Number: AD0-E722
Passing Score: 800
Time Limit: 120
File Version: 4.0

Exam Code: AD0-E722

Exam Name: Adobe Commerce Architect Master

Exam A

QUESTION 1
An existing Adobe Commerce website is moving to a headless implementation.
The existing website features an 'All Brands'' page, as well as individual pages for each brand. All brand-related pages are cached in Varnish using tags in the same manner as products and categories.
Two new GraphQL queries have been created to make this information available to the frontend for the new headless implementation:

```
type Query {
    brands (
        search: String @doc(description: "Search against brand names (partial matches)")
        pageSize: Int = 20 @doc(description: "Number of brand results to return")
        currentPage: Int = 1 @doc(description: "Page number to return")
    ) : BrandsResult
    @resolver(class: "ClientName\\ProductBrandsGraphQl\\Model\\Resolver\\Brands")
    brand (
        urlKey: String! @doc(description: "Match against brand url key")
    ) : Brand
    @resolver(class: "ClientName\\ProductBrandsGraphQl\\Model\\Resolver\\BrandByUrlKey")
}
```

During testing, the queries sometimes return out-of-date information. How should this problem be solved while maintaining performance?

A. Specify a @cacgecacheable(cacheable: false) directive for each GraphQL query, making sure that the data returned is not cached, and is up to date
B. Specify a $cache(cacheidentity: Path\\To\\identityclass) directive for each GraphQL query, corresponding to a class that adds cache tags for relevant brands and associated products
C. Each GraphQL query's resolver class should inject \Magento\GraphQlcache\Model\cacheableQuery and call setcachevalidity(true) on it as part of the resolver's resolve function.

Correct Answer: B
Section:
Explanation:
This solution ensures that the data returned by the GraphQL queries is up to date, while also maintaining performance. By specifying a $cache(cacheidentity: Path\To\identityclass) directive for each GraphQL query, the relevant brands and associated products will be added as cache tags.

QUESTION 2
An Adobe Commerce Architect is investigating a case where some EAV product attributes are no longer updated.
* The catalog is composed of 20.000 products with 100 attributes each.
* The product updates are run by recurring Adobe commerce imports that happen multiple times a day.
* The Architect finds an error in the logs that indicates an integrity constraint while trying to insert row with id 2147483647.
What is causing this error?

A. Magento framework uses INSERT on DUPLICATE, which leads to reaching the max limit of the increment of the column.
B. Integrity constraints were dropped after upgrading to the latest version, and the integrity checks were missed.
C. EAV attribute import uses REPLACE, which leads to reaching the max limit of the increment of the column

Correct Answer: C
Section:
Explanation:
EAV attribute import uses the REPLACE statement, which deletes and inserts a new row with the same primary key value. This causes the auto-increment column to increase by one for each row, even if the row already exists. If the auto-increment column reaches its maximum value, which is 2147483647 for a signed INT, then any further REPLACE statement will fail with an integrity constraint violation error.Reference:
EAV and extension attributes | Magento 2 Developer Documentation

GitHub - techdivision/import-attribute: This library provides the functionality for the Magento 2 import of EAV attributes
Data integrity in JSON (B) when replacing EAV - Stack Overflow

QUESTION 3
An Adobe Commerce Architect is planning to create a new action that will add gift registry items to the customer's quote. What should the Architect do to guarantee that private content blocks are updated?

A. Mark the controller by setting no-cache HTTP headers
B. Invalidate the status of gift registry indexers
C. Specify a new action in a sections.xml configuration file

Correct Answer: C
Section:
Explanation:
Private content blocks are sections of the page that are specific to each customer and are not cached by the server. To update these blocks when a customer performs an action, such as adding a gift registry item to the quote, the Adobe Commerce Architect needs to specify the new action in a sections.xml configuration file. This file defines which blocks need to be updated for each action and how often they should be updated. By doing this, the Architect can ensure that the private content blocks are refreshed with the latest data from the server.Reference:
Private content | Magento 2 Developer Documentation
Configure private content | Magento 2 Developer Documentation

QUESTION 4
An Adobe Commerce Architect needs to scope a bespoke news section for a merchants Adobe Commerce storefront. The merchant's SEO agency requests that the following URL structure:
news/{date}/{article_url_key}, where {date} is the publication date of the article, and {article_url_key} is the URL key of the article.
The Architect scopes that a news entity type will be created. The date and URL key data will be stored against each record and autogenerated on save. The values will be able to be manually overridden.
A. The Architect needs to manage routing this functionality and adhere to best practice. Which two options should the Architect consider to meet these requirements? (Choose two.)
B. Create a standard controller route and mapping the internal URLs (such as news/article/view/id/i) to rewrites that are generated on save and then stored in the URL rewrites table.
C. Create a custom router that runs before the standard router and matches the news portion of the URL, then looks for and loads a news article by matching the date and URL key parts of the URL
D. Create a plugin that intercepts Magento\Framework\App\Action: :(), looks for the news portion of the URL, and if it matches, loads the relevant news article by matching the URL date and URL key parts.
E. Create a standard controller route and an index/index controller class that loads the relevant news article by matching the URL date and URL key parts.

Correct Answer: B, C
Section:
Explanation:
These two options are both valid ways to manage routing for the bespoke news section and adhere to best practice. Option B leverages the existing URL rewrite functionality of Adobe Commerce, which allows creating custom URLs for any entity type and storing them in the database. This option requires creating a standard controller route for the news entity type, such as news/article/view/id/i, where i is the news article ID. Then, on saving each news article, a rewrite rule is generated that maps the internal URL to the desired SEO-friendly URL, such as news/{date}/{article_url_key}. The rewrite rule is stored in the url_rewrite table, which is used by the standard router to match and redirect requests.
Option C involves creating a custom router class that implements \Magento\Framework\App\RouterInterface and runs before the standard router in the routing process. The custom router class can match the news portion of the URL and extract the date and URL key parts from it. Then, it can look for and load a news article that matches those values using a model or repository class. If a match is found, it can set the request parameters accordingly and dispatch the request to a controller action that renders the news article page.
Routing | Adobe Commerce Developer Guide
URL Rewrites | Adobe Commerce Developer Guide
Custom Router | Adobe Commerce Developer Guide

QUESTION 5
An external system integrates functionality of a product catalog search using Adobe Commerce GraphQL API. The Architect creates a new attribute my_attribute in the admin panel with frontend type select-Later, the Architect sees that ProductInterf ace already has the field my_attribute, but returns an Int value. The Architect wants this field to be a new type that contains both option id and label.
To meet this requirement, an Adobe Commerce Architect creates a new module and file etc/schema.graphqls that declares as follows:

```
interface ProductInterface {
    my_attribute: SelectableOption @resolver(class:"Vendor\\CatalogGraphQl\\Model\\Resolver\\SelectableOption")
}
```

After calling command setup:upgrade, the introspection of ProductInterface field my_attribute remains Int. What prevented the value type of field my_attribute from changing?

A. The Magento_CatalogGraphQI module occurs later in sequence than the Magento_GraphQI module and merging output of dynamic attributes schema reader overrides types declared in schema.graphqls
B. The fields of ProductInterface are checked during processing schema.graphqls files. If they have a corresponding attribute, then the backendjype of product attribute is set for field type.
C. The interface ProductInterface is already declared in Magento.CatalogGraphQI module. Extending requires use of the keyword extend before a new declaration of ProductInterface.

Correct Answer: C
Section:
Explanation:
According to the Adobe Commerce documentation, to extend an existing GraphQL interface, the keyword extend must be used before the interface name. This indicates that the new declaration is adding or modifying fields to the existing interface, rather than redefining it. If the keyword extend is omitted, the new declaration will be ignored and the original interface will be used. In this case, the Architect wants to change the type of the my_attribute field in the ProductInterface interface, which is already declared in the Magento.CatalogGraphQl module. Therefore, the Architect should use the keyword extend before declaring the ProductInterface interface in the schema.graphqls file of the custom module. This will allow the Architect to override the type of the my_attribute field from Int to MyAttributeType.
Extend existing schema | Adobe Commerce Developer Guide
Schema language with GraphQL | Adobe Commerce

QUESTION 6
An Adobe Commerce store owner sets up a custom customer attribute 'my.attribute'.
An Architect needs to display additional content on the home page, which should display only to Customers with 'my.attribute' of a certain value and be the same content for all of them. The website is running Full Page Cache.
With simplicity in mind, which two steps should the Architect take to implement these requirements? (Choose two.)

A. Add a new context value of 'my_attribute' to Magento\Framework\App\Http\Context
B. Create a Customer Segment and use 'my.attribute' in the conditions
C. Add a custom block and a pHTML template with the content to the cmsjndexjndex.xml layout
D. Add a dynamic block with the content to the Home Page
E. Use customer-data JS library to retrieve 'my.attribute' value

Correct Answer: A, D
Section:
Explanation:
To display additional content on the home page based on a custom customer attribute, the Architect needs to do the following steps:
Add a new context value of ''my_attribute'' to Magento\Framework\App\Http\Context. This will allow the Full Page Cache to generate different versions of the page for customers with different values of ''my.attribute''. The context value can be set using a plugin on the Magento\Customer\Model\Context class.
Add a dynamic block with the content to the Home Page. A dynamic block is a type of content block that can be configured to display only to specific customer segments or conditions. The Architect can use the 'my.attribute' in the conditions of the dynamic block and assign it to the Home Page in the Content > Blocks section of the Admin Panel.Reference:
Private content | Magento 2 Developer Documentation
Dynamic Blocks | Adobe Commerce 2.3 User Guide - Magento

QUESTION 7
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 steps are required when adding a product type with custom pricing? (Choose three.)

A. Content of the etc/product_types.xml file

B. Data patch to register the new product type
C. Hydrator for attributes belonging to the new product type
D. New price model extending \Magento\Catalog\Model\Product\Type\Price
E. Custom type model extended from the abstract Product Type model
F. A new class with custom pricing logic, extending the abstract Product model class

Correct Answer: A, D, E
Section:
Explanation:
To add a product type with custom pricing, the Architect needs to do the following steps:
Create a content of the etc/product_types.xml file that defines the new product type, its label, model, index priority, and price model.This file is used to register the new product type and its associated classes in Magento1.
Create a new price model that extends \Magento\Catalog\Model\Product\Type\Price and implements the custom pricing logic for the new product type.The price model is responsible for calculating the final price of the product based on various factors, such as special price, tier price, catalog price rules, etc2.
Create a custom type model that extends from the abstract Product Type model (\Magento\Catalog\Model\Product\Type\AbstractType) and overrides the methods related to the product type behavior, such as prepareForCart, getAssociatedProducts, etc.The type model defines how the product type interacts with other components, such as quote, order, cart, etc3.Reference:
How to add a new product type in Magento 2? (MageStackDay mystery question 1) - Magento Stack Exchange
Magento 2: How to create custom product types - BelVG Blog
Magento 2: How to create custom product types - BelVG Blog

QUESTION 8
An Adobe Commerce Architect needs to log the result of a ServiceClass:: getData method execution after all plugins have executed. The method is public, and there are a few plugins declared for this method. Among those plugins are after and around types, and all have sortOrder specified.
Which solution should be used to meet this requirement?

A. Declare a new plugin with the sortOrder value lower than the lowest declared plugin sortOrder and implement aroundGetData method.
B. Declare a new plugin with the sortOrder value higher than the highest declared plugin sortOrder and implement afterGetData method.
C. Declare a new plugin with the sortOrder value higher than the highest declared plugin sortOrder and implement aroundGetData method.

Correct Answer: B
Section:
Explanation:
This solution ensures that the new plugin will execute after all the existing plugins for the ServiceClass::getData method, and will be able to log the final result of the method execution. The afterGetData method of the new plugin will receive the result of the method as a parameter, and can use any logging mechanism to record it. The sortOrder value of the new plugin should be higher than the highest declared plugin sortOrder, so that it will run last in the sequence of plugins. The after type of plugin is preferred over the around type of plugin, because it is simpler and more efficient, and does not require calling the proceed() method.
Plugins (Interceptors) | Adobe Commerce Developer Guide
Plugin best practices | Adobe Commerce Developer Guide

QUESTION 9
While developing a new functionality for a website in developer mode with all cache types enabled, an Adobe Commerce Developer needs to add \Magento\Sales\Model\Service\InvoiceService SinvoiceService as a new dependency to an existing page action controller in Vendor\CustomModule\Controller\Index\Index . This is accomplished as follows:

```
[...]
public function __construct(
    \Magento\Framework\App\Action\Context $context,
    \Magento\Sales\Model\Service\InvoiceService $invoiceService
    \Magento\Framework\View\Result\PageFactory $resultPageFactory
) {
[...]
```

After cleaning the f ull_page cache and reloading the page, the developer encounters the following exception:
Recoverable Error: Argument 2 passed to Vendor\CustomModule\Controller\Index\Index::__construct() must be an instance of

\Magento\Sales\Model\Service\InvoiceService [...]
Which action should the Architect recommend to the developer to fix this error?

A. Clean the block_html cache along with full_page cache.
B. Add the new \Magento\sales\Model\service\invoiceService Sinvoiceservice dependency at the end of the constructor signature.
C. Remove the generated Child ClaSS from generated/code/Vendor/CustomModule/Controller/Index/Index.

Correct Answer: C
Section:
Explanation:
The error is caused by the generated child class not being updated with the new dependency. Removing the generated child class will allow the system to generate a new child class with the correct dependency. The generated child class is a proxy class that extends the original controller class and overrides the constructor to inject the dependencies using the object manager. The generated child class is created when the system runs in developer mode with cache enabled, to avoid performance issues. However, when a new dependency is added to the original controller class, the generated child class does not reflect the change and causes a mismatch in the constructor arguments. Therefore, deleting the generated child class from the generated/code directory will solve the problem.
Generated code | Adobe Commerce Developer Guide
Constructor signature change | Adobe Commerce Developer Guide

QUESTION 10
A representative of a small business needs an Adobe Commerce Architect to design a custom integration of a third-party payment solution. They want to reduce the list of controls identified in their Self-Assessment Questionnaire as much as possible to achieve PCI compliance for their existing Magento application.
Which approach meets the business needs?

A. Utilize the Advanced Encryption standard (aes-256) algorithm to encrypt all customer-sensitive data from the payment module.
B. Utilize the payment provider iframe system to isolate content of the embedded frame from the parent web page.
C. Utilize a trusted signed certificate issued by a Certification Authority (CA) to secure each connection made by the payment solution protocol via https.

Correct Answer: B
Section:
Explanation:
Using an iframe system for payment integration can help reduce the PCI scope and compliance burden for the merchant, as the payment data is collected and processed by the payment service provider (PSP) within the iframe, without touching the merchant's website or server. This way, the merchant can leverage the PSP's PCI certification and avoid storing or transmitting any sensitive cardholder data on their own system. The iframe also provides a secure barrier between the host webpage and the loaded page, preventing any access or manipulation of the payment data by malicious actors.To implement this approach, the merchant needs to embed the PSP's payment form in their checkout page using an iframe element, and configure the communication between the iframe and the host page using JavaScript123.

QUESTION 11
A merchant asks for a new category attribute to allow uploading an additional mobile image against categories. The merchant utilizes the content staging and preview feature in Adobe Commerce and wants to schedule and review changes to this new mobile image field.
A developer creates the attribute via a data patch and adds it to view/adminhtml/ui_component/category_f orm. xml. The attribute appears against the category in the main form, but does not appear in the additional form when scheduled updates are made.
To change this attribute when scheduling new category updates, which additional action should the Architect ask the developer to take?

A. The attribute must have its apply_to field set to 'staging' in the data patch file.
B. The attribute must have <item- name=''allow_staging' xsi:type=''boolean''>true</item> set in the cjt.gopy_for-.xni file under the attributes config' section.
C. The attribute must also be added to view/adminhtml/ui_co-component/catalogstaging_category_update_form.xml.

Correct Answer: C
Section:
Explanation:

This action is necessary to make the attribute available for content staging and preview. According to the Adobe Commerce documentation, the catalogstaging_category_update_form.xml file defines the fields that are displayed in the Scheduled Changes section of the category form. The file extends the category_form.xml file and adds additional fields that are specific to content staging, such as start and end dates, campaign name, description, etc. To include a custom category attribute in the Scheduled Changes section, the attribute must also be declared in the catalogstaging_category_update_form.xml file with the same configuration as in the category_form.xml file.
Content staging | Adobe Commerce Developer Guide
Create a category attribute | Adobe Commerce Developer Guide

QUESTION 12
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 Vendor\Module\Model\Segment\Condition\AverageSalesAmount with all of its requirements:

```php
<?php

namespace Vendor\Module\Plugin;

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

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

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

During testing, the following error appears:

```
"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"
```

What should the Architect do to fix the problem?
A)
Set the class to be `\Vendor\Module\Model\Segment\Condition\AverageSalesAmount` for the `$conditions value` attribute

B)
Use a preference `<preference for="Magento\CustomerSegment\Model\Segment\Condition\AverageSalesAmount"`

`type="Vendor\Module\Model\Segment\Condition\AverageSalesAmount"/>`

C)
Use a virtualType `<virtualType name="Magento\CustomerSegment\Model\Segment\Condition\AverageSalesAmount"`

`type="Vendor\Module\Model\Segment\Condition\AverageSalesAmount"/>`

A. Option A
B. Option B
C. Option C

Correct Answer: B
Section:
Explanation:
The error is caused by the missing class declaration for the custom condition class. According to the Adobe Commerce documentation, to create a custom customer segment condition, the class must extend the \Magento\CustomerSegment\Model\Condition\AbstractCondition class and implement the \Magento\CustomerSegment\Model\Condition\Combine\Interface interface. The class must also declare its name, label, value type, and attribute code properties. Option B is the only option that correctly declares the class with the required properties and inheritance. Option A and Option C are incorrect because they do not extend the AbstractCondition class or implement the CombineInterface interface, and they do not declare the name, label, value type, or attribute code properties.
Create a customer segment condition | Adobe Commerce Developer Guide
AbstractCondition | Adobe Commerce Developer Guide

QUESTION 13
A single Adobe Commerce Cloud instance is set up with two websites (each with a single store view) with different domains.
* The default website is website_one, with store view store_one, and domain storeone. com.
* The second website is website_two, with store view store_two, and domain storetwo. com.
The magento-vars. php file is set up as follows to determine which website each request runs against:

```
<?php
function isHttpHost($host)
{
    if (!isset($_SERVER['HTTP_HOST'])) {
        return false;
    }
    return $_SERVER['HTTP_HOST'] ===  $host;
}

$_SERVER["MAGE_RUN_TYPE"] = "website";
if (isHttpHost("storetwo.com")) {
    $_SERVER["MAGE_RUN_CODE"] = "website_two";
} else {
    $_SERVER["MAGE_RUN_CODE"] = "website_one";
}
```

When testing a new GraphQL integration, all requests returned data relating to the default website, regardless of the domain. What is causing this issue?

A. The magento-vars.php file is not processed for any GraphQL requests, so the default website is always processed.
B. $_server['mage_run_cooe') needs to be set to store and *$_SERVER['MAGE_RUN_TYPE'] needs to be set to the store code instead.
C. GraphQL requests are always run against the default store view unless a store header or store cookie is provided.

Correct Answer: C
Section:
Explanation:
The magento-vars.php file is used to set the website or store view based on the HTTP host, but it does not affect GraphQL requests. GraphQL requests are handled by a separate controller that does not use the magento-vars.php file. Instead, GraphQL requests use the default store view of the default website, unless a store header or store cookie is provided in the request. The store header or cookie should contain the store code of the desired store view.For example, to query data from website_two, the request should include a header likestore: store_twoor a cookie likestore=store_two12.Reference:
GraphQL overview | Adobe Commerce 2.4 User Guide - Magento
How to set up multiple websites with Magento 2 - Mageplaza

QUESTION 14
An Architect needs to integrate an Adobe Commerce store with a new Shipping Carrier. Cart data is sent to the Shipping Carrier's API to retrieve the price and display to the customer. After the feature is implemented on the store, the API hits its quota and returns the error 'Too many requests'. The Shipping Carrier warns the store about sending too many requests with the same content to the API.
In the carrier model, what should the Architect change to fix the problem?

A. In_doShipmentRequest()f call canCollectRates() before sending request to the API.

B. Override getResponse, save the response to a variable, check if the response exists, then return.
C. Implement _setCachedQuotes() and _getCachedQuotes(), return the data if the request matches.

Correct Answer: C
Section:
Explanation:
The carrier model class can implement caching methods to store and retrieve the quotes from the API based on the request parameters. This can reduce the number of API calls and improve the performance of the shipping rate calculation. The _setCachedQuotes() method can save the response from the API to a cache storage, and the _getCachedQuotes() method can check if there is a cached response for the current request and return it if it exists.Reference:Caching in carrier model,Carrier model interface

QUESTION 15
An Architect working on a headless Adobe Commerce project creates a new customer attribute named my_attribute. Based on the attribute value of the customer, the results of GraphQl queries are modified using a plugin. The frontend application is communicating with Adobe Commerce through Varnish by Fastly. which is already caching the queries that will be modified. The Adobe Commerce Fastly extension is installed, and no other modifications are made to the application.
Which steps should the Architect take to make sure the vcl_hash function of Varnish also considers the newly created attribute?

A. Create a new ClaSS inheriting from Magento\GraphQlCache\Model\CacheId\CacheIdFactorProvidftrInterface and returning the Value of my_attribute from the getFactorValue function and my_attribute from the getFactorName function. Then add this class through Dl to the idFactorProviders array of Magento\GraphQlCache\Model\CacheId\CacheIdCalculator.
B. Create a new class inheriting from Magento\Framework\GraphQi\Query\Resolver\identityinterfaca and returning the value of my_attribute from the getidentities function. Then specify a ecache(cacheidentity: Path\\To\\identityclass) directive for each GraphQL query to include the newly created IdentityClass to each query that adds the cache tags for each customer.
C. Create a new class inheriting from Magento\customer\customerData\stctionSourceinterface and returning the value of my_attribute from the getSectionData function. Then add this ClaSS through the sectionSourceMap array Of Magento\Customer\CustomerData\SectionPoolInterface.

Correct Answer: A
Section:
Explanation:
To make sure the vcl_hash function of Varnish considers the newly created attribute, the Architect needs to do the following steps:
Create a new class that implements the Magento\GraphQlCache\Model\CacheId\CacheIdFactorProviderInterface interface. This interface defines two methods: getFactorName and getFactorValue. The getFactorName method should return the name of the attribute, in this case, my_attribute.The getFactorValue method should return the value of the attribute for the current customer, which can be obtained from the customer session or customer repository1.
Add this class to the idFactorProviders array of Magento\GraphQlCache\Model\CacheId\CacheIdCalculator through dependency injection. The CacheIdCalculator is responsible for generating a cache ID for each GraphQL request based on the factors provided by the idFactorProviders.By adding the new class to this array, the Architect ensures that the cache ID will include the value of my_attribute1.
The cache ID is then used by Varnish to hash and lookup the cached response for each request.By including my_attribute in the cache ID, the Architect ensures that Varnish will serve different responses based on the attribute value of the customer2.Reference:
Magento_GraphQlCache module | Magento 2 Developer Documentation
Varnish caching | Adobe Commerce 2.4 User Guide - Magento

QUESTION 16
An Architect wants to create an Integration Test that does the following:
* Adds a product using a data fixture
* Executes $this->someLogic->execute($product) on the product
* Checks if the result is true.
$this->someLogic has the correct object assigned in the setup() method.
Product creation and the tested logic must be executed in the context of two different store views with IDs of 3 and 4, which have been created and are available for the test.
How should the Architect meet these requirements?

A. Create two test classes with one test method each. Use the @magentoExecuteinstoreContext 3 and $MagentoExecuteinStoreContext 4 annotations on the class level.
B. Create one test class with two test methods. Use the emagentostorecontext 3 annotation in one method and amagentostorecontext 4 in the other one.
C. Create one test class with one test method. Use the \Magento\TestFramework\store\ExecuteinstoreContext class once in the fixture and another time in the test.

Correct Answer: C
Section:
Explanation:
To create an integration test that executes different logic in different store views, the Architect needs to do the following steps:
Create one test class that extends \Magento\TestFramework\TestCase\AbstractController or \Magento\TestFramework\TestCase\AbstractBackendController, depending on the type of controller being tested1.
Create one test method that uses the @magentoDataFixture annotation to specify the data fixture file that creates the product2.
Use the \Magento\TestFramework\Store\ExecuteInStoreContext class to execute the fixture and the tested logic in different store views. This class has a method called executeInStoreContext, which takes two parameters: the store ID and a callable function.The callable function will be executed in the context of the given store ID, and then the original store ID will be restored3. For example:
PHPAI-generated code. Review and use carefully.More info on FAQ.

```
public function testSomeLogic()
{
// Get the product from the fixture
$product = $this->getProduct();
// Get the ExecuteInStoreContext instance from the object manager
$executeInStoreContext = $this->_objectManager->get(\Magento\TestFramework\Store\ExecuteInStoreContext::class);
// Execute the fixture in store view 3
$executeInStoreContext->executeInStoreContext(3, function () use ($product) {
// Do some operations on the product in store view 3
});
// Execute the tested logic in store view 4
$result = $executeInStoreContext->executeInStoreContext(4, function () use ($product) {
// Call the tested logic on the product in store view 4
return $this->someLogic->execute($product);
});
// Assert that the result is true
$this->assertTrue($result);
}
```

Integration tests | Magento 2 Developer Documentation
Data fixtures | Magento 2 Developer Documentation
Magento\TestFramework\Store\ExecuteInStoreContext | Magento 2 Developer Documentation

QUESTION 17
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.)

A. Set mage_ihdexer_threads_COUNT environment variable to enable parallel mode
B. Move catalog_Price_index indexer to another custom indexer group
C. Enable price index customer group merging for products without tier prices
D. Set Customer Share Customer Accounts Option to Global
E. Edit customer groups to exclude websites that they are not using

Correct Answer: A, C
Section:
Explanation:
The product price index can be optimized by using parallel mode and customer group merging. Parallel mode allows the indexer to run multiple threads simultaneously, which can speed up the indexing process. Customer group merging reduces the number of rows in the price index table by merging customer groups that have the same product prices. This can improve the performance of the price index queries and reduce the index size.Reference:Indexing optimization,Price index customer group merging

QUESTION 18

While reviewing a newly developed pull request that refactors multiple custom payment methods, the Architect notices multiple classes that depend on \Magento\Framework\Encryption\EncryptorInterface to decrypt credentials for sensitive dat a. The code that is commonly repeated is as follows:

```
namespace Vendor\PaymentModule\Gateway\Config;

class Config extends \Magento\Payment\Gateway\Config\Config
{
    ...
    public function __construct(
        ...
        ScopeConfigInterface $scopeConfig,
        EncryptorInterface $encryptor,
        ...
    ) {
        parent::__construct($scopeConfig, $methodCode, $pathPattern);
        $this->scopeConfig = $scopeConfig;
        $this->encryptor = $encryptor;
    }

    public function getUserSecret(): string
    {
        return $this->encryptor->decrypt(
            $this->scopeConfig->getValue('payment/method_code/user_secret')
        );
    }
```

The Architect needs to recommend an optimal solution to avoid redundant dependency and duplicate code among the methods. Which solution should the Architect recommend?

A. Create a common config service class vndor\Pay-ient\Gatway\conf ig\conf ig under Vendor.Payment and use it as a parent class for all of the
B. Replace all Vendor\PaymentModule\Gateway\Config\Config ClaSSeS With virtualType Of Magento\Payiaent\Gateway\Conf ig\Conf ig and Set <user_secret backend_Model='Magento\Config\Model\Config\Backend\Encrypted' /> Under config.xml
C. Add a plugin after the getvalue method of $scopeConfig, remove the $encryptor from dependency and use it in the plugin to decrypt the value if the config name is user.secret'

Correct Answer: B
Section:
Explanation:
The Architect should recommend replacing all Vendor\PaymentModule\Gateway\Config\Config Classes with virtualType of Magento\Payment\Gateway\Config\Config and setting <user_secret backend_Model=''Magento\Config\Model\Config\Backend\Encrypted'' /> under config.xml. This will avoid redundant dependency and duplicate code among the methods. The virtualType of Magento\Payment\Gateway\Config\Config will inherit the functionality of the base class and allow the customization of the constructor arguments, such as the pathPattern and valueHandlerPool.The backend_Model attribute of the user_secret field will specify that the value of this field should be encrypted and decrypted by the Magento\Config\Model\Config\Backend\Encrypted class, which implements the \Magento\Framework\App\Config\ValueInterface interface and uses the \Magento\Framework\Encryption\EncryptorInterface internally12.This way, the payment modules do not need to depend on the \Magento\Framework\Encryption\EncryptorInterface or the \Magento\Framework\App\Config\ScopeConfigInterface directly, and can use the getValue method of the Magento\Payment\Gateway\Config\Config class to get the decrypted value of the user_secret field3.Reference:
How to encrypt system configuration fields in Magento 2 - Mageplaza
Magento 2: How to Encrypt/Decrypt System Configuration Fields - Webkul Blog
Magento 2: How to create custom payment method - BelVG Blog

QUESTION 19
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 AwesomeAgency\CodingStandard\ namespace
* The ruleset. xml file extending the Magento 2 Coding Standard
What should the Architect do to implement the new code rule?
A)

Implement `\PHP_CodeSniffer\Sniffs\Sniff` under your `\AwesomeAgency\CodingStandard\Sniff\HelperNamespaceSniff`. Provide required implementation in `process` method.

B)

Create a new class `\AwesomeAgency\CodingStandard\Ruleset\HelperNamespaceRule`, extend `\PHP_CodeSniffer\Ruleset` and specify your rule inside `processRule` method.

C)

Adjust the `ruleset.xml` file with the new rule:

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

A. Option A
B. Option B
C. Option C

Correct Answer: C
Section:
Explanation:
Option C is correct because adjusting the ruleset.xml file with the new rule is the simplest and most effective way to implement the new code rule. The ruleset.xml file defines the coding standards that are applied by PHP_CodeSniffer. By extending the Magento 2 Coding Standard and adding a new rule, the Architect can customize the code analysis and enforce the company coding standards.The new rule can use the Magento2.Namespaces.ForbiddenNamespaces sniff to check for any usage of Helper classes in the code and report them as errors or warnings1.
Option A is incorrect because creating a new composer package under the AwesomeAgency\CodingStandard\ namespace is not enough to implement the new code rule. The composer package is just a way to distribute and install the coding standard, but it does not define the rules themselves.The Architect still needs to create a ruleset.xml file and register it with PHP_CodeSniffer2.
Option B is incorrect because creating a new class \AwesomeAgency\CodingStandard\Ruleset\ForbiddenNamespaces and specifying the rule inside the process method is unnecessary and complicated. The Architect does not need to create a new class or a new sniff for this rule, as there is already an existing sniff in the Magento 2 Coding Standard that can be used for this purpose.The Magento2.Namespaces.ForbiddenNamespaces sniff can be configured with an include-pattern element to specify which namespaces are forbidden1.
1: Magento 2 Coding Standards | Adobe Commerce Developer Guide
2: How to create a custom coding standard | PHP_CodeSniffer Documentation

QUESTION 20
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.)

A. Reduce the frequency of the cron job to 5 minutes so the items have more time to process.
B. Make sure that no custom or third-party modules modify the changelog and indexing process.
C. 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.
D. Invalidate the catalog_Product_price indexer in the Adobe Commerce Admin Panel so that it is fully reindexed next time the cron runs.
E. Manually reindex the catalog_product_price index from the command line: bin/magento indexer:reindex catalog_product_price.

Correct Answer: B, C

Section:
Explanation:
The issue here is that the product price changes are not reflected on the storefront, even though the indexes are set to update by schedule and there are no items in the backlog. This indicates that there might be some problem with the changelog and indexing process, which are responsible for tracking and applying the data changes to the index tables. Therefore, the Architect should recommend the developer to check if any custom or third-party modules interfere with the changelog and indexing process, and disable or fix them if needed. Additionally, the Architect should recommend the developer to verify that the version_id for the price indexer in the mview_state table is consistent with the last entry for the same column in the changelog table, and re-synchronize them if they are out of sync.This will ensure that the indexer can process all the data changes correctly and update the index tables accordingly.Reference: https://experienceleague.adobe.com/docs/commerce-admin/systems/tools/index-management.html?lang=en#cron-groups-and-processes1https://devdocs.magento.com/guides/v2.4/extension-dev-guide/indexing.html#m2devgde-mview2

QUESTION 21
The development of an Adobe Commerce website is complete. The website is ready to be rolled out on the production environment.
An Architect designed the system to run in a distributed architecture made up of multiple backend webservers that process requests behind a Load Balancer.
After deploying the system and accessing the website for the first time, users cannot access the Customer Dashboard after logging in. The website keeps redirecting users to the sign-in page even though the users have successfully logged in The Architect determines that the session is not being saved properly.
In the 'app/etc/env.php', the session is configured as follows:

```
'session' => [
    'save' => 'redis',
    'redis' => [
        'host' => '127.0.0.1'
    ]
]
```

What should the Architect do to correct this issue?

A. Update the session host value to a shared Redis instance
B. increase the session size with the command config:set system/security/max_session_size_admin
C. Utilize the Remote Storage module to synchronize sessions between the servers

Correct Answer: A
Section:
Explanation:
Option A is correct because updating the session host value to a shared Redis instance in the ''app/etc/env.php'' file will allow the session to be saved properly and prevent users from being redirected to the sign-in page after logging in. Redis is a fast and reliable in-memory data store that can be used for session storage in Magento 2. By using a shared Redis instance, the session data can be accessed by any of the backend web servers behind the load balancer, regardless of which server handled the initial request.This ensures that the user's session is maintained and consistent across different servers1.
Option B is incorrect because increasing the session size with the command config:set system/security/max_session_size_admin will not solve the issue of session not being saved properly. This command only affects the admin session size limit, not the customer session size limit.Moreover, this command does not address the root cause of the issue, which is that the session data is not shared among the backend web servers2.
Option C is incorrect because utilizing the Remote Storage module to synchronize sessions between the servers is not a viable solution for this issue. The Remote Storage module is a feature of Magento Commerce Cloud that allows storing media files and other static content on a remote storage service such as AWS S3 or Azure Blob Storage.This module does not support synchronizing sessions between servers, as sessions are dynamic and transient data that need to be stored in a fast and accessible data store such as Redis3.
1: Use Redis for session storage | Adobe Commerce Developer Guide
2: Security | Adobe Commerce User Guide
3: Remote storage | Adobe Commerce Developer Guide

QUESTION 22
An Adobe Commerce Architect designs and implements functionality that introduces a new Complex Product Type to the existing Adobe Commerce website. Besides visual demonstration of the new product type, the changes include adjustments to the price index.
The website utilizes a multi-dimensional indexer feature to store the price index. The Architect decides to cover it with integration tests. After creating and running one test, the Architect discovers that database storage is not being fully cleaned.
The test method has the following annotation declaration:

```
**
 * @magentoAppArea frontend
 * @magentoDbIsolation disabled
 * @magentoIndexerDimensionMode catalog_product_price website_and_customer_group
 *
 * @magentoDataFixture Custom_ProductType::Test/_files/create_websites.php
 * @magentoDataFixture Custom_ProductType::Test/_files/create_customer_groups.php
 * @magentoDataFixture Custom_ProductType::Test/_files/create_test_products.php
 * @magentoDataFixture Custom_ProductType::Test/_files/reindex_prices.php
 */
public function testCustomProductTypePriceIndex(): void
```

Which adjustment should the Architect make to fix this issue?

A. Add annotation @magentoAppIsolation enabled to method PHPDoc
B. Modify method PHPDoc and change annotation @magentoDbIsolation to enabled
C. Create Customer_ProductType: :Test/_files/{fixture_name)_rollback.php for every fixture

Correct Answer: B
Section:
Explanation:
The issue here is that the database storage is not being fully cleaned after the test is run. The solution is to modify the method PHPDoc and change the annotation @magentoDbIsolation to enabled.This will ensure that the database storage is fully cleaned after the test is run.Reference: https://developer.adobe.com/commerce/testing/guide/integration/#database-isolation1

QUESTION 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.)

A. Enable Config flag Under deployement/blue_green/enabled
B. Run bin/magento setup:upgrade --dry-run=true to upgrade database
C. Run bin/magento setup:upgrade - -keep-generated to Upgrade database
D. Run bin/magento setup:upgrad --convert-old-scripts-true to Upgrade database
E. Enable Config flag Under developer/zero_down_time/enabled

Correct Answer: A, C
Section:
Explanation:
Option A is correct because enabling the config flag under deployment/blue_green/enabled is one of the steps to ensure zero downtime during the deployment process of Magento 2 on-premises. This flag enables the blue-green deployment feature, which allows deploying a new version of the Magento application to a separate environment (blue) without affecting the current live environment (green).Once the new version is ready, the traffic can be switched from green to blue with minimal or no downtime1.
Option C is correct because running bin/magento setup:upgrade --keep-generated is another step to ensure zero downtime during the deployment process of Magento 2 on-premises. This command updates the database schema and data without deleting the generated code and static view files.This way, the Magento application can still serve requests from the cache while the database is being upgraded2.
Option B is incorrect because running bin/magento setup:upgrade --dry-run=true does not upgrade the database, but only checks if there are any errors or conflicts in the database schema or data.This command can be used for testing purposes, but it does not affect the deployment process or the downtime3.
Option D is incorrect because there is no such option as --convert-old-scripts-true for the bin/magento setup:upgrade command. This option does not exist in Magento 2 and does not have any effect on the deployment process or the downtime.
Option E is incorrect because there is no such config flag as developer/zero_down_time/enabled in Magento 2. This flag does not exist in Magento 2 and does not have any effect on the deployment process or the downtime.
1: Blue-green deployment | Adobe Commerce Developer Guide
2: Deploy Magento to production | Adobe Commerce Developer Guide
3: Command-line installation options | Adobe Commerce Developer Guide