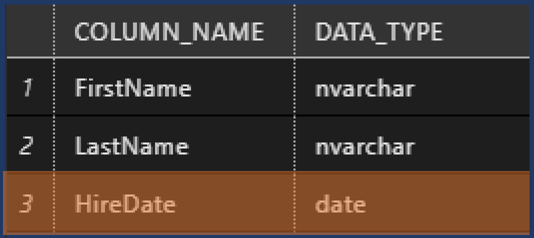
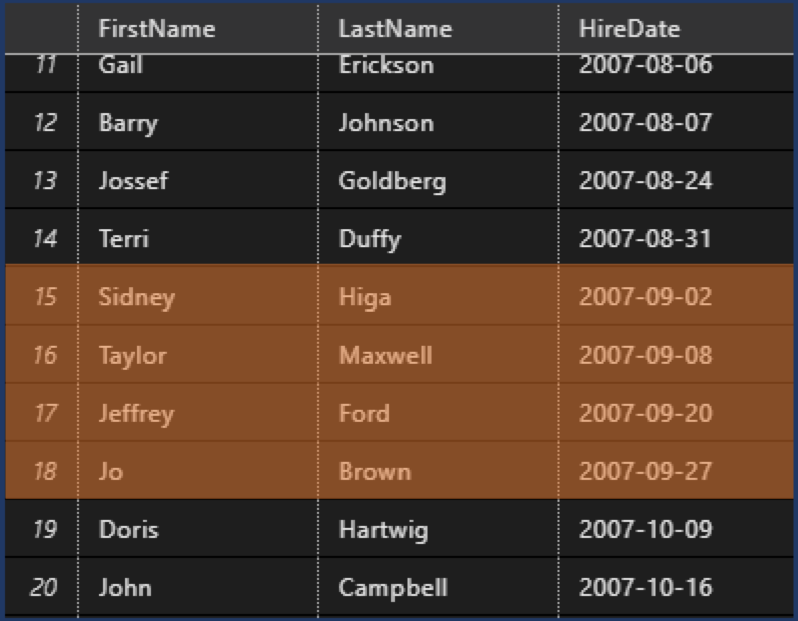
SQL:
Handling date literals in SQL Server

SQL Server can interpret date strings as dates

```sql
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'DimEmployee'
AND COLUMN_NAME IN (
                    'FirstName',
                    'LastName',
                    'HireDate'
                    );
```

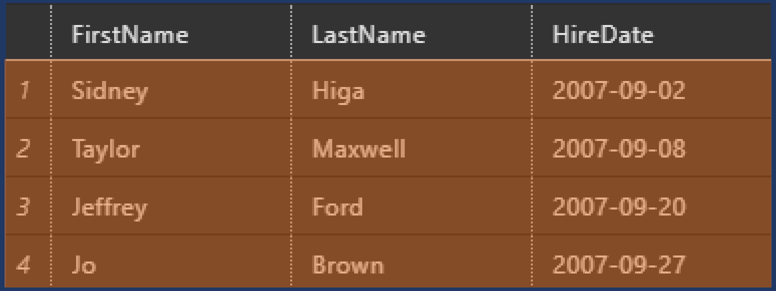


	COLUMN_NAME	DATA_TYPE
1	FirstName	nvarchar
2	LastName	nvarchar
3	HireDate	date



	FirstName	LastName	HireDate
11	Gail	Erickson	2007-08-06
12	Barry	Johnson	2007-08-07
13	Jossef	Goldberg	2007-08-24
14	Terri	Duffy	2007-08-31
15	Sidney	Higa	2007-09-02
16	Taylor	Maxwell	2007-09-08
17	Jeffrey	Ford	2007-09-20
18	Jo	Brown	2007-09-27
19	Doris	Hartwig	2007-10-09
20	John	Campbell	2007-10-16

```sql
SELECT FirstName, LastName, HireDate
FROM DimEmployee
WHERE HireDate
    BETWEEN '2007-09-01' AND '2007-09-30';
```



	FirstName	LastName	HireDate
1	Sidney	Higa	2007-09-02
2	Taylor	Maxwell	2007-09-08
3	Jeffrey	Ford	2007-09-20
4	Jo	Brown	2007-09-27

While many other formats are supported, when writing SQL *queries*, best practice is to always use the format:

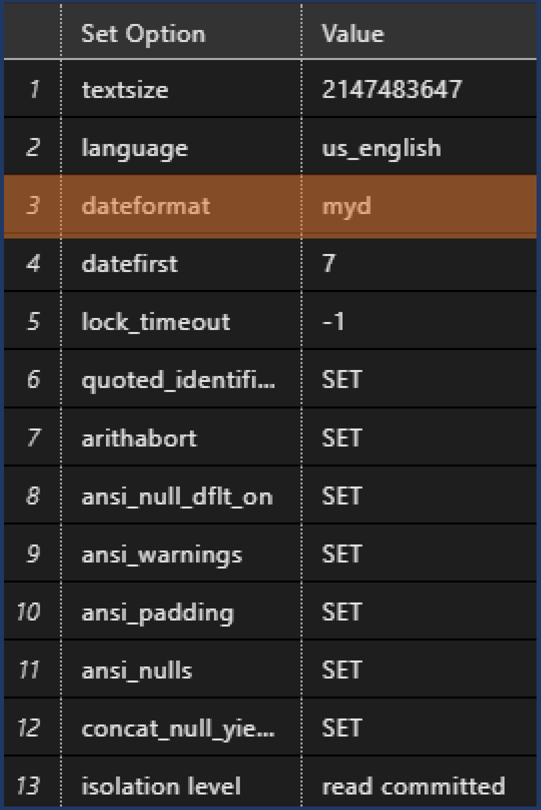
YYYY-MM-DD or **YYYYMMDD**

This is the international standard ISO 8601 and it will ensure your code is *portable* to other database systems and *transferable* to other locales

But what if you receive raw strings in another format?

Run this statement to view the settings for the current session.

```
DBCC useroptions;
```



	Set Option	Value
1	textsize	2147483647
2	language	us_english
3	dateformat	myd
4	datefirst	7
5	lock_timeout	-1
6	quoted_identifi...	SET
7	arithabort	SET
8	ansi_null_dflt_on	SET
9	ansi_warnings	SET
10	ansi_padding	SET
11	ansi_nulls	SET
12	concat_null_yie...	SET
13	isolation level	read committed

As well as recognizing YYYY-MM-DD, SQL Server interprets date strings according to the DATEFORMAT setting, which determines in which order to expect the day, month and year in date string literals

We can change the current session's DATEFORMAT

```
SET DATEFORMAT dym;
```

Valid parameters for DATEFORMAT:
mdy, dmy, ymd, ydm, myd, dym

Implicit conversion uses current DATEFORMAT

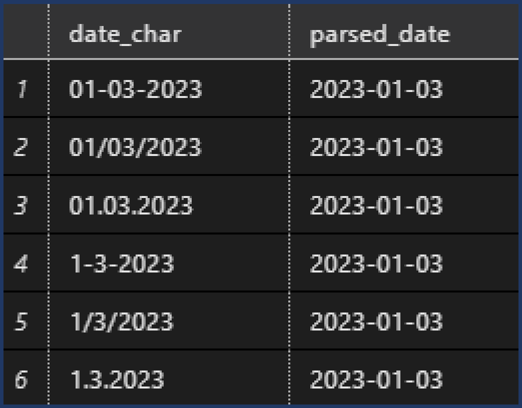
```sql
DROP TABLE IF EXISTS #datetest;
CREATE TABLE #datetest (date_char varchar(20),
                        parsed_date date);

INSERT INTO #datetest (date_char) VALUES
--mdy
('01-03-2023'),
('01/03/2023'),
('01.03.2023'),
('1-3-2023'),
('1/3/2023'),
('1.3.2023');

UPDATE #datetest
SET parsed_date = date_char
WHERE parsed_date IS NULL;

SELECT date_char, parsed_date
FROM #datetest;
```

Since the current DATEFORMAT is mdy, this UPDATE of parsed_date uses *implicit type conversion* to update the text in the date_char column to a date data type



	date_char	parsed_date
1	01-03-2023	2023-01-03
2	01/03/2023	2023-01-03
3	01.03.2023	2023-01-03
4	1-3-2023	2023-01-03
5	1/3/2023	2023-01-03
6	1.3.2023	2023-01-03

Dash, forward-slash and period are all valid date-part separators in SQL Server

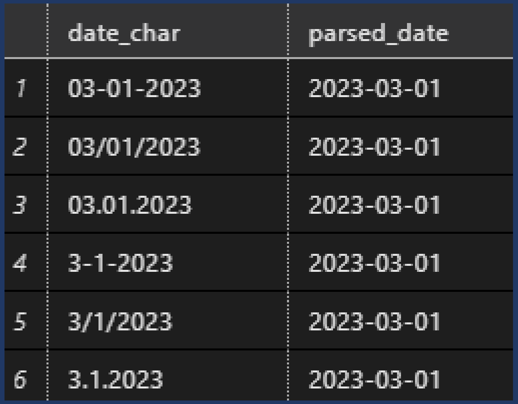
Change the DATEFORMAT when needed

```sql
INSERT INTO #datetest (date_char) VALUES
--dmy
('03-01-2023'),
('03/01/2023'),
('03.01.2023'),
('3-1-2023'),
('3/1/2023'),
('3.1.2023');

UPDATE #datetest
SET parsed_date = date_char;

SELECT date_char, parsed_date
FROM #datetest;
```

If you have strings in the dmy format and you don't change the DATEFORMAT, they'll be interpreted incorrectly

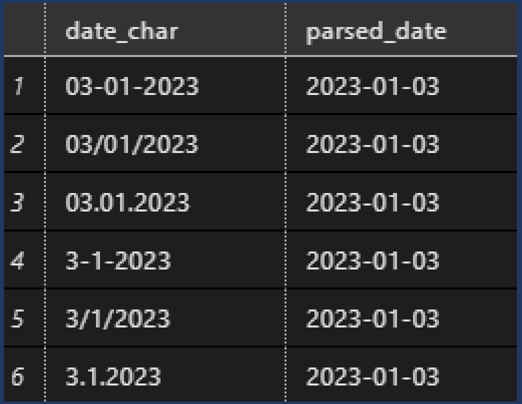


	date_char	parsed_date
1	03-01-2023	2023-03-01
2	03/01/2023	2023-03-01
3	03.01.2023	2023-03-01
4	3-1-2023	2023-03-01
5	3/1/2023	2023-03-01
6	3.1.2023	2023-03-01

This setting will persist for the remainder of the current session

```sql
SET DATEFORMAT dmy;
UPDATE #datetest
SET parsed_date = date_char;

SELECT date_char, parsed_date
FROM #datetest;
```



	date_char	parsed_date
1	03-01-2023	2023-01-03
2	03/01/2023	2023-01-03
3	03.01.2023	2023-01-03
4	3-1-2023	2023-01-03
5	3/1/2023	2023-01-03
6	3.1.2023	2023-01-03

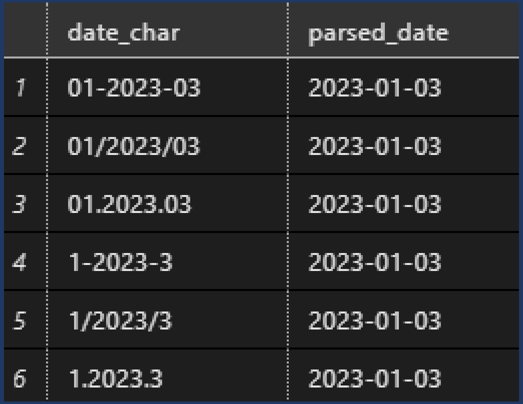
This works similarly for other formats

```sql
INSERT INTO #datetest (date_char) VALUES
--myd
('01-2023-03'),
('01/2023/03'),
('01.2023.03'),
('1-2023-3'),
('1/2023/3'),
('1.2023.3');

SET DATEFORMAT myd;
UPDATE #datetest
SET parsed_date = date_char;

SELECT date_char, parsed_date
FROM #datetest;
```

Because DATEFORMAT is set according to the format of the strings, the dates are parsed correctly



	date_char	parsed_date
1	01-2023-03	2023-01-03
2	01/2023/03	2023-01-03
3	01.2023.03	2023-01-03
4	1-2023-3	2023-01-03
5	1/2023/3	2023-01-03
6	1.2023.3	2023-01-03

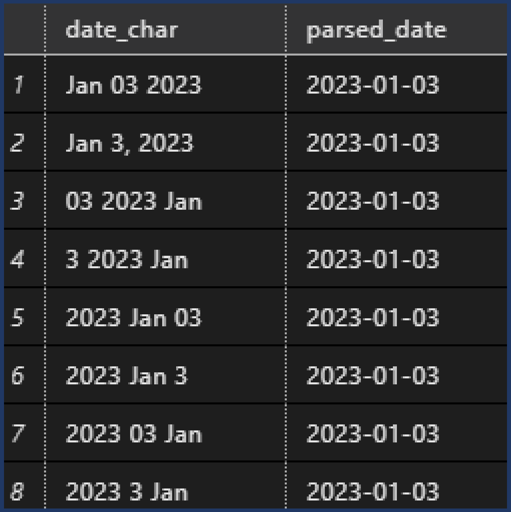
If the month is spelled out, the DATEFORMAT is not relevant

```sql
INSERT INTO #datetest (date_char) VALUES
('Jan 03 2023'),
('Jan 3, 2023'),
('03 2023 Jan'),
('3 2023 Jan'),
('2023 Jan 03'),
('2023 Jan 3'),
('2023 03 Jan'),
('2023 3 Jan');

SET DATEFORMAT myd; --default for my locale
UPDATE #datetest
SET parsed_date = date_char;

SELECT date_char, parsed_date
FROM #datetest;
```

The month can be abbreviated or fully spelled out.

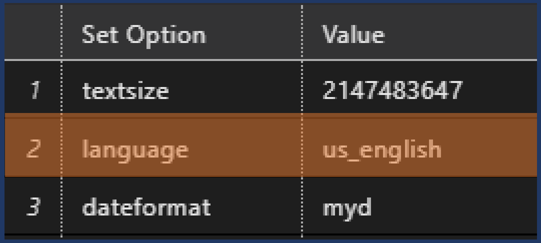


	date_char	parsed_date
1	Jan 03 2023	2023-01-03
2	Jan 3, 2023	2023-01-03
3	03 2023 Jan	2023-01-03
4	3 2023 Jan	2023-01-03
5	2023 Jan 03	2023-01-03
6	2023 Jan 3	2023-01-03
7	2023 03 Jan	2023-01-03
8	2023 3 Jan	2023-01-03

Dates in languages other than your own can be handled with SET LANGUAGE

```
DBCC useroptions;
```

The language used to interpret the month is according to the LANGUAGE setting

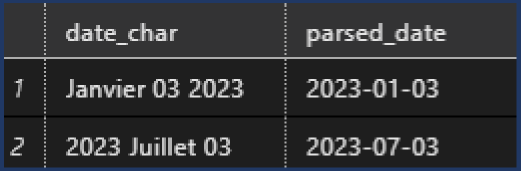


	Set Option	Value
1	textsize	2147483647
2	language	us_english
3	dateformat	myd

```
INSERT INTO #datetest (date_char) VALUES
('Janvier 03 2023'),
('2023 Juillet 03');

SET LANGUAGE French;
UPDATE #datetest
SET parsed_date = date_char;

SELECT date_char, parsed_date
FROM #datetest;
SET LANGUAGE us_english;
```

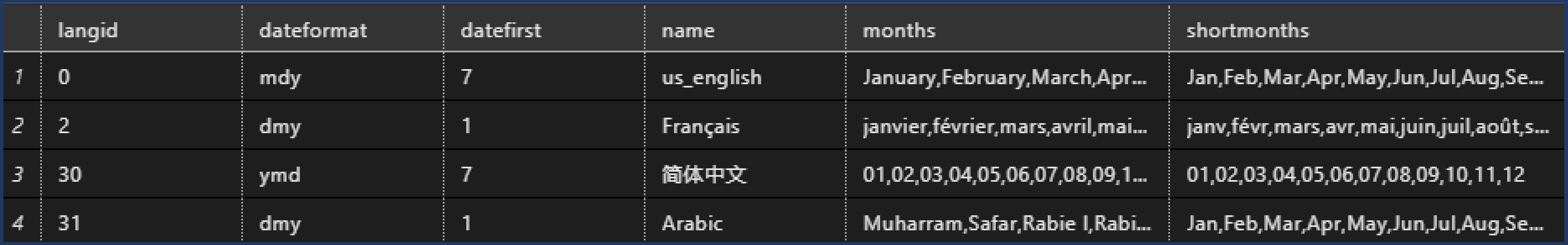


	date_char	parsed_date
1	Janvier 03 2023	2023-01-03
2	2023 Juillet 03	2023-07-03

Be sure to reset the LANGUAGE after parsing the dates

Valid languages and month spellings are listed in sys.syslanguages

```sql
SELECT langid, dateformat, datefirst, name, months, shortmonths
FROM sys.syslanguages
WHERE langid IN (0,2,30,31); --for example
```



	langid	dateformat	datefirst	name	months	shortmonths
1	0	mdy	7	us_english	January,February,March,Apr...	Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Se...
2	2	dmy	1	Français	janvier,février,mars,avril,mai...	janv,févr,mars,avr,mai,juin,juil,août,s...
3	30	ymd	7	简体中文	01,02,03,04,05,06,07,08,09,1...	01,02,03,04,05,06,07,08,09,10,11,12
4	31	dmy	1	Arabic	Muharram,Safar,Rabie I,Rabi...	Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Se...

SET LANGUAGE implicitly sets DATEFORMAT

Some date formats are not supported by implicit conversion

```sql
INSERT INTO #datetest (date_char) VALUES
('01 03 2023'),
('1 3, 2023');

SET DATEFORMAT myd; --default for my locale
```

The date-parts in these examples are separated by spaces

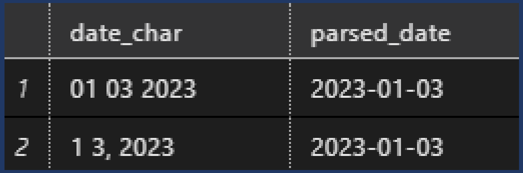
```sql
/*
ERROR:
"Conversion failed when converting
date and/or time from character string."
*/
UPDATE #datetest
SET parsed_date = date_char;
```

This UPDATE fails because the space is not a valid date-part separator

The TRY_PARSE function can be used for non-standard date formats.

```sql
UPDATE #datetest
SET parsed_date = TRY_PARSE(date_char AS date);

SELECT date_char, parsed_date
FROM #datetest;
```



	date_char	parsed_date
1	01 03 2023	2023-01-03
2	1 3, 2023	2023-01-03

TRY_PARSE returns NULL if the date cannot be parsed.

Takeaways

1. Use YYYY-MM-DD or YYYYMMDD when *querying* date columns with string literals

2. To handle date formats from different locales, use SET DATEFORMAT x; , where x is one of: mdy, dmy, ymd, ydm, myd, dym

3. To handle months spelled in different languages, use SET LANGUAGE x; , where x is a language listed in sys.syslanguages.name