Section 1:

You can start this section following *session 1*, but you may want to wait until you feel comfortable wtih basic SQL query writing.

Steps to complete this part of the assignment:

- Design a logical data model
- Duplicate the logical data model and add another table to it following the instructions
- Write, within this markdown file, an answer to Prompt 3

Design a Logical Model

Prompt 1

Design a logical model for a small bookstore. 🗄

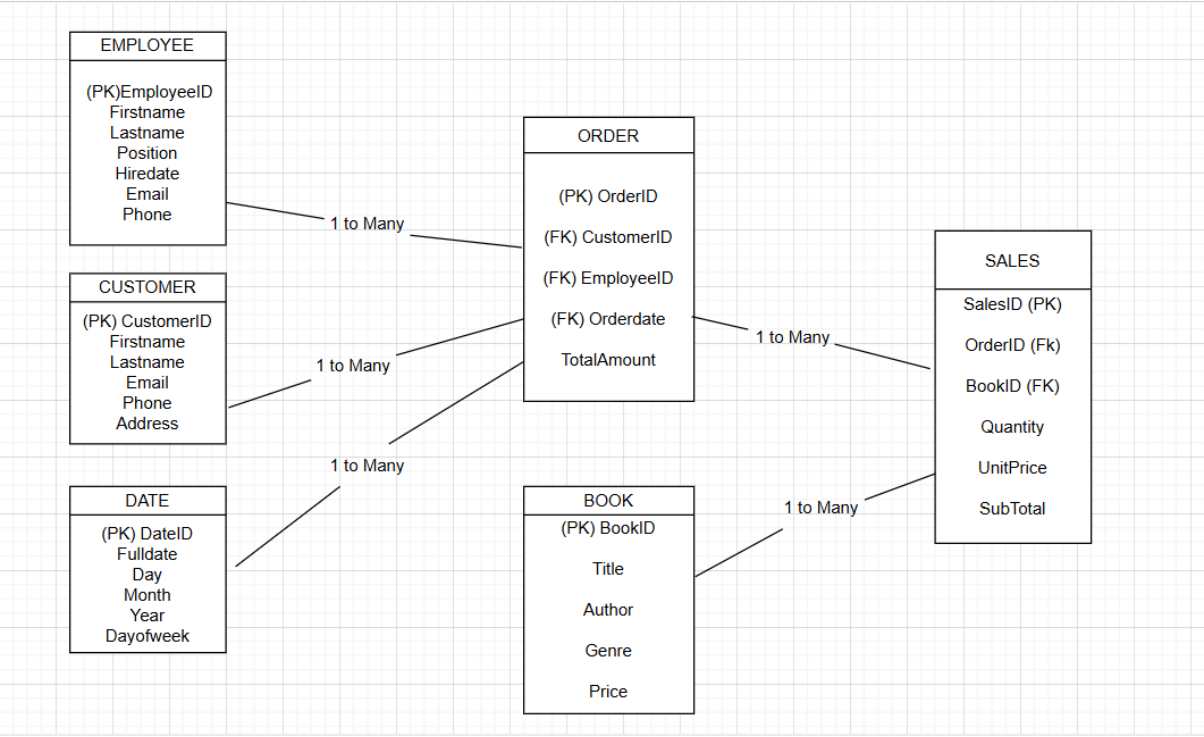
At the minimum it should have employee, order, sales, customer, and book entities (tables). Determine sensible column and table design based on what you know about these concepts. Keep it simple, but work out sensible relationships to keep tables reasonably sized.

Additionally, include a date table.

There are several tools online you can use, I'd recommend Draw.io or LucidChart.

HINT: You do not need to create any data for this prompt. This is a logical model (ERD) only.

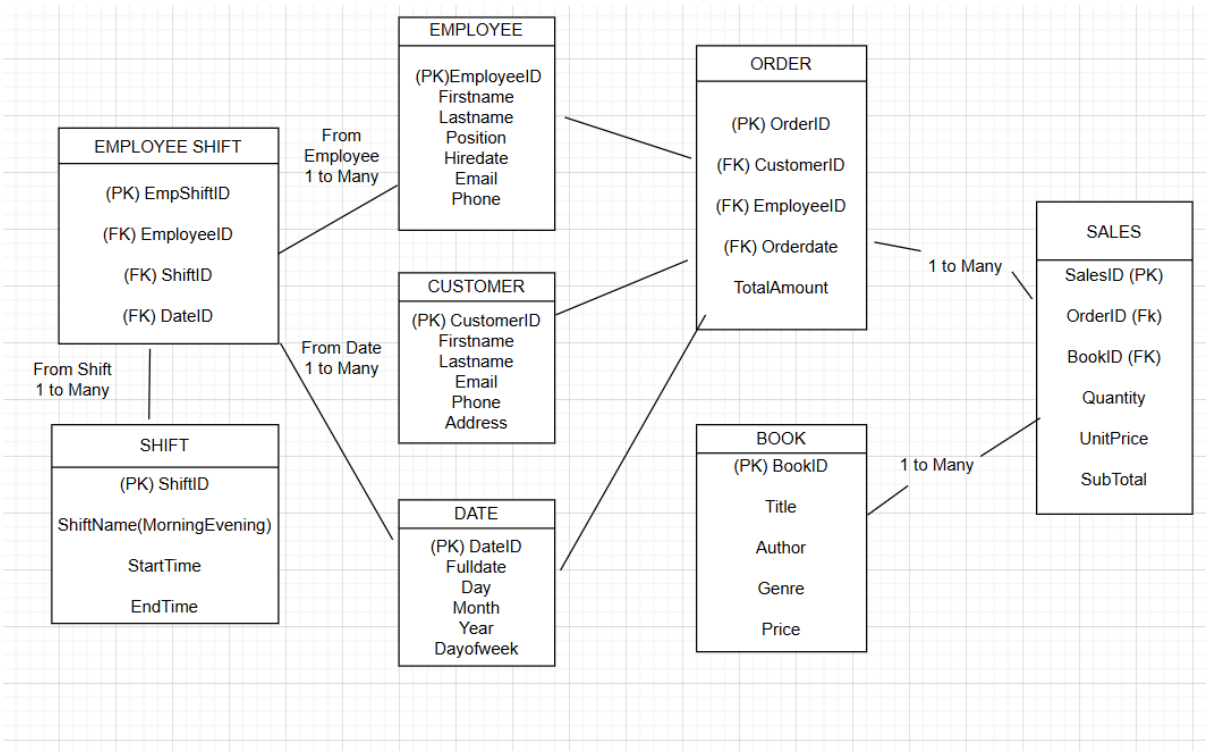
ANSWER 1



Prompt 2

We want to create employee shifts, splitting up the day into morning and evening. Add this to the ERD.

ANSWER 2



Prompt 3

The store wants to keep customer addresses. Propose two architectures for the CUSTOMER_ADDRESS table, one that will retain changes, and another that will overwrite. Which is type 1, which is type 2?

HINT: search type 1 vs type 2 slowly changing dimensions.

Your answer...

ANSWER 3

Type 1 and Type 2 are two different ways of handling data changes, known as Slowly Changing Dimensions (SCDs). In Type 1, changes overwrite the old data, so only the current information is stored and no history is maintained. In Type 2, changes are recorded as new entries, which allows the database to preserve the history of past values along with the current one.

Type 1 (Overwrite): If we only need to store a customer's current address, we can simply add address fields (e.g., AddressLine1, City, PostalCode) directly to the CUSTOMER table. Each time an address is updated, the old value is replaced.

Type 2 (Retain History): If we need to keep the full history of customer addresses, we should create a separate table, such as CUSTOMER_ADDRESS_HISTORY. This table would contain the address fields along with StartDate and EndDate (or a flag for current). When a customer moves, a new record is inserted with the new address, and the old record is marked as expired, allowing us to track all historical addresses.