**Practical 1:Hadoop installation steps**:

Installation of Java

Step1: download java8 from the official site: <https://www.oracle.com/in/java/technologies/downloads/#java8>

Download 64 bit for windows

Step2:Install the java directly in the C drive directly. Create a new folder java in c drive and install java in that folder and complete the installation

Step3: Go to the java folder and next to the bin folder you can see there is no jdk folder in it . Then go to Program files java folder cut the jdk folder in it and add it to the java folder recently created in c drive. Then go to the program files and delete the java folder in it to avoid duplication

step4:Now set the environmental variables:

Under User variable - -> JAVA\_HOME: and set the path of java bin folder which is present in (java - -> jdk 1.8 – - > bin)

Under System variable – - > Path - -> select new and paste the same path what you have set on the user variables click on ok to set changes

Step5: Check if java is installed correctly - -> java -version and press enter

Next command - -> javac

Step6: Java is installed successfully

Installation of hadoop:

Step1: Download hadoop from the official site : <https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz>

Step2: extract it in the C drive directly

Step3: Open the hadoop 3.4 folder- ->etc folder - ->hadoop - -> There are 4 files to be edited

Step4: Edit core-site.xml

Add this

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

Step5: Edit mapreduce-site.xml

Add this

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

Step6: Edit Yarn-site.xml

Add this

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle</name>

<value>org.apache.hadoop.mapred.shufflehandler</value>

</property>

</configuration>

Before editing hdfs-site.xml we need to create a data folder in hadoop folder

Hadoop→data →namenode and datanode create these two folders in data folder

step7:Edit hdfs-site.xml

ADD THIS

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>C:\hadoop\data\namenode</value>

</property>

<property>

<name>dfs.datanode.name.dir</name>

<value>C:\hadoop\data\datanode</value>

</property>

</configuration>

Step8: Edit hadoop-env file

Make these changes:

In section set JAVA\_HOME=put the java path that is C:\Java\jdk-1.8

Step9: set environmental variables for hadoop:

Under User variable:

HADOOP\_HOME value: path of the hadoop’s bin folder

Under system variable:

Set the path for both bin and sbin folder

eg:C:\hadoop\bin

C:\hadoop\sbin

Step9: To check if it is running or not

In command prompt add “hdfs namenode -format” and hit enter

Now if you want to check further go to sbin folder cmd type this command “start-all.cmd”

Note: If you are getting error then go to etc again select hadoop-env and edit the username to your username but without spaces

Step10:verify hdfs web portal ui

<http://localhost:9870/dfshealth.html#tab-overview>

**Pract2: Installation of scala and apache spark**

#Steps to download apache spark and scala

Step 1: Download java 11.0.3.

Step 2: Download the scala from the website https://www.scala-lang.org/download/

Step 3: Click on the scala installer for windows and after downloading it, extract the file in C drive.

Step 4: Now open the downloaded cmd for scala and it will ask for y/n : Type Y.

Now open the following path in c drive

C:\Users\Admin\AppData\Local\Coursier\data\bin

Step 5: Now set the Environmental Variable

step6:Copy this path and Add it into both the system variable and path.

Now Download the Spark from the link below select the specific version shown.

https://spark.apache.org/downloads.htm

Step7:Extract it to c drive

Now open the following path in c drive

C:\spark-3.4.3-bin-hadoop\spark-3.4.3-bin-hadoop 3\bin

Step 8: Now set the Environmental Variable again

step9:Now navigate to the C:\spark-3.4.3-bin-hadoop\spark-3.4.3-bin-hadoop 3\bin Open cmd and type , to check if spark is installed properly.

Step 10: To check if scala is installed properly

Navigate to the C:\Users\Admin\AppData\Local\Coursier\data\bin

step11:Now go back to the Spark bin folder

C:\spark-3.4.3-bin-hadoop\spark-3.4.3-bin-hadoop 3\bin And type spark-shell

1) val a =spark.read.json("C:/spark/spark-3.5.2-bin-hadoop3/examples/src/main/resources/people.json")

2)a.createOrReplaceTempView("people")

3)val sqlDF = spark.sql("SELECT \* FROM people")

4)sqlDF.show()

Global Temoprary view

1)a.createGlobalTempView("people")

2)spark.sql("SELECT \* FROM global\_temp.people").show()

3)spark.newSession().sql("SELECT \* FROM global\_temp.people").show()

creating datasets

case class Person(name: String, age: Long)

1)val caseClassDS = Seq(Person("Andy", 32)).toDS()

2)caseClassDS.show()

3) val primitiveDS = Seq(1, 2, 3).toDS()

4)primitiveDS.map(\_ + 1).collect()

5)val path = spark.read.json("C:/spark/spark-3.5.2-bin-hadoop3/examples/src/main/resources/people.json")

6)val peopleDS = spark.read.json("C:/spark/spark-3.5.2-bin-hadoop3/examples/src/main/resources/people.json"). as[Person] scala> peopleDS.show()

Inferring the schema using reflection

1)import spark.implicits.\_ import spark.implicits.\_ scala

2)val peopleDF = spark.sparkContext val peopleDF: org.apache.spark.SparkContext = org.apache.spark.SparkContext@739785c5

3).textFile("C:/spark/spark-3.5.2-bin-hadoop3/examples/src/main/resources/people.txt")

4) .map(\_.split(","))

5)import org.apache.spark.sql.SparkSession

6)val spark = SparkSession.builder().appName("Text File Example").config("spark.master",

"local").getOrCreate()

7) import spark.implicits.\_

8)case class Person(name: String, age: Int)

9)val peopleDF =

spark.sparkContext.textFile("C:/spark/spark-3.5.2-bin-hadoop3/examples/src/main/resources/pe ople.txt").map(\_.split(",")).map(attributes => Person(attributes(0), attributes(1).trim.toInt)).toDF()

10)peopleDF.createOrReplaceTempView("people")

11)val teenagersDF = spark.sql("SELECT name, age FROM people WHERE age BETWEEN 13 AND 19")

12)teenagersDF.map(teenager => "Name: " + teenager(0)).show()

13)teenagersDF.map(teenager => "Name: " + teenager.getAs[String]("name")).show()

14)implicit val mapEncoder = org.apache.spark.sql.Encoders.kryo[Map[String, Any]]

15)teenagersDF.map(teenager => teenager.getValuesMap[Any](List("name",

"age"))).collect()

**Practical no:3 spark graphx**

1)import org.apache.spark.\_ scala

2)import org.apache.spark.rdd.RDD

3)import org.apache.spark.graphx.\_

4)val vertices = Array((1L,("A")),(2L,("B")),(3L,("C"))) scala> val vRDD = sc.parallelize(vertices)

5)vRDD.take(2)

6)val edges = Array(Edge(1L,2L,1800),Edge(2L,3L,800),Edge(3L,1L,1400))

7)val eRDD = sc.parallelize(edges) scala> eRDD.take(2)

8)val nowhere = "nowhere" scala> val graph = Graph(vRDD,eRDD,nowhere)

9)graph.vertices.collect.foreach(print)

10)graph.edges.collect.foreach(print)

11) #To check number of airports

val numairports = graph.numVertices

12)#To check routes

val numairports = graph.numEdges

13)(graph.edges.filter{case Edge(src,dst,prop)=>prop>1000}.collect.foreach(println))

14)#check triplets

graph.triplets.take(3).foreach(println)

15)#Indegree

val i = graph.inDegrees

16)i.collect()

17)#Outdegree

val o= graph.outDegrees scala

o.collect()

17)#Total degree

val t = graph.degrees

t.collect()

**Practical 4:Pyspark**

!pip install pyspark

import pyspark

import pandas as pd

pd.read\_csv('/content/data.csv')

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('Practice').getOrCreate()

spark = SparkSession.builder.appName('Practice').getOrCreate()

df\_pyspark = spark.read.option('header', 'true').csv('/content/data.csv',

inferSchema=True)

df\_pyspark.show()

type(df\_pyspark)

df\_pyspark.printSchema()

df\_pyspark.head()

df\_pyspark.columns

df\_pyspark.select('Name').show()

df\_pyspark.select(['Name', 'Experience']).show()

data types

df\_pyspark = df\_pyspark.withColumn('Experience After 2 years',

df\_pyspark['Experience']+2)

df\_pyspark.show()

df\_pyspark=df\_pyspark.drop('Experience After 2 years')

df\_pyspark.show()

df\_pyspark.withColumnRenamed('Name', 'New Name').show()

df\_pyspark.na.drop().show()

df\_pyspark.na.drop(how='all').show()

df\_pyspark.na.drop(how='any').show()

df\_pyspark.na.drop(how='any', thresh=2).show()

df\_pyspark.na.drop(how='any', subset=['Experience']).show()

df\_pyspark.na.fill('Missing Values', ['Experience', 'Age']).show()

df\_pyspark.na.fill('Missing Values').show()

#Writing the function to calculate the value at the place of null.

from pyspark.ml.feature import Imputer

imputer = Imputer(

inputCols=['Age', 'Experience', 'Salary'],

outputCols=["{}\_imputed".format(c) for c in ['Age', 'Experience',

'Salary']]

).setStrategy('mean')

#Add imputation columns to df

imputer.fit(df\_pyspark).transform(df\_pyspark).show()

**Practical no5: Installation of hbase**

1. Download the HBase bin file from here.

https://archive.apache.org/dist/hbase/2.2.5/hbase-2.2.5-bin.tar.gz

2. Create a new folder in C drive named hbasesetup. Extract the files and paste it in the hbasesetup. Create 2 new folders hbase and zookeeper inside.

3. Open the bin folder. Search for the file hbase.cmd and edit it in notepad. Search for java\_arguments and remove %HEAP\_SETTINGS%.

4. Open the conf folder and edit the file hbase-env.cmd in notepad. Add the following lines inside.

paste this

set JAVA\_HOME=C:\Progra~1\Java\jdk1.8.0\_202

set HBASE\_CLASSPATH=%HBASE\_HOME%\lib\client-facing-thirdparty\\*

set HBASE\_HEAPSIZE=8000

set HBASE\_OPTS="-XX:+UseConcMarkSweepGC" "-Djava.net.preferIPv4Stack=true"

set SERVER\_GC\_OPTS="-verbose:gc" "-XX:+PrintGCDetails" "-XX:+PrintGCDateStamps" %HBASE\_GC\_OPTS%

set HBASE\_USE\_GC\_LOGFILE=true

set HBASE\_JMX\_BASE="-Dcom.sun.management.jmxremote.ssl=false" "-Dcom.sun.management.jmxremote.authenticate=false"

set HBASE\_MASTER\_OPTS=%HBASE\_JMX\_BASE% "-Dcom.sun.management.jmxremote.port=10101"

set HBASE\_REGIONSERVER\_OPTS=%HBASE\_JMX\_BASE% "-Dcom.sun.management.jmxremote.port=10102"

set HBASE\_THRIFT\_OPTS=%HBASE\_JMX\_BASE% "-Dcom.sun.management.jmxremote.port=10103"

set HBASE\_ZOOKEEPER\_OPTS=%HBASE\_JMX\_BASE%-Dcom.sun.management.jmxremote.port=10104"

set HBASE\_REGIONSERVERS=%HBASE\_HOME%\conf\regionservers

set HBASE\_LOG\_DIR=%HBASE\_HOME%\logs

set HBASE\_IDENT\_STRING=%USERNAME%

set HBASE\_MANAGES\_ZK=true

**Step 5:** Open the conf folder and edit the file hbase-site.xml in notepad. Add the following lines after the last property tag.

<property>

<name>hbase.rootdir</name>

<value>file:///C:/hbasesetup/hbase-2.2.5/hbase</value>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/C:/hbasesetup/hbase-2.2.5/zookeeper</value>

</property>

<property>

<name> hbase.zookeeper.quorum</name>

<value>localhost</value>

</property>

**Step 6:** Set the HBase environment variables and it too path as well.

**Step 7:** Open command prompt and navigate to the hbase bin folder. Run the command start-hbase.cmd to start hbase

**Step 8:** Using the command jps you can check that our HMaster is running.

**Step 9:** Start the HBase Shell now with the command hbase shell. Initial startup may take some time.

**Step 10:** Your HBase Shell has been started. Ignore the warnings received while starting the shell.