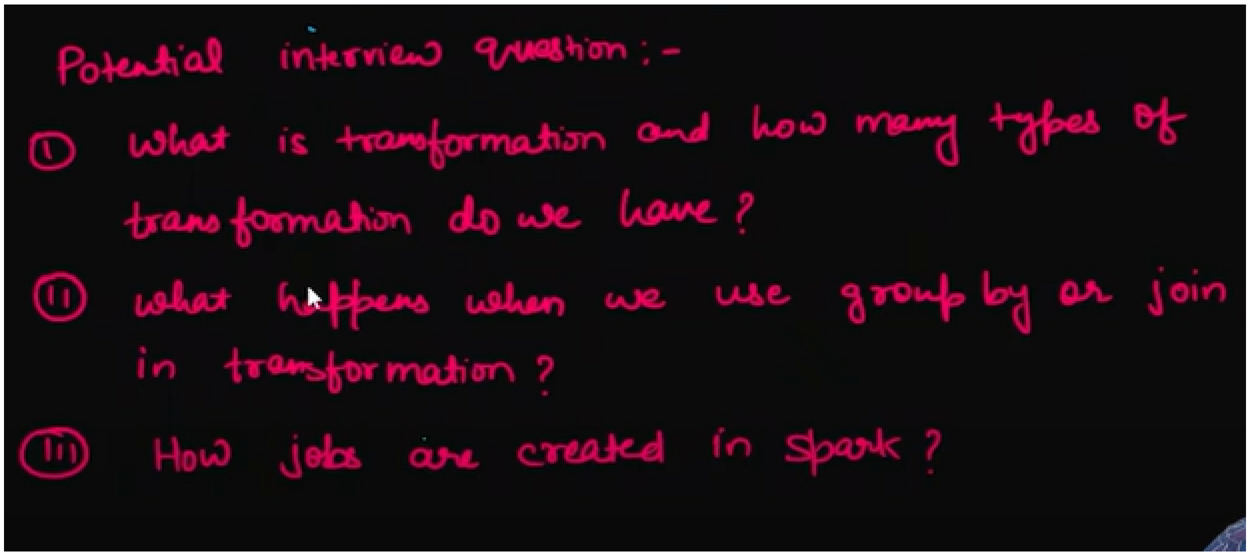
Transformation and action in Spark

1. Now we will see how our code runs inside the spark
2. Potential interview questions

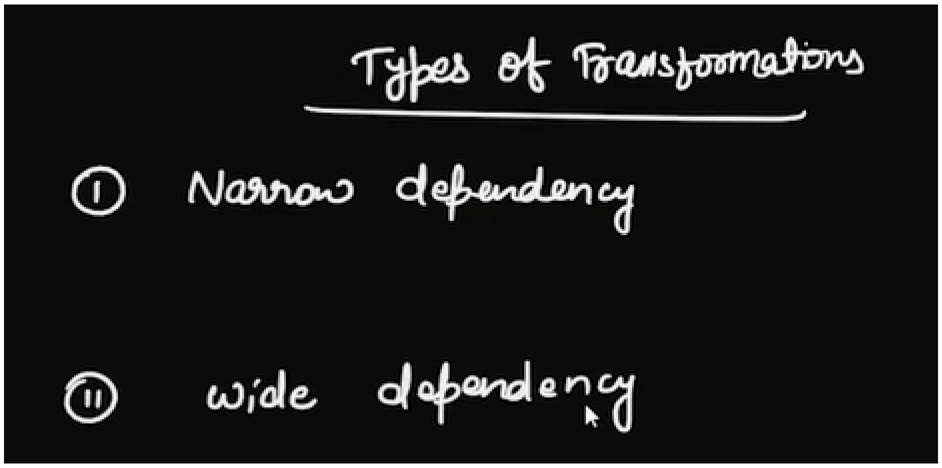


Potential interview question :-

(i) what is transformation and how many types of transformation do we have?

(ii) what happens when we use group by or join in transformation?

(iii) How jobs are created in spark?

3. If we are doing something or retrieving something from the data is called transformations
4. Action is something like ..after transformation we use .show() to see the table..they can be said as actions and .count() etc
5. Types of transformations
6. 

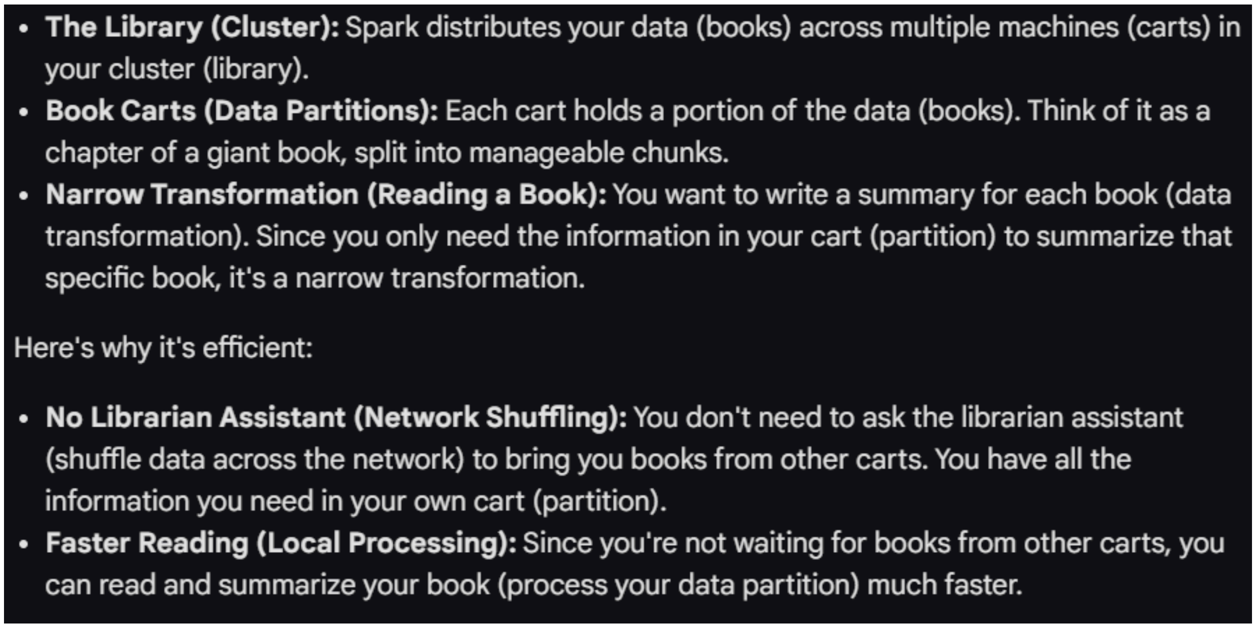
Types of Transformations

(i) Narrow dependency

(ii) wide dependency

7. Narrow : The transformed data partitions which does not depend upon other data partitions then it is called narrow dependency

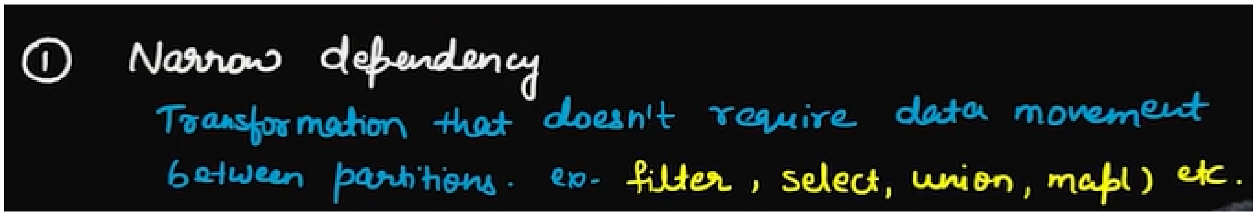
8. Example of narrow dependency



- **The Library (Cluster):** Spark distributes your data (books) across multiple machines (carts) in your cluster (library).
- **Book Carts (Data Partitions):** Each cart holds a portion of the data (books). Think of it as a chapter of a giant book, split into manageable chunks.
- **Narrow Transformation (Reading a Book):** You want to write a summary for each book (data transformation). Since you only need the information in your cart (partition) to summarize that specific book, it's a narrow transformation.

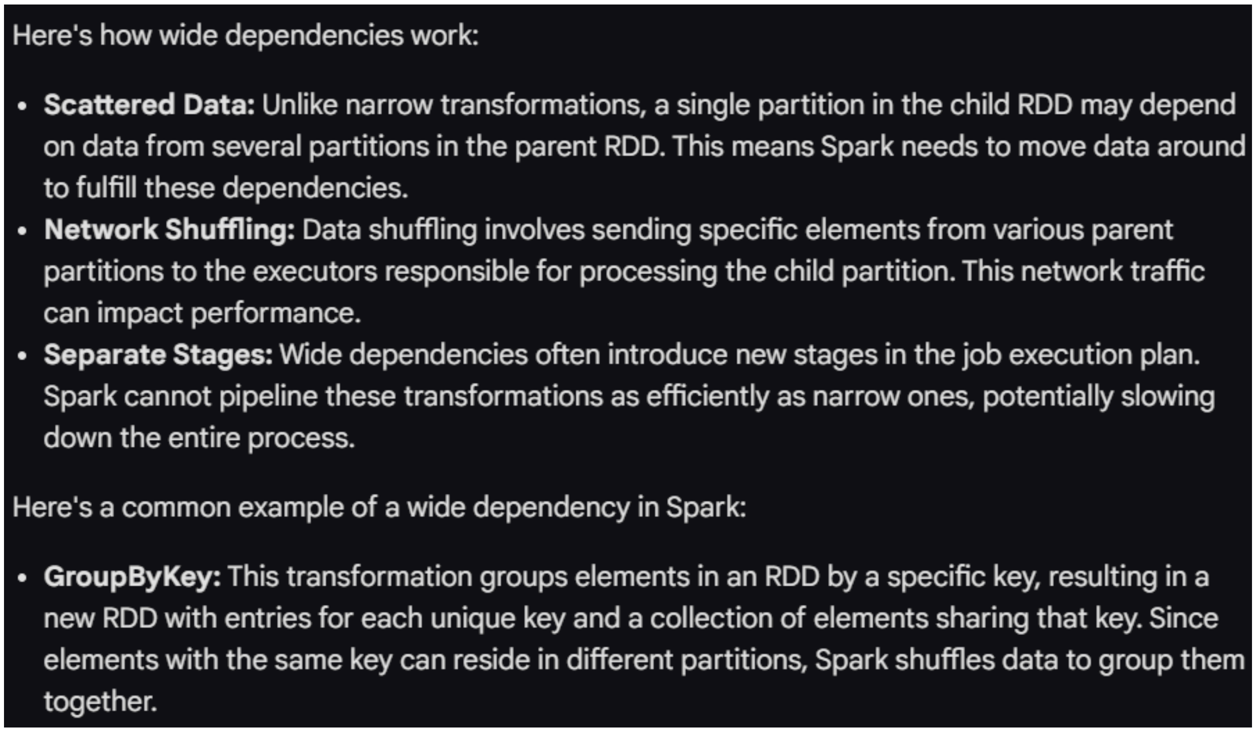
Here's why it's efficient:

- **No Librarian Assistant (Network Shuffling):** You don't need to ask the librarian assistant (shuffle data across the network) to bring you books from other carts. You have all the information you need in your own cart (partition).
- **Faster Reading (Local Processing):** Since you're not waiting for books from other carts, you can read and summarize your book (process your data partition) much faster.

9. 

① Narrow dependency
Transformation that doesn't require data movement between partitions. ex- filter, select, union, map() etc.

10. Wide dependency



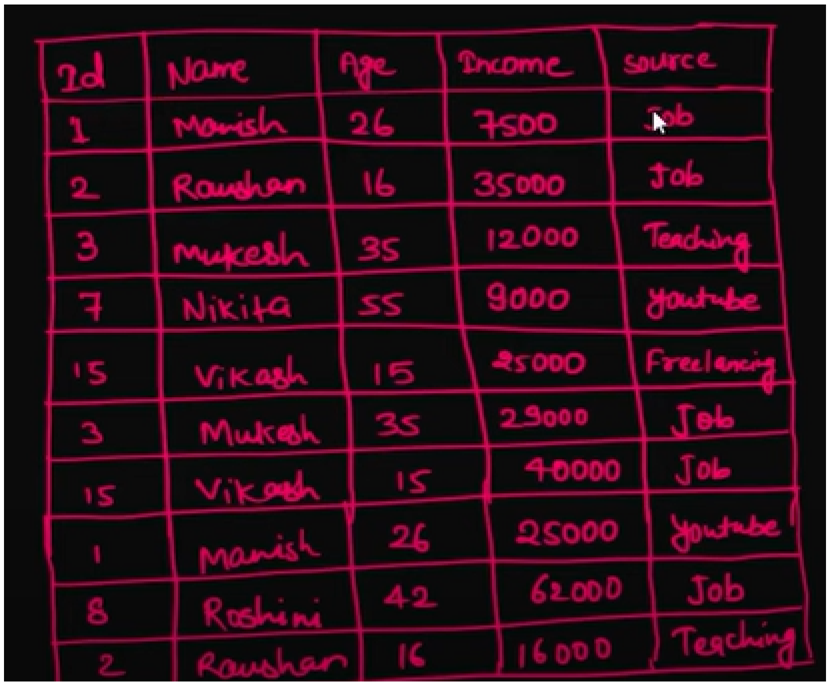
Here's how wide dependencies work:

- **Scattered Data:** Unlike narrow transformations, a single partition in the child RDD may depend on data from several partitions in the parent RDD. This means Spark needs to move data around to fulfill these dependencies.
- **Network Shuffling:** Data shuffling involves sending specific elements from various parent partitions to the executors responsible for processing the child partition. This network traffic can impact performance.
- **Separate Stages:** Wide dependencies often introduce new stages in the job execution plan. Spark cannot pipeline these transformations as efficiently as narrow ones, potentially slowing down the entire process.

Here's a common example of a wide dependency in Spark:

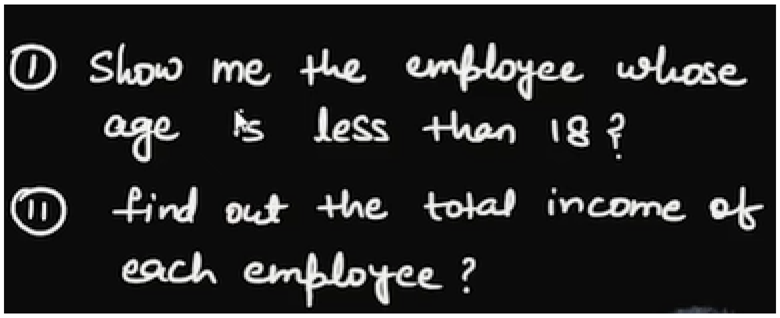
- **GroupByKey:** This transformation groups elements in an RDD by a specific key, resulting in a new RDD with entries for each unique key and a collection of elements sharing that key. Since elements with the same key can reside in different partitions, Spark shuffles data to group them together.

11. Lets take a sample csv data and understand this



Id	Name	Age	Income	source
1	Manish	26	7500	Job
2	Roushan	16	35000	Job
3	Mukesh	35	12000	Teaching
7	Nikita	55	9000	youtube
15	Vikash	15	25000	Freelancing
3	Mukesh	35	29000	Job
15	Vikash	15	40000	Job
1	Manish	26	25000	youtube
8	Roshini	42	62000	Job
2	Roushan	16	16000	Teaching

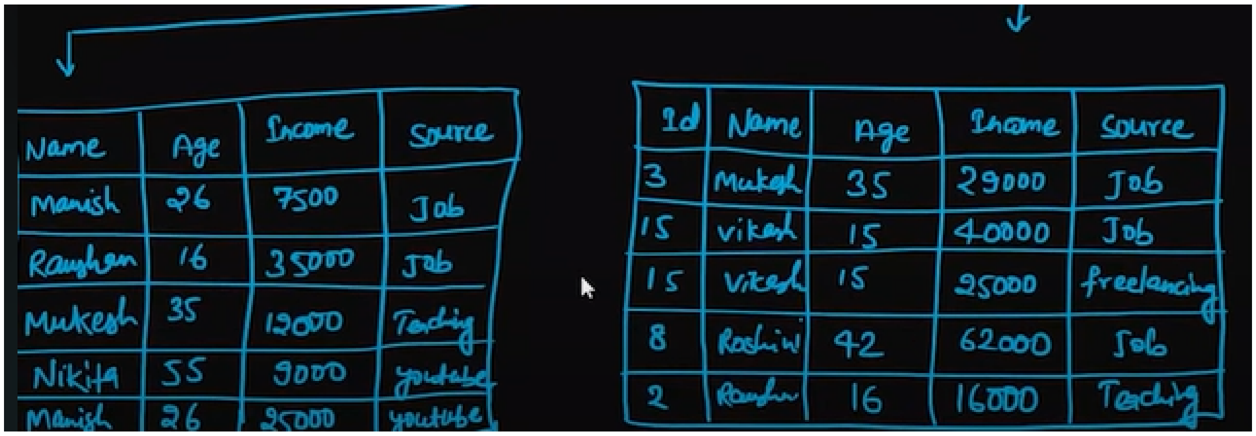
12. Now lets consider these two questions



(I) Show me the employee whose age is less than 18 ?

(II) find out the total income of each employee ?

13. Lets suppose our data has 2 partitions

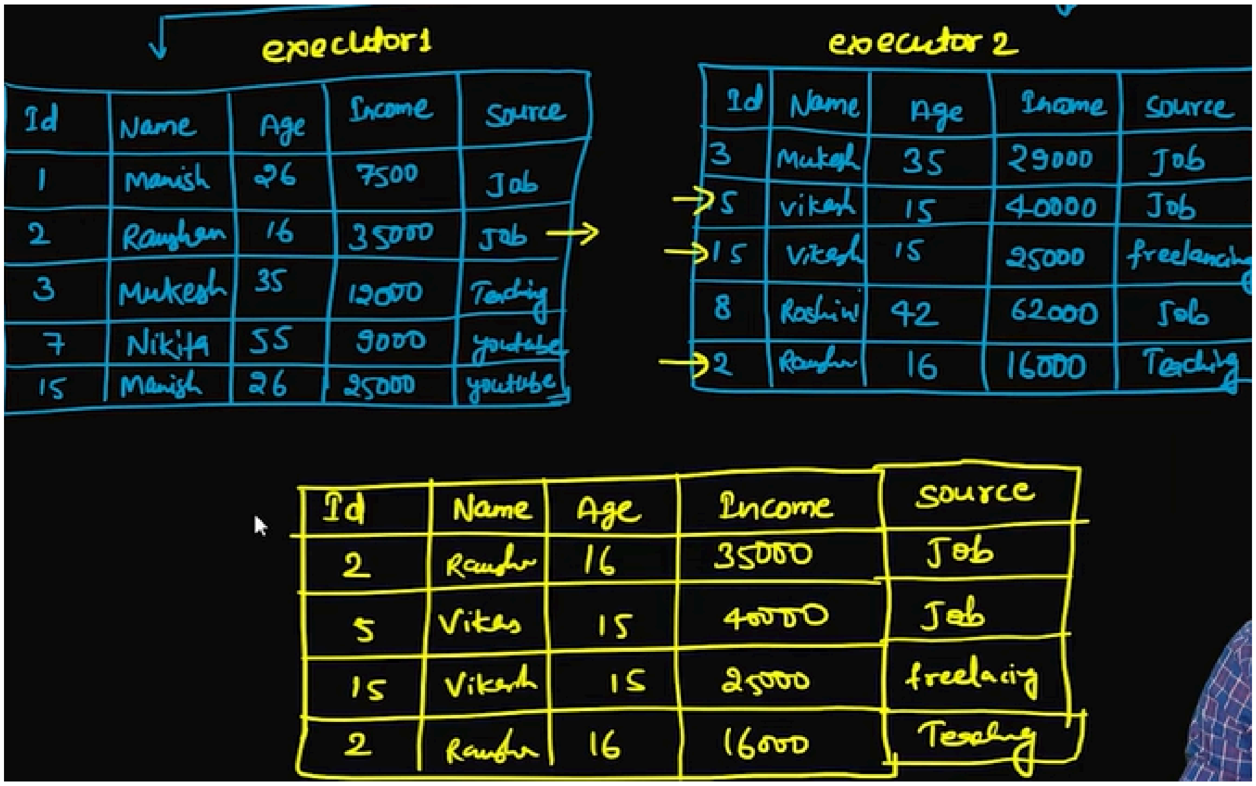


Name	Age	Income	Source
Manish	26	7500	Job
Raushan	16	35000	Job
Mukesh	35	12000	Teaching
Nikita	55	9000	youtube
Manish	26	25000	youtube

Id	Name	Age	Income	Source
3	Mukesh	35	29000	Job
15	vikash	15	40000	Job
15	vikash	15	25000	freelancing
8	Roshini	42	62000	Job
2	Raushan	16	16000	Teaching

14. Now each partition will be in a separate executors

15. And our first query will return this



executor1

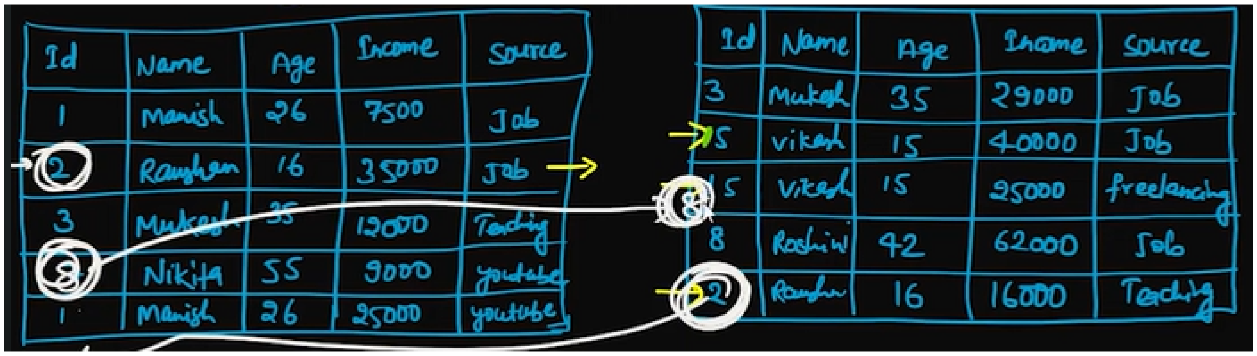
Id	Name	Age	Income	Source
1	Manish	26	7500	Job
2	Raushan	16	35000	Job
3	Mukesh	35	12000	Teaching
7	Nikita	55	9000	youtube
15	Manish	26	25000	youtube

executor 2

Id	Name	Age	Income	Source
3	Mukesh	35	29000	Job
5	vikesh	15	40000	Job
15	vikesh	15	25000	freelancing
8	Roshini	42	62000	Job
2	Raushan	16	16000	Teaching

Id	Name	Age	Income	Source
2	Raushan	16	35000	Job
5	Vikas	15	40000	Job
15	Vikesh	15	25000	freelancing
2	Raushan	16	16000	Teaching

16. Here our data was not moved
17. Now coming to 2nd ques
18. Here we are performing group by operation…so here the problem arises when same id are in two diff executor's
19. Now to calculate the group by…it shuffles the data



Id	Name	Age	Income	Source
1	Manish	26	7500	Job
2	Raushan	16	35000	Job
3	Mukesh	35	12000	Teaching
8	Nikita	55	9000	youtube
1	Manish	26	25000	youtube

Id	Name	Age	Income	Source
3	Mukesh	35	29000	Job
5	vikesh	15	40000	Job
8 5	vikesh	15	25000	freelancing
8	Roshini	42	62000	Job
2	Raushan	16	16000	Teaching

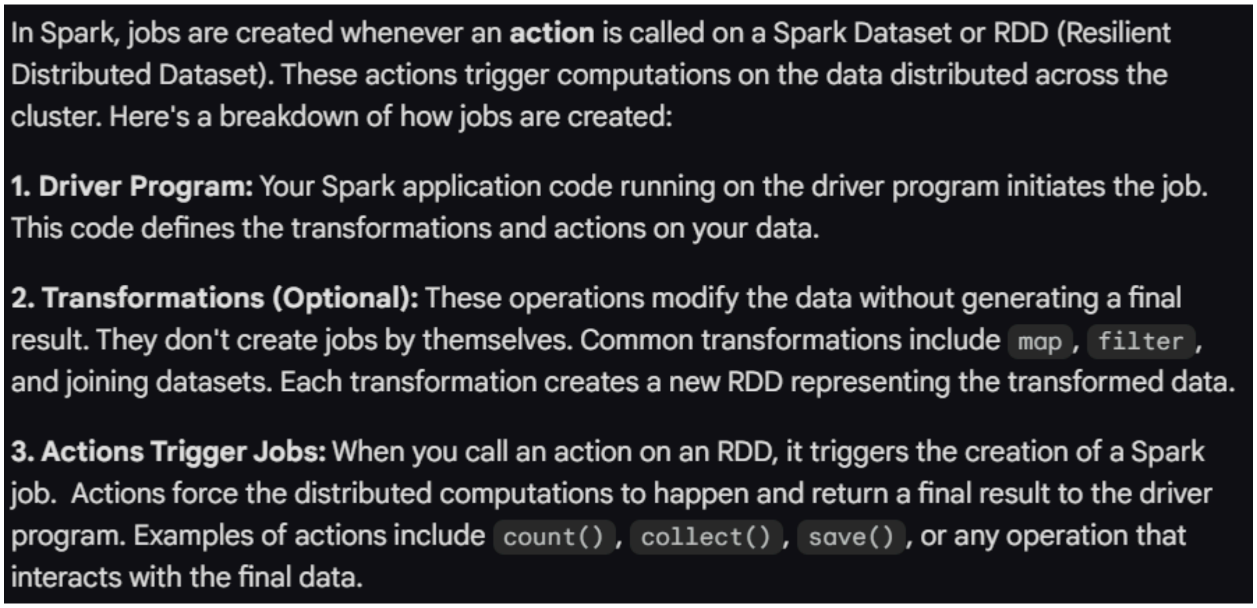
20. And this shuffling will be very expensive in the real time
21. Example of wide transformation

(ii) wide dependency

ex: Join, groupby, distinct

22. How jobs are created in spark?

23. So whenever we do any actions(count,show,collect) then the jobs will get created



In Spark, jobs are created whenever an **action** is called on a Spark Dataset or RDD (Resilient Distributed Dataset). These actions trigger computations on the data distributed across the cluster. Here's a breakdown of how jobs are created:

1. Driver Program: Your Spark application code running on the driver program initiates the job. This code defines the transformations and actions on your data.

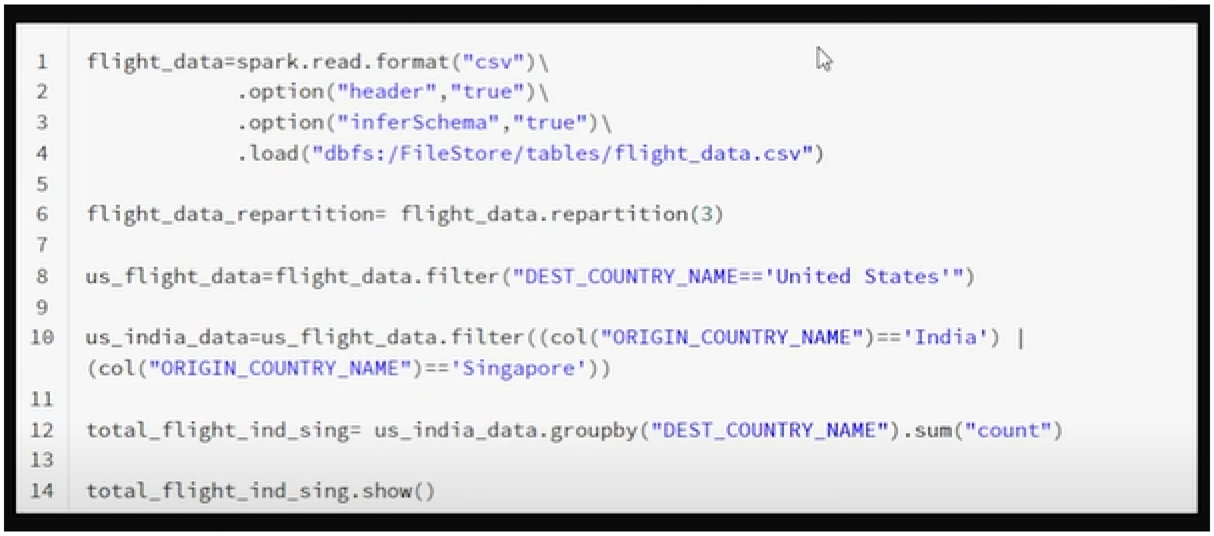
2. Transformations (Optional): These operations modify the data without generating a final result. They don't create jobs by themselves. Common transformations include `map`, `filter`, and joining datasets. Each transformation creates a new RDD representing the transformed data.

3. Actions Trigger Jobs: When you call an action on an RDD, it triggers the creation of a Spark job. Actions force the distributed computations to happen and return a final result to the driver program. Examples of actions include `count()`, `collect()`, `save()`, or any operation that interacts with the final data.

24. And removing wide dependencies is a type of optimization in spark

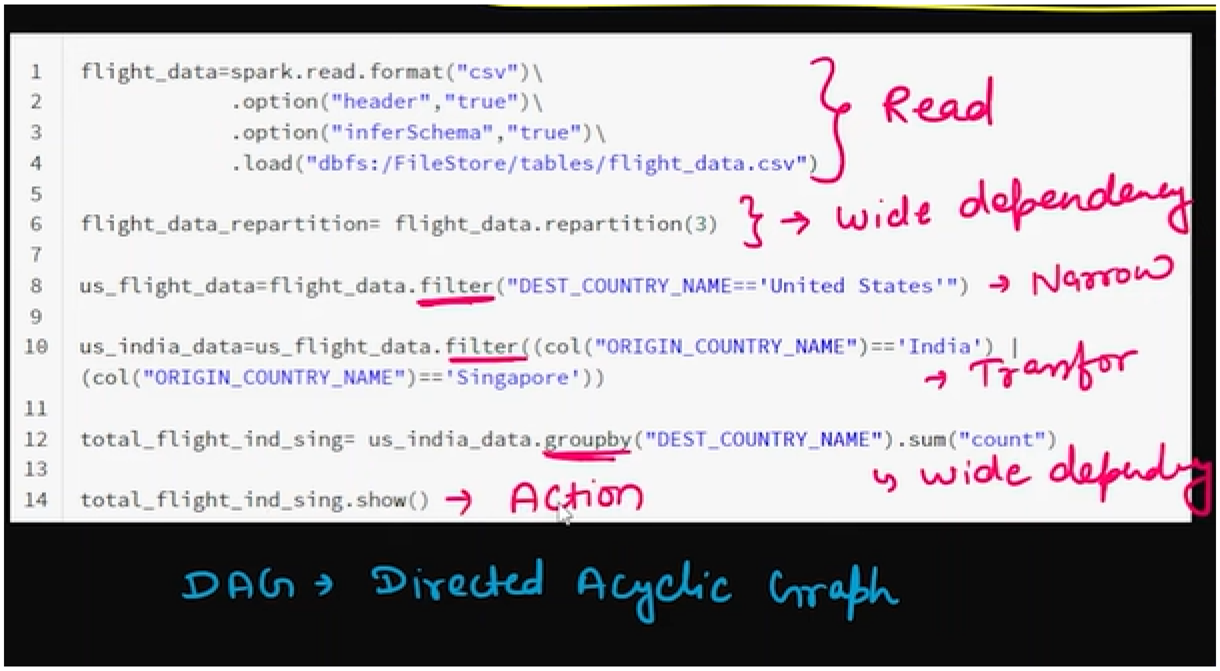
DAG and Lazy Evaluation in spark

1. Lets take a sample code snipped as shown below

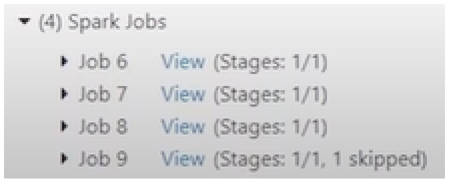


```
flight_data=spark.read.format("csv")\
              .option("header","true")\
              .option("inferSchema","true")\
              .load("dbfs:/FileStore/tables/flight_data.csv")

flight_data_repartition= flight_data.repartition(3)

us_flight_data=flight_data.filter("DEST_COUNTRY_NAME=='United States'")

us_india_data=us_flight_data.filter((col("ORIGIN_COUNTRY_NAME")=='India') |
(col("ORIGIN_COUNTRY_NAME")=='Singapore'))

total_flight_ind_sing= us_india_data.groupby("DEST_COUNTRY_NAME").sum("count")

total_flight_ind_sing.show()
```

2. Consider above code as a spark application and it also contains transformations and actions
3. DAG will create a graph for every job present in our code

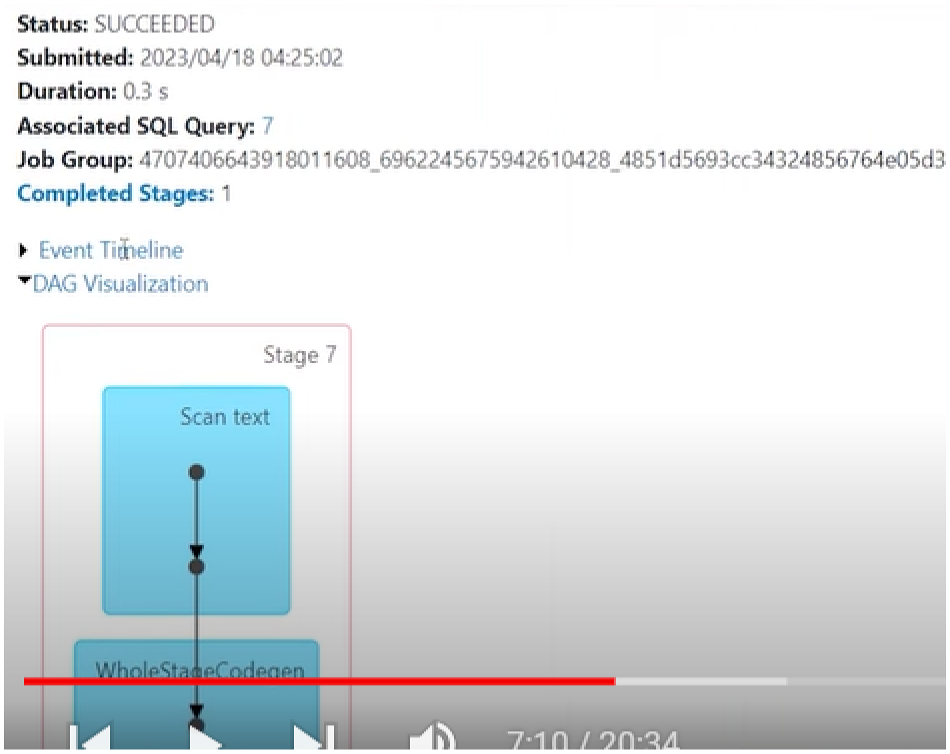
4. 

```
flight_data=spark.read.format("csv")\
            .option("header","true")\
            .option("inferSchema","true")\
            .load("dbfs:/FileStore/tables/flight_data.csv")

flight_data_repartition= flight_data.repartition(3)

us_flight_data=flight_data.filter("DEST_COUNTRY_NAME=='United States'")

us_india_data=us_flight_data.filter((col("ORIGIN_COUNTRY_NAME")=='India') |
(col("ORIGIN_COUNTRY_NAME")=='Singapore'))

total_flight_ind_sing= us_india_data.groupby("DEST_COUNTRY_NAME").sum("count")

total_flight_ind_sing.show()
```

5. Here read and inferschema is also a action
6. Now lets head to db(databricks) and implement it practically
7. Now if we run this code in db
8. It will create 4 jobs as there are 4 actions in our code 

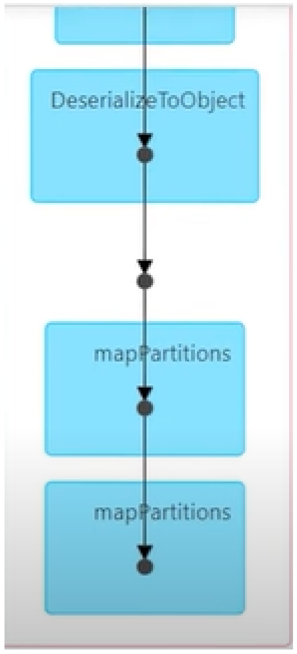
```
▾ (4) Spark Jobs
  ▸ Job 6   View (Stages: 1/1)
  ▸ Job 7   View (Stages: 1/1)
  ▸ Job 8   View (Stages: 1/1)
  ▸ Job 9   View (Stages: 1/1, 1 skipped)
```

9. Now if click on job we get



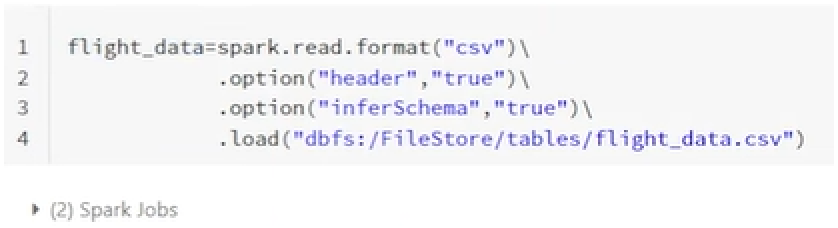
Status: SUCCEEDED
Submitted: 2023/04/18 04:25:02
Duration: 0.3 s
Associated SQL Query: 7
Job Group: 4707406643918011608_6962245675942610428_4851d5693cc34324856764e05d3
Completed Stages: 1

10. Here our first action is read and it is our first job…so it scanning the data and generated a java byte code
11. And our next action is inferschema…which read our data and gives the schema of our data



12. Now we have seen ..how DAG looks
13. Lets see lazy evaluation now

14. Here we will execute the read code



```
flight_data=spark.read.format("csv")\
            .option("header","true")\
            .option("inferSchema","true")\
            .load("dbfs:/FileStore/tables/flight_data.csv")
```

15. SO here it created two spark jobs(read ,inferschema)
16. Lazy evaluation : https://g.co/gemini/share/e12605ef68b9