ENTREGA 1 – PROJETO INTERDISCIPLINAR

Introdução

Este documento apresenta uma análise estruturada do código Python desenvolvido no ambiente Google Colab para a análise dos dados disponibilizados pela *PicMoney* em suas quatro tabelas (base_players, transacoes_cupons, pedestres_paulista e lojas_valores). O objetivo principal deste documento é documentar as etapas feitas para cumprir os requisitos propostos: entender os dados, coletar dados iniciais, descrever os dados, explorar os dados e verificar a qualidade dos dados.

Bloco 1: Importação de Bibliotecas e Configurações

Neste bloco inicial, foram importadas as bibliotecas fundamentais para a execução das tarefas de manipulação e visualização de dados. A biblioteca Pandas (pd) é utilizada para a criação e gestão de DataFrames, que são estruturas tabulares que organizam os dados. A biblioteca NumPy (np) oferece suporte a operações numéricas de alto desempenho. Para a visualização de dados, as bibliotecas Matplotlib (plt) e Seaborn (sns) foram importadas, permitindo a criação de gráficos e a customização de seus elementos visuais. Adicionalmente, foram aplicadas configurações para o estilo visual dos gráficos e para garantir que todas as colunas de um DataFrame sejam exibidas integralmente.

```
# Bloco 1: Imports e Configurações
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Configuração visual
plt.style.use("seaborn-v0_8")
sns.set_palette("pastel")
pd.set_option("display.max_columns", None)
```

Bloco 2: Carregamento dos Dados

A segunda etapa do processo consiste no carregamento dos conjuntos de dados a partir de arquivos CSV. O código carrega quatro arquivos distintos: base_players.csv, lojas_valores.csv, transacoes_cupons.csv e pedestres_paulista.csv. Cada arquivo é lido por meio da biblioteca Pandas, utilizando o ponto e vírgula (;) como delimitador de colunas. Após o carregamento, a forma (dimensões) de cada DataFrame é exibida, indicando o número de linhas e colunas de cada conjunto de dados.

players: 10.000 linhas e 11 colunas.

lojas: 10.000 linhas e 11 colunas.

transacoes: 100.000 linhas e 12 colunas.

pedestres: 100.000 linhas e 15 colunas.

```python
# Bloco 2: Carregar os datasets
players = pd.read_csv("base_players.csv", sep=";")
lojas = pd.read_csv("lojas_valores.csv", sep=";")
transacoes = pd.read_csv("transacoes_cupons.csv", sep=";")
pedestres = pd.read_csv("pedestres_paulista.csv", sep=";")

# Mostrar dimensões
print("Players:", players.shape)
print("Lojas:", lojas.shape)
print("Transações:", transacoes.shape)
print("Pedestres:", pedestres.shape)
```

Bloco 3: Verificação da Qualidade dos Dados

Para assegurar a integridade dos dados, o terceiro bloco implementa uma função (qualidade_dados) que realiza uma análise detalhada de cada DataFrame. Esta função verifica as dimensões dos conjuntos, os tipos de dados de cada coluna, a presença de valores ausentes (NaN) e o número de linhas duplicadas.

A análise revelou:

players: As colunas cidade_trabalho e bairro_trabalho apresentam 3.969 valores ausentes, enquanto cidade_escola e bairro_escola possuem 6.932 valores ausentes. Não foram encontradas linhas duplicadas.

lojas: Não foram identificados valores ausentes ou linhas duplicadas.

pedestres: As colunas data_ultima_compra, ultimo_tipo_cupom, ultimo_valor_capturado e ultimo_tipo_loja apresentam 40.043 valores ausentes. Não foram encontradas linhas duplicadas.

```python
# Bloco 3: Verificar qualidade dos dados

def qualidade_dados(df, nome):
    print(f"\nDataset: {nome}")
    print("Dimensões:", df.shape)
    print("\nTipos de dados:")
    print(df.dtypes)
    print("\nValores ausentes:")
    print(df.isnull().sum())
    print("\nDuplicados:", df.duplicated().sum())
    print("="*60)

qualidade_dados(players, "Players")
qualidade_dados(lojas, "Lojas")
qualidade_dados(transacoes, "Transações")
qualidade_dados(pedestres, "Pedestres")
```

Bloco 4: Tratamento de Dados Duplicados e Ausentes

Neste bloco, é realizado um tratamento inicial para a remoção de possíveis linhas duplicadas e valores ausentes (NaN) nos quatro DataFrames. Embora a análise anterior tenha indicado que não há duplicados, a execução do método drop_duplicates() garante um processo de limpeza padronizado e robusto. Adicionalmente, o método dropna() é aplicado para remover as linhas que contêm valores ausentes. É importante notar que essa etapa de tratamento de valores ausentes pode reduzir significativamente o tamanho dos DataFrames, dependendo da quantidade de dados faltantes.

```
# Bloco 4: Tratamento básico dos dados

# Remover possíveis duplicados
players = players.drop_duplicates()
lojas = lojas.drop_duplicates()
transacoes = transacoes.drop_duplicates()
pedestres = pedestres.drop_duplicates()

# Remover possíveis valores ausentes
players = players.dropna()
lojas = lojas.dropna()
transacoes = transacoes.dropna()
pedestres = pedestres.dropna()
```

Bloco 5: Pré-visualização dos Dados

Este bloco apresenta uma pré-visualização das cinco primeiras linhas de cada DataFrame (players, lojas, transacoes e pedestres) para verificar o formato e o conteúdo dos dados após as etapas de carregamento e tratamento. Esta etapa é crucial para uma inspeção visual rápida e para confirmar que o processamento inicial foi bem-sucedido. A função display() é utilizada para exibir os DataFrames de forma formatada.

```
# Bloco 5: Pré-visualização
print("Players:")
display(players.head())

print("Lojas:")
display(lojas.head())

print("Transações:")
display(transacoes.head())

print("Pedestres:")
display(pedestres.head())
```

Bloco 6: Análise de Dados Agrupados

Este bloco de código realiza análises estatísticas dos dados, agrupando-os por categoria e calculando a soma e a média das transações. Os resultados são exibidos em um formato tabulado, revelando insights sobre a frequência e o valor das transações em cada categoria.

```python
# Bloco 6: Estatísticas descritivas
print("Descrição - Players")
display(players.describe(include="all").T)

print("Descrição - Lojas")
display(lojas.describe(include="all").T)

print("Descrição - Transações")
display(transacoes.describe(include="all").T)

print("Descrição - Pedestres")
display(pedestres.describe(include="all").T)
```

Bloco 7: KPIs e Análises

```python
# Bloco 7: KPIs e análises

# KPI 1: Número de clientes únicos
clientes_totais = players.shape[0]

# KPI 2: Número de lojas parceiras
lojas_unicas = transacoes['nome_estabelecimento'].unique()

# KPI 3: Volume total transacionado
volume_total = lojas["valor_compra"].sum()

# KPI 4: Valor médio dos cupons
valor_medio_cupom = lojas["valor_cupom"].mean()

# KPI 5: Ticket médio por compra
ticket_medio = lojas["valor_compra"].mean()

# KPI 6: Número de pedestres únicos
pedestres_unicos = pedestres.shape[0]


print("📌 KPIs para CEO & CFO")
print(f"Clientes únicos: {clientes_totais}")
print(f"Lojas parceiras: {len(lojas_unicas)}")
print(f"Volume total transacionado: R$ {volume_total:,.2f}")
print(f"Valor médio dos cupons: R$ {valor_medio_cupom:,.2f}")
print(f"Ticket médio: R$ {ticket_medio:,.2f}")
print(f"Pedestres únicos registrados: {pedestres_unicos}")

```

```python

# Visualizações
fig, axs = plt.subplots(1, 2, figsize=(12,5))

# Distribuição de valores de compra
sns.histplot(lojas["valor_compra"], bins=30, ax=axs[0], kde=True)
axs[0].set_title("Distribuição dos Valores de Compra")

# Relação cupom vs compra
sns.scatterplot(data=lojas.sample(5000), x="valor_cupom", y="valor_compra", alpha=0.5, ax=axs[1])
axs[1].set_title("Cupom vs Valor da Compra")

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