**First Week Programs**

**1.Basic Codes**

REPORT ZDATATYPES.  
  
data a type i VALUE 123.  
WRITE a.  
  
  
**2.Assingning and Printing**  
  
data lv\_a type i VALUE 123.  
WRITE lv\_a.  
  
**3. size**  
  
data lv\_a(20) type c VALUE 'ust hyd'. ""character will take default value as 1 if we can,t declare the size  
WRITE lv\_a.  
  
**4. Addition program with chain operator**  
  
data :lv\_a TYPE i VALUE 10,  
      lv\_b TYPE i VALUE 20,  
      lv\_c TYPE i.  
  
lv\_c =  lv\_a + lv\_b.  
  
WRITE: 'the value is' , lv\_c.  
  
  
**5. By parameters**   ""   Taking values at run time instead of giving in program  
  
PARAMETERS:  p\_a TYPE i ,  
             p\_b TYPE i .  
  
data  lv\_c TYPE i.  
  
lv\_c =  p\_a + p\_b.  
  
WRITE: 'the value is' , lv\_c.  
  
  
PARAMETERS:  p\_a TYPE i DEFAULT 10,  
             p\_b TYPE i DEFAULT 20.  
  
data  lv\_c TYPE i.  
  
lv\_c =  p\_a + p\_b.  
  
WRITE: 'the value is' , lv\_c.  
  
  
\*6.  
  
WRITE: 'The current date is' ,  sy-datum,/    "System fields starts with sy  
       'Time : '  , sy-uzeit,/                 " def:  System directly gives the value to system fields----- Inline  line comment  
       'sap user', sy-uname.  
  
**7. Constants** ""keyword  
  
CONSTANTS CS\_ABC type i VALUE 210.  
  
cs\_abc = 234.  
WRITE cs\_abc.  
  
  
**8. FIELD STRING**  
  
data: begin of b1,       " FIELD STRING    :  store the bunch of values in a single entity or filed.  
        empid       type i value  123,  
        empname(20) type c value 'UST',  
      end of b1.  
data: begin of b2,  
        empid        type i,  
        empaddrs(30) type c,  
        empname(20)  type c,  
  
      end of b2.  
      move b1 to b2.  
\*move-corresponding b1 to b2.  
write :/ b2-empid , b2-empname , b2-empaddrs.

**2.Control Statements**

REPORT ZNAGESH\_CTRL\_STATEMENTS.  
  
  
**1.program**  
  
do 5 times.  
  write / 'SAP UST GLOBAL'.  
enddo.  
  
**2.Using while**  
  
data m type i VALUE 2.  
  
WHILE  m < 10.  
  
WRITE / m.  
  
m = m  + 1.  
  
ENDWHILE.  
  
  
**3.EXIT /STOP.**  
  
do.  
  write sy-index.  
  if sy-index eq 10.  
 \*EXIT.  
    stop.  
  endif.  
enddo.  
write / 'SAP UST GLOBAL'.  
  
  
  
  
  
**4.CONTINUE**  
do.  
  if sy-index eq 2.  
    continue.  
  endif.  
  
  write sy-index.  
  if sy-index eq 10.  
    exit.  
  endif.  
enddo.  
write / 'SAP UST GLOBAL'.

**5.Check**

do.  
    write sy-index.  
  if sy-index eq 10.  
    exit.  
  endif.  
  CHECK sy-index eq 2.  
    write sy-index.  
enddo.  
write / 'SAP UST GLOBAL'.  
  
  
  
**6.case**

PARAMETERS p\_val type i DEFAULT 2.  
  
CASE p\_val.  
  WHEN 1.  
    WRITE : ' first val' , p\_val.  
  WHEN 2.  
    WRITE :' second val' , p\_val.  
  
  WHEN OTHERS.  
    WRITE :' default val' , p\_val.  
  
ENDCASE.  
  
\*WRITE  p\_val

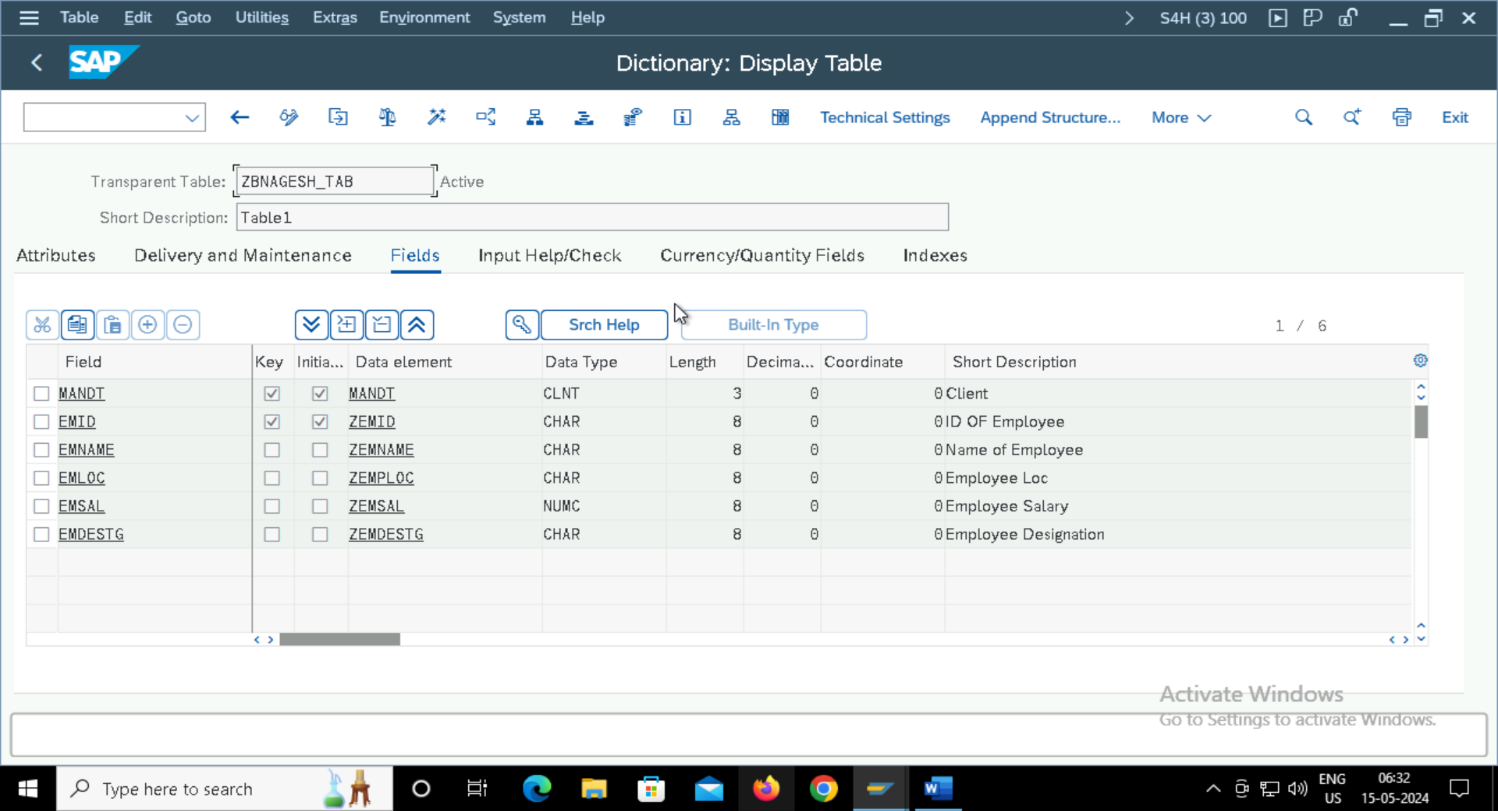
**3.Strings**

REPORT ZNAG\_SRINGS.  
  
**1.Translate**  
PARAMETERS: p\_a(20),p\_b(20).  
TRANSLATE  p\_a to UPPER CASE.  
TRANSLATE p\_b to LOWER CASE.  
write:/ p\_a,p\_b.  
  
  
**2. String Length**  
  
PARAMETERS p\_a(20).  
data lv\_a type i.  
lv\_a = STRLEN( p\_a ).  
write lv\_a.  
  
**3.CONCATENATE**  
  
PARAMETERS : P\_A(20) , P\_B(20).  
DATA LV\_STR(20) TYPE C.  
CONCATENATE P\_A P\_B INTO LV\_STR.  
WRITE LV\_STR.  
  
  
  
**4.REPLACE**  
  
parameters p\_A(20).  
  
DO.  
  IF p\_a ca ','.  
    replace ',' with '$' into p\_a.  
  ELSE.  
    exit.  
  ENDIF.  
ENDDO.  
write: p\_a.  
  
  
  
  
**5.SPLIT**  
  
PARAMETERS p\_a(20).  
data :  lv\_a(20), lv\_b(20).  
split p\_a at ',' into lv\_a lv\_b.  
write : / lv\_A, lv\_b.

**6.OFFSETTING**  
  
  
PARAMETERS P\_A(20).  
DATA LV\_DEST(20).  
LV\_DEST = P\_A+2(4).  
WRITE LV\_DEST.

**4.Table Creations**

**Table 1**

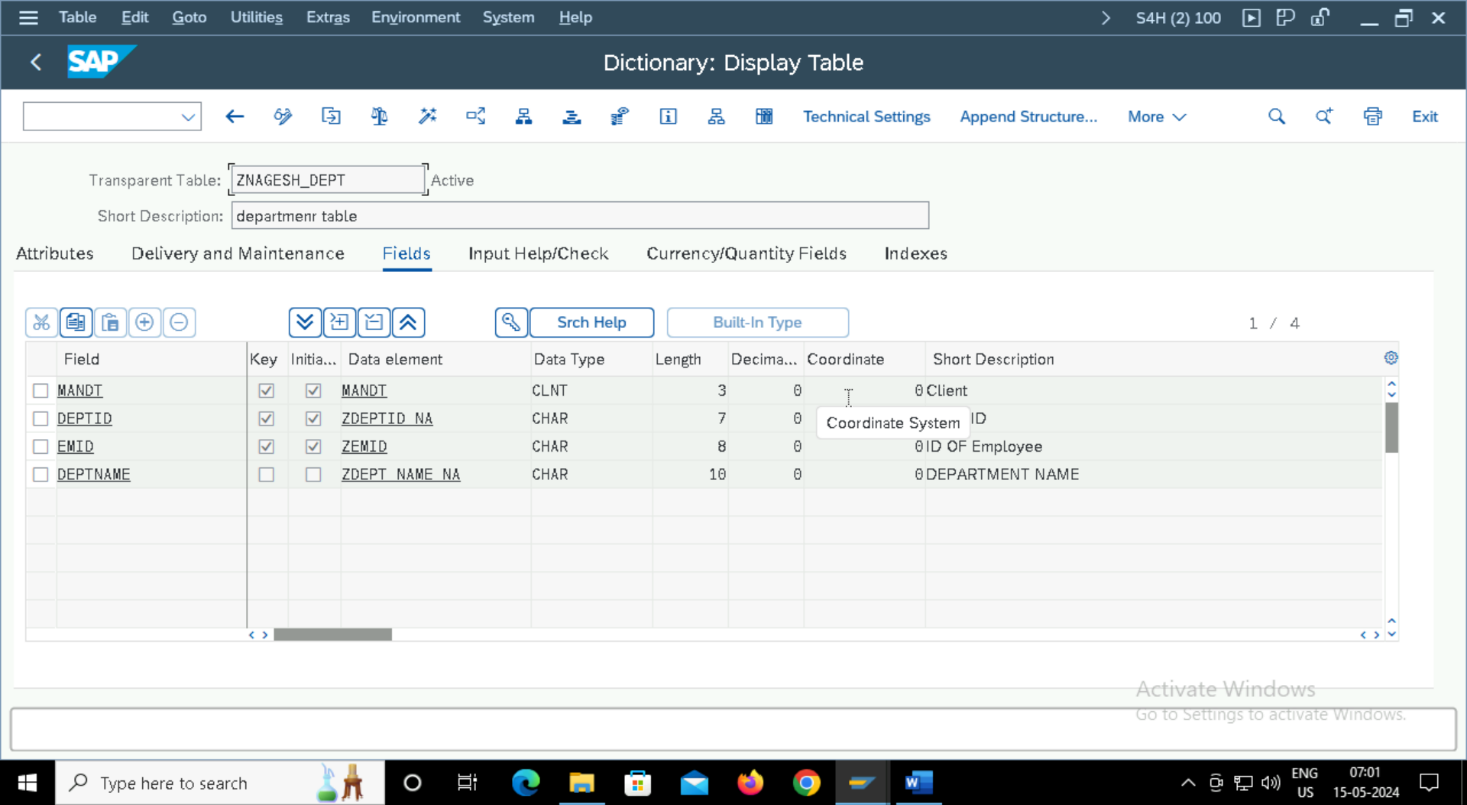
****

**Output:**

**A screenshot of a computer

Description automatically generated**

**Table 2:**

****

**Foreign Key:**

**A screenshot of a computer

Description automatically generated**

**Output:**

**A screenshot of a computer

Description automatically generated**

**5.Select Statements**

REPORT ZNAGESH\_SELECT\_STATEMENTS.  
  
  
**\*1st program for retrieving the data from all columns and specified columns**  
  
Data:  
lv\_EMPID TYPE  ZEMPID,  *"" assigned varuables as corresponded table columns*  
lv\_EMPNAME TYPE  ZEMPNAME,    *" either table name- column name or Data elem*  
lv\_EMPSAL TYPE  ZEMPSAL,  
lv\_EMPADDRS TYPE  ZEMPADDRS.  
  
select EMPID EMPNAME EMPSAL EMPADDRS  
       into ( lv\_empid, lv\_empname , lv\_empsal , lv\_empaddrs )  
       from ZNAGESH\_TABLE  
  
Where EMPID = '222'.  
  WRITE : / lv\_EMPID, lv\_EMPNAME , lv\_EMPSAL , lv\_EMPADDRS.  
  
  ENDSELECT.  
  
  
  
 **2. WITH parameter :  Taking the input values at runtime**   
  
data:  
lv\_EMPID TYPE  ZEMPID,  *"" assigned varuables as corresponded table coloumns*  
lv\_EMPNAME TYPE  ZEMPNAME,    *" either table name- column name or Data element*  
lv\_EMPSAL TYPE  ZEMPSAL,  
lv\_EMPADDRS TYPE  ZEMPADDRS.  
PARAMETERS p\_a type ZEMPID.       
select EMPID EMPNAME EMPSAL EMPADDRS  
       into ( lv\_empid, lv\_empname , lv\_empsal , lv\_empaddrs )  
       from ZNAGESH\_TABLE  
  
Where EMPID = p\_a.  
  WRITE : / lv\_EMPID, lv\_EMPNAME , lv\_EMPSAL , lv\_EMPADDRS.  
  
  ENDSELECT.  
  
 **3.Range between values:**  
  
data:  
lv\_EMPID TYPE  ZEMPID,  *"" assigned varuables as corresponded table coloumns*  
lv\_EMPNAME TYPE  ZEMPNAME,    *" either table name- column name or Data element*  
lv\_EMPSAL TYPE  ZEMPSAL,  
lv\_EMPADDRS TYPE  ZEMPADDRS.  
SELECT-OPTIONS s\_a for lv\_empid.  *""if we want to range as input*  
  
select EMPID EMPNAME EMPSAL EMPADDRS  
       into ( lv\_empid, lv\_empname , lv\_empsal , lv\_empaddrs )  
       from ZNAGESH\_TABLE  
  
Where EMPID IN s\_a.  *"" selection variable*  
  WRITE : / lv\_EMPID, lv\_EMPNAME , lv\_EMPSAL , lv\_EMPADDRS.  
  
  ENDSELECT.  
  
  
 **"4.selection and parameters**  
  
  data:  
lv\_EMPID TYPE  ZEMPID,  *"" assigned varuables as corresponded table coloumns*  
lv\_EMPNAME TYPE  ZEMPNAME,    *" either table name- column name or Data element*  
lv\_EMPSAL TYPE  ZEMPSAL,  
lv\_EMPADDRS TYPE  ZEMPADDRS.  
SELECT-OPTIONS s\_a for lv\_empid.    *""if we want to range as input*  
PARAMETERS p\_a type ZEMPSAL.  
  
select EMPID EMPNAME EMPSAL EMPADDRS  
       into ( lv\_empid, lv\_empname , lv\_empsal , lv\_empaddrs )  
       from ZNAGESH\_TABLE  
  
Where EMPID IN s\_a and EMPSAL EQ p\_a.    *""*  
  *"" selection variable*  
  WRITE : / lv\_EMPID, lv\_EMPNAME , lv\_EMPSAL , lv\_EMPADDRS.  
  
  ENDSELECT.

**6.Internal Tables**

REPORT znagesh\_itab.  *"Internal Tables*  
  
  
  
**1.With Header**  
  
  data: lt\_tab type ZBNAGESH\_TAB occurs 0 WITH HEADER LINE,  
        lv\_emid type ZEMID.  
  
  SELECT-OPTIONS s\_a for lv\_emid.    *""if we want to range as input*  
 *\*PARAMETERS p\_a type ZEMPSAL.*  
  
select \*  
       into lt\_tab *"into header*  
       from ZBNAGESH\_TAB  
       Where EMID IN s\_a. *""and EMPSAL EQ p\_a.*  
  Append lt\_tab to lt\_tab. *"h to b*  
  endselect.  
  
  loop at lt\_tab into lt\_tab. *"b to h*  
  
  WRITE : / lt\_tab-EMID, lt\_tab-EMNAME , lt\_tab-EMLOC , lt\_tab-EMSAL, lt\_tab-EMDESTG.  
  
 endloop.

**2. Without Append And Endselect:**  
  
  data: lt\_tab type ZBNAGESH\_TAB occurs 0 WITH HEADER LINE,  *"" click lt\_tab and lv\_emid for debugging*  
        lv\_emid type ZEMID.  
  
  SELECT-OPTIONS s\_a for lv\_emid.    *""if we want to range as input*  
*\*PARAMETERS p\_a type ZEMPSAL.*  
  
select \*  
       into table lt\_tab *"directly enter into internal table*  
       from ZBNAGESH\_TAB  
       Where EMID IN s\_a. *""and EMPSAL EQ p\_a.*  
  *""Append lt\_tab to lt\_tab. "h to b*  
  *""endselect.*  
  
  loop at lt\_tab into lt\_tab. *"b to h*  
  
  WRITE : / lt\_tab-EMID, lt\_tab-EMNAME , lt\_tab-EMLOC , lt\_tab-EMSAL, lt\_tab-EMDESTG.  
  
 endloop.  
 **3. Without Header And With Strucure And Work Area**

TYPES : BEGIN OF ls\_tab,  
          lv\_EMID    TYPE  zemid,  
          lv\_EMNAME  TYPE  zemname,  
          lv\_EMloc   TYPE  zemploc,  
          lv\_EMSAL   TYPE zemsal,  
          lv\_EMDESTG TYPE zemdestg,  
        END OF ls\_tab.  
  
DATA: lt\_tab  TYPE STANDARD TABLE OF ls\_tab, *"" creating internal table by  reference local structure*  
      wa\_tab  TYPE LS\_tAB, *"" create the wa by rfefering the structure*  
      lv\_emid TYPE zemid.  
  
SELECT-OPTIONS s\_a FOR lv\_emid.    *""if we want to range as input*  
PARAMETERS p\_a TYPE zemsal.  
  
SELECT emid  
emname  
emloc  
emsal  
  emdestg  
       INTO TABLE lt\_tab *"directly enter into internal table*  
       FROM zbnagesh\_tab  
       WHERE emid IN s\_a. *""and EMPSAL EQ p\_a.*  
*"Append lt\_tab to lt\_tab. "h to b*  
*"endselect.*  
  
LOOP AT lt\_tab INTO wa\_tab. *"b to h*  
  
  WRITE : / wa\_tab-lv\_EMID, wa\_tab-lv\_EMNAME , wa\_tab-lv\_EMloc , wa\_tab-lv\_EMsal, wa\_tab-lv\_EMDESTG.  
  
ENDLOOP.

**7.Views**

**1.DataBase View:**

**A screenshot of a computer

Description automatically generated**

1. **Selection Condition**

A screenshot of a computer

Description automatically generated

**(b)Output:**

**A screenshot of a computer

Description automatically generated**

**Table 2:**

**QPMK - QPNMZ**

**A screenshot of a computer

Description automatically generated**

**Output :**

**A screenshot of a computer

Description automatically generated**

**View Calling Program:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**2.Projection View:**

*A screenshot of a computer

Description automatically generated*

**Output:**

*A screenshot of a computer

Description automatically generated*

**3.Maintenance View:**

A screenshot of a computer

Description automatically generated

**Output:**

A screenshot of a computer

Description automatically generated

**8.JOINS**

REPORT ZNAGESH\_UST\_JOINS.

&---------------------------------------------------------------------\*

\*1.Program:

TYPES: BEGIN OF ls\_tab, ""Fields of VBAK

vbeln TYPE vbeln\_va,

audat TYPE audat,

vbtyp TYPE vbtypl,

trvog TYPE trvog,

auart TYPE auart,

END OF ls\_tab.

DATA : lt\_tab TYPE STANDARD TABLE OF ls\_tab, ""These two are referred to local structure

wa\_tab TYPE ls\_tab, ""Work area stores single record

lv\_vbeln type vbeln.

SELECT-OPTIONS s\_vbeln FOR lv\_vbeln. "" to enter from the value 1 - up to range like 100 or etc.,

START-OF-SELECTION.

SELECT vbeln audat vbtyp trvog auart

INTO TABLE lt\_tab

FROM vbak

WHERE vbeln IN s\_vbeln.

IF sy-subrc EQ 0. ""success rc means record

WRITE : 'sales order info ', sy-dbcnt.

ELSE.

WRITE : 'sales order records are not found ', sy-dbcnt.

ENDIF.

END-OF-SELECTION.

LOOP AT lt\_tab INTO wa\_tab."" body to header

WRITE : / wa\_tab-vbeln,

wa\_tab-audat,

wa\_tab-vbtyp,

wa\_tab-trvog,

wa\_tab-auart.

ENDLOOP.

**Output:**

A screenshot of a computer

Description automatically generated

**\* 2 Two Tables For Inner Join**

TYPES: BEGIN OF ls\_tab,

vbeln TYPE vbeln\_va, " vbak

audat TYPE audat,

vbtyp TYPE vbtypl,

trvog TYPE trvog,

auart TYPE auart,

posnr TYPE posnr\_va, " vbap

matnr TYPE matnr,

matwa TYPE MATWA,

END OF ls\_tab.

DATA : lt\_tab TYPE STANDARD TABLE OF ls\_tab,

wa\_tab TYPE ls\_tab.

SELECT-OPTIONS s\_vbeln FOR wa\_tab-vbeln.

START-OF-SELECTION.

SELECT vbak~vbeln

vbak~audat

vbak~vbtyp

vbak~trvog

vbak~auart

vbap~posnr

vbap~matnr

vbap~matwa

INTO TABLE lt\_tab ""storing into internal table

FROM vbak

INNER JOIN vbap

ON vbak~vbeln = vbap~vbeln ""The columns which are common from two tables will display in output upto limit we specified

WHERE vbak~vbeln IN s\_vbeln. ""range upto

IF sy-subrc EQ 0.

WRITE : 'sales order info ', sy-dbcnt.

ELSE.

WRITE : 'sales order records are not found ', sy-dbcnt.

ENDIF.

END-OF-SELECTION.

LOOP AT lt\_tab INTO wa\_tab.

WRITE : / wa\_tab-vbeln,

wa\_tab-posnr,

wa\_tab-matnr,

wa\_tab-audat,

wa\_tab-vbtyp,

wa\_tab-trvog,

wa\_tab-auart,

wa\_tab-matwa.

ENDLOOP.

OUTPUT:



**3.program (2 two tables with alias)**

TYPES: BEGIN OF ls\_tab,

vbeln TYPE vbeln\_va, " vbak

audat TYPE audat,

vbtyp TYPE vbtypl,

trvog TYPE trvog,

auart TYPE auart,

posnr TYPE posnr\_va, " vbap

matnr TYPE matnr,

matwa TYPE matwa,

END OF ls\_tab.

DATA : lt\_tab TYPE STANDARD TABLE OF ls\_tab,

wa\_tab TYPE ls\_tab.

SELECT-OPTIONS s\_vbeln FOR wa\_tab-vbeln.

START-OF-SELECTION.

SELECT so\_h~vbeln

so\_h~audat

so\_h~vbtyp

so\_h~trvog

so\_h~auart

so\_i~posnr

so\_i~matnr

so\_i~matwa

INTO TABLE lt\_tab

FROM vbak AS so\_h

INNER JOIN vbap AS so\_i

ON so\_h~vbeln = so\_i~vbeln

WHERE so\_h~vbeln IN s\_vbeln.

IF sy-subrc EQ 0.

WRITE : 'sales order info ', sy-dbcnt.

ELSE.

WRITE : 'sales order records are not found ', sy-dbcnt.

ENDIF.

END-OF-SELECTION.

LOOP AT lt\_tab INTO wa\_tab.

WRITE : / wa\_tab-vbeln,

wa\_tab-posnr,

wa\_tab-matnr,

wa\_tab-audat,

wa\_tab-vbtyp,

wa\_tab-trvog,

wa\_tab-auart,

wa\_tab-matwa.

ENDLOOP.

OUTPUT:

A screenshot of a computer

Description automatically generated

**\* 4 Left Outer join (2 two tables with alias)**

TYPES: BEGIN OF ls\_tab,

vbeln TYPE vbeln\_va, " vbak

audat TYPE audat,

vbtyp TYPE vbtypl,

trvog TYPE trvog,

auart TYPE auart,

posnr TYPE posnr\_va, " vbap

matnr TYPE matnr,

matwa TYPE matwa,

END OF ls\_tab.

DATA : lt\_tab TYPE STANDARD TABLE OF ls\_tab,

wa\_tab TYPE ls\_tab.

SELECT-OPTIONS s\_vbeln FOR wa\_tab-vbeln.

START-OF-SELECTION.

SELECT so\_h~vbeln

so\_h~audat

so\_h~vbtyp

so\_h~trvog

so\_h~auart

so\_i~posnr

so\_i~matnr

so\_i~matwa

INTO TABLE lt\_tab

FROM vbak AS so\_h

LEFT OUTER JOIN vbap AS so\_i

ON so\_h~vbeln = so\_i~vbeln

WHERE so\_h~vbeln IN s\_vbeln.

IF sy-subrc EQ 0.

WRITE : 'sales order info ', sy-dbcnt.

ELSE.

WRITE : 'sales order records are not found ', sy-dbcnt.

ENDIF.

END-OF-SELECTION.

LOOP AT lt\_tab INTO wa\_tab.

WRITE : / wa\_tab-vbeln,

wa\_tab-posnr,

wa\_tab-matnr,

wa\_tab-audat,

wa\_tab-vbtyp,

wa\_tab-trvog,

wa\_tab-auart,

wa\_tab-matwa.

ENDLOOP.

OUTPUT:

A screenshot of a computer

Description automatically generated