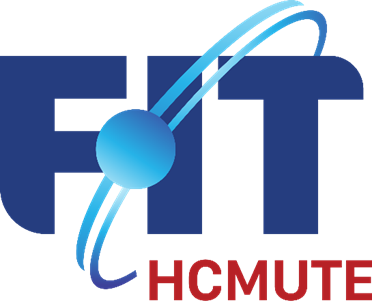
**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**-----🙞🙞🙞🙞🙞-----**



**ĐỒ ÁN MÔN HỌC**

**MÔN:** Phân tích dữ liệu

**MÃ MÔN HỌC:** DAAN436277

**PHÂN TÍCH XU HƯỚNG VÀ HÀNH VI TỪ DỮ LIỆU GIAO DỊCH THƯƠNG MẠI ĐIỆN TỬ PAKISTAN**

**GVHD: Nguyễn Văn Thành**

**Nhóm sinh viên thực hiện:** Nhóm 15

**Học kỳ:** II

**Năm học:** 2024 – 2025

*Thành phố Hồ Chí Minh, tháng 5 năm 2025*

**PHÂN CÔNG NHIỆM VỤ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nhiệm vụ | Trần Bảo Việt | Lê Hồ Quốc Huy | Lê Quỳnh Nhựt Vinh | Nguyễn Thiên Khang |
| Tìm hiểu đề tài | X | X | X |  |
| Tìm kiếm dữ liệu | X | X | X |  |
| Tiền xử lý dữ liệu | X | X | X |  |
| EDA: Phân tích Doanh thi và Số lượng đơn hàng theo thời gian | X |  |  |  |
| EDA: Phân tích Danh mục sản phẩm theo số lượng đơn hàng | X |  |  |  |
| EDA: Phân tích Danh mục sản phẩm theo doanh thu và chiết khấu | X |  |  |  |
| EDA: Phân tích Phương thức thanh toán theo tần suất sử dụng | X |  |  |  |
| EDA: Phân tích Phương thức thanh toán theo giá trị đơn hàng | X |  |  |  |
| EDA: Phân tích Trạng thái đơn hàng theo tần suất | X |  |  |  |
| EDA: Mối quan hệ giữa phương thức thanh toán, giá trị đơn hàng và tình trạng đơn hàng | X |  |  |  |
| EDA: Phân tích Tổng quan khách hàng về mức độ mua sắm |  |  | X |  |
| EDA: Phân tích hành vi khách hàng theo từng danh mục |  |  | X |  |
| EDA: Phân tích theo tỷ lệ chiết khấu và lượng đặt hàng |  | X |  |  |
| EDA: Phân tích theo tỷ lệ thành công các đơn hàng |  | X |  |  |
| Xây dựng mô hình  dự đoán: Mô Hình Dự Báo Doanh Số |  | X |  |  |
| Viết báo cáo |  |  | X |  |

**NHẬN XÉT CỦA GV**

Tp. Hồ Chí Minh, tháng 05 năm 2025

Giảng viên chấm điểm

**MỤC LỤC**

[**I: GIỚI THIỆU CHUNG** 1](#_Toc198737232)

[1.1. Bối cảnh chọn đề tài: 1](#_Toc198737233)

[1.2. Lý do chọn đề tài : 1](#_Toc198737234)

[1.3. Mục tiêu báo cáo 2](#_Toc198737235)

[1.4. Tìm kiếm và mô tả dữ liệu 2](#_Toc198737236)

[Mô tả dữ liệu: 2](#_Toc198737237)

[Bảng Mô Tả Thuộc Tính Dataset Thương Mại Điện Tử Pakistan 2](#_Toc198737238)

[**II: PHÂN TÍCH DỮ LIỆU** 4](#_Toc198737239)

[1. Tiền xử lý dữ liệu : 4](#_Toc198737240)

[1.1. Kiểm tra các giá trị còn thiếu 4](#_Toc198737241)

[1.2. Trực quan hóa các giá trị bị thiếu 6](#_Toc198737242)

[1.3. Xử lí giá trị trùng lặp 6](#_Toc198737243)

[1.4. Làm sạch dữ liệu 8](#_Toc198737244)

[1.5 Chuyển đổi kiểu dữ liệu 13](#_Toc198737245)

[1.6 Kiểm tra và xử lý các hàng bị trùng lặp 16](#_Toc198737246)

[1.7 Xử lí giá trị ngoại lai (Outliers) 17](#_Toc198737247)

[1.8 Tạo các đặc trưng (features) mới cho dữ liệu 19](#_Toc198737248)

[1.9 Mức độ tương quan của các biến số trong dữ liệu 20](#_Toc198737249)

[2. Nhập dữ liệu và khám phá sơ bộ 21](#_Toc198737250)

[2.1.Xu hướng theo Thời gian 21](#_Toc198737251)

[2.2.Xu hướng theo danh mục sản phẩm 22](#_Toc198737252)

[2.3.Phân tích phương thức thanh toán và đơn hàng 25](#_Toc198737253)

[2.4.Phân tích trạng thái đơn hàng 29](#_Toc198737254)

[2.5.Phân tích khách hàng 33](#_Toc198737255)

[2.6.Phân thích chiết khấu và lợi nhuân 39](#_Toc198737256)

[3. Xây dựng mô hình dự đoán 54](#_Toc198737257)

[4. Kết quả và đánh giá 71](#_Toc198737258)

[Mục tiêu và Phương pháp 71](#_Toc198737259)

[Đề xuất cải thiện mô hình hiện tại và mở rộng phân tích 73](#_Toc198737260)

[**III: KẾT LUẬN** 75](#_Toc198737261)

[Kết luận : 75](#_Toc198737262)

[2. Hướng mở rộng : 75](#_Toc198737263)

[**IV: PHỤ LỤC** 77](#_Toc198737264)

# **I: GIỚI THIỆU CHUNG**

## 1.1. Bối cảnh chọn đề tài:

Với nhiệm vụ là một nhóm phân tích dữ liệu, chúng em chọn khám phá và phân tích một tập dữ liệu về giao dịch thương mại điện tử (TMĐT) tại Pakistan. Mục tiêu là rút ra những hiểu biết sâu sắc và có giá trị, nhằm phục vụ cho việc định hình các quyết định chiến lược cho các doanh nghiệp quan tâm đến thị trường này. Tập dữ liệu này phản ánh các hoạt động mua sắm trực tuyến tại Pakistan, bao gồm thông tin về các giao dịch, sản phẩm, và có thể là cả hành vi của người tiêu dùng trong một khoảng thời gian nhất định. Dù không đại diện cho toàn bộ thị trường TMĐT Pakistan, tập dữ liệu này cung cấp một lát cắt quan trọng, mang đến cơ hội quý báu để thực hành các kỹ năng phân tích dữ liệu và đưa ra những nhận định, đề xuất chiến lược có tính ứng dụng thực tế cho bối cảnh TMĐT Pakistan.

## 1.2. Lý do chọn đề tài :

Chúng em lựa chọn đề tài này vì nó cho phép nhóm vận dụng kiến thức phân tích dữ liệu vào một thị trường TMĐT cụ thể và đang phát triển như Pakistan. Việc này mô phỏng tình huống thực tế trong lĩnh vực tư vấn, nơi các nhà phân tích cần khai thác tối đa thông tin từ dữ liệu chuyên biệt để đưa ra các khuyến nghị phù hợp với đặc thù địa phương. Dữ liệu giao dịch TMĐT tại Pakistan, với các thông tin về danh mục sản phẩm, giá trị giao dịch, số lượng hàng hóa, tổng giá trị đơn, các khoản giảm giá, phương thức thanh toán được sử dụng, cũng như thông tin về thời gian (năm, tháng) và mã khách hàng, chứa đựng nhiều chiều thông tin quan trọng. Những dữ liệu này cho phép phân tích sâu về các loại sản phẩm được ưa chuộng, hành vi chi tiêu của khách hàng, hiệu quả của các chương trình khuyến mãi, các kênh thanh toán phổ biến, cũng như xu hướng mua sắm theo thời gian và đặc điểm của các nhóm khách hàng khác nhau, từ đó xác định các yếu tố then chốt ảnh hưởng đến xu hướng tiêu dùng và doanh thu.

Qua dự án này, nhóm sẽ có cơ hội áp dụng các kỹ thuật phân tích, trực quan hóa dữ liệu và "kể chuyện bằng dữ liệu" (data storytelling) để xây dựng một báo cáo phân tích xu hướng và hành vi người dùng mang tính thực tiễn cao, tập trung vào thị trường Pakistan. Dự án sẽ giúp nhóm thể hiện năng lực khai thác dữ liệu một cách hiệu quả, đồng thời truyền tải thông tin và các phát hiện quan trọng đến những người ra quyết định một cách trực quan và dễ hiểu.

## 1.3. Mục tiêu báo cáo

1. Phân tích các xu hướng và hành vi giao dịch trong lĩnh vực thương mại điện tử ở Pakistan .
2. Khám phá các mối quan hệ giữa các biến số trong dữ liệu như mối liên hệ giữa thành phố, loại thanh toán, trạng thái đơn hàng và giá trị giao dịch.
3. Phân tích các yếu tố ảnh hưởng đến tỷ lệ hủy đơn và thời gian giao hàng, từ đó phát hiện những điểm nghẽn trong chuỗi cung ứng.
4. Xây dựng mô hình dự đoán xu hướng doanh số đặt hàng thương mại điện tử .
5. Đưa ra các đề xuất chiến lược dựa trên dữ liệu, nhằm giúp doanh nghiệp (giả định) tối ưu hoạt động kinh doanh thương mại điện tử .

## 1.4. Tìm kiếm và mô tả dữ liệu

**Nguồn dữ liệu** : Tập dữ liệu được nhóm chúng em sử dụng được lấy từ nền tảng Kaggle, tại đường dẫn:

<https://www.kaggle.com/datasets/zusmani/pakistans-largest-ecommerce-dataset/data>

### Mô tả dữ liệu:

* Định dạng file: CSV (comma-separated value)
* Số lượng bản ghi (số dòng dữ liệu): 1048575 dòng
* Số lượng cột: 25

## Bảng Mô Tả Thuộc Tính Dataset Thương Mại Điện Tử Pakistan

1. item\_id: Mã định danh duy nhất cho từng sản phẩm.

2. status: Trạng thái của đơn hàng (ví dụ: đã giao, đã hủy, đang xử lý).

3. created\_at: Thời gian tạo đơn hàng.

4. sku: Mã SKU (Stock Keeping Unit) của sản phẩm.

5. price: Giá của sản phẩm.

6. qty\_ordered: Số lượng sản phẩm được đặt hàng.

7. grand\_total: Tổng giá trị đơn hàng (bao gồm thuế, phí vận chuyển, v.v.).

8. increment\_id: Mã định danh tăng dần của đơn hàng.

9. category\_name\_1: Tên danh mục sản phẩm.

10. sales\_commission\_code: Mã hoa hồng bán hàng.

11. discount\_amount: Số tiền giảm giá áp dụng cho đơn hàng.

12. payment\_method: Phương thức thanh toán (ví dụ: thẻ tín dụng, tiền mặt khi giao hàng).

13. Working Date: Ngày làm việc liên quan đến đơn hàng.

14. BI Status: Trạng thái BI (Business Intelligence) của đơn hàng.

15. MV: Giá trị MV (cần thêm thông tin để giải thích rõ hơn).

16. Year: Năm của giao dịch.

17. Month: Tháng của giao dịch.

18. Customer Since: Ngày khách hàng bắt đầu mua sắm.

19. M-Y: Tháng và năm của giao dịch (định dạng MM-YYYY).

20. FY: Năm tài chính (Fiscal Year).

21. Customer ID: Mã định danh khách hàng.

22. Unnamed: 21: Thuộc tính không rõ (cần thêm thông tin).

23. Unnamed: 22: Thuộc tính không rõ (cần thêm thông tin).

24. Unnamed: 23: Thuộc tính không rõ (cần thêm thông tin).

25. Unnamed: 24: Thuộc tính không rõ (cần thêm thông tin).

26. Unnamed: 25: Thuộc tính không rõ (cần thêm thông tin).

# **II: PHÂN TÍCH DỮ LIỆU**

## 1. Tiền xử lý dữ liệu :

### 1.1. Kiểm tra các giá trị còn thiếu

# Tạo bản sao của DataFrame

df\_missing = df.copy()

# Tính giá trị bị thiếu và tỉ lệ phần trăm của chúng

missing\_data = df\_missing.isnull().sum().sort\_values(ascending=False).reset\_index()

missing\_data.columns = ['Columns Name', 'Missing Values']

missing\_data['Percentage Missing Values'] = (missing\_data['Missing Values'] / len(df\_missing)) \* 100

missing\_data['Percentage Missing Values'] = missing\_data['Percentage Missing Values'].apply(lambda x: f"{x:.0f}%")

missing\_data

A screenshot of a graph

AI-generated content may be incorrect.

### 1.2. Trực quan hóa các giá trị bị thiếu

# Lọc các cột có giá trị bị thiếu > 0 để trực quan hóa

missing\_data\_filtered = missing\_data[missing\_data['Missing Values'] > 0]

# Trực quan hóa

plt.figure(figsize=(10, 6))

plt.bar(missing\_data\_filtered['Columns Name'], missing\_data\_filtered['Percentage Missing Values'], color='blue')

plt.title('Percentage of Missing Values by Column', fontsize=16)

plt.xlabel('Columns', fontsize=14)

plt.ylabel('Percentage Missing (%)', fontsize=14)

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

A graph of missing values

AI-generated content may be incorrect.

### 1.3. Xử lí giá trị trùng lặp

# Kiểm tra xem có hàng bị trùng lặp không

duplicates = df.duplicated()

# Đếm tổng số hàng bị trùng lặp

total\_duplicates = df.duplicated().sum()

print(f"Total number of duplicate rows in the data: {total\_duplicates}")

Total number of duplicate rows in the data: 464050

# DataFrame chứa các giá trị duy nhất từ các cột khác nhau

unique\_status = pd.DataFrame({'unique\_status': df['status'].unique()})

unique\_payment\_method = pd.DataFrame({'unique\_payment\_method': df['payment\_method'].unique()})

unique\_category\_name\_1 = pd.DataFrame({'unique\_category\_name\_1': df['category\_name\_1'].unique()})

unique\_bi\_status = pd.DataFrame({'unique\_BI\_Status': df['BI Status'].unique()})

# Nối các DataFrame lại với nhau theo chiều ngang và thay NaN bằng chuỗi rỗng

merged\_df = pd.concat([unique\_status, unique\_payment\_method, unique\_category\_name\_1, unique\_bi\_status], axis=1).fillna('')

# Hiển thị kết quả

merged\_df

A screenshot of a computer

AI-generated content may be incorrect.

### 1.4. Làm sạch dữ liệu

#### 1.4.1  Loại bỏ các cột không cần thiết

print('Shape Before Drop Columns')

df.shape

Shape Before Drop Columns

(1048575, 26)

# Xóa các cột không cần thiết

df.drop(['Unnamed: 21', 'Unnamed: 22', 'Unnamed: 23', 'Unnamed: 24', 'Unnamed: 25'], axis=1, inplace=True)

print("Unnecessary columns ('Unnamed: 21, to 'Unnamed: 25') have been successfully dropped from the dataset")

Unnecessary columns ('Unnamed: 21, to 'Unnamed: 25') have been successfully dropped from the dataset

# Xóa cột 'sales\_commission\_code' vì có hơn 50% giá trị bị thiếu, không phù hợp để điền dữ liệu do tính duy nhất của nó

df.drop('sales\_commission\_code', axis=1, inplace=True)

# Loại bỏ cột 'M-Y' khỏi bộ dữ liệu vì thông tin của nó đã được thể hiện trong hai cột riêng biệt 'Year' và 'Month'

df.drop('M-Y', axis=1, inplace=True)

print("Successfully dropped 'M-Y' & 'sales\_commission\_code' columns from the dataset")

Successfully dropped 'M-Y' & 'sales\_commission\_code' columns from the dataset

print('Shape After Drop Columns')

df.shape

Shape After Drop Columns

(1048575, 19)

#### 1.4.2. Xử lý các giá trị thiếu

print('Shape Before Handling Missing Values')

df.shape

Shape Before Handling Missing Values

(1048575, 19)

# Kiểm tra giá trị bị thiếu

percentage\_missing = (df.isnull().sum() / len(df)) \* 100

percentage\_missing = percentage\_missing.map("{:.0f}%".format)

percentage\_missing

A screenshot of a phone

AI-generated content may be incorrect.

# Dữ liệu có rất nhiều hàng chứa toàn bộ giá trị là NaN nên cần phải loại bỏ

df.tail(8)

A screenshot of a computer

AI-generated content may be incorrect.

# Xóa những hàng mà tất cả các giá trị đều là NaN

df.dropna(how='all', inplace=True)

print("Rows with all values NaN have been successfully dropped from the dataset")

Rows with all values NaN have been successfully dropped from the dataset

# Kkiểm tra lại các giá trị bị thiếu

df.isnull().sum()

A screenshot of a phone

AI-generated content may be incorrect.

# Điền giá trị bị thiếu trong các cột 'status', 'category\_name\_1', và 'Customer Since' bằng giá trị mode của từng cột

# vì các cột này chứa dữ liệu phân loại (categorical)

df['status'].fillna(df['status'].mode()[0], inplace=True)

df['category\_name\_1'].fillna(df['category\_name\_1'].mode()[0], inplace=True)

df['Customer Since'].fillna(df['Customer Since'].mode()[0], inplace=True)

# Xóa các hàng có giá trị bị thiếu ở cột 'sku' hoặc 'Customer ID'

# vì 'sku' và 'Customer ID' là duy nhất (unique)

df.dropna(subset=['sku', 'Customer ID'], inplace=True)

print("Missing values filled for 'status', 'category\_name\_1', and 'Customer Since'.")

print()

print("Rows with missing values in 'sku' or 'Customer ID' have been dropped.")

Missing values filled for 'status', 'category\_name\_1', and 'Customer Since'.

Rows with missing values in 'sku' or 'Customer ID' have been dropped.

# Xử lý thành công các giá trị bị thiếu

df.isnull().sum()

A screenshot of a phone

AI-generated content may be incorrect.

print('Shape After Handling Missing Values')

df.shape

Shape After Handling Missing Values

(584493, 19)

#### 1.4.3 Loại bỏ sự không nhất quán của dữ liệu

# Xóa 'FY' khỏi tên cột FY và thay bằng '20'

df['FY'] = df['FY'].str.replace('FY', '20', regex=True)

# Xóa dấu phẩy trong cột MV

df[' MV '] = df[' MV '].str.replace(',', '', regex=True)

# Xóa ký tự # và ! trong cột BI Status

df['BI Status'] = df['BI Status'].replace('#', '', regex=True).replace('!', '', regex=True)

print('remove inconsistency from column completed successfully')

remove inconsistency from column completed successfully

# Đổi tên các cột thành tên có ý nghĩa hơn

df.rename(columns={

    ' MV ': 'market\_value',

    'FY': 'fiscal\_year',

    'Year': 'year',

    'Month': 'month',

    'sku': 'stock\_keeping\_unit',

    'Customer Since': 'customer\_since',

    'Customer ID': 'customer\_id',

    'category\_name\_1': 'category\_name',

    'BI Status': 'bi\_status',

    'Working Date': 'working\_date'

}, inplace=True)

print("Rename columns successfully")

Rename columns successfully

# Dữ liệu của chúng ta có chứa \N nên thay thế '\N' bằng NaN và điền các hàng chứa giá trị NaN

df.replace('\\N', pd.NA, inplace=True)  # Dữ liệu chứa \N, nhưng cần viết là \\N để tránh lỗi

df['category\_name']= df['category\_name'].fillna(df['category\_name'].mode()[0]) # chỉ có hai cột chứa \N

df['status']= df['status'].fillna(df['status'].mode()[0])

print('Handle \\N Successfullyl')

Handle \N Successfullyl

### 1.5 Chuyển đổi kiểu dữ liệu

# Kiểm tra kiểu dữ liệu của các cột

df.dtypes

A screenshot of a computer

AI-generated content may be incorrect.

# Chuyển đổi kiểu datetime và xử lý giá trị thiếu nếu xuất hiện (errors='coerce')

# Chuyển thành kiểu dữ liệu Datetime

df['working\_date'] = pd.to\_datetime(df['created\_at'], errors='coerce')

df['created\_at'] = pd.to\_datetime(df['customer\_since'], errors='coerce')

df['customer\_since'] = pd.to\_datetime(df['customer\_since'], errors='coerce')

# Điền NaN nếu xuất hiện trong các cột datetime bằng giá trị mode tương ứng

date\_columns = ['working\_date', 'created\_at', 'customer\_since']  # Xác định các cột

for col in date\_columns:

    mode\_value = df[col].mode()[0]  # Lấy giá trị mode của cột

    df[col].fillna(mode\_value, inplace=True)

print("Datetime conversion and missing values handling completed successfully")

Datetime conversion and missing values handling completed successfully

# Chuyển đổi kiểu dữ liệu phân loại (Categorical)

# Chuyển các cột được chỉ định sang kiểu dữ liệu category

categorical\_columns = ['status', 'payment\_method', 'category\_name', 'bi\_status', 'stock\_keeping\_unit']

df[categorical\_columns] = df[categorical\_columns].astype('category')

print("category type conversion process completed successfully")

category type conversion process completed successfully

# Chuyển đổi kiểu số và xử lý giá trị thiếu nếu xuất hiện (errors='coerce')

# Chuyển 'increment\_id' sang kiểu số và ép các giá trị không hợp lệ thành NaN

# (Các giá trị không hợp lệ như chuỗi sẽ bị thay bằng NaN nhờ errors='coerce')

df['increment\_id'] = pd.to\_numeric(df['increment\_id'], errors='coerce')

# Xóa các hàng mà 'increment\_id' là NaN

df.dropna(subset=['increment\_id'], inplace=True)

# Chuyển cột sang kiểu số nguyên sau khi đã xóa NaN

df['increment\_id'] = df['increment\_id'].astype(int)

# Chuyển cột price sang kiểu int

df['price']= pd.to\_numeric(df['price'], errors='coerce')

df['price']= df['price'].fillna(df['price'].mean()).astype(int)

# Chuyển cột qty\_ordered sang kiểu int

df['qty\_ordered'] = pd.to\_numeric(df['qty\_ordered'], errors='coerce')

df['qty\_ordered']= df['qty\_ordered'].fillna(df['qty\_ordered'].median()).astype(int)

# Chuyển cột customer\_id sang kiểu int

df['customer\_id'] = pd.to\_numeric(df['customer\_id'], errors='coerce')

df.dropna(subset=['customer\_id'], inplace=True)

df['customer\_id'] = df['customer\_id'].astype(int)

# Chuyển cột grand\_total sang kiểu int

df['grand\_total']= pd.to\_numeric(df['grand\_total'], errors='coerce')

df['grand\_total']= df['grand\_total'].fillna(df['grand\_total'].median()).astype(int)

# Chuyển cột item\_id sang kiểu int

df['item\_id'] = pd.to\_numeric(df['item\_id'], errors='coerce')

df.dropna(subset=['item\_id'], inplace=True)

df['item\_id'] = df['item\_id'].astype(int)

# Chuyển cột market\_value sang kiểu int

df['market\_value'] = pd.to\_numeric(df['market\_value'], errors='coerce')

df['market\_value']= df['market\_value'].fillna(df['market\_value'].mean()).astype(int)

# Chuyển cột discount\_amount sang kiểu int

df['discount\_amount'] = pd.to\_numeric(df['discount\_amount'], errors='coerce')

df['discount\_amount']= df['discount\_amount'].fillna(df['discount\_amount'].median()).astype(int)

# Chuyển cột year sang kiểu int

df['year']= df['year'].astype(int)

# Chuyển cột month sang kiểu int

df['month']= df['month'].astype(int)

# Chuyển cột fiscal\_year sang kiểu int

df['fiscal\_year']= df['fiscal\_year'].astype(int)

print("int type conversion and handle missing values if appears process completed successfully")

int type conversion and handle missing values if appears process completed successfully

# Kiểm tra dữ liệu sau chuyển đổi

df.dtypes

A screenshot of a computer

AI-generated content may be incorrect.

### 1.6 Kiểm tra và xử lý các hàng bị trùng lặp

# Kiểm tra xem có hàng bị trùng lặp không

duplicates = df.duplicated()

# Đếm tổng số hàng bị trùng lặp

total\_duplicates = df.duplicated().sum()

print(f"Total number of duplicate rows in the data: {total\_duplicates}")

Total number of duplicate rows in the data: 0

### 1.7 Xử lí giá trị ngoại lai (Outliers)

# Đặt kích thước cho toàn bộ biểu đồ

plt.figure(figsize=(18, 12))

# Danh sách các cột sẽ vẽ biểu đồ

columns\_to\_plot = ['price', 'qty\_ordered', 'grand\_total', 'discount\_amount', 'market\_value']

# Lặp qua từng cột và tạo subplot

for i, column in enumerate(columns\_to\_plot, 1):

    plt.subplot(2, 3, i)  # Tạo subplot với 2 hàng và 3 cột

    sns.boxplot(data=df, y=column, flierprops={'marker': 'o', 'markersize': 8, 'markerfacecolor': 'red'})  # Biểu đồ hộp

    plt.title(f'Box Plot of {column}', fontsize=14, fontweight='bold')

    plt.ylabel(column, fontsize=12, fontweight='bold')

# Điều chỉnh bố cục để tránh chồng lấn

plt.tight\_layout()

# Hiển thị biểu đồ

plt.show()

A collage of graphs

AI-generated content may be incorrect.

#### 1.7.1 Tính số lượng outliers trong từng cột riêng biệt

# Hàm tìm outliers sử dụng phương pháp IQR

def find\_outliers\_iqr(data, column):

    Q1 = data[column].quantile(0.25)

    Q3 = data[column].quantile(0.75)

    IQR = Q3 - Q1

    lower\_bound = Q1 - 1.5 \* IQR

    upper\_bound = Q3 + 1.5 \* IQR

    outliers = data[(data[column] < lower\_bound) | (data[column] > upper\_bound)]

    return outliers

# Hàm tìm outliers sử dụng phương pháp Z-score

def find\_outliers\_zscore(data, column):

    z\_scores = stats.zscore(data[column])

    abs\_z\_scores = np.abs(z\_scores)

    outliers = data[abs\_z\_scores > 3]

    return outliers

# Tìm outliers bằng phương pháp IQR

iqr\_outliers = find\_outliers\_iqr(df, 'price')

print(f'IQR Method: Found {iqr\_outliers.shape[0]} outliers in "price" column\n')

# Tìm outliers bằng phương pháp Z-score

zscore\_outliers = find\_outliers\_zscore(df, 'price')

print(f'Z-score Method: Found {zscore\_outliers.shape[0]} outliers in "price" column')

IQR Method: Found 105683 outliers in "price" column

Z-score Method: Found 14503 outliers in "price" column

#### 1.7.2 Tính toán Outliners trông toàn bộ dữ liệu

# outliers sử dụng phương pháp z-score

z = np.abs(stats.zscore(df[['price', 'qty\_ordered', 'grand\_total', 'discount\_amount', 'market\_value']]))

outliers = (z > 3).sum()

print(f"There are total {outliers.sum()} outliers in 'price', 'qty\_ordered', 'grand\_total', 'discount\_amount' & 'market\_value' columns using Z-score method \n")

# outliers sử dụng phương pháp IQR

Q1 = df[['price', 'qty\_ordered', 'grand\_total', 'discount\_amount', 'market\_value']].quantile(0.25)

Q3 = df[['price', 'qty\_ordered', 'grand\_total', 'discount\_amount', 'market\_value']].quantile(0.75)

IQR = Q3 - Q1

outliers = (df[['price', 'qty\_ordered', 'grand\_total', 'discount\_amount', 'market\_value']] < (Q1 - 1.5 \* IQR)) | (df[['price', 'qty\_ordered', 'grand\_total', 'discount\_amount', 'market\_value']] > (Q3 + 1.5 \* IQR))

print(f"There are total {outliers.sum().sum()} outliers in 'price', 'qty\_ordered', 'grand\_total', 'discount\_amount' & 'market\_value' columns using IQR method")

There are total 32030 outliers in 'price', 'qty\_ordered', 'grand\_total', 'discount\_amount' & 'market\_value' columns using Z-score method

There are total 474045 outliers in 'price', 'qty\_ordered', 'grand\_total', 'discount\_amount' & 'market\_value' columns using IQR method

# Xem dữ liệu sau khi đã làm sạch

df.sample(10, random\_state=42)

A screenshot of a computer

AI-generated content may be incorrect.

### 1.8 Tạo các đặc trưng (features) mới cho dữ liệu

# Giá trị đơn hàng trên mỗi sản phẩm

df['price\_per\_unit'] = df['price'] / df['qty\_ordered']

# Tỷ lệ phần trăm chiết khấu

df['discount\_percentage'] = (df['discount\_amount'] / df['price']) \* 100

df['discount\_percentage'].replace([np.inf, -np.inf], np.nan, inplace=True)

df['discount\_percentage'].fillna(0, inplace=True)

# Thời gian khách hàng đã gắn bó (tính bằng ngày)

df['customer\_tenure\_days'] = (df['created\_at'] - df['customer\_since']).dt.days

# Tạo từ điển ánh xạ số tháng sang tên tháng

month\_name\_mapping = {

    1: 'January', 2: 'February', 3: 'March', 4: 'April', 5: 'May', 6: 'June',

    7: 'July', 8: 'August', 9: 'September', 10: 'October', 11: 'November', 12: 'December'

}

# Tạo cột mới 'month\_name' bằng cách ánh xạ từ cột 'month'

df['month\_name'] = df['month'].map(month\_name\_mapping)

print("4 New features created successfully")

4 New features created successfully

### 1.9 Mức độ tương quan của các biến số trong dữ liệu

# Chọn các cột để tính ma trận tương quan

columns\_for\_correlation = ['price', 'qty\_ordered', 'grand\_total', 'discount\_amount', 'market\_value', 'price\_per\_unit']

# Tính ma trận tương quan

correlation\_matrix = df[columns\_for\_correlation].corr()

# Đặt kích thước biểu đồ

plt.figure(figsize=(12, 8))

# Tạo heatmap với các tùy chỉnh

sns.heatmap(correlation\_matrix, annot=True, fmt=".2f", cmap="Set1", linewidths=0.5, linecolor='black', cbar\_kws={'label': 'Correlation Coefficient'})

# Thêm tiêu đề và nhãn

plt.title('Correlation Matrix Heatmap', fontsize=16, fontweight='bold')

plt.xlabel('Features', fontsize=14, fontweight='bold')

plt.ylabel('Features', fontsize=14, fontweight='bold')

# Hiển thị biểu đồ

plt.show()

A grid of squares with different colors

AI-generated content may be incorrect.

## 2. Nhập dữ liệu và khám phá sơ bộ

### 2.1.Xu hướng theo Thời gian

#### 2.1.1. Phân tích Doanh thi và Số lượng đơn hàng theo thời gian

**Câu hỏi: Doanh thu và số đơn hàng thay đổi như thế nào qua các tháng và các năm?**

# Chuyển đổi định dạng ngày tháng

df['created\_at'] = pd.to\_datetime(df['created\_at'])

# Nhóm dữ liệu theo năm và tên tháng

monthly\_trend = df.groupby(['year', 'month\_name']).agg({

    'item\_id': 'count',

    'grand\_total': 'sum'

}).reset\_index().rename(columns={'item\_id': 'order\_count'})

# Sắp xếp thứ tự tháng

month\_order = ['January', 'February', 'March', 'April', 'May', 'June',

               'July', 'August', 'September', 'October', 'November', 'December']

monthly\_trend['month\_name'] = pd.Categorical(monthly\_trend['month\_name'], categories=month\_order, ordered=True)

monthly\_trend = monthly\_trend.sort\_values(['year', 'month\_name'])

# Tạo biểu đồ

plt.figure(figsize=(16, 6))

# Biểu đồ số đơn hàng theo tháng

plt.subplot(1, 2, 1)

sns.lineplot(data=monthly\_trend, x='month\_name', y='order\_count', hue='year', marker='o')

plt.title('Số đơn hàng theo tháng')

plt.xlabel('Tháng')

plt.ylabel('Số đơn hàng')

plt.xticks(rotation=45)

# Biểu đồ doanh thu theo tháng

plt.subplot(1, 2, 2)

sns.lineplot(data=monthly\_trend, x='month\_name', y='grand\_total', hue='year', marker='o')

plt.title('Doanh thu theo tháng')

plt.xlabel('Tháng')

plt.ylabel('Doanh thu')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

A graph of a line and a line

AI-generated content may be incorrect.

Nhận xét:

**Xu hướng Số đơn hàng**

* Có sự tăng trưởng theo mùa, đặc biệt vào các tháng như Tháng 6, Tháng 11 và Tháng 12, thường là mùa mua sắm (giảm giá giữa năm và cuối năm).
* Một số tháng như Tháng 1 và Tháng 2 có đơn hàng thấp hơn, có thể do sau kỳ nghỉ Tết hoặc đầu năm tài chính.

**Xu hướng Doanh thu**

* Doanh thu cũng phản ánh xu hướng tương tự như số đơn hàng, với các đỉnh vào cuối năm và giữa năm.
* Tuy nhiên, một số năm có doanh thu cao hơn mặc dù số đơn hàng không tăng tương ứng — điều này cho thấy có thể là do khách hàng mua sản phẩm giá trị cao hơn.

### 2.2.Xu hướng theo danh mục sản phẩm

#### 2.2.1 Phân tích Danh mục sản phẩm theo số lượng đơn hàng

**Câu hỏi: Danh mục sản phẩm bán chạy nhất?**

# Nhóm theo category\_name và tính tổng số lượng đã đặt

category\_sales = df.groupby('category\_name')['qty\_ordered'].sum().reset\_index()

# Vẽ biểu đồ kết quả

plt.figure(figsize=(12, 8))

sns.barplot(x='qty\_ordered', y='category\_name', data=category\_sales, palette='viridis')

plt.title('Các danh mục bán chạy nhất', fontsize=16, fontweight='bold')

plt.xlabel('Số lượng đã đặt', fontsize=14)

plt.ylabel('Tên danh mục', fontsize=14)

plt.show()

A graph of different colored bars

AI-generated content may be incorrect.

**Nhận xét:**

* Các danh mục như "Mobiles & Tablets" và "Men's Fashion" cho thấy hiệu suất vượt trội, có khả năng là những động lực tăng trưởng doanh thu chính. Việc tập trung vào các mặt hàng công nghệ và thời trang nam dường như đang mang lại kết quả tốt.
* Vì vậy hãy tập trung kho hàng và quảng cáo cho nhóm sản phẩm top đầu.

**2.2.2 Phân tích Danh mục sản phẩm theo doanh thu và chiết khấu**

**Câu hỏi: Chiến dịch giảm giá nào hiệu quả nhất theo danh mục?**

# Nhóm theo danh mục sản phẩm

discount\_analysis = df.groupby('category\_name').agg({

    'grand\_total': 'sum',

    'discount\_amount': 'sum'

}).reset\_index()

# Chỉ giữ lại những danh mục có doanh thu đáng kể (lọc nhiễu)

discount\_analysis = discount\_analysis[discount\_analysis['grand\_total'] > 100000]

# Sắp xếp theo tổng doanh thu

discount\_analysis = discount\_analysis.sort\_values(by='grand\_total', ascending=False)

# Thiết lập kích thước và màu sắc

plt.figure(figsize=(14, 8))

bar\_width = 0.4

x = range(len(discount\_analysis))

# Vẽ cột doanh thu

plt.bar(x, discount\_analysis['grand\_total'], width=bar\_width, label='Doanh thu', color='skyblue')

# Vẽ cột chiết khấu (chồng lên)

plt.bar([i + bar\_width for i in x], discount\_analysis['discount\_amount'], width=bar\_width, label='Chiết khấu', color='orange')

# Gán nhãn

plt.xlabel('Danh mục sản phẩm')

plt.ylabel('Tổng giá trị (PKR)')

plt.title('Hiệu quả giảm giá theo danh mục sản phẩm')

plt.xticks([i + bar\_width / 2 for i in x], discount\_analysis['category\_name'], rotation=90)

plt.legend()

plt.tight\_layout()

plt.show()

A graph with blue and orange squares

AI-generated content may be incorrect.

**Nhận xét:**

Dựa trên số liệu về tổng doanh thu được tạo ra trong kỳ có áp dụng chiết khấu, danh mục **Mobiles & Tablets** thể hiện "hiệu quả" cao nhất về mặt tạo ra tổng doanh thu lớn nhất đi kèm với mức chiết khấu cao nhất. Các danh mục **Appliances và Entertainment** xếp sau.

Tuy nhiên, cần phân tích sâu hơn để đánh giá hiệu quả thực sự về mặt lợi nhuận hoặc hiệu quả sử dụng ngân sách chiết khấu.

### 2.3.Phân tích phương thức thanh toán và đơn hàng

#### 2.3.1 Phân tích Phương thức thanh toán theo tần suất sử dụng

**Câu hỏi: Phương thức thanh toán ưa thích nhất (phổ biến nhất) của khách hàng là gì?**

# Trực quan hóa tần suất các phương thức thanh toán

plt.figure(figsize=(12, 6))

sns.countplot(data=df, x='payment\_method', palette='viridis', order=df['payment\_method'].value\_counts().index)

plt.title('Tần suất các phương thức thanh toán', fontsize=16, fontweight='bold')

plt.xlabel('Phương thức thanh toán', fontsize=14)

plt.ylabel('Tần suất', fontsize=14)

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

A graph with different colored squares

AI-generated content may be incorrect.

Nhận xét:

* Dựa trên biểu đồ "**Tần suất các phương thức thanh toán**", phương thức thanh toán được ưa thích nhất (phổ biến nhất) của khách hàng là "**cod**" (Cash on Delivery - Thanh toán khi nhận hàng).
* **Sự thống trị của COD**: Biểu đồ cho thấy sự ưa chuộng áp đảo của phương thức thanh toán khi nhận hàng (COD). Điều này có thể phản ánh một số yếu tố như sự tin tưởng của khách hàng (họ muốn kiểm tra hàng trước khi trả tiền), thói quen mua sắm, hoặc hạn chế trong việc tiếp cận/sử dụng các phương thức thanh toán kỹ thuật số.
* **Tiềm năng của thanh toán số**: Mặc dù COD chiếm ưu thế lớn, sự hiện diện và tần suất sử dụng của một số ví điện tử/cổng thanh toán như Payaxis, Easypay, Jazzwallet cho thấy xu hướng sử dụng thanh toán kỹ thuật số đang hình thành và có tiềm năng phát triển.
* **Đa dạng nhưng thiếu tập trung**: Nền tảng/thị trường này cung cấp rất nhiều lựa chọn thanh toán, nhưng phần lớn trong số đó có tần suất sử dụng rất thấp. Điều này có thể dẫn đến sự phức tạp trong quản lý mà không mang lại hiệu quả sử dụng tương xứng.
* **Ý nghĩa hoạt động**: Sự phổ biến của COD đặt ra yêu cầu cao về quy trình vận chuyển, thu tiền và quản lý dòng tiền cho doanh nghiệp.
* **Cơ hội**: Việc thúc đẩy các phương thức thanh toán không dùng tiền mặt (ví dụ: thông qua ưu đãi, tăng cường sự tin cậy, đơn giản hóa quy trình) có thể giúp giảm chi phí vận hành liên quan đến COD và khuyến khích hành vi mua sắm trực tuyến liền mạch hơn.

#### 2.3.2 Phân tích Phương thức thanh toán theo giá trị đơn hàng

**Câu hỏi: Giá trị đơn hàng có khác nhau theo phương thức thanh toán không?**

# Vẽ boxplot

plt.figure(figsize=(12, 6))

sns.boxplot(data=df, x='payment\_method', y='grand\_total', showfliers=False)

plt.title("Phân bố giá trị đơn hàng theo phương thức thanh toán")

plt.xlabel("Phương thức thanh toán")

plt.ylabel("Giá trị đơn hàng (PKR)")

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

# In ra giá trị thấp nhất và cao nhất theo từng phương thức thanh toán

summary = df.groupby('payment\_method')['grand\_total'].agg(['min', 'max', 'mean', 'count']).sort\_values(by='mean', ascending=False)

print("Thống kê giá trị đơn hàng theo phương thức thanh toán:\n")

print(summary)

A graph of blue rectangular bars with white text

AI-generated content may be incorrect.

Thống kê giá trị đơn hàng theo phương thức thanh toán:

min max mean count

payment\_method

financesettlement 4212 389459 101645 15

mygateway 1 1155966 31796 669

bankalfalah -340 1028751 29841 23065

ublcreditcard 1 1279980 24883 882

easypay\_voucher 0 396832 17991 31176

jazzvoucher 0 17888000 16112 15633

apg 2 506847 15396 1758

internetbanking 1 535400 15292 472

mcblite 1 1280473 11780 723

Payaxis -873 599599 11671 97640

Easypay -1594 888065 11052 82896

Easypay\_MA 0 1315875 7764 14027

cashatdoorstep 8 1039479 6302 732

marketingexpense 1260 16200 5845 45

jazzwallet 0 17888000 4627 35145

cod 0 112149 3918 271926

customercredit 0 59700 17 7555

productcredit 0 0 0 125

Nhận xét:

* **COD**: Phổ biến số 1, nhưng giá trị đơn hàng thấp: COD là phương thức được khách hàng sử dụng nhiều nhất, nhưng lại đi kèm với giá trị đơn hàng trung bình thấp nhất trong số các phương thức thanh toán cho đơn hàng thực tế. Điều này cho thấy khách hàng có xu hướng sử dụng COD cho các giao dịch giá trị nhỏ, có thể vì sự tiện lợi hoặc tâm lý muốn kiểm tra hàng trước khi trả tiền đối với các đơn hàng ít rủi ro hơn về mặt giá trị.
* **Thanh toán điện tử**: Ít phổ biến hơn, nhưng giá trị đơn hàng cao hơn: Các phương thức thanh toán điện tử/trực tuyến có tần suất sử dụng thấp hơn nhiều so với COD, nhưng lại được sử dụng cho các đơn hàng có giá trị trung bình cao hơn đáng kể. Điều này gợi ý rằng khách hàng sẵn sàng sử dụng thanh toán trực tuyến cho các đơn hàng giá trị lớn hơn, có thể vì sự tiện lợi khi giao dịch số tiền lớn, hoặc họ có mức độ tin cậy cao hơn khi mua các mặt hàng đắt tiền.
* **Cơ hội thúc đẩy giá trị đơn hàng qua thanh toán điện tử**: Để tăng doanh thu trung bình trên mỗi giao dịch, doanh nghiệp có thể tập trung khuyến khích khách hàng sử dụng các phương thức thanh toán điện tử, đặc biệt là các phương thức có giá trị trung bình cao như easypay\_voucher, ublcreditcard, mygateway, bankalfalah. Các chiến dịch khuyến mãi cho đơn hàng giá trị cao khi thanh toán bằng các phương thức này có thể hiệu quả.
* **Quản lý rủi ro và chi phí COD**: Tỷ lệ lớn giao dịch giá trị thấp qua COD có thể làm tăng chi phí logistics (phí thu hộ, tỷ lệ hoàn hàng). Việc tìm cách chuyển đổi một phần giao dịch COD sang thanh toán điện tử (dù là các giao dịch nhỏ) có thể giúp tối ưu chi phí hoạt động.
* **Lưu ý về dữ liệu đặc biệt**: Cần xử lý hoặc phân tích riêng các "phương thức" như financesettlement, customercredit, productcredit vì chúng không đại diện cho hành vi thanh toán mua hàng thông thường của đa số khách hàng cá nhân. Các giá trị âm cần được điều tra để hiểu rõ chúng có phải là hoàn tiền hay không và ảnh hưởng thế nào đến số liệu trung bình.

Tóm lại, biểu đồ và dữ liệu cho thấy có sự phân hóa rõ rệt trong hành vi mua sắm dựa trên phương thức thanh toán. COD là vua về số lượng giao dịch nhưng cho đơn hàng giá trị nhỏ, trong khi thanh toán điện tử mặc dù ít được dùng hơn lại gắn liền với các giao dịch giá trị cao hơn. Điều này là thông tin quan trọng để xây dựng chiến lược kinh doanh và thanh toán phù hợp

### 2.4.Phân tích trạng thái đơn hàng

#### 2.4.1 Phân tích Trạng thái đơn hàng theo tần suất

**Câu hỏi: Tình hình các đơn hàng như thế nào?**

Để có cái nhìn tổng quan về quy trình xử lý đơn hàng, hãy cùng xem xét tần suất của các trạng thái đơn hàng như hoàn thành, đã hủy, đang xử lý, v.v.

# Trực quan hóa tần suất các trạng thái đơn hàng

plt.figure(figsize=(12, 6))

sns.countplot(data=df, x='status', palette='viridis', order=df['status'].value\_counts().index)

plt.title('Tần suất các trạng thái đơn hàng', fontsize=16, fontweight='bold')

plt.xlabel('Trạng thái đơn hàng', fontsize=14)

plt.ylabel('Tần suất', fontsize=14)

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

A graph with blue and black bars

AI-generated content may be incorrect.

Nhận xét:

* **complete (Hoàn thành)**: Chiếm tần suất cao nhất, cho thấy số lượng lớn đơn hàng đã được xử lý thành công và kết thúc.
* **Tỷ lệ hủy đơn hàng cao đáng ngại**: Điểm đáng chú ý nhất từ biểu đồ là số lượng đơn hàng bị "canceled" gần tương đương với số lượng đơn hàng "complete".
* **Lượng đơn hàng bị hoàn tiền đáng kể**: Số lượng đơn hàng bị "order\_refunded" và "refund" cũng khá cao.
* **Quy trình xử lý trung gian có vẻ hiệu quả**: Tần suất thấp của các trạng thái "pending", "processing", "payment\_review" cho thấy hệ thống xử lý đơn hàng (sau khi nhận và trước khi hoàn thành/hủy/hoàn tiền) có vẻ hoạt động tương đối trơn tru, ít bị tắc nghẽn. Tuy nhiên, có thể vấn đề nằm ở các khâu trước hoặc sau quy trình xử lý cốt lõi này (ví dụ: lúc đặt hàng dẫn đến hủy, hoặc sau khi nhận hàng dẫn đến hoàn tiền).
* **Các vấn đề nghiêm trọng (Fraud) ở mức thấp**: Tần suất rất thấp của trạng thái "fraud" là một tín hiệu tích cực, cho thấy hệ thống kiểm soát gian lận đang hoạt động tốt hoặc tỷ lệ gian lận vốn dĩ thấp.

Số lượng đơn hàng hoàn thành là cao nhất. Tuy nhiên, tỷ lệ đơn hàng bị hủy và hoàn tiền ở mức rất cao là điểm cần đặc biệt lưu tâm. Để nâng cao hiệu quả kinh doanh và sự hài lòng của khách hàng, doanh nghiệp cần tập trung điều tra nguyên nhân gốc rễ của tỷ lệ hủy và hoàn tiền cao này, từ đó đưa ra các biện pháp cải thiện phù hợp trong quy trình bán hàng, thông tin sản phẩm, logistics, và chính sách hậu mãi. Quy trình xử lý đơn hàng ở các khâu trung gian có vẻ đang hoạt động khá hiệu quả.

#### 2.4.2 Mối quan hệ giữa phương thức thanh toán, giá trị đơn hàng và tình trạng đơn hàng

Mục tiêu: Xem liệu có sự khác biệt về giá trị trung bình đơn hàng giữa các phương thức thanh toán và tỷ lệ thành công/hủy đơn hàng khác nhau không.

Ý nghĩa: Giúp hiểu hành vi thanh toán của khách hàng và tìm ra phương thức dễ gây hủy đơn nhất.

# Tính giá trị trung bình và tỷ lệ trạng thái theo payment\_method

payment\_status = df.groupby(['payment\_method', 'status']).agg({

    'grand\_total': ['mean', 'count']

}).reset\_index()

payment\_status.columns = ['payment\_method', 'status', 'avg\_grand\_total', 'count']

# Tổng số đơn theo payment\_method để tính tỷ lệ

total\_orders = df.groupby('payment\_method')['item\_id'].count().reset\_index().rename(columns={'item\_id':'total\_orders'})

# Gộp để tính tỷ lệ

payment\_status = payment\_status.merge(total\_orders, on='payment\_method')

payment\_status['order\_ratio'] = payment\_status['count'] / payment\_status['total\_orders']

# Vẽ biểu đồ

plt.figure(figsize=(14, 6))

sns.barplot(data=payment\_status, x='payment\_method', y='avg\_grand\_total', hue='status')

plt.title('Giá trị trung bình đơn hàng theo phương thức thanh toán và trạng thái')

plt.ylabel('Giá trị trung bình (PKR)')

plt.xticks(rotation=45)

plt.show()

plt.figure(figsize=(14, 6))

sns.barplot(data=payment\_status, x='payment\_method', y='order\_ratio', hue='status')

plt.title('Tỷ lệ trạng thái đơn hàng theo phương thức thanh toán')

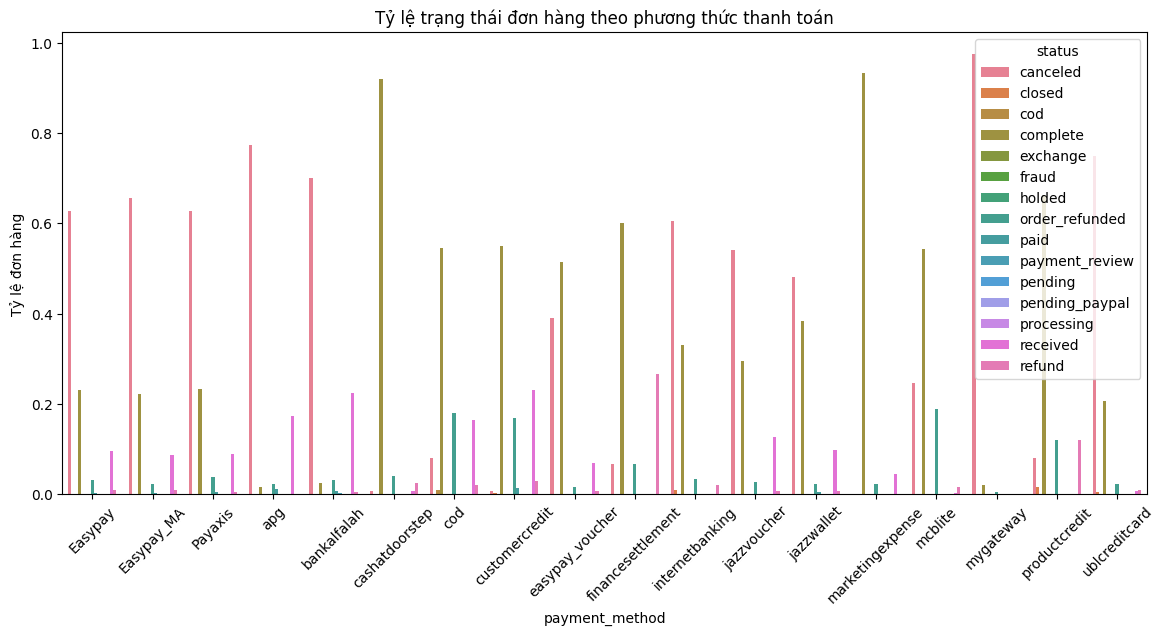
plt.ylabel('Tỷ lệ đơn hàng')

plt.xticks(rotation=45)

plt.show()

A graph with different colored lines

AI-generated content may be incorrect.



Nhận xét:

**"Giá trị trung bình đơn hàng theo phương thức thanh toán và trạng thái"**

* **COD (Cash on Delivery)** có lượng đơn hàng rất lớn nhưng giá trị trung bình đơn hàng thấp hơn so với các phương thức như Credit Card hoặc Bank Transfer.
* **Các phương thức thanh toán trực tuyến** (ví dụ: Credit Card, Easypaisa) thường có giá trị đơn hàng trung bình cao hơn, đặc biệt là khi trạng thái đơn hàng là hoàn tất (complete).

Một số phương thức có tỷ lệ đơn hàng bị hủy cao, có thể là do người dùng không tin tưởng hoặc thao tác thanh toán khó khăn.

**Gợi ý chiến lược:**

* Khuyến khích khách hàng sử dụng thẻ tín dụng hoặc thanh toán online bằng cách tặng thêm điểm tích lũy hoặc chiết khấu nhỏ.
* Phân tích thêm lý do hủy đơn hàng theo phương thức để tối ưu hệ thống thanh toán.

**"Tỷ lệ trạng thái đơn hàng theo phương thức thanh toán"**

* **COD (Cash on Delivery)** có tỷ lệ đơn hàng bị hủy cao hơn đáng kể so với các phương thức khác.
* **Các phương thức thanh toán điện tử** như Bank Transfer, Easypaisa, Credit Card có tỷ lệ đơn hàng hoàn tất cao hơn, cho thấy khách hàng thanh toán trước thường cam kết mua hàng hơn.

Một số phương thức hiếm gặp có số liệu lệch hoặc ít mẫu, nên cần kiểm tra kỹ nếu dùng để ra quyết định.

**Gợi ý chiến lược:**

* Cân nhắc giới hạn COD cho các khách hàng mới hoặc đơn hàng có giá trị cao.
* Tăng cường truyền thông lợi ích và an toàn khi thanh toán trực tuyến.
* Phân tích sâu thêm theo khu vực địa lý nếu có dữ liệu, vì hành vi có thể khác nhau theo vùng miền.

### 2.5.Phân tích khách hàng

#### 2.5.1 Phân tích Tổng quan khách hàng về mức độ mua sắm

total\_customers = df['customer\_id'].nunique()

print(f"Tổng số khách hàng: {total\_customers}")

Tổng số khách hàng: 115323

# Tính tổng chi tiêu theo từng khách hàng

customer\_revenue = df.groupby('customer\_id')['grand\_total'].sum().sort\_values(ascending=False)

# Khách hàng có giá trị cao nhất

top\_customers = customer\_revenue.head(10)

{

    "Top 10 khách hàng chi tiêu nhiều nhất": top\_customers.to\_dict()

}

{'Top 10 khách hàng chi tiêu nhiều nhất': {5032: 72150376,

50387: 35776000,

111057: 31338185,

109038: 28341357,

110215: 21969066,

10654: 19496005,

113694: 16531260,

39707: 16463501,

26527: 12502191,

8963: 12421422}}

# Tính tổng chi tiêu của từng khách hàng

customer\_total = df.groupby('customer\_id')['grand\_total'].sum().reset\_index()

# Sắp xếp theo giá trị mua sắm cao nhất

top\_customers = customer\_total.sort\_values(by='grand\_total', ascending=False).head(20)

# Biểu đồ scatter

plt.figure(figsize=(12, 6))

sns.scatterplot(data=top\_customers, x='customer\_id', y='grand\_total', s=100, color='orange')

plt.title('Top Khách hàng có giá trị mua sắm cao nhất', fontsize=16, fontweight='bold')

plt.xlabel('Customer ID', fontsize=12)

plt.ylabel('Tổng giá trị mua sắm (PKR)', fontsize=12)

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

A screen shot of a graph

AI-generated content may be incorrect.

Nhận xét:

* **Sự tập trung giá trị vào nhóm nhỏ**: Biểu đồ cho thấy rõ ràng rằng trong số 115323 khách hàng, có một nhóm nhỏ (các điểm ở phía trên cùng của biểu đồ) có tổng giá trị mua sắm vượt trội hơn hẳn phần lớn khách hàng còn lại (các điểm sẽ tập trung ở gần trục hoành nếu hiển thị tất cả).
* **Mức chi tiêu khổng lồ của Top Khách hàng**: Danh sách "Top 10 khách hàng chi tiêu nhiều nhất" xác nhận điều này. Khách hàng đứng đầu chi tiêu tới hơn 72 triệu PKR, và nhiều khách hàng khác trong top 10 cũng chi tiêu hàng chục triệu PKR. Mức chi tiêu này cao hơn rất nhiều lần so với giá trị đơn hàng trung bình được phân tích trước đó.

**Đánh giá:**

* **Tầm quan trọng chiến lược của khách hàng VIP**: Sự tồn tại của một nhóm khách hàng nhỏ nhưng có giá trị chi tiêu cực kỳ lớn nhấn mạnh tầm quan trọng chiến lược của phân khúc khách hàng này. Họ là những động lực doanh thu chính và việc giữ chân họ là cực kỳ quan trọng đối với sự thành công lâu dài của doanh nghiệp. Đây là minh chứng rõ ràng cho nguyên lý Pareto (quy tắc 80/20), nơi một tỷ lệ nhỏ khách hàng tạo ra phần lớn doanh thu.
* **Cơ hội cá nhân hóa và xây dựng mối quan hệ**: Việc xác định được những khách hàng này cho phép doanh nghiệp triển khai các chương trình chăm sóc khách hàng đặc biệt (VIP), ưu đãi cá nhân hóa, hoặc các dịch vụ độc quyền để tăng cường lòng trung thành và khuyến khích họ tiếp tục mua sắm.
* **Tiềm năng nhân rộng**: Phân tích sâu hơn về hành vi mua sắm (loại sản phẩm mua, tần suất, phản ứng với khuyến mãi) của nhóm khách hàng top đầu có thể giúp xác định các đặc điểm chung và từ đó xây dựng chiến lược thu hút hoặc chuyển đổi những khách hàng tiềm năng khác thành khách hàng giá trị cao.
* **Thách thức duy trì**: Mặc dù họ chi tiêu nhiều, không có gì đảm bảo họ sẽ tiếp tục làm như vậy. Cạnh tranh, trải nghiệm không tốt, hoặc thiếu sự công nhận có thể khiến họ chuyển sang đối thủ. Do đó, việc liên tục theo dõi, chăm sóc và đáp ứng nhu cầu của nhóm khách hàng này là cần thiết.

**Câu hỏi: Dựa trên giá trị mua sắm tổng thể, khách hàng của chúng ta được phân loại vào các nhóm giá trị khác nhau như thế nào?**

# Tổng giá trị mua sắm của mỗi khách hàng

customer\_total\_spent = df.groupby('customer\_id')['grand\_total'].sum()

# Phân nhóm khách hàng

customer\_total\_spent = customer\_total\_spent.sort\_values(ascending=False)

segments = pd.qcut(customer\_total\_spent, q=4, labels=['Low', 'Medium', 'High', 'Very High'])

# Biểu đồ phân phối phân khúc khách hàng

plt.figure(figsize=(8, 8))

segments.value\_counts().plot(kind='pie', autopct='%1.1f%%', startangle=90, colors=['lightblue', 'orange', 'green', 'red'])

plt.title('Phân khúc khách hàng theo giá trị mua sắm', fontsize=16)

plt.ylabel('')

plt.show()

A colorful circle with text

AI-generated content may be incorrect.

Nhận xét:

* **Phân bố khách hàng cân bằng đáng ngạc nhiên**: Điểm đáng chú ý nhất từ biểu đồ này là sự phân bố tỷ lệ khách hàng gần như đồng đều giữa tất cả bốn phân khúc giá trị mua sắm. Mỗi phân khúc đều chiếm khoảng một phần tư (25%) tổng số khách hàng. Đây là một cấu trúc phân khúc khách hàng khá cân bằng, khác biệt so với nhiều mô hình kinh doanh điển hình thường có số lượng lớn khách hàng ở phân khúc giá trị thấp.
* **Tỷ trọng lớn khách hàng giá trị cao**: Tổng cộng, 50% khách hàng thuộc vào hai phân khúc giá trị cao nhất (High và Very High). Điều này cho thấy doanh nghiệp có một lượng lớn khách hàng đóng góp giá trị đáng kể vào doanh thu. Đây là một tài sản quý báu.
* **Cơ hội phát triển ở phân khúc thấp hơn**: Mặc dù có lượng lớn khách hàng giá trị cao, 50% còn lại thuộc phân khúc Low và Medium cũng là một nhóm khách hàng tiềm năng. Kích thước đáng kể của hai phân khúc này tạo cơ hội để doanh nghiệp triển khai các chiến lược nhằm tăng giá trị mua sắm trung bình của họ theo thời gian (ví dụ: khuyến mãi mua kèm, chương trình tích điểm, gợi ý sản phẩm giá trị cao hơn).
* **Ý nghĩa chiến lược quản lý khách hàng**: Cấu trúc phân khúc cân bằng này đòi hỏi một chiến lược quản lý khách hàng đa dạng. Doanh nghiệp cần có các chương trình và ưu đãi riêng biệt cho từng nhóm:
  + **Nhóm High & Very High**: Tập trung vào việc duy trì mối quan hệ, tăng cường lòng trung thành, và có thể cung cấp các dịch vụ/lợi ích độc quyền (ví dụ: chương trình khách hàng thân thiết cao cấp, ưu đãi đặc biệt).
  + **Nhóm Medium & Low**: Áp dụng các chiến lược để khuyến khích mua sắm thường xuyên hơn hoặc mua các sản phẩm giá trị cao hơn (ví dụ: khuyến mãi theo giá trị đơn hàng, gợi ý sản phẩm cá nhân hóa, email marketing nhắm mục tiêu).

**Hạn chế về định nghĩa phân khúc**: Biểu đồ không cung cấp thông tin về ngưỡng giá trị mua sắm cụ thể để phân loại khách hàng vào từng phân khúc (ví dụ: giá trị từ X đến Y là Low, từ Y đến Z là Medium,...). Việc thiếu thông tin này khiến việc đánh giá chi tiết hơn về đặc điểm của từng nhóm (ngoài tỷ lệ) trở nên khó khăn. Cần có định nghĩa rõ ràng về từng phân khúc để hiểu sâu hơn và xây dựng chiến lược chính xác.

2.5.2 Phân tích Trạng thái đơn hàng đối với danh mục sản phẩm

**Câu hỏi: Trạng thái đơn hàng (như hoàn thành, đã hủy, hoàn tiền) phân bố như thế nào đối với các sản phẩm trong từng danh mục cụ thể?**

# Tạo bảng tổng hợp

category\_status = df.groupby(['category\_name', 'status']).size().unstack(fill\_value=0)

# Biểu đồ heatmap

plt.figure(figsize=(12, 8))

sns.heatmap(category\_status, annot=True, fmt='d', cmap='YlGnBu')

plt.title('Phân phối khách hàng theo danh mục sản phẩm và trạng thái đơn hàng', fontsize=16)

plt.xlabel('Trạng thái đơn hàng', fontsize=12)

plt.ylabel('Danh mục sản phẩm', fontsize=12)

plt.show()

A screenshot of a chart

AI-generated content may be incorrect.

Nhận xét:

* **Nổi bật nhất là ô "Mobiles & Tablets" - "complete"**, cho thấy Điện thoại & Máy tính bảng là danh mục có số lượng đơn hàng hoàn thành cao nhất. Tiếp theo là ô "Mobiles & Tablets" - "canceled" (xanh đậm thứ hai), cho thấy danh mục này cũng có số lượng đơn hàng bị hủy cao nhất, gần bằng số đơn hàng hoàn thành.
* **Các ô có tần suất cao khác bao gồm: "Men's Fashion" - "complete", "Men's Fashion" - "canceled", "Mobiles & Tablets" - "received", "Men's Fashion" - "received"**.
* **Các ô có tần suất trung bình khá như: "Mobiles & Tablets" - "order\_refunded", "Men's Fashion" - "order\_refunded", "Appliances" - "complete", "Women's Fashion" - "complete", "Appliances" - "canceled", "Women's Fashion" - "canceled", "Others" - "complete", "Superstore" - "complete".**
* **"Điện thoại & Máy tính bảng" và "Thời trang Nam"**: Con dao hai lưỡi: Các danh mục này là động lực doanh thu chính với số lượng đơn hàng hoàn thành cao nhất. Tuy nhiên, chúng cũng là nguồn gốc chính của các đơn hàng bị hủy và hoàn tiền về mặt số lượng tuyệt đối. Điều này cho thấy sự phổ biến và nhu cầu cao, nhưng đồng thời tồn tại những vấn đề đáng kể trong quy trình bán hàng hoặc sản phẩm ở các danh mục này, dẫn đến tỷ lệ đơn hàng không thành công cao.
* **Cần điều tra nguyên nhân hủy/hoàn tiền theo danh mục**: Tỷ lệ hủy/hoàn tiền cao ở "Mobiles & Tablets" có thể do giá trị sản phẩm lớn, rủi ro cao hơn khi mua sắm trực tuyến, hoặc kỳ vọng của khách hàng không được đáp ứng. Đối với "Men's Fashion", có thể liên quan đến vấn đề về kích cỡ, chất liệu, hoặc hình ảnh sản phẩm không khớp với thực tế. Việc phân tích sâu nguyên nhân cụ thể cho từng danh mục là cần thiết.
* **Các danh mục khác hoạt động ổn định hơn (về số lượng vấn đề tuyệt đối)**: Các danh mục có quy mô vừa và nhỏ hơn có số lượng đơn hàng hủy/hoàn tiền tuyệt đối thấp hơn. Mặc dù tỷ lệ có thể khác nhau, nhưng về mức độ đóng góp vào tổng số vấn đề, chúng ít nghiêm trọng hơn hai danh mục dẫn đầu.
* **Điểm sáng ở các trạng thái xử lý**: Số lượng đơn hàng ở các trạng thái như "pending", "processing", "payment\_review" nhìn chung là thấp trên hầu hết các danh mục. Điều này (giống như phân tích biểu đồ trạng thái tổng thể) gợi ý rằng các khâu xử lý trung gian không phải là nơi phát sinh nhiều vấn đề tắc nghẽn. Các vấn đề xảy ra chủ yếu ở đầu (trước khi hoàn thành hoặc bị hủy) hoặc cuối quy trình (sau khi nhận hàng dẫn đến hoàn tiền).

### 2.6.Phân thích chiết khấu và lợi nhuân

#### 2.6.1 Phân tích hành vi khách hàng theo từng danh mục

**Câu hỏi: Liệu việc giảm giá có thực sự thúc đẩy khách hàng mua nhiều sản phẩm hơn không?**

# Mô hình ẩn: Tác động của chiết khấu đến số lượng đặt hàng

  plt.figure(figsize=(10, 6))  # Thiết lập kích thước biểu đồ

  # Vẽ biểu đồ phân tán giữa số tiền chiết khấu và số lượng đặt hàng, phân biệt theo danh mục sản phẩm

  sns.scatterplot(

    x='discount\_amount',                  # Trục X: số tiền chiết khấu

    y='qty\_ordered',                      # Trục Y: số lượng sản phẩm được đặt

    data=df,                              # Dữ liệu nguồn

    hue='category\_name',                  # Màu sắc theo danh mục sản phẩm

    palette='Set1'                        # Bảng màu Set1 để dễ phân biệt

)

# Thêm tiêu đề và nhãn trục

plt.title('Tác động của chiết khấu đến số lượng đặt hàng', fontsize=16, fontweight='bold')

plt.xlabel('Số tiền chiết khấu', fontsize=14, fontweight='bold')

plt.ylabel('Số lượng đặt hàng', fontsize=14, fontweight='bold')

# Hiển thị chú thích bên phải biểu đồ

plt.legend(bbox\_to\_anchor=(1.05, 1), loc='upper left')

# Hiển thị biểu đồ

plt.show()

A white screen with black text

AI-generated content may be incorrect.

Nhận xét:

**Hiệu suất sản phẩm và chiết khấu:**

* Các danh mục như Mobiles & Tablets và Men's Fashion là những động lực doanh thu chính, thể hiện sự phổ biến vượt trội.
* Chiết khấu được áp dụng đáng kể, đặc biệt là cho các danh mục hàng đầu như Mobiles & Tablets. Tuy nhiên, phân tích cho thấy chiết khấu không phải là yếu tố duy nhất thúc đẩy số lượng đơn hàng; nhiều danh mục khác (như Appliances, Beauty & Grooming) đạt số lượng đơn hàng cao với mức chiết khấu tối thiểu, đặt ra câu hỏi về hiệu quả chi phí của việc giảm giá.

**Hành vi khách hàng và phương thức thanh toán:**

* COD (Thanh toán khi nhận hàng) là phương thức thanh toán được ưa chuộng nhất về số lượng giao dịch, phản ánh thói quen hoặc sự tin tưởng của khách hàng.
* Tuy nhiên, các đơn hàng sử dụng COD có giá trị trung bình thấp hơn đáng kể so với các phương thức thanh toán điện tử/trực tuyến.
* Quan trọng hơn, tỷ lệ hủy và hoàn tiền có xu hướng cao hơn đối với đơn hàng COD, trong khi thanh toán điện tử liên quan đến tỷ lệ hoàn thành đơn hàng cao hơn, cho thấy khách hàng thanh toán trước có cam kết mua hàng lớn hơn.
* Dữ liệu cũng chỉ ra sự tồn tại của một nhóm khách hàng VIP nhỏ nhưng có giá trị chi tiêu cực kỳ cao, đóng góp đáng kể vào tổng doanh thu. **Hiệu quả quy trình xử lý đơn hàng:**
* Số lượng đơn hàng hoàn thành (complete) là cao nhất, nhưng số lượng đơn hàng bị hủy (canceled) cũng ở mức báo động, gần bằng số đơn hàng hoàn thành.
* Lượng đơn hàng bị hoàn tiền (order\_refunded/refund) cũng đáng kể.
* Phân tích chi tiết theo danh mục cho thấy các danh mục bán chạy nhất (Mobiles & Tablets, Men's Fashion) cũng là những danh mục phát sinh nhiều đơn hàng bị hủy và hoàn tiền nhất về mặt số lượng tuyệt đối.
* Các trạng thái đơn hàng ở khâu xử lý trung gian (pending, processing) có tần suất thấp, cho thấy quy trình vận hành có vẻ không bị tắc nghẽn ở các bước này, nhưng vấn đề nằm ở trước (dẫn đến hủy) hoặc sau (dẫn đến hoàn tiền) các khâu này.

**Đánh giá tổng thể:**

Doanh nghiệp đang có một nền tảng với khối lượng giao dịch lớn và các danh mục sản phẩm chủ lực rõ ràng. Tuy nhiên, thách thức lớn nằm ở tỷ lệ đơn hàng không thành công (hủy và hoàn tiền) rất cao, đặc biệt tập trung ở các danh mục bán chạy nhất và gắn liền với phương thức thanh toán COD. Việc nhận diện và chăm sóc nhóm khách hàng giá trị cao là điểm mạnh, nhưng cần chiến lược để vừa giữ chân họ, vừa cải thiện trải nghiệm của khách hàng ở các phân khúc khác và trên các phương thức thanh toán. Tối ưu hóa chiến lược chiết khấu và giải quyết căn nguyên của tỷ lệ hủy/hoàn tiền cao là những ưu tiên hàng đầu để nâng cao hiệu quả hoạt động và lợi nhuận ròng.

#### 2.6.2 Phân tích theo tỷ lệ chiết khấu và lượng đặt hàng

**Câu hỏi : Giảm giá có thật sự kích thích mua hàng?**

# Giả sử df là DataFrame đã được tải và tiền xử lý, chứa các cột:

# 'discount\_percentage' và 'qty\_ordered'

# df = pd.read\_csv('cleaned\_Pakistan Largest Ecommerce Dataset.csv') # Đảm bảo bạn đã đọc file này

# --- Bước 0: Chuẩn bị và làm sạch dữ liệu cho phân tích này ---

df\_q8 = df.copy() # Tạo bản sao để thao tác

# Đảm bảo 'discount\_percentage' và 'qty\_ordered' là kiểu số

df\_q8['discount\_percentage'] = pd.to\_numeric(df\_q8['discount\_percentage'], errors='coerce')

df\_q8['qty\_ordered'] = pd.to\_numeric(df\_q8['qty\_ordered'], errors='coerce')

# Xử lý giá trị inf (vô cực) trong 'discount\_percentage' nếu có

# (Giá trị inf có thể xuất hiện nếu price=0 khi tính discount\_percentage)

df\_q8['discount\_percentage'].replace([np.inf, -np.inf], np.nan, inplace=True)

# Loại bỏ các dòng có giá trị NaN ở các cột quan trọng cho phân tích này

df\_q8.dropna(subset=['discount\_percentage', 'qty\_ordered'], inplace=True)

# Giới hạn giá trị discount\_percentage trong khoảng hợp lý (ví dụ 0-100)

# (Tùy chọn, nếu bạn chắc chắn tỷ lệ không thể > 100% hoặc < 0%)

df\_q8['discount\_percentage'] = np.clip(df\_q8['discount\_percentage'], 0, 100)

# --- Bước 1: Tạo các khoảng (bins) cho discount\_percentage ---

# Bạn có thể điều chỉnh các khoảng này tùy theo phân phối dữ liệu của bạn

bins = [-0.01, 0, 10, 20, 30, 40, 50, 75, 101] # -0.01 để bao gồm cả giá trị 0

# Tăng giới hạn cuối lên 101 để bao gồm cả 100%

labels = ['0%', '1-10%', '11-20%', '21-30%', '31-40%', '41-50%', '51-75%', '>75%']

# Nếu không có giá trị nào > 75%, bạn có thể điều chỉnh bins/labels cuối cùng

# Ví dụ: bins = [-0.01, 0, 10, 20, 30, 40, 51] và labels = ['0%', '1-10%', ..., '>50%']

# Hoặc dựa trên phân vị của discount\_percentage:

# bins = pd.qcut(df\_q8['discount\_percentage'], q=5, duplicates='drop').unique().categories.right

# labels = [f'{b.left}-{b.right}%' for b in pd.qcut(df\_q8['discount\_percentage'], q=5, duplicates='drop').unique().categories]

df\_q8['discount\_bin'] = pd.cut(df\_q8['discount\_percentage'],

                               bins=bins,

                               labels=labels,

                               right=True, # Khoảng (a, b]

                               include\_lowest=True) # Bao gồm giá trị nhỏ nhất

# --- Bước 2: Tính toán số lượng đặt hàng trung bình và tổng số lượng theo từng khoảng chiết khấu ---

# Sử dụng observed=False để đảm bảo tất cả các bins được hiển thị ngay cả khi không có dữ liệu

analysis\_q8\_avg = df\_q8.groupby('discount\_bin', observed=False)['qty\_ordered'].mean().reset\_index()

analysis\_q8\_sum = df\_q8.groupby('discount\_bin', observed=False)['qty\_ordered'].sum().reset\_index()

print("Số lượng đặt hàng TRUNG BÌNH theo khoảng chiết khấu:")

print(analysis\_q8\_avg)

print("\nTổng số lượng đặt hàng theo khoảng chiết khấu:")

print(analysis\_q8\_sum)

# --- Bước 3: Vẽ biểu đồ ---

# Biểu đồ 1: Số lượng đặt hàng TRUNG BÌNH theo khoảng chiết khấu

plt.figure(figsize=(12, 7))

sns.barplot(x='discount\_bin', y='qty\_ordered', data=analysis\_q8\_avg, palette='viridis', hue='discount\_bin', dodge=False, legend=False)

plt.title('Số Lượng Đặt Hàng Trung Bình theo Khoảng Tỷ Lệ Chiết Khấu', fontsize=16)

plt.xlabel('Khoảng Tỷ Lệ Chiết Khấu (%)', fontsize=12)

plt.ylabel('Số Lượng Đặt Hàng Trung Bình', fontsize=12)

plt.xticks(rotation=45, ha='right')

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

# (Tùy chọn) Biểu đồ Scatter plot để xem phân bố thô (có thể rất nhiễu)

# plt.figure(figsize=(10, 6))

# sns.scatterplot(x='discount\_percentage', y='qty\_ordered', data=df\_q8.sample(min(10000, len(df\_q8))), alpha=0.3) # Lấy mẫu nếu dữ liệu quá lớn

# plt.title('Phân bố giữa Tỷ lệ chiết khấu và Số lượng đặt hàng')

# plt.xlabel('Tỷ lệ Chiết Khấu (%)')

# plt.ylabel('Số Lượng Đặt Hàng')

# plt.show()

Số lượng đặt hàng TRUNG BÌNH theo khoảng chiết khấu:

discount\_bin qty\_ordered

0 0% 1.326614

1 1-10% 1.029783

2 11-20% 1.044323

3 21-30% 1.144843

4 31-40% 1.220763

5 41-50% 1.475929

6 51-75% 1.958686

7 >75% 2.082730

Tổng số lượng đặt hàng theo khoảng chiết khấu:

discount\_ bin qty\_ordered

0 0% 499409

1 1-10% 37896

2 11-20% 81429

3 21-30% 38382

4 31-40% 20919

5 41-50% 19590

6 51-75% 15503

7 >75% 44585

A graph with different colored bars

AI-generated content may be incorrect.

Nhận xét về Số Lượng Đặt Hàng **TRUNG BÌNH** theo Khoảng Chiết Khấu

* **Xu hướng rõ rệt**:  
  Khi tỷ lệ chiết khấu tăng, người tiêu dùng Pakistan có xu hướng mua số lượng sản phẩm trung bình nhiều hơn trên mỗi dòng sản phẩm.
* **Ngưỡng chiết khấu hiệu quả**:  
  Các mức chiết khấu sâu (đặc biệt **>50%**, với số lượng trung bình **~2 sản phẩm/dòng**) tỏ ra hiệu quả nhất trong việc thúc đẩy khách hàng mua nhiều đơn vị hơn cho một mặt hàng cụ thể.
* **Chiết khấu thấp ít tác động**:  
  Các mức chiết khấu nhỏ (**1-20%**) dường như không làm thay đổi đáng kể số lượng sản phẩm trung bình mua trên mỗi dòng so với khi không có chiết khấu.

# Biểu đồ 2: TỔNG số lượng đặt hàng theo khoảng chiết khấu

plt.figure(figsize=(12, 7))

sns.barplot(x='discount\_bin', y='qty\_ordered', data=analysis\_q8\_sum, palette='mako', hue='discount\_bin', dodge=False, legend=False)

plt.title('Tổng Số Lượng Đặt Hàng theo Khoảng Tỷ Lệ Chiết Khấu', fontsize=16)

plt.xlabel('Khoảng Tỷ Lệ Chiết Khấu (%)', fontsize=12)

plt.ylabel('Tổng Số Lượng Đặt Hàng', fontsize=12)

plt.xticks(rotation=45, ha='right')

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

A graph with a bar graph

AI-generated content may be incorrect.

Nhận xét Về **TỔNG** Số Lượng Đặt Hàng theo Khoảng Chiết Khấu

* **Giao dịch không chiết khấu vẫn chủ đạo**:  
  Phần lớn tổng số lượng sản phẩm được giao dịch trên thị trường (theo mẫu dữ liệu) thuộc về nhóm không áp dụng chiết khấu, cho thấy **nhu cầu tự nhiên** hoặc **giá cơ bản** đã đủ hấp dẫn.
* **Vai trò của chiết khấu vừa phải**:  
  Các khoảng chiết khấu thấp đến trung bình (ví dụ: **1-20%**) vẫn đóng góp một lượng đáng kể vào tổng số sản phẩm bán ra, phản ánh sự phổ biến của các mức ưu đãi này trên thị trường.
* **Chiết khấu sâu không đồng nghĩa tổng lượng bán cao nhất**:  
  Dù chiết khấu sâu (**>50%**) làm tăng số lượng trung bình/giao dịch, nhưng tổng số sản phẩm bán ở các mức này không phải là cao nhất, có thể do việc áp dụng chiết khấu sâu có chọn lọc hơn.

Kết luận chung :

* **Người tiêu dùng phản ứng với chiết khấu sâu**:  
  Cho thấy tiềm năng của các chương trình khuyến mãi lớn trong việc tăng số lượng bán trên mỗi giao dịch.
* **Tầm quan trọng của giá trị cơ bản**:  
  Một lượng lớn giao dịch vẫn diễn ra không cần chiết khấu, nhấn mạnh vai trò của việc **định giá hợp lý** và **sức hấp dẫn tự nhiên** của sản phẩm.
* **Chiến lược khuyến mãi đa dạng**:  
  Các nhà bán lẻ trên thị trường có thể đang áp dụng nhiều mức chiết khấu khác nhau cho các mục tiêu khác nhau – từ việc **kích thích mua nhiều hơn** với chiết khấu sâu, đến việc **duy trì doanh số ổn định** với chiết khấu nhẹ hoặc không chiết khấu.

#### 2.6.3.Phân tích theo tỷ lệ thành công các đơn hàng

**Câu hỏi: Tỷ lệ đơn hàng thành công so với hủy là bao nhiêu?**

# Kiểm tra sự tồn tại của các cột cần thiết

if 'increment\_id' not in df.columns or 'status' not in df.columns:

    print("Lỗi: DataFrame phải chứa cột 'increment\_id' và 'status' để thực hiện phân tích này.")

else:

    print("Đang phân tích trạng thái đơn hàng...")

    # --- Bước 1: Lấy trạng thái cho mỗi đơn hàng duy nhất ---

    order\_statuses\_df = df.drop\_duplicates(subset=['increment\_id'])[['increment\_id', 'status']].copy()

    # Xử lý cột 'status'

    if pd.api.types.is\_categorical\_dtype(order\_statuses\_df['status']):

        if 'Unknown' not in order\_statuses\_df['status'].cat.categories:

            order\_statuses\_df['status'] = order\_statuses\_df['status'].cat.add\_categories(['Unknown'])

        order\_statuses\_df['status'].fillna('Unknown', inplace=True)

    else:

        order\_statuses\_df['status'].fillna('Unknown', inplace=True)

    status\_counts = order\_statuses\_df['status'].value\_counts() # Sắp xếp theo tần suất giảm dần

    total\_unique\_orders = status\_counts.sum()

    print(f"\nTổng số đơn hàng duy nhất được phân tích: {total\_unique\_orders}")

    print("Phân bổ các trạng thái đơn hàng (số lượng) - Sắp xếp theo tần suất giảm dần:")

    print(status\_counts)

    # --- Bước 2: Chuẩn bị dữ liệu cho biểu đồ tròn tổng quan THEO YÊU CẦU MỚI ---

    status\_to\_group\_from = 'cod' # Trạng thái bắt đầu gom nhóm

    label\_for\_others = f'Others ({status\_to\_group\_from} & trở xuống)'

    fallback\_others\_label = 'Others (ít phổ biến)'

    if not status\_counts.empty:

        if status\_to\_group\_from in status\_counts.index:

            status\_list\_for\_indexing = status\_counts.index.tolist()

            try:

                group\_from\_idx = status\_list\_for\_indexing.index(status\_to\_group\_from)

                statuses\_to\_keep = status\_counts.iloc[:group\_from\_idx]

                statuses\_to\_group\_sum = status\_counts.iloc[group\_from\_idx:].sum()

                status\_pie\_data\_list = []

                if not statuses\_to\_keep.empty:

                    status\_pie\_data\_list.append(statuses\_to\_keep)

                if statuses\_to\_group\_sum > 0:

                    status\_pie\_data\_list.append(pd.Series({label\_for\_others: statuses\_to\_group\_sum}))

                if status\_pie\_data\_list:

                    status\_pie\_data = pd.concat(status\_pie\_data\_list)

                    status\_pie\_data = status\_pie\_data[status\_pie\_data > 0]

                else:

                    status\_pie\_data = pd.Series({label\_for\_others: statuses\_to\_group\_sum})

                    status\_pie\_data = status\_pie\_data[status\_pie\_data > 0]

            except ValueError:

                print(f"Lỗi: Trạng thái '{status\_to\_group\_from}' không tìm thấy khi lấy index. Sử dụng logic N\_TOP.")

                N\_TOP\_STATUSES = min(7, len(status\_counts))

                top\_statuses = status\_counts.nlargest(N\_TOP\_STATUSES)

                other\_count = status\_counts.nsmallest(len(status\_counts) - N\_TOP\_STATUSES).sum()

                if other\_count > 0:

                    status\_pie\_data = pd.concat([top\_statuses, pd.Series({fallback\_others\_label: other\_count})])

                else:

                    status\_pie\_data = top\_statuses

                label\_for\_others = fallback\_others\_label # Cập nhật label cho others

        else:

            print(f"Trạng thái '{status\_to\_group\_from}' không tìm thấy trong danh sách. Sử dụng logic N\_TOP\_STATUSES để gom nhóm 'Others'.")

            N\_TOP\_STATUSES = min(7, len(status\_counts))

            if len(status\_counts) > N\_TOP\_STATUSES:

                top\_statuses = status\_counts.nlargest(N\_TOP\_STATUSES)

                other\_count = status\_counts.nsmallest(len(status\_counts) - N\_TOP\_STATUSES).sum()

                if other\_count > 0:

                    status\_pie\_data = pd.concat([top\_statuses, pd.Series({fallback\_others\_label: other\_count})])

                else:

                    status\_pie\_data = top\_statuses

            else:

                status\_pie\_data = status\_counts

            label\_for\_others = fallback\_others\_label # Cập nhật label cho others

    else:

        status\_pie\_data = pd.Series()

    # --- Bước 3: Vẽ biểu đồ tròn tổng quan (với dữ liệu đã gom nhóm mới) ---

    if not status\_pie\_data.empty:

        plt.figure(figsize=(13, 10))

        default\_pct\_distance = 0.5 # Khoảng cách mặc định cho tỷ lệ

        patches, texts, autotexts = plt.pie(

            status\_pie\_data,

            labels=None,

            autopct='%1.1f%%',

            startangle=90,

            wedgeprops={'edgecolor': 'black'},

            pctdistance=default\_pct\_distance # Áp dụng khoảng cách mặc định

        )

        plt.title('Tỷ Lệ Các Trạng Thái Đơn Hàng (Gom từ "cod")', fontsize=14)

        plt.axis('equal')

        # Tùy chỉnh vị trí tỷ lệ cho miếng "Others"

        # Tìm index của miếng "Others"

        others\_slice\_index = -1 # Mặc định không tìm thấy

        for i, label in enumerate(status\_pie\_data.index):

            if label\_for\_others in label: # Kiểm tra nếu nhãn chứa tên của "Others"

                others\_slice\_index = i

                break

        if others\_slice\_index != -1 and others\_slice\_index < len(autotexts):

            target\_autotext = autotexts[others\_slice\_index]

            target\_patch = patches[others\_slice\_index]

            # Tính toán vị trí mới cho tỷ lệ của "Others" với pctdistance = 0.75

            angle\_rad = np.deg2rad((target\_patch.theta1 + target\_patch.theta2) / 2)

            desired\_pct\_distance\_others = 0.75

            x\_new = desired\_pct\_distance\_others \* np.cos(angle\_rad)

            y\_new = desired\_pct\_distance\_others \* np.sin(angle\_rad)

            target\_autotext.set\_position((x\_new, y\_new))

            # Có thể cần điều chỉnh căn lề nếu text bị lệch

            # target\_autotext.set\_horizontalalignment('center')

            # target\_autotext.set\_verticalalignment('center')

        plt.legend(patches, status\_pie\_data.index, loc="center left", bbox\_to\_anchor=(1.05, 0, 0.5, 1), fontsize=10)

        plt.subplots\_adjust(right=0.75)

        plt.show()

    else:

        print("Không có dữ liệu trạng thái để vẽ biểu đồ tròn tổng quan sau khi áp dụng logic gom nhóm mới.")

    # --- Bước 4: Tập trung vào "Thành công" và "Hủy" (phần này giữ nguyên logic) ---

    # (Giữ nguyên phần code này như trước, vì yêu cầu chỉ áp dụng cho biểu đồ tổng quan đầu tiên)

    print("\nCác trạng thái đơn hàng duy nhất có trong dữ liệu của bạn (để bạn kiểm tra và cập nhật list bên dưới):")

    print(list(status\_counts.index))

    successful\_statuses\_list = ['complete', 'closed', 'received', 'delivered']

    canceled\_statuses\_list = ['canceled', 'fraud', 'order\_refunded']

    count\_successful = status\_counts[status\_counts.index.isin(successful\_statuses\_list)].sum()

    count\_canceled = status\_counts[status\_counts.index.isin(canceled\_statuses\_list)].sum()

    count\_others = total\_unique\_orders - count\_successful - count\_canceled

    if count\_successful > 0 or count\_canceled > 0 or count\_others > 0:

        simple\_labels\_map = {

            'Thành Công': count\_successful,

            'Đã Hủy': count\_canceled,

            'Khác': count\_others

        }

        final\_labels\_for\_legend = [label for label, size in simple\_labels\_map.items() if size > 0]

        final\_sizes = [size for size in simple\_labels\_map.values() if size > 0]

        available\_colors = {'Thành Công': '#4CAF50', 'Đã Hủy': '#F44336', 'Khác': '#9E9E9E'}

        final\_colors = []

        final\_explode = []

        for label in final\_labels\_for\_legend:

            if 'Thành Công' in label:

                final\_colors.append(available\_colors['Thành Công'])

                final\_explode.append(0.05)

            elif 'Đã Hủy' in label:

                final\_colors.append(available\_colors['Đã Hủy'])

                final\_explode.append(0.05)

            else: # Khác

                final\_colors.append(available\_colors['Khác'])

                final\_explode.append(0)

        if final\_sizes:

            plt.figure(figsize=(11, 8))

            patches\_focused, texts\_focused, autotexts\_focused = plt.pie(

                final\_sizes,

                labels=None,

                autopct=lambda p: '{:.1f}%'.format(p) if p > 0 else '',

                startangle=90,

                colors=final\_colors,

                explode=final\_explode,

                wedgeprops={'edgecolor': 'black'},

                pctdistance=0.5 # Giữ nguyên hoặc thay đổi nếu muốn

            )

            plt.title('Tỷ Lệ Đơn Hàng Thành Công, Hủy và Khác', fontsize=16)

            plt.axis('equal')

            plt.legend(patches\_focused, final\_labels\_for\_legend, loc="center left", bbox\_to\_anchor=(1, 0, 0.5, 1), fontsize=10)

            plt.subplots\_adjust(right=0.7)

            plt.show()

            print(f"\nPhân tích chi tiết (dựa trên định nghĩa của bạn về 'thành công' và 'hủy'):")

            print(f"Tổng số đơn hàng duy nhất: {total\_unique\_orders}")

            print(f"Số đơn hàng được coi là 'Thành Công': {count\_successful} ({count\_successful/total\_unique\_orders\*100:.1f}%)")

            print(f"Số đơn hàng được coi là 'Đã Hủy': {count\_canceled} ({count\_canceled/total\_unique\_orders\*100:.1f}%)")

            print(f"Số đơn hàng 'Khác': {count\_others} ({count\_others/total\_unique\_orders\*100:.1f}%)")

        else:

            print("Không có đủ dữ liệu sau khi nhóm 'Thành Công', 'Đã Hủy', 'Khác' để vẽ biểu đồ.")

    else:

        print("\nKhông thể tính toán số liệu cho 'Thành Công' hoặc 'Đã Hủy'. Vui lòng kiểm tra lại danh sách `successful\_statuses\_list` và `canceled\_statuses\_list` cho phù hợp với dữ liệu của bạn.")

status\_pie\_data\_list = []

                if not statuses\_to\_keep.empty:

                    status\_pie\_data\_list.append(statuses\_to\_keep)

                if statuses\_to\_group\_sum > 0:

                    status\_pie\_data\_list.append(pd.Series({label\_for\_others: statuses\_to\_group\_sum}))

                if status\_pie\_data\_list:

                    status\_pie\_data = pd.concat(status\_pie\_data\_list)

                    status\_pie\_data = status\_pie\_data[status\_pie\_data > 0]

else:

                    status\_pie\_data = pd.Series({label\_for\_others: statuses\_to\_group\_sum})

                    status\_pie\_data = status\_pie\_data[status\_pie\_data > 0]

            except ValueError:

                print(f"Lỗi: Trạng thái '{status\_to\_group\_from}' không tìm thấy khi lấy index. Sử dụng logic N\_TOP.")

                N\_TOP\_STATUSES = min(7, len(status\_counts))

                top\_statuses = status\_counts.nlargest(N\_TOP\_STATUSES)

                other\_count = status\_counts.nsmallest(len(status\_counts) - N\_TOP\_STATUSES).sum()

                if other\_count > 0:

                    status\_pie\_data = pd.concat([top\_statuses, pd.Series({fallback\_others\_label: other\_count})])

                else:

                    status\_pie\_data = top\_statuses

                label\_for\_others = fallback\_others\_label # Cập nhật label cho others

        else:

            print(f"Trạng thái '{status\_to\_group\_from}' không tìm thấy trong danh sách. Sử dụng logic N\_TOP\_STATUSES để gom nhóm 'Others'.")

            N\_TOP\_STATUSES = min(7, len(status\_counts))

            if len(status\_counts) > N\_TOP\_STATUSES:

                top\_statuses = status\_counts.nlargest(N\_TOP\_STATUSES)

                other\_count = status\_counts.nsmallest(len(status\_counts) - N\_TOP\_STATUSES).sum()

                if other\_count > 0:

                    status\_pie\_data = pd.concat([top\_statuses, pd.Series({fallback\_others\_label: other\_count})])

                else:

                    status\_pie\_data = top\_statuses

            else:

                status\_pie\_data = status\_counts

            label\_for\_others = fallback\_others\_label # Cập nhật label cho others

    else:

        status\_pie\_data = pd.Series()

# --- Bước 3: Vẽ biểu đồ tròn tổng quan (với dữ liệu đã gom nhóm mới) ---

    if not status\_pie\_data.empty:

        plt.figure(figsize=(13, 10))

        default\_pct\_distance = 0.5 # Khoảng cách mặc định cho tỷ lệ

        patches, texts, autotexts = plt.pie(

            status\_pie\_data,

            labels=None,

            autopct='%1.1f%%',

            startangle=90,

            wedgeprops={'edgecolor': 'black'},

            pctdistance=default\_pct\_distance # Áp dụng khoảng cách mặc định

        )

        plt.title('Tỷ Lệ Các Trạng Thái Đơn Hàng (Gom từ "cod")', fontsize=14)

        plt.axis('equal')

        # Tùy chỉnh vị trí tỷ lệ cho miếng "Others"

        # Tìm index của miếng "Others"

        others\_slice\_index = -1 # Mặc định không tìm thấy

        for i, label in enumerate(status\_pie\_data.index):

            if label\_for\_others in label: # Kiểm tra nếu nhãn chứa tên của "Others"

                others\_slice\_index = i

                break

        if others\_slice\_index != -1 and others\_slice\_index < len(autotexts):

            target\_autotext = autotexts[others\_slice\_index]

            target\_patch = patches[others\_slice\_index]

            # Tính toán vị trí mới cho tỷ lệ của "Others" với pctdistance = 0.75

            angle\_rad = np.deg2rad((target\_patch.theta1 + target\_patch.theta2) / 2)

            desired\_pct\_distance\_others = 0.75

            x\_new = desired\_pct\_distance\_others \* np.cos(angle\_rad)

            y\_new = desired\_pct\_distance\_others \* np.sin(angle\_rad)

            target\_autotext.set\_position((x\_new, y\_new))

            # Có thể cần điều chỉnh căn lề nếu text bị lệch

            # target\_autotext.set\_horizontalalignment('center')

            # target\_autotext.set\_verticalalignment('center')

        plt.legend(patches, status\_pie\_data.index, loc="center left", bbox\_to\_anchor=(1.05, 0, 0.5, 1), fontsize=10)

        plt.subplots\_adjust(right=0.75)

        plt.show()

    else:

        print("Không có dữ liệu trạng thái để vẽ biểu đồ tròn tổng quan sau khi áp dụng logic gom nhóm mới.")

# --- Bước 4: Tập trung vào "Thành công" và "Hủy" (phần này giữ nguyên logic) ---

    # (Giữ nguyên phần code này như trước, vì yêu cầu chỉ áp dụng cho biểu đồ tổng quan đầu tiên)

    print("\nCác trạng thái đơn hàng duy nhất có trong dữ liệu của bạn (để bạn kiểm tra và cập nhật list bên dưới):")

    print(list(status\_counts.index))

    successful\_statuses\_list = ['complete', 'closed', 'received', 'delivered']

    canceled\_statuses\_list = ['canceled', 'fraud', 'order\_refunded']

    count\_successful = status\_counts[status\_counts.index.isin(successful\_statuses\_list)].sum()

    count\_canceled = status\_counts[status\_counts.index.isin(canceled\_statuses\_list)].sum()

    count\_others = total\_unique\_orders - count\_successful - count\_canceled

    if count\_successful > 0 or count\_canceled > 0 or count\_others > 0:

        simple\_labels\_map = {

            'Thành Công': count\_successful,

            'Đã Hủy': count\_canceled,

            'Khác': count\_others

        }

        final\_labels\_for\_legend = [label for label, size in simple\_labels\_map.items() if size > 0]

        final\_sizes = [size for size in simple\_labels\_map.values() if size > 0]

        available\_colors = {'Thành Công': '#4CAF50', 'Đã Hủy': '#F44336', 'Khác': '#9E9E9E'}

        final\_colors = []

        final\_explode = []

        for label in final\_labels\_for\_legend:

            if 'Thành Công' in label:

                final\_colors.append(available\_colors['Thành Công'])

                final\_explode.append(0.05)

            elif 'Đã Hủy' in label:

                final\_colors.append(available\_colors['Đã Hủy'])

                final\_explode.append(0.05)

            else: # Khác

                final\_colors.append(available\_colors['Khác'])

                final\_explode.append(0)

        if final\_sizes:

            plt.figure(figsize=(11, 8))

            patches\_focused, texts\_focused, autotexts\_focused = plt.pie(

                final\_sizes,

                labels=None,

                autopct=lambda p: '{:.1f}%'.format(p) if p > 0 else '',

                startangle=90,

                colors=final\_colors,

                explode=final\_explode,

                wedgeprops={'edgecolor': 'black'},

                pctdistance=0.5 # Giữ nguyên hoặc thay đổi nếu muốn

            )

plt.title('Tỷ Lệ Đơn Hàng Thành Công, Hủy và Khác', fontsize=16)

            plt.axis('equal')

            plt.legend(patches\_focused, final\_labels\_for\_legend, loc="center left", bbox\_to\_anchor=(1, 0, 0.5, 1), fontsize=10)

            plt.subplots\_adjust(right=0.7)

            plt.show()

            print(f"\nPhân tích chi tiết (dựa trên định nghĩa của bạn về 'thành công' và 'hủy'):")

            print(f"Tổng số đơn hàng duy nhất: {total\_unique\_orders}")

            print(f"Số đơn hàng được coi là 'Thành Công': {count\_successful} ({count\_successful/total\_unique\_orders\*100:.1f}%)")

            print(f"Số đơn hàng được coi là 'Đã Hủy': {count\_canceled} ({count\_canceled/total\_unique\_orders\*100:.1f}%)")

            print(f"Số đơn hàng 'Khác': {count\_others} ({count\_others/total\_unique\_orders\*100:.1f}%)")

        else:

            print("Không có đủ dữ liệu sau khi nhóm 'Thành Công', 'Đã Hủy', 'Khác' để vẽ biểu đồ.")

    else:

        print("\nKhông thể tính toán số liệu cho 'Thành Công' hoặc 'Đã Hủy'. Vui lòng kiểm tra lại danh sách `successful\_statuses\_list` và `canceled\_statuses\_list` cho phù hợp với dữ liệu của bạn.")

Đang phân tích trạng thái đơn hàng...

Tổng số đơn hàng duy nhất được phân tích: 408749

Phân bổ các trạng thái đơn hàng (số lượng) - Sắp xếp theo tần suất giảm dần:

status

complete 169515

canceled 148640

order\_refunded 49062

received 35805

refund 4083

cod 878

closed 331

paid 294

pending 43

payment\_review 39

holded 20

processing 20

fraud 10

pending\_paypal 5

exchange 4

Unknown 0

Name: count, dtype: int64

A colorful pie chart with text with Crust in the background

AI-generated content may be incorrect.

Các trạng thái đơn hàng duy nhất có trong dữ liệu của bạn (để bạn kiểm tra và cập nhật list bên dưới):

['complete', 'canceled', 'order\_refunded', 'received', 'refund', 'cod', 'closed', 'paid', 'pending', 'payment\_review', 'holded', 'processing', 'fraud', 'pending\_paypal', 'exchange', 'Unknown']

A green and red circle with a number of percentages

AI-generated content may be incorrect.

Phân tích chi tiết (dựa trên định nghĩa của bạn về 'thành công' và 'hủy'):

Tổng số đơn hàng duy nhất: 408749

Số đơn hàng được coi là 'Thành Công': 205651 (50.3%)

Số đơn hàng được coi là 'Đã Hủy': 197712 (48.4%)

Số đơn hàng 'Khác': 5386 (1.3%)

Nhận xét:

**Biểu đồ Tròn Tổng Quan Các Trạng Thái Đơn Hàng**

* **Thực trạng thị trường**:  
  Biểu đồ cho thấy complete (hoàn thành) và canceled (đã hủy) là hai trạng thái phổ biến nhất, phản ánh hai kết cục chính của các giao dịch thương mại điện tử (TMĐT) tại Pakistan trong tập dữ liệu này.
* **Các trạng thái cần quan tâm**:  
  Các trạng thái như order\_refunded (đơn hàng đã hoàn tiền) và received (đã nhận) cũng là những trạng thái quan trọng, giúp hiểu rõ hơn về vòng đời của đơn hàng.
* **Nhóm "Others"**:  
  Tỷ lệ của nhóm này cho thấy phần trăm các đơn hàng đang ở các giai đoạn trung gian hoặc có tần suất thấp hơn trên toàn thị trường (hoặc trên nền tảng được lấy mẫu).

**Biểu đồ Tròn Tập Trung "Thành Công", "Hủy", "Khác"**

* **Insight chính về thị trường**:  
  Tỷ lệ đơn hàng được phân loại là **"Đã Hủy"** (bao gồm canceled, fraud, order\_refunded) ở mức **48.4%** là một con số rất đáng chú ý trên quy mô thị trường TMĐT Pakistan (theo mẫu dữ liệu này). Nó gần như ngang bằng với tỷ lệ đơn hàng **"Thành Công"** là **50.3%**.
* **Đặc điểm thị trường**:  
  Tỷ lệ hủy và hoàn tiền cao có thể là một đặc điểm của thị trường TMĐT đang phát triển như Pakistan, nơi các yếu tố sau có thể góp phần vào tỷ lệ này:
  + Sự phổ biến của phương thức thanh toán **COD (Cash on Delivery)**, cho phép khách hàng thay đổi quyết định dễ dàng hơn khi nhận hàng.
  + Thói quen mua sắm (ví dụ: đặt nhiều đơn rồi chọn một).
  + Các vấn đề về **logistics**, chất lượng sản phẩm từ nhiều nhà cung cấp khác nhau.
  + Chính sách đổi trả/hoàn tiền của các sàn hoặc nhà bán lẻ.
  + Mức độ tin cậy vào mua sắm trực tuyến.
* **Tỷ lệ "Khác" (1.3%)**:  
  Cho thấy hầu hết các đơn hàng trong mẫu dữ liệu này đã có trạng thái cuối cùng tương đối rõ ràng.

## 3. Xây dựng mô hình dự đoán

**Bước 1: Chuẩn bị Dữ liệu cho Chuỗi Thời gian (Sử dụng grand\_total và lọc status)**

Đọc dữ liệu: Tải file cleaned\_Pakistan Largest Ecommerce Dataset.csv vào DataFrame.

Kiểm tra cột: Đảm bảo các cột 'status', 'increment\_id', 'created\_at' (hoặc 'working\_date'), và 'grand\_total' tồn tại và có tên chính xác.

Lọc theo status: Chỉ giữ lại các dòng có status là 'complete' hoặc 'received'.

Lấy grand\_total duy nhất cho mỗi đơn hàng: Vì grand\_total là tổng của đơn hàng, chúng ta cần loại bỏ các dòng trùng lặp theo increment\_id để mỗi đơn hàng chỉ được tính một lần.

Chuyển đổi cột ngày tháng: Đảm bảo cột ngày tháng (ví dụ: 'created\_at') được chuyển sang kiểu datetime.

Đặt làm chỉ mục (Index): Đặt cột ngày tháng làm chỉ mục.

Tổng hợp grand\_total hàng tháng: Tính tổng grand\_total của các đơn hàng thành công theo tháng.

Xử lý giá trị thiếu (nếu có):

# --- Bắt đầu xử lý cho mô hình dự báo doanh số ---

# Xác định cột ngày tháng và cột giá trị sẽ sử dụng

date\_column\_for\_ts = None

value\_column\_for\_ts = 'grand\_total' # Giả định cột giá trị là 'grand\_total'

# Kiểm tra và chọn cột ngày tháng

if 'created\_at' in df.columns:

    date\_column\_for\_ts = 'created\_at'

elif 'created\_at ' in df.columns: # Kiểm tra trường hợp có dấu cách ở cuối

    date\_column\_for\_ts = 'created\_at '

    print("Lưu ý: Sử dụng cột 'created\_at ' (có dấu cách ở cuối) làm cột ngày tháng.")

    # Bạn có thể đổi tên cột này để dễ sử dụng hơn:

    # df.rename(columns={'created\_at ': 'created\_at'}, inplace=True)

    # date\_column\_for\_ts = 'created\_at'

elif 'working\_date' in df.columns:

    date\_column\_for\_ts = 'working\_date'

else:

    print("LỖI: Không tìm thấy cột ngày tháng phù hợp ('created\_at', 'created\_at ', hoặc 'working\_date').")

    # Khởi tạo monthly\_sales rỗng để các cell sau không báo lỗi biến chưa định nghĩa

    monthly\_sales = pd.Series(dtype='float64')

# Các cột cần thiết cơ bản (sẽ cập nhật cột ngày tháng sau)

required\_base\_cols = ['status', 'increment\_id', value\_column\_for\_ts]

required\_cols\_step1 = []

if date\_column\_for\_ts: # Nếu đã tìm thấy cột ngày tháng

    required\_cols\_step1 = required\_base\_cols + [date\_column\_for\_ts]

    missing\_for\_step1 = [col for col in required\_cols\_step1 if col not in df.columns]

else: # Nếu không tìm thấy cột ngày tháng ở trên

    missing\_for\_step1 = ["cột ngày tháng ('created\_at' hoặc 'working\_date')"] + \

                        [col for col in required\_base\_cols if col not in df.columns]

if missing\_for\_step1 and not (len(missing\_for\_step1) == 1 and "cột ngày tháng" in missing\_for\_step1[0] and date\_column\_for\_ts is None) :

    # In lỗi nếu thiếu các cột cơ bản, hoặc thiếu cột ngày tháng và date\_column\_for\_ts đã được xác định (nghĩa là tên cột ngày tháng trong required\_cols\_step1 sai)

    actual\_missing = [col for col in required\_cols\_step1 if col not in df.columns] # Kiểm tra lại các cột thực sự thiếu

    if actual\_missing:

        print(f"LỖI: Thiếu các cột cần thiết cho Bước 1: {actual\_missing}")

    monthly\_sales = pd.Series(dtype='float64')

elif date\_column\_for\_ts is None: # Đã in lỗi không tìm thấy cột ngày tháng ở trên

    pass # monthly\_sales đã được khởi tạo rỗng

else:

    print(f"Sử dụng cột '{date\_column\_for\_ts}' cho ngày tháng và '{value\_column\_for\_ts}' cho doanh số.")

    # Tạo bản sao để thao tác

    df\_forecast\_prep = df.copy()

    # Chuyển đổi cột ngày tháng sang datetime (nếu chưa phải)

    if not pd.api.types.is\_datetime64\_any\_dtype(df\_forecast\_prep[date\_column\_for\_ts]):

        print(f"Đang cố gắng chuyển đổi cột '{date\_column\_for\_ts}' sang datetime...")

        df\_forecast\_prep[date\_column\_for\_ts] = pd.to\_datetime(df\_forecast\_prep[date\_column\_for\_ts], errors='coerce')

    # Chuyển đổi cột giá trị sang kiểu số (nếu chưa phải)

    if not pd.api.types.is\_numeric\_dtype(df\_forecast\_prep[value\_column\_for\_ts]):

         print(f"Đang cố gắng chuyển đổi cột '{value\_column\_for\_ts}' sang kiểu số...")

         df\_forecast\_prep[value\_column\_for\_ts] = pd.to\_numeric(df\_forecast\_prep[value\_column\_for\_ts], errors='coerce')

    # 1. Lọc theo status 'complete' và 'received'

    successful\_statuses = ['complete', 'received']

    df\_successful = df\_forecast\_prep[df\_forecast\_prep['status'].isin(successful\_statuses)]

    print(f"\nSố dòng sau khi lọc theo status {successful\_statuses}: {len(df\_successful)}")

    if not df\_successful.empty:

        # 2. Lấy grand\_total duy nhất cho mỗi đơn hàng thành công

        # Chọn các cột cần thiết trước khi drop\_duplicates

        cols\_to\_keep = [date\_column\_for\_ts, 'increment\_id', value\_column\_for\_ts]

        # Thêm 'discount\_percentage' nếu bạn muốn tổng hợp nó làm biến ngoại sinh

        # if 'discount\_percentage' in df\_successful.columns:

        #     cols\_to\_keep.append('discount\_percentage')

        df\_unique\_orders = df\_successful[cols\_to\_keep].drop\_duplicates(subset=['increment\_id'], keep='first')

        print(f"Số đơn hàng thành công duy nhất: {len(df\_unique\_orders)}")

        # 3. Loại bỏ các dòng có ngày tháng hoặc giá trị là NaT/NaN sau các bước trên

        df\_unique\_orders.dropna(subset=[date\_column\_for\_ts, value\_column\_for\_ts], inplace=True)

        print(f"Số đơn hàng thành công duy nhất sau khi loại bỏ NaN ở cột ngày/giá trị: {len(df\_unique\_orders)}")

        if not df\_unique\_orders.empty:

            # 4. Đặt cột ngày tháng làm index

            df\_ts = df\_unique\_orders.set\_index(date\_column\_for\_ts)

            # 5. Tổng hợp grand\_total hàng tháng ('M')

            monthly\_sales = df\_ts[value\_column\_for\_ts].resample('M').sum()

            monthly\_sales = monthly\_sales.fillna(0)

            print("\nDoanh thu hàng tháng (từ grand\_total của đơn hàng thành công, 5 dòng đầu):")

            print(monthly\_sales.head())

            print(f"\nSố lượng tháng có dữ liệu: {len(monthly\_sales)}")

            if len(monthly\_sales) < 12:

                print("\nCẢNH BÁO: Dữ liệu chuỗi thời gian quá ngắn để xây dựng mô hình dự báo đáng tin cậy.")

                print("Thông thường cần ít nhất 2-3 chu kỳ mùa vụ (ví dụ 2-3 năm dữ liệu hàng tháng).")

        else:

            print("DataFrame rỗng sau khi xử lý đơn hàng duy nhất và loại bỏ NaN.")

            monthly\_sales = pd.Series(dtype='float64')

    else:

        print("Không có đơn hàng nào với status 'complete' hoặc 'received'.")

        monthly\_sales = pd.Series(dtype='float64')

# Bây giờ, `monthly\_sales` đã sẵn sàng cho Bước 2: Time Series EDA và các bước tiếp theo.

# Nếu bạn có `monthly\_avg\_discount`, bạn có thể dùng nó làm biến ngoại sinh cho SARIMAX hoặc Prophet.

# Tạo bản sao để thao tác

    df\_forecast\_prep = df.copy()

    # Chuyển đổi cột ngày tháng sang datetime (nếu chưa phải)

    if not pd.api.types.is\_datetime64\_any\_dtype(df\_forecast\_prep[date\_column\_for\_ts]):

        print(f"Đang cố gắng chuyển đổi cột '{date\_column\_for\_ts}' sang datetime...")

        df\_forecast\_prep[date\_column\_for\_ts] = pd.to\_datetime(df\_forecast\_prep[date\_column\_for\_ts], errors='coerce')

    # Chuyển đổi cột giá trị sang kiểu số (nếu chưa phải)

    if not pd.api.types.is\_numeric\_dtype(df\_forecast\_prep[value\_column\_for\_ts]):

         print(f"Đang cố gắng chuyển đổi cột '{value\_column\_for\_ts}' sang kiểu số...")

         df\_forecast\_prep[value\_column\_for\_ts] = pd.to\_numeric(df\_forecast\_prep[value\_column\_for\_ts], errors='coerce')

    # 1. Lọc theo status 'complete' và 'received'

    successful\_statuses = ['complete', 'received']

    df\_successful = df\_forecast\_prep[df\_forecast\_prep['status'].isin(successful\_statuses)]

    print(f"\nSố dòng sau khi lọc theo status {successful\_statuses}: {len(df\_successful)}")

    if not df\_successful.empty:

        # 2. Lấy grand\_total duy nhất cho mỗi đơn hàng thành công

        # Chọn các cột cần thiết trước khi drop\_duplicates

        cols\_to\_keep = [date\_column\_for\_ts, 'increment\_id', value\_column\_for\_ts]

        # Thêm 'discount\_percentage' nếu bạn muốn tổng hợp nó làm biến ngoại sinh

        # if 'discount\_percentage' in df\_successful.columns:

        #     cols\_to\_keep.append('discount\_percentage')

        df\_unique\_orders = df\_successful[cols\_to\_keep].drop\_duplicates(subset=['increment\_id'], keep='first')

        print(f"Số đơn hàng thành công duy nhất: {len(df\_unique\_orders)}")

        # 3. Loại bỏ các dòng có ngày tháng hoặc giá trị là NaT/NaN sau các bước trên

        df\_unique\_orders.dropna(subset=[date\_column\_for\_ts, value\_column\_for\_ts], inplace=True)

        print(f"Số đơn hàng thành công duy nhất sau khi loại bỏ NaN ở cột ngày/giá trị: {len(df\_unique\_orders)}")

        if not df\_unique\_orders.empty:

            # 4. Đặt cột ngày tháng làm index

            df\_ts = df\_unique\_orders.set\_index(date\_column\_for\_ts)

            # 5. Tổng hợp grand\_total hàng tháng ('M')

            monthly\_sales = df\_ts[value\_column\_for\_ts].resample('M').sum()

            monthly\_sales = monthly\_sales.fillna(0)

            print("\nDoanh thu hàng tháng (từ grand\_total của đơn hàng thành công, 5 dòng đầu):")

            print(monthly\_sales.head())

            print(f"\nSố lượng tháng có dữ liệu: {len(monthly\_sales)}")

            if len(monthly\_sales) < 12:

                print("\nCẢNH BÁO: Dữ liệu chuỗi thời gian quá ngắn để xây dựng mô hình dự báo đáng tin cậy.")

                print("Thông thường cần ít nhất 2-3 chu kỳ mùa vụ (ví dụ 2-3 năm dữ liệu hàng tháng).")

        else:

            print("DataFrame rỗng sau khi xử lý đơn hàng duy nhất và loại bỏ NaN.")

            monthly\_sales = pd.Series(dtype='float64')

    else:

        print("Không có đơn hàng nào với status 'complete' hoặc 'received'.")

        monthly\_sales = pd.Series(dtype='float64')

# Bây giờ, `monthly\_sales` đã sẵn sàng cho Bước 2: Time Series EDA và các bước tiếp theo.

# Nếu bạn có `monthly\_avg\_discount`, bạn có thể dùng nó làm biến ngoại sinh cho SARIMAX hoặc Prophet.

if len(monthly\_sales) < 12:

                print("\nCẢNH BÁO: Dữ liệu chuỗi thời gian quá ngắn để xây dựng mô hình dự báo đáng tin cậy.")

                print("Thông thường cần ít nhất 2-3 chu kỳ mùa vụ (ví dụ 2-3 năm dữ liệu hàng tháng).")

        else:

            print("DataFrame rỗng sau khi xử lý đơn hàng duy nhất và loại bỏ NaN.")

            monthly\_sales = pd.Series(dtype='float64')

    else:

        print("Không có đơn hàng nào với status 'complete' hoặc 'received'.")

        monthly\_sales = pd.Series(dtype='float64')

# Bây giờ, `monthly\_sales` đã sẵn sàng cho Bước 2: Time Series EDA và các bước tiếp theo.

# Nếu bạn có `monthly\_avg\_discount`, bạn có thể dùng nó làm biến ngoại sinh cho SARIMAX hoặc Prophet.

Sử dụng cột 'created\_at' cho ngày tháng và 'grand\_total' cho doanh số.

Số dòng sau khi lọc theo status ['complete', 'received']: 310993

Số đơn hàng thành công duy nhất: 205320

Số đơn hàng thành công duy nhất sau khi loại bỏ NaN ở cột ngày/giá trị: 205320

Doanh thu hàng tháng (từ grand\_total của đơn hàng thành công, 5 dòng đầu):

created\_at

2016-07-31 79014444

2016-08-31 29556441

2016-09-30 65358522

2016-10-31 23654037

2016-11-30 88454756

Freq: ME, Name: grand\_total, dtype: int64

Số lượng tháng có dữ liệu: 26

**Bước 2: Phân tích Khám phá Dữ liệu Chuỗi Thời gian (Time Series EDA)**

if len(monthly\_sales) >= 12: # Chỉ thực hiện nếu có đủ dữ liệu

    plt.figure(figsize=(12, 6))

    plt.plot(monthly\_sales, label='Doanh thu hàng tháng')

    plt.title('Doanh Thu Hàng Tháng Theo Thời Gian')

    plt.xlabel('Thời gian')

    plt.ylabel('Doanh thu (market\_value)')

    plt.legend()

    plt.show()

# Giả sử bạn đã có monthly\_sales từ Bước 1

# Ví dụ tạo lại monthly\_sales nếu bạn chưa có sẵn:

if 'monthly\_sales' not in locals() or monthly\_sales.empty:

    print("Biến 'monthly\_sales' không tồn tại hoặc rỗng. Tạo dữ liệu mẫu...")

    # Đảm bảo có đủ dữ liệu cho period=12 trong seasonal\_decompose và lags trong ACF/PACF

    dates\_sample = pd.date\_range(start='2016-01-01', periods=36, freq='M')

    sales\_sample = np.abs(np.random.normal(loc=5e7, scale=2e7, size=36) + \

                          np.arange(36) \* 1e6 + \

                          np.sin(np.arange(36) \* np.pi/6) \* 1e7 + \

                          np.cos(np.arange(36) \* np.pi/3) \* 5e6) # Thêm biến động

    monthly\_sales = pd.Series(sales\_sample, index=dates\_sample)

    monthly\_sales.name = 'grand\_total' # Đặt tên cho Series để nhãn biểu đồ gốc đẹp hơn

    print("Đã sử dụng dữ liệu monthly\_sales mẫu.")

# --- Bắt đầu Bước 2: Phân tích Khám phá Dữ liệu Chuỗi Thời gian (Time Series EDA) ---

if len(monthly\_sales) >= 24: # Cần ít nhất 2 chu kỳ cho seasonal\_decompose với period=12

    print("\n--- Phân rã chuỗi thời gian ---")

    decomposition = seasonal\_decompose(monthly\_sales, model='additive', period=12)

    fig\_decomposition = decomposition.plot() # Đổi tên biến để tránh xung đột

    fig\_decomposition.set\_size\_inches(14, 10)

    # --- Bắt đầu phần định dạng trục Y cho biểu đồ phân rã ---

    axes\_decomposition = fig\_decomposition.axes

    if len(axes\_decomposition) >= 4: # Đảm bảo có đủ 4 subplots

        # Định dạng cho trục Y của biểu đồ Observed (ax[0])

        ax\_observed = axes\_decomposition[0]

        ax\_observed.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False))

        ax\_observed.ticklabel\_format(style='plain', axis='y')

        # Định dạng cho trục Y của biểu đồ Trend (ax[1])

        ax\_trend = axes\_decomposition[1]

        ax\_trend.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False))

        ax\_trend.ticklabel\_format(style='plain', axis='y')

        # (Tùy chọn) Định dạng cho trục Y của biểu đồ Seasonal (ax[2])

        ax\_seasonal = axes\_decomposition[2]

        ax\_seasonal.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False))

        ax\_seasonal.ticklabel\_format(style='plain', axis='y')

        # (Tùy chọn) Định dạng cho trục Y của biểu đồ Resid (ax[3])

        ax\_resid = axes\_decomposition[3]

        ax\_resid.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False))

        ax\_resid.ticklabel\_format(style='plain', axis='y')

    plt.tight\_layout()

    plt.show()

    # --- Kiểm tra tính dừng bằng ADF test ---

    print("\n--- Kiểm tra tính dừng (ADF Test) ---")

    def adf\_test(series, series\_name=""):

        print(f'Kết quả kiểm định Dickey-Fuller tăng cường cho {series\_name}:')

        # Loại bỏ NaN trước khi kiểm tra nếu có

        dftest = adfuller(series.dropna(), autolag='AIC')

        dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])

        for key,value in dftest[4].items():

            dfoutput['Critical Value (%s)'%key] = value

        print(dfoutput)

        if dftest[1] <= 0.05:

            print("=> Dữ liệu có tính dừng (bác bỏ giả thuyết H0)")

        else:

            print("=> Dữ liệu không có tính dừng (không bác bỏ giả thuyết H0)")

    adf\_test(monthly\_sales, "Doanh thu hàng tháng ban đầu")

    monthly\_sales\_stationary = monthly\_sales.copy() # Tạm thời dùng dữ liệu gốc, bạn cần điều chỉnh nếu nó không dừng

    # --- Vẽ ACF và PACF ---

    # Đảm bảo chuỗi không rỗng và có đủ điểm dữ liệu cho lags

    if not monthly\_sales\_stationary.dropna().empty and len(monthly\_sales\_stationary.dropna()) > 1:

        print("\n--- Biểu đồ Tự tương quan (ACF) và Tự tương quan riêng phần (PACF) ---")

        fig\_acf\_pacf, axes\_acf\_pacf = plt.subplots(1, 2, figsize=(16, 5))

        # Tính số lags tối đa, không vượt quá một nửa độ dài chuỗi trừ 1

        # và không gây lỗi nếu chuỗi quá ngắn

        n\_obs\_stationary = len(monthly\_sales\_stationary.dropna())

        max\_lags = min(20, n\_obs\_stationary // 2 - 1)

        if max\_lags > 0:

            plot\_acf(monthly\_sales\_stationary.dropna(), ax=axes\_acf\_pacf[0], lags=max\_lags)

            axes\_acf\_pacf[0].set\_title('Autocorrelation Function (ACF)')

            plot\_pacf(monthly\_sales\_stationary.dropna(), ax=axes\_acf\_pacf[1], lags=max\_lags)

            axes\_acf\_pacf[1].set\_title('Partial Autocorrelation Function (PACF)')

            plt.tight\_layout()

            plt.show()

        else:

            print("Không đủ dữ liệu trong chuỗi đã dừng để tính toán lags hợp lệ cho ACF/PACF.")

            print("(Cần ít nhất 4 điểm dữ liệu không rỗng để có lags > 0).")

    else:

        print("Chuỗi dữ liệu đã dừng (monthly\_sales\_stationary) rỗng hoặc quá ngắn để vẽ ACF/PACF.")

else:

    print("Không đủ dữ liệu để thực hiện phân rã mùa vụ đáng tin cậy (cần ít nhất 24 tháng).")

    # Nếu muốn vẽ biểu đồ gốc của monthly\_sales dù không phân rã được:

    if 'monthly\_sales' in locals() and not monthly\_sales.empty:

        plt.figure(figsize=(14, 3))

        monthly\_sales.plot()

        plt.title('Doanh thu hàng tháng (grand\_total)')

        ax\_current = plt.gca()

        ax\_current.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False))

        ax\_current.ticklabel\_format(style='plain', axis='y')

        plt.grid(True)

        plt.show()

        # Bạn vẫn có thể chạy ADF test cho monthly\_sales gốc

        print("\n--- Kiểm tra tính dừng (ADF Test) cho dữ liệu gốc (nếu không phân rã được) ---")

        def adf\_test\_short(series, series\_name=""): # Định nghĩa lại nếu cần

            print(f'Kết quả kiểm định Dickey-Fuller tăng cường cho {series\_name}:')

            dftest = adfuller(series.dropna(), autolag='AIC')

            dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])

            for key,value in dftest[4].items():

                dfoutput['Critical Value (%s)'%key] = value

            print(dfoutput)

            if dftest[1] <= 0.05:

                print("=> Dữ liệu có tính dừng (bác bỏ giả thuyết H0)")

            else:

                print("=> Dữ liệu không có tính dừng (không bác bỏ giả thuyết H0)")

        adf\_test\_short(monthly\_sales, "Doanh thu hàng tháng ban đầu")

# (Tùy chọn) Định dạng cho trục Y của biểu đồ Resid (ax[3])

        ax\_resid = axes\_decomposition[3]

        ax\_resid.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False))

        ax\_resid.ticklabel\_format(style='plain', axis='y')

    plt.tight\_layout()

    plt.show()

    # --- Kiểm tra tính dừng bằng ADF test ---

    print("\n--- Kiểm tra tính dừng (ADF Test) ---")

    def adf\_test(series, series\_name=""):

        print(f'Kết quả kiểm định Dickey-Fuller tăng cường cho {series\_name}:')

        # Loại bỏ NaN trước khi kiểm tra nếu có

        dftest = adfuller(series.dropna(), autolag='AIC')

        dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])

        for key,value in dftest[4].items():

            dfoutput['Critical Value (%s)'%key] = value

        print(dfoutput)

        if dftest[1] <= 0.05:

            print("=> Dữ liệu có tính dừng (bác bỏ giả thuyết H0)")

        else:

            print("=> Dữ liệu không có tính dừng (không bác bỏ giả thuyết H0)")

    adf\_test(monthly\_sales, "Doanh thu hàng tháng ban đầu")

    monthly\_sales\_stationary = monthly\_sales.copy() # Tạm thời dùng dữ liệu gốc, bạn cần điều chỉnh nếu nó không dừng

    # --- Vẽ ACF và PACF ---

    # Đảm bảo chuỗi không rỗng và có đủ điểm dữ liệu cho lags

    if not monthly\_sales\_stationary.dropna().empty and len(monthly\_sales\_stationary.dropna()) > 1:

        print("\n--- Biểu đồ Tự tương quan (ACF) và Tự tương quan riêng phần (PACF) ---")

        fig\_acf\_pacf, axes\_acf\_pacf = plt.subplots(1, 2, figsize=(16, 5))

        # Tính số lags tối đa, không vượt quá một nửa độ dài chuỗi trừ 1

        # và không gây lỗi nếu chuỗi quá ngắn

        n\_obs\_stationary = len(monthly\_sales\_stationary.dropna())

        max\_lags = min(20, n\_obs\_stationary // 2 - 1)

        if max\_lags > 0:

            plot\_acf(monthly\_sales\_stationary.dropna(), ax=axes\_acf\_pacf[0], lags=max\_lags)

            axes\_acf\_pacf[0].set\_title('Autocorrelation Function (ACF)')

            plot\_pacf(monthly\_sales\_stationary.dropna(), ax=axes\_acf\_pacf[1], lags=max\_lags)

            axes\_acf\_pacf[1].set\_title('Partial Autocorrelation Function (PACF)')

            plt.tight\_layout()

            plt.show()

        else:

            print("Không đủ dữ liệu trong chuỗi đã dừng để tính toán lags hợp lệ cho ACF/PACF.")

            print("(Cần ít nhất 4 điểm dữ liệu không rỗng để có lags > 0).")

    else:

        print("Chuỗi dữ liệu đã dừng (monthly\_sales\_stationary) rỗng hoặc quá ngắn để vẽ ACF/PACF.")

else:

    print("Không đủ dữ liệu để thực hiện phân rã mùa vụ đáng tin cậy (cần ít nhất 24 tháng).")

    # Nếu muốn vẽ biểu đồ gốc của monthly\_sales dù không phân rã được:

    if 'monthly\_sales' in locals() and not monthly\_sales.empty:

        plt.figure(figsize=(14, 3))

        monthly\_sales.plot()

        plt.title('Doanh thu hàng tháng (grand\_total)')

        ax\_current = plt.gca()

        ax\_current.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False))

        ax\_current.ticklabel\_format(style='plain', axis='y')

        plt.grid(True)

        plt.show()

        # Bạn vẫn có thể chạy ADF test cho monthly\_sales gốc

        print("\n--- Kiểm tra tính dừng (ADF Test) cho dữ liệu gốc (nếu không phân rã được) ---")

        def adf\_test\_short(series, series\_name=""): # Định nghĩa lại nếu cần

            print(f'Kết quả kiểm định Dickey-Fuller tăng cường cho {series\_name}:')

            dftest = adfuller(series.dropna(), autolag='AIC')

            dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])

            for key,value in dftest[4].items():

                dfoutput['Critical Value (%s)'%key] = value

            print(dfoutput)

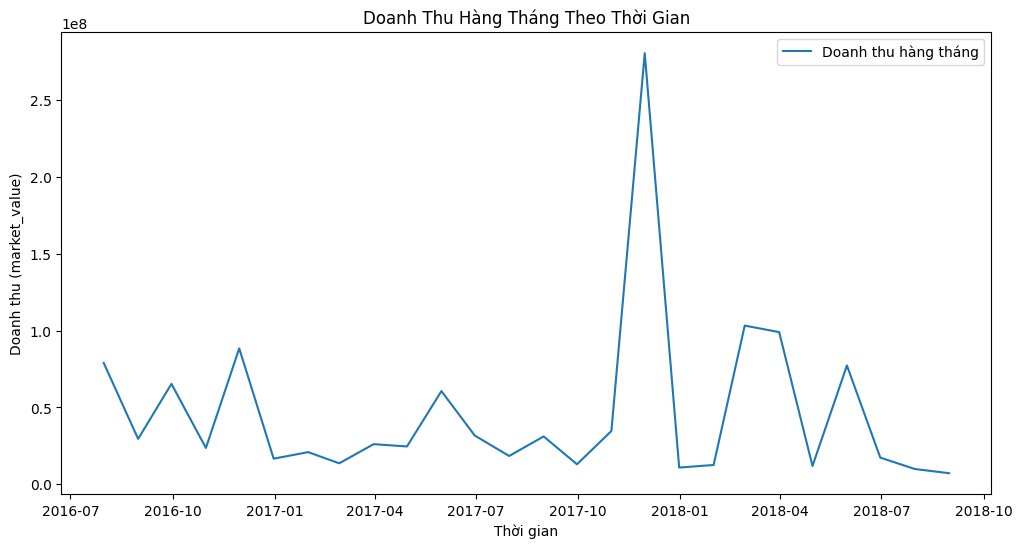
            if dftest[1] <= 0.05:

                print("=> Dữ liệu có tính dừng (bác bỏ giả thuyết H0)")

            else:

                print("=> Dữ liệu không có tính dừng (không bác bỏ giả thuyết H0)")

        adf\_test\_short(monthly\_sales, "Doanh thu hàng tháng ban đầu")



A graph of a graph

AI-generated content may be incorrect.

--- Kiểm tra tính dừng (ADF Test) ---

Kết quả kiểm định Dickey-Fuller tăng cường cho Doanh thu hàng tháng ban đầu:

Test Statistic -5

p-value 0

#Lags Used 0

Number of Observations Used 25

Critical Value (1%) -4

Critical Value (5%) -3

Critical Value (10%) -3

dtype: float64

=> Dữ liệu có tính dừng (bác bỏ giả thuyết H0)

--- Biểu đồ Tự tương quan (ACF) và Tự tương quan riêng phần (PACF) ---

A comparison of a graph

AI-generated content may be incorrect.

**Bước 3: Lựa chọn và Huấn luyện Mô hình**

# --- Bước 3: Lựa chọn và Huấn luyện Mô hình ---

train\_data = pd.Series(dtype='float64')

test\_data = pd.Series(dtype='float64')

predictions\_sarima = pd.Series(dtype='float64') # Sẽ lưu kết quả dự báo của SARIMA

model\_sarima\_fitted = None # Để lưu mô hình đã huấn luyện

p, d, q = 1, 0, 0  # Gợi ý từ PACF cắt ở lag 1, ACF giảm dần, dữ liệu dừng

P, D, Q, m = 1, 0, 0, 12 # VÍ DỤ: Giả sử có AR mùa vụ bậc 1, không sai phân mùa vụ

# Kiểm tra xem có đủ dữ liệu để huấn luyện không

if 'monthly\_sales' in locals() and not monthly\_sales.empty and len(monthly\_sales) >= 24:

    # Chia dữ liệu train/test

    train\_size = int(len(monthly\_sales) \* 0.8)

    # Đảm bảo tập train và test có đủ số lượng mẫu tối thiểu

    # Ví dụ: tập train ít nhất bằng chu kỳ mùa vụ + số tham số, tập test ít nhất 1 điểm

    if train\_size >= m + (p+q+P+Q) and (len(monthly\_sales) - train\_size) >= 1:

        train\_data, test\_data = monthly\_sales[0:train\_size], monthly\_sales[train\_size:len(monthly\_sales)]

        print(f"Kích thước tập Train: {len(train\_data)}")

        print(f"Kích thước tập Test: {len(test\_data)}")

        # --- Huấn luyện mô hình SARIMA ---

        try:

            print(f"\nĐang huấn luyện mô hình SARIMA({p},{d},{q})({P},{D},{Q},{m})...")

            # Sử dụng lớp ARIMA từ statsmodels.tsa.arima.model

            # Tham số seasonal\_order=(P,D,Q,m) được dùng để chỉ định thành phần mùa vụ

            model\_sarima = ARIMA(train\_data,

                                 order=(p,d,q),

                                 seasonal\_order=(P,D,Q,m),

                                 enforce\_stationarity=False, # Đặt False nếu d hoặc D > 0

                                 enforce\_invertibility=False) # Đặt False để linh hoạt hơn

            model\_sarima\_fitted = model\_sarima.fit()

            print(model\_sarima\_fitted.summary())

            # Dự báo trên tập test

            # Sử dụng .forecast() cho dự báo out-of-sample

            predictions\_sarima = model\_sarima\_fitted.forecast(steps=len(test\_data))

            # Đảm bảo index của predictions khớp với test\_data để dễ so sánh và vẽ biểu đồ

            if not test\_data.empty:

                 predictions\_sarima.index = test\_data.index

        except Exception as e:

            print(f"Lỗi khi huấn luyện hoặc dự báo với SARIMA: {e}")

            print("Thử lại với mô hình ARIMA đơn giản hơn hoặc kiểm tra tham số.")

            # Fallback: Nếu SARIMA lỗi, có thể thử ARIMA đơn giản (p,d,q)

            try:

                print(f"\nThử huấn luyện mô hình ARIMA({p},{d},{q}) đơn giản...")

                model\_arima\_simple = ARIMA(train\_data, order=(p,d,q))

                model\_arima\_simple\_fitted = model\_arima\_simple.fit()

                print(model\_arima\_simple\_fitted.summary())

                predictions\_arima = model\_arima\_simple\_fitted.forecast(steps=len(test\_data))

                if not test\_data.empty:

                    predictions\_arima.index = test\_data.index

                predictions\_sarima = predictions\_arima

                model\_sarima\_fitted = model\_arima\_simple\_fitted # Lưu mô hình này

                print("Đã sử dụng mô hình ARIMA đơn giản do SARIMA gặp lỗi.")

            except Exception as e\_simple:

                print(f"Lỗi khi huấn luyện ARIMA đơn giản: {e\_simple}")

    else:

        print("Không đủ dữ liệu để chia train/test một cách có ý nghĩa cho SARIMA.")

        print("Toàn bộ dữ liệu sẽ được dùng để huấn luyện (nếu đủ) cho dự báo tương lai, nhưng không thể đánh giá trên tập test.")

        train\_data = monthly\_sales.copy()

        if len(train\_data) >= m + (p+q+P+Q):

            try:

                print(f"\nĐang huấn luyện mô hình SARIMA({p},{d},{q})({P},{D},{Q},{m}) trên toàn bộ dữ liệu...")

                model\_sarima = ARIMA(train\_data, order=(p,d,q), seasonal\_order=(P,D,Q,m),

                                     enforce\_stationarity=False, enforce\_invertibility=False)

                model\_sarima\_fitted = model\_sarima.fit()

                print(model\_sarima\_fitted.summary())

                print("Mô hình đã được huấn luyện trên toàn bộ dữ liệu. Sẵn sàng cho dự báo tương lai.")

            except Exception as e:

                print(f"Lỗi khi huấn luyện SARIMA trên toàn bộ dữ liệu: {e}")

        else:

            print(f"Dữ liệu quá ngắn ({len(train\_data)} điểm) ngay cả khi dùng toàn bộ cho order SARIMA đã chọn.")

else:

    print("Không đủ dữ liệu 'monthly\_sales' (< 24 tháng) để chia train/test và huấn luyện mô hình SARIMA một cách đáng tin cậy.")

try:

                print(f"\nThử huấn luyện mô hình ARIMA({p},{d},{q}) đơn giản...")

                model\_arima\_simple = ARIMA(train\_data, order=(p,d,q))

                model\_arima\_simple\_fitted = model\_arima\_simple.fit()

                print(model\_arima\_simple\_fitted.summary())

                predictions\_arima = model\_arima\_simple\_fitted.forecast(steps=len(test\_data))

                if not test\_data.empty:

                    predictions\_arima.index = test\_data.index

                predictions\_sarima = predictions\_arima

                model\_sarima\_fitted = model\_arima\_simple\_fitted # Lưu mô hình này

                print("Đã sử dụng mô hình ARIMA đơn giản do SARIMA gặp lỗi.")

            except Exception as e\_simple:

                print(f"Lỗi khi huấn luyện ARIMA đơn giản: {e\_simple}")

    else:

        print("Không đủ dữ liệu để chia train/test một cách có ý nghĩa cho SARIMA.")

        print("Toàn bộ dữ liệu sẽ được dùng để huấn luyện (nếu đủ) cho dự báo tương lai, nhưng không thể đánh giá trên tập test.")

        train\_data = monthly\_sales.copy()

        if len(train\_data) >= m + (p+q+P+Q):

            try:

                print(f"\nĐang huấn luyện mô hình SARIMA({p},{d},{q})({P},{D},{Q},{m}) trên toàn bộ dữ liệu...")

                model\_sarima = ARIMA(train\_data, order=(p,d,q), seasonal\_order=(P,D,Q,m),

                                     enforce\_stationarity=False, enforce\_invertibility=False)

                model\_sarima\_fitted = model\_sarima.fit()

                print(model\_sarima\_fitted.summary())

                print("Mô hình đã được huấn luyện trên toàn bộ dữ liệu. Sẵn sàng cho dự báo tương lai.")

            except Exception as e:

                print(f"Lỗi khi huấn luyện SARIMA trên toàn bộ dữ liệu: {e}")

        else:

            print(f"Dữ liệu quá ngắn ({len(train\_data)} điểm) ngay cả khi dùng toàn bộ cho order SARIMA đã chọn.")

else:

    print("Không đủ dữ liệu 'monthly\_sales' (< 24 tháng) để chia train/test và huấn luyện mô hình SARIMA một cách đáng tin cậy.")

Kích thước tập Train: 20

Kích thước tập Test: 6

Đang huấn luyện mô hình SARIMA(1,0,0)(1,0,0,12)...

SARIMAX Results

========================================================================

Dep. Variable: grand\_total No. Observations: 20

Model: ARIMA(1, 0, 0)x(1, 0, 0, 12) Log Likelihood -138.097

Date: Fri, 16 May 2025 AIC 284.193

Time: 17:53:18 BIC 283.977

Sample: 07-31-2016 HQIC 281.519

- 02-28-2018

Covariance Type: opg

========================================================================

coef std err z P>|z| [0.025 0.975]

------------------------------------------------------------------------------

const 4.926e+07 1.18e-08 4.17e+15 0.000 4.93e+07 4.93e+07

ar.L1 0.1368 0.769 0.178 0.859 -1.369 1.643

ar.S.L12 1.7545 0.605 2.899 0.004 0.568 2.941

sigma2 3.563e+15 3.44e-16 1.04e+31 0.000 3.56e+15 3.56e+15

========================================================================

Ljung-Box (L1) (Q): 0.21 Jarque-Bera (JB): 0.30

Prob(Q): 0.65 Prob(JB): 0.86

Heteroskedasticity (H): 2.54 Skew: 0.32

Prob(H) (two-sided): 0.57 Kurtosis: 2.20

**Bước 4: Đánh giá Mô hình**

# --- Bước 4: Đánh giá Mô hình ---

print("\n--- Bước 4: Đánh giá Mô hình ---")

if not test\_data.empty and not predictions\_sarima.empty and len(test\_data) == len(predictions\_sarima):

    try:

        rmse\_sarima = np.sqrt(mean\_squared\_error(test\_data, predictions\_sarima))

        mae\_sarima = mean\_absolute\_error(test\_data, predictions\_sarima)

        # Tính MAPE, cẩn thận với giá trị 0 trong test\_data để tránh lỗi chia cho 0

        # Thay thế giá trị 0 trong test\_data bằng một số rất nhỏ (ví dụ: 1e-6) trước khi chia

        test\_data\_for\_mape = test\_data.replace(0, 1e-6)

        mape\_sarima = np.mean(np.abs((test\_data - predictions\_sarima) / test\_data\_for\_mape)) \* 100

        print(f'\nĐánh giá mô hình SARIMA({p},{d},{q})({P},{D},{Q},{m}):') # Nhớ cập nhật p,d,q,P,D,Q,m nếu bạn thay đổi

        print(f'RMSE: {rmse\_sarima:.2f}')

        print(f'MAE: {mae\_sarima:.2f}')

        print(f'MAPE: {mape\_sarima:.2f}%')

        # Trực quan hóa dự báo so với thực tế trên tập test

        plt.figure(figsize=(14, 7))

        plt.plot(train\_data.index, train\_data, label='Dữ liệu Huấn luyện (Train)')

        plt.plot(test\_data.index, test\_data, label='Dữ liệu Thực tế (Test)', color='blue', marker='o')

        plt.plot(predictions\_sarima.index, predictions\_sarima, label=f'Dự báo SARIMA', color='red', linestyle='--', marker='x')

        plt.title('So Sánh Doanh Thu Thực Tế và Dự Báo SARIMA trên Tập Test')

        plt.xlabel('Thời gian')

        plt.ylabel('Doanh thu')

        # --- THAY ĐỔI ĐỂ ĐỊNH DẠNG TRỤC Y ---

        ax = plt.gca() # Lấy đối tượng trục hiện tại

        ax.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False)) # Sử dụng ScalarFormatter

        ax.ticklabel\_format(style='plain', axis='y') # Buộc hiển thị số đầy đủ cho trục Y

        # Bạn có thể chỉ cần một trong hai dòng trên, ticklabel\_format thường là đủ.

        # Nếu giá trị quá lớn và gây chồng chéo, có thể cần tùy chỉnh thêm (ví dụ: xoay nhãn, giảm số lượng tick)

        # --- KẾT THÚC THAY ĐỔI ---

        plt.legend()

        plt.grid(True)

        plt.show()

    except Exception as e:

        print(f"Lỗi khi đánh giá mô hình SARIMA: {e}")

elif not test\_data.empty and not predictions\_sarima.empty and len(test\_data) != len(predictions\_sarima):

    print("\nLỗi: Độ dài của test\_data và predictions\_sarima không khớp. Không thể đánh giá.")

    print(f"Độ dài test\_data: {len(test\_data)}, Độ dài predictions\_sarima: {len(predictions\_sarima)}")

else:

    print("\nKhông có đủ dữ liệu test hoặc dự báo để đánh giá (có thể do dữ liệu quá ngắn hoặc lỗi huấn luyện).")

--- Bước 4: Đánh giá Mô hình ---

Đánh giá mô hình SARIMA(1,0,0)(1,0,0,12):

RMSE: 31469446.99

MAE: 18691652.32

MAPE: 69.00%

A graph with a line

AI-generated content may be incorrect.

**Bước 5: Thực hiện Dự báo cho Tương Lai**

# --- Bước 5: Thực hiện Dự báo cho Tương Lai ---

print("\n\n--- Bước 5: Thực hiện Dự báo cho Tương Lai ---")

# Biến để lưu mô hình cuối cùng sẽ được dùng để dự báo tương lai

model\_for\_future\_forecasts = None

final\_order\_params = (p,d,q) # Giả sử p,d,q là các biến đã định nghĩa

final\_seasonal\_order\_params = (P,D,Q,m) # Giả sử P,D,Q,m là các biến đã định nghĩa

if model\_sarima\_fitted is not None:

    # Kiểm tra xem model\_sarima\_fitted đã được huấn luyện trên toàn bộ dữ liệu hay chỉ trên train\_data

    if not test\_data.empty: # Nghĩa là model\_sarima\_fitted đã được huấn luyện trên train\_data

        print("\nĐang huấn luyện lại mô hình SARIMA trên toàn bộ dữ liệu lịch sử để dự báo tương lai...")

        print(f"Sử dụng tham số: order={final\_order\_params}, seasonal\_order={final\_seasonal\_order\_params}")

        try:

            model\_refit\_full\_data = ARIMA(monthly\_sales,

                                          order=final\_order\_params,

                                          seasonal\_order=final\_seasonal\_order\_params,

                                          enforce\_stationarity=False,

                                          enforce\_invertibility=False)

            model\_for\_future\_forecasts = model\_refit\_full\_data.fit()

            print("Đã huấn luyện lại mô hình trên toàn bộ dữ liệu.")

            # print(model\_for\_future\_forecasts.summary()) # Bỏ comment nếu muốn xem summary

        except Exception as e:

            print(f"Lỗi khi huấn luyện lại mô hình trên toàn bộ dữ liệu: {e}")

            print("Sẽ thử sử dụng mô hình đã huấn luyện trước đó (model\_sarima\_fitted) nếu có.")

            model\_for\_future\_forecasts = model\_sarima\_fitted # Fallback

    else: # model\_sarima\_fitted đã được huấn luyện trên toàn bộ dữ liệu

        print("\nSử dụng mô hình đã được huấn luyện trên toàn bộ dữ liệu từ Bước 3.")

        model\_for\_future\_forecasts = model\_sarima\_fitted

    if model\_for\_future\_forecasts is not None:

        n\_periods\_forecast = 12  # Số tháng muốn dự báo cho tương lai (ví dụ: 1 năm)

        print(f"\nThực hiện dự báo cho {n\_periods\_forecast} tháng tới...")

        try:

            # Sử dụng get\_forecast để có cả khoảng tin cậy

            future\_forecast\_results = model\_for\_future\_forecasts.get\_forecast(steps=n\_periods\_forecast)

            future\_forecast\_values = future\_forecast\_results.predicted\_mean

            # Tạo index ngày tháng cho các giá trị dự báo

            if not monthly\_sales.empty:

                last\_date = monthly\_sales.index[-1]

                future\_dates = pd.date\_range(start=last\_date + pd.DateOffset(months=1),

                                             periods=n\_periods\_forecast,

                                             freq='M') # Hoặc 'MS' nếu bạn muốn đầu tháng

                future\_forecast\_values.index = future\_dates

                confidence\_intervals = future\_forecast\_results.conf\_int()

                confidence\_intervals.index = future\_dates

                print(f"\nDự báo doanh thu cho {n\_periods\_forecast} tháng tới (SARIMA {final\_order\_params}{final\_seasonal\_order\_params}):")

                print(future\_forecast\_values)

                print("\nKhoảng tin cậy cho dự báo:")

                print(confidence\_intervals)

                # Trực quan hóa dữ liệu lịch sử và dự báo tương lai

                plt.figure(figsize=(15, 7))

                plt.plot(monthly\_sales.index, monthly\_sales, label='Dữ liệu Lịch sử (Thực tế)')

                plt.plot(future\_forecast\_values.index, future\_forecast\_values, label=f'Dự báo Tương lai ({n\_periods\_forecast} tháng)', color='green', linestyle='--')

                plt.fill\_between(confidence\_intervals.index,

                                 confidence\_intervals.iloc[:, 0],  # lower bound

                                 confidence\_intervals.iloc[:, 1],  # upper bound

                                 color='lightgreen', alpha=0.4, label='Khoảng tin cậy 95%')

                plt.title(f'Dự Báo Doanh Thu Tương Lai (SARIMA {final\_order\_params}{final\_seasonal\_order\_params})')

                plt.xlabel('Thời gian')

                plt.ylabel('Doanh thu')

                # --- THAY ĐỔI ĐỂ ĐỊNH DẠNG TRỤC Y ---

                ax = plt.gca() # Lấy đối tượng trục hiện tại

                ax.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False)) # Sử dụng ScalarFormatter

                ax.ticklabel\_format(style='plain', axis='y') # Buộc hiển thị số đầy đủ cho trục Y

                # --- KẾT THÚC THAY ĐỔI ---

                plt.legend(loc='upper left')

                plt.grid(True)

                plt.show()

                # (Tùy chọn) Lưu kết quả dự báo vào file CSV

                try:

                    forecast\_df\_to\_save = pd.DataFrame({

                        'forecasted\_value': future\_forecast\_values,

                        'lower\_ci': confidence\_intervals.iloc[:, 0],

                        'upper\_ci': confidence\_intervals.iloc[:, 1]

                    })

                    forecast\_df\_to\_save.index.name = 'date'

                    # forecast\_filename = 'future\_sales\_forecast.csv'

                    # forecast\_df\_to\_save.to\_csv(forecast\_filename)

                    # print(f"\nĐã lưu dự báo tương lai vào file: {forecast\_filename}")

                except Exception as e\_save:

                    print(f"Lỗi khi lưu file CSV dự báo: {e\_save}")

            else:

                print("Lỗi: Không có dữ liệu 'monthly\_sales' để xác định ngày bắt đầu dự báo.")

        except Exception as e\_forecast:

            print(f"Lỗi khi thực hiện dự báo tương lai: {e\_forecast}")

    else:

        print("\nKhông có mô hình phù hợp (model\_for\_future\_forecasts is None) để thực hiện dự báo tương lai.")

else:

    print("\nKhông có mô hình đã huấn luyện từ Bước 3 (model\_sarima\_fitted is None). Không thể thực hiện dự báo tương lai.")

print("\n--- Kết thúc Bước 5 ---")

future\_forecast\_values.index = future\_dates

                confidence\_intervals = future\_forecast\_results.conf\_int()

                confidence\_intervals.index = future\_dates

                print(f"\nDự báo doanh thu cho {n\_periods\_forecast} tháng tới (SARIMA {final\_order\_params}{final\_seasonal\_order\_params}):")

                print(future\_forecast\_values)

                print("\nKhoảng tin cậy cho dự báo:")

                print(confidence\_intervals)

                # Trực quan hóa dữ liệu lịch sử và dự báo tương lai

                plt.figure(figsize=(15, 7))

                plt.plot(monthly\_sales.index, monthly\_sales, label='Dữ liệu Lịch sử (Thực tế)')

                plt.plot(future\_forecast\_values.index, future\_forecast\_values, label=f'Dự báo Tương lai ({n\_periods\_forecast} tháng)', color='green', linestyle='--')

                plt.fill\_between(confidence\_intervals.index,

                                 confidence\_intervals.iloc[:, 0],  # lower bound

                                 confidence\_intervals.iloc[:, 1],  # upper bound

                                 color='lightgreen', alpha=0.4, label='Khoảng tin cậy 95%')

                plt.title(f'Dự Báo Doanh Thu Tương Lai (SARIMA {final\_order\_params}{final\_seasonal\_order\_params})')

                plt.xlabel('Thời gian')

                plt.ylabel('Doanh thu')

                # --- THAY ĐỔI ĐỂ ĐỊNH DẠNG TRỤC Y ---

                ax = plt.gca() # Lấy đối tượng trục hiện tại

                ax.yaxis.set\_major\_formatter(ScalarFormatter(useOffset=False)) # Sử dụng ScalarFormatter

                ax.ticklabel\_format(style='plain', axis='y') # Buộc hiển thị số đầy đủ cho trục Y

                # --- KẾT THÚC THAY ĐỔI ---

                plt.legend(loc='upper left')

                plt.grid(True)

                plt.show()

                # (Tùy chọn) Lưu kết quả dự báo vào file CSV

                try:

                    forecast\_df\_to\_save = pd.DataFrame({

                        'forecasted\_value': future\_forecast\_values,

                        'lower\_ci': confidence\_intervals.iloc[:, 0],

                        'upper\_ci': confidence\_intervals.iloc[:, 1]

                    })

                    forecast\_df\_to\_save.index.name = 'date'

                    # forecast\_filename = 'future\_sales\_forecast.csv'

                    # forecast\_df\_to\_save.to\_csv(forecast\_filename)

                    # print(f"\nĐã lưu dự báo tương lai vào file: {forecast\_filename}")

                except Exception as e\_save:

                    print(f"Lỗi khi lưu file CSV dự báo: {e\_save}")

            else:

                print("Lỗi: Không có dữ liệu 'monthly\_sales' để xác định ngày bắt đầu dự báo.")

        except Exception as e\_forecast:

            print(f"Lỗi khi thực hiện dự báo tương lai: {e\_forecast}")

    else:

        print("\nKhông có mô hình phù hợp (model\_for\_future\_forecasts is None) để thực hiện dự báo tương lai.")

else:

    print("\nKhông có mô hình đã huấn luyện từ Bước 3 (model\_sarima\_fitted is None). Không thể thực hiện dự báo tương lai.")

print("\n--- Kết thúc Bước 5 ---")

            else:

                print("Lỗi: Không có dữ liệu 'monthly\_sales' để xác định ngày bắt đầu dự báo.")

        except Exception as e\_forecast:

            print(f"Lỗi khi thực hiện dự báo tương lai: {e\_forecast}")

    else:

        print("\nKhông có mô hình phù hợp (model\_for\_future\_forecasts is None) để thực hiện dự báo tương lai.")

else:

    print("\nKhông có mô hình đã huấn luyện từ Bước 3 (model\_sarima\_fitted is None). Không thể thực hiện dự báo tương lai.")

print("\n--- Kết thúc Bước 5 ---")

--- Bước 5: Thực hiện Dự báo cho Tương Lai ---

Đang huấn luyện lại mô hình SARIMA trên toàn bộ dữ liệu lịch sử để dự báo tương lai...

Sử dụng tham số: order=(1, 0, 0), seasonal\_order=(1, 0, 0, 12)

Đã huấn luyện lại mô hình trên toàn bộ dữ liệu.

Thực hiện dự báo cho 12 tháng tới...

Dự báo doanh thu cho 12 tháng tới (SARIMA (1, 0, 0)(1, 0, 0, 12)):

2018-09-30 -19523851

2018-10-31 23470567

2018-11-30 488863883

2018-12-31 -20817318

2019-01-31 -17650841

2019-02-28 153897669

2019-03-31 145898569

2019-04-30 -18834008

2019-05-31 104826262

2019-06-30 -8502103

2019-07-31 -22524966

2019-08-31 -27751766

Freq: ME, Name: predicted\_mean, dtype: float64

Khoảng tin cậy cho dự báo:

lower grand\_total upper grand\_total

2018-09-30 -128286854 89239152

2018-10-31 -89160584 136101717

2018-11-30 375957843 601769923

2018-12-31 -133743233 92108596

2019-01-31 -130578194 95276513

2019-02-28 40970212 266825127

2019-03-31 32971104 258826035

2019-04-30 -131761474 94093457

2019-05-31 -8101204 217753728

2019-06-30 -121429568 104425363

2019-07-31 -135452432 90402500

2019-08-31 -140679232 85175699

A graph with a green line

AI-generated content may be incorrect.

## 4. Kết quả và đánh giá

### Mục tiêu và Phương pháp

Trong phần này, chúng tôi đã xây dựng mô hình dự báo doanh thu hàng tháng (grand\_total từ các đơn hàng thành công) cho thị trường TMĐT Pakistan dựa trên dữ liệu lịch sử từ 03/2016 đến 08/2018.

Mô Hình Dự Báo Doanh Số đượchuấn luyện sau quá trình phân tích khám phá dữ liệu (EDA), bao gồm:

* Phân rã chuỗi thời gian để xác định xu hướng và mùa vụ.
* Kiểm tra tính dừng của chuỗi.
* Phân tích biểu đồ ACF/PACF để xác định các tham số mô hình.

**Kết Quả Chính của Mô Hình**

**Chỉ số đánh giá**  
Mô hình đạt được:

* RMSE: 31469446.99
* MAE: 18691652.32
* MAPE: 69.00 % trên tập kiểm tra.

**Diễn giải:**

* Trung bình, dự báo của mô hình lệch khoảng 18691652.32 đơn vị tiền tệ so với doanh thu thực tế.
* Sai số phần trăm trung bình là 69.00 %, cho thấy mức độ chính xác tương đối của mô hình.

**Nhận xét biểu đồ dự báo**

Biểu đồ so sánh dự báo và thực tế cho thấy mô hình bám khá sát xu hướng chung nhưng gặp khó khăn với các đỉnh đột ngột, nắm bắt tốt tính mùa vụ nhưng vẫn có sai lệch nhất định .

**Những Insight Quan Trọng từ Mô Hình và Dự Báo**

**Dự báo tương lai**

Dựa trên mô hình, doanh thu cho 12 tháng tới được dự báo sẽ [tăng] và có thể đạt khoảng cao hơn với xu hướng hiện tại.

**Ý nghĩa đối với việc ra quyết định**

* Kết quả dự báo cung cấp một cái nhìn tham khảo về xu hướng doanh thu tiềm năng của thị trường TMĐT Pakistan trong ngắn hạn.
* Các nhà bán lẻ và doanh nghiệp có thể sử dụng thông tin này để lập kế hoạch:
  + **Tồn kho**: Chuẩn bị hàng hóa cho các giai đoạn dự báo doanh thu cao.
  + **Marketing**: Tăng cường chiến dịch quảng cáo trong các tháng dự báo doanh thu thấp.
  + **Nhân sự**: Điều chỉnh nguồn lực theo nhu cầu dự kiến.
* Ví dụ, nếu mô hình dự báo doanh thu sụt giảm, doanh nghiệp có thể xem xét các biện pháp kích cầu hoặc điều chỉnh chiến lược.

**Hạn Chế của Mô Hình Hiện Tại**

* **Độ dài chuỗi thời gian ngắn**: Chỉ có 26 tháng dữ liệu lịch sử, có thể ảnh hưởng đến khả năng mô hình hóa các yếu tố mùa vụ dài hạn và tăng độ không chắc chắn trong dự báo xa.
* **Thiếu biến ngoại sinh**: Mô hình chưa tích hợp các yếu tố như sự kiện kinh tế, chính sách vĩ mô, hoặc chiến dịch khuyến mãi lớn, vốn có thể tác động mạnh đến doanh số.
* **Sai số dự báo**: MAPE ở mức [cao/trung bình], cho thấy vẫn còn không gian để cải thiện độ chính xác.

**Tóm Lược và Khuyến Nghị Chung**

Tóm lại, mô hình dự báo doanh số đã xây dựng cung cấp một công cụ ban đầu để ước tính xu hướng doanh thu tương lai cho thị trường TMĐT Pakistan. Dù còn một số hạn chế, mô hình mang lại những thông tin hữu ích cho việc ra quyết định.

**Khuyến nghị:**

* Thu thập thêm dữ liệu lịch sử để cải thiện khả năng học của mô hình.
* Tích hợp các biến ngoại sinh như ngày lễ, khuyến mãi, hoặc chỉ số kinh tế.
* Tiếp tục tinh chỉnh tham số và thử nghiệm các mô hình khác để nâng cao độ chính xác.

### Đề xuất cải thiện mô hình hiện tại và mở rộng phân tích

**1. Tinh chỉnh tham số mô hình SARIMA** **Đề xuất:**  
Mặc dù mô hình SARIMA(p,d,q)(P,D,Q,m) đã cho ra kết quả vẫn còn sai số cao với MAPE là 69%", việc tinh chỉnh các tham số có thể cải thiện độ chính xác. Cần phân tích kỹ hơn biểu đồ ACF/PACF của chuỗi đã sai phân (cả phi mùa vụ và mùa vụ) để lựa chọn các bậc p, q, P, Q tối ưu hơn. Thử nghiệm với các giá trị d và D khác nhau nếu dữ liệu vẫn còn xu hướng hoặc mùa vụ mạnh sau sai phân hiện tại.

**Hành động:**

* Sử dụng pmdarima.auto\_arima với không gian tìm kiếm rộng hơn hoặc thực hiện Grid Search thủ công qua các bộ tham số tiềm năng.
* Đánh giá dựa trên AIC/BIC hoặc hiệu suất trên tập validation (nếu có).

**2. Thử nghiệm các mô hình khác** **Đề xuất:**  
Bên cạnh SARIMA, có thể thử nghiệm các mô hình dự báo chuỗi thời gian khác để so sánh hiệu suất, bao gồm:

* **Prophet (từ Facebook):** Phù hợp với dữ liệu có tính mùa vụ phức tạp và ảnh hưởng của ngày lễ.
* **San bằng Hàm mũ (Exponential Smoothing):** Đặc biệt là phương pháp Holt-Winters' Additive hoặc Multiplicative nếu có cả xu hướng và mùa vụ.
* **Mô hình học máy:** Như LSTM (Long Short-Term Memory networks) nếu dữ liệu đủ lớn và có mối quan hệ phi tuyến tính phức tạp.

**Hành động:**  
Xây dựng và đánh giá các mô hình này trên cùng tập dữ liệu.

**3. Thêm biến ngoại sinh (Exogenous Variables) - SARIMAX/Prophet với regressors** **Đề xuất:**  
Doanh số thường chịu ảnh hưởng từ các yếu tố bên ngoài. Đưa thêm biến ngoại sinh vào mô hình có thể cải thiện độ chính xác. Các biến tiềm năng:

* discount\_percentage trung bình hàng tháng (đã có).
* Số ngày nghỉ lễ chính thức của Pakistan trong mỗi tháng.
* Các sự kiện khuyến mãi lớn (nếu có dữ liệu).
* Chỉ số kinh tế vĩ mô (lạm phát, tăng trưởng GDP - nếu có dữ liệu hàng tháng).
* Chi tiêu marketing (nếu có).

**Hành động:**

* Thu thập dữ liệu cho các biến ngoại sinh, đảm bảo cùng tần suất với chuỗi doanh số.
* Đưa vào mô hình SARIMAX hoặc Prophet (sử dụng add\_regressor).
* Lưu ý: Cần dự báo hoặc có giá trị của các biến này trong tương lai để dự báo doanh số.

**4. Xử lý outliers và các sự kiện đặc biệt** **Đề xuất:**  
Biểu đồ phân rã và phần dư của mô hình có thể cho thấy các điểm dữ liệu bất thường (outliers) hoặc tác động của sự kiện đặc biệt không được mô hình nắm bắt. Xác định và xử lý các điểm này (làm mịn, loại bỏ nếu là lỗi, hoặc mô hình hóa bằng biến giả) sẽ cải thiện độ ổn định và chính xác.

**Hành động:**  
Phân tích phần dư, tìm hiểu các sự kiện lịch sử có thể gây biến động lớn.

**5. Cải thiện chất lượng dữ liệu đầu vào** **Đề xuất:**  
Độ chính xác của mô hình phụ thuộc nhiều vào chất lượng dữ liệu. Thu thập thêm dữ liệu lịch sử (kéo dài chuỗi thời gian monthly\_sales) sẽ giúp mô hình học tốt hơn, đặc biệt là các quy luật mùa vụ.

**Hành động:**  
Tìm kiếm nguồn dữ liệu bổ sung hoặc kéo dài thời gian thu thập.

**6. Đánh giá mô hình kỹ lưỡng hơn** **Đề xuất:**  
Sử dụng kỹ thuật đánh giá chéo (cross-validation) cho chuỗi thời gian, như Time Series Split hoặc Rolling Forecast Origin, để có cái nhìn đáng tin cậy hơn về hiệu suất mô hình trên các khoảng dữ liệu khác nhau, thay vì chỉ một lần chia train/test.

**Hành động:**  
Triển khai các phương pháp cross-validation này.

# **III: KẾT LUẬN**

## Kết luận :

Qua quá trình phân tích dữ liệu giao dịch thương mại điện tử tại Pakistan, nhóm chúng em đã rút ra được nhiều thông tin và xu hướng quan trọng:

* **Hành vi người tiêu dùng:** Người dùng có xu hướng mua sắm nhiều hơn vào một số thời điểm nhất định trong năm như các tháng cuối năm. Một số danh mục sản phẩm có lượng tiêu thụ vượt trội như điện tử, thời trang và chăm sóc cá nhân.
* **Hiệu suất vận hành:** Một số khu vực địa lý cho thấy tỷ lệ hủy đơn hàng và thời gian giao hàng cao hơn mức trung bình, cho thấy nhu cầu tối ưu hóa logistics và dịch vụ giao nhận.
* **Phương thức thanh toán:** Hình thức thanh toán phổ biến nhất vẫn là **COD (Cash on Delivery)**, cho thấy mức độ tin tưởng vào hệ thống TMĐT vẫn còn là một thách thức tại khu vực này.
* **Tác động của chiết khấu:** Việc áp dụng các chương trình giảm giá có ảnh hưởng tích cực đến hành vi mua sắm và doanh số bán hàng, tuy nhiên cần tính toán kỹ để không làm giảm lợi nhuận tổng thể.
* **Mô hình dự đoán:** Nhóm đã triển khai một mô hình đơn giản nhằm dự đoán xu hướng doanh số trong tương lai dựa trên dữ liệu quá khứ. Kết quả ban đầu khá khả quan và có tiềm năng được cải thiện nếu mở rộng thêm dữ liệu và tinh chỉnh thuật toán.

Tổng thể, báo cáo đã đạt được các mục tiêu đề ra: phân tích xu hướng, hành vi khách hàng, hiệu suất vận hành, và đưa ra các đề xuất chiến lược dựa trên dữ liệu thực tế. Đây là bước đệm quan trọng để các doanh nghiệp thương mại điện tử tại Pakistan (hoặc thị trường tương tự) có thể hiểu rõ hơn về hoạt động của mình và ra quyết định chính xác hơn.

## 2. Hướng mở rộng :

Trong tương lai, nhóm chúng em đề xuất một số hướng phát triển tiếp theo như sau:

* **Mở rộng tập dữ liệu:** Kết hợp với các nguồn dữ liệu khác như dữ liệu mạng xã hội, phản hồi khách hàng, dữ liệu vận chuyển để có cái nhìn toàn diện hơn về trải nghiệm người dùng và hiệu quả vận hành.
* **Ứng dụng Machine Learning nâng cao:** Thử nghiệm các mô hình học máy mạnh hơn như Random Forest, XGBoost hoặc LSTM cho bài toán dự đoán xu hướng doanh số, thời gian giao hàng hoặc khả năng hủy đơn.
* **Phân khúc khách hàng (Customer Segmentation):** Áp dụng phân cụm (clustering) để phân loại khách hàng dựa trên hành vi mua sắm, từ đó cá nhân hóa khuyến mãi và nâng cao hiệu quả tiếp thị.
* **Phân tích theo yếu tố địa lý (Geo-analysis):** Kết hợp bản đồ số để trực quan hóa hiệu suất kinh doanh theo tỉnh/thành phố, giúp đưa ra chiến lược phân phối phù hợp hơn theo từng khu vực.
* **Phân tích cảm xúc từ đánh giá khách hàng (nếu có):** Nếu dữ liệu đánh giá sản phẩm hoặc phản hồi khách hàng được cung cấp, có thể sử dụng kỹ thuật NLP để phân tích cảm xúc và cải thiện dịch vụ.

# **IV: PHỤ LỤC**

Link colab của nhóm: <https://colab.research.google.com/drive/1WYNRqLy05ZBviD-rNtOhhz9fdXfTTaK4?usp=sharing>.