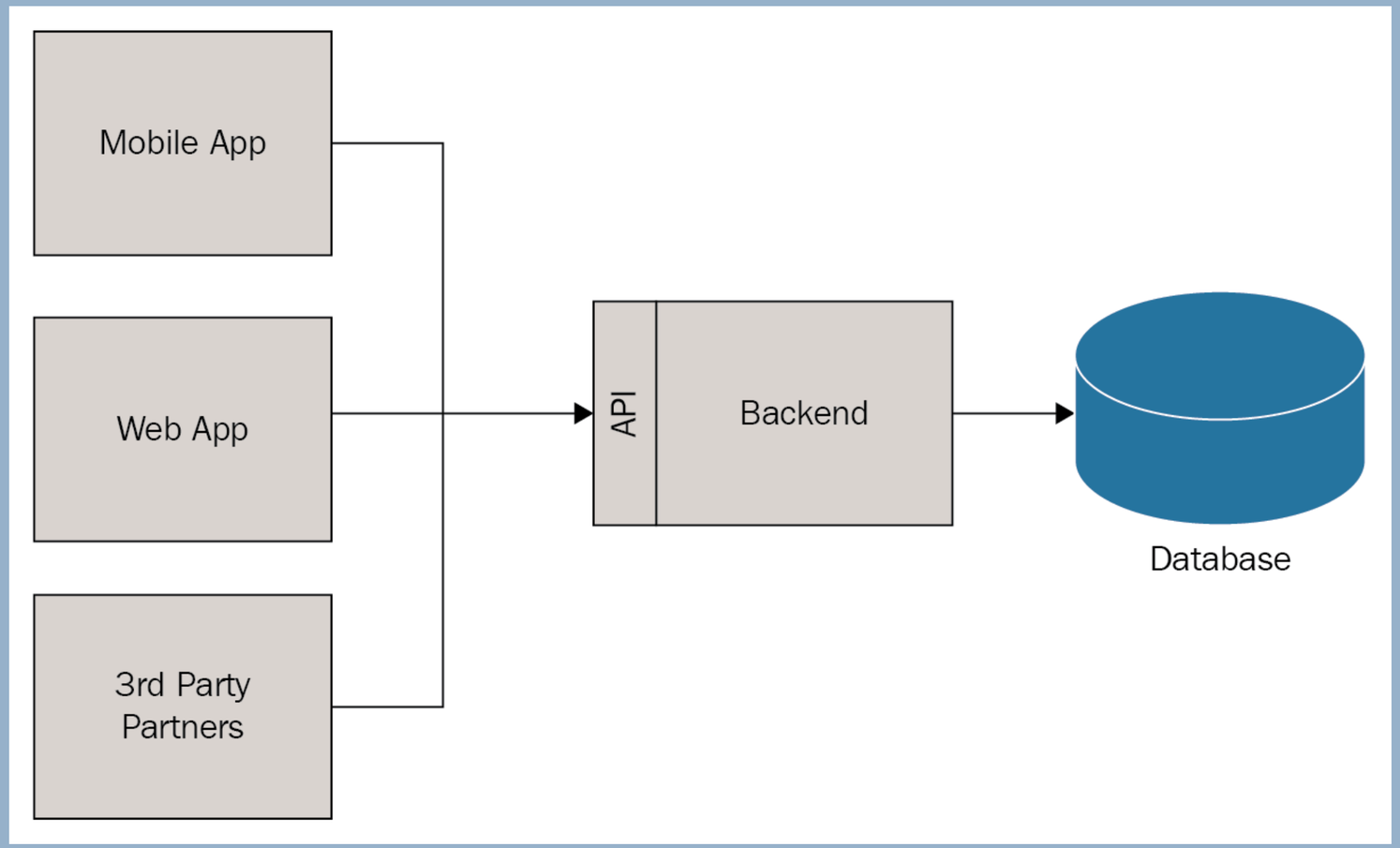
Backend Development

DATABASE & SQL

Guide + Solution SQL

This is database

Why we need to learn Database



What is Database

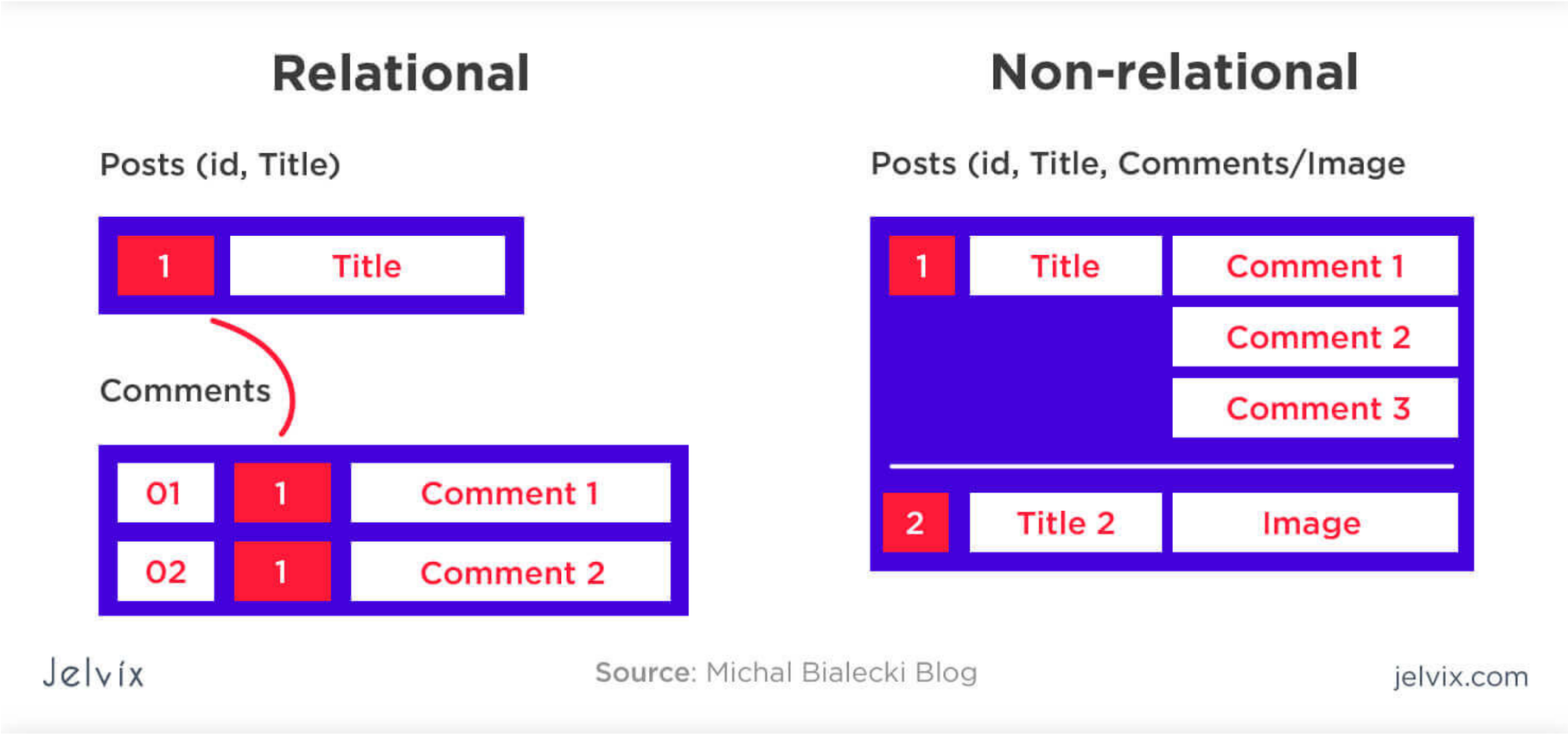
- **A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.**

Types of databases

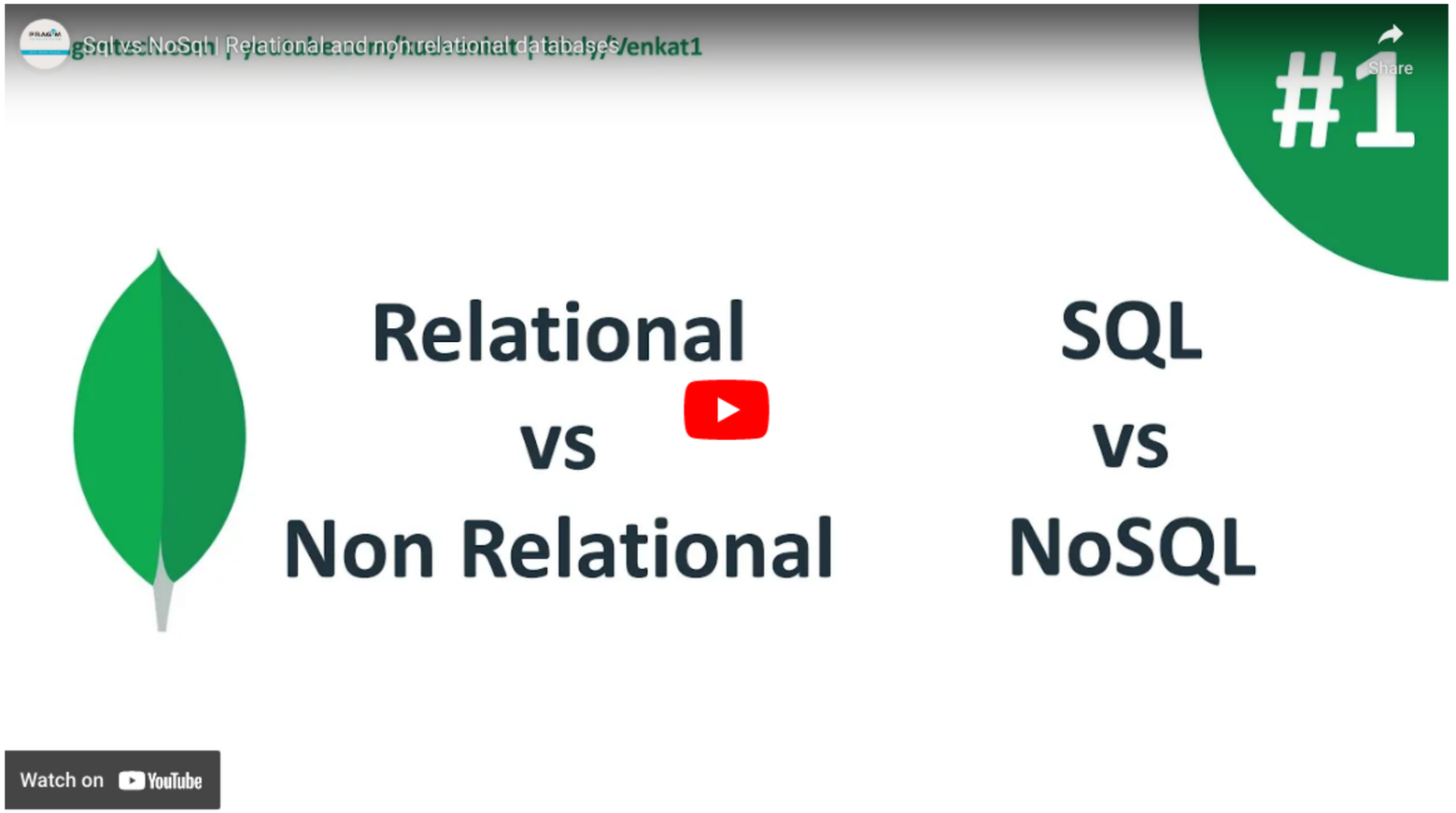
- **Relational databases**
  - **Relational databases became dominant in the 1980s. Items in a relational database are organized as a set of tables with columns and rows. Relational database technology provides the most efficient and flexible way to access structured information.**

- **Non-relational database**
  - **A non-relational database is a database that does not use the tabular schema of rows and columns found in most traditional database systems. Instead, non-relational databases use a storage model that is optimized for the specific requirements of the type of data being stored. For example, data may be stored as simple key/value pairs, as JSON documents, or as a graph consisting of edges and vertices.**

Example


Relational
Posts (id, Title)
1
Title
Comments
01
1
Comment 1
02
1
Comment 2
Non-relational
Posts (id, Title, Comments/Image
1
Title
Comment 1
Comment 2
Comment 3
2
Title 2
Image
Jelvix
Source: Michal Bialecki Blog
jelvix.com

You can learn more

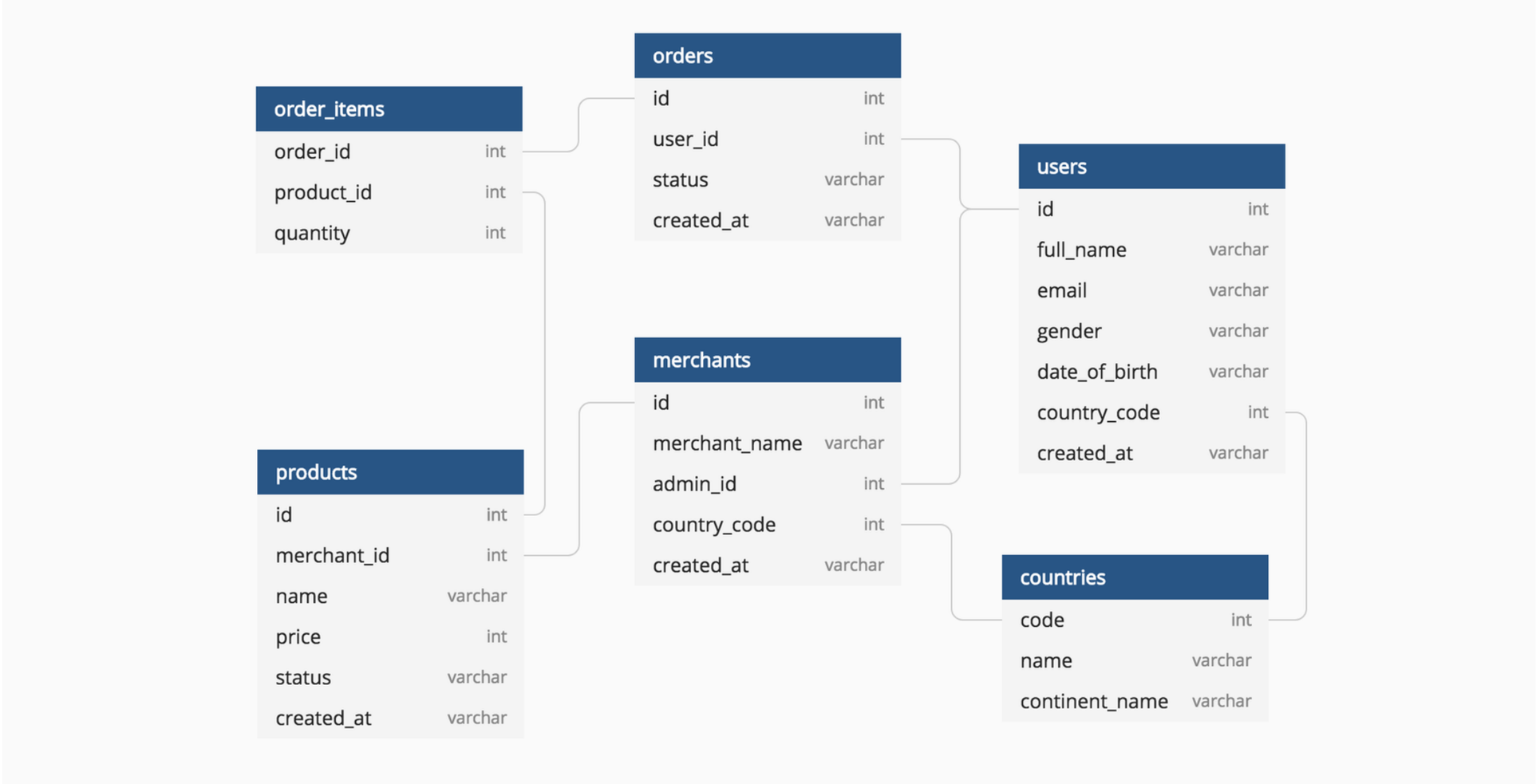


Database design

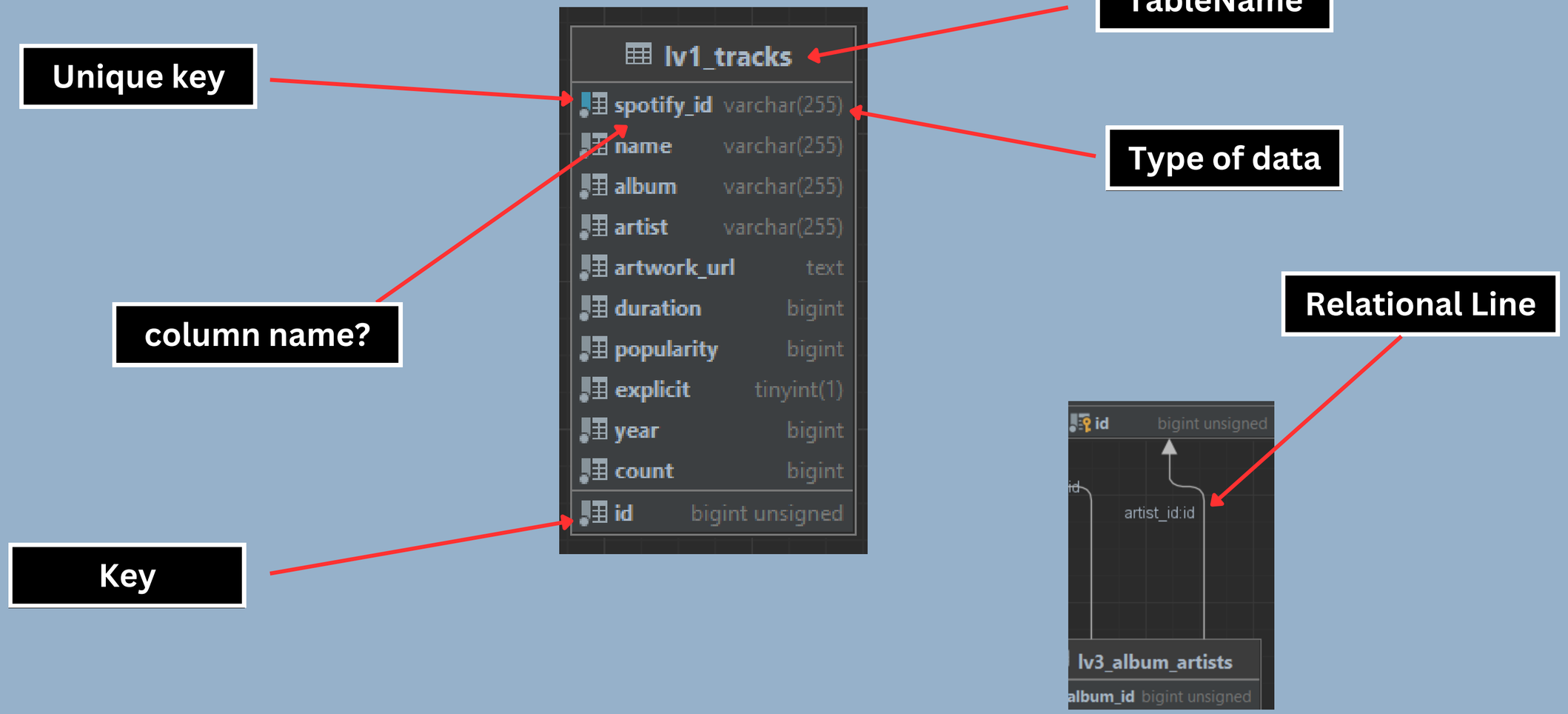
- **What is Database Design?**
  - **Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data into the database model. A database management system manages the data accordingly.**

- **What is a database schema?**
  - **A database schema defines how data is organized within a relational database; this is inclusive of logical constraints such as table names, fields, data types, and the relationships between these entities. Schemas commonly use visual representations to communicate the architecture of the database, becoming the foundation for an organization’s data management discipline. This process of database schema design is also known as data modeling.**

Example


order_items
order_id int
product_id int
quantity int
orders
id int
user_id int
status varchar
created_at varchar
users
id int
full_name varchar
email varchar
gender varchar
date_of_birth varchar
country_code int
created_at varchar
merchants
id int
merchant_name varchar
admin_id int
country_code int
created_at varchar
products
id int
merchant_id int
name varchar
price int
status varchar
created_at varchar
countries
code int
name varchar
continent_name varchar

Component


TableName
Unique key
lv1_tracks
spotify_id varchar(255)
name varchar(255)
album varchar(255)
artist varchar(255)
artwork_url text
duration bigint
popularity bigint
explicit tinyint(1)
year bigint
count bigint
id bigint unsigned
Type of data
column name?
Relational Line
id bigint unsigned
artist_id:id
lv3_album_artists
album_id bigint unsigned
Key

Key

- **What are the Keys in DBMS?**
  - A key in DBMS is an attribute or a set of attributes that help to uniquely identify a tuple (or row) in a relation (or table). Keys are also used to establish relationships between the different tables and columns of a relational database.

- **Why are the Keys Required?**
  - A key is used in the definitions of various kinds of integrity constraints. A table in a database represents a collection of records or events for a particular relation. Now there can be thousands and thousands of such records, some of which may be duplicated.

- Types of Keys in DBMS
  - There are broadly seven types of keys in DBMS. All these types of keys in SQL must be implemented appropriately for the relevant database to negate redundancy. Correct identification will lead to database accuracy, improving results in a limited time. Let’s explore these DBMS keys to learn more about what are keys in SQL.

Key

- **7 Types of Keys**
  - Primary Key
    - the primary key is a column of a table or a set of columns that helps to identify every record present in that table uniquely.
  - Super Key
    - the primary key is a column of a table or a set of columns that helps to identify every record present in that table uniquely.
  - Candidate Key
    - Candidate keys are those attributes that uniquely identify rows of a table.
  - Alternate Key
    - Candidate keys are those attributes that uniquely identify rows of a table.
  - Foreign Key
    - Foreign Key is used to establish relationships between two tables.
    - A foreign key will require each value in a column or set of columns to match the Primary Key of the referential table.
  - Composite Key
    - A composite Key is a set of two or more attributes that help identify each tuple in a table uniquely.
  - Unique Key
    - A composite Key is a set of two or more attributes that help identify each tuple in a table uniquely.

DataType

String Data Types

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum string length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The *size* parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The *size* parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

DataType

Numeric Data Types

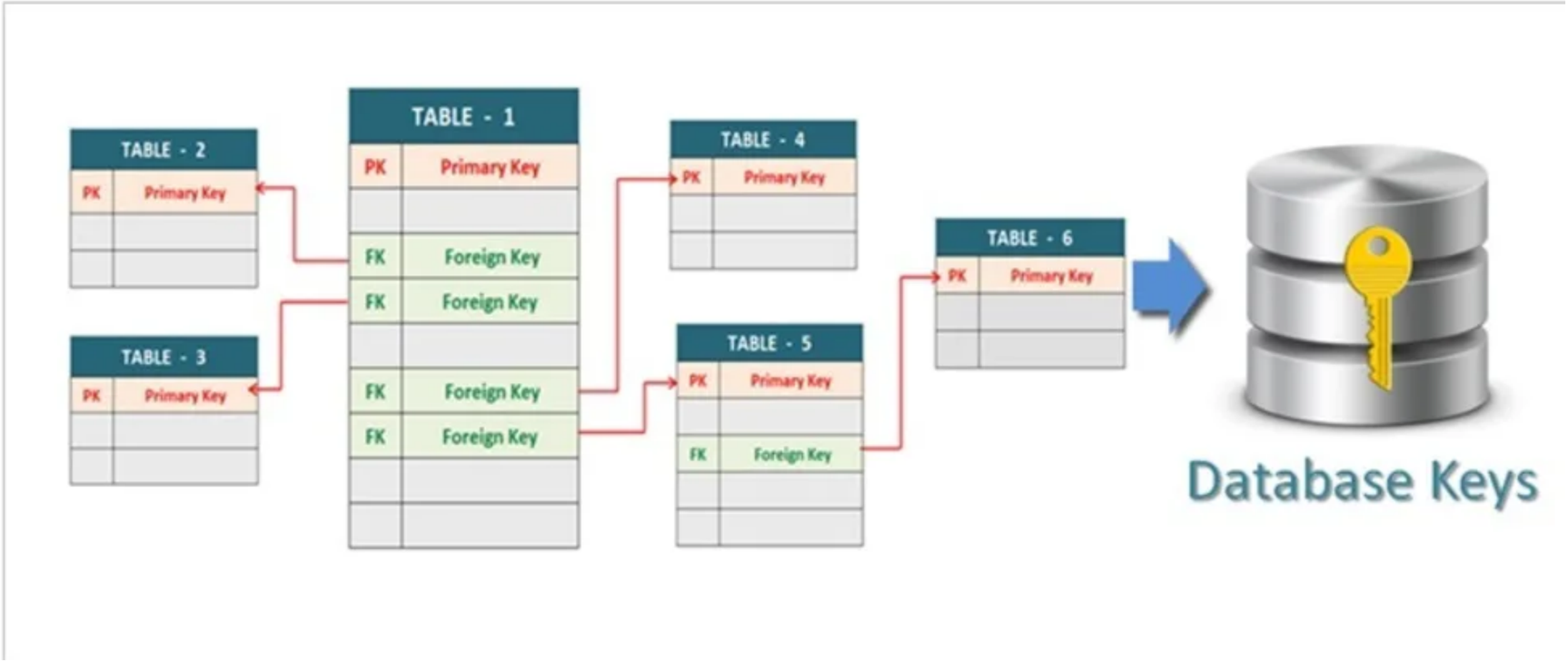
Data type	Description
BIT(*size*)	A bit-value type. The number of bits per value is specified in *size*. The *size* parameter can hold a value from 1 to 64. The default value for *size* is 1.
TINYINT(*size*)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The *size* parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(*size*)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The *size* parameter specifies the maximum display width (which is 255)
MEDIUMINT(*size*)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The *size* parameter specifies the maximum display width (which is 255)
INT(*size*)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)
INTEGER(*size*)	Equal to INT(size)
BIGINT(*size*)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255)
FLOAT(*size*, *d*)	A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(*p*)	A floating point number. MySQL uses the *p* value to determine whether to use FLOAT or DOUBLE for the resulting data type. If *p* is from 0 to 24, the data type becomes FLOAT(). If *p* is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(*size*, *d*)	A normal-size floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter
DOUBLE PRECISION(*size*, *d*)	
DECIMAL(*size*, *d*)	An exact fixed-point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. The maximum number for *size* is 65. The maximum number for *d* is 30. The default value for *size* is 10. The default value for *d* is 0.
DEC(*size*, *d*)	Equal to DECIMAL(size,d)

DataType

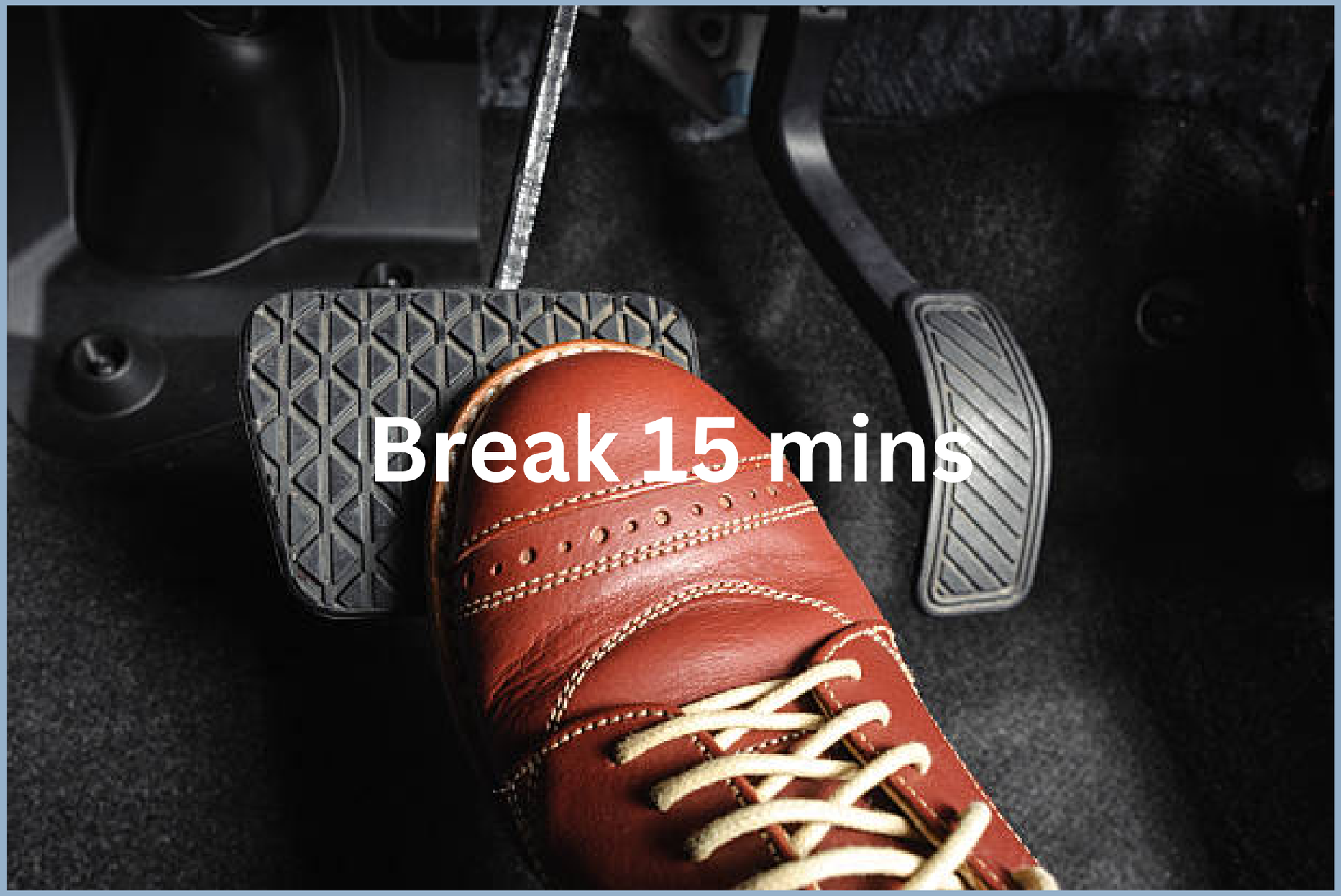
Date and Time Data Types

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(*fsp*)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(*fsp*)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(*fsp*)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

Primary key & Foreign key



Start Trying To Design Database


Break 15 mins


Create Database by SQL

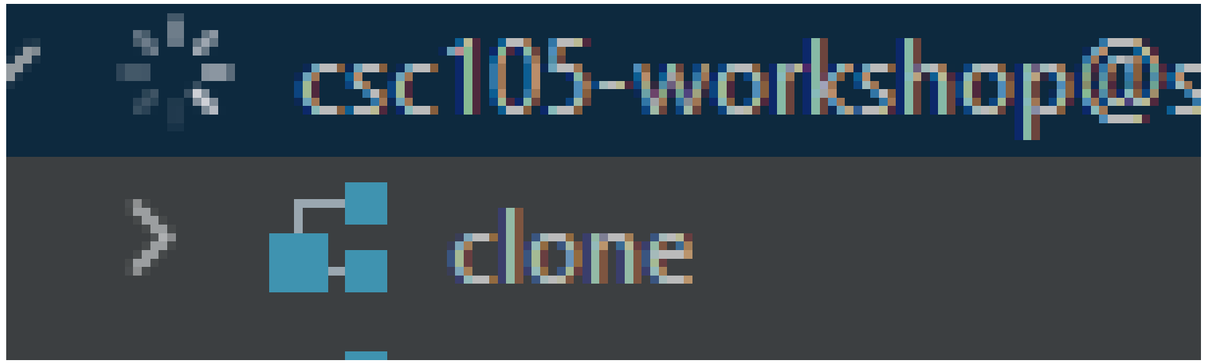
CREATE DATABASE Statement

Syntax
CREATE DATABASE databasename;

EXAMPLE
CREATE DATABASE clone;

```
CREATE DATABASE clone;
```

```
csc105-workshop> CREATE DATABASE clone
```



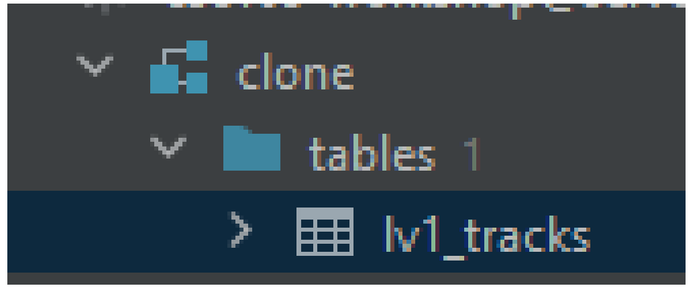
CREATE TABLE Statement

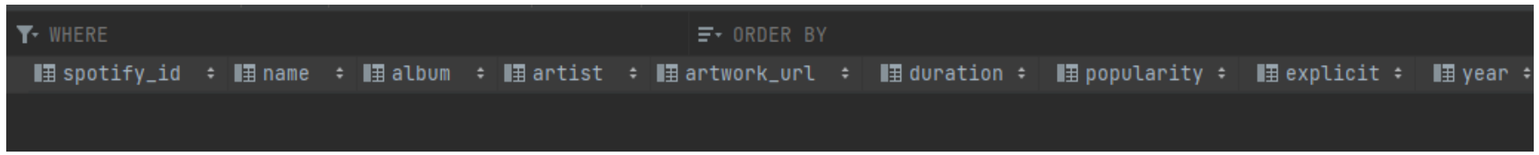
Syntax

```
CREATE TABLE table_name (
  column1 datatype,
  column2 datatype,
  column3 datatype,
  ....
);
```

EXAMPLE

```
create table lv1_tracks
(
   spotify_id  varchar(255)  ,name      varchar(255)  ,
   album       varchar(255)  , artist    varchar(255)  ,
   artwork_url text          , duration  bigint        ,
   popularity  bigint        , explicit  tinyint(1)    ,
   year        bigint        , count     bigint
);
```





SQL NOT NULL on CREATE TABLE

- By default, a column can hold NULL values.
- The NOT NULL constraint enforces a column to NOT accept NULL values.
- This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

Syntax

```
CREATE TABLE Persons (
  ID int NOT NULL,
   LastName varchar(255) NOT NULL,
  FirstName varchar(255) NOT NULL,
  Age int
);
```

EXAMPLE

```
create table testnotnull
( spotify_id  varchar(255) not null,
name       varchar(255) not null,
album      varchar(255) not null
);
```

SQL AUTO INCREMENT Field

- Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.
- Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

Syntax

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
);
```

EXAMPLE

```
create table testnotnull
( spotify_id  varchar(255) not null AUTO_INCREMENT,
name        varchar(255) not null,
album       varchar(255) not null
);
```

SQL AUTO INCREMENT Field

- MySQL uses the AUTO_INCREMENT keyword to perform an auto-increment feature.
- By default, the starting value for AUTO_INCREMENT is 1, and it will increment by 1 for each new record.
- To let the AUTO_INCREMENT sequence start with another value, use the following SQL statement:

Syntax

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
) AUTO_INCREMENT = 261 ;
```

EXAMPLE

```
create table test_increment
( spotify_id  int not null AUTO_INCREMENT,
name        varchar(255) not null,
album       varchar(255) not null
) AUTO_INCREMENT = 55 ;
```

SQL DEFAULT Constraint

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

Syntax

```
CREATE TABLE Persons (
  ID int NOT NULL,
  LastName varchar(255) NOT NULL,
  FirstName varchar(255),
  Age int,
  City varchar(255) DEFAULT 'Sandnes'
);
```

EXAMPLE

```
create table test_default
(spotify_id int not null ,
 name       varchar(255) not null ,
 album      varchar(255) not null,
 count      int not null DEFAULT 0
) ;
```

SQL PRIMARY KEY Constraint

Syntax

```
CREATE TABLE Persons (
  column1 datatype,
  PRIMARY KEY (column1)
);
```

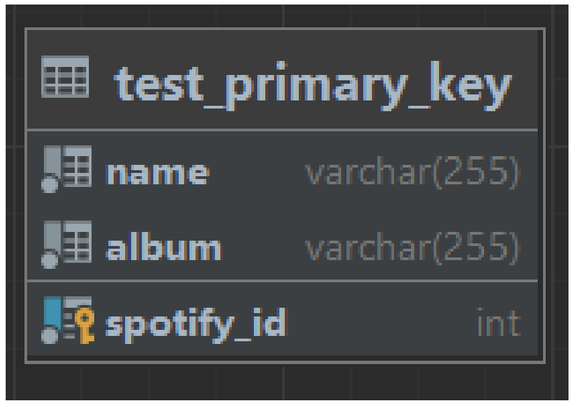
EXAMPLE

```
create table test_primary_key
( spotify_id  int not null AUTO_INCREMENT,
 name       varchar(255) not null,
 album      varchar(255) not null,
 PRIMARY KEY (spotify_id),
   constraint spotify_id
      unique (spotify_id)
);
```

Unique KEY

Syntax

```
CREATE TABLE Persons (
  column1 datatype,
  PRIMARY KEY (column1),
  constraint column1
    unique (column1)
);
```

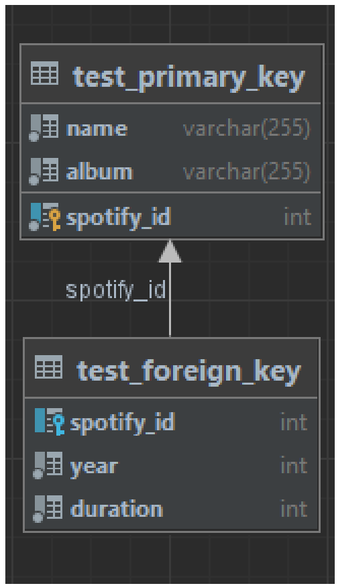


SQL FOREIGN KEY on CREATE TABLE

- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Syntax

```
CREATE TABLE test_foreign_key (
    spotify_id int,
    year int NOT NULL,
    duration int NOT NULL,
    FOREIGN KEY (spotify_id) REFERENCES test_primary_key(spotify_id)
);
```



SQL ALTER TABLE Statement

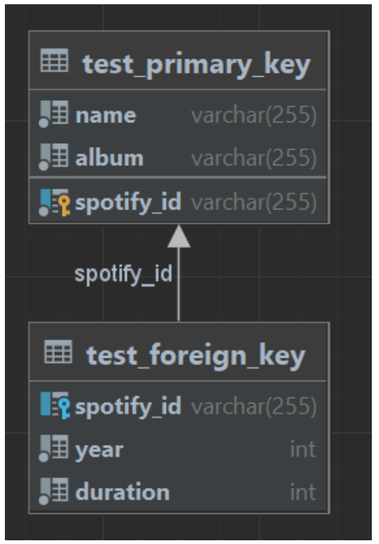
- The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.
- The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

Syntax

```
ALTER TABLE table_name
ADD column_name datatype;
```

Example

```
ALTER TABLE test_foreign_key
ADD count int not null DEFAULT 0;
```



DROP

- The DROP TABLE statement is used to drop an existing table in a database.

Syntax
```
DROP TABLE table_name;
```

Example
```
DROP TABLE testnotnull;
```

- The DROP TABLE statement is used to drop an existing table in a database.

Syntax
```
DROP DATABASE databasename;
```

Example
```
DROP DATABASE clone;
```

See you on Thursday with QUIZ

Content:

- SQL
- DB
- Create DB