Task 1:

```
/*Given a list of numbers - List[Int] (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
- find the sum of all numbers
  - find the total elements in the list
- calculate the average of the numbers in the list
- find the sum of all the even numbers in the list
- find the total number of elements in the list divisible by both 5 and 3 */
```

```
scala> /* List declaration */
     | val nums: List[Int]=List(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15)
nums: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)

scala> /* Sum of all the numbers in the list */
     | nums.reduce(_+_)
res2: Int = 120

scala> /*Total elements in the list*/
     | nums.length
res4: Int = 15

scala> /*Avg no of the element in the List*/
     | nums.reduce(_+_).toFloat/nums.length
res6: Float = 8.0
```

```
scala> /*Sum of all the even number of the list */
     | num.filter(_%2==0).redyce(_+_)
<console>:13: error: not found: value num
       num.filter(_%2==0).redyce(_+_)
       ^

scala> nums.filter(_%2==0).redyce(_+_)
<console>:13: error: value redyce is not a member of List[Int]
       nums.filter(_%2==0).redyce(_+_)
                           ^

scala> nums.filter(_%2==0).reduce(_+_)
res10: Int = 56

scala> /*Total number of element lenght that are divisible by 3 and 5*/
     | nums.filter(x=>(x%5==0)&&(x%3==0)).length
res12: Int = 1
```

Task 2:

1) Pen down the limitations of MapReduce.

2) What is RDD? Explain few features of RDD?

3) List down few Spark RDD operations and explain each of them.

1. Pen down the limitations of MapReduce.

Limitations of MapReduce:

- Cannot handle interactive processing
- Cannot perform real-time (stream) processing
- Cannot do iterative (delta) processing
- Cannot do in-memory processing or cacheing
- Cannot perform graph processing
- It has high latency which may be unsuitable in various applications
- It is not easy to use since it requires complex code for each and every operation
- Most often cannot handle complex machine-learning type algorithms
- With too many keys, sorting may take a very long time

2. What is RDD? Explain few features of RDD?

RDD stands for “Resilient Distributed Dataset” and is a fundamental data structure of Apache Spark.

RDD Features:

- In-memory computation: stores intermediate results in RAM instead of disk
- Lazy evaluations: do not compute right away, but keep track of required
- transformations and compute only at the right time
- Fault tolerance: track data lineage and rebuild lost data automatically upon failure
- Immutability: data is safe to share across processes
- Partitioning: each partition is a logical division of data
- Persistence: user can state which RDD they would want to reuse
- Coarse-grained operations: achieved by map or filter or group-by operations
- Location stickiness: can define placement preference to compute partitions

3. List down few Spark RDD operations and explain each of them.

RDD in Apache Spark supports 2 types of operations –

a. Transformation,

b. Action

RDD Transformations are functions that take an RDD as input and produce one or many RDDs as output via lazy operations.

Narrow Transformations:

Data is from a single partition, and is result of map, filter, Flatmap, MapPartition,Sample and Union functions

Wide or Shuffle Transformations:

Data may live in many partitions of the parent RDD and use functions such as Intersection, Distinct, ReduceByKey, GroupByKey, Join, Cartesian, Repartition and Coalesce RDD Actions returns final result of RDD computations.

It triggers execution using lineage graph to load the data into original RDD, carry out

all intermediate transformations and return final results to Driver program or write it out to file system.

Actions produce non-RDD values.

A few Spark RDD operations:

Transformations:

Map:

Map will take each row as input and return an RDD for the row.

Flat map:

flatMap will take an iterable data as input and returns the RDD as the contents of the iterator.

Filter:

filter returns an RDD which meets the filter condition. Below is the sample demonstration of the above scenario.

ReduceByKey:

reduceByKey takes a pair of key and value pairs and combines all the values for each unique key. Below is the sample demonstration of the above scenario.

Actions:

Collect:

collect is used to return all the elements in the RDD. Refer the below screen shot for the same.

Count:

count is used to return the number of elements in the RDD.

CountByValue:

countByValue is used to count the number of occurrences of the elements in the RDD.

Take:

take will display the number of records we explicitly specify.

Task 3

1. Write a program to read a text file and print the number of rows of data in the document.

2. Write a program to read a text file and print the number of words in the document.

3. We have a document where the word separator is -, instead of space. Write a spark

code, to obtain the count of the total number of words present in the document.

Sample document :

This-is-my-first-assignment.

It-will-count-the-number-of-lines-in-this-document.

The-total-number-of-lines-is-3

1. **Write a program to read a text file and print the number of rows of data in the document.**

```
scala> /* Read the text document and count the no of words*/
     | val textLines=sc.textFile("text.txt")
textLines: org.apache.spark.rdd.RDD[String] = text.txt MapPartitionsRDD[8] at textFile at <console>:25

scala> /*No of rows in the file*/
     | textLines.count()
res14: Long = 3

scala> /*display content of the file*/
     | textLines.foreach(println)
This is my first assignment in the spark today and tommorows.
It will count the number of lines in this document.
The total number of lines is three.
```

2. **Write a program to read a text file and print the number of words in the document**.

```
scala> /* no of words in the file*/
     | val res=textLines.flatMap(line=>line.split(" "))
res: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:26

scala> res
res6: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:26

scala> res.count()
res7: Long = 28
```

3. We have a document where the word separator is -, instead of space. Write a spark code, to obtain the count of the total number of words present in the document.

Sample document :

This-is-my-first-assignment.

It-will-count-the-number-of-lines-in-this-document.

The-total-number-of-lines-is-3

```
scala> /* Read the text document and count the no of words*/
     | val textLines=sc.textFile("text.txt")
textLines: org.apache.spark.rdd.RDD[String] = text.txt MapPartitionsRDD[5] at textFile at <console>:25

scala> /*Read the file and display the contents*/
     | textLines.foreach(println)
This-is-my-first-assignment.
It-will-count-the-number-of-lines-in-this-document.
The-total-number-of-lines-is-3.

scala> /*split the data "-" and count the no of words in the text file*/
     | val allWords=textLines.flatMap(lines=>lines.split("-"))
allWords: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[6] at flatMap at <console>:26

scala> allWords.count()
res9: Long = 22
```