



# output\_calc\_fields.pql

by *Pequel*

---

[sample@youraddress.com](mailto:sample@youraddress.com)

## Output Calculated Fields Example Script

2.2



# Table of Contents

## Output Calculated Fields Example Script

SCRIPT NAME	1
DESCRIPTION	1
1. PROCESS DETAILS	1
1.1 PRODUCT_CODE	1
Description	1
1.2 MIN_COST_PRICE	1
Description	1
1.3 MAX_COST_PRICE	1
Description	1
1.5 SALES_CODE_1	1
Description	1
Aggregation condition	1
1.6 SALES_CODE_2	1
Description	1
Aggregation condition	1
1.7 SALES_CODE_3	2
Description	2
Aggregation condition	2
1.8 SALES_CODE_4	2
Description	2
Aggregation condition	2
1.9 SALES_CODE_5	2
Description	2
Aggregation condition	2
1.10 COMMENT	2
Description	2
Derived Field Evaluation	2
1.11 COMMENT2	2
Description	2
Derived Field Evaluation	2
2. CONFIGURATION SETTINGS	3
2.1 pequeldoc	3
2.2 detail	3
2.3 script_name	3
2.4 header	3
2.5 optimize	3
2.6 doc_title	3
2.7 doc_email	3
2.8 doc_version	3
3. TABLES	4
4. TABLE INFORMATION SUMMARY	5
4.1 Table List Sorted By Table Name	5
5. OUTPUT_CALC_FIELDS.PQL	6
options	6
description	6
sort by	6
group by	6
input section	6
output section	6
6. PEQUEL GENERATED PROGRAM	7
7. ABOUT PEQUEL	9
COPYRIGHT	9



**SCRIPT NAME**

output\_calc\_fields.pql

**DESCRIPTION**

Demonstrate use of calculated output fields.

**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 **11** fields. Fields are delimited by the '|' character.

Input stream is **sorted** by the input field **PRODUCT\_CODE** (*string*).

Input records are **grouped** by the input field **PRODUCT\_CODE** (*string*).

**1.1 PRODUCT\_CODE**

Output Field

**Description**Set to input field **PRODUCT\_CODE****1.2 MIN\_COST\_PRICE**

Output Field

**Description****Min** aggregation on input field **COST\_PRICE**.**1.3 MAX\_COST\_PRICE**

Output Field

**Description****Max** aggregation on input field **COST\_PRICE**.**1.5 SALES\_CODE\_1**

Output Field

**Description****First** aggregation on input field **SALES\_CODE**.**Aggregation condition****\_DISTINCT\_SALES\_CODE** == 1;**1.6 SALES\_CODE\_2**

Output Field

**Description****First** aggregation on input field **SALES\_CODE**.**Aggregation condition**

\_DISTINCT\_SALES\_CODE == 2;

### 1.7 SALES\_CODE\_3

Output Field

**Description**

**First** aggregation on input field **SALES\_CODE**.

**Aggregation condition**

\_DISTINCT\_SALES\_CODE == 3;

### 1.8 SALES\_CODE\_4

Output Field

**Description**

**First** aggregation on input field **SALES\_CODE**.

**Aggregation condition**

\_DISTINCT\_SALES\_CODE == 4;

### 1.9 SALES\_CODE\_5

Output Field

**Description**

**First** aggregation on input field **SALES\_CODE**.

**Aggregation condition**

\_DISTINCT\_SALES\_CODE == 5;

### 1.10 COMMENT

Output Field

**Description**

Derived (calculated) field.

**Derived Field Evaluation**

### 1.11 COMMENT2

Output Field

**Description**

Derived (calculated) field.

**Derived Field Evaluation**

## 2. CONFIGURATION SETTINGS

### 2.1 *pequeldoc*

generate pod / pdf pequel script Reference Guide.: pdf

### 2.2 *detail*

Include Pequel Generated Program chapter in Pequeldoc: 1

### 2.3 *script\_name*

script filename: output\_calc\_fields.pql

### 2.4 *header*

write header record to output.: 1

### 2.5 *optimize*

optimize generated code.: 1

### 2.6 *doc\_title*

document title.: Output Calculated Fields Example Script

### 2.7 *doc\_email*

document email entry.: sample@youraddress.com

### 2.8 *doc\_version*

document version for pequel script.: 2.2

### 3. TABLES



## 4. TABLE INFORMATION SUMMARY

### 4.1 Table List Sorted By Table Name

## 5. OUTPUT\_CALC\_FIELDS.PQL

### *options*

```

pequeldoc(pdf)
detail(1)
script_name(output_calc_fields.pql)
header(1)
optimize(1)
doc_title(Output Calculated Fields Example Script)
doc_email(sample@youraddress.com)
doc_version(2.2)

```

### *description*

```

Demonstrate use of calculated output fields.

```

### *sort by*

```

PRODUCT_CODE string

```

### *group by*

```

PRODUCT_CODE string

```

### *input section*

```

PRODUCT_CODE
COST_PRICE
DESCRIPTION
SALES_CODE
SALES_PRICE
SALES_QTY
SALES_DATE
LOCATION

```

### *output section*

string	PRODUCT_CODE	PRODUCT_CODE
numeric	MIN_COST_PRICE	min COST_PRICE
numeric	MAX_COST_PRICE	max COST_PRICE
numeric	_DISTINCT_SALES_CODE	distinct SALES_CODE
string	SALES_CODE_1	first SALES_CODE where _DISTINCT_SALES_CODE == 1
string	SALES_CODE_2	first SALES_CODE where _DISTINCT_SALES_CODE == 2
string	SALES_CODE_3	first SALES_CODE where _DISTINCT_SALES_CODE == 3
string	SALES_CODE_4	first SALES_CODE where _DISTINCT_SALES_CODE == 4
string	SALES_CODE_5	first SALES_CODE where _DISTINCT_SALES_CODE == 5
string	COMMENT	= _DISTINCT_SALES_CODE > 5 ? "Product has more than five sales codes" : "Product has "
string	COMMENT2	= "Product has " . (_DISTINCT_SALES_CODE > 5 ? "more than five" : _DISTINCT_SALES_CODE) .

## 6. PEQUEL GENERATED PROGRAM

```
# vim: syntax=perl ts=4 sw=4
#-----
#Generated By: pequel Version 2.2-9, Build: Tuesday September 13 08:43:08 BST 2005
#           : https://sourceforge.net/projects/pequel/
#Script Name : output_calc_fields.pql
#Created On  : Tue Sep 13 10:31:35 2005
#For         :
#-----
#Options:
#pequeldoc(pdf) generate pod / pdf pequel script Reference Guide.
#detail(1) Include Pequel Generated Program chapter in Pequeldoc
#script_name(output_calc_fields.pql) script filename
#header(1) write header record to output.
#optimize(1) optimize generated code.
#doc_title(Output Calculated Fields Example Script) document title.
#doc_email(sample@youraddress.com) document email entry.
#doc_version(2.2) document version for pequel script.
#-----
use strict;
local $\\="\\n"; local $,="|";
print STDERR '[output_calc_fields.pql ' . localtime() . "] Init";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 7;
my @I_VAL;
my @O_VAL;
my %DISTINCT;
my $key__I_PRODUCT_CODE;
my $previous_key__I_PRODUCT_CODE = undef;
foreach my $f (1..11) { $O_VAL[$f] = undef; }
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_PRODUCT_CODE      => int 1;
use constant _O_MIN_COST_PRICE    => int 2;
use constant _O_MAX_COST_PRICE    => int 3;
use constant _O_DISTINCT_SALES_CODE => int 4;
use constant _O_SALES_CODE_1      => int 5;
use constant _O_SALES_CODE_2      => int 6;
use constant _O_SALES_CODE_3      => int 7;
use constant _O_SALES_CODE_4      => int 8;
use constant _O_SALES_CODE_5      => int 9;
use constant _O_COMMENT           => int 10;
use constant _O_COMMENT2          => int 11;
open(DATA, q{cat - | sort -t'|' -y -k 1,1 |}) || die "Cannot open input: $!";
&PrintHeader();
print STDERR '[output_calc_fields.pql ' . localtime() . "] Start";
use Benchmark;
my $benchmark_start = new Benchmark;
while (<DATA>)
{
    print STDERR '[output_calc_fields.pql ' . localtime() . "] $. records." if ($. % VERBOSE == 0);
    chomp;
    @I_VAL = split("[|]", $_);
    $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
    if (!defined($previous_key__I_PRODUCT_CODE))
    {
        $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
    }

    elsif ($previous_key__I_PRODUCT_CODE ne $key__I_PRODUCT_CODE)
    {
        $O_VAL[_O_COMMENT] = $O_VAL[_O_DISTINCT_SALES_CODE] > 5 ? "Product has more than five sales codes" :
        "Product has " . $O_VAL[_O_DISTINCT_SALES_CODE] . " sales codes";
        $O_VAL[_O_COMMENT2] = "Product has " . ($O_VAL[_O_DISTINCT_SALES_CODE] > 5 ? "more than five" : $O_VAL
        L[_O_DISTINCT_SALES_CODE]) . " sales codes";
        print
            $O_VAL[_O_PRODUCT_CODE],
            $O_VAL[_O_MIN_COST_PRICE],
            $O_VAL[_O_MAX_COST_PRICE],
            $O_VAL[_O_SALES_CODE_1],
            $O_VAL[_O_SALES_CODE_2],
            $O_VAL[_O_SALES_CODE_3],
            $O_VAL[_O_SALES_CODE_4],
            $O_VAL[_O_SALES_CODE_5],
            $O_VAL[_O_COMMENT],
```

```

        $O_VAL[_O_COMMENT2]
    ;
    $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
    @O_VAL = undef;
    %DISTINCT = undef;
}

$O_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
$O_VAL[_O_MIN_COST_PRICE] = $I_VAL[_I_COST_PRICE]
    if (!defined($O_VAL[_O_MIN_COST_PRICE]) || $I_VAL[_I_COST_PRICE] < $O_VAL[_O_MIN_COST_PRICE]);
$O_VAL[_O_MAX_COST_PRICE] = $I_VAL[_I_COST_PRICE]
    if (!defined($O_VAL[_O_MAX_COST_PRICE]) || $I_VAL[_I_COST_PRICE] > $O_VAL[_O_MAX_COST_PRICE]);
$O_VAL[_O_DISTINCT_SALES_CODE]++ if (defined($I_VAL[_I_SALES_CODE]) && ++$DISTINCT{__O_DISTINCT_SALES_COD
E}{qq{$I_VAL[_I_SALES_CODE]}} == 1);

if ($O_VAL[_O_DISTINCT_SALES_CODE] == 1) {
    $O_VAL[_O_SALES_CODE_1] = $I_VAL[_I_SALES_CODE] if (!defined($O_VAL[_O_SALES_CODE_1]));
}
elseif ($O_VAL[_O_DISTINCT_SALES_CODE] == 2) {
    $O_VAL[_O_SALES_CODE_2] = $I_VAL[_I_SALES_CODE] if (!defined($O_VAL[_O_SALES_CODE_2]));
}
elseif ($O_VAL[_O_DISTINCT_SALES_CODE] == 3) {
    $O_VAL[_O_SALES_CODE_3] = $I_VAL[_I_SALES_CODE] if (!defined($O_VAL[_O_SALES_CODE_3]));
}
elseif ($O_VAL[_O_DISTINCT_SALES_CODE] == 4) {
    $O_VAL[_O_SALES_CODE_4] = $I_VAL[_I_SALES_CODE] if (!defined($O_VAL[_O_SALES_CODE_4]));
}
elseif ($O_VAL[_O_DISTINCT_SALES_CODE] == 5) {
    $O_VAL[_O_SALES_CODE_5] = $I_VAL[_I_SALES_CODE] if (!defined($O_VAL[_O_SALES_CODE_5]));
}
}

$O_VAL[_O_COMMENT] = $O_VAL[_O_DISTINCT_SALES_CODE] > 5 ? "Product has more than five sales codes" : "Product
has " . $O_VAL[_O_DISTINCT_SALES_CODE] . " sales codes";
$O_VAL[_O_COMMENT2] = "Product has " . ($O_VAL[_O_DISTINCT_SALES_CODE] > 5 ? "more than five" : $O_VAL[_O_DI
STINCT_SALES_CODE]) . " sales codes";
print
    $O_VAL[_O_PRODUCT_CODE],
    $O_VAL[_O_MIN_COST_PRICE],
    $O_VAL[_O_MAX_COST_PRICE],
    $O_VAL[_O_SALES_CODE_1],
    $O_VAL[_O_SALES_CODE_2],
    $O_VAL[_O_SALES_CODE_3],
    $O_VAL[_O_SALES_CODE_4],
    $O_VAL[_O_SALES_CODE_5],
    $O_VAL[_O_COMMENT],
    $O_VAL[_O_COMMENT2]
;
print STDERR '[output_calc_fields.pql ' . localtime() . "] $. records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR '[output_calc_fields.pql ' . localtime() . "] Code statistics: @{{timestr($benchmark_timediff)}}"
;
#-+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
sub PrintHeader
{
    local $\="\n";
    local $,="|";
    print
        'PRODUCT_CODE',
        'MIN_COST_PRICE',
        'MAX_COST_PRICE',
        'SALES_CODE_1',
        'SALES_CODE_2',
        'SALES_CODE_3',
        'SALES_CODE_4',
        'SALES_CODE_5',
        'COMMENT',
        'COMMENT2'
    ;
}

```

## 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

