Outline

Often ingestion tasks are written to simply dump data to disk. A second job then had to perform a disk sort. This is slow.

This notebook shows how to perform a distributed ingest while also sorting.

Create sample table

```
//Imports and setup
p)import warnings
p)warnings.filterwarnings("ignore")
p)import pandas as pd
p)import numpy as np
p)import pyarrow as pa
p)import pyarrow.parquet as pq
```

```
p)times=[np.datetime64('2012-06-30T21:00:00.000000000-0400')] * 4
p)table=pd.DataFrame(columns=['time','sym','price','size'])
p)table['time'] = times
p)table['sym'] = ['a','b','a','b']
p)table['price'] = [4.0,3.0,2.0,1.0]
p)table['size'] = [100,200,300,400]
p)print(table)
```

```
                 time sym  price  size
0 2012-07-01 01:00:00   a    4.0   100
1 2012-07-01 01:00:00   b    3.0   200
2 2012-07-01 01:00:00   a    2.0   300
3 2012-07-01 01:00:00   b    1.0   400
```

```
p)pq.write_table(pa.Table.from_pandas(table), 'example.parquet')
```

Sorting

The important change in this example is that we extract the columns we wish to sort on in the master process.

Using these the correct sort index for the data is created.

This is then sent to all slave processes which will use it to correctly save each column in the same sort order

```
sortTab:flip `time`sym!{[c] $[(`$c) in key conversionsQ;conversionsQ[`$c];(::)]
                        getColumnWithConversion[file;c;conversionsPY`$c]} each
2#columns;
sortTab:update ind:i from sortTab;
sortTab:update `p#sym from `sym`time xasc sortTab;
.Q.dd[destination;`] set .Q.en[`.] delete ind from sortTab;

-25!(.z.pd;({sortInd::x};sortTab`ind));
```

Running the example

Start your worker processes

```
q qparquet.q - p 5001 &
q qparquet.q - p 5002 &
q qparquet.q - p 5003 &
```

Run the master process to distribute the work

```
q convert.q -s -3 -slaves 5001 5002 5003
```

The output shows that the qparquet data is now successfully a q splayed table with correct sort and attributes

```
`:splayed/.d
"Took 0D00:00:00.064986000"
`splayed
time                          sym price size
-----------------------------------------
2012.07.01D01:00:00.000000000 a   4     100
2012.07.01D01:00:00.000000000 a   2     300
2012.07.01D01:00:00.000000000 b   3     200
2012.07.01D01:00:00.000000000 b   1     400
c    | t f a
-----| -----
time | p
sym  | s   p
price| f
size | j
```

Files

convert.q

This script coordinates distributing the work of converting the parquet file across multiple processes

```
//Load needed functions
\l qparquet.q

//Open handles to worker processes
.z.pd:`u#asc hopen each"J"$(.Q.opt .z.X)`slaves

file:"example.parquet";

columns:-1_getColumnNames[file]`

destination:`:splayed

conversionsPY:enlist[`time]!enlist
"table[\"time\"]=pd.to_numeric(table[\"time\"])";
conversionsQ:`time`sym!({`timestamp$x-`long$2000.01.01D-1970.01.01D};`$);

start:.z.p;

sortTab:flip `time`sym!{[c] $[(`$c) in key conversionsQ;conversionsQ[`$c];(::)]
                          getColumnWithConversion[file;c;conversionsPY`$c]} each
2#columns;
sortTab:update ind:i from sortTab;
sortTab:update `p#sym from `sym`time xasc sortTab;
.Q.dd[destination;`] set .Q.en[`.] delete ind from sortTab;

-25!(.z.pd;({sortInd::x};sortTab`ind));

//Distribute tasks to workers
//Each worker reads a column at a time
{[f;d;convPY;convQ;c]
  show string[.z.p],"  ",c;
  .Q.dd[d;`$c] set {x[sortInd]}
    $[(`$c) in key convQ;
      convQ[`$c];
          (::)]
        $[(`$c) in key convPY;
             getColumnWithConversion[f;c;convPY`$c];
                 getColumn[f;c]]
    }[file;destination;conversionsPY;conversionsQ] peach 2_columns;

//Add a .d file to the destination to inform q of the order of columns
.Q.dd[destination;`.d] set `$columns

end:.z.p;

show "Took ",string end-start;

//Load the converted table
\l splayed

//Query the q table
show select from splayed
```

```
show meta splayed
```

qparquet.q

This file contains needed imports and functions

```
//parquet library prints many warnings - ignore for this example
p)import warnings
p)warnings.filterwarnings("ignore")

//Import pandas, numpy, and pyarrow
p)import pandas as pd
p)import numpy as np
p)import pyarrow as pa
p)import pyarrow.parquet as pq

p)def getColumnNames(file):    return (pq.read_schema(file)).names

getColumnNames:.p.get`getColumnNames

p)def getColumns(file, cols): table=pq.read_table(file, columns=cols); return
(table.to_pandas()).to_dict('list')

getColumns:.p.get`getColumns

getColumn:{[file;column] first value getColumns[file;enlist column]`}

.p.e "def getColumnWithConversion(file, col, conversion): ",
                "table=pq.read_table(file, columns=[col]);",
                "table=table.to_pandas();",
                "exec(conversion);",
                "return table.to_dict('list')";

getColumnWithConversionPY:.p.get`getColumnWithConversion

getColumnWithConversion:{[f;c;conv] first value
getColumnWithConversionPY[f;c;conv]`}
```