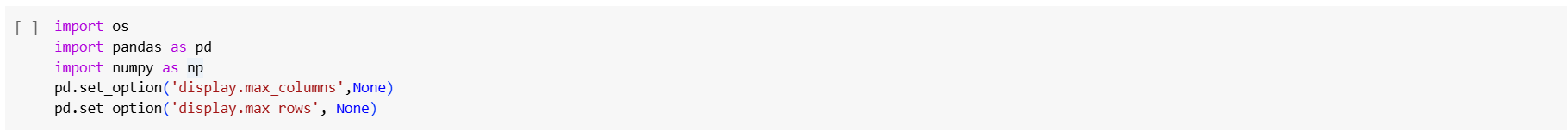
**TASK ORIENTED DOCUMENTATION**

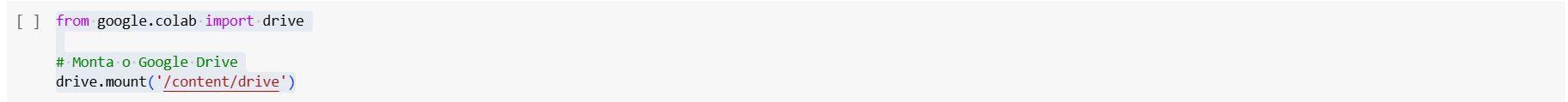
**Configuração do Ambiente para Análise de Dados:**



***import:*** Importa a biblioteca pandas para manipulação e análise de dados e importa a biblioteca numpy que é utilizada para cálculos numéricos.

***pd.set\_option:*** Configura todas as linhas e colunas, sem limite, para exibição no *DataFrame.*

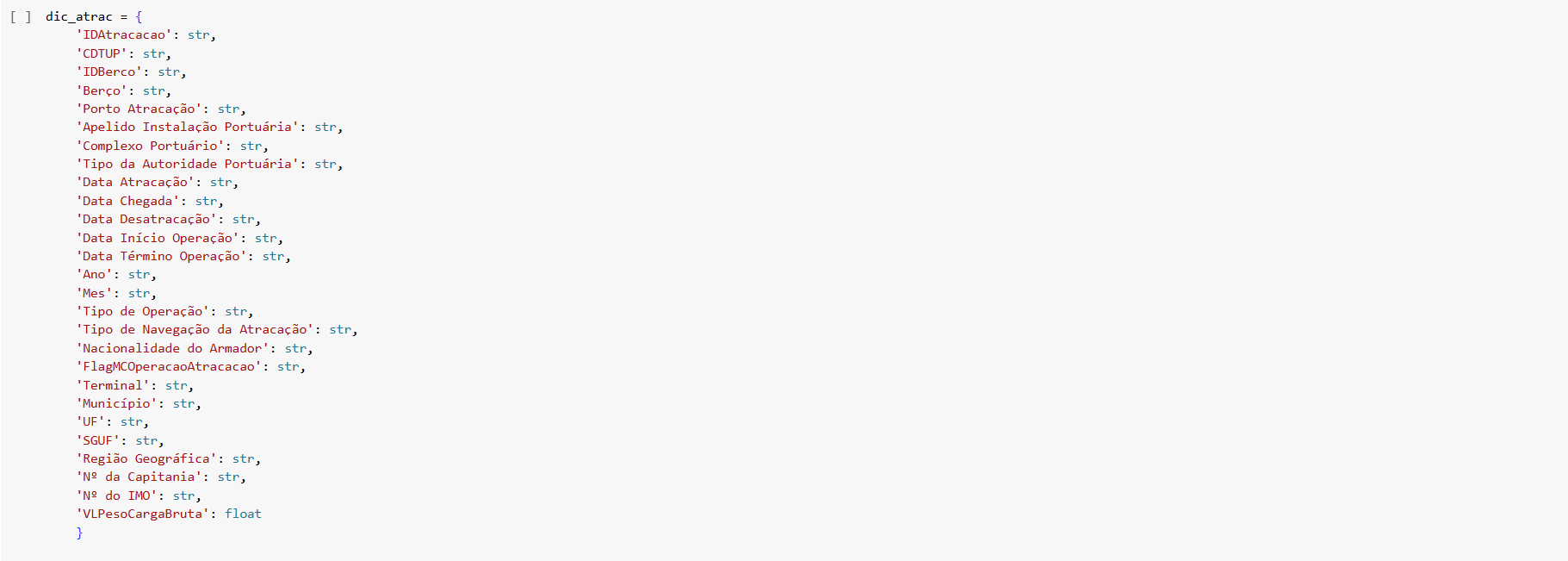
**Montagem do Google Drive no Google Colab:**



Realiza a montagem do Google Drive no ambiente do *Google Colab*, permitindo que você acesse arquivos armazenados na sua conta do Google Drive diretamente do notebook/PC.

A primeira linha importa a biblioteca drive do *Google Colab*, que fornece funcionalidades para interagir com o Google Drive. Já a segunda, monta o Google Drive no diretório especificado.

**Criação de um Dicionário de Tipos de Dados:**



***dic\_atrac:*** Especifica os tipos de dados para diferentes colunas que serão utilizadas em um DataFrame. Nesse caso, específico de atracação.

Todos os valores associados a essas chaves são *str*, indicando que todas as colunas devem ser interpretadas como **strings.** Isso porque em dados onde os valores não são necessariamente numéricos, que podem ser manipuladas como texto. Exceto pelo *VLPesoCargaBruta* que foi interpretada como *float* (para números com ponto decimal).

**Definição do Caminho do Arquivo no Google Drive:**



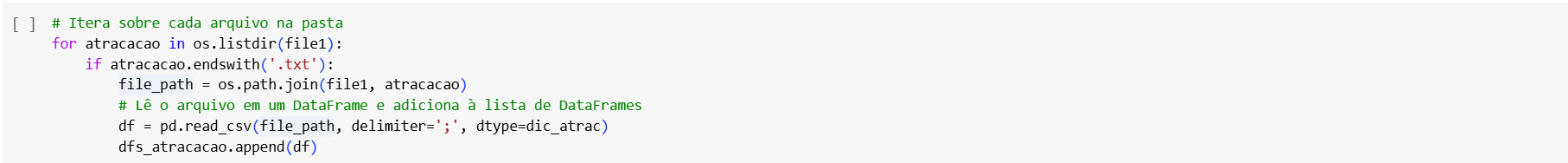
O código apresentado define uma variável chamada *file1*, que contém o caminho para um diretório específico no Google Drive.

**Inicialização de uma Lista para DataFrames:**



Cria uma variável e [ ] representa uma lista vazia em *python*.

**Leitura de Arquivos TXT e Armazenamento em DataFrames:**



***for atracacao in os.listdir(file1):*** Esta linha percorre todos os arquivos e diretórios presentes na pasta especificada por file1.

***if atracacao.endswith('.txt'):*** Verifica se o nome do arquivo (atracacao) termina com a extensão .txt, garantindo que apenas arquivos de texto sejam processados.

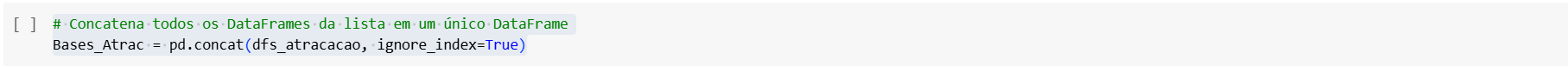
***file\_path = os.path.join(file1, atracacao):*** Cria um caminho completo para o arquivo, juntando o diretório com o nome do arquivo.

***delimiter=';'*** especifica que o delimitador dos dados no arquivo é o ponto e vírgula.

***dtype=dic\_atrac*** aplica o dicionário de tipos de dados que foi definido anteriormente, garantindo que cada coluna tenha o tipo apropriado.

***dfs\_atracacao.append(df):*** Adiciona o DataFrame (df) lido à lista dfs\_atracacao. Isso permite que você colete todos os DataFrames em um único objeto para análise futura.

**Concatenação de DataFrames em um Único DataFrame:**



Combina todos os DataFrames armazenados em uma lista em um único DataFrame.

**Exibição de Informações do DataFrame:**



É usado para obter um resumo conciso sobre o DataFrame *Bases\_Atrac.*

**Verificação de Duplicados na Coluna:**

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

Verifica se há valores duplicados na coluna IDAtracacao do DataFrame Bases\_Atrac e exibe os resultados.

O código *keep=False* significa que todas as ocorrências de duplicatas serão marcadas como *True.*

***else: print ("Não existem valores duplicados na coluna 'IDAtracacao'."):***Se não houver duplicatas, essa mensagem será impressa, informando que todos os valores na coluna são únicos.

**Definição do Caminho do Diretório para Carga de Dados:**



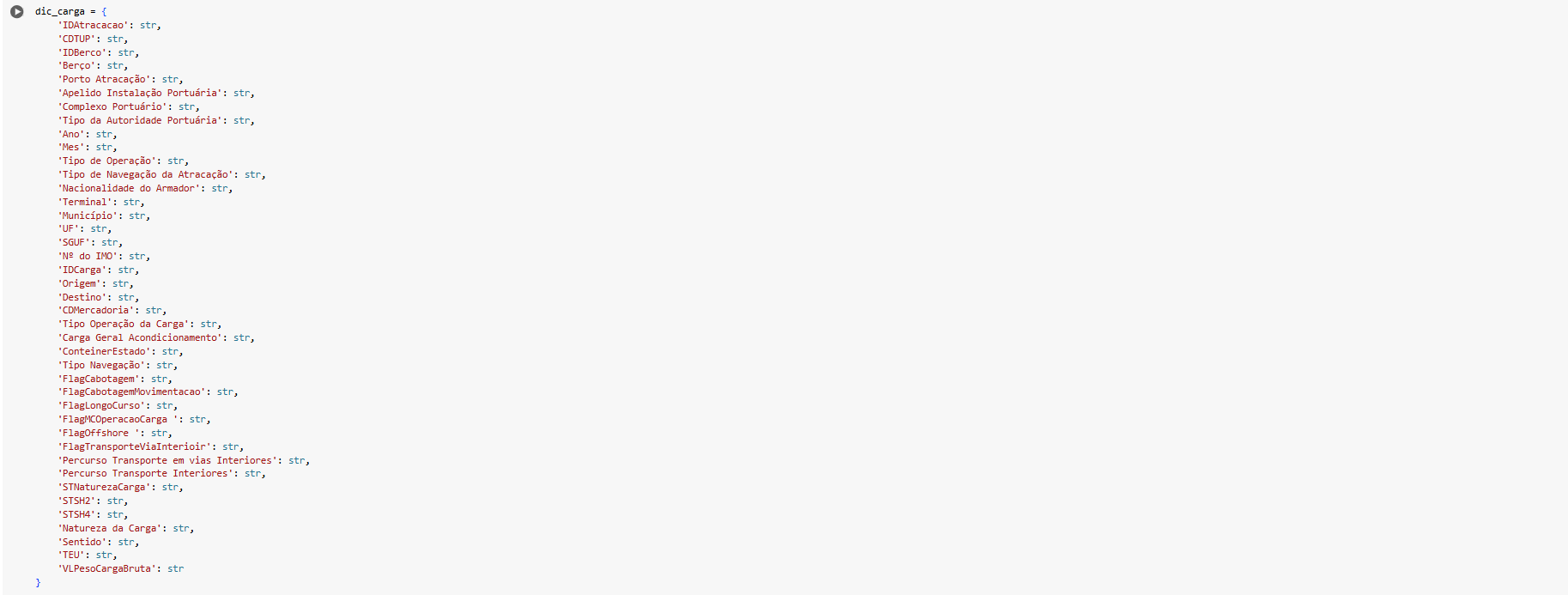
*file2:* É o nome da variável que armazena o caminho do diretório onde estão os arquivos relacionados à carga.

**Inicialização de uma Lista para DataFrames de Carga:**



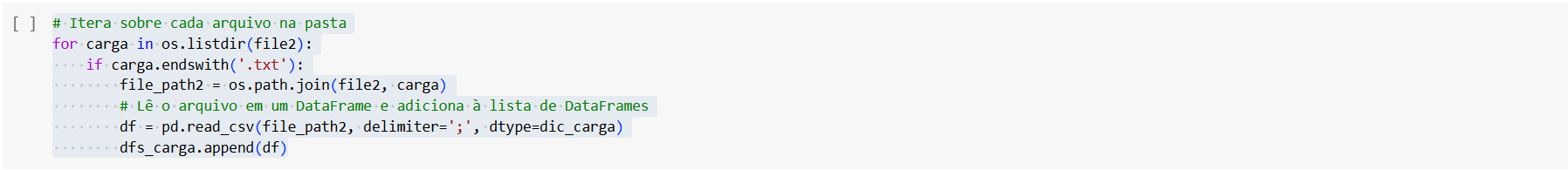
Cria uma nova variável e [ ] representa uma lista vazia em *python.*

**Criação de um Dicionário de Tipos de Dados para Carga:**



Especifica os tipos de dados para diferentes colunas que serão utilizadas em um *DataFrame*. Nesse caso, específico de carga.

**Leitura de Arquivos TXT e Armazenamento em DataFrames de Carga:**



***for carga in os.listdir(file2):*** Percorre todos os arquivos e diretórios na pasta especificada por file2.

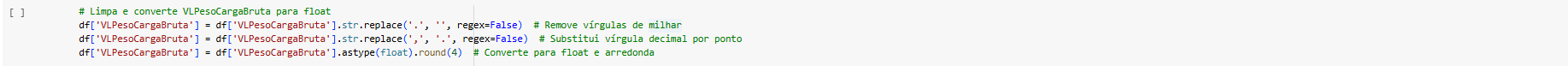
***if carga.endswith('.txt'):*** Verifica se o nome do arquivo (carga) termina com a extensão .txt, garantindo que apenas arquivos de texto sejam processados.

***file\_path2 = os.path.join(file2, carga):*** Cria um caminho completo para o arquivo, combinando o diretório (file2) com o nome do arquivo (carga).

***df = pd.read\_csv(file\_path2, delimiter=';', dtype=dic\_carga):***

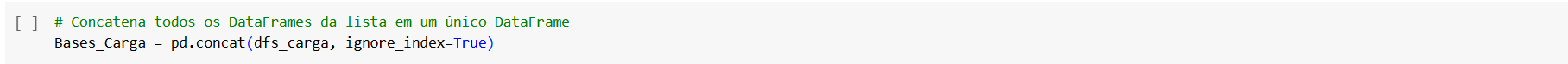
* Lê o arquivo de texto como um DataFrame usando a função *pd.read\_csv().*
* ***delimiter=';'*:** Especifica que o delimitador dos dados no arquivo é o ponto e vírgula.
* ***dtype=dic\_carga*:** Aplica o dicionário de tipos de dados definido anteriormente, garantindo que cada coluna tenha o tipo apropriado.
* ***dfs\_carga.append(df):***Adiciona o DataFrame (df) lido à lista dfs\_carga.

**Limpeza e Conversão do Campo de Peso Bruto de Carga para Float:**



Este trecho de código realiza a limpeza e conversão de dados do campo VLPesoCargaBruta em um DataFrame. Ele processa os valores de peso bruto da carga, inicialmente armazenados como strings formatadas com separadores de milhar e decimal, para convertê-los em valores numéricos de ponto flutuante (float).

**Concatenação de DataFrames de Carga em um Único DataFrame:**



Combina todos os DataFrames armazenados na lista *dfs\_carga* em um único DataFrame chamado *Bases\_Carga*.

**Exibição de Informações do DataFrame:**



É usado para obter um resumo conciso sobre o DataFrame *Bases\_Carga.*

**Filtrando Dados para Cargas Exclusivas:**



Cria uma condição que verifica se o valor da coluna STNaturezaCarga é igual a "Exclusivo". Para cada linha, se o valor nessa coluna for "Exclusivo", o resultado será *True*; caso contrário, será *False.*

**Visualizando a Primeira Linha do DataFrame Filtrado:**

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Essa operação é usada para visualizar rapidamente a primeira linha de *df\_carga\_exclusiva*. É útil para verificar se o *DataFrame* foi corretamente filtrado.

**Filtrando Cargas Embarcadas em um DataFrame Exclusivo:**



**df\_carga\_exclusiva['Sentido'] == 'Embarcados'**: Esta condição verifica se o valor da coluna Sentido é igual a "Embarcados". Para cada linha, se o valor nessa coluna for "Embarcados", o resultado será *True*, e a linha será selecionada; caso contrário, será ignorada.

**df\_carga\_exclusiva\_embarcados**: O resultado da filtragem é armazenado nesse novo DataFrame, que agora contém apenas as cargas que têm o "Sentido" como "Embarcados" e a natureza de carga "Exclusivo".

**Visualizando a Primeira Linha das Cargas Exclusivas Embarcadas:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Essa operação é usada para visualizar rapidamente a primeira linha de *df\_carga\_exclusiva\_embaracdos*. É útil para verificar se o *DataFrame* foi corretamente filtrado.

**Filtrando Cargas de Granel Sólido no DataFrame de Cargas Exclusivas Embarcadas:**



**df\_carga\_exclusiva\_embarcados['Natureza da Carga'] == 'Granel Sólido'**: Essa condição verifica se o valor da coluna Natureza da Carga é igual a *"Granel Sólido".* Para cada linha, se o valor nessa coluna for *"Granel Sólido"*, o resultado será *True*, e a linha será selecionada.

**Visualizando a Primeira Linha das Cargas de Granel Sólido Embarcadas e Exclusivas:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Essa operação é usada para visualizar rapidamente a primeira linha de *df\_carga\_granel\_solido*. É útil para verificar se o *DataFrame* foi corretamente filtrado.

**Exibindo Informações Gerais sobre o DataFrame de Cargas de Granel Sólido:**

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

É usado para obter um resumo conciso sobre o DataFrame *df\_carga\_granel\_solido.*

**info()**: Esse método do pandas exibe um resumo informativo sobre o DataFrame, incluindo:

* O número total de entradas no DataFrame.
* O número de colunas e seus respectivos nomes.
* O tipo de dado de cada coluna (inteiros, floats, strings, etc.).
* O número de valores não nulos em cada coluna.

**Verificação e Conversão de Valores para Float na Coluna VLPesoCargaBruta:**



Verifica e converte os valores da coluna VLPesoCargaBruta no DataFrame df\_carga\_granel\_solido para o tipo float. Ele assegura que todos os valores dessa coluna sejam numéricos, permitindo cálculos precisos e consistentes para análise de carga bruta sólida.

**Agrupamento de Cargas por ID de Atracação e Soma do Peso Bruto:**



**groupby(['IDAtracacao', 'CDMercadoria', 'STNaturezaCarga', 'Natureza da Carga', 'Sentido'])**: O método groupby() agrupa o DataFrame com base nas colunas especificadas.

**agg():** permite aplicar uma função de agregação para as colunas desejadas

**reset\_index():** transforma esses índices de volta em colunas, restaurando a estrutura do DataFrame.

**Exibindo Informações Gerais sobre o DataFrame Agrupado de Cargas:**



É usado para obter um resumo conciso sobre o DataFrame *df\_carga\_agrupada.*

 **df\_carga\_agrupada**: Se refere ao DataFrame resultante da agregação, onde as cargas foram agrupadas por "IDAtracacao", "CDMercadoria", "STNaturezaCarga", "Natureza da Carga", e "Sentido", com a soma dos valores da coluna "VLPesoCargaBruta".

 **info()**: exibe um resumo do DataFrame, incluindo:

* O número total de entradas e colunas.
* O nome de cada coluna no DataFrame.
* O tipo de dado de cada coluna (como int64, float64, object para strings).
* Quantidade de valores não nulos em cada coluna (o que ajuda a identificar colunas com dados ausentes).

**Identificando a Mercadoria com o Maior Peso Bruto de Carga:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente com confiança média

d**f\_carga\_agrupada['VLPesoCargaBruta'].idxmax()**: Retorna o índice da linha onde a coluna *VLPesoCargaBruta* atinge o seu maior valor. Ou seja, ele encontra a linha que contém a mercadoria com o maior peso bruto de carga.

**df\_carga\_agrupada.loc[...]**: É usado para selecionar essa linha específica do DataFrame, com base no índice retornado por *idxmax()*. Dessa forma, obtemos todas as informações da linha que contém a mercadoria com o maior valor de *VLPesoCargaBruta.*

**mercadoria\_maior\_peso**: O resultado é armazenado na variável *mercadoria\_maior\_peso*, que contém todas as informações da mercadoria.

**print(...)**: Exibe as informações da mercadoria identificada, incluindo o nome das colunas e seus respectivos valores.

**Filtrando Registros da Mercadoria com Código 2601:**



Verifica cada linha do DataFrame para identificar se o valor na coluna CDMercadoria é igual a 2601. Para cada linha, se a condição for verdadeira, ela será incluída no resultado.

**loc[]:** É utilizado para selecionar as linhas do DataFrame que atendem à condição especificada.

O resultado da filtragem é armazenado na variável *df\_mercadoria\_2601*, que agora contém apenas as informações sobre a mercadoria com código 2601.

**Exibindo as Primeiras Linhas do DataFrame Filtrado para a Mercadoria 2601:**



Essa operação é usada para visualizar rapidamente a primeira linha de *df\_mercadoria\_2601*. É útil para verificar se o *DataFrame* foi corretamente filtrado.

**Unindo DataFrames com Base no ID de Atracação:**



**pd.merge(...)**: Utilizada para combinar dois DataFrames, neste caso, *Bases\_Atrac* e *df\_mercadoria\_2601.*

**on='IDAtracacao'**: Indica que a junção entre os DataFrames deve ser realizada com base na coluna IDAtracacao, que deve estar presente em ambos os DataFrames.

**how='inner'**: Especifica o tipo de junção a ser realizada, significa que apenas as linhas que têm correspondência em ambos os DataFrames (com base na coluna IDAtracacao) serão incluídas no resultado.

**df\_merge**: O resultado da junção é armazenado neste novo DataFrame, que agora contém apenas as informações de atracações e cargas da mercadoria 2601 onde há correspondência.

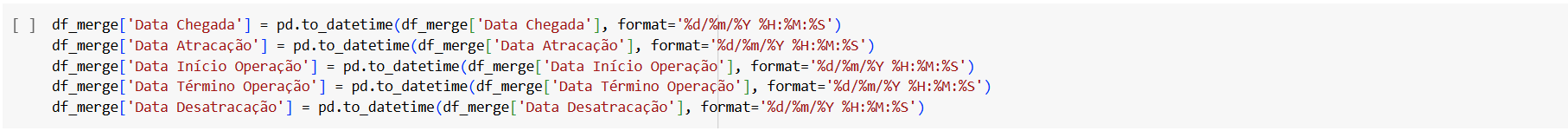
**Exibindo as Primeiras Linhas do DataFrame Resultante da Junção:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Essa operação é usada para visualizar rapidamente as cinco primeiras linha de *df\_merge*. É útil para verificar se o *DataFrame* foi corretamente filtrado.

**Convertendo Colunas de Data e Hora para o Formato Datetime:**

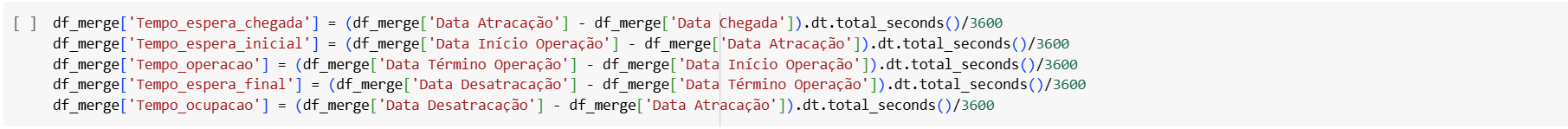


**d.to\_datetime(...)**: É utilizada para converter strings que representam datas em objetos do tipo datetime.

**format='%d/%m/%Y %H:%M:%S':** Especifica o formato da string original.

O resultado da conversão é atribuído novamente à mesma coluna no DataFrame *df\_merge.*

**Calculando Tempos de Espera e Operação em Horas:**



**Tempo\_espera\_chegada**: Calcula a diferença entre a *Data Atracação* e a *Data Chegada*, representando o tempo de espera até a atracação.

**Tempo\_espera\_inicial**: Calcula a diferença entre a Data Início Operação e a Data Atracação, representando o tempo de espera até o início da operação.

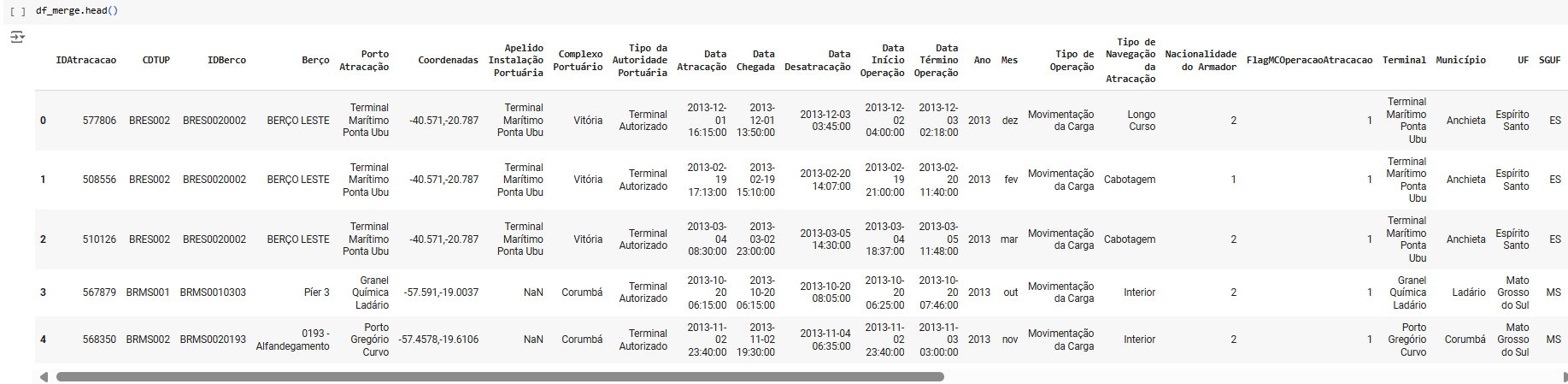
**Tempo\_operacao**: Calcula a diferença entre a Data Término Operação e a Data Início Operação, representando a duração total da operação.

**Tempo\_espera\_final**: Calcula a diferença entre a Data Desatracação e a Data Término Operação, representando o tempo de espera após o término da operação até a desatracação.

**Tempo\_ocupacao**: Calcula a diferença entre a Data Desatracação e a Data Atracação, representando o tempo total em que a mercadoria esteve ocupando o espaço de atracação.

Para todas as linhas, o resultado é convertido para segundos usando *.dt.total\_seconds(),* e em seguida, dividido por 3600 para obter o valor em horas.

**Exibindo as Primeiras Linhas do DataFrame com Tempos Calculados:**

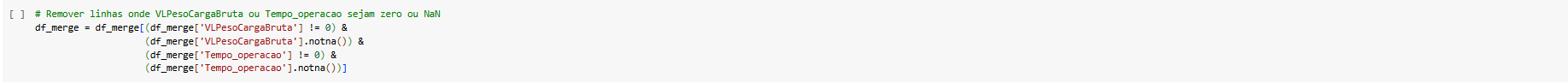


Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

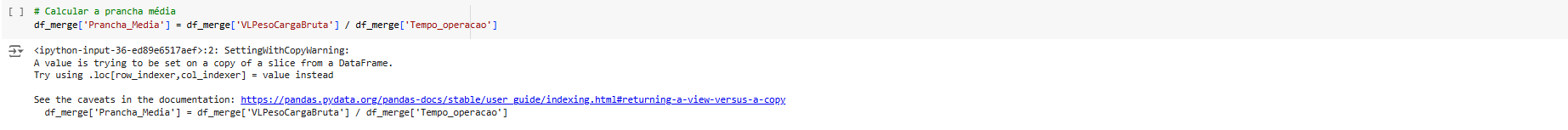
Essa operação é usada para visualizar rapidamente as cinco primeiras linha de *df\_merge*. É útil para verificar se o *DataFrame* foi corretamente filtrado.

**Remoção de Linhas com Valores Zerados ou Ausentes em VLPesoCargaBruta e Tempo\_operacao:**



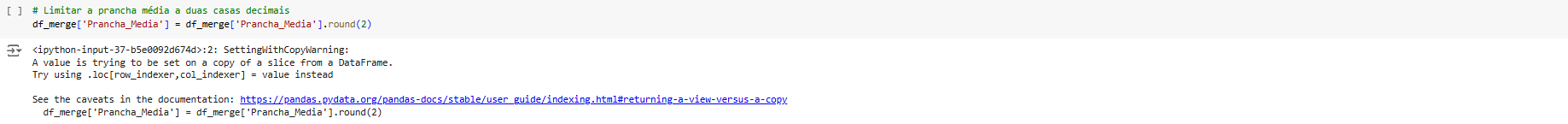
Este código remove linhas no DataFrame df\_merge onde as colunas *VLPesoCargaBruta* ou *Tempo\_operacao* contêm valores nulos (NaN) ou são iguais a zero. A limpeza desses dados é essencial para evitar valores incorretos ou irrelevantes em análises subsequentes.

**Cálculo da Prancha Média de Operação Portuária:**



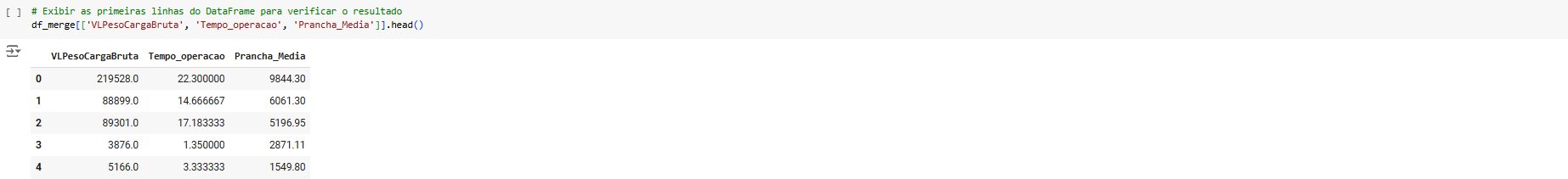
Este código calcula a **Prancha Média** para cada operação no DataFrame df\_merge, adicionando os resultados em uma nova coluna chamada *Prancha\_Media*. Esse cálculo é uma métrica usada para avaliar a eficiência da operação de carga e descarga em portos.

**Arredondamento da Prancha Média para Duas Casas Decimais:**



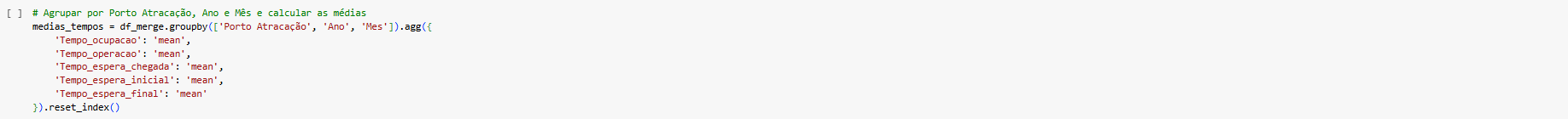
Este código limita os valores da coluna Prancha\_Media a duas casas decimais no DataFrame df\_merge. Isso ajuda a apresentar os resultados de maneira mais limpa e facilita a interpretação.

**Exibição das Primeiras Linhas para Verificação dos Resultados:**



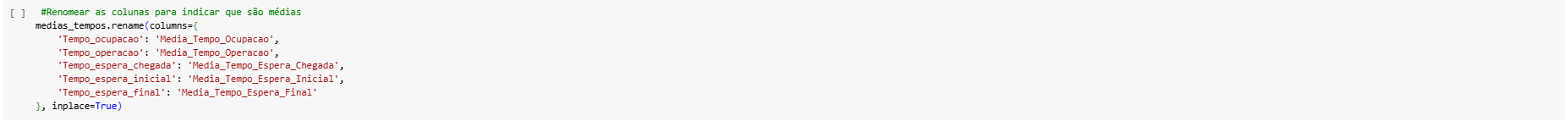
Essa operação é usada para visualizar rapidamente as cinco primeiras linha de *df\_merge*, incluindo apenas as colunas *VLPesoCargaBruta, Tempo\_operacao e Prancha\_Media*. É útil para verificar se o *DataFrame* foi corretamente filtrado.

**Cálculo das Médias de Tempos Agrupadas por Porto, Ano e Mês:**



Este código agrupa o DataFrame df\_merge pelos campos Porto Atracação, Ano e Mes, calculando a média de várias métricas de tempo para cada combinação desses agrupamentos. O resultado é armazenado no DataFrame medias\_tempos.

**Renomeação de Colunas para Indicar Média dos Tempos:**



Este código renomeia as colunas do DataFrame medias\_tempos para refletir que os valores correspondem a médias de tempos. Essa renomeação ajuda a tornar os dados mais claros e específicos.

As novas colunas se tornam:

* Media\_Tempo\_Ocupacao
* Media\_Tempo\_Operacao
* Media\_Tempo\_Espera\_Chegada
* Media\_Tempo\_Espera\_Inicial
* Media\_Tempo\_Espera\_Final

O parâmetro *inplace=True* aplica as mudanças diretamente ao DataFrame medias\_tempos, evitando a necessidade de criar uma cópia.

**Mesclagem das Médias de Tempos ao DataFrame Original:**



Mescla o DataFrame *medias\_tempos*, que contém as médias de tempos calculadas, de volta ao DataFrame original df\_merge, adicionando as colunas de médias por porto, ano e mês para cada linha correspondente.

**Função para Converter Valores em Horas para o Formato HH:MM:**

Uma imagem contendo Forma

Descrição gerada automaticamente

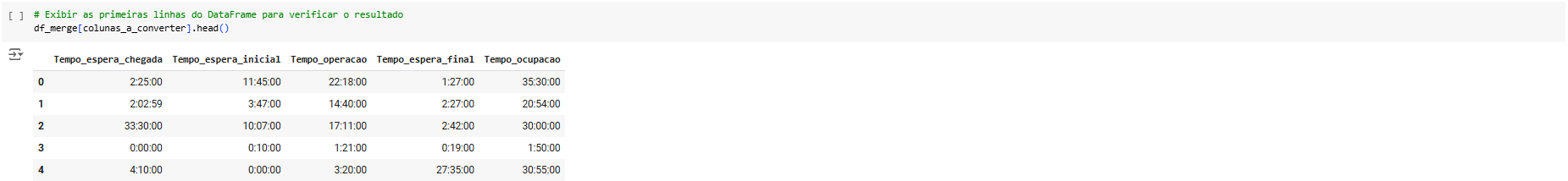
Define uma função, converter\_horas\_para\_hhmmss, que converte valores em horas para o formato HH:MM:SS. Além disso, uma lista chamada colunas\_a\_converter especifica as colunas que devem ser convertidas usando essa função.

**Aplicação da Função de Conversão HH:MM nas Colunas de Tempo:**



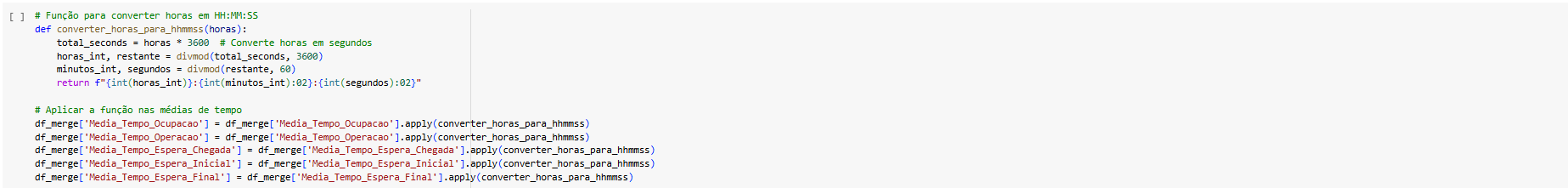
Aplica a função converter\_horas\_para\_hhmmss a cada coluna especificada na lista colunas\_a\_converter no DataFrame df\_merge. Cada coluna é sobrescrita com os valores convertidos no formato HH:MM:SS, facilitando a leitura e padronização dos dados de tempo.

**Exibição das Primeiras Linhas para Verificação das Colunas de Tempo Convertidas:**



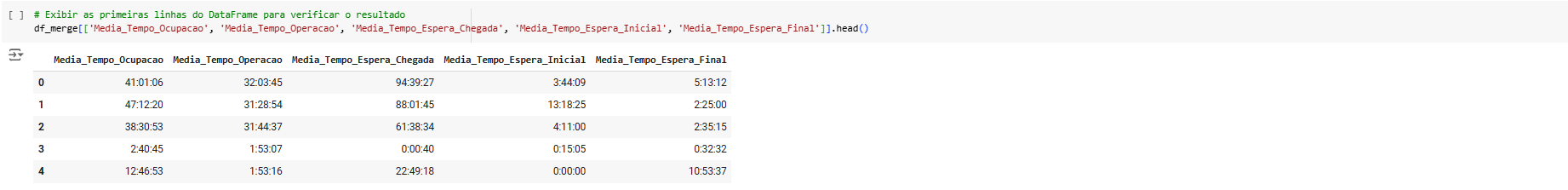
Este comando exibe as primeiras linhas do DataFrame df\_merge, mostrando apenas as colunas de tempo especificadas em colunas\_a\_converter. O objetivo é verificar se os valores foram corretamente convertidos para o formato HH:MM:SS.

**Conversão dos Tempos Médios para o Formato HH:MM:**



Este código aplica a função converter\_horas\_para\_hhmmss às colunas de médias de tempo no DataFrame df\_merge. O objetivo é converter os valores das colunas que representam médias de tempo (como ocupação, operação e espera) para o formato HH:MM:SS.

**Exibição das Primeiras Linhas para Verificação das Médias de Tempo Convertidas:**



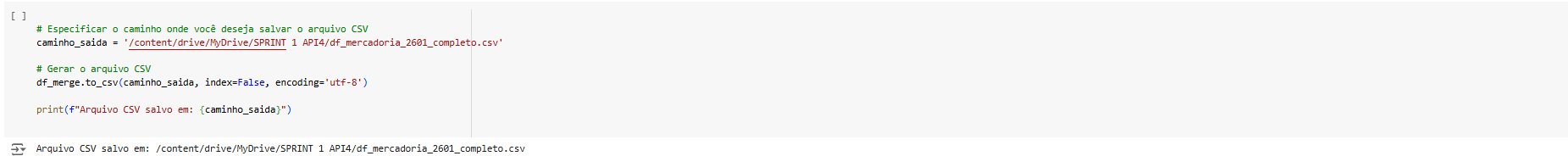
Este comando exibe as primeiras linhas do DataFrame df\_merge, mostrando as colunas de médias de tempo após a conversão para o formato HH:MM:SS. O objetivo é verificar se a aplicação da função de conversão foi bem-sucedida.

**Substituição de Pontos por Vírgulas nas Colunas Antes de Salvar como CSV:**



Este código substitui os pontos por vírgulas nas colunas VLPesoCargaBruta e Prancha\_Media do DataFrame df\_merge. Esse ajuste é feito para garantir que os valores dessas colunas sejam apresentados no formato adequado, de acordo com convenções de países que utilizam vírgula como separador decimal (como no Brasil).

**Salvando o DataFrame como um Arquivo CSV:**



c**aminho\_saida**: Especifica o caminho e o nome do arquivo CSV que será criado

**df\_merge.to\_csv(...)**: Utilizado para exportar o DataFrame para um arquivo CSV.

**index=False**: Especifica que os índices do DataFrame não devem ser escritos no arquivo CSV.

**encoding='utf-8'**: Este argumento define a codificação do arquivo, garantindo que caracteres especiais sejam tratados corretamente.

**print (f"Arquivo CSV salvo em: {caminho\_saida}")**: Esse comando imprime uma mensagem confirmando que o arquivo CSV foi salvo e exibe o caminho onde ele foi armazenado.