# 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?**

Relational database is a type of database that stores and provides access to data points that are related to one another.

2. **What is data modeling? Can you give an example and explain data modeling in terms of entities and their relationships?**

Data Modeling is the organization between different data and how they relate to each other.
-One to One: when one row in a table may be linked with only one row in another table and vice versa. Ex. One User and One Shopping Cart
-One to Many: when one row in table A may be linked with many rows in table B, but one row in table B is linked to only one row in table A. Ex. One User owning different music
-Many to Many: when multiple records in a table are associated with multiple records in another table. ex. customers purchase various products, and products is purchased by many customers

3. **What is a primary key and a foreign key?**

Primary key ensures that data in the specific column is unique. Foreign Key is an id that provides a link between data in two tables.

4. **Explain normalization and why it is useful.**

Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.

5. **What is a transaction? Assuming part of it succeeded and another failed, what happens?**

Transaction is a sequence of multiple operations performed on a database, and all served as a single logical unit of work.
-if it is successful, all changes are put to the table –If it fails, changes are deleted or rolled back.

6. **Explain ACID.**

ACID is a set of 4 key properties that define a transaction:
**Atomicity:** either all of the transaction succeeds or none of it does
**Consistency:** all data will be consistent and valid according to all defined rules, including any constraints, cascades, and triggers.
**Isolation:** all transactions will occur in isolation. No transaction will be affected by any other transaction
**Durability:** once a transaction is committed, it will remain and be permanent in the system

7. **What is a non-relational database?**

Non-relational database is a database that does not use the tabular schema of rows and columns found in most traditional database systems

8. **Explain CAP.**

CAP is a belief from about data that claims it is possible to provide either consistency or availability—but not both.

- **Consistency.** All reads receive the most recent write or an error.
- **Availability.** All reads contain data, but it might not be the most recent.
- **Partition tolerance.** The system continues to operate despite network failures

9. **What does it mean to have eventual consistency?**

Eventual consistency means that updates made to databases will eventually be reflected across all nodes.

10. **What is the difference between vertical and horizontal scaling?**

Horizontal scaling involves adding additional servers and partitioning the system dataset and load over those servers. Vertical scaling involves expanding the resources used by a single server

11. **What are the differences between database scalability techniques like replication, partitioning, and sharding?**

**Replication** (Copying data)— Keeping a copy of same data on multiple servers that are connected.
**Partitioning** — Splitting up a large monolithic database into multiple smaller

databases.
**Sharding** (Horizontal Partitioning)— A type of horizontal partitioning that splits large databases into smaller components, which are faster and easier to manage.

12. **When would you choose a RDBMS vs NoSQL database?**
13. RDBMS databases are used for normalized structured (tabular) data strictly adhering to a relational schema. NoSQL datastores are used for non-relational data,
14. **What is mongoose? Why would we use it instead of interacting with the native MongoDB shell?**

Mongoose is a MongoDB ODM library that manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB. Mongoose is use to work with asynchronous environment with Promises and callbacks, buffering queries, and Data validation

15. **Explain embedded vs reference relationships.**

Embedded documents are stored as children inside a parent document.
Referenced documents are stored in a separate collection to their parent document

# 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.
   - There shouldn't be any file-system related code.
   - All CRUD operations should involve the database.

2. **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.

3. **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.