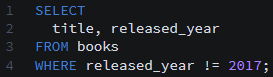
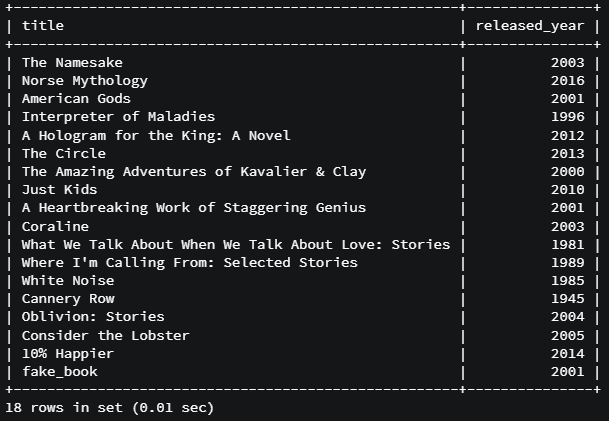
Section slides: <http://webdev.slides.com/coltsteele/mysql-99-103>

* Operators allow us to further refine our selections by narrowing down the selection criteria in specific ways
* We’ll do things like select birthdays between two defined years, or selecting all items in stock and priced below a certain price

# NOT EQUAL

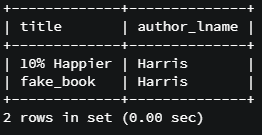
* The **not equal** operator, **“!=”**, is used to *exclude* the defined values
  + A fairly univers
* Documetation: <https://www.w3resource.com/mysql/comparision-functions-and-operators/not-equal-operator.php>
* It’s very similar to using the “=” operator, but it does the opposite
* Let’s seen an example where we *select the titles of all books that were NOT released in 2017*





* Another example where we exclude particular authors
  + To motivate the idea, we first select all books by authors with the last name “Harris”





* + Now we simply flip the switch and use the NOT EQUAL operator to select titles if books by all authors except those with the last name “Harris”





* Code summary

SELECT title FROM books WHERE released\_year = 2017;

SELECT title FROM books WHERE released\_year != 2017;

SELECT title, author\_lname FROM books;

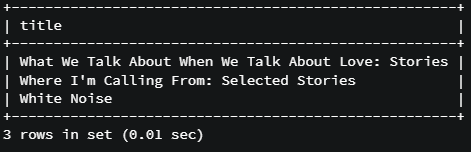
SELECT title, author\_lname FROM books WHERE author\_lname = 'Harris';

SELECT title, author\_lname FROM books WHERE author\_lname != 'Harris';

# NOT LIKE

* The NOT LIKE function is the opposite of LIKE. Recall that “LIKE” allows us to match patterns, usually in strings
* As a refresher, let’s do a LIKE selection for all books that start with “W”





* Now we’ll try a NOT LIKE selection. Here, we’ll select all titles that DO NOT start with the letter “W”
  + Notice how this does not use the exclamation point operator. We simply say “NOT LIKE”





* Code summary

SELECT title FROM books WHERE title LIKE 'W';

SELECT title FROM books WHERE title LIKE 'W%';

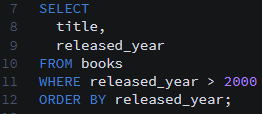
SELECT title FROM books WHERE title LIKE '%W%';

SELECT title FROM books WHERE title LIKE 'W%';

SELECT title FROM books WHERE title NOT LIKE 'W%';

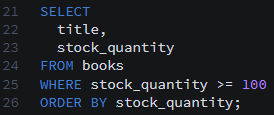
# GREATER THAN and GREATER THAN OR EQUAL TO

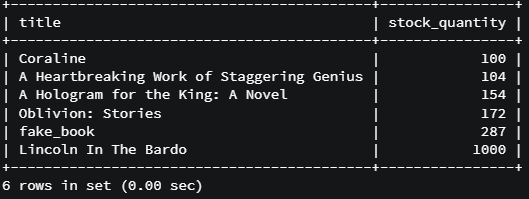
* The greater than operator is the “>” symbol. It is also fairly universal across programming languages
* With this operator, we can do things like *select all books released after the year 2000*





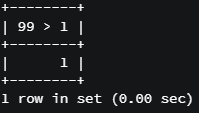
* The **greater than or equal to** operator, represented by “>=”, is inclusive of the defining value.
* Let’s do an example where we want to *select all books with stock of greater than or equal to 100*





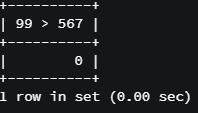
* What happens if we try the following?





* + What returns is the Boolean value of 1, representing “True”. Since 99 is in fact greater than 1, this selection will return “1”, which is equivalent to “True”.
  + To verify, let’s write a false statement. This will return “0”, equivalent to “False”





* Exercise: What will these evaluate to?
  + 100 > 5
    - 1 (True)
  + -15 > 15
    - 0 (False)
  + 9 > -10
    - 1 (True)
  + 1 > 1
    - 0 (False)
  + ‘a’ > ‘b’
    - 0 (False)
  + ‘A’ > ‘a’
    - 0 (False)
    - A and a are actually equivalent in MySQL. We’ve seen this in instances were we select text in a case-insensitive manner
* It is best to avoid logical operators on strings because it is confusing and inconsistent between different languages
* Code summary

SELECT title, released\_year FROM books ORDER BY released\_year;

SELECT title, released\_year FROM books

WHERE released\_year > 2000 ORDER BY released\_year;

SELECT title, released\_year FROM books

WHERE released\_year >= 2000 ORDER BY released\_year;

SELECT title, stock\_quantity FROM books;

SELECT title, stock\_quantity FROM books WHERE stock\_quantity >= 100;

SELECT 99 > 1;

SELECT 99 > 567;

100 > 5

-- true

-15 > 15

-- false

9 > -10

-- true

1 > 1

-- false

'a' > 'b'

-- false

'A' > 'a'

-- false

'A' >= 'a'

-- true

SELECT title, author\_lname FROM books WHERE author\_lname = 'Eggers';

SELECT title, author\_lname FROM books WHERE author\_lname = 'eggers';

SELECT title, author\_lname FROM books WHERE author\_lname = 'eGGers';

# LESS THAN and LESS THAN OR EQUAL TO

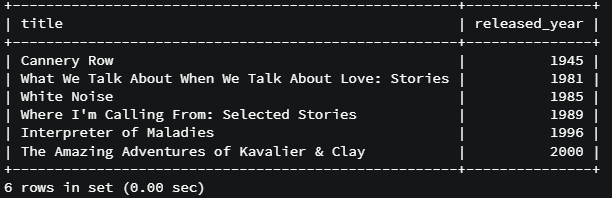
* The LESS THAN operator “<” works similarly to GREATER THAN
* We can do things like *select* *all books that were released before the year 2000*





* The LESS THAN OR EQUAL TO operator, “<=”, also functions similarly. If we wanted to include books published in 2000, we could do the following





* Exercises: What will these evaluate to?
  + 3 < -10
    - False
  + -10 < -9
    - True
  + 42 <= 42
    - True
  + ‘h’ < ‘p’
    - True
    - This is because ‘h’ comes before ‘p’ and both are lower-case
  + ‘Q’ <= ‘q’
    - True
    - Remember that case does not matter in MySQL
* Code summary

SELECT title, released\_year FROM books;

SELECT title, released\_year FROM books

WHERE released\_year < 2000;

SELECT title, released\_year FROM books

WHERE released\_year <= 2000;

SELECT 3 < -10;

-- false

SELECT -10 < -9;

-- true

SELECT 42 <= 42;

-- true

SELECT 'h' < 'p';

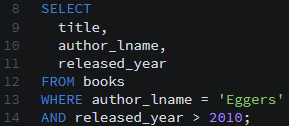
-- true

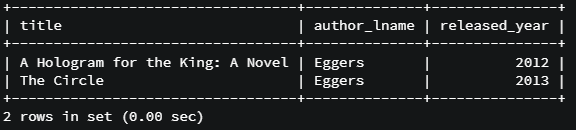
SELECT 'Q' <= 'q';

-- true

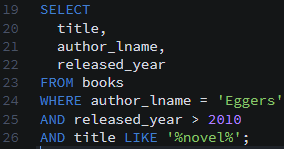
# Logical AND

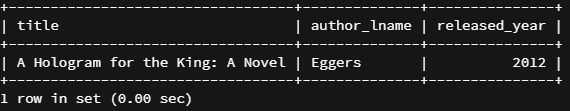
* The logical AND (“AND” or “&&”) necessitates that selections satisfy all conditions
  + The “&&” operator is deprecated and will be removed from a future version of MySQL
* Documentation: <https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html#operator_and>
* For example, suppose we want to *select books written by Dave Eggers published after the year 2010*. How would we do this?
  + This is a compound selection, so we need to use two conditions and the AND operator





* AND logical exercises
  + 1 < 5 && 7 = 9
    - False
  + -10 > -20 && 0 <= 0
    - True
  + -40 <= -0 AND 10 > 40
    - False
  + 54 <= 54 && ‘a’ = ‘A’
    - True
* The AND operator is NOT limited to just 2 components. There is no practical upper limit, though we generally will use only 2 or 3
* Example – *select all books whose author’s last name is Eggers, was released after 2010, and has the world “novel” in the title*





* Code Summary

SELECT title, author\_lname, released\_year FROM books

WHERE author\_lname='Eggers';

SELECT title, author\_lname, released\_year FROM books

WHERE released\_year > 2010;

SELECT

title,

author\_lname,

released\_year FROM books

WHERE author\_lname='Eggers'

AND released\_year > 2010;

SELECT 1 < 5 && 7 = 9;

-- false

SELECT -10 > -20 && 0 <= 0;

-- true

SELECT -40 <= 0 AND 10 > 40;

--false

SELECT 54 <= 54 && 'a' = 'A';

-- true

SELECT \*

FROM books

WHERE author\_lname='Eggers'

AND released\_year > 2010

AND title LIKE '%novel%';