Module 12 - Discussion Prompt Questions
Jot down short answers for each prompt question as you go along. Review and update your answers after finishing and submit before Tuesday midnight. Click on "Module 12 - Discussion Prompt Questions" above to submit.
The prompt questions for this module are:

1. What are some of the largest employers with which you are familiar? What do you imagine are some of the greatest challenges facing them in terms of information management?

Places like GE, Apple, Amazon, etc. Some challenges include storing information securely, but simply storing the information so that it can be accessed quickly is a problem. Common information includes customer information like email, address, and contact info. That's all probably done with a proven database software like postgresql but underneath that software, data structures like b-trees and hash tables are probably used. I imagine attributes like email which are often searched for are probably indexed by either b-tree or hash table. The B-tree might ultimately be quicker than the hash table because the hashing itself might take too much time even though its O(1).

It might look up customer records in a database when a customer accesses their account page. For ex: on Amazon, when I go to my account page, I want to see my orders, contact info, address, credit card, etc.

Also, when I place an order, that order needs to be stored somewhere so that it could be processed at a later date. That order needs to be stored in a database somewhere so that the warehouse management system can process that order maybe one day later and send out the order to various fulfillment centers.

3. What are the pros and cons of using a social security number as a primary key? What are some good alternatives?

Social security numbers are unique. That's one advantage. If you look up by social security number, you'll get a unique person. However, it's sensitive info so you'll need to hash that information which takes time. That might be too long of a process when constantly reading and inserting by that information. Hashing takes time.

instead, you could use less sensitive information that still has a high likelihood to be unique. You could combine first and last name, email, and address and that string will be fairly unique.

To grader: care to elaborate if this is wrong? I was wondering, is there such a thing as too many buckets? Can you create too many buckets for a hash table and is this a problem?

4. How big an impact, on your design considerations in general, is it to need a sorted file only periodically (for a directory, for example)?

If it's only on the fly, I think the data could be calculated on the fly as well and so it doesn't need to be stored. You could just take all of the data and put it in a tree and then get the in order traversal or put all records in a list and sort it on the fly. The sorted data doesn't need to be stored, it could be done lazily or when needed. So it doesn't seem like too big of a design problem?

To grader: care to elaborate if this is wrong?

5. How big an impact, on your design considerations in general, is it to sort by different fields of the data periodically (a directory is likely ordered by last name, first name, for example)?

It seems like if different fields needs to be sorted, each field would need a different tree or sorted storage structure. That structure needs to be stored, maintained, or calculated on the fly. That seems actually like a pretty big storage problem.