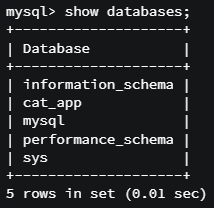
# Creating Databases

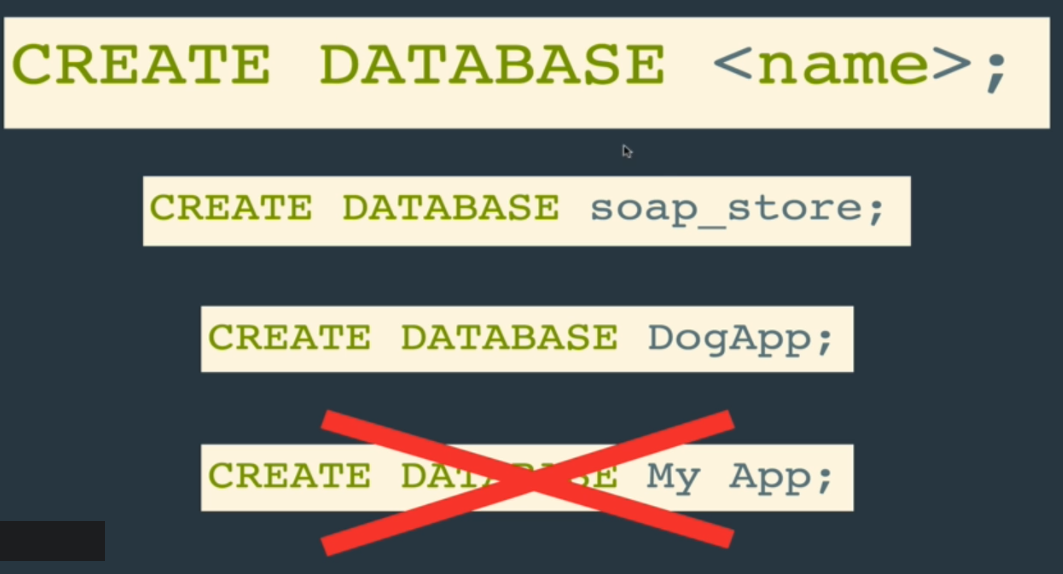
* Section slides: <https://webdev.slides.com/coltsteele/mysql-97-98>
* Graphical representation of a database, where we have a **database server** running on goorm IDE. From here, we can set up actual individual databases within the database server.



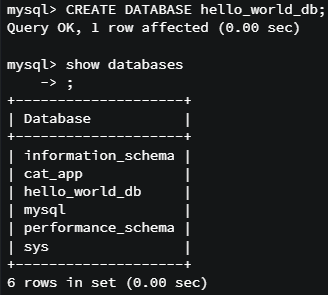
* + Each of these four databases is separate and individual, but are all hosted by the same database server.
  + This is important because some databases may have some of the same information, such as Users who are members of two or more databases. If we didn’t keep the databases separate, then the “steams might cross” so to speak.
* The **show databases**; command shows you all of the databases that exist in the MySQL server



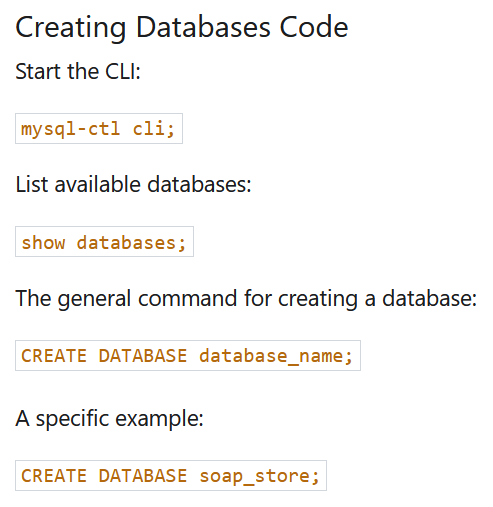
* The **CREATE DATABASE <name>;** command creates a new database with the name defined within the <name> carrots
  + It is recommended to not have standard spaces in database names
  + Stay consistent with how your name databases



* + Example of creating a database:

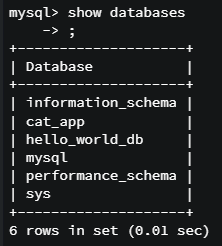
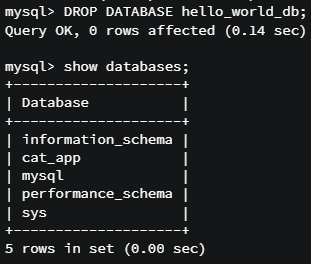


* + - A note on capitalization: you don’t strictly need to capitalize letters in SQL commands, but some people like to do it to differentiate what is coming form SQL versus names of things
* Section summary code:



# Dropping Databases

* The command is **DROP DATABASE <database\_name>**
  + The **DROP** term is the SQL form of “delete”, and will allow us to drop various things including databases and tables

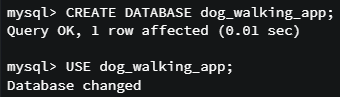
**🡪** 

* + Remember that code will not execute unless it’s followed by a semicolon
* Code summary

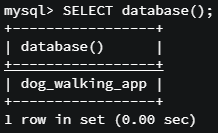


# Using Databases

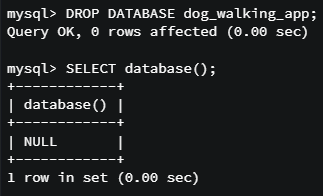
* So far we’ve only created and destroyed databases. Now we have to learn how to *use* them
* The **USE <database name>;** command tells MySQL which database we want to work with



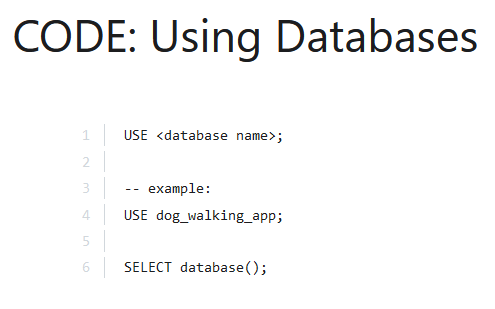
* After running the USE command, we can then use **SELECT database();** to ensure we are using it



* What happens if we drop a database that we’re currently using and then try to select it?



* + Deleting a database that you’re using will result in the **SELECT database();** command returning NULL, as we are no longer using a database
* Code summary



# Introduction to Tables

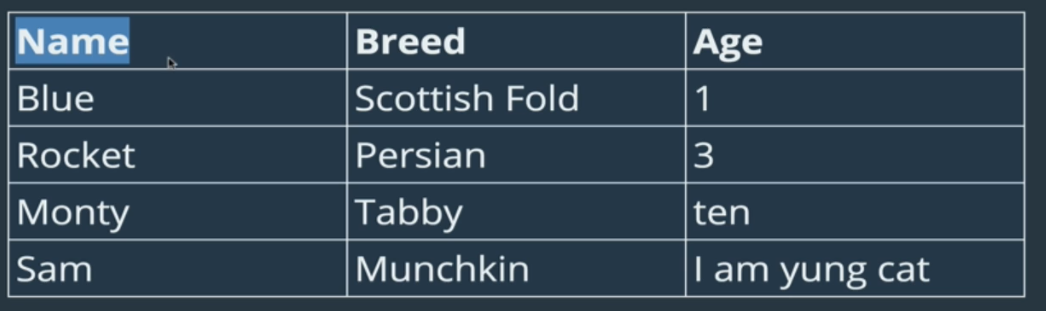
* Tables are at the true heart of SQL. At its core, a relational database is just a bunch of tables that contain data
  + Not every database in the world is composed of tables
* A **table** is a collection of related data held in a structured format within a database
  + Colt’s definition: tables hold the data
* Example of a table using CATS, which consists of three columns – Name, Breed, and Age



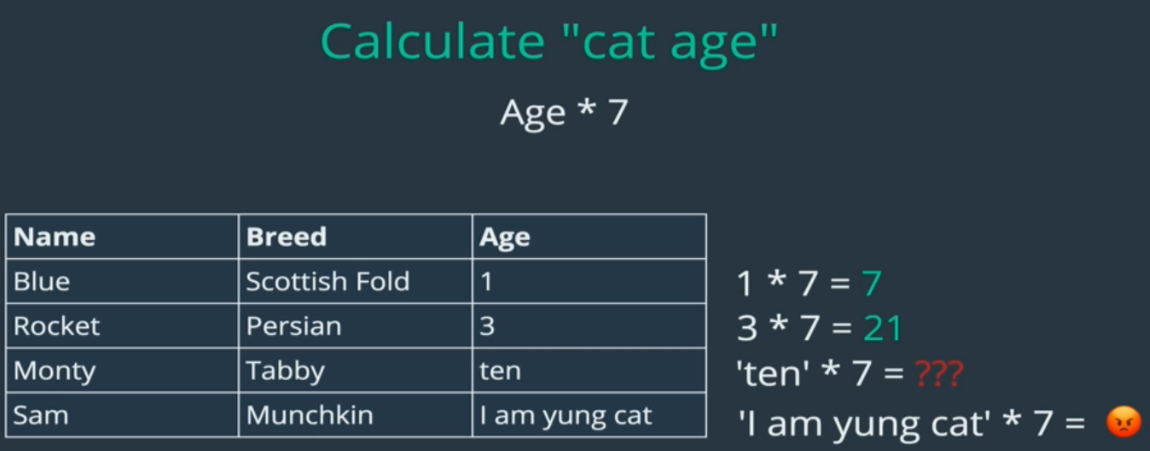
* “Columns” refers to headers of a column. In this case it refers to Name, Breed, and Age
* “Rows” refers to the actual data that is populated row-rise
* Most databases are made up of multiple tables. It is quite rare that a full database consists of only a single table due to the natural complexity of data, especially for such things as applications

# Basic Datatypes

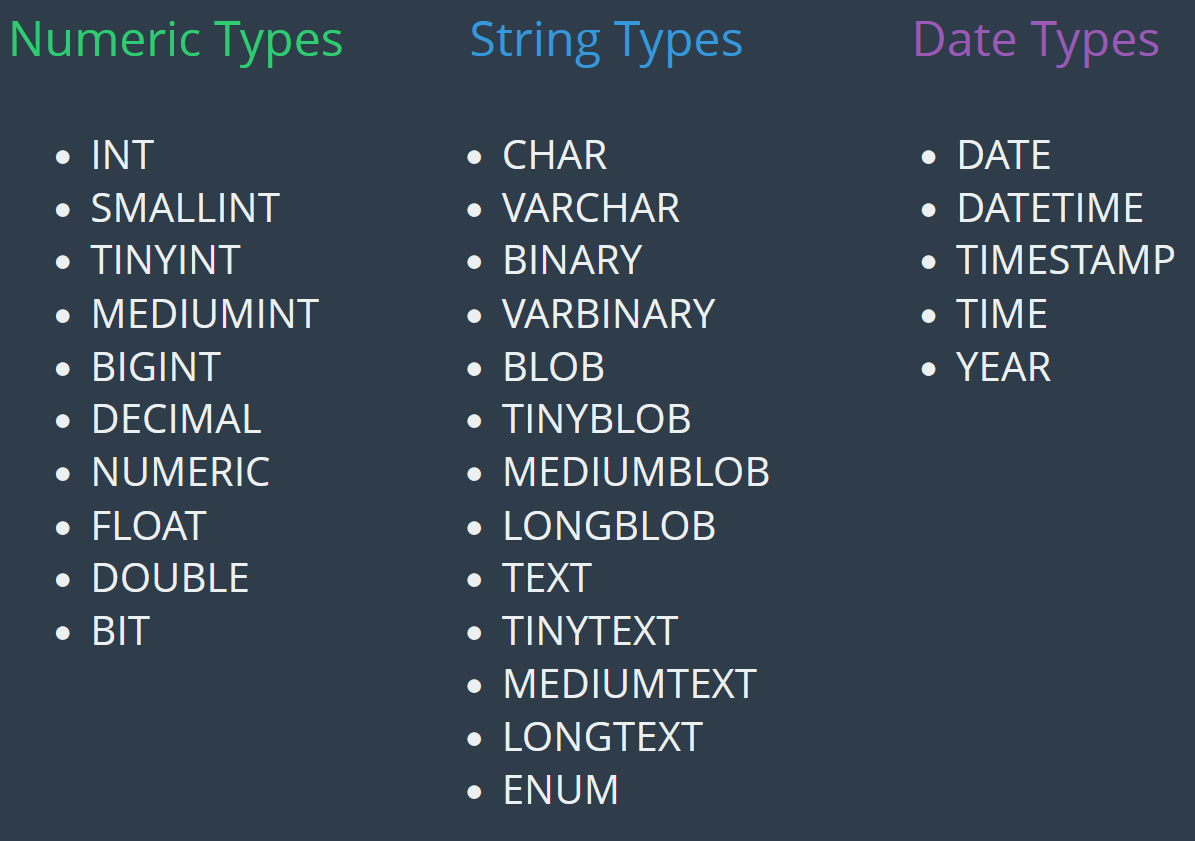
* To understand the importance of datatypes, we return to a modified version of the cats table.



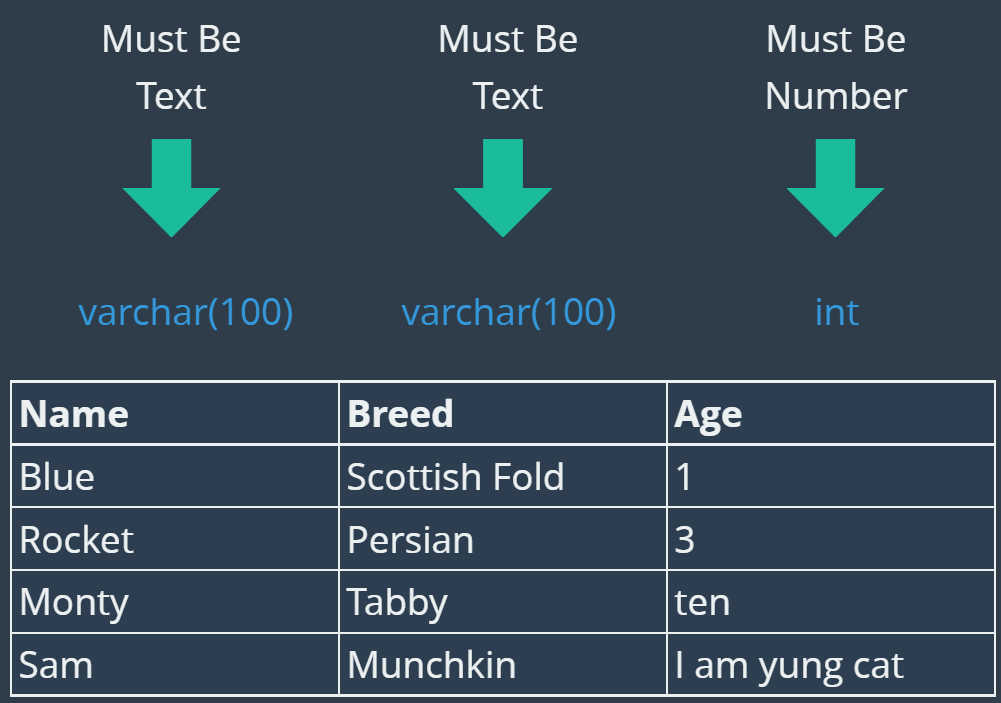
* + Name and Breed consist of simple text, but Age consists of numbers and text. This is actually not allowed in SQL. When creating headers (columns), specifying the datatypes is required
* Suppose we were to calculate “cat age”, which consists of multiplying the human age by 7. We would not be able to do this with the values in the “Age” column because the text cannot be multiplied by 7. MySQL will enforce these datatype rules



* There are a LOT of different MySQL **datatypes** (“text” and “number” are not real datatypes, but we can use these terms loosely to talk about them)

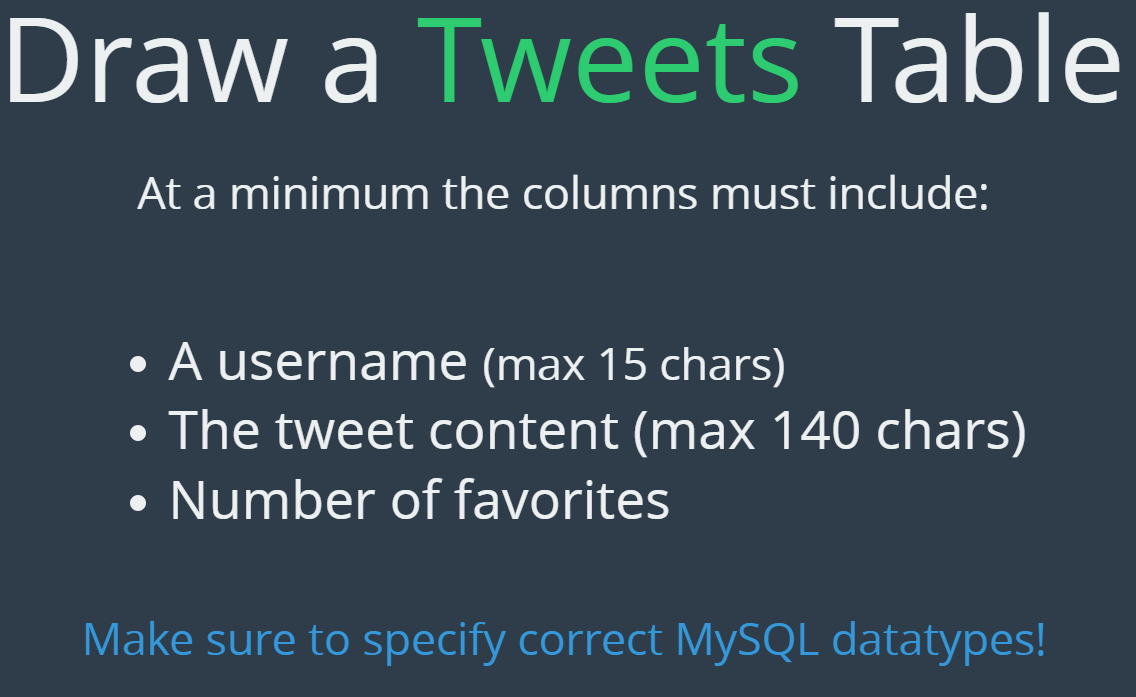


* + It is NOT necessary to know all of these datatypes at any one time. In reality, you use the ones you need depending on your situation. Many datatypes you may use very infrequently or not at all
* **INT** represents whole numbers
  + You cannot use it for decimals
  + It has a maximum value of 4294967295
    - See here for a clarifying explanation of signed and unsigned INT: https://www.udemy.com/course/the-ultimate-mysql-bootcamp-go-from-sql-beginner-to-expert/learn/lecture/8338722#questions/3002798
  + Examples: 1, 4, 3434, 2340985
* **VARCHAR** represents text or strings
  + Stands for variable length of characters
  + Can be between 1 and 255 characters
  + Examples: any length of text from 1 to 255 characters
    - When we declare VARCHAR we specify the maximum character length

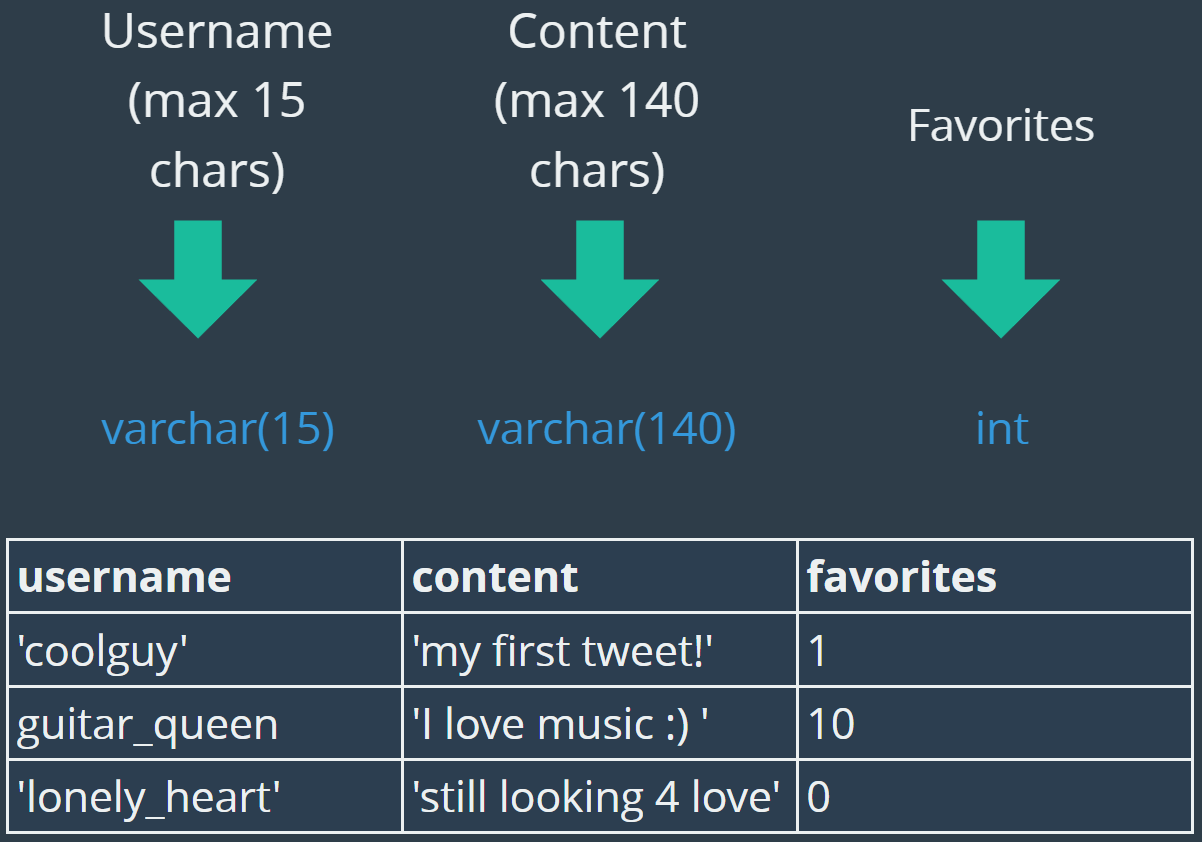


# Basic Datatypes Challenge

* The challenge is to draw up a simple table describing Tweets as follows:
  + Specify the correct MySQL datatypes used for each of these columns

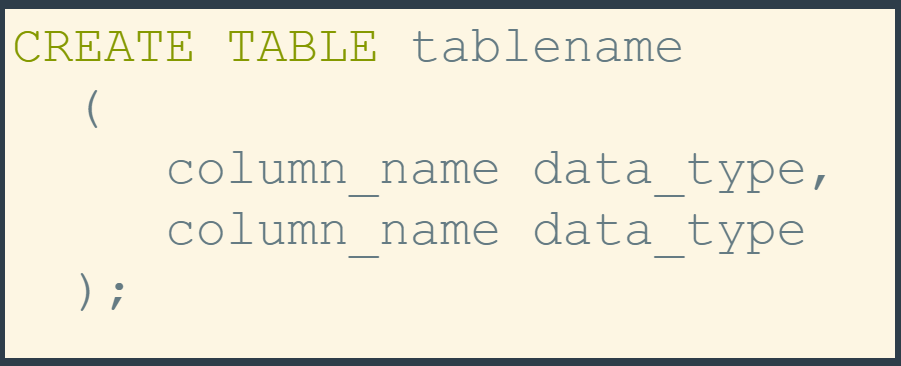


* The solution
  + The username is best served by the datatype **varchar** restricted at 15 characters [varchar(15)]
  + The content is best served by the datatype **varchar** restricted to 140 characters [varchar(140)]
  + The favorites is best served by the datatype **int**

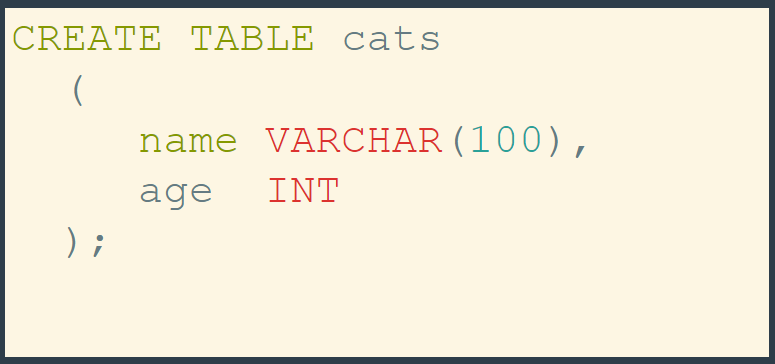


# Creating Your Own Tables

* The code below is the generic, basic version of how to create a table in MySQL
  + This generic example table has two columns
  + Columns are grouped by a set of parentheses and separated by commas
  + It is not required to separate columns with commas, but is aesthetically pleasing

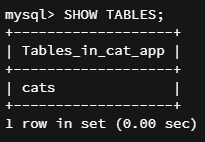


* A more salient example for creating a table called “cats”
  + The convention is to use plural nouns for table names

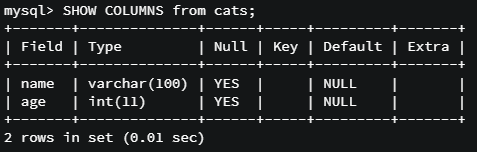


# How Do We Know Our Table Was Created?

* After a table has been created, there are several options for determining whether that creation worked
* One option is to use the **SHOW TABLES;** command. This will show you all of the tables in the current database
  + However, it does not show you anything else about the tables. For instance, you don’t know if the table was created with correct columns

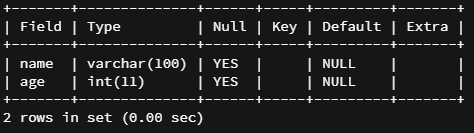


* The **SHOW COLUMNS from <table name>;** command gives you a nice table with each column name, the datatype, and some other data that we’ll discuss in the next section

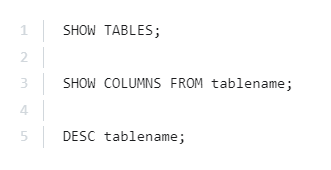


* A shorter version of the above is the **DESC <table name>;** command
  + It is technically a different command, but within this context it accomplishes the same task with less verbiage





* Code summary for checking tables and the columns within



# Dropping Tables

* Tables are deleted using the **DROP TABLE <TABLENAME>;**
  + Note how similar this is to the DROP DATABASEcommand





* To check if your drop worked, you can run SHOW TABLES or DESC <tablename>
  + SHOW TABLES; will return an empty set



* + DESC <tablename> will return an error reporting that the table does not exist



* Be careful when dropping tables – the command is permanent



# Challenge: Creating Your Own Tables

* URL for activity: <http://webdev.slides.com/coltsteele/mysql-97-98?token=owYEcglY#/41>
* Code from challenge

