examples/divert_record.pql by *Pequel*

sample@youraddress.com

Divert Record Example Script

2.3

Table of Contents
Divert Record Example Script

SCRIPT NAME

examples/divert_record.pql

DESCRIPTION

1. PROCESS DETAILS

Input records are read from chain_pequel_pt1.pql. The input record contains ***3*** fields. Fields are delimited by the ‘**|**’ character.

Output records are written to standard output. The output record contains ***4*** fields. Fields are delimited by the ‘**|**’ character.

1.1 *CATEGORY*

Output Field

Description

Set to input field ***CATEGORY***

Derived Input Field Evaluation

```
=> 'HIGH'
```

1.2 *LOCATION*

Output Field

Description

Set to input field ***LOCATION***

1.3 *PRODUCT_CODE*

Output Field

Description

Set to input field ***PRODUCT_CODE***

1.4 *SALES_TOTAL*

Output Field

Description

Set to input field ***SALES_TOTAL***

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/divert_record.pql

2.5 ***input_file***
input data filename: chain_pequel_pt1.pql

2.6 ***optimize***
optimize generated code.: 1

2.7 ***doc_title***
document title.: Divert Record Example Script

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

2.9 ***doc_version***
document version for pequel script.: 2.3

3. TABLES

4. TABLE INFORMATION SUMMARY

4.1 Table List Sorted By Table Name

5. EXAMPLES/DIVERT_RECORD.PQL

options

```
prefix(examples)
pequeldoc(pdf)
detail(1)
script_name(examples/divert_record.pql)
input_file(chain_pequel_pt1.pql)
optimize(1)
doc_title(Divert Record Example Script)
doc_email(sample@youraddress.com)
doc_version(2.3)
```

input section

```
LOCATION
PRODUCT_CODE
SALES_TOTAL
CATEGORY => 'HIGH'
```

divert record(diverted_record_low.pql)

```
SALES_TOTAL <= 100000
```

divert record(diverted_record_med.pql)

```
SALES_TOTAL > 100000 && SALES_TOTAL <= 200000
```

output section

```
string    CATEGORY     CATEGORY
string    LOCATION     LOCATION
string    PRODUCT_CODE PRODUCT_CODE
decimal   SALES_TOTAL  SALES_TOTAL
```

sort output

```
SALES_TOTAL numeric
```

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 : divert_record.pql
#Created On  : Wed Nov 16 14:02:16 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/divert_record.pql) script filename
#input_file(chain_pequel_pt1.pql) input data filename
#optimize(1) optimize generated code.
#doc_title(Divert Record Example Script) document title.
#doc_email(sample@youraddress.com) document email entry.
#doc_version(2.3) document version for pequel script.
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
use strict;
use Fcntl ':flock';
use constant _I_LOCATION        => int    0;
use constant _I_PRODUCT_CODE    => int    1;
use constant _I_SALES_TOTAL     => int    2;
use constant _I_CATEGORY        => int    3;
use constant _O_CATEGORY        => int    1;
use constant _O_LOCATION        => int    2;
use constant _O_PRODUCT_CODE    => int    3;
use constant _O_SALES_TOTAL     => int    4;
local $\="\n";
local $,="|";
print STDERR '[examples/divert_record.pql ' . localtime() . "] Init";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 3;
my @I_VAL;
my @O_VAL;
my $_inprecs=0;
foreach my $f (1..4) { $O_VAL[$f] = undef; }
if (open(READ_CHAIN_PEQUEL_PT1, '-|') == 0) # Fork -- read from child
{
    &p_read_chain_pequel_pt1::read_chain_pequel_pt1;
    exit(0);
}

open(STDOUT, '|-', q{sort  -t'|' -y -k 4n,4n 2>/dev/null});
if (open(DIVERT_INPUT_DIVERTED_RECORD_LOW, '|-') == 0) # Fork -- write to child
{
    &p_divert_input_diverted_record_low::divert_input_diverted_record_low;
    exit(0);
}

if (open(DIVERT_INPUT_DIVERTED_RECORD_MED, '|-') == 0) # Fork -- write to child
{
    &p_divert_input_diverted_record_med::divert_input_diverted_record_med;
    exit(0);
}

print STDERR '[examples/divert_record.pql ' . localtime() . "] Start";
use Benchmark;
my $benchmark_start = new Benchmark;
while (<READ_CHAIN_PEQUEL_PT1>)
{
    ++$_inprecs;
    print STDERR '[examples/divert_record.pql ' . localtime() . "] $_inprecs records." if ($_inprecs % VERBOSE
 == 0);
    chomp;
    @I_VAL = split("[|]", $_);
    if (($I_VAL[_I_SALES_TOTAL] <= 100000))
    {
        print DIVERT_INPUT_DIVERTED_RECORD_LOW $_;
        next;
    }

    if (($I_VAL[_I_SALES_TOTAL] > 100000 && $I_VAL[_I_SALES_TOTAL] <= 200000))
    {
        print DIVERT_INPUT_DIVERTED_RECORD_MED $_;
        next;
    }

    $I_VAL[_I_CATEGORY] = 'HIGH';
    $O_VAL[_O_CATEGORY] = $I_VAL[_I_CATEGORY];
    $O_VAL[_O_LOCATION] = $I_VAL[_I_LOCATION];
    $O_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
    $O_VAL[_O_SALES_TOTAL] = $I_VAL[_I_SALES_TOTAL];
    flock(STDOUT, LOCK_EX);
    print STDOUT
        $O_VAL[_O_CATEGORY],
        $O_VAL[_O_LOCATION],
        $O_VAL[_O_PRODUCT_CODE],
        $O_VAL[_O_SALES_TOTAL]
    ;
    flock(STDOUT, LOCK_UN);
}

close(DIVERT_INPUT_DIVERTED_RECORD_MED);
close(DIVERT_INPUT_DIVERTED_RECORD_LOW);
close(STDOUT);
close(READ_CHAIN_PEQUEL_PT1);
print STDERR '[examples/divert_record.pql ' . localtime() . "] $_inprecs records.";
my $benchmark_end = new Benchmark;
my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
print STDERR '[examples/divert_record.pql ' . localtime() . "] Code statistics: @{[timestr($benchmark_timediff
)]}";
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
{
    package p_read_chain_pequel_pt1;
    sub read_chain_pequel_pt1
    {
#    !/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 : chain_pequel_pt1.pql
#    Created On  : Wed Nov 16 14:02:12 2005
#    Perl Version: /usr/bin/perl 5.6.1 on solaris
#    For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Options:
#        input_file(sample.data) input data filename
#        optimize(1) optimize generated code.
#        doc_title(Pequel Chaining Part-1 Example Script) document title.
#        doc_email(sample@youraddress.com) document email entry.
#        doc_version(2.3) 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 _I_SALES_TOTAL     => int    8;
        use constant _O_LOCATION        => int    1;
        use constant _O_PRODUCT_CODE    => int    2;
        use constant _O_SALES_TOTAL     => int    3;
        local $\="\n";
        local $,="|";
        print STDERR '[examples/chain_pequel_pt1.pql ' . localtime() . "] Init";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 8;
        my @I_VAL;
        my @O_VAL;
        my $_inprecs=0;
        my $key__I_LOCATION;
        my $previous_key__I_LOCATION = undef;
        my $key__I_PRODUCT_CODE;
        my $previous_key__I_PRODUCT_CODE = undef;
        foreach my $f (1..3) { $O_VAL[$f] = undef; }
#     Sort:LOCATION(asc:string) PRODUCT_CODE(asc:string)
        open(DATA, q{sort  -t'|' -y -k 8,8 -k 1,1 examples/sample.data 2>/dev/null |});
        open(STDOUT, '|-', q{sort  -t'|' -y -k 1,1 2>/dev/null});
        print STDERR '[examples/chain_pequel_pt1.pql ' . localtime() . "] Start";
        use Benchmark;
        my $benchmark_start = new Benchmark;
        while (<DATA>)
        {
            ++$_inprecs;
            print STDERR '[examples/chain_pequel_pt1.pql ' . localtime() . "] $_inprecs records." if ($_inprec
s % VERBOSE == 0);
            chomp;
            @I_VAL = split("[|]", $_);
            $key__I_LOCATION = $I_VAL[_I_LOCATION];
            $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
            if (!defined($previous_key__I_LOCATION) || !defined($previous_key__I_PRODUCT_CODE))
            {
                $previous_key__I_LOCATION = $key__I_LOCATION;
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
            }

            elsif ($previous_key__I_LOCATION ne $key__I_LOCATION || $previous_key__I_PRODUCT_CODE ne $key__I_P
RODUCT_CODE)
            {
                print STDOUT
                    $O_VAL[_O_LOCATION],
                    $O_VAL[_O_PRODUCT_CODE],
                    $O_VAL[_O_SALES_TOTAL]
                ;
                $previous_key__I_LOCATION = $key__I_LOCATION;
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
                @O_VAL = undef;
            }

            $O_VAL[_O_LOCATION] = $I_VAL[_I_LOCATION];
            $O_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
            $I_VAL[_I_SALES_TOTAL] = $I_VAL[_I_SALES_QTY] * $I_VAL[_I_SALES_PRICE];
            $O_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq ’’);
        }

        print STDOUT
            $O_VAL[_O_LOCATION],
            $O_VAL[_O_PRODUCT_CODE],
            $O_VAL[_O_SALES_TOTAL]
        ;
        close(STDOUT);
        close(DATA);
        print STDERR ’[examples/chain_pequel_pt1.pql ’ . localtime() . "] $_inprecs records.";
        my $benchmark_end = new Benchmark;
        my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
        print STDERR ’[examples/chain_pequel_pt1.pql ’ . localtime() . "] Code statistics: @{[timestr($benchma
rk_timediff)]}";
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    }

}

{
    package p_divert_input_diverted_record_med;
    sub divert_input_diverted_record_med
    {
#    !/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 : diverted_record_med.pql
#    Created On  : Wed Nov 16 14:02:15 2005
#    Perl Version: /usr/bin/perl 5.6.1 on solaris
#    For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Options:
#        optimize(1) optimize generated code.
#        doc_title(Diverted Record Example Script) document title.
#        doc_email(sample@youraddress.com) document email entry.
#        doc_version(2.3) document version for pequel script.
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        use strict;
        use Fcntl ’:flock’;
        use constant _I_LOCATION        => int    0;
        use constant _I_PRODUCT_CODE    => int    1;
        use constant _I_SALES_TOTAL     => int    2;
        use constant _I_CATEGORY        => int    3;
        use constant _O_CATEGORY        => int    1;
        use constant _O_LOCATION        => int    2;
        use constant _O_PRODUCT_CODE    => int    3;
        use constant _O_SALES_TOTAL     => int    4;
        local $\="\n";
        local $,="|";
        print STDERR ’[examples/diverted_record_med.pql ’ . localtime() . "] Init";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 3;
        my @I_VAL;
        my @O_VAL;
        my $_inprecs=0;
        my $key__I_LOCATION;
        my $previous_key__I_LOCATION = undef;
        my $key__I_PRODUCT_CODE;
        my $previous_key__I_PRODUCT_CODE = undef;
        foreach my $f (1..4) { $O_VAL[$f] = undef; }
#     Sort:LOCATION(asc:string) PRODUCT_CODE(asc:string)
        open(DATA, q{cat  - | sort  -t'|' -y -k 1,1 -k 2,2 2>/dev/null |}) || die "Cannot open input: $!";
        print STDERR '[examples/diverted_record_med.pql ' . localtime() . "] Start";
        use Benchmark;
        my $benchmark_start = new Benchmark;
        while (<DATA>)
        {
            ++$_inprecs;
            print STDERR '[examples/diverted_record_med.pql ' . localtime() . "] $_inprecs records." if ($_inp
recs % VERBOSE == 0);
            chomp;
            @I_VAL = split("[|]", $_);
            $key__I_LOCATION = $I_VAL[_I_LOCATION];
            $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
            if (!defined($previous_key__I_LOCATION) || !defined($previous_key__I_PRODUCT_CODE))
            {
                $previous_key__I_LOCATION = $key__I_LOCATION;
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
            }

            elsif ($previous_key__I_LOCATION ne $key__I_LOCATION || $previous_key__I_PRODUCT_CODE ne $key__I_P
RODUCT_CODE)
            {
                flock(STDOUT, LOCK_EX);
                print STDOUT
                    $O_VAL[_O_CATEGORY],
                    $O_VAL[_O_LOCATION],
                    $O_VAL[_O_PRODUCT_CODE],
                    $O_VAL[_O_SALES_TOTAL]
                ;
                flock(STDOUT, LOCK_UN);
                $previous_key__I_LOCATION = $key__I_LOCATION;
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
                @O_VAL = undef;
            }

            $I_VAL[_I_CATEGORY] = 'MEDIUM';
            $O_VAL[_O_CATEGORY] = $I_VAL[_I_CATEGORY];
            $O_VAL[_O_LOCATION] = $I_VAL[_I_LOCATION];
            $O_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
            $O_VAL[_O_SALES_TOTAL] = $I_VAL[_I_SALES_TOTAL];
        }

        flock(STDOUT, LOCK_EX);
        print STDOUT
            $O_VAL[_O_CATEGORY],
            $O_VAL[_O_LOCATION],
            $O_VAL[_O_PRODUCT_CODE],
            $O_VAL[_O_SALES_TOTAL]
        ;
        flock(STDOUT, LOCK_UN);
        close(DATA);
        print STDERR '[examples/diverted_record_med.pql ' . localtime() . "] $_inprecs records.";
        my $benchmark_end = new Benchmark;
        my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
        print STDERR '[examples/diverted_record_med.pql ' . localtime() . "] Code statistics: @{[timestr($benc
hmark_timediff)]}";
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    }

}

{
    package p_divert_input_diverted_record_low;
    sub divert_input_diverted_record_low
    {
#    !/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 : diverted_record_low.pql
#    Created On  : Wed Nov 16 14:02:13 2005
#    Perl Version: /usr/bin/perl 5.6.1 on solaris
#    For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Options:
#        optimize(1) optimize generated code.
#        doc_title(Diverted Record Example Script) document title.
#        doc_email(sample@youraddress.com) document email entry.
#        doc_version(2.3) document version for pequel script.
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        use strict;
        use Fcntl ':flock';
        use constant _I_LOCATION        => int    0;
        use constant _I_PRODUCT_CODE    => int    1;
        use constant _I_SALES_TOTAL     => int    2;
        use constant _I_CATEGORY        => int    3;
        use constant _O_CATEGORY        => int    1;
        use constant _O_LOCATION        => int    2;
        use constant _O_PRODUCT_CODE    => int    3;
        use constant _O_SALES_TOTAL     => int    4;
        local $\="\n";
        local $,="|";
        print STDERR '[examples/diverted_record_low.pql ' . localtime() . "] Init";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 3;
        my @I_VAL;
        my @O_VAL;
        my $_inprecs=0;
        my $key__I_LOCATION;
        my $previous_key__I_LOCATION = undef;
        my $key__I_PRODUCT_CODE;
        my $previous_key__I_PRODUCT_CODE = undef;
        foreach my $f (1..4) { $O_VAL[$f] = undef; }
#      Sort:LOCATION(asc:string) PRODUCT_CODE(asc:string)
        open(DATA, q{cat  - | sort  -t'|' -y -k 1,1 -k 2,2 2>/dev/null |}) || die "Cannot open input: $!";
        print STDERR '[examples/diverted_record_low.pql ' . localtime() . "] Start";
        use Benchmark;
        my $benchmark_start = new Benchmark;
        while (<DATA>)
        {
            ++$_inprecs;
            print STDERR '[examples/diverted_record_low.pql ' . localtime() . "] $_inprecs records." if ($_inp
recs % VERBOSE == 0);
            chomp;
            @I_VAL = split("[|]", $_);
            $key__I_LOCATION = $I_VAL[_I_LOCATION];
            $key__I_PRODUCT_CODE = $I_VAL[_I_PRODUCT_CODE];
            if (!defined($previous_key__I_LOCATION) || !defined($previous_key__I_PRODUCT_CODE))
            {
                $previous_key__I_LOCATION = $key__I_LOCATION;
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
            }

            elsif ($previous_key__I_LOCATION ne $key__I_LOCATION || $previous_key__I_PRODUCT_CODE ne $key__I_P
RODUCT_CODE)
            {
                flock(STDOUT, LOCK_EX);
                print STDOUT
                    $O_VAL[_O_CATEGORY],
                    $O_VAL[_O_LOCATION],
                    $O_VAL[_O_PRODUCT_CODE],
                    $O_VAL[_O_SALES_TOTAL]
                ;
                flock(STDOUT, LOCK_UN);
                $previous_key__I_LOCATION = $key__I_LOCATION;
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
                @O_VAL = undef;
            }

            $I_VAL[_I_CATEGORY] = 'LOW';
            $O_VAL[_O_CATEGORY] = $I_VAL[_I_CATEGORY];
            $O_VAL[_O_LOCATION] = $I_VAL[_I_LOCATION];
            $O_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
            $O_VAL[_O_SALES_TOTAL] = $I_VAL[_I_SALES_TOTAL];
        }

        flock(STDOUT, LOCK_EX);
        print STDOUT
            $O_VAL[_O_CATEGORY],
            $O_VAL[_O_LOCATION],
            $O_VAL[_O_PRODUCT_CODE],
            $O_VAL[_O_SALES_TOTAL]
        ;
        flock(STDOUT, LOCK_UN);
        close(DATA);
        print STDERR '[examples/diverted_record_low.pql ' . localtime() . "] $_inprecs records.";
        my $benchmark_end = new Benchmark;
        my $benchmark_timediff = timediff($benchmark_start, $benchmark_end);
        print STDERR '[examples/diverted_record_low.pql ' . localtime() . "] Code statistics: @{[timestr($benc
hmark_timediff)]}";
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        }
    }
```

7. ABOUT PEQUEL

This document was generated by Pequel.

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