* WHERE and Conditionals
  + We can filter our data using a set of criteria (just like Advanced Filtering in Excel)
    - For example, now that we know more about the “flights” table we could ask for:
      * Flights on Virgin Atlantic, leaving from either Gatwick or Heathrow airport, that cost less than or equal to $600
        + Each of these criteria can be thought of as a **condition**, which means that it evaluates as either true or false
      * Lets’ rewrite our request as a series of conditions:
        + The airline is Virgin Atlancit
        + The flight leaves from either Gatwick or Heathrow airport
        + The flight costs less than or equal to $600
        + If all of these conditions are true, then we want our query to return that flight information
  + One keyword that takes conditions is WHERE
    - WHERE returns records from our dataset where the condition is true
* SELECT origin, destination airline, price
* FROM flights
* WHERE airline = 'Virgin Atlantic';
  + - The equals (=) sign here is known as a **comparison operator**.
      * A comparison operator tests or defines the relationship between two values
      * Comparison Operator:
        + = equal to
        + <> not equal to
        + < less than
        + > greater than
        + <= less than or equal to
        + >= greater than or equal to
  + If we weren’t 100% sure of the name of the airline we want, we could use the LIKE operator or the negative version NOT LIKE
    - LIKE is used in place of the equals sign.
    - For example, the query below would return all airlines that start with the letter V:

SELECT origin, destination airline, price

FROM flights

WHERE airline = 'V%';

* + - The % sign is a wildcard and fills in for the letters we don’t know.
  + Wildcards
    - Wildcards come in handy for situaitons like this, where we want to search for a string that follows a specific pattern
      * For example, to include the misspelled values in our search for Virgin Atlantic flights, we could search for words that start with a “V,” followed by two unknown letters, and ending in “gin:”

SELECT origin, destination airline, price

FROM flights

WHERE airline = 'V\_gin%';

* + - * The underscore (\_) wildcard character fills in for one character, whether it’s a letter, number, or symbol
      * Here, we used two of them to fill in for the two letters between “V” and “g”
* String Functions and CASE
  + There are several different categories of functions used with SQL. We already covered many different types:
    - Numeric functions
    - Aggregate functions
    - Distinct functions
    - Joint operators
    - Predicate operators (such as WHERE or HAVING)
    - Boolean operators
  + In this lesson we will cover two more important categories:
    - String functions
      * String functions make it possible for you to find and alter text values, such as VARCHAR and CHAR datatypes
      * Using these functions you can alter a text value such as changing “Smith, Joe” to “Joe Smith”
    - Time and date functions
      * Time and date functions perform operations on date and time inputs values and return string, numeric, or date and time values
      * The most difficult part of working with dates is to be sure the format of the date you are trying to insert matches the format of the date column in the database
  + String Functions
    - The IF function is a popular function in Excel — it’s commonly used for creating a column that categorizes another column.
    - You can do something similar in SQL using the CASE statement. It uses the same IF…THEN…ELSE, and ELSEIF logic.
    - The syntax in SQL is slightly different, but the outcome is the same.
    - Using CASE statements allows you to to map decision trees into SQL, and then you can use them for other functions or grouping.
    - CASE syntax looks like:

SELECT dimensions

CASE

WHEN condition THEN result

WHEN condition THEN result

ELSE

END

AS output name

FROM table;

* + - Concatenate
      * The CONCAT function combines two fields or expressions together. Here’s the formal syntax for most SQL programs:
        + CONCAT(field1, field2, field3…)
      * It’s comparable to the CONCATENATE or CONCAT functions in Excel, or using the “&” in Excel
      * In SQLite which we will be learning about in the next unit, || is used instead of CONCAT(). So in SQLite the syntax is field1 || field2 || field3…
      * When concatenating fields, we need to add in spaces, dashes, or other delimiters that we want to appear on the screen.

SELECT CONCAT(item\_description, ‘- ‘, bottle\_size)

AS our\_offerings

FROM products

* + In SQLite:

SELECT item\_description || '- ' || bottle\_size

AS our\_offerings

FROM products

* + - Letter Case
      * Since SQL is case sensitive, sometimes you want to transform all values in a column to appear upper case or lower case so you can remove duplicates or group them. There are two functions for this:
        + LOWER: converts a filed or expression to lower case:

Syntax: LOWER(field1)

* + - * + UPPER: Converts a field or expression to upper case

Syntax: UPPER(field1)

* + - * LEFT / RIGHT Selection
        + A different way of combining the characters in a field is using the left and right selection fuctions
        + LEFT: Selects a given number of characters from the left side

Syntax: LEFT(field1, length)

* + - * + RIGHT: Selects a given number of characters from the right side

Syntax: RIGHT (field1, length)

* + - SUBSTRING
      * This function allows you to isolate a section of characters within a field to retrieve
        + Syntax: SUBSTRING (field1, starting position, number of characters to retrieve from starting positions)
    - LENGTH
      * The LENGTH function counts the length of characters in a field
        + Syntax: LENGTH (field1)
    - Left / Right Trim
      * These functions trim blanks from the given side
        + LTRIM: Trims all blanks from the left side.

Syntax: LTRIM(field1)

* + - * + RTRIM: Trims all blanks from the right side.

Syntax: RTRIM(field1)

* + - TRIM
      * This function removes specified characters from start of field (leading characters), end of field (trailing characters), or both.
        + Syntax:

TRIM(leading ‘characters’, from field1)

TRIM(trailing ‘characters’, from field1)

TRIM(both ‘characters’, from field1)

* + - REPLACE
      * This function is similar to the Excel function substitute; it allows you to replace a value in a field with another value.
        + Syntax: REPLACE(field\_to\_change, content\_to\_replace, new\_content)
  + Time and Date Functions
    - * It can be difficult to get into learning about time and date functions.
      * Syntax can be vastly different depending on the SQL platform.
      * For example, IBM’s current query tool uses a TIMESTAMPDIFF function, whereas pgAdmin uses a simple AGE function to the same effect.
      * The best approach is to have an overall understanding of what DATES can do, and have your particular vendor's DATE syntax documentation (usually found online) close by.
      * Having said that, we will take a look at CURRENT\_DATE and AGE. These two date functions tend to be the most popular.
    - CURRRENT\_DATE brings back the current date from the system.
      * Syntax: CURRENT\_DATE
    - AGE or DATEDIFF (in MySQL) returns the difference between two dates in days.
      * Syntax: AGE( date1, date2)
      * DATEDIFF( date1, date2) in MySQL
    - DATEDIFF in SQL Server calculates the difference between two dates in units specified in the query.
      * Syntax: DATEDIFF (datepart, expression1, expression2)
      * Example:
        + SELECT DATEDIFF (month, '2017-03-31 23:59:59.000','2017-04-01 00:00:00.000');

The result is: 1

Even though the actual difference between the two date/time values is 1 second, the DATEDIFF function returns 1 when we ask for the difference in months.

This is because SQL Server only compares the year and month fields. All other fields (day, hour, minute, second, millisecond) are ignored.

DATEDIFF returns an integer. It only compares the values down to the lowest level of the given datepart and does not look at anything that is lower.