
SECTION 1 — Top 50 Technical Questions

Accenture | SAP ABAP Developer | Fresher to 2 Years

Q1. What is SAP ABAP? What does it stand for?

Frequency: ★★★★★ | Probability: 95%

Answer: ABAP stands for **Advanced Business Application Programming**. It is SAP's proprietary programming language used to develop applications on the SAP platform. It runs inside the SAP NetWeaver Application Server (AS ABAP) and is used to build reports, forms, interfaces, enhancements, and custom applications.

Beginner-friendly explanation: Think of ABAP like Java or Python, but specifically built for SAP systems. Just like Python is used to build general software, ABAP is used to build SAP-specific business applications.

Real-world example: If a company wants a custom report showing all sales orders above ₹1 lakh, an ABAP developer writes the program to fetch and display that data.

Keywords to mention: NetWeaver, AS ABAP, business applications, SAP platform, proprietary language

Common mistakes: Saying ABAP is a database language. It is an application programming language that communicates with databases.

Follow-up questions:

- What are the types of ABAP programs?
- Difference between ABAP and Java?
- What is the ABAP workbench?

-

What interviewer expects: A clear, confident definition with real-world context.

Q2. What is the difference between a Transparent Table, Pooled Table, and Cluster Table?

Frequency: ★★★★★ | Probability: 92%

Answer:

Feature	Transparent Table	Pooled Table	Cluster Table
DB Table	1:1 mapping	Many:1 (Table Pool)	Many:1 (Table Cluster)
Storage	Stored directly	Stored in TABPOOL	Stored in TABCLUST
Used for	Master/Transaction data	Small config tables	Long text, document data
Example	MARA, VBAK	ATAB	CDPOS, BSEG
Joins	Possible	Not possible	Not possible

Beginner-friendly explanation: A transparent table is like a regular Excel sheet — what you see in ABAP is exactly what is stored in the database. Pooled and cluster tables are compressed storage formats used for internal SAP tables.

Real-world example: MARA (Material Master) is a transparent table. You can directly query it using SE16. BSEG (Accounting document segment) is a cluster table — you cannot directly join it with other tables efficiently.

Keywords: SE11, dictionary, database table, TABPOOL, TABCLUST, BSEG, MARA

Common mistakes: Saying you can use joins on pooled/cluster tables. You cannot.

Follow-up: What is SE11? What is the difference between a table and a structure?

Q3. What are Internal Tables? What are their types?

Frequency: ★★★★★ | Probability: 95%

Answer: Internal tables are temporary data storage objects in ABAP used to hold multiple rows of data during program execution. They exist only in memory and are lost when the program ends.

-

Types:

Type	Key	Duplicates	Access
Standard Table	Default key	Allowed	Index / Key
Sorted Table	Sorted key	Not allowed	Binary search / Key
Hashed Table	Unique key	Not allowed	Key only

Syntax:

```
DATA: lt_mara TYPE STANDARD TABLE OF mara,
      lt_sorted TYPE SORTED TABLE OF mara WITH UNIQUE KEY matnr,
      lt_hashed TYPE HASHED TABLE OF mara WITH UNIQUE KEY matnr.
```

Beginner-friendly explanation: Think of internal tables like a temporary Excel sheet inside your program. Standard table = normal sheet. Sorted table = sheet sorted by a column. Hashed table = sheet where each row has a unique ID and can be found instantly.

Real-world example: You fetch 10,000 material records from MARA into an internal table, loop through them, and display only active materials.

Keywords: STANDARD TABLE, SORTED TABLE, HASHED TABLE, header line, work area, TYPE TABLE OF

Common mistakes:

- Using standard table with linear search on large data (performance issue)
- Confusing internal table with database table

Follow-up: What is a work area? What is the difference between header line and work area?

Q4. What is the difference between a Work Area and a Header Line?

Frequency: ★★★★★ | Probability: 88%

Answer:

Feature	Work Area	Header Line
Declaration	Separate variable	Implicit with internal table

SAP recommendation	Recommended	Not recommended (old style)
Syntax	`DATA: wa_mara TYPE mara`	`DATA: lt_mara TYPE mara WITH HEADER LINE`
Clarity	Clear and safe	Can cause confusion

Beginner-friendly explanation: A work area is like a single row holder — you move one row into it, process it, then move to the next. A header line is an old technique where the internal table itself had a built-in single row area on top.

SAP best practice: Always use explicit work areas. Header lines are obsolete.

Keywords: INTO wa, LOOP AT ... INTO wa, WITH HEADER LINE (obsolete)

Common mistakes: Using header line in new ABAP code. Interviewers will notice this negatively.

Q5. What is a SELECT statement in ABAP? Explain its variants.

Frequency: ★★★★★ | Probability: 93%

Answer: SELECT is used to fetch data from database tables into internal tables or work areas.

Variants:

```
" Single row
SELECT SINGLE * FROM mara INTO wa_mara WHERE matnr = '100'.

" Multiple rows into internal table
SELECT * FROM mara INTO TABLE lt_mara WHERE mtart = 'FERT'.

" Specific fields
SELECT matnr mtart mbrsh FROM mara INTO TABLE lt_mara.

" With condition
SELECT matnr meins FROM mara INTO TABLE lt_mara
  WHERE matnr IN s_matnr AND mtart = 'ROH'.
```

New Open SQL (S/4HANA style):

```
SELECT matnr, mtart, mbrsh
```

-

```
FROM mara
INTO TABLE @lt_mara
WHERE mtart = 'FERT'.
```

Keywords: SELECT SINGLE, SELECT *, INTO TABLE, WHERE clause, New Open SQL, @ symbol

Common mistakes:

- Using SELECT * instead of selecting only required fields (performance issue)
- Using SELECT SINGLE when multiple rows expected
- Not using INTO TABLE — causes dump

Follow-up: What is SELECT FOR ALL ENTRIES? What is the difference between JOIN and FOR ALL ENTRIES?

Q6. What is SELECT FOR ALL ENTRIES?

Frequency: ★★★★★ | **Probability: 90%**

Answer: FOR ALL ENTRIES is used to fetch data from a database table based on values in an internal table. It acts like a dynamic WHERE condition.

```
SELECT vbeln posnr matnr
  FROM vbap
  INTO TABLE lt_vbap
  FOR ALL ENTRIES IN lt_vbak
  WHERE vbeln = lt_vbak-vbeln.
```

Important rules:

1. The driver internal table (lt_vbak) must NOT be empty before using FOR ALL ENTRIES
2. Always check: `IF lt_vbak IS NOT INITIAL`
3. It removes duplicate rows automatically
4. Cannot use ORDER BY with FOR ALL ENTRIES

Beginner-friendly explanation: Imagine you have 100 sales order headers. FOR ALL ENTRIES lets you fetch all line items for those 100 orders in one shot instead of looping and querying one by one.

Real-world example: Fetch VBAK (header) first, then use FOR ALL ENTRIES to get VBAP (items) for those headers.

Keywords: FOR ALL ENTRIES IN, driver table, empty check, IS NOT INITIAL

-

Common mistakes:

- Not checking if the driver table is empty — this fetches ALL records from the database (critical bug)
- Using it when a JOIN would be better

Q7. What is the difference between JOIN and FOR ALL ENTRIES?

Frequency: ★★★★★ | Probability: 85%

Answer:

Feature	JOIN	FOR ALL ENTRIES
Executed at	Database level	Application server level
Performance	Better for small data	Better for large data sets
Empty check	Not needed	Mandatory
Duplicates	Can return duplicates	Removes duplicates
Usage	When tables are related	When data is already in memory

Keywords: INNER JOIN, LEFT OUTER JOIN, FOR ALL ENTRIES, application server, database level

Common mistakes: Using FOR ALL ENTRIES when a simple JOIN would be more efficient.

Q8. What are Field Symbols?

Frequency: ★★★★★ | Probability: 87%

Answer: Field symbols are like pointers in ABAP. They do not hold data themselves — they point to an existing data object in memory. Changes through a field symbol directly modify the original data.

```
FIELD-SYMBOLS: <fs_mara> TYPE mara.

LOOP AT lt_mara ASSIGNING <fs_mara>.
  <fs_mara>-mtart = 'FERT'.  " Directly modifies the internal table row
```

-

ENDLOOP.

vs LOOP INTO work area:

```
LOOP AT lt_mara INTO wa_mara.
  wa_mara-mtart = 'FERT'.
  MODIFY lt_mara FROM wa_mara.  " Extra MODIFY statement needed
ENDLOOP.
```

Beginner-friendly explanation: Field symbol is like a sticky note pointing to a drawer. When you write on the sticky note, the content in the drawer changes directly. Work area is like taking out the content, editing a copy, and putting it back.

Performance: Field symbols are faster because they avoid data copying.

Keywords: FIELD-SYMBOLS, ASSIGNING, pointer, memory reference, performance

Common mistakes:

- Forgetting to check if field symbol is assigned before using it
- Confusing field symbols with reference variables

Q9. What are Function Modules? How are they different from Subroutines?

Frequency: ★★★★★ | Probability: 88%

Answer:

Feature	Function Module	Subroutine (FORM)
Reusability	Global — across programs	Local — within program
Exception handling	Yes	No
RFC capable	Yes	No
Group	Function Group	Same program
Transaction	SE37	Same program

Function Module structure:

- IMPORTING parameters (input)

-

- EXPORTING parameters (output)
- CHANGING parameters (input + output)
- TABLES parameters (internal tables)
- EXCEPTIONS (error handling)

```
CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT'
  EXPORTING
    input  = lv_matnr
  IMPORTING
    output = lv_matnr.
```

Keywords: SE37, Function Group, CALL FUNCTION, IMPORTING, EXPORTING, EXCEPTIONS

Common mistakes: Saying subroutines and function modules are the same. They are not.

Q10. What is BAPI? Give an example.

Frequency: ★★★★★ | **Probability: 85%**

Answer: BAPI stands for **Business Application Programming Interface**. It is a standardized function module that provides external access to SAP business processes and data. BAPIs are RFC-enabled and follow SAP's business object model.

Examples:

BAPI	Purpose
BAPI_SALESORDER_CREATEFROMDAT2	Create Sales Order
BAPI_GOODSMVT_CREATE	Post Goods Movement
BAPI_PO_CREATE1	Create Purchase Order
BAPI_TRANSACTION_COMMIT	Commit changes

Important: After calling a BAPI, always call `BAPI_TRANSACTION_COMMIT` to save changes.

```
CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
  EXPORTING
    wait = 'X'.
```

-

Keywords: RFC-enabled, Business Object, BAPI_TRANSACTION_COMMIT, SE37, standard SAP

Common mistakes: Forgetting to call COMMIT after BAPI. Changes will not be saved.

Q11. What is RFC? What are its types?

Frequency: ★★★★★ | **Probability: 82%**

Answer: RFC stands for **Remote Function Call**. It allows communication between two SAP systems or between SAP and external systems.

Types:

Type	Description
Synchronous RFC (sRFC)	Waits for response. Most common.
Asynchronous RFC (aRFC)	Does not wait. Parallel processing.
Transactional RFC (tRFC)	Guaranteed exactly-once execution
Queued RFC (qRFC)	tRFC with defined processing sequence
Background RFC (bgRFC)	Modern replacement for tRFC/qRFC

Keywords: RFC destination, SM59, CALL FUNCTION ... DESTINATION, remote-enabled

Common mistakes: Confusing RFC with BAPI. BAPI is a type of RFC-enabled function module. Not all RFCs are BAPIs.

Q12. What is an Enhancement? Types of Enhancements in ABAP?

Frequency: ★★★★★ | **Probability: 85%**

Answer: Enhancements allow you to add custom code to standard SAP programs without modifying the original SAP objects (modification-free).

Types:

Type	Description	Example

•

User Exits	FORM routines in SAP programs	EXIT_SAPMV45A_001
Customer Exits	Function modules SAP provides for enhancement	EXIT_*
BADIs	OO-based enhancement spots	ME_PROCESS_PO_CUST
Enhancement Spots	Modern ABAP enhancement framework	ENHANCEMENT SECTION
Implicit Enhancements	Add code at start/end of any include	SE80

Keywords: SE18, SE19, BADI, User Exit, Enhancement Framework, modification-free

Common mistakes: Modifying standard SAP objects directly instead of using enhancements. This causes issues during SAP upgrades.

Q13. What is a BADI? How do you implement one?

Frequency: ★★★★★ | **Probability: 83%**

Answer: BADI stands for **Business Add-In**. It is an OOP-based SAP enhancement technique that allows you to implement custom code at predefined enhancement spots in standard SAP programs.

Steps to implement a BADI:

1. Go to **SE18** — find the BADI definition
2. Go to **SE19** — create BADI implementation
3. Create a class implementing the BADI interface
4. Write code in the method
5. Activate the implementation

New BADIs (S/4HANA): Use `GET BADI` and `CALL BADI` syntax.

```
DATA: lo_badi TYPE REF TO badi_name.
GET BADI lo_badi.
CALL BADI lo_badi->method_name
  EXPORTING ...
  IMPORTING ...
```

Keywords: SE18, SE19, interface, implementation, OOP, enhancement spot

Common mistakes: Confusing BADI with User Exit. BADIs are OOP-based and more flexible.

-

Q14. What is ALV? How do you create a basic ALV report?

Frequency: ★★★★★ | **Probability: 90%**

Answer: ALV stands for **ABAP List Viewer**. It is a standard SAP tool to display data in a formatted, interactive grid or list with built-in features like sorting, filtering, totals, and export to Excel.

Types of ALV:

- Classical ALV (Function Module based) — `REUSE_ALV_GRID_DISPLAY`
- OOP ALV — `CL_GUI_ALV_GRID`
- SAP List Viewer (SALV) — `CL_SALV_TABLE` (recommended for S/4HANA)

Basic SALV ALV:

```
DATA: lo_alv  TYPE REF TO cl_salv_table,
      lt_mara TYPE STANDARD TABLE OF mara.

SELECT * FROM mara INTO TABLE lt_mara UP TO 100 ROWS.

cl_salv_table=>factory(
  IMPORTING r_salv_table = lo_alv
  CHANGING  t_table      = lt_mara ).

lo_alv->display( ).
```

Keywords: CL_SALV_TABLE, REUSE_ALV_GRID_DISPLAY, field catalog, layout, FCAT

Common mistakes: Building field catalog manually when it's not needed for simple reports.

Q15. What is the difference between Smart Forms and Adobe Forms?

Frequency: ★★★★★ | **Probability: 80%**

Answer:

Feature	Smart Forms	Adobe Forms

-

Tool	SAP Smart Forms	Adobe LiveCycle Designer
Transaction	SMARTFORMS	SFP
Output	SAPscript-based	PDF-based
Interactivity	No	Yes (fillable PDFs)
S/4HANA preference	Less preferred	Preferred
Language	ABAP	ABAP + XML/PDF

Keywords: SFP, SMARTFORMS, PDF, interactive forms, ADS (Adobe Document Services)

Common mistakes: Saying Smart Forms are used in S/4HANA for new development. Adobe Forms is the preferred choice.

Q16. What are CDS Views? Why are they important in S/4HANA?

Frequency: ★★★★★ | **Probability: 88%**

Answer: CDS stands for **Core Data Services**. CDS Views are database-level views defined in ABAP that push data processing logic to the HANA database instead of the application server (Code Pushdown principle).

```
@AbapCatalog.sqlViewName: 'ZMARA_VIEW'
@EndUserText.label: 'Material CDS View'
define view ZMARA_CDS_VIEW as select from mara {
  key matnr,
      mtart,
      mbrsh,
      meins
}
```

Advantages:

- Code pushdown to HANA
- Reusable across multiple applications
- Supports OData (used in Fiori apps)
- Better performance than ABAP SELECT

Keywords: @AbapCatalog, @EndUserText, define view, SE11, ADT (Eclipse), associations, annotations

-

Common mistakes: Thinking CDS is just a regular database view. CDS views support annotations, associations, and business semantics.

Q17. What is Code Pushdown? Why is it important?

Frequency: ★★★★★ | **Probability: 82%**

Answer: Code Pushdown means moving data-intensive processing logic from the ABAP application server to the SAP HANA database layer where it executes much faster using in-memory computing.

Traditional approach (bad):

- Fetch millions of records to application server
- Process/filter in ABAP

Code Pushdown approach (good):

- Filter, aggregate, and process at database level using CDS or AMDP
- Only fetch required result set to application server

Tools for code pushdown:

- CDS Views
- AMDP (ABAP Managed Database Procedures)
- New Open SQL aggregations

Keywords: In-memory, HANA database, application server, CDS, AMDP, performance

---

Q18. What is AMDP?

Frequency: ★★★★ | **Probability: 72%**

Answer: AMDP stands for **ABAP Managed Database Procedures**. It allows writing SAP HANA SQLScript directly inside ABAP classes, enabling complex database-side processing that cannot be done through Open SQL alone.

```
CLASS zcl_amdp_example DEFINITION.
  PUBLIC SECTION.
    INTERFACES if_amdp_marker_hdb.
    CLASS-METHODS get_data
      IMPORTING VALUE(iv_mtart) TYPE mtart
      EXPORTING VALUE(et_mara)  TYPE mara_tt.
ENDCLASS.
```

-

```
CLASS zcl_amdp_example IMPLEMENTATION.
  METHOD get_data BY DATABASE PROCEDURE
        FOR HDB LANGUAGE SQLSCRIPT
        USING mara.
    et_mara = SELECT * FROM mara WHERE mtart = :iv_mtart;
  ENDMETHOD.
ENDCLASS.
```

Keywords: IF_AMDP_MARKER_HDB, BY DATABASE PROCEDURE, SQLSCRIPT, HANA, code pushdown

---

Q19. What is Object-Oriented ABAP? Explain Class and Object.

Frequency: ★★★★★ | Probability: 85%

Answer: OO ABAP is the object-oriented programming approach in ABAP, supporting concepts like encapsulation, inheritance, and polymorphism.

Class: Blueprint or template. **Object:** Instance of a class.

```
CLASS lcl_car DEFINITION.
  PUBLIC SECTION.
    DATA: mv_color TYPE string.
    METHODS: start_engine.
ENDCLASS.

CLASS lcl_car IMPLEMENTATION.
  METHOD start_engine.
    WRITE: 'Engine started!'.
  ENDMETHOD.
ENDCLASS.

" Create object
DATA: lo_car TYPE REF TO lcl_car.
CREATE OBJECT lo_car.
lo_car->start_engine( ).
```

OOP concepts in ABAP:

- **Encapsulation** — PUBLIC / PROTECTED / PRIVATE sections
- **Inheritance** — INHERITING FROM
- **Polymorphism** — Method overriding

-
- **Abstraction** — Abstract classes and interfaces

Keywords: CLASS, OBJECT, CREATE OBJECT, REF TO, ->method, PUBLIC SECTION

Q20. What is the difference between Abstract Class and Interface?

Frequency: ★★★★ | Probability: 78%

Answer:

Feature	Abstract Class	Interface
Instantiation	Cannot be instantiated	Cannot be instantiated
Method implementation	Can have implemented methods	All methods are abstract
Multiple inheritance	Not supported	Supported
Keyword	ABSTRACT	INTERFACE
Usage	Partial implementation	Full contract

Q21. What is Exception Handling in ABAP?

Frequency: ★★★★ | Probability: 80%

Answer: Exception handling in ABAP catches and handles runtime errors gracefully using TRY-CATCH blocks (class-based exceptions).

```
TRY.
  DATA(lv_result) = 10 / 0.
CATCH cx_sy_zerodivide INTO DATA(lo_ex).
  WRITE: lo_ex->get_text( ).
CATCH cx_root INTO DATA(lo_root).
  WRITE: 'Unknown error'.
ENDTRY.
```

Keywords: TRY, CATCH, ENDTRY, CX_ROOT, CX_SY_ZERODIVIDE, GET_TEXT, RAISE EXCEPTION

Common mistakes: Using SY-SUBRC instead of proper exception handling for OO code.

-

Q22. What is Debugging in ABAP? Important Debugging tools?

Frequency: ★★★★★ | **Probability: 88%**

Answer: Debugging is the process of finding and fixing errors in ABAP programs by stepping through code execution.

How to start debugger:

- Add `/h` in the command field and press Enter
- Set external breakpoint: `BREAK username` in code
- Set dynamic breakpoint in SE80/SE38

Key Debugging features:

Feature	Purpose
Breakpoints	Pause execution at a point
Watchpoints	Pause when a variable changes
Step Into (F5)	Go inside called function/method
Step Over (F6)	Skip over called function
Step Out (F7)	Exit current function
Display variable	See current value

Keywords: /h, BREAK-POINT, watchpoint, SE38, ABAP debugger, external breakpoint

---

Q23. What is ST05? What is SQL Trace?

Frequency: ★★★★ | **Probability: 80%**

Answer: ST05 is the SAP Performance Trace transaction. It captures and analyzes all database SQL statements executed by an ABAP program.

Steps:

1. Go to ST05
2. Click "Activate Trace"
3. Run your program

-

4. Come back to ST05 → "Deactivate Trace"
5. Click "Display Trace"
6. Analyze slow queries

What to look for:

- Full table scans (missing WHERE clause)
- Missing indexes
- Expensive JOINs
- Frequent small queries (use FOR ALL ENTRIES instead)

Related tools:

Tool	Purpose
ST05	SQL Trace
SAT / SE30	Runtime analysis
SM50/SM66	Work process monitor

Keywords: ST05, SQL trace, performance, full table scan, index, SAT

Q24. What is Transport Management? What is a Transport Request?

Frequency: ★★★★ | Probability: 82%

Answer: Transport Management is the process of moving ABAP objects (programs, tables, function modules) from one SAP system to another (DEV → QAS → PRD).

Systems:

- **DEV** — Development system (where you code)
- **QAS** — Quality/Testing system
- **PRD** — Production system

Transport Request types:

Type	Purpose
Workbench Request	Repository objects (programs, classes)
Customizing Request	Configuration objects

-

Transaction codes:

- **SE10** — Manage transport requests
- **STMS** — Transport Management System
- **SCC1** — Copy client

Keywords: SE10, STMS, DEV/QAS/PRD landscape, workbench request, release transport

Common mistakes: Releasing a transport without testing in QAS first.

Q25. What is a Lock Object? Why is it used?

Frequency: ★★★★ | Probability: 75%

Answer: Lock Objects are used to prevent simultaneous access or modification of the same data by multiple users, ensuring data consistency.

Created in: SE11 → Lock Objects

Generates two function modules:

- `ENQUEUE_<lockobject>` — to lock
- `DEQUEUE_<lockobject>` — to unlock

```
CALL FUNCTION 'ENQUEUE_EZMATNR'
  EXPORTING
    matnr = lv_matnr
  EXCEPTIONS
    foreign_lock = 1
    system_failure = 2.

IF sy-subrc = 1.
  MESSAGE 'Record is locked by another user' TYPE 'E'.
ENDIF.
```

Keywords: ENQUEUE, DEQUEUE, SE11, lock mode (Shared/Exclusive), foreign_lock

---

Q26. What are IDocs? Explain their structure.

Frequency: ★★★★ | Probability: 78%

Answer: IDoc stands for **Intermediate Document**. It is SAP's standard format for exchanging data between SAP systems or between SAP and external systems (EDI).

-

Structure:

Component	Description
Control Record	Header — sender/receiver info
Data Records	Actual business data (segments)
Status Records	Processing status

Key transactions:

T-Code	Purpose
WE60	IDoc documentation
WE02/WE05	Display IDocs
WE19	Test IDoc processing
BD87	Reprocess failed IDocs

Keywords: IDoc type, message type, segment, partner profile, WE20, BD10

Q27. What is the difference between MODIFY and UPDATE in Open SQL?

Frequency: ★★★★ | Probability: 78%

Answer:

Feature	MODIFY	UPDATE
If record exists	Updates it	Updates it
If record doesn't exist	Inserts it	Does nothing
Behavior	INSERT + UPDATE combined	Only UPDATE
Use case	Upsert scenario	Update known records

```
" MODIFY — insert or update
MODIFY mara FROM wa_mara.
```

-

" UPDATE — only update existing
UPDATE mara FROM wa_mara.

Keywords: MODIFY, UPDATE, INSERT, DELETE, Open SQL DML

Q28. What is SY-SUBRC? When is it used?

Frequency: ★★★★★ | **Probability: 90%**

Answer: SY-SUBRC is a system field in ABAP that stores the return code of the last executed statement.

- **0** — Successful
- **4** — Not successful / record not found
- **8 or above** — Error

Usage:

```
SELECT SINGLE * FROM mara INTO wa_mara WHERE matnr = lv_matnr.
IF sy-subrc = 0.
  WRITE: 'Record found'.
ELSE.
  WRITE: 'Record not found'.
ENDIF.
```

Keywords: System field, return code, sy-subrc = 0, error handling

Common mistakes: Not checking SY-SUBRC after database operations.

Q29. What is the difference between TYPE and LIKE?

Frequency: ★★★★ | **Probability: 80%**

Answer:

Keyword	Usage	Example
TYPE	Reference to a data type	`DATA: lv_matnr TYPE matnr`

-

LIKE	Reference to an existing data object	`DATA: lv_mat2 LIKE lv_matnr`

Best practice: Use TYPE for cleaner, more readable code. LIKE is old style.

Q30. What are Selection Screens? What are SELECT-OPTIONS?

Frequency: ★★★★ | **Probability: 82%**

Answer: Selection screens are input screens displayed to users before a report runs, allowing them to enter filter values.

```
SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME TITLE text-001.
  SELECT-OPTIONS: s_matnr FOR mara-matnr.
  PARAMETERS: p_mtart TYPE mtart.
SELECTION-SCREEN END OF BLOCK b1.
```

SELECT-OPTIONS generates a range table with fields:

- SIGN (I = Include, E = Exclude)
- OPTION (EQ, BT, CP, etc.)
- LOW (lower value)
- HIGH (upper value)

Keywords: SELECTION-SCREEN, PARAMETERS, SELECT-OPTIONS, range table, LOW, HIGH

Q31. What is the LOOP statement? What are its variants?

Frequency: ★★★★★ | **Probability: 88%**

Answer:

```
" Basic loop
LOOP AT lt_mara INTO wa_mara.
  WRITE: wa_mara-matnr.
ENDLOOP.

" Loop with condition
```

-

```
LOOP AT lt_mara INTO wa_mara WHERE mtart = 'FERT'.
  WRITE: wa_mara-matnr.
ENDLOOP.

" Loop with field symbol (performance)
LOOP AT lt_mara ASSIGNING <fs_mara>.
  <fs_mara>-mtart = 'ROH'.
ENDLOOP.

" Loop with index
LOOP AT lt_mara INTO wa_mara.
  WRITE: sy-tabix.  " Current row index
ENDLOOP.
```

Keywords: LOOP AT, INTO, ASSIGNING, WHERE, sy-tabix, ENDLOOP

Q32. What is READ TABLE? How is Binary Search used?

Frequency: ★★★★★ | **Probability: 85%**

Answer: READ TABLE is used to read a single row from an internal table.

```
" Read by index
READ TABLE lt_mara INTO wa_mara INDEX 1.

" Read by key (linear search)
READ TABLE lt_mara INTO wa_mara WITH KEY matnr = '100'.

" Read with binary search (table must be sorted first)
SORT lt_mara BY matnr.
READ TABLE lt_mara INTO wa_mara WITH KEY matnr = '100' BINARY SEARCH.

" Check if found
IF sy-subrc = 0.
  WRITE: 'Found'.
ENDIF.
```

Binary Search performance: O(log n) vs O(n) for linear search. Always sort before binary search.

Keywords: READ TABLE, BINARY SEARCH, WITH KEY, sy-subrc, sy-tabix

---

•

Q33. What is SORT in ABAP?

Frequency: ★★★★ | Probability: 80%

Answer:

```
" Sort by one field ascending (default)
SORT lt_mara BY matnr.

" Sort descending
SORT lt_mara BY matnr DESCENDING.

" Sort by multiple fields
SORT lt_mara BY mtart ASCENDING matnr DESCENDING.
```

Important: Sort is required before BINARY SEARCH and DELETE ADJACENT DUPLICATES.

Q34. What is DELETE ADJACENT DUPLICATES?

Frequency: ★★★★ | Probability: 80%

Answer: Used to remove duplicate consecutive rows from a sorted internal table.

```
SORT lt_mara BY matnr.
DELETE ADJACENT DUPLICATES FROM lt_mara COMPARING matnr.
```

Important: Always SORT before DELETE ADJACENT DUPLICATES. It only removes consecutive duplicates.

Q35. What is the difference between COLLECT and APPEND?

Frequency: ★★★ | Probability: 65%

Answer:

Feature	APPEND	COLLECT
Behavior	Always adds a new row	Adds new row or sums numeric fields if key exists

Use case	Simple addition	Aggregation
Performance	Fast	Slower (checks for existing key)

```
" APPEND — always adds
APPEND wa_data TO lt_data.

" COLLECT — aggregates numeric fields
wa_total-matnr = '100'.
wa_total-qty   = 50.
COLLECT wa_total INTO lt_total.
```

Q36. What are the important System Fields in ABAP?

Frequency: ★★★★★ | Probability: 90%

Answer:

Field	Description
SY-SUBRC	Return code of last operation
SY-TABIX	Current row index in LOOP
SY-DBCNT	Number of rows affected by DB operation
SY-UNAME	Current logged-in username
SY-DATUM	Current date
SY-UZEIT	Current time
SY-MANDT	Current client
SY-LANGU	Current language
SY-REPID	Current program name
SY-MSGTY	Message type

Keywords: SY-SUBRC, SY-TABIX, SY-DBCNT, system fields

Q37. What is the difference between CLASS-DATA and DATA in OO ABAP?

-

Frequency: ★★★★ | Probability: 72%

Answer:

Feature	DATA (Instance)	CLASS-DATA (Static)
Scope	Per object instance	Shared across all instances
Memory	Separate for each object	Single copy for all objects
Access	Via object reference	Via class name directly

```
CLASS lcl_example DEFINITION.
  PUBLIC SECTION.
    DATA: mv_instance TYPE string.          " Instance attribute
    CLASS-DATA: mv_static TYPE string.      " Static attribute
ENDCLASS.

lcl_example=>mv_static = 'Shared'.  " Access static directly
```

Q38. What is an Interface in ABAP OOP?

Frequency: ★★★★ | Probability: 75%

Answer: An interface defines a contract — a set of methods that any implementing class must provide. It enables multiple inheritance-like behavior.

```
INTERFACE lif_vehicle.
  METHODS: start_engine,
           stop_engine.
ENDINTERFACE.

CLASS lcl_car DEFINITION.
  PUBLIC SECTION.
    INTERFACES: lif_vehicle.
ENDCLASS.

CLASS lcl_car IMPLEMENTATION.
  METHOD lif_vehicle~start_engine.
    WRITE: 'Car engine started'.
  ENDMETHOD.
  METHOD lif_vehicle~stop_engine.
    WRITE: 'Car engine stopped'.
  ENDMETHOD.
ENDCLASS.
```

-

Keywords: INTERFACE, INTERFACES, tilde (~), method contract, polymorphism

---

Q39. What is SAP HANA? How is it different from a traditional database?

Frequency: ★★★★★ | Probability: 85%

Answer:

Feature	Traditional DB	SAP HANA
Storage	Disk-based	In-memory
Processing	Row-based	Column-based + Row-based
Speed	Slower	Much faster
Analytics	Separate system needed	Built-in
Compression	Low	High

HANA key concepts:

- In-memory computing
- Column store (better for analytics)
- Row store (better for OLTP)
- Code pushdown
- CDS Views, AMDP

Keywords: In-memory, column store, OLTP, OLAP, S/4HANA, code pushdown

---

Q40. What is S/4HANA? How is it different from ECC?

Frequency: ★★★★★ | Probability: 85%

Answer:

Feature	SAP ECC	SAP S/4HANA

Database	Any DB (Oracle, DB2)	Only SAP HANA
Architecture	Old ABAP stack	Simplified, new architecture
Simplification	Complex data model	Simplified tables (ACDOCA)
UI	SAP GUI	SAP Fiori
Real-time	Batch processing	Real-time analytics

ABAP changes in S/4HANA:

- New Open SQL (with @, inline declarations)
- CDS Views replace traditional views
- AMDP for complex database logic
- Obsolete statements removed

Keywords: ACDOCA, FIORI, simplification, HANA database, code adaptation

---

Q41. What are important T-Codes every ABAP developer should know?

Frequency: ★★★★★ | **Probability: 92%**

Answer:

T-Code	Purpose
SE38	ABAP Editor
SE37	Function Module
SE11	Data Dictionary
SE16/SE16N	Table browser
SE80	Object Navigator
SE18/SE19	BADI
SE10	Transport requests
STMS	Transport Management
ST05	SQL Trace
SAT	Runtime analysis

-

SM50	Work process monitor
SU01	User maintenance
SMARTFORMS	Smart Forms
SFP	Adobe Forms
WE02	IDoc display
BD87	IDoc reprocess

Q42. What is the difference between AT NEW and ON CHANGE OF?

Frequency: ★★★ | Probability: 65%

Answer:

Feature	AT NEW	ON CHANGE OF
Used in	LOOP (Classic reports)	Flat code
Triggers when	Field value changes during loop	Variable changes
Context	Control break statements	General

```
LOOP AT lt_data INTO wa_data.
  AT NEW field1.
    " Header processing
  ENDAT.
  AT END OF field1.
    " Footer/summary
  ENDAT.
ENDLOOP.
```

Q43. What is a Background Job? How do you schedule one?

Frequency: ★★★★ | Probability: 78%

Answer: Background jobs run ABAP programs without user interaction, typically for large data processing or scheduled tasks.

-

T-Codes:

T-Code	Purpose
SM36	Schedule background job
SM37	Monitor background jobs
SM38	Job queue

Steps to schedule:

1. Go to SM36
2. Enter job name and class
3. Add step (program name)
4. Set start condition (immediate / periodic / event-based)
5. Save and release

Keywords: SM36, SM37, job class (A/B/C), spool, background processing

---

Q44. What is the difference between MOVE and MOVE-CORRESPONDING?

Frequency: ★★★★ | Probability: 75%

Answer:

```
" MOVE — copies entire structure
MOVE wa_source TO wa_target.

" MOVE-CORRESPONDING — copies only matching field names
MOVE-CORRESPONDING wa_mara TO wa_custom.
```

MOVE-CORRESPONDING is useful when source and target have different structures but share common field names.

Q45. What is the MESSAGE statement in ABAP?

Frequency: ★★★★ | Probability: 78%

Answer: MESSAGE is used to display messages to users.

-

Message types:

Type	Description	Behavior
S	Success	Green — informational
I	Information	Popup
W	Warning	Yellow — can continue
E	Error	Red — stops processing
A	Abend	Terminates program
X	Exit	Short dump

```
MESSAGE 'Record saved successfully' TYPE 'S'.
MESSAGE e001(zmsg) WITH lv_matnr.
```

Keywords: MESSAGE, message class, SE91, message type, T100

Q46. What is the difference between SUBMIT and CALL TRANSACTION?

Frequency: ★★★★ | Probability: 75%

Answer:

Feature	SUBMIT	CALL TRANSACTION
Calls	Reports / Programs	Transactions (T-codes)
Data exchange	Via memory / parameters	Via BDC/BAPI
Return	Optional	Returns to caller

```
" Submit a report
SUBMIT zmy_report WITH s_matnr IN lr_matnr AND RETURN.

" Call a transaction
CALL TRANSACTION 'MM01'.
```

•

Q47. What is BDC (Batch Data Communication)?

Frequency: ★★★★ | Probability: 75%

Answer: BDC is a technique to upload data into SAP by simulating screen transactions programmatically. It fills in screen fields and triggers function keys automatically.

Methods:

1. **Call Transaction** — Online, immediate processing
2. **Session Method** — Batch processing via SM35

```
" BDC structure
DATA: lt_bdcdata TYPE TABLE OF bdcdata.

" Fill screen data
PERFORM bdc_dynpro USING 'SAPMM60X' '0100'.
PERFORM bdc_field  USING 'BDC_CURSOR' 'MATNR'.
PERFORM bdc_field  USING 'MATNR'      '100'.

CALL TRANSACTION 'MM60' USING lt_bdcdata MODE 'N'.
```

Keywords: BDCDATA, CALL TRANSACTION, SM35, session method, recording (SHDB)

Q48. What is the difference between IMPORT and EXPORT with MEMORY ID?

Frequency: ★★★ | Probability: 65%

Answer: Used to pass data between programs using SAP memory.

```
" Export to memory
EXPORT lv_matnr TO MEMORY ID 'ZMAT'.

" Import from memory (in another program)
IMPORT lv_matnr FROM MEMORY ID 'ZMAT'.

" Clear memory
FREE MEMORY ID 'ZMAT'.
```

Keywords: EXPORT TO MEMORY, IMPORT FROM MEMORY, FREE MEMORY, cross-program communication

---

-

Q49. What is the difference between SY-DATUM and SY-UZEIT?

Frequency: ★★★★ | Probability: 82%

Answer:

- `SY-DATUM` — Current system date (type DATS, format YYYYMMDD)
- `SY-UZEIT` — Current system time (type TIMS, format HHMMSS)

WRITE: 'Date:', sy-datum, 'Time:', sy-uzeit.

Q50. What is New Open SQL? What are its advantages?

Frequency: ★★★★★ | Probability: 88%

Answer: New Open SQL was introduced with S/4HANA and ABAP 7.4+. It brings modern syntax closer to standard SQL.

Key features:

```
" Inline declaration
SELECT matnr, mtart
  FROM mara
  INTO TABLE @DATA(lt_mara)
  WHERE mtart = 'FERT'.

" Aggregate functions
SELECT matnr, COUNT(*) AS cnt
  FROM mara
  INTO TABLE @DATA(lt_count)
  GROUP BY matnr.

" CASE expression
SELECT matnr,
       CASE mtart WHEN 'FERT' THEN 'Finished' ELSE 'Other' END AS type
  FROM mara
  INTO TABLE @DATA(lt_result).

" String functions
SELECT UPPER( matnr ) AS matnr_upper
  FROM mara INTO TABLE @DATA(lt_upper).
```

Advantages:

-

- Inline declarations with DATA()
- Aggregate functions in SELECT
- String functions
- CASE expressions
- Subqueries
- No need for separate data declaration

Keywords: @DATA(), inline declaration, ABAP 7.4, S/4HANA, aggregate functions, CASE

SECTION 2 — Top 30 HR Questions

Accenture | SAP ABAP Developer | Fresher to 2 Years

Q1. Tell me about yourself.

Frequency: ★★★★★ | Probability: 99%

What interviewer evaluates: Communication, confidence, relevance, how well you present yourself professionally.

Structure to follow: Education → Skills → Projects → Why this role

Sample Answer (Fresher): "Good morning! My name is [Your Name]. I completed my B.Tech in Computer Science from [College Name] in [Year] with [CGPA]. During my academics, I developed a strong interest in ERP systems and enterprise application development, which led me to pursue SAP ABAP training. I have hands-on experience in ABAP programming covering data dictionary, reports, ALV, function modules, BADIs, and basic S/4HANA concepts like CDS Views and New Open SQL. I completed a project where I built custom sales reports using ALV and integrated data from VBAK and VBAP tables. I am a quick learner, a team player, and I am genuinely excited about this opportunity at Accenture to apply my SAP skills in a real-world environment."

STAR Format:

- **Situation:** Fresh graduate with SAP ABAP training
- **Task:** Present background clearly and confidently
- **Action:** Highlight education, skills, project
- **Result:** Show readiness for the role

Common mistakes:

- Talking for more than 2–3 minutes
- Reading from memory robotically
- Mentioning irrelevant personal details (family background, hobbies unrelated to work)
- Starting with "I am a very hardworking person" — too generic

Keywords to use: SAP ABAP, S/4HANA, Data Dictionary, ALV, CDS Views, team player, quick learner, problem solving

---

Q2. Why do you want to join Accenture?

Frequency: ★★★★★ | Probability: 95%

-

What interviewer evaluates: Company knowledge, genuine motivation, cultural fit.

Sample Answer: "Accenture is one of the world's leading technology and consulting companies with a strong SAP practice globally. I am particularly drawn to Accenture because of its focus on innovation, digital transformation, and its investment in SAP S/4HANA projects. The learning culture, global exposure, and structured career development that Accenture offers align perfectly with my career goals. I want to grow as an SAP ABAP developer working on large-scale enterprise projects, and I believe Accenture is the ideal platform for that journey."

Common mistakes:

- Saying "for a good salary" or "for job security"
- Giving a generic answer without mentioning Accenture specifically
- Not showing any research about the company

Tips: Mention Accenture's SAP Center of Excellence, their digital transformation focus, or any recent news about Accenture's SAP projects.

Q3. Why SAP ABAP? Why did you choose this technology?

Frequency: ★★★★★ | Probability: 92%

What interviewer evaluates: Passion for the technology, clarity of career direction.

Sample Answer: "During my academics, I became fascinated by how large enterprises manage their operations using ERP systems. When I explored SAP, I realized it powers some of the largest companies in the world. ABAP, being the backbone of SAP development, opened a world of possibilities — from building custom reports to integrating systems using BAPIs and RFCs. The transition to S/4HANA and the introduction of modern concepts like CDS Views and AMDP made it even more exciting. I saw ABAP as a skill with strong industry demand and long-term career growth, which is why I invested in learning it deeply."

Common mistakes:

- Saying "my college told me" or "my friend suggested"
- Lacking enthusiasm in the answer
- Not connecting SAP to real business value

Q4. What are your strengths?

Frequency: ★★★★★ | Probability: 90%

-

What interviewer evaluates: Self-awareness, relevance to the job role.

Sample Answer: "My key strengths are analytical thinking, attention to detail, and adaptability. As an ABAP developer, I enjoy breaking down complex business requirements into clean, optimized code. I am also a fast learner — for example, I independently learned CDS Views and New Open SQL concepts while preparing for S/4HANA development. Additionally, I communicate well with team members, which helps during requirement gathering and testing phases."

STAR Format:

- Strength: Fast learner
- Situation: Self-learning CDS Views and S/4HANA concepts
- Action: Studied documentation, practiced in SAP trial system
- Result: Successfully implemented CDS view in my training project

Common mistakes:

- Listing too many strengths (pick 2–3 strong ones)
- Saying generic strengths like "I am hardworking" without proof
- Not connecting strengths to the job role

Q5. What are your weaknesses?

Frequency: ★★★★★ | **Probability: 88%**

What interviewer evaluates: Self-awareness, honesty, growth mindset.

Sample Answer: "One area I am actively working on is public speaking and presenting technical solutions to non-technical stakeholders. In group settings, I sometimes focus too much on technical details rather than simplifying the message for a business audience. I have been addressing this by practicing structured explanations and documenting my solutions clearly. I believe this is something I will naturally improve as I gain more client-facing experience."

Golden rule: Always mention a real weakness, then immediately explain what you are doing to overcome it.

Common mistakes:

- Saying "I have no weaknesses" — red flag
- Saying a strength disguised as weakness: "I work too hard" — interviewers see through this
- Mentioning a weakness critical to the job (e.g., "I am not good at coding")

-

Q6. Where do you see yourself in 5 years?

Frequency: ★★★★★ | Probability: 88%

What interviewer evaluates: Ambition, career planning, commitment to the company.

Sample Answer: "In the next 5 years, I see myself growing into a Senior SAP ABAP Developer with strong expertise in S/4HANA, CDS Views, and Fiori integration. I aim to earn SAP certifications and take on responsibilities like leading development sprints, mentoring junior developers, and contributing to architecture decisions. Within Accenture, I hope to work on global SAP implementation projects and grow into a consultant role where I can bridge business and technical requirements effectively."

Common mistakes:

- Saying "I want to start my own company in 2 years" — shows you plan to leave soon
- Being too vague: "I just want to grow"
- Setting unrealistic expectations like "I want to be a manager in 1 year"

Q7. Tell me about a challenge you faced and how you overcame it.

Frequency: ★★★★★ | Probability: 85%

What interviewer evaluates: Problem-solving ability, resilience, practical thinking.

STAR Answer:

- **Situation:** During my training project, I had to fetch data from multiple SAP tables and display it in an ALV report. The report was running very slowly with large datasets.
- **Task:** I needed to optimize the performance without changing the business logic.
- **Action:** I used ST05 to trace the SQL queries and identified a full table scan on VBAP. I added a proper WHERE clause, replaced a nested SELECT inside a loop with FOR ALL ENTRIES, and used field symbols in the LOOP instead of a work area.
- **Result:** The report execution time reduced by approximately 60%, and the solution was accepted in testing.

Common mistakes:

- Making up a story that sounds unbelievable
- Not explaining what YOU specifically did
- Ending without a positive result

-

Q8. Are you comfortable with relocation?

Frequency: ★★★★★ | Probability: 90%

What interviewer evaluates: Flexibility, commitment to the role.

Sample Answer: "Yes, I am completely open to relocation. I understand that Accenture works on projects across different locations and I am prepared to adapt accordingly. Starting my career with exposure to different environments will only accelerate my growth."

Tip: If you have genuine constraints, be honest but frame it positively: "I prefer [city] but I am open to discussing relocation based on project requirements."

Q9. Are you comfortable working in shifts or rotational shifts?

Frequency: ★★★★★ | Probability: 85%

What interviewer evaluates: Flexibility, client-facing work readiness.

Sample Answer: "Yes, I am comfortable with rotational shifts. I understand that in a global company like Accenture, project timelines and client time zones sometimes require flexible working hours, and I am fully prepared to manage that."

Q10. What do you know about Accenture?

Frequency: ★★★★★ | Probability: 88%

What interviewer evaluates: Company research, genuine interest.

Sample Answer: "Accenture is a global professional services company headquartered in Dublin, Ireland, operating in over 120 countries with more than 700,000 employees. It provides services in Strategy and Consulting, Technology, Interactive, and Operations. Accenture has one of the largest SAP practices in the world and is a key partner in SAP S/4HANA transformation projects for Fortune 500 companies. The company is known for its innovation culture, investment in employee training through platforms like Accenture Learning, and its focus on responsible business and sustainability."

Common mistakes:

- Saying "I don't know much about Accenture"
- Giving only surface-level information like "it is a big IT company"

-

Q11. Why should we hire you?

Frequency: ★★★★★ | Probability: 88%

What interviewer evaluates: Confidence, value proposition, self-awareness.

Sample Answer: "You should hire me because I bring a combination of strong SAP ABAP fundamentals, a practical project background, and a genuine passion for enterprise technology. I have hands-on experience with ABAP programming, data dictionary, ALV reports, function modules, and S/4HANA concepts like CDS Views and New Open SQL. I am a quick learner who adapts fast to new technologies. More importantly, I am someone who takes ownership of my work, pays attention to detail, and is committed to delivering quality output. I am confident that I can contribute meaningfully from day one and grow into a strong asset for Accenture's SAP practice."

Common mistakes:

- Being too humble: "I am not sure but I will try my best"
- Repeating your resume without adding value
- Not being specific about what you bring

Q12. Tell me about your project.

Frequency: ★★★★★ | Probability: 95%

What interviewer evaluates: Technical depth, communication, real experience.

Sample Answer (Fresher with training project): "During my SAP ABAP training, I worked on a Sales Report Optimization project. The objective was to develop a custom ALV report that displays sales order data by fetching information from VBAK (Sales Order Header), VBAP (Sales Order Items), and KNA1 (Customer Master). I created the selection screen with SELECT-OPTIONS for sales order range and date range. The data was fetched using JOIN and FOR ALL ENTRIES to optimize performance. The output was displayed using CL_SALV_TABLE with sorting, filtering, and subtotal features. I also created a transport request in SE10 and moved the object through the DEV landscape. One key challenge I faced was optimizing the query performance, which I resolved using ST05 to identify and fix a full table scan."

Keywords to mention: VBAK, VBAP, KNA1, FOR ALL ENTRIES, ALV, CL_SALV_TABLE, ST05, SE10, transport request

---

Q13. What is your greatest achievement?

Frequency: ★★★★ | Probability: 78%

-

STAR Answer: "My greatest achievement so far is successfully completing my SAP ABAP certification and delivering a fully functional custom sales report as part of my training project. I started with zero SAP knowledge and within [X months] I was able to build end-to-end ABAP programs covering reports, function modules, and basic S/4HANA concepts. This achievement gave me confidence in my technical abilities and validated my decision to pursue SAP as a career."

Q14. How do you handle pressure and tight deadlines?

Frequency: ★★★★ | Probability: 80%

STAR Answer:

- **Situation:** During my training project, I had to deliver a working ALV report within 3 days while also preparing for a certification.
- **Task:** Complete both without compromising quality.
- **Action:** I created a priority list, broke the work into smaller tasks, worked in focused 2-hour blocks, and avoided distractions. I also asked my trainer for clarification on one tricky requirement rather than wasting time guessing.
- **Result:** I delivered the report on time and cleared the certification in the same week.

Common mistakes:

- Saying "I don't handle pressure well"
- Giving a vague answer with no example

Q15. Are you a team player or do you prefer working alone?

Frequency: ★★★★ | Probability: 78%

Sample Answer: "I am comfortable with both. I enjoy collaborating with team members because diverse perspectives lead to better solutions — for example, during code reviews, I learned better optimization techniques from peers. At the same time, I can work independently and take ownership of tasks without constant supervision. I believe the best developers know when to collaborate and when to focus individually."

Q16. What motivates you?

Frequency: ★★★★ | Probability: 75%

-

Sample Answer: "I am motivated by problem-solving and seeing my work create real business impact. In SAP development, when a custom report I built helps a business user make faster decisions, or when an optimization I applied reduces a report's runtime significantly — that sense of impact drives me. I am also motivated by continuous learning, especially in a fast-evolving field like S/4HANA and Fiori development."

Q17. Do you have any questions for us?

Frequency: ★★★★★ | Probability: 95%

What interviewer evaluates: Curiosity, engagement, preparation.

Good questions to ask:

- "What does the typical onboarding and training process look like for a new SAP ABAP developer at Accenture?"
- "What kind of SAP projects will I be working on initially?"
- "How does Accenture support continuous learning and SAP certification for developers?"
- "What does a typical day look like for a junior SAP ABAP developer on the team?"

Common mistakes:

- Saying "No, I have no questions" — missed opportunity
- Asking about salary in the first technical round
- Asking questions that are easily found on Accenture's website

---

Q18. Tell me about a time you worked in a team.

Frequency: ★★★★ | Probability: 80%

STAR Answer:

- **Situation:** During my college final year project, our team of 4 had to build a web application within 2 months.
- **Task:** I was responsible for the backend database module.
- **Action:** I coordinated daily with the frontend team to align API structures, tracked progress using a shared document, and helped a teammate debug a critical issue two days before submission.
- **Result:** We delivered the project on time and received the highest grade in our batch.

Q19. How do you keep yourself updated with new technology?

Frequency: ★★★★ | Probability: 75%

Sample Answer: "I follow SAP Community (community.sap.com) regularly for ABAP and S/4HANA updates. I also watch SAP TechEd sessions on YouTube, read SAP blogs, and practice on SAP's free trial systems. For general technology trends, I follow LinkedIn Learning and relevant GitHub repositories. Recently, I have been focusing on understanding Fiori and CDS Views to align with the S/4HANA roadmap."

Q20. What is your expected salary?

Frequency: ★★★★★ | Probability: 85%

Sample Answer (Fresher): "As a fresher, I am open to the industry-standard compensation that Accenture offers for this role. I am more focused on the learning opportunity, the project exposure, and the growth path at this stage of my career. I trust Accenture's HR team to offer a fair package."

Tip: If pressed for a number, research the Accenture fresher package for your city/role on Glassdoor or AmbitionBox and quote a realistic range.

Q21. How quickly can you learn new technologies?

Frequency: ★★★★ | Probability: 78%

Sample Answer: "I believe I am a fast learner with a structured approach. When I started SAP ABAP, I had no prior exposure to ERP systems. Within [X months] of focused learning, I was able to build custom reports, function modules, and understand S/4HANA concepts. My approach is — first understand the concept, then practice with a small example, then apply it in a real scenario. This cycle helps me retain new knowledge effectively."

Q22. Describe yourself in three words.

Frequency: ★★★★ | Probability: 72%

Sample Answer: "Curious, disciplined, and dependable. Curious because I always want to understand how things work at a deeper level. Disciplined because I follow through on

-

commitments and manage my time well. Dependable because my teammates and trainers know they can count on me to deliver."

Q23. What do you do outside of work/study?

Frequency: ★★★ | Probability: 62%

Sample Answer: "Outside of studying, I enjoy solving logical puzzles and reading about technology trends in the SAP ecosystem. I also spend time on SAP Community forums and occasionally help peers debug their ABAP code. I find that engaging with the community keeps me sharp and exposes me to real-world problems I might not encounter in training alone."

Q24. Have you ever failed at something? What did you learn?

Frequency: ★★★★ | Probability: 75%

STAR Answer:

- **Situation:** In my first attempt at a mock SAP certification test, I did not pass.
- **Task:** I needed to analyze my gaps and prepare better.
- **Action:** I reviewed all the topics where I scored low, practiced more hands-on exercises, and focused on time management during the test.
- **Result:** I cleared the actual certification with a confident score and learned that structured preparation matters more than last-minute studying.

Q25. How do you prioritize tasks when given multiple assignments?

Frequency: ★★★★ | Probability: 78%

Sample Answer: "I use a simple priority framework — I first identify which tasks are urgent and high-impact, then sequence them accordingly. I communicate proactively with my supervisor if I see a conflict between deadlines. I also break large tasks into smaller milestones so progress is visible and manageable. In my training project, I used this approach to balance report development, testing, and documentation simultaneously."

Q26. What do you know about Agile methodology?

Frequency: ★★★★ | Probability: 80%

Sample Answer: "Agile is an iterative software development methodology where work is delivered in short cycles called sprints, typically 2 weeks long. The key practices include daily standups, sprint planning, backlog grooming, and sprint retrospectives. In SAP projects, Agile is increasingly used especially in S/4HANA implementations using SAP Activate methodology. I am familiar with Agile concepts and comfortable working in sprint-based delivery models."

Keywords: Sprint, Scrum, SAP Activate, backlog, standup, retrospective, iterative

---

Q27. Are you open to learning other SAP modules?

Frequency: ★★★★ | Probability: 78%

Sample Answer: "Absolutely. While my core strength is SAP ABAP development, I understand that having functional knowledge of modules like MM, SD, or FI makes me a better developer because I can understand business requirements better. I am open to learning adjacent technologies like SAP Fiori, BTP (Business Technology Platform), and integration tools like SAP Integration Suite as the project demands."

Q28. How do you handle constructive criticism?

Frequency: ★★★ | Probability: 65%

Sample Answer: "I welcome constructive criticism because it helps me improve. During my training, my mentor pointed out that my SELECT statements were fetching unnecessary fields, impacting performance. Instead of being defensive, I understood the reasoning, corrected it, and from that point made it a habit to always select only required fields. I believe feedback is one of the fastest ways to grow."

Q29. What makes you different from other candidates?

Frequency: ★★★★ | Probability: 78%

Sample Answer: "What sets me apart is the combination of solid ABAP fundamentals and my genuine passion for S/4HANA development. I have not just learned syntax — I understand why certain approaches like using FOR ALL ENTRIES, field symbols, and CDS Views lead to better performance. I have applied these in practical projects. Additionally, I

-

am someone who documents my work well, communicates clearly, and takes initiative in learning beyond what is taught in class."

Q30. Do you have any SAP certifications?

Frequency: ★★★★ | Probability: 82%

Sample Answer (if certified): "Yes, I have completed [SAP certification name]. This certification validated my knowledge of ABAP programming, data dictionary, and core development concepts."

Sample Answer (if not certified): "I do not have a formal certification yet, but I have completed a structured SAP ABAP training program covering all core development areas. I am planning to pursue the SAP Certified Development Associate certification in the near future and am actively preparing for it."

Common mistakes:

- Lying about certifications — never do this
- Being apologetic about not having one — frame it positively with a plan

SECTION 3 — Most Important SAP Topics

Accenture | SAP ABAP Developer | Fresher to 2 Years

TOPIC 1 — ABAP Basics

Interview Frequency: ★★★★★

What is ABAP?

ABAP (Advanced Business Application Programming) is SAP's proprietary 4th generation programming language. It runs on the SAP NetWeaver Application Server and is used to develop SAP applications, reports, interfaces, and enhancements.

ABAP Program Structure

```
REPORT zmy_program.

" Data declarations
DATA: lv_name TYPE string VALUE 'SAP ABAP'.

" Selection screen
PARAMETERS: p_name TYPE string.

" Start of selection
START-OF-SELECTION.
  WRITE: 'Hello', lv_name.
```

Types of ABAP Programs

Type	Description	T-Code
Report	Standalone program for output	SE38
Module Pool	Screen-based transaction program	SE80
Function Group	Container for function modules	SE37
Class Pool	Container for global classes	SE24
Interface Pool	Container for interfaces	SE24
Type Group	Container for type definitions	SE11
Include	Reusable code included in programs	SE38

Important ABAP Keywords

Keyword	Purpose
DATA	Declare variables
TYPES	Define custom types
CONSTANTS	Define constants
WRITE	Output to screen
MOVE	Assign value
IF / ELSE / ENDIF	Conditional logic
DO / ENDDO	Fixed loop

•

WHILE / ENDWHILE	Conditional loop
CASE / WHEN / ENDCASE	Switch statement
PERFORM	Call subroutine

Data Types in ABAP

Type	Description	Example
C	Character	`DATA: lv_name TYPE c LENGTH 10`
N	Numeric character	`DATA: lv_num TYPE n LENGTH 5`
I	Integer	`DATA: lv_count TYPE i`
F	Floating point	`DATA: lv_price TYPE f`
P	Packed decimal	`DATA: lv_amt TYPE p DECIMALS 2`
D	Date (YYYYMMDD)	`DATA: lv_date TYPE d`
T	Time (HHMMSS)	`DATA: lv_time TYPE t`
STRING	Variable length string	`DATA: lv_text TYPE string`
XSTRING	Binary string	`DATA: lv_bin TYPE xstring`

TOPIC 2 — Data Dictionary (SE11)

Interview Frequency: ★★★★★

What is Data Dictionary?

The ABAP Data Dictionary (DDIC) is a central repository for all data definitions in SAP. It manages database tables, views, data elements, domains, structures, and type groups.

Key Objects in SE11

•

Object	Description
Table	Physical database table
Structure	Flat data structure (no DB table)
Data Element	Describes a field's business meaning
Domain	Defines technical attributes (data type, length, fixed values)
View	Logical combination of tables
Search Help	F4 help for input fields
Lock Object	Controls concurrent data access
Type Group	Custom type definitions

Domain vs Data Element

Feature	Domain	Data Element
Defines	Technical properties	Business meaning
Contains	Data type, length, fixed values	Field label, search help, documentation
Reused by	Multiple data elements	Multiple table fields

Example:

- Domain: `MATNR` — Character, length 18
- Data Element: `MATNR` — Uses domain MATNR, label "Material Number"
- Table field: `MARA-MATNR` — Uses data element MATNR

Important SAP Tables to Know

Table	Description
MARA	Material Master (General)
MARC	Material Master (Plant)
MARD	Material Master (Storage Location)
KNA1	Customer Master (General)
LFA1	Vendor Master (General)

VBAK	Sales Order Header
VBAP	Sales Order Items
EKKO	Purchase Order Header
EKPO	Purchase Order Items
BKPF	Accounting Document Header
BSEG	Accounting Document Segment
T001	Company Codes
TSTC	T-Code table
DD02L	SAP Tables metadata

TOPIC 3 — Internal Tables

Interview Frequency: ★★★★★

Types of Internal Tables

Type	Key	Duplicates	Access Method	Best For
Standard Table	Default (all fields)	Allowed	Index / Linear search	Sequential processing
Sorted Table	Defined key (sorted)	Optional	Binary search	Sorted access
Hashed Table	Unique key	Not allowed	Hash key only	Random access

Declaration Syntax

```
" Standard Table
DATA: lt_mara TYPE STANDARD TABLE OF mara.

" Sorted Table — unique key
DATA: lt_sorted TYPE SORTED TABLE OF mara
      WITH UNIQUE KEY matnr.

" Sorted Table — non-unique key
DATA: lt_sorted2 TYPE SORTED TABLE OF mara
```

-

  WITH NON-UNIQUE KEY mtart.

```
" Hashed Table
DATA: lt_hashed TYPE HASHED TABLE OF mara
      WITH UNIQUE KEY matnr.

" Internal table with custom structure
TYPES: BEGIN OF ty_material,
         matnr TYPE matnr,
         mtart TYPE mtart,
         mbrsh TYPE mbrsh,
       END OF ty_material.

DATA: lt_material TYPE STANDARD TABLE OF ty_material,
      wa_material TYPE ty_material.
```

Operations on Internal Tables

```
" Append a row
APPEND wa_material TO lt_material.

" Insert at specific position
INSERT wa_material INTO lt_material INDEX 1.

" Modify a row
MODIFY lt_material FROM wa_material INDEX 1.

" Delete a row
DELETE lt_material INDEX 1.
DELETE lt_material WHERE mtart = 'ROH'.

" Clear entire table
CLEAR lt_material.    " Clears header line only (old style)
REFRESH lt_material.  " Clears body
FREE lt_material.     " Clears and releases memory

" Modern — use CLEAR for work area, FREE for table
```

TOPIC 4 — Work Areas

Interview Frequency: ★★★★★

-

What is a Work Area?

A work area is a flat structure (single row) used to process data row by row from an internal table.

```
" Declare work area
DATA: wa_mara TYPE mara.

" Fill work area and append to table
wa_mara-matnr = '100'.
wa_mara-mtart = 'FERT'.
APPEND wa_mara TO lt_mara.

" Read from table into work area
LOOP AT lt_mara INTO wa_mara.
  WRITE: wa_mara-matnr, wa_mara-mtart.
ENDLOOP.
```

Work Area vs Header Line

Feature	Work Area	Header Line
Declaration	Separate DATA statement	Implicit (old WITH HEADER LINE)
SAP recommendation	Recommended	Obsolete — avoid
Code clarity	High	Low — error prone

TOPIC 5 — Field Symbols

Interview Frequency: ★★★★★

What are Field Symbols?

Field symbols are ABAP pointers. They reference an existing memory area directly without copying data.

```
FIELD-SYMBOLS: <fs_mara> TYPE mara.

" Assign and use
LOOP AT lt_mara ASSIGNING <fs_mara>.
```

```
  <fs_mara>-mtart = 'FERT'.  " Directly modifies table
ENDLOOP.

" Dynamic field access
FIELD-SYMBOLS: <fs_value> TYPE any.
ASSIGN COMPONENT 'MATNR' OF STRUCTURE wa_mara TO <fs_value>.
IF sy-subrc = 0.
  WRITE: <fs_value>.
ENDIF.

" Check if assigned
IF <fs_mara> IS ASSIGNED.
  " Safe to use
ENDIF.
```

Performance Comparison

Method	Speed	Memory
LOOP INTO work area	Slower	Copies data
LOOP ASSIGNING field symbol	Faster	No copy — direct reference

---

TOPIC 6 — Open SQL

Interview Frequency: ★★★★★

Basic Open SQL Statements

```
" SELECT single row
SELECT SINGLE matnr mtart
  FROM mara
  INTO (lv_matnr, lv_mtart)
  WHERE matnr = '100'.

" SELECT multiple rows
SELECT matnr mtart mbrsh
  FROM mara
  INTO TABLE lt_mara
  WHERE mtart = 'FERT'
  AND   mbrsh = 'M'.

" INSERT
INSERT mara FROM wa_mara.
```

```
INSERT mara FROM TABLE lt_mara.

" UPDATE
UPDATE mara SET mtart = 'ROH' WHERE matnr = '100'.
UPDATE mara FROM wa_mara.

" MODIFY (insert or update)
MODIFY mara FROM wa_mara.

" DELETE
DELETE mara WHERE matnr = '100'.
DELETE mara FROM TABLE lt_mara.
```

Aggregate Functions

```
SELECT COUNT(*) INTO lv_count FROM mara WHERE mtart = 'FERT'.
SELECT MAX( ersda ) INTO lv_date FROM mara.
SELECT MIN( ersda ) INTO lv_date FROM mara.
SELECT SUM( lbstf ) INTO lv_sum FROM mara.
SELECT AVG( lbstf ) INTO lv_avg FROM mara.
```

GROUP BY and ORDER BY

```
SELECT mtart COUNT(*) AS cnt
  FROM mara
  INTO TABLE lt_result
  GROUP BY mtart
  ORDER BY mtart.
```

TOPIC 7 — New Open SQL (S/4HANA)

Interview Frequency: ★★★★★

Key Differences from Classic Open SQL

Feature	Classic Open SQL	New Open SQL
Inline declaration	Not supported	`@DATA(lt_result)`
Host variable prefix	Not needed	`@` prefix required

•

String functions	Not available	UPPER, LOWER, CONCAT
CASE expression	Not available	Supported
Subqueries	Limited	Full support
Aggregate in SELECT	Limited	Full support

New Open SQL Examples

```
" Inline declaration
SELECT matnr, mtart, mbrsh
  FROM mara
  INTO TABLE @DATA(lt_mara)
  WHERE mtart = 'FERT'.

" CASE expression
SELECT matnr,
       CASE mtart
         WHEN 'FERT' THEN 'Finished Good'
         WHEN 'ROH'  THEN 'Raw Material'
         ELSE 'Other'
       END AS mat_type
  FROM mara
  INTO TABLE @DATA(lt_result).

" String functions
SELECT matnr,
       UPPER( matnr ) AS upper_mat,
       LENGTH( matnr ) AS mat_len
  FROM mara
  INTO TABLE @DATA(lt_str).

" Subquery
SELECT matnr FROM mara
  INTO TABLE @DATA(lt_sub)
  WHERE matnr IN ( SELECT matnr FROM marc WHERE werks = '1000' ).
```

TOPIC 8 — Joins

Interview Frequency: ★★★★★

Types of Joins

•

Join Type	Returns
INNER JOIN	Only matching rows in both tables
LEFT OUTER JOIN	All rows from left + matching from right
RIGHT OUTER JOIN	All rows from right + matching from left (rare in ABAP)
CROSS JOIN	Cartesian product (avoid in ABAP)

Join Examples

```
" INNER JOIN
SELECT a~vbeln a~erdat b~matnr b~netwr
  FROM vbak AS a
  INNER JOIN vbap AS b ON a~vbeln = b~vbeln
  INTO TABLE @DATA(lt_sales)
  WHERE a~erdat >= '20240101'.

" LEFT OUTER JOIN
SELECT k~kunnr k~name1 a~vbeln
  FROM kna1 AS k
  LEFT OUTER JOIN vbak AS a ON k~kunnr = a~kunnr
  INTO TABLE @DATA(lt_cust_orders).
```

JOIN vs FOR ALL ENTRIES

Feature	JOIN	FOR ALL ENTRIES
Processing location	Database	Application server
Performance	Better for small data	Better for large datasets
Empty table check	Not needed	Mandatory
Duplicates	Possible	Removed automatically

TOPIC 9 — Reports

Interview Frequency: ★★★★★

Types of Reports in ABAP

•

Type	Description
Classical Report	WRITE-based list output
Interactive Report	Drill-down using AT LINE-SELECTION
ALV Report	Grid/List with built-in features

Classical Report Structure

```
REPORT zclassical_report.

" Data declarations
DATA: lt_mara TYPE STANDARD TABLE OF mara,
      wa_mara TYPE mara.

" Selection screen
SELECT-OPTIONS: s_matnr FOR mara-matnr.
PARAMETERS:     p_mtart TYPE mtart.

" Events
INITIALIZATION.
  " Initialize default values

START-OF-SELECTION.
  SELECT * FROM mara INTO TABLE lt_mara
    WHERE matnr IN s_matnr
    AND   mtart  = p_mtart.

END-OF-SELECTION.
  LOOP AT lt_mara INTO wa_mara.
    WRITE: / wa_mara-matnr, wa_mara-mtart.
  ENDLOOP.
```

Report Events

Event	When triggered
INITIALIZATION	Before selection screen displays
AT SELECTION-SCREEN	During selection screen processing
START-OF-SELECTION	When user clicks Execute
END-OF-SELECTION	After all data fetched

TOP-OF-PAGE	At top of each new page
END-OF-PAGE	At bottom of each page

TOPIC 10 — ALV (ABAP List Viewer)

Interview Frequency: ★★★★★

Types of ALV

Type	Class/FM	Recommended
Function Module ALV	REUSE_ALV_GRID_DISPLAY	Old style
OOP ALV	CL_GUI_ALV_GRID	Mid-level
SALV ALV	CL_SALV_TABLE	Recommended for S/4HANA

SALV ALV — Complete Example

```
REPORT zalv_example.

DATA: lt_mara  TYPE STANDARD TABLE OF mara,
      lo_salv  TYPE REF TO cl_salv_table,
      lo_funcs TYPE REF TO cl_salv_functions_list,
      lo_cols  TYPE REF TO cl_salv_columns_table,
      lo_col   TYPE REF TO cl_salv_column_table.

START-OF-SELECTION.
  SELECT matnr mtart mbrsh meins
    FROM mara
    INTO TABLE lt_mara
    UP TO 100 ROWS.

  TRY.
    cl_salv_table=>factory(
      IMPORTING r_salv_table = lo_salv
      CHANGING  t_table      = lt_mara ).

    " Enable standard functions
    lo_funcs = lo_salv->get_functions( ).
    lo_funcs->set_all( abap_true ).

    " Display
    lo_salv->display( ).
```

```
CATCH cx_salv_msg INTO DATA(lo_ex).
  MESSAGE lo_ex->get_text( ) TYPE 'E'.
ENDTRY.
```

TOPIC 11 — Smart Forms

Interview Frequency: ★★★★

What are Smart Forms?

Smart Forms are SAP's tool for creating business forms like invoices, purchase orders, and delivery notes. They replace SAPscript.

Transaction: SMARTFORMS **Function module generated:** Automatically by SAP

Smart Form Structure

Component	Purpose
Global Settings	Page format, styles
Form Interface	Import/Export parameters
Global Definitions	Internal data declarations
Pages	Output pages
Windows	Areas on a page
Text Elements	Static/dynamic text
Table	Tabular data display
Program Lines	ABAP code nodes

How to Call a Smart Form

```
DATA: lv_fm_name TYPE rs38l_fnam.

" Get function module name
CALL FUNCTION 'SSF_FUNCTION_MODULE_NAME'
  EXPORTING
    formname = 'ZMY_SMARTFORM'
  IMPORTING
```

-

```
  fm_name  = lv_fm_name.

" Call the generated function module
CALL FUNCTION lv_fm_name
  EXPORTING
    control_parameters = ls_control
    ...
```

TOPIC 12 — Adobe Forms

Interview Frequency: ★★★★

What are Adobe Forms?

Adobe Forms (SAP Interactive Forms) are PDF-based forms designed using Adobe LiveCycle Designer. They are the preferred form technology in S/4HANA.

Transaction: SFP **Components:** Interface + Form Layout

Adobe vs Smart Forms

Feature	Smart Forms	Adobe Forms
Transaction	SMARTFORMS	SFP
Output format	SAPscript/PDF	PDF only
Interactive	No	Yes
S/4HANA preferred	No	Yes
Design tool	SAP GUI	Adobe LiveCycle Designer
Service	None	ADS (Adobe Document Services)

TOPIC 13 — Module Pool (Screen Programming)

Interview Frequency: ★★★★

-

What is Module Pool?

Module Pool programs are screen-based ABAP programs that create custom SAP transactions with multiple screens, input fields, and user interactions.

Transaction to create: SE80 or SE38 (type M) **Screens created in:** Screen Painter (SE51)

Module Pool Components

Component	Description
PBO	Process Before Output — runs before screen displays
PAI	Process After Input — runs after user action
Modules	ABAP code blocks called from PBO/PAI
Flow Logic	Screen flow control code
GUI Status	Toolbar buttons and menu options
GUI Title	Screen title bar

Basic Flow Logic

```
" Screen flow logic (in SE51)
PROCESS BEFORE OUTPUT.
  MODULE status_0100.

PROCESS AFTER INPUT.
  MODULE user_command_0100.

" ABAP code
MODULE status_0100 OUTPUT.
  SET PF-STATUS 'ZMAIN'.
  SET TITLEBAR 'ZTITLE'.
ENDMODULE.

MODULE user_command_0100 INPUT.
  CASE sy-ucomm.
    WHEN 'SAVE'.
      PERFORM save_data.
    WHEN 'BACK'.
      LEAVE TO SCREEN 0.
  ENDCASE.
ENDMODULE.
```

-

TOPIC 14 — Function Modules

Interview Frequency: ★★★★★

Structure of a Function Module

```
FUNCTION zmy_function_module.
*"----------------------------------------------
*"  IMPORTING
*"     VALUE(IV_MATNR) TYPE MATNR
*"  EXPORTING
*"     VALUE(EV_MTART) TYPE MTART
*"  EXCEPTIONS
*"     NOT_FOUND = 1
*"----------------------------------------------

  SELECT SINGLE mtart FROM mara
    INTO ev_mtart
    WHERE matnr = iv_matnr.

  IF sy-subrc <> 0.
    RAISE not_found.
  ENDIF.

ENDFUNCTION.
```

Calling a Function Module

```
CALL FUNCTION 'ZMY_FUNCTION_MODULE'
  EXPORTING
    iv_matnr = lv_matnr
  IMPORTING
    ev_mtart = lv_mtart
  EXCEPTIONS
    not_found = 1
    OTHERS    = 2.

IF sy-subrc = 1.
  MESSAGE 'Material not found' TYPE 'E'.
ENDIF.
```

TOPIC 15 — BAPI

Interview Frequency: ★★★★★

What is BAPI?

Business Application Programming Interface — RFC-enabled, standardized function modules for SAP business processes.

Important BAPIs

BAPI	Purpose
BAPI_SALESORDER_CREATEFROMDAT2	Create Sales Order
BAPI_SALESORDER_CHANGE	Change Sales Order
BAPI_GOODSMVT_CREATE	Post Goods Movement
BAPI_PO_CREATE1	Create Purchase Order
BAPI_ACC_DOCUMENT_POST	Post FI Document
BAPI_MATERIAL_SAVEDATA	Create/Change Material
BAPI_TRANSACTION_COMMIT	Commit BAPI changes
BAPI_TRANSACTION_ROLLBACK	Rollback BAPI changes

BAPI Call Pattern

```
" Always check return table for errors
CALL FUNCTION 'BAPI_SALESORDER_CREATEFROMDAT2'
  EXPORTING
    order_header_in = ls_header
  TABLES
    order_items_in  = lt_items
    return          = lt_return.

" Check errors
READ TABLE lt_return INTO wa_return
  WITH KEY type = 'E'.
IF sy-subrc = 0.
  " Error handling
  CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
ELSE.
```

```
  CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
    EXPORTING wait = 'X'.
ENDIF.
```

TOPIC 16 — RFC (Remote Function Call)

Interview Frequency: ★★★★

Types of RFC

Type	Behavior	Use Case
sRFC	Synchronous — waits for result	Real-time data exchange
aRFC	Asynchronous — no wait	Parallel processing
tRFC	Exactly-once delivery	Financial postings
qRFC	Ordered delivery	Sequential processing
bgRFC	Modern background RFC	Replaces tRFC/qRFC

RFC Call Syntax

```
CALL FUNCTION 'ZMY_FUNCTION'
  DESTINATION 'RFC_DESTINATION_NAME'
  EXPORTING
    iv_param = lv_value
  IMPORTING
    ev_result = lv_result
  EXCEPTIONS
    system_failure        = 1
    communication_failure = 2.
```

T-Code for RFC destinations: SM59

TOPIC 17 — Enhancements, User Exits, BADIs

-

Interview Frequency: ★★★★★

Enhancement Types Comparison

Type	Technology	Where	Flexibility
User Exit	FORM routine	Standard program includes	Low
Customer Exit	Function Module	EXIT_* function modules	Medium
BADI (Old)	OOP Interface	SE18/SE19	High
BADI (New)	Enhancement Framework	SE18/SE19	High
Enhancement Spot	Source code plug	Any ABAP program	Highest

User Exit vs BADI

Feature	User Exit	BADI
Technology	Procedural	Object-oriented
Multiple implementations	No	Yes (filter-based)
Modification-free	Yes	Yes
Flexibility	Limited	High

BADI Implementation Steps

1. **SE18** — Find BADI definition
2. **SE19** — Create implementation
3. Give implementation a name
4. Create class for the interface
5. Implement required methods
6. Activate

TOPIC 18 — Object-Oriented ABAP

Interview Frequency: ★★★★★

OOP Concepts in ABAP

Encapsulation

```
CLASS lcl_employee DEFINITION.
  PUBLIC SECTION.
    METHODS: get_salary RETURNING VALUE(rv_salary) TYPE p.
  PRIVATE SECTION.
    DATA: mv_salary TYPE p DECIMALS 2.
ENDCLASS.
```

Inheritance

```
CLASS lcl_manager DEFINITION
  INHERITING FROM lcl_employee.
  PUBLIC SECTION.
    METHODS: get_bonus RETURNING VALUE(rv_bonus) TYPE p.
ENDCLASS.
```

Polymorphism

```
" Method overriding
CLASS lcl_child DEFINITION INHERITING FROM lcl_parent.
  PUBLIC SECTION.
    METHODS: display REDEFINITION.
ENDCLASS.
```

Constructor

```
CLASS lcl_car DEFINITION.
  PUBLIC SECTION.
    METHODS: constructor IMPORTING iv_color TYPE string.
  PRIVATE SECTION.
    DATA: mv_color TYPE string.
ENDCLASS.

CLASS lcl_car IMPLEMENTATION.
  METHOD constructor.
    mv_color = iv_color.
  ENDMETHOD.
ENDCLASS.

" Usage
DATA: lo_car TYPE REF TO lcl_car.
CREATE OBJECT lo_car EXPORTING iv_color = 'Red'.
```

-

Access Modifiers

Modifier	Accessible From
PUBLIC	Anywhere
PROTECTED	Class itself + subclasses
PRIVATE	Class itself only

TOPIC 19 — Exception Handling

Interview Frequency: ★★★★

Class-Based Exceptions (Recommended)

```
TRY.
  " Risky code
  DATA(lv_result) = 10 / 0.

CATCH cx_sy_zerodivide INTO DATA(lo_div).
  WRITE: 'Division by zero:', lo_div->get_text( ).

CATCH cx_sy_conversion_no_number INTO DATA(lo_conv).
  WRITE: 'Conversion error:', lo_conv->get_text( ).

CATCH cx_root INTO DATA(lo_root).
  WRITE: 'Unknown error:', lo_root->get_text( ).

FINALLY.
  " Always executes — cleanup code
  WRITE: 'Processing complete'.
ENDTRY.
```

Exception Hierarchy

```
CX_ROOT
├── CX_STATIC_CHECK     (must be declared in method signature)
├── CX_DYNAMIC_CHECK    (runtime exceptions)
│   ├── CX_SY_ZERODIVIDE
│   ├── CX_SY_CONVERSION_NO_NUMBER
│   └── CX_SY_OPEN_SQL_DB
└── CX_NO_CHECK         (system exceptions — always possible)
```

-

Raising Custom Exceptions

```
CLASS cx_material_not_found DEFINITION
  INHERITING FROM cx_static_check.
ENDCLASS.

METHOD get_material.
  IF lv_found = abap_false.
    RAISE EXCEPTION TYPE cx_material_not_found.
  ENDIF.
ENDMETHOD.
```

TOPIC 20 — Debugging

Interview Frequency: ★★★★★

How to Start ABAP Debugger

Method	How
Command field	Type `/h` and press Enter
Hard breakpoint	`BREAK-POINT` in code
User breakpoint	`BREAK username` in code
External breakpoint	Set in SE38/SE80
Dynamic breakpoint	Set during debugging session

Debugger Key Controls

Key	Action
F5	Step Into
F6	Step Over
F7	Step Out / Return
F8	Continue to next breakpoint

-

Watchpoints

Pause execution when a specific variable changes value — very useful for tracking bugs in loops.

Debugging Tips

- Use **Display Variable** to check internal table content
- Use **Change Variable** to test different values
- Check **SY-SUBRC** after every database call
- Use **Table Control** view for internal tables

TOPIC 21 — Performance Optimization

Interview Frequency: ★★★★★

Golden Rules of ABAP Performance

Rule	Bad Practice	Good Practice
Select fields	`SELECT *`	`SELECT matnr mtart`
Loop + Select	SELECT inside LOOP	FOR ALL ENTRIES
Search	Linear search	Binary search / Hashed table
Table type	Standard for large random access	Hashed table
Field symbol	LOOP INTO work area	LOOP ASSIGNING field symbol
Aggregation	Fetch all, sum in ABAP	Use SUM/COUNT in SELECT
Joins	Multiple selects	Single JOIN
Buffering	No buffering	Use table buffering where applicable

Select Inside Loop — Classic Performance Bug

```
" BAD — N database calls
LOOP AT lt_vbak INTO wa_vbak.
  SELECT SINGLE * FROM vbap INTO wa_vbap
    WHERE vbeln = wa_vbak-vbeln.
ENDLOOP.
```

-

```
" GOOD — 1 database call
SELECT * FROM vbap INTO TABLE lt_vbap
  FOR ALL ENTRIES IN lt_vbak
  WHERE vbeln = lt_vbak-vbeln.
```

TOPIC 22 — ST05 / SAT / SQL Trace

Interview Frequency: ★★★★★

ST05 — Performance Trace

Steps:

1. Transaction ST05
2. Activate Trace
3. Execute program
4. Deactivate Trace
5. Display Trace

What to analyze:

- Full table scans (no WHERE clause)
- Missing WHERE conditions
- Expensive joins
- Repeated identical queries

SAT (Runtime Analysis — replaces SE30)

Purpose: Measures time spent in each ABAP statement, method, and function module.

Steps:

1. Transaction SAT
2. Set measurement variant
3. Execute
4. Analyze call hierarchy and time breakdown

Performance Tools Summary

T-Code	Purpose
ST05	SQL and database trace

-

SAT	ABAP runtime analysis
SM50	Active work process monitor
SM66	Global work process overview
AL11	Application server file directory
SM12	Lock entries

TOPIC 23 — CDS Views

Interview Frequency: ★★★★★

CDS View Syntax

```
@AbapCatalog.sqlViewName: 'ZSALES_VIEW'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Sales CDS View'

define view ZSALES_CDS as select from vbak
  association [0..*] to vbap as _items
    on $projection.vbeln = _items.vbeln
{
  key vbeln,
      erdat,
      kunnr,
      _items
}
```

CDS Annotations

Annotation	Purpose
@AbapCatalog.sqlViewName	DB view name
@EndUserText.label	Description
@AccessControl.authorizationCheck	Auth check
@Analytics.dataCategory	For analytical views
@OData.publish	Expose as OData service

-

@UI.lineItem	Fiori list column

CDS View Types

Type	Purpose
Basic View	Simple table access
Composite View	Combines multiple basic views
Consumption View	Fiori/OData exposure
Interface View	Reusable data model

TOPIC 24 — AMDP

Interview Frequency: ★★★★

When to use AMDP?

- Complex calculations impossible in Open SQL
- Need native HANA SQLScript features
- Maximum performance pushdown

AMDP Structure

```
CLASS zcl_amdp_demo DEFINITION
  PUBLIC FINAL CREATE PUBLIC.

  PUBLIC SECTION.
    INTERFACES if_amdp_marker_hdb.

    TYPES: tt_mara TYPE STANDARD TABLE OF mara
           WITH DEFAULT KEY.

    CLASS-METHODS get_materials
      IMPORTING VALUE(iv_mtart) TYPE mtart
      EXPORTING VALUE(et_mara)  TYPE tt_mara
      RAISING   cx_amdp_error.

ENDCLASS.

CLASS zcl_amdp_demo IMPLEMENTATION.
```

```
  METHOD get_materials
    BY DATABASE PROCEDURE FOR HDB
    LANGUAGE SQLSCRIPT
    USING mara.

   et_mara = SELECT * FROM mara
          WHERE mtart = :iv_mtart;

  ENDMETHOD.

ENDCLASS.
```

TOPIC 25 — SAP HANA Concepts

Interview Frequency: ★★★★★

Key HANA Concepts

Concept	Description
In-Memory Computing	Data stored in RAM for ultra-fast access
Column Store	Data stored column-wise — ideal for analytics
Row Store	Data stored row-wise — ideal for OLTP
Compression	HANA compresses column data highly
Parallel Processing	Multiple CPU cores process data simultaneously
MVCC	Multi-Version Concurrency Control for transactions

HANA vs Traditional Database

Feature	Traditional DB	SAP HANA
Storage	Disk	Memory (RAM)
Data layout	Row-based	Column + Row
Analytics	Separate OLAP system	Built-in
Speed	Seconds/minutes	Milliseconds

Compression	Low	Very high

TOPIC 26 — S/4HANA Basics

Interview Frequency: ★★★★★

What changed in S/4HANA for ABAP Developers?

Area	ECC	S/4HANA
Database	Any DB	HANA only
Data model	Complex/redundant	Simplified (ACDOCA)
UI	SAP GUI	SAP Fiori
SQL	Classic Open SQL	New Open SQL
Views	Database views	CDS Views
Reporting	ABAP reports	CDS + Fiori
Obsolete	None	Many statements removed

Simplified Data Model — Example

- ECC had: BKPF + BSEG + GLT0 + FAGLFLEXA (multiple FI tables)
- S/4HANA has: **ACDOCA** (Universal Journal — single source of truth)

Obsolete ABAP in S/4HANA

Obsolete	Replacement
SELECT *	SELECT specific fields
Header lines	Work areas
OCCURS	TYPE STANDARD TABLE OF
MOVE-CORRESPONDING (in some cases)	Explicit field mapping
Logical databases	Direct SELECT

•

TOPIC 27 — Transport Management

Interview Frequency: ★★★★

System Landscape

DEV (Development) → QAS (Quality/Testing) → PRD (Production)

Transport Request Lifecycle

1. Developer creates object in DEV
2. Object is automatically added to a transport request
3. Developer releases the task
4. Developer releases the transport request (SE10)
5. Basis team imports to QAS (STMS)
6. Testing done in QAS
7. Basis imports to PRD after approval

Key T-Codes

T-Code	Purpose
SE10	Create/manage transport requests
STMS	Transport Management System
SE09	Workbench transport requests
SCC1	Client copy within system

TOPIC 28 — Background Jobs

Interview Frequency: ★★★★

Job Classes

Class	Priority	Description
A	Highest	Critical business jobs

-

B	Medium	Standard batch jobs
C	Lowest	Low-priority background work

Job Scheduling (SM36)

1. Job Name
2. Job Class (A/B/C)
3. Steps — Program name + variant
4. Start Condition:
    - Immediate
    - Date/Time specific
    - After job event
    - After another job
    - Periodic (hourly/daily/weekly)

Monitoring (SM37)

- Check job status: Scheduled / Active / Finished / Cancelled
- View spool output
- Check job logs for errors

TOPIC 29 — Lock Objects

Interview Frequency: ★★★★

Lock Modes

Mode	Description
Shared Lock (S)	Multiple readers, no writers
Exclusive Lock (E)	Single user — no others can read or write
Exclusive Non-Cumulative (X)	Like E but cannot be set twice by same user
Optimistic Lock (O)	Allows reading, converts to exclusive on write

Lock Object Example

```
" Lock
CALL FUNCTION 'ENQUEUE_EZMATERIAL'
  EXPORTING
```

```
    matnr         = lv_matnr
  EXCEPTIONS
    foreign_lock   = 1
    system_failure = 2.

" Unlock
CALL FUNCTION 'DEQUEUE_EZMATERIAL'
  EXPORTING
    matnr = lv_matnr.
```

TOPIC 30 — IDoc Basics

Interview Frequency: ★★★★

IDoc Structure

Layer	Description
Control Record (EDIDC)	Header — sender, receiver, message type
Data Records (EDID4)	Segments containing business data
Status Records (EDIDS)	Processing status history

IDoc Terminology

Term	Description
IDoc Type	Technical structure definition
Message Type	Business process (ORDERS, INVOIC)
Partner Profile	WE20 — sender/receiver configuration
Port	Communication channel definition

Important IDoc T-Codes

T-Code	Purpose
WE60	IDoc type documentation

WE02/WE05	Display IDocs
WE19	Test IDoc processing
WE20	Partner profiles
BD87	Reprocess failed IDocs
SM58	tRFC monitor

Now Moving to Section 4 — Coding Questions

SECTION 4 — Top 30 Coding Questions

SAP ABAP | Fresher to 2 Years | Accenture

CODE Q1. Write a program to display all materials of type FERT using ALV.

```
REPORT zcode_q1.

DATA: lt_mara TYPE STANDARD TABLE OF mara,
      lo_salv TYPE REF TO cl_salv_table.

START-OF-SELECTION.

  SELECT matnr mtart mbrsh meins ersda
    FROM mara
    INTO TABLE lt_mara
    WHERE mtart = 'FERT'.

  IF lt_mara IS INITIAL.
    MESSAGE 'No records found' TYPE 'I'.
    RETURN.
  ENDIF.

  TRY.
    cl_salv_table=>factory(
```

```
    IMPORTING r_salv_table = lo_salv
    CHANGING  t_table      = lt_mara ).

  lo_salv->get_functions( )->set_all( abap_true ).
  lo_salv->display( ).

 CATCH cx_salv_msg INTO DATA(lo_ex).
  MESSAGE lo_ex->get_text( ) TYPE 'E'.
ENDTRY.
```

Expected output: ALV grid with FERT materials from MARA table.

CODE Q2. Write a program to join VBAK and VBAP and display sales order data.

```
REPORT zcode_q2.

TYPES: BEGIN OF ty_sales,
         vbeln TYPE vbak-vbeln,
         erdat TYPE vbak-erdat,
         kunnr TYPE vbak-kunnr,
         posnr TYPE vbap-posnr,
         matnr TYPE vbap-matnr,
         netwr TYPE vbap-netwr,
       END OF ty_sales.

DATA: lt_sales TYPE STANDARD TABLE OF ty_sales,
      lo_salv  TYPE REF TO cl_salv_table.

SELECT-OPTIONS: s_vbeln FOR vbak-vbeln.

START-OF-SELECTION.

  SELECT a~vbeln, a~erdat, a~kunnr,
         b~posnr, b~matnr, b~netwr
    FROM vbak AS a
    INNER JOIN vbap AS b ON a~vbeln = b~vbeln
    INTO TABLE @lt_sales
    WHERE a~vbeln IN @s_vbeln.

  TRY.
    cl_salv_table=>factory(
      IMPORTING r_salv_table = lo_salv
      CHANGING  t_table      = lt_sales ).
```

```
  lo_salv->get_functions( )->set_all( abap_true ).
  lo_salv->display( ).
CATCH cx_salv_msg INTO DATA(lo_ex).
  MESSAGE lo_ex->get_text( ) TYPE 'E'.
ENDTRY.
```

CODE Q3. Write a program using FOR ALL ENTRIES to fetch VBAP for given VBAK records.

```
REPORT zcode_q3.

DATA: lt_vbak TYPE STANDARD TABLE OF vbak,
      lt_vbap TYPE STANDARD TABLE OF vbap,
      wa_vbak TYPE vbak.

SELECT-OPTIONS: s_erdat FOR vbak-erdat.

START-OF-SELECTION.

 " Step 1 — Fetch header
 SELECT * FROM vbak
   INTO TABLE lt_vbak
   WHERE erdat IN s_erdat.

 " Step 2 — Always check before FOR ALL ENTRIES
 IF lt_vbak IS NOT INITIAL.
   SELECT * FROM vbap
     INTO TABLE lt_vbap
     FOR ALL ENTRIES IN lt_vbak
     WHERE vbeln = lt_vbak-vbeln.
 ENDIF.

 " Display count
 WRITE: / 'Headers:', lines( lt_vbak ),
        / 'Items:',   lines( lt_vbap ).
```

CODE Q4. Demonstrate LOOP with field symbols vs work area — performance difference.

```
REPORT zcode_q4.

DATA: lt_mara TYPE STANDARD TABLE OF mara.
```

```
SELECT * FROM mara INTO TABLE lt_mara UP TO 1000 ROWS.

" Method 1 — Work Area (slower)
DATA: wa_mara TYPE mara.
LOOP AT lt_mara INTO wa_mara.
  wa_mara-mtart = 'FERT'.
  MODIFY lt_mara FROM wa_mara.
ENDLOOP.

" Method 2 — Field Symbol (faster — direct reference)
FIELD-SYMBOLS: <fs_mara> TYPE mara.
LOOP AT lt_mara ASSIGNING <fs_mara>.
  <fs_mara>-mtart = 'ROH'.  " Direct modification — no MODIFY needed
ENDLOOP.

WRITE: 'Processing complete. Rows:', lines( lt_mara ).
```

CODE Q5. Write a program to sort an internal table and remove duplicates.

```
REPORT zcode_q5.

TYPES: BEGIN OF ty_data,
         matnr TYPE matnr,
         mtart TYPE mtart,
       END OF ty_data.

DATA: lt_data TYPE STANDARD TABLE OF ty_data,
      wa_data TYPE ty_data.

" Fill with duplicates
wa_data-matnr = '100'. wa_data-mtart = 'FERT'. APPEND wa_data TO lt_data.
wa_data-matnr = '200'. wa_data-mtart = 'ROH'.  APPEND wa_data TO lt_data.
wa_data-matnr = '100'. wa_data-mtart = 'FERT'. APPEND wa_data TO lt_data.
wa_data-matnr = '300'. wa_data-mtart = 'HALB'. APPEND wa_data TO lt_data.

WRITE: / 'Before:', lines( lt_data ), 'rows'.

" Sort first — mandatory
SORT lt_data BY matnr mtart.

" Remove duplicates
DELETE ADJACENT DUPLICATES FROM lt_data COMPARING matnr mtart.
```

```
WRITE: / 'After:', lines( lt_data ), 'rows'.

LOOP AT lt_data INTO wa_data.
  WRITE: / wa_data-matnr, wa_data-mtart.
ENDLOOP.
```

Expected output:

```
Before: 4 rows
After: 3 rows
100  FERT
200  ROH
300  HALB
```

CODE Q6. Write a program to READ TABLE with Binary Search.

```
REPORT zcode_q6.

DATA: lt_mara TYPE SORTED TABLE OF mara
        WITH NON-UNIQUE KEY matnr,
      wa_mara TYPE mara.

SELECT * FROM mara INTO TABLE lt_mara UP TO 500 ROWS.

" Binary search on sorted table — O(log n)
READ TABLE lt_mara INTO wa_mara
  WITH KEY matnr = '000000000000000100'
  BINARY SEARCH.

IF sy-subrc = 0.
  WRITE: / 'Found:', wa_mara-matnr, wa_mara-mtart.
ELSE.
  WRITE: / 'Not found'.
ENDIF.
```

CODE Q7. Write a String manipulation program in ABAP.

```
REPORT zcode_q7.

DATA: lv_str    TYPE string VALUE 'Hello SAP ABAP World',
```

```
      lv_upper  TYPE string,
      lv_lower  TYPE string,
      lv_length TYPE i,
      lv_sub    TYPE string,
      lv_pos    TYPE i.

" String functions (New ABAP)
lv_upper  = to_upper( lv_str ).
lv_lower  = to_lower( lv_str ).
lv_length = strlen( lv_str ).
lv_sub    = substring( val = lv_str off = 6 len = 3 ).

" Find position
FIND 'SAP' IN lv_str MATCH OFFSET lv_pos.

WRITE: / 'Original:', lv_str.
WRITE: / 'Upper:',    lv_upper.
WRITE: / 'Lower:',    lv_lower.
WRITE: / 'Length:',   lv_length.
WRITE: / 'Substring (pos 6, len 3):', lv_sub.
WRITE: / 'Position of SAP:', lv_pos.

" Concatenate
DATA: lv_result TYPE string.
CONCATENATE 'Hello' ' ' 'World' INTO lv_result.
WRITE: / 'Concat:', lv_result.

" New style
lv_result = |Hello { 'SAP' } World|.
WRITE: / 'Template:', lv_result.
```

CODE Q8. Write a program using COLLECT to aggregate data.

```
REPORT zcode_q8.

TYPES: BEGIN OF ty_sales,
         region  TYPE char10,
         revenue TYPE p DECIMALS 2,
       END OF ty_sales.

DATA: lt_raw    TYPE STANDARD TABLE OF ty_sales,
      lt_total  TYPE STANDARD TABLE OF ty_sales,
      wa_sales  TYPE ty_sales.
```

•

```
" Raw data
wa_sales-region = 'NORTH'. wa_sales-revenue = 5000. APPEND wa_sales TO lt_raw.
wa_sales-region = 'SOUTH'. wa_sales-revenue = 3000. APPEND wa_sales TO lt_raw.
wa_sales-region = 'NORTH'. wa_sales-revenue = 2000. APPEND wa_sales TO lt_raw.
wa_sales-region = 'SOUTH'. wa_sales-revenue = 1500. APPEND wa_sales TO lt_raw.

" Aggregate using COLLECT
LOOP AT lt_raw INTO wa_sales.
  COLLECT wa_sales INTO lt_total.
ENDLOOP.

" Display
WRITE: / 'Region    Revenue'.
WRITE: / '--------------------'.
LOOP AT lt_total INTO wa_sales.
  WRITE: / wa_sales-region, wa_sales-revenue.
ENDLOOP.
```

Expected output:

```
NORTH   7000.00
SOUTH   4500.00
```

CODE Q9. Exception handling with TRY-CATCH.

```
REPORT zcode_q9.

DATA: lv_num1   TYPE i VALUE 10,
      lv_num2   TYPE i VALUE 0,
      lv_result TYPE i.

TRY.
  lv_result = lv_num1 / lv_num2.
  WRITE: / 'Result:', lv_result.

CATCH cx_sy_zerodivide INTO DATA(lo_ex).
  WRITE: / 'Error: Division by zero caught!'.
  WRITE: / lo_ex->get_text( ).

FINALLY.
  WRITE: / 'Program execution complete.'.
ENDTRY.
```

CODE Q10. Write a program to create a simple OOP class and use it.

```
REPORT zcode_q10.

CLASS lcl_calculator DEFINITION.
  PUBLIC SECTION.
    METHODS:
      add      IMPORTING iv_a TYPE i iv_b TYPE i
               RETURNING VALUE(rv_result) TYPE i,
      subtract IMPORTING iv_a TYPE i iv_b TYPE i
               RETURNING VALUE(rv_result) TYPE i,
      multiply IMPORTING iv_a TYPE i iv_b TYPE i
               RETURNING VALUE(rv_result) TYPE i.
ENDCLASS.

CLASS lcl_calculator IMPLEMENTATION.
  METHOD add.
    rv_result = iv_a + iv_b.
  ENDMETHOD.
  METHOD subtract.
    rv_result = iv_a - iv_b.
  ENDMETHOD.
  METHOD multiply.
    rv_result = iv_a * iv_b.
  ENDMETHOD.
ENDCLASS.

START-OF-SELECTION.
  DATA: lo_calc TYPE REF TO lcl_calculator.
  CREATE OBJECT lo_calc.

  WRITE: / 'Add:',      lo_calc->add(      iv_a = 10 iv_b = 5 ).
  WRITE: / 'Subtract:', lo_calc->subtract( iv_a = 10 iv_b = 5 ).
  WRITE: / 'Multiply:', lo_calc->multiply( iv_a = 10 iv_b = 5 ).
```

Expected output:

```
Add:      15
Subtract: 5
Multiply: 50
```

CODE Q11. Write a report with selection screen and display data.

```
REPORT zcode_q11.

TYPES: BEGIN OF ty_output,
         matnr TYPE mara-matnr,
         mtart TYPE mara-mtart,
         mbrsh TYPE mara-mbrsh,
         ersda TYPE mara-ersda,
       END OF ty_output.

DATA: lt_output TYPE STANDARD TABLE OF ty_output,
      lo_salv   TYPE REF TO cl_salv_table.

SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME TITLE text-001.
  SELECT-OPTIONS: s_matnr FOR mara-matnr.
  PARAMETERS:     p_mtart TYPE mara-mtart.
SELECTION-SCREEN END OF BLOCK b1.

START-OF-SELECTION.
  SELECT matnr, mtart, mbrsh, ersda
    FROM mara
    INTO TABLE @lt_output
    WHERE matnr IN @s_matnr
    AND   mtart  = @p_mtart.

  IF lt_output IS INITIAL.
    MESSAGE 'No data found for the selection' TYPE 'I'.
    RETURN.
  ENDIF.

  TRY.
    cl_salv_table=>factory(
      IMPORTING r_salv_table = lo_salv
      CHANGING  t_table      = lt_output ).
    lo_salv->get_functions( )->set_all( abap_true ).
    lo_salv->display( ).
  CATCH cx_salv_msg INTO DATA(lo_ex).
    MESSAGE lo_ex->get_text( ) TYPE 'E'.
  ENDTRY.
```

CODE Q12. Write a program to demonstrate Hashed Table performance.

```
REPORT zcode_q12.

TYPES: BEGIN OF ty_data,
         id    TYPE i,
         value TYPE string,
       END OF ty_data.

" Hashed table — O(1) access by key
DATA: lt_hashed TYPE HASHED TABLE OF ty_data
        WITH UNIQUE KEY id,
      wa_data TYPE ty_data.

" Fill table
DO 1000 TIMES.
  wa_data-id    = sy-index.
  wa_data-value = |Value_{ sy-index }|.
  INSERT wa_data INTO TABLE lt_hashed.
ENDDO.

" Direct key access — no loop needed
READ TABLE lt_hashed INTO wa_data WITH TABLE KEY id = 500.
IF sy-subrc = 0.
  WRITE: / 'Found ID 500:', wa_data-value.
ENDIF.
```

CODE Q13. Write a program demonstrating inheritance.

```
REPORT zcode_q13.

CLASS lcl_animal DEFINITION.
  PUBLIC SECTION.
    DATA: mv_name TYPE string.
    METHODS: speak.
ENDCLASS.

CLASS lcl_animal IMPLEMENTATION.
  METHOD speak.
    WRITE: / mv_name, 'makes a sound'.
  ENDMETHOD.
ENDCLASS.

CLASS lcl_dog DEFINITION INHERITING FROM lcl_animal.
  PUBLIC SECTION.
    METHODS: speak REDEFINITION.
ENDCLASS.
```

```
CLASS lcl_dog IMPLEMENTATION.
  METHOD speak.
    WRITE: / mv_name, 'says: Woof!'.
  ENDMETHOD.
ENDCLASS.

CLASS lcl_cat DEFINITION INHERITING FROM lcl_animal.
  PUBLIC SECTION.
    METHODS: speak REDEFINITION.
ENDCLASS.

CLASS lcl_cat IMPLEMENTATION.
  METHOD speak.
    WRITE: / mv_name, 'says: Meow!'.
  ENDMETHOD.
ENDCLASS.

START-OF-SELECTION.
  DATA: lo_dog TYPE REF TO lcl_dog,
        lo_cat TYPE REF TO lcl_cat.

  CREATE OBJECT lo_dog. lo_dog->mv_name = 'Rex'.
  CREATE OBJECT lo_cat. lo_cat->mv_name = 'Whiskers'.

  lo_dog->speak( ).
  lo_cat->speak( ).
```

Expected output:

```
Rex says: Woof!
Whiskers says: Meow!
```

CODE Q14. Write a CDS View for Sales Order data.

```
@AbapCatalog.sqlViewName: 'ZSALES_CDS_V'
@AbapCatalog.compiler.compareFilter: true
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Sales Order CDS View'

define view ZSALES_ORDER_CDS
  as select from vbak as header
  inner join vbap as items
    on header.vbeln = items.vbeln
{
  key header.vbeln    as SalesOrder,
```

```
    header.erdat    as CreationDate,
    header.kunnr    as Customer,
    items.posnr     as ItemNumber,
    items.matnr     as Material,
    items.netwr     as NetValue,
    items.waerk     as Currency
}
where header.auart = 'OR'
```

CODE Q15. Write a program to find duplicate records in an internal table.

```
REPORT zcode_q15.

TYPES: BEGIN OF ty_data,
        matnr TYPE matnr,
        werks TYPE werks_d,
       END OF ty_data.

DATA: lt_data  TYPE STANDARD TABLE OF ty_data,
      lt_dupes TYPE STANDARD TABLE OF ty_data,
      wa_data  TYPE ty_data.

" Sample data with duplicates
wa_data-matnr = '100'. wa_data-werks = '1000'. APPEND wa_data TO lt_data.
wa_data-matnr = '200'. wa_data-werks = '1000'. APPEND wa_data TO lt_data.
wa_data-matnr = '100'. wa_data-werks = '1000'. APPEND wa_data TO lt_data.
wa_data-matnr = '300'. wa_data-werks = '2000'. APPEND wa_data TO lt_data.

SORT lt_data BY matnr werks.

" Find duplicates
LOOP AT lt_data INTO wa_data.
  AT END OF matnr.
    IF sy-tabix > 1.
      READ TABLE lt_data INTO wa_data INDEX sy-tabix.
      " Check previous row
    ENDIF.
  ENDAT.
ENDLOOP.

" Simpler duplicate detection
DATA: lt_unique TYPE STANDARD TABLE OF ty_data.
lt_unique = lt_data.
DELETE ADJACENT DUPLICATES FROM lt_unique COMPARING matnr werks.
```

-

```
WRITE: / 'Total rows:',  lines( lt_data ).
WRITE: / 'Unique rows:', lines( lt_unique ).
WRITE: / 'Duplicates:',  lines( lt_data ) - lines( lt_unique ).
```

Remaining Coding Questions — Quick Reference

#	Question	Key Concept
Q16	Reverse a string without built-in functions	LOOP + string manipulation
Q17	Check if a string is a palindrome	String comparison
Q18	Convert string to uppercase manually	ASCII manipulation
Q19	Count occurrences of a character in string	FIND ALL OCCURRENCES
Q20	Fibonacci series using DO loop	DO / ENDDO
Q21	Factorial using recursion in OO ABAP	Recursive method call
Q22	Bubble sort on internal table manually	Nested LOOP
Q23	Write a function module with exception	SE37, RAISE
Q24	Call a BAPI and handle return messages	BAPI_TRANSACTION_COMMIT
Q25	Write AMDP for data fetch	BY DATABASE PROCEDURE
Q26	Display top 5 customers by revenue using ALV	ORDER BY + UP TO
Q27	Update records using MODIFY	Open SQL MODIFY
Q28	Background program — write to application log	BAL_LOG_CREATE
Q29	Lock an object before update	ENQUEUE/DEQUEUE
Q30	New Open SQL with CASE and aggregation	S/4HANA SQL

SECTION 5 — Scenario-Based Questions

Accenture | SAP ABAP Developer | Fresher to 2 Years

-

> These questions test your **practical thinking and problem-solving ability**. Interviewers want to see HOW you think, not just what you know.

SCENARIO Q1. Your report is running very slowly in production. How do you troubleshoot it?

Frequency: ★★★★★ | Probability: 92%

Model Answer:

"My approach to troubleshoot a slow running report:

Step 1 — Identify the bottleneck using ST05: I would run ST05 (SQL Trace) while executing the report to capture all database calls. This shows me:

- Which SELECT statements are slow
- Whether full table scans are happening
- How many database calls are being made
- Total time per statement

Step 2 — Check for common performance antipatterns:

Problem Found	Fix
SELECT *	Replace with specific field selection
SELECT inside LOOP	Replace with FOR ALL ENTRIES
Missing WHERE clause	Add proper filter conditions
Full table scan	Check if proper index exists
Large data to app server	Push logic to DB using CDS/aggregation

Step 3 — Use SAT (Runtime Analysis): If the bottleneck is in ABAP processing (not DB), SAT shows time spent in each statement, helping identify expensive loops or redundant processing.

Step 4 — Check SM50/SM66: Verify if the issue is a work process bottleneck — if the system is overloaded, the report may be slow due to resource contention.

Step 5 — Implement and test fix: Apply the fix in DEV, use ST05 again to compare before/after performance, then transport to QAS for validation."

-

SCENARIO Q2. Your report is giving wrong output. How do you find the bug?

Frequency: ★★★★★ | Probability: 90%

Model Answer:

"My systematic approach to find a wrong output bug:

Step 1 — Understand the expected output: First clarify with the user exactly what correct output should look like for a specific set of inputs.

Step 2 — Reproduce with minimal input: Run the report with a small, specific input (e.g., a single sales order number) so I can trace the issue easily.

Step 3 — Verify data at each stage using debugger: Set breakpoints at:

- After each SELECT — check if correct records are fetched from DB (cross-check with SE16)
- After data merging — check if JOIN/FOR ALL ENTRIES result is correct
- Before ALV display — check final output table content

Step 4 — Verify against database directly: Use SE16/SE16N to directly query the relevant tables with the same keys and compare the result with what my program is fetching.

Step 5 — Common causes to check:

Common Bug	How to Find
Wrong WHERE condition	Compare with SE16 result
Missing INNER JOIN record	Check if record exists in both tables
Data type mismatch	MATNR needs leading zeros (ALPHA conversion)
Wrong field mapping	Check MOVE-CORRESPONDING field names
SY-SUBRC not checked	Data overwritten with previous value
FOR ALL ENTRIES removing duplicates	Expected duplicate removed

---

SCENARIO Q3. You get a short dump (runtime error) in production. What do you do?

-

Frequency: ★★★★★ | Probability: 88%

Model Answer:

"When a short dump occurs in production, my approach:

Step 1 — Analyze the dump in ST22: Transaction ST22 (ABAP Short Dump Analysis) shows:

- Error type (CX_SY_ZERODIVIDE, DBIF_RSQL_SQL_ERROR, etc.)
- Exact program, include, and line number
- Variable values at the time of dump
- Full call stack

Step 2 — Identify error type and cause:

Dump Type	Likely Cause
DBIF_RSQL_SQL_ERROR	Database error — wrong SQL syntax or table issue
DYNPRO_FIELD_CONVERSION	Type mismatch in field assignment
DATA_REFERENCE_NOT_ASSIGNED	Field symbol or reference not assigned
ITAB_LINE_NOT_FOUND	READ TABLE without TRANSPORTING NO FIELDS
CALL_FUNCTION_NOT_FOUND	Called FM doesn't exist in target system
TIME_OUT	Program running too long

Step 3 — Reproduce in DEV: Try to reproduce the exact scenario in DEV using the same inputs.

Step 4 — Fix in DEV, test, transport: Never fix directly in PRD. Fix in DEV → test → transport to QAS → validate → transport to PRD.

Step 5 — Temporary workaround if critical: If business is impacted and fix takes time, discuss with the team whether a temporary manual workaround is possible while the fix is prepared."

SCENARIO Q4. FOR ALL ENTRIES is fetching all records from the database. What happened?

Frequency: ★★★★★ | Probability: 90%

-

Model Answer:

"This is a classic and critical FOR ALL ENTRIES bug. The cause is:

Root Cause: The driver internal table (lt_vbak in our example) is EMPTY.

When the driver table is empty, FOR ALL ENTRIES ignores the WHERE condition and fetches ALL records from the target table — this can return millions of records and crash the system.

The Bug:

```
" If lt_vbak is empty, this fetches ALL of VBAP!
SELECT * FROM vbap
  INTO TABLE lt_vbap
  FOR ALL ENTRIES IN lt_vbak
  WHERE vbeln = lt_vbak-vbeln.
```

The Fix:

```
" Always check before FOR ALL ENTRIES
IF lt_vbak IS NOT INITIAL.
  SELECT * FROM vbap
    INTO TABLE lt_vbap
    FOR ALL ENTRIES IN lt_vbak
    WHERE vbeln = lt_vbak-vbeln.
ENDIF.
```

How to detect it:

- ST05 trace shows the SELECT without any WHERE condition being executed
- SY-DBCNT returns an unexpectedly huge number after the SELECT
- Memory usage spikes dramatically

This is one of the most important rules in ABAP performance — interviewers specifically ask about it."

SCENARIO Q5. A transport request was released but the objects are not in QAS. What do you check?

Frequency: ★★★★ | Probability: 82%

Model Answer:

"My troubleshooting checklist for missing transport in QAS:

-

Step 1 — Verify release in SE10: Check if the transport is truly released (status = Released/Exported). Releasing a task ≠ releasing the main request. Both must be released.

Step 2 — Check STMS import queue: In STMS → Overview → Imports → Select QAS system. Check if the transport appears in the import queue and what its status is.

Step 3 — Check transport logs in STMS: STMS → Transport Logs → look for error messages during import. Common errors:

- Object already exists with different ownership
- Syntax error in imported program
- Missing prerequisite transport

Step 4 — Check if all tasks are released: In SE10, expand the main request — all individual developer tasks must be released before the main request can be imported successfully.

Step 5 — Re-import if needed: If the transport is in the queue but not imported, ask Basis to trigger the import. If it failed, analyze the error log and fix the issue.

Step 6 — Check object locks: If another transport has locked the same object, the import may fail. Check STMS for conflicting transports."

SCENARIO Q6. A background job failed. How do you analyze and fix it?

Frequency: ★★★★ | Probability: 82%

Model Answer:

"When a background job fails, my analysis steps:

Step 1 — Check SM37: Transaction SM37 → check job status. A failed job shows status 'Cancelled'.

Step 2 — View job log: In SM37, select the job → Job Log. The log shows:

- Which step failed
- Error messages
- Short dump reference if applicable

Step 3 — Check spool output: If the program writes output, check the spool (SP01) for any error messages the program produced before failing.

Step 4 — Check ST22 for short dumps: If the job terminated with a short dump, ST22 will have the dump details.

-

Step 5 — Common background job failure reasons:

Reason	Fix
Program syntax error	Fix program, re-transport
Variant missing or changed	Recreate variant for the job
Authorization failure	Check SU53 for missing auth objects
Timeout	Optimize program performance
Database lock	Check SM12 for lock entries
Incorrect job parameters	Review SM36 job definition

Step 6 — Restart the job: After fixing the root cause, reschedule in SM36 or use SM37 to repeat the job."

SCENARIO Q7. Two users are trying to edit the same record simultaneously. How do you handle it?

Frequency: ★★★★ | Probability: 80%

Model Answer:

"This is a data consistency problem solved using SAP Lock Objects.

Solution — Implement Enqueue/Dequeue:

Step 1 — Create Lock Object in SE11: Create a lock object EZMATERIAL with lock argument MATNR (the key field). SAP generates two function modules automatically:

- ENQUEUE_EZMATERIAL — to lock
- DEQUEUE_EZMATERIAL — to unlock

Step 2 — Lock before edit:

```
CALL FUNCTION 'ENQUEUE_EZMATERIAL'
  EXPORTING
    matnr          = lv_matnr
    mode_mara      = 'E'  " Exclusive lock
  EXCEPTIONS
    foreign_lock   = 1    " Another user has the lock
    system_failure = 2.
```

-

```
IF sy-subrc = 1.
  MESSAGE 'This record is currently being edited by another user.
          Please try again later.' TYPE 'W'.
  RETURN.
ENDIF.
```

Step 3 — Perform edit and save

Step 4 — Release lock after save:

```
CALL FUNCTION 'DEQUEUE_EZMATERIAL'
  EXPORTING
    matnr = lv_matnr.
```

Important: Locks are automatically released when the user's session ends or rolls back, preventing permanent lock situations."

SCENARIO Q8. Your SELECT is returning duplicate records. How do you handle it?

Frequency: ★★★★ | Probability: 82%

Model Answer:

"Duplicate records can come from different sources. My approach:

Identify the source of duplicates:

Case 1 — Duplicates from JOIN: When joining tables with 1:N relationships, the left table record repeats for each matching right table record.

Fix: Select only required fields and use DISTINCT:

```
SELECT DISTINCT kunnr name1
  FROM kna1
  INTO TABLE @lt_kna1.
```

Case 2 — FOR ALL ENTRIES removes duplicates you need: FOR ALL ENTRIES automatically removes duplicates from result. If you need duplicates, use JOIN instead.

Case 3 — Duplicates in internal table processing:

" Sort first, then remove duplicates

-

```
SORT lt_data BY key_field1 key_field2.
DELETE ADJACENT DUPLICATES FROM lt_data
  COMPARING key_field1 key_field2.
```

Case 4 — Data quality issue in source table: Sometimes duplicates exist in the actual database. Analyze with SE16 and report to functional team.

Prevention: Always design your SELECT with specific key fields in WHERE clause and use DISTINCT when needed."

SCENARIO Q9. How do you optimize a SELECT that is doing a full table scan?

Frequency: ★★★★★ | Probability: 88%

Model Answer:

"A full table scan means the database is reading every row in the table instead of using an index. This is extremely slow for large tables.

How to detect: ST05 trace shows the SELECT with 'FULL' in the access type column, or execution time is very high.

Causes and fixes:

Cause	Fix
No WHERE clause	Add proper key field conditions
WHERE fields not indexed	Create secondary index in SE11
WHERE uses non-key fields	Use primary key fields first
Leading wildcard in LIKE	Avoid `WHERE field LIKE '%value'`
Type mismatch in WHERE	Ensure correct data types
Function on WHERE field	Avoid `WHERE UPPER(field) = 'X'`

Creating a secondary index: If a field is frequently used in WHERE but is not the primary key, create a secondary index in SE11 → Table → Indexes. This dramatically improves performance.

Example fix:

•

" BAD — full table scan on large table
SELECT * FROM vbap INTO TABLE lt_vbap
  WHERE matnr = lv_matnr. " MATNR is not the primary key

" BETTER — use primary key fields
SELECT * FROM vbap INTO TABLE lt_vbap
  WHERE vbeln = lv_vbeln   " Primary key
  AND   posnr = lv_posnr. " Primary key
```"

---

## SCENARIO Q10. A user says the report shows different numbers than what SE16 shows for the same table. What do you investigate?

**Frequency:** ★★★★ | **Probability: 78%**

**Model Answer:**

"This is a data discrepancy issue. My investigation steps:

**Step 1 — Verify the exact input:**
Check if the user is using the same filters in the report as the SE16 query. Often the date range, client, or plant filter is different.

**Step 2 — Check table buffering:**
If the table has buffering enabled, the report may show buffered (cached) data while SE16 shows current database data.
Check in SE11 → Table Properties → Buffering setting.
Solution: Add `%_HINTS ORACLE 'BYPASS_BUFFER'` or use `SELECT ... BYPASSING BUFFER`.

**Step 3 — Check for authorization filters:**
The report may have AUTHORITY-CHECK limiting which records a user sees. SE16 may not have the same restriction.

**Step 4 — Check FOR ALL ENTRIES duplicate removal:**
FOR ALL ENTRIES removes duplicate rows. If the user expects duplicates in output, this could explain fewer records.

**Step 5 — Check aggregate logic:**
If the report uses COLLECT or SUM, it aggregates records. SE16 shows raw records.

**Step 6 — Check WHERE clause carefully:**
Reproduce the exact SE16 query conditions in the ABAP SELECT and compare line by line."

---
```

-

SCENARIO Q11. How do you handle a situation where your BAPI call is not saving data?

Frequency: ★★★★ | **Probability: 80%**

Model Answer:

"The most common reason a BAPI is not saving data:

Reason 1 — Missing BAPI_TRANSACTION_COMMIT:
Every BAPI that writes data requires an explicit COMMIT. Without it, data is in a temporary LUW (Logical Unit of Work) and gets rolled back when the session ends.

```abap
" After successful BAPI call
CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
  EXPORTING wait = 'X'.  " WAIT = 'X' ensures sync commit
```

Reason 2 — Error in RETURN table not handled: The BAPI returned errors in the RETURN table but the code did not check and called COMMIT anyway — or the opposite, there were errors and no ROLLBACK was called.

```abap
" Correct pattern
CALL FUNCTION 'BAPI_SALESORDER_CREATEFROMDAT2'
  ...
  TABLES return = lt_return.

READ TABLE lt_return INTO wa_return WITH KEY type = 'E'.
IF sy-subrc = 0.
  " Error exists — rollback
  CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
  MESSAGE wa_return-message TYPE 'E'.
ELSE.
  " Success — commit
  CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
    EXPORTING wait = 'X'.
ENDIF.
```

Reason 3 — Authorization issue: The user running the BAPI may not have authorization for the business transaction. Check SU53 after the BAPI call."

SCENARIO Q12. How do you find which program last changed a record in a SAP table?

-

Frequency: ★★★ | Probability: 65%

Model Answer:

"To find who or what changed a record, I would use:

Option 1 — Change Documents (CDHDR/CDPOS): If change documents are configured for the table, CDHDR and CDPOS store all field-level changes with user, date, time, and old/new values.

- CDHDR — Change Document Header (object, user, date, time)
- CDPOS — Change Document Item (field, old value, new value)

Option 2 — Table Logging (SE13): If table logging is activated in SE13, use transaction SCU3 to view all changes to the table records.

Option 3 — Standard audit tables: For business objects like sales orders, use standard transactions:

- VA03 → Environment → Changes (for sales order changes)
- FB03 → Environment → Document Changes (for FI documents)

Option 4 — Application Log (SLG1): If the program writes to the application log, SLG1 shows historical execution logs.

Option 5 — ABAP program analysis: If you suspect a specific program, search for all programs that write to the table using SE80 → Where Used List on the table name."

SCENARIO Q13. Production has a performance issue during month-end. How do you approach it?

Frequency: ★★★★ | Probability: 78%

Model Answer:

"Month-end performance issues are common in SAP and usually involve multiple interacting factors:

Immediate Analysis (within hours):

SM50/SM66: Check active work processes — are they all busy? Is there a PRIV mode process blocking others?

SM12: Check for long-running database locks that are blocking other processes.

ST05: If possible, trace the slowest running programs during peak load.

-

DB02/DBA Cockpit: Check database health — tablespace usage, long-running DB queries, index health.

Common Month-End Causes:

Cause	Solution
Too many parallel batch jobs	Stagger job schedules in SM36
Programs not optimized	SELECT *, nested selects
Missing database indexes	Create secondary indexes in SE11
Table statistics outdated	Update DB statistics
Lack of parallel processing	Use parallel RFC or background jobs

Medium-term fix: Identify the top 5 slowest programs using ST05 and SM37 logs, optimize them one by one, and retest during next month-end.

Long-term: Consider scheduling heavy batch jobs during off-peak hours (night) and splitting large jobs into smaller parallel jobs."

SCENARIO Q14. How would you handle a requirement to read a large file uploaded by a user?

Frequency: ★★★ | Probability: 65%

Model Answer:

"For uploading and processing large files in ABAP:

Step 1 — Use GUI_UPLOAD for smaller files (dialog program):

```
DATA: lt_file TYPE TABLE OF string.

CALL FUNCTION 'GUI_UPLOAD'
  EXPORTING
    filename = 'C:\input.txt'
    filetype = 'ASC'
  TABLES
    data_tab = lt_file
  EXCEPTIONS
    file_open_error = 1.
```

-

Step 2 — For CSV files, parse each line:

```
LOOP AT lt_file INTO lv_line.
  SPLIT lv_line AT ',' INTO wa_data-field1
                            wa_data-field2
                            wa_data-field3.
  APPEND wa_data TO lt_data.
ENDLOOP.
```

Step 3 — For very large files — use application server path: Use AL11 to place the file on the application server, then read using OPEN DATASET:

```
OPEN DATASET lv_filepath FOR INPUT IN TEXT MODE.
DO.
  READ DATASET lv_filepath INTO lv_line.
  IF sy-subrc <> 0. EXIT. ENDIF.
  " Process lv_line
ENDDO.
CLOSE DATASET lv_filepath.
```

Step 4 — Validate data: After reading, validate each row for mandatory fields, correct data types, and duplicates before processing further."

SCENARIO Q15. How do you handle a scenario where you need to send an email from ABAP?

Frequency: ★★★ | Probability: 65%

Model Answer:

"Sending email from ABAP is done using the Business Communication Service (BCS):

```
DATA: lo_send_req TYPE REF TO cl_bcs,
      lo_document TYPE REF TO cl_document_bcs,
      lo_recipient TYPE REF TO if_recipient_bcs.

TRY.
  " Create send request
  lo_send_req = cl_bcs=>create_persistent( ).

  " Create email body
  DATA: lt_body TYPE soli_tab,
        wa_body TYPE soli.
  wa_body-line = 'Hello, this is an automated email from SAP.'.
```

•

```
  APPEND wa_body TO lt_body.

  lo_document = cl_document_bcs=>create_document(
    i_type    = 'RAW'
    i_subject = 'SAP Automated Notification'
    i_text    = lt_body ).

  lo_send_req->set_document( lo_document ).

  " Add recipient
  lo_recipient = cl_cam_address_bcs=>create_internet_address(
    'recipient@email.com' ).
  lo_send_req->add_recipient( lo_recipient ).

  " Send
  lo_send_req->send( ).
  COMMIT WORK.

CATCH cx_bcs INTO DATA(lo_ex).
  MESSAGE lo_ex->get_text( ) TYPE 'E'.
ENDTRY.
```

SCENARIO Q16. A user reports that data entered in a screen is not getting saved. What do you check?

Frequency: ★★★★ | **Probability: 78%**

Model Answer:

"For a Module Pool screen where data is not saving:

Step 1 — Check PAI module:
Verify the SAVE/ENTER command in PAI module is correctly coded and being triggered.

```abap
MODULE user_command_0100 INPUT.
  CASE sy-ucomm.
    WHEN 'SAVE'.
      PERFORM save_data.  " Is this being called?
  ENDCASE.
ENDMODULE.
```

Step 2 — Check COMMIT WORK: Verify `COMMIT WORK` is called after the database INSERT/UPDATE. Without it, data stays in buffer and is lost.

-

Step 3 — Check SY-SUBRC after INSERT/UPDATE: Verify the database operation succeeded:

```
INSERT ztable FROM wa_data.
IF sy-subrc <> 0.
  MESSAGE 'Save failed!' TYPE 'E'.
ENDIF.
COMMIT WORK.
```

Step 4 — Check lock objects: Is the record locked by another user? ENQUEUE might be failing silently.

Step 5 — Check authorization: User may not have write authorization for the relevant object. Check SU53.

Step 6 — Check message type: If an ERROR message (type E) is being triggered before the save logic, PAI processing stops and save never executes."

SCENARIO Q17. How do you test your ABAP program before transporting to production?

Frequency: ★★★★ | Probability: 80%

Model Answer:

"My complete testing approach before transport to production:

Level 1 — Developer Unit Testing in DEV:

- Test all selection screen input combinations
- Test boundary conditions (empty input, maximum values, special characters)
- Test error scenarios (invalid data, missing records)
- Verify SY-SUBRC handling at every step
- Check performance with realistic data volume using ST05

Level 2 — Peer Code Review: Share code with a senior developer for review covering:

- Logic correctness
- Performance antipatterns
- Missing error handling
- Hardcoded values that should be configurable

Level 3 — QAS Testing after transport:

- Functional testing by business user or functional consultant
- Test with production-like data volume

-

- Verify authorization (correct users can/cannot access)
- Test in correct client and language settings

Level 4 — Performance validation:

- Run ST05 in QAS with realistic data
- Verify runtime is acceptable for business users
- Check SM37 for any job execution issues (for batch programs)

Level 5 — Sign-off: Get formal sign-off from business user and functional consultant before PRD transport."

SCENARIO Q18. How do you identify which index a SELECT is using?

Frequency: ★★★ | Probability: 65%

Model Answer:

"To identify index usage:

Method 1 — ST05 SQL Trace: The SQL trace shows the execution plan including which index is being used. Look for:

- Index name in the trace output
- 'FULL' access type = no index = full table scan

Method 2 — SE11 → Table → Technical Settings: View the primary key and all secondary indexes defined for the table.

Method 3 — EXPLAIN PLAN (via DB02): In DB02 (Database performance monitor), you can run EXPLAIN PLAN on a SQL statement to see the execution plan including index usage.

Creating a secondary index when missing:

SE11 → Enter table name → Display
→ Extras → Index → Create
→ Add the fields used in WHERE clause
→ Activate

Note: Do not create too many indexes as they slow down INSERT/UPDATE/DELETE operations. Only create indexes for frequently used WHERE conditions on large tables."

SCENARIO Q19. How would you migrate data from a legacy system to SAP?

Frequency: ★★★ | Probability: 65%

Model Answer:

"Data migration from legacy to SAP involves:

Common ABAP tools for data migration:

Tool	When to use
LSMW (Legacy System Migration Workbench)	Simple field mappings, configuration data
BDC (Batch Data Communication)	Simulate screen transactions, complex validations
BAPI	Recommended for master/transaction data with validation
Direct INSERT	Only for custom Z-tables, never for standard SAP tables
SAP Data Services	Large volume, complex transformation

My approach for a migration task:

Step 1: Understand source data structure and target SAP structure Step 2: Identify the correct BAPI or BDC approach Step 3: Create a mapping document (source field → SAP field) Step 4: Write ABAP program to read source file and call BAPI Step 5: Run in test mode first (simulation without save) Step 6: Fix errors from test run Step 7: Run actual migration with COMMIT Step 8: Validate migrated data in SAP using SE16 and reports

Using BAPI for migration (recommended):

```
LOOP AT lt_source INTO wa_source.
  " Map fields
  ls_header-matnr = wa_source-material_id.

  " Call BAPI
  CALL FUNCTION 'BAPI_MATERIAL_SAVEDATA'
    EXPORTING headdata = ls_header
    TABLES    return   = lt_return.

  " Check and commit
  READ TABLE lt_return WITH KEY type = 'E'
    INTO wa_return.
```

-

```
  IF sy-subrc <> 0.
    CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
      EXPORTING wait = 'X'.
  ELSE.
    CALL FUNCTION 'BAPI_TRANSACTION_ROLLBACK'.
    " Log error for this record
  ENDIF.
ENDLOOP.
```

SCENARIO Q20. How do you handle a requirement where output needs to be in Excel format?

Frequency: ★★★★ | **Probability: 78%**

Model Answer:

"There are multiple ways to generate Excel output from ABAP:

Method 1 — ALV Export (simplest — recommended for reports):
When using CL_SALV_TABLE and enabling all standard functions, the user gets a built-in 'Export to Spreadsheet' button. No extra coding needed.

```abap
lo_funcs->set_all( abap_true ).  " Includes Excel export
```

Method 2 — GUI_DOWNLOAD for simple CSV/Excel:

```
CALL FUNCTION 'GUI_DOWNLOAD'
  EXPORTING
    filename = 'C:\output.xls'
    filetype = 'DAT'
  TABLES
    data_tab = lt_output.
```

Method 3 — OLE Automation (for formatted Excel): Use ABAP OLE to create a real Excel file with formatting, formulas, and charts. More complex but gives full Excel control.

Method 4 — XLSX Workbench or third-party tools: For complex formatted Excel with multiple sheets, logos, and charts, specialized tools are available.

For Accenture interviews: Mentioning Method 1 (ALV standard export) and Method 2 (GUI_DOWNLOAD) is usually sufficient. If asked about formatted Excel, mention OLE Automation."

•

Quick Reference — Scenario Problem-Solving Framework

Scenario Type	First Tool	Second Tool	Fix Location
Performance issue	ST05	SAT	DEV → Transport
Short dump	ST22	Debugger	DEV → Transport
Wrong output	Debugger	SE16 comparison	DEV → Transport
Transport missing	SE10	STMS	Basis team
Background job failed	SM37	Job log / ST22	DEV → Transport
Data not saving	Debugger	SM12 (locks)	DEV → Transport
Full table scan	ST05	SE11 (index)	Secondary index
Duplicate records	SE16	SORT + DELETE	Program logic

SECTION — One Day Revision Sheet

SAP ABAP | Accenture | Fresher to 2 Years

PART 1 — Top 100 One-Line Definitions

#	Term	One-Line Definition
1	ABAP	SAP's proprietary programming language — Advanced Business Application Programming
2	Transparent Table	Database table with 1:1 mapping to physical DB table
3	Pooled Table	Multiple logical tables stored in one physical DB pool table
4	Cluster Table	Multiple logical tables stored in one physical cluster table
5	Data Dictionary	Central repository for all data definitions in SAP (SE11)

6	Domain	Defines technical attributes of a field — data type, length, fixed values
7	Data Element	Defines business meaning of a field — label, documentation, search help
8	Structure	Flat data object with no physical DB table
9	Standard Table	Internal table allowing duplicates, accessed by index or key
10	Sorted Table	Internal table sorted by key, no duplicates if unique key defined
11	Hashed Table	Internal table with unique key, O(1) access via hash algorithm
12	Work Area	Single row structure used to process data from internal table
13	Header Line	Obsolete implicit work area built into internal table — avoid using
14	Field Symbol	ABAP pointer that references existing memory without copying data
15	SELECT SINGLE	Fetches exactly one record from database — use when key is unique
16	FOR ALL ENTRIES	Fetches DB records based on values in a driver internal table
17	INNER JOIN	Returns only rows that match in both tables
18	LEFT OUTER JOIN	Returns all rows from left table plus matching rows from right
19	BINARY SEARCH	O(log n) search on sorted internal table — must sort first
20	Function Module	Globally reusable, RFC-capable ABAP subroutine managed in SE37
21	Function Group	Container that holds related function modules
22	BAPI	RFC-enabled standardized function module for SAP business processes
23	RFC	Remote Function Call — communication between SAP systems or external apps

24	sRFC	Synchronous RFC — waits for result before continuing
25	aRFC	Asynchronous RFC — does not wait for result
26	tRFC	Transactional RFC — guarantees exactly-once execution
27	qRFC	Queued RFC — tRFC with defined processing sequence
28	User Exit	FORM routine in standard SAP program for customer enhancement
29	Customer Exit	Function module provided by SAP for customer-specific enhancement
30	BADI	Business Add-In — OOP-based enhancement technique (SE18/SE19)
31	Enhancement Spot	Modern enhancement framework allowing code insertion anywhere
32	ALV	ABAP List Viewer — interactive grid/list display with sort/filter/export
33	CL_SALV_TABLE	OOP class for creating ALV reports — recommended in S/4HANA
34	Smart Forms	SAP form tool for creating business forms (transaction SMARTFORMS)
35	Adobe Forms	PDF-based interactive form tool preferred in S/4HANA (transaction SFP)
36	Module Pool	Screen-based ABAP program type for creating custom SAP transactions
37	PBO	Process Before Output — screen event triggered before screen displays
38	PAI	Process After Input — screen event triggered after user action
39	CDS View	Core Data Services — database-level view for code pushdown in HANA
40	AMDP	ABAP Managed Database Procedure — HANA SQLScript inside ABAP class

41	Code Pushdown	Moving data processing logic from app server to HANA database layer
42	New Open SQL	Modern ABAP SQL with inline declarations, CASE, string functions
43	SY-SUBRC	System return code — 0 means success, non-zero means failure
44	SY-TABIX	Current row index during LOOP AT internal table
45	SY-DBCNT	Number of rows affected by last database operation
46	SY-UNAME	Current logged-in SAP username
47	SY-DATUM	Current system date in YYYYMMDD format
48	SY-UZEIT	Current system time in HHMMSS format
49	SY-MANDT	Current SAP client number
50	SY-REPID	Name of currently executing ABAP program
51	Transport Request	Container for moving SAP objects between DEV, QAS, PRD systems
52	Workbench Request	Transport for repository objects like programs and classes
53	Customizing Request	Transport for configuration/IMG settings
54	SE10	Transaction to create and manage transport requests
55	STMS	SAP Transport Management System — used to import transports
56	Lock Object	DD object that generates ENQUEUE/DEQUEUE FMs for data locking
57	ENQUEUE	Function module to lock a record and prevent concurrent access
58	DEQUEUE	Function module to release a previously set lock
59	IDoc	Intermediate Document — SAP's standard format for data exchange
60	Control Record	IDoc header containing sender, receiver, and message type info
61	Data Record	IDoc segment containing actual business data
62	Status Record	IDoc component recording processing status history

63	Message Type	IDoc business process identifier e.g. ORDERS, INVOIC, MATMAS
64	Partner Profile	WE20 configuration defining IDoc communication with trading partners
65	MODIFY	Open SQL statement that inserts if not exists or updates if exists
66	UPDATE	Open SQL statement that updates only existing records
67	DELETE ADJACENT DUPLICATES	Removes consecutive duplicate rows — must sort first
68	COLLECT	Appends new row or accumulates numeric fields if key already exists
69	APPEND	Always adds a new row to the end of an internal table
70	MOVE-CORRESPONDING	Copies values between structures matching by field name
71	TRY-CATCH	ABAP class-based exception handling block
72	CX_ROOT	Root exception class — catches all class-based exceptions
73	ST05	Performance trace transaction for SQL query analysis
74	SAT	ABAP runtime analysis tool — replaces SE30
75	ST22	Short dump analysis transaction — shows runtime error details
76	SM36	Transaction to schedule background jobs
77	SM37	Transaction to monitor and manage background jobs
78	SM50	Work process monitor — shows active processes on current server
79	SM59	Transaction to configure RFC destinations
80	SM12	Lock entry monitor — view and delete stuck lock entries
81	SU01	User maintenance transaction
82	SU53	Authorization check — shows missing auth objects for a user

83	SE16/SE16N	Table browser — view database table contents directly
84	SE38	ABAP Editor — write and test ABAP programs
85	SE37	Function Module editor and tester
86	SE80	Object Navigator — central ABAP workbench tool
87	SE18	BADI definition viewer
88	SE19	BADI implementation creator
89	SE24	Class Builder — create and maintain ABAP classes
90	WE02	IDoc display transaction
91	BD87	IDoc reprocessing transaction for failed IDocs
92	VBAK	Sales Order Header table
93	VBAP	Sales Order Items table
94	KNA1	Customer Master General Data table
95	MARA	Material Master General Data table
96	EKKO	Purchase Order Header table
97	EKPO	Purchase Order Items table
98	BKPF	Accounting Document Header table
99	BSEG	Accounting Document Segment — cluster table
100	ACDOCA	Universal Journal in S/4HANA — replaces multiple FI tables

---

PART 2 — Frequently Asked Syntax

#	What	Syntax
1	Declare variable	`DATA: lv_name TYPE string.`
2	Declare internal table	`DATA: lt_mara TYPE STANDARD TABLE OF mara.`
3	Declare work area	`DATA: wa_mara TYPE mara.`
4	Declare field symbol	`FIELD-SYMBOLS: <fs_mara> TYPE mara.`

5	Select single	`SELECT SINGLE * FROM mara INTO wa_mara WHERE matnr = lv_matnr.`
6	Select into table	`SELECT * FROM mara INTO TABLE lt_mara WHERE mtart = 'FERT'.`
7	New Open SQL	`SELECT matnr FROM mara INTO TABLE @DATA(lt_m) WHERE mtart = 'FERT'.`
8	FOR ALL ENTRIES	`SELECT * FROM vbap INTO TABLE lt_vbap FOR ALL ENTRIES IN lt_vbak WHERE vbeln = lt_vbak-vbeln.`
9	INNER JOIN	`SELECT a~vbeln b~matnr FROM vbak AS a INNER JOIN vbap AS b ON a~vbeln = b~vbeln INTO TABLE @lt_data.`
10	LOOP with work area	`LOOP AT lt_mara INTO wa_mara. ENDLOOP.`
11	LOOP with field symbol	`LOOP AT lt_mara ASSIGNING <fs_mara>. ENDLOOP.`
12	LOOP with WHERE	`LOOP AT lt_mara INTO wa_mara WHERE mtart = 'FERT'. ENDLOOP.`
13	READ TABLE by key	`READ TABLE lt_mara INTO wa_mara WITH KEY matnr = lv_matnr.`
14	READ TABLE binary search	`SORT lt_mara BY matnr. READ TABLE lt_mara INTO wa_mara WITH KEY matnr = lv_matnr BINARY SEARCH.`
15	SORT	`SORT lt_mara BY matnr ASCENDING mtart DESCENDING.`
16	DELETE ADJACENT DUPLICATES	`SORT lt_data BY matnr. DELETE ADJACENT DUPLICATES FROM lt_data COMPARING matnr.`
17	APPEND	`APPEND wa_mara TO lt_mara.`
18	INSERT into table	`INSERT wa_mara INTO lt_mara INDEX 1.`
19	MODIFY internal table	`MODIFY lt_mara FROM wa_mara INDEX sy-tabix.`

20	DELETE from internal table	`DELETE lt_mara WHERE mtart = 'ROH'.`
21	Check SY-SUBRC	`IF sy-subrc = 0. ENDIF.`
22	IF IS INITIAL	`IF lt_vbak IS NOT INITIAL.`
23	TRY-CATCH	`TRY. CATCH cx_root INTO DATA(lo_ex). ENDTRY.`
24	Message	`MESSAGE 'Text' TYPE 'S'.`
25	CALL FUNCTION	`CALL FUNCTION 'FM_NAME' EXPORTING iv_p = lv_v IMPORTING ev_r = lv_r EXCEPTIONS error = 1.`
26	BAPI commit	`CALL FUNCTION 'BAPI_TRANSACTION_COMMIT' EXPORTING wait = 'X'.`
27	Create object	`DATA: lo_obj TYPE REF TO lcl_class. CREATE OBJECT lo_obj.`
28	Call method	`lo_obj->method_name( ).`
29	SALV ALV	`cl_salv_table=>factory( IMPORTING r_salv_table = lo_salv CHANGING t_table = lt_data ). lo_salv->display( ).`
30	String template	`lv_result =

PART 3 — Important T-Codes

Category	T-Code	Purpose
Development	SE38	ABAP Editor
	SE37	Function Module
	SE11	Data Dictionary
	SE16/SE16N	Table Browser

-

	SE80	Object Navigator
	SE24	Class Builder
	SE51	Screen Painter
	SE91	Message Maintenance
Enhancement	SE18	BADI Definition
	SE19	BADI Implementation
Transport	SE10	Transport Requests
	STMS	Transport Management
	SE09	Workbench Requests
Performance	ST05	SQL Trace
	SAT	Runtime Analysis
	SM50	Work Process Monitor
	SM66	Global Work Processes
	DB02	Database Monitor
Background Jobs	SM36	Schedule Job
	SM37	Monitor Jobs
Debugging	ST22	Short Dump Analysis
	SM12	Lock Entries
	SM59	RFC Destinations
Forms	SMARTFORMS	Smart Forms
	SFP	Adobe Forms
IDocs	WE02/WE05	Display IDocs
	WE19	Test IDocs
	WE20	Partner Profiles
	BD87	Reprocess IDocs
User/Auth	SU01	User Maintenance

-

SU53	Auth Check Trace
PFCG	Role Maintenance

PART 4 — Frequently Confused Concepts

Confusion	Concept A	Concept B
MODIFY vs UPDATE	INSERT + UPDATE combined	Only updates existing records
TYPE vs LIKE	References a data type	References an existing object
Work area vs Header line	Explicit — recommended	Implicit — obsolete
JOIN vs FOR ALL ENTRIES	DB level — better for small data	App server — better for large data
BAPI vs RFC	Standardized business FM	Any remote-callable FM
User Exit vs BADI	Procedural FORM routine	OOP interface method
SELECT SINGLE vs SELECT UP TO 1 ROWS	Returns 1 row by key	Returns first row of any result
SORT vs ORDER BY	Sorts internal table in ABAP	Sorts result at DB level
APPEND vs COLLECT	Always adds new row	Aggregates numeric fields
CLEAR vs REFRESH vs FREE	Clears work area or header	Clears table body
Standard vs Sorted vs Hashed	Duplicates, index access	Sorted key, binary
Domain vs Data Element	Technical properties	Business meaning
Transparent vs Cluster	Direct DB mapping	Compressed storage
PBO vs PAI	Before screen displays	After user input
sRFC vs aRFC vs tRFC	Synchronous	Asynchronous
SE18 vs SE19	BADI definition	BADI implementation

•

SE10 vs STMS	Create/manage requests	Import/manage transports
SM36 vs SM37	Schedule jobs	Monitor jobs
ST05 vs SAT	SQL trace	ABAP runtime trace

PART 5 — Memory Tricks

Concept	Memory Trick
SY-SUBRC = 0 means success	**S**ubRC = **S**uccess = **0** — Zero is good
FOR ALL ENTRIES driver table check	**F**or **A**ll **E**ntries = **F**irst **A**lways **E**nsure not initial
BAPI needs COMMIT	**B**API = **B**ring **A** **C**ommit — never forget
SORT before BINARY SEARCH	**S**ort **B**efore **B**inary — SBB
SORT before DELETE ADJACENT DUPLICATES	**S**ort **D**uplicates **A**way — SDA
PBO vs PAI	**PBO** = **P**repare **B**efore **O**utput, **PAI** = **P**rocess **A**fter **I**nput
ENQUEUE to lock, DEQUEUE to unlock	**EN**ter = lock in, **DE**part = release
DEV → QAS → PRD	**D**evelop, **Q**uality check, **P**roduce
SE11 = Dictionary, SE38 = Editor, SE37 = Function	**11** fingers in dictionary, **38** = report (3+8=11 letters), **37** = function
Field symbol is a pointer	Field symbol = **F**inger pointing at data
Hashed table = O(1) access	Hash = **H**ighway — direct route, no searching
ACDOCA = Universal Journal	**A**ll **C**ompany **D**ata **O**ne **C**entral **A**rchive

---

PART 6 — Common Interview Keywords to Use

Technical Depth Keywords: FOR ALL ENTRIES, IS NOT INITIAL, BINARY SEARCH, Field Symbols, Code Pushdown, CDS View, AMDP, New Open SQL, CL_SALV_TABLE, TRY-CATCH, BAPI_TRANSACTION_COMMIT, SE10 transport

-

Performance Keywords: ST05, full table scan, secondary index, SELECT specific fields, avoid SELECT inside LOOP, SAT runtime analysis, application server vs database level

S/4HANA Keywords: In-memory computing, column store, ACDOCA, Universal Journal, SAP Fiori, CDS Views, AMDP, code pushdown, simplified data model, New Open SQL

OOP Keywords: Encapsulation, inheritance, polymorphism, abstraction, CREATE OBJECT, REF TO, PUBLIC/PRIVATE/PROTECTED, interface, redefinition

Enhancement Keywords: Modification-free, User Exit, BADI, Enhancement Spot, SE18, SE19, implicit enhancement, explicit enhancement

---

SECTION — Crack the Interview Guide

Accenture | SAP ABAP | Fresher to 2 Years

🔥 PART 1 — Top 15 Questions You MUST Answer Perfectly

#	Question	Why It Is Critical
1	Tell me about yourself	First impression — sets the tone for entire interview
2	What is the difference between Transparent, Pooled, and Cluster tables?	Asked in almost every ABAP interview
3	What are internal tables and their types?	Core ABAP concept — always asked
4	What is FOR ALL ENTRIES and what happens if driver table is empty?	Classic performance trap question
5	What is the difference between Field Symbol and Work Area?	Separates good candidates from average
6	How do you optimize a slow ABAP program?	Tests real-world thinking
7	What is a BADI? How do you implement it?	Enhancement — always asked

8	What is CDS View? Why is it important in S/4HANA?	S/4HANA must-know topic
9	What is code pushdown?	Fundamental S/4HANA concept
10	Explain your project end-to-end	Tests ownership and depth
11	What is a transport request? How did you use it?	Every developer must know this
12	What is BAPI? How do you call it and what is mandatory after calling it?	BAPI_TRANSACTION_COMMIT trap
13	What is the difference between JOIN and FOR ALL ENTRIES?	Performance thinking question
14	What tools do you use for performance analysis?	ST05, SAT — shows practical exposure
15	Why SAP ABAP? Why Accenture?	HR + motivation check

🔥 PART 2 — Top 20 Mistakes Freshers Make

#	Mistake	What to Do Instead
1	Not checking driver table before FOR ALL ENTRIES	Always add IF lt_table IS NOT INITIAL
2	Forgetting BAPI_TRANSACTION_COMMIT after BAPI	Always commit after successful BAPI write
3	Using SELECT * instead of specific fields	Always select only required fields
4	Writing SELECT inside a LOOP	Use FOR ALL ENTRIES or JOIN instead
5	Not checking SY-SUBRC after SELECT or READ TABLE	Always check return code
6	Using header lines in new code	Use explicit work areas always
7	Using BINARY SEARCH without sorting first	Always SORT before BINARY SEARCH

8	Saying they know everything	Be honest about knowledge gaps — show learning mindset
9	Not knowing their own project deeply	Prepare every aspect of your project thoroughly
10	Saying BAPI and RFC are the same thing	BAPI is a type of RFC — not the same
11	Confusing SE18 and SE19	SE18 = definition, SE19 = implementation
12	Not knowing basic T-Codes	Memorize at least 20 key T-Codes
13	Using MODIFY on standard SAP objects	Always use enhancements — never modify standard
14	Forgetting SORT before DELETE ADJACENT DUPLICATES	It only removes consecutive duplicates
15	Saying pooled/cluster tables support JOINs	They do not — only transparent tables support JOINs
16	Not mentioning IS ASSIGNED check for field symbols	Always verify field symbol is assigned before use
17	Confusing CLEAR / REFRESH / FREE	CLEAR = work area, REFRESH = table body, FREE = memory release
18	Weak answer to "Tell me about yourself"	Prepare and practice a 90-second structured answer
19	Saying "I don't know" and stopping	Say what you know, what you think, and that you are willing to learn
20	Not asking questions at the end	Always ask 1-2 smart questions to show genuine interest

---

🔥 PART 3 — Accenture Interviewer Expectations

Expectation	What They Look For
Communication	Clear, structured, confident English — not perfect but coherent

Technical Basics	Strong fundamentals — internal tables, SELECT, performance, enhancements
S/4HANA Awareness	CDS Views, New Open SQL, code pushdown — even basic knowledge valued
Problem-Solving	Logical approach when given a scenario — show thought process
Project Ownership	Speak about your project confidently — know every line you wrote
Honesty	If you don't know something, say so and explain what you would do to find out
Attitude	Eagerness to learn, team orientation, positive energy
Accenture Knowledge	Know what Accenture does, its SAP practice, and why you want to join
No Mugging	They can tell if you memorized answers — understand concepts deeply
Follow-up Readiness	Every answer will have a follow-up — be prepared to go deeper

---

🔥 PART 4 — Interview Flow

STAGE 1 — INTRODUCTION (5 minutes)

→ Interviewer introduces themselves
→ "Tell me about yourself"
→ Quick background check — college, CGPA, training
→ First impression is set here

STAGE 2 — TECHNICAL BASICS (15-20 minutes)

→ Data Dictionary questions
→ Internal tables — types, operations
→ SELECT queries — FOR ALL ENTRIES, JOIN
→ Field symbols vs work area
→ Performance — ST05, optimization rules
→ Enhancements — BADI, User Exit
→ OOP ABAP basics

STAGE 3 — S/4HANA TOPICS (5-10 minutes)

→ What is CDS View?
→ What is code pushdown?

•

→ New Open SQL syntax
→ AMDP basics
→ Difference between ECC and S/4HANA

STAGE 4 — PROJECT DISCUSSION (10-15 minutes)

→ Explain your project
→ Which tables did you use?
→ What challenges did you face?
→ How did you optimize?
→ Transport requests
→ Debugging approach
→ What would you do differently?

STAGE 5 — SCENARIO QUESTIONS (5-10 minutes)

→ Performance issue scenario
→ Short dump scenario
→ Wrong output scenario
→ FOR ALL ENTRIES trap question

STAGE 6 — HR QUESTIONS (5-10 minutes)

→ Why Accenture?
→ Strengths and weaknesses
→ Relocation and shift comfort
→ Salary expectation
→ Do you have questions for us?

.