# Aula 16 - Mineração de dados Parte 2: Pandas

---

Um dos pacotes mais conhecidos, e utilizados, do Python é o **Pandas**. Com o Pandas, é possível extrair dados de diferentes tipos de arquivos para análise e montagem de dashboards, o que o torna a ferramenta perfeita para ciência de dados e inteligência artificial. É considerado um tópico praticamente esssencial para quem quer aprender Python.

O Pandas é um pacote externo. Portanto, é necessário a instalação no seu projeto usando o gerenciador **pip**, através do comando `pip install pandas`.

## Importação da base de dados

---

O ideal ao fazer um trabalho que envolva análise de dados é trabalhar com os **notebooks**, ensinados na aula passada. Portanto, desta vez, iremos criar dentro da pasta do nosso projeto um arquivo com a extensão **.ipynb**. Não se esqueça de criar um **ambiente virtual *.venv*** e instalar o pacote pandas ( `pip install pandas` ) logo em seguida.

Para conseguirmos trabalhar com extração de dados, precisaremos de um arquivo externo. Iremos pegar para o nosso exemplo uma planilha com a extensão **.csv**. As planilhas com essa extensão estão entre as bases de dados mais utilizadas para análise de dados. Esses arquivos costumam ser gerados pelo Excel, ou exportando base de dados de Sistemas de cadastro, ou mesmo sendo gerados pos SGBD. O arquivo que iremos extrair os dados se chama **cancelamentos.csv**, e deverá estar dentro da pasta do seu projeto.

Vamos começar improtando a biblioteca Pandas, e atribuindo um alias, e logo em seguida iremos armazenar os dados da planilha em uma variável e exibi-la na tela:

In [ ]:
```
# importação da biblioteca pandas
import pandas as pd

# lê os dados da tabela
tabela = pd.read_csv('cancelamentos.csv')

# mostra os dados da tabela na tela
display(tabela)
```

| | CustomerID | idade | sexo | tempo_como_cliente | frequencia_uso | ligacoes_callcen |
|---|---|---|---|---|---|---|
| **0** | 2.0 | 30.0 | Female | 39.0 | 14.0 | |
| **1** | 3.0 | 65.0 | Female | 49.0 | 1.0 | 1 |
| **2** | 4.0 | 55.0 | Female | 14.0 | 4.0 | |
| **3** | 5.0 | 58.0 | Male | 38.0 | 21.0 | |
| **4** | 6.0 | 23.0 | Male | 32.0 | 20.0 | |
| **...** | ... | ... | ... | ... | ... | |
| **881661** | 449995.0 | 42.0 | Male | 54.0 | 15.0 | |
| **881662** | 449996.0 | 25.0 | Female | 8.0 | 13.0 | |
| **881663** | 449997.0 | 26.0 | Male | 35.0 | 27.0 | |
| **881664** | 449998.0 | 28.0 | Male | 55.0 | 14.0 | |
| **881665** | 449999.0 | 31.0 | Male | 48.0 | 20.0 | |

881666 rows × 12 columns

O comando `display()` mostra os dados em forma de tabela, como se fosse em uma planilha.

### Mas espera!!! Tem mais...

Vamos agora fazer uma espécie de filtro. Vamos refazer a consulta, mas retirando a coluna do ID, pois não precisaremos dela para a nossa consulta.

**Obs:** na próxima célula, conforme falado na aula anterior, iremos executar apenas os comandos que irão eliminar a coluna de ID da consulta e o comando do `display()` novamente para reexibir o novo resultado. Não precisaremos importar novamente o pandas, nem ler novameente o arquivo, basta apenas que executemos novamente a célula de código anterior:

In [ ]:
```
tabela = tabela.drop('CustomerID', axis=1)
display(tabela)
```

| | idade | sexo | tempo_como_cliente | frequencia_uso | ligacoes_callcenter | dias_atra |
|---|---|---|---|---|---|---|
| **0** | 30.0 | Female | 39.0 | 14.0 | 5.0 | 18 |
| **1** | 65.0 | Female | 49.0 | 1.0 | 10.0 | 8 |
| **2** | 55.0 | Female | 14.0 | 4.0 | 6.0 | 18 |
| **3** | 58.0 | Male | 38.0 | 21.0 | 7.0 | 7 |
| **4** | 23.0 | Male | 32.0 | 20.0 | 5.0 | 8 |
| **...** | ... | ... | ... | ... | ... | |
| **881661** | 42.0 | Male | 54.0 | 15.0 | 1.0 | 3 |
| **881662** | 25.0 | Female | 8.0 | 13.0 | 1.0 | 20 |
| **881663** | 26.0 | Male | 35.0 | 27.0 | 1.0 | 5 |
| **881664** | 28.0 | Male | 55.0 | 14.0 | 2.0 | 0 |
| **881665** | 31.0 | Male | 48.0 | 20.0 | 1.0 | 14 |

881666 rows × 11 columns

# Tratamento de dados

**Não é feitiçaria, é tecnologia...**

Graças ao Python, agora temos uma poderosa ferramenta para análise de dados muito mais perfeita que muita ferramenta famosa por aí. É por isso que Ciência de Dados paga tanto (

), rsrs....

Mas não terminamos por aqui. Repare na análise de cima que temos linhas em branco, sem dados. Eles estão aí só para nos atrapalhar, e é por isso que iremos retirá-los da nossa próxima análise. Primeiro, vamos identificar quantas linhas vazias temos na planilha:

In [ ]:
```
display(tabela.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 881666 entries, 0 to 881665
Data columns (total 11 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   idade                   881664 non-null  float64
 1   sexo                    881664 non-null  object 
 2   tempo_como_cliente      881663 non-null  float64
 3   frequencia_uso          881663 non-null  float64
 4   ligacoes_callcenter     881664 non-null  float64
 5   dias_atraso             881664 non-null  float64
 6   assinatura              881661 non-null  object 
 7   duracao_contrato        881663 non-null  object 
 8   total_gasto             881664 non-null  float64
 9   meses_ultima_interacao  881664 non-null  float64
 10  cancelou                881664 non-null  float64
dtypes: float64(8), object(3)
memory usage: 74.0+ MB
None
```

O resultado acima mostra a quantidade de linhas de cada campo, contando inclusive com as linhas vazias. Repare que os números de linhas é diferente em algumas colunas. Na próxima célula de código, iremos resolver isso, eliminando as linhas sem dados:

In [ ]:
```
tabela = tabela.dropna()
display(tabela.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 881659 entries, 0 to 881665
Data columns (total 11 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   idade                   881659 non-null  float64
 1   sexo                    881659 non-null  object 
 2   tempo_como_cliente      881659 non-null  float64
 3   frequencia_uso          881659 non-null  float64
 4   ligacoes_callcenter     881659 non-null  float64
 5   dias_atraso             881659 non-null  float64
 6   assinatura              881659 non-null  object 
 7   duracao_contrato        881659 non-null  object 
 8   total_gasto             881659 non-null  float64
 9   meses_ultima_interacao  881659 non-null  float64
 10  cancelou                881659 non-null  float64
dtypes: float64(8), object(3)
memory usage: 80.7+ MB
None
```

Voilá. Veja agora que o número de linhas é igual para todas as colunas. Show!!! Agora vamos para o mais importante...

# Análise de dados

---

Agora que os dados já foram tratados, podemos iniciar a **Análise de dados** propriamente dita. Que tal começarmos analisar a **taxa de cancelamento da empresa**,

supondo que o nosso objetivo seja diminuir essa taxa. Para isso, vamos ter que verificar qual é essa taxa e de onde ela vem:

In [ ]:
```
display(tabela['cancelou'].value_counts())
```

```
cancelou
1.0    499993
0.0    381666
Name: count, dtype: int64
```

Pronto! Temos um número. A linha do **1.0** representa quem cancelou, enquanto a linha do **0.0** representa quem não cancelou. Mas esse número representa qual porcentagem do total? Vamos analisar:

In [ ]:
```
display(tabela['cancelou'].value_counts(normalize=True).map('{:.2%}'.format))
```

```
cancelou
1.0    56.71%
0.0    43.29%
Name: proportion, dtype: object
```

O `.map()` atribui à lista uma operação, que nesse caso é a transformação dos valores em **porcentagem**.

Veja que, de acordo com a nossa análsie, mais da metade dos clientes cancelou o serviço. O próximo passo é analisar e tentar entender de onde vem uma taxa tão alta de cancelamento.

**IMPORTANTE: Na parte de análise de dados, não tem uma informação correta para ser analisada logo de cara. Esse é um processo que vai tomar tempo, pois você de fato precisa analisar os dados e entender o que está acontecendo na sua base de dados. Então pode ser que demore mais em alguns casos para encontrar o que procura antes de propor sua solução.**

Dito isso, vamos analisar por partes. Começaremos pela duração dos contratos. Existem 3 tipos de contratos na base de dados: **Anual**, **Trimestral** e **Mensal**. Vejamos a duração de cada um deles:

In [ ]:
```
display(tabela['duracao_contrato'].value_counts())
display(tabela['duracao_contrato'].value_counts(normalize=True).map('{:.2%}'.for
```

```
duracao_contrato
Annual       354395
Quarterly    353059
Monthly      174205
Name: count, dtype: int64
duracao_contrato
Annual       40.20%
Quarterly    40.04%
Monthly      19.76%
Name: proportion, dtype: object
```

Observe algo interessante: os contratos anuais e trimestrais tem uma divisão quase igual, mas há uma discrepância muito maior na duração dos contratos mensais.

## Algo de errado não está certo aí...

Vamos agrupar as informações da colina **duracao_contrato** e depois fazer a média das informações. Para isso, iremos usar a função `groupby()` e repassar o nome da coluna nela:

In [ ]:
```
display(tabela.groupby('duracao_contrato').mean(numeric_only=True))
```

| duracao_contrato | idade | tempo_como_cliente | frequencia_uso | ligacoes_callcenter | dia |
|---|---|---|---|---|---|
| **Annual** | 38.842165 | 31.446186 | 15.880213 | 3.263401 | 12 |
| **Monthly** | 41.552407 | 30.538555 | 15.499274 | 4.985649 | 15 |
| **Quarterly** | 38.830938 | 31.419916 | 15.886662 | 3.265245 | 12 |

Aqui temos uma informação alarmante: veja que a taxa de cancelamento dos contratos anuais e trimestrais é da ordem de 46% aproximadamente, mas o os contratos anuais tiveram **EXATOS 100% DE CANCELAMENTO!!!**

## O que pode ser feito?

Vamos remover as informações desse contrato e continuar analisando. Aqui vale lembrar que nem sempre que encontrar algo que seja ruim na sua análise de dados você retira e para por ali. A ideia é ir analisando até que chegue em um valor aceitável dentro do seu projeto. Então é importante definir esse "valor aceitável" ou seu objetivo para não ficar trabalhando sem ter um ponto de parada.

In [ ]:
```
tabela = tabela[tabela['duracao_contrato'] != 'Monthly']
display(tabela)
display(tabela['cancelou'].value_counts())
display(tabela['cancelou'].value_counts(normalize=True).map('{:.2%}'.format))
```

| | idade | sexo | tempo_como_cliente | frequencia_uso | ligacoes_callcenter | dias_atra |
|---|---|---|---|---|---|---|
| **0** | 30.0 | Female | 39.0 | 14.0 | 5.0 | 18 |
| **2** | 55.0 | Female | 14.0 | 4.0 | 6.0 | 18 |
| **5** | 51.0 | Male | 33.0 | 25.0 | 9.0 | 26 |
| **6** | 58.0 | Female | 49.0 | 12.0 | 3.0 | 16 |
| **7** | 55.0 | Female | 37.0 | 8.0 | 4.0 | 15 |
| **...** | ... | ... | ... | ... | ... | |
| **881661** | 42.0 | Male | 54.0 | 15.0 | 1.0 | 3 |
| **881662** | 25.0 | Female | 8.0 | 13.0 | 1.0 | 20 |
| **881663** | 26.0 | Male | 35.0 | 27.0 | 1.0 | 5 |
| **881664** | 28.0 | Male | 55.0 | 14.0 | 2.0 | 0 |
| **881665** | 31.0 | Male | 48.0 | 20.0 | 1.0 | 14 |

707454 rows × 11 columns

```
cancelou
0.0    381666
1.0    325788
Name: count, dtype: int64
cancelou
0.0    53.95%
1.0    46.05%
Name: proportion, dtype: object
```

A proporção de cancelamentos já caiu para **46,05%**, mas esse número ainda é alto, então vamos continuar para abaixar esse valor para algo longe dos 50%. Vamos agora fazer uma análise das assinaturas para verificar se podemos tirar alguma conclusão para melhorar o índice de cancelamentos. Primeiro vamos fazer a contagem dos valores na coluna de assinaturas para saber quantas assinaturas temos em cada um dos planos. Em seguida vamos agrupar as informações por assinatura e obter a média das linhas para cada uma das colunas.

In [ ]:
```
display(tabela['assinatura'].value_counts(normalize=True).map('{:.2%}'.format))
display(tabela.groupby('assinatura').mean(numeric_only=True))
```

```
assinatura
Standard    33.96%
Premium     33.81%
Basic       32.22%
Name: proportion, dtype: object
```

| assinatura | idade | tempo_como_cliente | frequencia_uso | ligacoes_callcenter | dias_atrasc |
|---|---|---|---|---|---|
| **Basic** | 38.904813 | 32.316031 | 15.876921 | 3.310021 | 12.507054 |
| **Premium** | 38.817814 | 30.977869 | 15.889673 | 3.235886 | 12.433427 |
| **Standard** | 38.790478 | 31.048621 | 15.883393 | 3.249275 | 12.450690 |

Infelizmente, ainda não conseguimos resolver o nosso problema, pois a taxa de cancelamentos continua na proporção de 1 para 3 em cada assinatura. É hora de irmos mais a fundo.

# Gráficos

Uma das melhores formas de se analisar dados é através de gráficos, pois fica muito mais fácil visualizar os dados e obter as informações necessárias.

O Python é extremamente rico em bibliotecas que geram gráficos, e disso definitivamente não temos do que reclamar. Para o nosso caso específico, iremos instalar e importar a biblioteca **plotly.express**, usando, é claro, o comando `pip install plotly`.

**OBS:** Caso tenha problemas na visualização dos gráficos depois de instalar a biblioteca, volte ao terminal e escreva `pip install nbformat`. Se mesmo após isso não funcionar, reinicie o VSCode e rode todos os códigos do notebook novamente.

Vamos usar o plotly para criar um histograma com cada uma das colunas, assim podemos analisar cada uma das informações e verificar como elas se comportam em relação aos cancelamentos da empresa.

In [ ]:
```
import plotly.express as px

for coluna in tabela.columns:
    grafico = px.histogram(tabela, x=coluna, color="cancelou", width=600)
    grafico.show()
```

Observe que o código gerou 11 gráficos: um para cada coluna. É claro que teremos que analisar cada um deles, mas vamos dar um *spoiler*: observe os gráficos **dias_atraso** e **ligacoes_callcenter**:

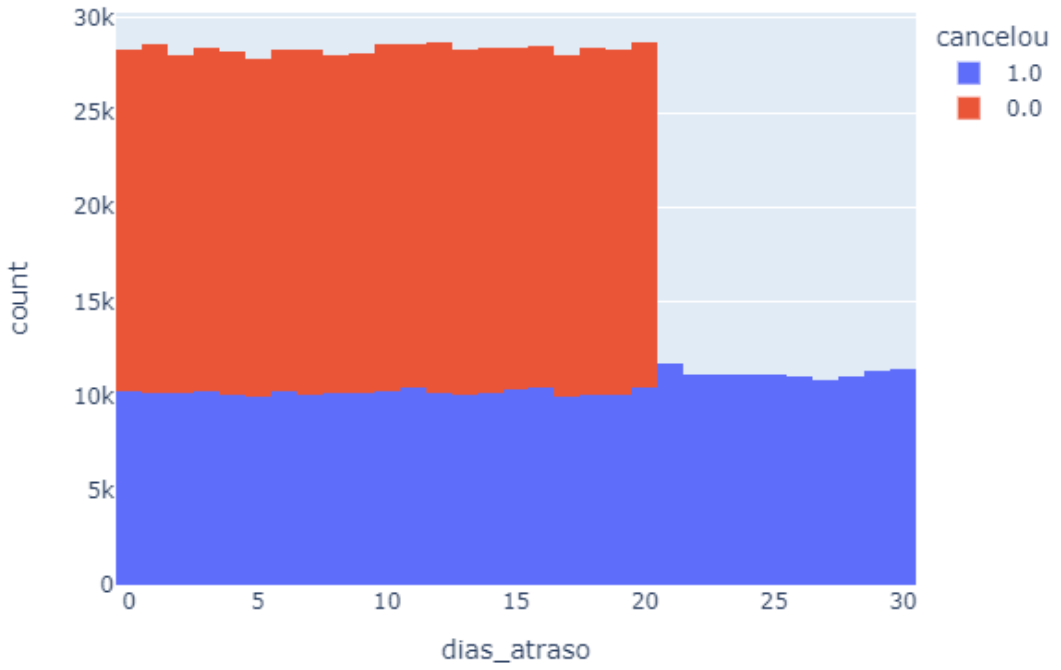


Gráfico dos Dias de Atraso

O gráfico dos **Dias de Atraso** indica que clientes com mais de **20 dias de atraso** cancelam suas assinaturas.

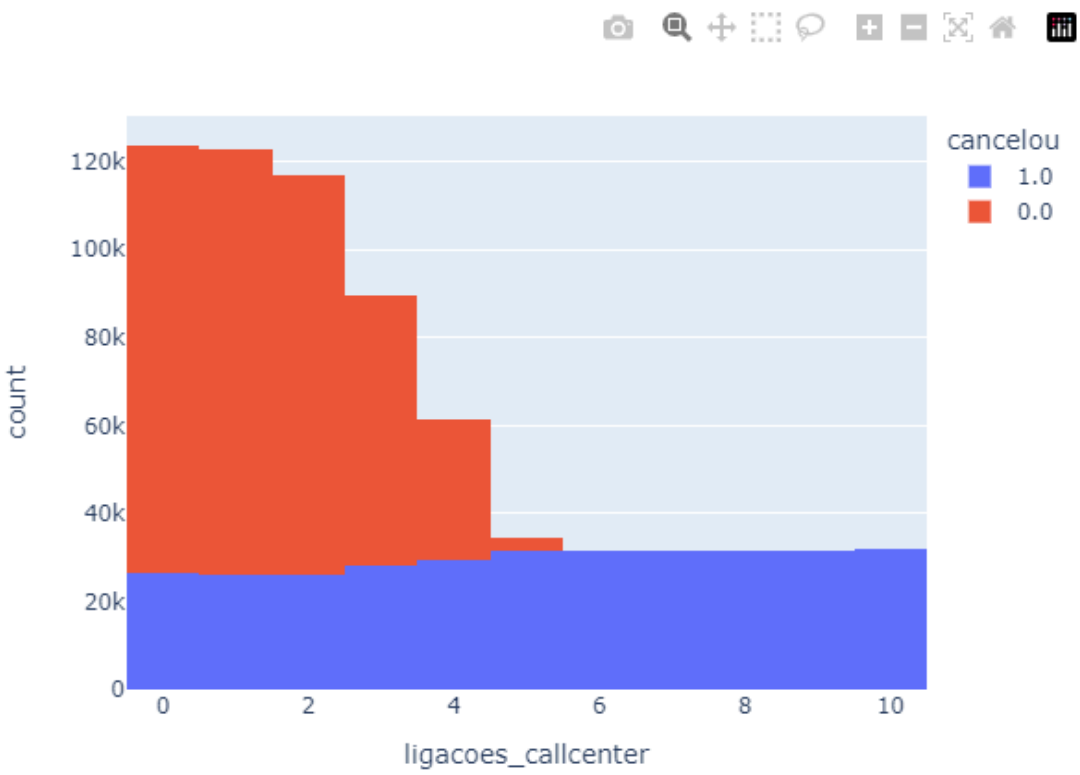


Gráfico das Ligações ao Call Center

Já no gráfico de **Ligações ao Call Center**, os clientes que realizaram mais de **5 ligações ao call center** cancelaram suas assinaturas.

# Resultado

Analisando apenas os dois gráficos indicados, nós podemos manter na tabela as ligações ao call center que são menores do que 5. E podemos manter também as informações onde os dias de atraso são menores do que 0. Com isso podemos visualizar a base de dados e fazer o cálculo novamente para verificar o percentual de cancelamento. Veja que só removendo as informações dessas duas colunas passamos de um percentual de quase 50% de cancelamento para 18,4%. Conseguiu perceber que tivemos que fazer várias etapas até chegar em um valor "aceitável"? Não é apenas fazer um único tratamento que vamos ter esse resultado.

In [ ]:
```
tabela = tabela[tabela['ligacoes_callcenter'] < 5]
tabela = tabela[tabela['dias_atraso'] <= 20]
display(tabela)
display(tabela['cancelou'].value_counts())
display(tabela['cancelou'].value_counts(normalize=True).map('{:.2%}'.format))
```

| | idade | sexo | tempo_como_cliente | frequencia_uso | ligacoes_callcenter | dias_atra |
|---|---|---|---|---|---|---|
| **6** | 58.0 | Female | 49.0 | 12.0 | 3.0 | 16 |
| **7** | 55.0 | Female | 37.0 | 8.0 | 4.0 | 15 |
| **9** | 64.0 | Female | 3.0 | 25.0 | 2.0 | 1 |
| **13** | 48.0 | Female | 35.0 | 25.0 | 1.0 | 13 |
| **19** | 42.0 | Male | 15.0 | 16.0 | 2.0 | 14 |
| **...** | ... | ... | ... | ... | ... | |
| **881661** | 42.0 | Male | 54.0 | 15.0 | 1.0 | 3 |
| **881662** | 25.0 | Female | 8.0 | 13.0 | 1.0 | 20 |
| **881663** | 26.0 | Male | 35.0 | 27.0 | 1.0 | 5 |
| **881664** | 28.0 | Male | 55.0 | 14.0 | 2.0 | 0 |
| **881665** | 31.0 | Male | 48.0 | 20.0 | 1.0 | 14 |

464479 rows × 11 columns

```
cancelou
0.0    379032
1.0     85447
Name: count, dtype: int64
cancelou
0.0    81.60%
1.0    18.40%
Name: proportion, dtype: object
```