CSD 310: Database Development and Use
Module 5.2 Assignment: MySQL Functions
Isaac Ellingson
11/16/2025

Let's focus on functions that can be run on "pure input" since those can be folded into the middle of queries to create more readable or useful output.

LENGTH – yields the length of the string passed in. So, LENGTH('fog') will return 3.

Examples:

```
SELECT LENGTH('Some input') AS 'String Length';
```

or,

```
SET @Foo = 'This is a magic string, so I can show how more complex queries can happen';
SELECT @Foo AS 'Foo', LENGTH(@Foo) AS 'Length of Foo';
```

This should produce:

Foo	Length of Foo
This is a magic string, so I can show how more com...	73

(output is from phpMyAdmin which is easy to compose and query from)

I think this is most useful in WHERE clauses where we want to find either the most concise or the most verbose output.

Let's do two functions at once that often get used together: **PI** and **DEGREES**.

PI is the mathematical circle constant, π. MySQL doesn't really have a first-class way to represent constants so they're all exposed as functions.

DEGREES takes a number of radians as input, and converts that angle to degrees. It happens that PI is half a circle, so,

```
SELECT DEGREES(PI()) as 'Half a Circle';
```

Half a Circle
180

will produce for us exactly 180 degrees.

And sure enough, that's what we get. This can be useful if we're dealing with anything geometric in our data.

COALESCE is a “null-coalescing function”. In other areas, I’m used to the “null-coalescing operator” so this sparked my interest. Null-coalescing is about taking some combination of null, possibly-null, and not-null input and having it “come together” as non-null output. In this particular case, it gives us the FIRST non-null output in our input list.

So if we were to:

```
SELECT null, null, null, 'thing', 'other thing';
```

We get our input back as output:

NULL	NULL	NULL	thing	other thing
NULL	*NULL*	*NULL*	thing	other thing

But if we use COALESCE:

```
SELECT COALESCE(null, null, null, 'thing', 'other thing') AS
'It\'s Not Null';
```

We can see that the FIRST non-null element of the list is returned.

It's Not Null
thing

If we have a subquery that contains a lot of nulls in it, and we just want to find the first relevant piece of information, this can be useful. Or if there’s a JOIN happening of unlike types, where you will always have data in EITHER column A OR column B, but one of them will always be null, it can be much easier to read the data coalesced into one column.