# Advanced data-cleaning functions, part 2

Hey there. Great to see you again.

So far, we've seen some SQL functions in action.

In this video, we'll go over more uses for CAST,

and then learn about CONCAT and

COALESCE. Let's get started.

Earlier we talked about the CAST function,

which let us typecast text strings into floats.

I called out that the CAST function can be

used to change into other data types too.

Let's check out another example of how

you can use CAST in your own data work.

We've got the transaction data we were working

with from our Lauren's Furniture Store example.

But now, we'll check out the purchase date field.

The furniture store owner has asked us to look at

purchases that occurred during

their sales promotion period in December.

Let's write a SQL query that will pull date and

purchase\_price for all purchases

that occurred between December 1st,

2020, and December 31st, 2020.

We start by writing the basic SQL structure:

SELECT, FROM, and WHERE.

We know the data comes from

the customer\_purchase table in the customer\_data dataset,

so we write customer\_data.customer\_purchase after FROM.

Next, we tell SQL what data to pull.

Since we want date and purchase\_price,

we add them into the SELECT statement.

Play video starting at :1:38 and follow transcript1:38

Finally, we want SQL to filter for

purchases that occurred in December only.

We type date BETWEEN '2020-12-01' AND

'2020-12-31' in the WHERE clause.

Let's run the query.

Four purchases occurred in December,

but the date field looks odd.

That's because the database recognizes

this date field as datetime,

which consists of the date and time.

Our SQL query still works correctly, even if

the date field is datetime instead of date.

But we can tell SQL to convert

the date field into the date

data type so we see just the day and not the time.

To do that, we use the CAST() function again.

We'll use the CAST() function to replace the date field in

our SELECT statement with

the new date field that will

show the date and not the time.

We can do that by typing CAST() and adding

the date as the field we want to change.

Then we tell SQL the data type we want instead,

which is the date data type.

Play video starting at :2:57 and follow transcript2:57

There. Now we can have

cleaner results for purchases that

occurred during the December sales period.

CAST is a super useful function

for cleaning and sorting data,

which is why I wanted you to see

it in action one more time.

Next up, let's check out the CONCAT function.

CONCAT lets you add strings together to create

new text strings that can be used as unique keys.

Going back to our customer\_purchase table,

we see that the furniture store sells

different colors of the same product.

The owner wants to know if customers prefer

certain colors, so the owner can

manage store inventory accordingly.

The problem is, the product\_code

is the same, regardless of the product color.

We need to find another way to separate products by color,

so we can tell if customers prefer

one color over the others.

We'll use CONCAT to produce a unique key that'll

help us tell the products apart by

color and count them more easily.

Let's write our SQL query by starting with

the basic structure: SELECT, FROM, and WHERE.

We know our data comes from

the customer\_purchase table

and the customer\_data dataset.

We type "customer\_data.customer\_purchase" after FROM

Next, we tell SQL what data to pull.

We use the CONCAT() function here to get

that unique key of product and color.

So we type CONCAT(),

the first column we want,

product\_code, and the other column

we want, product\_color.

Play video starting at :4:49 and follow transcript4:49

Finally, let's say we want to look at couches,

so we filter for couches by typing

product = 'couch' in the WHERE clause.

Now we can count how many times

each couch was purchased and

figure out if customers preferred

one color over the others.

Play video starting at :5:14 and follow transcript5:14

With CONCAT, the furniture store can find out

which color couches are the most popular and order more.

I've got one last advanced function

to show you, COALESCE.

COALESCE can be used to return non-null values in a list.

Null values are missing values.

If you have a field that's optional in your table,

it'll have null in that field for rows

that don't have appropriate values to put there.

Let's open the customer\_purchase table

so I can show you what I mean.

In the customer\_purchase table,

we can see a couple rows where

product information is missing.

That is why we see nulls there.

But for the rows where product name is null,

we see that there is product\_code data

that we can use instead.

We'd prefer SQL to show us the product name,

like bed or couch,

because it's easier for us to read.

But if the product name doesn't exist,

we can tell SQL to give us the product\_code instead.

That is where the COALESCE function comes into play.

Let's say we wanted a list

of all products that were sold.

We want to use the product\_name column

to understand what kind of product was sold.

We write our SQL query with

the basic SQL structure: Select, From, AND Where.

We know our data comes from

customer\_purchase table and the customer\_data dataset.

We type "customer\_data.customer\_purchase" after FROM.

Next, we tell SQL the data we want.

We want a list of product names,

but if names aren't available,

then give us the product code.

Here is where we type "COALESCE."

then we tell SQL which column to check first, product,

and which column to check second

if the first column is null, product\_code.

We'll name this new field as product\_info.

Finally, we are not filtering out any data,

so we can take out the WHERE clause.

This gives us product information for each purchase.

Now we have a list of all products that

were sold for the owner to review.

COALESCE can save you

time when you're making calculations

too by skipping any null values

and keeping your math correct.

Those were just some of the

advanced functions you can use to clean

your data and get it ready for

the next step in the analysis process.

You'll discover more as you continue working in SQL.

But that's the end of this video and this module.

Great work. We've covered a lot of ground.

You learned the different data-

cleaning functions in spreadsheets and

SQL and the benefits of

using SQL to deal with large datasets.

We also added some SQL formulas

and functions to your toolkit,

and most importantly, we got to experience some of

the ways that SQL can help you get

data ready for your analysis.

After this, you'll get to spend

some time learning how to verify and report

your cleaning results so that your data is

squeaky clean and your stakeholders know it.

But before that, you've got

another weekly challenge to tackle. You've got this.

Some of these concepts might seem challenging at first,

but they'll become second nature to

you as you progress in your career.

It just takes time and practice.

Speaking of practice, feel free to go back to any of

these videos and rewatch or

even try some of these commands on your own.

Good luck. I'll see you again when you're ready.