bell.q

```
/ http://inferno.bell-labs.com/cm/cs/who/bwk/interps/pap.html

/ do one million increments on initial state 0
\t do[1000000;i+:1]

/ ackermann's
\t {$[x;.z.s[x-1;$[y;.z.s[x;y-1];1]];y+1]}[3;7]

/ array indexed forwards and backwards
\t x(x;reverse x:til 200000)

/ while(x>count string) join chop join ... on "abcdef"
f:{{500000>count x}{(i _ x),(1+i:floor .5*count x)#x:raze("123";x↩
    ;"456";x;"789")}/x}
\t do[10;f"abcdef"]

/ lookup hex strings in decimal strings
\t {sum("0123456789abcdef"16 vs'x)in string x}til 200000

`f 0:(30000?300)#\:"king "; /james

/ write read file
\t `f 0:read0`f

/ (lines;words;chars) file
\t (count;sum sum each" "=;sum count each)@\:read0`f

/ write reverse read file
\t `f 0:reverse read0`f

`f 0:100000#enlist"-123.456"; / some numbers

/ sum float-from-ascii file
\t sum"F"$read0`f

\
/ approximate times on 100MHZ pentium (32MB)
t:( 2    10     .15   2.2   1   3.5   3.2    4    5.7 /q
       .3   1      .8     5    25  80     50    125   15 /java
     3    40    8       1      6   4     15       8   10. /perl
  100  1000  100      20     12  80     15      70   50.   ) /tcl

the 9 tests are loops, text-processing and text file io.
```

```
even though q avoids all these
 loops - rare, e.g. none in kdb+.
 text  - we prefer data. binary is better.
 stdio - we prefer mmap to read/write.

q is faster (sum of times)
 q(32) perl(95) java(300) tcl(1400+)

q is shorter (lines of code)
 q(9) awk(66) perl(96) tcl(105) scheme(170) vb(200) java(350)
```

cvsguess.q

```
/ guess a reasonable loadstring for a csv file (kdb+ 2.4 or greater)
"kdb+csvguess 0.43 2009.09.19"
/ 2009.09.19 cleanup tests
/ 2009.09.15 add conservative checks for N&P (2.6)
/ 2008.03.03 describe -savedb etc in savescript
/ 2007.12.01 add POSTSAVEALL for SAVE/BULKSAVE - allow disk `p# etc
/ 2007.10.20 catch 0W etc when cancast_ing, don't try and create E
/ 2007.10.17 cleanup D+M support for 2.4, add -z1
/ 2007.09.13 use .Q.res
/ 2007.07.24 allow hhmmss.mmm <-> T
/ 2007.07.13 POSTLOADALL

o:.Q.opt .z.x;if[1>count .Q.x;-2"usage: q ",(string .z.f)," CSVFILE [↩
   -noheader|nh] [-discardempty|de] [-semicolon|sc] [-tab|tb] [-↩
   zaphdrs|zh] [-savescript|ss] [-saveinfo|si] [-zeuro|z1] [-exit]\n↩
   ";exit 1]
/ -noheader|nh - the csv file doesn't have headers, so create some (↩
   c00..)
/ -discardempty|de - if a column is empty don't bother to load it
/ -semicolon|sc - use semicolon as delimiter in place of the default ↩
   comma
/ -tab|tb - use tab as delimiter in place of default comma
/ -zaphdrs|zh - by default junk characters are removed from column ↩
   headers, so for example "Profit & Loss_2005" will become "↩
   ProfitLoss_2005". Use the zaphdrs flag to force the name to ↩
   lowercase and to remove the underscores ("profitloss2005")
/ -savescript|ss - save a standalone load script for the data. Do ↩
   this manually (perhaps after adjusting <info>) by calling ↩
   savescript[]
/ -saveinfo|si - *append* the table information to a shared csv - ↩
   potentially with information from other tables
/ -zeur|zeuro|z1 - set \z 1 for european format dates dd/mm/yy (↩
   default \z 0 mm/dd/yy)
/ −exit − exit on completion, only makes sense in conjunction with ↩
   savescript or saveinfo
/ example:
/ for %1 in (import\*.csv) do q csvguess.q %1 −zh −ss −si −exit

if[(any‘semicolon‘sc in key o)&any‘tab‘tb in key o;−2”delimiter: −tab↩
    OR −semicolon (default \”,\”)”;exit 1]

FILE:LOADFILE:hsym‘${x[where”\\”=x]:”/”;x}first .Q.x
NOHEADER:any‘noheader‘nh in key o
DISCARDEMPTY:any‘discardempty‘de in key o
DELIM:$[any‘semicolon‘sc in key o;”;”;any‘tab‘tb in key o;”\t”;”,”]
ZAPHDRS:any‘zaphdrs‘zh in key o
ZAPHDRS:ZAPHDRS and not NOHEADER
SAVESCRIPT:any‘savescript‘ss in key o
SAVEINFO:any‘saveinfo‘si in key o
if[any‘zeuro‘zeur‘z1 in key o;system”z 1”]
EXIT:‘exit in key o
SYMMAXWIDTH:30 / max symbol width before we just give up and keep as ↩
   * string
SYMMAXGR:10 / max symbol granularity% before we give up and keep as a↩
    * string
WIDTHHDR:25000 / initial width read to look for header record
READLINES:5555 / approximate number of records to check
FORCECHARWIDTH:30 / width beyond which we just set a column to be ↩
   text and finished
CHUNKSIZE:50000000 / chunksize read when bulk load/save − much larger↩
    than safe default in .Q.fs
SAVEDB:‘:csvdb / database top level, where things like ‘:sym live
SAVEPTN:‘ / individual partition, 2006.12.25 frinstance; ‘ => none
PRESAVEEACH:{x} / function to be run before each incremental save (↩
   delete date field?)
POSTLOADEACH:{x} / function to be run after each incremental load ↩
   from file
/ POSTLOADALL:{update ‘p#sym from‘sym‘time xasc x}
POSTLOADALL:{x} / function to be run after complete load from file (↩
   LOAD/BULKLOAD only, not BULKSAVE as never all data in one place)
/ POSTSAVEALL:{@[‘sym‘time xasc x;‘sym;‘p#]}
/ POSTSAVEALL:{dasc[x;‘sym‘time;‘p#]} / faster than xasc on disk
POSTSAVEALL:{x} / function to be run after all saved, to set ‘p# on ‘↩
   sym for example: {@[x;‘sym;‘p#]} or sort by sym {‘sym xasc x}
@[.:;”\\l csvguess.custom.q”;::]; / save your custom settings in ↩
   csvguess.custom.q to override those set above
```

```
if[0=hcount LOADFILE;-2"empty file: ",first .Q.x;exit 1]
sample:last head:read0(LOADFILE;0;1+last where 0xa=read1(LOADFILE;0;↩
   WIDTHHDR))
if[not DELIM in first head;-2"delimiter \"",DELIM,"\" not found in ↩
   first row";exit 1]
readwidth:floor(10+READLINES)*WIDTHHDR%count head
nas:count as:((1+sum DELIM=first head)#"S";enlist DELIM)0:(LOADFILE↩
   ;0;1+last where 0xa=read1(LOADFILE;0;readwidth))
if[0=nas;-2"empty file: ",first .Q.x;exit 1]

cancast:{nw:x$"";if[not x in"BXCS";nw:(min 0#;max 0#;::)@\:nw];$[not ↩
   any nw in x$(11&count y)#y;$[11<count y;not any nw in x$y;1b];0b]}
k)nameltrim:{$[~@x;.z.s'x;~(*x)in aA:.Q.a,.Q.A;(+/&\~x in aA)_x;x]}

info:([]c:key flip as;v:value flip as);as:()
if[NOHEADER;info:update c:{`$"c",string 1000+x}each i from info]
zh0:{$[(count distinct r)=count r:`$"}"vs 1_x[where(x:raze"}",'↩
   nameltrim string x)in"}",.Q.an];r;'`hdrs.not.distinct]} / remove ↩
   junk chars, leading underscores and spaces, preserve case
info:update c:zh0 c from info
zh1:{$[(count distinct r)=count r:`$"}"vs 1_x[where(x:raze"}",'string↩
    lower x)in"}",.Q.an except"_"];r;'`zaphdrs.not.distinct]} / ↩
   lowercase and remove underscores
if[ZAPHDRS;info:update c:zh1 c from info]
/ check for reserved words used as colnames
reserved:key`.q;reserved,:.Q.res;reserved,:`i
info:update res:c in reserved from info
info:update ci:i,t:"?",ipa:0b,mdot:0,mw:0,rule:0,gr:0,ndv:0,maybe:0b,↩
   empty:0b,j10:0b,j12:0b from info
info:update ci:`s#ci from info
info:update sdv:{string(distinct x)except`}peach v from info
info:update ndv:count each sdv from info
info:update gr:floor 0.5+100*ndv%nas,mw:{max count each x}peach sdv ↩
   from info where 0<ndv
/ rule:10 only in csvutil.q
info:update t:"*",rule:20 from info where mw>FORCECHARWIDTH / long ↩
   values
info:update t:"C "[DISCARDEMPTY],rule:30,empty:1b from info where t="↩
   ?",mw=0 / empty columns
info:update dchar:{asc distinct raze x}peach sdv from info where t="?↩
   "
info:update mdot:{max sum each"."=x}peach sdv from info where t="?",↩
   {"."in x}each dchar
info:update t:"n",rule:40 from info where t="?",{any x in↩
   "0123456789"}each dchar / vaguely numeric..
```

```
info:update t:"I",rule:50,ipa:1b from info where t="n",mw within 7 15↩
    ,mdot=3,{all x in".0123456789"}each dchar,cancast["I"]peach sdv / ↩
    ip-address
info:update t:"J",rule:60 from info where t="n",mdot=0,{all x in"+-↩
    0123456789"}each dchar,cancast["J"]peach sdv
info:update t:"I",rule:70 from info where t="J",mw<12,cancast["I"]↩
    peach sdv
info:update t:"H",rule:80 from info where t="I",mw<7,cancast["H"]↩
    peach sdv
info:update t:"F",rule:90 from info where t="n",mdot<2,mw>1,cancast["↩
    F"]peach sdv
info:update t:"E",rule:100,maybe:1b from info where t="F",mw<9
info:update t:"M",rule:110,maybe:1b from info where t in"nIHEF",mdot<↩
    2,mw within 4 7,cancast["M"]peach sdv
info:update t:"D",rule:120,maybe:1b from info where t in"nI",mdot in ↩
    0 2,mw within 6 11,cancast["D"]peach sdv
info:update t:"V",rule:130,maybe:1b from info where t="I",mw in 5 6,7↩
    <count each dchar,{all x like"*[0-9][0-5][0-9][0-5][0-9]"}peach ↩
    sdv,cancast["V"]peach sdv / 235959 12345
info:update t:"U",rule:140,maybe:1b from info where t="H",mw in 3 4,7↩
    <count each dchar,{all x like"*[0-9][0-5][0-9]"}peach sdv,cancast↩
    ["U"]peach sdv /2359
info:update t:"U",rule:150,maybe:0b from info where t="n",mw in 4 5,↩
    mdot=0,{all x like"*[0-9]:[0-5][0-9]"}peach sdv,cancast["U"]peach ↩
    sdv
info:update t:"T",rule:160,maybe:0b from info where t="n",mw within 7↩
     12,mdot<2,{all x like"*[0-9]:[0-5][0-9]:[0-5][0-9]*"}peach sdv,↩
    cancast["T"]peach sdv
info:update t:"V",rule:170,maybe:0b from info where t="T",mw in 7 8,↩
    mdot=0,cancast["V"]peach sdv
info:update t:"T",rule:180,maybe:1b from info where t in"EF",mw ↩
    within 7 10,mdot=1,{all x like"*[0-9][0-5][0-9][0-5][0-9].*"}peach↩
     sdv,cancast["T"]peach sdv
info:update t:"Z",rule:190,maybe:0b from info where t="n",mw within ↩
    11 24,mdot<4,cancast["Z"]peach sdv
info:update t:"P",rule:200,maybe:1b from info where t="n",mw within ↩
    12 29,mdot<4,{all x like"[12]*"}peach sdv,cancast["P"]peach sdv
info:update t:"N",rule:210,maybe:1b from info where t="n",mw within 3↩
     28,mdot=1,cancast["N"]peach sdv
info:update t:"?",rule:220,maybe:0b from info where t="n" / reset ↩
    remaining maybe numeric
info:update t:"C",rule:230,maybe:0b from info where t="?",mw=1 / char
info:update t:"B",rule:240,maybe:0b from info where t in"HC",mw=1,↩
    mdot=0,{$[all x in"01tTfFyYnN";(any"0fFnN"in x)and any"1tTyY"in x↩
    ;0b]}each dchar / boolean
```

```
info:update t:”B”,rule:250,maybe:1b from info where t in”HC”,mw=1,↩
   mdot=0,{all x in”01tTfFyYnN”}each dchar / boolean
info:update t:”X”,rule:260,maybe:0b from info where t=”?”,mw=2,{$[all↩
    x in”0123456789abcdefABCDEF”;(any .Q.n in x)and any”abcdefABCDEF”↩
   in x;0b]}each dchar /hex
info:update t:”S”,rule:270,maybe:1b from info where t=”?”,mw<↩
   SYMMAXWIDTH,mw>1,gr<SYMMAXGR / symbols (max width permitting)
info:update t:”*”,rule:280,maybe:0b from info where t=”?” / the rest ↩
   as strings
/ flag those S/* columns which could be encoded to integers (.Q.j10/↩
   x10/j12/x12) to avoid symbols
info:update j12:1b from info where t in”S*”,mw<13,{all x in .Q.nA}↩
   each dchar
info:update j10:1b from info where t in”S*”,mw<11,{all x in .Q.b6}↩
   each dchar
if[”?”in exec t from info;’‘unknown.field]; / check all done

info:select c,ci,t,maybe,empty,res,j10,j12,ipa,mw,mdot,rule,gr,ndv,↩
   dchar from info
/ make changes to <info>, test with: LOAD10 LOADFILE, or sba[]
/ update t:” ” from‘info where not t=”S” / only load symbols
/ update t:”*” from‘info where t=”S” / load all char as strings, no ↩
   need to enumerate before save
/ run savescript[] when results are correct

k)fs2:{[f;s]((-7!s)>){[f;s;x]i:1+last@&0xa=r:1:(s;x;CHUNKSIZE);f@‘\:i↩
   #r;x+i}[f;s]/0j} / .Q.fs with bigger chunks
disksort:{[t;c;a]if[not‘s~attr(t:hsym t)c;if[count t;ii:iasc iasc ↩
   flip c!t c,:();if[not$[(0,-1+count ii)~(first;last)@\:ii;@[{‘s#x;1↩
   b};ii;0b];0b];{v:get y;if[not$[all(fv:first v)~/:256#v;all fv~/:v↩
   ;0b];v[x]:v;y set v];}[ii]each‘ sv’t,’get‘ sv t,‘.d]];@[t;first c;↩
   a]];t}

SAVENAME:LOADNAME:‘${x where((first x)in .Q.a),1_ x in .Q.an}lower ↩
   first”.”vs last”/”vs 1_string LOADFILE
SAVEPATH:{‘ sv((‘. ‘SAVEDB‘SAVEPTN‘SAVENAME)except‘),‘}
SAVE:{(r:SAVEPATH[])set PRESAVEEACH .Q.en[‘. ‘SAVEDB] x;POSTSAVEALL r↩
   ;r}
DATA:() / delete from‘DATA

/ rebuild globals from <info>
LOADFMTS::raze exec t from‘ci xasc select ci,t from info
JUSTSYMFMTS::{x[where not x=”S”]:” ”;x}LOADFMTS
LOADHDRS::exec c from‘ci xasc select ci,c from info where not t=” ”
JUSTSYMHDRS::LOADHDRS where LOADFMTS=”S”
```

```
status:{ / loadability..
    -1(string`second$.z.t)," FILE:`",(string FILE),"; SAVEDB:`",(↩
       string SAVEDB),"; SAVEPTN:`",(string SAVEPTN),"; SAVENAME:`",(↩
       string SAVENAME),"; \\z ",(string system"z"),"; DELIM:\"",↩
       DELIM,"\"";
    -1(string`second$.z.t)," ",(string count info)," column(s); ",(↩
       string exec count i from info where maybe)," flagged maybe; ",↩
       (string exec count i from info where empty)," empty; ",(string↩
        exec count i from info where res)," with reserved names";}

status[]

LOADDEFN:{(LOADFMTS;$[NOHEADER;DELIM;enlist DELIM])}
JUSTSYMDEFN:{(JUSTSYMFMTS;$[NOHEADER;DELIM;enlist DELIM])}
/ DATA:LOAD LOADFILE / for files loadable in one go
LOAD:{[file] POSTLOADALL POSTLOADEACH$[NOHEADER;flip LOADHDRS!↩
    LOADDEFN[]0:;LOADHDRS xcol LOADDEFN[]0:]file}
/ (10#DATA):LOAD10 LOADFILE / load just the first 10 rows, convenient↩
     when debugging column types
LOAD10:{[file] LOAD(file;0;1+last(11-NOHEADER)#where 0xa=read1(file↩
    ;0;20000))}
BULKLOAD:{[file] fs2[{`DATA insert POSTLOADEACH$[NOHEADER or count ↩
    DATA;flip LOADHDRS!(LOADFMTS;DELIM)0:x;LOADHDRS xcol LOADDEFN[]0: ↩
    x]}file];count DATA::POSTLOADALL DATA}
BULKSAVE:{[file] .tmp.bsc:0;fs2[{.[SAVEPATH[];();,;]PRESAVEEACH t:.Q.↩
    en[`. `SAVEDB]POSTLOADEACH$[NOHEADER or .tmp.bsc;flip LOADHDRS!(↩
    LOADFMTS;DELIM)0:x;LOADHDRS xcol LOADDEFN[]0: x];.tmp.bsc+:count t↩
    }]file;POSTSAVEALL SAVEPATH[];.tmp.bsc}
JUSTSYM:{[file] .tmp.jsc:0;fs2[{.tmp.jsc+:count .Q.en[`. `SAVEDB]↩
    POSTLOADEACH$[NOHEADER or .tmp.jsc;flip JUSTSYMHDRS!(JUSTSYMFMTS;↩
    DELIM)0:x;JUSTSYMHDRS xcol JUSTSYMDEFN[]0: x]}]file;.tmp.jsc}

/ create a standalone load script - savescript[]
/ call it with:
/ q xxx.q / to define all the necessary functions and variables
/ q xxx.q FILENAME  / to define the global FILE as <FILENAME>
/ q xxx.q FILENAME -bl / to bulkload FILENAME to DATA
/ q xxx.q -bl / to bulkload original filename (LOADFILE) to DATA
/ q xxx.q -bs / to bulksave original filename to directory SAVEDB
/ q xxx.q -bs -savedb foo / to bulksave original filename to ↩
    directory foo
/ q xxx.q FILENAME -bs -savedb foo / to bulksave FILENAME to ↩
    directory foo
/ q xxx.q FILENAME -js -savedb foo / to just save the symbols from ↩
```

```
    FILENAME to directory foo (allow parallel load+save thereafter)
/ q xxx.q FILENAME -bs -savedb foo -saveptn 2006.12.25 / to bulksave ↩
    FILENAME to directory foo in the 2006.12.25 date partition
/ q xxx.q FILENAME -bs -savedb foo -saveptn 2006.12.25 -savename goo ↩
    / to bulksave FILENAME to directory foo in the 2006.12.25 date ↩
    partition as table goo
/ q xxx.q ... -exit / exit on completion of commands (only makes ↩
    sense with -bs and -js)
/ q xxx.q .. -chunksize NN / non-default read chunksize - default is ↩
    25
savescript:{f:`$":",(string LOADNAME),".load.q";f 1:"";hs:neg hopen f↩
    ;
     hs"/ ",(string .z.z)," ",(string .z.h)," ",(string .z.u);
     hs"/ q ",(string LOADNAME),".load.q FILE [-bl|bulkload] [-bs|↩
         bulksave] [-js|justsym] [-exit] [-savedb SAVEDB] [-saveptn ↩
         SAVEPTN] [-savename SAVENAME] [-chunksize NNN (in MB)] ";
     hs"/ q ",(string LOADNAME),".load.q FILE";
     hs"/ q ",(string LOADNAME),".load.q";
     hs"/ q ",(string LOADNAME),".load.q -chunksize 11 / test to find ↩
         optimum for your file";
     hs"/ q ",(string LOADNAME),".load.q -savedb DB -saveptn PTN -↩
         savename NAME / to save to `:DB/PTN/NAME/";
     hs"/ q ",(string LOADNAME),".load.q -savedb taq -saveptn 2008.04.↩
         01 -savename trade / to save to `:taq/2008.04.01/trade/";
     hs"/ q ",(string LOADNAME),".load.q -help";
     hs"FILE:LOADFILE:`$\"",(string LOADFILE),"\"";
     hs"o:.Q.opt .z.x;if[count .Q.x;FILE:hsym`${x[where\"\\\\\"=x]:\"/↩
         \";x}first .Q.x]";
     hs"if[`help in key o;-1\"usage: q ",(string LOADNAME),".load.q [↩
         FILE(default",(string LOADFILE),")] [-help] [-bl|bulkload] [-↩
         bs|bulksave] [-js|justsym] [-savedb SAVEDB] [-saveptn SAVEPTN]↩
          [-savename SAVENAME] [-chunksize NNN (in MB)] [-exit]\\n\";↩
         exit 1]";
     hs"SAVEDB:",-3!SAVEDB;hs"SAVEPTN:",-3!SAVEPTN;
     hs"if[`savedb in key o;if[count first o[`savedb];SAVEDB:hsym`$↩
         first o[`savedb]]]";
     hs"if[`saveptn in key o;if[count first o[`saveptn];SAVEPTN:`$↩
         first o[`saveptn]]]";
     hs"NOHEADER:",-3!NOHEADER;hs"DELIM:",-3!DELIM;
     hs"\\z ",(string system"z")," / D date format 0 => mm/dd/yyyy or ↩
         1 => dd/mm/yyyy (yyyy.mm.dd is always ok)";
     hs"LOADNAME:",-3!LOADNAME;hs"SAVENAME:",-3!SAVENAME;hs"LOADFMTS:\↩
         "",LOADFMTS,"\"";hs"LOADHDRS:",raze"`",'string LOADHDRS;
     hs"if[`savename in key o;if[count first o[`savename];SAVENAME:`$↩
         first o[`savename]]]";
     hs"SAVEPATH:",-3!SAVEPATH;hs"LOADDEFN:",-3!LOADDEFN;
     hs"PRESAVEEACH:",-3!PRESAVEEACH;hs"POSTLOADEACH:",-3!POSTLOADEACH↩
         ;hs"POSTLOADALL:",-3!POSTLOADALL;hs"POSTSAVEALL:",-3!↩
         POSTSAVEALL;hs"LOAD:",-3!LOAD;hs"LOAD10:",(-3!LOAD10)," / just↩
          load first 10 records";
     hs"JUSTSYMFMTS:\"",JUSTSYMFMTS,"\"";hs"JUSTSYMHDRS:",$[0=count ↩
         JUSTSYMHDRS;"0#`";raze"`",'string JUSTSYMHDRS];hs"JUSTSYMDEFN:↩
         ",-3!JUSTSYMDEFN;
     hs"CHUNKSIZE:",string CHUNKSIZE;hs"DATA:()";
     hs"if[`chunksize in key o;if[count first o[`chunksize];CHUNKSIZE:↩
         floor 1e6*1|\"I\"$first o[`chunksize]]]";
     hs"k)fs2:",2_ last value fs2;
     hs"disksort:",-3!disksort;
     hs"BULKLOAD:",-3!BULKLOAD;hs"SAVE:",-3!SAVE;hs"BULKSAVE:",-3!↩
         BULKSAVE;hs"JUSTSYM:",-3!JUSTSYM;
     hs"if[any`js`justsym in key o;-1(string`second$.z.t),\" saving `↩
         sym for <\",(1_string FILE),\"> to directory \",1_string ↩
         SAVEDB;.tmp.st:.z.t;.tmp.rc:JUSTSYM FILE;.tmp.et:.z.t;.tmp.fs:↩
         hcount FILE;-1(string`second$.z.t),\" done (\",(string .tmp.rc↩
         ),\" records; \",(string floor .tmp.rc%1e-3*`int$.tmp.et-.tmp.↩
         st),\" records/sec; \",(string floor 0.5+.tmp.fs%1e3*`int$.tmp↩
         .et-.tmp.st),\" MB/sec; CHUNKSIZE \",(string floor 0.5+↩
         CHUNKSIZE%1e6),\")\"]";
     hs"if[any`bs`bulksave in key o;-1(string`second$.z.t),\" saving <↩
         \",(1_string FILE),\"> to directory \",1_string` sv(SAVEDB,↩
         SAVEPTN,SAVENAME)except`;.tmp.st:.z.t;.tmp.rc:BULKSAVE FILE;.↩
         tmp.et:.z.t;.tmp.fs:hcount FILE;-1(string`second$.z.t),\" done↩
          (\",(string .tmp.rc),\" records; \",(string floor .tmp.rc%1e-↩
         3*`int$.tmp.et-.tmp.st),\" records/sec; \",(string floor 0.5+.↩
         tmp.fs%1e3*`int$.tmp.et-.tmp.st),\" MB/sec; CHUNKSIZE \",(↩
         string floor 0.5+CHUNKSIZE%1e6),\")\"]";
     hs"if[any`bl`bulkload in key o;-1(string`second$.z.t),\" loading ↩
         <\",(1_string FILE),\"> to variable DATA\";.tmp.st:.z.t;↩
         BULKLOAD FILE;.tmp.et:.z.t;.tmp.rc:count DATA;.tmp.fs:hcount ↩
         FILE;-1(string`second$.z.t),\" done (\",(string .tmp.rc),\" ↩
         records; \",(string floor .tmp.rc%1e-3*`int$.tmp.et-.tmp.st),\↩
         " records/sec; \",(string floor 0.5+.tmp.fs%1e3*`int$.tmp.et-.↩
         tmp.st),\" MB/sec; CHUNKSIZE \",(string floor 0.5+CHUNKSIZE%1↩
         e6),\")\"]";
     hs"if[`exit in key o;exit 0]";
     hs"/ DATA:(); BULKLOAD LOADFILE / incremental load all to DATA";
     hs"/ BULKSAVE LOADFILE / incremental save all to SAVEDB[/SAVEPTN↩
         ]";
     hs"/ DATA:LOAD10 LOADFILE / only load the first 10 rows";
     hs"/ DATA:LOAD LOADFILE / load all in one go";
```

```
    hs"/ SAVE LOAD LOADFILE / save all in one go to SAVEDB[/SAVEPTN↩
       ]";
    hclose neg hs;f}
if[SAVESCRIPT;-1(string`second$.z.t)," savescript file <",(1_string ↩
   savescript[]),"> written"]

/ save (append) info about the csv columns to INFOFILE - saveinfo[]
/ tbl -tablename; c - column name; ci - column index in csv; t - load↩
    type
/ maybe - set if type is a guess based on name+content (M/D/V/U) or ↩
   could-be symbol
/ mw - maxwidth; j10,j12 - could be encoded using .Q.j10/j12; ipa - ↩
   ip-address
/ gr - granularity% of unique values; dchar - distinct characters
/ info:getinfo[]; update multi:c in exec c from(select count i by c ↩
   from info)where x>1 from`info
INFOFILE:`$":",(lower first"."vs last"/"vs string .z.f),".info.csv"
INFOFMTS:"SSICBBIBBBIS"
readinfo:{(INFOFMTS;enlist",")0:INFOFILE}
saveinfo:{savedinfo:$[@[hcount;INFOFILE;0j];(INFOFMTS;enlist",")0:↩
   INFOFILE;()];
    if[count savedinfo;savedinfo:delete from savedinfo where tbl=↩
       LOADNAME];
    savedinfo,:select tbl:LOADNAME,c,ci,t,maybe,res,mw,j10,j12,ipa,gr↩
       ,`$dchar from info;
    (`$(string INFOFILE),".load.q")1:"info:(",(-3!INFOFMTS),";enlist\↩
       ",\")0:`$\"",(string INFOFILE),"\"\ndups::`c`t xasc select ↩
       from info where c in exec c from select from(select count i by↩
        c from info)where x>1\ninconsistent::select from dups where c↩
        in exec c from(select count i by c from distinct select c,t ↩
       from dups)where x>1\n";
    INFOFILE 0:.h.cd`tbl`c xasc savedinfo;INFOFILE}
if[SAVEINFO;-1(string`second$.z.t)," saveinfo file <",(1_string ↩
   saveinfo[]),"> updated"]
if[EXIT;exit 0]

sba:{update before:(({x[where not x=" "]:"*";x}LOADFMTS;DELIM)0:↩
   sample),after:(LOADFMTS;DELIM)0:sample from select c,t from info} ↩
   / show before+after
forceS:{update t:"S" from`info where t="*"} / no string cols
\
first LOAD10 FILE
select from info where maybe
allfiles:{x where(lower x)like"*.csv"}key`:.
```