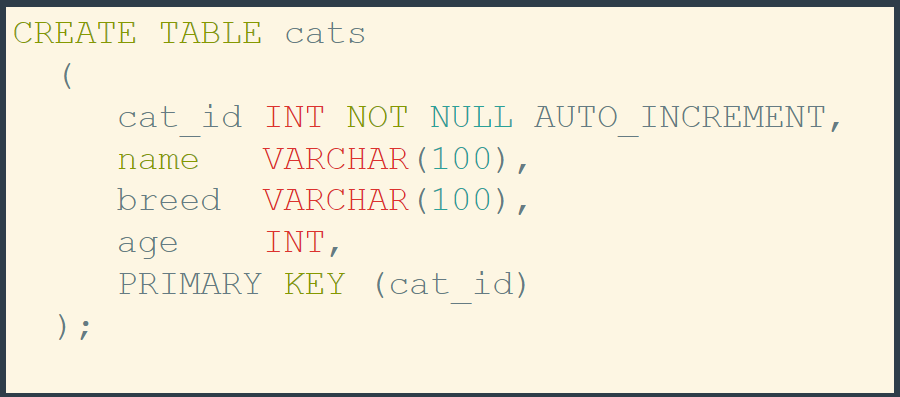
Section slides: <http://webdev.slides.com/coltsteele/mysql-99-100>

Introduction to CRUD

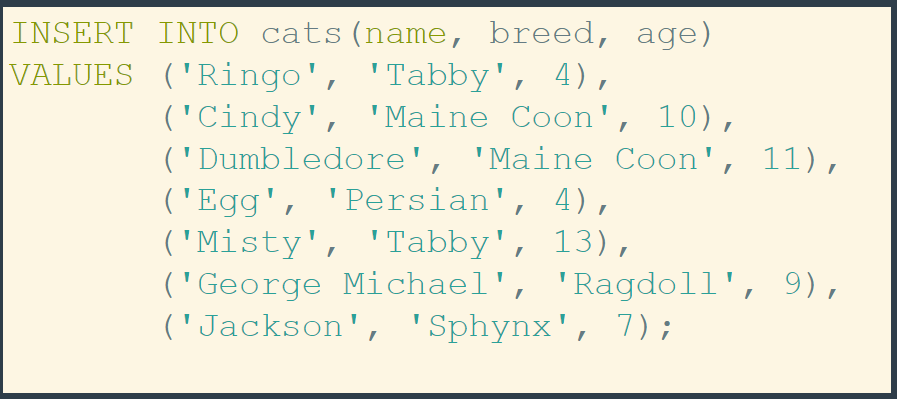
* CRUD stands for **C**reate, **R**ead, **U**pdate, **D**elete
  + These are the four main operations that we perform on database data
  + It also applies to other facets of programming, but we focus on its database applications in this course
* We’ve already seen Create in the context of creating tables and data. In this section we’ll focus on Reading, Updating, and Deleting

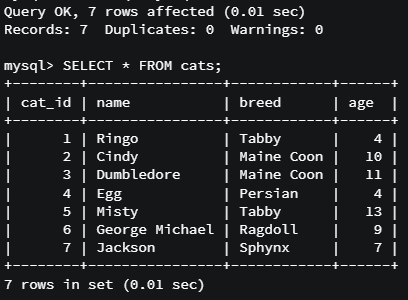
Preparing Our Data

* We begin this section with a clean slate, deleting the original *cats* table and re-adding it with more sophistication
* We create our new table with the following code. Our new *cats* table will have and auto-incrementing ID (primary key), name, breed, and age



* Now let’s add the following data to our *cats* table:





* Code: Preparing Our Data

#### CODE: Preparing Our Data

Let's drop the existing cats table:

DROP TABLE cats;

Recreate a new cats table:

1. CREATE TABLE cats
2. (
3. cat\_id INT NOT NULL AUTO\_INCREMENT,
4. name VARCHAR(100),
5. breed VARCHAR(100),
6. age INT,
7. PRIMARY KEY (cat\_id)
8. );

DESC cats;

And finally insert some new cats:

1. INSERT INTO cats(name, breed, age)
2. VALUES ('Ringo', 'Tabby', 4),
3. ('Cindy', 'Maine Coon', 10),
4. ('Dumbledore', 'Maine Coon', 11),
5. ('Egg', 'Persian', 4),
6. ('Misty', 'Tabby', 13),
7. ('George Michael', 'Ragdoll', 9),
8. ('Jackson', 'Sphynx', 7);

An Official Introduction to the “SELECT” Method

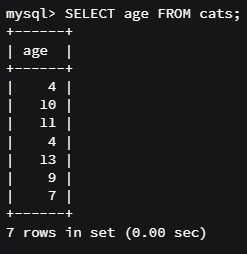
* The magical **SELECT;** command is used to *read* data from a database. We’ve seen it before, but how does it work?
* When we say **SELECT \*;** we are asking for ALL columns in a table



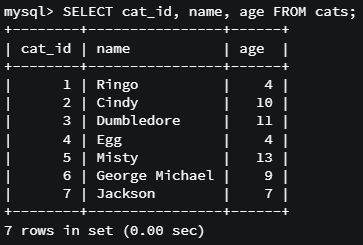
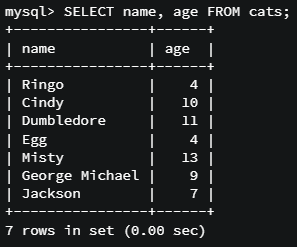
* + Using this on our recently re-created *cats* table:



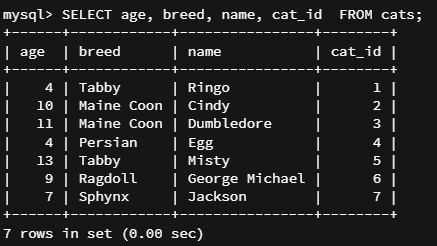
* We can also get more specific about the data that we want back, in particular the columns that we want to look at. How? We use the command **SELECT <column> from <table>**

* We can also select more than one column using **SELECT <column1,column2,….columnN> from <table>;**



* + The order of column name declaration matters here. The columns will be returned in the order that you **SELECT;** them in



* Code summary: Introduction to SELECT

#### Various Simple SELECT statements:

SELECT \* FROM cats;

SELECT name FROM cats;

SELECT age FROM cats;

SELECT cat\_id FROM cats;

SELECT name, age FROM cats;

SELECT cat\_id, name, age FROM cats;

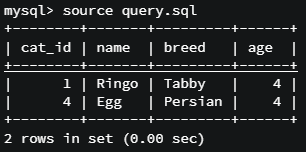
SELECT age, breed, name, cat\_id FROM cats;

SELECT cat\_id, name, age, breed FROM cats;

An Introduction to WHERE

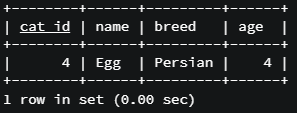
* We’ve learned how to select columns with SELECT. However, this method without modification returns every single record of the table. What if we want specific records?
* The **WHERE** keyword allows us to specify the particular records we want. Let’ see a simple example of selecting cats of age
  + Note that we use the integer 4





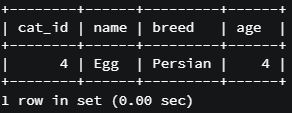
* Another example where we select a cat with a specific name:
  + Note that in this case we have to use quotes around “Egg”, since the datatype is VARCHAR





* + By default, the WHERE keyword is **case-insensitive**
    - We’ll see ways around this later on, but generally speaking this tends to be a useful property





* Code summary – Introduction to WHERE

#### CODE: Introduction to WHERE

Select by age:

SELECT \* FROM cats WHERE age=4;

Select by name:

SELECT \* FROM cats WHERE name='Egg';

Notice how it deals with case:

SELECT \* FROM cats WHERE name='egG';

SELECT Challenges Solution Code

#### CODE: Select Challenges Solution

SELECT cat\_id FROM cats;

SELECT name, breed FROM cats;

SELECT name, age FROM cats WHERE breed='Tabby';

SELECT cat\_id, age FROM cats WHERE cat\_id=age;

SELECT \* FROM cats WHERE cat\_id=age;

* Note the last line two lines of code where we are directly comparing “cat\_id” and “age”, instead of using a hard-coded value. MySQL understands that these two variables are numbers and is able to make that comparison