1. Create dataframes using **EMP.csv**, **DEPT.csv**files by explicitly creating schema as per fields.

from pyspark.sql.types import \*

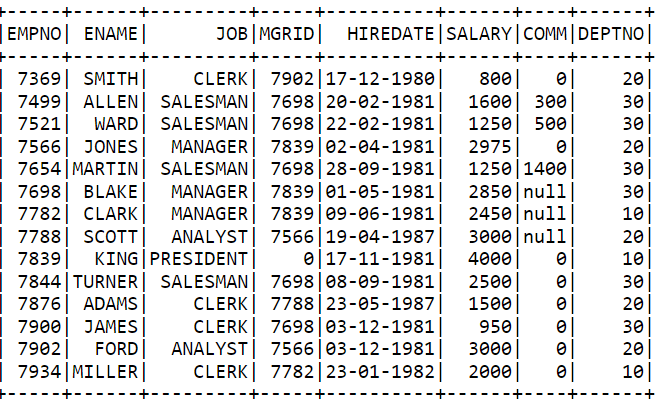
empColumns = [

StructField("EMPNO",LongType()), StructField("ENAME",StringType()), StructField("JOB",StringType()), StructField("MGRID",LongType()), StructField("HIREDATE",TimestampType()), StructField("SALARY",LongType()), StructField("COMM",LongType()), StructField("DEPTNO",LongType())]

empschema = StructType(empColumns)

empdf = spark.read.schema(empschema).option('sep',',').csv("/user/nivassbig/EMP.csv")

empdf.show()



deptColumns = [

StructField("DEPTNO",LongType()),

StructField("DNAME",StringType()),

StructField("LOCATION",StringType())]

deptschema = StructType(deptColumns)

deptdf = spark.read.schema(deptschema).option('sep',',').csv("/user/nivassbig/dept.csv")

deptdf.show()

A white paper with black text

Description automatically generated

1. Using **EMP.json**file find out number of employees present in each department.

empDF=spark.read.json("/user/nivassbig/EMP.json")

empDF.show()

emp\_count\_dept = empDF.groupBy("DEPTNO").count()

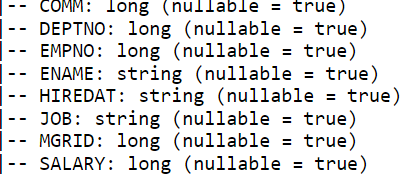
emp\_count\_dept.show()

A close up of a number

Description automatically generated

1. Convert the Hiredate column datatype to date type. Give the column new name as "**HiringDate**"

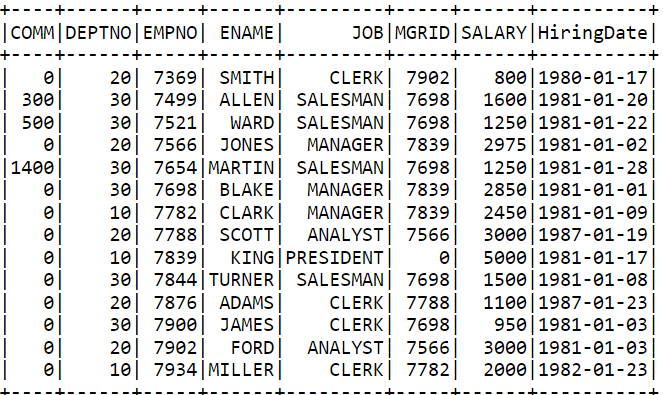
empDF.printSchema()



from pyspark.sql.functions import \*

hiredtdf = empDF.withColumn("HiringDate", to\_date(col("HIREDAT"),"dd-mm-yyyy")).drop("HIREDAT")

hiredtdf.show()



1. Find out **average salaries**of each Department, and print department name, and average salary.

average\_salary = empDF.groupBy("DEPTNO").agg(avg("SALARY").alias("AverageSalary"))

average\_salary.show()

A close up of numbers

Description automatically generated

1. Create Dataframe using **accountdevices.snappy.parquet file**.

accountdevdf=spark.read.parquet("/user/nivassbig/accountdevices.snappy.parquet")

accountdevdf.show()

A table of numbers and names

Description automatically generated with medium confidence

1. Using above dataframe find out each user having how many **device\_ids**.

device\_count= accountdevdf.groupBy("first\_name").agg(countDistinct("device\_id").alias("DeviceCount"))

device\_count.show()

A screenshot of a computer

Description automatically generated

7.   Create dataframe using all csv files present under **accountzips**

folder, by explicitly defining the schema.

accountschema = StructType([ StructField("acct\_num", IntegerType(), True), StructField("acct\_create\_dt", TimestampType(), True), StructField("acct\_clos

e\_dt", TimestampType(), True), StructField("first\_name", StringType(), True), StructField("last\_name", StringType(), True), StructField("address", StringTy

pe(), True), StructField("city", StringType(), True), StructField("state", StringType(), True), StructField("zipcode", StringType(), True), StructField

("phone\_number", StringType(), True), StructField("created", TimestampType(), True), StructField("modified", StringType(), True) ])

accountdf= spark.read.csv("/user/nivassbig/accountzips/\*.csv", header=True, schema=accountschema)

accountdf.show()

A close-up of a white sheet

Description automatically generated

8.   Filter out all account numbers and their date of creation from

above dataframe, and store the results in local file system under

**/root**in **parquet**format.

filterdf = accountdf.select("acct\_num", "acct\_create\_dt")

filterdf.show()

A white paper with black numbers

Description automatically generated

filterdf.write.parquet("/user/nivassbig/account\_creation\_dates.parquet")

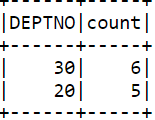


9.From **EMP.json**find out the departments having more than 3 employees.

employee\_department = empDF.groupBy("DEPTNO").count()

dept\_more\_3\_employees = employee\_department.filter(col("count") > 3)

dept\_more\_3\_employees.show()



10.     Using EMP.json and Dept.json, print **employees ids**, their **names**, their **department names**and **location**. Save the result on hdfs in **csv** format.

deptDF=spark.read.json("/user/nivassbig/DEPT.json")

joinedDF = empDF.join(deptDF, empDF["DEPTNO"] == deptDF["DEPTNO"], "inner")

optDF = joinedDF.select(empDF["EMPNO"], empDF["ENAME"], deptDF["DNAME"], deptDF["LOCATION"])

optDF.show()

A white paper with black text

Description automatically generated

optDF.write.csv("/user/nivassbig/output\_join")

