



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.

- o **Analizar os dados :**

- Analisar especificamente para extrair insights.

- o **Visualizar os dados :**

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

- o **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
  1 # -*- coding: utf-8 -*-
  2 # Sistema de Análise de Dados de Vendas
  3 # Utilizando SQLite, Pandas, Matplotlib e Seaborn
  4
  5 import sqlite3
  6 import pandas as pd
  7 import matplotlib.pyplot as plt
  8 import seaborn as sns
  9 from datetime import datetime
 10 import numpy as np
 11 import warnings
 12 import os
 13
 14 warnings.filterwarnings('ignore')
 15
 16 # Configuração do estilo visual
 17 plt.style.use('default')
 18 sns.set_palette("bright")
 19
 20 # Passo 1: Conectar ao banco de dados SQLite e criar tabela
 21 def criar_banco_dados():
 22     """Cria o banco de dados SQLite e popula com dados de exemplo"""
 23     try:
 24         # Definir caminho do banco dentro da pasta U3
 25         caminho_bd = 'U3_A4_VISUALIZACAO_DE_DADOS_EM PYTHON/data_base_sales.db'
 26
 27         # Garantir que a pasta existe
 28         os.makedirs(os.path.dirname(caminho_bd), exist_ok=True)
 29
 30         # Conectar ao banco de dados (cria se não existir)
 31         conexao = sqlite3.connect(caminho_bd)
 32         cursor = conexao.cursor()
 33
 34         # Criar tabela (se não existir)
 35         cursor.execute("""
 36             CREATE TABLE IF NOT EXISTS vendas (
 37                 id_venda INTEGER PRIMARY KEY AUTOINCREMENT,
 38                 data_venda DATE,
 39                 produto TEXT,
 40                 categoria TEXT,
 41                 valor_venda REAL
 42             );
 43
 44         # Verificar se a tabela já tem dados
 45         cursor.execute("SELECT COUNT(*) FROM vendas")
 46         if cursor.fetchone()[0] == 0:
 47             # Inserir dados de exemplo
 48             dados_exemplo = [
 49                 ('2023-01-01', 'Produto A', 'Eletrônicos', 1250.00),
 50                 ('2023-01-05', 'Produto B', 'Roupas', 250.00),
 51                 ('2023-02-10', 'Produto C', 'Eletrônicos', 1200.00),
 52                 ('2023-02-15', 'Produto D', 'Livros', 200.00),
 53                 ('2023-03-20', 'Produto E', 'Eletrônicos', 1500.00),
 54                 ('2023-04-02', 'Produto F', 'Roupas', 400.00),
 55                 ('2023-05-05', 'Produto G', 'Livros', 150.00),
 56                 ('2023-05-20', 'Produto H', 'Eletrônicos', 1000.00),
 57                 ('2023-06-28', 'Produto I', 'Roupas', 600.00),
 58                 ('2023-07-20', 'Produto J', 'Eletrônicos', 700.00),
 59                 ('2023-08-25', 'Produto K', 'Livros', 300.00),
 60                 ('2023-09-30', 'Produto L', 'Roupas', 800.00),
 61                 ('2023-10-15', 'Produto M', 'Eletrônicos', 700.00),
 62                 ('2023-11-15', 'Produto N', 'Livros', 250.00),
 63                 ('2023-12-20', 'Produto O', 'Roupas', 1000.00)
 64             ]
 65
 66             cursor.executemany("""
 67                 INSERT INTO vendas (data_venda, produto, categoria, valor_venda)
 68                 VALUES (?, ?, ?, ?)
 69             """, dados_exemplo)
 70
 71             conexao.commit()
 72             print(" Banco de dados criado e populado com dados de exemplo!")
 73         else:
 74             print(" Banco de dados já existe com dados!")
 75
 76         conexao.close()
 77
 78     except Exception as e:
 79         print(f" Erro ao criar banco de dados: {e}")
 80
 81     # Passo 2: Explorar e preparar os dados
 82     def carregar_dados():
 83         """Carrega os dados de SQLite para um DataFrame do Pandas"""
 84         try:
 85             # Usar o mesmo caminho da criação
 86             caminho_bd = 'U3_A4_VISUALIZACAO_DE_DADOS_EM PYTHON/data_base_sales.db'
 87             conexao = sqlite3.connect(caminho_bd)
 88             df_vendas = pd.read_sql_query("SELECT * FROM vendas", conexao)
 89             conexao.close()
 90
 91             # Converter tipos de dados
 92             df_vendas['data_venda'] = pd.to_datetime(df_vendas['data_venda'])
 93             df_vendas['mes'] = df_vendas['data_venda'].dt.month
 94             df_vendas['trimestre'] = df_vendas['data_venda'].dt.quarter
 95
 96             print(" Dados carregados com sucesso!")
 97             print(f" Total de registros: {len(df_vendas)}")
 98
 99             return df_vendas
 100
 101        except Exception as e:
 102            print(f" Erro ao carregar dados: {e}")
 103            return None
 104
 105    # Passo 3: Analise dos dados
 106    def analisar_dados(df):
 107        """Realiza análises específicas nos dados de vendas"""
 108        print("\nANALISE DE DADOS DE VENDAS")
 109        print("=*80")
 110
 111        # Estatísticas básicas
 112        print("\nESTATÍSTICAS GERAIS:")
 113        print(f"Total de vendas: R$ {df['valor_venda'].sum():.2f}")
 114        print(f"Média de vendas: R$ {df['valor_venda'].mean():.2f}")
 115        print(f"Maior venda: R$ {df['valor_venda'].max():.2f}")
 116        print(f"Menor venda: R$ {df['valor_venda'].min():.2f}")
 117
 118        # Vendas por categoria
 119        print("\nVENDAS POR CATEGORIA:")
 120        vendas_categoria = df.groupby('categoria')['valor_venda'].agg(['sum', 'count'])
 121        vendas_categoria['sum'] = vendas_categoria['sum'].round(2)
 122        print(vendas_categoria)
 123
 124        # Vendas por mês
 125        print("\nVENDAS POR MÊS:")
 126        vendas_mes = df.groupby('mes')['valor_venda'].sum().round(2)
 127        print(vendas_mes)
 128
 129        return {
 130            'vendas_categoria': df.groupby('categoria')['valor_venda'].sum(),
 131            'vendas_mes': df.groupby('mes')['valor_venda'].sum(),
 132            'contagem_categoria': df['categoria'].value_counts(),
 133        }
 134
 135    # Passo 4: Visualização dos dados
 136    def visualizar_dados(df, analyses):
 137        """Cria visualizações dos dados de vendas"""
 138        print("\n CRIANDO VISUALIZAÇÕES ...")
 139
 140        # Configuração do layout dos gráficos
 141        fig, axes = plt.subplots(2, 2, figsize=(15, 12))
 142        fig.suptitle('Análise de Vendas - Ano 2023', fontsize=16, fontweight='bold', color="#2E4053")
 143
 144
  
```

=> Figura 1: Print da tela com o código 1º trecho.

```

  File Edit Selection View Go ... ↵ atividades
  unit_three.lesson_four.data_visualization.py X
  U3_A4_VISUALIZACAO_DE_DADOS_EM_PYTHON > unit_three.lesson_four.data_visualization.py > visualizar_dados
  137 def visualizar_dados(df, analises):
  138
  139     # Configuração do layout dos gráficos
  140     fig, axes = plt.subplots(2, 2, figsize=(15, 12))
  141     fig.suptitle('Análise de Vendas - Ano 2023', fontsize=16, fontweight='bold', color="#2E6A05")
  142
  143     cores_pizza = ['#00FFFF', '#000000', '#FFFF00']
  144     cores_barras = ['#00FFFF', '#000000', '#0000FF']
  145     cores_boxplot = ['#00FFFF', '#000000', '#FFFF00']
  146
  147     # Gráfico 1: Vendas por Categoria (Pizza)
  148     categorias = analises['vendas_categoria'].index
  149     valores = analises['vendas_categoria'].values
  150
  151     axes[0, 0].pie(valores, labels=categorias, autopct='%1.1f%%', colors=cores_pizza, textprops={'fontsize': 10,
  152     'fontweight': 'bold', 'color': 'black'})
  153     axes[0, 0].set_title('Distribuição de Vendas por Categoria', fontweight='bold', fontsize=12)
  154
  155     # Gráfico 2: Vendas por Mês (Barras)
  156     meses = ['Jan', 'Fev', 'Mar', 'Abr', 'Mai', 'Jun', 'Jul', 'Ago', 'Set', 'Out', 'Nov', 'Dez']
  157     vendas_mes_completo = analises['vendas_mes'].reindex(range(1, 13), fill_value=0)
  158
  159     bars = axes[0, 1].bar(meses, vendas_mes_completo.values, color=cores_barras, alpha=0.8)
  160     axes[0, 1].set_title('Valor das Vendas Mensais', fontweight='bold', fontsize=12)
  161     axes[0, 1].set_ylabel('Valor das Vendas (R$)', fontweight='bold')
  162     axes[0, 1].set_xlabel('Mês', fontweight='bold')
  163     axes[0, 1].xaxis.set_tick_params(axis='x', rotation=45)
  164     axes[0, 1].grid(True, alpha=0.3)
  165
  166
  167     # Adicionar valores nas barras
  168     for bar in bars:
  169         height = bar.get_height()
  170         if height > 0:
  171             axes[0, 1].text(bar.get_x(), bar.get_width()/2., height + 40,
  172                             f'R${height:.0f}', ha='center', va='bottom',
  173                             fontweight='bold', fontsize=9)
  174
  175     # Gráfico 3: Distribuição de Vendas (Boxplot)
  176     boxplot = sns.boxplot(data=df, x='categoria', y='valor_venda', ax=axes[1, 0], palette=cores_boxplot)
  177     axes[1, 0].set_title('Distribuição de Valores por Categoria', fontweight='bold', fontsize=12)
  178     axes[1, 0].set_ylabel('Valor da Venda (R$)', fontweight='bold')
  179     axes[1, 0].set_xlabel('Categoria', fontweight='bold')
  180     axes[1, 0].grid(True, alpha=0.3)
  181
  182     # Gráfico 4: Contagem de Vendas por Categoria (Barras)
  183     countplot = sns.countplot(data=df, x='categoria', ax=axes[1, 1], palette=cores_boxplot, hue='categoria', legend=False)
  184     axes[1, 1].set_title('Número de Vendas por Categoria', fontweight='bold', fontsize=12)
  185     axes[1, 1].set_ylabel('Número de Vendas', fontweight='bold')
  186     axes[1, 1].set_xlabel('Categoria', fontweight='bold')
  187     axes[1, 1].grid(True, alpha=0.3)
  188
  189     # Adicionar valores nas barras do countplot
  190     for container in countplot.containers:
  191         for bar in container:
  192             height = bar.get_height()
  193             axes[1, 1].text(bar.get_x(), bar.get_width()/2., height + 0.1,
  194                             f'{int(height)}', ha='center', va='bottom',
  195                             fontweight='bold', fontsize=10)
  196
  197     plt.tight_layout()
  198     plt.show()
  199
  200     # Gráfico adicional: Tendência temporal
  201     plt.figure(figsize=(12, 8))
  202     df.groupby('data_venda')['valor_venda'].sum().plot(
  203         marker='o', color='#00FFFF', linewidth=3, markersize=8
  204     )
  205     plt.title('Tendência de Vendas ao Longo do Tempo', fontweight='bold', fontsize=14)
  206     plt.ylabel('Valor de Vendas (R$)', fontweight='bold')
  207     plt.xlabel('Data', fontweight='bold')
  208     plt.xticks(rotation=45)
  209     plt.tight_layout()
  210     plt.show()
  211
  212     # Passo 5: Conclusão e insights
  213     def gerar_insights(df, analises):
  214         "Gera insights baseados na análise dos dados"
  215         print("\n" + "="*80)
  216         print("INSIGHTS E RECOMENDAÇÕES")
  217         print(" = *80")
  218
  219         # Encontrar o mês com maior venda
  220         mes_max_venda = analises['vendas_mes'].idxmax()
  221         valor_max_venda = analises['vendas_mes'].max()
  222
  223         # Encontrar a categoria mais vendida
  224         categoria_max_venda = analises['vendas_categoria'].idxmax()
  225         valor_categoria_max = analises['vendas_categoria'].max()
  226
  227         print(f"\n**Insight 1:** Mês de maior faturamento foi {mes_max_venda}º mês com R$ {valor_max_venda:.2f}")
  228         print(f"\n**Insight 2:** Categoria '{categoria_max_venda}' foi a mais lucrativa (R$ {valor_categoria_max:.2f})")
  229
  230         # Calcular participação percentual
  231         participacao = (analises['vendas_categoria'] / analises['vendas_categoria'].sum() * 100).round(1)
  232
  233         print(f"\n**Insight 3:** Participação por categoria:")
  234         for categoria in participacao.items():
  235             print(f" - {categoria}: ({percentual}%)")
  236
  237         # Recomendações
  238         print("Recomendações para as recomendações Estratégicas:")
  239         print("1. Investir mais em marketing para a categoria de maior faturamento")
  240         print("2. Analisar sazonalidade para planejar estoques")
  241         print("3. Desenvolver promoções para categorias com menor participação")
  242         print("4. Expander portfólio na categoria mais lucrativa")
  243         print("5. Focar em estratégias para os meses de menor performance")
  244
  245     # Função principal
  246     def main():
  247         "Função principal que executa todo o fluxo de análise"
  248         print("\nSISTEMA DE ANÁLISE DE DADOS DE VENDAS")
  249         print(" = *80")
  250
  251         # Passo 1: Criar banco de dados
  252         criar_banco_dados()
  253
  254         # Passo 2: Carregar dados
  255         df_vendas = carregar_dados()
  256
  257         if df_vendas is not None:
  258             # Exibir primeiras linhas
  259             print("\n PRIMEIRAS LINHAS DO DATASET:")
  260             print(df_vendas.head())
  261
  262             # Passo 3: Analisar dados
  263             resultados_analise = analisar_dados(df_vendas)
  264
  265             # Passo 4: Visualizar dados
  266             visualizar_dados(df_vendas, resultados_analise)
  267
  268             # Passo 5: Gerar insights
  269             gerar_insights(df_vendas, resultados_analise)
  270
  271         # Executar o programa
  272         if __name__ == "__main__":
  273             # Instalar dependências se necessário (no Colab)
  274             try:
  275                 import seaborn
  276             except ImportError:
  277                 print("Instalando bibliotecas necessárias...")
  278                 import subprocess
  279                 subprocess.run(["pip", "install", "seaborn", "matplotlib", "pandas"])
  280
  281         main()
  
```

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Python: unit_three_lesson_four_data_visualization +

20:57:18 atividades 8 72ms
20:57:18 atividades 8 72ms
C:/Users/edrad/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/edrad/OneDrive/Área de Trabalho/ed/Anhangue_a_Eng_Software/2º_semestre/5º-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!
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

VENDAS POR CATEGORIA:

categoria	sum	count
Eletrônicos	5650.0	6
Livros	900.0	4
Roupas	1700.0	4

VENDAS POR MÊS:

mes	valor_venda
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

CREANDO VISUALIZAÇÕES...

INSIGHTS E RECOMENDAÇÕES

=====
Insight 1: Mês de maior faturamento foi 1º mês com R\$ 1600.00
Insight 2: Categoria 'Eletrônicos' foi a mais lucrativa (R\$ 5650.00)
Insight 3: Participação por categoria:

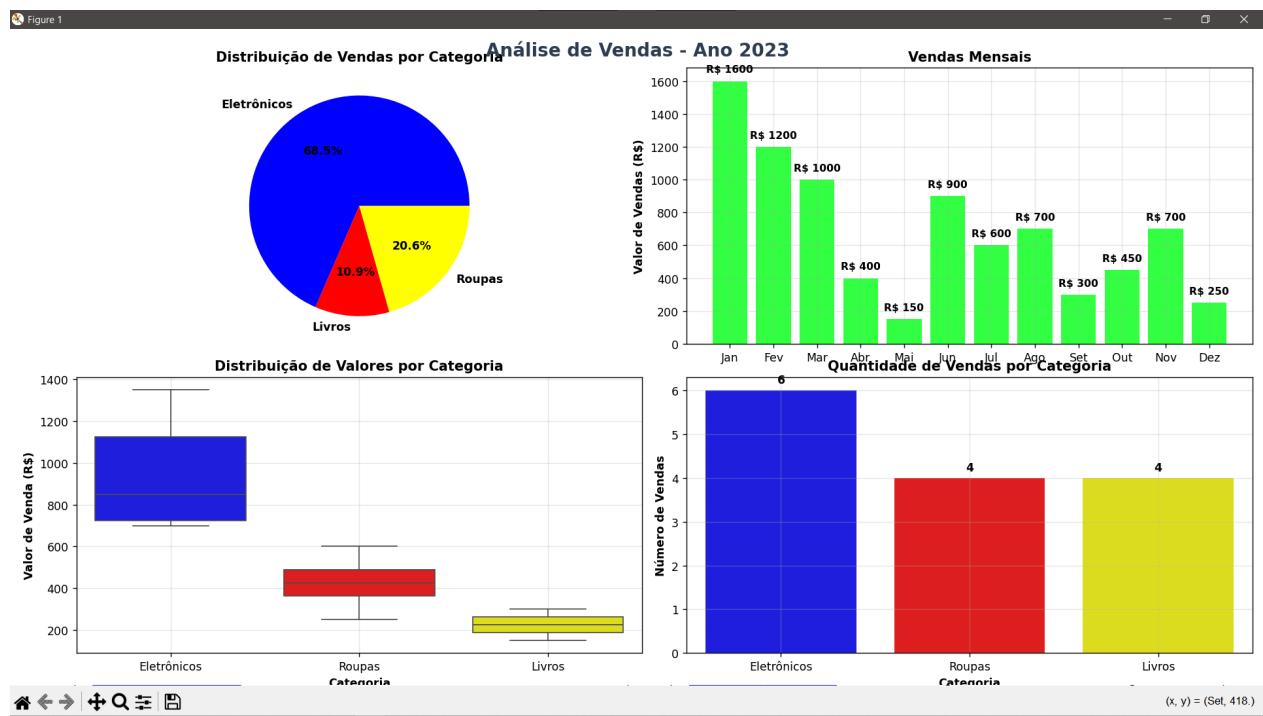
- Eletrônicos: 68.5%
- Livros: 10.9%
- Roupas: 20.6%

Recomendações Estratégicas:

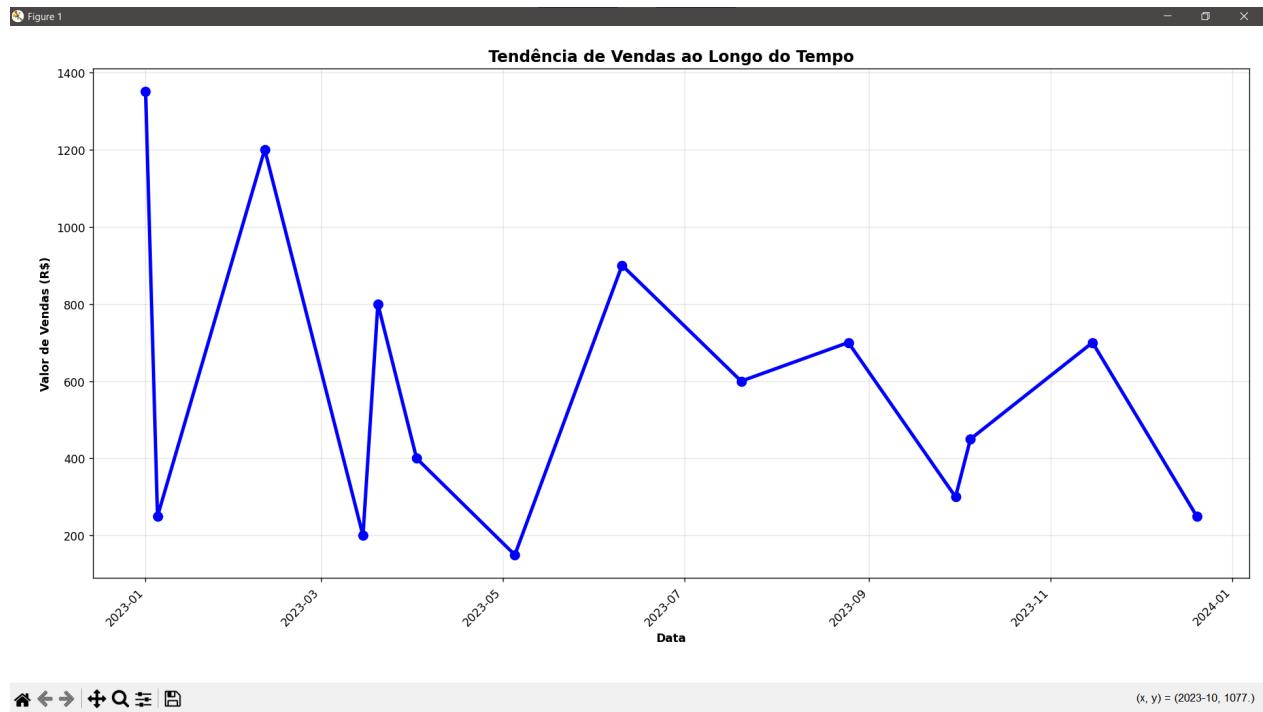
- Investir mais em marketing para a categoria de maior faturamento
- Analizar sazonalidade para planejar estoques
- Desenvolver promoções para categorias com menor participação
- Expandir portfólio na categoria mais lucrativa
- Focar em estratégias para os meses de menor performance

20:57:39 atividades 8 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.



1. Tratamento de Erros:

```
try:  
    # Operações críticas  
except Exception as e:  
    print(f"X 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 açãoáveis.



O sistema transforma dados brutos em:

- Informações estruturadas. Visualizações comprehensíveis.
- Insights açãoáveis. 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.