**Important Instructions:**

* Please read the document thoroughly before you code.
* Please do not change the Business Requirements.

**Coverage:**

1. SQOOP
2. HBase
3. Cassandra
4. MongoDB
5. Scala

For all the hands-on, please consider the following,

**UNIX User name: training**

**Home Directory: /home/training**

**SQOOP**

**Requirement 1:**

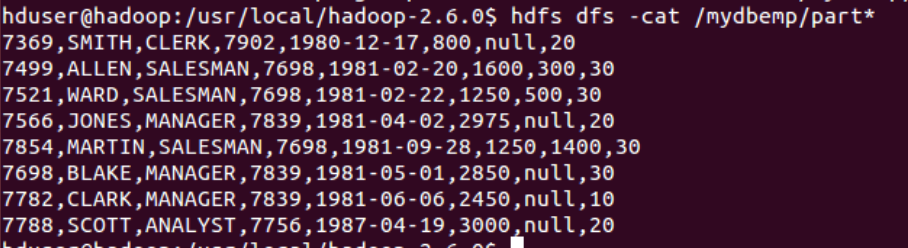
We have the table named EMP which is inside the database named MyDB in the local instance of the MySQL. You are supposed to write a SQOOP command to read the data from the table and put inside a file with comma separated values in HDFS. The credentials are given below,

Server instance: localhost

Username: root

Password: password-1

Ans: sqoop-import –connect jdbc:mysql://localhost/mydb –username root –table EMP –target-dir /mydbemp -m 1 -P



**Requirement 2:**

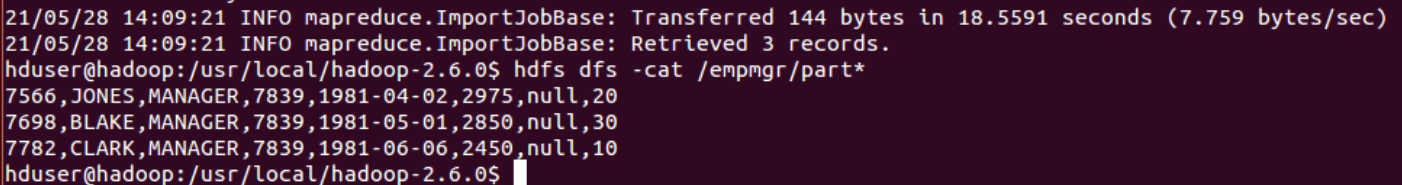
We have the table named EMP which is inside the database named MyDB in the local instance of the MySQL. You are supposed to write a SQOOP command to read the employees who work as MANAGER from the table and put inside a file with comma separated values inside the directory named MGR under HDFS. The credentials are given below,

Server instance: localhost

Username: root

Password: password-1

Ans: sqoop-import –connect jdbc:mysql://localhost/mydb –username root –table EMP –where “job=’MANAGER’” –target-dir /empmgr -m 1 -P



**Requirement 3:**

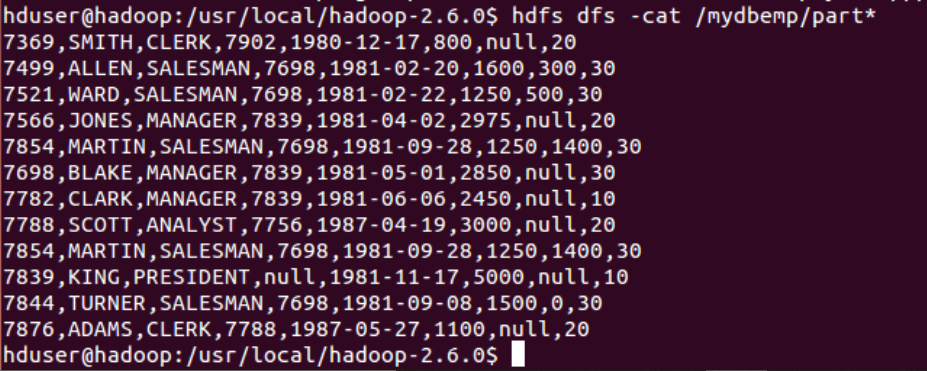
We have the already imported the data from the table EMP into HDFS in the 1st requirement. Few new employees have been added and the newly added employees should be appended at the end of file located in HDFS. You are supposed to write a SQOOP command to read the newly added employee data from the table and append at the end of file in HDFS. The credentials are given below,

Server instance: localhost

Username: root

Password: password-1

Ans: sqoop-import –connect jdbc:mysql://localhost/mydb –username root –table EMP –incremental append –check-column empno –last-value 7788 –target-dir /mydbemp -m 1 -P



**Requirement 4:**

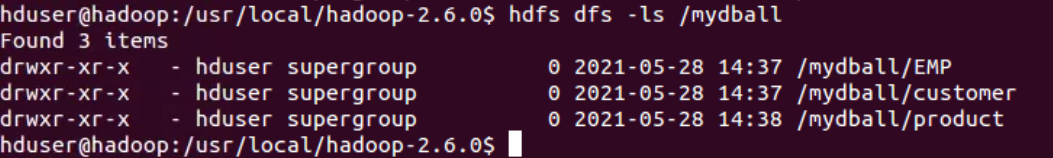
We have six tables available in the database named MyDB in MySQL instance. Requirement is load all the six tables’ data into files (Each file should be placed in a separate directory) in HDFS. You are supposed to write a SQOOP command to perform this task. The credentials are given below,

Server instance: localhost

Username: root

Password: password-1

Ans: sqoop import-all-tables –connect jdbc:mysql://localhost/mydb –username root –warehouse-dir /mydball -m 1 -P



**Requirement 5:**

We have a file named Product located in HDFS. We have a new table named Product created in the database MyDB. Requirement is to export the data in the product file into the MySQL database table named Product. The credentials are given below,

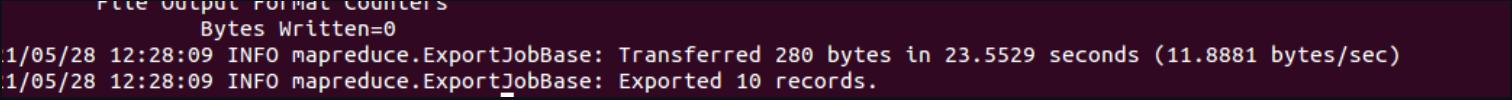
Server instance: localhost

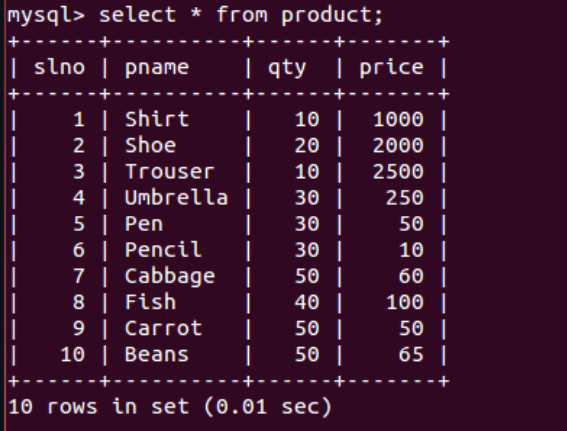
Username: root

Password: password-1

The product file and the script to create the product table is given below.





**Requirement 6:**

We have the table named Product created in the database MyDB (Populated by the previous activity). Requirement is to read the data from the MySQL table Product and load the data in the Hive table. The credentials are given below,

Server instance: localhost

Username: root

Password: password-1

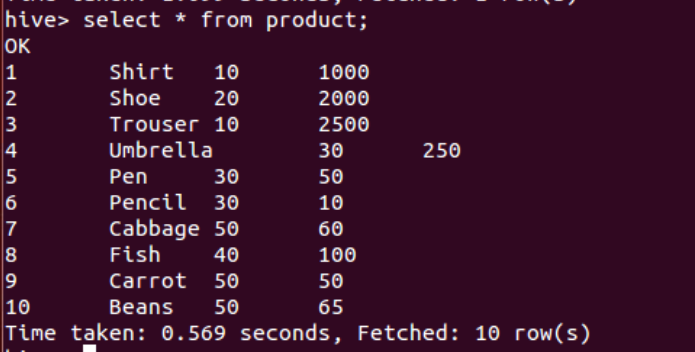
Write SQOOP commands to load the data from MySQL table to hive table, when

1. In hive, when there is no table with the name Product.

Ans: sqoop-import –connect jdbc:mysql://localhost/mydb –username root –table product –hive-import -m 1 -P

1. In hive, we have the product table existing.

Ans: sqoop-import –connect jdbc:mysql://localhost/mydb –username root –table product –hive-overwrite -m 1 -P



**HBase**

**Requirement 7:**

We need to have an HBase table named 'users' as follows

**Row Key Column Family "Name" Column Family "Image"**

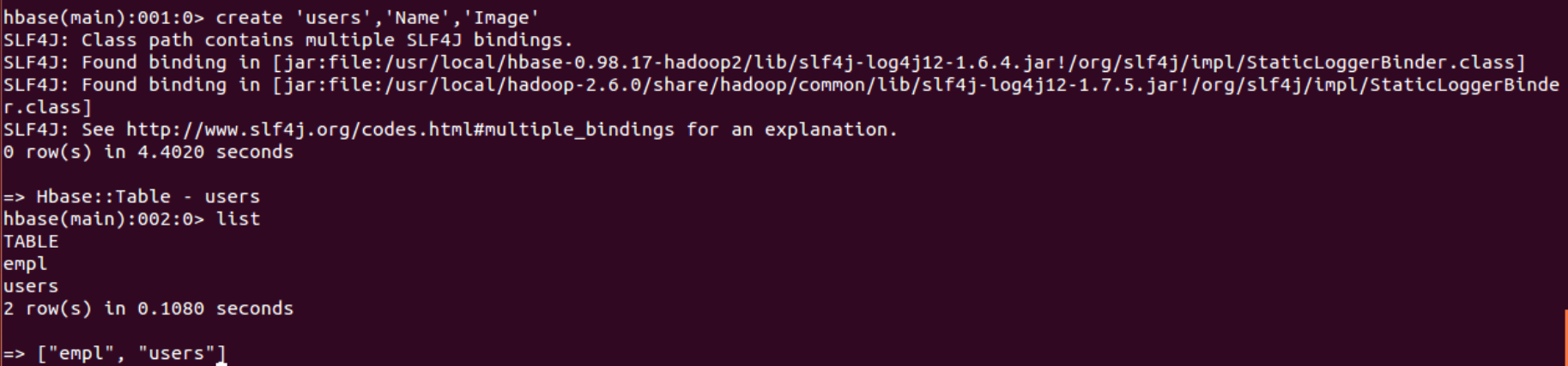
user1 Fname: Mahesh, Lname: Kumar <mahesh.jpg>

user2 Fname: David, Lname: John <david.jpg>

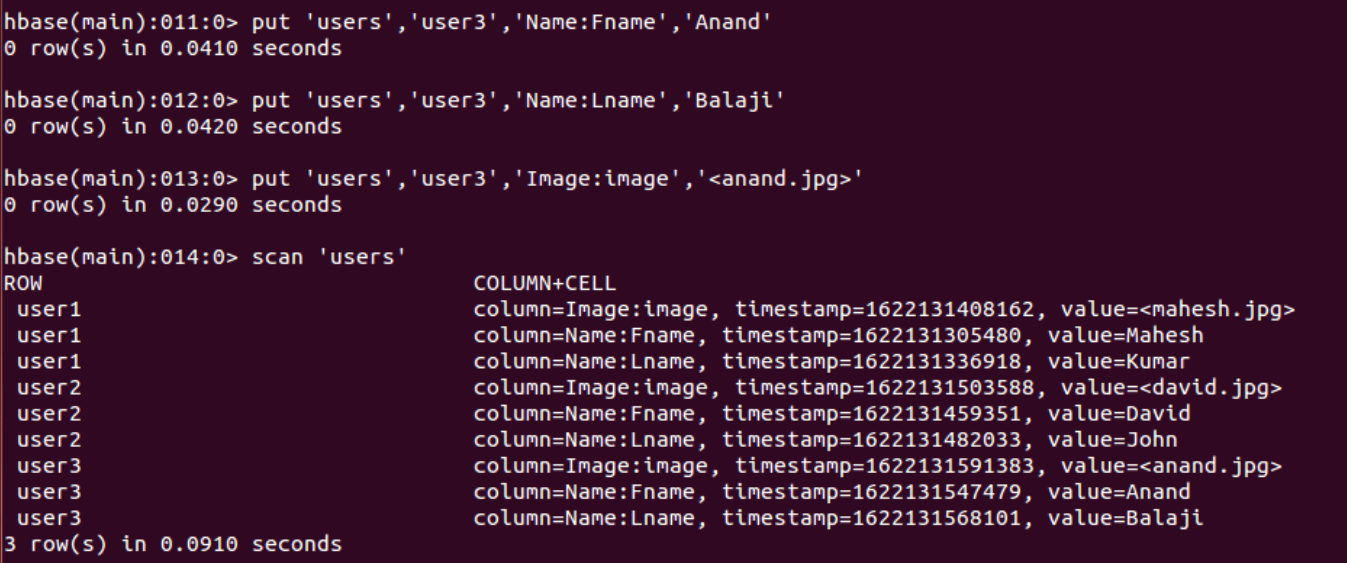
user3 Fname: Anand, Lname: Balaji <anand.jpg>

Write commands,

i. To create the table



ii. To insert the above 3 rows



**Requirement 8:**

Assume that we have 10 rows in 'users' table. Write appropriate commands to perform the following

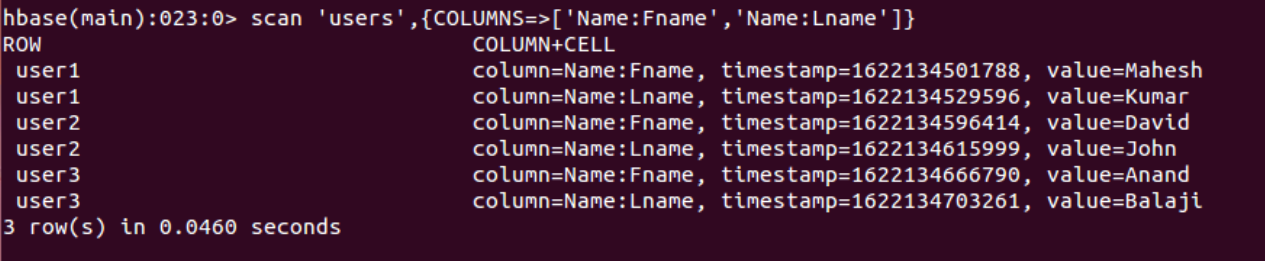
i. to read all rows from 'users' table

Ans: **scan ‘users’**

ii. To read the first 3 rows from 'users' table

Ans: **scan ‘users’,{STOPROW=>’user4’}**

iii. To read only the column family "Name"



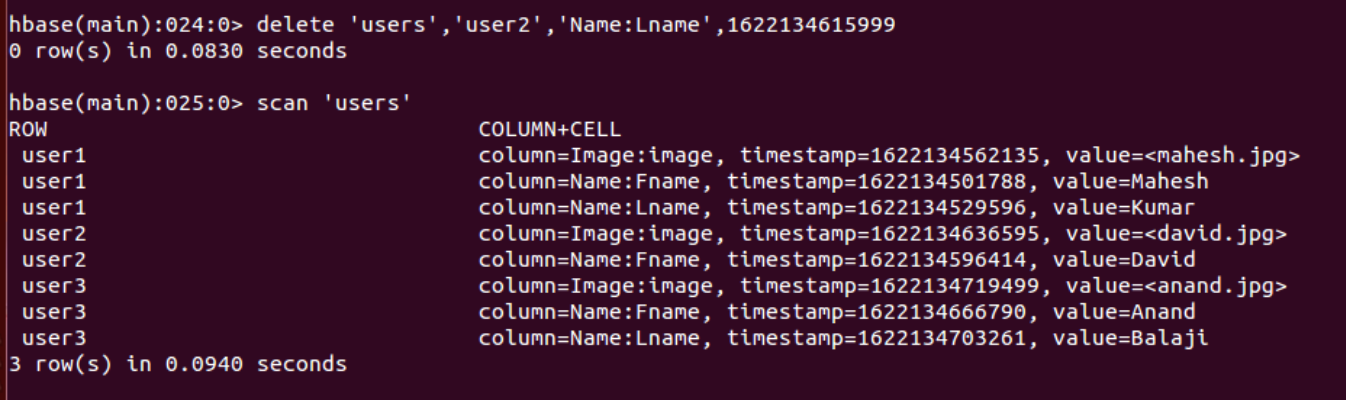
iv. To read the rows starting from 4th rows up to the 10th row

Ans: **scan ‘users’,{STARTROW=>’user4’}**

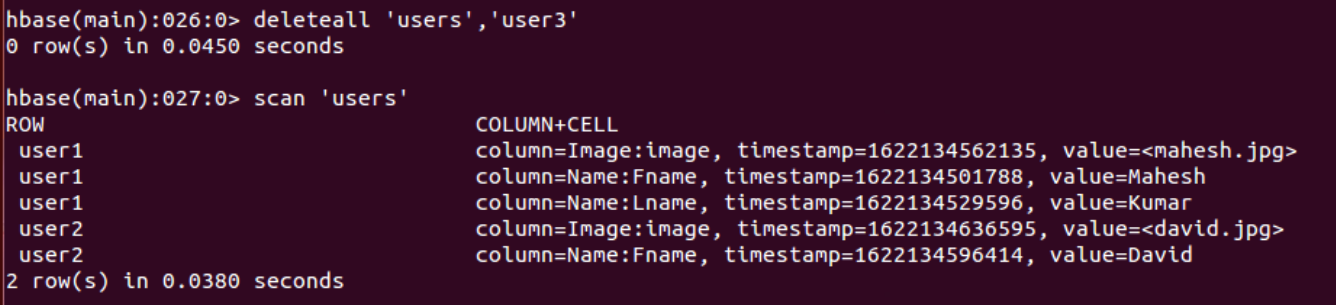
**Requirement 9:**

Assume that we have 10 rows in 'users' table. Write appropriate commands to perform the following

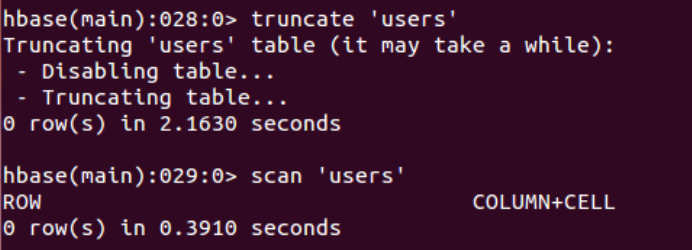
i. to delete the Lname value for the 2nd row



ii. to remove 3rd row from 'users' table



iii. to remove all the rows from the table



**Requirement 10:**

We have an HBase table named 'users' as follows

**Row Key Column Family "Name" Column Family "Image"**

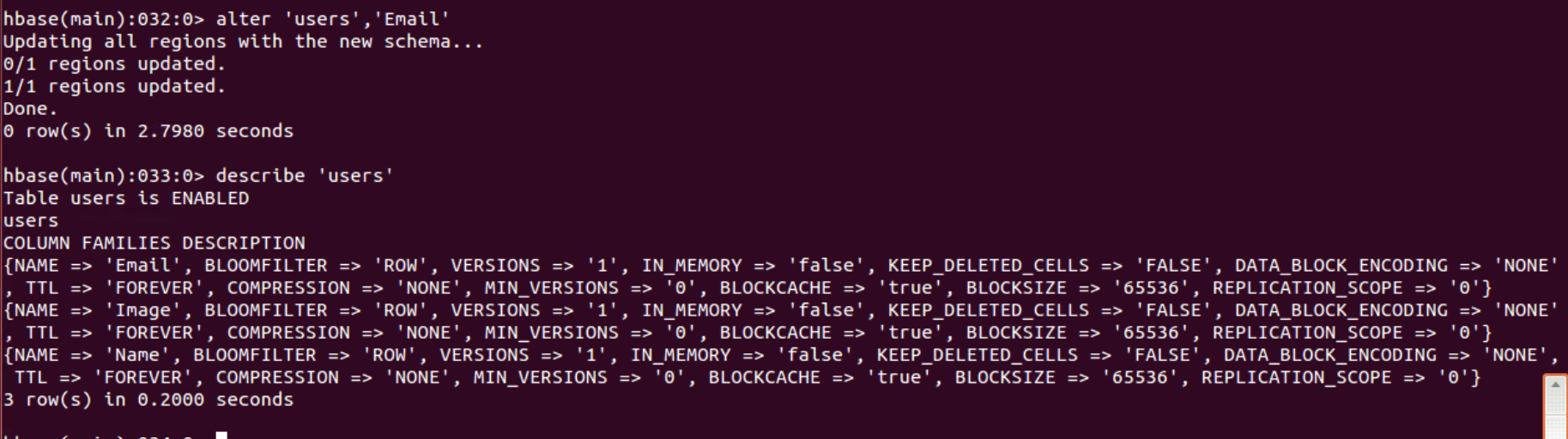
user1 Fname: Mahesh, Lname: Kumar <mahesh.jpg>

user2 Fname: David, Lname: John <david.jpg>

user3 Fname: Anand, Lname: Balaji <anand.jpg>

Write commands,

1. To add a new Column family named ‘Email’



1. To remove the existing Column family named ‘Email’



**Requirement 11:**

We have the following movies.dat file available under data folder in HDFS.



Create an HBase table named movies and load the contents of the above file into movies table.

**Requirement 12:**

We need to have an HBase table named 'users' as follows

**Row Key Column Family "Name" Column Family "Image"**

user1 Fname: Mahesh, Lname: Kumar <mahesh.jpg>

user2 Fname: David, Lname: John <david.jpg>

user3 Fname: Anand, Lname: Balaji <anand.jpg>

To create the table using **HBaseAdmin.**

**Cassandra**

**Requirement 13:**

In Cassandra, Create a key space named **DEMO** and create a table called student with the following columns

regno INT

sname TEXT

mark1 INT

mark2 INT

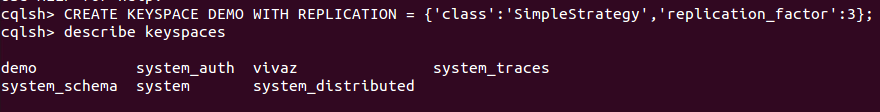
mark3 INT

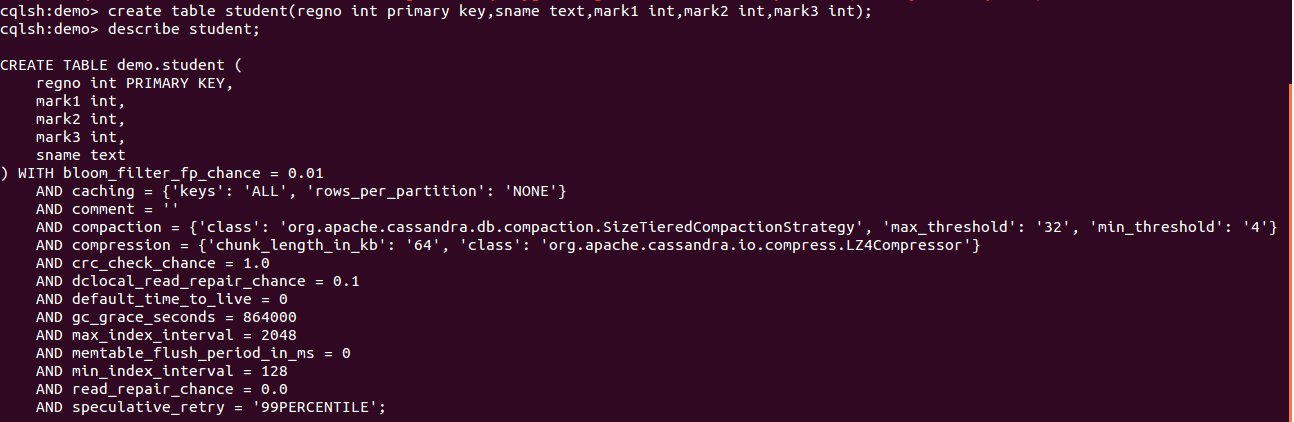
Add the following data in the student table,

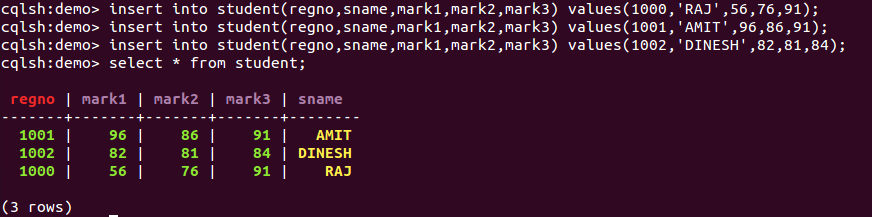
1000,'RAJ',56,76,91

1001,'AMIT',96,86,91

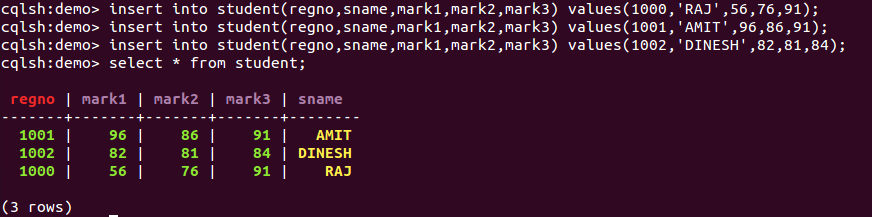
1002,'DINESH',82,81,84







Write a Query to display the details of the students whose name is AMIT.

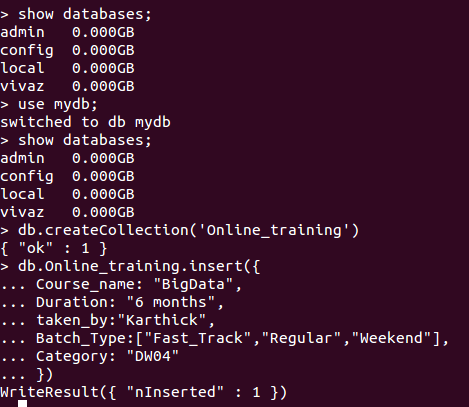


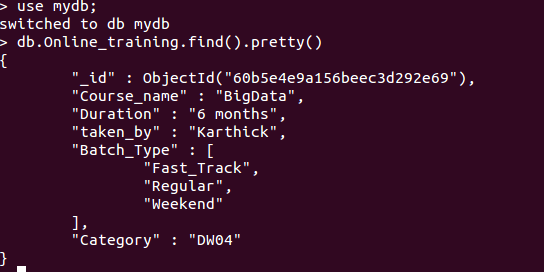
**MONGO DB**

**Requirement 14:**

A website which provides Online Trainings on various technologies, wants to record all the enquires in MongoDB. You need to create a document to achieve this, based on the following requirements

1. Course Name
2. Duration and Trainer of the course
3. Batch type – Fast Track, Regular or Weekend
4. Category.





**SCALA**

**Requirement 15:**

Write a Scala program that includes a class named Rational to generate the rational with the following requirement

Re-implement the toString method

Check the divisor part should not be zero

Example:

When the object is constructed as

val r = new Rational(1, 2)

The output should be,

r: Rational = 1/2

When the object is constructed as

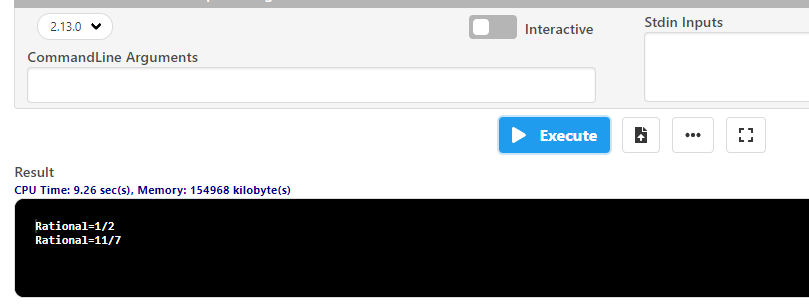
val r = new Rational(66, 42)

The output should be,

r: Rational = 11/7

Ans:

**class Rational(n: Int, d: Int) {  
 require(d != 0)  
 private val g = gcd(n.abs, d.abs)  
 val numer = n / g  
 val denom = d / g  
 override def toString = "Rational = "+numer +"/"+ denom  
 private def gcd(a: Int, b: Int): Int =   
 if (b == 0) a else gcd(b, a % b)  
 }  
object HelloWorld {  
 def main(args: Array[String]) {  
 val r = new Rational(12, 8)  
 println(r)  
 }  
}**



**Requirement 16:**

Modify the above Scala program to include the overloading of operators as follows,

When the object is constructed as

val r = new Rational(2, 3)

The output should be,

r: Rational = 2/3

Now, when you say

r \* r

The output should be,

r: Rational = 4/9

And when you say

r \* 2

The output should be,

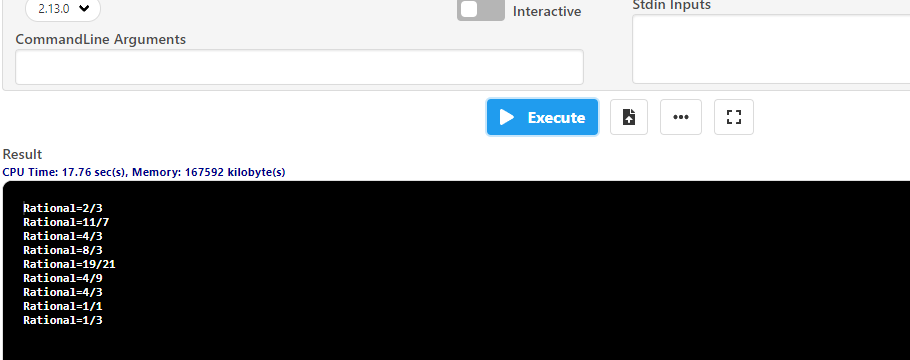
r: Rational = 4/3

So all the operators (+, -, \*, / ) should be overloaded to either take a Rational or a number next to it.

Ans:

**class Rational(n: Int, d: Int) {  
 require(d != 0)  
 private val g = gcd(n.abs, d.abs)  
 val numer = n / g  
 val denom = d / g  
 def this(n: Int) = this(n, 1)  
 def + (that: Rational): Rational =  
 new Rational(  
 numer \* that.denom + that.numer \* denom,  
 denom \* that.denom  
 )  
 def + (i: Int): Rational =  
 new Rational(numer + i \* denom, denom)  
 def - (that: Rational): Rational =  
 new Rational(  
 numer \* that.denom - that.numer \* denom,  
 denom \* that.denom  
 )  
 def - (i: Int): Rational =  
 new Rational(numer - i \* denom, denom)  
 def \* (that: Rational): Rational =  
 new Rational(numer \* that.numer, denom \* that.denom)  
 def \* (i: Int): Rational =  
 new Rational(numer \* i, denom)  
 def / (that: Rational): Rational =  
 new Rational(numer \* that.denom, denom \* that.numer)  
 def / (i: Int): Rational =  
 new Rational(numer, denom \* i)  
 override def toString = "Rational = "+numer +"/"+ denom**

**private def gcd(a: Int, b: Int): Int =   
 if (b == 0) a else gcd(b, a % b)  
 }  
object HelloWorld {  
 def main(args: Array[String]) {  
 val r = new Rational(12, 8)  
 println(r\*r)  
 }  
}**

****

**Requirement 17:**

Write a function in Scala to return the bigger of 2 numbers passed as parameters with the following signature

max: (Int,Int)Int

**Ans: object greater{**

**def main(args:Array[String]){**

**println(max(2,5)+" is greater");**

**}**

**def max(a:Int,b:Int):Int={**

**if(a>b){**

**return a;**

**}**

**else**

**return b;**

**}**

**}**



**Requirement 18:**

Write a Scala program that accepts a set of numbers as Command line parameters and display the sum of all the numbers

For Example, when this program is executed in the Command prompt as,

Scala sum.scala 100 25 200 50

The output should be,

375

**Ans:object sumarg{**

**def main(args:Array[String]){**

**var sum=0;**

**for(arg<-args){**

**sum+=arg.toInt;**

**}**

**println(sum)**

**}**

**}**



**Requirement 19:**

Write a Scala program and introduce 2 list lists with few values. Introduce a tuple named employee with ename and salary. Introduce a set named fruit with the values Apple and Orange. Introduce a map named TaskMap and with 2 tasks - 1. Read Big Data and 2. Understand Map Reduce. Write code to perform the following

1. Combine both the list and put them into the 3rd list and display the 3rd list
2. Display each of the value of the employee tuple
3. Add one more fruit Banana into the set and print the set
4. Add one more task 3.Try HDFS Commands and then print the 2nd task in the Map.

**Ans:**

**import scala.collection.mutable.Set**

**import scala.collection.mutable.Map**

**object testCollections{**

**def main(args:Array[String]){**

**val list1 = List(1,2,3);**

**val list2 = List(4,5,6);**

**val list3 = list1:::list2;**

**println("List3 is: "+list3);**

**val employee = ("Ramesh",30000,"Ganesh",50000);**

**employee.productIterator.foreach{i=>println(i)};**

**var fruit = Set("Apple","Orange")**

**fruit+="Banana"**

**println(fruit)**

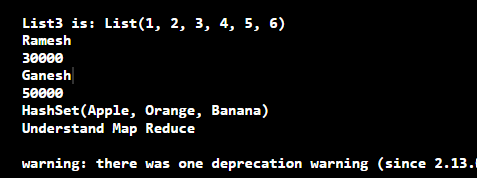
**var TaskMap = Map(1->"Read Big Data and", 2->"Understand Map Reduce");**

**TaskMap+=(3->"Try HDFS Commands")**

**println(TaskMap(2))**

**}**

**}**



**Requirement 20:**

Write a Scala program to find the GCD (Greatest Common Divisor) of 2 numbers. For example, if we consider the following 2 numbers 42 and 14, the GCD should be 7.

**Ans:**

**object findGCD{**

**def main(args:Array[String]){**

**var a=63;**

**var b =42;**

**var r = b;**

**while(a%b!=0){**

**r=a%b;**

**a=b;**

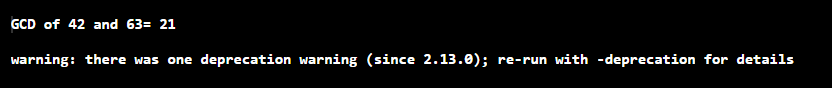
**b=r;**

**}**

**println("GCD of 42 and 63= "+r)**

**}**

**}**



**Requirement 21:**

Write a Scala program to introduce a String variable and reverse the string. For Example, if the string is initialized to ‘Welcome’,

The output should be ‘emocleW’

**Ans:**

**object revString {**

**def reverseString(newString: String): String = {**

**var revString = ""**

**val n = newString.length()**

**for(i <- 0 to n-1){**

**revString = revString.concat(newString.charAt(n-i-1).toString)**

**}**

**return revString**

**}**

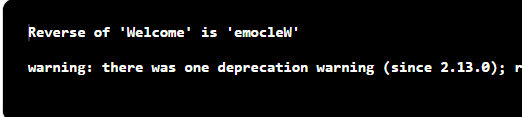
**def main(args: Array[String]) {**

**var newString = "Welcome"**

**println("Reverse of '" + newString + "' is '" + reverseString(newString) + "'")**

**}**

**}**



**Requirement 22:**

Write a Scala program to list the Scala program files in the current directory (File names ends with ‘.scala’)

**import java.io.File**

**def getListOfFiles(dir: String): List[String] = {**

**val file = new File(dir)**

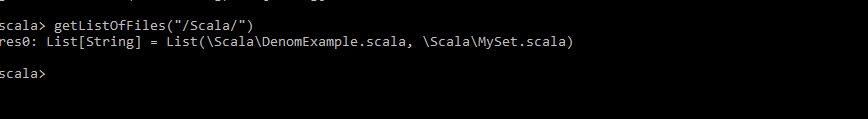
**file.listFiles.filter(\_.isFile)**

**.filter(\_.getName.endsWith(".scala"))**

**.map(\_.getPath).toList**

**}**

**getListOfFiles("Scala/")**

****

**Requirement 23:**

Write a Scala function that takes Amount as parameter to find the minimum number of denominations. For Example, if the amount is 5000, then

2000 \* 2 = 4000

500 \* 2 = 1000

So, minimum no of denomination for 5000 = 4

For Example, if the amount is 8941, then

2000 \* 4 = 8000

500 \* 1 = 500

200 \* 2 = 400

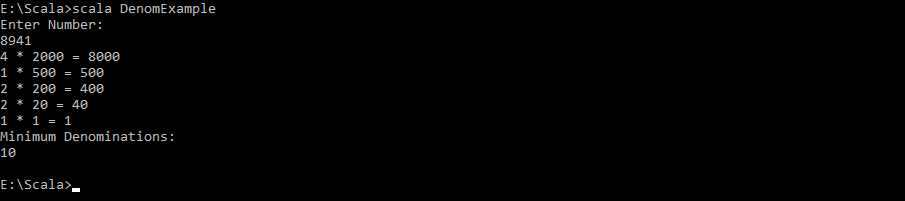
20 \* 2 = 40

1 \* 1 = 1

So, minimum no of denomination for 8941 = 10

Ans:

**object Demo {  
 var z = Array(2000, 1000, 500, 200, 100, 50, 20, 10, 1)  
 var count: Int = 0  
 def denomination(amount: Int) {  
 var temp: Int = amount  
 var den: Int = 0  
 for (i <- 0 to z.length - 1) {  
 count = temp / z(i)  
 if (count >= 1) {  
 println(z(i) + " x " + count + " = " + z(i) \* count);  
 temp = temp - z(i) \* count  
 den = den + count  
 }  
 }  
 print("The minimum number of denomination for "+amount+" ="+den);  
 }  
 def main(args: Array[String]) {  
 denomination(8941)  
 }  
}**

****

**Requirement 24:**

Write a Scala function that takes the number of rows as parameter and generate the Pyramid of numbers for the specified rows.

For Example, if the number of rows is 5, then

1

1 2 1

1 2 3 2 1

1 2 3 4 3 2 1

1 2 3 4 5 4 3 2 1

**def pattern(rows:Int){**

**var k:Int = 0**

**var m:Int = 1**

**var count:Int = 0**

**var count1:Int = 0**

**for (i<-1 to rows) {**

**for (j<-1 to (rows -i)) {**

**print(" ");**

**count=count+1;**

**}**

**m=i**

**while (k != 2 \* i - 1) {**

**if (count <= rows - 1) {**

**print((k+1)+" ");**

**count=count+1;**

**}**

**else {**

**count1=count1+1;**

**print((m-1)+" ");**

**m=m-1**

**}**

**k=k+1;**

**}**

**count1 = 0**

**count = 0**

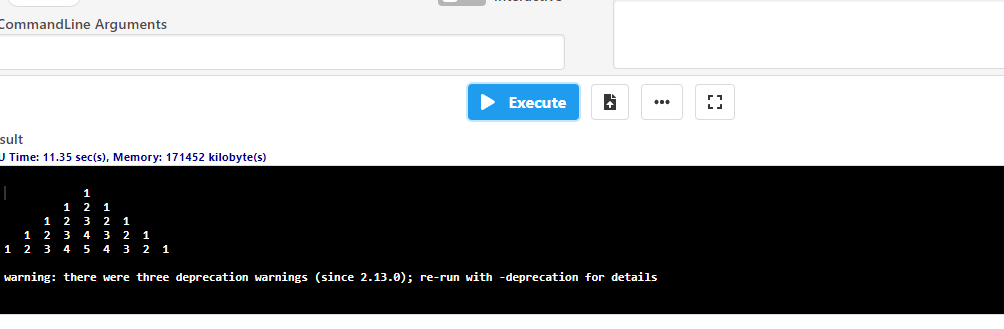
**k = 0;**

**print("\n");**

**}**

**}**

**var p=pattern(5)**

****

**Requirement 25:**

Write a Scala program that takes the filename, include a function to display the longest line in the file. The line number, line and the length of the line should be displayed as follows

LineNo Line length

… … …

**import scala.io.Source**

**def file(){**

**val lines = Source.fromFile("/Scala/hello.text").getLines.toList**

**var min:Int=lines(0).length()**

**var no\_line:Int=0**

**for(i<-1 to lines.length-1){**

**if(lines(i).length()>min){**

**min=lines(i).length()**

**no\_line=i+1**

**}**

**}**

**println("LineNo: "+"Line "+"length")**

**println(no\_line+" "+lines(no\_line-1)+" "+min)**

**}**

**file()**

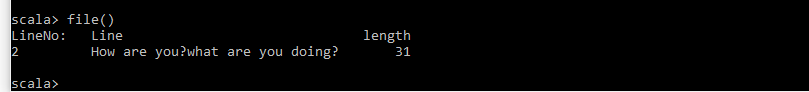
**File’s content:**

hello Arunava.

How are you?what are you doing?

let's play chess.

chess is my favourite game.



**Requirement 26:**

Write a Scala program to introduce an array of numbers. Implement Bubble Sort to arrange the elements of the array in ascending order. Print the sorted array.

**object BubbleSort {**

**def main(args : Array[String]) : Unit = {**

**println("Sorted array:")**

**bubbleSort(Array(50,33,62,21,100)) foreach println**

**}**

**def bubbleSort(a:Array[Int]):Array[Int]={**

**for(i<- 1 to a.length-1){**

**for(j <- (i-1) to 0 by -1){**

**if(a(j)>a(j+1)){**

**val temp=a(j+1)**

**a(j+1)=a(j)**

**a(j)=temp**

**}**

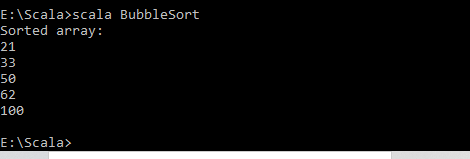
**}**

**}**

**a**

**}**

**}**

****

**Requirement 27:**

Create a list that has 100 values from 1 to 100. From the list, take all the even numbers and load them into the 2nd List. Except the numbers lies between 20 to 50 from the 2nd list, load the rest of the numbers into the 3rd List. Create the 4th list that should has double the values in 3rd list.

**object ListFilter{**

**def main(args : Array[String]) : Unit = {**

**var a=List.range(1,101)**

**println("1st list: "+a)**

**val b=a.filter(x => x%2==0 && (x<20 || x>50))**

**println("2nd list: "+b)**

**val c=a.filter(x => x%2 !=0 || (x>20 && x<50) )**

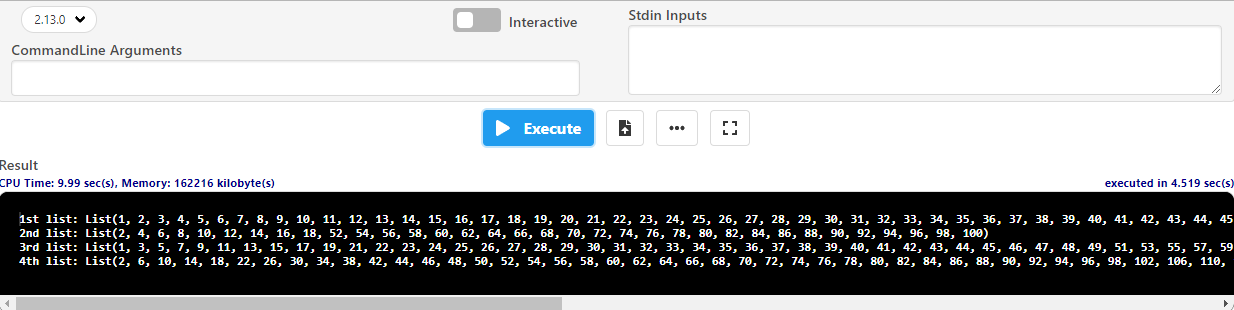
**println("3rd list: "+c)**

**val d=c.map(x=>x\*2)**

**println("4th list: "+d)**

**}**

**}**

****

**Requirement 28:**

Create a Set with 3 elements – 1, 2 and 3. Add a list that has (10, 20, 30). Now add a value 5 to the set and then add 2 more values 6 and 7 to the set. Apply your logic to ensure the Set has only the values greater than 2. If the set has a value 10 then remove the value from the List.

**import scala.collection.mutable.\_**

**object Myset{**

**def main(args : Array[String]) : Unit = {**

**var myset = Set(1,2,3)**

**myset++==List(10,20,30)**

**myset+=5**

**myset+=6**

**myset+=7**

**for(e<-myset){**

**if(e<=2){**

**myset-=e**

**}**

**}**

**println("set contains value greater than 2: "+myset)**

**for(e<-myset){**

**if(e==10){**

**myset-=e**

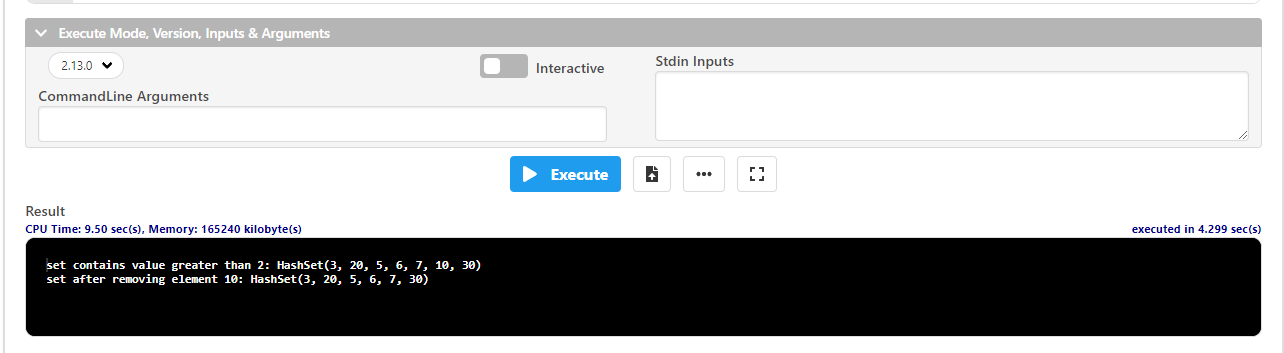
**}**

**}**

**print("set after removing element 10: "+myset)**

**}**

**}**

****