Project Overview

Business Objective/Problem

- Công ty X chủ yếu bán các sản phẩm là quà tặng dành cho những dịp đặc biệt. Nhiều khách hàng của công ty là khách hàng bán buôn.
- Công ty X mong muốn có thể bán được nhiều sản phẩm hơn cũng như giới thiệu sản phẩm đến đúng đối tượng khách hàng, chăm sóc và làm hài lòng khách hàng.

EDA

```
df = pd.read_csv('OnlineRetail.csv', encoding= 'unicode_escape')
```

```
df.info()
```

```
<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
```

- Dữ liệu là file OnlineRetail.csv
- Gồm 541909 records là thông tin chi tiết về từng mặt hang trong từng đơn hàng
- Dữ liệu null ở Description và CustomerID
- InvoiceDate có kiểu dữ liệu không phải datetime

EDA

- Chuyển kiểu dữ liệu cột InvoiceDate từ object về datetime

```
string_to_date = lambda x : datetime.strptime(x, "%d-%m-%Y %H:%M").date()

#convert
df['InvoiceDate'] = df['InvoiceDate'].apply(string_to_date)
df['InvoiceDate'] = df['InvoiceDate'].astype('datetime64[ns]')
```

- Hiển thị 5 dòng đầu dữ liệu

```
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01	3.39	17850.0	United Kingdom

EDA

- Sử dụng Pandas Profiling, nhận xét:

Số lượng Khách hàng và Đơn hàng ít trên tổng mẫu dữ liệu. Nhiều KH mua nhiều mặt hàng với số lượng lớn trong 1 đơn hàng => KH của DN chủ yếu là KH bán buôn

- Loại bỏ dữ liệu trùng:

```
# loại bỏ DL trùng:
print('Trước khi drop, records = ', df.shape[0])
df.drop_duplicates(inplace = True)
print('Sau khi drop, records = ', df.shape[0])
```

```
Trước khi drop, records =  541909
Sau khi drop, records =  536640
```

EDA

- Dữ liệu có nhiều giá trị null tại CustomerID và Description. Xử lý: Xóa bỏ null tại cột CustomerID do đây là bài toán phân cụm KH, CustomerID không được phép null; Cột Description không ảnh hưởng tới bài toán phân cụm nên xem xét loại bỏ

```
# kiểm tra dữ liệu null:
df.isnull().sum()
```

```
InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID    135037
Country            0
dtype: int64
```

```
m = []
for c in ['Description', 'CustomerID']:
  m.append({
      'feature': c,
      'Missing_Value': df[df[c].isnull()].shape[0],
      'Percentage': df[df[c].isnull()].shape[0]/df.shape[0]*100
  })
df_null = pd.DataFrame(m)
df_null
```

	feature	Missing_Value	Percentage
0	Description	1454	0.270945
1	CustomerID	135037	25.163424

EDA

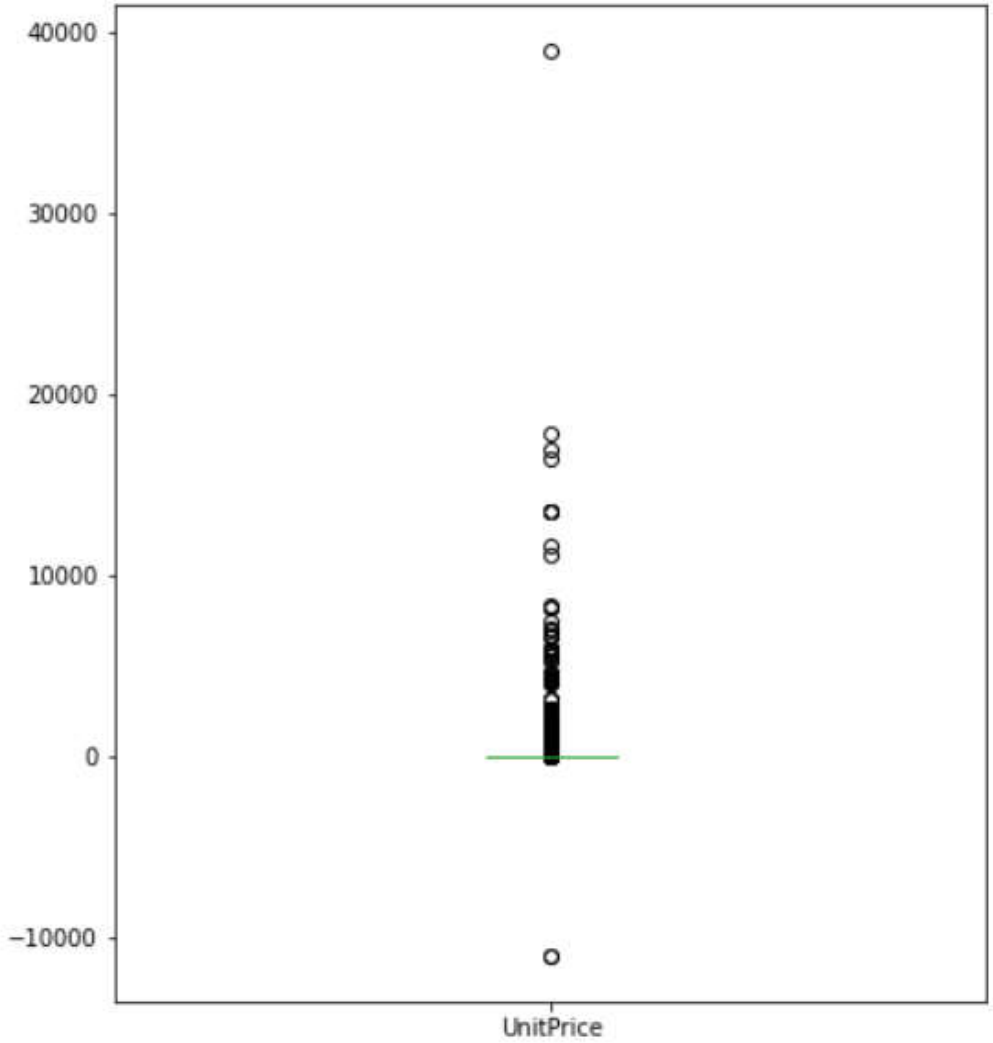
- Dữ liệu có giá trị âm ở Quantity hoặc UnitPrice liên quan tới đơn hàng trả lại/hủy đơn đúng với thực tế nên vẫn giữ lại.
- Trung bình số lượng mỗi sản phẩm trong mỗi đơn hàng là 9,62; max là 80.995; range rộng
- Trung bình giá bán mỗi sản phẩm là 4,63; max là 38.970; range rộng

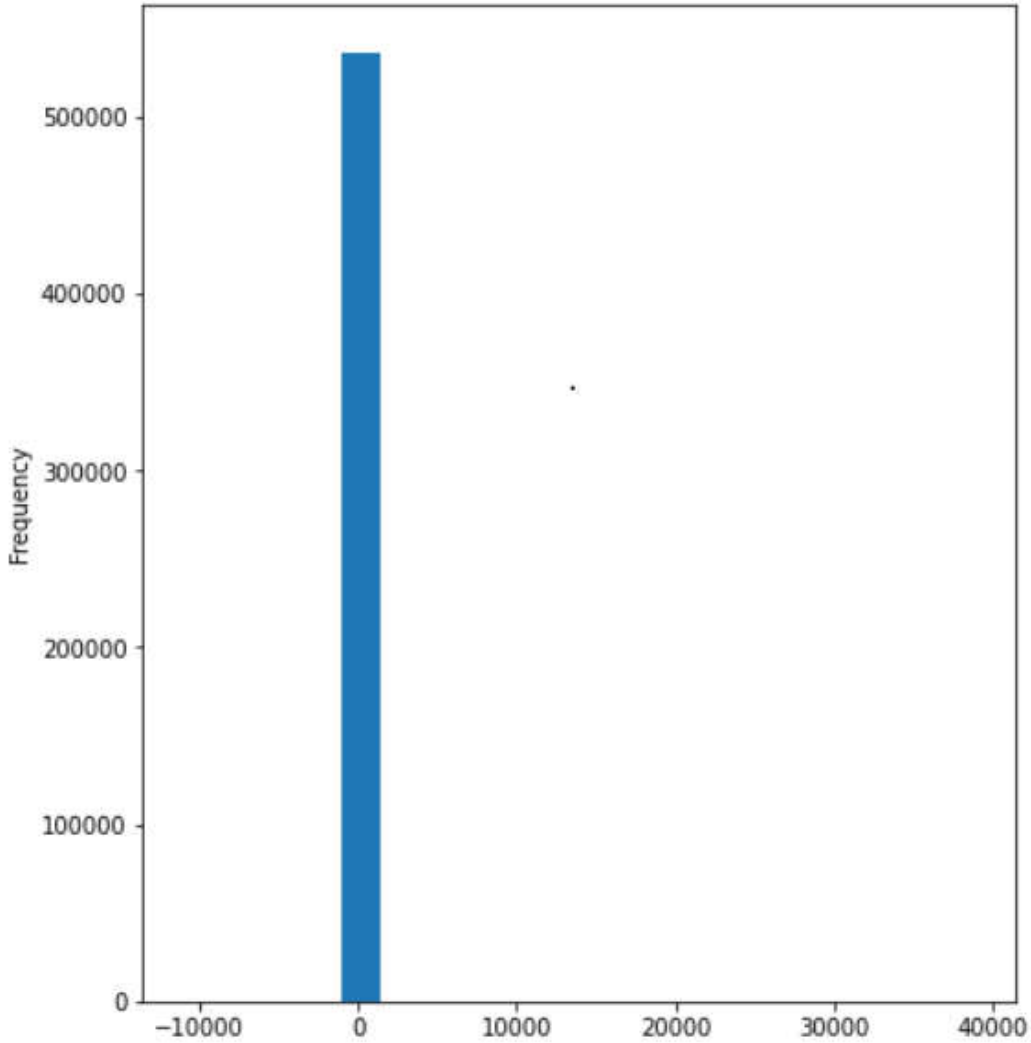
```
pd.options.display.float_format = '{:,.2f}'.format
df[['Quantity', 'UnitPrice']].describe().T
```

	count	mean	std	min	25%	50%	75%	max
Quantity	536,640.00	9.62	219.13	-80,995.00	1.00	3.00	10.00	80,995.00
UnitPrice	536,640.00	4.63	97.23	-11,062.06	1.25	2.08	4.13	38,970.00

EDA

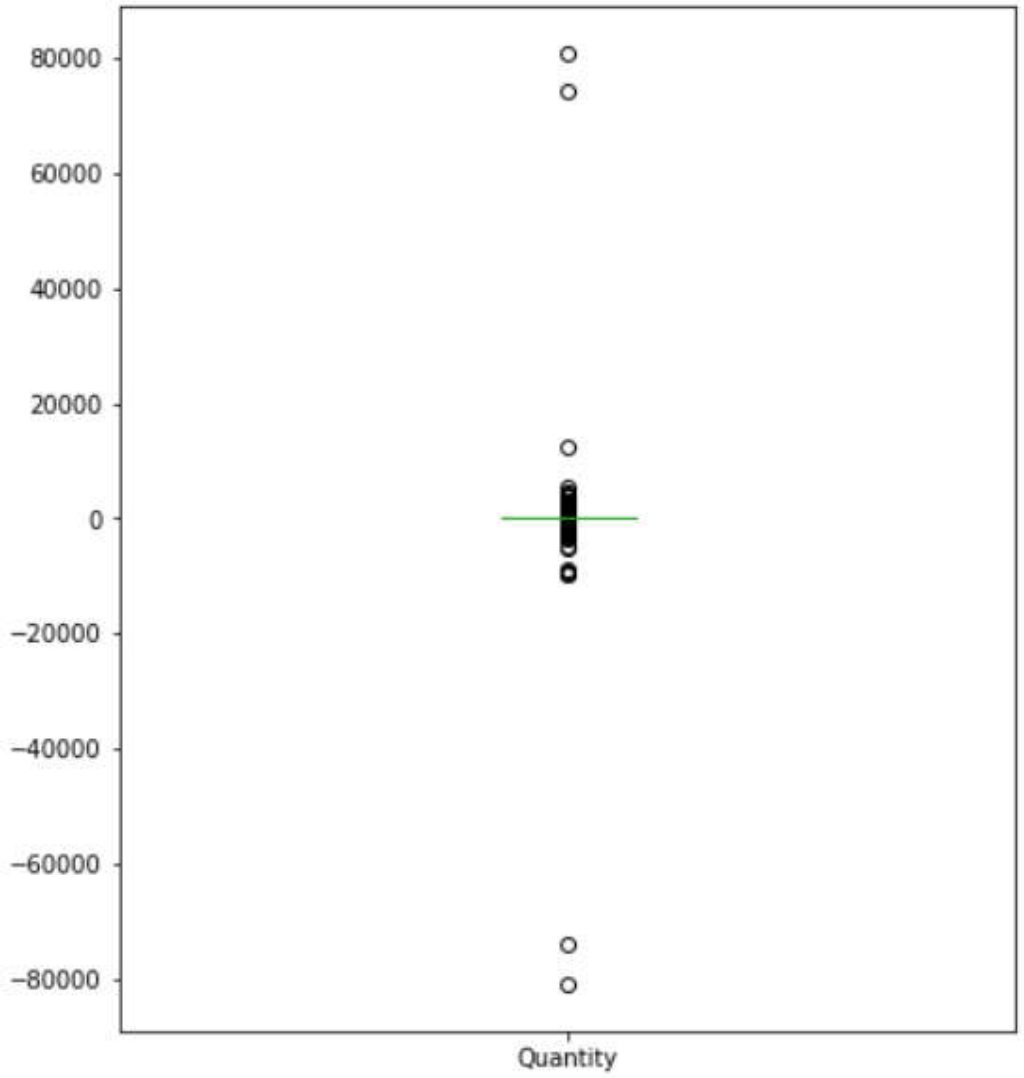
- Phần lớn sản phẩm có giá UnitPrice < 10.000

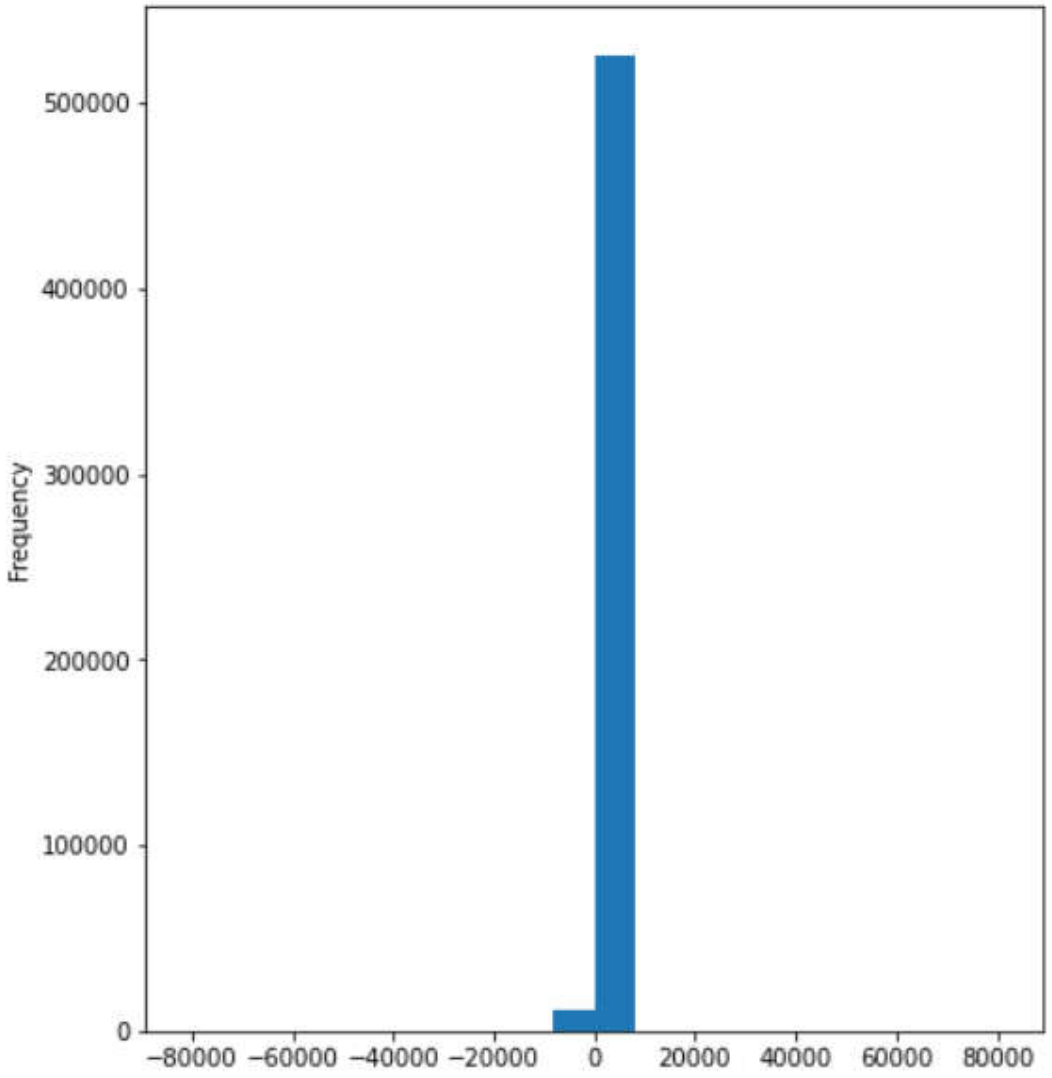




EDA

- Hầu hết Quantity có giá trị < 10.000



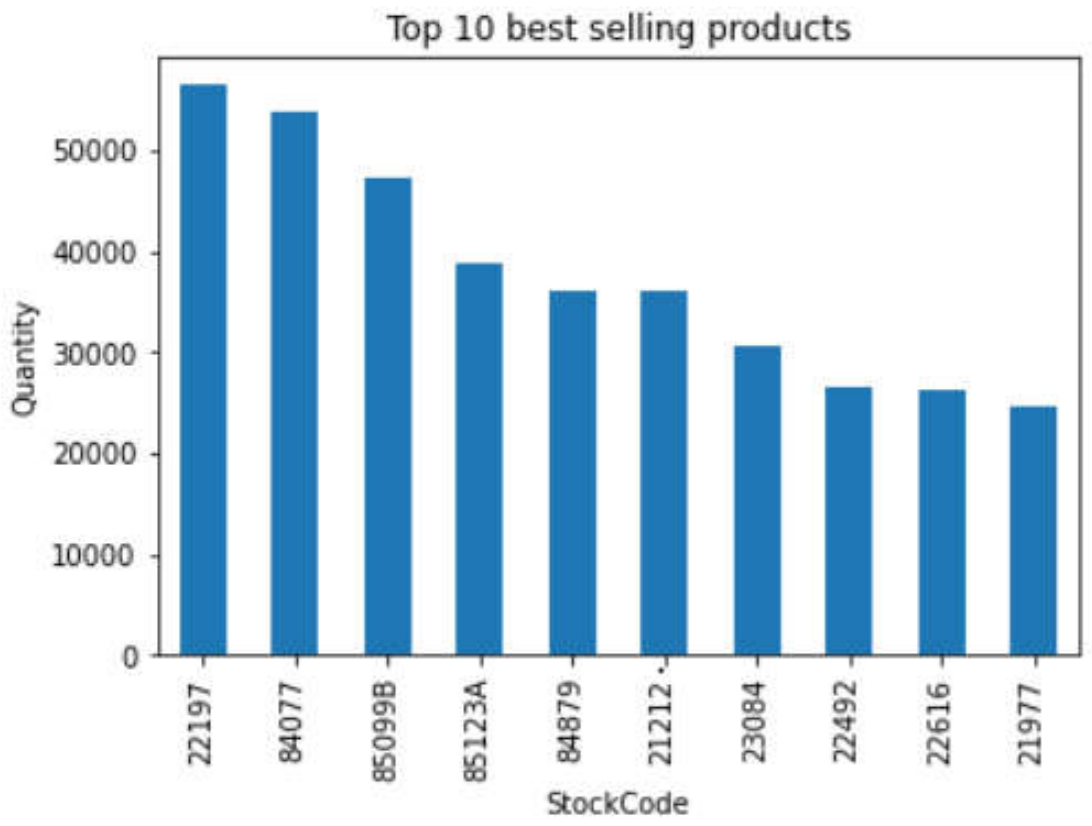


EDA

- Sản phẩm có StockCode là 22197 được mua nhiều nhất (56.427)
- Sản phẩm có StockCode là 23005 bị trả nhiều nhất (14.468)
- Top 10 sản phẩm bán chạy nhất

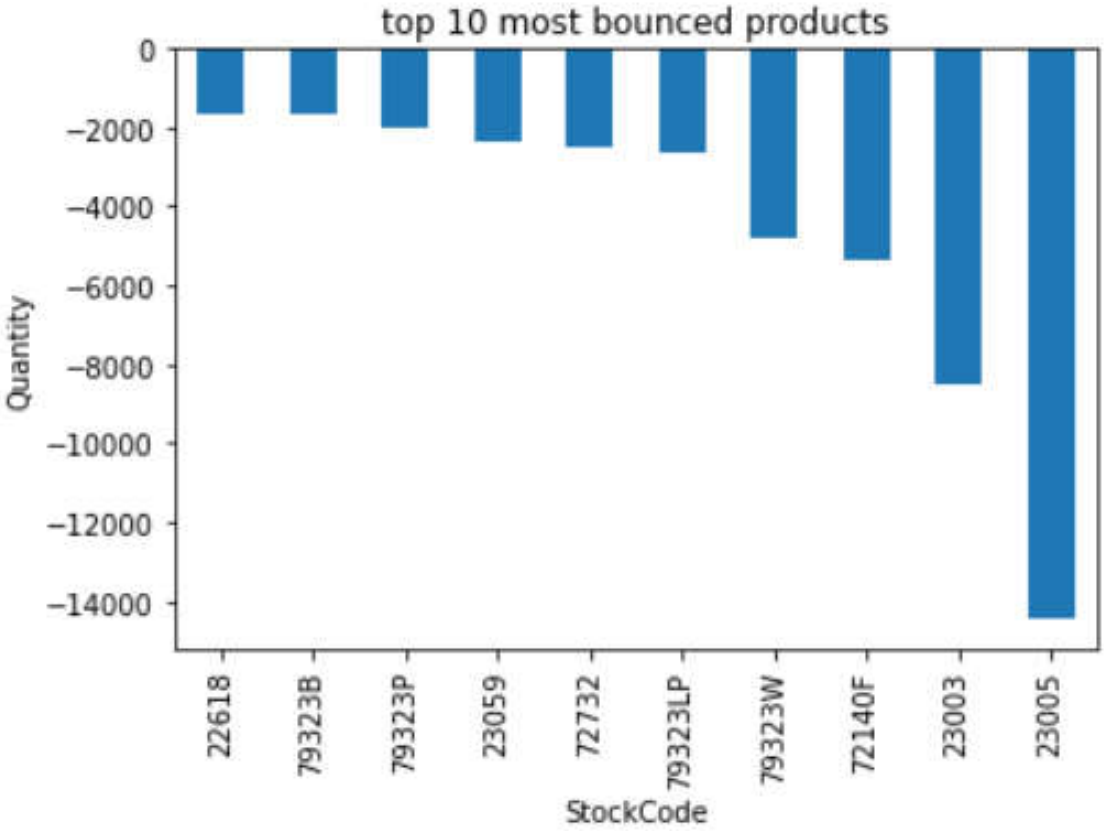
```
StockCode = df.groupby('StockCode')['Quantity'].sum().sort_values(ascending = False)
StockCode
```

```
StockCode
22197      56427
84077      53751
85099B     47260
85123A     38811
84879      36122
           ...
79323LP    -2618
79323W     -4838
72140F     -5368
23003      -8516
23005     -14468
```



EDA

- Sản phẩm bị trả hàng nhiều nhất là '23005' với 14.468 sp
- 6 sp bị trả hàng khoảng ~2.000 sp
- 3 sp còn lại trong top 10 bị trả hàng trong khoảng 5.000-8.000 sp



EDA

- Sản phẩm có Description là 'WORLD WAR 2 GLIDERS ASSTD DESIGNS' được mua nhiều nhất (53.751 sản phẩm)
- Sản phẩm có Description là 'printing smudges/thrown away' bị trả nhiều nhất (19.200 sản phẩm)
- Số liệu này không khớp với StockCode ở trên do:

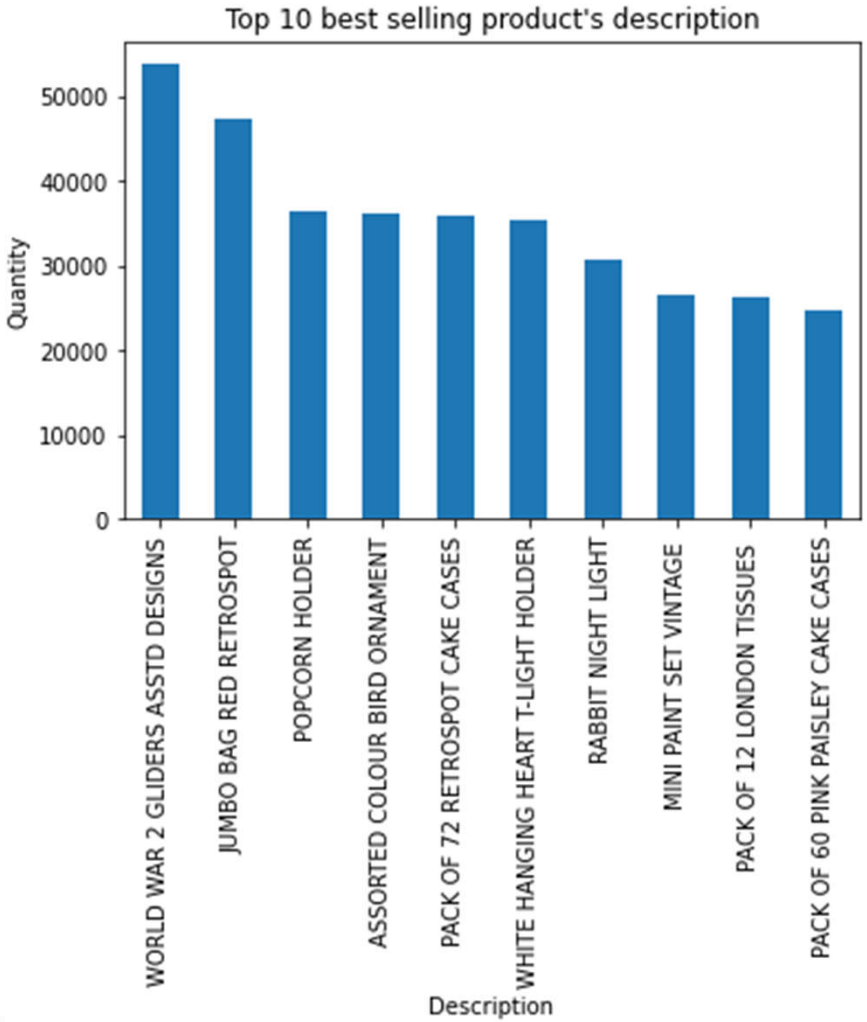
  - Description bị null ở một số records

  - StockCode là duy nhất cho mỗi sản phẩm (hiểu là mã sản phẩm) trong khi Description có thể giống nhau với các sản phẩm có cùng đặc tính.

```
Description
WORLD WAR 2 GLIDERS ASSTD DESIGNS    53751    Damaged                                 -7540
JUMBO BAG RED RETROSPOT              47260    Printing smudges/thrown away            -9058
POPCORN HOLDER                       36322    check                                  -12030
ASSORTED COLOUR BIRD ORNAMENT        36282    Unsaleable, destroyed.                 -15644
PACK OF 72 RETROSPOT CAKE CASES      36016    printing smudges/thrown away           -19200
                                     ...      Name: Quantity, Length: 4223, dtype: int64
```

EDA

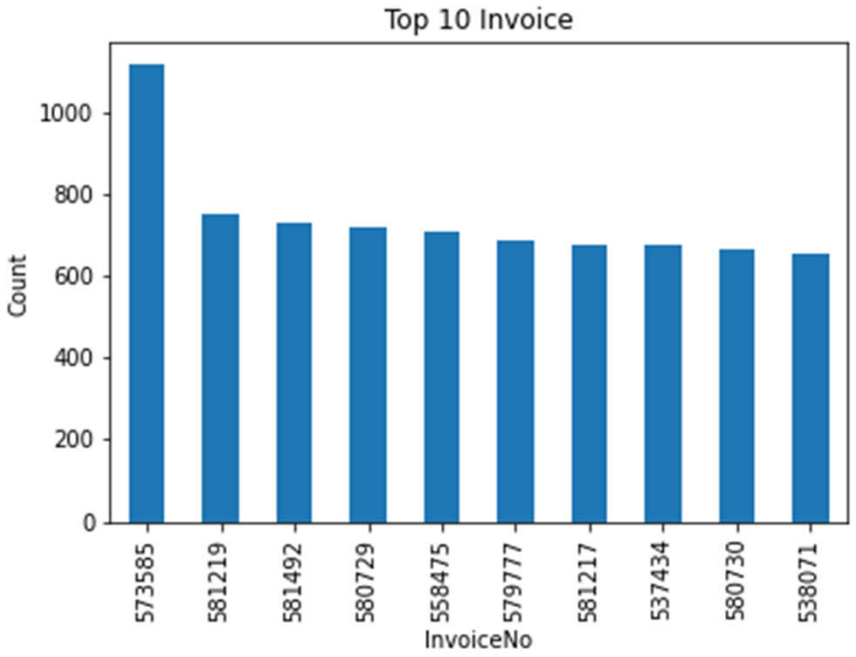
- Top 2 được bán từ 48.000-54.000 sp
- số còn lại trong top 10 được bán từ 28.000-37.000 sp



EDA

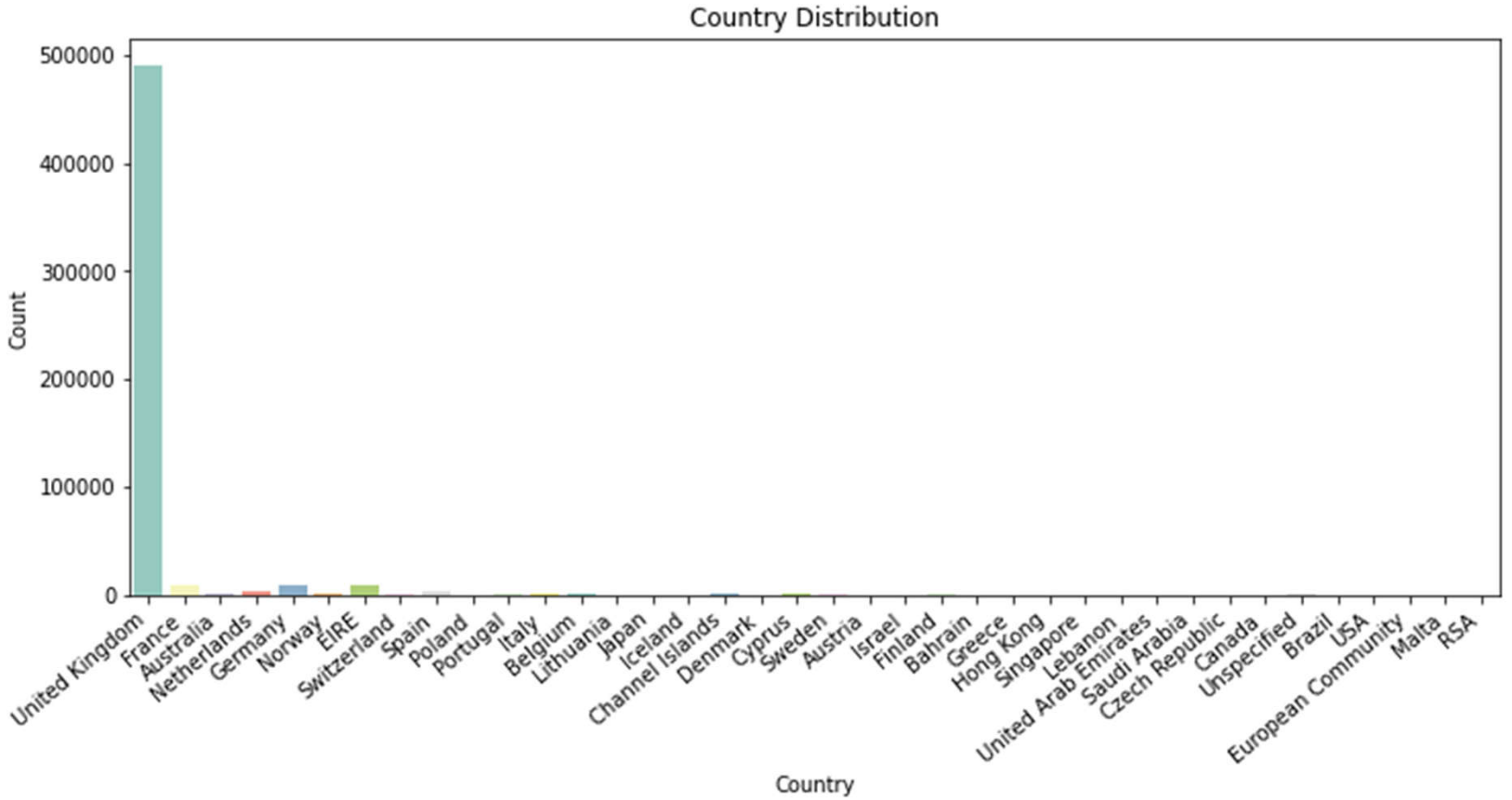
- Đơn hàng có mã 573585 mua nhiều mặt hàng nhất với 1.114 mặt hàng
- Một số đơn hàng chỉ mua 1 mặt hàng.

```
InvoiceNo
573585    1114
581219     749
581492     731
580729     721
558475     705
          ...
540272       1
540279       1
557501       1
573926       1
538092       1
```



EDA

- KH chủ yếu đến từ United Kingdom (91,36%)

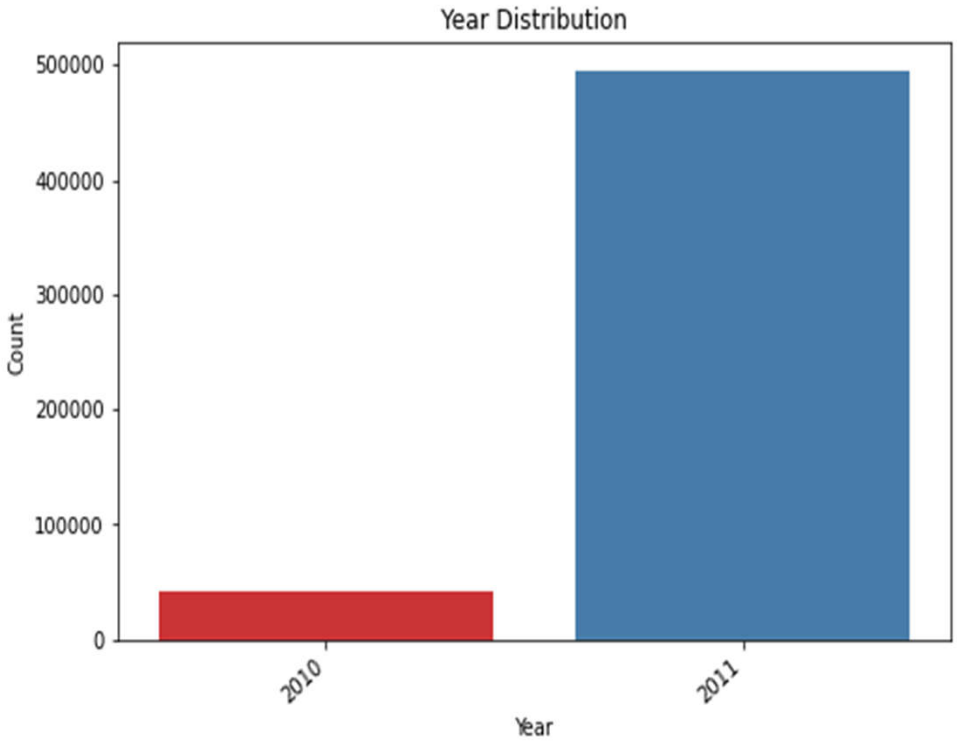


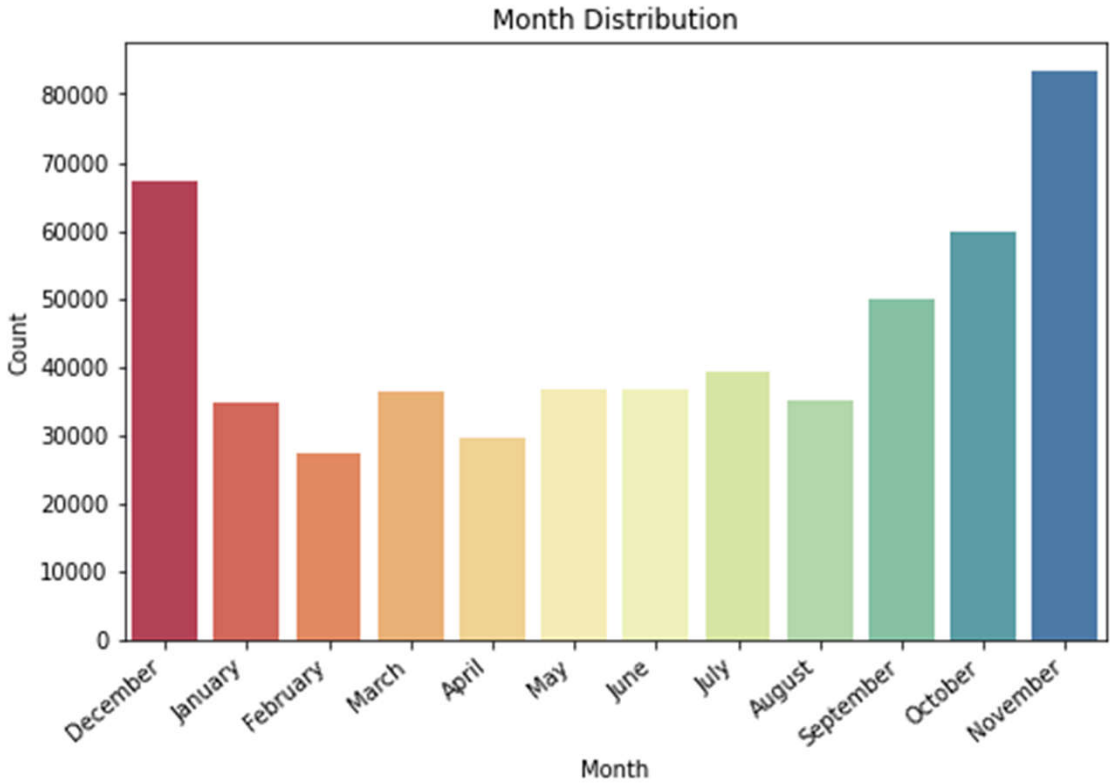

```
_UK = df[df['Country']=='United Kingdom'].shape[0]/df.shape[0]*100
_UK
```

```
91.36460196779964
```

EDA

- Dữ liệu thu thập về đơn hàng chủ yếu của năm 2011 (92.177%)
- Những tháng cuối năm (9-12) bán nhiều hàng hơn đầu năm.





Xử lý dữ liệu

- Chỉ sử dụng các cột InvoiceNo, Quantity, UnitPrice, InvoiceDate và CustomerID để phân cụm KH.

```
df = df[['InvoiceNo', 'Quantity', 'InvoiceDate', 'UnitPrice', 'CustomerID']]
df = df.dropna()
```

```
# Let's take a closer look at the data we will need to manipulate.
print('Transactions timeframe from {} to {}'.format(df['InvoiceDate'].min(), df['InvoiceDate'].max()))
print('{:,} transactions don\'t have a customer id'.format(df[df.CustomerID.isnull()].shape[0]))
print('{:,} unique CustomerID'.format(len(df.CustomerID.unique())))
```

```
Transactions timeframe from 2010-12-01 00:00:00 to 2011-12-09 00:00:00
0 transactions don't have a customer id
4,372 unique CustomerID
```

- Tạo cột total_sales bằng Quantity * UnitPrice

```
df['total_sales'] = df['Quantity']*df['UnitPrice']
```

Xử lý dữ liệu

- Tạo dataframe df_RFM với index là CustomerID, cột Recency = ngày tham chiếu (ngày lớn nhất trong bảng dữ liệu) – ngày KH đó mua đơn hàng cuối cùng), Frequency là số đơn hàng, Monetary là tổng số tiền mua hàng

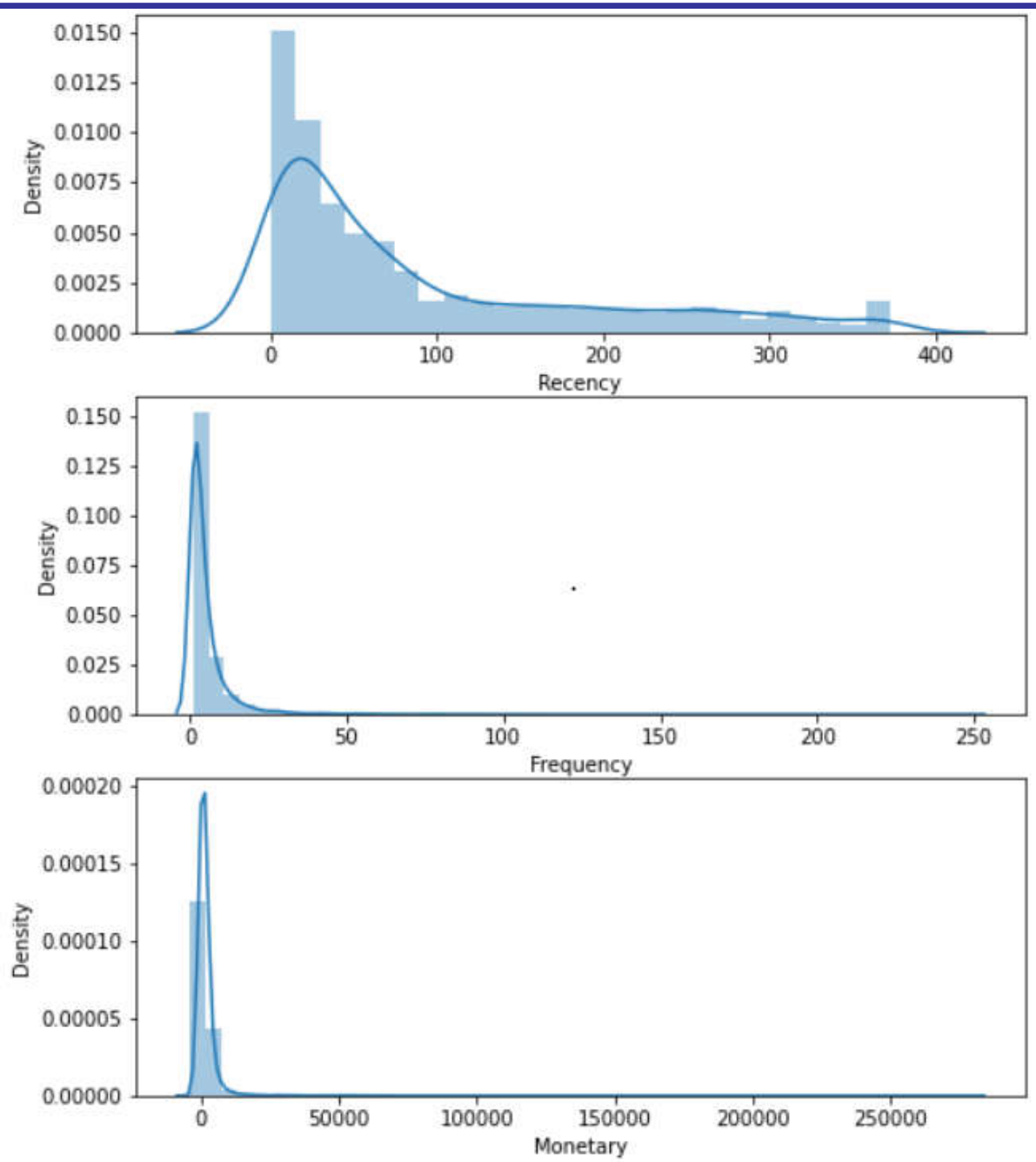
```
df_RFM.head()
```

CustomerID	Recency	Frequency	Monetary
14646.0	1	77	279489.02
18102.0	0	62	256438.49
17450.0	8	55	187322.17
14911.0	1	248	132458.73
12415.0	24	26	123725.45

```
df_RFM.shape
```

```
(4372, 3)
```

Xử lý dữ liệu



- Recency tập trung từ 0-100 ngày, 100-400 có tỷ lệ ít hơn
- Frequency tập trung từ 0-20 đơn hàng, từ 20-50 đơn tỷ lệ giảm dần và từ 50-250 đơn có nhưng rất ít
- Monetary tập trung từ 0-10000, từ 10000-280000 có nhưng rất ít
- phân phối lệch phải

Xử lý dữ liệu

- Chỉ số Hopkins = 0.0058 < 0.5 cho thấy dữ liệu có khả năng phân cụm lớn
- Tạo cột R, F, M là thứ hạng (chia 4 mức) của Recency (ngày càng gần thì R càng lớn), Frequency, Monetary (càng lớn thì F, R càng lớn), RFM_Segment là chuỗi ghép từ R, F, M và RFM_Score là tổng của R, F, M
- Dữ liệu có 63 nhóm theo RFM_Segment

CustomerID	Recency	Frequency	Monetary	R	F	M	RFM_Segment	RFM_Score
14646.0	1	77	279489.02	4	4	4	444	12
18102.0	0	62	256438.49	4	4	4	444	12
17450.0	8	55	187322.17	4	4	4	444	12
14911.0	1	248	132458.73	4	4	4	444	12
12415.0	24	26	123725.45	3	4	4	344	11

RFM truyền thống

- Sử dụng RFM truyền thống, phân cụm KH theo tiêu chí (theo function rfm_level), thu được df rfm_agg gồm 8 nhóm KH với giá trị trung bình của Recency, Frequency, Monetary và số lượng, tỷ lệ KH mỗi nhóm như sau:

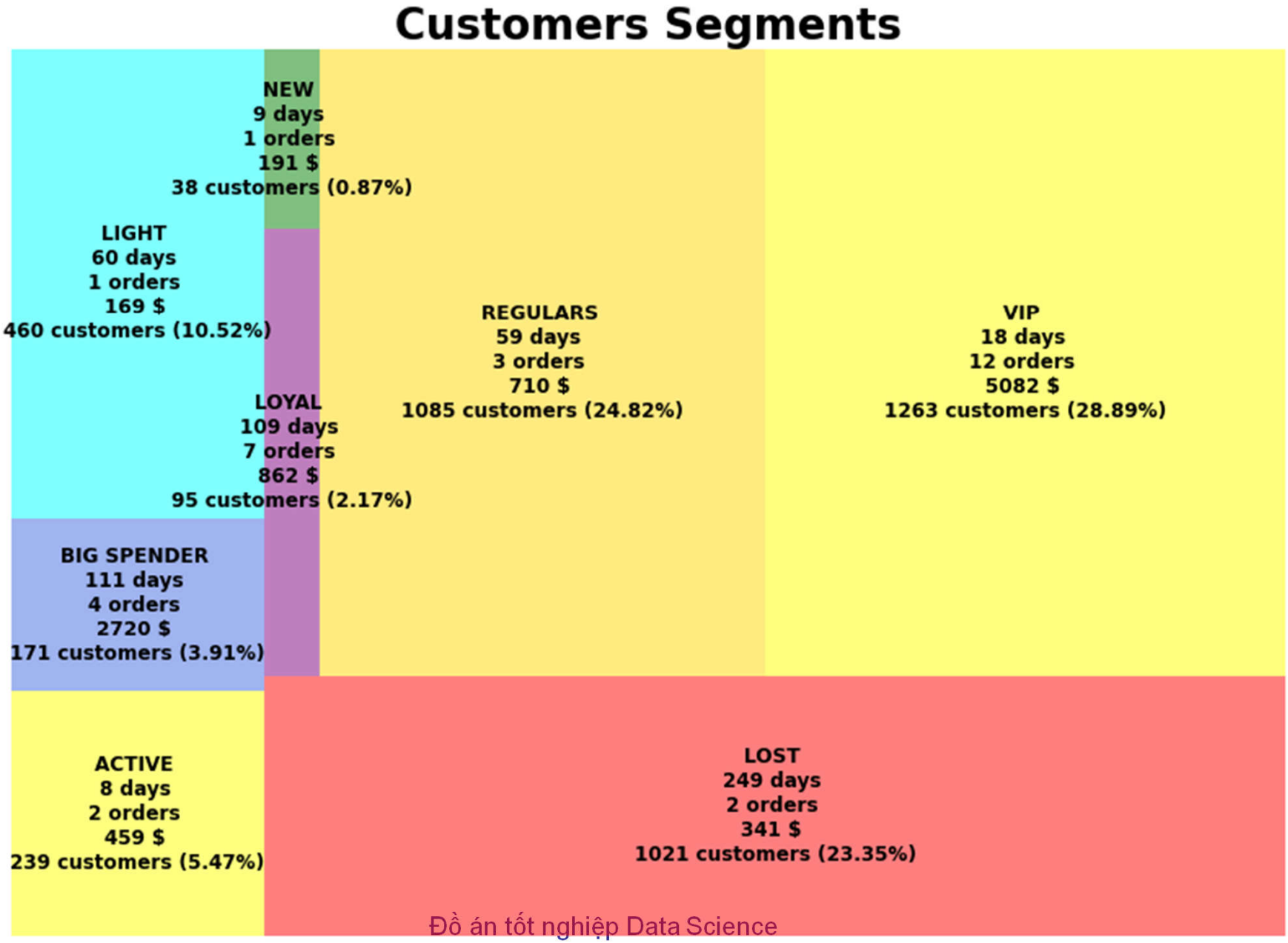
```
def rfm_level(df):
    if (df['RFM_Score'] >= 10)  :
        return 'VIP'

    elif (df['R'] == 4 and df['F'] ==1 and df['M'] == 1):
        return 'NEW'

    else:
        if df['M'] == 4:
            return 'BIG SPENDER'

        elif df['F'] == 4:
            return 'LOYAL'

        elif df['R'] == 4:
            return 'ACTIVE'

        elif df['R'] == 1:
            return 'LOST'

        elif df['M'] == 1:
            return 'LIGHT'

        return 'REGULARS'
```

```
rfm_agg
```

	RFM_Level	RecencyMean	FrequencyMean	MonetaryMean	Count	Percent
0	ACTIVE	8.0	2.0	459.0	239	5.47
1	BIG SPENDER	111.0	4.0	2720.0	171	3.91
2	LIGHT	60.0	1.0	169.0	460	10.52
3	LOST	249.0	2.0	341.0	1021	23.35
4	LOYAL	109.0	7.0	862.0	95	2.17
5	NEW	9.0	1.0	191.0	38	0.87
6	REGULARS	59.0	3.0	710.0	1085	24.82
7	VIP	18.0	12.0	5082.0	1263	28.89

RFM truyền thống

- Trực quan:

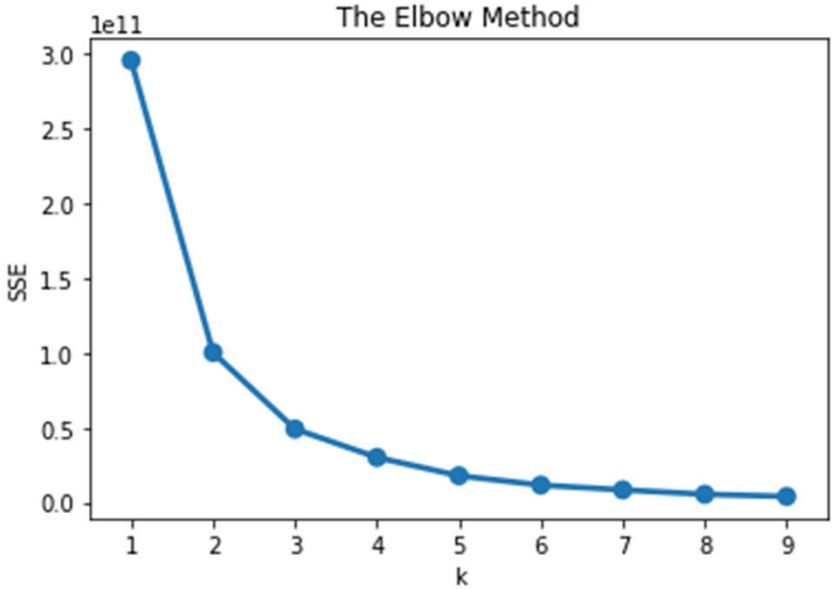


RFM truyền thống

- Số lượng khách hàng có nguy cơ ra đi (LOST) chiếm tỷ lệ rất lớn (23.35%) -> cần có giải pháp giữ chân khách hàng vì số lượng quá lớn sẽ ảnh hưởng nghiêm trọng đến doanh thu của doanh nghiệp
- Số lượng khách hàng VIP (mua hàng gần đây, số lượng đơn hàng nhiều, số tiền mua hàng lớn) chiếm 28.89%
- Số lượng khách hàng BIG SPENDER (chi nhiều tiền mua hàng) chiếm 3.91%
- Số lượng khách hàng LIGHT (chi ít tiền mua hàng) chiếm 10.52%
- Số lượng khách hàng ACTIVE (còn mua hàng gần đây) chiếm 5.47%
- Số lượng khách hàng LOYAL (số lượng đơn hàng nhiều) chiếm 2.17%
- Số lượng khách hàng NEW (mua hàng gần đây, số lượng đơn hàng ít, số tiền ít) chiếm 0.87%
- Số lượng khách hàng REGULARS (thông thường) chiếm 24.82%

RFM kết hợp KMeans

- Sử dụng các thuộc tính Recency, Frequency, Monetary, áp dụng Kmeans để phân cụm KH
- Sử dụng Elbow Method để lựa chọn số cụm k
- K có thể nhận giá trị trong khoảng 3-6. Có 2 điểm gãy ở k=3 và k=5 tuy nhiên nếu chia KH làm 3 nhóm thì số lượng nhóm ít và dễ dẫn tới những chiến lược không phù hợp cho việc chăm sóc KH.
- Đầu tiên em sẽ phân cụm với k=5



RFM kết hợp KMeans

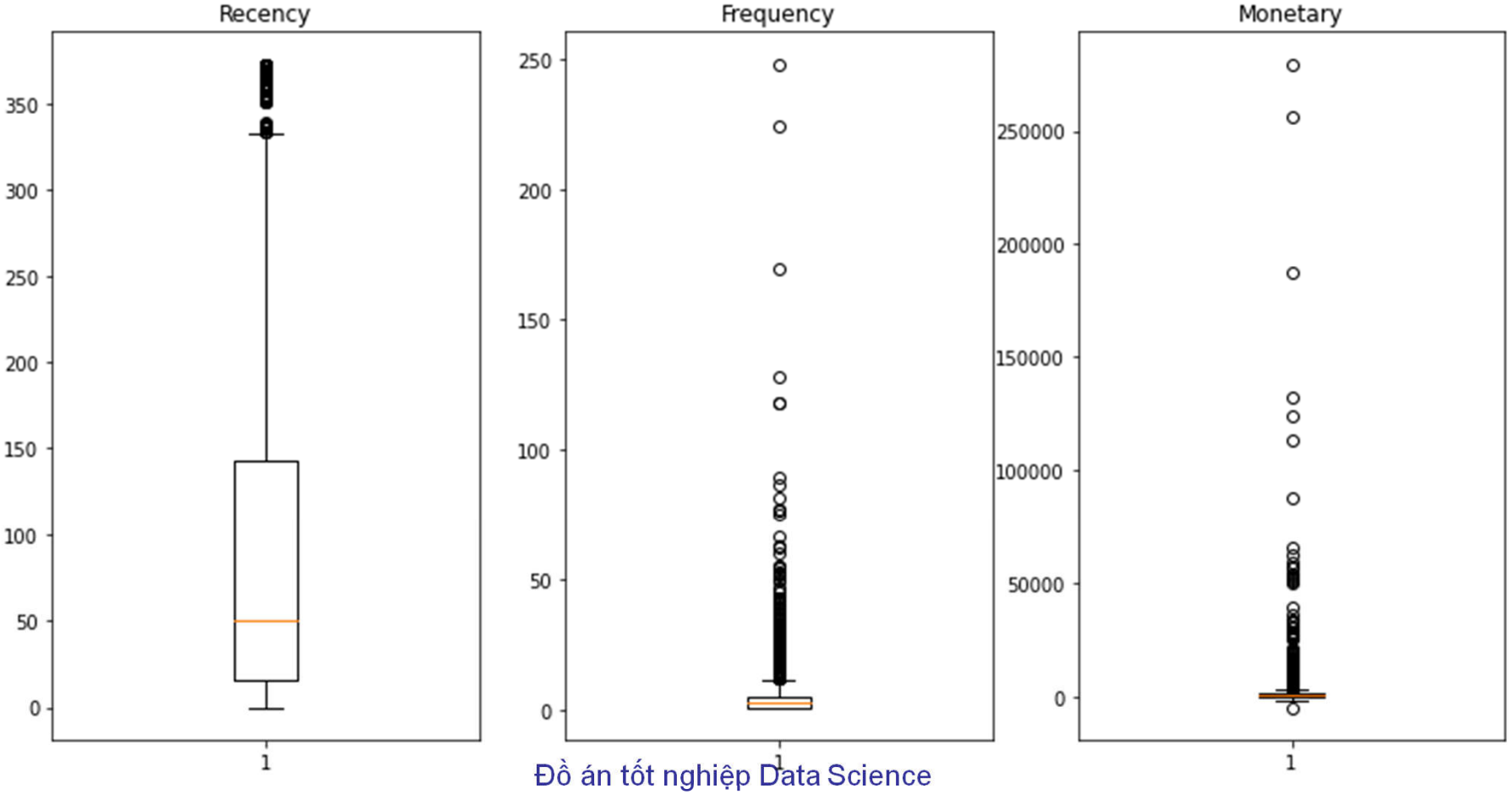
- Với k = 5, KH được phân cụm như sau:

	Cluster	RecencyMean	FrequencyMean	MonetaryMean	Count	Percent
0	Cluster 0	20.0	20.0	8073.0	267	6.11
1	Cluster 1	9.0	88.0	128969.0	5	0.11
2	Cluster 2	0.0	70.0	267964.0	2	0.05
3	Cluster 3	6.0	58.0	40961.0	27	0.62
4	Cluster 4	97.0	4.0	942.0	4071	93.12

- Với k = 5, dữ liệu tập trung tới 93.12% vào 1 nhóm, trong khi 4 nhóm còn lại chỉ chiếm tỷ lệ rất nhỏ.
- Nhận thấy kết quả không phù hợp nên em sẽ phân cụm với k khác để tìm k phù hợp hơn.
- Với k từ 3 đến 6 thì khoảng 90-99.4% dữ liệu tập trung ở 1 phân cụm khách hàng. Sử dụng RFM kết hợp với KMeans trong bài toán này cho kết quả không tốt so với chỉ sử dụng RFM thuần túy.
- Cần phải scale dữ liệu trước khi áp dụng Kmeans

Trực quan

- Các biến đều có số lượng outliers lớn, ở nhóm monetary là đặc biệt lớn. Tuy nhiên, chính những KH có số tiền mua lớn là những KH quan trọng nhất, mang lại doanh thu cao cho doanh nghiệp nên được giữ lại => chỉ loại outliers của recency (KH đã mua từ quá lâu) và frequency (KH mua quá nhiều lần).



Scale dữ liệu

- Sau khi loại bỏ outliers ở Recency, Frequency và scale, dữ liệu còn lại 3.921 dòng, với giá trị đã scale như sau:

	Recency	Frequency	Monetary
0	1.633793	0.177912	14.389649
1	-0.942958	2.630524	14.029790
2	-0.509797	3.331271	11.614459
3	-0.865211	3.331271	9.230721
4	-0.976278	1.229032	8.653150

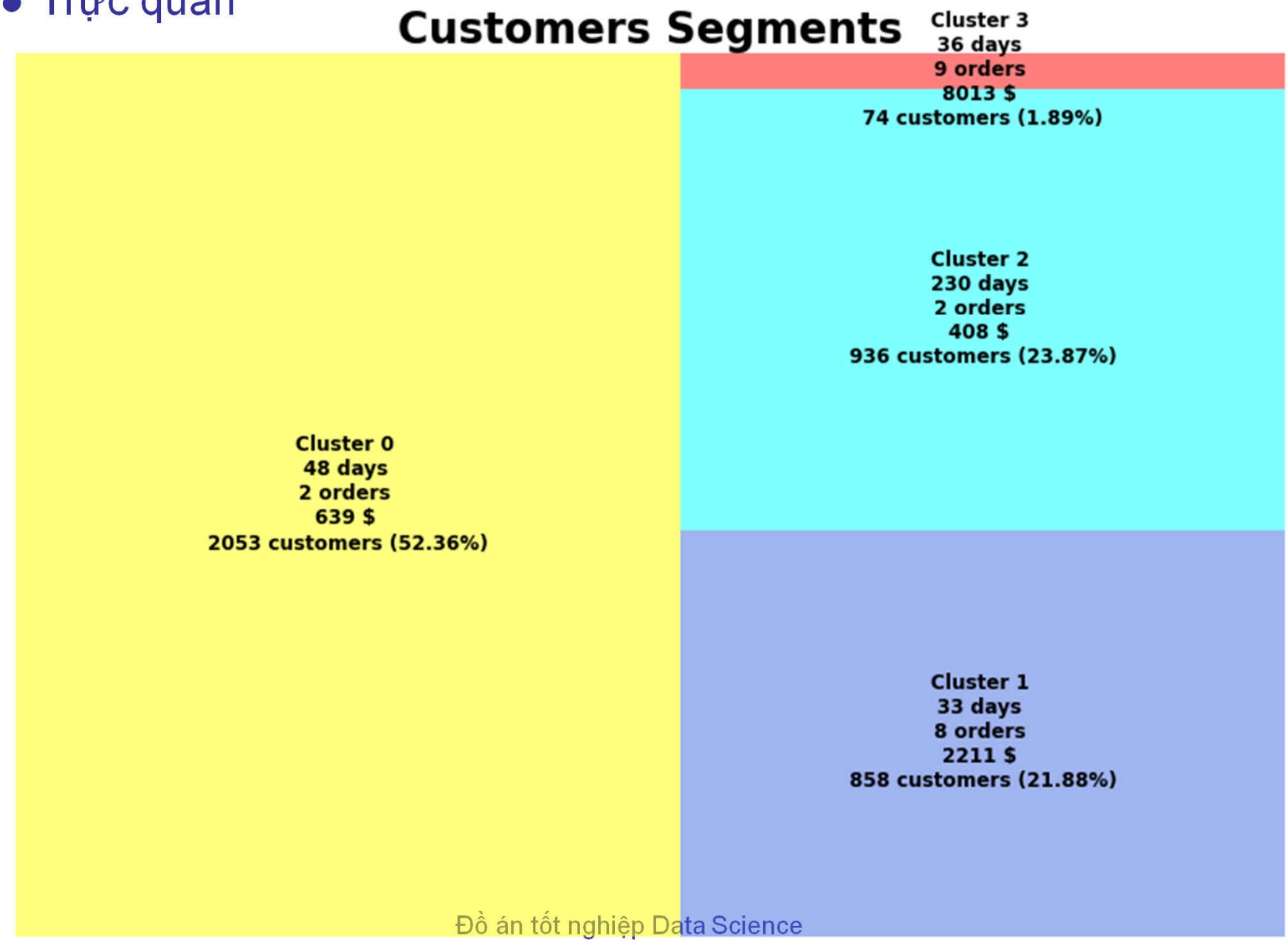
Dữ liệu đã scaled + KMeans

- Áp dụng Kmeans với dữ liệu đã scaled
- Sử dụng elbow method để chọn k
- Có các điểm gãy ở k=3 và k=4 tuy nhiên em sẽ chọn k=4 (phân KH làm 3 nhóm thì ít quá, dễ dẫn đến CSKH không phù hợp)
- KH được phân cụm như sau:

	Cluster	RecencyMean	FrequencyMean	MonetaryMean	Count	Percent
0	Cluster 0	48.0	2.0	639.0	2053	52.36
1	Cluster 1	33.0	8.0	2211.0	858	21.88
2	Cluster 2	230.0	2.0	408.0	936	23.87
3	Cluster 3	36.0	9.0	8013.0	74	1.89

Dữ liệu đã scaled + KMeans

- Trực quan



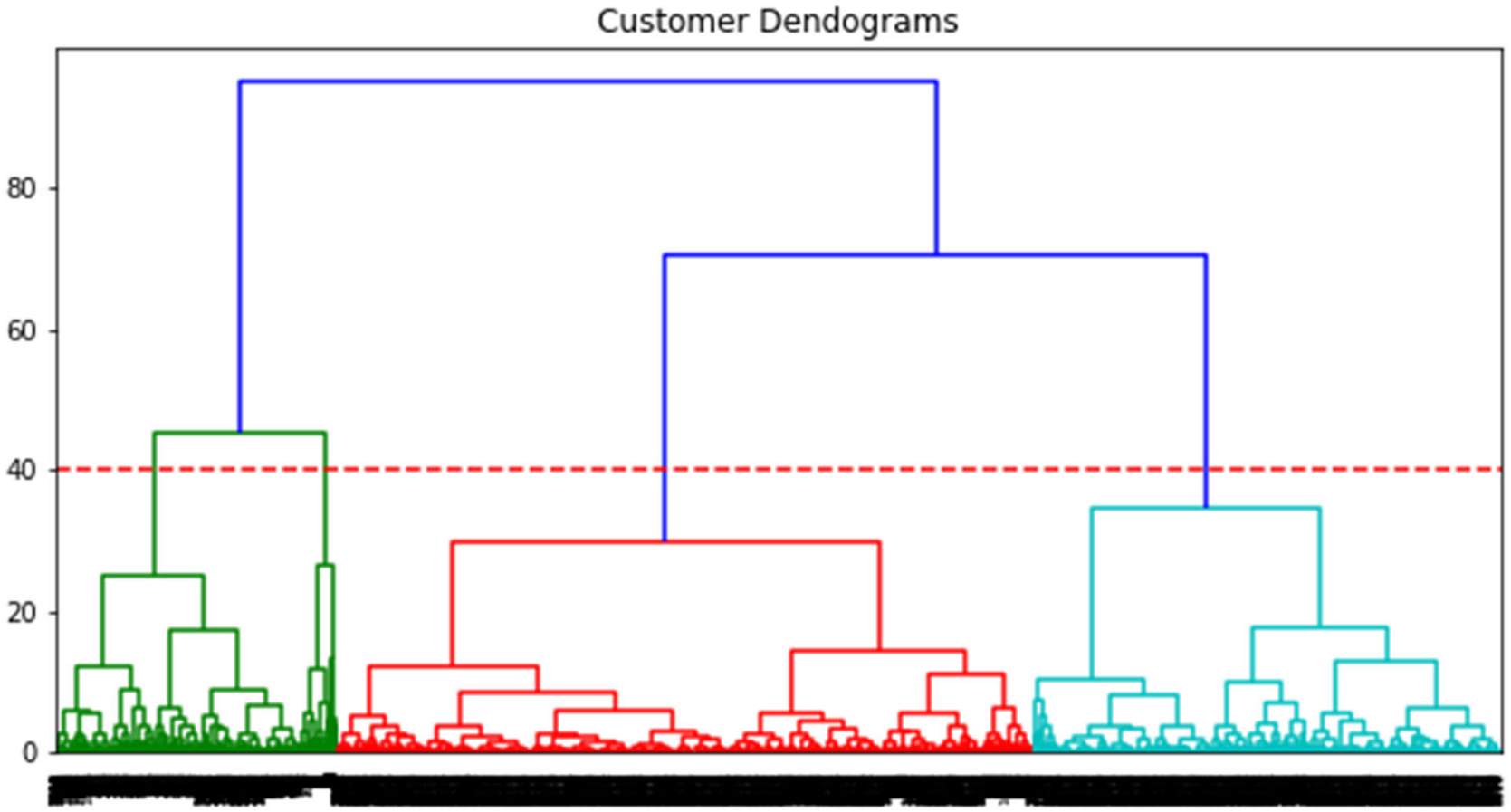
Dữ liệu đã scaled + KMeans

Chia KH thành 4 nhóm:

- Cluster 0: thời gian mua hàng trung bình, số đơn hàng trung ít, số tiền trung bình chiếm 52.36%
- Cluster 1: mua hàng gần đây, số đơn hàng lớn, số tiền lớn (KH VIP) chiếm 21.88%
- Cluster 2: thời gian mua hàng xa, số đơn hàng ít, số tiền ít (KH có nguy cơ ra đi) chiếm 23.87%
- Cluster 3: thời gian mua hàng gần đây, số đơn hàng lớn, số tiền rất lớn (KH VIP) chiếm 1.89%

Dữ liệu đã scaled + Hierarchy

- Áp dụng Hierarchy với dữ liệu đã scaled
- Sử dụng dendogram để lựa chọn số cụm là 4



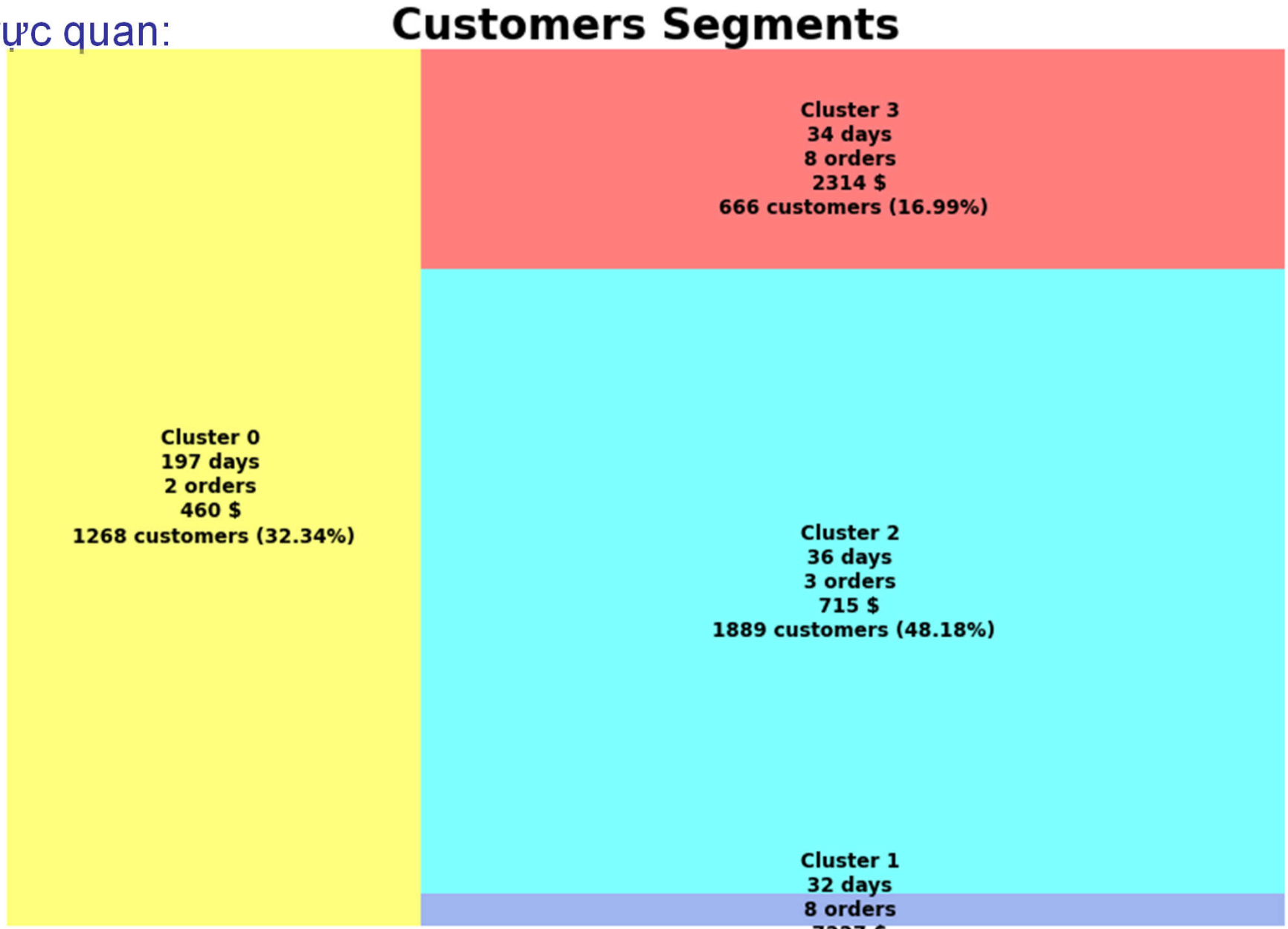
Dữ liệu đã scaled + Hierarchy

- KH được phân cụm như sau:

	Cluster	RecencyMean	FrequencyMean	MonetaryMean	Count	Percent
0	Cluster 0	197.0	2.0	460.0	1268	32.34
1	Cluster 1	32.0	8.0	7227.0	98	2.50
2	Cluster 2	36.0	3.0	715.0	1889	48.18
3	Cluster 3	34.0	8.0	2314.0	666	16.99

Dữ liệu đã scaled + Hierarchy

- Trực quan:



Dữ liệu đã scaled + Hierarchy

Khách hàng chia làm 4 nhóm:

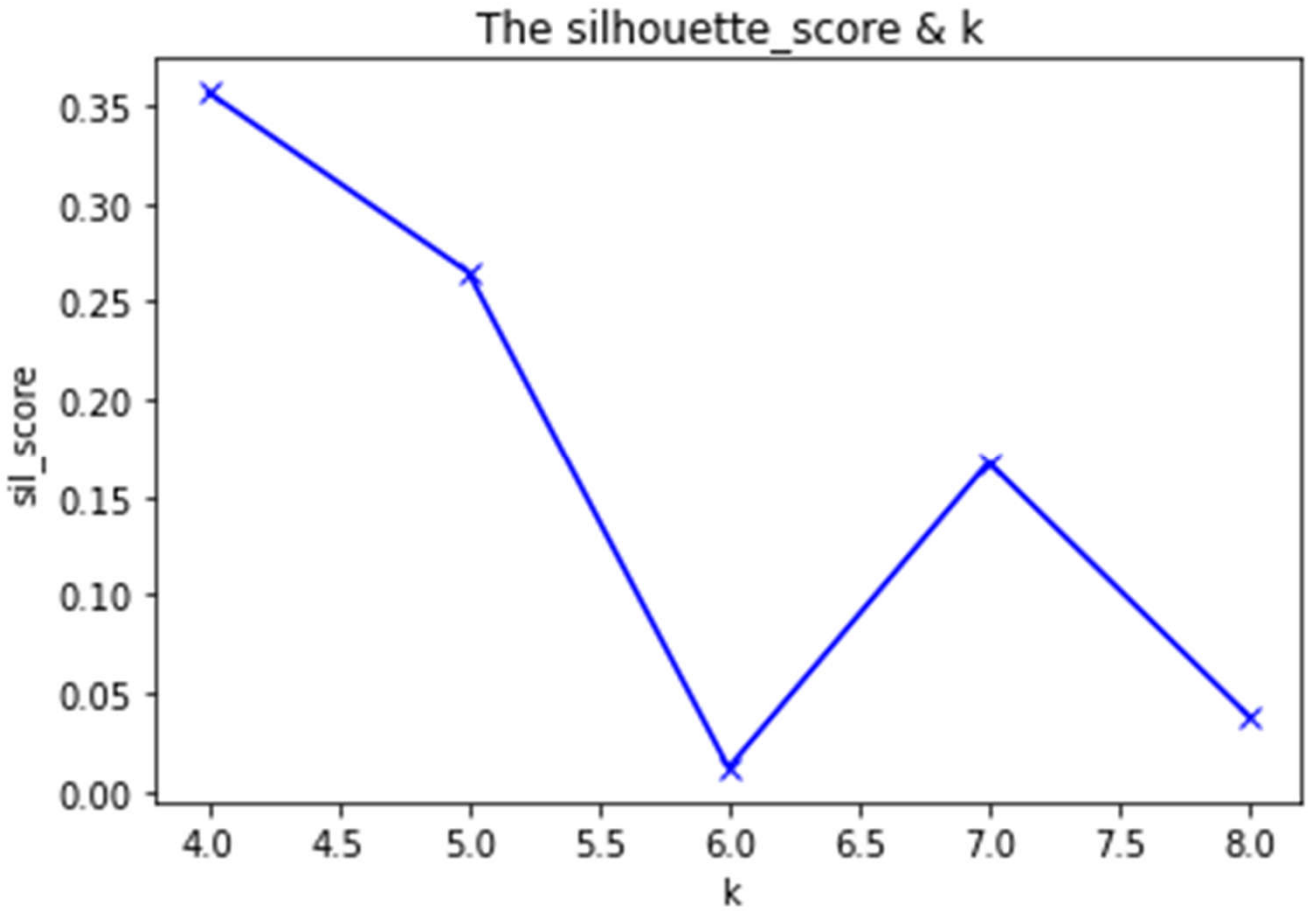
- Cluster 0: Khách hàng có thời gian mua hàng xa, số đơn hàng ít, số tiền ít (KH có nguy cơ ra đi) chiếm 32.34%
- Cluster 1: Thời gian mua hàng gần đây, số đơn hàng lớn, số tiền rất lớn (KH VIP) chiếm 2.5%
- Cluster 2: Thời gian mua hàng trung bình, số đơn hàng trung bình, số tiền trung bình (KH thông thường) chiếm 48.18%
- Cluster 3: Thời gian mua hàng gần đây, số đơn hàng lớn, số tiền lớn chiếm 16.99%

Dữ liệu đã scaled + GMM

- Áp dụng GMM với dữ liệu đã scaled
- Sử dụng silhouette score để lựa chọn số cụm là 4

Dữ liệu đã scaled + GMM

- Áp dụng GMM với dữ liệu đã scaled
- Sử dụng silhouette score để lựa chọn số cụm là 4



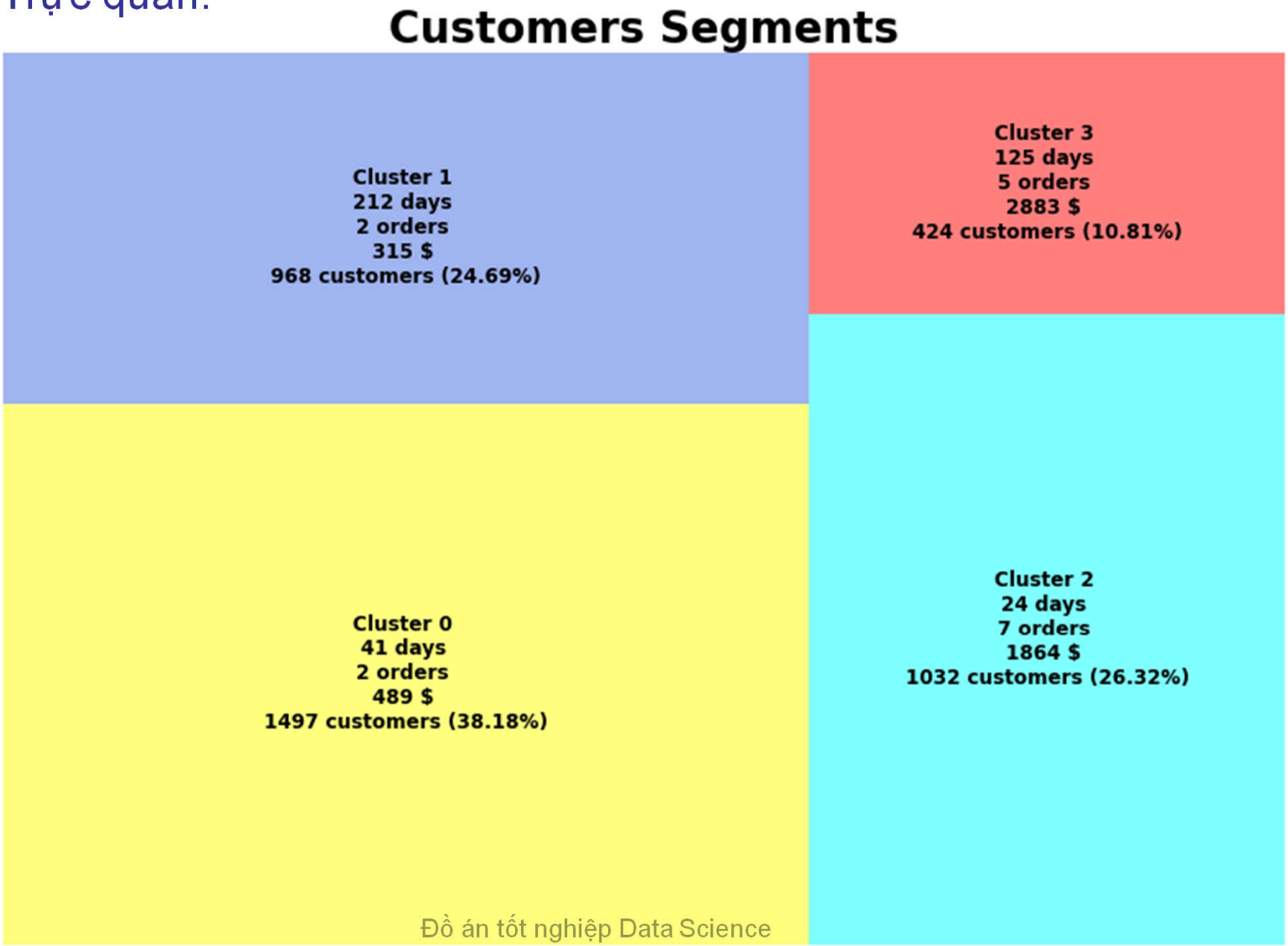
Dữ liệu đã scaled + GMM

- KH được phân cụm như sau:

	Cluster	RecencyMean	FrequencyMean	MonetaryMean	Count	Percent
0	Cluster 0	41.0	2.0	489.0	1497	38.18
1	Cluster 1	212.0	2.0	315.0	968	24.69
2	Cluster 2	24.0	7.0	1864.0	1032	26.32
3	Cluster 3	125.0	5.0	2883.0	424	10.81

Dữ liệu đã scaled + GMM

- Trực quan:



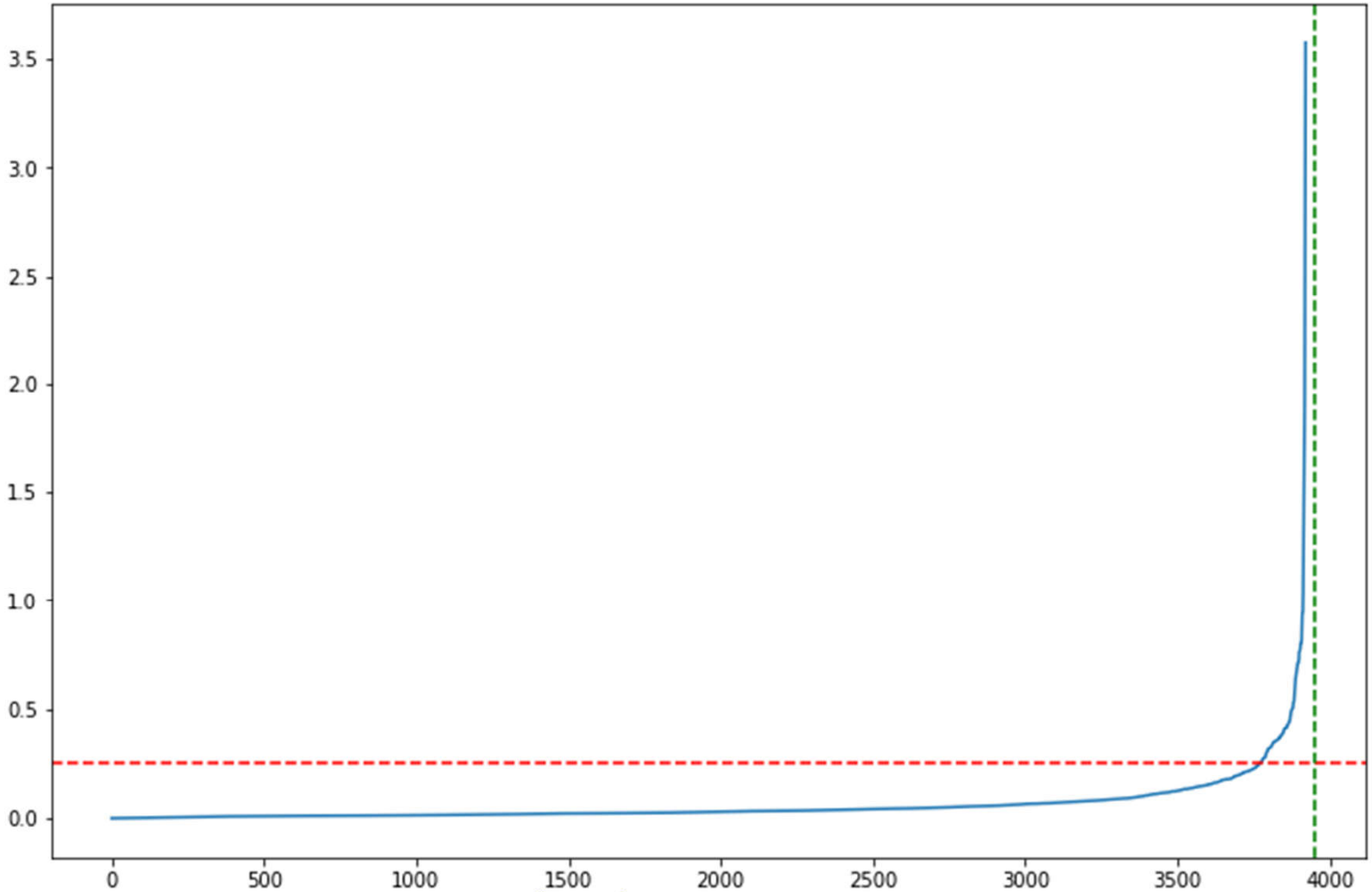
Dữ liệu đã scaled + GMM

Khách hàng chia làm 4 nhóm:

- Cluster 0: Thời gian mua hàng trung bình, số đơn hàng ít, số tiền ít chiếm 38.18%
- Cluster 1: Thời gian mua hàng xa, số đơn hàng ít, số tiền ít (KH có nguy cơ ra đi) chiếm 24.69%
- Cluster 2: Thời gian mua hàng gần, số đơn hàng lớn, số tiền lớn chiếm 26.32%
- Cluster 3: Khách hàng có thời gian mua hàng xa, số đơn hàng khá, số tiền rất lớn chiếm 10.81%

Dữ liệu đã scaled + DBSCAN

- Áp dụng DBSCAN với dữ liệu đã scaled
- Sử dụng elbow method để lựa chọn tham số epsilon (khoảng cách tối đa giữa 2 mẫu trong 1 cụm) = 0.25; min_samples = 2 * 3 = 6



Dữ liệu đã scaled + DBSCAN

- Với tham số đã chọn, KH được phân thành 26 cụm. Số lượng này quá lớn với bài toán phân cụm KH nên không phù hợp.

```
unique(dbscan_model.labels_)
```

```
array([-1,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
       16, 17, 18, 19, 20, 21, 22, 23, 24])
```

Kết luận

- Rất khó để nói cách phân cụm KH nào tốt hơn, điều này tùy thuộc vào tiêu chí cụ thể doanh nghiệp đưa ra
- Cá nhân em lựa chọn cách phân cụm KH có sự kết hợp giữa RFM và GMM do tỷ lệ các nhóm hợp lý và tính chất KH trong mỗi nhóm rõ ràng hơn.
- Tùy theo chiến lược của doanh nghiệp, cần phối hợp với các phòng ban khác để đề ra chiến lược marketing và chăm sóc KH phù hợp cho mỗi nhóm KH để bán được nhiều sản phẩm hơn từ đó tăng doanh thu và lợi nhuận cho doanh nghiệp.