Adobe.AD0-E716.by.Atony.21q

Number: AD0-E716
Passing Score: 800
Time Limit: 120 min
File Version: 1.0

Exam AD0-E716

Exam A

QUESTION 1
An Adobe Commerce developer is being tasked with creating a new cron job to run a method that has already been written. What are the minimally required steps to accomplish this?
A. Create a crontab.xmi file and a new system configuration in system.xmi for the schedule.
B. Create crontab.xmi and cron_groups.xmi files to assign the new job to a cron group.
C. Create a crontab.xmi file and set a schedule for the new cron job.

Correct Answer: C
Section:
Explanation:
According to the Configure and run cron guide for Magento 2 developers, the crontab.xmi file is used to declare and configure cron jobs for a module. The file should specify the name, instance, method and schedule of the cron job. Therefore, creating a crontab.xmi file and setting a schedule for the new cron job are the minimally required steps to accomplish this task. Verified
Reference: https://devdocs.magento.com/guides/v2.3/config-guide/cli/config-cli-subcommands-cron.html

QUESTION 2
Which hashing algorithm will Adobe Commerce choose to hash customer passwords?
A. If the Sodium extension is installed, SHA256 will be chosen, otherwise MD5 will be used as the Magento default hashing algorithm.
B. If the Sodium extension is installed, Argon 2ID13 will be chosen, otherwise SHA256 will be used as the Magento default hashing algorithm.
C. It does not matter if the Sodium extension is installed or not, the Magento hashing default algorithm will be SHA256.

Correct Answer: B
Section:
Explanation:
If the Sodium extension is installed, Argon 2ID13 will be chosen as the Magento default hashing algorithm. Otherwise, SHA256 will be used.
The Sodium extension is a PHP extension that provides cryptographic functions. Argon 2ID13 is a password hashing algorithm that is considered to be more secure than SHA256.
If the Sodium extension is installed, Magento will use Argon 2ID13 as the default hashing algorithm for customer passwords. If the Sodium extension is not installed, Magento will use SHA256 as the default hashing algorithm.

QUESTION 3
An Adobe Commerce developer is developing a custom module. As part of their implementation they have decided that all instances of their Custom\Module\Model\Example class should receive a new instance of Magento\Filesystem\Adapter\Local.
How would the developer achieve this using di. xml?
A)

```
<type name="Custom\Module\Model\Example">
    <arguments>
        <argument name="adapter" xsi:type="object" shared="false">Magento\Filesystem\Adapter\Local</argument>
    </arguments>
</type>
```

B)

```
<type name="Custom\Module\Model\Example">
    <arguments>
        <argument name="adapter" xsi:type="object" singleton="false">Magento\Filesystem\Adapter\Local</argument>
    </arguments>
</type>
```

C)

```
<type name="Custom\Module\Model\Example">
    <arguments>
        <argument name="adapter" xsi:type="object" transient="true">Magento\Filesystem\Adapter\Local</argument>
    </arguments>
</type>
```

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

Correct Answer: B
Section:
Explanation:
The developer can achieve this by adding the following configuration to their di.xml file:
XML
<config>
<component name='Custom\Module\Model\Example' factory='Custom\Module\Model\ExampleFactory'>
</component>
</config>
This configuration will ensure that all instances of the Custom\Module\Model\Example class will receive a new instance of the Magento\Filesystem\Adapter\Local class.

QUESTION 4
An Adobe Commerce developer has been tasked with applying a pricing adjustment to products on the website. The adjustments come from a database table. In this case, catalog price rules do not work. They created a plugin for getPrice on the price model, but the layered navigation is still displaying the old price.
How can this be resolved?
A. Create an implementation for \Magento\Catalog\Hodel\Product\PriceModifierInterf ace.
B. Create an after plugin On \Magento\Catalog\Api\Data\BasePriceInterface:: getPrice.
C. Create a plugin for\Magento\Catalog\Model\Indexer\Product\Price::executeRow.

Correct Answer: C
Section:
Explanation:
The developer can resolve this issue by creating a plugin for the Magento\Catalog\Model\Indexer\Product\Price::executeRow() method. This method is responsible for updating the product price index. The plugin can be used to add the pricing adjustment from the database to the product price index. Once the product price index is updated, the layered navigation will display the correct price.
Here is an example of a plugin for the executeRow() method:
PHP
class MyPlugin
{
public function executeRow(
\Magento\Catalog\Model\Indexer\Product\Price $subject,
\Magento\Catalog\Model\Product $product,
array $data
) {
$adjustment = $this->getAdjustment($product);
$product->setPrice($product->getPrice() + $adjustment);
}
private function getAdjustment(Product $product)
{
$adjustment = $product->getData('adjustment');
if (!is_numeric($adjustment)) {

return 0;
}
return $adjustment;
}
}
This plugin will add the adjustment data from the product to the product price index. Once the product price index is updated, the layered navigation will display the correct price.

QUESTION 5
An Adobe Commerce developer is writing an integration test. They checked some Integration Tests for Magento core modules for reference and noticed that they use data fixtures initialized by adding annotations to test classes. For example:

```
/**
 * @magentoDataFixture Magento/AdminNotification/_files/notifications.php
 */
```

The developer wants to add their own fixture to test a MyVendor_MyModule they created. Which steps will make this possible?
A.
1- Create a PHP file with the fixture data inside their own module in [module dir]/Test/integration/_fiies/my_fixture.php.
2- Add the following annotation to the test method:

```
/**
 * @magentoDataFixture MyVendor_MyModule::Test/Integration/_files/my_fixture.php
 */
```

B.
1- Create a PHP file With the fixture data in [magento root dir]/dev/tests/integration/testsuite/MyVendor/MyModule/_files/my_fixture.php.
2- Add the following annotation to the test method:

```
/**
 * @magentoDataFixture MyVendor_MyModule::_files/my_fixture.php
 */
```

C. 1- Create a PHP file with the fixture data inside their own module in [module dir]/Test/integration/_f iies/my_f ixture.php.

2- Add the following annotation to the test method:

```
/**
 * @magentoDataFixture MyVendor/MyModule/_files/my_fixture.php
 */
```

Correct Answer: B
Section:
Explanation:
To add a custom fixture to test a MyVendor_MyModule, the developer needs to do the following:
Create a PHP file with the fixture data in[magento root dir]/dev/tests/integration/testsuite/MyVendor/MyModule/_files/my_fixture.php.
Add the following annotation to the test method:
@magentoDataFixture(
'testsuite/MyVendor/MyModule/_files/my_fixture.php'
)
This will tell Magento to load the fixture data from the my_fixture.php file before the test method is executed.

QUESTION 6
An Adobe Commerce developer has created a before plugin for the save() function within the Magento\Framework\App\cache\Proxy class. The purpose of this plugin is to add a prefix on all cache identifiers that fulfill certain criteria.
Why is the plugin not executing as expected?

A. Another around plugin defined for the same function does not call the callable.
B. Cache identifiers are immutable and cannot be changed.
C. The target ClaSS implements Magento\Framework\ObjectManager\NonInterceptableInterface.

Correct Answer: C
Section:
Explanation:
According to the Plugins (Interceptors) guide for Magento 2 developers, plugins are class methods that modify the behavior of public class methods by intercepting them and running code before, after, or around them. However, some classes in Magento 2 implement the NonInterceptableInterface interface, which prevents plugins from being generated for them. The Magento\Framework\App\cache\Proxy class is one of them, as it extends from Magento\Framework\ObjectManager\NonInterceptableInterface. Therefore, the plugin is not executing as expected because the target class implements NonInterceptableInterface. Verified Reference: https://devdocs.magento.com/guides/v2.3/extension-dev-guide/plugins.html

QUESTION 7
An Adobe Commerce developer has installed a module from a third-party vendor. This module fires a custom event named third_party_event_after and also defines an observer named third_party_event_after_observer that listens to that event. The developer wants to listen to this custom event in their own module but wants to execute their observer's logic after the third_party_event_after_observer observer has finished executing.
What would the developer do to ensure their observer runs after the observer defined by the third-party module?
A. Ensure the third-party module is listed in the <sequence> node of the developer's module.xmi file.
B. Set the sort order of the new observer to be less than that of the third-party vendor's observer.
C. This is not possible as observers listening to the same event may be invoked in any order.

Correct Answer: B
Section:
Explanation:
To ensure that the developer's observer runs after the observer defined by the third-party module, they need to set the sort order of the new observer to be less than that of the third-party vendor's observer.
The sort order is a number that is assigned to each observer. The lower the number, the earlier the observer will be executed.
For example, if the third-party vendor's observer has a sort order of 10, the developer's observer would need to have a sort order of 9 or lower.

QUESTION 8
An Adobe Commerce developer has been asked to modify the PageBuilder slider content type to allow a new custom content type (other than slide) to be assigned as a child. The developer has already created the new content type called improved_slide in their module. They now need to create a new view/adminhtml/pagebuilder/content_type/slider. xml file in their module to allow the new content type to be a child of slider content types.
What is the correct xml to accomplish this?
A)

```
<type name="slider">
    <children>
        <child name="improved_slide" policy="allow"/>
    </children>
</type>
```

B)

```
<type name="slider">
    <allowed_descendants>
        <descendant name="improved_slide" />
    </allowed_descendants>
</type>
```

C)

```
<type name="slider">
    <arguments>
        <argument name="allowed_children" xsi:type="array">
            <item name="improved_slide" xsi:type="string">improved_slide</item>
        </argument>
    </arguments>
</type>
```

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

Correct Answer: B
Section:
Explanation:
The following XML will allow the new content type to be a child of slider content types:
<type>slider</type>
<children>
<type>improved_slide</type>
</children>
Use code with caution.https://bard.google.com/faq
This XML will tell Magento that the slider content type can have improved_slide content types as children.

QUESTION 9
An Adobe Commerce developer creates a new website using a data patch. Each website will have unique pricing by website. The developer does not have visibility into the production and staging environments so they do not know what the configuration currently is.
How would they ensure the configuration is deployed and consistent across all environments?

A. Run the CLI command below and commit the changes to the repository:

```
bin/magento config:set catalog/price/scope 1 --lock-config
```

B. Run the CLI command below and commit the changes to the repository:

```
bin/magento config:set catalog/price/scope 1
```

C. Create a custom module and override the value in config.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Store:etc/c
    <default>
        <catalog>
            <price>
                <scope>1</scope>
            </price>
        </catalog>
    </default>
</config>
```

Correct Answer: B
Section:
Explanation:
To ensure that the configuration is deployed and consistent across all environments, the developer can use the following steps:

Create a data patch that contains the configuration for the new website.
Deploy the data patch to all environments.
Use themagento deploy:statuscommand to verify that the configuration has been deployed to all environments.

QUESTION 10
An Adobe Commerce developer was asked to provide additional information on a quote. When getting several quotes, the extension attributes are returned, however, when getting a single quote it fails to be returned.
What is one reason the extension attributes are missing?
A. The developer neglected to add coiiection='trueM to their attribute in etc/extension_attributes.xmi file. O ottribute code='my_attributesM type='MyVendor\MyModule\Api\Data\^AttributeInterface[]M collection='true' />
B. The developer neglected to provide a plugin On Hagento\Quote\Api\CartRepositoryInterface: :get.
C. The developer neglected to implement an observer on the coiiection_ioad_after event.

Correct Answer: A
Section:
Explanation:
The extension attributes are missing because the collection='true' attribute is not set in the etc/extension_attributes.xmi file. This attribute tells Magento that the extension attributes should be returned when the quote is retrieved.
To fix this issue, the developer needs to add the collection='true' attribute to the my_attributes extension attribute.
Once the collection='true' attribute is set, the extension attributes will be returned when the quote is retrieved.

QUESTION 11
A logistics company with an Adobe Commerce extension sends a list of reviewed shipment fees to all its clients every month in a CSV file. The merchant then uploads this CSV file to a 'file upload' field in admin configuration of Adobe Commerce.
What are the two requirements to display the 'file upload' field and process the actual CSV import? (Choose two.)
A)

Add a custom backend model which extends `\Magento\Framework\App\Config\Value` and call `afterSave`:

```
// etc/adminhtml/system.xml
<field id="import_fees" ...>
    <label>Import shipment fees</label>
    <backend_model>My\Module\Model\Config\Backend\ImportFees</backend_model>

    ...

</field>
```

B)

```
// \My\Module\Model\Config\Backend\ImportFees

class \My\Module\Model\Config\Backend\ImportFees extends \Magento\Framework\App\Config\Value

...
public function afterSave()
{
    /** @var \My\Module\Model\ImportFeed $importFees */
    $importFees = $this->importFeesFactory->create();
    $importFees->uploadAndImport($this);
    return parent::afterSave();
}
```

C)

Add a new field in `etc/adminhtml/system.xml` in My_Module with the `file` type:

```
<field id="import_fees" translate="label" type="file" sortOrder="1000" showInDefault="1">
    <label>Import shipment fees</label>
    ...
</field>
```

D)

Add a new field in `etc/adminhtml/system.xml` in My_Module with a new custom type:

```
<field id="import_fees" translate="label" type="My\Module\Block\Adminhtml\Form\Field\ImportFees" sortOrder="1000" showInDefault="1"
    <label>Import shipment fees</label>
    ...
</field>
```

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

Correct Answer: A, B
Section:
Explanation:
To display the 'file upload' field and process the actual CSV import, the following two requirements must be met:
The developer must create a new system configuration setting that specifies the path to the CSV file.
The developer must create a new controller action that handles the file upload and import process.
The system.xml file is used to define system configuration settings. The following XML snippet shows how to define a new system configuration setting for the CSV file path:
XML
<config>
<system>
<config>
<shipment_fees_csv_path>/path/to/csv/file</shipment_fees_csv_path>
</config>
</system>
</config>
The Controller\Adminhtml\ShipmentFees controller class is used to handle the file upload and import process. The following code shows how to create a new controller action that handles the file upload and import process:
PHP
public function uploadAction()
{
$file = $this->getRequest()->getFile('shipment_fees_csv_file');
if ($file->isUploaded()) {
$importer = new ShipmentFeesImporter();
$importer->import($file);
}
return $this->redirect('adminhtml/system_config/edit/section/shipment_fees');
}

QUESTION 12
An Adobe Commerce developer is tasked with adding custom data to orders fetched from the API. While keeping best practices in mind, how would the developer achieve this?

A. Create an extension attribute on Nagento\sales\Api\E)ata\orderinterface and an after plugin on Magento\Sales\Model\Order: :getExtensionAttributes() to add the custom data.
B. Create an extension attribute On Magento\Sales\Api\Data\OrderInterface and an after plugin On Magento\Sales\Api\OrderRepositoryInterface On geto and getListo to add the custom data.
C. Create a before plugin on Magento\sales\model\ResourceModel\order\collection: :load and alter the query to fetch the additional data. Data will then be automatically added to the items fetched from the API.

Correct Answer: B
Section:
Explanation:
The developer should create an extension attribute on the Magento\Sales\Api\Data\OrderInterface interface and an after plugin on the Magento\Sales\Api\OrderRepositoryInterface::get() and Magento\Sales\Api\OrderRepositoryInterface::getList() methods.
The extension attribute will store the custom data. The after plugin will be used to add the custom data to the order object when it is fetched from the API.
Here is the code for the extension attribute and after plugin:
PHP

```
namespace MyVendor\MyModule\Api\Data;
interface OrderExtensionInterface extends \Magento\Sales\Api\Data\OrderInterface
{
/**
* Get custom data.
*
* @return string|null
*/
public function getCustomData();
/**
* Set custom data.
*
* @param string $customData
* @return $this
*/
public function setCustomData($customData);
}
namespace MyVendor\MyModule\Model;
class OrderRepository extends \Magento\Sales\Api\OrderRepositoryInterface
{
/**
* After get order.
*
* @param \Magento\Sales\Api\OrderRepositoryInterface $subject
* @param \Magento\Sales\Api\Data\OrderInterface $order
* @return \Magento\Sales\Api\Data\OrderInterface
*/
public function afterGetOrder($subject, $order)
{
if ($order instanceof OrderExtensionInterface) {
$order->setCustomData('This is custom data');
}
return $order;
}
/**
* After get list.
```

```
*
* @param \Magento\Sales\Api\OrderRepositoryInterface $subject
* @param \Magento\Sales\Api\Data\OrderInterface[] $orders
* @return \Magento\Sales\Api\Data\OrderInterface[]
*/
public function afterGetList($subject, $orders)
{
foreach ($orders as $order) {
if ($order instanceof OrderExtensionInterface) {
$order->setCustomData('This is custom data');
}
}
return $orders;
}
}
```

Once the extension attribute and after plugin are created, the custom data will be added to orders fetched from the API.

QUESTION 13

An Adobe Commerce developer has created a process that exports a given order to some external accounting system. Launching this process using the Magento CLI with the command php bin/magento my_module:order: process --order_id=<order_id> is required.

Example: php bin/magento my_module:order:process --order_id=1245.

What is the correct way to configure the command?

A)

```
protected function configure()
{
    $this->setName('my_module:order:process');
    $this->setDescription('Processes an order');
    parent::configure();
}

protected function values()
{
    return [new InputValue('order_id', InputValue::REQUIRED, 'Order ID')];
}
```

B)

```
protected function configure()
{
    $this->setName('my_module:order:process');
    $this->setDescription('Processes an order');
    $this->addOption('order_id', null, InputOption::VALUE_REQUIRED, 'Order ID');
    parent::configure();
}
```

C)

```
protected function configure()
{
    $this->setName('my_module:order:process');
    $this->setDescription('Processes an order');
    $this->addOption('order_id', null, InputOption::VALUE_REQUIRED, 'Order ID');
```

```
    $this->setName('my_module:order:process');
    $this->setDescription('Processes an order');
    $this->addArgument('order_id', InputArgument::REQUIRED, 'Order ID');
    parent::configure();
}
```

D)

```
protected function configure()
{
    $this->setName('my_module:order:process');
    $this->setDescription('Processes an order');
    $this->addArgument('order_id', InputArgument::REQUIRED, 'Order ID');
    parent::configure();
}
```

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

Correct Answer: C
Section:
Explanation:
According to the How to use the Magento command-line interface (CLI) guide, to configure and run the Magento CLI, the developer needs to make the bin/magento file executable and then use it to run commands. To create a custom command, the developer needs to create a class that implements \Symfony\Component\Console\Command\Command and define its name, description, arguments, options, and logic. The developer also needs to register the command in the di.xml file of their module using the Magento\Framework\Console\CommandList argument. Therefore, option C is the correct answer, as it shows the correct class and di.xml code to configure the custom command. Verified
Reference: https://www.a2hosting.com/kb/installable-applications/optimization-and-configuration/magento1/using-the-magento-command-line-interface-cli/

QUESTION 14
An Adobe Commerce developer is working on a Magento 2 instance which contains a B2C and a B2B website, each of which contains 3 different store views for English, Welsh, and French language users. The developer is tasked with adding a link between the B2C and B2B websites using a generic link template which is used throughout the sites, but wants these links to display in English regardless of the store view. The developer creates a custom block for use with this template, before rendering sets the translate locale and begins environment emulation using the following code:

```
/** @var $this->_translate \Magento\Framework\TranslateInterface */
$this->_translate->setLocale($newLocaleCode);

/** @var $this->_emulation \Magento\Store\Model\App\Emulation */
$this->_emulation->startEnvironmentEmulation($storeId, \Magento\Framework\App\Area::AREA_FRONTEND);
```

They find that the template text is still being translated into each stores language. Why does this occur?
A. startEnvironmffntEmulation() SetS and locks the locale by Using the setLocale() Optional Second $lock parameter, i.e. setLocale($newLocaleCode, true), to override and lock the locale of the emulated store. If this is set and locked initially then the environment emulation will not be able to override this.
B. startEnvironmentEmuiation() resets the translation locale to the one of the emulated stores, which overrides the locale the developer has set when the order of setLocate and startEnvironmentEmulation is used as displayed above.
C. setLocate() does not change translation locale after it has been initially set, the $this->_translate->emulate($newLocaiecode) method exists to temporarily modify this by pushing the new locale to the top of the current emuiatedLocales stack.

Correct Answer: B
Section:
Explanation:
The startEnvironmentEmulation() method resets the translation locale to the one of the emulated stores, which overrides the locale the developer has set when the order of setLocale() and startEnvironmentEmulation() is used as displayed above.
The correct way to achieve the desired result is to use the emulate() method to temporarily modify the translation locale. The following code shows how to do this:
PHP
$this->_translate->emulate('en_US');
// Render the template
$this->_translate->revert();
This code will set the translation locale to English before rendering the template, and then revert the locale back to the default value after the template has been rendered.
The startEnvironmentEmulation() method is used to emulate a different store view or website. This can be useful for testing purposes, or for developing features that need to work in different environments.
The emulate() method is used to temporarily modify the translation locale. This can be useful for rendering templates in a specific language, or for testing features that need to work in different languages.

QUESTION 15
An Adobe Commerce Developer wishes to add an action to a pre-existing route, but does not wish to interfere with the functionality of the actions from the original route.
What must the developer do to ensure that their action works without any side effects in the original module?
A. In the route declaration, use the before or after parameters to load their module in before or after the original module.
B. Inject the new action into the standard router constructor's $actiomist parameter.
C. Add the action into to the controllers/front_name/ in My.Module, Magento will automatically detect and use it.

Correct Answer: A
Section:
Explanation:
To add an action to a pre-existing route without interfering with the functionality of the original route, the developer must use the before or after parameters in the route declaration. This will load the developer's module in before or after the original module, respectively.
For example, the following code would add an action to the my_module/index route before the action from the original module:
<route id='my_module/index'>
<before>my_module_before</before>
</route>
The my_module_before action would be executed before the MyModule\Controller\Index action, which would allow the developer to perform any necessary setup before the original action is executed.

QUESTION 16
An Adobe Commerce developer is asked to change the tracking level on a custom module for free downloading of pdf and images.
The module contains following models:
Download class has a parameter for tracking_level.
How will the developer configure the tracking_level parameter, in di.xml.to have a value of 4 for Download class and all classes that extend Download?
A)

Configure the parameter on a child class and add `parent` attribute as it will be propagated to siblings and parent.

```
<type
  name="Vendor\FreeDownload\Model\DownloadPdf"
  parent="Vendor\FreeDownload\Model\Download"
>
    <arguments>
        <argument name="tracking_level" xsi:type="integer">4</argument>
    </arguments>
</type>
```

B)

Configure the parameter on the all child classes and set the `parent` attribute on one of them.

```
<type name="Vendor\FreeDownload\Model\DownloadPdf"
  parent="Vendor\FreeDownload\Model\Download">
    <arguments>
        <argument name="tracking_level" xsi:type="number">4</argument>
...
<type name="Vendor\FreeDownload\Model\DownloadImage">
    <arguments>
        <argument name="tracking_level" xsi:type="number">4</argument>
...
```

C)

Configure the parameter on parent class, as it will be propagated on descendant classes.

```
<type name="Vendor\FreeDownload\Model\Download">
    <arguments>
        <argument name="tracking_level" xsi:type="number">4</argument>
    </arguments>
</type>
```

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

Correct Answer: B
Section:
Explanation:
To configure the tracking_level parameter in di.xml to have a value of 4 for the Download class and all classes that extend Download, the developer would use the following code:
<config>
<global>
<models>
<Vendor\FreeDownload\Model\Download>
<setting name='tracking_level' value='4'/>
</Vendor\FreeDownload\Model\Download>
<Vendor\FreeDownload\Model\DownloadPdf>
<rewrite name='tracking_level' value='4'/>
</Vendor\FreeDownload\Model\DownloadPdf>
<Vendor\FreeDownload\Model\DownloadImage>
<rewrite name='tracking_level' value='4'/>
</Vendor\FreeDownload\Model\DownloadImage>
</models>
</global>
</config>
The setting element is used to set a configuration value for a specific model. The rewrite element is used to override the default configuration value for a specific model. In this case, the tracking_level parameter is set to 4 for all models that extend Download.

QUESTION 17
An Adobe Commerce developer is tasked with creating a custom block that will be displayed on every page in the footer of the site.
After completing and optimizing the development, the developer notices that the block takes too much time to be generated on each page and decides to store it in the system cache after enabling it for all cache

groups.
What would be the minimum requirement to achieve this?
A. Set a value for the cache_Lifetime data property of the block.
B. Set a value for cache_key data property of the block.
C. Set values for both cache_lifetime and cache_key data properties of the block.

Correct Answer: C
Section:
Explanation:
To store a block in the system cache, the developer needs to set values for both the cache_lifetime and cache_key data properties of the block. The cache_lifetime property specifies how long the block should be cached, and the cache_key property specifies a unique identifier for the block.
The following code shows how to set the cache_lifetime and cache_key data properties of a block:
PHP
$block->setData('cache_lifetime', 600);
$block->setData('cache_key', 'my_custom_block');
Once the cache_lifetime and cache_key data properties have been set, the block will be stored in the system cache and will not be regenerated on each page load.

QUESTION 18
There is the task to create a custom product attribute that controls the display of a message below the product title on the cart page, in order to identify products that might be delivered late.
The new EAV attribute is_delayed has been created as a boolean and is working correctly in the admin panel and product page.
What would be the next implementation to allow the is_delayed EAV attribute to be used in the .phtml cart page such as $block->getProduct()->getIsDelayed()?
A)
Create a new file etc/catalog_attributes.xmi:

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Catalog:etc/catalog_attributes.x
    <group name="quote_item">
        <attribute name="is_delayed" />
    </group>
</config>
```

B)
Create a new file etc/extension attributes.xmi:

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Api/etc/extension_attributes.xsd"
    <extension_attributes for="Magento\Catalog\Api\Data\ProductRenderInterface>
        <attribute code="is_delayed" type="bool" />
    </extension_attributes>
</config>
```

C)
Create a new file etc/eav attributes.xmi:

```
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Eav:etc/eav_attributes.xsd">
    <entity type="quote_item">
        <attribute code="is_delayed">
            <field code="is_visible" locked="true" />
        </attribute>
    </entity>
</config>
```

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

Correct Answer: A
Section:

Explanation:
To allow the is_delayed EAV attribute to be used in the .phtml cart page, the developer needs to create a new file called etc/catalog_attributes.xmi. This file will contain the definition of the is_delayed attribute.
The following code shows how to create the etc/catalog_attributes.xmi file:
XML
<?xml version='1.0'?>
<catalog_attributes>
<label>Is Delayed</label>
<note>This attribute indicates whether the product is delayed.</note>
<sort_order>10</sort_order>
<required>false</required>
</catalog_attributes>
Once the etc/catalog_attributes.xmi file has been created, the is_delayed attribute will be available in the .phtml cart page. The attribute can be accessed using the getIsDelayed() method of the Product class.
PHP
$product = $block->getProduct();
$isDelayed = $product->getIsDelayed();
The isDelayed variable will contain the value of the is_delayed attribute. If the value of the attribute is 1, then the product is delayed. If the value of the attribute is 0, then the product is not delayed.

QUESTION 19
An Adobe Commerce developer is creating a module (Vendor.ModuleName) to be sold on the Marketplace. The new module creates a database table using declarative schema and now the developer needs to make sure the table is removed when the module is disabled.
What must the developer do to accomplish this?
A. There is nothing further the developer needs to do. The table will be removed when the module is disabled and bin/magento setup:upgrade is run.
B. There is nothing further the developer needs to do. The table will be removed when the when bin/magento module:uninstall vendor_ModuleName is run.
C. Add a schema patch that implements Magento\Framework\setup\Patch\PatchRevertabieinterface and drops the table in the revert function.

Correct Answer: C
Section:
Explanation:
According to the Declarative Schema Overview guide for Magento 2 developers, declarative schema is a new feature that allows developers to declare the final desired state of the database and has the system adjust to it automatically, without performing redundant operations. However, declarative schema does not support uninstalling modules or reverting changes. To remove a table when a module is disabled, the developer needs to add a schema patch that implements Magento\Framework\setup\Patch\PatchRevertabieinterface and drops the table in the revert function. The revert function will be executed when the module is disabled using bin/magento module:disable command. Verified
Reference: https://devdocs.magento.com/guides/v2.3/extension-dev-guide/declarative-schema/

QUESTION 20
An Adobe Commerce developer is trying to create a custom table using declarative schema, but is unable to do so.

```
<table name="student_details" resource="default" engine="innodb" comment="Students Detail Table">
       <column xsi:type="int" name="entity_id" padding="10" unsigned="true" nullable="false" identity="false"
               comment="Entity Id"/>
       <column xsi:type="smallint" name="roll_no" padding="2" unsigned="true" nullable="false"
               identity="true" default="null" comment="Student Roll No"/>
       <column xsi:type="text" name="student_name" nullable="false" comment="Student Name"/>
       <column xsi:type="varchar" name="student_class" length="5" nullable="false" comment="Student Class"/>
   </table>
```

What are two errors in the snippet above? (Choose two.)
A. Column (roll_no) does not have index. It is needed since attribute identity is set to true.
B. Column (entity_id) does not have index. It is needed since attribute identity is set to false.
C. Column (student_name) does not have attribute length.
D. null is not a valid value for column (roll_no).

Correct Answer: A, C
Section:
Explanation:
The correct answers are A and C.
The errors in the snippet are:
Columnroll_nodoes not have an index. It is needed sinceattribute_identityis set totrue.
Columnstudent_namedoes not have an attribute length.
The attribute_identity attribute specifies whether the primary key of the table should be auto-incremented. If attribute_identity is set to true, then the roll_no column must have an index. The student_name column does not have an attribute length, which is required for string columns.
The following code shows how to fix the errors:
XML
<table name='vendor_module_table'>
<entity_id>
<type>int</type>
<identity>true</identity>
<unsigned>true</unsigned>
<nullable>false</nullable>
</entity_id>
<roll_no>
<type>int</type>
<identity>false</identity>
<unsigned>true</unsigned>
<nullable>false</nullable>
true
<index>true</index>
</roll_no>
<student_name>
<type>string</type>
<length>255</length>
<nullable>false</nullable>
</student_name>
</table>
Once the errors have been fixed, the table can be created successfully.

QUESTION 21
An Adobe Commerce developer wants to create a product EAV attribute programmatically which should appear as WYSIWYG in the admin panel. They have made sure that wysiwyg_enabled has been set to true, however, the attribute is not appearing as WYSIWYG in the admin panel.
What would be a possible reason?
A. The is_html_allowed_on_front Option iS Set tO false.
B. The input type is not set to text.
C. The input type is not set to textarea.

Correct Answer: C
Section:
Explanation:
The input_type attribute of a product EAV attribute specifies the type of input field that will be used to enter the value of the attribute in the admin panel. The textarea input type is used for WYSIWYG fields. If the input_type attribute is not set to textarea, then the attribute will not appear as WYSIWYG in the admin panel.
To fix this, the developer should set the input_type attribute to textarea.