**Data Architecture Document**

**1. Introduction**

This document outlines the data architecture for processing and storing data in an AWS environment, utilizing a landing zone, ETL processes with EMR, and a data warehouse with Amazon Redshift.

**2. Architecture Overview**

**2.1 Data Flow Diagram**



**2.2 Components**

* **Landing Zone**: An AWS S3 bucket that receives data every hour.
* **ETL Process**: AWS EMR for transforming and loading data.
* **Silver Layer**: An AWS S3 bucket that stores processed data as CSV files.
* **Data Warehouse**: Amazon Redshift for analytical querying.

**3. Detailed Components**

**3.1 Landing Zone**

* **Bucket Name**: landingzone48
* **Data Ingestion Frequency**: Every hour
* **Data Structure**: The bucket contains directories for:
  + source\_data/
  + archived/
  + failedfiles/
* **Data Schema**:
  + event\_time
  + event\_type
  + product\_id
  + category\_id
  + category\_code
  + brand
  + price
  + user\_id
  + user\_session

**3.2 ETL Process**

* **Service Used**: AWS EMR
* **Steps**:
  1. **Data Extraction**: Read raw data from the landing zone.
  2. **Data Transformation**:
     + Clean and preprocess data.
     + Convert data into CSV format for the Silver Layer.
  3. **Data Loading**: Write transformed CSV files to the Silver Layer.
* **Output Bucket**:
  1. **Bucket Name**: silverlayer

**3.3 Silver Layer**

* **Bucket Name**: silverlayer
* **Purpose**: Store transformed CSV files ready for loading into Redshift.
* **Data Structure**: Contains CSV files corresponding to the following Redshift tables:
  + **dim\_user**:

sql

Copy code

CREATE TABLE dim\_user (

user\_id INT PRIMARY KEY,

first\_transaction\_date TIMESTAMP

);

* + **dim\_product**:

sql

Copy code

CREATE TABLE dim\_product (

product\_id INT PRIMARY KEY,

brand VARCHAR(255)

);

* + **dim\_category**:

sql

Copy code

CREATE TABLE dim\_category (

category\_id INT PRIMARY KEY,

category\_code VARCHAR(255)

);

* + **fact\_events**:

sql

Copy code

CREATE TABLE fact\_events (

event\_time TIMESTAMP,

event\_type VARCHAR(50),

category\_id INT,

product\_id INT,

user\_id INT,

price DECIMAL(10, 2),

user\_session VARCHAR(255),

PRIMARY KEY (event\_time, user\_session), -- Composite primary key for uniqueness

FOREIGN KEY (category\_id) REFERENCES dim\_category(category\_id),

FOREIGN KEY (product\_id) REFERENCES dim\_product(product\_id),

FOREIGN KEY (user\_id) REFERENCES dim\_user(user\_id)

);

**3.4 Data Warehouse**

* **Service Used**: Amazon Redshift
* **Database Name**: databasecapstone
* **Data Loading**:
  + Load data from the Silver Layer into respective tables in Redshift using COPY commands.
  + Each CSV file corresponds to a specific table.

**4. ETL Script: From Landing Zone to Silver Layer**

**Overview**

This script processes CSV files from an S3 bucket (landingzone48), transforms the data using PySpark, and stores the cleaned data as CSV files in a silver layer S3 bucket.

**Key Steps**

1. **Initialization**:
   * Import necessary libraries (Boto3 for S3 interactions and PySpark for data processing).
   * Create a Spark session and initialize the Boto3 S3 client.
2. **Define S3 Paths**:
   * Specify input (landing zone) and output (silver layer) S3 paths for categories, products, users, and events.
3. **Data Processing**:
   * List files in the source\_data/ directory of the landing zone.
   * For each file:
     + Read the CSV file into a PySpark DataFrame.
     + Convert the event\_time column from UTC to IST (Indian Standard Time).
     + Extract unique categories and products, filling missing values with default labels ("UnknownCategory" and "UnknownBrand").
     + Aggregate user data to determine the first transaction time.
4. **Write Results**:
   * Write the transformed DataFrames (categories, products, users, and events) to the specified silver layer S3 paths in CSV format.
5. **File Management**:
   * Move successfully processed files to an archived/ folder and delete them from the source.
   * In case of an error during processing, move the problematic file to a failedfiles/ folder.
6. **Error Handling**:
   * The script includes a try-except block to handle errors, ensuring files are moved appropriately based on processing success.
7. **Cleanup**:
   * Stop the Spark session at the end of the process.

**5. ETL Script: From Silver Layer to Redshift**

**Overview**

This script transfers data from the silver layer (CSV files) into Amazon Redshift, enabling analytical querying.

**Key Steps**

1. **Initialization**:
   * Import necessary libraries (Boto3 for S3 interactions and PySpark for data processing).
   * Create a Spark session and initialize the Boto3 S3 client.
2. **Define S3 Paths**:
   * Specify the source (silver layer) and target (Redshift) paths.
3. **Data Loading**:
   * For each table in Redshift:
     + Read the corresponding CSV file from the silver layer into a PySpark DataFrame.
     + Perform any necessary transformations (if applicable).
     + Write the DataFrame to Redshift using the appropriate JDBC connection and the write method.
4. **Error Handling**:
   * Include error handling to manage any issues that arise during data loading.
5. **Cleanup**:
   * Stop the Spark session at the end of the process.

**6. Scheduling and Automation**

* **Trigger**: Scheduled AWS Lambda function or Amazon CloudWatch Events to start EMR jobs every hour after data ingestion.
* **Monitoring**: Use AWS CloudWatch for monitoring ETL jobs and alerting for failures.

**7. Security**

* **Data Encryption**: Enable server-side encryption for S3 buckets.
* **Access Control**: Use IAM roles to manage permissions for EMR and Redshift.

**8. Conclusion**

This data architecture enables efficient data processing from ingestion to analytical querying, leveraging AWS services for scalability and reliability.