**LỜI CẢM ƠN**

Đến nay, khi đã hoàn thành báo cáo môn Phân Tích Dữ Liệu, nhóm em xin gửi lời cảm ơn chân thành nhất đến Trường Đại học Sư phạm Kỹ thuật Thành phố Hồ Chí Minh đã đưa môn học này vào chương trình giảng dạy của chúng em.

Đặc biệt, nhóm em xin gửi lời cảm ơn sâu sắc đến thầy Nguyễn Văn Thành, người đã không ngại khó khăn, luôn tận tình dạy dỗ và truyền đạt những kiến thức thú vị và giàu tính thực tế, ứng dụng trong cuộc sống cho chúng em. Nhờ có sự trợ giúp của thầy, chúng em đã hiểu được những kiến thức nền tảng và phương hướng giải quyết vấn đề. Đó thực sự là hành trang cần thiết mà chúng em nghĩ bất kỳ sinh viên nào cũng muốn có được.

Tuy nhiên, dù đã cố gắng bằng tất cả những gì chúng em có được, với vốn kiến thức còn nhiều hạn chế và kinh nghiệm thực hành non nớt, chúng em không thể tránh khỏi thiếu sót và còn nhiều chỗ chưa chính xác trong báo cáo. Vì thế, chúng em kính mong được thầy xem xét và góp ý để chúng em có thể ngày một cải thiện và hoàn thiện tốt nhất báo cáo này. Cuối cùng, chúng em xin chúc thầy và mọi người sẽ gặt hái được nhiều thành công trên suốt hành trình sự nghiệp mai này.

Chúng em xin chân thành cảm ơn!

**Nhận xét của giáo viên:**

......................................................................................................................

......................................................................................................................

......................................................................................................................

......................................................................................................................

......................................................................................................................

......................................................................................................................

*Ngày ......... tháng ......... năm .........*

*Giáo viên chấm điểm*

**MỤC LỤC**

[**1. Giới thiệu** 1](#_Toc168605312)

[2. Chuẩn bị dữ liệu 1](#_Toc168605313)

[3. Khai phá dữ liệu 5](#_Toc168605314)

[3.1. Quốc gia 5](#_Toc168605315)

[3.2. Khách hàng và Sản phẩm 6](#_Toc168605316)

[4. Thông tin chi tiết về các danh mục sản phẩm 12](#_Toc168605317)

[4.1. Những sản phẩm được mua nhiều nhất 12](#_Toc168605318)

[4.2. Mô tả sản phẩm - Description 12](#_Toc168605319)

[**4.3. Xác định danh mục sản phẩm** 15](#_Toc168605320)

[5. Phân loại khách hàng 19](#_Toc168605321)

[5.1. Định dạng dữ liệu 19](#_Toc168605322)

[5.1.1. Nhóm các sản phẩm 20](#_Toc168605323)

[5.1.2. Tách dữ liệu theo thời gian 20](#_Toc168605324)

[5.1.3. Kết hợp đơn hàng của người tiêu dùng 21](#_Toc168605325)

[5.2. Tạo danh mục khách hàng 21](#_Toc168605326)

[5.2.1. Mã hoá - định dạng dữ liệu 21](#_Toc168605327)

[5.2.2. Tạo danh mục khách hàng 23](#_Toc168605328)

[5.3. Phân tích hành vi mua hàng 27](#_Toc168605329)

[6. Phân tích và dự đoán doanh thu 29](#_Toc168605330)

[6.1. Phân tích doanh thu 31](#_Toc168605331)

[6.1.1. Doanh thu hằng tháng 31](#_Toc168605332)

[6.1.2. Doanh số bán hàng diễn ra vào ngày trong tuần 31](#_Toc168605333)

[6.1.3. Doanh số bán hàng diễn ra vào thời điểm trong ngày 32](#_Toc168605334)

[6.1.4. Tổng doanh thu theo ngày trong tháng 32](#_Toc168605335)

[6.2. Phân tích và dự đoán doanh thu 33](#_Toc168605336)

[7. Sử dụng các model để phân loại khách hàng 35](#_Toc168605337)

[7.1. Support Vector Machine 36](#_Toc168605338)

[7.2. k-Nearest Neighbors 36](#_Toc168605339)

[7.3. Decision Tree 36](#_Toc168605340)

[7.4. Random Forest 37](#_Toc168605341)

[8. Kết luận 37](#_Toc168605342)

**1. Giới thiệu**

**Bối cảnh:**

Thường thì các bộ dữ liệu thương mại điện tử là tài sản riêng và do đó khó tìm thấy trong số các dữ liệu có sẵn công khai. Tuy nhiên, Bộ lưu trữ Học máy của Đại học California, Irvine ([UCI Machine Learning Repository](http://archive.ics.uci.edu/)) đã cung cấp bộ dữ liệu này chứa các giao dịch thực tế từ năm 2010 và 2011. Bộ dữ liệu được duy trì trên trang web của họ, nơi có thể tìm thấy bộ dữ liệu bằng tiêu đề "[Online Retail](http://archive.ics.uci.edu/dataset/352/online+retail)". Theo UCI Machine Learning Repository, dữ liệu này đã được cung cấp bởi Tiến sĩ Daqing Chen, Giám đốc nhóm Phân tích Công cộng.

**Nội dung:**

Đây là một bộ dữ liệu quốc tế chứa tất cả các giao dịch diễn ra từ 01/12/2010 đến 09/12/2011 cho một doanh nghiệp bán lẻ trực tuyến không có cửa hàng ở tại Vương Quốc Anh và được đăng ký. Công ty chủ yếu bán các sản phẩm quà tặng độc đáo phù hợp cho mọi dịp. Nhiều khách hàng của công ty là các nhà buôn, bán sỉ.

# 2. Chuẩn bị dữ liệu

**Chuẩn bị các thư viện cần thiết**

import re

import pandas as pd

import numpy as np

import matplotlib as mpl

import matplotlib.pyplot as plt

import seaborn as sns

import datetime, nltk, warnings

import matplotlib.cm as cm

import itertools

from pathlib import Path

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_samples, silhouette\_score

from sklearn import preprocessing, model\_selection, metrics, feature\_selection

from sklearn.model\_selection import GridSearchCV, learning\_curve

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix

from sklearn import neighbors, linear\_model, svm, tree, ensemble

from wordcloud import WordCloud, STOPWORDS

from sklearn.ensemble import AdaBoostClassifier

from sklearn.decomposition import PCA

from IPython.display import display, HTML

import plotly.graph\_objs as go

from plotly.offline import init\_notebook\_mode,iplot

init\_notebook\_mode(connected=True)

warnings.filterwarnings("ignore")

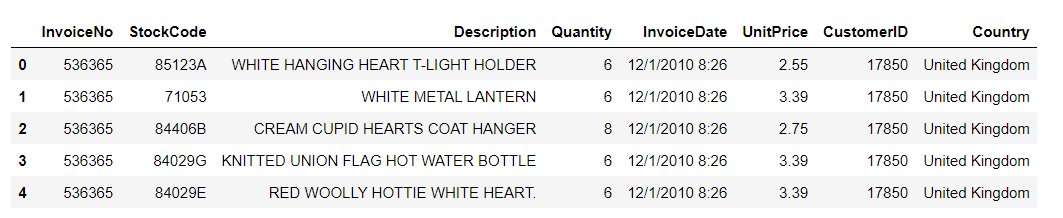
plt.rcParams["patch.force\_edgecolor"] = True

plt.style.use('fivethirtyeight')

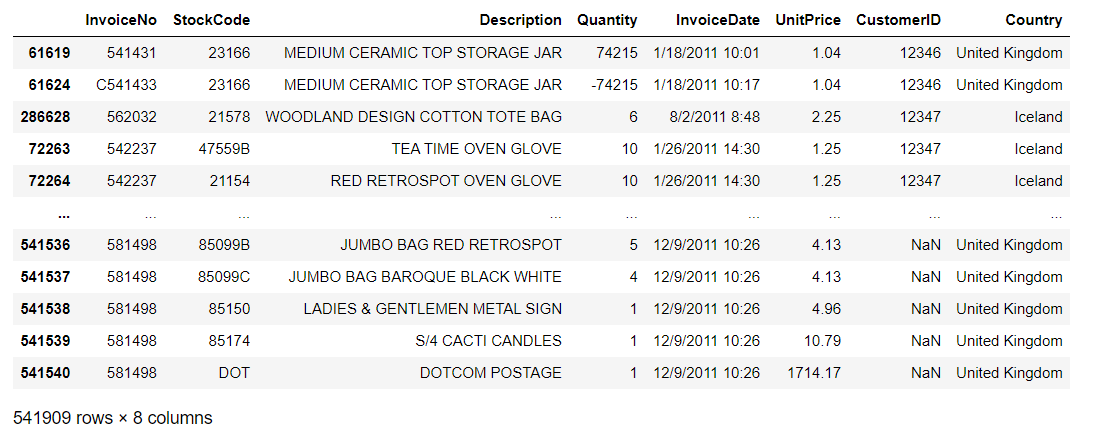
mpl.rc('patch', edgecolor = 'dimgray', linewidth=1)

%matplotlib inline

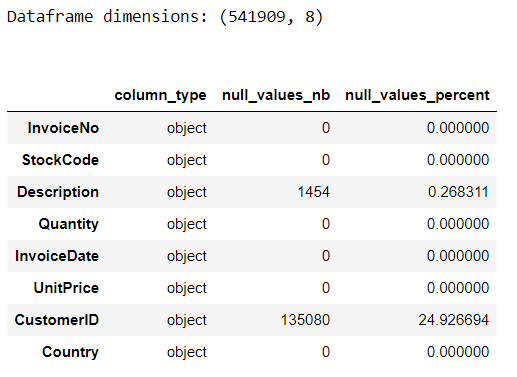
**Đọc file .csv**



**Hiển thị DataFrame sau khi sắp xếp theo cột 'CustomerID'**



**Mô tả cơ bản về dữ liệu: kiểu dữ liệu của các cột, số lượng giá trị null và tỉ lệ phần trăm so với số lượng giá trị của các cột**



**Vì có khoảng 25% record không được chỉ định cho một khách hàng cụ thể. Với dữ liệu có sẵn, không thể áp đặt các giá trị cho người dùng và do đó, các record này không có ý nghĩa đối với phân tích hiện tại. Vì vậy, nhóm quyết định xóa chúng khỏi dataframe**

# Loại bỏ các dòng có giá trị thiếu trong cột 'CustomerID'

df\_initial.dropna(subset=['CustomerID'], inplace=True)

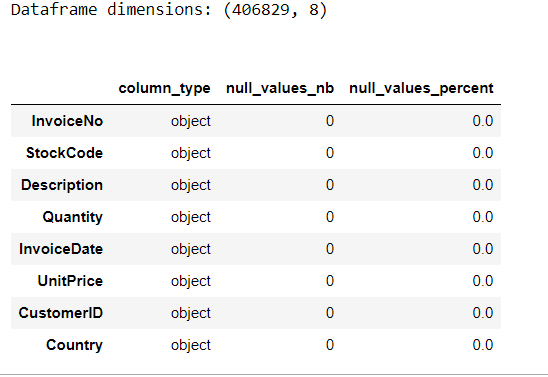
# In kích thước của DataFrame sau khi loại bỏ

print("Dataframe dimensions:", df\_initial.shape, "\n")

# Tính toán thông tin về cột

tab\_info = column\_info(df\_initial)

tab\_info

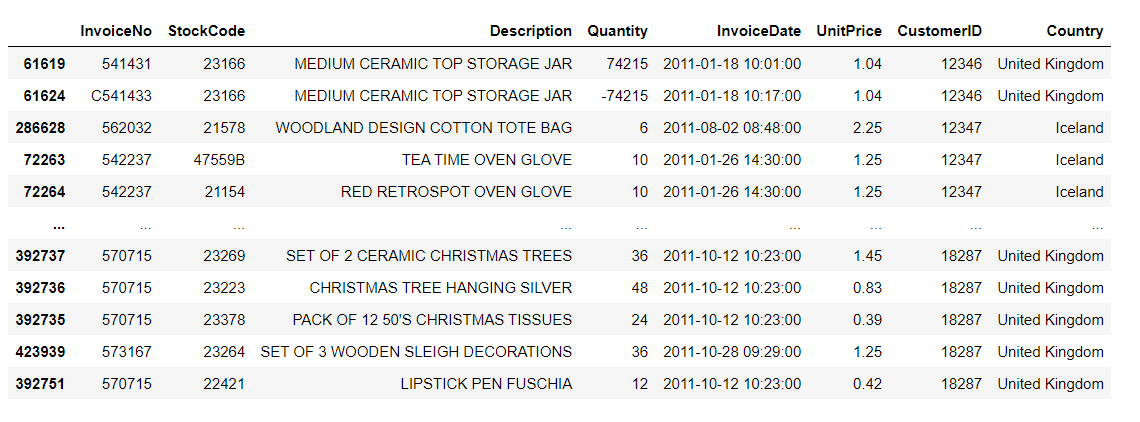


Sau khi xử lý các dòng dữ liệu null, ta kiểm tra các dòng dữ liệu trùng nhau và xử lý chúng bằng cách loại bỏ



Cuối cùng, ta chuyển đổi các kiểu dữ liệu của từng cột sao cho hợp lý Chuyển kiểu dữ liệu của cột InvoiceDate thành dạng thời gian , chuyển kiểu dữ liệu của cột CustomerID thành dạng chuỗi

Sau khi quá trình chuẩn bị dữ liệu hoàn tất, ta có một tập dữ liệu tương đối hoàn chỉnh



# 3. Khai phá dữ liệu

**Dataframe này chứa 8 biến tương ứng với:**

* InvoiceNo: Số hóa đơn - một chuỗi có mô tả như sau: số nguyên gồm 6 chữ số được gán duy nhất cho mỗi giao dịch. Nếu mã này bắt đầu bằng chữ cái 'c' thì nó biểu thị việc hủy.
* StockCode: Mã sản phẩm (mặt hàng) - một chuỗi có mô tả như sau: một số nguyên gồm 5 chữ số được gán duy nhất cho từng sản phẩm riêng biệt.
* Description: Tên sản phẩm (mặt hàng).
* Quantity: Số lượng của từng sản phẩm (mặt hàng) trên mỗi giao dịch.
* InvoiceDate: Ngày và giờ lập hóa đơn. Số, ngày và giờ mỗi giao dịch được tạo.
* UnitPrice: Đơn giá - một số thực: Giá sản phẩm trên mỗi đơn vị bằng đồng bảng Anh.
* CustomerID: Mã số khách hàng - một chuỗi có mô tả như sau: một số nguyên gồm 5 chữ số được gán duy nhất cho mỗi khách hàng.
* Country: Tên quốc gia: tên quốc gia nơi mỗi khách hàng cư trú.

## 3.1. Quốc gia

**Xem nhanh các quốc gia thực hiện đơn hàng**



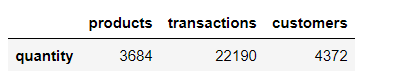
**Vẽ bản đồ thống kê chloropleth map**



Ta có thể thấy rằng tập dữ liệu phần lớn bị chi phối bởi các đơn đặt hàng được thực hiện từ Vương quốc Anh. (xấp xỉ 20/400 nghìn đơn)

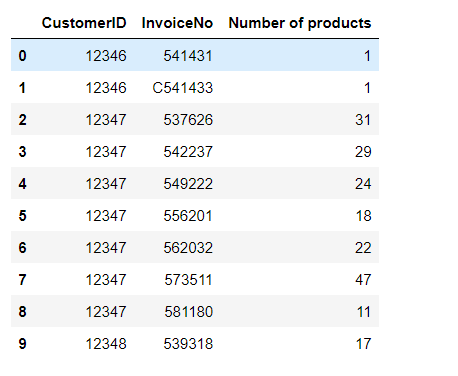
## 3.2. Khách hàng và Sản phẩm

Dataframe chứa khoảng 400.000 record. Vậy có bao nhiêu khách hàng, sản phẩm và đơn hàng trong dataframe này?



Có thể thấy rằng dữ liệu có 4372 khách hàng và họ đã mua 3684 sản phẩm khác nhau. Tổng số đơn hàng được thực hiện vào khoảng 22000 đơn.

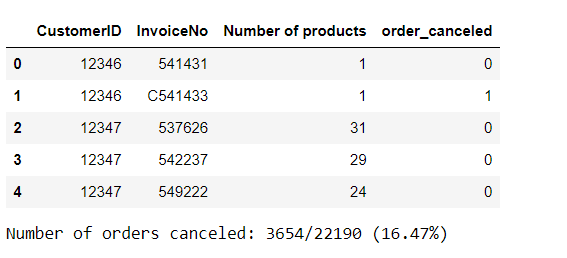
**Nhóm sẽ tiếp tục xác định số lượng sản phẩm mua trong mỗi đơn hàng:**



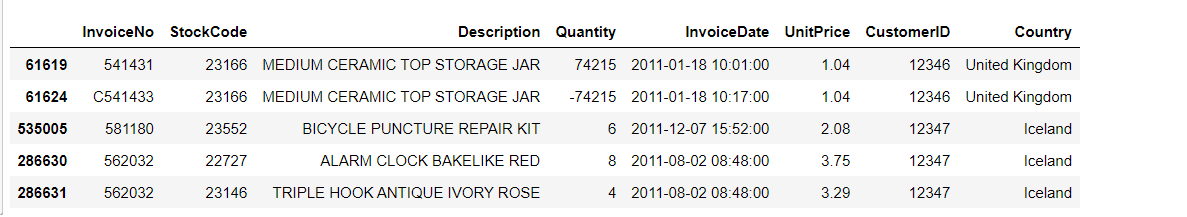
**Từ tập dữ liệu trên, ta có thể thấy:**

* Tiền tố C cho biến InvoiceNo cho biết các đơn hàng đã bị hủy.
* Có những khách hàng chỉ đến mua một lần và chỉ mua một sản phẩm (ví dụ: số 12346).
* Có những khách hàng thường xuyên mua số lượng lớn các mặt hàng trong mỗi đơn hàng.

#### 3.2.1. Các đơn hàng bị huỷ



**Có thể thấy số lượng đơn hàng bị huỷ cũng khá là nhiều (khoảng 16% trên tổng số đơn)**



Ở một vài dòng dữ liệu trên, ta có thể thấy rằng khi một đơn hàng bị hủy, ta sẽ có một đơn hàng khác trong dataframe, hầu hết giống hệt nhau ngoại trừ các biến về Số lượng và Ngày lập hóa đơn. Nhóm sẽ quyết định kiểm tra xem điều này có đúng với tất cả các record không.

Để làm điều này, nhóm sẽ tìm các record có chứa số lượng âm (trong cột Quantity) và kiểm tra xem có một đơn hàng nào chỉ ra cùng một số lượng một cách hệ thống (nhưng dương), với cùng một mô tả (ID khách hàng, Mô tả và Đơn giá) hay không:

# Chuyển đổi cột 'Quantity' sang kiểu số (float)

df\_initial['Quantity'] = df\_initial['Quantity'].astype(float)

# Lọc các hàng có Quantity < 0

df\_check = df\_initial[df\_initial['Quantity'] < 0][['CustomerID','Quantity',

'StockCode','Description','UnitPrice']]

# Lặp qua từng hàng của df\_check

for index, row in df\_check.iterrows():

if df\_initial[(df\_initial['CustomerID'] == row['CustomerID']) &

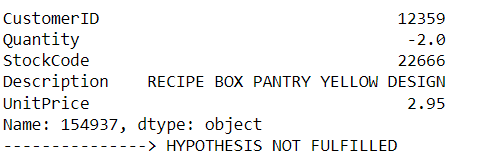
(df\_initial['Quantity'] == -row['Quantity']) &

(df\_initial['Description'] == row['Description'])].shape[0] == 0:

print(row)

print(15\*'-'+'>'+' HYPOTHESIS NOT FULFILLED')

break



df\_check = df\_initial[(df\_initial['Quantity'] < 0) & (df\_initial['Description'] != 'Discount')][

['CustomerID','Quantity','StockCode',

'Description','UnitPrice']]

for index, col in df\_check.iterrows():

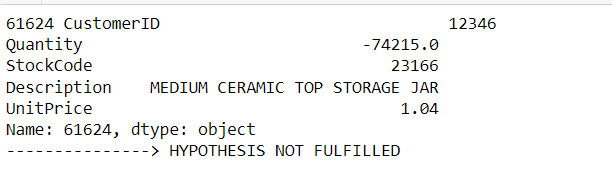
if df\_initial[(df\_initial['CustomerID'] == col[0]) & (df\_initial['Quantity'] == -col[1])

& (df\_initial['Description'] == col[2])].shape[0] == 0:

print(index, df\_check.loc[index])

print(15\*'-'+'>'+' HYPOTHESIS NOT FULFILLED')

break



Một lần nữa, ta thấy rằng giả thuyết ban đầu không được xác minh. Do đó, việc hủy bỏ đơn hàng không nhất thiết phải tương ứng với các đơn đặt hàng đã được thực hiện trước đó.

Đối với những lần hủy không có đối tác, một số trong số đó có thể là do lệnh mua được thực hiện trước tháng 12 năm 2010 (thời điểm nhập cơ sở dữ liệu). Nhóm sẽ tiếp tục thực hiện kiểm tra các đơn hàng hủy và kiểm tra sự tồn tại của các đơn hàng tương ứng:

df\_cleaned = df\_initial.copy(deep = True)

df\_cleaned['QuantityCanceled'] = 0

entry\_to\_remove = [] ; doubtfull\_entry = []

for index, col in df\_initial.iterrows():

if (col['Quantity'] > 0) or col['Description'] == 'Discount': continue

df\_test = df\_initial[(df\_initial['CustomerID'] == col['CustomerID']) &

(df\_initial['StockCode'] == col['StockCode']) &

(df\_initial['InvoiceDate'] < col['InvoiceDate']) &

(df\_initial['Quantity'] > 0)].copy()

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Cancelation WITHOUT counterpart

if (df\_test.shape[0] == 0):

doubtfull\_entry.append(index)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Cancelation WITH a counterpart

elif (df\_test.shape[0] == 1):

index\_order = df\_test.index[0]

df\_cleaned.loc[index\_order, 'QuantityCanceled'] = -col['Quantity']

entry\_to\_remove.append(index)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Various counterparts exist in orders: we delete the last one

elif (df\_test.shape[0] > 1):

df\_test.sort\_index(axis=0 ,ascending=False, inplace = True)

for ind, val in df\_test.iterrows():

if val['Quantity'] < -col['Quantity']: continue

df\_cleaned.loc[ind, 'QuantityCanceled'] = -col['Quantity']

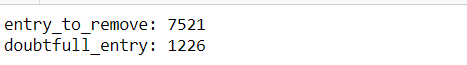
entry\_to\_remove.append(index)

break

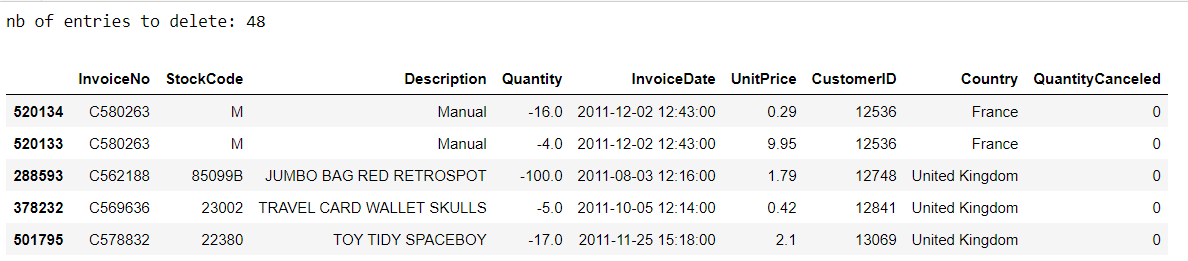
**Trong đoạn code trên, nhóm đã kiểm tra hai trường hợp:**

* Lệnh hủy tồn tại mà không có đơn đặt
* Có ít nhất một bản sao của hoá đơn có cùng số lượng với nhau

Chỉ mục của lệnh hủy tương ứng được giữ lần lượt trong doubtful\_entry (các dòng dữ liệu nghi ngờ) và entry\_to\_remove (các dòng dữ liệu loại bỏ) sẽ có kích thước là:



**Loại bỏ các dòng được chọn trong entry\_to\_remove và doubtful\_entry**



df\_cleaned[(df\_cleaned['CustomerID'] == 14048) & (df\_cleaned['StockCode'] == '22464')]



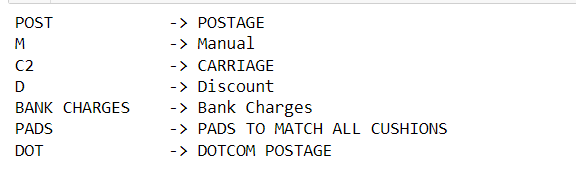
list\_special\_codes = df\_cleaned[df\_cleaned['StockCode'].str.contains('^[a-zA-Z]+', regex=True)]['StockCode'].unique()

list\_special\_codes



for code in list\_special\_codes:

print("{:<15} -> {:<30}".format(code, df\_cleaned[df\_cleaned['StockCode'] == code]['Description'].unique()[0]))



#### 3.2.2. Giá sản phẩm kèm theo giao dịch

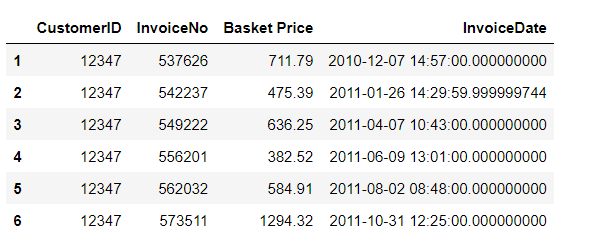
**Nhóm sẽ tạo một biến mới cho biết tổng giá của mỗi lần mua hàng:**

df\_cleaned['TotalPrice'] = df\_cleaned['UnitPrice'].astype(float) \* (df\_cleaned['Quantity'].astype(float) - df\_cleaned['QuantityCanceled']).astype(float)

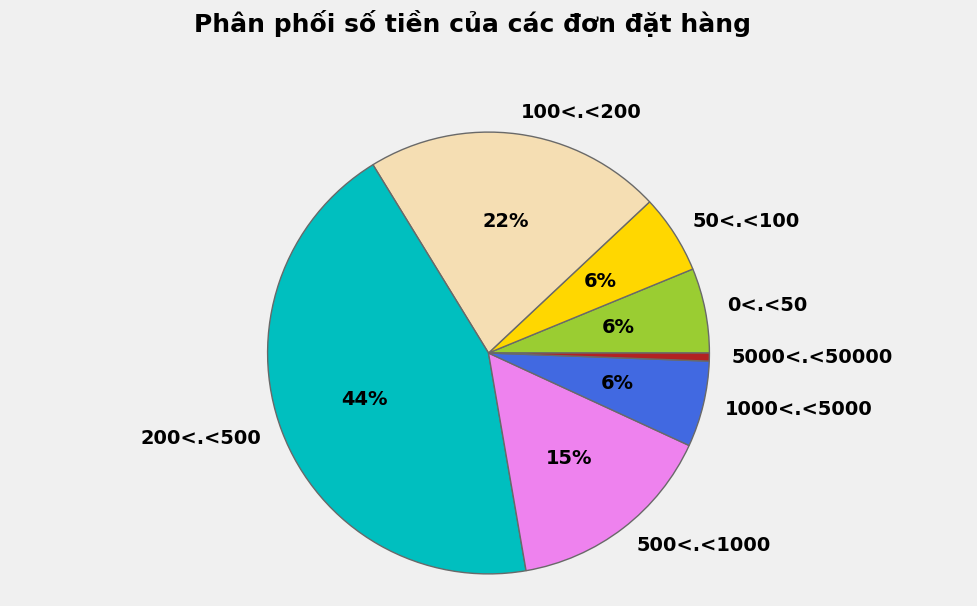
df\_cleaned.sort\_values('CustomerID')[:5]

df\_cleaned\_f = df\_cleaned.drop\_duplicates()

Mỗi dòng trong dataframe đề cập đến giá của một loại sản phẩm duy nhất. Do đó, các đơn đặt hàng được chia thành nhiều dòng. Nhóm thu thập tất cả các mua hàng được thực hiện trong một đơn đặt hàng duy nhất để tính tổng giá trị của đơn hàng:



Để có cái nhìn toàn cảnh về loại đơn hàng được thực hiện trong tập dữ liệu này, nhóm xác định cách phân chia các giao dịch mua theo tổng giải thưởng:



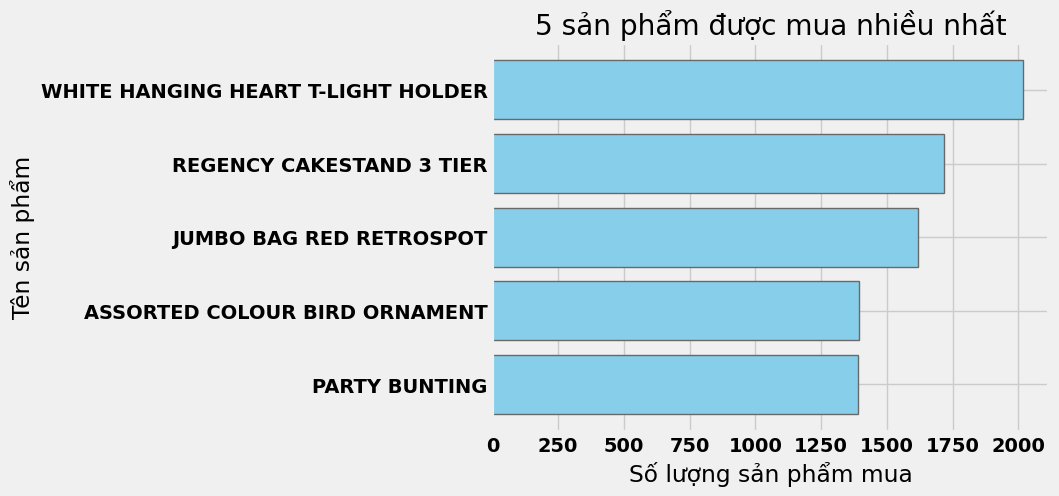
Có thể thấy rằng phần lớn các đơn đặt hàng liên quan đến các giao dịch mua tương đối lớn vì có khoảng 65% số lần mua hàng có giá trị vượt quá 200 £.

# 4. Thông tin chi tiết về các danh mục sản phẩm

Trong dataframe, các sản phẩm được xác định duy nhất thông qua cột StockCode. Mô tả ngắn gọn của các sản phẩm được cung cấp trong cột Description. Trong phần này, nhóm sử dụng 2 cột được nên trên để nhóm các sản phẩm thành các danh mục khác nhau.

## 4.1. Những sản phẩm được mua nhiều nhất

**Lưu ý: không tính các đơn hàng đã huỷ**



## 4.2. Mô tả sản phẩm - Description

**Bước đầu tiên, nhóm sẽ trích xuất thông tin từ cột Description:**

**Thu thập danh sách các sản phẩm:**

is\_noun = lambda pos: pos[:2] == 'NN'

def keywords\_inventory(dataframe, colonne = 'Description'):

stemmer = nltk.stem.SnowballStemmer("english")

keywords\_roots = dict() # collect the words / root

keywords\_select = dict() # association: root <-> keyword

category\_keys = []

count\_keywords = dict()

icount = 0

for s in dataframe[colonne]:

if pd.isnull(s): continue

lines = s.lower()

tokenized = nltk.word\_tokenize(lines)

nouns = [word for (word, pos) in nltk.pos\_tag(tokenized) if is\_noun(pos)]

for t in nouns:

t = t.lower() ; racine = stemmer.stem(t)

if racine in keywords\_roots:

keywords\_roots[racine].add(t)

count\_keywords[racine] += 1

else:

keywords\_roots[racine] = {t}

count\_keywords[racine] = 1

for s in keywords\_roots.keys():

if len(keywords\_roots[s]) > 1:

min\_length = 1000

for k in keywords\_roots[s]:

if len(k) < min\_length:

clef = k ; min\_length = len(k)

category\_keys.append(clef)

keywords\_select[s] = clef

else:

category\_keys.append(list(keywords\_roots[s])[0])

keywords\_select[s] = list(keywords\_roots[s])[0]

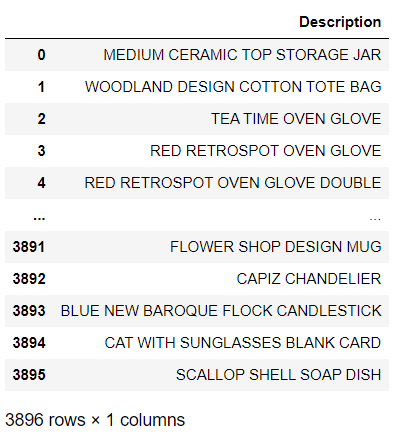
print("Nb of keywords in variable '{}': {}".format(colonne,len(category\_keys)))

return category\_keys, keywords\_roots, keywords\_select, count\_keywords

**Hàm này nhận bảng dữ liệu làm đầu vào và phân tích nội dung của cột Description bằng cách thực hiện các thao tác sau:**

* Trích xuất các tên (tên riêng, phổ thông) xuất hiện trong mô tả sản phẩm
* Đối với mỗi tên, tôi trích xuất gốc của từ và tổng hợp tập hợp các tên được liên kết với gốc từ cụ thể này
* Đếm số lần xuất hiện của mỗi gốc từ trong bảng dữ liệu
* Khi có nhiều từ được liệt kê cho cùng một gốc từ, tôi coi từ khóa được liên kết với gốc từ này là tên ngắn nhất (điều này luôn chọn từ số ít khi có các biến thể số ít/số nhiều)

**Bước đầu tiên của quá trình phân tích là lấy danh sách sản phẩm:**



**Khi danh sách này được tạo, nhóm sử dụng hàm mà nhóm đã xác định trước đó để phân tích mô tả của các sản phẩm khác nhau:**

import nltk

keywords, keywords\_roots, keywords\_select, count\_keywords = keywords\_inventory(df\_produits)



**Việc thực thi hàm này trả về ba biến quan trọng chính:**

* keywords (hay category\_keys): danh sách các từ khóa được trích xuất
* keywords\_roots: một list tượng trung cho từ điển với các khóa là các từ gốc của các từ khóa và các giá trị là danh sách các từ được liên kết với những từ gốc đó
* count\_keywords: một list tượng trung cho từ điển liệt kê số lần mỗi từ được sử dụng

**Nhóm chuyển đổi count\_keywords thành một danh sách, để sắp xếp các từ khóa theo số lần xuất hiện của chúng:**

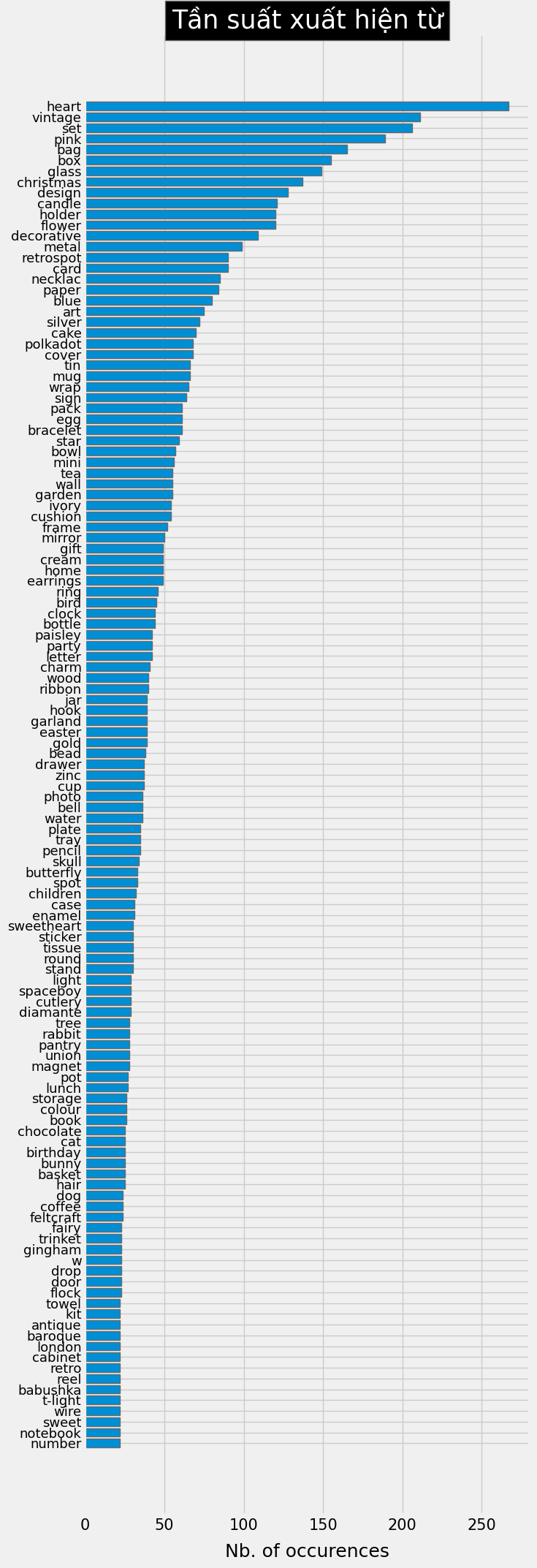
list\_products = []

for k,v in count\_keywords.items():

list\_products.append([keywords\_select[k],v])

list\_products.sort(key = lambda x:x[1], reverse = True)

**Sau đó vẽ biểu đồ thống kê tần suất xuất hiện của các từ khoá:**



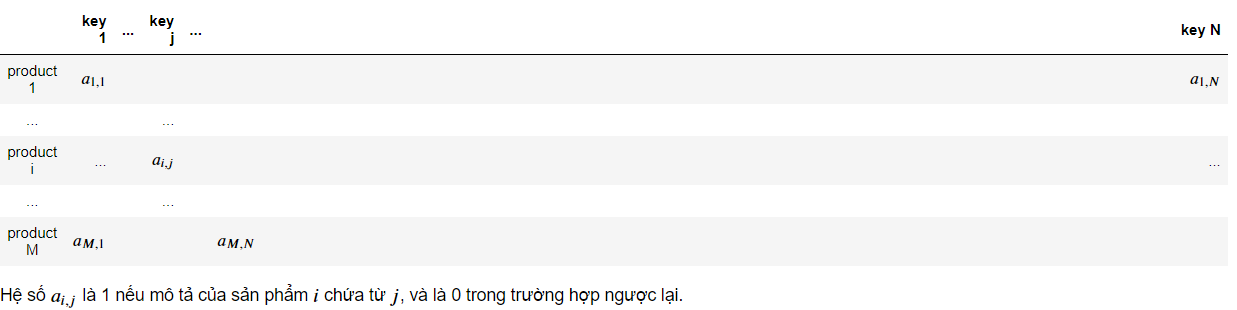
**4.3. Xác định danh mục sản phẩm**

Danh sách trên thu được chứa hơn từ khóa và các từ khóa phổ biến nhất xuất hiện trong hơn 200 sản phẩm. Tuy nhiên, khi xem xét nội dung của danh sách, ta có thể nhận thấy một số tên là không cần thiết. Một số khác không chứa thông tin, như các từ chỉ màu sắc. Do đó, nhóm sẽ thực hiện loại bỏ những từ này khỏi phân tích tiếp theo và cũng quyết định chỉ xem xét các từ xuất hiện hơn 13 lần

****

#### 4.3.1 Mã hoá dữ liệu

**Sử dụng những từ khóa này để tạo các nhóm sản phẩm. Đầu tiên, xác định ma trận 𝑋 như sau:**



Ma trận X chỉ ra các từ được chứa trong mô tả của các sản phẩm bằng cách sử dụng nguyên tắc mã hóa one-hot. Trong thực tế, nhóm đã phát hiện rằng việc đưa ra khoảng giá sẽ dẫn đến các nhóm cân đối hơn về số lượng phần tử. Do đó, nhóm sẽ thêm 6 cột bổ sung vào ma trận này và chỉ ra khoảng giá của các sản phẩm:

liste\_produits = df\_cleaned['Description'].unique()

X = pd.DataFrame()

for key, occurence in list\_products:

X.loc[:, key] = list(map(lambda x:int(key.upper() in x), liste\_produits))

# Kiểm tra và loại bỏ các giá trị không hợp lệ trong cột UnitPrice

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

df\_cleaned = df\_cleaned.dropna(subset=['UnitPrice'])

# Tiếp tục thực hiện các phép toán khác sau khi loại bỏ các giá trị không hợp lệ

threshold = [0, 1, 2, 3, 5, 10]

label\_col = []

for i in range(len(threshold)):

if i == len(threshold)-1:

col = '.>{}'.format(threshold[i])

else:

col = '{}<.<{}'.format(threshold[i],threshold[i+1])

label\_col.append(col)

X.loc[:, col] = 0

for i, prod in enumerate(liste\_produits):

prix = df\_cleaned[ df\_cleaned['Description'] == prod]['UnitPrice'].mean()

j = 0

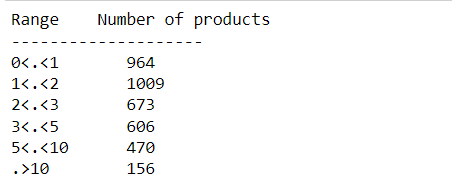
while prix > threshold[j]:

j+=1

if j == len(threshold): break

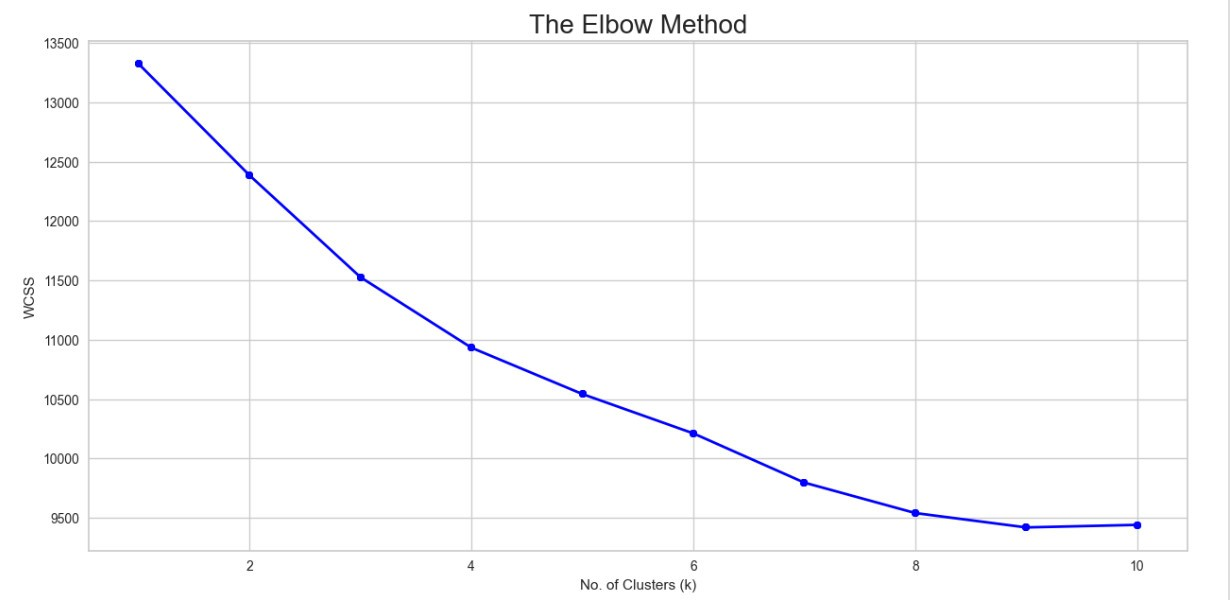
X.loc[i, label\_col[j-1]] = 1

**Và để chọn các khoảng phù hợp, nhóm sẽ kiểm tra số lượng sản phẩm trong các nhóm khác nhau:**



#### 4.3.2. Phân cụm sản phẩm

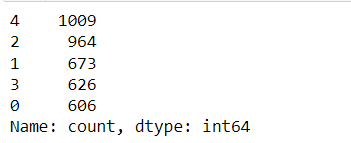
Trong phần này, nhóm sẽ phân cụm các sản phẩm thành các cụm khác nhau. Để xác định (xấp xỉ) số lượng cụm tốt nhất đại diện cho dữ liệu, nhóm sử dụng phương pháp Elbow:



Điểm silhouette cho 5 cụm cao nhất (*do độ dốc của đồ thị sau đó không lớn*).

#### 4.3.3. Đặc điểm hóa nội dung của các cụm (clusters)

**Kiểm tra số phần tử trong mỗi lớp:**



##### a. Điểm Silhouette nội cụm

A graph showing a number of different colored bars

Description automatically generated with medium confidence

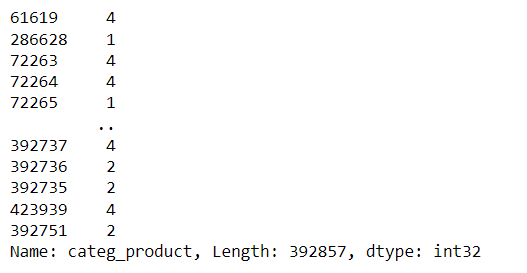
##### b. Biểu đồ WordCloud



# 5. Phân loại khách hàng

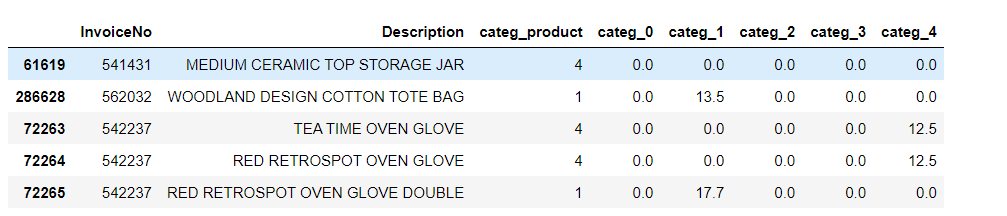
## 5.1. Định dạng dữ liệu

Trong phần trước đó, các sản phẩm khác nhau đã được nhóm thành 5 cụm. Để chuẩn bị phần còn lại của phân tích, bước đầu tiên bao gồm việc nhập thông tin này vào bảng dữ liệu. Để làm điều này, nhóm tạo thêm cột hạng mục categ\_product trong đó tôi chỉ định cụm của mỗi sản phẩm:

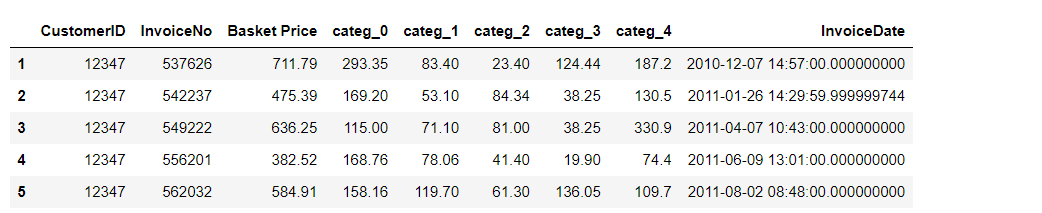


### 5.1.1. Nhóm các sản phẩm

**Tạo các biến categ\_N (với [categ\_product) chứa số tiền đã chi tiêu trong mỗi danh mục sản phẩm:**



Cho đến nay, thông tin liên quan đến một đơn hàng đã được chia thành nhiều dòng của dataframe (một dòng cho mỗi sản phẩm). Nhóm quyết định thu thập thông tin liên quan đến một đơn hàng cụ thể và đặt nó trong một mục nhập duy nhất. Do đó, nhóm tạo một dataframe mới chứa, cho mỗi đơn hàng, số tiền của giỏ hàng, cũng như cách nó được phân phối qua 5 loại sản phẩm.



### 5.1.2. Tách dữ liệu theo thời gian

Dataframe basket\_price chứa thông tin trong một khoảng thời gian là 12 tháng. Sau này, một trong các mục tiêu sẽ là phát triển một mô hình có khả năng đặc trưng và dự đoán thói quen của các khách hàng truy cập trang web và điều này, từ lần truy cập đầu tiên của họ. Để có thể kiểm tra mô hình một cách thực tế, nhóm chia tách tập dữ liệu bằng cách giữ lại 10 tháng đầu để phát triển mô hình và hai tháng tiếp theo để kiểm tra nó:

print(basket\_price['InvoiceDate'].min(), '->', basket\_price['InvoiceDate'].max())



#set\_entrainement = basket\_price[basket\_price['InvoiceDate'] < datetime.date(2011,10,1)]

#set\_test = basket\_price[basket\_price['InvoiceDate'] >= datetime.date(2011,10,1)]

#basket\_price = set\_entrainement.copy(deep = True)

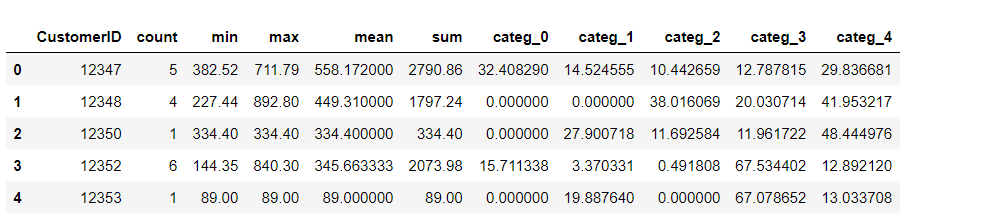
set\_entrainement = basket\_price[basket\_price['InvoiceDate'] < pd.to\_datetime('2011-10-01')]

set\_test = basket\_price[basket\_price['InvoiceDate'] >= pd.to\_datetime('2011-10-01')]

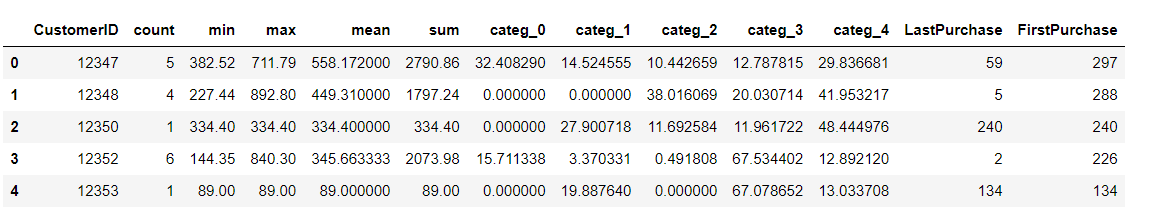
basket\_price = set\_entrainement.copy(deep=True)

### 5.1.3. Kết hợp đơn hàng của người tiêu dùng

Nhóm các mục khác nhau mà tương ứng với cùng một người dùng. Nhóm xác định số lần mua hàng của người dùng, cũng như các số lượng tối thiểu, tối đa, trung bình và tổng số tiền đã chi trả trong tất cả các lần mua hàng:



Cuối cùng, nhóm xác định hai biến bổ sung cung cấp số ngày trôi qua kể từ lần mua đầu tiên ( FirstPurchase ) và số ngày kể từ lần mua cuối cùng ( LastPurchase ):



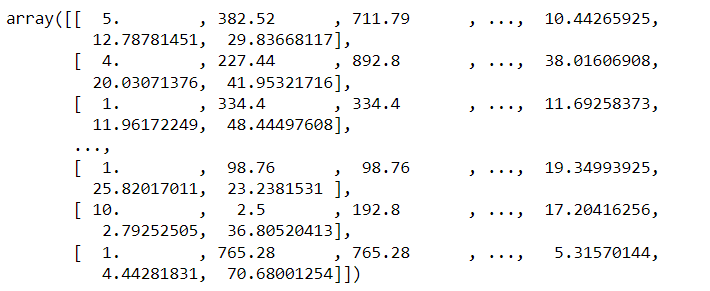
Một danh mục khách hàng đặc biệt quan trọng là những khách hàng chỉ thực hiện một lần mua. Trong phần này, nhóm nhận thấy rằng loại khách hàng này chiếm 1/3 số lượng khách hàng được liệt kê:



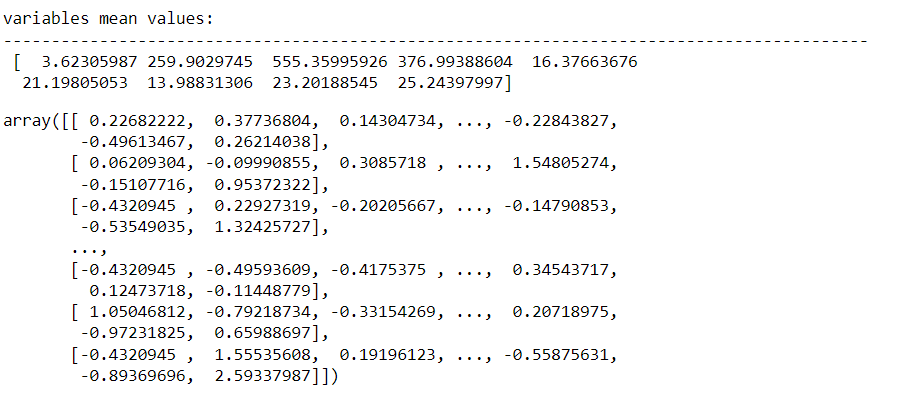
## 5.2. Tạo danh mục khách hàng

### 5.2.1. Mã hoá - định dạng dữ liệu

Bảng dữ liệu "transactions\_per\_user" chứa một tóm tắt của tất cả các giao dịch đã được thực hiện. Mỗi mục trong bảng này tương ứng với một khách hàng cụ thể. Nhóm sử dụng thông tin này để phân loại các loại khách hàng khác nhau và chỉ giữ một phần nhỏ của các biến:



Trong thực tế, các biến khác nhau mà tôi đã chọn có phạm vi biến đổi khá khác nhau và trước khi tiếp tục phân tích, nhóm tạo một ma trận trong đó các dữ liệu này được chuẩn hóa:



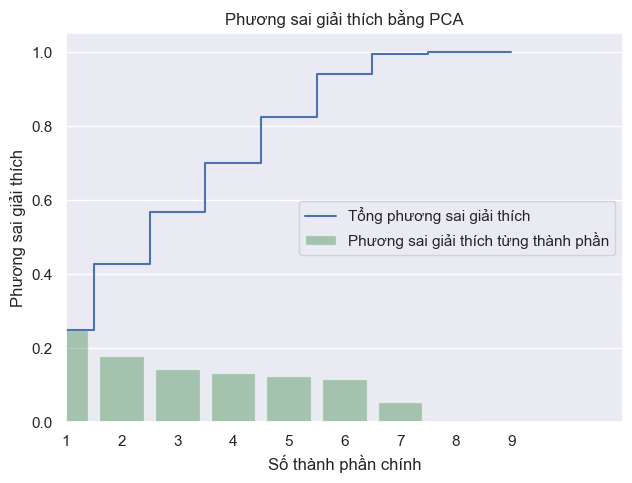
Ở bước tiếp theo, chúng ta sẽ tạo các nhóm khách hàng. Trong thực tế, trước khi tạo các nhóm này, việc xác định một cơ sở có số chiều nhỏ hơn để mô tả ma trận scaled\_matrix là rất quan trọng. Trong trường hợp này, chúng ta sẽ sử dụng cơ sở này để tạo ra một biểu đồ biểu diễn các nhóm khác nhau và kiểm tra chất lượng của sự phân tách giữa các nhóm khác nhau. Do đó, trước tiên chúng ta sẽ thực hiện PCA:

pca = PCA()

pca.fit(scaled\_matrix)

pca\_samples = pca.transform(scaled\_matrix)

**Biểu thị lượng phương sai được giải thích bởi từng thành phần:**



### 5.2.2. Tạo danh mục khách hàng

Nhóm định nghĩa các cụm khách hàng từ ma trận đã được chuẩn hóa, được định nghĩa trước đó và sử dụng thuật toán k-means. Chọn số cụm dựa trên điểm silhouette và tìm thấy rằng điểm số tốt nhất được đạt được với 11 cụm:

n\_clusters = 11

kmeans = KMeans(init='k-means++', n\_clusters = n\_clusters, n\_init=100)

kmeans.fit(scaled\_matrix)

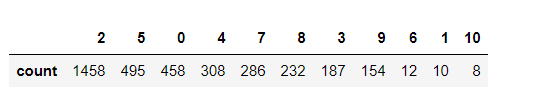
clusters\_clients = kmeans.predict(scaled\_matrix)

silhouette\_avg = silhouette\_score(scaled\_matrix, clusters\_clients)

print('score de silhouette: {:<.3f}'.format(silhouette\_avg))



**Trước tiên, nhóm xem số lượng khách hàng trong mỗi cụm:**



Báo cáo qua PCA

Có sự chênh lệch nhất định về quy mô của các nhóm khác nhau đã được tạo ra. Do đó bây giờ chúng ta sẽ cố gắng hiểu nội dung của các cụm này để xác thực (hoặc không) sự phân tách cụ thể này. Lúc đầu, chúng ta sử dụng kết quả của PCA:

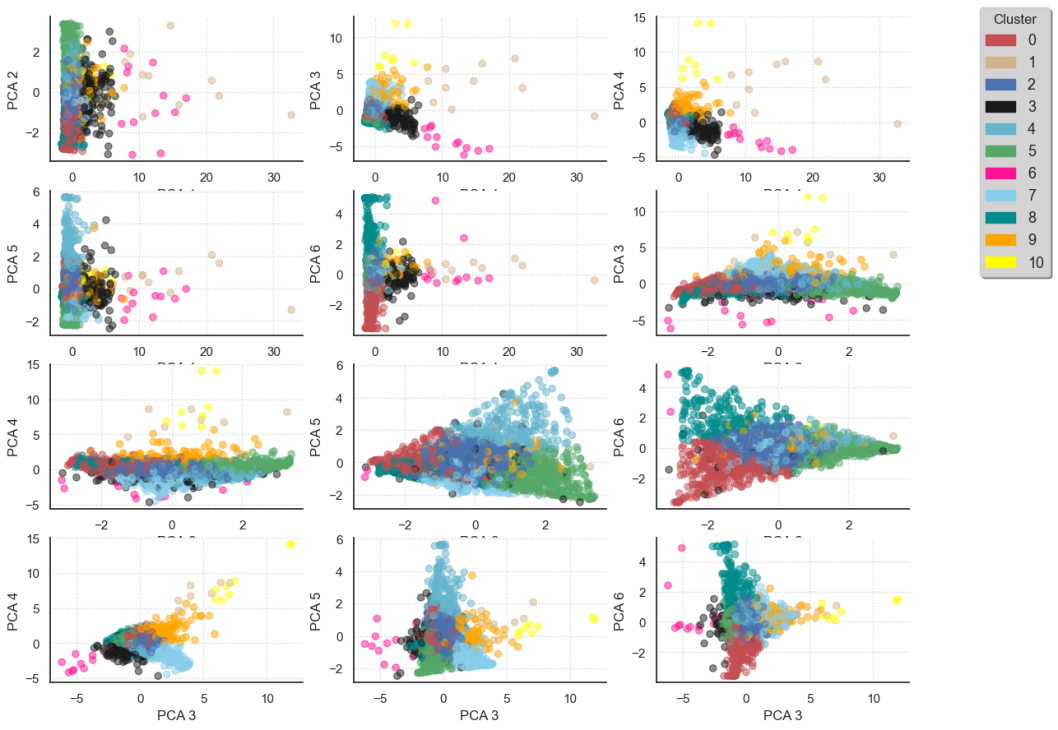
pca = PCA(n\_components=6)

matrix\_3D = pca.fit\_transform(scaled\_matrix)

mat = pd.DataFrame(matrix\_3D)

mat['cluster'] = pd.Series(clusters\_clients)

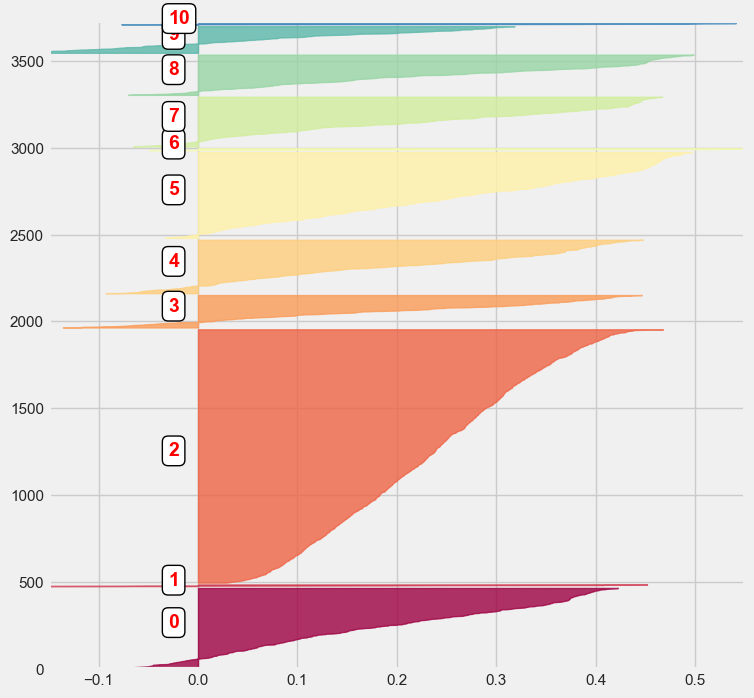
**Tạo ra một đại diện của các cụm khác nhau:**



Từ biểu đồ này, chúng ta có thể thấy, ví dụ, rằng thành phần chính đầu tiên cho phép phân tách các cụm nhỏ nhất ra khỏi phần còn lại. Nói chung, chúng ta thấy rằng luôn có một biểu diễn trong đó hai cụm sẽ xuất hiện là riêng biệt.

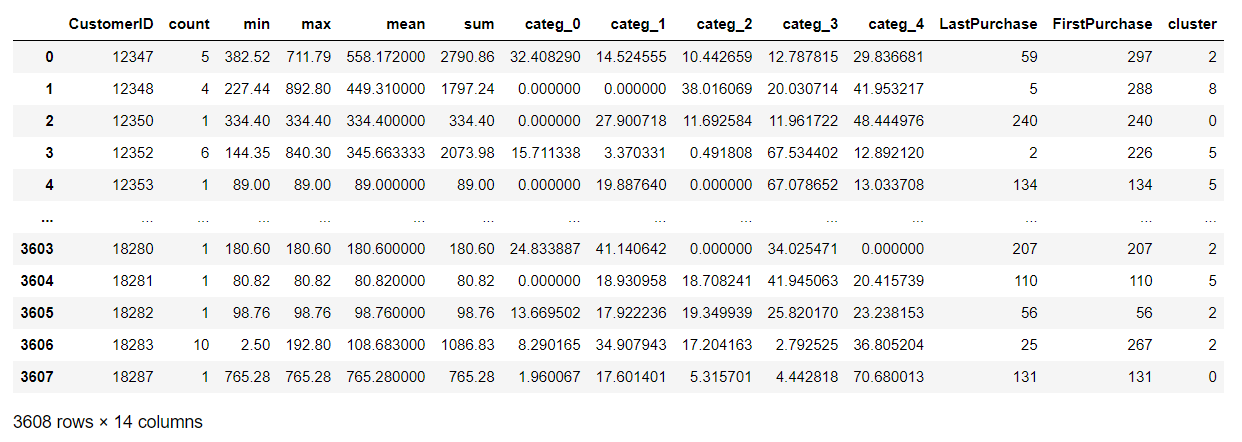
#### a. Điểm Silhoutte nội cụm

Tương tự như với các danh mục sản phẩm, một cách khác để đánh giá chất lượng của sự phân tách là xem các điểm silhouette trong các cụm khác nhau:



#### b. Hình thái khách hàng

Ở giai đoạn này, chúng ta đã xác nhận rằng các cụm khác nhau thực sự là không giao nhau (ít nhất là một cách toàn cục). Còn lại là hiểu các thói quen của khách hàng trong mỗi cụm. Để làm điều này, chúng ta bắt đầu bằng cách thêm vào dataframe selected\_customers một biến xác định cụm mà mỗi khách hàng thuộc về:



Sau đó, chúng ta tính trung bình các nội dung của dataframe này bằng cách đầu tiên chọn các nhóm khác nhau của khách hàng. Điều này cho phép truy cập vào, ví dụ, giá trung bình của các giỏ hàng, số lần ghé thăm hoặc tổng số tiền đã chi bởi các khách hàng thuộc các cụm khác nhau. Chúng ta cũng xác định số lượng khách hàng trong mỗi nhóm (biến kích thước):

merged\_df = pd.DataFrame()

for i in range(n\_clusters):

test = pd.DataFrame(selected\_customers[selected\_customers['cluster'] == i].mean())

test = test.T.set\_index('cluster', drop = True)

test['size'] = selected\_customers[selected\_customers['cluster'] == i].shape[0]

merged\_df = pd.concat([merged\_df, test])

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

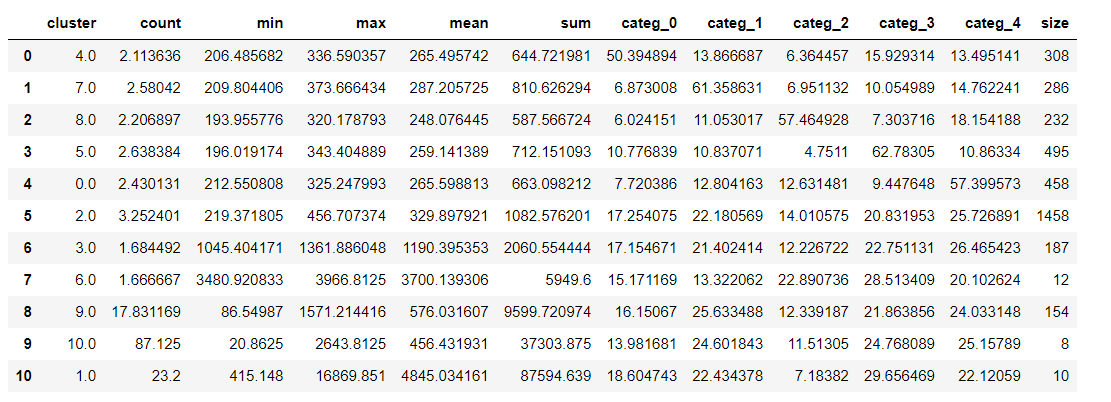
merged\_df.drop('CustomerID', axis = 1, inplace = True)

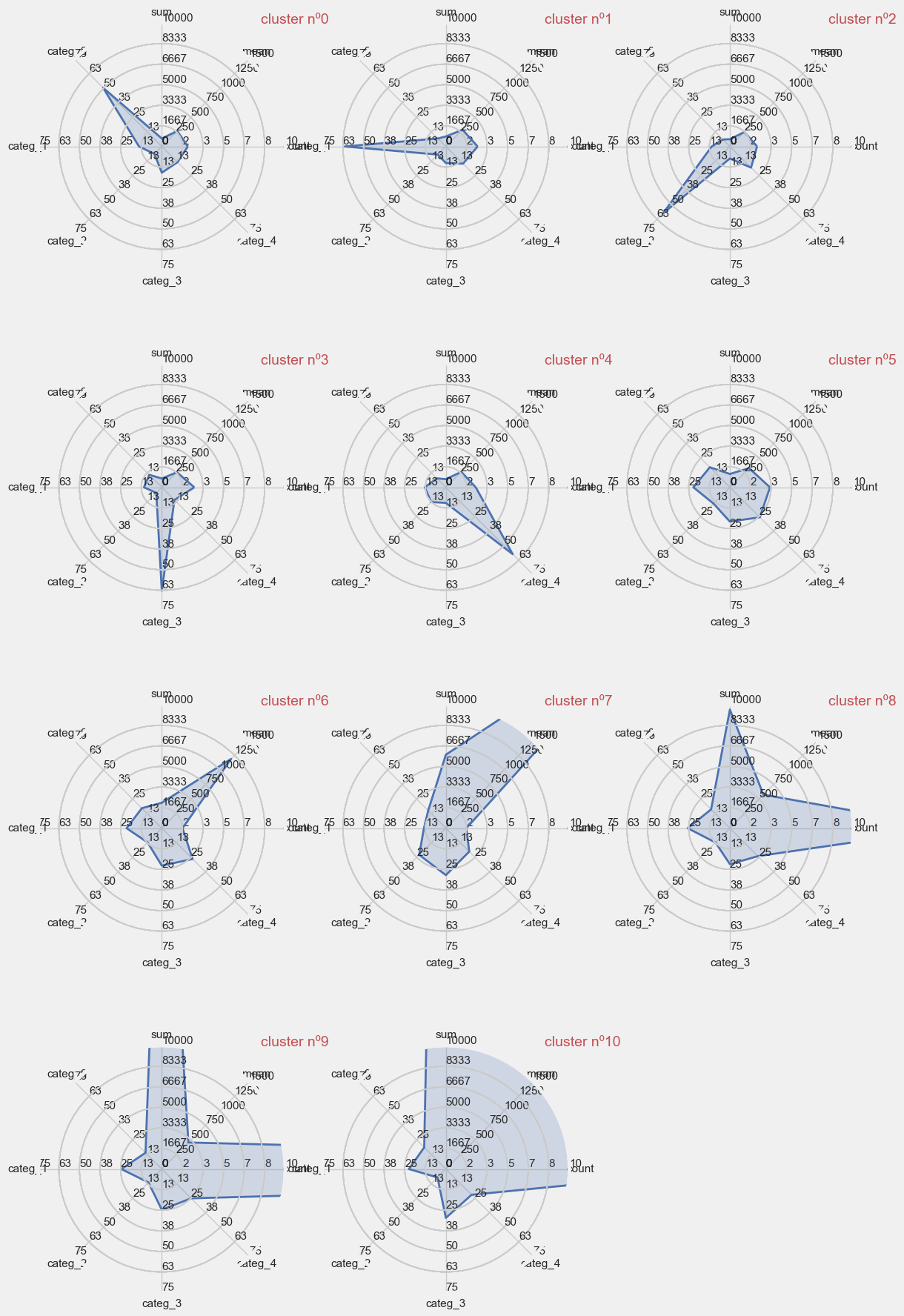
print('number of customers:', merged\_df['size'].sum())

merged\_df = merged\_df.sort\_values('sum')



Cuối cùng, chúng ta sắp xếp lại nội dung của dataframe bằng cách sắp xếp các cụm khác nhau: đầu tiên, liên quan đến số tiền đã chi trong mỗi danh mục sản phẩm và sau đó, theo tổng số tiền đã chi:





Dựa vào biểu đồ ta có thể thấy rằng 5 cụm đầu tiên tương ứng với lượng mua hàng chiếm ưu thế trong một danh mục sản phẩm cụ thể.

## 5.3. Phân tích hành vi mua hàng

Phần 5.3 này sẽ phân tích về:

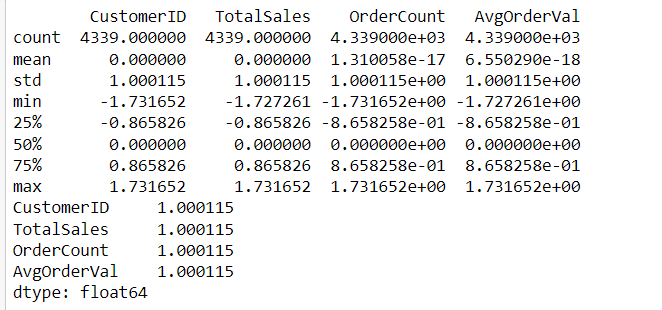
TotalSales: Tổng giá trị tất cả đơn hàng mà khách hàng đã mua.

OrderCount: Số lượng đơn hàng mà khách hàng đã đặt.

AvgOrderVal: Giá trị trung bình mỗi đơn hàng của khách hàng.



**Giá trị được tập trung sau khi chuẩn hoá có độ lệch chuẩn là 1. Bây giờ, đem dữ liệu này để phân tích phân cụm (dùng k-means):**



from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

from sklearn.metrics import pairwise\_distances

# Khởi tạo mảng chứa silhouette scores

silhouette\_scores = []

# Lặp qua số lượng clusters từ 3 đến 9

for n\_clusters in range(3, 10):

# Áp dụng thuật toán KMeans

cluster = KMeans(n\_clusters=n\_clusters, random\_state=42)

cluster.fit(df\_scale\_cus[['TotalSales', 'OrderCount', 'AvgOrderVal']])

# Tính silhouette score

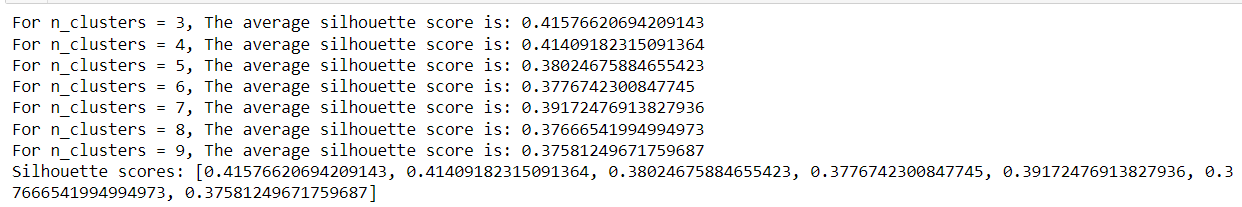
silhouette\_avg = silhouette\_score(df\_scale\_cus[['TotalSales', 'OrderCount', 'AvgOrderVal']], cluster.labels\_, metric='euclidean')

silhouette\_scores.append(silhouette\_avg)

print(f"For n\_clusters = {n\_clusters}, The average silhouette score is: {silhouette\_avg}")

# In ra silhouette scores

print("Silhouette scores:", silhouette\_scores)



# Số lượng clusters ban đầu

n\_clusters = 3

# Ngưỡng silhouette score mong muốn

target\_silhouette\_score = 0.4157639

# Khởi tạo silhouette score ban đầu

silhouette\_avg = -1

# Lặp cho đến khi silhouette score vượt qua ngưỡng mong muốn

while silhouette\_avg <= target\_silhouette\_score:

# Áp dụng thuật toán KMeans

cluster = KMeans(n\_clusters=n\_clusters, random\_state=42)

cluster.fit(df\_scale\_cus[['TotalSales', 'OrderCount', 'AvgOrderVal']])

# Tính silhouette score

silhouette\_avg = silhouette\_score(df\_scale\_cus[['TotalSales', 'OrderCount', 'AvgOrderVal']], cluster.labels\_, metric='euclidean')

# In ra silhouette score

print(f"For n\_clusters = {n\_clusters}, The average silhouette score is: {silhouette\_avg}")

# Tăng số lượng clusters

n\_clusters += 1



# Áp dụng thuật toán KMeans và lấy nhãn của các cụm

cluster = KMeans(n\_clusters=n\_clusters, random\_state=42)

cluster\_labels = cluster.fit\_predict(df\_scale\_cus[['TotalSales', 'OrderCount', 'AvgOrderVal']])

# Thêm cột Cluster vào DataFrame df\_scale\_cus

df\_scale\_cus['Cluster'] = cluster\_labels

# Tạo biểu đồ scatter plot

plt.figure(figsize=(8, 6))

for cluster\_id in df\_scale\_cus['Cluster'].unique():

cluster\_data = df\_scale\_cus[df\_scale\_cus['Cluster'] == cluster\_id]

plt.scatter(cluster\_data['AvgOrderVal'], cluster\_data['OrderCount'], label=f'Cluster {cluster\_id}')

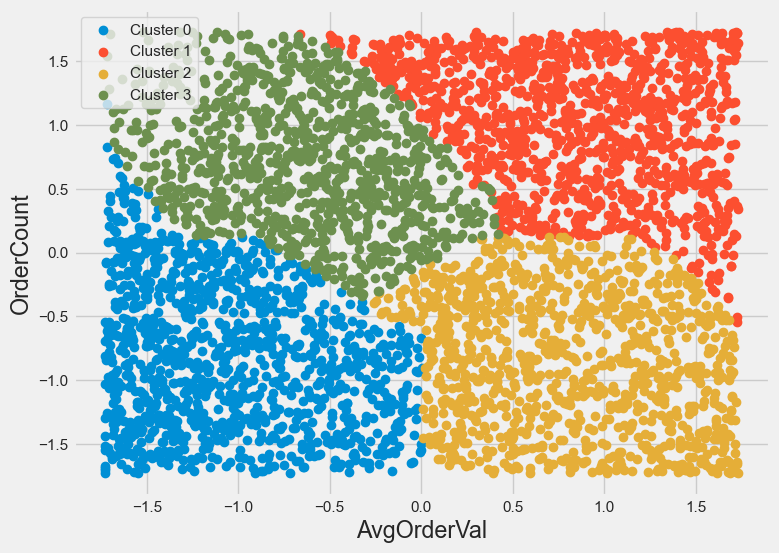
plt.xlabel('AvgOrderVal')

plt.ylabel('OrderCount')

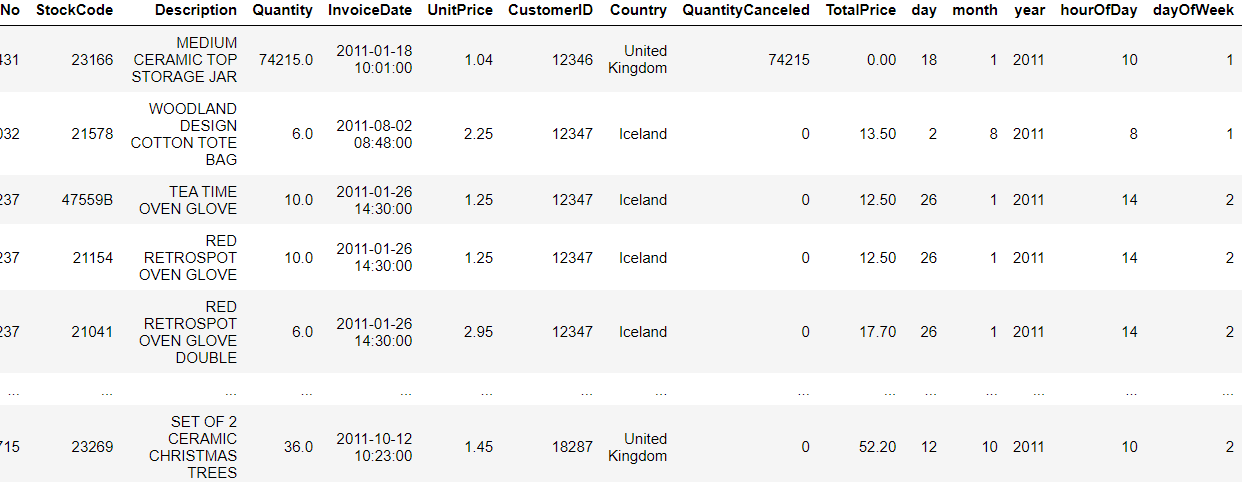
plt.legend()

plt.grid(True)

plt.show()

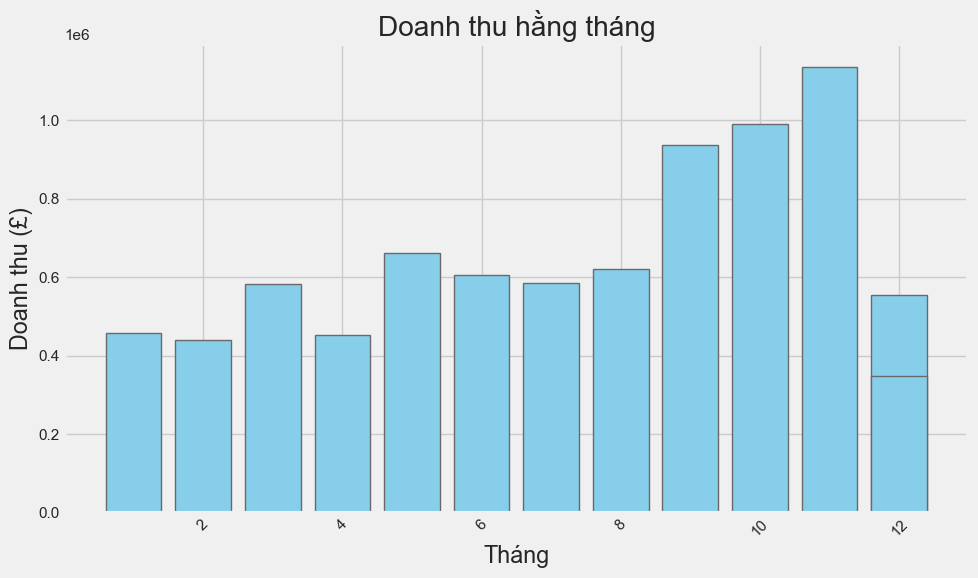


# 6. Phân tích và dự đoán doanh thu

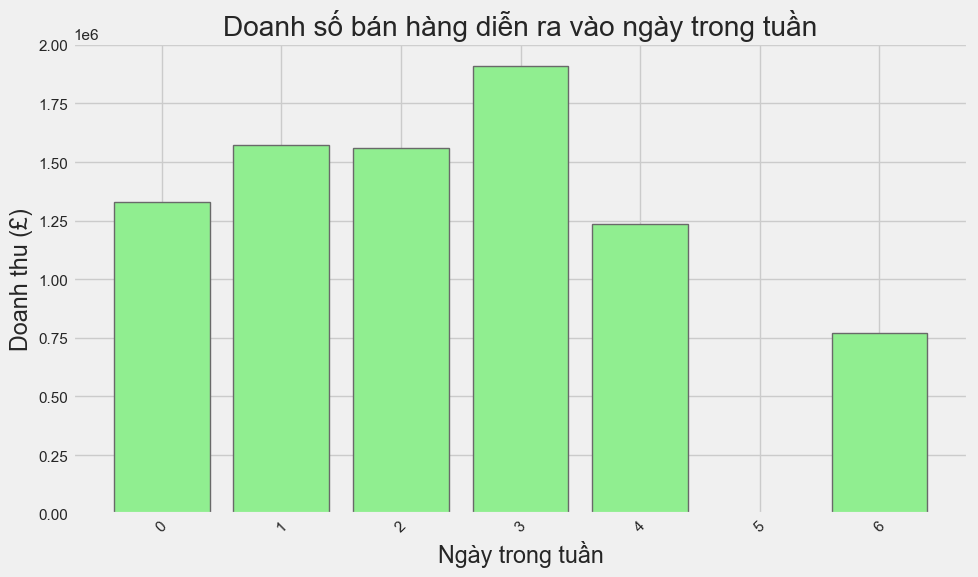


## 6.1. Phân tích doanh thu

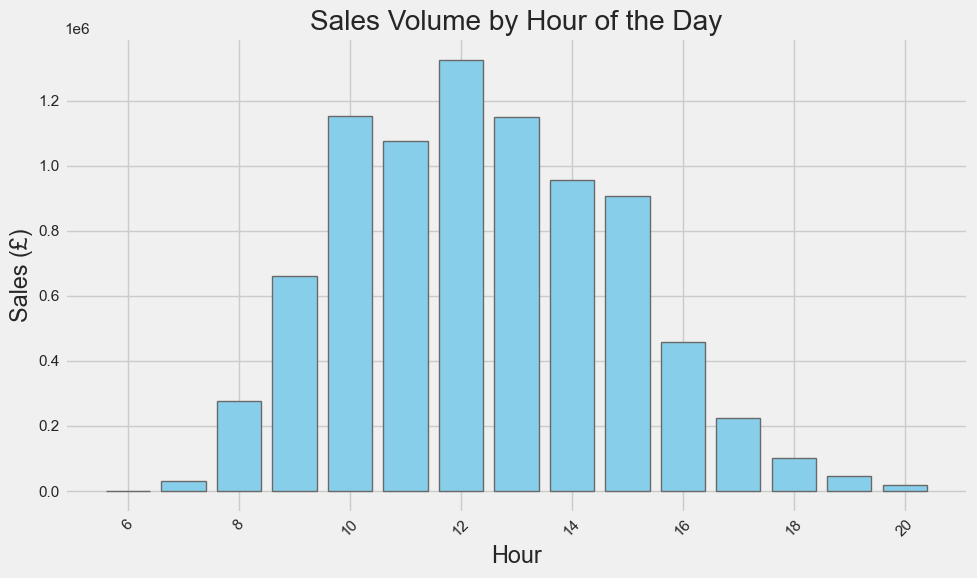
### 6.1.1. Doanh thu hằng tháng



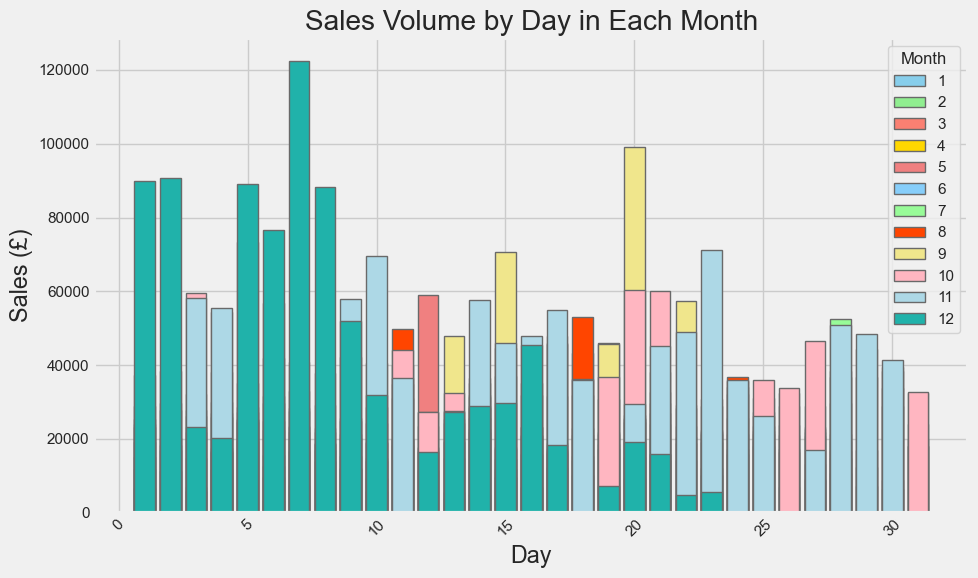
### 6.1.2. Doanh số bán hàng diễn ra vào ngày trong tuần



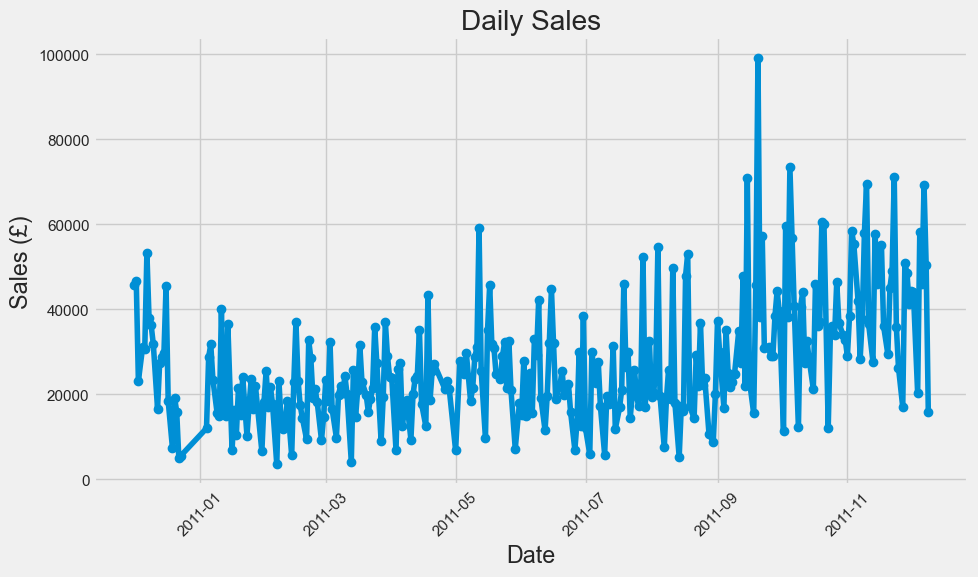
### 6.1.3. Doanh số bán hàng diễn ra vào thời điểm trong ngày



### 6.1.4. Tổng doanh thu theo ngày trong tháng



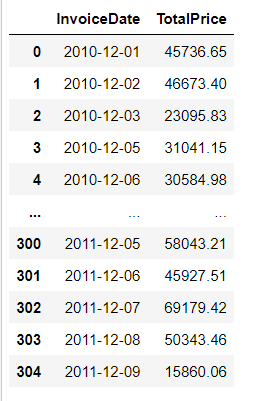
## 6.2. Phân tích và dự đoán doanh thu



sale\_data = sale\_data.rename(columns={'date': 'ds', 'revenue': 'y'})

# Display the DataFrame

sale\_data



# Tạo DataFrame cho các ngày nghỉ cuối tuần

weekend = pd.DataFrame({

'holiday': 'weekend',

'ds': pd.to\_datetime(['2010-12-04', '2010-12-11', '2010-12-18',

'2011-01-08', '2011-01-15', '2011-01-22',

'2011-01-29', '2011-02-05', '2011-02-12',

'2011-02-19', '2011-02-26', '2011-03-05',

'2011-03-12', '2011-03-12', '2011-03-19',

'2011-03-26', '2011-04-02', '2011-04-09',

'2011-04-16', '2011-04-23', '2011-04-30',

'2011-05-07', '2011-05-14', '2011-05-21',

'2011-05-28', '2011-06-04', '2011-06-11',

'2011-04-30', '2011-06-18', '2011-06-25',

'2011-07-02', '2011-07-09', '2011-07-16',

'2011-07-23', '2011-07-30', '2011-08-06',

'2011-08-13', '2011-08-20', '2011-08-27',

'2011-09-03', '2011-09-10', '2011-09-17',

'2011-09-24', '2011-10-01', '2011-10-08',

'2011-10-15', '2011-10-22', '2011-10-29',

'2011-11-05', '2011-11-12', '2011-11-19',

'2011-11-26', '2011-12-03']),

'lower\_window': 0,

'upper\_window': 1

})

# Tạo DataFrame cho các ngày nghỉ lễ

festival = pd.DataFrame({

'holiday': 'festival',

'ds': pd.to\_datetime(['2010-12-24', '2010-12-25', '2010-12-26',

'2010-12-27', '2010-12-28', '2010-12-29',

'2010-12-30', '2010-12-31', '2011-01-01',

'2011-01-02', '2011-01-03', '2011-04-22',

'2011-04-24', '2011-04-25', '2011-04-29',

'2011-05-02', '2011-05-30', '2011-08-29']),

'lower\_window': 0,

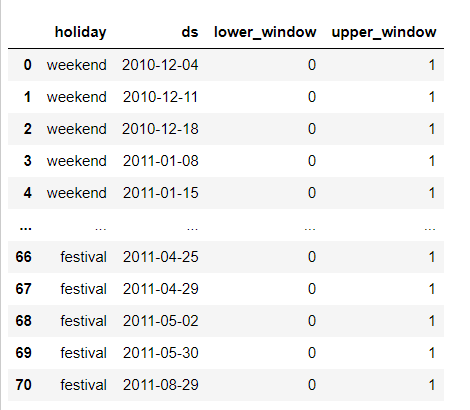
'upper\_window': 1

})

# Kết hợp DataFrame cho các ngày nghỉ cuối tuần và ngày nghỉ lễ

holidays = pd.concat([weekend, festival], ignore\_index=True)

holidays



# 7. Sử dụng các model để phân loại khách hàng

Trong phần này, mục tiêu sẽ là điều chỉnh một bộ phân loại sẽ phân loại người tiêu dùng vào các loại khách hàng khác nhau đã được thiết lập ở phần trước. Mục tiêu là làm cho phân loại này có thể thực hiện được ngay từ lần truy cập đầu tiên.

Vì mục tiêu là xác định lớp mà một khách hàng thuộc về và điều này, ngay từ lần truy cập đầu tiên, nhóm chỉ giữ lại các biến mô tả nội dung của giỏ hàng và không xem xét các biến liên quan đến tần suất truy cập hoặc biến thiên của giá giỏ hàng theo thời gian:

class Class\_Fit(object):

def \_\_init\_\_(self, clf, params=None):

if params:

self.clf = clf(\*\*params)

else:

self.clf = clf()

def train(self, x\_train, y\_train):

self.clf.fit(x\_train, y\_train)

def predict(self, x):

return self.clf.predict(x)

def grid\_search(self, parameters, Kfold):

self.grid = GridSearchCV(estimator = self.clf, param\_grid = parameters, cv = Kfold)

def grid\_fit(self, X, Y):

self.grid.fit(X, Y)

def grid\_predict(self, X, Y):

self.predictions = self.grid.predict(X)

print("Precision: {:.2f} % ".format(100\*metrics.accuracy\_score(Y, self.predictions)))

Vì mục tiêu là xác định lớp mà một khách hàng thuộc vào và điều này, ngay từ lần ghé thăm đầu tiên của họ, chúng ta chỉ giữ lại các biến mô tả nội dung của giỏ hàng và không tính đến các biến liên quan đến tần suất ghé thăm hoặc sự biến đổi của giá giỏ hàng theo thời gian:

columns = ['mean', 'categ\_0', 'categ\_1', 'categ\_2', 'categ\_3', 'categ\_4' ]

X = selected\_customers[columns]

Y = selected\_customers['cluster']

**Cuối cùng, nhóm chia bộ dữ liệu thành tập train và tập test:**

X\_train, X\_test, Y\_train, Y\_test = model\_selection.train\_test\_split(X, Y, train\_size = 0.8)

## 7.1. Support Vector Machine

svc = Class\_Fit(clf = svm.LinearSVC)

svc.grid\_search(parameters = [{'C':np.logspace(-2,2,10)}], Kfold = 5)

svc.grid\_fit(X = X\_train, Y = Y\_train)

svc.grid\_predict(X\_test, Y\_test)

****

## 7.2. k-Nearest Neighbors

knn = Class\_Fit(clf = neighbors.KNeighborsClassifier)

knn.grid\_search(parameters = [{'n\_neighbors': np.arange(1,50,1)}], Kfold = 5)

knn.grid\_fit(X = X\_train, Y = Y\_train)

knn.grid\_predict(X\_test, Y\_test)



## 7.3. Decision Tree

tr = Class\_Fit(clf = tree.DecisionTreeClassifier)

tr.grid\_search(parameters = [{'criterion' : ['entropy', 'gini'], 'max\_features' :['sqrt', 'log2']}], Kfold = 5)

tr.grid\_fit(X = X\_train, Y = Y\_train)

tr.grid\_predict(X\_test, Y\_test)



## 7.4. Random Forest

rf = Class\_Fit(clf = ensemble.RandomForestClassifier)

param\_grid = {'criterion' : ['entropy', 'gini'], 'n\_estimators' : [20, 40, 60, 80, 100],

'max\_features' :['sqrt', 'log2']}

rf.grid\_search(parameters = param\_grid, Kfold = 5)

rf.grid\_fit(X = X\_train, Y = Y\_train)

rf.grid\_predict(X\_test, Y\_test)



# 8. Kết luận

Bài báo cáo dựa trên một cơ sở dữ liệu cung cấp thông tin về các giao dịch mua sắm trên một nền tảng thương mại điện tử trong một khoảng thời gian một năm. Mỗi mục trong bộ dữ liệu mô tả việc mua sắm một sản phẩm bởi một khách hàng cụ thể và vào một ngày nhất định. Tổng cộng, có khoảng 4000 khách hàng xuất hiện trong cơ sở dữ liệu. Dựa trên thông tin có sẵn, nhóm em quyết định phát triển một bộ phân loại cho phép dự đoán loại mua sắm mà một khách hàng sẽ thực hiện, cũng như số lần ghé thăm mà khách hàng này sẽ thực hiện trong một năm. Giai đoạn đầu tiên của công việc này bao gồm việc mô tả các sản phẩm khác nhau được bán bởi trang web, nhóm các sản phẩm khác nhau thành 5 danh mục chính. Trong bước tiếp theo, nhóm thực hiện một phân loại về khách hàng bằng cách phân tích thói quen tiêu dùng của họ. Nhóm em đã phân loại khách hàng thành 11 danh mục chính dựa trên loại sản phẩm họ thường mua, số lần ghé thăm họ thực hiện và số tiền họ chi tiêu. Một khi các danh mục này được xác định, nhóm sẽ tiếp tục phân loại người tiêu dùng bằng các mô hình học máy như là Support Vector Machine Classifier (SVM), k-Nearest Neighbors, Decision Tree, Random Forest. Sau đó sẽ xem xét mô hình nào có kết quả tốt nhất. Ngoài ra còn có vẽ các loại biểu đồ để mô tả các thông tin quan trọng của bộ dữ liệu như là doanh thu theo ngày, tháng, năm và nhiều thông tin quan trọng khác.