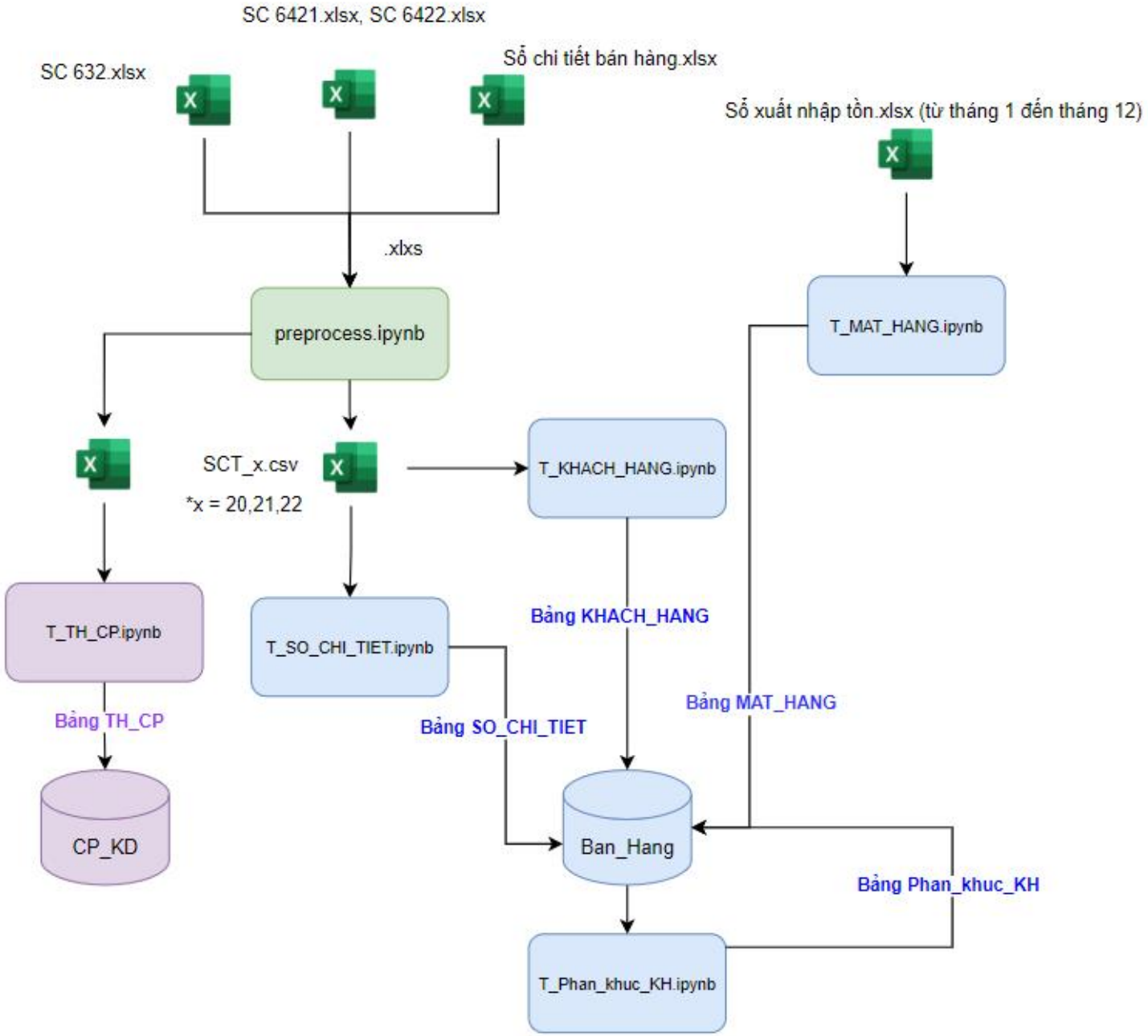
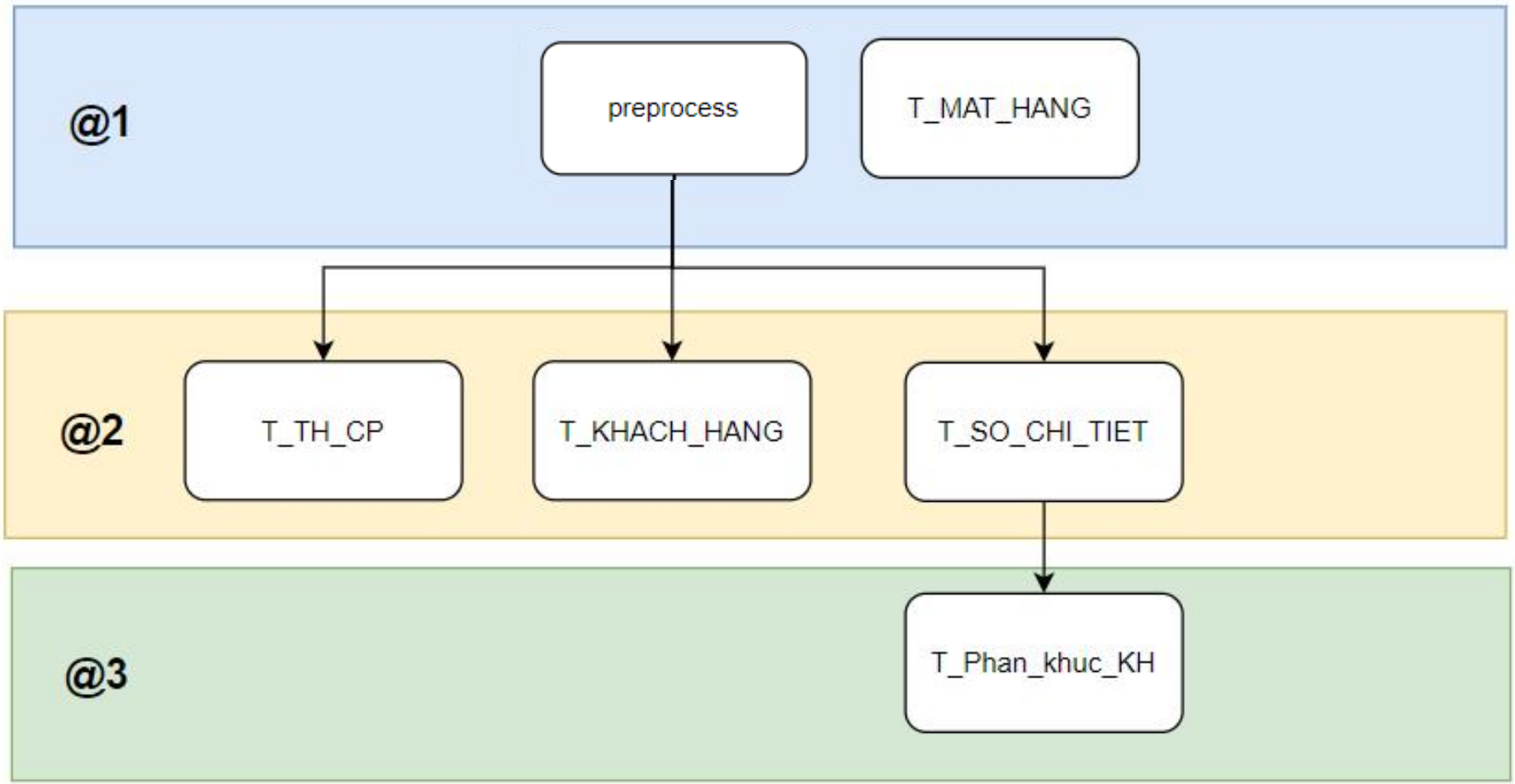
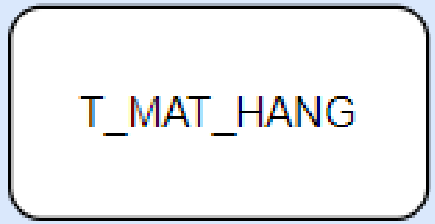


sơ đồ tóm tắt đường đi của dữ liệu (Xử lý bằng python)



Sơ đồ thứ tự chạy các file python (ipynb) để xử lý dữ liệu

**** Cùng hàng không cần xét thứ tự***



@1

GHI CHÚ

I) File này dùng để tổng hợp và phân loại dựa trên danh sách mặt hàng xuất nhập trong năm. Sau đó tạo và đưa dữ liệu vào bảng MAT_HANG trong CSDL Ban_Hang

II) Sử dụng dữ liệu từ sổ xuất nhập tồn

In [6]:
```
import pandas as pd
import numpy as np
import os
import pyodbc
import pandas as pd
```

B1 Tổng hợp 12 file xuất nhập tồn theo tháng của mỗi năm

In [7]:
```
def Process_Product(directory,skip_rows=4):
    combined_df = pd.DataFrame()
    for i in range(1, 13):
        # Construct the file name
        file_name = os.path.join(directory, f"TONG_HOP_TON_KHO {i}.xlsx")

        # Read the Excel file into a DataFrame
        df = pd.read_excel(file_name,skiprows = skip_rows)

        # Remove the last record from the DataFrame
        df = df[:-1]

        # Select columns from index 1 to 3
        df_selected = df.iloc[:, 1:4]

        # Append the selected columns DataFrame to the combined DataFrame
        combined_df = combined_df.append(df_selected, ignore_index=True)

    # Drop duplicates to keep only distinct values
    combined_df = combined_df.drop_duplicates()
    #Reset index
    combined_df = combined_df.reset_index(drop=True)
    # Rename column
    combined_df = combined_df.rename(columns={
                        'Unnamed: 1': 'Mã hàng',
                        'Unnamed: 2': 'Tên hàng',
                        'Unnamed: 3': 'ĐVT'}
                                )
    return combined_df
```

In [8]:
```
# Directory where Excel files are located
nam_2020 = 'D://THUCTAP//XUAT NHAP TON 2020//'
nam_2021 = 'D://THUCTAP//XUAT NHAP TON 2021//'
nam_2022 = 'D://THUCTAP//XUAT NHAP TON 2022//'
```

In [9]:
```
DSSP_20 = Process_Product(nam_2020)
DSSP_21 = Process_Product(nam_2021)
DSSP_22 = Process_Product(nam_2022,3)
```

In [10]:
```
DSSP_22
```

Out[10]:

	Mã hàng	Tên hàng	ĐVT
0	BH_150G	Bò hầm 150g	Thùng
1	BH_150GL	Bò hầm (lon in) hộp 150g	Hộp
2	BV_BO500G	Bò viên 3 ngon_gói 500g	Gói
3	BX_170G	Bò xay 170g	Thùng
4	CA_FUNNY500	Cá viên Funny gói 500g	Gói
...	...	...	...
215	TL_MEVIOSOB	Thuốc lá điếu lọc Mevius Original Blue	Gói
216	NY_HC460KM	Nước yến hạt chia Nutizen chai 460ml	Chai
217	TL_LOTUSRCND	Thuốc lá bao Lotus nâu - RCNĐ	Bao
218	X_BB20GH1KG	XXTT Boom Boom 20g hũ 1kg	Hũ
219	X_BB38G5+1	XXTT Boom Boom 38g (5+1) gói 228g	Thùng

220 rows × 3 columns

B2 Phân loại hàng dựa trên kí tự trong mã hàng

In [11]:
```
# Lấy ký tự từ Mã hàng để phân loại hàng
def extract(value):
    return value.split('_')[0]
```

In [12]:
```
# phân loại hàng
def map_chr_to_new_column(chr_value):
    if chr_value == 'TL':
        return 'Thuốc lá'
    elif chr_value in ['X','XX','CHA','BH','HH','BV','C','H','PT','HN','BX','G','HQ']:
        return 'SP từ Thịt'
    else:
        return 'Các SP khác'
```

In [13]:
```
def Apply(df):
    df['chr'] = df['Mã hàng'].apply(extract)
    df['Loại hàng'] = df['chr'].apply(map_chr_to_new_column)
    df.drop(columns=['chr'], inplace=True)
    return df
```

In [14]:
```
DSSP20 = Apply(DSSP_20)
DSSP21 = Apply(DSSP_21)
DSSP22 = Apply(DSSP_22)
```

In [15]:
```
DSSP21
```

Out[15]:

	Mã hàng	Tên hàng	ĐVT	Loại hàng
0	BH_150G	Bò hầm 150g	Thùng	SP từ Thịt
1	BH_150GM	Bò hầm 150g	Hộp	SP từ Thịt
2	BV_BO500G	Bò viên 3 ngon_gói 500g	Gói	SP từ Thịt
3	BX_170G	Bò xay 170g	Thùng	SP từ Thịt
4	C_CNXD	Cá ngừ xốt dầu	Thùng	SP từ Thịt
...	...	...	...	...
156	CF_SNGV240G	Ông Bầu OB cà phê sữa nóng hoà tan hộp giấy và...	Hộp	Các SP khác
157	CF_TGV240G	OB cà phê trứng giấy vàng 240g (10 gói x 24g. ...	Hộp	Các SP khác
158	HH_150GL	Heo hầm (lon in) hộp 150g	Hộp	SP từ Thịt
159	X_BRMO500	Ba rọi xốt mật ong_gói 500g	Gói	SP từ Thịt
160	X_SG500G	Xúc xích sụn giòn 500g ĐL	Gói	SP từ Thịt

161 rows × 4 columns

In [16]:
```
DSSP20.to_csv('D://THUCTAP//data_da_xuly//DSSP20.csv', index=False, encoding='utf-16', sep='\t')
DSSP21.to_csv('D://THUCTAP//data_da_xuly//DSSP21.csv', index=False, encoding='utf-16', sep='\t')
DSSP22.to_csv('D://THUCTAP//data_da_xuly//DSSP22.csv', index=False, encoding='utf-16', sep='\t')
```

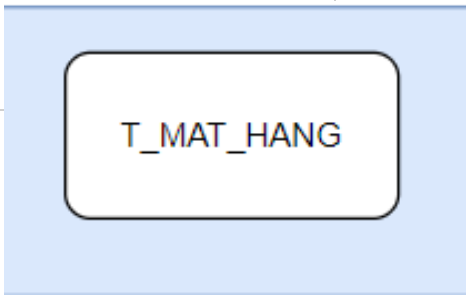
B3 ĐƯA DỮ LIỆU VÀO CSDL

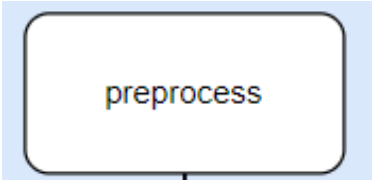
In [17]:
```
# Kết nối với CSDL
conn = pyodbc.connect(
Trusted_Connection = "Yes",
Driver = '{ODBC Driver 17 for SQL Server}',
Server = "DESKTOP-MDDIIDJ\MSSQLSERVER01",
Database = 'Ban_Hang')
cursor = conn.cursor()
```

In [18]:
```
#Tạo bảng MAT_HANG
#cursor.execute("CREATE TABLE MAT_HANG(Ma_hang varchar(20) PRIMARY KEY,Ten_hang nvarchar(150),DVT nvarchar(10
```

In [19]:
```
#Tạo hàm để đẩy dữ liệu vào bảng MAT_HANG trong CSDL
def Insert_Mat_hang(df):
    for row in df.itertuples():
        cursor.execute('''
                        IF NOT EXISTS (SELECT 1 FROM BAN_HANG.dbo.MAT_HANG WHERE Ma_hang = ?)
                        BEGIN
                            INSERT INTO BAN_HANG.dbo.MAT_HANG(Ma_hang, Ten_hang, DVT, Loai_hang)
                            VALUES(?, ?, ?, ?)
                        END
                        ''',
                        row[1],
                        row[1],  # Ma_hang
                        row[2],  # Ten_hang
                        row[3],  # DVT
                        row[4]   # Loai_hang
                       )
    conn.commit()
```

In [20]:
```
#Insert_Mat_hang(DSSP20)
#Insert_Mat_hang(DSSP21)
#Insert_Mat_hang(DSSP22)
```





@1

GHI CHÚ

I) File này dùng để xử lý dữ liệu kế toán ban đầu thu thập được

II) Sử dụng dữ liệu sổ chi tiết bán hàng, sổ cái 6421, sổ cái 6422, sổ cái 632

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
import re
```

XU LY TK 632

In [3]:
```
TK632_20 = pd.read_excel('D://THUCTAP//SO CAI 2020//SC 632.xlsx')
TK632_21 = pd.read_excel('D://THUCTAP//SO CAI 2021//SC 632.xlsx')
TK632_22 = pd.read_excel('D://THUCTAP//SO CAI 2022//SC 632.xlsx')
```

In [4]:
```
#Format chung cho dữ liệu SC 632,SC 642x
def Format_SC(df):
    #Bỏ cột không cần thiết và Hàng bị lỗi format
    df = df.iloc[:, 0:8]
    df = df.dropna()

    df.reset_index(drop=True, inplace=True)
    # Lưu dòng đầu tiên vào biến temp và loại bỏ nó khỏi DataFrame
    temp = df.iloc[0]
    df = df[1:]
    # Gán tên cột từ temp
    df.columns = temp
    # Bỏ hàng "kết chuyển sang tk 911"
    df = df[~df['TK đối ứng'].str.contains('911')]
    df.reset_index(drop=True, inplace=True)
    #
    df['Ngày chứng từ'] = pd.to_datetime(df['Ngày chứng từ'])
    #
    columns_to_drop = ['Ngày hạch toán','TK đối ứng','Phát sinh Có']
    df.drop(columns=columns_to_drop, inplace=True)
    #
    df.rename(columns={'Phát sinh Nợ': 'Phát sinh'}, inplace=True)
    #
    return df
```

In [5]:
```
#Thêm tách mục diễn giải để lấy tên KH và hoá đơn cho SC 632
def Format_632(df):
    df = Format_SC(df)
    pattern = r'(?i)(?:bán hàng\s*)(.*?)(?:\s*theo hóa đơn\s*)(\b\d+\b)$'
    df[['Tên khách hàng','Số hóa đơn']] = df['Diễn giải'].str.extract(pattern)
    df['Số hóa đơn'] = "'" + df['Số hóa đơn'].astype(str)
    df.drop(columns=['Diễn giải'], inplace=True)
    return df
```

In [6]:
```
TK632_20 = Format_632(TK632_20)
TK632_21 = Format_632(TK632_21)
TK632_22 = Format_632(TK632_22)
```

In [7]:
```
TK632_20.head(3)
```

Out[7]:

	Ngày chứng từ	Số chứng từ	Tài khoản	Phát sinh	Tên khách hàng	Số hóa đơn
0	2020-01-02	XK001	632	163625453	Vũ Như Hùng	'0001390
1	2020-01-02	XK002	632	49985000	Hồ Thị Xuân	'0001391
2	2020-01-02	XK003	632	336824815	Đại lý Tú Hạnh	'0001384

In [8]:
```
TK632_22.head(3)
```

Out[8]:

	Ngày chứng từ	Số chứng từ	Tài khoản	Phát sinh	Tên khách hàng	Số hóa đơn
0	2022-01-02	PXK0001	632	191016361	Nguyễn Thị Lý	'1101
1	2022-01-02	PXK0002	632	252287302	Đậu Thanh Hiển	'1102
2	2022-01-03	PXK0003	632	244812458	Đại Lý Tiến Thúy	'1103

XU LY TAI KHOAN 6421

In [9]:
```
TK6421_20 = pd.read_excel('D://THUCTAP//SO CAI 2020//SC 6421.xlsx')
TK6421_21 = pd.read_excel('D://THUCTAP//SO CAI 2021//SC 6421.xlsx')
TK6421_22 = pd.read_excel('D://THUCTAP//SO CAI 2022//SC 6421.xlsx')
```

In [10]:
```
# sử dụng hàm Format_SC() để xử lý shape của TK 6422
TK6421_20 = Format_SC(TK6421_20)
TK6421_21 = Format_SC(TK6421_21)
TK6421_22 = Format_SC(TK6421_22)
```

In [11]:
```
TK6421_20.head(3)
```

Out[11]:

	Ngày chứng từ	Số chứng từ	Diễn giải	Tài khoản	Phát sinh
0	2020-01-30	NVK033	Mua xăng, dầu	6421	5459745
1	2020-02-01	PC00053	Chi thanh toán tiền mua dầu DO	6421	6229636
2	2020-02-04	PC00056	Chi thanh toán tiền mua dầu DO	6421	912227

In [12]:
```
TK6421_22.head(3)
```

Out[12]:

	Ngày chứng từ	Số chứng từ	Diễn giải	Tài khoản	Phát sinh
0	2022-01-11	NVK2349	Mua xăng, dầu DO	6421	6337163
1	2022-01-21	NVK2364	Mua xăng, dầu DO	6421	6134700
2	2022-01-30	PC22_049	Chi thanh toán tiền lương, ăn ca tháng 01	6421	3650000

XU LY TAI KHOAN 6422

In [13]:
```
TK6422_20 = pd.read_excel('D://THUCTAP//SO CAI 2020//SC 6422.xlsx')
TK6422_21 = pd.read_excel('D://THUCTAP//SO CAI 2021//SC 6422.xlsx')
TK6422_22 = pd.read_excel('D://THUCTAP//SO CAI 2022//SC 6422.xlsx')
```

In [14]:
```
TK6422_22.tail(5)
```

Out[14]:

	SỔ CHI TIẾT CÁC TÀI KHOẢN	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7
541	2022-12-31 00:00:00	2022-12-31 00:00:00	NVK3537	Trích khấu hao TSCĐ	6422	2141	111460601	0
542	2022-12-31 00:00:00	2022-12-31 00:00:00	NVK3544	Kết chuyển CP QLDN để xác định kết quả HĐKD	6422	911	0	749465285
543	NaN	NaN	NaN	Cộng	6422	NaN	749465285	749465285
544	NaN	NaN	NaN	Số dư cuối kỳ	6422	NaN	NaN	NaN
545	Số dòng = 543	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [15]:
```
# sử dụng hàm Format_SC() để xử lý shape của TK 6422
TK6422_20 = Format_SC(TK6422_20)
TK6422_21 = Format_SC(TK6422_21)
TK6422_22 = Format_SC(TK6422_22)
```

In [16]:
```
TK6422_20.head(3)
```

Out[16]:

	Ngày chứng từ	Số chứng từ	Diễn giải	Tài khoản	Phát sinh
0	2020-01-02	UNC001	Phí chuyển tiền	6422	928615
1	2020-01-05	PC00008	Chi thanh toán cước DV viễn thông	6422	72146
2	2020-01-05	PC00009	Chi thanh toán cước DV viễn thông	6422	177990

In [17]:
```
TK6422_22.head(3)
```

Out[17]:

	Ngày chứng từ	Số chứng từ	Diễn giải	Tài khoản	Phát sinh
0	2022-01-01	PC22_001	Chi thanh toán cước DV viễn thông	6422	65718
1	2022-01-02	PC22_002	Chi thanh toán cước DV viễn thông	6422	199280
2	2022-01-04	NVK3515	Hạch toán thuế môn bài năm 2022	6422	3000000

Gộp 6421 và 6422 đã xử lý qua 3 năm

In [18]:
```
#gộp 6421
TH_6421 = pd.concat([TK6421_20, TK6421_21, TK6421_22], ignore_index=True)
TH_6421.head(3)
```

Out[18]:

	Ngày chứng từ	Số chứng từ	Diễn giải	Tài khoản	Phát sinh
0	2020-01-30	NVK033	Mua xăng, dầu	6421	5459745
1	2020-02-01	PC00053	Chi thanh toán tiền mua dầu DO	6421	6229636
2	2020-02-04	PC00056	Chi thanh toán tiền mua dầu DO	6421	912227

In [19]:
```
#gộp 6422
TH_6422 = pd.concat([TK6422_20, TK6422_21, TK6422_22], ignore_index=True)
TH_6422.head(3)
```

Out[19]:

	Ngày chứng từ	Số chứng từ	Diễn giải	Tài khoản	Phát sinh
0	2020-01-02	UNC001	Phí chuyển tiền	6422	928615
1	2020-01-05	PC00008	Chi thanh toán cước DV viễn thông	6422	72146
2	2020-01-05	PC00009	Chi thanh toán cước DV viễn thông	6422	177990

Tổng hợp 6421 và 6422

In [20]:
```
def replace_dien_giai(value):
    if 'lương' in value:
        return 'Chi phí nhân sự'
    elif 'xăng' in value or 'dầu' in value:
        return 'Chi phí xăng dầu'
    elif 'đường bộ' in value:
        return 'chi phí đường bộ'
    elif 'khấu hao' in value and 'TSCĐ' in value:
        return 'Chi phí khấu hao TSCĐ'
    else:
        return value
```

In [21]:
```
def merger_642(df1,df2):
    df = pd.concat([df1, df2], ignore_index=True)
    df.rename(columns={'Tài khoản': 'Loại chi phí'}, inplace=True)
    df['Loại chi phí'] = df['Loại chi phí'].replace({'6421': 'Chi phí bán hàng','6422': 'Chi phí QLDN'})
    df['Diễn giải'] = df['Diễn giải'].apply(replace_dien_giai)
    return df
```

In [22]:
```
tong_hop_CP = merger_642(TH_6421, TH_6422)
tong_hop_CP
```

Out[22]:

	Ngày chứng từ	Số chứng từ	Diễn giải	Loại chi phí	Phát sinh
0	2020-01-30	NVK033	Chi phí xăng dầu	Chi phí bán hàng	5459745
1	2020-02-01	PC00053	Chi phí xăng dầu	Chi phí bán hàng	6229636
2	2020-02-04	PC00056	Chi phí xăng dầu	Chi phí bán hàng	912227
3	2020-02-29	NVK065	Chi phí xăng dầu	Chi phí bán hàng	10238328
4	2020-03-13	NVK079	Chi phí xăng dầu	Chi phí bán hàng	1068600
...	...	...	...	...	...
1614	2022-12-31	NVK3513	BHXH phải nộp	Chi phí QLDN	16796613
1615	2022-12-31	NVK3513	BHXH phải nộp	Chi phí QLDN	1682936
1616	2022-12-31	NVK3514	Chi phí nhân sự	Chi phí QLDN	157333400
1617	2022-12-31	NVK3536	Phân bổ chi phí trích trước	Chi phí QLDN	14982879
1618	2022-12-31	NVK3537	Chi phí khấu hao TSCĐ	Chi phí QLDN	111460601

1619 rows × 5 columns

Xu ly so chi tiet

In [23]:
```
dtype_dict = {
    'Số hóa đơn': str,
    'Số lượng bán': int}
```

In [24]:
```
SCT_20 = pd.read_excel('D://THUCTAP//SO CAI 2020//SO_CHI_TIET_BAN_HANG 2020.xlsx', skiprows = 2,dtype=dtype_d
SCT_21 = pd.read_excel('D://THUCTAP//SO CAI 2021//SO_CHI_TIET_BAN_HANG 2021.xlsx', skiprows = 2,dtype=dtype_d
SCT_22 = pd.read_excel('D://THUCTAP//SO CAI 2022//SO_CHI_TIET_BAN_HANG 2022.xlsx', skiprows = 2,dtype=dtype_d
```

In [25]:
```
def Format_SCT(df):
    df = df.dropna()
    df[["Ngày chứng từ"]] = df[["Ngày chứng từ"]].apply(pd.to_datetime)
    df['Số hóa đơn'] = "'" + df['Số hóa đơn'].astype(str)
    columns_to_drop = ['Ngày hạch toán','Số chứng từ','Ngày hóa đơn','Chiết khấu','Số lượng trả lại','Giá trị

    # Check if columns exist before attempting to drop them
    columns_to_drop_existing = [col for col in columns_to_drop if col in df.columns]
    if columns_to_drop_existing:
        df.drop(columns=columns_to_drop_existing, inplace=True)

    return df
```

In [26]:
```
SCT20 = Format_SCT(SCT_20)
SCT21 = Format_SCT(SCT_21)
SCT22 = Format_SCT(SCT_22)
```

```
D:\anaconda\lib\site-packages\pandas\core\frame.py:3641: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  self[k1] = value[k2]
C:\Users\TechCare\AppData\Local\Temp/ipykernel_24976/1214963531.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df['Số hóa đơn'] = "'" + df['Số hóa đơn'].astype(str)
D:\anaconda\lib\site-packages\pandas\core\frame.py:4906: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  return super().drop(
D:\anaconda\lib\site-packages\pandas\core\frame.py:3641: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  self[k1] = value[k2]
C:\Users\TechCare\AppData\Local\Temp/ipykernel_24976/1214963531.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df['Số hóa đơn'] = "'" + df['Số hóa đơn'].astype(str)
D:\anaconda\lib\site-packages\pandas\core\frame.py:4906: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
```

```
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  return super().drop(
D:\anaconda\lib\site-packages\pandas\core\frame.py:3641: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  self[k1] = value[k2]
C:\Users\TechCare\AppData\Local\Temp/ipykernel_24976/1214963531.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df['Số hóa đơn'] = "'" + df['Số hóa đơn'].astype(str)
D:\anaconda\lib\site-packages\pandas\core\frame.py:4906: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  return super().drop(
```

In [27]:
```
def merge_SCT(TK632,df):
    df = df.merge(TK632[['Số hóa đơn', 'Tên khách hàng']], how='left', on='Số hóa đơn')
    return df
```

In [28]:
```
SCT_20 = merge_SCT(TK632_20,SCT20)
SCT_21 = merge_SCT(TK632_21,SCT21)
```

In [29]:
```
SCT_21
```

Out[29]:

	Ngày chứng từ	Số hóa đơn	Mã hàng	Tên hàng	ĐVT	Số lượng bán	Đơn giá	Doanh số bán	Đơn giá vốn	Giá vốn	Tên khách hàng
0	2021-01-01	'0000228	TL_TL	Thuốc lá Thăng Long B.C	Thùng	3	3755000.0	11265000	3740000.00	11220000	Hoàng Thị Hương
1	2021-01-01	'0000228	TL_TLSLIM	Thuốc lá Thăng Long Slim	Thùng	2	4215000.0	8430000	4195000.00	8390000	Hoàng Thị Hương
2	2021-01-02	'0000229	TL_BLA	Thuốc lá Blue Seal Apple Slim	Thùng	1	3770000.0	3770000	3739999.91	3740000	Đại Lý Tiến Thúy
3	2021-01-02	'0000229	TL_RC	Thuốc lá Thăng Long Round Corner	Thùng	1	5750000.0	5750000	5735454.27	5735454	Đại Lý Tiến Thúy
4	2021-01-02	'0000229	TL_SP	Thuốc lá Sapa B.M	Thùng	2	2265000.0	4530000	2244990.71	4489981	Đại Lý Tiến Thúy
...	...	...	...	...	...	...	...	...	...	...	...
6640	2021-12-31	'0001100	TL_SP	Thuốc lá Sapa B.M	Thùng	2	2520000.0	5040000	2496594.68	4993189	Hà Toàn
6641	2021-12-31	'0001100	TL_TL	Thuốc lá Thăng Long B.C	Thùng	65	3910000.0	254150000	3912458.78	254309821	Hà Toàn
6642	2021-12-31	'0001100	TL_TLSLIM	Thuốc lá Thăng Long Slim	Thùng	15	4215000.0	63225000	4195000.00	62925000	Hà Toàn
6643	2021-12-31	'0001100	TL_VINADS	Thuốc lá Vinataba Demi Slims	Kiện	1	3470000.0	3470000	3457685.98	3457686	Hà Toàn
6644	2021-12-31	'0001100	D_TN	Diêm Thống Nhất	Kiện	1	356772.0	356772	354339.00	354339	Hà Toàn

6645 rows × 11 columns

In [30]:
```
def Extract_SCT22(df):
    pattern = r'(?i)bán hàng\s*(.*?)\s*theo hóa đơn'
    # Extract 'Ten_khach_hang' from the strings using the defined pattern
    df['Tên khách hàng'] = df['Diễn giải chung'].str.extract(pattern)
    return df
```

In [31]:
```
SCT_22 = Extract_SCT22(SCT22)
```

```
C:\Users\TechCare\AppData\Local\Temp/ipykernel_24976/65051391.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['Tên khách hàng'] = df['Diễn giải chung'].str.extract(pattern)
```

In [ ]:
```
def drop_saleoff(df):
```

In [32]:
```
SCT_22
```

Out[32]:

	Ngày chứng từ	Số hóa đơn	Diễn giải chung	Mã hàng	Tên hàng	ĐVT	Số lượng bán	Đơn giá	Doanh số bán	Đơn giá vốn	Giá trị hàng hoá xuất ra	Tên khách hàng
0	2022-01-02	'1101	Thu tiền bán hàng Nguyễn Thị Lý theo hóa đơn 1101	TL_TL	Thuốc lá Thăng Long B.C	Thùng	46	3955000.00	181930000	3912458.78	179973104	Nguyễn Thị Lý
1	2022-01-02	'1101	Thu tiền bán hàng Nguyễn Thị Lý theo hóa đơn 1101	HH_150GM	Heo hầm 150g	Hộp	144	23882.01	3439010	23716.00	3415104	Nguyễn Thị Lý
2	2022-01-02	'1101	Thu tiền bán hàng Nguyễn Thị Lý theo hóa đơn 1101	HH_3BM150G	Heo hầm 3 Bông Mai 150g	Thùng	2	1150257.50	2300515	1142261.83	2284524	Nguyễn Thị Lý
3	2022-01-02	'1101	Thu tiền bán hàng Nguyễn Thị Lý theo hóa đơn 1101	HN_THIT400G	Hạt nêm chuẩn vị thịt gói 400g	Gói	24	24168.00	580032	24000.00	576000	Nguyễn Thị Lý
4	2022-01-02	'1101	Thu tiền bán hàng Nguyễn Thị Lý theo hóa đơn 1101	X_3BM21GM	XXTT 3 Bông Mai bò 21g	Gói	180	7616.09	1370897	7563.15	1361367	Nguyễn Thị Lý
...	...	...	...	...	...	...	...	...	...	...	...	...
9588	2022-12-31	'1097	Thu tiền bán hàng Quyền Hướng theo hóa đơn 1097	TL_BLA	Thuốc lá Blue Seal Apple Slim	Thùng	1	3770000.00	3770000	3739999.98	3740000	Quyền Hướng
9589	2022-12-31	'1097	Thu tiền bán hàng Quyền Hướng theo hóa đơn 1097	TL_BR04	Thuốc lá Blue River MT BC (BR04)	Bao	300	5950.00	1785000	5908.84	1772652	Quyền Hướng

	Ngày chứng từ	Số hóa đơn	Diễn giải chung	Mã hàng	Tên hàng	ĐVT	Số lượng bán	Đơn giá	Doanh số bán	Đơn giá vốn	Giá trị hàng hoá xuất ra	Tên khách hàng
9590	2022-12-31	'1097	Thu tiền bán hàng Quyền Hướng theo hóa đơn 1097	TL_VINA	Thuốc lá Vinataba Sài Gòn	Thùng	1	7690000.00	7690000	7640000.00	7640000	Quyền Hướng
9591	2022-12-31	'1098	Thu tiền bán hàng Khách lẻ theo hóa đơn 1098	TL_TL	Thuốc lá Thăng Long B.C	Thùng	15	3955000.00	59325000	3939999.99	59100000	Khách lẻ
9592	2022-12-31	'1098	Thu tiền bán hàng Khách lẻ theo hóa đơn 1098	TL_TLSLIM	Thuốc lá Thăng Long Slim	Thùng	10	4650000.00	46500000	4629915.83	46299158	Khách lẻ

9593 rows × 12 columns

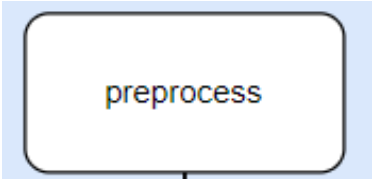
Save data

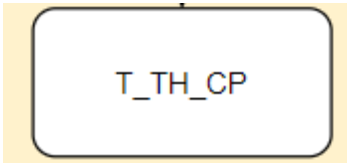
In [33]:
```
# 2020
TK632_20.to_csv('D://THUCTAP//data_da_xuly//632_20.csv', index=False,  encoding='utf-16', sep='\t')
#TK6421_20.to_csv('D://THUCTAP//data_da_xuly//6421_20.csv', index=False,  encoding='utf-16', sep='\t')
#TK6422_20.to_csv('D://THUCTAP//data_da_xuly//6422_20.csv', index=False,  encoding='utf-16', sep='\t')
SCT_20.to_csv('D://THUCTAP//data_da_xuly//SCT_20.csv', index=False,  encoding='utf-16', sep='\t')
```

In [34]:
```
# 2021
TK632_21.to_csv('D://THUCTAP//data_da_xuly//632_21.csv', index=False,  encoding='utf-16', sep='\t')
#TK6421_21.to_csv('D://THUCTAP//data_da_xuly//6421_21.csv', index=False,  encoding='utf-16', sep='\t')
#TK6422_21.to_csv('D://THUCTAP//data_da_xuly//6422_21.csv', index=False,  encoding='utf-16', sep='\t')
SCT_21.to_csv('D://THUCTAP//data_da_xuly//SCT_21.csv', index=False,  encoding='utf-16', sep='\t')
```

In [35]:
```
# 2022
TK632_22.to_csv('D://THUCTAP//data_da_xuly//632_22.csv', index=False,  encoding='utf-16', sep='\t')
#TK6421_22.to_csv('D://THUCTAP//data_da_xuly//6421_22.csv', index=False,  encoding='utf-16', sep='\t')
#TK6422_22.to_csv('D://THUCTAP//data_da_xuly//6422_22.csv', index=False,  encoding='utf-16', sep='\t')
SCT_22.to_csv('D://THUCTAP//data_da_xuly//SCT_22.csv', index=False,  encoding='utf-16', sep='\t')
```

In [36]:
```
#Tong_hop_CP
tong_hop_CP.to_csv('D://THUCTAP//data_da_xuly//tong_hop_CP.csv', index=False,  encoding='utf-16', sep='\t')
```





@2

GHI CHÚ

I) File này dùng để tổng hợp, phân loại chi phí. Sau đó đưa dữ liệu vào bảng TH_CP trong CSDL CP_KD

II) Sử dụng dữ liệu từ sổ cái 6421, 6422 đã được xử lý ở file preprocess

In [18]:
```
import pandas as pd
import numpy as np
```

In [14]:
```
#TK6421
```

In [2]:
```
TK6421_20 = pd.read_csv('D://THUCTAP//data_da_xuly//6421_20.csv', encoding='utf-16', sep='\t')
TK6421_21 = pd.read_csv('D://THUCTAP//data_da_xuly//6421_21.csv', encoding='utf-16', sep='\t')
TK6421_22 = pd.read_csv('D://THUCTAP//data_da_xuly//6421_22.csv', encoding='utf-16', sep='\t')
```

In [15]:
```
#TK6422
```

In [3]:
```
TK6422_20 = pd.read_csv('D://THUCTAP//data_da_xuly//6422_20.csv', encoding='utf-16', sep='\t')
TK6422_21 = pd.read_csv('D://THUCTAP//data_da_xuly//6422_21.csv', encoding='utf-16', sep='\t')
TK6422_22 = pd.read_csv('D://THUCTAP//data_da_xuly//6422_22.csv', encoding='utf-16', sep='\t')
```

In [16]:
```
#gộp 6421
```

In [4]:
```
result1 = pd.concat([TK6421_20, TK6421_21, TK6421_22], ignore_index=True)
result1['Loại chi phí']='Chi phí bán hàng'
result1
```

Out[4]:

	Ngay_chung_tu	So_chung_tu	Dien_giai	Phat_sinh	Loại chi phí
0	2020-02-01	PC00053	Chi thanh toán tiền mua dầu DO	6229636	Chi phí bán hàng
1	2020-02-04	PC00056	Chi thanh toán tiền mua dầu DO	912227	Chi phí bán hàng
2	2020-02-29	NVK065	Mua dầu DO	10238328	Chi phí bán hàng
3	2020-03-13	NVK079	Mua nợ dầu DO	1068600	Chi phí bán hàng
4	2020-03-15	NVK080	Mua nợ dầu DO	3493500	Chi phí bán hàng
...	...	...	...	...	...
138	2022-12-31	NVK3024	Mua dầu DO	5427391	Chi phí bán hàng
139	2022-12-31	NVK3514	Phân bổ lương năm 2022	1125899243	Chi phí bán hàng
140	2022-12-31	NVK3536	Phân bổ chi phí trích trước	126000000	Chi phí bán hàng
141	2022-12-31	NVK3537	Trích khấu hao TSCĐ	187773878	Chi phí bán hàng
142	2022-12-31	PC22_757	Chi thanh toán tiền lương, ăn ca tháng 12	4380000	Chi phí bán hàng

143 rows × 5 columns

In [17]:
```
#gộp 6422
```

In [5]:
```
result2 = pd.concat([TK6422_20, TK6422_21, TK6422_22], ignore_index=True)
result2['Loại chi phí']='Chi phí QLDN'
result2
```

Out[5]:

	Ngay_chung_tu	So_chung_tu	Dien_giai	Phat_sinh	Loại chi phí
0	2020-01-05	PC00008	Chi thanh toán cước DV viễn thông	72146	Chi phí QLDN
1	2020-01-05	PC00009	Chi thanh toán cước DV viễn thông	177990	Chi phí QLDN
2	2020-01-07	UNC011	Phí chuyển tiền	786761	Chi phí QLDN
3	2020-01-07	UNC012	Phí chuyển tiền	121880	Chi phí QLDN
4	2020-01-08	UNC019	Phí	9900	Chi phí QLDN
...	...	...	...	...	...
1467	2022-12-31	NVK3513	BHXH phải nộp	16796613	Chi phí QLDN
1468	2022-12-31	NVK3513	BHXH phải nộp	1682936	Chi phí QLDN
1469	2022-12-31	NVK3514	Phân bổ lương năm 2022	157333400	Chi phí QLDN
1470	2022-12-31	NVK3536	Phân bổ chi phí trích trước	14982879	Chi phí QLDN
1471	2022-12-31	NVK3537	Trích khấu hao TSCĐ	111460601	Chi phí QLDN

1472 rows × 5 columns

In [ ]:
```
#gộp 6421 và 6422
```

In [6]:
```
result = pd.concat([result1, result2], ignore_index=True)
```

In [7]:
```
result
```

Out[7]:

	Ngay_chung_tu	So_chung_tu	Dien_giai	Phat_sinh	Loại chi phí
0	2020-02-01	PC00053	Chi thanh toán tiền mua dầu DO	6229636	Chi phí bán hàng
1	2020-02-04	PC00056	Chi thanh toán tiền mua dầu DO	912227	Chi phí bán hàng
2	2020-02-29	NVK065	Mua dầu DO	10238328	Chi phí bán hàng
3	2020-03-13	NVK079	Mua nợ dầu DO	1068600	Chi phí bán hàng
4	2020-03-15	NVK080	Mua nợ dầu DO	3493500	Chi phí bán hàng
...	...	...	...	...	...
1610	2022-12-31	NVK3513	BHXH phải nộp	16796613	Chi phí QLDN
1611	2022-12-31	NVK3513	BHXH phải nộp	1682936	Chi phí QLDN
1612	2022-12-31	NVK3514	Phân bổ lương năm 2022	157333400	Chi phí QLDN
1613	2022-12-31	NVK3536	Phân bổ chi phí trích trước	14982879	Chi phí QLDN
1614	2022-12-31	NVK3537	Trích khấu hao TSCĐ	111460601	Chi phí QLDN

1615 rows × 5 columns

In [8]:
```
result
```

Out[8]:

	Ngay_chung_tu	So_chung_tu	Dien_giai	Phat_sinh	Loại chi phí
0	2020-02-01	PC00053	Chi thanh toán tiền mua dầu DO	6229636	Chi phí bán hàng
1	2020-02-04	PC00056	Chi thanh toán tiền mua dầu DO	912227	Chi phí bán hàng
2	2020-02-29	NVK065	Mua dầu DO	10238328	Chi phí bán hàng
3	2020-03-13	NVK079	Mua nợ dầu DO	1068600	Chi phí bán hàng
4	2020-03-15	NVK080	Mua nợ dầu DO	3493500	Chi phí bán hàng
...	...	...	...	...	...
1610	2022-12-31	NVK3513	BHXH phải nộp	16796613	Chi phí QLDN
1611	2022-12-31	NVK3513	BHXH phải nộp	1682936	Chi phí QLDN
1612	2022-12-31	NVK3514	Phân bổ lương năm 2022	157333400	Chi phí QLDN
1613	2022-12-31	NVK3536	Phân bổ chi phí trích trước	14982879	Chi phí QLDN
1614	2022-12-31	NVK3537	Trích khấu hao TSCĐ	111460601	Chi phí QLDN

1615 rows × 5 columns

In [9]:
```
def replace_dien_giai(value):
    if 'lương' in value:
        return 'Chi phí nhân sự'
    elif 'xăng' in value or 'dầu' in value:
        return 'Chi phí xăng dầu'
    elif 'đường bộ' in value:
        return 'chi phí đường bộ'
    elif 'khấu hao' in value and 'TSCĐ' in value:
        return 'Chi phí khấu hao TSCĐ'
    else:
        return value
```

In [11]:
```
result['Dien_giai'] = result['Dien_giai'].apply(replace_dien_giai)
```

In [13]:
```
result
```

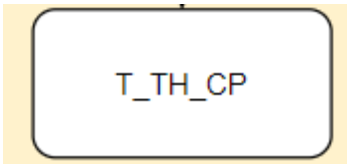
Out[13]:

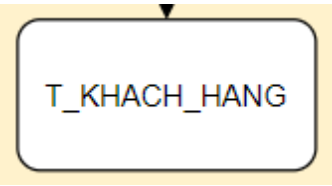
	Ngay_chung_tu	So_chung_tu	Dien_giai	Phat_sinh	Loại chi phí
0	2020-02-01	PC00053	Chi phí xăng dầu	6229636	Chi phí bán hàng
1	2020-02-04	PC00056	Chi phí xăng dầu	912227	Chi phí bán hàng
2	2020-02-29	NVK065	Chi phí xăng dầu	10238328	Chi phí bán hàng
3	2020-03-13	NVK079	Chi phí xăng dầu	1068600	Chi phí bán hàng
4	2020-03-15	NVK080	Chi phí xăng dầu	3493500	Chi phí bán hàng
...	...	...	...	...	...
1610	2022-12-31	NVK3513	BHXH phải nộp	16796613	Chi phí QLDN
1611	2022-12-31	NVK3513	BHXH phải nộp	1682936	Chi phí QLDN
1612	2022-12-31	NVK3514	Chi phí nhân sự	157333400	Chi phí QLDN
1613	2022-12-31	NVK3536	Phân bổ chi phí trích trước	14982879	Chi phí QLDN
1614	2022-12-31	NVK3537	Chi phí khấu hao TSCĐ	111460601	Chi phí QLDN

1615 rows × 5 columns

In [9]:
```
result.to_csv('D://THUCTAP//data_da_xuly//TH_CP.csv', index=False,  encoding='utf-16', sep='\t')
```

In [ ]:
```

```





@2

GHI CHÚ

I) File này dùng để tạo mã khách hàng mới. Sau đó tạo và đưa dữ liệu vào bảng KHACH_HANG trong CSDL Ban_Hang

II) Sử dụng dữ liệu sổ chi tiết bán hàng đã qua xử lý bằng file preprocess

In [1]:
```
import pandas as pd
import numpy as np
import pyodbc
```

In [2]:
```
SCT20 = pd.read_csv('D://THUCTAP//data_da_xuly//SCT_20.csv', encoding='utf-16', sep='\t')
SCT21 = pd.read_csv('D://THUCTAP//data_da_xuly//SCT_21.csv', encoding='utf-16', sep='\t')
SCT22 = pd.read_csv('D://THUCTAP//data_da_xuly//SCT_22.csv', encoding='utf-16', sep='\t')
```

B1 TẠO MÃ KH BAN ĐẦU

In [3]:
```
KHACH_HANG_20 = SCT20['Tên khách hàng'].unique()
```

In [4]:
```
KHACH_HANG_20 = pd.DataFrame({'Tên khách hàng': KHACH_HANG_20})
```

In [5]:
```
KHACH_HANG_20['Ma_KH'] = 'KH' + (KHACH_HANG_20.index + 1).astype(str).str.zfill(5)
```

In [6]:
```
KHACH_HANG_20.tail(5)
```

Out[6]:

	Tên khách hàng	Ma_KH
110	Phan Thị Thuỷ	KH00111
111	Siêu thị Vimart	KH00112
112	CÔNG TY CP ĐẦU TƯ XÂY DỰNG VÀ THƯƠNG MẠI KHÁNH...	KH00113
113	CÔNG TY TNHH TƯ VẤN XÂY DỰNG DÂN DỤNG HÀ TĨNH	KH00114
114	Cty CP XD và TM Linh Châu	KH00115

In [7]:
```
def Unique_name(df):
    KH = df['Tên khách hàng'].unique()
    KH = pd.DataFrame({'Tên khách hàng': KH})
    return KH
```

B2 Đưa dữ liệu vào CSDL

In [8]:
```
# Kết nối với CSDL
conn = pyodbc.connect(
Trusted_Connection = "Yes",
Driver = '{ODBC Driver 17 for SQL Server}',
Server = "DESKTOP-MDDIIDJ\MSSQLSERVER01",
Database = 'Ban_Hang')
cursor = conn.cursor()
```

In [9]:
```
#Tạo bảng KHACH_HANG
cursor.execute("CREATE TABLE KHACH_HANG(Ten_khach_hang nvarchar(150), Ma_KH varchar(8))")
```

Out[9]:
```
<pyodbc.Cursor at 0x2121560d230>
```

In [10]:
```
#Thêm vào DS khách hàng ban đầu
def Insert_KH(df):
    for row in df.itertuples():
        cursor.execute('''
                        INSERT INTO Ban_Hang.dbo.KHACH_HANG(Ten_khach_hang, Ma_KH)
                        VALUES(?, ?)
                        ''',
                        row[1],
                        row[2]
                       )
    conn.commit()
```

In [11]:
```
# chuyển định dạng dữ liệu của bảng KH
def convert_columns_to_string(df):
    return df.astype(str)
```

In [12]:
```
KHACH_HANG_20=convert_columns_to_string(KHACH_HANG_20)
```

In [13]:
```
Insert_KH(KHACH_HANG_20)
```

In [14]:
```
# Tạo hàm thêm mới DS Khách hàng từ các năm tiếp theo.
def Insert_KH_n(df):
    for row in df.itertuples():
        # Kiểm tra xem tên khách hàng đã tồn tại trong cơ sở dữ liệu hay chưa
        cursor.execute('''
                        SELECT COUNT(*)
                        FROM Ban_Hang.dbo.KHACH_HANG
                        WHERE Ten_khach_hang = ?
                        ''',
                        row[1]
                       )
        count = cursor.fetchone()[0]

        if count == 0:  # Nếu tên khách hàng chưa tồn tại
            # Lấy ra ID lớn nhất hiện tại
            cursor.execute('''
                            SELECT MAX(CAST(SUBSTRING(Ma_KH, 3, LEN(Ma_KH)) AS INT))
                            FROM Ban_Hang.dbo.KHACH_HANG
                            ''')
            max_id = cursor.fetchone()[0]  # Lấy ID lớn nhất
            if max_id is None:
                new_id = 1
            else:
                new_id = max_id + 1

            # Tạo mã khách hàng mới
            new_kh_id = 'KH' + str(new_id).zfill(5)

            # Thêm dữ liệu mới vào cơ sở dữ liệu
            cursor.execute('''
                            INSERT INTO Ban_Hang.dbo.KHACH_HANG(Ten_khach_hang, Ma_KH)
                            VALUES(?, ?)
                            ''',
                            row[1],
                            new_kh_id
                           )
    conn.commit()
```

KH năm 2021

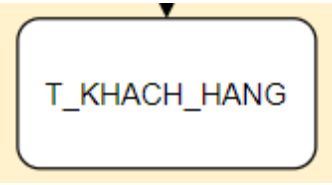
In [15]:
```
# Lấy ra tên KH giao dịch trong năm 2021
KHACH_HANG_21 = Unique_name(SCT21)
```

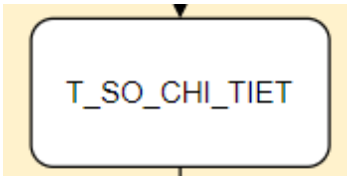
In [16]:
```
# thay đổi định dạng DL cho DS KH năm 2021
KHACH_HANG_21 = convert_columns_to_string(KHACH_HANG_21)
# Thêm DS KH năm 2021 vào CSDL
Insert_KH_n(KHACH_HANG_21)
```

KH năm 2022

In [17]:
```
# Lấy ra tên KH giao dịch trong năm 2022
KHACH_HANG_22 = Unique_name(SCT22)
```

In [18]:
```
# thay đổi định dạng DL cho DS KH năm 2022
KHACH_HANG_22 = convert_columns_to_string(KHACH_HANG_22)
# Thêm DS KH năm 2022 vào CSDL
Insert_KH_n(KHACH_HANG_22)
```





@2

GHI CHÚ

I) File này dùng để mã hoá tên khách hàng trong bảng sổ chi tiết bán hàng. Sau đó tạo và đưa dữ liệu vào bảng SO_CHI_TIET trong CSDL Ban_Hang

II) Sử dụng dữ liệu sổ chi tiết đã qua xử lý bằng file preprocess và dữ liệu từ bảng KHACH_HANG trong CSDL Ban_Hang ¶

In [2]:
```
import pandas as pd
import numpy as np
import pyodbc
```

In [3]:
```
SCT20 = pd.read_csv('D://THUCTAP//data_da_xuly//SCT_20.csv', encoding='utf-16', sep='\t')
SCT21 = pd.read_csv('D://THUCTAP//data_da_xuly//SCT_21.csv', encoding='utf-16', sep='\t')
SCT22 = pd.read_csv('D://THUCTAP//data_da_xuly//SCT_22.csv', encoding='utf-16', sep='\t')
```

In [4]:
```
SCT21.head(3)
```

Out[4]:

	Ngày chứng từ	Số hóa đơn	Mã hàng	Tên hàng	ĐVT	Số lượng bán	Đơn giá	Doanh số bán	Đơn giá vốn	Giá vốn	Tên khách hàng
0	2021-01-01	'0000228	TL_TL	Thuốc lá Thăng Long B.C	Thùng	3	3755000.0	11265000	3740000.00	11220000	Hoàng Thị Hương
1	2021-01-01	'0000228	TL_TLSLIM	Thuốc lá Thăng Long Slim	Thùng	2	4215000.0	8430000	4195000.00	8390000	Hoàng Thị Hương
2	2021-01-02	'0000229	TL_BLA	Thuốc lá Blue Seal Apple Slim	Thùng	1	3770000.0	3770000	3739999.91	3740000	Đại Lý Tiến Thúy

Nối mã KH

In [5]:
```
query = "SELECT * FROM KHACH_HANG"
KH = pd.read_sql_query(query, conn)
```

In [6]:
```
KH.head(3)
```

Out[6]:

	Ten_khach_hang	Ma_KH
0	Vũ Như Hùng	KH00001
1	Hồ Thị Xuân	KH00002
2	Đại lý Tú Hạnh	KH00003

In [7]:
```
def merge_SCT(SCT,KH):
    SCT = SCT.merge(KH, how='left', left_on = 'Tên khách hàng', right_on = 'Ten_khach_hang')
    # Drop the redundant key column 'Ma_hoa_don'
    SCT.drop(columns=['Ten_khach_hang'], inplace=True)
    return SCT
```

In [8]:
```
SCT_20 = merge_SCT(SCT20,KH)
SCT_21 = merge_SCT(SCT21,KH)
SCT_22 = merge_SCT(SCT22,KH)
```

In [9]:
```
SCT_21
```

Out[9]:

	Ngày chứng từ	Số hóa đơn	Mã hàng	Tên hàng	ĐVT	Số lượng bán	Đơn giá	Doanh số bán	Đơn giá vốn	Giá vốn	Tên khách hàng	Ma_KH
0	2021-01-01	'0000228	TL_TL	Thuốc lá Thăng Long B.C	Thùng	3	3755000.0	11265000	3740000.00	11220000	Hoàng Thị Hương	KH00116
1	2021-01-01	'0000228	TL_TLSLIM	Thuốc lá Thăng Long Slim	Thùng	2	4215000.0	8430000	4195000.00	8390000	Hoàng Thị Hương	KH00116
2	2021-01-02	'0000229	TL_BLA	Thuốc lá Blue Seal Apple Slim	Thùng	1	3770000.0	3770000	3739999.91	3740000	Đại Lý Tiến Thúy	KH00005
3	2021-01-02	'0000229	TL_RC	Thuốc lá Thăng Long Round Corner	Thùng	1	5750000.0	5750000	5735454.27	5735454	Đại Lý Tiến Thúy	KH00005
4	2021-01-02	'0000229	TL_SP	Thuốc lá Sapa B.M	Thùng	2	2265000.0	4530000	2244990.71	4489981	Đại Lý Tiến Thúy	KH00005
...	...	...	...	...	...	...	...	...	...	...	...	...
6640	2021-12-31	'0001100	TL_SP	Thuốc lá Sapa B.M	Thùng	2	2520000.0	5040000	2496594.68	4993189	Hà Toàn	KH00126
6641	2021-12-31	'0001100	TL_TL	Thuốc lá Thăng Long B.C	Thùng	65	3910000.0	254150000	3912458.78	254309821	Hà Toàn	KH00126
6642	2021-12-31	'0001100	TL_TLSLIM	Thuốc lá Thăng Long Slim	Thùng	15	4215000.0	63225000	4195000.00	62925000	Hà Toàn	KH00126
6643	2021-12-31	'0001100	TL_VINADS	Thuốc lá Vinataba Demi Slims	Kiện	1	3470000.0	3470000	3457685.98	3457686	Hà Toàn	KH00126
6644	2021-12-31	'0001100	D_TN	Diêm Thống Nhất	Kiện	1	356772.0	356772	354339.00	354339	Hà Toàn	KH00126

6645 rows × 12 columns

In [10]:
```
def Format_SCT(df):
    df['Ma_KH'] = df['Ma_KH'].fillna('null')
    columns_to_drop = ['Diễn giải chung','ĐVT','Tên hàng','Tên khách hàng']
    # Check if columns exist before attempting to drop them
    columns_to_drop_existing = [col for col in columns_to_drop if col in df.columns]
    if columns_to_drop_existing:
        df.drop(columns=columns_to_drop_existing, inplace=True)
    return df
```

In [11]:
```
SCT_20=Format_SCT(SCT_20)
SCT_21=Format_SCT(SCT_21)
SCT_22=Format_SCT(SCT_22)
```

B2 ĐƯA DỮ LIỆU VÀO CSDL

In [3]:
```
# Kết nối với CSDL
conn = pyodbc.connect(
Trusted_Connection = "Yes",
Driver = '{ODBC Driver 17 for SQL Server}',
Server = "DESKTOP-MDDIIDJ\MSSQLSERVER01",
Database = 'Ban_Hang')
cursor = conn.cursor()
```

In [12]:
```
#Tạo bảng

cursor.execute("CREATE TABLE SO_CHI_TIET(Ngay_chung_tu date, So_hoa_don varchar(10), Ma_hang varchar(20), SL_
```

Out[12]:
```
<pyodbc.Cursor at 0x28c7c656db0>
```

In [13]:
```
def Insert_SCT(df):
    for row in df.itertuples(index=False):
        so_hoa_don = row[1][1:]
        cursor.execute('''
                        INSERT INTO BAN_HANG.dbo.SO_CHI_TIET(Ngay_chung_tu, So_hoa_don, Ma_hang, SL_ban, Don_
                        VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?)
                        ''',
                       row[0], so_hoa_don, row[2], row[3], row[4], row[5], row[6], row[7], row[8]
                      )
    conn.commit()
```

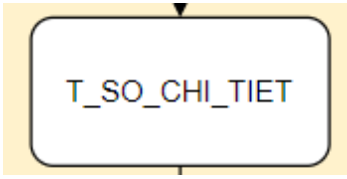
In [14]:
```
#thêm 2020
Insert_SCT(SCT_20)
```

In [15]:
```
#thêm 2021
Insert_SCT(SCT_21)
```

In [16]:
```
#lấy ra mã KH có trong CSDL để thực hiện nối
#query = "SELECT * FROM KHACH_HANG"
#KH = pd.read_sql_query(query, conn)
```

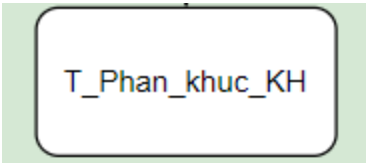
In [17]:
```
#SCT_22 = merge_SCT(SCT22,KH)
#SCT_22=Format_SCT(SCT_22)
##thêm 2022
Insert_SCT(SCT_22)
```

In [ ]:



@3

GHI CHÚ



I) File này dùng để tính toán các chỉ số RFM của khách hàng và tiến hành phân cụm. Sau đó đưa vào bảng Phan_khuc_KH trong CSDL Ban_HANG

II) Sử dụng dữ liệu từ bảng SO_CHI_TIET trong CSDL Ban_Hang

In [4]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pyodbc
import datetime
from scipy import stats
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from matplotlib.ticker import FuncFormatter
import numpy as np
import pyodbc
```

B1 tính RFM

In [3]:
```
def cal_RFM():
    pd.options.display.float_format = '{:.2f}'.format
    #Lay ra SO_CHI_TIET
    query = "SELECT * FROM SO_CHI_TIET where Ma_KH not in ('Sampling','Tu_thien')"
    df = pd.read_sql_query(query, conn)
    #Giả sử ngày hiện tại là ngày phát sinh giao dịch cuối cùng + 1 trong SO_CHI_TIET
    current_date = max(df['Ngay_chung_tu']) + datetime.timedelta(days=1)
    #Tính RFM
    df_customers = df.groupby(['Ma_KH']).agg(
    {'Ngay_chung_tu': lambda x: (current_date- x.max()).days,
     'So_hoa_don':'nunique',
     'Doanh_so':'sum'
     }
    )
    df_customers.rename(columns={'Ngay_chung_tu':'Recency','So_hoa_don':'Frequency','Doanh_so':'MonetaryValue
    return df_customers
```

In [4]:
```
df_customers = cal_RFM()
```

In [5]:
```
df_customers
```

Out[5]:

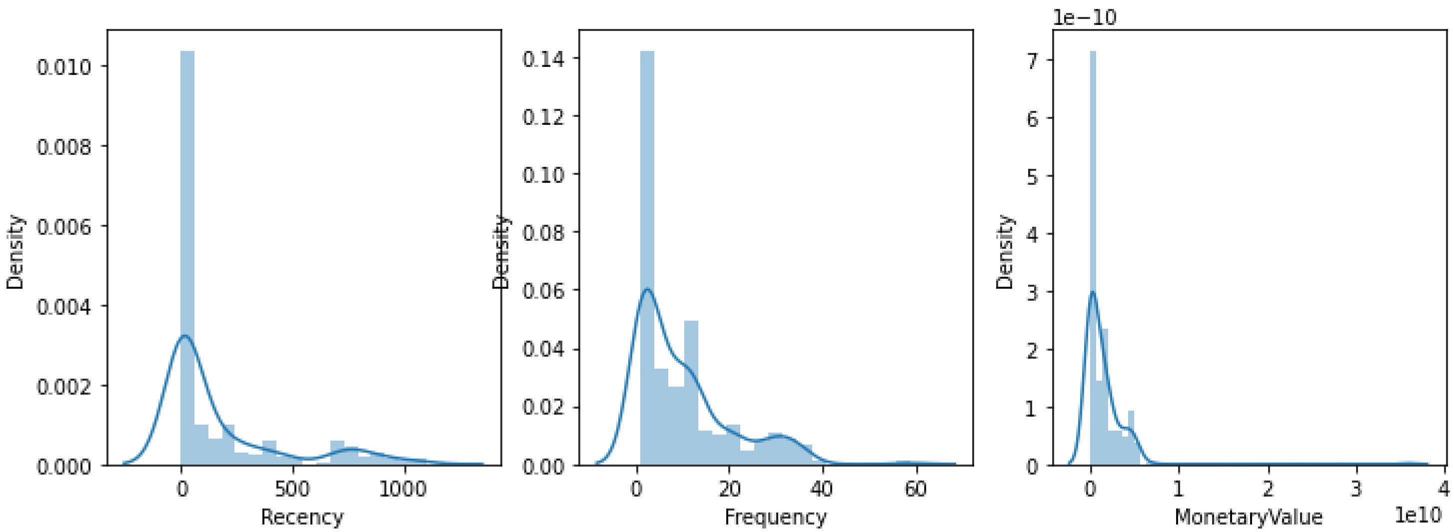
Ma_KH	Recency	Frequency	MonetaryValue
KH00001	1095	1	164320000.00
KH00002	858	2	96175000.00
KH00003	10	31	4363511324.00
KH00004	7	35	3945543723.00
KH00005	6	35	6895584252.00
...	...	...	...
KH00329	1	1	13204854.00
KH00330	1	1	10303904.00
KH00331	1	1	14016860.00
KH00332	1	1	10655000.00
KH00333	1	1	10900201.00

332 rows × 3 columns

B2: Vẽ phân phối để lựa chọn phương pháp Transform

In [6]:
```
# Ve phan phoi de kiem tra
fig, ax = plt.subplots(1, 3, figsize=(12,4))
sns.distplot(df_customers['Recency'], ax = ax[0])
sns.distplot(df_customers['Frequency'], ax = ax[1])
sns.distplot(df_customers['MonetaryValue'], ax = ax[2])
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

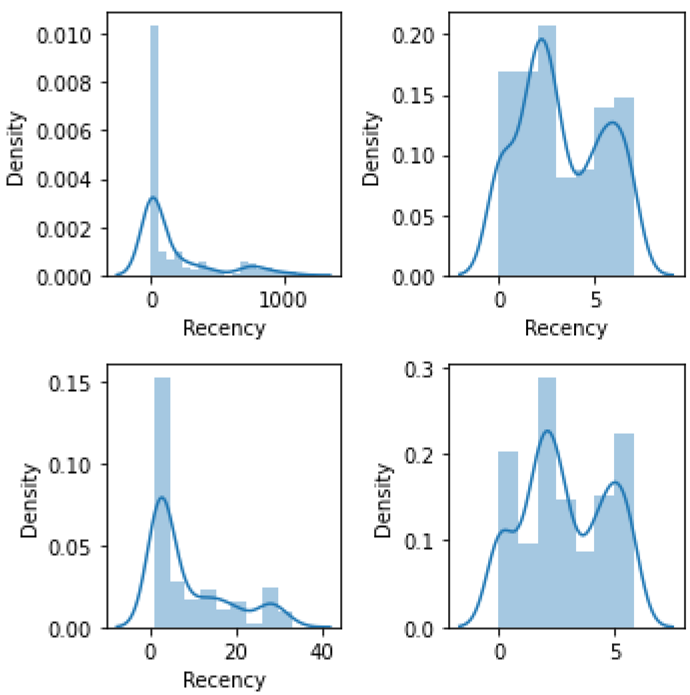


B3: phân tích skewness để lựa chọn phương pháp transform phù hợp với dữ liệu

In [7]:
```
def analyze_skewness(x):
    fig, ax = plt.subplots(2, 2, figsize=(5,5))
    sns.distplot(df_customers[x], ax=ax[0,0])
    sns.distplot(np.log(df_customers[x]), ax=ax[0,1])
    sns.distplot(np.sqrt(df_customers[x]), ax=ax[1,0])
    sns.distplot(stats.boxcox(df_customers[x])[0], ax=ax[1,1])
    plt.tight_layout()
    plt.show()

    print(df_customers[x].skew().round(2))
    print(np.log(df_customers[x]).skew().round(2))
    print(np.sqrt(df_customers[x]).skew().round(2))
    print(pd.Series(stats.boxcox(df_customers[x])[0]).skew().round(2))
```

In [8]:
```
analyze_skewness('Recency')
```

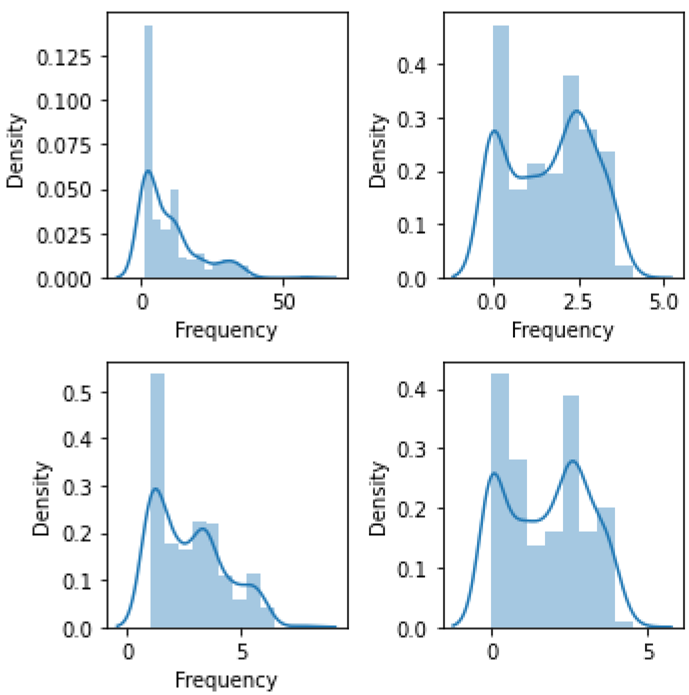
```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



```
1.82
0.19
1.18
0.05
```

In [9]:
```
analyze_skewness('Frequency')
```

```
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
D:\anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated funct
ion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



```
1.4
-0.09
0.6
-0.03
```

Lựa chọn Transform

In [10]:
```
#Tiến hành transform dữ liệu
df_customers_t = pd.DataFrame()
df_customers_t['Recency'] = stats.boxcox(df_customers['Recency'])[0]
df_customers_t['Frequency'] = stats.boxcox(df_customers['Frequency'])[0]
df_customers_t['MonetaryValue'] = pd.Series(np.cbrt(df_customers['MonetaryValue'])).values
df_customers_t.head(10)
```

Out[10]:

	Recency	Frequency	MonetaryValue
0	5.84	0.00	547.73
1	5.67	0.71	458.16
2	2.17	3.76	1634.10
3	1.85	3.91	1580.16
4	1.71	3.91	1903.37
5	4.45	3.24	1665.02
6	1.85	3.41	1691.70
7	1.71	3.94	1436.44
8	1.85	3.46	1704.49
9	4.12	3.30	1649.84

B4: Scale dữ liệu

In [11]:
```
# Tien hanh scale du lieu
scaler = StandardScaler()
scaler.fit(df_customers_t)
df_customers_t = scaler.transform(df_customers_t)
```

In [12]:
```
pd.DataFrame(df_customers_t).head()
```

Out[12]:

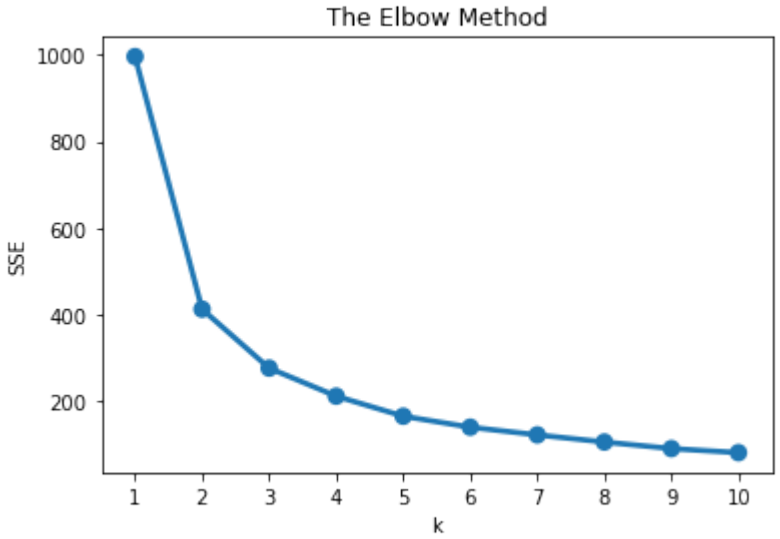
	0	1	2
0	1.63	-1.35	-0.60
1	1.54	-0.81	-0.77
2	-0.38	1.51	1.44
3	-0.56	1.63	1.34
4	-0.64	1.63	1.94

B5: tìm số tâm cụm k tối ưu theo phương pháp elbow

In [13]:
```
# Chọn số cụm bằng Elbow
def elbow(df):
    sse = {}
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(df)
        sse[k] = kmeans.inertia_

    plt.title('The Elbow Method')
    plt.xlabel('k')
    plt.ylabel('SSE')
    sns.pointplot(x=list(sse.keys()), y=list(sse.values()))
    plt.show()
```

In [14]:
```
# sử dụng đầu vào vào df đã transform
elbow(df_customers_t)
```

```
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```



B6: khởi chạy thuật toán kmeans

In [15]:
```
# đầu vào là k số tâm cụm, df dã transform, df đã tính RFM
def get_labels(k,df_t,df):
    model = KMeans(n_clusters=k, random_state=42)
    model.fit(df_t)
    df['Cluster'] = model.labels_
    return df
```

In [16]:
```
df = get_labels(3,df_customers_t,df_customers)
```

```
D:\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1334: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=2.
  warnings.warn(
```

B7: reset index và đổi các giá trị cụm cho phù hợp

In [17]:
```
df.groupby('Cluster').agg(
    {
        'Recency':'mean',
        'Frequency':'mean',
        'MonetaryValue':'mean'
    }
).round(2)
```

Out[17]:

Cluster	Recency	Frequency	MonetaryValue
0	434.11	2.17	156843796.77
1	7.86	2.79	197947207.05
2	32.79	17.39	2757230575.57

In [18]:
```
df = df.reset_index()
df['Cluster'].replace({2: 'A', 1: 'B', 0 : 'C'}, inplace=True)
```

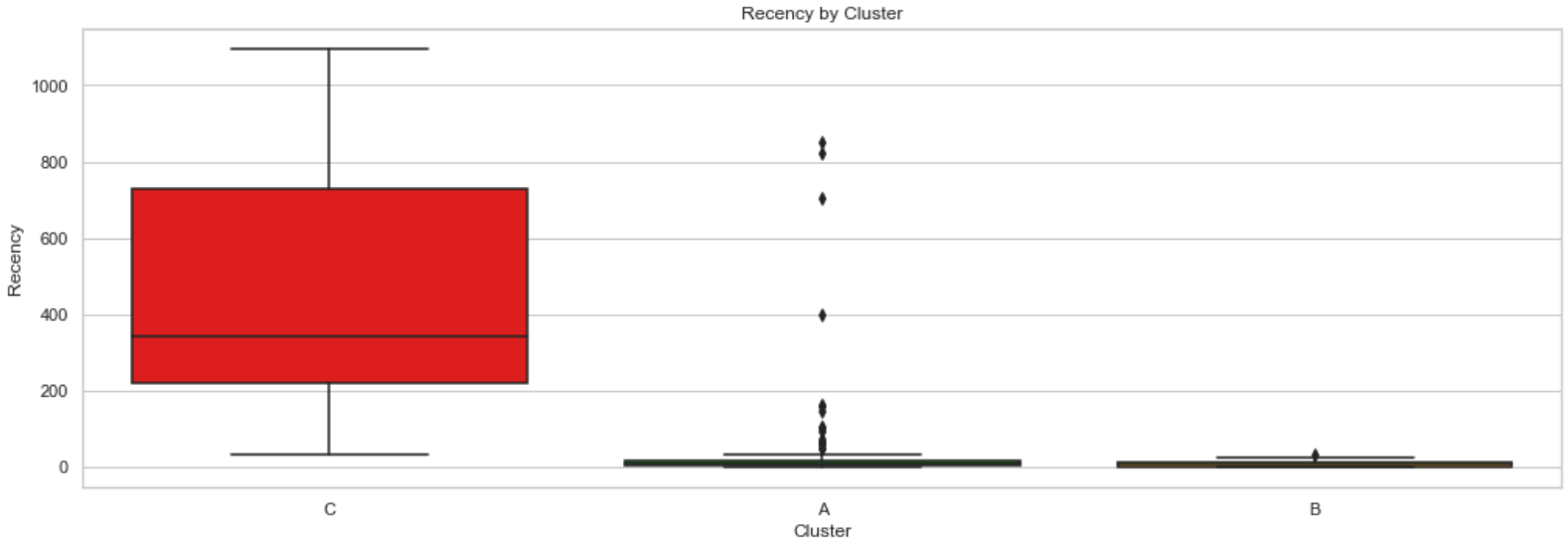
In [19]:
```
df
```

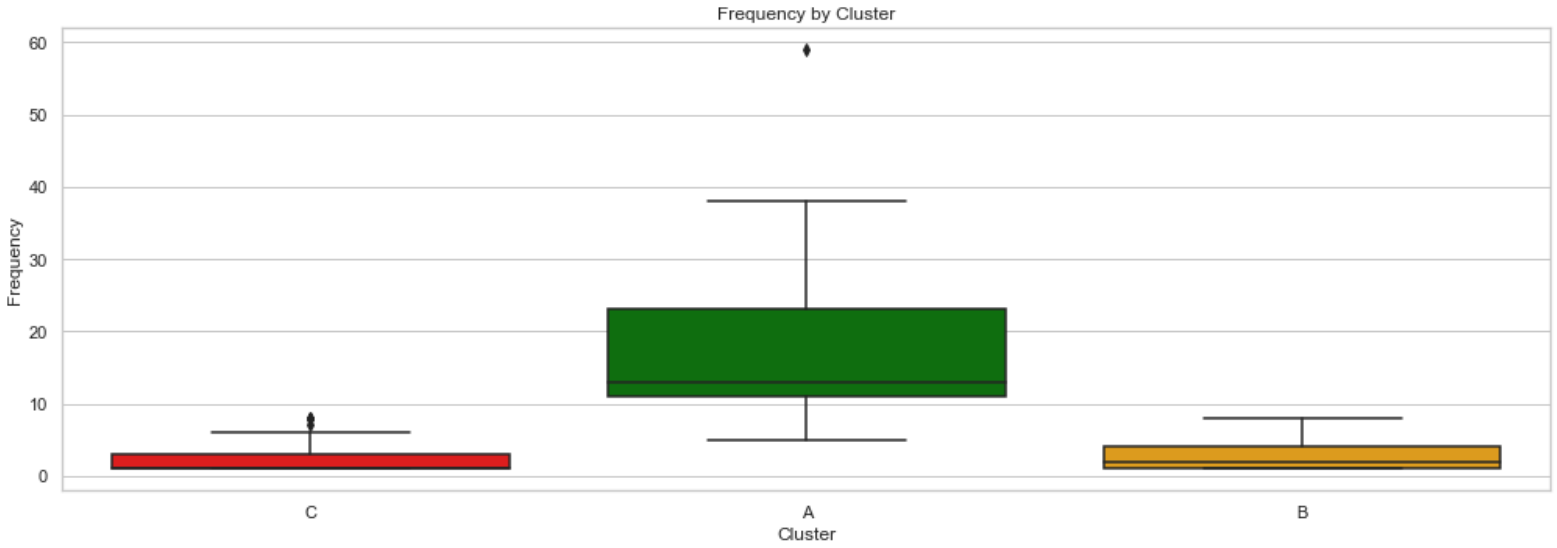
Out[19]:

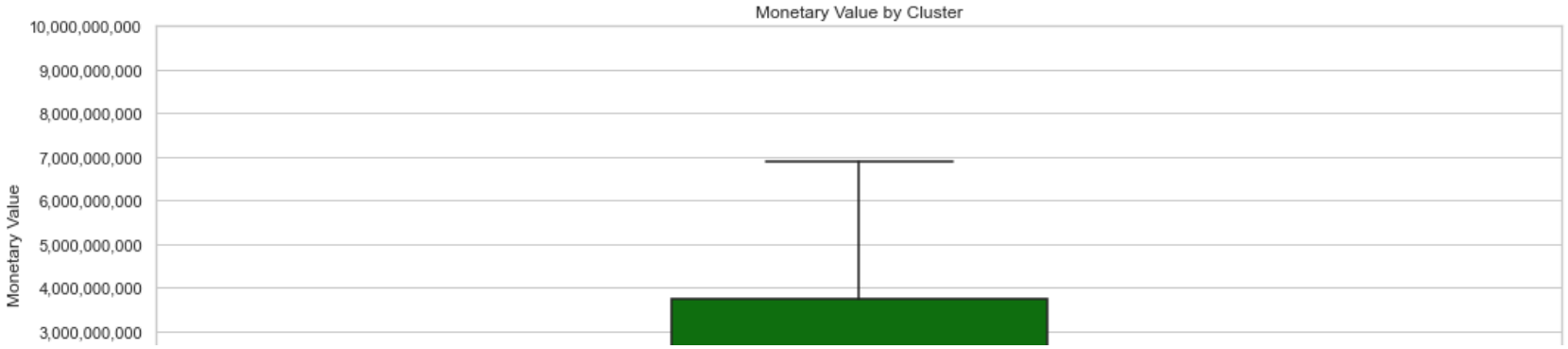
	Ma_KH	Recency	Frequency	MonetaryValue	Cluster
0	KH00001	1095	1	164320000.00	C
1	KH00002	858	2	96175000.00	C
2	KH00003	10	31	4363511324.00	A
3	KH00004	7	35	3945543723.00	A
4	KH00005	6	35	6895584252.00	A
...	...	...	...	...	...
327	KH00329	1	1	13204854.00	B
328	KH00330	1	1	10303904.00	B
329	KH00331	1	1	14016860.00	B
330	KH00332	1	1	10655000.00	B
331	KH00333	1	1	10900201.00	B

332 rows × 5 columns

In [68]:
```
df['Recency'] = df['Recency'].apply(lambda x: max(x, 0))
df['Frequency'] = df['Frequency'].apply(lambda x: max(x, 0))
df['MonetaryValue'] = df['MonetaryValue'].apply(lambda x: max(x, 0))

# Định dạng tiền tệ tùy chỉnh
def currency_format(x, pos):
    return '{:,.0f}'.format(x)

# Tạo biểu đồ boxplot cho từng chỉ số Recency, Frequency, MonetaryValue theo cluster
plt.figure(figsize=(15, 15))

# Biểu đồ boxplot cho Recency
plt.subplot(3, 1, 1)
sns.boxplot(x='Cluster', y='Recency', data=df, palette={'A': 'green', 'B': 'orange', 'C': 'red'})
plt.title('Recency by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Recency')

# Biểu đồ boxplot cho Frequency
plt.subplot(3, 1, 2)
sns.boxplot(x='Cluster', y='Frequency', data=df, palette={'A': 'green', 'B': 'orange', 'C': 'red'})
plt.title('Frequency by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Frequency')

# Biểu đồ boxplot cho MonetaryValue
plt.subplot(3, 1, 3)
sns.boxplot(x='Cluster', y='MonetaryValue', data=df, palette={'A': 'green', 'B': 'orange', 'C': 'red'})
plt.title('Monetary Value by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Monetary Value')
ax = plt.gca()
ax.yaxis.set_major_formatter(FuncFormatter(currency_format))
ax.set_ylim(0, 10**10)  # Đặt giới hạn trục y từ 0 đến 10 tỷ
ax.set_yticks(range(0, 10**10 + 1, 10**9))  # Đặt các bước nhảy trên trục y mỗi 1 tỷ

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


Recency by Cluster
Recency
1000
800
600
400
200
0
C
A
B
Cluster


Frequency by Cluster
Frequency
60
50
40
30
20
10
0
C
A
B
Cluster


Monetary Value by Cluster
Monetary Value
10,000,000,000
9,000,000,000
8,000,000,000
7,000,000,000
6,000,000,000
5,000,000,000
4,000,000,000
3,000,000,000

B8 ĐƯA DỮ LIỆU VÀO CSDL

In [5]:
```
# Kết nối với CSDL
conn = pyodbc.connect(
Trusted_Connection = "Yes",
Driver = '{ODBC Driver 17 for SQL Server}',
Server = "DESKTOP-MDDIIDJ\MSSQLSERVER01",
Database = 'Ban_Hang')
cursor = conn.cursor()
```

In [2]:
```
#Tạo bảng
#cursor.execute("CREATE TABLE PHAN_KHUC_KH(Ma_KH varchar(20),  Recency int, Frequency int, MonetaryValue floa
```

In [21]:
```
def Insert_PKKH(df):
    # Xóa toàn bộ dữ liệu từ bảng PHAN_KHUC_KH
    cursor.execute('''
                   DELETE FROM BAN_HANG.dbo.PHAN_KHUC_KH
                   ''')
    # Chèn dữ liệu vào bảng PHAN_KHUC_KH
    for row in df.itertuples(index=False):
        cursor.execute('''
                       INSERT INTO BAN_HANG.dbo.PHAN_KHUC_KH(Ma_KH, Recency, Frequency, MonetaryValue, Phan_
                       VALUES(?, ?, ?, ?, ?)
                       ''',
                      row[0], row[1], row[2], row[3], row[4]
                      )
    conn.commit()
```

In [22]:
```
#Insert_PKKH(df)
```

Tóm tắt các bước thực hiện

Các bước thực hiện:
B1: dùng cal_RFM() để tính RFM
ex: df_customers = cal_RFM()
B2: Kiểm tra phân phối của dữ liệu
B3: Phân tích skewness để lựa chọn pp Transform
ex: df_customers_t['Recency'] = stats.boxcox(df_customers['Recency'])[0]
B4: Scale dữ liệu để đưa vào mô hình phân cụm Kmeans
ex: scaler.fit(df_customers_t)
B5: lựa chọn số tâm k tối ưu theo phương pháp elbow
ex: elbow(df_customers_t)
B6: sử dụng hàm get_labels(k,df_t,df_customers) để khởi chạy thuận toán Kmeans, đầu vào là số tâm k tối ưu,dataframe đã transform, và dataframe tính RFM ban đầu (ở B1)
ex: df = get_labels(3,df_customers_t,df_customers)
B7: Reset index và đổi các giá trị tâm cụm
B8: đưa dữ liệu vào sql bằng hàm Insert_PKKH()
ex: Insert_PKKH(df)