Optimization Ideas for Queries

1. We will try to create a “interaction_datekey” column rather than interaction_date column in the fact_customer_interactions table.
    a. This will be an INT column which will be a compressed data type, will save space in storage making table smaller and run faster, since we will have group by clauses on this.
    b. In future for cloud data warehousing this can also be used as a partition key for performance and storage purposes.
2. We will create covering indexes
    a. user_id and interaction_count
    b. product_id and interaction_count.

    This will make data retrieval very fast as user_id will be part of non-clustered index and interaction_count will be a covering column in the include clause making the fetching faster since that is used in summing operation.
3. It is important to create a clustered index on the “fact_customer_interactions_id” for ordered storage. We can make the ordering of clustered index descending so that the latest data can be accessed first.
4. When this is on relational database and the data grows extremely high then there can be partitions created on interaction_date (interaction_datekey) so as to have better storage and performance. This will make queries filter data faster, make more optimized plans and fetch data faster.