Hi,

I wanted to summarize the results and recommendations of my project analyzing the receipt and rewards data.

1. There are some inconsistencies in the data that I analyzed; resolving these inconsistencies will help us improve the rewards program.
2. Based on the subset of database queries that I investigated, I have some recommendation to improve the scalability of the existing solution, which will result in more efficient and timely reports. I have provided more specific suggestions below.

Regarding the inconsistencies in the data:
1. There are many receipts that don't have any items purchased, or all items are listed as “ITEM NOT FOUND”.
2. There are many “Test brands” in the Brands table?

If we can get more accurate data, that would improve the quality of our results.
3. There are several branded items on individual receipts that do not have a corresponding brand from the brands table. I discovered this issue while comparing the brand IDs of the individual items purchased with the brand IDs from the brands table. As an example, in the case of “Heinz”, the record exists in the brands table, but the ID for "Heinz" is different in the brands table and the individual items on the receipt. This discrepancy means that some customers may not get the rewards they expect.
Here are some suggestions on how we could fix these discrepancies. For the missing brand we would need to know the correct brand information. It would also be useful to understand why they were excluded from the table originally. Also, in the case of "Heinz", it would be good to understand whether the values are supposed to be different between the receipt IDs and the brands table, and also which brand ID (the one in the receipts table or the one in the brands table) is the correct ID to use or whether the brand IDs in both tables are supposed to be the same.

Regarding scalability and performance issues:
We need to ensure that query and update performance doesn't degrade linearly as our database size grows.

We need to have a good understanding of which operations are worth optimizing. In the case of queries, we want to optimize for response time (how quickly can we answer the query), for update operations, we want to optimize for throughput. I would like to get a prioritized list of performance sensitive operations.

Assuming that we have this prioritized list, here are some general guidelines for optimizing database access.
1. Can we partition the data to take advantage of query parallelism?
2. Can we use materialized views to improve query performance?
3. Can we selectively denormalize some tables in the schema to reduce joins?
4. Can we selective cache small tables to avoid disk access?
5. Can we examine the execution plan that the optimizer chooses for sample data sets and provide

optimizer hints if appropriate?

These are just general guidelines, and we can apply them as appropriate once we have a better understanding of the performance-sensitive operations.

Please let me know your thoughts and any questions you may have. Looking forward to hearing from you at your earliest convenience.

Best,
Nishant