

# ĐỒ ÁN CUỐI KHÓA CS DATA 07

## I. Giới thiệu

**1. Nội dung:** Phân tích dữ liệu các lượt giao dịch mua bán trực tuyến trên 1 ứng dụng

### 2. Bộ dữ liệu đã dùng để phân tích

- Đây là bộ dữ liệu chứa thông tin của lịch sử các lượt giao dịch mua bán hàng trực tuyến theo thời gian với các mặt hàng, khách hàng đến từ nhiều quốc gia khác nhau
- Bộ dữ liệu thuộc dạng bảng có:
  - 541,909 dòng với mỗi dòng là 1 giao dịch
  - 8 cột gồm:
    - InvoiceNo: Mã hóa đơn (mỗi hóa đơn có thể chứa nhiều sản phẩm)
    - StockCode: Mã sản phẩm
    - Description: Tên sản phẩm
    - Quantity: Số lượng sản phẩm mua/bán (âm nếu trả hàng)
    - InvoiceDate: Ngày và giờ giao dịch
    - UnitPrice: Giá mỗi sản phẩm (£)
    - CustomerID: Mã khách hàng (có thể thiêu với một số giao dịch)
    - Country: Quốc gia bán sản phẩm
- Hình ảnh bộ dữ liệu:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom
536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850	United Kingdom
536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850	United Kingdom
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047	United Kingdom
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	12/1/2010 8:34	2.1	13047	United Kingdom
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	12/1/2010 8:34	2.1	13047	United Kingdom
536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	12/1/2010 8:34	3.75	13047	United Kingdom
536367	22310	IVORY KNITTED MUG COSY	6	12/1/2010 8:34	1.65	13047	United Kingdom
536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	12/1/2010 8:34	4.25	13047	United Kingdom
536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	12/1/2010 8:34	4.95	13047	United Kingdom
536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	12/1/2010 8:34	9.95	13047	United Kingdom
536367	21754	HOME BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047	United Kingdom
536367	21755	LOVE BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047	United Kingdom
536367	21777	RECIPE BOX WITH METAL HEART	4	12/1/2010 8:34	7.95	13047	United Kingdom
536367	48187	DOORMAT NEW ENGLAND	4	12/1/2010 8:34	7.95	13047	United Kingdom

## II. Quy trình thực hiện: Thực hiện phân tích tập dữ liệu bằng ngôn ngữ lập trình Python

### 1. Chuẩn bị các thư viện cần thiết:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker
from matplotlib.ticker import FuncFormatter
```

Gồm có:

- Thư viện *Pandas* để:
  - ◆ Đọc file CSV/Excel
  - ◆ Xử lý dữ liệu dạng bảng (DataFrame, Series)
  - ◆ Thao tác với dữ liệu: lọc, nhóm, thống kê, v.v.
- Thư viện *Seaborn* để:
  - ◆ Vẽ biểu đồ đẹp hơn Matplotlib (tích hợp tốt với Pandas)
  - ◆ Tạo các biểu đồ như barplot, heatmap, scatterplot, v.v.
  - ◆ Thiết lập theme và palette màu
- Thư viện *NumPy* để:
  - ◆ Xử lý mảng số học (array)
  - ◆ Thực hiện các phép tính toán nhanh (mean, sum, v.v.)
  - ◆ Hỗ trợ thao tác vector/matrix
- Thư viện *Matplotlib (pyplot)* để:
  - ◆ Vẽ biểu đồ cơ bản (line, bar, scatter, histogram, v.v.)
  - ◆ Tùy chỉnh giao diện: tiêu đề, trục, màu sắc
- Module *ticker* của Matplotlib: Điều chỉnh định dạng số liệu trên trục (ví dụ: thêm dấu phẩy, đổi đơn vị)
- Dùng *FuncFormatter* để: Tùy chỉnh cách hiển thị số trên trục (ví dụ: đổi  $1e6$  thành  $1,000,000$  hoặc  $1M$ )

## 2. Thiết lập style tổng thể cho tất cả các biểu đồ về sau

Mục đích:

- ◆ Làm cho các biểu đồ trông đẹp hơn, đồng nhất hơn
- ◆ Áp dụng theme sáng, nền lưới trắng, và màu sắc sinh động

```
● # Setting plotting style  
plt.style.use('seaborn-v0_8-whitegrid')  
sns.set_palette("husl")
```

## 3. Đưa bộ dữ liệu vào Python rồi xử lý, làm sạch để phân tích

- Bộ dữ liệu lúc ở trạng thái ban đầu, khi chưa được xử lý, làm sạch:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom
536414	22139		56	12/1/2010 11:52	0		
536545	21134		1	12/1/2010 14:32	0		
536546	22145		1	12/1/2010 14:33	0		
536547	37509		1	12/1/2010 14:33	0		
536549	85226A		1	12/1/2010 14:34	0		
536550	85044		1	12/1/2010 14:34	0		
536552	20950		1	12/1/2010 14:34	0		
536553	37461		3	12/1/2010 14:35	0		
563148	21033		-37	8/12/2011 11:35	0		
A563186	B	Adjust bad debt	1	8/12/2011 14:51	-11062.06		
A563187	B	Adjust bad debt	1	8/12/2011 14:52	-11062.06		
563244	22740		-36	8/15/2011 10:49	0		
C536391	21983	PACK OF 12 BLUE PAISLEY TISSUES	-24	12/1/2010 10:24	0.29	17548	United Kingdom
C536391	21980	PACK OF 12 RED RETROSPOT TISSUES	-24	12/1/2010 10:24	0.29	17548	United Kingdom
C536391	21484	CHICK GREY HOT WATER BOTTLE	-12	12/1/2010 10:24	3.45	17548	United Kingdom
C536391	22557	PLASTERS IN TIN VINTAGE PAISLEY	-12	12/1/2010 10:24	1.65	17548	United Kingdom
C536391	22553	PLASTERS IN TIN SKULLS	-24	12/1/2010 10:24	1.65	17548	United Kingdom

=> Có rất nhiều ô rỗng, không có giá trị và có giá trị ngoại lệ, không phù hợp. Ngoài ra, còn có thêm rất nhiều dòng bị trùng lặp giống nhau hoàn toàn ở các giá trị của các ô

- Kiểm tra bộ dữ liệu, cụ thể gồm:

- ◆ Kiểm tra kích thước và bộ nhớ
- ◆ Xem kiểu dữ liệu và giá trị thiếu
- ◆ Để hiểu tổng quan nội dung dataset
- ◆ Tìm lỗi dữ liệu (dòng trùng)

```
df = pd.read_csv("The_original_data_(uncleaned).csv", encoding='ISO-8859-1')

print(f"Dataset Shape: {df.shape}")
print(f"Memory Usage: {df.memory_usage(deep=True).sum() / 1024**2:.2f} MB")

# Quick data overview
print("\nDATA OVERVIEW:")
df.info()
print("\nFirst 5 rows:")
print(df.head())
print("\nStatistical Summary:")
print(df.describe(include='all'))
print(f"\nMissing values:\n{df.isnull().sum()}")
print(f"Duplicate rows: {df.duplicated().sum()}")
print(f"Return invoices (starting with 'C'): {df['InvoiceNo'].astype(str).str.startswith('C').sum()}")
print(f"\nCountry distribution:\n{df['Country'].value_counts().head(10)}")
```

=> Kết quả:

```
Dataset Shape: (541909, 8)
Memory Usage: 173.12 MB

DATA OVERVIEW:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
 ---  --          --          --      
 0   InvoiceNo   541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB

First 5 rows:
   InvoiceNo StockCode           Description  Quantity  \
0      536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER      6
1      536365    71053                  WHITE METAL LANTERN      6
2      536365    84406B  CREAM CUPID HEARTS COAT HANGER      8
...                                          
Switzerland        2002
Portugal          1519
Australia         1259
Name: count, dtype: int64
```

- Bộ dữ liệu có 541,909 dòng và 8 cột
  - Bộ nhớ là 173.12 MB
  - Các cột: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country không có dữ liệu bị thiếu (no missing values)
  - Cột CustomerID có 135017 ô không có giá trị (135017 missing values)
- ◆ Kiểm tra giá trị ngoại lệ (giá trị bằng 0 hoặc âm) so với giá trị của các ô khác trong cột có giá trị kiểu số (số nguyên, số thực...):

```
# Showing the number of rows where the UnitPrice and Quantity columns have values ≤ 0
quantity = (df['Quantity'] <= 0).sum()
print("- Số lượng các dòng có ô của cột 'Quanity' bé hơn hoặc bằng 0: ", quantity)
unitPrice = (df['UnitPrice'] <= 0).sum()
print("- Số lượng các dòng có ô của cột 'UnitPrice' bé hơn hoặc bằng 0: ", unitPrice)
```

=> Kết quả:

- Số lượng các dòng có ô của cột 'Quanity' bé hơn hoặc bằng 0: 10624
- Số lượng các dòng có ô của cột 'UnitPrice' bé hơn hoặc bằng 0: 2517

- Cột 'Quantity' có 10624 ô với giá trị không phải nguyên dương
- Cột 'UnitPrice' có 2517 ô với giá trị không phải nguyên dương

- Loại bỏ các dòng bị trùng lặp (duplicates) và xử lý các ô bị thiếu giá trị (missing values), cụ thể là:
  - ◆ Đèn cho các ô không có giá trị của cột 'CustomerID' giá trị 'Unknown'
  - ◆ Đèn cho các ô không có giá trị của cột 'Description' giá trị 'Unidentified'

```
# Removing duplicates
initial_rows = len(df)
df.drop_duplicates(inplace=True)
print(f"- Đã loại bỏ {initial_rows - len(df)} dòng bị trùng lặp")

# Dealing with missing values
df['CustomerID'] = df['CustomerID'].fillna('Unknown')
df['Description'] = df['Description'].fillna('Unidentified')
df['CustomerID'] = df['CustomerID'].astype(str)
```

- Loại bỏ các dòng có ô của cột Quantity, UnitPrice chứa giá trị không dương (giá trị ngoại lệ, không hợp lý): vì đây là 2 thuộc tính thiết yếu dùng để tính toán các chỉ số quan trọng , không thể bằng không hoặc có giá trị âm được

```
# Removing the rows with the UnitPrice and Quantity columns have values <= 0
df.drop(df[(df['Quantity'] <= 0) | (df['UnitPrice'] <= 0)].index, inplace=True)
```

- Kiểm tra xem còn các dòng bị trùng lặp, các ô có giá trị bị thiếu (missing values) và các ô có giá trị ngoại lệ hay không

```
print(f"- Đã loại bỏ {initial_rows - len(df)} dòng bị trùng lặp")

# Checking whether it exists the number of rows where the UnitPrice and Quantity columns have values ≤ 0
quantity = (df['Quantity'] <= 0).sum()
print("- Số lượng các dòng có ô của cột 'Quanity' bé hơn hoặc bằng 0: : ", quantity)
unitPrice = (df['UnitPrice'] <= 0).sum()
print("- Số lượng các dòng có ô của cột 'UnitPrice' bé hơn hoặc bằng 0: ", unitPrice)

# Saving cleaned data into a new file
clean_path = "The_cleaned_data.csv"
df.to_csv(clean_path, index=False)

df.info()
```

=> Kết quả:

```
- Đã loại bỏ 5268 dòng bị trùng lặp
- Số lượng các dòng có ô của cột 'Quanity' bé hơn hoặc bằng 0: :  0
- Số lượng các dòng có ô của cột 'UnitPrice' bé hơn hoặc bằng 0:  0

Index: 524878 entries, 0 to 541908
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
 ---  --  
 0   InvoiceNo         524878 non-null   object 
 1   StockCode          524878 non-null   object 
 2   Description        524878 non-null   object 
 3   Quantity           524878 non-null   int64  
 4   InvoiceDate        524878 non-null   datetime64[ns]
 5   UnitPrice          524878 non-null   float64 
 6   CustomerID         524878 non-null   object 
 7   Country             524878 non-null   object 
 8   TotalPrice          524878 non-null   float64 
 9   InvoiceYear         524878 non-null   int32  
 10  InvoiceMonth        524878 non-null   int32  
 11  InvoiceDay          524878 non-null   int32  
 12  InvoiceHour         524878 non-null   int32  
 13  InvoiceWeekday      524878 non-null   object 
 14  Season              524878 non-null   object 

dtypes: datetime64[ns](1), float64(2), int32(4), int64(1), object(7)
memory usage: 56.1+ MB
```

- Có 5268 dòng trong bảng bị trùng lặp đã được loại bỏ
- Tất cả các cột trong bảng dữ liệu đều không còn giá trị bị thiếu (missing value)
- 2 cột ‘Quantity’, ‘UnitPrice’ cũng không còn giá trị ngoại lệ (giá trị âm) nữa
- Thêm các cột TotalPrice, InvoiceYear, InvoiceMonth, InvoiceDay, InvoiceHour, InvoiceWeekday, Season vào bảng dữ liệu nhằm phục vụ cho phân tích

```

# Converting date and creating temporal features
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

# Cleaning text fields
df['Description'] = df['Description'].str.strip().str.upper()
df['Country'] = df['Country'].str.strip().str.title()

# Creating derived features
df['TotalPrice'] = df['Quantity'] * df['UnitPrice']
df['TotalPrice'] = df['TotalPrice'].round(2)
df['InvoiceYear'] = df['InvoiceDate'].dt.year
df['InvoiceMonth'] = df['InvoiceDate'].dt.month
df['InvoiceDay'] = df['InvoiceDate'].dt.day
df['InvoiceHour'] = df['InvoiceDate'].dt.hour
df['InvoiceWeekday'] = df['InvoiceDate'].dt.day_name()

# Creating seasonal features
def get_season(month):
    if month in [10, 11, 12]:
        return 'Winter'
    elif month in [1, 2, 3]:
        return 'Spring'
    elif month in [4, 5, 6]:
        return 'Summer'
    else:
        return 'Autumn'

df['Season'] = df['InvoiceMonth'].apply(get_season)

```

- Lưu các kết quả đã xử lý bộ dữ liệu vào 1 file mới rồi kiểm tra số dòng hợp lệ (count); giá trị trung bình (mean); độ lệch chuẩn (std); giá trị nhỏ nhất (min); các phân vị 25%, 50%, 75% (quartiles); giá trị lớn nhất (max)

```

# Sorting the data by revenue in descending order and take the top 10 products
data = pd.read_csv('The_cleaned_data.csv', encoding='ISO-8859-1')
top_10_products = data[['Description', 'TotalPrice']].sort_values(by='TotalPrice', ascending=False).head(10)

data.describe()

```

=> Kết quả:

	<b>Quantity</b>	<b>UnitPrice</b>	<b>TotalPrice</b>	<b>InvoiceYear</b>	<b>InvoiceMonth</b>	<b>InvoiceDay</b>	<b>InvoiceHour</b>
count	524878.000000	524878.000000	524878.000000	524878.000000	524878.000000	524878.000000	524878.000000
mean	10.616600	3.922573	20.275399	2010.921904	7.552237	15.022472	13.073991
std	156.280031	36.093028	271.693566	0.268323	3.508164	8.660738	2.442994
min	1.000000	0.001000	0.000000	2010.000000	1.000000	1.000000	6.000000
25%	1.000000	1.250000	3.900000	2011.000000	5.000000	7.000000	11.000000
50%	4.000000	2.080000	9.920000	2011.000000	8.000000	15.000000	13.000000
75%	11.000000	4.130000	17.700000	2011.000000	11.000000	22.000000	15.000000
max	80995.000000	13541.330000	168469.600000	2011.000000	12.000000	31.000000	20.000000

Dữ liệu có:

- 524,878 dòng giao dịch

- Giá bán và số lượng sản phẩm có độ phân tán cực kỳ lớn (std Quantity = 156.28 và std TotalPrice = 271.69)

- Giao dịch diễn ra trong vòng từ 6h đến 20h

- Bao phủ 2 năm (2010–2011) và đầy đủ các tháng, ngày

## ● Trang thái bộ dữ liệu sau khi đã được xử lý, làm sạch:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice	InvoiceYear	InvoiceMonth	InvoiceDay	InvoiceHour	InvoiceWeekday	Season
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850	United Kingdom	15.3	2010	12	1	8	Wednesday	Winter
536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34	2010	12	1	8	Wednesday	Winter
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850	United Kingdom	22	2010	12	1	8	Wednesday	Winter
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34	2010	12	1	8	Wednesday	Winter
536365	84029E	RED WOOLLY HOTTE WHIE HEART.	6	12/1/2010 8:26	3.39	17850	United Kingdom	20.34	2010	12	1	8	Wednesday	Winter
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850	United Kingdom	15.3	2010	12	1	8	Wednesday	Winter
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850	United Kingdom	25.5	2010	12	1	8	Wednesday	Winter
536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850	United Kingdom	11.1	2010	12	1	8	Wednesday	Winter
536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850	United Kingdom	11.1	2010	12	1	8	Wednesday	Winter
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047	United Kingdom	54.08	2010	12	1	8	Wednesday	Winter
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	12/1/2010 8:34	2.1	13047	United Kingdom	12.6	2010	12	1	8	Wednesday	Winter
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	12/1/2010 8:34	2.1	13047	United Kingdom	12.6	2010	12	1	8	Wednesday	Winter
536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	12/1/2010 8:34	3.75	13047	United Kingdom	30	2010	12	1	8	Wednesday	Winter
536367	22310	IVORY KNITTED MUG COSY	6	12/1/2010 8:34	1.65	13047	United Kingdom	9.9	2010	12	1	8	Wednesday	Winter
536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	12/1/2010 8:34	4.25	13047	United Kingdom	25.5	2010	12	1	8	Wednesday	Winter
536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	12/1/2010 8:34	4.95	13047	United Kingdom	14.85	2010	12	1	8	Wednesday	Winter
536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	12/1/2010 8:34	9.95	13047	United Kingdom	19.9	2010	12	1	8	Wednesday	Winter
536367	21754	HOME BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047	United Kingdom	17.85	2010	12	1	8	Wednesday	Winter
536367	21755	LOVE BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047	United Kingdom	17.85	2010	12	1	8	Wednesday	Winter
536367	21777	RECIPE BOX WITH METAL HEART	4	12/1/2010 8:34	7.95	13047	United Kingdom	31.8	2010	12	1	8	Wednesday	Winter
536367	48187	DOORMAT NEW ENGLAND	4	12/1/2010 8:34	7.95	13047	United Kingdom	31.8	2010	12	1	8	Wednesday	Winter
536368	22960	JAM MAKING SET WITH JARS	6	12/1/2010 8:34	4.25	13047	United Kingdom	25.5	2010	12	1	8	Wednesday	Winter
536368	22913	RED COAT RACK PARIS FASHION	3	12/1/2010 8:34	4.95	13047	United Kingdom	14.85	2010	12	1	8	Wednesday	Winter
536368	22912	YELLOW COAT RACK PARIS FASHION	3	12/1/2010 8:34	4.95	13047	United Kingdom	14.85	2010	12	1	8	Wednesday	Winter
536368	22914	BLUE COAT RACK PARIS FASHION	3	12/1/2010 8:34	4.95	13047	United Kingdom	14.85	2010	12	1	8	Wednesday	Winter
536369	21756	BATH BUILDING BLOCK WORD	3	12/1/2010 8:35	5.95	13047	United Kingdom	17.85	2010	12	1	8	Wednesday	Winter
536370	22728	ALARM CLOCK BAKELIKE PINK	24	12/1/2010 8:45	3.75	12583	France	90	2010	12	1	8	Wednesday	Winter
536370	22727	ALARM CLOCK BAKELIKE RED	24	12/1/2010 8:45	3.75	12583	France	90	2010	12	1	8	Wednesday	Winter
536370	22726	ALARM CLOCK BAKELIKE GREEN	12	12/1/2010 8:45	3.75	12583	France	45	2010	12	1	8	Wednesday	Winter
536370	21724	PANDA AND BUNNIES STICKER SHEET	12	12/1/2010 8:45	0.85	12583	France	10.2	2010	12	1	8	Wednesday	Winter
536370	21883	STARS GIFT TAPE	24	12/1/2010 8:45	0.65	12583	France	15.6	2010	12	1	8	Wednesday	Winter
536370	10002	INFATUABLE POLITICAL GLOBE	48	12/1/2010 8:45	0.85	12583	France	40.8	2010	12	1	8	Wednesday	Winter
536370	21791	VINTAGE HEADS AND TAILS CARD GAME	24	12/1/2010 8:45	1.25	12583	France	30	2010	12	1	8	Wednesday	Winter

## 4. Phân tích bộ dữ liệu

### 4.1. Phân tích theo yếu tố sản phẩm, khách hàng

- Xác định số lượng các loại hàng bán được:

```
# Counting the number of orders (the number of times each product appears in the Description column)
order_counts = data['Description'].value_counts().reset_index()
order_counts.columns = ['Description', 'OrderCount']

# Calculating the total revenue (TotalPrice) for each product
revenue_by_product = data.groupby('Description')['TotalPrice'].sum().reset_index()

# Counting the number of unique values in the 'Description' column
unique_count = data['Description'].nunique()
print(f"Số lượng các mặt hàng được bán là: {unique_count}")
```

- Tìm ra 10 sản phẩm có doanh thu cao nhất và số lượng đặt hàng tương ứng:

```
# Combining revenue and order count data
combined_data = pd.merge(revenue_by_product, order_counts, on='Description')

# Sorting by revenue in descending order and take the top 10 products
top_10_products = combined_data.sort_values(by='TotalPrice', ascending=False).head(10)

# Drawing the combined bar chart
fig, ax1 = plt.subplots(figsize=(17, 8))

# Setting column position
x = np.arange(len(top_10_products))
width = 0.35 # The width of columns

# Drawing a column for revenue (TotalPrice) on the left y-axis
bars1 = ax1.bar(x - width/2, top_10_products['TotalPrice'], width, label='Revenue', color='skyblue')
ax1.set_xlabel('Product', fontsize=12)
ax1.set_ylabel('Revenue (€)', fontsize=12, color='skyblue')
ax1.tick_params(axis='y', labelcolor='skyblue')

# Adding value labels to the revenue column
for bar in bars1:
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2, height, f'{height:.0f}', ha='center', va='bottom', fontsize=10, color='black')

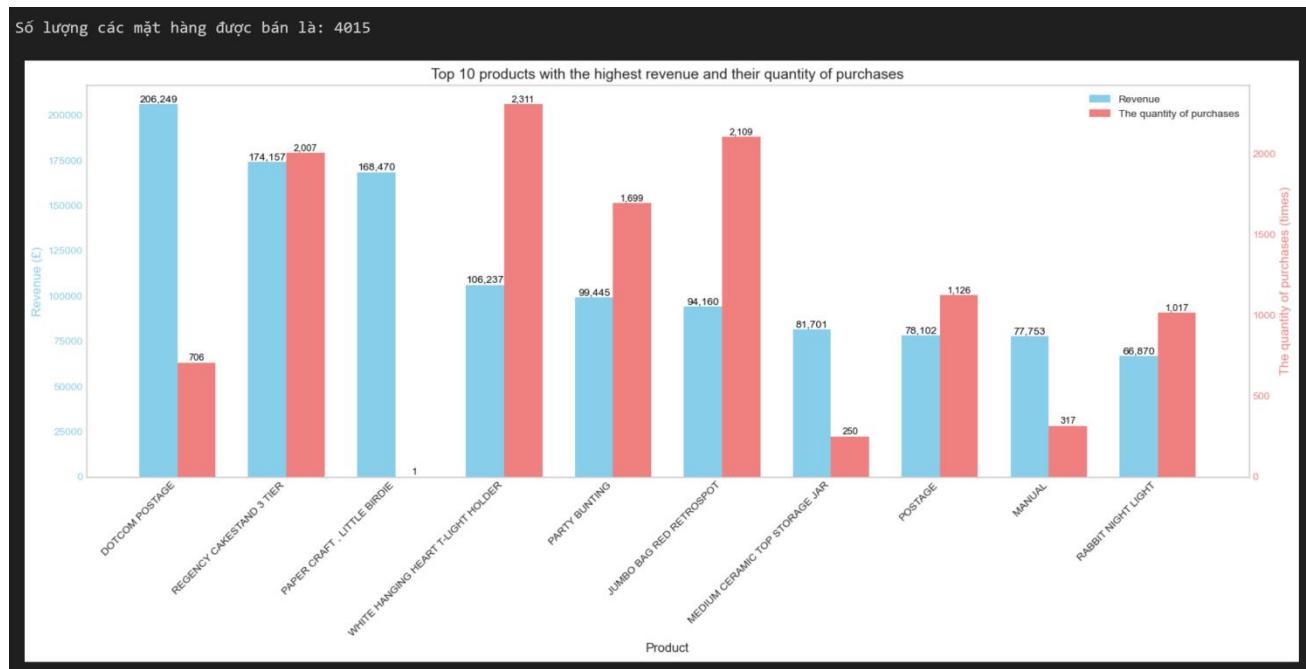
# Creating a second y-axis for the number of orders
ax2 = ax1.twinx()
bars2 = ax2.bar(x + width/2, top_10_products['OrderCount'], width, label='The quantity of purchases', color='lightcoral')
ax2.set_ylabel('The quantity of purchases (times)', fontsize=12, color='lightcoral')
ax2.tick_params(axis='y', labelcolor='lightcoral')
ax2.grid(False)

# Adding value labels to the order frequency column
for bar in bars2:
    height = bar.get_height()
    ax2.text(bar.get_x() + bar.get_width()/2, height, f'{height:.0f}', ha='center', va='bottom', fontsize=9, color='black')

# Setting up the title and x-axis label
plt.title('Top 10 products with the highest revenue and their quantity of purchases', fontsize=14)
ax1.set_xticks(x)
ax1.set_xticklabels(top_10_products['Description'], rotation=45, ha='right')

# Adding legend
lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper right')
ax1.grid(False)
# Ensuring the neat layout
plt.tight_layout()
plt.show()
```

=> Kết quả:



- Có 4015 mặt hàng được đặt mua trong các giao dịch
- Sản phẩm 'DOTCOM POSTAGE' có doanh thu cao nhất (206, 249 £) với 706 lượt đặt mua hàng
- Xác định số lượng các khách hàng tham gia mua hàng:

```
# Counting the number of unique CustomerIDs
num_customers = data['CustomerID'].nunique()

# Printing to the screen
print(f"Số lượng các khách hàng tham gia mua hàng là: {num_customers}")
```

=> Kết quả: Số lượng các khách hàng tham gia mua hàng là: 4339

- Tìm ra 10 khách hàng có số lượt mua nhiều nhất:

```
# Counting the number of purchases for each CustomerID and get the top 10
customer_order_count = data['CustomerID'].value_counts().head(10)

# Creating a horizontal bar chart
plt.figure(figsize=(13, 6))
bars = plt.barh(customer_order_count.index.astype(str), customer_order_count.values, color='tab:green')

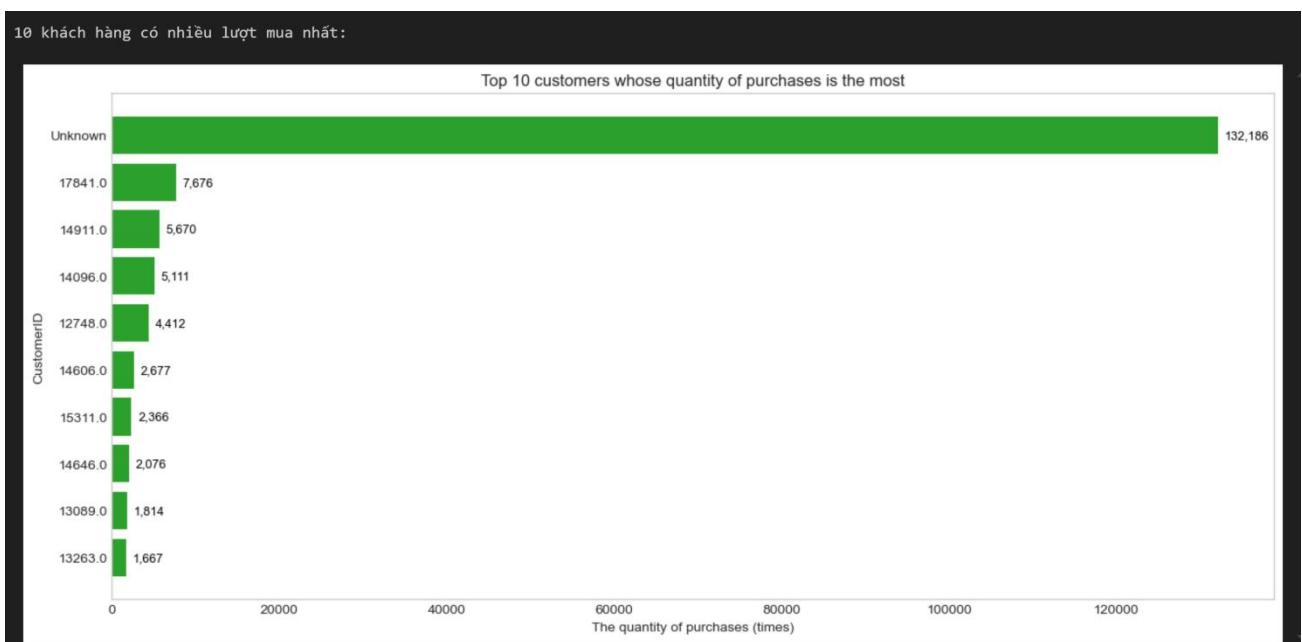
# Add annotations on each bar showing the purchase count
for bar in bars:
    width = bar.get_width()
    plt.annotate(f'{width:,}', # Format numbers with comma separators
                 xy=(width, bar.get_y() + bar.get_height() / 2),
                 xytext=(5, 0), textcoords="offset points",
                 ha='left', va='center', fontsize=9, color='black')

# Setting chart title and axis labels
plt.title('Top 10 customers whose quantity of purchases is the most')
plt.xlabel('The quantity of purchases (times)')
plt.ylabel('CustomerID')

# Reversing the y-axis so the customer with the most purchases appears at the top
plt.gca().invert_yaxis()

# Adjusting layout to fit all elements
plt.tight_layout()
plt.grid()
print('10 khách hàng có nhiều lượt mua nhất:')
plt.show()
```

=> Kết quả: Tổng số lượt mua nhiều nhất thuộc về các khách hàng không xác định được danh tính (Unknown)



- Xác định số lượng các khách hàng mua sản phẩm có doanh thu cao nhất (DOTCOM POSTAGE):

```
# Filtering the rows where Description is 'DOTCOM POSTAGE'
filtered_df = data[data['Description'] == "DOTCOM POSTAGE"]

# Counting unique CustomerID values
unique_customers = filtered_df['CustomerID'].nunique()
print(f"Số lượng các khách hàng mua sản phẩm 'DOTCOM POSTAGE': {unique_customers}")
```

=> Kết quả: Số lượng các khách hàng mua sản phẩm 'DOTCOM POSTAGE': 2

- Xác định xem 2 khách hàng tham gia mua sản phẩm ‘DOTCOM POSTAGE’ là ai:

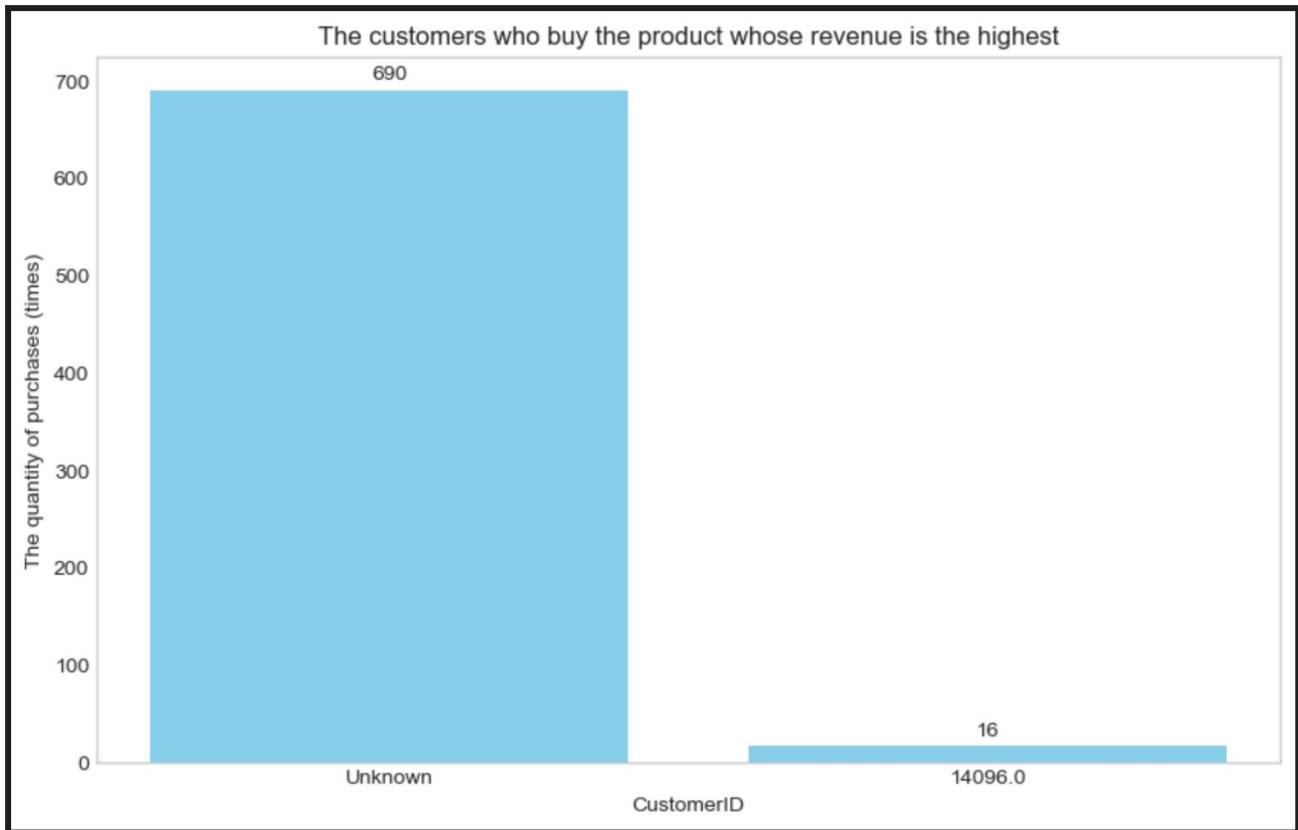
```
# Counting the occurrences of each CustomerID
customer_counts = filtered_df['CustomerID'].value_counts().head(10)

# Plotting a bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(customer_counts.index.astype(str), customer_counts.values, color='skyblue')

# Adding values on top of the columns
for bar in bars:
    height = bar.get_height()
    plt.annotate(f'{height}', xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 3), # khoảng cách giữa số và cột
                 textcoords="offset points",
                 ha='center', va='bottom')

# Adding title and axis labels
plt.title('The customers who buy the product whose revenue is the highest')
plt.xlabel('CustomerID')
plt.ylabel('The quantity of purchases (times)')
plt.grid(False)
plt.show()
```

=> Kết quả:



- Số lượng lượt mua sản phẩm này phần đông thuộc về các khách hàng không xác định được danh tính (Unknown) với 690 lượt mua
- Còn lại được khách hàng có mã ID ‘14096’ mua với 16 lượt

- Tìm ra 10 sản phẩm có số lượng lượt mua nhiều nhất và doanh thu tương ứng:

```
# Grouping by Description to get order count and total revenue
product_stats = data.groupby('Description').agg({
    'InvoiceNo': 'count',           # Number of orders
    'TotalPrice': 'sum'             # Total revenue
}).rename(columns={'InvoiceNo': 'Order Count', 'TotalPrice': 'Total Revenue'})

# Getting top 10 products by number of orders
top_10_products = product_stats.sort_values(by='Order Count', ascending=False).head(10)

# Plotting horizontal bar chart
fig, ax2 = plt.subplots(figsize=(17, 7))  # ax2 will now be the primary axis

bar_width = 0.4
y_positions = range(len(top_10_products))

# Plotting Total Revenue bars (now primary axis)
bars2 = ax2.bars(
    [y + bar_width/2 for y in y_positions],
    top_10_products['Total Revenue'],
    height=bar_width,
    color='tab:orange',
    label='Total Revenue (£)'
)
```

```

# Annotating Order Count
for bar in bars1:
    width = bar.get_width()
    ax1.annotate(f'{width}', 
                xy=(width, bar.get_y() + bar.get_height()/2),
                xytext=(5, 0), textcoords='offset points',
                ha='left', va='center', fontsize=9, color='black')

# Setting labels for ax1 (now secondary)
ax1.set_xlabel('The quantity of purchases (times)')
ax1.xaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x):,}'))

# Adding title and legends
plt.title('Top 10 products whose quantity of purchases is the most and their total revenue')
lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines1 + lines2, labels1 + labels2, loc='lower right')
ax1.grid(False)
plt.tight_layout()
plt.show()

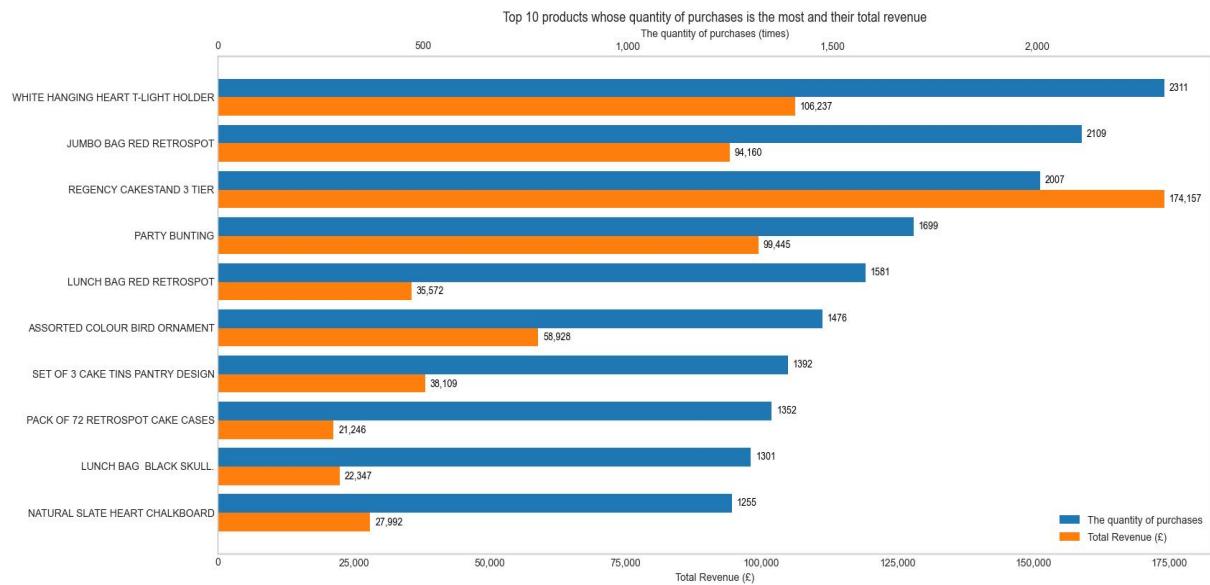
# Annotating Total Revenue
for bar in bars2:
    width = bar.get_width()
    ax2.annotate(f'{width:.0f}', 
                xy=(width, bar.get_y() + bar.get_height()/2),
                xytext=(5, 0), textcoords='offset points',
                ha='left', va='center', fontsize=9, color='black')

# Setting labels and title for ax2 (now primary)
ax2.set_xlabel('Total Revenue (£)')
ax2.set_yticks(y_positions)
ax2.set_yticklabels(top_10_products.index)
ax2.invert_yaxis() # Largest on top
ax2.xaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{int(x):,}'))
ax2.grid(False)
# Creating secondary axis for Order Count
ax1 = ax2.twiny()

# Plotting Order Count bars (secondary axis)
bars1 = ax1.barh([
    [y - bar_width/2 for y in y_positions],
    top_10_products['Order Count'],
    height=bar_width,
    color='tab:blue',
    label='The quantity of purchases'
])

```

=> Kết quả: Sản phẩm ‘WHITE HANGING HEART T-LIGHT HOLDER’ có tổng số lượt mua nhiều nhất với 2311 lượt mua và doanh thu của sản phẩm này là 106 237 bảng



- Tìm ra số lượng khách hàng tham gia mua sản phẩm ‘WHITE HANGING HEART T-LIGHT HODER’ và 10 khách hàng mua sản phẩm này nhiều nhất:

```
# Filtering rows whose value in the column 'Description' is 'WHITE HANGING HEART T-LIGHT HOLDER'
filtered= data[data['Description'] == 'WHITE HANGING HEART T-LIGHT HOLDER']

# Counting unique CustomerID values
unique_customer_ids = filtered['CustomerID'].nunique(dropna=True)
print(f"Số lượng khách hàng mua sản phẩm 'WHITE HANGING HEART T-LIGHT HOLDER': {unique_customer_ids}")

# Counting the number of occurrences of each CustomerID
customer_counts = filtered['CustomerID'].value_counts().head(10)

# Plotting a horizontal bar chart
plt.figure(figsize=(10, 6))
bars = plt.barh(customer_counts.index.astype(str), customer_counts.values, color='yellow')

# Adding data labels to each bar"
for bar in bars:
    plt.text(bar.get_width(), bar.get_y() + bar.get_height() / 2,
             f'{int(bar.get_width())}', va='center')

plt.xlabel('The quantity of purchases (times)')
plt.ylabel('CustomerID')
plt.title('Top 10 customers who buy the product [WHITE HANGING HEART T-LIGHT HOLDER] the most')
plt.gca().invert_yaxis() # Đảo ngược trục y để giá trị lớn nhất ở trên
plt.tight_layout()
plt.grid(False)
plt.show()
```

=> Kết quả:

Số lượng khách hàng mua sản phẩm 'WHITE HANGING HEART T-LIGHT HOLDER': 857



- Có 857 vị khách hàng tham gia mua sản phẩm này
  - Sản phẩm này cũng được mua nhiều nhất bởi các khách hàng không xác định được danh tính (Unknown)
- Xác định số lượng các khách hàng xác định được danh tính (có CustomerID), tỉ lệ phần trăm giữa các khách hàng xác định được danh tính (có CustomerID) và các khách hàng không xác định được danh tính (Unknown):

```
# Filtering the rows whose the column 'CustomerID' is not 'Unknown'
filtered = data[data['CustomerID'] != 'Unknown']

# Counting the number of unique CustomerID values
Unique_customer_ids = filtered['CustomerID'].nunique()

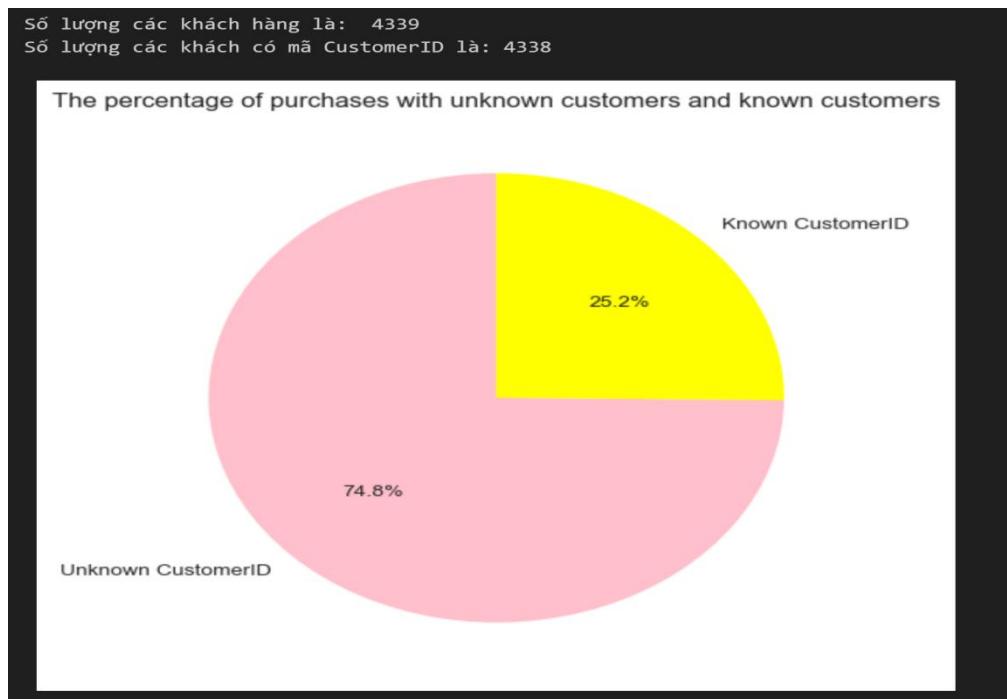
# Printing results
print("Số lượng các khách có mã CustomerID là:", Unique_customer_ids)

# Counting the number of rows whose CustomerID is 'unknown' and not 'unknown'
counts = data['CustomerID'].apply(lambda x: x == "Unknown").value_counts()

# Setting the label name
labels = ['Unknown CustomerID', 'Known CustomerID']

# Drawing the pie chart
plt.figure(figsize=(6, 6))
plt.pie(counts, labels=labels, autopct='%.1f%%', startangle=90, colors=['pink','yellow'])
plt.title('The percentage of purchases with unknown customers and known customers')
plt.show()
```

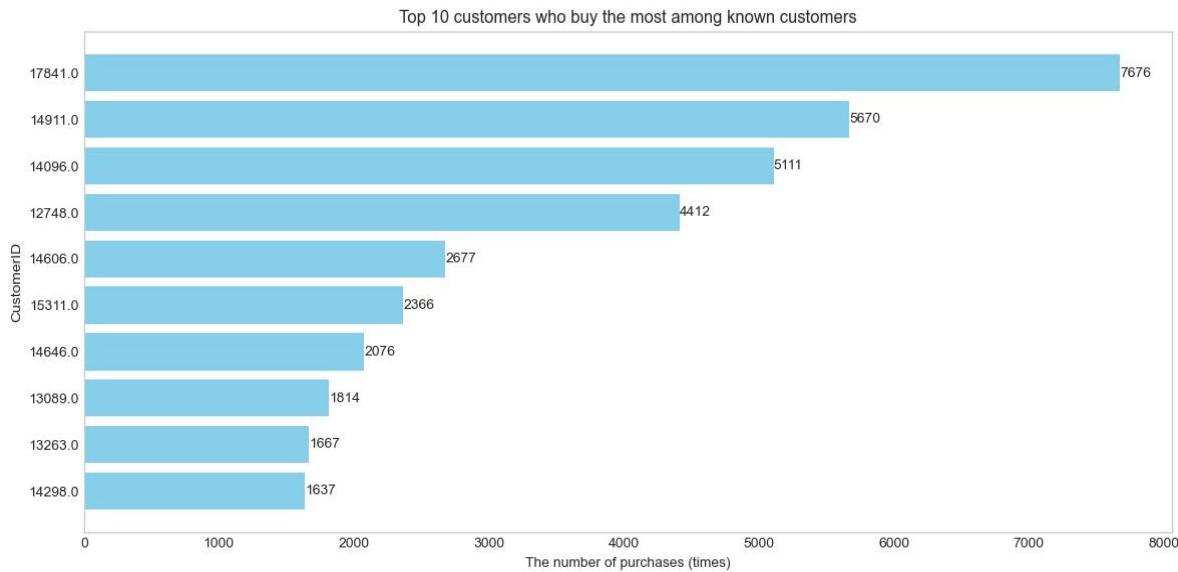
=> Kết quả:



- Có 4338 vị khách hàng xác định được danh tính (có CustomerID) chiếm tỉ lệ rất ít (25,2 %)
- Còn lại phần đông là các khách hàng không xác định được danh tính (Unknown) chiếm tỉ lệ rất lớn (74,8%)
- Xác định 10 khách hàng có số lượt mua nhiều nhất trong số các khách hàng xác định được danh tính (có CustomerID)

```
# Filtering out the rows whose CustomerID is 'unknown'  
data_filtered = data[data['CustomerID'] != "Unknown"]  
  
# Counting the number of occurrences of each CustomerID  
top_10_customers = data_filtered['CustomerID'].value_counts().head(10)  
  
# Drawing the horizontal bar chart  
plt.figure(figsize=(14, 6))  
bars = plt.barrh(top_10_customers.index, top_10_customers.values, color='skyblue')  
  
# Adding data labels to each bar  
for bar in bars:  
    width = bar.get_width()  
    plt.text(width + 0.3, bar.get_y() + bar.get_height()/2, str(int(width)), va='center')  
  
# Setting title and labels  
plt.xlabel('The number of purchases (times)')  
plt.ylabel('CustomerID')  
plt.title('Top 10 customers who buy the most among known customers')  
plt.gca().invert_yaxis() # Inverting the Y-axis so that the largest value is at the top  
plt.grid(False)  
plt.show()
```

=> Kết quả: Vị khách hàng có mã ID là ‘17841’ có số lượt mua nhiều nhất với 7676 lần mua



- Xác định số lượng các loại sản phẩm do vị khách hàng có mã ID là ‘17841’ mua và tìm ra 10 loại sản phẩm có số lượt mua nhiều nhất trong số các loại sản phẩm đó:

```
# Filtering rows where CustomerID is 17841
filtered_data = data[data['CustomerID'] == '17841']

# Converting CustomerID to string because its data type is float
data['CustomerID'] = data['CustomerID'].astype(str).str.strip('.0')

# Counting the number of unique values in the Description column
unique_descriptions_count = filtered_data['Description'].nunique()
print("Số lượng các loại hàng mà khách hàng có ID '17841' hay mua là:", unique_descriptions_count)

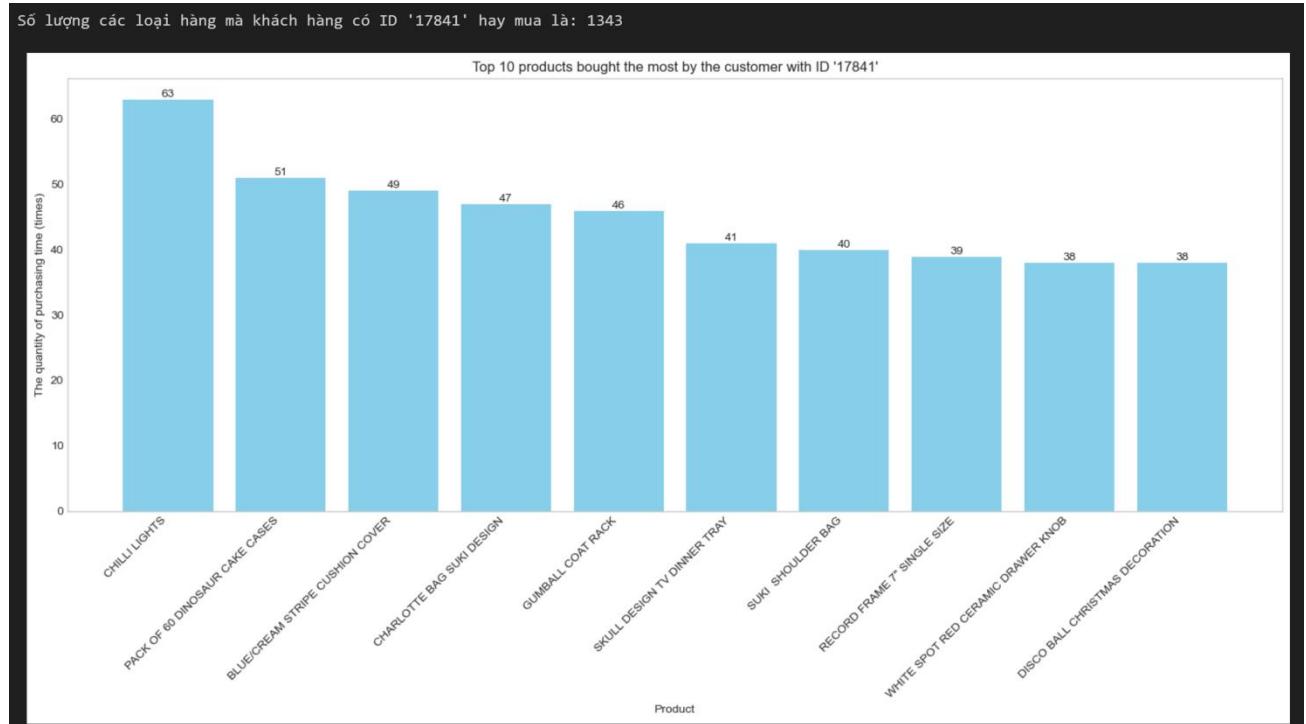
# Counting the occurrences of the Description values
description_counts = filtered_data['Description'].value_counts().nlargest(10)

# Drawing the bar chart
plt.figure(figsize=(15, 8))
bars = plt.bar(description_counts.index, description_counts.values, color='skyblue')

# Adding figures on each column
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, height, int(height), ha='center', va='bottom')

plt.title("Top 10 products bought the most by the customer with ID '17841'")
plt.xlabel('Product')
plt.ylabel('The quantity of purchasing time (times)')
plt.grid()
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

=> Kết quả:



- Vị khách hàng này đã mua tổng cộng 1343 mặt hàng khác nhau
- Sản phẩm ‘CHILLI LIGHTS’ được người này mua nhiều nhất với số lần mua là 63 lần

#### 4.2. Phân tích theo các yếu tố thời gian (InvoiceHour, InvoiceDay, InvoiceMonth, InvoiceYear)

- Xác định các năm có lượt mua hàng trong bộ dữ liệu và tìm ra tổng doanh thu, số lượng đơn hàng (luot giao dich) của các năm đó:

```
# Counting the number of unique CustomerID values
unique_year= data['InvoiceYear'].nunique()

# Printing results
print("Số lượng các năm có lượt mua hàng là: ", unique_year)
print("Các năm có lượt mua hàng là: ", data['InvoiceYear'].unique())

# Calculating total revenue and number of orders by year
summary = df.groupby('InvoiceYear').agg(
    TotalRevenue=('TotalPrice', 'sum'),
    OrderCount=('InvoiceNo', 'count')
).reset_index()

# Plotting a chart
fig, ax1 = plt.subplots(figsize=(16, 8))

# Column - Number of Orders (left axis)
bars1 = ax1.bar(summary['InvoiceYear'] - 0.2, summary['OrderCount'], width=0.4,
                 label='The quantity of purchases', color='tab:blue', alpha=0.7)
ax1.set_xlabel('Year')
ax1.set_ylabel('The quantity of purchases (times)', color='tab:blue')
ax1.tick_params(axis='y', labelcolor='tab:blue')

# Formatting the left vertical axis
ax1.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{int(x)}')) 

# The horizontal axis only displays 2010 and 2011
ax1.set_xticks([2010, 2011])
ax1.set_xticklabels([2010, 2011])

# Second axis - Total Revenue (right axis)
ax2 = ax1.twinx()
bars2 = ax2.bar(summary['InvoiceYear'] + 0.2, summary['TotalRevenue'], width=0.4,
                 label='Revenue', color='tab:red', alpha=0.7)
ax2.set_ylabel('Revenue', color='tab:red')
ax2.tick_params(axis='y', labelcolor='tab:red')

# Formatting the right y-axis (with thousand separators)
ax2.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{int(x)}'))
```

```

# Adding data labels on the columns
for bar in bars1:
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2, height, f'{int(height)}', ha='center', va='bottom', fontsize=8, color='tab:blue')

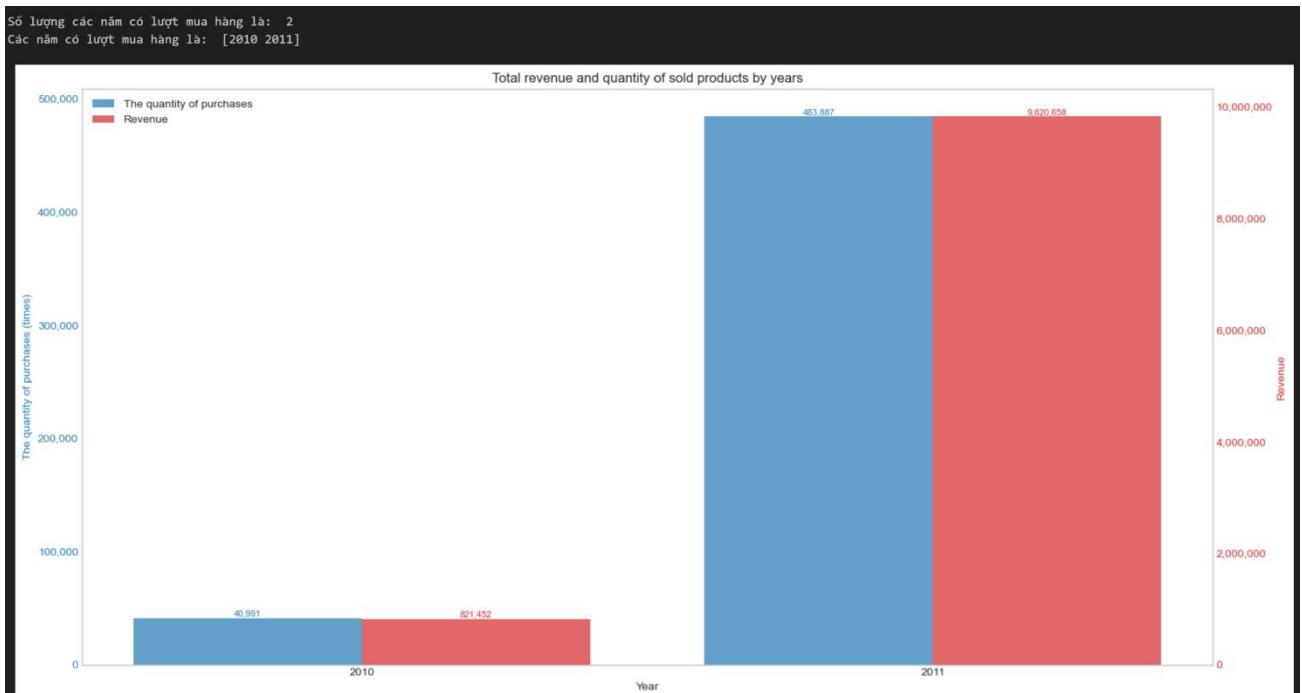
for bar in bars2:
    height = bar.get_height()
    ax2.text(bar.get_x() + bar.get_width()/2, height, f'{int(height)}', ha='center', va='bottom', fontsize=8, color='tab:red')

# Title and caption
plt.title('Total revenue and quantity of sold products by years')
ax2.grid()

# Showing
lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper left')
ax1.grid()
plt.tight_layout()
plt.show()

```

=> Kết quả:



- Các lượt giao dịch trong bộ dữ liệu diễn ra vào 2 năm 2010, 2011
- Năm 2011 có cả doanh thu lẫn số lượng đơn hàng nhiều hơn năm 2010
- Xác định số lượng các loại sản phẩm bán được qua các năm:

```

# Grouping the data to get the number of unique Descriptions by year
unique_products = data.groupby('InvoiceYear')['Description'].nunique()

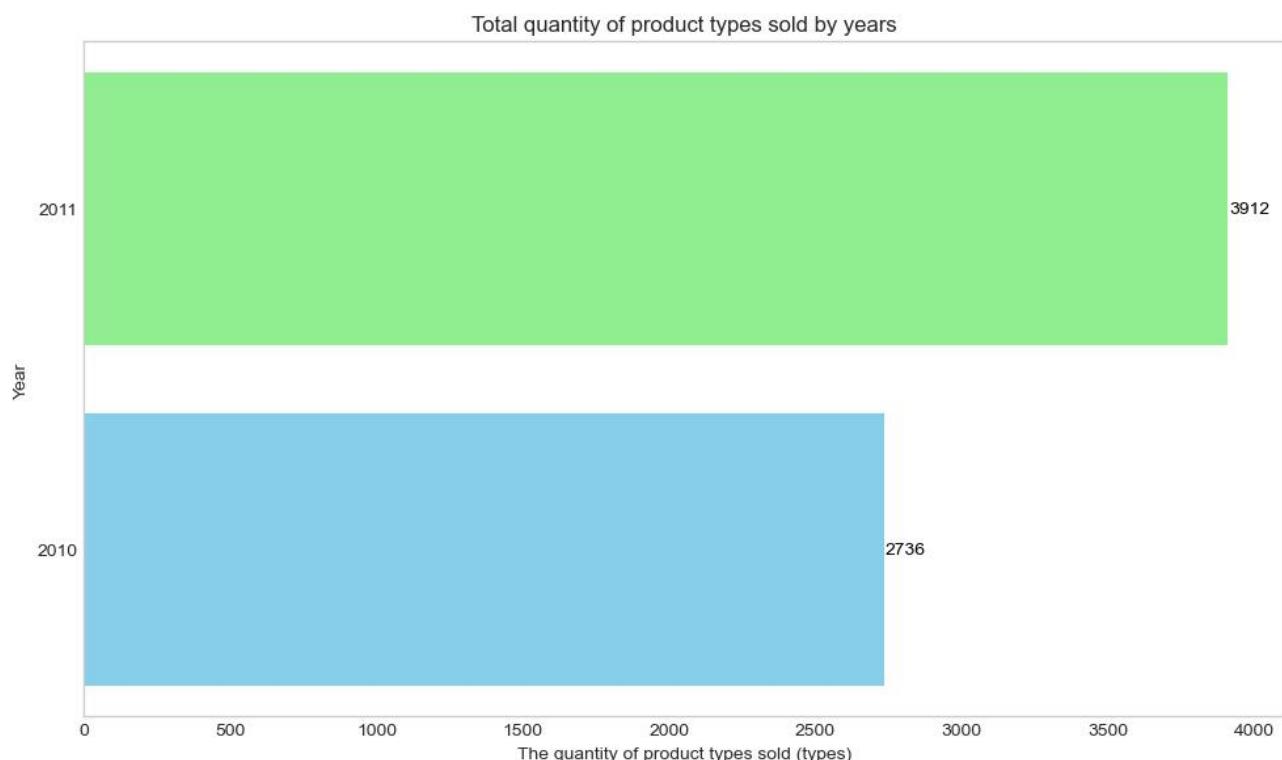
# Drawing a horizontal bar chart
fig, ax = plt.subplots(figsize=(10, 6))
bars = ax.barh(unique_products.index.astype(str), unique_products.values, color=['skyblue', 'lightgreen'])

# Adding data labels on each bar
for bar in bars:
    width = bar.get_width()
    ax.text(width + 5, bar.get_y() + bar.get_height() / 2,
            str(width),
            va='center', ha='left', fontsize=10, color='black')

# Setting title and labels
ax.set_xlabel('The quantity of product types sold (types)')
ax.set_ylabel('Year')
ax.set_title('Total quantity of product types sold by years')
ax.grid(False)
print('Số lượng các mặt hàng được mua qua các năm:')
plt.tight_layout()
plt.show()

```

=> Kết quả: Năm 2011 bán được nhiều loại sản phẩm hơn (3912 loại) so với năm 2010 (2736 loại)



- Xác định số lượng các đơn hàng (các lượt giao dịch) bán được qua các tháng của các năm:

```

order_counts = data.groupby(['InvoiceYear', 'InvoiceMonth']).size().reset_index(name='OrderCount')

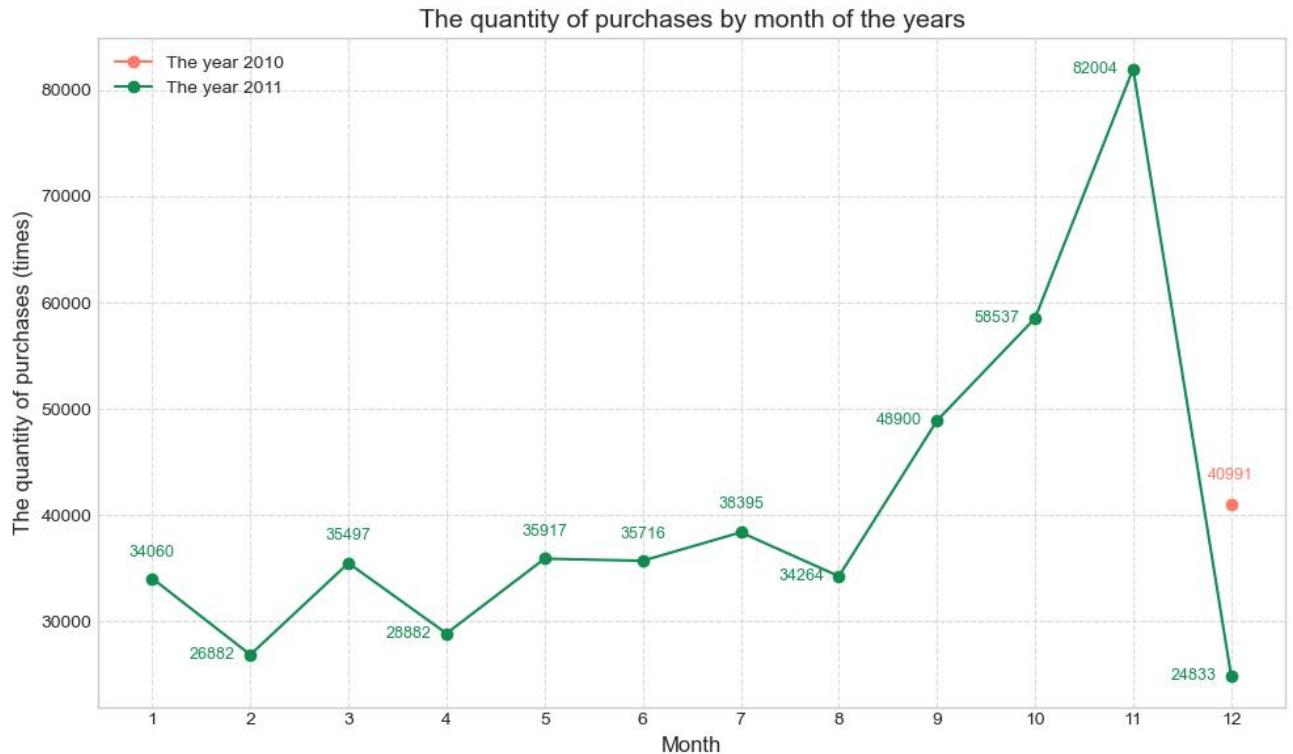
# Plotting the chart
plt.figure(figsize=(10, 6))
colors = {2010: "#FF7869", 2011: "#138B51"}

# Formatting the right y-axis (with comma as thousand separator)
ax2.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{int(x)}:,}'))
for year in data['InvoiceYear'].unique():
    year_data = order_counts[order_counts['InvoiceYear'] == year]
    plt.plot(year_data['InvoiceMonth'], year_data['OrderCount'], marker='o', color=colors[year], label=f'The year {year}')
    # Caption
    for x, y in zip(year_data['InvoiceMonth'], year_data['OrderCount']):
        # Condition to shift left
        if year == 2011 and x in [2, 4, 8, 9, 10, 12]:
            plt.text(x - 0.15, y, str(y), ha='right', va='center', fontsize=9, color=colors[year])
        elif year == 2011 and x == 11:
            plt.text(x - 0.15, y, str(y), ha='right', va='center', fontsize=9, color=colors[year])
        else:
            plt.text(x, y * 1.05, str(y), ha='center', va='bottom', fontsize=9, color=colors[year])

print("Số lượng các lượt mua hàng qua các tháng của các năm:")
plt.title('The quantity of purchases by month of the years', fontsize=14)
plt.xlabel('Month', fontsize=12)
plt.ylabel('The quantity of purchases (times)', fontsize=12)
plt.xticks(range(1, 13))
plt.legend()
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

```

=> Kết quả:



- Năm 2010 chỉ có mỗi tháng 12 có lượt giao dịch mua bán (có đơn hàng bán được) với 40991 lượt
- Năm 2011 thì hầu hết các tháng đều có lượt giao dịch mua bán (có đơn hàng bán được), trong đó tháng 11 có nhiều lượt nhất với 82004 lượt
- Xác định doanh thu qua các tháng của năm 2011:

```

# Filtering data for the year 2011
data_2011 = data[data['InvoiceYear'] == 2011]

# Calculating the total revenue for each month
monthly_revenue = data_2011.groupby('InvoiceMonth')['TotalPrice'].sum()

# Ensuring all 12 months are present (if any month is missing, set its revenue to 0)
monthly_revenue = monthly_revenue.reindex(range(1, 13), fill_value=0)

# Plotting a bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(monthly_revenue.index, monthly_revenue.values, color='yellow')

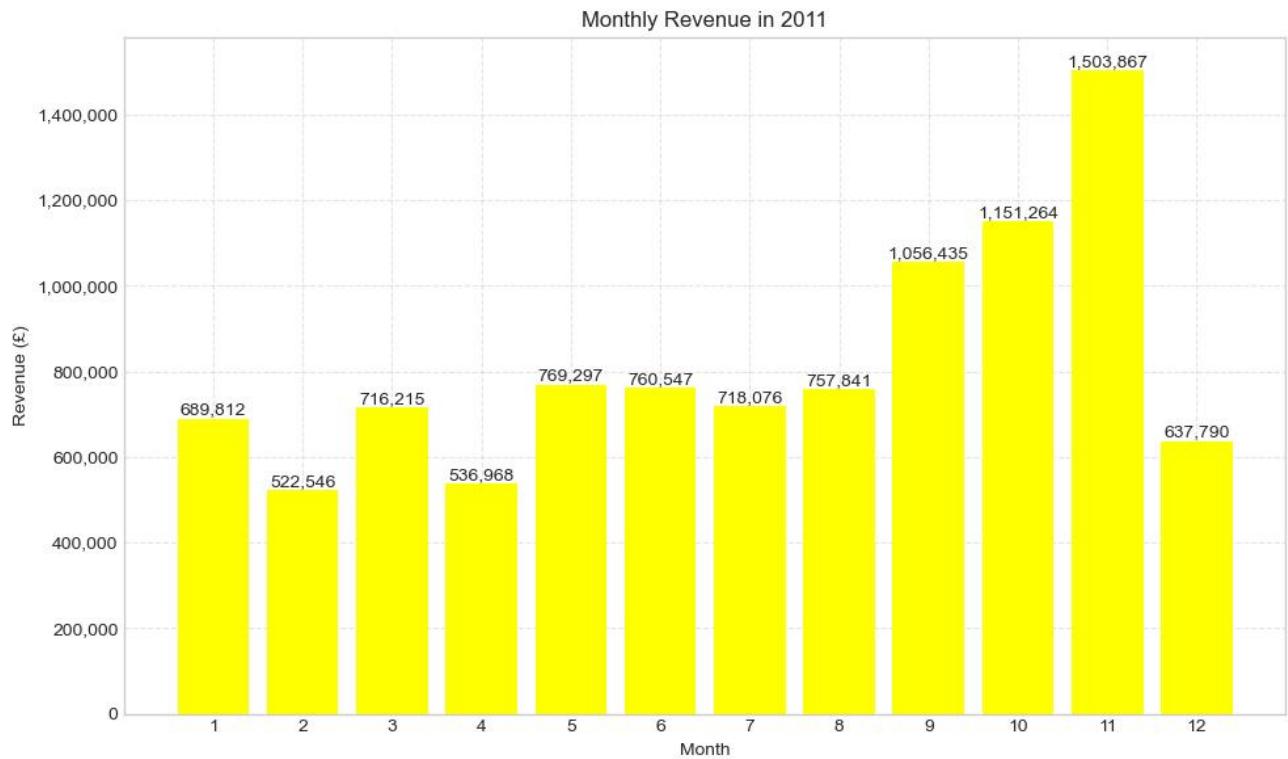
# Adding data labels on each column
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height,
              f'{height:.0f}', ha='center', va='bottom', fontsize=10)

# Formating y-axis to avoid scientific notation
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{x:.0f}'))

print('Doanh thu các tháng năm 2011:')
# Setting title and labels
plt.title('Monthly Revenue in 2011')
plt.xlabel('Month')
plt.ylabel('Revenue (£)')
plt.xticks(range(1, 13), [f'{m}' for m in range(1, 13)])
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.grid(axis='x', linestyle='--', alpha=0.5)
# Displaying the chart
plt.tight_layout()
plt.show()

```

=> Kết quả: Vẫn là tháng 11 có doanh thu cao nhất (1503867 bảng) và tháng 2 có doanh thu thấp nhất (522546 bảng)



- Tìm ra tổng số lượng các loại sản phẩm khác nhau bán được và các đơn hàng được đặt theo các ngày trong tuần (InvoiceWeekday) của tháng 11 năm 2011:

```
# Filtering data for InvoiceYear = 2011 and InvoiceMonth = 11
filtered_data = data[(data['InvoiceYear'] == 2011) & (data['InvoiceMonth'] == 11)]

# Counting the number of unique Weekday values
weekday_counts = filtered_data['InvoiceWeekday'].value_counts()

# Ensure the x-axis includes all 7 days from Monday to Sunday
ordered_days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
weekday_counts = weekday_counts.reindex(ordered_days, fill_value=0)

# Counting the number of unique Description values
unique_descriptions_count = filtered_data['Description'].nunique()

# Printing the result to the screen
print("- Số lượng các mặt hàng bán được trong tháng 11, năm 2011 là:", unique_descriptions_count, "mặt hàng")
```

```

# Drawing a line chart
plt.figure(figsize=(11, 6))
plt.plot(weekday_counts.index, weekday_counts.values, marker='o', linestyle='-', color='b')

# Adding data annotations on each point with position adjustment
for day, count in zip(weekday_counts.index, weekday_counts.values):
    # Adjusting positions
    if day in ['Thursday', 'Friday', 'Saturday', 'Sunday']:
        y_offset = 1000
    elif day in ['Monday', 'Tuesday', 'Wednesday']:
        y_offset = -1000
    else:
        y_offset = 0

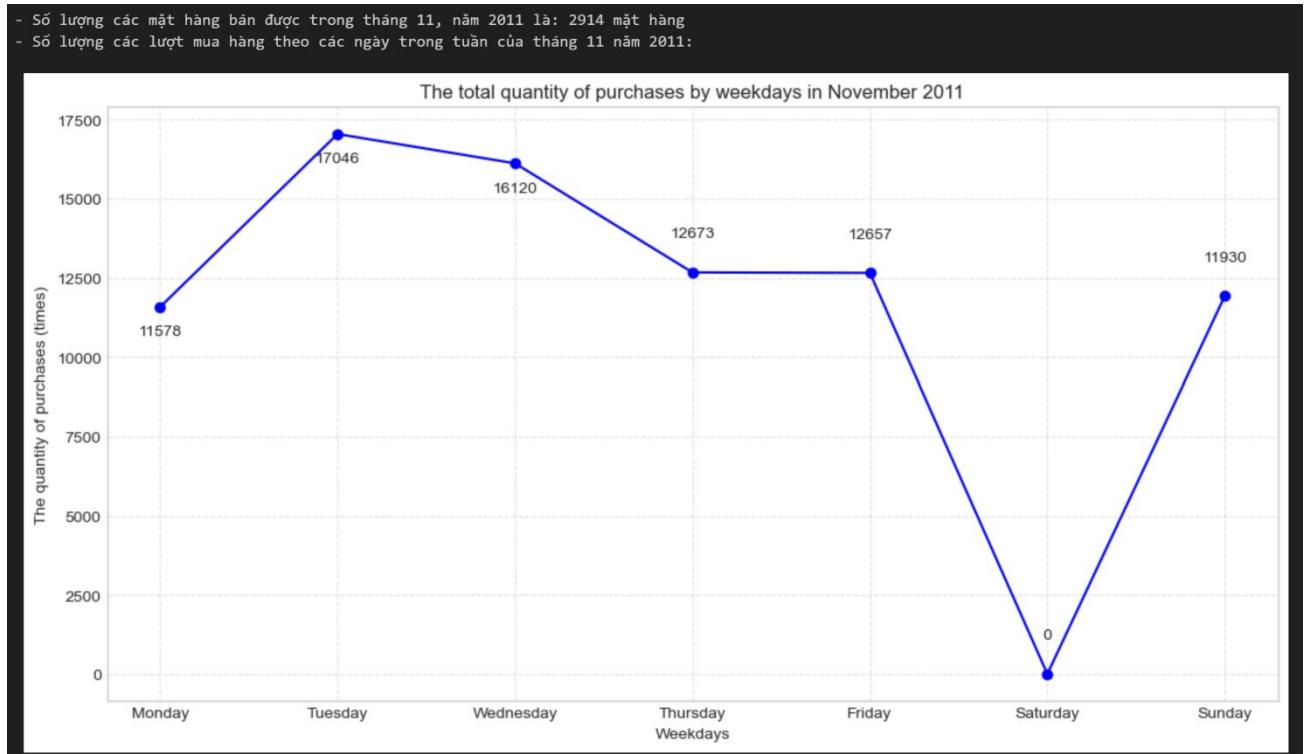
    plt.text(day, count + y_offset, str(int(count)), ha='center', va='bottom', fontsize=10)

# Setting title and labels
plt.title('The total quantity of purchases by weekdays in November 2011')
plt.xlabel('Weekdays')
plt.ylabel('The quantity of purchases (times)')
plt.xticks(ordered_days) # Ensuring the x-axis is in the correct order
plt.grid(True, linestyle='--', alpha=0.5)

print("- Số lượng các lượt mua hàng theo các ngày trong tuần của tháng 11 năm 2011:")
# Displaying the chart
plt.tight_layout()
plt.show()

```

=> Kết quả:



- Tháng 11, năm 2011 bán được tổng cộng 2914 loại sản phẩm
- Không có một đơn hàng nào được đặt mua vào các ngày thứ 7 (Saturday) của tháng và các ngày thứ 3 (Tuesday) của tháng có tổng số lượng đơn hàng bán được nhiều nhất

- Tìm ra số lượng các ngày có lượt giao dịch mua bán của tháng 11, năm 2011:

```
# Filtering for InvoiceMonth = 11 and InvoiceYear = 2011
november_2011 = data[(data['InvoiceMonth'] == 11) & (data['InvoiceYear'] == 2011)]

# Counting unique InvoiceDay values
unique_days_count = november_2011['InvoiceDay'].nunique()

# Printing the result
print(f"Số lượng các ngày có lượt mua hàng trong tháng 11, năm 2011: {unique_days_count}")
```

=> Kết quả: Số lượng các ngày có lượt mua hàng trong tháng 11, năm 2011: 26

- Xem các ngày có lượt giao dịch mua bán của tháng 11, năm 2011 đó là các ngày nào:

```
# Getting unique InvoiceDay values and sort them
unique_days = sorted(november_2011['InvoiceDay'].unique())
print(f"Các ngày có lượt mua hàng trong tháng 11, năm 2011: {unique_days}")
```

=> Kết quả: Các ngày có lượt mua hàng trong tháng 11, năm 2011: [1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30]

- Vẽ biểu đồ phân tán thể hiện doanh thu (TotalPrice) của các ngày trong tháng 11, năm 2011:

```
# Filtering data for November 2011
nov_2011 = data[(data['InvoiceMonth'] == 11) & (data['InvoiceYear'] == 2011)]

# Grouping by InvoiceDay and calculate total revenue
daily_revenue = nov_2011.groupby('InvoiceDay')['TotalPrice'].sum().sort_index()

# Plotting scatter chart
plt.figure(figsize=(12, 6))
plt.scatter(daily_revenue.index, daily_revenue.values, color='tab:red', s=60)

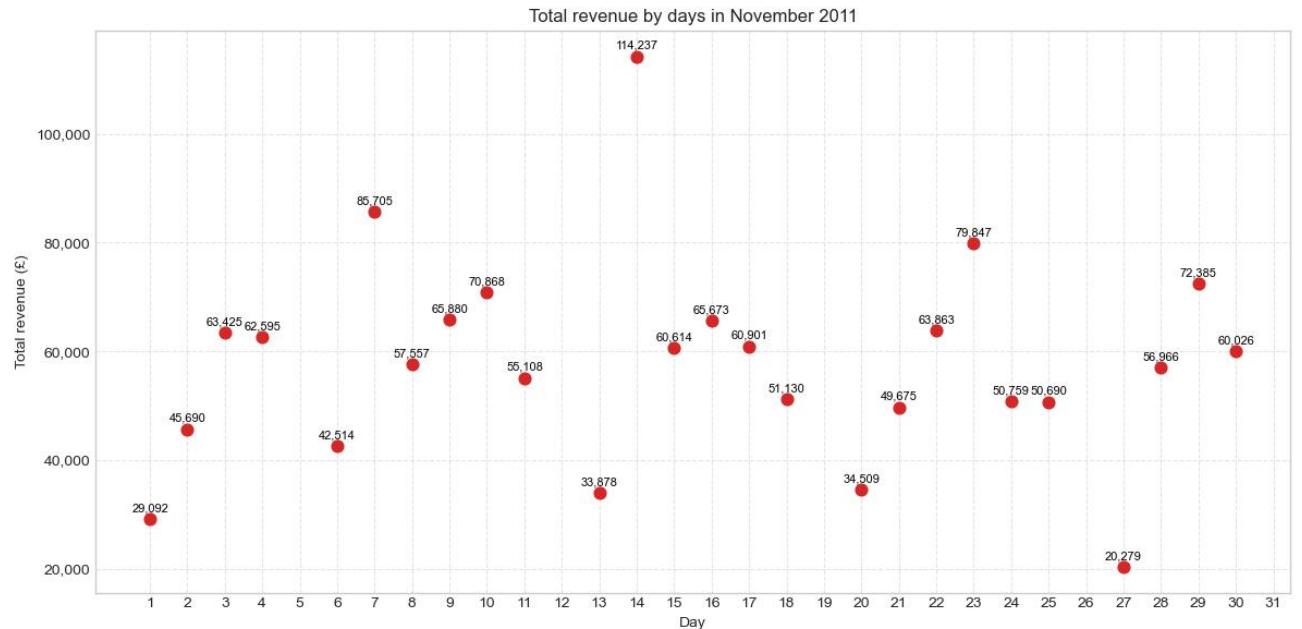
# Annotating each point with total revenue
for day, revenue in daily_revenue.items():
    plt.annotate(f'{revenue:.0f}', (day, revenue),
                 textcoords="offset points",
                 xytext=(0, 5),
                 ha='center', fontsize=8, color='black')

# Formating y-axis
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{x:.0f}'))

# Setting chart title and labels
plt.title('Total revenue by days in November 2011')
plt.xlabel('Day')
plt.ylabel('Total revenue (£)')
plt.xticks(range(1, 32)) # Days 1 to 31
plt.grid(False, linestyle='--', alpha=0.5)

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

=> Kết quả: Ngày có doanh thu cao nhất của tháng là ngày 14 với doanh thu là 114237 bảng và ngày có doanh thu thấp nhất của tháng là ngày 27 với doanh thu là 20279 bảng



- Tìm ra số lượng các khung giờ có lượt mua hàng trong ngày 14 tháng 11 năm 2011:

```
# Filtering data for 14 November 2011
day_14_nov_2011 = data[
    (data['InvoiceDay'] == 14) &
    (data['InvoiceMonth'] == 11) &
    (data['InvoiceYear'] == 2011)
]

# Getting unique InvoiceHour values and sort them
unique_hours = sorted(day_14_nov_2011['InvoiceHour'].unique())

# Counting the number of unique hours
unique_hours_count = len(unique_hours)

# Printing the results
print(f"Số lượng các khung giờ có lượt mua hàng trong ngày 14 tháng 11 năm 2011 là: {unique_hours_count}")
```

=> Kết quả: Số lượng các khung giờ có lượt mua hàng trong ngày 14 tháng 11 năm 2011 là: 10

- Xem các khung giờ có lượt giao dịch mua bán của ngày 14 tháng 11 năm 2011 đó là các ngày nào:

```
# Getting unique InvoiceHour values and sort them
unique_hours = sorted(day_14_nov_2011['InvoiceHour'].unique())
print(f"Các khung giờ có lượt mua hàng trong ngày 14 tháng 11 năm 2011 là: {unique_hours}")
```

=> Kết quả: Các khung giờ có lượt mua hàng trong ngày 14 tháng 11 năm 2011 là: [8, 9, 10, 11, 12, 13, 14, 15, 16, 17]

- Xem số lượng các lượt giao dịch mua bán của các khung giờ trong ngày 14 tháng 11 năm 2011:

```
# Filtering data for 14 November 2011
day_14_nov_2011 = data[
    (data['InvoiceDay'] == 14) &
    (data['InvoiceMonth'] == 11) &
    (data['InvoiceYear'] == 2011)
]

# Filtering only for the specified hours
selected_hours = [8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
hourly_order_count = (
    day_14_nov_2011[day_14_nov_2011['InvoiceHour'].isin(selected_hours)]
    ['InvoiceHour']
    .value_counts()
    .reindex(selected_hours, fill_value=0)
)

# Plotting bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(hourly_order_count.index, hourly_order_count.values, color='tab:orange')

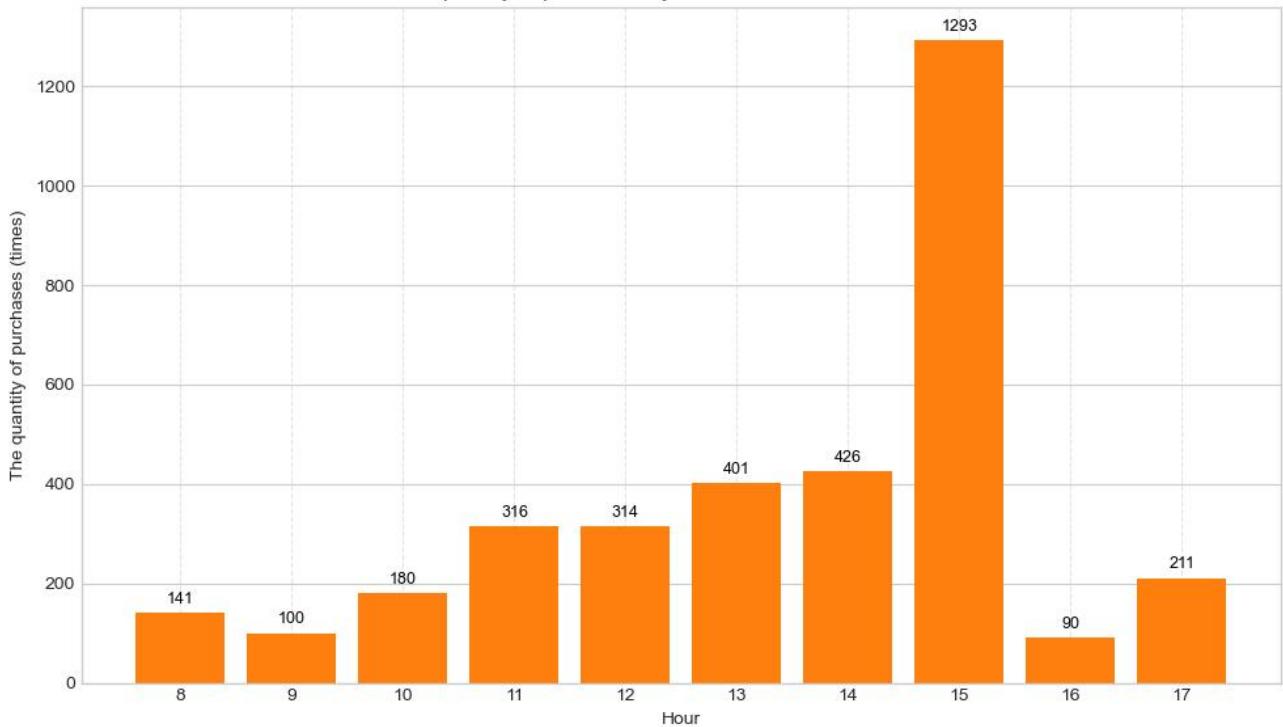
# Annotating number of orders on top of each bar
for bar in bars:
    height = bar.get_height()
    plt.annotate(f'{height}',
                 xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 3), # Slight offset above bar
                 textcoords="offset points",
                 ha='center', va='bottom', fontsize=9, color='black')

# Setting chart title and labels
plt.title('The quantity of purchases by hours on 14th November 2011')
plt.xlabel("Hour")
plt.ylabel('The quantity of purchases (times)')
plt.xticks(selected_hours)
plt.grid(axis='x', linestyle='--', alpha=0.5)

plt.tight_layout()
print(f"Số lượng các lượt mua hàng theo các giờ trong ngày 14 tháng 11 năm 2011:")
plt.show()
```

=> Kết quả: Khung giờ 15 giờ (3:00 PM) có số lượt giao dịch mua bán hàng nhiều nhất với 1293 lượt và khung giờ 16 giờ (4:00 PM) có số lượt giao dịch mua bán hàng ít nhất với 90 lượt

The quantity of purchases by hours on 14th November 2011



- Tìm ra số lượng thống kê và tỉ lệ phần trăm số lượng các đơn hàng được đặt theo mùa (InvoiceSeason) của các năm:

```

# Grouping by Season and Year, count occurrences
season_year_counts = data.groupby(['Season', 'InvoiceYear']).size().unstack(fill_value=0)

# Renaming columns for clarity
season_year_counts.columns = ['2010', '2011']

# Adding a Total row
season_year_counts.loc['Total'] = season_year_counts.sum()

# Resetting index to have Season as a column
summary_table = season_year_counts.reset_index()

# Printing the 5-row, 3-column table
print("** Số lượng các lượt mua hàng theo mùa của các năm**")
print(' ')
print(summary_table)
print(' ')

# Counting the number of unique Season values for each InvoiceYear
season_counts = data.groupby('InvoiceYear')['Season'].value_counts().unstack(fill_value=0)

print("Ti lệ phần trăm tổng số lượng các lượt mua hàng theo mùa của các năm**")
print(' ')
# Drawing the pie chart for each InvoiceYear
for year in season_counts.index:
    plt.figure(figsize=(6, 6))
    season_counts.loc[year].plot(kind='pie', autopct='%1.1f%%', startangle=90, counterclock=False)
    plt.title(f'Percentage of the total quantity of purchases by seasons in {year}')
    plt.ylabel('')
    plt.show()

```

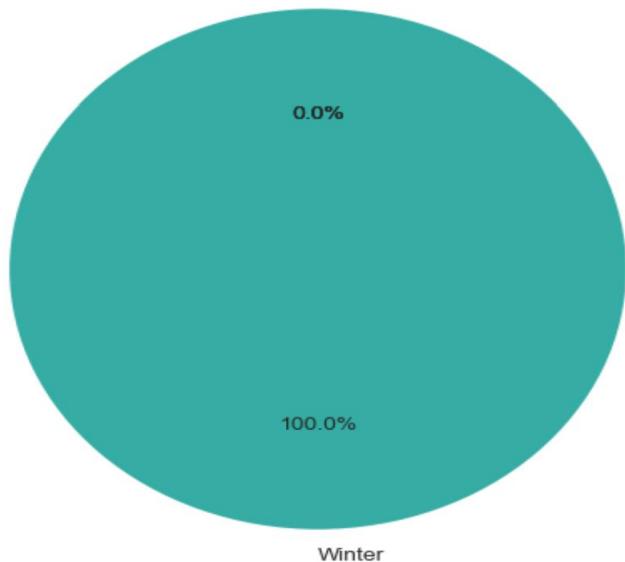
=> Kết quả:

\*\* Số lượng các lượt mua hàng theo mùa của các năm\*\*

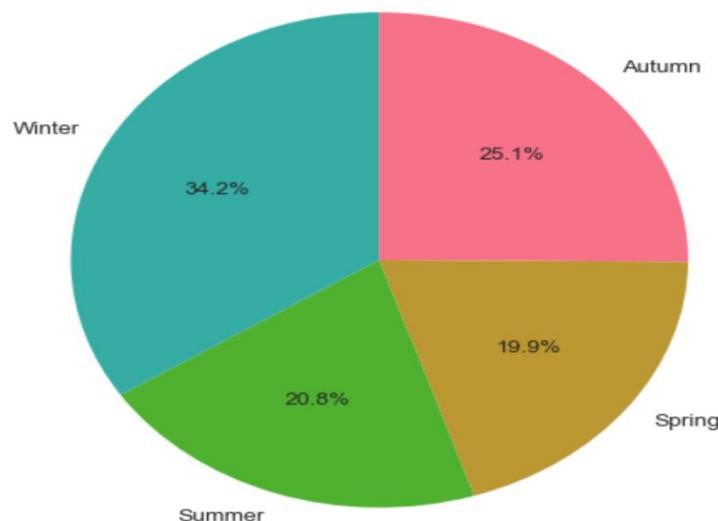
Season	2010	2011
0 Autumn	0	121559
1 Spring	0	96439
2 Summer	0	100515
3 Winter	40991	165374
4 Total	40991	483887

\*\*Tí lệ phần trăm tổng số lượng các lượt mua hàng theo mùa của các năm\*\*

Percentage of the total quantity of purchases by seasons in 2010



Percentage of the total quantity of purchases by seasons in 2011



- Năm 2010 các lượt giao dịch mua bán đều rơi vào mùa đông (100% Winter)
- Năm 2011 các lượt giao dịch mua bán rơi vào mùa đông là nhiều nhất, chiếm tỉ lệ cao nhất (34.2 % Winter) và các lượt giao dịch mua bán rơi vào xuân là ít nhất, chiếm tỉ lệ thấp nhất (19.9% Spring)

#### 4.3 . Phân tích theo yếu tố các quốc gia (Country)

- Tìm ra số lượng các quốc gia tham gia bán hàng cho các lượt giao dịch và 10 quốc gia có doanh thu cao nhất cùng với số lượng các đơn hàng bán được tương ứng trong các quốc gia đó:

```
# Counting the number of unique Country values
num_countries = data['Country'].nunique()

# Printing to screen
print(f"- Các đơn hàng đến từ {num_countries} quốc gia")

# Counting the number of orders (occurrences) for each country
order_counts = data['Country'].value_counts()

# Calculating the total revenue (TotalPrice) for each country
total_revenue = df.groupby('Country')['TotalPrice'].sum()

# Getting the top 10 countries with the highest total revenue
top_countries = total_revenue.sort_values(ascending=False).head(10).index

# Creating a combined DataFrame
summary = pd.DataFrame({
    'OrderCount': order_counts[top_countries],
    'TotalRevenue': total_revenue[top_countries]
}).reset_index().rename(columns={'index': 'Country'})

# Drawing a chart with two distinct y-axes
fig, ax1 = plt.subplots(figsize=(18, 7))

bar_width = 0.4
x = range(len(summary))

# Adding data labels for Order Count
bars1 = ax1.bar([i - bar_width/2 for i in x], summary['OrderCount'], width=bar_width, label='The quantity of purchases', color='red')
ax1.set_ylabel('The quantity of purchases (times)', color='black')
ax1.tick_params(axis='y', labelcolor='black')
ax1.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{int(x)}')) # Định dạng số nguyên có dấu phẩy

# Adding data labels for Order Count
for bar in bars1:
    height = bar.get_height()
    ax1.annotate(f'{int(height)}',
                 xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 3),
                 textcoords="offset points",
                 ha='center', va='bottom', color='black')
```

```

# Creating a second y-axis on the right for Total Revenue
ax2 = ax1.twinx()
bars2 = ax2.bar([i + bar_width/2 for i in x], summary['TotalRevenue'], width=bar_width, label='Total Revenue', color='green')
ax2.set_ylabel('Total Revenue (€)', color='black')
ax2.tick_params(axis='y', labelcolor='black')
ax2.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{int(x):,}')) # Định dạng số nguyên có dấu phẩy

# Adding data labels for Total Revenue
for bar in bars2:
    height = bar.get_height()
    ax2.annotate(f'{int(height):,}', xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 3), textcoords="offset points",
                 ha='center', va='bottom', color='black')

# Adding a legend for both axes
lines_labels = [ax.get_legend_handles_labels() for ax in [ax1, ax2]]
handles, labels = [sum(lol, []) for lol in zip(*lines_labels)]
ax1.legend(handles, labels, loc='upper right')

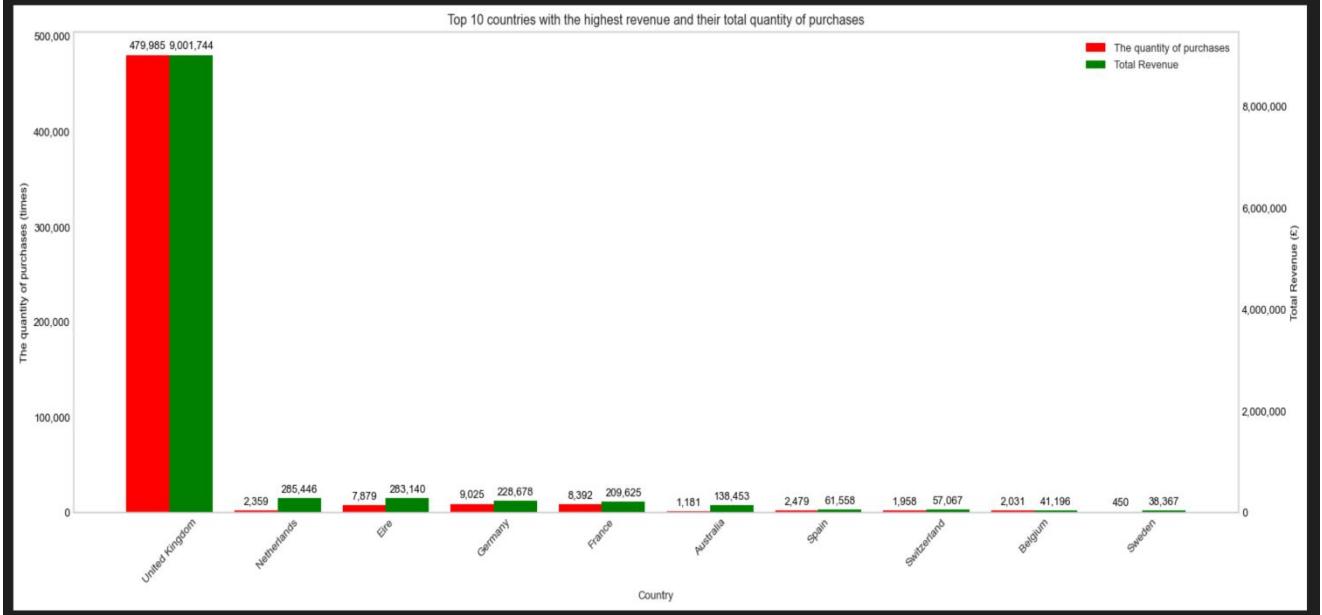
# Setting title and axis labels
ax1.set_xlabel('Country')
ax1.set_title('Top 10 countries with the highest revenue and their total quantity of purchases')
ax1.set_xticks(x)
ax1.set_xticklabels(summary['Country'], rotation=45)
ax1.grid()

fig.tight_layout()
plt.grid()
print('- Top 10 quốc gia có doanh thu cao nhất và số lượng đơn hàng bán được tương ứng:')
plt.show()

```

=> Kết quả: Có 38 quốc gia tham gia bán hàng cho các lượt giao dịch và quốc gia có doanh thu cao nhất là nước Anh (United Kingdom) với doanh thu 9001744 bảng, 479985 đơn hàng được đặt tổng cộng

- Các đơn hàng đến từ 38 quốc gia
- Top 10 quốc gia có doanh thu cao nhất và số lượng đơn hàng bán được tương ứng:



- Xác định 10 quốc gia có số lượt giao dịch (số đơn hàng được đặt) nhiều nhất và doanh thu tương ứng:

```

# Counting the number of orders (occurrences) for each country
order_counts = data['Country'].value_counts()

# Calculating the total revenue (TotalPrice) for each country
total_revenue = df.groupby('Country')['TotalPrice'].sum()

# Getting the top 10 countries with the highest total revenue
top_countries = order_counts.sort_values(ascending=False).head(10).index

# Creating a combined DataFrame
summary = pd.DataFrame({
    'OrderCount': order_counts[top_countries],
    'TotalRevenue': total_revenue[top_countries]
}).reset_index().rename(columns={'index': 'Country'})

# Plotting a chart with two separate y-axes
fig, ax1 = plt.subplots(figsize=(18, 7))

bar_width = 0.4
x = range(len(summary))

# Plotting the number of orders on the left y-axis
bars1 = ax1.bar([i - bar_width/2 for i in x], summary['TotalRevenue'], width=bar_width, label='Total Revenue', color='yellow')
ax1.set_ylabel('Total Revenue (£)', color='black')
ax1.tick_params(axis='y', labelcolor='black')
ax1.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{int(x):,}')) # Định dạng số nguyên có dấu phẩy

# Adding data labels for Order Count
for bar in bars1:
    height = bar.get_height()
    ax1.annotate(f'{int(height)}',
                 xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 3),
                 textcoords="offset points",
                 ha='center', va='bottom', color='black')

# Creating a second y-axis on the right for Total Revenue
ax2 = ax1.twinx()
bars2 = ax2.bar([i + bar_width/2 for i in x], summary['OrderCount'], width=bar_width, label='The quantity of purchases', color='blue')
ax2.set_ylabel('The quantity of purchases (times)', color='black')
ax2.tick_params(axis='y', labelcolor='black')
ax2.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{int(x):,}')) # Định dạng số nguyên có dấu phẩy

# Adding data labels for Total Revenue
for bar in bars2:
    height = bar.get_height()
    ax2.annotate(f'{int(height)}',
                 xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 3),
                 textcoords="offset points",
                 ha='center', va='bottom', color='black')

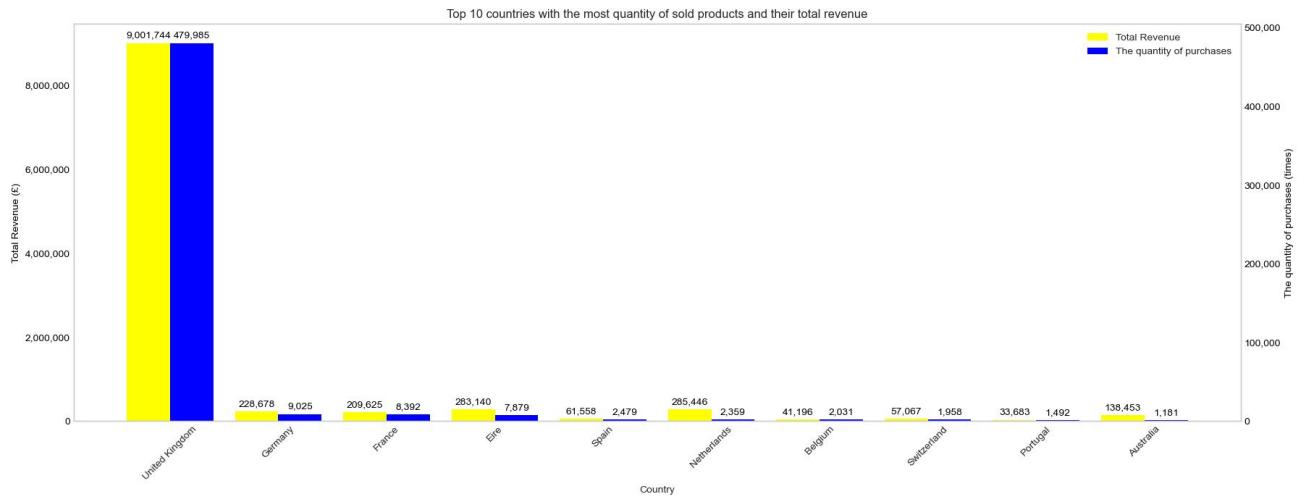
# Adding a legend for both axes
lines_labels = [ax.get_legend_handles_labels() for ax in [ax1, ax2]]
handles, labels = [sum(lol, []) for lol in zip(*lines_labels)]
ax1.legend(handles, labels, loc='upper right')

# Setting title and axis labels
ax1.set_xlabel('Country')
ax1.set_title('Top 10 countries with the most quantity of sold products and their total revenue')
ax1.set_xticks(x)
ax1.set_xticklabels(summary['Country'], rotation=45)
ax1.grid()

fig.tight_layout()
plt.grid()
print('Top 10 quốc gia có số lượng đơn hàng bán được nhiều nhất và doanh thu tương ứng:')
plt.show()

```

=> Kết quả: Nước Anh (United Kingdom) vẫn là quốc gia có số lượt giao dịch (số đơn hàng được đặt) nhiều nhất với 479985 lượt và doanh thu là 9001744 bảng



- Xác định số lượng các loại sản phẩm được mua từ nước Anh (United Kingdom):

```
# Filtering data for United Kingdom
uk_data = data[data['Country'] == 'United Kingdom']

# Counting the number of unique Descriptions
num_unique_descriptions = uk_data['Description'].nunique()

# Printing to screen
print(f"Số lượng các loại sản phẩm được mua thuộc nước Anh (United Kingdom): {num_unique_descriptions}")

Số lượng các loại sản phẩm được mua thuộc nước Anh (United Kingdom): 3996
```

- Xác định 10 loại sản phẩm có doanh thu cao nhất đến từ nước Anh (United Kingdom):

```
# Filtering data for United Kingdom
uk_data = data[data['Country'] == 'United Kingdom']

# Calculating total revenue by product
item_revenue = uk_data.groupby('Description')['TotalPrice'].sum().sort_values(ascending=False)

# Getting the 10 products with the highest revenue
top_10_items = item_revenue.head(10)

# Plotting a line chart
plt.figure(figsize=(16, 8))
plt.plot(top_10_items.index, top_10_items.values, marker='o', color='tab:blue', linestyle='--')

# The list of products requiring annotation position adjustment
move_up_items = [
    'PAPER CRAFT', 'LITTLE BIRDIE',
    'WHITE HANGING HEART T-LIGHT HOLDER'
]

move_down_items = [
    'DOTCOM POSTAGE',
    'REGENCY CAKESTAND 3 TIER',
]

# Adding data labels to each point
for i, (desc, value) in enumerate(top_10_items.items()):
    # Defaulting offset_y
    offset_y = 5
    # If the product needs to be moved up
    if desc in move_up_items:
        offset_y += 20
    # If the product needs to be moved down
    elif desc in move_down_items:
        offset_y -= 25

    plt.annotate(f'{value:.0f}', (i, value),
                textcoords="offset points",
                xytext=(0, offset_y),
                ha='center', fontsize=9, color='black')
```

```

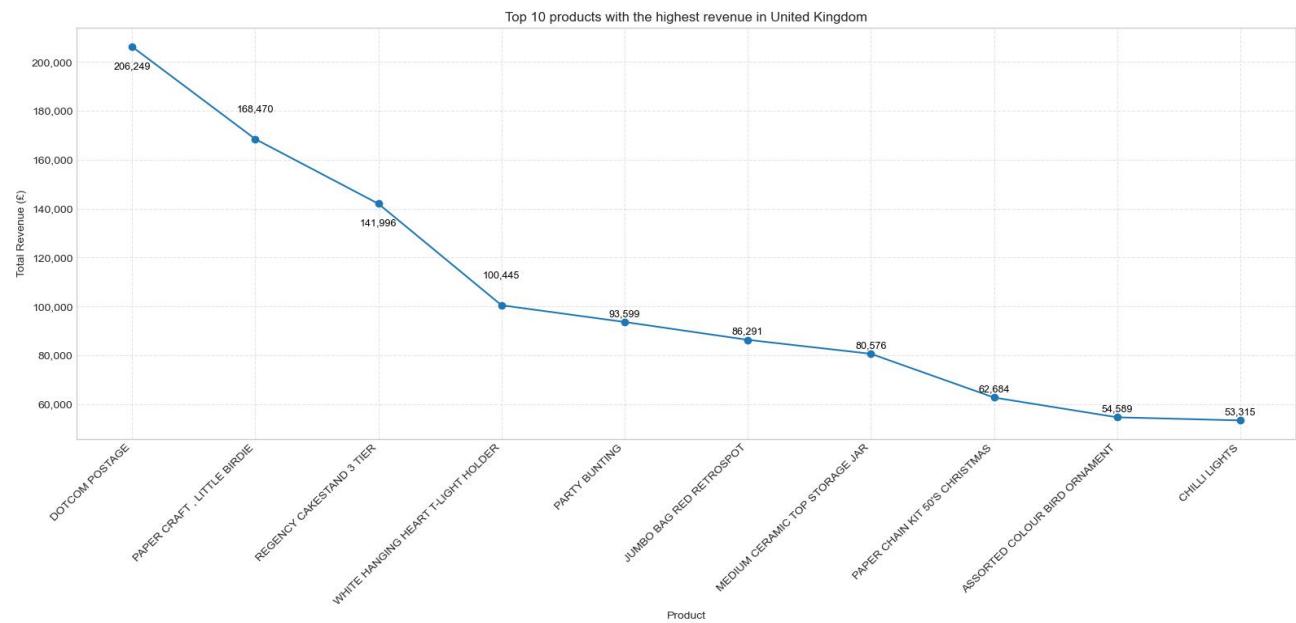
# Formating the y-axis not in 1e6 notation
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{x:.0f}'))

# Setting title and axis labels
plt.title('Top 10 products with the highest revenue in United Kingdom')
plt.xlabel('Product')
plt.ylabel('Total Revenue (£)')
plt.xticks(rotation=45, ha='right')

plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
print('Top 10 sản phẩm có doanh thu cao nhất thuộc về nước Anh (United Kingdom):')
plt.show()

```

=> Kết quả: Sản phẩm ‘DOTCOM POSTAGE’ có doanh thu cao nhất với doanh thu là 206249 bảng



- Xác định 10 loại sản phẩm có số lượng đơn hàng được đặt nhiều nhất từ nước Anh (United Kingdom):

```

# Filtering data for United Kingdom
uk_data = data[data['Country'] == 'United Kingdom']

# Counting the number of purchases for each product
item_order_count = uk_data['Description'].value_counts().head(10)

# Plotting a line chart
plt.figure(figsize=(14, 7))
plt.plot(item_order_count.index, item_order_count.values, marker='o', color='tab:orange', linestyle='--')

# The list of products that need annotation position adjustment
move_up_items = [
    'JUMBO BAG RED RETROSPOT',
    'REGENCY CAKESTAND 3 TIER'
]

# Adding data labels on each point
for i, (desc, value) in enumerate(item_order_count.items()):
    # Defaulting offset_y
    offset_y = 5
    # If the product needs to be moved up
    if desc in move_up_items:
        offset_y += 25

    plt.annotate(f'({value:.0f})',
                (i, value),
                textcoords="offset points",
                xytext=(0, offset_y),
                ha='center', fontsize=9, color='black')

```

```

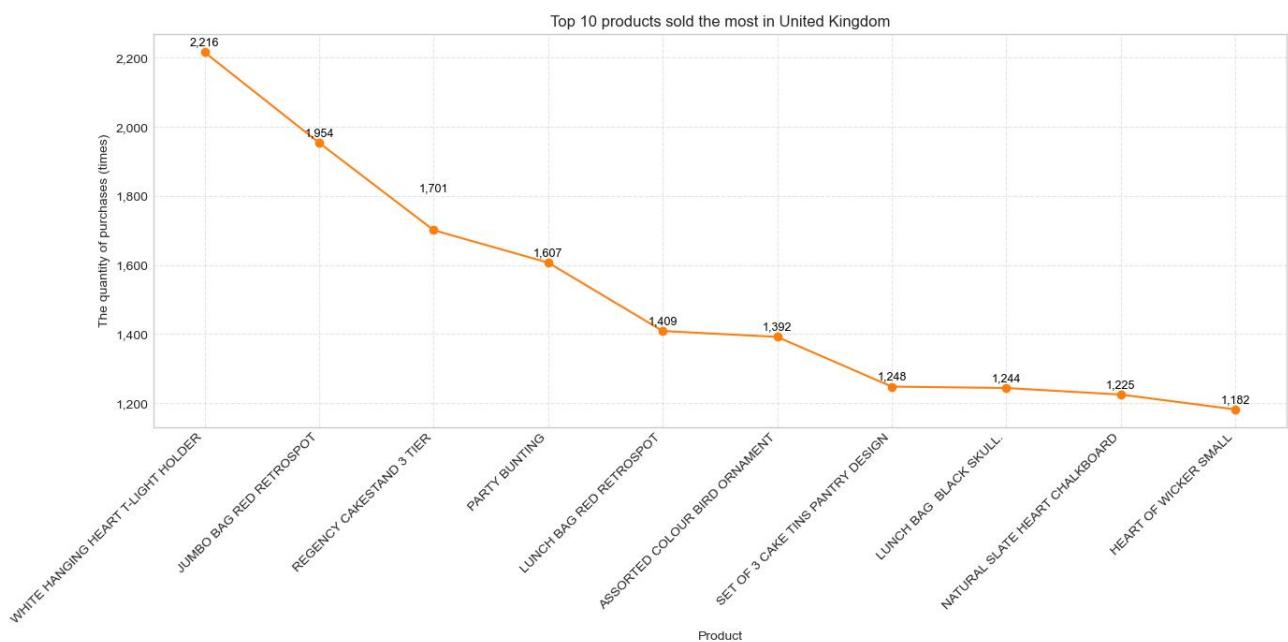
# Formating the y-axis not in 1e6 notation
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: f'{x:.0f}'))

# Setting title and axis labels
plt.title('Top 10 products sold the most in United Kingdom')
plt.xlabel('Product')
plt.ylabel('The quantity of purchases (times)')
plt.xticks(rotation=45, ha='right')

plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
print('Top 10 sản phẩm thuộc nước Anh (United Kingdom) được mua nhiều nhất:')
plt.show()

```

=> Kết quả: Sản phẩm ‘WHITE HANGING HEART T-LIGHT HOLDER’ có số lượng đơn hàng được đặt nhiều nhất với 2216 lượt đặt mua



### III. Tổng kết

#### 1. Nhận xét kết quả phân tích

- Khi phân tích theo yếu tố sản phẩm và khách hàng:
  - ✓ Sản phẩm bán chạy nhất về doanh thu là “DOTCOM POSTAGE” (206,249£) với 706 lượt mua, trong khi sản phẩm có lượt mua nhiều nhất là “WHITE HANGING HEART T-LIGHT HOLDER” (2,311 lượt, doanh thu 106,237£)

=> Doanh thu cao không nhất thiết đến từ tần suất mua hàng mà có thể từ giá trị đơn hàng

- ✓ Đa số giao dịch đến từ khách hàng không xác định (Unknown), chiếm 74.8% và chỉ vỏn vẹn 25.2% khách hàng có CustomerID

=> Điều này có lẽ là do phần đông các khách hàng tham gia mua chưa đăng ký tài khoản trên ứng dụng mua sắm trực tuyến

- ✓ Khách hàng có CustomerID 17841 là người mua tích cực nhất với 7,676 lượt mua và đã mua 1,343 loại sản phẩm khác nhau.

- Khi phân tích theo yếu tố theo thời gian:

- ✓ Giao dịch chỉ xuất hiện trong 2 năm (2010–2011). Năm 2011 vượt trội về cả doanh thu và số lượng đơn hàng so với 2010.
- ✓ Tháng 11/2011 là thời điểm “cao điểm” với 82,004 giao dịch và doanh thu 1,503,867£, có thể trùng với mùa lễ hội cuối năm (Black Friday, Giáng Sinh).
- ✓ Hoạt động mua bán chủ yếu diễn ra từ 6h đến 20h, với khung giờ 15h là thời gian sôi động nhất.

- Khi phân tích theo yếu tố quốc gia:

- ✓ Có 38 quốc gia tham gia giao dịch. United Kingdom (UK) dẫn đầu với 9001744£ doanh thu (chiếm gần 90% tổng doanh thu) và 479,985 giao dịch (gần như toàn bộ dữ liệu). Điều này chứng tỏ phần lớn hoạt động mua bán tập trung tại UK.
- ✓ Cả 2 sản phẩm dẫn đầu doanh thu và lượt mua cũng đến từ UK.

- Khi phân tích theo yếu tố mùa vụ: Các giao dịch của năm 2010 đều rơi vào mùa đông (Winter). Năm 2011, mùa đông tiếp tục chiếm tỷ lệ cao nhất (34.2%) trong khi mùa xuân (Spring) thấp nhất (19.9%). Điều này cũng có nhận định về sức mua tăng cao vào các kỳ nghỉ lễ cuối năm.

## 2. Đề xuất về mặt kinh doanh

- Tận dụng sản phẩm bán chạy để upsell/cross-sell:

- ✓ Với “WHITE HANGING HEART T-LIGHT HOLDER” là sản phẩm có lượt mua nhiều nhất và “DOTCOM POSTAGE” là sản phẩm doanh thu cao nhất, có thể tạo combo khuyến mãi hoặc gói quà tặng để khuyến khích khách hàng mua thêm các sản phẩm liên quan
- ✓ Triển khai chiến dịch quảng cáo tập trung cho 2 sản phẩm này vào các dịp cao điểm như Giáng Sinh hoặc Black Friday

- Có chính sách tăng tỷ lệ khách hàng xác định danh tính (CustomerID): Hiện tại 74.8% khách hàng là Unknown, nên cần khuyến khích khách hàng đăng ký tài khoản khi mua hàng để:

- ✓ Dễ quản lý hành vi mua sắm
- ✓ Cung cấp ưu đãi cá nhân hóa (personalized promotions)

- ✓ Có thể áp dụng các chính sách như giảm giá lần đầu, tích điểm đổi quà, hoặc miễn phí vận chuyển cho tài khoản đăng ký
- Đa dạng hóa thị trường ngoài UK:
  - ✓ Dữ liệu cho thấy hơn 90% doanh thu đến từ UK, vì vậy doanh nghiệp nên xem xét mở rộng sang các thị trường tiềm năng khác trong 38 quốc gia có giao dịch
  - ✓ Triển khai chiến dịch marketing địa phương hóa để thu hút khách hàng ở các quốc gia có tiềm năng tăng trưởng
- Tối ưu thời gian bán hàng và khung giờ vàng: Giao dịch tập trung từ 6h–20h, cao điểm lúc 15h, doanh nghiệp có thể:
  - ✓ Đẩy mạnh quảng cáo online và thông báo flash sale trong khung giờ này
  - ✓ Triển khai giảm giá giờ vàng để tối đa hóa doanh thu trong các khung giờ ít giao dịch hơn (ví dụ: trước 10h sáng hoặc sau 18h tối)
- Chiến lược giữ chân khách hàng trung thành:
  - ✓ Xây dựng hồ sơ khách hàng (Customer Profiles) để hiểu rõ hơn về nhóm khách hàng như ID 17841 – người có 7,676 lượt mua và mua 1,343 loại sản phẩm khác nhau
  - ✓ Triển khai chương trình thành viên VIP hoặc tặng quà sinh nhật, ưu đãi độc quyền để giữ chân nhóm khách hàng có giá trị cao này