1. **Architecture of Spark:**
   - Spark follows a master-slave architecture.
   - It consists of a cluster manager, a master node, and multiple worker nodes.
   - The cluster manager allocates resources and schedules tasks across the worker nodes.
   - Master node coordinates the execution of tasks and manages the overall job execution.
   - Worker nodes perform the actual data processing tasks and store data in memory or disk.

2. Difference between Hadoop and Spark:
   - Hadoop is a distributed storage (HDFS) and processing (MapReduce) framework, mainly for batch processing.
   - Spark is a fast, in-memory data processing engine that supports both batch processing and real-time stream processing.
   - Spark can cache data in memory, making it faster than Hadoop's disk-based processing.
   - Spark provides a more extensive set of libraries (Spark SQL, MLlib, GraphX, etc.) compared to Hadoop.

3. Difference between RDD, Dataframe, Dataset:
   - RDD (Resilient Distributed Dataset) is Spark's fundamental data structure, representing a distributed collection of objects that can be operated on in parallel.
   - DataFrame is a distributed collection of data organized into named columns, similar to a table in a relational database.
   - Dataset is an extension of DataFrame API with type-safety, offering the benefits of RDDs and DataFrames.

4. Similarities in all APIs of Spark:

- All APIs in Spark (RDD, DataFrame, Dataset) support distributed processing.
- They provide fault tolerance and resilience.
- They offer a wide range of transformation and action operations for data manipulation.
- They can be used for both batch and real-time data processing.

5. Transformation:

- Transformations in Spark are operations applied to RDDs or DataFrames to create a new RDD or DataFrame.
- Transformations are lazy, meaning they don't compute results immediately but instead create a lineage of transformations.
- Examples include map, filter, reduceByKey, join, etc.

6. Actions in Spark:

- Actions trigger the execution of the transformation operations and return a result to the driver program or write data to external storage.
- Examples of actions include collect, count, saveAsTextFile, reduce, etc.

7. Wide Transformation:

- Wide transformations involve shuffling data across partitions, typically resulting in a stage boundary in Spark's execution plan.
- Examples include groupByKey, sortByKey, join, etc.
- Example: groupByKey operation on a large dataset where data needs to be shuffled across partitions to group records with the same key.

8. Narrow Transformation:

- Narrow transformations do not require shuffling of data across partitions and can be performed within each partition independently.
- Examples include map, filter, flatMap, etc.
- Example: map operation where a function is applied to each element of an RDD without any data shuffling.

9. Query of wide and narrow transformation with example:

- Wide Transformation Example (groupByKey):

```
val data = sc.parallelize(Seq(("a", 1), ("b", 2), ("a", 3)))
val groupedData = data.groupByKey()
```

- Narrow Transformation Example (map):

```
val data = sc.parallelize(Seq(1, 2, 3, 4, 5))
val squaredData = data.map(x => x * x)
```

10. Kerberos Architecture:

- Kerberos is a network authentication protocol designed to provide strong authentication for client/server applications.
- It uses symmetric key cryptography and operates based on tickets.
- Components include the Key Distribution Center (KDC), which issues tickets, and clients and servers, which authenticate using these tickets.
- The architecture involves a Ticket Granting Ticket (TGT), which is obtained by the client to request service tickets from the Ticket Granting Service (TGS).
- Once authenticated, the client can access the requested services without sending its credentials again.

- Kerberos ensures secure communication over insecure networks by encrypting the tickets and credentials exchanged between parties.