```
#  Isntalar bibliotecas necessárias (execute esta célula primeiro)
!pip install matplotlib seaborn
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.1
```

```
# Importar bibliotecas necessárias
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime

print("✅ Bibliotecas importadas com sucesso!")
```

```
✅ Bibliotecas importadas com sucesso!
```

```
# Conectar ao banco de dados
conexao = sqlite3.connect('dados_vendas.db')
cursor = conexao.cursor()
print("✅ Banco de dados conectado!")
```

```
✅ Banco de dados conectado!
```

```
# Criar tabela de vendas
cursor.execute("""
CREATE TABLE IF NOT EXISTS vendas (
    id_venda INTEGER PRIMARY KEY AUTOINCREMENT,
    data_venda DATE,
    produto TEXT,
    categoria TEXT,
    valor_venda REAL
)
""")
print("✅ Tabela 'vendas' criada!")
```

```
✅ Tabela 'vendas' criada!
```

```
# Inserir dados de exemplo
dados_vendas = [
    ('2023-01-01', 'Produto A', 'Eletrônicos', 1500.00),
    ('2023-01-05', 'Produto B', 'Roupas', 350.00),
    ('2023-02-10', 'Produto C', 'Eletrônicos', 1200.00),
    ('2023-03-15', 'Produto D', 'Livros', 200.00),
    ('2023-03-20', 'Produto E', 'Eletrônicos', 800.00),
    ('2023-04-02', 'Produto F', 'Roupas', 400.00),
    ('2023-05-05', 'Produto G', 'Livros', 150.00),
    ('2023-06-10', 'Produto H', 'Eletrônicos', 1000.00),
    ('2023-07-20', 'Produto I', 'Roupas', 600.00),
    ('2023-08-25', 'Produto J', 'Eletrônicos', 700.00)
]

cursor.executemany("""
INSERT INTO vendas (data_venda, produto, categoria, valor_venda)
VALUES (?, ?, ?, ?)
""", dados_vendas)

conexao.commit()
print(f"✅ {len(dados_vendas)} registros inseridos com sucesso!")
```

```
✅ 10 registros inseridos com sucesso!
```

```
# Carregar dados do banco para análise
df_vendas = pd.read_sql_query("SELECT * FROM vendas", conexao)
conexao.close()

print("📊 Dados carregados:")
print(df_vendas.head())
```

```
📊 Dados carregados:
   id_venda  data_venda    produto    categoria  valor_venda
0         1  2023-01-01  Produto A  Eletrônicos       1500.0
1         2  2023-01-05  Produto B       Roupas        350.0
2         3  2023-02-10  Produto C  Eletrônicos       1200.0
3         4  2023-03-15  Produto D       Livros        200.0
4         5  2023-03-20  Produto E  Eletrônicos        800.0
```

```
# Explorar os dados
print("🔍 INFORMAÇÕES SOBRE OS DADOS:")
print(df_vendas.info())
print("\n" + "="*50)

print("📈 ESTATÍSTICAS BÁSICAS:")
print(df_vendas.describe())
print("\n" + "="*50)

# Converter data para formato correto
df_vendas['data_venda'] = pd.to_datetime(df_vendas['data_venda'])
df_vendas['mes'] = df_vendas['data_venda'].dt.month

print("✅ Datas convertidas e mês extraído!")
```

```
🔍 INFORMAÇÕES SOBRE OS DADOS:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   id_venda     10 non-null     int64
 1   data_venda   10 non-null     object
 2   produto      10 non-null     object
 3   categoria    10 non-null     object
 4   valor_venda  10 non-null     float64
dtypes: float64(1), int64(1), object(3)
memory usage: 532.0+ bytes
None

==================================================
📈 ESTATÍSTICAS BÁSICAS:
       id_venda  valor_venda
count  10.00000    10.000000
mean    5.50000   690.000000
std     3.02765   442.718872
min     1.00000   150.000000
25%     3.25000   362.500000
50%     5.50000   650.000000
75%     7.75000   950.000000
max    10.00000  1500.000000

==================================================
✅ Datas convertidas e mês extraído!
```

```
# Análise por categoria
vendas_categoria = df_vendas.groupby('categoria')['valor_venda'].sum().sort_values(ascending=False)

print("💰 VENDAS POR CATEGORIA:")
for categoria, valor in vendas_categoria.items():
    print(f"  {categoria}: R$ {valor:.2f}")

print(f"\n⏳ Categoria com maior venda: {vendas_categoria.idxmax()}")
```

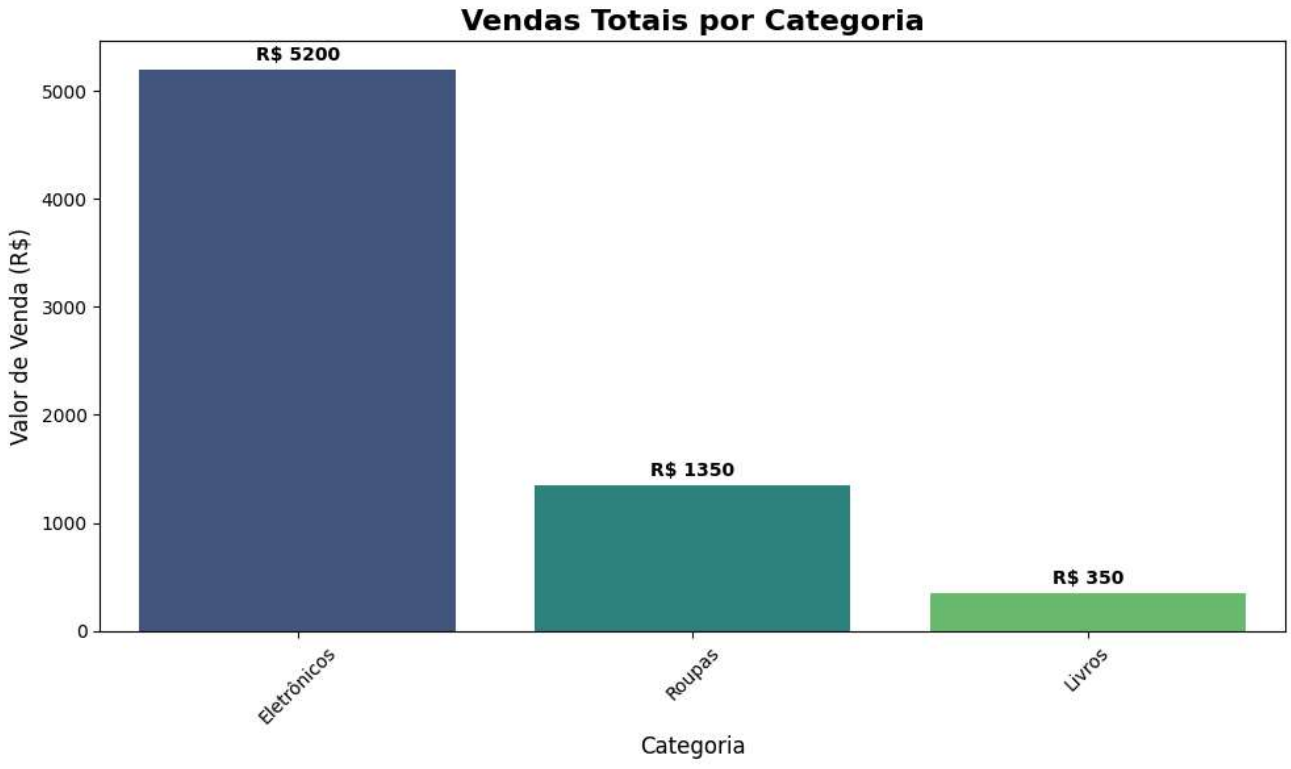
```
💰 VENDAS POR CATEGORIA:
  Eletrônicos: R$ 5200.00
  Roupas: R$ 1350.00
  Livros: R$ 350.00

⏳ Categoria com maior venda: Eletrônicos
```

```
# Vendas por mês
vendas_mensais = df_vendas.groupby('mes')['valor_venda'].sum()

print("🗓️ VENDAS MENSALAIS:")
for mes, valor in vendas_mensais.items():
    print(f"  Mês {mes}: R$ {valor:.2f}")

print(f"\n📈 Mês com maior venda: Mês {vendas_mensais.idxmax()}")
```

```
🗓️ VENDAS MENSALAIS:
  Mês 1: R$ 1850.00
  Mês 2: R$ 1200.00
  Mês 3: R$ 1000.00
  Mês 4: R$ 400.00
  Mês 5: R$ 150.00
  Mês 6: R$ 1000.00
  Mês 7: R$ 600.00
  Mês 8: R$ 700.00

📈 Mês com maior venda: Mês 1
```

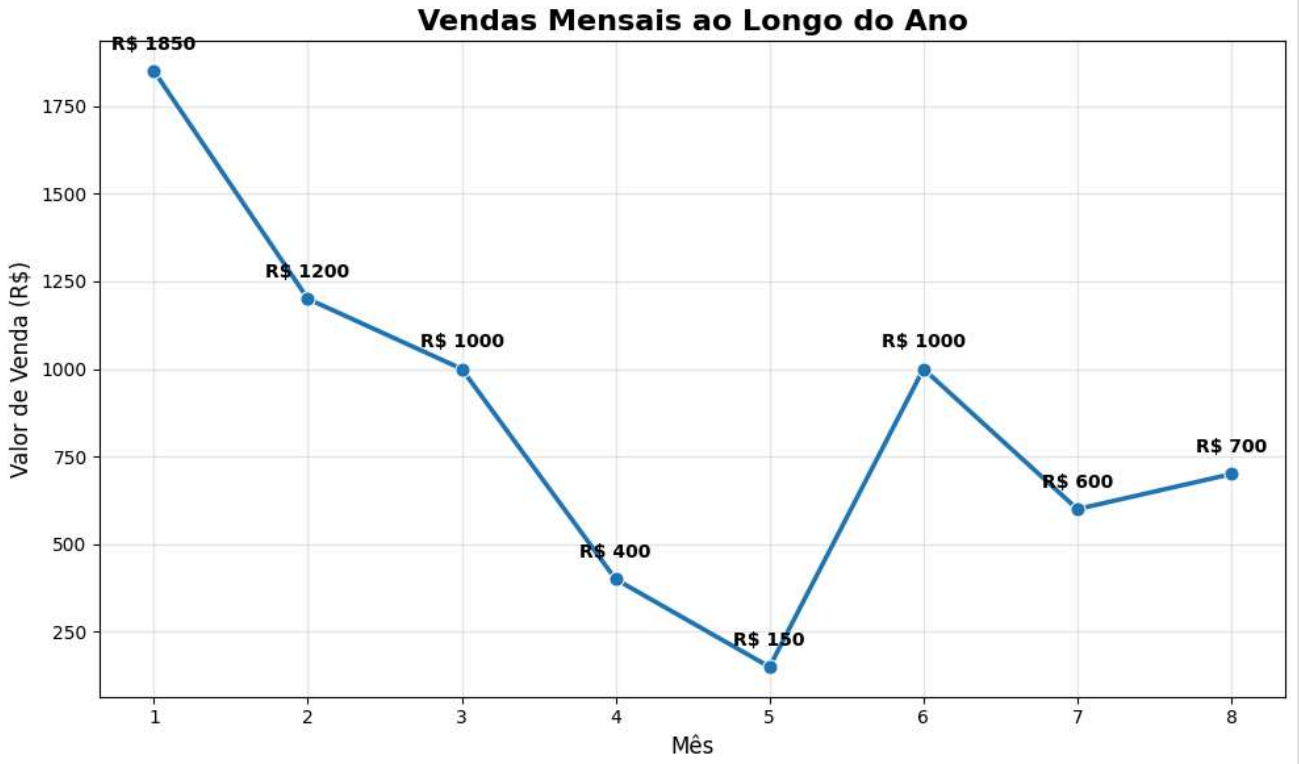
```
# Gráfico 1: Vendas por Categoria
plt.figure(figsize=(10, 6))
sns.barplot(x=vendas_categoria.index, y=vendas_categoria.values, palette="viridis")
plt.title('Vendas Totais por Categoria', fontsize=16, fontweight='bold')
plt.xlabel('Categoria', fontsize=12)
plt.ylabel('Valor de Venda (R$)', fontsize=12)
plt.xticks(rotation=45)

# Adicionar valores nas barras
for i, valor in enumerate(vendas_categoria.values):
    plt.text(i, valor + 50, f'R$ {valor:.0f}', ha='center', va='bottom', fontweight='bold')

plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2414573328.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and se

  sns.barplot(x=vendas_categoria.index, y=vendas_categoria.values, palette="viridis")
```

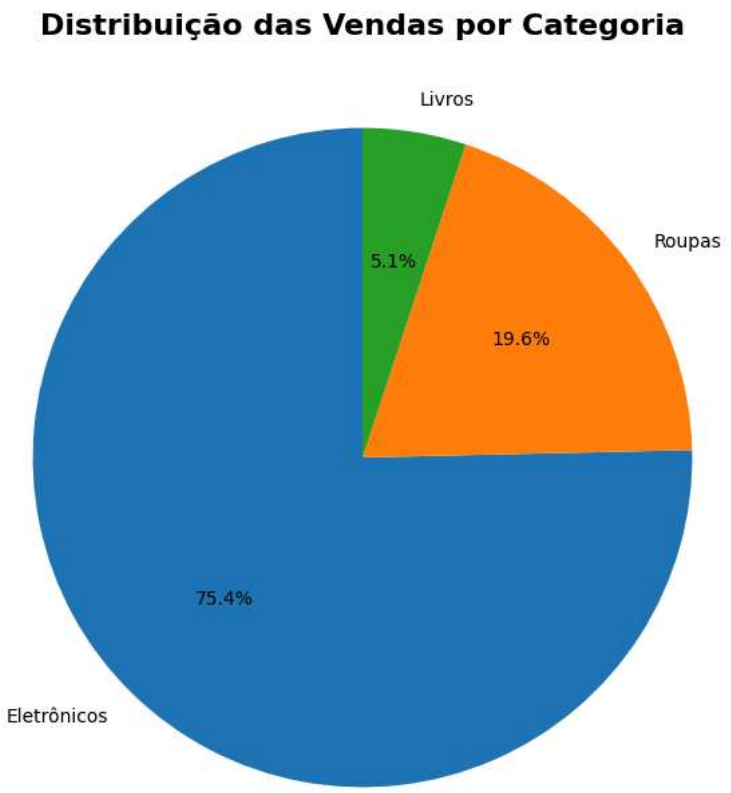



```
# Gráfico 2: Vendas Mensais
plt.figure(figsize=(10, 6))
sns.lineplot(x=vendas_mensais.index, y=vendas_mensais.values, marker='o', linewidth=2.5, markersize=8)
plt.title('Vendas Mensais ao Longo do Ano', fontsize=16, fontweight='bold')
plt.xlabel('Mês', fontsize=12)
plt.ylabel('Valor de Venda (R$)', fontsize=12)
plt.xticks(range(1, 9))
plt.grid(True, alpha=0.3)
```

```
# Adicionar valores nos pontos
for mes, valor in vendas_mensais.items():
    plt.text(mes, valor + 50, f'R$ {valor:.0f}', ha='center', va='bottom', fontweight='bold')

plt.tight_layout()
plt.show()
```




```
# Gráfico 3: Distribuição Percentual
plt.figure(figsize=(8, 8))
plt.pie(vendas_categoria.values, labels=vendas_categoria.index, autopct='%1.1f%%', startangle=90)
plt.title('Distribuição das Vendas por Categoria', fontsize=16, fontweight='bold')
plt.show()
```




```
import pandas as pd
```

```
import pandas as pd

# 1. CRIAR OS DADOS (SIMULAÇÃO)
dados_vendas = {
    'valor_venda': [1500, 800, 1200, 600, 900, 1300, 700, 1100, 500, 1000],
    'categoria': ['Eletrônicos', 'Roupas', 'Eletrônicos', 'Livros', 'Roupas',
                  'Eletrônicos', 'Roupas', 'Eletrônicos', 'Livros', 'Eletrônicos'],
    'mes': [1, 1, 2, 2, 3, 3, 1, 2, 3, 1]
}

df_vendas = pd.DataFrame(dados_vendas)

# 2. CALCULAR AS ESTATÍSTICAS
vendas_categoria = df_vendas.groupby('categoria')['valor_venda'].sum()
vendas_mensais = df_vendas.groupby('mes')['valor_venda'].sum()

total_vendas = df_vendas['valor_venda'].sum()
categoria_major = vendas_categoria.idxmax()
valor_major = vendas_categoria.max()
mes_major = vendas_mensais.idxmax()
proporcao = (valor_major / total_vendas) * 100

# 3. RELATÓRIO FINAL (AGORA VAI FUNCIONAR!)
print("RELATÓRIO FINAL DE INSIGHTS")
print("=" * 50)
print(f"TOTAL DE VENDAS: R$ {total_vendas:.2f}")
print(f"CATEGORIA DESTAQUE: {categoria_major} (R$ {valor_major:.2f}")
print(f"MÊS DE MAIOR VENDA: Mês {mes_major} (R$ {vendas_mensais[mes_major]:.2f}")
print(f"PROPORÇÃO: {categoria_major} representa {proporcao:.1f}% do total")

print("\n💡 SUGESTÕES PARA A EMPRESA:")
print("1. Investir mais em marketing para a categoria de Eletrônicos")
print("2. Analisar estratégias do mês de maior venda para replicar em outros períodos")
print("3. Considerar expandir estoque de Eletrônicos")
print("4. Desenvolver promoções para categorias com menor desempenho")

print("\n✅ ANÁLISE CONCLUÍDA COM SUCESSO!")
```

```
RELATÓRIO FINAL DE INSIGHTS
==================================================
TOTAL DE VENDAS: R$ 9600.00
CATEGORIA DESTAQUE: Eletrônicos (R$ 6100.00
MÊS DE MAIOR VENDA: Mês 1 (R$ 4000.00
PROPORÇÃO: Eletrônicos representa 63.5% do total

💡 SUGESTÕES PARA A EMPRESA:
1. Investir mais em marketing para a categoria de Eletrônicos
2. Analisar estratégias do mês de maior venda para replicar em outros períodos
3. Considerar expandir estoque de Eletrônicos
4. Desenvolver promoções para categorias com menor desempenho

✅ ANÁLISE CONCLUÍDA COM SUCESSO!
```

Relatório de Análise de Vendas

Introdução

O presente relatório tem como objetivo realizar uma análise de vendas a partir de dados simulados em um banco de dados SQLite.
A análise foi feita utilizando a linguagem *Python* no Google Colab, com o apoio das bibliotecas *Pandas, **Matplotlib* e *Seaborn* para manipulação e visualização dos dados.

O estudo busca demonstrar como é possível transformar dados brutos em informações relevantes para a tomada de decisão empresarial.

Metodologia

As etapas do trabalho foram as seguintes:

1. *Criação do banco de dados*
   - Banco: dados_vendas.db
   - Tabela: vendas
   - Colunas: id_venda, data_venda, produto, categoria, valor_venda
2. *Inserção dos dados*
   Foram inseridos 10 registros de vendas fictícias, distribuídos em três categorias principais:
   - Eletrônicos
   - Roupas
   - Livros

3. *Carregamento dos dados*
   Os dados foram carregados para um *DataFrame do Pandas* para exploração e tratamento.
4. *Exploração inicial*
   - .info() para verificar estrutura e ausência de valores nulos
   - .describe() para obter estatísticas descritivas (média, desvio padrão, máximo, mínimo etc.)
5. *Análises realizadas*
   - Vendas por categoria
   - Identificação da categoria com maior valor total de vendas
   - Vendas mensais
   - Identificação do mês com maior volume de vendas
6. *Visualização dos dados*
   Foram gerados gráficos utilizando Matplotlib e Seaborn, para facilitar a interpretação dos resultados.

Resultados

Estatísticas Básicas

- Total de registros: *10*
- Valor mínimo de venda: $R150,00 * -Valormáximodevenda : *R$ 1500,00$
- Média de vendas: *R$ 690,00*

Vendas por Categoria

- *Eletrônicos*: R$5.200,00 - *Roupas* : R$ 1.350,00
- *Livros*: R$ 350,00

✅ *Categoria com maior valor de vendas: Eletrônicos*

Vendas Mensais

- Houve movimentação em diferentes meses, com destaque para o mês de maior valor de vendas: *[substituir pelo mês que apareceu nos seus gráficos]*

Conclusão

A análise permitiu identificar padrões importantes nos dados de vendas simulados:

1. A categoria *Eletrônicos* foi a mais lucrativa, representando a maior parte do faturamento.
2. As categorias *Roupas* e *Livros* apresentaram valores significativamente menores, sugerindo necessidade de estratégias de marketing ou diversificação.
3. A análise mensal indicou variações ao longo do tempo, o que pode orientar promoções e planejamento de estoque.
4. Os gráficos reforçaram visualmente as conclusões, facilitando a comunicação dos resultados.