**What are the advantages and disadvantages of PySpark?**

* Simple to use: Parallelized code can be written in a simpler manner.
* Error Handling: PySpark framework easily handles errors.
* Inbuilt Algorithms: PySpark provides many of the useful algorithms in Machine Learning or Graphs.
* Library Support: Compared to Scala, Python has a huge library collection for working in the field of data science and data visualization.
* Easy to Learn: PySpark is an easy to learn

### What is PySpark SparkContext?

* PySpark SparkContext is an initial entry point of the spark functionality. It also represents Spark Cluster Connection and can be used for creating the Spark RDDs (Resilient Distributed Datasets) and broadcasting the variables on the cluster.

### What are PySpark serializers?

* The serialization process is used to conduct performance tuning on Spark. The data sent or received over the network to the disk or memory should be persisted

### What are the different cluster manager types supported by PySpark?

* Standalone – This is a simple cluster manager that is included with Spark.
* Apache Mesos – This manager can run Hadoop MapReduce and PySpark apps.
* Hadoop YARN – This manager is used in Hadoop2.
* Kubernetes – This is an open-source cluster manager that helps in automated deployment, scaling and automatic management of containerized apps.
* local – This is simply a mode for running Spark applications on laptops/desktops

### PySpark faster than pandas?

* PySpark supports parallel execution of statements in a distributed environment, i.e on different cores and different machines which are not present in Pandas. This is why PySpark is faster than panda

### What is SparkSession in Pyspark?

* SparkSession is the entry point to PySpark and is the replacement of SparkContext since PySpark version 2.0. This acts as a starting point to access all of the PySpark functionalities related to RDDs, DataFrame, Datasets etc. It is also a Unified API that is used in replacing the SQLContext, StreamingContext, HiveContext and all other contexts.

### Is it possible to create PySpark DataFrame from external data sources?

* Yes, it is! Realtime applications make use of external file systems like local, HDFS, HBase, MySQL table, S3 Azure etc. Following example shows how we can create DataFrame by reading data from a csv file present in the local system:

### What do you understand by Pyspark’s startsWith() and endsWith() methods?

* These methods belong to the Column class and are used for searching DataFrame rows by checking if the column value starts with some value or ends with some value. They are used for filtering data in applications.

### What do you understand by Pyspark Streaming? How do you stream data using TCP/IP Protocol?

* PySpark Streaming is scalable, fault-tolerant, high throughput based processing streaming system that supports streaming as well as batch loads for supporting real-time data from data sources like TCP Socket, S3, Kafka, Twitter, file system folders etc. The processed data can be sent to live dashboards, Kafka, databases, HDFS etc.

**Why are Partitions immutable in PySpark?**

* In PySpark, every transformation generates a new partition. Partitions use HDFS API to make partitions immutable, distributed, and fault-tolerant. Partitions are also aware of data locality.

**What is the usage of PySpark StorageLevel?**

* The PySpark StorageLevel is used to control the storage of RDD. It controls how and where the RDD is stored. PySpark StorageLevel decides if the RDD is stored on the memory, over the disk, or both.

**What is PySpark SparkConf?**

* PySpark SparkConf is mainly used if we have to set a few configurations and parameters to run a Spark application on the local/cluster. In other words, we can say that PySpark SparkConf is used to provide configurations to run a Spark application.

**What is PySpark Partition? How many partitions can you make in PySpark?**

* PySpark Partition is a method of splitting a large dataset into smaller datasets based on one or more partition keys. It enhances the execution speed as transformations on partitioned data run quicker because each partition's transformations are executed in parallel. PySpark supports both partitioning in memory (DataFrame) and partitioning on disc (File system). When we make a DataFrame from a file or table, PySpark creates the DataFrame in memory with a specific number of divisions based on specified criteria.

**What do you understand by RDD Lineage?**

The RDD lineage is a procedure that is used to reconstruct the lost data partitions. The Spark does not hold up data replication in the memory. If any data is lost, we have to rebuild it using RDD lineage. This is the best use case as RDD always remembers how to construct from other datasets.

**Explain the use of StructType and StructField classes in PySpark with examples.**

The StructType and StructField classes in PySpark are used to define the schema to the DataFrame and create complex columns such as nested struct, array, and map columns. StructType is a collection of StructField objects that determines column name, column data type, field nullability, and metadata.

**What are the different types of joins?**

Joins in PySpark are used to join two DataFrames together, and by linking them together, one may join several DataFrames. INNER Join, LEFT OUTER Join, RIGHT OUTER Join, LEFT ANTI Join, LEFT SEMI Join, CROSS Join, and SELF Join are among the SQL join types it supports.

#### What is the function of PySpark's pivot() method?

The pivot() method in PySpark is used to rotate/transpose data from one column into many Dataframe columns

### What is List the main attributes used in SparkConf.

Below-listed are the most commonly used attributes of SparkConf:

* **set(key, value) -** It sets the configuration property.
* **setMaster(value) -** It sets the master URL.
* **setAppName(value) -** It sets the application name.
* **get(key, defaultValue=None) -** It gets the configuration value of a key.
* **setSparkHome(value)**

### PySpark DAGScheduler?

The complete form of DAG is Direct Acyclic Graph. It controls the scheduling layer of Spark for executing the stage-oriented scheduled tasks. This scheduler executes stages DAG for each job. Developers can keep track of all stages in RDD. Even this DAG scheduler reduces the running time.

### Differentiate between transformations and actions in PySpark.

Transformations in PySpark are operations performed on RDDs to create new RDDs. They are lazy in nature and include functions such as map(), filter(), and reduceByKey(). Actions, on the other hand, trigger computations on RDDs and produce non-RDD results. Examples of actions include count(), collect(), and reduce(). Transformations are built up in a sequence, and actions execute the transformations to produce final results.

### How does PySpark handle missing or null values in DataFrames?

PySpark represents missing or null values using the special None object or the NULL SQL value. DataFrame operations and transformations have built-in support for handling missing data. Functions such as na.drop() and na.fill() allow you to drop rows with missing values or replace them with specified values. Additionally, SQL operations such as IS NULL or IS NOT NULL can be used to filter out or include null values.

### Describe the process of reading and writing data using PySpark's DataFrame API.

PySpark's DataFrame API provides convenient methods for reading data from various data sources such as CSV, Parquet, JSON, and databases. The spark.read object is used to create a DataFrame by specifying the data source, format, and options. Conversely, data can be written using the write object, specifying the destination, format, and options. PySpark's DataFrame API handles various data formats and provides options for controlling data compression, partitioning, and more.

### What is the purpose of the groupBy() and agg() functions in PySpark?

The groupBy() function in PySpark is used to group data in a DataFrame based on one or more columns. It creates grouped DataFrames that can be further aggregated using the agg() function. The agg() function is used to perform various aggregation operations such as sum, avg, min, and max, on grouped DataFrames

### Explain the concept of Broadcast Variables in PySpark.

Broadcast Variables in PySpark are read-only variables that can be cached and shared across all worker nodes in a Spark cluster. They are used to efficiently distribute relatively small amounts of data (e.g., lookup tables) to all tasks in a job, reducing the need for data shuffling.

### How can you optimise the performance of PySpark jobs?

Optimising PySpark performance involves various strategies. These include using appropriate transformations to minimise data shuffling, leveraging caching and persistence to avoid recomputation, adjusting the number of partitions for efficient parallelism, and using broadcast variables for small data.

### What is lazy evaluation in PySpark, and why is it important?

Lazy evaluation in PySpark refers to the delayed execution of transformations until an action is invoked. When transformations are called, they build up a logical execution plan (or DAG - Directed Acyclic Graph) without executing any computation.

### Explain the concept of accumulators in PySpark.

Accumulators in PySpark are variables used for aggregating information across all the nodes in a distributed computation. They provide a mechanism to update a variable in a distributed and fault-tolerant way, allowing values to be aggregated from various nodes and collected to the driver program

### What is the role of the Driver and Executor in a PySpark application?

In a PySpark application, the Driver is the main program that contains the user's code and orchestrates the execution of the job. It communicates with the cluster manager to acquire resources and coordinate task execution. Executors, on the other hand, are worker nodes that perform the actual computation. They execute the tasks assigned by the Driver and manage the data residing in their assigned partitions.

### What is Apache Spark and why is it preferred over Hadoop MapReduce?

Apache Spark is an open-source distributed computing system that provides a powerful and flexible framework for big data processing. It is preferred over Hadoop MapReduce due to its in-memory computation, which enhances speed and efficiency. Spark offers various APIs and libraries, including PySpark for Python, making it more versatile and developer-friendly than Hadoop MapReduce.

### What is Parquet file in PySpark?

The Parquet file in PySpark is defined as a column-type format supported by different data processing systems. It helps Spark SQL to perform read and write operations. Its column-type format storage offers numerous benefits, such as consuming less space, allowing you to retrieve specific columns for access, employing type-specific encoding, providing better-summarised data, and supporting limited I/O operations.

### Describe the concept of a shuffle operation in Spark.

A shuffle operation in Spark refers to the process of redistributing data across partitions, typically performed when there is a change in the partitioning of data. It involves a data exchange between different nodes in the cluster as records need to be shuffled to their appropriate partitions based on a new partitioning scheme. Shuffles are expensive operations in terms of network and disk I/O,

### How can you persist RDDs in Spark?

RDDs in Spark can be persisted in memory or on disk using the persist() or cache() methods. When an RDD is persisted, its partitions are stored in memory or on disk, depending on the storage level specified.

### How does Spark handle data partitioning?

Spark handles data partitioning by distributing the data across multiple partitions in RDDs or DataFrames. Data partitioning allows Spark to process the data in parallel across a cluster of machines.

Spark provides control over partitioning through partitioning functions or by specifying the number of partitions explicitly. It also performs automatic data partitioning during shuffle operations to optimize data locality and parallelism.

### Describe the difference between narrow and wide transformations in Spark.

In Spark, narrow transformations are operations where each input partition contributes to only one output partition. Narrow transformations are performed locally on each partition without the need for data shuffling across the network. Examples of narrow transformations include map(), filter(), and union().

Wide transformations, on the other hand, are operations that require data shuffling across partitions. They involve operations like grouping, aggregating, or joining data across multiple partitions. Wide transformations result in a change in the number of partitions and often require network communication. Examples of wide transformations are groupByKey(), reduceByKey(), and join().

### 

### What is the difference between cache() and persist() methods in Spark?

Both cache() and persist() methods in Spark allow you to persist RDDs or DataFrames in memory or on disk. The main difference lies in the default storage level used:

cache(): This method is a shorthand for persist(MEMORY\_ONLY) and stores the RDD or DataFrame partitions in memory by default.

persist(): This method allows you to specify the storage level explicitly. You can choose from different storage levels, such as MEMORY\_ONLY, MEMORY\_AND\_DISK or DISK\_ONLY, based on your requirements.

Both methods provide the same functionality of persisting the data. You can specify the storage level for persist() to achieve the same behavior as cache().