```
import json
import boto3
import time

glue = boto3.client('glue')
s3 = boto3.client('s3')
dynamodb = boto3.resource('dynamodb')
glue_spark_job = 'orders_processor_landing_to_staging'
audit_table = dynamodb.Table('orders-audit-table')

def lambda_handler(event, context):

    # Landing to Staging area ETL job

    start_job = 1

    for job_run in glue.get_job_runs(JobName = glue_spark_job)['JobRuns']:

        # print('The job run is ', job_run)
        if ((job_run['JobRunState'] == 'RUNNING') or (job_run['JobRunState'] == 'STARTING') or (job_run['J
obRunState'] == 'STOPPING') or (job_run['JobRunState'] == 'WAITING')):
            start_job = 0

    print('The value of start_job is ', start_job)

    if start_job == 1:

        try:

            start_job_response = glue.start_job_run(JobName = glue_spark_job)
            print('The start job response is ', start_job_response)

            get_job_response = glue.get_job_run(JobName = glue_spark_job, RunId = start_job_response['Jo
bRunId'])
            print('The get job response is ', get_job_response)

            job_run_state = get_job_response['JobRun']['JobRunState']
            print('The job run status is ', job_run_state)

            while (glue.get_job_run(JobName = glue_spark_job, RunId = start_job_response['JobRunId'])['Job
Run']['JobRunState'] != 'SUCCEEDED'):
                time.sleep(7)

            job_run_state = glue.get_job_run(JobName = glue_spark_job, RunId = start_job_response['JobRu
nId'])['JobRun']['JobRunState']
            print('The job run status is ', job_run_state)
            load_audit = 1

        except Exception as f:

            print('Unable to start the glue spark job. The exception is ', f)
            load_audit = 0

    else:
```

```
        print('Job is already running')
        load_audit = 0

    # Audit entries in DynamoDB
    # Once audit entries are made in DynamoDB, the streams / corresponding lambda function get triggere
d and the landing area gets cleaned up

    if load_audit == 1:

        items_to_add = []

        print('The event is ', event)
        for record in event['Records']:

            record_body = json.loads(record['body'])
            print('The record body is ', record_body)

            for s3_event in record_body['Records']:

                print('The s3 event is ', s3_event)

                bucket_name = s3_event['s3']['bucket']['name']
                file_name = s3_event['s3']['object']['key']
                file_size = str(s3_event['s3']['object']['size']/1000)
                file_etag = s3_event['s3']['object']['eTag']
                print('Name of the bucket is: ', bucket_name)
                print('Name of the file uploaded is: ', file_name)
                print('Size of the file uploaded in KB is: ', file_size)
                print('ETag of the file uploaded is: ', file_etag)

                item = {'file_name': file_name, 'file_etag': file_etag, 'file_size': file_size,  'pipeline_layer': 'landing
_area'}
                items_to_add.append(item)

        print('The final list is: ', items_to_add)

        try:

            with audit_table.batch_writer() as batch:
                for item in items_to_add:
                    batch.put_item(Item = item)
            print('Data loaded successfully in audit table for the landing layer')

        except Exception as e:

            print('Unable to complete audit entries for the landing layer. The exception is ', e)
```