q/kdb+ Cheat-sheet

Basic Commands

:	(Assign)	myVar:2 myFunc:{x+y}
$	(Cast)	`float$4 `$"IBM UN"
/	(Comment)	/ This is a comment // Also this
cut		3 cut til 9 -> (0 1 2;3 4 5)
deltas		deltas 2 3 5 8 -> 2 1 2 3
distinct		3 3 4 4 4 5 -> 3 4 5
_	(Drop)	1 _ 1 2 3 -> 2 3 -1 _ 2 3 -> 2
^	(Fill)	0 ^ 0N 1 2 -> 0 1 2
first		first 5 1 2 -> 5
,	(join)	"var",string 1 -> "var1"
last		last 5 1 2 -> 2
|	(max/or)	| 3 2 5 -> 2 (1=1)|`a=`b -> 1b
&	(min/and)	& 3 2 5 -> 5 (1=1)&`a=`b -> 0b
~	(match)	table1 ~ table1 -> 1b
neg		neg 1 -2 3 -> -1 2 -3
prev		prev 2 3 4 -> 0N 2 3
signum (sign)		signum -4 -2 0 1 -> -1 -1 0 1
reverse		reverse 1 2 3 -> 3 2 1
where		where 000101b -> 3 5

Math

+ - * %	(+;-;*;%) .\: 3 2 -> 5 1 6 1.5
div	10 div 3.14 -> 3
sqrt	sqrt 9 -> 3f
xexp | xlog	2 xexp 3 -> 8f | 2 xlog 8 -> 3f
mod	11 mod 3 -> 2
ceiling/floor	ceiling 1.99 2 2.01 -> 2 2 3
avg/wavg	wavg[100 200;2 5] -> 4f
mavg	mavg[2;1 3 4 5] -> 1 2 3.5 4.5
med	med 2 3 5 8 -> 4f
inv	matrix inversion

n (name)	Char	Literal	Null	Inf
0 (untyped)	*	()		
1 (boolean)	b	0b		
2 (guid)	g		0Ng	
4 (byte)	x	0x00	0x00	
5 (short)	h	0h	0Nh	0Wh
6 (int)	i	0i	0Ni (0N - 2.x)	0Wi
7 (long)	j	0j	0Nj (0N - 3.x)	0Wj
8 (real)	e	0e	0Ne	0We
9 (float)	f	0f	0n	0w
10 (char)	c	" "	" "	
11 (symbol)	s	`	`	
98 (table), 99 (dictionary), 100 (lambda func), etc				

* kdb 2.x default numerical type is int (i), 3.x uses long (j)

Can't find it here? Try code.kx.com

Basic Usage

Define a function:	f:{x + y % z}; f[1;4;2] -> 3
Projection:	g:f[;;2]; g[1;4] -> 3
Pass func as arg:	f:{[x;y;g] g[x;y]} f[2;3;xexp] -> 8
Create a list:	(1 2;`a`b`c;("all";"types";"ok"))
Element-wise ops:	1 + 2 3 4 -> 3 4 5
Indexing:	"string" 2 4 5 -> "rng"
	"string"[2 4 5] -> "rng"
Indexing at depth:	(`a`b;(1;2 3))[1;1;0] -> 2
Closing ")]}" |	myFunc:{[x;y]
Must be off the |	x + 2 * y
line -> |	};

Common System Commands

\c - console size	try \c 30 1000
\p - port	time to think about security!
\P - precision	significant digits
\l - load	kdb databases, files, etc.
\f - functions	lists functions in workspace

Common Built-in Tools

.z.D - today's date (local)
.z.d - today's date (GMT)
.z.T - time now (local) - see also .z.Z, .z.P
.z.t - time now (GMT) - see also .z.z, .z.p
.Q.w[] - Show memory usage - try .Q.w[]%1e9 for GB
tables[] - list tables in memory

Enumeration

enum:distinct l:`a`b`a`c; `enum$l; `enum?`a`b`d

n (name)	Char	Literal	Null	Inf
12 (timestamp)	p	[date]D[timespan]	0Np	0Wp
13 (month)	m	2000.01m	0Nm	
14 (date)	d	2000.01.01	0Nd	0Wd
15 (datetime)	z	[date]T[time]	0Nz	0wz
16 (timespan)	n	00:00:00.000000000	0Nn	0Wn
17 (minute)	u	00:00	0Nu	0Wu
18 (second)	v	00:00:00	0Nv	0Nv
19 (time)	t	00:00:00.000	0Nt	0Wt

Time Casting

`x$date or date.x	x = (year month week mm dd)
`x$time or time.x	x = (minute second hh mm ss)

Dictionary Basics

d:`a`b`c!(1 2;4 5;7 8) ->		`a | 1 2
d[`b] -> 4 5		`b | 4 5
d[`c`b] -> (7 8;4 5)		`c | 7 8
d[`e] -> 0N	(null type matches the type of the dictionary)	
d[`b][1] -> 4	(same as d[`b;1] -> 4)	
d ? 4 5 -> `b	(lookup by value)	
key d -> `a`b`c	value d -> (1 2;4 5;7 8)	

Table Basics

t1:flip d ->		a b c
or		1 4 7
t1:([]a:1 2;b:4 5;c:7 8)		2 5 8
t2:t1 , ([]a:enlist 3;		a b c
b:enlist 6;c:enlist 8) ->		1 4 7
or		2 5 8
t2:t1 upsert (3 6 8) ->		3 6 8

q-sql basics

select a, b from t1 ->		a b
		1 4
		2 5
exec a from t1 -> 1 2		
update d:a + b from t1		a b c d
		1 4 7 5
		2 5 8 7
delete b from t		a c
		2 6
Use '=' for most comparisons:		a b c
select from t where c = 6 ->		2 4 6
But 'like' for strings:		a b c
select from t where (string c) like "6" ->		2 4 6
select avg a by c from t2 ->		c| a
		7| 1
		8| 2.5
select from t2 where		a b c
i=(max;i) fby c ->		1 4 7
		3 6 8
In 'where' clause, put most restrictive filters		
first (usually the date!)		

Accumulators

/ (over)	(+/) 1 2 3 -> 6	
Takes the result of first item(s) and applies to the next item		
\ (scan)	(+\) 1 2 3 -> 1 3 6	
Same as over but shows interim results		

List Joins

, (join)	1 2 3, 2 3 4 -> 1 2 3 2 3 4
,' (zip)	1 2 3,'2 3 4 -> (1 2;2 3; 3 4)
inter	1 2 3 inter 2 3 4 -> 2 3
except	1 2 3 except 2 3 4 -> 1

Table Joins

lj (left join)	all records from left table,
e.g. t1 lj 1! t2	updating with right where applicable
ij (inner join)	only records in both tables
ej (equi join)	like ij, but specify join column(s)
uj (union join)	records from both, both must be keyed
aj (asof join)	joins the most recent match
wj (window join)	applies functions to rolling windows

Functional select/exec/update/delete

? (select/exec)	?[t;c;b;a] -> ?[t;();0b;()]
c is conditional ('where') -> e.g. enlist (<;`vol;11)	
b is by -> e.g. (enlist `DATE)!enlist `DATE	
a is select -> e.g. (`newName`c2)!`c1`c2	
! (update/delete)	![t;c;b;a]
delete from t where i=0 -> ![t;enlist (=;`i;0);0b;`$()]	

Iterators (formerly 'adverbs')

each	count each ("hi";"hat") -> 2 3
\: (each left)	1 2 3 xexp\: 2 3 -> ((1 1);(4 8);(9 27))
/: (each right)	"d" ,/: ("rum";"ac") -> ("drum";"dac")
' (each both)	("snare";"kick") like' ("*a*";"*i*") -> 11b
\:/: (all to all)	("snare";"kick") like\:/: ("*a*";"*i*") -> ((10b);(01b))
cross	("a";"b") cross ("1";"2") -> ("a1";"a2";"b1";"b2")

Networking

'one shot'	`:192.168.0.1:9000 "1+1"
open handle	h:hopen `:192.168.0.1:9000
Send to handle	h "1+1" (sync)
	neg[h] "1+1" (async)
close handle	hclose h
Escape chars:	h "myVar:\"timetorock\""
Also works:	h (set;`myVar;"timetorock")

Conditionals

$ (if/then/else)	$[1=1;2;3] -> 2
Can be extended: $[if;then;elseif;then;else]	
if (if/then)	a:1; if[1=1;a:2]; a -> 2
? (Vector cond)	?[2=1 2 2 3;`a;`b] -> `b`a`a`b