Passport Data Analysis

1) Change the timestamp formats to yyyy-MM-dd HH:mm:ss (remove the millisecond part)

2) Generate a new column "year\_left" showing years left in passport expiration

3) Check the number of invalid mobile numbers (having 11 digits) and create a new column telling the status 'invalid" or "valid"

4) Create a new Column with Network status (BNZ or Others) based on IP address starting with 190-199

5) Find out Invalid Email address, drop those records and get a new DF 'goodata'

6) Find People having SAME fname, lname and email. Drop these Duplicate Records

7) Create a new column 'priority' saying CRITICAL if passport expire left years are less than 5

8) Creat a new Pricipal taking first 3 letters of fname, full lname and appending "@HADOOP.COM"

9) Use of important pyspark.sql.functions.lit , replace, when, otherwise, substr, fill, isnan, isnull, ltrim, rtrim, udf, upper,lower

10) Save the final data to Dynamically Partitioned Table

Solution

1.Import Statements

**from** **pyspark.sql** **import** SparkSession

**from** **pyspark.sql** **import** functions **as** F

**from** **pyspark.sql** **import** types **as** T

2. Load the TSV Data

data = spark.read.format("csv").options(header=True, inferSchema=True, sep="**\t**",

dateFormat="yyyy-MM-dd", timestampFormat="yyyy-MM-dd HH:mm:ss.ssssss", ignoreLeadingWhiteSpace=True,\

ignoreTrailingWhiteSpace=True, path="<DBFS>etl\_sampledata.tsv").load()

3. Print the schema of the loaded file. Schema should be like this

root

|-- fname: string (nullable = true)

|-- lname: string (nullable = true)

|-- principal: string (nullable = true)

|-- emailid: string (nullable = true)

|-- mobile: long (nullable = true)

|-- passport\_make: timestamp (nullable = true)

|-- passport\_expire: timestamp (nullable = true)

|-- ipaddress: string (nullable = true)

4. Change the timestamp formats to yyyy-MM-dd HH:mm:ss (remove the millisecond part)

Functions Used: F.from\_unixtime(), F.unix\_timestamp(), F.round()

Spark withColumn function is used to rename, change the value, convert the datatype of an existing dataframe and also can be used to create new extracted columns

data = data.withColumn("passport\_make", F.from\_unixtime(F.unix\_timestamp(data.passport\_make, format='yyyy-MM-dd HH:mm:ss')))\

.withColumn("passport\_expire", F.from\_unixtime(F.unix\_timestamp(data.passport\_expire, format='yyyy-MM-dd HH:mm:ss')))

5. Generate a new column "year\_left" showing years left in passport expiration

Functions Used: F.datediff(), F.current\_timestamp(), F.round()

data = data.withColumn("years\_left", F.round(F.datediff(data.passport\_expire,F.current\_timestamp())/365,2))

data.select(data.passport\_make,data.passport\_expire,data.years\_left).show(5,False)

6. Check the number of invalid mobile numbers (having 11 digits) and create a new column telling the status 'invalid" or "valid

Functions Used: F.length(), F.when().otherwise, F.substring(), F.groupBy()

data = data.withColumn("mobile\_status",F.when(F.length(data.mobile) <= 10, "valid").otherwise("invalid"))\

data.select("mobile","mobile\_status").show(5,False)

data.groupby("mobile\_status").count().show()

data.select("mobile").filter(F.length(data.mobile) == 11).count()

Clean the Mobile Number by removing 11th digit

data = data.withColumn("mobile", F.substring("mobile", 1, 10))

data.select("mobile").filter(F.length(data.mobile) == 11).count()

Final output count should be 0

7. Create a new Column with Network status (BNZ or Others) based on IP address starting with 190-199

Functions Used: F.when().otherwise, F.substring\_index(),

data = data.withColumn("network", \

F.when(((F.substring\_index("ipaddress", ".",1)) < 200) & ((F.substring\_index("ipaddress", ".",1)) >= 190), "BNZ")\

.otherwise("others"))

data.select("ipaddress","network").filter(data.network == "BNZ").show(5)

*#data.select(F.substring\_index("ipaddress", ".",1)).show*

data.printSchema()

8. Find out Invalid Email address, drop those records and get a new DF 'goodata

Functions Used: F.like(), F.instr(), F.udf(), F.drop()

data.select("emailid").filter(data.emailid.like("%@@%") \

| data.emailid.like("%@#%") | data.emailid.like("**%##%**") \

| data.emailid.like("**%#%**") | (F.instr(data.emailid,"@") == 0 )).count()

Finding All invalid Emails using UDF

**import** **re**

**def** validate\_email(email):

mypat = r"**\"**?([-a-zA-Z0-9.`?{}\_]+@\w+\.\w+)**\"**?"

pattern = re.compile(mypat)

**if** **not** re.match(pattern, email):

**return** False

**else**:

**return** True

validate\_email\_udf = F.udf(validate\_email, T.StringType())

data = data.withColumn("email\_status", validate\_email\_udf(data.emailid))

data.select("emailid","email\_status").filter(data.email\_status == "false").count()

data.select("emailid","email\_status").filter(data.email\_status == "false").show(5,False)

We have the Data having valid Emails Addresses. Drop the temporary columns mobile\_status and email\_status

gooddata = data.filter(data.email\_status == "true").drop("mobile\_status").drop("email\_status")

gooddata.count()

gooddata.printSchema()

9. Find People having SAME fname, lname and email. Drop these Duplicate Records

Functions Used: F.dropDuplicates(), F.collect\_list(), F.size(), F.agg(), df.distinct()

Counting Distinct Records out of Total Good Records – 6873

*#gooddata.groupBy("fname","lname","emailid").count().sort("count").show()*

gooddata.createOrReplaceTempView("mytable")

spark.sql("SELECT DISTINCT (fname, lname, emailid) FROM mytable").count()

Dropping Duplicates recrods using dropDuplicates()

gooddata = gooddata.dropDuplicates(["fname","lname", "emailid"])

gooddata.groupBy("fname","lname","emailid").agg(F.collect\_list("emailid").alias("count"))\

.where(F.size("count") > 1).count()

So, we left with 6708 Unique Records

gooddata.select("fname", "lname","emailid").distinct().count()

10. Create a new column 'priority' saying CRITICAL if passport expire left years are less than 5

Functions Used: F.when(), F.lit(), df.fillna() {alias for df.na.fill()}

gooddata = gooddata.withColumn("priority",F.when(gooddata.years\_left < 3 , F.lit("CRITICAL"))).fillna("LOW")

gooddata.select("years\_left","priority").show(5,False)

11. Creating a new Pricipal taking first 3 letters of fname, full lname and appending "@HADOOP.COM"

Functions Used: F.concat(), F.substring(), F.lower(), F.lit()

gooddata = gooddata.withColumn("principal", F.concat(F.lower(F.substring(gooddata["fname"],1,3))\

, F.lower(gooddata["lname"]), F.lit("@HADOOP.COM")))

gooddata.select("fname","lname","principal").show(5,False)

12. Save the final data to Dynamically Partitioned Table