```
NOTE - THESE PROGRAM SAMPLES SPLIT, FILTER, POST, TRANSFORM AND ASSIGN INPUT DATA TO
NOTE - OUTPUT FILES, CHECK FOR LOGICAL ERRORS AND COUNT RECORDS FOR A NYSE COMPLIANCE PROJECT.
NOTE - CODING WAS DONE IN FOXPRO, WHICH IS SIMILAR TO VISUAL BASIC AND 4GL PROGRAMMING.
NOTE - CODE USES INDICES AND PROCESSES DATA IN LOOPS FOR DIRECT DATA ACCESS VS. SQL AND CURSOR LOOPS.

NOTE - PROGRAMS:
(1) PARSE PROGRAM - SPLIT INPUT INTO TWO OUTPUT FILES BASED ON RECORD TYPING AND COMPUTE
  SOME DERIVATIVE FIELD VALUES
(2) POST  PROGRAM - CALCULATE AGGREGATE STATISTICS FOR DETAIL DATA AND POST TO NEW TABLES
  AND CREATE EXCEL SPREADSHEETS TO REPORT ON RESULTS

*********************************************************************************
* PROGRAM: parse_mol
*
* AUTHOR: M FREEMAN
* DATE: 5/12/2004
*
* COMMENT: This program takes NYSE merged order log and parses the 1A/1B and 2A record types into
*          the mol1a and mol2a tables for additional processing.  Note: the stock symbol is changed
*          to remove any internal spaces to match NYSE OTS requirement
*********************************************************************************
set echo off
set status off
set talk off

select a
use "c:\vfp\mol1a"

select b
use "c:\vfp\mol2a"

NOTE- SPLIT molinput INTO mol1a and mol2a BY RECORD TYPE
select c
use "c:\vfp\molinput2"

set safety off

select a
zap

select b
zap

a1rcount = 0
b1rcount = 0
a2rcount = 0
b2rcount = 0
c2rcount = 0
d2rcount = 0
e2rcount = 0
a3rcount = 0
a4rcount = 0
orcount  = 0

select c
go top

*****************************************************************
do while not eof('c')
*****************************************************************

rec_type = left(c->input,2)

DO CASE
CASE inlist(rec_type, "1A", "1B")

        select a
        append blank

        replace a->type               with left(c->input,2)
        replace a->mnemonic    with substr(c->input,3,4)
        replace a->clearingid  with substr(c->input,7,4)

        NOTE - REMOVE INTERNAL SPACES IN STOCK SYMBOL BEFORE SAVING
        stock = alltrim(substr(c->input,11,11))
        pos = atc(" ",stock)
        if pos > 0
                replace a->symbol with left(stock, pos-1) + right(stock,len(stock)-pos)
        else
```

```
                replace a->symbol             with stock
        endif

        replace a->turnaround  with substr(c->input,26,6)
        replace a->branch      with substr(c->input,32,4)
        replace a->sequence    with substr(c->input,36,5)
        replace a->sunit       with substr(c->input,41,3)
        replace a->dupe        with substr(c->input,53,1)
        replace a->booth       with substr(c->input,54,1)
        replace a->otime       with substr(c->input,55,6)
        replace a->odate       with substr(c->input,61,8)
        replace a->ostatus     with substr(c->input,69,1)
        replace a->otype       with substr(c->input,70,1)
        replace a->buy_sell    with substr(c->input,71,1)
        replace a->otype       with substr(c->input,70,1)
        replace a->entry_type  with substr(c->input,72,1)
        replace a->tif         with substr(c->input,73,1)
        replace a->qty         with val(substr(c->input,74,7))
        replace a->lpricecde   with substr(c->input,81,1)
        replace a->lprice      with val(substr(c->input,82,12))/100
        replace a->spricecde   with substr(c->input,94,1)
        replace a->sprice      with val(substr(c->input,95,12))/100
        replace a->sterms      with substr(c->input,107,3)
        replace a->acct_type   with substr(c->input,110,1)
        replace a->cturnnum    with substr(c->input,112,6)
        replace a->cbranchseq  with substr(c->input,118,9)
        replace a->aon         with substr(c->input,127,1)
        replace a->tstop       with substr(c->input,128,1)
        replace a->dni_dnr     with substr(c->input,129,1)
        replace a->currency    with substr(c->input,130,3)
        replace a->ebooth_id   with substr(c->input,133,3)
        replace a->dbooth_id   with substr(c->input,136,3)
        replace a->asof        with substr(c->input,139,1)
        replace a->owdate      with substr(c->input,140,5)
        replace a->owtime      with substr(c->input,148,6)
        replace a->dbtime      with substr(c->input,154,10)
*       replace a->adot_ind    with substr(c->input,169,1)
*       replace a->cspecl      with substr(c->input,170,4)
        replace a->remainder   with right(c->input,164)

        do case
        case rec_type = "1A"
                a1rcount = a1rcount + 1
        case rec_type = "1B"
                b1rcount = b1rcount + 1
        endcase

CASE inlist(rec_type, "2A", "2B", "2C", "2D", "2E")


        select b
        append blank

        replace b->type        with left(c->input,2)
        replace b->mnemonic    with substr(c->input,3,4)
        replace b->clearingid  with substr(c->input,7,4)

        NOTE - REMOVE INTERNAL SPACES IN STOCK SYMBOL
        stock = alltrim(substr(c->input,11,11))
        pos = atc(" ",stock)
        if pos > 0
                replace b->symbol with left(stock, pos-1) + right(stock,len(stock)-pos)
        else
                replace b->symbol with stock
        endif

        replace b->turnaround  with substr(c->input,26,6)
        replace b->branch      with substr(c->input,32,4)
        replace b->sequence    with substr(c->input,36,5)
        replace b->sunit       with substr(c->input,41,3)
        replace b->activity    with substr(c->input,44,9)
        replace b->dupe        with substr(c->input,53,1)
        replace b->booth       with substr(c->input,54,1)
        replace b->rtime       with substr(c->input,55,6)
        replace b->rtype       with substr(c->input,61,1)
        replace b->principal   with substr(c->input,62,2)
        replace b->exclear     with substr(c->input,64,1)
        replace b->unit        with substr(c->input,65,1)
        replace b->leave_qty   with val(substr(c->input,66,7))
```

```
        replace b->broker_num  with substr(c->input,73,4)
        replace b->numdecimal  with val(substr(c->input,77,1))
        replace b->price       with val(substr(c->input,78,12))/100
        replace b->omnibus     with substr(c->input,90,1)
        replace b->bust_time   with substr(c->input,91,6)
        replace b->memoa       with substr(c->input,97,4)
        replace b->memob       with substr(c->input,101,6)
        replace b->ortime      with substr(c->input,107,6)
        replace b->last_flag   with substr(c->input,113,1)
        replace b->rsource     with substr(c->input,114,1)
        replace b->nx_ind      with substr(c->input,115,1)
        replace b->otime       with substr(c->input,122,6)
        replace b->otime_ind   with substr(c->input,128,1)
        replace b->cmnemonic   with substr(c->input,136,4)
        replace b->cqty        with val(substr(c->input,140,7))
        replace b->cbadge      with substr(c->input,147,4)
        replace b->cwtime      with substr(c->input,151,6)
        replace b->crflag      with substr(c->input,157,1)
        replace b->remainder   with right(c->input,158)

        DO CASE rec_type = "2A"
               a2rcount = a2rcount + 1

        CASE rec_type = "2B"

               b2rcount = b2rcount + 1

        CASE rec_type = "2C"

               c2rcount = c2rcount + 1

        CASE rec_type = "2D"

               d2rcount = d2rcount + 1

        CASE rec_type = "2E"

               e2rcount = e2rcount + 1

        ENDCASE

CASE rec_type = "3A"

        a3rcount = a3rcount + 1

CASE rec_type = "4A"

        a4rcount = a4rcount + 1
OTHERWISE
        orcount = orcount + 1
ENDCASE

select c
skip in c
enddo

?"1A    orders        = ", a1rcount
?"1B    orders        = ", b1rcount
?"Total orders        = ", a1rcount + b1rcount
?"2A    fills         = ", a2rcount
?"2B    fills         = ", b2rcount
?"2C    fills         = ", c2rcount
?"2D    fills         = ", d2rcount
?"2E    fills         = ", e2rcount
?"3A    admin msgs    = ", a3rcount
?"4A    rejects       = ", a4rcount
?"other rcount        = ", orcount

*********************************************************************
*********************************************************************
NOTE - POST EXECUTION REPORT RECORDS 2A TO ORDER RECORDS 1A
*********************************************************************
*********************************************************************
NOTE - ONLY POST ORIGINAL, REPLACEMENT or PRIOR DATE ORDER RECORDS, NOT CANCELS
select a
index on turnaround for inlist(ostatus,"O","R","P") tag turnaround

select b
go top
```

```
exec_posted = 0
exec_not_found = 0
exec_processed = 0

*****************************************************************
do while not eof('b')
*****************************************************************
        exec_processed = exec_processed + 1

        select a
        seek b->turnaround

        if found()

                exec_posted = exec_posted + 1

                NOTE - POST ORIGINAL ORDER QTY ON EXECUTION RECORD
                if empty(b->orig_qty)
                        select b
                        replace b->orig_qty with a->qty
                endif

                NOTE - CALCULATE QTY FOR EXECUTION FILL (NOT ON FILL RECORD)
                calc_fill_qty = a->qty - b->leave_qty - a->qty_filled
                calc_fill_value = calc_fill_qty * b->price

                replace b->fill_qty with calc_fill_qty

                select a
                replace a->num_fills with a->num_fills + 1

                NOTE - CALCULATE QTY FILLED BY COMPARING ORIGINAL QTY WITH REMAINDER
                replace a->qty_filled with a->qty - b->leave_qty

                replace a->tot_value with a->tot_value + calc_fill_value

                if a->low_price = 0 or a->low_price > b->price
                        replace a->low_price with b->price
                endif

                if a->high_price = 0 or a->high_price < b->price
                        replace a->high_price with b->price
                endif

                if a->avg_price = 0 or (a->tot_value > 0 and a->qty_filled > 0)
                        replace a->avg_price with a->tot_value / a->qty_filled
                endif

        else
                exec_not_found = exec_not_found + 1
        endif

select b
skip in b
enddo

?"executions processed= ", exec_processed
?"executions posted   = ", exec_posted
?"executions not found= ", exec_not_found


sele c
go top

sele b
go top

sele a
set order to
set filter to
go top
brow

NOTE - END OF PROGRAM #1


*************************************************************************************************
PROGRAM #2:
```

```
*************************************************************************************************

*************************************************************************************************
* PROGRAM: post_ots
*
* AUTHOR: M FREEMAN
* DATE: 5/17/2004
*
* COMMENT: This program takes order and execution data in the NYSE merged order log table mol1a
*          to post aggregate statistics for each branch, mnemonic and stock symbol.
*          NOTE: A separate table, identical in structure, is created for branch, mnemonic and symbol.
*************************************************************************************************
set echo off
set status off
set talk off
close databases all

*************************************************************************************************
NOTE - OPEN NYSE MERGED ORDER LOG ORDER/FILL SOURCE DATA FILE
NOTE - PROCESS FROM TOP TO BOTTOM -- NO INDEX ORDER OR INDEX FILTER REQUIRED
NOTE - FILTER RECORDS TO POST ORIGINAL/REPLACE/PRIOR DATE ORDER RECORDS ONLY
*************************************************************************************************

select a
use "c:\vfp\mol1a"
set order to
*set filter to inlist(a->ostatus, "O", "R", "P")

*************************************************************************************************
NOTE - CREATE STOCK SYMBOL STATISTICS
*************************************************************************************************
select b
use "c:\vfp\symbol"
index on symbol tag symbol

set safety off
zap

NOTE - NYSE STOCK SYMBOL/DESCRIPTION/CUSIP TABLE
select c
use "c:\vfp\nyse"
index on symbol tag nyse

stat_code = "SYMBOL"

Do POST_STATS

*************************************************************************************************
NOTE - CREATE BRANCH STATISTICS
*************************************************************************************************
select b
use "c:\vfp\branch"
index on branch tag branch

set safety off
zap

select c
NOTE - NEED A COMPLETE BRANCH WIRE CODE TABLE -- THIS TABLE IS ONLY A START
use "c:\vfp\wire_code2"
index on wire_code tag wire

stat_code = "BRANCH"

Do POST_STATS

*************************************************************************************************
NOTE - CREATE MNEMONIC STATISTICS
*************************************************************************************************
select b
use "c:\vfp\mnemonic"
index on mnemonic tag mnemonic

set safety off
zap

NOTE - NO DESCRIPTIVE LOOK-UP TABLE EXISTS FOR MNEMONIC CODES

stat_code = "MNEMONIC"
```

```
Do POST_STATS

*************************************************************************************************
NOTE - BROWSE MNEMONIC FILE
*************************************************************************************************
select b
go top
brow

return

*************************************************************************************************
NOTE - BEGINNING OF PROCEDURE SECTION
*************************************************************************************************

*************************************************************************************************
NOTE - POST STATS PROCEDURE IS DRIVER LOOP THROUGH SOURCE DATA
NOTE - AFTER PROCESSING ALL MERGED ORDER LOG RECORDS, TOTALS ARE ALSO POSTED
*************************************************************************************************
PROCEDURE POST_STATS
*************************************************************************************************
select a
go top

NOTE - PROCESS MERGED ORDER LOG DATA FROM TOP TO BOTTOM OF FILE
SCAN

        DO UPDATE_STATS

ENDSCAN

NOTE - POST AVERAGES -- DONE IN A SEPARATE STEP INSTEAD
DO POST_AVERAGES

NOTE - SUM-UP COLUMNS AND PUT TOTALS DIRECTLY ON STATISTICS TABLES
DO POST_TOTALS

return

*************************************************************************************************
NOTE - UPDATE STATS PROCEDURE COUNTS, ACCUMULATES, AVERAGES ORDER AND ORDER EXECUTION (FILL) DATA
*************************************************************************************************
PROCEDURE UPDATE_STATS
*************************************************************************************************

        select b

        NOTE - SEARCH FOR BRANCH/MNEMONIC/SYMBOL CODE
        DO CASE
        CASE stat_code = "BRANCH"
                seek a->branch
        CASE stat_code = "MNEMONIC"
                seek a->mnemonic
        CASE stat_code = "SYMBOL"
                seek alltrim(a->symbol)
        ENDCASE

        NOTE - ADD A NEW ROW IF NOT ALREADY IN TABLE
        IF NOT FOUND()

                select b
                append blank

                NOTE - SAVE NEW BRANCH/MNEMONIC/SYMBOL CODE
                DO CASE
                CASE stat_code = "BRANCH"
                        replace b->branch with a->branch
                CASE stat_code = "MNEMONIC"
                        replace b->mnemonic with a->mnemonic
                CASE stat_code = "SYMBOL"
                        replace b->symbol with a->symbol
                ENDCASE

                NOTE - COUNT ORDER RECORDS BY TYPE -- RETURN/LOOP IF A CANCEL RECORD; PROCESS IF AN ORDER RECORD
                replace b->tot_ord with 1

                if inlist(a->ostatus, "C", "X")
                        replace b->tot_cancel with 1
```

```
        return
else
        replace b->num_orders with 1
endif

NOTE - INITIALIZE TOTAL QUANTITY ORDERED AND TOTAL QUANTITY FILLED
replace b->tot_o_qty with a->qty
replace b->tot_f_qty with a->qty_filled

NOTE - COUNT ORDERS BASED ON QUANTITY FILLED -- FOR 'afill' = All Filled;
NOTE - 'pfill' = Partially Filled; 'nfill' = Not Filled

DO CASE
CASE a->qty = a->qty_filled
        replace b->num_afill with 1
CASE a->qty > a->qty_filled and a->qty_filled > 0
        replace b->num_pfill with 1
CASE a->qty_filled = 0
        replace b->num_nfill with 1
ENDCASE

if a->qty_filled > 0
        replace b->num_filled with 1
endif

NOTE - INITIALIZE TOTAL DOLLAR VALUE OF FILLED QUANTITY
replace b->tot_value with a->tot_value

NOTE - INITIALIZE ORDER TIME IN FORCE COUNTERS
if inlist(a->tif, "2","4","8")
        replace b->tot_gtc with 1
else
        replace b->tot_day with 1
endif

NOTE - INITIALIZE ORDER TYPE COUNTERS
do case
case inlist(a->otype, "0","4","5","6","8")
        replace b->tot_mkt with 1
case inlist(a->otype, "1","9","A","B","D")
        replace b->tot_limit with 1
case inlist(a->otype, "2","3")
        replace b->tot_stop with 1
otherwise
        replace b->tot_misc with 1
endcase

NOTE - INITIALIZE ORDER ACCOUNT TYPE COUNTERS
do case
case a->acct_type = "A"
        replace b->tot_agency with 1
case a->acct_type = "I"
        replace b->tot_indivi with 1
case a->acct_type = "P"
        replace b->tot_princp with 1
case a->acct_type = "Y"
        replace b->tot_progrm with 1
otherwise
        replace b->tot_otype with 1
endcase

NOTE - INITIALIZE ORDER SOURCE COUNTERS
do case
case a->entry_type = "0"
        replace b->tot_cms with 1
case a->entry_type = "2"
        replace b->tot_dbook with 1
case a->entry_type = "3"
        replace b->tot_bbss with 1
otherwise
        replace b->tot_osrc with 1
endcase

NOTE - INITIALIZE COUNTS, QUANTITIES, AND AVERAGES BASED ON BUY/SELL/SELL SHORT
NOTE - or OTHER ORDER TYPE
do case
case a->buy_sell = "1"
        replace b->tot_buy     with a->qty
        replace b->num_buy     with 1
```

```
                replace b->net_buy    with a->qty
                replace b->tot_fb_qty  with a->qty_filled
                replace b->tot_bvalue  with a->tot_value
                if a->qty_filled > 0
                        replace b->num_fbuy with 1
                endif
        case a->buy_sell = "2"
                replace b->tot_sell    with a->qty
                replace b->num_sell    with 1
                replace b->net_buy     with -1*a->qty
                replace b->tot_fs_qty  with a->qty_filled
                replace b->tot_svalue  with a->tot_value
                if a->qty_filled > 0
                        replace b->num_fsell with 1
                endif
        case a->buy_sell = "3"
                replace b->tot_short   with a->qty
                replace b->num_short   with 1
                replace b->net_buy     with -1*a->qty
                replace b->tot_fss_qt  with a->qty_filled
                replace b->tot_shtval  with a->tot_value
                if a->qty_filled > 0
                        replace b->num_fshort with 1
                endif
        otherwise
                replace b->tot_other   with a->qty
                replace b->num_other   with 1
        endcase

        NOTE - CALCULATE THE RATIO OF BUYING TO SELLING BASED ON TOTAL DOLLAR VALUE OF TRADES
        NOTE - NOTE: NET SELLING SET TO A NEGATIVE VALUE
        do case
        case b->tot_bvalue > (b->tot_svalue + b->tot_shtval) and (b->tot_svalue + b->tot_shtval) > 0
                replace b->buy_v_sell with b->tot_bvalue / (b->tot_svalue + b->tot_shtval)
        case (b->tot_svalue + b->tot_shtval) > b->tot_bvalue and b->tot_bvalue > 0
                replace b->buy_v_sell with -1*((b->tot_svalue + b->tot_shtval) / b->tot_bvalue)
        otherwise
*               replace b->buy_v_sell with 0
        endcase

        NOTE - CALCULATE BUY/SELL/SELL SHORT PERCENTAGES FOR ORDER QUANTITIES
        replace b->net_buy     with b->tot_buy - b->tot_sell - b->tot_short
        replace b->buy_perc    with ((b->tot_buy / b->tot_o_qty)*100)
        replace b->sell_perc   with ((b->tot_sell / b->tot_o_qty)*100)
        replace b->short_perc  with ((b->tot_short / b->tot_o_qty)*100)
        replace b->perc_buy    with ((b->num_buy / b->num_orders)*100)
        replace b->perc_sell   with ((b->num_sell / b->num_orders)*100)
        replace b->perc_short  with ((b->num_short / b->num_orders)*100)

        NOTE - LOOK-UP DESCRIPTION FOR BRANCH OR SYMBOL (NONE FOR MNEMONIC FOR NOW)
        NOTE - IF NOT FOUND, THE DESCRIPTION WILL BE BLANK -- CHECK FOR MISSING STOCK NAMES !!
        NOTE - BRANCH TABLE IS INCOMPLETE -- BASED ON WIRE CODES FOR NYSE
        DO CASE
        CASE stat_code = "BRANCH"
                NOTE - GET BRANCH NAME
                select c
                seek a->branch
                if found() and alltrim(a->branch) == alltrim(c->wire_code)
                        select b
                        replace b->name with c->descrip
                endif
        CASE stat_code = "MNEMONIC"
        CASE stat_code = "SYMBOL"
                NOTE - GET STOCK NAME
                select c
                seek a->symbol
                if found() and alltrim(a->symbol) == alltrim(c->symbol)
                        select b
                        replace b->name with c->name
                endif
        ENDCASE

NOTE - UPDATE EXISTING RECORD WITH ACCUMULATED COUNTS AND VALUES
else
        select b

        NOTE - COUNT ORDER RECORDS BY TYPE -- RETURN/LOOP IF A CANCEL RECORD; PROCESS IF AN ORDER RECORD
        replace b->tot_ord with b->tot_ord + 1
```

```
if inlist(a->ostatus, "C", "X")
        replace b->tot_cancel with b->tot_cancel + 1
        return
else
        replace b->num_orders with b->num_orders + 1
endif

NOTE - INITIALIZE TOTAL QUANTITY ORDERED AND TOTAL QUANTITY FILLED
replace b->tot_o_qty with b->tot_o_qty + a->qty
replace b->tot_f_qty with b->tot_f_qty + a->qty_filled

NOTE - COUNT ORDERS BASED ON QUANTITY FILLED -- FOR 'afill' = All Filled;
NOTE - 'pfill' = Partially Filled; 'nfill' = Not Filled
DO CASE
CASE a->qty = a->qty_filled
        replace b->num_afill with b->num_afill + 1
CASE a->qty > a->qty_filled and a->qty_filled > 0
        replace b->num_pfill with b->num_pfill + 1
CASE a->qty_filled = 0
        replace b->num_nfill with b->num_nfill + 1
ENDCASE

if a->qty_filled > 0
        replace b->num_filled with b->num_filled + 1
endif

NOTE - UPDATE TOTAL DOLLAR VALUE OF FILLED QUANTITY
replace b->tot_value with b->tot_value + a->tot_value

NOTE - INITIALIZE ORDER TIME IN FORCE COUNTERS
if inlist(a->tif, "2","4","8")
        replace b->tot_gtc with b->tot_gtc + 1
else
        replace b->tot_day with b->tot_day + 1
endif

NOTE - INITIALIZE ORDER TYPE COUNTERS
do case
case inlist(a->otype, "0","4","5","6","8")
        replace b->tot_mkt with b->tot_mkt + 1
case inlist(a->otype, "1","9","A","B","D")
        replace b->tot_limit with b->tot_limit + 1
case inlist(a->otype, "2","3")
        replace b->tot_stop with b->tot_stop + 1
otherwise
        replace b->tot_misc with b->tot_misc + 1
endcase

NOTE - INITIALIZE ORDER ACCOUNT TYPE COUNTERS
do case
case a->acct_type = "A"
        replace b->tot_agency  with b->tot_agency + 1
case a->acct_type = "I"
        replace b->tot_indivi  with b->tot_indivi + 1
case a->acct_type = "P"
        replace b->tot_princp  with b->tot_princp + 1
case a->acct_type = "Y"
        replace b->tot_progrm  with b->tot_progrm + 1
otherwise
        replace b->tot_otype   with b->tot_otype + 1
endcase

NOTE - INITIALIZE ORDER SOURCE COUNTERS
do case
case a->entry_type = "0"
        replace b->tot_cms     with b->tot_cms + 1
case a->entry_type = "2"
        replace b->tot_dbook   with b->tot_dbook + 1
case a->entry_type = "3"
        replace b->tot_bbss    with b->tot_bbss + 1
otherwise
        replace b->tot_osrc    with b->tot_osrc + 1
endcase

NOTE - UPDATE COUNTS, QUANTITIES, AND AVERAGES BASED ON BUY/SELL/SELL SHORT or OTHER ORDER TYPE
do case
case a->buy_sell = "1"
        replace b->tot_buy     with b->tot_buy + a->qty
        replace b->num_buy     with b->num_buy + 1
```

```
                        replace b->tot_fb_qty  with b->tot_fb_qty + a->qty_filled
                        replace b->tot_bvalue  with b->tot_bvalue + a->tot_value
                        if a->qty_filled > 0
                                replace b->num_fbuy with b->num_fbuy + 1
                        endif
                case a->buy_sell = "2"
                        replace b->tot_sell    with b->tot_sell + a->qty
                        replace b->num_sell    with b->num_sell + 1
                        replace b->tot_fs_qty  with b->tot_fs_qty + a->qty_filled
                        replace b->tot_svalue  with b->tot_svalue + a->tot_value
                        if a->qty_filled > 0
                                replace b->num_fsell with b->num_fsell + 1
                        endif
                case a->buy_sell = "3"
                        replace b->tot_short   with b->tot_short + a->qty
                        replace b->tot_fss_qt  with b->tot_fss_qt + a->qty_filled
                        replace b->tot_shtval  with b->tot_shtval + a->tot_value
                        replace b->num_short   with b->num_short + 1
                        if a->qty_filled > 0
                                replace b->num_fshort with b->num_fshort + 1
                        endif
                otherwise
                        replace b->tot_other   with b->tot_other + a->qty
                        replace b->num_other   with b->num_other + 1
                endcase

                NOTE - CALCULATE THE RATIO OF BUYING TO SELLING BASED ON TOTAL DOLLAR VALUE OF TRADES (NOTE: NET
                SELLING SET TO A NEGATIVE VALUE)
                do case
                case b->tot_bvalue > (b->tot_svalue + b->tot_shtval) and (b->tot_svalue + b->tot_shtval) > 0
                        replace b->buy_v_sell with b->tot_bvalue / (b->tot_svalue + b->tot_shtval)
                case (b->tot_svalue + b->tot_shtval) > b->tot_bvalue and b->tot_bvalue > 0
                        replace b->buy_v_sell with -1*((b->tot_svalue + b->tot_shtval) / b->tot_bvalue)
                otherwise
*                       replace b->buy_v_sell with 0
                endcase

                NOTE - CALCULATE BUY/SELL/SELL SHORT PERCENTAGES FOR ORDER QUANTITIES
                replace b->net_buy     with b->tot_buy - b->tot_sell - b->tot_short
                replace b->buy_perc    with ((b->tot_buy / b->tot_o_qty)*100)
                replace b->sell_perc   with ((b->tot_sell / b->tot_o_qty)*100)
                replace b->short_perc  with ((b->tot_short / b->tot_o_qty)*100)
                replace b->perc_buy    with ((b->num_buy / b->num_orders)*100)
                replace b->perc_sell   with ((b->num_sell / b->num_orders)*100)
                replace b->perc_short  with ((b->num_short / b->num_orders)*100)
        endif

return


*************************************************************************************************
NOTE - UPDATE AVERAGES FOR ORDER AND ORDER EXECUTION (FILL) DATA
*************************************************************************************************
PROCEDURE POST_AVERAGES
*************************************************************************************************
select b
go top

scan

        DO CALC_AVERAGES

endscan

*************************************************************************************************
NOTE - UPDATE AVERAGES FOR ORDER AND ORDER EXECUTION (FILL) DATA
*************************************************************************************************
PROCEDURE CALC_AVERAGES
*************************************************************************************************
                NOTE - UPDATE AVERAGE ORDER AND FILL QUANTITIES
                if b->num_orders > 0
                        replace b->avg_qty with b->tot_o_qty / b->num_orders
                endif
                if b->num_afill + b->num_pfill > 0
                        replace b->avg_fqty  with b->tot_f_qty / (b->num_afill + b->num_pfill)
                endif

                NOTE - CALCULATE PERCENTAGE OF ORDERS THAT RECEIVED A FULL OR PARTIAL FILL; CALCULATE THE
                PERCENTAGE OF QUANTITY ORDERED THAT IS FILLED
```

```
                if b->num_orders > 0
                        replace b->perc_fill with ((b->num_afill + b->num_pfill) / b->num_orders)*100
                endif
                if b->tot_o_qty > 0
                        replace b->fill_perc with (b->tot_f_qty / b->tot_o_qty)*100
                endif
                if b->num_buy > 0
                        replace b->avg_buy with (b->tot_buy / b->num_buy)
                endif
                if b->num_fbuy > 0
                        replace b->avg_bval with (b->tot_bvalue / b->num_fbuy)
                endif
                if b->tot_fb_qty > 0
                        replace b->avg_bprice with (b->tot_bvalue / b->tot_fb_qty)
                endif
                if b->num_sell > 0
                        replace b->avg_sell with (b->tot_sell / b->num_sell)
                endif
                if b->tot_fs_qty > 0
                        replace b->avg_sprice with (b->tot_svalue / b->tot_fs_qty)
                endif
                if b->num_fsell > 0
                        replace b->avg_sval with (b->tot_svalue / b->num_fsell)
                endif
                if b->num_fshort > 0
                        replace b->avg_short with (b->tot_short / b->num_fshort)
                endif

return

*********************************************************************************************
NOTE - POST TOTALS INTO FILE
*********************************************************************************************
PROCEDURE POST_TOTALS
*********************************************************************************************
select b

NOTE - ADD A BLANK ROW FOR TOTALS
go bottom
append blank

DO CASE
CASE stat_code = "BRANCH"
        replace b->branch with " TOTALS:"
CASE stat_code = "MNEMONIC"
        replace b->mnemonic with " TOTALS:"
CASE stat_code = "SYMBOL"
        replace b->symbol with " TOTALS:"
ENDCASE

replace b->name with alltrim(str(reccount())) + " Records"

sum b->tot_ord to t0
sum b->tot_cancel to tc

sum b->tot_mkt to m1
sum b->tot_limit to m2
sum b->tot_stop to m3

sum b->tot_day to m4
sum b->tot_gtc to m5

sum b->tot_agency to m6
sum b->tot_indivi to m7
sum b->tot_princp to m8
sum b->tot_progrm to m9
sum b->tot_otype  to m9b

sum b->tot_cms to m10
sum b->tot_dbook to m11
sum b->tot_bbss to m12

sum b->num_orders to tn1
sum b->num_buy to tn2
sum b->num_sell to tn3
sum b->num_short to tn4

sum b->tot_buy to tt1
sum b->tot_sell to tt2
```

```
sum b->tot_short to tt3

sum b->num_filled to n1
sum b->num_fbuy to n2
sum b->num_fsell to n3
sum b->num_fshort to n4

sum b->num_afill to f1
sum b->num_pfill to f2
sum b->num_nfill to f3

sum b->tot_value to tn5
sum b->tot_bvalue to tn6
sum b->tot_svalue to tn7
sum b->tot_shtval to tn8

gtot_val = int(tn5)
gtot_bval = int(tn6)
gtot_sval = int(tn7)
gtot_ssval = int(tn8)

sum b->tot_o_qty to q1
sum b->tot_f_qty to q2
sum b->tot_fb_qty to q3
sum b->tot_fs_qty to q4
sum b->tot_fss_qt to q5

NOTE - REPOSITION FILE ON 'TOTALS' ROW BY SEEKING THIS RECORD FIRST
seek " TOTALS:"

IF found()

        replace b->num_orders  with int(tn1)
        replace b->num_buy     with int(tn2)
        replace b->num_sell    with int(tn3)
        replace b->num_short   with int(tn4)
        replace b->num_afill   with int(f1)
        replace b->num_pfill   with int(f2)
        replace b->num_nfill   with int(f3)
        replace b->num_filled  with int(n1)
        replace b->num_fbuy    with int(n2)
        replace b->num_fsell   with int(n3)
        replace b->num_fshort  with int(n4)
        replace b->tot_value   with int(tn5)
        replace b->tot_bvalue  with int(tn6)
        replace b->tot_svalue  with int(tn7)
        replace b->tot_shtval  with int(tn8)
        replace b->tot_o_qty   with int(q1)
        replace b->tot_f_qty   with int(q2)
        replace b->tot_fb_qty  with int(q3)
        replace b->tot_fs_qty  with int(q4)
        replace b->tot_fss_qt  with int(q5)
        replace b->tot_buy     with int(tt1)
        replace b->tot_sell    with int(tt2)
        replace b->tot_short   with int(tt3)
        replace b->tot_ord     with int(t0)
        replace b->tot_cancel  with int(tc)
        replace b->tot_mkt     with int(m1)
        replace b->tot_limit   with int(m2)
        replace b->tot_stop    with int(m3)
        replace b->tot_day     with int(m4)
        replace b->tot_gtc     with int(m5)
        replace b->tot_agency  with int(m6)
        replace b->tot_indivi  with int(m7)
        replace b->tot_princp  with int(m8)
        replace b->tot_progrm  with int(m9)
        replace b->tot_otype   with int(m9b)
        replace b->tot_cms     with int(m10)
        replace b->tot_dbook   with int(m11)
        replace b->tot_bbss    with int(m12)

endif

seek " TOTALS:"

IF found()

        NOTE - CALCULATE THE RATIO OF BUYING TO SELLING BASED ON TOTAL DOLLAR VALUE OF TRADES
```

```
        NOTE - NET SELLING SET TO A

NEGATIVE VALUE)
        do case
        case b->tot_bvalue > (b->tot_svalue + b->tot_shtval) and (b->tot_svalue + b->tot_shtval) > 0
                replace b->buy_v_sell with b->tot_bvalue / (b->tot_svalue + b->tot_shtval)
        case (b->tot_svalue + b->tot_shtval) > b->tot_bvalue and b->tot_bvalue > 0
                replace b->buy_v_sell with -1*((b->tot_svalue + b->tot_shtval) / b->tot_bvalue)
        endcase

        NOTE - CALCULATE BUY/SELL/SELL SHORT PERCENTAGES FOR ORDER QUANTITIES
        replace b->net_buy     with b->tot_buy - b->tot_sell - b->tot_short
        replace b->buy_perc    with ((b->tot_buy / b->tot_o_qty)*100)
        replace b->sell_perc   with ((b->tot_sell / b->tot_o_qty)*100)
        replace b->short_perc  with ((b->tot_short / b->tot_o_qty)*100)
        replace b->perc_buy    with ((b->num_buy / b->num_orders)*100)
        replace b->perc_sell   with ((b->num_sell / b->num_orders)*100)
        replace b->perc_short  with ((b->num_short / b->num_orders)*100)

        DO CALC_AVERAGES
endif


*************************************************************************************************
NOTE - POST PERCENTAGE OF TOTAL VALUE TRADED FOR EACH RECORD
*************************************************************************************************
select b
go top

scan
        if gtot_val > 0
                replace perc_tval with (b->tot_value / gtot_val) * 100
        endif
        if gtot_bval > 0
                replace perc_bval with (b->tot_bvalue / gtot_bval) * 100
        endif
        if gtot_sval > 0
                replace perc_sval with (b->tot_svalue / gtot_sval) * 100
        endif
        if gtot_ssval > 0
                replace perc_ssval with (b->tot_shtval / gtot_ssval) * 100
        endif
endscan

*************************************************************************************************
NOTE - SAVE RESULTS TO EXCEL
*************************************************************************************************

select b
go top

DO CASE
CASE stat_code = "BRANCH"
        copy to c:\ots\branch.xls type xls
CASE stat_code = "MNEMONIC"
        copy to c:\ots\mnemonic.xls type xls
CASE stat_code = "SYMBOL"
        copy to c:\ots\symbol.xls type xls
ENDCASE

return
```