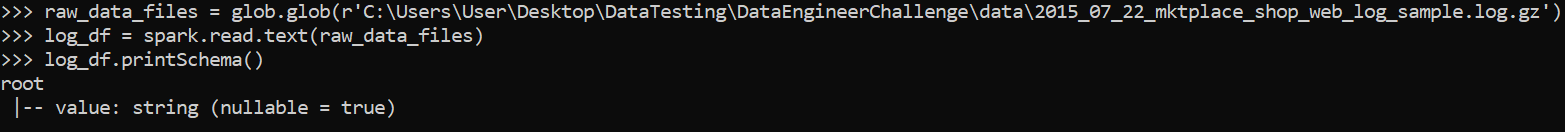
**#reading the log file**

raw\_data\_files = glob.glob(r'*C:\Users\User\Desktop\DataTesting\DataEngineerChallenge\data\2015\_07\_22\_mktplace\_shop\_web\_log\_sample.log.gz'*)

log\_df = spark.read.text(raw\_data\_files)

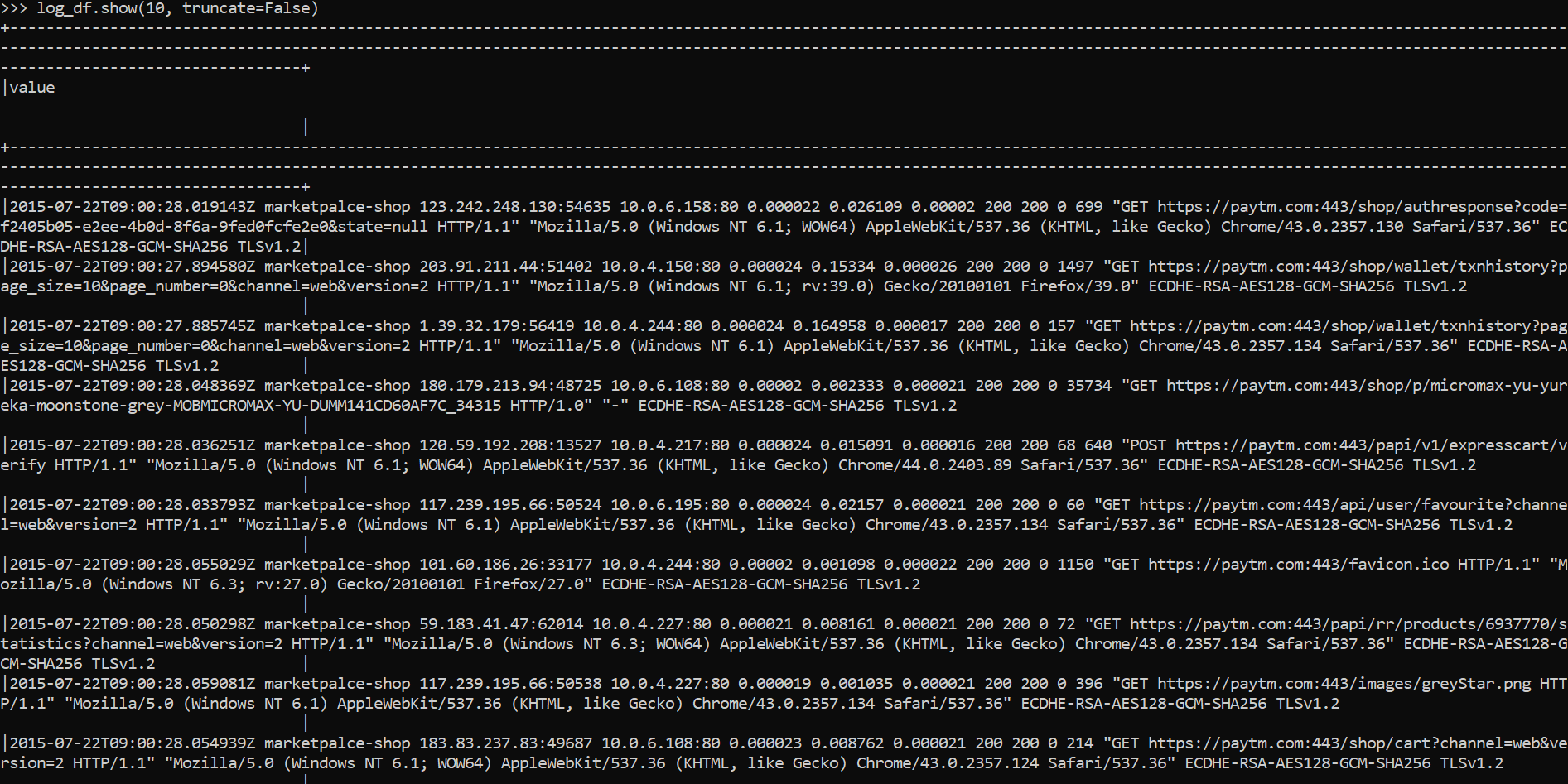
**#checking the schema**

log\_df.printSchema()



**#checking sample data in dataframe**

log\_df.show(10, truncate=False)



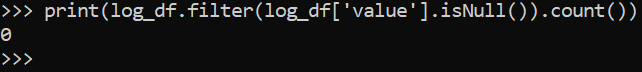
**#Checking row count to match after converting to multiple columns**

print((log\_df.count(), len(log\_df.columns)))



**#Checking null value**

print(log\_df.filter(log\_df['value'].isNull()).count())



**#Converting single column in dataframe to multiple columns based on the schema**

split\_col = split(log\_df['value'], ' ')

split\_ip = split(split\_log\_df['clientipport'], ':')

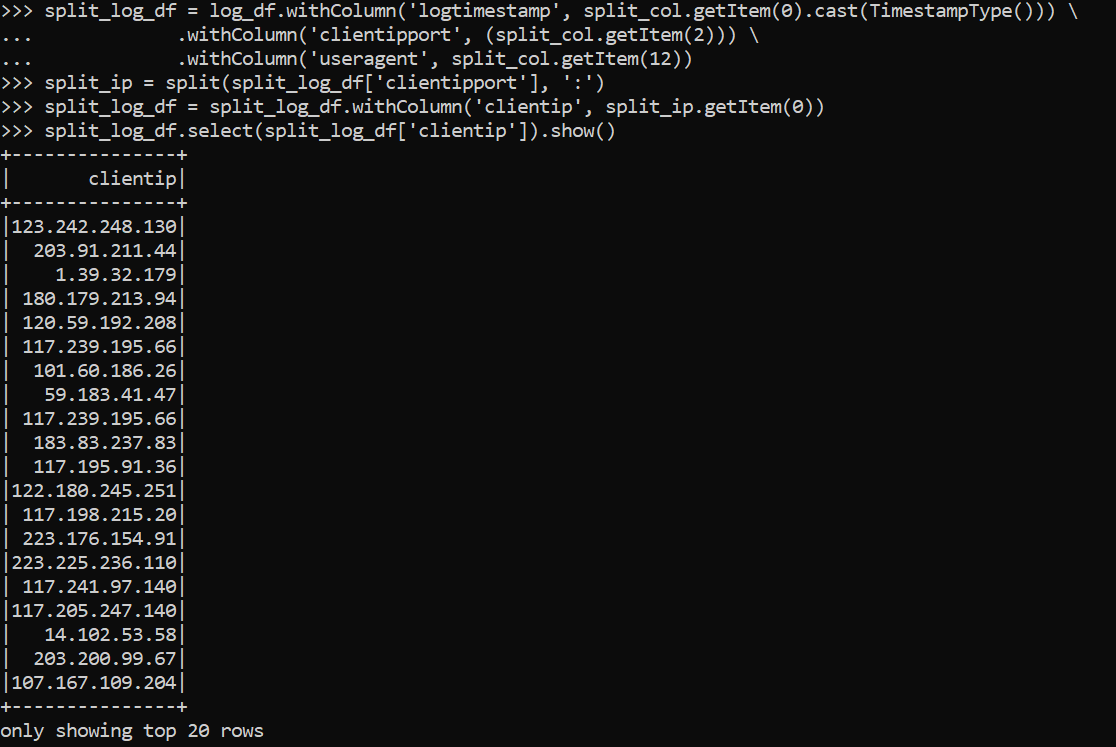
ip\_pattern = r'(^\S+\.[\S+\.]+\S+)\s'

split\_log\_df = log\_df.withColumn('logtimestamp', split\_col.getItem(0).cast(TimestampType())) \

.withColumn('clientipport', (split\_col.getItem(2))) \

.withColumn('useragent', split\_col.getItem(12))

split\_log\_df = split\_log\_df.withColumn('clientip', split\_ip.getItem(0))



**# sessionizing data based on 15 min fixed window time and counting the hits**

**# assigning an uniqeId to each session grouped by ip and count of hits**

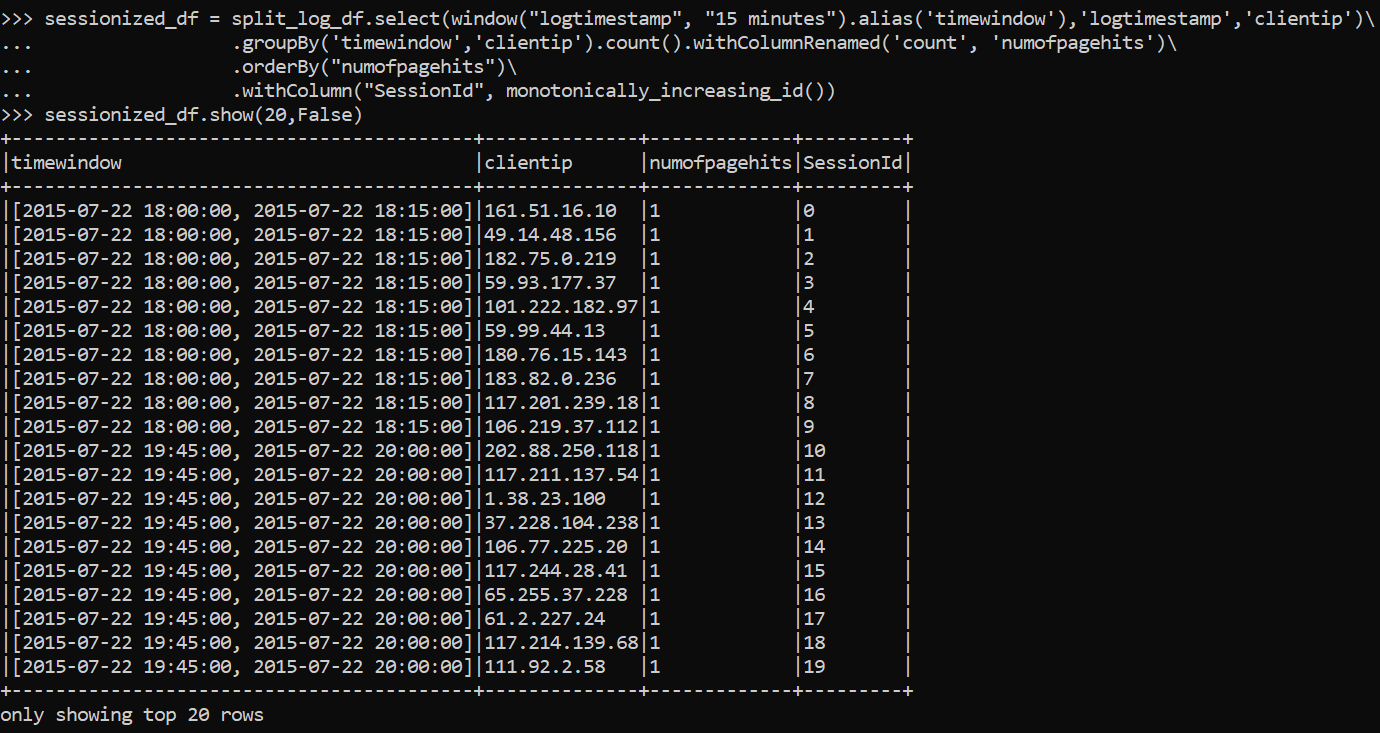
sessionized\_df = split\_log\_df.select(window("logtimestamp", "15 minutes").alias('timewindow'),'logtimestamp','clientip')\

.groupBy('timewindow','clientip').count().withColumnRenamed('count', 'numofpagehits')\

.orderBy("numofpagehits")\

.withColumn("sessionid", monotonically\_increasing\_id())

sessionized\_df.show(20,False)

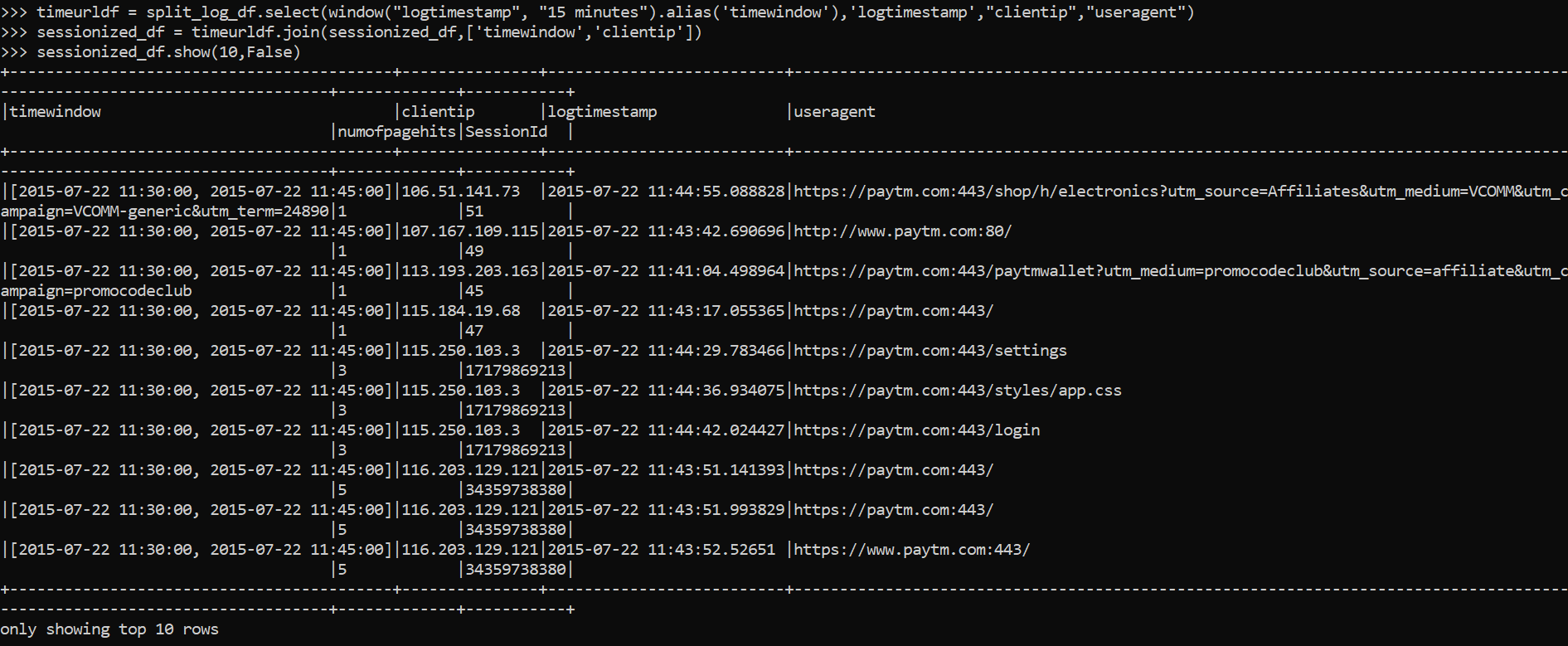


**# adding timestamp and url to the session dataframe for finding the first hit time**

timeurldf = split\_log\_df.select(window("logtimestamp", "15 minutes").alias('timewindow'),'logtimestamp',"clientip","useragent")

sessionized\_df = timeurldf.join(sessionized\_df,['timewindow','clientip'])

sessionized\_df.show(10,False)

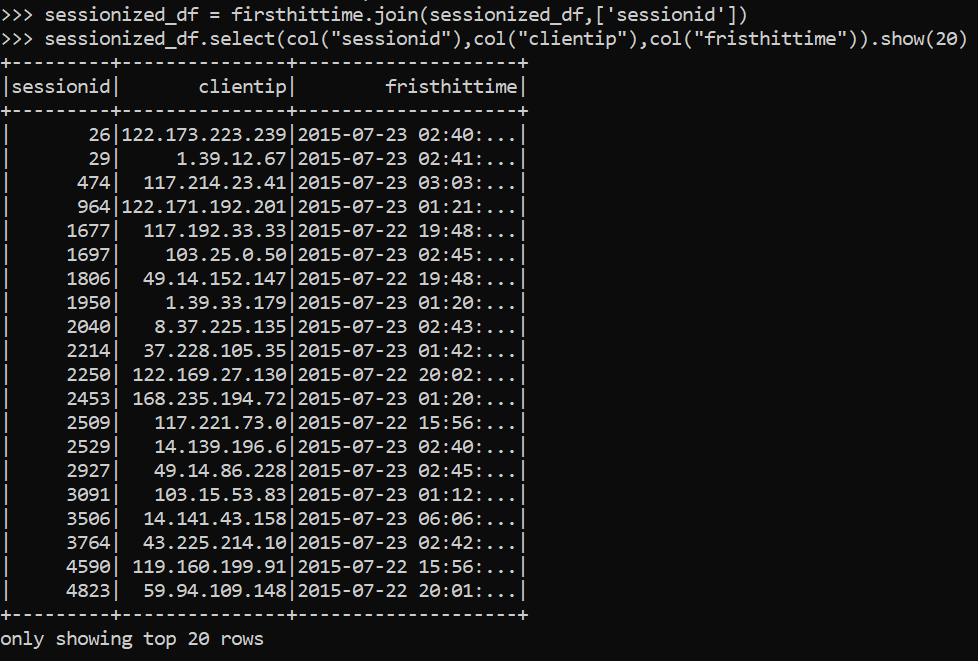


**# Finding the first hit timestamp for each ip for each session**

firsthittime = sessionized\_df.groupBy("sessionid").agg(min("logtimestamp").alias('fristhittime'))

sessionized\_df = firsthittime.join(sessionized\_df,['sessionid'])

sessionized\_df.select(col("sessionid"),col("clientip"),col("fristhittime")).show(20)



**# Average session time**

**# the time difference between first and last hit in a session is the duration of a session for each IP**

**# if there is only one hit in a session the duration is zero**

sessiontimediff = (unix\_timestamp(sessionized\_df.logtimestamp)-unix\_timestamp(sessionized\_df.fristhittime))

sessionized\_df = sessionized\_df.withColumn("timediff", sessiontimediff)

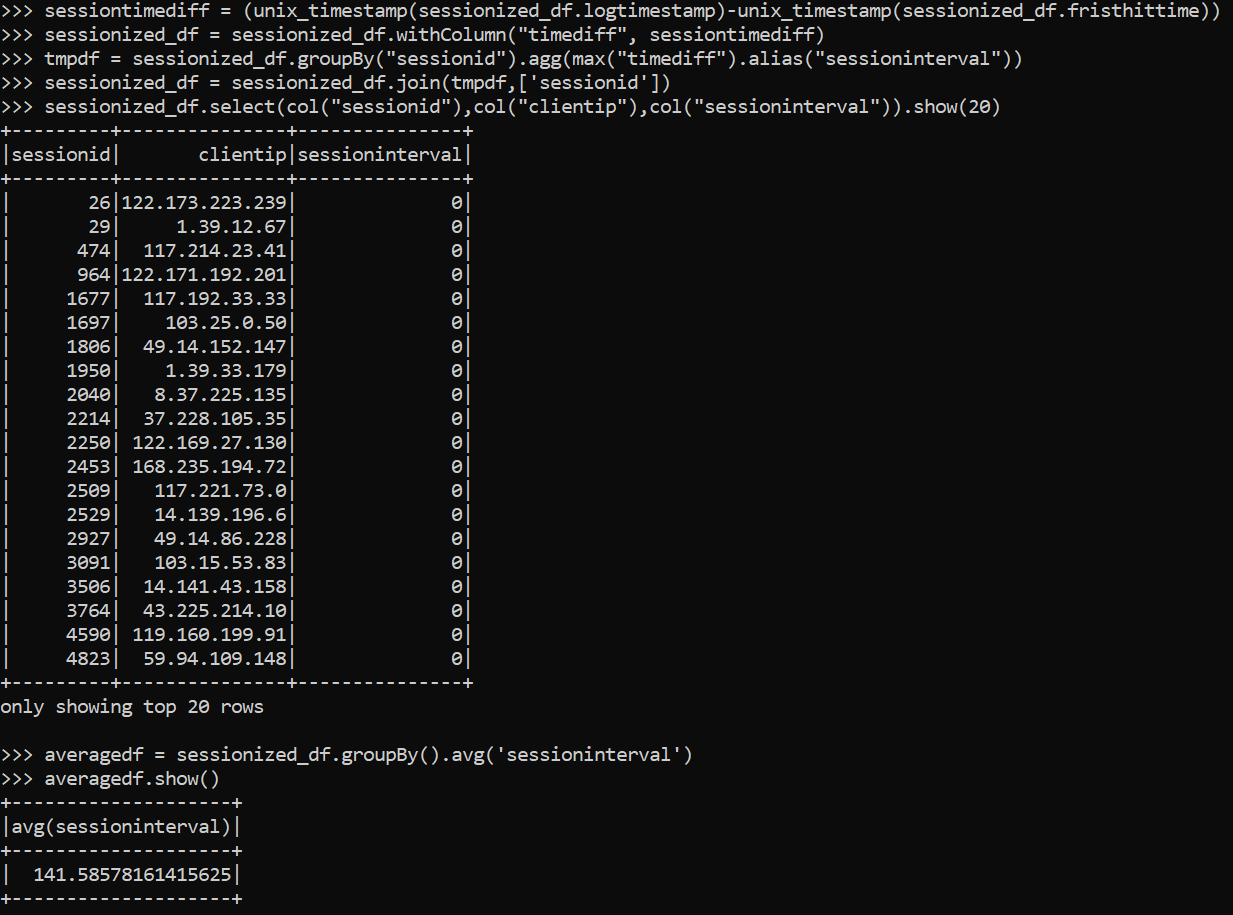
tmpdf = sessionized\_df.groupBy("sessionid").agg(max("timediff").alias("sessioninterval"))

sessionized\_df = sessionized\_df.join(tmpdf,['sessionid'])

sessionized\_df.select(col("sessionid"),col("clientip"),col("sessioninterval")).show(20)

averagedf = sessionized\_df.groupBy().avg('sessioninterval')

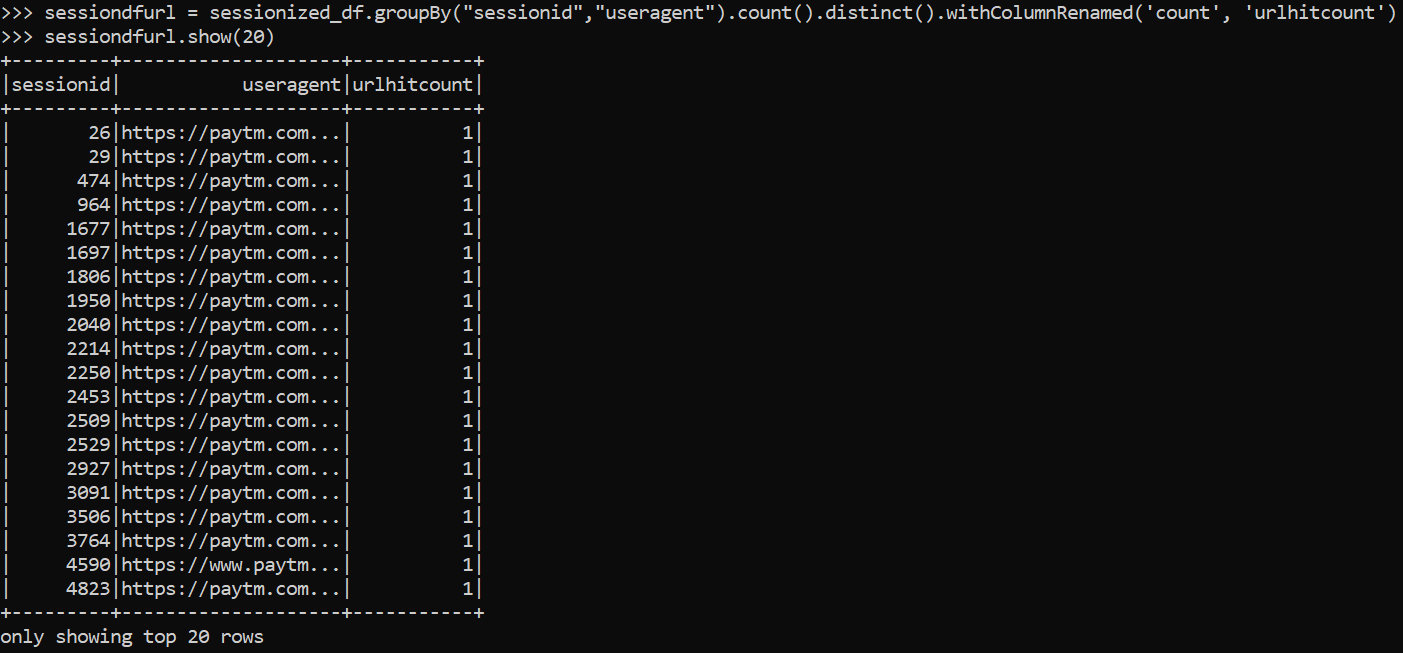
averagedf.show()



**#URL visits per session. unique URL count once per session**

sessiondfurl = sessionized\_df.groupBy("sessionid","useragent").count().distinct().withColumnRenamed('count', 'urlhitcount')

sessiondfurl.show(20)



**#most engaged users, IPs with the longest session times**

mostengageduser = sessionized\_df.select("clientip","sessionid","sessioninterval").sort(col("sessioninterval").desc()).distinct()

mostengageduser.show(20)

