examples/all_sections.pql by *Pequel*

sample@youraddress.com

All Section Types Example Script

2.4

Table of Contents
All Section Types Example Script

SCRIPT NAME

examples/all_sections.pql

DESCRIPTION

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

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

Input records are eliminated (**filtered**) unless ***PRODUCT_CODE !~ /^Z/***.

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

Input records are eliminated (**rejected**) if ***SALES_QTY == 0***. Rejected input records are written to the file examples/all_sections.pql.reject.

1.1 *PRODUCT_CODE*

Output Field

Description
Set to input field ***PRODUCT_CODE***

1.2 *RECORD_COUNT*

Output Field

Description
Count aggregation.

1.3 *SALES_QTY_SAMPLE1*

Output Field

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

Aggregation condition
exists %SAMPLE1(PRODUCT_CODE);

1.4 *SALES_QTY_SAMPLE2*

Output Field

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

Aggregation condition
exists %SAMPLE2(PRODUCT_CODE);

1.5 *S1_DESCRIPTION*
Output Field

Description
Set to input field ***S1_DESCRIPTION***

Derived Input Field Evaluation

```
=> %SAMPLE1(PRODUCT_CODE)->DESCRIPTION
```

1.6 *S1_LOCATION*
Output Field

Description
Set to input field ***S1_LOCATION***

Derived Input Field Evaluation

```
=> %SAMPLE1(PRODUCT_CODE)->LOCATION
```

1.7 *S2_DESCRIPTION*
Output Field

Description
Set to input field ***S2_DESCRIPTION***

Derived Input Field Evaluation

```
=> %SAMPLE2(PRODUCT_CODE)->DESCRIPTION
```

1.8 *S2_LOCATION*
Output Field

Description
Set to input field ***S2_LOCATION***

Derived Input Field Evaluation

```
=> %SAMPLE2(PRODUCT_CODE)->LOCATION
```

1.9 *PRODUCT_SALES_TOTAL*
Output Field

Description
Set to input field ***SALESBYPROD***

Derived Input Field Evaluation

```
=> %TSALESBYPROD(PRODUCT_CODE)->SALES_TOTAL
```

1.10 *LOCATION_SALES_TOTAL*
Output Field

Description
Set to input field ***SALESBYLOC***

Derived Input Field Evaluation

```
=> %TSALESBYLOC(LOCATION)->SALES_TOTAL
```

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 *noverbose*
do not progress counter: 1

2.4 *prefix*
directory pathname prefix.: examples

2.5 *script_name*
script filename: examples/all_sections.pql

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

2.7 *optimize*
optimize generated code.: 1

2.8 *doc_title*
document title.: All Section Types Example Script

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

2.10 *doc_version*
document version for pequel script.: 2.4

3. TABLES

3.1 *SAMPLE1*

Table Type: ***local***

Data

L103BJG04 — Toshiba 4000 IT P4-1800/1GB/60GB WA
A100AIX09 — Compaq 9000 GR P4-1700/256/40GB WA
B111KYK01 — Dell 1000 FR P4-1700/128/40GB PER
E100QTG07 — Fujitsu 7000 SP P4-1700/512/10GB NT
K113JAD05 — Fujitsu 5000 IT P3-1200/512/10GB PER
J115JBW09 — Compaq 9000 IT P3-1200/128/40GB SYD
J109NYP03 — HP 3000 IT P3-880/128/10GB MEL
A106UIH04 — Toshiba 4000 GR P4-1700/256/40GB ALIC
H107VAE06 — Toshiba 6000 FR P3-880/512/20GB WA
F104ICW08 — Compaq 8000 SP P4-1700/128/60GB PER
C103WEO02 — Cannon 2000 FR P4-1600/128/60GB WA
I108THJ06 — Dell 6000 GR P3-880/128/40GB VIC
D105BWE02 — IBM 2000 IT P4-1700/1GB/60GB PER
G111FOI06 — Toshiba 6000 FR P4-1900/512/60GB NT
I111AGN09 — Toshiba 9000 GR P4-1700/256/10GB PER
J102MLC05 — Fujitsu 5000 IT P3-1200/1GB/60GB VIC
G113WVH04 — Compaq 4000 SP P4-1800/256/20GB NT
I109JTE07 — IBM 7000 GR P3-1200/512/40GB MEL
C119GHQ10 — Dell 10000 FR P4-1700/1GB/30GB SYD
I115YVQ02 — Cannon 2000 EN P4-2000/256/10GB NSW
F105RTJ10 — Dell 10000 FR P3-900/512/20GB WA
A109IWD09 — Compaq 9000 IT P4-1700/128/20GB QLD
E119HQG01 — Dell 1000 GR P4-2000/1GB/40GB NT
A112HHM10 — Cannon 10000 FR P3-880/256/30GB SYD
K112WIS07 — Dell 7000 IT P3-1200/256/20GB PER
J112YXH07 — IBM 7000 EN P3-1400/256/40GB VIC
I105RHR09 — IBM 9000 FR P3-1200/512/40GB NT
L116RWV08 — Philips 8000 SP P3-900/128/10GB NSW
D117WMU02 — HP 2000 GR P4-1800/1GB/20GB QLD
C119HJM01 — Philips 1000 IT P3-1400/512/40GB NSW
L118PFA09 — Philips 9000 IT P4-1800/128/30GB SYD
E112SJD07 — IBM 7000 GR P3-1200/1GB/20GB SYD
F102EUR03 — Cannon 3000 EN P4-2000/512/30GB MEL
B117DAR07 — Cannon 7000 SP P4-1800/128/40GB ALIC
G103TKH08 — Fujitsu 8000 SP P4-1700/128/60GB ALIC
G106VOK04 — Fujitsu 4000 SP P3-900/512/40GB NT
F117WIP08 — IBM 8000 IT P3-900/1GB/10GB MEL
L105HMB07 — Philips 7000 FR P4-1600/1GB/10GB MEL
H113KDM07 — Compaq 7000 EN P3-880/512/40GB NT
C114ERT05 — IBM 5000 IT P4-1800/1GB/30GB VIC
H106LAF10 — Dell 10000 GR P4-2000/1GB/40GB SA
E100JMA04 — Cannon 4000 FR P3-1200/512/10GB VIC
E104HDH01 — Compaq 1000 EN P3-1200/256/20GB QLD
A109AYU10 — IBM 10000 FR P4-1700/512/10GB MEL
K111HOR02 — Cannon 2000 EN P4-1700/128/20GB NT
J112XUI05 — Dell 5000 EN P3-880/512/30GB PER
J117YTJ03 — IBM 3000 EN P4-1900/128/20GB VIC
D113QFU10 — Compaq 10000 SP P4-1900/1GB/30GB WA
K106NSX06 — Fujitsu 6000 IT P3-900/256/20GB NT
E108UFJ05 — Compaq 5000 SP P3-880/128/30GB VIC

3.2 *SAMPLE2*

Table Type: ***external***
Data Source Filename: ***sample.data***
Key Field Number: ***1***

3.2.1 *DESCRIPTION = 3*
3.2.2 *LOCATION = 8*

3.3 *LOC_DESCRIPT*

Table Type: ***local***

Data

NSW	New South Wales
WA	Western Australia
SYD	Sydney
MEL	Melbourne
SA	South Australia
NT	Northern Territory
QLD	Queensland
VIC	Victoria
PER	Perth
ALIC	Alice Springs

3.4 *TSALESBYLOC*

Table Type: ***external***
Data Source Filename: ***examples/sales_ttl_by_loc.pql***
Key Field Number: ***1***

3.4.1 *SALES_TOTAL = 2*
3.4.2 *TOP_PRODUCT = 3*

3.5 *TSALESBYPROD*

Table Type: ***external***
Data Source Filename: ***examples/sales_ttl_by_prod.pql***
Key Field Number: ***1***

3.5.1 *SALES_TOTAL = 2*

4. TABLE INFORMATION SUMMARY

4.1 Table List Sorted By Table Name

LOC_DESCRIPT — ***3*** (*local*)
SAMPLE1 — ***1*** (*local*)
SAMPLE2 — ***2*** (*external*)
TSALESBYLOC — ***4*** (*external*)
TSALESBYPROD — ***5*** (*external*)

5. EXAMPLES/ALL_SECTIONS.PQL

options

```
pequeldoc(pdf)
detail(1)
noverbose(1)
prefix(examples)
script_name(examples/all_sections.pql)
header(1)
optimize(1)
doc_title(All Section Types Example Script)
doc_email(sample@youraddress.com)
doc_version(2.4)
```

init table

```
LOC_DESCRIPT NSW New South Wales
LOC_DESCRIPT WA Western Australia
LOC_DESCRIPT SYD Sydney
LOC_DESCRIPT MEL Melbourne
LOC_DESCRIPT SA South Australia
LOC_DESCRIPT NT Northern Territory
LOC_DESCRIPT QLD Queensland
LOC_DESCRIPT VIC Victoria
LOC_DESCRIPT PER Perth
LOC_DESCRIPT ALIC Alice Springs

SAMPLE1 L103BJG04 Toshiba 4000 IT P4-1800/1GB/60GB WA
SAMPLE1 A100AIX09 Compaq 9000 GR P4-1700/256/40GB WA
SAMPLE1 B111KYK01 Dell 1000 FR P4-1700/128/40GB PER
SAMPLE1 E100QTG07 Fujitsu 7000 SP P4-1700/512/10GB NT
SAMPLE1 K113JAD05 Fujitsu 5000 IT P3-1200/512/10GB PER
SAMPLE1 J115JBW09 Compaq 9000 IT P3-1200/128/40GB SYD
SAMPLE1 J109NYP03 HP 3000 IT P3-880/128/10GB MEL
SAMPLE1 A106UIH04 Toshiba 4000 GR P4-1700/256/40GB ALIC
SAMPLE1 H107VAE06 Toshiba 6000 FR P3-880/512/20GB WA
SAMPLE1 F104ICW08 Compaq 8000 SP P4-1700/128/60GB PER
SAMPLE1 C103WEO02 Cannon 2000 FR P4-1600/128/60GB WA
SAMPLE1 I108THJ06 Dell 6000 GR P3-880/128/40GB VIC
SAMPLE1 D105BWE02 IBM 2000 IT P4-1700/1GB/60GB PER
SAMPLE1 G111FOI06 Toshiba 6000 FR P4-1900/512/60GB NT
SAMPLE1 I111AGN09 Toshiba 9000 GR P4-1700/256/10GB PER
SAMPLE1 J102MLC05 Fujitsu 5000 IT P3-1200/1GB/60GB VIC
SAMPLE1 G113WVH04 Compaq 4000 SP P4-1800/256/20GB NT
SAMPLE1 I109JTE07 IBM 7000 GR P3-1200/512/40GB MEL
SAMPLE1 C119GHQ10 Dell 10000 FR P4-1700/1GB/30GB SYD
SAMPLE1 I115YVQ02 Cannon 2000 EN P4-2000/256/10GB NSW
SAMPLE1 F105RTJ10 Dell 10000 FR P3-900/512/20GB WA
SAMPLE1 A109IWD09 Compaq 9000 IT P4-1700/128/20GB QLD
SAMPLE1 E119HQG01 Dell 1000 GR P4-2000/1GB/40GB NT
SAMPLE1 A112HHM10 Cannon 10000 FR P3-880/256/30GB SYD
SAMPLE1 K112WIS07 Dell 7000 IT P3-1200/256/20GB PER
SAMPLE1 J112YXH07 IBM 7000 EN P3-1400/256/40GB VIC
SAMPLE1 I105RHR09 IBM 9000 FR P3-1200/512/40GB NT
SAMPLE1 L116RWV08 Philips 8000 SP P3-900/128/10GB NSW
SAMPLE1 D117WMU02 HP 2000 GR P4-1800/1GB/20GB QLD
SAMPLE1 C119HJM01 Philips 1000 IT P3-1400/512/40GB NSW
SAMPLE1 L118PFA09 Philips 9000 IT P4-1800/128/30GB SYD
SAMPLE1 E112SJD07 IBM 7000 GR P3-1200/1GB/20GB SYD
SAMPLE1 F102EUR03 Cannon 3000 EN P4-2000/512/30GB MEL
SAMPLE1 B117DAR07 Cannon 7000 SP P4-1800/128/40GB ALIC
SAMPLE1 G103TKH08 Fujitsu 8000 SP P4-1700/128/60GB ALIC
SAMPLE1 G106VOK04 Fujitsu 4000 SP P3-900/512/40GB NT
SAMPLE1 F117WIP08 IBM 8000 IT P3-900/1GB/10GB MEL
SAMPLE1 L105HMB07 Philips 7000 FR P4-1600/1GB/10GB MEL
SAMPLE1 H113KDM07 Compaq 7000 EN P3-880/512/40GB NT
SAMPLE1 C114ERT05 IBM 5000 IT P4-1800/1GB/30GB VIC
SAMPLE1 H106LAF10 Dell 10000 GR P4-2000/1GB/40GB SA
SAMPLE1 E100JMA04 Cannon 4000 FR P3-1200/512/10GB VIC
SAMPLE1 E104HDH01 Compaq 1000 EN P3-1200/256/20GB QLD
SAMPLE1 A109AYU10 IBM 10000 FR P4-1700/512/10GB MEL
SAMPLE1 K111HOR02 Cannon 2000 EN P4-1700/128/20GB NT
SAMPLE1 J112XUI05 Dell 5000 EN P3-880/512/30GB PER
SAMPLE1 J117YTJ03 IBM 3000 EN P4-1900/128/20GB VIC
SAMPLE1 D113QFU10 Compaq 10000 SP P4-1900/1GB/30GB WA
SAMPLE1 K106NSX06 Fujitsu 6000 IT P3-900/256/20GB NT
SAMPLE1 E108UFJ05 Compaq 5000 SP P3-880/128/30GB VIC
```

load table

```
SAMPLE1 /* Table Name */ \
    sample.data /* Data Source Filename */ \
    1 /* Key Column Number */ \
     \
    DESCRIPTION = 3 \
    LOCATION = 8

SAMPLE2 /* Table Name */ \
    sample.data /* Data Source Filename */ \
    1 /* Key Column Number */ \
     \
    DESCRIPTION = 3 \
    LOCATION = 8

TSALESBYLOC /* Table Name */ \
    examples/sales_ttl_by_loc.pql /* Data Source Filename */ \
    1 /* Key Column Number */ \
     \
    SALES_TOTAL = 2 \
    TOP_PRODUCT = 3

TSALESBYPROD /* Table Name */ \
    examples/sales_ttl_by_prod.pql /* Data Source Filename */ \
    1 /* Key Column Number */ \
     \
    SALES_TOTAL = 2
```

input section

```
PRODUCT_CODE
COST_PRICE
DESCRIPTION
SALES_CODE
SALES_PRICE
SALES_QTY
SALES_DATE
LOCATION
S1_DESCRIPTION => %SAMPLE1(PRODUCT_CODE)->DESCRIPTION

S1_LOCATION => %SAMPLE1(PRODUCT_CODE)->LOCATION

S2_DESCRIPTION => %SAMPLE2(PRODUCT_CODE)->DESCRIPTION

S2_LOCATION => %SAMPLE2(PRODUCT_CODE)->LOCATION

LDESCRIPT => %LOC_DESCRIPT(LOCATION)

SALESBYLOC => %TSALESBYLOC(LOCATION)->SALES_TOTAL

SALESBYPROD => %TSALESBYPROD(PRODUCT_CODE)->SALES_TOTAL
```

divert record(diverted_record_low.pql)

```
SALES_QTY <= 100000
```

copy record(pequel:copy_record_SA.pql)

```
LOCATION eq ’SA’
```

copy record(pequel:copy_output_combiner.pql)

```
SALES_QTY > 0
```

filter

```
PRODUCT_CODE !~ /^Z/
```

sort by

```
PRODUCT_CODE string
```

group by

```
PRODUCT_CODE string
```

reject

```
SALES_QTY == 0
```

field preprocess

```
PRODUCT_CODE => &uc(PRODUCT_CODE)
```

output section

```
string    PRODUCT_CODE         PRODUCT_CODE
numeric   RECORD_COUNT         count *
numeric   SALES_QTY_SAMPLE1    sum SALES_QTY where exists %SAMPLE1(PRODUCT_CODE)
numeric   SALES_QTY_SAMPLE2    sum SALES_QTY where exists %SAMPLE2(PRODUCT_CODE)
string    S1_DESCRIPTION       S1_DESCRIPTION
string    S1_LOCATION          S1_LOCATION
string    S2_DESCRIPTION       S2_DESCRIPTION
string    S2_LOCATION          S2_LOCATION
decimal   PRODUCT_SALES_TOTAL  SALESBYPROD
decimal   LOCATION_SALES_TOTAL SALESBYLOC
```

field postprocess

```
RECORD_COUNT => &sprintf("%06d",RECORD_COUNT)
```

sort output

```
S2_LOCATION string des
```

6. PEQUEL GENERATED PROGRAM

```
#!/usr/bin/perl
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
# vim: syntax=perl ts=4 sw=4
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#Generated By: pequel Version 2.4-4, Build: Tuesday November  1 08:45:13 GMT 2005
#            : http://sourceforge.net/projects/pequel/
#Script Name : all_sections.pql
#Created On  : Thu Nov  3 15:42:39 2005
#Perl Version: /usr/bin/perl 5.6.1 on solaris
#For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#Options:
#pequeldoc(pdf) generate pod / pdf pequel script Reference Guide.
#detail(1) Include Pequel Generated Program chapter in Pequeldoc
#noverbose(1) do not progress counter
#prefix(examples) directory pathname prefix.
#script_name(examples/all_sections.pql) script filename
#header(1) write header record to output.
#optimize(1) optimize generated code.
#doc_title(All Section Types Example Script) document title.
#doc_email(sample@youraddress.com) document email entry.
#doc_version(2.4) document version for pequel script.
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
use strict;
use Fcntl ':flock';
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_S1_DESCRIPTION          => int    8;
use constant _I_S1_LOCATION             => int    9;
use constant _I_S2_DESCRIPTION          => int   10;
use constant _I_S2_LOCATION             => int   11;
use constant _I_LDESCRIPT               => int   12;
use constant _I_SALESBYLOC              => int   13;
use constant _I_SALESBYPROD             => int   14;
use constant _O_PRODUCT_CODE            => int    1;
use constant _O_RECORD_COUNT            => int    2;
use constant _O_SALES_QTY_SAMPLE1       => int    3;
use constant _O_SALES_QTY_SAMPLE2       => int    4;
use constant _O_S1_DESCRIPTION          => int    5;
use constant _O_S1_LOCATION             => int    6;
use constant _O_S2_DESCRIPTION          => int    7;
use constant _O_S2_LOCATION             => int    8;
use constant _O_PRODUCT_SALES_TOTAL     => int    9;
use constant _O_LOCATION_SALES_TOTAL    => int   10;
use constant _T_LOC_DESCRIPT_FLD_1             => int    0;
use constant _T_SAMPLE1_FLD_DESCRIPTION        => int    0;
use constant _T_SAMPLE1_FLD_LOCATION           => int    1;
use constant _T_SAMPLE2_FLD_DESCRIPTION        => int    0;
use constant _T_SAMPLE2_FLD_LOCATION           => int    1;
use constant _T_TSALESBYLOC_FLD_SALES_TOTAL    => int    0;
use constant _T_TSALESBYLOC_FLD_TOP_PRODUCT    => int    1;
use constant _T_TSALESBYPROD_FLD_SALES_TOTAL   => int    0;
use constant _I_SAMPLE1_PRODUCT_CODE_FLD_KEY              => int   15;
use constant _I_SAMPLE1_PRODUCT_CODE_FLD_DESCRIPTION      => int   16;
use constant _I_SAMPLE1_PRODUCT_CODE_FLD_LOCATION         => int   17;
use constant _I_SAMPLE2_PRODUCT_CODE_FLD_KEY              => int   18;
use constant _I_SAMPLE2_PRODUCT_CODE_FLD_DESCRIPTION      => int   19;
use constant _I_SAMPLE2_PRODUCT_CODE_FLD_LOCATION         => int   20;
use constant _I_LOC_DESCRIPT_LOCATION_FLD_KEY             => int   21;
use constant _I_LOC_DESCRIPT_LOCATION_FLD_1               => int   22;
use constant _I_TSALESBYLOC_LOCATION_FLD_KEY              => int   23;
use constant _I_TSALESBYLOC_LOCATION_FLD_SALES_TOTAL      => int   24;
use constant _I_TSALESBYLOC_LOCATION_FLD_TOP_PRODUCT      => int   25;
use constant _I_TSALESBYPROD_PRODUCT_CODE_FLD_KEY         => int   26;
use constant _I_TSALESBYPROD_PRODUCT_CODE_FLD_SALES_TOTAL  => int   27;
local $\="\n";
local $,="|";
use constant VERBOSE => int 10000;
use constant LAST_ICELL => int 14;
my @I_VAL;
my @O_VAL;
my $key__I_PRODUCT_CODE;
my $previous_key__I_PRODUCT_CODE = undef;
foreach my $f (1..10) { $O_VAL[$f] = undef; }
```

```
open(REJECT, ">examples/all_sections.pql.reject");
my $_TABLE_LOC_DESCRIPT = &InitLookupLOC_DESCRIPT; # ref to %$LOC_DESCRIPT hash
my $_TABLE_SAMPLE1 = &InitLookupSAMPLE1; # ref to %$SAMPLE1 hash
my $_TABLE_SAMPLE2 = &LoadTableSAMPLE2; # ref to %$SAMPLE2 hash
my $_TABLE_TSALESBYLOC = &LoadTableTSALESBYLOC; # ref to %$TSALESBYLOC hash
my $_TABLE_TSALESBYPROD = &LoadTableTSALESBYPROD; # ref to %$TSALESBYPROD hash
# Sort:PRODUCT_CODE(asc:string)
open(DATA, q{cat  - | sort  -t’|’ -y -k 1,1 2>/dev/null |}) || die "Cannot open input: $!";
open(STDOUT, ’|-’, q{sort  -t’|’ -y -k 8r,8r 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(COPY_INPUT_COPY_RECORD_SA, ’|-’) == 0) # Fork -- write to child
{
    &p_copy_input_copy_record_sa::copy_input_copy_record_sa;
    exit(0);
}

if (open(COPY_OUTPUT_COPY_OUTPUT_COMBINER, ’|-’) == 0) # Fork -- write to child
{
    &p_copy_output_copy_output_combiner::copy_output_copy_output_combiner;
    exit(0);
}

&PrintHeader();
while (<DATA>)
{
    chomp;
    @I_VAL = split("[|]", $_);
    $I_VAL[_I_PRODUCT_CODE] = uc($I_VAL[_I_PRODUCT_CODE]);
    next unless ($I_VAL[_I_PRODUCT_CODE] !~ /^Z/);
    if (( $I_VAL[_I_SALES_QTY] == 0 ))
    {
        local $\="\n";
        print REJECT $_;
        next;
    }

    if (($I_VAL[_I_SALES_QTY] <= 100000))
    {
        print DIVERT_INPUT_DIVERTED_RECORD_LOW $_;
        next;
    }

    if (($I_VAL[_I_LOCATION] eq ’SA’))
    {
        print COPY_INPUT_COPY_RECORD_SA $_;
    }

    if (( $I_VAL[_I_PRODUCT_CODE] =~ /^A/ ))
    {
        print STDERR "Product code: $I_VAL[_I_PRODUCT_CODE]";
    }

    if (( $I_VAL[_I_PRODUCT_CODE] =~ /^[0-9]/ ))
    {
        print STDERR "Invalid Product Code: $I_VAL[_I_PRODUCT_CODE]";
        print STDERR "Process aborted at record " . int($.);
        last;
    }

    $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)
    {
        flock(STDOUT, LOCK_EX);
        print STDOUT
            $O_VAL[_O_PRODUCT_CODE],
            $O_VAL[_O_RECORD_COUNT],
            $O_VAL[_O_SALES_QTY_SAMPLE1],
            $O_VAL[_O_SALES_QTY_SAMPLE2],
            $O_VAL[_O_S1_DESCRIPTION],
            $O_VAL[_O_S1_LOCATION],
            $O_VAL[_O_S2_DESCRIPTION],
            $O_VAL[_O_S2_LOCATION],
            $O_VAL[_O_PRODUCT_SALES_TOTAL],
            $O_VAL[_O_LOCATION_SALES_TOTAL]
        ;
        flock(STDOUT, LOCK_UN);
        if (( $O_VAL[_O_RECORD_COUNT] < 15 ))
        {
            print STDERR "Product $O_VAL[_O_PRODUCT_CODE] contains less than 5 transactions -- $O_VAL[_O_RECOR
D_COUNT]";
        }

        if (( $O_VAL[_O_RECORD_COUNT] > 500 ))
        {
            print STDERR "Invalid transaction count for Product $O_VAL[_O_PRODUCT_CODE] > 500 transactions --
$O_VAL[_O_RECORD_COUNT]";
            print STDERR "Process aborted at record " . int($.);
            last;
        }

        if ($I_VAL[_I_SALES_QTY] > 0)
        {
            flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
            print COPY_OUTPUT_COPY_OUTPUT_COMBINER
                $O_VAL[_O_PRODUCT_CODE],
                $O_VAL[_O_RECORD_COUNT],
                $O_VAL[_O_SALES_QTY_SAMPLE1],
                $O_VAL[_O_SALES_QTY_SAMPLE2],
                $O_VAL[_O_S1_DESCRIPTION],
                $O_VAL[_O_S1_LOCATION],
                $O_VAL[_O_S2_DESCRIPTION],
                $O_VAL[_O_S2_LOCATION],
                $O_VAL[_O_PRODUCT_SALES_TOTAL],
                $O_VAL[_O_LOCATION_SALES_TOTAL]
            ;
            flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
        }

        $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
        @O_VAL = undef;
    }

    $O_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
    $O_VAL[_O_RECORD_COUNT]++;
    $I_VAL[_I_S1_DESCRIPTION] = ${$$_TABLE_SAMPLE1{qq{$I_VAL[_I_PRODUCT_CODE]}}}[_T_SAMPLE1_FLD_DESCRIPTION];
    $O_VAL[_O_S1_DESCRIPTION] = $I_VAL[_I_S1_DESCRIPTION];
    $I_VAL[_I_S1_LOCATION] = ${$$_TABLE_SAMPLE1{qq{$I_VAL[_I_PRODUCT_CODE]}}}[_T_SAMPLE1_FLD_LOCATION];
    $O_VAL[_O_S1_LOCATION] = $I_VAL[_I_S1_LOCATION];
    $I_VAL[_I_S2_DESCRIPTION] = ${$$_TABLE_SAMPLE2{qq{$I_VAL[_I_PRODUCT_CODE]}}}[_T_SAMPLE2_FLD_DESCRIPTION];
    $O_VAL[_O_S2_DESCRIPTION] = $I_VAL[_I_S2_DESCRIPTION];
    $I_VAL[_I_S2_LOCATION] = ${$$_TABLE_SAMPLE2{qq{$I_VAL[_I_PRODUCT_CODE]}}}[_T_SAMPLE2_FLD_LOCATION];
    $O_VAL[_O_S2_LOCATION] = $I_VAL[_I_S2_LOCATION];
    $I_VAL[_I_SALESBYPROD] = $$_TABLE_TSALESBYPROD{qq{$I_VAL[_I_PRODUCT_CODE]}};
    $O_VAL[_O_PRODUCT_SALES_TOTAL] = $I_VAL[_I_SALESBYPROD];
    $I_VAL[_I_SALESBYLOC] = ${$$_TABLE_TSALESBYLOC{qq{$I_VAL[_I_LOCATION]}}}[_T_TSALESBYLOC_FLD_SALES_TOTAL];
    $O_VAL[_O_LOCATION_SALES_TOTAL] = $I_VAL[_I_SALESBYLOC];

    if (exists $$_TABLE_SAMPLE1{qq{$I_VAL[_I_PRODUCT_CODE]}}) {
        $O_VAL[_O_SALES_QTY_SAMPLE1] += $I_VAL[_I_SALES_QTY] unless ($I_VAL[_I_SALES_QTY] eq ’’);
    }

    if (exists $$_TABLE_SAMPLE2{qq{$I_VAL[_I_PRODUCT_CODE]}}) {
        $O_VAL[_O_SALES_QTY_SAMPLE2] += $I_VAL[_I_SALES_QTY] unless ($I_VAL[_I_SALES_QTY] eq ’’);
    }
    $O_VAL[_O_RECORD_COUNT] = sprintf("%06d",$O_VAL[_O_RECORD_COUNT]);
}

flock(STDOUT, LOCK_EX);
print STDOUT
    $O_VAL[_O_PRODUCT_CODE],
    $O_VAL[_O_RECORD_COUNT],
    $O_VAL[_O_SALES_QTY_SAMPLE1],
    $O_VAL[_O_SALES_QTY_SAMPLE2],
    $O_VAL[_O_S1_DESCRIPTION],
    $O_VAL[_O_S1_LOCATION],
    $O_VAL[_O_S2_DESCRIPTION],
    $O_VAL[_O_S2_LOCATION],
    $O_VAL[_O_PRODUCT_SALES_TOTAL],
    $O_VAL[_O_LOCATION_SALES_TOTAL]
;
flock(STDOUT, LOCK_UN);
if (( $O_VAL[_O_RECORD_COUNT] < 15 ))
{
    print STDERR "Product $O_VAL[_O_PRODUCT_CODE] contains less than 5 transactions -- $O_VAL[_O_RECORD_COUNT]
";
}

if (( $O_VAL[_O_RECORD_COUNT] > 500 ))
{
    print STDERR "Invalid transaction count for Product $O_VAL[_O_PRODUCT_CODE] > 500 transactions -- $O_VAL[_
```

```
O_RECORD_COUNT]";
    print STDERR "Process aborted at record " . int($.);
    last;
}

if ($I_VAL[_I_SALES_QTY] > 0)
{
    flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_EX);
    print COPY_OUTPUT_COPY_OUTPUT_COMBINER
        $O_VAL[_O_PRODUCT_CODE],
        $O_VAL[_O_RECORD_COUNT],
        $O_VAL[_O_SALES_QTY_SAMPLE1],
        $O_VAL[_O_SALES_QTY_SAMPLE2],
        $O_VAL[_O_S1_DESCRIPTION],
        $O_VAL[_O_S1_LOCATION],
        $O_VAL[_O_S2_DESCRIPTION],
        $O_VAL[_O_S2_LOCATION],
        $O_VAL[_O_PRODUCT_SALES_TOTAL],
        $O_VAL[_O_LOCATION_SALES_TOTAL]
    ;
    flock(COPY_OUTPUT_COPY_OUTPUT_COMBINER, LOCK_UN);
}

close(COPY_OUTPUT_COPY_OUTPUT_COMBINER);
close(COPY_INPUT_COPY_RECORD_SA);
close(DIVERT_INPUT_DIVERTED_RECORD_LOW);
close(STDOUT);
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#++++++ Table LOC_DESCRIPT --> Type :ETL::Pequel::Type::Table::Local ++++++
sub InitLookupLOC_DESCRIPT
{
    my %_TABLE_LOC_DESCRIPT;
    %_TABLE_LOC_DESCRIPT =
    (
        'ALIC' => 'Alice Springs',
        'MEL' => 'Melbourne',
        'NSW' => 'New South Wales',
        'NT' => 'Northern Territory',
        'PER' => 'Perth',
        'QLD' => 'Queensland',
        'SA' => 'South Australia',
        'SYD' => 'Sydney',
        'VIC' => 'Victoria',
        'WA' => 'Western Australia'
    );
    return \%_TABLE_LOC_DESCRIPT;
}

#++++++ Table SAMPLE1 --> Type :ETL::Pequel::Type::Table::Local ++++++
sub InitLookupSAMPLE1
{
    my %_TABLE_SAMPLE1;
    %_TABLE_SAMPLE1 =
    (
        'A100AIX09' => ['Compaq 9000 GR P4-1700/256/40GB', 'WA'],
        'A106UIH04' => ['Toshiba 4000 GR P4-1700/256/40GB', 'ALIC'],
        'A109AYU10' => ['IBM 10000 FR P4-1700/512/10GB', 'MEL'],
        'A109IWD09' => ['Compaq 9000 IT P4-1700/128/20GB', 'QLD'],
        'A112HHM10' => ['Cannon 10000 FR P3-880/256/30GB', 'SYD'],
        'B111KYK01' => ['Dell 1000 FR P4-1700/128/40GB', 'PER'],
        'B117DAR07' => ['Cannon 7000 SP P4-1800/128/40GB', 'ALIC'],
        'C103WEO02' => ['Cannon 2000 FR P4-1600/128/60GB', 'WA'],
        'C114ERT05' => ['IBM 5000 IT P4-1800/1GB/30GB', 'VIC'],
        'C119GHQ10' => ['Dell 10000 FR P4-1700/1GB/30GB', 'SYD'],
        'C119HJM01' => ['Philips 1000 IT P3-1400/512/40GB', 'NSW'],
        'D105BWE02' => ['IBM 2000 IT P4-1700/1GB/60GB', 'PER'],
        'D113QFU10' => ['Compaq 10000 SP P4-1900/1GB/30GB', 'WA'],
        'D117WMU02' => ['HP 2000 GR P4-1800/1GB/20GB', 'QLD'],
        'E100JMA04' => ['Cannon 4000 FR P3-1200/512/10GB', 'VIC'],
        'E100QTG07' => ['Fujitsu 7000 SP P4-1700/512/10GB', 'NT'],
        'E104HDH01' => ['Compaq 1000 EN P3-1200/256/20GB', 'QLD'],
        'E108UFJ05' => ['Compaq 5000 SP P3-880/128/30GB', 'VIC'],
        'E112SJD07' => ['IBM 7000 GR P3-1200/1GB/20GB', 'SYD'],
        'E119HQG01' => ['Dell 1000 GR P4-2000/1GB/40GB', 'NT'],
        'F102EUR03' => ['Cannon 3000 EN P4-2000/512/30GB', 'MEL'],
        'F104ICW08' => ['Compaq 8000 SP P4-1700/128/60GB', 'PER'],
        'F105RTJ10' => ['Dell 10000 FR P3-900/512/20GB', 'WA'],
        'F117WIP08' => ['IBM 8000 IT P3-900/1GB/10GB', 'MEL'],
        'G103TKH08' => ['Fujitsu 8000 SP P4-1700/128/60GB', 'ALIC'],
        'G106VOK04' => ['Fujitsu 4000 SP P3-900/512/40GB', 'NT'],
        'G111FOI06' => ['Toshiba 6000 FR P4-1900/512/60GB', 'NT'],
        'G113WVH04' => ['Compaq 4000 SP P4-1800/256/20GB', 'NT'],
        'H106LAF10' => ['Dell 10000 GR P4-2000/1GB/40GB', 'SA'],
        'H107VAE06' => ['Toshiba 6000 FR P3-880/512/20GB', 'WA'],
        'H113KDM07' => ['Compaq 7000 EN P3-880/512/40GB', 'NT'],
        'I105RHR09' => ['IBM 9000 FR P3-1200/512/40GB', 'NT'],
        'I108THJ06' => ['Dell 6000 GR P3-880/128/40GB', 'VIC'],
        'I109JTE07' => ['IBM 7000 GR P3-1200/512/40GB', 'MEL'],
        'I111AGN09' => ['Toshiba 9000 GR P4-1700/256/10GB', 'PER'],
        'I115YVQ02' => ['Cannon 2000 EN P4-2000/256/10GB', 'NSW'],
        'J102MLC05' => ['Fujitsu 5000 IT P3-1200/1GB/60GB', 'VIC'],
        'J109NYP03' => ['HP 3000 IT P3-880/128/10GB', 'MEL'],
        'J112XUI05' => ['Dell 5000 EN P3-880/512/30GB', 'PER'],
        'J112YXH07' => ['IBM 7000 EN P3-1400/256/40GB', 'VIC'],
        'J115JBW09' => ['Compaq 9000 IT P3-1200/128/40GB', 'SYD'],
        'J117YTJ03' => ['IBM 3000 EN P4-1900/128/20GB', 'VIC'],
        'K106NSX06' => ['Fujitsu 6000 IT P3-900/256/20GB', 'NT'],
        'K111HOR02' => ['Cannon 2000 EN P4-1700/128/20GB', 'NT'],
        'K112WIS07' => ['Dell 7000 IT P3-1200/256/20GB', 'PER'],
        'K113JAD05' => ['Fujitsu 5000 IT P3-1200/512/10GB', 'PER'],
        'L103BJG04' => ['Toshiba 4000 IT P4-1800/1GB/60GB', 'WA'],
        'L105HMB07' => ['Philips 7000 FR P4-1600/1GB/10GB', 'MEL'],
        'L116RWV08' => ['Philips 8000 SP P3-900/128/10GB', 'NSW'],
        'L118PFA09' => ['Philips 9000 IT P4-1800/128/30GB', 'SYD']
    );
    return \%_TABLE_SAMPLE1;
}

#++++++ Table SAMPLE2 --> Type :ETL::Pequel::Type::Table::External ++++++
sub LoadTableSAMPLE2
{
    my %_TABLE_SAMPLE2;
    my $dsf = 'examples/sample.data';
    open(SAMPLE2, "sort  -u -t'|' -k 1 $dsf |") || die("Unable to open table source file $dsf");
    while (<SAMPLE2>)
    {
        chomp;
        my (@flds) = split("[|]", $_, -1);
        $_TABLE_SAMPLE2{$flds[0]} = [ @flds[ 2,7 ]];
    }

    close(SAMPLE2);
    return \%_TABLE_SAMPLE2;
}

#++++++ Table TSALESBYLOC --> Type :ETL::Pequel::Type::Table::External::Pequel ++++++
sub LoadTableTSALESBYLOC
{
    my %_TABLE_TSALESBYLOC;
    my $pid = open(TSALESBYLOC, '-|'); # Fork
    my $count=0;
    if ($pid) # Parent
    {
        while (<TSALESBYLOC>)
        {
            chomp;
            my (@flds) = split("[|]", $_, -1);
            $_TABLE_TSALESBYLOC{$flds[0]} = [ @flds[ 1,2 ]];
        }

        $count=$.;
        close(TSALESBYLOC);
    }

    else # Child
    {
        &p_LoadTableTSALESBYLOC::LoadTableTSALESBYLOC;
        exit(0);
    }

    close(TSALESBYLOC);
    return \%_TABLE_TSALESBYLOC;
}

{
    package p_LoadTableTSALESBYLOC;
    sub LoadTableTSALESBYLOC
    {
#    !/usr/bin/perl
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#     vim: syntax=perl ts=4 sw=4
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Generated By: pequel Version 2.4-4, Build: Tuesday November  1 08:45:13 GMT 2005
#                : http://sourceforge.net/projects/pequel/
#    Script Name : sales_ttl_by_loc.pql
#    Created On  : Thu Nov  3 15:42:29 2005
#    Perl Version: /usr/bin/perl 5.6.1 on solaris
#    For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Options:
```

```
#        input_file(sample.data) input data filename
#        header(1) write header record to output.
#        optimize(1) optimize generated code.
#        hash(1) Generate in memory. Input data can be unsorted.
#        doc_title(Pequel Table 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 _I_TOP_PRODUCT     => int    9;
        use constant _O_LOCATION        => int    1;
        use constant _O_SALES_TOTAL     => int    2;
        use constant _O_TOP_PRODUCT     => int    3;
        use constant _T_TTOPPRODBYLOC_FLD_PRODUCT_CODE   => int    0;
        use constant _I_TTOPPRODBYLOC_LOCATION_FLD_KEY           => int   10;
        use constant _I_TTOPPRODBYLOC_LOCATION_FLD_PRODUCT_CODE  => int   11;
        local $\="\n";
        local $,="|";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 9;
        my @I_VAL;
        my %O_VAL;
        my $key;
        my $_TABLE_TTOPPRODBYLOC = &LoadTableTTOPPRODBYLOC; # ref to %$TTOPPRODBYLOC hash
        open(DATA, q{examples/sample.data})|| die "Cannot open examples/sample.data: $!";
        &PrintHeader();
        while (<DATA>)
        {
            chomp;
            @I_VAL = split("[|]", $_);
            $key = ( $I_VAL[_I_LOCATION] );
            $O_VAL{$key}{_O_LOCATION} = $I_VAL[_I_LOCATION];
            $I_VAL[_I_SALES_TOTAL] = $I_VAL[_I_SALES_QTY] * $I_VAL[_I_SALES_PRICE];
            $O_VAL{$key}{_O_SALES_TOTAL} += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
            $I_VAL[_I_TOP_PRODUCT] = $$_TABLE_TTOPPRODBYLOC{qq{$I_VAL[_I_LOCATION]}};
            $O_VAL{$key}{_O_TOP_PRODUCT} = $I_VAL[_I_TOP_PRODUCT];
        }

        foreach $key (sort  keys %O_VAL)
        {
            print STDOUT
                $O_VAL{$key}{_O_LOCATION},
                $O_VAL{$key}{_O_SALES_TOTAL},
                $O_VAL{$key}{_O_TOP_PRODUCT}
            ;
        }

#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    ++++++ Table TTOPPRODBYLOC --> Type :ETL::Pequel::Type::Table::External::Pequel ++++++
        sub LoadTableTTOPPRODBYLOC
        {
            my %_TABLE_TTOPPRODBYLOC;
            my $pid = open(TTOPPRODBYLOC, '-|'); # Fork
            my $count=0;
            if ($pid) # Parent
            {
                while (<TTOPPRODBYLOC>)
                {
                    chomp;
                    my (@flds) = split("[|]", $_, -1);
                    $_TABLE_TTOPPRODBYLOC{$flds[0]} = $flds[ 1 ];
                }

                $count=$.;
                close(TTOPPRODBYLOC);
            }

            else # Child
            {
                &p_LoadTableTTOPPRODBYLOC::LoadTableTTOPPRODBYLOC;
                exit(0);
            }

            close(TTOPPRODBYLOC);
            return \%_TABLE_TTOPPRODBYLOC;
        }
        {
            package p_LoadTableTTOPPRODBYLOC;
            sub LoadTableTTOPPRODBYLOC
            {
#           !/usr/bin/perl
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#            vim: syntax=perl ts=4 sw=4
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#           Generated By: pequel Version 2.4-4, Build: Tuesday November  1 08:45:13 GMT 2005
#                       : http://sourceforge.net/projects/pequel/
#           Script Name : top_prod_by_loc.pql
#           Created On  : Thu Nov  3 15:42:27 2005
#           Perl Version: /usr/bin/perl 5.6.1 on solaris
#           For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#           Options:
#               input_file(sample.data) input data filename
#               header(1) write header record to output.
#               optimize(1) optimize generated code.
#               hash(1) Generate in memory. Input data can be unsorted.
#               doc_title(Pequel Table 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__MAXSALES       => int    2;
                use constant _O_PRODUCT_CODE    => int    3;
                local $\="\n";
                local $,="|";
                use constant VERBOSE => int 10000;
                use constant LAST_ICELL => int 8;
                my @I_VAL;
                my %O_VAL;
                my $key;
                open(DATA, q{examples/sample.data})|| die "Cannot open examples/sample.data: $!";
                &PrintHeader();
                while (<DATA>)
                {
                    chomp;
                    @I_VAL = split("[|]", $_);
                    $key = ( $I_VAL[_I_LOCATION] );
                    $O_VAL{$key}{_O_LOCATION} = $I_VAL[_I_LOCATION];
                    $I_VAL[_I_SALES_TOTAL] = $I_VAL[_I_SALES_QTY] * $I_VAL[_I_SALES_PRICE];
                    $O_VAL{$key}{_O__MAXSALES} = $I_VAL[_I_SALES_TOTAL]
                        if (!defined($O_VAL{$key}{_O__MAXSALES}) || $I_VAL[_I_SALES_TOTAL] > $O_VAL{$key}{_O__
MAXSALES});

                    if (sprintf("%.2f",$I_VAL[_I_SALES_TOTAL]) eq sprintf("%.2f",$O_VAL{$key}{_O__MAXSALES}))
{
                        $O_VAL{$key}{_O_PRODUCT_CODE} = $I_VAL[_I_PRODUCT_CODE] if (!defined($O_VAL{$key}{_O_P
RODUCT_CODE}));
                    }
                }

                foreach $key (sort  keys %O_VAL)
                {
                    print STDOUT
                        $O_VAL{$key}{_O_LOCATION},
                        $O_VAL{$key}{_O_PRODUCT_CODE}
                    ;
                }

#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                sub PrintHeader
                {
                    local $\="\n";
                    local $,="|";
                    print STDOUT
                        'LOCATION',
                        'PRODUCT_CODE'
                        ;
                }

            }
        }

        sub PrintHeader
        {
            local $\="\n";
            local $,="|";
            print STDOUT
                'LOCATION',
                'SALES_TOTAL',
                'TOP_PRODUCT'
                ;
        }

    }

}

#++++++ Table TSALESBYPROD --> Type :ETL::Pequel::Type::Table::External::Pequel ++++++
sub LoadTableTSALESBYPROD
{
    my %_TABLE_TSALESBYPROD;
    my $pid = open(TSALESBYPROD, '-|'); # Fork
    my $count=0;
    if ($pid) # Parent
    {
        while (<TSALESBYPROD>)
        {
            chomp;
            my (@flds) = split("[|]", $_, -1);
            $_TABLE_TSALESBYPROD{$flds[0]} = $flds[ 1 ];
        }

        $count=$.;
        close(TSALESBYPROD);
    }

    else # Child
    {
        &p_LoadTableTSALESBYPROD::LoadTableTSALESBYPROD;
        exit(0);
    }

    close(TSALESBYPROD);
    return \%_TABLE_TSALESBYPROD;
}

{
    package p_LoadTableTSALESBYPROD;
    sub LoadTableTSALESBYPROD
    {
#    !/usr/bin/perl
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#     vim: syntax=perl ts=4 sw=4
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Generated By: pequel Version 2.4-4, Build: Tuesday November  1 08:45:13 GMT 2005
#                : http://sourceforge.net/projects/pequel/
#    Script Name : sales_ttl_by_prod.pql
#    Created On  : Thu Nov  3 15:42:31 2005
#    Perl Version: /usr/bin/perl 5.6.1 on solaris
#    For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Options:
#        input_file(sample.data) input data filename
#        header(1) write header record to output.
#        optimize(1) optimize generated code.
#        doc_title(Pequel Table 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_PRODUCT_CODE    => int    1;
        use constant _O_SALES_TOTAL     => int    2;
        local $\="\n";
        local $,="|";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 8;
        my @I_VAL;
        my @O_VAL;
        my $key__I_PRODUCT_CODE;
        my $previous_key__I_PRODUCT_CODE = undef;
        foreach my $f (1..2) { $O_VAL[$f] = undef; }
        open(DATA, q{examples/sample.data})|| die "Cannot open examples/sample.data: $!";
        &PrintHeader();
        while (<DATA>)
        {
            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)
            {
                print STDOUT
                    $O_VAL[_O_PRODUCT_CODE],
                    $O_VAL[_O_SALES_TOTAL]
                ;
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
                @O_VAL = undef;
            }

            $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_PRODUCT_CODE],
            $O_VAL[_O_SALES_TOTAL]
        ;
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        sub PrintHeader
        {
            local $\="\n";
            local $,="|";
            print STDOUT
                'PRODUCT_CODE',
                'SALES_TOTAL'
                ;
        }

    }

}

{
    package p_copy_input_copy_record_sa;
    sub copy_input_copy_record_sa
    {
#    !/usr/bin/perl
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#     vim: syntax=perl ts=4 sw=4
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Generated By: pequel Version 2.4-4, Build: Tuesday November  1 08:45:13 GMT 2005
#                : http://sourceforge.net/projects/pequel/
#    Script Name : copy_record_SA.pql
#    Created On  : Thu Nov  3 15:42:36 2005
#    Perl Version: /usr/bin/perl 5.6.1 on solaris
#    For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Options:
#        optimize(1) optimize generated code.
#        doc_title(Copy 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_LOCATION_NAME    => int    3;
        use constant _O_LOCATION_NAME    => int    1;
        use constant _O_PRODUCT_CODE     => int    2;
        use constant _O_SALES_TOTAL      => int    3;
        local $\="\n";
        local $,="|";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 3;
        my @I_VAL;
        my @O_VAL;
        my $key__I_PRODUCT_CODE;
        my $previous_key__I_PRODUCT_CODE = undef;
        foreach my $f (1..3) { $O_VAL[$f] = undef; }
#     Sort:PRODUCT_CODE(asc:string)
        open(DATA, q{cat  - | sort  -t’|’ -y -k 2,2 2>/dev/null |}) || die "Cannot open input: $!";
        while (<DATA>)
        {
            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)
            {
                flock(STDOUT, LOCK_EX);
                print STDOUT
                    $O_VAL[_O_LOCATION_NAME],
                    $O_VAL[_O_PRODUCT_CODE],
                    $O_VAL[_O_SALES_TOTAL]
                ;
                flock(STDOUT, LOCK_UN);
                $previous_key__I_PRODUCT_CODE = $key__I_PRODUCT_CODE;
                @O_VAL = undef;
            }

            $I_VAL[_I_LOCATION_NAME] = ’South Australia’;
            $O_VAL[_O_LOCATION_NAME] = $I_VAL[_I_LOCATION_NAME];
            $O_VAL[_O_PRODUCT_CODE] = $I_VAL[_I_PRODUCT_CODE];
            $O_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq ’’);
        }

        flock(STDOUT, LOCK_EX);
        print STDOUT
            $O_VAL[_O_LOCATION_NAME],
            $O_VAL[_O_PRODUCT_CODE],
            $O_VAL[_O_SALES_TOTAL]
        ;
        flock(STDOUT, LOCK_UN);
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    }

}

{
    package p_copy_output_copy_output_combiner;
    sub copy_output_copy_output_combiner
    {
#    !/usr/bin/perl
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#     vim: syntax=perl ts=4 sw=4
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Generated By: pequel Version 2.4-4, Build: Tuesday November  1 08:45:13 GMT 2005
#                : http://sourceforge.net/projects/pequel/
#    Script Name : copy_output_combiner.pql
#    Created On  : Thu Nov  3 15:42:38 2005
#    Perl Version: /usr/bin/perl 5.6.1 on solaris
#    For         :
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
#    Options:
#        optimize(1) optimize generated code.
#        doc_title(Copy Output 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_NAME    => int    0;
        use constant _I_PRODUCT_CODE     => int    1;
        use constant _I_SALES_TOTAL      => int    2;
        use constant _I_DESCRIPTION      => int    3;
        use constant _O_LOCATION_NAME    => int    1;
        use constant _O_DESCRIPTION      => int    2;
        use constant _O_SALES_TOTAL      => int    3;
        local $\="\n";
        local $,="|";
        use constant VERBOSE => int 10000;
        use constant LAST_ICELL => int 3;
        my @I_VAL;
        my @O_VAL;
        my $key__I_LOCATION_NAME;
```

```
            my $previous_key__I_LOCATION_NAME = undef;
            foreach my $f (1..3) { $O_VAL[$f] = undef; }
#       Sort:LOCATION_NAME(asc:string)
            open(DATA, q{cat  - | sort  -t'|' -y -k 1,1 2>/dev/null |}) || die "Cannot open input: $!";
            while (<DATA>)
            {
                chomp;
                @I_VAL = split("[|]", $_);
                $key__I_LOCATION_NAME = $I_VAL[_I_LOCATION_NAME];
                if (!defined($previous_key__I_LOCATION_NAME))
                {
                    $previous_key__I_LOCATION_NAME = $key__I_LOCATION_NAME;
                }

                elsif ($previous_key__I_LOCATION_NAME ne $key__I_LOCATION_NAME)
                {
                    flock(STDOUT, LOCK_EX);
                    print STDOUT
                        $O_VAL[_O_LOCATION_NAME],
                        $O_VAL[_O_DESCRIPTION],
                        $O_VAL[_O_SALES_TOTAL]
                    ;
                    flock(STDOUT, LOCK_UN);
                    $previous_key__I_LOCATION_NAME = $key__I_LOCATION_NAME;
                    @O_VAL = undef;
                }

                $O_VAL[_O_LOCATION_NAME] = $I_VAL[_I_LOCATION_NAME];
                $I_VAL[_I_DESCRIPTION] = 'State Total';
                $O_VAL[_O_DESCRIPTION] = $I_VAL[_I_DESCRIPTION];
                $O_VAL[_O_SALES_TOTAL] += $I_VAL[_I_SALES_TOTAL] unless ($I_VAL[_I_SALES_TOTAL] eq '');
            }

            flock(STDOUT, LOCK_EX);
            print STDOUT
                $O_VAL[_O_LOCATION_NAME],
                $O_VAL[_O_DESCRIPTION],
                $O_VAL[_O_SALES_TOTAL]
            ;
            flock(STDOUT, LOCK_UN);
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        }

}

{
    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 2.4-4, Build: Tuesday November  1 08:45:13 GMT 2005
#                : http://sourceforge.net/projects/pequel/
#    Script Name : diverted_record_low.pql
#    Created On  : Thu Nov  3 15:42:35 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.
#        hash(1) Generate in memory. Input data can be unsorted.
#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            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 $,="|";
            use constant VERBOSE => int 10000;
            use constant LAST_ICELL => int 3;
            my @I_VAL;
            my %O_VAL;
            my $key;
            while (<STDIN>)
        {
            chomp;
            @I_VAL = split("[|]", $_);
            $key = ( $I_VAL[_I_LOCATION] ) . '|' . ( $I_VAL[_I_PRODUCT_CODE] );
            $I_VAL[_I_CATEGORY] = 'LOW';
            $O_VAL{$key}{_O_CATEGORY} = $I_VAL[_I_CATEGORY];
            $O_VAL{$key}{_O_LOCATION} = $I_VAL[_I_LOCATION];
            $O_VAL{$key}{_O_PRODUCT_CODE} = $I_VAL[_I_PRODUCT_CODE];
            $O_VAL{$key}{_O_SALES_TOTAL} = $I_VAL[_I_SALES_TOTAL];
        }

        foreach $key (sort  keys %O_VAL)
        {
            flock(STDOUT, LOCK_EX);
            print STDOUT
                $O_VAL{$key}{_O_CATEGORY},
                $O_VAL{$key}{_O_LOCATION},
                $O_VAL{$key}{_O_PRODUCT_CODE},
                $O_VAL{$key}{_O_SALES_TOTAL}
            ;
            flock(STDOUT, LOCK_UN);
        }

#-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    }

}

sub PrintHeader
{
    local $\="\n";
    local $,="|";
    flock(STDOUT, LOCK_EX);
    print STDOUT
        'PRODUCT_CODE',
        'RECORD_COUNT',
        'SALES_QTY_SAMPLE1',
        'SALES_QTY_SAMPLE2',
        'S1_DESCRIPTION',
        'S1_LOCATION',
        'S2_DESCRIPTION',
        'S2_LOCATION',
        'PRODUCT_SALES_TOTAL',
        'LOCATION_SALES_TOTAL'
        ;
    flock(STDOUT, LOCK_UN);
}
```

7. ABOUT PEQUEL

This document was generated by Pequel.

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