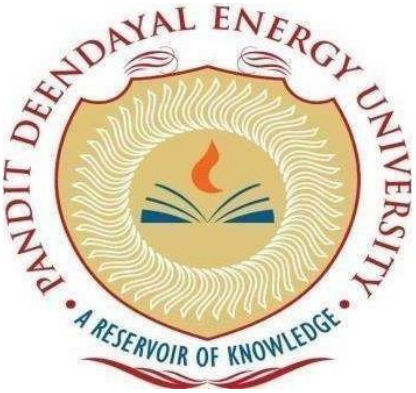
Pandit Deendayal Energy University, Gandhinagar

School Of Technology

Department of Computer Science and Engineering



Name: SHIVRAJ NAKUM

Enrolment No: 21BCP125

Division: 2

Batch: G4

Course Name: Big Data Analytics Lab

Course Code: 23CP309P

Faculty: Soham Vyas Sir

TABLE OF CONTENTS

PRACTICAL-1 Basic of Python

Cmd 1

```
print("21BCP125")
```

```
21BCP125
```

Command took 0.08 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

Cmd 2

```
print("hi")
```

```
hi
```

Command took 0.13 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

Cmd 3

```
def factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n*factorial(n-1)

ans = factorial(5)
print(ans)
```

```
120
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

Cmd 4

```
num = 11
# If given number is greater than 1
if num > 1:
    # Iterate from 2 to n / 2
    for i in range(2, int(num/2)+1):
        # If num is divisible by any number between
        # 2 and n / 2, it is not prime
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
    else:
        print(num, "is a prime number")
else:
    print(num, "is not a prime number")

```

```
11 is a prime number
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

Cmd 5

```
n = 10
num1 = 0
num2 = 1
next_number = num2
count = 1

while count <= n:
    print(next_number, end=" ")
    count += 1
    num1, num2 = num2, next_number
    next_number = num1 + num2
print()

```

```
1 2 3 5 8 13 21 34 55 89
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

Cmd 6

```
print("21BCP125")
```

```
21BCP125
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

Cmd 7

```
def reverse(s):
    if len(s) == 0:
        return s
    else:
        return reverse(s[1:]) + s[0]


s = "Hello"

print("The original string is : ", end="")
print(s)

print("The reversed string is : ", end="")
print(reverse(s))

```

```
The original string is : Hello
The reversed string is : olleH
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

```
def count_statistics(text):
    # Counting words
    word_count = len(text.split())

    # Counting characters (including spaces)
    character_count = len(text)

    # Counting spaces
    space_count = text.count(' ')

    return word_count, character_count, space_count

# Example usage
paragraph = "This is a sample paragraph with some words and spaces."

word_count, character_count, space_count = count_statistics(paragraph)

print("Word count:", word_count)
print("Character count (including spaces):", character_count)
print("Space count:", space_count)

```

```
Word count: 10
Character count (including spaces): 54
Space count: 9
```

Command took 0.18 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

```
%fs

ls /FileStore/tables
```

Table +

	path	name	size	modificationTime
1	dbfs:/FileStore/tables/Iris.csv	Iris.csv	5107	1704862655000
2	dbfs:/FileStore/tables/data_1-1.json	data_1-1.json	223	1708439851000
3	dbfs:/FileStore/tables/data_1-2.json	data_1-2.json	223	1708440107000
4	dbfs:/FileStore/tables/data_1.json	data_1.json	223	1708439474000

4 rows | 1.62 seconds runtime Refreshed now

Command took 1.62 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

Cmd 10

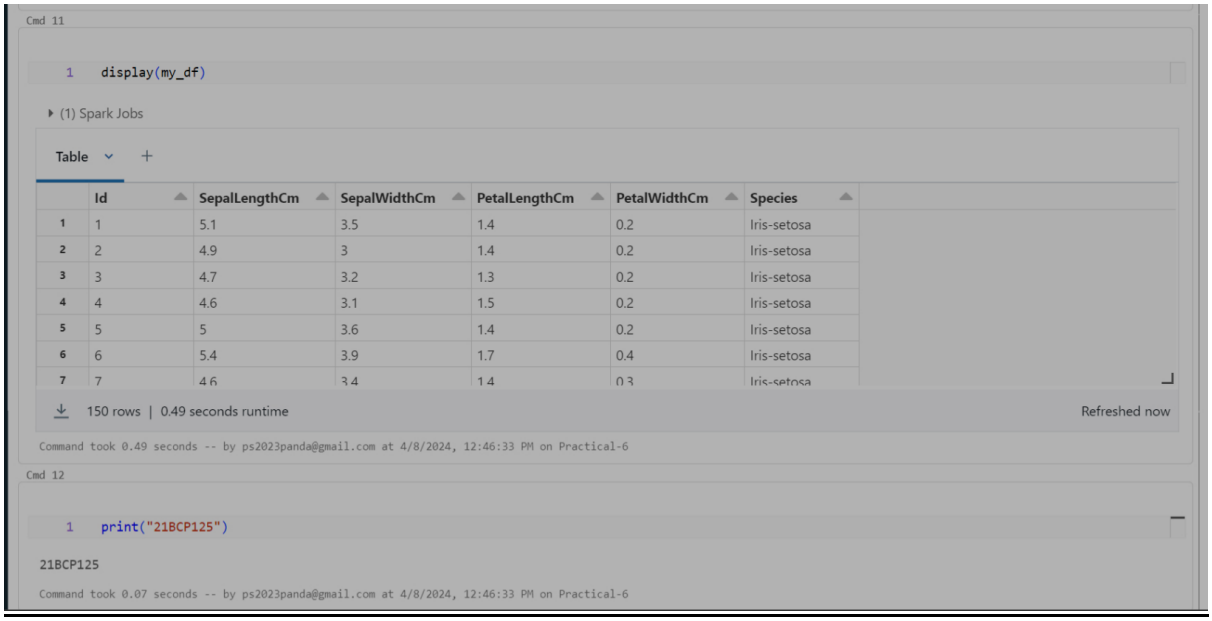
```
my_df = spark.read.format("csv").option("inferSchema","true").option("header","true").load("/FileStore/tables/Iris.csv")


```

▸ (2) Spark Jobs

▸ my_df: pyspark.sql.dataframe.DataFrame = [Id: integer, SepalLengthCm: double ... 4 more fields]

Command took 1.79 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6



Cmd 11

```
1  display(my_df)
```

▸ (1) Spark Jobs

Table +

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	1	5.1	3.5	1.4	0.2	Iris-setosa
2	2	4.9	3	1.4	0.2	Iris-setosa
3	3	4.7	3.2	1.3	0.2	Iris-setosa
4	4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	5	3.6	1.4	0.2	Iris-setosa
6	6	5.4	3.9	1.7	0.4	Iris-setosa
7	7	4.6	3.4	1.4	0.3	Iris-setosa

150 rows | 0.49 seconds runtime Refreshed now

Command took 0.49 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

Cmd 12

```
1  print("21BCP125")
```

21BCP125

Command took 0.07 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:46:33 PM on Practical-6

PRACTICAL-2 Basics of Scala

Cmd 1

```
%python
print("hi")
```

```
hi
```

Command took 0.14 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 2

```
print("21BCP125")
```

```
21BCP125
```

Command took 0.34 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 3

```
println("hey Scala")
```

```
hey Scala
```

Command took 0.34 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 4

```
var value= 5
```

```
value: Int = 5
```

Command took 0.34 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 5

```
println("hii"+value)
```

```
hii5
```

Command took 0.32 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 6

Cmd 6

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

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

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 7

```
num.head
```

```
res16: Int = 1
```

Command took 0.38 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 8

```
num.tail
```

```
res17: List[Int] = List(2, 3, 4)
```

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 9

```
num.sum
```

```
res18: Int = 10
```

Command took 0.32 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 10

```
print("21BCP125")
```

```
21BCP125
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 11

```
1  num.take(2)
```

```
res20: List[Int] = List(1, 2)
```

Command took 0.39 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 12

```
1  var temp = List(1,1,1,1,2,2,2,2,1)
```

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

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 13

```
1  temp.distinct
```

```
res21: List[Int] = List(1, 2)
```

Command took 0.31 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 14

```
1  temp(0)
```

```
res22: Int = 1
```

Command took 0.32 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 15

```
1  temp(5)
```

```
res23: Int = 2
```

Command took 0.30 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 16

```
1  print("21BCP125")
```

```
21BCP125
```

Command took 0.66 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 17

```
1  num(10)
```

```
⊞IndexOutOfBoundsException: 10
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:49:54 PM on Practical-6

Cmd 18

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

```
⊞
command-1093067116575058:1: error: value update is not a member of List[Int]
num(1)=10
^
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:10:46 PM on Practical-6

Cmd 19

```
1  num.length
```

```
res27: Int = 4
```

Command took 0.80 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:10:48 PM on Practical-6

Cmd 20

```
1  temp.size
```

```
res28: Int = 9
```

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:10:51 PM on Practical-6

Cmd 21

```
1  num.reverse
```

```
res29: List[Int] = List(4, 3, 2, 1)
```

Command took 0.33 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:10:53 PM on Practical-6

Cmd 22

```
1  num.min
```

```
res30: Int = 1
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:10:54 PM on Practical-6

Cmd 23

```
1  num.max
```

```
res31: Int = 4
```

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:10:57 PM on Practical-6

Cmd 24

```
1  num.isEmpty
```

```
res32: Boolean = false
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:10:58 PM on Practical-6

Cmd 25

```
1  var empty_list = List()
```

```
empty_list: List[Nothing] = List()
```

Command took 0.43 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:00 PM on Practical-6

Cmd 26

```
1  empty_list.isEmpty
```

```
res33: Boolean = true
```

Command took 0.30 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:02 PM on Practical-6

Cmd 27

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

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

Cmd 28

```
1  var lang = Array("scala","python","java")
```

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

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:04 PM on Practical-6

Cmd 29

```
1  lang.head
```

```
res34: String = scala
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:05 PM on Practical-6

Cmd 30

```
1  lang.tail
```

```
res35: Array[String] = Array(python, java)
```

Command took 0.41 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:07 PM on Practical-6

Cmd 31

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

Command took 0.81 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:09 PM on Practical-6

Cmd 32

```
1  number
```

```
res37: Array[Int] = Array(1, 10, 3, 4, 5, 6, 7, 8, 9)
```

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:10 PM on Practical-6

Cmd 33

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

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

Command took 0.37 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:12 PM on Practical-6

Cmd 34

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

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

Command took 0.44 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:14 PM on Practical-6

Cmd 35

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

```
res38: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW)
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:15 PM on Practical-6

Cmd 36

```
1  cars.append("Jaguar")
```

Command took 0.25 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:16 PM on Practical-6

Cmd 37

```
1  cars.length
```

```
res40: Int = 2
```

Command took 0.31 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:17 PM on Practical-6

Cmd 38

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

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:20 PM on Practical-6

Cmd 39

```
1  cars
```

```
res42: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW)
```

Command took 0.41 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:20 PM on Practical-6

Cmd 40

```
1  cars.insert(1,"bentley")
```

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:22 PM on Practical-6

Cmd 41

```
1  cars
```

```
res44: scala.collection.mutable.ArrayBuffer[String] = ArrayBuffer(BMW, bentley)
```

Command took 0.30 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:22 PM on Practical-6

Cmd 42

```
1  //Transform and Map
2  num
```

```
res45: List[Int] = List(1, 2, 3, 4)
```

Command took 0.23 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:24 PM on Practical-6

Cmd 43

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

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

Command took 0.67 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:25 PM on Practical-6

Cmd 44

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

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

Command took 0.42 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:26 PM on Practical-6

Cmd 45

```
1  var fruits = List("Orange","Banana","Apple")
```

```
fruits: List[String] = List(Orange, Banana, Apple)
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:26 PM on Practical-6

Cmd 46

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

```
res46: List[(String, Int)] = List((Orange,6), (Banana,6), (Apple,5))
```

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:27 PM on Practical-6

Cmd 47

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

```
res47: List[String] = List(Orange, Banana)
```

Command took 0.38 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:27 PM on Practical-6

Cmd 48

```
1  val ratings = List(2.4,5.6,8.9,7.3)
```

```
ratings: List[Double] = List(2.4, 5.6, 8.9, 7.3)
```

Command took 0.32 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:28 PM on Practical-6

Cmd 49

```
1  //Multiply by 10, filter more than 75
2  //filter and save between 60 to 75 , convert back to 1-10
```

Command took 0.21 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:28 PM on Practical-6

Cmd 50

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

```
alpha: List[Double] = List(24.0, 56.0, 89.0, 73.0)
```

Command took 0.31 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:32 PM on Practical-6

Cmd 51

```
1  val beta = alpha.filter(x => (x>75))
```

```
beta: List[Double] = List(89.0)
```

Command took 0.27 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:33 PM on Practical-6

Cmd 52

```
1  val gamma = alpha.filter(x => (x>60 && x<75))
```

```
gamma: List[Double] = List(73.0)
```

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:36 PM on Practical-6

Cmd 53

```
1  val fa = gamma.map(x => (x/10))
```

```
fa: List[Double] = List(7.3)
```

Command took 0.39 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:38 PM on Practical-6

Cmd 54

```
1  def add(a:Double=100, b:Double=200):Double =
2  {
3    var sum: Double = 0
4    sum = a+b
5    return sum
6  }
7  println("Sum:"+add(10,20))
```

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

Command took 0.34 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:39 PM on Practical-6

Cmd 55

```
var x=1
if(x<3)
{
  println("less than 3")
}
else
{
  println("greater than 3")
}
```

```
less than 3
x: Int = 1
```

Command took 0.32 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:41 PM on Practical-6

Cmd 56

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

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

Command took 0.34 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:42 PM on Practical-6

Cmd 57

```
object MatrixMultiplication {
  def main(args: Array[String]): Unit = {
    // Define two matrices
    val matrix1 = Array(Array(1, 2, 3), Array(4, 5, 6))
    val matrix2 = Array(Array(7, 8), Array(9, 10), Array(11, 12))

    // Perform matrix multiplication
    val resultMatrix = multiplyMatrices(matrix1, matrix2)

    // Print the result matrix
    for (row <- resultMatrix) {
      println(row.mkString(" "))
    }
  }

  def multiplyMatrices(matrix1: Array[Array[Int]], matrix2: Array[Array[Int]]): Array[Array[Int]] = {
    val rows1 = matrix1.length
    val cols1 = matrix1(0).length
    val rows2 = matrix2.length
    val cols2 = matrix2(0).length

    // Check if matrices can be multiplied
    if (cols1 != rows2) {
      throw new IllegalArgumentException("Matrices cannot be multiplied: incompatible dimensions.")
    }

    // Initialize the result matrix with zeros
    val resultMatrix = Array.ofDim[Int](rows1, cols2)

    // Perform matrix multiplication
    for (i <- 0 until rows1) {
      for (j <- 0 until cols2) {
        for (k <- 0 until cols1) {
          resultMatrix(i)(j) += matrix1(i)(k) * matrix2(k)(j)
        }
      }
    }

    resultMatrix
  }
}
```

```
defined object MatrixMultiplication
```

Command took 0.98 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:11:45 PM on Practical-6

PRACTICAL -3 Transformation Function

Cmd 1

```
%python

print("hello world")
```

```
hello world
```

Command took 0.10 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 2

```
print("21BCP125")
```

```
21BCP125
```

Command took 5.24 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 3

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

```
a: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[258] at parallelize at command-852948785906838:1
```

Command took 0.37 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 4

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

▸ (1) Spark Jobs

```
b: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[259] at map at command-852948785906839:1
res1: Array[(String, Int)] = Array((A,1), (B,1), (C,1), (D,1))
```

Command took 0.60 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 5

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

▸ (1) Spark Jobs

```
b: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[260] at map at command-852948785906840:1
res2: Array[(String, Int)] = Array((A,1), (B,1), (C,1), (D,1))
```

Command took 1.01 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 6

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

▸ (1) Spark Jobs

```
a: org.apache.spark.rdd.RDD[List[Int]] = MapPartitionsRDD[262] at map at command-852948785906842:1
res3: Array[List[Int]] = Array(List(1, 2, 3), List(2, 3, 4), List(3, 4, 5), List(4, 5, 6), List(5, 6, 7))
```

Command took 0.43 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 7

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

▸ (1) Spark Jobs

```
b: org.apache.spark.rdd.RDD[(List[Int], Int)] = MapPartitionsRDD[263] at map at command-852948785906841:1
res4: Array[(List[Int], Int)] = Array((List(1, 2, 3),3), (List(2, 3, 4),3), (List(3, 4, 5),3), (List(4, 5, 6),3), (List(5, 6, 7),3))
```

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 8

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

▸ (1) Spark Jobs

```
a: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[265] at flatMap at command-852948785906843:1
res5: Array[Int] = Array(1, 2, 3, 2, 3, 4, 3, 4, 5, 4, 5, 6, 5, 6, 7)
```

Command took 0.40 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 9

```
1  print("21BCP125")
```

```
21BCP125
```

Command took 0.24 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 10

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

▸ (1) Spark Jobs

```
rdda: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[266] at parallelize at command-852948785906844:1
res7: Array[String] = Array(aaaa)
```

Command took 0.40 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 11

```
1  rdda.filter(_.contains("a")).collect
```

▸ (1) Spark Jobs

```
res8: Array[String] = Array(aaaa)
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 12

```
1  val a = sc.parallelize(List(("Mumbai",4000,List("ask","stop","stop")),("Delhi",2000,List("ask1","stop1"))))
2  a.collect
```

▸ (1) Spark Jobs

```
a: org.apache.spark.rdd.RDD[(String, Int, List[String])] = ParallelCollectionRDD[269] at parallelize at command-852948785906846:1
res9: Array[(String, Int, List[String])] = Array((Mumbai,4000,List(ask, stop, stop)), (Delhi,2000,List(ask1, stop1)))
```

```
1  a.filter(_._3.contains("stop")).collect
```

▸ (1) Spark Jobs

```
res10: Array[(String, Int, List[String])] = Array((Mumbai,4000,List(ask, stop, stop)))
```

Command took 0.53 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 14

```
1  a.filter(_._1.contains("ai")).collect
```

▸ (1) Spark Jobs

```
res11: Array[(String, Int, List[String])] = Array((Mumbai,4000,List(ask, stop, stop)))
```

Command took 0.33 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 15

```
1  a.filter(_._2<=(3000)).collect
```

▸ (1) Spark Jobs

```
res12: Array[(String, Int, List[String])] = Array((Delhi,2000,List(ask1, stop1)))
```

Command took 0.43 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 16

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

▸ (1) Spark Jobs

```
res13: Array[(String, Int, List[String])] = Array((Mumbai,4000,List(ask, stop, stop)))
```

Command took 0.46 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 17

```
1  a.filter(x => x._2>3000 && x._2<6000).collect
```

▸ (1) Spark Jobs

```
res14: Array[(String, Int, List[String])] = Array((Mumbai,4000,List(ask, stop, stop)))
```

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 18

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

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[276] at parallelize at command-852948785906852:1
```

Command took 0.47 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

```
1  a.collect
```

▸ (1) Spark Jobs

```
res15: 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, 7
1, 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,
```

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 20

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

▸ (1) Spark Jobs

```
res16: Array[Int] = Array(3, 6, 7, 8, 14, 24, 25, 27, 28, 32, 39, 45, 53, 59, 67, 73, 74, 91, 93, 96, 98, 100, 103, 105, 111, 127, 130, 136, 138, 141, 1
42, 143, 145, 148, 154, 157, 159, 161, 164, 165, 171, 172, 173, 175, 189, 192, 199, 202, 204, 215, 221, 224, 225, 233, 244, 247, 256, 264, 270, 274, 27
6, 296, 305, 308, 312, 313, 314, 317, 318, 319, 320, 327, 331, 332, 337, 342, 351, 360, 364, 380, 393, 396, 405, 408, 415, 417, 422, 423, 440, 444, 445,
448, 458, 459, 461, 464, 483, 487, 499, 503, 514, 526, 527, 528, 530, 550, 551, 553, 561, 569, 580, 582, 584, 589, 591, 604, 606, 609, 611, 616, 617, 62
7, 629, 639, 643, 653, 662, 674, 678, 689, 693, 698, 711, 715, 717, 719, 726, 733, 737, 739, 740, 741, 750, 761, 764, 769, 772, 777, 779, 780, 782, 794,
803, 809, 821, 824, 825, 829, 832, 835, 838, 849, 853, 857, 858, 869, 878, 893, 895, 896, 900, 904, 907, 912, 915, 918, 920, 940, 942, 947, 950, 953, 95
4, 956, 959, 966, 968, 972, 978, 1000)
```

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 21

```
1  val temp = a.sample(false, 0.1).collect
2  temp.length
```

▸ (1) Spark Jobs

```
temp: Array[Int] = Array(1, 28, 30, 33, 38, 43, 51, 59, 69, 72, 85, 87, 90, 150, 151, 171, 172, 184, 206, 221, 230, 236, 240, 246, 257, 273, 274, 288, 2
93, 314, 317, 320, 331, 335, 349, 352, 354, 364, 369, 374, 384, 387, 413, 432, 438, 447, 452, 458, 472, 482, 487, 496, 497, 510, 517, 527, 546, 547, 54
9, 553, 561, 570, 571, 594, 607, 608, 611, 622, 638, 655, 660, 665, 705, 709, 731, 742, 746, 761, 762, 765, 776, 785, 803, 805, 810, 811, 816, 818, 831,
836, 842, 846, 853, 871, 880, 901, 911, 917, 923, 936, 946, 956, 960, 971, 989, 997)
res17: Int = 106
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

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

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[279] at parallelize at command-852948785906856:1
```

Command took 0.30 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 23

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

▸ (1) Spark Jobs

```
res18: Array[Int] = Array(2, 1)
```

Command took 0.50 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 24

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

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[281] at parallelize at command-852948785906858:1
```

Command took 0.28 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 25

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

```
b: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[282] at parallelize at command-852948785906859:1
```

Command took 0.26 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:28 PM on Practical-6

Cmd 26

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

▸ (1) Spark Jobs

```
res19: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 5, 6, 7, 8, 9, 10)
```

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:29 PM on Practical-6

Cmd 27

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

▸ (1) Spark Jobs

```
res20: Array[Int] = Array(5, 6, 7)
```

Command took 0.87 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:29 PM on Practical-6

Cmd 28

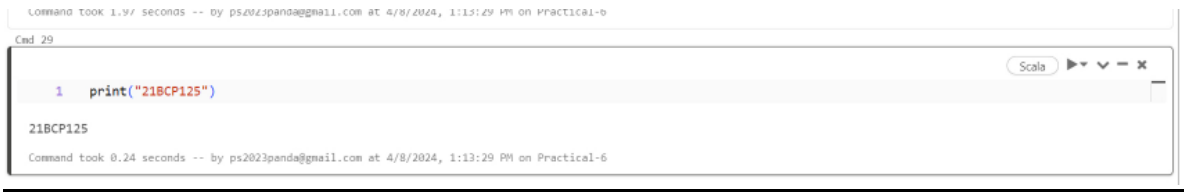
```
1  a.distinct(1).collect
```

▸ (1) Spark Jobs

```
res21: Array[Int] = Array(4, 1, 6, 3, 7, 5, 2)
```

Command took 1.97 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:13:29 PM on Practical-6

Cmd 29



PRACTICAL -4 Advanced Transformation Function

```
1  print("hi")
```

```
hi
```

Command took 5.90 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 2

```
1  print("21BCP125")
```

```
21BCP125
```

Command took 0.23 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 3

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

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

Command took 0.24 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 4

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

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

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 5

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

```
brodContries: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Map[String,String]] = Broadcast(150)
```

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 6

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

```
brodState: org.apache.spark.broadcast.Broadcast[scala.collection.immutable.Map[String,String]] = Broadcast(151)
```

Command took 0.30 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 7

```
1  val data = Seq(("Mit", "Patel", "IN","CA"),("Smit","Patel","USA","CA"),("Mire","Patel","IN","NY"),("Parth","Patel","USA","FL"))
2
```

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

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 8

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

```
rdd: org.apache.spark.rdd.RDD[(String, String, String, String)] = ParallelCollectionRDD[293] at parallelize at command-1995308254957706:1
```

Command took 0.28 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 9

```
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)
})
```

```
rdd2: org.apache.spark.rdd.RDD[(String, String, String, String)] = MapPartitionsRDD[294] at map at command-1995308254957707:1
```

Command took 0.51 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 10

```
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:_*)
```

```
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]
```

Command took 2.71 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 11

```
df2.show()
```

▸ (2) Spark Jobs

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

Command took 1.00 second -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 12

```
print("21BCP125")
```

```
21BCP125
```

Command took 0.53 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 13

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

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

```

```
longAcc: org.apache.spark.util.LongAccumulator = LongAccumulator(id: 9356, name: Some(SUM), value: 0)
rdd: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[297] at parallelize at command-1995308254957711:3
```

Command took 0.42 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 14

```
1  rdd.foreach(x=>longAcc.add(x))
2  rdd.collect
```

▸ (2) Spark Jobs

```
res4: Array[Int] = Array(1, 2, 3, 4, 5)
```

Command took 0.45 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 15

```
1  longAcc.value
```

```
res5: Long = 15
```

Command took 0.28 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 16

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

Command took 0.25 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 17

```
1  val inputRDD = spark.sparkContext.parallelize(List(("Z",1),("B",30),("A",20),("B",30),("C",40),("B",60)))
2
```

```
inputRDD: org.apache.spark.rdd.RDD[(String, Int)] = ParallelCollectionRDD[298] at parallelize at command-4181581249367484:1
```

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:15:59 PM on Practical-6

Cmd 18

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

```
listRDD: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[299] at parallelize at command-4181581249367485:1
```

Command took 0.27 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:16:00 PM on Practical-6

Cmd 19

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

▸ (1) Spark Jobs

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

Command took 0.48 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:16:00 PM on Practical-6

Cmd 20

```
1  def param3 = (acc:Int , v:(String , Int)) => acc + v._2
2  def param2 = (acc1:Int , v2:Int) => acc1 + v2
3  println("Aggregate :" + inputRDD.aggregate(0) (param3 , param2))
```

▸ (1) Spark Jobs

```
Aggregate :181
param3: (Int, (String, Int)) => Int
param2: (Int, Int) => Int
```

Command took 0.41 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:16:00 PM on Practical-6

Cmd 21

Scala

```
1  print("21BCP125")
```

```
21BCP125
```

Command took 0.25 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:16:00 PM on Practical-6

PRACTICAL -5 SQL Practice

Cmd 1

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

```
command-4181581249367490:1: warning: constructor SQLContext in class SQLContext is deprecated (since 2.0.0): Use SparkSession.builder instead
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
                 ^
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@53103874
```

Command took 5.38 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 2

```
1  print("21BCP125")
```

```
21BCP125
```

Command took 0.21 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 3

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

```
a: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[300] at parallelize at command-4181581249367491:1
```

Command took 0.28 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 4

```
1  a.collect
```

▸ (1) Spark Jobs

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

Command took 0.35 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 5

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

```
b: org.apache.spark.rdd.RDD[(Int, Int)] = MapPartitionsRDD[301] at map at command-4181581249367493:1
```

Command took 0.36 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 6

```
1  b.collect
```

▸ (1) Spark Jobs

```
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))
```

Command took 0.25 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 7

```
1  print("21BCP125")
```

```
21BCP125
```

Command took 0.27 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 8

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

▸ (3) Spark Jobs

▸ df: org.apache.spark.sql.DataFrame = [First: integer, Second: integer]

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

Cmd 9

```
1  val df1 = spark.read.format("json").load("dbfs:/FileStore/shared_uploads/ps2023panda@gmail.com/data_1.json")
```

▸ (1) Spark Jobs

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

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

Command took 1.58 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 10

```
1  df1.show
```

▸ (1) Spark Jobs

```
+---+----+-------+
|Age|  id|   name|
+---+----+-------+
| 25|1201| Satish|
| 28|1202|Krishna|
| 39|1203|  Amith|
| 23|1204|  Javed|
| 23|1205|Pruthvi|
+---+----+-------+
```

Command took 0.60 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 11

```
1  df1.select("name").show()
```

▸ (1) Spark Jobs

```
+-------+
|   name|
+-------+
| Satish|
|Krishna|
|  Amith|
|  Javed|
|Pruthvi|
+-------+
```

Command took 0.67 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

Cmd 12

Scala

```
1  print("21BCP125")
```

```
21BCP125
```

Command took 0.65 seconds -- by ps2023panda@gmail.com at 4/8/2024, 2:07:31 PM on Practical-6

PRACTICAL -6 Data Processing using PySpark

Cmd 1

```
print("21BCP125")
```

```
21BCP125
```

Command took 0.05 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 2

```
from pyspark.sql import SparkSession
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 3

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

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 4

```
df1 = spark.read.csv("dbfs:/FileStore/shared_uploads/ps2023panda@gmail.com/sample_data.csv",inferSchema=True,header=True)
```

(2) Spark Jobs

df1: pyspark.sql.dataframe.DataFrame = [ratings: integer, age: integer ... 3 more fields]

Command took 1.50 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 5

```
df1.count()
```

(2) Spark Jobs

```
Out[44]: 47
```

Command took 0.69 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 6

```
df1.printSchema()
```

```
root
 |-- ratings: integer (nullable = true)
 |-- age: integer (nullable = true)
 |-- experience: double (nullable = true)
 |-- family: integer (nullable = true)
 |-- mobile: string (nullable = true)
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 7

```
df1.select('ratings','age','mobile').show(5)
```

▸ (1) Spark Jobs

```
+-------+---+------------+
|ratings|age|      mobile|
+-------+---+------------+
|      3| 32|        Vivo|
|      4| 28|     Samsung|
|      5| 35|      iPhone|
|      2| 40|     OnePlus|
|      4| 27|Google Pixel|
+-------+---+------------+
only showing top 5 rows
```

Command took 0.50 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 8

```
df1.describe().show()
```

▸ (2) Spark Jobs

```
+-------+------------------+-----------------+-----------------+------------------+-------+
|summary|           ratings|              age|       experience|            family| mobile|
+-------+------------------+-----------------+-----------------+------------------+-------+
|  count|                47|               47|               47|                47|     47|
|   mean|3.6808510638297873|32.40425531914894|9.295744680851064|2.6808510638297873|   null|
| stddev| 1.023769313747283|3.8485433408548855| 1.17602082331618| 1.023769313747283|   null|
|    min|                 2|               26|              7.0|                 1|Alcatel|
|    max|                 5|               40|             12.0|                 4| iPhone|
+-------+------------------+-----------------+-----------------+------------------+-------+
```

Command took 1.60 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 9

```
from pyspark.sql.types import StringType, DoubleType, IntegerType
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 10

```
df1.withColumn("age_after_10_yrs", (df1["age"]+10)).show(10,False)
```

▸ (1) Spark Jobs

```
+-------+---+----------+------+------------+----------------+
|ratings|age|experience|family|mobile      |age_after_10_yrs|
+-------+---+----------+------+------------+----------------+
|3      |32 |9.0       |3     |Vivo        |42              |
|4      |28 |8.5       |2     |Samsung     |38              |
|5      |35 |10.2      |4     |iPhone      |45              |
|2      |40 |12.0      |1     |OnePlus     |50              |
|4      |27 |7.8       |3     |Google Pixel|37              |
|3      |33 |9.5       |2     |OPPO        |43              |
|5      |29 |8.0       |4     |Huawei      |39              |
|4      |36 |10.8      |3     |Xiaomi      |46              |
|3      |31 |9.2       |2     |LG          |41              |
|4      |34 |9.7       |3     |Sony        |44              |
+-------+---+----------+------+------------+----------------+
only showing top 10 rows
```

Command took 0.40 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 11

```
df1.withColumn("age_after_10_yrs", (df1["age"]+10)).show(10)
```

▸ (1) Spark Jobs

```
+-------+---+----------+------+------------+----------------+
|ratings|age|experience|family|      mobile|age_after_10_yrs|
+-------+---+----------+------+------------+----------------+
|      3| 32|       9.0|     3|        Vivo|              42|
|      4| 28|       8.5|     2|     Samsung|              38|
|      5| 35|      10.2|     4|      iPhone|              45|
|      2| 40|      12.0|     1|     OnePlus|              50|
|      4| 27|       7.8|     3|Google Pixel|              37|
|      3| 33|       9.5|     2|        OPPO|              43|
|      5| 29|       8.0|     4|      Huawei|              39|
|      4| 36|      10.8|     3|      Xiaomi|              46|
|      3| 31|       9.2|     2|          LG|              41|
|      4| 34|       9.7|     3|        Sony|              44|
+-------+---+----------+------+------------+----------------+
only showing top 10 rows
```

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 12

```
1  df1.withColumn("age_double", (df1["age"]).cast(DoubleType())).show(10,False)
```

▸ (1) Spark Jobs

```
+-------+---+----------+------+------------+----------+
|ratings|age|experience|family|mobile      |age_double|
+-------+---+----------+------+------------+----------+
|3      |32 |9.0       |3     |Vivo        |32.0      |
|4      |28 |8.5       |2     |Samsung     |28.0      |
|5      |35 |10.2      |4     |iPhone      |35.0      |
|2      |40 |12.0      |1     |OnePlus     |40.0      |
|4      |27 |7.8       |3     |Google Pixel|27.0      |
|3      |33 |9.5       |2     |OPPO        |33.0      |
|5      |29 |8.0       |4     |Huawei      |29.0      |
|4      |36 |10.8      |3     |Xiaomi      |36.0      |
|3      |31 |9.2       |2     |LG          |31.0      |
|4      |34 |9.7       |3     |Sony        |34.0      |
+-------+---+----------+------+------------+----------+
only showing top 10 rows
```

Command took 0.60 seconds by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical 6

Cmd 13

```
1  df1.filter(df1['mobile'] == 'Vivo').show()
```

▸ (1) Spark Jobs

```
+-------+---+----------+------+------+
|ratings|age|experience|family|mobile|
+-------+---+----------+------+------+
|      3| 32|       9.0|     3|  Vivo|
|      3| 31|       9.0|     2|  Vivo|
+-------+---+----------+------+------+
```

Command took 0.50 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 14

```
1  df1.filter(df1['mobile'] == 'Vivo').select('age','ratings','mobile').show()
```

▸ (1) Spark Jobs

```
+---+-------+------+
|age|ratings|mobile|
+---+-------+------+
| 32|      3|  Vivo|
| 31|      3|  Vivo|
+---+-------+------+
```

Command took 0.49 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 15

```
1  df1.filter(df1['mobile'] == 'OnePlus').filter(df1['experience']>10).show()
```

▸ (1) Spark Jobs

```
+-------+---+----------+------+-------+
|ratings|age|experience|family| mobile|
+-------+---+----------+------+-------+
|      2| 40|      12.0|     1|OnePlus|
+-------+---+----------+------+-------+
```

Command took 0.69 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 16

```
1  df1.filter((df1['mobile'] == 'OnePlus') & (df1['experience']>10)).show()
```

▸ (1) Spark Jobs

```
+-------+---+----------+------+-------+
|ratings|age|experience|family| mobile|
+-------+---+----------+------+-------+
|      2| 40|      12.0|     1|OnePlus|
+-------+---+----------+------+-------+
```

Command took 0.39 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 17

```
1  df1.select('mobile').distinct().show()
```

▸ (2) Spark Jobs

```
|      Sony|
|   Alcatel|
|  Motorola|
|      OPPO|
|    Realme|
|    iPhone|
|    Huawei|
|    Xiaomi|
|      Asus|
|    Lenovo|
|   Samsung|
|       HTC|
|Blackberry|
|        LG|
|     Pixel|
|   OnePlus|
|      Vivo|
|       ZTE|
+----------+
only showing top 20 rows
```

Command took 0.89 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 18

```
1  df1.groupBy('mobile').count().show(5,False)
```

▸ (2) Spark Jobs

```
+--------+-----+
|mobile  |count|
+--------+-----+
|Infinix |2    |
|Nokia   |2    |
|Sony    |2    |
|Alcatel |2    |
|Motorola|2    |
+--------+-----+
only showing top 5 rows
```

Command took 1.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 19

```
1  df1.groupBy('mobile').count().orderBy('count', ascending=False).show(5,False)
```

▸ (2) Spark Jobs

```
+-------+-----+
|mobile |count|
+-------+-----+
|OnePlus|3    |
|Infinix|2    |
|Nokia  |2    |
|Sony   |2    |
|Alcatel|2    |
+-------+-----+
only showing top 5 rows
```

Command took 0.90 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 20

```
1  df1.groupBy('mobile').mean().show(5,False)
```

▸ (2) Spark Jobs

```
+--------+------------+--------+---------------+-----------+
|mobile  |avg(ratings)|avg(age)|avg(experience)|avg(family)|
+--------+------------+--------+---------------+-----------+
|Infinix |3.0         |32.5    |9.25           |2.0        |
|Nokia   |3.5         |27.0    |7.6            |2.0        |
|Sony    |3.0         |36.5    |10.6           |2.0        |
|Alcatel |3.5         |34.0    |9.85           |2.5        |
|Motorola|3.5         |35.0    |9.9            |2.5        |
+--------+------------+--------+---------------+-----------+
only showing top 5 rows
```

Command took 1.19 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 21

```
1  df1.groupBy('mobile').sum().show(5,False)
```

▸ (2) Spark Jobs

```
+--------+-----------+--------+--------------+----------+
|mobile  |sum(ratings)|sum(age)|sum(experience)|sum(family)|
+--------+-----------+--------+--------------+----------+
|Infinix |6          |65      |18.5          |4         |
|Nokia   |7          |54      |15.2          |4         |
|Sony    |6          |73      |21.2          |4         |
|Alcatel |7          |68      |19.7          |5         |
|Motorola|7          |70      |19.8          |5         |
+--------+-----------+--------+--------------+----------+
only showing top 5 rows
```

Command took 1.11 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 22

```
1  df1.groupBy('mobile').max().show(5,False)
```

▸ (2) Spark Jobs

```
+--------+-----------+--------+--------------+----------+
|mobile  |max(ratings)|max(age)|max(experience)|max(family)|
+--------+-----------+--------+--------------+----------+
|Infinix |4          |37      |10.3          |3         |
|Nokia   |4          |28      |8.2           |3         |
|Sony    |4          |39      |11.5          |3         |
|Alcatel |5          |39      |11.0          |4         |
|Motorola|4          |37      |10.0          |3         |
+--------+-----------+--------+--------------+----------+
only showing top 5 rows
```

Command took 1.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 23

```
1  df1.groupBy('mobile').min().show(5,False)
```

▸ (2) Spark Jobs

```
+--------+-----------+--------+--------------+----------+
|mobile  |min(ratings)|min(age)|min(experience)|min(family)|
+--------+-----------+--------+--------------+----------+
|Infinix |2          |28      |8.2           |1         |
|Nokia   |3          |26      |7.0           |1         |
|Sony    |2          |34      |9.7           |1         |
|Alcatel |2          |29      |8.7           |1         |
|Motorola|3          |33      |9.8           |2         |
+--------+-----------+--------+--------------+----------+
only showing top 5 rows
```

Command took 1.10 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 24

```
df1.groupBy('mobile').agg({'experience':'sum'}).show(5,False)
```

▸ (2) Spark Jobs

```
+--------+---------------+
|mobile  |sum(experience)|
+--------+---------------+
|Infinix |18.5           |
|Nokia   |15.2           |
|Sony    |21.2           |
|Alcatel |19.7           |
|Motorola|19.8           |
+--------+---------------+
only showing top 5 rows
```

Command took 1.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 25

```
from  pyspark.sql.functions import udf
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 26

```
print("21BCP125")
```

```
21BCP125
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 27

```
def price_range(brand):
    if brand in ['Samsung','iPhone']:
        return 'High Price'
    elif brand == 'Xiomi':
        return 'Mid Price'
    else:
        return 'Low Price'
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 28

```
brand_udf = udf(price_range, StringType())
df1.withColumn('price_range', brand_udf(df1['mobile'])).show(10,False)
```

▸ (1) Spark Jobs

```
+-------+---+----------+------+------------+-----------+
|ratings|age|experience|family|mobile      |price_range|
+-------+---+----------+------+------------+-----------+
|3      |32 |9.0       |3     |Vivo        |Low Price  |
|4      |28 |8.5       |2     |Samsung     |High Price |
|5      |35 |10.2      |4     |iPhone      |High Price |
|2      |40 |12.0      |1     |OnePlus     |Low Price  |
|4      |27 |7.8       |3     |Google Pixel|Low Price  |
|3      |33 |9.5       |2     |OPPO        |Low Price  |
|5      |29 |8.0       |4     |Huawei      |Low Price  |
|4      |36 |10.8      |3     |Xiaomi      |Low Price  |
|3      |31 |9.2       |2     |LG          |Low Price  |
|4      |34 |9.7       |3     |Sony        |Low Price  |
+-------+---+----------+------+------------+-----------+
only showing top 10 rows
```

Command took 1.20 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 29

```
from pyspark.sql.functions import pandas_udf , PandasUDFType
```

Command took 0.07 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 30

```
def remaining_yrs(age):
    yrs_left = 100-age
    return yrs_left
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 31

```
length_udf = pandas_udf(remaining_yrs, IntegerType())

df1.withColumn("yrs_left", length_udf(df1['age'])).show(10,False)
```

▸ (1) Spark Jobs

```
+-------+---+----------+------+------------+--------+
|ratings|age|experience|family|mobile      |yrs_left|
+-------+---+----------+------+------------+--------+
|3      |32 |9.0       |3     |Vivo        |68      |
|4      |28 |8.5       |2     |Samsung     |72      |
|5      |35 |10.2      |4     |iPhone      |65      |
|2      |40 |12.0      |1     |OnePlus     |60      |
|4      |27 |7.8       |3     |Google Pixel|73      |
|3      |33 |9.5       |2     |OPPO        |67      |
|5      |29 |8.0       |4     |Huawei      |71      |
|4      |36 |10.8      |3     |Xiaomi      |64      |
|3      |31 |9.2       |2     |LG          |69      |
|4      |34 |9.7       |3     |Sony        |66      |
+-------+---+----------+------+------------+--------+
only showing top 10 rows
```

Command took 1.89 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 32

```
def prod(rating, exp):
    x = rating*exp
    return x
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 33

```
prod_udf = pandas_udf(prod, DoubleType())

df1.withColumn("product", prod_udf(df1['ratings'], df1['experience'])).show(10,False)
```

▸ (1) Spark Jobs

```
+-------+---+----------+------+------------+-----------------+
|ratings|age|experience|family|mobile      |product          |
+-------+---+----------+------+------------+-----------------+
|3      |32 |9.0       |3     |Vivo        |27.0             |
|4      |28 |8.5       |2     |Samsung     |34.0             |
|5      |35 |10.2      |4     |iPhone      |51.0             |
|2      |40 |12.0      |1     |OnePlus     |24.0             |
|4      |27 |7.8       |3     |Google Pixel|31.2             |
```

Cmd 36

Python

```
!pwd
```

```
/databricks/driver
```

Command took 1.14 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

Cmd 37

```
write_uri= '/home/jovyan/work/df_csv'
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:50:44 PM on Practical-6

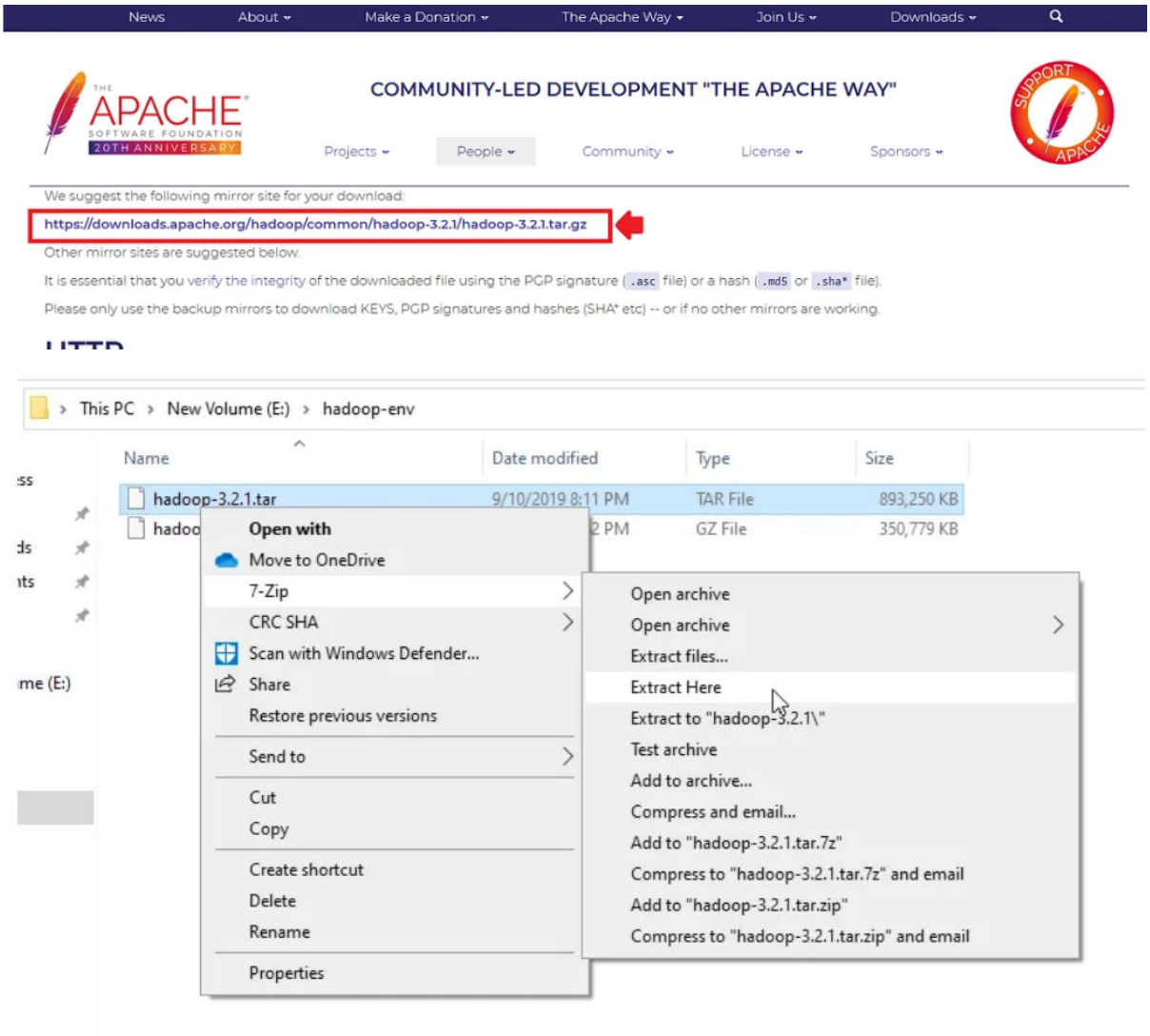
Cmd 38

PRACTICAL -7 Install Hadoop, Pig and Hive

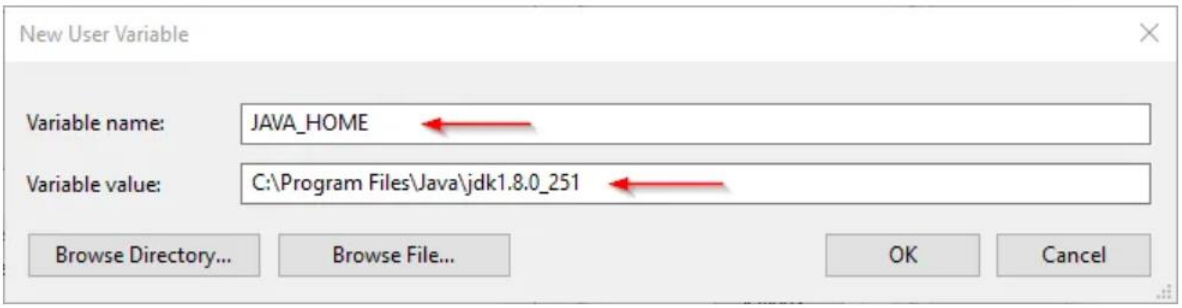
Hadoop install

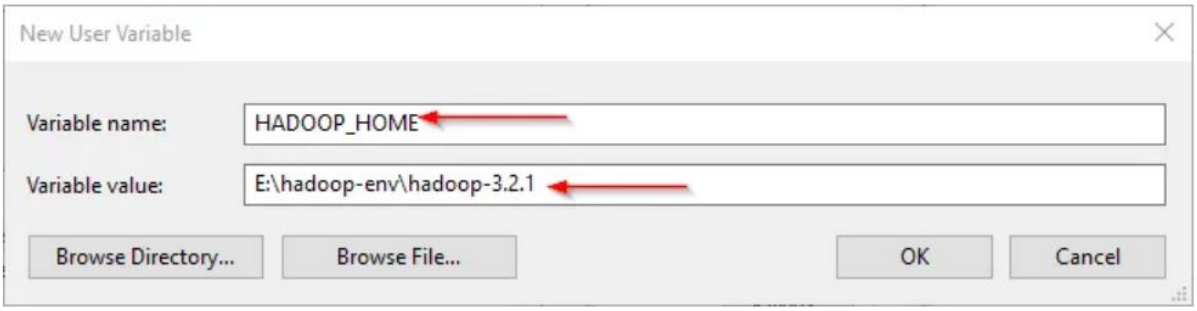
Step-1 Download Java-8(jdk)

Step-2 Download Hadoop Binaries

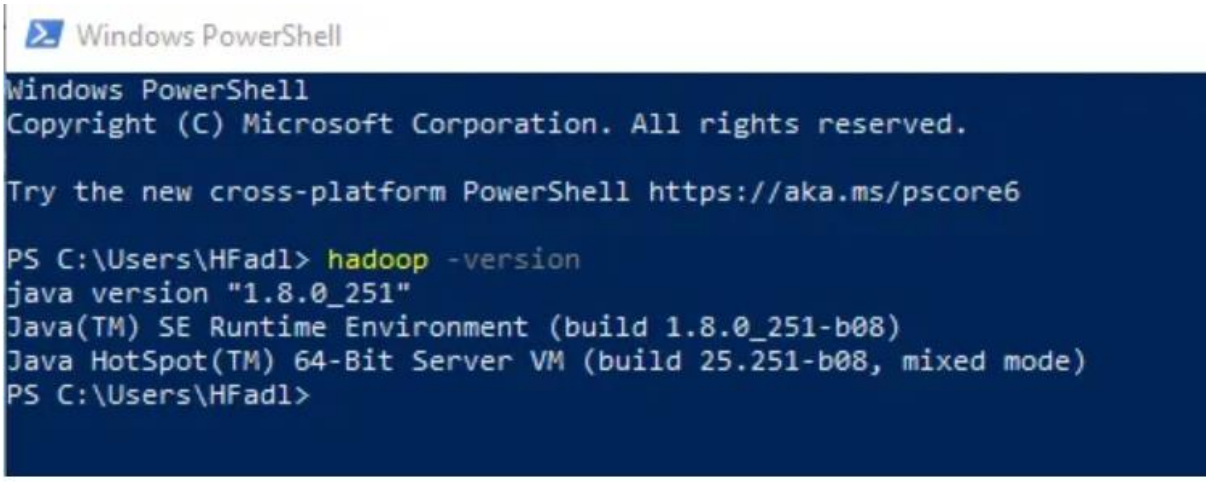


Step-3 Add environment variables





Step-4 Check if installed or not



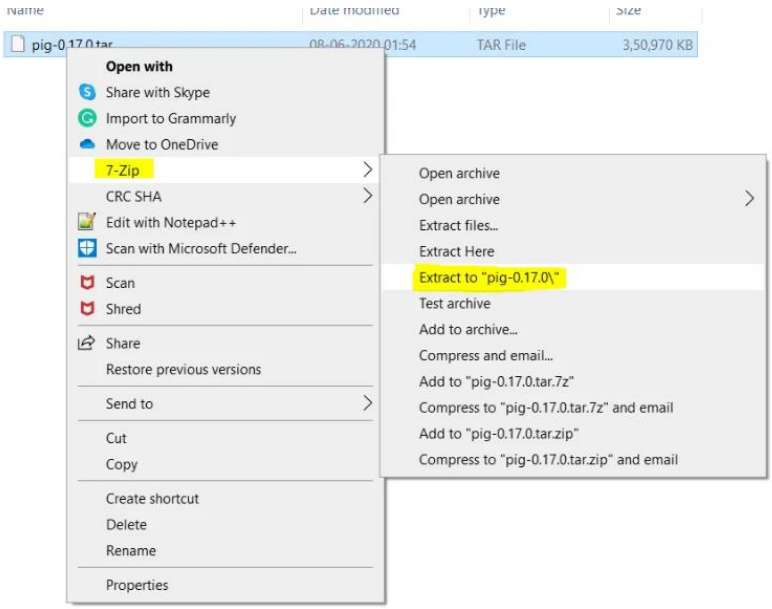
Step-5 Setup Hadoop

Change this 3 files

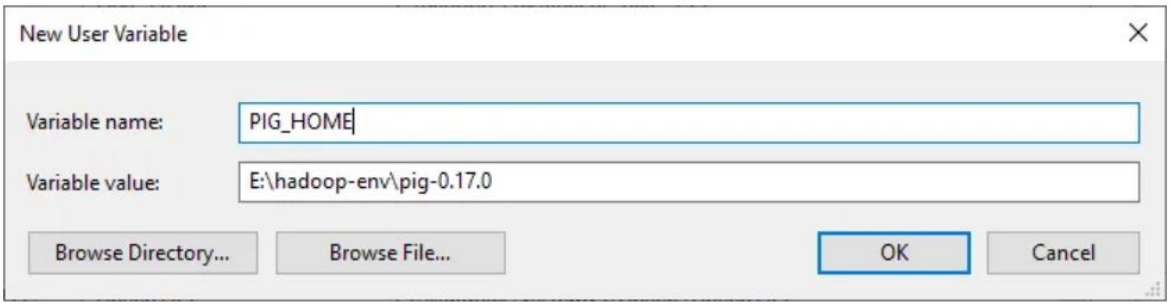
- core-site.xml
- hdfs-site.xml
- mapred-site.xml

Pig install

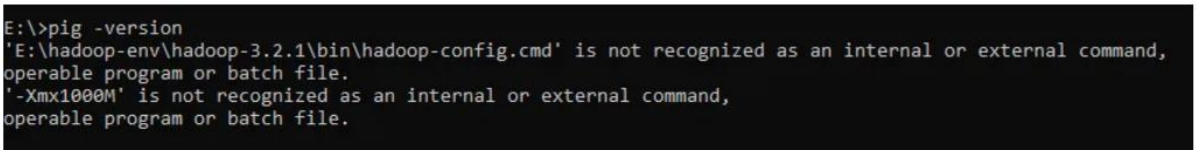
Step-1 Install pig and unzip



Step-2 Add environment variable



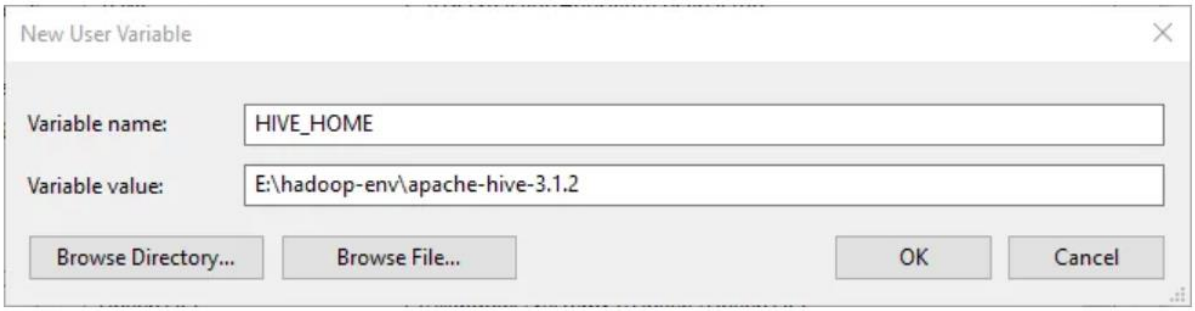
Step-3 Check if installed or not



Hive Install

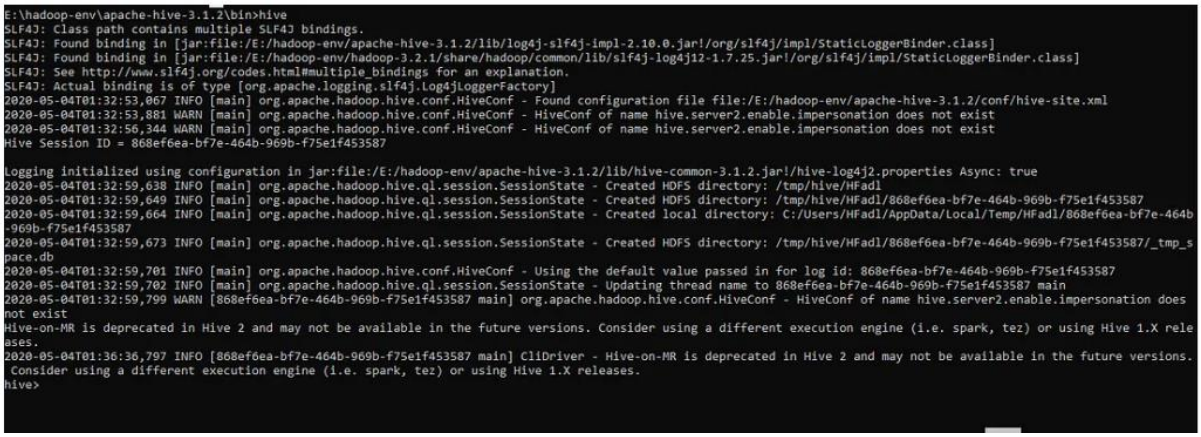
Step-1 install hive and unzip

Step-2 Add environment variable



Step-3 Copy all the *.jar files in hadoop folder to Hive libraries directory and configure hive-site.xml

Step-4 Check if installed or not



PRACTICAL -8 Matplotlib

bdalab_27_3_24 Python

File Edit View Run Help Last edit was 1 minute ago New cell UI: OFF

Run all | Practical-6 | Share | Publish

Cmd 1

```
1  print("21BCP125")
```

```
21BCP125
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/8/2024, 12:58:53 PM on Practical-6

Cmd 2

```
1  %pip install networkx
2  %pip install scipy
3  %pip install --upgrade scipy
```

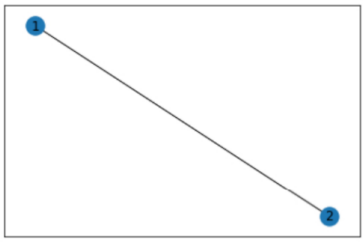
```
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in /databricks/python3/lib/python3.9/site-packages (from scipy) (1.21.5)
Python interpreter will be restarted.
Python interpreter will be restarted.
Requirement already satisfied: scipy in /databricks/python3/lib/python3.9/site-packages (1.7.3)
Collecting scipy
  Downloading scipy-1.13.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (38.6 MB)
Collecting numpy<2.3,>=1.22.4
  Downloading numpy-1.26.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.2 MB)
Installing collected packages: numpy, scipy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.21.5
```

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

Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:18 PM on Practical-6

Cmd 4

```
1  G = nx.Graph()
2
3  G.add_edge(1,2)
4
5  nx.draw_networkx(G)
6  plt.show()
```



Command took 0.30 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:20 PM on Practical-6

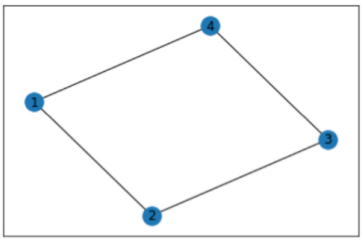
Cmd 5

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

Command took 0.27 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:23 PM on Practical-6

Cmd 6

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



Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:27 PM on Practical-6

Cmd 7

```
print("21BCP125")
```

```
21BCP125
```

Cmd 8

```
G.nodes
```

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

Command took 0.08 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:31 PM on Practical-6

Cmd 9

```
G.edges
```

```
Out[7]: EdgeView([(1, 2), (1, 4), (2, 3), (3, 4)])
```

Command took 0.10 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:32 PM on Practical-6

Cmd 10

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

```
Out[8]: ['1 2 4', '2 3', '3 4', '4']
```

Command took 0.13 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:35 PM on Practical-6

Cmd 11

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

```
Out[9]: {1: [2, 4], 2: [1, 3], 3: [4, 2], 4: [3, 1]}
```

Command took 0.03 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:36 PM on Practical-6

Cmd 12

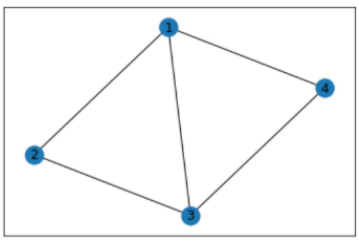
```
nx.to_edgelist(G)
```

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

Command took 0.04 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:38 PM on Practical-6

Cmd 13

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



Command took 0.30 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:39 PM on Practical-6

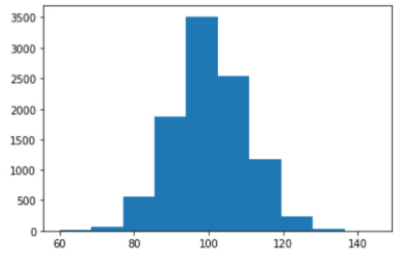
```
1  G.degree
```

```
Out[12]: DegreeView({1: 3, 2: 2, 3: 3, 4: 2})
```

Command took 0.07 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:42 PM on Practical-6

Cmd 15

```
1  kg=nx.fast_gnp_random_graph(10000,0.01)
2  k = kg.degree()
3  plt.hist(list(dict(k).values()))
4  plt.show()
```

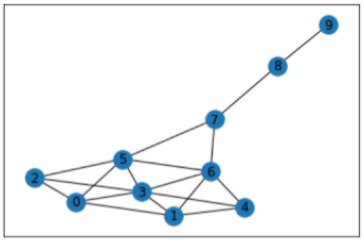


Command took 1.63 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:01:45 PM on Practical-6

Cmd 16

Cmd 16

```
1  G=nx.krackhardt_kite_graph()
2  nx.draw_networkx(G)
3  plt.show()
```



Command took 0.30 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:04:13 PM on Practical-6

Cmd 17

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

```
True
[1, 6, 7, 8, 9]
4
Out[16]: {5: [5],
 0: [5, 0],
 2: [5, 2],
 3: [5, 3],
 6: [5, 6],
 7: [5, 7],
 1: [5, 0, 1],
 4: [5, 3, 4],
 8: [5, 7, 8],
 9: [5, 7, 8, 9]}
```

Command took 0.06 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:04:16 PM on Practical-6

Cmd 18

```
1  nx.betweenness_centrality(G)
```

```
Out[17]: {0: 0.023148148148148143,
 1: 0.023148148148148143,
 2: 0.0,
 3: 0.10185185185185183,
 4: 0.0,
 5: 0.23148148148148148,
 6: 0.23148148148148148,
 7: 0.38888888888888884,
 8: 0.2222222222222222,
 9: 0.0}
```

Command took 0.08 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:04:18 PM on Practical-6

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

```
Out[18]: {0: 0.4444444444444444,
 1: 0.4444444444444444,
 2: 0.3333333333333333,
 3: 0.6666666666666666,
 4: 0.3333333333333333,
 5: 0.5555555555555556,
 6: 0.5555555555555556,
 7: 0.3333333333333333,
 8: 0.2222222222222222,
 9: 0.1111111111111111}
```

Command took 0.10 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:04:19 PM on Practical-6

Cmd 20

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

```
Out[19]: {0: 0.5294117647058824,
 1: 0.5294117647058824,
 2: 0.5,
 3: 0.6,
 4: 0.5,
 5: 0.6428571428571429,
 6: 0.6428571428571429,
 7: 0.6,
 8: 0.42857142857142855,
 9: 0.3103448275862069}
```

Command took 0.06 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:04:21 PM on Practical-6

Cmd 21

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

```
Out[20]: {0: 6.083333333333333,
 1: 6.083333333333333,
 2: 5.583333333333333,
 3: 7.083333333333333,
 4: 5.583333333333333,
 5: 6.833333333333333,
 6: 6.833333333333333,
 7: 6.0,
 8: 4.666666666666666,
 9: 3.4166666666666665}
```

Command took 0.12 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:04:24 PM on Practical-6

Cmd 22

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

```
Out[21]: {0: 0.3522089813920359,
 1: 0.3522089813920358,
 2: 0.28583473531632403,
 3: 0.48102048812210046,
 4: 0.28583473531632403,
 5: 0.3976910106255469,
 6: 0.39769101062554685,
 7: 0.19586185175360382,
 8: 0.048074775014202924,
 9: 0.011164058575824235}
```

Command took 0.08 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:04:26 PM on Practical-6

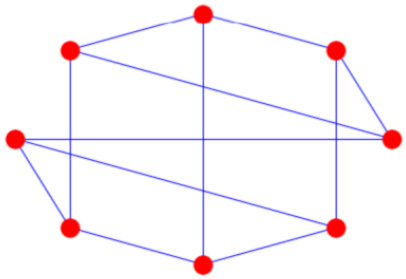
Cmd 23

```
1  nx.clustering(G)
```

```
Out[22]: {0: 0.6666666666666666,
 1: 0.6666666666666666,
 2: 1.0,
 3: 0.5333333333333333,
 4: 1.0,
 5: 0.5,
 6: 0.5,
 7: 0.3333333333333333,
 8: 0,
 9: 0}
```

Cmd 24

```
1  G=nx.cubical_graph(G)
2
3  nx.draw(G, pos=nx.circular_layout(G),node_color='r',edge_color='b')
```



Command took 0.17 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:04:29 PM on Practical-6

Cmd 25

```
1  print("21BCP125")
```

```
21BCP125
```

Command took 0.12 seconds -- by ps2023panda@gmail.com at 4/8/2024, 1:04:31 PM on Practical-6

PRACTICAL -9 Regression

Cmd 1

Python

```
1  import pandas as pd
2  import io
3  columns_names =['CRIM','ZN','CHAS','NOX','RM','AGE','DIS','RAD','TAX','PTRATIO','B','LSTAT','MEDV']
4  csv_string = dbutils.fs.head("dbfs:/FileStore/shared_uploads/ps2023panda@gmail.com/housing.csv")
5  data = pd.read_csv(io.StringIO(csv_string), header=None , delimiter=r"\s+", names=columns_names)
6  data
```

	CRIM	ZN	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

506 rows × 13 columns

Command took 2.01 seconds -- by ps2023panda@gmail.com at 4/3/2024, 9:23:27 AM on 3_april

Cmd 2

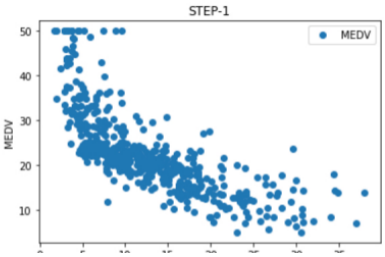
```
df_1 = data.loc[:,['LSTAT','MEDV']]
df_1.head()
#csv_string.head()
```

	LSTAT	MEDV
0.00632	4.98	24.0
0.02731	9.14	21.6
0.02729	4.03	34.7
0.03237	2.94	33.4
0.06905	5.33	36.2

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/3/2024, 9:25:52 AM on 3_april

Cmd 3

```
import matplotlib.pyplot as plt
df_1.plot(x='LSTAT',y='MEDV',style='o')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
plt.title('STEP-1')
plt.show()
```



Cmd 4

```
#x= pd.DataFrame(df_1["LSTAT"])
#y = pd.DataFrame(df_1["MEDV"])


# Assuming you have the dataset loaded into a DataFrame called df_1
x = df_1.drop('MEDV',axis=1)
y = pd.DataFrame(df_1["MEDV"])
```

Command took 0.12 seconds -- by ps2023panda@gmail.com at 4/3/2024, 10:08:22 AM on 3_april

Cmd 5

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 1)

```

Command took 0.13 seconds -- by ps2023panda@gmail.com at 4/3/2024, 10:08:45 AM on 3_april

Cmd 6

```
print(x_train.shape)
print (x_test.shape)
print(y_train.shape)
print(y_test.shape)

```

```
(354, 1)
(152, 1)
(354, 1)
(152, 1)
```

Command took 0.09 seconds -- by ps2023panda@gmail.com at 4/3/2024, 10:08:50 AM on 3_april

Cmd 7

Cmd 7

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

```
Out[29]: LinearRegression()
```

Command took 0.10 seconds -- by ps2023panda@gmail.com at 4/3/2024, 10:08:55 AM on 3_april

Cmd 8

```
print(regressor.intercept_)
```

```
[34.22183685]
```

Command took 0.12 seconds -- by ps2023panda@gmail.com at 4/3/2024, 10:08:57 AM on 3_april

Cmd 9

```
print(regressor.coef_)
```

```
[[-0.9166916]]
```

Command took 0.06 seconds -- by ps2023panda@gmail.com at 4/3/2024, 10:09:02 AM on 3_april

Cmd 10

```
y_pred = regressor.predict(x_test)
print(y_pred[:10])
```

```
[[27.31914909]
 [27.63999115]
 [16.98803475]
 [26.79663488]
 [24.88074943]
 [24.02822625]
 [29.91338632]
 [22.26817837]
 [17.79472336]
 [26.14578384]]
```

Command took 0.07 seconds -- by ps2023panda@gmail.com at 4/3/2024, 9:39:48 AM on 3_april

Cmd 11

```
from sklearn import metrics
import numpy as np
print("Mean Absolute error", metrics.mean_absolute_error(y_test , y_pred))
print("Mean Squared eror" , metrics.mean_squared_error(y_test , y_pred))
print("Root mean sq error" ,np.sqrt(metrics.mean_squared_error(y_test , y_pred)))
```

```
Mean Absolute error 4.815209094507989
Mean Squared eror 42.6202347153971
Root mean sq error 6.528418144661057
```

Command took 0.10 seconds -- by ps2023panda@gmail.com at 4/3/2024, 9:48:03 AM on 3_april

Cmd 12

```
def plot_regression_line(x,y,b):
    plt.scatter(x,y,color="m",marker="o",s=30)

    plt.plot(x,y_pred,color="g")

    plt.xlabel('x')
    plt.ylabel('y')

    plt.show()

import numpy as np
import matplotlib.pyplot as plt

def estimate_coef(x,y):

    return (regressor.intercept_ , regressor.coef_)
```

Command took 0.08 seconds -- by ps2023panda@gmail.com at 4/3/2024, 9:48:24 AM on 3_april

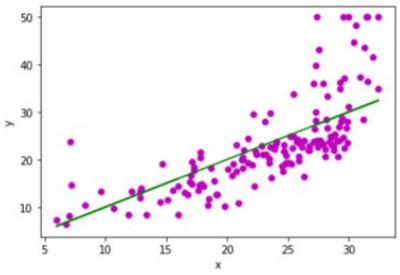
Cmd 13

```
b = estimate_coef(x_test , y_test)
print("Estimated cofficients:\nb_0={} nb_1 = {}".format(regressor.intercept_ , regressor.coef_))

#single variable regression
#plot_regression_line(x_test , y_test , b)

#multivariable regression
plot_regression_line(y_pred, y_test, b)
```

```
Estimated cofficients:
b_0=[34.22183685] nb_1 = [[-0.9166916]]
```



Command took 0.29 seconds -- by ps2023panda@gmail.com at 4/3/2024, 10:42:39 AM on 3_april

Cmd 14