FECAP - CIÊNCIA DA COMPUTAÇÃO

Título: Entrega 2

Autor: GRUPO 04 – Barões do Cupom

Disciplina: Projeto Interdisciplinar – Ciência de Dados

Ano: 2025

Resumo

Este documento formaliza as análises e transformações realizadas no notebook ENTREGA_2_PI.ipynb. O objetivo foi **preparar os dados** para posterior análise/ modelagem: seleção, limpeza/uniformização, derivação, integração e formatação dos dados. Cada etapa é descrita detalhadamente e acompanhada de justificativas técnicas, descrição das operações realizadas no código Python (pandas) e recomendações para qualidade e reprodutibilidade.

Sumário

1. Objetivo: PREPARAR OS DADOS

Descrição: centralizar e transformar as bases de origem em um único *DataFrame* pronto para análise exploratória e modelagem. O notebook carrega três arquivos CSV (base_players.csv, lojas_valores.csv, transacoes_cupons.csv) para DataFrames do pandas, inspeciona estrutura, trata problemas de qualidade, deriva colunas úteis (por exemplo, concatenação de data e hora, idade a partir de data de nascimento), integra as tabelas através de chaves e formata os campos para um padrão consistente.

Justificativa técnica: limpeza e padronização evitam vieses e erros ao calcular estatísticas ou treinar modelos; a integração permite relacionar transações com perfis de jogadores e atributos das lojas.

2. Selecionar os Dados

2.1 Arquivos carregados

No notebook, os arquivos são lidos com pandas.read_csv. Exemplo técnico:

- df_base_jogadores = pd.read_csv('/content/base_players.csv', sep=';')
- df_lojas_valores = pd.read_csv('/content/lojas_valores.csv', sep=';')
- df_transacoes = pd.read_csv('/content/transacoes_cupons.csv', sep=';')

3. Limpar / Uniformizar os Dados

3.1 Inspeção inicial

O notebook inicia com inspeção das estruturas (.info(), .head(), .describe()), o que é essencial para detectar tipos, valores nulos e colunas com formato incorreto.

Operações observadas no código:

- df.info() e display(df.head()) para cada DataFrame.

3.2 Tratamento de valores ausentes

Técnicas aplicadas no notebook (descritas):

- Identificação de colunas com isnull().sum().
- Estratégias adotadas:
    - Preencher campos de localização ou texto ausente com 'Não Informado' quando a ausência for aceitável e o preenchimento fizer sentido sem viés nas análises.
    - Para colunas numéricas críticas (ex.: valores monetários), decisão entre fillna(0) ou excluir linhas depende do contexto — o notebook documenta justificativas para as escolhas feitas.

Justificativa: evitar que operações numéricas resultem em NaN e que agregações sejam enviesadas.

3.3 Conversão de tipos e padronização de texto

Conversões observadas:

- Conversão de colunas de datas (data_nascimento, data_captura, data) com pd.to_datetime(col, dayfirst=True, errors='coerce') onde aplicável.
- Conversão de latitude/longitude para numérico: pd.to_numeric(col, errors='coerce') após limpeza (remover vírgulas, espaços, símbolos monetários).
- Padronização de texto: .str.strip() para remover espaços, .str.lower() para uniformizar caixa, e .str.replace() para remover caracteres indesejados.

3.4 Normalização de valores monetários e separadores decimais

O notebook contém rotinas para limpar campos financeiros (ex.: remover R$, substituir , por .) e converter para float. Exemplo típico:

```
col = col.str.replace('R\$', '').str.replace('.', '').str.replace(',', '.').astype(float)
```

4. Derivar Dados

4.1 Criação de colunas derivadas observadas

O notebook descreve e implementa as seguintes derivadas comuns:

- **Coluna Datetime Combinada:** combinar data + hora em data_hora com pd.to_datetime(df['data'] + ' ' + df['hora']). Isso facilita agrupamentos por dia/hora, cálculo de intervalos e séries temporais.
- **Hora da transação:** extrair hora em hora_transacao = df['data_hora'].dt.hour ou .dt.time.
- **Idade do jogador:** calcular idade aproximada a partir de data_nascimento usando relativedelta or ((hoje - data_nascimento).days // 365).
- **Indicadores binários / flags:** por exemplo, flag de transação com cupom, tem_cupom = df['valor_cupom'].notnull().

4.2 Justificativa e impacto

Derivações fornecem features prontas para análise estatística e modelos de machine learning, além de simplificar filtros e agregações.

5. Integrar os Dados

5.1 Chaves e estratégia de junção

O notebook integra as três tabelas usando chaves observadas como celular e numero_celular — essa é uma junção do tipo *left join* para preservar linhas do conjunto principal (por exemplo, transações) e anexar atributos dos jogadores e das lojas:

- merged_df = df_transacoes.merge(df_base_jogadores, how='left', left_on='celular', right_on='celular')
- merged_df = merged_df.merge(df_lojas_valores, how='left', left_on='numero_loja', right_on='numero_loja')

Observações críticas:

- *Chave única:* verificar que a chave de junção identifica registros de forma consistente (consistência de formato, falta de duplicatas). Se celular não for único, a junção pode duplicar linhas e inflar contagens.
- *Redundância de colunas:* após a junção aparecem sufixos _x e _y. O notebook renomeou colunas para nomes significativos e removeu colunas redundantes.

5.2 Renomeação e limpeza pós-join

O notebook mostra operações para renomear colunas com sufixos e descartar colunas redundantes (por exemplo, celular_y renomeada para celular_transacao e uma coluna redundante descartada). Também reordena colunas para melhor legibilidade.

Exemplo (trecho observado no notebook):

```
df_mesclado = df_mesclado.rename(columns={
  'tipo_cupom_x': 'tipo_cupom_transacao',
  'valor_cupom_x': 'valor_cupom_transacao',
  'tipo_cupom_y': 'tipo_cupom_loja',
  'valor_cupom_y': 'valor_cupom_loja',
})
```

```
# remover coluna redundante
df_mesclado = df_mesclado.drop(columns=['numero_celular'])
```

Ponto de verificação dupla: após cada junção executar merged_df.shape, merged_df.duplicated(subset=[keys]).sum() e merged_df.isnull().sum() para garantir que a integração ocorreu conforme esperado.

6. Formatar os Dados

6.1 Ordenação e tipos finais

Objetivo: preparar merged_df com tipos corretos para exportação e análise. As ações incluem:

- Converter colunas categóricas para category quando aplicável para economia de memória.
- Converter datas para datetime64[ns].
- Garantir que colunas numéricas tenham float ou int apropriados.
- Reordenar colunas em sequência lógica (identificadores, temporal, geográfico, financeiros, flags).

6.2 Exportação

O notebook indica que o dataframe final pode ser exportado com to_csv('merged_df_preparado.csv', index=False) para ser utilizado em etapas posteriores.

7. Conclusão

Resumo das ações realizadas:

- Seleção dos três arquivos origem e carregamento em pandas.
- Inspeção inicial e identificação de problemas estruturais e de qualidade.
- Tratamento de valores ausentes, conversão de tipos, limpeza de textos e normalização de valores monetários.
- Derivação de colunas temporais e indicadores relevantes para análise.
- Integração das tabelas com junções *left*, renomeação de colunas e eliminação de redundâncias.
- Formatação final do DataFrame e preparação para exportação.

8. Apêndice: Trechos de Código Relevantes

Observação: este apêndice contém trechos do notebook utilizados para documentar cada etapa. Para reprodução completa, ver ENTREGA_2_PI.ipynb.

A.1 Leitura dos arquivos

```
import pandas as pd

df_base_jogadores = pd.read_csv('/content/base_players.csv', sep=';')
df_lojas_valores = pd.read_csv('/content/lojas_valores.csv', sep=';')
df_transacoes = pd.read_csv('/content/transacoes_cupons.csv', sep=';')
```

A.2 Inspeção inicial

```
print(df_base_jogadores.info())
display(df_base_jogadores.head())
```

A.3 Limpeza e padronização (exemplos)

```
# converter datas
df_base_jogadores['data_nascimento'] =
pd.to_datetime(df_base_jogadores['data_nascimento'], dayfirst=True, errors='coerce')

# padronizar texto
```

```
df_base_jogadores['nome'] = df_base_jogadores['nome'].str.strip().str.lower()

# limpar valores monetários
df_lojas_valores['preco'] = df_lojas_valores['preco'].astype(str).str.replace('R\$', '', regex=True).str.replace('.', '', regex=False).str.replace(',', '.', regex=False).astype(float)
```

A.4 Derivação de colunas

```
df_transacoes['data_hora'] = pd.to_datetime(df_transacoes['data'] + ' ' + df_transacoes['hora'], dayfirst=True, errors='coerce')
df_transacoes['hora_transacao'] = df_transacoes['data_hora'].dt.hour
```

A.5 Integração

```
merged_df = df_transacoes.merge(df_base_jogadores, how='left', left_on='celular', right_on='celular')
merged_df = merged_df.merge(df_lojas_valores, how='left', left_on='numero_loja', right_on='numero_loja')

# renomear e limpar sufixos
merged_df = merged_df.rename(columns={
    'tipo_cupom_x': 'tipo_cupom_transacao',
    'valor_cupom_x': 'valor_cupom_transacao',
    'tipo_cupom_y': 'tipo_cupom_loja',
    'valor_cupom_y': 'valor_cupom_loja',
})

# descartar colunas redundantes
merged_df = merged_df.drop(columns=['numero_celular'])
```