1.What is Spark?

- ➔ Spark is a general purpose in-memory compute engine.
- ➔ Let us try to deep dive into this line:
    - GENERAL PURPOSE:
- ➔ In Hadoop, cleaning, querying and data ingestion we have different tools.
- ➔ Its not the same kind of code we are writing, everything is different and it takes time to learn.
- ➔ But Spark is a General-Purpose Compute Engine, it will help us what way as possible to minimize the different tools usage.
- ➔ We need to learn one style of writing the code, in that we can manage to do all the things like cleaning, querying and data ingestion.

Operation	**Hadoop**	**Spark**
Cleaning	PIG	Spark SQL
Querying	HIVE	Spark SQL for SQL-based queries or the Data Frame API for more programmatic queries
Data Ingestion	Sqoop	We can use several tools such as Apache Kafka, Apache Flume, or Spark Streaming for real-time data ingestion. Additionally, Spark provides built-in support for reading and writing data from various file formats such as CSV, JSON, and Parquet through Spark SQL or the Data Frame API.
Problem-Solving	Only bound to use Map-Reduce. Which means Everything fits in Map-Reduce or not we are trying to fit it unnecessarily	In Spark, we have plenty of operations to solve the problem efficiently, it's not like we restricted to solve using Map-Reduce operations only.

2.What is RDD?

- ➔ RDD defined as Resilient Distributed Datasets.
- ➔ RDD is the basic unit which holds the data in spark.
- ➔ RDD as a large list that is distributed across multiple machines in the cluster.
- ➔ RDD is distributed but in-memory (RAM) rather than in Disk.

For more Follow:
https://www.linkedin.com/in/nandeshreddy/

3.How to Recover the lost RDD?

- ➔ RDD is resilient to failure, if we lost any RDD we can recover it back using Lineage Graph/DAG-Directed Acyclic Graph.
- ➔ It captures the lineage or the history of the RDD and the sequence of transformations applied on it.
- ➔ That's the reason why RDD's are Immutable. It allows us to get the lost RDD.
- ➔ When a partition of an RDD is lost due to node failure, Spark can use the lineage graph to identify the transformations that led to that partition and recompute it.For Ex:

RDD 1
| map ()

RDD 2

| groupByKey ()

RDD 3 (lost it due to some reasons)

- ➔ Now it will check for its parent RDD using DAG.
- ➔ It will quickly apply the RDD2 used transformation to get RDD3 output (Here its groupByKey ())
- ➔ Using this method, it will easily recover the Lost RDD.

	HDFS	**RDD**
Fault Tolerance	We get Resiliency by using Replication Factor	We get Resiliency by using Lineage Graph

4.What the types 2 types of Operations in Spark?

- ➔ Transformations
- ➔ Actions

- ➔ Transformations:

  - Transformations are always Lazy in Spark, meaning they do not execute immediately, but only when an action is called.
  - Until or unless we call action nothing will execute.
  - Spark Engine creates Execution Plan/ DAG Diagram in backend to remember in which sequence it need to execute the code, when an action is called.
  - So, whenever we apply transformations, an entry to the execution plan will get added.
  - There are 2 types of transformations,
    - Narrow transformations

For more Follow:
https://www.linkedin.com/in/nandeshreddy/

- Wide transformations

Narrow Transformations	Wide Transformations
These transformations operate on a single partition at a time and do not require shuffling of data across the network.	These transformations require shuffling of data across the network, which means data from multiple partitions is combined to form new partitions.
The input and output partitions of these transformations are one-to-one	The input and output partitions of these transformations are not one-to-one.
Examples of narrow transformations include map (), filter (), union (), etc.	Examples of wide transformations include groupByKey (), reduceByKey (), join (), etc.

➔ In general, narrow transformations are preferred over wide transformations as they are faster due to less data shuffling across the network.

Actions:

➔ It will execute the Transformation Execution Plan in Sequence.
➔ Unlike transformations, actions are not lazy and execute immediately.

Transformations	Actions
Whenever we call transformations on RDD, we ger resultant RDD.	Whenever we call Actions on RDD, we get Local Variables.
Ex: map (), filter (), union (), groupByKey (), reduceByKey (), join ()	Ex: collect (), count (), reduce (), take (), foreach (), saveAsTextFile ()

5.Why Transformations are Lazy?

➔ Assume Transformations are not Lazy for a moment.
  - Consider we have 2 GB file in HDFS
    - Rdd1 = load (file1 from HDFS)
    - Rdd1.print(line 1)
  - To print just one line, we ended up loading 2 GB file in memory.
  - If Transformations are not lazy, we will face these kind of challenges

➔ Now, Consider the TRUTH, Transformations are LAZY.
    - Rdd1 = load (file1 from HDFS)
      - Nothing will happen, entry in the execution plan is created
    - Rdd1.print(line 1)
      - Entire Execution plan is built, it knows the complete context of what user is trying to do.
      - It knows that user is just trying to get first line, then why I need to load whole file into memory.

For more Follow:
https://www.linkedin.com/in/nandeshreddy/

- It will optimize internally (predicate push down etc) and load only first line into the memory.

➔ This optimization is possible only because of spark transformations are Lazy and entire execution plan will build in advance.

6.What are the differences between ReduceByKey () and CountByValue ()?

reduceByKey ()	**countByValue ()**
Need to use map () and ReduceByKey () to get same result as CountByValue ()	Instead of using map () and doing ReduceByKey () later we can directly use countByValue ()
Map () and reduceByKey () both are transformations, so the result we get is RDD	countByValue () is an action, so the result we get is in local variable
If we want to perform any operations on the reduceByKey () output, we can do because it stores the result in RDD. So we will get parallelism	Use it only when it's a final operation. Because result will be in local, we won't get parallelism if we perform any operations on the result.

Example:
file_path_1 ="/FileStore/tables/demo_file_3.txt"
rdd1 = sc. textFile(file_path_1)
rdd2 = rdd1.map (lambda x: (x,1))
rdd3 = rdd2.reduceByKey(lambda x, y: x+y)
rdd4 = rdd3.collect()
print(rdd4)
	Example:
	file_path_1 ="/FileStore/tables/demo_file_3.txt"
	rdd1 = sc. textFile(file_path_1)
	local_var = rdd1.countByValue()
	#countByValue () = map () + reduceByKey ()
	print(local_var)

7.What are the differences between Hadoop-1 and Hadoop-2?

➔ In Hadoop-1, we have only one Job-Tracker and Many Task Trackers.

➔ But Task Trackers was doing very less work. So, some of the responsibilities of Job Tracker could be delegated to Task Tracker.

➔ This was the main agenda while developing the YARN.

Hadoop-1	**Hadoop-2(YARN)**
➢ Job-Tracker was doing ➔ Scheduling ➔ Monitoring	➢ Resource Manager will be doing ➔ Scheduling ➢ Application Master will be doing ➔ Monitoring
➢ Task Tracker was monitoring the local Map-Reduce Tasks	➢ Node Manager monitors the local tasks

For more Follow:
https://www.linkedin.com/in/nandeshreddy/

8.What are the advantages of Hadoop-2(YARN) over Hadoop-1?

Drawbacks of Hadoop-1	Handling those Drawbacks in Hadoop-2
➢ Scalability Issue, if cluster size goes beyond 4k, then ➔ Data Node ➔ Job Tracker Used to be bottleneck.	➢ It handled using, ➢ Resource Manager -> Scheduling ➢ Application Master -> Monitoring
Fixed no of Map Reduce slots	With the concept of containers, the resource allocation is much more dynamic and we can request for any amount of CPU & Memory.
Only Map Reduce jobs was possible	We can have other jobs also apart from Map Reduce.Ex:Spark,Giraph etc.

9.Explain Spark Architecture?

➔ Spark follows the Master-Slave Architecture just like Hadoop
➔ Spark Driver acts as a Master
➔ A Bunch of Executors acts as Slaves

Driver:

➔ Responsible for Analysing the work
➔ Divides the work in many tasks
➔ Distribute the Tasks
➔ Schedule the Task and Monitors
➔ Each Application will have its own drivers, which means if there are 10 applications then there will be 10 different drivers.
➔ Everything depends on the driver, if driver crashes things are gone. Because if the master is gone then there is no recovery.

Executor:

➔ It's a JVM process holding CPU + Memory which can execute our code
➔ Each driver will have its own set of Executors.

For more Follow:
https://www.linkedin.com/in/nandeshreddy/

➔ No of Executors will wary for each Driver.
➔ No matter what executors always placed on the cluster

Driver:

There are 2 types of driver modes,

Client mode driver	**Cluster mode driver**
➢ Its used just for ad-hoc analysis, trial & error and just for exploratory purpose	➢ Its production ready purpose
➢ Client mode is not preferrable because if the client machine goes down or shutdown then the driver will stop.	➢ Client should have the flexibility to submit Job and go away, so cluster mode is better approach.

10.Who controls the cluster and how spark gets driver and cluster?

➔ Cluster Manager is the one who manages the Cluster.
➔ Most famous cluster managers are:
  - YARN
  - Mesos
  - Kubernetes
  - Spark standalone

11.What are the differences between reduceByKey () and reduce ()?

reduceByKey()	**reduce()**
➔ Wide transformation	➔ It's an action
➔ **Why Spark developers made it as Transformation?**	➔ **Why spark developers made it as an action?**
➔ In reduceByKey () - output we will get lot of (Key, Value) pairs. So, we can still have huge amount of data and we might be willing to do further operations in parallel, that's the reason why spark developers made it as a transformation.	➔ It gives single line output which is very small, that's why spark developers made it as an action

12.What are the differences between reduceByKey () and groupByKey ()?

reduceByKey ()	**groupByKey ()**
Wide transformation	Wide transformation
We get advantage of local aggregation. So, more work in Parallel Less shuffling is required	We don't' get any local aggregation, all (Key, Value) pairs sent to other machines So, We get Less parallelism Shuffle more Data,
We can think it as same as combiner acting at mapper	NA
Always prefer reduceByKey ()	Never prefer groupByKey ()

13.Difference between repartition () and coalesce ()?

repartition ()	**coalesce ()**
Repartition () has an intention to have final partitions of exactly equal size and for this it has to go through complete shuffling.	Coalesce () has an intention to minimize the shuffling and combines the existing partitions on each machine to avoid full shuffling. Note: The exact partitioning after applying coalesce () cannot be predicted and depends on factors such as, ➔ the number of partitions, ➔ the size of each partition, ➔ the available resources, and ➔ the distribution of data across partitions.

14.Difference between cache () and persist ()?

cache ()	**persist ()**

For more Follow:
https://www.linkedin.com/in/nandeshreddy/

Cache ()- will store data in memory (RAM)	Persists can store data in memory and disk also, it comes with various storage levels. ➔ MEMORY_ONLY ➔ MEMORY_ONLY_SER ➔ MEMORY_AND_DISK ➔ MEMORY_AND_DISK_SER ➔ DISK_ONLY ➔ MEMORY_ONLY_2 ➔ MEMORY_ONLY_SER_2
	MEMORY_ONLY -> storage level in persist is equal to cache () operation

15.Difference between Serialization () and Non-serialization ()?

Serialization ()	**Deserialization ()**
Data stored in Bytes format	The data is stored in its original format, which could be an object, a string, an integer, or any other data type that is supported by the programming language being used.
It required less storage.	It required more Storage.
It increases processing cost to convert to data from bytes to actual data	Processing is bit fast because data is already in original format.

16.Difference between DAG and Lineage?

DAG (Directed Acyclic Graph)	**Lineage graph**
It's an Acyclic graph	Its dependency graph
It will be helpful to know about jobs, storage and tasks.	It will be helpful to regenerate the failed RDDs.
There is no specific "bottom-to-top" or "top-to-bottom" reading approach	Its Bottom to Top reading approach and shows dependency on various RDDs and it's a Logical Plan

17.Difference between SparkContext () and SparkSession ()?

For more Follow:
https://www.linkedin.com/in/nandeshreddy/

SparkContext ()	SparkSession ()
We can't create Data Frames using SparkContext ()	SparkSession () which is higher level API build on top of 'SparkContext' does supports Data Frame
Separate context for each and everything, like • Spark context • Hive context • SQL context	• Unified entry point of spark application, it provides a way to interact with various spark functionalities with a lesser no of constructs. • Instead of having • Spark context • Hive context • SQL context Now all of it is encapsulated in a Spark Session
• Spark Context is singleton objects in Apache Spark, ensuring that there is a single-entry point to the Spark core for a given application and maintaining a consistent state across all Spark features.	• Spark Session also a is singleton objects in Apache Spark, ensuring that there is a single-entry point to the Spark core for a given application and maintaining a consistent state across all Spark features.

18.Differences between DataFrames and DataSets?

DataFrames	DataSets
When we are dealing with DataFrames, serialization is manged by 'tungsten binary format' and its very efficient and fast.	When we are dealing with DataSets, serialization is manged by Java Serialization and its bit slow compared to DataFrames.
DataFrames are often referred to as "**Spark-specific**" because they are a feature that was introduced in Spark and are specifically designed to work with Spark's execution engine	Datasets are often referred to as "**Java-specific**" because they are based on Java's type system and provide a strongly-typed API.

19.Different type of Read Modes and Write Modes in spark?

Read Modes	Explanation
permissive	It sets all fields to NULL, when it encounters a corrupted record, it's the default read mode and places the corrupted record in a string column called '_corrupt_record'.
dropMalformed	It will simply ignore the corrupted record

For more Follow:
https://www.linkedin.com/in/nandeshreddy/

failFast	Whenever the corrupted record is found, it will raise an exception and it won't display any output.

Write Modes	**Explanation**
append	Appends the output files to the directory if it already exists
overwrite	First it will delete the existing directory and then it will create a new directory.
errorIfExists	Throws an error if the directory already exists.
Ignore	If output directory exists ,it won't do anything.

20.What are the different types of Schemas in Spark?

Schema Types	**Explanation**
InferSchema	System analyses the data and fetch the schema for each column
ImplicitSchema	Some file formats like JSON. Parquet by default have schema.
Explicit schema:	1. **Programmatically Specified Schema**: Specifies the schema of the data source programmatically using the StructType and StructField classes. 2. **DDL string schema**: Define the schema of a DataFrame or Dataset by providing a DDL string and we should be using python/scala datatype approach not Struct Type.

For more Follow:
https://www.linkedin.com/in/nandeshreddy/