Introduction to Databases

Get familiar with what a database is, the types of information a database can store, and two common classifications of databases.

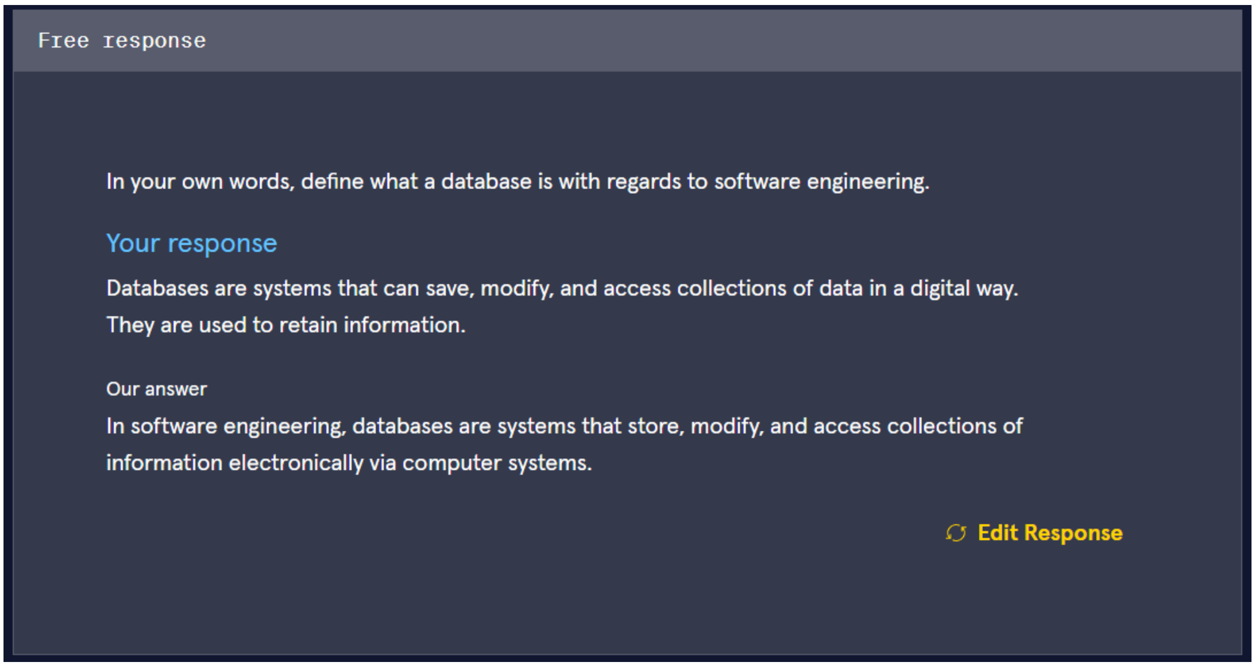
What is a Database?

Each time you log into Codecademy.com, you're met with an abundance of content organized into courses, articles, projects, and more. You can also see a dashboard, customized for you, with shortcuts to your courses and projects. How can the Codecademy site, or any site, retain and retrieve such large amounts of information in an organized and consistent manner? Enter databases!

In software engineering, databases are systems that store, modify, and access collections of information electronically. To better understand the different database classifications and how they can be useful to us as developers, in this article, we will:

- Discuss why we use databases in software development and their benefits for our end users.
- Examine the various types of data a database can store.
- Consider the features, advantages, and disadvantages of the two most common classes of databases: relational and non-relational.

Let's get started!



In your own words, define what a database is with regards to software engineering.

Your response

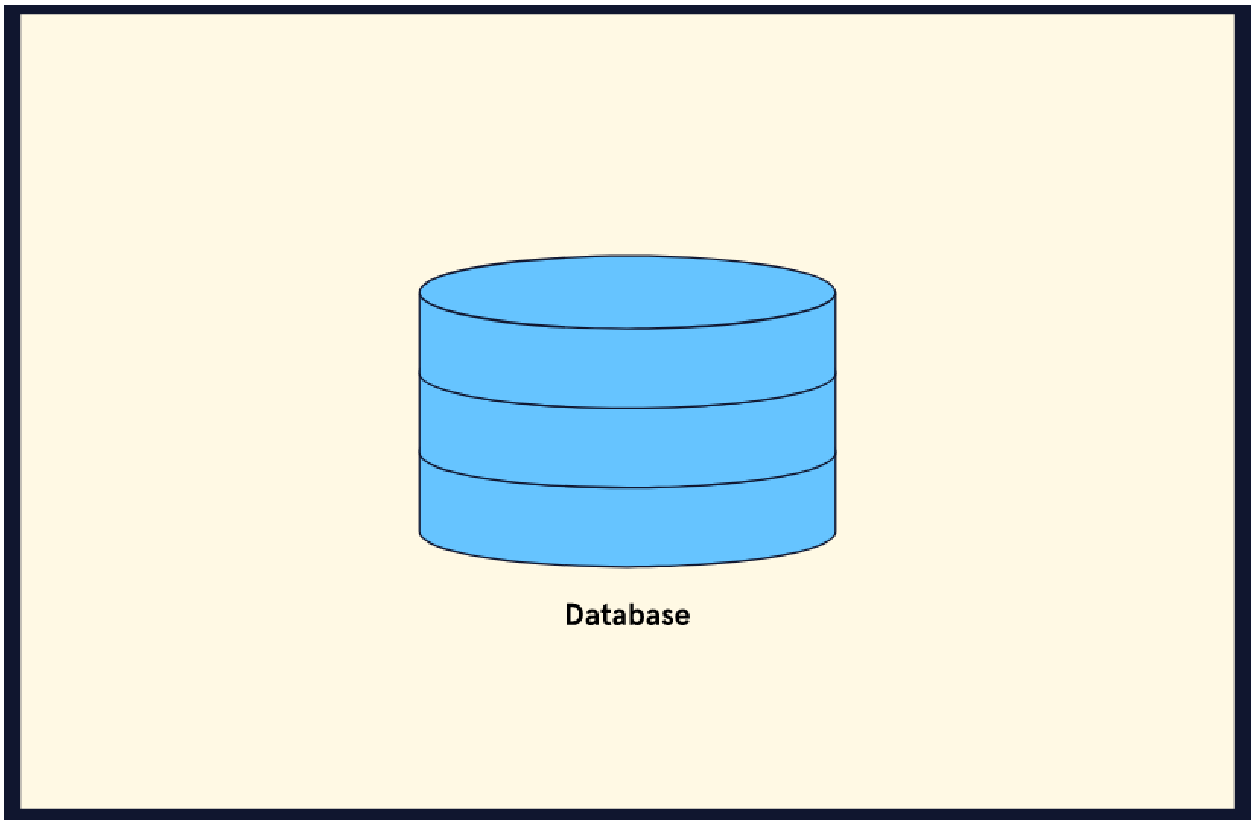
Databases are systems that can save, modify, and access collections of data in a digital way. They are used to retain information.

Our answer

In software engineering, databases are systems that store, modify, and access collections of information electronically via computer systems.

Why Databases?

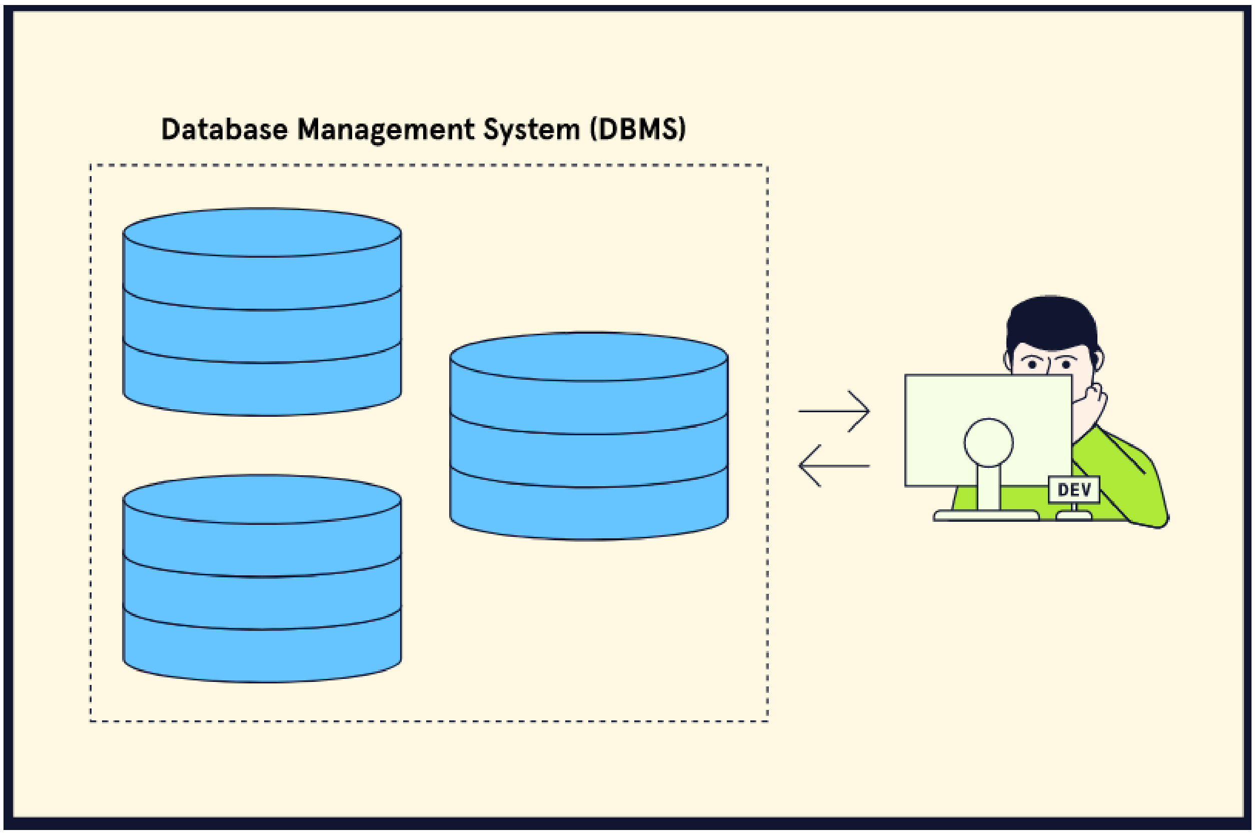
Almost any type of software application needs a way of storing data. Whether it is user account information for social media websites like Twitter or entire user-generated videos on websites like YouTube, databases are a crucial component of any application we use today. Without them, developers would have no way to meaningfully persist information that is important for the application’s functionality. To paint a picture, we can think of a database as a bucket that can store any type of information we want. Here is how most resources on the internet depict a database:



But, how do the applications we use (or build) interact with a database? When we are looking to interact with a database, we are actually looking to interact with a **database management system**.

Database Management Systems (DBMS)

Suppose a database is a bucket that stores our data. In that case, a DBMS is the software that encapsulates said bucket (our database(s)) and lets us work with the database using a programming language or easy-to-use graphical interface (GUI). Here is the same image from before, but now illustrating how a DBMS fits into the picture:

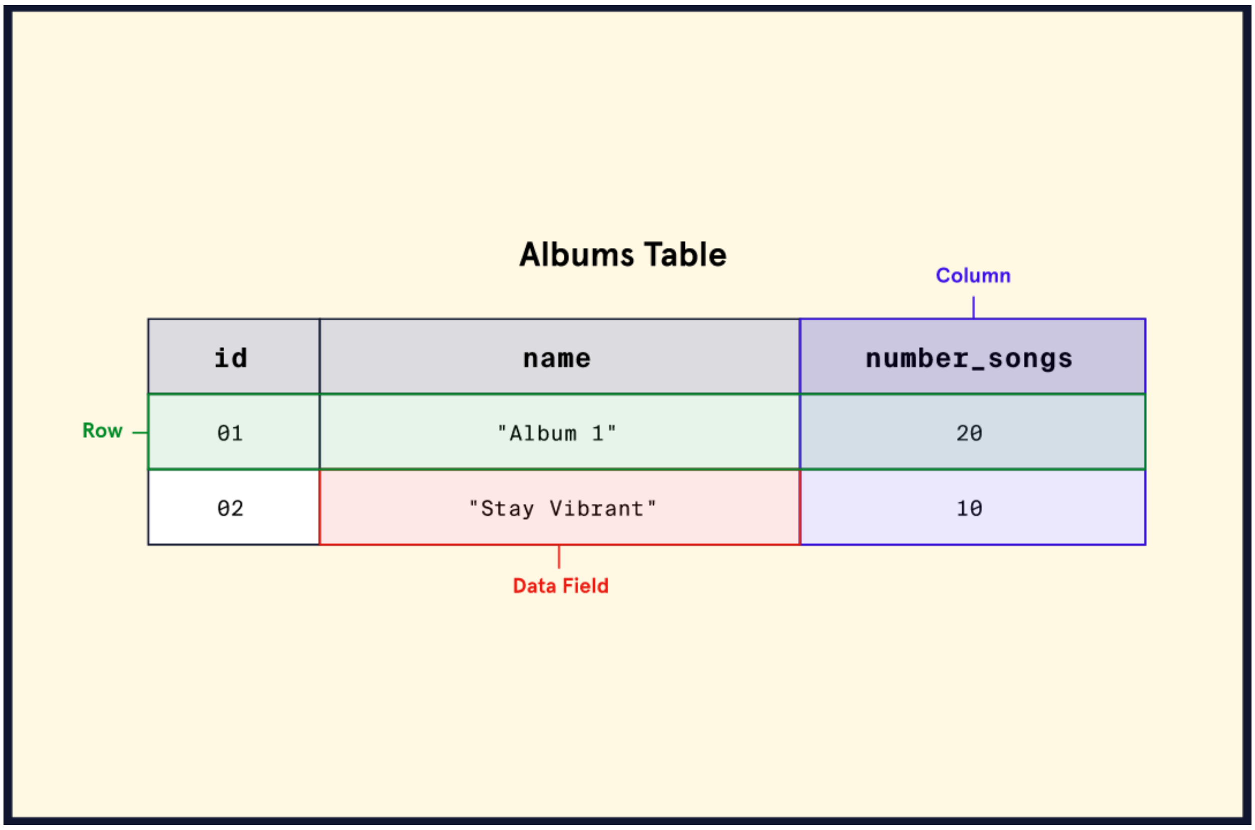


Note: Most information on the internet doesn't do a great job distinguishing between a database and a DBMS. In this article, when we talk about a database, we also inherently refer to its associated DBMS.
When working with a DBMS, we won't only have the ability to store multiple databases but also will be able to capitalize on its unique features for maintaining data. Additionally, each DBMS allows the database it manages to store different types of data. This means we can store a variety of data types such as strings of text, numeric data (integers, decimals, and floating-point numbers), date and time types, and booleans. We can also store more unique data like images and audio files (although it's worth noting a database transforms all data into binary and won't natively know its an image or audio file). The ability to store this variety of data allows DBMSs to have a variety of use cases. Whether we need to store simple data like user information (e.g., email, name, password) or more complex data like videos, a DBMS can likely handle it!

Since each DBMS is unique, they vary in their implementation and, as such, provide different advantages and disadvantages. Let's get familiar with two of the most common types of databases (and their associated DBMS) we're likely to encounter and examine the greatest distinctions in how they store data and operate.

Relational Databases

One of the most common classes of databases is a **relational database**. Relational databases, commonly referred to as SQL databases (more on this later), structure data in **tabular form**. This means data is grouped into tables, using rows and columns to organize and store individual records of data. Here is what the general structure looks like:



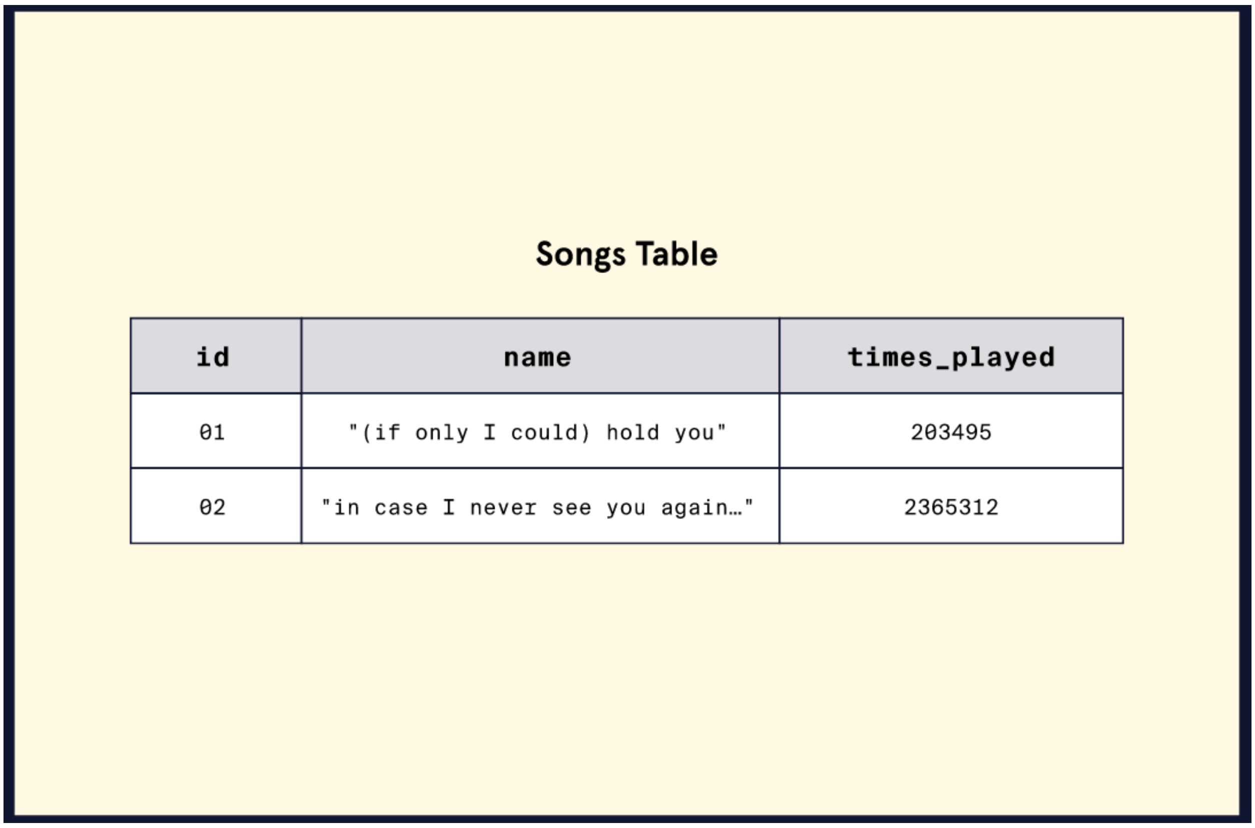
Albums Table

id	name	number_songs
01	"Album 1"	20
02	"Stay Vibrant"	10

Column
Row
Data Field

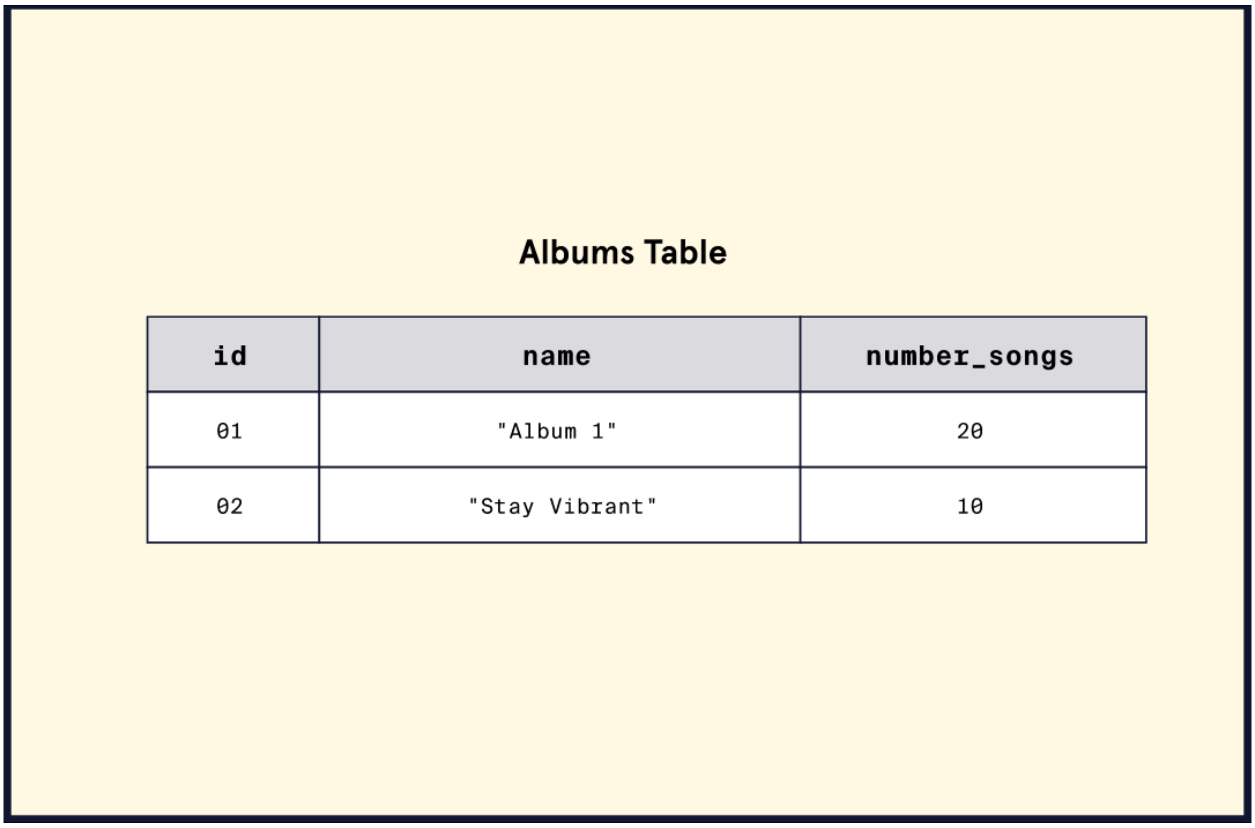
Let’s apply the idea of a relational database to a concrete example. Imagine we launched a music streaming website where users find information about artists and their catalog of songs. How might we organize and store this content in a relational database?

We’d probably have one table that stores information about albums. Within the albums table, we’d have several columns to store specific information about each album, such as its title and release year. Additionally, we could store songs, and artist information in separate tables. Here is what our tables might look like:



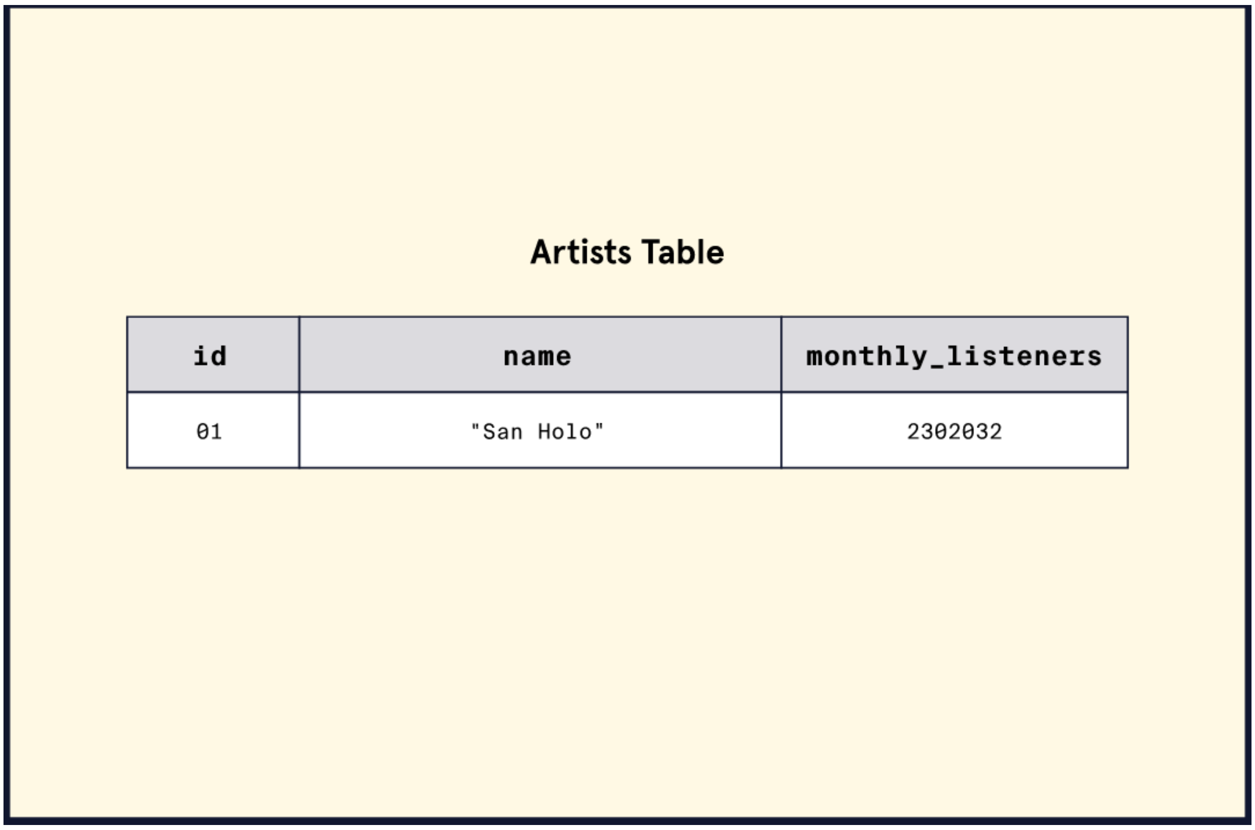
Songs Table

id	name	times_played
01	"(if only I could) hold you"	203495
02	"in case I never see you again…"	2365312



Albums Table

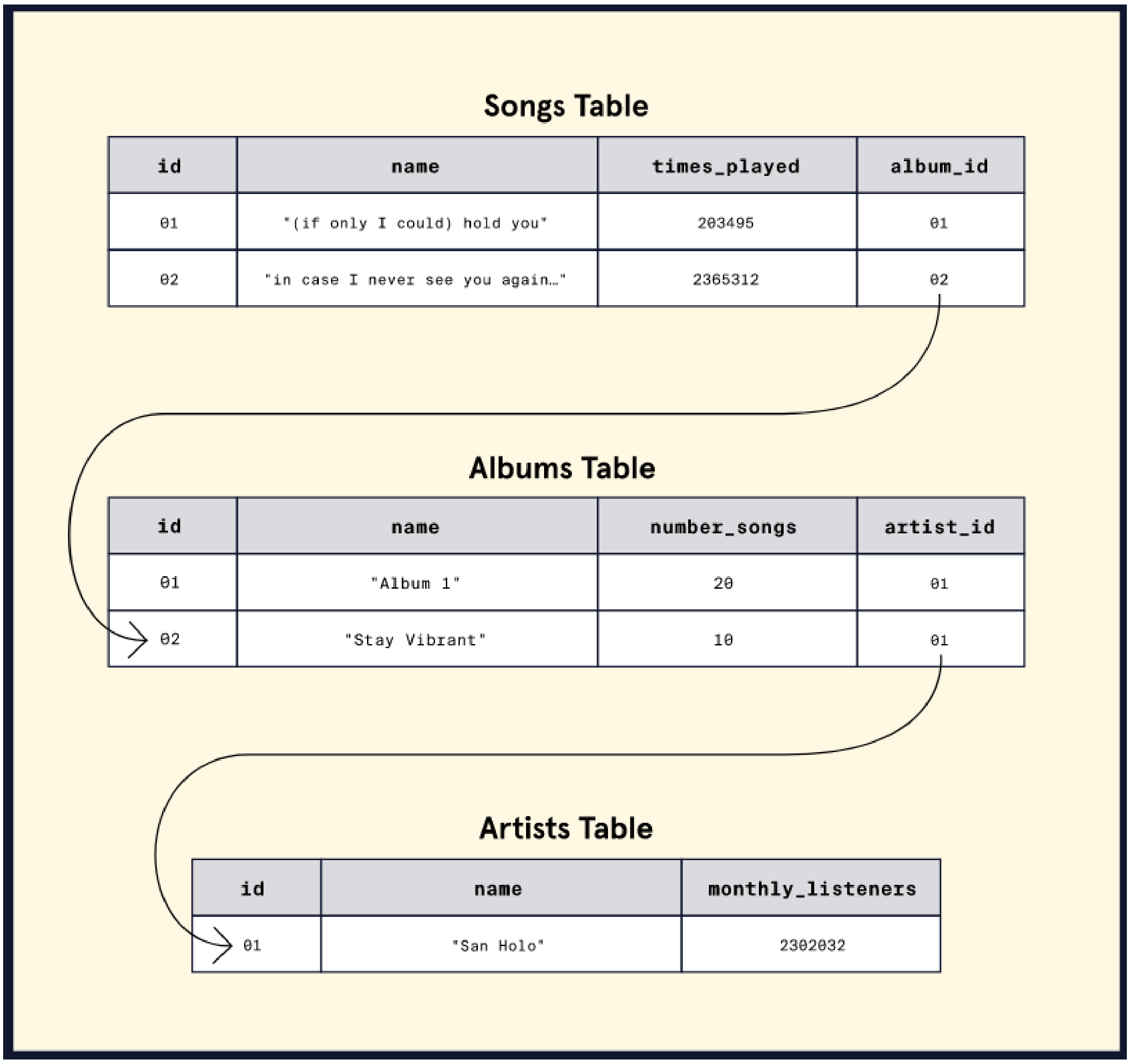
id	name	number_songs
01	"Album 1"	20
02	"Stay Vibrant"	10



Artists Table

id	name	monthly_listeners
01	"San Holo"	2302032

Relational databases are unique because they are based on presenting data in terms of relationships. To accomplish this, relational databases enable associations between tables to be defined, the most common associations being “one-to-one”, “one-to-many”, and “many-to-many”. For example, in our music database, we could establish the following relationships between our data:



Songs Table

id	name	times_played	album_id
01	"(if only I could) hold you"	203495	01
02	"in case I never see you again…"	2365312	02

Albums Table

id	name	number_songs	artist_id
01	"Album 1"	20	01
02	"Stay Vibrant"	10	01

Artists Table

id	name	monthly_listeners
01	"San Holo"	2302032

In the above image, we have established two relationships:

- A relationship between the Songs and Albums table. A song "belongs" to an album via the `album_id` field.
- A relationship between the Albums and Artists table. An album "belongs" to an artist via the `artist_id` field.

Lastly, note the following important properties of relational databases:

- **Pre-defined Schema**: Relational databases are unique because the **database schema** - the "blueprint" of the database structure - is typically determined before any data is ever inserted. This would mean we would likely decide the specific tables, and their associated relationships, before even inserting any data into them.
- **SQL Use**: Developers communicate with a relational database using **SQL** (Structured Query Language). SQL is an industry-standard database language that has been used for decades. Extensive documentation and readable

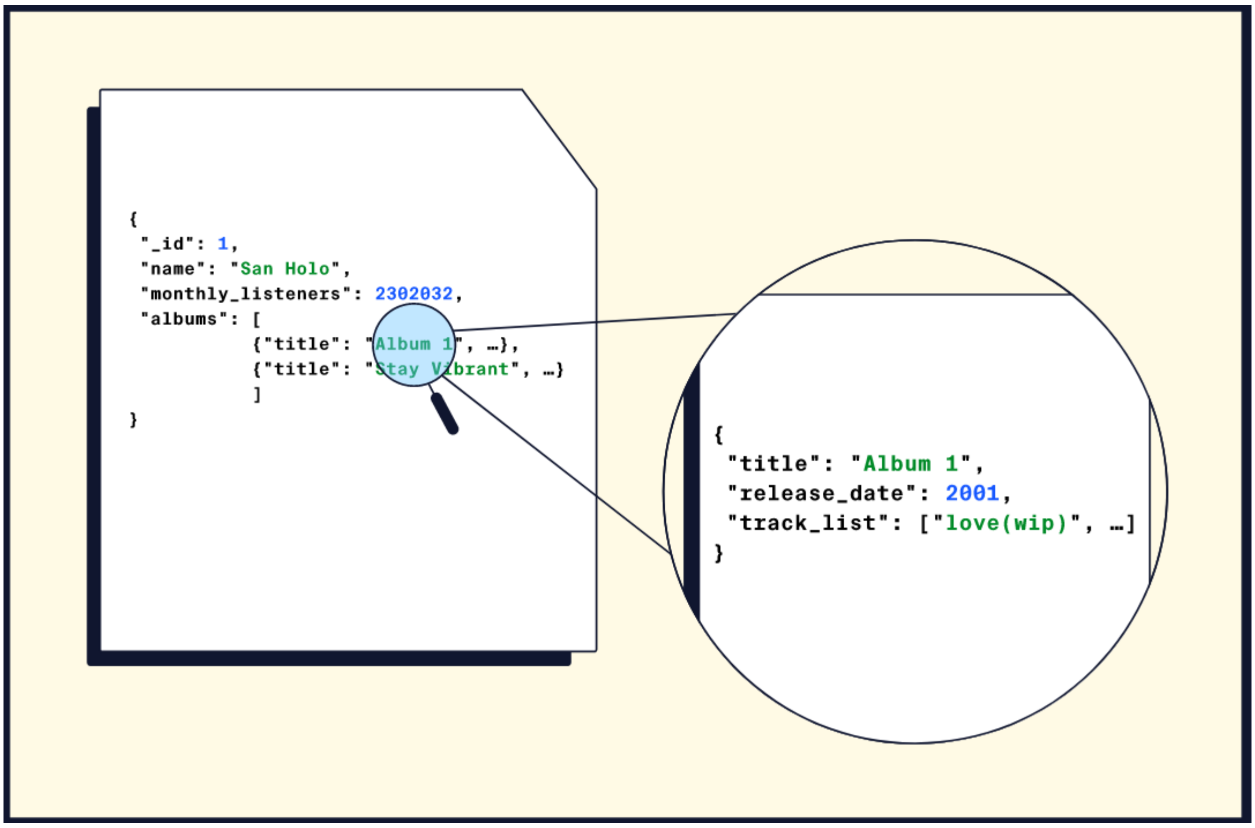
syntax make it approachable for beginners. The dependence of relational databases on SQL is why some developers and documentation sometimes refer to relational databases as SQL databases.

- **Relational Database Management System**: Any relational database is managed by a relational database management system (RDBMS). This type of DBMS allows the data to follow a relational model (e.g., setup relationships) and manage the data using SQL. Two of the most popular RDBMSs are PostgreSQL and MySQL.
- **Unique Disadvantages**: At the enterprise level, where data sets are massive, setting up a relational database can be costly, and the expenses required to maintain and scale it can compound significantly over time. Furthermore, rows and columns consume a great deal of physical space which can lead to implications for performance and cost.

Non-Relational Databases

The second most common class of databases is **non-relational databases**. A non-relational database, commonly referred to as a NoSQL database, is any database that does not follow the relational model. This means these types of databases typically don't store data in tables, but more importantly, data isn't strictly represented with relationships. Under the umbrella of non-relational databases are many different types of databases, each with its own framework for organizing data. Some examples are document databases, graph databases, and key-value databases. Collectively, non-relational databases specialize in storing unstructured data that doesn't fit neatly into rows and columns.

To visualize the difference, let's return to our music database and examine what the data would look like in a NoSQL format. Here is the same database but stored inside of a document database - a special category of NoSQL database where data is stored as JSON:

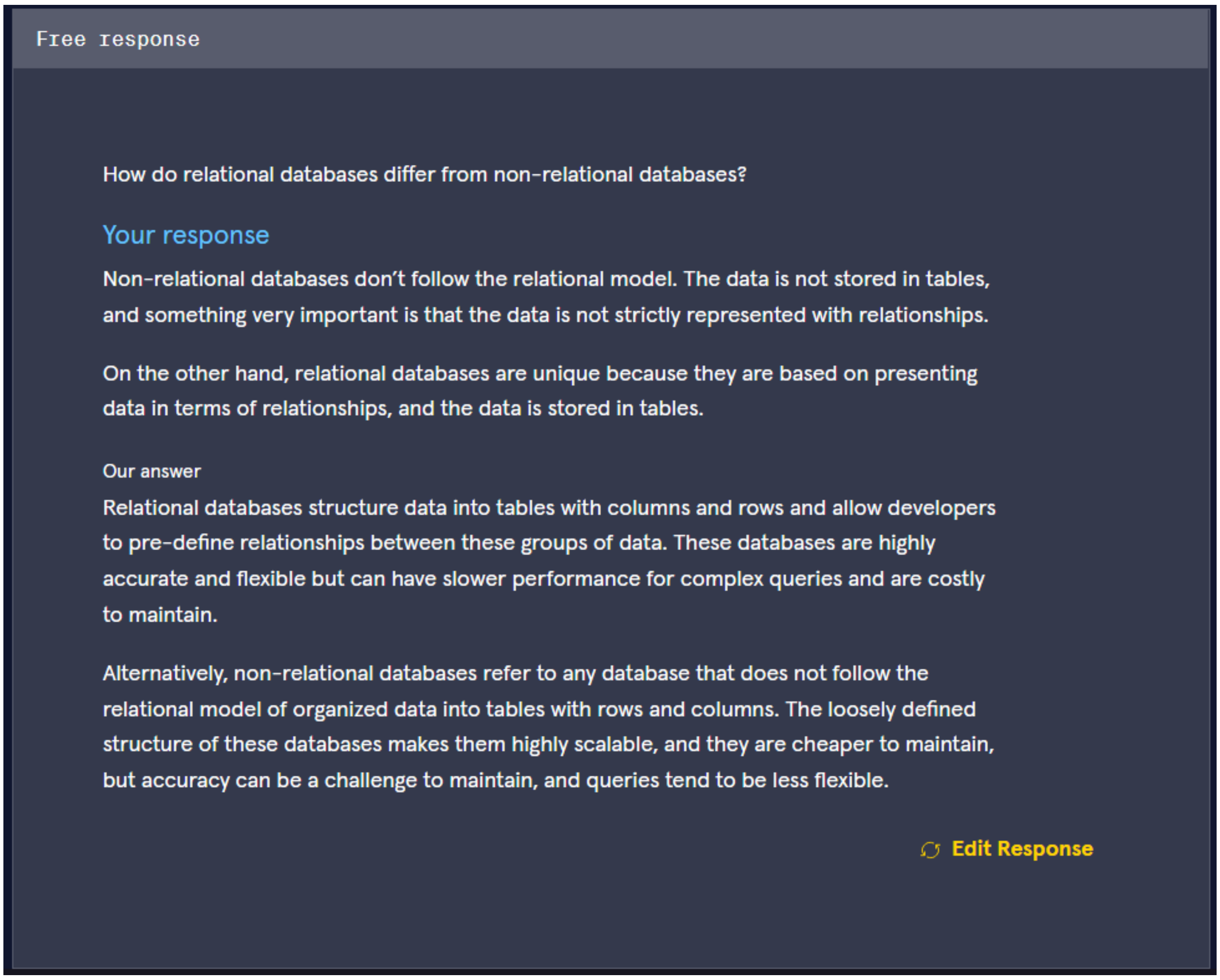


```
{
 "_id": 1,
 "name": "San Holo",
 "monthly_listeners": 2302032,
 "albums": [
           {"title": "Album 1", …},
           {"title": "Stay Vibrant", …}
           ]
}
```

```
{
 "title": "Album 1",
 "release_date": 2001,
 "track_list": ["love(wip)", …]
}
```

Additionally, note the following properties of non-relational databases:

- **Flexibility and Scalability**: Non-relational database's unstructured nature facilitates the design of flexible schemas (schemas that do not need to be defined beforehand) and makes these types of databases highly adaptable to the changing needs of an application. Additionally, non-relational databases are well suited for expansion or scalability and are relatively inexpensive to maintain compared to relational databases.
- **Custom Query Language**: Unlike relational databases that all use SQL as a standard query language, most NoSQL databases have their own custom language.
- **Unique Disadvantages**: Since the data in non-relational databases is mainly unstructured, data can often become hard to maintain and keep track of. Additionally, since every NoSQL database uses its own custom query language, there is a new learning curve for each one we choose to work with.


Free response

How do relational databases differ from non-relational databases?

Your response

Non-relational databases don't follow the relational model. The data is not stored in tables, and something very important is that the data is not strictly represented with relationships.

On the other hand, relational databases are unique because they are based on presenting data in terms of relationships, and the data is stored in tables.

Our answer

Relational databases structure data into tables with columns and rows and allow developers to pre-define relationships between these groups of data. These databases are highly accurate and flexible but can have slower performance for complex queries and are costly to maintain.

Alternatively, non-relational databases refer to any database that does not follow the relational model of organized data into tables with rows and columns. The loosely defined structure of these databases makes them highly scalable, and they are cheaper to maintain, but accuracy can be a challenge to maintain, and queries tend to be less flexible.

Wrap Up

In this article, we familiarized ourselves with what databases are and two of their most common implementations. Let's take a moment to recap what we've learned:

- Databases
  - Databases are systems that store, modify, and access structured collections of information electronically.
  - Database Management Systems (DBMS) allow developers to communicate via code or a graphical user interface with a database.
  - Databases can store a wide range of data types, including text, numbers, dates and times, and files of various types.
- Relational Databases
  - Relational databases, a common database class, organize data into tables of rows and columns of information and rely on relationships to organize data.
  - A relational database schema is typically pre-defined before data is entered.
  - All relational databases use SQL to allow developers to communicate with the database using a common language.

    - Relational databases can be costly to set up and scale. Performance and cost are big factors in using a relational database in an application.
- Non-Relational Databases
    - Non-relational databases, another common database class, refer to any database that does not follow the relational model.
    - Non-relational databases typically have a more flexible schema and are more easily scaled than relational databases. However, the unstructured nature of the data can make it difficult to maintain, and each non-relational database has its own query language.

In the world of databases, there are many options for how our data can be stored. To continue learning, check out the following resources:

- Comprehensive list of database classifications
- A deeper dive into NoSQL vs SQL