Limiting Your Results

Jason P. Browning

DIRECTOR OF DATA ANALYTICS, UNIVERSITY OF TULSA

@jason_from_ky

Overview

The WHERE clause

Relational operators

Pattern matching using LIKE

Null values

END CLIP ONE

Don’t forget
two seconds
of silence.

```
SELECT first_name, last_name
  FROM person;
```

Querying Specific Fields

Code returns first name and last name for every record in person table

SELECT keyword specifies fields of interest

FROM specifies table where records are stored

Expanding The SELECT Framework

SELECT	FROM	WHERE
Specify columns of interest	**Where these columns are stored**	**Criteria to filter matching rows**

The WHERE Clause

```
SELECT first_name, last_name

 FROM person

 WHERE first_name = “Shelby”
```

Filtering Results

All records

```
SELECT first_name, last_name
  FROM person;
```

first_name	last_name
Katie	Brown
Brenna	Davis
Shelby	Garza
Brenda	Jones
Shelby	Smith
Tom	Stephens
Elmo	Nathanson

Limiting criteria

```
SELECT first_name, last_name
  FROM person
 WHERE first_name = “Shelby”;
```

first_name	last_name
Shelby	Garza
Shelby	Smith

Criteria in the WHERE clause is case-sensitive

END CLIP TWO

Don’t forget
two seconds
of silence.

Comparison Operators

=

Equal

<>

Not equal

>

Greater than

<

Less than

>=

Greater than or equal

<=

Less than or equal

```
SELECT city,
       state,
       population
  FROM city_population
 WHERE city = “Louisville”;
```

city	state	Population
Louisville	CO	21,128
Louisville	KY	616,261

Population table

city	state	Population
Boise	ID	226,570
Cincinnati	OH	301,301
Cleveland	OH	385,525
Louisville	CO	21,128
Louisville	KY	616,261
Marquette	MI	20,629
Seattle	WA	724.745

```
SELECT city,
       state,
       population
  FROM city_population
 WHERE city = “Louisville”
  AND state = “KY”;
```

city	state	Population
Louisville	KY	616,261

Population table

city	state	Population
Boise	ID	226,570
Cincinnati	OH	301,301
Cleveland	OH	385,525
Louisville	CO	21,128
Louisville	KY	616,261
Marquette	MI	20,629
Seattle	WA	724.745

- Use AND to combine multiple criteria

Demo

Using comparison operators

Filtering for specific conditions

END CLIP THREE

Don’t forget
two seconds
of silence.

Matching Patterns

Relational operators match fields to a specific value

LIKE matches fields to a specific pattern

Functionality of LIKE

first_name	last_name
Katie	Brown
Brenna	Davis
Shelby	Garza
Brenda	Jones
Shelby	Smith
Tom	Stephens
Elmo	Nathanson

```
SELECT first_name, last_name
  FROM person
 WHERE first_name = “Shelby”;
```

```
SELECT first_name, last_name
  FROM person
 WHERE first_name LIKE “Shelby”;
```

first_name	last_name
Shelby	Garza
Shelby	Smith

Pattern Wildcards

Pattern to match can include wildcards

% represents zero or more characters

_ represents exactly one character

Demo

Implementing the LIKE keyword

Criteria for pattern matching

Fields That Do Not Match Pattern

WHERE

Criteria to filter matching rows

NOT

Find those fields that do not match the pattern

LIKE

Specify the pattern to compare against field

Usefulness

Dealing with messy data sets

Fuzzy-matching

Regular expressions

END CLIP FOUR

Don’t forget
two seconds
of silence.

Null Values

Null is not equivalent to zero

Null indicates that the value of a field is missing or unknown

A field is null only when no data exists in that field

Representing Missing Data

first_name	last_name	age
Katie	Brown	18
Brenna	Davis	
Shelby	Garza	26
Brenda	Jones	25
Shelby	Smith	41
Tom	Stephens	35
Elmo	Nathanson	38

Substituting 0 could skew calculations

Using a blank space character is poor practice

SQL recognizes an empty field as a null value

Keywords for Null Values

IS NULL

This is the same as saying

Field = “”

IS NOT NULL

This is the same as saying

Field <> “”

Keywords for Null Values

You cannot use standard relational operators to find null values

Must use IS NULL or IS NOT NULL syntax

Demo

Finding null values

Finding values that are not null

Checking for Nulls

Check for null values to diagnose issues in a dataset

Arithmetic operations that include a null value will always return a null value

END CLIP FIVE

Don’t forget two seconds of silence.

Logical Operators

AND

If a row from the table matches **both** conditions, it will be included

OR

If a row from the table matches **either** condition, it will be included

```
SELECT first_name,
       age
  FROM person
 WHERE age >= 19
   AND age <= 35;
```

```
SELECT first_name,
       age
  FROM person
 WHERE age BETWEEN 19 and 35;
```

The BETWEEN Keyword

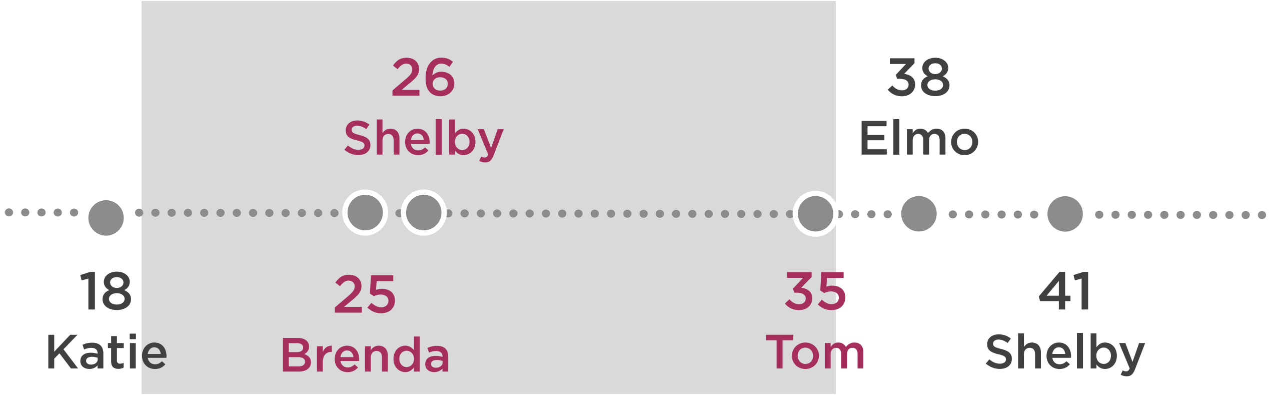
BETWEEN looks for matches within specified boundaries

Boundaries are inclusive

The BETWEEN Keyword

first_name	last_name	age
Katie	Brown	18
Shelby	Garza	26
Brenda	Jones	25
Shelby	Smith	41
Tom	Stephens	35
Elmo	Nathanson	38

```
WHERE AGE BETWEEN 19 AND 35
```



The IN Keyword

Instead of multiple OR statements

Can use the SQL keyword IN

Provide a list of options for a given field

```
SELECT first_name,
       age
  FROM person
 WHERE first_name = “Jimmy”
 WHERE first_name IN
    OR first_name = “Brenna”
    (”Jimmy”, ”Brenna”, ”Elmo”)
    OR first_name = “Elmo”
```

- Each matching value appears in a list following the variable name
- Use IN to replaces multiple OR statements
- IN checks for equality conditions

The IN Keyword

NOT IN

Used to return records that do not match any of the listed values

END CLIP SIX

Don’t forget
two seconds
of silence.

Operator precedence

The sequence in which operations are performed

```
SELECT first_name,
       last_name,
       hometown
  FROM person
 WHERE first_name = “Shelby”
    OR first_name = “Tom”
   AND hometown = “Boston”;
```

first_name	last_name	hometown
Tom	Stephens	Boston
Shelby	Garza	Boston
Shelby	Smith	Denver

- AND has higher operator precedence than OR

first_name	last_name	hometown
Katie	Brown	Detroit
Brenna	Davis	Sacramento
Shelby	Garza	Boston
Brenda	Jones	Baltimore
Shelby	Smith	Denver
Tom	Stephens	Boston
Elmo	Nathanson	Orlando

```
SELECT first_name,
       last_name,
       hometown
  FROM person
 WHERE (first_name = “Shelby”
   OR first_name = “Tom”)
  AND hometown = “Boston”;
```

first_name	last_name	hometown
Tom	Stephens	Boston
Shelby	Garza	Boston

- SQL will evaluate contents of parentheses first

first_name	last_name	hometown
Katie	Brown	Detroit
Brenna	Davis	Sacramento
Shelby	Garza	Boston
Brenda	Jones	Baltimore
Shelby	Smith	Denver
Tom	Stephens	Boston
Elmo	Nathanson	Orlando

Use parentheses when writing complex expressions

END CLIP SEVEN

Don't forget
two seconds
of silence.

Summary

WHERE prescribes criteria

Comparison operators

LIKE matches specific patterns

AND and OR for multiple criteria