# Short Answer:

Answer the following questions with complete sentences in **your own words**. You are encouraged to conduct your own research online or through other methods before answering the questions. If you research online, please consult multiple sources before you write down your answers.

1. What is a relational database?
2. What is data modeling? Can you give an example and explain data modeling in terms of entities and their relationships?
3. What is a primary key and a foreign key?
4. Explain normalization and why it is useful.
5. What is a transaction? Assuming part of it succeeded and another failed, what happens?
6. Explain ACID.
7. What is a non-relational database?
8. Explain CAP.
9. What does it mean to have eventual consistency?
10. What is the diﬀerence between vertical and horizontal scaling?
11. What are the diﬀerences between database scalability techniques like replication, partitioning, and sharding?
12. When would you choose a RDBMS vs NoSQL database?
13. What is mongoose? Why would we use it instead of interacting with the native MongoDB shell?
14. Explain embedded vs reference relationships.

# Coding Questions:

Use HTML/CSS/JS to solve the following problems. You are highly encouraged to present more than one way to answer the questions. Please follow best practices when you write the code so that it would be easily readable, maintainable, and eﬃcient. Clearly state your assumptions if you have any. You may discuss with others on the questions, but please write your own code.

## To-Do App (Express.js, EJS, SCSS, MongoDB)

Modify the backend for yesterday’s to-do application so that it interacts with a remote MongoDB collection for data storage instead of using json files.

* + There shouldn’t be any file-system related code.
  + All CRUD operations should involve the database.

## Spotify Lite (Express.js, MongoDB)

You will create a mini-Spotify application with the following features:

* + Users can browse the most popular songs based on their language and genre.
  + Users can like a song.
  + Users can search for songs based on the artist name or song title.
  + Users can follow an artist.
  + Users can view & edit their own profile (username, email, password).

## Step 1: Data Modeling

Decide on the diﬀerent entities and schemas that you will use. Include them here (you can take screenshots of the schema code).

## Step 2: Implement the API

Since we haven’t yet discussed login/registration/user auth, you can hardcode some users

in the database. Then you can manually add the userID in the requests to these routes.

GET /user/songs : display all the songs that a user liked

GET /songs?language=“” : display matching songs based on their language GET /songs/:category : display matching songs based on their category PUT /songs/:song\_id : user likes a song

PUT /artists/:artist\_id : user follows an artist

PUT /user/info : user updates their username, email, or password.

## Step 3: Test the API with Postman

Include screenshots of the server responses when using Postman.

## OPTIONAL Frontend (this will be required in the next assignment)

Use EJS & SCSS to create a frontend for this mini-Spotify application.

* + Focus on the basic functionality (the features listed above).
  + For styling & layout, try recreating the Spotify website.