# ABAP Code

## Global Declaration

[Error: No requirements found]

## Selection Screen

[Error: No requirements found]

## Processing Logic

START-OF-SELECTION.  
  
 PERFORM get\_data.  
  
 IF p\_close = 'X'.  
 PERFORM close\_pos USING p\_test.  
 PERFORM get\_data. "refresh output after closing  
 ENDIF.  
  
 PERFORM display\_alv.  
  
  
  
FORM get\_data.  
  
 TYPES: BEGIN OF ty\_join,  
 ebeln TYPE ekpo-ebeln,  
 ebelp TYPE ekpo-ebelp,  
 matnr TYPE ekpo-matnr,  
 txz01 TYPE ekpo-txz01,  
 menge TYPE ekpo-menge,  
 meins TYPE ekpo-meins,  
 netpr TYPE ekpo-netpr,  
 netwr TYPE ekpo-netwr,  
 elikz TYPE ekpo-elikz,  
 loekz TYPE ekpo-loekz,  
 werks TYPE ekpo-werks,  
 lifnr TYPE ekko-lifnr,  
 bedat TYPE ekko-bedat,  
 bsart TYPE ekko-bsart,  
 waers TYPE ekko-waers,  
 mtart TYPE mara-mtart,  
 END OF ty\_join.  
  
 TYPES: BEGIN OF ty\_wemng,  
 ebeln TYPE eket-ebeln,  
 ebelp TYPE eket-ebelp,  
 wemng TYPE eket-wemng,  
 END OF ty\_wemng.  
  
 TYPES: BEGIN OF ty\_lfa1,  
 lifnr TYPE lfa1-lifnr,  
 name1 TYPE lfa1-name1,  
 END OF ty\_lfa1.  
  
 TYPES: BEGIN OF ty\_out,  
 ebeln TYPE ekpo-ebeln,  
 ebelp TYPE ekpo-ebelp,  
 matnr TYPE ekpo-matnr,  
 txz01 TYPE ekpo-txz01,  
 lifnr TYPE ekko-lifnr,  
 name1 TYPE lfa1-name1,  
 bedat TYPE ekko-bedat,  
 bsart TYPE ekko-bsart,  
 menge TYPE ekpo-menge,  
 meins TYPE ekpo-meins,  
 netpr TYPE ekpo-netpr,  
 netwr TYPE ekpo-netwr,  
 wemng TYPE eket-wemng,  
 to\_deliver TYPE ekpo-menge,  
 werks TYPE ekpo-werks,  
 waers TYPE ekko-waers,  
 status TYPE c LENGTH 1,  
 message TYPE string,  
 END OF ty\_out.  
  
 DATA: lt\_join TYPE STANDARD TABLE OF ty\_join,  
 lt\_join\_filt TYPE STANDARD TABLE OF ty\_join,  
 lt\_wemng TYPE STANDARD TABLE OF ty\_wemng,  
 lt\_lfa1 TYPE STANDARD TABLE OF ty\_lfa1,  
 ls\_join TYPE ty\_join,  
 ls\_out TYPE ty\_out.  
  
 DATA: lv\_wemng TYPE eket-wemng.  
  
 FIELD-SYMBOLS: <ls\_join> TYPE ty\_join,  
 <ls\_out> TYPE ty\_out.  
  
 CLEAR gt\_out.  
 REFRESH gt\_out.  
  
 SELECT ekpo~ebeln,  
 ekpo~ebelp,  
 ekpo~matnr,  
 ekpo~txz01,  
 ekpo~menge,  
 ekpo~meins,  
 ekpo~netpr,  
 ekpo~netwr,  
 ekpo~elikz,  
 ekpo~loekz,  
 ekpo~werks,  
 ekko~lifnr,  
 ekko~bedat,  
 ekko~bsart,  
 ekko~waers,  
 mara~mtart  
 FROM ekpo  
 INNER JOIN ekko ON ekko~ebeln = ekpo~ebeln  
 LEFT OUTER JOIN mara ON mara~matnr = ekpo~matnr  
 INTO TABLE @lt\_join  
 WHERE ekpo~loekz = @space  
 AND ekpo~elikz <> 'X'.  
  
 IF sy-subrc <> 0 OR lt\_join IS INITIAL.  
 RETURN.  
 ENDIF.  
  
 "Apply selection filters from selection-screen (if provided)  
 LOOP AT lt\_join ASSIGNING <ls\_join>.  
 IF s\_matnr[] IS NOT INITIAL AND <ls\_join>-matnr NOT IN s\_matnr.  
 CONTINUE.  
 ENDIF.  
 IF s\_werks[] IS NOT INITIAL AND <ls\_join>-werks NOT IN s\_werks.  
 CONTINUE.  
 ENDIF.  
 IF s\_lifnr[] IS NOT INITIAL AND <ls\_join>-lifnr NOT IN s\_lifnr.  
 CONTINUE.  
 ENDIF.  
 IF s\_mtart[] IS NOT INITIAL AND <ls\_join>-mtart NOT IN s\_mtart.  
 CONTINUE.  
 ENDIF.  
 APPEND <ls\_join> TO lt\_join\_filt.  
 ENDLOOP.  
  
 IF lt\_join\_filt IS INITIAL.  
 RETURN.  
 ENDIF.  
  
 "Delivered quantity by item  
 SELECT ebeln,  
 ebelp,  
 SUM( wemng ) AS wemng  
 FROM eket  
 INTO TABLE @lt\_wemng  
 FOR ALL ENTRIES IN @lt\_join\_filt  
 WHERE ebeln = @lt\_join\_filt-ebeln  
 AND ebelp = @lt\_join\_filt-ebelp  
 GROUP BY ebeln, ebelp.  
  
 SORT lt\_wemng BY ebeln ebelp.  
  
 "Vendor names  
 SELECT lifnr,  
 name1  
 FROM lfa1  
 INTO TABLE @lt\_lfa1  
 FOR ALL ENTRIES IN @lt\_join\_filt  
 WHERE lifnr = @lt\_join\_filt-lifnr.  
  
 SORT lt\_lfa1 BY lifnr.  
  
 LOOP AT lt\_join\_filt INTO ls\_join.  
  
 READ TABLE lt\_wemng WITH KEY ebeln = ls\_join-ebeln  
 ebelp = ls\_join-ebelp  
 TRANSPORTING NO FIELDS BINARY SEARCH.  
 IF sy-subrc = 0.  
 lv\_wemng = lt\_wemng[ ebeln = ls\_join-ebeln ebelp = ls\_join-ebelp ]-wemng.  
 ELSE.  
 lv\_wemng = 0.  
 ENDIF.  
  
 CLEAR ls\_out.  
 ls\_out-ebeln = ls\_join-ebeln.  
 ls\_out-ebelp = ls\_join-ebelp.  
 ls\_out-matnr = ls\_join-matnr.  
 ls\_out-txz01 = ls\_join-txz01.  
 ls\_out-lifnr = ls\_join-lifnr.  
 ls\_out-bedat = ls\_join-bedat.  
 ls\_out-bsart = ls\_join-bsart.  
 ls\_out-menge = ls\_join-menge.  
 ls\_out-meins = ls\_join-meins.  
 ls\_out-netpr = ls\_join-netpr.  
 ls\_out-netwr = ls\_join-netwr.  
 ls\_out-wemng = lv\_wemng.  
 ls\_out-to\_deliver = ls\_join-menge - lv\_wemng.  
 ls\_out-werks = ls\_join-werks.  
 ls\_out-waers = ls\_join-waers.  
  
 READ TABLE lt\_lfa1 WITH KEY lifnr = ls\_out-lifnr TRANSPORTING NO FIELDS BINARY SEARCH.  
 IF sy-subrc = 0.  
 ls\_out-name1 = lt\_lfa1[ lifnr = ls\_out-lifnr ]-name1.  
 ENDIF.  
  
 "Only open items: positive remaining quantity and not completed/deleted  
 IF ls\_out-to\_deliver > 0.  
 APPEND ls\_out TO gt\_out.  
 ENDIF.  
  
 ENDLOOP.  
  
 SORT gt\_out BY ebeln ebelp.  
  
ENDFORM.  
  
  
  
FORM close\_pos USING iv\_test TYPE abap\_bool.  
  
 DATA: lt\_poitem TYPE TABLE OF bapimepoitem,  
 lt\_poitemx TYPE TABLE OF bapimepoitemx,  
 lt\_return TYPE TABLE OF bapiret2,  
 ls\_poitem TYPE bapimepoitem,  
 ls\_poitemx TYPE bapimepoitemx,  
 lv\_ebeln TYPE ekko-ebeln,  
 lv\_auth\_ok TYPE abap\_bool.  
  
 FIELD-SYMBOLS: <ls\_out> TYPE any.  
  
 IF gt\_out IS INITIAL.  
 RETURN.  
 ENDIF.  
  
 SORT gt\_out BY ebeln ebelp.  
  
 LOOP AT gt\_out ASSIGNING <ls\_out>.  
 <ls\_out>-status = space.  
 <ls\_out>-message = space.  
 ENDLOOP.  
  
 DATA: lv\_current\_ebeln TYPE ekko-ebeln VALUE ''.  
  
 LOOP AT gt\_out ASSIGNING <ls\_out>.  
 IF <ls\_out>-to\_deliver <= 0.  
 CONTINUE.  
 ENDIF.  
  
 "Authorization per plant to change PO items  
 CLEAR lv\_auth\_ok.  
 AUTHORITY-CHECK OBJECT 'M\_BEST\_WRK'  
 ID 'ACTVT' FIELD '02'  
 ID 'WERKS' FIELD <ls\_out>-werks.  
 IF sy-subrc = 0.  
 lv\_auth\_ok = abap\_true.  
 ENDIF.  
  
 IF lv\_auth\_ok IS INITIAL.  
 <ls\_out>-status = 'E'.  
 <ls\_out>-message = |No change authorization for plant { <ls\_out>-werks }|.  
 CONTINUE.  
 ENDIF.  
  
 IF lv\_current\_ebeln IS INITIAL OR lv\_current\_ebeln <> <ls\_out>-ebeln.  
 "Process any pending PO first  
 IF lv\_current\_ebeln IS NOT INITIAL AND lt\_poitem IS NOT INITIAL.  
 CLEAR lt\_return.  
 CALL FUNCTION 'BAPI\_PO\_CHANGE'  
 EXPORTING  
 purchaseorder = lv\_current\_ebeln  
 TABLES  
 return = lt\_return  
 poitem = lt\_poitem  
 poitemx = lt\_poitemx.  
  
 DATA(lv\_has\_error) = abap\_false.  
 LOOP AT lt\_return TRANSPORTING NO FIELDS WHERE type = 'E' OR type = 'A'.  
 lv\_has\_error = abap\_true.  
 EXIT.  
 ENDLOOP.  
  
 IF lv\_has\_error = abap\_false AND iv\_test IS INITIAL.  
 CALL FUNCTION 'BAPI\_TRANSACTION\_COMMIT'  
 EXPORTING  
 wait = 'X'.  
 ELSE.  
 CALL FUNCTION 'BAPI\_TRANSACTION\_ROLLBACK'.  
 ENDIF.  
  
 "Update messages back to items of lv\_current\_ebeln  
 LOOP AT gt\_out ASSIGNING <ls\_out> WHERE ebeln = lv\_current\_ebeln.  
 READ TABLE lt\_return WITH KEY type = 'E' TRANSPORTING NO FIELDS.  
 IF sy-subrc = 0.  
 <ls\_out>-status = 'E'.  
 CONCATENATE LINES OF lt\_return INTO <ls\_out>-message SEPARATED BY space.  
 ELSE.  
 IF iv\_test IS INITIAL.  
 <ls\_out>-status = 'S'.  
 <ls\_out>-message = 'Closed (NO\_MORE\_GR set)'.  
 ELSE.  
 <ls\_out>-status = 'I'.  
 <ls\_out>-message = 'Test run: would set NO\_MORE\_GR'.  
 ENDIF.  
 ENDIF.  
 ENDLOOP.  
  
 CLEAR: lt\_poitem, lt\_poitemx, lt\_return.  
 ENDIF.  
  
 lv\_current\_ebeln = <ls\_out>-ebeln.  
 ENDIF.  
  
 CLEAR: ls\_poitem, ls\_poitemx.  
 ls\_poitem-po\_item = <ls\_out>-ebelp.  
 ls\_poitem-no\_more\_gr = 'X'.  
  
 ls\_poitemx-po\_item = <ls\_out>-ebelp.  
 ls\_poitemx-no\_more\_gr = 'X'.  
  
 APPEND ls\_poitem TO lt\_poitem.  
 APPEND ls\_poitemx TO lt\_poitemx.  
  
 AT END OF ebeln.  
 "Finalize last PO in group  
 IF lv\_current\_ebeln IS NOT INITIAL AND lt\_poitem IS NOT INITIAL.  
 CLEAR lt\_return.  
 CALL FUNCTION 'BAPI\_PO\_CHANGE'  
 EXPORTING  
 purchaseorder = lv\_current\_ebeln  
 TABLES  
 return = lt\_return  
 poitem = lt\_poitem  
 poitemx = lt\_poitemx.  
  
 DATA(lv\_has\_error2) = abap\_false.  
 LOOP AT lt\_return TRANSPORTING NO FIELDS WHERE type = 'E' OR type = 'A'.  
 lv\_has\_error2 = abap\_true.  
 EXIT.  
 ENDLOOP.  
  
 IF lv\_has\_error2 = abap\_false AND iv\_test IS INITIAL.  
 CALL FUNCTION 'BAPI\_TRANSACTION\_COMMIT'  
 EXPORTING  
 wait = 'X'.  
 ELSE.  
 CALL FUNCTION 'BAPI\_TRANSACTION\_ROLLBACK'.  
 ENDIF.  
  
 LOOP AT gt\_out ASSIGNING <ls\_out> WHERE ebeln = lv\_current\_ebeln.  
 READ TABLE lt\_return WITH KEY type = 'E' TRANSPORTING NO FIELDS.  
 IF sy-subrc = 0.  
 <ls\_out>-status = 'E'.  
 CONCATENATE LINES OF lt\_return INTO <ls\_out>-message SEPARATED BY space.  
 ELSE.  
 IF iv\_test IS INITIAL.  
 <ls\_out>-status = 'S'.  
 <ls\_out>-message = 'Closed (NO\_MORE\_GR set)'.  
 ELSE.  
 <ls\_out>-status = 'I'.  
 <ls\_out>-message = 'Test run: would set NO\_MORE\_GR'.  
 ENDIF.  
 ENDIF.  
 ENDLOOP.  
  
 CLEAR: lt\_poitem, lt\_poitemx, lt\_return.  
 CLEAR lv\_current\_ebeln.  
 ENDIF.  
 ENDAT.  
  
 ENDLOOP.  
  
ENDFORM.  
  
  
  
FORM display\_alv.  
  
 TRY.  
 DATA(lo\_alv) = NEW cl\_salv\_table( ).  
 cl\_salv\_table=>factory(  
 IMPORTING  
 r\_salv\_table = lo\_alv  
 CHANGING  
 t\_table = gt\_out ).  
  
 lo\_alv->get\_columns( )->set\_optimize( abap\_true ).  
 lo\_alv->get\_display\_settings( )->set\_list\_header( 'Open Purchase Orders' ).  
 lo\_alv->get\_functions( )->set\_all( abap\_true ).  
  
 "Column optimizations and texts  
 DATA(lo\_cols) = lo\_alv->get\_columns( ).  
 lo\_cols->get\_column( 'EBELN' )->set\_long\_text( 'Purchasing Document' ).  
 lo\_cols->get\_column( 'EBELP' )->set\_long\_text( 'Item' ).  
 lo\_cols->get\_column( 'MATNR' )->set\_long\_text( 'Material' ).  
 lo\_cols->get\_column( 'TXZ01' )->set\_long\_text( 'Short Text' ).  
 lo\_cols->get\_column( 'LIFNR' )->set\_long\_text( 'Vendor' ).  
 lo\_cols->get\_column( 'NAME1' )->set\_long\_text( 'Vendor Name' ).  
 lo\_cols->get\_column( 'BEDAT' )->set\_long\_text( 'Document Date' ).  
 lo\_cols->get\_column( 'BSART' )->set\_long\_text( 'Document Type' ).  
 lo\_cols->get\_column( 'MENGE' )->set\_long\_text( 'PO Quantity' ).  
 lo\_cols->get\_column( 'MEINS' )->set\_long\_text( 'Order Unit' ).  
 lo\_cols->get\_column( 'NETPR' )->set\_long\_text( 'Net Price' ).  
 lo\_cols->get\_column( 'NETWR' )->set\_long\_text( 'Net Value' ).  
 lo\_cols->get\_column( 'WEMNG' )->set\_long\_text( 'Quantity Delivered' ).  
 lo\_cols->get\_column( 'TO\_DELIVER' )->set\_long\_text( 'To Be Delivered' ).  
  
 lo\_alv->display( ).  
  
 CATCH cx\_salv\_msg INTO DATA(lx\_msg).  
 MESSAGE lx\_msg->get\_text( ) TYPE 'S' DISPLAY LIKE 'E'.  
 ENDTRY.  
  
ENDFORM.

## Output Display

[Error: No requirements found]