**Code Snippets for Paytm code challenge test**

I used **Databricks Community Edition** to execute this below code snippets

1. Sessionize the web log by IP. Sessionize = aggregrate all page hits by visitor/IP during a session.

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.expressions.Window

import org.apache.spark.sql.RelationalGroupedDataset

val logDataDF = spark.read.format("csv")

.option("delimiter"," ")

.option("header","true")

.load("/FileStore/tables/2015\_07\_22\_mktplace\_shop\_web\_log\_sample.csv")

var logDataFinalDF= logDataDF.withColumn("url", split($"request"," ").getItem(1))

.withColumn("ip",split($"client\_port",":").getItem(0)) .withColumn("timestamp",$"timestamp".cast("timestamp"))

.withColumn("time",unix\_timestamp($"timestamp", "yyyy-MM-dd hh:mm:ss"))

val sessionWindow = Window.partitionBy(window(logDataFinalDF("timestamp"),"15 minutes")).orderBy(window(logDataFinalDF("timestamp"),"15 minutes"))

var sessionizedDF = logDataFinalDF.withColumn("rn",row\_number().over(sessionWindow ))

.withColumn("total\_count",max('rn).over(sessionWindow ))

.withColumn("ip\_cnt",count("ip").over(sessionWindow ))

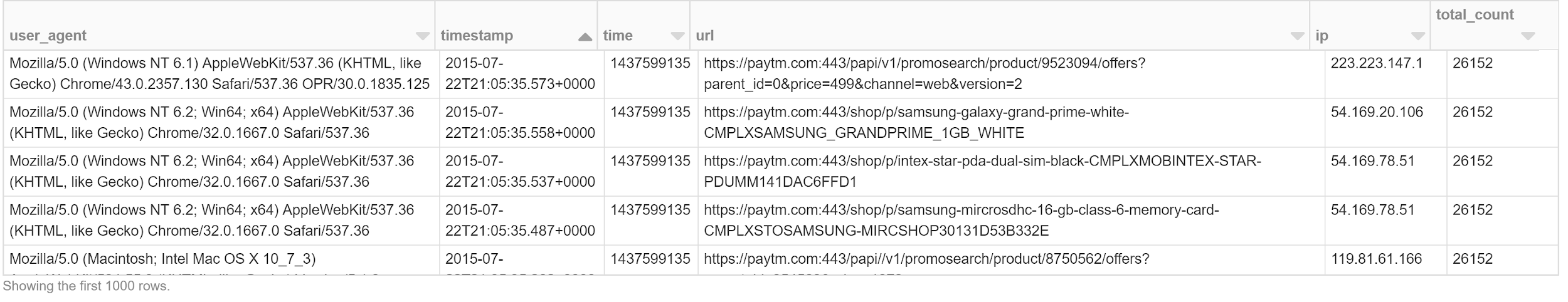
.withColumn("url\_cnt",count("url").over(sessionWindow ))

.select("user\_agent","timestamp","time","url","ip","total\_count")

.distinct()

display(sessionizedDF)

Output Screenshot :



2) Determine the average session time ( deriving users average session time)

val userWindow= Window.partitionBy(logDataFinalDF("user\_agent"))

.orderBy(logDataFinalDF("user\_agent").desc)

var avgSessionDF = logDataFinalDF.withColumn("rn",row\_number().over(userWindow))

.withColumn("total\_count",max('rn).over(userWindow))

.withColumn("session\_start\_time",min(logDataFinalDF("time")).over(userWindow))

.withColumn("session\_end\_time",max(logDataFinalDF("time")).over(userWindow))

.withColumn("user\_session\_time",'session\_end\_time-'session\_start\_time)

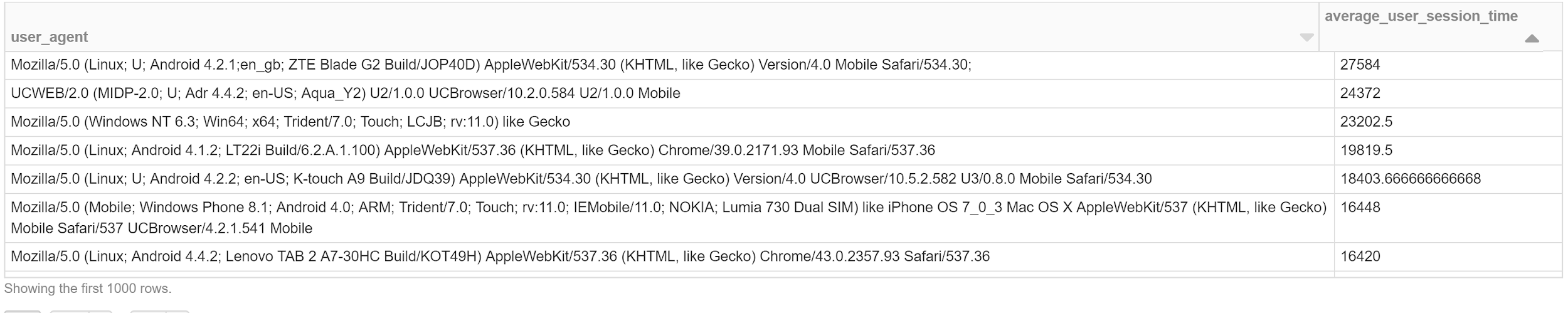
.where("rn=1")

avgSessionDF = avgSessionDF .withColumn("average\_user\_session\_time",'user\_session\_time/'total\_count)

.select("user\_agent","average\_user\_session\_time")

display(avgSessionDF)

Output is (average\_user\_session\_time is in seconds) :



3) Determine unique URL visits per session. To clarify, count a hit to a unique URL only once per session.

var uniqueUrlDF= logDataFinalDF.select("\*")

.groupBy(window('timestamp,"15 minutes"),'url)

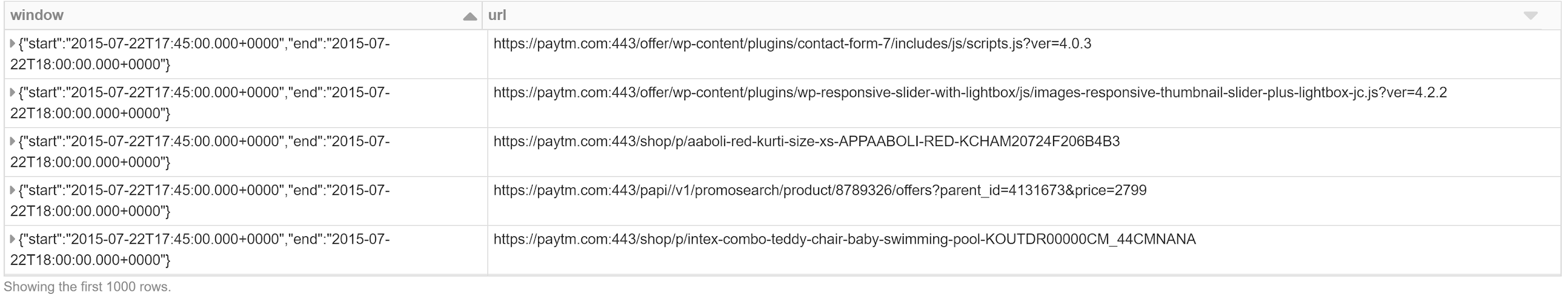
.count()

var uniqueUrlFinalDF= uniqueUrlDF.select("window","url")

.distinct()

display(uniqueUrlFinalDF)

Output is:



4) Find the most engaged users, ie the IPs with the longest session times

val ipWindow=Window.partitionBy(logDataFinalDF("ip"))

.orderBy(logDataFinalDF("ip").asc)

var ipSessionDF = logDataFinalDF.withColumn("rn",row\_number().over(ipWindow))

.withColumn("total\_count",max('rn).over(ipWindow))

.withColumn("session\_start\_time", min(logDataFinalDF("time")).over(ipWindow)) .withColumn("session\_end\_time",max(logDataFinalDF("time")).over(ipWindow))

.withColumn("ip\_session\_time",'session\_end\_time-'session\_start\_time)

.select("user\_agent","ip","ip\_session\_time","url")

.where("rn=1").orderBy('ip\_session\_time.desc)

display(ipSessionDF)

Output is:

