**SWAROOPA DEVARI:**

**Hadoop project examples:**

Read data from local file system and HDFS(Hadoop distributed file system), then apply various transformations and load the file back to HDFS. (Reading and writing data to HDFS). All datasets are in the folder named Swaroopa. This involves working on various data formats like csv, parquet, avro, json for data validations and data cleansing.

Data is related to telecommunications domain from stores channel. ETL is performed using spark. Below are two datasets (orders and orders\_items) of textfiles.

Orders.txt and orders\_items.txt:

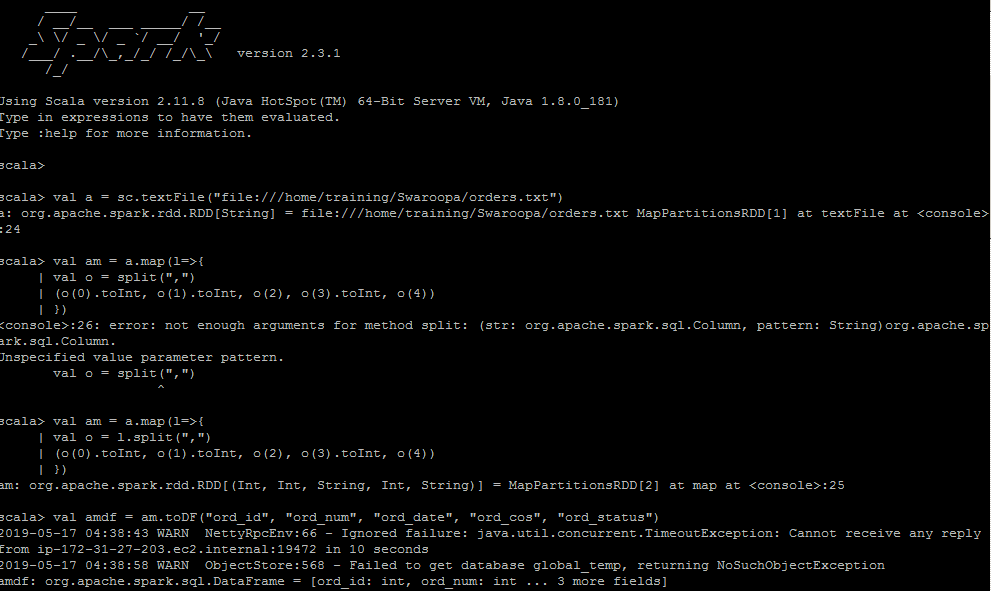


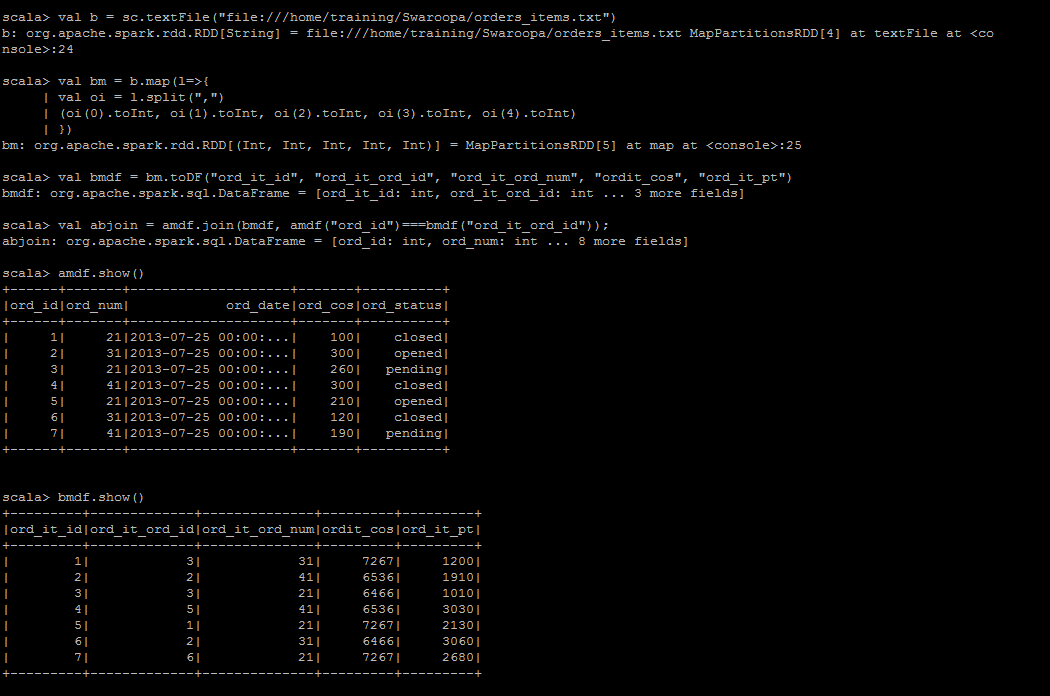
**Use Case 1: Using dataframes (spark 2.3.1, scala 2.11.8)**

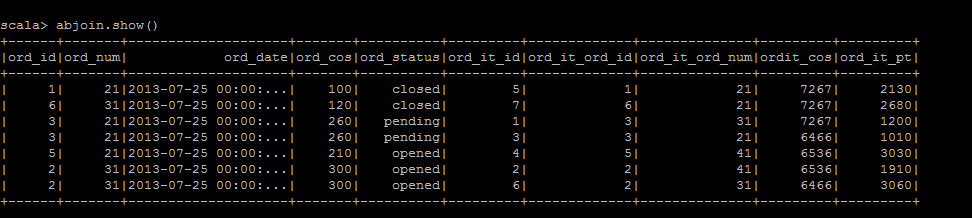
1. Using data frames find order items total cost, maximum order items cost, minimum order items cost and average order items cost for distinct order numbers per status per day (date should not have timestamp)
2. Code: using spark-shell (spark 2.3.1, scala 2.11.8)

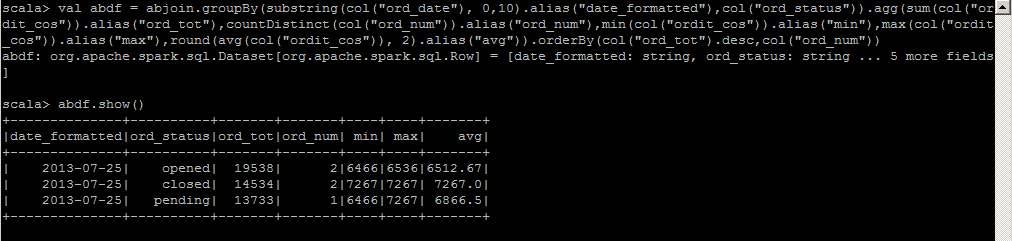
**Explanation:**

* Create rdd from external storage (sc.textFile(file path)).
* Applied map to both the datasets, then applied dataframes and gave structure to the data where we can see the data in table form and joined both the dataframes with common key ord\_id.
* Using dataframes functions like groupBy, agg, sum countDistinct, min, max, avg, col, alias, orderBy , substring, round and worked on the given usecase.







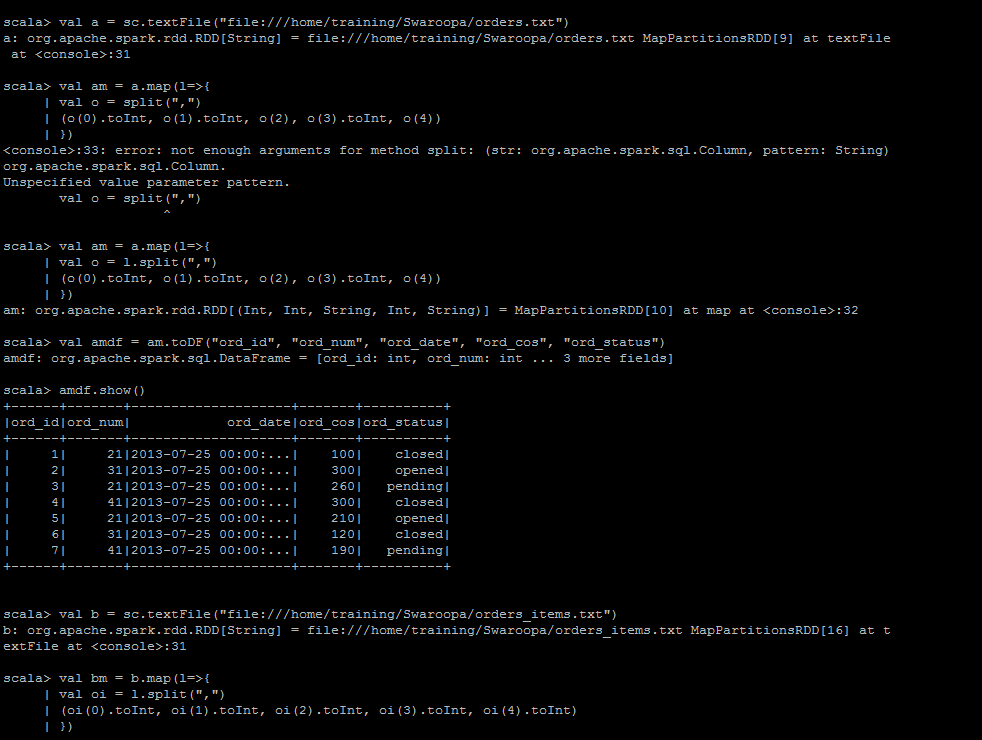


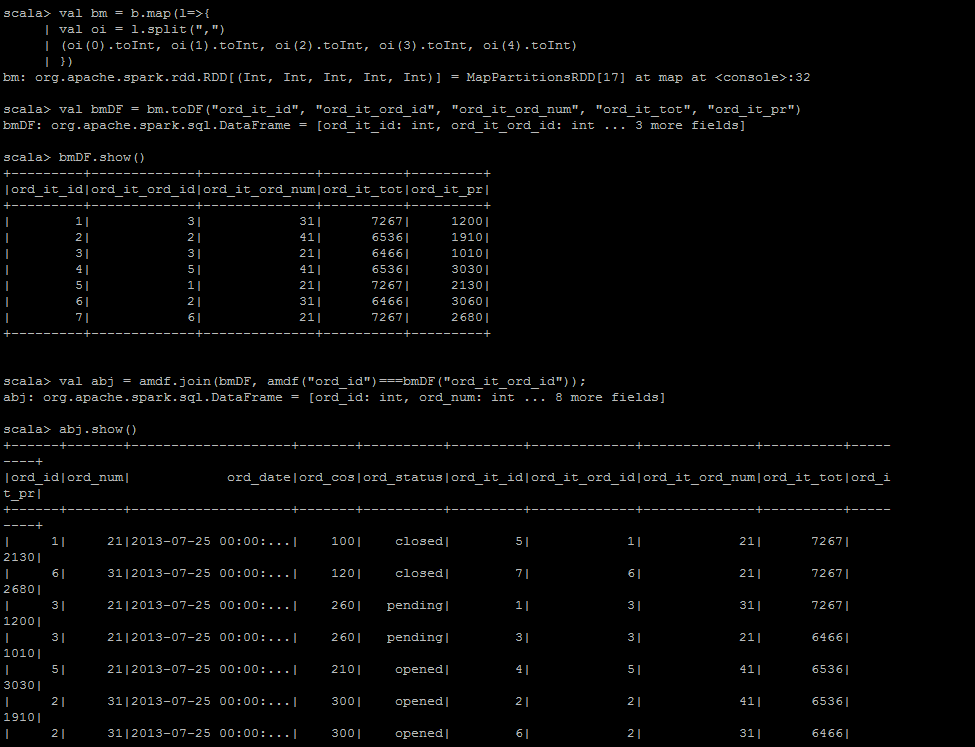
**Use Case 2: Using Spark SQL**

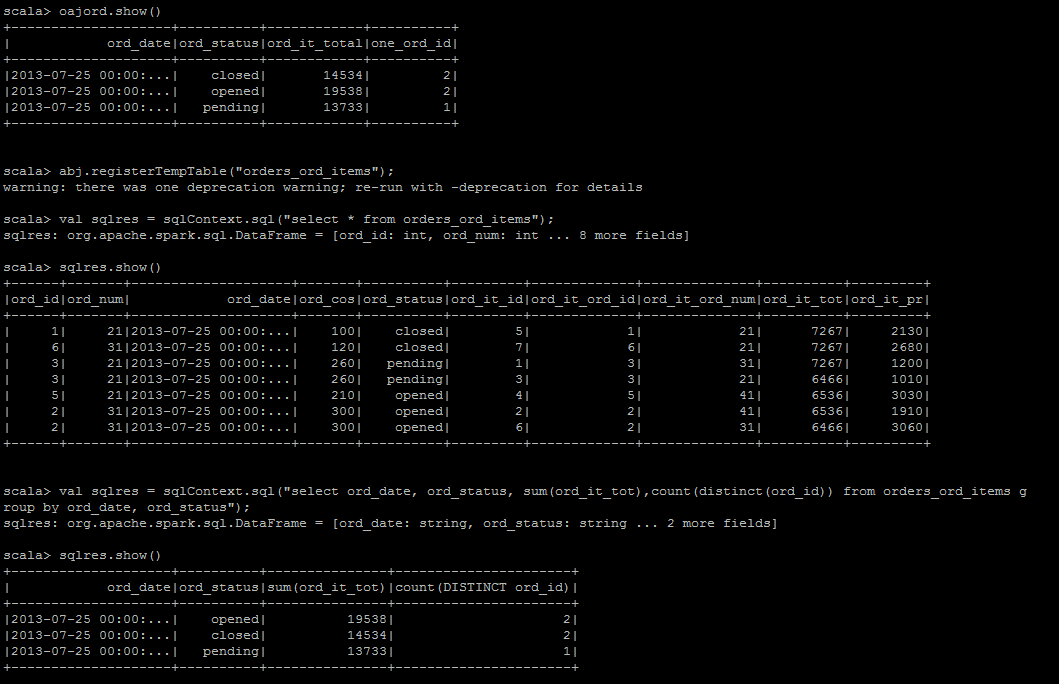
**Use spark sql for usecase1(date should be ‘dd/MM’ format)**

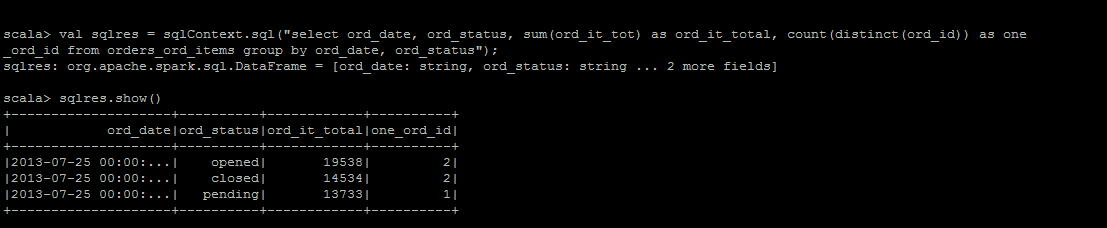
**Explanation:**

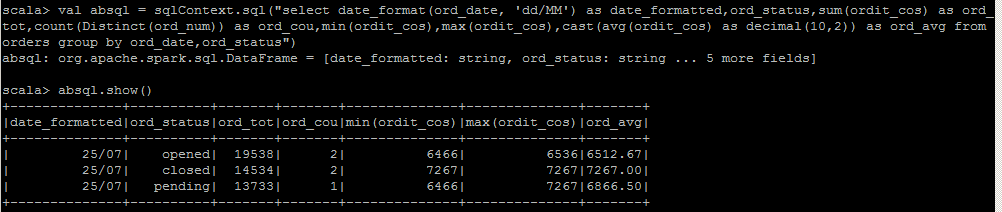
* Create rdd from external storage(sc.textFile(file path)).
* Applied map to both the above datasets, then applied dataframes and gave structure to the data where we can see the data in table form and joined both the dataframes with common key ord\_id.
* Worked on sparksql using sqlContext by registering temp table and applied various sql functions like select, date\_format, sum, count(Distinct), min, max, avg, cast, group by, order by and worked on given usecase.

****

****

****

****

****

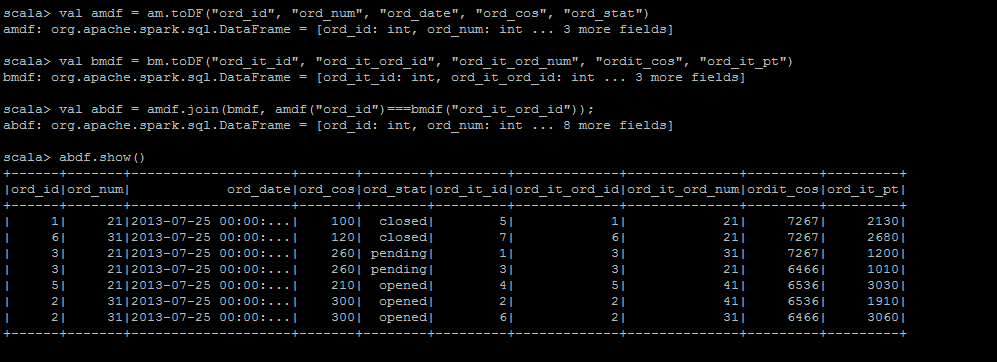
**Use Case 3: Using RDD and DataFrame**

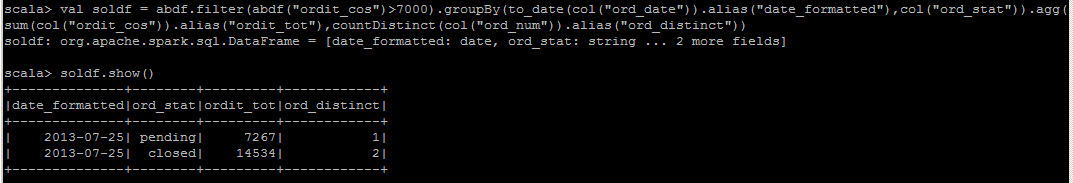
Using the below data for orders and orders\_items, find total order items cost whose cost is greater than 7000



**Explanation:**

* Create rdd from external storage(sc.textFile(file path)).
* Applied map to both the above datasets, then applied dataframes and gave structure to the data where we can see the data in table form and joined both the dataframes with common key ord\_id.
* After join, applied filter for “ordit\_cos” greater than 7000, then used functions like to\_date, col, groupBy, alias, agg, sum, countDistinct and worked on given usecase.





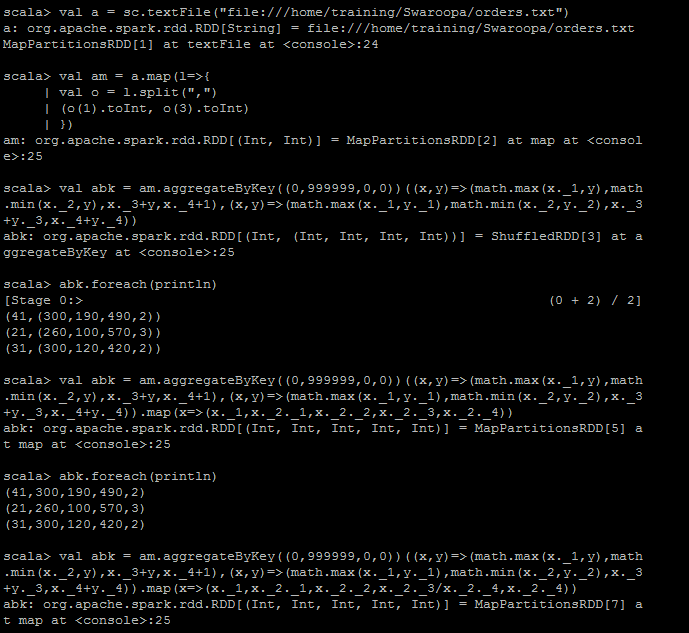
**Use Case 4: Using RDD and AggregateByKey**

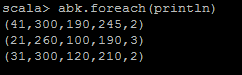
**Using AggregateByKey find max, min, avg and for distinct order numbers for “orders cos”**



**Explanation:**

* Create rdd from external storage(sc.textFile(file path)).
* Applied map to orders.txt dataset on key value pairs of ord\_num and ord\_cos, then applied aggregateByKey on key value pairs by initializing the values, applying lamda functions for math.max, math.min, finding average by summing the order cos and dividing it with distinct order\_num , then we get in tuple form.
* Applied map to bring it to rdd normal form.

****

****

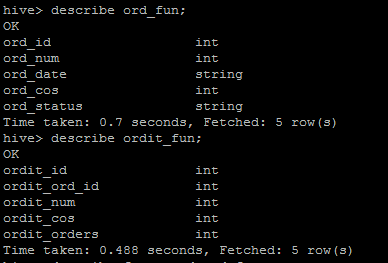
**Use Case 5: Hive Analytical Aggregations:**

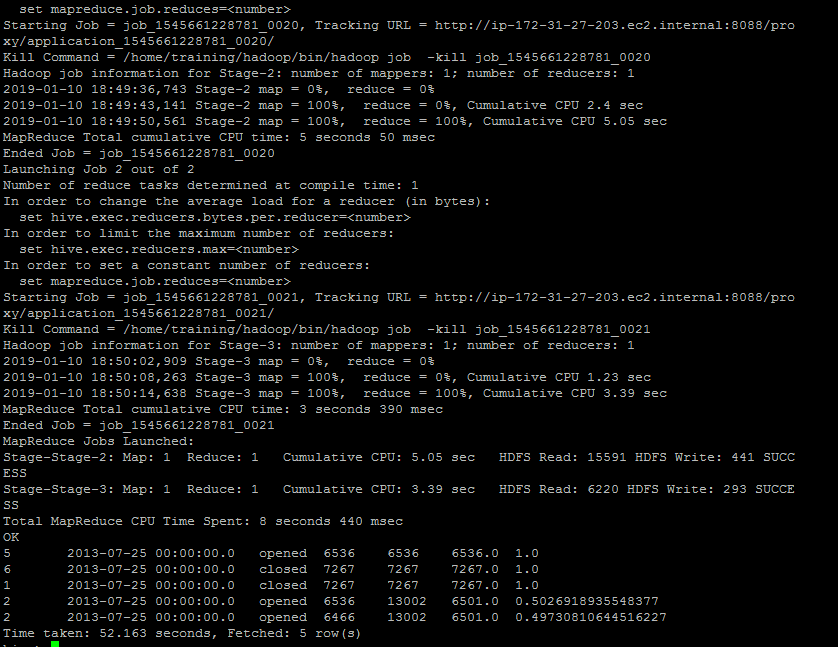
**Analytical aggregations like sum,avg, ranking and windowing functions on orders.txt and order\_items.txt dataset.**



**Explanation:**

* **Use database, Create table ord\_fun and ordit\_fun and store it as textFile**
* **Apply all the analytical functions, aggregate functions, ranking and windowing functions like below**



****

**Use Case 6: Hive - Working on AVRO, creating avro schema, schema evolution :**

****

**Use Case 7: Using SBT tool: Scala Build Tool:**

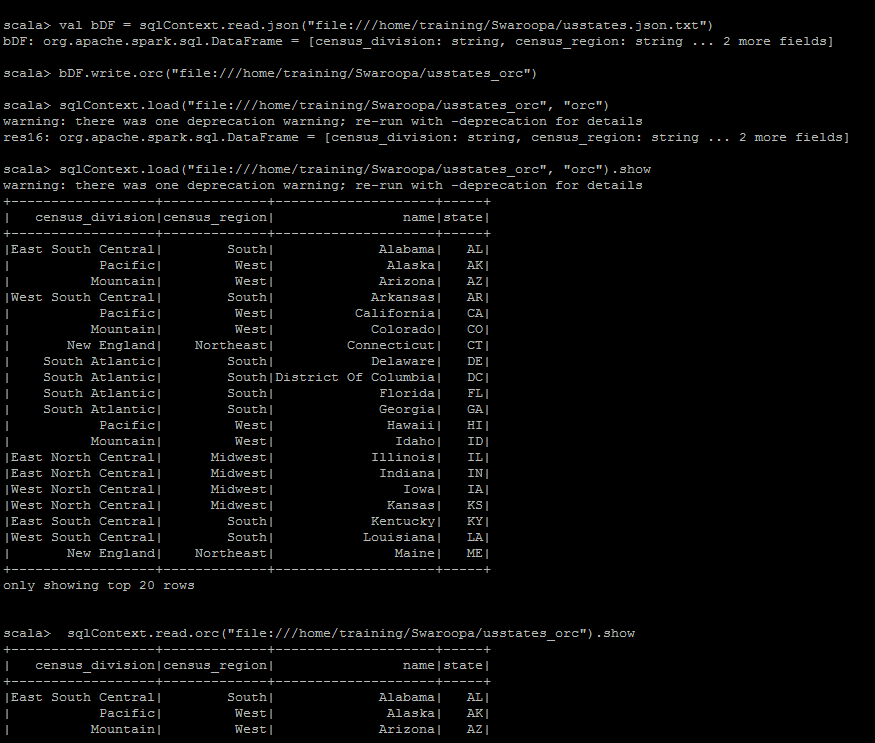
**Please view all the images from bottom to top provided in folder below:**

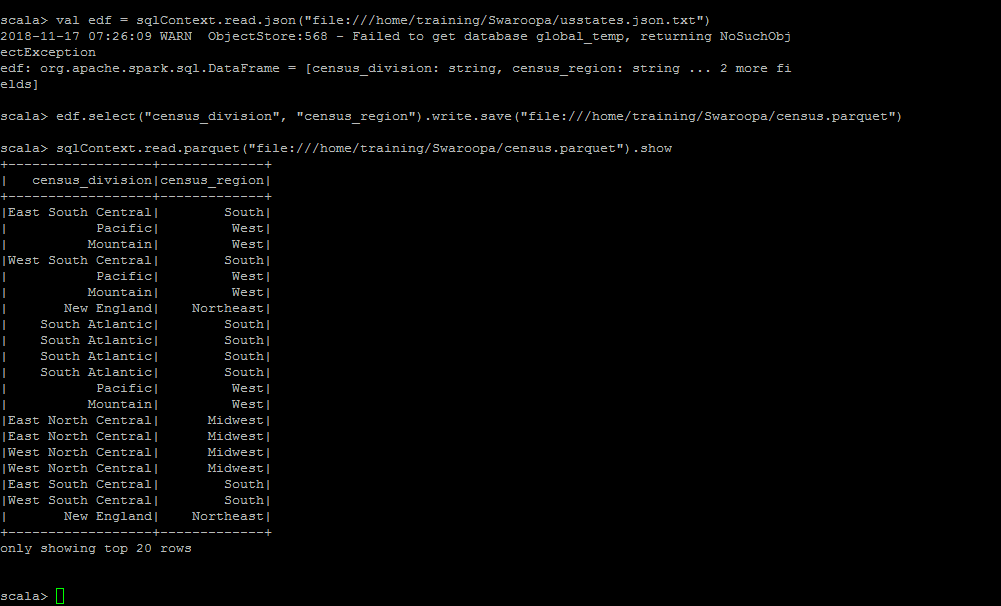
**Explanation:**

* **Create rdd from external storage, apply filter for ord\_num =21 then apply map to ord\_cos then apply reduce for total**
* **Create mkdir –p src/main/scala and vi src/main/scala/folder.scala**
* **Create SBT by giving the project version, scala version, library dependencies(maven dependencies)**
* **Create Sbt package then jar is created, then spark-submit —master yarn with jar**

****

**Use Case 8: Reading a semi structured data “JSON data” from HDFS and saving it as “Parquet and ORC” file formats**

****

****