qStudio/PRQL Lesson #4 - More Transforms

PRQL has more *transforms* (statements) that modify a table as it passes through its pipeline. *Much of the discussion below comes from the PRQL Reference Tutorial*

We have already introduced several PRQL transforms. Let's review how each changes the "shape" of the table:

- `from` - begins a pipeline and passes the entire table to the next transform.
- `select` - Changes the number of columns by retaining only those named within the tuple (the list of column names inside the `{ ... }`) but never the number of rows.
- `filter` - Changes the number of rows by excluding the ones that don't match the criteria. Never changes the number of columns.
- `derive` - Changes the number of columns by adding a new column, calculated from other columns in the row. Never changes the number of rows.
- `sort` - Changes the *order* of the rows, but leaves the number of rows and columns unchanged.

This lesson talks about these new transforms.

`take`

The `take` transform picks rows to pass through based on their position within the table. The number of columns is unchanged. The set of rows picked can be specified in two ways:

- a plain number `x`, which will pick the first x rows, or
- an inclusive range of rows `start..end`

For example:

```
from invoices
take 4         # takes the first four lines

# - or -

from invoices
take 4..7      # takes rows 4, 5, 6, and 7
```

`aggregate`

The `aggregate` transform takes a tuple (list of column names) that “distill down” data from all the rows into a single row. This is frequently used for statistical analysis.

```
from invoices
aggregate { sum_of_orders = sum total }
```

If the "invoices" table has a column named `total` (perhaps it's the total of a single order in that row), the query above woud compute the sum of the total column of all rows of the invoices table to produce a single row.

The number of columns is equal to the number of items within the tuple. In the example above, the result would be *one* column. In the example below, the resulting table has *two* columns - `sum_of_orders` and `avg_of_orders`

```
from invoices
aggregate {
    sum_of_orders = sum total,
    avg_of_orders = average total,
}
```

`group`

I didn't finish this...

See the "grouping" section of the PRQL docs

`join`

I didn't finish this...

See the "join" section of the PRQL docs