**Understanding Process execution In Spark**

**Defining Transformations**:

* Transformation is an operation applied to an RDD or DataFrame that produces another RDD or DataFrame.
* Spark adds all the transformation steps to the DAG.
* Eg. map(), filter().

**Building the DAG:**

* DAG is Directed Acyclic Graph.
* Spark constructs a DAG representing the sequence of transformations.

**Triggering Execution with an Action:**

* Actions are operations that trigger the execution of the transformations and produce a result.
* Transformations are executed according to the DAG.
* Eg. collect(), count()

**Example code :**

order\_items = spark.read.csv("order\_items.csv",header = True, inferSchema = True)

order\_items.show(5)

**Transformation**: spark.read.csv("order\_items.csv", header=True, inferSchema=True)

* This transformation defines the steps to read the CSV file and infer the schema. It adds a step to the DAG but does not execute immediately.

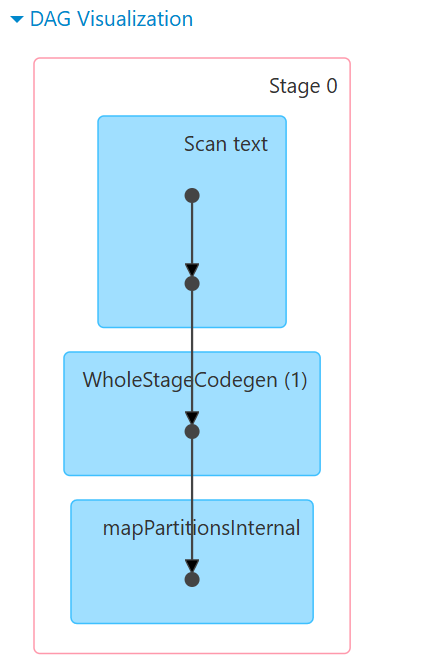
**DAG**:

* When this transformation is defined, Spark adds a step to the DAG to read the CSV file. No data is read at this point; only the plan for reading the data is created.

**Action**: order\_items.show(5)

* This action triggers the execution of the DAG. Spark reads the CSV file, processes the data according to the transformations, and displays the first 5 rows.

**The DAG for this process:** from Spark UI at localhost:4040



**1.Scan text**

* FileScanRDD [0] Reads raw data from files.
* MapPartitionsRDD [1] Creates partitions of the read data.

**2. WholeStageCodegen (1)**

* WholeStageCodegen Optimizes query execution by generating efficient Java code.

**3. mapPartitionsInternal**

* Processes the partitions by applying functions to each one.

**order\_items.show(5)** triggers the execution of these stages.

**Query Execution Plan:**

**order\_items.explain(True)**

When we run order\_items.explain(True), Spark provides detailed information about how it plans to execute the operations defined on the order\_items DataFrame.

1. == Parsed Logical Plan ==  
Relation [order\_id#98,amount#99,profit#100,quantity#101,product\_id#102] csv

* This is the initial representation of the DataFrame as understood by Spark. It shows the schema inferred from the CSV file.
* The Relation node indicates that the data source is a CSV file and lists the columns (order\_id, amount, profit, quantity, product\_id) with unique identifiers assigned to each column (e.g., order\_id#98).

2. == Analyzed Logical Plan ==  
order\_id: string, amount: double, profit: double, quantity: int, product\_id: int  
Relation [order\_id#98,amount#99,profit#100,quantity#101,product\_id#102] csv

* The analyzed logical plan includes the data types for each column after inferring the schema (header=True and inferSchema=True).
* order\_id is inferred as string, amount as double, profit as double, quantity as int, and product\_id as int.

3. == Optimized Logical Plan ==  
Relation [order\_id#98,amount#99,profit#100,quantity#101,product\_id#102] csv

* The optimized logical plan is usually where Spark applies various optimizations to improve the efficiency of query execution. However, since this example only involves reading a CSV file with no additional transformations or filters, the optimized plan looks the same as the analyzed logical plan.

4. == Physical Plan ==  
FileScan csv [order\_id#98,amount#99,profit#100,quantity#101,product\_id#102] Batched: false, DataFilters: [], Format: CSV, Location: InMemoryFileIndex(1 paths)[file:/C:/Users/ShrutiSharma/Downloads/Dataset for PySpark Assignments/..., PartitionFilters: [], PushedFilters: [], ReadSchema: struct<order\_id:string,amount:double,profit:double,quantity:int,product\_id:int>  
  
The physical plan details the exact steps Spark will take to execute the query, including information on file reading and data retrieval.

* FileScan csv indicates that Spark will perform a file scan to read the CSV file.
* It lists the columns to be read: [order\_id#98,amount#99,profit#100,quantity#101,product\_id#102].
* Batched: false means that batch reading is not being used.
* DataFilters: [] shows that no data filters are applied.
* Format: CSV confirms the file format.
* Location provides the path to the file being read.
* PartitionFilters: [] indicates no partition filters are applied.
* PushedFilters: [] shows that no filters are pushed down to the data source.
* ReadSchema reaffirms the schema of the DataFrame with detailed data types.