Apache Spark 2 using Scala

Processing Column Data using Pre-defined Functions

1. Overview

In Spark 2 (with Scala), **pre-defined functions** are ready-made methods provided by Spark's **org.apache.spark.sql.functions** object.
They help you manipulate, transform, and derive new columns from DataFrames **without writing custom UDFs**.

🔹 Advantages:

- Optimized at **Catalyst optimizer** level (faster than UDFs)
- Works well with Spark SQL expressions
- Supports **null-safe operations**
- Available for **string, date/time, math, aggregate, and conditional operations**

2. Commonly Used Categories of Pre-defined Functions

Category	Example Functions
String Operations	concat, concat_ws, upper, lower, substring, length, trim
Date/Time	current_date, current_timestamp, datediff, year, month, day of month
Math Operations	abs, round, ceil, floor, pow, sqrt
Conditional Logic	when, otherwise, lit
Array/Map	split, size, array_contains, explode
Aggregations	count, sum, avg, max, min

3. Code Example – Processing Columns with Pre-defined Functions

```
import org.apache.spark.sql.{SparkSession, functions => F}
object ColumnFunctionsExample {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder()
      .appName("PreDefinedFunctionsExample")
      .master("local[*]")
      .getOrCreate()
    import spark.implicits._
    // Sample DataFrame
    val df = Seq(
      (1, "john doe", 5000.5, "2024-01-15"),
      (2, "jane SMITH", 7000.0, "2023-12-20"),
      (3, "mike Brown", 4500.75, "2024-02-10")
```

```
    ).toDF("id", "name", "salary", "join_date")
    println("Original DataFrame")
    df.show()
    // Applying Pre-defined Functions
    val processedDF = df
      .withColumn("name_upper", F.upper(F.col("name")))
      .withColumn("name_first_4", F.substring(F.col("name"), 1, 4))
      .withColumn("annual_salary", F.round(F.col("salary") * 12, 2))
      .withColumn("years_since_join", F.round(F.datediff(F.current_date(), F.col("join_date")) / 365, 1))
      .withColumn("salary_category",
        F.when(F.col("salary") >= 6000, "High")
         .when(F.col("salary") >= 5000, "Medium")
         .otherwise("Low")
      )
    println("Processed DataFrame")
    processedDF.show(false)
    spark.stop()
  }
}
```

4. Code Explanation

- **F.upper** – Converts column value to uppercase.
- **F.substring** – Extracts substring from a given position.
- **F.round** – Rounds values to the nearest decimal places.
- **F.datediff** – Finds difference in days between two dates.
- **F.when + otherwise** – Conditional logic for categorization.
- **F.col** – Refers to a column without using string literals everywhere.

5. Sample Output

Original DataFrame

```
+---+----------+-------+----------+
|id |name      |salary |join_date |
+---+----------+-------+----------+
|1  |john doe  |5000.5 |2024-01-15|
|2  |jane SMITH|7000.0 |2023-12-20|
|3  |mike Brown|4500.75|2024-02-10|
+---+----------+-------+----------+
```

Processed DataFrame

```
+---+----------+-------+----------+-----------+-------------+-------------+---------------+---------------+
|id |name      |salary |join_date |name_upper |name_first_4 |annual_salary|years_since_join|salary_category|
+---+----------+-------+----------+-----------+-------------+-------------+---------------+---------------+
|1  |john doe  |5000.5 |2024-01-15|JOHN DOE   |john         |60006.0      |0.6            |Medium         |
|2  |jane SMITH|7000.0 |2023-12-20|JANE SMITH |jane         |84000.0      |0.6            |High           |
|3  |mike Brown|4500.75|2024-02-10|MIKE BROWN |mike         |54009.0      |0.5            |Low            |
+---+----------+-------+----------+-----------+-------------+-------------+---------------+---------------+
```

6. Good GitHub Repositories for Reference

- Apache Spark Scala Examples – MrPowers – Many column transformation examples.
- Learning Journal – Spark in Scala – Beginner to advanced examples.

- Awesome Spark by datamechanics – Curated Spark resources.

7. Quick Quiz

Q1. Which function is used to extract year from a date column?
a) year()
b) extractYear()
c) getYear()
d) yearof()

Q2. What is the main advantage of pre-defined functions over UDFs in Spark?
a) Easier to write
b) Faster due to Catalyst optimizer
c) No need to import libraries
d) Runs only in Python

Q3. Which category does concat_ws belong to?
a) Math
b) String
c) Conditional
d) Aggregate

Q4. Which function can be used for conditional column creation?
a) case
b) if
c) when
d) switch