Documentação — Pipeline InovaTech (merge_final.csv)

Equipe: InovaTech
Objetivo: transformar o arquivo `merge_final.csv` em dados prontos para dashboards/KPIs, com um fluxo profissional: ingestão → curadoria → qualidade → métricas → integração → formatação final.

Visão geral do processo

1. **Preparar os Dados** – ingestão e exploração inicial (estrutura, nulos, amostras).
2. **Selecionar os Dados** – curadoria de colunas com valor para o negócio.
3. **Limpar / Uniformizar** – nomes, datas, números, duplicatas e nulos.
4. **Derivar Dados** – métricas novas (tempo total, faixas de ticket, idade, eficiência).
5. **Integrar os Dados** – modelo estrela: dimensões (cliente, loja, calendário) + fato.
6. **Formatar os Dados** – organização de colunas, arredondamento, dicionário de dados.

Principais saídas:
`merge_inovatech_selected.csv`, `merge_inovatech_clean.csv`, `merge_inovatech_derived.csv`, `dim_customer_inovatech.csv`, `dim_store_inovatech.csv`, `dim_calendar_inovatech.csv`, `fact_orders_inovatech.csv`, `fact_orders_integrated_inovatech.csv`, `inovatech_prepared.csv`, `inovatech_data_dictionary.csv`.

---

1) Preparar os Dados (ingestão InovaTech)

O que fizemos

- **Leitura** do `merge_final.csv` usando `sep=';'` (padrão do arquivo).

- **Exploração**: `shape`, lista de colunas, `head(5)`, contagem de nulos por coluna e `dtypes`.
- **Backup** do bruto (`merge_final_raw_backup.csv`) para rastreabilidade.

Como o código funciona (resumo):

- `pd.read_csv('merge_final.csv', sep=';', dtype=str, engine='python')`: garante a separação correta e mantém tudo como string na ingestão.
- Impressões e `display` para observar amostra e qualidade.
- `df_raw.to_csv('merge_final_raw_backup.csv')`: snapshot do bruto.

Por que isso importa: valida a leitura, confirma o separador e identifica problemas (nulos, formatos) antes de seguir.

---

2) Selecionar os Dados (curadoria analítica)

O que fizemos

- Escolhemos **colunas alinhadas aos KPIs**:
  - **IDs**: `id`, `storeId`, `companyId`, `customerId`
  - **Datas**: `createdAt`, `updatedAt`, `scheduledAt`, `sendAt`, `dateOfBirth`
  - **Métricas**: `totalAmount`, `preparationTime`, `takeOutTimeInSeconds`
  - **Cliente**: `name`, `gender`, `phone`, `email`, `status`
- Selecionamos **apenas as colunas existentes** no CSV (robustez).
- Exportamos `merge_inovatech_selected.csv`.

Como funciona:

- Montamos `colunas_chave` e filtramos com `[c for c in colunas if c in df_raw.columns]`.

- `df_selected.to_csv('merge_inovatech_selected.csv')`.

Racional: reduz ruído e foca no que alimenta os dashboards (valor, tempo, cliente, datas e chaves).

---

3) Limpar / Uniformizar (qualidade e consistência)

O que fizemos

- **Padronização de nomes** para `snake_case`.
- **Normalização de strings** (trim, transformar `'nan'`, `''`, `None` em `NaN`).
- **Remoção de duplicatas** por `id` (quando existe).
- **Conversão de datas** por heurística: colunas que terminam com `_at` ou contêm `date`.
- **Conversão numérica** de `total_amount`, `preparation_time`, `take_out_time_in_seconds`:
  remove milhar (`.`) e converte vírgula decimal para ponto.
- **Preenchimento de nulos** em métricas com `0` (onde faz sentido).
- Exportamos `merge_inovatech_clean.csv`.

Como funciona (pontos-chave do código):

- Função `to_snake_case` transforma `camelCase` → `snake_case`.
- Loop em colunas para `strip()` e mapear strings inválidas → `NaN`.
- `drop_duplicates(subset=['id'])` quando a chave existe.
- `pd.to_datetime(..., errors='coerce')` nas colunas de data.
- Numéricos: `str.replace('.', '')` (milhar) e `str.replace(',', '.')` (decimal) + `to_numeric`.

Por que isso importa: garante **consistência técnica**; sem isso, KPIs e joins quebram.

---

4) Derivar Dados (inteligência de negócio)

O que criamos e por quê

- `service_time_total` = `preparation_time` + `take_out_time_in_seconds`
  → mede **eficiência operacional** (tempo total por pedido).

- `ticket_categoria` (Baixo/Médio/Alto) por **tercis** de `total_amount`
  → facilita análise financeira segmentada.

- `idade` calculada a partir de `date_of_birth`
  → habilita segmentações demográficas.

- `valor_por_segundo` = `total_amount` / `service_time_total`
  → KPI de **produtividade financeira do atendimento**.

Como funciona:

- `quantile([0.33, 0.66])` para definir cortes automáticos do bucket.

- Conversão de `date_of_birth` para datetime e cálculo de idade via diferença de dias.

- Proteções para divisão por zero e valores nulos.

Saída: `merge_inovatech_derived.csv`.

---

5) Integrar os Dados (modelo estrela InovaTech)

O que fizemos

- **`dim_customer_inovatech.csv`**: 1 linha por `customer_id` (nome, gênero, contato, status, idade).

- **`dim_store_inovatech.csv`**: 1 por `store_id` (e `company_id` quando existir).

- **`dim_calendar_inovatech.csv`**: chaves de calendário YYYYMMDD + data.

- **`fact_orders_inovatech.csv`**: fato com IDs, FKs (calendário/cliente/loja), métricas e atributos degenerados (ex.: `ticket_categoria`).

- **`fact_orders_integrated_inovatech.csv`**: *join* (fato + dimensões) para exploração/validação rápida.

Como funciona:

- Dimensões: `drop_duplicates` pelas chaves (`customer_id`, `store_id`) e normalização das datas para calendário.

- FKs de calendário: `dt.strftime('%Y%m%d')`.

- Fato: seleciona **IDs**, **FKs**, **métricas** e alguns **atributos de negócio** úteis.

- *Joins* left para montar `fact_orders_integrated_inovatech.csv`.

Por que isso importa: separa **atributos estáveis** (dimensões) de **ocorrências** (fatos) — padrão de BI escalável.

---

6) Formatar os Dados (consumo final)

O que fizemos

- **Normalização final** de datas e numéricos (garantia de tipo) e **arredondamento**.

- **Organização** das colunas por grupos:
  IDs/FKs → **Datas** → **Métricas** → **Categóricas** (melhora leitura e consumo).

- **Ordenação por recência** quando disponíveis (`updated_at`, `created_at` etc.).

- **Dicionário de dados automático**: `coluna`, `tipo`, `grupo`, `nulos` → `inovatech_data_dictionary.csv`.

- **Hotfix para colunas duplicadas** após joins: renomeia com sufixo `__dupN` e só então gera o dicionário.

- Saída final: `inovatech_prepared.csv`.

Como funciona:

- Identifica datas por heurística e converte; identifica métricas por nome e converte.
- Arredonda `total_amount` (2 casas) e outras métricas (3 casas, ajustável).
- Reclassifica colunas nos quatro grupos e reordena.
- Hotfix de duplicadas: torna os nomes **únicos** antes de iterar `dtype`.

Decisões-chave e racional

- **Separador `;`** confirmado na inspeção — evita colunas “coladas”.
- **Heurística de datas** (sufixo `_at` ou contém `date`) cobre `created_at`, `updated_at`, `scheduled_at`, `send_at`, `date_of_birth`.
- **Conversão numérica robusta**: remove milhar e padroniza decimal → evita que números virem texto.
- **Buckets por tercis**: não depende de faixas fixas; adapta ao comportamento real da base.
- **Modelo estrela**: reduz redundância, acelera consultas e facilita manutenção de dashboards.
- **Dicionário automático**: dá transparência e rastreabilidade para quem consome o dado.

Troubleshooting (erros comuns)

- **`AttributeError: 'DataFrame' object has no attribute 'dtype'`**
  Causa: colunas **duplicadas** após join → a seleção por nome retorna várias colunas.
  Solução: hotfix que torna os nomes **únicos** com sufixos `__dup1`, `__dup2` antes de gerar o dicionário.
- **Datas virando NaT**
  Causa: formatos inválidos; usamos `errors='coerce'`.
  Solução: rastrear a origem; se necessário, tratar padrões específicos (ex.: `dd/mm/yyyy`).

- **Números com vírgula/ponto “misturados”**
  Solução: normalização aplicada (remove milhar com ponto e troca vírgula por ponto) + `to_numeric`.

KPIs e uso em dashboards (InovaTech)

- **Tempo total de atendimento** (`service_time_total`).
- **Ticket médio e distribuição por faixa** (`total_amount`, `ticket_categoria`).
- **Produtividade por tempo** (`valor_por_segundo`).
- **Perfil do cliente** (`idade`, `gender`).
- **Volume e evolução temporal** (dimensão calendário + datas).

Esses indicadores permitem monitorar **eficiência operacional**, **valor gerado**, **sazonalidade** e **segmentos de cliente**.

Execução (passo a passo)

1. Suba `merge_final.csv` no Colab.
2. Rode as **6 células** (versão InovaTech) **na ordem**.
3. Verifique os **arquivos de saída** após cada etapa.
4. Se aparecer erro de duplicadas na etapa 6, aplique o **hotfix** (renomeação `__dupN`) e gere o dicionário novamente.
5. Consuma **`inovatech_prepared.csv`** (dataset final) e **`inovatech_data_dictionary.csv`** no seu BI.