blue',
    'Chi tiêu lớn': 'purple',
    'Nguy cơ rời bỏ': 'orange',
    'Đã rời bỏ': 'red',
    'Khác (Others)': 'gray'
}
for segment in rfm['Segment'].unique():
    segment_data = rfm[rfm['Segment'] == segment]
    axes[1, 1].scatter(segment_data['Recency'], segment_data['Monetary'],
                       label=segment, alpha=0.6, s=50, color=segment_colors.
 ↪get(segment, 'gray'))
axes[1, 1].set_title('Recency vs Monetary by Segment', fontweight='bold')
axes[1, 1].set_xlabel('Recency (days)')
axes[1, 1].set_ylabel('Monetary (£)')
axes[1, 1].set_yscale('log')
axes[1, 1].legend(bbox_to_anchor=(1.05, 1), loc='upper left', fontsize=8)
axes[1, 1].grid(alpha=0.3)

plt.tight_layout()
plt.show()
```

```
Customers with valid ID: 4,334
Snapshot date: 2011-12-10 12:50:00
\n================================================================================
==
RFM METRICS SUMMARY
================================================================================
       Recency  Frequency  Monetary
count  4334.00    4334.00   4334.00
mean     92.70       4.25   2017.39
std     100.18       7.64   8919.81
min       1.00       1.00      3.75
25%      18.00       1.00    304.31
50%      51.00       2.00    663.71
75%     143.00       5.00   1631.62
max     374.00     206.00 279138.02
\n================================================================================
==
```

```
CUSTOMER SEGMENTS
================================================================================
                        Recency  Frequency   Monetary  Count  AvgMonetary   Pct
Segment
Chi tiêu lớn              15.05      11.96 5716027.96    906      6309.08 20.90
Khách hàng trung thành   115.10       4.05  983481.56    644      1527.15 14.86
Nguy cơ rời bỏ            21.03       3.32  736718.46    605      1217.72 13.96
Đã rời bỏ                 24.52       1.44  482297.37    641       752.41 14.79
Khách hàng tiềm năng     254.85       1.21  437848.47    910       481.15 21.00
Khác (Others)             83.00       1.31  339151.14    616       550.57 14.21
Khách hàng VIP           210.58      10.83   47836.83     12      3986.40  0.28
\n================================================================================
==
TOP 20 CUSTOMERS (by Monetary)
================================================================================
 1. Customer 14646 | R:  2d F: 72 M:£279,138.02 | Chi tiêu lớn
 2. Customer 18102 | R:  1d F: 60 M:£259,657.30 | Chi tiêu lớn
 3. Customer 17450 | R:  8d F: 46 M:£194,390.79 | Chi tiêu lớn
 4. Customer 16446 | R:  1d F:  2 M:£168,472.50 | Đã rời bỏ
 5. Customer 14911 | R:  1d F:199 M:£140,336.83 | Chi tiêu lớn
 6. Customer 12415 | R: 24d F: 20 M:£124,564.53 | Chi tiêu lớn
 7. Customer 14156 | R: 10d F: 55 M:£117,210.08 | Chi tiêu lớn
 8. Customer 17511 | R:  3d F: 31 M:£ 91,062.38 | Chi tiêu lớn
 9. Customer 12346 | R:326d F:  1 M:£ 77,183.60 | Khách hàng tiềm năng
10. Customer 16029 | R: 39d F: 62 M:£ 72,708.09 | Chi tiêu lớn
11. Customer 16684 | R:  4d F: 28 M:£ 66,653.56 | Chi tiêu lớn
12. Customer 13694 | R:  4d F: 50 M:£ 65,039.62 | Chi tiêu lớn
13. Customer 15311 | R:  1d F: 91 M:£ 60,632.75 | Chi tiêu lớn
14. Customer 13089 | R:  3d F: 97 M:£ 58,762.08 | Chi tiêu lớn
15. Customer 17949 | R:  1d F: 44 M:£ 58,030.48 | Chi tiêu lớn
16. Customer 15769 | R:  7d F: 26 M:£ 56,252.72 | Chi tiêu lớn
17. Customer 15061 | R:  4d F: 48 M:£ 54,534.14 | Chi tiêu lớn
18. Customer 14096 | R:  4d F: 17 M:£ 53,258.43 | Chi tiêu lớn
19. Customer 14298 | R:  8d F: 44 M:£ 51,527.30 | Chi tiêu lớn
20. Customer 14088 | R: 10d F: 13 M:£ 50,491.81 | Chi tiêu lớn
\n================================================================================
==
RFM METRICS SUMMARY
================================================================================
       Recency  Frequency  Monetary
count  4334.00    4334.00   4334.00
mean     92.70       4.25   2017.39
std     100.18       7.64   8919.81
min       1.00       1.00      3.75
25%      18.00       1.00    304.31
50%      51.00       2.00    663.71
75%     143.00       5.00   1631.62
max     374.00     206.00 279138.02
```

```
\n================================================================================
==
CUSTOMER SEGMENTS
================================================================================
                        Recency  Frequency   Monetary  Count  AvgMonetary   Pct
Segment
Chi tiêu lớn              15.05      11.96 5716027.96    906      6309.08 20.90
Khách hàng trung thành   115.10       4.05  983481.56    644      1527.15 14.86
Nguy cơ rời bỏ            21.03       3.32  736718.46    605      1217.72 13.96
Đã rời bỏ                 24.52       1.44  482297.37    641       752.41 14.79
Khách hàng tiềm năng     254.85       1.21  437848.47    910       481.15 21.00
Khác (Others)             83.00       1.31  339151.14    616       550.57 14.21
Khách hàng VIP           210.58      10.83   47836.83     12      3986.40  0.28
\n================================================================================
==
TOP 20 CUSTOMERS (by Monetary)
================================================================================
 1. Customer 14646 | R:  2d F: 72 M:£279,138.02 | Chi tiêu lớn
 2. Customer 18102 | R:  1d F: 60 M:£259,657.30 | Chi tiêu lớn
 3. Customer 17450 | R:  8d F: 46 M:£194,390.79 | Chi tiêu lớn
 4. Customer 16446 | R:  1d F:  2 M:£168,472.50 | Đã rời bỏ
 5. Customer 14911 | R:  1d F:199 M:£140,336.83 | Chi tiêu lớn
 6. Customer 12415 | R: 24d F: 20 M:£124,564.53 | Chi tiêu lớn
 7. Customer 14156 | R: 10d F: 55 M:£117,210.08 | Chi tiêu lớn
 8. Customer 17511 | R:  3d F: 31 M:£ 91,062.38 | Chi tiêu lớn
 9. Customer 12346 | R:326d F:  1 M:£ 77,183.60 | Khách hàng tiềm năng
10. Customer 16029 | R: 39d F: 62 M:£ 72,708.09 | Chi tiêu lớn
11. Customer 16684 | R:  4d F: 28 M:£ 66,653.56 | Chi tiêu lớn
12. Customer 13694 | R:  4d F: 50 M:£ 65,039.62 | Chi tiêu lớn
13. Customer 15311 | R:  1d F: 91 M:£ 60,632.75 | Chi tiêu lớn
14. Customer 13089 | R:  3d F: 97 M:£ 58,762.08 | Chi tiêu lớn
15. Customer 17949 | R:  1d F: 44 M:£ 58,030.48 | Chi tiêu lớn
16. Customer 15769 | R:  7d F: 26 M:£ 56,252.72 | Chi tiêu lớn
17. Customer 15061 | R:  4d F: 48 M:£ 54,534.14 | Chi tiêu lớn
18. Customer 14096 | R:  4d F: 17 M:£ 53,258.43 | Chi tiêu lớn
19. Customer 14298 | R:  8d F: 44 M:£ 51,527.30 | Chi tiêu lớn
20. Customer 14088 | R: 10d F: 13 M:£ 50,491.81 | Chi tiêu lớn
```



1.9.1 Nhận xét về RFM Analysis:

Quan sát chính: - **Customer Segmentation**: Khách hàng được phân loại thành 7 nhóm chính, mỗi nhóm có đặc điểm và giá trị khác nhau. - **High-Value Segments**: - **Khách hàng VIP**: Nhóm có giá trị cao nhất, mua gần đây, thường xuyên và chi tiêu nhiều. - **Khách hàng trung thành**: Đóng góp doanh thu ổn định, cần chăm sóc để duy trì. - **At-Risk Segments**: - **Nguy cơ rời bỏ**: Từng mua nhiều nhưng đã lâu không quay lại, cần chiến dịch win-back. - **Đã rời bỏ**: Không hoạt động lâu, chi phí re-acquisition cao. - **Opportunity Segments**: - **Khách hàng tiềm năng**: Mua gần đây, có tiềm năng trở thành loyal nếu được nurture đúng cách. - **Chi tiêu lớn**: Mua ít nhưng mỗi lần chi tiêu lớn.

Phân bố Recency vs Monetary: Biểu đồ scatter cho thấy mối quan hệ giữa độ gần đây và giá trị chi tiêu, giúp xác định nhóm ưu tiên.

Hành động khuyến nghị: 1. **Khách hàng VIP & Trung thành**: - Chương trình VIP với ưu đãi đặc biệt, early access sản phẩm mới. - Personalized communication và exclusive offers.

2. **Khách hàng tiềm năng**:
   - Gửi email marketing với sản phẩm liên quan.
   - Chương trình loyalty để khuyến khích mua lại.
3. **Nguy cơ rời bỏ**:
   - Win-back campaigns với discount codes.
   - Survey để hiểu lý do không quay lại.
4. **Chi tiêu lớn**:
   - Cross-sell và up-sell sản phẩm cao cấp.

- Premium customer service.

5. **Đã rời bỏ**:
   - Đánh giá ROI của re-acquisition trước khi đầu tư.
   - Có thể loại khỏi danh sách marketing để giảm chi phí.