

Pontifícia Universidade Católica do Rio de Janeiro

Ciência de Dados e Analytics

FILIPPE RODRIGUES MUSSATO

RA: 4052023001430

**MVP**

**Sprint: Engenharia de Dados (40530010057\_20240\_01)**

ITAJAÍ

2024

## Sumário

Objetivo .....	3
Problema.....	3
Busca pelos dados .....	3
Coleta .....	4
Banco de dados.....	4
Criando conexão no banco de dados.....	4
Executando consultas.....	5
Lendo dados do Excel a partir de uma URL e retornando um Dataframe.....	5
Criar tabela no MySQL.....	6
Inserindo dados no Dataframe no MySql.....	6
Conectando no banco de dados .....	6
URL e Carga do arquivo da URL.....	7
Exclusão de coluna, Criação de tabela e inserção de dados .....	7
Modelagem .....	7
Clusterização.....	8
.....	9
Catalogo de dados .....	9
Linhagem dos dados.....	9
Carga.....	10
ETL.....	10

Seleção e criação do novo dataframe.....	10
Transformação de valores monetários.....	11
Filtro de dados.....	11
Análise.....	11
Qualidade dados.....	11
Análise/Solução do problema .....	12
Variação dos valores ao longo do tempo .....	12
Rússia.....	13
Ucrânia .....	14
Autoavaliação.....	15

## Objetivo

Construir um pipeline de dados utilizando tecnologias na nuvem. O pipeline irá envolver a busca, coleta, modelagem, carga e análise dos dados.

## Problema

Considerando os problemas interessantes para estudar, pensei nas possíveis e esperadas oscilações nas negociações internacionais de países em guerra, especificamente Rússia e Ucrânia. Acredito que é viável identificar questões a partir dessa premissa. Assim, os seguintes problemas foram levantados:

- É possível, à partir de análise dados, verificar se o número de exportações do Brasil para Rússia e Ucrânia oscilaram devido a guerra em que ambos países se encontram?
- Se houve oscilação, qual país entre Rússia e Ucrânia diminuiu mais, percentualmente?

Antes de iniciar, deixo abaixo o link com o notebook do databricks:

## Busca pelos dados

Diante do problema proposto, procurei por dados públicos de exportações que abrangessem o período pré e pós-guerra. Os dados foram obtidos a partir do link: <https://comexstat.mdic.gov.br/pt/geral>. Foi extraído o CSV desse link e os dados foram armazenados em um banco MySQL na nuvem. O arquivo CSV bruto está anexado no GitHub. No tópico 'Coleta', veremos como hospedar os dados na nuvem e trabalhar com eles utilizando o Databricks

Link encurtado: <https://encurtador.com.br/8ewsj>

Link longo: <https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/605142183097994/2707862637624503/4887151886012638/latest.html>

## Coleta

Nesta etapa do projeto, optei por usar o Railway para hospedar o dataset na nuvem. A partir do notebook do Databricks, criei o banco de dados, estabeleci a conexão e, por fim, conectei ao mesmo.

## Banco de dados

```
# Banco de Dados
MYDATABASE = "railway"
HOST = "roundhouse.proxy.rlwy.net"
PASSWORD = "KunTwQxfRPzhwqORnPdWiLVsHVcPpjKl"
PORT = 18305
USER = "root"
```

## Criando conexão no banco de dados

```
# Função para criar a conexão
def create_connection(host_name, user_name, user_password, db_name, port):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password,
            database=db_name,
            port=port
        )
        print("Conexão com MySQL bem-sucedida")
    except Error as e:
        print(f"Erro: '{e}'")
    return connection
```

## Executando consultas

```
# Função para executar consultas
def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        print("Consulta executada com sucesso")
    except Error as e:
        print(f"Erro: '{e}'")
```

## Lendo dados do Excel a partir de uma URL e retornando um Dataframe

```
# Função para ler dados do Excel a partir de uma URL e retornar um DataFrame
def load_excel_from_url(url):
    response = requests.get(url)
    if response.status_code == 200:
        file_bytes = BytesIO(response.content)
        df = pd.read_excel(file_bytes, engine='openpyxl') # Especifica o engine
        return df
    else:
        raise Exception(f"Erro ao baixar o arquivo: Status code {response.status_code}")
```

## Criar tabela no MySQL

```
# Função para criar a tabela no MySQL com base no DataFrame
def create_table_from_df(connection, df, table_name):
    cursor = connection.cursor()
    # Verificar o tamanho máximo de cada coluna
    column_types = []
    for col in df.columns:
        max_length = df[col].astype(str).map(len).max()
        # Adicionar margem ao tamanho máximo
        max_length_with_margin = int(max_length * 1.2)
        column_type = f"{col} VARCHAR({max_length_with_margin})"
        column_types.append(column_type)
    cols = ", ".join(column_types)
    create_table_query = f"CREATE TABLE IF NOT EXISTS {table_name} ({cols});"
    execute_query(connection, create_table_query)
```

## Inserindo dados no Dataframe no MySql

```
# Função para inserir dados do DataFrame no MySQL
def insert_data_from_df(connection, df, table_name):
    cursor = connection.cursor()
    for _, row in df.iterrows():
        placeholders = ", ".join(["%s"] * len(row))
        insert_query = f"INSERT INTO {table_name} VALUES ({placeholders})"
        cursor.execute(insert_query, tuple(row))
    connection.commit()
    print("Dados inseridos com sucesso")
```

## Conectando no banco de dados

```
connection = create_connection(HOST, USER, PASSWORD, MYDATABASE,
PORT)
```

## URL e Carga do arquivo da URL

```
# URL do arquivo Excel no Google Drive
file_url = 'https://drive.google.com/uc?export=download&id=1NYjYXbjNc20jN4IbRgx2wIP11y6akSO_'

# Carregar o arquivo Excel da URL
df = load_excel_from_url(file_url)
```

## Exclusão de coluna, Criação de tabela e inserção de dados

```
# Excluir a coluna 'Descrição NCM'
df = df.drop(columns=['Descrição NCM'])

# Criar a tabela no MySQL com base no DataFrame
create_table_from_df(connection, df, "Russi_Ucrania_MVP")

# Inserir os dados do DataFrame no MySQL
insert_data_from_df(connection, df, "Russi_Ucrania_MVP")
```

## Modelagem

Como estamos lidando com uma tabela flat, não foi necessário adotar esquemas como snowflake ou estrela. Neste capítulo, abordaremos a clusterização no Databricks, catálogo de dados do dataset e linhagem dos dados.




## Clusterização

Abaixo as evidências da criação do cluster no Databricks:

Compute > Preview

### My Cluster - MVP PUC

[Configuration](#) [Notebooks \(1\)](#) [Libraries](#) [Event log](#) [Spark UI](#) [Driver logs](#) [Metrics](#) [Apps](#) [Spark compute UI - Master](#) 

---

Databricks Runtime Version

12.2 LTS (includes Apache Spark 3.3.2, Scala 2.12)

Driver type

Community Optimized 15.3 GB Memory, 2 Cores, 1 DBU


Instance

Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For more configuration options [link](#), please upgrade your Databricks subscription. [link](#)

---

[Instances](#) [Spark](#) [JDBC/ODBC](#)

---

Availability zone 

auto

×


### Attach to a compute resource

Your notebook must be attached to a compute resource to execute commands. Create new compute to run your command.

Name

My Cluster

Runtime

Runtime: 12.2 LTS (Scala 2.12, Spark 3.3.2) 

[Advanced Configuration](#) [Cancel](#) [Create, Attach, & Run](#)

Connected

Go to last run cell

● My Cluster - MVP PUC

Runtime

Driver

DBR 12.2 LTS • Spark 3.3.2 • Scala 2.12

dev-tier-node • 15.25 GB • 2 Cores

## Catálogo de dados

Catálogo de Dados		
Nome da Coluna	Tipo de Dado	Descrição
Index	Integer	Índice dos dados, numeração sequencial.
Fluxo	String	Tipo de operação (Exportação e importação).
Ano	Integer	Ano da operação
Mês	String	Mês da exportação com formato "MM. Nome do Mês" (e.g., 04. Abril).
Países	String	País de destino da exportação (e.g., Rússia).
Bloco Econômico	String	Bloco econômico do país de destino (e.g., Europa).
Código NCM	String	Código NCM do produto exportado.
UF do Produto	String	Unidade Federativa de origem do produto (e.g., Mato Grosso).
Via	String	Modal de transporte utilizado (e.g., Marítima).
URF	String	Unidade da Receita Federal responsável pela exportação (e.g., 0217800 - ALF - BELÉM).
Unidade Estatística	String	Unidade de medida estatística utilizada (e.g., Tonelada Métrica Líquida).
Valor US\$ FOB	Float	Valor da exportação em dólares americanos FOB (Free on Board).
Quilograma Líquido	Float	Peso líquido em quilogramas da exportação.
Quantidade Estatística	Float	Quantidade estatística medida.

Obs.: Planilha 'Catálogo de dados' está anexa no Github para melhor visualização.

## Linhagem dos dados

Os dados foram obtidos a partir do link: <https://comexstat.mdic.gov.br/pt/geral>. Foi extraído o CSV desse link e os dados foram armazenados em um banco MySQL na nuvem. O arquivo CSV bruto está anexado no GitHub. Todo o caminho para coleta e transformação dos dados estão aqui documentados.

## Carga

Após estabelecer a conexão com o banco de dados, criaremos um novo dataframe chamado 'df\_analysis'. Utilizaremos a função 'select' para extrair apenas as colunas necessárias para a análise e resposta ao problema proposto. Em seguida, transformaremos o tipo de valor de uma coluna (os detalhes serão explicados no tópico 'Qualidade de dados'), filtraremos os dados relevantes para os países Brasil, Rússia e Ucrânia, deixando o dataframe pronto para a análise.

## ETL

### Seleção e criação do novo dataframe

Função select para extração das principais colunas para análise de dados. Com essa seleção, carregaremos o novo dataframe chamado 'df\_analysis'.

```
# Consultando os dados da nova tabela
query = """
SELECT
    `Fluxo`,
    `Ano`,
    `Mês`,
    `Países`,
    `Bloco Econômico`,
    `Código NCM`,
    `Valor US$ FOB`
FROM Russi_Ucrania_MVP
"""

df_analysis = read_query(connection, query)
```

## Transformação de valores monetários.

Durante o processo de ETL, antes de preparar o dataframe para análise, tornou-se evidente a necessidade de converter os valores monetários da coluna 'Valor US\$ FOB'. Essa coluna continha valores com separação por vírgula ao invés de ponto, o que impossibilitava o uso de funções matemáticas e inviabilizava a análise e a solução do problema proposto.

```
# 1 - Transformar os valores monetários
df_analysis['Valor US$ FOB'] = pd.to_numeric(df_analysis['Valor US$ FOB'].str.replace(',', ''), errors='coerce')
```

## Filtro de dados

Filtraremos os dados relevantes para os países Brasil, Rússia e Ucrânia.

```
# 2 - Filtrar dados relevantes para o Brasil, Rússia e Ucrânia
df_analysis = df_analysis[df_analysis['Países'].isin(['Brasil', 'Rússia', 'Ucrânia'])]
```

## Análise

### Qualidade dados

Para resolver o problema proposto, não foi necessário analisar a qualidade de todos os dados do dataset, pois apenas algumas colunas foram suficientes para responder à questão principal. Focamos nossa análise de qualidade de dados nas colunas 'Data', 'Valor US\$' e 'Países'. A única coluna que apresentou problemas foi 'Valor US\$', onde os valores estavam separados por vírgulas em vez de pontos, o que impossibilitava a realização de operações matemáticas essenciais para a resolução do problema, especialmente na soma dos valores FOB para análise de oscilações ao longo do tempo.

A solução encontrada foi simples: substituir as vírgulas por pontos. Assim, os valores foram corrigidos e estão prontos para operações matemáticas. Não foi necessário preocupar-se com os valores após a vírgula, já que esta coluna utiliza o ponto como separador decimal. Essa solução proposta mostrou-se suficiente e eficaz.

```
# 1 - Transformar os valores monetários
df_analysis['Valor US$ FOB'] = pd.to_numeric(df_analysis['Valor US$ FOB'].str.replace(',', ''), errors='coerce')
```

## Análise/Solução do problema

O problema proposto foi:

- É possível, à partir de análise dados, verificar se o número de exportações do Brasil para Rússia e Ucrânia oscilaram devido a guerra em que ambos países se encontram?

A resposta é sim. Uma vez que os dados estão tratados e carregados, podemos plotar um gráfico que mostra o valor importado por país ao longo do tempo. Sabendo a data de início da guerra, podemos tirar conclusões comparando período pré e pós.

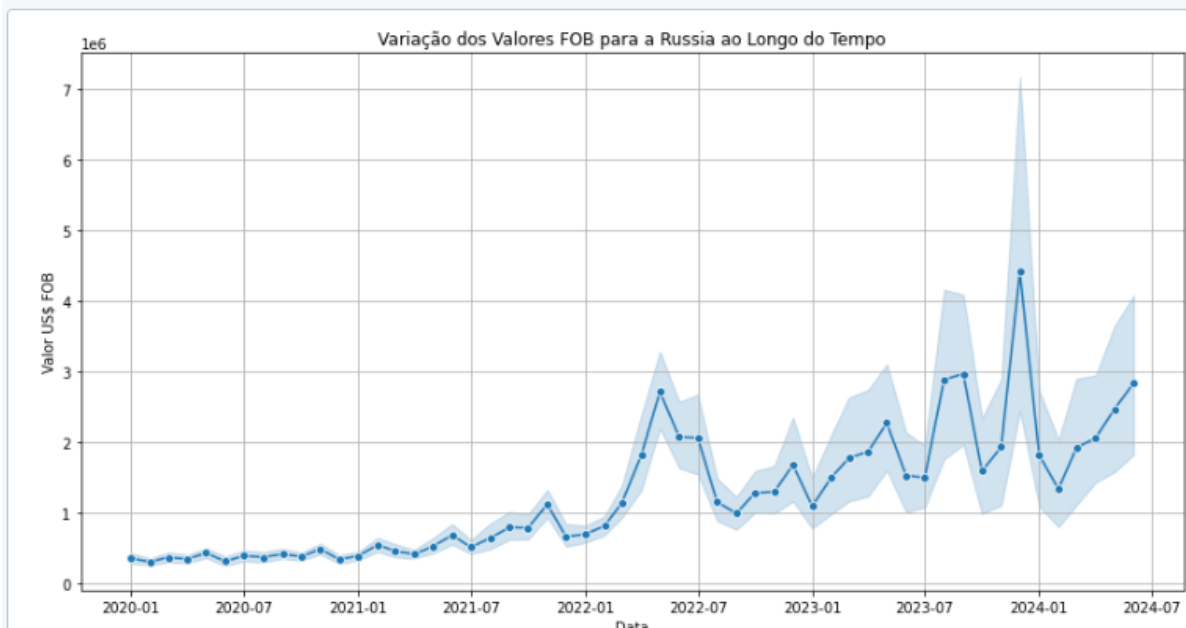
## Variação dos valores ao longo do tempo

```
# Analisar a variação dos valores FOB ao longo do tempo
df_analysis['Data'] = pd.to_datetime(df_analysis['Ano'].astype(str) + '-' + df_analysis['Mês'].str.split('.').str[0] + '-01')
df_analysis = df_analysis.sort_values(by='Data')
```

## Rússia

```
# Filtrando apenas os dados da Rússia
df_russia = df_analysis[df_analysis['Países'] == 'Rússia']

# Plotando os valores FOB da Rússia ao longo do tempo
plt.figure(figsize=(14, 7))
sns.lineplot(data=df_russia, x='Data', y='Valor US$ FOB', marker='o')
plt.title('Variação dos Valores FOB para a Rússia ao Longo do Tempo')
plt.xlabel('Data')
plt.ylabel('Valor US$ FOB')
plt.grid(True)
plt.show()
```



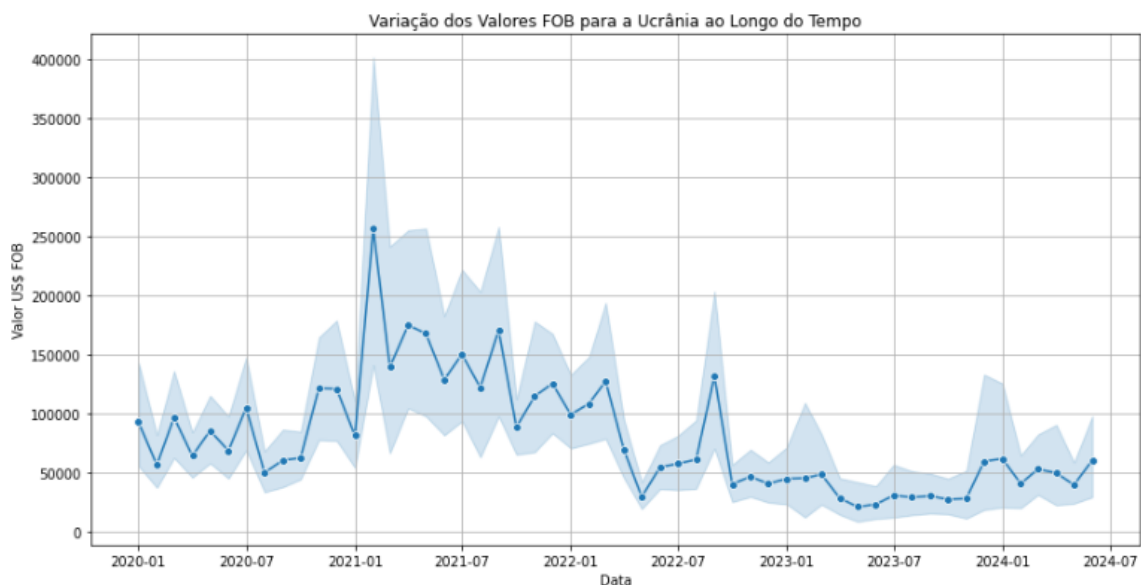
Sabendo que a guerra iniciou em 24 de fevereiro de 2022, conseguimos visualizar o período pré e pós guerra. Tinha a hipótese que os números de importações da Rússia iriam despencar pós 24 de fevereiro de 2022, mas os dados não nos mostram isso, pelo contrário, vemos um aumento significativo pós início da guerra. Não tenho dados suficientes para saber o motivo e permanência do aumento dos valores importados. Algumas hipóteses podem ser levantadas, como por exemplo, aumento na importação de itens usados na guerra como munições, armamentos, etc. Mas isso são apenas suposições, nosso MVP não tem como

objetivo explicar o motivo, apenas conseguir aferir as oscilações, e este objetivo foi alcançado.

## Ucrânia

```
# Filtrando apenas os dados da Ucrânia
df_ukraine = df_analysis[df_analysis['Países'] == 'Ucrânia']

# Plotando os valores FOB da Ucrânia ao longo do tempo
plt.figure(figsize=(14, 7))
sns.lineplot(data=df_ukraine, x='Data', y='Valor US$ FOB', marker='o')
plt.title('Variação dos Valores FOB para a Ucrânia ao Longo do Tempo')
plt.xlabel('Data')
plt.ylabel('Valor US$ FOB')
plt.grid(True)
plt.show()
```



Neste cenário, observamos algo totalmente diferente. A hipótese lógica e aparentemente óbvia de que o volume de importações de um país em guerra diminuiria após o início do conflito é verificada pelos dados. É claro e evidente a queda no volume de importação pós início da guerra. Com isso, respondemos a segunda parte do primeiro problema proposto.

## Autoavaliação

Tudo isso é extremamente novo para mim, sou gerente comercial de uma empresa e sempre imaginei que os dados iriam me ajudar a tomar decisões de maneira mais estratégica. Estava certo, aprender a entender e manipular dados (mesmo que com muito menos destreza que meus colegas e professores) tem me ajudado muito na compreensão de dados dentro do universo comercial.

Em Junho, fui a Taiwan em uma viagem internacional e isso comprometeu muito minha agenda de estudos, sofri para finalizar as aulas e muitas delas foram realizadas de maneira apressada.

Achei muito legal o problema que desenhei, trazendo algo da vida real para o ambiente acadêmico. Tenho muita curiosidade em entender movimentos comerciais de países durante conflitos.

Usar o Databricks foi um desafio muito grande. Não fazia a menor ideia de como usar e tive que ver MUITOS vídeos para entender como operar. Desenvolver esse MVP foi muito legal porque como eu nunca trabalhei na área de governança de dados, consegui sentir um gostinho de como é pensar em formas de armazenamento e governança de dados.

Não consegui pensar em uma forma de montar uma resposta para segunda parte do problema proposto: “Se houve oscilação, qual país entre Rússia e Ucrânia diminuiu mais, percentualmente”.

Finalizo o MVP com o sentimento de que o suficiente foi entregue. Nada além do suficiente. Embora eu tenha gostado do problema proposto, acredito que poderia ter ido mais fundo na análise, ter “brincado” mais com os dados, ter levantado mais insights, mais combinações de dados.

Por outro lado, embora tenha sido apenas “ok”, finalizo com um sentimento de grande alívio.