FUNDAÇÃO ESCOLA DE COMÉRCIO ÁLVARES PENTEADO
FECAP

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

João Pedro Brosselin de Albuquerque Souza - 24026155

Marcella Santana Gonçalves Diniz Rocha - 24025750

Thays Helyda da Silva Pontes - 24026610

Gustavo de souza Castro - 20021558

Preparação dos Dados

Fidelize

São Paulo–SP

2025

01. INTRODUÇÃO

O intuito desse documento é o detalhamento da etapa de preparação e enriquecimento dos dados do projeto Fidelize, dedicada à análise de comportamento de clientes.

O foco principal do trabalho foi converter dados operacionais brutos (pedidos e clientes) em um dataset analítico, pronto para alimentar o nosso dashboard.

Diferente de uma simples limpeza, o processo foi centrado na criação de novas variáveis de inteligência de marketing. Isso foi alcançado através da aplicação da metodologia RFM (Recência, Frequência e Valor) e da definição de estágios de Ciclo de Vida do Cliente, permitindo uma segmentação de clientes profunda e acionável que contribui com o propósito do nosso projeto, a fidelização de clientes.

02. PREPARAÇÃO DOS DADOS

O fluxo de trabalho foi estruturado em etapas lógicas, desde a leitura dos dados brutos até a geração de arquivos CSV enriquecidos, prontos para análise.

02.1. Seleção e Carregamento dos Dados

O script inicia carregando os arquivos CSVs do projeto. Para o escopo desta análise, focada no comportamento transacional do consumidor, os arquivos `Customer_semicolon.csv` e `Order_semicolon.csv` foram selecionados como as fontes de dados primárias.

```python
import pandas as pd
import numpy as np
import datetime as dt
import io

arquivos_csv = {
    'customers': 'Customer_semicolon.csv',
    'campaigns': 'Campaign_semicolon.csv',
    'orders': 'Order_semicolon.csv',
    'campaign_sends': 'CampaignQueue_semicolon.csv'
}

dataframes = {}

for nome, arquivo in arquivos_csv.items():
    try:
        df = pd.read_csv(arquivo, sep=';')
        dataframes[nome] = df
    except FileNotFoundError:
        print(f'{arquivo} não encontrado.')

df_customers = dataframes.get('customers', pd.DataFrame())
df_orders = dataframes.get('orders', pd.DataFrame())
df_campaigns = dataframes.get('campaigns', pd.DataFrame())
df_campaign_sends = dataframes.get('campaign_sends', pd.DataFrame())
```

02.2. Padronização e Correção de Dados

Para garantir a qualidade e a precisão dos cálculos futuros, duas ações de limpeza e padronização foram executadas:

1. **Tratamento de Datas (Tabela Orders):** A coluna `createdAt` foi corretamente convertida para o tipo `datetime`. Foi essencial especificar o parâmetro `dayfirst=True`, assegurando que o formato de data brasileiro (DD/MM/AAAA) fosse interpretado corretamente, o que é vital para o cálculo da Recência.
2. **Padronização de Gênero (Tabela Customers):** O campo `gender` foi normalizado. Valores nulos foram substituídos pela categoria "Não informado", e as abreviações ("M", "F", "O") foram mapeadas para seus rótulos completos ("Masculino", "Feminino", "Outro") para facilitar a leitura nos dashboards.

```
# TABELA DE PEDIDOS(Orders)

df_orders_cleaned = df_orders.dropna(subset=['createdAt']).copy()
# Converte a data, especificando o formato brasileiro (dayfirst)
df_orders_cleaned['createdAt'] = pd.to_datetime(df_orders_cleaned['createdAt'], dayfirst=True)

# TABELA DE CLIENTES(Customers)

df_customers['gender'] = df_customers['gender'].fillna('Não informado')
df_customers['gender'] = df_customers['gender'].replace({
    'M': 'Masculino',
    'F': 'Feminino',
    'O': 'Outro'
})
```

02.3. Criação de Métricas de Comportamento (RFM)

Nesta etapa os dados brutos de pedidos são transformados em inteligência sobre o cliente.

- **Cálculo RFM:** Para cada cliente, o script calcula as três métricas fundamentais de RFM, considerando apenas pedidos com status válidos ("CONCLUDED" e “DISPATCHED”).
    - **Recência:** Dias desde a última compra.
    - **Frequência:** Número total de pedidos.
    - **Valor Monetário:** Soma total gasta.

```
# CÁLCULO DE RECÊNCIA, FREQUÊNCIA E VALOR (RFM)

status_validos = ['CONCLUDED', 'DISPATCHED']
df_orders_rfm = df_orders_cleaned[df_orders_cleaned['status'].isin(status_validos)].copy()

# snapshot_date definido como um dia após a última compra registrada
snapshot_date = df_orders_rfm['createdAt'].max() + dt.timedelta(days=1)

rfm_data = df_orders_rfm.groupby('customer').agg(
    Recencia=('createdAt', lambda x: (snapshot_date - x.max()).days),
    Frequencia=('id', 'count'),
    ValorMonetario=('totalAmount', 'sum')
).reset_index()
```

- **Derivação de Segmentos:** Com base no RFM, o script deriva:
    - **Quartis e `RFM_Score`:** classifica os clientes em quartis para R, F e M, permitindo a criação de segmentos como "Clientes de Ouro" (Score '111').
    - **`TicketMedio`:** calcula o valor médio gasto por pedido.

```
# SEGMENTAÇÃO RFM E TICKET MÉDIO

# Recência: Menor é melhor
rfm_data['R_Quartil'] = pd.qcut(rfm_data['Recencia'], 4, labels=[1, 2, 3, 4]).astype(int)
# Frequência: Maior é melhor (usamos rank(method='first') para evitar valores duplicados nos quartis)
rfm_data['F_Quartil'] = pd.qcut(rfm_data['Frequencia'].rank(method='first'), 4, labels=[4, 3, 2, 1]).astype(int)
# Monetário: Maior é melhor
rfm_data['M_Quartil'] = pd.qcut(rfm_data['ValorMonetario'], 4, labels=[4, 3, 2, 1]).astype(int)

rfm_data['RFM_Score'] = rfm_data['R_Quartil'].astype(str) + rfm_data['F_Quartil'].astype(str) + rfm_data['M_Quartil'].astype(str)

# 1 é bom, 4 é ruim
condicoes_segmento = [
    (rfm_data['RFM_Score'] == '111'), # R=1, F=1, M=1
    (rfm_data['R_Quartil'] <= 2) & (rfm_data['F_Quartil'] <= 2), # R e F bons (Leais)
    (rfm_data['R_Quartil'] >= 3) & (rfm_data['F_Quartil'] <= 2), # R ruim, F bom (Em Risco)
    (rfm_data['R_Quartil'] <= 2) & (rfm_data['F_Quartil'] >= 3), # R bom, F ruim (Novos/Ocasionais)
    (rfm_data['R_Quartil'] >= 3) & (rfm_data['F_Quartil'] >= 3)  # R e F ruins (Perdidos/Baixo Valor)
]

rotulos_segmento = [
    'Clientes de Ouro',
    'Clientes Leais',
    'Em Risco (Leais)',
    'Novos/Ocasionais',
    'Perdidos/Baixo Valor'
]

rfm_data['Segmento_RFM'] = np.select(condicoes_segmento, rotulos_segmento, default='Regular')

# Cálculo do Ticket Médio por Cliente
rfm_data['TicketMedio'] = rfm_data['ValorMonetario'] / rfm_data['Frequencia']
```

- **`CicloDeVida`:** aplica uma regra de negócio para classificar clientes em estágios como "Novo", "Ativo", "Em Risco" ou "Perdido", com base na sua recência e frequência de compra.

```python
# CICLO DE VIDA

def classificar_ciclo_vida_refinado(row):
    frequencia = row['Frequencia']
    recencia = row['Recencia']

    if frequencia == 1:
        if recencia <= 90: # Novo cliente que comprou recentemente
            return 'Novo'
        else: # Cliente que comprou uma vez, há muito tempo
            return 'Perdido (One-timer)'
    elif frequencia > 1:
        if recencia <= 60: # Cliente recorrente e ativo
            return 'Ativo'
        elif recencia <= 120: # Cliente recorrente em risco
            return 'Em Risco'
        else: # Cliente recorrente que não compra há mais de 120 dias
            return 'Perdido (Ex-Ativo)'
    return 'Indefinido' # Não deve acontecer em rfm_data

rfm_data['CicloDeVida'] = rfm_data.apply(classificar_ciclo_vida_refinado, axis=1)
```

02.4. Consolidação dos Dados do Cliente

Uma vez que todas as métricas de comportamento foram calculadas (a partir da tabela `Orders`), elas foram integradas de volta à tabela principal de `Customers`.

Este `merge` resultou no `df_customers_final`, um dataset único e enriquecido. Clientes que não possuíam histórico de pedidos válidos foram corretamente identificados e rotulados com o ciclo de vida "Inativo".

```python
# Juntamos a base de clientes com os dados derivados de RFM
df_customers_final = df_customers.merge(
    rfm_data,
    left_on='id',
    right_on='customer',
    how='left'
)

# Clientes que não estão na tabela RFM (nunca fizeram um pedido válido) são classificados como 'Inativo'
df_customers_final['CicloDeVida'] = df_customers_final['CicloDeVida'].fillna('Inativo')

# Removemos a coluna 'customer' duplicada após o merge
if 'customer' in df_customers_final.columns:
    df_customers_final = df_customers_final.drop(columns=['customer'])
```

02.5. Exportação dos Datasets Finais

Como resultado final do processo, o script salva dois arquivos CSV prontos para o consumo:

1. **`Fidelize_Customers_Enriquecido.csv`:** contém a lista de clientes com todas as novas métricas de RFM, segmento e ciclo de vida.
2. **`Fidelize_Orders_Limpo.csv`:** contém o histórico de pedidos com datas e tipos de dados corretos.

```
try:
    df_customers_final.to_csv(
        'Fidelize_Customers_Enriquecido.csv',
        index=False,
        sep=';',
        decimal=',',
        encoding='utf-8-sig'
    )

    df_orders_cleaned.to_csv(
        'Fidelize_Orders_Limpo.csv',
        index=False,
        sep=';',
        decimal=',',
        encoding='utf-8-sig'
    )

    print("Arquivos 'Fidelize_Customers_Enriquecido.csv' e 'Fidelize_Orders_Limpo.csv' salvos com sucesso.")

except Exception as e:
    print(f"Erro ao salvar os arquivos: {e}")
```

03. Exemplo de Aplicação: Análise de Segmentação RFM

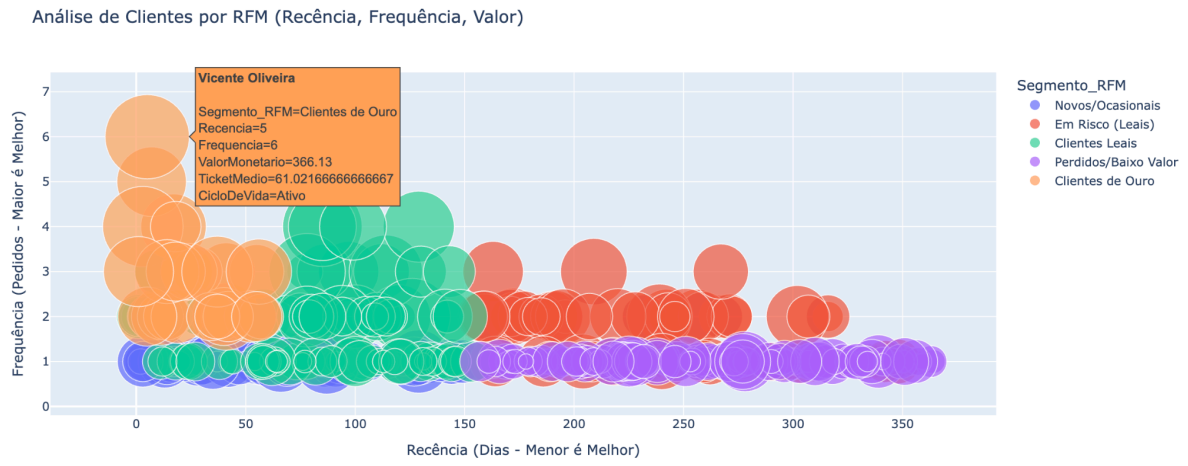
Para ilustrar a aplicação prática dos dados preparados, podemos usar o *dataset* `Fidelize_Customers_Enriquecido.csv` para gerar visualizações de inteligência de negócio. O Gráfico Análise de Clientes por RFM é um exemplo direto de como as métricas derivadas se transformam em insights acionáveis.

Este gráfico de dispersão (ou bolhas) utiliza as colunas que criamos:

- **Eixo X (Recência):** Posiciona os clientes com compras mais recentes à esquerda.
- **Eixo Y (Frequência):** Posiciona os clientes mais leais (com mais compras) na parte superior.
- **Tamanho da Bolha (Valor Monetário):** O tamanho de cada ponto representa o gasto total do cliente. Bolhas maiores indicam clientes de maior valor financeiro.
- **Cor (Segmento RFM):** As cores agrupam os clientes nos segmentos que derivamos, como "Clientes de Ouro" ou "Novos/Ocasionais".

Em uma única visualização, a gestão pode identificar seus melhores clientes (bolhas grandes no canto superior esquerdo), clientes em risco (bolhas no canto superior direito) ou clientes novos com alto potencial (bolhas grandes no canto inferior esquerdo).

```
# Segmentação RFM (Dispersão/Bolhas)
df_rfm_plot = df_customers_final.dropna(subset=['Recencia'])
fig_bubble_rfm = px.scatter(df_rfm_plot,
                            x='Recencia',
                            y='Frequencia',
                            size='ValorMonetario',
                            color='Segmento_RFM', # Usando a segmentação refinada
                            title='Análise de Clientes por RFM (Recência, Frequência, Valor)',
                            hover_name='name', # Mostra o nome do cliente
                            hover_data=['TicketMedio', 'CicloDeVida'],
                            size_max=60)
fig_bubble_rfm.update_layout(xaxis_title='Recência (Dias - Menor é Melhor)',
                             yaxis_title='Frequência (Pedidos - Maior é Melhor)')
fig_bubble_rfm.show()
```



04. CONCLUSÃO

Ao final da execução, os arquivos gerados contêm uma base de clientes segmentada e pronta para utilização. Podemos enfim focar diretamente na análise de KPIs de marketing, como a retenção de clientes por segmento e o impacto financeiro dos "Clientes de Ouro", pois todo o trabalho pesado de preparação de dados já foi realizado.

O diferencial deste trabalho não se limitou à limpeza e padronização de dados, mas avançou para a criação efetiva de inteligência de marketing, que é um dos objetivos centrais do Fidelize. A equipe de análise pode agora focar diretamente na geração de insights estratégicos.