The CAST command

This lesson demonstrates how to use the CAST command to convert values of any data type into another data type.

The one requirement when using numeric commands is that they can only be applied to numeric data types. Similarly, string commands can only be applied to string data types.

This can become an issue when a column looks like a string but has been stored in the table as a numeric type, such as in a year field. You wouldn't be able to perform string commands on this field because it's a numeric field. Similarly, if there is a column that contains numbers, such as prices or quantities, but that data is actually stored as a string, then you wouldn't be able to perform numeric commands on that field.

Though this seems like a big obstacle, SQL has a command that lets you convert the data type of a value to another specified data type. It's called the CAST command. This is the structure of a query using CAST:

```
SELECT CAST(expression AS datatype(length));
```

As with most queries, you can simply add the appropriate FROM clause if you want to convert the data type of an entire column, and use the column name as the expression. That looks like this:

```
SELECT CAST(column_name AS datatype(length))
FROM table_name;
```

Here are some notes about this syntax:

- The **expression** is the value that you're converting.
- The **datatype** is the data type to convert the **expression** to. It can be one of the following:
    - **bigint**, **int**, **smallint**, **tinyint**, **bit**, **decimal**, **numeric**, **money**, **smallmoney**, **float**, **real**, **datetime**, **smalldatetime**, **date**, **time**, **timestamp**, **char**, **varchar**, **text**, **nchar**, **nvarchar**, **ntext**, **binary**, **varbinary**, or **image**.
- The **length** is an optional argument which defines the length of the resulting data type.
    - This optional argument is used for **char**, **varchar**, **nchar**, **nvarchar**, **binary**, and **varbinary**.

he price column contains prices, but it's stored as a text column in this table. Say you want to know the average price of all prices in the table. Try executing the following query to find the average price.

```
SELECT AVG(price)
FROM cheapest;
```

Query Editor | Query History

```
1  SELECT
2  AVG(price)
3  FROM cheapest;
```

Data Output | Explain | Messages | Notifications

```
ERROR:  function avg(text) does not exist
LINE 2: AVG(price)
        ^
HINT:  No function matches the given name and argument types. You
type casts.
SQL state: 42883
Character: 9
```

As you can see, although price contains prices in decimals, numeric functions don't work with this field because it's technically a text field.

To work around this, revise your query using the CAST command to convert the data type for the price column to decimal. Your query should look like this:

```
1  SELECT AVG(CAST(price AS decimal))
2  FROM cheapest;
```

Execute the above query to get this result:

Data Output | Explain | Messages

	avg numeric
1	533.6542459903921917

You successfully converted the price column to a decimal data type and found the average!

Similarly, if you want to see the cheap_date column in the timestamp format, rather than the current date format, you can apply CAST using this query:

```
SELECT CAST(cheap_date AS timestamp)
FROM cheapest;
```

Execute this query to get the following output:

Data Output | Explain | Messages

	cheap_date timestamp without time zone
1	2018-11-01 00:00:00
2	2018-11-02 00:00:00
3	2018-11-03 00:00:00
4	2018-11-04 00:00:00
5	2018-11-05 00:00:00
6	2018-11-06 00:00:00
7	2018-11-07 00:00:00
8	2018-11-08 00:00:00
9	2018-11-09 00:00:00
10	2018-11-10 00:00:00
11	2018-11-11 00:00:00
12	2018-11-12 00:00:00

Lastly, say you want to extract the year from the cheap_date column. Try executing the following query to do this.

```
SELECT LEFT(cheap_date,4)
FROM cheapest;
```

This query produces an error because the cheap_date column is a date type and the LEFT command is a string function. Now, add CAST to your query to convert this field into a varchar type so that you can apply the LEFT command.

```
SELECT LEFT(CAST(cheap_date AS varchar), 4)
FROM cheapest;
```