



# Assignment

**Week14:** Apache Spark - Optimization  
Part-2

# IMPORTANT

## Self-assessment enables students to develop:

1. A sense of responsibility for their own learning and the ability & desire to continue learning,
2. Self-knowledge & capacity to assess their own performance critically & accurately, and
3. An understanding of how to apply their knowledge and abilities in different contexts.

All assignments are for self-assessment. Solutions will be released on every subsequent week. Once the solution is out, evaluate yourself.

No discussions/queries allowed on assignment questions in slack channel.

*Note: You can raise your doubts in the subsequent week once the solution is released*

TRENDYTECH 9108179578

## Solution1:

```
import org.apache.log4j.Level
import org.apache.log4j.Logger
import org.apache.spark.SparkConf
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions.broadcast
import org.apache.spark.sql.functions.col
import org.apache.spark.sql.functions.countDistinct
import org.apache.spark.sql.functions.round
import org.apache.spark.sql.functions.sum
import org.apache.spark.sql.functions.to_date
import org.apache.spark.sql.types.FloatType
import org.apache.spark.sql.types.IntegerType
import org.apache.spark.sql.types.StringType
import org.apache.spark.sql.types.StructField
import org.apache.spark.sql.types.StructType
import org.apache.spark.storage.StorageLevel
```

```
object W14_problem_1 extends App {
```

TRENDYTECH 9108179578

```
Logger.getLogger("org").setLevel(Level.ERROR)
```

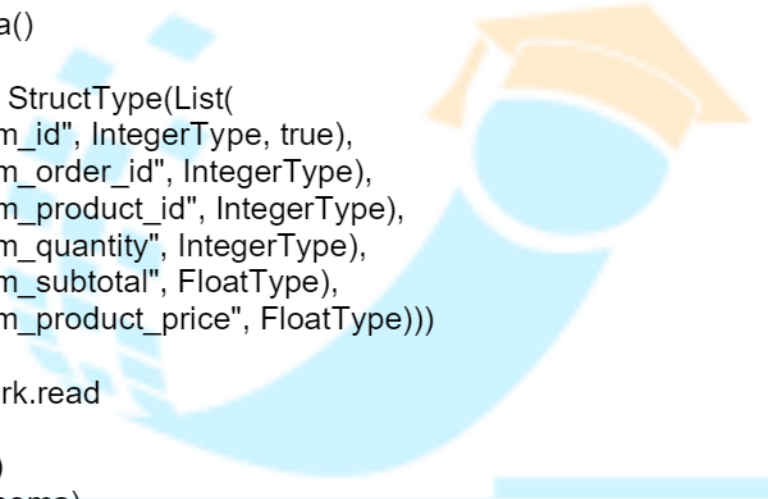
```
val sparkConf = new SparkConf()  
sparkConf.set("spark.app.name", "WEEk_14_Assignment")  
sparkConf.set("spark.master", "local[2]")
```

```
val spark = SparkSession.builder()  
  .config(sparkConf)  
  .getOrCreate()
```

```
val orderSchema = StructType(List(  
  StructField("order_id", IntegerType, true),  
  StructField("order_date", StringType),  
  StructField("order_customer_id", IntegerType),  
  StructField("order_status", StringType)))
```

```
val ordersDF = spark.read  
  .format("csv")  
  .option("header", true)  
  .schema(orderSchema)  
  .option("path", "G:/New folder/ORDERS_File.txt")  
  .load()
```

TRENDYTECH 9108179578



//ordersDF.show(false)  
//ordersDF.printSchema()  
  
val customerSchema = StructType(List(  
 StructField("order\_item\_id", IntegerType, true),  
 StructField("order\_item\_order\_id", IntegerType),  
 StructField("order\_item\_product\_id", IntegerType),  
 StructField("order\_item\_quantity", IntegerType),  
 StructField("order\_item\_subtotal", FloatType),  
 StructField("order\_item\_product\_price", FloatType)))  
  
val customersDF = spark.read  
 .format("csv")  
 .option("header", true)  
 .schema(customerSchema)  
 .option("path", "D:/New folder/ORDER\_ITEMS\_File.txt")  
 .load()  
//customersDF.show(false)  
//customersDF.printSchema()  
  
//AUTO BroadcastJoin OFF  
spark.sql("SET spark.sql.autoBroadcastJoinThreshold=-1")

TRENDYTECH 9108179578

```
//JOIN CONDITION
val joinedCondition = ordersDF.col("order_id") === customersDF.col("order_item_order_id")

//JOIN TYPE...
val joinType = "inner"

//JOIN EXPRESSION
val joinedOrderDataDF = customersDF.join(broadcast(ordersDF), joinedCondition, joinType)
    .persist(StorageLevel.MEMORY_AND_DISK_SER) //show()
```

//Now we have a joined result, lets implement the functionality using dataframe

```
val dataframeResult = joinedOrderDataDF.
    groupBy(to_date(col("order_date")).alias("order_formatted_date"), col("order_status"))
    .agg(
        round(sum("order_item_subtotal"), 2).alias("total_amount"),
        countDistinct("order_id").alias("total_orders")).
    orderBy(
        col("order_formatted_date").desc,
        col("order_status"),
```

TRENDYTECH 9108179578

```
col("total_amount").desc,  
col("total_orders"))  
  
dataFrameResult.show();  
  
//Now we have a joined result, lets implement the functionality using spark sql  
  
joinedOrderDataDF.createOrReplaceTempView("order_joined")  
  
val sqlResult = spark.sql("""select cast(to_date(order_date)as String ) as order_formatted_date,  
order_status, cast(sum(order_item_subtotal) as DECIMAL (10,2)) as total_amount,  
count(distinct(order_id)) as total_orders from order_joined  
group by to_date(order_date),  
order_status order by order_formatted_date desc,order_status,total_amount desc, total_orders""")  
//.explain() //<-- it is EXPLAIN~PLAN FOR THE QUERY //Using "HASH AGGREGATION"  
  
sqlResult.show()  
  
scala.io.StdIn.readLine()  
spark.stop()  
  
}
```

TRENDYTECH 9108179578



## Solution 2:

```
C:\Users\ASUS\Downloads\spark-2.4.4-bin-hadoop2.7\bin>spark-submit --class  
W14_problem_1 \Users\ASUS\Desktop\ W14_problem_1 _jar.jar
```

## Solution 3:

```
spark2-submit \  
--conf spark.dynamicAllocation.enabled=false \  
--deploy-mode cluster \  
--master yarn \  
--class Week_14_Q2_B \  
--num-executors 6 \  
--executor-memory 3G \  
--executor-cores 2 \  
--conf spark.ui.port=4077  
W14_problem_1 _jar.jar
```

TRENDYTECH 9108179578





**5** Star Google Rated  
Big Data Course

**LEARN FROM THE EXPERT**



**9108179578**

**Call for more details**