1. Dim\_Channel – Provides the channel through which the product is sold. The grain here is each individual channel identifier. You might notice that the channel category is in this table as well; I had flattened this hierarchy since it was a simple rather than a ragged hierarchical structure.
2. Dim\_Product – Another scenario where I flattened a simple hierarchical structure. There were a couple of other options for dealing with this, but the assignment said to stay away from snowflaking. The grain for this table was the individual product identifier, meaning each individual unit. This was where I had the most difficulty since this essentially determined the grain for my entire ERD; I could have run with product type instead, aggregating individual products into line items that consisted of multiple units of a product type being purchased at once, but I chose to provide the most granular data available since I did not see any business requirements to base my decision off of. My choice here would harm performance but allow analysts and end-users to dig deeper into the data than if I had gone with the alternative.
3. Dim\_Date – I chose to create a Date Dimension and am thankful we don’t need to snowflake this thing. I provided that same type of flattened structure from the prior two tables. If it is not clear, the gain for this dimension is the date. One thing of note here: I chose to have every table reference the Date Dimension because they had their ‘created\_date’ and ‘modified\_date’ attributes. I don’t know if this was correct. I’d certainly believe an argument that claimed these deserved their own dimensions.
4. Dim\_Reseller/Dim\_Customer/Dim\_Store – The grain for these dimensions focused on individual units (one store, one customer, etc.), and all that was required here was to shift the information over to this new format. I decided to keep the address/geolocational information separate, but I could see why we would want to give them their own dimension.
5. Dim\_SalesHeader – I think of the sales header as the shopping cart with many items inside of it. This is the grain: one shopping cart/purchase that can include a diverse number of goods within it.
6. Dim\_LineItem – I used this name because it felt more fitting than sale\_details from the original DB. This represents the selection of products within a product type within a sales header. If I purchase three phones and a tv, each phone is a product under one product type and orderline; the tv is treated the same way. We’ll have two order lines under one sales header. Writing it out like this, the two (orderline and sales header) have a hierarchical relationship, but they feel distinct enough to warrant separate tables.
7. Fact\_Sale – The fact sale table provides information about each individual item/product that is sold (each time something is scanned at checkout for example). The facts represented here are the price, quantity, cogs, wholesale price, and weight. I was hesitant to include quantity because you could count by orderline\_id for that information, but it felt like a nice timesave. (It may have also been an incorrect decision.) Of course, most of the dimensions are linked to the table via their keys.

**Addendum:**

1. Fact\_Targets – The fast targets table provides a view on what the goal is for the business unit. It is provided on a yearly basis, but I wanted to align it, perhaps wrongly, with the fast sale grain. As a result, I replaced the year with the date\_id from the date table. This would be a relatively easy transformation (divide the target by 365), and it would allow for a comparison on a day-by-day basis for both the sales and quantity metrics.
2. Dim\_Address -- I took on the optional task by adding a dimension representing addresses. I obviously hadn’t fully fleshed out this dimension due to the intensive nature of doing so for both this and the date dimension, but I believe this satisfies the requirements of the assignment.