

## Assignment 5 June 2022

NAME: **ASHOK KUMAR**

ROLL NUMBER: DXC-262-AB-1233

BATCH: DXC-262-ANALYTICS-B12-AZURE

COMPANY: DXC TECHNOLOGY

EMPLOYEE DOMAIN: AZURE ANALYTICS

TRAINING UNDER: MANIPAL PRO LEARN

TRAINER NAME: MR. AJAY KUMAR

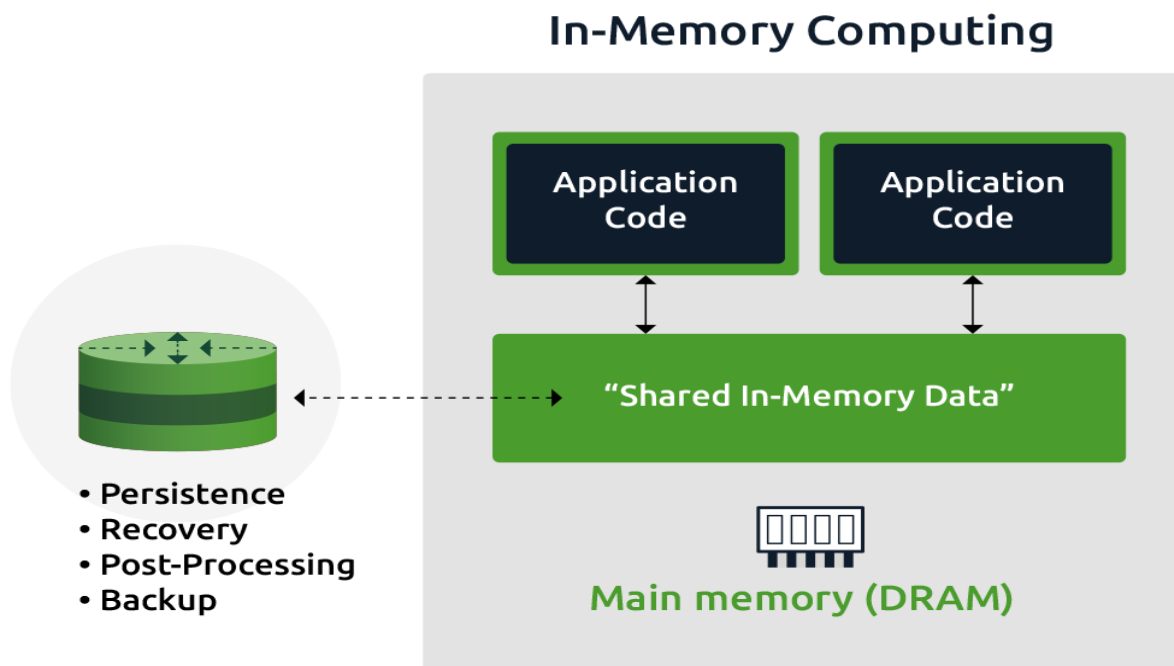
DATE OF SUBMISSION: 5 June 2022

NO. OF Questions: 9

### QUESTION1. Explain what is in-Memory computation in details?

**ANSWER:** In-memory computing is the storage of information in the main random-access memory (RAM) of dedicated servers rather than in complicated relational databases operating on comparatively slow disk drives. In-memory computing helps business customers, including retailers, banks and utilities, to quickly detect patterns, analyze massive data volumes on the fly, and perform their operations quickly. The drop in memory prices in the present market is a major factor contributing to the increasing popularity of in-memory computing technology. This has made in-memory computing economical among a wide variety of applications.

Many technology companies are making use of this technology. For example, the in-memory computing technology developed by SAP, called High-Speed Analytical Appliance (HANA), uses a technique called sophisticated data compression to store data in the random-access memory. HANA's performance is 10,000 times faster when compared to standard disks, which allows companies to analyze data in a matter of seconds instead of long hours.



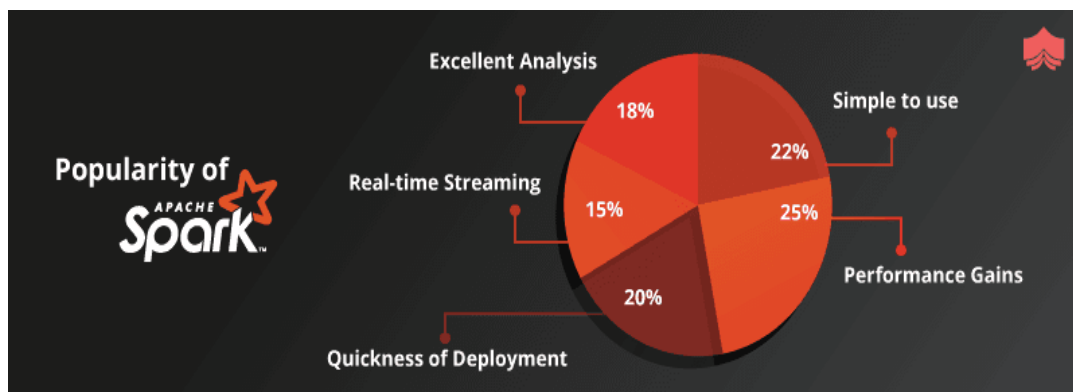
## QUESTION 2. Explain advantages of Spark framework?

**ANSWER:** Apache Spark is one of the hottest new trends in the technology domain. It is the framework with probably the highest potential to realize the fruit of the marriage between Big Data and Machine Learning.

It runs fast (up to 100x faster than traditional Hadoop MapReduce due to in-memory operation, offers robust, distributed, fault-tolerant data objects (called RDD), and integrates beautifully with the world of machine learning and graph analytics through supplementary packages like Mlib and GraphX.

### Benefits of Apache Spark:

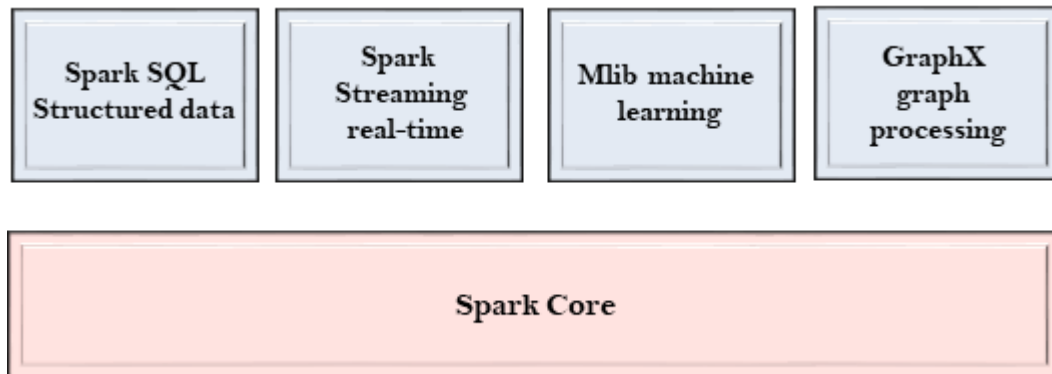
1. **Speed:** When comes to Big Data, processing speed always matters. Apache Spark is wildly popular with data scientists because of its speed. Spark is 100x faster than Hadoop for large scale data processing. Apache Spark uses in-memory computing system whereas Hadoop uses local memory space to store data. Spark can handle multiple petabytes of clustered data of more than 8000 nodes at a time.
2. **Ease of Use:** 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. The below pictorial representation will help you understand the importance of Apache Spark.



3. **Advanced Analytics:** Spark not only supports 'MAP' and 'reduce'. It also supports Machine learning (ML), Graph algorithms, Streaming data, SQL queries, etc.
4. **Dynamic in Nature:** With Apache Spark, you can easily develop parallel applications. Spark offers you over 80 high-level operators.
5. **Apache Spark is powerful:** 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.

### QUESTION 3. Explain components of Spark with block diagram?

**ANSWER:** Components of Spark:



#### 1. Spark Core:

- The Spark Core is the heart of Spark and performs the core functionality.
- It holds the components for task scheduling, fault recovery, interacting with storage systems and memory management.

#### 2. Spark SQL:

- The Spark SQL is built on the top of Spark Core. It provides support for structured data.
- It allows to query the data via SQL (Structured Query Language) as well as the Apache Hive variant of SQL?called the HQL (Hive Query Language).
- It supports JDBC and ODBC connections that establish a relation between Java objects and existing databases, data warehouses and business intelligence tools.
- It also supports various sources of data like Hive tables, Parquet, and JSON.

#### 3. Spark Streaming:

- Spark Streaming is a Spark component that supports scalable and fault-tolerant processing of streaming data.
- It uses Spark Core's fast scheduling capability to perform streaming analytics.
- It accepts data in mini-batches and performs RDD transformations on that data.
- Its design ensures that the applications written for streaming data can be reused to analyze batches of historical data with little modification.
- The log files generated by web servers can be considered as a real-time example of a data stream.

#### 4. MLlib:

- The MLlib is a Machine Learning library that contains various machine learning algorithms.

- These include correlations and hypothesis testing, classification and regression, clustering, and principal component analysis.
- It is nine times faster than the disk-based implementation used by Apache Mahout.

#### 5. GraphX:

- The GraphX is a library that is used to manipulate graphs and perform graph-parallel computations.
- It facilitates to create a directed graph with arbitrary properties attached to each vertex and edge.
- To manipulate graph, it supports various fundamental operators like subgraph, join Vertices, and aggregate Messages.

### **QUESTION 4. Explain benefits of in-Memory computation?**

**ANSWER:** The biggest advantage of in-memory computation is speed. Working from RAM or flash memory removes many of the bottlenecks found in disk-based processing. Thus, businesses are able to analyse large datasets in real-time, which generates better insights from data analytics. Along with better processing speed comes higher storage capacity and better transfer speed. These advantages are possible because data can be stored in in-memory databases, while several processing units (computers) work together to deliver different clusters of data.

Big data comes in two formats, structured and unstructured. In the past, businesses have struggled to store unstructured data like images and videos in conventional databases. However, with in-memory data processing, this is no longer an issue because it is easier to store both structured and unstructured data. Thus, it is easier to get richer and deeper insights from data analytics.

Some other advantages of in-memory computing are:

- The ability to cache countless amounts of data constantly. This ensures extremely fast response times for searches.
- The ability to store session data, allowing for the customization of live sessions and ensuring optimum website performance.
- The ability to process events for improved complex event processing

### QUESTION 5. Explain major difference between Hadoop & Spark?

**ANSWER:** the major difference between Hadoop MapReduce and Spark lies in the approach to processing: Spark can do it in-memory, while Hadoop MapReduce has to read from and write to a disk. As a result, the speed of processing differs significantly – Spark may be up to 100 times faster.

Spark a top-level Apache project focused on processing data in parallel across a cluster, but the biggest difference is that it works in memory. Whereas Hadoop reads and writes files to HDFS, Spark processes data in RAM using a concept known as an RDD, Resilient Distributed Dataset.

Hadoop	Spark
1. Idea: Hadoop is one of the primary frameworks developed based on distributed computing.	1. Idea: Due to the slower processing rendered by Hadoop which is not able to efficiently big data lead to the creation of the spark framework.
2. Design: Hadoop was primarily developed to store and process big data.	2. Design: Spark is primarily developed to offer faster processing.
3. Hadoop shuffles its data between mapper and disk for its successive executions causing slowness in processing.	3. Spark is used to process the data in memory avoiding the data to be shuffled between memory and disk resulting in faster processing.
4. Hadoop offers fault tolerance by keeping copies of data objects in the disk.	4. Spark follows a lineage process where the failed RDD can be recomputed again and doesn't lead to having more data copies in memory to prevent fault tolerance.
5. Hadoop offered map-reduce programs that are very restricted to batch processing.	5. Spark along with batch processing provides support for stream processing and running ML algorithms at one place.
6. Hadoop doesn't offer cache and persists methods extensively as the data is stored in the disk.	6. Spark offers cache and persists with different storage levels as well, where data can exist between memory and disk.

## QUESTION 6. Explain features of Spark?

### ANSWER:

1. **Lightning- fast processing speed:** Big Data processing is all about processing large volumes of complex data. Hence, when it comes to Big Data processing, organizations and enterprises want such frameworks that can process massive amounts of data at high speed. As we mentioned earlier, Spark apps can run up to 100x faster in memory and 10x faster on disk in Hadoop clusters. It relies on Resilient Distributed Dataset (RDD) that allows Spark to transparently store data on memory and read/write it to disc only if needed. This helps to reduce most of the disc read and write time during data processing.
2. **Ease of use:** Spark allows you to write scalable applications in Java, Scala, Python, and R. So, developers get the scope to create and run Spark applications in their preferred programming languages. Moreover, Spark is equipped with a built-in set of over 80 high-level operators. You can use Spark interactively to query data from Scala, Python, R, and SQL shells.
3. **It offers support for sophisticated analytics:** Not only does Spark support simple “map” and “reduce” operations, but it also supports SQL queries, streaming data, and advanced analytics, including ML and graph algorithms. It comes with a powerful stack of libraries such as SQL & DataFrames and MLlib (for ML), GraphX, and Spark Streaming. What’s fascinating is that Spark lets you combine the capabilities of all these libraries within a single workflow/application.
4. **Real-time stream processing:** Spark is designed to handle real-time data streaming. While MapReduce is built to handle and process the data that is already stored in Hadoop clusters, Spark can do both and also manipulate data in real-time via Spark Streaming. Unlike other streaming solutions, Spark Streaming can recover the lost work and deliver the exact semantics out-of-the-box without requiring extra code or configuration. Plus, it also lets you reuse the same code for batch and stream processing and even for joining streaming data to historical data.
5. **It is flexible:** Spark can run independently in cluster mode, and it can also run on Hadoop YARN, Apache Mesos, Kubernetes, and even in the cloud. Furthermore, it can access diverse data sources. For instance, Spark can run on the YARN cluster manager and read any existing Hadoop data. It can read from any Hadoop data sources like HBase, HDFS, Hive, and Cassandra. This aspect of Spark makes it an ideal tool for migrating pure Hadoop applications, provided the apps’ use-case is Spark-friendly.

**QUESTION 7. Write a Py-Spark program to create Data frame from RDD & explain with screenshots & steps?**

**ANSWER:**

**Install pyspark =>** `pip install pyspark`

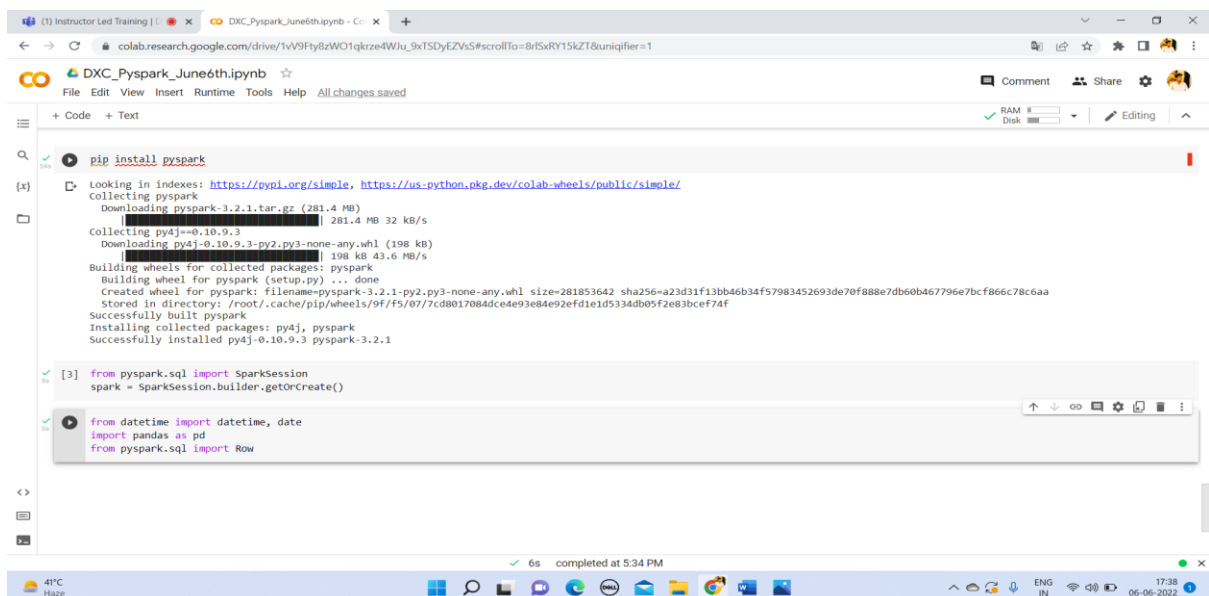
**Import and create a SparkSession =>**

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
```

**import date, datetime (to include date and time in our dataframe), pandas and Row =>**

```
from datetime import datetime, date
import pandas as pd
from pyspark.sql import Row
```



**PySpark dataframe creation using RDD =>**

**#Create PySpark dataframe from RDD consisting of a list of tuples**

```
rdd=spark.sparkContext.parallelize([
    (1,2., 'string1', date(2022,6,6), datetime(2022,6,6,12,30)),
    (2,3., 'string2', date(2022,7,6), datetime(2022,6,7,12,30)),
    (3,4., 'string3', date(2022,8,6), datetime(2022,6,8,12,30)),
])
df=spark.createDataFrame(rdd, schema=['a', 'b', 'c', 'd', 'e'])
df
```

**USE df.show() to see the dataframe.**

(1) Instructor Led Training | DXC\_Pyspark\_June6th.ipynb

colab.research.google.com/drive/1vV9Fy8zWO1qkrze4WJu\_9xtSDyEZv5S#scrollTo=6shCtOZb7ORq&uniquifier=1

DXC\_Pyspark\_June6th.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

✓ [3] spark = SparkSession.builder.getOrCreate()

✓ [5] from datetime import datetime, date  
import pandas as pd  
from pyspark.sql import Row

✓ [6] #PySpark dataframe creation using RDD  
rdd=spark.sparkContext.parallelize([  
    (1,2,,'string1',date(2022,6,6),datetime(2022,6,6,12,30)),  
    (2,3,,'string2',date(2022,7,6),datetime(2022,6,7,12,30)),  
    (3,4,,'string3',date(2022,8,6),datetime(2022,6,8,12,30)),  
])  
df=spark.createDataFrame(rdd,schema=['a','b','c','d','e'])  
df  
  
DataFrame[a: bigint, b: double, c: string, d: date, e: timestamp]

✓ [7] df.show()  

	a	b	c	d	e
1	2.0	string1	2022-06-06	2022-06-06 12:30:00	
2	3.0	string2	2022-07-06	2022-06-07 12:30:00	
3	4.0	string3	2022-08-06	2022-06-08 12:30:00	

1s completed at 5:41 PM

41°C Haze

ENG IN

17:41

06-06-2022



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

**ANSWER:** Over the time, Big Data analysis has reached a new magnitude which, in turn, has changed its mode of operation and expectation as well. Today's big data analysis is not only dealing with massive data but also with a set target of fast turnaround time. Though Hadoop is the unbeatable technology behind the big data analysis, it has some shortfalls concerning fast processing. However, with the entry of Spark in the Hadoop world, data processing speed has met up most expectations.

when we talk about Spark, the first term comes into our mind is Resilient Distributed Datasets (RDD) or Spark RDD which makes data processing faster. Also, this is the key feature of Spark that enables logical partitioning of data sets during computation.

RDD stands for Resilient Distributed Dataset where each of the terms signifies its features.

- Resilient: means it is fault tolerant by using RDD lineage graph (DAG). Hence, it makes it possible to do recomputation in case of node failure.
- Distributed: As datasets for Spark RDD resides in multiple nodes.
- Dataset: records of data that you will work with.

In Hadoop designing, RDD is a challenge. However, with Spark RDD the solution seems very effective due to its lazy evaluation. RDDs in Spark works on-demand basis. Hence, it saves lots of data processing time as well efficiency of the whole process.

Hadoop Map-reduce has many shortcomings which are overcome by Spark RDD through its features, and this is the main reason for the popularity of Spark RDD.

RDD - Resilient Distributed Dataset:

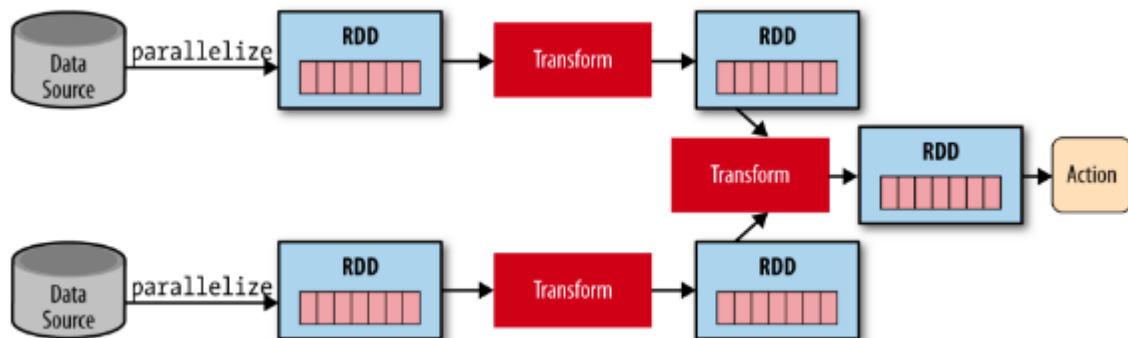
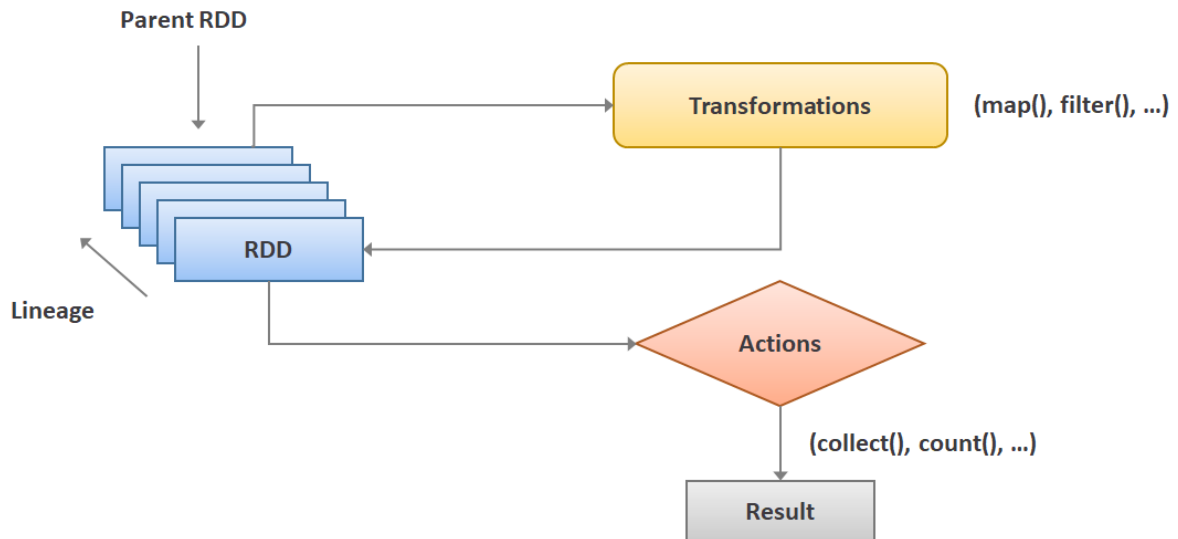
- It is basis building block of Spark,
- The RDD (Resilient Distributed Dataset) is the Spark's core abstraction.
- It is a collection of elements, partitioned across the nodes of the cluster so that we can execute various parallel operations on it.

There are two ways to create RDDs:

- Parallelizing an existing data in the driver program
- Referencing a dataset in an external storage system, such as a shared filesystem, HDFS, HBase, or any data source offering a Hadoop InputFormat.



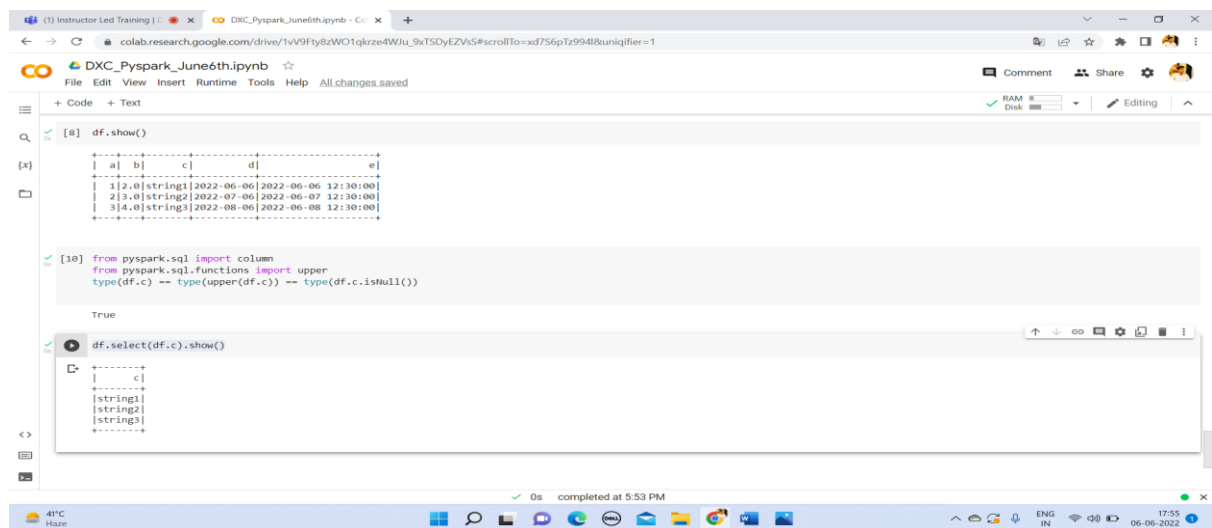
## Spark RDD (Unstructured) Operations



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

**ANSWER: upper() Function** takes up the column name as argument and converts the column to upper case

- ⇒ `from pyspark.sql import column`  
`from pyspark.sql.functions import upper`  
`type(df.c) == type(upper(df.c)) == type(df.c.isNull())`
- ⇒ `df.select(df.c).show()` → this will show c column of df dataframe that we already created



```
[8] df.show()
+-----+
| a | b | c | d | e |
+-----+
| 1 | 2 | 0 | string1 | 2022-06-06 | 2022-06-06 | 12:30:00 |
| 2 | 3 | 0 | string2 | 2022-07-06 | 2022-06-07 | 12:30:00 |
| 3 | 4 | 0 | string3 | 2022-08-06 | 2022-06-08 | 12:30:00 |
+-----+

[10] from pyspark.sql import column
from pyspark.sql.functions import upper
type(df.c) == type(upper(df.c)) == type(df.c.isNull())

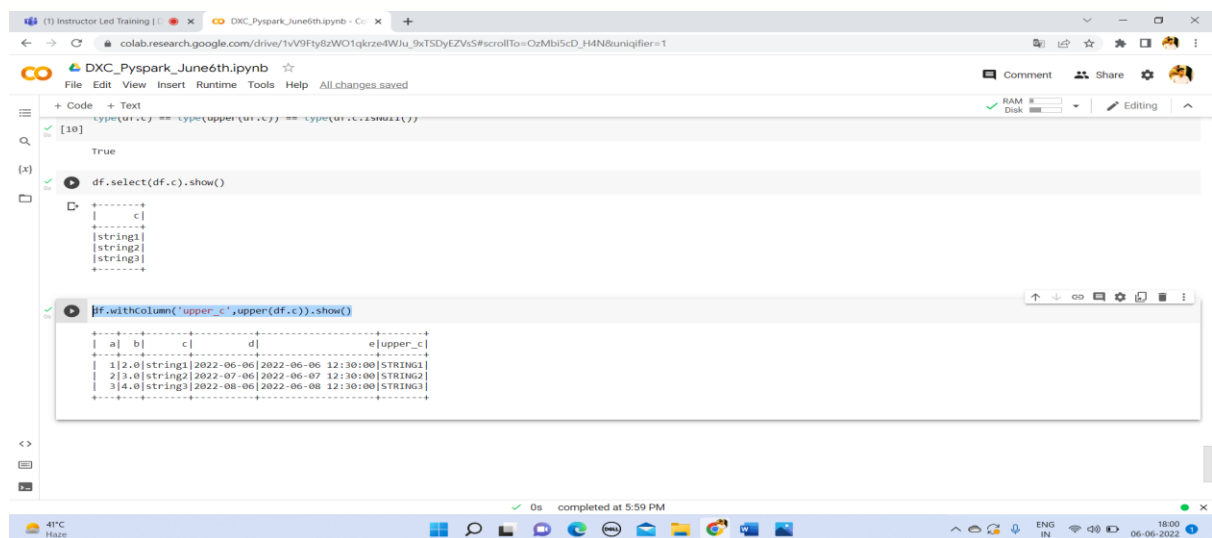
True

df.select(df.c).show()
+-----+
| c |
+-----+
| string1 |
| string2 |
| string3 |
+-----+
```

We can see that **c** column of **df** dataframe is in lowercase.

We will convert this column into uppercase that will save in a new column **upper\_c** of df dataframe.

- ⇒ `df.withColumn('upper_c', upper(df.c)).show()` → this will convert c column into uppercase



```
[10] type(df.c) == type(upper(df.c)) == type(df.c.isNull())

True

df.select(df.c).show()
+-----+
| c |
+-----+
| string1 |
| string2 |
| string3 |
+-----+

df.withColumn('upper_c', upper(df.c)).show()
+-----+
| a | b | c | d | e | upper_c |
+-----+
| 1 | 2 | 0 | string1 | 2022-06-06 | 2022-06-06 | 12:30:00 | STRING1 |
| 2 | 3 | 0 | string2 | 2022-07-06 | 2022-06-07 | 12:30:00 | STRING2 |
| 3 | 4 | 0 | string3 | 2022-08-06 | 2022-06-08 | 12:30:00 | STRING3 |
+-----+
```

***THANK YOU!***  
***ASHOK KUMAR***