Relatório Técnico — Progresso do Pipeline (Bronze e Silver)

1) Resumo executivo

- **Objetivo:** montar um pipeline reprodutível para um recomendador de filmes utilizando movies.csv e ratings.csv, **neste momento restrito às camadas Bronze (ingestão/armazenamento bruto) e Silver (transformação/limpeza)**.
- **Status:**
    - **Bronze:** implementado e validado (Parquet colunar para movies e ratings).
    - **Silver:** implementado e validado (tipagem, normalização, limpeza, parsing de campos JSON-like, coerção de tipos e filtros).

2) Escopo e fontes

- **Escopo desta etapa: apenas Bronze e Silver**. Gold será implementada futuramente.
- **Fontes:**
    - movies.csv com genres, belongs_to_collection, production_*, spoken_languages, etc.
    - ratings.csv com userId, movieId, rating, timestamp.

3) Arquitetura de dados (resumo)

- **Bronze:** ingestão batch (com opção de micro-lotes para simular streaming), gravação **Parquet** em bronze, preservando estrutura e tipos originais quando possível.
- **Silver:** transformações:
    - **Tipagem:** datas (release_date), numéricos (budget, revenue, runtime, popularity, vote_*).
    - **Booleanos:** adult e video via função robusta to_bool.
    - **Parsing JSON-like:** campos com **aspas simples** e estruturas tipo Python tratados por parse_json_flexible (tenta json.loads e cai para ast.literal_eval), expostos como **listas/dicionários reais**:
        - genres → genres_list (lista de dicts com id, name);
        - belongs_to_collection → collection_* (id, name, poster/backdrop);

        - production_companies, production_countries, spoken_languages → listas de dicts.
    - **Ratings:** coerção de userId/movieId para Int64, rating numérico com filtro **[0,5]**, timestamp → datetime.

4) Implementação (arquivos e principais decisões)

- **src/ingest.py**
    - Lê dados/raw/movies.csv e dados/raw/ratings.csv.
    - Suporta **chunksize** opcional em ratings para simular “micro-lotes”.
    - Gera: bronze/movies_raw.parquet e bronze/ratings_raw.parquet.
- **src/utils.py**
    - _is_blank: robusto para “nan/none/null/…”.
    - parse_json_flexible: tenta JSON; fallback para ast.literal_eval (aspas simples).
    - parse_list, parse_dict: garantem retorno do tipo esperado.
    - to_bool: converte variantes comuns de booleanos string/numéricos.
- **src/transform.py**
    - **Movies (Silver):**
        - Tipagem numérica/datas/booleanos; normalização de original_language.
        - genres → genres_list; belongs_to_collection → collection_id/name/poster/backdrop.
        - production_* e spoken_languages parseados para listas.
        - Seleção defensiva de colunas e **deduplicação por id**.
        - Saída: silver/movies_clean.parquet.
    - **Ratings (Silver):**
        - userId/movieId coeridos para Int64; rating coerido para numérico; timestamp → ts.
        - Filtro de inválidos (nulos e fora de [0,5]).
        - Saída: silver/ratings_clean.parquet.

5) Validações e qualidade de dados

- **Inspeção**: head(), dtypes, **relatório de nulos** por coluna e contagens de distintos.
- **Consistência de chaves:** padronização **pré-merge** (movies_clean.id e ratings_clean.movieId → Int64).
- **Robustez**: tratamento de strings JSON com aspas simples, URLs sem aspas, e casos vazios.

6) Notebooks e visualizações

6.1. Notebook 01_ingest_validacao.ipynb (Bronze)

- **Gráfico 1 — Distribuição de atividade por usuário**
  Agregação: ratings_raw.groupby('userId').size() → histograma com 50 *bins* (entende concentração de poder de avaliação).
- **Gráfico 2 — Volume de ratings por mês**
  Conversão de timestamp (segundos) → datetime; agregação para Period('M') e série temporal (evolução de uso/entrada de dados).

6.2. Notebook 02_transform_exploracao.ipynb (Silver)

- **Gráfico 1 — Média de rating por gênero (com limiar adaptativo e fallback)**
  Explosão de genres_list → genre; merge com ratings; agregação por genre → média e contagem.
  - **Limiar adaptativo**: percentil 75 de n (piso 10).
  - **Fallback**: top 15 por contagem quando não há gêneros acima do limiar.
  - **Motivo:** evita **IndexError** por série vazia em amostras pequenas.
- **Gráfico 2 — Média de rating por faixa de runtime (buckets)**
  Coerção de id/movieId para Int64; runtime numérico; *buckets* [≤60, 60–90, 90–120, 120–150, >150]; merge e agregação por faixa.
  - **Guards:** mensagens claras se não houver interseção suficiente (sem quebrar o notebook).

Observação: todos os gráficos respeitam as restrições da apresentação (um gráfico por *plot* e **sem** definir cores).

7) Incidentes e correções aplicadas

1. **Mermaid parse error** (labels com aspas) → **substituição por texto corrido** (documento fluido) e, quando necessário, rótulos sem aspas/caracteres especiais.
2. **IndexError em gráfico de gêneros** (conjunto vazio após filtro min_n=100) → **limiar adaptativo** + **fallback por contagem**.
3. **ValueError: merge Int64 × object** no gráfico de runtime → **coerção explícita**: pd.to_numeric(...).astype('Int64') tanto em movieId quanto em id antes do merge.

8) Reprodutibilidade (execução local)

1. Criar venv e instalar dependências (requirements.txt).
2. Colocar movies.csv e ratings.csv em dados/raw/.
3. Executar:
   - python src/ingest.py → **Bronze**

    - python src/transform.py → **Silver**
4. Abrir e executar notebooks:
    - 01_ingest_validacao.ipynb (verificações + 2 gráficos Bronze)
    - 02_transform_exploracao.ipynb (verificações + 2 gráficos Silver)

9) Próximos passos

- Consolidar **testes rápidos** para utils.py (casos com aspas simples, listas vazias, dicionários, booleanos variados).
- Enriquecer validações Silver (ex.: *outliers* de runtime, faixas etárias com adult, normalização de idiomas).
- Ampliar visualizações Silver:
    - **Distribuição de vote_average por idioma/ano**;
    - **Evolução de runtime médio por década**;
    - **Top N companhias por volume de títulos** (se production_companies for necessário).
- Planejar **entrada na camada Gold** com métricas e avaliações.