Exp Name: SPARK Framework UDF (User Defined Function) Experiment:

Name: Gouri Ramachandran

Aim

To implement and execute a **User Defined Function (UDF)** in the Apache Spark framework using Scala.

Algorithm/Procedure

1. Environment Setup

- o Open Oracle VirtualBox.
- Start the Hadoop virtual machine.
- Open the terminal and start Spark Shell by typing:
- o spark-shell
- o Confirm Spark version and wait for the Scala prompt to appear.

```
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 setLogLevel(newLevel).
25/03/24 10:53:18 WARN NativeCodeLoader: Unable to load native-hadoop library for ryour platform... using builtin-java classes where applicable
25/03/24 10:53:18 WARN Utils: Your hostname, ubuntu resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface eth0)
25/03/24 10:53:18 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/03/24 10:53:26 WARN ObjectStore: Version information not found in metastore. hive metastore.schema.verification is not enabled so recording the schema version 1.2.0
25/03/24 10:53:27 WARN ObjectStore: Failed to get database default, returning No SuchObjectException
25/03/24 10:53:27 WARN ObjectStore: Failed to get database global_temp, returning NoSuchObjectException
Spark context Web UI available at http://10.0.2.15:4040
Spark context Web UI available at http://10.0.2.15:4040
Spark session available as 'sc' (master = local[*], app id = local-174279380169:
).
Spark session available as 'spark'.
Welcome to

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.
```

2. Creating and Using a UDF in Spark

- o In the Spark shell, follow these steps:
- Import Spark SQL functions
- Create a sample DataFrame
- Define a UDF to add 5 years to age
- o Register the UDF and use it in DataFrame

```
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", "nandhini"),
("3", "saravana")
data: Seq[(String, String)] = List((1,gowtham), (2,nandhini), (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
    |nandhini
    saravana
13
scala> val customUDF = udf(Ucase)
<console>:26: error: not found: value Ucase
       val customUDF = udf(Ucase)
```

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

Program (UDF in Spark using Scala)

```
import spark.implicits
```

val cols = Seq("sno", "name")

Output

The output in Spark shell will be:

sno	name
1	Gowtham
2	Nandhini
3	Saravana

Result

The User Defined Function (UDF) was successfully implemented and executed in the Spark framework. The program effectively added 5 years to each employee's age and displayed the updated values.