Part 1 (Guys, follow the steps)

Step 1: What are we doing in Part 1?

We want to:

Read a CSV file (employee\_data.csv) that has employee details.

Store it in a Spark DataFrame (a table-like structure in memory).

Perform these operations:

Find average salary per department.

Add a column Salary Increase = % difference from average salary of department.

Add a column YearswithCompany = how long the employee has been in the company (from join\_date).

Add a column Salary Category = Low, Medium, or High.

=================================================

Step 2: Prepare your files on Windows

On your Windows computer, open Notepad.

Copy this content and save it as employee\_data.csv:

emp\_id,name,department,salary,join\_date

101,Anita,HR,52000,2018-07-15

102,Ravi,IT,85000,2016-03-10

103,Meera,Finance,76000,2019-11-05

104,Arjun,IT,61000,2021-01-12

105,Farah,HR,48000,2017-06-25

106,Vinod,Finance,83000,2015-09-18

107,Latha,IT,92000,2014-12-01

108,Imran,HR,50000,2020-08-30

Open Notepad, copy this Python program and save it as task10\_part1.py

# task10\_part1.py (for Spark 1.6 on Cloudera QuickStart VM, using local filesystem)

from pyspark import SparkContext

from pyspark.sql import SQLContext, Row

from datetime import datetime, date

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

# Step 1: Start Spark Context and SQL Context

sc = SparkContext(appName="Task10\_Part1")

sqlContext = SQLContext(sc)

# Step 2: Read CSV into RDD (from LOCAL path)

raw\_rdd = sc.textFile("file:///home/cloudera/employee\_data.csv")

header = raw\_rdd.first()

data\_rdd = raw\_rdd.filter(lambda line: line != header)

# Step 3: Parse each line into a Row object

def parse\_line(line):

parts = line.split(",")

return Row(

emp\_id=int(parts[0]),

name=parts[1],

department=parts[2],

salary=int(parts[3]),

join\_date=parts[4]

)

rows\_rdd = data\_rdd.map(parse\_line)

# Step 4: Create DataFrame from RDD

df = sqlContext.createDataFrame(rows\_rdd)

print("=== Original Data ===")

df.show()

# Step 5: Department average salary

dept\_avg = df.groupBy("department").agg(avg("salary").alias("dept\_avg\_salary"))

print("=== Department Average Salary ===")

dept\_avg.show()

# Step 6: Join department avg back to df

df2 = df.join(dept\_avg, "department")

# Step 7: Salary Increase (% vs dept avg)

df2 = df2.withColumn(

"Salary\_Increase",

((col("salary") - col("dept\_avg\_salary")) / col("dept\_avg\_salary")) \* 100

)

# Step 8: Years with Company (using Python UDF)

def years\_since(dstr):

try:

dt = datetime.strptime(dstr, "%Y-%m-%d").date()

return float((date.today() - dt).days) / 365.25

except:

return None

from pyspark.sql.types import DoubleType

from pyspark.sql.functions import udf

years\_udf = udf(years\_since, DoubleType())

df2 = df2.withColumn("YearswithCompany", years\_udf(col("join\_date")))

# Step 9: Salary Category (Low/Medium/High)

df2 = df2.withColumn(

"Salary\_Category",

when(col("salary") < 55000, "Low")

.when((col("salary") >= 55000) & (col("salary") < 80000), "Medium")

.otherwise("High")

)

# Step 10: Show final result

print("=== Final Result ===")

df2.select("emp\_id", "name", "department", "salary", "dept\_avg\_salary",

"Salary\_Increase", "YearswithCompany", "Salary\_Category").show()

# Step 11: Save result to LOCAL CSV (single file)

(df2.select("emp\_id", "name", "department", "salary", "dept\_avg\_salary",

"Salary\_Increase", "YearswithCompany", "Salary\_Category")

.coalesce(1) # ensures one CSV file instead of many

.write

.mode("overwrite")

.format("com.databricks.spark.csv")

.option("header", "true")

.save("file:///home/cloudera/employee\_result\_out"))

sc.stop()

Copy “employee\_data.csv” and “task10\_part1.py” to the VM environment (/home/cloudera/) via WinSCP.

Run:

spark-submit task10\_part1.py

See results printed in terminal