Data Formatting and Raw SQL Cheat Sheet

String Formatting

If you have a DateTime format value you can use the .toString() Method to apply formatting and specify the output format in your desired structure.

Variable.ToString("{format declaration}")

In the format declaration you can put any of the following to extract the desired values and define how they display.

DATE FORMATTING

d => Day of month number with no leading zero (1 -31)

dd -> Day of the month as a number with leading zero (01 through 31)

ddd-> Represents the abbreviated name of the day (Mon, Tues, Wed, etc).

dddd-> Represents the full name of the day (Monday, Tuesday, etc)

MM-> Month number with leading zero(eg.04)

MMM-> Abbreviated Month Name (e.g. Dec)

MMMM-> Full month name (e.g. December)

y-> Year, no leading zero (e.g. 2015 would be 15)

yy-> Year, leading zero (e.g. 2015 would be 015)

yyy-> Year, (e.g. 2015)

yyyy-> Year, (e.g. 2015)

K-> Represents the time zone information of a date and time value (e.g. +05:00)

TIME FORMATTING

h-> 12-hour clock hour (e.g. 4).

hh-> 12-hour clock, with a leading zero (e.g. 06)

H-> 24-hour clock hour (e.g. 15)

HH-> 24-hour clock hour, with a leading zero (e.g. 22)

m-> Minutes – no leading zero

mm-> Minutes with a leading zero

s-> Seconds

ss-> Seconds with leading zero

t-> Abbreviated AM / PM (e.g. A or P)

tt-> AM / PM (e.g. AM or PM)

f - fffffff-> Represents between 1-7 most significant digit of the seconds' fraction; that is, it represents the fractions of a second in a date and time value.

NUMERIC FORMATTING

For numeric formatting you can use the ToString() method then pass it a format value in the form of a string to specify a formatting rule for when it is converted.

Number.ToString("{identifier}")

"C" or "c" : Convert to currency format with 2 decimal places (**Interger or Floating point**)

"D" or "d" : Show as full number with minus sign if necessary. If a number is placed after the indicator (e.g. "D6") it will add leading zeros to make it the number of digits shown.(**Integer only**)

"F" or "f" : Show as number with minus sign if necessary and 2 decimal places. If a number is placed after the indicator (e.g. "F3") then it will ad extra places or leading zeros to meet the given amount of places. (**Interger or Floating point**)

"**P" or "p"** : Shows number as percentage format and adds percentage sign at end of string. Default shows up to 2 decimal places but can be overrider by adding a number after the identifier (e.g. "P3"). (**Interger or Floating point**)

If you also wish to perform these converions in a specific language format you can add a Culture Reference as part of the string method as the 2nd parameter.

Number.ToString("{identifier}",{culture info});

This can be done by either adding it directly into the parameter.

Number.ToString("C", new CultureInfo("en-US"));

Or you can create it as a variable and add it to the string afterwards.

CultureInfo culture = new CultureInfo("en-US");
Number.ToString("C", culture);

There are many various cultures you can use by indicating the proper culture key.

en-US : America
en-GB : Great Britain
en-CA : Canada
en-Au : Australia
en-ZW : Zimbabwe
ja-JP : Japan
zh-CN : China
es-MX : Mexico
pt-PT : Portugal

See More at: http://www.codedigest.com/CodeDigest/207-Get-All-Language-Country-Code-List-for-all-Culture-in-C---ASP-Net.aspx

STRING FORMAT METHOD

You can manually do formatting using the String.Format() method. This system takes a string with numeric indicators withing the string which are contained in curly braces. After this string the desired values are given, separated by commas. These indicators start at 0 like array indexing and increment upwards by 1 to create a counter of the index order the values are to be inserted to the string. Each provided value after the format string is considered an index starting at 0 for the first value.

String.Format("The first number is {0} and the second is {1}", first, second);

In the above example the value of the variable first will be inserted at the 0 position and the value of second will be inserted at position 1.

You can also add additional conversions to each bracket position by following the adding a colon followed by the conversion code(see above) to the end of the format bracket.

{0:C} Insert first variable at position zero and convert to Currency.
{1:P} Insert second variable(index 1) variable at position 1 and convert to Percentage.
{1:yyyy} Insert second variable(index 1) variable at position 1 in Year only format

You can even specify a substring length for each insert section which will make the section a set size with the inserted value within it. This is done by following the insertion number with a comma then follow it with a number of how long you want the substring to be. The inserted value will be right hand oriented with any empty space applied to its left.

{0, 10} Insert value in a substring of length of 10.
{2, 6} Insert value in a substring of length of 6.

With this formatting, if the inserted value is longer then the specified size, the substring will match the data value size. If you wish the inserted value to be left hand oriented, insert a negative number as the length.

{0, -10 } Insert value in a substring of length of 10, oriented to the left.

You can even mix and match these formatting parameters by putting the index, then the size, followed by the conversion type.

{0, 10:C} Insert value in a substring of length 10 with the value converted to currency.
{2, 6:P} Insert value in a substring of length 6 with the value converted to percentage.

STRING INTERPOLATION METHOD

The same formatting rules for the String Format brackets can be applied to string interpolation. But instead of inserting a index as the first parameter, you add the variable name instead.

```
Console.WriteLine($"The date is {dateTime: yyyy-MMM-dd} and the time is {dateTime: hh:mm tt}");
```

```
Console.WriteLine($”{value, 10:P}”)
```

USING RAW SQL IN EF CORE

You can use direct SQL queries in your EF Core context class for database requests by simply adding the FromSqlRaw() command as part of the LINQ statement in your request.

EXAMPLE: `context.Table.FromSqlRaw(“{string query here}”).ToLIst();`

This allows you to customise the query or to trigger Stored Procedures to get the desired data.

With this method you need to be careful when entering data handed by the user to avoid SQL injections. This can be done by creating a parameter and passing it as part of the query request.

var **param** = new SqlParameter("LastName", "{value}");

var author = await _context.Authors.FromSqlRaw("SELECT * FROM Authors WHERE LastName = @LastName", **param**).FirstOrDefaultAsync();

If you wish to add more parameters, just create more and ad each one separated by commas after the query string.

To see how to create and use stored procedures go to.
https://www.entityframeworktutorial.net/efcore/working-with-stored-procedure-in-ef-core.aspx

IMPROVEMENT NOTES: Add as needed for future revision of content.