

JOBS, STAGES & TASKS

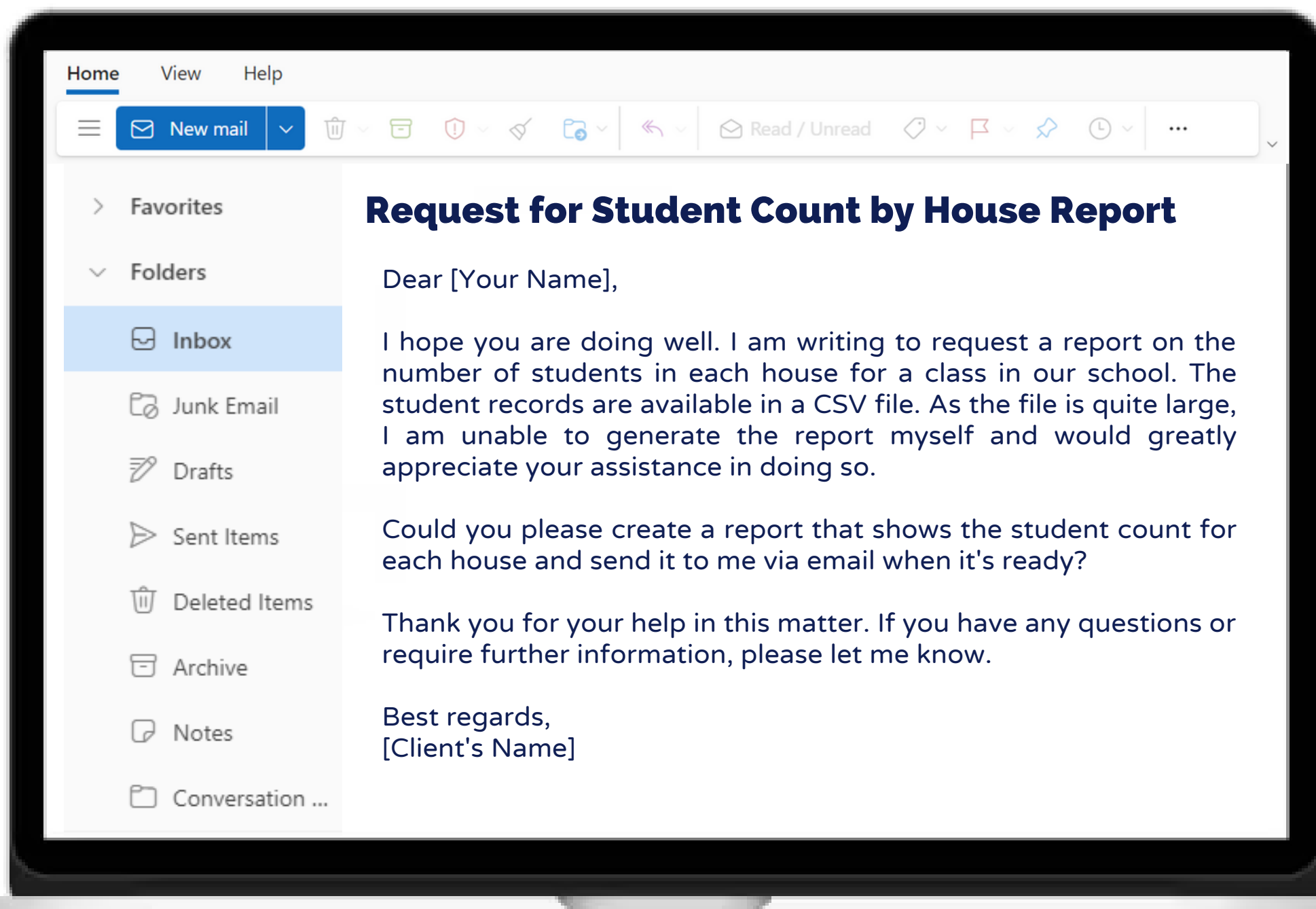
LETS BREAK IT DOWN

linkedin.com/in/arudsekaberne/

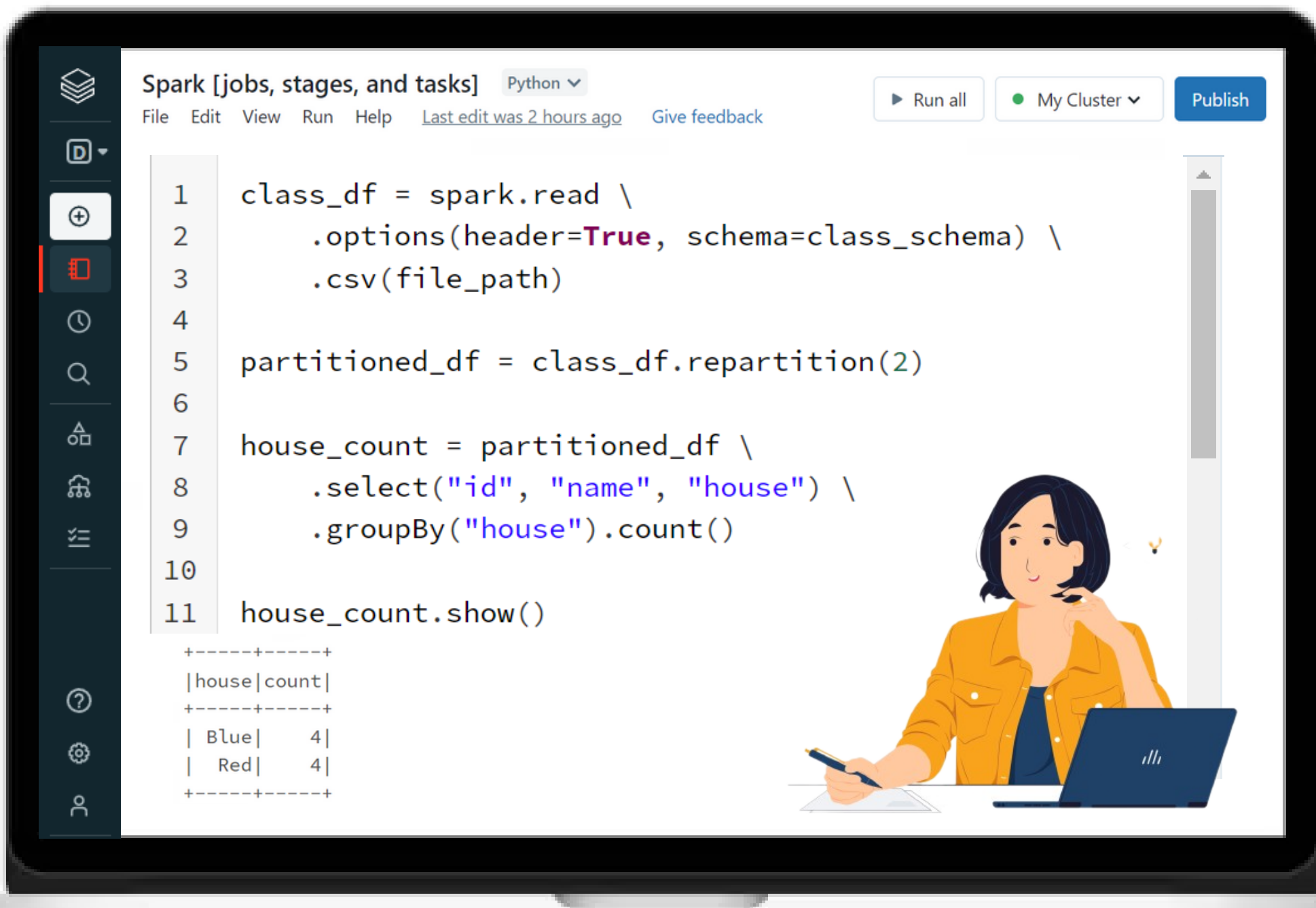


Oh! Looks like, I got an email notification
Let's have a look



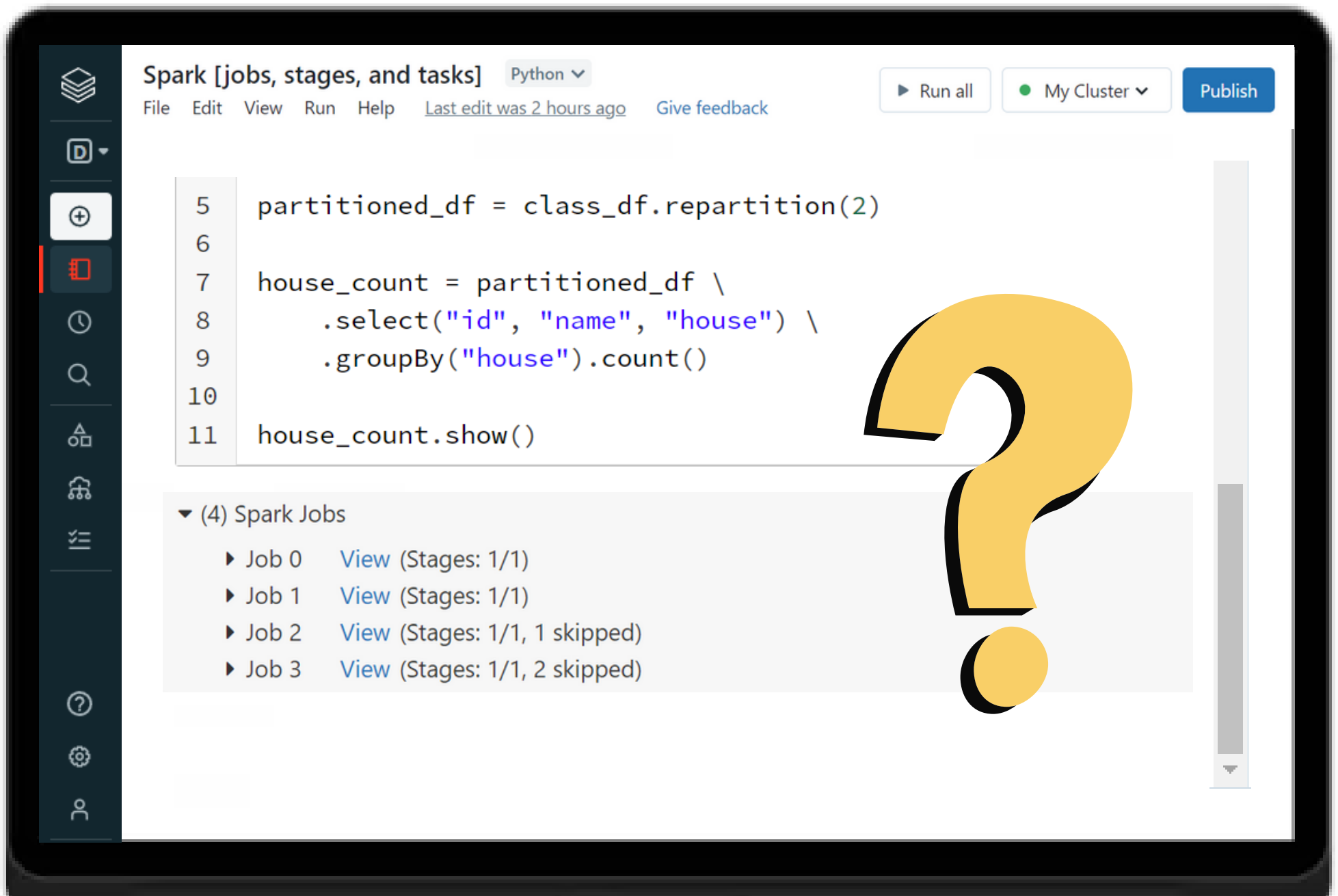


Analyzing Requirement



Let's write solution in PySpark

But, Have you noticed this?



The screenshot displays the Databricks Spark IDE interface. At the top, the title bar reads "Spark [jobs, stages, and tasks]" with a "Python" language selector. Below the title bar is a menu bar with "File", "Edit", "View", "Run", and "Help". To the right of the menu bar are buttons for "Run all", "My Cluster" (with a green dot and a dropdown arrow), and "Publish".

The main editor area contains a Python script with the following code:

```
5 partitioned_df = class_df.repartition(2)
6
7 house_count = partitioned_df \
8     .select("id", "name", "house") \
9     .groupBy("house").count()
10
11 house_count.show()
```

Below the code editor, the "Spark Jobs" section is expanded, showing a list of four jobs:

- ▶ Job 0 [View](#) (Stages: 1/1)
- ▶ Job 1 [View](#) (Stages: 1/1)
- ▶ Job 2 [View](#) (Stages: 1/1, 1 skipped)
- ▶ Job 3 [View](#) (Stages: 1/1, 2 skipped)

A large yellow question mark is overlaid on the right side of the interface, pointing towards the "Spark Jobs" section.

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

But, Why Spark breaks our application

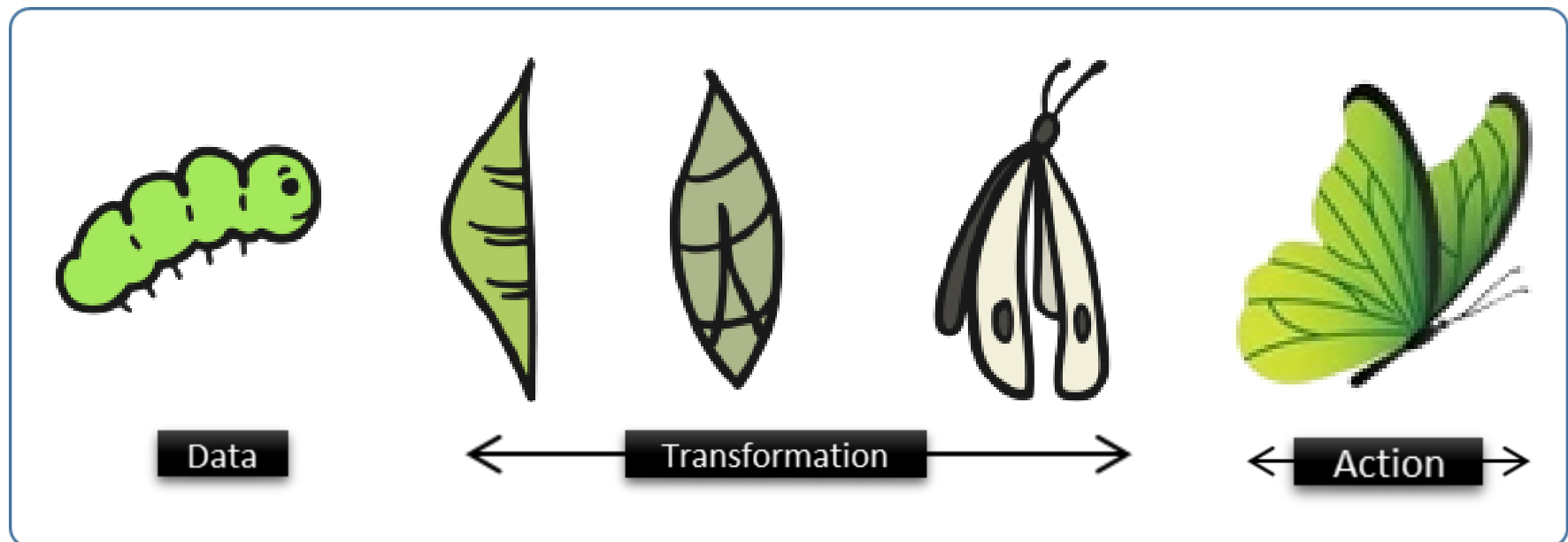
into Jobs, Stages, & Task?



- 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.

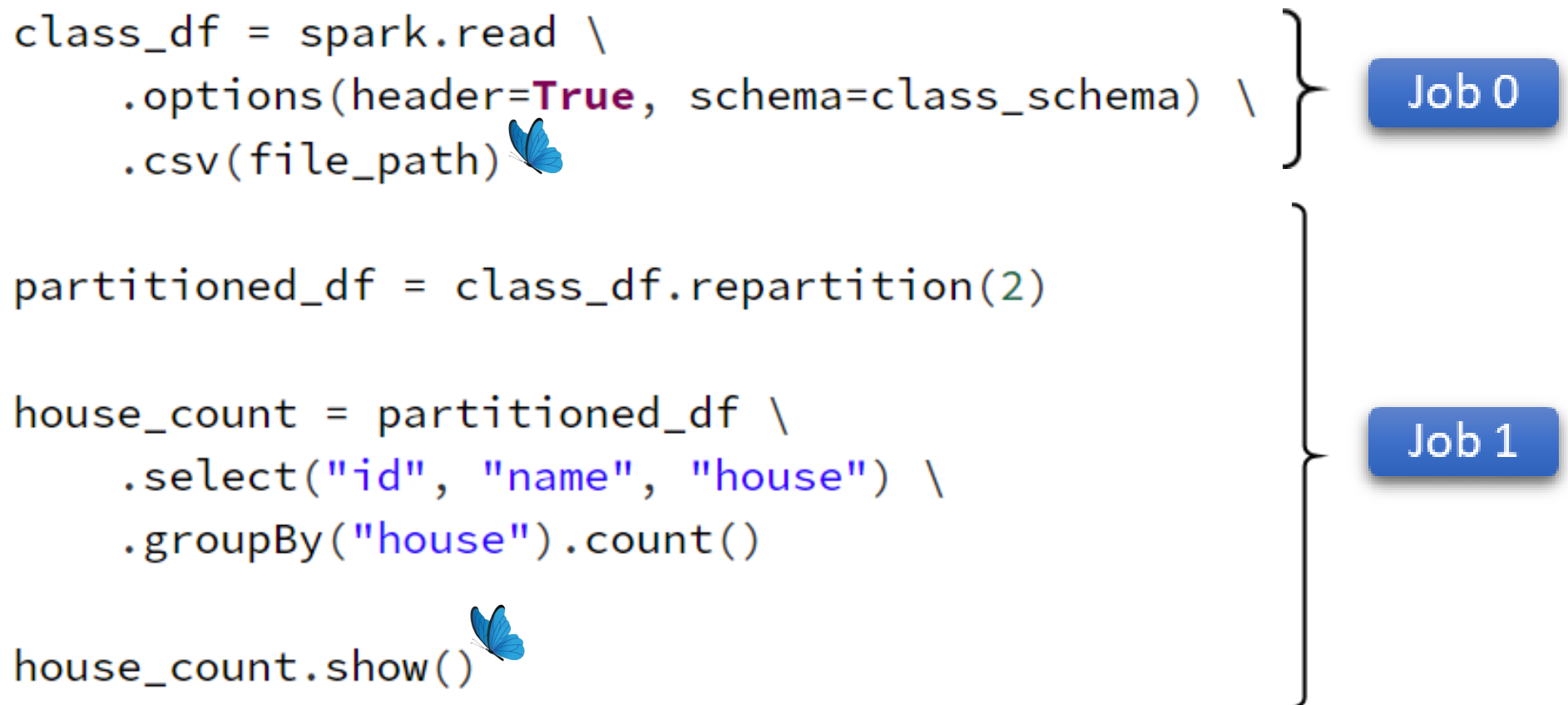
2. Actions: return a value to the driver program or write data to an external storage system.

- 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



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

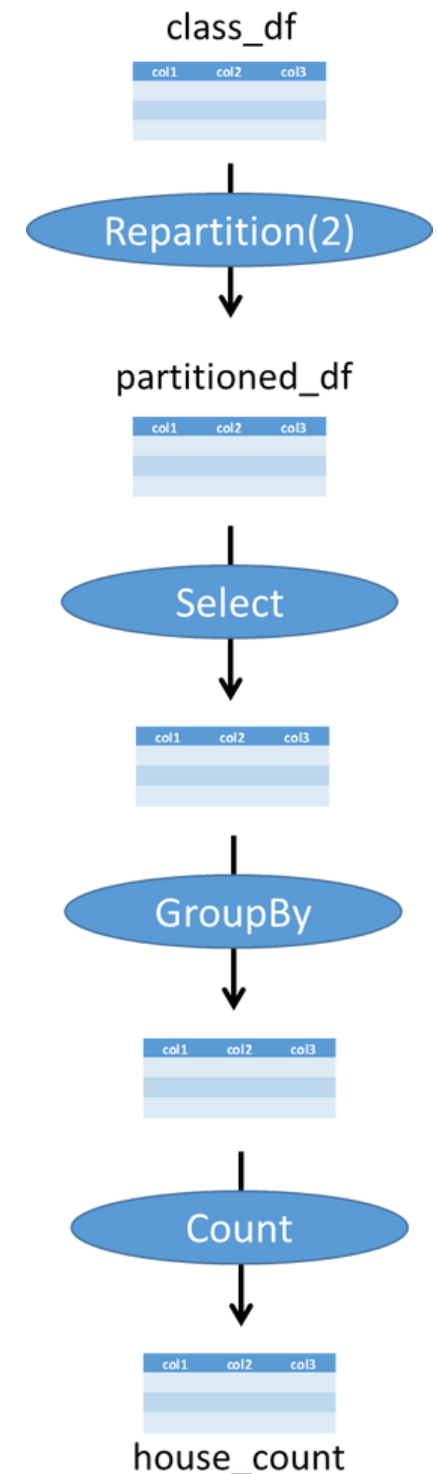
The Driver creates an unresolved Logical Plan for each Job. Let's understand this with the help of **Job 1**

```
partitioned_df = class_df.repartition(2)
```

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

```
house_count.show()
```

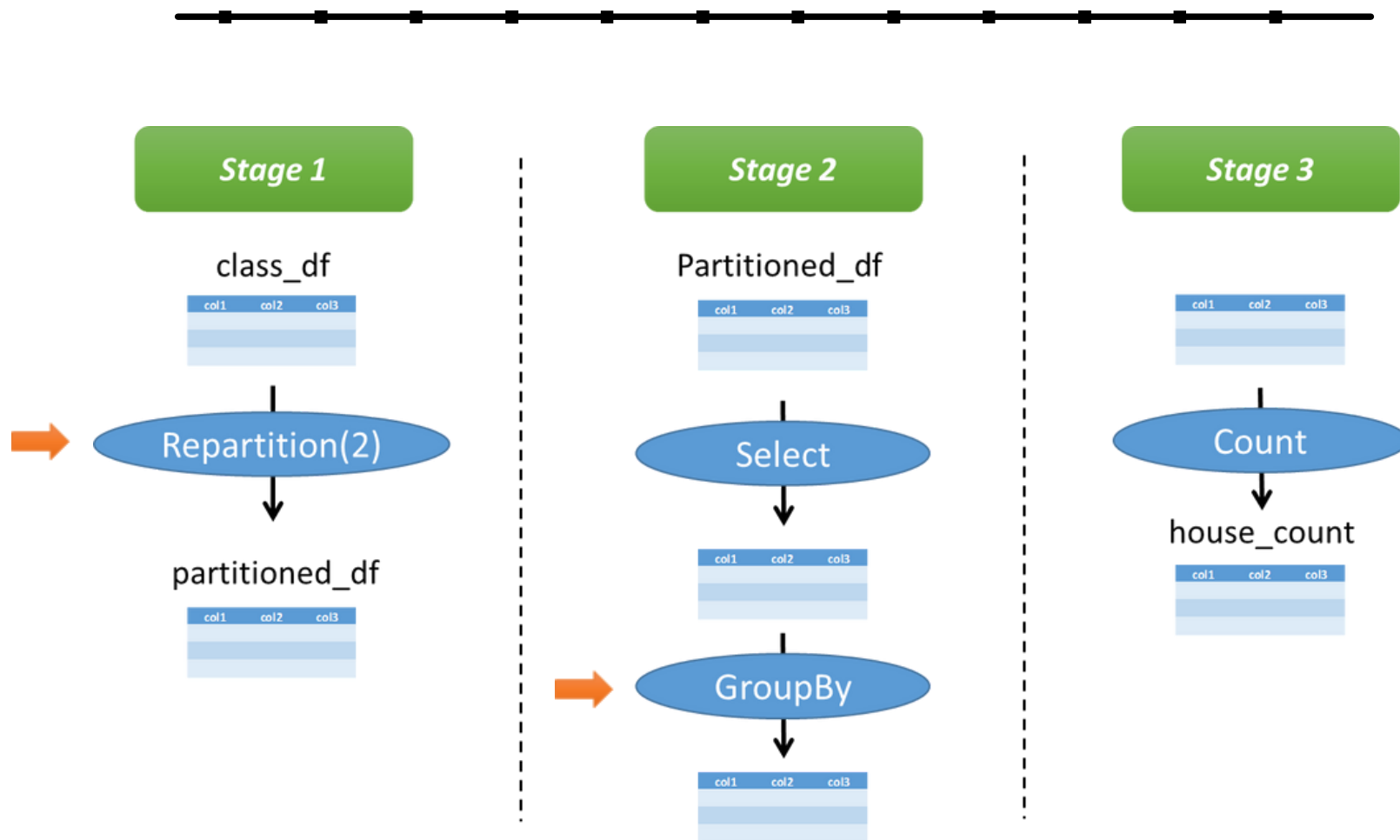
- These Unresolved Logical Plan will be converted into Physical Plan by Catalyst Optimizer
- Driver starts breaking this Physical Plan into Stages



Creation of Unresolved Logical Plan

How Stages and Tasks get created?

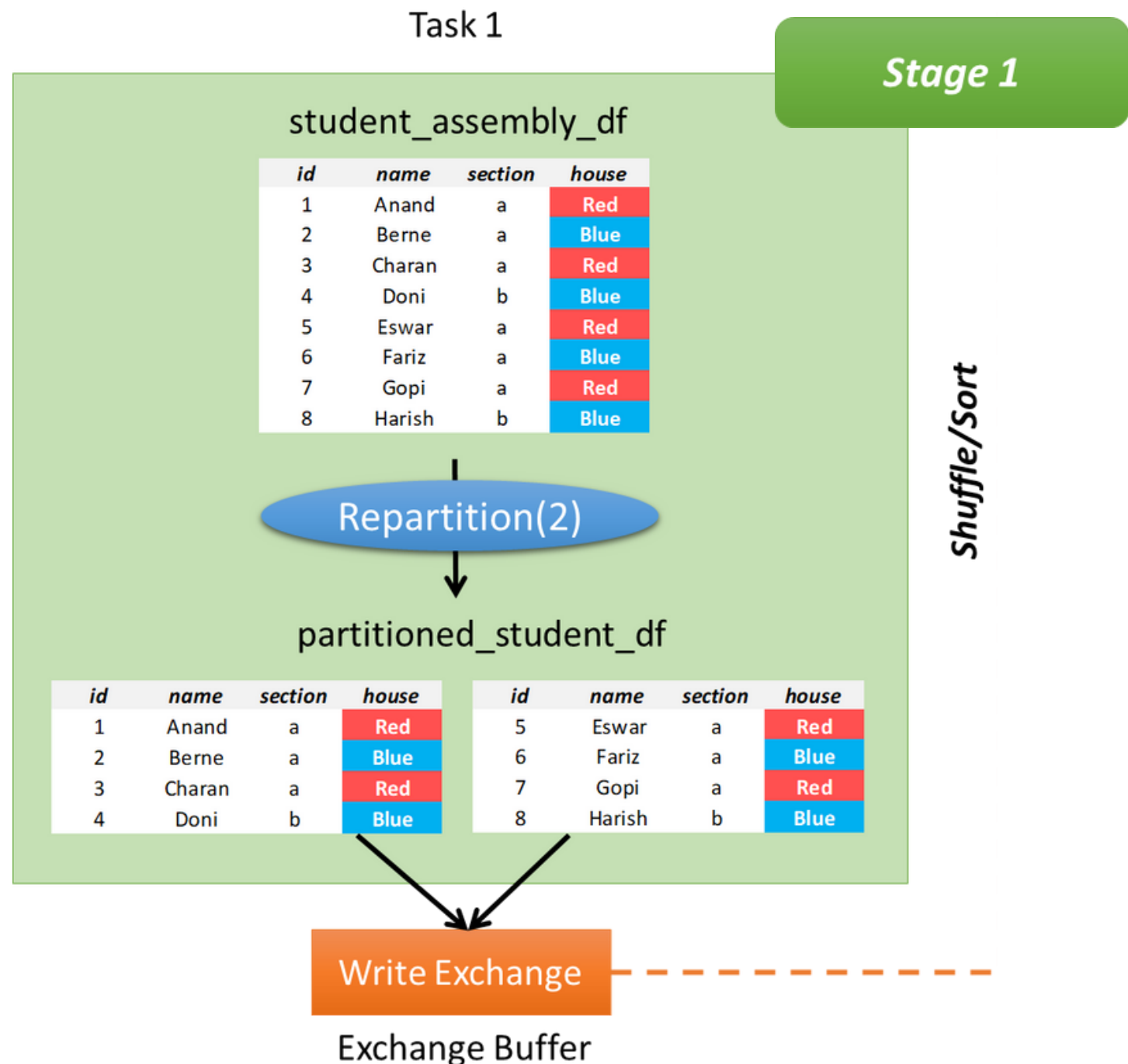
- The plan gets further divided into Stages and Tasks
- Each wide transformation and the above narrow transformation packed together forms a stage.
- Transformations specific to a partitions are known as Task.



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.

Stage 2

Read Exchange

Task2

Task 3

id	name	section	house
1	Anand	a	Red
2	Berne	a	Blue
3	Charan	a	Red
4	Doni	b	Blue

Select

id	name	house
1	Anand	Red
2	Berne	Blue
3	Charan	Red

GroupBy

id	name	house
1	Anand	Red
3	Charan	Red
2	Berne	Blue

id	name	section	house
5	Eswar	a	Red
6	Fariz	a	Blue
7	Gopi	a	Red
8	Harish	b	Blue

Select

id	name	house
5	Eswar	Red
6	Fariz	Blue
7	Gopi	Red

GroupBy

id	name	house
5	Eswar	Red
7	Gopi	Red
6	Fariz	Blue

Shuffle/Sort

Shuffle/Sort

Write Exchange

Note: Transformation != Task

Transformations specific to a partitions are known as Task.

Stage 3

Read Exchange

Task 4

Shuffle/Sort

<i>house</i>	<i>id</i>	<i>name</i>
Red	1	Anand
Red	3	Charan
Red	5	Eswar
Red	7	Gopi
Blue	2	Berne
Blue	6	Fariz

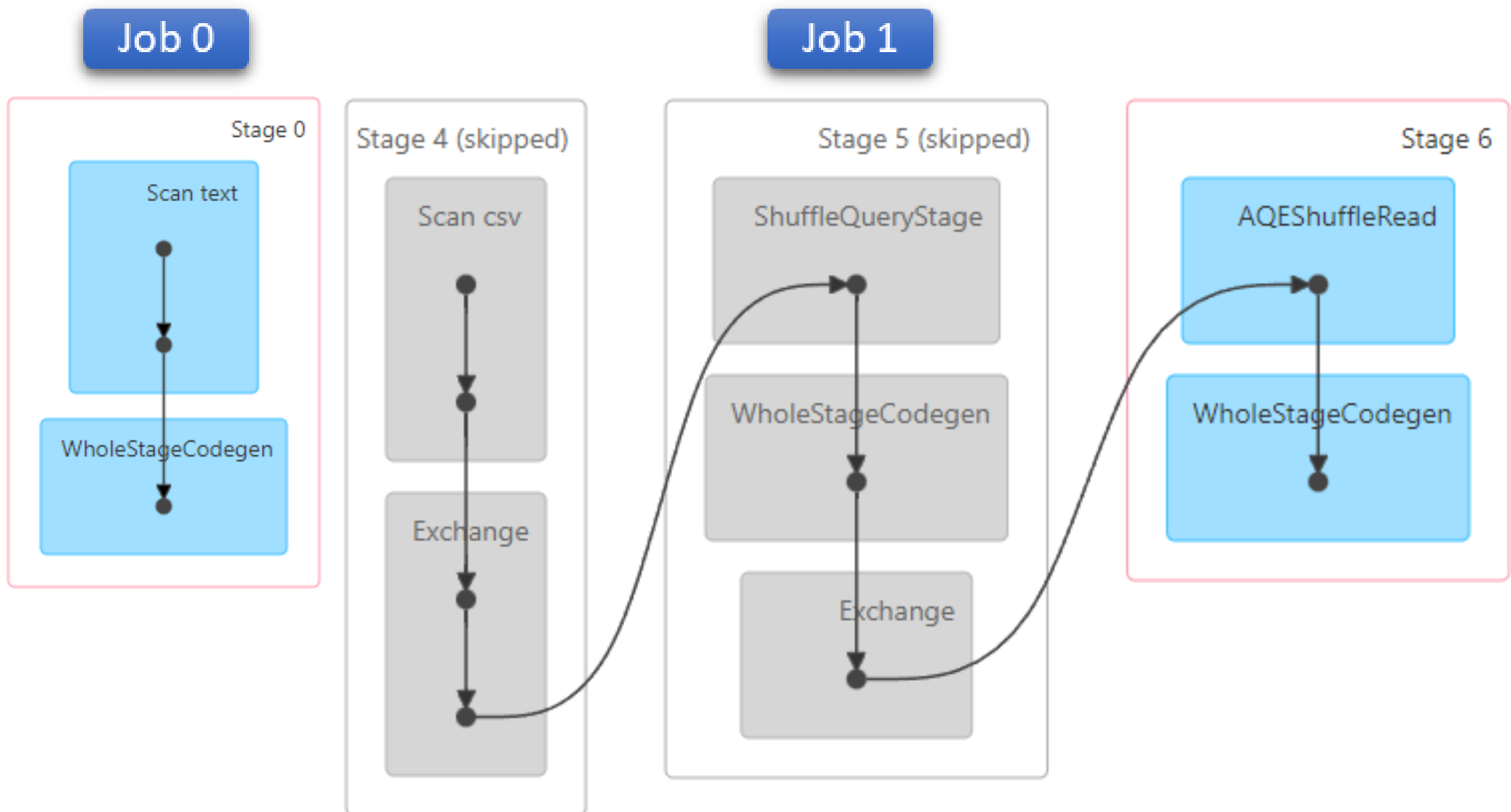
Count

<i>house</i>	<i>count</i>
Red	4
Blue	2

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.

