# ETL Workflow Orchestration Project

## Project Overview

This project showcases a comprehensive implementation of an end-to-end data pipeline using AWS services to manage and analyze sales orders data efficiently. The pipeline integrates data extraction, transformation, loading (ETL), and analytics, emphasizing automation and scalability.Used Step function to enable data orchestration to coordinate and sequence multiple AWS Lambda function, AWS Glue jobs, allowing for the seamless flow and transformation of data through pipeline

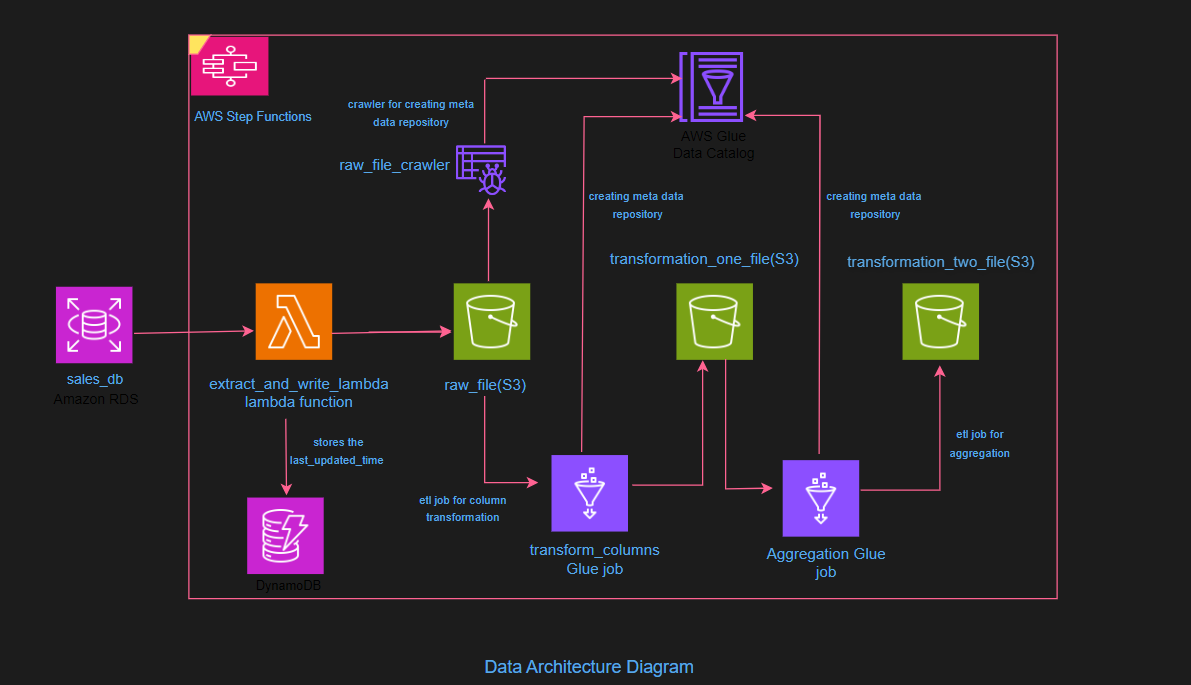


Fig:Data Architecture Diagram

## Data Collection and Incremental Load

**RDS Configuration**: Configured an Amazon RDS instance to store continuous sales order data. A specific 'last\_updated' timestamp column tracks the last modification time for each record.RDS Instance has been assigned the default VPC and subnet groups.

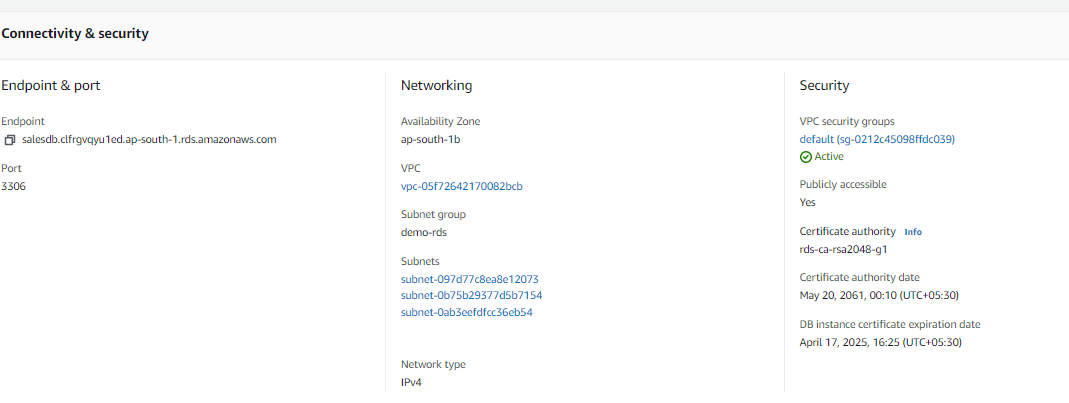


Fig:Connection and endpoint details of RDS instance

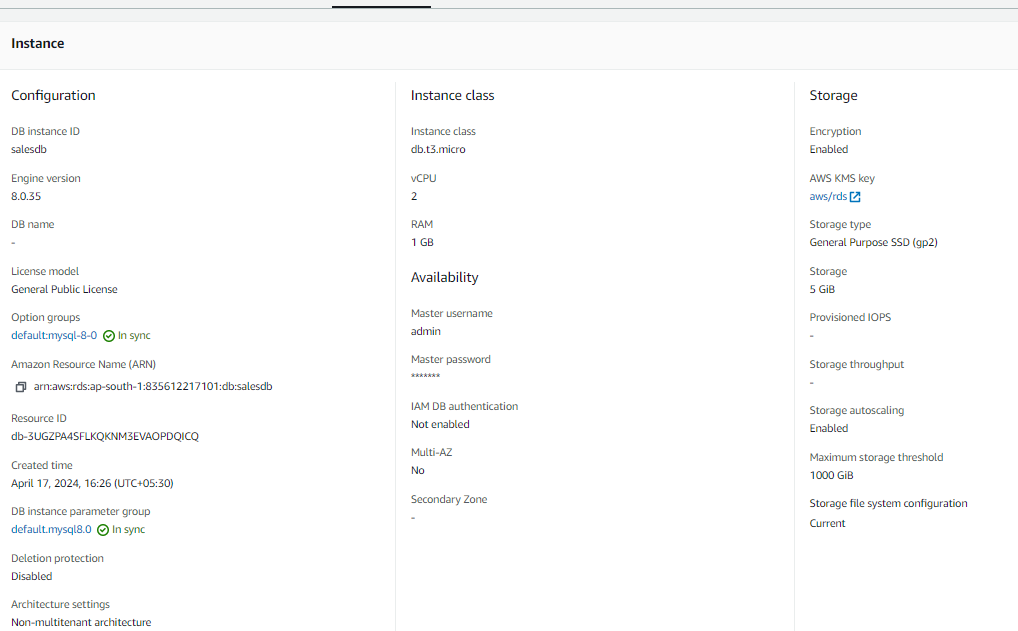


Fig:Configuration of RDS Instance

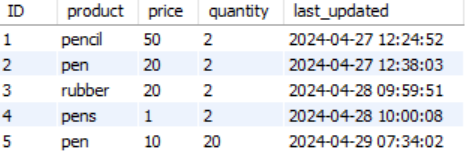


Fig:Sales orders table hosted on RDS

**Lambda for Incremental Extraction**: Developed an AWS Lambda function to extract only new and updated records based on the 'last\_updated' timestamp stored in Amazon DynamoDB, ensuring efficient data processing.This lambda has been assigned the same VPC as that of RDS Instance ,also suitable Security groups were assigned to this Lambda so that it can access RDS instance DB through port 3306.

Addition of custom Python module layer:

AWS Lambda has a limited set of pre-installed libraries. By adding custom modules, can extend the functionality of Lambda functions to include additional libraries and dependencies required for their specific use case so that modules like pymysql could be used in lambda environment.Lambda function needs to connect to a MySQL database, for this will need pymysql to handle the database operations.

The lambda reads from RDS using boto3 client and stores the records in a location in S3 .Also it stores the latest time\_stamp from the records using the last\_updated column.this is stored in dynamoDb.So that next time when the lambda function gets invoked ,it only processes the records added after the that timestamp.This helps to read and store only incremental records .

code link:<https://github.com/gojira12345/ETL_Worflow_Orchestration_Project/blob/main/lambda_for_extraction_from_RDS_and_writing_to_S3.txt>

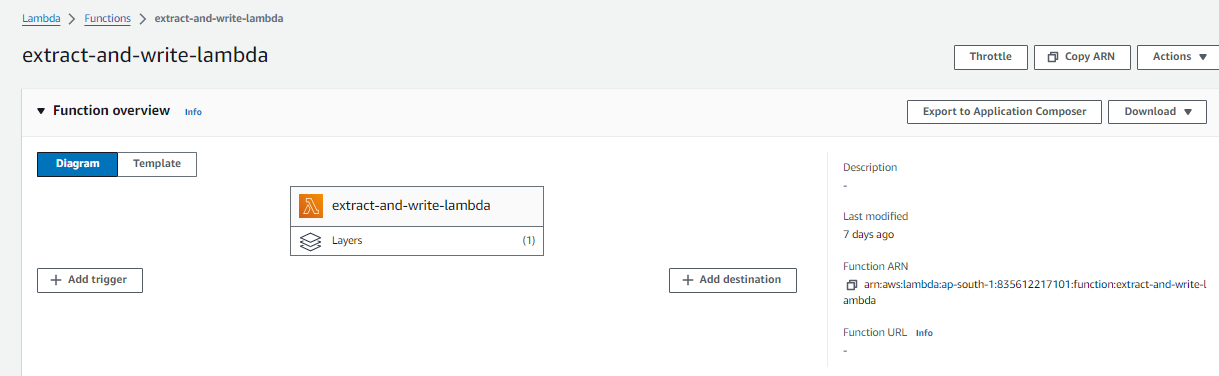


Fig:Lambda function

## Data Storage and Incremental Tracking

**S3 for Data Storage**: Configured AWS S3 to store incremental data extracts securely.

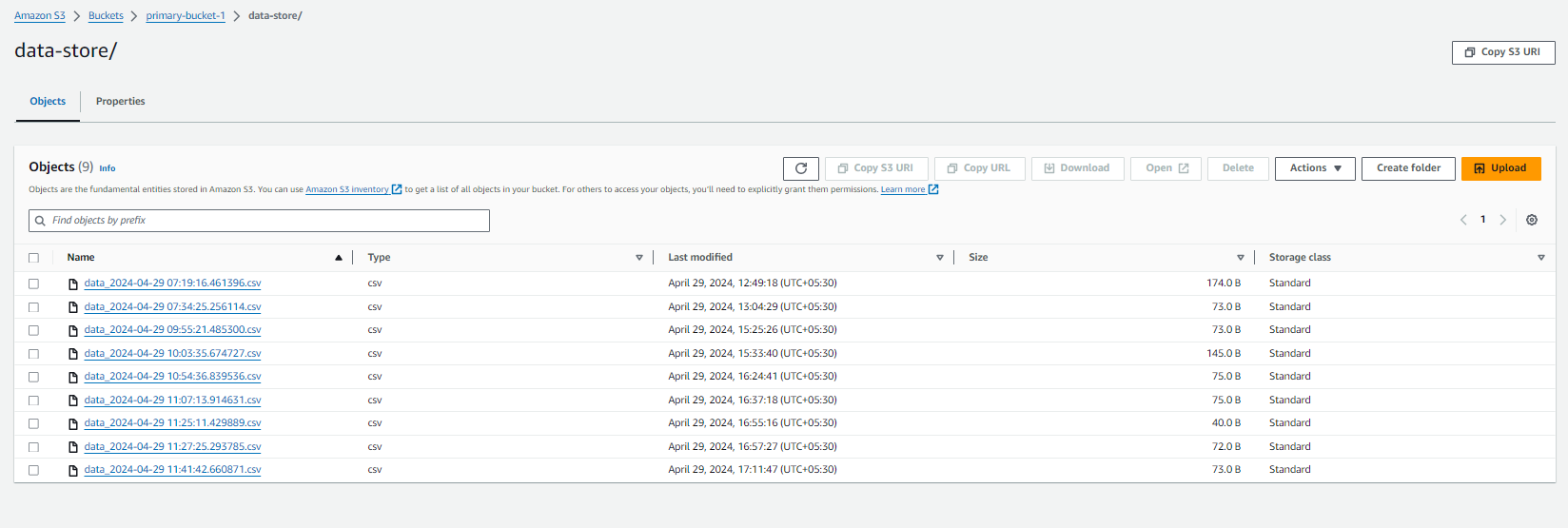


Fig:Raw file storage in S3

**DynamoDB for State Management**: Used DynamoDB to maintain the state of data extracts, storing the most recent 'last\_updated' timestamp which the Lambda function references.

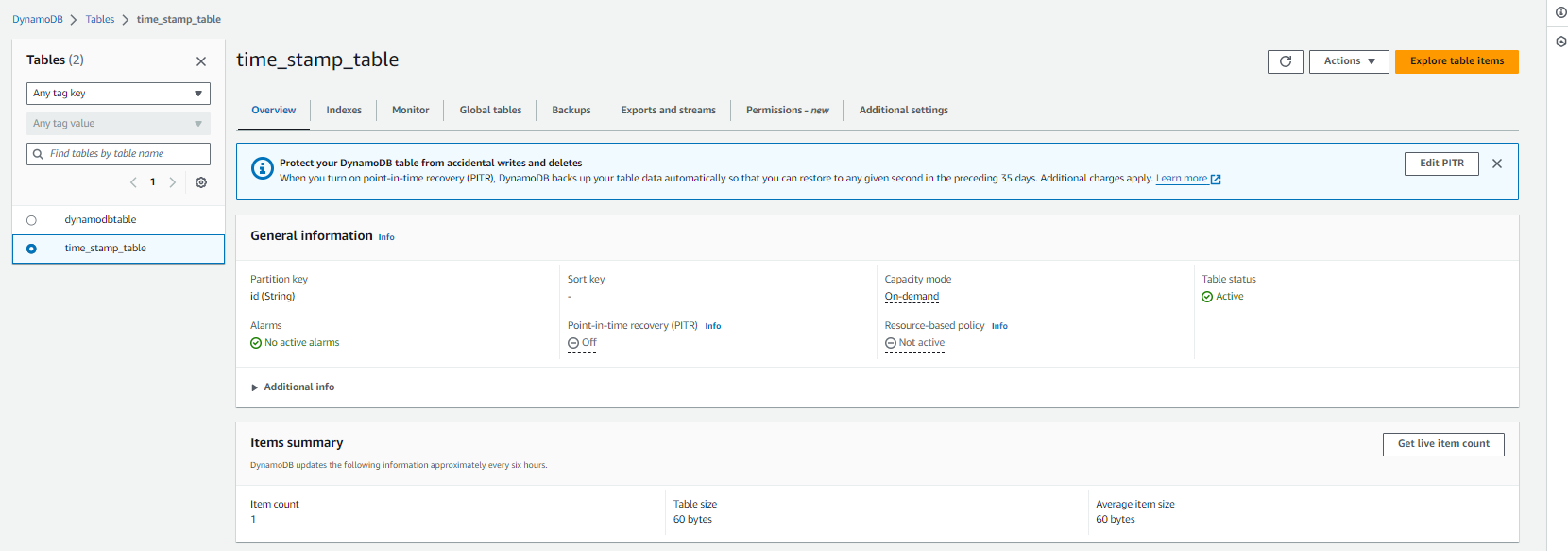


Fig:Dynamodb table that for storing the timestamp value

## Data Processing and Orchestration

**Glue for ETL and Cataloging**:

1.Transform Columns ETL job:

Configured AWS Glue ETL job to perform schema transformations and storing results back in S3. Enabled **job bookmarks** in Glue to track and process only new or modified data, preventing duplicates.

code link:<https://github.com/gojira12345/ETL_Worflow_Orchestration_Project/blob/main/column_transfomartion_etl_script.txt>

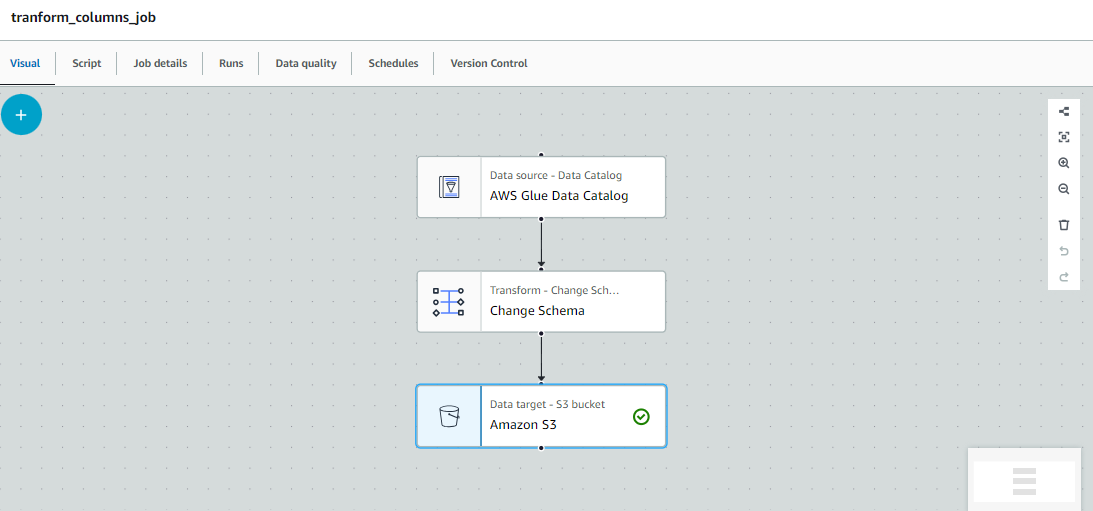


Fig:Column Transformation ETL job

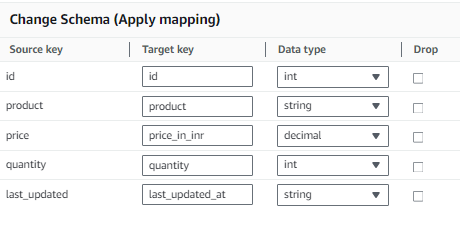


Fig:Schema Change transformation Applied

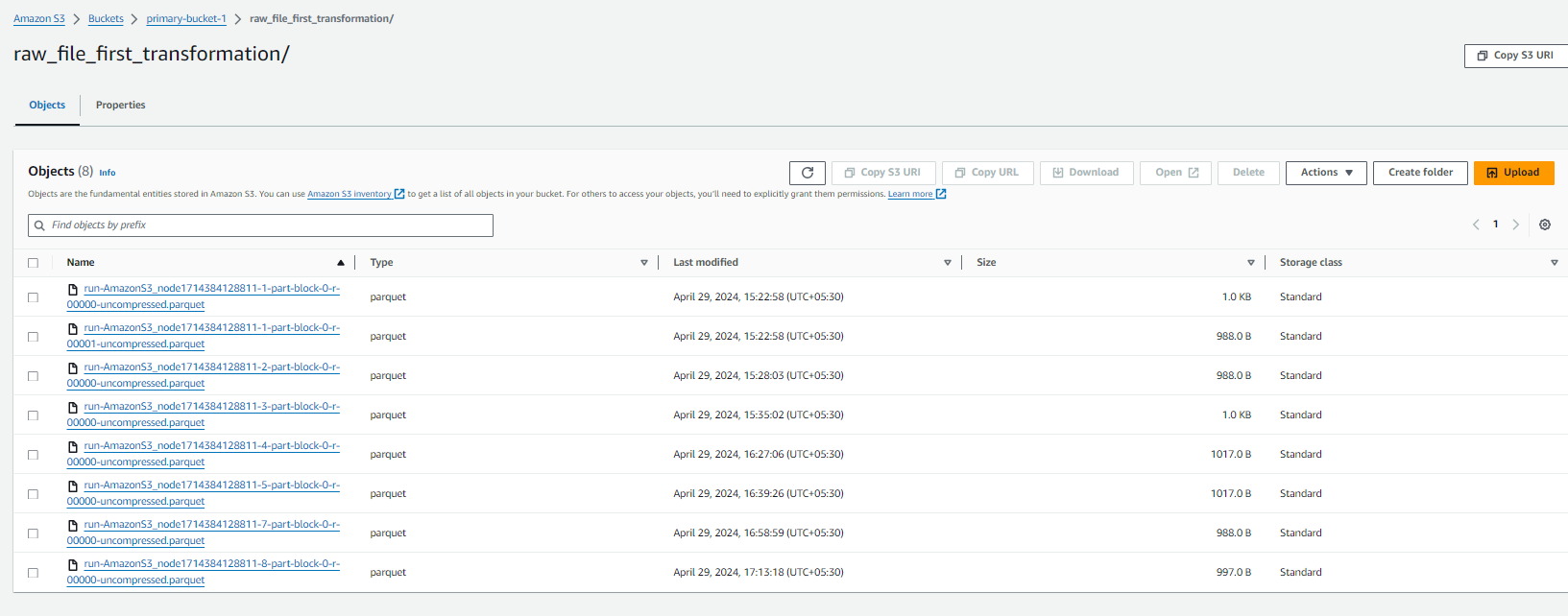


Fig:S3 location after first transformation

2.Aggregation ETL job

Configured AWS Glue ETL job to aggregate transformed data, providing summarized datasets ready for analytics .the data is then stored into S3.

code link:

<https://github.com/gojira12345/ETL_Worflow_Orchestration_Project/blob/main/aggregation_etl_script.txt>

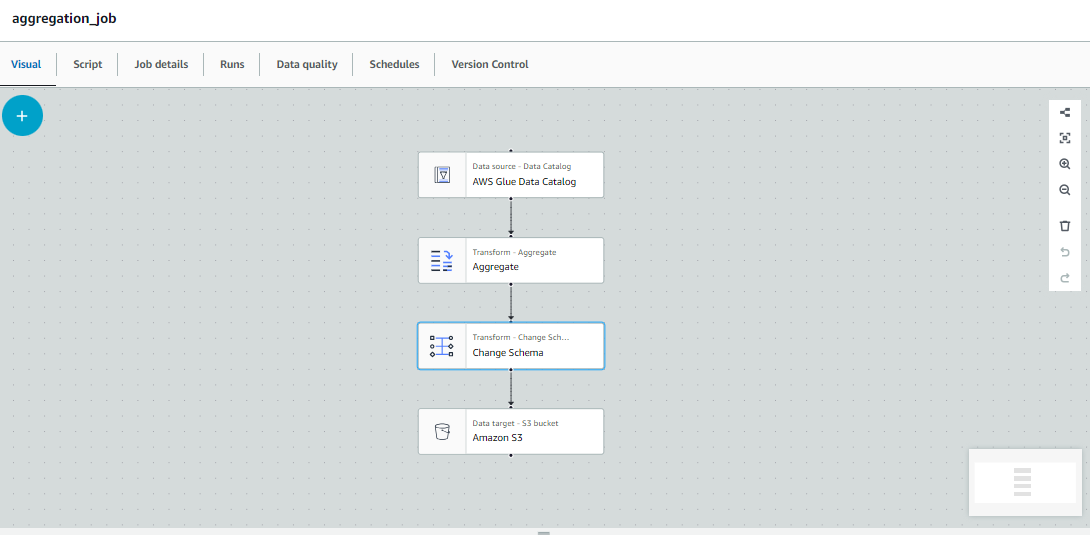


Fig:Aggregation ETL job

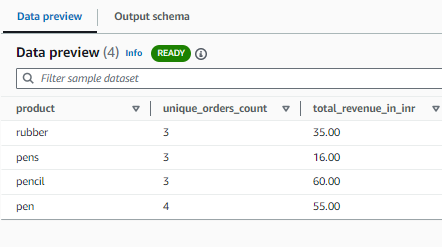


Fig:Aggregated data after transformation

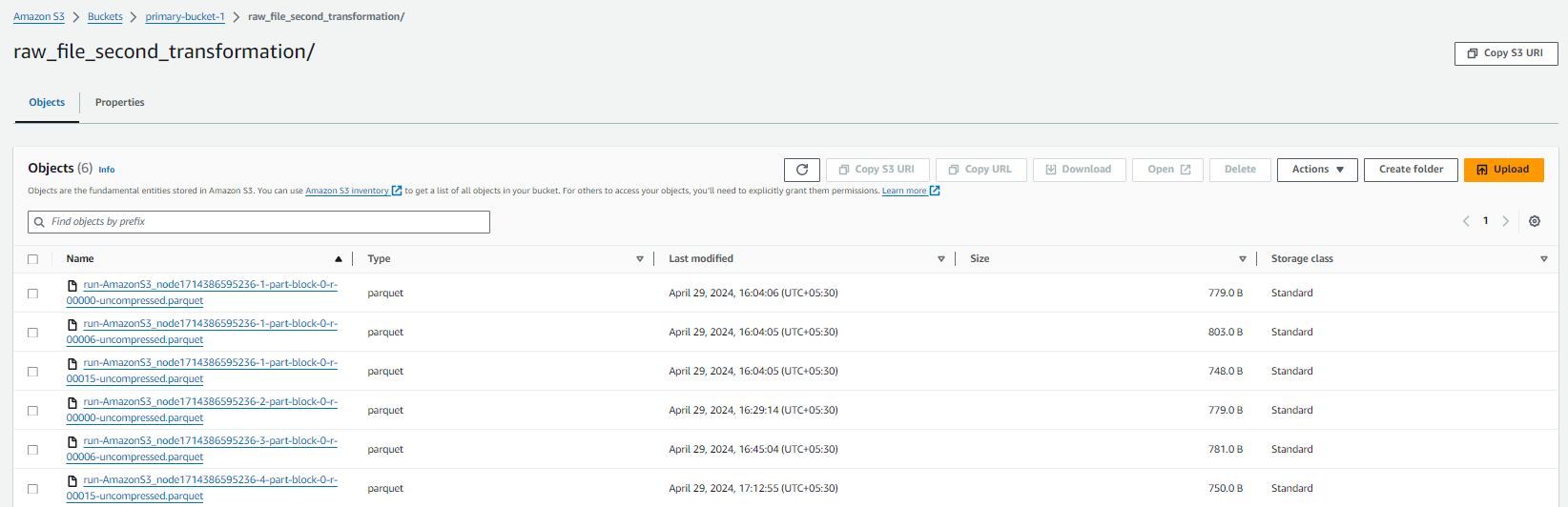


Fig:S3 location after second transformation

**AWS Step Functions for Orchestration**: Implemented AWS Step Functions to manage the ETL workflow, coordinating between the Lambda function, AWS Glue jobs, and data storage processes. This orchestration layer ensures robust management of the pipeline's execution, handling task failures, retries, and state transitions seamlessly.Added all the steps right from lambda function execution to last etl job as states to Step Function State Machine.This way was able to orchestrate the entire workflow.For now ,have setup up a Cloudwatch Event rule that will invoke the Step function at regular intervals.

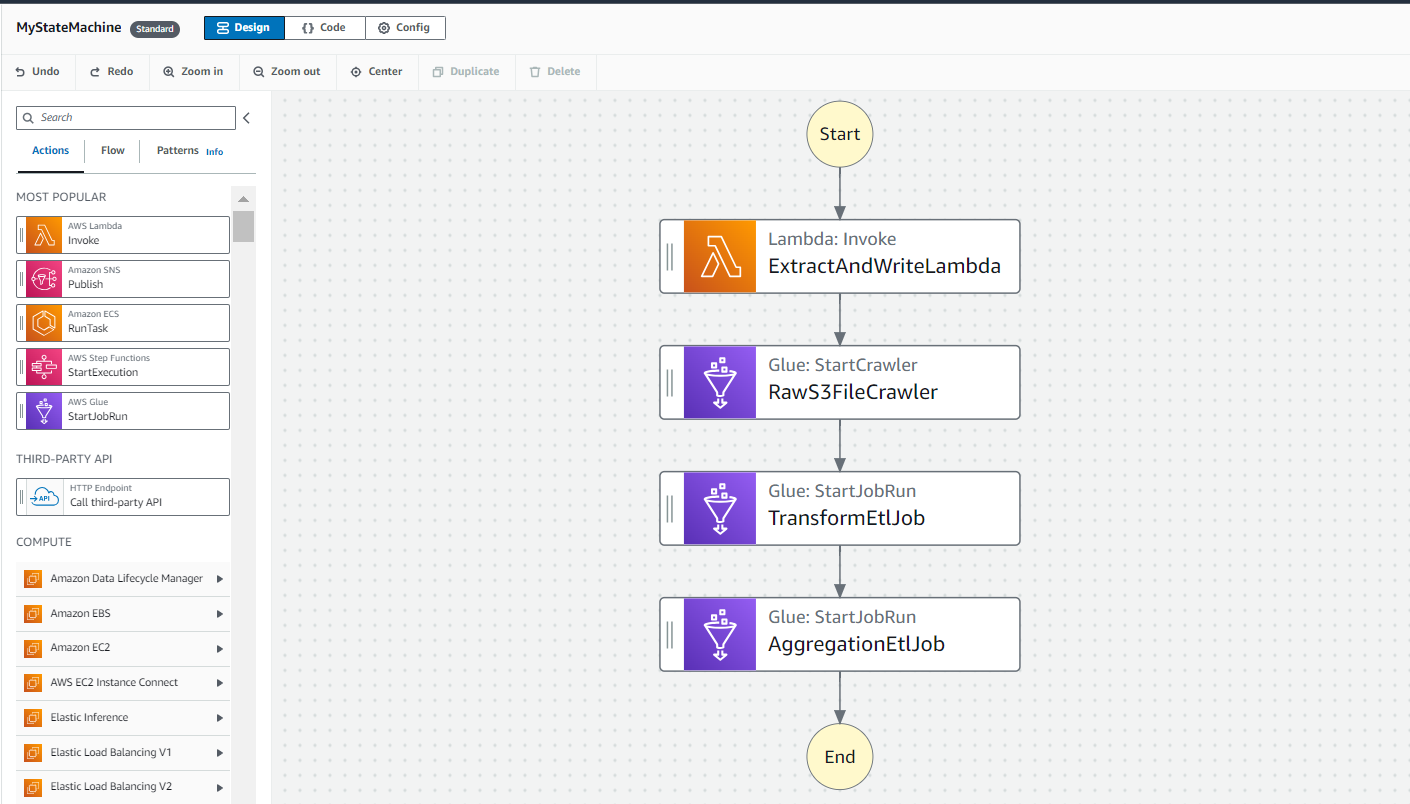


Fig:Step function

code link:

<https://github.com/gojira12345/ETL_Worflow_Orchestration_Project/blob/main/step_function_workflow_orchestration_script.txt>

## Security and Compliance

Ensured all components comply with organizational security policies, including configuring VPC security, proper IAM roles, and ensuring encrypted data transfers.

## Project Impact

The project significantly reduced data processing time and costs, improved real-time data availability and efficiency, and enhanced analytical capabilities.

## Skills and Technologies

AWS RDS, Lambda, DynamoDB, S3, AWS Glue, Step Functions, security practices, data visualization integration.