https://classroom.udacity.com/courses/ud002-bert/lessons/03f64082-fa4d-4aff-80be-d48597867e07/concepts/d529924c-52ba-4a54-821d-549ba998272a

SQL Data cleaning LN 32

Here we looked at three new functions:

1. **LEFT**
2. **RIGHT**
3. **LENGTH**

LEFT pulls a specified number of characters for each row in a specified column starting at the beginning (or from the left). As you saw here, you can pull the first three digits of a phone number using LEFT(phone_number, 3).

This means the phone_number will be splitted and written as three numbers

Ex : 333-222-555 since it is given as LEFT(phone_number,3)

RIGHT pulls a specified number of characters for each row in a specified column starting at the end (or from the right). As you saw here, you can pull the last eight digits of a phone number using RIGHT(phone_number, 8).

LENGTH provides the number of characters for each row of a specified column. Here, you saw that we could use this to get the length of each phone number as LENGTH(phone_number).

In this lesson, you learned about:

1. **POSITION**
2. **STRPOS**
3. **LOWER**
4. **UPPER**

POSITION takes a character and a column, and provides the index where that character is for each row. The index of the first position is 1 in SQL. If you come from another programming language, many begin indexing at 0. Here, you saw that you can pull the index of a comma as POSITION(',' IN city_state).

STRPOS provides the same result as POSITION, but the syntax for achieving those results is a bit different as shown here: STRPOS(city_state, ',').

Note, both POSITION and STRPOS are case sensitive, so looking for A is different than looking for a.

Therefore, if you want to pull an index regardless of the case of a letter, you might want to use LOWER or UPPER to make all of the characters lower or uppercase.
In this lesson you learned about:

1. **CONCAT**
2. **Piping `||`**

Each of these will allow you to combine columns together across rows. In this video, you saw how first and last names stored in separate columns could be combined together to create a full name: CONCAT(first_name, ' ', last_name) or with piping as first_name || ' ' || last_name.
1.Each company in the `accounts` table wants to create an email address for each `primary_poc`. The email address should be the first name of the primary_poc `.` last name primary_poc `@` company name `.com`.

```
WITH t1 AS (
 SELECT LEFT(primary_poc,     STRPOS(primary_poc, ' ') -1 ) first_name,
RIGHT(primary_poc, LENGTH(primary_poc) - STRPOS(primary_poc, ' ')) last_name,
name
 FROM accounts)
SELECT first_name, last_name, CONCAT(first_name, '.', last_name, '@', name,
'.com')
FROM t1;
```

In this video, you saw additional functionality for working with dates including:

1. TO_DATE
2. CAST
3. Casting with `::`

DATE_PART('month', TO_DATE(month, 'month')) here changed a month name into the number associated with that particular month.

Then you can change a string to a date using CAST. CAST is actually useful to change lots of column types. Commonly you might be doing as you saw here, where you change a `string` to a `date` using CAST(date_column AS DATE). However, you might want to make other changes to your columns in terms of their data types. You can see other examples here.

In this example, you also saw that instead of CAST(date_column AS DATE), you can use date_column::DATE.

Expert Tip

Most of the functions presented in this lesson are specific to strings. They won’t work with dates, integers or floating-point numbers. However, using any of these functions will automatically change the data to the appropriate type.

LEFT, RIGHT, and TRIM are all used to select only certain elements of strings, but using them to select elements of a number or date will treat them as strings for the purpose of the function. Though we didn't cover TRIM in this lesson explicitly, it can be used to remove characters from the beginning and end of a string. This can remove unwanted spaces at the beginning or end of a row that often happen with data being moved from Excel or other storage systems.

There are a number of variations of these functions, as well as several other string functions not covered here. Different databases use subtle variations on these functions, so be sure to look up the appropriate database syntax if you're connected to a private database.The Postgres literature contains a lot of the related functions.

http://www.postgresqltutorial.com/postgresql-cast/

In the following examples, we convert various strings to dates data type:

```
SELECT
CAST ('2015-01-01' AS DATE);

SELECT
CAST ('01-OCT-2015' AS DATE);
```

First, we converted 2015-01-01 literal string into January 1st 2015. Second, we converted 01-OCT-2015 to October 1st 2015.

COALESCE:

In general, COALESCE returns the first non-NULL value passed for each row.

https://www.youtube.com/watch?v=AzkHAO4JzNQ

There are a few other functions that work similarly. You can read more about those here. You can also get a walk through of many of the functions you have seen throughout this lesson here.

Learn more about Sql from here : https://community.modeanalytics.com/sql/