import org.apache.spark.SparkConf  
import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.types.{DateType, IntegerType, StringType, StructField, StructType}  
import org.apache.spark.sql.functions.{*col*, *explode*, *split*, *lower*, regexp\_replace}  
  
object dataframe extends App {  
 val *sparkconf*=new SparkConf()  
 *sparkconf*.set("spark.app.name","firstDFDemo")  
 *sparkconf*.set("spark.master","local[1]")  
 val *spark* = SparkSession.*builder*().config(*sparkconf*).getOrCreate()  
  
 val *DF* = *spark*.read.text("D:\\hi.txt")  
  
 val *wordsDF* = *DF* .select(*explode*(*split*(*regexp\_replace*(*lower*(*col*("value")), "[^a-zA-Z0-9\\s]", ""), "\\s+")).alias("word"))  
 .filter(*col*("word") =!= "")  
  
 val *wordCountsDF* = *wordsDF*.groupBy("word").count().sort(*col*("count").desc)  
  
 *wordCountsDF*.show()  
  
}

Output:

+-----+-----+

| word|count|

+-----+-----+

|hello| 2|

| hi| 2|

| how| 2|

| you| 1|

| is| 1|

| are| 1|

| big| 1|

| data| 1|

+-----+-----+

import org.apache.spark.SparkConf  
import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.expressions.Window  
import org.apache.spark.sql.functions.{*col*, count, countDistinct, *desc*, first, *rank*, sum, to\_date}  
import org.apache.spark.sql.types.{DateType, IntegerType, StringType, StructField, StructType}  
  
object dataframe extends App{  
  
 val *sparkconf*=new SparkConf()  
 *sparkconf*.set("spark.app.name","firstDFDemo")  
 *sparkconf*.set("spark.master","local[1]")  
 val *spark*=SparkSession.*builder*().config(*sparkconf*).getOrCreate()  
  
  
 val *salesSchema*="customer\_id string, order\_date date, product\_id int"  
 var *salesdf*=*spark*.read.option("header",true).schema(*salesSchema*).csv(path="D:\\sales.csv")  
  
 val *membersSchema*="customer\_id string, join\_date date"  
 var *membersdf*=*spark*.read.option("header",true).schema(*membersSchema*).csv(path="D:\\members.csv")  
  
 val *menuSchema*="product\_id int, product\_name string,price int"  
 var *menudf*=*spark*.read.option("header",true).schema(*menuSchema*).csv(path="D:\\menu.csv")  
  
 // 1. Total amount each customer spent at the restaurant  
 val *amountSpentPerCustomerDF* = *salesdf* .join(*menudf*, *salesdf*("product\_id") === *menudf*("product\_id"))  
 .groupBy("customer\_id")  
 .agg(*sum*("price").alias("total\_amount\_spent"))  
  
 *amountSpentPerCustomerDF*.show()  
  
 // 2. Number of days each customer visited the restaurant  
 val *daysVisitedPerCustomerDF* = *salesdf* .groupBy("customer\_id")  
 .agg(*countDistinct*("order\_date").alias("days\_visited"))  
  
 *daysVisitedPerCustomerDF*.show()  
  
 // 3. First item purchased from the menu by each customer  
 val *firstItemPerCustomerDF* = *salesdf* .join(*menudf*, *salesdf*("product\_id") === *menudf*("product\_id"))  
 .withColumn("rank\_order", *rank*().over(Window.*partitionBy*("customer\_id").orderBy("order\_date")))  
 .filter(*col*("rank\_order") === 1)  
 .groupBy("customer\_id")  
 .agg(*first*("product\_name").alias("first\_item\_purchased"))  
  
 *firstItemPerCustomerDF*.show()  
  
 // 4. Most purchased item on the menu and its count  
 val *mostPurchasedItemDF* = *salesdf* .join(*menudf*, *salesdf*("product\_id") === *menudf*("product\_id"))  
 .groupBy("product\_name")  
 .agg(*count*("product\_id").alias("purchase\_count"))  
 .orderBy(*desc*("purchase\_count"))  
 .limit(1)  
  
 *mostPurchasedItemDF*.show()  
  
 // 5. Most popular item for each customer  
 val *mostPopularItemPerCustomerDF* = *salesdf* .join(*menudf*, *salesdf*("product\_id") === *menudf*("product\_id"))  
 .groupBy("customer\_id", "product\_name")  
 .agg(*count*("product\_id").alias("order\_count"))  
 .withColumn("rank\_order", *rank*().over(Window.*partitionBy*("customer\_id").orderBy(*desc*("order\_count"))))  
 .filter(*col*("rank\_order") === 1)  
  
 *mostPopularItemPerCustomerDF*.show()  
  
 // 6. First item purchased by each customer after becoming a member  
 val *firstItemAfterJoinDF* = *salesdf* .join(*menudf*, *salesdf*("product\_id") === *menudf*("product\_id"))  
 .join(*membersdf*, *salesdf*("customer\_id") === *membersdf*("customer\_id") && *salesdf*("order\_date") >= *membersdf*("join\_date"))  
 .withColumn("rank\_order", *rank*().over(Window.*partitionBy*("customer\_id").orderBy("order\_date")))  
 .filter(*col*("rank\_order") === 1)  
 .groupBy("customer\_id")  
 .agg(*first*("product\_name").alias("first\_item\_after\_join"))  
  
 *firstItemAfterJoinDF*.show()  
  
  
  
  
}

+-----------+------------------+

|customer\_id|total\_amount\_spent|

+-----------+------------------+

| B| 74|

| C| 36|

| A| 76|

+-----------+------------------+

+-----------+------------+

|customer\_id|days\_visited|

+-----------+------------+

| B| 6|

| C| 2|

| A| 4|

+-----------+------------+

+-----------+--------------------+

|customer\_id|first\_item\_purchased|

+-----------+--------------------+

| B| curry|

| C| ramen|

| A| sushi|

+-----------+--------------------+

+------------+--------------+

|product\_name|purchase\_count|

+------------+--------------+

| ramen| 8|

+------------+--------------+

+-----------+------------+-----------+----------+

|customer\_id|product\_name|order\_count|rank\_order|

+-----------+------------+-----------+----------+

| B| sushi| 2| 1|

| B| ramen| 2| 1|

| B| curry| 2| 1|

| C| ramen| 3| 1|

| A| ramen| 3| 1|

+-----------+------------+-----------+----------+

+-----------+---------------------+

|customer\_id|first\_item\_after\_join|

+-----------+---------------------+

| B| sushi|

| A| curry|

+-----------+---------------------+

import org.apache.spark.SparkConf  
import org.apache.spark.sql.SparkSession  
import org.apache.spark.sql.expressions.Window  
import org.apache.spark.sql.functions.{*col*, count, countDistinct, *desc*, first, *rank*, sum, to\_date}  
import org.apache.spark.sql.types.{DateType, IntegerType, StringType, StructField, StructType}  
  
object sample extends App{  
  
 val *sparkconf*=new SparkConf()  
 *sparkconf*.set("spark.app.name","firstDFDemo")  
 *sparkconf*.set("spark.master","local[1]")  
 val *spark*=SparkSession.*builder*().config(*sparkconf*).getOrCreate()  
  
  
 val *salesSchema*="customer\_id string, order\_date string, product\_id int,amount int"  
 var *salesdf*=*spark*.read.option("header",true).option("dateFormat", "dd-MM-yyyy").schema(*salesSchema*).csv(path="D:\\sales1.csv")  
  
 var *salesnewdf*=*salesdf*.withColumn("newdate",*to\_date*(*col*("order\_date"),"dd-MM-yyyy"))  
  
 // Define window specification  
 val *windowSpec* = Window.*partitionBy*("customer\_id").orderBy("newdate")  
  
 // Calculate cumulative sum of orders for each customer  
 val *dfWithCumulativeOrders* = *salesnewdf*.withColumn("cumulative\_orders", *sum*("amount").over(*windowSpec*))  
  
 // Calculate rank of each order within each customer's order history  
 val *dfWithOrderRank* = *dfWithCumulativeOrders*.withColumn("order\_rank", *rank*().over(*windowSpec*))  
  
 *dfWithOrderRank*.show()  
  
  
}

+-----------+----------+----------+------+----------+-----------------+----------+

|customer\_id|order\_date|product\_id|amount| newdate|cumulative\_orders|order\_rank|

+-----------+----------+----------+------+----------+-----------------+----------+

| B|01-01-2021| 2| 56|2021-01-01| 56| 1|

| B|02-01-2021| 2| 87|2021-01-02| 143| 2|

| B|04-01-2021| 1| 32|2021-01-04| 175| 3|

| B|11-01-2021| 1| 32|2021-01-11| 207| 4|

| B|16-01-2021| 3| 56|2021-01-16| 263| 5|

| B|01-02-2021| 3| 25|2021-02-01| 288| 6|

| C|01-01-2021| 3| 38|2021-01-01| 67| 1|

| C|01-01-2021| 3| 29|2021-01-01| 67| 1|

| C|07-01-2021| 3| 58|2021-01-07| 125| 3|

| A|01-01-2021| 1| 10|2021-01-01| 30| 1|

| A|01-01-2021| 2| 20|2021-01-01| 30| 1|

| A|07-01-2021| 2| 5|2021-01-07| 35| 3|

| A|10-01-2021| 3| 45|2021-01-10| 80| 4|

| A|11-01-2021| 3| 56|2021-01-11| 225| 5|

| A|11-01-2021| 3| 89|2021-01-11| 225| 5|

+-----------+----------+----------+------+----------+-----------------+----------+