Project Overview: Resolving Database Schema Inconsistencies

I encountered a foreign key constraint issue within my database in this project. Specifically, the `product` table had `product_id`s ranging only from 1 to 16, yet there were references to `product_id` values from 17 onwards in the `order_items`, `reviews`, and `cart_items` tables.

To resolve this, I did a simple fix by adding products with those missing product IDs. I adopted a basic naming strategy, assigning names such as "Product 17" for `product_id` 17, and followed a similar pattern for other absent product IDs. This approach was chosen for its simplicity and effectiveness, considering that the project emphasized constructing accurate SQL queries and ensuring database operational integrity rather than in-depth product analysis.

Key Considerations in My Approach:

- **Consistency:** While employing this straightforward naming convention, I maintained consistency in other attributes, such as pricing and category allocation, aligning them with the existing data values in other tables.

- **Documentation:** I've ensured to mark these product names as placeholders or simplified entries. This is crucial for the clarity of the project, especially given its nature as a technical exercise where the methodology needs to be transparent.

- **Data Utility**: I carefully considered the ultimate use of this data. While the placeholder data ensures operational continuity for database functions, it's acknowledged that for analyses where product specifics are critical, additional detailed product information would be necessary to enhance the dataset's richness and applicability.

Example Implementation:

We used straightforward placeholder names for product IDs like 17, 18, 19, etc. Considering the existing data pattern, prices and category IDs were determined based on what seemed fitting. For instance:

- 17, Product 17, Description for Product 17, [appropriate price], [appropriate category_id]
- 18, Product 18, Description for Product 18, [appropriate price], [appropriate category_id]
- 19, Product 19, Description for Product 19, [appropriate price], [appropriate category_id]
- ...

This method uses entries like "Product 17", "Product 18", and so forth as basic placeholders. The prices and category fields are filled to align with the existing values in the other dataset's tables.

Furthermore, for the `product` table, we used the `unit_price` from the `order_items` table as a replacement for the `price` for products 17 to 30, as `order_items` includes details up to product 30. For products 31 to 36, I opted for a placeholder value of 1.00 for the price.

I adhered to a presumed rule for the `category_id` in the `product` table. I followed the existing pattern, sequentially assigning category IDs, such as 9 for products 17 and 18, 10 for products 19 and 20, and so on, to maintain a logical pattern and consistency within the dataset.

Impact of Placeholder Data on SQL Queries

Here is how this approach may affect the outcomes of each problem:

Problem 1: While we have not explicitly assigned a 'Sports' category to products 17 through 36 in the placeholder data, some of these products might fall under the 'Sports' category in a real-world scenario. If that were the case, the query results would be different.

Problem 2: The query calculates the total number of orders per user. Placeholder products, if ordered, will be included in the order counts, accurately reflecting user activity.

Problem 3: The query determines the average ratings for each product.
In our dataset, products 1 to 30 have been assigned reviews, which means they will be included in calculating average ratings. For products 31 to 36, no reviews have been recorded. Hence, the average rating for these products will be represented as NULL.

Problem 4: The query identifies the top 5 users based on their spending on orders. It's important to note that prices for products 17 to 30 have been approximated based on data from the order_items table. While these are not the actual prices, they serve as a stand-in to ensure our analysis can proceed. This estimation means that these placeholder values could influence the identified top spenders.

Problem 5: The query factors in the average product ratings, and the results include the product_name. It's important to note that while we have ratings for products 17 to 30, we do not have their actual names; they are listed with placeholder names in our dataset. As such, while these products will contribute to the analysis of the highest-rated items, the product_name field does not reflect real product names. Products 31 to 36, which lack ratings, will not appear in this query's results.

Problem 6: This query aims to identify users who have placed orders for at least one product from every category. It's important to note that for products 17 to 36, we do not have information about their real categories; they are listed with placeholder data in our dataset. If these placeholder products represent unique categories not covered by existing products, they will contribute to satisfying this condition. However, it's crucial to acknowledge that our dataset includes 50 categories, while we only have 36 products (including placeholders) after adding them. Given this

mismatch, it may not be possible for users to have placed orders for at least one product from each category due to the disparity in the number of categories and products.

Problem 7: The query seeks products without reviews. It's important to note that for products 31 to 36, we do not have review data available. Therefore, these placeholder products will be listed here if no reviews exist for them, accurately reflecting the current state of product feedback for the products we have data for.

Problem 8: Users who order on consecutive days will be identified by this query. Placeholder products are considered like any other product and will validate consecutive ordering behaviour if involved in such orders.

Problems 9, 10, 11, and 12: All these problems involve aggregation and pattern recognition within user behaviour and product characteristics. Placeholder products will be part of these aggregations and patterns, affecting the results as actual products would. Their influence will be subject to the same calculations and conditions applied to the rest of the products.

In generating this placeholder data, I have assigned prices and categories that align with existing products to ensure that analytical outcomes remain consistent and meaningful. This approach underscores the importance of a cohesive dataset while preserving the ability to perform nuanced SQL queries for insightful data analysis.