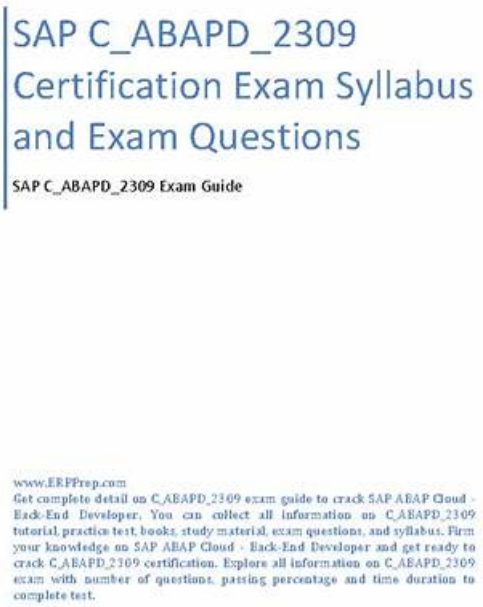
C-ABAPD-2309 Certification Questions - C-ABAPD-2309 PDF Guide



P.S. Free 2025 SAP C-ABAPD-2309 dumps are available on Google Drive shared by Easy4Engine: https://drive.google.com/open?id=1fl8LRZ38KrtvPiDoEytf5a9-eCM5Xzbm

Our C-ABAPD-2309 exam prep can bring you high quality learning platform to pass the variety of exams. C-ABAPD-2309 guide dumps are elaborately composed with major questions and answers. C-ABAPD-2309 test question only needs 20 hours to 30 hours to practice. There is important to get the C-ABAPD-2309 Certification as you can. There is a fabulous product to prompt the efficiency--the C-ABAPD-2309 exam prep, as far as concerned, it can bring you high quality learning platform to pass the variety of exams.

If you come to our website to choose C-ABAPD-2309 study materials, you will enjoy humanized service. Firstly, we have chat windows to wipe out your doubts about our C-ABAPD-2309 study materials. You can ask any question about our study materials. All of our online workers are going through special training. They are familiar with all details of our C-ABAPD-2309 Study Materials. Also, you have easy access to our free demo. Once you apply for our free trials of the study materials, our system will quickly send it via email.

>> C-ABAPD-2309 Certification Questions <<

SAP C-ABAPD-2309 PDF Guide - C-ABAPD-2309 Vce Download

Are you aware of the importance of the C-ABAPD-2309 certification? If your answer is not, you may place yourself at the risk of be eliminated by the labor market. As we know, the C-ABAPD-2309 certification is the main reflection of your ability. If you want to maintain your job or get a better job for making a living for your family, it is urgent for you to try your best to get the C-ABAPD-

2309 Certification. We are glad to help you get the certification with our best C-ABAPD-2309 study materials successfully.

SAP C-ABAPD-2309 Exam Syllabus Topics:

Topic	Details
Topic 1	• Object-oriented design: It measures your knowledge about encapsulation, upcast, inheritance, polymorphism, and interfaces. Moreover, the topic evaluates your knowledge about constructor calls, Exception classes, and singleton pattern.
Topic 2	• ABAP RESTful Application Programming Model: This topic explains the ABAP Restful Application Programming model, ABAP development, and the architecture of the ABAP Restful Application Programming model.
Topic 3	• ABAP SQL and code pushdown: It discusses ABAP SQL, arithmetic expressions, manage dates, and create joins.
Topic 4	• Core ABAP programming: This topic covers ABAP data types, the ABAP dictionary, modularization, exceptions SAP HANA database tables, and logical expressions, operator precedence.
Topic 5	• SAP clean core extensibility and ABAP cloud: The topic explains extension pattern, extension rules, ABAP cloud development, and ABAP cloud rules.

SAP Certified Associate - Back-End Developer - ABAP Cloud Sample Questions (Q40-Q45):

NEW QUESTION # 40
Using ABAP SQL, which select statement selects the mat field on line #17?

- A. SELECT mat FROM Material...
- B. SELECT mat FROM demo_sales_so_i...
- C. SELECT mat FROM demo sales cds material ve...
- D. SELECT mat FROM demo_sales_cds_so_i_ve...

Answer: D

Explanation:
Using ABAP SQL, the select statement that selects the mat field on line #17 is:
SELECT mat FROM demo_sales_cds_so_i_ve...
This statement selects the mat field from the CDS view demo_sales_cds_so_i_ve, which is defined on line #1.
The CDS view demo_sales_cds_so_i_ve is a projection view that projects the fields of the CDS view demo_sales_cds_so_i, which is defined on line #2. The CDS view demo_sales_cds_so_i is a join view that joins the fields of the database table demo_sales_so_i, which is defined on line #3, and the CDS view demo_sales_cds_material_ve, which is defined on line #4. The CDS view demo_sales_cds_material_ve is a value help view that provides value help for the material field of the database table demo_sales_so_i. The mat field is an alias for the material field of the database table demo_sales_so_i, which is defined on line #91.
The other options are not valid because:
* A. SELECT mat FROM Material... is not valid because Material is not a valid data source in the given code. There is no CDS view or database table named Material.
* C. SELECT mat FROM demo_sales_so_i... is not valid because demo_sales_so_i is not a valid data source in the given code. There is no CDS view named demo_sales_so_i, only a database table. To access a database table, the keyword TABLE must be used, such as SELECT mat FROM TABLE demo_sales_so_i...
* D. SELECT mat FROM demo sales cds material ve... is not valid because demo sales cds material ve is not a valid data source in the given code. There is no CDS view or database table named demo sales cds material ve. The correct name of the CDS view is demo_sales_cds_material_ve, with underscores instead of spaces.
References: 1: Projection Views - ABAP Keyword Documentation

NEW QUESTION # 41

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

Answer: B

Explanation:
Explanation
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 CDS entity 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 Help 2: ABAP CDS - Service Definitions - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION # 42
You want to provide a short description of the data definition for developers that will be attached to the database view Which of the following annotations would do this if you inserted it on line #27

- A. @EndUserText label
- B. @EndUserText.quickInfo
- C. @UI.badge.title.label
- D. @UI headerinto description label

Answer: A

Explanation:
The annotation that can be used to provide a short description of the data definition for developers that will be attached to the database view is the @EndUserText.label annotation. This annotation is used to specify a text label for the data definition that can be displayed in the development tools or in the documentation. The annotation can be inserted on line #27 in the code snippet provided in the question12. For example:
* The following code snippet uses the @EndUserText.label annotation to provide a short description of the data definition for the CDS view ZCDS_VIEW:
@AbapCatalog.sqlViewName: 'ZCDS_VIEW' @AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true @AccessControl.authorizationCheck: #CHECK @EndUserText.label:
'CDS view for flight data' "short description for developers define view ZCDS_VIEW as select from sflight { key carrid, key connid, key fldate, seatsmax, seatsocc } You cannot do any of the following:
* @UI.headerInfo.description.label: This annotation is used to specify a text label for the description field of the header information of a UI element. This annotation is not relevant for the data definition of a database view12.
* @UI.badge.title.label: This annotation is used to specify a text label for the title field of a badge UI element. This annotation is not relevant for the data definition of a database view12.
* @EndUserText.quickInfo: This annotation is used to specify a quick information text for the data definition that can be displayed as a tooltip in the development tools or in the documentation. This annotation is not the same as a short description or a label for the data definition12.
References: 1: ABAP CDS - SAP Annotations - ABAP Keyword Documentation - SAP Online Help 2: ABAP CDS - Data Definitions - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION # 43
Exhibit:

```
1  INTERFACE if1.
2      METHODS m1.
3  ENDINTERFACE.
4
5  CLASS cl1 DEFINITION.
6  PUBLIC SECTION.
7      INTERFACES if1.
8      METHODS m2.
9  ENDCLASS.
10 ...
11 *In a method of another class
12 DATA go_if1 TYPE REF TO if1.
13 DATA go_cl1 TYPE REF TO cl1.
14 go_cl1 = NEW #(...).
15 go_if1 = go_cl1.
```

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

- A. go_ifl may call method m2 with go if->m2(...).
- B. go_if 1 may call method ml with go_ift->ml().
- C. go_cll may call method ml with go_dl->ifl-ml().
- D. Instead of go_cll = NEW #() you could use go_iff - NEW #(...).
- E. Instead of go ell = NEW #(...) you could use go ifl = NEW cll(. ...).

Answer: A,B,E

Explanation:
Explanation
The following are the explanations for each statement:
A: This statement is valid. go_ifl may call method ml with go_ifl->ml(). This is because go_ifl is a data object 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 go_ifl. The class cll implements the interface ifl, which means that it provides an implementation of the method ml. The data object go_ifl is assigned to a new instance of the class cll using the NEW operator and the inline declaration operator @DATA. Therefore, when go_ifl->ml() is called, the implementation of the method ml in the class cll is executed123 B: This statement is valid. Instead of go_cll = NEW #(...) you could use go_ifl = NEW cll(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The class cll implements the interface ifl, which means that it is compatible with the interface ifl. Therefore, go_ifl can be assigned to a new instance of the class cll using the NEW operator and the class name cll. The inline declaration operator @DATA is optional in this case, as go_ifl is already declared. The parentheses after the class name cll can be used to pass parameters to the constructor of the class cll, if any123 E: This statement is valid. go_ifl may call method m2 with go_ifl->m2(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The class cll implements the

interface ifl, which means that it inherits all the components of the interface ifl. The class cll also defines amethod m2, which is a public method of the class cll. Therefore, go_ifl can call the method m2 using the reference variable go_ifl. The method m2 is not defined in the interface ifl, but it is accessible through the interface ifl, as the interface ifl is implemented by the class cll. The parentheses after the method name m2 can be used to pass parameters to the method m2, if any123 The other statements are not valid, as they have syntax errors or logical errors. These statements are:
C: This statement is not valid. go_cll may call method ml with go_cll->ifl~ml(). This is because go_cll is a data object of type REF TO cll, which is a reference to the class cll. The class cll implements the interface ifl, which means that it inherits all the components of the interface ifl. The interface ifl defines a method ml, which can be called using the reference variable go_cll. However, the syntax for calling an interface method using a class reference is go_cll->ml(), not go_cll->ifl~ml(). The interface component selector ~ is only used when calling an interface method using an interface reference, such as go_ifl->ifl~ml(). Using the interface component selector ~ with a class reference will cause a syntax error123 D: This statement is not valid. Instead of go_cll = NEW #() you could use go_ifl = NEW #(...). This is because go_ifl is a data object of type REF TO ifl, which is a reference to the interface ifl. The interface ifl cannot be instantiated, as it does not have an implementation. Therefore, go_ifl cannot be assigned to a new instance of the interface ifl using the NEW operator and the inline declaration operator @DATA.
This will cause a syntax error or a runtime error. To instantiate an interface, you need to use a class that implements the interface, such as the class cll123 References: INTERFACES - ABAP Keyword Documentation, CLASS - ABAP Keyword Documentation, NEW - ABAP Keyword Documentation

NEW QUESTION # 44
What are the effects of this annotation? Note: There are 2 correct answers to this question.

- A. The value of sy-langu will be passed to the CDS view automatically when you use the CDS view in ABAP but not when you use it in another view entity
- B. It is no longer possible to pass your own value to the parameter.
- C. You can still override the default value with a value of your own.
- D. The value of sy-langu will be passed to the CDS view automatically both when you use the -1 CDS view in ABAP and in another CDS view entity (view on view).

Answer: C,D

Explanation:
The annotation @Environment.systemField: #LANGUAGE is used to assign the ABAP system field sy-langu to an input parameter of a CDS view or a CDS table function. This enables the implicit parameter passing in Open SQL, which means that the value of sy-langu will be automatically passed to the CDS view without explicitly specifying it in the WHERE clause. This also applies to the CDS views that use the annotated CDS view as a data source, which means that the value of sy-langu will be propagated to the nested CDS views (view on view)12. For example:
* The following code snippet defines a CDS view ZI_FLIGHT_TEXTS with an input parameter p_langu that is annotated with @Environment.systemField: #LANGUAGE:
define view ZI_FLIGHT_TEXTS with parameters p_langu : syst_langu @<Environment.systemField:
#LANGUAGE as select from sflight left outer join scarr on sflight.carrid = scarr.carrid left outer join stext on scarr.carrid = stext.carrid { sflight.carrid, sflight.connid, sflight.fldate, scarr.carrname, stext.text as carrtext } where stext.langu = :p_langu
* The following code snippet shows how to use the CDS view ZI_FLIGHT_TEXTS in ABAP without specifying the value of p_langu in the WHERE clause. The value of sy-langu will be automatically passed to the CDS view:
SELECT carrid, connid, fldate, carrname, carrtext FROM zi_flight_texts INTO TABLE @DATA(lt_flights).
* The following code snippet shows how to use the CDS view ZI_FLIGHT_TEXTS in another CDS view ZI_FLIGHT_REPORT. The value of sy-langu will be automatically passed to the nested CDS view ZI_FLIGHT_TEXTS:
define view ZI_FLIGHT_REPORT with parameters p_langu : syst_langu @<Environment.systemField:
#LANGUAGE as select from zi_flight_texts(p_langu) { carrid, connid, fldate, carrname, carrtext, count(*) as flight_count } group by carrid, connid, fldate, carrname, carrtext The annotation @Environment.systemField: #LANGUAGE does not prevent the

possibility of overriding the default value with a value of your own. You can still specify a different value for the input parameter p_langu in the WHERE clause, either in ABAP or in another CDS view. This will override the value of sy-langu and pass the specified value to the CDS view12. For example:
* The following code snippet shows how to use the CDS view ZI_FLIGHT_TEXTS in ABAP with a specified value of p_langu in the WHERE clause. The value 'E' will be passed to the CDS view instead of the value of sy-langu:
SELECT carrid, connid, fldate, carrname, carrtext FROM zi_flight_texts WHERE p_langu = 'E' INTO TABLE @DATA(lt_flights).
* The following code snippet shows how to use the CDS view ZI_FLIGHT_TEXTS in another CDS view ZI_FLIGHT_REPORT with a specified value of p_langu in the WHERE clause. The value 'E' will be passed to the nested CDS view ZI_FLIGHT_TEXTS instead of the value of sy-langu:
define view ZI_FLIGHT_REPORT with parameters p_langu : syst_langu @<Environment.systemField:
#LANGUAGE as select from zi_flight_texts(p_langu) { carrid, connid, fldate, carrname, carrtext, count(*) as flight_count } where p_langu = 'E' group by carrid, connid, fldate, carrname, carrtext References: 1: ABAP CDS - parameter_annot - ABAP Keyword Documentation - SAP Online Help 2: ABAP CDS - session_variable - ABAP Keyword Documentation - SAP Online Help

NEW QUESTION # 45
.....

P.S. Free & New C-ABAPD-2309 dumps are available on Google Drive shared by Easy4Engine: https://drive.google.com/open?id=1fl8LRZ38KrtvPiDoEytf5a9-eCM5Xzbm