PANDIT DEENDAYAL ENERGY UNIVERSITY

SCHOOL OF TECHNOLOGY

Course: Big Data Analytics

Course Code: 20DS506P

LAB MANUAL

M.Tech. (Data Science)

Semester 1

Submitted To:
Dr. Samir patel

Submitted By:
Dev Jethva
23MDS003

Index

databricks Scala basic Notebook 2023-08-07 11:12:22

(https://databricks.com)

```
var num= List(1,2,3,4)
```

```
num: List[Int] = List(1, 2, 3, 4)
```

```
num.head
```

```
res0: Int = 1
```

```
// first two element are show
num.take(2)
```

```
res3: List[Int] = List(1, 2)
```

```
num.sum
```

```
res4: Int = 10
```

```
var dev = List(1,1,1,2,3,4,2,2)
```

```
dev: List[Int] = List(1, 1, 1, 2, 3, 4, 2, 2)
```

```
// this element show only value to not same like 1,1,2,2,3,4,4 == 1,2,3,4
dev.distinct
```

```
res7: List[Int] = List(1, 2, 3, 4)
```

```
num(0)
```

```
res8: Int = 1
```

```
num(1)= 10
```

```
command-1298605597657410:1: error: value update is not a member of List[Int]
num(1)= 10
^
```

```
dev.size
```

```
res10: Int = 8
```

```
dev.reverse
```

```
res11: List[Int] = List(2, 2, 4, 3, 2, 1, 1, 1)
```

```
dev.min
```

```
res12: Int = 1
```

```
dev.max
```

```
res13: Int = 4
```

```
dev.isEmpty
```

```
res14: Boolean = false
```

```
var dev1 = List()
dev1.isEmpty
```

```
dev1: List[Nothing] = List()
res16: Boolean = true
```

```
var number = Array(1,2,3,4,5,6,7)
```

```
number: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7)
```

```
var lang = Array("Scall", "Python","Spark")
```

```
lang: Array[String] = Array(Scall, Python, Spark)
```

```
lang.head
```

```
res17: String = Scall
```

```
lang.tail
```

```
res18: Array[String] = Array(Python, Spark)
```

```
number(1)=10
```

```
number
```

```
res20: Array[Int] = Array(1, 10, 3, 4, 5, 6, 7)
```

```
import scala.collection.mutable.ArrayBuffer
```

```
import scala.collection.mutable.ArrayBuffer
```

```
var cars = new ArrayBuffer[String]()
```

```
cars: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer()
```

```
cars +="BMW"
```

```
res23: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW)
```

```
cars +="Jaguar"
cars +="Jaguar"
```

```
res24: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Jaguar)
```

```
cars +="TATA"
```

```
res25: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Jaguar, TATA)
```

```
cars.length
```

```
res26: Int = 4
```

```
cars.trimEnd(1)
```

```
cars
```

```
res28: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Jaguar)
```

```
cars.insert(2,"Bentley")
```

```
cars
```

```
res30: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Bentley, Jaguar)
```

```
cars.insert(4,"Bugatti")
```

```
cars
```

```
res33: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, Jaguar, Bentley, Jaguar, Bugatti)
```

```
val num= List(1,2,3,4,5)
```

```
num: List[Int] = List(1, 2, 3, 4, 5)
```

```
num.map(x => x*x)
```

```
res34: List[Int] = List(1, 4, 9, 16, 25)
```

```
num.map(y => y*y)
```

```
res35: List[Int] = List(1, 4, 9, 16, 25)
```

```
val a = num.map(x => x+1)
```

```
a: List[Int] = List(2, 3, 4, 5, 6)
```

```
// Nested map
val b = a.map(x => x*x)
```

```
b: List[Int] = List(4, 9, 16, 25, 36)
```

```
val c = b.map(x => x-1)
```

```
c: List[Int] = List(3, 8, 15, 24, 35)
```

```
val d = c.map(x => -x)
```

```
d: List[Int] = List(-3, -8, -15, -24, -35)
```

```
num.map(x => x+1).map(x => x*x).map(x => x-1).map(x => -x)
```

```
res36: List[Int] = List(-3, -8, -15, -24, -35)
```

```
val fruits = List("orange", "banana", "Apple")
```

```
fruits: List[String] = List(orange, banana, Apple)
```

```
// Length of the words
fruits.map(x => (x,x.length))
```

```
res39: List[(String, Int)] = List((orange,6), (banana,6), (Apple,5))
```

```
// how many corrector of the words of their like banana word is geater than 5.
fruits.filter(x => x.length > 5)
```

```
res40: List[String] = List(orange, banana)
```

```
val ratings=List(2.4,5.6,7.8,9.5)
```

```
ratings: List[Double] = List(2.4, 5.6, 7.8, 9.5)
```

```
fruits.filter(x => x.length == 5)
```

```
res42: List[String] = List(Apple)
```

```
fruits.filter(x => x.length != 5)
```

```
res43: List[String] = List(orange, banana)
```

```scala
val marks=ratings.map(x => x+10)
```

```
marks: List[Double] = List(12.4, 15.6, 17.8, 19.5)
```

```scala
val marks=ratings.map(x => x*10)
```

```
marks: List[Double] = List(24.0, 56.0, 78.0, 95.0)
```

```scala
val marks1=ratings.filter(x=> x>=60 && x<=74)
```

```
marks1: List[Double] = List()
```

```scala
// Funtion in Scala
```

```scala
def add(a:Double = 100 , b:Double = 200) : Double =
{
  var sum : Double = 0
  sum = a+b
  return sum
}
println("Sum:" + add(a = 10, b=20))
```

```
Sum:30.0
add: (a: Double, b: Double)Double
```

```scala
// if value is not given in add funcation return add funcation of a = 100 and b = 200
println("Sum1:" + add())
```

```
Sum1:300.0
```

```scala
var x = 1
  val y = if(x<3)
    println("value of x is less than 3")
  else
    println("value of x is greater than or equal to 3")
```

```
value of x is less than 3
x: Int = 1
y: Unit = ()
```

```
var marks = 75
  if(marks>=75){
    println("Distinction")
  }else if(marks >= 60 && marks <70){
    println("First class")
  }else if(marks >= 50 && marks <60){
    println("Seocnd class")
  }else if(marks >= 40 && marks <50){
    println("Pass class")
  }else println("fail")
```

```
Distinction
marks: Int = 75
```

```
// 3 funcations are calls and returns 3 times
def square(x:Double) :Double = {
  return x*x
}
def sumsquare(x:Double, y:Double) :Double = {
  return square(x) + square(x)
}
println("Sum of squares:"+ sumsquare(4,5))
```

```
Sum of squares:32.0
square: (x: Double)Double
sumsquare: (x: Double, y: Double)Double
```

```
def time() :Long = {
  println("Inside the funcations")
  return System.nanoTime()
}
def exect(t:Long) :Long = {
  println("Inside the Exect funcations")
  println("Time :"+t)
  println("Exiting From exect funcations")
  return t
}
println("Main Functions:"+ exect(time()))
```

```
Inside the funcations
Inside the Exect funcations
Time :4627384344930
Exiting From exect funcations
Main Functions:4627384344930
time: ()Long
exect: (t: Long)Long
```

```
var i =10
  while(i>0){
    println("hello :"+i)
    i = i-1
  }
```

```
hello :10
hello :9
hello :8
hello :7
hello :6
hello :5
hello :4
hello :3
hello :2
hello :1
i: Int = 0
```

```
var a =2
  do{
    println(a)
    a = a +2
  }while(a <= 10)
```

```
2
4
6
8
10
a: Int = 12
```

```
;// how to create object and how to create a class

object classEg{
  def main(arg:Array[String]){
    var obj = new NewClass("Hello World")
    obj.sayHi()
  }
}

class NewClass(mssg:String){
  def sayHi()= println(mssg)
}
```

```
defined object classEg
defined class NewClass
```

```
for(i <-1 to 10)
  println(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

```
// factorial number

var Factorial =1
  var num =5
  while(num>0){
    Factorial = Factorial * num
    num = num-1
  }
  println(Factorial)
```

```
120
Factorial: Int = 120
num: Int = 0
```

```
def isPrime(i: Int): Boolean =
    if (i <= 1)
        false
    else if (i == 2)
        true
    else
        !(2 until i).exists(n => i % n == 0)
```

```
isPrime: (i: Int)Boolean
```

databricks Scala Notebook 2023-08-21 11:07:59

(https://databricks.com)

```
// comment in scala
```

```
var a  = Array(List(1,2,3), List(1,2,3), List(1,2,3))
var b  = Array(List(4,5,6), List(4,5,6), List(4,5,6))
var c  = Array(Array(0,0,0), Array(0,0,0), Array(0,0,0))
var sum = 0

for (i<-0 to 2){
  for (j<-0 to 2){
    sum=0
    for(k<-0 to 2){
      sum = sum + (a(i)(k) * b(k)(j))
    }
    c(i)(j)=sum
  }
}
println(c)
```

```
[[I@696395e5
a: Array[List[Int]] = Array(List(1, 2, 3), List(1, 2, 3), List(1, 2, 3))
b: Array[List[Int]] = Array(List(4, 5, 6), List(4, 5, 6), List(4, 5, 6))
c: Array[Array[Int]] = Array(Array(24, 30, 36), Array(24, 30, 36), Array(24, 30, 36))
sum: Int = 36
```

```
a(1)(1)
```

```
res2: Int = 2
```

```
var a =Array(Array(1,2,3), List(1,2,3), List(1,2,3))
```

```
a: Array[java.io.Serializable] = Array(Array(1, 2, 3), List(1, 2, 3), List(1, 2, 3))
```

```
a(1)
```

```
res3: java.io.Serializable = List(1, 2, 3)
```

```
a(0)
```

```
res4: java.io.Serializable = Array(1, 2, 3)
```

Transportation function examples

```
import org.apache.spark.rdd.RDD
import org.apache.spark.sql.SparkSession
```

```
import org.apache.spark.rdd.RDD
import org.apache.spark.sql.SparkSession
```

```
object RDDParallelize{
  def main(args: Array[String]): Unit = {
    val spark:SparkSession = SparkSession.builder().master("local[1]").appName("SparkByExamples.com").getOrCreate()
    val rdd:RDD[Int]= spark.sparkContext.parallelize(List(1,2,3,4,5))
    val rddCollect:Array[Int]= rdd.collect()
    println("Number of partitions:" +rdd.getNumPartitions)
    println("Action: First element:" +rdd.first)
    println("Action: RDD converted to Array[Int]: ")
    rddCollect.foreach(println)
  }
}
```

```
defined object RDDParallelize
```

```
val rdda = sc.parallelize(List(1,2,3,4,5))
    val rddb = rdda.collect
    println("Number of Partition:" +rdda.getNumPartitions)
    println("Action: First element:" + rdda.first())
    rdda.foreach(println)
```

```
Number of Partition:8
Action: First element:1
rdda: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at command-3160921003574745:1
rddb: Array[Int] = Array(1, 2, 3, 4, 5)
```

```
val  rdda = sc.parallelize(List("mumbai", "Delhi", "Chennai", "Kolkatta"))
```

```
rdda: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[1] at parallelize at command-3160921003574746:1
```

```
var rddb = sc.parallelize(Array(1,2,3,4,5,6,7,8,9,10))
```

```
rddb: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[2] at parallelize at command-3160921003574747:1
```

```
var rddc = sc.parallelize(Seq.empty[String])
```

```
rddc: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[3] at parallelize at command-3160921003574748:1
```

```
rdda.collect
```

```
res8: Array[String] = Array(mumbai, Delhi, Chennai, Kolkatta)
```

```
rddb.collect
```

```
res9: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

```
rddc.collect
// Collect is action
```

```
res11: Array[String] = Array()
```

```
val b = rdda.map(x => (x,1))
//  map, filter is trnasformation
```

```
b: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[5] at map at command-3160921003574752:1
```

```
b.collect
```

```
res12: Array[(String, Int)] = Array((mumbai,1), (Delhi,1), (Chennai,1), (Kolkatta,1))
```

```
val b = rdda.map((_,1))
```

```
b: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[6] at map at command-3160921003574754:1
```

```
b.collect
```

```
res13: Array[(String, Int)] = Array((mumbai,1), (Delhi,1), (Chennai,1), (Kolkatta,1))
```

```
val b = rdda.map(x =>(x,x.length))
```

```
b: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[7] at map at command-3160921003574756:1
```

```
b.collect
```

```
res14: Array[(String, Int)] = Array((mumbai,6), (Delhi,5), (Chennai,7), (Kolkatta,8))
```

```
// Word Length

val a = sc.parallelize(List(1,2,3,4,5)).map(x=>List(x,x,x)).collect
```

```
a: Array[List[Int]] = Array(List(1, 1, 1), List(2, 2, 2), List(3, 3, 3), List(4, 4, 4), List(5, 5, 5))
```

```
val a = sc.parallelize(List(1,2,3,4,5)).flatMap(x=>List(x,x,x)).collect
```

```
a: Array[Int] = Array(1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5)
```

```
// What is difference between the map and flatmap
```

```
// applying filter on rdda city data
val rdda = sc.parallelize(List("Mumbai", "Mumbai", "Delhi", "Chennai", "Kolakatta")).filter(_.equals("Mumbai")).count
```

```
rdda: Long = 2
```

```
// applying filter which contain a in city data
val rdda = sc.parallelize(List("Mumbai", "Delhi", "Chennai", "Kolakatta")).filter(_.contains("e")).collect
```

```
rdda: Array[String] = Array(Delhi, Chennai)
```

```
// Creating an rdda with city,Count
val a = sc.parallelize(List(("Mumbai",4000),( "Delhi", 2000), ("Chennai",1000),("Kolakatta", 7000)))
```

```
a: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[18] at parallelize at command-3160921003574763:2
```

```
// Perform filter operation where value euals 4000
// _._1 is for key _._2 is for value
val a = sc.parallelize(List(("Mumbai",4000),( "Delhi", 2000), ("Chennai",1000),("Kolakatta",
7000))).filter(_._2.equals(4000)).collect
```

```
a: Array[(String, Int)] = Array((Mumbai,4000))
```

```
// Perform filter operation where values grather than 3000
val a = sc.parallelize(List(("Mumbai",4000),( "Delhi", 2000), ("Chennai",1000),("Kolakatta", 7000))).filter(_._2 > 3000).collect

// _._1 is for key _._2 is for value
```

```
a: Array[(String, Int)] = Array((Mumbai,4000), (Kolakatta,7000))
```

```
// Perform filter which start with C

val a = sc.parallelize(List(("Mumbai",4000),( "Delhi", 2000), ("Chennai",1000),("Kolakatta", 7000))).filter(a=>
a._1.startsWith("C")).collect

// Perform filter by range between 3000, 9000

val b = sc.parallelize(List((4000,"Mumbai"),(2000, "Delhi"), (1000, "Chennai"),(7000, "Kolakatta"))).filterByRange(3000,
9000).collect
```

```
a: Array[(String, Int)] = Array((Chennai,1000))
b: Array[(Int, String)] = Array((4000,Mumbai), (7000,Kolakatta))
```

```
// sample (flase/true, fraction, seed)
// false - can not have repeated values
// true- will have repeated values
//  seed - result will be same if the seed is kept same
```

```
val a = sc.parallelize(1 to 100)
```

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[33] at parallelize at command-3160921003574768:1
```

```
a.sample(false, .2,5).collect
```

```
res17: Array[Int] = Array(2, 3, 7, 8, 13, 22, 25, 39, 43, 56, 59, 61, 66, 71, 73, 79, 83, 92)
```

```
a.sample(true, .2,5).collect
```

```
res19: Array[Int] = Array(2, 3, 9, 10, 13, 14, 19, 20, 24, 40, 43, 46, 56, 61, 62, 67, 80, 81, 82, 82, 92)
```

```
a.sample(false, .2).collect
```

```
res20: Array[Int] = Array(2, 18, 23, 32, 38, 45, 46, 47, 57, 60, 64, 73, 74, 81, 83, 86, 96)
```

```
a.sample(false,1,5).collect
```

```
res21: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
96, 97, 98, 99, 100)
```

```
val a = sc.parallelize(List(1,2,1,1,1,2))
```

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[38] at parallelize at command-3160921003574773:1
```

```
a.sample(true, .4,5).collect
```

```
res22: Array[Int] = Array(1)
```

```
val a = sc.parallelize(1 to 7)
```

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[40] at parallelize at command-3160921003574775:1
```

```
val b = sc.parallelize(5 to 10)
```

```
b: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[41] at parallelize at command-3160921003574776:1
```

```
a.union(b).collect
```

```
res23: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 5, 6, 7, 8, 9, 10)
```

```
a.intersection(b).collect
```

```
res24: Array[Int] = Array(5, 6, 7)
```

```
a.union(b).distinct.collect
```

```
res25: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

```
val b = sc.parallelize(1 to 9,3)
```

```
b: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[53] at parallelize at command-3160921003574780:1
```

```
b.mapPartitions(x=>List(x.next).iterator).collect
```

```
res28: Array[Int] = Array(1, 4, 7)
```

```
val a = sc.parallelize(1 to 9,4)
```

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[57] at parallelize at command-3160921003574782:1
```

```
a.mapPartitions(x=>List(x.next).iterator).collect
```

```
res29: Array[Int] = Array(1, 3, 5, 7)
```

```
def practfunct(index: Int, iter:Iterator[(Int)]): Iterator[String] = {
  iter.toList.map(x => "[index :" +index + ", val : " +x +"]").iterator
}
```

```
practfunct: (index: Int, iter: Iterator[Int])Iterator[String]
```

```
val a  = sc.parallelize(List(1,2,3,4,5,6),2)
```

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[59] at parallelize at command-3160921003574785:1
```

```
a.collect
```

```
res30: Array[Int] = Array(1, 2, 3, 4, 5, 6)
```

```
a.mapPartitionsWithIndex(practfunct).collect
```

```
res31: Array[String] = Array([index :0, val : 1], [index :0, val : 2], [index :0, val : 3], [index :1, val : 4], [index :1, val :
5], [index :1, val : 6])
```

```
val a = sc.parallelize(List(1,2,3,4,5,6),3)
```

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[61] at parallelize at command-3160921003574788:1
```

```
a.mapPartitionsWithIndex(practfunct).collect
```

```
res32: Array[String] = Array([index :0, val : 1], [index :0, val : 2], [index :1, val : 3], [index :1, val : 4], [index :2, val :
5], [index :2, val : 6])
```

```
val boradcastVar = sc.broadcast(Array(1,2,3))
boradcastVar.value
```

```
boradcastVar: org.apache.spark.broadcast.Broadcast[Array[Int]] = Broadcast(41)
res33: Array[Int] = Array(1, 2, 3)

a: Double = 2500.0
res34: Double = 2500.0
```

databricks 2023-08-27 - How to upload File DBFS Example

(https://databricks.com)

Overview

This notebook will show you how to create and query a table or DataFrame that you uploaded to DBFS. DBFS (https://docs.databricks.com/user-guide/dbfs-databricks-file-system.html) is a Databricks File System that allows you to store data for querying inside of Databricks. This notebook assumes that you have a file already inside of DBFS that you would like to read from.

This notebook is written in **Python** so the default cell type is Python. However, you can use different languages by using the `%LANGUAGE` syntax. Python, Scala, SQL, and R are all supported.

```
# File location and type
file_location = "/FileStore/tables/airline_safety-1.csv"
file_type = "csv"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

Table

	_c0	_c1	_c2	_c3	_c4	_c5
1	airline	avail_seat_km_per_week	incidents_85_99	fatal_accidents_85_99	fatalities_85_99	incidents_00_14
2	Aer Lingus	320906734	2	0	0	0
3	Aeroflot*	1197672318	76	14	128	6
4	Aerolineas Argentinas	385803648	6	0	0	1
5	Aeromexico*	596871813	3	1	64	5
6	Air Canada	1865253802	2	0	0	2
7	Air France	3004002661	14	4	79	6

57 rows

```
# Create a view or table

temp_table_name = "airline_safety-1_csv"

df.createOrReplaceTempView(temp_table_name)
```

```
AnalysisException: Invalid view name: airline_safety-1_csv.
```

```
%sql

/* Query the created temp table in a SQL cell */

select * from `airline_safety-1_csv`
```

databricks 2023-08-28 - DBFS Example

(https://databricks.com)

Overview

This notebook will show you how to create and query a table or DataFrame that you uploaded to DBFS. DBFS (https://docs.databricks.com/user-guide/dbfs-databricks-file-system.html) is a Databricks File System that allows you to store data for querying inside of Databricks. This notebook assumes that you have a file already inside of DBFS that you would like to read from.

This notebook is written in **Python** so the default cell type is Python. However, you can use different languages by using the `%LANGUAGE` syntax. Python, Scala, SQL, and R are all supported.

```
# File location and type
file_location = "/FileStore/tables/file.json"
file_type = "json"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

```
# Create a view or table

temp_table_name = "file_json"

df.createOrReplaceTempView(temp_table_name)
```

```
%sql

/* Query the created temp table in a SQL cell */

select * from `file_json`
```

databricks G3G4 transformation extended Notebook 2023-08-28 11:05:13

(https://databricks.com)

```
print("helloworld")
```

```
helloworld
```

```
val accum = sc.longAccumulator("Sum Accumulator")
accum.value
```

```
accum: org.apache.spark.util.LongAccumulator = LongAccumulator(id: 0, name: Some(Sum Accumulator), value: 0)
res1: Long = 0
```

```
val broadcastVar = sc.broadcast(Array(1,2,3))
```

```
broadcastVar: org.apache.spark.broadcast.Broadcast[Array[Int]] = Broadcast(0)
```

```
broadcastVar.value
```

```
res2: Array[Int] = Array(1, 2, 3)
```

```
import org.apache.spark.sql.SparkSession
val spark = SparkSession.builder().appName("SparkExample").master("local").getOrCreate()
```

```
import org.apache.spark.sql.SparkSession
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@eb4526a
```

```
val states = Map(("NY", "New York"),("CA", "California"),("FL", "Florida"))
```

```
states: scala.collection.immutable.Map[String,String] = Map(NY -> New York, CA -> California, FL -> Florida)
```

```
val countries = Map(("USA", "United states of america"), ("IN", "India"))
```

```
countries: scala.collection.immutable.Map[String,String] = Map(USA -> United states of america, IN -> India)
```

```
val broadcaststates = spark.sparkContext.broadcast(states)
```

```
broadcaststates: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Map[String,String]] = Broadcast(1)
```

```
val broadcastcountries = spark.sparkContext.broadcast(countries)
```

```
broadcastcountries: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Map[String,String]] = Broadcast(2)
```

```
val data = Seq(("james", "Suraj", "USA", "CA"),
              ("mes", "Sura", "USA", "NY"),
              ("megan", "mathews", "USA", "CA"),
              ("Maria", "Mohan", "USA", "FL"))
```

```
data: Seq[(String, String, String, String)] = List((james,Suraj,USA,CA), (mes,Sura,USA,NY), (megan,mathews,USA,CA), (Maria,Mohan,US
A,FL))
```

```
val rdd = spark.sparkContext.parallelize(data)
```

```
rdd: org.apache.spark.rdd.RDD[(String, String, String, String)] = ParallelCollectionRDD[0] at parallelize at command-260941957792899
1:1
```

```
val rdd2 = rdd.map(f=> {
  val country = f._3
  val state = f._4
  val fullCountry = broadcastcountries.value.get(country).get
  val fullState = broadcaststates.value.get(state).get
  (f._1, f._2,fullCountry, fullState)
})
```

```
rdd2: org.apache.spark.rdd.RDD[(String, String, String, String)] = MapPartitionsRDD[1] at map at command-2609419577928992:1
```

```
println(rdd2.collect().mkString("\n"))
```

```
(james,Suraj,United states of america,California)
(mes,Sura,United states of america,New York)
(megan,mathews,United states of america,California)
(Maria,Mohan,United states of america,Florida)
```

```
val states = Map(("NY", "New York"),("CA", "California"),("FL", "Florida"))
val countries = Map(("USA", "United states of america"), ("IN", "India"))
val broadcaststates = spark.sparkContext.broadcast(states)
val broadcastcountries = spark.sparkContext.broadcast(countries)
val data = Seq(("james", "Suraj", "USA", "CA"),
              ("mes", "Sura", "USA", "NY"),
              ("megan", "mathews", "USA", "CA"),
              ("Maria", "Mohan", "USA", "FL"))
val columns = Seq("First name", "Lastname", "Country", "States")
import spark.sqlContext.implicits._
val df = data.toDF(columns:_*)
val df2 = df.map(row =>{
  val country = row.getString(2)
  val state = row.getString(3)
    val fullcountry = broadcastcountries.value.get(country).get
    val fullstate = broadcaststates.value.get(state).get
    (row.getString(0), row.getString(1), fullcountry,fullstate)
}).toDF(columns:_*)
```

```
states: scala.collection.immutable.Map[String,String] = Map(NY -> New York, CA -> California, FL -> Florida)
countries: scala.collection.immutable.Map[String,String] = Map(USA -> United states of america, IN -> India)
broadcaststates: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Map[String,String]] = Broadcast(4)
broadcastcountries: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Map[String,String]] = Broadcast(5)
data: Seq[(String, String, String, String)] = List((james,Suraj,USA,CA), (mes,Sura,USA,NY), (megan,mathews,USA,CA), (Maria,Mohan,US
A,FL))
columns: Seq[String] = List(First name, Lastname, Country, States)
import spark.sqlContext.implicits._
df: org.apache.spark.sql.DataFrame = [First name: string, Lastname: string ... 2 more fields]
df2: org.apache.spark.sql.DataFrame = [First name: string, Lastname: string ... 2 more fields]
```

```
df2.show(false)
```

```
+----------+--------+------------------------+----------+
|First name|Lastname|Country                 |States    |
+----------+--------+------------------------+----------+
|james     |Suraj   |United states of america|California|
|mes       |Sura    |United states of america|New York  |
|megan     |mathews |United states of america|California|
|Maria     |Mohan   |United states of america|Florida   |
+----------+--------+------------------------+----------+
var longAcc = spark.sparkContext.longAccumulator("SumAccumulator")
```

```
longAcc: org.apache.spark.util.LongAccumulator = LongAccumulator(id: 151, name: Some(SumAccumulator), value: 0)
```

```
val rdd  = spark.sparkContext.parallelize(Array(1,2,3))
```

```
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[4] at parallelize at command-2609419577928997:1
```

```
rdd.foreach(x => longAcc.add(x))
```

```
println(longAcc.value)
```

```
6
```

```
spark.sparkContext.setLogLevel("Error")
```

```
val inputRDD = spark.sparkContext.parallelize(List(("Z", 1),("A", 20),("B", 30),("C", 40),("B", 30),("B", 60)))
```

```
inputRDD: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[5] at parallelize at command-2609419577929001:1
```

```
val listRDD = spark.sparkContext.parallelize(List(1,2,3,4,5,3,2))
```

```
listRDD: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[6] at parallelize at command-2609419577929002:1
```

```
// aggregate

def param0 = (accu:Int, v:Int) => accu + v
def param1 = (accu1:Int, accu2:Int) => accu1 + accu2
print("Aggregate :" +listRDD.aggregate(0)(param0, param1))
```

```
Aggregate :20param0: (Int, Int) => Int
param1: (Int, Int) => Int
```

```
// aggregate

def param3 = (accu:Int, v:(String, Int)) => accu + v._2
def param4 = (accu1:Int, accu2:Int) => accu1 + accu2
print("Aggregate :" +inputRDD.aggregate(0)(param3, param4))
```

```
Aggregate :181param3: (Int, (String, Int)) => Int
param4: (Int, Int) => Int
```

```
// tree Aggregate

def param8 = (accu:Int, v: Int) => accu + v
def param9 = (accu1:Int, accu2:Int) => accu1 + accu2
print("Tree Aggregate :" +listRDD.treeAggregate(0)(param8, param9))
```

```
Tree Aggregate :20param8: (Int, Int) => Int
param9: (Int, Int) => Int
```

```
// fold action

println("fold :" +listRDD.fold(0) {(acc,v) =>
 val sum = acc+ v
  sum
})
```

```
fold :20
```

```
println("fold :" +inputRDD.fold(("total", 0)) {(acc:(String, Int), v:(String, Int)) =>
 val sum = acc._2+v._2
  ("Total",sum)
})
```

```
fold :(Total,181)
```

```
// what is difference between aggregate, tree aggregate and fold
```

```
val data = listRDD.collect()
```

```
data: Array[Int] = Array(1, 2, 3, 4, 5, 3, 2)
```

```
data.foreach(println)
```

```
1
2
3
4
5
3
2
```

```
// reduce

println("Reduce :" +listRDD.reduce(_ + _))
```

```
Reduce :20
```

```
println("Reduce :" +listRDD.reduce(_ - _))
```

```
Reduce :-10
```

```
println("reduce Alternate :" + inputRDD.reduce((x,y)=> ("Total",x._2+y._2)))
```

```
reduce Alternate :(Total,181)
```

```
println("tree reduce: " +listRDD.treeReduce(_+_))
```

```
tree reduce: 20
```

```
inputRDD.count
```

```
res24: Long = 6
```

```
inputRDD.countApproxDistinct()
```

```
res26: Long = 5
```

```
listRDD.first
```

```
res27: Int = 1
```

```
listRDD.take(2)
```

```
res28: Array[Int] = Array(1, 2)
```

```
listRDD.top(2)
```

```
res29: Array[Int] = Array(5, 4)
```

```
inputRDD.min._2
```

```
res30: Int = 20
```

```
inputRDD.max._2
```

```
res31: Int = 1
```

```
inputRDD.min
```

```
res32: (String, Int) = (A,20)
```

```
inputRDD.max
```

```
res33: (String, Int) = (Z,1)
```

```
<console>:8: error: identifier expected but integer literal found.
       partitionied_rdd = rdd.partitionBy(2, lambda x: x % 2)
                                                         ^
```

databricks NLP_PySpark Notebook 2023-10-16 10:59:49

(https://databricks.com)

```
# create spark session
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName('nlp').getOrCreate()
```

```
df = spark.createDataFrame([(1, 'I Really Liked this movie'),
                            (2, 'I Would recommend this movie to my friends'),
                           (3, 'movie was alright but acting was horrible'),
                           (4, 'I am never watching that movie ever again')],
                           ['user_id', 'review'])
```

```
df.show(5,False)
```

```
+-------+------------------------------------------+
|user_id|review                                    |
+-------+------------------------------------------+
|1      |I Really Liked this movie                 |
|2      |I Would recommend this movie to my friends|
|3      |movie was alright but acting was horrible |
|4      |I am never watching that movie ever again |
+-------+------------------------------------------+
```

```
# tokenization
```

```
from pyspark.ml.feature import Tokenizer
```

```
tokenization = Tokenizer(inputCol='review', outputCol='tokens')
```

```
tokenized_df = tokenization.transform(df)
```

```
tokenized_df.show(4,False)
```

```
+-------+------------------------------------------+---------------------------------------------------+
|user_id|review                                    |tokens                                             |
+-------+------------------------------------------+---------------------------------------------------+
|1      |I Really Liked this movie                 |[i, really, liked, this, movie]                    |
|2      |I Would recommend this movie to my friends|[i, would, recommend, this, movie, to, my, friends]|
|3      |movie was alright but acting was horrible |[movie, was, alright, but, acting, was, horrible]  |
|4      |I am never watching that movie ever again |[i, am, never, watching, that, movie, ever, again] |
+-------+------------------------------------------+---------------------------------------------------+
```

```
# Stopwords removal
```

```
from pyspark.ml.feature import StopWordsRemover
```

```
stopword_removal = StopWordsRemover(inputCol='tokens', outputCol='refined_tokens')
```

```
refined_df = stopword_removal.transform(tokenized_df)
```

```
refined_df.select(['user_id', 'tokens', 'refined_tokens']).show(10,False)
```

```
+-------+--------------------------------------------------+-------------------------------+
|user_id|tokens                                            |refined_tokens                 |
+-------+--------------------------------------------------+-------------------------------+
|1      |[i, really, liked, this, movie]                   |[really, liked, movie]         |
|2      |[i, would, recommend, this, movie, to, my, friends]|[recommend, movie, friends]    |
|3      |[movie, was, alright, but, acting, was, horrible] |[movie, alright, acting, horrible]|
|4      |[i, am, never, watching, that, movie, ever, again]|[never, watching, movie, ever] |
+-------+--------------------------------------------------+-------------------------------+
```

```
# Count Vectorizer
```

```
from pyspark.ml.feature import CountVectorizer
```

```
count_vec = CountVectorizer(inputCol='refined_tokens', outputCol='features')
```

```
cv_df= count_vec.fit(refined_df).transform(refined_df)
```

```
cv_df.select(['user_id', 'refined_tokens', 'features']).show(4, False)
```

```
+-------+----------------------------------+-------------------------------------+
|user_id|refined_tokens                    |features                             |
+-------+----------------------------------+-------------------------------------+
|1      |[really, liked, movie]            |(11,[0,3,7],[1.0,1.0,1.0])           |
|2      |[recommend, movie, friends]       |(11,[0,6,9],[1.0,1.0,1.0])           |
|3      |[movie, alright, acting, horrible]|(11,[0,2,4,5],[1.0,1.0,1.0,1.0])     |
|4      |[never, watching, movie, ever]    |(11,[0,1,8,10],[1.0,1.0,1.0,1.0])    |
+-------+----------------------------------+-------------------------------------+
```

```
count_vec.fit(refined_df).vocabulary
```

```
Out[23]: ['movie',
 'liked',
 'horrible',
 'friends',
 'ever',
 'alright',
 'recommend',
 'acting',
 'really',
 'never',
 'watching']
```

```
# tf_idf
```

```
from pyspark.ml.feature import HashingTF,IDF
```

```
hashing_vec = HashingTF(inputCol="refined_tokens", outputCol='tf_features')
```

```
hashing_df = hashing_vec.transform(refined_df)
```

```
hashing_df.select(['user_id', 'refined_tokens', 'tf_features']).show(4, False)
```

```
+-------+----------------------------------+---------------------------------------------------------------------+
|user_id|refined_tokens                    |tf_features                                                          |
+-------+----------------------------------+---------------------------------------------------------------------+
|1      |[really, liked, movie]            |(262144,[99172,210223,229264],[1.0,1.0,1.0])                         |
|2      |[recommend, movie, friends]       |(262144,[68228,130047,210223],[1.0,1.0,1.0])                         |
|3      |[movie, alright, acting, horrible]|(262144,[95685,171118,210223,236263],[1.0,1.0,1.0,1.0])|
|4      |[never, watching, movie, ever]    |(262144,[63139,113673,203802,210223],[1.0,1.0,1.0,1.0])|
+-------+----------------------------------+---------------------------------------------------------------------+
```

```
tf_idf_vec = IDF(inputCol='tf_features', outputCol='tf_idf_features')
```

```
tf_idf_df = tf_idf_vec.fit(hashing_df).transform(hashing_df)
```

```
tf_idf_df.select(['user_id', 'tf_idf_features']).show(4, False)
```

```
+-------+----------------------------------------------------------------------------------------------------------------------+
|user_id|tf_idf_features                                                                                                       |
+-------+----------------------------------------------------------------------------------------------------------------------+
|1      |(262144,[99172,210223,229264],[0.9162907318741551,0.0,0.9162907318741551])                                            |
|2      |(262144,[68228,130047,210223],[0.9162907318741551,0.9162907318741551,0.0])                                            |
|3      |(262144,[95685,171118,210223,236263],[0.9162907318741551,0.9162907318741551,0.0,0.9162907318741551])|
|4      |(262144,[63139,113673,203802,210223],[0.9162907318741551,0.9162907318741551,0.9162907318741551,0.0])|
+-------+----------------------------------------------------------------------------------------------------------------------+
```

```
# classification
```

```
text_df = spark.read.csv('dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/Movie_reviews.csv', inferSchema=True,
header=True,sep=',')
```

```
text_df.printSchema()
```

```
root
 |-- Review: string (nullable = true)
 |-- Sentiment: string (nullable = true)
```

```
text_df.count()
```

```
Out[36]: 7087
```

```
from pyspark.sql.functions import rand
```

```
text_df.orderBy(rand()).show(10,False)
```

```
+-------------------------------------------------------------------------+---------+
|Review                                                                   |Sentiment|
+-------------------------------------------------------------------------+---------+
|Harry Potter is AWESOME I don't care if anyone says differently!..      |1        |
|Harry Potter is AWESOME I don't care if anyone says differently!..      |1        |
|Always knows what I want, not guy crazy, hates Harry Potter..           |0        |
|I either LOVE Brokeback Mountain or think it's great that homosexuality|1        |
|dudeee i LOVED brokeback mountain!!!!                                   |1        |
|I liked the movie Brokeback Mountain.                                   |1        |
|I love Harry Potter.                                                    |1        |
|i hate brokeback mountain!!                                             |0        |
|Brokeback Mountain was an AWESOME movie.                                |1        |
|Harry Potter dragged Draco Malfoy ’ s trousers down past his hips and  |0        |
+-------------------------------------------------------------------------+---------+
only showing top 10 rows
```

```
text_df= text_df.filter(((text_df.Sentiment =='1') | (text_df.Sentiment =='0')))
```

```
text_df.count()
```

```
Out[41]: 6990
```

```
text_df.groupBy("Sentiment").count().show()
```

```
+---------+-----+
|Sentiment|count|
+---------+-----+
|        0| 3081|
|        1| 3909|
+---------+-----+
```

```
text_df.printSchema()
```

```
root
 |-- Review: string (nullable = true)
 |-- Sentiment: string (nullable = true)
```

```
text_df = text_df.withColumn("Label", text_df.Sentiment.cast('float')).drop('Sentiment')
```

```
text_df.orderBy(rand()).show(10,False)
```

```
+------------------------------------------------------------------------+-----+
|Review                                                                  |Label|
+------------------------------------------------------------------------+-----+
|I either LOVE Brokeback Mountain or think it's great that homosexuality |1.0  |
|Da Vinci Code sucks.                                                    |0.0  |
|we're gonna like watch Mission Impossible or Hoot.(                     |1.0  |
|I love Brokeback Mountain....                                           |1.0  |
|I would give you more Kudos on this blog but you had to go and talk abou|0.0  |
|So Brokeback Mountain was really depressing.                            |0.0  |
|No matter how much I love Brokeback Mountain, Crash definitely deserved |1.0  |
|I think I hate Harry Potter because it outshines much better reading mat|0.0  |
|friday hung out with kelsie and we went and saw The Da Vinci Code SUCKED|0.0  |
|da vinci code sucks...                                                  |0.0  |
+------------------------------------------------------------------------+-----+
only showing top 10 rows
```

```
text_df.groupBy('label').count().show()
```

```
+-----+-----+
|label|count|
+-----+-----+
|  1.0| 3909|
|  0.0| 3081|
+-----+-----+
```

```
# Add length of the dataframe
from pyspark.sql.functions import length
```

```
text_df = text_df.withColumn('length', length(text_df['Review']))
```

```
text_df.orderBy(rand()).show(10,False)
```

```
+------------------------------------------------------------------------+-----+------+
|Review                                                                  |Label|length|
+------------------------------------------------------------------------+-----+------+
|Da Vinci Code = Up, Up, Down, Down, Left, Right, Left, Right, B, A, SUCK|0.0  |72    |
|I love The Da Vinci Code...                                             |1.0  |27    |
|Combining the opinion / review from Gary and Gin Zen, The Da Vinci Code |0.0  |71    |
|The Da Vinci Code was absolutely AWESOME!                               |1.0  |41    |
|You know, the Harry Potter books are decent enough, and I ’ m glad the  |1.0  |70    |
|Brokeback mountain was beautiful...                                     |1.0  |35    |
|The Da Vinci Code was absolutely AWESOME!                               |1.0  |41    |
|I want to be here because I love Harry Potter, and I really want a place|1.0  |72    |
|I love Brokeback Mountain....                                           |1.0  |29    |
|"I liked the first "" Mission Impossible."                              |1.0  |42    |
+------------------------------------------------------------------------+-----+------+
only showing top 10 rows
```

```
text_df.groupBy('Label').agg({'Length': 'mean'}).show()
```

```
+-----+-----------------+
|Label|      avg(Length)|
+-----+-----------------+
|  1.0|47.61882834484523|
|  0.0|50.95845504706264|
+-----+-----------------+
```

```
# data Cleaning
```

```
tokenization= Tokenizer(inputCol='Review', outputCol='tokens')
```

```
tokenized_df= tokenization.transform(text_df)
```

```
tokenized_df.show()
```

```
+--------------------+-----+------+--------------------+
|              Review|Label|length|              tokens|
+--------------------+-----+------+--------------------+
|The Da Vinci Code...|  1.0|    39|[the, da, vinci, ...|
|this was the firs...|  1.0|    72|[this, was, the, ...|
|i liked the Da Vi...|  1.0|    32|[i, liked, the, d...|
|i liked the Da Vi...|  1.0|    32|[i, liked, the, d...|
|I liked the Da Vi...|  1.0|    72|[i, liked, the, d...|
|that's not even a...|  1.0|    72|[that's, not, eve...|
|I loved the Da Vi...|  1.0|    72|[i, loved, the, d...|
|i thought da vinc...|  1.0|    57|[i, thought, da, ...|
|The Da Vinci Code...|  1.0|    45|[the, da, vinci, ...|
|I thought the Da ...|  1.0|    51|[i, thought, the,...|
|The Da Vinci Code...|  1.0|    68|[the, da, vinci, ...|
|The Da Vinci Code...|  1.0|    62|[the, da, vinci, ...|
|then I turn on th...|  1.0|    66|[then, i, turn, o...|
|The Da Vinci Code...|  1.0|    34|[the, da, vinci, ...|
|i love da vinci c...|  1.0|    24|[i, love, da, vin...|
|i loved da vinci ...|  1.0|    23|[i, loved, da, vi...|
|TO NIGHT:: THE DA...|  1.0|    52|[to, night::, the...|
|THE DA VINCI CODE...|  1.0|    40|[the, da, vinci, ...|
```

```
stopword_removal = StopWordsRemover(inputCol='tokens', outputCol='refined_tokens')
```

```
refined_text_df = stopword_removal.transform(tokenized_df)
```

```
refined_text_df.show()
```

```
+--------------------+-----+------+--------------------+--------------------+
|              Review|Label|length|              tokens|      refined_tokens|
+--------------------+-----+------+--------------------+--------------------+
|The Da Vinci Code...|  1.0|    39|[the, da, vinci, ...|[da, vinci, code,...|
|this was the firs...|  1.0|    72|[this, was, the, ...|[first, clive, cu...|
|i liked the Da Vi...|  1.0|    32|[i, liked, the, d...|[liked, da, vinci...|
|i liked the Da Vi...|  1.0|    32|[i, liked, the, d...|[liked, da, vinci...|
|I liked the Da Vi...|  1.0|    72|[i, liked, the, d...|[liked, da, vinci...|
|that's not even a...|  1.0|    72|[that's, not, eve...|[even, exaggerati...|
|I loved the Da Vi...|  1.0|    72|[i, loved, the, d...|[loved, da, vinci...|
|i thought da vinc...|  1.0|    57|[i, thought, da, ...|[thought, da, vin...|
|The Da Vinci Code...|  1.0|    45|[the, da, vinci, ...|[da, vinci, code,...|
|I thought the Da ...|  1.0|    51|[i, thought, the,...|[thought, da, vin...|
|The Da Vinci Code...|  1.0|    68|[the, da, vinci, ...|[da, vinci, code,...|
|The Da Vinci Code...|  1.0|    62|[the, da, vinci, ...|[da, vinci, code,...|
|then I turn on th...|  1.0|    66|[then, i, turn, o...|[turn, light, rad...|
|The Da Vinci Code...|  1.0|    34|[the, da, vinci, ...|[da, vinci, code,...|
|i love da vinci c...|  1.0|    24|[i, love, da, vin...|[love, da, vinci,...|
|i loved da vinci ...|  1.0|    23|[i, loved, da, vi...|[loved, da, vinci...|
|TO NIGHT:: THE DA...|  1.0|    52|[to, night::, the...|[night::, da, vin...|
```

```
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType
from pyspark.sql.functions import *
```

```
len_udf = udf(lambda s: len(s), IntegerType())

refined_text_df = refined_text_df.withColumn("token_count", len_udf(col('refined_tokens')))
```

```
refined_text_df.orderBy(rand()).show(10)
```

```
+--------------------+-----+------+--------------------+--------------------+-----------+
|              Review|Label|length|              tokens|      refined_tokens|token_count|
+--------------------+-----+------+--------------------+--------------------+-----------+
|The Da Vinci Code...|  1.0|    30|[the, da, vinci, ...|[da, vinci, code,...|          4|
|i heard da vinci ...|  0.0|    53|[i, heard, da, vi...|[heard, da, vinci...|          9|
|Which is why i sa...|  1.0|    72|[which, is, why, ...|[said, silent, hi...|          8|
|I used to hate Ha...|  0.0|    28|[i, used, to, hat...|[used, hate, harr...|          4|
|I finished The Da...|  1.0|    54|[i, finished, the...|[finished, da, vi...|          6|
|Then snuck into B...|  0.0|    72|[then, snuck, int...|[snuck, brokeback...|          5|
|""" I hate Harry ...|  0.0|    25|[""", i, hate, ha...|[""", hate, harry...|          4|
|Brokeback Mountai...|  0.0|    37|[brokeback, mount...|[brokeback, mount...|          3|
|we're gonna like ...|  1.0|    51|[we're, gonna, li...|[gonna, like, wat...|          6|
|Brokeback Mountai...|  0.0|    30|[brokeback, mount...|[brokeback, mount...|          3|
+--------------------+-----+------+--------------------+--------------------+-----------+
only showing top 10 rows
```

```
count_vec = CountVectorizer(inputCol='refined_tokens', outputCol='features')
```

```
cv_text_df = count_vec.fit(refined_text_df).transform(refined_text_df)
```

```
cv_text_df.select(['refined_tokens', 'token_count','features','Label']).show(10)
```

```
+--------------------+-----------+--------------------+-----+
|      refined_tokens|token_count|            features|Label|
+--------------------+-----------+--------------------+-----+
|[da, vinci, code,...|          5|(2302,[0,1,4,43,2...|  1.0|
|[first, clive, cu...|          9|(2302,[11,51,229,...|  1.0|
|[liked, da, vinci...|          5|(2302,[0,1,4,52,3...|  1.0|
|[liked, da, vinci...|          5|(2302,[0,1,4,52,3...|  1.0|
|[liked, da, vinci...|          8|(2302,[0,1,4,52,7...|  1.0|
|[even, exaggerati...|          6|(2302,[46,229,272...|  1.0|
|[loved, da, vinci...|          8|(2302,[0,1,22,30,...|  1.0|
|[thought, da, vin...|          7|(2302,[0,1,4,228,...|  1.0|
|[da, vinci, code,...|          6|(2302,[0,1,4,33,2...|  1.0|
|[thought, da, vin...|          7|(2302,[0,1,4,223,...|  1.0|
+--------------------+-----------+--------------------+-----+
only showing top 10 rows
```

```
# select data for building work
model_text_df = cv_text_df.select(['features', 'token_count','Label'])
```

```
from pyspark.ml.feature import VectorAssembler
```

```
df_assembler = VectorAssembler(inputCols=['features', 'token_count'], outputCol='features_vec')
model_text_df =df_assembler.transform(model_text_df)
```

```
model_text_df.printSchema()
```

```
root
 |-- features: vector (nullable = true)
 |-- token_count: integer (nullable = true)
 |-- Label: float (nullable = true)
 |-- features_vec: vector (nullable = true)
```

```
from pyspark.ml.classification import LogisticRegression
```

```
# splitting tha train data
training_df , test_df = model_text_df.randomSplit([0.75, 0.25])
```

```
training_df.groupBy('Label').count().show()
```

```
+-----+-----+
|Label|count|
+-----+-----+
|  1.0| 2916|
|  0.0| 2296|
+-----+-----+
```

```
test_df.groupBy('Label').count().show()
```

```
+-----+-----+
|Label|count|
+-----+-----+
|  1.0|  993|
|  0.0|  785|
+-----+-----+
log_reg = LogisticRegression(featuresCol = 'features_vec', labelCol = 'Label').fit(training_df)
```

```
results = log_reg.evaluate(test_df).predictions
```

```
results.show()
```

```
+--------------------+-----------+-----+--------------------+--------------------+--------------------+----------+
|            features|token_count|Label|         features_vec|       rawPrediction|         probability|prediction|
+--------------------+-----------+-----+--------------------+--------------------+--------------------+----------+
|(2302,[0,1,4,5,30...|          5|  1.0|(2303,[0,1,4,5,30...|[-20.033223790660...|[1.99379935936333...|       1.0|
|(2302,[0,1,4,5,36...|          5|  1.0|(2303,[0,1,4,5,36...|[-36.496611275475...|[1.41163726618194...|       1.0|
|(2302,[0,1,4,5,75...|          5|  1.0|(2303,[0,1,4,5,75...|[-24.189057523923...|[3.12482567921014...|       1.0|
|(2302,[0,1,4,5,10...|          6|  1.0|(2303,[0,1,4,5,10...|[-24.600819449628...|[2.07014069441782...|       1.0|
|(2302,[0,1,4,12,1...|         10|  1.0|(2303,[0,1,4,12,1...|[-25.485560392607...|[8.54597766156175...|       1.0|
|(2302,[0,1,4,12,1...|          5|  1.0|(2303,[0,1,4,12,1...|[-30.350012582238...|[6.59412248487400...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
|(2302,[0,1,4,12,3...|          5|  1.0|(2303,[0,1,4,12,3...|[-31.821377509241...|[1.51408878001981...|       1.0|
```

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
```

```
# confusion matrix
true_positives = results[(results.Label == 1) & (results.prediction == 1)].count()
true_negatives = results[(results.Label == 0) & (results.prediction == 0)].count()
false_positives = results[(results.Label == 0) & (results.prediction == 1)].count()
false_negatives = results[(results.Label == 1) & (results.prediction == 0)].count()
```

```
recall = float(true_positives)/(true_positives + false_negatives)
print(recall)
```

```
0.9798590130916415
```

```
precision = float(true_positives)/(true_positives + false_positives)
print(precision)
```

```
0.9778894472361809
```

```
0.9763779527559056
```

databricks Linear_regression 09-10-2023 Notebook 2023-10-09 11:05:48

(https://databricks.com)

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('lin_reg').getOrCreate()
```

```
# import
from pyspark.ml.regression import LinearRegression
```

```
# Load Dataset
df = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/Linear_regression_d
```

```
# validate the size of the data
print((df.count(), len(df.columns)))
```

```
(1232, 6)
```

```
# explore the data
df.printSchema()
```

```
root
 |-- var_1: integer (nullable = true)
 |-- var_2: integer (nullable = true)
 |-- var_3: integer (nullable = true)
 |-- var_4: double (nullable = true)
 |-- var_5: double (nullable = true)
 |-- output: double (nullable = true)
```

```
# view statistical mesures of the data
df.describe().show(5,False)
```

```
+-------+-----------------+-----------------+------------------+--------------------+---------------------+--------------------+
|summary|var_1            |var_2            |var_3             |var_4               |var_5                |output              |
+-------+-----------------+-----------------+------------------+--------------------+---------------------+--------------------+
|count  |1232             |1232             |1232              |1232                |1232                 |1232                |
|mean   |715.0819805194806|715.0819805194806|80.90422077922078 |0.3263311688311693  |0.25927272727272715  |0.39734172077922014|
|stddev |91.5342940441652 |93.07993263118064|11.458139049993724|0.015012772334166148|0.012907228928000298|0.03326689862173776|
|min    |463              |472              |40                |0.277               |0.214                |0.301               |
|max    |1009             |1103             |116               |0.373               |0.294                |0.491               |
+-------+-----------------+-----------------+------------------+--------------------+---------------------+--------------------+
```

```
# sneak into the dataset
df.head(3)
```

```
Out[78]: [Row(var_1=734, var_2=688, var_3=81, var_4=0.328, var_5=0.259, output=0.418),
 Row(var_1=700, var_2=600, var_3=94, var_4=0.32, var_5=0.247, output=0.389),
 Row(var_1=712, var_2=705, var_3=93, var_4=0.311, var_5=0.247, output=0.417)]
```

```
# import corr function from pyspark functions
from pyspark.sql.functions import corr
```

```
# check for correlation & how to add variable using corr functions
df.select(corr('var_1', 'output')).show()
```

```
+-------------------+
|corr(var_1, output)|
+-------------------+
| 0.9187399607627283|
+-------------------+
```

```
# import
from pyspark.ml.linalg import Vector
from pyspark.ml.feature import VectorAssembler
```

```
# select the columns to create the input vector
df.columns
```

```
Out[82]: ['var_1', 'var_2', 'var_3', 'var_4', 'var_5', 'output']
```

```
# create the vector assembler
vec_assembler = VectorAssembler(inputCols = ['var_1', 'var_2', 'var_3', 'var_4', 'var_5'], outputCol='features')
```

```
# transform
features_df = vec_assembler.transform(df)
```

```
features_df.printSchema()
```

```
root
 |-- var_1: integer (nullable = true)
 |-- var_2: integer (nullable = true)
 |-- var_3: integer (nullable = true)
 |-- var_4: double (nullable = true)
 |-- var_5: double (nullable = true)
 |-- output: double (nullable = true)
 |-- features: vector (nullable = true)
```

```
features_df.select('features').show(5,False)
```

```
+------------------------------+
|features                      |
+------------------------------+
|[734.0,688.0,81.0,0.328,0.259]|
|[700.0,600.0,94.0,0.32,0.247] |
|[712.0,705.0,93.0,0.311,0.247]|
|[734.0,806.0,69.0,0.315,0.26] |
|[613.0,759.0,61.0,0.302,0.24] |
+------------------------------+
only showing top 5 rows
```

```
model_df = features_df.select('features', 'output')
```

```
model_df.show(5,False)
```

```
+------------------------------+---------+
|features                      |output|
+------------------------------+---------+
|[734.0,688.0,81.0,0.328,0.259]|0.418 |
|[700.0,600.0,94.0,0.32,0.247] |0.389 |
|[712.0,705.0,93.0,0.311,0.247]|0.417 |
|[734.0,806.0,69.0,0.315,0.26] |0.415 |
|[613.0,759.0,61.0,0.302,0.24] |0.378 |
+------------------------------+---------+
only showing top 5 rows
```

```
print((model_df.count(), len(model_df.columns)))
```

```
(1232, 2)
```

Spilt data - Train & Test Data

```
# sec
```

```
train_df, test_df = model_df.randomSplit([0.7,0.3])
```

```
print((train_df.count(), len(train_df.columns)))
```

```
(880, 2)
```

```
print((test_df.count(), len(test_df.columns)))
```

```
(352, 2)
```

```
train_df.describe().show()
```

```
+-------+--------------------+
|summary|              output|
+-------+--------------------+
|  count|                 880|
|   mean|  0.3979874999999994|
| stddev|0.033438290887800266|
|    min|               0.301|
|    max|               0.491|
+-------+--------------------+
```

```
# Build Linear Regression Model
```

```
lin_reg = LinearRegression(labelCol='output')
```

```
# fit the linear model on training dataset
lr_model = lin_reg.fit(train_df)
```

```
lr_model.intercept
```

```
Out[100]: 0.18626961441250367
```

```
print(lr_model.coefficients)
```

```
[0.0003371220458572444,5.991498706752434e-05,0.0002556814872303228,-0.6686443805619972,0.48044283582971575]
```

```
training_predictions = lr_model.evaluate(train_df)
```

```
training_predictions.meanSquaredError
```

```
Out[103]: 0.00013704630021768094
```

```
training_predictions.r2
```

```
Out[104]: 0.8772919740056858
```

```
# make predictions on the test data
test_results= lr_model.evaluate(test_df)
```

```
# view the residual errors based on the predictions
test_results.residuals.show(10)
```

```
+--------------------+
|           residuals|
+--------------------+
|0.009429260836439135|
|-9.42564848742444...|
|0.013713981224977412|
|-0.01210437809228...|
|-0.00679164860545...|
|0.010048355854590463|
| 8.97412483862492E-4|
|  -0.012396549858743|
|-0.01181377619641...|
|-0.00117635064512...|
+--------------------+
only showing top 10 rows
```

```
# coefficient of determination value for model
test_results.r2
```

```
Out[107]: 0.8476131884871057
```

```
test_results.rootMeanSquaredError
```

```
Out[108]: 0.012796007916146983
```

```
Out[109]: 0.00016373781859009624
```

databricks ML Example 09-1-23 Notebook 2023-10-09 11:54:18

(https://databricks.com)

```
# Extracting features form the text
some_text = spark.createDataFrame([
    ['''
Apache Spark achieves high performance for both batch
and streaming data, using a state-of-art DAG scheduler,
a query optimizer, and a physical execution engine.
''']
,['''
Apache Spark is a fast and general-purpose cluseter computing
system. It provides high-level APIs in Java, Scala, Python
and R, and an optimized engine that supports general execution
graphs. It also supports a rich set of higher-level tools including
Spark SQL for SQL and structured data processing, MLlib for machine
learning, GraphX for graph processing, and Spark Streaming.
''']
,['''
Machine learning is a filed of computer science that often uses
statistical techniques to give computers the ability to "learn"
(i.e., progessively improve performance on a specific task)
with data, without being explicitly programmed.
''']
], ['text'])
```

```
import pyspark.ml.feature as feat
import pyspark.sql.functions as f
```

```
splitter = feat.RegexTokenizer(
    inputCol = 'text',
    outputCol = 'text_split',
    pattern = '\s+|[,.\"]'
)
```

```
splitter.transform(some_text).select('text_split').take(1)
```

```
Out[7]: [Row(text_split=['apache', 'spark', 'achieves', 'high', 'performance', 'for', 'both', 'batch', 'and', 'streaming', 'data',
'using', 'a', 'state-of-art', 'dag', 'scheduler', 'a', 'query', 'optimizer', 'and', 'a', 'physical', 'execution', 'engine'])]
```

```
sw_remover = feat.StopWordsRemover(
    inputCol = splitter.getOutputCol()
    ,outputCol = 'no_stopWords'
)
```

```
sw_remover.transform(splitter.transform(some_text)).select('no_stopWords').take(1)
```

```
Out[9]: [Row(no_stopWords=['apache', 'spark', 'achieves', 'high', 'performance', 'batch', 'streaming', 'data', 'using', 'state-of-ar
t', 'dag', 'scheduler', 'query', 'optimizer', 'physical', 'execution', 'engine'])]
```

```
hasher = feat.HashingTF(
    inputCol = sw_remover.getOutputCol()
    ,outputCol = 'hashed'
    ,numFeatures = 20
)
```

```
hasher.transform(sw_remover.transform(splitter.transform(some_text))).select('hashed').take(1)
```

```
Out[11]: [Row(hashed=SparseVector(20, {0: 1.0, 3: 1.0, 6: 1.0, 8: 2.0, 9: 1.0, 11: 1.0, 12: 1.0, 13: 1.0, 15: 2.0, 16: 2.0, 17: 2.0,
18: 1.0, 19: 1.0}))]
```

```
idf = feat.IDF(
    inputCol= hasher.getOutputCol()
    ,outputCol='features'
)
```

```
idfModel = idf.fit(hasher.transform(sw_remover.transform(splitter.transform(some_text))))
```

```
idfModel.transform(hasher.transform(sw_remover.transform(splitter.transform(some_text)))).select('features').take(1)
```

```
Out[14]: [Row(features=SparseVector(20, {0: 0.0, 3: 0.0, 6: 0.2877, 8: 0.0, 9: 0.2877, 11: 0.2877, 12: 0.0, 13: 0.0, 15: 0.0, 16: 0.
0, 17: 1.3863, 18: 0.6931, 19: 0.0}))]
```

```
from pyspark.ml import Pipeline
```

```
pipeline = Pipeline(stages=[splitter, sw_remover, hasher, idf])
pipelineModel = pipeline.fit(some_text)
```

```
pipelineModel.transform(some_text).select('text', 'features').take(1)
```

```
Out[17]: [Row(text='\nApache Spark achieves high performance for both batch\nand streaming data, using a state-of-art DAG schedule
r,\na query optimizer, and a physical execution engine.\n', features=SparseVector(20, {0: 0.0, 3: 0.0, 6: 0.2877, 8: 0.0, 9: 0.2877,
11: 0.2877, 12: 0.0, 13: 0.0, 15: 0.0, 16: 0.0, 17: 1.3863, 18: 0.6931, 19: 0.0}))]
```

```
w2v = feat.Word2Vec(
    vectorSize=5
    ,minCount=2
    ,inputCol= sw_remover.getOutputCol()
    ,outputCol='vector'
)
```

```
model = w2v.fit(sw_remover.transform(splitter.transform(some_text)))
```

```
Out[21]: [Row(vector=DenseVector([0.0076, 0.0077, -0.0017, -0.015, -0.004]))]
```

databricks Demo of Page Rank Calculation 2023-10-30 16:15:32

(https://databricks.com)

```
import org.apache.spark.HashPartitioner

val links  = sc.parallelize(List(("MapR", List("A","B")), ))
```

```
import org.apache.spark.HashPartitioner
```

```
import org.apache.spark.HashPartitioner

val links = sc.parallelize(List(("MapR",List("Baidu","Blogger")),("Baidu", List("MapR")),("Blogger",List("Google","Baidu")),("Google",
()
var ranks = links.mapValues(v => 1.0)
```

```
import org.apache.spark.HashPartitioner
links: org.apache.spark.rdd.RDD[(String, List[String])] = ShuffledRDD[1] at partitionBy at command-1660944887008766:3
ranks: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[2] at mapValues at command-1660944887008766:4
```

```
// val ranks = contributions.reduceByKey((x,y) => x + y).mapValues(v => 0.15+0.85*v)
```

```
val contributions = links.join(ranks).flatMap { case (url, (links, rank)) => links.map(dest => (dest, rank / links.size)) }
```

```
contributions: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[6] at flatMap at command-1660944887008768:1
```

```
val ranks = contributions.reduceByKey((x, y) => x + y).mapValues(v => 0.15 + 0.85*v)
```

```
ranks: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[8] at mapValues at command-1660944887008769:1
```

```
ranks.collect
```

```
res1: Array[(String, Double)] = Array((Google,0.575), (MapR,1.8499999999999999), (Blogger,0.575), (Baidu,1.0))
```

```
val lines= spark.read.textFile("dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/links.txt").rdd
val iters = 20
val links = lines.map{ s =>
      val parts = s.split("\\s+")
      (parts(0), parts(1))
    }.distinct().groupByKey().cache()

 var ranks = links.mapValues(v => 1.0)

    for (i <- 1 to iters) {
      val contribs = links.join(ranks).values.flatMap{ case (urls, rank) =>
        val size = urls.size
        urls.map(url => (url, rank / size))
      }
      ranks = contribs.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
    }
    val output = ranks.collect()
    output.foreach(tup => println(tup._1 + " has rank:" + tup._2))
    println("==================")
    output.foreach(tup => println(tup._1 + " has rank:" + f"${tup._2}%.3f"))
    println("==================")
ranks.collect()
val r = ranks.toDF("URL", "PageRank")
r.show()
```

```
a has rank:1.0
b has rank:1.0455994483347224
c has rank:1.038759531084514
===================
a has rank:1.000
b has rank:1.046
c has rank:1.039
===================
+---+------------------+
|URL|          PageRank|
+---+------------------+
|  a|               1.0|
|  b|1.0455994483347224|
|  c| 1.038759531084514|
+---+------------------+

lines: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[883] at rdd at command-1660944887008771:1
iters: Int = 20
links: org.apache.spark.rdd.RDD[(String, Iterable[String])] = ShuffledRDD[888] at groupByKey at command-1660944887008771:6
ranks: org.apache.spark.rdd.RDD[(String, Double)] = MapPartitionsRDD[1029] at mapValues at command-1660944887008771:15
output: Array[(String, Double)] = Array((a,1.0), (b,1.0455994483347224), (c,1.038759531084514))
```

```
import org.apache.spark.sql.SparkSession
r.createOrReplaceTempView("Table_2")
val r1=sqlContext.sql("select PageRank from Table_2 where PageRank <2")
```

```
import org.apache.spark.sql.SparkSession
r1: org.apache.spark.sql.DataFrame = [PageRank: double]

formattedArray: org.apache.spark.sql.Dataset[String] = [value: string]
res16: Array[String] = Array(1.000, 1.046, 1.039)
```

databricks Graph Example 2023-10-23 11:00:49

(https://databricks.com)

```
%python
# file location and type
file_location = "dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/vertex.csv"
file_type = "csv"

# csv Options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# the applied options are for csv files.for other file types, these will be ignored.
df = spark.read.format(file_type) \
    .option("inferSchema", infer_schema)\
    .option("header",first_row_is_header)\
    .option("sep",delimiter)\
    .load(file_location)

display(df)
```

_c2
48
45
25
53
22
52

```
%python
# file location and type
file_location = "dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/edges.csv"
file_type = "csv"

# csv Options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# the applied options are for csv files.for other file types, these will be ignored.
df1 = spark.read.format(file_type) \
    .option("inferSchema", infer_schema)\
    .option("header",first_row_is_header)\
    .option("sep",delimiter)\
    .load(file_location)

display(df1)
```

Table

	_c0	_c1	_c2
1	6	1	Sister
2	1	2	Husband
3	2	1	Wife
4	5	1	Daughter
5	5	2	Daughter

6	3	1	Son
7	3	2	Son

12 rows

```
%scala
import org.apache.spark.rdd.RDD
```

```
import org.apache.spark.rdd.RDD
```

```
%scala
import org.apache.spark.graphx._
```

```
import org.apache.spark.graphx._
```

```
%scala
val vertexRDD = sc.textFile("dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/vertex.csv")
val edgeRDD = sc.textFile("dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/edges.csv")
edgeRDD.collect()
```

```
vertexRDD: org.apache.spark.rdd.RDD[String] = dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/vertex.csv MapPartitionsRDD[23]
at textFile at command-223217164383677:1
edgeRDD: org.apache.spark.rdd.RDD[String] = dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/edges.csv MapPartitionsRDD[25] at
textFile at command-223217164383677:2
res0: Array[String] = Array(6,1,Sister, 1,2,Husband, 2,1,Wife, 5,1,Daughter, 5,2,Daughter, 3,1,Son, 3,2,Son, 4,1,Friend, 1,5,Father,
1,3,Father, 2,5,Mother, 2,3,Mother)
```

```
%scala
vertexRDD.collect()
```

```
res1: Array[String] = Array(1,Jacob,48, 2,Jessica,45, 3,Andrew,25, 4,Ryan,53, 5,Emily,22, 6,Lily,52)
```

```
%scala
val vertices: RDD[(VertexId, (String, String))] = vertexRDD.map{ line => val fields = line.split(",")
                                                                (fields(0).toLong, ( fields(1), fields(2)))}
vertices.collect()
```

```
vertices: org.apache.spark.rdd.RDD[(org.apache.spark.graphx.VertexId, (String, String))] = MapPartitionsRDD[26] at map at command-22
3217164383679:1
res2: Array[(org.apache.spark.graphx.VertexId, (String, String))] = Array((1,(Jacob,48)), (2,(Jessica,45)), (3,(Andrew,25)), (4,(Rya
n,53)), (5,(Emily,22)), (6,(Lily,52)))
```

```
%scala
val edges: RDD[Edge[String]] = edgeRDD.map{ line => val fields = line.split(",")
                                                                Edge(fields(0).toLong, fields(1).toLong, fields(2))}
edges.collect()
```

```
edges: org.apache.spark.rdd.RDD[org.apache.spark.graphx.Edge[String]] = MapPartitionsRDD[27] at map at command-223217164383680:1
res3: Array[org.apache.spark.graphx.Edge[String]] = Array(Edge(6,1,Sister), Edge(1,2,Husband), Edge(2,1,Wife), Edge(5,1,Daughter), E
dge(5,2,Daughter), Edge(3,1,Son), Edge(3,2,Son), Edge(4,1,Friend), Edge(1,5,Father), Edge(1,3,Father), Edge(2,5,Mother), Edge(2,3,Mo
ther))
```

```
%scala
val default = ("unknown", "missing")
val graph = Graph(vertices, edges, default)
```

```
default: (String, String) = (unknown,missing)
graph: org.apache.spark.graphx.Graph[(String, String),String] = org.apache.spark.graphx.impl.GraphImpl@27608e58
```

```
%scala
// joining graphs
case class MoviesWatched(Movie: String, Genre: String)
val movies: RDD[(VertexId, MoviesWatched)] = sc.parallelize(List(
                                                            (1, MoviesWatched("Toy Story 3", "Kids")), (2, MoviesWatched
                                                            ("Titanic", "Love")),
                                                            (3, MoviesWatched("The Hangover ", "Comedy"))
))
```

```
defined class MoviesWatched
movies: org.apache.spark.rdd.RDD[(org.apache.spark.graphx.VertexId, MoviesWatched)] = ParallelCollectionRDD[40] at parallelize at co
mmand-223217164383682:3
```

```
%scala
val movieOuterJoinedGraph = graph.outerJoinVertices(movies)((_,name, movies) => (name, movies))
```

```
movieOuterJoinedGraph: org.apache.spark.graphx.Graph[((String, String), Option[MoviesWatched]),String] = org.apache.spark.graphx.imp
l.GraphImpl@42a47cbc
```

```
%scala
movieOuterJoinedGraph.vertices.map(t =>t).collect.foreach(println)
```

```
(4,((Ryan,53),None))
(6,((Lily,52),None))
(2,((Jessica,45),Some(MoviesWatched(Titanic,Love))))
(1,((Jacob,48),Some(MoviesWatched(Toy Story 3,Kids))))
(3,((Andrew,25),Some(MoviesWatched(The Hangover ,Comedy))))
(5,((Emily,22),None))
```

```
%scala
val movieOuterJoinedGraph = graph.outerJoinVertices(movies)((_,name, movies) => (name, movies.getOrElse(MoviesWatched("NA",
"NA"))))
```

```
movieOuterJoinedGraph: org.apache.spark.graphx.Graph[((String, String), MoviesWatched),String] = org.apache.spark.graphx.impl.GraphI
mpl@7980553c
```

```
%scala
movieOuterJoinedGraph.vertices.map(t=>t).collect.foreach(println)
```

```
(4,((Ryan,53),MoviesWatched(NA,NA)))
(6,((Lily,52),MoviesWatched(NA,NA)))
(2,((Jessica,45),MoviesWatched(Titanic,Love)))
(1,((Jacob,48),MoviesWatched(Toy Story 3,Kids)))
(3,((Andrew,25),MoviesWatched(The Hangover ,Comedy)))
(5,((Emily,22),MoviesWatched(NA,NA)))
```

```
%scala
val tCount = graph.triangleCount().vertices
```

```
tCount: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[102] at RDD at VertexRDD.scala:57
```

```
%scala
println(tCount.collect().mkString("\n"))
```

```
(4,0)
(6,0)
(2,2)
(1,2)
(3,1)
(5,1)
```

```
%scala
val iterations = 1000
```

```
iterations: Int = 1000
```

```
%scala
val connected = graph.connectedComponents().vertices
```

```
connected: org.apache.spark.graphx.VertexRDD[org.apache.spark.graphx.VertexId] = VertexRDDImpl[126] at RDD at VertexRDD.scala:57
```

```
%scala
val connections = graph.stronglyConnectedComponents(iterations).vertices
```

```
connections: org.apache.spark.graphx.VertexRDD[org.apache.spark.graphx.VertexId] = VertexRDDImpl[377] at RDD at VertexRDD.scala:57
```

```
%scala
val connByPerson = vertices.join(connected).map{ case(id, ((person,age), conn)) => (conn, id,person)}
val connByPersonS = vertices.join(connected).map{ case(id, ((person,age), conn)) => (conn, id,person)}
connByPerson.collect().foreach{ case (conn, id, person) => println(f"Weak $conn $id $person")}
```

```
Weak 1 4 Ryan
Weak 1 6 Lily
Weak 1 2 Jessica
Weak 1 1 Jacob
Weak 1 3 Andrew
Weak 1 5 Emily
connByPerson: org.apache.spark.rdd.RDD[(org.apache.spark.graphx.VertexId, org.apache.spark.graphx.VertexId, String)] = MapPartitions
RDD[427] at map at command-223217164383692:1
connByPersonS: org.apache.spark.rdd.RDD[(org.apache.spark.graphx.VertexId, org.apache.spark.graphx.VertexId, String)] = MapPartition
sRDD[431] at map at command-223217164383692:2
```

```
%scala
println("Vertices count:" +graph.vertices.count)
```

```
Vertices count:6
```

```
%scala
println("Vertices count:" +graph.edges.count)
```

```
Vertices count:12
```

```
%scala
val cnt1 = graph.vertices.filter{ case (id,(name, age)) => age.toLong > 40}.count
```

```
cnt1: Long = 4
```

```
%scala
val cnt2 = graph.edges.filter{ case Edge(from ,to, property) => property == "Father" | property == "Mother"}.count
```

```
cnt2: Long = 4
```

```
%scala
def max (a: (VertexId, Int), b: (VertexId, Int)) : (VertexId, Int)={
  if(a._2 > b._2) a else b
}
```

```
max: (a: (org.apache.spark.graphx.VertexId, Int), b: (org.apache.spark.graphx.VertexId, Int))(org.apache.spark.graphx.VertexId, Int)
```

```
%scala
val maxInDegree : (VertexId, Int) = graph.inDegrees.reduce(max)
val maxOutDegree : (VertexId, Int) = graph.outDegrees.reduce(max)
val maxDegree : (VertexId, Int) = graph.degrees.reduce(max)
```

```
maxInDegree: (org.apache.spark.graphx.VertexId, Int) = (1,5)
maxOutDegree: (org.apache.spark.graphx.VertexId, Int) = (2,3)
maxDegree: (org.apache.spark.graphx.VertexId, Int) = (1,8)
```

```
%scala
val minDegrees = graph.outDegrees.filter(_._2 <=1)
minDegrees.collect()
```

```
minDegrees: org.apache.spark.graphx.VertexRDD[Int] = VertexRDDImpl[451] at RDD at VertexRDD.scala:57
res10: Array[(org.apache.spark.graphx.VertexId, Int)] = Array((4,1), (6,1))
```

```
%scala
graph.triplets.map(
  triplet => triplet.srcAttr._1 + "is the " + triplet.attr +  "of " + triplet.dstAttr._1
).collect.foreach(println)
```

```
Jacobis the Husbandof Jessica
Jessicais the Wifeof Jacob
Andrewis the Sonof Jacob
Emilyis the Daughterof Jacob
Emilyis the Daughterof Jessica
Lilyis the Sisterof Jacob
Jacobis the Fatherof Andrew
Jacobis the Fatherof Emily
Jessicais the Motherof Andrew
Jessicais the Motherof Emily
Andrewis the Sonof Jessica
Ryanis the Friendof Jacob
```

```
%scala
print(sc.version)
```

```
3.3.2
```

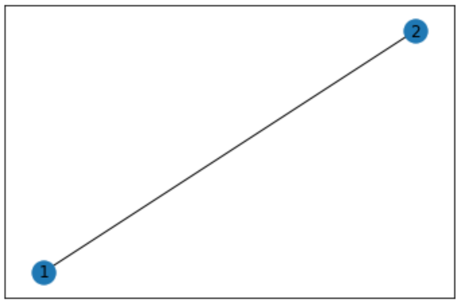
```
%scala
// import org.graphframes._
//  val vertex = spark.createDataFrame(List(
//    ("1", "Jacob", "48"),
//    ("2", "Jessica", "45"),
//    ("3", "Andrew", "25"),
//    ("4", "Jacob", "53"),
//    ("5", "Jaco", "22"),
//    ("6", "Jab", "52"),
//  ).toDF("id", "name", "age"))
```

```
pip install networkx
```

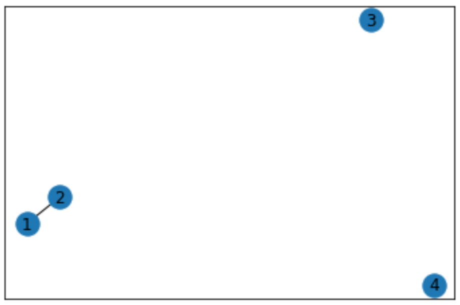
```
Python interpreter will be restarted.
Collecting networkx
  Downloading networkx-3.2-py3-none-any.whl (1.6 MB)
Installing collected packages: networkx
Successfully installed networkx-3.2
Python interpreter will be restarted.
```

```
import networkx as nx
```

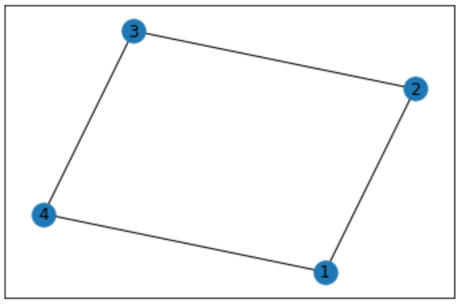
```
# create an Empty undirected graph
G  = nx.Graph()

import matplotlib.pyplot as plt
# adding
G.add_edge(1,2)
nx.draw_networkx(G)
plt.show()
```



```
G.add_nodes_from([3,4])
nx.draw_networkx(G)
plt.show()
```



```
G.add_edge(3,4)
G.add_edges_from([(2,3), (4,1)])
nx.draw_networkx(G)
plt.show()
```



```
G.nodes
```

```
Out[5]: NodeView((1, 2, 3, 4))
```

```
G.edges
```

```
Out[6]: EdgeView([(1, 2), (1, 4), (2, 3), (3, 4)])
```

```
list(nx.generate_adjlist(G))
```

```
Out[7]: ['1 2 4', '2 3', '3 4', '4']
```

```
nx.to_dict_of_lists(G)
```

```
Out[8]: {1: [2, 4], 2: [1, 3], 3: [4, 2], 4: [3, 1]}
```

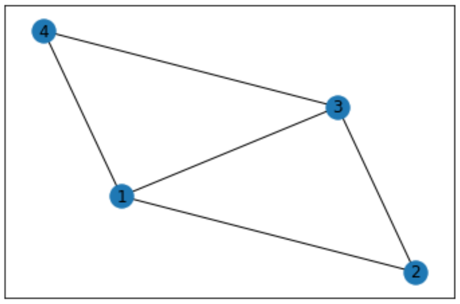
```
nx.to_edgelist(G)
```

```
Out[10]: EdgeDataView([(1, 2, {}), (1, 4, {}), (2, 3, {}), (3, 4, {})])
```

```
nx.to_pandas_adjacency(G)
```

	1	2	3	4
1	0.0	1.0	0.0	1.0
2	1.0	0.0	1.0	0.0
3	0.0	1.0	0.0	1.0
4	1.0	0.0	1.0	0.0

```
G.add_edge(1,3)
nx.draw_networkx(G)
plt.show()
```



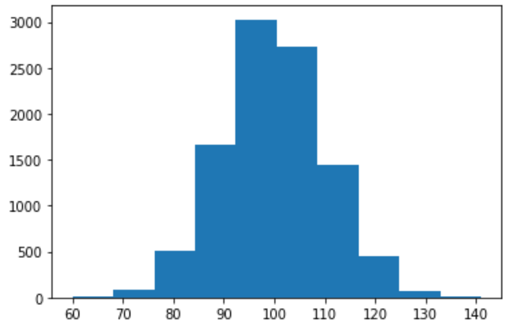
```
# print(nx.to_scipy_sparse_matrix(G))
```

```
G.degree
```

```
Out[13]: DegreeView({1: 3, 2: 2, 3: 3, 4: 2})
```

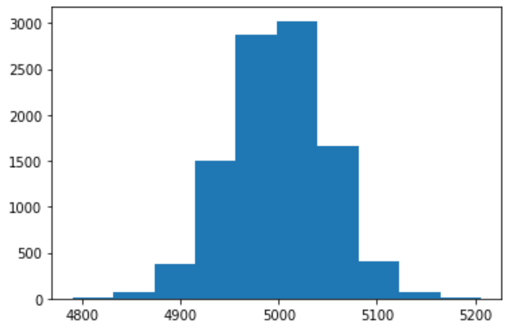
```
k = nx.fast_gnp_random_graph(10000, 0.01).degree()
plt.hist(list(dict(k).values()))
```

```
Out[14]: (array([   5.,   82.,  502., 1671., 3028., 2730., 1450.,  455.,   71.,
           6.]),
 array([ 60. ,  68.1,  76.2,  84.3,  92.4, 100.5, 108.6, 116.7, 124.8,
        132.9, 141. ]),
 <BarContainer object of 10 artists>)
```

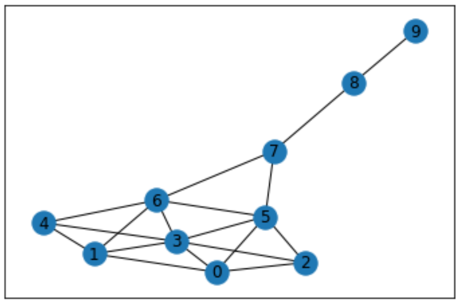


```
k = nx.fast_gnp_random_graph(10000, 0.50).degree()
plt.hist(list(dict(k).values()))
```

```
Out[15]: (array([   6.,   64.,  379., 1507., 2879., 3025., 1660.,  407.,   65.,
            8.]),
 array([4790. , 4831.6, 4873.2, 4914.8, 4956.4, 4998. , 5039.6, 5081.2,
        5122.8, 5164.4, 5206. ]),
 <BarContainer object of 10 artists>)
```



```
G = nx.krackhardt_kite_graph()
nx.draw_networkx(G)
plt.show()
```



```
print(nx.has_path(G, source =1 , target=9))
print(nx.shortest_path(G, source =1 , target=9))
print(nx.shortest_path_length(G, source =1 , target=9))
print(list(nx.shortest_simple_paths(G, source =1 , target=9)))
paths = list(nx.all_pairs_shortest_path(G))
paths[5][1]
```

```
True
[1, 6, 7, 8, 9]
4
[[1, 6, 7, 8, 9], [1, 0, 5, 7, 8, 9], [1, 6, 5, 7, 8, 9], [1, 3, 5, 7, 8, 9], [1, 4, 6, 7, 8, 9], [1, 3, 6, 7, 8, 9], [1, 0, 2,
5, 7, 8, 9], [1, 0, 5, 6, 7, 8, 9], [1, 6, 3, 5, 7, 8, 9], [1, 3, 5, 6, 7, 8, 9], [1, 4, 3, 5, 7, 8, 9], [1, 4, 6, 5, 7, 8, 9],
[1, 3, 0, 5, 7, 8, 9], [1, 3, 6, 5, 7, 8, 9], [1, 0, 3, 5, 7, 8, 9], [1, 4, 3, 6, 7, 8, 9], [1, 3, 2, 5, 7, 8, 9], [1, 0, 3, 6,
7, 8, 9], [1, 3, 4, 6, 7, 8, 9], [1, 0, 2, 3, 5, 7, 8, 9], [1, 0, 2, 5, 6, 7, 8, 9], [1, 0, 5, 3, 6, 7, 8, 9], [1, 6, 4, 3, 5, 7,
8, 9], [1, 6, 3, 0, 5, 7, 8, 9], [1, 4, 3, 5, 6, 7, 8, 9], [1, 4, 6, 3, 5, 7, 8, 9], [1, 3, 0, 2, 5, 7, 8, 9], [1, 3, 0, 5, 6, 7,
8, 9], [1, 0, 3, 5, 6, 7, 8, 9], [1, 4, 3, 0, 5, 7, 8, 9], [1, 4, 3, 6, 5, 7, 8, 9], [1, 3, 2, 0, 5, 7, 8, 9], [1, 3, 2, 5, 6, 7,
8, 9], [1, 0, 3, 2, 5, 7, 8, 9], [1, 0, 3, 6, 5, 7, 8, 9], [1, 3, 4, 6, 5, 7, 8, 9], [1, 0, 2, 3, 6, 7, 8, 9], [1, 6, 3, 2, 5, 7,
8, 9], [1, 4, 3, 2, 5, 7, 8, 9], [1, 0, 3, 4, 6, 7, 8, 9], [1, 0, 2, 3, 5, 6, 7, 8, 9], [1, 0, 2, 5, 3, 6, 7, 8, 9], [1, 0, 5, 2,
3, 6, 7, 8, 9], [1, 0, 5, 3, 4, 6, 7, 8, 9], [1, 6, 4, 3, 0, 5, 7, 8, 9], [1, 6, 3, 0, 2, 5, 7, 8, 9], [1, 4, 6, 3, 0, 5, 7, 8,
9], [1, 3, 0, 2, 5, 6, 7, 8, 9], [1, 4, 3, 0, 2, 5, 7, 8, 9], [1, 4, 3, 0, 5, 6, 7, 8, 9], [1, 3, 2, 0, 5, 6, 7, 8, 9], [1, 0, 3,
2, 5, 6, 7, 8, 9], [1, 0, 2, 3, 4, 6, 7, 8, 9], [1, 0, 2, 3, 6, 5, 7, 8, 9], [1, 6, 3, 2, 0, 5, 7, 8, 9], [1, 4, 3, 2, 0, 5, 7,
8, 9], [1, 4, 3, 2, 5, 6, 7, 8, 9], [1, 0, 3, 4, 6, 5, 7, 8, 9], [1, 6, 4, 3, 2, 5, 7, 8, 9], [1, 4, 6, 3, 2, 5, 7, 8, 9], [1, 0,
2, 5, 3, 4, 6, 7, 8, 9], [1, 0, 5, 2, 3, 4, 6, 7, 8, 9], [1, 6, 4, 3, 0, 2, 5, 7, 8, 9], [1, 4, 6, 3, 0, 2, 5, 7, 8, 9], [1, 4,
3, 0, 2, 5, 6, 7, 8, 9], [1, 0, 2, 3, 4, 6, 5, 7, 8, 9], [1, 4, 3, 2, 0, 5, 6, 7, 8, 9], [1, 6, 4, 3, 2, 0, 5, 7, 8, 9], [1, 4,
6, 3, 2, 0, 5, 7, 8, 9]]
Out[18]: {5: [5],
 0: [5, 0],
 2: [5, 2],
# importance of nodes inside the network
nx.betweenness_centrality(G)
```

```
Out[19]: {0: 0.023148148148148143,
 1: 0.023148148148148143,
 2: 0.0,
 3: 0.10185185185185183,
 4: 0.0,
 5: 0.23148148148148148,
 6: 0.23148148148148148,
 7: 0.38888888888888884,
 8: 0.2222222222222222,
 9: 0.0}
```

```
nx.degree_centrality(G)
```

```
Out[20]: {0: 0.4444444444444444,
 1: 0.4444444444444444,
 2: 0.3333333333333333,
 3: 0.6666666666666666,
 4: 0.3333333333333333,
 5: 0.5555555555555556,
 6: 0.5555555555555556,
 7: 0.3333333333333333,
 8: 0.2222222222222222,
 9: 0.1111111111111111}
```

```
nx.closeness_centrality(G)
```

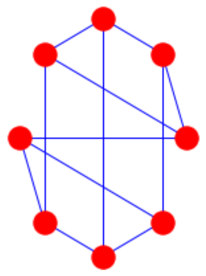
```
Out[21]: {0: 0.5294117647058824,
 1: 0.5294117647058824,
 2: 0.5,
 3: 0.6,
 4: 0.5,
 5: 0.6428571428571429,
 6: 0.6428571428571429,
 7: 0.6,
 8: 0.42857142857142855,
 9: 0.3103448275862069}
```

```
nx.harmonic_centrality(G)
```

```
Out[22]: {0: 6.083333333333333,
 1: 6.083333333333333,
 2: 5.583333333333333,
 3: 7.083333333333333,
 4: 5.583333333333333,
 5: 6.833333333333333,
 6: 6.833333333333333,
 7: 6.0,
 8: 4.666666666666666,
 9: 3.4166666666666665}
```

```
nx.eigenvector_centrality(G)
```

```
Out[23]: {0: 0.3522089813920359,
 1: 0.3522089813920358,
 2: 0.28583473531632403,
 3: 0.48102048812210046,
 4: 0.28583473531632403,
 5: 0.3976910106255469,
 6: 0.39769101062554685,
 7: 0.19586185175360382,
 8: 0.048074775014202924,
 9: 0.011164058575824235}
```



databricks Page Rank 2023-10-27 15:05:09

(https://databricks.com)

```
%python
pip install networkx
```

```
Python interpreter will be restarted.
Collecting networkx
  Downloading networkx-3.2-py3-none-any.whl (1.6 MB)
Installing collected packages: networkx
Successfully installed networkx-3.2
Python interpreter will be restarted.
```

```
from pylab import rcParams
rcParams['figure.figsize']= (3,3)
```

```
def nice_print(v , digits = 3):
    format = '%%.%df' %digits
    print(', '.join([format % e for e in v]))
```

```
nice_print([.12333122, .13432221, .64442143])
nice_print([.12333122, .13432221, .64442143], digits = 4)
```

```
0.123, 0.134, 0.644
0.1233, 0.1343, 0.6444
```

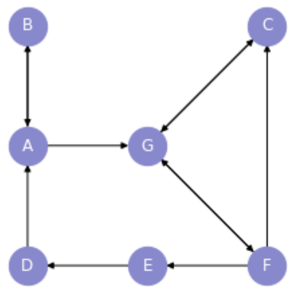
```
labels = ['A',
          'B',
          'C',
          'D',
          'E',
          'F',
          'G']
pages = range(len(labels))

positions = [(0,1),
             (0,2),
             (2,2),
             (0,0),
             (1,0),
             (2,0),
             (1,1)]

# this dictionary accosciates the numbers in pages to labels
page_labels = {p: l for p, l in zip(pages, labels)}
page_labels
```

```
Out[10]: {0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G'}
```

```
links = [(1,0),
         (3,0),
         (0,1),
         (5,2),
         (6,2),
         (6,5),
         (5,6),
         (2,6),
         (0,6),
         (5,4),
         (4,3)]
```

```
import networkx as nx
import matplotlib.pyplot as plt

g = nx.DiGraph()

for p in pages:
    node = g.add_node(p)

for (a,b) in links:
    g.add_edge(pages[a], pages[b])
```

```
plt.clf()
display(nx.draw(g, with_labels = True, labels = page_labels, node_size = 800, node_color = '#8888CC', font_color = 'white', pos
= positions))
```



```
adjacency ={}
for u in range(len(pages)):
    adjacency[u] = []

for (a,b) in links:
    adjacency[a].append(b)

print(adjacency)
```

```
{0: [1, 6], 1: [0], 2: [6], 3: [0], 4: [3], 5: [2, 6, 4], 6: [2, 5]}
```

```
connection_matrix =[]
for a in adjacency:
    for b in adjacency[a]:
        connection_matrix.append((b,a,1./len(adjacency[a])))
connection_matrix
```

```
Out[17]: [(1, 0, 0.5),
 (6, 0, 0.5),
 (0, 1, 1.0),
 (6, 2, 1.0),
 (0, 3, 1.0),
 (3, 4, 1.0),
 (2, 5, 0.3333333333333333),
 (6, 5, 0.3333333333333333),
 (4, 5, 0.3333333333333333),
 (2, 6, 0.5),
 (5, 6, 0.5)]
```

```
links_RDD = sc.parallelize(connection_matrix).cache()
```

```
links_RDD.take(3)
```

```
Out[19]: [(1, 0, 0.5), (6, 0, 0.5), (0, 1, 1.0)]
```

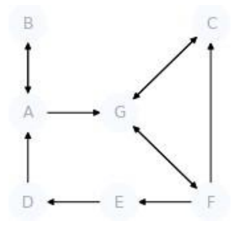
```
import numpy as np
n = len(pages)
page_rank = np.ones(n)/n
old_page_rank = np.ones(n)
print("Page rank is", page_rank)
print("Old Page rank is", old_page_rank)
```

```
Page rank is [0.14285714 0.14285714 0.14285714 0.14285714 0.14285714 0.14285714
 0.14285714]
Old Page rank is [1. 1. 1. 1. 1. 1. 1.]
```

```
def l2distance(v,q):
    if len(v) != len(q):
        raise ValueError('Cannot compute the distance of two vectors of different size')
    return sum([(q_el-v_el)**2 for v_el, q_el in zip(v,q)])
```

```
tolerance = 10e-7
max_iterations = 1000
iteration = 0
print("iiiiiiii")
while(l2distance(old_page_rank, page_rank) >= tolerance and iteration < max_iterations):
    old_page_rank = page_rank
    page_rank_values  = links_RDD.map(lambda x:(x[0], x[2]*page_rank[x[1]])).reduceByKey(lambda a, b: a+b).sortByKey().collect()
    page_rank = np.array([c for (i,c) in page_rank_values])
    nice_print(page_rank)
    print("Page rank", page_rank)
    iteration += 1
```

```
iiiiiiii
```



databricks PySpark

(https://databricks.com)

```
'''
PySpark Join is used to combine two DataFrames and by chaining these you can join multiple DataFrames; it supports all basic join type
OUTER, RIGHT OUTER, LEFT ANTI, LEFT SEMI, CROSS, SELF JOIN. PySpark Joins are wider transformations that involve data shuffling across

PySpark SQL Joins comes with more optimization by default (thanks to DataFrames) however still there would be some performance issues

In this PySpark SQL Join section, you will learn different Join syntaxes and using different Join types on two or more DataFrames and

PySpark Join Syntax
PySpark Join Types
Inner Join DataFrame
Full Outer Join DataFrame
Left Outer Join DataFrame
Right Outer Join DataFrame
Left Anti Join DataFrame
Left Semi Join DataFrame
Self Join DataFrame
Using SQL Expression

1. PySpark Join Syntax
PySpark SQL join has a below syntax and it can be accessed directly from DataFrame.
join() operation takes parameters as below and returns DataFrame.

join(self, other, on=None, how=None)

param other: Right side of the join
param on: a string for the join column name
param how: default inner. Must be one of inner, cross, outer,full, full_outer, left, left_outer, right, right_outer,left_semi, and lef
You can also write Join expression by adding where() and filter() methods on DataFrame and can have Join on multiple columns.

2. PySpark Join Types
Below are the different Join Types PySpark supports.

Join String                          Equivalent SQL Join
inner                              INNER JOIN
outer, full, fullouter, full_outer   FULL OUTER JOIN
left, leftouter, left_outer          LEFT JOIN
right, rightouter, right_outer       RIGHT JOIN
cross
anti, leftanti, left_anti
semi, leftsemi, left_semi

Before we jump into PySpark SQL Join examples, first, let’s create an "emp" and "dept" DataFrames. here, column "emp_id" is unique on
emp_dept_id from emp has a reference to dept_id on dept dataset.
'''
```

```
emp = [(1,"Smith",-1,"2018","10","M",3000), \
    (2,"Rose",1,"2010","20","M",4000), \
    (3,"Williams",1,"2010","10","M",1000), \
    (4,"Jones",2,"2005","10","F",2000), \
    (5,"Brown",2,"2010","40","",-1), \
      (6,"Brown",2,"2010","50","",-1) \
  ]
empColumns = ["emp_id","name","superior_emp_id","year_joined", \
       "emp_dept_id","gender","salary"]

empDF = spark.createDataFrame(data=emp, schema = empColumns)
empDF.printSchema()
empDF.show(truncate=False)

dept = [("Finance",10), \
    ("Marketing",20), \
    ("Sales",30), \
    ("IT",40) \
  ]
deptColumns = ["dept_name","dept_id"]
deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
deptDF.printSchema()
deptDF.show(truncate=False)

'''This prints “emp” and “dept” DataFrame to the console. Refer complete example below on how to create spark object.'''
```

```
root
 |-- emp_id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- superior_emp_id: long (nullable = true)
 |-- year_joined: string (nullable = true)
 |-- emp_dept_id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)

+------+--------+---------------+-----------+-----------+------+------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |
|2     |Rose    |1              |2010       |20         |M     |4000  |
|3     |Williams|1              |2010       |10         |M     |1000  |
|4     |Jones   |2              |2005       |10         |F     |2000  |
|5     |Brown   |2              |2010       |40         |      |-1    |
|6     |Brown   |2              |2010       |50         |      |-1    |
+------+--------+---------------+-----------+-----------+------+------+

root
```

```
'''
3. PySpark Inner Join DataFrame
Inner join is the default join in PySpark and it’s mostly used. This joins two datasets on key columns, where keys don’t match
the rows get dropped from both datasets (emp & dept).

When we apply Inner join on our datasets, It drops “emp_dept_id” 50 from “emp” and “dept_id” 30 from “dept” datasets. Below is
the result of the above Join expression.
'''
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"inner") \
     .show(truncate=False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance  |10     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
'''
4. PySpark Full Outer Join
Outer a.k.a full, fullouter join returns all rows from both datasets, where join expression doesn’t match it returns null on
respective record columns.

From our “emp” dataset’s “emp_dept_id” with value 50 doesn’t have a record on “dept” hence dept columns have null and “dept_id”
30 doesn’t have a record in “emp” hence you see null’s on emp columns. Below is the result of the below Join expression.
'''
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"outer") \
    .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"full") \
    .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"fullouter") \
    .show(truncate=False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance  |10     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|null  |null    |null           |null       |null       |null  |null  |Sales    |30     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
|6     |Brown   |2              |2010       |50         |      |-1    |null     |null   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance  |10     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|null  |null    |null           |null       |null       |null  |null  |Sales    |30     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
```

```
  '''
  5. PySpark Left Outer Join
Left a.k.a Leftouter join returns all rows from the left dataset regardless of match found on the right dataset when join
expression doesn’t match, it assigns null for that record and drops records from right where match not found.

From our dataset, “emp_dept_id” 50 doesn’t have a record on “dept” dataset hence, this record contains null on “dept” columns
(dept_name & dept_id). and “dept_id” 30 from “dept” dataset dropped from the results. Below is the result of the above Join
expression.
  '''
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"left") \
    .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftouter") \
    .show(truncate=False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance  |10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
|6     |Brown   |2              |2010       |50         |      |-1    |null     |null   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance  |10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
|6     |Brown   |2              |2010       |50         |      |-1    |null     |null   |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

```
'''
6. Right Outer Join
Right a.k.a Rightouter join is opposite of left join, here it returns all rows from the right dataset regardless of math found
on the left dataset, when join expression doesn't match, it assigns null for that record and drops records from left where match
not found.

From our example, the right dataset "dept_id" 30 doesn't have it on the left dataset "emp" hence, this record contains null on
"emp" columns. and "emp_dept_id" 50 dropped as a match not found on left. Below is the result of the above Join expression.
'''
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"right") \
   .show(truncate=False)
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"rightouter") \
   .show(truncate=False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance  |10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|null  |null    |null           |null       |null       |null  |null  |Sales    |30     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance  |10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|null  |null    |null           |null       |null       |null  |null  |Sales    |30     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
```

```
'''
7. Left Semi Join
leftsemi join is similar to inner join difference being leftsemi join returns all columns from the left dataset and ignores all
columns from the right dataset. In other words, this join returns columns from the only left dataset for the records match in
the right dataset on join expression, records not matched on join expression are ignored from both left and right datasets.

The same result can be achieved using select on the result of the inner join however, using this join would be efficient.
Below is the result of the above join expression.
'''
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftsemi") \
   .show(truncate=False)
```

```
+------+--------+---------------+-----------+-----------+------+------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |
|3     |Williams|1              |2010       |10         |M     |1000  |
|4     |Jones   |2              |2005       |10         |F     |2000  |
|2     |Rose    |1              |2010       |20         |M     |4000  |
|5     |Brown   |2              |2010       |40         |      |-1    |
+------+--------+---------------+-----------+-----------+------+------+
```

```
'''
8. Left Anti Join
leftanti join does the exact opposite of the leftsemi, leftanti join returns only columns from the left dataset for non-matched records.

Yields below output
'''
empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftanti") \
   .show(truncate=False)
```

```
+------+-----+---------------+-----------+-----------+------+------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+-----+---------------+-----------+-----------+------+------+
|6     |Brown|2              |2010       |50         |      |-1    |
+------+-----+---------------+-----------+-----------+------+------+
```

```
'''
Joins are not complete without a self join, Though there is no self-join type available, we can use any of the above-explained join types to join DataFrame to itself. below example use inner self join.

Here, we are joining emp dataset with itself to find out superior emp_id and name for all employees.
'''
from pyspark.sql.functions import col
empDF.alias("emp1").join(empDF.alias("emp2"), \
    col("emp1.superior_emp_id") == col("emp2.emp_id"),"inner") \
    .select(col("emp1.emp_id"),col("emp1.name"), \
      col("emp2.emp_id").alias("superior_emp_id"), \
      col("emp2.name").alias("superior_emp_name")) \
   .show(truncate=False)
```

```
+------+--------+---------------+-----------------+
|emp_id|name    |superior_emp_id|superior_emp_name|
+------+--------+---------------+-----------------+
|2     |Rose    |1              |Smith            |
|3     |Williams|1              |Smith            |
|4     |Jones   |2              |Rose             |
|5     |Brown   |2              |Rose             |
|6     |Brown   |2              |Rose             |
+------+--------+---------------+-----------------+
'''
4. Using SQL Expression
Since PySpark SQL support native SQL syntax, we can also write join operations after creating temporary tables on DataFrames and
use these tables on spark.sql().


'''
empDF.createOrReplaceTempView("EMP")
deptDF.createOrReplaceTempView("DEPT")

joinDF = spark.sql("select * from EMP e, DEPT d where e.emp_dept_id == d.dept_id") \
  .show(truncate=False)

joinDF2 = spark.sql("select * from EMP e INNER JOIN DEPT d ON e.emp_dept_id == d.dept_id") \
  .show(truncate=False)
```

```
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance  |10     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+

+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |Finance  |10     |
|3     |Williams|1              |2010       |10         |M     |1000  |Finance  |10     |
|4     |Jones   |2              |2005       |10         |F     |2000  |Finance  |10     |
|2     |Rose    |1              |2010       |20         |M     |4000  |Marketing|20     |
|5     |Brown   |2              |2010       |40         |      |-1    |IT       |40     |
+------+--------+---------------+-----------+-----------+------+------+---------+-------+
  '''
  6. PySpark SQL Join Complete Example
  '''
  import pyspark
  from pyspark.sql import SparkSession
  from pyspark.sql.functions import col

  spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

  emp = [(1,"Smith",-1,"2018","10","M",3000), \
      (2,"Rose",1,"2010","20","M",4000), \
      (3,"Williams",1,"2010","10","M",1000), \
      (4,"Jones",2,"2005","10","F",2000), \
      (5,"Brown",2,"2010","40","",-1), \
        (6,"Brown",2,"2010","50","",-1) \
    ]
  empColumns = ["emp_id","name","superior_emp_id","year_joined", \
         "emp_dept_id","gender","salary"]

  empDF = spark.createDataFrame(data=emp, schema = empColumns)
  empDF.printSchema()
  empDF.show(truncate=False)


  dept = [("Finance",10), \
      ("Marketing",20), \
      ("Sales",30), \
      ("IT",40) \
    ]
  deptColumns = ["dept_name","dept_id"]
  deptDF = spark.createDataFrame(data=dept, schema = deptColumns)
  deptDF.printSchema()
  deptDF.show(truncate=False)

  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"inner") \
       .show(truncate=False)

  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"outer") \
      .show(truncate=False)
  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"full") \
      .show(truncate=False)
  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"fullouter") \
      .show(truncate=False)

  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"left") \
      .show(truncate=False)
  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftouter") \
     .show(truncate=False)

  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"right") \
     .show(truncate=False)
  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"rightouter") \
     .show(truncate=False)

  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftsemi") \
     .show(truncate=False)

  empDF.join(deptDF,empDF.emp_dept_id ==  deptDF.dept_id,"leftanti") \
     .show(truncate=False)

  empDF.alias("emp1").join(empDF.alias("emp2"), \
      col("emp1.superior_emp_id") == col("emp2.emp_id"),"inner") \
      .select(col("emp1.emp_id"),col("emp1.name"), \
        col("emp2.emp_id").alias("superior_emp_id"), \
        col("emp2.name").alias("superior_emp_name")) \
     .show(truncate=False)
```

```
empDF.createOrReplaceTempView("EMP")
deptDF.createOrReplaceTempView("DEPT")

joinDF = spark.sql("select * from EMP e, DEPT d where e.emp_dept_id == d.dept_id") \
  .show(truncate=False)

joinDF2 = spark.sql("select * from EMP e INNER JOIN DEPT d ON e.emp_dept_id == d.dept_id") \
  .show(truncate=False)
```

```
root
 |-- emp_id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- superior_emp_id: long (nullable = true)
 |-- year_joined: string (nullable = true)
 |-- emp_dept_id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)

+------+--------+---------------+-----------+-----------+------+------+
|emp_id|name    |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+--------+---------------+-----------+-----------+------+------+
|1     |Smith   |-1             |2018       |10         |M     |3000  |
|2     |Rose    |1              |2010       |20         |M     |4000  |
|3     |Williams|1              |2010       |10         |M     |1000  |
|4     |Jones   |2              |2005       |10         |F     |2000  |
|5     |Brown   |2              |2010       |40         |      |-1    |
|6     |Brown   |2              |2010       |50         |      |-1    |
+------+--------+---------------+-----------+-----------+------+------+

root
```

```
'''
PySpark distinct() function is used to drop/remove the duplicate rows (all columns) from DataFrame and dropDuplicates() is used
to drop rows based on selected (one or multiple) columns. In this article, you will learn how to use distinct() and
dropDuplicates() functions with PySpark example.

Before we start, first let’s create a DataFrame with some duplicate rows and values on a few columns. We use this DataFrame to
demonstrate how to get distinct multiple columns.

On the above table, record with employer name James has duplicate rows, As you notice we have 2 rows that have duplicate values
on all columns and we have 4 rows that have duplicate values on department and salary columns.

'''
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import expr
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

data = [("James", "Sales", 3000), \
    ("Michael", "Sales", 4600), \
    ("Robert", "Sales", 4100), \
    ("Maria", "Finance", 3000), \
    ("James", "Sales", 3000), \
    ("Scott", "Finance", 3300), \
    ("Jen", "Finance", 3900), \
    ("Jeff", "Marketing", 3000), \
    ("Kumar", "Marketing", 2000), \
    ("Saif", "Sales", 4100) \
  ]
columns= ["employee_name", "department", "salary"]
df = spark.createDataFrame(data = data, schema = columns)
df.printSchema()
df.show(truncate=False)
'''
1. Get Distinct Rows (By Comparing All Columns)
On the above DataFrame, we have a total of 10 rows with 2 rows having all values duplicated, performing distinct on this
DataFrame should get us 9 after removing 1 duplicate row.
distinct() function on DataFrame returns a new DataFrame after removing the duplicate records. This example yields the below
output.
'''
#Distinct
distinctDF = df.distinct()
print("Distinct count: "+str(distinctDF.count()))
distinctDF.show(truncate=False)
'''
Alternatively, you can also run dropDuplicates() function which returns a new DataFrame after removing duplicate rows.
'''
#Drop duplicates
df2 = df.dropDuplicates()
print("Distinct count: "+str(df2.count()))
df2.show(truncate=False)

'''
2. PySpark Distinct of Selected Multiple Columns
PySpark doesn’t have a distinct method which takes columns that should run distinct on (drop duplicate rows on selected multiple
columns) however, it provides another signature of dropDuplicates() function which takes multiple columns to eliminate
duplicates.

Note that calling dropDuplicates() on DataFrame returns a new DataFrame with duplicate rows removed.

Yields below output. If you notice the output, It dropped 2 records that are duplicate.
'''
#Drop duplicates on selected columns
dropDisDF = df.dropDuplicates(["department","salary"])
print("Distinct count of department salary : "+str(dropDisDF.count()))
dropDisDF.show(truncate=False)

'''
```

```
In this PySpark SQL article, you have learned distinct() method which is used to get the distinct values of rows (all columns)
and also learned how to use dropDuplicates() to get the distinct and finally learned using dropDuplicates() function to get
distinct of multiple columns.
'''
```

```
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- salary: long (nullable = true)

+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James        |Sales     |3000  |
|Michael      |Sales     |4600  |
|Robert       |Sales     |4100  |
|Maria        |Finance   |3000  |
|James        |Sales     |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3900  |
|Jeff         |Marketing |3000  |
|Kumar        |Marketing |2000  |
|Saif         |Sales     |4100  |
+-------------+----------+------+

Distinct count: 9
```

```
'''
PySpark JSON Functions with Examples
PySpark JSON functions are used to query or extract the elements from JSON string of DataFrame column by path, convert it to
struct, mapt type e.t.c, In this article, I will explain the most used JSON SQL functions with Python examples.

1. PySpark JSON Functions
from_json() – Converts JSON string into Struct type or Map type.

to_json() – Converts MapType or Struct type to JSON string.

json_tuple() – Extract the Data from JSON and create them as a new columns.

get_json_object() – Extracts JSON element from a JSON string based on json path specified.

schema_of_json() – Create schema string from JSON string

1.1. Create DataFrame with Column contains JSON String
In order to explain these JSON functions first, let’s create DataFrame with a column contains JSON string.
'''

from pyspark.sql import SparkSession,Row
spark = SparkSession.builder.appName('SparkByExamples.com').getOrCreate()

jsonString="""{"Zipcode":704,"ZipCodeType":"STANDARD","City":"PARC PARQUE","State":"PR"}"""
df=spark.createDataFrame([(1, jsonString)],["id","value"])
df.show(truncate=False)

'''
2.1. from_json()
PySpark from_json() function is used to convert JSON string into Struct type or Map type. The below example converts JSON string
to Map key-value pair. I will leave it to you to convert to struct type. Refer, Convert JSON string to Struct type column.
'''

#Convert JSON string column to Map type
from pyspark.sql.types import MapType,StringType
from pyspark.sql.functions import from_json
df2=df.withColumn("value",from_json(df.value,MapType(StringType(),StringType())))
df2.printSchema()
df2.show(truncate=False)

'''
2.2. to_json()
to_json() function is used to convert DataFrame columns MapType or Struct type to JSON string. Here, I am using df2 that created
from above from_json() example.
'''

from pyspark.sql.functions import to_json,col
df2.withColumn("value",to_json(col("value"))) \
   .show(truncate=False)

'''
2.3. json_tuple()
Function json_tuple() is used the query or extract the elements from JSON column and create the result as a new columns.
'''

from pyspark.sql.functions import json_tuple
df.select(col("id"),json_tuple(col("value"),"Zipcode","ZipCodeType","City")) \
    .toDF("id","Zipcode","ZipCodeType","City") \
    .show(truncate=False)
'''
2.4. get_json_object()
get_json_object() is used to extract the JSON string based on path from the JSON column.
'''

from pyspark.sql.functions import get_json_object
df.select(col("id"),get_json_object(col("value"),"$.ZipCodeType").alias("ZipCodeType")) \
    .show(truncate=False)
```

```
'''
2.5. schema_of_json()
Use schema_of_json() to create schema string from JSON string column.
'''

from pyspark.sql.functions import schema_of_json,lit
schemaStr=spark.range(1) \
    .select(schema_of_json(lit("""{"Zipcode":704,"ZipCodeType":"STANDARD","City":"PARC PARQUE","State":"PR"}"""))) \
    .collect()[0][0]
print(schemaStr)
```

```
+---+--------------------------------------------------------------------------+
|id |value                                                                     |
+---+--------------------------------------------------------------------------+
|1  |{"Zipcode":704,"ZipCodeType":"STANDARD","City":"PARC PARQUE","State":"PR"}|
+---+--------------------------------------------------------------------------+

root
 |-- id: long (nullable = true)
 |-- value: map (nullable = true)
 |    |-- key: string
 |    |-- value: string (valueContainsNull = true)

+---+---------------------------------------------------------------------------+
|id |value                                                                      |
+---+---------------------------------------------------------------------------+
|1  |{Zipcode -> 704, ZipCodeType -> STANDARD, City -> PARC PARQUE, State -> PR}|
+---+---------------------------------------------------------------------------+

+---+----------------------------------------------------------------------------+
|id |value                                                                       |
+---+----------------------------------------------------------------------------+
```

```
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- salary: long (nullable = true)

+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James        |Sales     |3000  |
|Michael      |Sales     |4600  |
|Robert       |Sales     |4100  |
|Maria        |Finance   |3000  |
|James        |Sales     |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3900  |
|Jeff         |Marketing |3000  |
|Kumar        |Marketing |2000  |
|Saif         |Sales     |4100  |
+-------------+----------+------+

+-------------+----------+------+----------------+
```

databricks (https://databricks.com)

Pyspark join types and dataframes Notebook 2023-09-04 11:03:40

```
root
 |-- emp_id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- superior_emp_id: long (nullable = true)
 |-- year_joined: string (nullable = true)
 |-- emp_dept_id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: long (nullable = true)

+------+-----+---------------+-----------+-----------+------+------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+-----+---------------+-----------+-----------+------+------+
|1     |Smith|-1             |2018       |10         |M     |3000  |
|2     |Rose |1              |2010       |20         |M     |4000  |
|4     |Jones|2              |2005       |10         |F     |2000  |
|5     |Brown|2              |2016       |40         |      |-1    |
|6     |Brown|2              |2010       |50         |      |-1    |
+------+-----+---------------+-----------+-----------+------+------+
```

```
root
 |-- dept_name: string (nullable = true)
 |-- dept_id: long (nullable = true)

+---------+-------+
|dept_name|dept_id|
+---------+-------+
|Finance  |10     |
|Marketing|20     |
|Sales    |30     |
|IT       |40     |
+---------+-------+
```

```
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith|-1             |2018       |10         |M     |3000  |Finance  |10     |
|4     |Jones|2              |2005       |10         |F     |2000  |Finance  |10     |
|2     |Rose |1              |2010       |20         |M     |4000  |Marketing|20     |
|5     |Brown|2              |2016       |40         |      |-1    |IT       |40     |
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
+------+-----+--------------+-----------+-----------+------+------+---------+-----
--+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_
id|
+------+-----+--------------+-----------+-----------+------+------+---------+-----
--+
|1     |Smith|-1             |2018       |10         |M     |3000  |Finance  |10
|
|4     |Jones|2              |2005       |10         |F     |2000  |Finance  |10
|
|2     |Rose |1              |2010       |20         |M     |4000  |Marketing|20
|
|null  |null |null           |null       |null       |null  |null  |Sales    |30
|
|5     |Brown|2              |2016       |40         |      |-1    |IT       |40
|
|6     |Brown|2              |2010       |50         |      |-1    |null     |null
|
+------+-----+--------------+-----------+-----------+------+------+---------+-----
--+
```

```
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith|-1             |2018       |10         |M     |3000  |Finance  |10     |
|2     |Rose |1              |2010       |20         |M     |4000  |Marketing|20     |
|4     |Jones|2              |2005       |10         |F     |2000  |Finance  |10     |
|5     |Brown|2              |2016       |40         |      |-1    |IT       |40     |
|6     |Brown|2              |2010       |50         |      |-1    |null     |null   |
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
```

```
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith|-1             |2018       |10         |M     |3000  |Finance  |10     |
|2     |Rose |1              |2010       |20         |M     |4000  |Marketing|20     |
|4     |Jones|2              |2005       |10         |F     |2000  |Finance  |10     |
|5     |Brown|2              |2016       |40         |      |-1    |IT       |40     |
|6     |Brown|2              |2010       |50         |      |-1    |null     |null   |
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
```

```
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|4     |Jones|2              |2005       |10         |F     |2000  |Finance  |10     |
|1     |Smith|-1             |2018       |10         |M     |3000  |Finance  |10     |
|2     |Rose |1              |2010       |20         |M     |4000  |Marketing|20     |
|null  |null |null           |null       |null       |null  |null  |Sales    |30     |
|5     |Brown|2              |2016       |40         |      |-1    |IT       |40     |
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
```

```
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|4     |Jones|2              |2005       |10         |F     |2000  |Finance  |10     |
|1     |Smith|-1             |2018       |10         |M     |3000  |Finance  |10     |
|2     |Rose |1              |2010       |20         |M     |4000  |Marketing|20     |
|null  |null |null           |null       |null       |null  |null  |Sales    |30     |
|5     |Brown|2              |2016       |40         |      |-1    |IT       |40     |
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
```

```
+------+-----+---------------+-----------+-----------+------+------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+-----+---------------+-----------+-----------+------+------+
|1     |Smith|-1             |2018       |10         |M     |3000  |
|4     |Jones|2              |2005       |10         |F     |2000  |
|2     |Rose |1              |2010       |20         |M     |4000  |
|5     |Brown|2              |2016       |40         |      |-1    |
+------+-----+---------------+-----------+-----------+------+------+
```

```
+------+-----+---------------+-----------+-----------+------+------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|
+------+-----+---------------+-----------+-----------+------+------+
|6     |Brown|2              |2010       |50         |      |-1    |
+------+-----+---------------+-----------+-----------+------+------+
```

```
+------+-----+---------------+-----------------+
|emp_id|name |superior_emp_id|superior.emp_name|
+------+-----+---------------+-----------------+
|2     |Rose |1              |Smith            |
|4     |Jones|2              |Rose             |
|5     |Brown|2              |Rose             |
|6     |Brown|2              |Rose             |
+------+-----+---------------+-----------------+
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith|-1             |2018       |10         |M     |3000  |Finance  |10     |
|4     |Jones|2              |2005       |10         |F     |2000  |Finance  |10     |
|2     |Rose |1              |2010       |20         |M     |4000  |Marketing|20     |
|5     |Brown|2              |2016       |40         |      |-1    |IT       |40     |
+------+-----+---------------+-----------+-----------+------+------+---------+-------+

+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|emp_id|name |superior_emp_id|year_joined|emp_dept_id|gender|salary|dept_name|dept_id|
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
|1     |Smith|-1             |2018       |10         |M     |3000  |Finance  |10     |
|4     |Jones|2              |2005       |10         |F     |2000  |Finance  |10     |
|2     |Rose |1              |2010       |20         |M     |4000  |Marketing|20     |
|5     |Brown|2              |2016       |40         |      |-1    |IT       |40     |
+------+-----+---------------+-----------+-----------+------+------+---------+-------+
root
 |-- employee_name: string (nullable = true)
 |-- department: string (nullable = true)
 |-- salary: long (nullable = true)

+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James        |Sales     |3000  |
|Michael      |Sales     |4600  |
|Robert       |Sales     |4100  |
|Maria        |Finance   |3000  |
|James        |Sales     |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3988  |
|Jeff         |Marketing |3000  |
|Kumar        |Marketing |2000  |
|Saif         |Sales     |4100  |
+-------------+----------+------+
```

```
Disnict count : 9
+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James        |Sales     |3000  |
|Michael      |Sales     |4600  |
|Robert       |Sales     |4100  |
|Maria        |Finance   |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3988  |
|Jeff         |Marketing |3000  |
|Kumar        |Marketing |2000  |
|Saif         |Sales     |4100  |
+-------------+----------+------+
Disnict count : 9
+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|James        |Sales     |3000  |
|Michael      |Sales     |4600  |
|Robert       |Sales     |4100  |
|Maria        |Finance   |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3988  |
|Jeff         |Marketing |3000  |
|Kumar        |Marketing |2000  |
|Saif         |Sales     |4100  |
+-------------+----------+------+
```

```
Disnict count Selected items: 8
+-------------+----------+------+
|employee_name|department|salary|
+-------------+----------+------+
|Maria        |Finance   |3000  |
|Scott        |Finance   |3300  |
|Jen          |Finance   |3988  |
|Kumar        |Marketing |2000  |
|Jeff         |Marketing |3000  |
|James        |Sales     |3000  |
|Robert       |Sales     |4100  |
|Michael      |Sales     |4600  |
+-------------+----------+------+
+---+--------------------------------------------------------------------------------+
|id |value                                                                           |
+---+--------------------------------------------------------------------------------+
|1  |{"Zipcode" : 704,"ZipcodeType" : "STANDARD", "City":"PARC", "State": "PR"}|
+---+--------------------------------------------------------------------------------+
```

databricks Spark SQL G1 and G2 Notebook 2023-08-28 12:31:23

(https://databricks.com)

```
val sqlConstext = new org.apache.spark.sql.SQLContext(sc)
```

```
command-4344394284453045:1: warning: constructor SQLContext in class SQLContext is deprecated (since 2.0.0): Use SparkSession.builde
r instead
val sqlConstext = new org.apache.spark.sql.SQLContext(sc)
                  ^
sqlConstext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@508225e8
```

```
val a  = sc.parallelize(1 to 10)
```

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[17] at parallelize at command-4344394284453046:1
```

```
val b = a.map(x=> (x , x+1))
```

```
b: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[18] at map at command-4344394284453047:1
```

```
b.collect
```

```
res0: Array[(Int, Int)] = Array((1,2), (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (8,9), (9,10), (10,11))
```

```
val df = b.toDF("First", "Second")
```

```
df: org.apache.spark.sql.DataFrame = [First: int, Second: int]
```

```
df.show
```

```
+-----+------+
|First|Second|
+-----+------+
|    1|     2|
|    2|     3|
|    3|     4|
|    4|     5|
|    5|     6|
|    6|     7|
|    7|     8|
|    8|     9|
|    9|    10|
|   10|    11|
+-----+------+
```

```
val a  = List(("Tom", 5),("Jerry", 2),("Donald", 7))
```

```
a: List[(String, Int)] = List((Tom,5), (Jerry,2), (Donald,7))
```

```
val df = a.toDF("Name", "Age")
```

```
df: org.apache.spark.sql.DataFrame = [Name: string, Age: int]
```

```
df.show
```

```
+------+---+
|  Name|Age|
+------+---+
|   Tom|  5|
| Jerry|  2|
|Donald|  7|
+------+---+
```

```
val a  = Seq(("Tom", 5),("Jerry", 2),("Donald", 7))
```

```
a: Seq[(String, Int)] = List((Tom,5), (Jerry,2), (Donald,7))
```

```
val df = a.toDF("Name", "Age")
```

```
df: org.apache.spark.sql.DataFrame = [Name: string, Age: int]
```

```
df.show
```

```
+------+---+
|  Name|Age|
+------+---+
|   Tom|  5|
| Jerry|  2|
|Donald|  7|
+------+---+
```

```
df.registerTempTable("Cartoon")
```

```
command-4344394284453057:1: warning: method registerTempTable in class Dataset is deprecated (since 2.0.0): Use createOrReplaceTempV
iew(viewName) instead.
df.registerTempTable("Cartoon")
   ^
```

```
df.createOrReplaceTempView("Cartoon")
```

```
sqlContext.sql("select * from Cartoon where Name = 'Tom'").show
```

```
+----+---+
|Name|Age|
+----+---+
| Tom|  5|
+----+---+
```

```
sqlContext.sql("select * from Cartoon").show
```

```
+------+---+
|  Name|Age|
+------+---+
|   Tom|  5|
| Jerry|  2|
|Donald|  7|
+------+---+
sqlContext.sql("select count(*) from Cartoon").show
```

```
+--------+
|count(1)|
+--------+
|       3|
+--------+
```

```
// questions : to create a json file, upoad it open dbfs and perform the following operations on it.

// printSchema()
// select the query with all the names
// filter and identify age > 23
// groupBy Age Count it and show it

// how ro read file

// var df1 = spark.read.format("json").load("dbfs:/FileStore/shared_uploads/........./.………………json")
```

```
var df1 = spark.read.format("json").load("/FileStore/tables/file.json")
```

```
df1: org.apache.spark.sql.DataFrame = [_corrupt_record: string]
```

```
display(df1)
```

```
   AnalysisException: Since Spark 2.3, the queries from raw JSON/CSV files are disallowed when the
referenced columns only include the internal corrupt record column
(named _corrupt_record by default). For example:
spark.read.schema(schema).csv(file).filter($"_corrupt_record".isNotNull).count()
and spark.read.schema(schema).csv(file).select("_corrupt_record").show().
Instead, you can cache or save the parsed results and then send the same query.
For example, val df = spark.read.schema(schema).csv(file).cache() and then
df.filter($"_corrupt_record".isNotNull).count().
```

```
val df1 = spark.read.format("json").load("dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/emp_1.json")
```

```
df1: org.apache.spark.sql.DataFrame = [age: string, id: string ... 1 more field]
```

```
df1.show
```

```
+---+----+---------+
|age|  id|     name|
+---+----+---------+
| 25|1201|       om|
| 25|1202|     some|
| 25|1203|    thing|
| 25|1204|different|
| 25|1205|    going|
| 25|1206|       on|
| 25|1207|    kavan|
+---+----+---------+
```

```
   AnalysisException: [TABLE_OR_VIEW_NOT_FOUND] The table or view `df1` cannot be found. Verify the spelling and correctness of the s
chema and catalog.
If you did not qualify the name with a schema, verify the current_schema() output, or qualify the name with the correct schema and ca
```

```
talog.
To tolerate the error on drop use DROP VIEW IF EXISTS or DROP TABLE IF EXISTS.; line 1 pos 17;
'Project ['name]
+- 'UnresolvedRelation [df1], [], false
```

Hadoop Installation

Prepared by: VASAVA VIPULKUMAR

DINESHBHAI (23MDS003)

Branch: MTech (Data Science)

❖ Prerequisites

1. Java 8 runtime environment (JRE): Hadoop 3 requires a Java 8 installation.

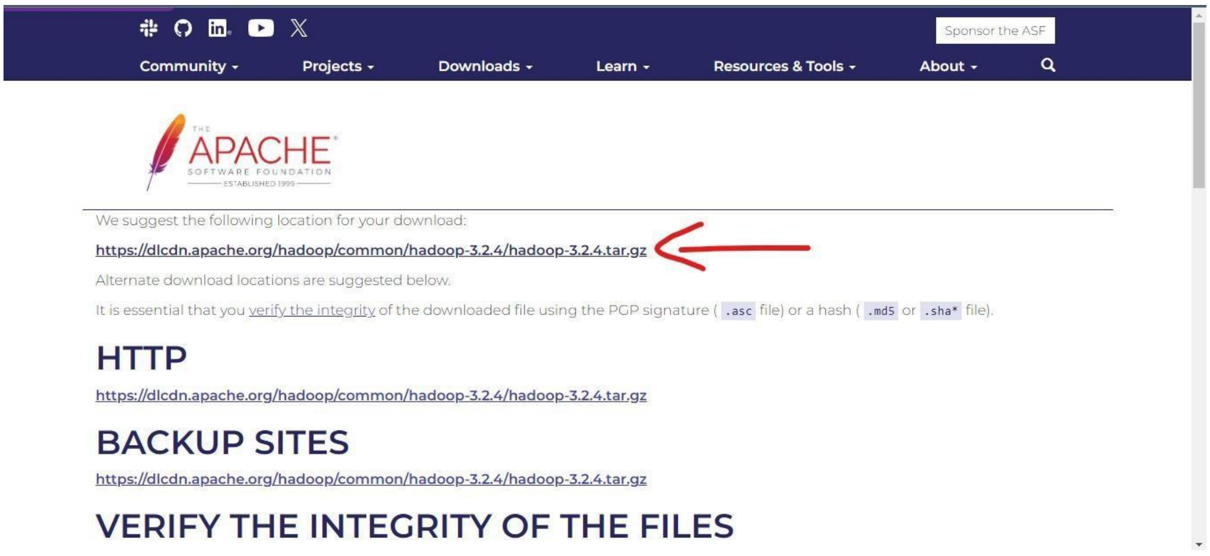
2. Java 8 development Kit (JDK)

3. To unzip downloaded Hadoop binaries, we should install 7zip.

❖ Download Hadoop binaries

The first step is to download Hadoop binaries from the official website. we need to install Hadoop 3.2.4 The binary package size is about 470 MB.

https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz



We suggest the following location for your download:

https://dlcdn.apache.org/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz

Alternate download locations are suggested below.

It is essential that you verify the integrity of the downloaded file using the PGP signature (.asc file) or a hash (.md5 or .sha* file).

HTTP

https://dlcdn.apache.org/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz

BACKUP SITES

https://dlcdn.apache.org/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz

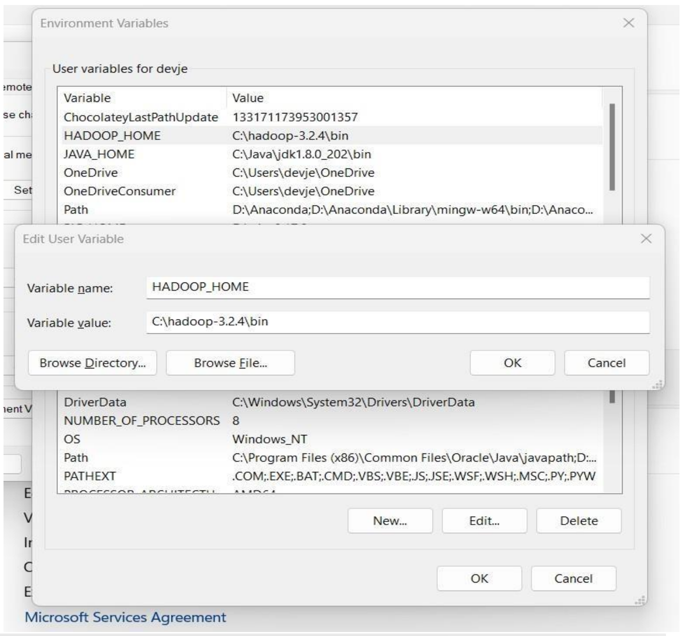
VERIFY THE INTEGRITY OF THE FILES

After finishing the file download, we should unpack the package using 7zip. First, we should extract the hadoop-3.2.1.tar.gz library, and then, we should unpack the extracted tar file.

❖ Setting up environment variables

After installing Hadoop and its prerequisites, we should configure the environment variables to define Hadoop and Java default paths.

To edit environment variables, go to Control Panel > System and Security > System (or right-click > properties on My Computer icon) and click on the “Advanced system settings” link.



Environment Variables

User variables for devje

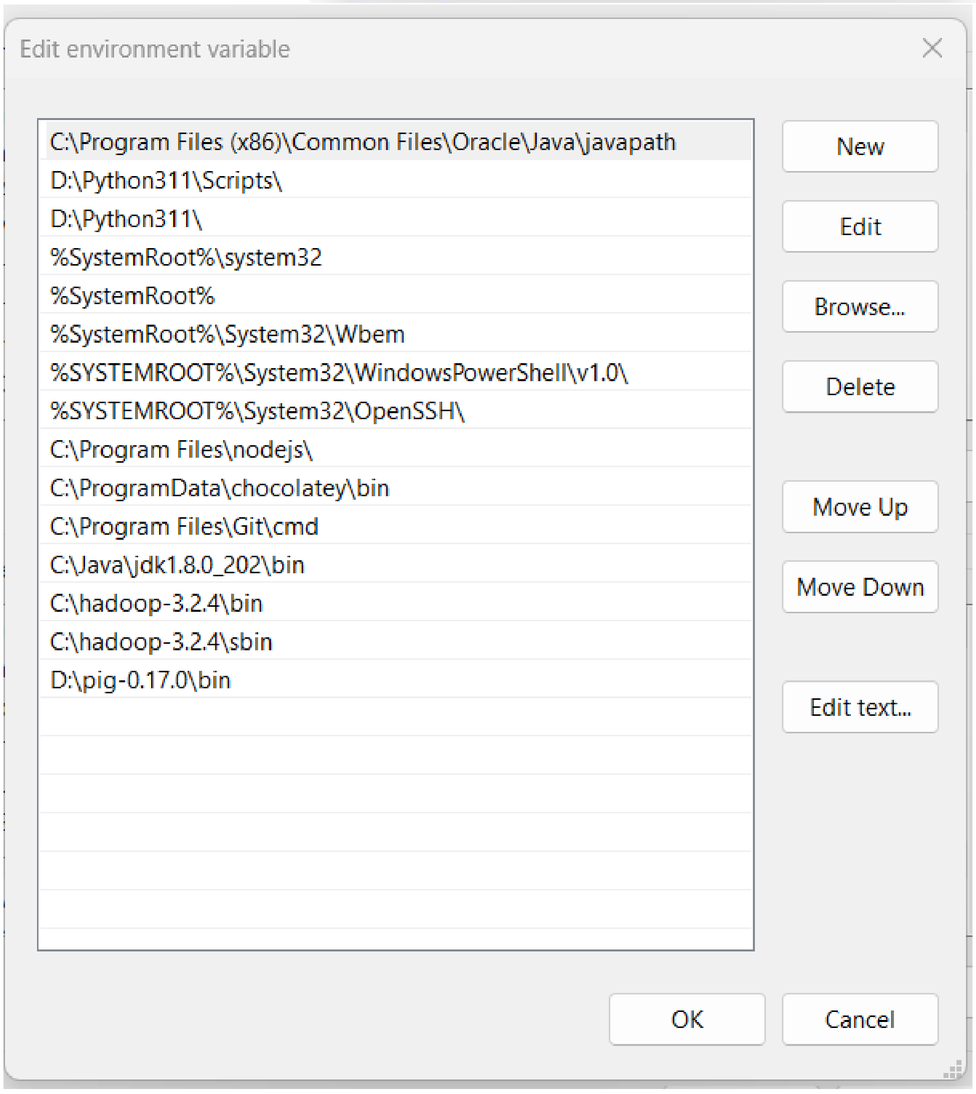
Variable	Value
ChocolateyLastPathUpdate	133171173953001357
HADOOP_HOME	C:\hadoop-3.2.4\bin
JAVA_HOME	C:\Java\jdk1.8.0_202\bin
OneDrive	C:\Users\devje\OneDrive
OneDriveConsumer	C:\Users\devje\OneDrive
Path	D:\Anaconda;D:\Anaconda\Library\mingw-w64\bin;D:\Anaco...

Edit User Variable

Variable name: HADOOP_HOME

Variable value: C:\hadoop-3.2.4\bin

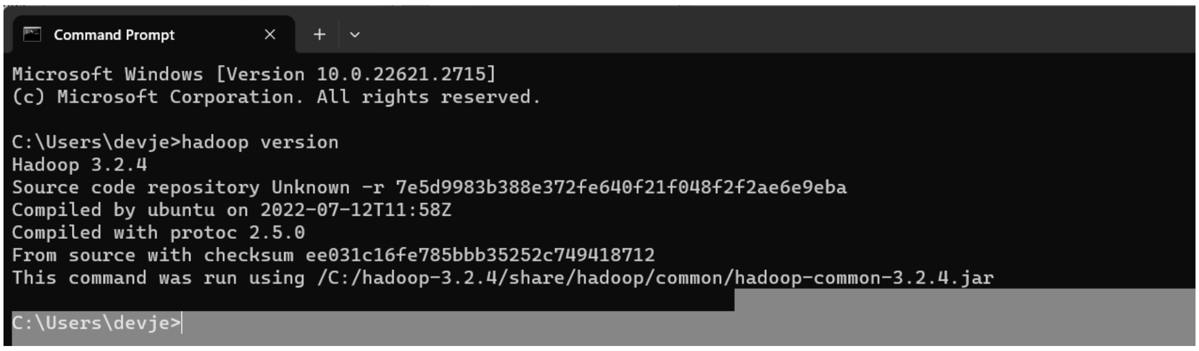
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;D:...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW



Edit environment variable

```
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
D:\Python311\Scripts\
D:\Python311\
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\nodejs\
C:\ProgramData\chocolatey\bin
C:\Program Files\Git\cmd
C:\Java\jdk1.8.0_202\bin
C:\hadoop-3.2.4\bin
C:\hadoop-3.2.4\sbin
D:\pig-0.17.0\bin
```

Checking java and hadoop installation :



```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\devje>hadoop version
Hadoop 3.2.4
Source code repository Unknown -r 7e5d9983b388e372fe640f21f048f2f2ae6e9eba
Compiled by ubuntu on 2022-07-12T11:58Z
Compiled with protoc 2.5.0
From source with checksum ee031c16fe785bbb35252c749418712
This command was run using /C:/hadoop-3.2.4/share/hadoop/common/hadoop-common-3.2.4.jar

C:\Users\devje>
```

❖ **Configuring Hadoop cluster**

There are four files we should alter to configure Hadoop cluster:

1. %HADOOP_HOME%\etc\hadoop\hdfs-site.xml

   As we know, Hadoop is built using a master-slave paradigm. Before altering the HDFS configuration file, we should create a directory to store all master node (name node) data and another one to store data (data node). In this example, we created the following directories:

   D:\hadoop\hadoop-3.2.4\data\namenode

   D:\hadoop\hadoop-3.2.4\data\datanode

   Now, let's open "hdfs-site.xml" file located in "%HADOOP_HOME%\etc\hadoop" directory, and we should add the following properties within the <configuration></configuration> element:

   ```
   <property>
   <name>dfs.replication</name>
   <value>1</value>
   </property>
   <property>
   <name>dfs.namenode.name.dir</name>
   <value> file:/// D:\hadoop\hadoop-3.2.4\data\namenode</value>
   </property>
   <property>
   <name>dfs.datanode.data.dir</name>
   <value> file:/// D:\hadoop\hadoop-3.2.4\data\datanode</value>
   </property>
   ```

   Note that we have set the replication factor to 1 since we are creating a single node cluster.

2. %HADOOP_HOME%\etc\hadoop\core-site.xml

   Now, we should configure the name node URL adding the following XML code into the <configuration></configuration> element within "core-site.xml":

   ```
   <property>
   <name>fs.default.name</name>
   ```

```
<value>hdfs://localhost:9820</value>
</property>
```

3. %HADOOP_HOME%\etc\hadoop\mapred-site.xml

   Now, we should add the following XML code into the <configuration></configuration> element within “mapred-site.xml”

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
<description>MapReduce framework name</description>
</property>
```

4. %HADOOP_HOME%\etc\hadoop\yarn-site.xml

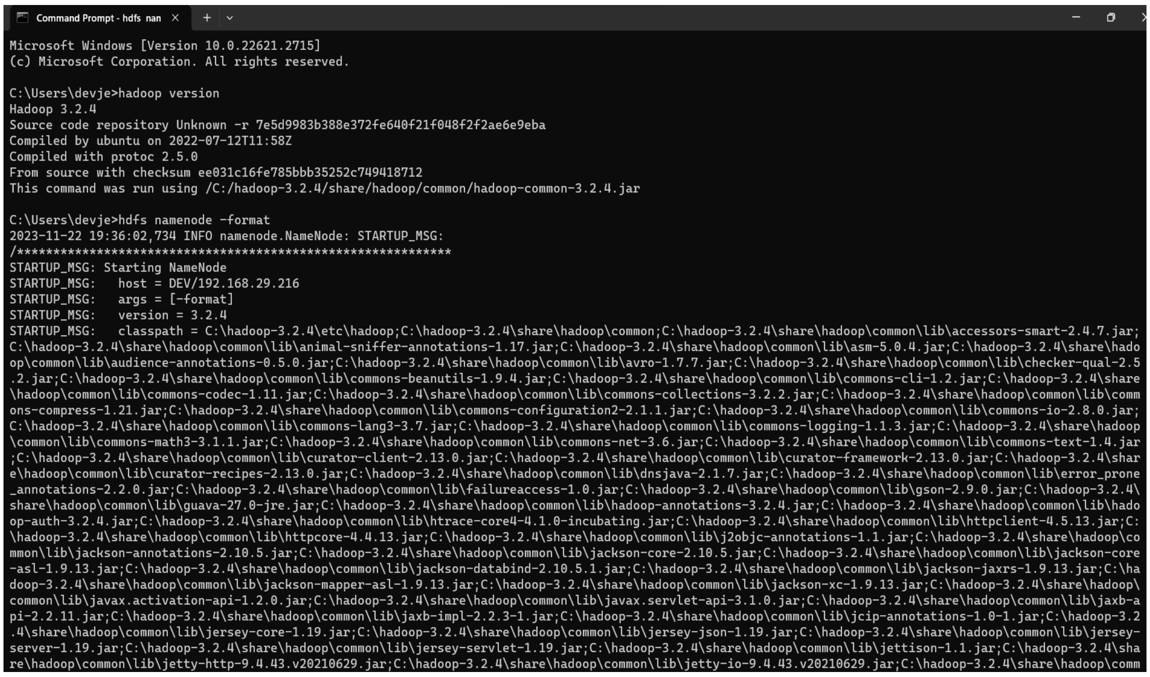
   Now, we should add the following XML code into the <configuration></configuration> element within “yarn-site.xml”

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
<description>Yarn Node Manager Aux Service</description>
</property>
```

- **Formatting Name node**

  After finishing the configuration, let’s try to format the name node using the following command:

```
hdfs namenode -format
```



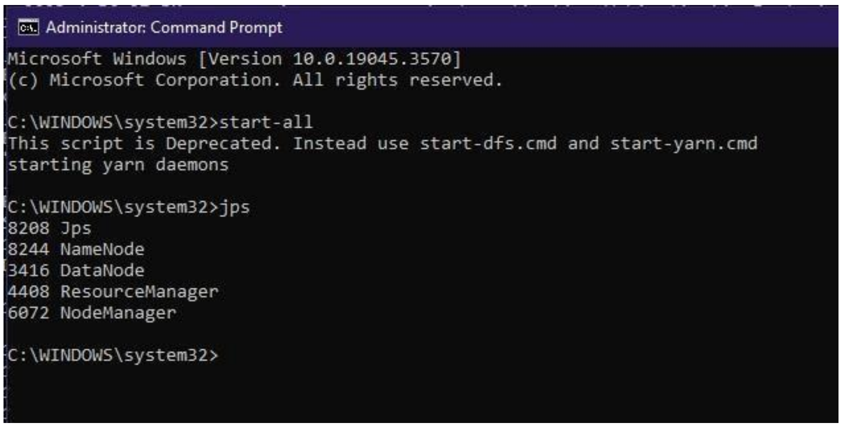
```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\devje>hadoop version
Hadoop 3.2.4
Source code repository Unknown -r 7e5d9983b388e372fe640f21f048f2f2ae6e9eba
Compiled by ubuntu on 2022-07-12T11:58Z
Compiled with protoc 2.5.0
From source with checksum ee031c16fe785bbb35252c749418712
This command was run using /C:/hadoop-3.2.4/share/hadoop/common/hadoop-common-3.2.4.jar

C:\Users\devje>hdfs namenode -format
2023-11-22 19:36:02,734 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = DEV/192.168.29.216
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.2.4
STARTUP_MSG:   classpath = C:\hadoop-3.2.4\etc\hadoop;C:\hadoop-3.2.4\share\hadoop\common;C:\hadoop-3.2.4\share\hadoop\common\lib\accessors-smart-2.4.7.jar;
C:\hadoop-3.2.4\share\hadoop\common\lib\animal-sniffer-annotations-1.17.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\asm-5.0.4.jar;C:\hadoop-3.2.4\share\hado
op\common\lib\audience-annotations-0.5.0.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\avro-1.7.7.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\checker-qual-2.5
.2.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\commons-beanutils-1.9.4.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\commons-cli-1.2.jar;C:\hadoop-3.2.4\share
\hadoop\common\lib\commons-codec-1.11.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\commons-collections-3.2.2.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\comm
ons-compress-1.21.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\commons-configuration2-2.1.1.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\commons-io-2.8.0.jar;
C:\hadoop-3.2.4\share\hadoop\common\lib\commons-lang3-3.7.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\commons-logging-1.1.3.jar;C:\hadoop-3.2.4\share\hadoop
\common\lib\commons-math3-3.1.1.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\commons-net-3.6.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\commons-text-1.4.jar
;C:\hadoop-3.2.4\share\hadoop\common\lib\curator-client-2.13.0.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\curator-framework-2.13.0.jar;C:\hadoop-3.2.4\shar
e\hadoop\common\lib\curator-recipes-2.13.0.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\dnsjava-2.1.7.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\error_prone
_annotations-2.2.0.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\failureaccess-1.0.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\gson-2.9.0.jar;C:\hadoop-3.2.4\
share\hadoop\common\lib\guava-27.0-jre.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\hadoop-annotations-3.2.4.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\hado
op-auth-3.2.4.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\htrace-core4-4.1.0-incubating.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\httpclient-4.5.13.jar;C:
\hadoop-3.2.4\share\hadoop\common\lib\httpcore-4.4.13.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\j2objc-annotations-1.1.jar;C:\hadoop-3.2.4\share\hadoop\co
mmon\lib\jackson-annotations-2.10.5.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jackson-core-2.10.5.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jackson-core
-asl-1.9.13.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jackson-databind-2.10.5.1.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jackson-jaxrs-1.9.13.jar;C:\ha
doop-3.2.4\share\hadoop\common\lib\jackson-mapper-asl-1.9.13.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jackson-xc-1.9.13.jar;C:\hadoop-3.2.4\share\hadoop\
common\lib\javax.activation-api-1.2.0.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\javax.servlet-api-3.1.0.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jaxb-a
pi-2.2.11.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jaxb-impl-2.2.3-1.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jcip-annotations-1.0-1.jar;C:\hadoop-3.2
.4\share\hadoop\common\lib\jersey-core-1.19.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jersey-json-1.19.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jersey-
server-1.19.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jersey-servlet-1.19.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jettison-1.1.jar;C:\hadoop-3.2.4\sha
re\hadoop\common\lib\jetty-http-9.4.43.v20210629.jar;C:\hadoop-3.2.4\share\hadoop\common\lib\jetty-io-9.4.43.v20210629.jar;C:\hadoop-3.2.4\share\hadoop\comm
```

❖ Starting Hadoop services

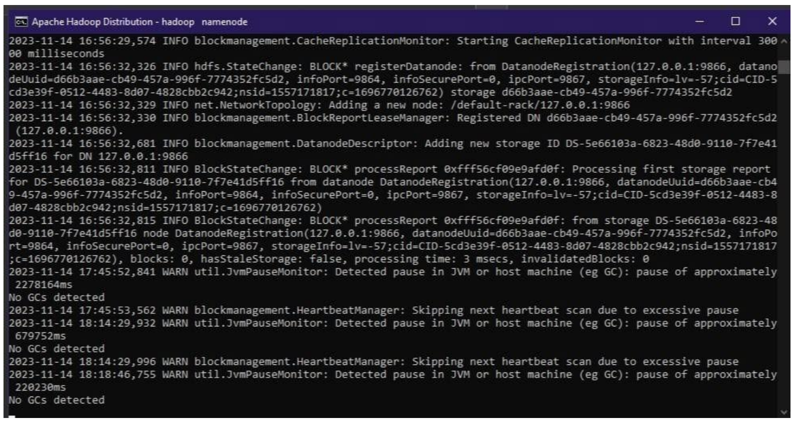
Now, we will open PowerShell, and navigate to "%HADOOP_HOME%\sbin" directory or just open cmd as admin. Then we will run the following command to start the Hadoop nodes:

```
start-all
```

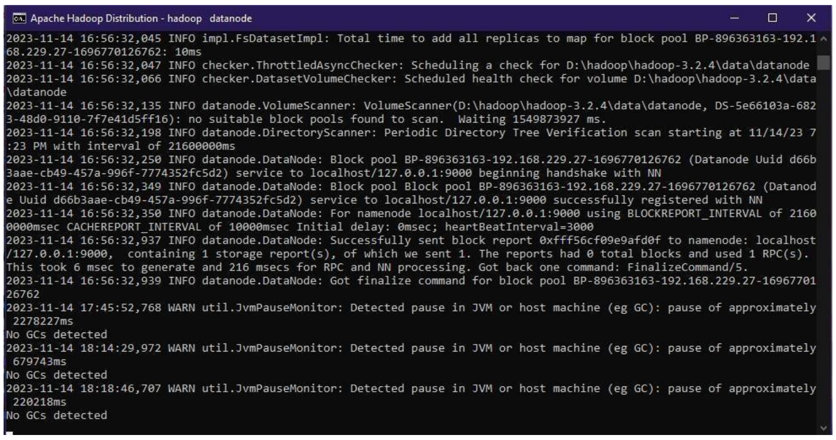
This will run both dfs and yarn, must have to run all 4 terminal , no one have to shutdown , than installation was successful also check this with 'jps' it display all running services.



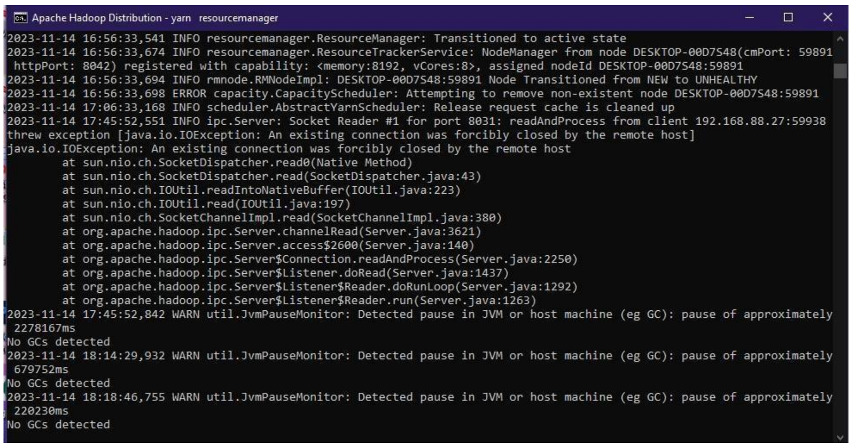
```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>start-all
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\WINDOWS\system32>jps
8208 Jps
8244 NameNode
3416 DataNode
4408 ResourceManager
6072 NodeManager

C:\WINDOWS\system32>
```



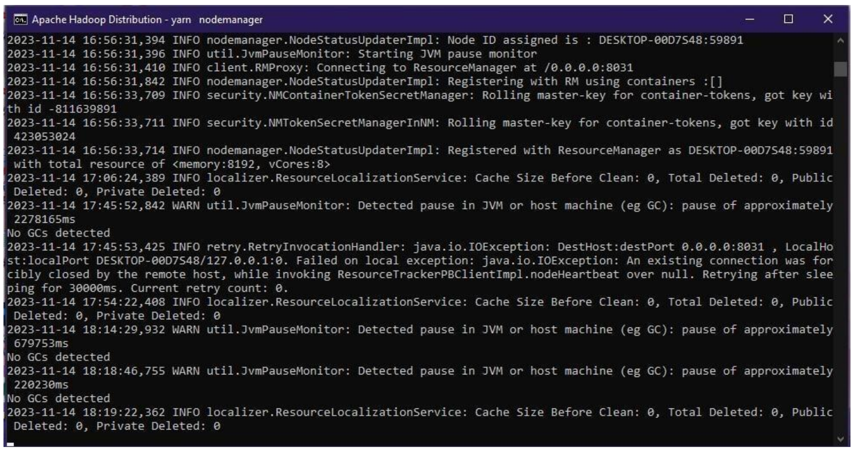
```
2023-11-14 16:56:29,574 INFO blockmanagement.CacheReplicationMonitor: Starting CacheReplicationMonitor with interval 300
00 milliseconds
2023-11-14 16:56:32,326 INFO hdfs.StateChange: BLOCK* registerDatanode: from DatanodeRegistration(127.0.0.1:9866, datano
deUuid=d66b3aae-cb49-457a-996f-7774352fc5d2, infoPort=9864, infoSecurePort=0, ipcPort=9867, storageInfo=lv=-57;cid=CID-5
cd3e39f-0512-4483-8d07-4828cbb2c942;nsid=1557171817;c=1696770126762) storage d66b3aae-cb49-457a-996f-7774352fc5d2
2023-11-14 16:56:32,329 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866
2023-11-14 16:56:32,330 INFO blockmanagement.BlockReportLeaseManager: Registered DN d66b3aae-cb49-457a-996f-7774352fc5d2
 (127.0.0.1:9866).
2023-11-14 16:56:32,681 INFO blockmanagement.DatanodeDescriptor: Adding new storage ID DS-5e66103a-6823-48d0-9110-7f7e41
d5ff16 for DN 127.0.0.1:9866
2023-11-14 16:56:32,811 INFO BlockStateChange: BLOCK* processReport 0xfff56cf09e9afd0f: Processing first storage report
for DS-5e66103a-6823-48d0-9110-7f7e41d5ff16 from datanode DatanodeRegistration(127.0.0.1:9866, datanodeUuid=d66b3aae-cb4
9-457a-996f-7774352fc5d2, infoPort=9864, infoSecurePort=0, ipcPort=9867, storageInfo=lv=-57;cid=CID-5cd3e39f-0512-4483-8
d07-4828cbb2c942;nsid=1557171817;c=1696770126762)
2023-11-14 16:56:32,815 INFO BlockStateChange: BLOCK* processReport 0xfff56cf09e9afd0f: from storage DS-5e66103a-6823-48
d0-9110-7f7e41d5ff16 node DatanodeRegistration(127.0.0.1:9866, datanodeUuid=d66b3aae-cb49-457a-996f-7774352fc5d2, infoPo
rt=9864, infoSecurePort=0, ipcPort=9867, storageInfo=lv=-57;cid=CID-5cd3e39f-0512-4483-8d07-4828cbb2c942;nsid=1557171817
;c=1696770126762), blocks: 0, hasStaleStorage: false, processing time: 3 msecs, invalidatedBlocks: 0
2023-11-14 17:45:52,841 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 2278164ms
No GCs detected
2023-11-14 17:45:53,562 WARN blockmanagement.HeartbeatManager: Skipping next heartbeat scan due to excessive pause
2023-11-14 18:14:29,932 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 679752ms
No GCs detected
2023-11-14 18:14:29,996 WARN blockmanagement.HeartbeatManager: Skipping next heartbeat scan due to excessive pause
2023-11-14 18:18:46,755 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 220230ms
No GCs detected
```



```
2023-11-14 16:56:32,045 INFO impl.FsDatasetImpl: Total time to add all replicas to map for block pool BP-896363163-192.1
68.229.27-1696770126762: 10ms
2023-11-14 16:56:32,047 INFO checker.ThrottledAsyncChecker: Scheduling a check for D:\hadoop\hadoop-3.2.4\data\datanode
2023-11-14 16:56:32,066 INFO checker.DatasetVolumeChecker: Scheduled health check for volume D:\hadoop\hadoop-3.2.4\data
\datanode
2023-11-14 16:56:32,135 INFO datanode.VolumeScanner: VolumeScanner(D:\hadoop\hadoop-3.2.4\data\datanode, DS-5e66103a-682
3-48d0-9110-7f7e41d5ff16): no suitable block pools found to scan.  Waiting 1549873927 ms.
2023-11-14 16:56:32,198 INFO datanode.DirectoryScanner: Periodic Directory Tree Verification scan starting at 11/14/23 7
:23 PM with interval of 21600000ms
2023-11-14 16:56:32,250 INFO datanode.DataNode: Block pool BP-896363163-192.168.229.27-1696770126762 (Datanode Uuid d66b
3aae-cb49-457a-996f-7774352fc5d2) service to localhost/127.0.0.1:9000 beginning handshake with NN
2023-11-14 16:56:32,349 INFO datanode.DataNode: Block pool Block pool BP-896363163-192.168.229.27-1696770126762 (Datanod
e Uuid d66b3aae-cb49-457a-996f-7774352fc5d2) service to localhost/127.0.0.1:9000 successfully registered with NN
2023-11-14 16:56:32,350 INFO datanode.DataNode: For namenode localhost/127.0.0.1:9000 using BLOCKREPORT_INTERVAL of 2160
0000msec CACHEREPORT_INTERVAL of 10000msec Initial delay: 0msec; heartBeatInterval=3000
2023-11-14 16:56:32,937 INFO datanode.DataNode: Successfully sent block report 0xfff56cf09e9afd0f to namenode: localhost
/127.0.0.1:9000,  containing 1 storage report(s), of which we sent 1. The reports had 0 total blocks and used 1 RPC(s).
This took 6 msec to generate and 216 msecs for RPC and NN processing. Got back one command: FinalizeCommand/5.
2023-11-14 16:56:32,939 INFO datanode.DataNode: Got finalize command for block pool BP-896363163-192.168.229.27-16967701
26762
2023-11-14 17:45:52,768 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 2278227ms
No GCs detected
2023-11-14 18:14:29,972 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 679743ms
No GCs detected
2023-11-14 18:18:46,707 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 220218ms
No GCs detected
```



```
2023-11-14 16:56:33,541 INFO resourcemanager.ResourceManager: Transitioned to active state
2023-11-14 16:56:33,674 INFO resourcemanager.ResourceTrackerService: NodeManager from node DESKTOP-00D7S48(cmPort: 59891
 httpPort: 8042) registered with capability: <memory:8192, vCores:8>, assigned nodeId DESKTOP-00D7S48:59891
2023-11-14 16:56:33,694 INFO rmnode.RMNodeImpl: DESKTOP-00D7S48:59891 Node Transitioned from NEW to UNHEALTHY
2023-11-14 16:56:33,698 ERROR capacity.CapacityScheduler: Attempting to remove non-existent node DESKTOP-00D7S48:59891
2023-11-14 17:06:33,168 INFO scheduler.AbstractYarnScheduler: Release request cache is cleaned up
2023-11-14 17:45:52,551 INFO ipc.Server: Socket Reader #1 for port 8031: readAndProcess from client 192.168.88.27:59938
threw exception [java.io.IOException: An existing connection was forcibly closed by the remote host]
java.io.IOException: An existing connection was forcibly closed by the remote host
        at sun.nio.ch.SocketDispatcher.read0(Native Method)
        at sun.nio.ch.SocketDispatcher.read(SocketDispatcher.java:43)
        at sun.nio.ch.IOUtil.readIntoNativeBuffer(IOUtil.java:223)
        at sun.nio.ch.IOUtil.read(IOUtil.java:197)
        at sun.nio.ch.SocketChannelImpl.read(SocketChannelImpl.java:380)
        at org.apache.hadoop.ipc.Server.channelRead(Server.java:3621)
        at org.apache.hadoop.ipc.Server.access$2600(Server.java:140)
        at org.apache.hadoop.ipc.Server$Connection.readAndProcess(Server.java:2250)
        at org.apache.hadoop.ipc.Server$Listener.doRead(Server.java:1437)
        at org.apache.hadoop.ipc.Server$Listener$Reader.doRunLoop(Server.java:1292)
        at org.apache.hadoop.ipc.Server$Listener$Reader.run(Server.java:1263)
2023-11-14 17:45:52,842 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 2278167ms
No GCs detected
2023-11-14 18:14:29,932 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 679752ms
No GCs detected
2023-11-14 18:18:46,755 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 220230ms
No GCs detected
```

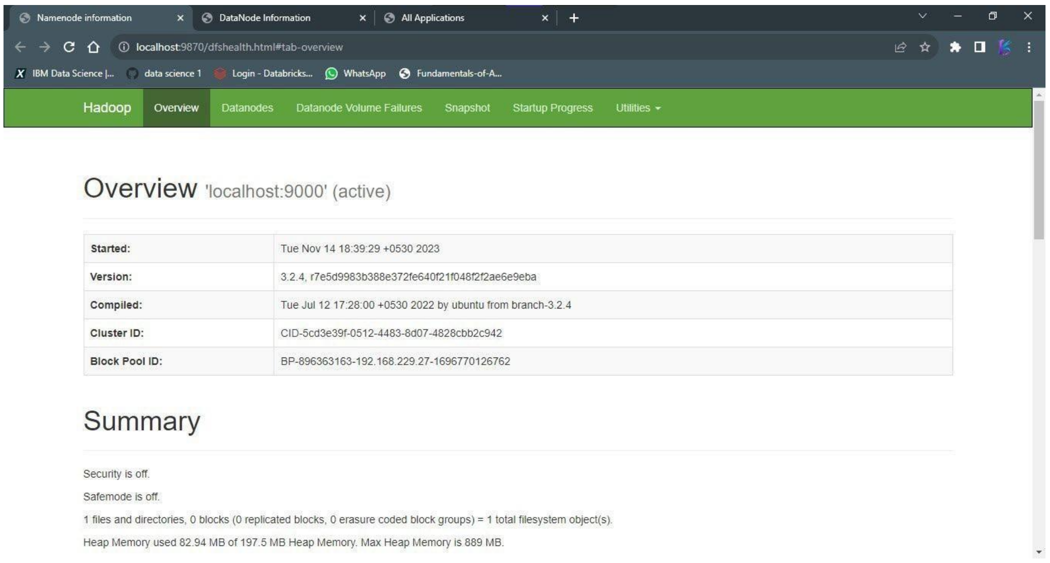


```
2023-11-14 16:56:31,394 INFO nodemanager.NodeStatusUpdaterImpl: Node ID assigned is : DESKTOP-00D7S48:59891
2023-11-14 16:56:31,396 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2023-11-14 16:56:31,410 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8031
2023-11-14 16:56:31,842 INFO nodemanager.NodeStatusUpdaterImpl: Registering with RM using containers :[]
2023-11-14 16:56:33,709 INFO security.NMContainerTokenSecretManager: Rolling master-key for container-tokens, got key wi
th id -811639891
2023-11-14 16:56:33,711 INFO security.NMTokenSecretManagerInNM: Rolling master-key for container-tokens, got key with id
 423053024
2023-11-14 16:56:33,714 INFO nodemanager.NodeStatusUpdaterImpl: Registered with ResourceManager as DESKTOP-00D7S48:59891
 with total resource of <memory:8192, vCores:8>
2023-11-14 17:06:24,389 INFO localizer.ResourceLocalizationService: Cache Size Before Clean: 0, Total Deleted: 0, Public
 Deleted: 0, Private Deleted: 0
2023-11-14 17:45:52,842 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 2278165ms
No GCs detected
2023-11-14 17:45:53,425 INFO retry.RetryInvocationHandler: java.io.IOException: DestHost:destPort 0.0.0.0:8031 , LocalHo
st:localPort DESKTOP-00D7S48/127.0.0.1:0. Failed on local exception: java.io.IOException: An existing connection was for
cibly closed by the remote host, while invoking ResourceTrackerPBClientImpl.nodeHeartbeat over null. Retrying after slee
ping for 30000ms. Current retry count: 0.
2023-11-14 17:54:22,408 INFO localizer.ResourceLocalizationService: Cache Size Before Clean: 0, Total Deleted: 0, Public
 Deleted: 0, Private Deleted: 0
2023-11-14 18:14:29,932 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 679753ms
No GCs detected
2023-11-14 18:18:46,755 WARN util.JvmPauseMonitor: Detected pause in JVM or host machine (eg GC): pause of approximately
 220230ms
No GCs detected
2023-11-14 18:19:22,362 INFO localizer.ResourceLocalizationService: Cache Size Before Clean: 0, Total Deleted: 0, Public
 Deleted: 0, Private Deleted: 0
```

❖ Hadoop Web UI

There are three web user interfaces to be used:

Name node web page: http://localhost:9870/dfshealth.html



Overview 'localhost:9000' (active)

Started:	Tue Nov 14 18:39:29 +0530 2023
Version:	3.2.4, r7e5d9983b388e372fe640f21f048f2f2ae6e9eba
Compiled:	Tue Jul 12 17:28:00 +0530 2022 by ubuntu from branch-3.2.4
Cluster ID:	CID-5cd3e39f-0512-4483-8d07-4828cbb2c942
Block Pool ID:	BP-896363163-192.168.229.27-1696770126762

Summary

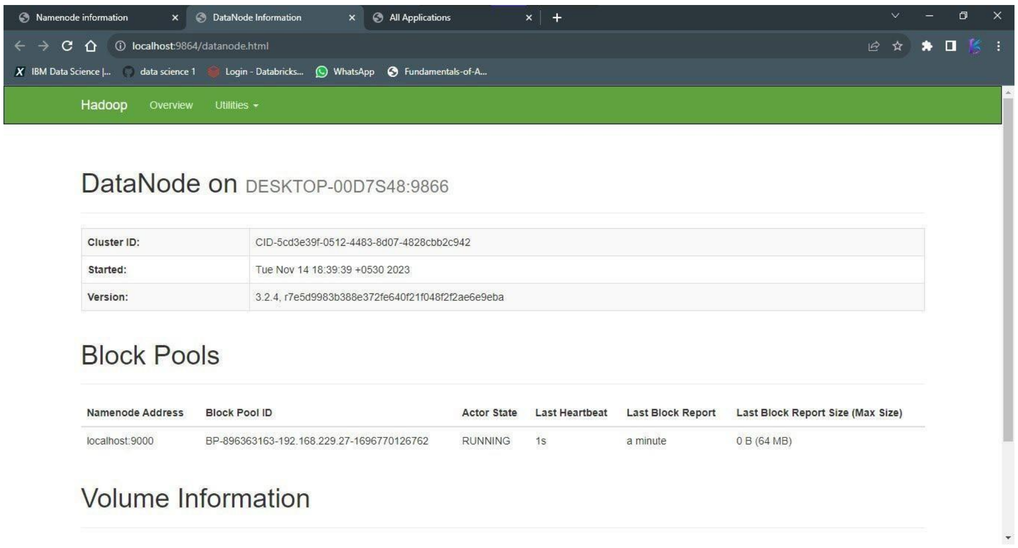
Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 82.94 MB of 197.5 MB Heap Memory. Max Heap Memory is 889 MB.

Data node web page: http://localhost:9864/datanode.html



DataNode on DESKTOP-00D7S48:9866

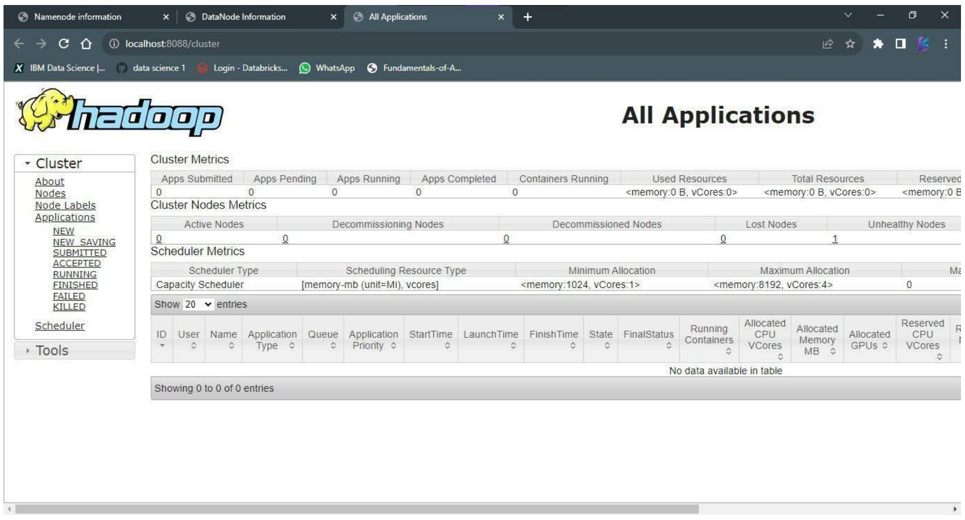
Cluster ID:	CID-5cd3e39f-0512-4483-8d07-4828cbb2c942
Started:	Tue Nov 14 18:39:39 +0530 2023
Version:	3.2.4, r7e5d9983b388e372fe640f21f048f2f2ae6e9eba

Block Pools

Namenode Address	Block Pool ID	Actor State	Last Heartbeat	Last Block Report	Last Block Report Size (Max Size)
localhost:9000	BP-896363163-192.168.229.27-1696770126762	RUNNING	1s	a minute	0 B (64 MB)

Volume Information

Yarn web page: http://localhost:8088/cluster



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:0 B, vCores:0>	<memory:0 B

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
0	0	0	0	1

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Ma
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Allocated GPUs	Reserved CPU VCores
No data available in table															

Showing 0 to 0 of 0 entries

Pig Installation

Prepared by:

VASAVA VIPULKUMAR DINESHBHAI
(23MDS003)
Branch: MTech (Data Science)

❖ Prerequisites

1. Java 8 development Kit (JDK)

2. Hadoop 3.2.4 must be installed.

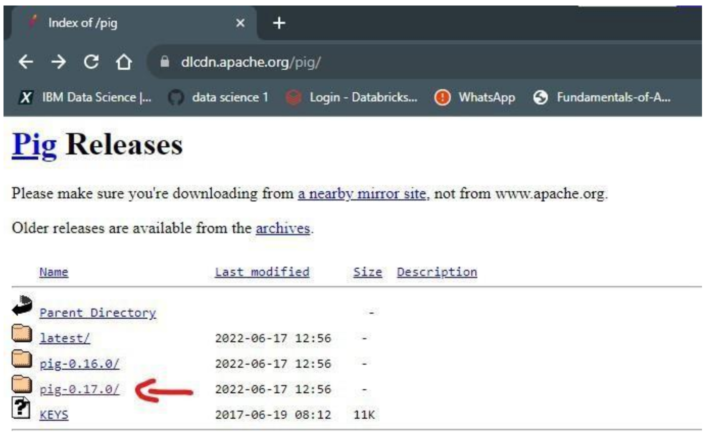
3. 7zip

❖ Download PIG file

The first step is to download pig form the apache official server , it can be downloaded at following link :

https://dlcdn.apache.org/pig/

under this link go to the pig-0.17.0/ and then download pig-0.17.0.tar.gz this file. The package size is about 220MB.



Index of /pig
dlcdn.apache.org/pig/

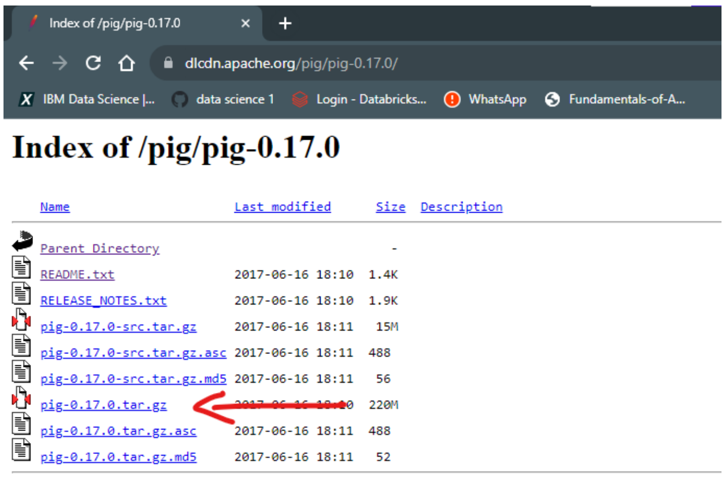
Pig Releases

Please make sure you're downloading from a nearby mirror site, not from www.apache.org.

Older releases are available from the archives.

```
Name                   Last modified      Size  Description
Parent Directory                           -
latest/                2022-06-17 12:56    -
pig-0.16.0/            2022-06-17 12:56    -
pig-0.17.0/            2022-06-17 12:56    -
KEYS                   2017-06-19 08:12   11K
```

`



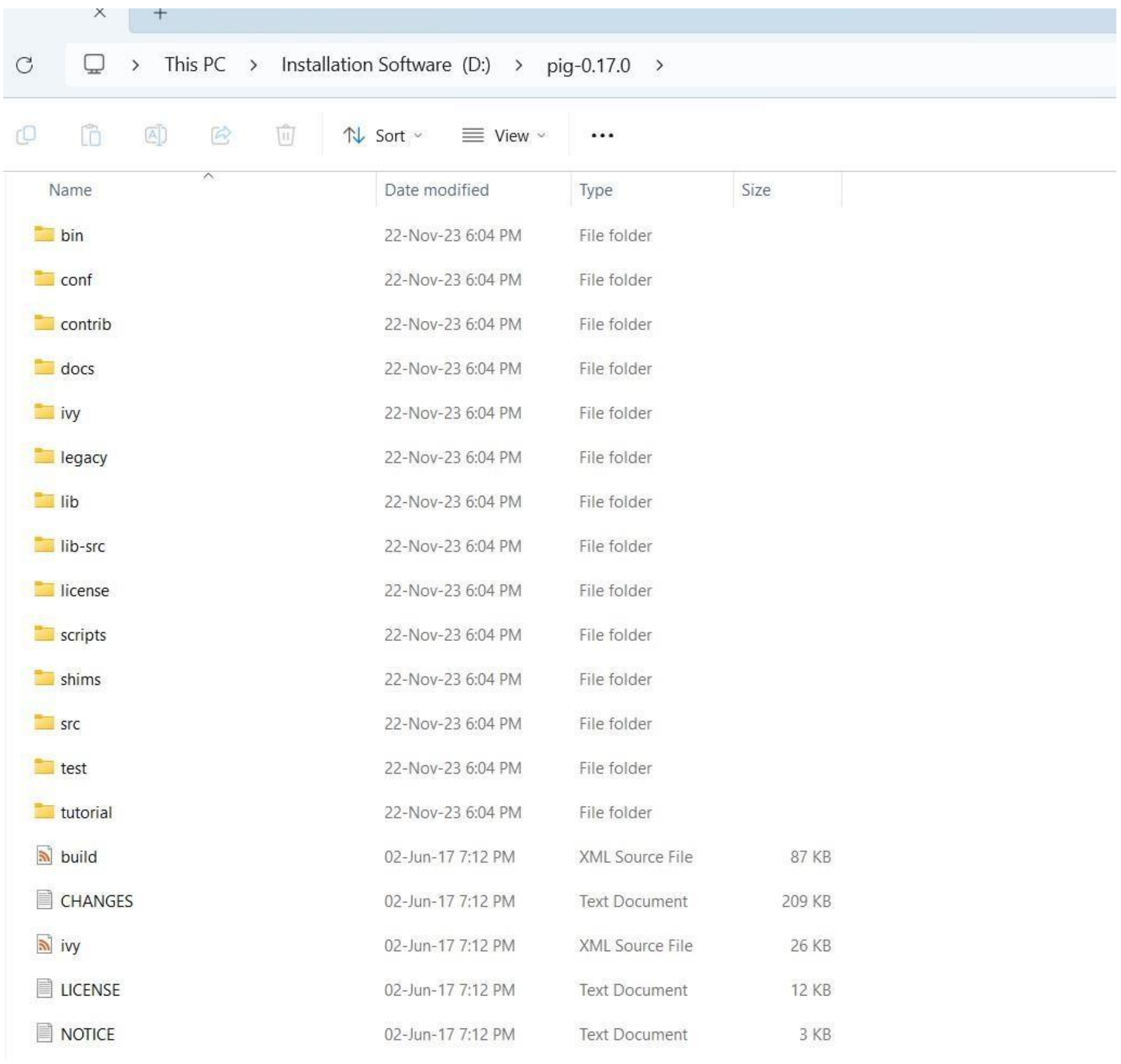
Index of /pig/pig-0.17.0

```
Name                       Last modified     Size  Description

Parent Directory                               -
README.txt                 2017-06-16 18:10  1.4K
RELEASE_NOTES.txt          2017-06-16 18:10  1.9K
pig-0.17.0-src.tar.gz      2017-06-16 18:11   15M
pig-0.17.0-src.tar.gz.asc  2017-06-16 18:11  488
pig-0.17.0-src.tar.gz.md5  2017-06-16 18:11   56
pig-0.17.0.tar.gz          2017-06-16 18:10  220M
pig-0.17.0.tar.gz.asc      2017-06-16 18:11  488
pig-0.17.0.tar.gz.md5      2017-06-16 18:11   52
```

❖ Unzip and Install PIG

After Downloading the PIG, we need to Unzip the pig-0.17.0.tar.gz file.

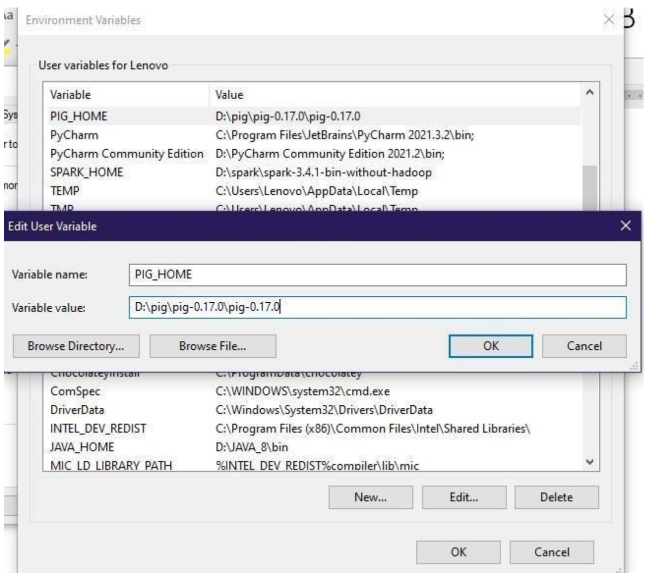
After unzipping the folder it will be look like this:



Name	Date modified	Type	Size
bin	22-Nov-23 6:04 PM	File folder	
conf	22-Nov-23 6:04 PM	File folder	
contrib	22-Nov-23 6:04 PM	File folder	
docs	22-Nov-23 6:04 PM	File folder	
ivy	22-Nov-23 6:04 PM	File folder	
legacy	22-Nov-23 6:04 PM	File folder	
lib	22-Nov-23 6:04 PM	File folder	
lib-src	22-Nov-23 6:04 PM	File folder	
license	22-Nov-23 6:04 PM	File folder	
scripts	22-Nov-23 6:04 PM	File folder	
shims	22-Nov-23 6:04 PM	File folder	
src	22-Nov-23 6:04 PM	File folder	
test	22-Nov-23 6:04 PM	File folder	
tutorial	22-Nov-23 6:04 PM	File folder	
build	02-Jun-17 7:12 PM	XML Source File	87 KB
CHANGES	02-Jun-17 7:12 PM	Text Document	209 KB
ivy	02-Jun-17 7:12 PM	XML Source File	26 KB
LICENSE	02-Jun-17 7:12 PM	Text Document	12 KB
NOTICE	02-Jun-17 7:12 PM	Text Document	3 KB

❖ Setting Up Environment Variables

- Another important step in setting up a work environment is to set your Systems environment variable.
- Go to Control Panel > System > click on the “Advanced system settings” link to edit environment variables.
- Open environment Variable and click on “New” in “User Variable”.
- On clicking “New”, we get the below screen.


Environment Variables

User variables for Lenovo

Variable	Value
PIG_HOME	D:\pig\pig-0.17.0\pig-0.17.0
PyCharm	C:\Program Files\JetBrains\PyCharm 2021.3.2\bin;
PyCharm Community Edition	D:\PyCharm Community Edition 2021.2\bin;
SPARK_HOME	D:\spark\spark-3.4.1-bin-without-hadoop
TEMP	C:\Users\Lenovo\AppData\Local\Temp
TMP	C:\Users\Lenovo\AppData\Local\Temp

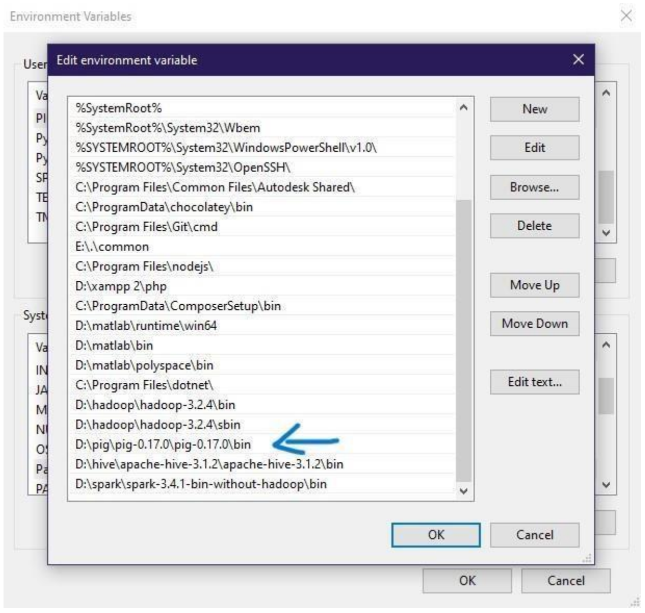
Edit User Variable

Variable name: PIG_HOME

Variable value: D:\pig\pig-0.17.0\pig-0.17.0

Variable	Value
ChocolateyInstall	C:\ProgramData\chocolatey
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
INTEL_DEV_REDIST	C:\Program Files (x86)\Common Files\Intel\Shared Libraries\
JAVA_HOME	D:\JAVA_8\bin
MIC_LD_LIBRARY_PATH	%INTEL_DEV_REDIST%compiler\lib\mic

- The last step in setting the Environment variable is setting Path in System Variable.


Environment Variables

Edit environment variable

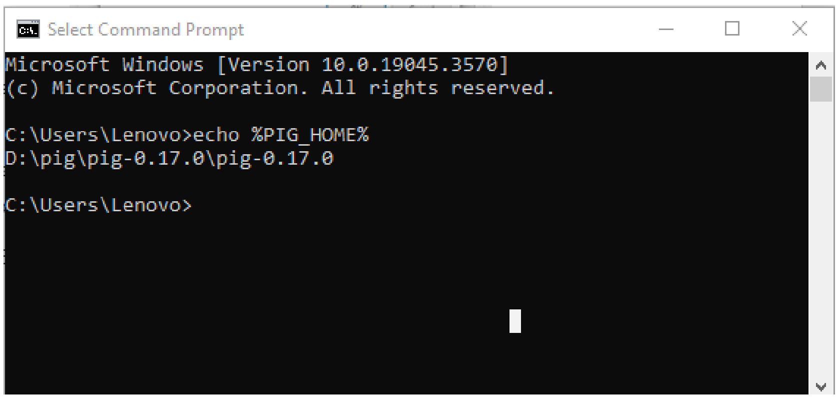
```
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Common Files\Autodesk Shared\
C:\ProgramData\chocolatey\bin
C:\Program Files\Git\cmd
E:\.\common
C:\Program Files\nodejs\
D:\xampp 2\php
C:\ProgramData\ComposerSetup\bin
D:\matlab\runtime\win64
D:\matlab\bin
D:\matlab\polyspace\bin
C:\Program Files\dotnet\
D:\hadoop\hadoop-3.2.4\bin
D:\hadoop\hadoop-3.2.4\sbin
D:\pig\pig-0.17.0\pig-0.17.0\bin
D:\hive\apache-hive-3.1.2\apache-hive-3.1.2\bin
D:\spark\spark-3.4.1-bin-without-hadoop\bin
```

❖ Verify the Paths

Now we need to verify that what we have done is correct and reflecting.

- Open a NEW Command Window
- Run following commands

```
echo %PIG_HOME%
```



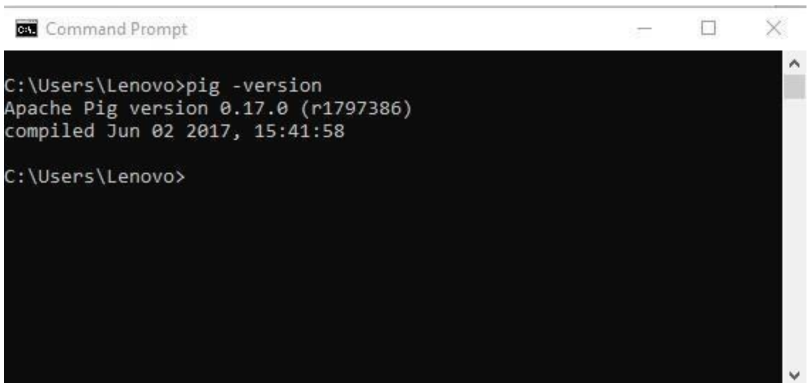
```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>echo %PIG_HOME%
D:\pig\pig-0.17.0\pig-0.17.0

C:\Users\Lenovo>
```

❖ Verifying Setup

- We are done with setting up the PIG on our System.
- Now we need to check if everything works smoothly…
- Open a cmd window, run the below command to test the connection and PIG.

```
pig -version
```

Upon running the command we should get the version of PIG. i.e 0.17.0 in our case.



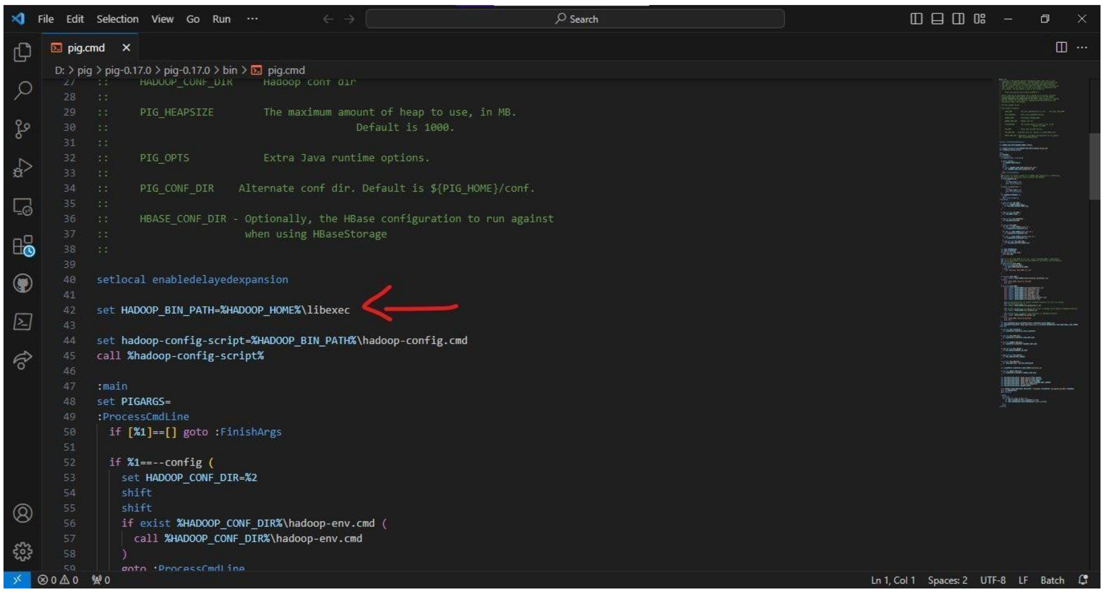
```
C:\Users\Lenovo>pig -version
Apache Pig version 0.17.0 (r1797386)
compiled Jun 02 2017, 15:41:58

C:\Users\Lenovo>
```

Don't worry some of us can get the below error after running pig -version

'-Xmx1000M' is not recognized as an internal or external command,operable program or batch file.

To resolve this we will need to perform the following steps:-

1. Open the pig. cmd file in edit mode. We can find the file in the bin folder.
2. Now we need to change the value of the HADOOP_BIN_PATH



```
::     HADOOP_CONF_DIR     Hadoop conf dir
::
::     PIG_HEAPSIZE        The maximum amount of heap to use, in MB.
::                                     Default is 1000.
::
::     PIG_OPTS            Extra Java runtime options.
::
::     PIG_CONF_DIR    Alternate conf dir. Default is ${PIG_HOME}/conf.
::
::     HBASE_CONF_DIR - Optionally, the HBase configuration to run against
::                   when using HBaseStorage
::

setlocal enabledelayedexpansion

set HADOOP_BIN_PATH=%HADOOP_HOME%\libexec

set hadoop-config-script=%HADOOP_BIN_PATH%\hadoop-config.cmd
call %hadoop-config-script%

:main
set PIGARGS=
:ProcessCmdLine
  if [%1]==[] goto :FinishArgs

  if %1==--config (
    set HADOOP_CONF_DIR=%2
    shift
    shift
    if exist %HADOOP_CONF_DIR%\hadoop-env.cmd (
      call %HADOOP_CONF_DIR%\hadoop-env.cmd
    )
    goto :ProcessCmdLine
```

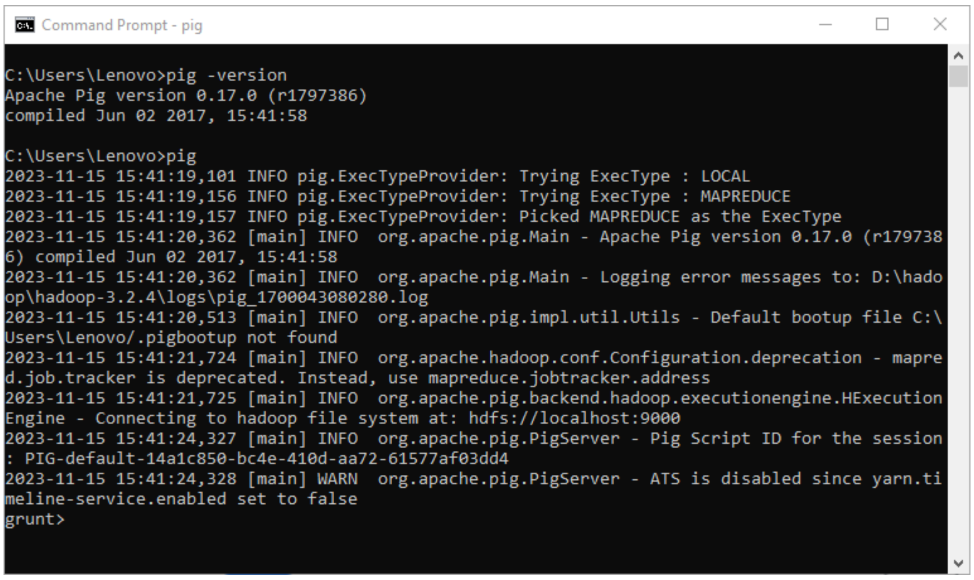
3. Save the file.

The next step is to verify the setup once again. So, we need to execute the pig-version command once again.

❖ Starting PIG

Now we need to start a new Command Prompt remember to run it as administrator to avoid permission issues and execute the below commands:

pig



```
C:\Users\Lenovo>pig -version
Apache Pig version 0.17.0 (r1797386)
compiled Jun 02 2017, 15:41:58

C:\Users\Lenovo>pig
2023-11-15 15:41:19,101 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2023-11-15 15:41:19,156 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2023-11-15 15:41:19,157 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2023-11-15 15:41:20,362 [main] INFO  org.apache.pig.Main - Apache Pig version 0.17.0 (r179738
6) compiled Jun 02 2017, 15:41:58
2023-11-15 15:41:20,362 [main] INFO  org.apache.pig.Main - Logging error messages to: D:\hado
op\hadoop-3.2.4\logs\pig_1700043080280.log
2023-11-15 15:41:20,513 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file C:\
Users\Lenovo/.pigbootup not found
2023-11-15 15:41:21,724 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapre
d.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address
2023-11-15 15:41:21,725 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecution
Engine - Connecting to hadoop file system at: hdfs://localhost:9000
2023-11-15 15:41:24,327 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the session
: PIG-default-14a1c850-bc4e-410d-aa72-61577af03dd4
2023-11-15 15:41:24,328 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.ti
meline-service.enabled set to false
grunt>
```

We can see grunt> once the pig starts.

Kafka Installation

Prepared by:

VASAVA VIPULKUMAR

DINESHBHA (23MDS003)

Branch: MTech. (Data Science)

❖ Prerequisites

1. Java 8 development Kit (JDK)

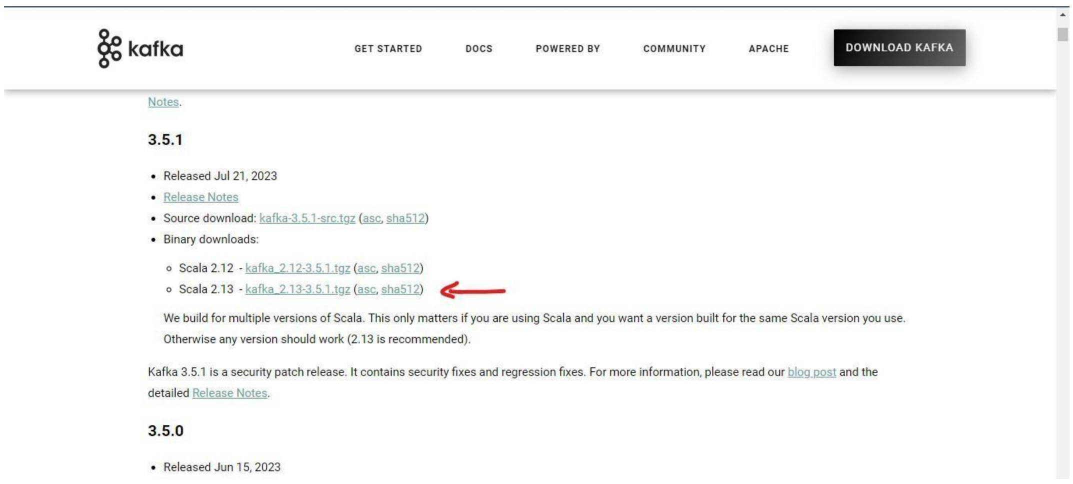
2. 7zip

❖ Download Kafka file

The first step is to download pig form the apache official server , it can be downloaded at following link :

https://kafka.apache.org/downloads

In this link download scala 2.13 version under binary downloads.



Notes.

3.5.1

- Released Jul 21, 2023
- Release Notes
- Source download: kafka-3.5.1-src.tgz (asc, sha512)
- Binary downloads:
  - Scala 2.12 - kafka_2.12-3.5.1.tgz (asc, sha512)
  - Scala 2.13 - kafka_2.13-3.5.1.tgz (asc, sha512)

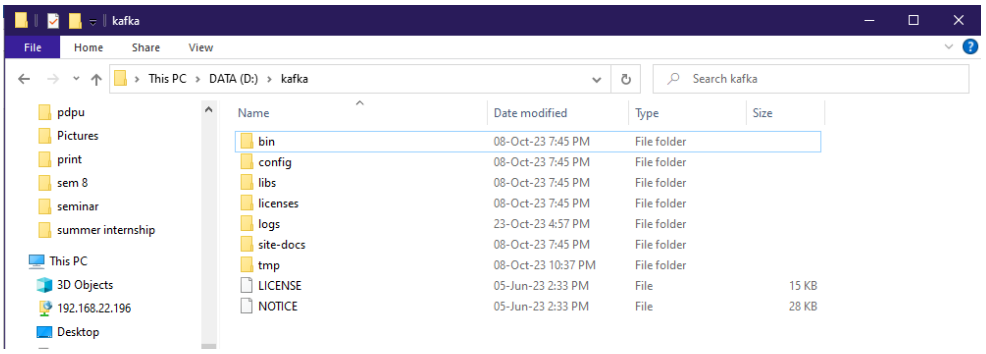
  We build for multiple versions of Scala. This only matters if you are using Scala and you want a version built for the same Scala version you use. Otherwise any version should work (2.13 is recommended).

Kafka 3.5.1 is a security patch release. It contains security fixes and regression fixes. For more information, please read our blog post and the detailed Release Notes.

3.5.0

- Released Jun 15, 2023

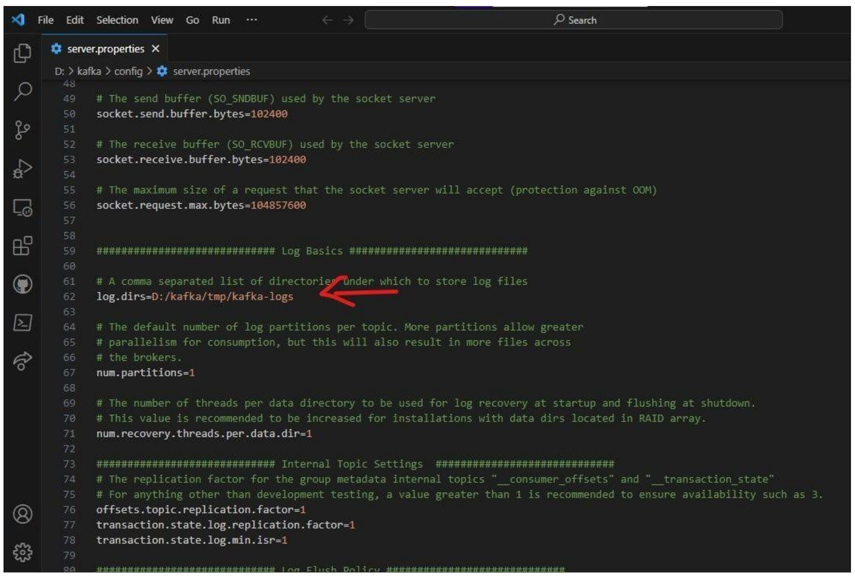
After downloading unzip this folder so it look like this



Name	Date modified	Type	Size
bin	08-Oct-23 7:45 PM	File folder	
config	08-Oct-23 7:45 PM	File folder	
libs	08-Oct-23 7:45 PM	File folder	
licenses	08-Oct-23 7:45 PM	File folder	
logs	23-Oct-23 4:57 PM	File folder	
site-docs	08-Oct-23 7:45 PM	File folder	
tmp	08-Oct-23 10:37 PM	File folder	
LICENSE	05-Jun-23 2:33 PM	File	15 KB
NOTICE	05-Jun-23 2:33 PM	File	28 KB

❖ Configure files

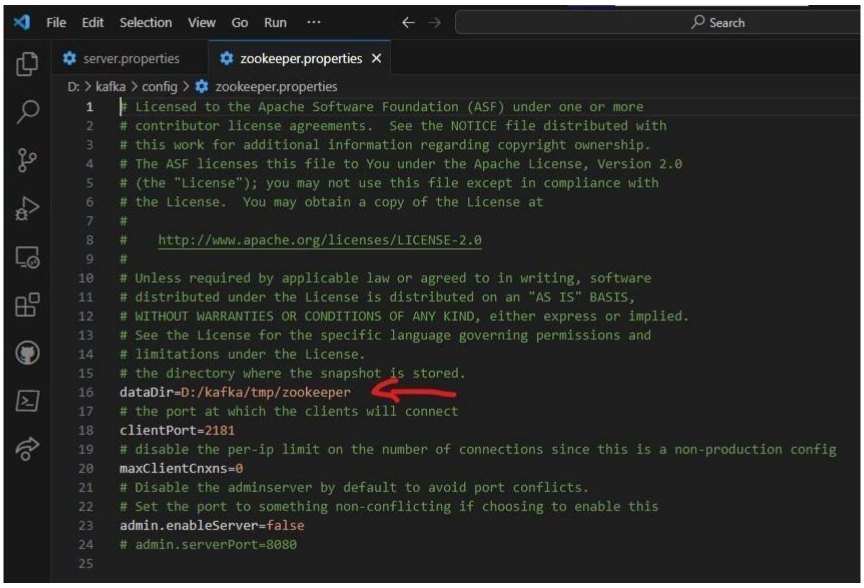
Now go to kafka>config and open server.properties file , we have to change log directory path to save log on specific location.



```
# The send buffer (SO_SNDBUF) used by the socket server
socket.send.buffer.bytes=102400

# The receive buffer (SO_RCVBUF) used by the socket server
socket.receive.buffer.bytes=102400

# The maximum size of a request that the socket server will accept (protection against OOM)
socket.request.max.bytes=104857600


############################# Log Basics #############################

# A comma separated list of directories under which to store log files
log.dirs=D:/kafka/tmp/kafka-logs

# The default number of log partitions per topic. More partitions allow greater
# parallelism for consumption, but this will also result in more files across
# the brokers.
num.partitions=1

# The number of threads per data directory to be used for log recovery at startup and flushing at shutdown.
# This value is recommended to be increased for installations with data dirs located in RAID array.
num.recovery.threads.per.data.dir=1

############################# Internal Topic Settings  #############################
# The replication factor for the group metadata internal topics "__consumer_offsets" and "__transaction_state"
# For anything other than development testing, a value greater than 1 is recommended to ensure availability such as 3.
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1

############################# Log Flush Policy #############################
```

As shown in above image set log.dirs path to your specific location.

Now for the same open zookeeper.properties and edit dataDir path to specific location as shown below image.

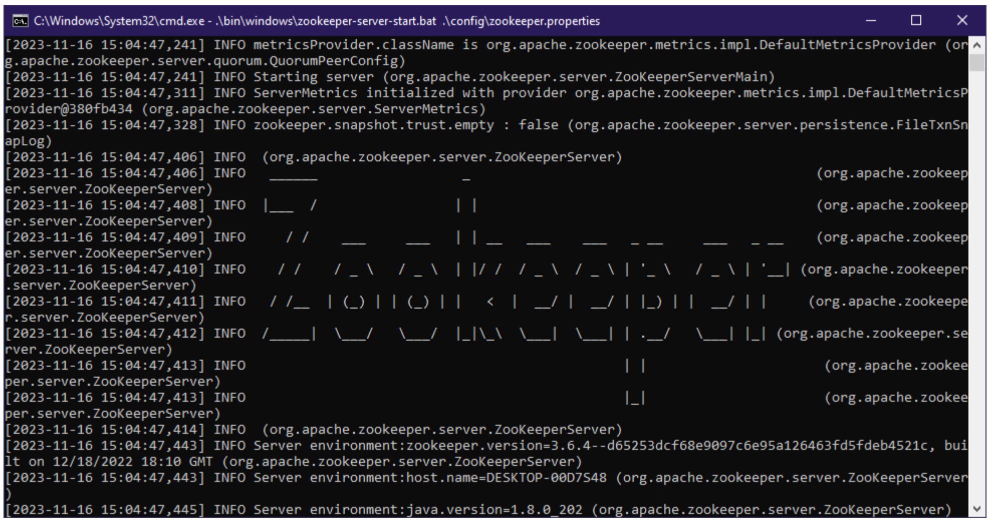


```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# the directory where the snapshot is stored.
dataDir=D:/kafka/tmp/zookeeper
# the port at which the clients will connect
clientPort=2181
# disable the per-ip limit on the number of connections since this is a non-production config
maxClientCnxns=0
# Disable the adminserver by default to avoid port conflicts.
# Set the port to something non-conflicting if choosing to enable this
admin.enableServer=false
# admin.serverPort=8080
```

❖ Run kafka

To run kafka server first have to start zookeeper server. Now locate to kafka folder where you have bin folder and then open cmd at this place and run following command :

```
.\bin\windows\zookeeper-server-start.bat   .\config\zookeeper.properties
```



```
C:\Windows\System32\cmd.exe - .\bin\windows\zookeeper-server-start.bat  .\config\zookeeper.properties
[2023-11-16 15:04:47,241] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (or
g.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2023-11-16 15:04:47,241] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2023-11-16 15:04:47,311] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsP
rovider@380fb434 (org.apache.zookeeper.server.ServerMetrics)
[2023-11-16 15:04:47,328] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSn
apLog)
[2023-11-16 15:04:47,406] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-11-16 15:04:47,406] INFO   ______                  _                                          (org.apache.zookeep
er.server.ZooKeeperServer)
[2023-11-16 15:04:47,408] INFO  |___  /                 | |                                         (org.apache.zookeep
er.server.ZooKeeperServer)
[2023-11-16 15:04:47,409] INFO     / /    ___     ___   | | __   ___    ___   _ __     ___   _ __   (org.apache.zookeep
er.server.ZooKeeperServer)
[2023-11-16 15:04:47,410] INFO    / /    / _ \   / _ \  | |/ /  / _ \  / _ \ | '_ \   / _ \ | '__| (org.apache.zookeeper
.server.ZooKeeperServer)
[2023-11-16 15:04:47,411] INFO   / /__  | (_) | | (_) | |   <  |  __/ |  __/ | |_) | |  __/ | |     (org.apache.zookeepe
r.server.ZooKeeperServer)
[2023-11-16 15:04:47,412] INFO  /_____|  \___/   \___/  |_|\_\  \___|  \___| | .__/   \___| |_| (org.apache.zookeeper.se
rver.ZooKeeperServer)
[2023-11-16 15:04:47,413] INFO                                               | |                     (org.apache.zookee
per.server.ZooKeeperServer)
[2023-11-16 15:04:47,413] INFO                                               |_|                     (org.apache.zookee
per.server.ZooKeeperServer)
[2023-11-16 15:04:47,414] INFO  (org.apache.zookeeper.server.ZooKeeperServer)
[2023-11-16 15:04:47,443] INFO Server environment:zookeeper.version=3.6.4--d65253dcf68e9097c6e95a126463fd5fdeb4521c, bui
lt on 12/18/2022 18:10 GMT (org.apache.zookeeper.server.ZooKeeperServer)
[2023-11-16 15:04:47,443] INFO Server environment:host.name=DESKTOP-00D7S48 (org.apache.zookeeper.server.ZooKeeperServer
)
[2023-11-16 15:04:47,445] INFO Server environment:java.version=1.8.0_202 (org.apache.zookeeper.server.ZooKeeperServer)
```

If you seen this zookeeper than server started successfully.

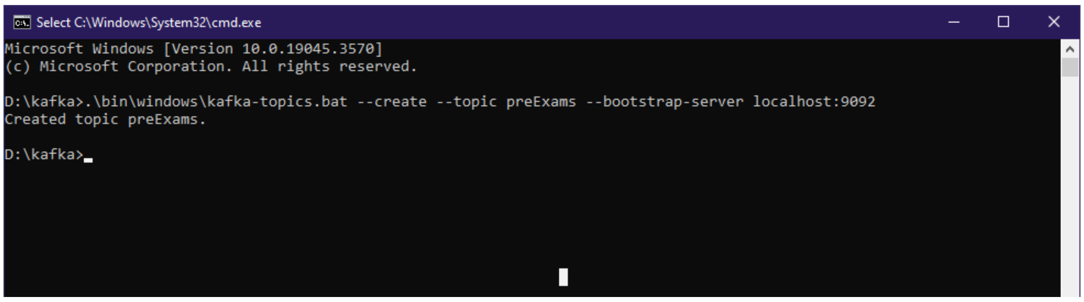
Now we have to start kafka server , so open new terminal at same place and run following command :

```
.\bin\windows\kafka-server-start.bat      .\config\server.properties
```

Kafka works on producer and consumer model so first we have to create one topic , open new terminal at same place and create a topic by following command :

```
.\bin\windows\kafka-topics.bat --create --topic {topic name} --bootstrap-
server localhost:9092
```

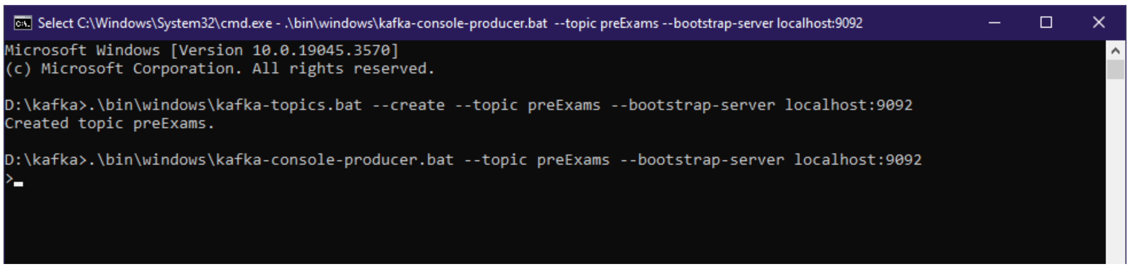
Write topic name of your choice don't take space or any special character in topic name it consider only one word of name.



```
Select C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

D:\kafka>.\bin\windows\kafka-topics.bat --create --topic preExams --bootstrap-server localhost:9092
Created topic preExams.

D:\kafka>_
```

After successfully creating a topic , lets start producer by this command , give same topic name that created :

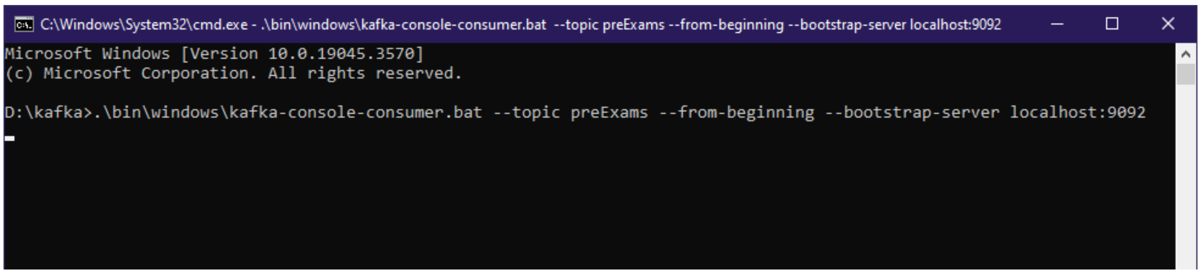
```
.\bin\windows\kafka-console-producer.bat --topic {topic name} --bootstrap-
server localhost:9092
```


Select C:\Windows\System32\cmd.exe - .\bin\windows\kafka-console-producer.bat --topic preExams --bootstrap-server localhost:9092

```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

D:\kafka>.\bin\windows\kafka-topics.bat --create --topic preExams --bootstrap-server localhost:9092
Created topic preExams.

D:\kafka>.\bin\windows\kafka-console-producer.bat --topic preExams --bootstrap-server localhost:9092
>
```

For the consumer open new terminal and type following command , give same topic name which have given in producer :

.\bin\windows\kafka-console-consumer.bat --topic {topic name} --from-beginning --bootstrap-server localhost:9092


C:\Windows\System32\cmd.exe - .\bin\windows\kafka-console-consumer.bat --topic preExams --from-beginning --bootstrap-server localhost:9092

```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

D:\kafka>.\bin\windows\kafka-console-consumer.bat --topic preExams --from-beginning --bootstrap-server localhost:9092
```

Now set producer and consumer side by side , type something in producer and hit enter after some millisecond you can see same thing in consumer terminal.