examples/statistics_aggr_2.pql by *Pequel*

sample@youraddress.com

Statistics Aggregates Example Script

2.2

Table of Contents
Statistics Aggregates Example Script

SCRIPT NAME

examples/statistics_aggr_2.pql

DESCRIPTION

Demonstrate various statistical aggregates.

1. PROCESS DETAILS

Input records are read from standard input. The input record contains ***8*** fields. Fields are delimited by the '**|**' character.

Output records are written to standard output. The output record contains ***16*** fields. Fields are delimited by the '**|**' character.

Input stream is **sorted** by the input field ***SALES_CODE*** (*string*).

Input records are **grouped** by the input field ***SALES_CODE*** (*string*).

1.1 *SALES_CODE*

Output Field

Description
Set to input field ***SALES_CODE***

1.2 *NUM_PRODUCTS*

Output Field

Description
Distinct aggregation on input field ***PRODUCT_CODE***.

1.3 *AVG_COST_PRICE*

Output Field

Description
Avg aggregation on input field ***COST_PRICE***.

1.4 *VALUES_QTY*

Output Field

Description
Values_all aggregation on input field ***SALES_QTY***.

1.5 *DISTINCT_QTY*

Output Field

Description
Distinct aggregation on input field ***SALES_QTY***.

1.6 *MEDIAN_QTY*

Output Field

Description

Median aggregation on input field ***SALES_QTY***.

1.7 *VARIANCE_QTY*

Output Field

Description

Variance aggregation on input field ***SALES_QTY***.

1.8 *STDDEV_QTY*

Output Field

Description

Stddev aggregation on input field ***SALES_QTY***.

1.9 *MAX_QTY*

Output Field

Description

Max aggregation on input field ***SALES_QTY***.

1.10 *MIN_QTY*

Output Field

Description

Min aggregation on input field ***SALES_QTY***.

1.11 *TEST_4*

Output Field

Description

Derived (calculated) field.

Derived Field Evaluation

1.12 *RANGE_QTY*

Output Field

Description

Range aggregation on input field ***SALES_QTY***.

1.13 *MODE_QTY*

Output Field

Description

Mode aggregation on input field ***SALES_QTY***.

1.14 *RANGE_QTY_2*

Output Field

Description

Derived (calculated) field.

Derived Field Evaluation

1.15 *RANGE_QTY_3*

Output Field

Description

Derived (calculated) field.

Derived Field Evaluation

1.16 *LOCATION*

Output Field

Description

Set to input field ***LOCATION***

2. CONFIGURATION SETTINGS

2.1 *prefix*
directory pathname prefix.: examples

2.2 *pequeldoc*
generate pod / pdf pequel script Reference Guide.: pdf

2.3 *detail*
Include Pequel Generated Program chapter in Pequeldoc: 1

2.4 *script_name*
script filename: examples/statistics_aggr_2.pql

2.5 *header*
write header record to output.: 1

2.6 *discard_header*
Input file has header record - must be discarded.: 1

2.7 *optimize*
optimize generated code.: 1

2.8 *doc_title*
document title.: Statistics Aggregates Example Script

2.9 *doc_email*
document email entry.: sample@youraddress.com

2.10 *doc_version*
document version for pequel script.: 2.2

3. TABLES

4. TABLE INFORMATION SUMMARY

4.1 Table List Sorted By Table Name

5. EXAMPLES/STATISTICS_AGGR_2.PQL

options

```
prefix(examples)
pequeldoc(pdf)
detail(1)
script_name(examples/statistics_aggr_2.pql)
header(1)
discard_header(1)
optimize(1)
doc_title(Statistics Aggregates Example Script)
doc_email(sample@youraddress.com)
doc_version(2.2)
```

description

```
Demonstrate various statistical aggregates.
```

input section

```
PRODUCT_CODE
COST_PRICE
DESCRIPTION
SALES_CODE
SALES_PRICE
SALES_QTY
SALES_DATE
LOCATION
```

sort by

```
SALES_CODE string
```

group by

```
SALES_CODE string
```

output section

```
string    SALES_CODE     SALES_CODE
numeric   NUM_PRODUCTS   distinct PRODUCT_CODE
numeric   AVG_COST_PRICE avg COST_PRICE
string    VALUES_QTY     values_all SALES_QTY
numeric   DISTINCT_QTY   distinct SALES_QTY
numeric   MEDIAN_QTY     median SALES_QTY
numeric   VARIANCE_QTY   variance SALES_QTY
numeric   STDDEV_QTY     stddev SALES_QTY
numeric   MAX_QTY        max SALES_QTY
numeric   MIN_QTY        min SALES_QTY
numeric   TEST_4         = MEDIAN_QTY
numeric   RANGE_QTY      range SALES_QTY
numeric   MODE_QTY       mode SALES_QTY
numeric   RANGE_QTY_2    = RANGE_QTY * 2
numeric   RANGE_QTY_3    = RANGE_QTY_2 * 3
numeric   LOCATION       LOCATION
```

6. PEQUEL GENERATED PROGRAM

```
#!/usr/bin/perl
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
# vim: syntax=perl ts=4 sw=4
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#Generated By: pequel Version 2.4-5, Build: Wednesday November 16 21:56:42 GMT 2005
#            : http://sourceforge.net/projects/pequel/
#Script Name : statistics_aggr_2.pql
#Created On  : Wed Nov 16 14:21:04 2005
#Perl Version: /usr/bin/perl 5.6.1 on solaris
#For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#Options:
#prefix(examples) directory pathname prefix.
#pequeldoc(pdf) generate pod / pdf pequel script Reference Guide.
#detail(1) Include Pequel Generated Program chapter in Pequeldoc
#script_name(examples/statistics_aggr_2.pql) script filename
#header(1) write header record to output.
#discard_header(1) Input file has header record - must be discarded.
#optimize(1) optimize generated code.
#doc_title(Statistics Aggregates Example Script) document title.
#doc_email(sample@youraddress.com) document email entry.
#doc_version(2.2) document version for pequel script.
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
use strict;
use constant _I_PRODUCT_CODE      => int    0;
use constant _I_COST_PRICE        => int    1;
use constant _I_DESCRIPTION       => int    2;
use constant _I_SALES_CODE        => int    3;
use constant _I_SALES_PRICE       => int    4;
use constant _I_SALES_QTY         => int    5;
use constant _I_SALES_DATE        => int    6;
use constant _I_LOCATION          => int    7;
use constant _O_SALES_CODE        => int    1;
use constant _O_NUM_PRODUCTS      => int    2;
use constant _O_AVG_COST_PRICE    => int    3;
use constant _O_VALUES_QTY        => int    4;
use constant _O_DISTINCT_QTY      => int    5;
use constant _O_MEDIAN_QTY        => int    6;
use constant _O_VARIANCE_QTY      => int    7;
use constant _O_STDDEV_QTY        => int    8;
use constant _O_MAX_QTY           => int    9;
use constant _O_MIN_QTY           => int   10;
use constant _O_TEST_4            => int   11;
use constant _O_RANGE_QTY         => int   12;
use constant _O_MODE_QTY          => int   13;
use constant _O_RANGE_QTY_2       => int   14;
use constant _O_RANGE_QTY_3       => int   15;
use constant _O_LOCATION          => int   16;
local $\="\n";
local $,="|";
print STDERR '[examples/statistics_aggr_2.pql ' . localtime() . "] Init";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 7;
my @I_VAL;
my @O_VAL;
my $_inprecs=0;
my %DISTINCT;
my %AVERAGE;
my %VALUES_ALL;
my %MEDIAN;
my %MEDIAN_COUNT;
my %VARIANCE;
my %STDDEV;
my %RANGE;
my %MODE;
my $key__I_SALES_CODE;
my $previous_key__I_SALES_CODE = undef;
foreach my $f (1..16) { $O_VAL[$f] = undef; }
# Sort:SALES_CODE(asc:string)
open(DATA, q{cat  - | sort  -t'|' -y -k 4,4 2>/dev/null |}) || die "Cannot open input: $!";
&PrintHeader();
print STDERR '[examples/statistics_aggr_2.pql ' . localtime() . "] Start";
use Benchmark;
my $benchmark_start = new Benchmark;
my $discard_header = <DATA>;
while (<DATA>)
{
    ++$_inprecs;
    print STDERR '[examples/statistics_aggr_2.pql ' . localtime() . "] $_inprecs records." if ($_inprecs % VER
BOSE == 0);
```

```
    chomp;
    @I_VAL = split("[|]", $_);
    $key__I_SALES_CODE = $I_VAL[_I_SALES_CODE];
    if (!defined($previous_key__I_SALES_CODE))
    {
        $previous_key__I_SALES_CODE = $key__I_SALES_CODE;
    }

    elsif ($previous_key__I_SALES_CODE ne $key__I_SALES_CODE)
    {
        $O_VAL[_O_AVG_COST_PRICE] = ($AVERAGE{_O_AVG_COST_PRICE}{_COUNT} == 0 ? 0 : $AVERAGE{_O_AVG_COST_PRICE
}{_SUM} / $AVERAGE{_O_AVG_COST_PRICE}{_COUNT});
        $O_VAL[_O_VALUES_QTY] = join(qq{,}, grep(length, @{$VALUES_ALL{_O_VALUES_QTY}}));
        $O_VAL[_O_MEDIAN_QTY] = $MEDIAN_COUNT{_O_MEDIAN_QTY} % 2 == 0 ? &{sub{($_[0] + $_[1]) / 2}}((( sort {$
a <=> $b} keys %{$MEDIAN{_O_MEDIAN_QTY}} )[$MEDIAN_COUNT{_O_MEDIAN_QTY}/2-1, $MEDIAN_COUNT{_O_MEDIAN_QTY}/2])[
0,1]) : (sort {$a <=> $b} keys %{$MEDIAN{_O_MEDIAN_QTY}} )[(($MEDIAN_COUNT{_O_MEDIAN_QTY}+1)/2)-1];
        $O_VAL[_O_VARIANCE_QTY] = ($VARIANCE{_O_VARIANCE_QTY}{_SUM_SQUARES} / ($VARIANCE{_O_VARIANCE_QTY}{_COU
NT} == 0 ? 1 : $VARIANCE{_O_VARIANCE_QTY}{_COUNT}))- (($VARIANCE{_O_VARIANCE_QTY}{_SUM} / $VARIANCE{_O_VARIANC
E_QTY}{_COUNT}) ** 2);
        $O_VAL[_O_STDDEV_QTY] = sqrt(($STDDEV{_O_STDDEV_QTY}{_SUM_SQUARES} / ($STDDEV{_O_STDDEV_QTY}{_COUNT} =
= 0 ? 1 : $STDDEV{_O_STDDEV_QTY}{_COUNT}))- (($STDDEV{_O_STDDEV_QTY}{_SUM} / $STDDEV{_O_STDDEV_QTY}{_COUNT}) *
* 2));
        $O_VAL[_O_TEST_4] = $O_VAL[_O_MEDIAN_QTY];
        $O_VAL[_O_RANGE_QTY] = $RANGE{_O_RANGE_QTY}{_MAX} - $RANGE{_O_RANGE_QTY}{_MIN};
        $O_VAL[_O_MODE_QTY] = join(’ ’, &{sub{my @top; foreach my $k (sort { $MODE{_O_MODE_QTY}{$b} <=> $MODE{
_O_MODE_QTY}{$a} } keys %{$MODE{_O_MODE_QTY}} ){ last if ($MODE{_O_MODE_QTY}{$k} != $MODE{_O_MODE_QTY}{$_[0]})
; push(@top, $k);} @top; }}((sort { $MODE{_O_MODE_QTY}{$b} <=> $MODE{_O_MODE_QTY}{$a} } keys %{$MODE{_O_MODE_Q
TY}} )[0]));
        $O_VAL[_O_RANGE_QTY_2] = $O_VAL[_O_RANGE_QTY] * 2;
        $O_VAL[_O_RANGE_QTY_3] = $O_VAL[_O_RANGE_QTY_2] * 3;
        print STDOUT
            $O_VAL[_O_SALES_CODE],
            $O_VAL[_O_NUM_PRODUCTS],
            $O_VAL[_O_AVG_COST_PRICE],
            $O_VAL[_O_VALUES_QTY],
            $O_VAL[_O_DISTINCT_QTY],
            $O_VAL[_O_MEDIAN_QTY],
            $O_VAL[_O_VARIANCE_QTY],
            $O_VAL[_O_STDDEV_QTY],
            $O_VAL[_O_MAX_QTY],
            $O_VAL[_O_MIN_QTY],
            $O_VAL[_O_TEST_4],
            $O_VAL[_O_RANGE_QTY],
            $O_VAL[_O_MODE_QTY],
            $O_VAL[_O_RANGE_QTY_2],
            $O_VAL[_O_RANGE_QTY_3],
            $O_VAL[_O_LOCATION]
        ;
        $previous_key__I_SALES_CODE = $key__I_SALES_CODE;
        @O_VAL = undef;
        %DISTINCT = undef;
        %AVERAGE = undef;
        %VALUES_ALL = undef;
        %MEDIAN = undef;
        %MEDIAN_COUNT = undef;
        %VARIANCE = undef;
        %STDDEV = undef;
        %RANGE = undef;
        %MODE = undef;
    }

    $O_VAL[_O_SALES_CODE] = $I_VAL[_I_SALES_CODE];
    $O_VAL[_O_NUM_PRODUCTS]++
        if (defined($I_VAL[_I_PRODUCT_CODE]) && ++$DISTINCT{_O_NUM_PRODUCTS}{qq{$I_VAL[_I_PRODUCT_CODE]}} == 1
);
    $AVERAGE{_O_AVG_COST_PRICE}{_SUM} += $I_VAL[_I_COST_PRICE];
    $AVERAGE{_O_AVG_COST_PRICE}{_COUNT}++;
    push(@{$VALUES_ALL{_O_VALUES_QTY}}, qq{$I_VAL[_I_SALES_QTY]});
    $O_VAL[_O_DISTINCT_QTY]++
        if (defined($I_VAL[_I_SALES_QTY]) && ++$DISTINCT{_O_DISTINCT_QTY}{qq{$I_VAL[_I_SALES_QTY]}} == 1);
    $MEDIAN_COUNT{_O_MEDIAN_QTY}++ if (++$MEDIAN{_O_MEDIAN_QTY}{qq{$I_VAL[_I_SALES_QTY]}} == 1);
    $VARIANCE{_O_VARIANCE_QTY}{_SUM} += $I_VAL[_I_SALES_QTY];
    $VARIANCE{_O_VARIANCE_QTY}{_SUM_SQUARES} += $I_VAL[_I_SALES_QTY] ** 2;
    $VARIANCE{_O_VARIANCE_QTY}{_COUNT}++;
    $STDDEV{_O_STDDEV_QTY}{_SUM} += $I_VAL[_I_SALES_QTY];
    $STDDEV{_O_STDDEV_QTY}{_SUM_SQUARES} += $I_VAL[_I_SALES_QTY] ** 2;
    $STDDEV{_O_STDDEV_QTY}{_COUNT}++;
    $O_VAL[_O_MAX_QTY] = $I_VAL[_I_SALES_QTY]
        if (!defined($O_VAL[_O_MAX_QTY]) || $I_VAL[_I_SALES_QTY] > $O_VAL[_O_MAX_QTY]);
    $O_VAL[_O_MIN_QTY] = $I_VAL[_I_SALES_QTY]
        if (!defined($O_VAL[_O_MIN_QTY]) || $I_VAL[_I_SALES_QTY] < $O_VAL[_O_MIN_QTY]);
    $RANGE{_O_RANGE_QTY}{_MIN} = $I_VAL[_I_SALES_QTY]
        if
        (
            !defined($RANGE{_O_RANGE_QTY}{_MIN})
```

```
                || $I_VAL[_I_SALES_QTY] < $RANGE{_O_RANGE_QTY}{_MIN}
            );

        $RANGE{_O_RANGE_QTY}{_MAX} = $I_VAL[_I_SALES_QTY]
            if
            (
                !defined($RANGE{_O_RANGE_QTY}{_MAX})
                || $I_VAL[_I_SALES_QTY] > $RANGE{_O_RANGE_QTY}{_MAX}
            );

        $MODE{_O_MODE_QTY}{qq{$I_VAL[_I_SALES_QTY]}}++;
        $O_VAL[_O_LOCATION] = $I_VAL[_I_LOCATION];
    }

    $O_VAL[_O_AVG_COST_PRICE] = ($AVERAGE{_O_AVG_COST_PRICE}{_COUNT} == 0 ? 0 : $AVERAGE{_O_AVG_COST_PRICE}{_SUM}
    / $AVERAGE{_O_AVG_COST_PRICE}{_COUNT});
    $O_VAL[_O_VALUES_QTY] = join(qq{,}, grep(length, @{$VALUES_ALL{_O_VALUES_QTY}}));
    $O_VAL[_O_MEDIAN_QTY] = $MEDIAN_COUNT{_O_MEDIAN_QTY} % 2 == 0 ? &{sub{($_[0] + $_[1]) / 2}}((( sort {$a <=> $b
    } keys %{$MEDIAN{_O_MEDIAN_QTY}} )[$MEDIAN_COUNT{_O_MEDIAN_QTY}/2-1, $MEDIAN_COUNT{_O_MEDIAN_QTY}/2])[0,1]) :
    (sort {$a <=> $b} keys %{$MEDIAN{_O_MEDIAN_QTY}} )[(($MEDIAN_COUNT{_O_MEDIAN_QTY}+1)/2)-1];
    $O_VAL[_O_VARIANCE_QTY] = ($VARIANCE{_O_VARIANCE_QTY}{_SUM_SQUARES} / ($VARIANCE{_O_VARIANCE_QTY}{_COUNT} == 0
     ? 1 : $VARIANCE{_O_VARIANCE_QTY}{_COUNT}))- (($VARIANCE{_O_VARIANCE_QTY}{_SUM} / $VARIANCE{_O_VARIANCE_QTY}{_
    COUNT}) ** 2);
    $O_VAL[_O_STDDEV_QTY] = sqrt(($STDDEV{_O_STDDEV_QTY}{_SUM_SQUARES} / ($STDDEV{_O_STDDEV_QTY}{_COUNT} == 0 ? 1
    : $STDDEV{_O_STDDEV_QTY}{_COUNT}))- (($STDDEV{_O_STDDEV_QTY}{_SUM} / $STDDEV{_O_STDDEV_QTY}{_COUNT}) ** 2));
    $O_VAL[_O_TEST_4] = $O_VAL[_O_MEDIAN_QTY];
    $O_VAL[_O_RANGE_QTY] = $RANGE{_O_RANGE_QTY}{_MAX} - $RANGE{_O_RANGE_QTY}{_MIN};
    $O_VAL[_O_MODE_QTY] = join(' ', &{sub{my @top; foreach my $k (sort { $MODE{_O_MODE_QTY}{$b} <=> $MODE{_O_MODE_
    QTY}{$a} } keys %{$MODE{_O_MODE_QTY}} ){ last if ($MODE{_O_MODE_QTY}{$k} != $MODE{_O_MODE_QTY}{$_[0]}); push(@
    top, $k);} @top; }}((sort { $MODE{_O_MODE_QTY}{$b} <=> $MODE{_O_MODE_QTY}{$a} } keys %{$MODE{_O_MODE_QTY}} )[0
    ]));
    $O_VAL[_O_RANGE_QTY_2] = $O_VAL[_O_RANGE_QTY] * 2;
    $O_VAL[_O_RANGE_QTY_3] = $O_VAL[_O_RANGE_QTY_2] * 3;
    print STDOUT
        $O_VAL[_O_SALES_CODE],
        $O_VAL[_O_NUM_PRODUCTS],
        $O_VAL[_O_AVG_COST_PRICE],
        $O_VAL[_O_VALUES_QTY],
        $O_VAL[_O_DISTINCT_QTY],
        $O_VAL[_O_MEDIAN_QTY],
        $O_VAL[_O_VARIANCE_QTY],
        $O_VAL[_O_STDDEV_QTY],
        $O_VAL[_O_MAX_QTY],
        $O_VAL[_O_MIN_QTY],
        $O_VAL[_O_TEST_4],
        $O_VAL[_O_RANGE_QTY],
        $O_VAL[_O_MODE_QTY],
        $O_VAL[_O_RANGE_QTY_2],
        $O_VAL[_O_RANGE_QTY_3],
        $O_VAL[_O_LOCATION]
    ;
    close(DATA);
    print STDERR '[examples/statistics_aggr_2.pql ' . localtime() . "] $_inprecs records.";
    my $benchmark_end = new Benchmark;
    my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
    print STDERR '[examples/statistics_aggr_2.pql ' . localtime() . "] Code statistics: @{[timestr($benchmark_time
    diff)]}";
    #-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    sub PrintHeader
    {
        local $\="\n";
        local $,="|";
        print STDOUT
            'SALES_CODE',
            'NUM_PRODUCTS',
            'AVG_COST_PRICE',
            'VALUES_QTY',
            'DISTINCT_QTY',
            'MEDIAN_QTY',
            'VARIANCE_QTY',
            'STDDEV_QTY',
            'MAX_QTY',
            'MIN_QTY',
            'TEST_4',
            'RANGE_QTY',
            'MODE_QTY',
            'RANGE_QTY_2',
            'RANGE_QTY_3',
            'LOCATION'
            ;
    }
```

7. ABOUT PEQUEL

This document was generated by Pequel.

https://sourceforge.net/projects/pequel/

COPYRIGHT

Copyright ©1999-2005, Mario Gaffiero. All Rights Reserved.
'Pequel' TM Copyright ©1999-2005, Mario Gaffiero. All Rights Reserved.

This program and all its component contents is copyrighted free software by Mario Gaffiero and is released under the GNU General Public License (GPL), Version 2, a copy of which may be found at http://www.opensource.org/licenses/gpl-license.html

Pequel is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Pequel is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Pequel; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA