C_ABAPD_2309 Q&As

SAP Certified Associate - Back-End Developer - ABAP Cloud

Pass SAP C_ABAPD_2309 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.certbus.com/c_abapd_2309.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by SAP Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

Which of the following is a generic internal table type?

A. SORTED TABLE

B. INDEX TABLE

C. STANDARD TABLE

D. HASHED TABLE

Correct Answer: B

A generic internal table type is a table type that does not define all the attributes of an internal table in the ABAP Dictionary; it leaves some of these attributes undefined. A table type is generic in the following cases1:

You have selected Index Table or Not Specified as the access type. You have not specified a table key or specified an incomplete table key.

You have specified a generic secondary table key.

A generic table type can be used only for typing formal parameters or field symbols. A generic table type cannot be used for defining data objects or constants2. Therefore, the correct answer is B. INDEX TABLE, which is a generic table type

that does not specify the access type or the table key. The other options are not generic table types, because:

A. SORTED TABLE is a table type that specifies the access type as sorted and the table key as a unique or non-unique primary key3.

C. STANDARD TABLE is a table type that specifies the access type as standard and the table key as a non-unique standard key that consists of all the fields of the table row in the order in which they are defined4. D. HASHED TABLE is a table type that specifies the access type as hashed and the table key as a unique primary key5. References: 1: Generic Table Types - ABAP Dictionary - SAP Online Help 2: Generic ABAP Types - ABAP Keyword Documentation - SAP Online Help 3: Sorted Tables - ABAP Keyword Documentation - SAP Online Help 4: Standard Tables

- ABAP Keyword Documentation - SAP Online Help 5: Hashed Tables - ABAP Keyword Documentation - SAP Online Help

QUESTION 2

For what kind of applications would you consider using on-stack developer extensions? Note: There are 2 correct answers to this question.

A. Applications that provide APIs for side by side SAP BTP apps

B. Applications that access SAP S/4HANA data using complex SQL

C. Applications that integrate data from several different systems

D. Applications that run separate from SAP S/4HANA

Correct Answer: AB

On-stack developer extensibility is a type of extensibility that allows you to create development projects directly on the SAP S/4HANA Cloud technology stack. It gives you the opportunity to develop cloud-ready and upgrade-stable custom ABAP applications and services inside the SAP S/4HANA Cloud, public edition system. You can use the ABAP Development Tools in Eclipse to create and deploy your on-stack extensions. On-stack developer extensibility is suitable for the following kinds of applications: Applications that provide APIs for side by side SAP BTP apps. On-stack developer extensibility allows you to create OData services or RESTful APIs based on CDS view entities or projection views. These services or APIs can expose SAP S/4HANA data and logic to other applications that run on the SAP Business Technology Platform (SAP BTP) or other platforms. This way, you can create a loosely coupled integration between your SAP S/4HANA system and your side by side SAP BTP apps. Applications that access SAP S/4HANA data using complex SQL. On-stack developer extensibility allows you to use ABAP SQL to access SAP S/4HANA data using complex queries, such as joins, aggregations, filters, parameters, and code pushdown techniques. You can also use ABAP SQL to perform data manipulation operations, such as insert, update, delete, and upsert. This way, you can create applications that require advanced data processing and analysis on SAP S/4HANA data. The other kinds of applications are not suitable for on-stack developer extensibility, as they have different requirements and challenges. These kinds of applications are: Applications that integrate data from several different systems. On-stack developer extensibility is not meant for creating applications that integrate data from multiple sources, such as other SAP systems, third-party systems, or cloud services. This is because on-stack developer extensibility does not support remote access or data replication, and it may cause performance or security issues. For this kind of applications, you should use side by side extensibility, which allows you to create applications that run on the SAP BTP and communicate with the SAP S/4HANA system via public APIs or events. Applications that run separate from SAP S/4HANA. On-stack developer extensibility is not meant for creating applications that run independently from the SAP S/4HANA system, such as standalone apps, microservices, or web apps. This is because on-stack developer extensibility requires a tight coupling with the SAP S/4HANA system, and it may limit the scalability, flexibility, and portability of the applications. For this kind of applications, you should use side by side extensibility, which allows you to create applications that run on the SAP BTP and leverage the cloud-native features and services of the platform. References: Developer Extensibility in SAP S/4HANA Cloud ABAP Environment, SAP S/4HANA Extensibility ?Simplified Guide for Beginners

QUESTION 3

Which of the following integration frameworks have been released for ABAP cloud development? Note: There are 3 correct answers to this question.

A. SOAP consumption

B. CDS Views

C. Business Add-ins (BAdls)

D. Business Events

E. OData services

Correct Answer: ADE

The following are the integration frameworks that have been released for ABAP cloud development: SOAP consumption: This framework allows you to consume SOAP web services from ABAP cloud applications. You can use the ABAP Development Tools in Eclipse to create a service consumption model based on a WSDL file or URL. The service consumption model generates the required ABAP artifacts, such as proxy classes, data types, and constants, to access the web service. You can then use the proxy classes to call the web service operations from your ABAP code1 Business Events: This framework allows you to publish and subscribe to business events from ABAP cloud applications. Business events are messages that represent a change in the state of a business object or process. You can use the ABAP Development Tools in Eclipse to create a business event definition based on a CDS view entity or a projection view. The business event definition specifies the event key, the event payload, and the event metadata. Youcan then

use the ABAP Messaging Channel (AMC) framework to publish and subscribe to business events using the AMC API2 OData services: This framework allows you to expose and consume OData services from ABAP cloud applications. OData is a standardized protocol forcreating and consuming RESTful APIs. You can use the ABAP RESTful Application Programming Model (RAP) to create OData services based on CDS view entities or projection views. The RAP framework generates the required OData metadata and runtime artifacts, such as service definitions, service bindings, and service implementations. You can then use the SAP Gateway framework to register and activate your OData services. You can also use the ABAP Development Tools in Eclipse to consume OData services from other sources using the service consumption model3 The other integration frameworks are not released for ABAP cloud development, as they are either not supported or not recommended for cloud scenarios. These frameworks are: CDS Views: CDS views are not an integration framework, but a data modeling framework. CDS views are used to define data models based on database tables or other CDS view entities. CDS views can have associations, aggregations, filters, parameters, and annotations. CDS views can also be used as the basis for other integration frameworks, such as OData services or business events4 Business Add-ins (BAdIs): BAdIs are not supported for ABAP cloud development, as they are part of the classic ABAP enhancement framework. BAdIs are used to implement custom logic in predefined enhancement spots in the standard SAP code. BAdIs are not compatible with the cloud strategy and the clean core paradigm, as they modify the SAP code and can cause upgrade and maintenance issues. For ABAP cloud development, SAP recommends using the key user extensibility tools or the side-by-side extensibility approach instead of BAdIs. References: Consuming SOAP Services - ABAP Keyword Documentation, Business Events - ABAP Keyword Documentation, OData Services - ABAP Keyword Documentation, CDS Data Model Views - ABAP Keyword Documentation, [Business Add- Ins (BAdIs) - ABAP Keyword Documentation]

QUESTION 4

Setting a field to read-only in which object would make the field read-only in all applications of the RESTful Application Programming model?

A. Service definition

B. Behaviour definition

C. Projection view

D. Metadata extension

Correct Answer: B

The object that can be used to set a field to read-only in all applications of the RESTful Application Programming model (RAP) is the behaviour definition. The behaviour definition is a CDS artefact that defines the business logic and the UI

behaviour of a business object. A business object is a CDS entity that represents a business entity or concept, such as a customer, an order, or a product. The behaviour definition can specify the properties of the fields of a business object,

such as whether they are mandatory, read-only, or transient. These properties are valid for all applications that use the business object, such as transactional, analytical, or draft-enabled apps12. For example:

The following code snippet defines a behaviour definition for a business object ZI_PB_APPLICATION. It sets the field APPLICATION to read-only for all applications that use this business object:

define behavior for ZI_PB_APPLICATION { field (read only) APPLICATION; ... } You cannot do any of the following:

A. Service definition: A service definition is a CDS artefact that defines the interface and the binding of a service. A service is a CDS entity that exposes the data and the functionality of one or more business objects as OData, InA, or SQL services. A service definition can specify the properties of the fields of a service, such as whether they are filterable, sortable, or aggregatable. However, these properties are only valid for the specific service that uses the business object, not for all applications that use the business object12. C. Projection view: A projection view is a CDS

artefact that defines a view on one or more data sources, such as tables, views, or associations. A projection view can select, rename, or aggregate the fields of the data sources, but it cannot change the properties of the fields, such as whether they are read-only or not. The properties of the fields are inherited from the data sources or the behaviour definitions of the business objects12.

D. Metadata extension: A metadata extension is a CDS artefact that defines additional annotations for a CDS entity, such as a business object, a service, or a projection view. A metadata extension can specify the properties of the fields of a CDS entity for UI or analytical purposes, such as whether they are visible, editable, or hidden. However, these properties are only valid for the specific UI or analytical application that uses the metadata extension, not for all applications that use the CDS entity12.

References: 1: ABAP CDS - Data Definitions - ABAP Keyword Documentation - SAP Online Help 2: ABAP CDS - Behavior Definitions - ABAP Keyword Documentation - SAP Online Help

QUESTION 5

Which of the following ABAP SQL statements are valid? Note: There are 2 correct answers to this question.

A. SELECT FROM /dmo/connection FIELDS carrid O airpfrom, MAX(distance) AS dist_max, MIN(distance) AS dist_min GROUP BY carrid, airpfrom INTO TABLE @DATA(It_hits)

B. SELECT FROM /dmo/connection FIELDS \/ O carrid, airpfrom, MAX(distance) AS dist_max, MIN(distance) AS dist_min INTO TABLE @DATA(It_hits)

C. SELECT FROM /dmo/connection FIELDS V D MAX(distance) AS dist_max MIN(distance) AS dist_min INTO TABLE @DATA(It_hits).

D. SELECT FROM /dmo/connection FIELDS r--i carrid, airpfrom u GROUP BY carrid, connid INTO TABLE @DATA(It_hits).

Correct Answer: AB

Explanation: The following are the explanations for each ABAP SQL statement:

A: This statement is valid. It selects the fields carrid, airpfrom, and the aggregate functions MAX(distance) and MIN(distance) from the table /dmo/connection, and groups the results by carrid and airpfrom. The aggregate functions are aliased

as dist_max and dist_min. The results are stored in an internal table named It_hits, which is created using the inline declaration operator @DATA.

B: This statement is valid. It is similar to statement A, except that it does not specify the GROUP BY clause. This means that the aggregate functions are applied to the entire table, and the results are stored in an internal table named It_hits,

which is created using the inline declaration operator @DATA.

C: This statement is invalid. It selects the aggregate functions MAX(distance) and MIN(distance) from the table /dmo/connection, but it does not specify any grouping or non-aggregate fields. This is not allowed in ABAP SQL, as the SELECT

list must contain at least one non-aggregate field or a GROUP BY clause. The statement will cause a syntax error.

D: This statement is invalid. It selects the fields carrid and airpfrom from the table /dmo/connection, and groups the results by carrid and connid. However, the field connid is not included in the SELECT list, which is not allowed in ABAP SQL,

as the GROUP BY clause must contain only fields that are also in the SELECT list.

The statement will cause a syntax error.

References: SELECT - ABAP Keyword Documentation, GROUP BY - ABAP Keyword Documentation

QUESTION 6

Which of the following are ABAP Cloud Development Model rules? Note: There are 2 correct answers to this question.

A. Use public SAP APIs and SAP extension points.

B. Build ABAP RESTful application programming model-based services.

C. Reverse modifications when a suitable public SAP API becomes available.

D. Build ABAP reports with either ABAP List Viewer (ALV) or SAP Fiori.

Correct Answer: AB

Use public SAP APIs and SAP extension points. This rule ensures that the ABAP Cloud code is stable, reliable, and compatible with the SAP solutions and the cloud operations. Public SAP APIs and SAP extension points are the only allowed interfaces and objects to access the SAP platform and the SAP applications. They are documented, tested, and supported by SAP. They also guarantee the lifecycle stability and the upgradeability of the ABAP Cloud code1. Build ABAP RESTful application programming model-based services. This rule ensures that the ABAP Cloud code follows the state-of-the-art development paradigm for building cloud-ready business services. The ABAP RESTful application programming model (RAP) is a framework that provides a consistent end-to-end programming model for creating, reading, updating, and deleting (CRUD) business data. RAP also supports draft handling, authorization checks, side effects, validations, and custom actions. RAP exposes the business services as OData services that can be consumed by SAP Fiori apps or other clients2.

QUESTION 7

Which of the following are features of Core Data Services? Note: There are 3 correct answers to this question.

A. Inheritance

B. Associations

C. Annotations

D. Delegation

E. Structured Query Language (SQL)

Correct Answer: BCE

Core Data Services (CDS) is a framework for defining and consuming semantically rich data models in SAP HANA. CDS supports various features that enhance the capabilities of SQL and enable developers to create data models that are optimized for performance, readability, and extensibility12. Some of the features of CDS are: Associations: Associations are a way of defining relationships between CDS entities, such as tables or views. Associations enable navigation and path expressions in CDS queries, which allow accessing data from related entities without explicit joins. Associations also support cardinality, referential constraints, and cascading options34. Annotations: Annotations are a

way of adding metadata to CDS entities or their elements, such as fields or parameters. Annotations provide additionalinformation or instructions for the CDS compiler, the database, or the consumers of the CDS views. Annotations can be used for various purposes, such as defining access control, UI rendering, OData exposure, or search capabilities5 . Structured Query Language (SQL): SQL is the standard language for querying and manipulating data in relational databases. CDS is based on SQL and extends it with additional features and syntax. CDS supports SQL features such as joins, aggregations, filters, expressions, functions, and subqueries. CDS also supports SQL Script, which is a scripting language for stored procedures and functions in SAP HANA . You cannot do any of the following: Inheritance: Inheritance is not a feature of CDS. Inheritance is a concept in object- oriented programming that allows a class to inherit the properties and methods of another class. CDS does not support object-oriented programming or classes. Delegation: Delegation is not a feature of CDS. Delegation is a concept in object- oriented programming that allows an object to delegate some of its responsibilities to another object. CDS does not support object-oriented programming or objects. References: 1: Core Data Services (CDS) | CAPire 2: Core Data Services [CDS] in SAP S/4 HANA | SAP Blogs 3: Associations in Core Data Services (CDS) | SAP Help Portal 4: [CDS DDL - Association - ABAP Keyword Documentation - SAP Online Help] 5: [Annotations in Core Data Services (CDS) | SAP Help Portal] : [CDS DDL - Annotation - ABAP Keyword Documentation - SAP Online Help] : [Structured Query Language (SQL) | SAP Help Portal] : [CDS DDL - SQL Features - ABAP Keyword Documentation - SAP Online Help] : [Object-Oriented Programming in ABAP | SAP Help Portal]

QUESTION 8

In a program you find this source code

AUTHORITY-CHECK OBJECT \\'/DWO/TRVL (ID \\'CNTRY\\' FIELD \\'DE* ID ACTVT FIELD \\'03".

Which of the following apply? Note: There are 2 correct answers to this question.

A. If the user is authorized for \\'CNTRY = \\'DE\\' then the return code is always 0.

B. If the user is NOT authorized for \\'CNTRY\\' = \\'DE\\' OR for \\'ACTVT\\' = \\'03 then the program will terminate.

C. If the user is authorized for \\'CNTRY = \\'DE\\' AND for \\'ACTVT = \\'03 then the return code is 0.

D. AUTHORITY CHECK verifies whether a user is authorized for/DMO/TRVL" with the listed field values.

Correct Answer: CD

QUESTION 9

Given this code,

```
INTERFACE if1.
  METHODS m1.
ENDINTERFACE.

CLASS cl1 DEFINITION.
...
  INTERFACES if1.
ENDCLASS.
...
CLASS cl2 DEFINITION.
...
  DATA mo_if1 TYPE REF TO if1.
...
ENDCLASS.
...
```

What are valid statements? Note: There are 3 correct answers to this question

A. In class CL1, the interface method is named if-ml.

B. Class CL2 uses the interface.

C. Class CL1 uses the interface.

D. In class CL2, the interface method is named ifl-ml.

E. Class CL1 implements the interface.

Correct Answer: CDE

The following are the explanations for each statement:

C: This statement is valid. Class CL1 uses the interface. This is because class CL1 implements the interface ifl using the INTERFACES statement in the public section of the class definition. The INTERFACES statement makes the class compatible with the interface and inherits all the components of the interface. The class can then use the interface components, such as the method ml, by using the interface component selector ~, such as ifl~ml12

E: This statement is valid. Class CL1 implements the interface. This is because class CL1 implements the interface ifl

using the INTERFACES statement in the public section of the class definition. The INTERFACES statement makes the class compatible with the interface and inherits all the components of the interface. The class must then provide an implementation for the interface method ml in the implementation part of the class, unless the method is declared as optional or abstract12

D: This statement is valid. In class CL2, the interface method is named ifl~ml. This is because class CL2 has a data member named m0_ifl of type REF TO ifl, which is a reference to the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable m0_ifl. The interfacemethod ml has the name ifl~ml in the class, where ifl is the name of the interface and the character ~ is the interface component selector12 The other statements are not valid, as they have syntax errors or logical errors. These statements are: A: This statement is not valid. In class CL1, the interface method is named ifl~ml, not if-ml. This is because class CL1 implements the interface ifl using the INTERFACES statement in the public section of the class definition. The interface ifl defines a method ml, which can be called using the class name or a reference to the class. The interface method ml has the name ifl~ml in the class, where ifl is the name of the interface and the character ~ is the interface component selector. Using the character - instead of the character ~ will cause a syntax error12

B: This statement is not valid. Class CL2 does not use the interface, but only has a reference to the interface. This is because class CL2 has a data member named m0_ifl of type REF TO ifl, which is a reference to the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable m0_ifl. However, class CL2 does not implement the interface ifl, nor does it inherit the interface components. Therefore, class CL2 does not use the interface, but only references the interface12 References: INTERFACES - ABAP Keyword Documentation, CLASS - ABAP Keyword Documentation

---

QUESTION 10

When processing an internal table with the statement LOOP AT itab... ENDLOOP, what system variable contains the current row number?

A. sy-index

B. sy-subrc

C. sy-linno

D. sy-tabix

Correct Answer: D

When processing an internal table with the statement LOOP AT itab... ENDLOOP, the system variable that contains the current row number is sy-tabix. The sy-tabix variable is a predefined field of the system structure sy that holds the index or the row number of the current line in an internal table loop. The sy-tabix variable is initialized with the value 1 for the first loop pass and is incremented by 1 for eachsubsequent loop pass. The sy-tabix variable can be used to access or modify the current line of the internal table using the index access12. References: 1: LOOP AT itab - ABAP Keyword Documentation - SAP Online Help 2: System Fields - ABAP Keyword Documentation - SAP Online Help

---

QUESTION 11

After you created a database table in the RESTful Application Programming model, what do you create next?

A. A metadata extension

B. A projection view

C. A data model view

D. A service definition

Correct Answer: B

After you created a database table in the RESTful Application Programming model (RAP), the next step is to create a projection view on the database table. A projection view is a CDS artefact that defines a view on one or more data sources,

such as tables, views, or associations. A projection view can select, rename, or aggregate the fields of the data sources, but it cannot change the properties of the fields, such as whether they are read- only or not.The properties of the fields

are inherited from the data sources or the behaviour definitions of the business objects12. For example:

The following code snippet defines a projection view ZI_AGENCY on the database table /DMO/AGENCY:

define view ZI_AGENCY as select from /dmo/agency { key agency_id, agency_name, street, city, region, postal_code, country, phone_number, url } The projection view is used to expose the data of the database table to the service definition,

which is the next step in the RAP. The service definition is a CDS artefact that defines the interface and the binding of a service. A service is a CDS entity that exposes the data and the functionality of one or more business objects as OData,

InA, or SQL services.A service definition can specify the properties of the fields of a service, such as whether they are filterable, sortable, or aggregatable12. For example:

The following code snippet defines a service definition ZI_AGENCY_SRV that exposes the projection view ZI_AGENCY as an OData service:

define service ZI_AGENCY_SRV { expose ZI_AGENCY as Agency; } You cannot do any of the following:

A. A metadata extension: A metadata extension is a CDS artefact that defines additional annotations for a CDS entity, such as a business object, a service, or a projection view. A metadata extension can specify the properties of the fields of a CDS entity for UI or analytical purposes, such as whether they are visible, editable, or hidden. However, a metadata extension is not the next step after creating a database table in the RAP, as it is not required to expose the data of the database table to the service definition.A metadata extension can be created later to customize the UI or analytical application that uses the service12. C. A data model view: A data model view is a CDS artefact that defines a view on one or more data sources, such as tables, views, or associations. A data model view can select, rename, or aggregate the fields of the data sources, and it can also change the properties of the fields, such as whether they are read-only or not. The properties of the fields are defined by the annotations or the behaviour definitions of the data model view. A data model view is used to define the data model of a business object, which is a CDSentity that represents a business entity or concept, such as a customer, an order, or a product. However, a data model view is not the next step after creating a database table in the RAP, as it is not required to expose the data of the database table to the service definition.A data model view can be created later to define a business object that uses the database table as a data source12.

D. A service definition: A service definition is a CDS artefact that defines the interface and the binding of a service. A service is a CDS entity that exposes the data and the functionality of one or more business objects as OData, InA, or SQL services. A service definition can specify the properties of the fields of a service, such as whether they are filterable, sortable, or aggregatable. However, a service definition is not the next step after creating a database table in the RAP, as it requires a projection view or a data model view to expose the data of the database table.A service definition can be created after creating a projection view or a data model view on the database table12. References:1:ABAP CDS - Data Definitions - ABAP Keyword Documentation - SAP Online Help2:ABAP CDS - Service Definitions - ABAP Keyword Documentation - SAP Online Help

QUESTION 12

Which of the following are parts of answers to this question? Note: There are 2 correct answers to this question.

A. Partitioning attributes

B. Extension

C. Field list

D. Semantic table attributes

Correct Answer: BC

A CDS view is a data definition that defines a data structure and a data selection from one or more data sources. A CDS view consists of several parts, but two of them are:

Extension: An extension is an optional clause that allows a CDS view to extend another CDS view by adding new elements, annotations, or associations. The extension clause has the syntax EXTEND VIEW view_name WITH view_name.

The first view_name is the name of the CDS view that is being extended, and the second view_name is the name of the CDS view that is doing the extension1.

Field list: A field list is a mandatory clause that specifies the elements of the CDS view. The field list has the syntax SELECT FROM data_source { element_list }. The data_source is the name of the data source that the CDS view selects data

from, and the element_list is a comma-separated list of elements that the CDS view exposes. The elements can be fields of the data source, expressions, associations, or annotations2.

The following example shows a CDS view that extends another CDS view and defines a field list:

@AbapCatalog.sqlViewName: `ZCDS_EXT\\' define view Z_CDS_Extension extend view Z_CDS_Base with Z_CDS_Extension as select from ztable { // field list key ztable.id as ID, ztable.name as Name, ztable.age as Age, // extension

@Semantics.currencyCode: true ztable.currency as Currency }

The other options are not parts of a CDS view, but rather related concepts:

Partitioning attributes: Partitioning attributes are attributes that are used to partition a table into smaller subsets of data. Partitioning attributes are defined in the ABAP Dictionary for transparent tables and can improve the performance and

scalability of data access. Partitioning attributes are not part of the CDS view definition, but rather the underlying table definition3.

Semantic table attributes: Semantic table attributes are attributes that provide additional information about the meaning and usage of a table. Semantic table attributes are defined in the ABAP Dictionary for transparent tables and can be used

to enhance the data modeling and consumption of the table. Semantic table attributes are not part of the CDS view definition, but rather the underlying table definition4.

References: 1: Extending CDS Views | SAP Help Portal 2: SELECT List - ABAP Keyword Documentation 3: Partitioning

Attributes - ABAP Keyword Documentation 4: Semantic Table Attributes - ABAP Keyword Documentation

QUESTION 13

What is the sequence priority when evaluating a logical expression?

A. NOT 1

B. OR 3

C. AND 2

D. A B C

E. CAB

F. A C B

G. B A C

Correct Answer: C

The sequence priority when evaluating a logical expression is C. A C B, which means NOT, AND, OR. This is the order of precedence of the Boolean operators in ABAP, which determines how the system implicitly parenthesizes all logical expressions that are not closed by explicit parentheses. The operator with the highest priority is evaluated first, and the operator with the lowest priority is evaluated last. The order of precedence of the Boolean operators in ABAP is as follows12: NOT: The NOT operator is a unary operator that negates the logical expression that follows it. It has the highest priority and is evaluated before any other operator. For example, in the expression NOT a AND b, the NOT operator is applied to a first, and then the AND operator is applied to the result and b. AND: The AND operator is a binary operator that returns true if both logical expressions on its left and right are true, and false otherwise. It has the second highest priority and is evaluated before the OR and EQUIV operators. For example, in the expression a AND b OR c, the AND operator is applied to a and b first, and then the OR operator is applied to the result and c. OR: The OR operator is a binary operator that returns true if either or both logical expressions on its left and right are true, and false otherwise. It has the third highest priority and is evaluated after the NOT and AND operators, but before the EQUIV operator. For example, in the expression a OR b EQUIV c, the OR operator is applied to a and b first, and then the EQUIV operator is applied to the result and c. EQUIV: The EQUIV operator is a binary operator that returns true if both logical expressions on its left and right have the same truth value, and false otherwise. It has the lowest priority and is evaluated after all other operators. For example, in the expression a AND b EQUIV c OR d, the EQUIV operator is applied to a AND b and c last, after the AND and OR operators are applied. References: 1: log_exp - Boolean Operators and Parentheses - ABAP Keyword Documentation SAP Online Help 2: Logical Expressions (log_exp) - ABAP Keyword Documentation - SAP Online Help

QUESTION 14

Exhibit

(Sorry, we do not have a more clear image. If we have a better resource for the image, we will update this one immediately.)

Which of the following ABAP SQL snippets are syntactically correct ways to provide a value for the parameter on line #4? Note: There are 2 correct answers to this question

A. ...SELECT * FROM deno_cds_param_view_entity (p_date = @ (cl_abap_context_info- >get_system_date ())...

B. ...SELECT * FROM deno_cds_param_view_entity (p_date - \\'20230101\\')...)

C. ...SELECT * FROM demo_cds_param_view_entity (p_date: 20238181\\')...)

D. ...SELECT * FROM demo_cds_param_view entity (p_date: $session.system_date)...

Correct Answer: AB

QUESTION 15

Class super has subclass sub. Which rules are valid for the sub constructor? Note: There are 2 correct answers to this question.

A. The method signature can be changed.

B. Import parameters can only be evaluated after calling the constructor of super.

C. The constructor of super must be called before using any components of your own instance.

D. Events of your own instance cannot be raised before the registration of a handler in super.

Correct Answer: AC

The sub constructor is the instance constructor of the subclass sub that inherits from the superclass super. The sub constructor has some rules that it must follow when it is defined and implemented12. Some of the valid rules are:

The method signature can be changed: This is true. The sub constructor can have a different method signature than the super constructor, which means that it can have different input parameters, output parameters, or exceptions. However,

the sub constructor must still call the super constructor with appropriate actual parameters that match its interface12.

The constructor of super must be called before using any components of your own instance: This is true. The sub constructor must ensure that the super constructor is called explicitly using super->constructor before accessing any instance

components of its own class, such as attributes or methods. This is because the super constructor initializes the inherited components of the subclass and sets the self-reference me-> to the current instance12.

You cannot do any of the following:

Import parameters can only be evaluated after calling the constructor of super:

This is false. The sub constructor can evaluate its own import parameters before calling the constructor of super, as long as it does not access any instance components of its own class. For example, the sub constructor can use its import

parameters to calculate some values or check some conditions that are needed for calling the super constructor12.

Events of your own instance cannot be raised before the registration of a handler in super: This is false. The sub constructor can raise events of its own instance before calling the constructor of super, as long as it does not access any

instance components of its own class. For example, the sub constructor can raise an event to notify the consumers of the subclass about some status or error that occurred during the initialization of the subclass12.

References: 1: Inheritance and Constructors - ABAP Keyword Documentation - SAP Online Help 2: Using Static and Instance constructor methods | SAP Blogs