Building dates

TIME SERIES ANALYSIS IN SQL SERVER

Kevin Feasel
CTO, Envizage

datacamp

What you will learn

- Working with component date parts
- Translating strings to dates, including date-time offsets and invalid dates
- Filtering, grouping, and aggregating data by time periods
- Upsampling and downsampling data
- Aggregations over windows
- Calculating running totals and moving averages
- Finding overlap in date ranges

(Photo by **Aron Visuals**)

Building a date

```
SELECT
    GETDATE() AS DateTime_LTz,
    GETUTCDATE() AS DateTime_UTC;
```

```
SELECT
    SYSDATETIME() AS DateTime2_LTz
    SYSUTCDATETIME() AS DateTime2_UTC;
```

Results:

DateTime_LTz	DateTime_UTC	DateTime2_LTz	DateTime2_UTC
2019-03-07 21:21:33.670	2019-03-08 02:21:33.670	2019-03-07 21:21:33.6716402	2019-03-08 02:21:33.6716402

Breaking down a date

```
DECLARE
    @SomeDate DATETIME2(3) = '2019-03-01 08:17:19.332';
```

```
SELECT YEAR(@SomeDate);
```

```
SELECT MONTH(@SomeDate);
```

```
SELECT DAY(@SomeDate);
```

`YEAR` = 2019

`MONTH` = 3

`DAY` = 1

Parsing dates with date parts

Functions

`DATEPART()`

```
SELECT
  DATEPART(YEAR, @dt) AS TheYear;
```

`DATENAME()`

```
SELECT
  DATENAME(MONTH, @dt) AS TheMonth;
```

Parts

- Year / Month / Day
- Day of year
- Day of week
- Week of year
- ISO week of year
- Minute / Second
- Millisecond / Nanosecond

Adding and subtracting dates

```
DECLARE
    @SomeTime DATETIME2(7) = '1992-07-14 14:49:36.2294852';
SELECT
    DATEADD(DAY, 1, @SomeTime) AS NextDay,
    DATEADD(DAY, -1, @SomeTime) AS PriorDay;
SELECT
    DATEADD(HOUR, -3, DATEADD(DAY, -4, @SomeTime)) AS Minus4Days3Hours;
```

NextDay	PriorDay
1992-07-15 14:49:36.2294852	1992-07-13 14:49:36.2294852

Minus4Days3Hours
1992-07-10 11:49:36.2294852

Comparing dates

```
DECLARE
    @StartTime DATETIME2(7) = '2012-03-01 14:29:36',
    @EndTime DATETIME2(7) = '2012-03-01 18:00:00';
```

```
SELECT
    DATEDIFF(SECOND, @StartTime, @EndTime) AS SecondsElapsed,
    DATEDIFF(MINUTE, @StartTime, @EndTime) AS MinutesElapsed,
    DATEDIFF(HOUR, @StartTime, @EndTime) AS HoursElapsed;
```

SecondsElapsed	MinutesElapsed	HoursElapsed
12624	211	4

Let's practice!

TIME SERIES ANALYSIS IN SQL SERVER

Formatting dates for reporting

TIME SERIES ANALYSIS IN SQL SERVER

Kevin Feasel
CTO, Envizage

datacamp

Formatting functions

`CAST()`

`CONVERT()`

`FORMAT()`

The CAST() function

- Supported going back at least to SQL Server 2000
- Useful for converting one data type to another data type, including date types
- No control over formatting from dates to strings
- ANSI SQL standard, meaning any relational and most non-relational databases have this function

Using the CAST() function

```
DECLARE
    @SomeDate DATETIME2(3) = '1991-06-04 08:00:09',
    @SomeString NVARCHAR(30) = '1991-06-04 08:00:09',
    @OldDateTime DATETIME = '1991-06-04 08:00:09';
```

```
SELECT
    CAST(@SomeDate AS NVARCHAR(30)) AS DateToString,
    CAST(@SomeString AS DATETIME2(3)) AS StringToDate,
    CAST(@OldDateTime AS NVARCHAR(30)) AS OldDateToString;
```

DateToString	StringToDate	OldDateToString
1991-06-04 08:00:09.000	1991-06-04 08:00:09.000	Jun 4 1991 8:00AM

The CONVERT() function

- Supported going back at least to SQL Server 2000
- Useful for converting one data type to another data type, including date types
- Some control over formatting from dates to strings using the style parameter
- Specific to T-SQL

Using the CONVERT() function

```
DECLARE
    @SomeDate DATETIME2(3) = '1793-02-21 11:13:19.033';
```

```
SELECT
    CONVERT(NVARCHAR(30), @SomeDate, 0) AS DefaultForm,
    CONVERT(NVARCHAR(30), @SomeDate, 1) AS US_mdy,
    CONVERT(NVARCHAR(30), @SomeDate, 101) AS US_mdyyyy,
    CONVERT(NVARCHAR(30), @SomeDate, 120) AS ODBC_sec;
GO
```

DefaultForm	US_mdy	US_mdyyyy	ODBC_sec
Feb 21 1793 11:13 AM	02/21/93	02/21/1793	1793-02-21 11:13:19

Sample CONVERT() styles

Style Code	Format
1 / 101	United States m/d/y
3 / 103	British/French d/m/y
4 / 104	German d.m.y
11 / 111	Japanese y/m/d
12 / 112	ISO standard yyyymmdd
20 / 120	ODBC standard (121 for ms)
126	ISO8601 yyyy-mm-dd hh:mi:ss.mmm
127	yyyy-mm-ddThh:mi:ss.mmmZ

The FORMAT() function

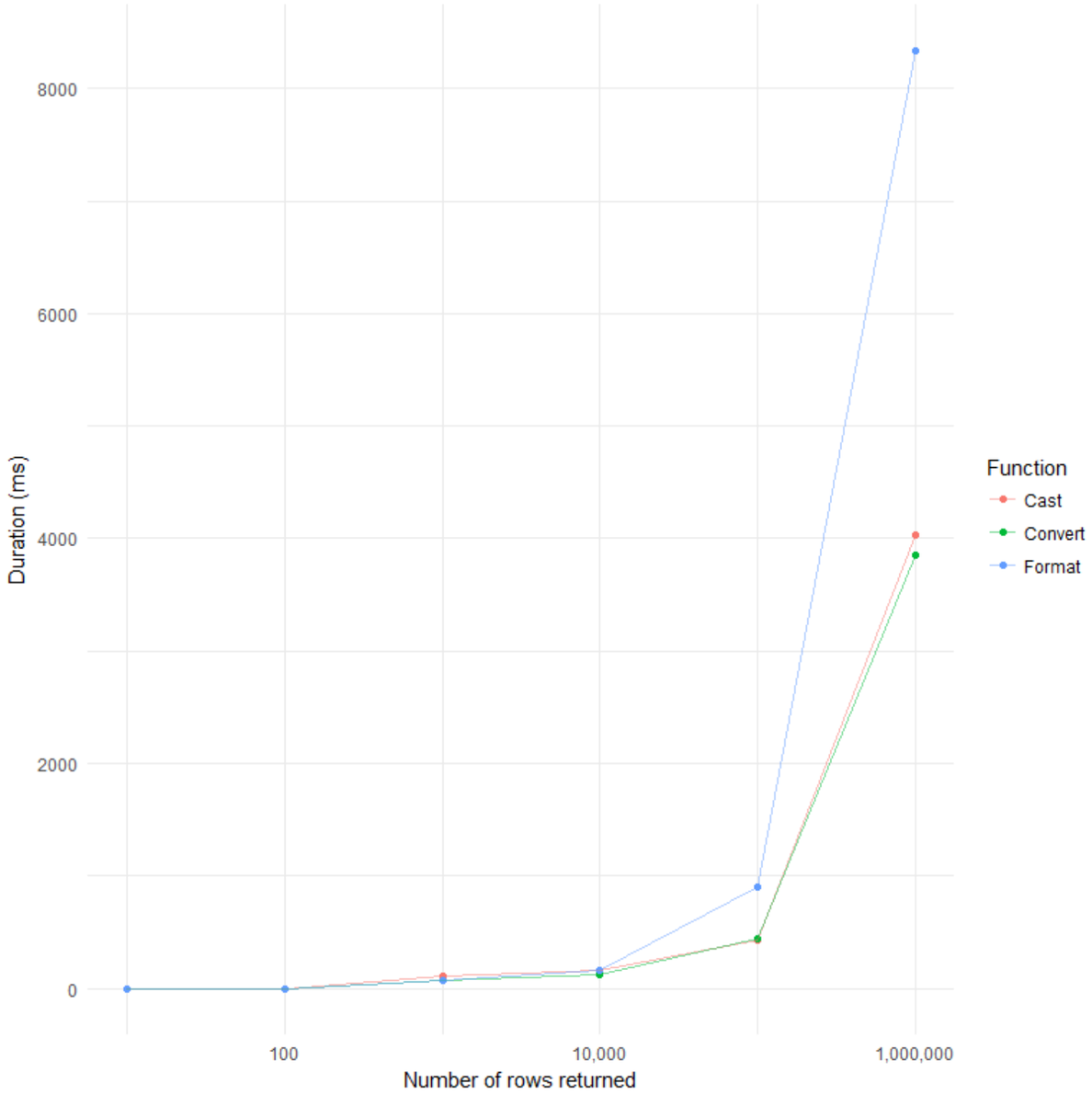
- Supported as of SQL Server 2012
- Useful for formatting a date or number in a particular way for reporting
- Much more flexibility over formatting from dates to strings than either `CAST()` or `CONVERT()`
- Specific to T-SQL
- Uses the .NET framework for conversion
- Can be slower as you process more rows

Using the FORMAT() function

```
DECLARE
    @SomeDate DATETIME2(3) = '1793-02-21 11:13:19.033';
```

```
SELECT
    FORMAT(@SomeDate, 'd', 'en-US') AS US_d,
    FORMAT(@SomeDate, 'd', 'de-DE') AS DE_d,
    FORMAT(@SomeDate, 'D', 'de-DE') AS DE_D,
    FORMAT(@SomeDate, 'yyyy-MM-dd') AS yMd;
```

US_d	DE_d	DE_D	yMd
2/21/1793	21.02.1793	Donnerstag, 21. February 1793	1793-02-21


8000
6000
4000
2000
0
Duration (ms)
100
10,000
1,000,000
Number of rows returned
Function
Cast
Convert
Format

Let's practice!

TIME SERIES ANALYSIS IN SQL SERVER

Working with calendar tables

TIME SERIES ANALYSIS IN SQL SERVER

Kevin Feasel
CTO, Envizage

datacamp

What is a calendar table?

```
SELECT *
FROM dbo.Calendar;
```

DateKey	Date	Day	DayOfWeek	DayName	...
20000101	2000-01-01	1	7	Saturday	...
20000102	2000-01-02	2	1	Sunday	...
20000103	2000-01-03	3	2	Monday	...

Contents of a calendar table

General Columns

- Date
- Day Name
- Is Weekend

Calendar Year

- Calendar month
- Calendar quarter
- Calendar year

Fiscal Year

- Fiscal week of year
- Fiscal quarter
- Fiscal first day of year

Specialized Columns

- Holiday name
- Lunar details
- ISO week of year

Building a calendar table

```
CREATE TABLE dbo.Calendar
(
    DateKey INT NOT NULL,
    [Date] DATE NOT NULL,
    [Day] TINYINT NOT NULL,
    DayOfWeek TINYINT NOT NULL,
    DayName VARCHAR(10) NOT NULL,
    ...
)
```

```
SELECT
    CAST(D.DateKey AS INT) AS DateKey,
    D.[DATE] AS [Date],
    CAST(D.[day] AS TINYINT) AS [day],
    CAST(d.[dayofweek] AS TINYINT) AS [DayOfWeek],
    CAST(DATENAME(WEEKDAY, d.[Date]) AS VARCHAR(10)) AS [DayName],
    ...
```

Using a calendar table

```
SELECT
    c.Date
FROM dbo.Calendar c
WHERE
    c.MonthName = 'April'
    AND c.DayName = 'Saturday'
    AND c.CalendarYear = 2020
ORDER BY
    c.Date;
```

Date
2020-04-04
2020-04-11
2020-04-18
2020-04-25

Using a calendar table

```
SELECT
    c.Date
FROM dbo.Calendar c
WHERE
    c.MonthName = 'April'
    AND c.DayName = 'Saturday'
    AND c.CalendarYear = 2020
ORDER BY
    c.Date;
```

Date
2020-04-04
2020-04-11
2020-04-18
2020-04-25

A quick note on APPLY()

```
SELECT
    FYStart =
        DATEADD(MONTH, -6,
            DATEADD(YEAR,
                DATEDIFF(YEAR, 0,
                    DATEADD(MONTH, 6, d.[date])), 0)),
    FiscalDayOfYear =
        DATEDIFF(DAY,
            DATEADD(MONTH, -6,
                DATEADD(YEAR,
                    DATEDIFF(YEAR, 0,
                        DATEADD(MONTH, 6, d.[date])), 0)), d.[Date]) + 1,
    FiscalWeekOfYear =
        DATEDIFF(WEEK,
            DATEADD(MONTH, -6,
                DATEADD(YEAR,
                    DATEDIFF(YEAR, 0,
                        DATEADD(MONTH, 6, d.[date])), 0)), d.[Date]) + 1
FROM dbo.Calendar d;
```

A quick note on APPLY()

```
SELECT
    fy.FYStart,
    FiscalDayOfYear = DATEDIFF(DAY, fy.FYStart, d.[Date]) + 1,
    FiscalWeekOfYear = DATEDIFF(WEEK, fy.FYStart, d.[Date]) + 1
FROM dbo.Calendar d
    CROSS APPLY
    (
        SELECT FYStart =
            DATEADD(MONTH, -6,
                DATEADD(YEAR,
                    DATEDIFF(YEAR, 0,
                        DATEADD(MONTH, 6, d.[date]))), 0))
    ) fy;
```

Let's practice!

TIME SERIES ANALYSIS IN SQL SERVER