from pyspark.sql import SparkSession

from pyspark.sql.functions import col, udf, when

from pyspark.sql.types import DoubleType

from fuzzywuzzy import fuzz

# ✅ Initialize Spark Session

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

# ✅ Fuzzy Matching Functions

def fuzz\_ratio(str1, str2):

return fuzz.ratio(str1, str2) / 100.0 if str1 and str2 else 0.0

def fuzz\_partial\_ratio(str1, str2):

return fuzz.partial\_ratio(str1, str2) / 100.0 if str1 and str2 else 0.0

def fuzz\_token\_sort\_ratio(str1, str2):

return fuzz.token\_sort\_ratio(str1, str2) / 100.0 if str1 and str2 else 0.0

def fuzz\_token\_set\_ratio(str1, str2):

return fuzz.token\_set\_ratio(str1, str2) / 100.0 if str1 and str2 else 0.0

# ✅ Register UDFs

fuzz\_ratio\_udf = udf(fuzz\_ratio, DoubleType())

fuzz\_partial\_ratio\_udf = udf(fuzz\_partial\_ratio, DoubleType())

fuzz\_token\_sort\_ratio\_udf = udf(fuzz\_token\_sort\_ratio, DoubleType())

fuzz\_token\_set\_ratio\_udf = udf(fuzz\_token\_set\_ratio, DoubleType())

# ✅ Read CSV File (Assumes `Name\_Class` & `Variant\_Class` are already classified)

df = spark.read.csv("your\_file.csv", header=True, inferSchema=True)

# ✅ Apply All Fuzzy Matching in a \*\*Single Step\*\*

df = df.withColumns({

"Fuzz\_Ratio": when(

(col("Name\_Class").isin(["Alphanumeric", "Character Only"])) &

(col("Variant\_Class").isin(["Alphanumeric", "Character Only"])),

fuzz\_ratio\_udf(col("Name"), col("Name\_Variant"))

).otherwise(0.0),

"Fuzz\_Partial\_Ratio": when(

(col("Name\_Class").isin(["Alphanumeric", "Character Only"])) &

(col("Variant\_Class").isin(["Alphanumeric", "Character Only"])),

fuzz\_partial\_ratio\_udf(col("Name"), col("Name\_Variant"))

).otherwise(0.0),

"Fuzz\_Token\_Sort\_Ratio": when(

(col("Name\_Class").isin(["Alphanumeric", "Character Only"])) &

(col("Variant\_Class").isin(["Alphanumeric", "Character Only"])),

fuzz\_token\_sort\_ratio\_udf(col("Name"), col("Name\_Variant"))

).otherwise(0.0),

"Fuzz\_Token\_Set\_Ratio": when(

(col("Name\_Class").isin(["Alphanumeric", "Character Only"])) &

(col("Variant\_Class").isin(["Alphanumeric", "Character Only"])),

fuzz\_token\_set\_ratio\_udf(col("Name"), col("Name\_Variant"))

).otherwise(0.0)

})

# ✅ Show Final Results

df.select("Name", "Name\_Variant", "Name\_Class", "Variant\_Class",

"Fuzz\_Ratio", "Fuzz\_Partial\_Ratio",

"Fuzz\_Token\_Sort\_Ratio", "Fuzz\_Token\_Set\_Ratio").show(truncate=False)

# ✅ Function to check if two numbers are within a tolerance range

def numeric\_with\_tolerance(val1, val2, tolerance=0.01): # Default tolerance is 1%

try:

num1 = float(val1)

num2 = float(val2)

# Check if numbers are within tolerance

return 1 if abs(num1 - num2) / max(abs(num1), abs(num2)) <= tolerance else 0

except ValueError:

return 0 # If one or both values are not numbers, return 0

# Register as PySpark UDF

numeric\_with\_tolerance\_udf = udf(numeric\_with\_tolerance, IntegerType())

# ✅ Function to check if two dates match exactly

def exact\_date\_match(date1, date2):

date\_formats = ["%Y-%m-%d", "%d/%m/%Y", "%m-%d-%Y", "%Y/%m/%d"] # Common formats

for fmt in date\_formats:

try:

parsed\_date1 = datetime.datetime.strptime(date1, fmt)

parsed\_date2 = datetime.datetime.strptime(date2, fmt)

return 1 if parsed\_date1 == parsed\_date2 else 0

except ValueError:

continue # If parsing fails, try another format

return 0 # Return 0 if they are not valid dates

# Register as PySpark UDF

exact\_date\_match\_udf = udf(exact\_date\_match, IntegerType())

# ✅ Function to determine whether to apply numeric or date comparison

def classify\_and\_compare(val1, val2):

# Try numeric comparison

try:

float(val1)

float(val2)

return numeric\_with\_tolerance(val1, val2) # Use tolerance check

except ValueError:

pass # Not both numeric, continue to date check

# Try date comparison

if exact\_date\_match(val1, val2) == 1:

return 1 # Dates match exactly

return 0 # If neither numeric nor date, return 0

# Register UDF

classify\_and\_compare\_udf = udf(classify\_and\_compare, IntegerType())