Master PySpark: From Zero to Big Data Hero!!

Union and UnionByName in PySpark

In PySpark, both Union and UnionByName are operations that allow you to combine two or more DataFrames. However, they do this in slightly different ways, particularly regarding how they handle column names.

1. Union

Definition: The union() function is used to combine two DataFrames with the same schema (i.e., the same number of columns with the same data types). It appends the rows of one DataFrame to the other.

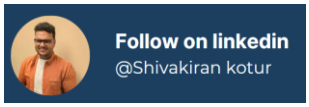
Key Characteristics:

- The DataFrames must have the same number of columns.
- The columns must have compatible data types.
- It does not automatically handle column names that differ between DataFrames.

Syntax:

```
DataFrame.union(otherDataFrame)
```

```
from pyspark.sql import SparkSession
# Create a Spark session
spark = SparkSession.builder.appName("Union Example").getOrCreate()

# Create two DataFrames with the same schema
data1 = [("Alice", 1), ("Bob", 2)]
data2 = [("Cathy", 3), ("David", 4)]

columns = ["Name", "Id"]

df1 = spark.createDataFrame(data1, columns)
df2 = spark.createDataFrame(data2, columns)

# Perform union
result_union = df1.union(df2)

# Show the result
result_union.show()
```

Follow me on LinkedIn – Shivakiran kotur

```
+-----+---+
| Name| Id|
+-----+---+
|Alice|  1|
|  Bob|  2|
|Cathy|  3|
|David|  4|
+-----+---+
```

2. UnionByName

Definition: The unionByName() function allows you to combine two DataFrames by matching column names. If the DataFrames do not have the same schema, it will fill in missing columns with null.

Key Characteristics:

- It matches DataFrames by column names rather than position.
- If the DataFrames have different columns, it will include all columns and fill in null for missing values in any DataFrame.
- You can specify allowMissingColumns=True to ignore missing columns.

Syntax:

```
DataFrame.unionByName(otherDataFrame, allowMissingColumns=False)

# Create two DataFrames with different schemas
data3 = [("Eve", 5), ("Frank", 6)]
data4 = [("Grace", "New York"), ("Hannah", "Los Angeles")]

columns1 = ["Name", "Id"]
columns2 = ["Name", "City"]

df3 = spark.createDataFrame(data3, columns1)
df4 = spark.createDataFrame(data4, columns2)

# Perform unionByName
result_union_by_name = df3.unionByName(df4,
allowMissingColumns=True)
```

```
# Show the result
result_union_by_name.show()
```

```
        Name: string
        Id: long
        City: string

+------+----+-----------+
|  Name|  Id|       City|
+------+----+-----------+
|   Eve|   5|       null|
| Frank|   6|       null|
| Grace|null|   New York|
|Hannah|null|Los Angeles|
+------+----+-----------+
```

Summary of Differences

Feature	Union	UnionByName
Column Matching	Positional	By Name
Missing Columns Handling	Does not allow	Allows with `null` for missing
Schema Requirement	Must be identical	Can differ

Conclusion

In PySpark, use union() when you have DataFrames with the same schema and need a straightforward concatenation. Use unionByName() when your DataFrames have different schemas and you want to combine them by matching column names while handling missing columns.