1. Test the spark environment by executing the spark's HdfsTest.scala example.

- Create a folder in the normal file system

```
mkdir training_project
```

- Create build code

```
vi build.sbt

name := "SparkMe Project"
version := "1.0"
organization := "edureka"
scalaVersion := "2.11.8"
val sparkVersion = "2.1.0"
libraryDependencies += "org.apache.spark" %% "spark-core" % sparkVersion %"provided"
libraryDependencies += "org.apache.spark" %% "spark-sql" % sparkVersion % "provided"
resolvers += Resolver.mavenLocal
```

Save and exit

```
Press ESC, :wq
```

- Verify build.sbt. It creates project and target directory

```
sbt compile
```

- Create src folder directory

```
mkdir -p src/main/scala/com/edureka/training
```

- Create a new directory in hdfs

```
hdfs dfs -mkdir use_cases/
```

- Upload input_sort_py.txt using Jupyter notebook
- Put a sample file in hdfs

```
hdfs dfs -put -f input_sort_py.txt use_cases/
```

- Execute the example. Remove extra spaces

```
spark2-submit --class org.apache.spark.examples.HdfsTest --deploy-mode client
/opt/cloudera/parcels/SPARK2/lib/spark2/examples/jars/spark-examples_2.11-2.1.0.cloudera2.jar
use_cases/input_sort_py.txt
```

2. **Try to implement the same example in pyspark and perform spark-submit.**

- List existing examples

```
ls /opt/cloudera/parcels/SPARK2/lib/spark2/examples/src/main/python/
```

- Copy example files to training_project

```
cd training_project

cp /opt/cloudera/parcels/SPARK2/lib/spark2/examples/src/main/python/sort.py sort.py
```

- Implement in python

```
from __future__ import print_function

import sys,time

from pyspark.sql import SparkSession

if __name__ == "__main__":

   if len(sys.argv) != 2:

      print("Usage: hdfstest.py <file>", file=sys.stderr)

      exit(-1)

   spark = SparkSession.builder.appName("HdfsTest_py_TGA").getOrCreate()

   file_ = spark.read.text(sys.argv[1]).rdd

   mapped = file_.map(lambda s:len(s)).cache()

   for i in range(10):

      start_time = time.time()

      mapped.map(lambda x: x+2)

      end_time = time.time()

      print('----> Iteration took:', end_time - start_time,'ms')

      spark.stop()
```

- Execute it
  spark2-submit hdfstest.py use_cases/input_sort_py.txt

3. **Analyze the behavior of spark application on Spark web UI**

   I went to spark web UI http://bdlabs.edureka.co:50014

   I checked the Jobs, Stages, and Executors

4. Edit the application and add custom logs. Once executed check the Spark logs.

- Create source directory “m4” in project source folder

```
mkdir -p src/main/scala/com/edureka/training/m4/
```

- Copy HdfsTest.scala in “m4”

```
cp/opt/cloudera/parcels/SPARK2/lib/spark2/examples/src/main/scala/org/apache/spark/examples/HdfsTest.scala src/main/scala/com/edureka/training/m4/
```

- Change package to

```
com.edureka.training.m4
```

- Add this dependency to build.sbt

```
libraryDependencies += "com.typesafe.scala-logging" %% "scala-logging" % "3.9.0"
```

- Add this import on top of HdfsTest.scala and add custom logs

```
import com.typesafe.scalalogging.Logger
```

```
logger.info("Hello there!")
```

- Compile, Package

```
sbt compile
sbt package
```

- Submit

```
spark2-submit --class com.edureka.training.m4.HdfsTest --deploy-mode client target/scala-2.11/sparkme-project_2.11-1.0.jar use_cases/input_sort_py.txt
```

- Collect logs from yarn

```
yarn logs --applicationId application_1528714825862_137643
```

5. Transfer the sample dataset from RDBMS to HDFS

Upload csv file using ftp

Login to mysql, change database, create table

```
mysql -h mysqldb.edu.cloudlab.com -u labuser --password=edureka
```

```
use sq672184
```

```
create table financial_regulation (SYMBOL varchar(100),SERIES varchar(50),OPEN double,HIGH double, LOW double,CLOSE double,LAST double,PREVCLOSE double, TOTTRDQTY double,TOTTRDVAL double,TIMESTAMP date,TOTALTRADES int,ISIN varchar(50));
```

```
show tables;
```

- Load csv into table

```
load data local infile '/mnt/home/edureka_672184/data/FINAL_FROM_DF.csv' into table
financial_regulation;
```

- Sqoop import

```
sqoop import --connect jdbc:mysql://sqoopdb.edu.cloudlab.com/sq672184 --username labuser -
password edureka --table financial_regulation -m 1 --target-dir /user/edureka_672184/use_cases/fr/
```

6. **Validate the loaded data by comparing the statistics of data both in source and HDFS**

- Check files in HDFS using recursive list

```
 hdfs dfs -ls -R /user/edureka_672184/use_cases/
```

- Count the lines

```
hdfs dfs -cat /user/edureka_672184/use_cases/fr/*|wc -l
```

this returns 846405

- Match first 5 rows

```
hdfs dfs -cat /user/edureka_672184/use_cases/fr/*|head -5
```

7. **Create a new directory EQ in HDFS and transfer the data where series is EQ**

```
hdfs dfs -mkdir /user/edureka_672184/use_cases/fr/eq
```

```
sqoop import --connect jdbc:mysql://sqoopdb.edu.cloudlab.com/sq672184 --username labuser --
password edureka --table financial_regulation --where "SERIES='EQ'" -m 1 -- target-dir
/user/edureka_321047/use_cases/fr/eq/
```

8. **Set total trades which are less than 500 to 0 and and transfer only updated rows.**

- Add new column updated in mysql

```
alter table financial_regulation add updated bit;
```

- Update table

```
update financial_regulation set updated=0 where TRADES
```

- Step 12.3: Transfer data

```
sqoop import --connect jdbc:mysql://sqoopdb.edu.cloudlab.com/use_cases --username labuser --
password edureka --table financial_regulation --incremental append --checkcolumn 'updated' --where
"SERIES='EQ'" -m 1 --target-dir /user/edureka_321047/use_cases/fr/eq/
```