**RDDs in Apache Spark**

**1. What is an RDD?**

* **RDD** = **Resilient Distributed Dataset**
* Spark’s **core data structure**.
* **Immutable, distributed collection of objects** that can be processed in parallel across a cluster.

**2. Key Properties of RDD**

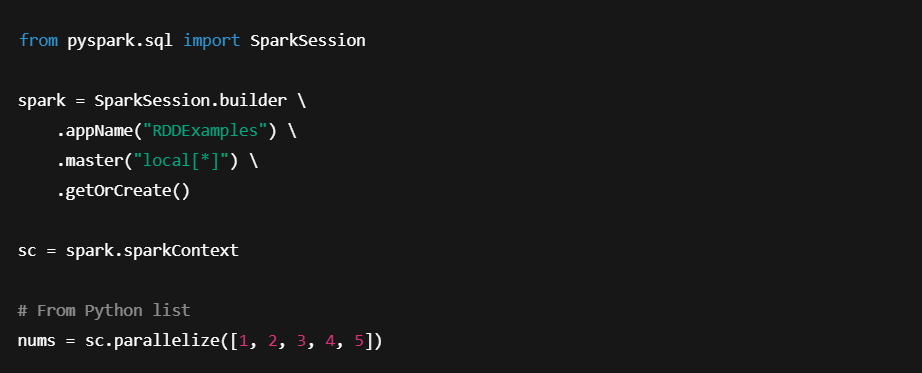
1. **Resilient** – Fault-tolerant.
   * If part of data is lost (e.g., node failure), Spark recomputes it using original transformations (**lineage**).
2. **Distributed** –
   * Data is **split into partitions** across multiple nodes in a cluster.
3. **Dataset** –
   * Collection of elements (like list/array) that can be processed using **functional transformations** (map, filter, reduce).

**3. How RDD Works**

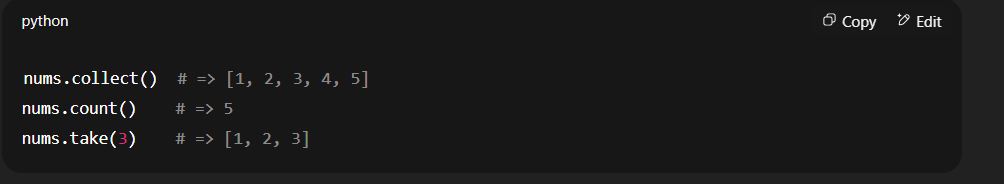
* **Creation:**
  + From existing data sources (Python list, HDFS, S3, local files).
  + By transforming existing RDDs.
* **Lazy Operations:**
  + Transformations (map, filter) are **lazy** – they are not executed until an **action** (collect, count) is called.
* **Lineage:**
  + Spark tracks **how the RDD was created**.
  + Helps in **rebuilding lost data** if a node fails.

**Creating and Using RDDs**

**1. Creating RDDs**



**2. Basic Actions on RDD**

****

* **collect()** → Returns all elements.
* **count()** → Returns total number of elements.
* **take(n)** → Returns first n elements.

**3. Why RDDs?**

* **Low-level control** over data.
* Suitable for **complex data transformations**.
* **Fault tolerance** through lineage.
* Works well for **iterative algorithms** (e.g., ML).

**Transformations & Lazy Evaluation**

**1. Transformations in RDD**

* Operations that create a **new RDD from an existing one**.
* **Lazy** – they don’t execute immediately.
* Examples:
  + .map() → Applies a function to each element.
  + .flatMap() → Similar to map but flattens results.
  + .filter() → Keeps only elements satisfying a condition.
  + .reduceByKey() → Aggregates values by key.

**2. Lazy Evaluation**

* Transformations **build a logical plan** (DAG – Directed Acyclic Graph).
* Execution happens **only when an action** is called.
* Benefits:
  + **Optimization** – Spark can rearrange operations for efficiency.
  + **No unnecessary computation**.

**3. Actions in RDD**

* Trigger the **execution** of transformations.
* Examples:
  + .collect() → Brings results to driver.
  + .count() → Counts elements.
  + .saveAsTextFile() → Saves results to storage.

**4. DAG & Execution**

1. Transformations create **DAG** of stages.
2. Spark **optimizes** the DAG.
3. Execution starts when an **action** is called.
4. Tasks are **distributed** across cluster.

**Step-by-Step Understanding**

1. **flatMap()**
   * Breaks each sentence into individual words.
   * Unlike map(), it **flattens** the result into one list.
2. **map()**
   * Converts each word into a (word, 1) pair for counting.
3. **filter()**
   * Removes unwanted words (e.g., "is").
4. **reduceByKey()**
   * Groups by key (word) and sums the counts.
5. **collect()**
   * Brings all results back to the driver as a Python list.
6. **count()**
   * Counts the number of unique words in the result.

