**🚀 Relearning PySpark with Databricks: A Quick Dive into Data Reading & Transformations!**

🔹 **Scenario 2: Drop Multiple Columns**  
Remove both Item\_Visibility and Item\_Type from the dataset.

df.drop('Item\_Visibility', 'Item\_Type').display()

**🔹 Scenario - 1: Drop Duplicates**

Remove all duplicate rows from the dataset.

df.dropDuplicates().display()

**🔹 Scenario - 2: Drop Duplicates Based on Column(s)**

Remove duplicates based on a subset of columns (for example, Item\_Type).

df.drop\_duplicates(subset=['Item\_Type']).display()

**🔹 Scenario - 3: Drop Duplicates Using Distinct**

Return a DataFrame with only distinct rows.

df.distinct().display()

**🔹 Scenario - 4: Using UNION and UNION BY NAME**

# Preparing DataFrames

data1 = [('1','kad'), ('2','sid')]

schema1 = 'id STRING, name STRING'

df1 = spark.createDataFrame(data1, schema1)

data2 = [('3','rahul'), ('4','jas')]

schema2 = 'id STRING, name STRING'

df2 = spark.createDataFrame(data2, schema2)

# Displaying DataFrames

df1.display()

df2.display()

# Performing UNION operation

df1.union(df2).display()

# Performing UNION BY NAME operation (this matches the column names)

df1.unionByName(df2).display()

This scenario prepares two DataFrames and performs UNION and UNION BY NAME operations.

**🔹 Scenario - 5: String Functions (Initcap Example)**

# Applying string function to capitalize the first letter of each word

df.select(initcap('Item\_Type').alias('capitalized\_Item\_Type')).display()

**🔹 Scenario - 6: Date Functions**

Using date functions like current\_date(), date\_add(), date\_sub(), and date\_diff().

# Adding current date to the DataFrame

df = df.withColumn('curr\_date', current\_date())

df.display()

# Adding a week to the current date

df = df.withColumn('week\_after', date\_add('curr\_date', 7))

df.display()

# Subtracting a week from the current date

df = df.withColumn('week\_before', date\_sub('curr\_date', 7))

df.display()

# Date difference between two columns

df = df.withColumn('datediff', datediff('week\_after', 'curr\_date'))

df.display()

# Formatting the date to a specific format

df = df.withColumn('week\_before', date\_format('week\_before', 'dd-MM-yyyy'))

df.display()

**🔹 Scenario - 7: Handling Nulls**

Dropping or filling null values in the DataFrame.

# Dropping rows where all values are null

df.dropna('all').display()

# Dropping rows where any value is null

df.dropna('any').display()

# Dropping rows where 'Outlet\_Size' is null

df.dropna(subset=['Outlet\_Size']).display()

# Filling null values with a specific value

df.fillna('NotAvailable').display()

# Filling null values in a specific column

df.fillna('NotAvailable', subset=['Outlet\_Size']).display()

**🔹 Scenario - 8: SPLIT and Indexing**

Splitting a column by space and indexing the results.

# Splitting a column by a space

df.withColumn('Outlet\_Type', split('Outlet\_Type', ' ')).display()

# Indexing the result of the split (getting the second element from the split list)

df.withColumn('Outlet\_Type', split('Outlet\_Type', ' ')[1]).display()

**🔹 Scenario - 9: Explode Function**

Exploding an array column into separate rows.

# Exploding an array column into individual rows

df\_exp = df.withColumn('Outlet\_Type', split('Outlet\_Type', ' '))

df\_exp.display()

df\_exp.withColumn('Outlet\_Type', explode('Outlet\_Type')).display()

**🔹 Scenario - 10: GroupBy Operations**

Performing groupBy operations on the DataFrame.

# Group by a column and aggregate with sum

df.groupBy('Item\_Type').agg(sum('Item\_MRP')).display()

# Group by a column and aggregate with average

df.groupBy('Item\_Type').agg(avg('Item\_MRP')).display()

# Group by multiple columns and aggregate

df.groupBy('Item\_Type', 'Outlet\_Size').agg(sum('Item\_MRP').alias('Total\_MRP')).display()

# Group by multiple columns and aggregate with multiple functions

df.groupBy('Item\_Type', 'Outlet\_Size').agg(sum('Item\_MRP'), avg('Item\_MRP')).display()