Lab 2

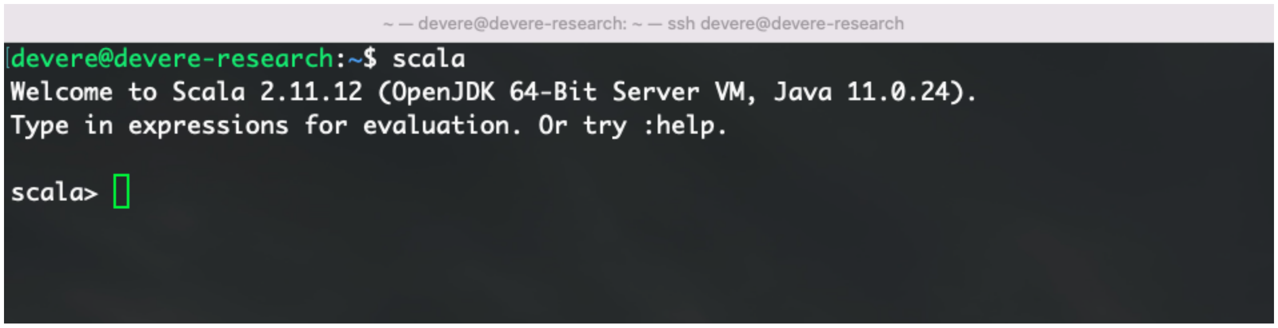
COSC760 - Big Data Analytics

Devere Anthony Weaver

1 Installing Scala

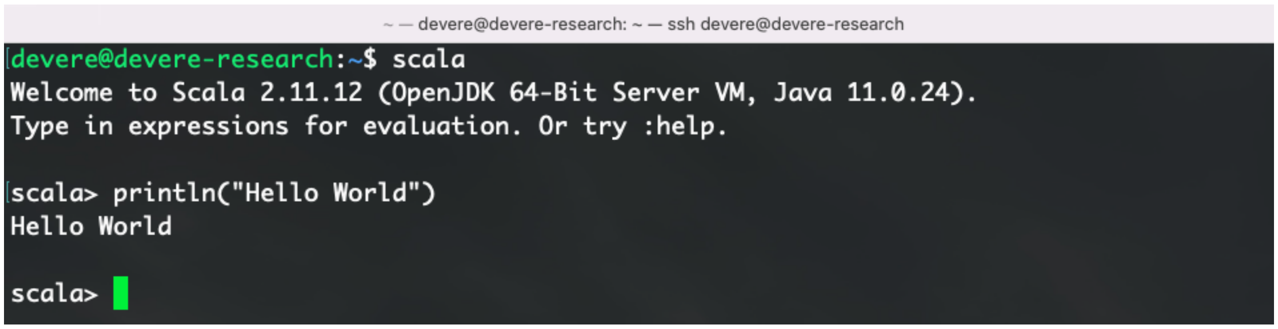
The following steps were used to successfully install Scala and Spark:

1. Download Scala onto Ubuntu machine



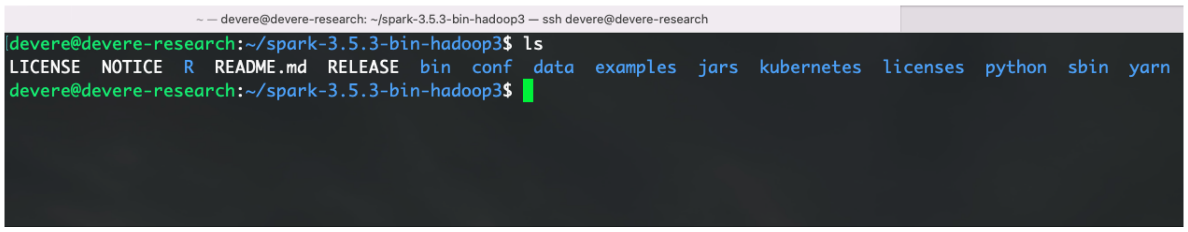
```
devere@devere-research:~$ scala
Welcome to Scala 2.11.12 (OpenJDK 64-Bit Server VM, Java 11.0.24).
Type in expressions for evaluation. Or try :help.

scala>
```

2. Test the Scala install



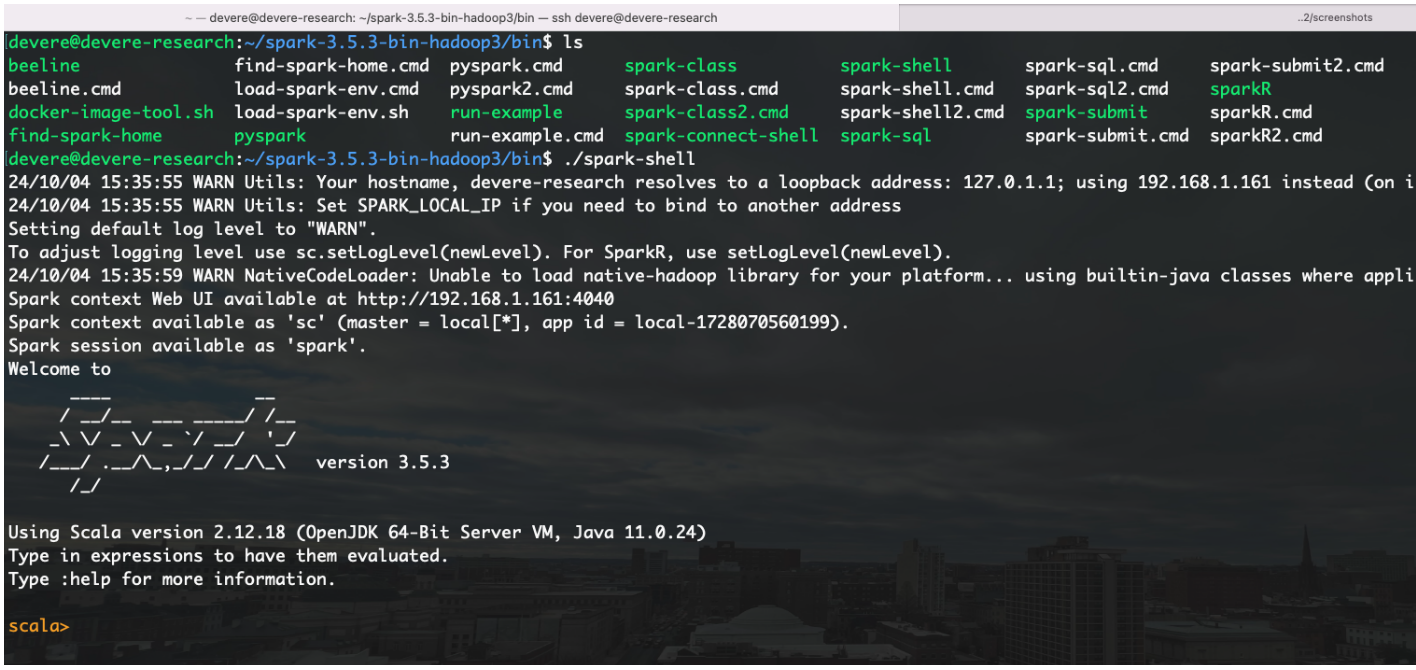
```
devere@devere-research:~$ scala
Welcome to Scala 2.11.12 (OpenJDK 64-Bit Server VM, Java 11.0.24).
Type in expressions for evaluation. Or try :help.

scala> println("Hello World")
Hello World

scala>
```

3. Download Spark using `wget`
4. Start the Spark shell using `./spark-shell`



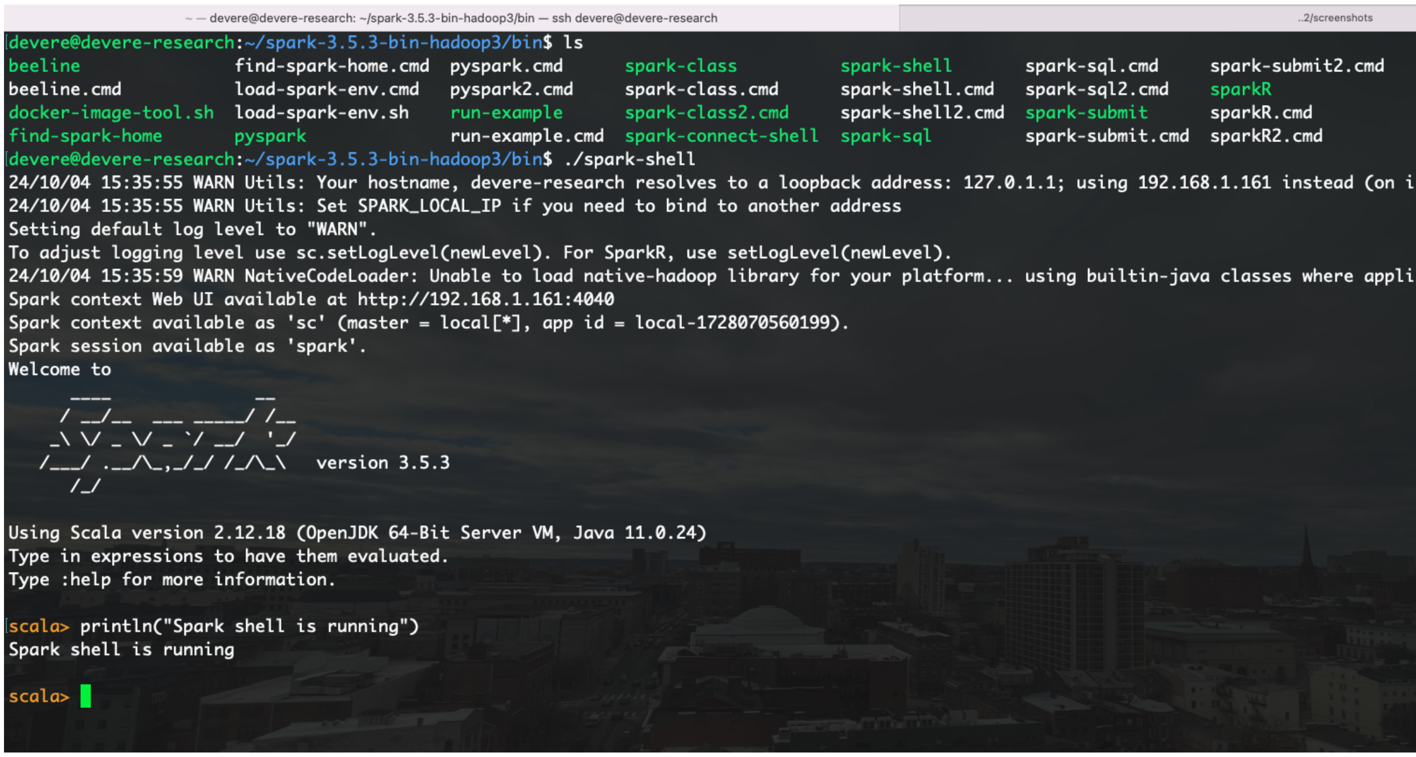
~ — devere@devere-research: ~/spark-3.5.3-bin-hadoop3 — ssh devere@devere-research

```
devere@devere-research:~/spark-3.5.3-bin-hadoop3$ ls
LICENSE  NOTICE  R  README.md  RELEASE  bin  conf  data  examples  jars  kubernetes  licenses  python  sbin  yarn
devere@devere-research:~/spark-3.5.3-bin-hadoop3$
```


~ — devere@devere-research: ~/spark-3.5.3-bin-hadoop3/bin — ssh devere@devere-research
..2/screenshots

```
devere@devere-research:~/spark-3.5.3-bin-hadoop3/bin$ ls
beeline              find-spark-home.cmd  pyspark.cmd       spark-class         spark-shell       spark-sql.cmd     spark-submit2.cmd
beeline.cmd          load-spark-env.cmd   pyspark2.cmd      spark-class.cmd     spark-shell.cmd   spark-sql2.cmd    sparkR
docker-image-tool.sh load-spark-env.sh    run-example       spark-class2.cmd    spark-shell2.cmd  spark-submit      sparkR.cmd
find-spark-home      pyspark              run-example.cmd   spark-connect-shell spark-sql         spark-submit.cmd  sparkR2.cmd
devere@devere-research:~/spark-3.5.3-bin-hadoop3/bin$ ./spark-shell
24/10/04 15:35:55 WARN Utils: Your hostname, devere-research resolves to a loopback address: 127.0.1.1; using 192.168.1.161 instead (on i
24/10/04 15:35:55 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/10/04 15:35:59 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
Spark context Web UI available at http://192.168.1.161:4040
Spark context available as 'sc' (master = local[*], app id = local-1728070560199).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.3
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.24)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

5. Verify Spark shell is working by running a basic command

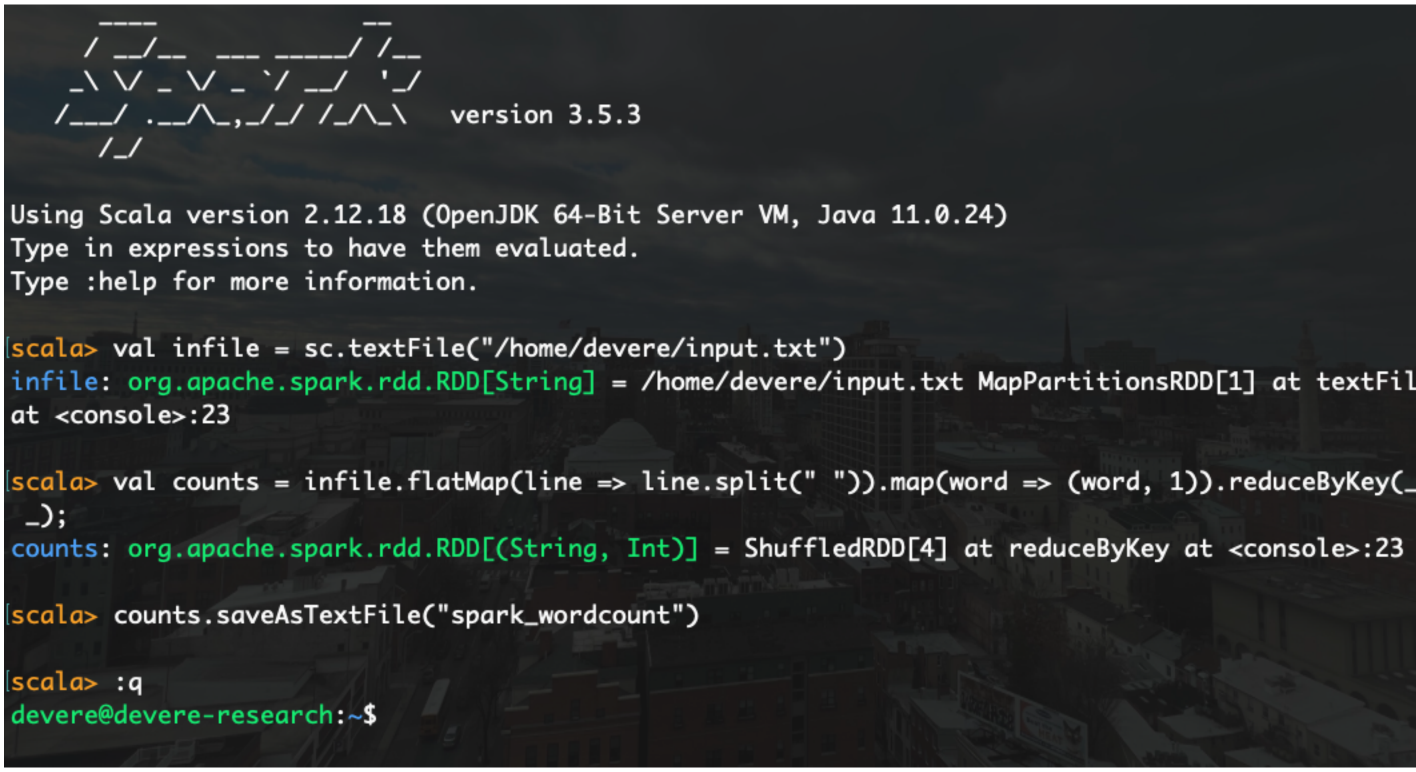

~ — devere@devere-research: ~/spark-3.5.3-bin-hadoop3/bin — ssh devere@devere-research
..2/screenshots

```
devere@devere-research:~/spark-3.5.3-bin-hadoop3/bin$ ls
beeline              find-spark-home.cmd  pyspark.cmd       spark-class         spark-shell       spark-sql.cmd     spark-submit2.cmd
beeline.cmd          load-spark-env.cmd   pyspark2.cmd      spark-class.cmd     spark-shell.cmd   spark-sql2.cmd    sparkR
docker-image-tool.sh load-spark-env.sh    run-example       spark-class2.cmd    spark-shell2.cmd  spark-submit      sparkR.cmd
find-spark-home      pyspark              run-example.cmd   spark-connect-shell spark-sql         spark-submit.cmd  sparkR2.cmd
devere@devere-research:~/spark-3.5.3-bin-hadoop3/bin$ ./spark-shell
24/10/04 15:35:55 WARN Utils: Your hostname, devere-research resolves to a loopback address: 127.0.1.1; using 192.168.1.161 instead (on i
24/10/04 15:35:55 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/10/04 15:35:59 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where appli
Spark context Web UI available at http://192.168.1.161:4040
Spark context available as 'sc' (master = local[*], app id = local-1728070560199).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.3
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.24)
Type in expressions to have them evaluated.
Type :help for more information.

scala> println("Spark shell is running")
Spark shell is running

scala>
```

2 Wordcount Program

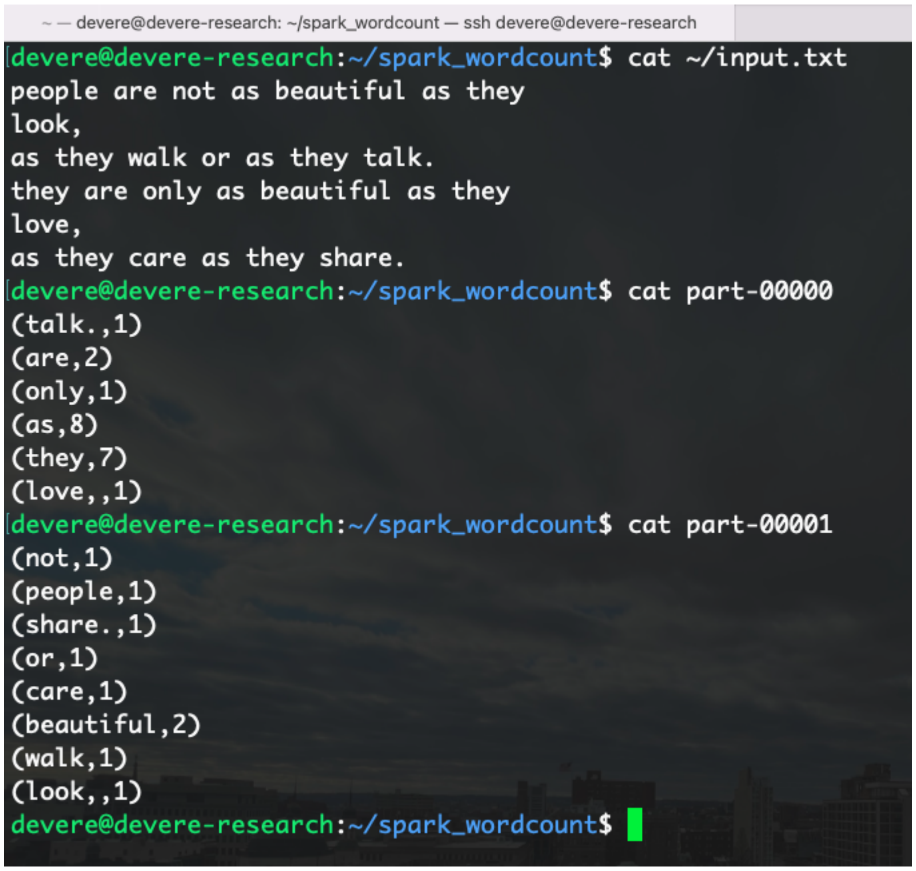
The following commands were used to successfully implement the word count program using Spark:

1. Start the Spark shell
2. Create an RDD from a text file, count the number of words, and output the results



```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.3
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.24)
Type in expressions to have them evaluated.
Type :help for more information.

[scala> val infile = sc.textFile("/home/devere/input.txt")
infile: org.apache.spark.rdd.RDD[String] = /home/devere/input.txt MapPartitionsRDD[1] at textFil
at <console>:23

[scala> val counts = infile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_
 _);
counts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:23

[scala> counts.saveAsTextFile("spark_wordcount")

[scala> :q
devere@devere-research:~$
```

3. The following is the count output and the text file contents

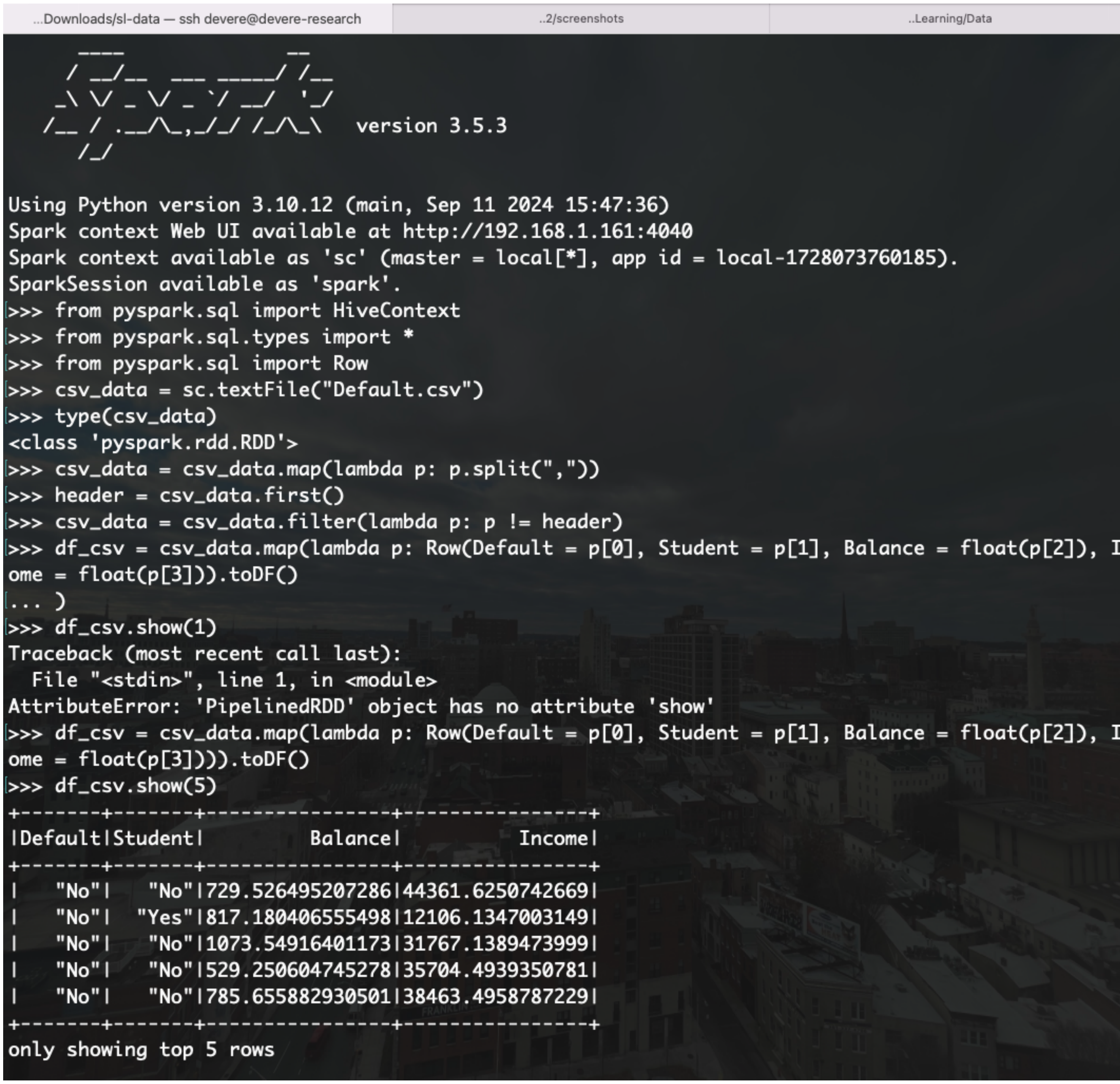


```
~ — devere@devere-research: ~/spark_wordcount — ssh devere@devere-research
devere@devere-research:~/spark_wordcount$ cat ~/input.txt
people are not as beautiful as they
look,
as they walk or as they talk.
they are only as beautiful as they
love,
as they care as they share.
devere@devere-research:~/spark_wordcount$ cat part-00000
(talk.,1)
(are,2)
(only,1)
(as,8)
(they,7)
(love,,1)
devere@devere-research:~/spark_wordcount$ cat part-00001
(not,1)
(people,1)
(share.,1)
(or,1)
(care,1)
(beautiful,2)
(walk,1)
(look,,1)
devere@devere-research:~/spark_wordcount$
```

3 Import CSV Files into HIVE using PySpark

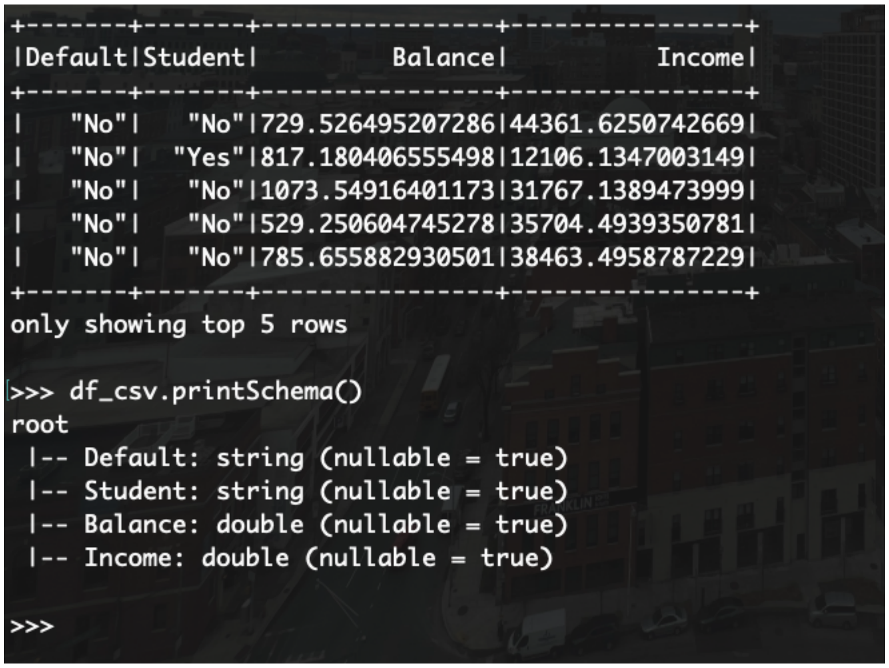
The following commands were used to successfully import a CSV file into HIVE and create a table:

1. Open Pyspark shell
2. Import a CSV file containing default data, parse the attributes, and create a tablew



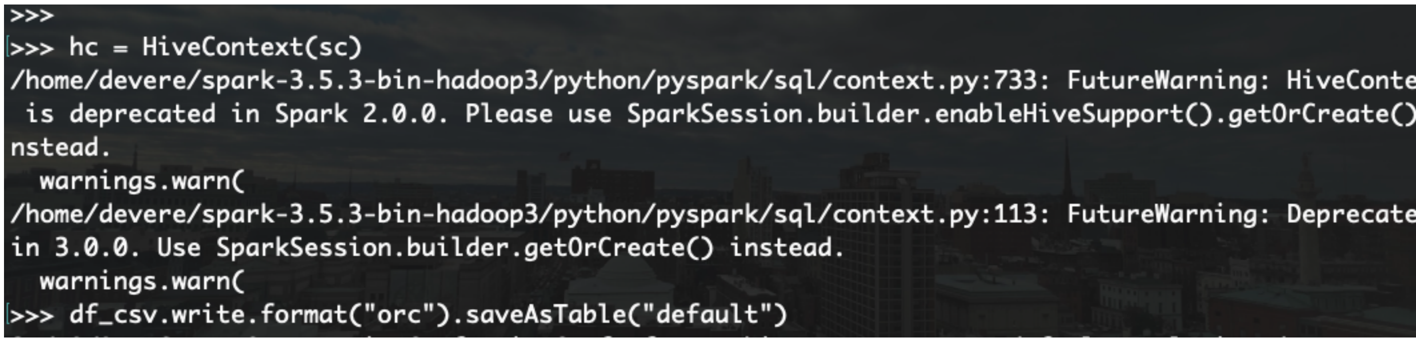
```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.5.3
      /_/

Using Python version 3.10.12 (main, Sep 11 2024 15:47:36)
Spark context Web UI available at http://192.168.1.161:4040
Spark context available as 'sc' (master = local[*], app id = local-1728073760185).
SparkSession available as 'spark'.
>>> from pyspark.sql import HiveContext
>>> from pyspark.sql.types import *
>>> from pyspark.sql import Row
>>> csv_data = sc.textFile("Default.csv")
>>> type(csv_data)
<class 'pyspark.rdd.RDD'>
>>> csv_data = csv_data.map(lambda p: p.split(","))
>>> header = csv_data.first()
>>> csv_data = csv_data.filter(lambda p: p != header)
>>> df_csv = csv_data.map(lambda p: Row(Default = p[0], Student = p[1], Balance = float(p[2]), I
ome = float(p[3])).toDF()
... )
>>> df_csv.show(1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'PipelinedRDD' object has no attribute 'show'
>>> df_csv = csv_data.map(lambda p: Row(Default = p[0], Student = p[1], Balance = float(p[2]), I
ome = float(p[3]))).toDF()
>>> df_csv.show(5)
+-------+-------+----------------+----------------+
|Default|Student|         Balance|          Income|
+-------+-------+----------------+----------------+
|   "No"|   "No"|729.526495207286|44361.6250742669|
|   "No"|  "Yes"|817.180406555498|12106.1347003149|
|   "No"|   "No"|1073.54916401173|31767.1389473999|
|   "No"|   "No"|529.250604745278|35704.4939350781|
|   "No"|   "No"|785.655882930501|38463.4958787229|
+-------+-------+----------------+----------------+
only showing top 5 rows
```

3. Check the newly created schema



```
+-------+-------+----------------+----------------+
|Default|Student|         Balance|          Income|
+-------+-------+----------------+----------------+
|   "No"|   "No"|729.526495207286|44361.6250742669|
|   "No"|  "Yes"|817.180406555498|12106.1347003149|
|   "No"|   "No"|1073.54916401173|31767.1389473999|
|   "No"|   "No"|529.250604745278|35704.4939350781|
|   "No"|   "No"|785.655882930501|38463.4958787229|
+-------+-------+----------------+----------------+
only showing top 5 rows

>>> df_csv.printSchema()
root
 |-- Default: string (nullable = true)
 |-- Student: string (nullable = true)
 |-- Balance: double (nullable = true)
 |-- Income: double (nullable = true)

>>>
```

4. Output the new table



```
>>>
>>> hc = HiveContext(sc)
/home/devere/spark-3.5.3-bin-hadoop3/python/pyspark/sql/context.py:733: FutureWarning: HiveConte
 is deprecated in Spark 2.0.0. Please use SparkSession.builder.enableHiveSupport().getOrCreate()
nstead.
  warnings.warn(
/home/devere/spark-3.5.3-bin-hadoop3/python/pyspark/sql/context.py:113: FutureWarning: Deprecate
in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
>>> df_csv.write.format("orc").saveAsTable("default")
```