# 

# Spark Assignment

Data Analysis with Spark

### The purpose of this exercise is for you to practice using Spark to query records and retrieve information, leveraging the speed which makes Spark so popular.

### The first thing you need to do is pull in all 4 files accompanying these instructions, into your user directory on Ambari. The files should all begin with NW in the name.

### Once you feel you are ready, start a Jupyter notebook, assigning Apache Scala as the kernel engine:

### 

### Here is the first bit of code you can type in:

**import org.apache.spark.sql.SparkSession**  
**import org.apache.log4j.{Level, Logger}**  
**import org.apache.spark.sql.functions.{to\_date,month,year}**

### After you’ve added the import code to your Jupyter cell, type in this code just to check to make sure Spark is working and the files are where they should be:

### 

import org.apache.spark.sql.SparkSession

import org.apache.log4j.{Level, Logger}

import org.apache.spark.sql.functions.{to\_date,month,year}

val orderDetails = spark.read.option("header","true").

option("inferschema","true").

csv("NW-Order-Details.csv")

println("Order Details has "+orderDetails.count()+" rows" )

orderDetails.show(5)

orderDetails.printSchema()

### I named each table similarly, and you can use my code here to declare all 4 tables:

### Employees

### 

val employees = spark.read.option("header","true").

csv("NW-Employees.csv")

println("Employees has "+orderDetails.count()+" rows" )

employees.show(5)

### Orders

### 

val orders = spark.read.option("header","true").

csv("NW-Orders-01.csv")

println("Orders has "+orders.count()+" rows" )

orders.show(5)

orders.dtypes //show the data types for this table

### Products

### 

val products = spark.read.option("header","true").

option("inferSchema","true").

csv("NW-Products.csv")

println("Products has "+products.count()+" rows" )

products.show(5)

### Questions to Answer

### How many orders were placed by each customer?

### How many orders were placed in each country?

### How many orders were placed per month? per year?

### 4. What are the Total Sales for each customer, by year?

### 5. What is the average order by customer, by year?

### First 2 are easy - let us answer them first

### 

val orderByCustomer = orders.groupBy("CustomerID").count()

orderByCustomer.sort(orderByCustomer("count").desc).show(5)

Graphical user interface, text, application

Description automatically generated

### (you write it).

val orderByShipCountry = orders.groupBy("ShipCountry").count()

orderByShipCountry.sort(orderByShipCountry("count").desc).show(5)

### Graphical user interface, text, application Description automatically generated

### For the next set of questions, let us transform the data (we walk you through this below)

### 1. Add OrderTotal column to the Orders DataFrame

### 1.1. Add Line total to order details

### 1.2. Aggregate total by order id

### 1.3. Join order details & orders to add the order total

### 1.4. Check if there are any null columns

### 2. Add a date column

### 3. Add month and year

### We will guide you through this. You will need to type the following in to Jupyter:

### 

### 

// add line total to order details

val orderDetails1 = orderDetails.select(orderDetails("OrderID"),

(

(orderDetails("UnitPrice") \*

orderDetails("Qty")) -

(

(orderDetails("UnitPrice") \*

orderDetails("Qty")) \* orderDetails("Discount")

)

).as("OrderPrice") )

orderDetails1.show(5)

Graphical user interface, text, application

Description automatically generated

**// 1.2 Aggregate total by order id**

**val orderTot = orderDetails1.groupBy("OrderID").sum("OrderPrice").alias("OrderTotal")**

**orderTot.sort("OrderID").show(5)**

Graphical user interface, text, application

Description automatically generated

**//1.3 join order details & orders to add the order total**

**val orders1 = orders.join(orderTot, orders("OrderID").equalTo(orderTot("OrderID")), "inner")**

**.select(orders("OrderID"),**

**orders("CustomerID"),**

**orders("OrderDate"),**

**orders("ShipCountry").alias("ShipCountry"),**

**orderTot("sum(OrderPrice)").alias("Total"))**

**orders1.sort("CustomerID").show()**

Table

Description automatically generated

// 1.4 check if there are any null columns

orders1.filter(orders1("Total").isNull).show()

// #2 add a date column

val orders2 = orders1.withColumn("Date", to\_date(orders1("OrderDate")))

orders2.show(2)

orders2.printSchema()

**// #3 add month and year**

**val orders3 = orders2.withColumn("Month", month(orders2("OrderDate"))).withColumn("Year",year(orders2("OrderDate")))**

**orders3.show(2)**

### Graphical user interface, text, application Description automatically generated

### Okay, back to what we want you to answer:

### How many orders by month/year?

val ordersMY = orders3.groupBy("Month","Year").count()

ordersMY.sort(ordersMY("count").desc).show(5)

Table

Description automatically generated

### Total Sales for each customer by year?

// #4 total Sales for each customer by year

val orderTotCY = orders3.groupBy("CustomerID", "Year").sum("Total")

ordersCY.sort(ordersCY("sum(Total)").desc).show(5)

Text, letter

Description automatically generated

### Average order total by customer by year?

// #5 total Sales for each customer by year

val ordersAvgCY = orders3.groupBy("CustomerID", "Year").avg("Total")

ordersAvgCY.sort(ordersAvgCY("avg(Total)").desc).show(5)

Text

Description automatically generated

### Average order total by customer?

// #6 total by customer

val ordersAvgTC = orders3.groupBy("CustomerID").avg("Total")

ordersAvgTC.sort(ordersAvgTC("avg(Total)").desc).show(5)

Text

Description automatically generated