Master Pyspark Zero to Hero:

Unpivot in PySpark

The **unpivot** operation (also called **melting**) is used to transform a **wide-format table** into a **long-format table**. This means columns are turned into rows, effectively reversing the pivot operation. PySpark doesn't have a direct unpivot function like Pandas' melt, but you can achieve it using the **selectExpr** method or a combination of **stack** and other DataFrame transformations.

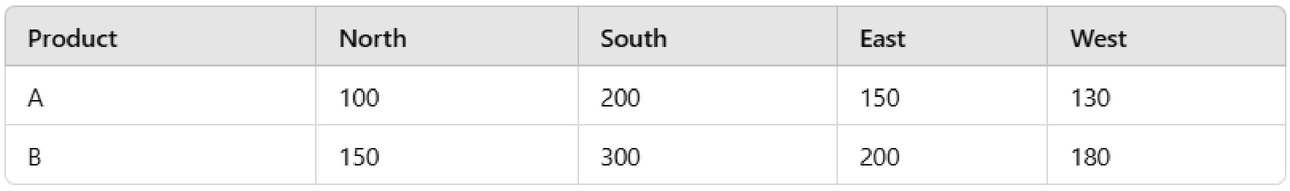
Key Concepts

1. **Purpose of Unpivot:**
    - Simplifies data analysis by converting column headers into a single column (e.g., categorical variables).
    - Ideal for scenarios where you need to aggregate data further or visualize it in a long format.
2. **Syntax Overview:**
    - Use the **stack** function inside a **selectExpr** to unpivot.
    - Stack reshapes the DataFrame by creating multiple rows for specified columns.
3. **Performance:**
    - Unpivoting can generate many rows, especially if the original DataFrame is wide with numerous columns. Ensure your environment can handle the resulting data volume.

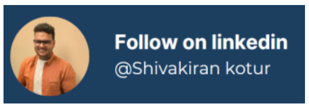
Example: Unpivot in PySpark

Sample Data

Suppose we have the following DataFrame:



Product	North	South	East	West
A	100	200	150	130
B	150	300	200	180



Follow me on LinkedIn – Shivakiran kotur

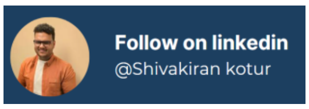
We want to unpivot it to the following format:

Product	Region	Sales
A	North	100
A	South	200
A	East	150
A	West	130
B	North	150
B	South	300
B	East	200
B	West	180

Code Implementation

```
from pyspark.sql import SparkSession

# Create a Spark session
spark = SparkSession.builder.appName("UnpivotExample").getOrCreate()

# Sample data
data = [
    ("A", 100, 200, 150, 130),
    ("B", 150, 300, 200, 180)
]
columns = ["Product", "North", "South", "East", "West"]

# Create the DataFrame
df = spark.createDataFrame(data, columns)

# Unpivot the DataFrame using stack
unpivoted_df = df.selectExpr(
    "Product",
    "stack(4, 'North', North, 'South', South, 'East', East, 'West', West) as (Region, Sales)"
)

# Show the results
unpivoted_df.show()
```

Follow me on LinkedIn – Shivakiran kotur



Explanation of Code

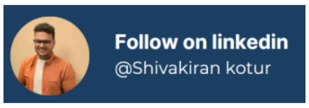
1. **Input DataFrame:**
   - Each column (North, South, East, West) represents a region's sales for each product.
2. **selectExpr with stack:**
   - The **stack** function takes two arguments:
     - The number of columns being unpivoted (4 in this case).
     - A sequence of column-value pairs: 'ColumnName1', ColumnValue1, 'ColumnName2', ColumnValue2, ...
   - The result is two new columns: the first contains the column names (now rows, Region), and the second contains the corresponding values (Sales).
3. **Aliasing Columns:**
   - The stack result is aliased as (Region, Sales) to give meaningful names to the new columns.

Alternative Methods

Using withColumn and union:

If stack isn't flexible enough, you can manually combine rows for each column:

```
from pyspark.sql import functions as F

# Create a DataFrame with union operations for unpivoting
north = df.select("Product", F.lit("North").alias("Region"), F.col("North").alias("Sales"))
south = df.select("Product", F.lit("South").alias("Region"), F.col("South").alias("Sales"))
east = df.select("Product", F.lit("East").alias("Region"), F.col("East").alias("Sales"))
west = df.select("Product", F.lit("West").alias("Region"), F.col("West").alias("Sales"))

# Combine all rows using union
unpivoted_df = north.union(south).union(east).union(west)

# Show results
unpivoted_df.show()
```



Notes

1. **Performance Considerations:**
   - stack is efficient for unpivoting a large number of columns.
   - The union method may become unwieldy for many columns, but it offers more control over the transformation process.
2. **Dynamic Column Unpivoting:** If the column names are not fixed (dynamic), you can:
   - Collect the column names dynamically using df.columns.
   - Construct the selectExpr or union queries programmatically.
3. **Resulting Format:**
   - After unpivoting, the data will have more rows but fewer columns.
   - Ensure downstream processes are optimized to handle the increased row count.

Unpivoting is a powerful operation for restructuring data and is frequently used in data preprocessing, reporting, and machine learning pipelines.