Intelligent Customer Review Analyzer for E-Commerce

- **Project Overview :-**

  This project implements an end-to-end, cloud-native Customer Review Analytics Platform using Amazon Web Services (AWS). The system automatically processes raw customer reviews, performs sentiment analysis, extracts key phrases, generates text summaries, stores enriched insights, and visualizes results through interactive dashboards. The solution integrates multiple AWS services—S3, Lambda, Comprehend, SageMaker, DynamoDB, and QuickSight—to create a fully automated, scalable, and serverless analytics pipeline.

  The workflow begins when a CSV containing customer reviews is uploaded to Amazon S3. This event triggers an AWS Lambda function that reads each review, analyzes sentiment and key phrases using Amazon Comprehend, and enriches the data with additional metadata such as rating and product category. The enriched records are stored in DynamoDB for fast retrieval. A SageMaker JumpStart text-summarization endpoint generates an executive summary of overall customer feedback. Finally, Amazon QuickSight uses the processed dataset to build dashboards that display KPIs, sentiment trends, key phrase clouds, rating distributions, and detailed review tables.

  This architecture enables organizations to efficiently convert unstructured customer feedback into actionable insights, supporting product improvements, marketing decisions, and customer experience optimization.

- **Architecture Diagram :-**

Amazon S3 (input CSV)

↓

AWS Lambda (ETL + Comprehend)

↓

Amazon Comprehend (Sentiment + Key Phrases)

↓

Amazon DynamoDB (Processed data storage)

↓

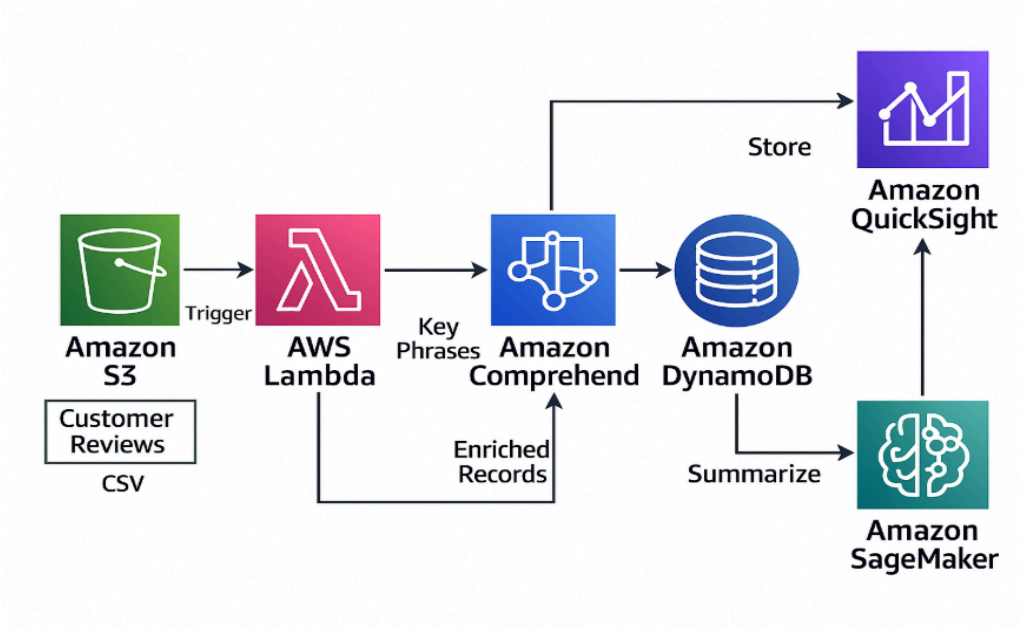
Amazon SageMaker (Summarization model)

↓

AWS Lambda (Summary generator)

↓

Amazon QuickSight (Dashboard)



- **Services Used :-**

1. Amazon S3
2. AWS Lambda (Serverless Compute)
3. Amazon Comprehend (Sentiment Analysis)
4. Amazon DynamoDB (NoSQL Database)
5. Amazon SageMaker (ML Inference)
6. Amazon QuickSight (Dashboard & BI)
7. IAM (Identity & Access Management)

- **Step-by-Step Deployment Guide :-**

1. **Create & Configure S3 Bucket :-**

1.1 Create S3 Bucket

1. Go to Amazon S3 → Create bucket
2. Bucket name: customer-review-storage-ak
3. Region: us-east-1
4. Keep default settings → Create Bucket

1.2 Upload Dataset

1. Open your S3 bucket
2. Click Upload
3. Upload your `.csv` dataset

2. **Create IAM Role :-**

1. FOR LAMBDA

- Open IAM → Roles → Create Role
- Choose Lambda
- Attach these AWS policies

Policy Name

AmazonS3FullAccess

AmazonDynamoDBFullAccess

AmazonComprehendFullAccess

AmazonSageMakerFullAccess

CloudWatchLogsFullAccess

- Name the role: lambda-review-role
- Click Create

2. FOR SAGEMAKER

- Open IAM → Roles → Create Role
- Choose SageMaker
- Attach these AWS policies

AmazonS3FullAccess

AmazonSageMakerFullAccess

CloudWatchLogsFullAccess

- Name it and Create the Role

- **STEP 3 — Create DynamoDB Table**
  - Open Amazon DynamoDB → Tables → Create Table
  - Configure:
    - Table name: ReviewAnalysis
    - Partition key: ReviewID (String)
  - Leave rest as default → Create Table

- **STEP 4 — Deploy Lambda (ETL + Comprehend Processor)**

4.1 Create Lambda

1. Go to AWS Lambda → Create Function
2. Function Name: reviewIngestProcessor
3. Runtime: Python 3.10
4. Execution Role → Choose lambda-review-role
5. Click on Create Function.
6. Add the Lambda.py Code in the creation function.
7. Change the JSon File in Test
8. Click on deploy
9. Click on Test.
10. We can check the logs in cloud Watch to confirm everything is working fine or not.
11. Go to Configuration → General
    - Timeout: 2 minutes
    - Memory: 1024 MB

- **STEP 5 — Add S3 Trigger**
    1. Go to Lambda → reviewIngestProcessor
    2. Click Add Trigger
    3. Select:
        - Source: S3
        - Bucket: your S3 bucket
        - Event: All object create events

- **STEP 6 — Deploy SageMaker Summarization Model**
    1. Open Amazon SageMaker → JumpStart
    2. Search: DistilBART CNN 6-6 Summarization
    3. Deploy the model as an endpoint
       Example endpoint name:
       jumpstart-dft-hf-summarization-dist-20251114-140145Wait
    4. until endpoint status = InService

- **STEP 7 — Deploy Lambda (Summarizer Generator)**
    1. Create Lambda Function

2. Name: reviewSummarizer
3. Runtime: Python 3.10
4. Role: lambda-review-role
5. Click on Create
6. Add the Lambda.py and change the JSon File.
7. Click on Test.

- **STEP 8 — Export DynamoDB to S3 for QuickSight**

1. Go to:
   DynamoDB → Tables → ReviewAnalysis → Export to S3
2. Choose your S3 bucket
   Output: `results/` folder in your bucket

- **STEP 9 — Connect QuickSight**
  1. Open Amazon QuickSight
  2. Go to Datasets → New Dataset
  3. Choose S3
  4. Upload Manifest File

- **STEP 10 — Build QuickSight Dashboard**
  1. Make the KPI Cards and Charts by using Keyphrases, Values.
  2. Save dashboard → Publish → Share.

- **LAMBDA CODE EXPLANATION**
  - For Ingest Summary

    - The Lambda function is triggered automatically whenever a new CSV file is uploaded to the S3 bucket.
    - It extracts the bucket name and file key from the S3 event notification.
    - The CSV file is read from S3 and processed using `csv.DictReader` to map each row to a dictionary.
    - It checks the `"Review Text"` column for each row and skips empty or invalid reviews.

    - Amazon Comprehend is used to run:
      - Sentiment analysis (`POSITIVE`, `NEGATIVE`, `NEUTRAL`, `MIXED`)

        - Key phrase extraction
    - Comprehend returns sentiment scores as floating values, which are converted to Decimal because DynamoDB does not allow floating-point numbers.
    - A unique ReviewID is generated by hashing the review text to avoid duplicates.
    - A structured item is created containing:
        - ReviewID
        - ReviewText
        - Rating
        - DetectedSentiment
        - SentimentScore (Decimal map)
        - KeyPhrases
        - ProductCategory
    - Each processed review is saved into the ReviewAnalysis DynamoDB table.
    - The function keeps a count of successful entries and logs detailed messages for processing status.
    - If any errors occur during processing, the function catches them and continues to the next row.

    - After processing all reviews, the function returns a summary with the total number of reviews saved.

- For Summary Generator

    - The Lambda function connects to DynamoDB and SageMaker Runtime using boto3.
    - It reads all review records from the DynamoDB table ReviewAnalysis using `table.scan()`.
    - It extracts the `ReviewText` field from every item and combines all review texts into a single long string.
    - To avoid exceeding the SageMaker model's input size limit, the combined text is trimmed to 6000 characters if it is too long.

    - The text is wrapped inside a JSON payload and sent to the SageMaker summarization endpoint using `invoke_endpoint`.
    - SageMaker returns a response containing `generated_text`, which is the summary of all reviews.

- The Lambda function extracts this summary from the response and prepares it for storage.
- It saves the summary back into DynamoDB under a special key:
    - `ReviewID = "SUMMARY#ALL"`
    - This makes it easy to retrieve later for QuickSight or other services.
- The function then returns a success message along with the generated summary.

- **Comprehend Sentiment Logic**

    - Comprehend analyzes the text to detect emotional tone.
    - It returns a sentiment label: Positive, Negative, Neutral, or Mixed.
    - It also provides confidence scores for all four sentiment categories.
    - The label is chosen based on the highest confidence score.
    - Scores are converted to Decimal before storing in DynamoDB.
    - Comprehend also extracts key phrases from the review text.
    - These outputs enable deeper insight into customer feedback.

- **SageMaker Summarization Workflow**

    - DynamoDB reviews are combined into one input text for summarization.
    - Lambda formats the text into the required JSON payload (`{"text_inputs": "..."}`).
    - The payload is sent to the SageMaker Runtime using `invoke_endpoint()`.
    - The deployed JumpStart model processes the text and generates a summary.
    - SageMaker returns the summary as `generated_text` in the response body.
    - Lambda extracts the summary from the JSON response.
    - The summary is saved back into DynamoDB under `ReviewID = "SUMMARY#ALL"`.
    - This summary is later used in QuickSight dashboards or reports.

- **DynamoDB Schema**

- Table Name: `ReviewAnalysis`
- Primary Key: `ReviewID` (String, unique identifier for each review)
- ReviewText: Stores the full customer review text.
- Rating: Stores the numerical rating extracted from the CSV.
- DetectedSentiment: Stores the overall sentiment label (Positive/Negative/Neutral/Mixed).
- SentimentScore: A map containing Decimal scores for Positive, Negative, Neutral, and Mixed.
- KeyPhrases: A list of important words/phrases detected by Comprehend.
- ProductCategory: Category or department of the product (e.g., Dresses, Tops).
- CreatedAt (Optional): Timestamp for when the record was inserted.
- Summary (Optional Item): Used only once for summary entry (`ReviewID = "SUMMARY#ALL"`).

- **Future Enhancement**

  - Implement personalized recommendation insights based on sentiment trends and customer behavior.
  - Replace CSV ingestion with API Gateway + Lambda so external applications can send live reviews.
  - Use Amazon OpenSearch to enable full-text search and advanced analytics on review data.
  - Integrate SNS or email alerts to notify teams about sudden spikes in negative sentiment.
  - Add auto-scaling SageMaker endpoints for heavy summarization workloads.
  - Build a mobile-friendly dashboard using QuickSight Embedded Analytics.
  - Store enriched data in Amazon Redshift for deeper BI and cross-department analytics.