



CENTRO UNIVERSITÁRIO ANHANGUERA PITÁGORAS UNOPAR DE CAMPO GRANDE

CURSO: ENGENHARIA DE SOFTWARE

DISCIPLINA: LINGUAGEM DE PROGRAMAÇÃO

ATIVIDADE PRÁTICA – UNIDADE 3, AULA 4

VISUALIZAÇÃO DE DADOS EM PYTHON

Aluno: Edmar Radanovis.

RA: 2025223493

Polo: Itapira / SP-UN944038

Ano 2025 / 2º Semestre

ANHANGUERA EDUCACIONAL

ENGENHARIA DE SOFTWARE

DISCIPLINA: Linguagem de Programação

PROFESSOR: Anderson I. S. Abreu / Vanessa Matias Leite

ALUNO: Edmar Radanovis

RA: 2025223493

TÍTULO: Relatório da Aula Prática – Visualização de Dados em Python – Unidade 3 –
Aula 4

POLO: Itapira / SP-UN944038

Monte Sião, 09 de Setembro de 2025.

RESULTADOS DA ATIVIDADE PRÁTICA

Proposta:



Introdução à Análise de Dados com Python - Visualização de Dados.

- Você trabalha em uma empresa de varejo e precisa analisar os dados de vendas do último ano para identificar padrões e insights para melhorar o desempenho. Os dados estão armazenados em um banco de dados SQLite, e você utilizará a biblioteca Pandas para manipular e analisar esses dados, além de gerar visualizações utilizando Matplotlib e Seaborn.
 - **Conectar ao banco de dados SQLite e criar uma tabela:**
 - Estabelecer uma conexão com o banco de dados SQLite e carregar os dados relevantes para análise.
 - Criar e conectar ao banco de dados.
 - Criar um cursor.
 - Criar uma tabela.
 - Inserir dados.
 - Confirmar as mudanças.
 - **Explorar e preparar os dados:**
 - Com os dados devidamente carregados em um DataFrame do Pandas, explorá-los e prepará-los para análise.

- **Analisar os dados :**

- Analisar especificamente para extrair insights.

- **Visualizar os dados :**

- Utilizar Matplotlib e Seaborn para criar visualizações que ajudem a interpretar os resultados.

- **Concluir e analisar os insights dos dados :**

- Finalizar o exercício com uma breve análise dos insights obtidos e sugestões para a empresa com base nos dados analisados.

link do repositório no GitHub:

https://github.com/ed-radanovis/Eng_Software_L-P_U3-A4_09-2025-.git


```
File Edit Selection View Go ... atividades
unit_three_lesson_four_data_visualization.py
U3_A4_VISUALIZACAO_DE_DADOS_EM_PYTHON > unit_three_lesson_four_data_visualization.py > visualizar_dados
137 def visualizar_dados(df, analises):
138     # Configuração do layout dos gráficos
139     fig, axes = plt.subplots(3, 2, figsize=(15, 12))
140     fig.suptitle('Análise de Vendas - Ano 2023', fontsize=16, fontweight='bold', color='#2E4053')
141     cores_pizza = ['#0000FF', '#FF0000', '#FFFF00']
142     cores_barra = ['#0000FF', '#FF0000', '#FFFF00']
143     cores_boxplot = ['#0000FF', '#FF0000', '#FFFF00']
144
145     # Gráfico 1: Vendas por Categoria (Pizza)
146     categorias = analises['vendas_categoria'].index
147     valores = analises['vendas_categoria'].values
148
149     axes[0, 0].pie(valores, labels=categorias, autopct='%1.1f%%', colors=cores_pizza, textprops={'fontsize': 10,
150     'fontweight': 'bold'})
151     axes[0, 0].set_title('Distribuição de Vendas por Categoria', fontweight='bold', fontsize=12)
152
153     # Gráfico 2: Vendas por Mês (Barra)
154     meses = ['Jan', 'Fev', 'Mar', 'Abr', 'Mai', 'Jun', 'Jul', 'Ago', 'Set', 'Out', 'Nov', 'Dez']
155     vendas_mes_completo = analises['vendas_mes'].reindex(range(1, 13), fill_value=0)
156
157     bars = axes[0, 1].bar(meses, vendas_mes_completo.values, color=cores_barra, alpha=0.8)
158     axes[0, 1].set_title('Vendas Mensais', fontweight='bold', fontsize=12)
159     axes[0, 1].set_ylabel('Valor de Vendas (R$)', fontweight='bold')
160     axes[0, 1].tick_params(axis='x', rotation=45)
161     axes[0, 1].grid(True, alpha=0.3)
162
163     # Adicionar valores nas barras
164     for bar in bars:
165         height = bar.get_height()
166         if height > 0:
167             axes[0, 1].text(bar.get_x() + bar.get_width()/2., height + 40,
168                 f'R$ {height:0f}', ha='center', va='bottom',
169                 fontweight='bold', fontsize=9)
170
171     # Gráfico 3: Distribuição de Vendas (Boxplot)
172     boxplot = sns.boxplot(data=df, x='categoria', y='valor_venda', ax=axes[1, 0], palette=cores_boxplot)
173     axes[1, 0].set_ylabel('Valor de Venda (R$)', fontweight='bold', fontsize=12)
174     axes[1, 0].set_xlabel('Categoria', fontweight='bold')
175     axes[1, 0].grid(True, alpha=0.3)
176
177     # Gráfico 4: Contagem de Vendas por Categoria (Barra)
178     countplot = sns.countplot(data=df, x='categoria', ax=axes[1, 1], palette=cores_boxplot, hue='categoria', legend=False)
179     axes[1, 1].set_title('Quantidade de Vendas por Categoria', fontweight='bold', fontsize=12)
180     axes[1, 1].set_ylabel('Número de Vendas', fontweight='bold')
181     axes[1, 1].set_xlabel('Categoria', fontweight='bold')
182     axes[1, 1].grid(True, alpha=0.3)
183
184     # Adicionar valores nas barras de countplot
185     for container in countplot.containers:
186         for bar in container:
187             height = bar.get_height()
188             axes[1, 1].text(bar.get_x() + bar.get_width()/2., height + 0.1,
189                 f'{int(height)}', ha='center', va='bottom',
190                 fontweight='bold', fontsize=10)
191
192     plt.tight_layout()
193     plt.show()
194
195     # Gráfico adicional: Tendência temporal
196     df.groupby('data_venda')['valor_venda'].sum().plot(
197         markers='o', color='#0000FF', linewidth=3, markersize=8)
198
199     plt.title('Tendência de Vendas ao Longo do Tempo', fontweight='bold', fontsize=14)
200     plt.ylabel('Valor de Vendas (R$)', fontweight='bold')
201     plt.xlabel('Data', fontweight='bold')
202     plt.grid(True, alpha=0.3)
203     plt.xticks(rotation=45)
204     plt.tight_layout()
205     plt.show()
206
207 # Passo 3: Conclusão e Insights
208 def gerar_insights(df, analises):
209     """Gera insights baseados na análise dos dados"""
210     print("\n" + "="*80)
211     print("INSIGHTS E RECOMENDAÇÕES")
212     print("="*80)
213
214     # Encontrar o mês com maior venda
215     mes_max_venda = analises['vendas_mes'].idxmax()
216     valor_max_venda = analises['vendas_mes'].max()
217
218     # Encontrar a categoria mais vendida
219     categoria_max_venda = analises['vendas_categoria'].idxmax()
220     valor_categoria_max = analises['vendas_categoria'].max()
221
222     print(f"📊 **Insight 1:** Mês de maior faturamento foi {mes_max_venda}º mês com R$ {valor_max_venda:2f}")
223     print(f"📊 **Insight 2:** Categoria '{categoria_max_venda}' foi a mais lucrativa (R$ {valor_categoria_max:2f})")
224
225     # Calcular participação percentual
226     participacao = (analises['vendas_categoria'] / analises['vendas_categoria'].sum() * 100).round(1)
227
228     print(f"📊 **Insight 3:** Participação por categoria:")
229     for categoria, percentual in participacao.items():
230         print(f"   - {categoria}: {percentual}%")
231
232     # Recomendações
233     print("\n💡 **Recomendações Estratégicas:**")
234     print("1. Investir mais em marketing para a categoria de maior faturamento")
235     print("2. Analisar sazonalidade para planejar estoques")
236     print("3. Desenvolver promoções para categorias com menor participação")
237     print("4. Expandir portfólio na categoria mais lucrativa")
238     print("5. Focar em estratégias para os meses de menor performance")
239
240 # Função principal
241 def main():
242     """Função principal que executa todo o fluxo de análise"""
243     print("\nSISTEMA DE ANÁLISE DE DADOS DE VENDAS")
244     print("="*80)
245
246     # Passo 1: Criar banco de dados
247     criar_banco_dados()
248
249     # Passo 2: Carregar dados
250     df_vendas = carregar_dados()
251
252     if df_vendas is not None:
253         # Exibir primeiras linhas
254         print("\n📄 PRIMEIRAS LINHAS DO DATASET:")
255         print(df_vendas.head())
256
257         # Passo 3: Analisar dados
258         resultados_analise = analisar_dados(df_vendas)
259
260         # Passo 4: Visualizar dados
261         visualizar_dados(df_vendas, resultados_analise)
262
263         # Passo 5: Gerar insights
264         gerar_insights(df_vendas, resultados_analise)
265
266 # Executar o programa
267 if __name__ == "__main__":
268     # Instalar dependências se necessário (no Colab)
269     try:
270         import seaborn
271     except ImportError:
272         print("Instalando bibliotecas necessárias...")
273         import subprocess
274         subprocess.run(["pip", "install", "seaborn", "matplotlib", "pandas"])
275
276     main()
277
278 U2_A4_BIBLIOTECAS_E_MODULOS_EM_PYTHON master 0 UTF-8 CRLF Python 3.13.7 Go Live Quokka Prettier
```

=> Figura 2: Print da tela com o código 2º trecho.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python: unit_three_lesson_four_data_visualization + - [ ] [ ] ... | X
20:57:18 atividades 72ms
20:57:18 atividades 72ms
-> & C:/Users/edrad/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/edrad/OneDrive/Área de Trabalho/ed/Anhanguer
a_Eng_Software/2º_semestre/5ª-Linguagem_de_Programacao/atividades/U3_A4_VISUALIZACAO_DE_DADOS_EM_PYTHON/unit_three_lesson_fo
...
SISTEMA DE ANÁLISE DE DADOS DE VENDAS
=====
[✓] Banco de dados já existe com dados!
[✓] Dados carregados com sucesso!
[M] Total de registros: 14

[+] PRIMEIRAS LINHAS DO DATASET:
id_venda data_venda produto categoria valor_venda mes trimestre
0 1 2023-01-01 Produto A Eletrônicos 1350.0 1 1
1 2 2023-01-05 Produto B Roupas 250.0 1 1
2 3 2023-02-10 Produto C Eletrônicos 1200.0 2 1
3 4 2023-03-15 Produto D Livros 200.0 3 1
4 5 2023-03-20 Produto E Eletrônicos 800.0 3 1
=====

ANÁLISE DE DADOS DE VENDAS
=====

[✓] ESTATÍSTICAS GERAIS:
Total de vendas: R$ 8250.00
Média de vendas: R$ 589.29
Maior venda: R$ 1350.00
Menor venda: R$ 150.00

[V] VENDAS POR CATEGORIA:
sum count
categoria
Eletrônicos 5650.0 6
Livros 900.0 4
Roupas 1700.0 4

[V] VENDAS POR MÊS:
mes
1 1600.0
2 1200.0
3 1000.0
4 400.0
5 150.0
6 900.0
7 600.0
8 700.0
9 300.0
10 450.0
11 700.0
12 250.0
Name: valor_venda, dtype: float64

[C] CRIANDO VISUALIZAÇÕES...

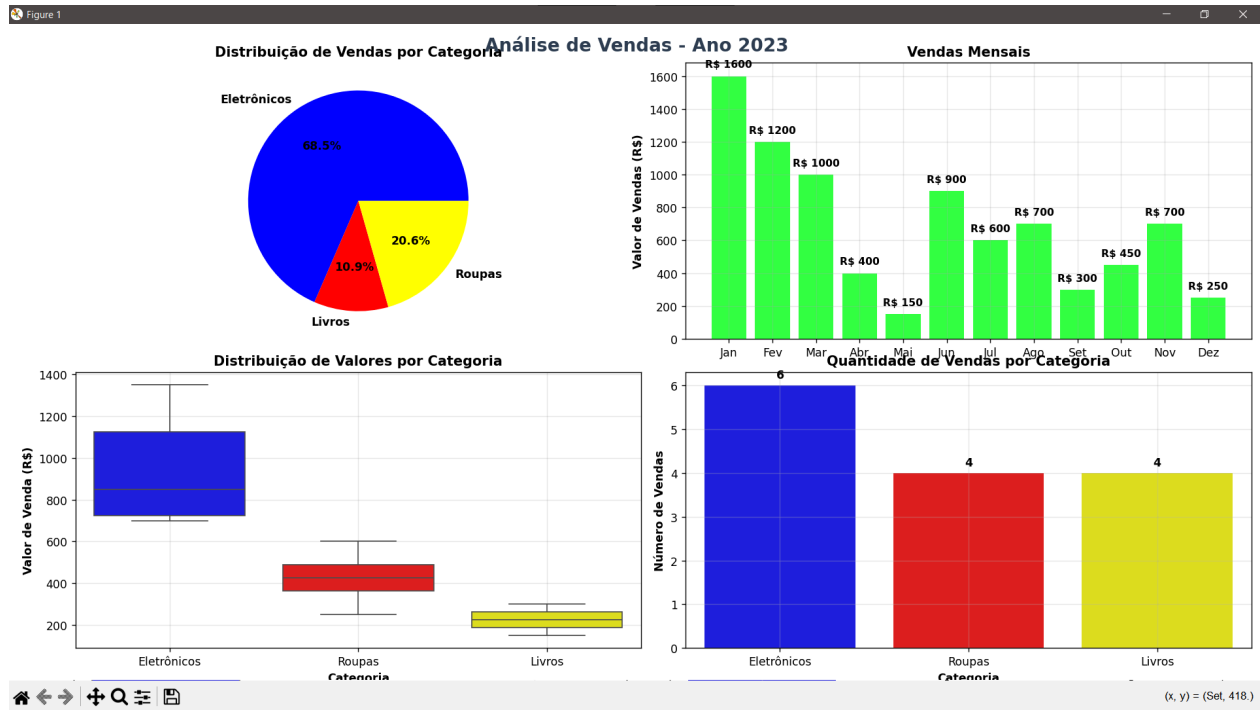
=====
INSIGHTS E RECOMENDAÇÕES
=====

[✓] **Insight 1:** Mês de maior faturamento foi 1º mês com R$ 1600.00
[V] **Insight 2:** Categoria 'Eletrônicos' foi a mais lucrativa (R$ 5650.00)
[M] **Insight 3:** Participação por categoria:
• Eletrônicos: 68.5%
• Livros: 10.9%
• Roupas: 20.6%

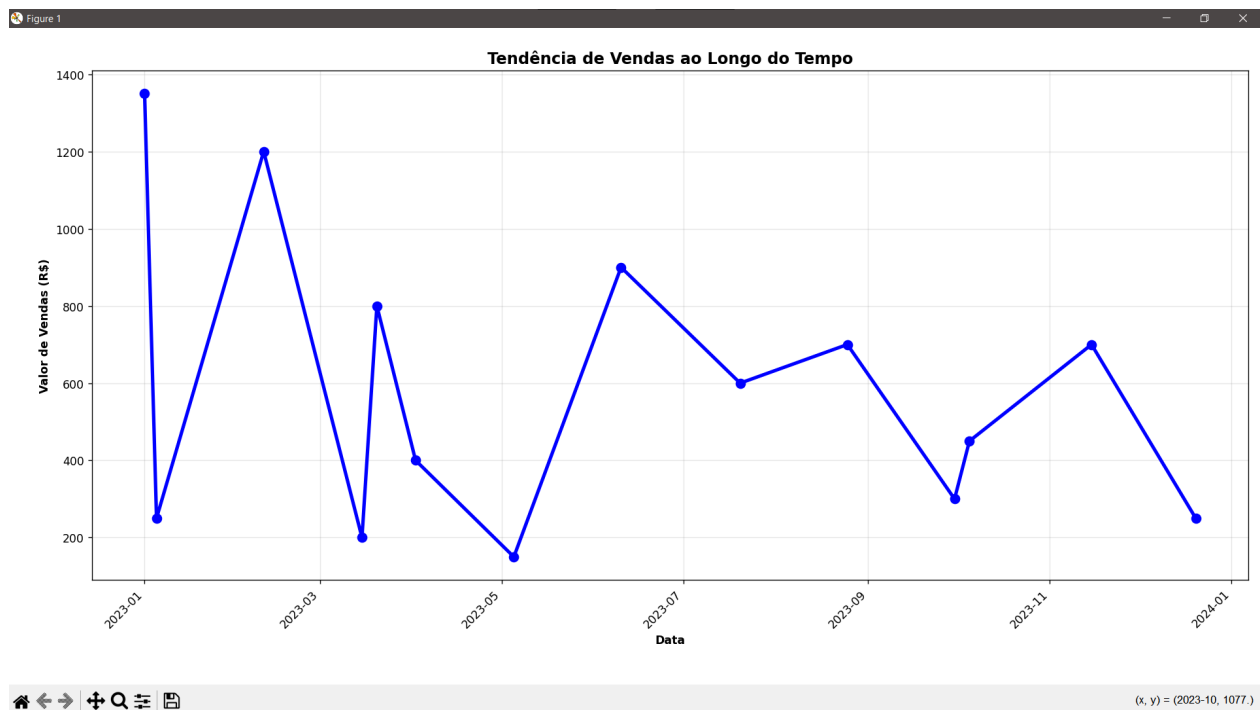
[V] **Recomendações Estratégicas:**
1. Investir mais em marketing para a categoria de maior faturamento
2. Analisar sazonalidade para planejar estoques
3. Desenvolver promoções para categorias com menor participação
4. Expandir portfólio na categoria mais lucrativa
5. Focar em estratégias para os meses de menor performance
20:57:39 atividades 21.387s
-> 

```

=> Figura 3: Print da tela do terminal executando as funções propostas.



=> Figura 4: Resultado do gráfico de análise das vendas 2023.



=> Figura 5: Resultado do gráfico de tendências de vendas.

Lógica Utilizada para a Atividade de Análise de Dados

O sistema foi desenvolvido seguindo uma abordagem modular e orientada a objetos, implementando os seguintes componentes principais:

Arquitetura do Sistema:

1. Estrutura em Camadas:

- ❖ **Banco de Dados SQLite:** Armazenamento persistente dos dados de vendas.
- ❖ **Camada de ETL:** Extração, Transformação e Carregamento com *Pandas*.
- ❖ **Camada de Análise:** Processamento e cálculo de métricas.
- ❖ **Camada de Visualização:** Geração de gráficos com *Matplotlib/Seaborn*.
- ❖ **Camada de Apresentação:** Relatório final com insights.

2. Fluxo de Dados:

```
SQLite → Pandas (DataFrame) → Análise Estatística → Visualização → Insights
```

Lógica de Implementação:

1. Gestão do Banco de Dados:

- ❖ Criação automática da tabela vendas se não existir.
- ❖ Inserção de dados de exemplo com transação controlada.
- ❖ Conexão segura com tratamento de exceções.
- ❖ Caminho relativo para portabilidade do projeto.

2. Processamento de Dados com Pandas:

```
# Transformações aplicadas:
df['data_venda'] = pd.to_datetime(df['data_venda']) # Conversão para datetime
df['mes'] = df['data_venda'].dt.month                # Extração do mês
df['trimestre'] = df['data_venda'].dt.quarter         # Extração do trimestre
```

3. Análise Estatística:

- ❖ **Agregações por categoria:** Soma e contagem de vendas.
- ❖ **Análise temporal:** Vendas mensais e sazonais.
- ❖ **Estatísticas descritivas:** Média, máximo, mínimo, soma total.
- ❖ **Cálculo de participação:** percentual por categoria.

4. Visualização de Dados:

- **Gráficos implementados:**
 - ❖ **Pizza:** Distribuição percentual por categoria.
 - ❖ **Barras:** Vendas mensais com valores anotados.
 - ❖ **Boxplot:** Distribuição de valores por categoria.
 - ❖ **Countplot:** Quantidade de vendas por categoria.
 - ❖ **Linha:** Tendência temporal das vendas.

5. Palette de Cores:

- **Cores vibrantes para melhor visualização:**

- ❖  **#0000FF**.
- ❖  **#FF0000**.
- ❖  **#FFFF00**.
- ❖  **#00FF15**.

6. Geração Automática de Insights:

- ❖ Identificação do mês de pico de vendas.
- ❖ Categoria mais lucrativa.
- ❖ Participação percentual por segmento.
- ❖ Recomendações estratégicas baseadas em dados.

Pontos Técnicos Destacados:

1. Tratamento de Erros:

```
try:
    # Operações críticas
except Exception as e:
    print(f"❌ Erro: {e}") # Feedback amigável ao usuário
```

2. Otimizações:

- ❖ Re-indexação para meses sem vendas (evita gaps nos gráficos).
- ❖ Prevenção de duplicação de dados no banco .
- ❖ Layout responsivo com `plt.tight_layout()`.
- ❖ Labels claros e informações anotadas.

3. Padrões de Projeto:

- ❖ Separação de concerns em funções especializadas.
- ❖ Data-driven programming com *Pandas*.
- ❖ Visualization pipeline com *Matplotlib*.
- ❖ Relatório automático com insights acionáveis.

Valor Gerado:

O sistema transforma dados brutos em:

- ✅ Informações estruturadas. ✅ Visualizações compreensíveis.
- ✅ Insights acionáveis. ✅ Recomendações estratégicas.

Resultado: Uma solução completa de business intelligence em Python que permite à empresa tomar decisões baseadas em dados concretos sobre performance de vendas, sazonalidade e mix de produtos. 