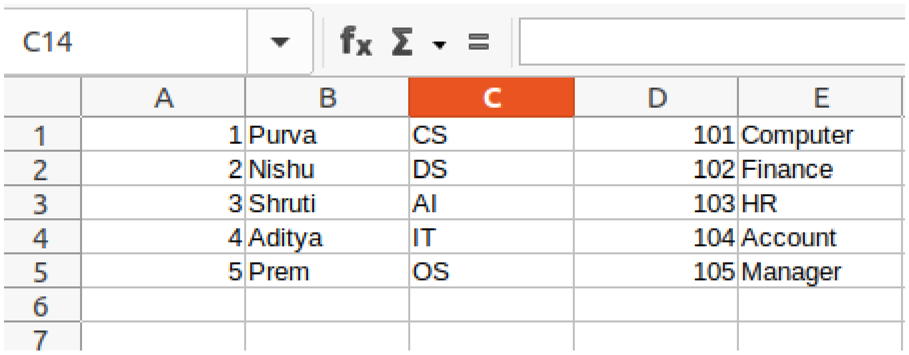
BVIMIT
MCA Sem 3
BDAV

Apache Spark Lab Assignment

Q1. Create the following Text File and perform the operations:

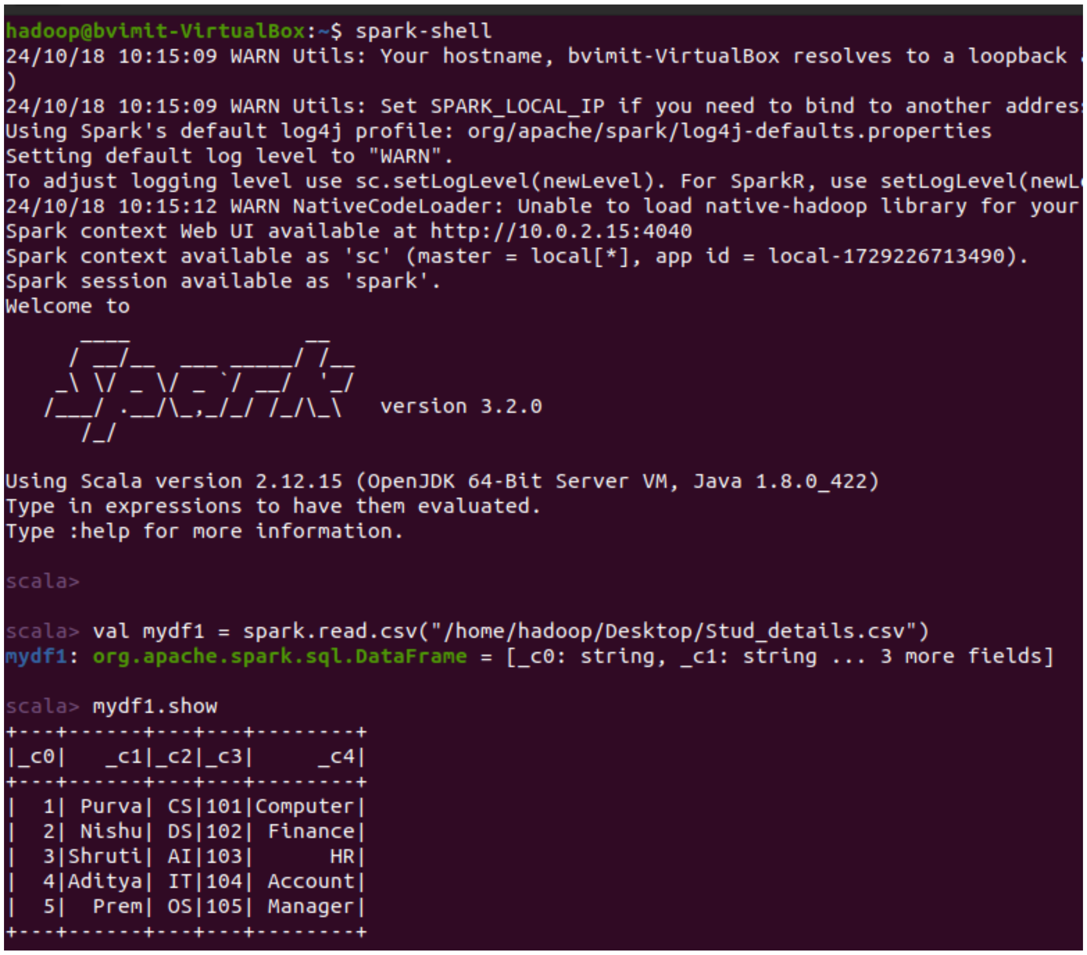
1. Student_details(sid,sname,course,did,dname)



C14

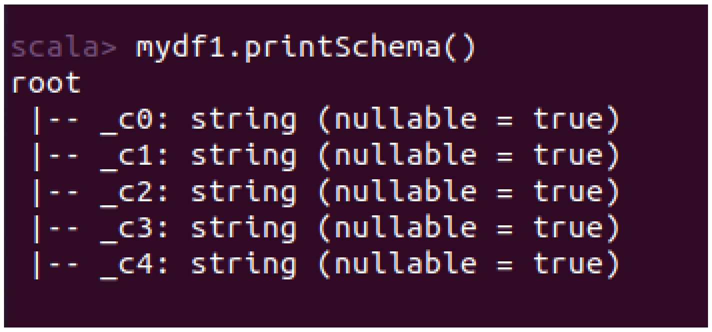
	A	B	C	D	E
1	1	Purva	CS	101	Computer
2	2	Nishu	DS	102	Finance
3	3	Shruti	AI	103	HR
4	4	Aditya	IT	104	Account
5	5	Prem	OS	105	Manager
6					
7					

2. Create a dataframe to read the text file



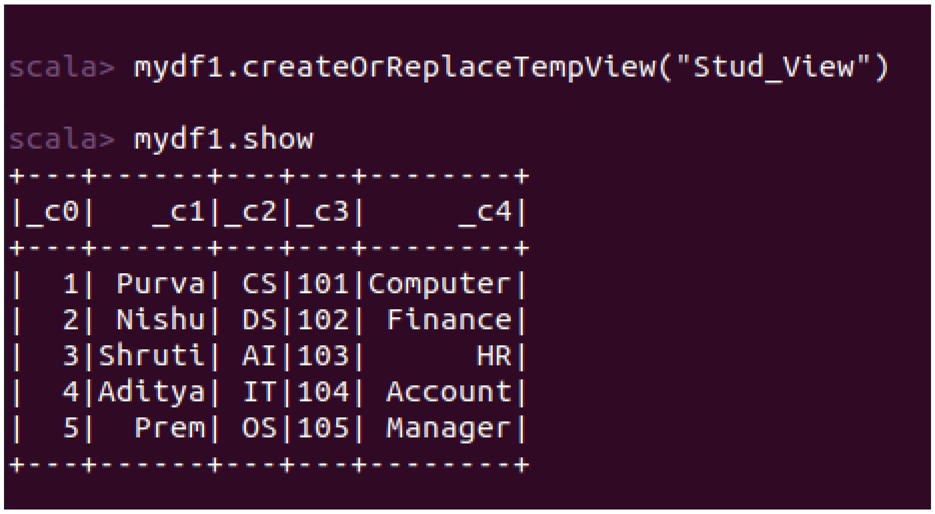
```
hadoop@bvimit-VirtualBox:~$ spark-shell
24/10/18 10:15:09 WARN Utils: Your hostname, bvimit-VirtualBox resolves to a loopback
)
24/10/18 10:15:09 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another addres
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newL
24/10/18 10:15:12 WARN NativeCodeLoader: Unable to load native-hadoop library for your
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-1729226713490).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.2.0
      /_/

Using Scala version 2.12.15 (OpenJDK 64-Bit Server VM, Java 1.8.0_422)
Type in expressions to have them evaluated.
Type :help for more information.

scala>

scala> val mydf1 = spark.read.csv("/home/hadoop/Desktop/Stud_details.csv")
mydf1: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 3 more fields]

scala> mydf1.show
+---+------+---+---+--------+
|_c0|   _c1|_c2|_c3|     _c4|
+---+------+---+---+--------+
|  1| Purva| CS|101|Computer|
|  2| Nishu| DS|102| Finance|
|  3|Shruti| AI|103|      HR|
|  4|Aditya| IT|104| Account|
|  5|  Prem| OS|105| Manager|
+---+------+---+---+--------+
```

3. Display the schema of the dataframe



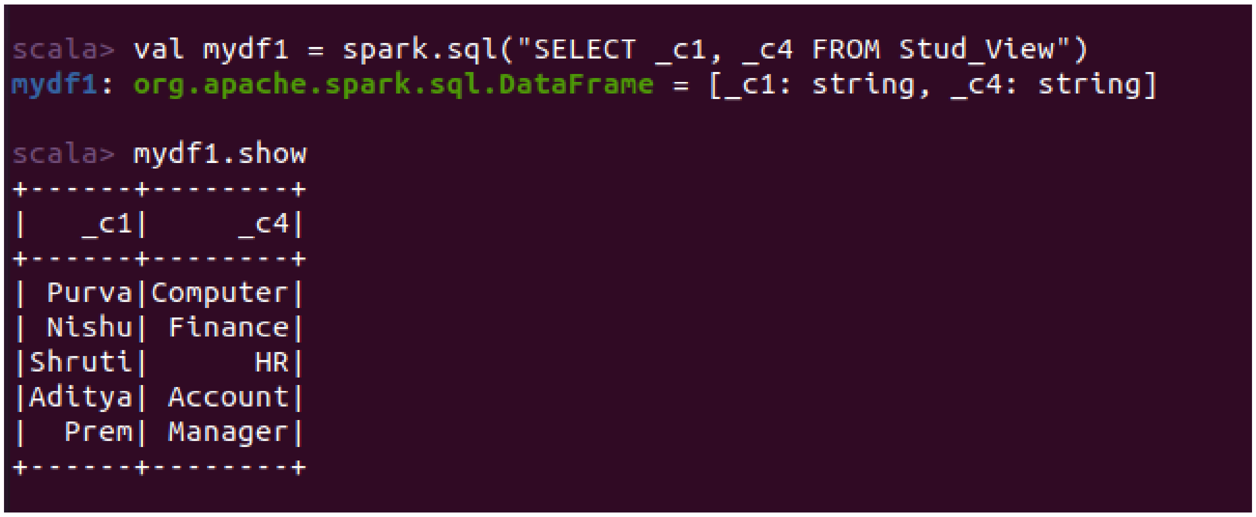
```
scala> mydf1.printSchema()
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
```

4. Create a view “Stud_View” using the above dataframe



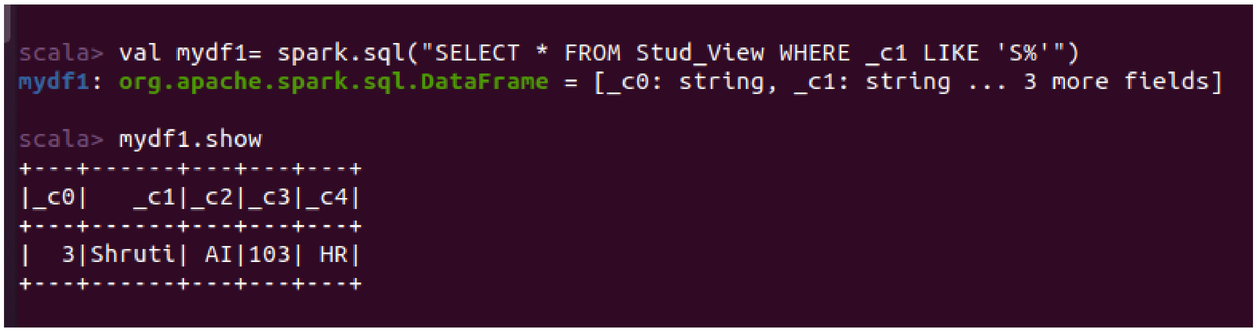
```
scala> mydf1.createOrReplaceTempView("Stud_View")

scala> mydf1.show
+---+------+---+---+--------+
|_c0|   _c1|_c2|_c3|     _c4|
+---+------+---+---+--------+
|  1| Purva| CS|101|Computer|
|  2| Nishu| DS|102| Finance|
|  3|Shruti| AI|103|      HR|
|  4|Aditya| IT|104| Account|
|  5|  Prem| OS|105| Manager|
+---+------+---+---+--------+
```

5. Display student name,dname from the above view



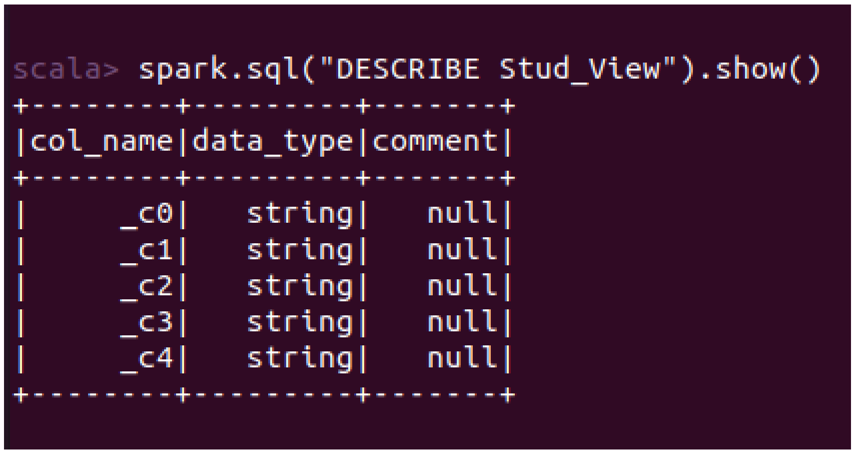
```
scala> val mydf1 = spark.sql("SELECT _c1, _c4 FROM Stud_View")
mydf1: org.apache.spark.sql.DataFrame = [_c1: string, _c4: string]

scala> mydf1.show
+------+--------+
|   _c1|     _c4|
+------+--------+
| Purva|Computer|
| Nishu| Finance|
|Shruti|      HR|
|Aditya| Account|
|  Prem| Manager|
+------+--------+
```

6. Display all the student details where the student name begins with “S”



```
scala> val mydf1= spark.sql("SELECT * FROM Stud_View WHERE _c1 LIKE 'S%'")
mydf1: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 3 more fields]

scala> mydf1.show
+---+------+---+---+---+
|_c0|   _c1|_c2|_c3|_c4|
+---+------+---+---+---+
|  3|Shruti| AI|103| HR|
+---+------+---+---+---+
```

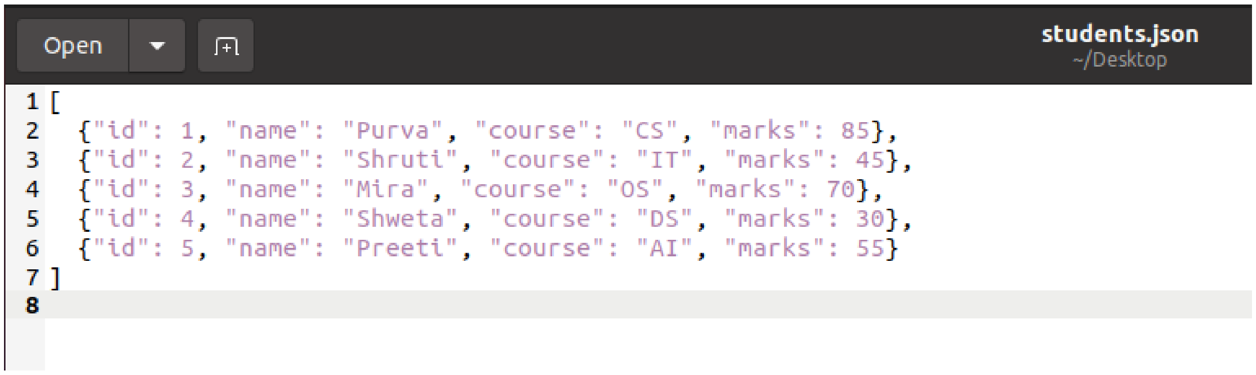
7. Describe the structure of the view



```
scala> spark.sql("DESCRIBE Stud_View").show()
+--------+---------+-------+
|col_name|data_type|comment|
+--------+---------+-------+
|     _c0|   string|   null|
|     _c1|   string|   null|
|     _c2|   string|   null|
|     _c3|   string|   null|
|     _c4|   string|   null|
+--------+---------+-------+
```

Q2. Process the following in Apache Spark:

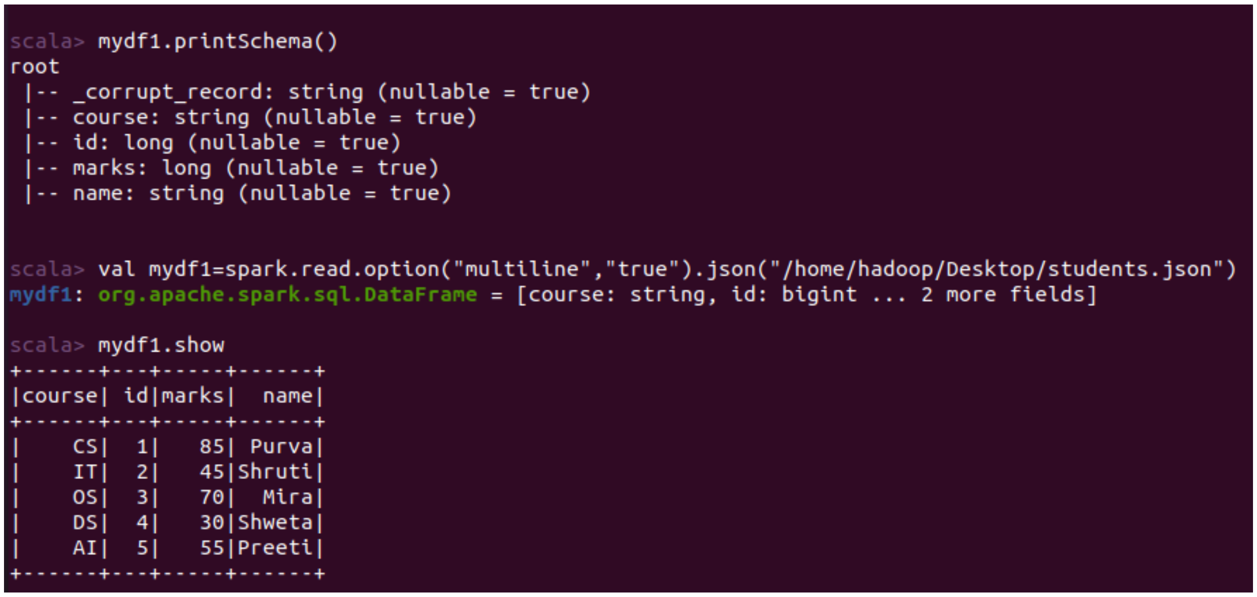
1. Create dataframe from json file which contains student data



students.json
~/Desktop

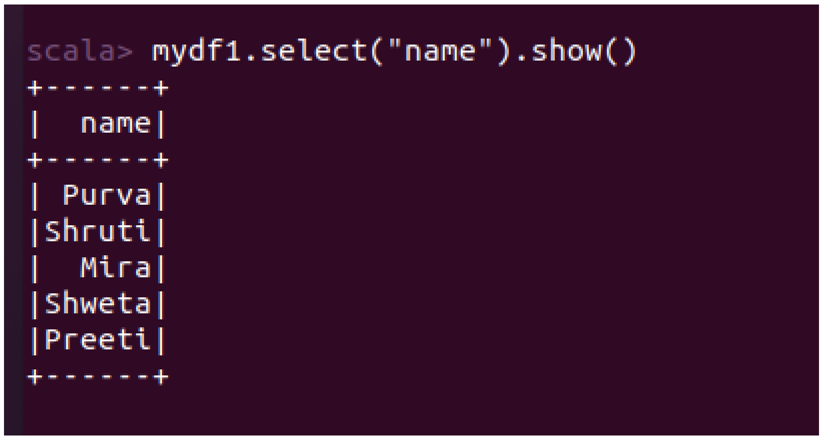
```
[
  {"id": 1, "name": "Purva", "course": "CS", "marks": 85},
  {"id": 2, "name": "Shruti", "course": "IT", "marks": 45},
  {"id": 3, "name": "Mira", "course": "OS", "marks": 70},
  {"id": 4, "name": "Shweta", "course": "DS", "marks": 30},
  {"id": 5, "name": "Preeti", "course": "AI", "marks": 55}
]
```

2. Print the schema in a tree format



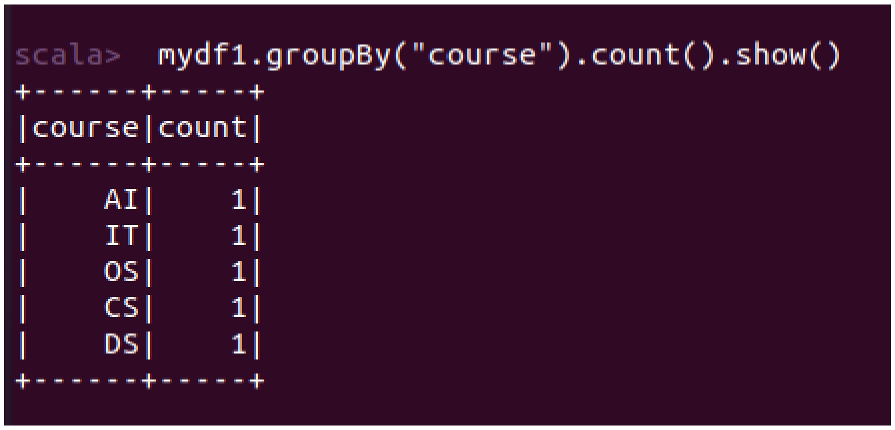
```
scala> mydf1.printSchema()
root
 |-- _corrupt_record: string (nullable = true)
 |-- course: string (nullable = true)
 |-- id: long (nullable = true)
 |-- marks: long (nullable = true)
 |-- name: string (nullable = true)


scala> val mydf1=spark.read.option("multiline","true").json("/home/hadoop/Desktop/students.json")
mydf1: org.apache.spark.sql.DataFrame = [course: string, id: bigint ... 2 more fields]

scala> mydf1.show
+------+---+-----+------+
|course| id|marks|  name|
+------+---+-----+------+
|    CS|  1|   85| Purva|
|    IT|  2|   45|Shruti|
|    OS|  3|   70|  Mira|
|    DS|  4|   30|Shweta|
|    AI|  5|   55|Preeti|
+------+---+-----+------+
```

3. Select only the "name" column



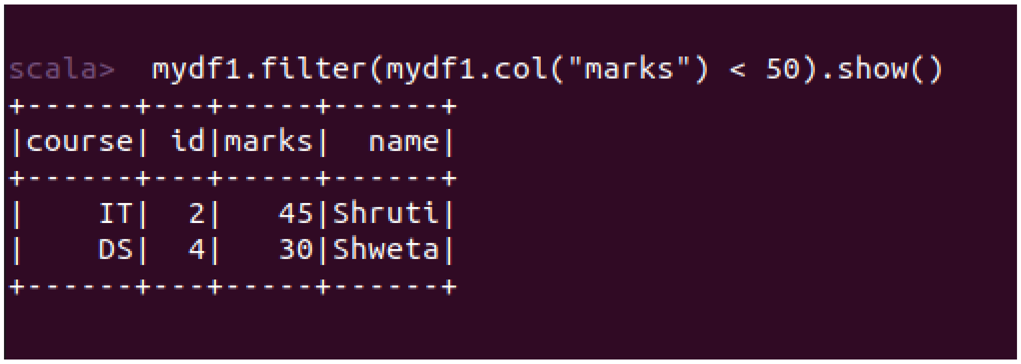
```
scala> mydf1.select("name").show()
+------+
|  name|
+------+
| Purva|
|Shruti|
|  Mira|
|Shweta|
|Preeti|
+------+
```

4. Count students by their course



```
scala>  mydf1.groupBy("course").count().show()
+------+-----+
|course|count|
+------+-----+
|    AI|    1|
|    IT|    1|
|    OS|    1|
|    CS|    1|
|    DS|    1|
+------+-----+
```

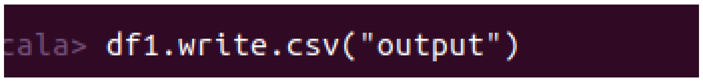
5. Display students having marks less than 50



```
scala>  mydf1.filter(mydf1.col("marks") < 50).show()
+------+---+-----+------+
|course| id|marks|  name|
+------+---+-----+------+
|    IT|  2|   45|Shruti|
|    DS|  4|   30|Shweta|
+------+---+-----+------+
```

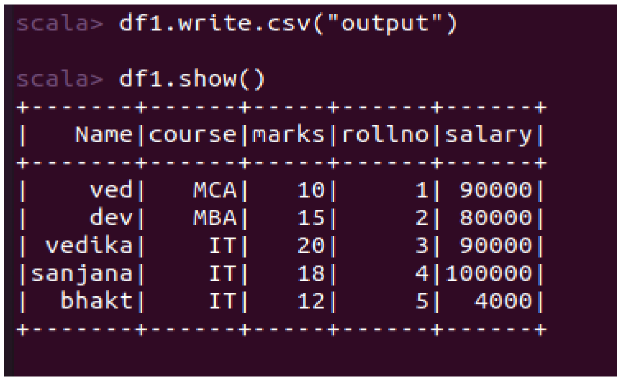
Q3. Process the following in Apache Spark:

1. Consider the Employee.json file and save each of the following output in csv file.



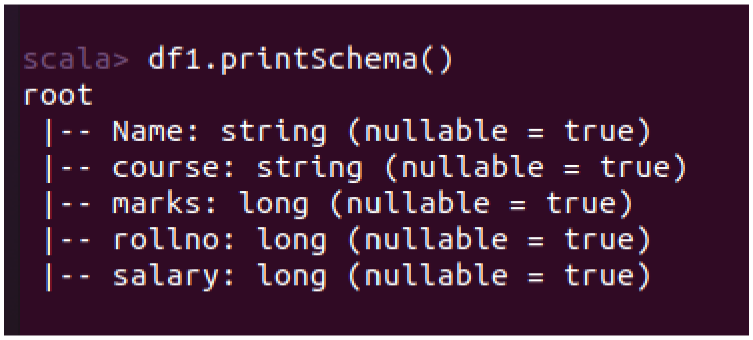
```
cala> df1.write.csv("output")
```

2. Displays the content of the DataFrame to stdout



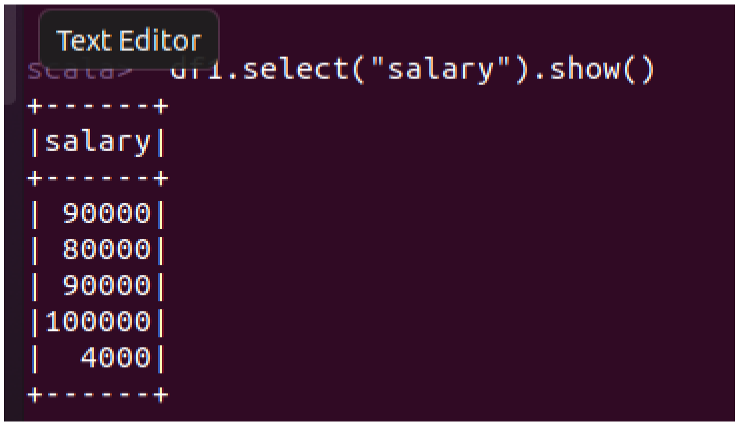
```
scala> df1.write.csv("output")

scala> df1.show()
+-------+------+-----+------+------+
|   Name|course|marks|rollno|salary|
+-------+------+-----+------+------+
|    ved|   MCA|   10|     1| 90000|
|    dev|   MBA|   15|     2| 80000|
| vedika|    IT|   20|     3| 90000|
|sanjana|    IT|   18|     4|100000|
|  bhakt|    IT|   12|     5|  4000|
+-------+------+-----+------+------+
```

3. Print the schema in a tree format



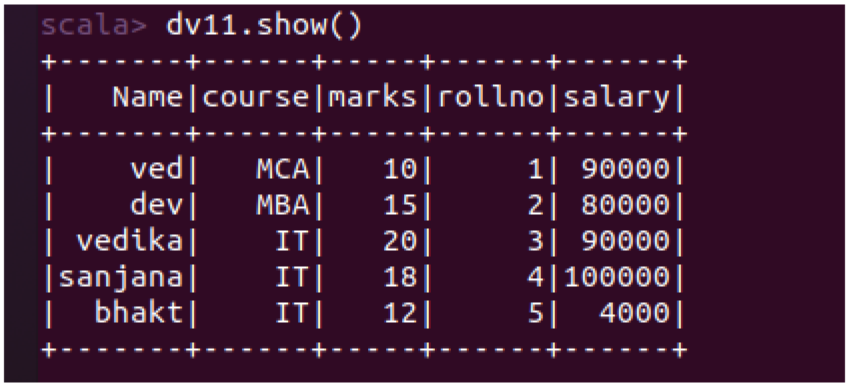
```
scala> df1.printSchema()
root
 |-- Name: string (nullable = true)
 |-- course: string (nullable = true)
 |-- marks: long (nullable = true)
 |-- rollno: long (nullable = true)
 |-- salary: long (nullable = true)
```

4. Select only the “salary" column.



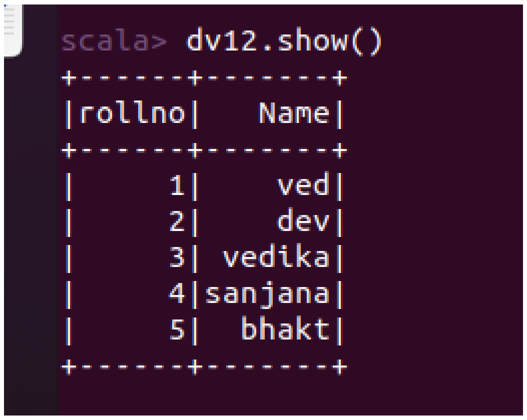
```
scala>  df1.select("salary").show()
+------+
|salary|
+------+
| 90000|
| 80000|
| 90000|
|100000|
|  4000|
+------+
```

5. Register the DataFrame as a SQL temporary view and display all information



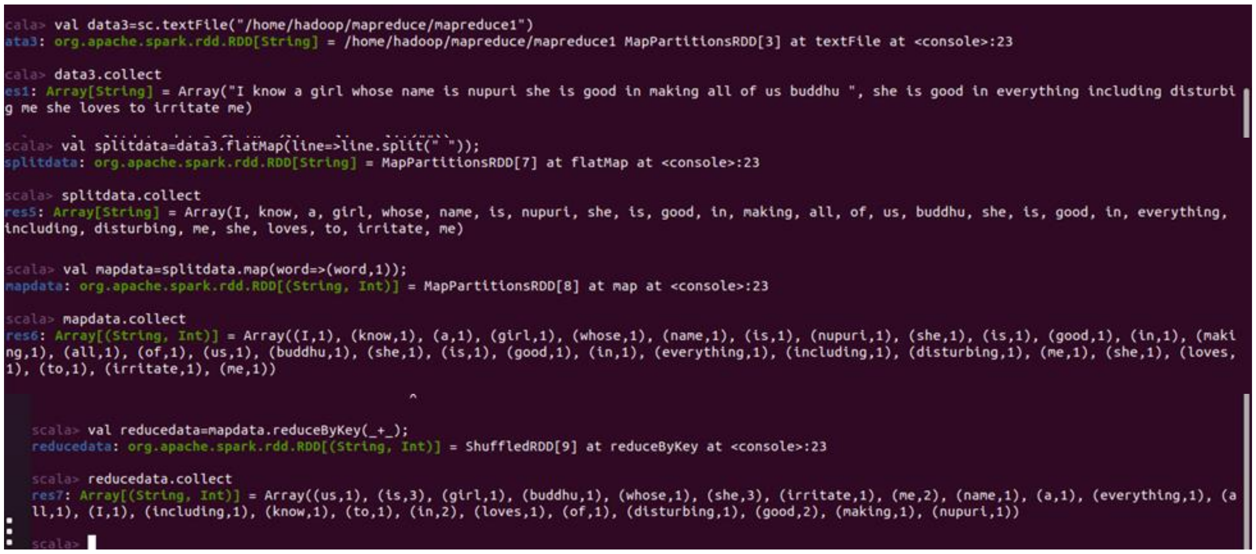
```
scala> dv11.show()
+-------+------+-----+------+------+
|   Name|course|marks|rollno|salary|
+-------+------+-----+------+------+
|    ved|   MCA|   10|     1| 90000|
|    dev|   MBA|   15|     2| 80000|
| vedika|    IT|   20|     3| 90000|
|sanjana|    IT|   18|     4|100000|
|  bhakt|    IT|   12|     5|  4000|
+-------+------+-----+------+------+
```

6. Using the same dataframe display rollno and employee_name from the view



```
scala> dv12.show()
+------+-------+
|rollno|   Name|
+------+-------+
|     1|    ved|
|     2|    dev|
|     3| vedika|
|     4|sanjana|
|     5|  bhakt|
+------+-------+
```

Q4. Implement Word count program in Spark



```
cala> val data3=sc.textFile("/home/hadoop/mapreduce/mapreduce1")
ata3: org.apache.spark.rdd.RDD[String] = /home/hadoop/mapreduce/mapreduce1 MapPartitionsRDD[3] at textFile at <console>:23

cala> data3.collect
es1: Array[String] = Array("I know a girl whose name is nupuri she is good in making all of us buddhu ", she is good in everything including disturbi
g me she loves to irritate me)

scala> val splitdata=data3.flatMap(line=>line.split(" "));
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[7] at flatMap at <console>:23

scala> splitdata.collect
res5: Array[String] = Array(I, know, a, girl, whose, name, is, nupuri, she, is, good, in, making, all, of, us, buddhu, she, is, good, in, everything,
including, disturbing, me, she, loves, to, irritate, me)

scala> val mapdata=splitdata.map(word=>(word,1));
mapdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[8] at map at <console>:23

scala> mapdata.collect
res6: Array[(String, Int)] = Array((I,1), (know,1), (a,1), (girl,1), (whose,1), (name,1), (is,1), (nupuri,1), (she,1), (is,1), (good,1), (in,1), (maki
ng,1), (all,1), (of,1), (us,1), (buddhu,1), (she,1), (is,1), (good,1), (in,1), (everything,1), (including,1), (disturbing,1), (me,1), (she,1), (loves,
1), (to,1), (irritate,1), (me,1))
                                        ^

  scala> val reducedata=mapdata.reduceByKey(_+_);
  reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[9] at reduceByKey at <console>:23

  scala> reducedata.collect
  res7: Array[(String, Int)] = Array((us,1), (is,3), (girl,1), (buddhu,1), (whose,1), (she,3), (irritate,1), (me,2), (name,1), (a,1), (everything,1), (a
  ll,1), (I,1), (including,1), (know,1), (to,1), (in,2), (loves,1), (of,1), (disturbing,1), (good,2), (making,1), (nupuri,1))

  scala>
```

****