*Advanced KDB Solutions – Kevin Quigley*

*Hello and welcome to my repo generating solutions to the Advanced KDB exam from First Derivatives*

*Q Common libraries taken from* [*https://github.com/BuaBook/kdb-common/tree/master/src/cron.q*](https://github.com/BuaBook/kdb-common/tree/master/src/cron.q)

*Q Tick libraries taken from* [*https://github.com/KxSystems/kdb-tick*](https://github.com/KxSystems/kdb-tick)

**Requirements**

Kdb+ must be installed – with the q executable in ~/q/m64/q. This must be added to the .bashrc folder

**Useful information**

- Q Common libraries are taken from[*https://github.com/BuaBook/kdb-common/tree/master/src/cron.q*](https://github.com/BuaBook/kdb-common/tree/master/src/cron.q) *-* Q Tick libraries taken from[*https://github.com/KxSystems/kdb-tick*](https://github.com/KxSystems/kdb-tick)

- The variables for ports and other environmental variables are in  *~/advancedKDB-master/src/env.sh*

-The command *“Source ~/advancedKDB-master/src/env.sh”* sets environmental variables, if they are not set already.

**Initialization**

1. Unzipping. The unzip location for the Advanced kdb folder can be set anywhere. *(For this example, the project is unzipped into the home-directory (~)).*
2. To start all processes mentioned in exercises (1-6)
   1. cd to the src folder, by executing

* 1. Execute the following in terminal
  2. Type “y” when prompted if you would like to start all processes, or one process

*Please ensure that when stopping processes, use the above bash STOP.sh command. This prevents processes from being incorrectly shown as being available when it is down.*

***Q1 –Tickerplant***

* Defined in *~/advancedKDB-master/src/tick.q*. The schema for all other tables is defined in sym.q. During startup, the ports are set via env.sh. The port values are set in /env.sh
* To launch the tickerplant without a bash script:-
  + First cd to the src folder
  + Execute

***Q2 RDB***

* Prior to launch the tp must be active, and the port of the tp must be passed. In this case, it’s ‘6800’
* To launch the rdb without a bash script:-
  + First cd to the src folder
* Execute

***Q3 Feed handler***

* To launch the feed handler without a bash script:-
  + First cd to the src folder
* Execute

***Q 4 CEP***

* To launch the feed handler without a bash script:-
  + First cd to the src folder
* Execute

***Q5 Logging***

* Logging file instructions:*- ~/advancedKDB-master/log4q-master/log4q.q*
* *Execution example*
* First cd to the src folder
* Execute
* In the q terminal, type
* This will add the following to the log of the associated instance, in this case, to view in another terminal
* Should show a line similar to below

*“INFO [2022.09.26D20:03:09.303421000]:PID=41010:./fh.q: TEST”*

***Q6 Startup Shutdown scripts***

1. To start all processes
   1. cd to the src folder, by executing

* 1. Execute the following in terminal
  2. Type “y” when prompted if you would like to start all processes, or one process

1. To start a single process, say the tickerplant
2. cd to the src folder, as above in step 2.a.
3. Execute the following in terminal
4. Type *“o”* when prompted if you would like to start all processes, or one process
   1. Type *“tp”* when prompted to specify the tickerplant.
5. To stop all processes
   1. cd to the src folder, as shown in step 2.a.
   2. Execute
   3. Type “y” when prompted if you would like to stop all processes, or one process
6. To stop a single process, say the tickerplant
7. cd to the src folder, as above in step 1.a.
8. Execute the following in terminal
9. Type *“o”* when prompted if you would like to stop all processes, or one process
10. Type *“tp”* when prompted to specify the tickerplant.
11. To test all processes
    1. cd to the src folder, as shown in step 2.a.
    2. Execute
    3. Type “y” when prompted if you would like to test all processes, or one process
12. To test a single process, say the tickerplant
13. cd to the src folder, as above in step 1.a.
14. Execute the following in terminal
15. Type *“o”* when prompted if you would like to test all processes, or one process
16. Type *“tp”* when prompted to specify the tickerplant.

***Q7 TP Replay***

1. To replay a tp log and create a record for IBM.n
   1. cd to the src folder, by executing

* 1. Execute the following in terminal, where the date is set to an available date
  2. *Output Location*

***Q8 CSV Loader***

1. To replay a tp log and create a record for IBM.n
2. cd to the src folder, by executing

1. Execute the following in terminal to load the csv file ~/advancedKDB-master/src/Quote.csv
2. To view the new quotes by launching a q process and connecting the rdb

***Q9 HDB Compression***

1. To replay a tp log and create a record for IBM.n
2. cd to the src folder, by executing

1. Execute the following in terminal to compress data from 2022.09.28

1. This data is outputted to *~/advancedKDB-master/src/hdb*

***Q10 Effect of schema change***

1. The symfile (In *~/advancedKDB-master/src/sym.q*) would have to updated
2. Changing the names of the columns could result in issues in compression, if the date or symbol column is changed.
3. Re-ordering the columns could cause issues with functions dependend on specific columns
4. the old hdb would need to backfilled with the new order/ new column. This will involve modifying the file on each partition to account for the new column
5. The TP would need to be taken offline, after EOD. This would be the best time to execute such a change, as opposed to during a release.
6. If the table is changed during the day, all subscriber which hold data will need to be updated in memory.

***PART 2:- Debugging***

***Q1. TP\_log***

Running the line

Yields the following issues:-

* The log entry is missing rows between 6-8
* The type of the size column is inconsistant between rows
* Some of the types in the sym columns are inconsistant

Due to the very small size of the log file, my solution is to read the old broken log into memory, modify fields where required and the save down a new log file ‘new\_log’. This can be renamed to tplog if required.

**Steps:**

* Assign the old log table

* Correcct the corrupted line which was split across multiple lines
* Update datatypes on sym column
* Ensure that the types of the numerical column are the cast to float
* Resave the table as follows

Following above the trade table can be queried. The steps above are saved a q file, logDebugger.q.

**Part 2 – located in db folder- tableDebugger.q - Splay Table problem**

* Moving the .d file in table number 1
* Fixing the price columns in table 2
* Fixing table 3 enumeration

**Debugging Part 3:- Blocking calls**

If multiple users are connecting to a single kdb process each query may be ran an inefficient manner, or it could be the case that slowness is being caused by having each query executed one after another

The most obvious solution is to have two or more hdb’s located in the same directory. Having a single gateway process which routes traffic and balances it between each hdb would help to minimize slowness.

Another advantage to routing queries through a gateway is that by setting the api calls, you can make sure that there are limits being applied to the calls, ie no query that calls to a splayed table without having the date as the first paramater, or setting date limits on queries on very large tables.

**Debugging Part 4:- Query Performance**

***API***

**API Part 1**

* Completed using python2
* Execute:
* This publishes the table *~/advancedKDB-master/src/Quote.csv* to the RDB.

**API Part 2**

* The C-API is located in the advancedKDB/src/CAPI folder, it needs to be build with the locally build c-libaray for kdb (k.h, will error unless this is build correctly).
* File is built using
* Executed using the following in the folder *~/advancedKDB-master/src/CAPI*

**API Part 3**

* The html is located in the folder *~/advancedKDB-master/src/CAPI/html/ .*
* First the q query process is launched using
* On a remote machine, the html service can be run with the line:-
* To query from a remote machine, enitialize port forwarding:-
* On your local machine, open localhost:8787 in browser and kdb commands can be run. Note that the permissions between the python process and the html process must be be set to allow traffic
* Got to site <http://localhost:8787/>