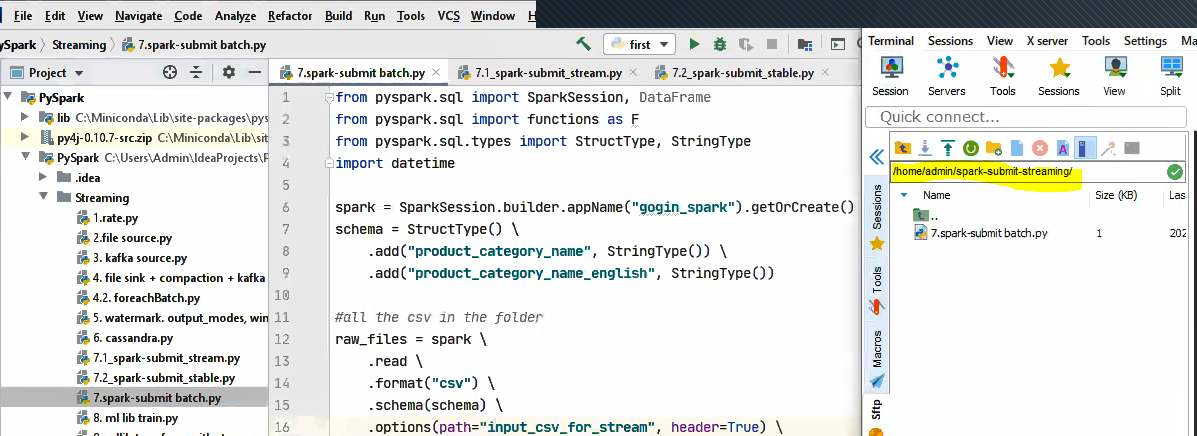
**ЗАДАНИЕ 07:**

Повторить запуск Spark приложений с такими параметрами (можно еще добавлять свои) /spark2.4/bin/spark-submit --driver-memory 512m --driver-cores 1 --master local[1] my\_script.py

# **SPARK SUBMIT BATCH**

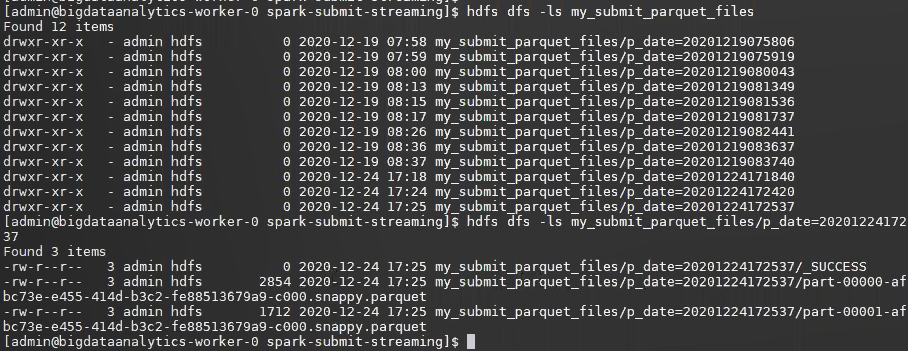
from pyspark.sql import SparkSession, DataFrame  
from pyspark.sql import functions as F  
from pyspark.sql.types import StructType, StringType  
import datetime  
  
spark = SparkSession.builder.appName(**"gogin\_spark"**).getOrCreate()  
schema = StructType() \  
 .add(**"product\_category\_name"**, StringType()) \  
 .add(**"product\_category\_name\_english"**, StringType())  
  
#читаем все csv в батче  
raw\_files = spark \  
 .read \  
 .format(**"csv"**) \  
 .schema(schema) \  
 .options(path=**"input\_csv\_for\_stream"**, header=True) \  
 .load()  
  
#fix timestamp  
load\_time = datetime.datetime.now().strftime(**"%Y%m%d%H%M%S"**)  
print(**"START BATCH LOADING. TIME = "** + load\_time)  
  
#пишем паркеты в партиции  
raw\_files.withColumn(**"p\_date"**, F.lit(**"load\_time"**)) \  
 .write \  
 .mode(**"append"**) \  
 .parquet(**"my\_submit\_parquet\_files/p\_date="** + str(load\_time))  
  
print(**"FINISHED BATCH LOADING. TIME = "** + load\_time)  
  
spark.stop() #в конце stop не обязательно. SUBMIT сам вырубает spark в конце

Файл можно перетащить на сервер (mobaXterm + drag&drop):



Запускаем из консоли вот так:

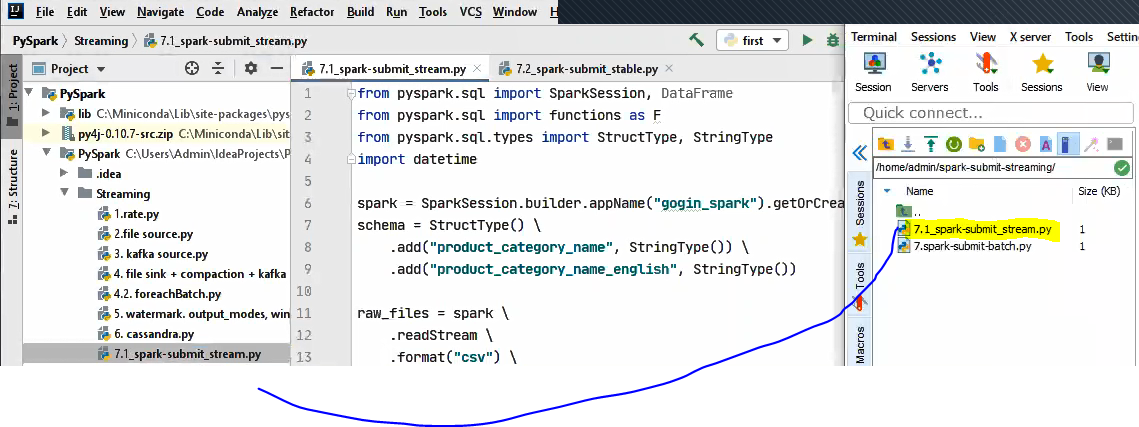




# **SPARK SUBMIT STREAM**

from pyspark.sql import SparkSession, DataFrame  
from pyspark.sql import functions as F  
from pyspark.sql.types import StructType, StringType  
import datetime  
  
spark = SparkSession.builder.appName(**"gogin\_spark"**).getOrCreate()  
schema = StructType() \  
 .add(**"product\_category\_name"**, StringType()) \  
 .add(**"product\_category\_name\_english"**, StringType())  
  
#читаем csv файлы в стриме  
raw\_files = spark \  
 .readStream \  
 .format(**"csv"**) \  
 .schema(schema) \  
 .options(path=**"input\_csv\_for\_stream"**, header=True) \  
 .load()  
  
# разово проставляем время загрузки  
load\_time = datetime.datetime.now().strftime(**"%Y%m%d%H%M%S"**)  
  
#ВСЕГДА ПИЩЕМ И ОДНУ ДИРЕКТОРИЮ  
def file\_sink(df, freq):  
 return df.writeStream.format(**"parquet"**) \  
 .trigger(processingTime=**'%s seconds'** % freq ) \  
 .option(**"path"**,**"my\_submit\_parquet\_files/p\_date="** + str(load\_time)) \  
 .option(**"checkpointLocation"**, **"checkpionts/my\_parquet\_checkpoint"**) \  
 .start()  
  
timed\_files = raw\_files.withColumn(**"p\_date"**, F.lit(**"load\_time"**))  
  
#запускаем стрим всегда в одну директорию  
stream = file\_sink(timed\_files,10)  
  
#will always spark.stop() at the end  
  
#СТРИМ ТУТ ЖЕ ЗАКОНЧИТСЯ ПОТОМУ ЧТО В КОНЦЕ SPARK.STOP()

Аналогично копируем на сервер:



Здесь имеем ситуацию, что приложение по коду доходит до конца и закрывается само. Бесконечно stream крутиться не будет

Поэтому попробуем модифицировать, привнеся ЦИКЛ

# **SPARK SUBMIT STABLE**

from pyspark.sql import SparkSession, DataFrame  
from pyspark.sql import functions as F  
from pyspark.sql.types import StructType, StringType  
import datetime  
  
spark = SparkSession.builder.appName(**"gogin\_spark"**).getOrCreate()  
schema = StructType() \  
 .add(**"product\_category\_name"**, StringType()) \  
 .add(**"product\_category\_name\_english"**, StringType())  
  
raw\_files = spark \  
 .readStream \  
 .format(**"csv"**) \  
 .schema(schema) \  
 .options(path=**"input\_csv\_for\_stream"**, header=True) \  
 .load()  
  
#пишем стрим в foreachBatch, чтобы делать логику в зависимости от каждого микробатча  
def file\_sink(df, freq):  
 return df.writeStream.foreachBatch(foreach\_batch\_function) \  
 .trigger(processingTime=**'%s seconds'** % freq ) \  
 .option(**"checkpointLocation"**, **"checkpionts/my\_parquet\_checkpoint"**) \  
 .start()  
  
#в каждом микробатче фиксируем время, логируем на экран, пишем файлы в свою директорию  
def foreach\_batch\_function(df, epoch\_id):  
 load\_time = datetime.datetime.now().strftime(**"%Y%m%d%H%M%S"**)  
 print(**"START BATCH LOADING. TIME = "** + load\_time)  
 df.withColumn(**"p\_date"**, F.lit(**"load\_time"**)) \  
 .write \  
 .mode(**"append"**) \  
 .parquet(**"my\_submit\_parquet\_files/p\_date="** + str(load\_time))  
 print(**"FINISHED BATCH LOADING. TIME = "** + load\_time)  
  
stream = file\_sink(raw\_files,10)  
  
#запускаем бесконечный цикл  
while(True):  
 print(**"I'M STILL ALIVE"**)  
 stream.awaitTermination(9)  
  
#unreachable  
spark.stop()

**ЗАДАНИЕ 08:**

В качестве итоговой работы необходимо написать свою ML модель. Обучить ее. Затем применять на стриме.  
Необходимо найти / сгененрировать собственные входные данные и объяснить, какую задачу решает ваша ML модель и почему именно так.