Big Data 2

PROJEKAT 1

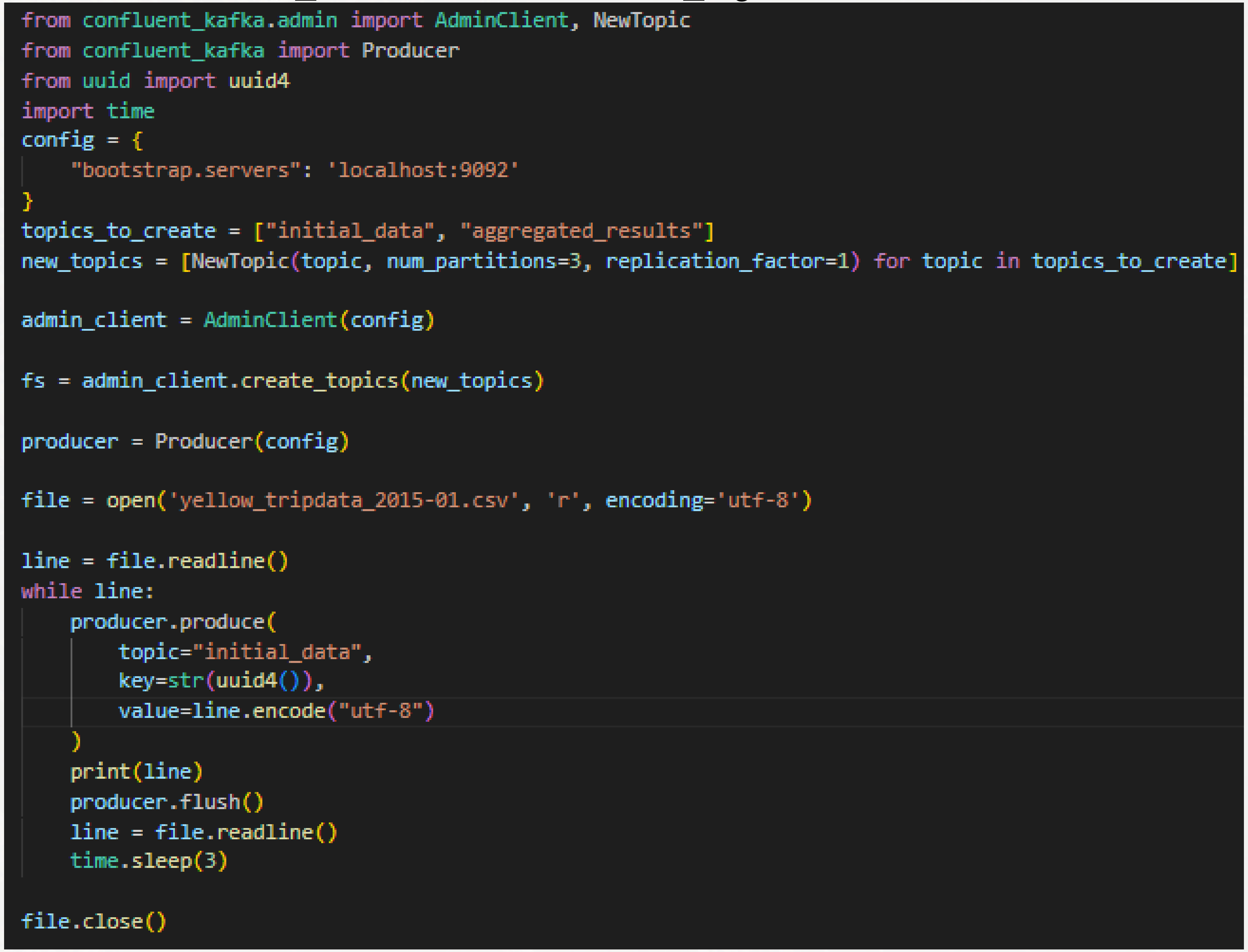
Veljko Veljović 1937

ELEKTRONSKI FAKULTET NIŠ

Dataset - NYC Yellow Taxi Trip Data

1. Skup podataka o vožnjama žutih taksi vozila u New York-u
2. Podaci su prikupljeni od januara 2015. do marta 2016. godine.
3. Dataset je veličine 1.99 GB
4. Nije bilo potrebe za sređivanjem dataseta.

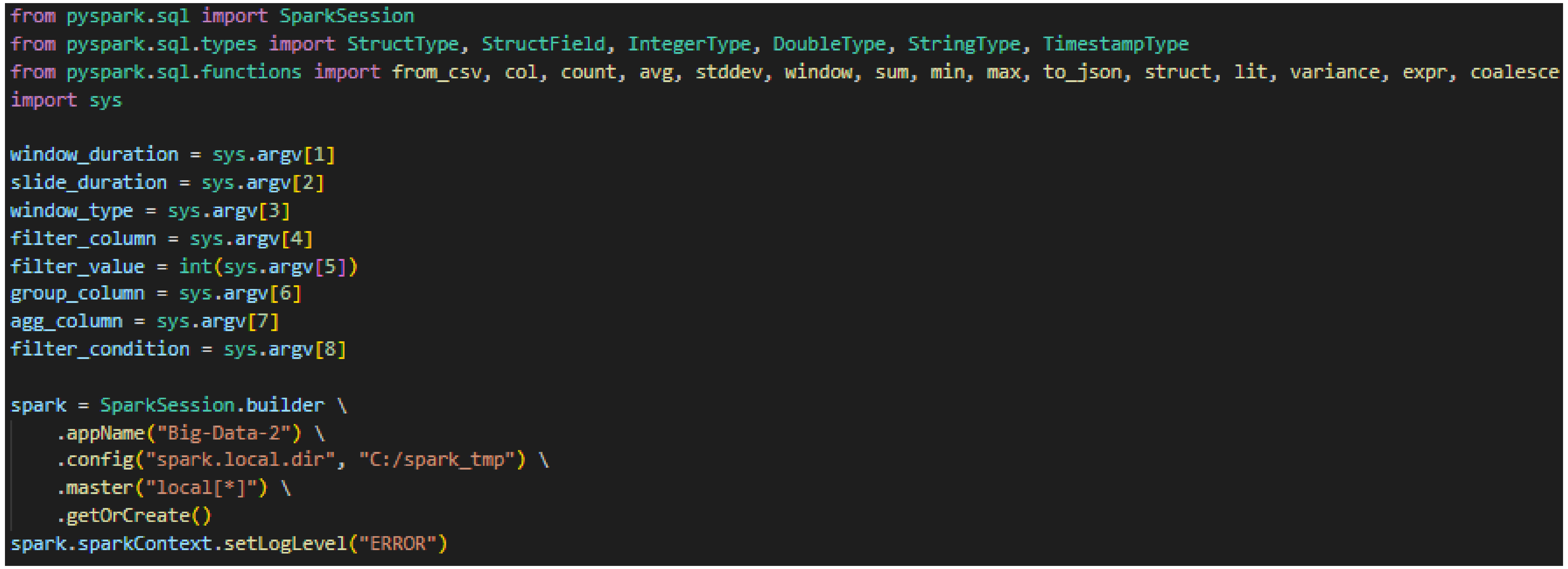
producer.py



```python
from confluent_kafka.admin import AdminClient, NewTopic
from confluent_kafka import Producer
from uuid import uuid4
import time
config = {
    "bootstrap.servers": 'localhost:9092'
}
topics_to_create = ["initial_data", "aggregated_results"]
new_topics = [NewTopic(topic, num_partitions=3, replication_factor=1) for topic in topics_to_create]

admin_client = AdminClient(config)

fs = admin_client.create_topics(new_topics)

producer = Producer(config)

file = open('yellow_tripdata_2015-01.csv', 'r', encoding='utf-8')

line = file.readline()
while line:
    producer.produce(
        topic="initial_data",
        key=str(uuid4()),
        value=line.encode("utf-8")
    )
    print(line)
    producer.flush()
    line = file.readline()
    time.sleep(3)

file.close()
```

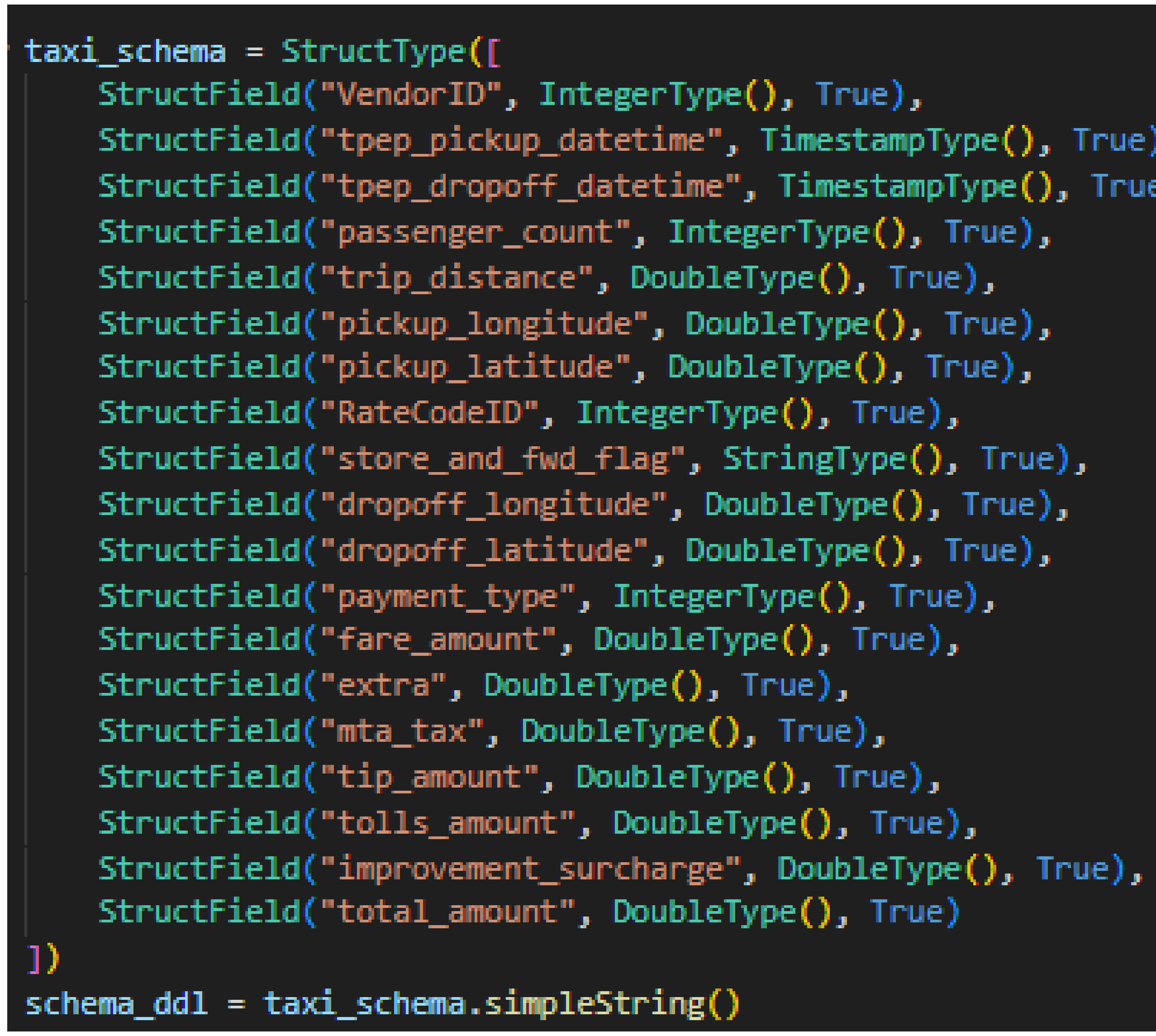
processor.py

Importi parsiranje argumenata i otvaranje spark sesije



```python
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, IntegerType, DoubleType, StringType, TimestampType
from pyspark.sql.functions import from_csv, col, count, avg, stddev, window, sum, min, max, to_json, struct, lit, variance, expr, coalesce
import sys

window_duration = sys.argv[1]
slide_duration = sys.argv[2]
window_type = sys.argv[3]
filter_column = sys.argv[4]
filter_value = int(sys.argv[5])
group_column = sys.argv[6]
agg_column = sys.argv[7]
filter_condition = sys.argv[8]

spark = SparkSession.builder \
    .appName("Big-Data-2") \
    .config("spark.local.dir", "C:/spark_tmp") \
    .master("local[*]") \
    .getOrCreate()
spark.sparkContext.setLogLevel("ERROR")
```

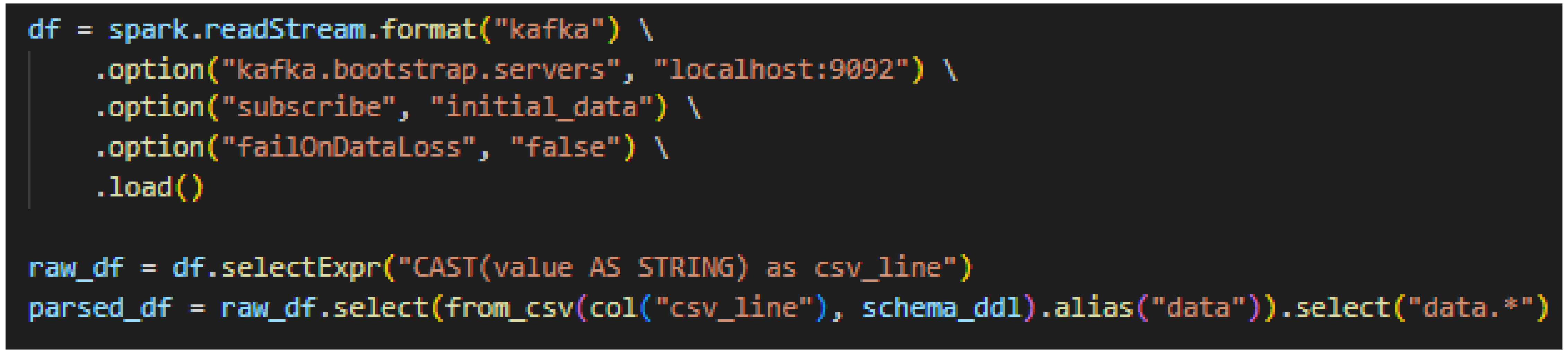
šema podataka



```python
taxi_schema = StructType([
    StructField("VendorID", IntegerType(), True),
    StructField("tpep_pickup_datetime", TimestampType(), True)
    StructField("tpep_dropoff_datetime", TimestampType(), True
    StructField("passenger_count", IntegerType(), True),
    StructField("trip_distance", DoubleType(), True),
    StructField("pickup_longitude", DoubleType(), True),
    StructField("pickup_latitude", DoubleType(), True),
    StructField("RateCodeID", IntegerType(), True),
    StructField("store_and_fwd_flag", StringType(), True),
    StructField("dropoff_longitude", DoubleType(), True),
    StructField("dropoff_latitude", DoubleType(), True),
    StructField("payment_type", IntegerType(), True),
    StructField("fare_amount", DoubleType(), True),
    StructField("extra", DoubleType(), True),
    StructField("mta_tax", DoubleType(), True),
    StructField("tip_amount", DoubleType(), True),
    StructField("tolls_amount", DoubleType(), True),
    StructField("improvement_surcharge", DoubleType(), True),
    StructField("total_amount", DoubleType(), True)
])
schema_ddl = taxi_schema.simpleString()
```

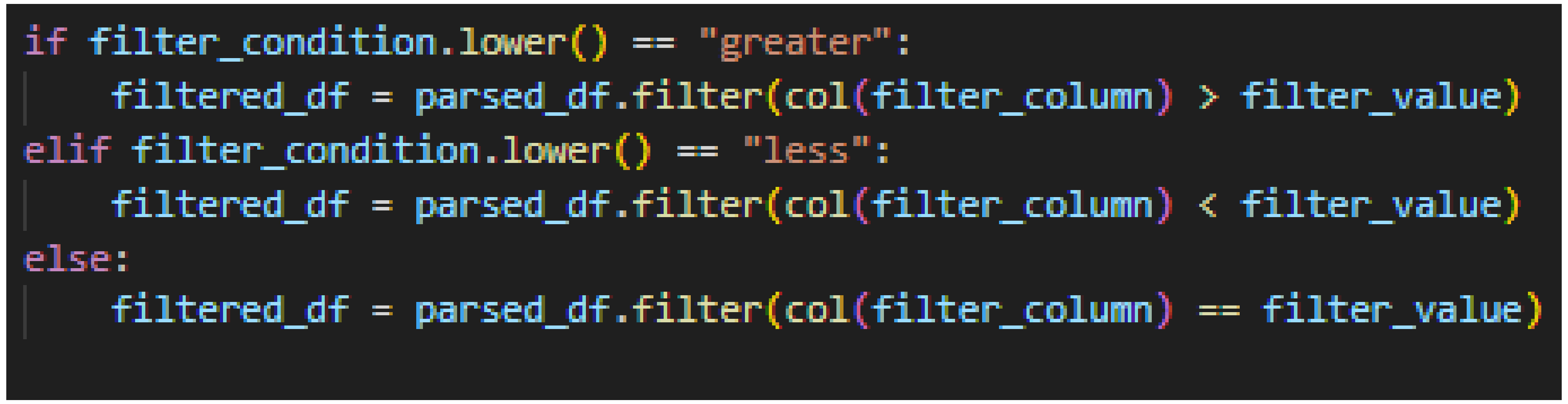
processor.py

kreiranje dataframe-a



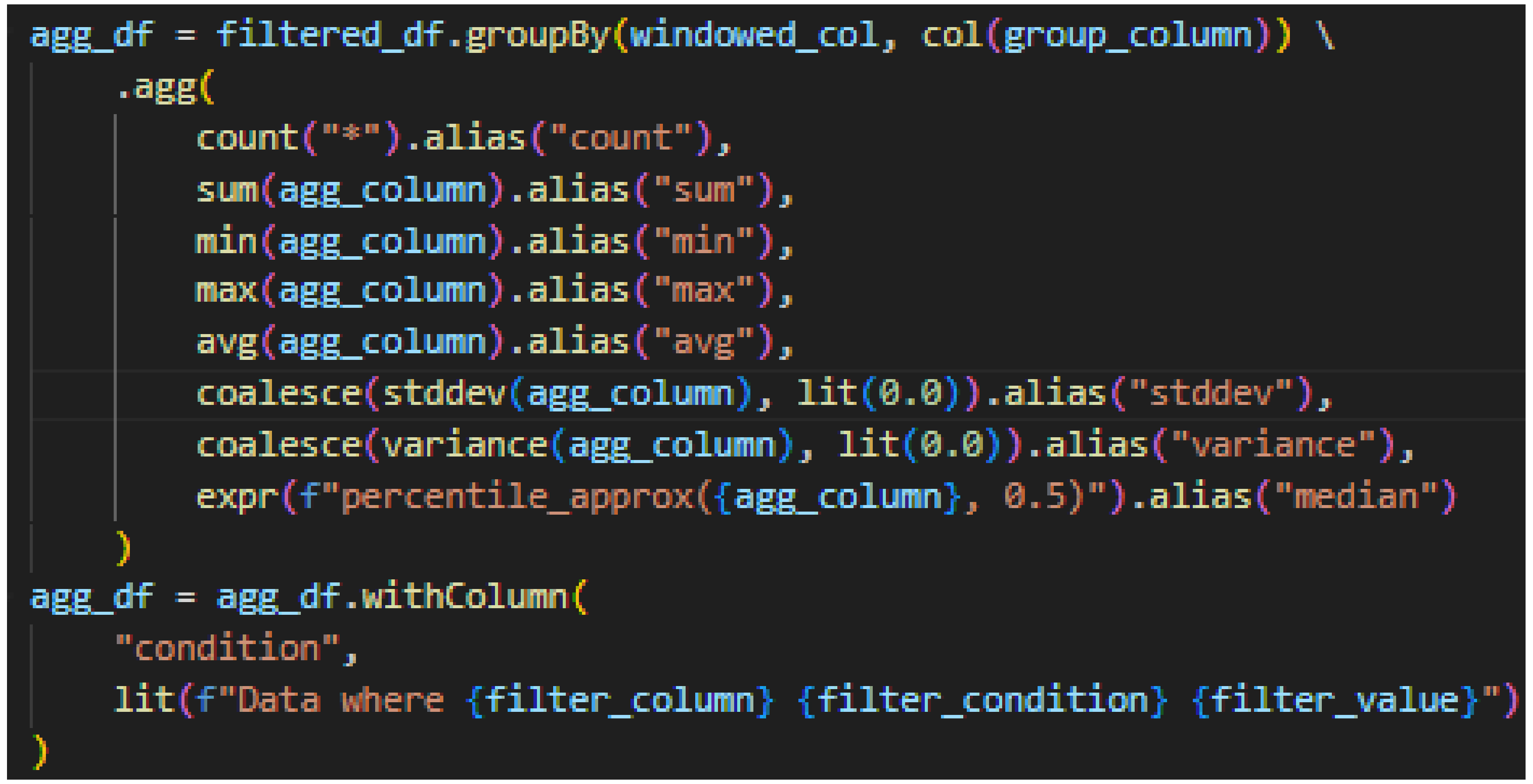
```python
df = spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "initial_data") \
    .option("failOnDataLoss", "false") \
    .load()

raw_df = df.selectExpr("CAST(value AS STRING) as csv_line")
parsed_df = raw_df.select(from_csv(col("csv_line"), schema_ddl).alias("data")).select("data.*")
```

definisanje uslova



```python
if filter_condition.lower() == "greater":
    filtered_df = parsed_df.filter(col(filter_column) > filter_value)
elif filter_condition.lower() == "less":
    filtered_df = parsed_df.filter(col(filter_column) < filter_value)
else:
    filtered_df = parsed_df.filter(col(filter_column) == filter_value)
```

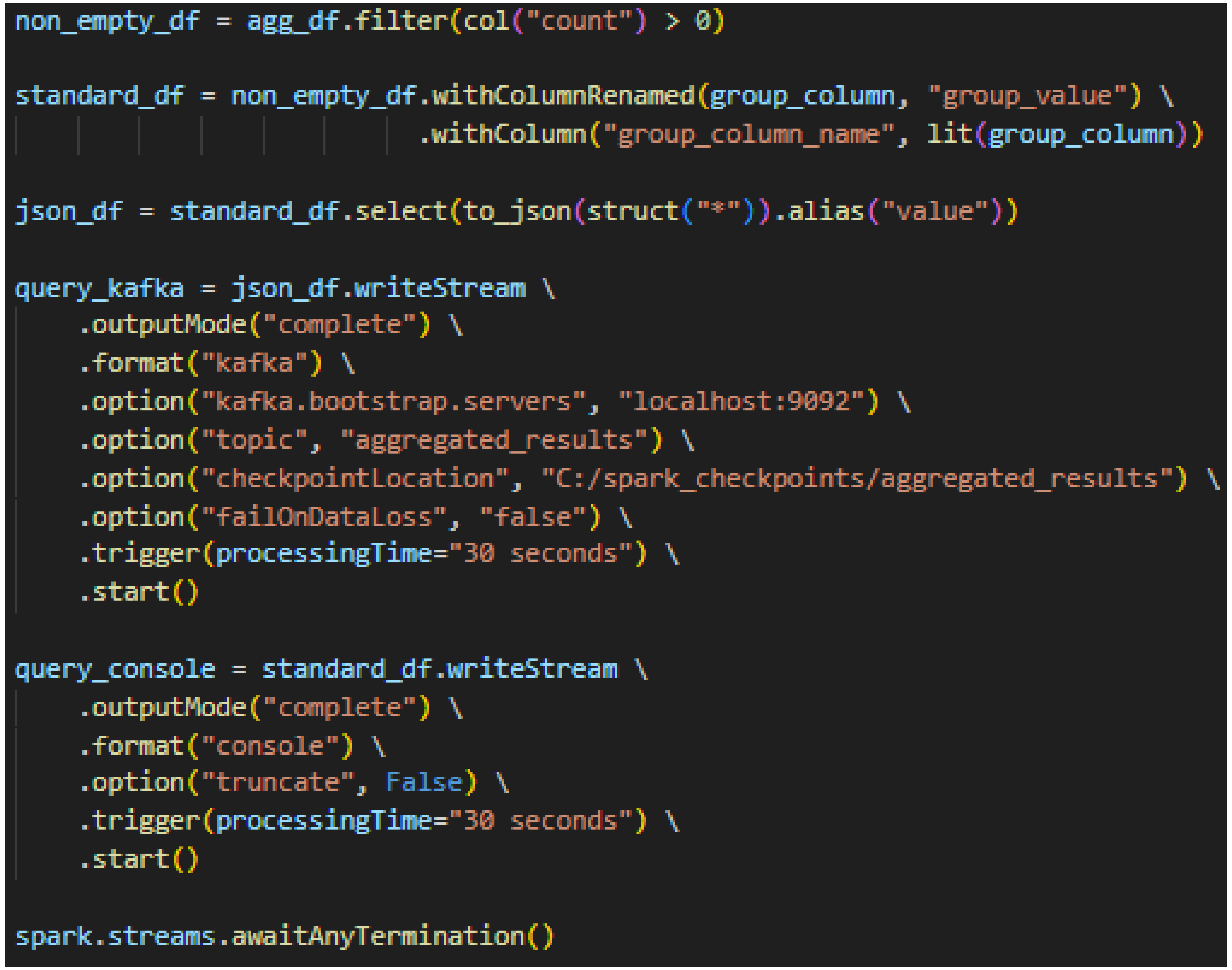
proračun



```python
agg_df = filtered_df.groupBy(windowed_col, col(group_column)) \
    .agg(
        count("*").alias("count"),
        sum(agg_column).alias("sum"),
        min(agg_column).alias("min"),
        max(agg_column).alias("max"),
        avg(agg_column).alias("avg"),
        coalesce(stddev(agg_column), lit(0.0)).alias("stddev"),
        coalesce(variance(agg_column), lit(0.0)).alias("variance"),
        expr(f"percentile_approx({agg_column}, 0.5)").alias("median")
    )
agg_df = agg_df.withColumn(
    "condition",
    lit(f"Data where {filter_column} {filter_condition} {filter_value}")
)
```

processor.py

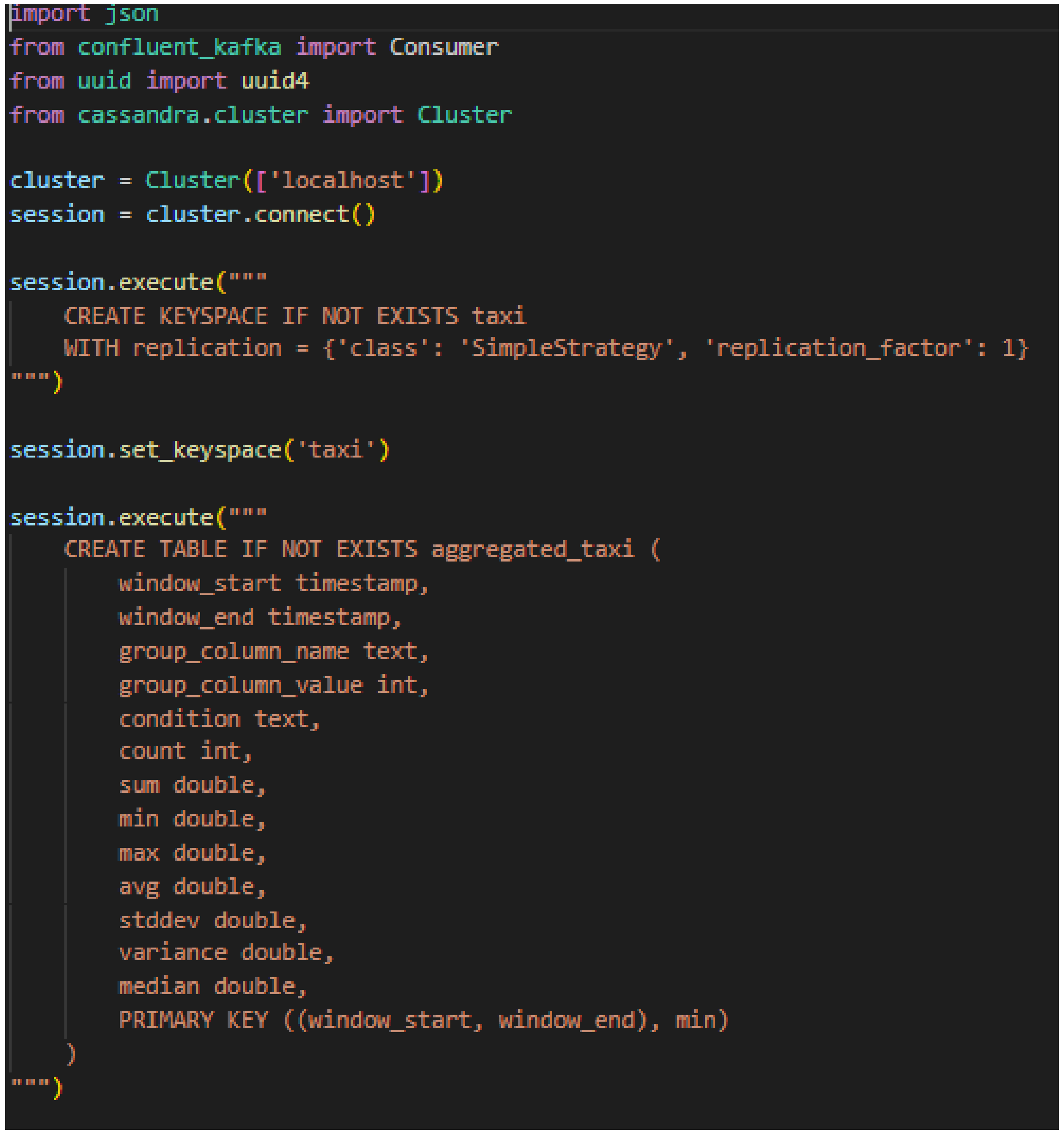
slanje podataka na novi topic i ispis na konzolu



```
non_empty_df = agg_df.filter(col("count") > 0)

standard_df = non_empty_df.withColumnRenamed(group_column, "group_value") \
                          .withColumn("group_column_name", lit(group_column))

json_df = standard_df.select(to_json(struct("*")).alias("value"))

query_kafka = json_df.writeStream \
    .outputMode("complete") \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("topic", "aggregated_results") \
    .option("checkpointLocation", "C:/spark_checkpoints/aggregated_results") \
    .option("failOnDataLoss", "false") \
    .trigger(processingTime="30 seconds") \
    .start()

query_console = standard_df.writeStream \
    .outputMode("complete") \
    .format("console") \
    .option("truncate", False) \
    .trigger(processingTime="30 seconds") \
    .start()

spark.streams.awaitAnyTermination()
```

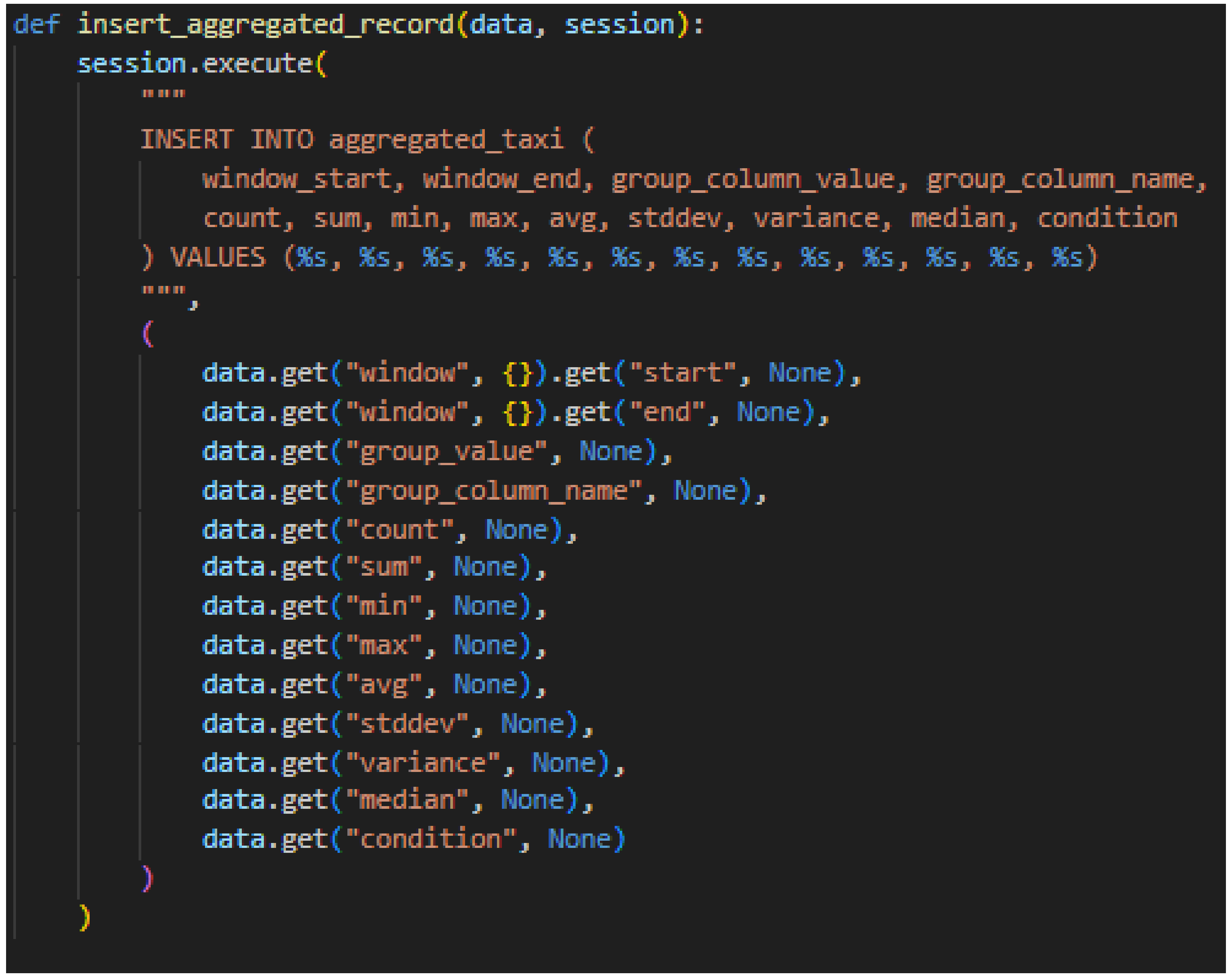
data-storage-service.py

kreiranje keypsace i tabele



```python
import json
from confluent_kafka import Consumer
from uuid import uuid4
from cassandra.cluster import Cluster

cluster = Cluster(['localhost'])
session = cluster.connect()

session.execute("""
    CREATE KEYSPACE IF NOT EXISTS taxi
    WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1}
""")

session.set_keyspace('taxi')

session.execute("""
    CREATE TABLE IF NOT EXISTS aggregated_taxi (
        window_start timestamp,
        window_end timestamp,
        group_column_name text,
        group_column_value int,
        condition text,
        count int,
        sum double,
        min double,
        max double,
        avg double,
        stddev double,
        variance double,
        median double,
        PRIMARY KEY ((window_start, window_end), min)
    )
""")
```

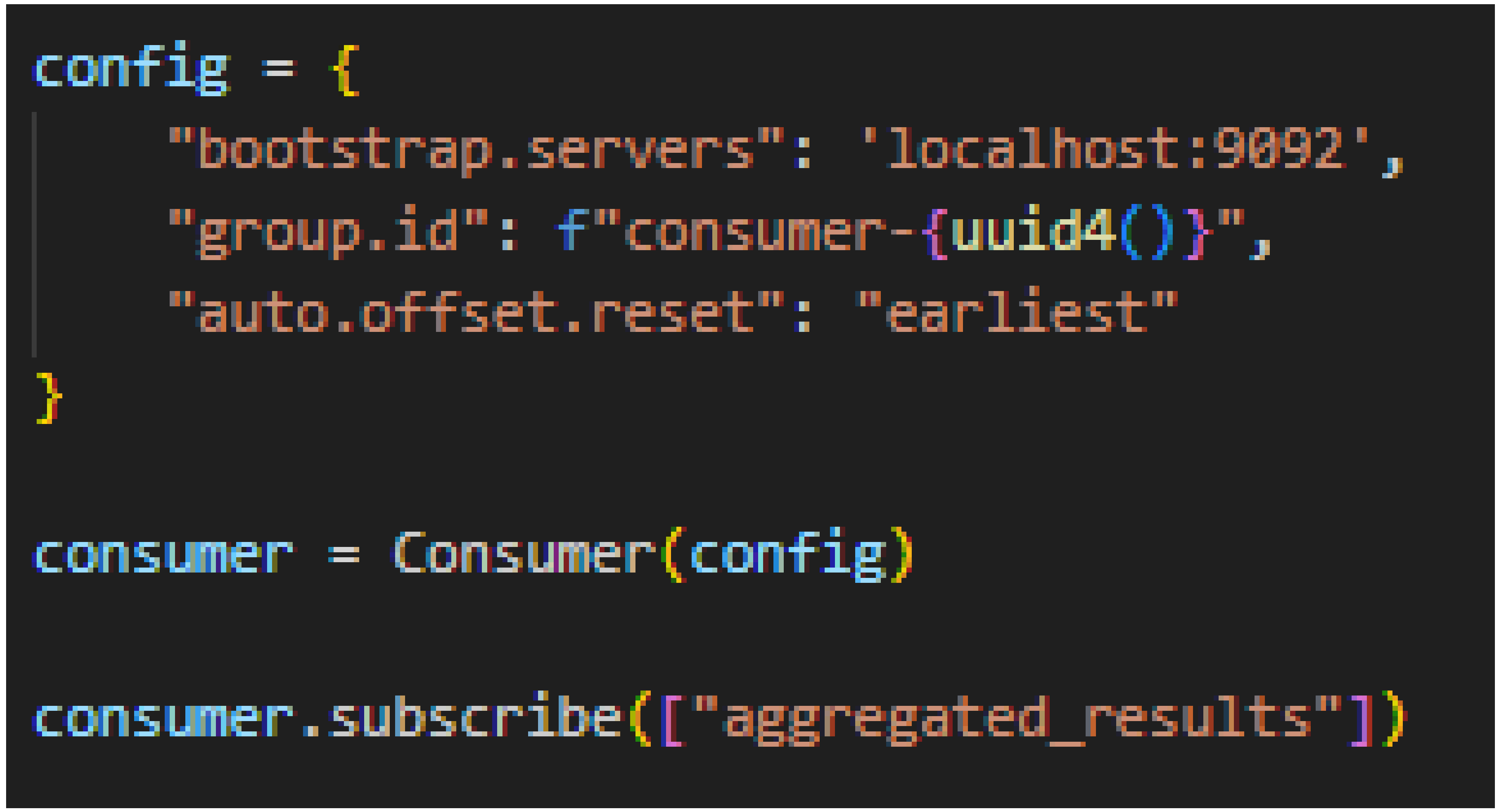
upis podataka u odgovarajuće tabele



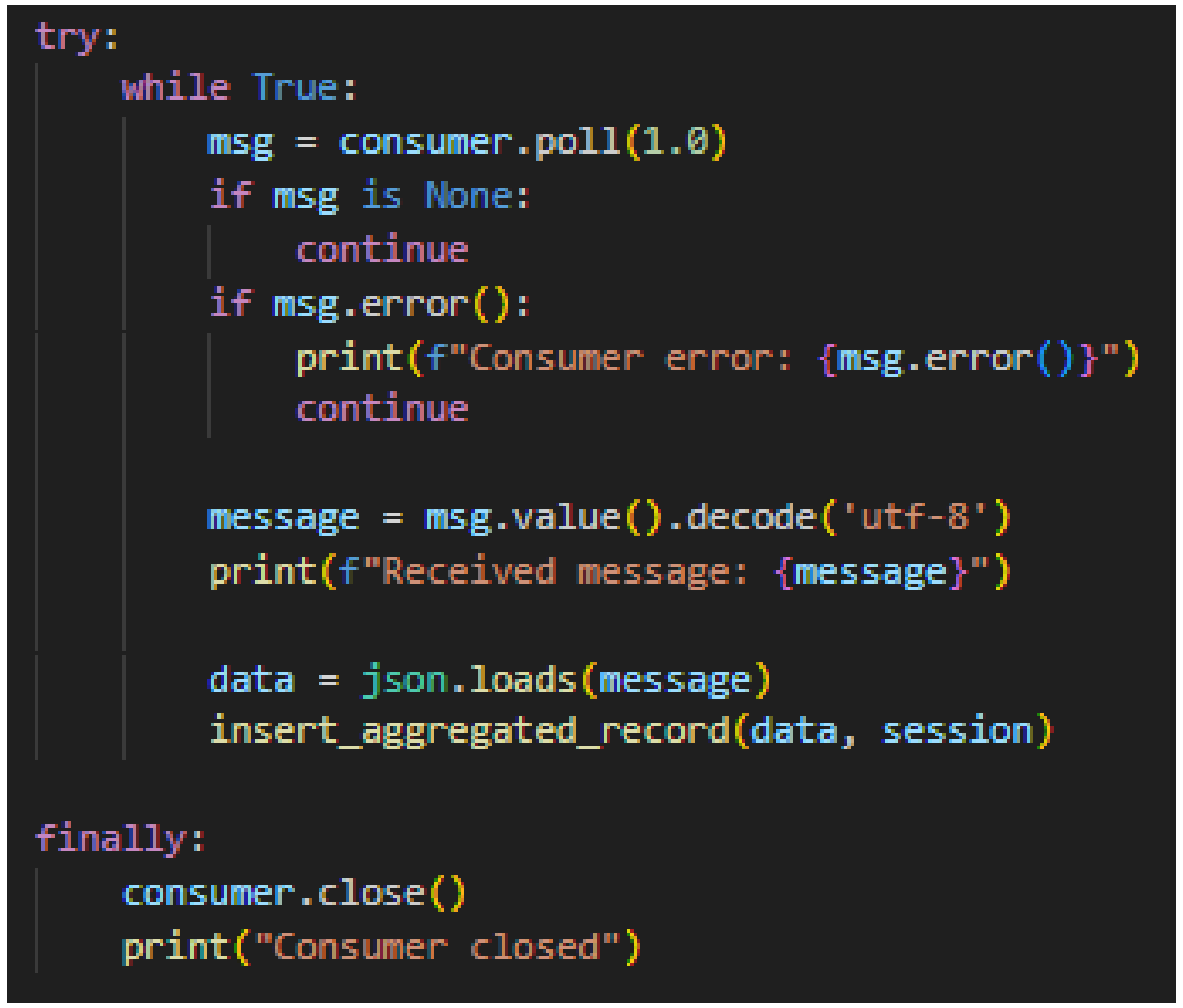
```python
def insert_aggregated_record(data, session):
    session.execute(
        """
        INSERT INTO aggregated_taxi (
            window_start, window_end, group_column_value, group_column_name,
            count, sum, min, max, avg, stddev, variance, median, condition
        ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
        """,
        (
            data.get("window", {}).get("start", None),
            data.get("window", {}).get("end", None),
            data.get("group_value", None),
            data.get("group_column_name", None),
            data.get("count", None),
            data.get("sum", None),
            data.get("min", None),
            data.get("max", None),
            data.get("avg", None),
            data.get("stddev", None),
            data.get("variance", None),
            data.get("median", None),
            data.get("condition", None)
        )
    )
```

data-storage-service.py

Subscribe na topike i insertovanje podataka u bazu



```python
config = {
    "bootstrap.servers": 'localhost:9092',
    "group.id": f"consumer-{uuid4()}",
    "auto.offset.reset": "earliest"
}

consumer = Consumer(config)

consumer.subscribe(["aggregated_results"])
```

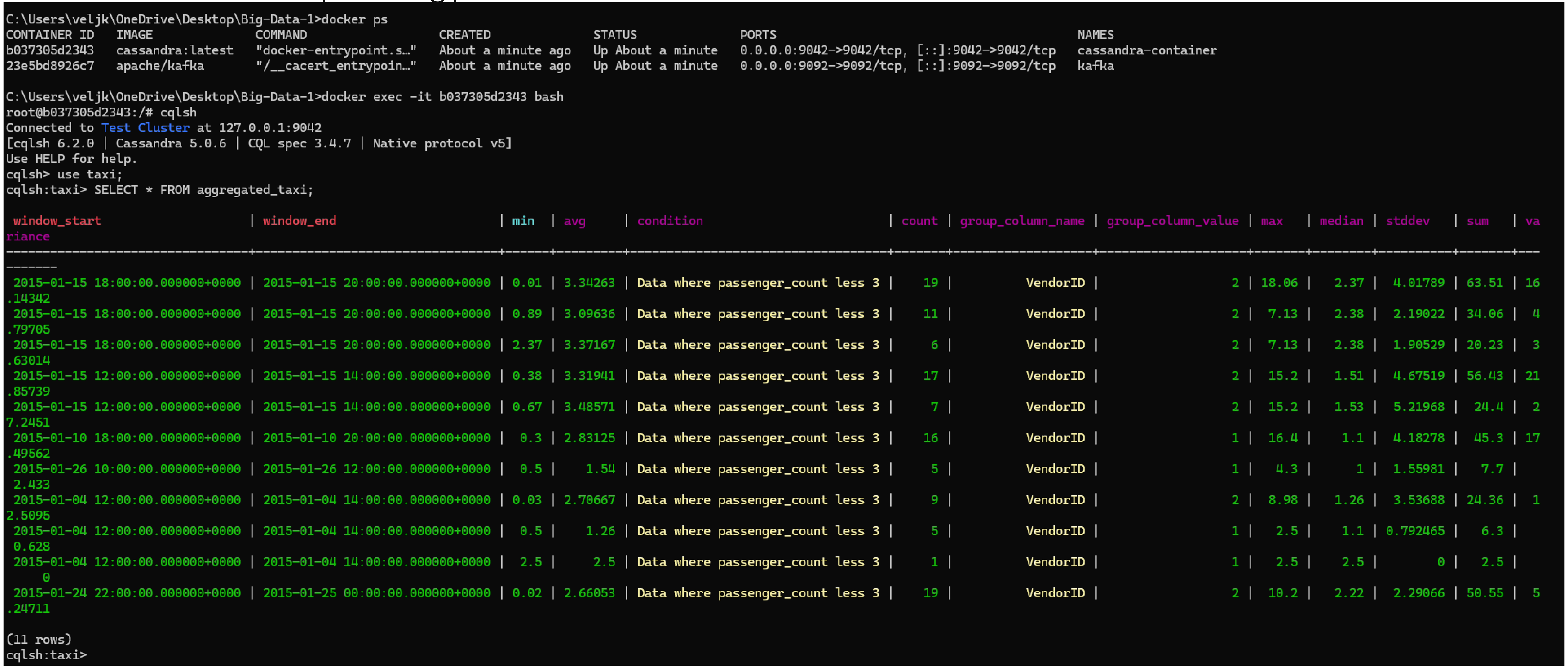


```python
try:
    while True:
        msg = consumer.poll(1.0)
        if msg is None:
            continue
        if msg.error():
            print(f"Consumer error: {msg.error()}")
            continue

        message = msg.value().decode('utf-8')
        print(f"Received message: {message}")

        data = json.loads(message)
        insert_aggregated_record(data, session)

finally:
    consumer.close()
    print("Consumer closed")
```

Testiranje

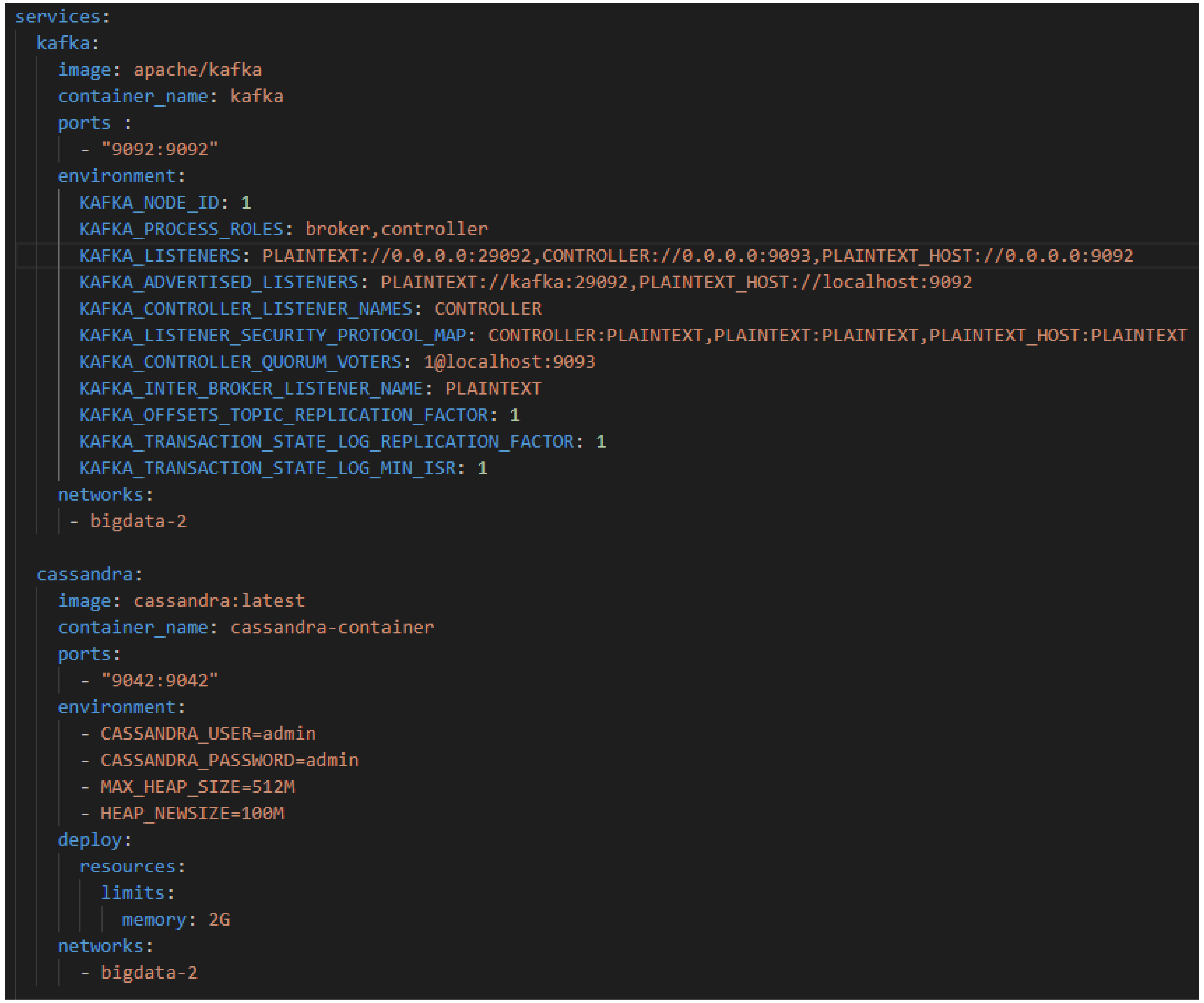
Filtrira taksi-vožnje sa manje od tri putnika i prebrojava ih, grupiše podatke u fiksne vremenske prozore od dva sata i izračunava statističke mere nad dužinom pređenog puta.

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.1 processor.py "120 minutes" "60 minutes" sliding passenger_count 3 VendorID trip_distance less

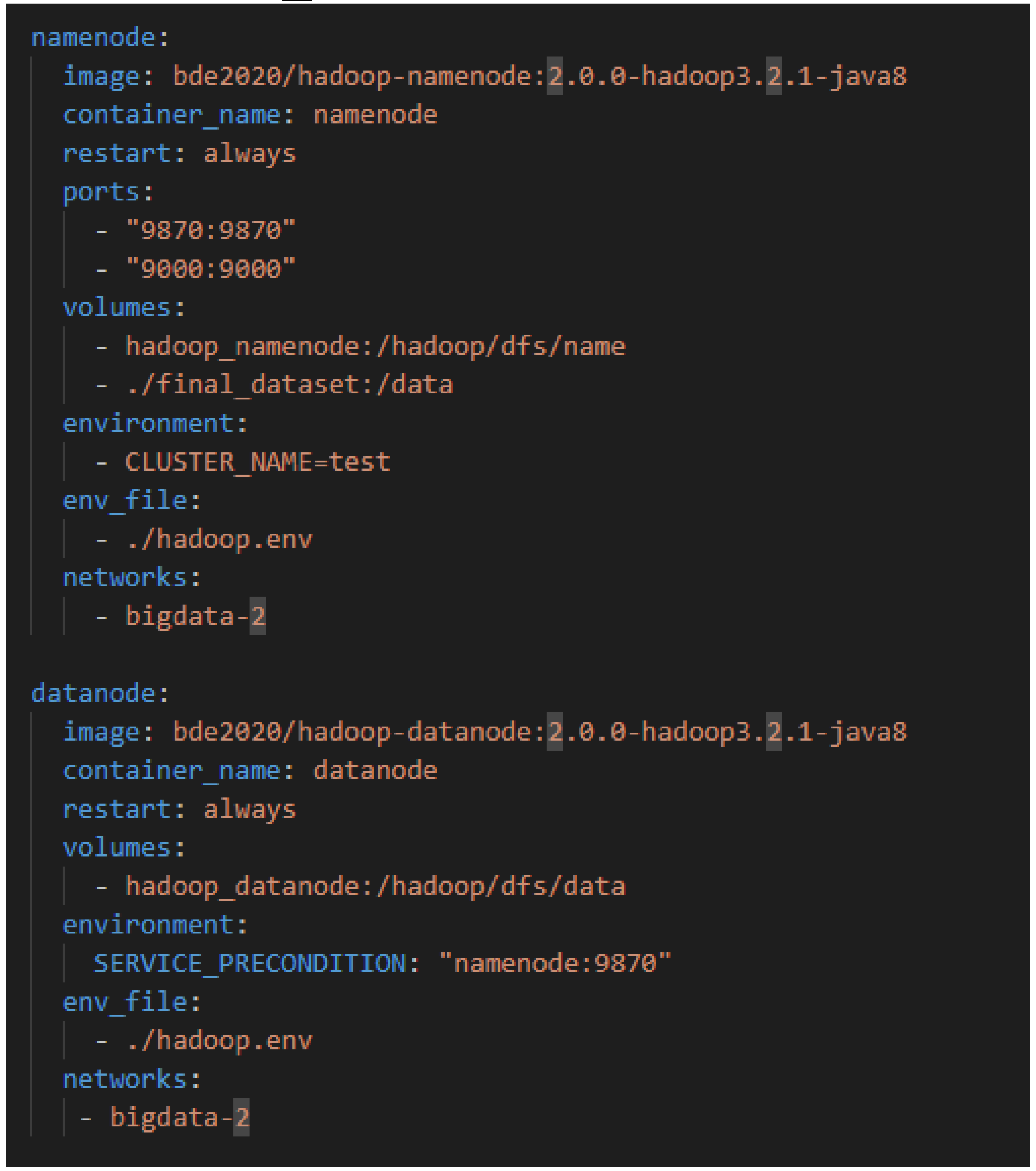


```
C:\Users\veljk\OneDrive\Desktop\Big-Data-1>docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED              STATUS              PORTS                                          NAMES
b037305d2343   cassandra:latest   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:9042->9042/tcp, [::]:9042->9042/tcp   cassandra-container
23e5bd8926c7   apache/kafka       "/__cacert_entrypoin…"   About a minute ago   Up About a minute   0.0.0.0:9092->9092/tcp, [::]:9092->9092/tcp   kafka

C:\Users\veljk\OneDrive\Desktop\Big-Data-1>docker exec -it b037305d2343 bash
root@b037305d2343:/# cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> use taxi;
cqlsh:taxi> SELECT * FROM aggregated_taxi;

 window_start                    | window_end                      | min  | avg     | condition                          | count | group_column_name | group_column_value | max   | median | stddev   | sum   | va
riance
---------------------------------+---------------------------------+------+---------+------------------------------------+-------+-------------------+--------------------+-------+--------+----------+-------+---
-------
 2015-01-15 18:00:00.000000+0000 | 2015-01-15 20:00:00.000000+0000 | 0.01 | 3.34263 | Data where passenger_count less 3 |    19 |          VendorID |                  2 | 18.06 |   2.37 |  4.01789 | 63.51 | 16
.14342
 2015-01-15 18:00:00.000000+0000 | 2015-01-15 20:00:00.000000+0000 | 0.89 | 3.09636 | Data where passenger_count less 3 |    11 |          VendorID |                  2 |  7.13 |   2.38 |  2.19022 | 34.06 |  4
.79705
 2015-01-15 18:00:00.000000+0000 | 2015-01-15 20:00:00.000000+0000 | 2.37 | 3.37167 | Data where passenger_count less 3 |     6 |          VendorID |                  2 |  7.13 |   2.38 |  1.90529 | 20.23 |  3
.63014
 2015-01-15 12:00:00.000000+0000 | 2015-01-15 14:00:00.000000+0000 | 0.38 | 3.31941 | Data where passenger_count less 3 |    17 |          VendorID |                  2 |  15.2 |   1.51 |  4.67519 | 56.43 | 21
.85739
 2015-01-15 12:00:00.000000+0000 | 2015-01-15 14:00:00.000000+0000 | 0.67 | 3.48571 | Data where passenger_count less 3 |     7 |          VendorID |                  2 |  15.2 |   1.53 |  5.21968 |  24.4 |  2
7.2451
 2015-01-10 18:00:00.000000+0000 | 2015-01-10 20:00:00.000000+0000 |  0.3 | 2.83125 | Data where passenger_count less 3 |    16 |          VendorID |                  1 |  16.4 |    1.1 |  4.18278 |  45.3 | 17
.49562
 2015-01-26 10:00:00.000000+0000 | 2015-01-26 12:00:00.000000+0000 |  0.5 |    1.54 | Data where passenger_count less 3 |     5 |          VendorID |                  1 |   4.3 |      1 |  1.55981 |   7.7 |
 2.433
 2015-01-04 12:00:00.000000+0000 | 2015-01-04 14:00:00.000000+0000 | 0.03 | 2.70667 | Data where passenger_count less 3 |     9 |          VendorID |                  2 |  8.98 |   1.26 |  3.53688 | 24.36 |  1
2.5095
 2015-01-04 12:00:00.000000+0000 | 2015-01-04 14:00:00.000000+0000 |  0.5 |    1.26 | Data where passenger_count less 3 |     5 |          VendorID |                  1 |   2.5 |    1.1 | 0.792465 |   6.3 |
 0.628
 2015-01-04 12:00:00.000000+0000 | 2015-01-04 14:00:00.000000+0000 |  2.5 |     2.5 | Data where passenger_count less 3 |     1 |          VendorID |                  1 |   2.5 |    2.5 |        0 |   2.5 |
     0
 2015-01-24 22:00:00.000000+0000 | 2015-01-25 00:00:00.000000+0000 | 0.02 | 2.66053 | Data where passenger_count less 3 |    19 |          VendorID |                  2 |  10.2 |   2.22 |  2.29066 | 50.55 |  5
.24711

(11 rows)
cqlsh:taxi>
```

Docker compose

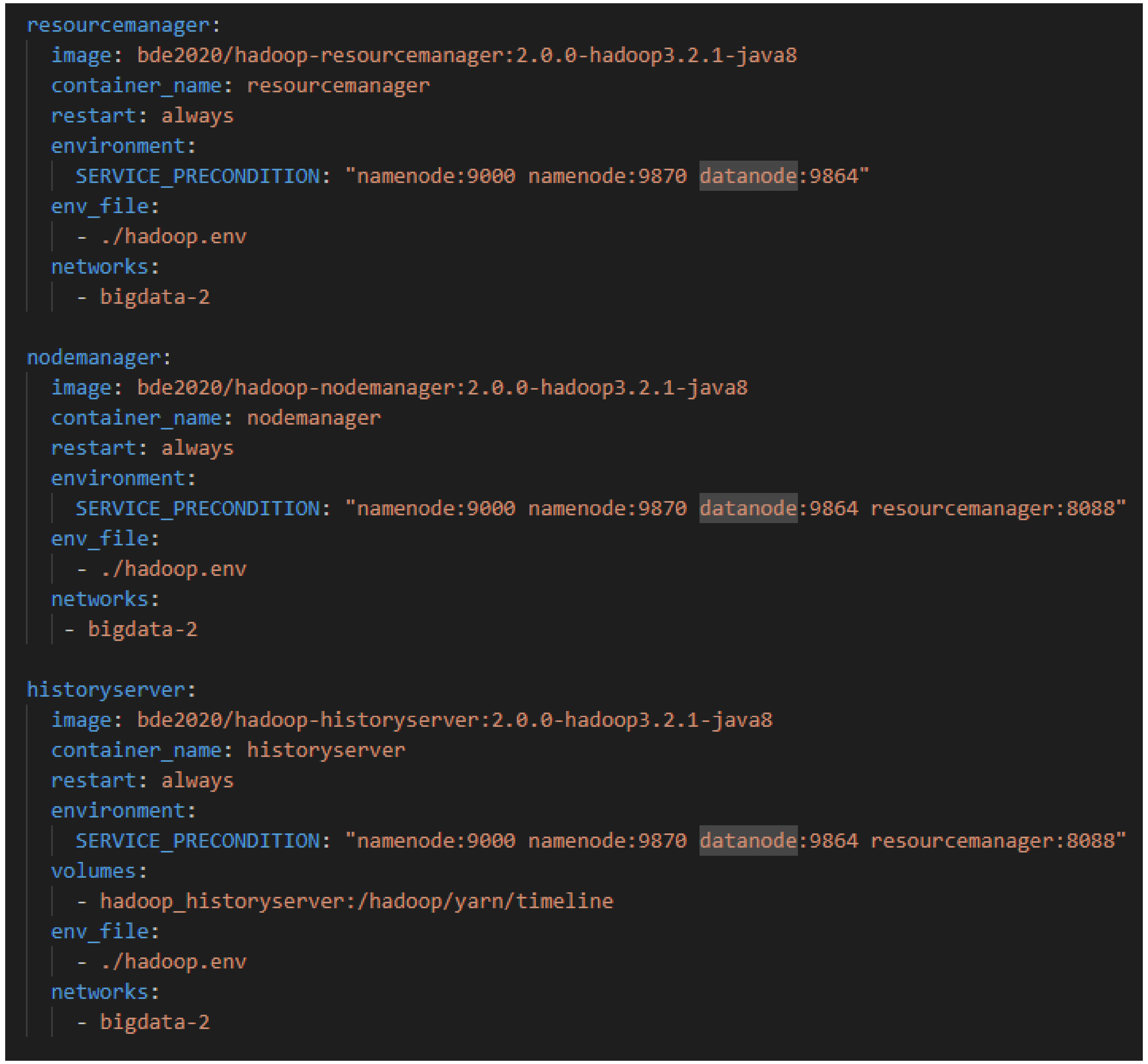


```yaml
services:
  kafka:
    image: apache/kafka
    container_name: kafka
    ports :
      - "9092:9092"
    environment:
      KAFKA_NODE_ID: 1
      KAFKA_PROCESS_ROLES: broker,controller
      KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:29092,CONTROLLER://0.0.0.0:9093,PLAINTEXT_HOST://0.0.0.0:9092
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:29092,PLAINTEXT_HOST://localhost:9092
      KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_CONTROLLER_QUORUM_VOTERS: 1@localhost:9093
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
    networks:
     - bigdata-2

  cassandra:
    image: cassandra:latest
    container_name: cassandra-container
    ports:
      - "9042:9042"
    environment:
      - CASSANDRA_USER=admin
      - CASSANDRA_PASSWORD=admin
      - MAX_HEAP_SIZE=512M
      - HEAP_NEWSIZE=100M
    deploy:
      resources:
        limits:
          memory: 2G
    networks:
      - bigdata-2
```

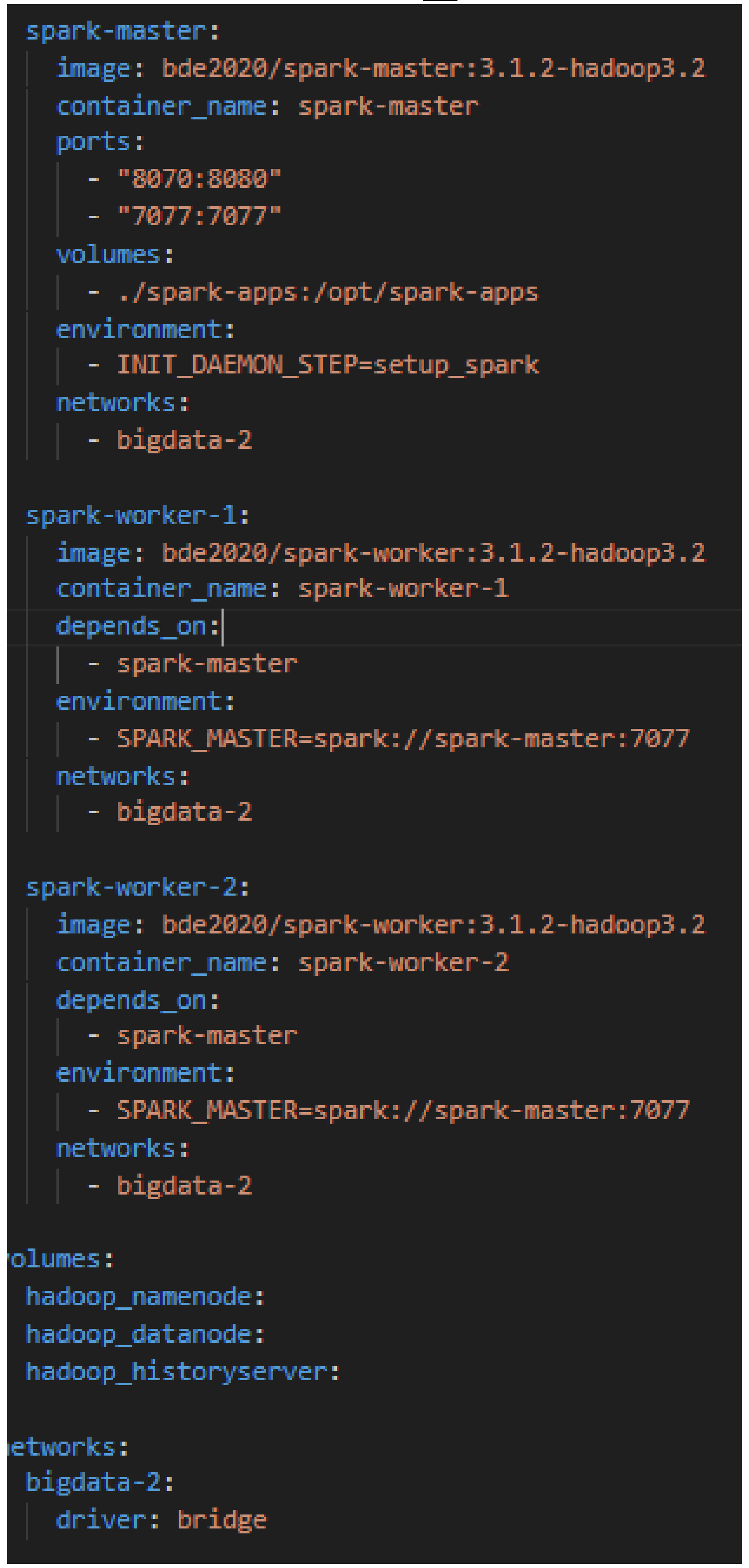


```yaml
namenode:
  image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
  container_name: namenode
  restart: always
  ports:
    - "9870:9870"
    - "9000:9000"
  volumes:
    - hadoop_namenode:/hadoop/dfs/name
    - ./final_dataset:/data
  environment:
    - CLUSTER_NAME=test
  env_file:
    - ./hadoop.env
  networks:
    - bigdata-2

datanode:
  image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8
  container_name: datanode
  restart: always
  volumes:
    - hadoop_datanode:/hadoop/dfs/data
  environment:
    SERVICE_PRECONDITION: "namenode:9870"
  env_file:
    - ./hadoop.env
  networks:
   - bigdata-2
```

Docker compose



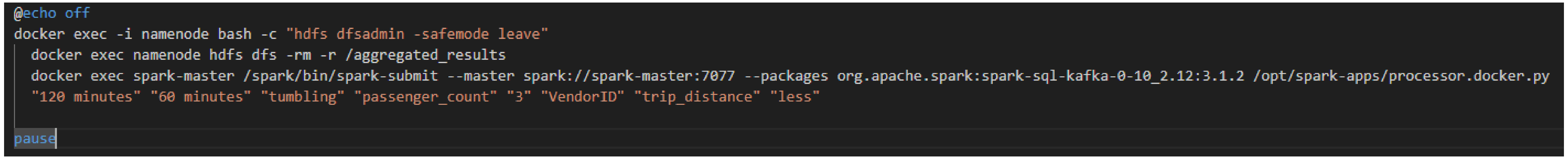
```yaml
resourcemanager:
  image: bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8
  container_name: resourcemanager
  restart: always
  environment:
    SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864"
  env_file:
    - ./hadoop.env
  networks:
    - bigdata-2

nodemanager:
  image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8
  container_name: nodemanager
  restart: always
  environment:
    SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"
  env_file:
    - ./hadoop.env
  networks:
   - bigdata-2

historyserver:
  image: bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8
  container_name: historyserver
  restart: always
  environment:
    SERVICE_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"
  volumes:
    - hadoop_historyserver:/hadoop/yarn/timeline
  env_file:
    - ./hadoop.env
  networks:
    - bigdata-2
```



```yaml
  spark-master:
    image: bde2020/spark-master:3.1.2-hadoop3.2
    container_name: spark-master
    ports:
      - "8070:8080"
      - "7077:7077"
    volumes:
      - ./spark-apps:/opt/spark-apps
    environment:
      - INIT_DAEMON_STEP=setup_spark
    networks:
      - bigdata-2

  spark-worker-1:
    image: bde2020/spark-worker:3.1.2-hadoop3.2
    container_name: spark-worker-1
    depends_on:
      - spark-master
    environment:
      - SPARK_MASTER=spark://spark-master:7077
    networks:
      - bigdata-2

  spark-worker-2:
    image: bde2020/spark-worker:3.1.2-hadoop3.2
    container_name: spark-worker-2
    depends_on:
      - spark-master
    environment:
      - SPARK_MASTER=spark://spark-master:7077
    networks:
      - bigdata-2

olumes:
  hadoop_namenode:
  hadoop_datanode:
  hadoop_historyserver:

etworks:
  bigdata-2:
    driver: bridge
```

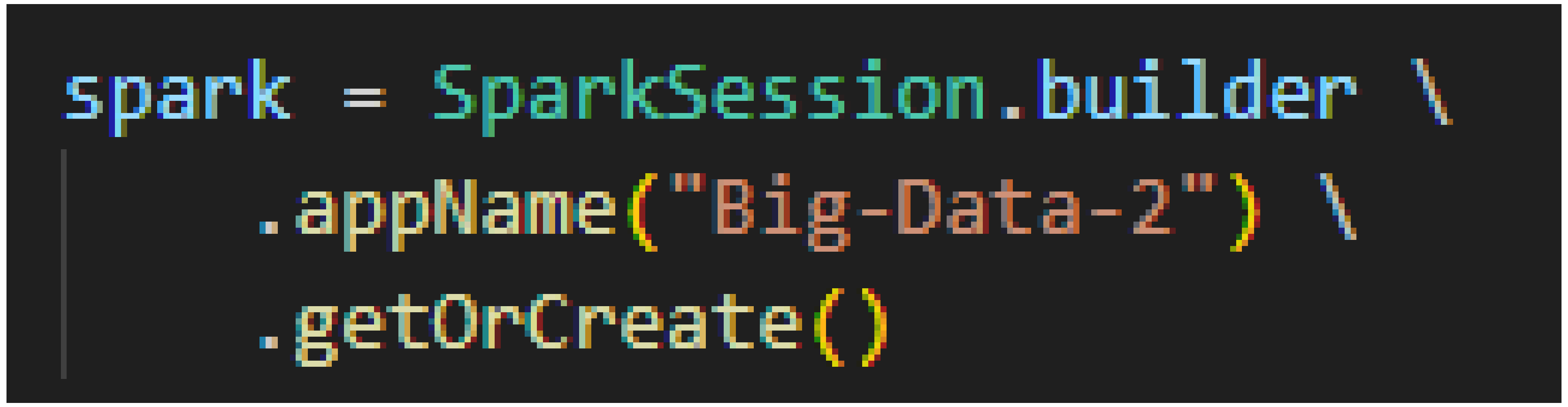
Docker compose

run.bat

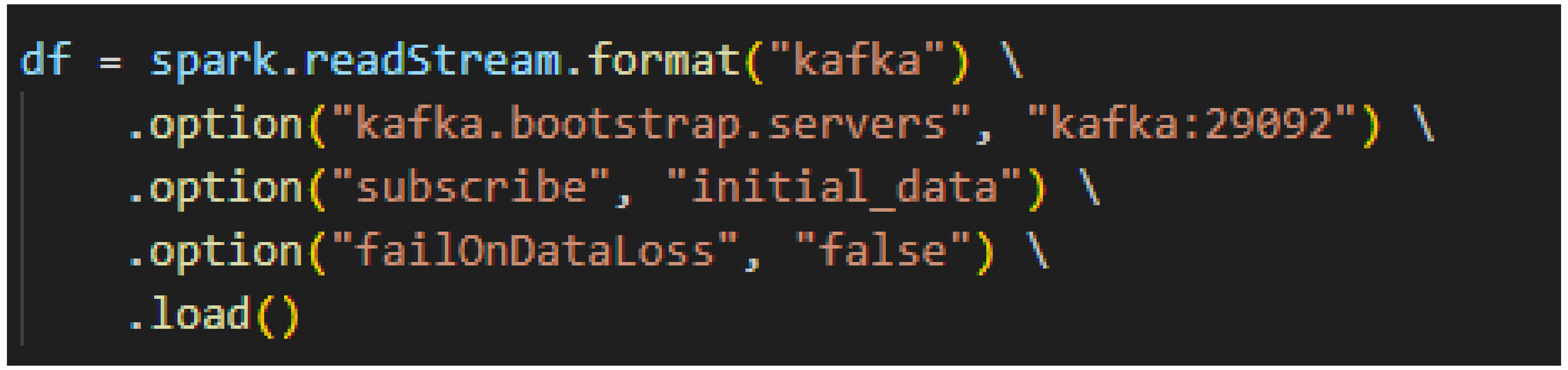


```
@echo off
docker exec -i namenode bash -c "hdfs dfsadmin -safemode leave"
  docker exec namenode hdfs dfs -rm -r /aggregated_results
  docker exec spark-master /spark/bin/spark-submit --master spark://spark-master:7077 --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2 /opt/spark-apps/processor.docker.py
  "120 minutes" "60 minutes" "tumbling" "passenger_count" "3" "VendorID" "trip_distance" "less"

pause
```

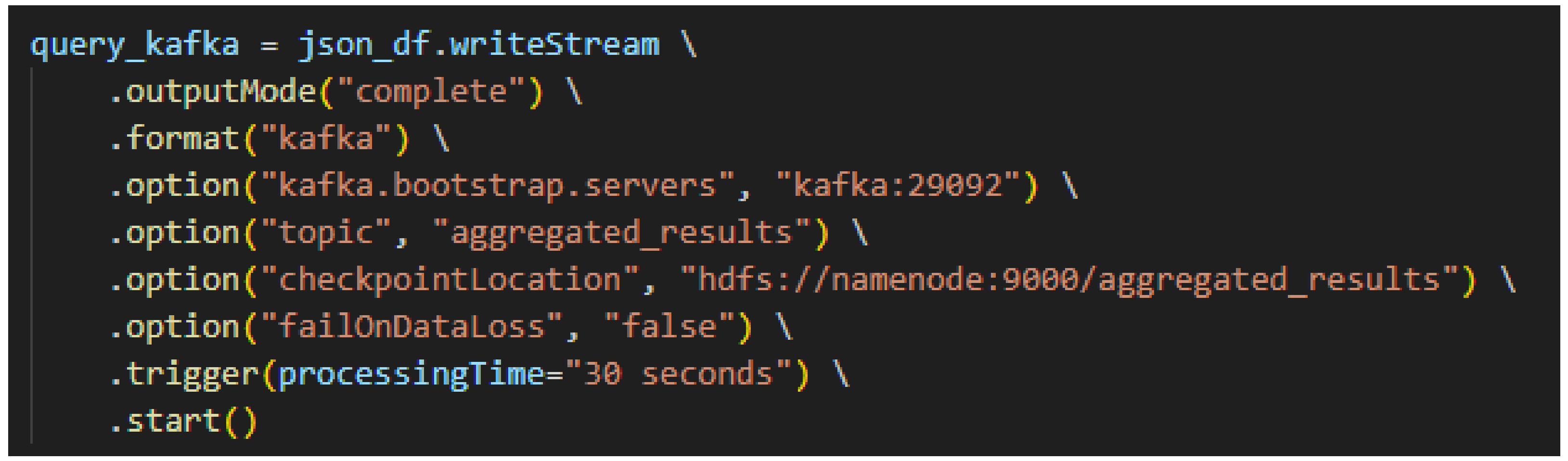
docker app



```
spark = SparkSession.builder \
    .appName("Big-Data-2") \
    .getOrCreate()
```



```
df = spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:29092") \
    .option("subscribe", "initial_data") \
    .option("failOnDataLoss", "false") \
    .load()
```



```
query_kafka = json_df.writeStream \
    .outputMode("complete") \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:29092") \
    .option("topic", "aggregated_results") \
    .option("checkpointLocation", "hdfs://namenode:9000/aggregated_results") \
    .option("failOnDataLoss", "false") \
    .trigger(processingTime="30 seconds") \
    .start()
```

Hvala na pažnji!