Relatório / Comentários

Alunas: Maria Luiza da Silva; Graziela da Costa Mesquita

1) Sentido do projeto:

Este mini-projeto simula um pipeline ETL (Extração → Transformação → Carga) aplicado a um estoque de materiais escolares. A ideia é simples e prática:

- Problema real: uma escola/loja precisa saber quantos itens tem, quanto valem no total e quais precisam ser repostos.

- Objetivo do projeto: automatizar uma tarefa básica de controle de estoque: ler os dados, limpar, calcular valores e gerar relatórios (arquivos) com as informações úteis — por exemplo, “itens com estoque baixo”.

- Por que materiais escolares: é um conjunto de dados simples, familiar e didático (produto, quantidade, preço), ótimo para aprender ETL sem se perder em complexidade.

A aplicação (o script Python) funciona assim na prática:

1. Extrai os dados de um CSV (estoque.csv).

2. Transforma: limpa dados faltantes, calcula valor_total (quantidade × preço), filtra itens com estoque baixo e prepara um resumo por item.

3. Carrega os resultados em novos arquivos CSV (estoque_transformado.csv e estoque_baixo.csv) e imprime tabelas para o usuário ver.

2) Código (exemplo usado)

Aqui está o código completo que usamos (copie para o Colab/terminal):

```
import pandas as pd

# EXTRAÇÃO — Ler o arquivo CSV
df = pd.read_csv("/content/estoque.csv")

print("=== DADOS ORIGINAIS ===")
print(df)
```

```
print("\n")

# TRANSFORMAÇÃO — Limpeza, filtro e criação de métricas

# a) Remover linhas com valores nulos
df = df.dropna()

# b) Filtrar produtos com quantidade abaixo de 20 (baixa quantidade)
estoque_baixo = df[df["quantidade"] < 20]

# c) Criar uma nova coluna: valor_total (quantidade × preço)
df["valor_total"] = df["quantidade"] * df["preco"]

print("=== ESTOQUE COM VALOR TOTAL ===")
print(df)
print("\n")

print("=== PRODUTOS COM BAIXO ESTOQUE (menos de 20 unidades) ===")
print(estoque_baixo)
print("\n")

# CARREGAMENTO — Salvar resultado final
df.to_csv("/content/estoque_transformado.csv", index=False)
estoque_baixo.to_csv("/content/estoque_baixo.csv", index=False)

print("Arquivos gerados com sucesso:")
print("- estoque_transformado.csv")
print("- estoque_baixo.csv")
```

3) Explicação linha a linha

import pandas as pd

- O que faz: importa a biblioteca pandas e a apelida de pd.
- Por que: pandas fornece estruturas de dados (DataFrame) e funções para ler CSV, manipular tabelas, agrupar, filtrar e salvar — é ideal para ETL simples.

df = pd.read_csv("/content/estoque.csv")

- O que faz: lê o arquivo CSV e cria um DataFrame chamado df.

- Por que: essa é a etapa de extração. No Colab o caminho padrão é /content/. Se você rodar localmente, pode usar apenas "estoque.csv".

print("=== DADOS ORIGINAIS ===") / print(df) / print("\n")

- O que faz: imprime um cabeçalho e mostra a tabela original na tela.

- Por que: ajuda a visualizar o antes — importante para comparação e para o relatório (mostrar “antes e depois”).

df = df.dropna()

- O que faz: remove linhas que contenham qualquer valor NaN (faltante).

- Por que: evita erros nos cálculos (por exemplo, multiplicar NaN por número) e garante que o dataset final só contenha registros completos.

- Observação: remoção completa é a estratégia mais simples; em trabalhos reais você poderia optar por preencher (imputar) valores em vez de descartar (ex.: df.fillna(0) ou preencher com médias).

estoque_baixo = df[df["quantidade"] < 20]

- O que faz: cria um novo DataFrame contendo apenas as linhas onde a coluna quantidade é menor que 20.

- Por que: identifica produtos que precisam de reposição. O limite 20 é arbitrário — você pode mudar para <= 5 ou outro valor conforme necessidade.

- Observação: esse filtro não altera df; cria uma nova visão (estoque_baixo) para relatório.

df["valor_total"] = df["quantidade"] * df["preco"]

- O que faz: multiplica quantidade por preco em cada linha e salva na nova coluna valor_total.

- Por que: adiciona uma métrica útil: quanto dinheiro o estoque tem daquele item (útil para avaliação financeira).

- Observação: se preco ou quantidade não forem numéricos, dará erro — por isso a limpeza antes é importante.

print("=== ESTOQUE COM VALOR TOTAL ===") / print(df) / print("\n")

- O que faz: imprime o DataFrame já transformado, agora com a coluna valor_total.

- Por que: mostra o “depois” do processamento — essencial para validar que a regra aplicada funcionou.

print("=== PRODUTOS COM BAIXO ESTOQUE (menos de 20 unidades) ===") / print(estoque_baixo) / print("\n")

- O que faz: imprime apenas os itens detectados como críticos por baixa quantidade.

- Por que: é o relatório operacional imediato — quem gerencia estoque quer ver isso primeiro.

df.to_csv("/content/estoque_transformado.csv", index=False)

- O que faz: salva o DataFrame transformado em um novo CSV.

- Por que: é a etapa de carga — persistir resultados para uso posterior (relatórios, sistemas, auditoria).

- index=False evita que o índice (0,1,2...) vire uma coluna no CSV.

estoque_baixo.to_csv("/content/estoque_baixo.csv", index=False)

- O que faz: salva somente os itens com baixo estoque em outro CSV.

- Por que: facilita enviar só a lista de reposição ao responsável pelas compras.

print("Arquivos gerados com sucesso:") + prints de nomes

- O que faz: confirma visualmente que os arquivos foram gravados.

- Por que: dá feedback ao usuário; importante para UX simples de scripts.

4) Boas práticas

- Validar tipos: df["quantidade"] = df["quantidade"].astype(int) (se necessário) para evitar erros.

- Tratar separadores/decimais: pd.read_csv(..., decimal=",") se seu CSV usar vírgula decimal.

- Configurar parâmetro de “baixo estoque” como variável no topo: LIMITE_BAIXO = 20 para fácil alteração.

- Log vs print: usar logging para projetos maiores.

- Backup/versão: salvar com timestamps (ex.: estoque_transformado_20250505.csv) para auditoria.

- Agrupamentos: se houver múltiplas linhas do mesmo item, usar df.groupby("item").sum() para consolidar.

5) Resumo final

O script lê um arquivo de estoque, limpa dados, calcula quanto vale cada item no estoque, identifica itens com pouca quantidade e salva relatórios prontos para uso (tudo automático).

Fotos do Código

main.py +

```
3  import  pandas  as  pd
2
7  # Ler o CSV enviado
6  df  =  pd.read_csv ( "/content/ estoque.csv " )
5
8  print ( "=== DADOS ORIGINAIS ===" )
9  imprimir ( df )
27 imprimir ( " \n " )
11
10 # Remover nulos
29 df  =  df.dropna ( )
28
1  # Filtro
31 estoque_baixo  =  df [ df [ "quantidade" ]  <  20 ]
30
```

Linha: 30, Coluna: 1

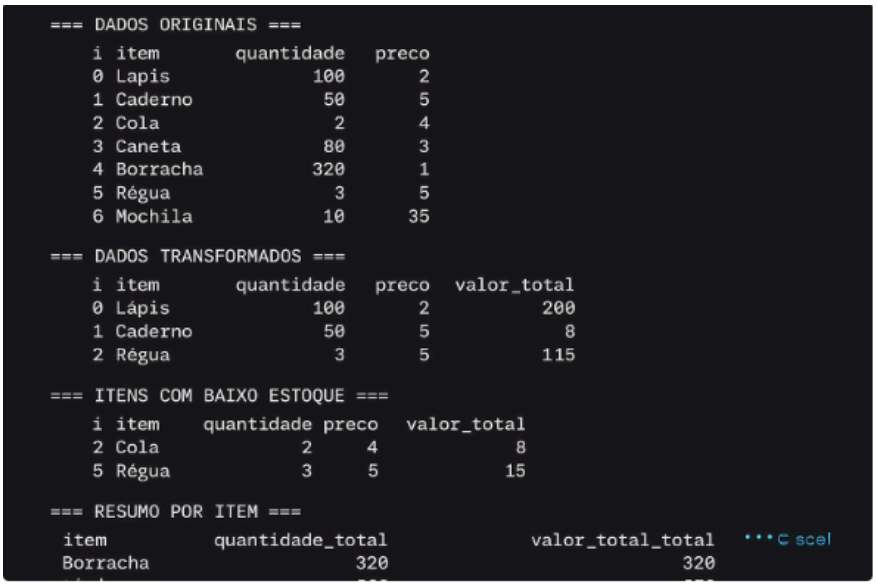
2)

main.py +

```
30 # Criar coluna
13 df [ "valor_total" ]  =  df [ "quantidade" ]  *  df [ "preco" ]
4
3  print ( "=== DADOS TRANSFORMADOS ===" )
2  imprimir ( df )
7  imprimir ( " \n " )
6
5  print ( "=== PRODUTOS COM BAIXO ESTOQUE ===" )
8  imprimir ( estoque_baixo )
9  imprimir ( " \n " )
27
11 # Salvar arquivos
10 df.to_csv ( " / content /estoque_transformado.csv" , index = False )
29 estoque_baixo . to_csv ( "/content/estoque_baixo.csv" ,  index = False )
28
```

Fotos da Compilação

1)



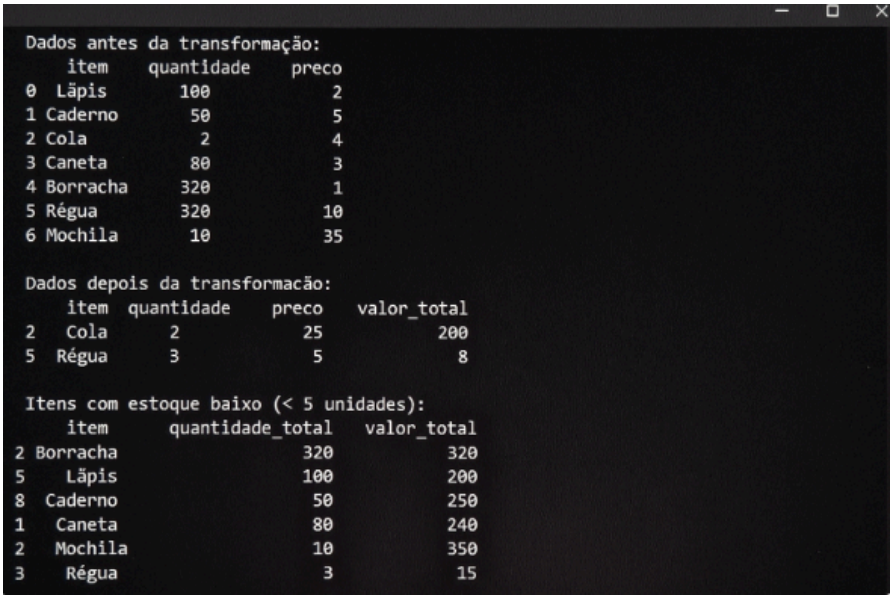
```
=== DADOS ORIGINAIS ===
    i item       quantidade   preco
    0 Lapis             100       2
    1 Caderno            50       5
    2 Cola                2       4
    3 Caneta             80       3
    4 Borracha          320       1
    5 Régua               3       5
    6 Mochila            10      35

=== DADOS TRANSFORMADOS ===
    i item       quantidade   preco  valor_total
    0 Lápis             100       2          200
    1 Caderno            50       5            8
    2 Régua               3       5          115

=== ITENS COM BAIXO ESTOQUE ===
    i item    quantidade preco   valor_total
    2 Cola             2     4             8
    5 Régua            3     5            15

=== RESUMO POR ITEM ===
 item          quantidade_total             valor_total_total   ···C scel
 Borracha                   320                          320
```

2)



```
Dados antes da transformação:
     item    quantidade    preco
0  Lãpis       100           2
1 Caderno       50           5
2 Cola           2           4
3 Caneta        80           3
4 Borracha     320           1
5 Régua        320          10
6 Mochila       10          35

Dados depois da transformacão:
     item  quantidade    preco   valor_total
2   Cola       2           25          200
5  Régua       3            5            8

Itens com estoque baixo (< 5 unidades):
     item      quantidade_total   valor_total
2 Borracha                320          320
5    Lãpis                100          200
8  Caderno                 50          250
1   Caneta                 80          240
2   Mochila                10          350
3    Régua                  3           15
```