Mainframe Fresher Interview Q&A

JCL, VSAM, COBOL, DB2 and CICS

150 questions with ready-to-speak answers for freshers who have completed mainframe training. Each section moves from basic to intermediate to scenario-based questions.

Section 1: JCL, 30 questions	2
Section 2: VSAM, 20 questions	9
Section 3: COBOL, 50 questions	13
Section 4: DB2, 30 questions	23
Section 5: CICS, 20 questions	29
Quick interview tips	34

*** FREQUENTLY ASKED** marks the questions asked most often in fresher interviews.

Section 1: JCL

Job card, DD and DISP, IBM utilities, PS and PDS, SORT utility, procedures, abends

BASIC | * FREQUENTLY ASKED

Q1. What is JCL and why do we use it?

Job Control Language tells z/OS what work to run: which programs to execute, which datasets they read and write, and what resources they need. A job is made of a **JOB** statement, one or more **EXEC** statements (steps) and **DD** statements (data definitions). Every JCL statement starts with // in columns 1-2 and is processed by the JES subsystem.

```
//MYJOB1  JOB (ACCT),'VENKAT',CLASS=A,MSGCLASS=X
//STEP01  EXEC PGM=EMPPGM
//INFILE  DD DSN=USER1.EMP.DATA,DISP=SHR
//OUTFILE DD SYSOUT=*
```

BASIC | * FREQUENTLY ASKED

Q2. Explain the JOB statement and its common parameters.

The JOB statement marks the start of a job: //jobname JOB (accounting),'programmer name',parameters. Common parameters: **CLASS** (job queue/initiator class), **MSGCLASS** (where the job log goes), **MSGLEVEL=(1,1)** (print all JCL and allocation messages), **NOTIFY=&SYSUID** (message to your TSO id at completion), **REGION** (memory limit), **TIME** (CPU limit), **TYPRUN=SCAN** (check syntax only) and **RESTART**. Defaults and allowed values are set by each installation.

```
//EMPJOB1 JOB (ACCT123),'VENKAT',CLASS=A,MSGCLASS=X,
//         MSGLEVEL=(1,1),NOTIFY=&SYSUID,
//         REGION=0M,TIME=(1,30)
```

BASIC

Q3. What does the EXEC statement do? Explain PGM, PROC and PARM.

EXEC starts a job step. **PGM=name** runs a load module (program or utility). **PROC=name** (or just the proc name) runs a procedure. **PARM** passes up to 100 characters to the program. Other useful parameters are COND, REGION and TIME at step level. Steps run one after another in the order coded.

```
//STEP01 EXEC PGM=EMPPGM,PARM='20261026',REGION=64M
//STEP02 EXEC PROC=BKPPROC
```

BASIC | * FREQUENTLY ASKED

Q4. Explain the DD statement and its main parameters.

A DD statement connects a program's file name (ddname) to a real dataset. **DSN** is the dataset name, **DISP** says how the dataset is used and what happens to it, **SPACE** and **UNIT** allocate a new dataset, **DCB** gives the record format (RECFM, LRECL, BLKSIZE), and **SYSOUT** sends output to the spool. The ddname must match the name in the program (COBOL ASSIGN TO ddname).

```
//INFILE  DD DSN=USER1.EMP.DATA,DISP=SHR
//OUTFILE DD DSN=USER1.EMP.OUT,DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(5,2),RLSE),UNIT=SYSDA,
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//RPT     DD SYSOUT=*
```

BASIC | * FREQUENTLY ASKED

Q5. What is DISP? Explain its three sub-parameters.

DISP=(status,normal,abnormal).
Status: NEW (create it), OLD (existing, exclusive use), SHR (existing, shared read) and MOD (append, or create if missing).
Normal (step ends well) and **abnormal** (step fails): CATLG, UNCATLG, KEEP, DELETE; PASS is also allowed for normal and passes the dataset to a later step.
A NEW dataset with no disposition given is deleted at the end of the step. Typical choice for output: **(NEW,CATLG,DELETE)**.

```
//OUT1 DD DSN=USER1.RPT.DATA,DISP=(NEW,CATLG,DELETE),
//         SPACE=(TRK,(10,5),RLSE)
//TMP  DD DSN=&&WORK,DISP=(NEW,PASS),SPACE=(CYL,(1,1))
```

BASIC | * FREQUENTLY ASKED

Q6. What is the difference between a PS file, a PDS and a PDSE?

A **PS** (physical sequential) dataset is a flat file read in order. A **PDS** (partitioned dataset) is a library with a directory and many members, such as COBOL source, copybooks, JCL or load modules; you refer to a member as DSN(MEMBER). Deleted or replaced members leave unusable gaps, so a PDS must be compressed from time to time (for example with IEBCOPY). A **PDSE** is the newer library type: no compress needed, directory grows automatically.

```
//SRC DD DSN=USER1.SRC.COBOL(EMPPGM),DISP=SHR
//NEWLIB DD DSN=USER1.MY.LIB,DISP=(NEW,CATLG),
//          SPACE=(CYL,(5,1,20)),
//          DCB=(RECFM=FB,LRECL=80,DSORG=PO)
```

BASIC

Q7. Explain the SPACE and DCB parameters.

SPACE=(unit,(primary,secondary,directory),RLSE): unit is TRK, CYL or a block size; primary is the first allocation, secondary is added when needed (limited number of extents); directory blocks are only for a PDS; RLSE frees unused space when the file closes.
DCB: RECFM (F, FB, V, VB), LRECL (record length), BLKSIZE (block size; 0 lets the system choose the best) and DSORG (PS or PO).

```
SPACE=(CYL,(10,5),RLSE)
DCB=(RECFM=FB,LRECL=100,BLKSIZE=0)
```

BASIC

Q8. How do you pass data inside the JCL itself? What are DD * and DD DATA?

DD * starts instream data that ends at a /* record (or at the next // statement). It is commonly used for utility control statements through SYSIN. **DD DATA** is used when the data itself contains lines starting with //, such as JCL stored in a file. The program reads the data like any other file through its ddname.

```
//SYSIN DD *
  SORT FIELDS=(1,5,CH,A)
/*
```

BASIC

Q9. What are DUMMY, SYSOUT, SYSPRINT and SYSUDUMP?

DD DUMMY is a null file: reads give end of file, writes are thrown away. **SYSOUT=*** sends output to the same class as MSGCLASS. **SYSPRINT** is the ddname many utilities use for messages. **SYSUDUMP** / **SYSABEND** / **SYSMDUMP** request a dump if the step abends, which helps in debugging.

```
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//OPTIONAL DD DUMMY
```

BASIC

Q10. What are JOBLIB and STEPLIB? How do you concatenate datasets?

JOBLIB is placed after the JOB statement and tells the system where to find load modules for all steps. **STEPLIB** does the same for one step and overrides JOBLIB. **Concatenation** means listing several datasets under one DD; the extra DD statements have no ddname, and they are searched or read in the order written. Concatenated input files should have compatible record formats.

```
//STEPLIB DD DSN=USER1.LOADLIB,DISP=SHR
//         DD DSN=SYS1.COBLIB,DISP=SHR
```

BASIC | * FREQUENTLY ASKED

Q11. What is IEFBR14 and how is it used?

IEFBR14 is a dummy program that does nothing and ends with return code 0. Because z/OS allocates and disposes datasets according to the DD statements, IEFBR14 is used just to **create** or **delete** datasets. DISP=(MOD,DELETE,DELETE) deletes a dataset and also works when the dataset does not exist, so the step does not fail on a rerun.

```
//DELSTEP EXEC PGM=IEFBR14
//DD1 DD DSN=USER1.OLD.DATA,DISP=(MOD,DELETE,DELETE),
//        SPACE=(TRK,(1,1))
//NEWSTEP EXEC PGM=IEFBR14
//DD2 DD DSN=USER1.NEW.DATA,DISP=(NEW,CATLG,DELETE),
//        SPACE=(CYL,(5,2)),DCB=(RECFM=FB,LRECL=80)
```

BASIC | * FREQUENTLY ASKED

Q12. What is IEBGENER and what are its DD names?

IEBGENER copies a sequential file (or a PDS member) to another file, or prints it to the spool; it can also create a member in a PDS. DD names: **SYSUT1** (input), **SYSUT2** (output), **SYSPRINT** (messages) and **SYSIN** (control statements; DUMMY when you only want a plain copy). Use SYSUT2 DD SYSOUT=* to browse a file in the job output.

```
//COPY1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=USER1.IN.FILE,DISP=SHR
//SYSUT2   DD DSN=USER1.OUT.FILE,DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(5,5),RLSE),DCB=*.SYSUT1
```

BASIC

Q13. What is IEBCOPY and when do you use it?

IEBCOPY copies, merges, compresses and reloads **partitioned** datasets (PDS/PDSE), including load libraries. Use SELECT MEMBER to copy only some members, and use the same dataset as input and output to compress a PDS in place. DD names are your own and are named in the control statement (INDD / OUTDD).

```
//COPY EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//IN  DD DSN=USER1.SRC.OLD,DISP=SHR
//OUT DD DSN=USER1.SRC.NEW,DISP=OLD
//SYSIN DD *
  COPY INDD=IN,OUTDD=OUT
  SELECT MEMBER=(EMPPGM,DEPTPGM)
/*
```

BASIC

Q14. Name other commonly used IBM utilities and what they do.

IDCAMS: VSAM and catalog commands (DEFINE, REPRO, DELETE, LISTCAT, PRINT). **IEBCOMPR**: compares two datasets. **IEBUPDTE**: creates or updates members from control statements. **IEHLIST**: lists catalog or VTOC information. **IEBPTPCH**: prints or punches records. **SORT / ICETOOL**: sorting and data manipulation. **IKJEFT01**: runs TSO commands in batch, for example DB2 programs. **ADRDSSU**: backup and restore.

BASIC | * FREQUENTLY ASKED

Q15. What is the SORT program (DFSORT/SYNCSORT) and how do you run a basic sort in JCL?

SORT is the system sort utility: it sorts, merges, copies, selects, reformats and summarizes records with simple control statements, without writing a COBOL program. Key DD names: **SORTIN** (input), **SORTOUT** (output), **SYSOUT** (messages) and **SYSIN** (control statements). Work files (SORTWKnn) are normally allocated dynamically by the sort product. Syntax is nearly identical in DFSORT and SYNCSORT for the common statements.

```
//SORT1   EXEC PGM=SORT
//SYSOUT  DD SYSOUT=*
//SORTIN  DD DSN=USER1.EMP.DATA,DISP=SHR
//SORTOUT DD DSN=USER1.EMP.SORTED,DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(5,2),RLSE)
//SYSIN   DD *
  SORT FIELDS=(1,5,CH,A)
/*
```

BASIC | * FREQUENTLY ASKED

Q16. What is a procedure (PROC)? What is the difference between an instream and a catalogued procedure?

A procedure is a saved block of JCL that can be reused by many jobs, which saves coding and keeps standards consistent. An **instream** procedure is defined inside the job, between PROC and PEND statements. A **catalogued** procedure is stored as a member of a procedure library (PROCLIB) and called by name; use JCLLIB ORDER to tell the job which library to search. A proc can have symbolic parameters with defaults.

```
//MYPROC PROC HLQ=USER1
//S1  EXEC PGM=EMPPGM
//STEPLIB DD DSN=&HLQ..LOADLIB,DISP=SHR
//INFILE  DD DSN=&HLQ..EMP.DATA,DISP=SHR
//    PEND
//RUN1 EXEC MYPROC,HLQ=TEST
```

BASIC | * FREQUENTLY ASKED

Q17. Name common JCL errors and abend codes and their causes.

JCL error: syntax or reference problem, job does not run (see IEF messages). **S806**: load module not found. **S0C7**: bad numeric data. **S0C4**: protection exception (bad address, subscript or parameter). **SB37 / SD37 / SE37**: no more space (IEC030I / IEC031I / IEC032I): out of primary/secondary or extents, secondary not specified, or no more volumes. **S222**: cancelled by operator. **S322**: CPU time limit exceeded (often a loop). **S913**: not authorized for the dataset. **S013**: OPEN error such as DCB conflict or missing member. **IEF212I**: dataset not found.

INTERMEDIATE | * FREQUENTLY ASKED

Q18. What is the COND parameter? How does IF/THEN/ELSE compare?

COND=(code,operator) on the EXEC statement is tested against the return codes of earlier steps, and the step is **bypassed if the test is true**. COND=(4,LT) means: skip this step if 4 is less than a previous return code, that is, run only when earlier codes are 4 or lower. COND=EVEN runs the step even after an abend; COND=ONLY runs it only after an abend. The newer **IF/THEN/ELSE/ENDIF** construct states the logic the natural way (run when true) and is easier to read.

```
//IF1  IF (STEP01.RC <= 4) THEN
//STEP02 EXEC PGM=REPORT1
//     ELSE
//STEP03 EXEC PGM=ERRPGM
//     ENDIF
```

INTERMEDIATE | * FREQUENTLY ASKED

Q19. Explain SORT FIELDS: position, length, format and order.

SORT FIELDS=(start,length,format,order). Start is the byte position, counted from 1 (for variable-length records the data starts at position 5 because of the 4-byte record descriptor word). Common formats: **CH** character, **ZD** zoned decimal, **PD** packed decimal, **BI** binary, **FS** signed numeric. Order is **A** (ascending) or **D** (descending). Repeat the four values for each additional key, and use FIELDS=COPY to copy without sorting.

```
* Sort by dept (1-3) ascending, then salary (10-17, zoned) descending
  SORT FIELDS=(1,3,CH,A,10,8,ZD,D)
```

INTERMEDIATE | * FREQUENTLY ASKED

Q20. How do you select or drop records in SORT using INCLUDE and OMIT?

INCLUDE COND=(...) keeps only the records for which the condition is true; **OMIT COND=(...)** removes them. Conditions have the form (position,length,format,operator,value). Operators: EQ, NE, GT, GE, LT, LE. Constants are written C'text', X'hex' or as numbers. Join conditions with AND / OR. A range test can use SS (substring) or, for numeric ranges, GE and LE together.

```
SORT FIELDS=(1,5,CH,A)
INCLUDE COND=(20,3,CH,EQ,C'CHE',AND,30,5,ZD,GT,1000)
```

INTERMEDIATE

Q21. What are INREC and OUTREC? How do you reformat records?

They change the layout of records. **INREC** reformats before the sort; **OUTREC** reformats after the sort (just before writing). **BUILD** (or FIELDS) lists what the new record contains: input fields as (position,length), constants like C'ACTIVE', and n X for blanks or nZ for zeros. You can drop, reorder and pad fields, for example to produce a report layout or match a COBOL record.

```
SORT FIELDS=(1,5,CH,A)
OUTREC BUILD=(1,5,3X,20,10,C'ACTIVE')
```

INTERMEDIATE | * FREQUENTLY ASKED

Q22. How do you remove duplicate records, or total up values, using SUM?

SUM FIELDS=NONE removes records with duplicate sort keys and keeps one (the first) of each. **SUM FIELDS=(position,length,format)** adds a numeric field across records with the same key and writes one record per key, so it also gives totals per key. The summed field must not be part of the sort key, and the format should match the data (ZD, PD, BI, FS).

```
* remove duplicates
  SORT FIELDS=(1,5,CH,A)
  SUM FIELDS=NONE
* total amount per key
  SORT FIELDS=(1,5,CH,A)
  SUM FIELDS=(20,7,ZD)
```

INTERMEDIATE

Q23. How do you merge or concatenate files with SORT?

MERGE FIELDS combines files that are **already sorted** on the same key into one sorted file, with each input on its own DD (SORTIN01, SORTIN02 and so on). To simply join files without ordering, concatenate the inputs under SORTIN and use **SORT FIELDS=COPY** (or OPTION COPY).

```
//SORTIN01 DD DSN=USER1.FILE.A,DISP=SHR
//SORTIN02 DD DSN=USER1.FILE.B,DISP=SHR
//SORTOUT  DD DSN=USER1.FILE.MERGED,DISP=(NEW,CATLG,DELETE),
//            SPACE=(CYL,(5,2),RLSE)
//SYSIN DD *
  MERGE FIELDS=(1,5,CH,A)
/*
```

INTERMEDIATE | * FREQUENTLY ASKED

Q24. How do you split one file into several using OUTFIL? Can you add headers and trailers?

OUTFIL writes output files with their own selection and layout. Give each its own DD with **FNAMES=** and filter with INCLUDE or OMIT. **SAVE** sends all records not chosen by other OUTFIL statements to one file. It can also add **HEADER1** and **TRAILER1** lines and record counts, which makes it handy for simple reports.

```
SORT FIELDS=COPY
OUTFIL FNAMES=OUT1,INCLUDE=(30,2,CH,EQ,C'TN')
OUTFIL FNAMES=OUT2,SAVE

OUTFIL FNAMES=RPT,HEADER1=('EMPLOYEE LIST'),
   TRAILER1=('TOTAL: ',COUNT=(M11,LENGTH=5))
```

INTERMEDIATE

Q25. What is ICETOOL? Give an example.

ICETOOL is a DFSORT utility that runs several operations in one step, each described by one control statement in TOOLIN: COPY, SORT, COUNT, SELECT, SPLICE, STATS, OCCUR and more. It needs TOOLMSG and DFSMSG for messages, and its own DD names for the files used. A frequent use is finding duplicate keys with SELECT ... ALLDUPS, or counting records with COUNT.

```
//STEP1 EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG  DD SYSOUT=*
//IN   DD DSN=USER1.EMP.DATA,DISP=SHR
//DUPS DD DSN=USER1.EMP.DUPS,DISP=(NEW,CATLG,DELETE),
//        SPACE=(CYL,(1,1),RLSE)
//TOOLIN DD *
  SELECT FROM(IN) TO(DUPS) ON(1,5,CH) ALLDUPS
/*
```

INTERMEDIATE

Q26. What are symbolic parameters, the SET statement and JCLLIB?

A **symbolic parameter** is a name starting with & whose value is substituted when the JCL is processed. The value can come from the proc's default, from the EXEC statement, or from a SET statement. When a symbolic is followed by a dot or another qualifier, code two dots (&HLQ..DATA) because the first dot ends the name. **JCLLIB ORDER=(...)** gives the libraries to search for procedures and INCLUDE members.

```
//JOBLIB1 JCLLIB ORDER=(USER1.PROCLIB)
//   SET HLQ=USER1
//S1 EXEC PGM=EMPPGM
//IN DD DSN=&HLQ..EMP.DATA,DISP=SHR
```

INTERMEDIATE | * FREQUENTLY ASKED

Q27. How do you override or add a DD, PARM or COND in a procedure?

Override on the EXEC statement that calls the proc: use **stepname.ddname** for a DD, **PARM.stepname=** for PARM and **COND.stepname=** for COND. Overrides must be coded in the same order as the steps in the procedure. A DD name that does not exist in the proc step is added to that step. To change only one parameter of an existing DD, code that DD again with the parameter you want to change.

```
//RUN1 EXEC MYPROC,PARM.S1='20261026'
//S1.INFILE DD DSN=USER1.OTHER.DATA,DISP=SHR
//S1.NEWDD  DD SYSOUT=*
```

INTERMEDIATE

Q28. How do you restart a job from a particular step?

Add **RESTART=stepname** on the JOB statement (for a step in a procedure: RESTART=procstep.stepname), fix the problem, and resubmit. The job starts at that step and skips earlier ones. Before restarting, check which datasets earlier steps created or passed: temporary datasets (&&name) and PASS dispositions are lost, and for GDG output the relative generation (+1) must be handled carefully so you do not create a second new generation.

```
//EMPJOB1 JOB (ACCT),'VENKAT',CLASS=A,MSGCLASS=X,
//         RESTART=STEP03
```

SCENARIO | * FREQUENTLY ASKED

Q29. Scenario: your job abends with SB37 in a step that creates an output file. What do you do?

1) Open the job log and read IEC030I: it names the dataset and the step.
2) Find the cause: primary and secondary space too small for the data, extent limit reached, or the volume is full.
3) Estimate the real size (records x record length) and raise SPACE, using a larger primary and a secondary, with RLSE to release the unused part.
4) The abnormal disposition (DELETE) normally removes the partial file; confirm that, then rerun from the failed step with RESTART=stepname.

```
//SORTOUT DD DSN=USER1.BIG.OUT,DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(50,20),RLSE),UNIT=SYSDA
```

SCENARIO | * FREQUENTLY ASKED

Q30. Scenario: write the JCL to run a COBOL-DB2 batch program.

DB2 programs in batch run under the TSO batch program **IKJEFT01**. The SYSTSIN statements connect to the DB2 subsystem and RUN the program with its PLAN. STEPLIB must contain your load library, and also the DB2 load library unless it is in the system link list. File ddnames for the program are added after STEPLIB; messages go to SYSTSPRT.

```
//DB2RUN  EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB DD DSN=DSN.SDSNLOAD,DISP=SHR
//        DD DSN=USER1.LOADLIB,DISP=SHR
//EMPIN   DD DSN=USER1.EMP.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  DSN SYSTEM(DSN1)
  RUN PROGRAM(EMPPGM) PLAN(EMPPLAN) LIB('USER1.LOADLIB')
  END
/*
```

Section 2: VSAM

KSDS, IDCAMS, REPRO, LISTCAT, reorganization and GDG

BASIC | * FREQUENTLY ASKED

Q1. What is VSAM? Name its dataset types.

VSAM (Virtual Storage Access Method) is IBM's high-performance file system for keyed and direct access. VSAM datasets are called **clusters** and are created with IDCAMS (not with DISP=NEW in JCL). Types: **KSDS** (key-sequenced, with an index), **ESDS** (entry-sequenced, records added at the end), **RRDS** (relative record, accessed by record number) and **LDS** (linear, a byte stream used by DB2). All VSAM datasets are cataloged.

BASIC | * FREQUENTLY ASKED

Q2. Explain KSDS: key, index and data components.

A KSDS cluster has a **data component** (the records, kept in key order) and an **index component** (which records the highest key in each control interval, so a key can be found quickly). Each record has a unique key at a fixed position and length (max 255). Records can be read sequentially, randomly by key, or dynamically (both). On REWRITE the primary key cannot be changed; to change it you delete and add the record.

BASIC

Q3. What are a control interval (CI) and a control area (CA)? What is FREESPACE?

The **CI** is the unit of data VSAM transfers between disk and memory (for example 4096 bytes); it holds several records, free space and control information. A **CA** is a group of CIs, often one cylinder. **FREESPACE(ci% ca%)** reserves part of each CI and some free CIs in each CA at load time, so later inserts do not cause splits. More free space means fewer splits but a bigger file.

BASIC | * FREQUENTLY ASKED

Q4. Write an IDCAMS DEFINE CLUSTER for a KSDS.

Run IDCAMS with SYSPRINT and SYSIN. **KEYS(length offset)** gives the key length and its offset (0 for the first byte), **RECORDSIZE(average maximum)** gives record lengths, FREESPACE reserves room for inserts, and CYLINDERS(primary secondary) gives space. DATA and INDEX names are optional.

```
//DEFKSDS EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEFINE CLUSTER (NAME(USER1.EMP.KSDS)  -
         INDEXED                         -
         KEYS(5 0)                       -
         RECORDSIZE(80 80)               -
         FREESPACE(10 10)                -
         CYLINDERS(5 1)                  -
         SHAREOPTIONS(2 3))              -
     DATA  (NAME(USER1.EMP.KSDS.DATA))   -
     INDEX (NAME(USER1.EMP.KSDS.INDEX))
/*
```

BASIC | * FREQUENTLY ASKED

Q5. What does REPRO do? How do you load a KSDS from a sequential file?

REPRO copies records: sequential to VSAM, VSAM to sequential (a backup or unload), VSAM to VSAM. Use INFILE/OUTFILE with DD names, or INDATASET/OUTDATASET with dataset names. Useful options: FROMKEY/TOKEY, COUNT, SKIP, REPLACE (overwrite records with the same key) and REUSE. When loading an **empty KSDS**, the input must be in **ascending key order** and without duplicate keys, so sort the file first.

```
//LOAD EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//IN  DD DSN=USER1.EMP.SORTED,DISP=SHR
//OUT DD DSN=USER1.EMP.KSDS,DISP=SHR
//SYSIN DD *
  REPRO INFILE(IN) OUTFILE(OUT)
/*
```

BASIC | * FREQUENTLY ASKED

Q6. What is LISTCAT and what information does it give?

LISTCAT lists catalog entries. **LISTCAT ENTRIES(name) ALL** shows a dataset's details; **LISTCAT LEVEL(hlq)** lists every dataset under a high-level qualifier. For a KSDS it shows key length and offset, record size, CI size, free space, number of records, extents, and the counts of **CI splits and CA splits**. For a GDG it shows the generations. Options such as NAME, VOLUME, ALLOCATION and HISTORY limit the output.

```
//LIST EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  LISTCAT ENTRIES(USER1.EMP.KSDS) ALL
  LISTCAT LEVEL(USER1.SALES)
/*
```

BASIC

Q7. How do you delete a VSAM cluster? What do PURGE and SCRATCH mean?

Use the IDCAMS DELETE command with the CLUSTER keyword. **PURGE** deletes even if the retention period has not expired. **SCRATCH** (default) removes the dataset from the volume; NOSCRATCH only removes the catalog entry. ERASE overwrites the data. A GDG base needs the GDG name (and FORCE if generations still exist).

```
DELETE USER1.EMP.KSDS CLUSTER PURGE
DELETE USER1.SALES.GDG GDG FORCE
```

BASIC

Q8. Compare KSDS, ESDS and RRDS.

KSDS: records have a unique key, with an index; can be accessed by key, inserted in the middle, and records can change length; the most common type. **ESDS**: records are added at the end in arrival order, no key, accessed by the relative byte address (RBA); cannot delete records. **RRDS**: fixed-length slots numbered 1 to n, accessed directly by record number; fast but needs a numeric key. Alternate indexes can add key access to an ESDS or KSDS.

BASIC | * FREQUENTLY ASKED

Q9. What is a GDG? How is a GDG base defined?

A Generation Data Group is a set of related datasets that share one base name and are numbered chronologically, so each run can create a new version and older ones are kept (for daily or monthly files). Each generation is named **base.GnnnnVnn**. A GDG base is only a catalog entry (it is created with IDCAMS but is not VSAM). **LIMIT** is the maximum number of generations kept. **SCRATCH** physically deletes a generation when it rolls off; NOSCRATCH only uncatalogs it. **EMPTY** removes all generations when the limit is reached; NOEMPTY removes just the oldest.

```
DEFINE GDG (NAME(USER1.SALES.GDG) -
       LIMIT(7) NOEMPTY SCRATCH)
```

BASIC | * FREQUENTLY ASKED

Q10. How do you refer to GDG generations in JCL?

Use a relative number: **(0)** is the latest generation, **(-1)** the one before it, and **(+1)** a new generation to be created. You can also use the absolute name (G0005V00). Referring to the base name alone reads all generations as a concatenation. The relative numbers are resolved when the job starts, so every step in the same job that uses (+1) means the same new generation, and the job's own new generation becomes (0) only for later jobs.

```
//NEWGEN DD DSN=USER1.SALES.GDG(+1),DISP=(NEW,CATLG,DELETE),
//          SPACE=(CYL,(5,2),RLSE),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//LATEST DD DSN=USER1.SALES.GDG(0),DISP=SHR
//PREV   DD DSN=USER1.SALES.GDG(-1),DISP=SHR
//ALLGEN DD DSN=USER1.SALES.GDG,DISP=SHR
```

INTERMEDIATE

Q11. What is an alternate index (AIX)? How is it created?

An AIX lets you access a KSDS or ESDS by a field other than the primary key. Steps: **DEFINE AIX** (the alternate key, RELATE to the base cluster, UNIQUEKEY or NONUNIQUEKEY, UPGRADE to keep it updated), **DEFINE PATH** (the name used to reach the data through the AIX), then **BLDINDEX** to build the index from existing records. Programs and JCL then open the PATH name; in COBOL declare it with ALTERNATE RECORD KEY.

```
DEFINE AIX (NAME(USER1.EMP.AIX) RELATE(USER1.EMP.KSDS) -
     KEYS(20 10) NONUNIQUEKEY UPGRADE CYLINDERS(1 1))
DEFINE PATH (NAME(USER1.EMP.PATH) PATHENTRY(USER1.EMP.AIX))
BLDINDEX INDATASET(USER1.EMP.KSDS) OUTDATASET(USER1.EMP.AIX)
```

INTERMEDIATE | * FREQUENTLY ASKED

Q12. What is VSAM reorganization (REORG) and how is it done?

Many inserts and deletes cause CI and CA splits and use up free space, which makes access slower and the file bigger (see the split counts in LISTCAT). **Reorganizing** rebuilds the file: (1) REPRO the cluster to a sequential backup, (2) DELETE the cluster, (3) DEFINE it again with proper FREESPACE, (4) REPRO the backup back in. Unloading a KSDS gives records in key order, so no sort is required. Take care to keep the backup until the reload has succeeded.

```
REPRO INDATASET(USER1.EMP.KSDS) OUTDATASET(USER1.EMP.BKP)
DELETE USER1.EMP.KSDS CLUSTER PURGE
DEFINE CLUSTER (NAME(USER1.EMP.KSDS) INDEXED KEYS(5 0) -
       RECORDSIZE(80 80) FREESPACE(15 10) CYLINDERS(5 1))
REPRO INDATASET(USER1.EMP.BKP) OUTDATASET(USER1.EMP.KSDS)
```

INTERMEDIATE

Q13. What is a CI split and a CA split? Why are they costly?

A **CI split** happens when a record must be inserted into a full CI: VSAM moves about half the records to a free CI in the same CA and updates the index. A **CA split** happens when there is no free CI in the CA: VSAM moves about half the CIs to a new CA. Both require extra I/O and make the physical order differ from the key order, which slows sequential processing. Prevent them with enough FREESPACE and by reorganizing regularly.

INTERMEDIATE | * FREQUENTLY ASKED

Q14. What are the common file status codes when working with VSAM in COBOL?

00 success. **10** end of file. **21** key sequence error (sequential write out of order). **22** duplicate key. **23** record not found. **35** file not found at OPEN. **37** open mode not permitted. **39** file attributes in the program do not match the cluster (record length, key). **41** already open. **43** REWRITE or DELETE without a successful READ first. **46** READ after end of file or no valid next record. VSAM-specific 9x codes (for example 92) mean a logic error or unavailable resource, so check the return and feedback codes.

INTERMEDIATE | * FREQUENTLY ASKED

Q15. What are IDCAMS return codes? How do you use MAXCC and LASTCC?

IDCAMS ends with **0** (success), **4** (warning), **8** (error in a command, for example deleting a dataset that does not exist), **12** (severe) or **16** (fatal). LASTCC is the code of the last command and MAXCC the highest so far. **SET MAXCC=0** resets the job's return code, so a harmless failure does not stop the next step, and IF LASTCC can control what runs next. This is the usual trick to make a delete-and-define job rerunnable.

```
DELETE USER1.EMP.KSDS CLUSTER PURGE
SET MAXCC=0
DEFINE CLUSTER (NAME(USER1.EMP.KSDS) INDEXED KEYS(5 0) -
       RECORDSIZE(80 80) CYLINDERS(5 1))
```

INTERMEDIATE

Q16. What problems can occur when rerunning a job that creates a GDG generation?

If a job creates (+1) and then fails after the dataset was cataloged, a plain rerun creates one more generation, and old generations may be pushed out (rolled off) of the base. Before rerunning, check with LISTCAT which generation exist, delete an incomplete one, or restart from the right step. Within one job, remember to read the new generation as (+1), not (0). Also, a new generation needs DCB attributes (code them in the DD or use a model dataset).

INTERMEDIATE

Q17. How do you print a VSAM record and rename or alter a dataset?

PRINT shows records from a dataset in the job output: CHARACTER, HEX or DUMP format, with COUNT, SKIP or FROMKEY/TOKEY to limit them. **ALTER** changes catalog attributes: rename with NEWNAME, change GDG LIMIT, FREESPACE, or add volumes.

```
PRINT INDATASET(USER1.EMP.KSDS) CHARACTER COUNT(10)
ALTER USER1.SALES.GDG LIMIT(10)
ALTER USER1.OLD.NAME NEWNAME(USER1.NEW.NAME)
```

INTERMEDIATE

Q18. What is SHAREOPTIONS in a VSAM definition?

SHAREOPTIONS(region system) says how many users can access the cluster together. Region value: **1** multiple readers or one updater, **2** multiple readers and one updater, **3** full sharing with no integrity checks by VSAM, **4** full sharing with buffers refreshed. The system value (3 or 4) is for sharing across systems. (2 3) is a typical default; the application must manage integrity when higher sharing is chosen.

SCENARIO | * FREQUENTLY ASKED

Q19. Scenario: your COBOL program gets file status 39 when it opens a KSDS. How do you find the cause?

Status 39 means the program's file description does not match the VSAM cluster. Check with LISTCAT: record length against the program's record size, and the key length and offset against RECORD KEY in the program (an offset or length mismatch is typical). Also confirm the JCL ddname points to the intended VSAM cluster and not to a flat file or a different dataset. Correct the program or the cluster definition and rerun.

SCENARIO

Q20. Scenario: a REPRO into a new KSDS fails because the input file is not in key sequence. How do you fix it?

An empty KSDS must be loaded in ascending key order, with no duplicate keys. Sort the sequential file on the key (and remove duplicates with SUM FIELDS=NONE if needed), then run REPRO again. A key sequence error or duplicate key message in IDCAMS output (return code 8 or higher) tells you which record is wrong. Never skip this step by relying on file order from an earlier extract.

```
//SORT1 EXEC PGM=SORT
//SYSOUT DD SYSOUT=*
//SORTIN DD DSN=USER1.EMP.RAW,DISP=SHR
//SORTOUT DD DSN=USER1.EMP.SORTED,DISP=(NEW,CATLG,DELETE),
//           SPACE=(CYL,(5,2),RLSE)
//SYSIN DD *
  SORT FIELDS=(1,5,CH,A)
  SUM FIELDS=NONE
/*
```

Section 3: COBOL

Divisions, data, control flow, files, tables, subprograms, abends

BASIC | * FREQUENTLY ASKED

Q1. What are the four divisions of a COBOL program?

IDENTIFICATION DIVISION names the program (PROGRAM-ID). **ENVIRONMENT** DIVISION describes the system and the files (SELECT / ASSIGN). **DATA** DIVISION declares variables and file layouts (FILE, WORKING-STORAGE and LINKAGE sections). **PROCEDURE** DIVISION holds the executable logic. They must appear in this order.

BASIC | * FREQUENTLY ASKED

Q2. What do 9, X, A, S and V mean in a PICTURE (PIC) clause?

9 = numeric digit, X = any character, A = alphabetic only, S = sign of the number (no extra byte in DISPLAY usage), V = implied decimal point (takes no storage). The field below holds a value like -12345.67 in 7 bytes.

```
05 WS-AMOUNT   PIC S9(5)V99.
05 WS-NAME     PIC X(20).
```

BASIC | * FREQUENTLY ASKED

Q3. What is the difference between COMP and COMP-3?

COMP (binary) keeps the number as a binary value: 1-4 digits use 2 bytes, 5-9 digits use 4 bytes, 10-18 digits use 8 bytes. **COMP-3** (packed decimal) stores two digits per byte with the sign in the last half-byte, so size = (digits + 1) / 2, rounded up. Plain DISPLAY usage takes one byte per digit. COMP-3 is very common for money fields.

```
05 WS-A  PIC S9(4)     COMP.      (2 bytes)
05 WS-B  PIC S9(7)V99  COMP-3.    (5 bytes)
```

BASIC | * FREQUENTLY ASKED

Q4. Explain level numbers 01, 05, 66, 77 and 88.

01 is a record (group) item and 02-49 are items under it. 66 is used with RENAMES. 77 is an independent elementary item that does not belong to any group. 88 is a **condition name**: a named TRUE/FALSE test on the value of the field above it, which makes IF statements readable.

```
05 WS-STATUS  PIC X.
   88 ACTIVE-REC  VALUE 'A'.
   88 CLOSED-REC  VALUE 'C'.
...
IF ACTIVE-REC
   PERFORM 2000-PROCESS-REC
END-IF.
```

BASIC | * FREQUENTLY ASKED

Q5. What are the different forms of the PERFORM statement?

Simple PERFORM paragraph, PERFORM ... THRU ..., PERFORM n TIMES, PERFORM UNTIL condition (condition is tested before each pass by default; WITH TEST AFTER makes it a do-while), PERFORM VARYING for counted loops, and an inline PERFORM ... END-PERFORM that needs no separate paragraph.

```
PERFORM 2000-PROCESS UNTIL WS-EOF = 'Y'.

PERFORM VARYING WS-I FROM 1 BY 1 UNTIL WS-I > 10
    ADD WS-I TO WS-TOTAL
END-PERFORM.
```

BASIC

Q6. What is the EVALUATE statement?

EVALUATE is COBOL's case/switch statement. It compares a subject with each WHEN value and runs the first match. WHEN OTHER is the default branch. It is cleaner than a long chain of nested IFs.

```
EVALUATE WS-GRADE
   WHEN 'A'    MOVE 'EXCELLENT' TO WS-MSG
   WHEN 'B'    MOVE 'GOOD'      TO WS-MSG
   WHEN OTHER  MOVE 'CHECK'     TO WS-MSG
END-EVALUATE.
```

BASIC

Q7. What is the difference between STOP RUN, GOBACK and EXIT PROGRAM?

STOP RUN ends the whole run unit, even when issued in a subprogram. **GOBACK** returns to the caller, and ends the run if it is issued in the main program. **EXIT PROGRAM** only works in a called subprogram; in a main program it does nothing. Use GOBACK in subprograms. In a CICS program, use EXEC CICS RETURN instead.

BASIC

Q8. What file organizations and access modes does COBOL support?

Organizations: **SEQUENTIAL** (records read in order, for example PS files), **INDEXED** (VSAM KSDS, accessed by key) and **RELATIVE** (VSAM RRDS, accessed by record number). Access mode can be SEQUENTIAL, RANDOM or DYNAMIC (both).

```
SELECT CUST-FILE ASSIGN TO CUSTDD
    ORGANIZATION IS INDEXED
    ACCESS MODE  IS DYNAMIC
    RECORD KEY   IS CUST-ID
    FILE STATUS  IS WS-FS.
```

BASIC | * FREQUENTLY ASKED

Q9. What is FILE STATUS? Name some common file status codes.

FILE STATUS is a 2-byte field COBOL updates after every file operation, so the program can check for errors. **00** success, **10** end of file, **22** duplicate key, **23** record not found, **35** file not found at OPEN (for example missing DD), **39** file attributes mismatch (LRECL/RECFM), **41** file already open, **46** read after end of file.

BASIC | * FREQUENTLY ASKED

Q10. Write the basic logic to read a sequential file until end of file.

Code the SELECT/ASSIGN and an FD, OPEN INPUT, do a priming READ, loop until end of file reading the next record at the end of each pass, then CLOSE the file.

```
OPEN INPUT IN-FILE.
READ IN-FILE AT END MOVE 'Y' TO WS-EOF END-READ.
PERFORM UNTIL WS-EOF = 'Y'
    PERFORM 2000-PROCESS-REC
    READ IN-FILE AT END MOVE 'Y' TO WS-EOF END-READ
END-PERFORM.
CLOSE IN-FILE.
```

BASIC | * FREQUENTLY ASKED

Q11. What is the difference between MOVE and MOVE CORRESPONDING?

MOVE copies one field or group to another. Alphanumeric data is left-justified and padded with spaces on the right (or truncated on the right); numeric data is aligned on the decimal point (extra high-order digits are truncated). **MOVE CORRESPONDING** copies every elementary item that has the **same name** in both groups, which is useful for similar input and output records. Use it with care: a name change in one group silently changes what is copied.

```
MOVE CORRESPONDING IN-EMP-REC TO OUT-EMP-REC.
```

BASIC

Q12. Explain the arithmetic statements and the ROUNDED and ON SIZE ERROR phrases.

ADD, SUBTRACT, MULTIPLY, DIVIDE (with REMAINDER) and COMPUTE do arithmetic; COMPUTE takes a full formula. **ROUNDED** rounds the result to the receiving field's decimals instead of truncating. **ON SIZE ERROR** runs your code when the result is too large for the field (or divide by zero). Without it, high-order digits are silently lost, which is a classic source of wrong totals.

```
COMPUTE WS-NET ROUNDED = WS-GROSS * (1 - WS-TAX-RATE)
    ON SIZE ERROR DISPLAY 'NET AMOUNT OVERFLOW'
END-COMPUTE.
```

BASIC | * FREQUENTLY ASKED

Q13. What are class conditions and sign conditions? Why test IS NUMERIC?

Class tests check the content of a field: IS NUMERIC, IS ALPHABETIC, IS ALPHABETIC-UPPER and the negative forms. Sign tests are POSITIVE, NEGATIVE and ZERO. Always test **IS NUMERIC** on input data (especially data read from a file) before doing arithmetic, because spaces or letters in a numeric field cause an S0C7 abend.

```
IF IN-QTY IS NUMERIC AND IN-QTY IS POSITIVE
    ADD IN-QTY TO WS-TOTAL-QTY
ELSE
    PERFORM 9100-REJECT-RECORD
END-IF.
```

BASIC | * FREQUENTLY ASKED

Q14. What is the difference between INITIALIZE and MOVE SPACES / MOVE ZEROS to a group?

INITIALIZE sets each elementary item by its type: numeric items to zero and alphanumeric items to spaces (FILLER items are skipped). MOVE SPACES to a group fills **everything** with spaces, including numeric fields, which then abend with S0C7 when used in arithmetic. So use INITIALIZE for a group that contains both types.

```
INITIALIZE WS-EMP-REC.
* not safe if WS-EMP-REC has numeric fields:
* MOVE SPACES TO WS-EMP-REC.
```

BASIC

Q15. What are figurative constants?

Reserved words standing for common values: **SPACE(S)**, **ZERO(S)**, **HIGH-VALUE(S)** (X'FF'), **LOW-VALUE(S)** (X'00'), **QUOTE(S)** and **ALL literal** (repeats the literal to fill the field). HIGH-VALUES is often used as an end-of-file key in match-merge logic.

```
MOVE HIGH-VALUES TO WS-MASTER-KEY.
MOVE ALL '*' TO WS-LINE.
MOVE ZEROS TO WS-TOTAL.
```

BASIC | * FREQUENTLY ASKED

Q16. Explain FILE SECTION, WORKING-STORAGE, LOCAL-STORAGE and LINKAGE SECTION.

FILE SECTION describes the records of the files named in SELECT (one FD per file). **WORKING-STORAGE** holds the program's own variables, initialized once with VALUE clauses. **LOCAL-STORAGE** is similar but is re-created and reinitialized on every call. **LINKAGE SECTION** describes data received from outside (a calling program, JCL PARM, or the CICS COMMAREA); no storage is allocated for it, it maps onto the caller's data.

BASIC | * FREQUENTLY ASKED

Q17. What is the difference between PERFORM and CALL?

PERFORM runs a paragraph or section **inside the same program**, sharing its data. CALL runs a **separate program** with its own data, passing parameters through USING. PERFORM is for structuring one program; CALL is for sharing a routine between many programs (for example a date-validation module).

BASIC

Q18. What is the difference between a paragraph and a section? What is fall-through?

A **paragraph** is a named group of sentences. A **section** is a named group of one or more paragraphs; PERFORM of a section runs all of them. **Fall-through** means control continues into the next paragraph when a paragraph ends without an explicit stop, because paragraphs are not separate routines. Keep a clear main paragraph that PERFORMs the others and then ends with GOBACK or STOP RUN.

```
0000-MAIN.
    PERFORM 1000-INIT.
    PERFORM 2000-PROCESS UNTIL WS-EOF = 'Y'.
    PERFORM 3000-TERMINATE.
    GOBACK.
```

BASIC | * FREQUENTLY ASKED

Q19. Explain editing (report) pictures: Z, *, $, comma, period, CR/DB.

Edited fields prepare numbers for printing: **Z** replaces leading zeros with spaces, ***** with asterisks (check protection), **$** inserts a currency sign, **, .** insert a comma and a real decimal point, **+ -** show the sign, **CR / DB** show credit or debit for negative values, and B, 0 or / insert a blank, zero or slash. Edited fields can be printed but **not used in arithmetic**; keep a numeric field for calculation and move it to the edited field for the report.

```
05 WS-AMT-OUT  PIC ZZ,ZZ9.99.
...
MOVE 1234.5 TO WS-AMT-OUT.
* result:  ' 1,234.50'
```

BASIC

Q20. What is the difference between READ INTO and WRITE FROM?

READ file INTO ws-area reads a record and also copies it to a WORKING-STORAGE area, so the data stays available after the next read. **WRITE record FROM ws-area** copies the WORKING-STORAGE data to the record area and then writes it. Remember: you READ the **file name** but WRITE the **record name**.

```
READ IN-FILE INTO WS-IN-REC
    AT END MOVE 'Y' TO WS-EOF
END-READ.
WRITE OUT-REC FROM WS-OUT-REC.
```

BASIC

Q21. What are the OPEN modes in COBOL?

INPUT: read an existing file. **OUTPUT**: create a file (overwrites existing contents). **I-O**: read and update records (required for REWRITE and DELETE). **EXTEND**: add records at the end of a sequential file. A file must be closed with CLOSE before the program ends.

```
OPEN INPUT  IN-FILE
     OUTPUT OUT-FILE.
OPEN I-O CUST-FILE.
OPEN EXTEND LOG-FILE.
```

BASIC | * FREQUENTLY ASKED

Q22. Explain the compile, link-edit and run steps of a COBOL program. Which compiler options matter?

Source is **compiled** (IGYCRCTL) into an object module, **link-edited** (IEWL) into a load module in a load library, and then run with **EXEC PGM=** and STEPLIB pointing to that library. A COBOL-DB2 program has a DB2 precompile step first. Compiler return codes: 0 clean, 4 warnings, 8 errors, 12 severe, 16 fatal. Useful options: **XREF** and **MAP** (cross reference and storage map), **LIST** / **OFFSET** (needed to find the failing statement from an abend offset), **SSRANGE** (runtime subscript checks) and **DYNAM / NODYNAM** (how CALL is resolved).

INTERMEDIATE | * FREQUENTLY ASKED

Q23. What is the OCCURS clause? Explain subscript vs index, and SEARCH vs SEARCH ALL.

OCCURS defines a table (array). A **subscript** is an occurrence number held in a numeric field. An **index** (INDEXED BY) holds a displacement, is faster, and is changed with SET. **SEARCH** is a linear search. **SEARCH ALL** is a binary search and needs the table sorted on the ASCENDING/DESCENDING KEY.

```
01 WS-TABLE.
   05 WS-ROW OCCURS 50 TIMES
         ASCENDING KEY IS WS-CODE
         INDEXED BY IDX.
      10 WS-CODE  PIC X(3).
      10 WS-DESC  PIC X(20).
...
SEARCH ALL WS-ROW
    AT END DISPLAY 'NOT FOUND'
    WHEN WS-CODE(IDX) = 'ABC'
         DISPLAY WS-DESC(IDX)
END-SEARCH.
```

INTERMEDIATE

Q24. What is the REDEFINES clause?

REDEFINES lets two or more data names share the same storage with different layouts. A typical use is to look at a date both as one 8-digit number and as year, month and day.

```
05 WS-DATE      PIC 9(8).
05 WS-DATE-R    REDEFINES WS-DATE.
   10 WS-YYYY   PIC 9(4).
   10 WS-MM     PIC 9(2).
   10 WS-DD     PIC 9(2).
```

INTERMEDIATE | * FREQUENTLY ASKED

Q25. Explain static vs dynamic CALL, and BY REFERENCE vs BY CONTENT.

A **static** call is link-edited into the caller's load module (CALL with a literal and the NODYNAM option). A **dynamic** call (CALL with a variable name, or the DYNAM option) loads the subprogram at run time, so it can be changed without relinking the caller. **BY REFERENCE** (default) passes the address, so the subprogram can change the caller's data. **BY CONTENT** passes a copy. The subprogram receives data through LINKAGE SECTION and PROCEDURE DIVISION USING.

```
CALL 'CALCTAX' USING BY REFERENCE WS-AMT
                     BY CONTENT   WS-RATE.

* in the called program CALCTAX:
PROCEDURE DIVISION USING LS-AMT LS-RATE.
```

INTERMEDIATE

Q26. How does the COBOL SORT statement work?

SORT orders records using a work file defined with SD. USING and GIVING do a straight sort of an input file to an output file. INPUT PROCEDURE (RELEASE) and OUTPUT PROCEDURE (RETURN) let you filter or change records during the sort. Many shops use the JCL SORT utility instead for simple sorts.

```
SORT SORT-FILE
    ON ASCENDING KEY SORT-CUST-ID
    USING  IN-FILE
    GIVING OUT-FILE.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q27. Name some common COBOL abend codes and their causes.

S0C7 data exception: invalid data in a numeric field (spaces or low-values used in arithmetic).
S0C4 protection exception: bad storage address, for example subscript out of range or wrong CALL parameters.
S0C1 operation exception: invalid instruction, for example calling something that is not a program.
S806 load module not found.
SB37 / SD37 out of space on a dataset.
S322 time limit exceeded, often an infinite loop.

INTERMEDIATE

Q28. What do STRING, UNSTRING and INSPECT do?

STRING joins several fields into one. UNSTRING splits one field into several using a delimiter. INSPECT counts (TALLYING) or replaces (REPLACING) characters inside a field.

```
UNSTRING WS-LINE DELIMITED BY ','
    INTO WS-ID WS-NAME WS-CITY
END-UNSTRING.
INSPECT WS-NAME REPLACING ALL '-' BY SPACE.
```

INTERMEDIATE

Q29. What is a copybook and how is it used?

A copybook is a reusable source member, usually data layouts, kept in a library. The COPY statement pulls it into the program at compile time, so many programs share one record layout and it is changed in a single place. Programs must be recompiled after a copybook changes.

```
WORKING-STORAGE SECTION.
COPY CUSTREC.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q30. How do you define a two-dimensional table and a variable-length table (OCCURS DEPENDING ON)?

Nest OCCURS clauses for two dimensions and use two subscripts. **OCCURS ... DEPENDING ON** defines a table whose logical size is held in a counter field; the program must set the counter before using the table and keep it within the maximum.

```
01 WS-SALES-TABLE.
   05 WS-REGION OCCURS 4 TIMES.
      10 WS-MONTH-SALES OCCURS 12 TIMES PIC 9(7)V99.
...
ADD WS-AMT TO WS-MONTH-SALES(WS-R, WS-M).

05 WS-CNT  PIC 9(3).
05 WS-ITEM OCCURS 1 TO 100 TIMES
           DEPENDING ON WS-CNT PIC X(10).
```

INTERMEDIATE | * FREQUENTLY ASKED

Q31. What are reference modification and intrinsic functions?

Reference modification takes part of a field: **field(start:length)**. **Intrinsic functions** are built-in functions called with FUNCTION, such as LENGTH, UPPER-CASE, LOWER-CASE, REVERSE, TRIM (newer compilers), NUMVAL (text to number), CURRENT-DATE and MAX/MIN. They are used inside MOVE, COMPUTE, IF and other statements.

```
MOVE WS-NAME(1:3) TO WS-SHORT-NAME.
MOVE FUNCTION UPPER-CASE(WS-NAME) TO WS-NAME-U.
COMPUTE WS-AMT = FUNCTION NUMVAL(WS-AMT-TEXT).
```

INTERMEDIATE | * FREQUENTLY ASKED

Q32. How do you get the current date and time in COBOL?

ACCEPT ... FROM DATE YYYYMMDD and **ACCEPT ... FROM TIME** are the simple forms. **FUNCTION CURRENT-DATE** returns a 21-character string: YYYYMMDD, hhmmsshh and the offset from GMT; use reference modification to take the part you need.

```
ACCEPT WS-DATE FROM DATE YYYYMMDD.
ACCEPT WS-TIME FROM TIME.
MOVE FUNCTION CURRENT-DATE(1:8) TO WS-RUN-DATE.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q33. How does a COBOL program receive the PARM value passed from JCL?

Through the LINKAGE SECTION: the parameter arrives as a halfword binary **length** followed by the text (up to 100 characters), and the PROCEDURE DIVISION is coded with USING.

```
LINKAGE SECTION.
01 LS-PARM.
   05 LS-PARM-LEN  PIC S9(4) COMP.
   05 LS-PARM-DATA PIC X(8).
PROCEDURE DIVISION USING LS-PARM.
    MOVE LS-PARM-DATA TO WS-RUN-DATE.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q34. How do you read a VSAM KSDS randomly? What does START do?

Open the file with ACCESS MODE RANDOM or DYNAMIC, move the key to the RECORD KEY field and READ; **INVALID KEY** runs when no record has that key (file status 23). **START** positions the file at a key (=, >, >=) and READ NEXT then reads sequentially from there, for example to list all customers from a given ID.

```
MOVE WS-ID TO CUST-ID.
READ CUST-FILE
    INVALID KEY DISPLAY 'NOT FOUND'
    NOT INVALID KEY PERFORM 2000-SHOW
END-READ.

MOVE 1000 TO CUST-ID.
START CUST-FILE KEY IS >= CUST-ID
    INVALID KEY DISPLAY 'NO RECORDS'
END-START.
READ CUST-FILE NEXT RECORD AT END MOVE 'Y' TO WS-EOF END-READ.
```

INTERMEDIATE

Q35. Explain WRITE, REWRITE and DELETE and the rules for using them.

WRITE adds a record (sequential files: in OUTPUT or EXTEND mode; KSDS: key must be unique, otherwise file status 22). **REWRITE** replaces the record that was **last read**; the file must be open I-O, the record length may not change for sequential files and the primary key may not change for KSDS (otherwise status 43 or 21 can occur). **DELETE** removes the last-read record of an indexed or relative file.

```
READ CUST-FILE INVALID KEY PERFORM 9200-NOT-FOUND END-READ.
MOVE WS-NEW-STATUS TO CUST-STATUS.
REWRITE CUST-REC
    INVALID KEY DISPLAY 'REWRITE FAILED ' WS-FS
END-REWRITE.
```

INTERMEDIATE

Q36. How do you use level 88 with SET ... TO TRUE?

A level 88 condition name makes the test readable, and **SET condition-name TO TRUE** moves the first VALUE of that 88 into the parent field, so you never code the literal twice. This is the usual way to handle flags such as end-of-file.

```
01 WS-EOF-SW  PIC X VALUE 'N'.
   88 END-OF-FILE VALUE 'Y'.
...
READ IN-FILE AT END SET END-OF-FILE TO TRUE END-READ.
PERFORM UNTIL END-OF-FILE
    PERFORM 2000-PROCESS
END-PERFORM.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q37. How is a COMP-3 (packed) number stored? What are the sign nibbles?

Each byte holds two decimal digits (one per half-byte, called a nibble), and the last nibble holds the sign: **C** positive, **D** negative, **F** unsigned. So PIC S9(3) COMP-3 holding +123 is X'123C' in 2 bytes, and -45 is X'045D'. Storage is (number of digits + 1) divided by 2, rounded up. Packed data cannot be read in a normal text editor; view it in HEX or through the copybook.

```
05 WS-AMT  PIC S9(5)V99 COMP-3.   (4 bytes)
* value +12345.67 -> X'1234567C'
```

INTERMEDIATE

Q38. What are COMP-1, COMP-2 and COMP-5?

COMP-1 is single-precision floating point (4 bytes) and **COMP-2** double-precision (8 bytes); neither has a PIC clause. **COMP-5** is native binary: the full range of the field is used (not cut by the number of 9s in the PIC), which suits data shared with C programs or APIs. Plain COMP (binary) may truncate values at the PIC limit, depending on the TRUNC compiler option.

INTERMEDIATE

Q39. What is the difference between CONTINUE, NEXT SENTENCE and EXIT?

CONTINUE is a no-operation: execution carries on with the next statement after the IF or EVALUATE. **NEXT SENTENCE** jumps to the statement after the next **period**, which surprises beginners when END-IF is used. **EXIT** is a no-op placed in an empty paragraph to mark the end of a PERFORM THRU range. Prefer CONTINUE and scope terminators.

```
IF WS-AMT > 0
    CONTINUE
ELSE
    PERFORM 9300-HANDLE-ZERO
END-IF.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q40. Why should you use scope terminators (END-IF, END-READ, END-PERFORM) instead of periods?

A period ends **all** open statements, so a stray period inside nested IFs silently changes the logic. Scope terminators close exactly one statement, make nested logic clear, and allow imperative statements inside conditional phrases like AT END or ON SIZE ERROR. Use a period only at the end of a paragraph.

```
IF WS-A > WS-B
   IF WS-A > WS-C
      MOVE WS-A TO WS-MAX
   END-IF
   ADD 1 TO WS-COUNT
END-IF.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q41. What are RECFM, LRECL and BLOCK CONTAINS 0 RECORDS?

RECFM is the record format: F fixed, FB fixed blocked, V variable, VB variable blocked. LRECL is the logical record length (for V and VB, add 4 bytes for the length header in the JCL LRECL; the COBOL record excludes it). **BLOCK CONTAINS 0 RECORDS** in the FD tells the system to take the block size from the JCL or choose the best one itself, so you do not hard-code BLKSIZE in the program.

```
FD IN-FILE
    RECORDING MODE IS F
    BLOCK CONTAINS 0 RECORDS
    RECORD CONTAINS 80 CHARACTERS.
```

INTERMEDIATE

Q42. Does WORKING-STORAGE keep its values between calls to a subprogram? How do you reset it?

Yes. A called subprogram keeps its WORKING-STORAGE values from the previous call unless it is cancelled (CANCEL statement, which frees it so the next call starts fresh) or declared **INITIAL** in PROGRAM-ID. Values in **LOCAL-STORAGE** are re-created on every call. This is why subprograms usually initialize counters and switches explicitly.

```
PROGRAM-ID. DATEVAL IS INITIAL.
* or in the caller:
CANCEL 'DATEVAL'.
```

INTERMEDIATE

Q43. Which data description clauses are commonly used besides PIC and VALUE?

JUSTIFIED RIGHT aligns alphanumeric data to the right. **BLANK WHEN ZERO** shows spaces for a zero value of an edited field. **SIGN IS LEADING/TRAILING SEPARATE** stores the sign as its own character. **SYNCHRONIZED** aligns binary items on word boundaries for speed (watch the extra slack bytes in record layouts). **REDEFINES**, **OCCURS** and **USAGE** were covered in earlier questions.

SCENARIO | * FREQUENTLY ASKED

Q44. Scenario: your batch job abends with S0C7. How do you find and fix the cause?

1) Read the abend message in the job output (SYSOUT/JES log) for the failing offset.
2) In the compile listing (compiled with the OFFSET or LIST option) find the statement at that offset.
3) Check the numeric fields in that statement. S0C7 usually means a field holds spaces, low-values or was never initialized.
4) Fix the cause: INITIALIZE fields, test with IS NUMERIC before arithmetic, or correct the bad input record. Tools like Abend-AID or Xpediter show field values at the abend.

```
IF WS-QTY IS NUMERIC
    COMPUTE WS-TOTAL = WS-QTY * WS-PRICE
ELSE
    PERFORM 9000-LOG-BAD-RECORD
END-IF.
```

SCENARIO

Q45. Scenario: a sorted input file has duplicate customer records. How do you write only one record per customer?

Duplicates are next to each other because the file is sorted by key. Keep the previous key in WORKING-STORAGE and compare it with the current key. Write the record only when the keys differ, then save the current key. Initialize the previous key to HIGH-VALUES so the first record is always written. The same compare-previous-key idea is used for control-break reports.

```
IF IN-CUST-ID NOT = WS-PREV-ID
    WRITE OUT-REC FROM IN-REC
    MOVE IN-CUST-ID TO WS-PREV-ID
END-IF.
```

SCENARIO

Q46. Scenario: DISPLAY of a COMP-3 field prints unreadable characters. Why, and what is the fix?

DISPLAY prints the raw bytes, and packed-decimal bytes are not printable characters. Move the COMP-3 field to a numeric-edited field and DISPLAY that. The same thing happens when you browse a data file in an editor: packed fields look like junk unless you view them in HEX or map them with a copybook.

```
01 WS-AMT-P   PIC S9(7)V99 COMP-3.
01 WS-AMT-D   PIC -9(7).99.
...
MOVE WS-AMT-P TO WS-AMT-D.
DISPLAY WS-AMT-D.
```

SCENARIO | * FREQUENTLY ASKED

Q47. Scenario: a program abends with S0C4. How do you find and fix the cause?

1) Read the message in the job log for the failing offset or statement, and use the compile listing (compiled with LIST or OFFSET) to find that statement.
2) Look for a **subscript or index beyond the table size**; recompile with SSRANGE to get a clear runtime message.
3) Check CALL parameters: different number or size of parameters in the caller and LINKAGE SECTION.
4) Check that all files opened successfully, since I/O on an unopened file or a bad record area can also cause S0C4.
5) Fix the logic, rebuild and rerun.

```
IF WS-IDX > 50
    DISPLAY 'TABLE FULL'
ELSE
    MOVE WS-CODE TO WS-ROW-CODE(WS-IDX)
END-IF.
```

SCENARIO | * FREQUENTLY ASKED

Q48. Scenario: update a master file from a transaction file (match-merge). Outline the logic.

Both files must be sorted on the same key. Read one record from each; at end of file move HIGH-VALUES to that file's key so the loop runs until both keys are HIGH-VALUES. Compare keys: master lower, write the master unchanged and read the next master; keys equal, apply the transaction (and read the next transaction); transaction lower, it is an add or an error because the master has no match. Write the updated master to a new file.

```
PERFORM UNTIL MAST-KEY = HIGH-VALUES AND TRAN-KEY = HIGH-VALUES
   EVALUATE TRUE
      WHEN MAST-KEY < TRAN-KEY
         PERFORM 3000-WRITE-MASTER
         PERFORM 1100-READ-MASTER
      WHEN MAST-KEY = TRAN-KEY
         PERFORM 4000-APPLY-TRAN
         PERFORM 1200-READ-TRAN
      WHEN OTHER
         PERFORM 5000-ADD-OR-REJECT
         PERFORM 1200-READ-TRAN
   END-EVALUATE
END-PERFORM.
```

SCENARIO

Q49. Scenario: write the logic to check whether a string is a palindrome.

Compare the string with its reverse. Remove the trailing spaces first (FUNCTION TRIM), otherwise the reversed text starts with spaces and never matches. If your compiler does not support TRIM, find the real length with a loop or by FUNCTION LENGTH on a trimmed copy, and compare the first half with the reversed second half.

```
IF FUNCTION TRIM(WS-STR) =
   FUNCTION REVERSE(FUNCTION TRIM(WS-STR))
    DISPLAY 'PALINDROME'
ELSE
    DISPLAY 'NOT A PALINDROME'
END-IF.
```

SCENARIO

Q50. Scenario: a CALL to a subprogram abends with S806. What do you check?

S806 means the system could not find the load module. Check that the subprogram is compiled and link-edited into a load library, that this library is in **STEPLIB or JOBLIB** of the failing step, and that the name in the CALL (case and spelling) matches the load module or alias name, not just PROGRAM-ID. Also check the DYNAM / NODYNAM setting: a static CALL (NODYNAM, literal) is resolved at link-edit time and shows up as an unresolved reference with a link-edit return code of 8, instead of an S806 at run time.

```
//STEPLIB DD DSN=USER1.LOADLIB,DISP=SHR
//         DD DSN=USER1.SUBS.LOADLIB,DISP=SHR
```

Section 4: DB2

SQL, embedded SQL, cursors, SQLCODEs, BIND, utilities and tuning basics

BASIC | * FREQUENTLY ASKED

Q1. What are DDL, DML, DCL and TCL?

DDL (CREATE, ALTER, DROP) defines objects. **DML** (SELECT, INSERT, UPDATE, DELETE) works with the data. **DCL** (GRANT, REVOKE) controls access. **TCL** (COMMIT, ROLLBACK) controls transactions.

BASIC | * FREQUENTLY ASKED

Q2. What is the difference between a primary key, a foreign key and a unique constraint?

A **primary key** uniquely identifies each row and cannot be NULL. In Db2 for z/OS it needs a unique index before the table can be used. A **foreign key** is a column that refers to the primary key of a parent table and enforces referential integrity. A **unique constraint** also blocks duplicates, but a table can have several of them.

```
CREATE TABLE EMP
 (EMP_ID    INTEGER      NOT NULL,
  EMP_NAME  VARCHAR(30)  NOT NULL,
  DEPT_ID   INTEGER,
  PRIMARY KEY (EMP_ID),
  FOREIGN KEY (DEPT_ID) REFERENCES DEPT);
```

BASIC

Q3. What is the difference between CHAR and VARCHAR? How do Db2 data types map to COBOL?

CHAR(n) is fixed length. VARCHAR(n) is variable length: COBOL gets a 2-byte length field plus the text.
SMALLINT = PIC S9(4) COMP
INTEGER = PIC S9(9) COMP
DECIMAL(p,s) = PIC S9(p-s)V9(s) COMP-3
DATE = PIC X(10), TIMESTAMP = PIC X(26)

BASIC | * FREQUENTLY ASKED

Q4. How is NULL handled in Db2 and in COBOL?

NULL means unknown or missing; it is not zero or blank. Test it with IS NULL (= NULL never works). COALESCE (or VALUE) replaces NULL with a default. In COBOL, every nullable column needs a **null indicator** variable (PIC S9(4) COMP): -1 means NULL, 0 means a real value came back. Without the indicator you get SQLCODE -305.

```
EXEC SQL
   SELECT BONUS INTO :WS-BONUS :WS-BONUS-IND
     FROM EMP
    WHERE EMP_ID = :WS-ID
END-EXEC.
IF WS-BONUS-IND < 0
   MOVE 0 TO WS-BONUS
END-IF.
```

BASIC | * FREQUENTLY ASKED

Q5. Explain the different types of joins.

INNER JOIN returns only matching rows. **LEFT OUTER JOIN** returns all rows from the left table plus the matches from the right (NULLs where nothing matches). **RIGHT OUTER JOIN** is the mirror image. **FULL OUTER JOIN** returns all rows from both tables.

```
SELECT E.EMP_NAME, D.DEPT_NAME
  FROM EMP E
  LEFT OUTER JOIN DEPT D
    ON E.DEPT_ID = D.DEPT_ID;
```

BASIC

Q6. What is the difference between WHERE and HAVING?

WHERE filters individual rows before grouping. HAVING filters groups after GROUP BY and can use aggregate functions such as COUNT, SUM, AVG, MIN and MAX.

```
SELECT DEPT_ID, COUNT(*)
  FROM EMP
 WHERE STATUS = 'A'
 GROUP BY DEPT_ID
HAVING COUNT(*) > 5;
```

BASIC

Q7. What is the difference between DELETE and DROP?

DELETE removes rows (all rows, or those matching a WHERE clause). It is a DML statement and can be rolled back. DROP removes the whole object (table, index) with its data and definition; it is DDL. Newer Db2 versions also offer TRUNCATE to empty a table quickly.

BASIC | * FREQUENTLY ASKED

Q8. What are SQLCA and SQLCODE?

The SQL Communication Area (SQLCA) is a structure Db2 fills after every SQL statement. Its main field is SQLCODE: **0** success, **+100** row not found or end of cursor, **negative** error, other positive values are warnings. Include it with EXEC SQL INCLUDE SQLCA and check SQLCODE after each statement.

```
EXEC SQL INCLUDE SQLCA END-EXEC.
...
EVALUATE SQLCODE
   WHEN 0     CONTINUE
   WHEN +100  DISPLAY 'NOT FOUND'
   WHEN OTHER DISPLAY 'DB2 ERROR ' SQLCODE
END-EVALUATE.
```

BASIC | * FREQUENTLY ASKED

Q9. What are embedded SQL and host variables?

Embedded SQL puts SQL statements between EXEC SQL and END-EXEC inside a COBOL program. Host variables are COBOL fields used in SQL, written with a leading colon. SELECT INTO fills them, and is only valid when the query returns at most one row.

```
EXEC SQL
   SELECT EMP_NAME, SALARY
     INTO :WS-NAME, :WS-SALARY
     FROM EMP
    WHERE EMP_ID = :WS-ID
END-EXEC.
```

BASIC | * FREQUENTLY ASKED

Q10. What is DCLGEN?

DCLGEN reads the table definition from the Db2 catalog and generates a DECLARE TABLE statement plus a matching COBOL host-variable structure. You bring it into the program with INCLUDE, which saves typing and avoids layout mismatches.

```
EXEC SQL INCLUDE EMPDCL END-EXEC.
```

BASIC | * FREQUENTLY ASKED

Q11. Name some commonly used SQL functions. What is the difference between COUNT(*) and COUNT(column)?

Aggregate: COUNT, SUM, AVG, MIN, MAX. Scalar: SUBSTR, LENGTH, UPPER, LOWER, CHAR, DIGITS, COALESCE (or VALUE), CAST, and date functions such as YEAR, MONTH, DAY and CURRENT DATE. **COUNT(*)** counts all rows; **COUNT(column)** counts only rows where the column is not NULL; **COUNT(DISTINCT column)** counts distinct non-NULL values.

```
SELECT COUNT(*), COUNT(BONUS), AVG(SALARY)
  FROM EMP;

SELECT EMP_ID, COALESCE(BONUS, 0) FROM EMP;
```

BASIC | * FREQUENTLY ASKED

Q12. Explain DISTINCT, ORDER BY, GROUP BY, LIKE, BETWEEN, IN and FETCH FIRST.

DISTINCT removes duplicate rows from the result. **ORDER BY** sorts (ASC/DESC). **GROUP BY** groups rows for aggregates. **LIKE** matches patterns: % any characters, _ one character. **BETWEEN** tests a range (both ends included). **IN** tests a list. **FETCH FIRST n ROWS ONLY** limits the result.

```
SELECT DISTINCT DEPT_ID
  FROM EMP
 WHERE EMP_NAME LIKE 'A%'
   AND SALARY BETWEEN 30000 AND 60000
   AND STATUS IN ('A','P')
 ORDER BY DEPT_ID
 FETCH FIRST 10 ROWS ONLY;
```

BASIC

Q13. What is the difference between UNION and UNION ALL?

Both combine the results of two queries, which must have the same number of columns with compatible data types. **UNION** removes duplicate rows (and so needs a sort); **UNION ALL** keeps every row and is faster. Use UNION ALL when you know there are no duplicates or you want them all.

```
SELECT EMP_ID FROM EMP_CHENNAI
UNION ALL
SELECT EMP_ID FROM EMP_MUMBAI;
```

BASIC | * FREQUENTLY ASKED

Q14. What is a view and why use one?

A view is a virtual table defined by a SELECT; it stores no data itself. Benefits: **security** (show only some columns or rows), **simplicity** (hides complex joins), and **logical data independence** (programs keep working if the base table changes). Views based on joins, GROUP BY or DISTINCT are usually read-only.

```
CREATE VIEW ACTIVE_EMP AS
  SELECT EMP_ID, EMP_NAME, DEPT_ID
    FROM EMP
   WHERE STATUS = 'A';
```

INTERMEDIATE | * FREQUENTLY ASKED

Q15. What is a cursor and what are the steps to use it?

A cursor lets a program process a multi-row result set one row at a time. Steps: **DECLARE** (defines the query), **OPEN** (runs it), **FETCH** (repeat until SQLCODE +100), **CLOSE**. For updates, declare it FOR UPDATE OF and use WHERE CURRENT OF.

```
EXEC SQL DECLARE C1 CURSOR FOR
   SELECT EMP_ID, EMP_NAME FROM EMP
    WHERE DEPT_ID = :WS-DEPT
END-EXEC.
EXEC SQL OPEN C1 END-EXEC.
PERFORM UNTIL SQLCODE NOT = 0
   EXEC SQL FETCH C1 INTO :WS-ID, :WS-NAME END-EXEC
   IF SQLCODE = 0 PERFORM 2000-PROCESS END-IF
END-PERFORM.
EXEC SQL CLOSE C1 END-EXEC.
```

INTERMEDIATE

Q16. What do COMMIT and ROLLBACK do?

COMMIT makes all changes since the last commit permanent and releases locks. ROLLBACK undoes uncommitted changes. A unit of work ends with either one. Batch programs commit every N rows. Note that COMMIT closes open cursors unless they were declared WITH HOLD.

```
EXEC SQL COMMIT END-EXEC.
EXEC SQL ROLLBACK END-EXEC.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q17. Explain DBRM, BIND, PLAN and PACKAGE. How is a COBOL-DB2 program prepared to run?

The precompiler removes the SQL from the source, replacing it with calls, and writes the SQL into a **DBRM**. The modified source is compiled and link-edited into a load module. **BIND PACKAGE** turns the DBRM into a package with chosen access paths. **BIND PLAN** lists the packages (PKLIST) so the program is allowed to run. Rebind after major changes such as new indexes or fresh statistics.

INTERMEDIATE

Q18. What is an index and when is it not used?

An index is a separate structure that lets Db2 find rows by key without scanning the whole table. A **unique** index enforces uniqueness (needed for a primary key); a **clustering** index controls the physical order of rows. Trade-off: faster SELECT, but slower INSERT/UPDATE/DELETE and more storage. An index may not help when a function is applied to the column in WHERE, or when most rows qualify.

INTERMEDIATE | * FREQUENTLY ASKED

Q19. What do SQLCODEs -811, -803, -305, -904 and +100 mean?

-811 SELECT INTO returned more than one row. Use a cursor or a more specific WHERE.
-803 INSERT/UPDATE would create a duplicate key in a unique index.
-305 NULL fetched into a host variable that has no null indicator.
-904 resource unavailable, for example a stopped or locked table space. Check the reason code.
+100 no row found, or end of cursor.

INTERMEDIATE

Q20. What is a subquery? What is the difference between IN and EXISTS?

A subquery is a SELECT nested inside another statement. IN compares a value with the list the subquery returns. EXISTS only checks whether the subquery returns any row, and is often faster for large lists. A **correlated** subquery refers to the outer query and is evaluated for each outer row.

```
SELECT EMP_NAME FROM EMP E
 WHERE EXISTS (SELECT 1 FROM DEPT D
                WHERE D.DEPT_ID = E.DEPT_ID
                  AND D.LOCATION = 'CHENNAI');
```

INTERMEDIATE

Q21. What are locks and isolation levels? What do -911 and -913 mean?

Db2 locks data so concurrent users do not corrupt it. Isolation levels control how long locks are held: **UR** (uncommitted read), **CS** (cursor stability, the common default), **RS** (read stability) and **RR** (repeatable read, the strictest). **-911** means deadlock or timeout and the unit of work was rolled back; **-913** means deadlock or timeout without an automatic rollback. Commit often and access tables in the same order to reduce deadlocks.

INTERMEDIATE | * FREQUENTLY ASKED

Q22. How do you update rows through a cursor (WHERE CURRENT OF)?

Declare the cursor with **FOR UPDATE OF column**, OPEN it and FETCH a row; then UPDATE or DELETE **WHERE CURRENT OF cursor-name** changes exactly the row the cursor is on. This avoids a second search for the row, and DB2 holds a lock on it. Commit regularly in long loops, and use WITH HOLD to keep the cursor open across commits.

```
EXEC SQL DECLARE C2 CURSOR FOR
   SELECT EMP_ID, SALARY FROM EMP
    WHERE DEPT_ID = :WS-DEPT
      FOR UPDATE OF SALARY
END-EXEC.
...
EXEC SQL FETCH C2 INTO :WS-ID, :WS-SAL END-EXEC.
EXEC SQL UPDATE EMP SET SALARY = SALARY * 1.05
          WHERE CURRENT OF C2
END-EXEC.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q23. What is EXPLAIN and the PLAN_TABLE? How do you check whether an index is used?

EXPLAIN writes the access path DB2 chose for a statement into the **PLAN_TABLE**. Read ACCESSTYPE: **I** means an index is used, **R** means a tablespace scan; MATCHCOLS shows how many index columns are matched, INDEXONLY = Y means the table is not read at all, and METHOD shows the join method (nested loop, merge scan). A scan on a large table in a frequently run query is the first thing to fix with an index or by rewriting the predicates.

```
EXPLAIN PLAN SET QUERYNO = 1 FOR
  SELECT EMP_NAME FROM EMP WHERE EMP_ID = 1001;

SELECT QUERYNO, TNAME, ACCESSTYPE, MATCHCOLS, INDEXONLY
  FROM PLAN_TABLE WHERE QUERYNO = 1;
```

INTERMEDIATE

Q24. Name the important DB2 utilities and what they do.

Run in batch with the utility program (DSNUTILB). **LOAD** loads data into a table. **UNLOAD** extracts data. **REORG** reorganizes table spaces or indexes to restore clustering. **RUNSTATS** collects the statistics the optimizer uses (run it after big changes and before rebinding). **COPY** takes an image copy backup. **RECOVER** restores from image copies and logs. **CHECK DATA** verifies constraints. **REBUILD INDEX** rebuilds an index.

INTERMEDIATE

Q25. What are a database, table space, index space and storage group?

A **database** is a logical group of related objects. A **table space** is the physical space that holds one or more tables, and it can be simple, segmented, partitioned or universal. An **index space** holds an index. A **storage group** (STOGROUP) names the disk volumes DB2 uses to allocate these objects. Information about all objects is kept in the **DB2 catalog** (tables such as SYSIBM.SYSTABLES), which you can query.

```
SELECT NAME, CREATOR FROM SYSIBM.SYSTABLES
 WHERE CREATOR = 'USER1';
```

INTERMEDIATE | * FREQUENTLY ASKED

Q26. Name more common SQLCODEs: -104, -204, -206, -407, -501, -502, -530, -532, -551 and -922.

-104 syntax error. **-204** table or object not found. **-206** column not in the table. **-407** NULL assigned to a NOT NULL column. **-501** FETCH or CLOSE on a cursor that is not open. **-502** OPEN on a cursor that is already open. **-530** insert or update of a foreign key with no matching parent row. **-532** delete restricted because child rows exist. **-551** not authorized for the object. **-922** authorization failure on connection.

SCENARIO | * FREQUENTLY ASKED

Q27. Scenario: after recompiling a program you get SQLCODE -818. What is wrong?

-818 means the timestamp in the load module does not match the DBRM/package. The program was recompiled but not re-bound, or the load module and DBRM came from different precompiles. Fix: run the steps in order again (precompile, compile, link-edit, BIND the package) and make sure the correct load library and DBRM library are used.

SCENARIO

Q28. Scenario: a batch program must update 1 million rows. How do you design the commits and restart?

Do not commit only once at the end: locks and log space would build up, and a failure would roll everything back. Declare the cursor WITH HOLD so COMMIT does not close it, commit every N rows (N from a parameter, say 500 to 1000), and save the last processed key in a checkpoint table or file so the job can restart from that point after an abend.

```
EXEC SQL DECLARE C1 CURSOR WITH HOLD FOR
   SELECT ACCT_ID FROM ACCOUNT ...
END-EXEC.
...
ADD 1 TO WS-COUNT.
IF WS-COUNT >= WS-COMMIT-FREQ
   EXEC SQL COMMIT END-EXEC
   MOVE 0 TO WS-COUNT
END-IF.
```

SCENARIO | * FREQUENTLY ASKED

Q29. Scenario: write queries for (a) the second highest salary and (b) duplicate email addresses.

(a) Use a subquery on MAX to find the largest salary below the maximum. (b) GROUP BY the column and keep groups with HAVING COUNT(*) greater than 1.

```
SELECT MAX(SALARY) FROM EMP
 WHERE SALARY < (SELECT MAX(SALARY) FROM EMP);

SELECT EMAIL, COUNT(*) FROM EMP
 GROUP BY EMAIL
HAVING COUNT(*) > 1;
```

SCENARIO | * FREQUENTLY ASKED

Q30. Scenario: your program gets -803 when inserting a row. How do you find the cause and handle it?

-803 means the insert (or update) would duplicate a key in a unique index. Check SQLERRMC for the index or table name, and look at the data being inserted. Business choices: skip the record and write it to a reject file, or treat it as an update instead. In the program test SQLCODE right after the INSERT and handle -803 separately from other errors so the batch does not abend because of one bad record. A parent-missing case (-530) is handled the same way.

```
EXEC SQL INSERT INTO EMP (EMP_ID, EMP_NAME)
     VALUES (:WS-ID, :WS-NAME)
END-EXEC.
EVALUATE SQLCODE
   WHEN 0     ADD 1 TO WS-INSERTED
   WHEN -803  PERFORM 9400-WRITE-REJECT
   WHEN OTHER PERFORM 9999-DB2-ERROR
END-EVALUATE.
```

Section 5: CICS

Pseudo-conversational design, BMS, EXEC CICS commands, COMMAREA, abends

BASIC | * FREQUENTLY ASKED

Q1. What is CICS?

CICS (Customer Information Control System) is IBM's online transaction processing (OLTP) monitor on z/OS. It runs application programs (COBOL, PL/I, Java and others) for thousands of terminal or API users at the same time, and manages terminals, files, Db2 access, security and recovery for them.

BASIC | * FREQUENTLY ASKED

Q2. What is the difference between conversational and pseudo-conversational programming?

A **conversational** program stays in memory and holds resources while it waits for the user to type, which wastes resources. A **pseudo-conversational** program ends after sending a screen (RETURN with TRANSID) and a new task starts when the user presses a key, so resources are used only while processing. Data between steps is carried in the COMMAREA. Pseudo-conversational is the standard design.

BASIC

Q3. What are a transaction, a task and a program? What are PCT and PPT?

A **transaction** is identified by a TRANSID of up to 4 characters that the user types. When it runs, CICS creates a **task**, one execution instance of it. The TRANSID is tied to a program through the transaction definition (traditionally the **PCT**), and programs and mapsets are defined in the program definition (traditionally the **PPT**). Today these are resource definitions in the CSD, maintained with CEDA.

BASIC | * FREQUENTLY ASKED

Q4. What is BMS? Explain map, mapset, physical map and symbolic map.

BMS (Basic Mapping Support) is used to design screens. A **mapset** (DFHMSD) is a group of maps, and a **map** (DFHMDI) is one screen, with fields defined by DFHMDF. Assembling the mapset creates a **physical map** (the load module that drives the screen) and a **symbolic map** (a COBOL copybook with the field names for your program).

```
MAPSET1 DFHMSD TYPE=&SYSPARM,MODE=INOUT,LANG=COBOL
MAP1    DFHMDI SIZE=(24,80)
EMPID   DFHMDF POS=(5,10),LENGTH=5,ATTRB=(UNPROT,NUM,IC)
        DFHMSD TYPE=FINAL
        END
```

BASIC | * FREQUENTLY ASKED

Q5. How do you send a screen and read user input?

SEND MAP displays the map on the terminal and RECEIVE MAP reads the user's input into the symbolic map. ERASE clears the screen first, DATAONLY sends only the data, and CURSOR positions the cursor.

```
EXEC CICS SEND MAP('MAP1') MAPSET('MAPSET1')
     FROM(MAP1O) ERASE
END-EXEC.

EXEC CICS RECEIVE MAP('MAP1') MAPSET('MAPSET1')
     INTO(MAP1I)
END-EXEC.
```

BASIC | * FREQUENTLY ASKED

Q6. What is the EIB? Name some important fields.

The EXEC Interface Block is a CICS area that is automatically available to every program. Common fields: **EIBCALEN** (length of the COMMAREA passed in; 0 on first entry), **EIBAID** (key pressed), **EIBTRNID** (transaction id), **EIBTRMID** (terminal id), EIBDATE / EIBTIME and EIBRESP (response of the last command).

```
IF EIBCALEN = 0
   PERFORM 1000-FIRST-TIME
ELSE
   PERFORM 2000-PROCESS-INPUT
END-IF.
```

BASIC

Q7. What are RESP and RESP2?

Adding RESP(variable) to a command makes CICS return the result code in your variable instead of abending. Compare it with DFHRESP values such as NORMAL, NOTFND, MAPFAIL, DUPREC and ENDFILE. RESP2 gives more detail. RESP is preferred over the older HANDLE CONDITION.

```
EXEC CICS READ FILE('EMPFILE') INTO(WS-REC)
     RIDFLD(WS-ID) RESP(WS-RESP)
END-EXEC.
IF WS-RESP = DFHRESP(NOTFND)
   MOVE 'NOT FOUND' TO MSGO
END-IF.
```

BASIC

Q8. How does a CICS program know which key the user pressed?

Attention keys (ENTER, PF1-PF24, PA keys, CLEAR) wake the program. EIBAID holds the key pressed; compare it with the names in the IBM copybook DFHAID. DFHBMSCA gives constants for attribute bytes.

```
COPY DFHAID.
COPY DFHBMSCA.
...
EVALUATE EIBAID
   WHEN DFHENTER  PERFORM 2100-PROCESS
   WHEN DFHPF3    PERFORM 9000-EXIT
   WHEN OTHER     MOVE 'INVALID KEY' TO MSGO
END-EVALUATE.
```

BASIC | * FREQUENTLY ASKED

Q9. Which commands read and change VSAM file records in CICS?

READ gets a record by the key in RIDFLD, WRITE adds a record, REWRITE updates a record and DELETE removes one. Use the UPDATE option on READ when you plan to REWRITE or DELETE, so CICS holds the record for you.

```
EXEC CICS READ FILE('EMPFILE') INTO(WS-REC)
     RIDFLD(WS-ID) UPDATE
END-EXEC.
MOVE WS-NEW-SAL TO WS-SALARY.
EXEC CICS REWRITE FILE('EMPFILE') FROM(WS-REC)
END-EXEC.
```

BASIC | * FREQUENTLY ASKED

Q10. What are CEDA, CEMT, CECI, CEDF and CESN?

CEDA defines and installs resources (programs, transactions, files). **CEMT** inquires about and sets resource status, for example NEWCOPY a program or open/close a file. **CECI** runs CICS commands interactively. **CEDF** is the execution diagnostic facility used to debug a program step by step. **CESN** signs a user on, and **CESF** signs off.

INTERMEDIATE | * FREQUENTLY ASKED

Q11. What is a COMMAREA and how is data passed with it?

COMMAREA is a block of memory used to pass data between pseudo-conversational steps or between programs (LINK/XCTL). The sender names it with COMMAREA(...) and the receiving program gets it as DFHCOMMAREA in the LINKAGE SECTION. It is limited to about 32 KB.

```
WORKING-STORAGE SECTION.
01 WS-COMMAREA.
   05 WS-CA-ID    PIC X(5).
LINKAGE SECTION.
01 DFHCOMMAREA    PIC X(5).
...
EXEC CICS RETURN TRANSID('EMP1')
     COMMAREA(WS-COMMAREA)
END-EXEC.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q12. What is the difference between LINK, XCTL and RETURN?

LINK calls another program like a subroutine; control comes back to the caller when the called program does RETURN. **XCTL** transfers control for good: the calling program is gone and nothing returns. **RETURN** ends the current program level. At the top level it ends the task, and with TRANSID(...) it names the next transaction to start on that terminal.

```
EXEC CICS LINK PROGRAM('CALCSAL')
     COMMAREA(WS-CA)
END-EXEC.
EXEC CICS XCTL PROGRAM('MENUPGM')
     COMMAREA(WS-CA)
END-EXEC.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q13. What are the L, F, A, I and O suffixes in a symbolic map? What is an attribute byte and MDT?

For a map field named EMPID, the symbolic map creates **EMPIDL** (length of data typed), **EMPIDF** (flag), **EMPIDA** (attribute byte), **EMPIDI** (input data) and **EMPIDO** (output data). Moving -1 to EMPIDL and using the CURSOR option places the cursor on that field. The attribute byte controls protection, brightness and the Modified Data Tag (MDT): only fields whose MDT is on are sent to the program on the next input.

INTERMEDIATE

Q14. How do you browse a VSAM file in CICS?

STARTBR positions at a key, READNEXT (or READPREV) reads the following records until the ENDFILE condition, and ENDBR finishes the browse.

```
EXEC CICS STARTBR FILE('EMPFILE')
     RIDFLD(WS-KEY)
END-EXEC.
PERFORM UNTIL WS-RESP = DFHRESP(ENDFILE)
   EXEC CICS READNEXT FILE('EMPFILE')
        INTO(WS-REC) RIDFLD(WS-KEY)
        RESP(WS-RESP)
   END-EXEC
   IF WS-RESP = DFHRESP(NORMAL)
      PERFORM 2000-SHOW-LINE
   END-IF
END-PERFORM.
EXEC CICS ENDBR FILE('EMPFILE') END-EXEC.
```

INTERMEDIATE | * FREQUENTLY ASKED

Q15. Name some common CICS abend codes and their causes.

ASRA program check (like S0C7 or S0C4 in batch): bad data or bad address.
AEI0 PGMIDERR: program or mapset not defined, not found or disabled.
AEI9 MAPFAIL: no data was received from the map.
AEIV LENGERR: length mismatch between your variable and the record or map.
AICA runaway task, often an infinite loop.
APCT program not found by CICS.
The AEI abends occur when the command has no RESP or HANDLE CONDITION.

INTERMEDIATE

Q16. What is the difference between Temporary Storage (TS) and Transient Data (TD) queues?

TS queue: a named queue you can read many times, by item number, and update or delete when done. It is used for scratch data and screen paging. **TD queue**: read in sequence and destroyed as it is read (read-once). Intrapartition queues can trigger a transaction; extrapartition queues write to external datasets such as logs and reports.

```
EXEC CICS WRITEQ TS QUEUE('EMPQ01')
     FROM(WS-REC) ITEM(WS-ITEM)
END-EXEC.
EXEC CICS READQ TS QUEUE('EMPQ01')
     INTO(WS-REC) ITEM(1)
END-EXEC.
```

INTERMEDIATE

Q17. What is a syncpoint and a logical unit of work (LUW)?

An LUW is a set of changes that must all succeed or all fail. SYNCPOINT commits the changes made so far to recoverable resources (files, Db2, queues). SYNCPOINT ROLLBACK undoes them. CICS also takes an implicit syncpoint at normal task end, and if the task abends, changes since the last syncpoint are backed out automatically.

```
EXEC CICS SYNCPOINT END-EXEC.
EXEC CICS SYNCPOINT ROLLBACK END-EXEC.
```

SCENARIO | * FREQUENTLY ASKED

Q18. Scenario: users get a MAPFAIL abend when they press ENTER without typing anything. How do you fix it?

MAPFAIL happens when RECEIVE MAP gets no data (no field was modified). Code RESP on RECEIVE MAP, and when it equals DFHRESP(MAPFAIL) show a message and send the map again instead of abending. Also test EIBAID before the RECEIVE so that PF3 (exit) works even when no data is typed.

```
EXEC CICS RECEIVE MAP('MAP1') MAPSET('MAPSET1')
     INTO(MAP1I) RESP(WS-RESP)
END-EXEC.
IF WS-RESP = DFHRESP(MAPFAIL)
   MOVE 'PLEASE ENTER EMPLOYEE ID' TO MSGO
   PERFORM 3000-SEND-MAP
END-IF.
```

SCENARIO | * FREQUENTLY ASKED

Q19. Scenario: design the flow of a pseudo-conversational employee-inquiry transaction.

1) First entry (EIBCALEN = 0): send the empty map and RETURN with TRANSID and COMMAREA.
2) Next entry (EIBCALEN > 0): RECEIVE MAP, validate the input, READ the file, move the data to the map, SEND MAP, then RETURN with TRANSID again.
3) On PF3: send a goodbye message and RETURN without TRANSID, which ends the conversation.

```
MAIN-PARA.
    IF EIBCALEN = 0
       PERFORM 1000-SEND-EMPTY-MAP
    ELSE
       IF EIBAID = DFHPF3
          EXEC CICS RETURN END-EXEC
       ELSE
          PERFORM 2000-RECEIVE-AND-READ
          PERFORM 3000-SEND-RESULT
       END-IF
    END-IF.
    EXEC CICS RETURN TRANSID('EMP1')
         COMMAREA(WS-CA)
    END-EXEC.
```

SCENARIO

Q20. Scenario: you recompiled and link-edited a CICS program, but users still see the old behaviour. What do you do?

CICS keeps a loaded copy of the program in memory. Tell CICS to load the new version with CEMT SET PROGRAM(name) NEWCOPY, or PHASEIN if the program is in use. Also check that the new load module is in a library of the DFHRPL concatenation, and refresh the mapset too if you changed a map.

```
CEMT SET PROGRAM(EMPPGM) NEWCOPY
CEMT SET PROGRAM(EMPPGM) PHASEIN
```

Quick Interview Tips for Freshers

Five habits that help you in the interview room

1. **Explain with a small example.**
   Interviewers like short answers backed by a one-line example or a code snippet from your lab programs.

2. **Know your own lab work end to end.**
   Be ready to walk through a COBOL-DB2-CICS program you wrote during training: files, SQL, screens and how you tested it.

3. **Revise abend codes and SQLCODEs.**
   S0C7, S0C4, S806, SB37, -811, -803, -305, -904, ASRA and AEI9 come up again and again.

4. **Be honest about what you have not done.**
   Say what you practised in labs. Add how you would find the answer, for example the compile listing, the job log, SDSF or the IBM manuals.

5. **Practise out loud.**
   Answer each question in 30 to 60 seconds. If an answer needs more than that, break it into steps.

Answers follow IBM z/OS JCL, DFSORT, IDCAMS, Enterprise COBOL, Db2 for z/OS and CICS Transaction Server. Exact option names, limits and defaults can differ between versions and installations, so confirm details against the manuals and standards used in your training and your workplace.