Documentation: Solution for Processing Event Data for BobbleAI Keyboard

Processed Data Parquet file link:
https://test-buckets165.s3.ap-south-1.amazonaws.com/processed-data/part-00000-ab50c6a1-097c-493e-b246-e8257ef81db1-c000.snappy.parquet

Project Overview
Title: Efficient Data Processing ETL Pipeline for Event Records
Objective: To process raw product event data, filter relevant event records from the last five days before July 1, 2024, expand JSON columns, and store the final data in a structured Apache Parquet format S3.

Algorithm

1. **Data Filtering**
   Objective: Retain event records from June 26, 2024, to June 30, 2024, based on `event_date`.
   - We applied date filtering to the dataset to include only events within this date range.
   - Filter logic was implemented using PySpark's `.filter()` function.

2. Data Expansion
   Objective: Extract specific parameters from the nested JSON structure present in the `data` column.
   - Method: PySpark's `from_json()` was used to extract key-value pairs from the JSON column, excluding any nested structures such as lists or JSON objects that were not required to be expanded.

3. Data Storage
   Objective: Write the filtered and transformed data in Parquet format to S3.
   - We used the `.write.mode("overwrite").parquet()` function in PySpark to save the processed data.
   - The output was partitioned to ensure efficient storage and querying.

Methodologies

1. AWS Glue and PySpark for ETL
   AWS Glue was utilized to handle the Extract, Transform, and Load (ETL) operations. The raw data was loaded from S3, transformed using PySpark, and the output stored back in S3 in a partitioned Parquet format.

2. Date Filtering with PySpark

To retain only the events from the 5 days leading to July 1, 2024, we used PySpark's `.filter()` to apply conditions on the `event_date` column.

3. Data Expansion using PySpark

The `data` column, which was a nested JSON, was expanded into individual fields using PySpark's `from_json()` and `schema_of_json()` functions.

- Schema definition: The schema of the JSON was explicitly defined to parse and extract only necessary parameters.

4. Writing Data to S3 in Parquet Format

The final data was written in Parquet format, which is a highly efficient format for Big Data analytics.

- Parquet supports efficient storage and querying, and the schema was aligned to the provided specification.

Technology Stack

- **Cloud Services:** AWS (S3, Lambda, Glue)
- **Data Processing:** PySpark on AWS Glue
- **Storage:** S3 (Parquet format)
- **IAM & Security:** Managed using AWS IAM roles and policies for access control.

Assumptions

1. Event Date Format: The `event_date` column is assumed to be in a format directly comparable to date strings such as `'2024-06-26'`.
2. JSON Schema Consistency: The JSON structure within the `data` column is assumed to be consistent across all rows for ease of parsing.
3. Data Integrity: All data in the specified period is correctly recorded in the dataset without missing records.