databricksMVP PUC RIO - FILIPE RODRIGUES MUSSATO (final version)

(https://databricks.com)

```
%pip install mysql-connector-python
%pip install openpyxl
```

```
Python interpreter will be restarted.
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.0.0-cp39-cp39-manylinux_2_17_x86_64.whl (19.3 MB)
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-9.0.0
Python interpreter will be restarted.
Python interpreter will be restarted.
Collecting openpyxl
  Downloading openpyxl-3.1.5-py2.py3-none-any.whl (250 kB)
Collecting et-xmlfile
  Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.1.5
Python interpreter will be restarted.
```

```
import mysql.connector
from mysql.connector import Error
import pandas as pd
import requests
from io import BytesIO
import matplotlib.pyplot as plt
import seaborn as sns
```

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

```
# 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

# 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}'")

# 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}")

# 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)

# 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")

# Conectar ao banco de dados
connection = create_connection(HOST, USER, PASSWORD, MYDATABASE, PORT)

# 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)

# 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")
```

```
Conexão com MySQL bem-sucedida
Erro: '1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Econômico VARCHAR(7), Código NCM VARCHAR(9), UF do Produto VARCHAR(22), Via VA' at line 1'
Dados inseridos com sucesso
```

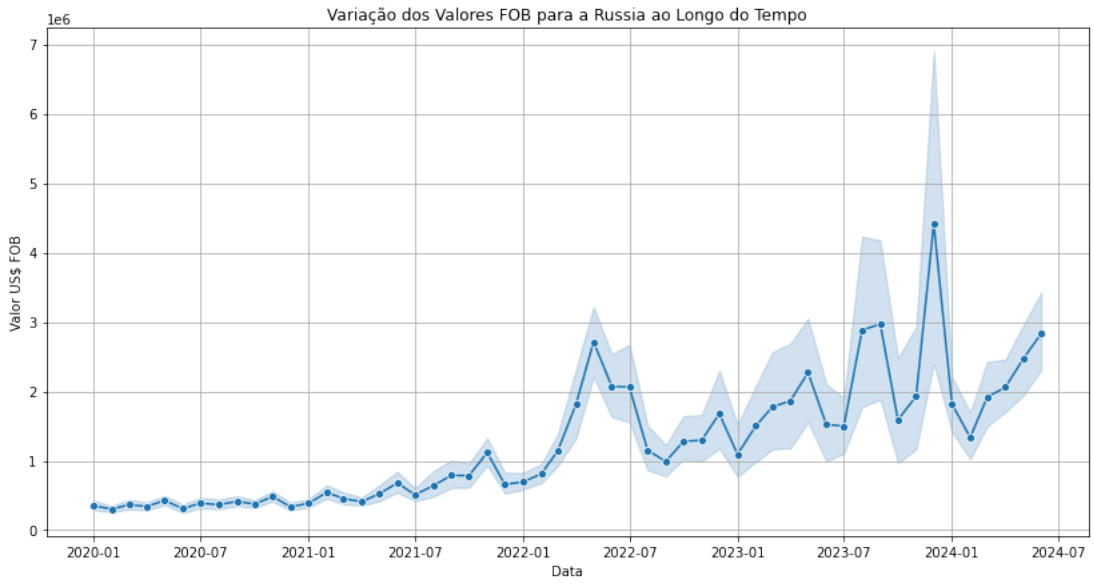
```
# Essa função, serve para executar consultas de leitura
def read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
        columns = cursor.column_names
        return pd.DataFrame(result, columns=columns)
    except Error as e:
        print(f"Erro: '{e}'")
```

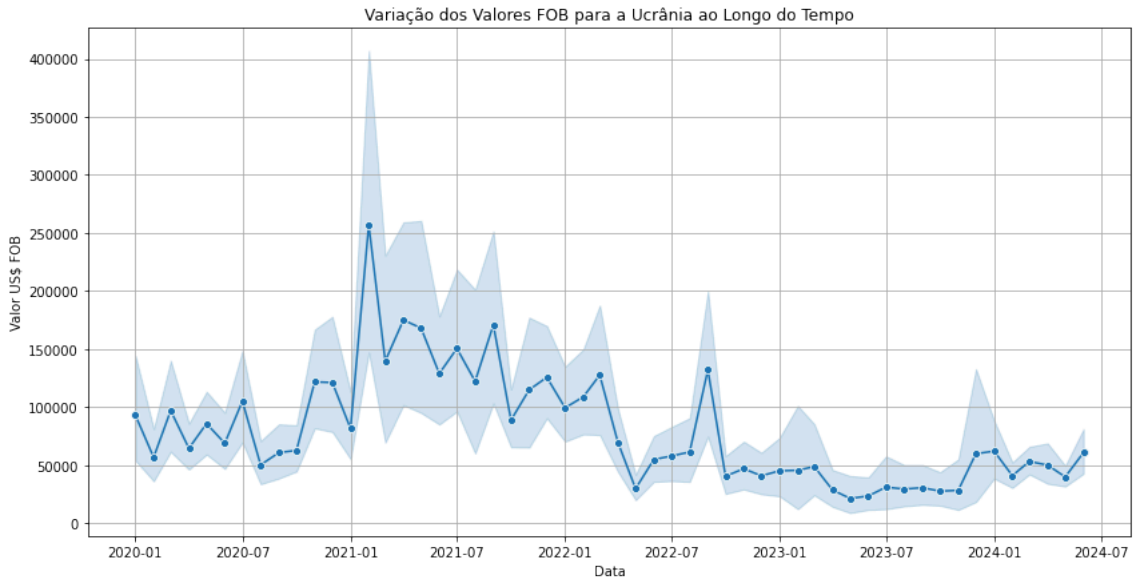
```
# Consultar os dados da nova tabela
query = """
SELECT
    `Fluxo`,
    `Ano`,
    `Mês`,
    `Países`,
    `Bloco Econômico`,
    `Código NCM`,
    `Valor US$ FOB`
FROM Russi_Ucrania_MVP
"""
df_analysis = read_query(connection, query)
```

```
# Processo de ETL:

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

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

```
# 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')
```

```
# Filtrar dados apenas para a Russia
df_ukraine = df_analysis[df_analysis['Países'] == 'Rússia']

# Plotar 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 Russia ao Longo do Tempo')
plt.xlabel('Data')
plt.ylabel('Valor US$ FOB')
plt.grid(True)
plt.show()
```


Variação dos Valores FOB para a Russia ao Longo do Tempo
1e6
Valor US$ FOB
Data
0
1
2
3
4
5
6
7
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
2024-07


Variação dos Valores FOB para a Ucrânia ao Longo do Tempo
Valor US$ FOB
Data
0
50000
100000
150000
200000
250000
300000
350000
400000
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
2024-01
2024-07