```scala
import org.apache.spark.SparkContext

import org.apache.spark.SparkConf

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

import org.apache.spark.sql.SQLContext

import org.apache.spark.sql.hive.HiveContext

import org.apache.spark.sql.hive.orc


val ordersRDD = sc.textFile("file:///root/retail_db/orders")

import sqlContext.implicits._

case class ORDERS(order_id: Int,order_date: String,order_customer_id:
Int, order_status: String)


val ordersDF = ordersRDD.map(rec => rec.split(",")).map( rec =>
ORDERS(rec(0).toInt,rec(1).toString,rec(2).toInt,rec(3).toString)).toD
F()

case class ORDER_ITEMS(order_item_id: Int,order_item_order_id:
Int,order_item_product_id: Int,order_item_quantity:
Int,order_item_subtotal: Float,order_item_product_price: Float)

val order_itemsDF = sc.textFile("file:///root/retail_db/
order_items").map( rec => rec.split(",")).map( rec =>
ORDER_ITEMS(rec(0).toInt,rec(1).toInt,rec(2).toInt,rec(3).toInt,rec(4)
.toFloat,rec(5).toFloat)).toDF()


val ordersFilteredDF = ordersDF.filter(ordersDF("order_status") ===
"COMPLETE")

val ordersJoin = ordersFilteredDF.join(order_itemsDF,
ordersFilteredDF("order_id") === order_itemsDF("order_item_order_id"))


val s =
ordersJoinDF.groupBy(ordersJoinDF("order_date"),ordersJoinDF("order_id
")).agg(avg(ordersJoinDF("order_item_subtotal")).as("Avg_rev"))


val s1 = s.withColumn("Avg_rev", 'Avg_rev.cast("Float"))
val s1 = s.withColumn("Avg_rev", round($"Avg_rev", 2))
```

or

```
val s1 = s.withColumn("Avg_rev", round(s("Avg_rev"), 2))
```