E-Commerce Data Pipeline Documentation

Overview: This document outlines the structure and operation of a real-time data pipeline developed for an e-commerce analytics system. The pipeline streams order data using Kafka, ingests it into Snowflake, and enables analytical reporting through structured queries on dimension and fact tables.

Pipeline Components:

1. **Kafka Producer**
    - Simulates e-commerce order data generation.
    - Sends messages to a Kafka topic named orders.
    - Reference code: kafka_order_producer.py
2. **Kafka Consumer with Snowflake Integration**
    - Consumes messages from the Kafka orders topic.
    - Parses the order data and performs upserts into dimension tables and inserts into the fact table in Snowflake.
    - Reference code: kafka_to_snowflake_consumer.py
3. **Snowflake Schema and Table Setup**
    - The database and schema used is E_COMMERCE.
    - Includes three dimension tables: DIM_CUSTOMER, DIM_PRODUCT, DIM_DATE.
    - Includes one fact table: FACT_ORDERS with foreign key constraints referencing the dimension tables.
    - Reference code: E_COMMERCE_Database_Schema.sql

Analytical Use Cases:

1. **Monthly Sales Analysis**
    - Calculates total sales per product category per region on a monthly basis.
2. **Customer Retention Analysis**
    - Identifies the top 5 repeat customers over the past 30 days.
3. **Product Performance**
    - Aggregates total quantity sold and average price per product.
4. **Customer Behavior Across Regions**
    - Detects customers who have placed orders in more than one region.

For detailed implementation, refer to the respective code files listed above, including the Snowflake DDL in E_COMMERCE_Database_Schema.sql.

Bonus: ***Explain your reasoning for dimension/fact design:***
A star schema structure exists throughout the Snowflake E_COMMERCE database which enables efficient data querying and reporting functionalities. Dimension tables (DIM_CUSTOMER, DIM_PRODUCT, DIM_DATE) contain descriptive attributes that have low cardinality numbers which serve as analysis context including customer demographics and product categories and calendar hierarchies. The tables achieve denormalization to minimize query joins which results in improved performance.

Measurable order events stored in the FACT_ORDERS table use foreign keys to link back to dimension tables. The table includes numeric metrics possessing data points for quantities, sales totals and price reductions that enable aggregation across different data categories.

As a result of its hierarchical structure the system provides scalable characteristics with quick analytical query processing capabilities as well as rapid filtering functions and drill-down analytical features perfect for business intelligence applications in e-commerce applications.

Bonus: Explain how you would handle slowly changing dimensions in this model:

To handle Slowly Changing Dimensions (SCD) in this model, I would use the following strategies based on business requirements:

Type 1: For corrections where historical data does not need to be preserved (e.g., fixing a typo in a customer name), I would overwrite the existing record in the dimension table.

Type 2: For changes that require historical tracking (e.g., a customer changing their address), I would insert a new row in the dimension table with a new surrogate key. I'd include `effective_date`, `end_date`, and an `is_current` flag to track the record's validity period. The fact table would then reference the correct version based on the order date.

Type 3: If limited history is needed (e.g., tracking the current and previous product category), I would add separate columns like `current_category` and `previous_category`. However, this approach is rarely used due to its limited flexibility.

Snowflake's support for streams and tasks can automate SCD logic, ensuring data integrity and reducing maintenance overhead.