Job Control Language

JCL is an interface between an Application Program and MVS OS. Using JCL we could execute programs in TSO written in COBOL, ASSEMBLER etc. JCL executes a program in batch mode.

JCL Coding Rules

1) JCL has a Job card. Job card is made of up of two parameters namely positional and keyword parameters. A Job has one or more Job steps to execute Programs and utilities to perform a certain functionality.

JOB CARD Parameters:-

Positional Parameters: Accounting Info is a positional parameter and has to be given and Programmer name, if omitted should be replaced by a comma

Keyword Parameters: This can appear in any order after keyword

parameters

CLASS: Depends on time duration and Resources (0-9, A-Z)

PRTY: Specifying Priority of a Job (0-15) within a Job class and is an Optional one.

NOTIFY: Sends the status of Job execution to the userid specified. &SYSUID denotes the user submitting the job

MSGCLASS: Denotes the destination for system messages

(A-Z,0-9)

MSGLEVEL: Determines the type of messages written to O/p destination specified in Job class

MSGLEVEL=(1,1) - Preferred

User Msgs: 1-JCL Satements

System Msgs: 1-Normal OR Abnormal Job completion

TYPRUN: SCAN/HOLD

SCAN: Checks for Syntax errors

HOLD: Puts the Job on hold, To release type 'A' against the job in spool

TIME: (mm,ss) or Time=ss, nolimit, min=1-1439 - 1day, seconds=1-59, max = 248 days

REGION=4096K or 0M (Larger Database access requires more virtual memory)

NOTIFY: &SYSUID (Notify the Successful or Abnormal Execution of Job)

2) The first 2 chars of a jcl statement should be //, 72nd column is left blank unless otherwise a jcl statement is following next line
3) Operand keywords should start from col 16
4) //* is a comment
5) We can have up to 255 job steps in a single jcl

Line Commands:

I – Insert a single line
I5 – Insert 5 lines
RR
RR – Repeat a block of lines
R - Repeat a line
C - Copy a single line and paste After or before a line by typing A or B on a line
D – Deletes a line
DD….DD - Blocks and deletes a set of lines
CC…CC - Blocks the lines and type CUT on command prompt to cut lines and type PASTE where you want to paste

Command Prompts
RES – Resets
REF – Refresh the info
COLS – Displays the column numbers
SAVE – SAVEs a Dataset
SUB – Submit a Job

1. **IEFBR14 – Dummy Utility to Create/Delete a Physical Sequential dataset or a PDS**

```
//STEP01       EXEC  PGM=IEFBR14
//*deletes an existing physical seq dataset
//DD1          DD    DSN=WPR085.INPUT, DISP=(OLD,DELETE,DELETE)
//*

//STEP01       EXEC  PGM=IEFBR14
//* creates a dataset
//DD1          DD    DSN=WPR085.OUTPUT,
//                   DISP=(NEW,CATLG,DELETE),
//                   SPACE=(TRK,(1,1),RLSE),
//                   DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//*

//STEP01       EXEC  PGM=IEFBR14
//* creates a pds dataset
//DD1          DD    DSN=WPR085.OUTPUT,
//                   DISP=(NEW,CATLG,DELETE),
//                   SPACE=(TRK,(1,1,5),RLSE),
//                  DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DSORG=PO)
//*
```

2. **IEBGENER**

Copy Seq files to Seq files

```
//STEP01       EXEC  PGM=IEBGENER
//SYSUT1       DD    DSN=WPR085.INPUT1, DISP=SHR
//SYSUT2       DD    DSN=WPR085.OUTPUT1,
//                   DISP=(NEW,CATLG,DELETE),
//                   SPACE=(TRK,(1,1),RLSE),
//                   DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSPRINT     DD    SYSOUT=*
//*
```

Copy A Member of PDS to Seq files

```
//STEP01       EXEC  PGM=IEBGENER
//SYSUT1       DD    DSN=WPR085.INPUT.PDS(TESTJCL1), DISP=SHR
//SYSUT2       DD    DSN=WPR085.OUTPUT2,
//                   DISP=(NEW,CATLG,DELETE),
//                   SPACE=(TRK,(1,1),RLSE),
//                   DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSPRINT     DD    SYSOUT=*
//*
```

Copy 3 PS FILES to a Seq file

```
//STEP01       EXEC  PGM=IEBGENER
```

```
//SYSUT1       DD    DSN=WPR085.INPUT1,DISP=SHR
//            DD    DSN=WPR085.INPUT2,DISP=SHR
//            DD    DSN=WPR085.INPUT3,DISP=SHR
//SYSUT2       DD    DSN=WPR085.OUTPUT3,
//                   DISP=(NEW,CATLG,DELETE),
//                   SPACE=(TRK,(1,1),RLSE),
//                   DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSPRINT     DD    SYSOUT=*
```

3. **IEBCOPY**

(CREATES A PDS FROM AN EXISTING PDS AND COPIES ALL ITS MEMBERS)

```
//STEP01       EXEC  PGM=IEBCOPY
//DDIN         DD    DSN=WPR085.JCL.LIB,DISP=SHR
//DDOUT        DD    DSN=WPR085.JCL.LIB.BKUP,
//                   DISP=(NEW,CATLG,DELETE),
//                   SPACE=(TRK,(1,1,1),RLSE),
//                   DCB=(RECFM=FB,DSORG=PO)
//SYSPRINT     DD   SYSOUT=*
//SYSIN        DD    *
 COPY OUTDD=DDOUT,INDD=DDIN
/*
//*
```

(CREATES A PDS FROM AN EXISTING PDS AND COPIES MEMBERS MEMBER1,MEMBER2 AND MEMBER3 TO THE NEW PDS)

```
//STEP01       EXEC  PGM=IEBCOPY
//DDIN         DD    DSN=WPR085.JCL.LIB1,DISP=SHR
//DDOUT        DD    DSN=WPR085.JCL.LIB1.BKUP,
//                   DISP=(NEW,CATLG,DELETE),
//                   SPACE=(TRK,(1,1,1),RLSE),
//                   DCB=(RECFM=FB,DSORG=PO)
//SYSPRINT     DD   SYSOUT=*
//SYSIN        DD    *
 COPY OUTDD=DDOUT,INDD=DDIN
 SELECT MEMBER=(MEMBER1,MEMBER2,MEMBER3)
/*
//*
```

(CREATES A PDS FROM AN EXISTING PDS AND COPIES ALL MEMBERS EXCEPT MEMBER2 TO THE NEW PDS)

```
//STEP01       EXEC  PGM=IEBCOPY
//DDIN         DD    DSN=WPR085.JCL.LIB1,DISP=SHR
//DDOUT        DD    DSN=WPR085.JCL.LIB1.BKUP,
//                   DISP=(NEW,CATLG,DELETE),
//                   SPACE=(TRK,(1,1,1),RLSE),
//                   DCB=(RECFM=FB,DSORG=PO)
```

```
//SYSPRINT     DD   SYSOUT=*
//SYSIN        DD    *
 COPY OUTDD=DDOUT,INDD=DDIN
 EXCLUDE MEMBER=(MEMBER2)
/*
//*
```

4. IEBCOMPR

Compares two Input files

```
//STEP01       EXEC  PGM=IEBCOMPR
//SYSUT1       DD    DSN=WPR085.INPUT1, DISP=SHR
//SYSUT2       DD    DSN=WPR085.INPUT2, DISP=SHR
//SYSOUT       DD    SYSOUT=*
//SYSPRINT     DD    SYSOUT=*
//SYSIN        DD    DUMMY
```

5. SORT & MERGE

a) Copies Input file to Output file

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=WPR085.SORTOUT,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=COPY
/*
//SYSOUT      DD  SYSOUT=*
//*
```

B) Copies Input file to Output file and sort first 10 chars in ascending order

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=WPR085.SORTOUT1,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(1,10,CH,A)
/*
//SYSOUT      DD  SYSOUT=*
```

C) Copies Input file to Output file and sort A NUMERIC field from 21st char to 25th char in descending order

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=WPR085.SORTOUT1,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(21,5,ZD,A)
/*
//SYSOUT      DD  SYSOUT=*
```

D) Copies Input file to Output file and Sort first ten chars in ascending order and within this sorted data, sort the file in ascending order of a numeric field (21-25 columns)

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=WPR085.SORTOUT1,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(1,10,CH,A,21,5,ZD,A)
/*
//SYSOUT      DD  SYSOUT=*
```

E) Copies Input file to Output file and Sort first ten chars in descending order and copy selective records

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=WPR085.SORTOUT1,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(1,20,CH,A)
  INCLUDE COND=(26,1,CH,EQ,C'M')
/*
//SYSOUT      DD  SYSOUT=*
```

F) Copies Input file to Output file and Sort first ten chars in descending order and copy selective records

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=WPR085.SORTOUT1,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(1,20,CH,A)
  OMIT COND=(26,1,CH,EQ,C'M')
/*
//SYSOUT      DD  SYSOUT=*
```

G) Copies Input file to Output file and Sort first ten chars in descending order and copy age greater than 40

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=WPR085.SORTOUT1,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(1,20,CH,A)
  INCLUDE COND=(21,5,ZD,GT,40)
/*
//SYSOUT      DD  SYSOUT=*
```

H) Merges two files (Both the input files should already be in the sorted order exactly same as the parameter given in the merge fields)

```
//STEP01    EXEC  PGM=SORT
//SORTIN01    DD  DSN=WPR085.MONTHLY.FILE,DISP=SHR
//SORTIN02    DD  DSN=WPR085.DAILY.FILE,DISP=SHR
//SORTOUT     DD  DSN=WPR085.SORTOUT,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  MERGE FIELDS=(1,20,CH,A)
/*
//SYSOUT      DD  SYSOUT=*
//*
```

I) Merging two or more files using SORT FIELDS (Here I/p files need not be presorted)

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.DAILY.FILE1,DISP=SHR
//            DD  DSN=WPR085.DAILY.FILE2,DISP=SHR
//            DD  DSN=WPR085.DAILY.FILE1,DISP=SHR
//SORTOUT     DD  DSN=WPR085.SORTOUT,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  MERGE FIELDS=(1,20,CH,A)
/*
//SYSOUT      DD  SYSOUT=*
//*
```

6. TEMPERORY DATASETS

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=&&TEMP1,
//                DISP=(NEW,PASS,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(1,10,CH,A)
/*
//SYSOUT      DD  SYSOUT=*
//*
//STEP02    EXEC  PGM=COBOLPGM1
//INFILE      DD  DSN=&&TEMP1,DISP=SHR
//OUTFILE     DD  DSN=WPR085.COBOL.OUTPUT,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSOUT      DD  SYSOUT=*
```

```
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=&&TEMP1,
//                DISP=(NEW,PASS,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
```

```
  SORT FIELDS=(1,10,CH,A)
/*
//SYSOUT      DD  SYSOUT=*
//*
//STEP02    EXEC  PGM=IEBGENER
//SYSUT1      DD  DSN=&&TEMP1,DISP=SHR
//SYSUT2      DD  DSN=WPR085.COBOL.OUTPUT1,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSPRINT    DD  SYSOUT=*
//SYSUDUMP    DD  SYSOUT=*
//SYSABEND    DD  SYSOUT=*
```

7. CONDITIONAL PROCESSING
COND PARM

COND parameter is used to test the condition code (return code) of one or more steps or all the previous steps and decide whether to execute the current step or not. This parm is usually given in Job steps.

COND=EVEN (EXECUTES EVEN IF ABOVE STEPS RUNS SUCCESSFULLY OR NOT)
COND=ONLY (EXECUTES only if any of the above job step fails)

COND=(4,LT)

COND=(4,NE,<STEPNAME>)
If the condition is true, it doesn’t execute. If it is false, then it executes
COND=((4,NE,<STEPNAME>)AND(4,NE,<STEPNAME>))

If the condition is satisfied, it doesn’t execute.
If the condition is not satisfied, it executes.

IF THEN ELSE ENDIF PARM

```
//    IF STEPNAME.RC = 0 THEN
//STEP01 EXEC PGM=TESTPGM1
//    ELSE
//STEP02 EXEC PGM=TESTPGM2
//    ENDIF
```

8. **WRITING INSTREAM DATA INTO A FILE THRU JOB**

```
//STEP01    EXEC  PGM=IEBGENER
//SYSUT1      DD  *
007M S DHONI       35
022SHIKAR DHAWAN   32
050AJINKYA RAHANE  29
010VIRAT KOHLI     26
011ROHIT SHARMA    28
099SURESH RAINA    29
100MOHD SHAMI      33
005UMESH YADAV     34
004ASHWIN          31
088RAVINDRA JADEJA 28
011BHUVA KUMAR     33
/*
//SYSUT2      DD  DSN=WPR085.COBOL.INPUT12,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSPRINT    DD  SYSOUT=*
//SYSUDUMP    DD  SYSOUT=*
//SYSABEND    DD  SYSOUT=*
```

PROCEDURE:
A PROC is nothing but a set of JOB steps without a Job card. It is executable only thru a JCL.

Need for a PROC. A JCL cannot have more than 255 steps. So we have a concept called PROC where the Job steps reside and it is executed via a JCL. Also, in production environment, during production issues, if we need to change any component in proc, it is highly risky and this could be achieved thru overriding parameters thru a JCL.

In JCL

```
//JOB CARD….
// JCLLIB ORDER DSN=WPR085.PROC.LIBRARY
//*
//STEP01  EXEC  SORTPROC
//*
```

SORTPROC is a PROC and present in WPR085.PROC.LIBRARY

A PROC will have all Job steps. The first statement will be having a PROC name and a operation keyword PROC as given below:-

```
//SORTPROC    PROC
//*
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=WPR085.SORTIN,DISP=SHR
//SORTOUT     DD  DSN=WPRO085.SORTOUT,
//                DISP=(NEW,CATLG,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(1,10,CH,A)
/*
//SYSOUT      DD  SYSOUT=*
//*
```

Usage OF Symbolic parm

```
//SORTPROC  PROC  HLQ=WPR085
//*
//STEP01    EXEC  PGM=SORT
//SORTIN      DD  DSN=&HLQ..SORTIN,DISP=SHR
//SORTOUT     DD  DSN=&HLQ..SORTOUT,
//                DISP=(NEW,PASS,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(1,10,CH,A)
/*
//SYSOUT      DD  SYSOUT=*
//*
```

OVERRIDING PARAMETERS IN A PROC THRU JCL

A) OVERRIDING A FILE THRU JCL

In JCL

```
//JOB CARD….
// JCLLIB ORDER DSN=WPR085.PROC.LIBRARY
//*
//STEP01  EXEC  SORTPROC
//STEPSORT.SORTIN DD DSN=WPR085.SORTIN.NEW,DISP=SHR
//*
```

```
//SORTPROC  PROC  HLQ=WPR085
//*
//STEPSORT    EXEC  PGM=SORT
//SORTIN      DD  DSN=&HLQ..SORTIN,DISP=SHR
```

```
//SORTOUT     DD  DSN=&HLQ..SORTOUT,
//                DISP=(NEW,PASS,DELETE),
//                SPACE=(TRK,(1,1),RLSE),
//                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
//SYSIN       DD  *
  SORT FIELDS=(1,10,CH,A)
/*
//SYSOUT      DD  SYSOUT=*
//*
```