Into. to Big-Data | OPPE (25-08-24) | report

By Chandresh J. Sutariya (21f3001415)

Report | Problem Statement:

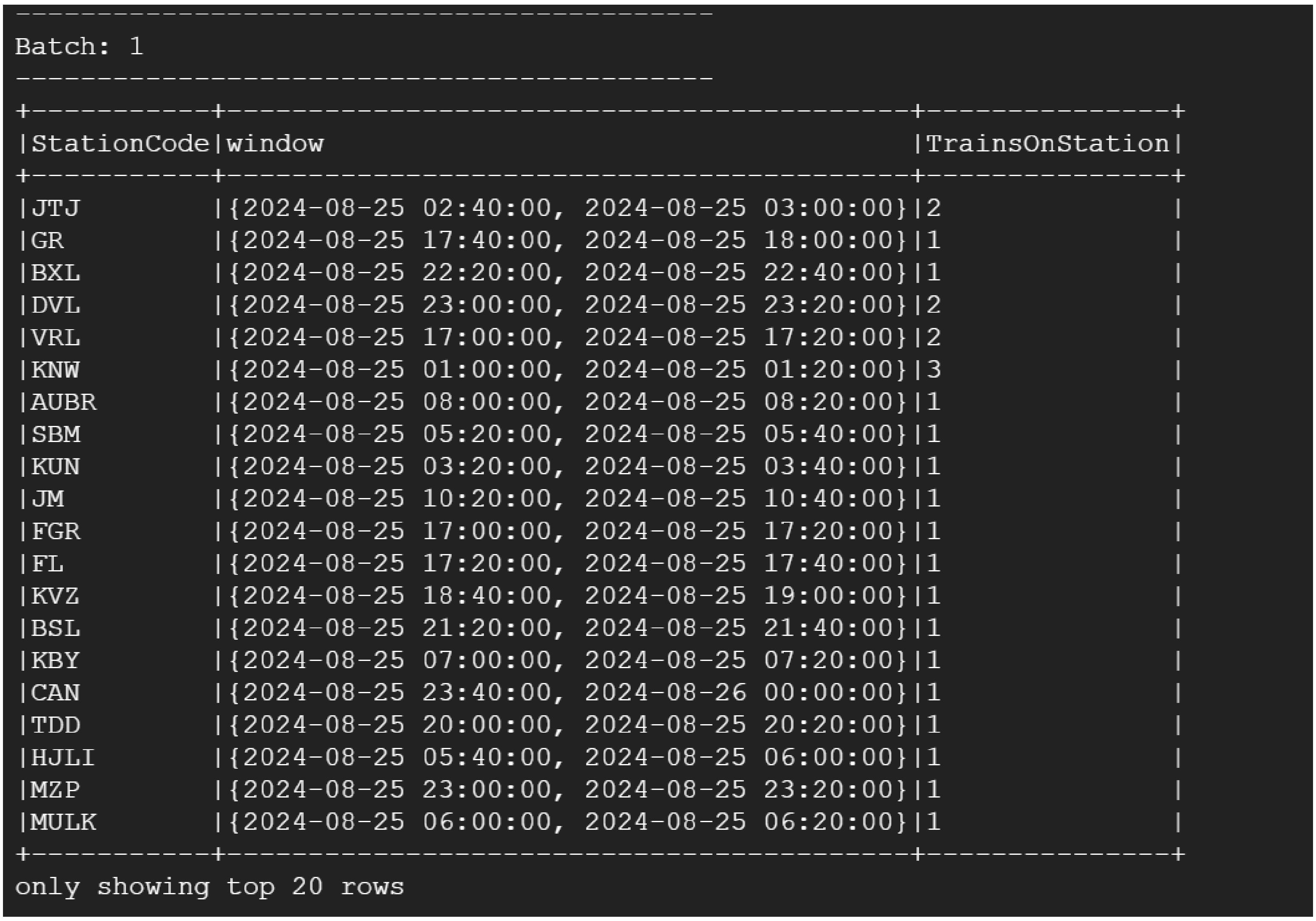
Problem Statement:

The station master at every train station is tasked with continuously assessing whether the number of platforms available in that station at any point in time during the day is sufficient for the trains expected to stop at that station. Each station master is looking for help to solve for this challenge by having at his/her fingerstep the total number of trains at that station in any 20 minute window. It is assumed that a typical train needs about 20 minutes at a station at the maximum for loading & unloading before moving on or being moved to the shed.

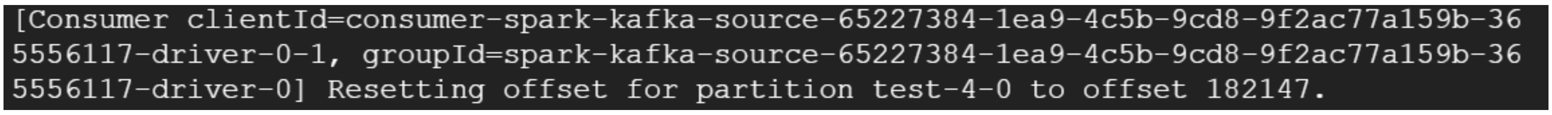
Your task is to compute the 20-minute rolling count of trains per station so that every 5 minutes, every station master that has any train stopped at their station gets this count information for their purposes. You are also required to determine per station the maximum of these counts across the entire time period of the data set.

Report | Step

1. setup kafka:
    - Zookeeper, kafka server
2. Create topic with name *Test-5*
3. Start producer file *prdcr.py* on kafka VM
4. Start consumer file *consmr.py* on DataProc Cluster VM



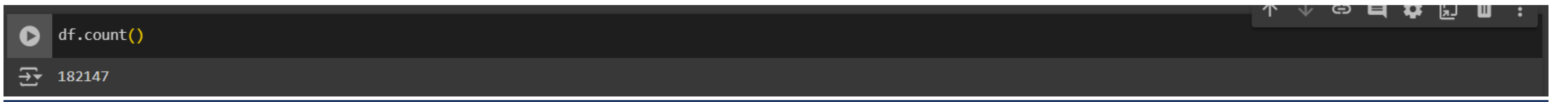
```
-------------------------------------------
Batch: 1
-------------------------------------------
+-----------+------------------------------------------+---------------+
|StationCode|window                                    |TrainsOnStation|
+-----------+------------------------------------------+---------------+
|JTJ        |{2024-08-25 02:40:00, 2024-08-25 03:00:00}|2              |
|GR         |{2024-08-25 17:40:00, 2024-08-25 18:00:00}|1              |
|BXL        |{2024-08-25 22:20:00, 2024-08-25 22:40:00}|1              |
|DVL        |{2024-08-25 23:00:00, 2024-08-25 23:20:00}|2              |
|VRL        |{2024-08-25 17:00:00, 2024-08-25 17:20:00}|2              |
|KNW        |{2024-08-25 01:00:00, 2024-08-25 01:20:00}|3              |
|AUBR       |{2024-08-25 08:00:00, 2024-08-25 08:20:00}|1              |
|SBM        |{2024-08-25 05:20:00, 2024-08-25 05:40:00}|1              |
|KUN        |{2024-08-25 03:20:00, 2024-08-25 03:40:00}|1              |
|JM         |{2024-08-25 10:20:00, 2024-08-25 10:40:00}|1              |
|FGR        |{2024-08-25 17:00:00, 2024-08-25 17:20:00}|1              |
|FL         |{2024-08-25 17:20:00, 2024-08-25 17:40:00}|1              |
|KVZ        |{2024-08-25 18:40:00, 2024-08-25 19:00:00}|1              |
|BSL        |{2024-08-25 21:20:00, 2024-08-25 21:40:00}|1              |
|KBY        |{2024-08-25 07:00:00, 2024-08-25 07:20:00}|1              |
|CAN        |{2024-08-25 23:40:00, 2024-08-26 00:00:00}|1              |
|TDD        |{2024-08-25 20:00:00, 2024-08-25 20:20:00}|1              |
|HJLI       |{2024-08-25 05:40:00, 2024-08-25 06:00:00}|1              |
|MZP        |{2024-08-25 23:00:00, 2024-08-25 23:20:00}|1              |
|MULK       |{2024-08-25 06:00:00, 2024-08-25 06:20:00}|1              |
+-----------+------------------------------------------+---------------+
only showing top 20 rows
```



```
[Consumer clientId=consumer-spark-kafka-source-65227384-1ea9-4c5b-9cd8-9f2ac77a159b-36
5556117-driver-0-1, groupId=spark-kafka-source-65227384-1ea9-4c5b-9cd8-9f2ac77a159b-36
5556117-driver-0] Resetting offset for partition test-4-0 to offset 182147.
```

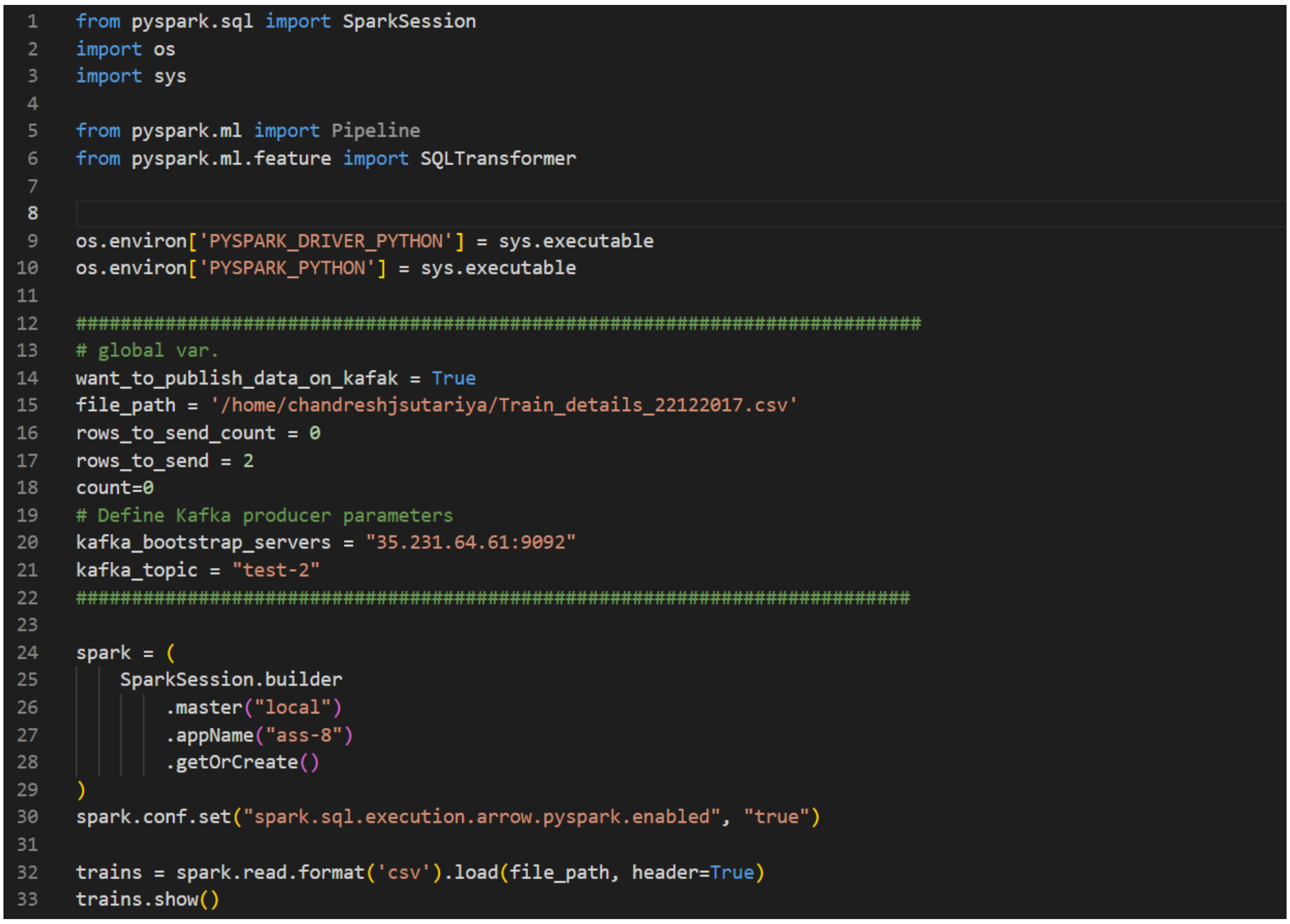
For the last batch, the offset will be set to total number of records that were there in the kafka-topic.

So, here it-is 1,82,147. And while I was preprocessing the data, the data after the preprocessing were also 182147. So that's one check!

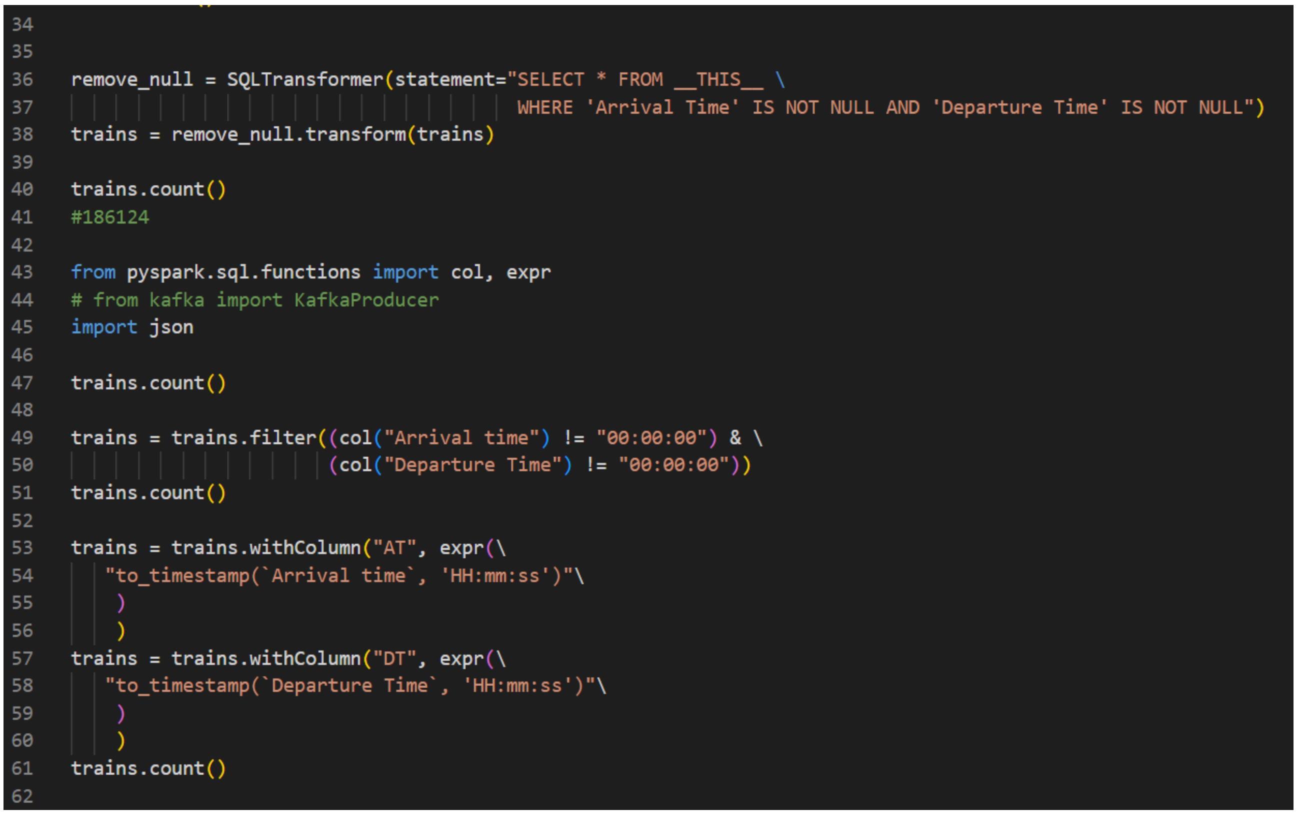


```
df.count()

182147
```

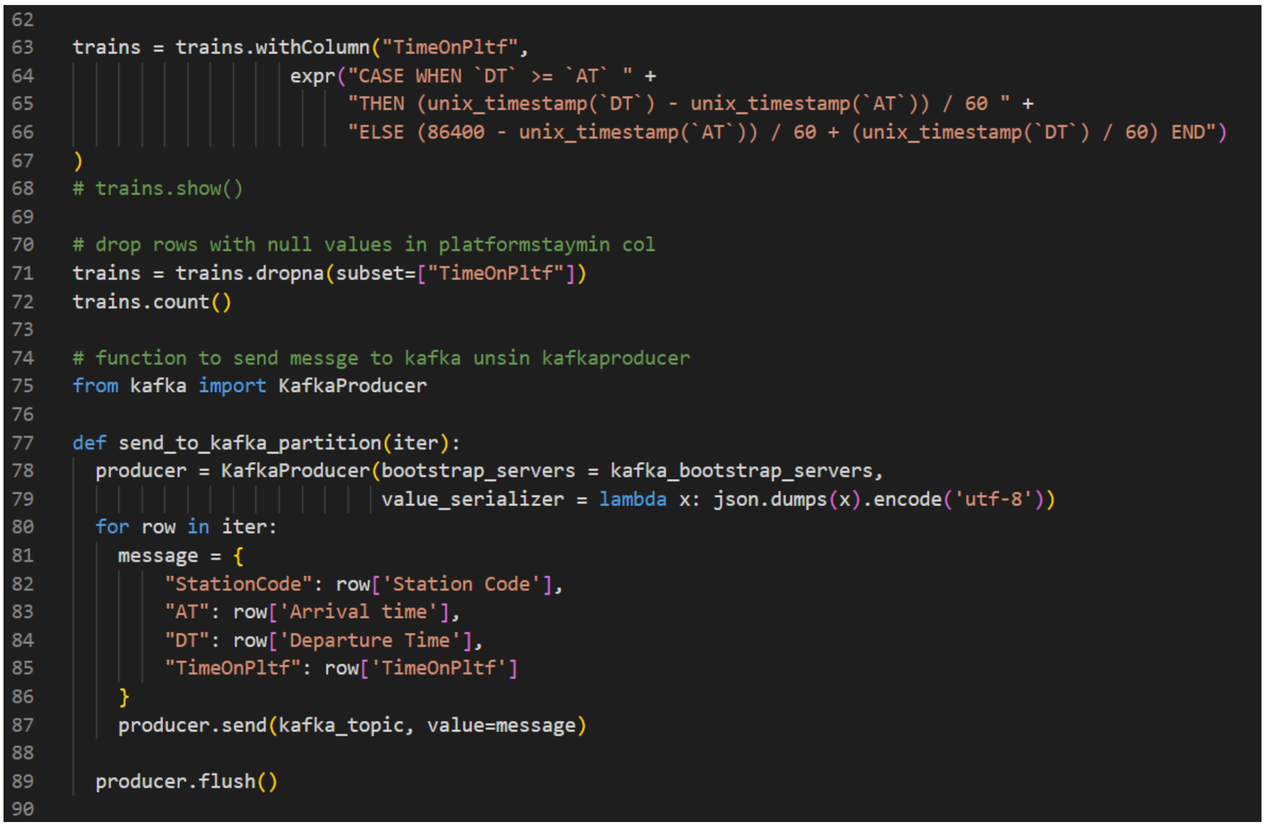
Report | Code | prdcr.py



```python
from pyspark.sql import SparkSession
import os
import sys

from pyspark.ml import Pipeline
from pyspark.ml.feature import SQLTransformer


os.environ['PYSPARK_DRIVER_PYTHON'] = sys.executable
os.environ['PYSPARK_PYTHON'] = sys.executable

##########################################################################
# global var.
want_to_publish_data_on_kafak = True
file_path = '/home/chandreshjsutariya/Train_details_22122017.csv'
rows_to_send_count = 0
rows_to_send = 2
count=0
# Define Kafka producer parameters
kafka_bootstrap_servers = "35.231.64.61:9092"
kafka_topic = "test-2"
#########################################################################

spark = (
    SparkSession.builder
        .master("local")
        .appName("ass-8")
        .getOrCreate()
)
spark.conf.set("spark.sql.execution.arrow.pyspark.enabled", "true")

trains = spark.read.format('csv').load(file_path, header=True)
trains.show()
```

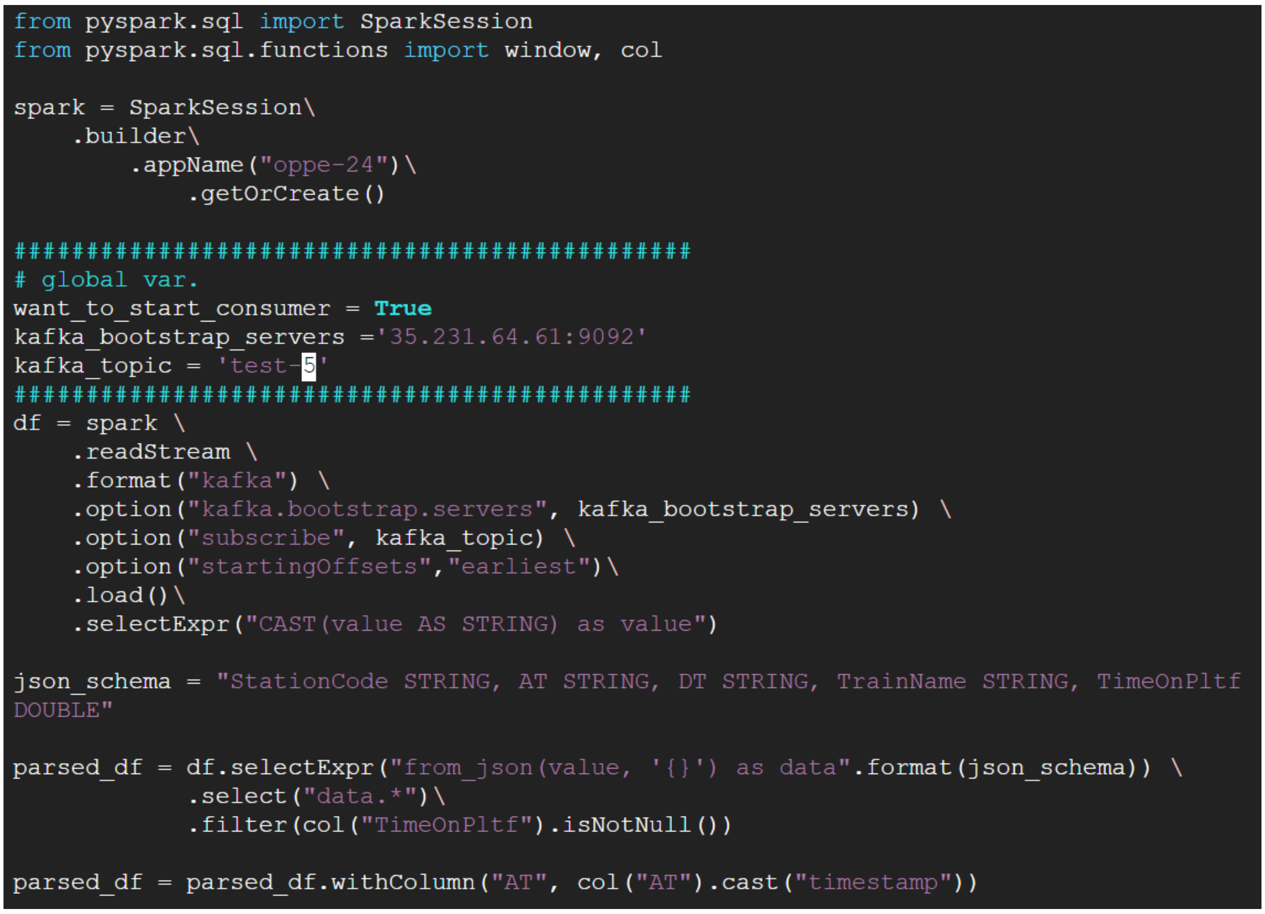


```python


remove_null = SQLTransformer(statement="SELECT * FROM __THIS__ \
                                        WHERE 'Arrival Time' IS NOT NULL AND 'Departure Time' IS NOT NULL")
trains = remove_null.transform(trains)

trains.count()
#186124

from pyspark.sql.functions import col, expr
# from kafka import KafkaProducer
import json

trains.count()

trains = trains.filter((col("Arrival time") != "00:00:00") & \
                        (col("Departure Time") != "00:00:00"))
trains.count()

trains = trains.withColumn("AT", expr(\
    "to_timestamp(`Arrival time`, 'HH:mm:ss')"\
     )
     )
trains = trains.withColumn("DT", expr(\
    "to_timestamp(`Departure Time`, 'HH:mm:ss')"\
     )
     )
trains.count()

```

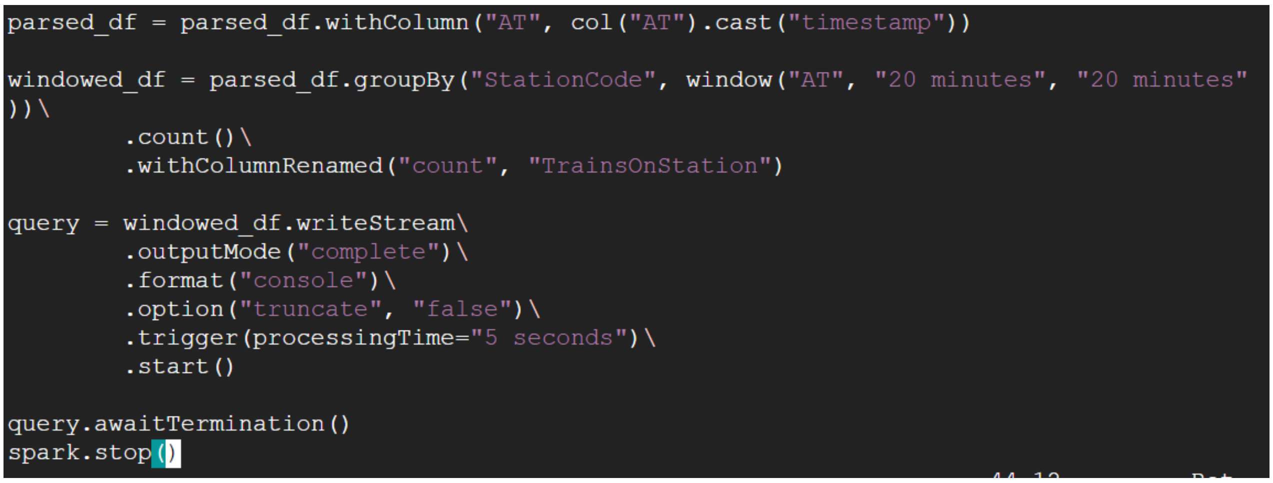


```python

trains = trains.withColumn("TimeOnPltf",
                   expr("CASE WHEN `DT` >= `AT` " +
                        "THEN (unix_timestamp(`DT`) - unix_timestamp(`AT`)) / 60 " +
                        "ELSE (86400 - unix_timestamp(`AT`)) / 60 + (unix_timestamp(`DT`) / 60) END")
)
# trains.show()

# drop rows with null values in platformstaymin col
trains = trains.dropna(subset=["TimeOnPltf"])
trains.count()

# function to send messge to kafka unsin kafkaproducer
from kafka import KafkaProducer

def send_to_kafka_partition(iter):
  producer = KafkaProducer(bootstrap_servers = kafka_bootstrap_servers,
                           value_serializer = lambda x: json.dumps(x).encode('utf-8'))
  for row in iter:
    message = {
        "StationCode": row['Station Code'],
        "AT": row['Arrival time'],
        "DT": row['Departure Time'],
        "TimeOnPltf": row['TimeOnPltf']
    }
    producer.send(kafka_topic, value=message)

  producer.flush()

```

Report | Code | consmr.py



```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import window, col

spark = SparkSession\
    .builder\
        .appName("oppe-24")\
            .getOrCreate()

###############################################
# global var.
want_to_start_consumer = True
kafka_bootstrap_servers ='35.231.64.61:9092'
kafka_topic = 'test-5'
###############################################
df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", kafka_bootstrap_servers) \
    .option("subscribe", kafka_topic) \
    .option("startingOffsets","earliest")\
    .load()\
    .selectExpr("CAST(value AS STRING) as value")

json_schema = "StationCode STRING, AT STRING, DT STRING, TrainName STRING, TimeOnPltf
DOUBLE"

parsed_df = df.selectExpr("from_json(value, '{}') as data".format(json_schema)) \
            .select("data.*")\
            .filter(col("TimeOnPltf").isNotNull())

parsed_df = parsed_df.withColumn("AT", col("AT").cast("timestamp"))
```



```python
parsed_df = parsed_df.withColumn("AT", col("AT").cast("timestamp"))

windowed_df = parsed_df.groupBy("StationCode", window("AT", "20 minutes", "20 minutes"
))\
        .count()\
        .withColumnRenamed("count", "TrainsOnStation")

query = windowed_df.writeStream\
        .outputMode("complete")\
        .format("console")\
        .option("truncate", "false")\
        .trigger(processingTime="5 seconds")\
        .start()

query.awaitTermination()
spark.stop()
```