Outline

This notebook combines previous examples. Scaling the ingestion with casting rules per columns.

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','one'])
p)table['time'] = times
p)table['one'] = [1,2,3,4]
p)print(table)
```

```
                 time  one
0 2012-07-01 01:00:00    1
1 2012-07-01 01:00:00    2
2 2012-07-01 01:00:00    3
3 2012-07-01 01:00:00    4
```

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

Castings rules

The difference with this example to previous is that a dictionary is used to define rules to apply to each column.

```
//eval is use is python to apply whatever code yo uneed to prepare the column
conversionsPY:enlist[`time]!enlist
"table[\"time\"]=pd.to_numeric(table[\"time\"])";
//Once in q you have a similar option to apply a rule
conversionsQ:enlist[`time]!enlist {`timestamp$x-`long$2000.01.01D-1970.01.01D};
```

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 distibute 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

```
`:splayed/time`:splayed/one
`:splayed/.d
"Took 0D00:00:00.021990000"
`splayed
time                          one
-------------------------------
2012.07.01D01:00:00.000000000 1
2012.07.01D01:00:00.000000000 2
2012.07.01D01:00:00.000000000 3
2012.07.01D01:00:00.000000000 4
```

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:enlist[`time]!enlist {`timestamp$x-`long$2000.01.01D-1970.01.01D};

start:.z.p;
```

```
//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
    $[(`$c) in key convQ;
      convQ[`$c];
          (::)]
        0N!$[(`$c) in key convPY;
             getColumnWithConversion[f;c;convPY`$c];
                 getColumn[f;c]]
    }[file;destination;conversionsPY;conversionsQ] peach 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
```

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]`}
```