**Day 12 Notes**

**Name : Podutur Lahari - DE126**

**Date : 21-11-2024**

**Key Sections**

1. **Setup and Initialization**
   * Imported libraries: pyspark.sql for working with Spark DataFrames.

Created a Spark session using:  
 SparkSession.builder.appName('pyspark - example join').getOrCreate()

1. **Data Creation**
   * Sample data initialized as a list of tuples with fields: Name, DOB, Gender, and salary.

DataFrame created with specified column names:  
 df = spark.createDataFrame(data=data, schema=columns)

1. **Renaming Columns**

Demonstrated renaming of columns using:  
 df.withColumnRenamed("DOB", "date of birth")

df.withColumnRenamed("Name", "personname")

1. **Selecting Specific Columns**

Used selectExpr to alias and select specific columns:  
data = df.selectExpr("Gender as category", "DOB", "Name as name", "salary")

* + Displayed the updated DataFrame.

1. **Aliasing with col Function**

Showed aliasing using col() for better control over column selection:  
 from pyspark.sql.functions import col

data = df.select(col("Name"), col("DOB"), col("Gender"), col("salary").alias("Amount"))

* + Displayed the DataFrame with the alias applied.

**Techniques Covered**

1. Column Renaming
   * Practical examples of renaming multiple columns for clarity.
2. Column Selection
   * selectExpr: SQL-style expressions for column aliasing.
   * col: A structured and programmatic approach to rename and access columns.
3. Data Display
   * Used .show() to output and inspect DataFrame contents after each operation.

### **Key Sections SQL spark**

1. **File Handling in PySpark**
   * File location: /FileStore/tables/simple\_zipcodes-1.csv
   * File type: CSV
2. **Reading CSV Data**

CSV file read into a PySpark DataFrame using:

df = spark.read.format("csv") \

.option("inferSchema", "false") \

.option("header", "false") \

.option("sep", ",") \

.load(file\_location)

* + Options used:
    - inferSchema: Ensures schema inference is off.
    - header: Specifies if the first row contains headers.
    - sep: Delimiter used in the CSV.

1. **SQL Table Creation**

Temporary SQL table created for the DataFrame using:  
df.createOrReplaceTempView("tempdata")

1. **SQL Queries on the DataFrame**

Display all data:  
spark.sql("SELECT \* FROM tempdata").show()

Select specific columns:

df.select("\_c0", "\_c1").show(5)

Filter rows based on conditions:  
spark.sql("SELECT \* FROM tempdata WHERE \_c4='AZ'").show(5)

1. **Handling Headers**

Modified the header option to true to include column names:  
df = spark.read.format("csv") \

.option("inferSchema", "false") \

.option("header", "true") \

.option("sep", ",") \

.load(file\_location)

DataFrame renamed to "customer":  
df.createOrReplaceTempView("customer")

**Techniques Covered**

1. Reading CSV Files
   * Handling various CSV options like schema inference and header rows.
2. SQL Operations on DataFrames
   * Temporary view creation for SQL-like queries.
   * Filtering, column selection, and table operations.
3. Dynamic File Loading
   * Demonstrated modular code to load different file configurations.

**Key Sections**

1. **Setup and Initialization**

SparkSession initialized:  
spark = SparkSession.builder.appName("example").getOrCreate()

1. **Data Creation**
   * Employee Data:
     + Created a list of tuples with attributes like emp\_id, name, superior\_emp\_id, year\_joined, emp\_dept\_id, gender, and salary.

Loaded into a DataFrame:  
empDF = spark.createDataFrame(data=emp, schema=empColumns)

empDF.show()

* + Department Data:
    - Created a list of tuples with attributes like dept\_name and dept\_id.

Loaded into a DataFrame:  
deptDF = spark.createDataFrame(data=dept, schema=deptColumns)

deptDF.show()

1. **Join Operations**

Inner Join: Matches rows where keys in both DataFrames align.  
empDF.join(deptDF, empDF.emp\_dept\_id == deptDF.dept\_id, "inner").show()

Outer Join: Includes all rows from both DataFrames, with unmatched rows filled with null.  
empDF.join(deptDF, empDF.emp\_dept\_id == deptDF.dept\_id, "outer").show()

Full Join (Alias of Outer Join):  
empDF.join(deptDF, empDF.emp\_dept\_id == deptDF.dept\_id, "full").show()

Left Join: Retains all rows from the left DataFrame and matched rows from the right.  
empDF.join(deptDF, empDF.emp\_dept\_id == deptDF.dept\_id, "left").show()

Right Join: Retains all rows from the right DataFrame and matched rows from the left.  
empDF.join(deptDF, empDF.emp\_dept\_id == deptDF.dept\_id, "right").show()

**Techniques Covered**

1. DataFrame Schema Definition
   * Custom schemas used for DataFrames.
2. Types of Joins
   * Practical examples for each join type: inner, outer, full, left, and right.
3. Join Syntax
   * Used the .join() function with join conditions and join type specified.