Data Science Assessment: Week 2 – SPARK

# Pre- assessment notes:

# Assessment:

Assessments will be a maximum of 1hr 15min and will include a maximum of 44 questions with a potential mixture of multiple choice, short answer questions and long answer questions.

Assessments will be invigilated by the course tutor; measures will need to be in place to prevent trainees “Googling” answers whilst taking the tests.

# Score:

This exam carries a total of

SECTION Available Awarded

1 20

2 10

3 10

4 10

# SECTION 1: Multiple Choice Section (select 1 answer):

## Question 1: 1 Mark

What of the following is TRUE about Apache SPARK?

1. All of the below.
2. It provides a high-level API only in Java.
3. Hadoop is consistently faster than Apache SPARK.
4. Apache SPARK is a fast and general-purpose open-source cluster computing System.

## Question 2: 1 Mark

Which of the types of loading mechanisms is utilised by Apache SPARK?

1. Eager Loading.
2. Lazy loading.
3. Both of the above depending on the action called.
4. None of the above.

## Question 3: 1 Mark

It is claimed that SPARK can be faster than Hadoop for batch processing, by how much?

1. 10 times faster
2. 20 times faster
3. 100 times faster
4. 200 times faster

## Question 4: 1 Mark

What programming language was SPARK built on top of?

1. .Net
2. Java
3. Python
4. Scala

## Question 5: 1 Mark

In PySpark, what command is used to read in a text file?

1. sc.readText()
2. sc.textFile()
3. text.readFile()
4. spark.textFileRead()

## Question 6: 1 Mark

Listed below are some statements about transformations in SPARK, which statement is true?

1. Transformations are the functions that are applied on an RDD.
2. Filter and Map are applied to each element of the RDD and creates a new RDD.
3. Transformations are not executed until an action is executed.
4. All of these.

## Question 7: 1 Mark

Maps are collections of \_\_\_\_\_.

## key value pairs

## doubles

## arrays

## sets

## Question 8: 1 Mark

Which type of processing is Apache SPARK capable of?

1. Batch Processing
2. Interactive Processing
3. Stream Processing
4. Graph Processing
5. All of the above

## Question 9: 1 Mark

Which of the following statements about SPARK is not true?

1. SPARK can run on the top of Hadoop
2. SPARK can process data stored in HDFS
3. SPARK can use Yarn as resource management layer
4. SPARK cannot be run in a standalone mode.

## Question 10: 1 Mark

Which of the following best describes an RDD?

1. An RDD is a database
2. An RDD is a distributed data structure
3. An RDD is a programming paradigm
4. None of the above

## Question 11: 1 Mark

Which of the following are types of RDD operations?

1. Transformations, Actions, Caching
2. Transformations, Actions, Error Logging
3. Error Flow Redirection, Distributing Results to Intended Recipients, Actions
4. None of the Above

## Question 12: 1 Mark

From the following list, select the correct Transformation(s).

1. Map
2. Filter
3. Join
4. All of the above

## Question 13: 1 Mark

From the following list, select the correct Action(s)

1. Reduce
2. Map
3. Filter
4. None of the above

## Question 14: 1 Mark

Which of the following is TRUE about RDD’s?

1. RDD is immutable.
2. ~~RDD provides one kind of Operation: Actions.~~
3. SPARK rarely needs to use an RDD.
4. All of the above.

## Question 15: 1 Mark

Which of the following characteristics is not shared by both Hadoop and SPARK?

1. Both are data processing platforms
2. Both are cluster computing environments
3. Both have their own file system
4. Both use open source APIs to link between different tools

## Question 16: 1 Mark

Which of the following best describes evaluation in SPARK?

1. All the transformations and actions are lazily evaluated
2. Evaluation starts with the call of Action
3. Evaluation starts with the call of Transformation
4. None of the above

## Question 17: 1 Mark

Which of the following is the base data structure of Spark?

1. DataFrame
2. Dataset
3. RDD
4. None of the above

## Question 18: 1 Mark

RDD’s in SPARK cannot currently be created from data stored on which of the following?

1. LocalFS
2. oracle
3. S3
4. HDFS

## Question 19: 1 Mark

Which component to SPARK enables integration to HDFS?

1. None of the below
2. Computing Engine
3. Storage Manager
4. Cluster Manager

## Question 20: 1 Mark

What is the driver program of SPARK?

1. SPARK Context.
2. Cluster Manager.
3. Worker Node.
4. SPARK Accelerator.

SECTION 2: True or False Questions

## Question 1: 1 Mark

Spark is currently included in every major distribution of Hadoop.

1. TRUE
2. FALSE

## Question 2: 1 Mark

Spark's core is a batch engine.

1. TRUE
2. FALSE

## Question 3: 1 Mark

It is possible to edit the data of RDD, like a conversion to uppercase.

1. TRUE
2. FALSE

## Question 4: 1 Mark

Spark is 100x faster than MapReduce due to In-memory computing

1. TRUE
2. FALSE

## Question 5: 1 Mark

Spark cache the data automatically in the memory as and when needed.

1. TRUE
2. FALSE

## Question 6: 1 Mark

SPARK supplies a high-level API for the RUBY programming language.

1. TRUE
2. FALSE

## Question 7: 1 Mark

It is possible to develop a MapReduce program in spark.

1. TRUE
2. FALSE

## Question 8: 1 Mark

The core abstraction in SPARK is the RDD.

1. TRUE
2. FALSE

## Question 9: 1 Mark

RDD’s are immutable but re-computable.

1. TRUE
2. FALSE

## Question 10: 1 Mark

RDD’s are not fault-tolerant.

1. TRUE
2. FALSE

# SECTION 3: Short Answer Questions: Transformations

## Question 1: 1 Mark

Given the following Pyspark code, what is the output?

trainers = (["Adam", "Stuart", "David"])

trainersRdd = sc.parallelize(trainers)

trainersLengthRdd=trainersRdd.map(lambda x: len(x))

print(trainersLengthRdd.collect())

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [4, 6, 5]

## Question 2: 1 Mark

Given the following Pyspark code, what is the output?

courses = (["ProSkills", "SQL", "PLSQL"])

coursesRdd = sc.parallelize(courses)

someCoursesRdd = coursesRdd.filter(lambda c: c[0] != "P")

print(someCoursesRdd.collect())

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ['SQL']

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Question 3: 1 Mark

Given the following Pyspark code, what is the output?

trainees = sc.parallelize(["Stuart", "Adam"])

score = sc.parallelize([20, 19])

both = trainees.union(score)

print(both.collect())

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ['Stuart', 'Adam', 20, 19]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Question 4: 1 Mark

Given the following Pyspark code, what is the output?

numbersRdd = sc.parallelize([20, 19])

lettersRdd = sc.parallelize(["Stuart", "Adam"])

zippedPairsRdd = lettersRdd.zip(numbersRdd)

print(zippedPairsRdd.collect())

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [('Stuart', 20), ('Adam', 19)]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Question 5: 1 Mark

Given the following Pyspark code, what is the output?

valueRdd = sc.parallelize([1, 2, 3, 4, 3, 2, 1])

uniqueValuesRdd = valueRdd.distinct()

print(uniqueValuesRdd.collect()) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [1, 2, 3, 4]

## Question 6: 1 Mark

Given the following Pyspark code, what is the output?

pairRdd1 = sc.parallelize([("Stuart", 1), ("Adam",2), ("David",3)])

pairRdd2 = sc.parallelize([("Adam", "second"), ("David","third"), ("Bob","fourth")])

joinRdd = pairRdd1.join(pairRdd2)

print(joinRdd.collect())

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [('Adam', (2, 'second')), ('David', (3, 'third'))]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Question 7: 2 Marks

Alter the code from question 6 to output:

[('Adam', (2, 'second')),

('David', (3, 'third')),

('Stuart', (1, None))]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ pairRdd1 = sc.parallelize([("Stuart", 1), ("Adam",2), ("David",3)])

pairRdd2 = sc.parallelize([("Adam", "second"), ("David","third"), ("Bob","fourth")])

joinRdd = pairRdd1.leftOuterJoin(pairRdd2)

print(joinRdd.collect())

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Question 8: 2 Marks

If I have a list of numbers 10, 5, 8, 2 write the Pyspark code to sort these in ascending order.

Numbers = sc.parallelize([10,5,8,2])

sortNumbers = Numbers.sortBy(lambda x: x)

print(sortNumbers.collect())

# SECTION 4: Short Answer Questions: Actions

## Question 1: 2 Mark

Why should you be cautious calling the collect() method action?

\_\_\_will show everything, which uses a lot of memory so it is better to be specific and use select.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Question 2: 1 Mark

Given the following code, what is the output?

namesRdd = sc.parallelize(["Stuart", "Adam", "David"])

namesCount = namesRdd.count()

print(namesCount)

3

## Question 3: 2 Mark

Briefly describe what the takeOrdered () method does.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ returns the new RDD from an existing RDD with elements in a sorting order

## Question 4: 1 Mark

Given the following code, what is the output?

scoresRdd = sc.parallelize([78, 99, 23, 38, 99, 10])

scores = scoresRdd.take(scoresRdd.count())

print(scores)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [78, 99, 23, 38, 99, 10]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Question 5: 2 Mark

Modify the action in question 4 to return the sum and mean of the values.

scoresRdd = sc.parallelize([78, 99, 23, 38, 99, 10])

scores = scoresRdd.sum()

scoresM = scoresRdd.mean()

print(scores, scoresM)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Question 6: 2 Mark

Modify the action in question 4 to return the 2 largest and smallest numbers.

high=scoresRdd.sortby(lambda x: x).take2

low= scoresRdd.sortby(lambda x:-x).take2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# END OF EXAM