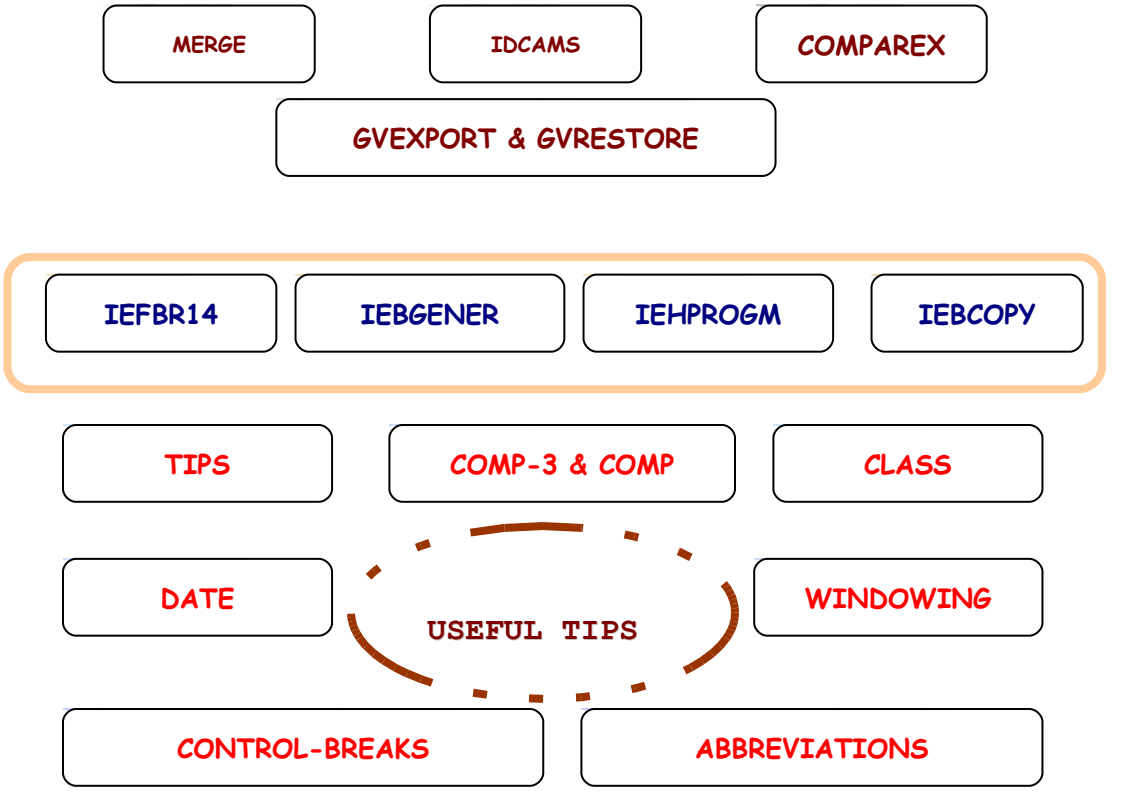
UTILITIES



TERMS

SORT

The SORT program is used to sort data into a certain sequence or to merge from 2 to 100 previously sorted input data sets into 1 output data set.

```
//STEP10     EXEC PGM=SORT
//SYSOUT     DD SYSOUT=*
```

```
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SORTIN    DD DISP=SHR,DSN=<< INPUT FILE NAME >>
//SORTOUT   DD DISP=(MOD,CATLG,DELETE),SPACE=(CYL,(5,5),RLSE),
//             UNIT=SYSDA,DCB=*.SORTIN,
//             DSN=<< OUTPUT FILE NAME >>
//SYSIN     DD *
//**************   INCLUDE SORT STEPS HERE
```

1) TO SORT ON POSITIONS say for eg. 1 to 7

```
 SORT FIELDS=(1,7,CH,A)
```

Where

Sort fields = (position ,length ,format ,sequence) or
Sort fields = (position ,length , sequence..),format =
CH/BI/PD/ZD.d
PD=packed Decimal(COMP-3), ZD=zone decimal.

NOTE :-

Instead of using JCL to perform SORT operation , there's one simple alternative,
For eg:- Open a Flat file in edit mode. On the command line type (say) SORT 1,7 and press ENTER, the file will be sorted on positions 1 to 7 bytes.

2) TO COPY ALL THE RECORDS FROM INPUT FILE TO OUTPUT FILE

```
 SORT FIELDS=COPY
```

3) TO COPY THOSE RECORDS WHICH SATISFY A PARTICULAR CONDITION.

```
 INCLUDE COND=(38,10,CH,EQ,C'57071509',OR,36,10,CH,EQ,C'
 57105779')
```

4) TO OMIT THOSE RECORDS WHICH SATISFY A PARTICULAR CONDITION.

```
 OMIT COND=(19,1,CH,EQ,C'S',OR,19,1,CH,EQ,C'S')
```

5) TO SKIP CERTAIN NO OF RECORDS

```
 SORT FIELDS=COPY,SKIPREC=1000
```

6) TO STOP AFTER COPYING CERTAIN NO OF RECORDS

```
 SORT FIELDS=COPY,STOPAFT=5000
```

7) SKIPREC AND STOPREC CAN BE USED IN COMBINATION

```
 SORT FIELDS=COPY,SKIPREC=1000,STOPAFT=5000
```

8) TO REMOVE DUPLICATES FROM THE FILE USING SORT

```
SORT FIELDS=(1,7,A),FORMAT=CH
SUM FIELDS=NONE
```

BACK

MERGE

The MERGE control statement defines the application as a MERGE application.

```
MERGE FIELDS=...
             {,FILES=n}
             {,EQUALS | NOEQUALS}
             {,CKPT | CHKPT}
             {,CENTWIN={0 | s | f}}
```

where
FIELDS=(pos1,len1,opt1,pos2,len2,opt2,...),**FORMAT**=type

Keyword explanations:

The **FIELDS**= keyword is used to identify the fields to use as merge keys. Each field is described using 4 values:

'pos', its position in the record, relative to 1;
'len', the field's length;
'type', the type of data stored in the field; and

```
  'opt', the sort order for the field which can be
        A for ascending, D for descending, or
        E as modified by an E61 exit.
Up to 128 fields can be sorted or merged using one
MERGE control statement.
```

BACK

IDCAMS

IDCAMS Return Codes

0	Command executed with no errors
4	Warning - execution may go successful
8	Serious error - execution may fail.
12	Serious error - execution impossible.
16	Fatal error - job step terminates

Defining ESDS cluster:(entry sequenced dataset)

```
 DEFINE CLUSTER (NAME(PUFAP.VSAM.APFT100)  -
                 CYL(50 50)                -
                 RECORDSIZE(814 814)       -
                 VOLUME(* * * * *)         -
                 REUSE                     -
                 NONINDEXED                -
        DATA     (NAME(PUFAP.VSAM.APFT100.DATA)  -
                 CISZ(4096))
```

Defining KSDS cluster:(KEY sequenced dataset)

```
DELETE  PUFAP.VSAM.APFT100           -
         CLUSTER                     -
         PURGE

 DEFINE CLUSTER (NAME(PUFAP.VSAM.APFT100)    -
                 CYL(50 50)                   -
                 KEYS(48 0)                   -
                 RECORDSIZE(814 814)         -
                 VOLUME(* * * * *)            -
                 SHAREOPTIONS(2 3)            -
```

```
              SPEED                          -
              REUSE                          -
              INDEXED                        -
              FREESPACE(5 5))                -
       INDEX   (NAME(PUFAP.VSAM.APFT100.INDEX) -
              CISZ(512))                     -
        DATA    (NAME(PUFAP.VSAM.APFT100.DATA)  -
              CISZ(4096))
```

SOME DEFINITIONS.

keys :
keys(length offset)
e.g. key(8 1) starting from 2nd byte to 9th byte

spanned
it allows record to span more than one control interval

Dataset type

indexed	(for ksds)	key sequenced dataset
nonindexed	(for esds)	entry sequenced dataset
numbered	(for rrds)	relative record dataset

freespace
it applies to ksds.it can be used for adding new records or expanding existing variable records.

space parameter:
cylinders(primary secondary)
tracks (primary secondary)
records (primary secondary)
kilobytes(primary secondary)
megabytes(primary secondary)

reuse
reuse specifies that cluster can be opened next time as a reusable cluster.*if it is opened in output mode it is treated as empty dataset*.

share options (cr-value cs-value)
values: 1 multiple read or single write
2 multiple read and single write
3 multiple read and multiple write

cr value: specifies value for cross region sharing.
cross region sharing is defined as different jobs running on the same system using global resource serialization,a resource control facility.

cs value: specifies the value for cross system sharing
means different jobs running on different system in a nongrs environment.

Listcat:
helps to view password and security information, usage statistics, space allocation info, creation and expiration dates etc

NOTE:-The following attributes are unalterable. You have to DELETE the cluster and redefine it with new attributes.

- CISZ
- Cluster type,
- IMBED/REPLICATE
- REUSE | NOREUSE

BACK

GVEXPORT AND GVRESTORE

Using these utilities multiple files before updation are backed up. **Faver** compresses all the input files. Using GVRESTOR the files are again uncompressed.

GVEXPORT

```
//JS01     EXEC  PGM=GVEXPORT
//SYSPRINT DD  SYSOUT=*
//SNAPDD   DD  SYSOUT=*
//DD01     DD DSN=OUFAP.PROD.CC9601.CCFM100,DISP=OLD
//DD02     DD DSN=OUFAP.PROD.CC9601.CCFM200,DISP=OLD
//FVROUT0  DD  DSN=&LVL.UFAP.FAVER.BKUP(+1),
//             DISP=(,CATLG,DELETE),
//             UNIT=ACART,
//             DCB=SYS2.DSCB
//SYSIN    DD  DSN=PAEPC.Y2K.SYSIN(APUF8201),
//             DISP=SHR
  EXPORT
  CLUSTER
  CL=OUFAP.PROD.CC9601.CCFM100
  CL=OUFAP.PROD.CC9601.CCFM200
```

GVRESTOR

```
//JS02     EXEC  PGM=GVRESTOR
//SYSPRINT DD  SYSOUT=*
//*
//SNAPDD   DD  SYSOUT=*
//*
//FVRIN0   DD  DSN=&LVL.UFAP.FAVER.BKUP(+1),
//             DISP=OLD
//SYSIN    DD  DSN=PAEPC.Y2K.SYSIN(APUF8101),
//             DISP=SHR
```

```
  RESTORE PURGE
  CLUSTER
  CL=OUFAP.PROD.CC9601.CCFM100
  CL=OUFAP.PROD.CC9601.CCFM200
```

Note: It delete defines the cluster and then restores it from back up.

BACK

REPRO

- Loads empty VSAM cluster with records.
- Creates backup of datasets
- Merges data from two VSAM datasets
- Can operate on Non-Vsam datasets
- Can copy from KSDS to ESDS
- In case of KSDS ,data & index component are build automatically

```
//STEP10  EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//DD1      DD DSN=PCEX.D300P010.VASW208.G0026V00,
//          DISP=SHR
//DD2      DD DSN=CPI206.CEX.D300P010.VASW208.G0026,
//          DISP=(MOD,CATLG,DELETE),
//          SPACE=(CYL,(10,20),RLSE),
//          RECFM=VB,LRECL=32604,BLKSIZE=32608
//SYSIN    DD  *
     REPRO INFILE(DD1) OUTFILE(DD2)
//
```

LIMITING INPUT AND OUTPUT RECORDS

Using **FROMKEY** AND **TOKEY**

```
REPRO            -
INFILE(DD1)      -
OUTFILE(DD2)     -
FROMKEY(A001)  -
TOKEY(A045)
```

Using **SKIP** AND **COUNT**

```
REPRO          -
INFILE(DD1)    -
OUTFILE(DD2)   -
SKIP(50)       -
COUNT(1000)
```

This example SKIPS **50** Records and copies next **1000** records.

BACK

COMPAREX

Similar to option 3.13 , i.e. compares two files.

COMPAREX allows you to restrict the compare to certain fields within each record, or to ignore certain fields.
FIELDs are used to specify which fields are to be compared.
MASKS,are used to specify which fields are **NOT** to be compared.
For example:

```
F=FIELD   --------FIELD ONE---------   ----FIELD TWO---------
M=MASK    DISPLACEMENT LENGTH FORMAT   DISP  LEN   FORMAT
   f        5           106
   f        111         4       p      112    3     b
   m        33          7       z

  Generates: FIELD=(5,106,C)
             FIELD1=(111,4,P) FIELD2=(112,3,B)
             MASK=(33,7,Z)
```

For example:

```
//JS30     EXEC PGM=COMPAREX
//SYSPRINT  DD SYSOUT=*
//SYSUT1    DD DISP=SHR,
//             DSN=TCEX.Q133P020.F02A.PREMOUT
//SYSUT2    DD DISP=SHR,
//             DSN=TCEX.Q133P020.F02A.PREMOUT.TEST
//SYSIN     DD *
 FORMAT=13
/*
```

COMPAREX USING MASK COMMAND
i.e.

```
//SYSIN     DD *
 FORMAT=13
 MASK=(271,20,C)
```

where
MASK= (Position, Length and Type)
For example MASK=(271,20,C) means, Do **NOT** compare the data from position 271 + 20 Characters (C stands for characters).

And where Format equals :-

FORMAT - xy specifies DATA formatting characteristics in how differences are displayed.
Two numerics x and y

where x equals

X	Equals
0	0-dump format
1	alphanumeric line
2	DITTO format (vertical hex)

& where y equals

Y	Equals
1	full display of SYSUT1 followed by full SYSUT2
2	full display of SYSUT1 followed by differing lines of SYSUT2
3	differing lines of SYSUT1 followed by differing lines of SYSUT2
4	full display of SYSUT1 interleaved with full display of SYSUT2
5	full display of SYSUT1 interleaved with differing lines of SYSUT2
6	differing lines of SYSUT1 interleaved with differing lines of SYSUT2

Note :See that I have used Format = 13 in the above example.

BACK

IEFBR14

Every time during Testing we encounter errors due to which we have to make modifications and rerun the job. It has been a regular practice that we always tend to do TSO DEL 'File name' and never include a simple **IEFBR14** step.
Now all you have to do is just Copy and Paste the code into your JCL and proceed ahead without any warnings like "FILE IS ALREADY CALALOGED"

```
//*************************************************************//
*  IEFBR14
//*************************************************************//
DD1      EXEC  PGM=IEFBR14
//DELDD    DD DSN=<< input file name >>,
//          DISP=(MOD,DELETE,DELETE),UNIT=SYSDA,
//          SPACE=(TRK,(1,1))
//*************************************************
```

IEFBR14 is also used to delete Temporary Work files

```
//*************************************************************
//*  IEFBR14 - DELETE WORK FILES
//*************************************************************
//PS160  EXEC PGM=IEFBR14
//DD1      DD DSN=TCEX.WORK.A186P010.VALPTEMP,
//            DISP=(OLD,DELETE)
//DD2      DD DSN=TCEX.WORK.A186P020.SRT,
//            DISP=(OLD,DELETE)
//DD3      DD DSN=TCEX.WORK.A186P040.VALPTEMP,
//            DISP=(OLD,DELETE)
*****************************************************************************
```

IEFBR14 is also used to delete Files that are on TAPE

```
//*************************************************
//*  IEFBR14 - *DELETE GDG BASED FILES *
//*************************************************
//PS500   EXEC PGM=IEFBR14
```

```
//DD1        DD DSN=TCEX.Q213P110.F03A.G0003V00,
//              DISP=(OLD,DELETE)
//DD2        DD DSN=TCEX.Q213P110.F03A.G0004V00,
//              DISP=(OLD,DELETE)
//DD3        DD DSN=TCEX.Q213P110.F03A.G0005V00,
//              DISP=(OLD,DELETE)
//DD4        DD DSN=TCEX.Q213P110.F03A.G0006V00,
//              DISP=(OLD,DELETE)
```

BACK

IEBCOPY

IEBCOPY is used to copy all or part of a Partitioned Data Set (PDS) . Selected members of a PDS can be copied to another or the same PDS and/or renamed. A sequential backup copy of a PDS can be made. IEBCOPY is used to "compress" a PDS when all of its unused internal space has been exhausted. The compress operation reorganizes a PDS so that all previously unused space inside the PDS is reclaimed.

Sample IEBCOPY JCL:

```
//JS10    EXEC PGM=IEBCOPY,REGION=1024K,
//             PARM='SIZE=nnnnnnnnK'                 Optional PARM
//SYSPRINT  DD SYSOUT=*                              IEBCOPY Messages
//ddname1   DD DSN=...,DISP=...                      Input File
//ddname2   DD DSN=...,DISP=...                      Output File
//SYSUT3    DD UNIT=SYSDA,SPACE=(TRK,(30,30),RLSE)   Work file 1
//SYSUT4    DD UNIT=SYSDA,SPACE=(TRK,(30,30),RLSE)   Work file 2
//SYSIN     DD *                                     Control Statements
 control statements...
/*
```

Valid control statements are

1) COPY
2) SELECT
3) EXCLUDE

COPY:- COPY:

This statement indicates the beginning of a copy operation and Identifies the DD statements to be used during the copy. The format of the COPY control statement is:

Format:

```
{label} COPY OUTDD=ddname,   (OUTPUT FILENAME)
```

```
          INDD=(ddname1,ddname2,(ddname3,R),...)
                    (INPUT FILENAME)
          {,LIST=NO}
   --------------------------------
```

The LIST=NO keyword is optional and tells IEBCOPY that you don't want a list of the members in the PDS.

COPY can be abbreviated as **'C'**, OUTDD as **'O'**, and INDD as **'I'**.

Note : When copying from a sequential file or a PDS to another PDS, specify the **'R'** parameter after the input DD name if you want ALL identically named members replaced on the output file.

Examples.

Identically named members are only replaced on a copy operation
if you request the REPLACE option on the COPY statement, or on
the SELECT statement, described later.
COPY Statement EXAMPLES follow:

Example 1 - Copy all with replace.

```
 {label} COPY OUTDD=O,INDD=((I,R))
--------------------------------
```

Example 2 - Copy without replace.

```
 {label} C O=TAPE,I=DASD
--------------------------------
```

Example 3 - Compress-in-place!

```
{label} COPY OUTDD=SYSUT1,I=SYSUT1
--------------------------------
```

SELECT:

The SELECT statement is used to name members to be included in a copy operation. The SELECT statement must be preceded by a COPY or COPYMOD statement, or the INDD= portion of a COPY statement. A SELECT statement may not appear in the same COPY operation as an EXCLUDE statement, neither can SELECT be used in a compress operation. A SELECT member is only replaced in the output data set if the REPLACE option ('R') is set on the SELECT statement or on the INDD portion of the COPY statement. Possible formats of the SELECT control statement are:

Format 1 - Copy selected members.

```
    {label} SELECT MEMBER=name
```

```
-  ------------------------------
```

Format 2 - Copy a list of members.

```
{label} SELECT MEMBER=(name1,name2,name3...)
------------------------------
```

Format 3 - Copy a list of members and rename them.

```
{label} SELECT
MEMBER=((name1,newname1),(name2,newname2),...)
------------------------------
```

Format 4 - Copy a list of members and replace them if they are already in the output data set.

```
{label} SELECT MEMBER=((name1,,R),(name2,,R),...)
------------------------------
```

EXCLUDE:

The EXCLUDE statement is used to name members to be excluded from A copy operation. The EXCLUDE statement must be preceded by a COPY or COPYMOD statement, or the INDD= portion of a COPY statement. An EXCLUDE statement may not appear in the same COPY operation as a SELECT statement, neither can EXCLUDE be used in a compress operation. The format of the SELECT control statement is:

Format -

```
{label} EXCLUDE MEMBER=(name1,name2,name3,...)
```

IEBCOPY Usage Examples:

JCL to compress a PDS:

```
//JS10    EXEC PGM=IEBCOPY,REGION=1M
//SYSPRINT  DD SYSOUT=*
//I1        DD DSN=my.pds,     same PDS for I1 & O1
//             DISP=OLD
//*
//O1        DD DSN=my.pds,
//             DISP=OLD
//SYSIN     DD *
COMP1 C O=O1,I=((I1,R))
```

OR

```
//COMPRESS EXEC  PGM=IEBCOPY,REGION=0K
//SYSPRINT DD  SYSOUT=*
//PDSIN    DD  DSN=PUFAP.PARMLIB.CYCLE,DISP=SHR
//PDSOUT   DD  DSN=PUFAP.PARMLIB.CYCLE,DISP=OLD
//SYSIN    DD  DSN=PAEPC.Y2K.SYSIN(APUF00D1),DISP=SHR
     COPY INDD=PDSIN,OUTDD=PDSOUT
```

JCL to unload a PDS to a tape:

```
//STEP1   EXEC PGM=IEBCOPY,REGION=1024K
//SYSPRINT  DD SYSOUT=*
//I1        DD DSN=my.pds,              PDS to unload
//             DISP=OLD
//*
//O1        DD DSN=my.pds.tape.copy,  tape to unload PDS
//             DISP=(,CATLG),
//             UNIT=TAPE,
//             VOL=SER=
//SYSIN     DD *
COPY1 C O=O1,I=((I1,R))
```

JCL to load a PDS to DASD from a sequential unloaded copy:

```
  //PDSLOAD EXEC PGM=IEBCOPY,REGION=1M
  //SYSPRINT  DD SYSOUT=*
  //I1        DD DSN=my.pds.seq.copy,DISP=OLD
  //*                      previously unloaded copy
  //*
  //O1        DD DSN=my.pds,
  //*                      PDS being created on DASD
  //             DISP=(,CATLG),
  //             UNIT=SYSDA,
  //             SPACE=(TRK,(30,30,10),RLSE)
  //SYSIN     DD *
  COPY1 C O=O1,I=((I1,R))
```

JCL to copy 4 members from one PDS to another:

```
//PDSCOPY EXEC PGM=IEBCOPY,REGION=1024K
//SYSPRINT  DD SYSOUT=*
//I1        DD DSN=my.pds.input,        copy from here
//             DISP=SHR
//*
//O1        DD DSN=my.pds.output,       to here
//             DISP=SHR
//SYSIN     DD *
COPY1 C O=O1,I=((I1,R))
SELC1 S M=MEMBER1,MEMBER2
SELC2 S M=((MEMBER3,NEWMEM3),MEMBER4)   rename MEMBER3 to NEWMEM3
```

BACK

IEBGENER

This utility is to copy, concatenate and to empty sequential datasets:-

Example for **Copy**:

```
//*-------------------------------------------------
//step01   EXEC  PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=&LEVL1.CCL.NN1.CCLC120(+1),
//             DISP=SHR
//SYSUT2   DD  DSN=&LEVL1.CCL.NN1.CCLC120.FTP,
//             DISP=MOD
//SYSIN    DD  DUMMY
//*-------------------------------------------------
```

Example for **Concatenation**:

```
//*-------------------------------------------------
//step01   EXEC  PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DSN=&LEVL1.CCL.NN1.CCLC120(+1),
//             DISP=SHR
//         DD  DSN=&LEVL1.CCL.NN1.CCLC121(+1),
//             DISP=SHR
//SYSUT2   DD  DSN=&LEVL1.CCL.NN1.CCLC120.FTP,
//             DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(CYL,(10,5),RLSE)
//*-------------------------------------------------
```

Example to **Empty Existing Data.**

```
//*-------------------------------------------------
//step01   EXEC  PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DUMMY,DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//SYSUT2   DD  DSN=&LEVL1.CCL.NN1.CCLC120.FTP,              //
DISP=SHR
//SYSIN    DD  DUMMY
//*-------------------------------------------------
```

BACK

IEHPROGM

IEHPROGM is used to maintain data sets and system control data.
The IEHPROGM utility can be used to:

1) Either scratch a data set or PDS/PDSE members
2) Either rename a data set or PDS/PDSE members
3) Change the OS CVOL for a non-VSAM data set
   through cataloging or uncataloging entries, building
   or deleting indexes or aliases, or creating and
   manipulating GDG indexes
4) Two OS CVOLs can be connected or released
5) Data set passwords maintenance

Please refer QW for getting Detail Information, Syntax about IEHPROGM

BACK

FILE-AID

```
//*UIDFAID JOB (UFLI-UFLPROD,S062,AAESPX00),SRINI,
//        USER=*UID,PASSWORD=*PSW, MSGCLASS=X,
//        CLASS=B,TIME=4,
//        NOTIFY=*UID
//*
//*=====================================================//* THIS IS A
SHELL FOR FILEAID                                        //*
//*=====================================================//STEP01
EXEC PGM=FILEAID                              <==FAID STEP
//*-----------------------------------------------------//* FOR VB
//*    IF ACCESSING DASD USE EXACT POSITION PLUS 4
//*    IF ACCESSING TAPE USE EXACT POSITION PLUS 4
//*-----------------------------------------------------//SYSPRINT DD
SYSOUT=*                                    <==SYSPRINT
//SYSUDUMP DD SYSOUT=*                                  <==SYSUDUMP
//SYSLIST  DD SYSOUT=*                                  <==SYSLIST
//SYSTOTAL DD SYSOUT=*                                  <==SYSTOTAL
//DD01     DD DSN=XXXXXXXXXXX                           <==INPUT FILE
//*
//DD01O    DD DSN=XXXXXXXXXXX,                          <==OUTPUT FILE
//            DISP=(,CATLG,DELETE),
//            UNIT=SYSDA,SPACE=(TRK,(30,10),RLSE),
//            DCB=(LRECL=XXXXX,RECFM=XB,BLKSIZE=XXXXX)
//SYSIN    DD   *                                       <==PARMS
```

```
 $$DD01 COPY IF=(1,EQ,C'XXXXXX')
/*
//
//*=====================================================
FUNCTIONS
COPY
DROP
DUMP
LIST
PRINT
SPACE
TALLY
UPDATE
USER
//*=====================================================MODIFIERS
ALL - PROCESSES ENTIRE FILE - REGARDLESS OF SELECTION CRITERIA
BACK- PROCESSES FILE BACKWARDS
MEM - PROCESSES PDS
//*===================================================== PARM LIST
EXAMPLES
```

CLICK HERE TO VIEW FILE-AID EXAMPLES

BACK

USEFUL TIPS

1) Different ways for coding **DCB** parameter:-

(a) DCB=*.ddname e.g. :- (DCB=*.DD1)
While performing operations like SORT, REPRO, FILE-AID, (wherein records are to be copied from an input file to an output file.)
The DCB parameter for the output file for SORT can be coded as
For eg:-

DCB=*.SORTIN (where SORTIN is the DD name for the input file for SORT)

(b) DCB=*.stepname.ddname
Requests that the DCB parameter be copied from the DD statement "ddname" found in the same step "stepname"
DCB=*.STEP2.DD1

(c) DCB=*.procexec.stepname.ddname

Requests that the DCB parameter be copied from DD statement "ddname" found in the previous step "stepname" found within a procedure "procexec"
(name of EXEC statement invoking the procedure.)

DCB=*.PR1.STEP2.DD1

2) Always use **RLSE** parameter, while specifying the SPACE Parameter. This parameter releases the unwanted space. For eg:
SPACE=(CYL,(10,10),**RLSE**)

3) To know the properties of a GDG version residing on TAPE, like

1) **Creation Date**:-the day on which the particular GDG version was created.
2) **Birth Date**:- the day on which the base cluster was created.
3) **LRECL :-** Record Length.
4) **BLKSIZE :-** Block Size.
5) **RECFM :-** Record Format.
6) **CJOB**:- Name of the job which created this version.
7) **CDDNAME**:- Name of the DD statement where this file was created
8) **USERID**:- The UserID that submitted the job

NOTE :- CA1 can only be used for GDG versions which are on TAPE.
To get the information regarding Sequential Datasets (FLATFILE's), USE 3.4 and Type 'S' corresponding to the file for which you want to know the properties.
To get inormation regarding VSAM file use File-Aid.

PASSWORD FOR CA1 FOR SYSA - USERINQ AND TYPE 1 AT COMMAND OPTION
PASSWORD FOR CA1 FOR SYS7 - LOOK AND TYPE 1 AT COMMAND OPTION

4) To Block Purge Jobs in Q:ST :

Use // at the beginning of the block and put //P at the end of the block.

The jobs within the block will be purged.

5) To get information about the JOBS which are in EXECUTION

Go to Q;ST, type DA at Command Line.

The **Display Active Users (DA)** panel allows authorized users to display information about jobs, users, started tasks, and initiators that are active on the system.
It also shows system-wide data,such as CPU usage, paging rate, and start I/O rate.

6) Display user information :- Command is **'WHO'**

Purpose:, Displays the following information about the User: User ID, TSO logon procedure, terminal identification, and index number and name of the group in ISFPARMS. Also shows the levels of MVS, JES, ISPF, RMF and SDSF; the JES name, the SDSF server name and whether the server is in use.

7) OWNER : Limit jobs displayed by owning user ID.

Format: OWNER (ownerid|?)

Examples: OWNER CPI377 (with no other filtering in effect)

Displays only jobs for that owner.
OWNER * (with no other filtering in effect)
Displays all jobs for all owner IDs.

OWNER with no parameters displays all jobs.

If you know the JOB name then you can use the following command

PRE (JobName) ; owner (OwnerID)
Wildcards are also supported. i.e.

For e.g. PRE TCEX* ; owner CPI377
This command will display all the jobs starting with TCEX for the owner CPI377.

8) To Delete the job from Q;ST. we can use the 'P' (Purge) or 'C' (Cancel) command.

The Purge command will delete the job completely from Q;st
The Cancel command will retain the job information ie. The **JESMSGLOG,SYSMSG,JCL,SYSOUT** etc are retained.

9) Very often we copy jobs from Production Region to Test Region. If you want to know how much time the job takes to complete or execute, you can find out in **JESMSGLG** of the particular job. It gives you the StartTime and EndTime for the job.

10) In JCL, we encounter statement like **DSN=&&name;**
A simple name preceded by two ampersands identifies a temporary dataset. Temporary datasets are not retained beyond JOB termination. If the DSN name is omitted from a DD statement(except DD *, SYSOUT and DUMMY) also indicates a temporary dataset.

11) A VSAM cluster cannot be deleted by coding DISP= (OLD,DELETE) as it defaults to DISP=(OLD,KEEP)

12) If you don't want the source listing in the SYSPRINT (i.e. in the output) when an EASYTRIEVE program is executed, then just include the below mentioned lines in the Environment Section of the program. i.e. at the beginning of the program.

PARM LIST (NOPARM) +
DEBUG (NOXREF)
LIST OFF

13) Use

HRECALL :- To recall files which are migrated.

HDELETE :-. The HDELETE command is used to delete one or more migrated data sets from migration volumes. DFSMShsm deletes the data set without recalling it to a primary volume. When DFSMShsm deletes the data set, it maintains any backup versions of the data set.

HRECOVER :-

Recover a data set from a backup version or restore a data set from a dump copy

Note: Your data set may be restored from a dump copy rather than recovered from a backup version. This will happen if the data set was on a volume that was dumped by DFSMShsm more recently than it was backed up by DFSMShsm, and if the class(es) to which the volume was dumped allow the restore of a data set from a dump copy, and if you do not specify GENERATION or VERSION.

Specifying GENERATION or VERSION will cause the recover to be done from a backup version, not a dump copy. Specifying GENERATION(0) will insure that the recovery is done from the most recent available backup version.

Note: You cannot recover the backup versions of a cataloged data set that is currently migrated, as specified in the computing system catalog or the MCDS, until DFSMShsm recalls or deletes the migrated data set.

Syntax

```
  HRECOVER (dsname/password...)
           EXTENDRC
           FROMVOLUME(volid)
                GENERATION(gennum) or DATE(date) or
                VERSION(vernum)
           NEWNAME(newdsname/password)
           REPLACE
           TOVOLUME(volid) UNIT(unittype)
           WAIT or NOWAIT


 ALIAS - HRECOV
 REQUIRED - dsname, FROMVOLUME if data set was
            uncataloged at the time of backup.
 DEFAULTS - NOWAIT
```

Example 1

```
  HRECOVER CL.TEXT/WRITE FROMVOLUME(VOL003) +
           DATE(1984/01/05) NEWNAME(VER1TEXT.TEXT) +
           TOVOLUME(VOL001) UNIT(3330-1) WAIT
```

Example 2

```
  HRECOVER PARTSTST.CNTL TOVOLUME(VOL007) UNIT(3350)
           REPLACE
```

Example 3

```
  HRECOVER OUTTESTS.TESTLIST GENERATION(2) WAIT EXTENDRC
```

14) Always take into consideration 4 additional bytes while dealing with Variable Block Records.

BACK

Computational Items

A computational item is defined with one of the USAGE clause phrases described below. A computational item is a value used in arithmetic operations. It must be numeric.

If the USAGE of a group item is described with any of these items, the elementary items within the group have this usage.

The maximum length of a computational item is **18 decimal** digits.

The PICTURE of a computational item can contain only:

9 One or more numeric character positions
S One operational sign
V One implied decimal point
P One or more decimal scaling positions

COMPUTATIONAL-1 and COMPUTATIONAL-2 items (internal floating-point) cannot have PICTURE strings.

BINARY

Specified for binary data items. Such items have a decimal equivalent consisting of the decimal digits 0 through 9, plus a sign. Negative numbers are represented as the two's complement of the positive number with the same absolute value.

The amount of storage occupied by a binary item depends on the number of decimal digits defined in its PICTURE clause:

Digits in PICTURE Clause	Storage Occupied
1 through 4	2 bytes (halfword)
5 through 9	4 bytes (fullword)
10 through 18	8 bytes (doubleword)

The operational sign for binary data is contained in the left most bit of the binary data.

PACKED-DECIMAL

Specified for internal decimal items. Such an item appears in storage in packed decimal format. There are 2 digits for each character position, except for the trailing character position, which is occupied by the low-order digit and the sign. Such an item can contain any of the digits 0 through 9, plus a sign, representing a value not exceeding 18 decimal digits.

The sign representation uses the same bit configuration as the 4-bit sign representation in zoned decimal fields.

COMPUTATIONAL or COMP (Binary)

This is the equivalent of BINARY. The COMPUTATIONAL phrase is synonymous with BINARY.

COMPUTATIONAL-1 or COMP-1 (Floating-Point)

Specified for internal floating-point items (single precision). COMP-1 items are 4 bytes long.

COMPUTATIONAL-2 or COMP-2 (Long Floating-Point)

Specified for internal floating-point items (double precision). COMP-2 items are 8 bytes long.

COMPUTATIONAL-3 or COMP-3 (Internal Decimal)

This is the equivalent of PACKED-DECIMAL.

COMPUTATIONAL-4 or COMP-4 (Binary)

This is the equivalent of BINARY.

Conversion

(A) **COMP-3 (Packed Decimal)** **formula - (n / 2) +1**

for e.g. **PIC S9(09) Comp-3**

will take (9 / 2) + 1 = 5 bytes of storage

PIC S9(08) Comp-3

will take (8 / 2) + 1 = 5 bytes of storage

(B) **COMP (Binary)** **formula - n / 2**

for e.g. **PIC S9(09) Comp**

will take 9 / 2 = 4 bytes of storage

PIC S9(08) Comp

will take 8 / 2 = 4 bytes of storage

BACK

IMPORTANT TERMS IN COBOL

EJECT :-

The EJECT statement is a compiler-directing statement which causes the compiler to perform a page eject operation on the source listing. The next source statement line (i.e., the line following the EJECT statement) is then printed at the top of the next page.

Format

EJECT

EJECT may appear in either Area A or Area B but must be the only statement on the line. It may be followed by an optional period.

The EJECT statement may be used in order to force the placement of the beginning of the next section of a source program (i.e., the next record description, the next processing routine or paragraph, etc.) at

the top of a new page within the source listing. This makes the source program more "readable".
The EJECT statement itself is never printed. It has no effect on the object code generated by the compiler.

SKIP:-

The **SKIP** statement is a compiler-directing statement, provided as an IBM extension, which causes blank lines to be inserted in the source program listing.

Format

SKIP1/SKIP2/SKIP3

SKIP1 causes a single blank line to be inserted in the source program listing.

SKIP2 causes two blank lines to be inserted in the source program listing.

SKIP3 causes three blank lines to be inserted in the source program listing.

The **SKIP** statement may appear anywhere in Area A or Area B but must be the only statement on the line. It may be terminated by a separator period.

The SKIP statement itself is never printed. It has no effect on the object code generated by the compiler.

BACK

IMPORTANT TERMS IN VSAM.

VSAM stands for VIRTUAL STORAGE ACCESS METHOD. It is a IBM high performance access method which allows you to access files of different organization such as sequential, indexed, relative record and linear datasets.

CLUSTERS

ESDS:

These are sequential datasets that can be read in the sequence in which they were created.

A VSAM file whose records are loaded without respect to their contents,
and whose relative byte addresses (RBAs) cannot change.
Records are retrieved and stored by addressed access, and new records
are added at the end of the file.

KSDS:

These datasets are stored in sequence of some key field in the record.
Locating the record is a two stage process.
1) first search for the key in the index
2) use the information in the index to locate the record.

BACK

JCL

JOBCAT

The JOBCAT DD statement is used to identify an ICF or VSAM catalog to search first when attempting to locate cataloged data sets during the job's execution.
The JOBCAT must be placed after the JOB statement and before the first EXEC statement in the job.
More than one catalog can be concatenated after the first one on a JOBCAT.

If a STEPCAT DD is specified in a job that also has a JOBCAT, the STEPCAT takes precedence.

If you use a JOBLIB DD statement, in goes in front of the JOBCAT.

Do not use the JOBCAT DD statement in a job that references an SMS-managed data set. SMS only accesses SMS-managed data sets that are cataloged in a system catalog.

Syntax:

```
//JOBCAT  DD  DISP=SHR, DSN=catalog-name
```

Example:

```
//EXAMPLE  JOB  WILLIAMS,MSGLEVEL=1
//JOBLIB   DD   DSNAME=USER.LIB,DISP=SHR
//JOBCAT   DD   DSNAME=LYLE,DISP=SHR
//         EXEC PGM=SCAN
```

In this example, the JOBCAT DD statement specifies a private catalog.
The JOBCAT DD statement follows the JOBLIB DD statement.

STEPCAT

The STEPCAT DD statement is used to identify an ICF or VSAM catalog to search first when attempting to locate cataloged data sets while the step is executing. The STEPCAT can be placed anywhere after the EXEC statement in the job step's JCL. More than one catalog can be concatenated after the first one on a STEPCAT.
If a STEPCAT DD is specified in a job that also has a JOBCAT, the STEPCAT takes precedence.
If you also use a STEPLIB DD statement in the job step, in goes in front of the STEPCAT.
Syntax:

```
//STEPCAT  DD  DISP=SHR, DSN=catalog-name
```

JCLLIB

JCLLIB is used to identify a private library or a system library from which INCLUDE groups and JCL procedures are to be retrieved. The order in which the library names appear on the JCLLIB statement is the order in which they are searched for any JCL procedures (PROCs) and INCLUDE groups referenced by this job. Only one JCLLIB statement is permitted in a job, and it must appear after the JOB statement and before the first EXEC statement in the job. The JCLLIB statement must not appear within an INCLUDE group. You can continue the JCLLIB statement by ending it with a comma followed by at least one blank, then starting the next libary name somewhere between columns 4 and 16 on the next statement.
The private libraries that you specify on the JCLLIB statement must comply with the following rules:

- The private library must be cataloged. However, the library cannot be cataloged in a catalog specified via a JOBCAT or STEPCAT DD statement.

- The private library must be accessible to the job. The library must be permanently resident and online.

- The JCLLIB data set cannot be a password-protected data set.

o The job must have read access to any system or private libraries specified on JCLLIB.

o The private library must have the same data set attributes as a system library, which are:

- Logical record length of 80 bytes (LRECL=80)
- Fixed length records (RECFM=F or RECFM=FB). If the JCLLIB data set is a PDSE, the record format can only be RECFM=FB.

- When multiple libraries are specified on the JCLLIB statement, these libraries will be concatenated.

EXAMPLES:

My home Ids Example 1: The JCLLIB statement below will cause JES to search TEST.PROCLIB1, TEST.PROCLIB2, and then SYS2.PROCLIB for any JCL procedures or INCLUDE groups referenced by this job that contains this statement.

```
//PROCS JCLLIB ORDER=(TEST.PROCLIB1,TEST.PROBLIB2,
//              SYS2.PROCLIB)
```

Example 2: The JCLLIB statement below causes a search of SYS1.PROCLIB, SYS3.USER.PROCLIB, then OPS420.JCLLIB for the ASMHCL procedure and the PRODINC3 INCLUDE group named by the job the JCLLIB statement appears in.

```
//PRODJOB1 JOB (S-1233),CLASS=A,TYPRUN=HOLD,MSGLEVEL=(1,1),
//              MSGCLASS=T
//       JCLLIB ORDER=(SYS1.PROCLIB,
//              SYS3.USER.PROCLIB,
//              OPS420.JCLLIB)
//JS10     EXEC ASMHCL,COND.L=(0,NE)
//JS20     EXEC PGM=IEBCOPY
//INC$ INCLUDE MEMBER=PRODINC3
//
```

FREE=CLOSE

When you specify FREE=CLOSE:

- If the job step abnormally terminates before the data set is closed, the system uses the abnormal termination disposition from the DISP parameter to process the data set. If a recovery routine, such as an ESTAE routine, gets control and closes the data set, however, it

uses the normal termination disposition.

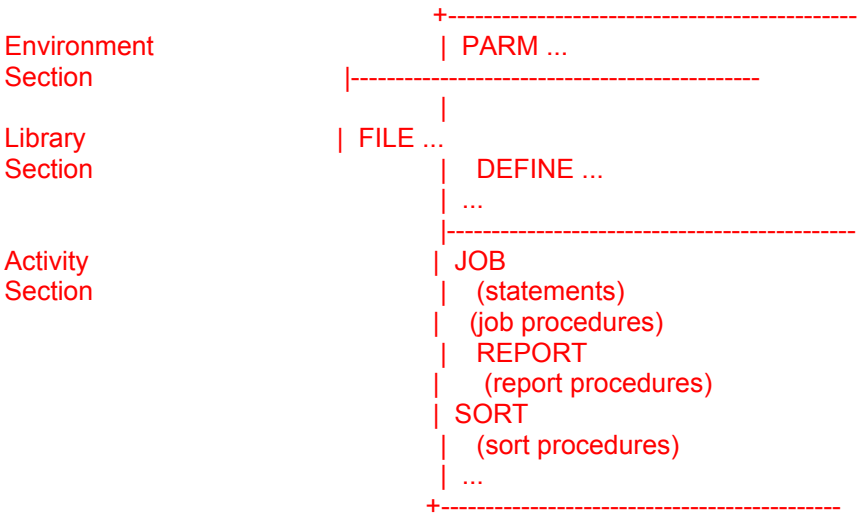
- If the job step abnormally terminates after the data set is closed, then the system has already processed the data set using the normal termination disposition.

Do not specify FREE=CLOSE on a DD statement with a ddname of JOBCAT, JOBLIB, STEPCAT, or STEPLIB; CLOSE is ignored.

BACK

IMPORTANT TERMS IN EAZYTRIEVE.

Different section in Eazytrieve program



TALLY

TALLY contains the number of detail records that comprise a control break. You can use TALLY on a LINE statement or you can use it in calculations within report procedures.
TALLY is commonly used to determine averages for a control level.

TALLY is a ten-byte packed decimal field with zero decimal places. This

definition is used for calculations contained within report procedures.

TALLYSIZE

The TALLYSIZE parameter of the REPORT statement defines the number of digits which are printed for TALLY.
A TALLY accumulator is created for each control break level.

PAGESIZE:
Lines per page (default is 58).

LINESIZE
Length of each line (default is 132 print positions). A linesize of 80 restricts report output to 80 characters per printed
line.

SKIP:
Number of blank lines to be inserted between line groups (default is 0).

SPACE:
Number of blanks inserted (horizontally) between field columns and between fields and literals in title and detail lines (default is 3).

TITLESKIP:
Number of blank lines inserted after last title line before first heading or detail line (default is 3).

SPREAD:
Requests that the columns of data be spread evenly over the entire line, overrides the SPACE parameter (default is NOSPREAD).

NOADJUST:
Requests that the title lines and report be left-justified on the page. The default is centering the report on the page. SPREAD and NOADJUST are mutually exclusive.

NODATE:
Inhibits printing the system date in positions one through eight of the first title line.

NOPAGE:
Inhibits printing a page number.

NOHEADING:
Inhibits printing column headings.

Spacing control parameters are all optional. When used, they can be coded on the REPORT Statement in any order. The general format for these parameters is:

```
                  REPORT  report-name  +

                   /     [PAGESIZE nn]    +
                  |      [LINESIZE nn]     +
                  |      [SKIP nn]          +
                  |      [SPACE nn]         +
                  |      [TITLESKIP nn]     +
                  |
        Spacing    |     +       +
        Control   <     | SPREAD      |
        Parameters |    |NOSPREAD |    +
                  |     +       +
                  |
                  |     [NOADJUST]      +
                  |
                  |     [NODATE]         +
                  |     [NOPAGE]         +
                   \     [NOHEADING]     +
```

SEQUENCE Statement

The **SEQUENCE** statement causes your report to be sorted on a specified key in ascending or descending order.

Ascending order is the default for the SEQUENCE statement. For descending order, you just put a D after the field name separated by a space.

CONTROL

A CONTROL statement specifies that a report should automatically accumulate and print totals. A control break occurs whenever the value of any control field changes or end-of-report is reached. Control fields can be any non-quantitative field from any input file or any W working storage field. At each control break, totals are printed for any quantitative fields specified in the report.

a) You can specify an unlimited number of control fields.

b) Fields are coded on the CONTROL statement in a major to minor order.

Syntax: The format of the CONTROL statement is:

```
                  +                +    +               +
   CONTROL        |     field-name |     | NEWPAGE  |    [NOPRINT]   ...
```

| FINAL | | RENUM |
+ + + +

CONTROL Statement Syntax

1. Final totals are automatically provided. You can alter the default by coding **FINAL NOPRINT.**

2. **NOPRINT**, following any field-name, supresses printing totals for that field (which are still accumulated) at the corresponding control break.

3. **NEWPAGE**, following any field or FINAL, causes a new page after printing the control break totals (or, in the case of FINAL, before printing the final totals). Page numbers continue.

4. **RENUM**, following any field or FINAL, causes a page break and restarts page numbers at 1 after printing the control break totals (or, in the case of FINAL, before printing the final totals).

Control Examples

CONTROL CO RENUM DIV DEPT NOPRINT

CONTROL FINAL NOPRINT CO NEWPAGE DIV

BACK

AN EASY APPROACH FOR PERFORMING CONTROL-BREAKS.

We can use the REDEFINES clause for performing any level Control-Breaks very easily.
This Example demonstrates a 2-level-Control-Break.The first Break is on first 20 characters (WK-COMPANY) and the second break is on the next 60 characters.
Just go through the code given below, in case there is any doubt or confusion feel free to come and ask me.
Also you can refer a program which performs a 3-level Control Break, first on Company, second on Classlab and third on Territory. The program also prints the **Total** (similar to Tally in Eazytrieve) for each Break that is performed.
The program is residing in " CPI213.PATNI.FINAL.SOURCE(CEXNY03B) ".

```
01 WK-CUR-KEY.
    05   WK-COMPANY          PIC X(20)  VALUE LOW-VALUES.
```

```
    05   WK-PLCY-NUMB         PIC X(11)  VALUE LOW-VALUES.
    05   WK-TRANS-TYPE        PIC X(3)   VALUE LOW-VALUES.
    05   WK-OPER-NAME         PIC X(30)  VALUE LOW-VALUES.
    05   WK-EFF-DATE          PIC X(8)   VALUE LOW-VALUES.
    05   WK-INF-DATE          PIC X(8)   VALUE LOW-VALUES.
01 WK-CUR-KEY-R1 REDEFINES WK-CUR-KEY.
    05   WK-CUR-KEY-1         PIC X(20).
    05   FILLER               PIC X(60).
01 WK-CUR-KEY-R2 REDEFINES WK-CUR-KEY.
    05   FILLER               PIC X(20).
    05   WK-CUR-KEY-2         PIC X(60).
```

Declare the old keys in the format described below for holding the read values.

```
01 WK-OLD-KEY                PIC X(80)  VALUE LOW-VALUES.
01 WK-OLD-KEY-R1 REDEFINES WK-OLD-KEY.
    05   WK-OLD-KEY-1         PIC X(20).
    05   FILLER               PIC X(60).
01 WK-OLD-KEY-R2 REDEFINES WK-OLD-KEY.
    05   FILLER               PIC X(20).
    05   WK-OLD-KEY-2         PIC X(60).


B-PROCESS SECTION.
B1-PARA.
    MOVE WK-CUR-KEY TO WK-OLD-KEY.
    MOVE ZERO       TO WS-TOT-PLCY .

    PERFORM BA-PROCESS UNTIL WK-CUR-KEY-1  NOT EQUAL
                             WK-OLD-KEY-1.
B9-PARA.
    EXIT.
BA-PROCESS SECTION.
BA1-PARA.
    MOVE WK-CUR-KEY TO WK-OLD-KEY.
    MOVE ZERO          TO WS-VEH-SEQ.
    PERFORM BAA-PROCESS-REC UNTIL WK-CUR-KEY-2 NOT EQUAL
                            WK-OLD-KEY-2.
     PERFORM YA-PRINT-LINE.
 BA9-PARA.
     EXIT.
 BAA-PROCESS-REC SECTION.
 BAA1-PARA.
     MOVE WK-CUR-KEY          TO WK-OLD-KEY.
     MOVE AZDMV-COMPANY-CODE  TO AZDMV-COMP-NUMB-OLD
     PERFORM YA-PRINT-LINE.
 BAA8-PARA.
     PERFORM XA-READ-AZDMV-FILE .
 BAA9-PARA.
     EXIT.
```

```
XA-READ-AZDMV-FILE SECTION.
XA1-READ.
    READ AZDMV-FILE INTO AZDMV-RECORD
       AT END MOVE HIGH-VALUES TO WK-CUR-KEY
              GO TO XA9-PARA.
    ADD 1 TO WS-REC-READ.
XA9-PARA.
    EXIT.
YA-PRINT-LINE SECTION.
YA1-PARA.
    IF WS-HDR       = 'Y'
       MOVE 'N'  TO WS-HDR
       PERFORM YB-HEADER.
    WRITE  FD-AZRPT-REC.
    ADD 1 TO WS-REC-WRITE.
YA9-PARA.
    EXIT.
```

BACK

Date

When System DATE is Accepted, it comes in
YYMMDD format.

Just consider the example listed below which captures System Date and converts it to the format
MM/DD/YY

```
WORKING-STORAGE SECTION.
01  HDR-WS-DATE.
     05  HDR-WS-MM        PIC X(02) VALUE ZEROS.
     05  FILLER           PIC X(01) VALUE '/'.
     05  HDR-WS-DD        PIC X(02) VALUE ZEROS.
     05  FILLER           PIC X(01) VALUE '/'.
     05  HDR-WS-YY        PIC X(02) VALUE SPACES.
 01  WS-DATE              PIC X(06) VALUE SPACES.
 01  WK-DATE REDEFINES WS-DATE.
     05  WK-YEAR          PIC X(02).
     05  WK-MONTH         PIC X(02).
     05  WK-DAY           PIC X(02).
 PROCEDURE DIVISION.
 0000-MAIN-PARA.
    ACCEPT  WS-DATE  FROM   DATE.
     MOVE     WK-YEAR        TO    HDR-WS-YY.
```

```
     MOVE     WK-MONTH        TO    HDR-WS-MM.
     MOVE     WK-DAY          TO    HDR-WS-DD.
     DISPLAY 'DATE: ' HDR-WS-DATE.
     STOP RUN.
```

INPUT :- 010423
OUTPUT :- DATE: 04/23/01

The above example can be coded using a different logic as follows

```
WORKING-STORAGE SECTION.
01  HDR-WS-DATE.
    05  HDR-WS-MM         PIC 9(02) VALUE ZEROS.
    05  FILLER            PIC X(01) VALUE '/'.
    05  HDR-WS-DD         PIC 9(02) VALUE ZEROS.
    05  FILLER            PIC X(01) VALUE '/'.
    05  HDR-WS-YY         PIC X(02) VALUE SPACES.
01  WS-DATE               PIC X(06) VALUE SPACES.
01  WS-MM-TEMP            PIC X(04) VALUE SPACES.
PROCEDURE DIVISION.
0000-MAIN-PARA.
    ACCEPT  WS-DATE  FROM   DATE.
    MOVE     WS-DATE         TO    HDR-WS-YY.
    MOVE     WS-DATE         TO    WS-MM-TEMP.
    MOVE     WS-MM-TEMP      TO    HDR-WS-MM.
    MOVE     WS-DATE         TO    HDR-WS-DD.
    DISPLAY 'DATE: ' HDR-WS-DATE.
    STOP RUN.
```

INPUT :- 010423
OUTPUT :- DATE: 04/23/01

In this example if you notice that when a variable of PIC X(06) is moved to a variable of PIC X(02), the first two characters are transferred . If you move PIC X(06) to PIC 9(02), then the last two characters get transferred.

System Information Transfer:

System information contained in the specified conceptual data items DATE,
DAY, DAY-OF-WEEK, or TIME, can be transferred into the identifier using ACCEPT

That is,

ACCEPT { IDENTIFIER } FROM DATE.
DAY.
DAY-OF-WEEK
TIME.

DATE

Has the implicit PICTURE 9(6).

The sequence of data elements (from left to right) is:

2 digits for the year
2 digits for the month
2 digits for the day

Thus, 27 April 1995 is expressed as: 950427

DAY

Has the implicit PICTURE 9(5).

The sequence of data elements (from left to right) is:

2 digits for the year
3 digits for the day

Thus, 27 April 1995 is expressed as: 95117

DAY-OF-WEEK

Has the implicit PICTURE 9(1).

The single data element represents the day of the week according to the following values:

1 represents Monday
2 represents Tuesday
3 represents Wednesday
4 represents Thursday
5 represents Friday
6 represents Saturday
7 represents Sunday

Thus, Wednesday is expressed as: 3

TIME

Has the implicit PICTURE 9(8).

The sequence of data elements (from left to right) is:

2 digits for hour of day
2 digits for minute of hour
2 digits for second of minute
2 digits for hundredths of second

Thus, 2:41 PM is expressed as: 14410000

BACK

Windowing

Windowing techinque is generally used for making the date Y2K Complaint.
For Windowing technique 50 is taken as the base year.
The structure of the date is as follows:-

```
01  WS-DATE.
    05  WS-CC                    PIC X(02).
    05  WS-YY                    PIC X(02).
    05  WS-MM                    PIC X(02).
    05  WS-DD                    PIC X(02).
```

It is done in the following manner:

```
IF  WS-TERM-YY   >   '50'
    MOVE  '19'  TO  WS-CC
ELSE
    MOVE  '20'  TO  WS-CC
END-IF.
```

Where **WS-TERM-YY** is the year for which windowing has to be done.

For Date-Of-Birth cases '10' is taken as the base year.

BACK

JOB-CLASS-PARAMETER

Job classes represent queues of work which exhibit similar processing characteristics. Once the JOB queues are specified, MVS initiators can then be assigned to take work out of these well defined queues based on the processing objectives of the center. Each initiator represents a unit of processing, and the total number of active initiators represents the maximum level of multi-programming to be achieved.

ADHOC (Risk, Marketing, Collections, etc.)	
CLASS	**DEFINITION**
R	This class is for batch jobs requiring 4 or more tape drives. No time parameter is required. DO NOT SUBMIT JOBS IN THIS CLASS UNLESS YOU REQUIRE 4 OR more TAPE DRIVES.
S	This class is for batch jobs requiring tape drives. No time parameter is required. This class is for jobs that require 3 or LESS tape drives. DO NOT SUBMIT JOBS IN THIS CLASS UNLESS YOU REQUIRE 3 OR LESS TAPE DRIVES.

T	Only jobs requiring less than 30 seconds of CPU time and NO tape drives are permitted in this class. TIME parameter is required on the job card. TIME=(,30).
U	This class is for jobs using up to 3 tape drives, and 60 CPU seconds. Time parameter is required. TIME=1.
W	This is the overnight job class for delayed processing.
DEVELOPMENT (RFSIS Development Staff, Systems Software)	
A	This is the quick turnaround class. Only jobs requiring 5 Seconds or less of CPU time. No tape drives are permitted. TIME=(,5) parameter on the JOB card is required.
Q	Only jobs requiring less than 30 seconds of CPU time and no tape drives are permitted in this class. TIME parameter is required on the job card. TIME=(,30).
1	This class is for jobs using up to 3 tape drives, and 60 CPU seconds. Time parameter is required. TIME=1.
X	This class is for jobs using more than 1 minute of CPU time and/or more than 3 tape drives. Jobs in this class will be run at the operator's discretion depending upon system load.
W	This is the overnight job class for development processing
Z	This class is for jobs using more than 1 minute of CPU time and no tape drives. Any jobs executing in this class allocating tape drives will be cancelled by operations.
PRODUCTION (Production control, Computer Operations, System Software only)	
B	Reserved for production jobs. (Jobs submitted by CA7.)
D	Reserved for production jobs. (Jobs submitted by CA7.)
C	Reserved for special CICS/IDMS batch jobs.
7	Reserved for CA7.
8	Reserved for CICS and IDMS journals.
9	Reserved for APC.
P	Reserved

BACK

_____

Abbreviations

VSAM	Virtual Storage Access Method.
ESDS	Entry Sequenced Dataset.
KSDS	Key Sequenced Dataset.
RRDS	Relative Record Dataset.
LDS	Linear Dataset.
ICF	Integrated Catalog Facility.
RACF	Resource Access Control Facility.
DASD	Direct Access Storage Device.
CIDF	Control Interval Descriptor Field
RDF	Record Descriptor Field
RBA	Relative Byte Address
HURBA	High-Used-RBA
HARBA	High-Alloc-RBA.
CISZ	Control Interval Size.
GRS	Global Resource Serialization.
GDG	Generation Data Group.
VTOC	Volume Table of Contents

JCL	JOB Control Language
CISC	Customer Information Control System.
MVS	Multiple Virtual Storage
JES	JOB Entry Subsystem
SPOOL	Simultaneous Peripheral Operation On-line.
SYSTEMS	
EXPR	Experienced Reporting System
MSS	Marketing and Sales System
CAMS	Cash Application Management System
NAII	National Association of Independent Insurers
STATS	Statistical Reporting System
DMV	Department of Motor Vehicle
ODS	Operational Data Source.
SRS	Sales Reporting System

BACK