```
REPORT zth_test7.
"In this report, I have a ztable which has already some records,
"however, I want to showcase the data and also allow the user
"to modify the records if they wish to do so.
"This requirement is very frequent in my case, sometimes,
"users want an alv report and also need a functionality
"to change the some of the data and save into the database.
"This requirement is so frequent that I have created this
"report which will take any z or y table and display its data
"in editable mode and users can modify the data of any field
"except keyfields.
CLASS lcl_exception DEFINITION INHERITING FROM cx_static_check.
  PUBLIC SECTION.
    DATA : message   TYPE string,
           mess_tab  TYPE bapiret2_tab,
           mess_line LIKE LINE OF mess_tab.

    METHODS : constructor IMPORTING REFERENCE(i_message)  TYPE string OPTIONAL
                                    REFERENCE(i_mess_tab) TYPE bapiret2_tab OPTIONAL
                                    REFERENCE(i_textid)   LIKE textid OPTIONAL
                                    REFERENCE(i_previous) LIKE previous OPTIONAL.
ENDCLASS.
CLASS lcl_exception IMPLEMENTATION.

  METHOD constructor.
    super->constructor( textid = i_textid previous = i_previous ).
    message = i_message.
    mess_tab = i_mess_tab.
  ENDMETHOD.

ENDCLASS.

CLASS lcl_alv DEFINITION.
  PUBLIC SECTION.
    CLASS-DATA : tabname       TYPE tabname.
    DATA    : lo_grid    TYPE REF TO cl_gui_alv_grid,
              lo_ref_sel TYPE REF TO data.

    METHODS : constructor IMPORTING REFERENCE(im_name) TYPE tabname RAISING lcl_exception,
      run RAISING lcl_exception.

  PROTECTED SECTION.
    METHODS :   create_dyn CHANGING REFERENCE(ch_ref) TYPE REF TO data,
      get_data       CHANGING REFERENCE(ch_ref)  TYPE REF TO data RAISING lcl_exception,
      get_components CHANGING REFERENCE(ch_components) TYPE cl_abap_structdescr=>component_table,
      create_table IMPORTING REFERENCE(im_components) TYPE cl_abap_structdescr=>component_table
                   EXPORTING REFERENCE(ex_ref_tab)    TYPE REF TO data RAISING lcl_exception,
      create_fcat  IMPORTING REFERENCE(im_ref_tab) TYPE REF TO data
                   CHANGING  REFERENCE(ch_fcat)    TYPE lvc_t_fcat,
      change_toolbar CHANGING REFERENCE(ch_exclude) TYPE ui_functions,
      display_alv RAISING lcl_exception.

  PRIVATE SECTION.
    DATA : lt_components TYPE cl_abap_structdescr=>component_table,
           lo_ref        TYPE REF TO data,
           lo_ref_tab    TYPE REF TO cl_abap_tabledescr,
           fcat          TYPE lvc_t_fcat,
           gr_exclude    TYPE ui_functions.

ENDCLASS.
```

```
CLASS lcl_alv IMPLEMENTATION.

  METHOD constructor.
    IF im_name+0(1) EQ 'Z' OR im_name+0(1) EQ 'z'
      OR  im_name+0(1) EQ 'Y' OR  im_name+0(1) EQ 'y'.
      tabname = im_name.
    ELSE.
      RAISE EXCEPTION TYPE lcl_exception EXPORTING i_message = 'No Z Table or Y Table Found'.
      RETURN.
    ENDIF.

  ENDMETHOD.

  METHOD create_fcat.
    DATA : lt_fcat TYPE lvc_t_fcat.
    FIELD-SYMBOLS : <ft_table2> TYPE table.
    ASSIGN lo_ref_sel->* TO <ft_table2>.

    cl_salv_table=>factory(
            IMPORTING
            r_salv_table   = DATA(lo_tab)    " Basis Class Simple ALV Tables
            CHANGING
              t_table        = <ft_table2>
              ).



    cl_salv_controller_metadata=>get_lvc_fieldcatalog(
    EXPORTING
      r_columns      = lo_tab->get_columns( )                " ALV Filter
      r_aggregations = lo_tab->get_aggregations( )           " ALV Aggregations
      RECEIVING
      t_fieldcatalog = lt_fcat                 " Field Catalog for List Viewer Control
      ).

    SELECT * FROM dd03l INTO TABLE @DATA(lt_dd03l) WHERE tabname = @tabname.

    LOOP AT lt_fcat ASSIGNING FIELD-SYMBOL(<f_fcat>).

      <f_fcat>-edit = COND #( WHEN <f_fcat>-fieldname NE 'SELECT'
                              THEN SWITCH #(
                                      lt_dd03l[ fieldname = <f_fcat>-fieldname ]-keyflag
                                                                  WHEN 'X' THEN ' '
                                                                  WHEN ' ' THEN 'X'  )
                                           ).

      <f_fcat>-col_opt = 'X'.
      IF <f_fcat>-fieldname EQ 'SELECT'.
        <f_fcat>-edit      = 'X'.
        <f_fcat>-scrtext_s = 'Select'.
        <f_fcat>-scrtext_m = 'Select'.
        <f_fcat>-scrtext_l = 'Select'.
        <f_fcat>-checkbox = 'X'.
      ENDIF.
    ENDLOOP.

    ch_fcat = lt_fcat.
  ENDMETHOD.
  METHOD change_toolbar.

    DATA : gs_exclude LIKE LINE OF me->gr_exclude.
```

```
  gs_exclude = cl_gui_alv_grid=>mc_fc_loc_insert_row.
  APPEND gs_exclude TO ch_exclude.
  gs_exclude = cl_gui_alv_grid=>mc_fc_loc_delete_row.
  APPEND gs_exclude TO ch_exclude.

ENDMETHOD.
METHOD create_table.

  DATA : lr_table TYPE REF TO cl_abap_tabledescr.
  DATA : lr_data TYPE REF TO data.
  me->get_components( CHANGING ch_components =  lt_components ).

  IF lines( lt_components ) IS NOT INITIAL.
    lr_table = cl_abap_tabledescr=>create(
    p_line_type  = cl_abap_structdescr=>create( p_components = lt_components )
    p_table_kind = cl_abap_tabledescr=>tablekind_std
    ).
  ELSE.
    RAISE EXCEPTION TYPE lcl_exception
      EXPORTING
        i_textid = cx_sy_struct_attributes=>empty_component_table.
  ENDIF.

  INSERT VALUE #( name = 'SELECT' type = cl_abap_elemdescr=>get_c( p_length = '1' ) )
                                                  INTO lt_components INDEX 1.
  CREATE DATA lr_data TYPE HANDLE lr_table.
  ASSIGN lr_data->* TO FIELD-SYMBOL(<ft_table2>).
  ASSIGN lo_ref->* TO FIELD-SYMBOL(<ft_table>).
  <ft_table2> = CORRESPONDING #( <ft_table>  ).
  ex_ref_tab = lr_data.

ENDMETHOD.

METHOD display_alv.

  ASSIGN lo_ref_sel->* TO FIELD-SYMBOL(<ft_table2>).

  "Design Field Catalogue
  me->create_fcat(
  EXPORTING
    im_ref_tab = lo_ref_sel
  CHANGING
    ch_fcat    = fcat
    ).

  "Exclude DELETE and INSERT icon because we will only
  "allow to modify the data.
  me->change_toolbar(
  CHANGING
    ch_exclude = gr_exclude
    ).

  "If you are about to use only one container, you can use
  "default screen for which you need to use cl_gui_container=>default_screen.
  "no need of creating and docking a container into the alv.
  CREATE OBJECT lo_grid EXPORTING i_parent = cl_gui_container=>default_screen.

  lo_grid->set_table_for_first_display(
 EXPORTING
      it_toolbar_excluding          = gr_exclude[]
```

```
  CHANGING
    it_outtab                       =   <ft_table2>               " Output Table
    it_fieldcatalog                 =   fcat                " Field Catalog
  EXCEPTIONS
    invalid_parameter_combination = 1               " Wrong Parameter
    program_error                 = 2               " Program Errors
    too_many_lines                = 3               " Too many Rows in Ready for Input Grid
    OTHERS                        = 4
).
    IF sy-subrc <> 0.
      RAISE EXCEPTION TYPE lcl_exception
        EXPORTING
          i_message = SWITCH #( sy-subrc  WHEN 1 THEN 'invalid_parameter_combination'
                                          WHEN 2 THEN 'program_error '
                                          WHEN 3 THEN 'too_many_lines'
                                          WHEN 4 THEN 'others'
                        ).
    ENDIF.
    CALL SCREEN 100.
  ENDMETHOD.

  METHOD get_components.

    DATA : gt_components TYPE cl_abap_structdescr=>component_table.

    ASSIGN lo_ref->* TO FIELD-SYMBOL(<ft_table>).

    "To get the name of the fields and other details.
    gt_components =
                      CAST cl_abap_structdescr(
                           CAST cl_abap_tabledescr(
                                cl_abap_typedescr=>describe_by_data( p_data = <ft_table> )
                                                  )->get_table_line_type( )
                                              )->get_components( ).

    DELETE gt_components WHERE name = 'MANDT'.
    ch_components = gt_components.

  ENDMETHOD.

  METHOD get_data.
    FIELD-SYMBOLS : <ft_table> TYPE table.
    ASSIGN ch_ref->* TO <ft_table>.
    SELECT * FROM (tabname) INTO TABLE <ft_table>.
    IF <ft_table> IS INITIAL.
      RAISE EXCEPTION TYPE lcl_exception
            EXPORTING i_message = |No Data Available in { me->tabname }|.
    ENDIF.
  ENDMETHOD.

  METHOD create_dyn.
    CREATE DATA ch_ref TYPE TABLE OF (tabname).
  ENDMETHOD.

  METHOD run.

    IF tabname IS INITIAL.
      RETURN.
    ENDIF.

    "Create a dynamic table
```

```
    me->create_dyn(
    CHANGING
      ch_ref = lo_ref
      ).

    "Get the table data into the report
    me->get_data(
    CHANGING
      ch_ref = lo_ref
      ).

    "Create a new table with one column named SELECT
    me->create_table(
    EXPORTING
      im_components = lt_components
    IMPORTING
      ex_ref_tab    = lo_ref_sel
      ).

    "Display the report.
    me->display_alv(  ).
  ENDMETHOD.

ENDCLASS.

START-OF-SELECTION.
  TRY .
      DATA(lo_obj) = NEW lcl_alv( 'ZCUST_EMAIL' ).
      TRY .
          lo_obj->run( ).
        CATCH lcl_exception INTO DATA(lo_exp).
          WRITE : lo_exp->message, lo_exp->get_text( ).
      ENDTRY.
    CATCH lcl_exception INTO lo_exp.
      WRITE : lo_exp->message, lo_exp->get_text( ).
  ENDTRY.


*&---------------------------------------------------------------------*
*&      Module  USER_COMMAND_0100  INPUT
*&---------------------------------------------------------------------*
MODULE user_command_0100 INPUT.
  CASE sy-ucomm.
    WHEN 'EXIT'.
      LEAVE PROGRAM.
    WHEN 'BACK'.
      LEAVE PROGRAM.
    WHEN  'SAVE'.
      PERFORM save.
      PERFORM refresh.
    WHEN OTHERS.
  ENDCASE.
ENDMODULE.

*&---------------------------------------------------------------------*
*&      Module  STATUS_0100  OUTPUT
*&---------------------------------------------------------------------*
MODULE status_0100 OUTPUT.
  SET PF-STATUS 'TEST'.
*  SET TITLEBAR 'xxx'.
ENDMODULE.
```

```
*&---------------------------------------------------------------------*
*&      Form  SAVE
*&---------------------------------------------------------------------*
FORM save .
  DATA : ir_itab TYPE REF TO data.
  CREATE DATA ir_itab TYPE TABLE OF (lcl_alv=>tabname).
  FIELD-SYMBOLS : <fs_table> TYPE table.
  ASSIGN ir_itab->* TO <fs_table>.

  "Check if there is any data changed.
  IF lo_obj->lo_grid IS BOUND.
    lo_obj->lo_grid->check_changed_data( ).
  ENDIF.

  "Move the data from one table to another
  "which has same type as ztable
  ASSIGN lo_obj->lo_ref_sel->* TO FIELD-SYMBOL(<fs_table_sel>).
  <fs_table_sel> = CORRESPONDING #( <fs_table> ).

  "Modify the same Z or Y table from the selected rows.
  MODIFY (lcl_alv=>tabname) FROM TABLE <fs_table>.
ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  REFRESH
*&---------------------------------------------------------------------*
FORM refresh .
  DATA : stable TYPE lvc_s_stbl.
  DATA : m_itab  TYPE bapiret2_tab.

  CALL METHOD lo_obj->lo_grid->set_ready_for_input
    EXPORTING
      i_ready_for_input = 0.

  "Input is disabled right after saving the values.

  stable-row = 'X'.
  stable-col = 'X'.

  "Refreshed alv is generated and displayed
  "with non-editable values.

  CALL METHOD lo_obj->lo_grid->refresh_table_display
    EXPORTING
      is_stable = stable
    EXCEPTIONS
      finished  = 1
      OTHERS    = 2.

  IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*            WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDIF.
ENDFORM.
```