Story: Enhancing Order Processing with Product Integration

Background:

As part of our continuous effort to improve the efficiency of our order processing system, we've identified the need to better track products in our order data. Currently, our model doesn't have a dedicated **Product** table, and product details are loosely coupled in the **Order** table. This limits our ability to efficiently manage product-related data, track inventory, and generate insightful reports.

Objective:

We want to introduce a **Product** table into our database model and create relationships between orders and products. This will help us track products more effectively, ensure accurate inventory management, and provide better analytics on product sales. The changes will also facilitate linking order details directly to products, improving data consistency across the system.

Proposed Changes

1. Customer Table

- **Current Status**: No changes are required.
- **Reasoning**: The **Customer** table already contains the necessary attributes (`Customer_ID`, `Age`, `Country`, `LName`, `FName`) and is functioning as expected. We will maintain this table as is.

2. Order Table

- **Current Status**:
  - The existing model contains `Order_ID`, `Amount`, and `Item`.
  - The `Item` field holds product-related data in a free-text format.
- **Change**:
  - **Add**: `Product_ID` field to link the **Order** table to the new **Product** table.
  - **Remove**: The `Item` field will no longer be necessary since it will be replaced by the structured `Product_ID` field.
- **Reasoning**: By adding `Product_ID`, we create a relationship between orders and products, which improves product tracking, stock management, and the ability to produce detailed product-based reports. Removing `Item` eliminates redundancy and prevents inconsistencies.
- **Technical Specification**:
  - Field Name: `Product_ID`
  - Type: `INTEGER`
  - Foreign Key: `Product_ID` references the new **Product** table.

- Remove `Item` (free-text) from the schema.

---

3. Product Table

- **Current Status**: No existing **Product** table.
- **Change**:
  - **Create a new table** to store product-related information. This will include `Product_ID`, `Product_Name`, and `Quantity` to track each product and its available stock.
- **Reasoning**: The introduction of the **Product** table will centralize product information and allow us to manage stock efficiently. This change provides a structured way to track individual product details for each order, which is crucial for generating detailed analytics on product sales, stock levels, and product performance.
- **Technical Specification**:
  - Table Name: **Product**
  - Fields:
    - `Product_ID` (Primary Key, INTEGER)
    - `Product_Name` (VARCHAR, length 100)
    - `Quantity` (INTEGER)

---

4. Relationships

- **Current Status**:
  - The existing model links **Customer** to **Order** via `Customer_ID`. There is no direct link between **Order** and **Product**.
- **Change**:
  - Establish a new **Foreign Key** relationship between **Order** and **Product** using the `Product_ID` field. This ensures that each order can be associated with a specific product, allowing for efficient querying and data integrity.
- **Reasoning**: This new relationship will improve data integrity by linking each order to a specific product, preventing manual errors in product information entry. It will also make it easier to generate reports on product performance.
- **Technical Specification**:
  - Foreign Key: `Product_ID` in **Order** references `Product_ID` in **Product**.
  - Ensure that the foreign key constraint is enfor

Additional Technical Notes:

- **Data Migration**: Existing data in the **Order** table will need to be updated to reference `Product_ID` rather than relying on the free-text `Item` field. The Data Engineer should map existing `Item` data to `Product_ID` in the new **Product** table where applicable.
- **Database Indexing**: To improve query performance, create indexes on `Product_ID` in both the **Order** and **Product** tables. This will allow for faster lookups and joins when querying product-related order data.

- **Testing**:
    - After the schema changes are implemented, thorough testing will be required to ensure that existing orders are correctly linked to the new **Product** table.
    - Ensure that queries involving customer orders, products, and shipping continue to function without issues.
- **Performance Considerations**: Once the **Product** table is in place, it's essential to monitor database performance, especially as new orders are placed and more products are added. Creating the necessary indexes will help maintain efficiency.

Expected Benefits:

1. **Data Integrity**: Linking orders to products via `Product_ID` ensures more consistent and reliable data.
2. **Improved Analytics**: The ability to track product sales, stock levels, and product performance will be enhanced, allowing for more granular analysis and reporting.
3. **Streamlined Processes**: Removing the free-text `Item` field in favor of a structured relationship between **Order** and **Product** simplifies order management and reduces the likelihood of data entry errors.