

```

In [1]: # Indicium Light House- desafio Análise de Dados
## Procederemos a análise de dados, utilizando os arquivos csv disponibilizados
## Faremos de modo didático e gradual, utilizando o notebook que ficará disponível posteriormente no github
## Buscaremos abordar os dados de modo a responder as perguntas e endereçar os KPIs

In [2]: ### Iniciamos importando as libs necessárias para o desafio
import pandas as pd # manipulação de dados
import numpy as np # operações numéricas
import matplotlib.pyplot as plt # cria algumas visualizações
import seaborn as sns # gerar gráficos mais complexos
from datetime import datetime # manipulação de datas

In [3]: ## próximo passo é carregar os dados que serão utilizados, após feito o upload para o jupyter
# Carregamento dos datasets
caminho_dados = "Indicium_light_house/"

# Carregar cada dataset em um DataFrame
df_agencias = pd.read_csv(caminho_dados + "agencias.csv")
df_clientes = pd.read_csv(caminho_dados + "clientes.csv")
df_colab_agencia = pd.read_csv(caminho_dados + "colaborador_agencia.csv")
df_colaboradores = pd.read_csv(caminho_dados + "colaboradores.csv")
df_contas = pd.read_csv(caminho_dados + "contas.csv")
df_propostas = pd.read_csv(caminho_dados + "propostas_credito.csv")
df_transacoes = pd.read_csv(caminho_dados + "transacoes.csv")

In [40]: # agora precisamos realizar a primeir análise dos dados, visual e também quanto a dimensão dos dados
# Verificando as primeiras 15 linhas de cada dataset

print(df_transacoes.head(15))
print(df_propostas.head(15))
print(df_contas.head(15))
print(df_clientes.head(15))
print(df_agencias.head(15))
print(df_colab_agencia.head(15))
print(df_colaboradores.head(15))

11      -175.0
12      -130.0
13      -140.0
14      -35.0
cod_proposta  cod_cliente  cod_colaborador  data_entrada_proposta \
0          116          338                  1  2014-05-30 18:43:12 UTC
1          715           45                  1  2021-06-17 13:09:32 UTC
2          755          494                  1  2021-08-10 04:24:02 UTC
3          953          381                  1  2011-02-23 07:57:58 UTC
4         1046          176                  1  2010-11-15 16:03:28 UTC
5         1171          732                  1  2020-09-04 21:51:58 UTC
6         1434          560                  1  2020-01-31 00:02:49 UTC
7         1744          159                  1  2016-03-04 20:31:58 UTC
8         1864          776                  1  2019-09-13 10:21:31 UTC
9         1928          421                  1  2014-05-08 15:43:27 UTC
10        41           608                  2  2021-03-04 08:08:46 UTC
11        416          342                  2  2015-10-11 14:28:01 UTC
12        581           81                  2  2022-03-19 23:43:49 UTC
13        678          788                  2  2011-02-03 06:12:10 UTC
14       1272          673                  2  2012-10-13 07:02:05 UTC

In [4]: ### Ficou claro que observar todos os datasets em ambiente python demonstra ser contraproducente
## Melhor criar uma função simples que utiliza um dicionário em que as chaves são categorias e os valores são os datasets
# Assim fica mais simples e rápido de codar para exibir informações básicas sobre os datasets

print("\nInformações sobre os datasets carregados:")
dfs = {
    "Agências": df_agencias,
    "Clientes": df_clientes,
    "Colaboradores": df_colab_agencia,
    "Colaboradores Agência": df_colaboradores,
    "Contas": df_contas,
    "Propostas Crédito": df_propostas,
    "Transações": df_transacoes,
}
# Exibir as primeiras informações descritivas de cada dataset
for nome, df in dfs.items():
    print(f"\n{nome} - Linhas: {df.shape[0]}, Colunas: {df.shape[1]}")
    print(df.head()) # Exibir as primeiras linhas do dataset
    print(f"\nInfo descritivas:")
    print(df.info())
    print()

4  tipo_conta      999 non-null  object
5  data_abertura   999 non-null  object
6  saldo_total     999 non-null  float64
7  saldo_disponivel  999 non-null  float64
8  data_ultimo_lancamento  999 non-null  object
dtypes: float64(2), int64(4), object(3)
memory usage: 70.4+ KB
None

Propostas Crédito - Linhas: 2000, Colunas: 12
cod_proposta  cod_cliente  cod_colaborador  data_entrada_proposta \
0          116          338                  1  2014-05-30 18:43:12 UTC
1          715           45                  1  2021-06-17 13:09:32 UTC
2          755          494                  1  2021-08-10 04:24:02 UTC
3          953          381                  1  2011-02-23 07:57:58 UTC
4         1046          176                  1  2010-11-15 16:03:28 UTC

    taxa_juros_mensal  valor_proposta  valor_financiamento  valor_entrada \
0            0.0194      36199.950355          50032.03      13832.079645
1            0.0131      12897.538285          19848.55      6951.011715

In [5]: # Vamos a primeira pergunta:
## Qual trimestre tem, em média, mais transações autorizadas e qual tem, também em média, maior volume movimentado?

```

```
## Quais trimestres temos em média, mais transações aprovadas e qual temos com menor em média, maior volume movimentado.
```

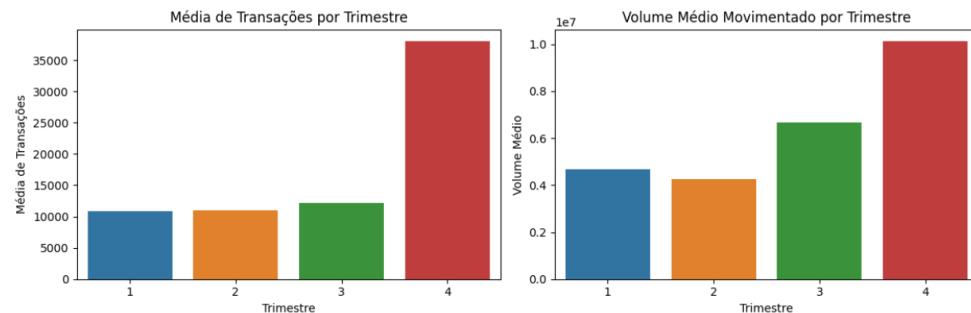
```
### Para responder a essa pergunta, precisamos seguir os seguintes passos:  
### Transformar os dados das colunas de data em formato datetime para melhor manipular  
### Criar uma coluna de trimestre na tabela de transações  
### Agrupar as transações por trimestre
```

```
In [6]: # Convertendo a coluna de data para o tipo datetime  
df_transacoes['data_transacao'] = pd.to_datetime(df_transacoes['data_transacao'])  
  
# Criando a coluna de trimestre  
df_transacoes['trimestre'] = df_transacoes['data_transacao'].dt.quarter
```

```
In [7]: # Agrupando as transações por trimestre e calculando a média usando o método groupby e .agg()  
transacoes_por_trimestre = df_transacoes.groupby('trimestre').agg(  
    media_transacoes=('cod_transacao', 'count'),  
    volume_medio=('valor_transacao', 'sum'))  
.reset_index()  
  
# Encontrando os trimestres com os maiores valores  
trimestre_mais_transacoes = transacoes_por_trimestre.loc[transacoes_por_trimestre['media_transacoes'].idxmax(), 'trimestre']  
trimestre_maior_volume = transacoes_por_trimestre.loc[transacoes_por_trimestre['volume_medio'].idxmax(), 'trimestre']  
  
print(f'O 4º trimestre com mais transações em média é o {trimestre_mais_transacoes}º trimestre.')  
print(f'O 4º trimestre com maior volume movimentado em média é o {trimestre_maior_volume}º trimestre.')  
  
O 4º trimestre com mais transações em média é o 4º trimestre.  
O 4º trimestre com maior volume movimentado em média é o 4º trimestre.
```

```
In [8]: # o quarto trimestre apresenta maior volume e mais transações, o que implica não só que são realizados mais negócios  
# como também que os negócios tendem a ser mais substanciais, ou seja, as pessoas precisam de mais dinheiro no final do ano
```

```
In [9]: # Visualizando os resultados  
fig, ax = plt.subplots(1, 2, figsize=(12, 4))  
  
sns.barplot(x='trimestre', y='media_transacoes', data=transacoes_por_trimestre, ax=ax[0])  
ax[0].set_title('Média de Transações por Trimestre')  
ax[0].set_xlabel('Trimestre')  
ax[0].set_ylabel('Média de Transações')  
  
sns.barplot(x='trimestre', y='volume_medio', data=transacoes_por_trimestre, ax=ax[1])  
ax[1].set_title('Volume Médio Movimentado por Trimestre')  
ax[1].set_xlabel('Trimestre')  
ax[1].set_ylabel('Volume Médio')  
  
plt.tight_layout()  
plt.show()
```



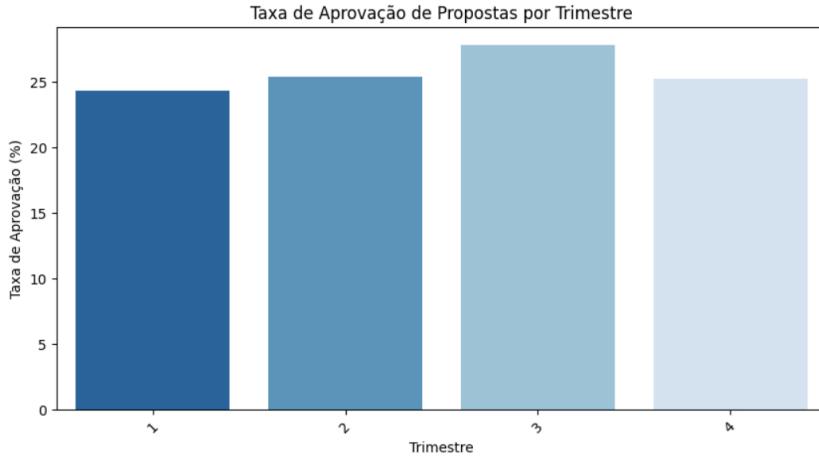
```
In [ ]:
```

```
In [10]: # Sabendo que o 4º trimestre tende a ser mais movimentado e volumoso, podemos responder sobre a taxa de aprovação  
# podemos chegar ao seguinte, número absoluto de propostas que se realizaram em forma de negócio / valor total de propostas  
# podemos usar função lambda para encontrar os valores de Aprovada e somar estes valores inserindo o resultado em uma variável  
# na tabela de propostas existe o status da proposta que apresenta valores como: em análise, aprovada..  
# Contar o total de propostas e aprovações por trimestre  
  
# Converter a coluna de data para o tipo datetime no DataFrame df_propostas  
df_propostas['data_entrega_proposta'] = pd.to_datetime(df_propostas['data_entrega_proposta'])  
  
# Criar a coluna de trimestre no DataFrame df_propostas  
df_propostas['trimestre'] = df_propostas['data_entrega_proposta'].dt.quarter  
  
# Contar o total de propostas e as aprovações por trimestre  
propostas_por_trimestre = df_propostas.groupby('trimestre').agg(  
    total_propostas=('cod_proposta', 'count'),  
    propostas_aprovadas=('status_proposta', lambda x: (x == 'Aprovada').sum()))  
.reset_index()  
  
# Calcular a taxa de aprovação por trimestre  
propostas_por_trimestre['taxa_aprovacao'] = (propostas_por_trimestre['propostas_aprovadas']) / propostas_por_trimestre['total_propostas']  
  
# Combinar com as transações para analisar a relação entre aprovações e transações  
df_trimestre = transacoes_por_trimestre.merge(propostas_por_trimestre, on='trimestre', how='left')  
  
# Mostrar o trimestre com a maior taxa de aprovação  
df_trimestre_sorted = df_trimestre.sort_values(by='taxa_aprovacao', ascending=False)  
trimestre_maior_aprovacao = df_trimestre_sorted.iloc[0]  
  
print("\nTrimestre com maior taxa de aprovação:")
print(f'Trimestre: {int(trimestre_maior_aprovacao['trimestre'])}')
print(f'Média de transações: {trimestre_maior_aprovacao['media_transacoes']:.0f}')
print(f'Volume médio movimentado: R$ {trimestre_maior_aprovacao['volume_medio']:.2f}')
print(f'Total de propostas: {int(trimestre_maior_aprovacao['total_propostas'])}')
print(f'Propostas aprovadas: {int(trimestre_maior_aprovacao['propostas_aprovadas'])}')
print(f'Taxa de aprovação: {trimestre_maior_aprovacao['taxa_aprovacao']:.2f}%")
```

```
Trimestre com maior taxa de aprovação:  
Trimestre: 3  
Média de transações: 12190  
Volume médio movimentado: R$ 6,654,486.96
```

```
Total de propostas: 522  
Propostas aprovadas: 145  
Taxa de aprovação: 27.78%
```

```
In [51]: # Visualizar taxa de aprovação por trimestre  
plt.figure(figsize=(10,5))  
sns.barplot(x=df_trimestre['trimestre'].astype(str), y=df_trimestre['taxa_aprovacao'], palette="Blues_r")  
plt.title("Taxa de Aprovação de Propostas por Trimestre")  
plt.xlabel("Trimestre")  
plt.ylabel("Taxa de Aprovação (%)")  
plt.xticks(rotation=45)  
plt.show()
```



```
In [ ]: #Pergunta do analista meses que contém R no seu nome e volume de transacoes  
# A despeito de parecer uma hipótese absurda, podemos responder aos nobre colega usando de  
# referencia estatístico de correlação, ou seja, o quanto um fator influencia em outro, no caso em específico  
# quanto o fato de um mes ter a letra R ou não afeta no numero de negócios.  
# como apontam os dados o 4 trimestre segue como o top em termos de volume de transacao, todos os meses deste trimestre  
# possuem a Letra R em seu nome, o que aparenta falsa relacao de causalidade.  
# o primeiro trimestre tambem possue R em todos os nomes dos meses, contudo, como apontam os dados, fica no bottom da analise  
# Portanto, sem perder o tempo com formulas ou dados ou graficos complicados, basta informar ao colega que não passa de uma  
#coicidencia comica.
```

```
In [ ]: #Pergunta 4: André Tech solicitou dados públicos para enriquecer a base de dados do Banvic  
# ampliar as possibilidades de análise, considerando principalmente a necessidade atual apresentada.  
# em que pese o IPCA ser um bom termômetro da inflação e economia, ele reflete uma realidade menos acurada do que o INPC  
# O IPCA indice de preços ao consumidor amplo, leva em conta mais itens supérfluos, logo, dialoga com população mais abastada  
# O INPC por seu turno trata de itens mais necessários e de uso comum, ele mede mais fidetidamente os impactos da inflação  
# O IPCA , por ser mais amplo, acaba diluindo mais as implicações de dinâmicas de preço na economia  
# já que o interesse seria enriquecer o banco de dados, melhor usar o INPC disponível por API no site do IBGE  
# disponibilizo o arquivo das séries históricas do INPC no mesmo link do projeto que está no github
```

```
In [ ]: #vamos demonstrar alguns KPIs - Key Performance Indicators que podem ser úteis ao tomador de decisões  
# Taxa de aprovação de crédito - o coração do negócio - pode indicar melhores oportunidades de vender o produto  
# Volume de transações por agência, possível integrar os dados de agências e de negócios realizados, encontrando qual  
# unidade vende mais e a partir daí tentar replicar os procedimentos das demais agências aumentando o aproveitamento  
# no mesmo sentido, quais os funcionários vendem mais, ou seja, quais profissionais podem ser os diretores ou gerentes  
# no propósito de que liderem times e campanhas para melhorar os índices e treinar mais colaboradores.
```

```
In [11]: # 1. Taxa de Aprovação de Crédito  
df_propostas['aprovada'] = df_propostas['status_proposta'].apply(lambda x: 1 if x == 'Aprovada' else 0)  
taxa_aprovacao = df_propostas['aprovada'].mean() * 100  
print(f'Taxa de Aprovação de Crédito: {taxa_aprovacao:.2f}%')
```

Taxa de Aprovação de Crédito: 25.70%

```
In [59]: # 2. Volume de Transações por Agência  
transacoes_por_agencia = df_transacoes.groupby('num_conta').agg(  
    total_transacoes=('valor_transacao', 'count'),  
    volume_total=('valor_transacao', 'sum'))  
transacoes_por_agencia.reset_index()  
  
df_contas_agencias = df_contas[['num_conta', 'cod_agencia']].merge(transacoes_por_agencia, on='num_conta', how='left')  
df_agencias_transacoes = df_contas_agencias.merge(df_agencias[['cod_agencia', 'nome', 'cidade']], on='cod_agencia', how='left')  
agencias_transacoes = df_agencias_transacoes.groupby(['nome', 'cidade']).agg(  
    total_transacoes=('total_transacoes', 'sum'),  
    volume_total=('volume_total', 'sum'))  
agencias_transacoes.reset_index()  
  
print("Top 10 Agências com Maior Volume de Transações:")  
print(agencias_transacoes.sort_values(by='volume_total', ascending=False).head(10))
```

	nome	cidade	total_transacoes	volume_total
1	Agência Digital	São Paulo	33167	11579143.34
4	Agência Matriz	São Paulo	8610	3549901.76
9	Agência Tatuapé	São Paulo	7156	3194382.65
0	Agência Campinas	Campinas	5500	1963250.24
6	Agência Porto Alegre	Porto Alegre	4474	1605153.92
5	Agência Osasco	Osasco	4697	1427088.95
8	Agência Rio de Janeiro	Rio de Janeiro	3779	1159050.46
2	Agência Florianópolis	Florianópolis	2133	634934.63
3	Agência Jardins	São Paulo	2109	479622.44
7	Agência Recife	Recife	374	112926.01

```
In [ ]:
```

```
In [12]: # 3. Funcionários que Mais Realizam Vendas
```

```

vendas_por_funcionario = df_propostas.groupby('cod_colaborador').agg(
    propostas_realizadas=('cod_proposta', 'count'),
    propostas_aprovadas=('aprovada', 'sum')
).reset_index()
vendas_por_funcionario['taxa_aprovacao'] = (vendas_por_funcionario['propostas_aprovadas'] / vendas_por_funcionario['propostas_realizadas'])

# Mesclar com os nomes dos colaboradores e suas respectivas agências
df_vendas_funcionarios = vendas_por_funcionario.merge(df_colaboradores[['cod_colaborador', 'primeiro_nome', 'ultimo_nome']], on='cod_colaborador',
df_vendas_funcionarios = df_vendas_funcionarios.merge(df_colab_agencia[['cod_colaborador', 'cod_agencia']]), on='cod_colaborador',
df_vendas_funcionarios = df_vendas_funcionarios.merge(df_agencias[['cod_agencia', 'nome']]), on='cod_agencia', how='left')

# Selecionar apenas as colunas necessárias
df_vendas_funcionarios = df_vendas_funcionarios[['primeiro_nome', 'ultimo_nome', 'nome', 'taxa_aprovacao']]

print("Top 10 Funcionários com Maior Taxa de Aprovação:")
print(df_vendas_funcionarios.sort_values(by='taxa_aprovacao', ascending=False).head(10))

```

Top 10 Funcionários com Maior Taxa de Aprovação:

	primeiro_nome	ultimo_nome	nome	taxa_aprovacao
20	Rafaela	Correia	Agência Porto Alegre	47.058824
64	Sabrina	Silva	Agência Tatuapé	45.833333
86	Luna	Teixeira	Agência Rio de Janeiro	45.000000
9	Vitor Hugo	Novais	Agência Porto Alegre	44.444444
22	Elisa	Fogaça	Agência Matriz	43.478261
81	Mirella	Sales	Agência Recife	41.666667
19	Alice	Aragão	Agência Digital	41.176471
13	Paulo	Dias	Agência Matriz	41.176471
23	Victoria	Vieira	Agencia Matriz	40.909091
21	Davi Luiz	Sales	Agência Tatuapé	38.888889

In []:

In []: *###podemos ir avante com as primeiras analises - verificar quais os clientes mais ativos, agencias que mais fecham negócios
###qual tipo de emprestimo mais realizado, ou seja, a media dos valores, para segmentar ofertas*

In []: