```
import org.apache.spark.SparkContext

import org.apache.spark.SparkConf

import org.apache.spark.storage.StorageLevel

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

case class Products(product_id: Int,product_category_id: Int)

import sqlContext.implicits._

sqlContext.setConf("spark.sql.shuffle.partitions", "2")

val productsDF = sc.textFile("file:///root/retail_db/
products").map(rec => rec.split(",")).map( rec =>
Products(rec(0).toInt,rec(1).toInt)).toDF()

case class Categories(category_id: Int,category_dep_id: Int)


val categoryDF = sc.textFile("file:///root/retail_db/
categories").map(rec => rec.split(",")).map( rec =>
Categories(rec(0).toInt,rec(1).toInt)).toDF()


val prod_categ_joinDF = productsDF.join(categoryDF,
productsDF("product_category_id") === categoryDF("category_id"))


case class Departments(department_id: Int,department_name: String)

val departmentsDF = sc.textFile("file:///root/retail_db/
departments").map(rec => rec.split(",")).map( rec =>
Departments(rec(0).toInt,rec(1).toString)).toDF()


val prod_dep_joinDF = prod_categ_joinDF.join(departmentsDF,
prod_categ_joinDF("category_dep_id") ===
departmentsDF("department_id"))

val prod_dep_DF = prod_dep_joinDF.select($"product_id",
$"department_name")
```

```
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 ord_join_DF = ordersJoin.select($"order_date",
$"order_item_product_id",$"order_item_subtotal")

val avgrevJoin = ord_join_DF.join(prod_dep_DF,
ord_join_DF("order_item_product_id") === prod_dep_DF("product_id"))

val Avg_Rev_Per_Dep_DayDF = avgrevJoin.groupBy($"order_date",
$"department_name").agg(sum($"order_item_subtotal").as("Revenue")).wit
hColumn("Revenue", round($"Revenue", 2))
```