from pyspark.sql import SparkSession

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

# Create a Spark session

spark = SparkSession.builder \

.appName("Healthcare Insurance Project") \

.config("spark.hadoop.fs.s3a.access.key", " AKIA2UC3CXAXAEAOULHQ") \

.config("spark.hadoop.fs.s3a.secret.key", " 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") \

.config("spark.hadoop.fs.s3a.endpoint", "s3.amazonaws.com") \

.getOrCreate()

# Read data from S3

patients\_df = spark.read.csv("s3a://bucket-bibek/input-data/patients.csv", header=True, inferSchema=True)

subscribers\_df = spark.read.csv("s3a:// bucket-bibek /input-data/subscribers.csv", header=True, inferSchema=True)

claims\_df = spark.read.csv("s3a:// bucket-bibek /input-data/claims.csv", header=True, inferSchema=True)

group\_subgroup\_df = spark.read.csv("s3a:// bucket-bibek /input-data/group\_subgroup.csv", header=True, inferSchema=True)

# Cleaning Patients Data

patients\_df = patients\_df.na.fill("NA").dropDuplicates()

# Cleaning Subscribers Data

subscribers\_df = subscribers\_df.na.fill("NA").dropDuplicates()

# Cleaning Claims Data

claims\_df = claims\_df.na.fill("NA").dropDuplicates()

# Cleaning Group-Subgroup Data

group\_subgroup\_df = group\_subgroup\_df.na.fill("NA").dropDuplicates()

# Save cleaned data back to S3

patients\_df.write.csv("s3a://bucket-bibek/cleaned-data/patients.csv", header=True)

subscribers\_df.write.csv("s3a:// bucket-bibek /cleaned-data/subscribers.csv", header=True)

claims\_df.write.csv("s3a:// bucket-bibek /cleaned-data/claims.csv", header=True)

group\_subgroup\_df.write.csv("s3a:// bucket-bibek /cleaned-data/group\_subgroup.csv", header=True)

-- Redshift Schema Design

CREATE TABLE patients (

patient\_id VARCHAR(50) PRIMARY KEY,

name VARCHAR(100),

age INT,

gender VARCHAR(10),

disease VARCHAR(100),

admission\_date DATE

);

CREATE TABLE subscribers (

subscriber\_id VARCHAR(50) PRIMARY KEY,

name VARCHAR(100),

age INT,

group\_id VARCHAR(50),

subgroup\_id VARCHAR(50)

);

CREATE TABLE claims (

claim\_id VARCHAR(50) PRIMARY KEY,

subscriber\_id VARCHAR(50),

hospital\_id VARCHAR(50),

amount DECIMAL(10,2),

status VARCHAR(50));

CREATE TABLE group\_subgroup (

group\_id VARCHAR(50),

subgroup\_id VARCHAR(50),

PRIMARY KEY (group\_id, subgroup\_id)

);

a. **Identify the Disease with Maximum Claims**

max\_claims\_disease=claims\_df.groupBy("disease").agg(count("claim\_id").alias("total\_claims")) \

.orderBy(col("total\_claims").desc()).first()

print(f"Disease with maximum claims: {max\_claims\_disease['disease']}, Total Claims: {max\_claims\_disease['total\_claims']}")

b. **Find Subscribers Under Age 30**

subscribers\_under\_30 = subscribers\_df.filter(subscribers\_df.age < 30)

subscribers\_under\_30.show()

c. **Hospital Serving the Most Patients**

top\_hospital = claims\_df.groupBy("hospital\_id").agg(count("patient\_id").alias("total\_patients")) \

.orderBy(col("total\_patients").desc()).first()

print(f"Hospital serving the most patients: {top\_hospital['hospital\_id']}, Total Patients: {top\_hospital['total\_patients']}")

d. **Save Results to Redshift**

max\_claims\_disease.write \

.format("jdbc") \

.option("url", "jdbc:redshift:// jdbc:redshift://default-workgroup.730335459374.us-east-2.redshift-serverless.amazonaws.com:5439/dev

") \

.option("dbtable", "project\_output.max\_claims\_disease") \

.option("user", "admin") \

.option("password", "Redshift.00") \

.save()