**Học Viện VTI**

**DAS 2406 HCM**

**BÁO CÁO ĐỒ ÁN CUỐI KHÓA**

*ĐỀ TÀI***: PHÂN TÍCH TÌNH HÌNH KINH DOANH NHẠC SỐ**

**Hướng dẫn viên: GV. Võ Thanh Phong Nhóm thực hiện:**

**Nguyễn Ngọc Xuân Anh**

**Phạm Quốc Huy**

**Nguyễn Kim Hiếu**

**Thành phố Hồ Chí Minh, tháng 01 năm 2025**

# **MỤC LỤC**

[**BẢNG PHÂN CÔNG CÔNG VIỆC** 3](#_Toc187786974)

[**1. Tổng quan đồ án** 4](#_Toc187786975)

[1.1. Bài toán: 4](#_Toc187786976)

[1.2. Quá trình thực hiện đồ án: 4](#_Toc187786977)

[**2. Phân tích chi tiết** 5](#_Toc187786978)

[2.1. Bài toán 1: 5](#_Toc187786979)

[2.1.1 Mô tả dữ liệu: 5](#_Toc187786980)

[2.1.2. Tiền xử lý dữ liệu: 8](#_Toc187786981)

[2.1.3. Phân loại khách hàng dựa trên bảng RFM Score: 8](#_Toc187786982)

[2.1.4. Trực quan hóa dữ liệu 10](#_Toc187786983)

[2.1.5 Kết luận 14](#_Toc187786984)

[2.2. Bài toán 2 14](#_Toc187786985)

[2.2.1. Mô tả dữ liệu 14](#_Toc187786986)

[2.2.2. Mô tả các biến 15](#_Toc187786987)

[2.2.3. Tiền xử lý dữ liệu: 15](#_Toc187786988)

[2.2.4. Tiến hành hành bài toán dự đoán: 16](#_Toc187786989)

[2.2.5. Kết luận 20](#_Toc187786990)

[**3. Các file đính kèm** 21](#_Toc187786991)

[3.1 Raw data: 21](#_Toc187786992)

[3.2 Visualize data bằng công cụ Power BI: 21](#_Toc187786993)

[3.3 Data xử lý bài toán dự đoán: 21](#_Toc187786994)

[3.4 Full source code machine learing xử lý bài toán dự đoán: 21](#_Toc187786995)

# **BẢNG PHÂN CÔNG CÔNG VIỆC**

| **Thành viên** | **Công việc** |
| --- | --- |
| Nguyễn Ngọc Xuân Anh | * Phát biểu bài toán, đặt ra câu hỏi từ đó đặt ra các yêu cầu để thu thập dữ liệu. * Thu thập dữ liệu mua hàng từ link Github [Chinook database](https://github.com/lerocha/chinook-database). * Clean data. * Thực hiện slide thuyết trình. |
| Phạm Quốc Huy | * Phát biểu bài toán, đặt ra câu hỏi từ đó đặt ra các yêu cầu để thu thập dữ liệu. * Visualize các dữ liệu bằng công cụ **Power BI** * Training với mô hình **Decision Tree Classifier** và đưa ra nhận xét. |
| Nguyễn Kim Hiếu | * Phát biểu bài toán, đặt ra câu hỏi từ đó đặt ra các yêu cầu để thu thập dữ liệu. * Clean data. * Phân loại khách hàng dựa trên **RFM score** bằng thuật toán phân cụm **Kmean**. |

# **1. Tổng quan đồ án**

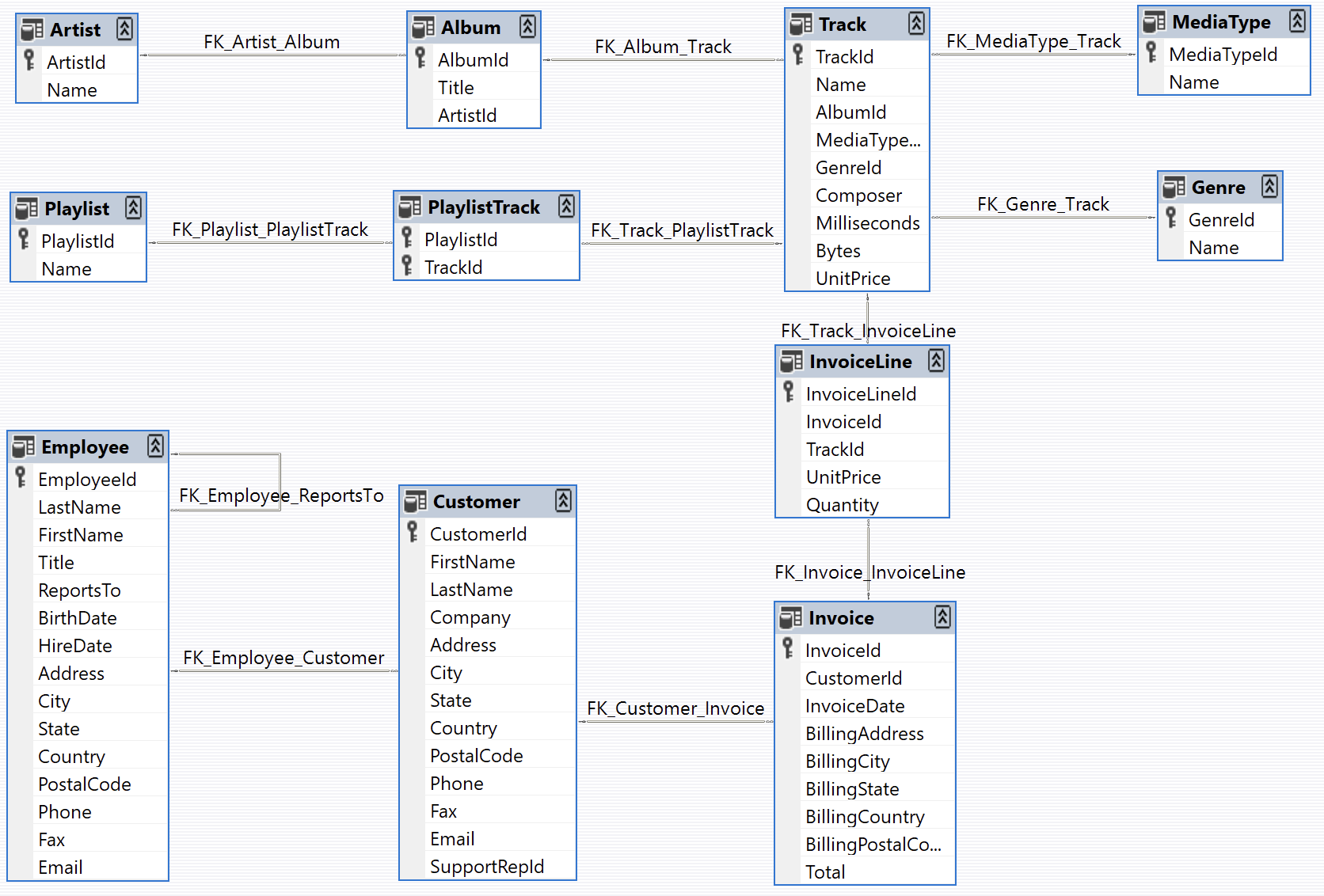
## 1.1. Bài toán:

1. Phân tích tình hình kinh doanh nhạc số theo Doanh thu, Sản phẩm, Khách hàng và Nhân viên trong 4 năm trở lại đây.
2. Dự đoán khoảng bao lâu thì khách hàng sẽ quay lại mua hàng.

## ****1.2. Quá trình thực hiện đồ án****:

**- Giai đoạn 1**: Nhóm giải quyết bài toán **phân tích tình hình kinh doanh nhạc số trong 4 năm trở lại đây; nhóm lấy code slq từ** [Chinook database](https://github.com/lerocha/chinook-database) trên github, sau đó dùng công cụ **Sql Server Management Studio** để tạo database **DB\_Music** trên database local.

* **Cấu trúc dữ liệu:**



* **Vấn đề**: Dữ liệu còn thiếu khá nhiều, format ngày tháng năm chưa đồng nhất, kiểu dữ liệu là số nhưng hiện là kiểu chữ. Vì thế phải làm sạch dữ liệu, định dạng lại ngày tháng năm, điền những dữ liệu trống bằng chuỗi “Unknow”.

**- Giai đoạn 2**: Nhóm cùng nhau đặt ra các câu hỏi để từ đó tìm ra insight có liên quan để phân tích tình hình kinh doanh nhạc số trong 4 năm trở lại đây. Sau đó trực quan hóa dữ liệu bằng công cụ Power BI.

- **Giai đoạn 3**: Nhóm tiến hành giải quyết bài toán dự đoánbao lâu thì khách hàng sẽ quay lại mua hàng bằng thuật toán Decision Tree Classifier trên công cụ Visual Studio Code.

# **2. Phân tích chi tiết**

## ****2.1. Bài toán 1:****

### **2.1.1 Mô tả dữ liệu:**

**Track:**

* TrackId: Mã số bài hát
* Name: Tên bài hát
* AlbumId: Mã số Album
* MediaTypeId: Mã số loại phương tiện truyền thông
* GenreId: Mã dòng nhạc
* Composer: Đơn vị biên soạn
* Milliseconds: số giây bài hát
* Bytes: Dung lượng bài hát
* UnitPrice: Giá bài hát

**MediaType:**

* MediaTypeId: Mã loại phương tiện truyền thông.
* MediaType\_Name: Tên loại phương tiện truyền thông.

**Genre:**

* GenreId: Mã dòng nhạc
* Name: Tên dòng nhạc

**Artist:**

* ArtistId: Mã nghệ sĩ
* Name: Tên nghệ sĩ

**Album:**

* AlbumId: Mã album
* Title: Tên album
* ArtistId: Mã nghệ sĩ

**Playlist**:

* PlaylistId: Mã danh sách phát.
* Playlist\_Name: Tên danh sách phát.

**PlaylistTrack:**

* PlaylistId: Mã danh sách phát
* TrackId: Mã bài hát.

**Invoice:**

* InvoiceId: Mã hóa đơn
* CustomerId: Mã khách hàng mua hóa đơn
* InvoiceDate: Ngày mua hàng.
* BillingAddress: Địa chỉ mua hàng.
* BillingCity: Thành phố nơi mua hàng.
* BillingState: Bang nơi mua hàng.
* BillingCountry: Quốc gia nơi mua hàng
* BillingPostalCode: Mã quốc gia nơi mua hàng
* Total: tổng tiền hóa đơn.

**InvoiceLine:**

* InvoiceLineId: Mã dòng hóa đơn (có nhiều dòng hóa đơn trong 1 hóa đơn mua hàng)
* InvoiceId: Mã hóa đơn
* TrackId: Mã bài hát
* UnitPrice: Giá bán bài hát
* Quantity: số lượng mua bài hát.

**Customer:**

* CustomerId: Mã khách hàng.
* FirstName: Tên khách hàng.
* LastName: Họ khách hàng.
* Company: Công ty nơi khác hàng làm việc.
* Address: địa chỉ khách hàng.
* City: Thành phố nơi khách hàng ở.
* State: Bang nơi khách hàng ở.
* Country: Quốc gia nơi khách hàng ở.
* PostalCode: Mã quốc gia của khách hàng.
* Phone: số điện thoại khách hàng.
* Fax: mã fax khách hàng.
* Email: email khách hàng.
* SupportRepId: Mã nhân viên đã hỗ trợ khách hàng.

**Employee:**

* EmployeeId: Mã nhân viên
* LastName: Họ nhân viên
* FirstName: Tên nhân viên
* Title: Chức vụ
* ReportsTo: Mã nhân viên của người quản lý trực tiếp mà nhân viên báo cáo công việc.
* BirthDate: Ngày sinh nhân viên.
* HireDate: Ngày vào làm nhân viên.
* Address: Địa chỉ nhân viên.
* City: Thành phố của nhân viên.
* State: Bang của nhân viên.
* Country: Quốc gia của nhân viên.
* PostalCode: Mã quốc gia của nhân viên.
* Phone: Số điện thoại của nhân viên.
* Fax: Fax của nhân viên.
* Email: email của nhân viên.

### **2.1.2. Tiền xử lý dữ liệu:**

Sử dụng công cụ Microsoft Excel để xử lý: định dạng lại các cột ngày tháng thành 1 định dạng chung là yyyy-MM-dd, định dạng các cột vừa có dữ liệu dạng chữ và dạng số thành 1 dạng chung là kiểu dữ liệu chuỗi (text), điền những dòng dữ liệu trống bằng chuỗi “Unkow”.

Hình ảnh ví dụ:



Dữ liệu được lưu thành file **Music\_Project.xlsx**



### **2.1.3. Phân loại khách hàng dựa trên bảng RFM Score:**

**Tìm Recency khách hàng:**

tx\_max\_purchase = tx\_data.groupby('CustomerId').InvoiceDate.max().reset\_index()

tx\_max\_purchase.columns = ['CustomerId','MaxPurchaseDate']

tx\_max\_purchase.info()

tx\_max\_purchase['MaxPurchaseDate'] = pd.to\_datetime(tx\_max\_purchase['MaxPurchaseDate'])

tx\_max\_purchase['Recency'] = (tx\_max\_purchase['MaxPurchaseDate'].max() - tx\_max\_purchase['MaxPurchaseDate']).dt.days

**Phân cụm Recency thành điểm bằng thuật toán Kmeans:**

kmeans = KMeans(n\_clusters=5).fit(tx\_recency[['Recency']])

tx\_user['RecencyCluster'] = kmeans.predict(tx\_recency[['Recency']])

**Tìm Frequency khách hàng:**

tx\_frequency = tx\_data.groupby('CustomerId').InvoiceDate.count().reset\_index()

tx\_frequency.columns = ['CustomerId','Frequency']

**Phân cụm Frequency thành điểm bằng thuật toán Kmeans**

kmeans = KMeans(n\_clusters=5).fit(tx\_frequency[['Frequency']])

tx\_frequency['FrequencyCluster'] = kmeans.predict(tx\_frequency[['Frequency']])

**Tìm Monetary khách hàng:**

tx\_data['Revenue'] = tx\_data['Total']

tx\_revenue = tx\_data.groupby('CustomerId').Revenue.sum().reset\_index()

**Phân cụm Monetary thành điểm bằng thuật toán Kmeans:**

kmeans = KMeans(n\_clusters=5).fit(tx\_revenue[['Monetary']])

tx\_revenue['MonetaryCluster'] = kmeans.predict(tx\_revenue[['Monetary']])

**Gộp các điểm cụm thành RFM score**

tx\_user\_rfm['RFM\_Score'] = tx\_user\_rfm.RecencyClusterSort.astype(str) + tx\_user\_rfm.FrequencyClusterSort.astype(str) + tx\_user\_rfm.MonetaryClusterSort.astype(str)

**Định nghĩa Score sang Segment và mapping từ RFM Score để phân loại khách hàng:**

score\_to\_segment = {

    '01\_Champions': [555, 554, 544, 545, 454, 455, 445],

    '02\_Loyal': [543, 444, 435, 355, 354, 345, 344, 335],

    '03\_Potential Loyalist': [553, 551, 552, 541, 542, 533, 532, 531, 452, 451, 442, 441, 431, 453, 433, 432, 423, 353, 352, 351, 342, 341, 333, 323],

    '04\_New Customers': [512, 511, 422, 421, 412, 411, 311],

    '05\_Promising': [525, 524, 523, 522, 521, 515, 514, 513, 425, 424, 413, 414, 415, 315, 314, 313],

    '06\_Need Attention': [535, 534, 443, 434, 343, 334, 325, 324],

    '07\_About To Sleep': [331, 321, 312, 221, 213, 231, 241, 251],

    '08\_At Risk': [255, 254, 245, 244, 253, 252, 243, 242, 235, 234, 225, 224, 153, 152, 145, 143, 142, 135, 134, 133, 125, 124],

    '09\_Cannot Lose Them': [155, 154, 144, 214, 215, 115, 114, 113],

    '10\_Hibernating customers': [332, 322, 231, 241, 251, 233, 232, 223, 222, 132, 123, 122, 212, 211],

    '11\_Lost customers': [111, 112, 121, 131, 141, 151]

}

# Đảo ngược mapping để dễ tra cứu

mapping = {score: segment for segment, scores in score\_to\_segment.items() for score in scores}

# Map RFM\_Score sang Segment

tx\_user\_rfm['Segment'] = tx\_user\_rfm['RFM\_Score'].astype(int).map(mapping)

grouped = tx\_user\_rfm.groupby('Segment').agg(Count=('CustomerId', 'count')

, Avg\_Recency=('Recency', 'mean')

, Avg\_Frequency=('Frequency', 'mean')

 , Avg\_Monetary=('Monetary', 'mean')

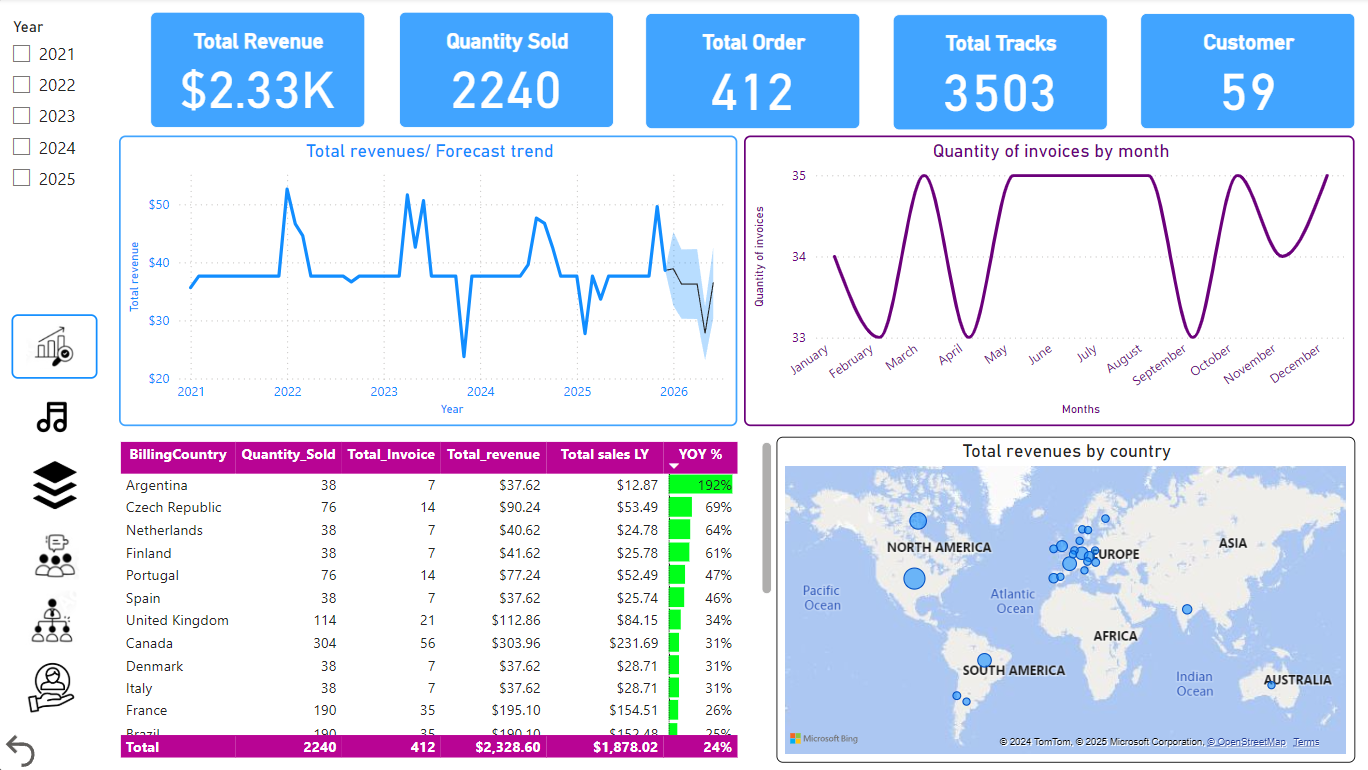
)

**Kết quả được lưu vào file CustomerSegment.csv**

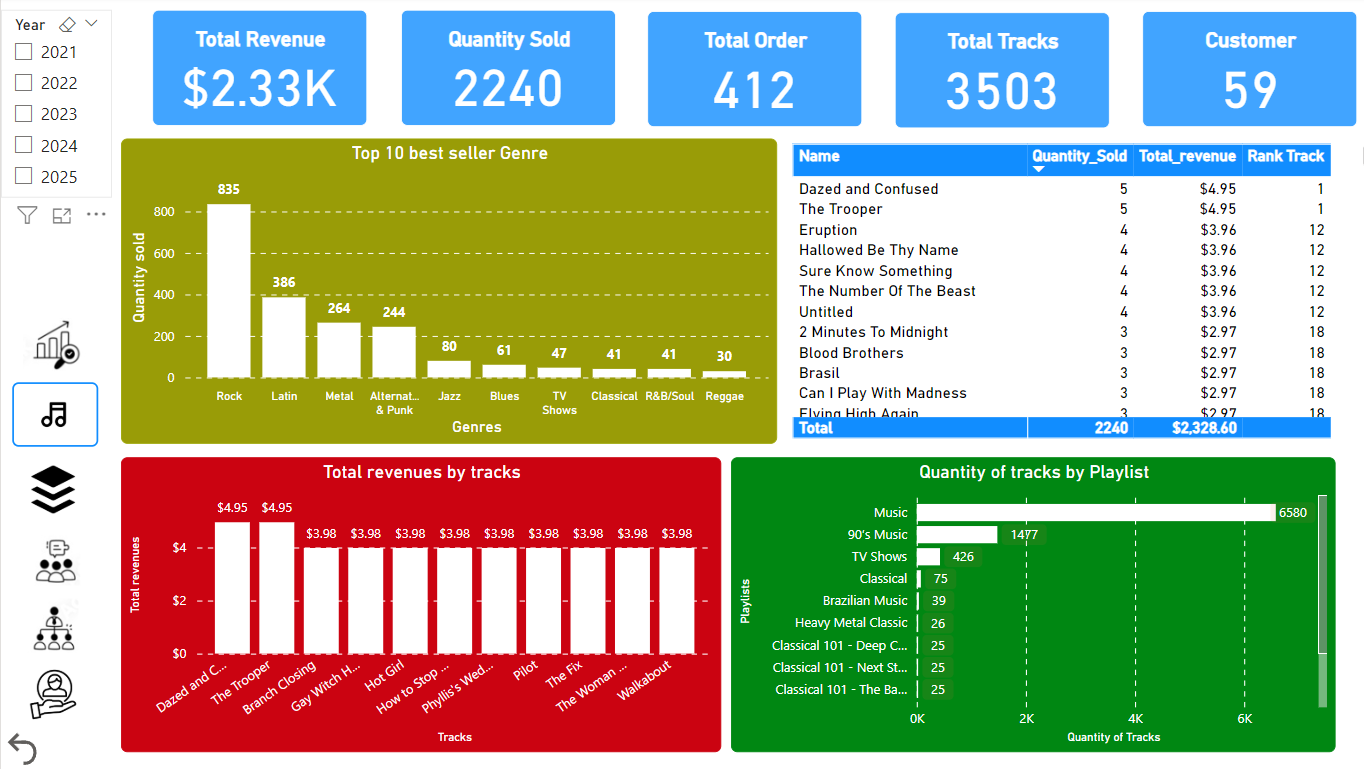


### **2.1.4. Trực quan hóa dữ liệu**

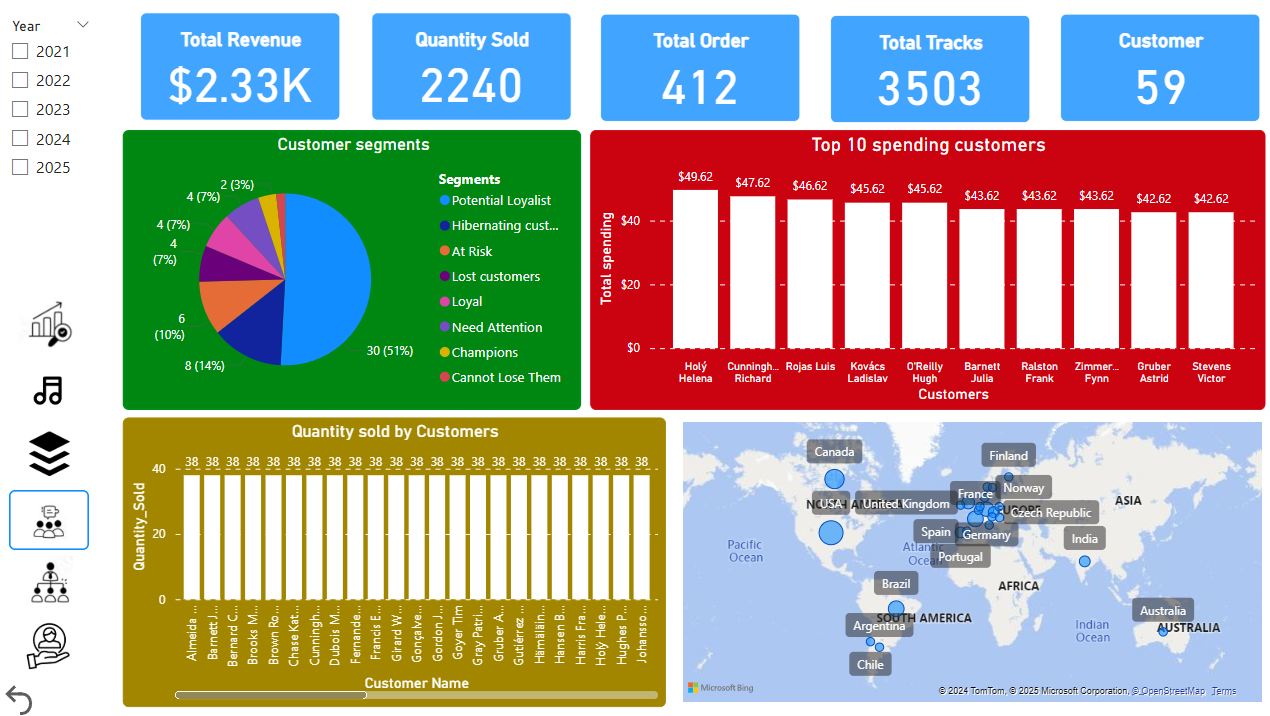
Tình hình doanh thu theo thời gian:



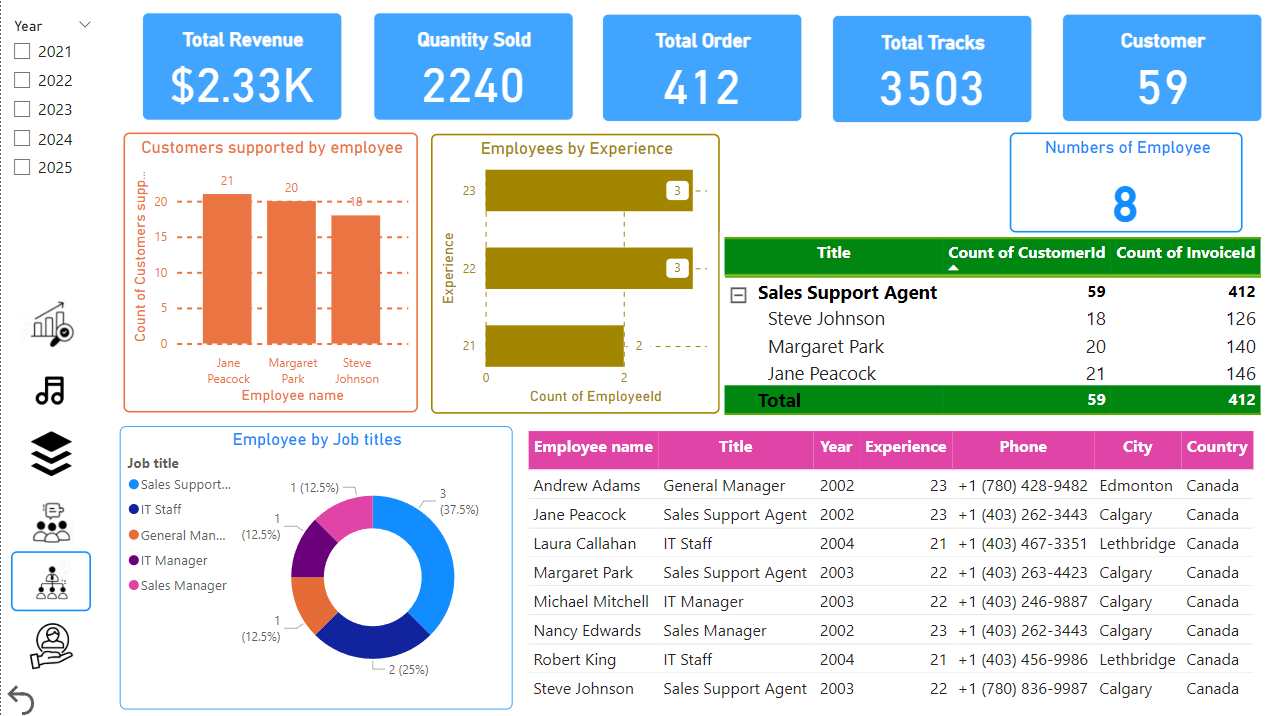
Tình hình doanh thu theo bài hát:



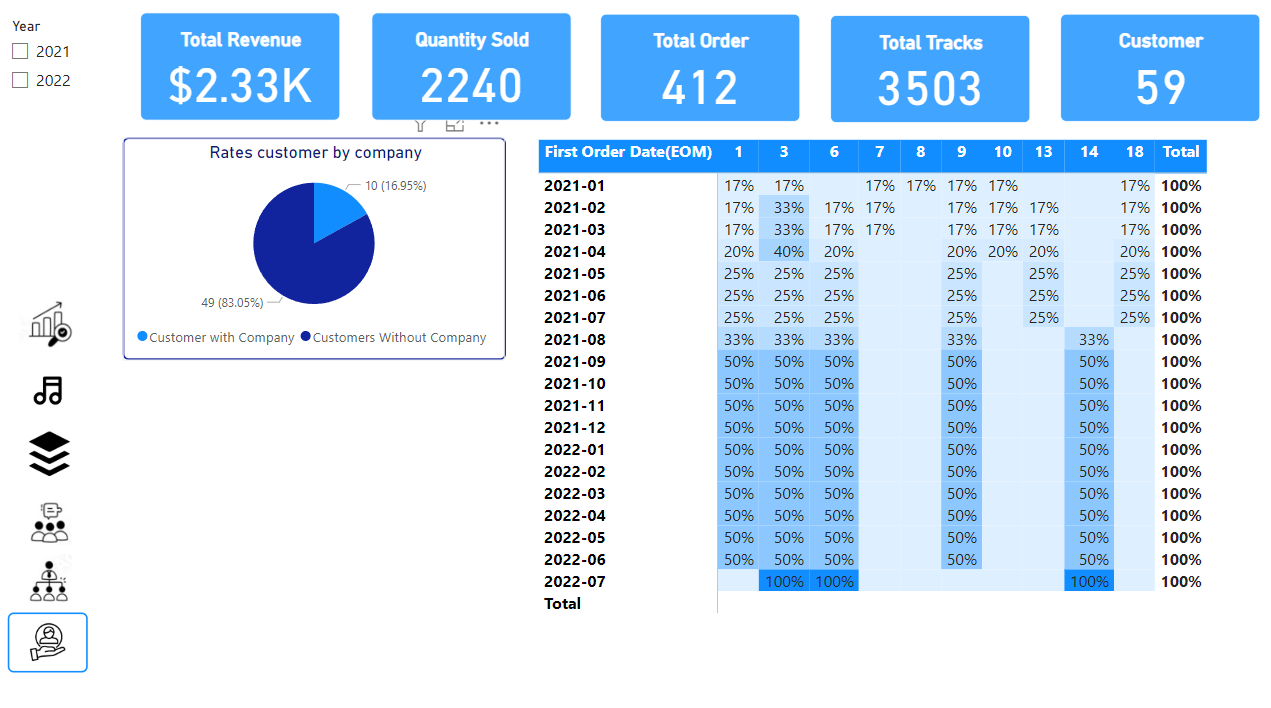
Tình hình doanh thu theo khách hàng:



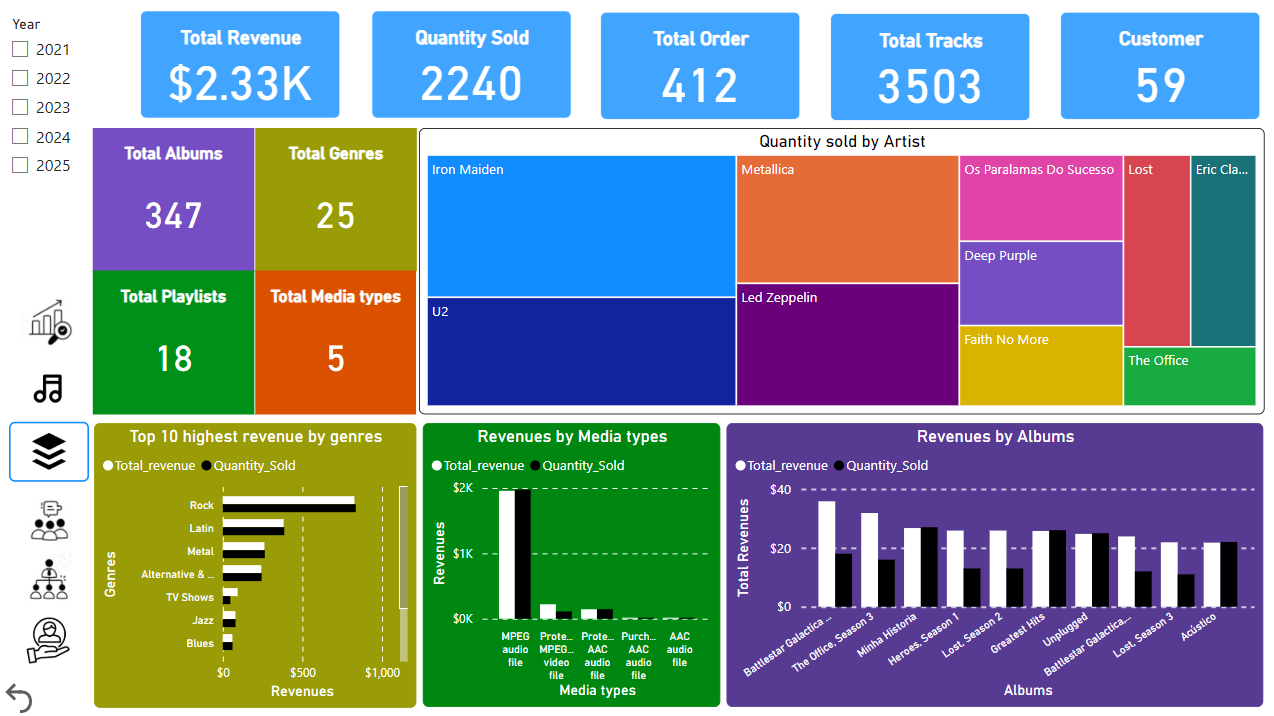
Thống kê thông tin nhân viên:



Phần trăm quay lại của khách hàng theo thời gian:



Tình hình doanh thu theo các đối tượng khác:



### **2.1.5 Kết luận**

**- Doanh thu tăng trưởng không ổn định qua các năm, có những thời điểm giảm mạnh. Các thời điểm doanh thu tăng mạnh là 01/2022, 04/2023, 06/2023, 11/2025. Giảm mạnh ở các thời điểm 11/2023 và 02/2025.**

**- Dự báo doanh thu đến 2026 cho thấy có xu hướng giảm và có khả năng phục hồi nhẹ.**

**- Tuy nhiên tổng số bài hát bán được giữa các tháng không chênh lệch cao (33-35 bài hát) 🡺 số lượng bài hát bán ra ổn định.**

**- Vì tập khách hàng tập trung chủ yếu là người Châu Âu và Châu Mỹ nên những bài hát thuộc dòng nhạc Rock và Latin được ưa chuộng nhất.**

**- Tập khách hàng Potential Loyalist là nhiều nhất, chiếm đến 51% trên tổng 8 tập khách hàng. Nên đầu tư giữ chân các khách hàng tiềm năng và triển khai chương trình tri ân các khách hàng thuộc tập khách hàng Champion.**

**- Phần trăm khách hàng quay lại tương đối cao, những khách hàng bắt đầu mua từ năm 2022 quay lại mua lên đến 50%, tuy nhiên lượng khách hàng còn ít (59 khách hàng) và thời gian quay lại còn cách khá xa (từ 3 tháng trở lên) nên còn cần phải quảng bá để tiếp cận nhiều khách hàng hơn.**

## ****2.2. Bài toán 2****

### **2.2.1. Mô tả dữ liệu**

Từ file **Music\_Project.xlsx** sẽ chỉ lấy những thông tin về hóa đơn, khách hàng, ngày mua hàng, số lượng mua, tổng tiền, địa chỉ mua hàng để giải quyết bài toán.

Các thu thập dữ liệu: Dùng công cụ SQL Server Management Studio → trỏ đến database DB\_Music → Chạy câu truy vấn sau:

Select I.InvoiceId,CustomerId,InvoiceDate,BillingCountry,Total,sum(quantity) as Quantity

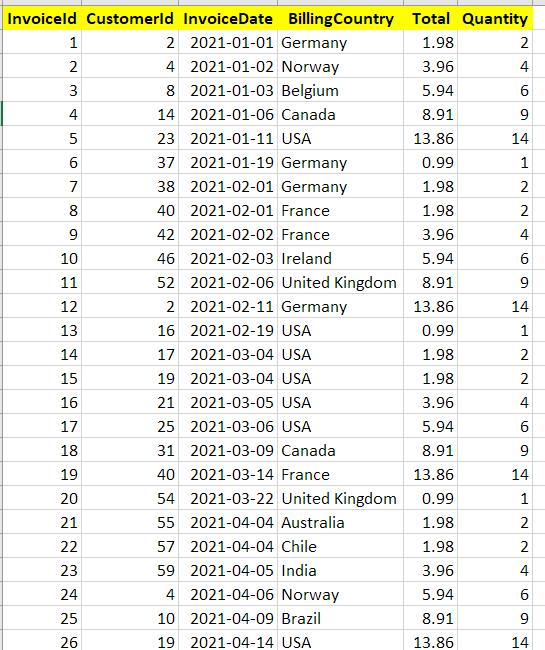
from Invoice I

join InvoiceLine IL on I.InvoiceId = IL.InvoiceId

GROUP By I.InvoiceId,CustomerId,InvoiceDate,BillingCountry,Total

ORDER BY I.InvoiceId

**→ Sau đó lưu vào file Dataset.csv** 

****

**Tổng quan dữ liệu**

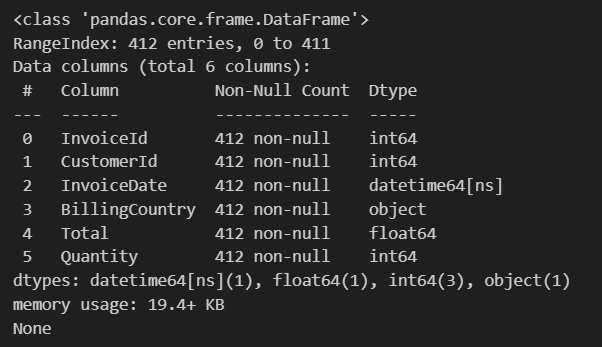
### **2.2.2. Mô tả các biến**

* InvoiceId: Mã hóa đơn.
* CustomerId: Mã khách hàng mua hóa đơn.
* InvoiceDate: Ngày mua hàng.
* BillingCountry: Quốc gia nơi khách mua hàng.
* Total: tổng tiền hóa đơn khách hàng phải thoanh toán.
* Quantiy: Số lượng bài hát khách hàng mua.

### **2.2.3. Tiền xử lý dữ liệu:**

**- Xử lý định dạng cột InvoiceDate thành kiểu Datetime:**

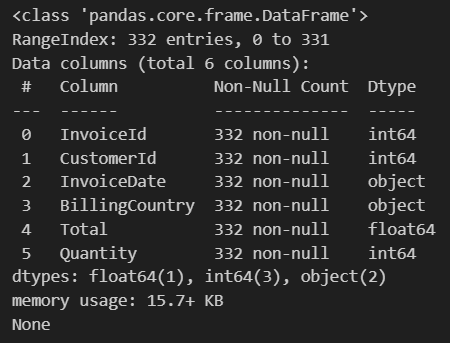
tx\_data['InvoiceDate']=pd.to\_datetime(tx\_data['InvoiceDate'])

****

**→ Có 412 dòng dữ liệu.**

**- Lấy dữ liệu 4 năm trở lại đây:**

tx\_3y = tx\_data[(tx\_data.InvoiceDate < date(2025,1,1))].reset\_index(drop=True)

****

**→ Có 332 dòng dữ liệu**

### **2.2.4. Tiến hành hành bài toán dự đoán:**

**Bước 1: Tìm số ngày giữa 3 lần mua gần nhất của khách hàng**

tx\_day\_order['PrevInvoiceDate'] = tx\_day\_order.groupby('CustomerId')['InvoiceDay'].shift(1)

tx\_day\_order['T2InvoiceDate'] = tx\_day\_order.groupby('CustomerId')['InvoiceDay'].shift(2)

tx\_day\_order['T3InvoiceDate'] = tx\_day\_order.groupby('CustomerId')['InvoiceDay'].shift(3)

tx\_day\_order['PrevInvoiceDate'] = pd.to\_datetime(tx\_day\_order['PrevInvoiceDate'])

tx\_day\_order['T2InvoiceDate'] = pd.to\_datetime(tx\_day\_order['T2InvoiceDate'])

tx\_day\_order['T3InvoiceDate'] = pd.to\_datetime(tx\_day\_order['T3InvoiceDate'])

tx\_day\_order['InvoiceDay'] = pd.to\_datetime(tx\_day\_order['InvoiceDay'])

tx\_day\_order['DayDiff'] = (tx\_day\_order['InvoiceDay'] - tx\_day\_order['PrevInvoiceDate']).dt.days

tx\_day\_order['DayDiff2'] = (tx\_day\_order['InvoiceDay'] - tx\_day\_order['T2InvoiceDate']).dt.days

tx\_day\_order['DayDiff3'] = (tx\_day\_order['InvoiceDay'] - tx\_day\_order['T3InvoiceDate']).dt.days

**Bước 2: Lấy ngày mua hàng gần nhất của khách hàng**

tx\_last\_purchase = tx\_3y.groupby('CustomerId').InvoiceDate.max().reset\_index()

tx\_last\_purchase.columns = ['CustomerId','MaxPurchaseDate']

**Bước 3: Lấy dữ liệu 1 năm tiếp theo để tính khoảng thời gian khách hàng quay lại**

tx\_next = tx\_data[(tx\_data.InvoiceDate >= date(2025,1,1)) & (tx\_data.InvoiceDate < date(2026,1,1))].reset\_index(drop=True)

tx\_next\_first\_purchase = tx\_next.groupby('CustomerId').InvoiceDate.min().reset\_index()

tx\_next\_first\_purchase.columns = ['CustomerId','MinPurchaseDate']

**Bước 4: Tìm ngày gần nhất khách hàng quay trở lại mua**

tx\_purchase\_dates = pd.merge(tx\_last\_purchase,tx\_next\_first\_purchase,on='CustomerId',how='left')

tx\_purchase\_dates['MaxPurchaseDate'] = pd.to\_datetime(tx\_purchase\_dates['MaxPurchaseDate'])

tx\_purchase\_dates['MinPurchaseDate'] = pd.to\_datetime(tx\_purchase\_dates['MinPurchaseDate'])

tx\_purchase\_dates['NextPurchaseDay'] = (tx\_purchase\_dates['MinPurchaseDate'] - tx\_purchase\_dates['MaxPurchaseDate']).dt.days

tx\_user = pd.merge(tx\_user, tx\_purchase\_dates[['CustomerId','NextPurchaseDay']],on='CustomerId',how='left')

**Bước 5: Đặt cho những người chưa quay lại mua tính tới 1 năm tiếp theo giá trị là 9999**

tx\_user = tx\_user.fillna(9999)

**Bước 6: Định nghĩa các khoảng thời gian khách hàng quay lại theo cụm**

* **Trên 500 ngày mới quay trở lại thì thuộc cụm 0**
* **Trên 200 ngày mới quay trở lại thì thuộc cụm 1**
* **Từ 200 ngày trở xuống thì thuộc cụm 2**

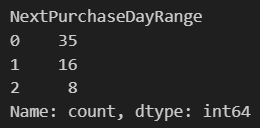
tx\_class['NextPurchaseDayRange'] = 2

tx\_class.loc[tx\_class.NextPurchaseDay>200,'NextPurchaseDayRange'] = 1

tx\_class.loc[tx\_class.NextPurchaseDay>500,'NextPurchaseDayRange'] = 0

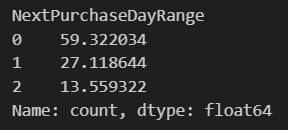
**→ Số lượng khách hàng theo cụm:**

tx\_class.NextPurchaseDayRange.value\_counts()

****

**→ Tỷ lệ khách hàng trong các cụm:**

tx\_class.NextPurchaseDayRange.value\_counts()

****

**Bước 7: Chạy thuật toán dự đoán khoảng thời gian:**

**Bước 7.1: Xác định biến phụ thuộc X, y:**

tx\_class = tx\_class.drop('NextPurchaseDay',axis=1)

X, y = tx\_class.drop('NextPurchaseDayRange',axis=1), tx\_class.NextPurchaseDayRange

**Bước 7.2: Chia tập dữ liệu thành 2 tập:**

* **Tập Train: X\_train, y\_train**
* **Tập test: X\_test, y\_test**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**Bước 7.3:** Training data

* Import thư viện của thuật toán:

from sklearn.model\_selection import KFold, cross\_val\_score, train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.tree import DecisionTreeClassifier, plot\_tree,DecisionTreeRegressor

* Chạy thuật toán training:

tree\_model = DecisionTreeClassifier(criterion="gini", max\_depth=3, random\_state=42)

tree\_model.fit(X\_train, y\_train)

**Bước 8: Kết quả dự đoán và đánh giá mô hình**

* **Kết quả dự đoán:**

y\_pred\_train = tree\_model.predict(X\_train)

X\_train['Predict']=y\_pred\_train

* **Đánh giá mô hình tập Train:**

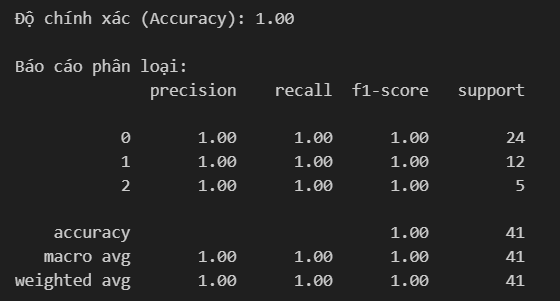
# Đánh giá mô hình

accuracy = accuracy\_score(y\_train, y\_pred\_train)

print(f"Độ chính xác (Accuracy): {accuracy:.2f}")

print("\nBáo cáo phân loại:")

print(classification\_report(y\_train, y\_pred\_train))

****

* **Áp dụng để lấy kết quả dự đoán tập Test:**

y\_pred = tree\_model.predict(X\_test)

X\_test['predict'] = y\_pred

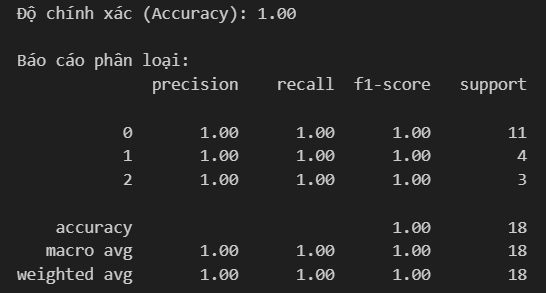
* **Đánh giá mô hình tập Test:**

accuracy = accuracy\_score(y\_test, y\_pred)

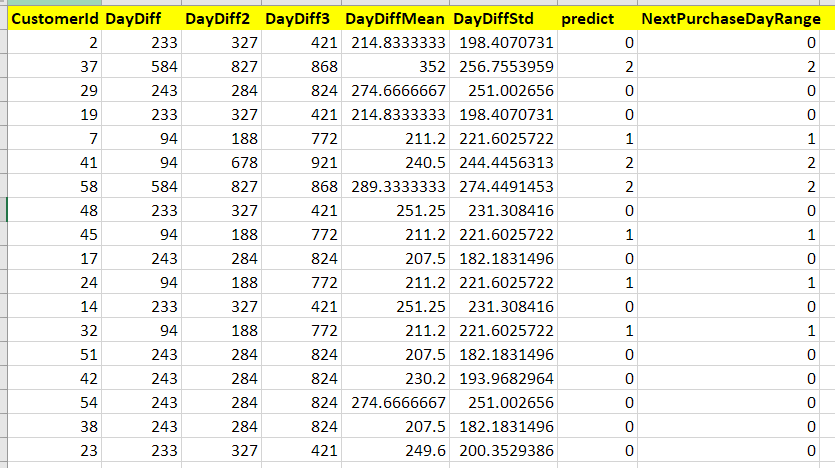
print(f"Độ chính xác (Accuracy): {accuracy:.2f}")

print("\nBáo cáo phân loại:")

print(classification\_report(y\_test,y\_pred))

****

**🡺** Dựa trên kết quả tập Test ta thấy được mô hình đạt độ chính xác 100% thời gian khách hàng quay lại thực tế như hình dưới đây. Suy ra mô hình đã hoạt động rất tốt.

****

### **2.2.5. Kết luận**

- Mô hình sử dụng dự đoán hoạt động rất tốt do có độ chính xác cao (100%)

- Thời gian để khách hàng quay lại mua hàng là khá cao, một phần vì sản phẩm không có tính hư hao, chỉ mang tính chất về tinh thần cho khách hàng.

- Lượng khác hàng quay lại dưới 200 ngày còn thấp (chỉ chiếm 17%), khách hàng sau hơn 500 ngày quay lại thì khá cao (chiếm 61%).

# **3. Các file đính kèm**

## 3.1 Raw data



## 3.2 Visualize data bằng công cụ Power BI:

## 3.3 Data xử lý bài toán dự đoán:



## 3.4 Full source code python:

from datetime import datetime, timedelta,date

import pandas as pd

import numpy as np

from \_\_future\_\_ import division

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import plotly.offline as pyoff

import plotly.graph\_objs as go

#------------------------------------------------------------------------------#

#                      PHÂN LOẠI KHÁCH HÀNG THEO RFM SCORE                     #

#------------------------------------------------------------------------------#

url=r'D:\DAS 2046 HCM\Silde and Assignment\Mock Project\Dataset.csv'

# url='https://drive.google.com/uc?id=' + url.split('/')[-2]

tx\_data = pd.read\_csv(url, encoding='unicode\_escape')

print(tx\_data.info())

print(tx\_data.describe())

tx\_data['InvoiceDate'] = pd.to\_datetime(tx\_data['InvoiceDate'])

tx\_data['BillingCountry'].value\_counts()

# ==============================Danh sách khách hàng===============================================#

tx\_user = pd.DataFrame(tx\_data['CustomerId'].unique())

tx\_user.columns = ['CustomerId']

tx\_user.info()

#-----------------------------------------------------------------#

#                             Recency                             #

#-----------------------------------------------------------------#

tx\_max\_purchase = tx\_data.groupby('CustomerId').InvoiceDate.max().reset\_index()

tx\_max\_purchase.columns = ['CustomerId','MaxPurchaseDate']

tx\_max\_purchase.info()

tx\_max\_purchase['MaxPurchaseDate'] = pd.to\_datetime(tx\_max\_purchase['MaxPurchaseDate'])

tx\_max\_purchase['Recency'] = (tx\_max\_purchase['MaxPurchaseDate'].max() - tx\_max\_purchase['MaxPurchaseDate']).dt.days

tx\_user = pd.merge(tx\_user, tx\_max\_purchase[['CustomerId','Recency']], on='CustomerId')

tx\_user.Recency.describe()

tx\_recency = tx\_user[['CustomerId','Recency']]

kmeans = KMeans(n\_clusters=5).fit(tx\_recency[['Recency']])

tx\_user['RecencyCluster'] = kmeans.predict(tx\_recency[['Recency']])

tx\_user['RecencyCluster'].value\_counts()

sse = {}

for k in range(1, 11):

  kmeans = KMeans(n\_clusters=k).fit(tx\_user[['Recency']])

  tx\_recency['clusters'] = kmeans.labels\_

  sse[k] = kmeans.inertia\_

  print(k,"-> ",kmeans.inertia\_)

import seaborn as sns

plt.title('The Elbow Method')

plt.xlabel('k')

plt.ylabel('SSE')

sns.pointplot(x=list(sse.keys()), y=list(sse.values()))

plt.show()

#-----------------------------------------------------------------#

#                             Frequency                           #

#-----------------------------------------------------------------#

tx\_frequency = tx\_data.groupby('CustomerId').InvoiceDate.count().reset\_index()

tx\_frequency.columns = ['CustomerId','Frequency']

kmeans = KMeans(n\_clusters=5).fit(tx\_frequency[['Frequency']])

tx\_frequency['FrequencyCluster'] = kmeans.predict(tx\_frequency[['Frequency']])

#-----------------------------------------------------------------#

#                             Monetary                            #

#-----------------------------------------------------------------#

tx\_data['Revenue'] = tx\_data['Total']

tx\_revenue = tx\_data.groupby('CustomerId').Revenue.sum().reset\_index()

tx\_revenue.columns = ['CustomerId','Monetary']

kmeans = KMeans(n\_clusters=5).fit(tx\_revenue[['Monetary']])

tx\_revenue['MonetaryCluster'] = kmeans.predict(tx\_revenue[['Monetary']])

tx\_user\_frequency = pd.merge(tx\_user, tx\_frequency, on='CustomerId')

tx\_user\_rfm = pd.merge(tx\_user\_frequency, tx\_revenue, on='CustomerId')

#-----------Chuẩn hóa giá trị theo thứ tự điểm cụm-----------#

#-----------------------------Recency------------------------#

# Tính giá trị trung bình của Recency trong mỗi cụm

cluster\_means = tx\_user\_rfm.groupby('RecencyCluster')['Recency'].mean().sort\_values(ascending=False)

print(cluster\_means)

# Tạo ánh xạ lại các cụm: Nhóm với trung bình LỚN nhất sẽ được đánh số 1, tiếp theo là 2, ...

cluster\_map = {cluster: rank + 1 for rank, cluster in enumerate(cluster\_means.index)}

print(cluster\_map)

# Gán lại nhãn cụm theo thứ tự mong muốn

tx\_user\_rfm['RecencyClusterSort'] = tx\_user\_rfm['RecencyCluster'].map(cluster\_map)

cluster\_means = tx\_user\_rfm.groupby('RecencyClusterSort')['Recency'].mean().sort\_values()

print(cluster\_means)

#-----------------------------Frequency------------------------#

# Tính giá trị trung bình của Frequency trong mỗi cụm

cluster\_means = tx\_user\_rfm.groupby('FrequencyCluster')['Frequency'].mean().sort\_values()

print(cluster\_means)

# Tạo ánh xạ lại các cụm: Nhóm với trung bình NHỎ nhất sẽ được đánh số 1, tiếp theo là 2, ...

cluster\_map = {cluster: rank + 3 for rank, cluster in enumerate(cluster\_means.index)}

print(cluster\_map)

# Gán lại nhãn cụm theo thứ tự mong muốn

tx\_user\_rfm['FrequencyClusterSort'] = tx\_user\_rfm['FrequencyCluster'].map(cluster\_map)

cluster\_means = tx\_user\_rfm.groupby('FrequencyClusterSort')['Frequency'].mean().sort\_values()

print(cluster\_means)

#---------------------------------Monetary--------------------------------#

# Tính giá trị trung bình của Monetary trong mỗi cụm

cluster\_means = tx\_user\_rfm.groupby('MonetaryCluster')['Monetary'].mean().sort\_values()

print(cluster\_means)

# Tạo ánh xạ lại các cụm: Nhóm với trung bình NHỎ nhất sẽ được đánh số 1, tiếp theo là 2, ...

cluster\_map = {cluster: rank + 1 for rank, cluster in enumerate(cluster\_means.index)}

print(cluster\_map)

# Gán lại nhãn cụm theo thứ tự mong muốn

tx\_user\_rfm['MonetaryClusterSort'] = tx\_user\_rfm['MonetaryCluster'].map(cluster\_map)

cluster\_means = tx\_user\_rfm.groupby('MonetaryClusterSort')['Monetary'].mean().sort\_values()

print(cluster\_means)

# Gộp các điểm cụm thành RFM score

tx\_user\_rfm['RFM\_Score'] = tx\_user\_rfm.RecencyClusterSort.astype(str) + tx\_user\_rfm.FrequencyClusterSort.astype(str) + tx\_user\_rfm.MonetaryClusterSort.astype(str)

#=============Mapping RFM score sang Customer Segment===============#

# Bảng định nghĩa từ Scores sang Segment

score\_to\_segment = {

    '01\_Champions': [555, 554, 544, 545, 454, 455, 445],

    '02\_Loyal': [543, 444, 435, 355, 354, 345, 344, 335],

    '03\_Potential Loyalist': [553, 551, 552, 541, 542, 533, 532, 531, 452, 451, 442, 441, 431, 453, 433, 432, 423, 353, 352, 351, 342, 341, 333, 323],

    '04\_New Customers': [512, 511, 422, 421, 412, 411, 311],

    '05\_Promising': [525, 524, 523, 522, 521, 515, 514, 513, 425, 424, 413, 414, 415, 315, 314, 313],

    '06\_Need Attention': [535, 534, 443, 434, 343, 334, 325, 324],

    '07\_About To Sleep': [331, 321, 312, 221, 213, 231, 241, 251],

    '08\_At Risk': [255, 254, 245, 244, 253, 252, 243, 242, 235, 234, 225, 224, 153, 152, 145, 143, 142, 135, 134, 133, 125, 124],

    '09\_Cannot Lose Them': [155, 154, 144, 214, 215, 115, 114, 113],

    '10\_Hibernating customers': [332, 322, 231, 241, 251, 233, 232, 223, 222, 132, 123, 122, 212, 211],

    '11\_Lost customers': [111, 112, 121, 131, 141, 151]

}

# Đảo ngược mapping để dễ tra cứu

mapping = {score: segment for segment, scores in score\_to\_segment.items() for score in scores}

# Map RFM\_Score sang Segment

tx\_user\_rfm['Segment'] = tx\_user\_rfm['RFM\_Score'].astype(int).map(mapping)

grouped = tx\_user\_rfm.groupby('Segment').agg(Count=('CustomerId', 'count')

                                                    , Avg\_Recency=('Recency', 'mean')

                                                    , Avg\_Frequency=('Frequency', 'mean')

                                                    , Avg\_Monetary=('Monetary', 'mean')

                                                    )

print(grouped)

tx\_user\_rfm.to\_csv(r'D:\DAS 2046 HCM\Silde and Assignment\Mock Project\CustomerSegment.csv',index=False)

#------------------------------------------------------------------------------#

#             DỰ ĐOÁN NGÀY MUA HÀNG TIẾP THEO CỦA KHÁCH HÀNG                   #

#------------------------------------------------------------------------------#

tx\_data['InvoiceDate']=pd.to\_datetime(tx\_data['InvoiceDate']).dt.date

tx\_3y = tx\_data[(tx\_data.InvoiceDate < date(2025,1,1))].reset\_index(drop=True)

print(tx\_3y.info())

print(tx\_data.info())

tx\_day\_order = tx\_3y[['CustomerId','InvoiceDate']]

tx\_day\_order['InvoiceDay'] = pd.to\_datetime(tx\_3y['InvoiceDate']).dt.date

tx\_day\_order = tx\_day\_order.sort\_values(['CustomerId','InvoiceDate'])

# Loại bỏ những ngày trùng

tx\_day\_order = tx\_day\_order.drop\_duplicates(subset=['CustomerId','InvoiceDay'],keep='first')

# Tìm số ngày giữa 3 lần mua gần nhất của khách hàng

tx\_day\_order['PrevInvoiceDate'] = tx\_day\_order.groupby('CustomerId')['InvoiceDay'].shift(1)

tx\_day\_order['T2InvoiceDate'] = tx\_day\_order.groupby('CustomerId')['InvoiceDay'].shift(2)

tx\_day\_order['T3InvoiceDate'] = tx\_day\_order.groupby('CustomerId')['InvoiceDay'].shift(3)

tx\_day\_order['PrevInvoiceDate'] = pd.to\_datetime(tx\_day\_order['PrevInvoiceDate'])

tx\_day\_order['T2InvoiceDate'] = pd.to\_datetime(tx\_day\_order['T2InvoiceDate'])

tx\_day\_order['T3InvoiceDate'] = pd.to\_datetime(tx\_day\_order['T3InvoiceDate'])

tx\_day\_order['InvoiceDay'] = pd.to\_datetime(tx\_day\_order['InvoiceDay'])

tx\_day\_order['DayDiff'] = (tx\_day\_order['InvoiceDay'] - tx\_day\_order['PrevInvoiceDate']).dt.days

tx\_day\_order['DayDiff2'] = (tx\_day\_order['InvoiceDay'] - tx\_day\_order['T2InvoiceDate']).dt.days

tx\_day\_order['DayDiff3'] = (tx\_day\_order['InvoiceDay'] - tx\_day\_order['T3InvoiceDate']).dt.days

tx\_day\_diff = tx\_day\_order.groupby('CustomerId').agg({'DayDiff': ['mean','std']}).reset\_index()

tx\_day\_diff.columns = ['CustomerId', 'DayDiffMean','DayDiffStd']

# Ngày order gần nhất của khách hàng, loại những ngày lần 2 trở đi

tx\_day\_order\_last = tx\_day\_order.drop\_duplicates(subset=['CustomerId'],keep='last')

print(tx\_day\_order.info())

print(tx\_day\_order\_last.info())

tx\_day\_order\_last = tx\_day\_order\_last.dropna()

tx\_day\_order\_last = pd.merge(tx\_day\_order\_last, tx\_day\_diff, on='CustomerId')

# Lấy khách hàng duy nhất

tx\_user = pd.DataFrame(tx\_3y['CustomerId'].unique())

tx\_user.columns = ['CustomerId']

# Lấy ngày mua hàng gần nhất của khách hàng

tx\_last\_purchase = tx\_3y.groupby('CustomerId').InvoiceDate.max().reset\_index()

tx\_last\_purchase.columns = ['CustomerId','MaxPurchaseDate']

# Lấy dữ liệu 1 năm tiếp theo để tính khoảng thời gian khách hàng quay lại

tx\_next = tx\_data[(tx\_data.InvoiceDate >= date(2025,1,1)) & (tx\_data.InvoiceDate < date(2026,1,1))].reset\_index(drop=True)

tx\_next\_first\_purchase = tx\_next.groupby('CustomerId').InvoiceDate.min().reset\_index()

tx\_next\_first\_purchase.columns = ['CustomerId','MinPurchaseDate']

# Tìm ngày gần nhất khách hàng quay trở lại mua

tx\_purchase\_dates = pd.merge(tx\_last\_purchase,tx\_next\_first\_purchase,on='CustomerId',how='left')

tx\_purchase\_dates['MaxPurchaseDate'] = pd.to\_datetime(tx\_purchase\_dates['MaxPurchaseDate'])

tx\_purchase\_dates['MinPurchaseDate'] = pd.to\_datetime(tx\_purchase\_dates['MinPurchaseDate'])

tx\_purchase\_dates['NextPurchaseDay'] = (tx\_purchase\_dates['MinPurchaseDate'] - tx\_purchase\_dates['MaxPurchaseDate']).dt.days

tx\_user = pd.merge(tx\_user, tx\_purchase\_dates[['CustomerId','NextPurchaseDay']],on='CustomerId',how='left')

# Đặt cho những người chưa quay lại mua giá trị là 9999

tx\_user.info()

tx\_user = tx\_user.fillna(9999)

tx\_user = pd.merge(tx\_user, tx\_day\_order\_last[['CustomerId','DayDiff','DayDiff2','DayDiff3','DayDiffMean','DayDiffStd']], on='CustomerId')

tx\_user.info()

#=======Chuyển đổi cột có kiểu dữ liệu chuỗi thành số=======#

tx\_class = tx\_user.copy()

tx\_class = pd.get\_dummies(tx\_class)

tx\_class.info()

#=======Định nghĩa các khoảng thời gian khách hàng quay lại theo cụm============#

# >500 ngày   : 0

# >200 ngày   : 1

# <= 200 ngày : 2

tx\_class['NextPurchaseDayRange'] = 2

tx\_class.loc[tx\_class.NextPurchaseDay>200,'NextPurchaseDayRange'] = 1

tx\_class.loc[tx\_class.NextPurchaseDay>500,'NextPurchaseDayRange'] = 0

tx\_class.NextPurchaseDayRange.value\_counts()

# NextPurchaseDayRange

# 0    35

# 1    16

# 2     8

tx\_class.NextPurchaseDayRange.value\_counts()/len(tx\_user)\*100

#NextPurchaseDayRange

# 0  :  59.322034 ~= 59.32%

# 1  :  27.118644 ~= 27.11%

# 2  :  13.559322 ~= 13.56%

# Xem mối tương quan giữa các biến

corr = tx\_class[tx\_class.columns].corr()

plt.figure(figsize = (30,20))

sns.heatmap(corr, annot = True, linewidths=0.2, fmt=".2f")

# Chia tập dữ liệu

tx\_class = tx\_class.drop('NextPurchaseDay',axis=1)

X, y = tx\_class.drop('NextPurchaseDayRange',axis=1), tx\_class.NextPurchaseDayRange

from sklearn.cluster import KMeans

import plotly.offline as pyoff

import plotly.graph\_objs as go

from sklearn.metrics import classification\_report,confusion\_matrix

from sklearn.svm import SVC

from sklearn.multioutput import MultiOutputClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

import xgboost as xgb

from sklearn.model\_selection import KFold, cross\_val\_score, train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

from sklearn.tree import DecisionTreeClassifier, plot\_tree,DecisionTreeRegressor

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Tìm thuật toán dự đoán tốt nhất

models = []

models.append(("LR",LogisticRegression()))

models.append(("RF",RandomForestClassifier()))

models.append(("Dtree",DecisionTreeClassifier()))

models.append(("XGB",xgb.XGBClassifier()))

for name,model in models:

    kfold = KFold(n\_splits=2)

    cv\_result = cross\_val\_score(model,X\_train,y\_train, cv = kfold,scoring = "accuracy")

    print(name, ': ',cv\_result)

# LR :  [0.85714286 1.        ]

# RF :  [0.85714286 1.        ]

# Dtree :  [0.85714286 1.        ]

# XGB :  [0.80952381 1.        ]

#======Chọn Decision Tree Clasifier======#

tree\_model = DecisionTreeClassifier(criterion="gini", max\_depth=3, random\_state=42)

tree\_model.fit(X\_train, y\_train)

y\_pred\_train = tree\_model.predict(X\_train)

X\_train['Predict']=y\_pred\_train

# Đánh giá mô hình

accuracy = accuracy\_score(y\_train, y\_pred\_train)

print(f"Độ chính xác (Accuracy): {accuracy:.2f}")

print("\nBáo cáo phân loại:")

print(classification\_report(y\_train, y\_pred\_train))

y\_pred = tree\_model.predict(X\_test)

X\_test['predict'] = y\_pred

# Đánh giá mô hình

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Độ chính xác (Accuracy): {accuracy:.2f}")

print("\nBáo cáo phân loại:")

print(classification\_report(y\_test,y\_pred))