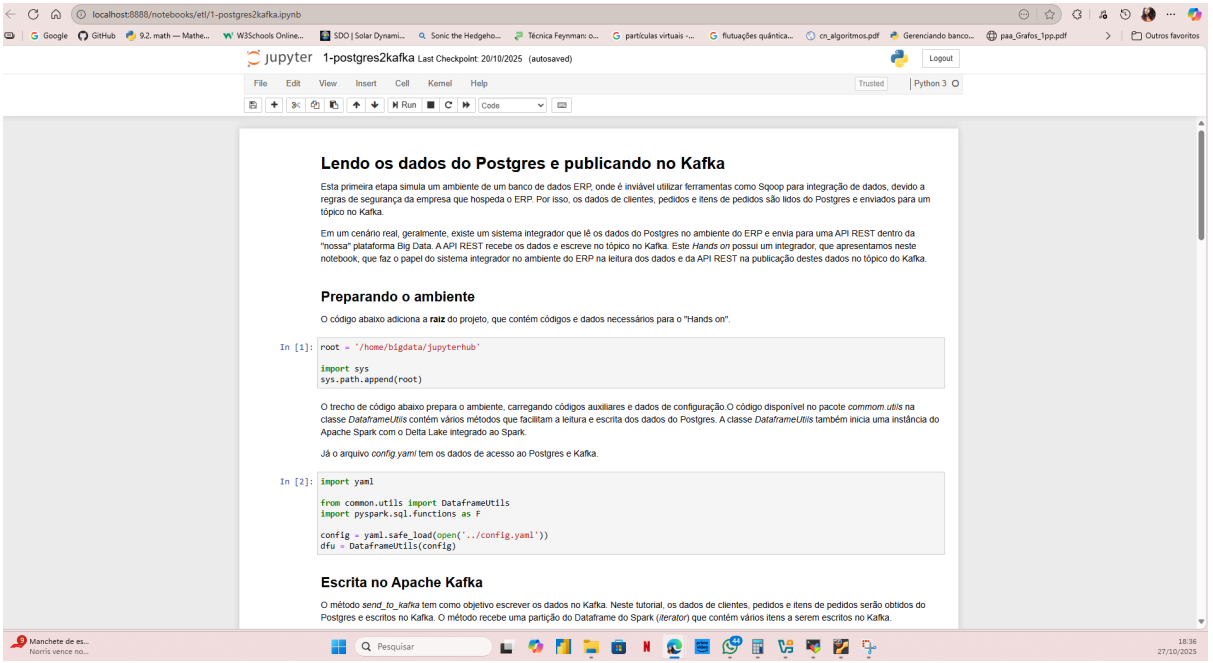
Evidências de execução do notebook 1-postgres1kafka



Lendo os dados do Postgres e publicando no Kafka

Esta primeira etapa simula um ambiente de um banco de dados ERP, onde é inviável utilizar ferramentas como Sqoop para integração de dados, devido a regras de segurança da empresa que hospeda o ERP. Por isso, os dados de clientes, pedidos e itens de pedidos são lidos do Postgres e enviados para um tópico no Kafka.

Em um cenário real, geralmente, existe um sistema integrador que lê os dados do Postgres no ambiente do ERP e envia para uma API REST dentro da "nossa" plataforma Big Data. A API REST recebe os dados e escreve no tópico no Kafka. Este *Hands on* possui um integrador, que apresentamos neste notebook, que faz o papel do sistema integrador no ambiente do ERP na leitura dos dados e da API REST na publicação destes dados no tópico do Kafka.

Preparando o ambiente

O código abaixo adiciona a **raiz** do projeto, que contém códigos e dados necessários para o "Hands on".

In [1]:
```
root = '/home/bigdata/jupyterhub'

import sys
sys.path.append(root)
```

O trecho de código abaixo prepara o ambiente, carregando códigos auxiliares e dados de configuração.O código disponível no pacote *commom.utils* na classe *DataframeUtils* contém vários métodos que facilitam a leitura e escrita dos dados do Postgres. A classe *DataframeUtils* também inicia uma instância do Apache Spark com o Delta Lake integrado ao Spark.

Já o arquivo *config.yaml* tem os dados de acesso ao Postgres e Kafka.

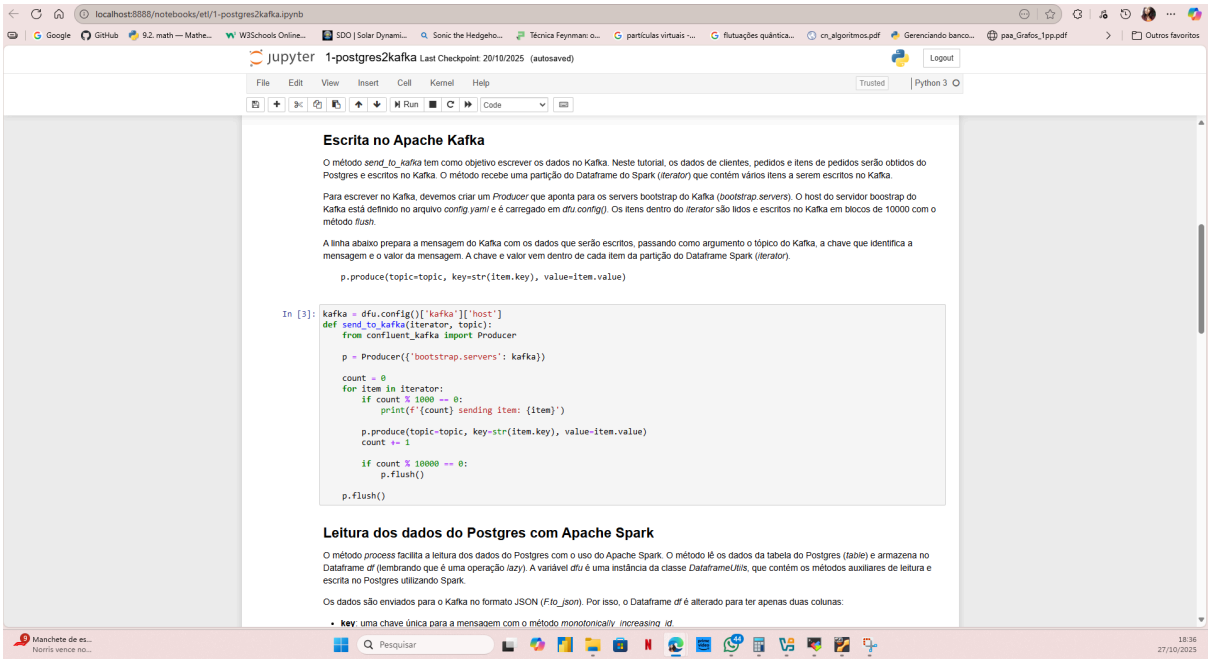
In [2]:
```
import yaml

from common.utils import DataframeUtils
import pyspark.sql.functions as F

config = yaml.safe_load(open('../config.yaml'))
dfu = DataframeUtils(config)
```

Escrita no Apache Kafka

O método *send_to_kafka* tem como objetivo escrever os dados no Kafka. Neste tutorial, os dados de clientes, pedidos e itens de pedidos serão obtidos do Postgres e escritos no Kafka. O método recebe uma partição do Dataframe do Spark (*iterator*) que contém vários itens a serem escritos no Kafka.



Escrita no Apache Kafka

O método *send_to_kafka* tem como objetivo escrever os dados no Kafka. Neste tutorial, os dados de clientes, pedidos e itens de pedidos serão obtidos do Postgres e escritos no Kafka. O método recebe uma partição do Dataframe do Spark (*iterator*) que contém vários itens a serem escritos no Kafka.

Para escrever no Kafka, devemos criar um *Producer* que aponta para os servers bootstrap do Kafka (*bootstrap.servers*). O host do servidor boostrap do Kafka está definido no arquivo *config.yaml* e é carregado em *dfu.config()*. Os itens dentro do *iterator* são lidos e escritos no Kafka em blocos de 10000 com o método *flush*.

A linha abaixo prepara a mensagem do Kafka com os dados que serão escritos, passando como argumento o tópico do Kafka, a chave que identifica a mensagem e o valor da mensagem. A chave e valor vem dentro de cada item da partição do Dataframe Spark (*iterator*).

```
p.produce(topic=topic, key=str(item.key), value=item.value)
```

In [3]:
```
kafka = dfu.config()['kafka']['host']
def send_to_kafka(iterator, topic):
    from confluent_kafka import Producer

    p = Producer({'bootstrap.servers': kafka})

    count = 0
    for item in iterator:
        if count % 1000 == 0:
            print(f'{count} sending item: {item}')

        p.produce(topic=topic, key=str(item.key), value=item.value)
        count += 1

        if count % 10000 == 0:
            p.flush()

    p.flush()
```

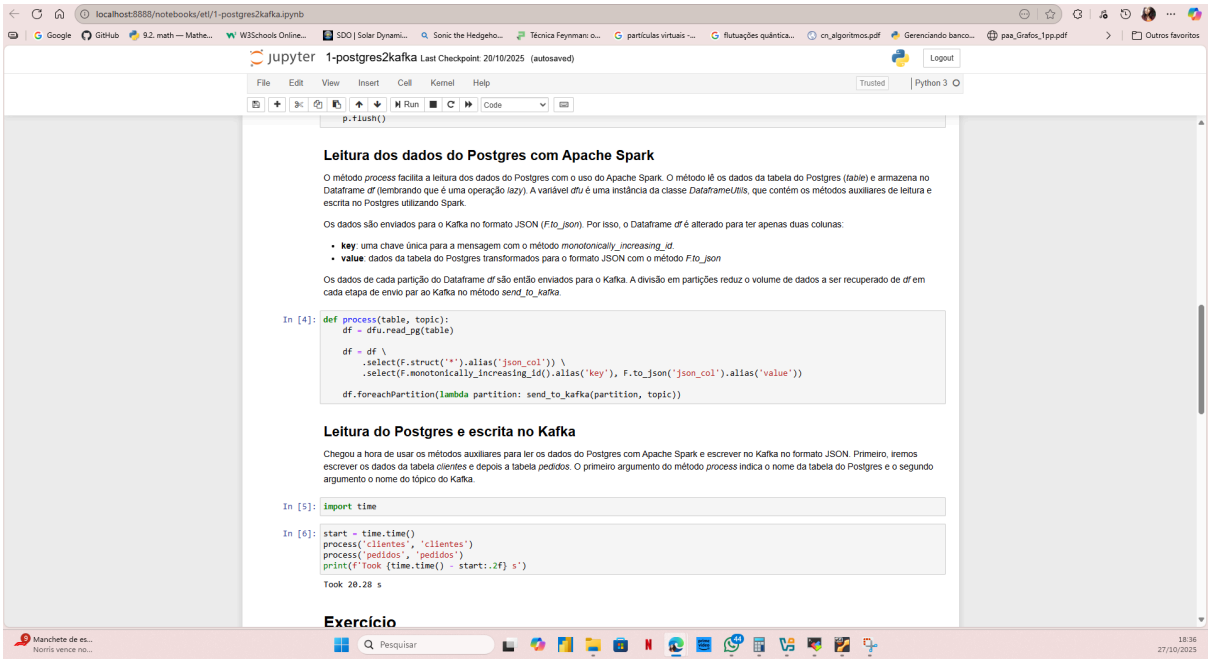
Leitura dos dados do Postgres com Apache Spark

O método *process* facilita a leitura dos dados do Postgres com o uso do Apache Spark. O método lê os dados da tabela do Postgres (*table*) e armazena no Dataframe *df* (lembrando que é uma operação *lazy*). A variável *dfu* é uma instância da classe *DataframeUtils*, que contém os métodos auxiliares de leitura e escrita no Postgres utilizando Spark.

Os dados são enviados para o Kafka no formato JSON (*F.to_json*). Por isso, o Dataframe *df* é alterado para ter apenas duas colunas:

- **key**: uma chave única para a mensagem com o método *monotonically_increasing_id*

Evidências de execução do notebook 1-postgres1kafka



p.flush()

Leitura dos dados do Postgres com Apache Spark

O método *process* facilita a leitura dos dados do Postgres com o uso do Apache Spark. O método lê os dados da tabela do Postgres (*table*) e armazena no Dataframe *df* (lembrando que é uma operação *lazy*). A variável *dfu* é uma instância da classe *DataframeUtils*, que contém os métodos auxiliares de leitura e escrita no Postgres utilizando Spark.

Os dados são enviados para o Kafka no formato JSON (*F.to_json*). Por isso, o Dataframe *df* é alterado para ter apenas duas colunas:

- **key**: uma chave única para a mensagem com o método *monotonically_increasing_id*.
- **value**: dados da tabela do Postgres transformados para o formato JSON com o método *F.to_json*

Os dados de cada partição do Dataframe *df* são então enviados para o Kafka. A divisão em partições reduz o volume de dados a ser recuperado de *df* em cada etapa de envio par ao Kafka no método *send_to_kafka*.

In [4]:
```
def process(table, topic):
    df = dfu.read_pg(table)

    df = df \
        .select(F.struct('*').alias('json_col')) \
        .select(F.monotonically_increasing_id().alias('key'), F.to_json('json_col').alias('value'))

    df.foreachPartition(lambda partition: send_to_kafka(partition, topic))
```

Leitura do Postgres e escrita no Kafka

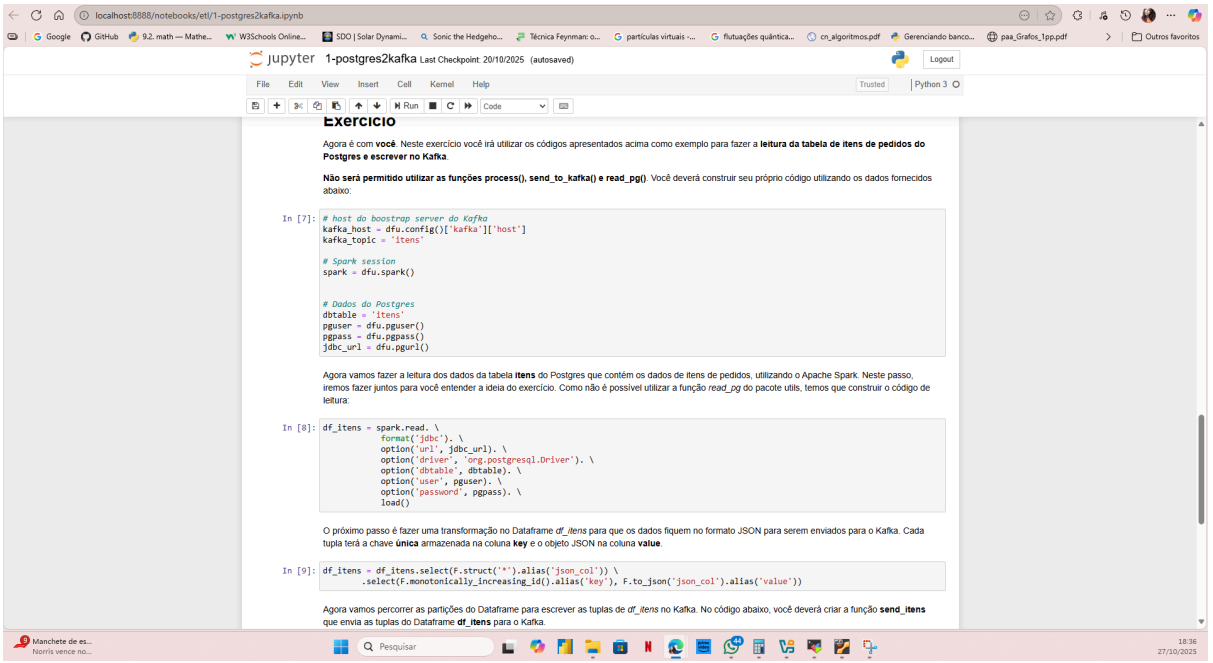
Chegou a hora de usar os métodos auxiliares para ler os dados do Postgres com Apache Spark e escrever no Kafka no formato JSON. Primeiro, iremos escrever os dados da tabela *clientes* e depois a tabela *pedidos*. O primeiro argumento do método *process* indica o nome da tabela do Postgres e o segundo argumento o nome do tópico do Kafka.

In [5]:
```
import time
```

In [6]:
```
start = time.time()
process('clientes', 'clientes')
process('pedidos', 'pedidos')
print(f'Took {time.time() - start:.2f} s')
```

```
Took 20.28 s
```

Exercício



Exercício

Agora é com **você**. Neste exercício você irá utilizar os códigos apresentados acima como exemplo para fazer a **leitura da tabela de itens de pedidos do Postgres e escrever no Kafka**.

Não será permitido utilizar as funções process(), send_to_kafka() e read_pg(). Você deverá construir seu próprio código utilizando os dados fornecidos abaixo.

In [7]:
```
# host do boostrap server do Kafka
kafka_host = dfu.config()['kafka']['host']
kafka_topic = 'itens'

# Spark session
spark = dfu.spark()


# Dados do Postgres
dbtable = 'itens'
pguser = dfu.pguser()
pgpass = dfu.pgpass()
jdbc_url = dfu.pgurl()
```

Agora vamos fazer a leitura dos dados da tabela **itens** do Postgres que contém os dados de itens de pedidos, utilizando o Apache Spark. Neste passo, iremos fazer juntos para você entender a ideia do exercício. Como não é possível utilizar a função *read_pg* do pacote utils, temos que construir o código de leitura:

In [8]:
```
df_itens = spark.read. \
            format('jdbc'). \
            option('url', jdbc_url). \
            option('driver', 'org.postgresql.Driver'). \
            option('dbtable', dbtable). \
            option('user', pguser). \
            option('password', pgpass). \
            load()
```

O próximo passo é fazer uma transformação no Dataframe *df_itens* para que os dados fiquem no formato JSON para serem enviados para o Kafka. Cada tupla terá a chave **única** armazenada na coluna **key** e o objeto JSON na coluna **value**.

In [9]:
```
df_itens = df_itens.select(F.struct('*').alias('json_col')) \
        .select(F.monotonically_increasing_id().alias('key'), F.to_json('json_col').alias('value'))
```

Agora vamos percorrer as partições do Dataframe para escrever as tuplas de *df_itens* no Kafka. No código abaixo, você deverá criar a função **send_itens** que envia as tuplas do Dataframe **df_itens** para o Kafka.

Evidências de execução do notebook 1-postgres1kafka