JOBS, STAGES & TASKS

LETS BREAK IT DOWN

linkedin.com/in/arudsekaberne/

Oh! Looks like, I got an email notification

Let's have a look

Analyzing Requirement

Spark [jobs, stages, and tasks] Python

File Edit View Run Help Last edit was 2 hours ago Give feedback

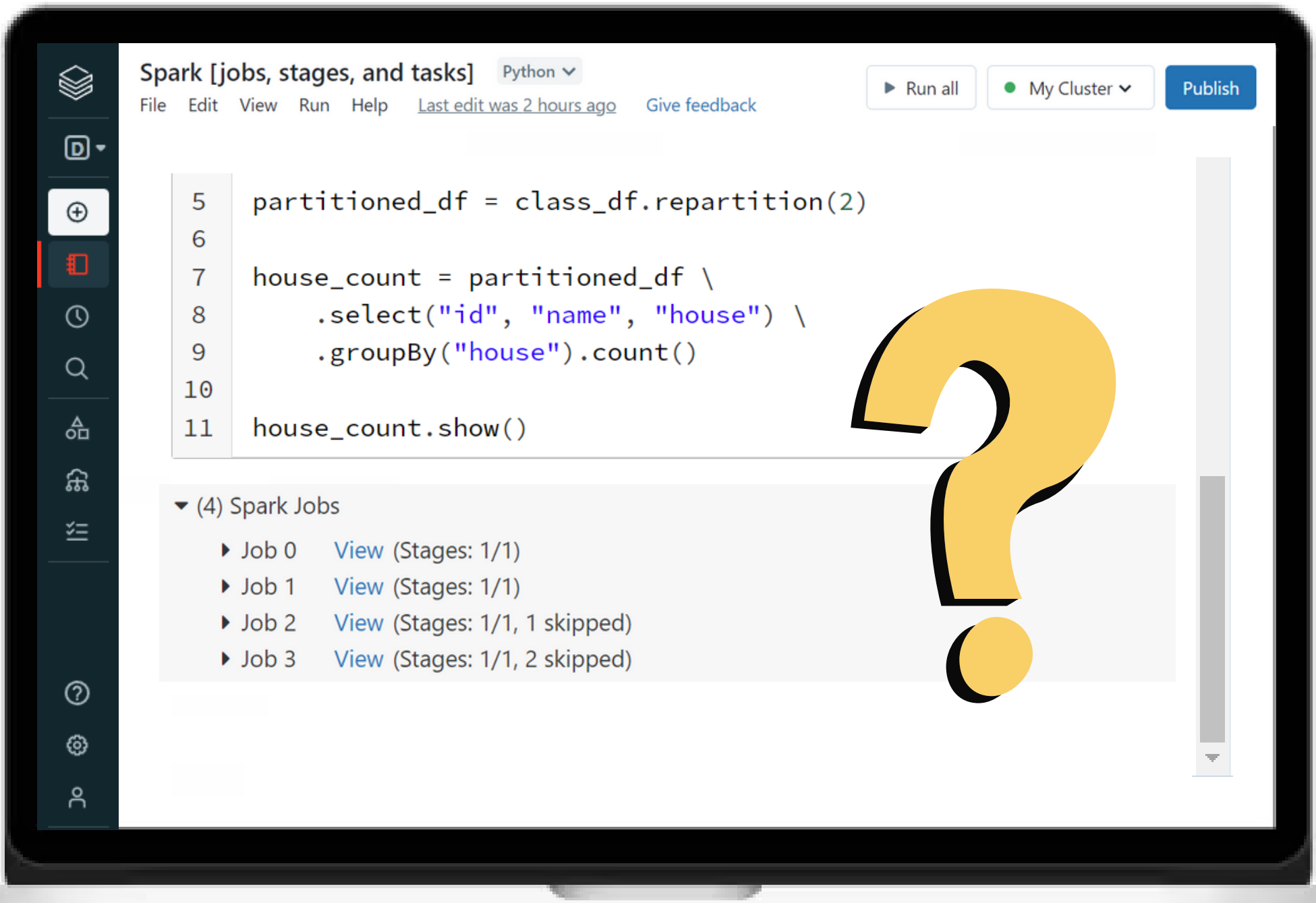
Run all My Cluster Publish

```python
class_df = spark.read \
    .options(header=True, schema=class_schema) \
    .csv(file_path)

partitioned_df = class_df.repartition(2)

house_count = partitioned_df \
    .select("id", "name", "house") \
    .groupBy("house").count()

house_count.show()
```

```
+-----+-----+
|house|count|
+-----+-----+
| Blue|    4|
|  Red|    4|
+-----+-----+
```

Let's write solution in PySpark

But, Have you noticed this?



It is nothing but, Spark breaks our application into Jobs, Stages, and Tasks.

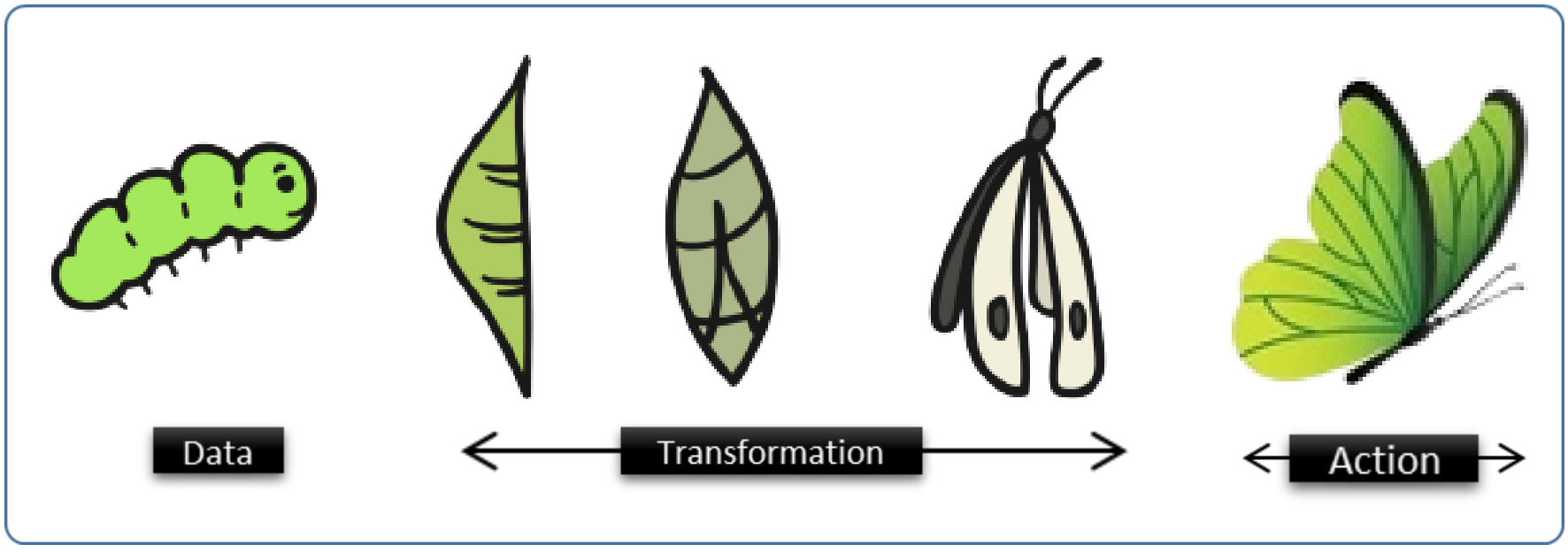
But, Why Spark breaks our application

into Jobs, Stages, & Task?

- Driver doesn't perform any data processing.
- Therefore, it breaks our application into jobs, stages, and tasks and assign them to Executors.
- Tasks enable distribution of work across multiple nodes.
- Stages optimize performance by executing sets of tasks in parallel.

Making into smaller chunks

Before that, We must aware of two things



1.Transformations: create a new RDD from an existing RDD.

- Narrow transformations: only depend on a single input partition, such as select, filter, drop, etc.
- Wide transformations: depend on multiple input partitions, such as groupBy, join, repartition, etc.

2. Actions: return a value to the driver program or write data to an external storage system.

- Example: show, collect, take, count, write, etc.

Transformations and Action!

How Jobs are created?

Each action and the above transformations packed together forms a Job

```
class_df = spark.read \
    .options(header=True, schema=class_schema) \
    .csv(file_path)
```

Job 0

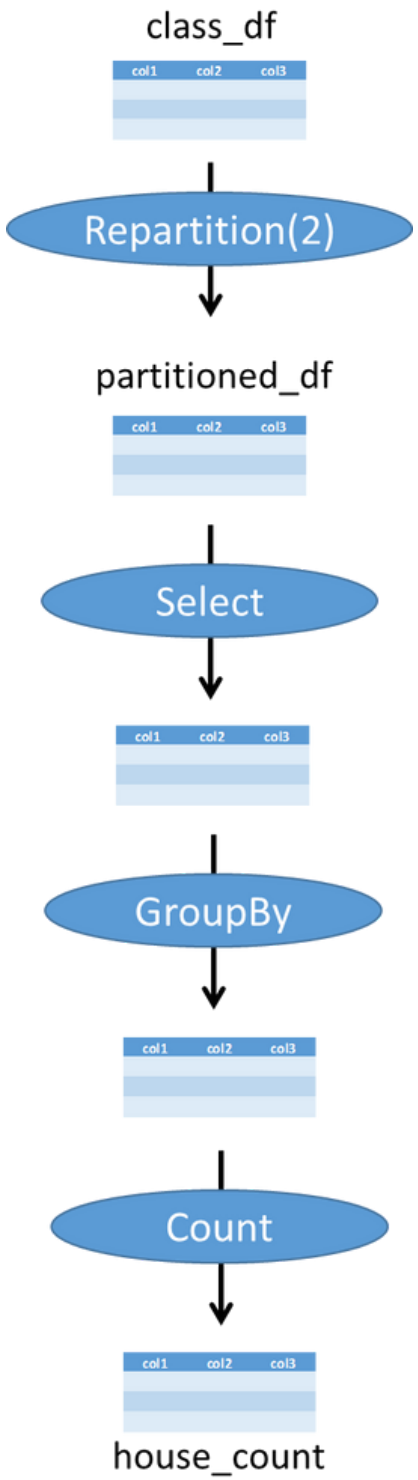
```
partitioned_df = class_df.repartition(2)

house_count = partitioned_df \
    .select("id", "name", "house") \
    .groupBy("house").count()

house_count.show()
```

Job 1

Spark Job is a single computation that gets instantiated to complete a Spark Action.

The Driver creates an unresolved Logical Plan for each Job. Let's understand this with the help of **Job 1**

```
partitioned_df = class_df.repartition(2)

house_count = partitioned_df \
    .select("id", "name", "house") \
    .groupBy("house") \
    .count()

house_count.show()
```

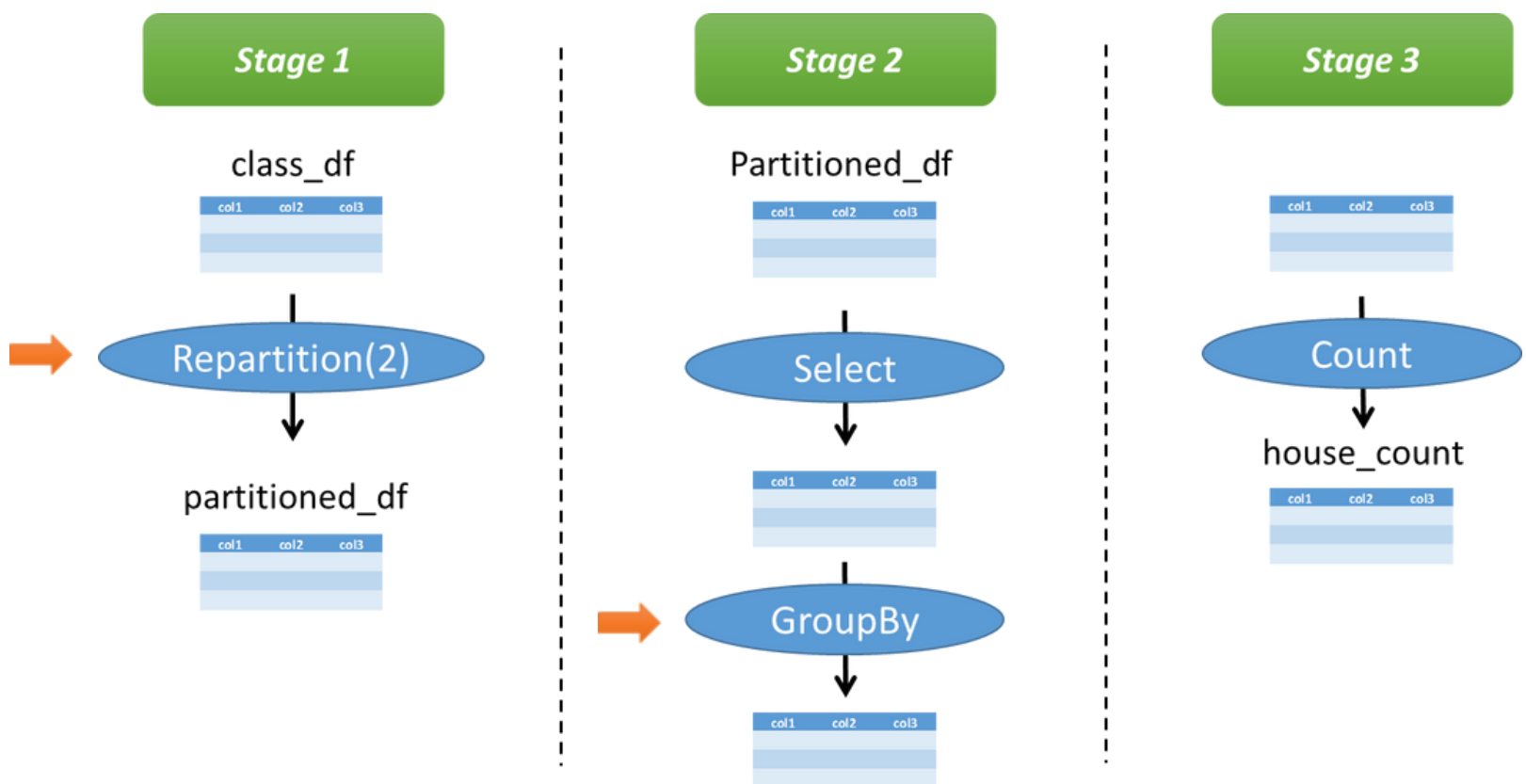


- These Unresolved Logical Plan will be converted into Physical Plan by Catalyst Optimizer
- Driver starts breaking this Physical Plan into Stages

Creation of Unresolved Logical Plan

How Stages and Tasks get created?

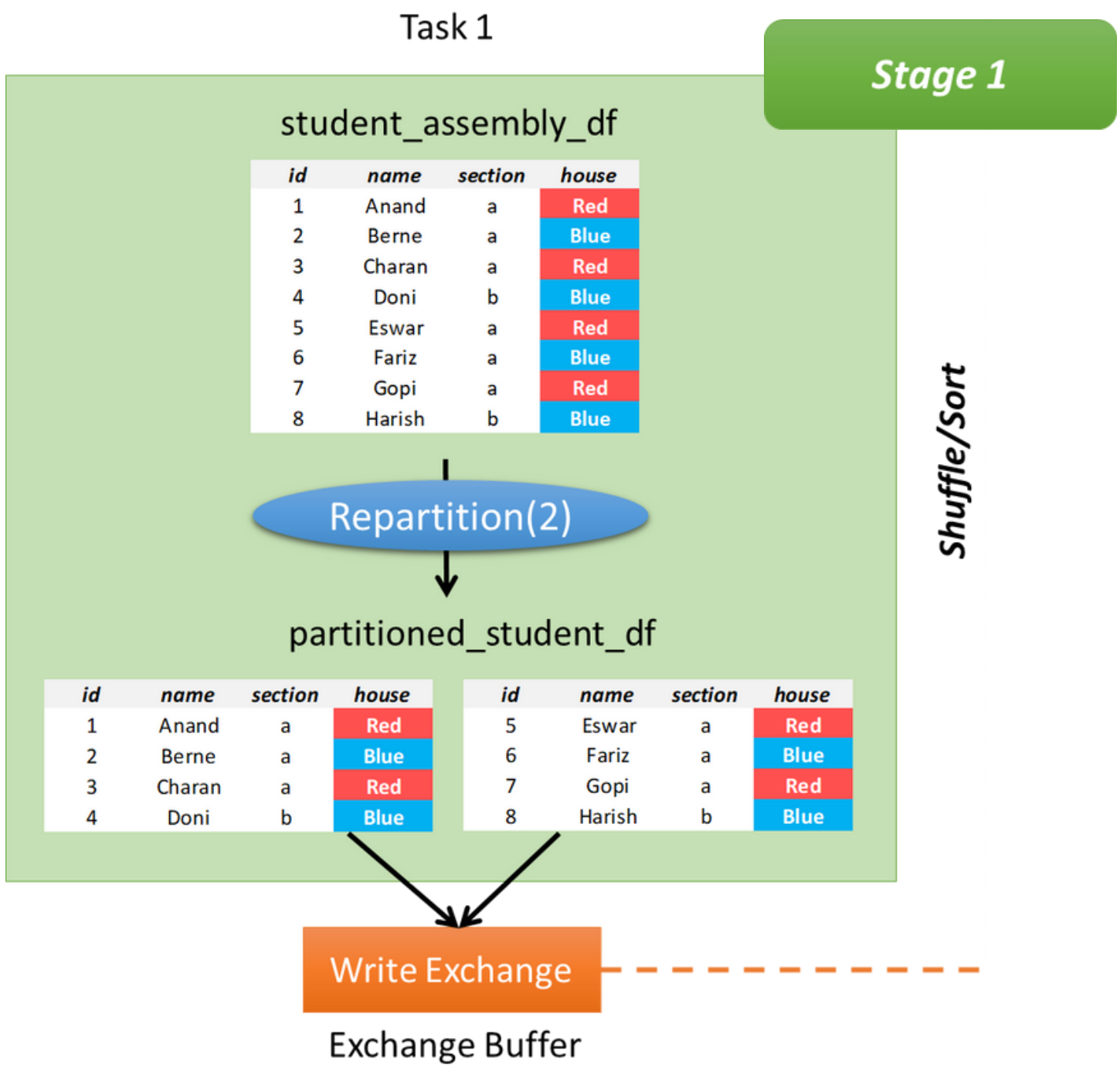
- The plan gets further divided into Stages and Tasks
- Each wide transformation and the above narrow transformation packed together forms a stage.
- Transformations specific to a partitions are known as Task.



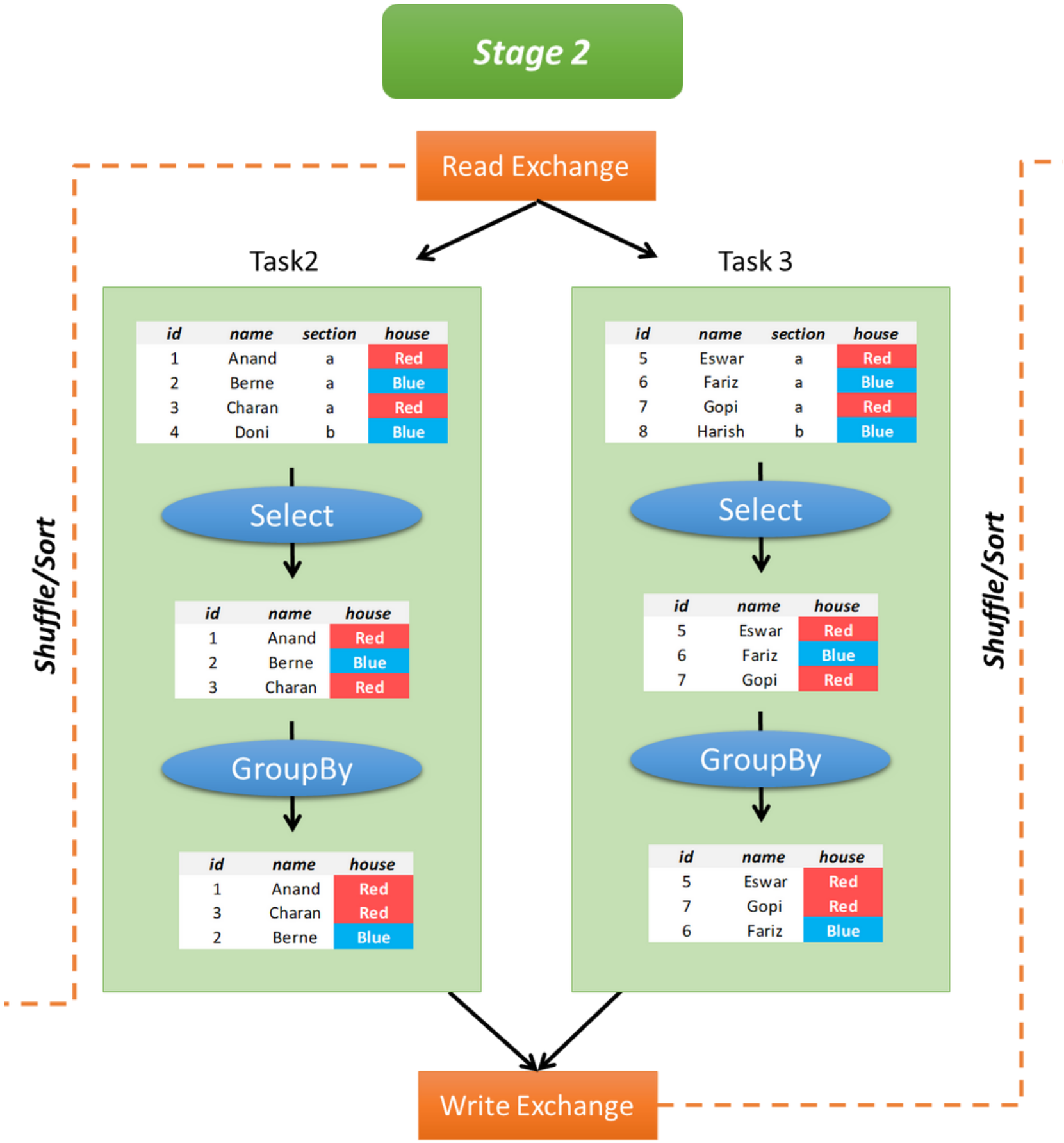
Stages further optimize performance by executing sets of tasks in parallel.

Let's see how

Each partitions and transformation works in each Stage

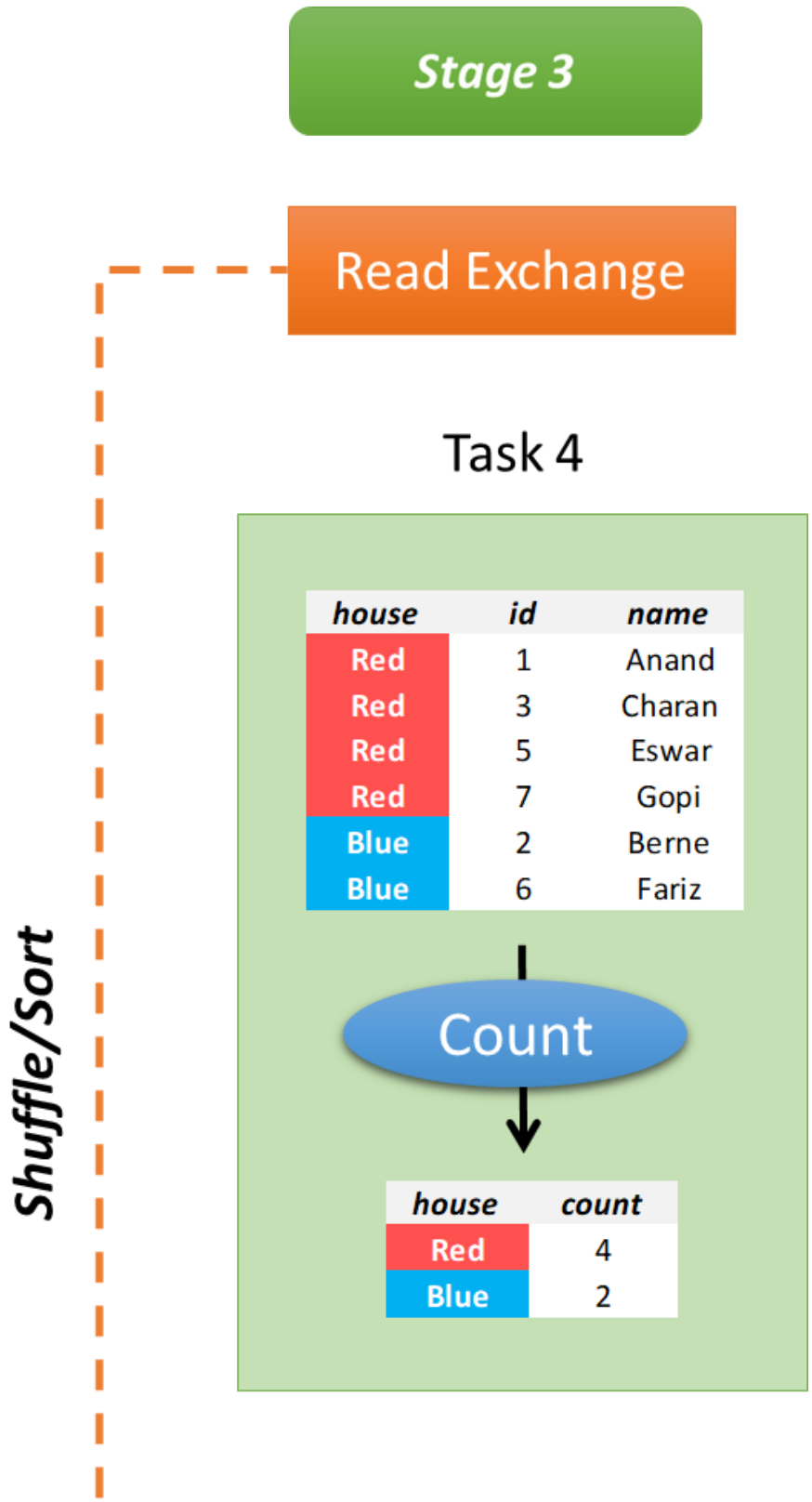


Note: Repartition(2) is a wide transformation and create 2 partitions Therefore the upcoming stage will have 2 Task.



Note: Transformation != Task

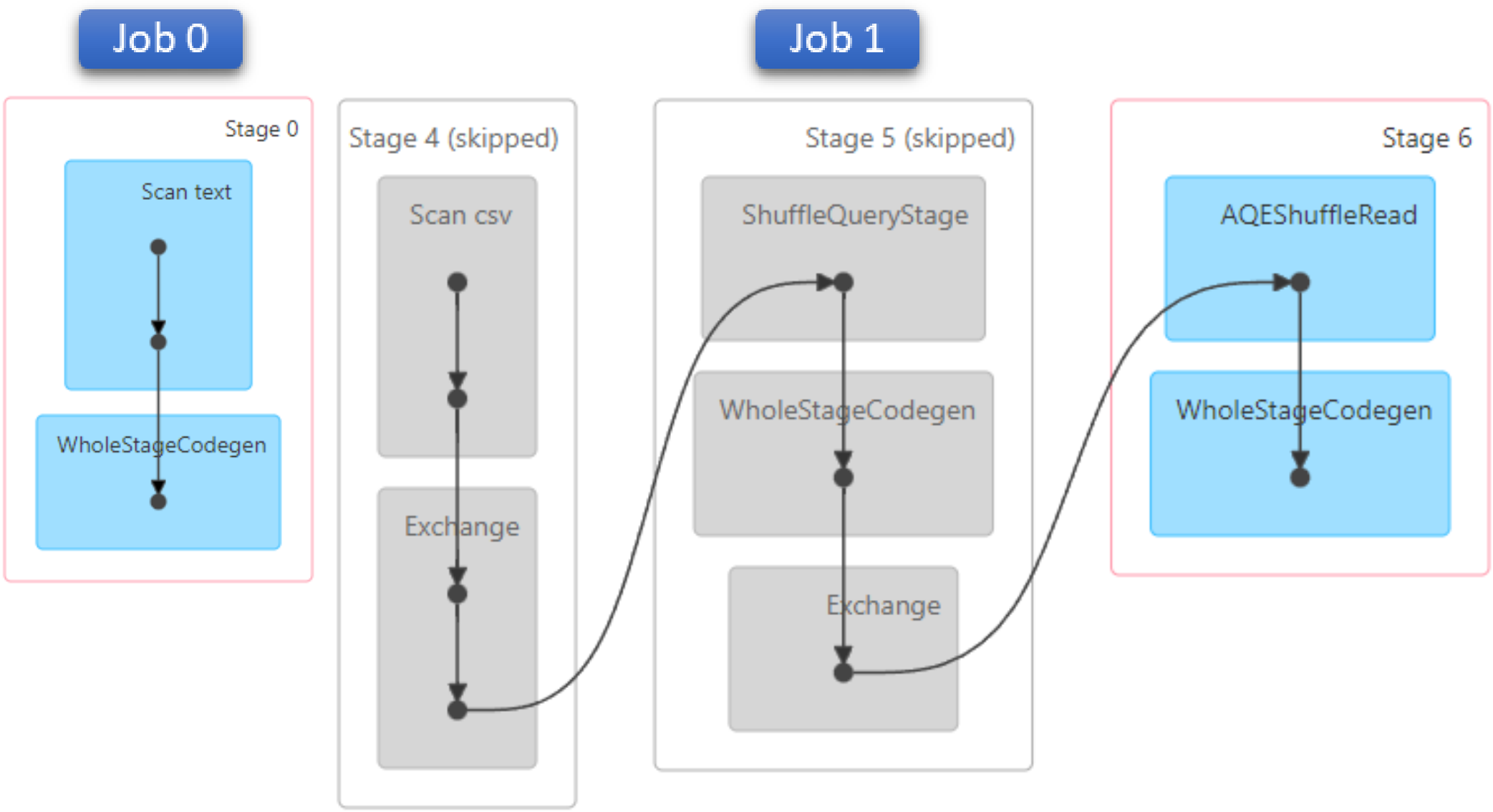
Transformations specific to a partitions are known as Task.



Note: The number of partitions after group/join depends on factors like data size, available resources, and partitioning scheme

The Complete DAG

- Job 0: 1 Stage
- Job 1 : 3 Stages (2 wide transformation -[SHUFFLING]-> 2 exchange)



Databricks typically runs each stage of a job as a separate job in order to optimize resource utilization and ensure fault tolerance.

<<Therefore we can see 4 Jobs>>

Have you enjoyed this overview of Spark Stages, and Tasks?

Arud Seka Berne S

Follow Me for more in-depth technical content on big data and related topics.