Big Data Engineering Group Assignment

Salmoli Chandra 24PGAI0057

Meera Karamta 24PGAI0003

Kishnu Srivastava 24PGAI0042

Praveen Kumar Singh 24PGAI0068


```
spark
```

Install Confluent Libraries

```
confluent-kafka[avro,json,protobuf]>=1.4.2
```

Streaming data from confluent servers

Endpoints

```
Bootstrap server : pkc-12576z.us-west2.gcp.confluent.cloud:9092
```

ConfluentIO Adaptor

```
Select output record value format: JSON
Select a template: RATINGS
```

Stream Ingestion

```
# Subscribe to 1 topic

confluentApiKey = 'VO4DBIVRWZQV6V4Q'

confluentSecret =
'Y5wqrwJTNldowy6/R6RqAgStwYc/I4GREAhmW+1BEkVyaphYevD0x01ttllpFcv2'

confluentTopicName = 'topic_0'

stream_df = ( spark
              .readStream
              .format("kafka")
              .option("kafka.bootstrap.servers", "pkc-12576z.us-
west2.gcp.confluent.cloud:9092")
              .option("kafka.security.protocol", "SASL_SSL")
              .option("kafka.sasl.jaas.config",
"kafkashaded.org.apache.kafka.common.security.plain.PlainLoginModule
```

```
required username='{}' password='{}';".format(confluentApiKey,
confluentSecret))
            .option("kafka.ssl.endpoint.identification.algorithm",
"https")
            .option("kafka.sasl.mechanism", "PLAIN")
            .option("subscribe", confluentTopicName)
            .option("startingOffsets", "earliest")
            .option("maxOffsetsPerTrigger", 10)
            .option("failOnDataLoss", "false")
            .load() )

message_df = stream_df.selectExpr("CAST(key AS STRING)", "CAST(value
AS STRING)")

from pyspark.sql.types import StructType, StructField, DoubleType,
IntegerType, StringType
from pyspark.sql.functions import col, from_json, split
from pyspark.sql.types import StructType, StructField, IntegerType,
StringType


schema = StructType([
    StructField("rating_id", IntegerType(), True),
    StructField("user_id", IntegerType(), True),
    StructField("stars", IntegerType(), True),
    StructField("route_id", IntegerType(), True),
    StructField("rating_time", IntegerType(), True),
    StructField("channel", StringType(), True),
    StructField("message", StringType(), True)
])


new_rating_df = message_df.select
(from_json("value",schema).alias("data")).select(col("data.*"))

new_rating_df = new_rating_df.withColumn("rating_time",
expr("from_unixtime(unix_timestamp('2000-01-01 00:00:00') +
rating_time*3600)"))

display(new_rating_df)

from pyspark.sql.functions import *

spark.conf.set("spark.sql.shuffle.partitions", "2")

usercount_df = new_rating_df.groupby(col("stars")).count()

query = (
  usercount_df
    .writeStream
    .trigger(processingTime='5 seconds')
```

```
    .format("memory")        # memory = store in-memory table (for testing only in Spark 2.0)
    .queryName("nstarcounts")     # counts = name of the in-memory table
    .outputMode("complete")  # complete = all the counts should be in the table
    .start()
)


new_rating_df.writeStream\
  .format("delta")\
  .outputMode("append")\
  .option("checkpointLocation", "/tmp/delta/ratings/_checkpoints/")\
  .start("/delta/ratings")


Out[71]: <pyspark.sql.streaming.query.StreamingQuery at 0x7f7678908bb0>

# %fs
# rm -r /delta/ratings
# ls /delta/ratings
```