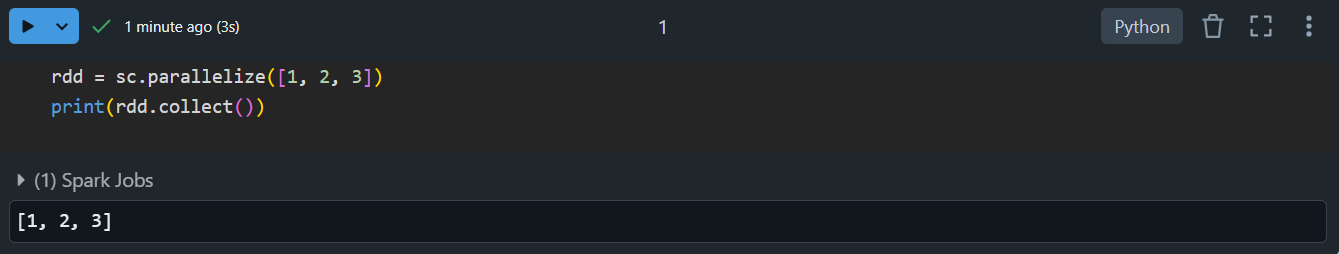
**Assignment 3**

Name: **E.R Harish**

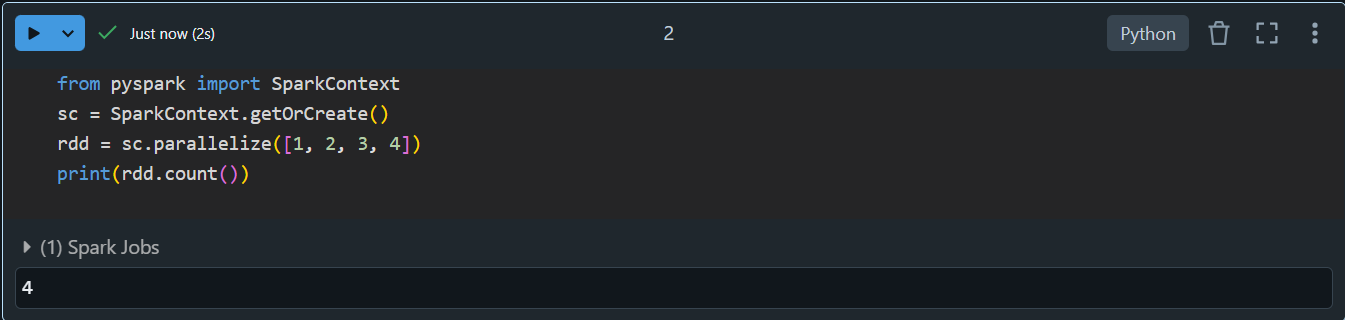
1. **Actions in PySpark RDDs**
2. **The .collect() Action**

****

**Summary:**

Returns all elements of the RDD as a list. Useful for debugging and small datasets.

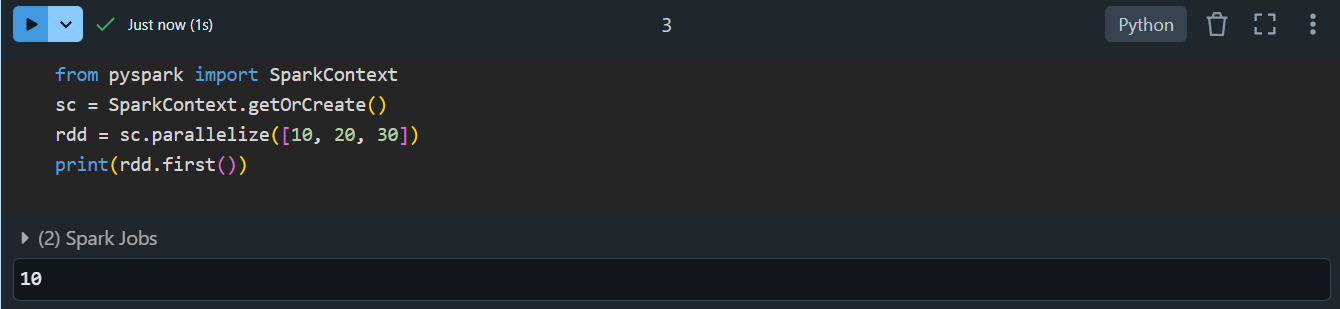
1. **The .count() Action**

****

**Summary:**

Counts the total number of elements in an RDD. Helps verify dataset size.

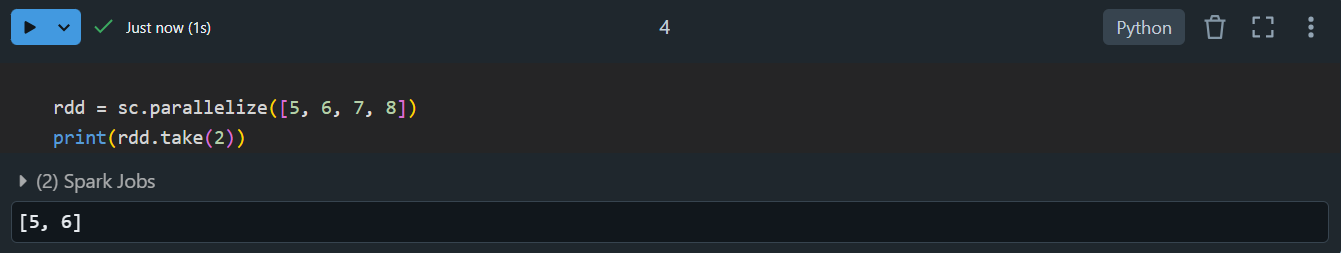
1. **The .first() Action**

****

**Summary:**

Retrieves the first element of an RDD. Good for quick data validation.

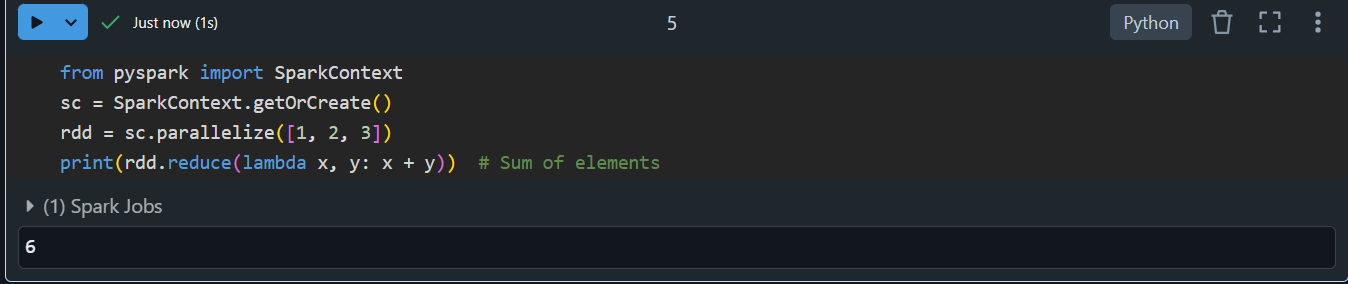
1. **The .take() Action**

****

**Summary:**

Fetches the first n elements of the RDD. Useful for inspecting a subset of data.

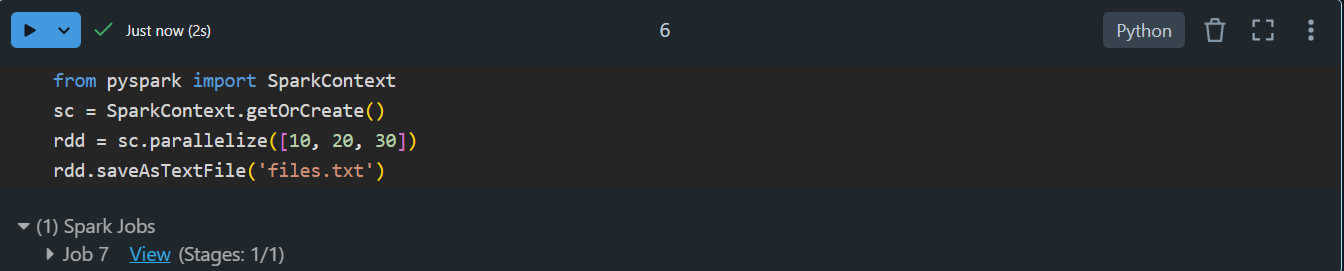
1. **The .reduce() Action**

****

**Summary:**

Aggregates elements using a specified binary operation (e.g., summing all values).

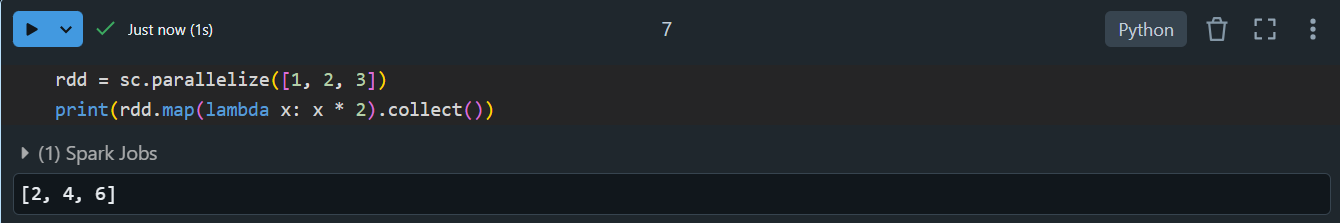
1. **The .saveAsTextFile() Action**

****

**Summary:**

Saves the RDD’s content as a text file in the specified directory. Creates partitions as separate files.

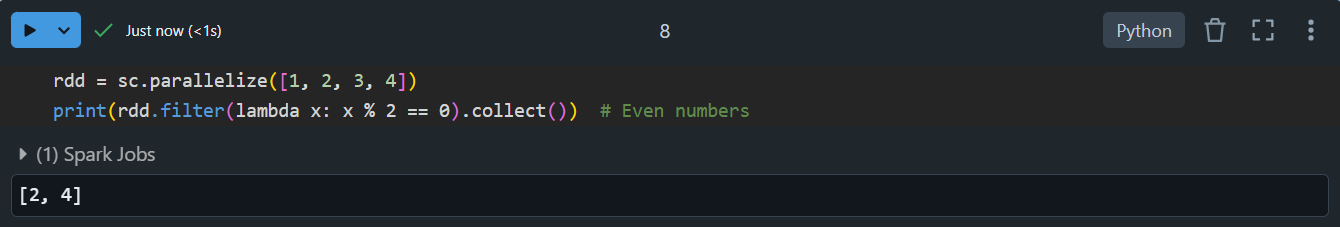
1. **Transformations in PySpark RDDs**
2. **The .map() Transformation**

****

**Summary:**

Applies a function to each element and returns a new RDD. Example: Add 10 to every number.

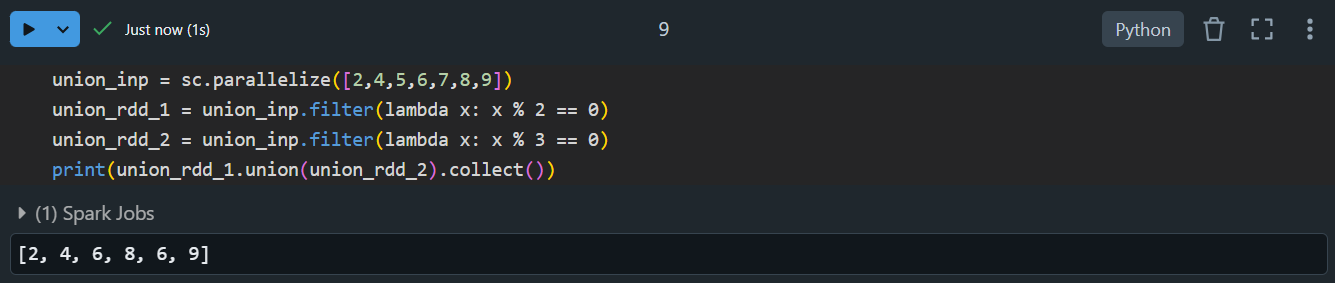
1. **The .filter() Transformation**

****

**Summary:**

Filters elements based on a condition, returning a new RDD. Example: Retain only even numbers.

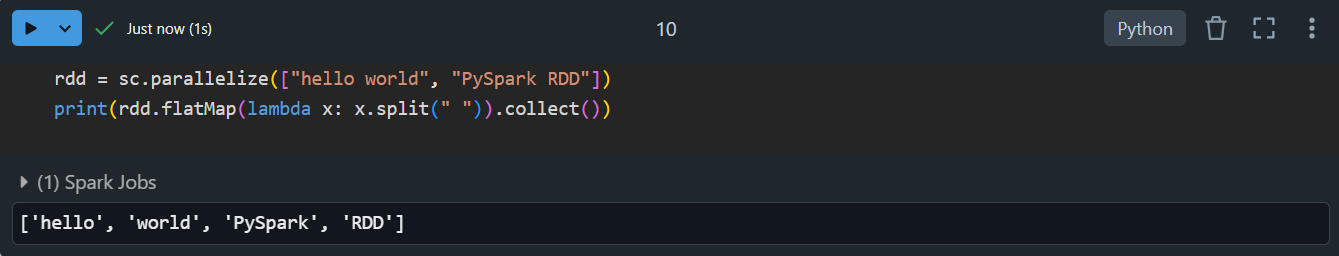
1. **The .union() Transformation**

****

**Summary:**

Combines two RDDs into one containing all elements from both RDDs.

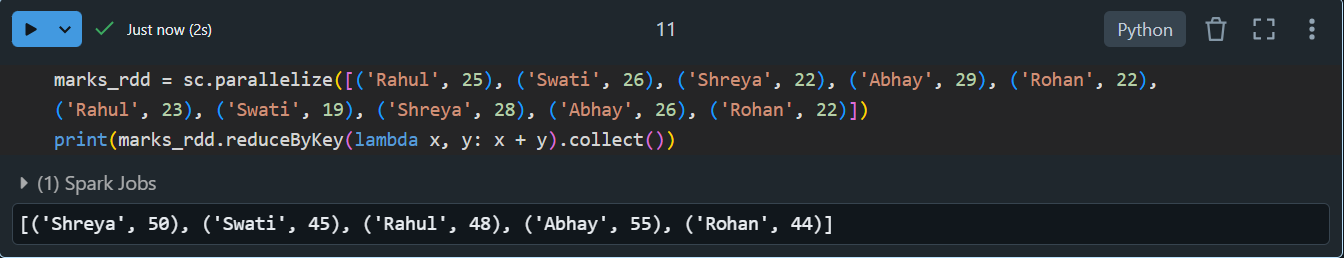
1. **The .flatMap() Transformation**

****

**Summary:**

Similar to .map(), but flattens the output. Useful for splitting strings into words.

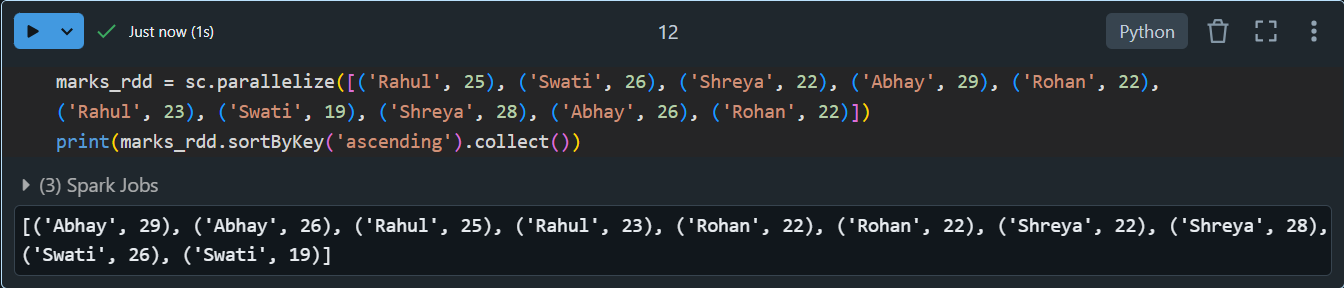
1. **Transformations in Pair RDDs**
2. **The .reduceByKey() Transformation**

****

**Summary:**

Combines values for each key using a binary operation (e.g., summing marks for students with the same name).

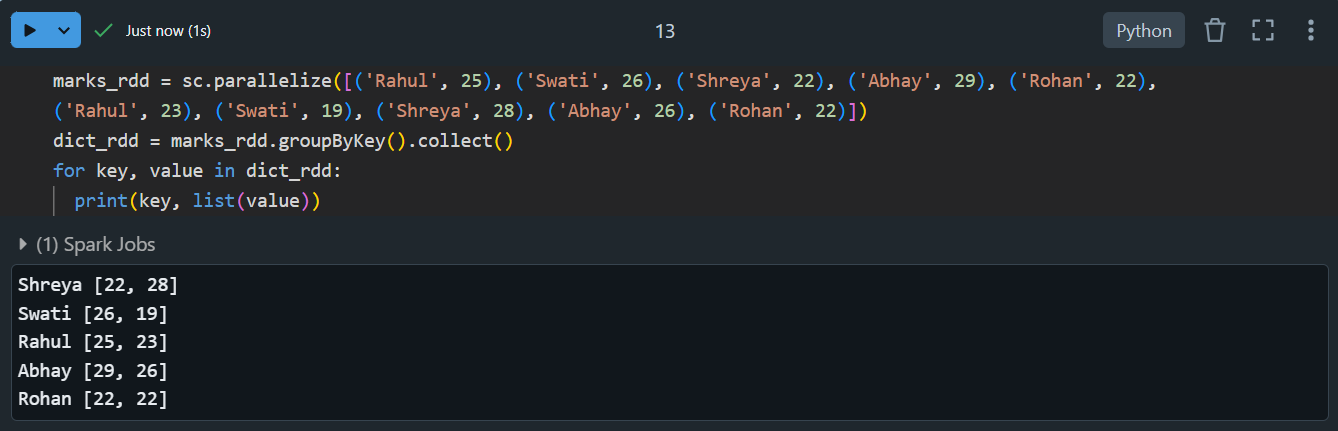
1. **The .sortByKey() Transformation**

****

**Summary:**

Sorts key-value pairs by keys in ascending or descending order.

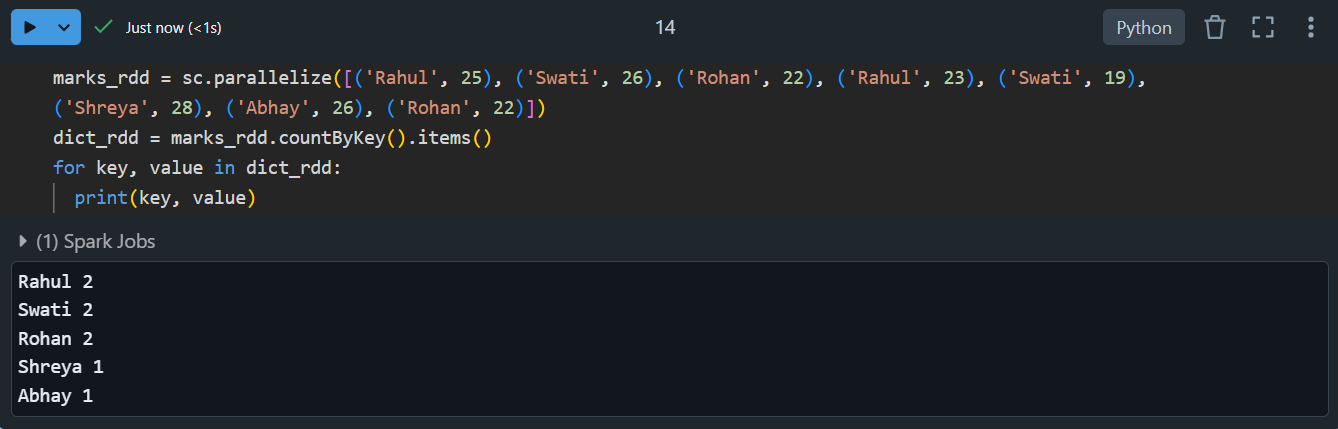
1. **The .groupByKey() Transformation**

****

**Summary:**

Groups values by their keys, returning an RDD with each key and its associated list of values.

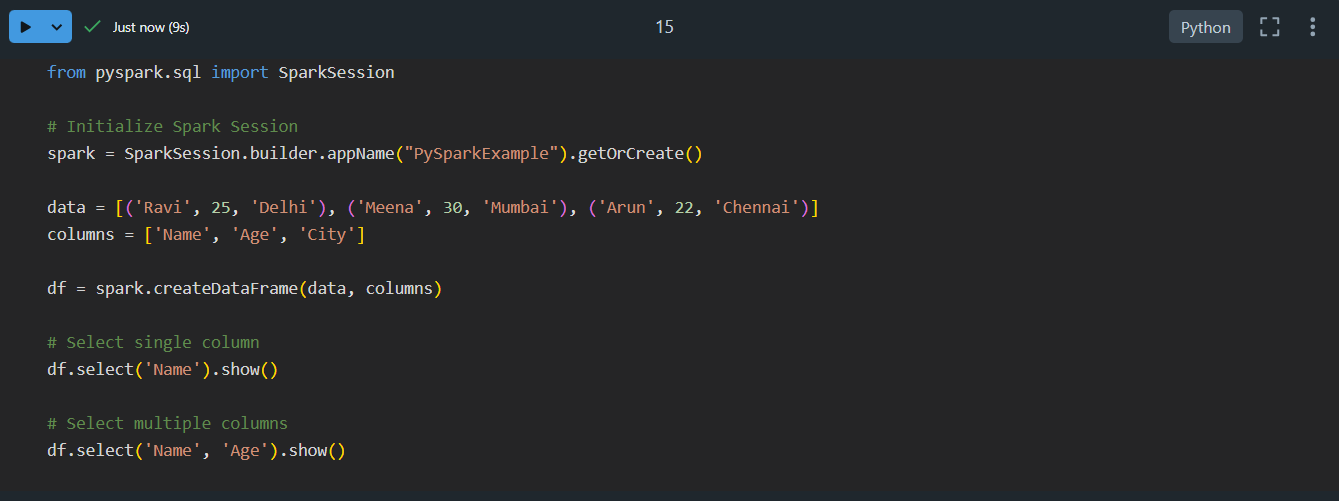
1. **Actions in Pair RDDs**
2. **The countByKey() Action**

****

**Summary:**

Counts the number of values for each key and returns a dictionary.

1. **Working with Pandas**
2. **Selecting, Renaming, and Filtering Data in a DataFrame.**

****

****

**Summary:**

* **Selecting Columns:**

Retrieve specific columns using df['column\_name'] or df[['col1', 'col2']].

* **Renaming Columns:**

Use .rename(columns={'old\_name': 'new\_name'}) to rename columns.

* **Filtering Rows:**

Apply conditions like df[df['column'] > value] to filter rows based on criteria.

1. **Manipulating, Dropping, Sorting, Aggregating, Joining, and Grouping DataFrames**

****

****

**Summary:**

* **Manipulating Data:**

Modify values or create new columns using operations like df['new\_col'] = df['col'] \* 2.

* **Dropping Data:**

Use .drop(columns=['col']) to remove columns or .drop(index) for rows.

* **Sorting Data:**

Sort values using .sort\_values(by='col', ascending=True/False).

* **Aggregations:**

Apply functions like .sum(), .mean(), or .agg({'col1': 'sum', 'col2': 'max'}).

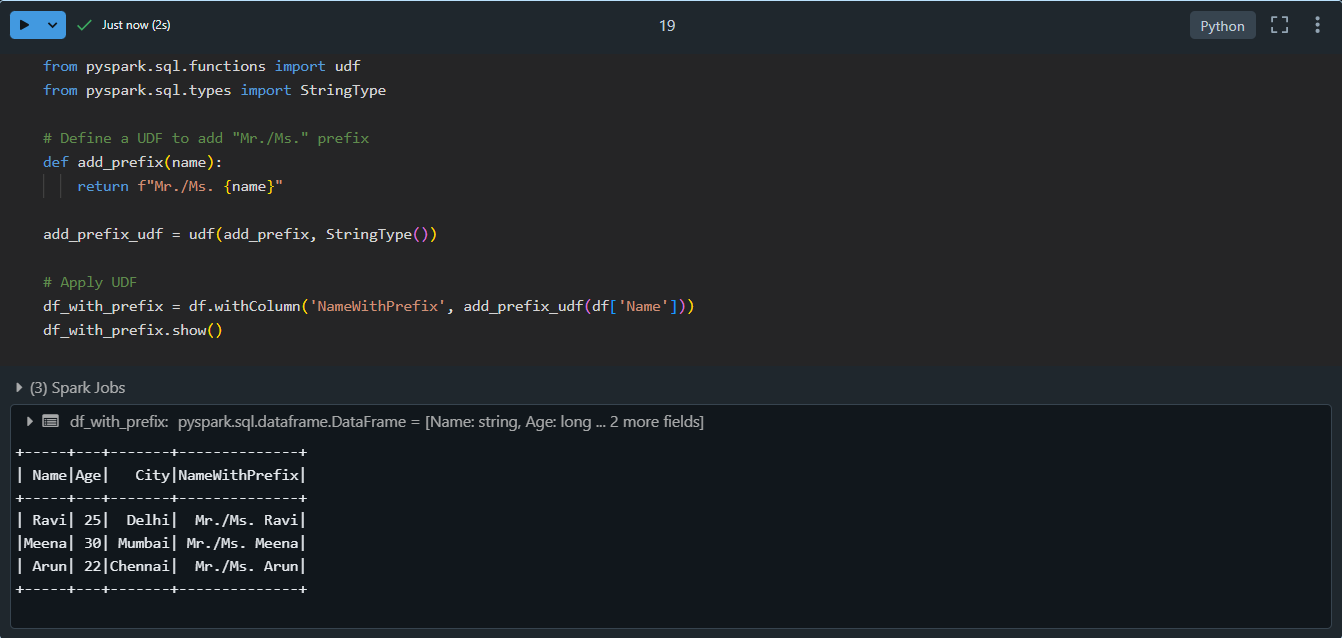
* **Joining DataFrames:**

Combine DataFrames with .merge() for relational joins or .concat() for stacking.

* **Grouping Data:**

Use .groupby('col').agg() for operations like grouping and applying aggregations.

1. **Applying Functions in a DataFrame**

****

**Summary:**

* **Element-wise Operations:**

Use .apply() to apply a function to rows or columns, e.g., df['col'].apply(lambda x: x\*2).

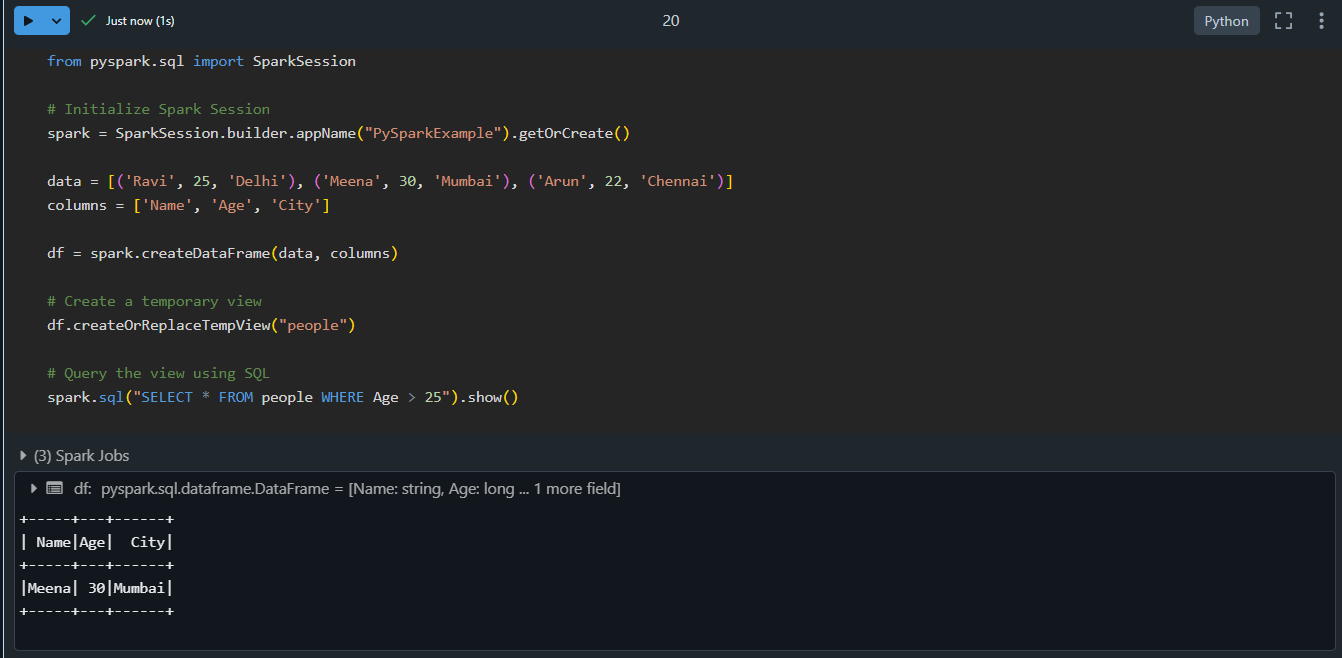
* **Row/Column-wise Operations:**

Apply functions row-wise (axis=1) or column-wise (axis=0).

* **Vectorized Operations:**

Leverage NumPy or Pandas for efficient operations directly on columns, e.g., df['col'] \* 10.

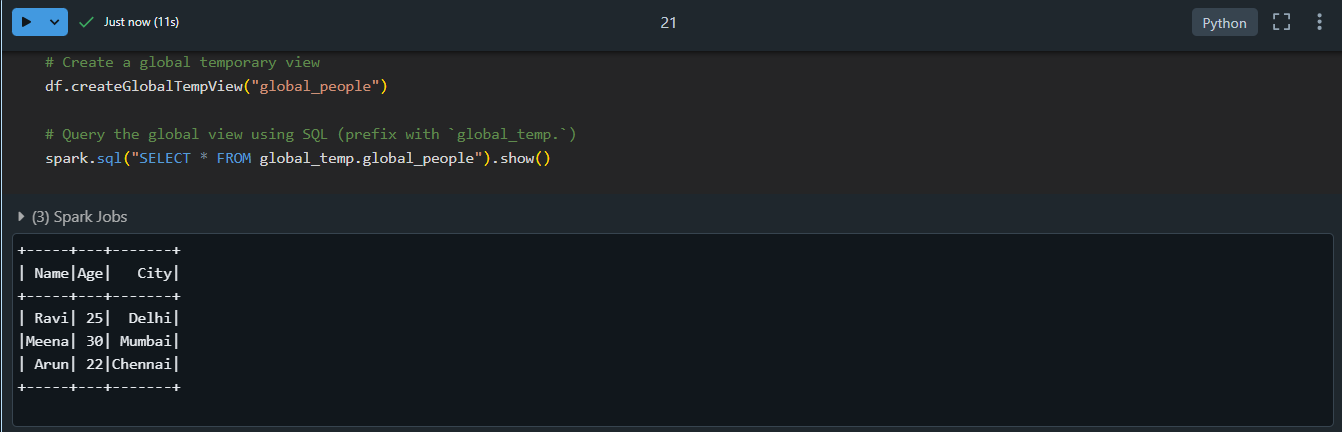
1. **PySpark: Creating Local and Temporary Views**
2. **Creating Temporary Views**

****

**Summary:**

* **Temporary Views:** Session-scoped views created using createOrReplaceTempView. Useful for querying DataFrames with SQL.

1. **Creating Global Temporary Views**

****

**Summary:**

* Global Temporary Views: Accessible across multiple sessions within the same Spark application using createGlobalTempView. Use the global\_temp prefix to query them.