Título: Explicação das Etapas — *Preparar os Dados*

Base de referência: *analise_de_dados.ipynb*
Tema: Fluxo de preparação, integração e formatação de dados para análise.

🎯 Introdução

Este relatório descreve detalhadamente as etapas do processo de **preparação de dados** realizadas no arquivo *analise_de_dados.ipynb*.
Cada fase — desde a seleção até a formatação — é explicada de forma técnica e narrativa, demonstrando como o tratamento correto dos dados garante **qualidade, consistência e confiabilidade** para análises posteriores.

O fluxo de preparação analisado segue as seguintes etapas:

Selecionar → Limpar → Derivar → Integrar → Formatar

🧩 1. Selecionar os Dados

O primeiro passo da preparação é **selecionar as fontes de dados** relevantes para o objetivo da análise.
Nesta análise, foram utilizados três arquivos CSV representando diferentes conjuntos de informações:

- **Base de Cadastro dos Players**
- **Base de Transações (Cupons Capturados)**
- **Base Simulada de Pedestres (Av. Paulista)**

📘 Código utilizado:

```
!pip install pandas sqlalchemy chardet unidecode python-dotenv


import pandas as pd
```

print("\nCarregando os arquivos CSV...")

df_cadastro = pd.read_csv('/content/PicMoney-Base_Cadastral_de_Players-10_000 linhas (1).csv', delimiter=';')

df_transacoes = pd.read_csv('/content/PicMoney-Base_de_Transa__es_-_Cupons_Capturados-100000 linhas (1).csv', delimiter=';')

df_pedestres = pd.read_csv('/content/PicMoney-Base_Simulada_-_Pedestres_Av__Paulista-100000 linhas.csv', delimiter=';')

✅ **Objetivo:** Garantir que apenas as bases realmente necessárias sejam carregadas, evitando redundâncias e otimizando a análise.

🧹 2. Limpar / Uniformizar os Dados

Após a seleção das bases, inicia-se a fase de **limpeza e padronização dos dados**.
Essa etapa busca corrigir inconsistências, padronizar formatos e remover duplicatas ou dados ausentes.

📘 **Código utilizado:**

print("\nTratando o DataFrame: Cadastro de Players")

Verificação de valores ausentes

print("\nContagem de valores ausentes por coluna:")

print(df_cadastro.isnull().sum())

Conversão de tipos de dados

print("\nConvertendo tipos de dados...")

df_cadastro['data_nascimento'] = pd.to_datetime(df_cadastro['data_nascimento'], errors='coerce')

Verificação de duplicatas

duplicated_rows = df_cadastro.duplicated().sum()

print(f"\nNúmero de linhas duplicadas: {duplicated_rows}")

🧠 **Análise técnica:**

- **Valores ausentes:** Detectados com isnull(), permitem decidir entre remoção ou preenchimento.
- **Conversão de tipos:** Com to_datetime, assegura coerência nos formatos de data.
- **Remoção de duplicatas:** Evita duplicações que prejudicam cálculos e estatísticas.

📊 **Resultado:** Dados limpos e padronizados, prontos para derivação.

🧮 **3. Derivar Dados**

A derivação consiste em **criar novos dados** a partir dos já existentes, enriquecendo a base e permitindo análises mais completas.
No notebook, foi criada uma tabela de **Merchants (Lojas)** derivada da massa de teste.

📘 **Código utilizado:**

print("\nDerivando Merchants da Massa de Teste...")

merchants = df_massa_teste[['nome_loja', 'tipo_loja', 'local_captura',

'latitude', 'longitude', 'endereco_loja', 'valor_compra']] \

.dropna(subset=['nome_loja']).drop_duplicates()

print(f"Merchants derivados: {merchants.shape}")

🧠 **Análise técnica:**
A derivação cria novas estruturas úteis (ex.: lojas, categorias, localizações), ampliando o escopo da análise e agregando contexto ao comportamento dos usuários.

📊 **Resultado:** Base complementar de *merchants* que pode ser integrada às transações e cadastros.

🔗 4. Integrar os Dados

Com as bases limpas e derivadas, é preciso **integrá-las** para formar um conjunto de dados consolidado.
Essa integração garante que as informações de diferentes fontes possam ser cruzadas corretamente.

📘 **Código utilizado:**

print("\nIntegrando dados...")

Exemplo: Juntar Cadastro com Transações

df_integrado = pd.merge(df_cadastro, df_transacoes, left_on='id_cliente', right_on='id_cliente', how='inner')

🧠 **Análise técnica:**

- A função merge() combina dados com base em uma chave comum (id_cliente).
- O parâmetro how='inner' mantém apenas registros que existem em ambas as bases.
- Isso assegura **consistência relacional** e elimina registros sem correspondência.

📊 **Resultado:** Dataset consolidado, unindo informações de usuários e transações.

🗂️ 5. Formatar os Dados

A última etapa é **formatar e refinar** os dados, mantendo apenas as colunas relevantes e organizando a estrutura final do DataFrame.

📘 **Código utilizado:**

```
import matplotlib.pyplot as plt

import seaborn as sns


sns.set_style("whitegrid")

plt.rcParams['figure.figsize'] = (12, 7)


print("\nSelecionando dados relevantes...")


df_final = df_integrado[['celular', 'idade', 'sexo', 'nome_loja',

                         'valor_cupom', 'repasse_picmoney']]


print(f"DataFrame final selecionado: {df_final.shape}")

print("\nAmostra dos dados finais:")

print(df_final.head())
```

🧠 **Análise técnica:**

- **Seleção de colunas úteis:** Foco apenas em informações relevantes para a análise.
- **Formatação final:** Estrutura organizada para exportação, visualização e modelagem.
- **Verificação visual:** A função head() confirma se os dados finais estão consistentes.

📊 **Resultado:** Dados prontos para análise estatística, visualização ou machine learning.

📜 Conclusão

O processo de **preparação de dados** é um pilar essencial em qualquer projeto analítico.
Cada etapa é interdependente e contribui para garantir que o resultado final seja confiável e interpretável.

Etapa	Objetivo	Resultado Esperado
Selecionar	Escolher bases relevantes	Redução de ruído
Limpar	Corrigir inconsistências	Dados confiáveis
Derivar	Criar novas informações	Dados enriquecidos
Integrar	Combinar fontes distintas	Base unificada
Formatar	Filtrar e organizar	Dados prontos para análise

🔁 **Fluxo final:**

Selecionar → Limpar → Derivar → Integrar → Formatar → Analisar

💡 **Síntese:**
Essas etapas garantem a qualidade do pipeline de dados, permitindo extrair **insights consistentes, mensuráveis e úteis** para decisões estratégicas.