Master Pyspark Zero to Hero:

Pivot in PySpark

The pivot operation in PySpark is used to transpose rows into columns based on a specified column's unique values. It's particularly useful for creating wide-format data where values in one column become new column headers, and corresponding values from another column fill those headers.

Key Concepts

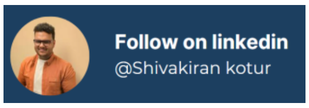
1. **groupBy and pivot:**
    - The pivot method is typically used in combination with groupBy. You group by certain columns and pivot one column to create new columns.
2. **Aggregation Function:**
    - You need to specify an aggregation function (like sum, avg, count, etc.) to fill the values in the pivoted columns.
3. **Performance Consideration:**
    - Pivoting can be computationally expensive, especially with a high number of unique values in the pivot column. For better performance, explicitly specify the values to pivot if possible.
4. **Syntax:**

```
dataframe.groupBy("group_column").pivot("pivot_column").agg(aggregation_function)
```

Example Code: Pivot in PySpark

Sample Data

Imagine we have a DataFrame of sales data with the following schema:



Follow me on LinkedIn – Shivakiran kotur

Product	Region	Sales
A	North	100
B	North	150
A	South	200
B	South	300

We want to pivot the data so that regions (North, South) become columns and the sales values are aggregated.

Code Implementation

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import sum

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

# Create a sample DataFrame
data = [
    ("A", "North", 100),
    ("B", "North", 150),
    ("A", "South", 200),
    ("B", "South", 300)
]
columns = ["Product", "Region", "Sales"]

df = spark.createDataFrame(data, columns)

# Pivot the DataFrame
pivoted_df = df.groupBy("Product").pivot("Region").agg(sum("Sales"))

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

Output

Product	North	South
A	100	200
B	150	300

Follow on linkedin
@Shivakiran kotur

Follow me on LinkedIn – Shivakiran kotur

Explanation of Code

1. **groupBy("Product"):**
   - Groups the data by the Product column.
2. **pivot("Region"):**
   - Transforms unique values in the Region column (North, South) into new columns.
3. **agg(sum("Sales")):**
   - Computes the sum of Sales for each combination of Product and new columns created by the pivot.

Notes

- **Explicit Pivot Values:** To improve performance, you can specify the pivot values explicitly:

  ```
  df.groupBy("Product").pivot("Region", ["North", "South"]).agg(sum("Sales"))
  ```

- **Handling Null Values:** If some combinations of groupBy and pivot values have no corresponding rows, the resulting cells will contain null.
- **Alternative Aggregations:** You can use other aggregation functions like avg, max, min, etc.

This approach is commonly used in creating summary reports or preparing data for machine learning models where wide-format data is required.