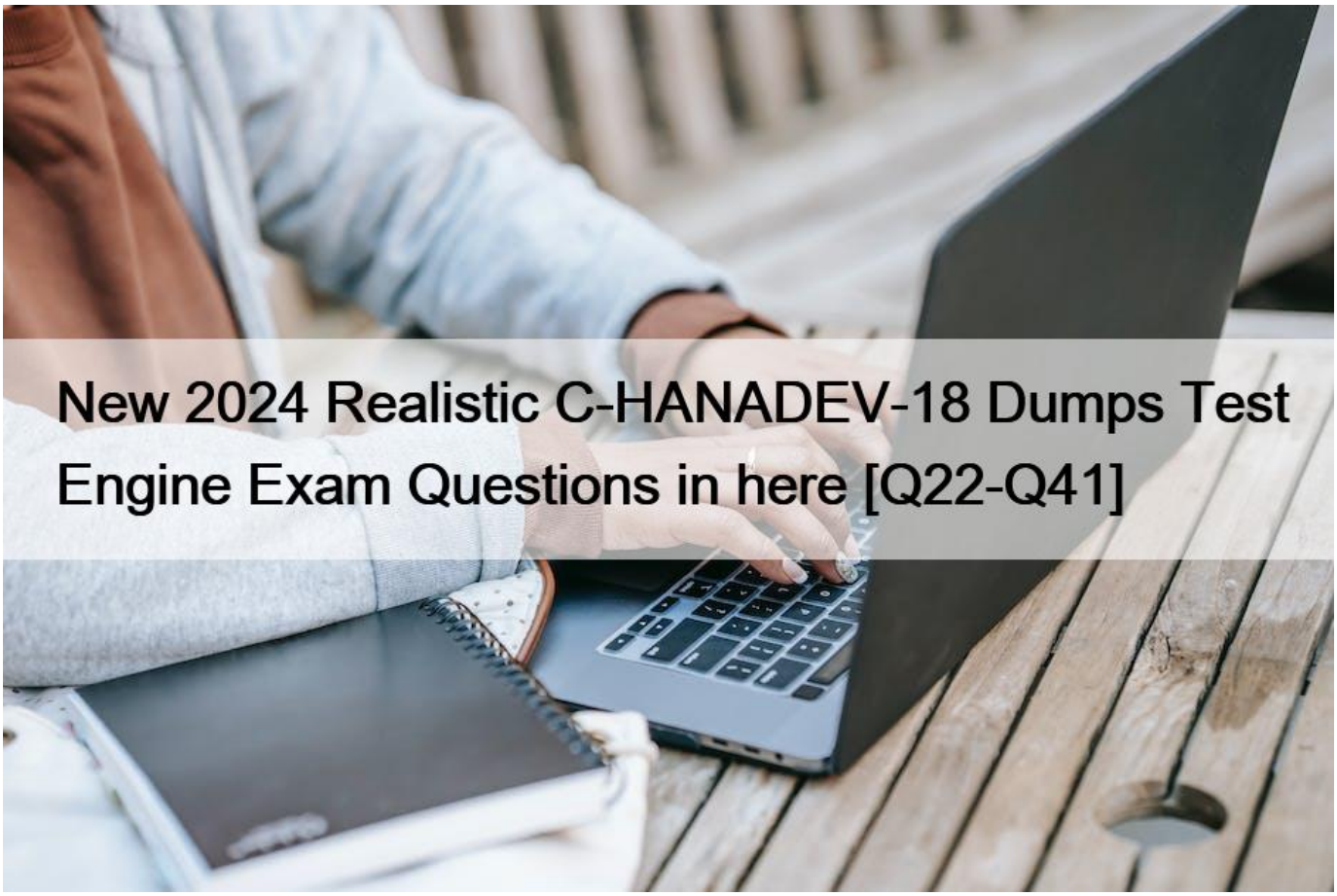
New 2024 Realistic C-HANADEV-18 Dumps Test Engine Exam Questions in here [Q22-Q41



New 2024 Realistic C-HANADEV-18 Dumps Test Engine Exam Questions in here
Updated Official licence for C-HANADEV-18 Certified by C-HANADEV-18 Dumps PDF

QUESTION 22

You created an HDI database role, using the SAP Web IDE for SAP HANA and deployed your project. Afterward, you made some modifications to the runtime role.

What happens when you change and re-build the design-time role? Please choose the correct answer.
* The runtime modifications are kept.
* The deployment of the role fails.
* You need to confirm the runtime modifications
* The runtime modifications are overwritten.

QUESTION 23

A Node.js module is executed for the first time in SAP HANA extended application services, advanced model (XS advanced).

Which of the following activities are performed automatically? There are 2 correct answers to this question.
* The source code is compiled to create an executable binary file.
* A new SAP HANA Deployment Infrastructure (HDI) container is created.
* The required modules are downloaded based on module dependencies.
* A new service is executed on the application server.

When a Node.js module is executed for the first time in SAP HANA extended application services, advanced model (XS advanced), the following activities are performed automatically12:

* A new SAP HANA Deployment Infrastructure (HDI) container is created: An HDI container is a logical database schema that contains the database objects and data that are required by the Node.js module. An HDI container is created based on the configuration and definition files of the Node.js module, such as the package.json, the mta.yaml, and the hdi-config.json. The HDI container is bound to the Node.js module as a service, and it can be accessed using the HDI client library or the SQL client library.

* The required modules are downloaded based on module dependencies: The Node.js module may depend on other modules or libraries that provide additional functionality or services, such as express, hdb, or passport. These dependencies are specified in the package.json file of the Node.js module, and they are downloaded and installed automatically from the npm registry or the SAP npm registry when the Node.js module is executed for the first time.

The other options are not correct because they are not activities that are performed automatically when a Node.js module is executed for the first time in XS advanced. The source code is not compiled to create an executable binary file, but rather interpreted and executed by the Node.js runtime environment. A new service is not executed on the application server, but rather the existing Node.js module is executed as a service on the application server. References:

* SAP HANA Platform, Developing Applications with SAP HANA Cloud Platform, Developing Multi-Target Applications, Developing Node.js Modules

* SAP HANA Platform, SAP HANA Extended Application Services, Advanced Model, Developing and Deploying Applications, Developing Node.js Applications

QUESTION 24

You need to synchronize all unsynchronized changes of the remote Git branch with a local Git branch. Which operation do you use? Please choose the correct answer.
* Push
* Rebase
* Pull
* Reset

QUESTION 25

You created several database tables in a multi-target application and need to keep their names as short as possible. Which parameter of the .hdinamespace file do you set to ignore? Please choose the correct answer.
* name
* hdbtable
* mixinTypes
* subfolder

QUESTION 26

Which parameter do you adapt to change the file that is initially delivered by an HTML5 module? Please choose the correct answer.

* Service_url parameter of the mta.xml file
* Welcome file parameter of the xs-app.json file
* Start parameter of the package.json file
* Runners parameter of the projectjson file

QUESTION 27

Which activities do you perform on an application in the SAP HANA XS Advanced Cockpit? There are 2 correct answers to this question.
* Monitor applications assigned to the current space.
* Deploy applications to SAP Cloud Platform.
* Display HDI container content.
* Start, stop, and restart applications.

The SAP HANA XS Advanced Cockpit is a web-based administration tool that allows you to manage the XS Advanced runtime environment, such as organizations, spaces, applications, services, and users. The XS Advanced Cockpit provides a graphical user interface that is similar to the SAP Cloud Platform Cockpit, to provide a consistent user experience in cloud and on-premise. The XS Advanced Cockpit is available for SAP HANA 2.0 SPS03 and later releases, and it replaces the deprecated XS Advanced Administration Tools.

One of the activities that you can perform on an application in the XS Advanced Cockpit is to monitor the applications assigned to the current space. A space is a logical grouping of applications and services that share a common development and runtime environment. Each space belongs to an organization, which is a collection of users and spaces that share a common quota of resources. You can use the XS Advanced Cockpit to view the list of applications deployed in a particular space, and see their status, instances, memory usage, disk usage, and bound services. You can also filter, sort, and search the applications by name, status, or type.

You can also drill down into the details of each application, such as logs, events, environment variables, routes, and service bindings.

Another activity that you can perform on an application in the XS Advanced Cockpit is to start, stop, and restart the applications. You can use the XS Advanced Cockpit to control the lifecycle of the applications deployed in a space, such as starting an application that is stopped, stopping an application that is running, or restarting an application that is experiencing issues. You can also scale the applications by changing the number of instances or the memory and disk quota allocated to each instance. You can also delete the applications that are no longer needed.

The other options are incorrect because they are not activities that you can perform on an application in the XS Advanced Cockpit. You cannot deploy applications to SAP Cloud Platform from the XS Advanced Cockpit, as the XS Advanced Cockpit is only for managing the XS Advanced runtime environment on SAP HANA. To deploy applications to SAP Cloud Platform, you need to use the SAP Cloud Platform Cockpit or the Cloud Foundry Command Line Interface (CF CLI). You also cannot display the HDI container content from the XS Advanced Cockpit, as the HDI container is a database object that is not directly related to the application. To display the HDI container content, you need to use the SAP HANA Database Explorer or the SAP Web IDE for SAP HANA. References:

* SAP HANA Platform 2.0 SPS06: SAP HANA XS Advanced Cockpit, Section 1

* SAP HANA Platform 2.0 SPS06: SAP HANA XS Advanced Cockpit, Section 2

* SAP HANA Platform 2.0 SPS06: SAP HANA XS Advanced Cockpit, Section 3

* SAP HANA Platform 2.0 SPS06: SAP HANA XS Advanced Cockpit, Section 4

* SAP HANA Platform 2.0 SPS06: SAP HANA XS Advanced Cockpit, Section 5

QUESTION 28

Which elements can you specify with the SAP WeblDE for SAP HANA graphical editor for Core Data Services data models? There are 3 correct answers to this question.
* Entity
* Synonym
* Procedure
* Context
* Association

The SAP Web IDE for SAP HANA graphical editor for Core Data Services (CDS) data models allows you to specify the following elements12:

* Entity: An entity is a CDS artifact that defines a data structure with attributes, keys, and associations. An entity can be mapped to a database table or view, or it can be used as a source for another entity or a calculation view. You can create an entity by dragging the entity icon from the palette to the canvas, and then defining its properties and attributes in the properties panel.

* Context: A context is a CDS artifact that defines a namespace for other CDS artifacts, such as entities, views, types, or associations. A context can be used to organize and group related CDS artifacts, and to avoid name conflicts. You can create a context by dragging the context icon from the palette to the canvas, and then defining its name and description in the properties panel. You can also add other CDS artifacts to the context by dragging them from the project explorer or the palette to the context node in the canvas.

* Association: An association is a CDS artifact that defines a relationship between two entities, based on a join condition. An association can be used to navigate from one entity to another, and to filter, aggregate, or project data from the target entity. You can create an association by dragging the association icon from the palette to the canvas, and then connecting it to the source and target entities.

You can also define the cardinality, join condition, and exposed attributes of the association in the properties panel.

The other options are not correct because they are not elements that can be specified with the SAP Web IDE for SAP HANA graphical editor for CDS data models. A synonym is a database object that provides an alternative name for another database object, such as a table, view, or procedure. A synonym is not a CDS artifact, and it cannot be created or edited with the graphical editor. A procedure is a database object that contains a set of SQL statements that perform a specific task. A procedure can be created or edited with the SQL editor, but not with the graphical editor. A procedure can also be exposed as a CDS artifact, but it cannot be specified with the graphical editor. References:

* SAP HANA Platform, SAP HANA Modeling Guide for SAP HANA Web Workbench, Core Data Services

* SAP HANA Platform, SAP HANA Developer Guide for SAP HANA Web IDE, Developing Core Data Services Models, Using the Graphical Editor

QUESTION 29

A user is dropped with the “cascade” option. The user schema also contains objects owned by other users, or on which other users have privileges. What happens to the objects? There are 2 correct answers to this question.
* The objects on which other users have privileges are NOT dropped.
* The objects on which other users have privileges are dropped.

* The objects owned by other users are NOT dropped.
* The objects owned by other users are dropped.

When a user is dropped with the “cascade” option, the user and all the objects owned by the user are deleted from the database. However, the objects that are owned by other users, or on which other users have privileges, are not affected by the drop command. The privileges granted to or by the dropped user are also revoked automatically. Therefore, the objects on which other users have privileges are not dropped, and the objects owned by other users are not dropped12.

The other options are not correct because they are not the consequences of dropping a user with the “cascade” option. The objects on which other users have privileges are not dropped, because they are not owned by the dropped user, and they may still be needed by the other users. The objects owned by other users are not dropped, because they are not related to the dropped user, and they may have dependencies or references to other objects. References:

* SAP HANA Platform, SAP HANA SQL and System Views Reference, SQL Reference Manual, SQL Statements, DROP USER

* SAP HANA Platform, SAP HANA Administration Guide, Security, User Management, Dropping Users

QUESTION 30

What key words are contained in the application descriptor file (xs-app.json)? There are 2 correct answers to this question.

* routes
* role-templates
* tenant-mode
* authentication Method

The application descriptor file (xs-app.json) is a JSON file that defines the routing and authentication configuration for an HTML5 module in a multi-target application (MTA) project. It is located in the root folder of the HTML5 module and is used by the managed application router to dispatch requests to the appropriate destinations and to authenticate users. Some of the key words that are contained in the xs-app.json file are:

* routes: This is an array of objects that specify the rules for forwarding requests to the back-end microservices or destinations. Each route object has a source property that defines a regular expression to match the request path, and a destination property that defines the name of the destination to which the request should be forwarded. Optionally, a route object can also have other properties, such as authentication, csrfProtection, service, or scope, to further configure the routing behavior.

* authenticationMethod: This is a property that defines the authentication method to be used for the HTML5 module. It can have one of the following values: route, which means that the authentication method is defined by the route configuration; none, which means that no authentication is required; or approuter, which means that the authentication is delegated to the approuter service. The authenticationMethod property can be set at the global level for the entire HTML5 module, or at the route level for a specific route.

The following key words are not contained in the xs-app.json file, but in other files related to the MTA project:

* role-templates: This is an array of objects that define the roles and their corresponding scopes and attributes for the application. The role-templates are defined in the xs-security.json file, which is the application security descriptor file that specifies the security configuration for the application. The xs-security.json file is located in the root folder of the db module and is used by the User Account and Authentication (UAA) service to manage the authorization and trust management for the application.

* tenant-mode: This is a property that defines the tenant mode for the application. It can have one of the following values: dedicated, which means that the application runs in a dedicated schema for each tenant; shared, which means that the application runs in a shared schema for all tenants; or mixed, which means that the application runs in a mixed mode depending on the tenant context. The tenant-mode property is defined in the mta.yaml file, which is the deployment descriptor file that specifies the metadata and dependencies for the MTA project. The mta.yaml file is located in the root folder of the MTA project and is used by the Cloud

Foundry environment to deploy the application.

References:

* [SAP HANA Deployment Infrastructure Reference], Chapter 5: HDI with XS Advanced, Section 5.1:

Developing with the SAP Web IDE for SAP HANA, Subsection 5.1.2: Configure Application Routing (xs-app.json), pp. 101-104.

* [SAP HANA Platform Documentation], SAP HANA Developer Guide for SAP HANA XS Advanced Model, Chapter 4: Developing HTML5 Applications, Section 4.1: Developing HTML5 Applications Using SAP Web IDE for SAP HANA, Subsection 4.1.3: Configure Application Routing (xs-app.json), pp. 77-80.

QUESTION 31

Which OData capacity do you use when you need to restrict the number or selection of exposed columns?

Please choose the correct answer.
* Parameter entity sets
* Aggregation
* Key specification
* Property projection

Property projection is an OData capability that allows you to restrict the number or selection of exposed columns in an OData service. Property projection is achieved by using the $select query option, which specifies a subset of properties to be included in the response. The $select query option can be applied to a single entity, a collection of entities, or a complex type. Property projection can be used to reduce the payload size and improve the performance of the OData service.

For example, suppose you have an OData service that exposes a Products entity set with the following properties: ID, Name, Category, Price, and Description. If you want to restrict the number or selection of exposed columns to only ID and Name, you can use the $select query option as follows:

GET /Products?$select=ID,Name

The result is: { “@odata.context”: “$metadata#Products(ID,Name)”, “value”: [{ “ID”: 1, “Name”: “Laptop” }, { “ID”: 2, “Name”: “Mouse” }, { “ID”: 3, “Name”: “Keyboard” }] }

The following OData capabilities are not used to restrict the number or selection of exposed columns, but for other purposes:

* Parameter entity sets: Parameter entity sets are a way to define entity sets that require one or more parameters to be specified in the request. Parameter entity sets can be used to implement function imports or actions that return a collection of entities. Parameter entity sets can also be used to filter or sort the results based on the parameters.

* Aggregation: Aggregation is a way to apply aggregate functions, such as sum, count, min, max, or average, to the properties of an entity set or a complex type. Aggregation can be used to perform calculations or analysis on the data. Aggregation is achieved by using the $apply query option, which specifies a transformation pipeline with various operators, such as groupby, aggregate, filter, or order by.

* Key specification: Key specification is a way to define the key properties of an entity type, which uniquely identify an entity instance within an entity set. Key specification is part of the entity type definition in the metadata document of the OData service. Key specification can be used to retrieve a single entity by its key values.

References:

* [OData Version 4.0 Part 2: URL Conventions], Section 5.1.1: System Query Option $select, pp. 10-11.

* [OData Version 4.0 Part 3: Common Schema Definition Language (CSDL)], Section 13: Entity Model, Subsection 13.2: Entity Sets, pp. 88-89.

QUESTION 32

You need to install SAP HANA 2.0, express edition to develop a native SAP HANA application. Which of the following deployment options do you have?

There are 2 correct answers to this question.
* Installation on Windows Server
* Installation on Mac OS
* Installation on Linux OS
* Usage of virtual machine on Microsoft Windows

SAP HANA 2.0, express edition is a streamlined version of SAP HANA that can run on laptops and other resource-constrained hosts. It supports native SAP HANA application development and can be installed on Linux OS or used as a virtual machine on Microsoft Windows. Installation on Windows Server or Mac OS is not supported by SAP HANA 2.0, express edition. References:

* SAP HANA 2.0 SPS06 – Application Development for SAP HANA1, Section 1.1, p. 5

* SAP HANA, express edition – Installation Guide, Section 1.1, p. 7

* SAP HANA, express edition – FAQ, Question 1

QUESTION 33

What are the nodes where filter expressions can be used in a calculation view? There are 2 correct answers to this question.
* Aggregation
* Star join
* Union
* Rank

QUESTION 34

What are the issue categories that SQL Script Code Analyser scans for? There are 3 correct answers to this question.
* Performance
* Privileges
* Security
* Business logic
* Consistency

QUESTION 35

Which of the following elements can be part of the UI5 application’s index.html file? There are 2 correct answers to this question
* Framework Reference

* Bootstrap
* Backend Connection
* UI-Area

The index.html file is the entry point of the UI5 application. It contains the following elements:

* Framework Reference: This element specifies the location of the UI5 resources, such as libraries, themes, and controls. It can be done by using the data-sap-ui-resourceroots attribute or the sap.ui.localResources() function.

* Bootstrap: This element initializes the UI5 framework and loads the required libraries and components.

It can be done by using the script tag with the src attribute pointing to the sap-ui-core.js file and the data-sap-ui-xx attributes to configure the bootstrap parameters.

* Backend Connection: This element is optional and depends on the application logic. It can be used to establish a connection to the backend system, such as an OData service, by using the data-sap-ui-bindingSyntax attribute or the sap.ui.model.odata.v2.ODataModel() constructor.

* UI-Area: This element defines the HTML element where the UI5 content will be rendered. It can be done by using the data-sap-ui-oninit attribute or the sap.ui.getCore().attachInit() function to register a callback function that creates and places the UI5 root component or view.

References:

* SAP HANA 2.0 SPS06 – Application Development for SAP HANA1, Section 5.1.1, p. 131-132

* SAP HANA 2.0 SPS06 – Developing Web Apps with SAPUI52, Section 2.1, p. 17-20

* SAP HANA 2.0 SPS06 – Developing Web Apps with SAPUI52, Section 3.1, p. 37-40

* SAP HANA 2.0 SPS06 – Developing Web Apps with SAPUI52, Section 4.1, p. 59-62

* SAP HANA 2.0 SPS06 – Developing Web Apps with SAPUI52, Section 5.1, p. 81-84

QUESTION 36

Which configuration file indicates the program that is executed upon start up when you run the Node.js module? Please choose the correct answer.

* package.json
* project.json
* server.js
* mta.yaml

The package.json file is the main configuration file for a Node.js module. It contains various metadata about the module, such as its name, version, dependencies, scripts, and main entry point. The main entry point is the program that is executed when the module is run or required by another module. By default, the main entry point is index.js, but it can be changed by specifying the main property in the package.json file12. For example, if the main property is set to “server.js”, then the server.js file will be executed upon start up when the module is run.

The other options are not correct because:

* project.json: This is not a standard configuration file for Node.js modules. It may be used by some frameworks or tools, but it is

not recognized by Node.js itself3.

* server.js: This is a common name for a file that contains the server logic for a Node.js module, but it is not a configuration file. It may be the main entry point for the module, but only if it is specified in the package.json file4.

* mta.yaml: This is a configuration file for multi-target applications (MTAs) in SAP HANA XS advanced model. It defines the modules, resources, and dependencies of an MTA, but it is not specific to Node.js modules5.

References: 1: package.json | Node.js v17.0.1 Documentation 2: Debugging in Visual Studio Code 3: How to run Node.js module initialization properly – Stack Overflow 4: Running Node.js Apps with PM2 (Complete Guide) – Better Stack 5: SAP HANA Developer Guide for SAP HANA XS Advanced Model (SAP Web IDE)

QUESTION 37

Which keywords do you use to define an OData association? There are 3 correct answers to this question.
* JOIN
* DEPENDENT
* UNION
* MULTIPLICITY
* PRINCIPAL
OData associations are used to define the relationships between two or more entity types in an OData service.

Associations can be simple or complex, depending on whether the relationship information is stored in one of the participating entities or in a separate association table. Associations are composed of two ends, each of which has a role name, a multiplicity, and a set of properties that form the referential constraint. The keywords that are used to define an OData association are:

* DEPENDENT: This keyword is used to specify the role name of the dependent end of the association, which is the entity type that contains the foreign key properties that refer to the principal entity type. For example, in the association Customer_Orders, the entity type Orders is the dependent end, as it contains the property CustomerID that refers to the entity type Customers.

* MULTIPLICITY: This keyword is used to specify the cardinality of each end of the association, which indicates how many instances of one entity type can be related to one instance of another entity type.

The possible values for multiplicity are: 0…1 (zero or one), 1 (exactly one), or * (many). For example, in the association Customer_Orders, the multiplicity of the principal end Customers is 1, meaning that each customer can have only one instance, and the multiplicity of the dependent end Orders is *, meaning that each customer can have many orders.

* PRINCIPAL: This keyword is used to specify the role name of the principal end of the association, which is the entity type that is referenced by the foreign key properties of the dependent entity type. For example, in the association Customer_Orders, the entity type Customers is the principal end, as it is referenced by the property CustomerID of the entity type Orders.

The following keywords are not used to define an OData association, but for other purposes:

* JOIN: This keyword is used to specify the join condition between two entity sets in a query expression, which is used to retrieve data from multiple sources. For example, the query expression Customers?$expand=Orders($filter=Status eq ‘Open’) uses a join condition to filter the orders that have the status ‘Open’ for each customer.

* UNION: This keyword is used to specify the union operation between two query expressions, which is used to combine the results of both queries into one result set. For example, the query expression Customers?$filter=Country eq ‘US’ union Customers?$filter=Country eq ‘CA’ uses a union operation to get the customers that are from either the US or

Canada.

QUESTION 38

What do you reference in a role template? Please choose the correct answer.
* Role Collections
* Roles
* Routes
* Scopes

A role template is a JSON file that defines the permissions and scopes for a specific role in a multi-target application. A scope is a string that represents a specific access right or privilege for a resource or service. For example, a scope can be used to grant read or write access to a database table or an OData service. A role template references the scopes that are required for the role to function properly. A role collection is a group of roles that can be assigned to a user or a group. A role is a collection of permissions and scopes that define what a user can do in an application. A route is a mapping between a URL path and a destination service in a multi-target application. A route is not related to a role template. References:

* SAP HANA 2.0 SPS06 – Application Development for SAP HANA1, Section 6.1.1, p. 147-148

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.1, p. 25-26

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.2, p. 27-28

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.3, p. 29-30

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.4, p. 31-32

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.5, p. 33-34

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.6, p. 35-36

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.7, p. 37-38

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.8, p. 39-40

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.9, p. 41-42

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.10, p. 43-44

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.11, p. 45-46

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.12, p. 47-48

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.13, p. 49-50

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.14, p. 51-52

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.15, p. 53-54

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.16, p. 55-56

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.17, p. 57-58

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.18, p. 59-60

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.19, p. 61-62

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.20, p. 63-64

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.21, p. 65-66

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.22, p. 67-68

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.23, p. 69-70

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.24, p. 71-72

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.25, p. 73-74

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.26, p. 75-76

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.27, p. 77-78

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.28, p. 79-80

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.29, p. 81-82

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.30, p. 83-84

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.31, p. 85-86

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.32, p. 87-88

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.33, p. 89-90

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.34, p. 91-92

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.35, p. 93-94

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.36, p. 95-96

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.37, p. 97-98

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.38, p. 99-100

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.39, p. 101-102

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.40, p. 103-104

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.41, p. 105-106

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.42, p. 107-108

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.43, p. 109-110

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.44, p. 111-112

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.45, p. 113-114

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.46, p. 115-116

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.47, p. 117-118

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.48, p. 119-120

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.49, p. 121-122

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.50, p. 123-124

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.51, p. 125-126

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment2, Section 3.1.52, p. 127-128

* SAP HANA 2.0 SPS06 – Cloud Foundry Environment

QUESTION 39

An OData service contains an entity set called Products. Which resource path do you add to the OData service URL to view all available products?

Please choose the correct answer.
* /Products/$metadata
* /Products/’
* /Products
* /Products/ALL

According to the SAP HANA Developer Guide, the resource path of an OData service URL identifies the entity or entity set that is requested. The resource path consists of the service root URI followed by a slash (/) and the name of the entity set. For example, to view all available products in the Products entity set, the resource path is /Products. The other options are incorrect, because:

* /Products/$metadata is not a valid resource path, but a system query option that returns the metadata document of the OData service.

* /Products/’ is not a valid resource path, but a syntax error. The single quotation mark is not needed after the entity set name.

* /Products/ALL is not a valid resource path, but a filter expression that can be used as a query option.

The filter expression must be preceded by a question mark (?) and the $filter system query option. For example, /Products?$filter=ALL.

References: SAP HANA Developer Guide, Chapter 6, Section 6.4.2, page 2111.

QUESTION 40

What are the nodes where filter expressions can be used in a calculation view? There are 2 correct answers to this question.
* Aggregation
* Star join
* Union
* Rank

Filter expressions can be used in a calculation view to restrict or modify the data that is displayed or processed by the view. Filter expressions can be used in the following nodes of a calculation view12:

* Aggregation: An aggregation node is a node that applies aggregation functions, such as sum, count, or average, to the data that is passed from the previous node. Filter expressions can be used in an aggregation node to filter the data before or after the aggregation, or to define the aggregation level or the measure attributes. For example, you can use a filter expression to show only the sales data for a specific region or product category, or to calculate the average revenue per customer.

* Union: A union node is a node that combines the data from two or more nodes that have the same structure and data types. Filter expressions can be used in a union node to filter the data from each input node, or to filter the data from the output node. For example, you can use a filter expression to exclude the duplicate rows from the union result, or to show only the data that matches a certain condition from each input node.

The other options are not correct because filter expressions cannot be used in these nodes of a calculation view. A star join node is a node that joins a fact table with one or more dimension tables, based on the common key attributes. A star join node does not support filter expressions, but it supports input parameters, which are variables that can be used to filter the data at runtime. A rank node is a node that ranks the data according to a specified order and criteria. A rank node does not support filter expressions, but it supports rank filters, which are conditions that can be used to limit the number or percentage of rows in the rank result.

References:

* SAP HANA Platform, SAP HANA Modeling Guide for SAP HANA Web Workbench, Calculation Views

* SAP HANA Platform, SAP HANA Developer Guide for SAP HANA Web IDE, Developing Database Modules, Developing Calculation Views, Using Filter Expressions

QUESTION 41

What characterizes the Node.js environment? There are 2 correct answers to this question.
* It uses a synchronous programming model.
* It is a client-side design-time environment for JavaScript.
* It is single-threaded.
* It is built on Google’s V8 JavaScript engine.

According to the SAP Web IDE for SAP HANA Developer Guide, the Node.js environment is characterized by the following features:

* It is single-threaded: Node.js uses a single thread to handle multiple concurrent requests, instead of creating a new thread for each request. This reduces the memory and CPU overhead, and enables high scalability and performance. However, it also means that any blocking or long-running operation can affect the responsiveness of the entire application. Therefore, Node.js relies on asynchronous and non-blocking I/O operations, callbacks, promises, and events to handle concurrency and avoid blocking the main thread.

* It is built on Google’s V8 JavaScript engine: Node.js uses the V8 engine to execute JavaScript code.

The V8 engine is a fast and powerful engine that compiles JavaScript code into native machine code, and supports the latest ECMAScript standards and features. The V8 engine also provides access to low-level system resources, such as files, network, and processes, through the Node.js API.

The other options are incorrect, because:

* Node.js does not use a synchronous programming model, but an asynchronous and event-driven programming model. A synchronous programming model means that each operation blocks the execution until it is completed, and the next operation can only start after the previous one is finished.

An asynchronous programming model means that each operation can start without waiting for the previous one to finish, and the execution can continue with other tasks while the operation is in progress.

An event-driven programming model means that each operation can trigger an event when it is completed, and the event can invoke a callback function that handles the result or the error of the operation.

* Node.js is not a client-side design-time environment for JavaScript, but a server-side run-time environment for JavaScript. A client-side design-time environment for JavaScript means that the JavaScript code is written and executed in the browser, and it can manipulate the HTML document and interact with the user interface. A server-side run-time environment for JavaScript means that the JavaScript code is written and executed on the server, and it can handle HTTP requests and responses, communicate with databases, and perform business logic.

References: SAP Web IDE for SAP HANA Developer Guide, Chapter 6, Section 6.4.2, page 2111.