Relatório de Análise de Dados – PicMoney

Etapa 1 – Coleta e Descrição dos Dados

Propósito:
O objetivo desta etapa é carregar os conjuntos de dados no ambiente Google Colab, utilizando bibliotecas de análise de dados e visualização. Aqui garantimos que os dados das diferentes planilhas estejam disponíveis para manipulação.

Descrição:
Foram utilizadas três bases:

- **Transações**: registros de cupons capturados (valores, estabelecimentos e datas).
- **Lojas**: massa de teste com informações de estabelecimentos e valores de compra.
- **Pedestres**: base simulada de potenciais usuários da Av. Paulista.

Código:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
from google.colab import files

sns.set_style('whitegrid')
pd.set_option('display.float_format', lambda x: '%.2f' % x)


print("Selecione as 3 planilhas para upload:")
uploaded = files.upload()

# Carregar planilhas Excel
df_transacoes = pd.read_excel("PicMoney-Base_de_Transa__es_-
_Cupons_Capturados-100000 linhas (1).xlsx")
df_lojas = pd.read_excel("PicMoney-Massa_de_Teste_com_Lojas_e_Valores-10000
linhas (1).xlsx")
```

```
df_pedestres = pd.read_excel("PicMoney-Base_Simulada_-_Pedestres_Av__Paulista-100000 linhas (1).xlsx")

print("Arquivos carregados com sucesso!")
print("Transações:", df_transacoes.shape)
print("Lojas:", df_lojas.shape)
print("Pedestres:", df_pedestres.shape)
```

Etapa 2 – Verificação da Qualidade e Limpeza dos Dados

Propósito:
Nesta etapa garantimos a consistência dos dados, removendo valores nulos, eliminando duplicatas e convertendo colunas para os tipos adequados. Isso evita distorções nas análises.

Descrição:
Foi criada a função limpar_dados que realiza:

1. Identificação e remoção de valores ausentes.
2. Verificação e remoção de duplicados.
3. Conversão de colunas de datas para o formato datetime.
4. Impressão do resultado final com o número de linhas/colunas após a limpeza.

Código:

```
def limpar_dados(df, nome_df):
    """
    Função que limpa os dados de um DataFrame.
    """
    print(f"--- Limpeza de Dados da Tabela: {nome_df} ---")

    print("Valores ausentes antes da limpeza:")
    print(df.isnull().sum())

    df_limpo = df.dropna()
    print("\nDataFrame após a remoção de valores ausentes:")
    print(df_limpo.isnull().sum())
```

```
    linhas_duplicadas = df_limpo.duplicated().sum()
    if linhas_duplicadas > 0:
        print(f"\n{linhas_duplicadas} linhas duplicadas encontradas. Removendo...")
        df_limpo = df_limpo.drop_duplicates()
        print(f"Linhas duplicadas removidas. Total de linhas: {len(df_limpo)}")
    else:
        print("\nNenhuma linha duplicada encontrada.")

    if 'data' in df_limpo.columns:
        df_limpo['data'] = pd.to_datetime(df_limpo['data'], errors='coerce', dayfirst=True)
    if 'data_captura' in df_limpo.columns:
        df_limpo['data_captura'] = pd.to_datetime(df_limpo['data_captura'],
errors='coerce', dayfirst=True)

    print("\nTipos de dados após a conversão:")
    print(df_limpo.dtypes)

    print(f"\nDados após a limpeza na tabela {nome_df}:")
    print(f"Linhas: {df_limpo.shape[0]} | Colunas: {df_limpo.shape[1]}")
    print("-" * 50)

    return df_limpo


df_transacoes_limpo = limpar_dados(df_transacoes.copy(), "Transações")
df_lojas_limpo = limpar_dados(df_lojas.copy(), "Lojas")
df_pedestres_limpo = limpar_dados(df_pedestres.copy(), "Pedestres")
```

Etapa 3 – Exploração dos Dados (EDA)

Propósito:
Nesta fase realizamos a análise exploratória de dados (EDA), investigando padrões, frequências e estatísticas para obter insights iniciais.

3.1 Análise da Tabela de Transações

- Identificação das **Top 10 categorias de estabelecimentos**.
- Estatísticas descritivas do repasse_picmoney.

Código:

```
# Análise Exploratória da Tabela de Transações
print("\n--- Análise da Tabela de Transações ---")

print("\nTop 10 categorias de estabelecimento:")
print(df_transacoes['categoria_estabelecimento'].value_counts().head(10))

plt.figure(figsize=(12, 6))
sns.countplot(y="categoria_estabelecimento",
              data=df_transacoes,

order=df_transacoes['categoria_estabelecimento'].value_counts().head(10).index)
plt.title("Top 10 Categorias de Estabelecimento")
plt.xlabel("Número de Transações")
plt.ylabel("Categoria")
plt.show()

# Estatísticas do repasse
print("\nResumo do repasse:")
print(df_transacoes['repasse_picmoney'].describe())
```

3.2 Análise da Tabela de Lojas

- Estatísticas descritivas de valor_compra e valor_cupom.
- Ranking das **Top 10 lojas por valor acumulado de compra**.
- Gráficos de barras e histograma mostrando a distribuição dos valores.

Código:

```
# Análise Exploratória da Tabela de Lojas
print("\n--- Análise da Tabela de Lojas ---")

# Estatísticas descritivas dos valores de compra e cupom
print("\nEstatísticas de Valor de Compra e Cupom:")
print(df_lojas[['valor_compra', 'valor_cupom']].describe())

# Top 10 lojas por valor de compra
top_lojas_compra = df_lojas.groupby('nome_loja')['valor_compra'].sum().nlargest(10)
print("\nTop 10 Lojas por Valor de Compra:")
print(top_lojas_compra)
```

```
# Gráfico de barras das 10 lojas com maior valor de compra
plt.figure(figsize=(12, 6))
top_lojas_compra.plot(kind='bar', color='skyblue')
plt.title('Top 10 Lojas por Valor de Compra Acumulado')
plt.xlabel('Nome da Loja')
plt.ylabel('Valor de Compra (R$)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Distribuição do valor de compra
plt.figure(figsize=(8, 5))
sns.histplot(df_lojas['valor_compra'], bins=30, kde=True)
plt.title('Distribuição dos Valores de Compra')
plt.xlabel('Valor de Compra (R$)')
plt.ylabel('Frequência')
plt.show()
```

Etapa 4 – KPIs e Indicadores de Negócio

Propósito:
Nesta etapa foram calculados indicadores-chave (KPIs) para avaliar o desempenho do negócio, correlacionando receita com usuários ativos.

Descrição:

- Receita mensal = soma do repasse_picmoney.
- Usuários ativos = número único de celulares.
- Gráfico com barras (receita) e linha (usuários ativos) mostrando evolução temporal.

Código:

```
# Evolução mensal da receita e usuários ativos
df_transacoes['mes_ano'] = df_transacoes['data'].dt.to_period('M')

receita_mensal =
df_transacoes.groupby('mes_ano')['repasse_picmoney'].sum().reset_index()
usuarios_ativos = df_transacoes.groupby('mes_ano')['celular'].nunique().reset_index()
```

```
usuarios_ativos.rename(columns={'celular': 'usuarios_ativos'}, inplace=True)

# Juntar
df_kpi = pd.merge(receita_mensal, usuarios_ativos, on="mes_ano")
df_kpi['mes_ano'] = df_kpi['mes_ano'].astype(str)

# Plot
fig, ax1 = plt.subplots(figsize=(12,6))
ax1.bar(df_kpi['mes_ano'], df_kpi['repasse_picmoney'], color='tab:red', alpha=0.6,
label="Receita")
ax1.set_ylabel("Receita (R$)", color="tab:red")

ax2 = ax1.twinx()
ax2.plot(df_kpi['mes_ano'], df_kpi['usuarios_ativos'], color="tab:blue", marker="o",
label="Usuários Ativos")
ax2.set_ylabel("Usuários Ativos", color="tab:blue")

plt.title("Evolução da Receita e Usuários Ativos")
plt.xticks(rotation=45)
plt.show()
```