Pandit Deendayal Energy University

LABORATORY MANUAL

Branch: **Computer Engineering**

Semester: **VI**

20CP305P– Big Data Analytics Laboratory

Roll No: 19BCP080

Name of Student: Mire Kishorkumar Patel

Pandit Deendayal Energy University

20CP305P Big Data Analytics Laboratory

COURSE OBJECTIVES

- Identify the challenges of Big Data Management
- Recognize the key concepts of Hadoop framework, MapReduce and SPARK.
- Apply the tools, techniques and algorithms for big data analysis.

LIST OF EXPERIMENTS:

1. Introduction/Basics of Scala Programming
2. Introduction/Basics of Python Programming
3. Transformation functions
4. Pair RDD functions
5. Spark SQL, creating tables and querying in data bricks
6. PIG and Hive Demo
7. Structured Streaming using the Python Data Frames API
8. Page Rank
9. Machine Learning
10. GraphX
11. Kafka configuration and demo codes
12. MongoDB configuration and demo codes
13. Project work (Code + Paper/Report + PPT)

Demo of each is provided followed by practice to the students

TOTAL: 14 week x 2 Sessions (Each session of 2hrs. per week)

OUTCOMES:

COURSE OUTCOMES

On completion of the course, student will be able to
CO1- Understand Hadoop related tools for big data analytics
CO2- Deploy Hadoop ecosystem components
CO3- Demonstrate basic Hadoop administration.
CO4- Apply Map Reduce paradigm for Big Data Analysis.
CO5- Understand the working of tools (SPARK) and techniques to analyze Big Data
CO5- Build a solution for a given problem using suitable Big Data Techniques

Materials from national and international level like NPTEL, Web resources, etc. is shared related to subject domain

Apache Hadoop : hadoop.apache.org
Spark by Examples: https://sparkbyexamples.com/spark/
Hive Tutorial – Tutorialspoint : www.tutorialspoint.com › hive
Apache Pig Tutorial – Tutorialspoint : www.tutorialspoint.com › apache_pig
Apache Spark Tutorial – Tutorialspoint : www.tutorialspoint.com › apache_spark
https://www.udemy.com/course/big-data-harish/learn/lecture/14046353#overview

LIST OF HARDWARE REQUIREMENTS & SOFTWARE REQUIREMENTS

SOFTWARE REQUIREMENTS

- Java, Python
- Data Bricks community cloud setup

HARDWARE REQUIREMENTS

- Standalone desktops (or) Server supporting 30 terminals or more

INDEX

LAB 1: Basics of Scala Programming

AIM:

To know about Scala programming.

PROGRAM:

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

Output:

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

```
num.head
```

Output:

```
res0: Int = 1
```

```
num.tail
```

Output:

```
res1: List[Int] = List(2, 3, 4)
```

```
num.sum
```

Output:

```
res2: Int = 10
```

```
num.take(3)
```

Output:

```
res3: List[Int] = List(1, 2, 3)
```

```
var mirepatel = List(1,1,1,12,2,2,2,2,2)
```

Output:

```
mirepatel: List[Int] = List(1, 1, 1, 12, 2, 2, 2, 2, 2)
```

```
mirepatel.distinct
```

Output:

```
res4: List[Int] = List(1, 12, 2)
```

```
mirepatel(5)
```

Output:

```
res5: Int = 2
```

```
mirepatel(-1)
```

Output:

```
IndexOutOfBoundsException: -1
```

```
mirepatel(4)=4
```

Output:

```
command-1200946268936804:1: error: value update is not a
member of List[Int]
```

```
num.size
```

Output:

```
res9: Int = 4
```

```
num.reverse
```

Output:

```
res10: List[Int] = List(4, 3, 2, 1)
```

```
mirepatel.min
```

Output:

```
res11: Int = 1
```

```
mirepatel.max
```

Output:

```
res12: Int = 12
```

```
mirepatel.isEmpty
```

Output:

```
res13: Boolean = false
```

```
var num = Array(1,2,3,4,5,6,7,8,9)
```

Output:

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

```
val lang = Array("scala", "python", "R")
```

Output:

```
lang: Array[String] = Array(scala, python, R)
```

```
lang.head
```

Output:

```
res14: String = scala
```

```
lang.tail
```

Output:

```
res15: Array[String] = Array(python, R)
```

```
num(3) = 30

num
```

Output:

```
res17: Array[Int] = Array(1, 2, 3, 30, 5, 6, 7, 8, 9)
```

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

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

Output:

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

```
car.append("car1")
car.append("car2")
car.append("car1")


car += "car3"
```

Output:

```
res24: scala.collection.mutable.ArrayBuffer[String] =
ArrayBuffer(car1, car2, car1, car3)
```

```
car.length
```

Output:

```
res25: Int = 4
```

```
car
```

Output:

```
res26: scala.collection.mutable.ArrayBuffer[String] =
ArrayBuffer(car1, car2, car1, car3)
```

```
car.trimEnd(1)

car
```

Output:

```
res28: scala.collection.mutable.ArrayBuffer[String] =
ArrayBuffer(car1, car2, car1)
```

```
car.insert(3, "BMW")

car
```

Output:

```
res30: scala.collection.mutable.ArrayBuffer[String] =
ArrayBuffer(car1, car2, car1, BMW)
```

```
num
```

Output:

```
res31: Array[Int] = Array(1, 2, 3, 30, 5, 6, 7, 8, 9)
```

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

Output:

```
res32: Array[Int] = Array(1, 4, 9, 900, 25, 36, 49, 64,
81)
```

```
num
```

Output:

```
res33: Array[Int] = Array(1, 2, 3, 30, 5, 6, 7, 8, 9)
```

```
num.map(a=>a+3)
```

Output:

```
res34: Array[Int] = Array(4, 5, 6, 33, 8, 9, 10, 11, 12)
```

```
val a = num.map(aa => aa*(aa-1))
```

Output:

```
a: Array[Int] = Array(0, 2, 6, 870, 20, 30, 42, 56, 72)
```

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

Output:

```
b: Array[Int] = Array(4, 9, 16, 961, 36, 49, 64, 81,
100)
```

```
val fruits = List("orange", "banana", "apple", "pineapple")
```

Output:

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

```
fruits.map(x => (x, x.length))
```

Output:

```
res35: List[(String, Int)] = List((orange,6),
(banana,6), (apple,5), (pineapple,9))
```

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

Output:

```
res36: List[String] = List(orange, banana, pineapple)
```

```
var ratings = List(2.4, 5.6, 7.4, 8.9)
```

Output:

```
ratings: List[Double] = List(2.4, 5.6, 7.4, 8.9)
```

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

Output:

```
marks: List[Double] = List(24.0, 56.0, 74.0, 89.0)
```

```
val grade_B = marks.filter(x => x > 60 && x < 80)
```

Output:

```
grade_B: List[Double] = List(74.0)
```

```
grade_B.map(x => x/10)
```

Output:

```
res37: List[Double] = List(7.4)
```

```
def add(a: Double = 100, b: Double = 200): Double = {
  var sum: Double = 0
  sum = a + b
  return sum
}
add: (a: Double, b: Double)Double
```

```
add()
```

Output:

```
res38: Double = 300.0
```

```
var x =1
var b = if (x < 3) {
  println("less then 3")
} else {
  println("greather then 3")
}
less then 3
x: Int = 1
b: Unit = ()

var marks = 75

if (marks > 70) {
  print("A")
} else if (marks > 50 && marks < 70) {
  print("B")
} else if (marks>40) {
  print("c")
} else {
  print("F")
}
```

Output:

```
Amarks: Int = 75
```

```
def squ(x: Double) : Double = {
  return x*x
}

def sqqu(x: Double, y: Double) : Double = {
  return squ(x) + squ(y)
}

sqqu(3, 4)
```

Output:

```
squ: (x: Double)Double
sqqu: (x: Double, y: Double)Double
res40: Double = 25.0
```

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

Output:

```
1
2
```

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

```
// matrix multiplication
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)
```

Output:

```
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
```

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

Output:

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

```
a(1)
```

Output:

```
res45: java.io.Serializable = List(1, 2, 3)
```

```
// error
a(1)(1)
```

Output:

```
command-1200946268936859:1: error: java.io.Serializable
does not take parameters
```

LAB 2: Basics of Python Programming

AIM:

To know about python programming.

PROGRAM:

```
myint = 7
print(myint)
```

Output:

```
7
```

```
myfloat=7.2
print(myfloat)
```

Output:

```
7.2
```

```
mystring='hello'
print(mystring)
```

Output:

```
hello
```

```
mystring="hello"
print(mystring)
```

Output:

```
hello
```

```
one=1
two=2
three = one + two
print(three)
hello ="Hello"
world="India"
helloworld = hello + " " + world
print(helloworld)
```

Output:

```
Hello India
```

```
a,b = 3,4
print(a,b)
```

Output:

```
3 4
```

```
one =1
two =2
hello ="hello"
print(one + two + hello)
```

Output:

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
mylist=[1,2,3]
print(mylist[10])
```

Output:

```
IndexError: list index out of range
```

```
mylist.append(1)
mylist.append(2)
mylist.append(3)
print(mylist[0])
print(mylist[1])
print(mylist[2])
```

Output:

```
1 2 3
```

```
for x in mylist:
    print(x)
```

Output:

```
1 2 3 1 2 3
```

```
number = 10+2*3/4.0
print(number)
```

Output:

```
11.5
```

```
remainder = 10 % 3
print(remainder)
```

Output:

```
1
```

```
squared = 7**2
```

```
cubed = 2**3
print(squared, cubed)
```

Output:

```
49 8
```

```
lotsofhellos = "hello World"*15
print(lotsofhellos)
```

Output:

```
hello Worldhello Worldhello Worldhello Worldhello Worldhello
Worldhello Worldhello Worldhello Worldhello Worldhello
Worldhello Worldhello Worldhello Worldhello World
```

```
print([1,2,3]*5)
```

Output:

```
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

```
even_nos=[2,4,6,8]
odd_nos=[1,3,5,7]
all_nos= odd_nos + even_nos + odd_nos + even_nos
print(all_nos)
```

Output:

```
[1, 3, 5, 7, 2, 4, 6, 8, 1, 3, 5, 7, 2, 4, 6, 8]
```

```
para_str=""" hello \n how are you, \n this the first
laboratory of BDA, \n welcome to this BDA practical session
\n In the first session we will see basics of python, which
is prerequisite"""
print(para_str)
```

Output:

```
hello how are you, this the first laboratory of BDA, welcome
to this BDA practical session In the first session we will
see basics of python, which is prerequisite
```

```
n = “hellllllllllllllo woooooooooorllllllllllld”
new=’’
for I in n:
    if not(I in new):
        new=new + i
print(new)
```

Output:

```
h he hel helo helo helo w helo wr helo wrd
```

```
string ="hellllllllllllllo wooooooooooorllllllllllld hello
world"
i=0
while (i <len(string)):
    string = string[:i+1] +
string[i+1:].replace(string[i],'')
    i = i+1
print(string)
```

Output:

```
hellllllllllllllo wooooooooooorllllllllllld ello world
hellllllllllllllo wooooooooooorllllllllllld llo world helo
wooooooooooord o word helo wrd wrd helo wrdwrd helo wrdrd
helo wrdd helo wrd
```

```
num = 12
if(num<10):
    print("1")
elif(num <100):
    print("2")
elif(num <1000):
    print("3")
else:
    print("invalid")
```

Output:

```
2
```

```
num = 123
while(num >0):
    print("you entered" +str(num))
    num = num/10
    if (num <0):
        print("Number is -ve no")
    break
```

Output:

```
you entered123
```

```
name ="mirepatel"
print("Hello, %s!!!!!!!" %name)
```

Output:

```
Hello, mirepatel!!!!!!!
```

```
age = 25
print("%s is %d years old" %(name,age))
```

Output:

```
mirepatel is 25 years old
```

```
mylist = [1,2,3,4,5]
print("A list : %s" %mylist)
```

Output:

```
A list : [1, 2, 3, 4, 5]
```

```
astring ="length f Hello world!"
astring2 = 'Hello world!'
print("singel quotes are ' ")
```

Output:

```
singel quotes are '
```

```
print(len(astring))
```

Output:

```
21
```

```
print(astring.index("o"))
```

Output:

```
13
```

```
print(astring.count("l"))
```

Output:

```
4
```

```
print(astring[3:7])
```

Output:

```
gth
```

```
print(astring[3:7:2])
```

Output:

```
gh
```

```
print(astring[::-1])
```

Output:

```
!dlrow olleH f htgnel
```

```
print(astring.upper())
```

Output:

```
LENGTH F HELLO WORLD!
```

```
print(astring.lower())
```

Output:

```
length f hello world!
```

```
print(astring.startswith("Hello"))
```

Output:

```
False
```

```
print(astring.endswith("world!"))
```

Output:

```
True
```

```
afewwords= astring.split(" ")
print(afewwords)
```

Output:

```
['length', 'f', 'Hello', 'world!']
```

```
x=[1,2,3, 4]
y=[1,2,3,4]
print(x == y)
```

Output:

```
True
```

```
print(x is y)
```

Output:

```
False
```

```
print(not True)
```

Output:

```
False
```

```
print((not True) == (False))
```

Output:

```
True
```

```
for x in range(15):
    print(x)
```

Output:

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

```
for x in range(3,6):
    print(x)
```

Output:

```
3 4 5
```

```
for x in range(3,15,2):
    print(x)
```

Output:

```
3 5 7 9 11 13
```

```
for x in range(1,10):
    if(i % 5 == 0):
         break
    print(i)
```

Output:

```
8 8 8 8
```

```
def sum(a,b):
    return a+b

x=sum(300,33)
print(x)
```

Output:

```
333
```

```
tup1 =('physics','chemistry', 1997,2000)
tup2=(1,2,3,4,5,6,7,8,9,10)
print("tup1[3]:", tup1[2])
print("tup2[1:5]:",tup2[0:5])
```

Output:

```
tup1[3]: 1997
tup2[1:5]: (1, 2, 3, 4, 5)
```

```
tup1[0]=100
```

Output:

```
TypeError: 'tuple' object does not support item assignment
```

```
tup3 = tup1 + tup2
print(tup3)
```

Output:

```
('physics', 'chemistry', 1997, 2000, 1, 2, 3, 4, 5, 6, 7, 8,
9, 10)
```

```
del tup3

print(tup3)
```

Output:

```
NameError: name 'tup3' is not defined
```

```
len((1,2,3))
```

Output:

```
3
```

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

Output:

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

```
('Hi!,')*4
```

Output:

```
Hi!,Hi!,Hi!,Hi!,
```

```
4 in (1,2,3)
```

Output:

```
False
```

```
for x in (1,2,3,4,5):
    print(x, end = ' ')
```

Output:

```
1 2 3 4 5
```

```
dict ={'Name':'Samir','Age':7,'Class':'First'}
print("dict['Name']:", dict['Name'])
```

Output:

```
dict['Name']: Samir
```

```
dict['Age']=18
dict['School']="DPS Hi School";
```

```
print(dict)
```

Output:

```
{'Name': 'Samir', 'Age': 18, 'Class': 'First', 'School': 'DPS
Hi School'}
```

```
del dict['Name'];
dict.clear();
del dict;

dict ={'Name':'zara','Age':7,'Name':'Manni'}
print("dict['Name']:", dict['Name'])
```

Output:

```
dict['Name']: Manni
```

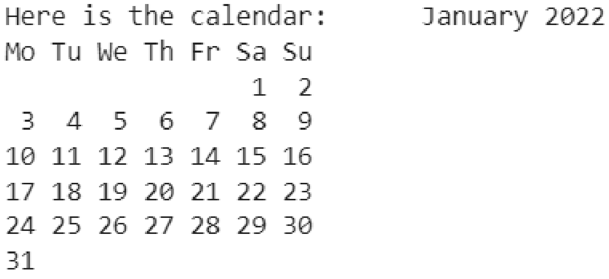
```
import time
ticks = time.time()

localtime = time.asctime(time.localtime(time.time()))

print("Local current time: ", localtime)
```

Output:

```
Local current time:  Sun May  1 08:54:29 2022
```

```
import calendar
cal = calendar.month(2022,1)
print("Here is the calendar: ", cal)
```

Output:



```
Here is the calendar:      January 2022
Mo Tu We Th Fr Sa Su
                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
```

```
2+3
```

Output:

```
5
```

LAB 3: Transformation functions

AIM:

To implement all transformation functions in scala/python using DataBricks cloud platform.

DESCRIPTION:

RDD Transformations are Spark operations when executed on RDD, it results in a single or multiple new RDD's. Since RDD are immutable in nature, transformations always create new RDD without updating an existing one hence, this creates an **RDD lineage**.

EXAMPLE:

```
val a = sc.parallelize(List("A", "B", "C", "D"))
```

Output:

```
a: org.apache.spark.rdd.RDD[String] =
ParallelCollectionRDD[26] at parallelize at command-
577761670161294:1
```

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

Output:

```
b: org.apache.spark.rdd.RDD[(String, Int)] =
MapPartitionsRDD[27] at map at command-577761670161304:1
res0: Array[(String, Int)] = Array((A,1), (B,1), (C,1),
(D,1))
```

```
val b = a.map((_,1))
b.collect
```

Output:

```
b: org.apache.spark.rdd.RDD[(String, Int)] =
MapPartitionsRDD[28] at map at command-577761670161303:1
res1: Array[(String, Int)] = Array((A,1), (B,1), (C,1),
(D,1))
```

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

Output:

```
b: org.apache.spark.rdd.RDD[(String, Int)] =
MapPartitionsRDD[29] at map at command-577761670161302:1
res2: Array[(String, Int)] = Array((A,1), (B,1), (C,1),
(D,1))
```

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

Output:

```
a: org.apache.spark.rdd.RDD[List[Int]] =
MapPartitionsRDD[31] at map at command-577761670161301:1
res3: 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))
a.collect
```

Output:

```
a: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[33]
at flatMap at command-577761670161300:1
res4: Array[Int] = Array(1, 1, 2, 2, 3, 3, 4, 4, 5, 5)
```

```
val rdda = sc.parallelize(List("aaaa", "bbbb", "ccc"))
rdda.filter(_.equals("aaaa")).collect
```

Output:

```
rdda: org.apache.spark.rdd.RDD[String] =
ParallelCollectionRDD[34] at parallelize at command-
577761670161299:1
res5: Array[String] = Array(aaaa)

rdda.filter(_.contains("a")).collect
res6: Array[String] = Array(aaaa)
```

```
val a = sc.parallelize(List(("Mumbai", 4000), ("Delhi",
2000), ("Chennai", 1000), ("Kolkatta", 7000)))
```

Output:

```
a: org.apache.spark.rdd.RDD[(String, Int)] =
ParallelCollectionRDD[37] at parallelize at command-
577761670161297:1
```

```
a.filter(_._2.equals(4000)).collect
```

Output:

```
res7: Array[(String, Int)] = Array((Mumbai,4000))
```

```
a.filter(_._2>3000).collect
```

Output:

```
res8: Array[(String, Int)] = Array((Mumbai,4000),
(Kolkatta,7000))
```

```
a.filter(_._2>3000).filter(_._2<6000).collect
```

Output:

```
res9: Array[(String, Int)] = Array((Mumbai,4000))
```

```
// sample(true/false, , )
// true have repieation
// false
// fraction 0 to 1no. of sample in o/p
// seed result same if same

val a = sc.parallelize(1 to 1000)
```

Output:

```
a: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[42] at parallelize at command-
577761670161311:1
```

```
a.sample(false, 0.2, 24).collect
```

Output:

```
res12: Array[Int] = Array(7, 10, 14, 16, 17, 19, 21, 27,
29, 32, 35, 46, 48, 54, 61, 63, 77, 80, 83, 84, 85, 86,
87, 88, 92, 100, 111, 114, 115, 116, 127, 144, 149, 151,
161, 169, 172, 191, 199, 204, 207, 217, 225, 236, 243,
250, 252, 254, 259, 262, 269, 272, 274, 275, 283, 285,
289, 299, 300, 307, 309, 322, 327, 332, 333, 342, 343,
345, 349, 364, 371, 386, 393, 397, 405, 406, 407, 411,
413, 415, 417, 419, 420, 425, 439, 442, 451, 453, 454,
456, 464, 466, 467, 470, 473, 477, 485, 490, 495, 498,
511, 517, 518, 519, 521, 525, 532, 542, 546, 549, 551,
552, 560, 564, 576, 578, 585, 590, 608, 610, 615, 617,
620, 626, 627, 646, 648, 656, 658, 659, 660, 663, 667,
669, 672, 683, 684, 687, 699, 711, 713, 721, 725, 726,
729, 732, 735, 736, 743, 747, 754, 756, 765, 767, 768,
776, 777, 778, 780, 798, 800, 804, 810, 814, 817, 820,
821, 822, 825, 827, 833, 843, 867, 874, 890, 892, 898,
900, 903, 908, 914, 920, 922, 924, 925, 931, 932, 933,
943, 945, 960, 965, 967, 972, 975, 978, 985, 987, 994)
```

```
a.sample(false, 0.1, 1022).collect
```

Output:

```
res13: Array[Int] = Array(21, 23, 44, 48, 55, 59, 66,
67, 71, 82, 106, 113, 115, 120, 123, 128, 135, 153, 162,
172, 214, 247, 249, 255, 269, 287, 322, 326, 330, 355,
359, 385, 388, 399, 408, 429, 432, 438, 465, 478, 484,
487, 490, 503, 510, 514, 541, 543, 559, 564, 566, 571,
572, 586, 588, 597, 635, 649, 662, 672, 682, 694, 737,
746, 754, 777, 784, 805, 808, 822, 823, 836, 838, 841,
854, 857, 861, 870, 874, 894, 908, 910, 947, 948, 955,
959, 968, 978, 979, 995)
```

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

Output:

```
a: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[45] at parallelize at command-
577761670161308:1
```

```
a.sample(true, 0.5, 15).collect
```

Output:

```
res14: Array[Int] = Array(1, 2, 2)
```

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

Output:

```
a: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[47] at parallelize at command-
577761670161306:1
```

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

Output:

```
b: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[48] at parallelize at command-
577761670161305:1
```

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

Output:

```
res15: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 5, 6, 7,
8, 9, 10)
```

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

Output:

```
res16: Array[Int] = Array(5, 6, 7)
```

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

Output:

```
res17: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

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

Output:

```
a: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[60] at parallelize at command-
577761670161326:1
```

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

Output:

```
res18: Array[Int] = Array(1, 4, 7)
```

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

Output:

```
a: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[62] at parallelize at command-
577761670161324:1
```

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

Output:

```
res19: Array[Int] = Array(1, 3, 5, 7)
```

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

Output:

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

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

Output:

```
a: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[64] at parallelize at command-
577761670161323:1
```

```
a.mapPartitionsWithIndex(pra).collect
```

Output:

```
res20: Array[String] = Array(1 0, 2 0, 3 0, 4 1, 5 1, 6
1)
```

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

Output:

```
a: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[66] at parallelize at command-
577761670161321:1
```

```
a.mapPartitionsWithIndex(pra).collect
```

Output:

```
res21: Array[String] = Array(1 0, 2 0, 3 1, 4 1, 5 2, 6
2)
```

```
def isPrime(num:Int):Boolean =
(num > 1) && !(2 to scala.math.sqrt(num).toInt).exists(x =>
num % x == 0)
isPrime: (num: Int)Boolean

val a = sc.parallelize(1 to 10)
val even = a.filter(x => (x % 2 == 0)).map(x => x*x).sum
val odd = a.filter(x => (x % 2 != 0)).map(x => x*x).sum
val prime = a.filter(isPrime).map(x => x*x).sum

println("Sum of odd is "+odd)
println("Sum of even is "+even)
println("Sum of prime is "+prime)
Sum of odd is 165.0
Sum of even is 220.0
Sum of prime is 87.0
```

Output:

```
a: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[68] at parallelize at command-
577761670161319:1
even: Double = 220.0
odd: Double = 165.0
prime: Double = 87.0
```

```
a.filter(i => (i%2==0)).sum
```

Output:

```
res23: Double = 30.0
```

```
def isPrime(num:Int):Boolean =
(num > 1) && !(2 to scala.math.sqrt(num).toInt).exists(x =>
num % x == 0)
```

Output:

```
isPrime: (num: Int)Boolean
```

```
val b = a.foreach(isPrime)
```

Output:

```
b: Unit = ()
```

```
a.filter(isPrime).collect
```

Output:

```
res24: Array[Int] = Array(2, 3, 5, 7)
```

```
val accum = sc.longAccumulator("Sum")
sc.parallelize(Array(1,2,3)).foreach(x=>accum.add(x))
```

Output:

```
accum: org.apache.spark.util.LongAccumulator =
LongAccumulator(id: 1459, name: Some(Sum), value: 6)
```

```
accum.value
```

Output:

```
res26: Long = 6
```

LAB 4: Pair RDD functions

AIM:

To know about RDD functions in spark using Scala.

DESCRIPTION:

Spark defines PairRDDFunctions class with several functions to work with Pair RDD or RDD key-value pair, In this tutorial, we will learn these functions with Scala examples. Pair RDD's are come in handy when you need to apply transformations like hash partition, set operations, joins etc.

All these functions are grouped into Transformations and Actions similar to regular RDD's.

PROGRAM:

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

Output:

```
spark: org.apache.spark.sql.SparkSession =
org.apache.spark.sql.SparkSession@34c210c8
```

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

Output:

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

```
val contries = Map(("USA", "America"), ("IN", "India"))
```

Output:

```
contries: scala.collection.immutable.Map[String,String]
= Map(USA -> America, IN -> India)
```

```
val brodState = spark.sparkContext.broadcast(state)
```

Output:

```
brodState:
org.apache.spark.broadcast.Broadcast[scala.collection.im
mutable.Map[String,String]] = Broadcast(46)
```

```
val brodContries = spark.sparkContext.broadcast(contries)
```

Output:

```
brodContries:
org.apache.spark.broadcast.Broadcast[scala.collection.im
mutable.Map[String,String]] = Broadcast(47)
```

```
val data = Seq(("Mirepatel", "Patel", "IN", "CA"),
               ("Smirepatel", "Patel", "USA", "CA"),
               ("Mire", "Patel", "IN", "NY"),
               ("Parth", "Patel", "USA", "FL"))
```

Output:

```
data: Seq[(String, String, String, String)] =
List((Mirepatel,Patel,IN,CA), (Smirepatel,Patel,USA,CA),
(Mire,Patel,IN,NY), (Parth,Patel,USA,FL))
```

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

Output:

```
rdd: org.apache.spark.rdd.RDD[(String, String, String,
String)] = ParallelCollectionRDD[82] at parallelize at
command-2853681421999185:1
```

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

Output:

```
rdd2: org.apache.spark.rdd.RDD[(String, String, String,
String)] = MapPartitionsRDD[84] at map at command-
2853681421999184:1
```

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

Output:

```
(Mirepatel,Patel,India,California)
(Smirepatel,Patel,America,California)
(Mire,Patel,India,New York)
(Parth,Patel,America,Florida)
```

```
val state = Map(("NY", "New York"), ("CA", "California"),
("FL", "Florida"))
val contries = Map(("USA", "America"), ("IN", "India"))
val brodState = spark.sparkContext.broadcast(state)
val brodContries = spark.sparkContext.broadcast(contries)

val data = Seq(("Mirepatel", "Patel", "IN", "CA"),
              ("Smirepatel", "Patel", "USA", "CA"),
              ("Mire", "Patel", "IN", "NY"),
              ("Parth", "Patel", "USA", "FL"))
val columns = Seq("firstname", "lastname", "Country",
"State")

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 fullState = brodState.value.get(state).get
  val fullCountry = brodContries.value.get(country).get
  (row.getString(0), row.getString(1), fullCountry,
fullState)
}).toDF(columns:_*)
```

Output:

```
state: scala.collection.immutable.Map[String,String] =
Map(NY -> New York, CA -> California, FL -> Florida)
contries: scala.collection.immutable.Map[String,String]
= Map(USA -> America, IN -> India)
brodState:
org.apache.spark.broadcast.Broadcast[scala.collection.im
mutable.Map[String,String]] = Broadcast(49)
brodContries:
org.apache.spark.broadcast.Broadcast[scala.collection.im
mutable.Map[String,String]] = Broadcast(50)
data: Seq[(String, String, String, String)] =
List((Mirepatel,Patel,IN,CA), (Smirepatel,Patel,USA,CA),
(Mire,Patel,IN,NY), (Parth,Patel,USA,FL))
columns: Seq[String] = List(firstname, lastname,
Country, State)
import spark.sqlContext.implicits._
df: org.apache.spark.sql.DataFrame = [firstname: string,
lastname: string ... 2 more fields]
df2: org.apache.spark.sql.DataFrame = [firstname:
string, lastname: string ... 2 more fields]
```

```
df2.show(4)
```

Output:

```
+----------+--------+-------+----------+
| firstname|lastname|Country|     State|
+----------+--------+-------+----------+
| Mirepatel|   Patel|  India|California|
|Smirepatel|   Patel|America|California|
|      Mire|   Patel|  India|  New York|
|     Parth|   Patel|America|   Florida|
+----------+--------+-------+----------+
```

```
val longAcc = spark.sparkContext.longAccumulator("SUM")
```

Output:

```
longAcc: org.apache.spark.util.LongAccumulator = 
LongAccumulator(id: 1620, name: Some(SUM), value: 0)
```

```
val rdd = spark.sparkContext.parallelize(Array(1,2,3,4,5))
```

Output:

```
rdd: org.apache.spark.rdd.RDD[Int] = 
ParallelCollectionRDD[87] at parallelize at command-
2853681421999191:1
```

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

rdd.collect
```

Output:

```
res31: Array[Int] = Array(1, 2, 3, 4, 5)
```

```
longAcc.value
```

Output:

```
res32: Long = 15
```

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

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

Output:

```
inputRDD: org.apache.spark.rdd.RDD[(String, Int)] = 
ParallelCollectionRDD[88] at parallelize at command-
2853681421999197:1
```

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

Output:

```
listRDD: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[89] at parallelize at command-
2853681421999196:1
```

```
def param0 = (acc:Int, v:Int) => acc + v
def param1 = (acc1:Int, acc2:Int) => acc1 + acc2
println("Aggregate : " + listRDD.aggregate(0) (param0,
param1))
```

Output:

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

```
def param3 = (acc:Int, v:(String, Int)) => acc + v._2
def param2 = (acc1:Int, v2:Int) => acc1 + v2

println("Aggregate : " + inputRDD.aggregate(0) (param3,
param2))
Aggregate : 181
```

Output:

```
param3: (Int, (String, Int)) => Int
param2: (Int, Int) => Int
```

```
def param4 = (acc:Int, v:Int) => acc + v
def param5 = (acc1:Int, v2:Int) => acc1 + v2

println("Tree Aggregate : " + listRDD.treeAggregate(0)
(param4, param5))

Tree Aggregate : 20
```

Output:

```
param4: (Int, Int) => Int
param5: (Int, Int) => Int
```

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

Output:

```
Fold : 20
```

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

Output:

```
Fold : (Total,181)
```

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

Output:

```
data: Array[(String, Int)] = Array((Z,1), (B,30),
(A,20), (B,30), (C,40), (B,60))
```

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

Output:

```
(Z,1)
(B,30)
(A,20)
(B,30)
(C,40)
(B,60)
```

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

Output:

```
Reduce : 20
```

```
println("Reduce alternate : " + listRDD.reduce((x, y) =>
x+y))
```

Output:

```
Reduce alternate : 20
```

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

Output:

```
Reduce : (Total,181)
```

```
println("Tree Reduce : " + inputRDD.treeReduce((x, y) =>
("Total", x._2 + y._2)))
```

Output:

```
Tree Reduce : (Total,181)
```

```
inputRDD.count

     Output:
     res44: Long = 6


inputRDD.countApproxDistinct()

     Output:
     res45: Long = 5


listRDD.first

     Output:   res46: Int = 1


listRDD.top(2)

     Output:   res47: Array[Int] = Array(5, 4)


listRDD.take(2)

     Output:   res48: Array[Int] = Array(1, 2)


inputRDD.top(2)

     Output:   res49: Array[(String, Int)] = Array((Z,1),
     (C,40))


inputRDD.take(2)

     Output:   res50: Array[(String, Int)] = Array((Z,1),
     (B,30))


listRDD.min

     Output:   res51: Int = 1


listRDD.max

     Output:   res52: Int = 5
```

LAB 5: Spark SQL, creating tables and querying in data bricks

AIM:

To know about Spark SQL using scala in databricks.

PROGRAM:

```
import org.apache.spark.sql
import org.apache.spark.sql

// no error only warning
val sqla = new sql.SQLContext(sc)
```

Output:

```
command-3025449918807021:1: warning: constructor
SQLContext in class SQLContext is deprecated (since
2.0.0): Use SparkSession.builder instead
val sqla = new sql.SQLContext(sc)
           ^
sqla: org.apache.spark.sql.SQLContext =
org.apache.spark.sql.SQLContext@4d5f2953
```

```
sc.setLogLevel("ERROR")

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

Output:

```
command-3025449918807042:1: warning: constructor
SQLContext in class SQLContext is deprecated (since
2.0.0): Use SparkSession.builder instead
val sqla = new org.apache.spark.sql.SQLContext(sc)
           ^
sqla: org.apache.spark.sql.SQLContext =
org.apache.spark.sql.SQLContext@49e11b70
```

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

Output:

```
a: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[1] at parallelize at command-
3025449918807041:1
```

```
a.collect
```

Output:

```
res1: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

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

Output:

```
b: org.apache.spark.rdd.RDD[(Int, Int)] =
MapPartitionsRDD[2] at map at command-3025449918807039:1
```

```
b.collect
```

Output:

```
res2: 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")
```

Output:

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

```
df.show
```

Output:

```
+-----+------+
|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))
```

Output:

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

```
val df2 = a.toDF("Name", "Age")
```

Output:

```
df2: org.apache.spark.sql.DataFrame = [Name: string,
Age: int]
```

```
df2.show
```

Output:

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

```
val a = Seq(("Tom", 5), ("Jerry", 2), ("Donald", 7))
val df3 = a.toDF("Name", "Age")
df3.show
```

Output:

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

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

Output:

```
df3: org.apache.spark.sql.DataFrame = [Name: string,
Age: int]
```

```
// no error only warning
df3.registerTempTable("Cartoon") // give df cartoon name for
SQL
```

Output:

```
command-3025449918807033:2: warning: method
registerTempTable in class Dataset is deprecated (since
2.0.0): Use createOrReplaceTempView(viewName) instead.
df3.registerTempTable("Cartoon")
^
```

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

Output:

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

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

     Output:

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

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

     Output:

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

```
val b = List((1201, "Satish", 25), (1202, "Krishna", 28),
(1203, "Amirepatelh", 39), (1204, "Javed", 23), (1205,
"Pruthvi", 23))
val df = b.toDF("ID", "NAME", "AGE")
df.registerTempTable("Employee")
df.show
```

     Output:

```
     +----+-------+---+
     |  ID|   NAME|AGE|
     +----+-------+---+
     |1201| Satish| 25|
     |1202|Krishna| 28|
     |1203|  Amirepatelh| 39|
     |1204|  Javed| 23|
     |1205|Pruthvi| 23|
     +----+-------+---+

     command-3025449918807029:3: warning: method
     registerTempTable in class Dataset is deprecated (since
     2.0.0): Use createOrReplaceTempView(viewName) instead.
     df.registerTempTable("Employee")
   ^
b: List[(Int, String, Int)] = List((1201,Satish,25),
(1202,Krishna,28), (1203,Amirepatelh,39), (1204,Javed,23),
(1205,Pruthvi,23))
df: org.apache.spark.sql.DataFrame = [ID: int, NAME: string
... 1 more field]

df.printSchema
```

Output:

```
root
 |-- ID: integer (nullable = false)
 |-- NAME: string (nullable = true)
 |-- AGE: integer (nullable = false)
```

```
sqlContext.sql("select NAME from Employee").show
```

Output:

```
+--------+
|   NAME|
+--------+
| Satish|
|Krishna|
|  Amirepatelh|
|  Javed|
|Pruthvi|
+--------+
```

```
sqlContext.sql("select * from Employee where AGE > 23").show
```

Output:

```
+----+-------+---+
|  ID|   NAME|AGE|
+----+-------+---+
|1201| Satish| 25|
|1202|Krishna| 28|
|1203|  Amirepatelh| 39|
+----+-------+---+
```

```
df.groupBy("AGE").count().show
```

Output:

```
+---+-----+
|AGE|count|
+---+-----+
| 25|    1|
| 28|    1|
| 39|    1|
| 23|    2|
+---+-----+
df.show
```

Output:

```
+----+-------+---+
|  ID|   NAME|AGE|
+----+-------+---+
|1201| Satish| 25|
|1202|Krishna| 28|
|1203|  Amirepatelh| 39|
|1204|  Javed| 23|
|1205|Pruthvi| 23|
+----+-------+---+
```

```
// you may not have iris data
val df1 =
spark.read.format("csv").load("dbfs:/FileStore/shared_uploads
/mirepateljpatel01@gmail.com/Iris.csv")
```

Output:

```
df1: org.apache.spark.sql.DataFrame = [_c0: string, _c1:
string ... 4 more fields]
```

```
df1.show
```

Output:

```
+----+-----------+----------+-----------+----------+-----------+
| _c0|        _c1|       _c2|        _c3|       _c4|        _c5|
+----+-----------+----------+-----------+----------+-----------+
|null|sepallength|sepalwidth|petallength|petalwidth|      class|
|   0|        5.1|       3.5|        1.4|       0.2|Iris-setosa|
|   1|        4.9|       3.0|        1.4|       0.2|Iris-setosa|
|   2|        4.7|       3.2|        1.3|       0.2|Iris-setosa|
|   3|        4.6|       3.1|        1.5|       0.2|Iris-setosa|
|   4|        5.0|       3.6|        1.4|       0.2|Iris-setosa|
|   5|        5.4|       3.9|        1.7|       0.4|Iris-setosa|
|   6|        4.6|       3.4|        1.4|       0.3|Iris-setosa|
|   7|        5.0|       3.4|        1.5|       0.2|Iris-setosa|
|   8|        4.4|       2.9|        1.4|       0.2|Iris-setosa|
|   9|        4.9|       3.1|        1.5|       0.1|Iris-setosa|
|  10|        5.4|       3.7|        1.5|       0.2|Iris-setosa|
|  11|        4.8|       3.4|        1.6|       0.2|Iris-setosa|
|  12|        4.8|       3.0|        1.4|       0.1|Iris-setosa|
|  13|        4.3|       3.0|        1.1|       0.1|Iris-setosa|
|  14|        5.8|       4.0|        1.2|       0.2|Iris-setosa|
|  15|        5.7|       4.4|        1.5|       0.4|Iris-setosa|
|  16|        5.4|       3.9|        1.3|       0.4|Iris-setosa|
|  17|        5.1|       3.5|        1.4|       0.3|Iris-setosa|
|  18|        5.7|       3.8|        1.7|       0.3|Iris-setosa|
+----+-----------+----------+-----------+----------+-----------+
only showing top 20 rows
```

```
df.withColumnRenamed("_c0", "id")
df.withColumnRenamed("_c1", "sepallength")
df.withColumnRenamed("_c2", "sepalwidth")
df.withColumnRenamed("_c3", "petallength")
df.withColumnRenamed("_c4", "petalwidth")
df.withColumnRenamed("_c5", "class")
df1.show
```

Output:

```
+----+-----------+----------+-----------+----------+-----------+
| _c0|        _c1|       _c2|        _c3|       _c4|        _c5|
+----+-----------+----------+-----------+----------+-----------+
|null|sepallength|sepalwidth|petallength|petalwidth|      class|
|   0|        5.1|       3.5|        1.4|       0.2|Iris-setosa|
|   1|        4.9|       3.0|        1.4|       0.2|Iris-setosa|
|   2|        4.7|       3.2|        1.3|       0.2|Iris-setosa|
|   3|        4.6|       3.1|        1.5|       0.2|Iris-setosa|
|   4|        5.0|       3.6|        1.4|       0.2|Iris-setosa|
|   5|        5.4|       3.9|        1.7|       0.4|Iris-setosa|
|   6|        4.6|       3.4|        1.4|       0.3|Iris-setosa|
|   7|        5.0|       3.4|        1.5|       0.2|Iris-setosa|
|   8|        4.4|       2.9|        1.4|       0.2|Iris-setosa|
|   9|        4.9|       3.1|        1.5|       0.1|Iris-setosa|
|  10|        5.4|       3.7|        1.5|       0.2|Iris-setosa|
|  11|        4.8|       3.4|        1.6|       0.2|Iris-setosa|
|  12|        4.8|       3.0|        1.4|       0.1|Iris-setosa|
|  13|        4.3|       3.0|        1.1|       0.1|Iris-setosa|
|  14|        5.8|       4.0|        1.2|       0.2|Iris-setosa|
|  15|        5.7|       4.4|        1.5|       0.4|Iris-setosa|
|  16|        5.4|       3.9|        1.3|       0.4|Iris-setosa|
|  17|        5.1|       3.5|        1.4|       0.3|Iris-setosa|
|  18|        5.7|       3.8|        1.7|       0.3|Iris-setosa|
+----+-----------+----------+-----------+----------+-----------+
only showing top 20 rows
```

```
val rdda = sc.parallelize(1 to 1000, 10)
```

Output:

```
rdda: org.apache.spark.rdd.RDD[Int] =
ParallelCollectionRDD[28] at parallelize at command-
3025449918807055:1
```

```
rdda.partitions.length
```

Output:

```
res26: Int = 10
```

```
rdda.collect()
```

Output:

```
res27: 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, 101, 102, 103, 104, 105, 106,
107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128,
129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139,
140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150,
151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161,
162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172,
173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183,
184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205,
206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216,
217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227,
228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238,
239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249,
250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260,
261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271,
272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282,
283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293,
294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304,
305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315,
316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326,
327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337,
338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348,
349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359,
360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370,
371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381,
382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392,
393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403,
404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414,
415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425,
426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436,
437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447,
448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458,
459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469,
470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480,
481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491,
492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502,
503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513,
514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524,
525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535,
536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546,
547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557,
558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568,
569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579,
580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590,
591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601,
602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612,
613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623,
624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634,
635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645,
646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656,
657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667,
668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678,
679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689,
690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700,
701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711,
712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722,
723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733,
734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744,
745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755,
756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766,
767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777,
778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788,
789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799,
800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810,
811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821,
822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832,
833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843,
844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854,
855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865,
866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876,
877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887,
888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898,
899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909,
910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920,
921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931,
932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942,
943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953,
954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964,
965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975,
976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986,
987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997,
998, 999, 1000)
```

```
rdda.take(11)
```

Output:

```
res28: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11)
```

```
rdda.count()
```

Output:

```
res29: Long = 1000
```

```
rdda.saveAsTextFile("dbfs:/FileStore/shared_uploads/mirepatel
jpatel01@gmail.com/par1.txt")
```

```
val rddd =
sc.textFile("dbfs:/FileStore/shared_uploads/mirepateljpatel01
@gmail.com/par1.txt")
rddd.count()
```

Output:

```
rddd: org.apache.spark.rdd.RDD[String] =
dbfs:/FileStore/shared_uploads/mirepateljpatel01@gmail.c
om/par1.txt MapPartitionsRDD[34] at textFile at command-
3025449918807066:1
res33: Long = 1000
```

LAB 6: PIG and Hive Demo

AIM:

To know about basic concepts of pig latin and hive query language.

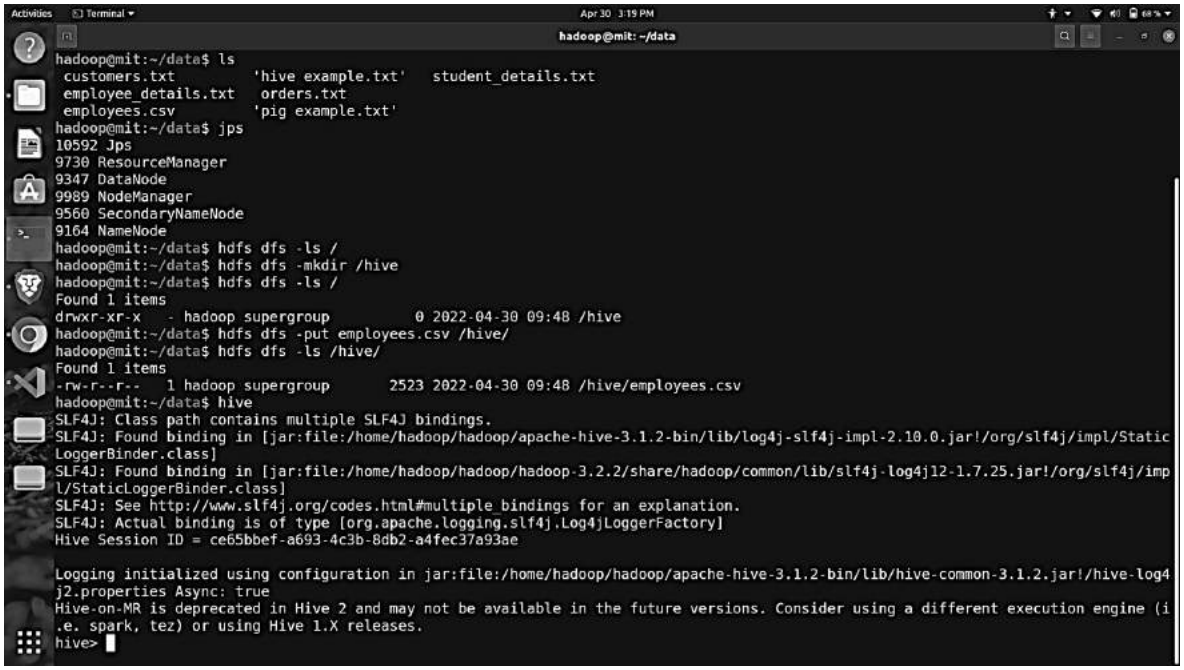
PROGRAM:

Install and configure hadoop and hive.

Start hadoop server and put all data in hadoop then start hive.

```
hdfs dfs -ls /
hdfs dfs -mkdir /hive
hdfs dfs -ls /
hdfs dfs -put employees.csv /hive/
hdfs dfs -ls /hive/
```

HIVE



```
hadoop@mit:~/data$ ls
 customers.txt          'hive example.txt'   student_details.txt
 employee_details.txt    orders.txt
 employees.csv          'pig example.txt'
hadoop@mit:~/data$ jps
10592 Jps
9730 ResourceManager
9347 DataNode
9989 NodeManager
9560 SecondaryNameNode
9164 NameNode
hadoop@mit:~/data$ hdfs dfs -ls /
hadoop@mit:~/data$ hdfs dfs -mkdir /hive
hadoop@mit:~/data$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2022-04-30 09:48 /hive
hadoop@mit:~/data$ hdfs dfs -put employees.csv /hive/
hadoop@mit:~/data$ hdfs dfs -ls /hive/
Found 1 items
-rw-r--r--   1 hadoop supergroup       2523 2022-04-30 09:48 /hive/employees.csv
hadoop@mit:~/data$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/Static
LoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/hadoop-3.2.2/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/imp
l/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = ce65bbef-a693-4c3b-8db2-a4fec37a93ae

Logging initialized using configuration in jar:file:/home/hadoop/hadoop/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4
j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i
.e. spark, tez) or using Hive 1.X releases.
hive> 
```

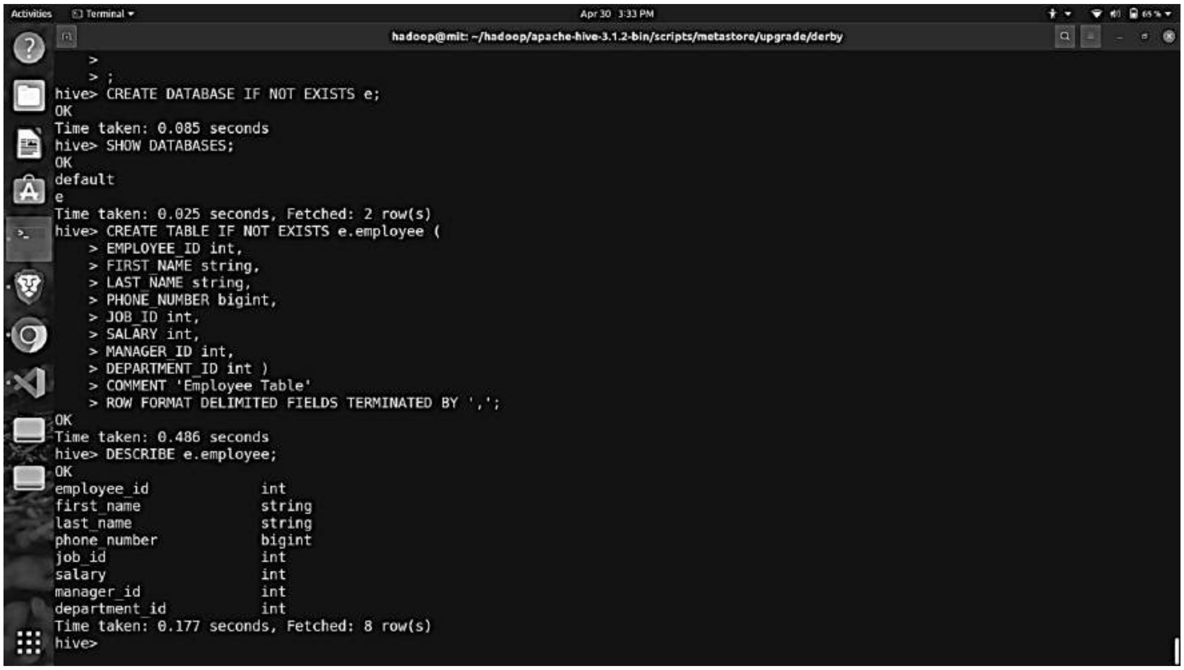
```
CREATE DATABASE IF NOT EXISTS e;
SHOW DATABASES;
```

```
hdfs dfs -ls /user/hive/warehouse
```

```
CREATE TABLE IF NOT EXISTS e.employee (
EMPLOYEE_ID int,
FIRST_NAME string,
LAST_NAME string,
PHONE_NUMBER bigint,
JOB_ID int,
SALARY int,
```

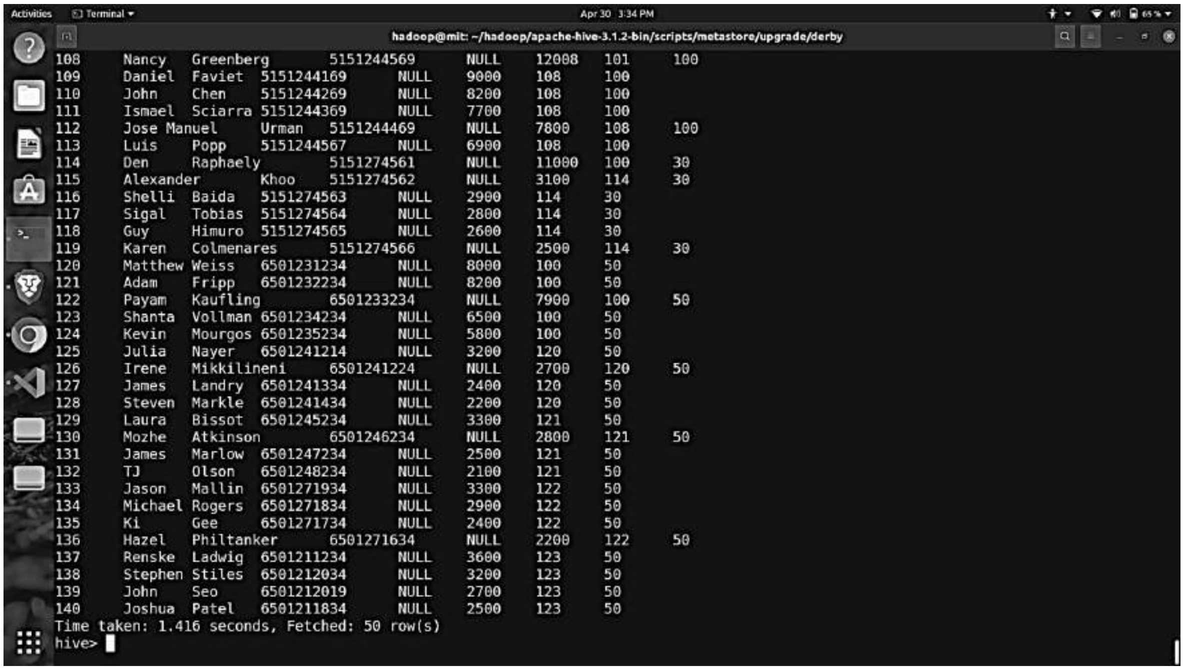
MANAGER_ID int,
DEPARTMENT_ID int)
COMMENT 'Employee Table'
ROW FORMAT DELIMIREPATELED FIELDS TERMINATED BY ',';

DESCRIBE e.employee;


Activities
Terminal
Apr 30 3:33 PM
hadoop@mit: ~/hadoop/apache-hive-3.1.2-bin/scripts/metastore/upgrade/derby

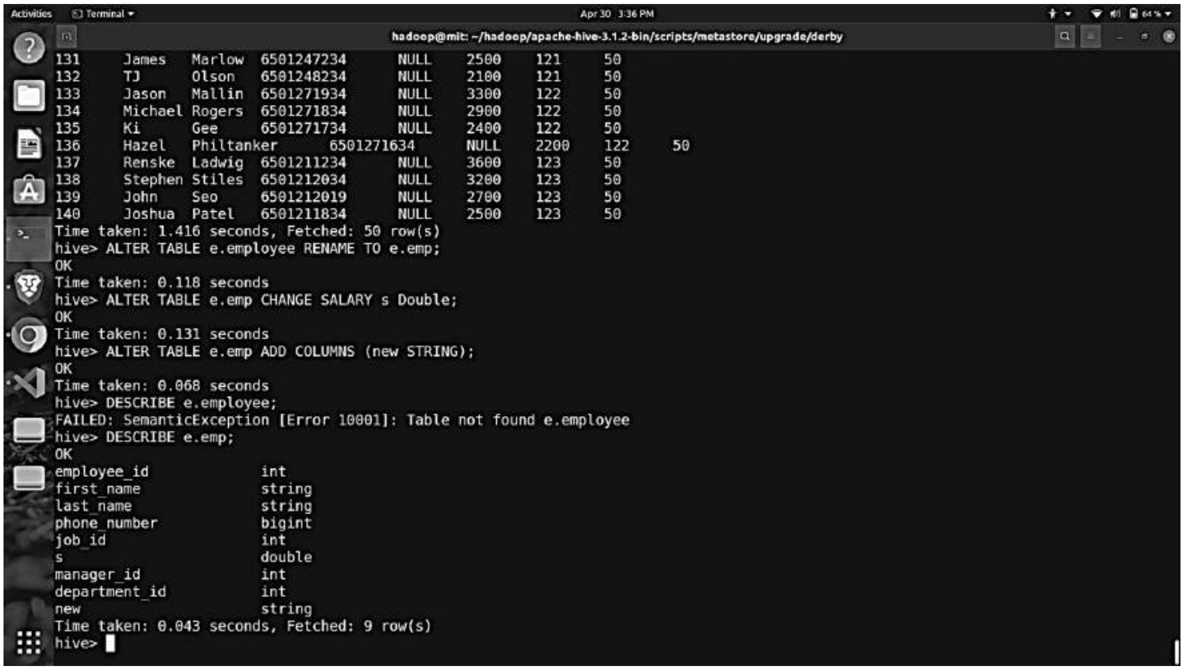
```
    >
    > ;
hive> CREATE DATABASE IF NOT EXISTS e;
OK
Time taken: 0.085 seconds
hive> SHOW DATABASES;
OK
default
e
Time taken: 0.025 seconds, Fetched: 2 row(s)
hive> CREATE TABLE IF NOT EXISTS e.employee (
    > EMPLOYEE_ID int,
    > FIRST_NAME string,
    > LAST_NAME string,
    > PHONE_NUMBER bigint,
    > JOB_ID int,
    > SALARY int,
    > MANAGER_ID int,
    > DEPARTMENT_ID int )
    > COMMENT 'Employee Table'
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
OK
Time taken: 0.486 seconds
hive> DESCRIBE e.employee;
OK
employee_id             int
first_name              string
last_name               string
phone_number            bigint
job_id                  int
salary                  int
manager_id              int
department_id           int
Time taken: 0.177 seconds, Fetched: 8 row(s)
hive>
```

LOAD DATA INPATH '/hive/employees.csv' INTO TABLE e.employee;
SELECT * FROM e.employee;


Activities
Terminal
Apr 30 3:34 PM
hadoop@mit: ~/hadoop/apache-hive-3.1.2-bin/scripts/metastore/upgrade/derby

```
108     Nancy   Greenberg       5151244569      NULL    12008   101     100
109     Daniel  Faviet  5151244169      NULL    9000    108     100
110     John    Chen    5151244269      NULL    8200    108     100
111     Ismael  Sciarra 5151244369      NULL    7700    108     100
112     Jose Manuel     Urman   5151244469      NULL    7800    108     100
113     Luis    Popp    5151244567      NULL    6900    108     100
114     Den     Raphaely        5151274561      NULL    11000   100     30
115     Alexander       Khoo    5151274562      NULL    3100    114     30
116     Shelli  Baida   5151274563      NULL    2900    114     30
117     Sigal   Tobias  5151274564      NULL    2800    114     30
118     Guy     Himuro  5151274565      NULL    2600    114     30
119     Karen   Colmenares      5151274566      NULL    2500    114     30
120     Matthew Weiss   6501231234      NULL    8000    100     50
121     Adam    Fripp   6501232234      NULL    8200    100     50
122     Payam   Kaufling        6501233234      NULL    7900    100     50
123     Shanta  Vollman 6501234234      NULL    6500    100     50
124     Kevin   Mourgos 6501235234      NULL    5800    100     50
125     Julia   Nayer   6501241214      NULL    3200    120     50
126     Irene   Mikkilineni     6501241224      NULL    2700    120     50
127     James   Landry  6501241334      NULL    2400    120     50
128     Steven  Markle  6501241434      NULL    2200    120     50
129     Laura   Bissot  6501245234      NULL    3300    121     50
130     Mozhe   Atkinson        6501246234      NULL    2800    121     50
131     James   Marlow  6501247234      NULL    2500    121     50
132     TJ      Olson   6501248234      NULL    2100    121     50
133     Jason   Mallin  6501271934      NULL    3300    122     50
134     Michael Rogers  6501271834      NULL    2900    122     50
135     Ki      Gee     6501271734      NULL    2400    122     50
136     Hazel   Philtanker      6501271634      NULL    2200    122     50
137     Renske  Ladwig  6501211234      NULL    3600    123     50
138     Stephen Stiles  6501212034      NULL    3200    123     50
139     John    Seo     6501212019      NULL    2700    123     50
140     Joshua  Patel   6501211834      NULL    2500    123     50
Time taken: 1.416 seconds, Fetched: 50 row(s)
hive> 
```

ALTER TABLE e.employee RENAME TO e.emp;
ALTER TABLE e.emp CHANGE SALARY s Double;
ALTER TABLE emp ADD COLUMNS (new STRING);



Activities
Terminal
Apr 30 3:36 PM
hadoop@mit: ~/hadoop/apache-hive-3.1.2-bin/scripts/metastore/upgrade/derby

```
131     James   Marlow  6501247234      NULL    2500    121     50
132     TJ      Olson   6501248234      NULL    2100    121     50
133     Jason   Mallin  6501271934      NULL    3300    122     50
134     Michael Rogers  6501271834      NULL    2900    122     50
135     Ki      Gee     6501271734      NULL    2400    122     50
136     Hazel   Philtanker      6501271634      NULL    2200    122     50
137     Renske  Ladwig  6501211234      NULL    3600    123     50
138     Stephen Stiles  6501212034      NULL    3200    123     50
139     John    Seo     6501212019      NULL    2700    123     50
140     Joshua  Patel   6501211834      NULL    2500    123     50
Time taken: 1.416 seconds, Fetched: 50 row(s)
hive> ALTER TABLE e.employee RENAME TO e.emp;
OK
Time taken: 0.118 seconds
hive> ALTER TABLE e.emp CHANGE SALARY s Double;
OK
Time taken: 0.131 seconds
hive> ALTER TABLE e.emp ADD COLUMNS (new STRING);
OK
Time taken: 0.068 seconds
hive> DESCRIBE e.employee;
FAILED: SemanticException [Error 10001]: Table not found e.employee
hive> DESCRIBE e.emp;
OK
employee_id             int
first_name              string
last_name               string
phone_number            bigint
job_id                  int
s                       double
manager_id              int
department_id           int
new                     string
Time taken: 0.043 seconds, Fetched: 9 row(s)
hive> █
```

You can use SQL like query.

SELECT * FROM e.emp WHERE salary>20000;

CREATE TABLE e.filter AS SELECT employee_id,first_name,last_name FROM e.emp WHERE salary<9000;

CREATE TABLE e.similar LIKE e.emp;

#Exports to LOCAL directory
INSERT OVERWRITE DIRECTORY '/tmp/export' ROW FORMAT DELIMIREPATELED FIELDS TERMINATED BY ',' SELECT * FROM e.emp;

#Exports to HDFS directory
bin/hive -e "INSERT OVERWRITE DIRECTORY '/user/data/output/export' ROW FORMAT DELIMIREPATELED FIELDS TERMINATED BY ',' SELECT * FROM emp.employee"

You can view your hive data in hdfs using following
hdfs dfs -ls /user/hive/warehouse

PIG

To run pig scripts first we have to configure some things.

cd hadoop/hadoop-3.2.2/sbin/

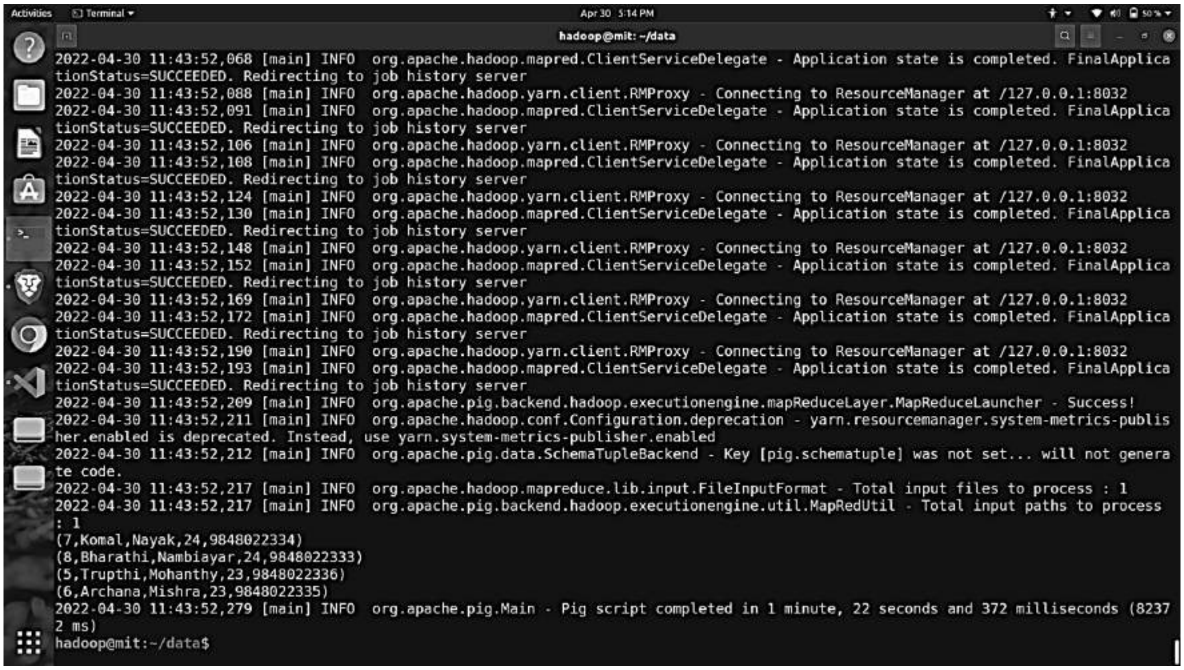
mr-jobhistory-daemon.sh start historyserver

Now make a pig script file and upload data to hadoop to run pig.

```
nano a.pig
“”””
student = LOAD '/hive/student_details.txt' USING PigStorage(',')
   as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray);

DESCRIBE student

student_order = ORDER student BY age DESC;

student_limirepatel = LIMIREPATEL student_order 4;

Dump student_limirepatel;
“”””
```

Ctrl+S and Ctrl+X

```
pig a.pig
```



```
2022-04-30 11:43:52,068 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplica
tionStatus=SUCCEEDED. Redirecting to job history server
2022-04-30 11:43:52,088 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /127.0.0.1:8032
2022-04-30 11:43:52,091 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplica
tionStatus=SUCCEEDED. Redirecting to job history server
2022-04-30 11:43:52,106 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /127.0.0.1:8032
2022-04-30 11:43:52,108 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplica
tionStatus=SUCCEEDED. Redirecting to job history server
2022-04-30 11:43:52,124 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /127.0.0.1:8032
2022-04-30 11:43:52,130 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplica
tionStatus=SUCCEEDED. Redirecting to job history server
2022-04-30 11:43:52,148 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /127.0.0.1:8032
2022-04-30 11:43:52,152 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplica
tionStatus=SUCCEEDED. Redirecting to job history server
2022-04-30 11:43:52,169 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /127.0.0.1:8032
2022-04-30 11:43:52,172 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplica
tionStatus=SUCCEEDED. Redirecting to job history server
2022-04-30 11:43:52,190 [main] INFO  org.apache.hadoop.yarn.client.RMProxy - Connecting to ResourceManager at /127.0.0.1:8032
2022-04-30 11:43:52,193 [main] INFO  org.apache.hadoop.mapred.ClientServiceDelegate - Application state is completed. FinalApplica
tionStatus=SUCCEEDED. Redirecting to job history server
2022-04-30 11:43:52,209 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2022-04-30 11:43:52,211 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - yarn.resourcemanager.system-metrics-publis
her.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
2022-04-30 11:43:52,212 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not genera
te code.
2022-04-30 11:43:52,217 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to process : 1
2022-04-30 11:43:52,217 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process
: 1
(7,Komal,Nayak,24,9848022334)
(8,Bharathi,Nambiayar,24,9848022333)
(5,Trupthi,Mohanthy,23,9848022336)
(6,Archana,Mishra,23,9848022335)
2022-04-30 11:43:52,279 [main] INFO  org.apache.pig.Main - Pig script completed in 1 minute, 22 seconds and 372 milliseconds (8237
2 ms)
hadoop@mit:~/data$
```

some other commands are

```
group_data = GROUP student by age;
dump group_data;
Illustrate group_data;
group_multiple = GROUP student by (age, firstname);
dump group_multiple;
```

LAB 7: Structured Streaming using the Python DataFrames API

AIM:

To know about Structured Streaming using the Python DataFrames API on spark.

PROGRAM:

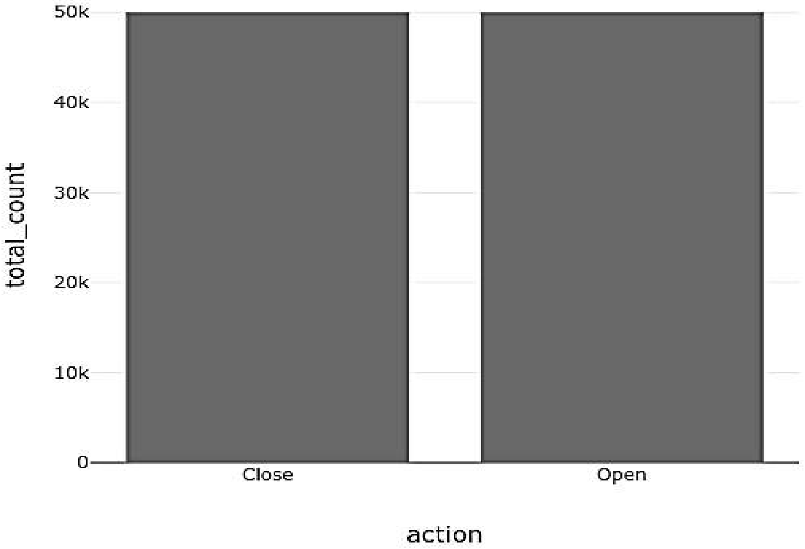
```
from pyspark.sql.types import *

inputPath = "/databricks-datasets/structured-
streaming/events/"

# Since we know the data format already, let's define the
schema to speed up processing (no need for Spark to infer
schema)
jsonSchema = StructType([ StructField("time",
TimestampType(), True), StructField("action", StringType(),
True) ])

# Static DataFrame representing data in the JSON files
staticInputDF = (
  spark
    .read
    .schema(jsonSchema)
    .json(inputPath)
)

display(staticInputDF)
```

Output:

```
2016-07-28T04:19:28.000+0000    Close
2016-07-28T04:19:28.000+0000    Close
2016-07-28T04:19:29.000+0000    Open
2016-07-28T04:19:31.000+0000    Close
2016-07-28T04:19:31.000+0000    Open
2016-07-28T04:19:31.000+0000    Open
2016-07-28T04:19:32.000+0000    Close
2016-07-28T04:19:33.000+0000    Close
2016-07-28T04:19:35.000+0000    Close
2016-07-28T04:19:36.000+0000    Open
2016-07-28T04:19:38.000+0000    Close
2016-07-28T04:19:40.000+0000    Open
2016-07-28T04:19:41.000+0000    Close
2016-07-28T04:19:42.000+0000    Open
2016-07-28T04:19:45.000+0000    Open
2016-07-28T04:19:47.000+0000    Open
2016-07-28T04:19:48.000+0000    Open
Truncated results, showing first 1000 rows.
```

```
from pyspark.sql.functions import *       # for window()
function

staticCountsDF = (
  staticInputDF
    .groupBy(
       staticInputDF.action,
       window(staticInputDF.time, "1 hour"))
    .count()
)
staticCountsDF.cache()

# Register the DataFrame as table 'static_counts'
staticCountsDF.createOrReplaceTempView("static_counts")
%sql select action, sum(count) as total_count from
static_counts group by action
```

Output:



```
%sql select action, date_format(window.end, "MMM-dd HH:mm")
as time, count from static_counts order by time, action
```

Output:

```
Close   Jul-26 03:00   11
Open    Jul-26 03:00   179
Close   Jul-26 04:00   344
Open    Jul-26 04:00   1001
Close   Jul-26 05:00   815
Open    Jul-26 05:00   999
Close   Jul-26 06:00   1003
```

```
from pyspark.sql.functions import *

# Similar to definition of staticInputDF above, just using
`readStream` instead of `read`
streamingInputDF = (
  spark
    .readStream
    .schema(jsonSchema)               # Set the schema of the
JSON data
    .option("maxFilesPerTrigger", 1)  # Treat a sequence of
files as a stream by picking one file at a time
    .json(inputPath)
)

# Same query as staticInputDF
streamingCountsDF = (
  streamingInputDF
    .groupBy(
      streamingInputDF.action,
      window(streamingInputDF.time, "1 hour"))
    .count()
)

# Is this DF actually a streaming DF?
streamingCountsDF.isStreaming
```

Output:

```
True
```

```
spark.conf.set("spark.sql.shuffle.partitions", "2")  # keep
the size of shuffles small

query = (
  streamingCountsDF
    .writeStream
    .format("memory")        # memory = store in-memory table
    .queryName("counts")     # counts = name of the in-memory
table
    .outputMode("complete")  # complete = all the counts
should be in the table
    .start()
)
```

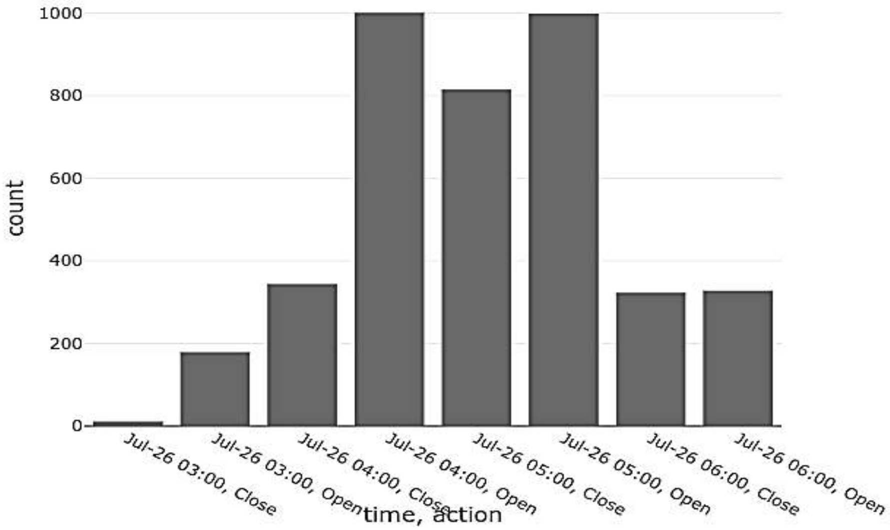
Output:

```
counts(id: 704d28e8-ea50-483a-96da-433d3a0a39fa)
Last updated: 3 days ago
```

```
from time import sleep
```

```
sleep(5)  # wait a bit for computation to start
%sql select action, date_format(window.end, "MMM-dd HH:mm")
as time, count from counts order by time, action
```

Output:



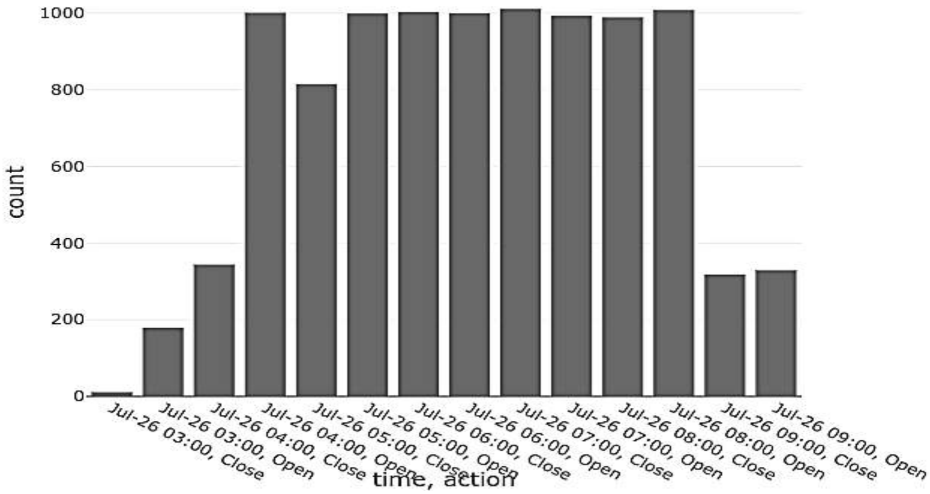
```
%sql select action, date_format(window.end, "MMM-dd HH:mm")
as time, count from counts order by time, action
```

Output:

```
Close   Jul-26 03:00    11
Open    Jul-26 03:00    179
Close   Jul-26 04:00    344
Open    Jul-26 04:00    1001
Close   Jul-26 05:00    815
Open    Jul-26 05:00    999
Close   Jul-26 06:00    1003
```

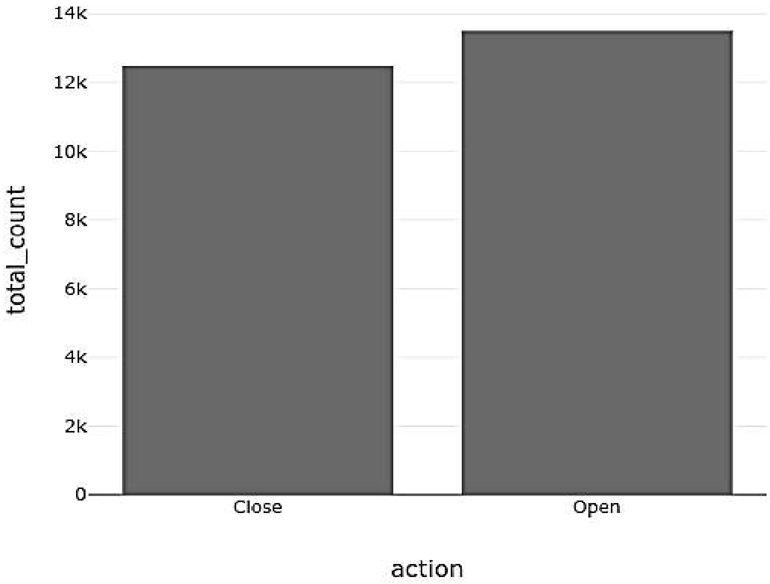
```
%sql select action, date_format(window.end, "MMM-dd HH:mm")
as time, count from counts order by time, action
```

Output:



```
%sql select action, sum(count) as total_count from counts
group by action order by action
```

Output:



```
query.stop()

%scala
import org.apache.spark.ml.feature.{HashingTF, IDF,
Tokenizer}

val sentenceData = spark.createDataFrame(Seq(
  (0.0, "Hi I heard about Spark"),
  (0.0, "I wish Java could use case classes"),
  (1.0, "Logistic regression models are neat")
)).toDF("label", "sentence")

val tokenizer = new
Tokenizer().setInputCol("sentence").setOutputCol("words")
val wordsData = tokenizer.transform(sentenceData)

val hashingTF = new HashingTF()

.setInputCol("words").setOutputCol("rawFeatures").setNumFeatu
res(20)

val featurizedData = hashingTF.transform(wordsData)
// alternatively, CountVectorizer can also be used to get
term frequency vectors

val idf = new
IDF().setInputCol("rawFeatures").setOutputCol("features")
val idfModel = idf.fit(featurizedData)

val rescaledData = idfModel.transform(featurizedData)
rescaledData.select("label", "features").show()
```

Output:

```
+-----+--------------------+
|label|            features|
+-----+--------------------+
|  0.0|(20,[6,8,13,16],[...|
|  0.0|(20,[0,2,7,13,15,...|
|  1.0|(20,[3,4,6,11,19]...|
+-----+--------------------+
```

LAB 8: Page Rank

AIM:

To know about google's page rank algorithm.

PROGRAM:

```
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)
```

Output:

```
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 assiciates number in pages to labels
page_labels = {p: l for p, l in zip(pages, labels)}
page_labels
```

Output:

```
Out[117]: {0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5:
'F', 6: 'G'}
```

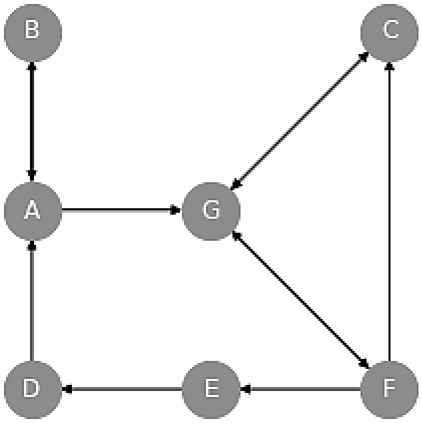
Connections between pages will be described by a list of pairs, whose elements respectively identify the starting and the ending page.

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

!pip install networkx

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])
```

The visualization of a graph is triggered by the invocation of nx.draw, and it highlights directed arcs (using a thicker part of an edge to show their direction), node labels and different coloring of nodes. We will use the latter feature later on.

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

Output:



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

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

print (adjacency)
```

Output:

```
{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]:
        # this builds the transition matrix (beware of
indices!)
        connection_matrix.append((b, a,
1./len(adjacency[a])))
connection_matrix
```

Output:

```
Out[123]: [(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)]
```

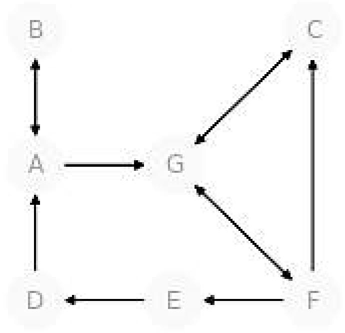
```
links_RDD = sc.parallelize(connection_matrix).cache()
Let's just peek into the first three elements of this RDD:

links_RDD.take(3)
```

Output:

```
Out[125]: [(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)

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)])

iteration = 0
while(l2distance(old_page_rank, page_rank) >= tolerance and 
iteration < max_iterations):
    old_page_rank = page_rank
    page_rank_values = (links_RDD
#                          .map(lambda (i, j, m): i, 
(m*page_rank[j]))
                        .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)
    iteration += 1

plt.cla()
display(nx.draw(g, with_labels=True, labels=page_labels,
node_color=page_rank,
                node_size=800, font_color='darkgrey',
cmap=plt.cm.Blues, pos=positions))
```

Output:



```
def get_graph(pages, links):
    g = nx.DiGraph()

    for p in pages:
        g.add_node(p)

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

    return g

def get_connection_matrix(pages, links):
    incidency = {}
    for u in range(len(pages)):
        incidency[u] = []

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

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

```
    return connection_matrix

def get_page_rank(pages, links, verbose=False, tolerance=10e-
7, max_iterations=100):
    connection_matrix = get_connection_matrix(pages, links)
    links_RDD = sc.parallelize(connection_matrix).cache()

    n = len(pages)
    page_rank = np.ones(n)/n
    old_page_rank = np.ones(n)

    iteration = 0
    while(l2distance(old_page_rank, page_rank) >= tolerance
and iteration < max_iterations):
        old_page_rank = page_rank
        page_rank_values = (links_RDD
#                              .map(lambda (i, j, m): (i,
m*page_rank[j]))
                            .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])

        if verbose:
            nice_print(page_rank)

        iteration += 1

    print(' %d iterations' % iteration)

    return page_rank

page_rank = get_page_rank(pages, links)
```

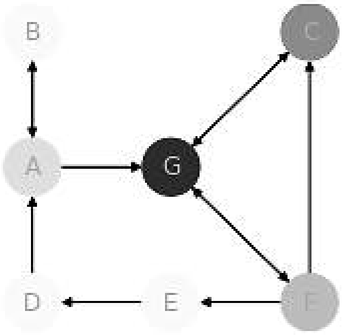
Output:**21 iterations**

```
nice_print(page_rank)
```

Output:**0.111, 0.056, 0.222, 0.055, 0.056, 0.166, 0.334**

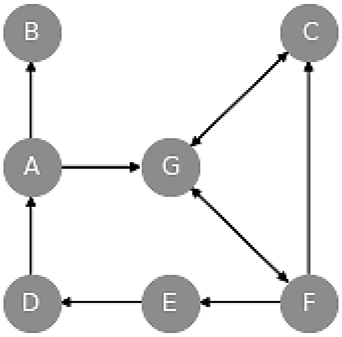
```
g = get_graph(pages, links)
plt.cla()
display(nx.draw(g, with_labels=True, labels=page_labels,
node_color=page_rank,
                node_size=800, font_color='darkgrey',
cmap=plt.cm.Blues, pos=positions))
```

Output:



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

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

Output:



```
page_rank = get_page_rank(pages, links_dead)
nice_print(page_rank)
```

Output: **73 iterations**

```
0.004, 0.002, 0.014, 0.004, 0.004, 0.010, 0.020
```

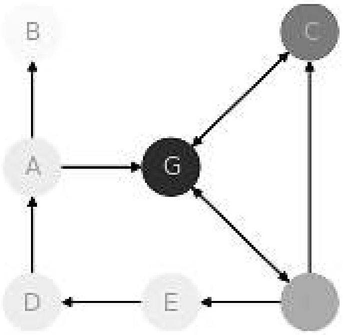
```
sum(page_rank)
```

Output: **Out[139]: 0.05721862657052279**

```
plt.cla()
```

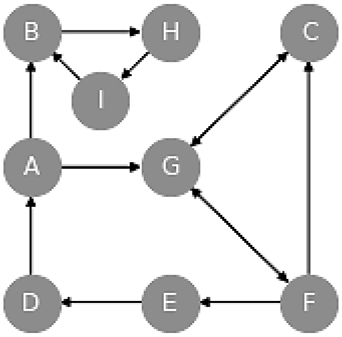
```
display(nx.draw(g, with_labels=True, labels=page_labels,
node_color=page_rank,
                node_size=800, font_color='darkgrey',
cmap=plt.cm.Blues, pos=positions))
```

Output:



```
labels_trap = ['A', 'B', 'C', 'D', 'E', 'F', 'G',  'H', 'I']

positions_trap = [(0, 1), (0, 2),  (2, 2), (0, 0),  (1, 0),
(2, 0),  (1, 1), (1, 2), (0.5, 1.5)]

pages_trap = range(len(labels_trap))

page_labels = {page: label for page, label in zip(pages_trap,
labels_trap)}

links_trap = [(3, 0),  (0, 1),  (5, 2), (6, 2), (6, 5), (5,
6),  (2, 6), (0, 6),  (5, 4),  (4, 3), (1, 7), (7, 8), (8,
1)]

g = get_graph(pages_trap, links_trap)
plt.cla()
display(nx.draw(g, with_labels=True, labels=page_labels,
                node_size=800, node_color='#8888CC',
font_color='white',
                pos=positions_trap))
```

Output:



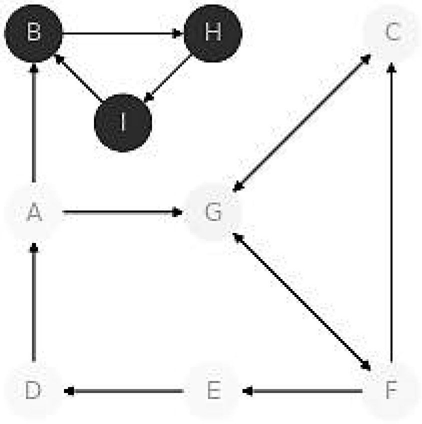
```
page_rank = get_page_rank(pages_trap, links_trap)
nice_print(page_rank)
```

***Output:* 100 iterations**

```
0.001, 0.336, 0.004, 0.001, 0.001, 0.003, 0.006, 0.324,
0.324
```

```
display(nx.draw(g, with_labels=True, labels=page_labels,
node_color=page_rank,
                node_size=800, font_color='darkgrey',
cmap=plt.cm.Blues, pos=positions_trap,
                vmin=0, vmax=1./7))
```

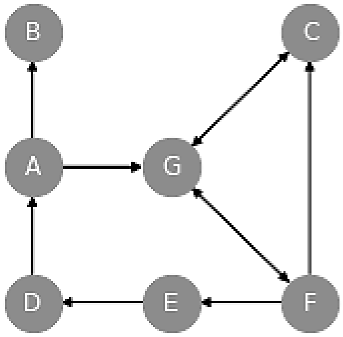
Output:



```
def get_page_rank(pages, links, beta=0.8, max_iterations=400,
tolerance=1.e-5, verbose=False):

    connection_matrix = get_connection_matrix(pages, links)
    links_RDD = sc.parallelize(connection_matrix).cache()
    n = len(pages)
    page_rank = np.ones(n)/n
    old_page_rank = np.ones(n)
    iteration = 0
    while(l2distance(old_page_rank, page_rank) >= tolerance
and iteration < max_iterations):
        old_page_rank = page_rank
        page_rank_values = (links_RDD
#                              .map(lambda (i, j, m): (i,
m*page_rank[j]))
                             .map(lambda x : (x[0],
x[2]*page_rank[x[1]]))
```

```
                            .reduceByKey(lambda a, b: a+b)
                            .sortByKey()
                            .collect()
                           )
        # TODO: add the page_rank vector computation, taking
into account the taxation method
        page_rank = np.array([beta * c + (1 - beta) * 1.0/n
for (i, c) in page_rank_values])

        if verbose:
            nice_print(page_rank)
        iteration += 1
    print(' %d iterations' % iteration)
    return page_rank

labels = ['A',  'B',  'C',  'D', 'E', 'F',  'G']
pages = range(len(labels))
page_labels = {page: label for page, label in zip(pages,
labels)}
g = get_graph(pages, links_dead)
plt.cla()
display(nx.draw(g, with_labels=True, labels=page_labels,
                node_size=800, node_color='#8888CC',
font_color='white',
                pos=positions))
```

Output:



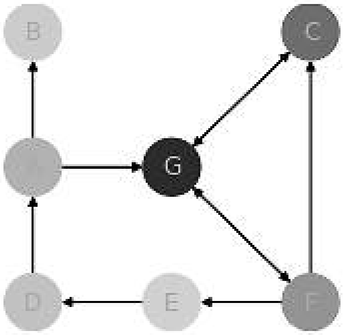
```
page_rank = get_page_rank(pages, links_dead)
nice_print(page_rank)
```

Output: **8 iterations**

```
0.089, 0.065, 0.143, 0.076, 0.059, 0.113, 0.210
```

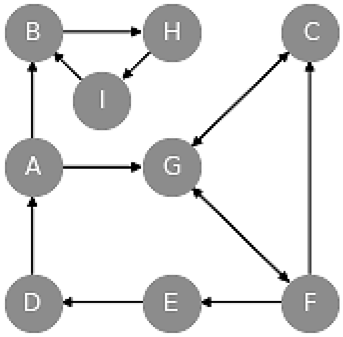
```
plt.cla()

display(nx.draw(g, with_labels=True, labels=page_labels,
node_color=page_rank,
                node_size=800, font_color='darkgrey',
cmap=plt.cm.Blues, pos=positions,
                vmin=0, vmax=0.2))
```

Output:



```
page_labels = {page: label for page, label in zip(pages_trap,
labels_trap)}


g = get_graph(pages_trap, links_trap)
plt.cla()
display(nx.draw(g, with_labels=True, labels=page_labels,
                 node_size=800, node_color='#8888CC',
font_color='white',
                 pos=positions_trap))
```

Output:



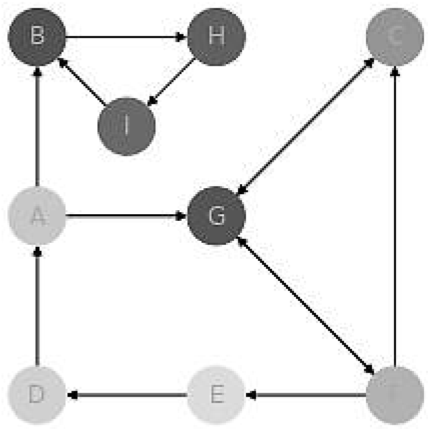
```
page_rank = get_page_rank(pages_trap, links_trap)
nice_print(page_rank)
```

***Output:* 9 iterations**

```
0.070, 0.165, 0.111, 0.059, 0.046, 0.088, 0.163, 0.154,
0.146
```

```
display(nx.draw(g, with_labels=True, labels=page_labels,
node_color=page_rank,
                node_size=800, font_color='darkgrey',
cmap=plt.cm.Blues, pos=positions_trap,
                vmin=0, vmax=0.2))
```

Output:



LAB 9: Machine Learning

AIM:

To know how to apply machine learning on big data.

PROGRAM:

```
import org.apache.spark.sql.Encoders

case class eCommerce(Email: String,
                     Avatar: String,
                     Avg_Session_Length: Double,
                     Time_on_App: Double,
                     Time_on_Website: Double,
                     Length_of_Membership: Double,
                     Yearly_Amount_Spent: Double)

val eCommerceSchema = Encoders.product[eCommerce].schema
val eCommerceDF =
spark.read.schema(eCommerceSchema).option("header",
"true").csv("dbfs:/FileStore/tables/ecommerce.csv")
display(eCommerceDF)

eCommerceDF.printSchema()
```

Output:

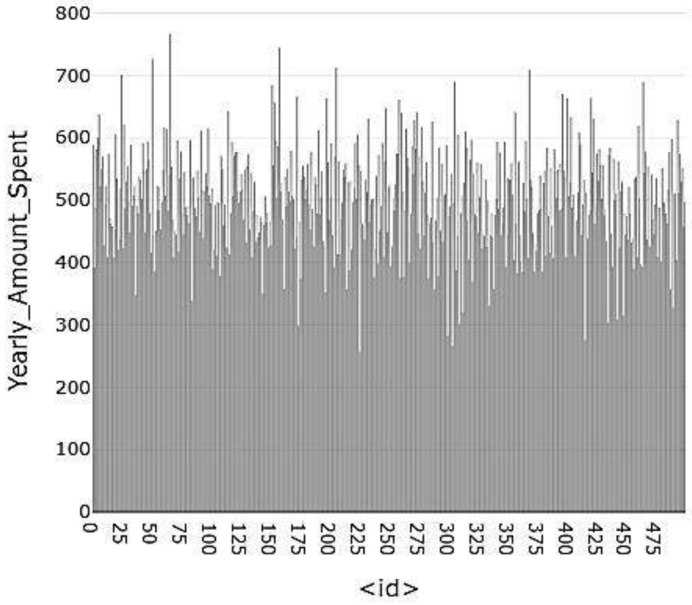
```
root
 |-- Email: string (nullable = true)
 |-- Avatar: string (nullable = true)
 |-- Avg_Session_Length: double (nullable = true)
 |-- Time_on_App: double (nullable = true)
 |-- Time_on_Website: double (nullable = true)
 |-- Length_of_Membership: double (nullable = true)
 |-- Yearly_Amount_Spent: double (nullable = true)
```

```
eCommerceDF.select("Avg_Session_Length","Time_on_App",
"Time_on_Website", "Length_of_Membership",
"Yearly_Amount_Spent").describe().show()
```

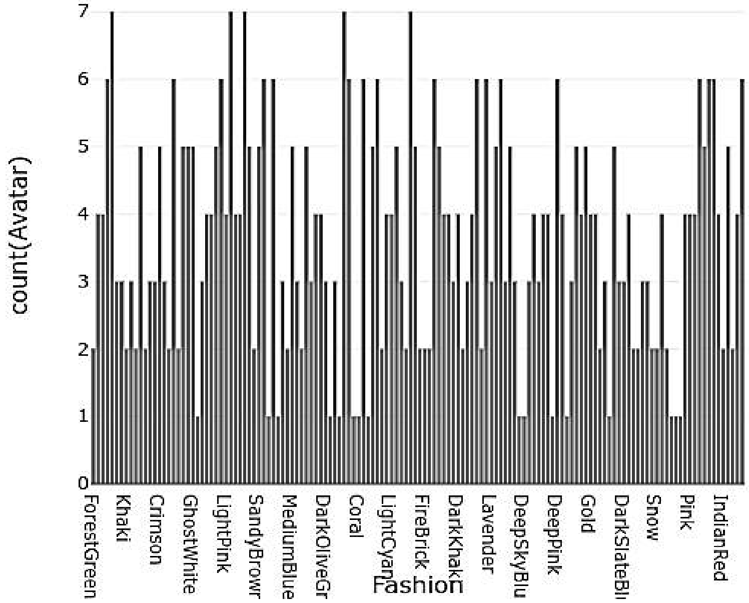
Output:

```
+-------+------------------+------------------+------------------+-
-------------------+-------------------+
|summary|Avg_Session_Length|       Time_on_App|
Time_on_Website|Length_of_Membership|Yearly_Amount_Spent|
+-------+------------------+------------------+------------------+-
-------------------+-------------------+
|  count|               500|               500|               500|
500|                500|
|   mean|    33.05319351824|12.052487936928012|37.060445421080004|
3.5334615559300007|  499.3140382608002|
| stddev|0.9925631111602911|0.9942156084624618|1.0104889068105993|
0.9992775024367542|  79.31478155115914|
|    min|       29.53242897|       8.508152176|       33.91384725|
0.26990109|        256.6705823|
|    max|       36.13966249|       15.12699429|       40.00518164|
6.922689335|        765.5184619|
+-------+------------------+------------------+------------------+-
-------------------+-------------------+
```

```
eCommerceDF.createOrReplaceTempView("EcommerceData")

%sql
select * from EcommerceData

%sql
select Yearly_Amount_Spent from EcommerceData
```

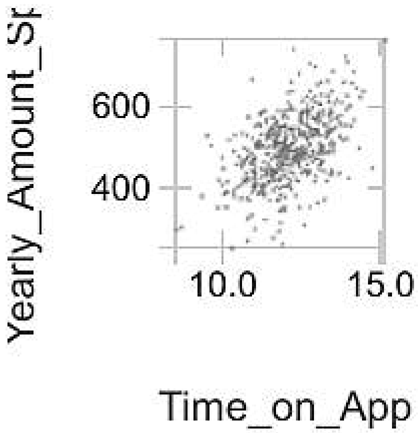


```
%sql
select Avatar as Fashion, count(Avatar) from EcommerceData
group by Avatar
```



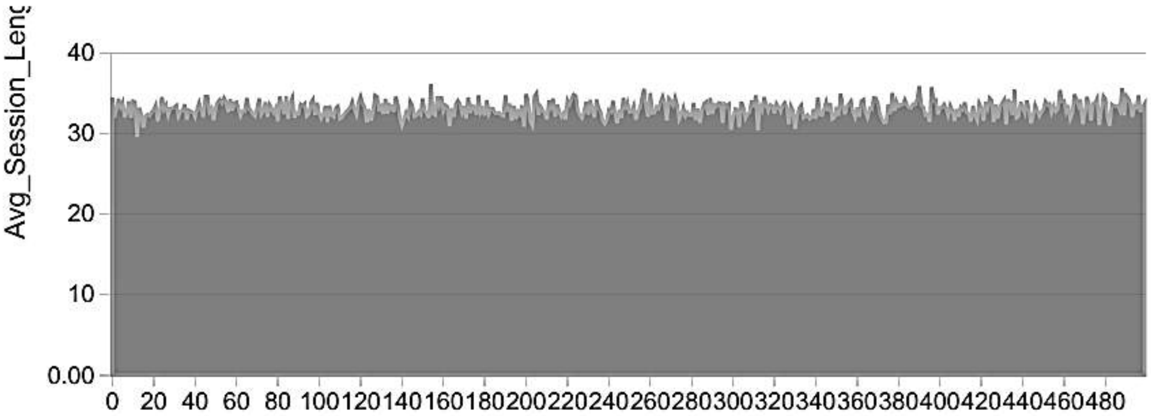
```
%sql
select Email, Avatar, Avg_Session_Length, Time_on_App,
Time_on_Website, Length_of_Membership, Yearly_Amount_Spent
from EcommerceData
```

```
%sql
select Yearly_Amount_Spent, Time_on_App from EcommerceData
```

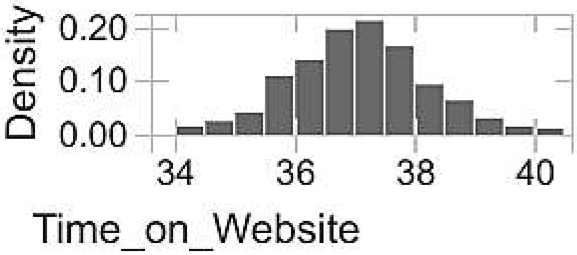


```
%sql
select Yearly_Amount_Spent, Avg_Session_Length from
EcommerceData
```

```
%sql
select Yearly_Amount_Spent, Avg_Session_Length from
EcommerceData
```



```
%sql
select Yearly_Amount_Spent, Time_on_Website from
EcommerceData
```



```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.Row
import org.apache.spark.sql.types._
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.feature.StringIndexer

var StringfeatureCol = Array("Email", "Avatar")

val df = spark.createDataFrame(
  Seq((0, "a"), (1, "b"), (2, "c"), (3, "a"), (4, "a"), (5,
"c"))
).toDF("id", "category")
```

Output:

```
df: org.apache.spark.sql.DataFrame = [id: int, category:
string]
```

```
val indexer = new StringIndexer()
  .setInputCol("category")
  .setOutputCol("categoryIndex")
val indexed = indexer.fit(df).transform(df)
```

Output:

```
indexer: org.apache.spark.ml.feature.StringIndexer =
strIdx_2f1680be82ba
indexed: org.apache.spark.sql.DataFrame = [id: int,
category: string ... 1 more field]
```

```
indexed.show()
```

Output:

```
+---+--------+-------------+
| id|category|categoryIndex|
+---+--------+-------------+
|  0|       a|          0.0|
|  1|       b|          2.0|
|  2|       c|          1.0|
|  3|       a|          0.0|
|  4|       a|          0.0|
|  5|       c|          1.0|
+---+--------+-------------+
```

```
import org.apache.spark.ml.attribute.Attribute
import org.apache.spark.ml.feature.{IndexToString,
StringIndexer}
import org.apache.spark.ml.{Pipeline, PipelineModel}

val indexers = StringfeatureCol.map { colName =>
  new
StringIndexer().setInputCol(colName).setOutputCol(colName +
"_indexed")
}

val pipeline = new Pipeline()
                    .setStages(indexers)

val FinalecommerceDF =
pipeline.fit(eCommerceDF).transform(eCommerceDF)
```

Output:

```
indexers:
Array[org.apache.spark.ml.feature.StringIndexer] =
Array(strIdx_47d73e6c8774, strIdx_b08f82939c6a)
pipeline: org.apache.spark.ml.Pipeline =
pipeline_c6ea616eba7f
FinalecommerceDF: org.apache.spark.sql.DataFrame =
[Email: string, Avatar: string ... 7 more fields]
```

```
FinalecommerceDF.printSchema()
```

Output:

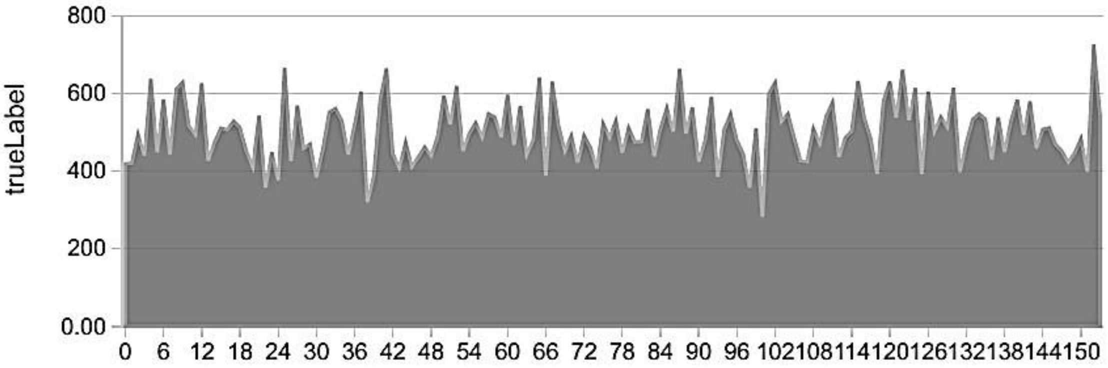
```
root
 |-- Email: string (nullable = true)
 |-- Avatar: string (nullable = true)
 |-- Avg_Session_Length: double (nullable = true)
 |-- Time_on_App: double (nullable = true)
 |-- Time_on_Website: double (nullable = true)
 |-- Length_of_Membership: double (nullable = true)
 |-- Yearly_Amount_Spent: double (nullable = true)
 |-- Email_indexed: double (nullable = false)
 |-- Avatar_indexed: double (nullable = false)
```

```
val splits = FinalecommerceDF.randomSplit(Array(0.7, 0.3))
val train = splits(0)
val test = splits(1)
val train_rows = train.count()
val test_rows = test.count()
println("Training Rows: " + train_rows + " Testing Rows: " +
test_rows)

import org.apache.spark.ml.feature.VectorAssembler
val assembler = new
VectorAssembler().setInputCols(Array("Email_indexed",
"Avatar_indexed", "Avg_Session_Length", "Time_on_App",
"Time_on_Website",
"Length_of_Membership")).setOutputCol("features")
val training = assembler.transform(train).select($"features",
$"Yearly_Amount_Spent".alias("label"))

import org.apache.spark.ml.regression.LinearRegression
val lr = new
LinearRegression().setLabelCol("label").setFeaturesCol("featu
res").setMaxIter(10).setRegParam(0.3)
val model = lr.fit(training)
println("Model Trained!")
```

Output:

```
Model Trained!
```

```
val testing = assembler.transform(test).select($"features",
$"Yearly_Amount_Spent".alias("trueLabel"))

%sql
select prediction, trueLabel from eCommerceData
```



```
import org.apache.spark.ml.evaluation.RegressionEvaluator
val evaluator = new
RegressionEvaluator().setLabelCol("trueLabel").setPredictionC
ol("prediction").setMetricName("rmse")
val rmse = evaluator.evaluate(prediction)
println("Root Mean Square Error (RMSE): " + (rmse))
```

Output:

```
Root Mean Square Error (RMSE): 10.614461962676273
import
org.apache.spark.ml.evaluation.RegressionEvaluator
evaluator:
org.apache.spark.ml.evaluation.RegressionEvaluator =
RegressionEvaluator: uid=regEval_2c72344b2ff4,
metricName=rmse, throughOrigin=false
rmse: Double = 10.614461962676273
```

LAB 10: GraphX

AIM:

To know about spark's graphX library.

PROGRAM:

```
val df11 =
spark.read.format("csv").load("dbfs:/FileStore/shared_uploads
/mirepateljpatel01@gmail.com/edges.csv")
val df2 =
spark.read.format("csv").load("dbfs:/FileStore/shared_uploads
/mirepateljpatel01@gmail.com/vertex.csv")
val df1 =
spark.read.format("csv").load("dbfs:/FileStore/shared_uploads
/mirepateljpatel01@gmail.com/vertex-1.csv")
```

Output:

```
df11: org.apache.spark.sql.DataFrame = [_c0: string,
_c1: string ... 1 more field]
df2: org.apache.spark.sql.DataFrame = [_c0: string, _c1:
string ... 2 more fields]
df1: org.apache.spark.sql.DataFrame = [_c0: string, _c1:
string ... 1 more field]
```

```
# File location and type
file_location =
"/FileStore/shared_uploads/mirepateljpatel01@gmail.com/vertex
-1.csv"
file_type = "csv"

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

# 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", delimirepateler) \
  .load(file_location)

display(df)
```

Output:

```
_c0 _c1 _c2
1   name1   23
2   name2   34
```

```
     3   name3   26
     4   name4   29
     5   name5   31
     6   name6   36
     7   name7   41
     8   name8   21
     9   name9   54
     10  name10  36
     Showing all 10 rows.


file_location =
"/FileStore/shared_uploads/mirepateljpatel01@gmail.com/edges.
csv"
file_type = "csv"

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

# 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", delimirepateler) \
  .load(file_location)

display(df1)
```

Output:

```
     _c0 _c1 _c2
     1   2   edge1
     2   1   edge2
     4   7   edge3
     2   7   edge4
     1   6   edge5
     5   8   edge6
     2   6   edge7
     6   3   edge8
     8   5   edge9
     3   10  edge10
     9   6  edge11
     7   3   edge12

     Showing all 18 rows.


import org.apache.spark.rdd.RDD
import org.apache.spark.rdd.RDD
import org.apache.spark.graphx._
import org.apache.spark.graphx._

val vertexRDD =
sc.textFile("/FileStore/shared_uploads/mirepateljpatel01@gmai
l.com/vertex-1.csv")
val edgeRDD =
sc.textFile("/FileStore/shared_uploads/mirepateljpatel01@gmai
l.com/edges.csv")
edgeRDD.collect()
```

Output:

```
vertexRDD: org.apache.spark.rdd.RDD[String] =
/FileStore/shared_uploads/mirepateljpatel01@gmail.com/ve
rtex-1.csv MapPartitionsRDD[745] at textFile at command-
1139883235947817:1
edgeRDD: org.apache.spark.rdd.RDD[String] =
/FileStore/shared_uploads/mirepateljpatel01@gmail.com/ed
ges.csv MapPartitionsRDD[747] at textFile at command-
1139883235947817:2
res1: Array[String] = Array(1,2,edge1, 2,1,edge2,
4,7,edge3, 2,7,edge4, 1,6,edge5, 5,8,edge6, 2,6,edge7,
6,3,edge8, 8,5,edge9, 3,10,edge10, 9,6,edge11,
7,3,edge12, 1,10,edge13, 9,3,edge14, 9,3,edge15,
4,2,edge16, 8,2,edge17, 8,4,edge18)
```

```
vertexRDD.collect()
```

Output:

```
res2: Array[String] = Array(1,name1,23, 2,name2,34,
3,name3,26, 4,name4,29, 5,name5,31, 6,name6,36,
7,name7,41, 8,name8,21, 9,name9,54, 10,name10,36)
```

```
val vertices: RDD[(VertexId, (String,String))] =
vertexRDD.map{ line => val fields = line.split(",")

(fields(0).toLong, ( fields(1), fields(2)))}
vertices.collect()
```

Output:

```
vertices:
org.apache.spark.rdd.RDD[(org.apache.spark.graphx.Vertex
Id, (String, String))] = MapPartitionsRDD[748] at map at
command-1139883235947815:1
res3: Array[(org.apache.spark.graphx.VertexId, (String,
String))] = Array((1,(name1,23)), (2,(name2,34)),
(3,(name3,26)), (4,(name4,29)), (5,(name5,31)),
(6,(name6,36)), (7,(name7,41)), (8,(name8,21)),
(9,(name9,54)), (10,(name10,36)))
```

```
val edges: RDD[Edge[String]] = edgeRDD.map{ line => val
fields = line.split(",")

Edge(fields(0).toLong, fields(1).toLong,fields(2))}
edges.collect()
```

Output:

```
edges:
org.apache.spark.rdd.RDD[org.apache.spark.graphx.Edge[St
ring]] = MapPartitionsRDD[749] at map at command-
1139883235947814:1
res4: Array[org.apache.spark.graphx.Edge[String]] =
Array(Edge(1,2,edge1), Edge(2,1,edge2), Edge(4,7,edge3),
Edge(2,7,edge4), Edge(1,6,edge5), Edge(5,8,edge6),
Edge(2,6,edge7), Edge(6,3,edge8), Edge(8,5,edge9),
Edge(3,10,edge10), Edge(9,6,edge11), Edge(7,3,edge12),
Edge(1,10,edge13), Edge(9,3,edge14), Edge(9,3,edge15),
Edge(4,2,edge16), Edge(8,2,edge17), Edge(8,4,edge18))
```

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

Output:

```
default: (String, String) = (unknown,missing)
graph: org.apache.spark.graphx.Graph[(String,
String),String] =
org.apache.spark.graphx.impl.GraphImpl@41b61c27
```

```
graph.vertices.collect()
```

Output:

```
res5: Array[(org.apache.spark.graphx.VertexId, (String,
String))] = Array((4,(name4,29)), (6,(name6,36)),
(8,(name8,21)), (10,(name10,36)), (2,(name2,34)),
(1,(name1,23)), (3,(name3,26)), (7,(name7,41)),
(9,(name9,54)), (5,(name5,31)))
```

```
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
```

Output:

```
movies:
org.apache.spark.rdd.RDD[(org.apache.spark.graphx.Vertex
Id, MoviesWatched)] = ParallelCollectionRDD[762] at
parallelize at command-1139883235947812:2
```

```
val movieOuterJoinedGraph =
graph.outerJoinVertices(movies)((_,name,movies) =>
(name,movies))
movieOuterJoinedGraph:
org.apache.spark.graphx.Graph[((String, String),
Option[MoviesWatched]),String] =
org.apache.spark.graphx.impl.GraphImpl@41373b70

movieOuterJoinedGraph.vertices.map(t=>t).collect.foreach(prin
tln)
(4,((name4,29),None))
(6,((name6,36),None))
(8,((name8,21),None))
(10,((name10,36),None))
(2,((name2,34),Some(MoviesWatched(Titanic,Love))))
(1,((name1,23),Some(MoviesWatched(Toy Story 3,Kids))))
(3,((name3,26),Some(MoviesWatched(The Hangover,Comedy))))
(7,((name7,41),None))
(9,((name9,54),None))
(5,((name5,31),None))

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

Output:

```
movieOuterJoinedGraph:
org.apache.spark.graphx.Graph[((String, String),
MoviesWatched),String] =
org.apache.spark.graphx.impl.GraphImpl@2efd0da4
```

```
movieOuterJoinedGraph.vertices.map(t=>t).collect.foreach(prin
tln)
(4,((name4,29),MoviesWatched(NA,NA)))
(6,((name6,36),MoviesWatched(NA,NA)))
(8,((name8,21),MoviesWatched(NA,NA)))
(10,((name10,36),MoviesWatched(NA,NA)))
(2,((name2,34),MoviesWatched(Titanic,Love)))
(1,((name1,23),MoviesWatched(Toy Story 3,Kids)))
(3,((name3,26),MoviesWatched(The Hangover,Comedy)))
(7,((name7,41),MoviesWatched(NA,NA)))
(9,((name9,54),MoviesWatched(NA,NA)))
(5,((name5,31),MoviesWatched(NA,NA)))

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

Output:

```
tCount: org.apache.spark.graphx.VertexRDD[Int] =
VertexRDDImpl[824] at RDD at VertexRDD.scala:57
```

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

Output:

```
(4,2)
(6,2)
(8,1)
(10,0)
(2,3)
(1,1)
(3,1)
(7,1)
(9,1)
(5,0)
```

```
val iterations = 1000
```

Output:

```
iterations: Int = 1000
```

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

Output:

```
connected:
org.apache.spark.graphx.VertexRDD[org.apache.spark.graph
x.VertexId] = VertexRDDImpl[883] at RDD at
VertexRDD.scala:57
```

```
val connectedS =
graph.stronglyConnectedComponents(iterations).vertices
```

Output:

```
connectedS:
org.apache.spark.graphx.VertexRDD[org.apache.spark.graph
x.VertexId] = VertexRDDImpl[1225] at RDD at
VertexRDD.scala:57
```

```
val connByPerson = vertices.join(connected).map{
case(id,((person,age), conn))=> (conn,id,person)}
```

Output:

```
connByPerson:
org.apache.spark.rdd.RDD[(org.apache.spark.graphx.Vertex
Id, org.apache.spark.graphx.VertexId, String)] =
MapPartitionsRDD[1277] at map at command-
1139883235947839:1
```

```
val connByPersonS = vertices.join(connectedS).map{
```

```
case(id,((person,age), conn))=> (conn,id,person)}
```

Output:

```
connByPersonS:
org.apache.spark.rdd.RDD[(org.apache.spark.graphx.Vertex
Id, org.apache.spark.graphx.VertexId, String)] =
MapPartitionsRDD[1281] at map at command-
1139883235947838:1
```

```
connByPerson.collect().foreach{ case (conn,id,person)=>
println(f"Weak $conn $id $person")}
```

Output:

```
Weak 1 4 name4
Weak 1 6 name6
Weak 1 8 name8
Weak 1 10 name10
Weak 1 2 name2
Weak 1 1 name1
Weak 1 3 name3
Weak 1 7 name7
Weak 1 9 name9
Weak 1 5 name5
```

```
connByPersonS.collect().foreach{case (conn,id,person)=>
println(f"Strong $conn $id $person")}
```

Output:

```
Strong 4 4 name4
Strong 6 6 name6
Strong 5 8 name8
Strong 10 10 name10
Strong 1 2 name2
Strong 1 1 name1
Strong 3 3 name3
Strong 7 7 name7
Strong 9 9 name9
```

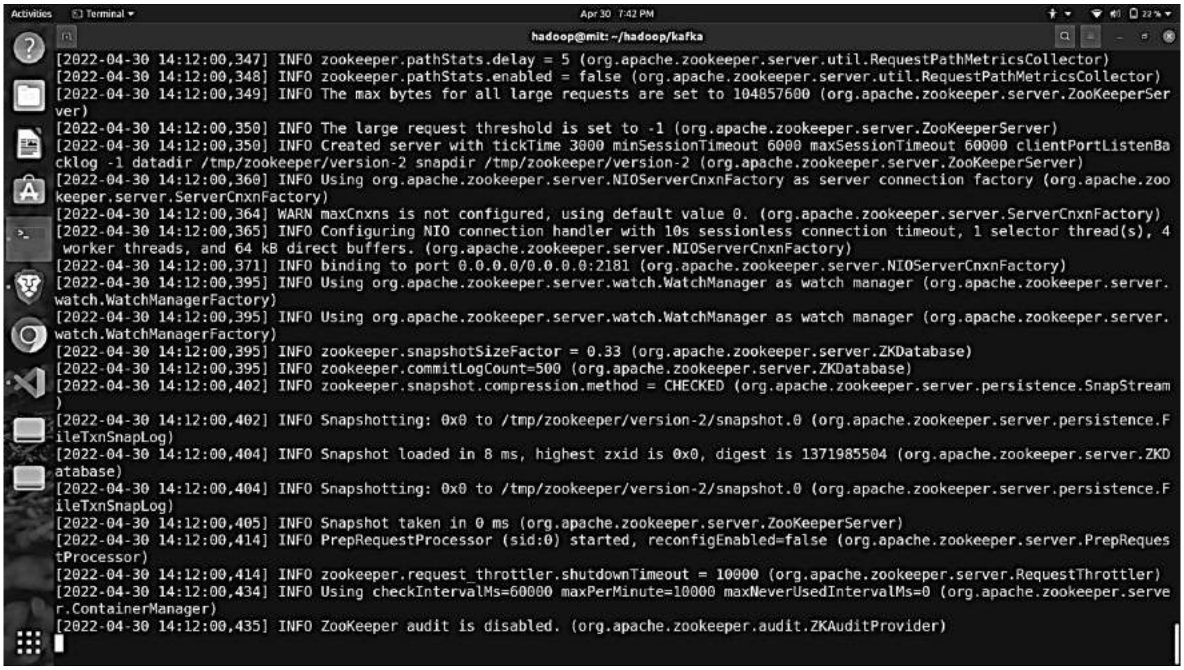
LAB 11: Kafka configuration and demo codes

AIM:

To know about kafka and how it works.

PROGRAM:

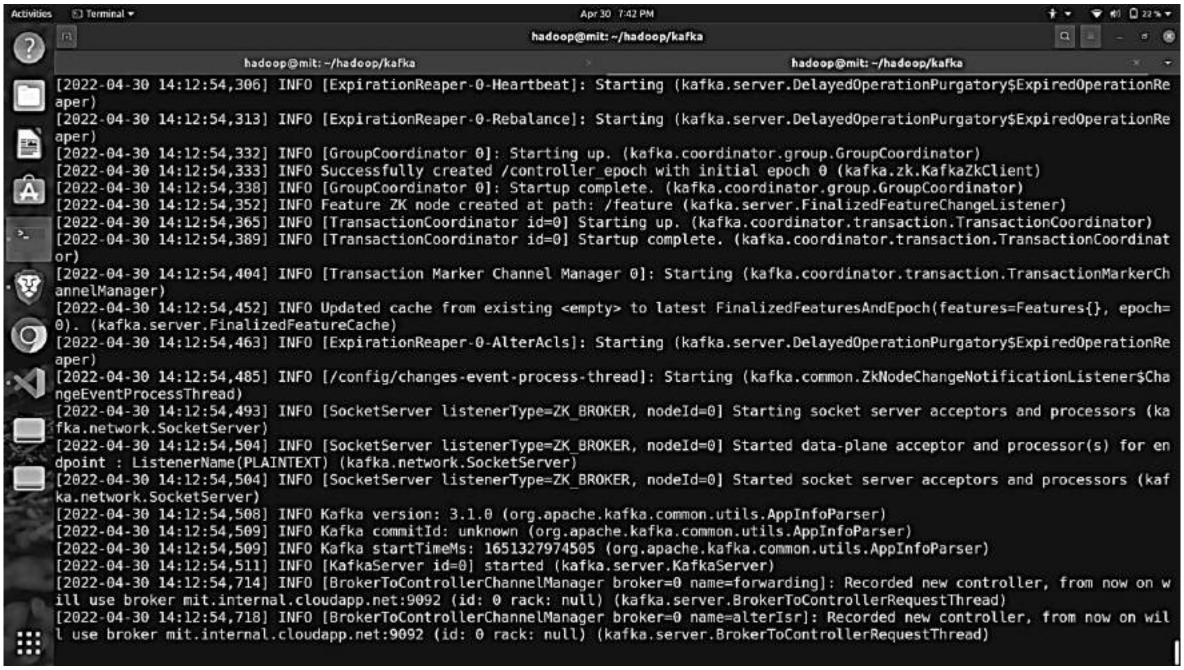
I am using Ubuntu to install kafka.
First of all, download the latest kafka sources from https://kafka.apache.org/downloads.
Then untar it. using "tar -xzf kafka*" command. Then go to that folder.
To start a Kafka server first we need a zookeeper. To start zookeeper run "bin/zookeeper-server-start.sh config/zookeeper.properties" command.
If you don't have scala installed then you get an error. To remove error run "./gradlew jar -PscalaVersion=2.13.6" command.



```
[2022-04-30 14:12:00,347] INFO zookeeper.pathStats.delay = 5 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)
[2022-04-30 14:12:00,348] INFO zookeeper.pathStats.enabled = false (org.apache.zookeeper.server.util.RequestPathMetricsCollector)
[2022-04-30 14:12:00,349] INFO The max bytes for all large requests are set to 104857600 (org.apache.zookeeper.server.ZooKeeperSer
ver)
[2022-04-30 14:12:00,350] INFO The large request threshold is set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2022-04-30 14:12:00,350] INFO Created server with tickTime 3000 minSessionTimeout 6000 maxSessionTimeout 60000 clientPortListenBa
cklog -1 datadir /tmp/zookeeper/version-2 snapdir /tmp/zookeeper/version-2 (org.apache.zookeeper.server.ZooKeeperServer)
[2022-04-30 14:12:00,360] INFO Using org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory (org.apache.zoo
keeper.server.ServerCnxnFactory)
[2022-04-30 14:12:00,364] WARN maxCnxns is not configured, using default value 0. (org.apache.zookeeper.server.ServerCnxnFactory)
[2022-04-30 14:12:00,365] INFO Configuring NIO connection handler with 10s sessionless connection timeout, 1 selector thread(s), 4
 worker threads, and 64 kB direct buffers. (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2022-04-30 14:12:00,371] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2022-04-30 14:12:00,395] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.
watch.WatchManagerFactory)
[2022-04-30 14:12:00,395] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.
watch.WatchManagerFactory)
[2022-04-30 14:12:00,395] INFO zookeeper.snapshotSizeFactor = 0.33 (org.apache.zookeeper.server.ZKDatabase)
[2022-04-30 14:12:00,395] INFO zookeeper.commitLogCount=500 (org.apache.zookeeper.server.ZKDatabase)
[2022-04-30 14:12:00,402] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.SnapStream
)
[2022-04-30 14:12:00,402] INFO Snapshotting: 0x0 to /tmp/zookeeper/version-2/snapshot.0 (org.apache.zookeeper.server.persistence.F
ileTxnSnapLog)
[2022-04-30 14:12:00,404] INFO Snapshot loaded in 8 ms, highest zxid is 0x0, digest is 1371985504 (org.apache.zookeeper.server.ZKD
atabase)
[2022-04-30 14:12:00,404] INFO Snapshotting: 0x0 to /tmp/zookeeper/version-2/snapshot.0 (org.apache.zookeeper.server.persistence.F
ileTxnSnapLog)
[2022-04-30 14:12:00,405] INFO Snapshot taken in 0 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2022-04-30 14:12:00,414] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepReques
tProcessor)
[2022-04-30 14:12:00,414] INFO zookeeper.request_throttler.shutdownTimeout = 10000 (org.apache.zookeeper.server.RequestThrottler)
[2022-04-30 14:12:00,434] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.serve
r.ContainerManager)
[2022-04-30 14:12:00,435] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
```

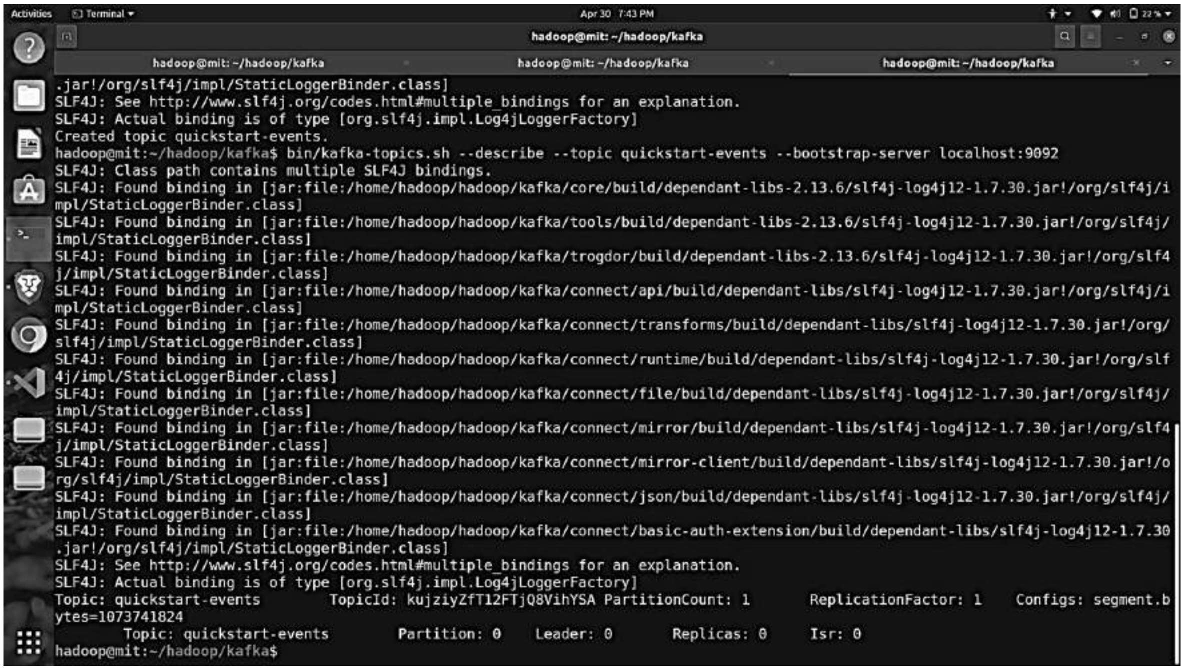
There are some warnings which we can ignore. And we can know all information about the zookeeper server.

Now open new terminal and start kafka server using "bin/kafka-server-start.sh config/server.properties"


Activities
Terminal
Apr 30 7:42 PM
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka

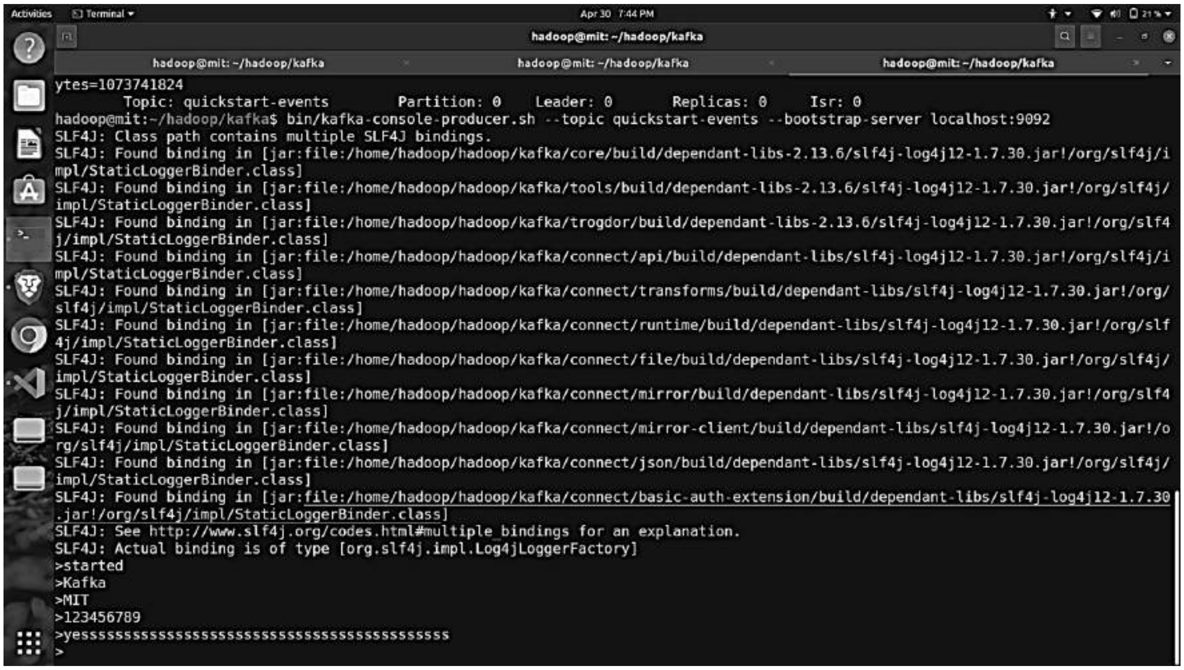
```
[2022-04-30 14:12:54,306] INFO [ExpirationReaper-0-Heartbeat]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationRe
aper)
[2022-04-30 14:12:54,313] INFO [ExpirationReaper-0-Rebalance]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationRe
aper)
[2022-04-30 14:12:54,332] INFO [GroupCoordinator 0]: Starting up. (kafka.coordinator.group.GroupCoordinator)
[2022-04-30 14:12:54,333] INFO Successfully created /controller_epoch with initial epoch 0 (kafka.zk.KafkaZkClient)
[2022-04-30 14:12:54,338] INFO [GroupCoordinator 0]: Startup complete. (kafka.coordinator.group.GroupCoordinator)
[2022-04-30 14:12:54,352] INFO Feature ZK node created at path: /feature (kafka.server.FinalizedFeatureChangeListener)
[2022-04-30 14:12:54,365] INFO [TransactionCoordinator id=0] Starting up. (kafka.coordinator.transaction.TransactionCoordinator)
[2022-04-30 14:12:54,389] INFO [TransactionCoordinator id=0] Startup complete. (kafka.coordinator.transaction.TransactionCoordinat
or)
[2022-04-30 14:12:54,404] INFO [Transaction Marker Channel Manager 0]: Starting (kafka.coordinator.transaction.TransactionMarkerCh
annelManager)
[2022-04-30 14:12:54,452] INFO Updated cache from existing <empty> to latest FinalizedFeaturesAndEpoch(features=Features{}, epoch=
0). (kafka.server.FinalizedFeatureCache)
[2022-04-30 14:12:54,463] INFO [ExpirationReaper-0-AlterAcls]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationRe
aper)
[2022-04-30 14:12:54,485] INFO [/config/changes-event-process-thread]: Starting (kafka.common.ZkNodeChangeNotificationListener$Cha
ngeEventProcessThread)
[2022-04-30 14:12:54,493] INFO [SocketServer listenerType=ZK_BROKER, nodeId=0] Starting socket server acceptors and processors (ka
fka.network.SocketServer)
[2022-04-30 14:12:54,504] INFO [SocketServer listenerType=ZK_BROKER, nodeId=0] Started data-plane acceptor and processor(s) for en
dpoint : ListenerName(PLAINTEXT) (kafka.network.SocketServer)
[2022-04-30 14:12:54,504] INFO [SocketServer listenerType=ZK_BROKER, nodeId=0] Started socket server acceptors and processors (kaf
ka.network.SocketServer)
[2022-04-30 14:12:54,508] INFO Kafka version: 3.1.0 (org.apache.kafka.common.utils.AppInfoParser)
[2022-04-30 14:12:54,509] INFO Kafka commitId: unknown (org.apache.kafka.common.utils.AppInfoParser)
[2022-04-30 14:12:54,509] INFO Kafka startTimeMs: 1651327974505 (org.apache.kafka.common.utils.AppInfoParser)
[2022-04-30 14:12:54,511] INFO [KafkaServer id=0] started (kafka.server.KafkaServer)
[2022-04-30 14:12:54,714] INFO [BrokerToControllerChannelManager broker=0 name=forwarding]: Recorded new controller, from now on w
ill use broker mit.internal.cloudapp.net:9092 (id: 0 rack: null) (kafka.server.BrokerToControllerRequestThread)
[2022-04-30 14:12:54,718] INFO [BrokerToControllerChannelManager broker=0 name=alterIsr]: Recorded new controller, from now on wil
l use broker mit.internal.cloudapp.net:9092 (id: 0 rack: null) (kafka.server.BrokerToControllerRequestThread)
```

Now our kafka server is up and running. We can create a producer and consumer but before that we need to create a topic.
To create a new topic run “bin/kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092” command.


Activities
Terminal
Apr 30 7:43 PM
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka

```
.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Created topic quickstart-events.
hadoop@mit:~/hadoop/kafka$ bin/kafka-topics.sh --describe --topic quickstart-events --bootstrap-server localhost:9092
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/core/build/dependant-libs-2.13.6/slf4j-log4j12-1.7.30.jar!/org/slf4j/i
mpl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/tools/build/dependant-libs-2.13.6/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/trogdor/build/dependant-libs-2.13.6/slf4j-log4j12-1.7.30.jar!/org/slf4
j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/api/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/i
mpl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/transforms/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/runtime/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf
4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/file/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/mirror/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4
j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/mirror-client/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/o
rg/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/json/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/basic-auth-extension/build/dependant-libs/slf4j-log4j12-1.7.30
.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Topic: quickstart-events        TopicId: kujziyZfT12FTjQ8VihYSA PartitionCount: 1       ReplicationFactor: 1    Configs: segment.b
ytes=1073741824
        Topic: quickstart-events        Partition: 0    Leader: 0       Replicas: 0     Isr: 0
hadoop@mit:~/hadoop/kafka$
```

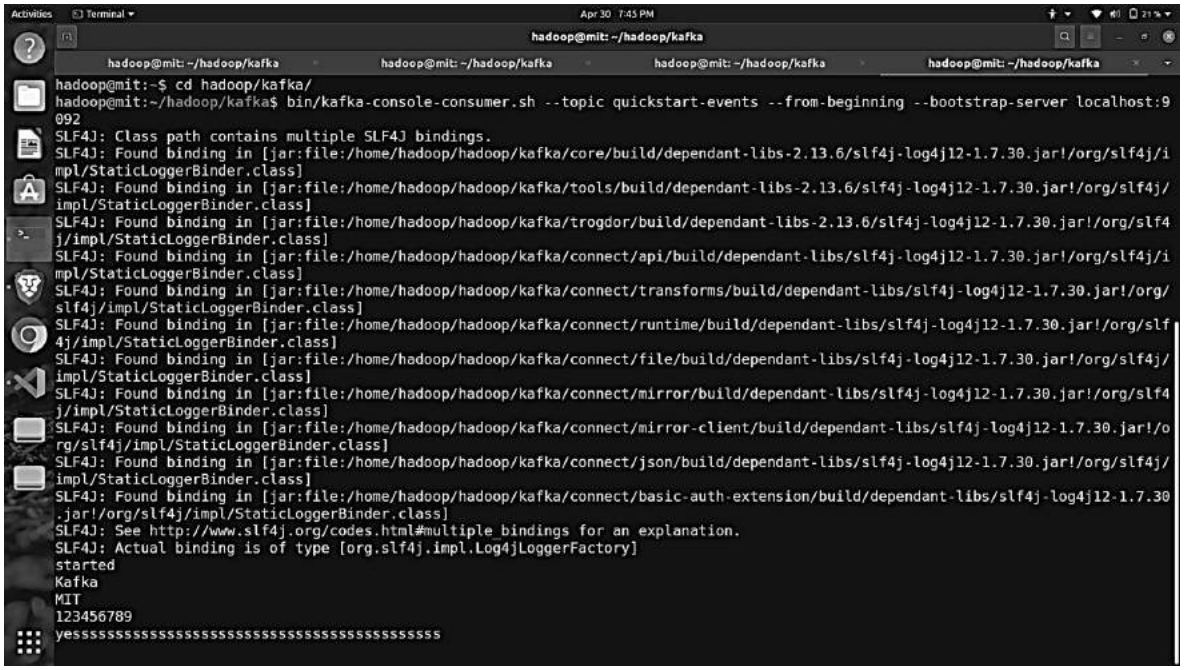
To know all information about a topic run “bin/kafka-topics.sh --describe --topic quickstart-events --bootstrap-server localhost:9092” command.
Now we can start producing. To start producer run “bin/kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092” command.



Activities
Terminal
Apr 30 7:44 PM
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka

```
ytes=1073741824
        Topic: quickstart-events        Partition: 0    Leader: 0       Replicas: 0     Isr: 0
hadoop@mit:~/hadoop/kafka$ bin/kafka-console-producer.sh --topic quickstart-events --bootstrap-server localhost:9092
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/core/build/dependant-libs-2.13.6/slf4j-log4j12-1.7.30.jar!/org/slf4j/i
mpl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/tools/build/dependant-libs-2.13.6/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/trogdor/build/dependant-libs-2.13.6/slf4j-log4j12-1.7.30.jar!/org/slf4
j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/api/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/i
mpl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/transforms/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/runtime/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf
4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/file/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/mirror/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4
j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/mirror-client/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/o
rg/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/json/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/basic-auth-extension/build/dependant-libs/slf4j-log4j12-1.7.30
.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
>started
>Kafka
>MIT
>123456789
>yessssssssssssssssssssssssssssssssssssssssssss
>
```

To get that message we need to start consumer. To start, the consumer runs the “bin/kafka-console-consumer.sh --topic quickstart-events --from-beginning --bootstrap-server localhost:9092” command.



Activities
Terminal
Apr 30 7:45 PM
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka
hadoop@mit: ~/hadoop/kafka

```
hadoop@mit:~$ cd hadoop/kafka/
hadoop@mit:~/hadoop/kafka$ bin/kafka-console-consumer.sh --topic quickstart-events --from-beginning --bootstrap-server localhost:9
092
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/core/build/dependant-libs-2.13.6/slf4j-log4j12-1.7.30.jar!/org/slf4j/i
mpl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/tools/build/dependant-libs-2.13.6/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/trogdor/build/dependant-libs-2.13.6/slf4j-log4j12-1.7.30.jar!/org/slf4
j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/api/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/i
mpl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/transforms/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/
slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/runtime/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf
4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/file/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/mirror/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4
j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/mirror-client/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/o
rg/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/json/build/dependant-libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/kafka/connect/basic-auth-extension/build/dependant-libs/slf4j-log4j12-1.7.30
.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
started
Kafka
MIT
123456789
yessssssssssssssssssssssssssssssssssssssssssss
```

LAB 12: MongoDB configuration and demo codes

AIM:

To know about spark's graphX library.

PROGRAM:

First of all, make an account in mongodb atlas. Then create a mongodb server. Then, in network access select all IP addresses so that we can connect to that from anywhere.

Start databrick's computing cluster then install python mongodb package.

```
!pip install pymongo[srv]

import pymongo
client = pymongo.MongoClient("{connection string from mongo
atlas}? retryWrites=true&w=majority")

for i in client.list_databases():
    print(i)
```

Output:

```
{'name': 'Sample', 'sizeOnDisk': 73728, 'empty': False}
{'name': 'sample_airbnb', 'sizeOnDisk': 57188352,
'empty': False}
{'name': 'sample_analytics', 'sizeOnDisk': 9695232,
'empty': False}
{'name': 'sample_geospatial', 'sizeOnDisk': 1425408,
'empty': False}
```

```
db = client["Sample"]
db.list_collection_names()
```

Output:

```
['samplecollection']
```

```
col = db["samplecollection"]

for i in col.find():
    print(i)
```

Output:

```
{'_id': ObjectId('625d3f17fc32e7066a3b870e'), 'name':
'Dwalin', 'age': 169}
{'_id': ObjectId('625d3f17fc32e7066a3b870f'), 'name':
'Bilbo Baggins', 'age': 50}
{'_id': ObjectId('625d3f17fc32e7066a3b8709'), 'name':
'Balin', 'age': 178}
{'_id': ObjectId('625d3f17fc32e7066a3b870a'), 'name':
'Kili', 'age': 77}
{'_id': ObjectId('625d3f17fc32e7066a3b870b'), 'name':
'Fili', 'age': 82}
{'_id': ObjectId('625d3f17fc32e7066a3b870c'), 'name':
'Bombur'}
{'_id': ObjectId('625d3f17fc32e7066a3b8710'), 'name':
'Gandalf', 'age': 1000}
{'_id': ObjectId('625d3f17fc32e7066a3b870d'), 'name':
'Thorin', 'age': 195}
```

```
for i in col.find({"name": { "$eq":"Oin" }}):
    print(i)
```

Output:

```
    {'_id': ObjectId('625d3f17fc32e7066a3b8708'), 'name':
    'Oin', 'age': 167}
    {'_id': ObjectId('625d470afc32e7066a3b872a'), 'name':
    'Oin', 'age': 167}
    {'_id': ObjectId('625d47bcfc32e7066a3b8735'), 'name':
    'Oin', 'age': 167}
```

```
import pandas as pd
df = pd.DataFrame(col.find())
df.head()
```

Output:

```
       _id                         name            age
    0 625d3f17fc32e7066a3b870e     Dwalin          169.0
    1 625d3f17fc32e7066a3b870f     Bilbo Baggins   50.0
    2 625d3f17fc32e7066a3b8709     Balin           178.0
    3 625d3f17fc32e7066a3b870a     Kili            77.0
    4 625d3f17fc32e7066a3b870b     Fili            82.0
```

LAB 13: Project work (Code + Paper/Report + PPT)