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. What is a non-relational database?
7. What does it mean to have eventual consistency?
8. What is the difference between vertical and horizontal scaling?
9. When would you choose a RDBMS vs NoSQL database?
10. What is mongoose? Why would we use it instead of interacting with the native MongoDB shell?
11. 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.

1. **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. **Follow the MVC file structure.**
   - There shouldn't be any file-system related code.
   - Do not include the **node_modules** folder in the submission.
   - All CRUD operations should involve the database.

2. **Spotify Lite (Express.js, MongoDB)**
   This application should have the following features:
   - Users can view all the songs that they've liked.
   - Users can search for songs based on the artist name or song title.
   - Users can filter song search results based on their language and genre.
   - Users can like a song.
   - 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?search=“”&language=“”&genre=“”` : display a list of songs that match the search input based on artist name or song title, and filtering based on optional inputs such as language and genre.
`PUT /songs` : user likes a song
`PUT /artists` : 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.

Step 4: Frontend
Use EJS or plain HTML to create a frontend.
- Ignore styling for now. Focus on creating the basic frontend features needed to interact with the backend. See if you can also make it functional (clicking buttons would send fetch requests to the backend). For example:
  - Input fields and buttons for login/signup/logout/editing profile.
  - A search bar (input field and button) to find songs.
  - A list of songs to display
  - Buttons to like songs and follow artists, etc.

If you have time, recreate the Spotify website based on the theme, layout, … SCSS is recommended but you can use any styling u want.
Do not include the node_modules folder in the submission.