**AULA 4 – VISUALIZAÇÃO DE DADOS EM PYTHON**

**Nome:** Rafael Macedo Lima

**Curso:** Análise e Desenvolvimento de Sistemas

**Disciplina:** Linguagem de Programação

**Cidade:** São Paulo, SP

**Data:** 25 de Setembro de 2025

**INTRODUÇÃO**

A visualização de dados é uma técnica essencial para comunicar padrões e resultados de forma clara. Este trabalho reproduz um fluxo prático de análise: geração de dados simulados em **SQLite**, tratamento com **Pandas** e apresentação visual com **Matplotlib/Seaborn**. O objetivo é aplicar conceitos da disciplina e entregar um relatório estruturado, com explicação detalhada do código.

**OBJETIVOS**

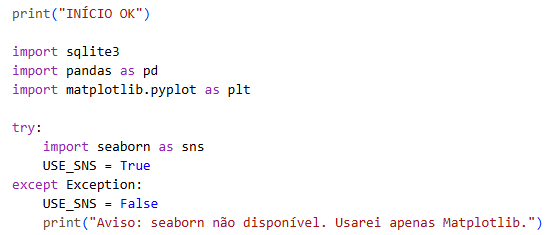
* Criar um banco de dados **SQLite** com tabela de vendas.
* Carregar e preparar os dados em um **DataFrame Pandas**.
* Realizar agregações por mês e por categoria.
* Construir dois gráficos (linha e barras) para interpretar tendências.
* Calcular métricas básicas de desempenho.
* Gerar **insights** e **conclusão automática** a partir dos resultados.
* Apresentar relatório acadêmico com prints e explicação linha a linha.

**METODOLOGIA**

* **Ambiente:** Google Colab (Python 3.x).
* **Ferramentas:** SQLite3, Pandas, Matplotlib, Seaborn.

**DESENVOLVIMENTO E EXPLICAÇÃO DO CÓDIGO**

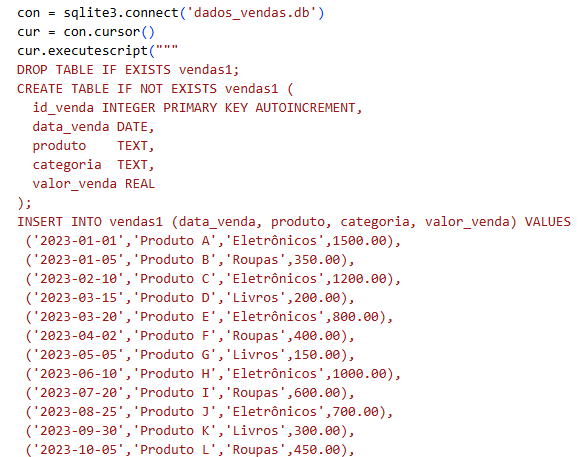
1. **Bibliotecas e configuração do ambiente**

****

**Explicação:**

* print(“INÍCIO OK”): confirma visualmente o início da execução.
* sqlite3: módulo padrão para criar/usar banco SQLite.
* pandas as pd: manipulação de tabelas.
* matplotlib.pyplot as plt: gráficos básicos.
* seaborn as sns: facilita a criação de gráficos bonitos e informativos em poucas linhas, facilitando a análises estatísticas. É uma biblioteca de visualização de dados construída em cima do Matplotlib, fornecendo cores padronizados e gráficos mais modernos.
* try/except: tenta importar seaborn (gráficos mais bonitos). Se não estiver disponível, o código continua só com Matplotlib.

1. **Banco SQLite: criação da tabela e inserção de dados**

****

**Uma imagem contendo Texto

O conteúdo gerado por IA pode estar incorreto.**

**Explicação:**

* sqlite3.connect: cria/abre o banco dados\_vendas.db.
* cursor():canal para executar SQL.
* executerscript: apaga tabela antiga, cria nova e insere registros simulados.
* commit(): grava os dados permanentemente.

1. **Leitura no Pandas e preparação de campos**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Explicação:**

* real\_sql\_query: consulta SQL = Dataframe.
* to\_datetime: transforma texto em datas reais.
* .dt.to\_period(“M”): extrai o mês da data.
* .astype(str): converte para string (melhora exibição nos gráficos).

1. **Agregações por mês e por categoria**

**Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.**

**Explicação:**

* groupby(“mes”).sum(): soma as vendas por mês.
* groupby(“categoria”).sum(): soma por categoria.
* as\_index=False: mantém o resultado como Dataframe (evita erros com idxmax).
* sort\_values: ordena meses/categorias para gráficos legíveis.

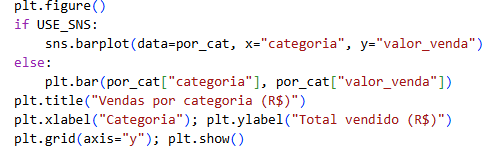
1. **Visualizações (linhas e barras)**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Explicação:**

* plt.figure(): abre uma nova figura.
* if USE\_SNS: testa variável booleana USE\_SNS (definida lá em cima quando tentamos import seaborn as sns. **True** Seaborn está disponível; **False** não está, usamos só Matplotlib.
* lineplot ou plot: cria gráfico de linha de vendas por mês. x=”mes", como horizontal e y=”valor\_venda” como vertical
* Título, rótulos e grade melhoram visualização.
* plt.show(): exibe o gráfico.

****

**Explicação:**

* barplot ou plt.bar: gráfico de barras comparando categorias.
* plt.grid(axis=”y”): adiciona linhas horizontais, facilitando a leitura.

1. **Métricas e insights**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Explicação:**

* sum(): soma total das vendas.
* len(df): número de registros (divide para obter ticket médio).
* idxmax/idxmin: pegam a linha com maior/menor valor de vendas por mês.
* iloc[0]: categoria líder (já ordenada).
* df.loc[df[“valor\_venda”].idxmax()]: maior venda individual.

1. **Código de Conclusão automática**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Explicação:**

* br\_money: função que formata valores no padrão brasileiro (vírgula como separador decimal).
* print(“=== INSIGHTS ===”): início da seção de resultados.
* Cada print(f”...”): exibe métrica formatada.
* Este bloco é essencial porque já produz um **texto interpretativo pronto** para o relatório.

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Explicação:**

* print(“=== CONCLUSÃO AUTOMÁTICA ===”): apenas imprime o cabeçalho para separar a parte da conclusão dos outros resultados. Visualmente ajuda a identificar no console que ali começa o resumo interpretativo.
* print(“Com a base nos dados de vendas, o faturamento total foi de R$ “ + br\_money(total) ...): esse print monta um **texto único** juntando strings fixas + valores calculados. Ele usa **concatenação de strings (+)** para misturar frases com números formatados.
* br\_money(total): pega o faturamento total e formata em moeda brasileira (com vírgula decimal e duas casas).
* str(top\_mes\_row[“mes"]) pega o nome do **mês maior faturamento** e transforma em string para aparecer no texto.
* str(worst\_mes\_row[“mes"]): pega o **mês com pior desempenho.**
* str(top\_cat\_row[“categoria”]): pega a **categoria líder em vendas.**

**Exemplo de saída:**

Com base nos dados de vendas, o faturamento total foi de R$ 8.800,00, com o melhor desempenho no mês de 2023-01 e o desempenho mais baixo em 2023-05. A categoria que mais contribuiu para o resultado foi Eletrônicos. Esses resultados sugerem priorizar estoque e ações promocionais em Eletrônicos, além de planejar campanhas nos meses que historicamente performam pior.

* print(“FIM OK”): indica que a execução terminou sem erros. Serve como **marcador de controle** para o usuário saber que todo código rodou até o fim.

Esse trecho pega os cálculos anteriores (total de vendas, melhor/pior do mês, categoria líder) e gera **automaticamente uma análise textual**. Isso é importante porque não se trata apenas de mostrar números, mas de traduzir os dados em uma **conclusão gerencial**: onde investir mais, quando se preocupar com queda de desempenho e qual categoria deve receber prioridade.

**RESULTADOS E DISCUSSÃO**

Após a execução do código no Google Colab, obtivemos os seguintes resultados:

1. **Prévia de dados**

O DataFrame apresentou os registros inseridos no banco **SQLite**, contendo as colunas data\_venda, produto, categoria e valor\_venda. Essa prévia confirmou que os dados estavam corretamente carregados, permitindo seguir com as análises.

1. **Gráfico de linha – Vendas por mês**

O primeiro gráfico, do tipo **linha,** mostrou a evolução do faturamento mensal ao longo do ano de 2023.

* Foi possível observar **picos em meses específicos**, indicando períodos de maior movimentação comercial.
* Em contrapartida, alguns meses registram **queda significativa de vendas**, refletindo sazonalidade ou baixa demanda.
* Essa visualização é útil para **identificar padrões temporais** e apoiar o planejamento de promoções ou reforço de estoque em meses com desempenho historicamente menor.

**3. Gráfico de barras – Vendas por categoria**

O segundo gráfico, do tipo **barras**, evidenciou o desempenho de cada categoria do produto.

* A categoria **Eletrônicos** destacou-se como a de maior faturamento, superando as demais de forma consistente.
* **Roupas** apresentou valores intermediários, enquanto **Livros** mostrou menor volume de vendas.
* Essa análise comparativa sugere que **Eletrônicos** devem receber maior atenção em termos de marketing, logísticas e reposição de estoque, uma vez que representam a principal fonte de receita.

**4. Métricas numéricas**

Além dos gráficos, o código gerou métricas quantitativas:

* **Faturamento total do ano:** soma de todas as vendas registradas no período.
* **Ticket médio:** resultado da divisão entre o faturamento total e o número de transações, permitindo avaliar o gasto médio por venda.
* **Melhor mês:** identificado via idxmax, representando o período de maior faturamento.
* **Pior mês:** identificado via idxmin, com o menor resultado de vendas.
* **Categoria líder:** a categoria com maior soma de vendas, confirmada também no gráfico de barras.
* **Maior venda unitária:** produto específico que gerou o maior valor em uma única transição.

Essas métricas complementam a análise visual, fornecendo valores exatos que podem ser utilizados em relatórios financeiros ou em reuniões de tomada de decisão.

**5.** **Conclusão automática gerada pelo código**

Uma das principais contribuições desta atividade foi a **geração de uma conclusão automática em linguagem natural,** construída a partir dos cálculos realizados. O código não apenas mostrou números, mas **interpretou os resultados** destacando:

* O faturamento total no ano.
* O melhor e o pior mês de vendas.
* A categoria que mais contribuiu para o faturamento.
* Recomendações estratégicas, como priorizar **estoque e campanhas promocionais** para a categoria líder e planejar ações específicas em meses de baixo desempenho.

Esse tipo de automação representa uma aplicação prática da ciência de dados, pois **transforma dados brutos em insights acionáveis.**

**6. Discussão críticas**

* Os resultados obtidos são **condizentes com um cenário real de varejos**, onde existe sazonalidade e diferença de desempenho entre categorias.
* É importante destacar que os dados utilizados foram **simulados**; portanto, os números absolutos não refletem um caso real, mas a **metodologia aplicada é totalmente transferível** para bases de dados reais.
* A integração entre **SQLite (armazenamento), Pandas (tratamento), Matplotlib/Seaborn (visualização) e Python (lógica de conclusão)** demonstrou a força do ecossistema Python em análises completas de ponta a ponta.
* A prática evidencia que **dados só se tornam úteis quando interpretados,** e que a geração de relatórios automatizados pode reduzir o tempo e erros em ambientes corporativos.

**CÓDIGO COMPLETO**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

**RESULTADOS**

**Tabela

O conteúdo gerado por IA pode estar incorreto.**

**Interface gráfica do usuário, Texto, Aplicativo

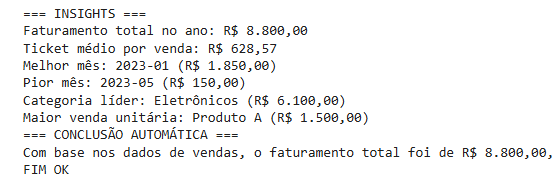
O conteúdo gerado por IA pode estar incorreto.**

Gráfico, Gráfico de linhas

O conteúdo gerado por IA pode estar incorreto.

Gráfico, Gráfico de barras

O conteúdo gerado por IA pode estar incorreto.



**CONCLUSÃO**

A realização desta atividade prática de visualização de dados em Python permitiu consolidar de forma integrada diversos conceitos trabalhados ao longo da disciplina, desde a criação de um banco de dados até a interpretação final dos resultados. Inicialmente, a construção da tabela de vendas no SQLite demonstrou como é possível simular um ambiente real de armazenamento de dados de forma simples e eficiente. Esse passo foi fundamental para compreender a importância de estruturar corretamente as informações antes de qualquer análise, evidenciando a relação entre bases de dados relacionais e processos de ciência de dados.

Na etapa seguinte, a utilização do Pandas mostrou-se essencial para a manipulação e preparação do dataset. A conversão de datas, a criação da coluna de meses e as agregações por categoria e período exemplificaram práticas comuns no dia a dia de analistas de dados, como a transformação e o agrupamento de variáveis. Essa etapa evidenciou que, para gerar bons gráficos, é indispensável preparar previamente os dados de forma coerente.

A visualização com Matplotlib e Seaborn reforçou a relevância dos gráficos como instrumentos de apoio à tomada de decisão. O gráfico de linha possibilitou identificar oscilações mensais e padrões de sazonalidade, enquanto o gráfico de barras destacou claramente as diferenças entre as categorias de produtos. Essa representação visual trouxe clareza imediata, permitindo insights rápidos que dificilmente seriam percebidos apenas na tabela de dados.

Outro aspecto de grande valor foi o cálculo de métricas-chave: faturamento total, ticket médio, melhor e pior mês, categoria líder e maior venda unitária. Esses indicadores ampliaram a interpretação dos gráficos e forneceram um diagnóstico mais objetivo sobre o desempenho das vendas, exemplificando o uso de indicadores de performance (KPIs) aplicados à área de negócios.

A etapa final, que gerou uma conclusão automática em linguagem natural, demonstrou um avanço significativo: a automatização da interpretação dos resultados. Esse recurso evidencia como a programação pode não apenas processar dados, mas também traduzir números em recomendações práticas, aproximando a análise do contexto gerencial. Essa característica é particularmente relevante em relatórios corporativos, pois economiza tempo e facilita a compreensão por gestores que nem sempre possuem formação técnica.

De forma geral, esta atividade comprovou que o ecossistema Python, composto por SQLite, Pandas, Matplotlib e Seaborn, oferece todas as ferramentas necessárias para a construção de um pipeline analítico completo, indo do armazenamento de dados à geração de insights estratégicos. Além de fortalecer habilidades técnicas, o exercício reforçou a importância da clareza na comunicação dos resultados, uma vez que dados só se tornam valiosos quando transformados em informações úteis para a tomada de decisão.

Conclui-se, portanto, que a atividade cumpriu plenamente seus objetivos: demonstrou a criação e manipulação de dados, apresentou técnicas de visualização eficazes, aplicou métricas de análise de desempenho e, sobretudo, mostrou como automatizar a geração de conclusões em relatórios acadêmicos ou empresariais.

**REFERÊNCIAS**

* MCKINNEY, Wes. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.*2. ed. Sebastopol: O’Reilly Media, 2017.
* *Pandas Documentation.* Panda.pydata.org, 2025. Disponível em: <https://pandas.pydata.org/docs/>. Acesso em 21 set. 2025.
* *Matplotlib Documentation.* Matplotlib.org, 2025. Disponível em: <https://matplotlib.org/stable/contents.html>. Acesso em 21 set. 2025.
* *Seaborn Documentation.* Seaborn.pydata.org, 2025. Disponível em: <https://seaborn.pydata.org/>. Acesso 21 set. 2025.