Universidade Federal do Amazonas
Instituto de Computação Computação
Bancos de Dados I – 2025/02
Professor: Altigran Soares da Silva <alti@icomp.ufam.edu.br>
Monitores:

- **Júnio da Silva Freitas <jusf@icomp.ufam.edu.br>**
- **Eduardo Alves da Silva <eduardo.silva@icomp.ufam.edu.br>**
- **Manoel Victor Florencio de Souza <manoel.souza@icomp.ufam.edu.br>**

Trabalho Prático I – 28/08/2025
Data da Entrega: 25/09/2025 até 23:59

1. Apresentação

Objetivo deste trabalho prático é projetar e implementar um banco de dados sobre produtos vendidos em uma loja de comércio eletrônico, incluindo avaliações e comentários de usuários sobre estes produtos. O trabalho consiste na criação de um Banco de Dados Relacional contendo dados sobre compras de produtos e elaboração de um *Dashboard, um* painel para monitoramento dos dados de compra, gerando uma série de relatórios. Os dados para o banco de dados serão fornecidos de um arquivo de entrada que será indicado aos alunos.

O trabalho deverá ser desenvolvido em **trios**, individualmente no caso de alunos do PPGI.

2. Sobre o Esquema do Banco de Dados

O **esquema** deve seguir o **modelo relacional**, projetado pela técnica **ascendente (bottom-up)**, observando **uma** das formas normais de alto nível (BCNF, 3FN ou 4FN). **O estudo dessas técnicas faz parte do trabalho**; recomenda-se consultar as referências [1] e [2] ou material equivalente.

O PDF (tp1_3.1.pdf) deve justificar escolhas de decomposição e restrições, indicando chaves, FKs, unicidade, domínios e regras derivadas do arquivo de entrada.

3. Sobre Dashboard

Todas as consultas abaixo **devem ser implementadas em SQL puro** (sem ORM). O script deve apenas executar as queries e formatar a saída.

1. **Dado um produto**, listar os **5 comentários mais úteis e com maior avaliação** e os **5 comentários mais úteis e com menor avaliação**.
2. **Dado um produto**, listar os **produtos similares** com **maiores vendas** (melhor **salesrank**) do que ele.
3. **Dado um produto**, mostrar a **evolução diária** das **médias de avaliação** ao longo do período coberto no arquivo.
4. Listar os **10 produtos líderes de venda** em **cada grupo** de produtos.

5. Listar os **10 produtos** com a **maior média de avaliações úteis positivas** por produto.

6. Listar as **5 categorias** com a **maior média de avaliações úteis positivas** por produto.
7. Listar os **10 clientes** que mais fizeram comentários por **grupo** de produto.

4. Arquivo de Entrada

O arquivo de entrada de onde serão extraídos os dados de entrada será o *"Amazon product co-purchasing network metadata"* que faz parte do Stanford Network Analysis Project (SNAP). Os dados foram coletados em 2006 do site Amazon.com e contém informações sobre produtos e comentários de clientes sobre 548.552 produtos diferentes (livros, CDs de música, DVDs e fitas de vídeo VHS). Para cada produto, a seguinte informação está disponível:

- Título
- Posição no ranking de vendas (Salesrank)
- Lista de produtos ``similares'' (que foram adquiridos junto com o produto)
- Informação de categorização do produto – Categorias e subcategorias ao qual o produto pertence
- Comentários sobre os produtos:
    - Informação data, id do cliente, classificação, número de votos, o número de pessoas que acharam a avaliação útil

Um trecho do arquivo é apresentado abaixo:

```
Id:   15
ASIN: 1559362022
  title: Wake Up and Smell the Coffee
  group: Book
  salesrank: 518927
  similar: 5  1559360968  1559361247  1559360828  1559361018  0743214552
  categories: 2
   |Books[283155]|Subjects[1000]|Literature & Fiction[17]|Drama[2159]|United States[2160]
   |Books[283155]|Subjects[1000]|Arts & Photography[1]|Performing Arts[521000]|Theater[2154]
reviews: total: 8  downloaded: 8  avg rating: 4
    2002-5-13  cutomer: A2IGOA66Y6O8TQ  rating: 5  votes:   3  helpful:   2
    2002-6-17  cutomer: A2OIN4AUH84KNE  rating: 5  votes:   2  helpful:   1
    2003-1-2   cutomer: A2HN382JNT1CIU  rating: 1  votes:   6  helpful:   1
    2003-6-27  cutomer: A39QMV9ZKRJXO5  rating: 4  votes:   1  helpful:   1
    2004-2-17  cutomer:  AUUVMSTQ1TXDI  rating: 1  votes:   2  helpful:   0
    2004-10-13 cutomer:  A5XYF0Z3UH4HB  rating: 5  votes:   1  helpful:   1
```

2. Ambiente de Desenvolvimento e Execução

Toda a solução **deve** ser empacotada em contêineres Docker e executada via `docker-compose`, atendendo aos requisitos:

- **Linguagem**: Python 3.11+ (em contêiner próprio).
- **SGBD**: PostgreSQL 15+ (em contêiner próprio).
- **Comunicação**: o contêiner da aplicação Python deve conectar-se ao PostgreSQL via rede interna do `docker-compose`.

- **Sem dependências locais** além de Docker e Docker Compose.
- **Reprodutibilidade**: um único comando deve preparar o banco, carregar dados e permitir executar o Dashboard.

2.1. Serviços obrigatórios no `docker-compose.yml`

- `db`: PostgreSQL com:
  - usuário: `postgres`
  - senha: `postgres`
  - database: `ecommerce`
  - porta mapeada: `5432`
  - volume persistente para dados
  - `healthcheck` habilitado

- `app`: aplicação Python com:
  - build via `Dockerfile` (instalação de dependências pelo `requirements.txt`)
  - diretório `/app` como `WORKDIR`
  - acesso ao arquivo de entrada via volume montado `./data:/data`
  - `depends_on` com `condition: service_healthy` para `db`

Observação: não usar ORMs; o acesso ao PostgreSQL deve ser feito por **SQL direto**.

2.2. Estrutura do Repositório (obrigatória)

```
tp1/
├── docker-compose.yml
├── Dockerfile
├── requirements.txt
├── src/
│   ├── tp1_3.2.py        # criação do esquema + carga (ETL)
│   ├── tp1_3.3.py        # consultas do Dashboard (SQL)
│   ├── db.py             # utilitários de conexão SQL (opcional, sem ORM)
│   └── utils.py          # parsing/validações (opcional)
├── sql/
│   └── schema.sql        # opcional: DDL em SQL puro, se preferir separar
├── data/
│   └── snap_amazon.txt   # arquivo de entrada (ou README com instruções de download)
├── docs/
│   ├── tp1_3.1.pdf       # documentação (diagrama + dicionário de dados)
│   └── esquema.png       # imagem do diagrama (se referenciada no PDF)
```

```
├── Makefile              # opcional (atalhos docker)
└── README.md             # como rodar (passo a passo)
```

3. O que entregar

1. **Documentação (tp1_3.1.pdf)**
   - Diagrama correspondente ao **esquema** do **banco de dados relacional.**
   - Dicionário de dados: descrição de cada relação, atributos, chaves, restrições de integridade (referencial e demais).
   - Observância das diretrizes da Seção 7 (formas normais e projeto).
   - Entregar **um único PDF** em `docs/tp1_3.1.pdf`.

2. **Script de carga (tp1_3.2.py)**
   - Em `src/tp1_3.2.py`.
   - Responsável por: (i) criar o **esquema** no PostgreSQL e (ii) **povoar** as relações lendo o arquivo de entrada (`/data/snap_amazon.txt`).
   - Deve **retornar código de saída 0** em sucesso; diferentes de 0 em erro (para avaliação automática).
   - Pode opcionalmente usar `sql/schema.sql` para DDL.

3. **Script de consultas do Dashboard (tp1_3.3.py)**
   - Em `src/tp1_3.3.py`.
   - Executa **todas** as consultas SQL definidas na Seção 6, imprimindo saídas legíveis em **STDOUT** (e, se desejarem, salvando CSVs em `/app/out`).
   - Deve aceitar **parâmetros** de execução (ver Seção 5) e terminar com **código 0** em sucesso.

4. **Conteinerização**
   - `Dockerfile` funcional (construindo a imagem da app).
   - `docker-compose.yml` orquestrando `db` (PostgreSQL) e `app` (Python).
   - `README.md` com **um comando** de ponta-a-ponta (Seção 5).

> **Atenção**: os scripts devem estar prontos para serem executados **sem erros** dentro do contêiner. Nada deve depender do ambiente local.

4. Como executar (padrão exigido)

No `README.md` inclua exatamente estes passos (ou equivalentes):

```
# 1) Construir e subir os serviços
docker compose up -d --build

# 2) (Opcional) conferir saúde do PostgreSQL
```

```
docker compose ps

# 3) Criar esquema e carregar dados
docker compose run --rm app python src/tp1_3.2.py \
  --db-host db --db-port 5432 --db-name ecommerce --db-user postgres --db-pass postgres \
  --input /data/snap_amazon.txt

# 4) Executar o Dashboard (todas as consultas)
docker compose run --rm app python src/tp1_3.3.py \
  --db-host db --db-port 5432 --db-name ecommerce --db-user postgres --db-pass postgres \
  --output /app/out
```

4.1. Parâmetros mínimos exigidos

Ambos os scripts (`tp1_3.2.py` e `tp1_3.3.py`) **devem aceitar**:

- `--db-host`, `--db-port`, `--db-name`, `--db-user`, `--db-pass`
- Para carga: `--input` (caminho do arquivo SNAP dentro do contêiner)
- Para dashboard:
  - `--product-asin` (usado nas consultas que exigem um produto específico)
  - `--output` (diretório para salvar CSVs/relatórios, opcional, padrão `/app/out`)

Padrão de saída: imprimir no terminal títulos das consultas, contagens e tabelas resumidas; ao salvar CSVs, nomear arquivos de forma autoexplicativa (ex.: `q1_top5_reviews_pos.csv`, `q4_top10_sales_by_group.csv`).

5. Requisitos técnicos adicionais

- **Logs**: a aplicação deve registrar passos principais (ex.: início/fim da carga, total de linhas processadas, tempo de execução por etapa, erros de parsing).
- **Erros e códigos de saída**: scripts devem encerrar com código ≠ 0 em caso de falha (parsing, conexão, violação de integridade etc.).
- **Desempenho mínimo**: evitar operações desnecessárias em parsing; recomenda-se carga em lotes (batch inserts) e uso de `COPY` quando adequado.
- **Integridade**: todas as restrições definidas no PDF devem estar **ativas** no esquema implantado.
- **Reprodutibilidade**: `docker compose down -v` deve permitir **recriar** tudo do zero e repetir o processo com os mesmos resultados.

6. Entrega

A entrega será feita via GitHub Classroom: Assignment: [A ser definido]

O commit final deve conter **todos** os arquivos listados na Seção 3, em especial:

- `docs/tp1_3.1.pdf`
- `src/tp1_3.2.py`
- `src/tp1_3.3.py`
- `Dockerfile`, `docker-compose.yml`, `requirements.txt`
- `README.md` com o passo a passo da Seção 5

7. Critérios de Avaliação

- **Projeto do esquema e normalização** (qualidade do diagrama, dicionário, justificativas): 30%
- **Carga e consistência dos dados** (confiabilidade do ETL, restrições ativas, reprodutibilidade): 25%
- **Consultas (SQL) e qualidade das saídas** (correção, desempenho razoável, clareza): 30%
- **Conteinerização e automação** (Dockerfile, docker-compose, comando único, logs, códigos de saída): 15%

Penalidades: scripts que não rodam no contêiner, ausência de `docker-compose`, falta de PDF, ou que exigem ajustes manuais fora do contêiner.

8. Referências

[1] GARCIA-MOLINA Hector, ULLMAN, Jeffrey D., WIDOM, Jennifer. Database Systems: The Complete Book. 2ª ed. Prentice Hall, 2008. Seções 3.1, 3.3, 3.5 e 3.6

[2] ELMASRI, Ramez; NAVATHE, Shamkant B. Fundamentals of Database Systems, 6th edition. Addison Wesley, 2010. Capítulos 10 e 11

[3] PostgreSQL Python https://www.postgresqltutorial.com/postgresql-python/

Apêndices (modelo mínimo recomendado)

A. Exemplo de `docker-compose.yml` (base)

```
version: "3.9"
services:
  db:
    image: postgres:15
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
```

```
      POSTGRES_DB: ecommerce
    ports:
      - "5432:5432"
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres -d ecommerce"]
      interval: 5s
      timeout: 5s
      retries: 10
    volumes:
      - pgdata:/var/lib/postgresql/data

  app:
    build: .
    depends_on:
      db:
        condition: service_healthy
    volumes:
      - ./src:/app/src
      - ./data:/data
      - ./out:/app/out
    working_dir: /app

volumes:
  pgdata:
```

B. Exemplo de `Dockerfile` (base)

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY src ./src
CMD ["python", "src/tp1_3.3.py", "--help"]
```

C. Exemplo de `requirements.txt`

```
psycopg[binary]>=3.1
pandas>=2.2
python-dateutil>=2.9
```