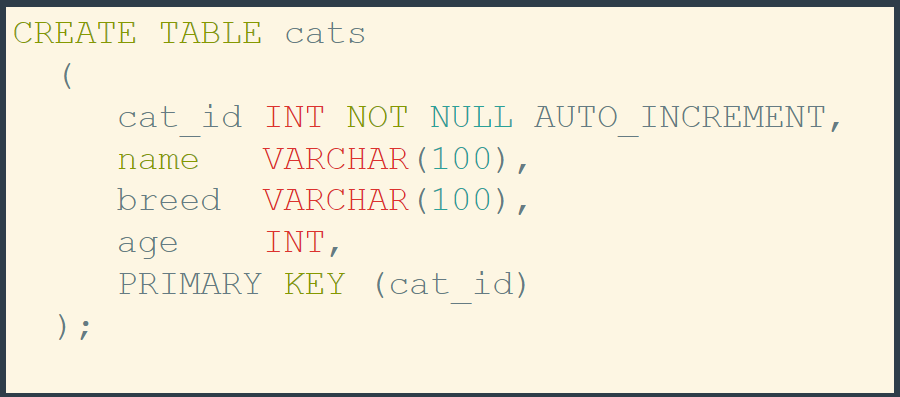
Section slides: <http://webdev.slides.com/coltsteele/mysql-99-100>

Introduction to CRUD

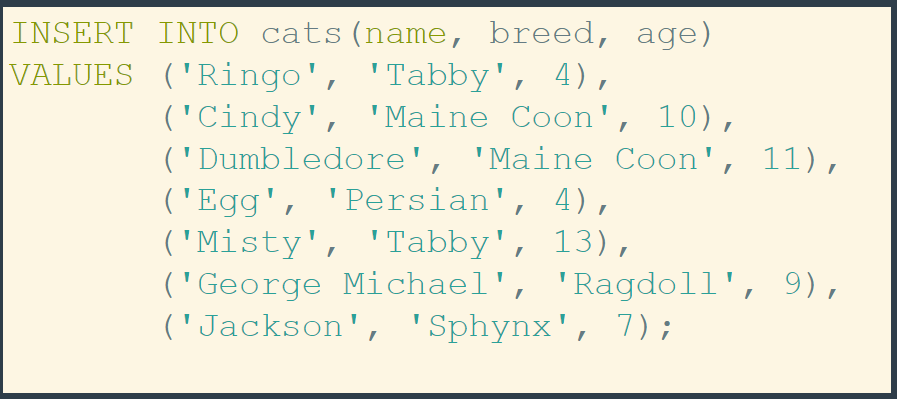
* CRUD stands for **C**reate, **R**ead, **U**pdate, **D**elete
  + These are the four main operations that we perform on database data
  + It also applies to other facets of programming, but we focus on its database applications in this course
* We’ve already seen Create in the context of creating tables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

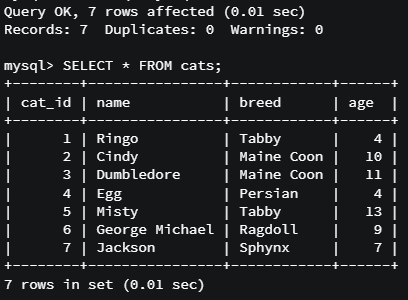
Preparing Our Data

* 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

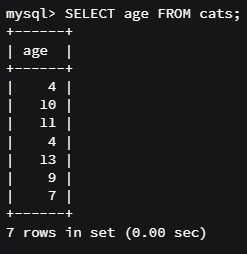
* The magical **SELECT;** command is used to *read* data from a database. We’ve seen it before, but how does it work?
* When we say **SELECT \*;** we are asking for ALL columns in a table



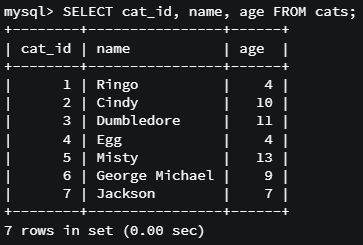
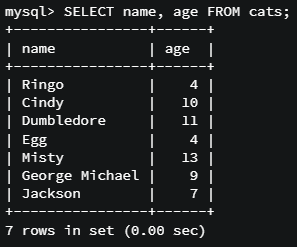
* + Using this on our recently re-created *cats* table:



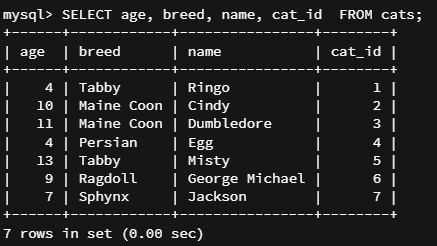
* We can also get more specific about the data that we want back, in particular the columns that we want to look at. How? We use the command **SELECT <column> from <table>**

* 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;

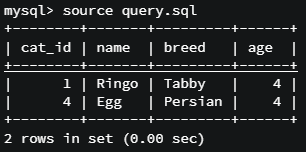
SELECT age, breed, name, cat\_id FROM cats;

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

An Introduction to WHERE

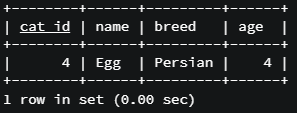
* We’ve learned how to select columns with SELECT. However, this method without modification returns every single record of the table. What if we want specific records?
* The **WHERE** keyword allows us to specify the particular records we want. Let’ see a simple example of selecting cats of age
  +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





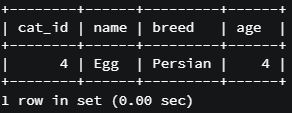
* Another example where we select a cat with a specific name:
  + Note that in this case we have to use quotes around “Egg”, since the datatype is VARCHAR





* + 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';

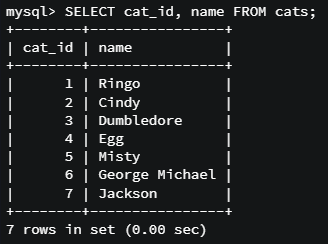
SELECT cat\_id, age FROM cats WHERE cat\_id=age;

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

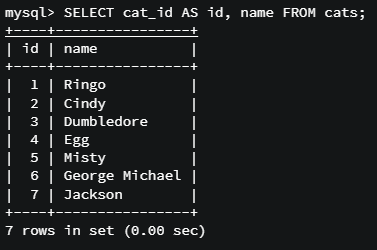
* Note the last line two lines of code where we are directly comparing “cat\_id” and “age”, instead of using a hard-coded value. MySQL understands that these two variables are numbers and is able to make that comparison

Introduction to Aliases

* **Aliases** allow us to specify the way data is *displayed* back to us. Essentially, it allows us to change how the column titles are displayed
  + The actual column names within the table and the underlying data are not changed
* These will come into play later when we, for example, select the two columns of the same name from two different tables
* An example from our *cats* table
  + Without alias selection:



* + With alias selection:



* Another example, where we can use quotes in order to allow for spaces in our aliases:



* Aliases code summary:

CODE: Introduction to Aliases

SELECT cat\_id AS id, name FROM cats;

SELECT name AS 'cat name', breed AS 'kitty breed' FROM cats;

DESC cats;

The Update Command

* In SQL, we can update and alter existing data that’s already present in the table
  + Updates can be made to correct a mistake, or simply to change a value to a new one (e.g. someone has changed his/her password)
* Updating utilize three keywords: **UPDATE**, **SET**, and **WHERE**
  + UPDATE <table\_name> SET <column\_name> = “new value” WHERE <column\_name> = “old value”
  + Translation – within the table <table\_name>, find where ever the value in the column <column\_name> is “old value” and update it to “new value”
* Example from our *cats* table, where we update all “Tabby” breed cats to be “Shorthair”





* You can also update a value in a column based on the value in a *different* column. Here, we update the age of all cats named “Misty” to 14. Notice how we are searching by cat name, not by the current age.





* A good rule of thumb is to SELECT the data BEFORE you do the update so that you know exactly what data is getting updated.
  + Run the select on the data first to ensure it returns the data that you want to update
  + Then run the update command
  + There is no “undo” button for updates, so to fix an error you’ll need to manually update it back
* Code summary: The UPDATE Command

#### CODE: Updating Data

Change tabby cats to shorthair:

UPDATE cats SET breed='Shorthair' WHERE breed='Tabby';

Another update:

UPDATE cats SET age=14 WHERE name='Misty';

Update Challenges Code

#### CODE: Update Challenges Solution

1. SELECT \* FROM cats WHERE name='Jackson';
3. UPDATE cats SET name='Jack' WHERE name='Jackson';
5. SELECT \* FROM cats WHERE name='Jackson';
7. SELECT \* FROM cats WHERE name='Jack';
9. SELECT \* FROM cats WHERE name='Ringo';
11. UPDATE cats SET breed='British Shorthair' WHERE name='Ringo';
13. SELECT \* FROM cats WHERE name='Ringo';
15. SELECT \* FROM cats;
17. SELECT \* FROM cats WHERE breed='Maine Coon';
19. UPDATE cats SET age=12 WHERE breed='Maine Coon';
21. SELECT \* FROM cats WHERE breed='Maine Coon';

Introduction to DELETE

* The **DELETE FROM;** command allows us to delete/destroy things in MySQL
  + It behaves similarly to selecting
* ALWAYS perform a selection before you delete, so that you know what you’re deleting
* Here we will delete the cat named “Egg” from the table
  + First select that cat so we know that there is something to delete



* + Now we do the deletion:



* + Notice that the cat\_id’s are NOT shifted. Remember that it is a primary key that uniquely identifies that cat. We’ll see why this is important later on.
* As a quick note, the command **DELETE FROM <table name>;** with no additional information will delete all entries in the table, leaving an empty table shell
  + This is similar to the SELECT clause with no WHERE statement, where everything is selected from the table
* Code summary

#### CODE: DELETING DATA

1. DELETE FROM cats WHERE name='Egg';
3. SELECT \* FROM cats;
5. SELECT \* FROM cats WHERE name='egg';
7. DELETE FROM cats WHERE name='egg';
9. SELECT \* FROM cats;
11. DELETE FROM cats;

Code from DELETE Challenges

#### CODE: DELETE Challenges Solution

1. SELECT \* FROM cats WHERE age=4;
3. DELETE FROM cats WHERE age=4;
5. SELECT \* FROM cats WHERE age=4;
7. SELECT \* FROM cats;
9. SELECT \* FROM cats WHERE cat\_id=age;
11. DELETE FROM cats WHERE cat\_id=age;
13. DELETE FROM cats;
15. SELECT \* FROM cats;