JOB Statement
Information about the job.

The JOB statement will vary at each installation of computer hardware and software. The next column allows you record notes about the following operands:

```
//jobname JOB (accounting),'name20',
//   CLASS=input-class,
//   MSGCLASS=output-class,
//   MSGLEVEL=(jcl,messages),
//   TIME=(minutes,seconds) | 1440 | NOLIMIT |
//              MAXIMUM,
//   REGION=storageK | M,
//   NOTIFY=tso-userid,
//   RESTART=(*[.stepname][.procstepname][,check-point-id]),
//   COND=((return-code,operator)[,...][,EVEN | ONLY]),
//   TYPRUN=SCAN | HOLD | JCLHOLD | COPY,
//   GROUP=racf-group,
//   USER=tso-userid,
//   PASSWORD=(password[,newpassword]),
//   SECLABEL=racf-label,
//   BYTES=(n[,CANCEL | DUMP | WARNING]),
//   CARDS=(n[,CANCEL | DUMP | WARNING]),
//   LINES=(n[,CANCEL | DUMP | WARNING]),
//   PAGES=(n[,CANCEL | DUMP | WARNING]),
//   RD=R | RNC | NR | NC,
//   PRTY=priority,
//   PERFORM=performance-group,
//   ADDRSPC=VIRT | REAL
```

JCL Advice
Use only uppercase in JCL.
Change your jobname character before you submit.
Delete old datasets before DISP=(NEW,CATLG),
or use a Generation dataset (see IDCAMS)
Never use DISP=(MOD,…) with a PDS (except DELETE).
datasetname(member) may be treated like sequential.
Do not use DSN=&&name or DISP=(…,PASS),
use Flavor #4 and delete when done.
Avoid referbacks except to the same step.
Use DISP=(OLD,DELETE) after ABEND or bad RC.

Find out about your shop using the next column!

Installation Specific Information
Your shop.

jobname
Usually: **useridX** any character, 8 max.
Other acceptables jobnames:

Accounting Write in your format:
(**)**

Input-classes	**Output-classes**

Dataset names	**Generic Units**
userid.*	SYSDA
project.group.type	TAPE
SYSn.*	

EXEC Statement
Execute a Program.

```
//JOBLIB DD DSN=loadlib,DISP=SHR
//stepname EXEC PGM=programname,
//    PARM='data100',
//    REGION=storageK | M,
//    TIME=(minutes,seconds),
//    ACCT=accounting,
//    COND=((return-code,operator[,stepname]
//                              [,EVEN | ONLY] ),
//    ADDRSPC=VIRT | REAL,
//    DPRTY=(priority[,dispatch-addto-priority] ),
//    DYNAMNBR=number-of-dynamic-allocations,
//    PERFORM=performance-group,
//    RD=R | RNC | NR | NC
//STEPLIB DD DSN=loadlib,DISP=SHR
```

Execute/Override a Procedure.

```
//name JCLLIB ORDER=(proclib[,…] )
//*   Variables override procedure and SET statement.
//stepname EXEC PROC=procname,
//    variable=value,
//    operand.procstepname=see-above-operands
//procstepname.ddname DD operands
```

PROC/PEND Statement

```
//*   Variables override job SET statement.
//procname PROC [variable=[value][,…] ]
//*   Body of procedure, use &variable.
//*   SET statement in body changes variable.
//    PEND
```

SET Statement

```
//*    Variables may appear in procedures
//*    and jobs, use &variable.
//name SET variable=value[,…]
```

INCLUDE Statement

```
//*    Copy JCL into a job or procedure.
//*    Requires JCLLIB Statement.
//name INCLUDE MEMBER=member-name
```

Data Definition Statement
Where's the data?

```
//ddname DD operands
//          DD input-concatenation
```

//* Flavor #1 - Instream data (Fixed, 80 characters)

```
//ddname DD *      OR DD DATA,DLM=$$
this is now data
/*                 OR $$ IF DATA HAS JCL IN IT
```

//* Flavor #2 - Printers (Output Queue)

```
//ddname DD SYSOUT=(class, writer,form),
//    DEST=printername,COPIES=number,
//    HOLD=YES | NO,OUTLIM=lines,
//    SEGMENT=pages,
//    FREE=CLOSE | END,SPIN=UNALLOC | NO,
//    OUTPUT=(*.stepname.procstepname.name[,...] )
```

//* Flavor #3 - Existing data (DASD or Tape)

```
//ddname DD DSN=datasetname[(membername)],
//    DISP=SHR       OLD  FOR EXCLUSIVE USE
//*                  MOD TO EXTEND DSORG=PS
```

//* Flavor #4 - New Permanent (DASD & Tape)

```
//ddname DD DSN=datasetname[(membername)],
// DISP=(NEW,CATLG,DELETE),
// UNIT=(device-type,count,[DEFER] ),
// VOL=(,[RETAIN],,,SER=(volumename,...)
//                  | REF=datasetname),
// SPACE=(TRK | CYL | blocksize | recordwidth,
//          (primary,secondary,directory),RLSE),
// AVGREC=U | K | M,DCB=(datasetname,overrides),
// DSORG=PO | PS,RECFM=FB | FBA | VB | VBA | U,
// LRECL=recordwidth,BLKSIZE=blocksize | 0,
// EXPDT=yyddd | yyyy/ddd  |  RETPD=days,
// LABEL=(sequence#,SL | SUL | NSL | NL | BLP | LTM | AL | AUL)
```

//* Flavor #5 - Scratch pad (1 step only)

```
//ddname DD UNIT=devicetype,
//    SPACE=(CYL,(primary,secondary))
```

//* Flavor #6 - No data (Empty)

```
//ddname DD DUMMY,BLKSIZE=blocksize
```

IDCAMS
General Purpose Utility

```
//stepname EXEC PGM=IDCAMS
//SYSPRINT  DD diagnostic-report
//anydd     DD operands
//SYSIN     DD *
  DELETE (datasetname[ (member) ][,...])  +
         ERASE | NOERASE  +
         PURGE | NOPURGE  +
         FORCE | NOFORCE  +
         SCRATCH | NOSCRATCH

  LISTC ENTRIES(datasetname[,...]) | LEVEL(dsn-prefix)  +
       NAME | HISTORY | ALLOCATION | VOLUME | ALL  +
       [OUTFILE(anydd) ]  +
       [CREATION(days) ]  +
       [EXPIRATION(days) ]

  DEFINE GENERATIONDATAGROUP (
          NAME(gdg-base-name)  +
          LIMIT(number-of-generations)  +
          EMPTY | NOEMPTY  +
          SCRATCH | NOSCRATCH  +
          [OWNER(text) ]  )
)

  DEFINE NONVSAM (NAME(datasetname)  +
          DEVICETYPES(device[,...] )  +
          VOLUMES(volser[,...] )  +
          [FILESEQUENCENUMBERS(n[,...] ) ]  +
          [OWNER(text) ]  +
          [TO(date) | FOR(days) ]  +
          RECATALOG | NORECATALOG  )

  PRINT INFILE(anydd) | INDATASET(datasetname)  +
       CHARACTER | DUMP | HEX  +
       [FROMKEY(key) | FROMADDRESS(rba) |
            FROMNUMBER(n) | SKIP(n) ]  +
       [TOKEY(key) | TOADDRESS(rba) |
          TONUMBER(n) | COUNT(n) ]  +
       [OUTFILE(anydd)]

  /* FROM and TO options like PRINT also available. */
  REPRO INFILE(anydd) | INDATASET(datasetname)  +
         OUTFILE(anydd) | OUTDATASET(datasetname)  +
         REPLACE | NOREPLACE  +
         REUSE | NOREUSE

  SET MAXCC | LASTCC = number
  IF LASTCC | MAXCC  operator  number  THEN command
  [ELSE command]
  DO ... END  /* Allows multiple commands in IF  */
  CANCEL      /* Halts processing                */
```

IEBGENER
Sequential Copy/Reformat Utility

```
//stepname EXEC PGM=IEBGENER
//SYSPRINT DD diagnostic-report
//SYSUT1     DD input-data
//SYSUT2     DD output-data
//SYSIN      DD *            OR DUMMY
  (none) - Exact duplicate copy.

  GENERATE MAXNAME=members,
           MAXFLDS=fields,
           MAXLITS=characters,
           MAXGPS=idents
  [MEMBER NAME=(member,[alias,...] )]
  [RECORD [IDENT=(length,'data',incolumn),]
    [FIELD=(length,incolumn,[PZ | ZP],outcolumn),...]
    [FIELD=(length,'data',,outcolumn),...] ]
```

SORT
Copy/Sort/Extract/Merge Utility

```
//stepname EXEC PGM=SORT
//SORTLIB    DD program-library (optional?)
//SYSOUT     DD diagnostic-report
//*   xx from 01 to 31 allowed.
//SORTWKxx DD Flavor #5
//*   nn blank for sort, 01 to 16 for merge.
//SORTINnn   DD input-data
//SORTOUT    DD output-data
//SYSIN      DD *
* Comment
 OPTION            [COPY] [,EQUALS] [,SKIPREC=n]
                          [,STOPAFT=n]
 SORT | MERGE   FIELDS=(field,sequence[,...] )[,EQUALS]
                or  FIELDS=COPY
 INCLUDE | OMIT COND=(condition[,AND | OR,...] )
 INREC | OUTREC FIELDS=(constant | field[,...] )
 SUM              FIELDS=(field[,...] )
                or  FIELDS=NONE
```

field (S,L,F)
start-column,field-length,CH | ZD | PD | BI
condition (F1,R,F2)
field,EQ | NE | GT | GE | LT | LE,field | constant
constant (n for occurrences of)
nX (blank), nC'...' (char), nX'...' (hex), nZ (x'00')
sequence (Ascending or descending)
A | D

IEFBR14
The Do Nothing Utility!

```
//*   Ensure a dataset is deleted.
//stepname EXEC PGM=IEFBR14
//anydd  DD DSN=datasetname,
//    DISP=(MOD,DELETE),          DASD
//    UNIT=device-type,SPACE=(TRK,0)
//    DISP=(MOD,UNCATLG),         TAPE
//    UNIT=(TAPE,,DEFER)

//*   Create a dataset for later OLD/MOD.
//*   Use Flavor #4 of the DD Statement.
//*   The first program must open output
//*   and may set the DCB instead.
```

IEBCOPY
PDS Copy/Merge/Compress Utility

```
//stepname EXEC PGM=IEBCOPY
//SYSPRINT DD diagnostic-report
//SYSUT2     DD Flavor #5
//SYSUT3     DD Flavor #5
//anydd      DD operands
//SYSIN      DD *
  COPY OUTDD=anydd,INDD=((anydd[,R]),...)
  [SELECT MEMBER=( (name[,newname[,R] ] ),...) ]
  [EXCLUDE MEMBER=(name,...)]
```

SELECT and EXCLUDE are mutually exclusive.

IEHLIST
DASD VTOC's and PDS Directories

```
//stepname EXEC PGM=IEHLIST
//SYSPRINT  DD diagnostic-report
//anydd     DD UNIT=device-type,DISP=SHR,
//               VOL=SER=volser
//SYSIN     DD *
  LISTVTOC [VOL=device-type=volser],
           [DUMP | FORMAT],
           [DATE=dddyy | dddyyyy],
           [DSN=(datasetname,...)]

  LISTPDS DSNAME=(datasetname,...),
          [VOL=device-type=volser],
          [DUMP | FORMAT]
```

JCL Reference Summary

```
//* The Future of JCL
//* Flavor #7  -  System Managed Storage (SMS)
//ddname DD DSN=datasetname,
//    DISP=(NEW,CATLG),
//    DATACLAS=name,
//    STORCLAS=name,
//    MGMTCLAS=name,
//    overriding-operands

//* Flavor #8  -  Modeling Datasets
//ddname DD DSN=datasetname,
//    DISP=(NEW,CATLG),
//    LIKE=datasetname,
//    overriding-operands

//* Flavor #9  -  VSAM Datasets
//ddname DD DSN=datasetname,
//    DISP=(NEW | SHR | OLD,
//          CATLG | KEEP | DELETE),
//    RECORG=KS | ES | RR | LS,
//    KEYOFF=offset,
//    KEYLEN=bytes,
//    DATACLAS=name
//*   Dataclas for more VSAM parameters
```

IF/THEN/ELSE
Alternative to COND operand

```
//name IF expression THEN
//*   JCL steps to execute if true
//name ELSE
//*   JCL steps to execute if false (optional)
//name ENDIF
```

expression (may be stepname.procstepname.)
RC > < ¬> ¬< = ¬= >= <= number AND ... OR ... NOT ...
ABEND=TRUE | FALSE
ABENDCC=Sxxx | Uxxxx
RUN=TRUE | FALSE