unext

Assessment as a Service

~Determine your Tech Quotient

Assessment as a service – Technologies covered

Full Stack Java

Full Stack Dot Net

Full Stack Python

DevOps

Cybersecurity

Embedded Systems

Cloud

Big Data

Data Analytics

AI/ML

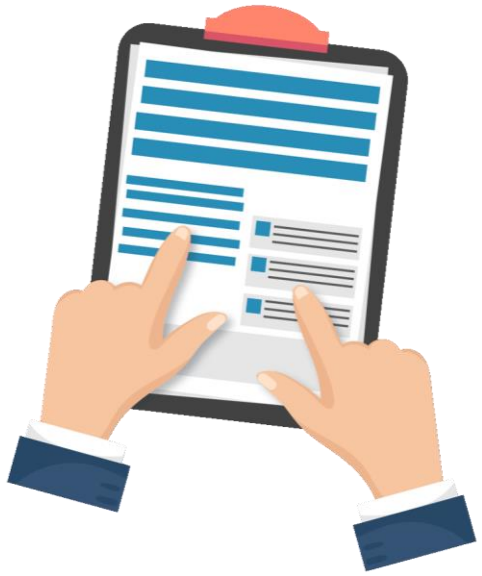
Generative AI

Automation Testing

Note : As per client requirements we are capable on delivering training across various tech domains relevant in the market

Assessment Features

- ❖ Knowledge Based Assessments
- ❖ Skill Based Assessments through our coding platform / sandbox integrated with LMS
- ❖ Coding platform enables auto evaluation
- ❖ AI based remote proctored assessments
- ❖ Coding Platform/Sandbox will support the following technologies
  - Java and .Net Full Stack
  - Python
  - Hadoop
  - PySpark
  - Databases(Oracle, MYSQL)
  - Cloud(AWS,Azure)
  - AI/ML
  - Generative AI



Assessment Design Framework

C – Conceptual Questions

A – Analytical Questions

M – Memory Recall Questions

P – Practise Based Questions

40% Knowledge Based Questions

C	A	M	P
10%	5%	15%	10% (Scenario Based MCQ)

Note: The above distribution will vary based on skill level

60% Skill Based Questions

C	A	M	P
0%	0%	0%	60% (Coding – Hands On)

Skill Levels

★	★ ★	★★★
Beginner	Intermediate	Advanced

unext

Assessment Design Framework

- ❖ Competency definition and skill depth is defined for the respective technology
- ❖ Questions are designed basis the skill depth and CAMP framework
- ❖ Questions will be a combination of Knowledge Based (40%) and Skill Based (60%)

Desired Qualification Criteria

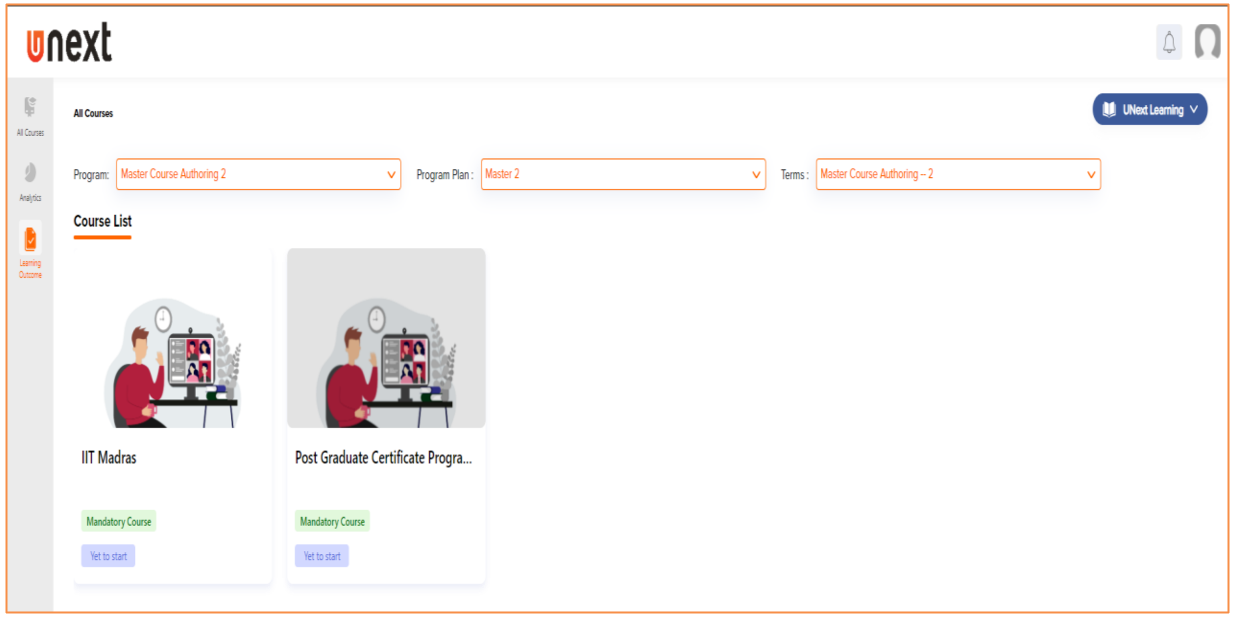
- ❖ (40 % weightage in knowledge based + 60 % weightage in skill based) >= 70%

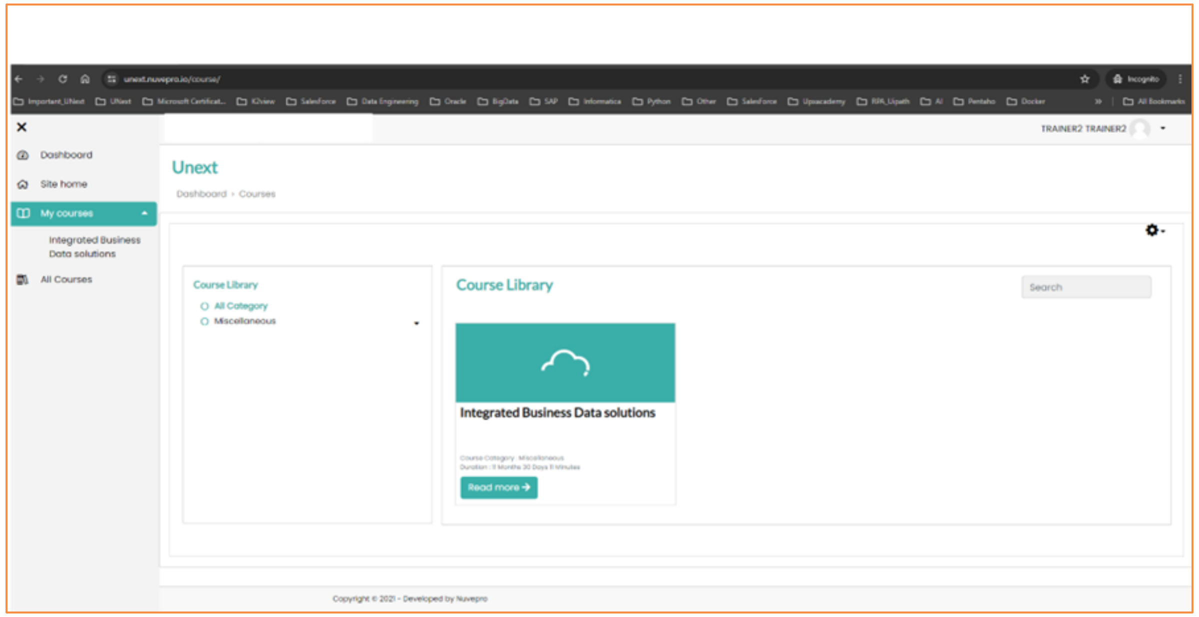
unext

Assessments will be driven through

UNext LMS

UNext Coding Platform / Sandbox





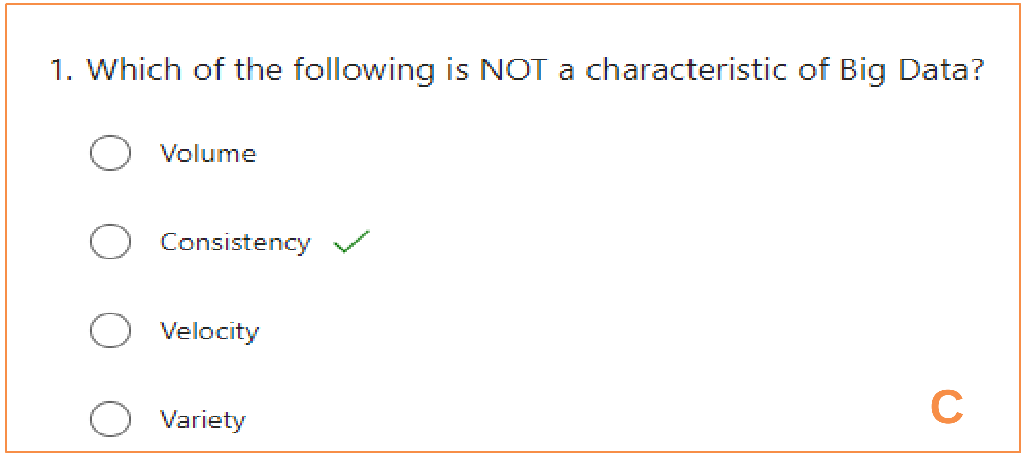
Knowledge Based

Skill Based

Sample Competency Framework for Big Data

Skill Depth	Beginner	Intermediate	Advanced
Big Data Competencies	Hadoop basics	Advanced understanding of Hadoop ecosystem	Proficiency in PySpark programming
	Understanding of HDFS	Proficiency in MapReduce programming paradigm	Ability to work with large-scale data processing frameworks like Apache Spark
	Basic data processing with Hadoop ecosystem tools	Familiarity with Hive, Sqoop, and HBase	Knowledge of Spark SQL, DataFrame API, and RDDs
	Importing and exporting data between Hadoop and relational databases	Experience with data ingestion and ETL processes Introduction to Apache Spark	Familiarity with related technologies like Apache Kafka, Apache Flink, and Apache Beam

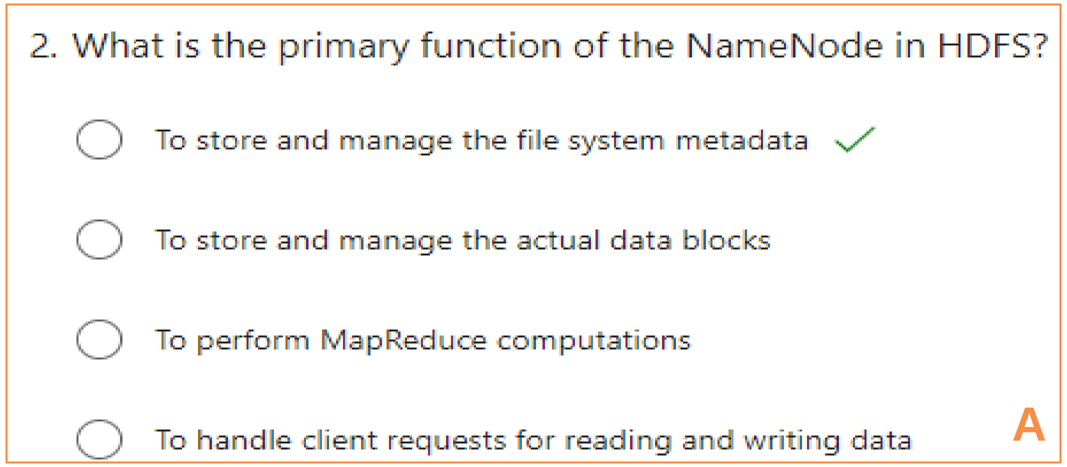
Knowledge Based - Beginner Skill Depth (CAMP)



1. Which of the following is NOT a characteristic of Big Data?

- ◯ Volume
- ◯ Consistency ✓
- ◯ Velocity
- ◯ Variety

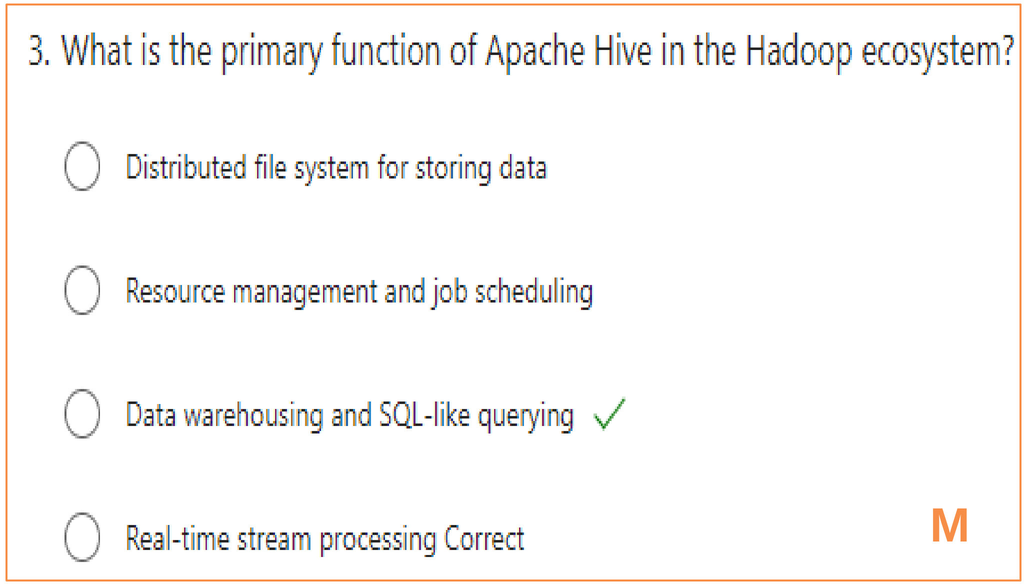
C



2. What is the primary function of the NameNode in HDFS?

- ◯ To store and manage the file system metadata ✓
- ◯ To store and manage the actual data blocks
- ◯ To perform MapReduce computations
- ◯ To handle client requests for reading and writing data

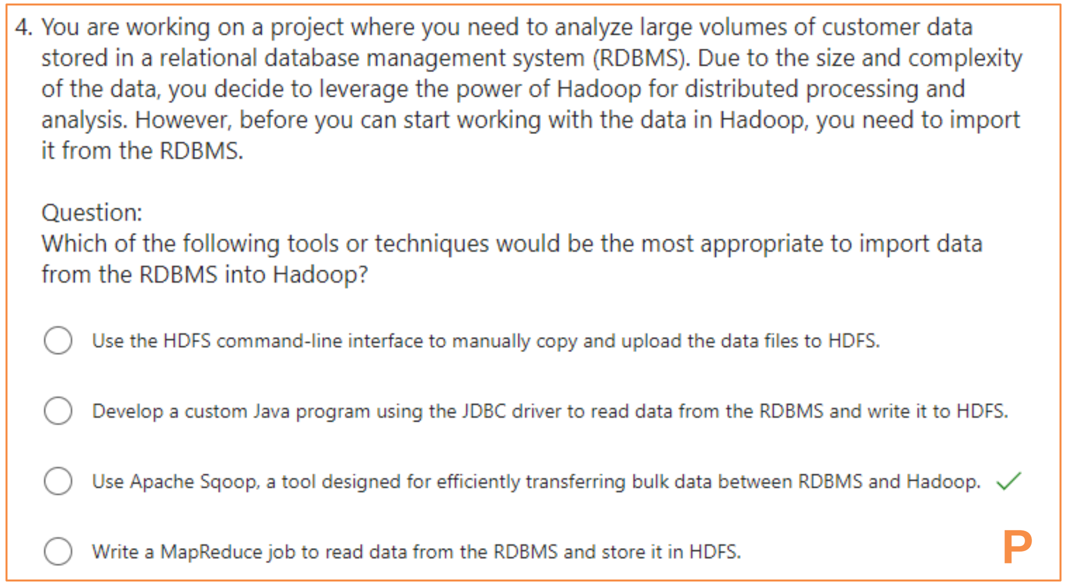
A



3. What is the primary function of Apache Hive in the Hadoop ecosystem?

- ◯ Distributed file system for storing data
- ◯ Resource management and job scheduling
- ◯ Data warehousing and SQL-like querying ✓
- ◯ Real-time stream processing Correct

M



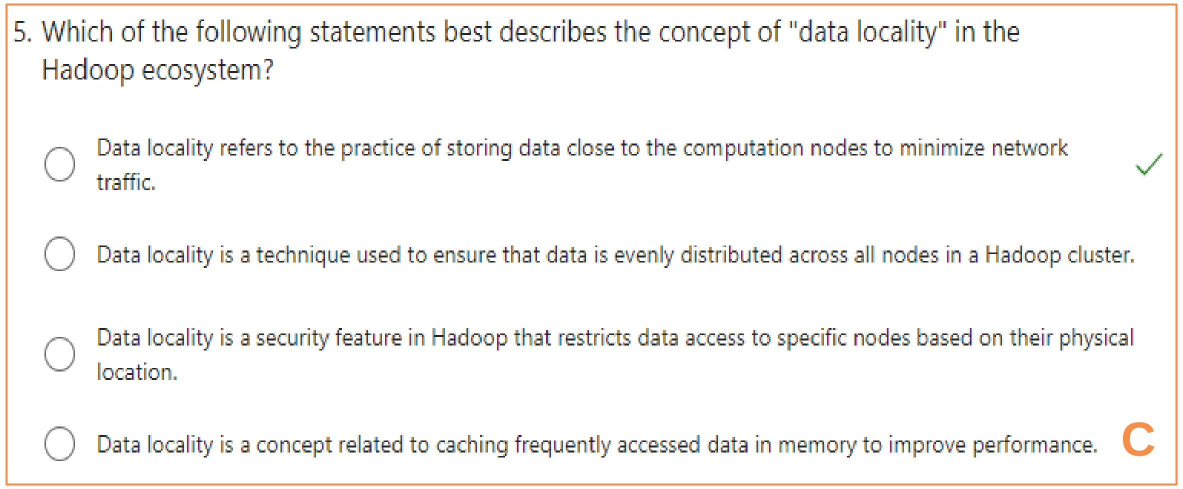
4. You are working on a project where you need to analyze large volumes of customer data stored in a relational database management system (RDBMS). Due to the size and complexity of the data, you decide to leverage the power of Hadoop for distributed processing and analysis. However, before you can start working with the data in Hadoop, you need to import it from the RDBMS.

Question:
Which of the following tools or techniques would be the most appropriate to import data from the RDBMS into Hadoop?

- ◯ Use the HDFS command-line interface to manually copy and upload the data files to HDFS.
- ◯ Develop a custom Java program using the JDBC driver to read data from the RDBMS and write it to HDFS.
- ◯ Use Apache Sqoop, a tool designed for efficiently transferring bulk data between RDBMS and Hadoop. ✓
- ◯ Write a MapReduce job to read data from the RDBMS and store it in HDFS.

P

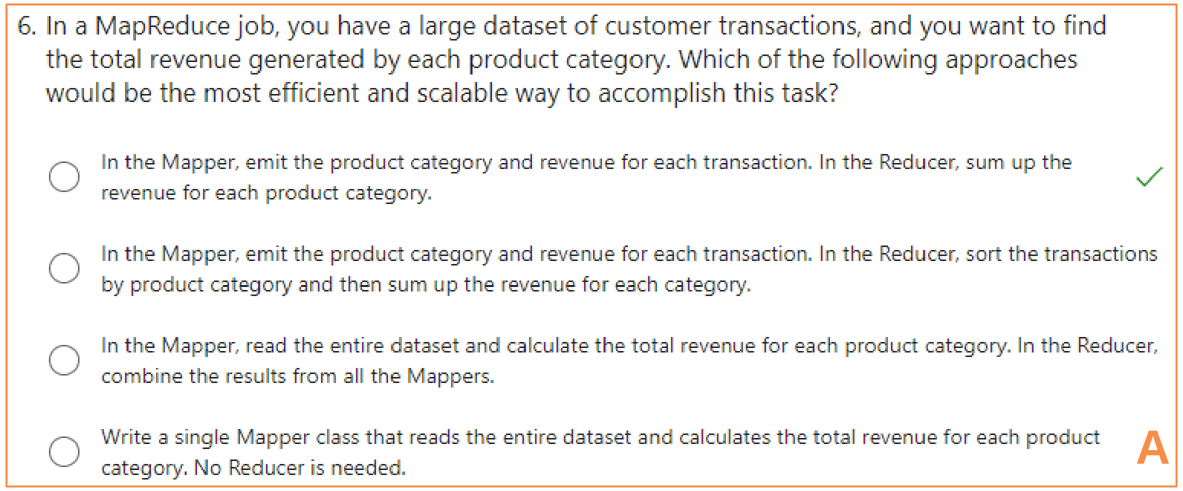
Knowledge Based - Intermediate Skill Depth (CAMP)



5. Which of the following statements best describes the concept of "data locality" in the Hadoop ecosystem?

- ○ Data locality refers to the practice of storing data close to the computation nodes to minimize network traffic. ✓
- ○ Data locality is a technique used to ensure that data is evenly distributed across all nodes in a Hadoop cluster.
- ○ Data locality is a security feature in Hadoop that restricts data access to specific nodes based on their physical location.
- ○ Data locality is a concept related to caching frequently accessed data in memory to improve performance.

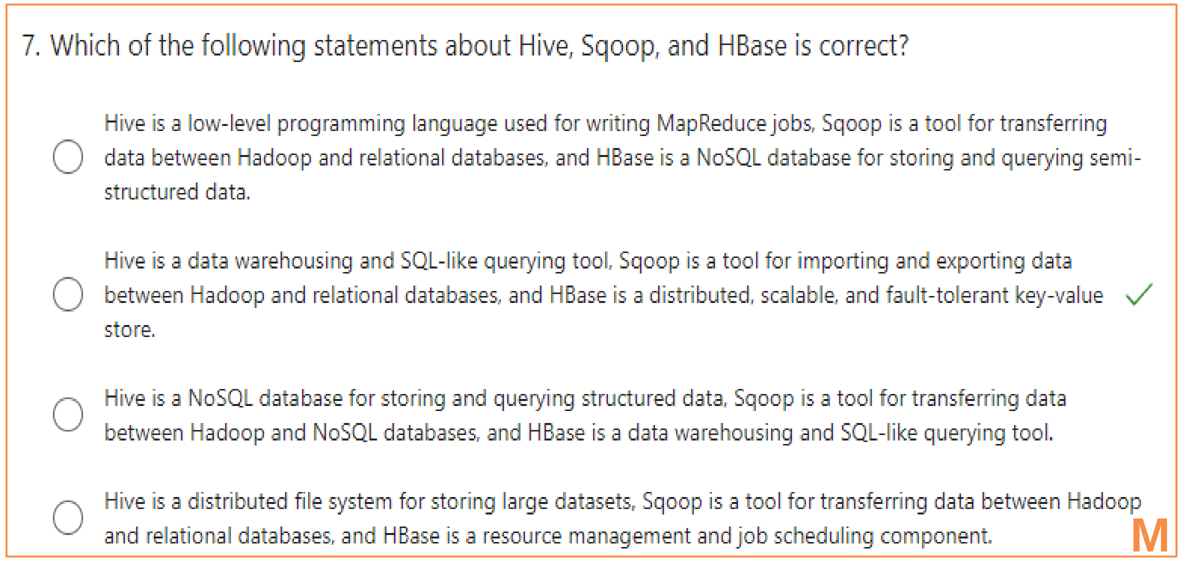
C



6. In a MapReduce job, you have a large dataset of customer transactions, and you want to find the total revenue generated by each product category. Which of the following approaches would be the most efficient and scalable way to accomplish this task?

- ○ In the Mapper, emit the product category and revenue for each transaction. In the Reducer, sum up the revenue for each product category. ✓
- ○ In the Mapper, emit the product category and revenue for each transaction. In the Reducer, sort the transactions by product category and then sum up the revenue for each category.
- ○ In the Mapper, read the entire dataset and calculate the total revenue for each product category. In the Reducer, combine the results from all the Mappers.
- ○ Write a single Mapper class that reads the entire dataset and calculates the total revenue for each product category. No Reducer is needed.

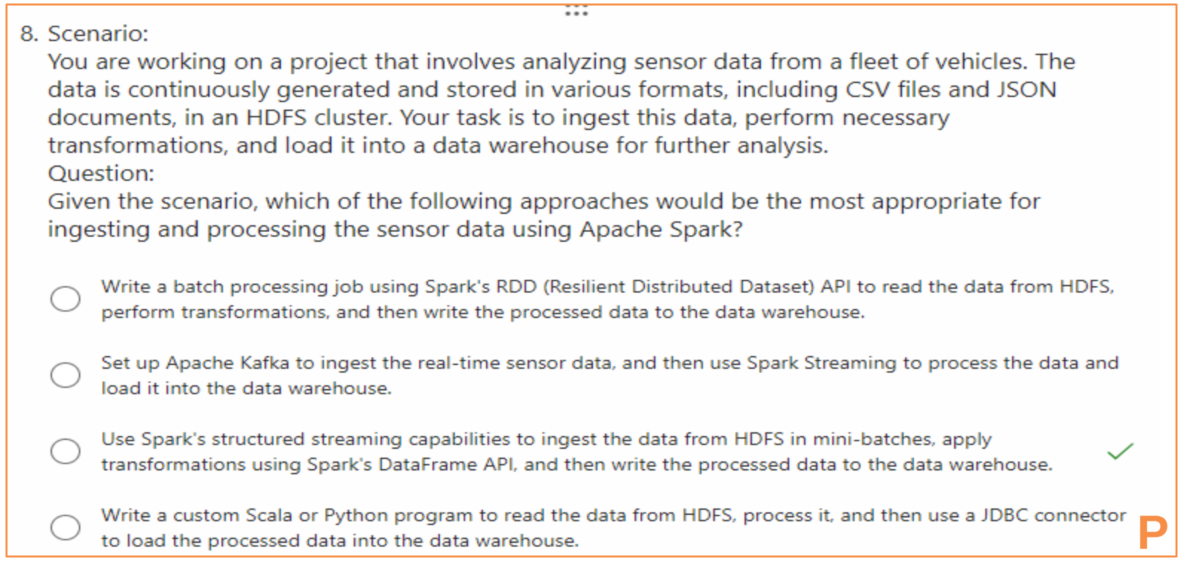
A



7. Which of the following statements about Hive, Sqoop, and HBase is correct?

- ○ Hive is a low-level programming language used for writing MapReduce jobs, Sqoop is a tool for transferring data between Hadoop and relational databases, and HBase is a NoSQL database for storing and querying semi-structured data.
- ○ Hive is a data warehousing and SQL-like querying tool, Sqoop is a tool for importing and exporting data between Hadoop and relational databases, and HBase is a distributed, scalable, and fault-tolerant key-value store. ✓
- ○ Hive is a NoSQL database for storing and querying structured data, Sqoop is a tool for transferring data between Hadoop and NoSQL databases, and HBase is a data warehousing and SQL-like querying tool.
- ○ Hive is a distributed file system for storing large datasets, Sqoop is a tool for transferring data between Hadoop and relational databases, and HBase is a resource management and job scheduling component.

M

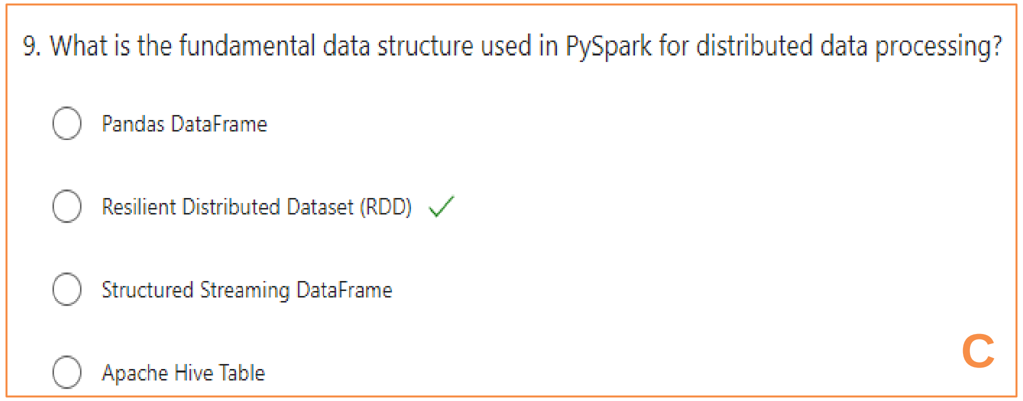


8. Scenario:
You are working on a project that involves analyzing sensor data from a fleet of vehicles. The data is continuously generated and stored in various formats, including CSV files and JSON documents, in an HDFS cluster. Your task is to ingest this data, perform necessary transformations, and load it into a data warehouse for further analysis.
Question:
Given the scenario, which of the following approaches would be the most appropriate for ingesting and processing the sensor data using Apache Spark?

- ○ Write a batch processing job using Spark's RDD (Resilient Distributed Dataset) API to read the data from HDFS, perform transformations, and then write the processed data to the data warehouse.
- ○ Set up Apache Kafka to ingest the real-time sensor data, and then use Spark Streaming to process the data and load it into the data warehouse.
- ○ Use Spark's structured streaming capabilities to ingest the data from HDFS in mini-batches, apply transformations using Spark's DataFrame API, and then write the processed data to the data warehouse. ✓
- ○ Write a custom Scala or Python program to read the data from HDFS, process it, and then use a JDBC connector to load the processed data into the data warehouse.

P

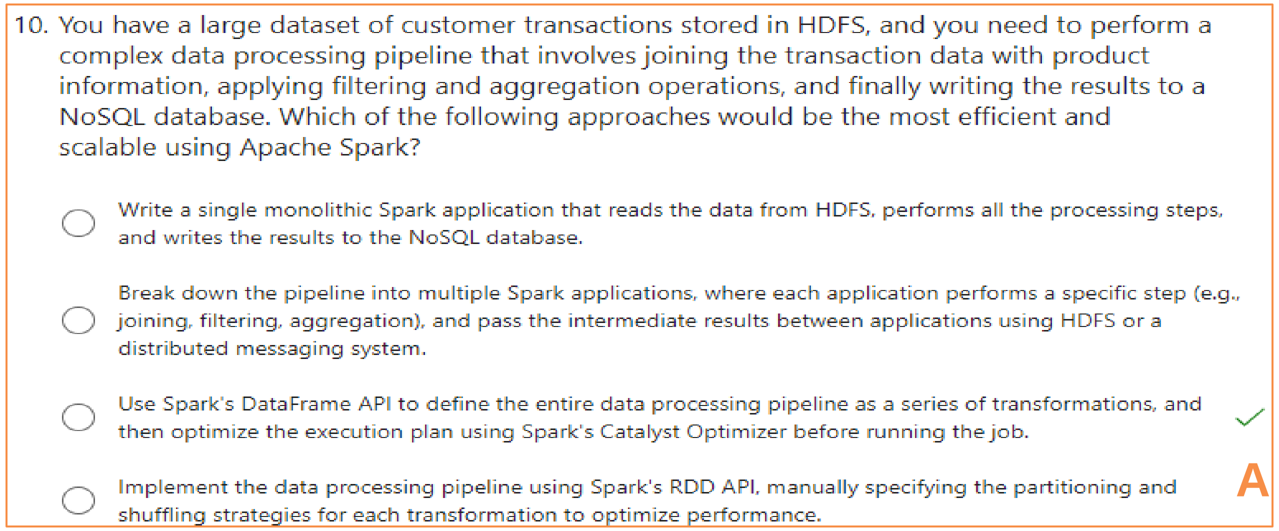
Knowledge Based - Advanced Skill Depth (CAMP)



9. What is the fundamental data structure used in PySpark for distributed data processing?

- Pandas DataFrame
- [✓ Resilient Distributed Dataset (RDD)]
- Structured Streaming DataFrame
- Apache Hive Table

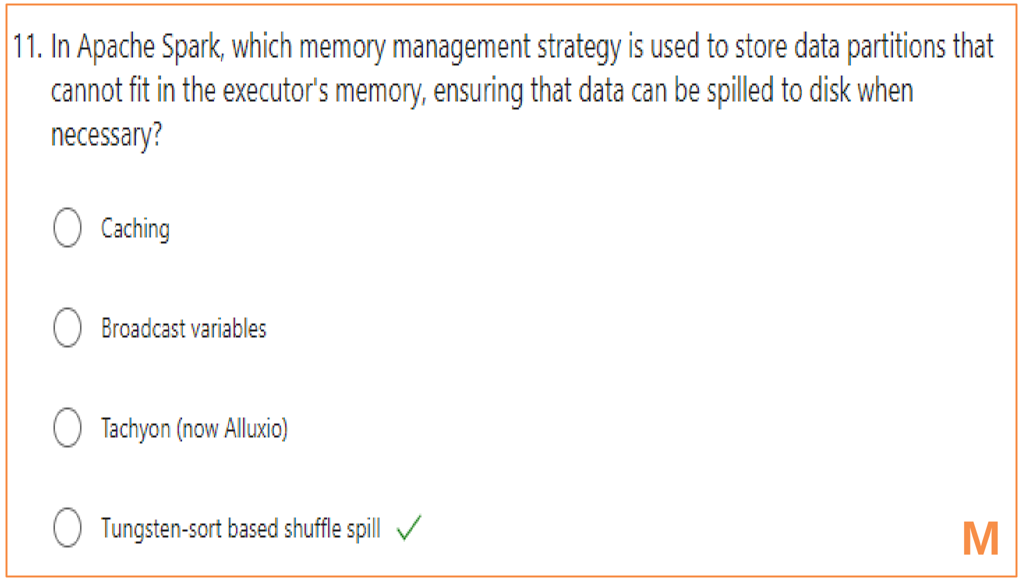
C



10. You have a large dataset of customer transactions stored in HDFS, and you need to perform a complex data processing pipeline that involves joining the transaction data with product information, applying filtering and aggregation operations, and finally writing the results to a NoSQL database. Which of the following approaches would be the most efficient and scalable using Apache Spark?

- Write a single monolithic Spark application that reads the data from HDFS, performs all the processing steps, and writes the results to the NoSQL database.
- Break down the pipeline into multiple Spark applications, where each application performs a specific step (e.g., joining, filtering, aggregation), and pass the intermediate results between applications using HDFS or a distributed messaging system.
- [✓ Use Spark's DataFrame API to define the entire data processing pipeline as a series of transformations, and then optimize the execution plan using Spark's Catalyst Optimizer before running the job.]
- Implement the data processing pipeline using Spark's RDD API, manually specifying the partitioning and shuffling strategies for each transformation to optimize performance.

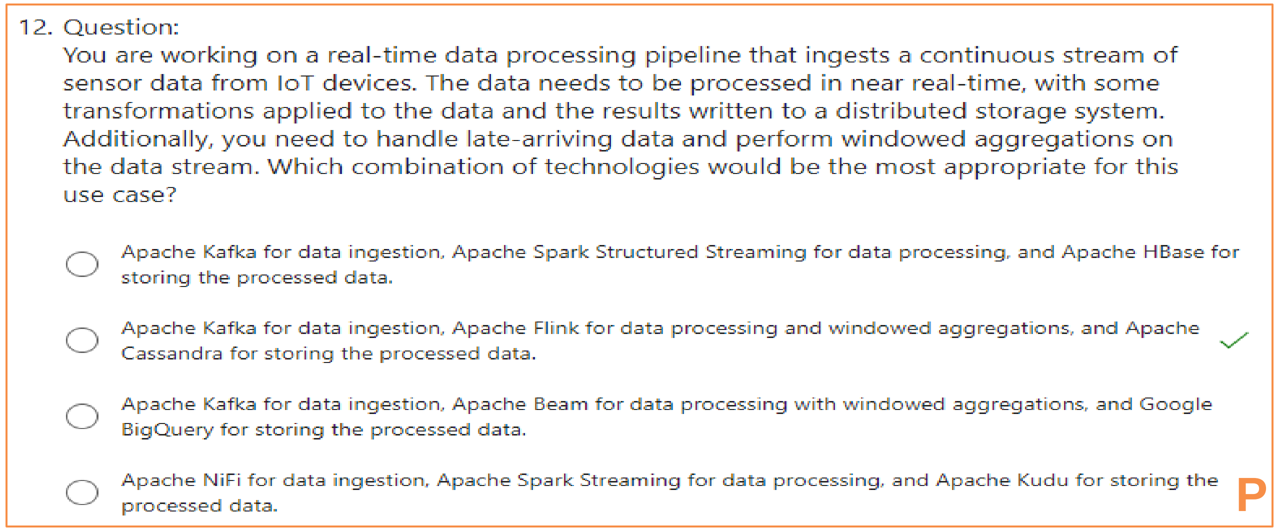
A



11. In Apache Spark, which memory management strategy is used to store data partitions that cannot fit in the executor's memory, ensuring that data can be spilled to disk when necessary?

- Caching
- Broadcast variables
- Tachyon (now Alluxio)
- [✓ Tungsten-sort based shuffle spill]

M



12. Question:
You are working on a real-time data processing pipeline that ingests a continuous stream of sensor data from IoT devices. The data needs to be processed in near real-time, with some transformations applied to the data and the results written to a distributed storage system. Additionally, you need to handle late-arriving data and perform windowed aggregations on the data stream. Which combination of technologies would be the most appropriate for this use case?

- Apache Kafka for data ingestion, Apache Spark Structured Streaming for data processing, and Apache HBase for storing the processed data.
- [✓ Apache Kafka for data ingestion, Apache Flink for data processing and windowed aggregations, and Apache Cassandra for storing the processed data.]
- Apache Kafka for data ingestion, Apache Beam for data processing with windowed aggregations, and Google BigQuery for storing the processed data.
- Apache NiFi for data ingestion, Apache Spark Streaming for data processing, and Apache Kudu for storing the processed data.

P

Skill Based - Beginner

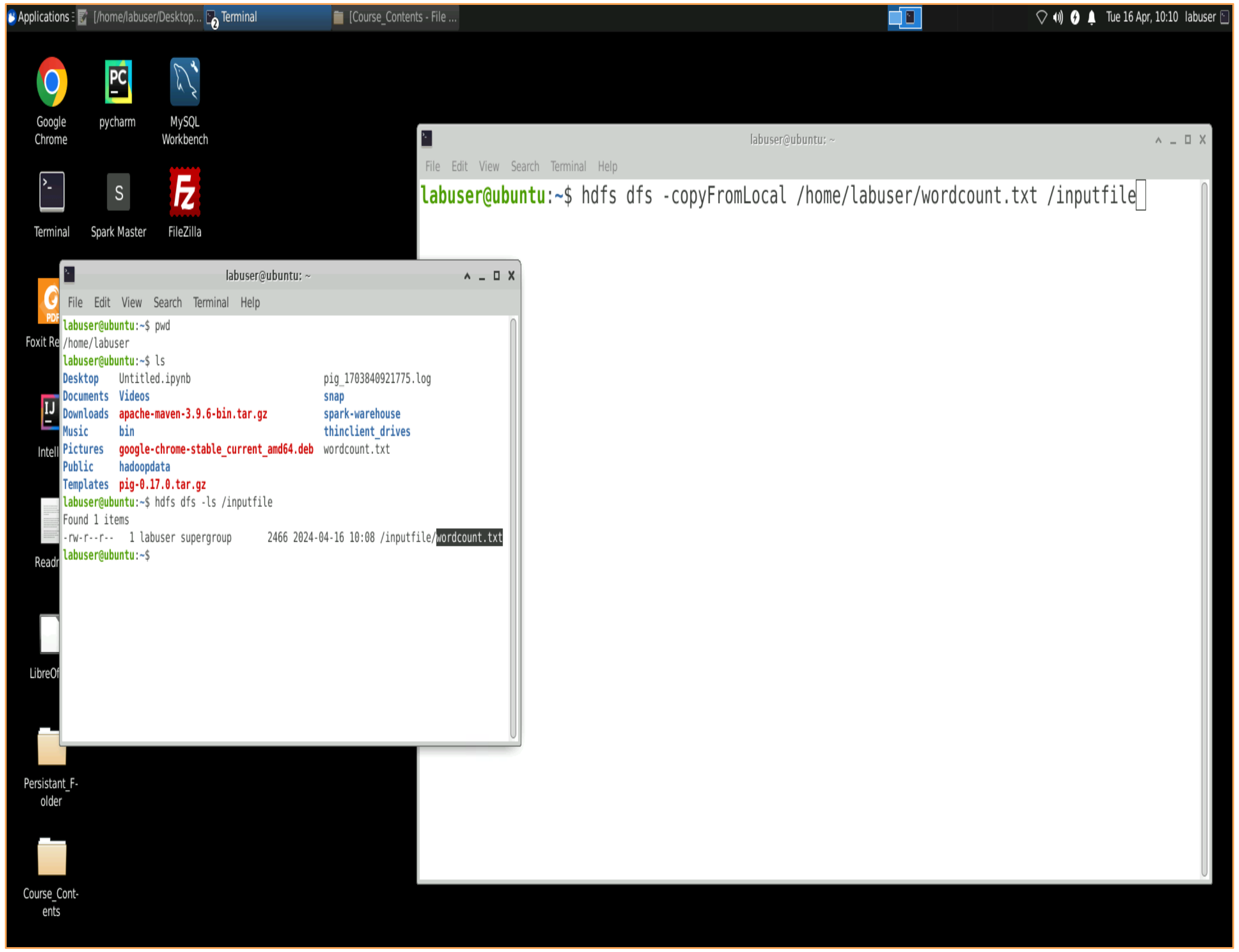
Problem Statement:

You are tasked with analyzing historical stock data for a company listed on the New York Stock Exchange (NYSE). The data includes daily stock prices and trading volume for a specific company (let's call it "JEF") for a certain period.

Problem: Analyze the historical stock data for the company "JEF" listed on the New York Stock Exchange (NYSE) and derive insights from it.

Requirements:

Load the provided historical stock data into the Hadoop Distributed File System (HDFS).



```
labuser@ubuntu:~$ hdfs dfs -copyFromLocal /home/labuser/wordcount.txt /inputfile
```

```
labuser@ubuntu:~$ pwd
/home/labuser
labuser@ubuntu:~$ ls
Desktop    Untitled.ipynb                          pig_1703840921775.log
Documents  Videos                                  snap
Downloads  apache-maven-3.9.6-bin.tar.gz           spark-warehouse
Music      bin                                     thinclient_drives
Pictures   google-chrome-stable_current_amd64.deb  wordcount.txt
Public     hadoopdata
Templates  pig-0.17.0.tar.gz
labuser@ubuntu:~$ hdfs dfs -ls /inputfile
Found 1 items
-rw-r--r--   1 labuser supergroup       2466 2024-04-16 10:08 /inputfile/wordcount.txt
labuser@ubuntu:~$
```

Skill Based - Intermediate

Problem Statement:

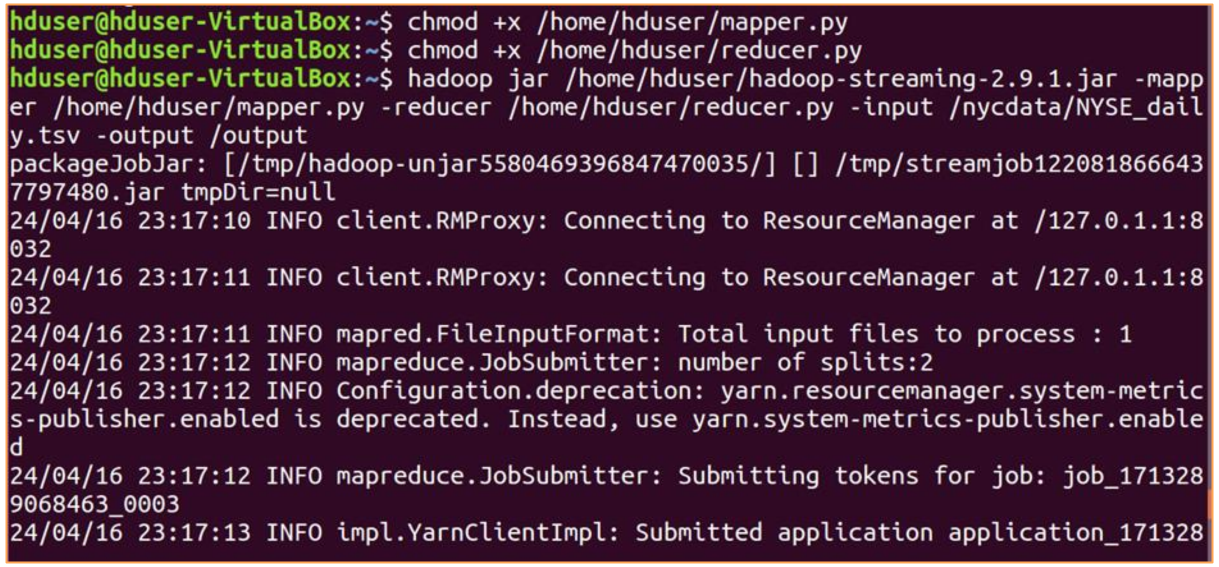
You are tasked with analyzing historical stock data for a company listed on the New York Stock Exchange (NYSE). The data includes daily stock prices and trading volume for a specific company (let's call it "JEF") for a certain period.

Problem: Analyze the historical stock data for the company "JEF" listed on the New York Stock Exchange (NYSE) and derive insights from it.

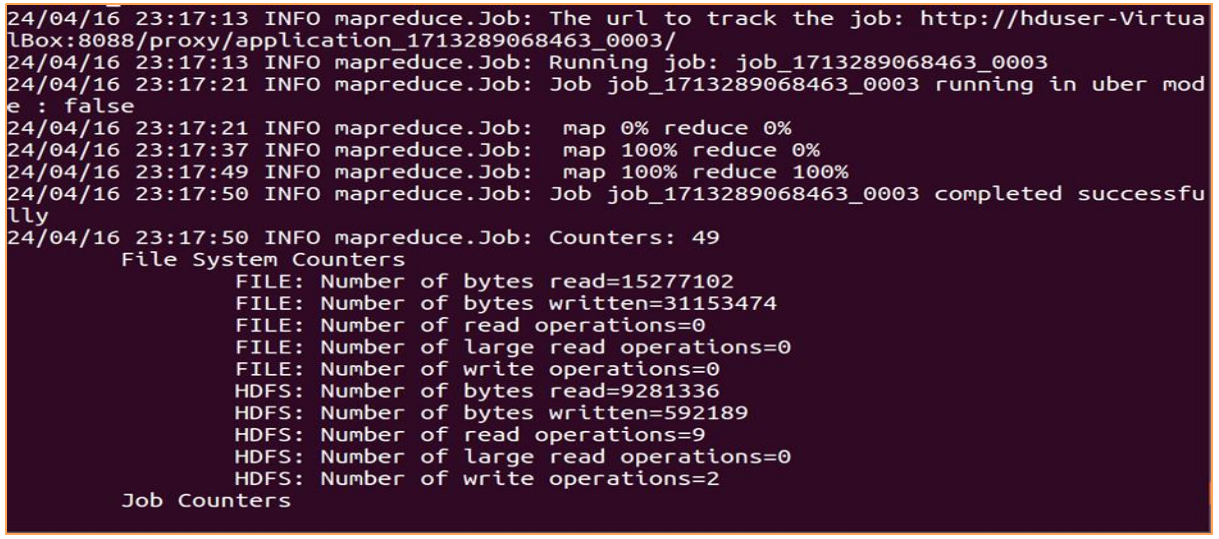
Requirements:

Load the provided historical stock data into the Hadoop Distributed File System (HDFS).

Implement MapReduce jobs to:
Calculate the average closing price for each month.
Determine the highest and lowest closing prices for each year.
Find the average trading volume for each quarter.



```
hduser@hduser-VirtualBox:~$ chmod +x /home/hduser/mapper.py
hduser@hduser-VirtualBox:~$ chmod +x /home/hduser/reducer.py
hduser@hduser-VirtualBox:~$ hadoop jar /home/hduser/hadoop-streaming-2.9.1.jar -mapp
er /home/hduser/mapper.py -reducer /home/hduser/reducer.py -input /nycdata/NYSE_dail
y.tsv -output /output
packageJobJar: [/tmp/hadoop-unjar5580469396847470035/] [] /tmp/streamjob122081866643
7797480.jar tmpDir=null
24/04/16 23:17:10 INFO client.RMProxy: Connecting to ResourceManager at /127.0.1.1:8
032
24/04/16 23:17:11 INFO client.RMProxy: Connecting to ResourceManager at /127.0.1.1:8
032
24/04/16 23:17:11 INFO mapred.FileInputFormat: Total input files to process : 1
24/04/16 23:17:12 INFO mapreduce.JobSubmitter: number of splits:2
24/04/16 23:17:12 INFO Configuration.deprecation: yarn.resourcemanager.system-metric
s-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enable
d
24/04/16 23:17:12 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_171328
9068463_0003
24/04/16 23:17:13 INFO impl.YarnClientImpl: Submitted application application_171328
```



```
24/04/16 23:17:13 INFO mapreduce.Job: The url to track the job: http://hduser-Virtua
lBox:8088/proxy/application_1713289068463_0003/
24/04/16 23:17:13 INFO mapreduce.Job: Running job: job_1713289068463_0003
24/04/16 23:17:21 INFO mapreduce.Job: Job job_1713289068463_0003 running in uber mod
e : false
24/04/16 23:17:21 INFO mapreduce.Job:  map 0% reduce 0%
24/04/16 23:17:37 INFO mapreduce.Job:  map 100% reduce 0%
24/04/16 23:17:49 INFO mapreduce.Job:  map 100% reduce 100%
24/04/16 23:17:50 INFO mapreduce.Job: Job job_1713289068463_0003 completed successfu
lly
24/04/16 23:17:50 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=15277102
                FILE: Number of bytes written=31153474
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=9281336
                HDFS: Number of bytes written=592189
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
```

Skill Based - Advanced

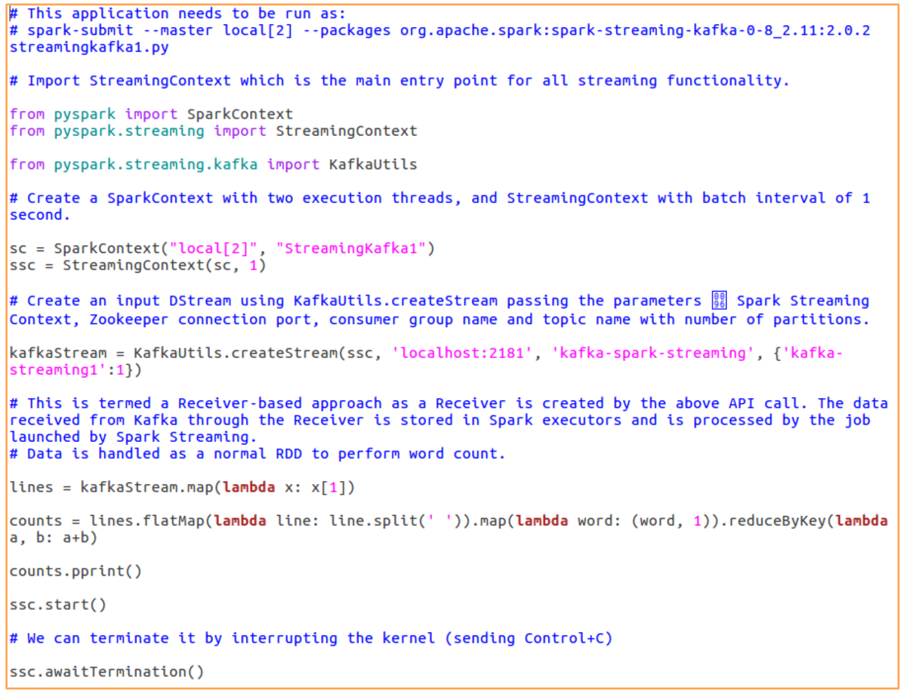
Problem Statement:

You are tasked with implementing a real-time streaming data pipeline to analyze and derive insights from the real-time stock market data of a company listed on the New York Stock Exchange (NYSE). The data includes continuous updates of stock prices and trading volume for the company "JEF".

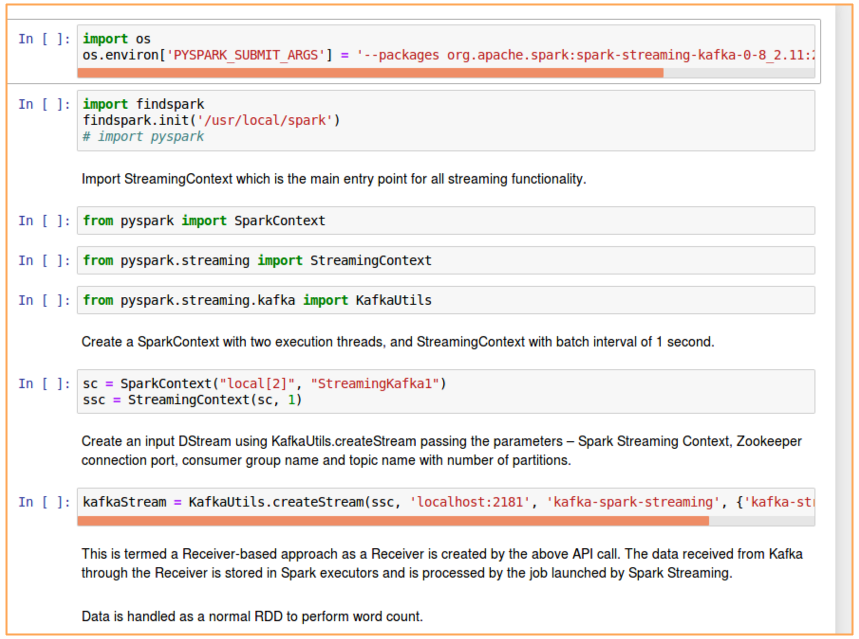
Problem: Develop a real-time streaming data pipeline using Apache Kafka and Apache Spark Streaming to monitor and analyze the real-time stock market data for the company "JEF" listed on the New York Stock Exchange (NYSE) and derive insights in real-time.

Requirements:

Set up Apache Kafka as a distributed streaming platform to ingest and buffer real-time stock market data.
Implement producers to continuously stream real-time stock market data for the company "JEF" to Kafka topics.
Develop Apache Spark Streaming applications to consume and process the real-time data from Kafka topics.
Perform real-time analytics on the streaming data to:
Calculate moving averages of stock prices over different time windows.
Identify sudden spikes or drops in stock prices and trading volume.
Detect anomalies or unusual patterns in the stock market data.
Generate alerts or notifications for significant events detected in the real-time data stream.



```
# This application needs to be run as:
# spark-submit --master local[2] --packages org.apache.spark:spark-streaming-kafka-0-8_2.11:2.0.2
streamingkafka1.py

# Import StreamingContext which is the main entry point for all streaming functionality.

from pyspark import SparkContext
from pyspark.streaming import StreamingContext

from pyspark.streaming.kafka import KafkaUtils

# Create a SparkContext with two execution threads, and StreamingContext with batch interval of 1
second.

sc = SparkContext("local[2]", "StreamingKafka1")
ssc = StreamingContext(sc, 1)

# Create an input DStream using KafkaUtils.createStream passing the parameters  Spark Streaming
Context, Zookeeper connection port, consumer group name and topic name with number of partitions.

kafkaStream = KafkaUtils.createStream(ssc, 'localhost:2181', 'kafka-spark-streaming', {'kafka-
streaming1':1})

# This is termed a Receiver-based approach as a Receiver is created by the above API call. The data
received from Kafka through the Receiver is stored in Spark executors and is processed by the job
launched by Spark Streaming.
# Data is handled as a normal RDD to perform word count.

lines = kafkaStream.map(lambda x: x[1])

counts = lines.flatMap(lambda line: line.split(' ')).map(lambda word: (word, 1)).reduceByKey(lambda
a, b: a+b)

counts.pprint()

ssc.start()

# We can terminate it by interrupting the kernel (sending Control+C)

ssc.awaitTermination()
```



```
In [ ]: import os
        os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages org.apache.spark:spark-streaming-kafka-0-8_2.11:

In [ ]: import findspark
        findspark.init('/usr/local/spark')
        # import pyspark
```

Import StreamingContext which is the main entry point for all streaming functionality.

```
In [ ]: from pyspark import SparkContext

In [ ]: from pyspark.streaming import StreamingContext

In [ ]: from pyspark.streaming.kafka import KafkaUtils
```

Create a SparkContext with two execution threads, and StreamingContext with batch interval of 1 second.

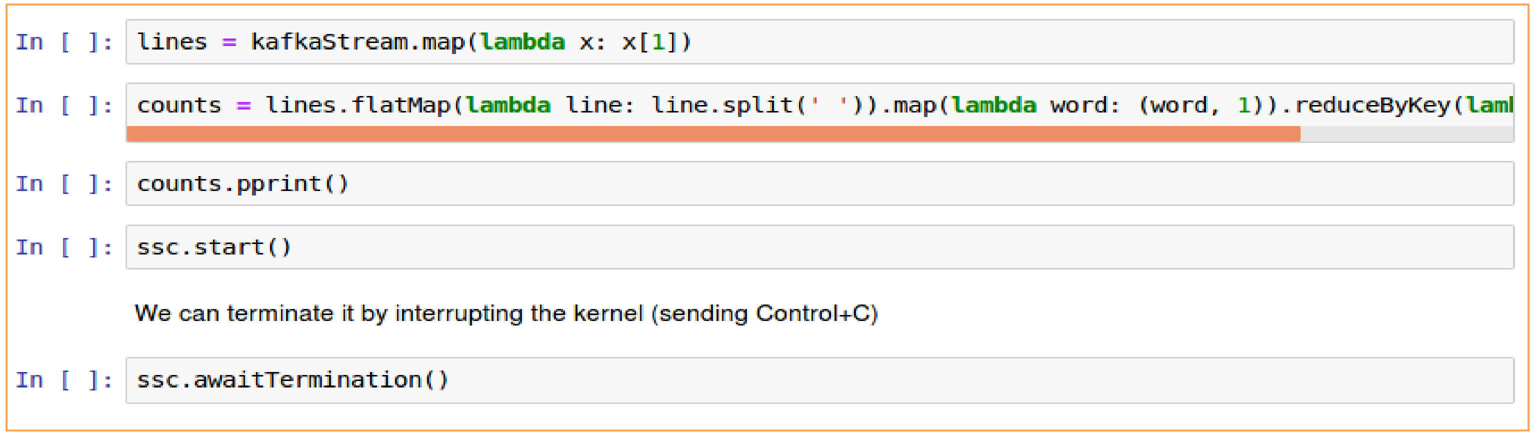
```
In [ ]: sc = SparkContext("local[2]", "StreamingKafka1")
        ssc = StreamingContext(sc, 1)
```

Create an input DStream using KafkaUtils.createStream passing the parameters – Spark Streaming Context, Zookeeper connection port, consumer group name and topic name with number of partitions.

```
In [ ]: kafkaStream = KafkaUtils.createStream(ssc, 'localhost:2181', 'kafka-spark-streaming', {'kafka-st
```

This is termed a Receiver-based approach as a Receiver is created by the above API call. The data received from Kafka through the Receiver is stored in Spark executors and is processed by the job launched by Spark Streaming.

Data is handled as a normal RDD to perform word count.

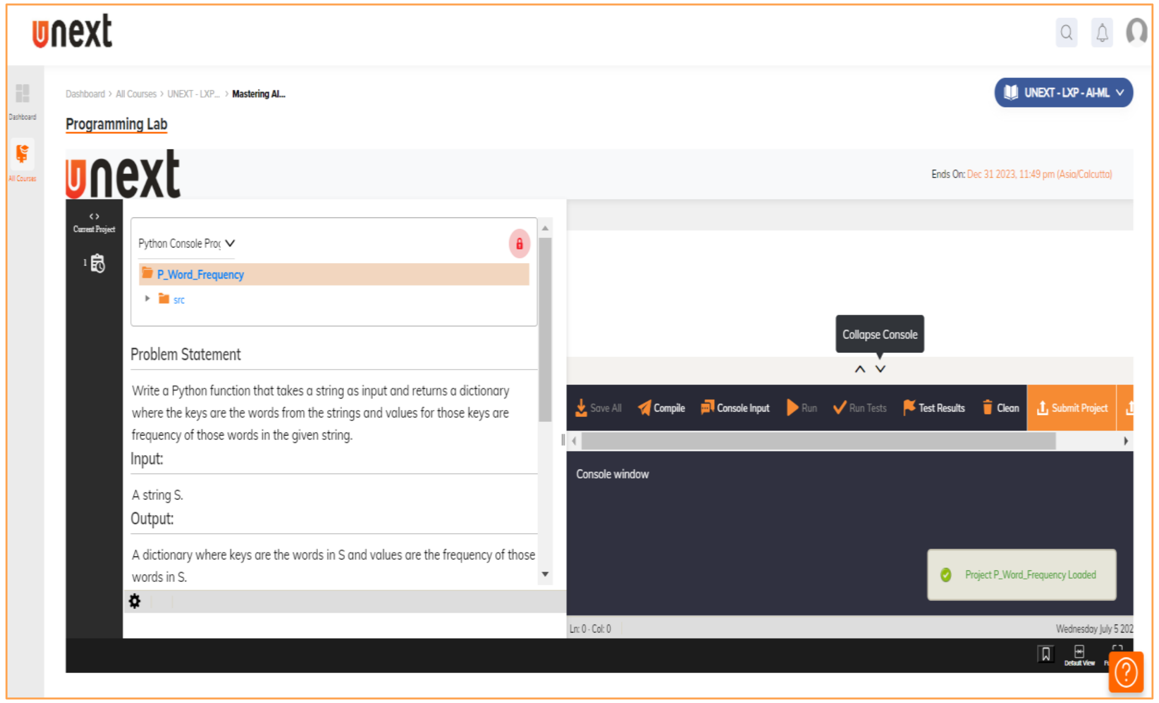


```
In [ ]: lines = kafkaStream.map(lambda x: x[1])

In [ ]: counts = lines.flatMap(lambda line: line.split(' ')).map(lambda word: (word, 1)).reduceByKey(laml

In [ ]: counts.pprint()

In [ ]: ssc.start()
```

We can terminate it by interrupting the kernel (sending Control+C)

```
In [ ]: ssc.awaitTermination()
```

Programming Environment Snapshot



Programming Lab

Ends On: Dec 31 2023, 11:49 pm (Asia/Calcutta)

Problem Statement

Write a Python function that takes a string as input and returns a dictionary where the keys are the words from the strings and values for those keys are frequency of those words in the given string.

Input:

A string S.

Output:

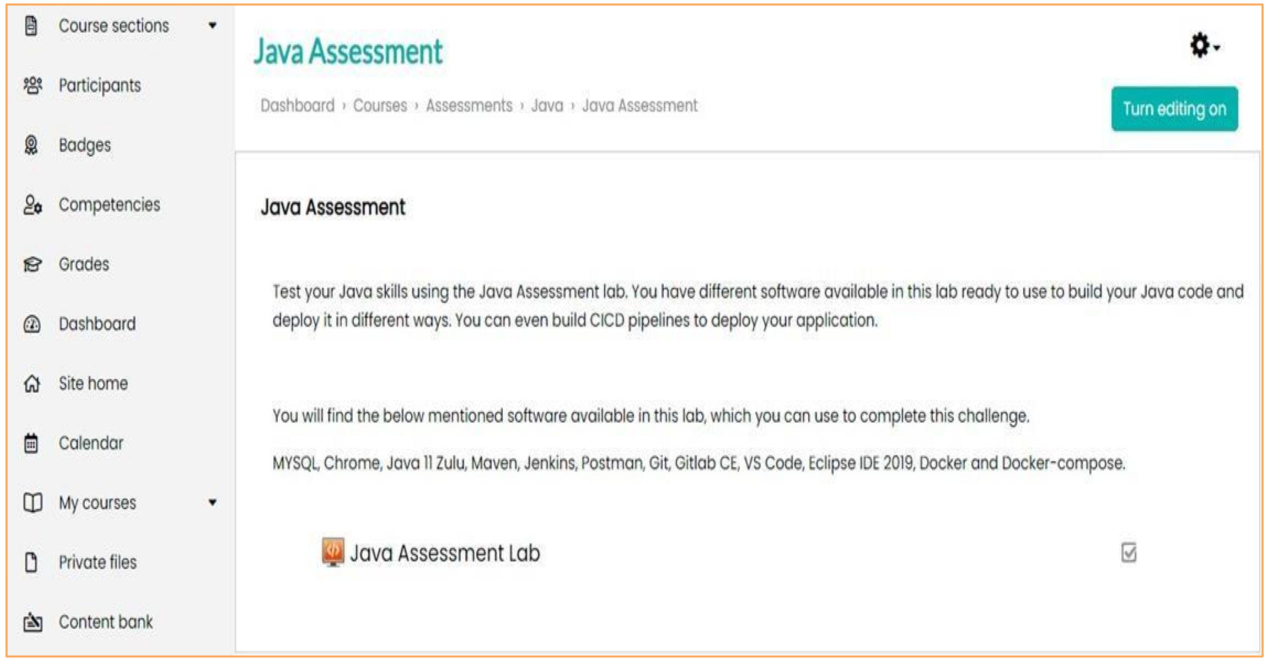
A dictionary where keys are the words in S and values are the frequency of those words in S.

Project P_Word_Frequency Loaded

Ln: 0 - Col: 0 Wednesday July 5 202

Python

Java Full Stack



Java Assessment

Java Assessment

Test your Java skills using the Java Assessment lab. You have different software available in this lab ready to use to build your Java code and deploy it in different ways. You can even build CICD pipelines to deploy your application.

You will find the below mentioned software available in this lab, which you can use to complete this challenge.

MYSQL, Chrome, Java 11 Zulu, Maven, Jenkins, Postman, Git, Gitlab CE, VS Code, Eclipse IDE 2019, Docker and Docker-compose.

Java Assessment Lab

Sample Remote Proctoring Report

CONCLUSION	FACE MATCH	CREDIBILITY
negative	96%	0%

Student profile

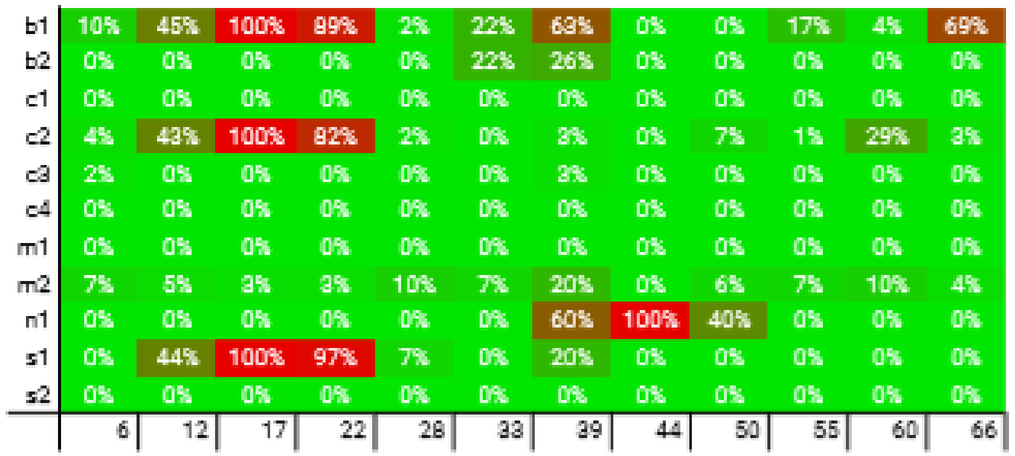
Login	AA04UTF9
Name	AA04UTF9
Gender	—
Age	—
Additionally	—

Proctoring session

ID	MT1014903
Subject	A024647_E227501
Start date and time	8/28/2019, 9:43:14 AM (UTC)
Duration	65 min
Estimation of credibility	0%
Face match	96%
Conclusion	Negative
Proctor	—
Comment	—

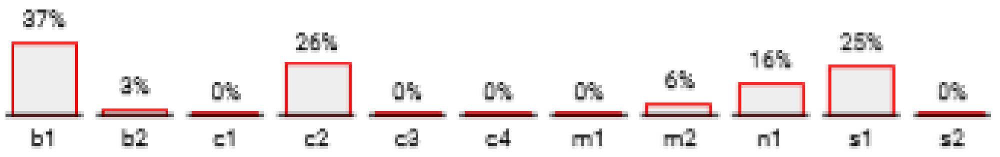
Distribution of events

By time, minutes



	6	12	17	22	28	33	39	44	50	55	60	66
b1	10%	45%	100%	89%	2%	22%	63%	0%	0%	17%	4%	69%
b2	0%	0%	0%	0%	0%	22%	26%	0%	0%	0%	0%	0%
c1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
c2	4%	43%	100%	82%	2%	0%	3%	0%	7%	1%	29%	3%
c3	2%	0%	0%	0%	0%	0%	3%	0%	0%	0%	0%	0%
c4	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
m1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
m2	7%	5%	3%	3%	10%	7%	20%	0%	6%	7%	10%	4%
n1	0%	0%	0%	0%	0%	0%	60%	100%	40%	0%	0%	0%
s1	0%	44%	100%	97%	7%	0%	20%	0%	0%	0%	0%	0%
s2	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

By events

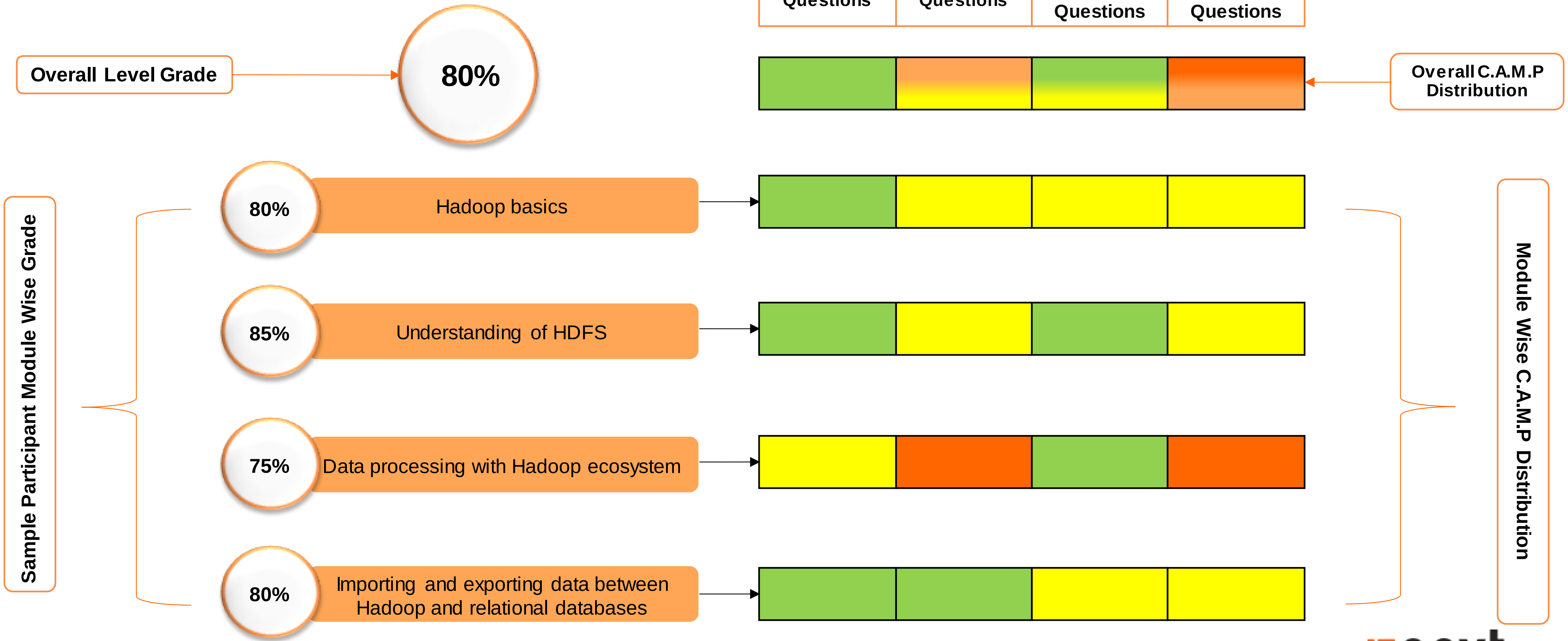


Legend

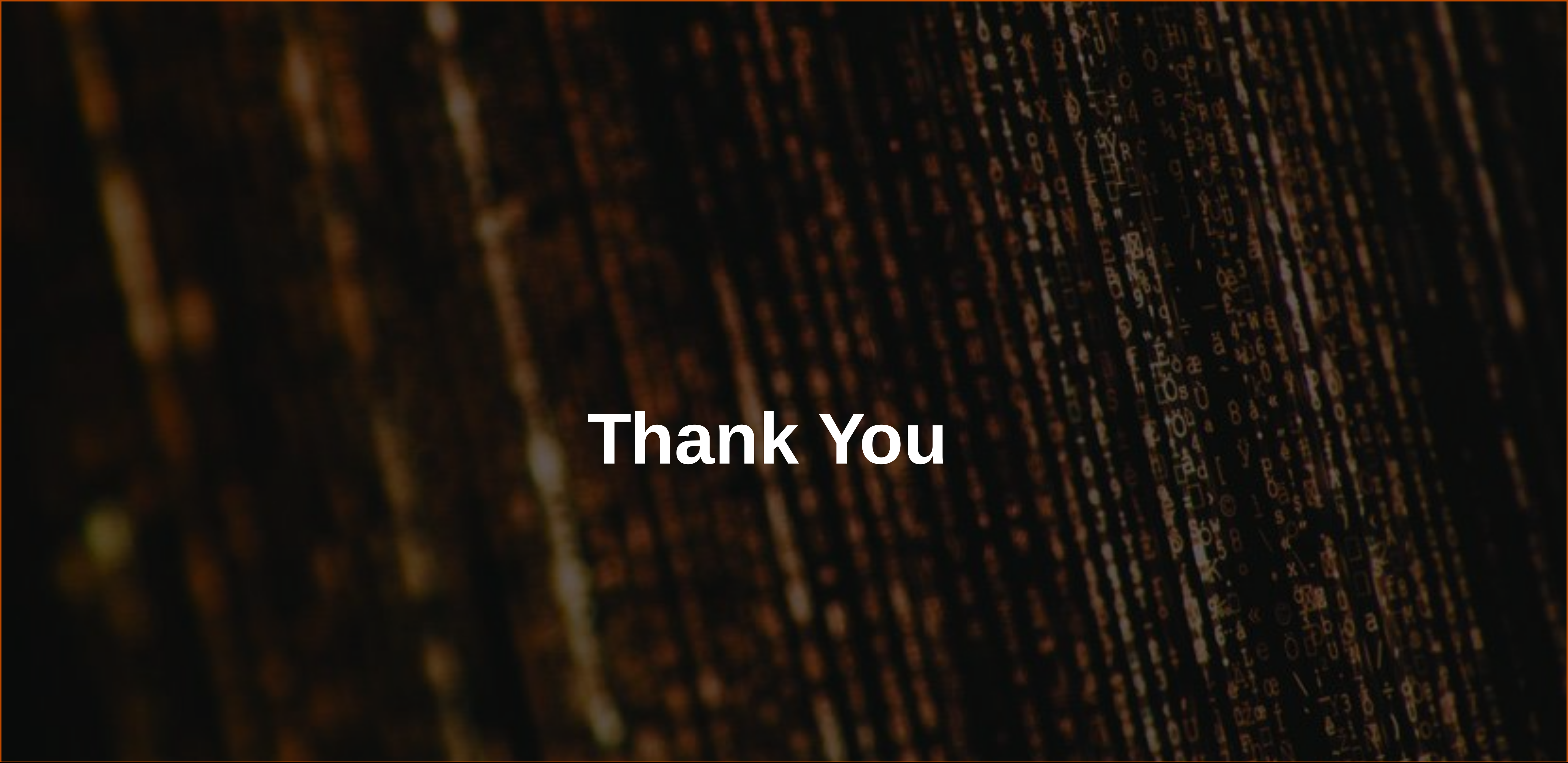
b1 – focus changed to a different window
b2 – full-screen mode is disabled
c1 – webcam is disabled
c2 – no face in front of camera
c3 – several faces in front of camera
c4 – unidentified face in front of camera
m1 – microphone is muted or low volume
m2 – conversation or noise in the background
n1 – no network connection
s1 – no video from screen
s2 – additional display is connected

Sample Reporting and Insights (Big Data) – For Beginner Level

Based on the assessments the following reports will be shared


Overall Level Grade
80%
Overall C.A.M.P Distribution
Conceptual Questions
Analytical Questions
Memory Recall Questions
Practise Based Questions
Sample Participant Module Wise Grade
80%
Hadoop basics
85%
Understanding of HDFS
75%
Data processing with Hadoop ecosystem
80%
Importing and exporting data between Hadoop and relational databases
Module Wise C.A.M.P Distribution

	Grade	Conceptual Questions	Analytical Questions	Memory Recall Questions	Practise Based Questions
Overall Level Grade	80%	🟩	🟧🟨	🟩🟨	🟧
Hadoop basics	80%	🟩	🟨	🟨	🟨
Understanding of HDFS	85%	🟩	🟨	🟩	🟨
Data processing with Hadoop ecosystem	75%	🟨	🟧	🟩	🟧
Importing and exporting data between Hadoop and relational databases	80%	🟩	🟩	🟨	🟨



Thank You