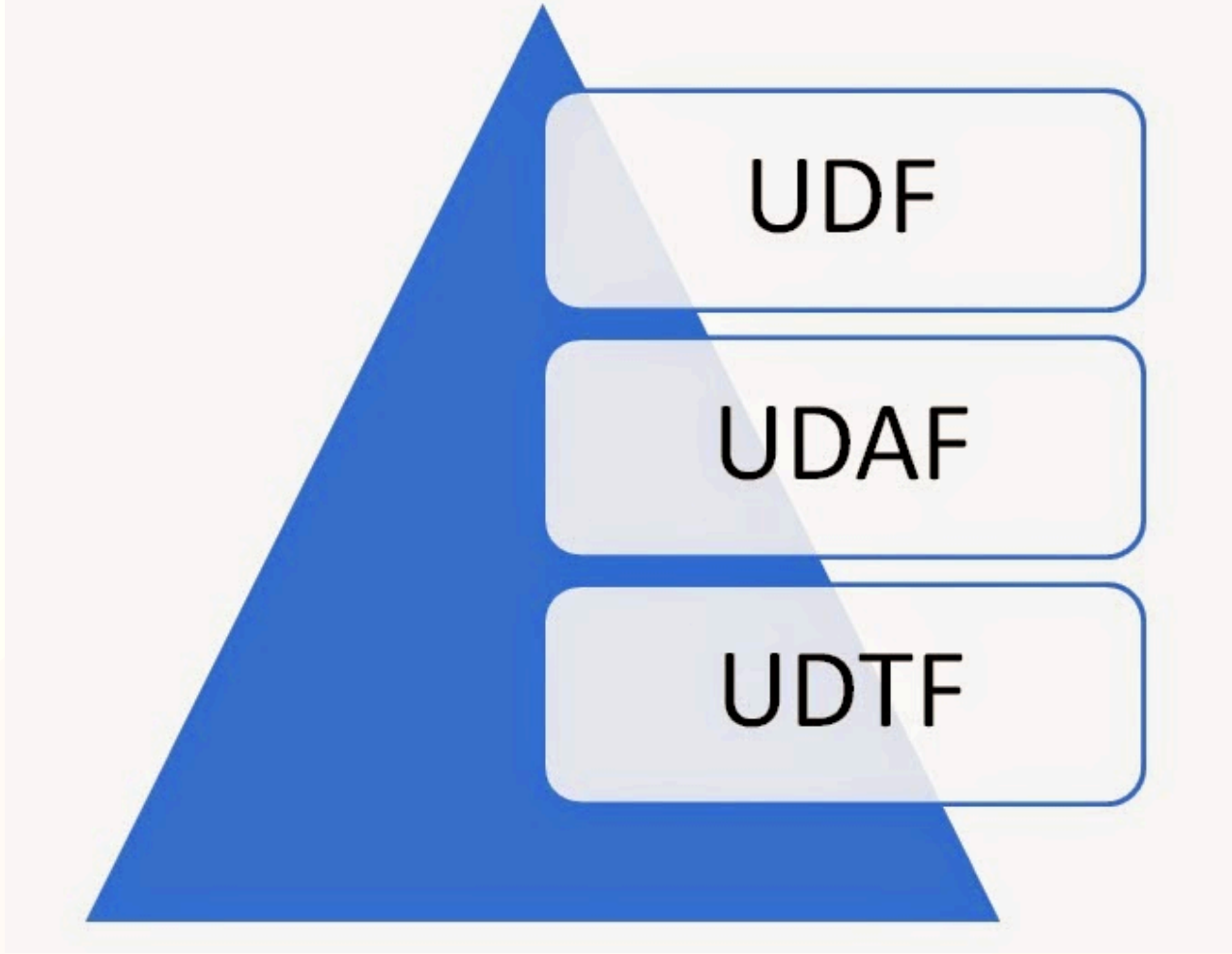
UDF vs UDFA vs UDFT in Databricks

Ganesh R
Azure Data Engineer

In Databricks, these terms might refer to different functionalities, but let's clarify them individually:

Why it matters: External user-defined functionsA UDF can act on a single row or act on multiple rows at once. Spark SQL also supports integration of existing Hive implementations of UDFs, user defined aggregate functions (UDAF), and user defined table functions (UDTF).



1. UDF (User-Defined Function):

- **What it is:** A function written in Python, Scala, Java, or R that can be registered and used within Spark SQL or DataFrame API for custom transformations.
- **Use case:** To apply complex transformations that are not supported by built-in Spark functions.

```python
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType

def square(x):
    return x * x

square_udf = udf(square, IntegerType())
df.withColumn("squared_value", square_udf(df["value"]))
```

P**ros:**

- Allows for highly customized transformations.
- Easy to write using familiar programming languages.

Cons:

- Slower than native Spark functions because it involves serialization/deserialization and JVM/Python interaction.

2. UDFA (User-Defined Aggregate Function):

- **What it is:** A custom aggregate function that can process groups of rows and return a single aggregated value.
- **Use case:** To perform aggregation computations that are not supported by built-in aggregation functions.

```python
from pyspark.sql.expressions import UserDefinedAggregateFunction
from pyspark.sql.types import *

class MySum(UserDefinedAggregateFunction):
    def inputSchema(self):
        return StructType([StructField("input", IntegerType())])

    def bufferSchema(self):
        return StructType([StructField("sum", IntegerType())])

    def dataType(self):
        return IntegerType()

    def deterministic(self):
        return True

    def initialize(self, buffer):
        buffer[0] = 0

    def update(self, buffer, input):
        buffer[0] += input

    def merge(self, buffer1, buffer2):
        buffer1[0] += buffer2[0]

    def evaluate(self, buffer):
        return buffer[0]

my_sum_udaf = MySum()
df.groupBy("group_col").agg(my_sum_u(df["value"]))
```

Pros:

- Useful for advanced aggregation logic.

Cons:

- More complex to implement than UDFs.
- Similar performance limitations as UDFs.

3. UDFT (User-Defined Table Function):

- **What it is:** A custom function that generates a table or DataFrame as output. It's often used to return multiple rows or columns based on a single input.
- **Use case:** To create functions that can return a table-like structure instead of a scalar value.

```python
from pyspark.sql.functions import explode
from pyspark.sql.types import ArrayType, StringType

def split_words(text):
    return text.split(" ")

split_words_udft = udf(split_words, ArrayType(StringType()))
df.withColumn("words", split_words_udft(df["text"]))
```

Pros:

- Useful for operations like expanding arrays or generating rows dynamically.

Cons:

- May require additional steps (like explode) to integrate into workflows.

Comparison:

Feature	UDF	UDFA	UDFT
Output	Scalar value	Aggregated value	Table-like structure
Performance	Slower than native	Slower than native	Similar to UDF/UDFAs
Typical Use Case	Row-based logic	Group-based logic	Generate multiple rows
Complexity	Simple	Moderate	Moderate

Pros:

- Useful for operations like expanding arrays or generating rows dynamically.

Cons:

- May require additional steps (like explode) to integrate into workflows.

If you're working in Databricks, and performance is critical, always prefer built-in Spark SQL or DataFrame API functions over UDFs/UDFA/UDFTs, since these custom implementations may slow down distributed operations. Let me know if you'd like examples for a specific context!

Follow for more content like this

Ganesh R

Senior Azure Data Engineer