E-Commerce Lakehouse ETL Pipeline Documentation

Overview

This project implements a scalable ETL (Extract, Transform, Load) pipeline for an e-commerce lakehouse architecture, designed to process transactional data (products, orders, and order items) and store it in a Delta Lake format on AWS S3. The system is event-driven, detecting new data dropped into an S3 bucket, queuing notifications via SQS, and using Lambda to trigger an AWS Step Functions state machine for orchestrated processing.

The pipeline ensures data quality through validation, deduplication, and referential integrity checks, transforming raw CSV data into a structured lakehouse format for analytics. After processing, the system updates the AWS Glue Data Catalog, validates data presence using Athena queries, and notifies stakeholders via SNS.

System Architecture

The pipeline leverages AWS services to create a serverless, event-driven workflow:

1. **S3 (Simple Storage Service)**:
   - Stores raw data (CSV files) in the s3://<bucket-name>/raw/ prefix, organized into subfolders: products/, orders/, and order_items/.
   - Processed data is stored in s3://<bucket-name>/lakehouse-dwh/ as Delta tables.
   - Rejected records (failed validation) are written to s3://<bucket-name>/rejected/.
   - Archived data (post-processing) is moved to s3://<bucket-name>/archived/.
2. **S3 Event Notification**:
   - Configured on the <bucket-name> bucket to detect new files in the raw/ prefix.
   - Triggers an event notification for each new file, sending the event to an SQS queue.
3. **SQS (Simple Queue Service)**:
   - Queues S3 event notifications to ensure reliable processing of new data arrivals.

- Acts as a buffer between S3 events and the Lambda function, allowing for retries and decoupling.

4. **Lambda Function (Trigger)**:
   - Polls the SQS queue for new S3 event messages.
   - Checks if all required datasets (products, orders, order_items) are available in the raw/ prefix before proceeding.
   - If all data is present, triggers the AWS Step Functions state machine to start the ETL process.
5. **AWS Step Functions**:
   - Orchestrates the ETL pipeline as a state machine (EcommerceETLPipeline).
   - Manages the execution of AWS Glue jobs, a Lambda function for archiving, a Glue Crawler to update the Data Catalog, Athena queries for validation, and an SNS notification for alerting stakeholders.
   - Ensures sequential execution and error handling.
6. **AWS Glue Jobs**:
   - Three Glue jobs (product_etl, orders_etl, order_items_etl) process the respective datasets.
   - Each job:
     - Reads raw CSV data from S3.
     - Validates data (schema enforcement, null checks, timestamp validation, referential integrity for order_items).
     - Deduplicates records based on primary keys.
     - Writes the transformed data to Delta tables in s3://<bucket-name>/lakehouse-dwh/, partitioned appropriately.
     - Writes rejected records to s3://<bucket-name>/rejected/.
7. **Lambda Function (Archiver)**:
   - Invoked by Step Functions after all Glue jobs complete.
   - Moves processed files from raw/ to archived/ in S3.
8. **AWS Glue Data Catalog and Crawler**:

  - A Glue Data Catalog database (ecommerce_dwh) stores metadata for the Delta tables in lakehouse-dwh/.
  - A Glue Crawler (ecommerce-lakehouse-crawler) updates the Data Catalog after data is processed.

9. **Amazon Athena**:
   - Executes queries to validate the presence of data in the Delta tables after the Data Catalog is updated.
10. **SNS (Simple Notification Service)**:
    - Sends notifications to stakeholders upon pipeline success or failure.

Workflow

1. **Data Ingestion**:
   - New CSV files are uploaded to s3://<bucket-name>/raw/products/, raw/orders/, or raw/order_items/.
   - S3 event notifications are triggered for each file upload.
2. **Event Queuing**:
   - S3 events are sent to an SQS queue (ecommerce-etl-queue).
   - The queue ensures events are not lost and can be processed reliably.
3. **Data Availability Check**:
   - A Lambda function (CheckDataAvailability) polls the SQS queue.
   - For each event, it checks if at least one file exists in each of the required prefixes (raw/products/, raw/orders/, raw/order_items/).
   - If all datasets are present, it triggers the Step Functions state machine.
4. **ETL Processing (Step Functions)**:
   - The state machine (EcommerceETLPipeline) executes the following steps:
     - **Step 1: Process Products**:
       - Runs the product_etl Glue job.
       - Reads CSVs from raw/products/, validates, deduplicates by product_id, and writes to lakehouse-dwh/products/ (partitioned by department_id).

- **Step 2: Process Orders**:
  - Runs the orders_etl Glue job.
  - Reads CSVs from raw/orders/, validates, deduplicates by order_id, and writes to lakehouse-dwh/orders/ (partitioned by date).
- **Step 3: Process Order Items**:
  - Runs the order_items_etl Glue job.
  - Reads CSVs from raw/order_items/, validates, checks referential integrity (against orders and products), deduplicates by id, and writes to lakehouse-dwh/order_items/ (partitioned by date).
- **Step 4: Archive Files**:
  - Invokes the ArchiveETLFiles Lambda function.
  - Moves processed files from raw/ to archived/.
- **Step 5: Update Data Catalog**:
  - Runs the ecommerce-lakehouse-crawler Glue Crawler.
  - Updates the ecommerce_dwh Data Catalog database with metadata for the Delta tables in lakehouse-dwh/.
- **Step 6: Validate Data Presence**:
  - Executes Athena queries to verify data presence in the products, orders, and order_items tables.
  - Example queries:
    - SELECT COUNT(*) FROM ecommerce_dwh.products
    - SELECT COUNT(*) FROM ecommerce_dwh.orders
    - SELECT COUNT(*) FROM ecommerce_dwh.order_items
  - Fails the pipeline if any table is empty.
- **Step 7: Notify Stakeholders**:
  - Publishes a message to an SNS topic (ecommerce-etl-notifications).

        - Sends a success message if all steps complete, or a failure message with error details if any step fails.

5. **Error Handling**:
    - Each Glue job logs errors and writes rejected records to the rejected/ prefix.
    - Step Functions retries failed steps (configurable) and logs execution status.
    - SQS supports message retries if the Lambda function fails to process an event.
    - SNS notifications alert stakeholders of failures with details from the state machine execution.

Data Schemas

Products

- **Schema**:
    - product_id (Integer, not null)
    - department_id (Integer, not null)
    - department (String, not null)
    - product_name (String, not null)
- **Primary Key**: product_id
- **Partitioning**: department_id

Orders

- **Schema**:
    - order_num (Integer, not null)
    - order_id (Integer, not null)
    - user_id (Integer, not null)
    - order_timestamp (Timestamp, not null)
    - total_amount (Double, not null)
    - date (String, not null)

- **Primary Key**: order_id
- **Partitioning**: date

Order Items

- **Schema**:
  - id (Integer, not null)
  - order_id (Integer, not null)
  - user_id (Integer, not null)
  - days_since_prior_order (Integer, nullable)
  - product_id (Integer, not null)
  - add_to_cart_order (Integer, not null)
  - reordered (Integer, not null)
  - order_timestamp (Timestamp, not null)
  - date (String, not null)
- **Primary Key**: id
- **Partitioning**: date
- **Referential Integrity**:
  - order_id must exist in the orders table.
  - product_id must exist in the products table.

Setup Instructions

Prerequisites

- AWS Account with permissions to create S3 buckets, SQS queues, Lambda functions, Glue jobs, Glue Crawlers, Athena, SNS topics, and Step Functions.
- AWS CLI configured with appropriate credentials.
- Python 3.9 and required dependencies (listed in code/requirements.txt).

Steps

1. **Create S3 Bucket**:
   - Create a bucket named <bucket-name>.

   - Set up the following prefixes: raw/, lakehouse-dwh/, rejected/, archived/.

2. **Set Up SQS Queue**:
   - Create an SQS queue named ecommerce-etl-queue.
   - Configure the queue to receive S3 event notifications.
3. **Configure S3 Event Notification**:
   - In the S3 bucket (<bucket-name>), set up an event notification for the raw/ prefix.
   - Filter for *.csv files and send events to the ecommerce-etl-queue SQS queue.
4. **Set Up AWS Glue Data Catalog**:
   - Create a Glue Data Catalog database named ecommerce_dwh:
   - aws glue create-database --database-input '{"Name":"ecommerce_dwh"}'
5. **Set Up Glue Crawler**:
   - Create a Glue Crawler named ecommerce-lakehouse-crawler:
     - Target the S3 path s3://<bucket-name>/lakehouse-dwh/.
     - Set the database to ecommerce_dwh.
     - Configure an IAM role with permissions to access S3 and Glue.
     - Example CLI command:
     - aws glue create-crawler --name ecommerce-lakehouse-crawler \
     -   --role arn:aws:iam::<account-id>:role/GlueCrawlerRole \
     -   --database-name ecommerce_dwh \
     -   --targets '{"S3Targets":[{"Path":"s3://<bucket-name>/lakehouse-dwh/"}]}'
6. **Set Up SNS Topic**:
   - Create an SNS topic named ecommerce-etl-notifications:
   - aws sns create-topic --name ecommerce-etl-notifications
   - Subscribe stakeholders (e.g., email addresses) to the topic:

   - aws sns subscribe --topic-arn arn:aws:sns:<region>:<account-id>:ecommerce-etl-notifications \
   - --protocol email --notification-endpoint <stakeholder-email>
   - Confirm subscriptions via email to enable notifications.
7. **Deploy Lambda Function (CheckDataAvailability)**:
   - Write a Lambda function to poll the SQS queue and check for data availability:
     - List objects in raw/products/, raw/orders/, and raw/order_items/.
     - If at least one file exists in each prefix, trigger the Step Functions state machine.
   - Deploy the Lambda function with an SQS trigger and permissions to invoke Step Functions.
8. **Deploy Glue Jobs**:
   - Deploy the three Glue jobs (product_etl, orders_etl, order_items_etl) from code/glue_scripts/.
   - Ensure the validation.py utility is uploaded to s3://<bucket-name>/etl-scripts/glue_jobs/utils/.
9. **Deploy Lambda Function (ArchiveETLFiles)**:
   - Deploy the lambda_function.py as a Lambda function named ArchiveETLFiles.
   - Ensure it has permissions to read/write to S3.
10. **Set Up Step Functions**:
    - Create a state machine named EcommerceETLPipeline using the definition in code/step_functions.json.
    - Update the state machine definition to include:
      - Running the Glue Crawler (ecommerce-lakehouse-crawler) after archiving.
      - Executing Athena queries to validate data presence.
      - Sending SNS notifications on success or failure.
    - Ensure the state machine role has permissions to:
      - Run Glue jobs and crawlers.

        - Execute Athena queries.
        - Publish to the SNS topic.

11. **Test the Pipeline**:
    - Upload sample CSV files to s3://<bucket-name>/raw/products/, raw/orders/, and raw/order_items/.
    - Verify that the SQS queue receives events, the Lambda function triggers the state machine, and the pipeline processes the data into lakehouse-dwh/.
    - Confirm that the Data Catalog is updated, Athena queries validate data, and SNS notifications are sent.

Testing

Unit Tests

- Located in code/tests/unit/.
- Tests the core functionality of each module:
    - test_lambda_function.py: Tests the file archiving logic.
    - test_product_etl.py, test_orders_etl.py, test_order_items_etl.py: Tests deduplication logic.
    - test_validation.py: Tests data validation logic.

Integration Tests

- Located in code/tests/integration/.
- test_etl_pipeline.py: Tests the end-to-end ETL pipeline using mocked S3 and Delta tables.

Running Tests

- Install dependencies: pip install -r code/requirements.txt.
- Run tests: pytest code/tests/unit/ --verbose and pytest code/tests/integration/ --verbose.

CI/CD Pipeline

The project includes a GitHub Actions workflow for continuous integration and deployment:

- **CI**: Runs unit and integration tests on every push or pull request to the main branch.
- **CD**: Deploys the Glue jobs, Lambda function, and Step Functions state machine to AWS on pushes to main.

Troubleshooting

- **S3 Events Not Triggering**:
  - Verify the S3 event notification configuration and ensure the SQS queue has permissions to receive events.
- **Lambda Not Triggering Step Functions**:
  - Check the Lambda function logs in CloudWatch for errors.
  - Ensure the Lambda role has permissions to invoke Step Functions.
- **Glue Jobs Failing**:
  - Check Glue job logs for validation or data issues.
  - Verify the S3 paths and permissions for reading/writing data.
- **Glue Crawler Not Updating Data Catalog**:
  - Check the crawler logs for errors.
  - Ensure the crawler role has permissions to access S3 and the Data Catalog.
- **Athena Queries Failing**:
  - Verify the Data Catalog database (ecommerce_dwh) has the correct table metadata.
  - Ensure the Athena query execution role has permissions to read the Data Catalog and S3.
- **SNS Notifications Not Sent**:
  - Confirm subscriptions to the ecommerce-etl-notifications topic.
  - Check the Step Functions role for permissions to publish to SNS.
- **Tests Failing**:

- Resolve import errors in the GitHub Actions workflow by fixing the test execution environment.