#Apache Spark

from pyspark.sql import SparkSession

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

import random

import time

spark = SparkSession.builder \

.appName("Experimental Spark Scalability") \

.config("spark.executor.memory", "2g") \

.config("spark.cores.max", 4) \

.getOrCreate()

def generate\_large\_dataset(num\_rows):

# Simulating large data by generating random values

data = [(i, f"Name\_{i}", random.randint(18, 65)) for i in range(num\_rows)]

return data

num\_rows = 10\*\*6 # You can increase this value for testing larger datasets

columns = ["id", "name", "age"]

df = spark.createDataFrame(generate\_large\_dataset(num\_rows), columns)

start\_time = time.time()

df\_transformed = df.withColumn(

"age\_group",

when(col("age") < 30, "Young")

.when(col("age") >= 30, "Mature")

.otherwise("Unknown")

)

df\_aggregated = df\_transformed.groupBy("age\_group").agg(count("id").alias("count"))

df\_aggregated.show()

end\_time = time.time()

print(f"Time taken for large dataset transformation: {end\_time - start\_time} seconds")

rdd = df.rdd

rdd2 = rdd.map(lambda x: (x[0], x[1].upper(), x[2]))

try:

rdd2.collect()

except Exception as e:

print(f"Error encountered (simulating failure): {e}")

df\_repartitioned = df.repartition(100) # Increase partitioning to handle large data

df\_repartitioned.cache()

df\_repartitioned\_transformed = df\_repartitioned.withColumn(

"adjusted\_age", col("age") + 5

)

df\_repartitioned\_transformed.show()

spark.stop()