PySpark Pivot and Unpivot

🔹 1. What is Pivot in PySpark?

- **Pivoting**is the process ofconverting **rowsinto columns**.
- It's commonly used in reporting and summarization (similar to Excel Pivot Table).
- In PySpark, we use the **pivot()** function along with an aggregation.

🔹 2. Syntax for Pivot

```
df.groupBy(<grouping_columns>).pivot(<pivot_column>,
<va lue s> ).a gg( <a ggr eg ati on >)
```

- **groupBy** → column(s) to keepfixed
- **pivot** → column whose valueswill become new columns
- **agg** → defines how values willbe aggregated (sum, count, avg, etc.)

🔹 3. Example — Pivot in PySpark

📊 Input Data

department	employee	salary
IT	A	3000
IT	B	4000
HR	C	2500
HR	D	2800
Finance	E	3500

✅ Pivot by Department

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

spark = SparkSession.builder.appName("PivotExample").getOrCreate()

data = [

    ("IT", "A", 3000),
    ("IT", "B", 4000),
    ("HR", "C", 2500),
    ("HR", "D", 2800),
    ("Finance", "E", 3500)
]

columns = ["department", "employee", "salary"]
df = spark.createDataFrame(data, columns)
# Pivot: Total salary per department
pivot_df = df.groupBy().pivot("department").agg(sum("salary"))
pivot_df.show()
```

🔹 Output

Finance	HR	IT
3500	5300	7000

🔹 4. Pivot with Multiple Grouping Columns

Example: Group by **department** and pivot on **employee**.

```
pivot_multi =
df.groupBy("department").pivot("employee").agg(sum("salary"))
pivot_multi.show()
```

This creates **employees as columns**, grouped by department.

🔹 5. Pivot with Specific Values

Instead of scanning all distinct values, you can **provide a list** of pivot values (better performance).

```
pivot_df = df.groupBy("department").pivot("employee", ["A", "B",
"C"]).agg(sum("salary"))
pivot_df.show()
```

Helps in large datasets to **avoid scanning all distinct values**.

🔹 6. What is Unpivot in PySpark?

- **Unpivot (or melt)** is the reverse of Pivot.
- It **converts columns back into rows**.
- PySpark **does not have a direct unpivot function**, but we can achieve it using:
  - `selectExpr with stack() function`
  - union method (manual way)

🔹 7. Unpivot using `stack()`

Suppose after pivot, we have:

department	A	B	C

IT	3000	4000	null
HR	null	null	2500

We want to unpivot it back.

```
unpivot_df = pivot_multi.selectExpr(
    "department",
    "stack(3, 'A', A, 'B', B, 'C', C) as (employee, salary)"
).where("salary is not null")
unpivot_df.show()
```

🔹 Output

department	employee	salary
IT	A	3000
IT	B	4000
HR	C	2500

🔹 8. Unpivot using Union

```
from pyspark.sql.functions import lit

unpivot_union = (
    df.select("department", lit("A").alias("employee"), df["A"].alias("salary"))
    .union(df.select("department", lit("B").alias("employee"), df["B"].alias("salary")))
    .union(df.select("department", lit("C").alias("employee"), df["C"].alias("salary")))
).where("salary is not null")
unpivot_union.show()
```

This is **less elegant** but useful when you want full control.

🔹 9. Best Practices & Interview Tips

1. Always **provide values in pivot()** when possible → improves performance.
2. For **large datasets**, prefer **pivot with aggregation** instead of wide joins.
3. **Unpivot using stack()** is the most efficient and clean way in PySpark.
4. **Null values** are common in pivot → handle them using na.fill() or `coalesce()`.

Let's build your Data Engineering journey together!

Call us directly at: 9989454737

https://seekhobigdata.com/