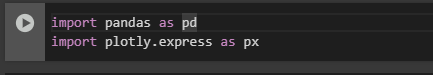
import pandas as pd

import plotly.express as px



import pandas as pd:

* Pandas - é uma biblioteca do Python usada para manipulação e análise de dados. Ela fornece estruturas de dados como DataFrame e Series, que são eficientes para lidar com grandes volumes de dados, realizar operações e análise estatísticas, entre outras tarefas.
* A palavra pd - é um apelido (alias) que estamos dando à biblioteca pandas. Esse alias torna o código mais compacto e fácil de escrever. Em vez de escrever pandas.DataFrame, podemos simplesmente escrever pd.DataFrame.

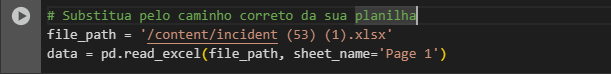
import plotly.express as px:

* Plotly.express - é uma biblioteca para visualização de dados interativa, baseada no Plotly, que facilita a criação de gráficos bonitos e interativos com apenas algumas linhas de código.
* Assim como com o pandas, a palavra px é um apelido (alias) que estamos utilizando para simplificar o código. Em vez de usar plotly.express.scatter, por exemplo, podemos usar px.scatter.

# Substitua pelo caminho correto da sua planilha

file\_path = '/content/incident (53) (1).xlsx'

data = pd.read\_excel(file\_path, sheet\_name='Page 1')



file\_path = '/content/incident (53) (1).xlsx':

* Essa linha está atribuindo o caminho do arquivo Excel à variável file\_path. O caminho do arquivo '/content/incident (53) (1).xlsx' parece indicar que o arquivo está armazenado em um ambiente onde o diretório /content/ é utilizado, como em uma plataforma de notebook na nuvem (por exemplo, Google Colab).
* O arquivo é um documento Excel (com a extensão .xlsx) e seu nome é incident (53) (1).xlsx.

data = pd.read\_excel(file\_path, sheet\_name='Page 1'):

* A função pd.read\_excel() - é usada para ler arquivos Excel e carregá-los como um DataFrame do Pandas.
* File\_path - Esse é o caminho do arquivo Excel que estamos carregando.
* sheet\_name='Page 1' - Aqui, estamos especificando o nome da aba que queremos carregar dentro do arquivo Excel. Nesse caso, estamos buscando a aba chamada Page 1. Se não especificarmos essa parte, o Pandas carregaria a primeira aba por padrão.
* O resultado é atribuído à variável data, que será um DataFrame contendo todos os dados da aba Page 1 do arquivo Excel.

# Carregar a planilha

data = pd.read\_excel(file\_path, sheet\_name='Page 1')

pd.read\_excel():

* A função read\_excel() do Pandas é utilizada para ler arquivos Excel e carregar seus dados em um DataFrame.
* Ela pode lidar com arquivos em formatos como .xlsx ou .xls.

file\_path:

* O argumento file\_path especifica o local do arquivo Excel a ser carregado. Este caminho foi definido previamente como '/content/incident (53) (1).xlsx'.

sheet\_name='Page 1':

* O parâmetro sheet\_name define qual aba (ou "planilha") do arquivo Excel será carregada.
* No seu caso, estamos carregando a aba chamada Page 1.
* Se você omitir este parâmetro, a função irá carregar a primeira aba por padrão.

data:

* O resultado da leitura do arquivo é armazenado na variável data, que será um DataFrame.
* Um DataFrame é uma estrutura de dados bidimensional (como uma tabela), onde as colunas podem ter tipos diferentes de dados, como números, texto ou datas.

fig = px.bar(calls\_per\_company,

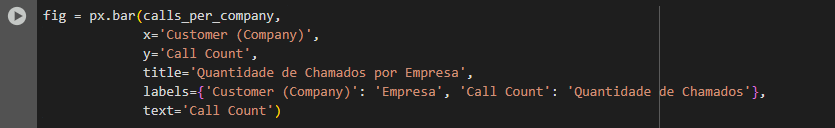
x='Customer (Company)',

y='Call Count',

title='Quantidade de Chamados por Empresa',

labels={'Customer (Company)': 'Empresa', 'Call Count': 'Quantidade de Chamados'},

text='Call Count')



Cria um gráfico de barras interativo usando a biblioteca plotly.express. Vamos detalhar cada parte:

1. px.bar():
   * A função bar() de plotly.express cria um gráfico de barras. Ela recebe um DataFrame como entrada (no caso, o calls\_per\_company) e plota as colunas especificadas nos eixos x e y.
2. Calls\_per\_company - Este é o DataFrame que você criou anteriormente, que contém o número de chamados por empresa.
3. x='Customer (Company)'- A coluna 'Customer (Company)' será usada no eixo x do gráfico, representando as empresas.
4. y='Call Count' - A coluna 'Call Count' será usada no eixo y, mostrando a quantidade de chamados para cada empresa.
5. title='Quantidade de Chamados por Empresa' - O título do gráfico será 'Quantidade de Chamados por Empresa', explicando claramente o que o gráfico está mostrando.
6. labels={'Customer (Company)': 'Empresa', 'Call Count': 'Quantidade de Chamados'} - Esse parâmetro é um dicionário que mapeia os nomes das colunas para rótulos mais amigáveis.
   * 'Customer (Company)' - será renomeado para 'Empresa' no gráfico.
   * 'Call Count' - será renomeado para 'Quantidade de Chamados'.
7. text='Call Count ‘ - O parâmetro text especifica que o valor de Call Count será mostrado no topo de cada barra do gráfico. Ou seja, a quantidade de chamados será exibida diretamente sobre cada barra, tornando o gráfico mais informativo.

### O que o gráfico faz:

O código irá gerar um gráfico de barras onde:

* O eixo x mostra as empresas (ou clientes).
* O eixo y mostra a quantidade de chamados para cada empresa.
* O título do gráfico será "Quantidade de Chamados por Empresa".
* As barras terão o valor da quantidade de chamados exibido acima de cada uma.

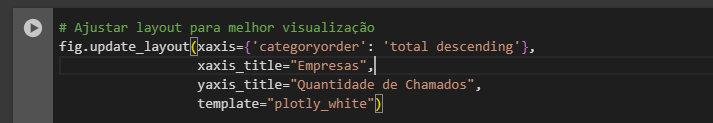
# Ajustar layout para melhor visualização

fig.update\_layout(xaxis={'categoryorder': 'total descending'},

xaxis\_title="Empresas",

yaxis\_title="Quantidade de Chamados",

template="plotly\_white")



Os ajustes no layout do gráfico para melhorar a visualização. Vamos entender cada parte:

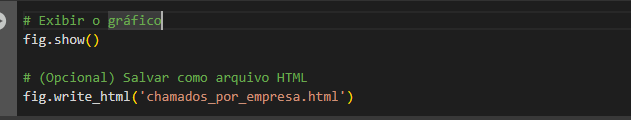
1. fig.update\_layout():
   * O método update\_layout() - é utilizado para modificar o layout do gráfico. O layout inclui características como título, rótulos de eixos, estilo e outros ajustes estéticos.
   * Aqui, fig é o gráfico gerado com px.bar(), que foi atribuído anteriormente à variável ig. No entanto, se você estava usando ig, o nome correto seria ig.update\_layout().
2. xaxis={'categoryorder': 'total descending'}:
   * Esse parâmetro organiza as categorias (no caso, as empresas) no eixo x de forma decrescente, com base no valor total de cada barra (ou seja, na quantidade de chamados).
   * A opção 'total descending' garante que as empresas com o maior número de chamados aparecerão primeiro, à esquerda, no gráfico.
3. xaxis\_title="Empresas":
   * Isso define o título do eixo x, que será exibido abaixo do eixo. No caso, ele será alterado para "Empresas".
4. yaxis\_title="Quantidade de Chamados":
   * Define o título do eixo y, que será exibido ao lado esquerdo do gráfico. Ele será alterado para "Quantidade de Chamados".
5. template="plotly\_white":
   * O parâmetro template altera o estilo do gráfico. Ao definir "plotly\_white", você está aplicando um estilo claro, com fundo branco e linhas mais suaves, o que melhora a legibilidade, especialmente em gráficos interativos.

# Exibir o gráfico

fig.show()

# (Opcional) Salvar como arquivo HTML

fig.write\_html('chamados\_por\_empresa.html')



O fig.show() exibe o gráfico gerado no ambiente atual, seja em um notebook Jupyter, em uma aplicação web ou em um ambiente local.

No caso de um notebook como o Jupyter ou Google Colab, o gráfico será renderizado diretamente na interface do notebook, permitindo interação (como passar o mouse sobre as barras para ver os valores) e visualização.

write\_html() - salva o gráfico interativo em um arquivo HTML. Esse arquivo pode ser aberto em qualquer navegador, mantendo a interatividade (por exemplo, você pode passar o mouse sobre as barras para ver os valores ou fazer zoom no gráfico).

O arquivo será salvo com o nome chamados\_por\_empresa.html no diretório atual onde o código está sendo executado.

Esse passo é útil caso você queira compartilhar o gráfico ou fazer a visualização em outro momento sem precisar gerar o gráfico novamente.

### Resumo:

* fig.show() exibe o gráfico na tela.
* fig.write\_html('chamados\_por\_empresa.html') salva o gráfico em um arquivo HTML, que pode ser visualizado em um navegador.