Processing DataFrames .

DESCRIPTION

Problem Statement: Using Scala, perform the below tasks on Spark DataFrames.

Create case classes with the following fields: Department, Employee, and DepartmentWithEmployees.

Note: Create the DepartmentWithEmployees instances from Departments and Employees.

Insert at least four values.

Create two DataFrames from the list of the above case classes.

Combine the two DataFrames and write the combined DataFrame to a parquet file.

Use filter() or where() clause to return the rows whose first name is either “Alice” or “John”

Note: Use first name filter as per your entries.

Retrieve rows with missing firstName or lastName.

Find the distinct (firstName, lastName) combinations.

Steps to Perform:

Create the case classes

case class Department(id: String, name: String)

case class Employee(firstName: String, lastName: String, email: String, salary: Int)

case class DepartmentWithEmployees(department: Department, employees: Seq[Employee])

Create the Departments

val department1 = new Department(“123456”, “Computer Science”)

val department2 = new Department(“789012”, “Mechanical Engineering”)

val department3 = new Department(“345678”, “Theater and Drama”)

val department4 = new Department(“901234”, “Indoor Recreation”)

Create the Employees

val employee1 = new Employee(“Alice”, “Mathew”, “no- reply@harvard.edu”, 100000)

val employee2 = new Employee(“John”, “Singleton”, “no-

reply@stanford.edu”, 120000)

```
val employee3 = new Employee(“matei”, null, “no-reply@oxford.edu”, 140000)

val employee4 = new Employee(null, “wendell”, “no-reply@princeton.edu”, 160000)

# Create the DepartmentWithEmployees instances from Departments and Employees

val departmentWithEmployees1 = new DepartmentWithEmployees(department1,
Seq(employee1, employee2))

val departmentWithEmployees2 = new DepartmentWithEmployees(department2,
Seq(employee3, employee4))

val departmentWithEmployees3 = new DepartmentWithEmployees(department3,
Seq(employee1, employee4))

val departmentWithEmployees4 = new DepartmentWithEmployees(department4,
Seq(employee2, employee3))

# Create DataFrames from a list of the case classes

val departmentsWithEmployeesSeq1 = Seq(departmentWithEmployees1,
departmentWithEmployees2)

val df1 = departmentsWithEmployeesSeq1.toDF() display(df1)

val departmentsWithEmployeesSeq2 = Seq(departmentWithEmployees3,
departmentWithEmployees4)

val df2 = departmentsWithEmployeesSeq2.toDF() display(df2)

# combining the two DataFrames

val unionDF = df1.unionAll(df2) display(unionDF)

# Write the combined DataFrame to a Parquet file

unionDF.write.parquet(“/user/simpli_learn/simplitest”)

# Explode the employees column

val flattenDF = parquetDF.select(functions.explode($”employees”)).flattenSchema

val columnsRenamed = Seq(“firstName”, “lastName”, “email”, “salary”)

val explodeDF = flattenDF.toDF(columnsRenamed: _*) explodeDF.show()


# Using filter() to return the rows where first name is either ‘Alice’ or ‘John’

val filterDF = explodeDF

.filter($”firstName” === “Alice” || $”firstName” === “John”)
```

```
.sort($”lastName”.asc)
display(filterDF)

# Retrieve rows with missing firstName or lastName
val filterNonNullDF = nonNullDF.filter($”firstName” === ““ || $”lastName”
=== ““).sort($”email”.asc)
display(filterNonNullDF)
# aggregations using agg() and countDistinct()
import org.apache.spark.sql.functions._
val countDistinctDF = nonNullDF.select($”firstName”, $”lastName”)
.groupBy($”firstName”, $”lastName”)
.agg(countDistinct($”firstName”) as “distinct_first_names”)
display(countDistinctDF)
```