import org.apache.spark.SparkContext

import org.apache.spark.sql.\_

import org.apache.spark.SparkContext.\_

import org.apache.spark.sql.{DataFrame, Row, SQLContext, SparkSession, Column}

import com.datastax.driver.core.\_

import com.datastax.driver.core.Session

import org.apache.log4j.{Level, LogManager, Logger}

import java.util.Date

import java.util.{Calendar, Date}

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

import org.apache.spark.sql.functions.{to\_date, to\_timestamp}

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

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

import org.apache.spark.sql.functions.rank

//import org.apache.spark.implicits.\_

import org.apache.spark.sql.functions.col

import com.datastax.driver.core.Cluster

import com.datastax.spark.connector.\_

import org.apache.spark.SparkConf

object ScalaDataChurn {

def main(args: Array[String]): Unit = {

println("Process start")

//System.setProperty("hadoop.home.dir", "c:\\\winutil\\\")

/\*

Logger.getLogger("org.apache").setLevel(Level.WARN)

//Connection with Cassandra 3.11

//Connection with Cassandra Datstax

val spark = SparkSession

.builder()

.master("local")

.appName("Spark\_Cassandra")

.config("spark.cassandra.connection.host", "127.0.0.1")

.getOrCreate()

val df = spark

.read

.format("org.apache.spark.sql.cassandra")

.options(Map( "table" -> "endpoint\_application\_months", "keyspace" -> "rcml"))

.load()

//df.show()

df.createOrReplaceTempView("endpoint")

val sqlDF = spark.sql("SELECT \* FROM endpoint")

//sqlDF.show()

// Fetching data from application\_groups

val cf = spark

.read

.format("org.apache.spark.sql.cassandra")

.options(Map( "table" -> "application\_groups", "keyspace" -> "rcml"))

.load()

cf.createOrReplaceTempView("AppGroup")

val sqlCF = spark.sql("SELECT \* FROM AppGroup")

//sqlCF.show()

val df3 = spark.sql("SELECT a.\*,b.traffic\_type\_name from endpoint a left join AppGroup b on a.application\_id=b.app\_id")

//df3.show()

//df3.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").save("C://output\_files2//myjoin.csv")

df3.createOrReplaceTempView("DateData")

val sqlJoin = spark.sql("SELECT \* FROM DateData where endpoint\_id> 0")

//sqlJoin.show()

//Check if there in not any Traffic\_type available, fill it with “Other

val newsqlJoin = sqlJoin.na.fill("Other",Seq("traffic\_type\_name"))

//newsqlJoin.show()

//To get Month and Year column from date\_time column

val MonthYear = newsqlJoin.withColumn("year", year(col("date\_time"))).withColumn("month", month(col("date\_time")))

// MonthYear.show()

//Subset of the data on the basis of the selected columns

val limitcolumn = MonthYear.select("org\_id", "endpoint\_id", "traffic\_type\_name", "year", "month",

"ds\_bytes", "ds\_mwt", "us\_bytes", "us\_mwt", "ds\_max\_bps",

"us\_max\_bps", "ds\_avg\_bps","us\_avg\_bps", "date\_time")

//limitcolumn.show()

//To replace blank spaces with underscore(\_)

val cleanTraffic1 = limitcolumn.withColumn("traffic\_type\_name", regexp\_replace(col("traffic\_type\_name"),"/","\_"))

val cleanTraffic2 = cleanTraffic1.withColumn("traffic\_type\_name", regexp\_replace(col("traffic\_type\_name"),",","\_"))

val cleanTraffic3 = cleanTraffic2.withColumn("traffic\_type\_name", regexp\_replace(col("traffic\_type\_name"),"&","and"))

val cleanTraffic = cleanTraffic3.withColumn("traffic\_type\_name", regexp\_replace(col("traffic\_type\_name")," ","\_"))

cleanTraffic.show(false)

//Aggregation of the selected columns

val AggreSum = { cleanTraffic.groupBy("org\_id","year","month","traffic\_type\_name","endpoint\_id").agg(sum("ds\_bytes").alias("Sum\_ds\_bytes"),

sum("us\_bytes").alias("Sum\_us\_bytes"),sum("ds\_mwt").alias("Sum\_ds\_mwt"),

sum("us\_mwt").alias("Sum\_us\_mwt"))

}

// AggreSum.show()

val AggreMax = { cleanTraffic.groupBy("org\_id","year","month","traffic\_type\_name","endpoint\_id").agg(max("ds\_max\_bps").alias("Max\_ds\_max\_bps"),

max("us\_max\_bps").alias("Max\_us\_max\_bps"))

}

// AggreMax.show()

val AggreAvg = { cleanTraffic.groupBy("org\_id","year","month","traffic\_type\_name","endpoint\_id").agg(avg("ds\_avg\_bps").alias("Avg\_ds\_avg\_bps"),

avg("us\_avg\_bps").alias("Avg\_us\_avg\_bps"))

}

// AggreAvg.show()

println("Agg show")

//Conversion of the bytes into MB and time into hours

val conversion = {

cleanTraffic.withColumn("ds\_megabyte", bround(col("ds\_bytes")/(1000\*1000),2))

.withColumn("us\_megabyte",bround(col("us\_bytes")/(1000\*1000),2))

.withColumn("ds\_max\_bps\_mb",bround(col("ds\_max\_bps")/(1000\*1000),2))

.withColumn("us\_max\_bps\_mb",bround(col("us\_max\_bps")/(1000\*1000),2))

.withColumn("ds\_avg\_bps\_mb",bround(col("ds\_avg\_bps")/(1000\*1000),2))

.withColumn("us\_avg\_bps\_mb",bround(col("us\_avg\_bps")/(1000\*1000),2))

.withColumn("ds\_Hours",bround(col("ds\_mwt")/(60\*60),2))

.withColumn("us\_Hours",bround(col("us\_mwt")/(60\*60),2))

}

println("Conversion show")

val conpivot = conversion.groupBy("org\_id","endpoint\_id","year","month").pivot("traffic\_type\_name").agg(sum("ds\_megabyte").alias("Sum\_ds\_mbytes"),

sum("us\_megabyte").alias("Sum\_us\_mbytes"),sum("ds\_Hours").alias("Sum\_ds\_mmwt"),

sum("us\_Hours").alias("Sum\_us\_mmwt"),max("ds\_max\_bps\_mb").alias("Max\_ds\_max\_mbps"),

max("us\_max\_bps\_mb").alias("Max\_us\_max\_mbps"),avg("ds\_avg\_bps\_mb").alias("Avg\_ds\_avg\_mbps"),

avg("us\_avg\_bps\_mb").alias("Avg\_us\_avg\_mbps")).na.fill(0)

// conpivot.show()

println("pivot show")

// To create Delta

val windowSpec = Window.partitionBy("endpoint\_id").orderBy("month")

val DiffCol = conpivot.withColumn("Diff\_Amazon\_Sum\_ds\_mbytes", col("Amazon\_Sum\_ds\_mbytes") - when((lag("Amazon\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Amazon\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Amazon\_Sum\_us\_mbytes", col("Amazon\_Sum\_us\_mbytes") - when((lag("Amazon\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Amazon\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Amazon\_Sum\_ds\_mmwt", col("Amazon\_Sum\_ds\_mmwt") - when((lag("Amazon\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Amazon\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Amazon\_Sum\_us\_mmwt", col("Amazon\_Sum\_us\_mmwt") - when((lag("Amazon\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Amazon\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Amazon\_Max\_ds\_max\_mbps", col("Amazon\_Max\_ds\_max\_mbps") - when((lag("Amazon\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Amazon\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Amazon\_Max\_us\_max\_mbps", col("Amazon\_Max\_us\_max\_mbps") - when((lag("Amazon\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Amazon\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Amazon\_Avg\_ds\_avg\_mbps", col("Amazon\_Avg\_ds\_avg\_mbps") - when((lag("Amazon\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Amazon\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Amazon\_Avg\_us\_avg\_mbps", col("Amazon\_Avg\_us\_avg\_mbps") - when((lag("Amazon\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Amazon\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_ds\_mbytes", col("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_ds\_mbytes") - when((lag("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_us\_mbytes", col("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_us\_mbytes") - when((lag("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_ds\_mmwt", col("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_ds\_mmwt") - when((lag("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_us\_mmwt", col("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_us\_mmwt") - when((lag("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Network\_Infrastructure\_and\_IP\_Protocols\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Network\_Infrastructure\_and\_IP\_Protocols\_Max\_ds\_max\_mbps", col("Network\_Infrastructure\_and\_IP\_Protocols\_Max\_ds\_max\_mbps") - when((lag("Network\_Infrastructure\_and\_IP\_Protocols\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Network\_Infrastructure\_and\_IP\_Protocols\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Network\_Infrastructure\_and\_IP\_Protocols\_Max\_us\_max\_mbps", col("Network\_Infrastructure\_and\_IP\_Protocols\_Max\_us\_max\_mbps") - when((lag("Network\_Infrastructure\_and\_IP\_Protocols\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Network\_Infrastructure\_and\_IP\_Protocols\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Network\_Infrastructure\_and\_IP\_Protocols\_Avg\_ds\_avg\_mbps", col("Network\_Infrastructure\_and\_IP\_Protocols\_Avg\_ds\_avg\_mbps") - when((lag("Network\_Infrastructure\_and\_IP\_Protocols\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Network\_Infrastructure\_and\_IP\_Protocols\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Network\_Infrastructure\_and\_IP\_Protocols\_Avg\_us\_avg\_mbps", col("Network\_Infrastructure\_and\_IP\_Protocols\_Avg\_us\_avg\_mbps") - when((lag("Network\_Infrastructure\_and\_IP\_Protocols\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Network\_Infrastructure\_and\_IP\_Protocols\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

/\*

.withColumn("Diff\_Web\_Cloud\_Storage\_Hosting\_Sum\_ds\_mbytes", col("Web\_Cloud\_Storage\_Hosting\_Sum\_ds\_mbytes") - when((lag("Web\_Cloud\_Storage\_Hosting\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Web\_Cloud\_Storage\_Hosting\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Web\_Cloud\_Storage\_Hosting\_Sum\_us\_mbytes", col("Web\_Cloud\_Storage\_Hosting\_Sum\_us\_mbytes") - when((lag("Web\_Cloud\_Storage\_Hosting\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Web\_Cloud\_Storage\_Hosting\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Web\_Cloud\_Storage\_Hosting\_Sum\_ds\_mmwt", col("Web\_Cloud\_Storage\_Hosting\_Sum\_ds\_mmwt") - when((lag("Web\_Cloud\_Storage\_Hosting\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Web\_Cloud\_Storage\_Hosting\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Web\_Cloud\_Storage\_Hosting\_Sum\_us\_mmwt", col("Web\_Cloud\_Storage\_Hosting\_Sum\_us\_mmwt") - when((lag("Web\_Cloud\_Storage\_Hosting\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Web\_Cloud\_Storage\_Hosting\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Web\_Cloud\_Storage\_Hosting\_Max\_ds\_max\_mbps", col("Web\_Cloud\_Storage\_Hosting\_Max\_ds\_max\_mbps") - when((lag("Web\_Cloud\_Storage\_Hosting\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Web\_Cloud\_Storage\_Hosting\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Web\_Cloud\_Storage\_Hosting\_Max\_us\_max\_mbps", col("Web\_Cloud\_Storage\_Hosting\_Max\_us\_max\_mbps") - when((lag("Web\_Cloud\_Storage\_Hosting\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Web\_Cloud\_Storage\_Hosting\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Web\_Cloud\_Storage\_Hosting\_Avg\_ds\_avg\_mbps", col("Web\_Cloud\_Storage\_Hosting\_Avg\_ds\_avg\_mbps") - when((lag("Web\_Cloud\_Storage\_Hosting\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Web\_Cloud\_Storage\_Hosting\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Web\_Cloud\_Storage\_Hosting\_Avg\_us\_avg\_mbps", col("Web\_Cloud\_Storage\_Hosting\_Avg\_us\_avg\_mbps") - when((lag("Web\_Cloud\_Storage\_Hosting\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Web\_Cloud\_Storage\_Hosting\_Sum\_ds\_mbytes", 1).over(windowSpec)))

\*/

.withColumn("Diff\_Streaming\_Media\_Sum\_ds\_mbytes", col("Streaming\_Media\_Sum\_ds\_mbytes") - when((lag("Streaming\_Media\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Streaming\_Media\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Streaming\_Media\_Sum\_us\_mbytes", col("Streaming\_Media\_Sum\_us\_mbytes") - when((lag("Streaming\_Media\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Streaming\_Media\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Streaming\_Media\_Sum\_ds\_mmwt", col("Streaming\_Media\_Sum\_ds\_mmwt") - when((lag("Streaming\_Media\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Streaming\_Media\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Streaming\_Media\_Sum\_us\_mmwt", col("Streaming\_Media\_Sum\_us\_mmwt") - when((lag("Streaming\_Media\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Streaming\_Media\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Streaming\_Media\_Max\_ds\_max\_mbps", col("Streaming\_Media\_Max\_ds\_max\_mbps") - when((lag("Streaming\_Media\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Streaming\_Media\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Streaming\_Media\_Max\_us\_max\_mbps", col("Streaming\_Media\_Max\_us\_max\_mbps") - when((lag("Streaming\_Media\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Streaming\_Media\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Streaming\_Media\_Avg\_ds\_avg\_mbps", col("Streaming\_Media\_Avg\_ds\_avg\_mbps") - when((lag("Streaming\_Media\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Streaming\_Media\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Streaming\_Media\_Avg\_us\_avg\_mbps", col("Streaming\_Media\_Avg\_us\_avg\_mbps") - when((lag("Streaming\_Media\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Streaming\_Media\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Messaging\_and\_Collaboration\_Sum\_ds\_mbytes", col("Messaging\_and\_Collaboration\_Sum\_ds\_mbytes") - when((lag("Messaging\_and\_Collaboration\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Messaging\_and\_Collaboration\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Messaging\_and\_Collaboration\_Sum\_us\_mbytes", col("Messaging\_and\_Collaboration\_Sum\_us\_mbytes") - when((lag("Messaging\_and\_Collaboration\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Messaging\_and\_Collaboration\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Messaging\_and\_Collaboration\_Sum\_ds\_mmwt", col("Messaging\_and\_Collaboration\_Sum\_ds\_mmwt") - when((lag("Messaging\_and\_Collaboration\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Messaging\_and\_Collaboration\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Messaging\_and\_Collaboration\_Sum\_us\_mmwt", col("Messaging\_and\_Collaboration\_Sum\_us\_mmwt") - when((lag("Messaging\_and\_Collaboration\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Messaging\_and\_Collaboration\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Messaging\_and\_Collaboration\_Max\_ds\_max\_mbps", col("Messaging\_and\_Collaboration\_Max\_ds\_max\_mbps") - when((lag("Messaging\_and\_Collaboration\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Messaging\_and\_Collaboration\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Messaging\_and\_Collaboration\_Max\_us\_max\_mbps", col("Messaging\_and\_Collaboration\_Max\_us\_max\_mbps") - when((lag("Messaging\_and\_Collaboration\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Messaging\_and\_Collaboration\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Messaging\_and\_Collaboration\_Avg\_ds\_avg\_mbps", col("Messaging\_and\_Collaboration\_Avg\_ds\_avg\_mbps") - when((lag("Messaging\_and\_Collaboration\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Messaging\_and\_Collaboration\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Messaging\_and\_Collaboration\_Avg\_us\_avg\_mbps", col("Messaging\_and\_Collaboration\_Avg\_us\_avg\_mbps") - when((lag("Messaging\_and\_Collaboration\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Messaging\_and\_Collaboration\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Apple\_Sum\_ds\_mbytes", col("Apple\_Sum\_ds\_mbytes") - when((lag("Apple\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Apple\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Apple\_Sum\_us\_mbytes", col("Apple\_Sum\_us\_mbytes") - when((lag("Apple\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Apple\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Apple\_Sum\_ds\_mmwt", col("Apple\_Sum\_ds\_mmwt") - when((lag("Apple\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Apple\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Apple\_Sum\_us\_mmwt", col("Apple\_Sum\_us\_mmwt") - when((lag("Apple\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Apple\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Apple\_Max\_ds\_max\_mbps", col("Apple\_Max\_ds\_max\_mbps") - when((lag("Apple\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Apple\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Apple\_Max\_us\_max\_mbps", col("Apple\_Max\_us\_max\_mbps") - when((lag("Apple\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Apple\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Apple\_Avg\_ds\_avg\_mbps", col("Apple\_Avg\_ds\_avg\_mbps") - when((lag("Apple\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Apple\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Apple\_Avg\_us\_avg\_mbps", col("Apple\_Avg\_us\_avg\_mbps") - when((lag("Apple\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Apple\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Gaming\_Sum\_ds\_mbytes", col("Gaming\_Sum\_ds\_mbytes") - when((lag("Gaming\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Gaming\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Gaming\_Sum\_us\_mbytes", col("Gaming\_Sum\_us\_mbytes") - when((lag("Gaming\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Gaming\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Gaming\_Sum\_ds\_mmwt", col("Gaming\_Sum\_ds\_mmwt") - when((lag("Gaming\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Gaming\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Gaming\_Sum\_us\_mmwt", col("Gaming\_Sum\_us\_mmwt") - when((lag("Gaming\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Gaming\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Gaming\_Max\_ds\_max\_mbps", col("Gaming\_Max\_ds\_max\_mbps") - when((lag("Gaming\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Gaming\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Gaming\_Max\_us\_max\_mbps", col("Gaming\_Max\_us\_max\_mbps") - when((lag("Gaming\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Gaming\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Gaming\_Avg\_ds\_avg\_mbps", col("Gaming\_Avg\_ds\_avg\_mbps") - when((lag("Gaming\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Gaming\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Gaming\_Avg\_us\_avg\_mbps", col("Gaming\_Avg\_us\_avg\_mbps") - when((lag("Gaming\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Gaming\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_File\_Sharing\_Transfer\_Sum\_ds\_mbytes", col("File\_Sharing\_Transfer\_Sum\_ds\_mbytes") - when((lag("File\_Sharing\_Transfer\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("File\_Sharing\_Transfer\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_File\_Sharing\_Transfer\_Sum\_us\_mbytes", col("File\_Sharing\_Transfer\_Sum\_us\_mbytes") - when((lag("File\_Sharing\_Transfer\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("File\_Sharing\_Transfer\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_File\_Sharing\_Transfer\_Sum\_ds\_mmwt", col("File\_Sharing\_Transfer\_Sum\_ds\_mmwt") - when((lag("File\_Sharing\_Transfer\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("File\_Sharing\_Transfer\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_File\_Sharing\_Transfer\_Sum\_us\_mmwt", col("File\_Sharing\_Transfer\_Sum\_us\_mmwt") - when((lag("File\_Sharing\_Transfer\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("File\_Sharing\_Transfer\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_File\_Sharing\_Transfer\_Max\_ds\_max\_mbps", col("File\_Sharing\_Transfer\_Max\_ds\_max\_mbps") - when((lag("File\_Sharing\_Transfer\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("File\_Sharing\_Transfer\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_File\_Sharing\_Transfer\_Max\_us\_max\_mbps", col("File\_Sharing\_Transfer\_Max\_us\_max\_mbps") - when((lag("File\_Sharing\_Transfer\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("File\_Sharing\_Transfer\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_File\_Sharing\_Transfer\_Avg\_ds\_avg\_mbps", col("File\_Sharing\_Transfer\_Avg\_ds\_avg\_mbps") - when((lag("File\_Sharing\_Transfer\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("File\_Sharing\_Transfer\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_File\_Sharing\_Transfer\_Avg\_us\_avg\_mbps", col("File\_Sharing\_Transfer\_Avg\_us\_avg\_mbps") - when((lag("File\_Sharing\_Transfer\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("File\_Sharing\_Transfer\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Business\_Sum\_ds\_mbytes", col("Business\_Sum\_ds\_mbytes") - when((lag("Business\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Business\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Business\_Sum\_us\_mbytes", col("Business\_Sum\_us\_mbytes") - when((lag("Business\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Business\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Business\_Sum\_ds\_mmwt", col("Business\_Sum\_ds\_mmwt") - when((lag("Business\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Business\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Business\_Sum\_us\_mmwt", col("Business\_Sum\_us\_mmwt") - when((lag("Business\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Business\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Business\_Max\_ds\_max\_mbps", col("Business\_Max\_ds\_max\_mbps") - when((lag("Business\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Business\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Business\_Max\_us\_max\_mbps", col("Business\_Max\_us\_max\_mbps") - when((lag("Business\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Business\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Business\_Avg\_ds\_avg\_mbps", col("Business\_Avg\_ds\_avg\_mbps") - when((lag("Business\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Business\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Business\_Avg\_us\_avg\_mbps", col("Business\_Avg\_us\_avg\_mbps") - when((lag("Business\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Business\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Social\_Sum\_ds\_mbytes", col("Social\_Sum\_ds\_mbytes") - when((lag("Social\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Social\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Social\_Sum\_us\_mbytes", col("Social\_Sum\_us\_mbytes") - when((lag("Social\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("Social\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_Social\_Sum\_ds\_mmwt", col("Social\_Sum\_ds\_mmwt") - when((lag("Social\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Social\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Social\_Sum\_us\_mmwt", col("Social\_Sum\_us\_mmwt") - when((lag("Social\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("Social\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_Social\_Max\_ds\_max\_mbps", col("Social\_Max\_ds\_max\_mbps") - when((lag("Social\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Social\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Social\_Max\_us\_max\_mbps", col("Social\_Max\_us\_max\_mbps") - when((lag("Social\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Social\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Social\_Avg\_ds\_avg\_mbps", col("Social\_Avg\_ds\_avg\_mbps") - when((lag("Social\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Social\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_Social\_Avg\_us\_avg\_mbps", col("Social\_Avg\_us\_avg\_mbps") - when((lag("Social\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("Social\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_VPN\_Sum\_ds\_mbytes", col("VPN\_Sum\_ds\_mbytes") - when((lag("VPN\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("VPN\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_VPN\_Sum\_us\_mbytes", col("VPN\_Sum\_us\_mbytes") - when((lag("VPN\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("VPN\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_VPN\_Sum\_ds\_mmwt", col("VPN\_Sum\_ds\_mmwt") - when((lag("VPN\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("VPN\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_VPN\_Sum\_us\_mmwt", col("VPN\_Sum\_us\_mmwt") - when((lag("VPN\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("VPN\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_VPN\_Max\_ds\_max\_mbps", col("VPN\_Max\_ds\_max\_mbps") - when((lag("VPN\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("VPN\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_VPN\_Max\_us\_max\_mbps", col("VPN\_Max\_us\_max\_mbps") - when((lag("VPN\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("VPN\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_VPN\_Avg\_ds\_avg\_mbps", col("VPN\_Avg\_ds\_avg\_mbps") - when((lag("VPN\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("VPN\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_VPN\_Avg\_us\_avg\_mbps", col("VPN\_Avg\_us\_avg\_mbps") - when((lag("VPN\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("VPN\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_SPEED\_TEST\_Sum\_ds\_mbytes", col("SPEED\_TEST\_Sum\_ds\_mbytes") - when((lag("SPEED\_TEST\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("SPEED\_TEST\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_SPEED\_TEST\_Sum\_us\_mbytes", col("SPEED\_TEST\_Sum\_us\_mbytes") - when((lag("SPEED\_TEST\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("SPEED\_TEST\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_SPEED\_TEST\_Sum\_ds\_mmwt", col("SPEED\_TEST\_Sum\_ds\_mmwt") - when((lag("SPEED\_TEST\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("SPEED\_TEST\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_SPEED\_TEST\_Sum\_us\_mmwt", col("SPEED\_TEST\_Sum\_us\_mmwt") - when((lag("SPEED\_TEST\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("SPEED\_TEST\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_SPEED\_TEST\_Max\_ds\_max\_mbps", col("SPEED\_TEST\_Max\_ds\_max\_mbps") - when((lag("SPEED\_TEST\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("SPEED\_TEST\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_SPEED\_TEST\_Max\_us\_max\_mbps", col("SPEED\_TEST\_Max\_us\_max\_mbps") - when((lag("SPEED\_TEST\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("SPEED\_TEST\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_SPEED\_TEST\_Avg\_ds\_avg\_mbps", col("SPEED\_TEST\_Avg\_ds\_avg\_mbps") - when((lag("SPEED\_TEST\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("SPEED\_TEST\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_SPEED\_TEST\_Avg\_us\_avg\_mbps", col("SPEED\_TEST\_Avg\_us\_avg\_mbps") - when((lag("SPEED\_TEST\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("SPEED\_TEST\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_COMPETITOR\_Sum\_ds\_mbytes", col("COMPETITOR\_Sum\_ds\_mbytes") - when((lag("COMPETITOR\_Sum\_ds\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("COMPETITOR\_Sum\_ds\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_COMPETITOR\_Sum\_us\_mbytes", col("COMPETITOR\_Sum\_us\_mbytes") - when((lag("COMPETITOR\_Sum\_us\_mbytes", 1).over(windowSpec)).isNull, 0).otherwise(lag("COMPETITOR\_Sum\_us\_mbytes", 1).over(windowSpec)))

.withColumn("Diff\_COMPETITOR\_Sum\_ds\_mmwt", col("COMPETITOR\_Sum\_ds\_mmwt") - when((lag("COMPETITOR\_Sum\_ds\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("COMPETITOR\_Sum\_ds\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_COMPETITOR\_Sum\_us\_mmwt", col("COMPETITOR\_Sum\_us\_mmwt") - when((lag("COMPETITOR\_Sum\_us\_mmwt", 1).over(windowSpec)).isNull, 0).otherwise(lag("COMPETITOR\_Sum\_us\_mmwt", 1).over(windowSpec)))

.withColumn("Diff\_COMPETITOR\_Max\_ds\_max\_mbps", col("COMPETITOR\_Max\_ds\_max\_mbps") - when((lag("COMPETITOR\_Max\_ds\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("COMPETITOR\_Max\_ds\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_COMPETITOR\_Max\_us\_max\_mbps", col("COMPETITOR\_Max\_us\_max\_mbps") - when((lag("COMPETITOR\_Max\_us\_max\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("COMPETITOR\_Max\_us\_max\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_COMPETITOR\_Avg\_ds\_avg\_mbps", col("COMPETITOR\_Avg\_ds\_avg\_mbps") - when((lag("COMPETITOR\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("COMPETITOR\_Avg\_ds\_avg\_mbps", 1).over(windowSpec)))

.withColumn("Diff\_COMPETITOR\_Avg\_us\_avg\_mbps", col("COMPETITOR\_Avg\_us\_avg\_mbps") - when((lag("COMPETITOR\_Avg\_us\_avg\_mbps", 1).over(windowSpec)).isNull, 0).otherwise(lag("COMPETITOR\_Avg\_us\_avg\_mbps", 1).over(windowSpec)))

val lcase = DiffCol.toDF(DiffCol.columns map(\_.toLowerCase):\_\*)

//lcase.show()

val lcaseAfterDrop=lcase.drop("other\_sum\_ds\_mbytes").drop("other\_sum\_us\_mbytes").drop("other\_sum\_ds\_mmwt").drop("other\_sum\_us\_mmwt").drop("other\_max\_ds\_max\_mbps").drop("other\_max\_us\_max\_mbps").drop("other\_avg\_ds\_avg\_mbps").drop("other\_avg\_us\_avg\_mbps")

//lcaseAfterDrop.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").save("C://output\_files1//pivot-delta-4Jan.csv")

lcaseAfterDrop.createOrReplaceTempView("pivotdeltaview")

val delta = spark.sql("SELECT \* FROM pivotdeltaview")

delta.write

.format("org.apache.spark.sql.cassandra")

.mode("Append")

.options(Map( "table" -> "pivotdelta", "keyspace" -> "rcml"))

.save()

//lcase.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").save("C://output\_files1//pivot-delta-3Jan.csv")

println("Pivot-Delta Written")

/\*

DiffCol.createOrReplaceTempView("pivotdeltaview")

val delta = spark.sql("SELECT \* FROM pivotdeltaview")

delta.write

.format("org.apache.spark.sql.cassandra")

.mode("Append")

.options(Map( "table" -> "pivotdelta", "keyspace" -> "testcassandra"))

.save()

\*/

// DiffCol.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").save("C://output\_files1//pivot-delta-2Jan.csv")

// Fetching data from endpoint\_application\_months.csv

\*/

//Code to insert csv value into the cassandra database

/\*

val sparkCSV = SparkSession.builder().master("local").appName("ValidationFrameWork").getOrCreate()

val Subdf = sparkCSV.read.format("csv")

.option("header", "true")

.option("inferSchema","true")

.load("Data//subscriber-subset.csv")

//Subdf.show()

Subdf.createOrReplaceTempView("SBData")

val delta = sparkCSV.sql("SELECT \* FROM SBData")

delta.write

.format("org.apache.spark.sql.cassandra")

.mode("Append")

.options(Map( "table" -> "subscriber\_billing", "keyspace" -> "rcml"))

.save()

println("Values inserted in the cassandra database")

\*/

// Now access the Subscriber\_Billing table values from Cassandra database

val spark = SparkSession

.builder()

.master("local")

.appName("Spark\_Cassandra")

.config("spark.cassandra.connection.host", "127.0.0.1")

.getOrCreate()

val Subdf = spark

.read

.format("org.apache.spark.sql.cassandra")

.options(Map( "table" -> "subscriber\_billing", "keyspace" -> "rcml"))

.load()

//Subdf.show()

Subdf.createOrReplaceTempView("Billing")

val SubBilling = spark.sql("SELECT \* FROM Billing")

//SubBilling.show()

val SBYearMonth = SubBilling.withColumn("year", year(col("date\_time"))).withColumn("month", month(col("date\_time")))

//SBYearMonth.show()

SBYearMonth.createOrReplaceTempView("subsetSB")

val statusDF = spark.sql("SELECT \* from subsetSB where status in (4, 3)")

// x.show()

/\*

//val windowSpec = Window.partitionBy("endpoint\_id").orderBy("month")

val ranked=statusDF.withColumn("rank",rank().over(Window.partitionBy("endpoint\_id").orderBy(col("date\_time").desc)))

ranked.show()

// val windowSpec = Window.partitionBy("endpoint\_id").orderBy("month")

ranked.createOrReplaceTempView("A")

/\*

val R1 = spark.sql("select count(endpoint\_id), endpoint\_id, date\_time, status, current, rank from A group by endpoint\_id, date\_time, status, rank, current having status=4 and current=0")

R1.createOrReplaceTempView("B")

val R2 = spark.sql("select A.\* from A, B where A.endpoint\_id = B.endpoint\_id order by endpoint\_id, date\_time desc")

R2.show(1000)

\*/

val R1 = spark.sql("select A.\* from A order by org\_id, endpoint\_id, date\_time asc")

R1.show()

\*/

val R2=statusDF.withColumn("timediff",rank().over(Window.partitionBy("endpoint\_id").orderBy(col("date\_time").asc)))

val windowSpec = Window.partitionBy("org\_id","endpoint\_id").orderBy("date\_time")

val SDiffCol = R2.withColumn("diff-month", col("month") - when((lag("month", 1).over(windowSpec)).isNull, 0).otherwise(lag("month", 1).over(windowSpec)))

//val ordered = spark.sql("select A.\* from DiffCol, B where A.endpoint\_id = B.endpoint\_id order by endpoint\_id, date\_time desc")

// val order =

SDiffCol.show()

SDiffCol.createOrReplaceTempView("SB1")

val R3= spark.sql("SELECT \* from SB1 where status = 4 and current =0")

R3.show()

val R4= spark.sql("SELECT \* from SB1 where status = 3")

R4.show()

//DiffCol.select("status").show()

//val R2 = spark.sql("select count(endpoint\_id), endpoint\_id, date\_time, status, current, rank from A group by endpoint\_id, date\_time, status, rank, current having status=4 and current=0")

//R1.createOrReplaceTempView("B")

//val R3 = spark.sql("select A.\* from A, B where A.endpoint\_id = B.endpoint\_id order by endpoint\_id, date\_time desc")

//R3.show()

// DiffCol.repartition(1).write.format("com.databricks.spark.csv").option("header", "true").save("C://output\_files1//churn-delta-8Jan.csv")

println("Done")

//spark.stop()

}

}