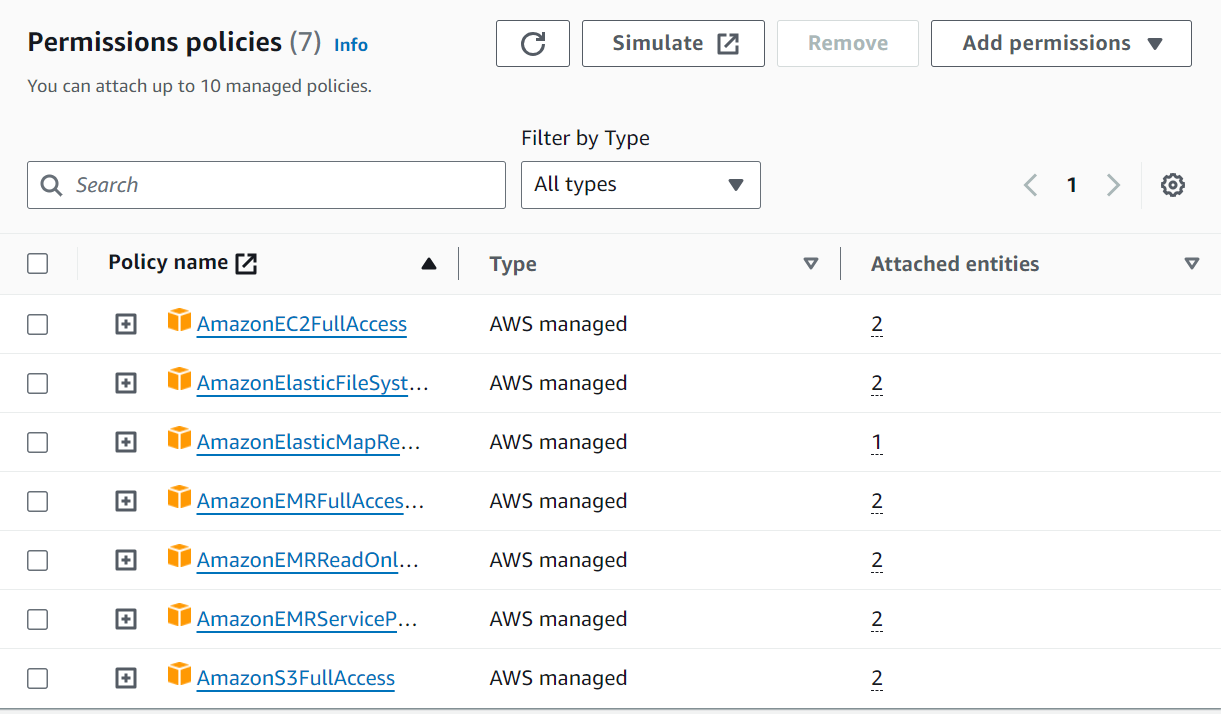
“Apache Spark™ is a unified analytics engine for large-scale data processing." It is an open-source, distributed computing engine, and it provides a productive environment for data analysis owing to its lightning speed and support for various libraries.

AmazonEMR-ServiceRole-20240706T212813



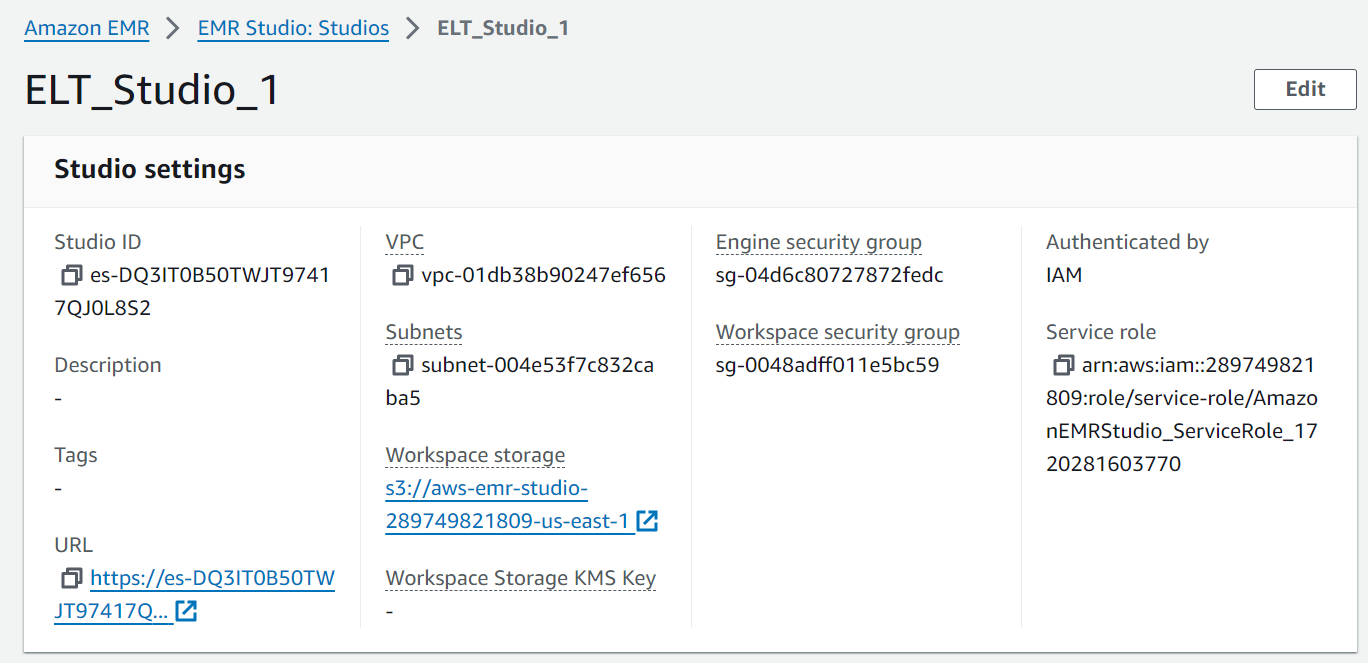
EMR CLUSER

ETL\_ASSIGNMENT

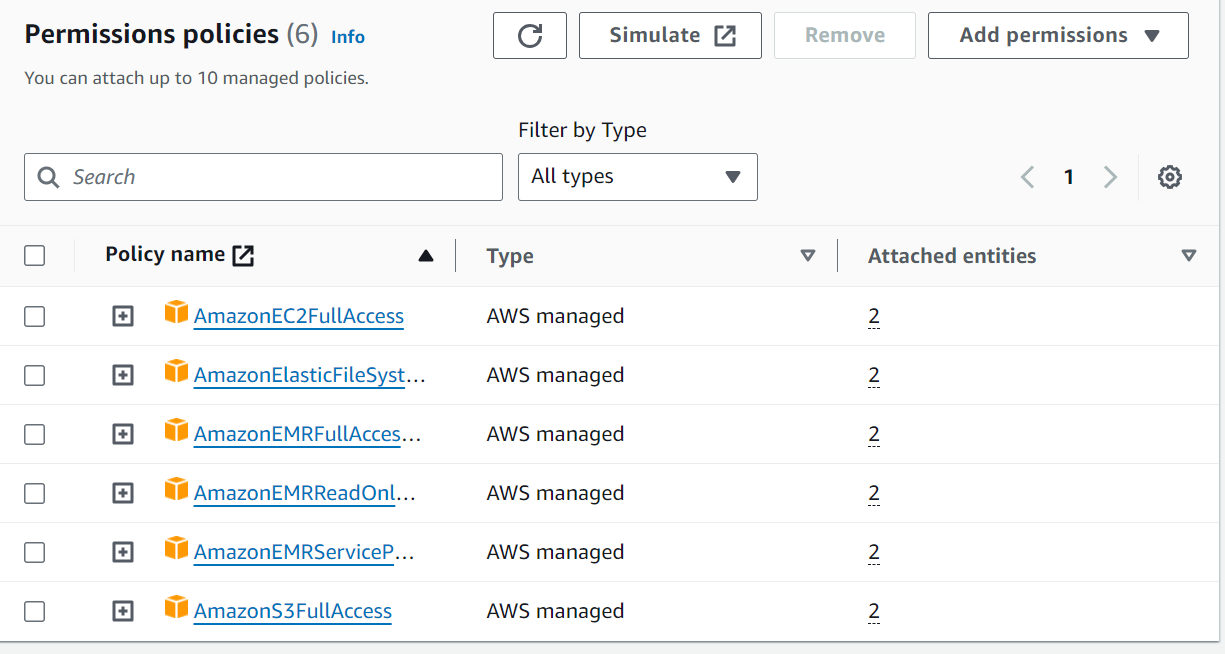
A screenshot of a computer

Description automatically generated

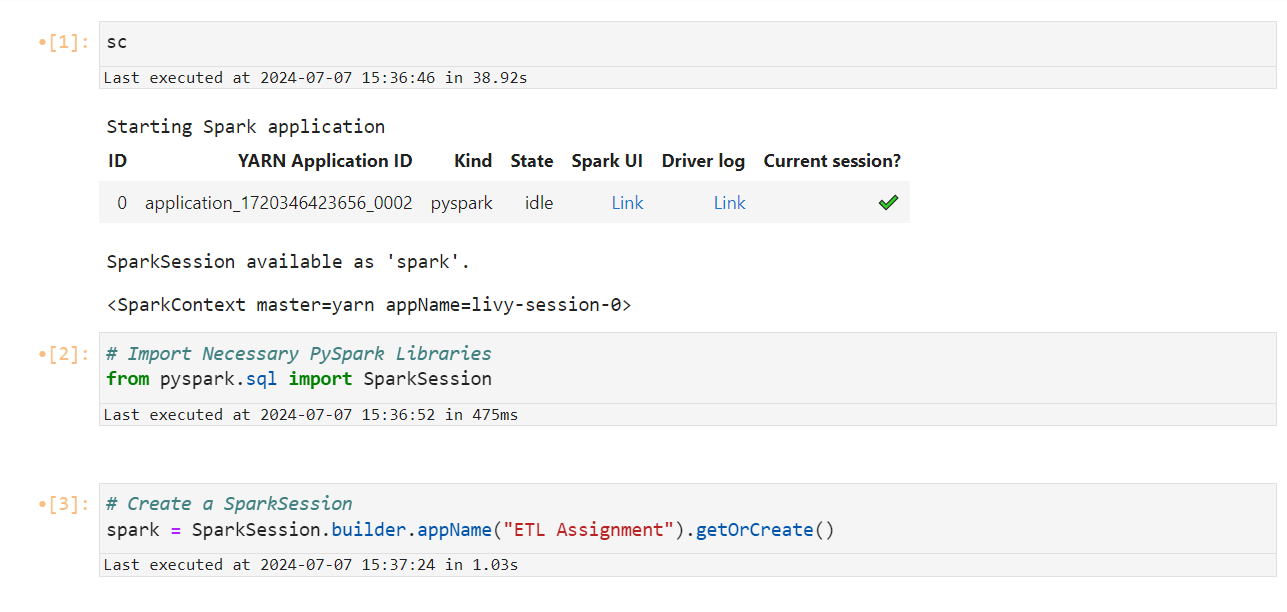
ETL STUDIO



AmazonEMRStudio\_ServiceRole\_1720281603770



Jupiter NoteBook



A screenshot of a computer program

Description automatically generated

# Start the Spark Application

sc

# Import Necessary PySpark Libraries

from pyspark.sql import SparkSession# Create a SparkSession

# Create a SparkSession

spark = SparkSession.builder.appName("ETL Assignment").getOrCreate()

# Read Data from Hadoop Cluster and Store it in a DataFrame

df = spark.read.format("csv").option("header", "false").option("inferSchema", "true").load("/user/hadoop/atm/part-m-00000")# Show DataFrame schema

# Count the Number of Rows Imported

df.count()

# Show First Few Rows of DataFrame

df.show()

# Show DataFrame Schema

df.printSchema()

# Import Necessary Libraries

from pyspark.sql.types import StructType, StructField, IntegerType, StringType, BooleanType, DoubleType, LongType

# Create the Columns for the Dataframe

df\_schema = StructType([StructField('year', IntegerType(), nullable = True),

StructField('month', StringType(), True),

StructField('day', IntegerType(), True),

StructField('weekday', StringType(), True),

StructField('hour', IntegerType(), True),

StructField('atm\_status', StringType(), True),

StructField('atm\_id', StringType(), True),

StructField('atm\_manufacturer', StringType(), True),

StructField('atm\_location', StringType(), True),

StructField('atm\_streetname', StringType(), True),

StructField('atm\_street\_number', IntegerType(), True),

StructField('atm\_zipcode', IntegerType(), True),

StructField('atm\_lat', DoubleType(), True),

StructField('atm\_lon', DoubleType(), True),

StructField('currency', StringType(), True),

StructField('card\_type', StringType(), True),

StructField('transaction\_amount', IntegerType(), True),

StructField('service', StringType(), True),

StructField('message\_code', StringType(), True),

StructField('message\_text', StringType(), True),

StructField('weather\_lat', DoubleType(), True),

StructField('weather\_lon', DoubleType(), True),

StructField('weather\_city\_id', IntegerType(), True),

StructField('weather\_city\_name', StringType(), True),

StructField('temp', DoubleType(), True),

StructField('pressure', IntegerType(), True),

StructField('humidity', IntegerType(), True),

StructField('wind\_speed', IntegerType(), True),

StructField('wind\_deg', IntegerType(), True),

StructField('rain\_3h', DoubleType(), True),

StructField('clouds\_all', IntegerType(), True),

StructField('weather\_id', IntegerType(), True),

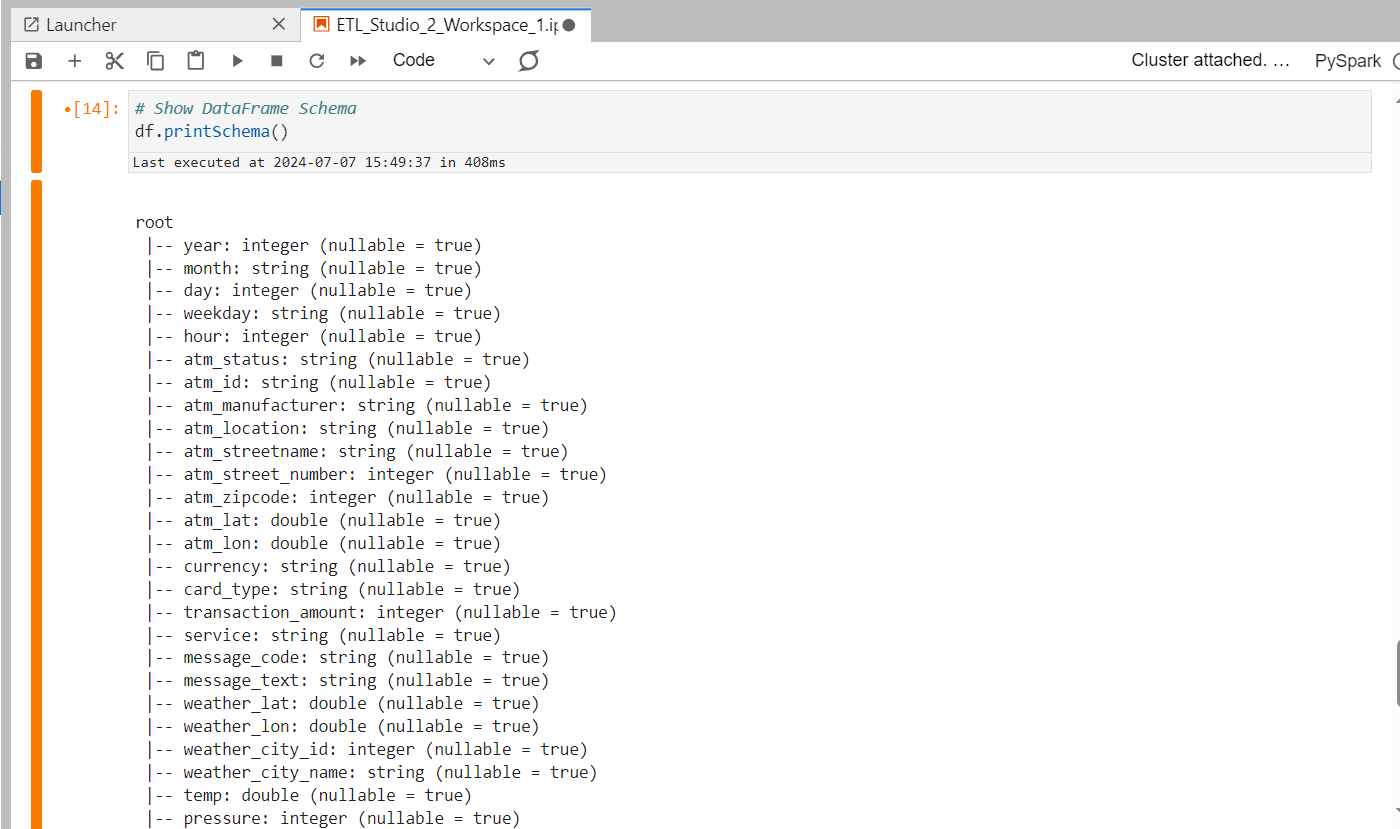
StructField('weather\_main', StringType(), True),

StructField('weather\_description', StringType(), True)])

# Load the Hadoop Data with Defined Schema

df = spark.read.format("csv").option("header", "false").schema(df\_schema).load("/user/hadoop/atm/part-m-00000")

**Data Processing with PySpark ● Check count after importing data into a dataframe Count of Records – 2468572**



# Drop Duplicates

df\_unique = df.dropDuplicates()

# Number of Records Post Removal of Duplicates

df\_unique.count()

rds\_sqoop\_hadoop\_spark\_df\_s3

# Specify the S3 bucket path where you want to save the files

s3\_bucket\_path = "s3a://rds-sqoop-hadoop-spark-df-s3/test/"

# Write DataFrame to S3 bucket as CSV files

df.write.csv(s3\_bucket\_path, mode='overwrite', header=True)