**Data Modeling**

In this exercise, you’ll be doing some data modeling and schema design! For each of these exercises, diagram what the tables and relationships should look like as well as some other things you might want to store/potential challenges with the model and/or information you are storing.

Once you’re satisfied with your design, write out the DDL operations you’d need in order to create the tables and columns that you’ve identified. Then try to insert some data into your database and ensure that it works the way you’d expect.

[Download exercise](https://curric.springboard.com/software-engineering-career-track/default/exercises/sql-ddl-design.zip)

**Part 1: Schema Design**

**After completing these schemas, be sure to chat with your mentor and ask for a code review.**

**Part One: Medical Center**

Design the schema for a medical center.

A medical center employs several doctors

A doctors can see many patients

A patient can be seen by many doctors

During a visit, a patient may be diagnosed to have one or more diseases.

* **Based on the part 1 requirements, we can identify the following entities (tables)**
  + Doctor
  + Patient
  + Disease
  + Visit
* **We can also identify the relationships** 
  + A medical center employs several doctors (One-to-Many relationship between MedicalCenter and Doctor)
  + A doctor can see many patients (One-to-Many relationship between Doctor and Patient)
  + A patient can be seen by many doctors (One-to-Many relationship between Patient and Doctor)
  + During a visit, a patient may be diagnosed with one or more diseases (One-to-Many relationship between Visit and Disease)
* Per the instructions, we are going to use a tool to design the schema, aka, show the relationships of the table with an interface
  + In this case, we are going to use the Quick DBD tool
    - (we actually need to pay for it so were just going to do it here in the word document)
  + Here is what the schema should look like with the tables and their primary keys
    - MedicalCenter
      * MedicalCenterID (PK)
      * Name
    - Doctor
      * DoctorID (PK)
      * MedicalCenterID (FK)
      * Name
      * Specialty
    - Patient
      * PatientID (PK)
      * Name
      * DateOfBirth
      * Gender
      * Address
    - Visit
      * VisitID (PK)
      * DoctorID (FK)
      * PatientID (FK)
      * Date
      * Diagnosis
    - Disease
      * DiseaseID (PK)
      * VisitID (FK)
      * Name
      * Description

**Part Two: Craigslist**

Design a schema for Craigslist! Your schema should keep track of the following

The region of the craigslist post (San Francisco, Atlanta, Seattle, etc)

Users and preferred region

Posts: contains title, text, the user who has posted, the location of the posting, the region of the posting

Categories that each post belongs to

Based on these requirements, here are the table that should be comprised of

* Region
* User
* Post
* Category

And here are the attributes and relationships for each entity

* Region
  + RegionID (PK)
  + Name
* User
  + UserID(PK)
  + Username
  + PreferredRegionID(FK referencing RegionID)
* Post
  + PostID (PK)
  + Title
  + Text
  + UserID (FK referencing UserID)
  + Location
  + RegionID (FK referencing RegionID)
* Category
  + CategoryID (PK)
  + Name
* The relationships are as follows
  + Each user has a preferred region
  + Each post is associated with a User, a region, and possibly one or more categories
  + Each post can belong to multiple categories, and each category can have multiple posts
* We also need to add this relationship table/schema
  + PostCategiry (Association Table)
    - PostID (FK)
    - CategoryID (FK)
  + This is used to repreent the many-to-many relationship between Post and Category
  + It also helps with data integrity and normalization, one of the three levels of data normalization
  + It also offers flexibility and Scalability

**Part Three: Soccer League**

Design a schema for a simple sports league. Your schema should keep track of

All of the teams in the league

All of the goals scored by every player for each game

All of the players in the league and their corresponding teams

All of the referees who have been part of each game

All of the matches played between teams

All of the start and end dates for season that a league has

The standings/rankings of each team in the league (This doesn’t have to be its own table if the data can be captured somehow).

Based on the requirements, here are the tables/entities we should have in our schema

* Team
* Player
* Game
* Goal
* Referee
* Match
* Season

Here are the attributes and relationships for each entity:

* Team:
  + TeamID (Primary Key)
  + Name
  + Country (optional)
* Player:
  + PlayerID (Primary Key)
  + Name
  + TeamID (Foreign Key referencing TeamID)
* Game:
  + GameID (Primary Key)
  + MatchID (Foreign Key referencing MatchID)
  + SeasonID (Foreign Key referencing SeasonID)
  + Date
* Goal:
  + GoalID (Primary Key)
  + PlayerID (Foreign Key referencing PlayerID)
  + GameID (Foreign Key referencing GameID)
  + MinuteScored
* Referee:
  + RefereeID (Primary Key)
  + Name
* Match:
  + MatchID (Primary Key)
  + HomeTeamID (Foreign Key referencing TeamID)
  + AwayTeamID (Foreign Key referencing TeamID)
  + RefereeID (Foreign Key referencing RefereeID)
  + Date
* Season:
  + SeasonID (Primary Key)
  + StartDate
  + EndDate

**Part 2: Schema Critique**

We’ve provided you with a handful of SQL files that will create some databases and populate them with some data. Run each of the seed files and take a look at the data that’s generated. Next, think about how you could improve the schema. Finally, modify the original seed files based on your updated schema!

**Schema One: Outer Space**

To get the data:

Shell

Copy

$psql < outer\_space.sql

Suggestions for this schema:

* use more specific data types for columns instead of just FLOAT
* the orbits\_around column could be a foreign key since its used in other areas
* the moons columns stores an arrya of moons, hinting that moons could be its own table
* add comments to explain each table and what it is used for

​

**Schema Two: Air Traffic**

To get the data:

Shell

Copy

$psql < air\_traffic.sql

* Instead of storying arrays in columns, we can make separate tables for artsits and producers
* We can also use more datatypes specifically
* Also add indexes and notes for what the tables are and what they are used for

​

**Schema Three: Music**

To get the data:

Shell

Copy

$psql < music.sql

* We can use a timestamp data type instead of a text type
* Do not sire city and countries in the tickets table, they should have ttheir own table
* Add maybe some constraints to the columns to prevent unwanted datatypes