1. Given a table t with columns (sym;Value;Sector;Date), create a function to allow the user to select certain columns from the table.

{[t;columns]

if[not all columns in tcol:cols t;

a:columns where not columns in tcol;

'raze "Invalid Column(s): ", "," sv string a

];

?[t;();0b;((),columns)!(),columns]

}

1. Given a table t with columns (sym;MarketCap;Industry), create a query that will filter the table to only show companies whose MarketCap is greater than their industry average.

select from t where MarketCap>(avg;MarketCap) fby Industry

1. Apply list of functions f1:{x+y+z},f2:{x+y-z},f3:{x-y+z} to vector a:1 2 3. Do not use "each"

f:(f1:{x+y+z};f2:{x+y-z};f3:{x-y+z})  
 a:1 2 3  
 f .\: a

1. Describe aj and wj and give examples of how they work

aj or asof join is a join that is specifically tailored for joins on a time column but can be handy for joining on all numeric column types. Here is a dummy example:

t:`time xasc ([] time:.z.t - 1000 \* til 10; sym:10? `VOD.L`AZN.L`BARC.L; price:10?100);

q:`time xasc update bidPrice:askPrice-1 from ([] time:.z.t - 333 \* til 30; sym:30?`VOD.L`AZN.L`BARC.L; askPrice:30?100);

aj[`sym`time;t;q]

For each record in the left table, the prevailing record in the right table will be joined on. The primary join column is the last column in the aj params (in our case time). In contrast to other joins, an exact match is not needed on the primary join column in order for two rows to be joined. The last row is in the right table (by time in our case)

that matches the non-primary join columns (all but the last column in the aj params – just sym in our example) will be the record that is matched. As an added benefit aj can also be used to join two unkeyed tables together.

wj or window join is generates aggregations over “windows in time” defined by the input parameters. For example the following code (assuming the same tables as above code):

wj[win;`sym`time;t;(q;(max;`askPrice);(min;`bidPrice))]

would add columns onto t that correspond to aggregations over q over the times defined in win. The join logic works in much the same way as an aj.

1. A simple way of creating a normally distributed random variable is to generate 12 uniformly distributed numbers between 0 and 1, sum them and subtract 6. Give some q code for creating a list of 100 such variables

1\_{-6+sum 12?1.0}\[100;1]

1. Given a list of integers l:til 10, write some q code that will give the sum of each number and the one previous to it

+':[l:til 10]

1. In kdb+, what is meant by a virtual column? Give an example of how this works.

A virtual column is a column that is not actually stored as part of the table but can be called on if needs be. The index column i is an example of a virtual column. It is not stored in memory or on disk for a table. If it is called upon, it will be added to the table and used as necessary.  
Another example of a virtual column is the date column that appears when a table is stored in date partitioned format. This column is inferred for each record based upon the partition that it sits in.

Example in use:

t:`time xasc ([] time:.z.t - 1000 \* til 10; sym:10? `VOD.L`AZN.L`BARC.L; price:10?100);  
 select count i by sym from t

1. Give some q code for generating a list of 10 random 4 character symbols (all upper case)

`$upper string 10?`4

1. Triangle numbers are generated by the formula:

T(n) = n\*(n+1)/2

Pentagonal numbers are given by:

P(n) = n(3\*n -1)/2

It can be shown that T(285) = P(165) = 40755

Write some q code to find the next triangle number that is also pentagonal.

.ikdb.prevVals:()!();

.ikdb.T:{"j"$%[x\*(x+1);2]}

.ikdb.P:{"j"$%[x\*(-1+3\*x);2]};

.ikdb.PaT:{[num]

//Calculate new values

tVal:.ikdb.T[num];

.ikdb.prevVals,:enlist[.ikdb.P num]!enlist num;

//Breakout condition

If[(tVal in key .ikdb.prevVals) & (tVal > 40755);

.ikdb.res:(num;.ikdb.prevVals[tVal];tVal);

.ikdb.prevVals:()!();

:num

];

//Clean up prevVals list

allKys:key .ikdb.prevVals;

.ikdb.prevVals:(allKys where allKys < tVal) \_ .ikdb.prevVals;

//Increment 1 to keep iteration going

num+1

};

Place the above code in a script, load it in and run the following lines:

.ikdb.PaT/[1]  
 .ikdb.res

1. Given the string “mississippi”, write some q code that will return a dictionary which maps each unique letter to the number of times it appears in the string.

dict:group "mississippi";  
 key[dict]!count each value dict

1. Given a table t with columns (sym;exch), where sym and exch are symbols, write a query which creates a new column “ric” which is of the form sym.exch, which is also a symbol

i.e.

sym exch sym exch ric

-------- => --------------

VOD L VOD L VOD.L

IBM N IBM N IBM.N

t:([] sym:`VOD`IBM; exch:`L`N);  
 update ric:(sv[`;] each sym,'exch) from t

1. Given a list

L:1 3 2 3 4 8 7 8 9 10 11 20 74 28 99

Write some code that will multiply each odd number by 10.

L\*count[L]#(10;1)

1. What is the functional form of the query

“select count distinct sym by date from trade where date within 2013.07.01 2013.07.05”

?[trade;enlist(within; `date; 2013.07.01 2017.07.05); enlist[`date]!enlist`date; enlist[`sym]!enlist(count; (distinct; `sym))];

1. Given a matrix/list of lists

M:(1 9 10;120 50 199;15 20 7)

Return only the rows where every element in the row is greater than 5.

M where all M>5

1. Given a table t with columns (time;sym;price), where some of the price fields are null, write a query to fill each null price field with the last non-null price for that sym

update fills price by sym from t