Original Qn:

Create a dummy data using python in your local computer: -> CSV file -> 100 rows, 5 columns(columns -> first name, last name, age, contact number, salary, gender). Age -> should be between 18-50, Salary should be between 100-200K. Once the file is created in your local computer -> push it to AWS $3 bucket(under RAW folder). Use AWS Glue to transform raw csv data to transformed data in parquet format and put it under Master folder in same s3 bucket. During Transformation Create a new column called status and Based on a condition: if a salary is greater than 120k put rich else poor. Partitioned data based on status. Create a hive table or redhshift table on top of the data and use athena to view the data.

I did all in Databricks:

Step 1: installing faker library to create dummy data

%pip install faker

Step2: creating dummy data

from pyspark.sql import SparkSession

import random

from faker import Faker

# Initialize SparkSession

spark = SparkSession.builder \

    .appName("Dummy Data Creation") \

    .getOrCreate()

# Initialize Faker to generate fake data

fake = Faker()

# Define the number of rows and columns

num\_rows = 10

num\_columns = 6

# Generate dummy data

data = []

for \_ in range(num\_rows):

    first\_name = fake.first\_name()

    last\_name = fake.last\_name()

    age = random.randint(18, 50)  # Age between 18 and 50

    # Generate phone number with country code +1 for USA

    contact\_number =str(random.randint(100, 999)) + "-" + str(random.randint(100, 999)) + "-" + str(random.randint(1000, 9999))

    salary = random.randint(100000, 200000)  # Salary between $100,000 and $200,000

    gender = random.choice(['Male', 'Female'])

    row\_data = [first\_name, last\_name, age, contact\_number, salary, gender]

    data.append(row\_data)

# Define column names

columns = ["first\_name", "last\_name", "age", "contact\_number", "salary", "gender"]

# Create DataFrame

df = spark.createDataFrame(data, columns)

# Show DataFrame

df.show()

snippet of dummy data:

A screenshot of a computer

Description automatically generated

Step3: Saving the dummy data into S3 bucket under folder named “raw”

spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.access.key", "XXXX")

spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.secret.key", "XXXX")

# Define the S3 path where you want to save the CSV files

s3\_output\_path = "s3://puskalawsbucket/side\_project/raw/"

# Write DataFrame to S3

df.write.csv(s3\_output\_path, header=True, mode="overwrite")

df.printSchema()

Step 4: using above df and creating new column named “status” with values “rich” and “poor” and printing new DF to showcase the success

from pyspark.sql.functions import col, when

# Cast the "salary" column to integer

df = df.withColumn("salary", col("salary").cast("int"))

# Add a new column based on "salary" value

newcoldf = df.withColumn("status", when(df["salary"] > 120000, 'Rich').otherwise('Poor'))

# Show the DataFrame

newcoldf.show()

newcoldf.printSchema()

snippet:

A screenshot of a computer

Description automatically generated

Step5: saving the new DF into S3 folder named “master” as parquet and portioned on “status”

spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.access.key", "XXXX")

spark.sparkContext.\_jsc.hadoopConfiguration().set("fs.s3a.secret.key", "XXXX")

# Define the S3 path I want to save the CSV files

s3\_output\_path = "s3://puskalawsbucket/side\_project/master/"

# Write DataFrame to S3 , patrition on status and as parquet file type

newcoldf.write.option("header",True) \

        .partitionBy("status") \

        .mode("overwrite") \

        .parquet("s3://puskalawsbucket/side\_project/master/")

Step6: Saving the final DF “newcoldf” into redshift warehouse as table “employee\_side\_project” under schema named “dev”

df1 = newcoldf.write.mode("overwrite").format("redshift").option("url", "jdbc:redshift://default-workgroup.533267297424.us-east-2.redshift-serverless.amazonaws.com:5439/dev").\

   option("dbtable", "test.employee\_side\_project").\

   option("aws\_iam\_role", "arn:aws:iam::XXXXXX:role/redshiftadmin").\

   option("user", "admin").\

   option("tempdir", "s3a://puskalawsbucket/tempdir/").\

   option("password", "XXXXXXX").save()

Step7: Snippet of table “employee\_side\_project” from redshift:

A screenshot of a computer

Description automatically generated