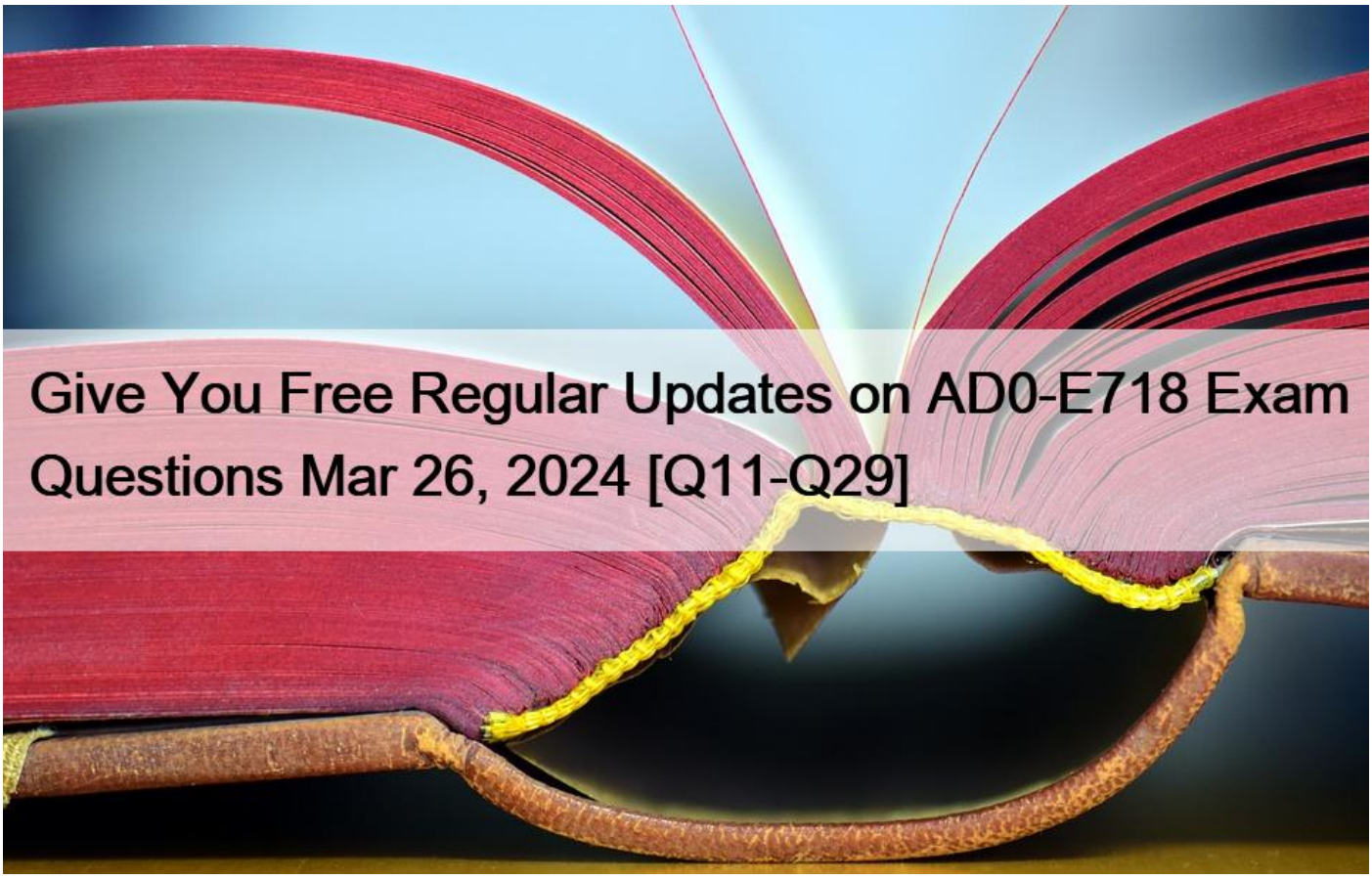
Give You Free Regular Updates on AD0-E718 Exam Questions Mar 26, 2024 [Q11-Q29



Give You Free Regular Updates on AD0-E718 Exam Questions Mar 26, 2024
Achieve the AD0-E718 Exam Best Results with Help from Adobe Certified Experts

The Adobe AD0-E718 exam is intended for individuals who have experience working with Adobe Commerce and have a deep understanding of its architecture and functionality. This includes architects, developers, and technical leads who are responsible for designing, implementing, and maintaining Adobe Commerce solutions.

NO.11 An Adobe Commerce Architect is creating a new GraphQL API mutation to alter the process of adding configurable products to the cart. The mutation accepts configurable product ID. If the given product has only one variant, then the mutation should add this variant to the cart and return not nullable cart type. If the configurable product has more variants, then the mutation should return not nullable conf igurableProduct type.

The mutation declaration looks as follows:

```
type Mutation {
    addConfigurableToCart(product_id: Int!): AddToCartOutput!
        @resolver(class: "Vendor\\MyModule\\Model\\Resolver\\AddConfigurableToCart")
}
```

How should the Adobe Commerce Architect declare output of this mutation?

*
```
interface AddToCartOutput
@typeResolver(class: "Vendor\\MyModule\\Model\\Resolver\\AddToCartOutputTypeResolver") {
}

type ConfigurableProduct implements AddToCartOutput {
}

type Cart implements AddToCartOutput {
}
```

premium.validexam.com

*
```
union AddToCartOutput
@typeResolver(class: "Vendor\\MyModule\\Model\\Resolver\\AddToCartOutputTypeResolver")
= ConfigurableProduct | Cart
```

*
```
type AddToCartOutput {
    product: ConfigurableProduct
    cart: Cart
}
```

NO.12 An Architect needs to review a custom product feed export module that a developer created for a merchant. During final testing before the solution is deployed, the product feed output is verified as correct. All unit and integration tests for code pass.

However, once the solution is deployed to production, the product price values in the feed are incorrect for several products. The products with incorrect data are all currently part of a content staging campaign where their prices have been reduced.

What did the developer do incorrectly that caused the feed output to be incorrect for products in the content staging campaign?
* The developer forgot to use the getContentStagingValue() method to retrieve the active campaign value of the product data
* The developer retrieved product data directly from the database using the entity_id column rather than a collection or repository.
* The developer did not check for an active content staging campaign and emulates the campaign state when retrieving product data.

Based on the given scenario, it is likely that option C – “The developer did not check for an active content staging campaign and emulates the campaign state when retrieving product data” – is the correct answer. It appears that the developer did not take into account the active content staging campaign and did not properly adjust the product data when generating the product feed. As a result, the feed output is incorrect for products that are part of the staging campaign and have their prices reduced. The correct solution would be to check for an active content staging campaign and properly adjust the product data to reflect the campaign state.

NO.13 An Adobe Commerce Architect needs to ensure zero downtime during the deployment process of Adobe Commerce on-premises. Which two steps should the Architect follow? (Choose two.)
* Run bin/magento setup:upgrade –keep-generated to Upgrade database
* Run bin/magento setup:upgrade -dry-run=true to upgrade database
* Rim bin/magento setup:upgrade –convert-old-scripts=true to Upgrade database
* Enable config flag under developer/zere _down_time/enabled
* Enable config flag under deployment/blue_ green/enabled

1. Running bin/magento setup:upgrade -dry-run=true allows you to check the upgrade scripts without applying any changes to the database. This can help you identify any potential issues before the actual upgrade. E. Enabling config flag under deployment/blue_ green/enabled allows you to use the blue-green deployment strategy, which creates a copy of the production environment and

switches traffic between the two environments after testing the new version. This can help you achieve zero downtime during the deployment process. Reference: https://devdocs.magento.com/guides/v2.4/comp-mgr/cli/cli-upgrade.html#upgrade-cli-dryrun https://devdocs.magento.com/cloud/live/stage-prod-migrate-prereq.html#blue-green-deployment

NO.14 An Adobe Commerce store owner sets up a custom customer attribute “my.attribute” (type int).

An Architect needs to display customer-specific content on the home page to Customers with “my.attribute” greater than 3. The website is running Full Page Cache.

Using best practices, which two steps should the Architect take to implement these requirements? (Choose two.)
* Use customer-data JS library to retrieve “my.attribute” value
* Add a new context value of “my.attribute” to MagentoFrameworkAppHttpContext
* Add a custom block and a phtml template with the content to the cmsjndexjndex.xml layout
* Create a Customer Segment and use “my.attribute” in the conditions
* Add a dynamic block with the content to the Home Page
Explanation

https://docs.magento.com/user-guide/v2.3/stores/attributes-customer.html displaying custom customer attributes on cached pages using best practices involves several steps, such as:

* Creating a custom block and a phtml template with the content to display

* Adding the custom block to the layout file of the page where it should appear

* Creating a section.xml file to declare a new section for the custom attribute

* Creating a plugin for MagentoCustomerCustomerDataSectionPoolInterface to add the custom attribute value to the section data

* Using customer-data JS library to retrieve and display the custom attribute value in the phtml template

NO.15 An Adobe Commerce store owner sets up a custom customer attribute “my.attribute” (type int).

An Architect needs to display customer-specific content on the home page to Customers with “my.attribute” greater than 3. The website is running Full Page Cache.

Using best practices, which two steps should the Architect take to implement these requirements? (Choose two.)
* Use customer-data JS library to retrieve “my.attribute” value
* Add a new context value of “my.attribute” to MagentoFrameworkAppHttpContext
* Add a custom block and a phtml template with the content to the cmsjndexjndex.xml layout
* Create a Customer Segment and use “my.attribute” in the conditions
* Add a dynamic block with the content to the Home Page
To display customer-specific content on the home page with Full Page Cache enabled, the Architect needs to add a new context value of “my.attribute” to MagentoFrameworkAppHttpContext. This will allow the cache to vary based on the value of “my.attribute”. Then, the Architect needs to add a dynamic block with the content to the Home Page. A dynamic block is a type of content block that can be personalized based on customer segments or other conditions. Reference: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/cache/page-caching/public-content.html https://docs.magento.com/user-guide/marketing/page-builder-add-content-block.html

NO.16 An Adobe Commerce Architect is troubleshooting an issue on an Adobe Commerce Cloud project that is not yet live.

The developers migrate the Staging Database to Production in readiness to Go Live. However, when the developers test their Product Import feature, the new products do not appear on the frontend.

The developers suspect the Varnish Cache is not being cleared. Staging seems to work as expected. Production was working before the database migration.

What is the likely cause?
* A deployment should have been done on Production to initialize Fastly caching.
* The site URLs in the Production Database are the URLs of the Staging Instance and must be updated.
* The Fastly credentials in the Production Database are incorrect.
Explanation

The likely cause of the issue is that a deployment should have been done on Production to initialize Fastly caching. This is because when the database is migrated from Staging to Production, any changes made to the Staging Database will not be reflected in the Production environment until a deployment is made. This includes any changes made to the Varnish Cache, which needs to be cleared in order for the new products to appear on the frontend.

NO.17 An Architect is reviewing a custom module that is logging customer activity data on storefront using observers. The client reports that logs were recorded while the client was previewing storefront catalog pages from Admin Panel for a future scheduled campaign, using Adobe Commerce staging preview functionality.

What should the Architect check first to address this issue9
* The plugin for the public method isAllowedObserver() from MagentoStagingModelEventManager that alters the return value
* The logging observers being copied from etcevents.xml to etcadminhtmlevents.xml with the attribute disabled=”true”
* The list of logging observers in bannedObservers parameter of

MagentostagingModelEventManagertype in di.xml
Explanation

It will allow you to exclude logging observers from being executed during staging preview functionality by adding them to bannedObservers parameter of MagentostagingModelEventManager type in di.xml file.

This will prevent customer activity data from being logged while previewing storefront catalog pages from Admin Panel for a future scheduled campaign.

NO.18 An Adobe Commerce Architect designs a data flow that contains a new product type with its own custom pricing logic to meet a merchant requirement.

Which three developments are valid when reviewing the implementation? (Choose three.)
* Content of the etc/product_types.xml file
* Hydrator for attributes belonging to the new product type
* Custom type model extended from the abstract Product Type model
* A new class with custom pricing logic, extending the abstract Product model class
* Data patch to register the new product type
* New price model extending MagentoCatalogModelProductTypePrice
To create a new product type, you need to extend the abstract Product Type model and register it using a data patch. You also need to create a new price model that extends MagentoCatalogModelProductTypePrice and implements the custom pricing logic.

NO.19 An Adobe Commerce Architect notices that the product price index takes too long to execute. The store is configured with

multiple websites and dozens of customer groups.

Which two ways can the Architect shorten the full price index execution time? (Choose two.)
* Enable price index customer group merging for products without tier prices
* Set Customer Share Customer Accounts Option to Global
* Edit customer groups to exclude websites that they are not using
* Set MaGE_INDEXER_THREADS_COUNT environment variable to enable parallel mode
* Move catalog price_index indexer to another custom indexer group

Explanation

The two best ways the Architect can shorten the full price index execution time are Option A. Enable price index customer group merging for products without tier prices, and Option D. Set MaGEINDEXERTHREADS_COUNT environment variable to enable parallel mode. Enablingcustomer group merging will help reduce the number of customer groups that need to be processed, while setting the environment variable will allow the indexer to use multiple threads and run in parallel mode, thus reducing the overall execution time.

NO.20 An Architect is working to implement Adobe Commerce into a pre-built ecosystem in a company.

Communication between different company domains uses event-driven design and is driven via AMQP protocol with using RabbitMQ.

The Architect needs to establish the data flow between the ERP system and Adobe Commerce.

The ERP system stores only customer data excluding customer addresses.

The role of Adobe Commerce is to provide Customer Address data to the enterprise ecosystem.

Primary Customer data should not be changed from Adobe Commerce side; it should only be updated by messages data from ERP.

Which three AMQP configurations should be considered to meet these requirements? (Choose three.)
* Create a queue_consumer.xml and communction.xml configuration files for Customer data messages
* Create a queue_publisher.xml configuration file for Customer data messages
* Create a nueue_publisher.xml configuration file for Customer Address messages
* Create a queue_topology.xml configuration file for Customer Address messages
* Create a queue_topology.xml configuration file for Customer data messages
* Create a queue_customer.xml and communication.xml configuration files for Customer Address messages

The Architect should consider three AMQP configurations to meet these requirements: A) Create a queue_consumer.xml and communication.xml configuration files for Customer data messages. These files will define the consumer and the topic for receiving customer data messages from the ERP system and updating the customer data in Adobe Commerce accordingly. B) Create a queue_publisher.xml configuration file for Customer Address messages. This file will define the publisher and the topic for sending customer address messages from Adobe Commerce to the enterprise ecosystem. C) Create a queue_topology.xml configuration file for Customer Address messages. This file will define the exchange, binding and queue for routing customer address messages to the appropriate destination. Option B is incorrect because creating a queue_publisher.xml configuration file for Customer data messages will not meet the requirement of not changing the primary customer data from Adobe Commerce side. Option E is incorrect because creating a queue_topology.xml configuration file for Customer data messages is not necessary, as Adobe Commerce only needs to consume these messages, not publish them. Option F is incorrect because creating a queue_consumer.xml and communication.xml configuration files for Customer Address messages is not necessary, as Adobe Commerce only needs to publish these messages, not consume them. Reference: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/message-queues/config-mq.html

NO.21 A merchant notices that product price changes do not update on the storefront.

The index management page in the Adobe Commerce Admin Panel shows the following:

* All indexes are set to ‘update by schedule’

* Their status is ‘ready’

* There are no items in the backlog

* The indexes were last updated 1 minute ago

A developer verifies that updating and saving product prices adds the relevant product IDs into the catalog_product_price_cl changelog table.

Which two steps should the Architect recommend to the developer to resolve this issue? (Choose two.)
* Invalidate the catalog_product_price indexer in the Adobe Commerce Admin Panel so that it is fully reindexed next time the cron runs.
* Manually reindex the catalog_product_price index from the Command line:binmagentor indexer:reindex catalog_product_price.
* Make sure that no custom or third-party modules modify the changelog and indexing process.
* Make sure that the version_id for the price indexer in the mview_state table is not higher than the last entry for the same column in the changelog table and re-synchronize.
* Reduce the frequency of the cron job to 5 minutes so the items have more time to process.

NO.22 An Adobe Commerce system is configured to run in a multi-tier architecture that includes:

* A cache server with Varnish installed

* A backend web server with Adobe Commerce installed

* A database server with MySQL installed

When an Adobe Commerce Architect tries to clean the cache from the Store Admin by using the “Flush Magento Cache” in Cache Management, the Full Page Cache does not clear.

Which two steps should the Architect take to make the Full Page Cache work properly? (Choose two.)
* Set the backend destination host to the frontend server’s address in the Store Admin Stores > Configuration > Advanced > System > Full Page Cache > Varnish Configuration > Backend Host
* Set the backend port destination to the frontend server’s Varnish port in the Store Admin Stores > Configuration > Advanced > System > Full Page Cache > Varnish Configuration > Backend Port
* Set the cache type to “Varnish Caching” in the Store Admin.

Stores > Configuration > Advanced > System > Full Page Cache > Caching Application
* Use “Flush Cache Storage” instead of “Flush Magento Cache”
* Set the cache destination host using magento CLI bin/magento setup:config:set –http-cache-hosts<cache_server>:<varrnish port>
To use Varnish as the full page cache, the cache type must be set to “Varnish Caching” in the Store Admin. This will enable the “Flush Magento Cache” button to send a purge request to Varnish. Additionally, the cache destination host must be specified using the magento CLI command to tell Magento which Varnish servers to purge. Reference: https://devdocs.magento.com/guides/v2.4/config-guide/varnish/config-varnish.html

NO.23 An Adobe Commerce Architect needs to ensure zero downtime during the deployment process of Adobe Commerce on-premises. Which two steps should the Architect follow? (Choose two.)
* Run bin/magento setup:upgrade –keep-generated to Upgrade database
* Run bin/magento setup:upgrade -dry-run=true to upgrade database
* Rim bin/magento setup:upgrade –convert-old-scripts=true to Upgrade database
* Enable config flag under developer/zere _down_time/enabled
* Enable config flag under deployment/blue_ green/enabled

To ensure zero downtime during the deployment process of Magento 2 on-premises, the Architect should follow two steps:

Run bin/magento setup:upgrade –keep-generated to upgrade database. This will skip the regeneration of static content and code files during the upgrade process, which can take a long time and cause downtime. The static content and code files should be generated separately before or after the upgrade process.

Enable config flag under deployment/blue_green/enabled. This will enable the blue-green deployment strategy, which creates a copy of the current production environment (blue) and deploys the new code to it (green). Then, it switches the traffic from the blue environment to the green environment without any downtime. This option can be enabled by adding a line like deployment/blue_green/enabled: true to the .magento.env.yaml file.

NO.24 In a custom module, an Architect wants to define a new xml configuration file. The module should be able to read all the xml configuration files declared in the system, merge them together, and use their values in PHP class.

Which two steps should the Architect make to meet this requirement? (Choose two.)
* Write a plugin for MagentoFrameworkConfigData::get() and read the custom xml files
* Append the custom xml file name in “MagentoConfigModelConfigStructureReader” in di.xml
* Create a Data class that implements “MagentoFrameworkConfigData’
* Inject a “reader” dependency for “MagentoFrameworkConfigData” in di.xml
* Make a Reader class that implements “MagentoFrameworkConfigReaderFilesystem”

Based on web searches, it seems that Magento uses different classes and interfaces to interact with configuration files, such as Data, Reader, and Converter12.

According to the documentation1, Data is a class that provides access to configuration data using a scope. Reader is an interface that reads configuration data from XML files. Converter is an interface that converts XML data into an array representation.

Based on these definitions, I would say that two possible steps that the Architect should make to meet the requirement are:

1. Create a Data class that implements “MagentoFrameworkConfigData”

2. Make a Reader class that implements “MagentoFrameworkConfigReaderFilesystem” These steps would allow the custom module to read all the XML configuration files declared in the system, merge them together, and use their values in PHP class.

The Architect should make two steps to meet this requirement: C) Create a Data class that implements “MagentoFrameworkConfigData”. This class will be responsible for reading and merging the custom xml configuration files and providing access to their values. The Data class should extend MagentoFrameworkConfigData and use the constructor to inject the Reader class and the cache type. E) Make a Reader class that implements “MagentoFrameworkConfigReaderFilesystem”. This class will be responsible for loading and validating the custom xml configuration files from different modules. The Reader class should extend MagentoFrameworkConfigReaderFilesystem and use the constructor to specify the file name, schema file, and validation state of the custom xml configuration files. Option A is incorrect because writing a plugin for MagentoFrameworkConfigData::get() will not define a new xml configuration file, but rather modify the existing one. Option B is incorrect because appending the custom xml file name in

“MagentoConfigModelConfigStructureReader” in di.xml will not define a new xml configuration file, but rather add it to the system configuration structure. Option D is incorrect because injecting a “reader” dependency for “MagentoFrameworkConfigData” in di.xml will not define a new xml configuration file, but rather use an existing one. Reference: https://devdocs.magento.com/guides/v2.4/extension-dev-guide/build/XSD-XML-validation.html

NO.25 An Adobe Commerce Architect is setting up a Development environment for an on-premises project that will be used for developers to specifically test functionality, not performance, before being passed to the Testing team.

The Magento application must run with the following requirements:

1. Errors should be logged and hidden from the user

2. Cache mode can only be changed from Command Line

3. Static files should be created dynamically and then cached

Which Application Mode is required to achieve this?
* Production Mode
* Developer Mode
* Default Mode

Developer Mode is the mode best suited to achieve the requirements set out by the Adobe Commerce Architect. In Developer Mode, errors are logged and hidden from the user, and the cache mode can only be changed from the command line. Additionally, static files are created dynamically and then cached, which is the desired behavior for this project.

Developer Mode is required to achieve the requirements. This is because developer mode enables the following features: Errors are logged and hidden from the user; cache mode can only be changed from command line; static files are created dynamically and then cached. See Application modes in the Adobe Commerce Help Center1. Reference: https://experienceleague.adobe.com/docs/commerce-operations/configuration-guide/setup/application-modes.html?lang=en1

NO.26 An Adobe Commerce Architect creates a new functionality called Customs Fee, which adds a new total that applies to additional costs for handling customs clearance expenses. The extension allows specifying fee value for every website separately via the Adobe Commerce Configuration System.

The Architect plans to cover new functionality with integration tests. One test case needs to confirm if the total is calculated correctly on different websites.

How should the Architect make sure that test configuration data is added to test methods according to best practices?
* Override setuo () method, receive instance of MagentoTestFrameworkAppconfig, and specify value via setValue () method
* Specify @magentoconfigFixture annotations for the test methods in PHPDoc
* Create a fixture file to configure Adobe Commerce and specify it in test method PHPDoc using the

@magentoconfigFixture annotation
Explanation

The best practice for making sure that test configuration data is added to test methods is to use the @magentoconfigFixture annotation in the PHPDoc for the test methods. This will allow the Architect to specify a fixture file which will configure Adobe Commerce, and the test method will then be able to access these configuration values. Additionally, the Architect can also override the setUp() method and receive an instance of MagentoTestFrameworkAppconfig and specify the value via the setValue() method.

NO.27 An Adobe Commerce Architect is working on a scanner that will pull prices from multiple external product feeds. The

Architect has a list of vendors and decides to create new config file marketplacejeeds.xml.

Which three steps can the Architect take to ensure validation of the configuration files with unique validation rules for the individual and merged files? (Choose three.)
* Implement validation rules in the Converter class for the Config Reader
* Add the Uniform Resource Name to the XSD file in the config XML file.
* Provide schema to validate a merged file.
* Provide schema to validate an individual file.
* Create a class that implements MagentoFrameworkConfigDataInterface.
* Create validation rules in marketplace.schema.xsd.

Explanation

The three steps that the Architect can take to ensure validation of the configuration files with unique validation rules for the individual and merged files are: D. Provide schema to validate an individual file, F. Create validation rules in marketplace.schema.xsd, and E. Create a class that implements

MagentoFrameworkConfigDataInterface. By providing schema to validate individual files, creating validation rules in marketplace.schema.xsd, and creating a class that implements

MagentoFrameworkConfigDataInterface, the Architect can ensure that the configuration files are validated with unique validation rules for the individual and merged files.

NO.28 An Adobe Commerce Architect needs to create a new customer segment condition to enable admins to specify an ‘Average sales amount’ condition for certain segments.

The Architect develops the custom condition under vendorModuleModelSegmentconditionAverageSalesAmount with all of its requirements:

```php
<?php

declare(strict_types=1);

namespace Vendor\Module\Plugin;

use Magento\CustomerSegment\Model\Segment\Condition\Combine;

class AddNewChildOption
{
    public function afterGetNewChildSelectOptions(
        Combine $subject,
        array $result
    ): array {
        $conditions = [
                [
                    'value' => \Vendor\Module\Model\Segment\Condition\AverageSalesAmount::class,
                    'label' => __('Average sales amount'),
                ]
        ];

        return array_merge_recursive($result, $conditions);
    }
}
```

premium.validexam.com

```
"Class Magento\\CustomerSegment\\Model\\Segment\\Condition\\Vendor\\Module\\
Model\\Segment\\Condition\\AverageSalesAmount does not exist at
/var/www/vendor/magento/framework/Code/Reader/ClassReader.php"
```

This page was exported from - Valid Premium Exam
Export date: Fri Sep 20 0:55:47 2024 / +0000 GMT

Which two steps should the Architect complete to fix the problem? (Choose two.)

* Use a virtualType `<virtualType name="Magento\CustomerSegment\Model\Segment\Condition\AverageSalesAmount" type="Vendor\Module\Model\Segment\Condition\AverageSalesAmount"/>`

* Remove the trailing path `Magento\CustomerSegment\Model\Segment\Condition\` from the `$conditions value` attribute

* Use a preference `<pref type="Vendor\Module\M`

NO.29 An Architect agrees to improve company coding standards and discourage using Helper classes in the code by introducing a new check with PHPCS.

The Architect creates the following:

* A new composer package under the AwesomeAgencyCodingStandard namespace

* The ruleset. xml file extending the Magento 2 Coding Standard

What should the Architect do to implement the new code rule?

A)

B)

```
<rule ref="Magento2.Namespaces.ForbiddenNamespaces">
    <include-pattern>AwesomeAgency\*\Helper\*</include-pattern>
</rule>
```

C)

* Option A
* Option B
* Option C

To implement the new code rule, the Architect should create a new class that extends the PHP_CodeSnifferSniffsSniff interface and implements the register() and process() methods. The register() method should return an array of tokens that the rule applies to, such as T_STRING for class names. The process() method should contain the logic to check if the class name contains Helper and report an error if so. The Architect should also add a reference to the new class in the ruleset.xml file under the <rule> element with a ref attribute. This will enable PHPCS to use the new rule when checking the code.

Adobe AD0-E718 certification exam is designed for professionals who possess a strong understanding of Adobe Commerce architecture and are seeking to validate their expertise in the field. AD0-E718 exam is intended for those who have a comprehensive understanding of Adobe Commerce and are capable of designing and implementing complex eCommerce solutions.

Detailed New AD0-E718 Exam Questions for Concept Clearance: https://www.validexam.com/AD0-E718-latest-dumps.html]