1. What is role of MapReduce in Spark?

   MapReduce has no role in spark.
   They two are completely different processing engine. MapReduce is original processing engine of Hadoop.
   - **Map Phase:** Breaks input data into smaller chunks and processes each chunk independently, generating intermediate key-value pairs.
   - **Reduce Phase:** Aggregates and merges these intermediate results to produce the final output, typically involving grouping, tabulating, or summarizing data

   MapReduce has two phase Map Phase – for filter and sorting the data and Reduce phase – for aggregate the data, with intermediate data being written to disk between each step it major drawback.

   spark replaces the processing engine.
   - **MapReduce = a parallel processing unit in Hadoop** (disk-based, rigid).
   - **Spark = a parallel processing engine that evolved from MapReduce concepts** (memory-based, DAG-driven, more efficient).

---

2. Diff btw spark and Hadoop

	Hadoop	Spark
speed	Hadoop's MapReduce model reads and writes from a disk, thus slowing down the processing speed.	Spark reduces the number of read/write cycles to disk and stores intermediate data in memory, hence faster-processing speed.
Fault tolerance	Replication	Lineage
	Hadoop is designed to handle batch processing efficiently.	Spark is designed to handle both batch and real-time data efficiently.
Cost	Hadoop is a cheaper option available while comparing it in terms of cost	Spark requires a lot of RAM to run in-memory, thus increasing the cluster and hence cost.
	YARN is the most common option for resource management.	Standalone, Mesos, YARN,
	Java	Python, java, R, SQL

3. what is spark?

Spark is open-source, distributed computing framework designed for big data processing.

It allows you to **process huge volumes of data quickly** by splitting work across multiple computers (a cluster) and performing tasks in **parallel**.

Unified engine: unlike older systems that require different tools for different task, spark provides a single, unified engine for a wide range of data processing.

---

4. what is in-memory processing in spark?

In-memory processing is core technology of spark which make spark so faster than older big data framework like HADOOP MapReduce.

Instead of writing intermediate data to a slow disk, spark keeps the data in RAM.

Spark achieves this through its main -- In Spark → data can be kept in **memory (RAM)** using **RDDs/DataFrames caching & persistence**, so the next operations can access it directly without expensive disk reads/writes.

How it Works

1. When you perform transformations (map, filter, join, etc.), Spark builds a **DAG (Directed Acyclic Graph)** of operations.
2. Spark keeps intermediate results in **memory** as long as resources allow.
3. If memory is not enough, Spark will **spill to disk** (still more efficient than Hadoop because only when needed).

---

5. diff btw cache and persist?

Cache and Persist in Spark are techniques to **store intermediate data in memory or disk** to avoid re-computation and improve performance.

- Shortcut for persist() with **MEMORY_ONLY** storage level.

Use it when the dataset is **small/medium, fits in memory**, and will be **reused multiple times.**

Persist:

- Provides **flexible storage options** like MEMORY_ONLY, MEMORY_AND_DISK, DISK_ONLY, MEMORY_ONLY_SER, etc.
- Use it when **dataset is large**, or you want **control over memory/disk usage**, especially for **iterative jobs or expensive transformations**.

---

6. **How Parallelism is Achieved in Spark**

Spark achieves parallelism by splitting data into partitions and running one task per partition across multiple executors in the cluster. The driver coordinates the DAG of stages, and each stage executes tasks in parallel. The degree of parallelism is controlled by the number of partitions and cluster resources (cores, executors)

Example: If you have 1 million rows split into 100 partitions, Spark can process 100 partitions in parallel.

---

7. **What is a Partition in Spark?**

- A **partition** is the **smallest logical chunk of data** in Spark.
- Each partition is processed by **a single task** in an executor.
- Spark achieves **parallelism** by processing multiple partitions at the same time across the cluster.

👉 Think of partition = a “slice” of the dataset.

How Partitions are Created in Spark?

1. **From Data Sources**
   - When reading data, Spark decides the number of partitions:
     - **HDFS / S3 / GCS** → based on file block size (default 128 MB in HDFS).

```
rdd = spark.sparkContext.parallelize([1,2,3,4,5,6], 3)

print(rdd.getNumPartitions())  # 3 partitions
```

Example:

- A file = **640 MB**
- Block size = **128 MB**
- 👉 Spark creates **5 partitions** (640 ÷ 128 = 5).

---

8. I Want see the Which brand has orange colour variant

```
car_list = [
  {'Brand': 'Hyundai', 'Year': 2021, 'Colors': ['Red', 'Blue', 'While', 'Black']},
  {'Brand': 'Maruthi', 'Year': 2022, 'Colors': ['Red', 'Blue', 'While', 'Green']},
  {'Brand': 'TATA', 'Year': 2022, 'Colors': ['Orange', 'Blue', 'While', 'Black']}
]
```

option 1. If you are using python, print the Brand name

```
for ch in car_list:
   if 'Orange' in ch['Colors']: print(ch['Brand'])
```

option 2. If you are using pyspark, print the whole row df = spark.createDataFrame(car_list)

df.filter(array_contains(df.Colors, 'Orange')).display()

---

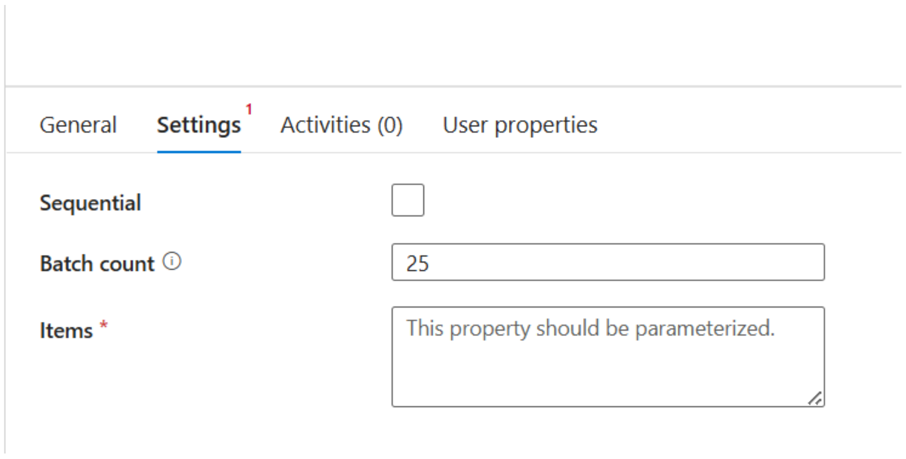
9. If there 100 file in source and I wanted to process those in concurrent way like 25, 25, 25, 25

Scenario

- **Source:** 100 files
- **Goal:** Process them concurrently in batches of 25 (so 4 batches in parallel).

Step 1: Use a ForEach Activity with Concurrency

1. **Get list of files**
   - Use a **Get Metadata** activity pointing to the folder.
   - Select **child items** → returns the list of 100 files.
2. **ForEach Activity**
   - Connect **Get Metadata** → **ForEach**.
   - In the ForEach activity, set **Items = @activity('GetMetadata').output.childItems**
3. **Enable concurrency**
   - In the **Settings** tab of ForEach:
     - **Sequential = false** → allows parallel execution.
     - **Batch Count = 25** → this controls how many files are processed concurrently.



Sequential ☐

Batch count: 25

Items *: This property should be parameterized.

Batch Count unspecified (empty) → ADF uses **the default maximum parallelism**, which is:

- **20 parallel iterations** by default.

- **Checked (true):** Iterations are **sequential** → one file processed at a time.
- **Unchecked (false):** Iterations are **parallel** → multiple files processed at the same time.

---

8. How do you handle the error in ADF?

Steps:

1. Add the main activity (e.g., Copy Data, Data Flow, Databricks Notebook).
2. Add a **Failure path** using the **On Failure** dependency.
3. Connect it to an **error handling activity**:
    - Send email (Logic App, Webhook, SMTP)
    - Log error to database or storage
    - Trigger alternate workflow

```
[Copy Data Activity]
  |-- On Success --> [Next Activity / Continue Pipeline]
  |
  |-- On Failure --> [Set Variable: ErrorFlag = true]
                  |
                  v
           [Send Email Notification / Log Error]
```

Have done any source Integration?

Yes, I have worked on source integration in Azure Data Factory (ADF)

Files / Storage: Azure Blob Storage, ADLS Gen2, CSV, Parquet, JSON

For example, in ADF, I used the **Copy Data activity** to extract data from a **Blob storage folder containing multiple CSV files**, applied schema mapping, and loaded the data into **Azure SQL / Delta Lake tables**.

Type of Cluster

Option	**Description**
New Job Cluster	- A **temporary cluster** created only for this job execution. - Automatically **terminated after the job finishes**. - Recommended for production jobs to **save cost** and avoid interfering with other workloads.
Existing Interactive Cluster	- Uses a **currently running interactive cluster**. - Useful for **ad-hoc development or testing**. - Not recommended for production jobs because the cluster might be busy with other tasks.
Existing Instance Pool	- Uses a **predefined pool of compute instances**. - Reduces **cluster startup time**. - You still need to attach a cluster from the pool.
Serverless	- Uses **Databricks serverless compute** (if your workspace supports it). - No cluster management required; Databricks auto-scales. - Only available in **premium workspaces and supported regions**.

In ADF Do We have any option to send the Notifications?

1 Using Web Activity with Logic Apps or Functions

- **Step: Add a Web Activity at the end of your pipeline.**
- **Purpose: Call an external service to send notifications (email, Teams, Slack, etc.).**
- **Example:**
  - **Create a Logic App that sends an email.**
  - **In ADF, Web Activity calls the Logic App URL with pipeline status as payload.**

2 Using Set Variable + If Condition + Web Activity

- **Pattern:**
  1. **On Failure → set a variable ErrorFlag = true**
  2. **At the end of pipeline, use If Condition → check ErrorFlag**
  3. **Inside True branch → use Web Activity or Logic App to send email**