Experiment 7: Spark User Defined Function

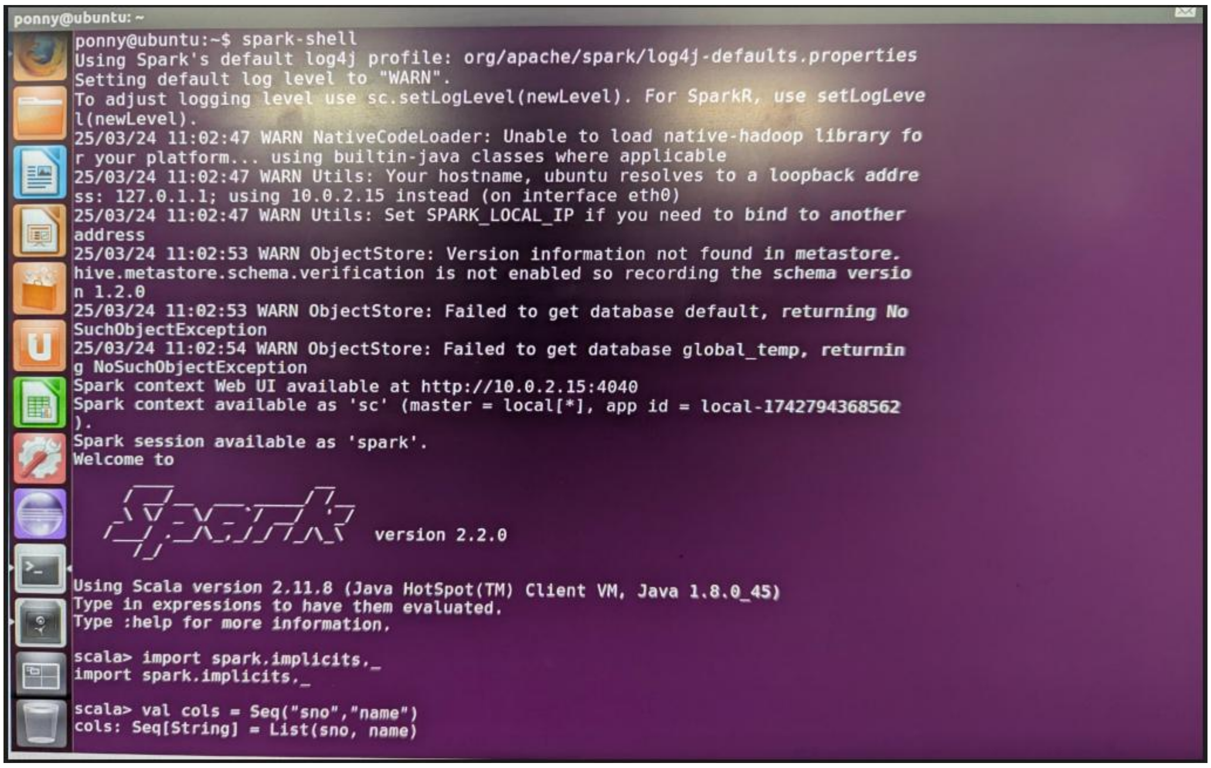
Namansh Singh Maurya
22MIA1034

Aim:

To use User Defined Functions in Spark.

Algorithm/Procedure:

1. Checking if Spark is present in the machine or not by using **spark-shell** command if it is present, we will spark version displayed on the screen, otherwise error.



```
ponny@ubuntu: ~
ponny@ubuntu:~$ spark-shell
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
25/03/24 11:02:47 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
25/03/24 11:02:47 WARN Utils: Your hostname, ubuntu resolves to a loopback addre
ss: 127.0.1.1; using 10.0.2.15 instead (on interface eth0)
25/03/24 11:02:47 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
25/03/24 11:02:53 WARN ObjectStore: Version information not found in metastore.
hive.metastore.schema.verification is not enabled so recording the schema versio
n 1.2.0
25/03/24 11:02:53 WARN ObjectStore: Failed to get database default, returning No
SuchObjectException
25/03/24 11:02:54 WARN ObjectStore: Failed to get database global_temp, returnin
g NoSuchObjectException
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-1742794368562
).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) Client VM, Java 1.8.0_45)
Type in expressions to have them evaluated.
Type :help for more information.

scala> import spark.implicits._
import spark.implicits._

scala> val cols = Seq("sno","name")
cols: Seq[String] = List(sno, name)
```

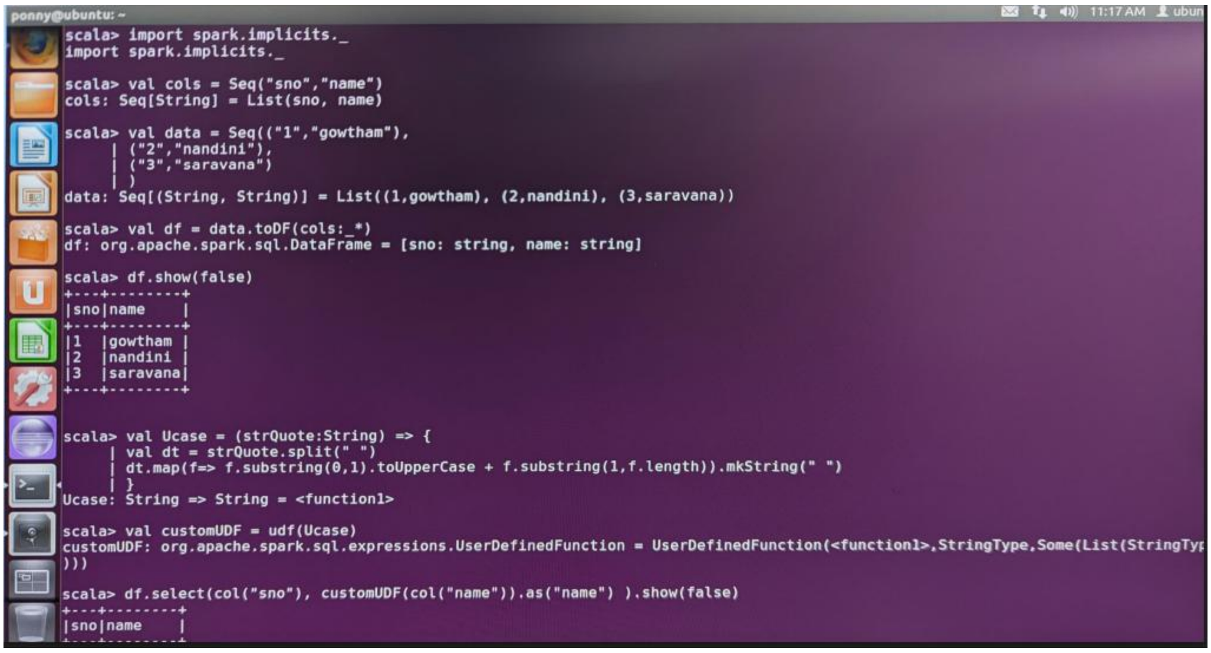
2. After Reviewing the version of Spark, we can move on to the predefined functions.

3. Below are all the codes that were used in doing this experiment.

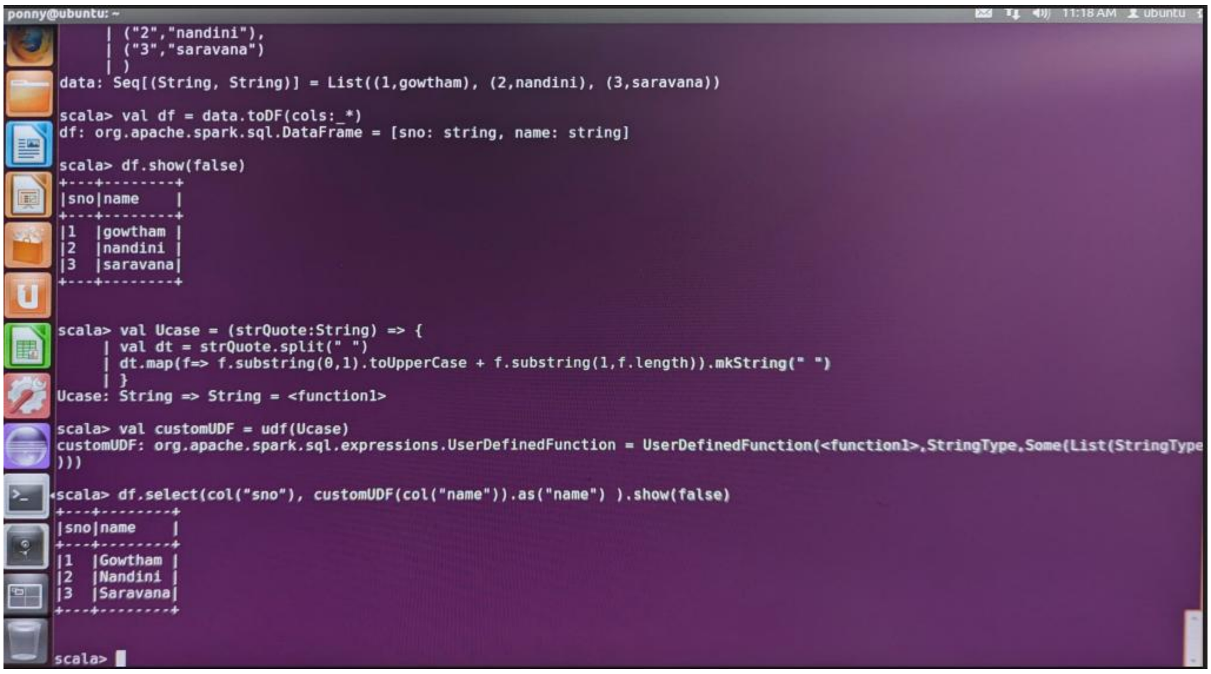
Program:

```
import spark.implicits._
val cols = Seq(“sno”,”name”)
val data = Seq((“1”,gowtham”),(“2”,”nandini”),(“3”,”saravana”))
val df = data.toDF(cols:_*)
df.show(false)
val Ucase  =  (strQuote: String) => { val dt = strQuote. split (" ")
dt.map(f=> f. substring (0, 1) . toUpperCase + f.substring (1, f.length) )
.mkString (" ") }
val customUDF = udf(Ucase)
df.select(col(“sno”),customUDF(col(“name”)).as(“name”)).show(false)
```

Output:



```
scala> import spark.implicits._
import spark.implicits._

scala> val cols = Seq("sno","name")
cols: Seq[String] = List(sno, name)

scala> val data = Seq(("1","gowtham"),
     | ("2","nandini"),
     | ("3","saravana")
     | )
data: Seq[(String, String)] = List((1,gowtham), (2,nandini), (3,saravana))

scala> val df = data.toDF(cols:_*)
df: org.apache.spark.sql.DataFrame = [sno: string, name: string]

scala> df.show(false)
+---+--------+
|sno|name    |
+---+--------+
|1  |gowtham |
|2  |nandini |
|3  |saravana|
+---+--------+


scala> val Ucase = (strQuote:String) => {
     | val dt = strQuote.split(" ")
     | dt.map(f=> f.substring(0,1).toUpperCase + f.substring(1,f.length)).mkString(" ")
     | }
Ucase: String => String = <function1>

scala> val customUDF = udf(Ucase)
customUDF: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,Some(List(StringTyp
)))

scala> df.select(col("sno"), customUDF(col("name")).as("name") ).show(false)
+---+--------+
|sno|name    |
```



```
ponny@ubuntu: ~                                                    11:18 AM  ubuntu
     | ("2","nandini"),
     | ("3","saravana")
     | )
data: Seq[(String, String)] = List((1,gowtham), (2,nandini), (3,saravana))

scala> val df = data.toDF(cols:_*)
df: org.apache.spark.sql.DataFrame = [sno: string, name: string]

scala> df.show(false)
+---+--------+
|sno|name    |
+---+--------+
|1  |gowtham |
|2  |nandini |
|3  |saravana|
+---+--------+


scala> val Ucase = (strQuote:String) => {
     | val dt = strQuote.split(" ")
     | dt.map(f=> f.substring(0,1).toUpperCase + f.substring(1,f.length)).mkString(" ")
     | }
Ucase: String => String = <function1>

scala> val customUDF = udf(Ucase)
customUDF: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function1>,StringType,Some(List(StringType
)))

scala> df.select(col("sno"), customUDF(col("name")).as("name") ).show(false)
+---+--------+
|sno|name    |
+---+--------+
|1  |Gowtham |
|2  |Nandini |
|3  |Saravana|
+---+--------+


scala> █
```

Result:

Hence we used spark to use the user defined functions and get the correct output as intended. Spark’s functions were successfully understood.