Cheatography

Devo LINQ, query language syntax Cheat Sheet

by Rafa Hernández (elpluto) via cheatography.com/46990/cs/21055/

Structure keywords

from	Point to a table for the query. Specify a table name using tags. 👁**from** my.app.we b.auth
where	Where clause to filter results. ☐✔**where** [filter1 expres sion], [filter2 expression] ☐👁String values in expres sions have to be surrounded by double quotes, single quotes are not allowed.
select	Add column to result set: ✔**select** source _column or column operation as destination_column 👁**select** uriHos t(uri) as host
group every / by	Group clause with an optional server and client aggregation period filter: ✔**group** [**every** server _pe riod] by column] [**every** client_period] 👁**group** by statusCode 👁**group** every 10m 👁**group** every 10m by statusCode every 1m
every	Client aggregation filter: ✔**every** period ☐👁Period syntax: an integer number follow by a symbol indicating the time period:**[s:sec onds, m:minutes, h:hours, d:days, no suffix :mi lli - sec onds]** ☐👁**every** 0 means no client period. ☐👁**every** 5m by serverIp
ifthenelse	**if/then/else** equivalent clause to set conditionally column values: ☐✔**ifthenelse**(condi tion, errorV - alue, succes sValue) ☐👁select **ifthenelse**(statu sCode != 200,"Er ror " ,"Su cce ss") as status Cod eDesc

Structure keywords (cont)

decode	**switch/case** equivalent clause to set conditionally column values: ☐✔**decode**(column, checkV alue, value, [ch - eck Value2, value2]) ☐👁Each pair of arguments **[check Value, value]** is equivalent to a case sentence of a switch statement. ☐👁**decode**(statu sCode, 200, " Suc ces s", 400, "Not Found", 406, " Err or", 404, " - Err or") as status Cod eDesc
nvl	**Null-Coalescing** operator. Allow to set an alternate value when input value is null. ✔**nvl**(column, alternate_value_when_null)`

Aggregation functions and operators

avg	Returns the **average** of a range of values [**avg**] or only over *not null* values [**nnavg**] of the results on each group: ☐✔**avg**(column); **nnavg**(column)
count	Returns the **count** of results on each group: ☐✔**count**([column]) ☐👁With argument, include only not null entries in the count.
first / nnfirst	Returns the **first** or the **not null first** entry of the results on each group: ☐✔**first**(column); **nnfirst**(column)

By **Rafa Hernández** (elpluto)
cheatography.com/elpluto/

Published 2nd January, 2023.
Last updated 2nd January, 2023.

Aggregation functions and operators (cont)

last / nnlast	Returns the **last** or the **not null last** entry of the results on each group: ✔ `last(column); nnlast(column)`
max / min	Returns the **maximum** or the **minimum** value for the columns provided, on each group: ✔ `max/min(col1, [col2], [col3]…)`
median	Returns the statistical **median** for a column on each group: ✔ `median(column)` 👁 `Restricted to columns of integer type`
sum	Returns the **sum** of the results on each group: ✔ `sum(column)`
sum2	Returns the **sum** of the squares of the results on each group: ✔ `sum2(column)`
percentile5 **percentile10** **percentile25** **percentile75** **percentile90** **percentile95**	Returns the specific statistic **percentileN**, using linear interpolation, of the results on each group: ✔ `percen tile[N](column)`
stddev / nnstddev	Returns the **biased standard deviation [stddev]** of the values or not null values **[nnstddev]** of the results on each group: ✔ `stddev /nn stddev(column)` 👁 `Biased`

Aggregation functions and operators (cont)

ustddev / nnustddev	Returns the **unbiased standard deviation [ustddev]** of the values or not null values **[unnstddev]** of the results on each group: ✔ `ustdde v/u nns tddev(column)` 👁 `Unbiased`
var / nnvar	Returns the **biased variance [var]** of the values or not null values **[nnvar]** of the results on each group: ✔ `var/nnvar(column)` 👁 `Biased`
uvar / nnuvar	Returns the **unbiased variance [uvar]** of the values or not null values **[unnvar]** of the results on each group: ✔ `uvar/u nnvar(column)` 👁 `Unbiased`
hllpp	Returns the estimated count of distinct values of the results on each group using the **HyperLogLog++** algorithm: ✔ `hllpp(column)` 👁 `Applies on DC (distinct count) data types.`
hllppcount	Returns the estimated count of distinct values of the results on each group using the **HyperLogLog++** algorithm: ✔ `hllppcount(column)` 👁 `Applies on float or integer data types.`

By **Rafa Hernández** (elpluto)
cheatography.com/elpluto/

Published 2nd January, 2023.
Last updated 2nd January, 2023.

String operators and functions

has, [->] Case sensitive **contains** comparison. Using the operator -> only allows check one value:

```
✔has(column, value1, [value2],…)
✔column -> value1
```

weakhas Case insensitive **contains** comparison:

```
weakha s(c olumn, value)
```

in, [<-] Case sensitive **is contained** comparison. Using the operator **'<-'** allows only one value:

```
in(value1, [value 2], [...], column)
value1 <- column
```

weakin Case insensitive **is contained** comparison:

```
weakin (value, column)
```

startswith Returns strings that start with specific value:

```
starts wit h(c olumn, value)
```

endswith Returns strings that end with a specific value:

```
endswi th( column, value)
```

toktains Specialized **contains** function for ASCII delimited tokens:

```
toktai ns( column, value, [bool_ left], [bool_ right])
```

length Returns the length of a string value:

```
length (co lumn)
```

locate Returns the position of a substring, **indexOf** function:

```
locate (co lumn, substr ing _to Locate)
```

lower Returns the transformation to lower case:

```
lower( column)
```

upper Returns the transformation to upper case:

```
upper( column)
```

String operators and functions (cont)

replace Replaces **only first occurrence** of a string with a substitute string:

```
replac e(c olumn, string ToS earch, string ToR eplace)
```

replaceall Replaces **all occurrences** of a search string with a substitute string:

```
replac eal l(c olumn, string ToS earch, string ToR eplace)
```

split Returns a specific piece of splitting operation by a separator:

```
pieceN umber begin at 0.
split( column, separa tor String, pieceN umber)
```

splitre Returns a specific piece of splitting operation by a regular expression:

```
pieceN umber begin at 0.
splitr e(c olumn, re(string) or regexp, pieceN umber)
```

substring Returns a substring beginning at specific index with the provided length:

```
substr ing (co lumn, index, length)
```

subs Returns a string replacing **first substring occurrence** based on a regular expression using a template string as substitution value:

```
FailValue is returned when is provided and no occurr ences found
subs(c olumn, regexp, template, [failValue])
subs(c olumn, re(str ing), templa - te( str ing), [failV alue])
```

By **Rafa Hernández** (elpluto)
cheatography.com/elpluto/

Published 2nd January, 2023.
Last updated 2nd January, 2023.

String operators and functions (cont)

subsall	Returns a string replacing **all substring occurrences** based on a regular expression using a template string as substitution value: 👁 `FailValue is returned when is provided and no occurr ences found` ✔ `subs(c olumn, regexp, template, [failValue])` ✔ `subs(c olumn, re(str ing), templa - te( str ing), [failV alue])`
trim	Returns the result of trimming **both sides**: `trim(c olumn)`
ltrim	Returns the result of trimming **left side**: `ltrim( column)`
rtrim	Returns the result of trimming **right side**: `rtrim( column)`
matches, [~]	Matches function that finds occurrences in a column using a regular expression: ✔`matche s(c olumn, re(string) or regexp value)` ✔`column ~ re(string) or regexp value`
peek	Returns the part of a string based on a regular expression, optionally indicating a specific part occurrence: 👁 `If no partNumber is provided then returns first part occurrence. peek(column, re(string) or regexp, [partN umber])`
formatnumber	Format a number with a specific mask and locale: `format num ber (nu mbe rCo lumn, mask, locale)` 👁`format num ber (to tal Amount, " - ###.## ", " en- GB")`

String operators and functions (cont)

damerau	Returns **Damerau** distance: `damera u(c olumn, value)`
hamming	Returns **Hamming** distance: `hammin g(c olumn, value)`
levenshtein	Returns **Levenstein** distance: `levens hte in( column, value)`
osa	Returns **osa** distance: `osa(co lumn, value)`
publicsuffix	Returns the main public suffix of a hostname: `publicsuffix(hostnameColumn))` 👁 `'www.m y.s ite.co.uk' = 'co.uk'`
rootdomain	Returns the root domain of a hostname part of an url: `rootdomain(hostnameUrlColumn)` 👁 `'www.m y.s ite.com' = 'site'`
rootprefix	Returns the root prefix of a hostname part of an url: `rootpredix(hostnameUrlColumn))` 👁 `'www.m y.s ite.com' = 'www.m - y.site'`
rootsuffix	Returns the root suffix of a hostname part of an url: `rootsuffix(hostnameUrlColumn))` 👁 `'www.m y.s ite.com' = 'my.si - te.com'`
subdomain	Returns the subdomain of a hostname part of an url: `subdomain(hostnameUrlColumn)` 👁 `'www.m y.s ite.com' = 'www'`
topleveldomain	Returns the top level domain of a hostname part of an url:: `topleveldomain(hostnameUrlColumn)` 👁 `'www.m y.s ite.co.uk' = 'uk'`
shannonentropy	

By **Rafa Hernández** (elpluto)
cheatography.com/elpluto/

Published 2nd January, 2023.
Last updated 2nd January, 2023.

Cryptographic functions

md5	
sha1	
sha256	
sha512	

Web functions

urischeme	
urihost	
uriport	
uripath	
urifragment	
uriquery	
uriuser	
urissp	
uriauthority	
absoluteuri	
opaqueuri	
urldecode	
uaurl	
uaname	
uatype	
uaversion	
uaicon	
uarobot	
uainfourl	
uafamily	
uacompany	
uacompanyurl	
uadeviceicon	
uadeviceinfourl	
uadevicetype	
uaosurl	
uaosname	
uaosicon	
uaosfamily	
uaoscompany	
uaoscompanyurl	
uaosversion	Possible **missing** function to filter by OS version.

Meta functions

pragmavalue	
tablename	

Data Types

str	String
int	Integer number: `1,58,12598`
float	Floating point number:`24.256`
boolean	Boolean: `true, false`
timestamp	Timestamp date in format: `yyyy-MM-dd HH:mm: - ss.SSS`
boxar(int)	Byte array in hexadecimal string format
duration	Amount of time: an integer following by a letter `[d]ays, [h]ours, [m]inutes, [s]econds, [No suffix ]:m ill ise conds`
geocord	Geographic coordinates set: ✔ `Latitu de/ lon gitude sexage simal values: 40°24'N 3°41'W` ✔ `Hash repres ent ation of coordi - nates (geohash)`
ip	IPv4 address format: 👁 `192.16 8.5.56`
ip6	IPv6 address format: 👁 `2001:0 db8 :85 a3: - 000 0:0 000 :8a 2e: 037 0:7334`
net4	IPv4 address in format: `{x.x.x.x/0`
net6	IPv6 address in format: `x.x.x.x.x.x/s`
regexp	Regular expression: 👁 `[^\w]`
template	Represents a substitution string mask.
dc	Represents a estimated count of distinct elements in a data stream.
image	Image as Base64 encoding image.
mac	MAC address in format: 👁 `00:0a: 95: 9d: - 68:16`
namepattern	Represents a part of a table name: 👁 `my.app, demo, ...`
set(name)	Represents a set of table names: 👁 `{my.ap - p.test, my.app.test2}`
json	String in json format: 👁 `{"id " :345, " nam - e":"J ohn "}`

Cheatography

Devo LINQ, query language syntax Cheat Sheet

by Rafa Hernández (elpluto) via cheatography.com/46990/cs/21055/

Data Types (cont)

jq	Represents a **jq** filter, **jq** is a command line json processor. `.email`

Common comparison functions and operators

eq, [=]	**Equals to** function and operator:
	`eq(column, value or column)`
	`column1 = value or column`

eqic	Case insensitive **Equals to** function: `eqic(c olumn, value or column)`

ge, [>=]	**Greater or equal** function and operator:
	`ge(column, value or column)`
	`column >= value or column`

gt, [>]	**Greater than** function and operator:
	`gt(column, value or column)`
	`column > value or column`

le, [<=]	**Less or equal** function and operator:
	`le(column, value or column)`
	`column <= value or column`

lt, [<]	**Less than** function and operator:
	`lt(column, value or column)`
	`column < value or column`

ne, [/=]	**Not equal** function and operator:
	`ne(column, value or column)`
	`column /= value or column`

isnull	Check if **is null** function: `isnull (co lumn)`
isnotnull	Check if **is not null** function: `isnotn ull (co lumn)`

Logig Functions

and

or

not

JSON related functions

jqeval

label

jsonparse

Math functions and operators

abs	
add / [+]	
sub / [-]	
mul / [*]	
div / [\]	
rdiv / [/]	Real division function and operator:
mod / [%%]	Module function:
rem / [%]	Return the remain of a division operation:
pow	Power function:
cbrt	Cube root function:
sqrt	Square root function:
ceil	
floor	
round	
signum	

Statistical and specialised statistical functions

estimation

pack

unpackhllpp

Network functions

ispublic

isprivate

ipip4

ipprotocol

purpose

host

routing

httpstatusdescription

httpstatustype

reputation

score

sbl

Cheatography

Devo LINQ, query language syntax Cheat Sheet

by Rafa Hernández (elpluto) via cheatography.com/46990/cs/21055/

Conversion functions

int
str
bool
float
image
ip4
net4
ip6
net6
mac
to16
from16
to64
from64
toutf8
fromutf8
toz85
fromz85
compatible
mapped
translated
template
timestamp
duration
re
parsedate
formatdate
humansize
mkboxar

Special comparison functions

matches Matches function finds occurrences in a column using a regular expression:

✔ `matche s(c olumn, re(string) or regexp value)`

✔ `column ~ re(string) or regexp value`

Special comparison functions (cont)

anymatches Find occurrences in a **set of names** type column against a namepattern:

`anymat che s(s etO fNames, namegl ob( - string) or namepattern)`

👁 `anymat che s(t ables, namegl ob( " - my.a pp.*.* "))`

nameglob Return a formmated string as a **namepattern** to use with **anymatches**:

`namegl ob( string)`

Date and time functions

day
dayofweek
dayofyear
month
year
epoch
hour
minute
second
millisecond
today
tomorrow
yesterday
period

Packet functions

hasio4
hastcp
hasudp
hasether
ip4proto
ip4src
ip4dst
ip4status
ip4ttl
ip4len
ip4payload
ip4flags

By **Rafa Hernández** (elpluto)
cheatography.com/elpluto/

Published 2nd January, 2023.
Last updated 2nd January, 2023.

Packet functions (cont)
ip4fragment
ip4cs
ip4hl
ip4ds
ip4ecn
ip4tos
etherdst
ethersrc
etherpayload
etherstatus
ethertag
ethertype
tcpdst
tcpsrc
tcpstatus
tcpflags
tcppack
tcpcs
tcpseq
tcphl
tcppayload
tcpurg
tcpwin
udpsrc
udpport
udpstatus
udpcs
udplen
udppayload

By **Rafa Hernández** (elpluto)
cheatography.com/elpluto/

Published 2nd January, 2023.
Last updated 2nd January, 2023.