

## ASSIGNMENT-6

(6<sup>th</sup> JUNE 2022)

NAME – MOHANA LIKHITHA THOTAKURA

ROLLNO – DXC-262AB-1219

BATCH – DXC-262-ANALYTICS-B12-AZURE

COMPANY – DXC TECHNOLOGY

EMPLOYEE DOMAIN – AZURE ANALYTICS

TRAINER NAME – MR. AJAY KUMAR

DATE OF SUBMISSION – 6 JUNE 2022

NO.OF QUESTIONS: 9

1. Explain what is in-Memory computation in detail?

In-Memory computation is a concept of increasing the processing of data. This works by keeping the data which is extensively used by applications like reporting , preprocessing , big data processing in the RAM itself so through the processing speed of data is switched to maximum.

2. Explain advantages of Spark framework ?

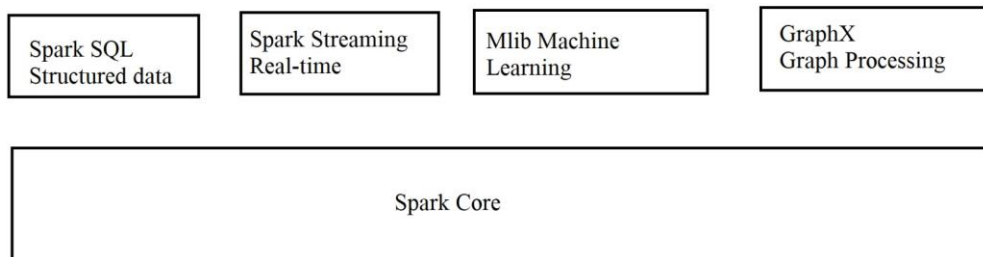
Spark framework is an open source distributed and computing data processing process for big data analysis.

Advantages

- Spark is specially designed to process big data therefore it's fast in processing huge chunks of data.
- Supported and available in multiple languages like Python , Java , Scala , R etc
- Apache Spark carries easy-to-use APIs for operating on large datasets. It offers over 80 high-level operators that make it easy to build parallel apps.
- Apache Spark can handle many analytics challenges because of its low-latency in-memory data processing capability. It has well-built libraries for graph analytics algorithms and machine learning.

3. Explain components of Spark with block diagrams?

Components of Apache Spark:



4. Explain benefits of in-Memory computation ?

The benefits of in-Memory computation are it increases the processing speed of data as the data is directly being stored in the RAM itself instead of slow hard drives therefore the speed of data processing and analysing increases.

5. Explain major differences between Hadoop & Spark ?

Hadoop	Spark
<ol style="list-style-type: none"><li>1. Hadoop is an open source framework which uses Map Reduce Algorithm</li><li>2. Hadoop is slow as it reads data from hard disks.</li><li>3. Hadoop is designed to handle batch processing efficiently</li></ol>	<ol style="list-style-type: none"><li>1. Spark is a lightning fast clustering algorithm , it is an extension of Hadoop's Map Reduce algorithm and gives more types of computation.</li><li>2. Spark is fast as it uses in-memory data storing technique.</li><li>3. Spark is designed to handle real-time data efficiently.</li></ol>

6. Explain features of Spark?

- Speed — Spark runs up to 100 times faster than Hadoop MapReduce for large-scale data processing. It is also able to achieve this speed through controlled partitioning.
- Powerful Caching — Simple programming layer provides powerful caching and disk persistence capabilities.
- Deployment — It can be deployed through Mesos, Hadoop via YARN, or Spark's own cluster manager.

- Real-Time — It offers Real-time computation & low latency because of in-memory computation.
- Polyglot — Spark provides high-level APIs in Java, Scala, Python, and R. Spark code can be written in any of these four languages. It also provides a shell in Scala and Python.

## 7. Write a Py-Spark program to create Dataframe from RDD & explain with screenshots & steps ?

For creating a RDD data frame here we are going to use Python language using Jupyter notebook to create a sample RDD dataframe. Firstly we will install the pyspark library into the environment or in your system.

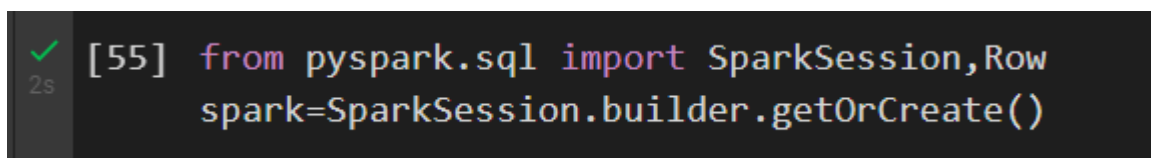


```

✓ ipip install pyspark
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyspark
  Downloading pyspark-3.2.1.tar.gz (281.4 MB)
    | 281.4 MB 32 kB/s
Collecting py4j==0.10.9.3
  Downloading py4j-0.10.9.3-py2.py3-none-any.whl (198 kB)
    | 198 kB 48.1 MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.2.1-py2.py3-none-any.whl size=281853642 sha256=8f4e27f237d9d1f2a4c0eba30e7748481a0082014dd5c370ed5a0c41fd922a59
  Stored in directory: /root/.cache/pip/wheels/9f/f5/07/7cd8017084dce4e93e84e92efd1e1d5334db05f2e83bcef74f
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.3 pyspark-3.2.1

```

After pyspark is installed we will import SparkSession and Row from spark module which is used to create the spark session and Row is used to define a row of dataframe

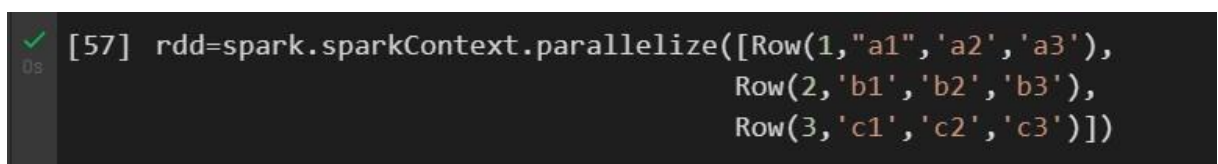


```

✓ 2s [55] from pyspark.sql import SparkSession, Row
      spark=SparkSession.builder.getOrCreate()

```

After the session/instance is created now we need to create a RDD instance of the data

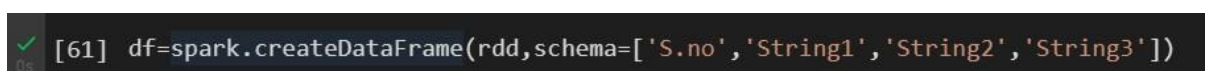


```

✓ 0s [57] rdd=spark.sparkContext.parallelize([Row(1,"a1",'a2','a3'),
      Row(2,'b1','b2','b3'),
      Row(3,'c1','c2','c3')])

```

After the process is done now we will finally create the dataframe



```

✓ 0s [61] df=spark.createDataFrame(rdd,schema=['S.no','String1','String2','String3'])

```

Printing the Dataframe in a tabular form

```
✓ 0s df.show()

+----+-----+-----+-----+
|S.no|String1|String2|String3|
+----+-----+-----+-----+
|  1  |    a1  |    a2  |    a3  |
|  2  |    b1  |    b2  |    b3  |
|  3  |    c1  |    c2  |    c3  |
+----+-----+-----+-----+
```

8. Explain what is RDD & why it is needed ?

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.

Formally, an RDD is a read-only, partitioned collection of records. RDDs can be created through deterministic operations on either data on stable storage or other RDDs. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

There are two ways to create RDDs – parallelizing an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.

9. Write a Py-Spark program to make the column in Upper case & explain with screenshots & steps ?

Let's take the example from 7th question and try to make String1 into upper case . For creating into upper case we need to import module named upper from spark.sql.functions

```
✓ 0s [64] from pyspark.sql.functions import upper
```

After module is loaded now we can just print the data in upper case as

✓

0s

▶

```
df.select(df.S_no,upper(df.String1),"String2","String3").show()
```

S_no	upper(String1)	String2	String3
1	A1	a2	a3
2	B1	b2	b3
3	C1	c2	c3