SKILLCERTPRO
IT CERTIFICATION TRAININGS

Cart/₹0.00 0

/ Salesforce / By SkillCertPro

Salesforce Platform Developer 1 Exam Questions Total Questions: 915 – 15 Mock Exams & 1 Master Cheat Sheet

Practice Set 1

Your results are here!! for" Salesforce Platform Developer 1 Practice Test 1 [New] "

0 of 50 questions answered correctly

Your time: 00:00:06

Your Final Score is : 0
You have attempted : 0
Number of Correct Questions : 0 and scored 0
Number of Incorrect Questions : 0 and Negative marks 0

You can review your answers by clicking view questions.
Important Note : Open Reference Documentation Links in New Tab (Right Click and Open in New Tab).

Restart Test View Answers

Review question | Pause | Summary

1. Question

Which SOQL query successfully returns the Accounts grouped by name?

- SELECT type,Max(CreatedDate) FROM Account GROUP BY NAME
- **SELECT Name,Max(CreatedDate) FROM Account GROUP BY Name**
- SELECT Id,Type,Max(CreatedDate) FROM Account GROUP BY Name
- SELECT TYpe,Name, Max(CreatedDate) FROM Account GROUP BY Name LIMIT 5

Unattempted
Correct:

✅ B. `SELECT Name,Max(CreatedDate) FROM Account GROUP BY Name` This query is correct. When using `GROUP BY`, you can select the grouping field (in this case, `Name`) and aggregate functions (like `MAX()`, `MIN()`, `AVG()`, `SUM()`, `COUNT()`) on other fields.

Incorrect:

❌ A. `SELECT type,Max(CreatedDate) FROM Account GROUP BY NAME` This is incorrect. You are grouping by `Name`, but trying to select `type` without either grouping by `type` as well or using an aggregate function on it. If you want to include `Type` in the `SELECT` clause, you must also include it in the `GROUP BY` clause.

❌ C. `SELECT Id,Type,Max(CreatedDate) FROM Account GROUP BY Name` This is also incorrect for the same reason as option A. You're grouping by `Name`, but selecting `Id` and `Type` without including them in the `GROUP BY` clause or using an aggregate function.

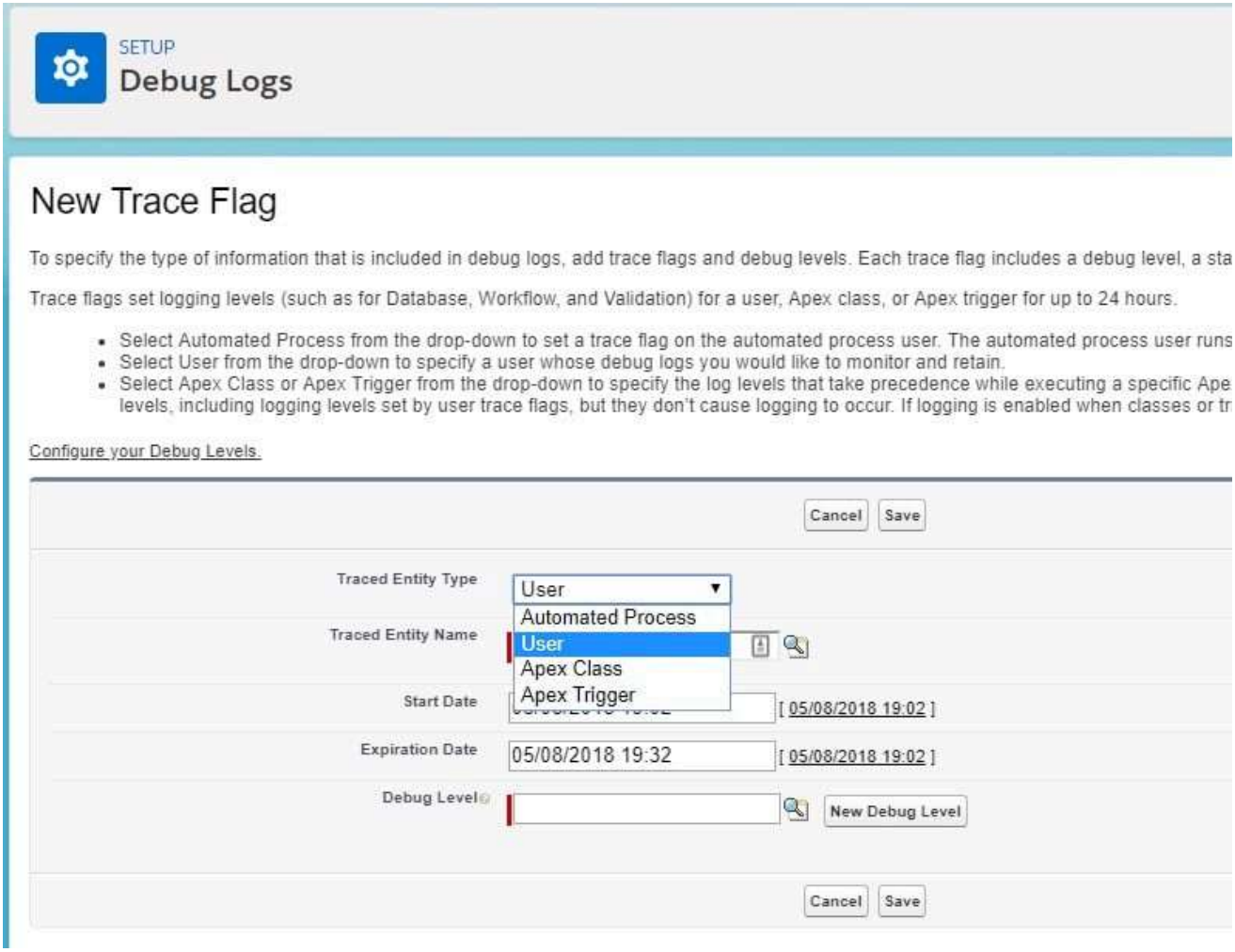
❌ D. `SELECT TYpe,Name, Max(CreatedDate) FROM Account GROUP BY Name LIMIT 5` This is incorrect. While `LIMIT` is allowed in SOQL queries, it cannot be used with `GROUP BY` clauses. If you want to limit the results, you would need to use a subquery or filter the results *before* grouping.

2. Question

For which three items can trace flag be configured? (Choose three)

- [ ] Apex Trigger
- [ ] Apex Class
- [ ] Process Builder
- [ ] User
- [ ] Visualforce

Unattempted



3. Question

Which of the following is a best practice when writing test classes? Choose 1 answer

- [ ] Use System.assert() methods to verify test results
- [ ] Use SeeAllData=true
- [ ] Use System.debug() method to verify test results
- [ ] Use @future annotation

Unattempted

Salesforce Apex Test Class Best Practices

1)Do not put (seeAllData = true) in test class otherwise, use it for exceptional cases.

2) Use @isTest at the Top for all the test classes.

3) Test in bulk: Test to see if your code can run on 200 records at once. This is the maximum number of records that can be evaluated in a single trigger by Salesforce.

4)Avoid Using Hard Coding Ids anywhere in test Class or any apex class.

5) Use System.assertEquals() to see if your code has the expected outcomes.

6) Use Test.startTest() and Test.stopTest() statement to increase Governor Limits of your test code.

4. Question

Which of the following ways can be used to throw a custom exception? Choose 3 answers

- [ ] **throw new myCustomException(e)**
- [ ] **throw new myCustomException('Error Message Here');**
- [ ] throw new myCustomException().addError('Error Message Here');
- [ ] **throw new myCustomException();**
- [ ] throw new myCustomException().setMessage('Error Message Here');

Unattempted

You have to create your own Exception class like this:

public class applicationException extends Exception {}

Then you can throw that:

throw new applicationException(You can't do that here');

5. Question

Which of the following statements about the IF-ELSE statement are true? Choose 2 answers

- [ ] The IF-ELSE statement provides a secondary path of execution when an IF clause evaluates to true.
- [ ] **An IF statement can be followed by a discretionary ELSE statement, which executes when the Boolean expression is false**

- [ ] An IF-ELSE statement can have a number of possible execution paths
- [ ] **The IF-ELSE statement permits a choice to be made between two possible execution paths**

Unattempted

Correct:

✅ B. An IF statement can be followed by a discretionary ELSE statement, which executes when the Boolean expression is false. The `ELSE` part of an `IF-ELSE` statement is optional (discretionary). If the `IF` condition is false, and an `ELSE` block is present, the code within the `ELSE` block will execute.

✅ D. The IF-ELSE statement permits a choice to be made between two possible execution paths. This is the fundamental purpose of an `IF-ELSE` statement: to execute one block of code if a condition is true and a different block of code if the condition is false, thus providing two distinct execution paths.

Incorrect:

❌ A. The IF-ELSE statement provides a secondary path of execution when an IF clause evaluates to *true*. This is incorrect. The `ELSE` block provides the secondary path when the `IF` clause evaluates to *false*. When the `IF` condition is *true*, the `IF` block itself executes.

❌ C. An IF-ELSE statement can have a number of possible execution paths. An `IF-ELSE` statement has *exactly two* possible execution paths: one for when the condition is true and one for when the condition is false. While you can *nest* `IF-ELSE` statements to create more complex logic with multiple paths, a *single* `IF-ELSE` statement itself only has two.

6. Question

Which static methods can be used in a test class to test governor limits? Choose 1 answer

- ○ Test.runAs() and Test.getLimits()
- ○ **Test.startTest() and Test.stopTest()**
- ○ Test.setFixedSearchResults()
- ○ Test.isTest() and Test.isnotTest()

Unattempted

Test.startTest/Test.stopTest

There are two additional system static methods provided by Apex. These methods, Test.startTest and Test.stopTest, are used when testing governor limits. Or in other words, executing test scenarios with a larger data set.

The Test.startTest method marks the point in your test code when your test actually begins. Each test method is allowed to call this method only once. All of the code before this method should be used to initialize variables, populate data structures, and so on, allowing you to set up everything you need in order to run your test. After you call this method, you get a fresh set of governor limits for the remainder of the test until you call Test.stopTest.

The Test.stopTest method marks the point in your test code when your test ends. Use this method in conjunction with thestartTest method. Each test method is allowed to call this method only once. After calling this method, any post assertions are done in the original context

```
@isTest

public class sampleTestMethodCls
{
private static testMethod void testAccountTrigger()
{
//First, prepare 200 contacts for the test data
Account acct = new Account(name='test account');

insert acct;

Contact[] contactsToCreate = new Contact[]{};

for(Integer x=0; x<200;x++)
{
Contact ct = new Contact(AccountId=acct.Id,lastname='test');
contactsToCreate.add(ct);
}

//Now insert data causing an contact trigger to fire.

Test.startTest();

insert contactsToCreate;

Test.stopTest();
}
}
```

7. Question

Cool Air Conditioners has been using Service cloud to manage cases, but are now considering using it to manage field service jobs. They would like to track field service activity and assignment to technicians. What is the recommended solution to meet these requirements? Choose 1 answer

- Install an AppExchange product that provides Field Service functionality
- Extend the Service Cloud configuration to handle Field Service cases
- **Utilize the standard objects Work Order and Work Order Line Items**
- Use real time API integration to connect Salesforce with an external field service application

Unattempted

Correct:

✅ **C. Utilize the standard objects Work Order and Work Order Line Items**

Explanation:

Salesforce provides native field service functionality using the standard objects **Work Order** and **Work Order Line Items**. These objects are designed specifically to manage field service jobs, track work assignments, and manage tasks related to service appointments. By using these standard objects, Cool Air Conditioners can effectively track field service activity and assignment to technicians within the Salesforce ecosystem, leveraging the built-in capabilities of Service Cloud.

Incorrect:

❌ **A. Install an AppExchange product that provides Field Service functionality**

While installing an AppExchange product may be an option, it is not the most recommended solution if the goal is to use native Salesforce features. The standard Work Order and Work Order Line Items offer an integrated solution without needing third-party products. AppExchange products might provide additional features but may add complexity and cost.

❌ **B. Extend the Service Cloud configuration to handle Field Service cases**

Extending Service Cloud to handle field service cases is not the most efficient approach. Service Cloud is primarily designed for managing customer support cases and not specifically for managing field service tasks. Using the Work Order and Work Order Line Items objects is a more streamlined approach for field service.

❌ **D. Use real-time API integration to connect Salesforce with an external field service application**

While using API integration to connect Salesforce with an external application is possible, it is not necessary if Salesforce's native field service functionality (Work Order and Work Order Line Items) already meets the business requirements. Relying on external applications may complicate the setup and maintenance.

8. Question

A developer is required to create a trigger that every time the Type field on Request object is updated, the Owner field should be changed as well, either to a User or Queue. What trigger event should the developer use? Choose 1 answer

- After Update
- **Before Update**

- Before Delete
- After Merge

Unattempted

What is Triggers in Salesforce?

A trigger is an Apex script that executes before or after data manipulation language (DML) events occur. Apex triggers enable you to perform custom actions before or after events to record in Salesforce, such as insertions, updates, or deletions. Just like database systems support triggers, Apex provides trigger support for managing records.

Syntax:

```
Trigger on (trigger Events) {
// Implement the Logic here
}
```

What are the various event on which a trigger can fire?

A trigger is a set of statement which can be executed on the following events. In above trigger events one or more of below events can be used with comma-separated.

Here is a list of events in Salesforce which can fire trigger

before insert

before update

before delete

after insert

after update

after delete

after undelete

What are different type of Triggers?

There are two types of triggers:

Before triggers are used to perform a task before a record is inserted or updated or deleted. These are used to update or validate record values before they are saved to the database.

After triggers are used if we want to use the information set by Salesforce system and to make changes in the other records. are used to access field values that are set by the system (such as a record's Id or LastModifiedDate field), and to affect changes in other records. The records that fire the after trigger are read-only.

9. Question

A developer needs to create a newContact record in his Apex Trigger. What DML statement should be used in the code? Choose 1 answer

- Merge
- **Insert**
- Upsert
- Create

Unattempted

Apex DML Statements

Use Data Manipulation Language (DML) statements to insert, update, merge, delete, and restore data in Salesforce.

The following Apex DML statements are available:

Insert Statement

The insert DML operation adds one or more sObjects, such as individual accounts or contacts, to your organization's data. insert is analogous to the INSERT statement in SQL.

```
Account newAcct = new Account(name = 'Acme');
try {
insert newAcct;
} catch (DmlException e) {
// Process exception here
}
```

Update Statement

The update DML operation modifies one or more existing sObject records, such as individual accounts or contactsinvoice statements, in your organization's data. update is analogous to the UPDATE statement in SQL.

```
Account a = new Account(Name='Acme2');
insert(a);

Account myAcct = [SELECT Id, Name, BillingCity FROM Account WHERE Id = :a.Id];
myAcct.BillingCity = 'San Francisco';

try {
update myAcct;
} catch (DmlException e) {
// Process exception here
}
```

Upsert Statement

The upsert DML operation creates new records and updates sObject records within a single statement, using a specified field to determine the presence of existing objects, or the ID field if no field is specified.

Delete Statement

The delete DML operation deletes one or more existing sObject records, such as individual accounts or contacts, from your organization's data. delete is analogous to the delete() statement in the SOAP API.

```
Account[] doomedAccts = [SELECT Id, Name FROM Account
WHERE Name = 'DotCom'];
try {
delete doomedAccts;
} catch (DmlException e) {
// Process exception here
}
```

10. Question

A developer would like to relate an external data object (Social Media Posts) to the contacts object in Salesforce to track every post the contact has made in the external platform. How can the developer achieve this? Choose 1 answer

- Create an external lookup relationship using a custom field with External ID and Unique attributes
- **Create an indirect lookup relationship using a custom field with External ID and Unique attributes**
- Create a lookup relationship and update the record ID through integration
- Create a master-detail relationship and update the record ID through integration

Unattempted

The best approach for a developer to relate an external data object (Social Media Posts) to the contacts object in Salesforce is to:

Create an indirect lookup relationship using a custom field with External ID and Unique attributes.

Here's why:

- **Indirect Lookup:**
  - Since Social Media Posts likely exist outside of Salesforce, a direct lookup from the Post object to Contact is not feasible.
  - An indirect approach allows you to create a bridge between the two systems.
- **Custom Field with External ID and Unique Attributes:**
  - Create a custom field on the Contact object (e.g., "Social Media User ID") to store the unique identifier of the contact on the external platform (e.g., Twitter handle, Facebook ID).
  - Mark this field as "External ID" and "Unique" to ensure data integrity and enforce the uniqueness of the identifier within Salesforce.
- **Integration:**
  - Implement an integration (e.g., using Salesforce Connect, APIs, or an integration tool) to:
    - Retrieve Social Media Posts from the external platform.
    - Match each Post to the corresponding Contact in Salesforce based on the "Social Media User ID".
    - Store the relevant information from the Post (e.g., post text, date, etc.) in a custom object in Salesforce.

Why not other options:

- **External Lookup Relationship:** While possible, it might be more complex to implement and maintain, especially if the external system is not directly supported by Salesforce.
- **Lookup Relationship and Update Record ID:** This would require modifying the external system's data, which is generally not recommended.
- **Master-Detail Relationship:** Not appropriate in this scenario as Social Media Posts are not directly owned by the Contact in the traditional sense.

This approach provides a flexible and maintainable solution for tracking Social Media Posts related to Salesforce Contacts.

11. Question

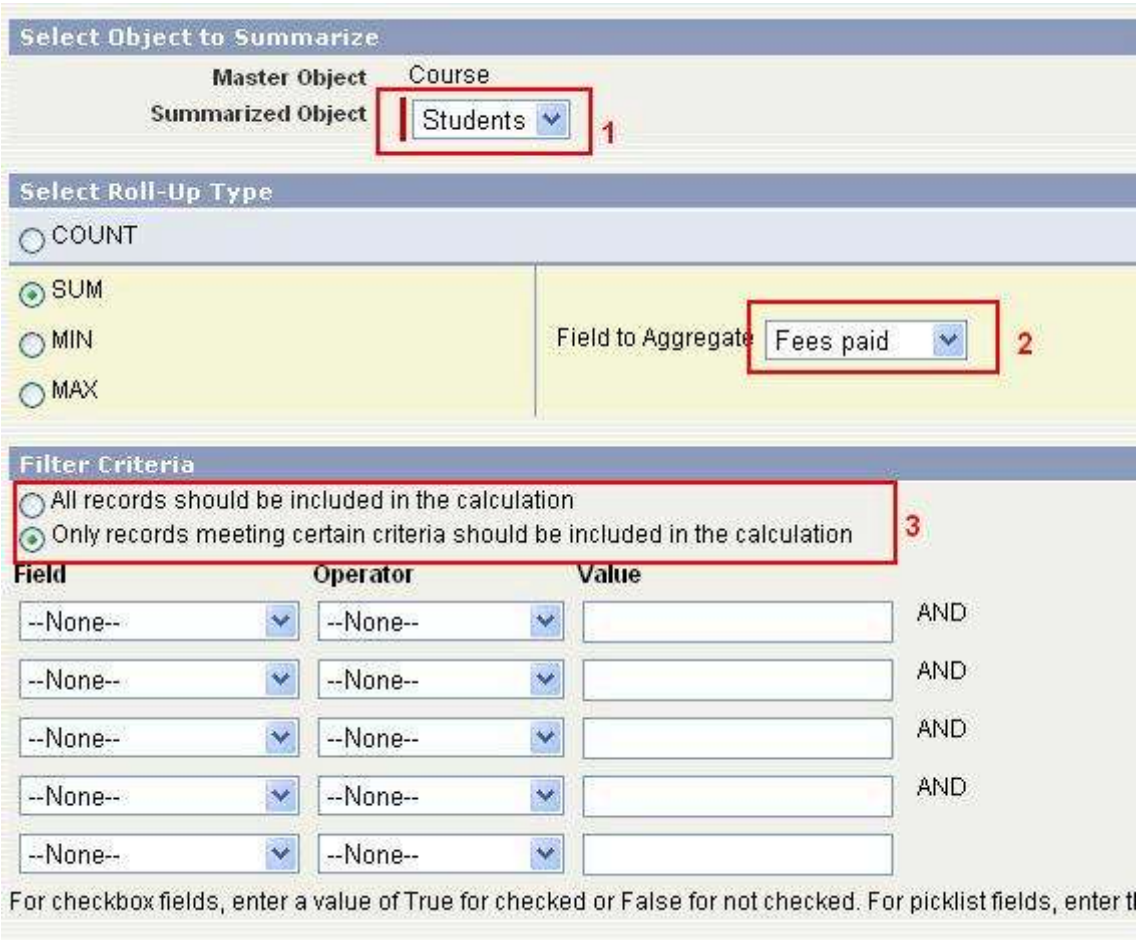
Which of the following field types can a Roll-Up Summary field calculate? Choose 1 answer

- Number

- Text
- Checkbox
- Picklist

Unattempted

A roll-up summary field calculates values from related records, such as those in a related list. You can create a roll-up summary field to display a value in a master record based on the values of fields in a detail record. The detail record must be related to the master through a master-detail relationship.



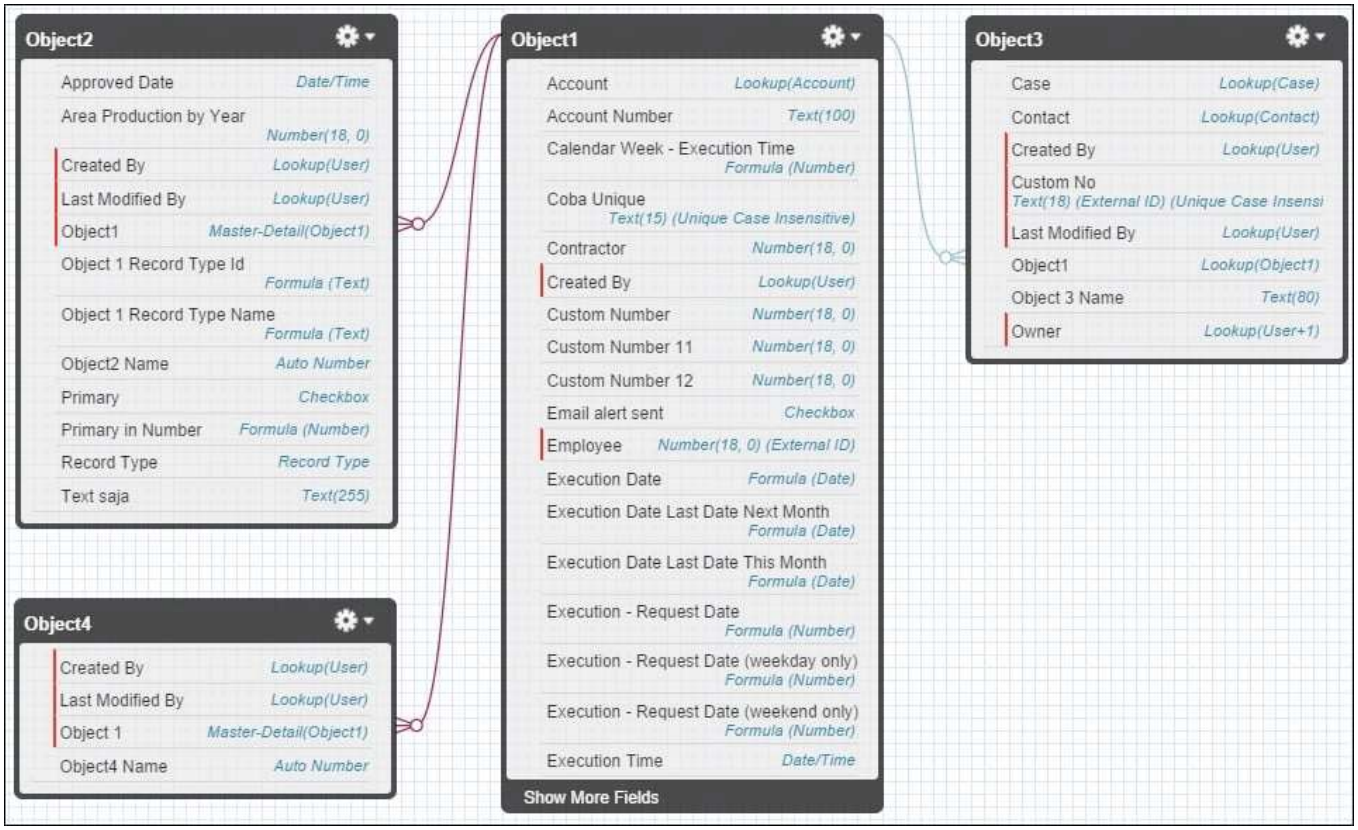
12. Question

Which of the following are capabilities of Schema Builder ? Choose 3 answers

- Creating a custom object
- Deleting a custom object
- Creating lookup and master-detail relationships
- Exporting schema definition
- Importing schema definition

Unattempted

Schema Builder is a tool within Salesforce.com that is used to view and manage objects, fields, and relations between objects in a graphical interface.



Here are a few actions you can perform with Schema Builder:

-Create and delete custom objects

-Edit custom object properties

-Create and delete custom fields

-Edit custom field properties

-Manage custom field permissions

13. Question

Which of the following statements are true about controller extensions? Choose 3 answers

- [ ] **A controller extension is an Apex class that extends the functionality of a standard or custom controller**
- [ ] Only one controller extension can be defined for a single page
- [ ] **The extension is associated with the page using the 'extensions' attribute of the component.**

- [ ] If an extension works in conjunction with a standard controller, the standard controller methods will also be available
- [ ] A standard or custom controller is an extension of the controller extension

Unattempted

Controller extension in Salesforce is an Apex class containing a constructor that is used to add or extend the functionalities of a Standard Controller or custom controller in Salesforce.

14. Question

What options are available to run unit tests in an org? Choose 3 answers

- [ ] Run all methods in a specific class that does not contain test methods
- [ ] Run all unit tests in an org
- [ ] Run a test suite
- [ ] Run all methods in a specific class

Unattempted

Correct:

✅ B. Run all unit tests in an org. You can execute all test classes and methods within your Salesforce org at once. This is a common practice, especially during deployments or release cycles.

✅ C. Run a test suite. Test suites allow you to group specific test classes together and run them as a set. This is useful for organizing and running related tests.

✅ D. Run all methods in a specific class. You can run all the test methods within a single test class. This allows you to focus on a particular area of your code.

Incorrect:

❌ A. Run all methods in a specific class that does not contain test methods. You cannot run methods in a class that does *not* contain test methods. Only classes annotated with `@isTest` and containing methods annotated with `@testMethod` (or the older `testSetup` and `testMethod` keywords) will be considered for test execution. If a class has no test methods, it won't be executed by the test runner.

15. Question

Which of the following best describe the Lightning Component Framework? Choose 2 answers

- [ ] It has an event-driven architecture

- [ ] **It is device-aware and supports cross-browser compatibility**
- [ ] It automatically upgrades all pre-existing Visualforce pages and components
- [ ] It require the Aura Components model to build Lightning components

Unattempted

✅ **It has an event-driven architecture** The Lightning Component Framework utilizes an event-driven architecture, allowing components to communicate with each other by firing and handling events. This promotes decoupling and better organization of the application.

✅ **It is device-aware and supports cross-browser compatibility** Lightning components are designed to be responsive and work seamlessly across various devices (desktops, tablets, phones) and modern web browsers without requiring developers to write separate code for each.

❌ It automatically upgrades all pre-existing Visualforce pages and components The Lightning Component Framework and Visualforce are different UI frameworks within Salesforce. Lightning components do not automatically upgrade or replace existing Visualforce pages and components. Developers need to build new UI elements using the Lightning framework if they want to leverage its benefits.

❌ It requires the Aura Components model to build Lightning components While Aura Components was the original programming model for the Lightning Component Framework, Salesforce introduced Lightning Web Components (LWC), which is a newer programming model that leverages web standards. Developers can build Lightning components using either Aura Components or LWC. Therefore, requiring only Aura Components is incorrect.

16. Question

A developer requires a variable numberOfStudents with a constant value of 25 that is accessible only within the Apex class in which it is defined. Which of the following is the best variable declaration? Choose 1 answer

- ○ protected final Integer numberOfStudents =25;
- ○ global static Integer numberOfStudents =25;
- ○ **private static final Integer numberOfStudents=25;**
- ○ public Integer numberOfStudents =25;

Unattempted

Correct:

✅ C. `private static final Integer numberOfStudents = 25;` This is the best declaration.

- `private`: Ensures the variable is only accessible within the class.

- `static`: Makes the variable belong to the class itself, not instances of the class. This is appropriate for a constant value.
- `final`: Makes the variable a constant; its value cannot be changed after initialization.
- `Integer`: Specifies the data type as an integer.

Incorrect:

✗ A. `protected final Integer numberOfStudents = 25;` `protected` allows access from subclasses, which is not required. The requirement is that the variable should be accessible *only* within the class, so `private` is more appropriate.

✗ B. `global static Integer numberOfStudents = 25;` `global` makes the variable accessible across all Apex code, which is much broader than the requirement. `private` is much more restrictive and thus is more appropriate for this requirement.

✗ D. `public Integer numberOfStudents = 25;` `public` makes the variable accessible from anywhere, which is again, much broader than required. It also doesn't make the variable a constant (it can be changed). `private` and `final` are both missing.

17. Question

A junior developer frequently experiences governor limit errors when running Apex Triggers. As a senior developer, which of the following are best practices that you could advise? Choose 2 answers

- ☐ **Use a combination of collections (e.g. maps, lists) and streamlined queries.**
- ☐ Use an Apex handler class for multiple triggers
- ☐ **Use lists to perform DML operations on multiple records**
- ☐ Use SOQL queries only within FOR loops

Unattempted
Correct:

✅ A. Use a combination of collections (e.g., maps, lists) and streamlined queries. This is *crucial* for avoiding governor limits. Using collections (especially Maps) allows you to process large numbers of records efficiently without repeatedly querying the database inside loops. Streamlined queries (using `WHERE` clauses effectively, selecting only necessary fields) minimize the amount of data retrieved, reducing query rows and SOQL limits.

✅ C. Use lists to perform DML operations on multiple records. Performing DML operations (inserts, updates, deletes) on lists of records in a single call is *much* more efficient than performing DML operations inside a loop, one record at a time. This reduces the number of DML statements, which is a common governor limit.

Incorrect:

❌ B. Use an Apex handler class for multiple triggers. While using a handler class is a good practice for *code organization* and *maintainability*, it does *not* directly address governor limits. It can make your code *easier* to optimize, but it doesn't automatically solve governor limit issues. The core problem (inefficient queries and DML) still needs to be addressed within the handler class.

❌ D. Use SOQL queries only within FOR loops. This is *exactly what you should *avoid*. Placing SOQL queries *inside* for loops (especially when processing a large number of records) is a *classic* cause of governor limit errors. This leads to many queries being executed, quickly hitting the SOQL query limit. You should always try to move SOQL queries *outside* of loops whenever possible.

18. Question

Which of the following use cases are valid for declarative customization ? Choose 3 answers

- [ ] **Determining a lead rating, based on checking the value of 3 lead fields**
- [ ] **Displaying the number of employees of the account related to an opportunity, on the opportunity page layout**
- [ ] Calculating the sales tax applicable to a quote, that is a complex calculation and based on various factors such as product, state, quantity
- [ ] **Calculating the number of days until an opportunity closes and displaying on a report**

Unattempted

Correct:

✅ A. Determining a lead rating, based on checking the value of 3 lead fields. This can be easily achieved using a *workflow rule* or *Process Builder* (or Flow). These declarative tools allow you to define criteria based on field values and then perform actions (like updating a lead rating field) without writing any code.

✅ B. Displaying the number of employees of the account related to an opportunity, on the opportunity page layout. A *formula field* on the Opportunity object can accomplish this. You would create a cross-object formula that looks up the number of employees field on the related Account.

✅ D. Calculating the number of days until an opportunity closes and displaying on a report. This can be done with a *formula field* on the Opportunity (calculating the difference between the close date and today's date). Formula fields are available on reports as well.

Incorrect:

❌ C. Calculating the sales tax applicable to a quote, that is a complex calculation and based on various factors such as product, state, quantity. While *simple* calculations can be done with formula fields, *complex* calculations involving various factors often require more sophisticated logic than what formula fields can

provide. This is a case where Apex (or possibly a Flow with extensive logic) would be more appropriate. Declarative tools are not ideal for complex calculations.

19. Question

A developer needs to create a trigger that will throw an error whenever the user tries to delete a contact that is not associated to an account. What trigger event should the developer use? Choose 1 answer.

- Before Delete
- After Insert
- After Delete
- Before Insert

Unattempted
Correct:

✅ A. Before Delete. This is the correct trigger event. You want to prevent the deletion if the Contact is not associated with an Account. Therefore, you need to check the relationship *before* the Contact is actually deleted. A `before delete` trigger allows you to do this and throw an exception to prevent the deletion.

Incorrect:

❌ B. After Insert. An `after insert` trigger fires *after* a record is inserted. This is irrelevant to the scenario where you're trying to prevent deletion.

❌ C. After Delete. An `after delete` trigger fires *after* a record is deleted. At this point, it's too late to prevent the deletion. You can't "undelete" a record using a trigger.

❌ D. Before Insert. A `before insert` trigger fires *before* a record is inserted. Again, this is not relevant to the requirement of preventing the deletion of a Contact.

20. Question

Which of the following are capabilities of Visual Studio Code? Choose 3 answers.

- [ ] Creating Change Sets
- [ ] Running Apex Tests
- [ ] Deploying metadata components from org to another
- [ ] Create a test suite for running tests
- [ ] Executing SOQL queries

Unattempted

Correct:

✅ B. Running Apex Tests. The Salesforce Extension Pack for VS Code allows you to run Apex tests directly within the IDE. You can run individual tests, all tests in a class, or all tests in your workspace.

✅ C. Deploying metadata components from one org to another. VS Code with the Salesforce Extension Pack provides commands and features for deploying metadata (Apex classes, Visualforce pages, etc.) between Salesforce orgs.

✅ E. Executing SOQL queries. You can execute SOQL queries and view the results directly within VS Code using the Salesforce Extension Pack. This is a very convenient feature for data exploration and testing.

Incorrect:

❌ A. Creating Change Sets. Change sets are a feature of the Salesforce Setup UI, not VS Code. You create and manage change sets within the Salesforce web interface.

❌ D. Create a test suite for running tests. While you can *run* test suites from VS Code, the *creation* and *definition* of test suites is done within the Salesforce Setup UI. VS Code doesn't have a built-in mechanism for defining test suites. You'd create the suite in Setup, and then VS Code can run it.

21. Question

Which of the following are part of the model layer in the MVC model? Choose 3 answers

- [ ] Fields
- [ ] Objects
- [ ] Tabs
- [ ] Page Layouts
- [ ] Relationships

Unattempted

Correct:

✅ A. Fields. Fields are part of the model layer. They define the data that an object can store.

✅ B. Objects. Objects are a fundamental part of the model layer. They represent the structure and type of data (e.g., Account, Contact, Opportunity).

✅ E. Relationships. Relationships between objects (e.g., lookup, master-detail) are also part of the model layer. They define how different pieces of data are connected and related to each other.

Incorrect:

❌ C. Tabs. Tabs are part of the *view* layer. They control how users navigate between different parts of the application.

❌ D. Page Layouts. Page layouts are also part of the *view* layer. They define how the data (from the model) is presented to the user on a page.

22. Question

Which standard objects in the following list are not supported by DML operations? Choose 2 answers

- [ ] **Record Type**
- [ ] **Profile**
- [ ] User
- [ ] Opportunity Line Item

Unattempted

Correct:

✅ **A. Record Type**

The **Record Type** object is not supported by DML operations because it is a metadata object used to categorize and manage business processes. You cannot perform DML operations such as insert, update, delete, or undelete on the Record Type object directly in Apex.

✅ **B. Profile**

The **Profile** object is also not supported by DML operations. Profiles are metadata that define user permissions and access settings. Since profiles are part of Salesforce's security model, DML operations cannot be used directly on the Profile object. Any changes to profiles require metadata deployment through tools like the Metadata API, not DML.

Incorrect:

❌ **C. User**

The **User** object can be manipulated using DML operations. You can insert, update, delete, or undelete records for users in Salesforce via Apex or the API, as long as proper permissions are in place.

❌ **D. Opportunity Line Item**

The **Opportunity Line Item** (also known as Opportunity Product) is supported by DML operations. You can

perform insert, update, and delete operations on Opportunity Line Items in Salesforce, as they are standard objects related to the Opportunity object used to track products associated with an opportunity.

23. Question

When writing an Apex Trigger,what should a developer keep in mind?Choose 2 answers

- [ ] An Apex Trigger should not cause another trigger to be fired
- [ ] An Apex Trigger should use @future annotation in performing DML operations
- [ ] **A single Apex Trigger is all you need for each object**
- [ ] **An Apex Trigger should be logic-less and delegate the logic responsibilities to a handler class**

Unattempted

Best Practices to follow using Triggers:

Best Practice #1: Bulkify your Code

Bulkifying Apex code refers to the concept of making sure the code properly handles more than one record at a time. When a batch of records initiates Apex, a single instance of that Apex code is executed, but it needs to handle all of the records in that given batch.

Best Practice #2: Avoid SOQL Queries or DML statements inside FOR Loops

There is a governor limit that enforces a maximum number of SOQL queries. There is another that enforces a maximum number of DML statements (insert, update, delete, undelete). When these operations are placed inside a for loop, database operations are invoked once per iteration of the loop making it very easy to reach these governor limits.

Instead, move any database operations outside of for loops. If you need to query, query once, retrieve all the necessary data in a single query, then iterate over the results. If you need to modify the data, batch up data into a list and invoke your DML once on that list of data.

Best Practice #3: Using Collections, Streamlining Queries, and Efficient For Loops

It is important to use Apex Collections to efficiently query data and store the data in memory. A combination of using collections and streamlining SOQL queries can substantially help writing efficient Apex code and avoid governor limits.

Best Practice #4: Streamlining Multiple Triggers on the Same Object

Best Practice #5: Querying Large Data Sets

The total number of records that can be returned by SOQL queries in a request is 50,000. If returning a large set of queries causes you to exceed your heap limit, then a SOQL query for loop must be used instead. It can process multiple batches of records through the use of internal calls to query and queryMore.

Best Practice #6: Use of the Limits Apex Methods to Avoid Hitting Governor Limits

Apex has a System class called Limits that lets you output debug messages for each governor limit.There are two versions of every method: the first returns the amount of the resource that has been used in the current context, while the second version contains the word limit and returns the total amount of the resource that is available for that context.

24. Question

When would a full copy sandbox be required ? Choose 2 answers

- [ ] **Performance and Load Testing**
- [ ] Integration Testing
- [ ] **Staging**
- [ ] Training

Unattempted

Full Sandbox

Full sandboxes copy your entire production organization and all its data, including standard and custom object records, documents, and attachments. You can refresh a Full sandbox every 29 days.

Sandbox templates allow you to pick specific objects and data to copy to your sandbox, so you can control the size and content of each sandbox. Sandbox templates are only available for Partial Data or Full sandboxes.

REFRESH LIMIT :- 29 Days

DATA LIMIT :- Same as Production

A Full sandbox is intended to be used as a testing environment. Only Full sandboxes support performance testing, load testing, and staging.

25. Question

What method can be used to obtain metadata information about all the sObjects in an organization and their fields? Choose 1 answer

describeObjects()

getGlobalDescribe()

getGlobalSObjects()

describeSObjects()

Unattempted

Schema Class

Contains methods for obtaining schema describe information.

Schema Methods

The following are methods for Schema. All methods are static.

getGlobalDescribe()
Returns a map of all sObject names (keys) to sObject tokens (values) for the standard and custom objects defined in your organization.

describeDataCategoryGroups(sObjectNames)
Returns a list of the category groups associated with the specified objects.

describeSObjects(sObjectTypes)
Describes metadata (field list and object properties) for the specified sObject or array of sObjects.

describeTabs()
Returns information about the standard and custom apps available to the running user.

GroupStructures(pairs)
Returns available category groups along with their data category structure for objects specified in the request.

26. Question

Which data type is appropriate for a numerical value of 726.234 stored in the variable totalCost? Choose 1 answer

Numeric totalCost;

Long totalCost;

Decimal totalCost;

Integer totalCost;

Unattempted

Understanding the Data Types

The Apex language is strongly typed so every variable in Apex will be declared with the specific data type.

Apex supports the following data types ?

-Primitive (Integer, Double, Long, Date, Datetime, String, ID, or Boolean)

-Collections (Lists, Sets and Maps)

-sObject

-Enums

-Classes, Objects and Interfaces

Let's cover Primitive data types:

Integer

A 32-bit number that does not include any decimal point.

```
Integer barrelNumbers = 1000;
system.debug(' value of barrelNumbers variable: '+barrelNumbers);
```

Boolean

This variable can either be true, false or null. Many times, this type of variable can be used as flag in programming to identify if the particular condition is set or not set.

```
Boolean shipmentDispatched=true;
System.debug('Value of shipmentDispatched '+shipmentDispatched);
```

Date

This variable type indicates a date. This can only store the date and not the time. For saving the date along with time, we will need to store it in variable of DateTime.

```
//ShipmentDate can be stored when shipment is dispatched.
Date ShipmentDate = date.today();
System.debug('ShipmentDate '+ShipmentDate);
```

Long

This is a 64-bit number without a decimal point. This is used when we need a range of values wider than those provided by Integer.

```
Long companyRevenue = 21474838973344648L;
system.debug('companyRevenue'+companyRevenue);
```

String

String is any set of characters within single quotes.

```
String companyName = 'MyTutorialRack';
System.debug('Value companyName variable'+companyName);
```

Time

This variable is used to store the particular time.

27. Question

What are valid use cases for using a custom controller in a Visualforce page? Choose 2 answers

- [ ] A Visualforce page needs to add new actions to a standard controller
- [ ] **A Visualforce page needs to run in system mode**
- [ ] **A Visualforce page should replace the functionality of the standard controller**
- [ ] A Visualforce page should override the save action of the standard controller

Unattempted

When we have standard controller and extensions feature in visualforce what is the need of custom controller?

A custom controller is an Apex class that implements all of the logic for a page without leveraging a standard controller. Use custom controllers when you want your Visualforce page to run entirely in system mode, which does not enforce the permissions and field-level security of the current user.

28. Question

What is true regarding projects in Visual Studio Code? Choose 2 answers

- [ ] **Projects can work in online or offline modes**
- [ ] **Each developer can define the metadata components to be included in a local project**

- [ ] Each developer will have the same set of files in their project
- [ ] Compiling of Apex code is done locally

Unattempted

Correct:

✅ B. Each developer can define the metadata components to be included in a local project. This is a key benefit of using VS Code for Salesforce development. Developers can choose which metadata components (Apex classes, Visualforce pages, etc.) they want to include in their local workspace, allowing them to work on specific parts of the codebase without downloading everything.

✅ A. Projects can work in online or offline modes. While VS Code is optimized for online development (connected to a Salesforce org), you can work in *local development mode* (offline) for certain tasks. This is especially useful for editing code, version control, and other activities that don't require immediate interaction with the org. When you're ready to deploy or test, you can then connect to your org.

Incorrect:

❌ C. Each developer will have the same set of files in their project. This is incorrect. One of the advantages of VS Code projects (and using a version control system like Git) is that developers can have *different* sets of files in their local projects. They can work on different features or modules independently and then merge their changes.

❌ D. Compiling of Apex code is done locally. Apex code is *not* compiled locally. Apex code is compiled on the Salesforce platform when it's saved or deployed to an org. VS Code is primarily a code editor; the actual compilation happens on the Salesforce servers.

29. Question

How would a developer write a query to return the number of leads for each source? Choose 1 answer

- ○ **SELECT LeadSource,COUNT(Name) FROM Lead GROUP BY LeadSource**
- ○ SELECT COUNT(*) FROM Lead GROUP BY LeadSource
- ○ SELECT GROUP(LeadSource) FROM LEad
- ○ SELECT COUNT(LeadSource) FROM Lead

Unattempted

WHAT IS SOQL ?

SOQL stands for Salesforce Object Query Language. It is very similar to the widely used language SQL (Structured Query Language), to query databases. SOQL is specifically designed for Salesforce data and is

used to query the Salesforce platform to retrieve data. SOQL is used within Apex & Visualforce to return sets of data.

Simple SOQL Statement

SELECT Id, Name FROM Account

30. Question

What trigger context variable will return a map of IDs to the old versions of the sObject records? Choose 1 answer

- newMap
- **oldMap**
- updateMap
- insertMap

Unattempted

Here is List of Trigger Context Variables

isExecuting: Returns true if the current context for the Apex code is a trigger, not a Visualforce page, a Web service, or an executeanonymous() API call.

isInsert: Returns true if this trigger was fired due to an insert operation, from the Salesforce user interface, Apex, or the API.

isUpdate: Returns true if this trigger was fired due to an update operation, from the Salesforce user interface, Apex, or the API.

isDelete: Returns true if this trigger was fired due to a delete operation, from the Salesforce user interface, Apex, or the API.

isBefore: Returns true if this trigger was fired before any record was saved.

isAfter: Returns true if this trigger was fired after all records were saved.

isUndelete: Returns true if this trigger was fired after a record is recovered from the Recycle Bin (that is, after an undelete operation from the Salesforce user interface, Apex, or the API.)

new: Returns a list of the new versions of the sObject records. This sObject list is only available in insert, update, and undelete triggers, and the records can only be modified in before triggers.

newMap: A map of IDs to the new versions of the sObject records. This map is only available in before update, after insert, after update, and after undelete triggers.

old : Returns a list of the old versions of the sObject records. This sObject list is only available in update and delete triggers.

oldMap: A map of IDs to the old versions of the sObject records. This map is only available in update and delete triggers.

size: The total number of records in a trigger invocation, both old and new.

31. Question

What does a Helper in a Lightning component contain? Choose 1 answer

- A description, sample code, and one or multiple references to example components
- Custom icon resource for components used in the LightningApp Builder or Community Builder
- Styles for the component
- **Controller functions that can be called from any part of the controller in a component bundle**

Unattempted

What is helper in lightning?

Put functions that you want to reuse in the component's helper. Helper functions also enable specialization of tasks, such as processing data and firing server-side actions. A helper function can be called from any JavaScript code in a component's bundle, such as from a client-side controller or renderer.

32. Question

Which of the following actions can be performed in the Before Update trigger? Choose 2 answers

- **Create a validation before accepting own field changes**
- Modifying trigger.old values
- Delete trigger.new values to avoid changes
- **Changes its own field values using trigger.new**

Unattempted

Correct:

- [x] A. Create a validation before accepting own field changes

`Before Update` triggers are designed for validating data. You can use `record.addError()` within the trigger to prevent the update if the incoming data in `Trigger.new` doesn't meet your requirements. This ensures data integrity.

✅ D. Changes its own field values using trigger.new

This is a primary function of `Before Update` triggers. You can modify the fields in `Trigger.new` to correct, normalize, or enrich the data before it's saved to the database.

Incorrect:

❌ B. Modifying trigger.old values

`Trigger.old` contains the record's values *before* the update. It's read-only. You cannot modify `Trigger.old` in a trigger. Attempting to do so will cause an error.

❌ C. Delete trigger.new values to avoid changes

Deleting records from `Trigger.new` does *not* prevent the update. `Trigger.new` represents the *proposed* changes. To prevent an update, you must use `record.addError()`. The original values would be saved if you just delete from `Trigger.new`.

33. Question

What is the procedure for unit testing ? Choose 1 answer

- Create test data, run all tests, delete any records created, and verify the results
- **Create test data, call the method being tested, verify results**
- Create test data, run all tests, verify results, delete any records created
- Create test data, call the method being tested, verify results, delete any records created

Unattempted

What Are Apex Unit Tests?

To facilitate the development of robust, error-free code, Apex supports the creation and execution of unit tests. Unit tests are class methods that verify whether a particular piece of code is working properly. Unit test methods take no arguments, commit no data to the database, and send no emails. Such methods are flagged with the @isTest annotation in the method definition. Unit test methods must be defined in test classes, that is, classes annotated with @isTest.

For example:

```
@isTest
private class myClass {
@isTest static void myTest() {
// code_block
}
}
```

This example of a test class contains two test methods.

```
@isTest
private class MyTestClass {

// Methods for testing
@isTest static void test1() {
// Implement test code
}

@isTest static void test2() {
// Implement test code
}

}
```

34. Question

Universal Insurance uses Salesforce for managing claims. Claim is a custom object. If an email sent from a claimant regarding an existing claim, it should be checked for a unique reference number and if found, attached to the related claim record, an the status of the claim record updated. What feature would be most appropriate for this requirement? Choose 1 answer

- Email to Custom Object
- Email-to-Case
- **Custom Email Handler**
- Process Builder

Unattempted

- **Email-to-Case** is a standard Salesforce feature that:
  - Automatically converts incoming emails to cases,
  - Can match incoming emails to existing cases using reference numbers,
  - Can update fields or statuses via workflow/process builder on case creation or updates.
- However, this is a **custom object (Claim)**, not the standard Case object.
- Still, **Email-to-Case** can be customized or extended to work with custom objects, but by default it works with Cases.

- **Email to Custom Object**: Salesforce does not have a built-in "Email to Custom Object" feature out of the box.
- **Custom Email Handler**: A custom Apex email service can be created to parse incoming emails, extract reference numbers, relate to claims, and update status. This gives maximum flexibility.
- **Process Builder** alone cannot process incoming emails — it reacts to record changes but doesn't handle email parsing or routing.

35. Question

The developer executes the code below and an error is returned indicating that a variable is not properly declared.

```
String str= new String();
public void stringDisplay(){
str='Hello';
System.debug(str);
}
```

What is the proper way of declaring a String data type variable? Choose 1 answer

- Text str;
- String new=str String();
- String=str;
- **String str;**

Unattempted
Correct:

✅ D. `String str;`

This is the correct way to declare a String variable in Apex (and most Java-like languages). It specifies the data type (`String`) followed by the variable name (`str`).

Incorrect:

❌ A. `Text str;`

`Text` is not a valid Apex data type. While Salesforce has a `Text` area field type, you use `String` in Apex code to represent text.

❌ B. `String new=str String();`

This has several issues. First, `new` is a keyword in Apex (used for object instantiation), so you can't use it as a variable name. Second, the syntax `str String()` is incorrect for initializing a String. You would typically assign a literal value (like `'Hello'`) or another String variable.

✗ C. `String=str;`

This is simply invalid syntax. You must provide a variable name after the data type (`String`). This option is missing the variable name.

36. Question

A developer created a lookup relationship between the standard object Account and a custom object Feedback. Which statement is correct? Choose 1 answer

- **If a feedback record is deleted, the account will not be deleted**
- Any user that can view the account records can also view the feedback records
- If a feedback record is deleted, the account will also be deleted
- The owner of the account record will be the owner its feedback record

Unattempted

Difference between Master Detail and Lookup Relationship:

Master–Detail Relationship :

We cannot create master – detail relationship type fields directly if records already exists. Instead we have to first create Look up fields then fill all the records with that lookup filed. After that we can convert the lookup fields to master – detail relationship.

If we delete master records then detail (Child) records are deleted.

It creates the parent(master) child(Detail) relationship between objects.

Lookup relationship :

Look up relationship creates relations between two objects.

If we delete any object then other object is not deleted.

37. Question

What will be the result if the code below is executed when the variable testRawScore is 75?

```
if(testRawScore >=90){
gradeEqual='A';
} else if(testRawScore >=80){
gradeEqual='B';
} else if(testRawScore >=70){
gradeEqual='C';
```

```
} else if(testRawScore >=60) {
gradeEqual='D';
}
System.debug(gradeEqual);
```

Choose 1 answer

- A
- B
- C
- D

Unattempted

Correct:

✅ C

When `testRawScore` is 75, the code will evaluate the `if` conditions in order.

1. `testRawScore >= 90`: False (75 is not greater than or equal to 90)
2. `testRawScore >= 80`: False (75 is not greater than or equal to 80)
3. `testRawScore >= 70`: True (75 is greater than or equal to 70)

Because the third condition is true, `gradeEqual` is assigned the value 'C'. The code will not evaluate any further conditions.

Incorrect:

✗ A

'A' would only be assigned if `testRawScore` was 90 or greater.

✗ B

'B' would only be assigned if `testRawScore` was between 80 and 89 (inclusive).

✗ D

'D' would only be assigned if `testRawScore` was between 60 and 69 (inclusive).

38. Question

A PageReference is a reference to an instantiation of a page. Which of the following are valid means of instantiating a PageReference? Choose 2 answers

- [ ] **PageReference pageRef=new PageReference('URL');**
- [ ] **Page.existingPageName**
- [ ] ApexPages.Page().existingPageName
- [ ] PageReference.page('URL')

Unattempted

PageReference Class

A PageReference is a reference to an instantiation of a page. Among other attributes, PageReferences consist of a URL and a set of query parameter names and values.

PageReference pageRef = new PageReference('partialURL');

39. Question

What feature can be used to find the latest news and information about the account and contacts you are interested in ? Choose 1 answer

- ○ **Social Accounts, Contacts and Leads**
- ○ Global Search
- ○ News Search
- ○ Twitter Search

Unattempted

Social Accounts and Contacts

The Social Accounts, Contacts, and Leads feature adds social network information from Twitter and YouTube to your records. To use it, you must have an account on each social network that you're using, and you have to link the account or contact record to a user profile on each social network.

40. Question

A developer has created the following trigger to update the description of existing Contract records:

```
List getContracts=new List();
for(Opportunity opp:(List)Trigger.New){
Contract con=[SELECT Id FROM Contract where Id=:opp.ContractId];
con.Description ='This is the contract for Opportunity'+ opp.Name;
getContracts.add(con);
}
update getContracts;
```

How many Contract records will be updated when a developer loads 2000 Opportunity records? Choose 1 answer

- 2000
- 100
- 1
- 0

Unattempted

The trigger will update **0** Contract records, even if 2000 Opportunity records are loaded.

Here's why:

- **Incorrect Filtering:** The trigger iterates through `Trigger.New`, which contains the newly created Opportunities. However, it attempts to find Contracts based on the `ContractId` within this loop.
- **Missing Link:** The Opportunity data likely doesn't contain the `ContractId` directly. The trigger needs a way to link the Opportunity to its associated Contract (e.g., a lookup field on Opportunity referencing the Contract).

Here's a breakdown of the issue:

1. **Looping through New Opportunities:** The trigger loops through each Opportunity record in `Trigger.New`.
2. **Incorrect Lookup:** Inside the loop, the trigger queries for a Contract based on the `opp.ContractId`. However, this field might not exist on the Opportunity object.
3. **No Contract Update:** Without a proper way to identify the related Contract, the `con` variable remains null, and no update occurs.

To fix this, the trigger needs to:

- **Utilize a Lookup Field:** If there's a lookup field on Opportunity that references the related Contract, use that field to identify the Contract within the loop.
- **Direct Update:** Alternatively, if the trigger's purpose is to update existing Contracts based on Opportunity creation, it could directly update the Contract Description field within the Opportunity trigger itself, eliminating the need to query for the Contract.

41. Question

How can the relationship between different accounts be recorded and viewed? Choose 2 answers

- Using the View Hierarchy link
- Using the Generate Relationship

- [ ] Using the Parent Account field
- [ ] Using the Related Account field

Unattempted

Correct:

✅ A. Using the View Hierarchy link

The “View Hierarchy” link (available on Account detail pages) is specifically designed to visualize the hierarchical relationships between accounts. It provides a visual representation of parent-child account structures.

✅ C. Using the Parent Account field

The “Parent Account” field is the standard way to establish hierarchical relationships between accounts in Salesforce. Populating this field creates the parent-child link that is then visualized by the “View Hierarchy” link.

Incorrect:

❌ B. Using the Generate Relationship

There’s no standard “Generate Relationship” feature in Salesforce for accounts. Relationships are created by populating the “Parent Account” field. Custom relationships can be created for other objects, but this question is specifically about *accounts*.

❌ D. Using the Related Account field

There is no standard “Related Account” field on the Account object. While you can create custom fields to relate accounts, the *standard* way to represent hierarchical relationships (parent-child) is via the “Parent Account” field, not a custom “Related Account” field. Also, simply having a related account field wouldn’t inherently provide the hierarchical view offered by the “View Hierarchy” functionality.

42. Question

Which action causes a before trigger to fire by default for Accounts?

- [ ] Renaming or replacing picklists.
- [ ] Converting Leads to Contact accounts.
- [ ] Importing data using the Data Loader and the Bulk API.
- [ ] Updating addresses using the Mass Address update tool.

Unattempted

This question is related to the concept of "Operations That Don't Invoke Triggers".

Some operations don't invoke triggers. riggers are invoked for data manipulation language (DML) operations that the Java application server initiates or processes. Therefore, some system bulk operations don't invoke triggers.

The correct answer is "Importing data using the Data Loader and the Bulk API" because it causes before triggers to fire by default.

The below answers are incorrect:

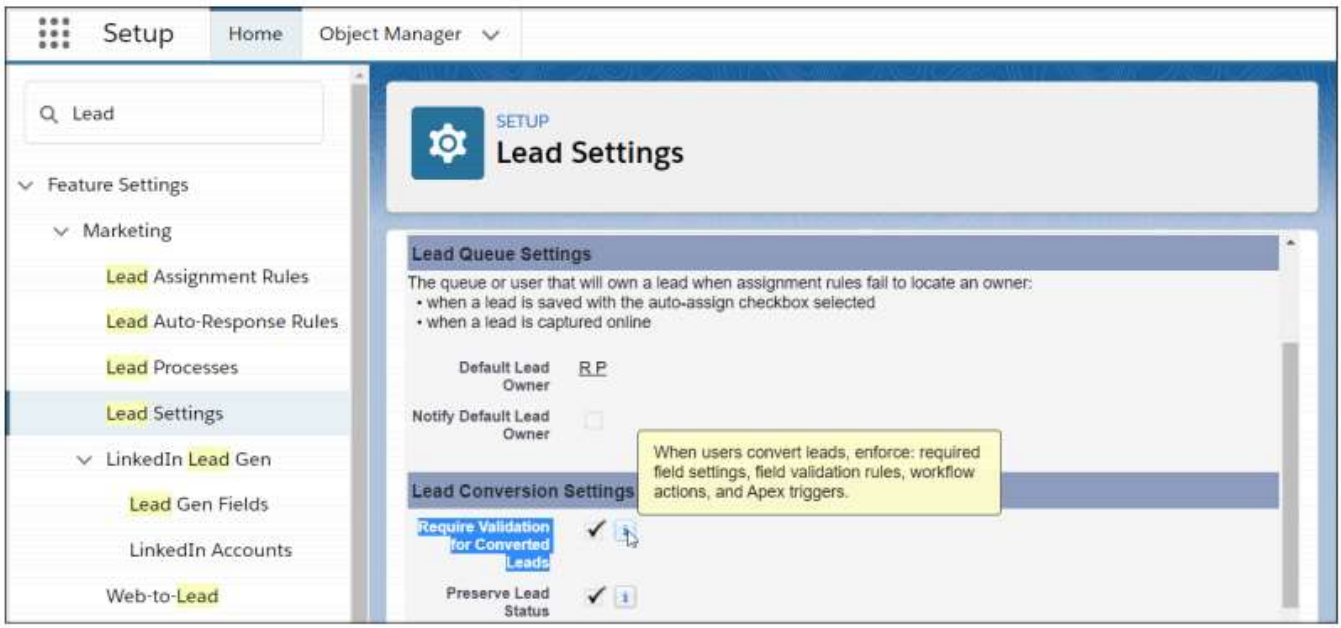
"Renaming or replacing picklists" is NOT correct because renaming or replacing picklists will not invoke triggers.

"Updating addresses using the Mass Address update tool" is NOT correct because Mass address updates don't invoke triggers.

"Converting Leads to Contact accounts" is NOT correct because the before triggers associated with these operations are fired during lead conversion only if validation and triggers for lead conversion are enabled in the organization.

There is a checkbox "Require Validation for Converted Leads" which decides that Apex Trigger will fire or not.

Please find the below screenshot:



43. Question

What are three capabilities of the <1tng:require> tag when loading JavaScript resources in Aura components? Choose 3 answers.

- [ ] **One-time loading for duplicate scripts.**
- [ ] Loading files from Documents.
- [ ] **Specifying loading order.**

- [ ] Loading scripts in parallel.
- [ ] Loading externally hosted scripts.

Unattempted

This question is related to using external JavaScript Libraries in Lightning Components.

To reference a JavaScript library, upload it as a static resource and use a tag in your .cmp or .app markup. The framework's content security policy mandates that external JavaScript libraries must be uploaded to Salesforce static resources.

Syntax of using external JavaScript in Aura Component:

ComponentWithExternalJS.cmp *

```
1  <!-- ComponentWithExternalJS
2     - It shows basic syntax for using external JS in Aura Component
3     - Here, resourceName is the name of Static Resource
4  -->
5  <aura:component >
6
7     <ltng:require scripts="{!$Resource.resourceName}"
8     afterScriptsLoaded="{!c.afterScriptsLoaded}" />
9
10 </aura:component>
```

Important Points to NOTE:

Loading Order

The scripts are loaded in the order that they are listed.

One-Time Loading

Scripts load only once, even if they're specified in multiple `<ltng:require>` tags in the same component or across different components.

Parallel Loading

Use separate `<ltng:require>` tags for parallel loading if you have multiple sets of scripts that are not dependent on each other.

Encapsulation

To ensure encapsulation and reusability, add the `<ltng:require>` tag to every `.cmp` or `.app` resource that uses the JavaScript library.

`ltng:require` also has a `styles` attribute to load a list of CSS resources. You can set the `scripts` and `styles` attributes in one `<ltng:require>` tag.

With this explanation and important points,

The first correct answer is:

"One-time loading for duplicate scripts" because Script only loads once, even if they are specified in multiple <1tng:require> tags.

The second correct answer is:

"Specifying loading order" because the scripts are loaded in the order they are listed.

The third correct answer is:

"Loading scripts in parallel" because we can use separate <1tng:require> tags to parallelly load multiple sets

of scripts.

Reference:

https://developer.salesforce.com/docs/atlas.en-us.lightning.meta/lightning/js_libs_platform.htm

44. Question

Which three resources in an Aura Component can contain JavaScript functions?

Choose 3 answers.

- [ ] Design
- [ ] Style
- [ ] Renderer
- [ ] Controller
- [ ] Helper

Unattempted

An Aura component is a combination of markup, JavaScript, and CSS. You first create a component bundle. Components are the functional units of Aura, which encapsulate modular and reusable sections of UI. They can contain other components or HTML markup. The public parts of a component are its attributes and events.

Coming on to the answer to our question please visit the below reference links.

Resource	Resource Name	Usage	See Also
Component or Application	`sample.cmp` or `sample.app`	The only required resource in a bundle. Contains markup for the component or app. Each bundle contains only one component or app resource.	Creating Components aura:application
CSS Styles	`sample.css`	Contains styles for the component.	CSS in Components
Controller	`sampleController.js`	Contains client-side controller methods to handle events in the component.	Handling Events with Client-Side Controllers
Design	`sample.design`	File required for components used in Lightning App Builder, Lightning pages, Experience Builder, or Flow Builder.	Aura Component Bundle Design Resources
Documentation	`sample.auradoc`	A description, sample code, and one or multiple references to example components	Writing Documentation for the Component Library
Renderer	`sampleRenderer.js`	Client-side renderer to override default rendering for a component.	Create a Custom Renderer
Helper	`sampleHelper.js`	JavaScript functions that can be called from any JavaScript code in a component's bundle	Sharing JavaScript Code in a Component Bundle
SVG File	sample.svg	Custom icon resource for components used in the Lightning App Builder or Experience Builder.	Configure Components for Lightning Pages and

A Lightning Component bundle developed with the Aura programming model can have three JavaScript files: controller.js, helper.js, and renderer.js.

The helper file is specifically designed for sharing code within a single component bundle for use by the controller and renderer files. To access the shared code of the helper.js file, reference the helper argument passed to the functions in your controller.js and renderer.js files

Here showing class named redditCloneHome with its bundle components

```
/* helper.js */
({
  callServer: function(component) {
    var action = component.get("c.someApexMethod");
    $A.enqueueAction(action);
  }
})
```

```
/* renderer.js */
({
  afterRender: function(component, helper) {
    this.superAfterRender();
    helper.callServer();
  }
})
```

In view of the above explanation,

“Controller“, “Helper“, and “Renderer“ are the correct answers.

Reference:

https://developer.salesforce.com/docs/atlas.en-us.lightning.meta/lightning/components_bundle.htm

The rest of the answers “Design“ and “Style“ are incorrect.

45. Question

A developer needs to have records with specific field values in order to test a new Apex class.

What should the developer do to ensure the data is available for the test?

- Use Test.loadData() and reference a CSV file.
- Use SOQL to query the org for the required data.
- Use Anonymous Apex to create the required data.
- **Use Test.loadData() and reference a static resource.**

Unattempted

This question is related to creating Apex Test Data in Salesforce. Apex test data is transient and isn’t committed to the database. This means that after a test method finishes execution, the data inserted by the test doesn’t persist in the database.

As a result, there is no need to delete any test data at the conclusion of a test. Likewise, all the changes to existing records, such as updates or deletions, don’t persist.

By default, existing organization data isn’t visible to test methods, with the exception of certain setup objects. You should create test data for your test methods whenever possible. There are different methods

to create test data.

The correct answer is "Use Test.loadData() and reference a static resource" because Using the Test.loadData method, you can populate data in your test methods without having to write many lines of code. Below steps need to be followed:

- Add the data in a .csv file.
- Create a static resource for this file.
- Call Test.loadData within your test method and passing it the sObject type token and the static resource name.

The rest of the answers are incorrect:

"Use Test.loadData() and reference a CSV file" is NOT correct because, from the above explanation, we need to reference Static Resource while using "Test.loadData".

"Use SOQL to query the org for the required data" because by annotating your class with @isTest (SeeAllData=true), you allow test methods to access all org records but this is not the best practice.

"Use Anonymous Apex to create the required data" is NOT correct because we cannot create test data using Anonymous Apex

References:

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_testing_load_data.htm

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_testing_seealldata_using.htm

46. Question

Which three statements are true regarding custom exceptions in Apex?

Choose 3 answers.

- [ ] **A custom exception class name must end with 'Exception'.**
- [ ] **A custom exception class can implement one or many interfaces.**
- [ ] A custom exception class can extend other classes besides the Exception class.
- [ ] A custom exception class cannot contain member variables or methods.
- [ ] **A custom exception class must extend the system Exception class.**

Unattempted

Let"s understand few points about Custom Exception:

- Exceptions can be top-level classes, that is, they can have member variables, methods and constructors, they can implement interfaces, and so on.
- To create your custom exception class, extend the built-in Exception class and make sure your class name ends with the word Exception, such as "MyException" or "PurchaseException". All exception classes extend the system-defined base class Exception, and therefore, inherits all common Exception methods.
- A class extends another class using the extends keyword in the class definition. A class can only extend one other class, but it can implement more than one interface.

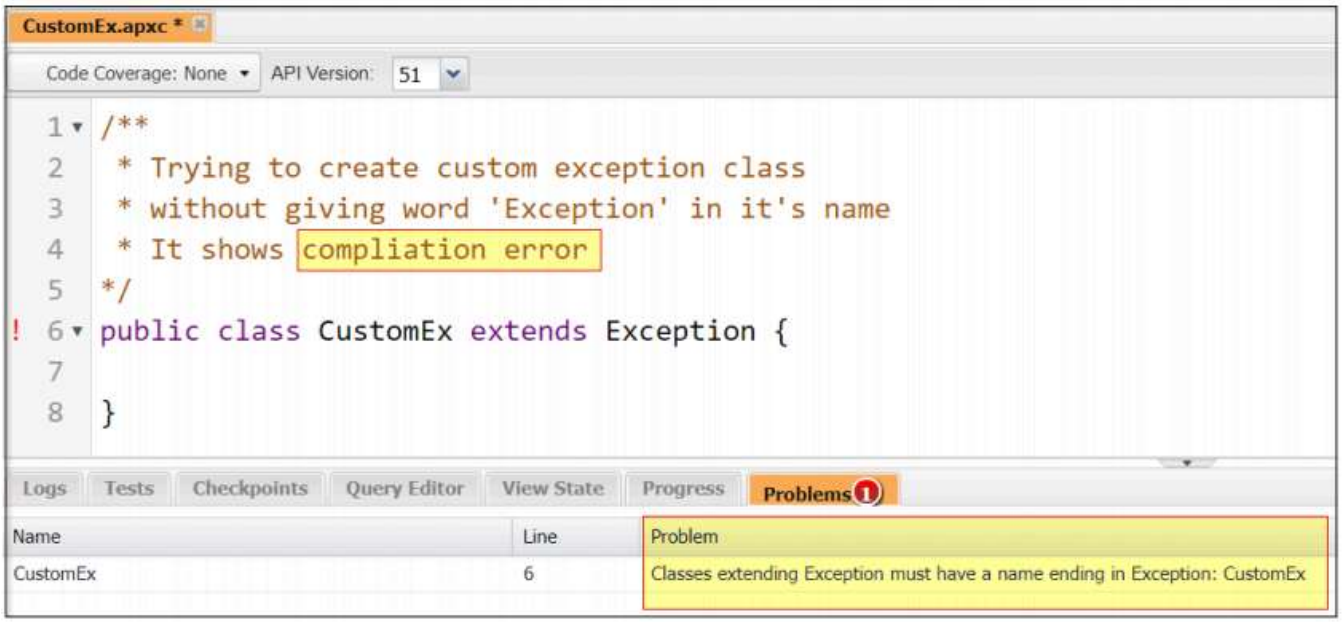
With this explanation, the correct answers are:

A custom exception class must extend the system Exception class is correct.

and

A custom exception class name must end with Exception is correct because Classes extending Exception must have a name ending in Exception.

Please find the below screenshot.



The third correct answer is A custom exception class can implement one or many interfaces because it can implement one or more interfaces.

References:

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_exception_custom.htm

https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_classes_extending.htm

The rest of the answers are not correct.

47. Question

Universal Containers has a support process that allows users to request support from its engineering team using a custom object, Engineering_Support_c.

Users should be able to associate multiple Engineering_Support_c records to a single Opportunity record.

Additionally, aggregate information about the Engineering_Support_c records should be shown on the Opportunity record.

What should a developer implement to support these requirements?

- Master-detail field from Opportunity to Engineering_Support_c
- Lookup field from Engineering_Support_c to Opportunity
- **Master-detail field from Engineering_Support_c to Opportunity**
- Lookup field from Opportunity to Engineering_Support_c

Unattempted

This question is more of an Administrator's knowledge than a Platform Developer 1 knowledge.
Let us understand it thoroughly.
We have Engineering_Support cases for the opportunity. Let's say we are offering some service/product to end-user and before that opportunity gets materialized/converted to Closed-Won. We need to address some support queries of the customer.
It means One opportunity record will have multiple Support records.
So, Opportunity is parent, and Engineering support is a child object.
With this, we either have to create either a Lookup or Master-Detail (MD) relationship field on the Engineering Support object.
So that Engineering support will have a lookup field and Opportunity will have a related list.
Now the second requirement mentions that we need to show aggregate information about the Engineering support records.
We can achieve it through the Roll-up summary field.
And in order to apply the Roll-up summary field, we need to compulsory create an MD relationship on the Engineering support object.
Hence, the "Master-detail field from Engineering_Support_c to Opportunity" is the correct answer selection.
You can gather more information about the MD relationship and Roll-up summary on the below reference links.
As these are Administrator concepts I am not elaborating them here in further detail.
Reference:
https://help.salesforce.com/s/articleView?id=sf.fields_defining_summary_fields.htm&type=5
Trailhead Module:
https://trailhead.salesforce.com/content/learn/modules/point_click_business_logic/roll_up_summary_fields
https://trailhead.salesforce.com/content/learn/modules/data_modeling/object_relationships

48. Question

A PrimeryId_c custom field exists on the Candidate_c custom object. The field is used to store each candidate's Id number and is marked as Unique in the schema definition.
As part of a data enrichment process, Universal Containers has a CSV file that contains updated data for all candidates in the system. The file contains each candidate's primary Id as a data point. Universal Containers wants to upload this information into Salesforce, while ensuring all data rows are correctly mapped to a candidate in the system.
Which technique should a developer implement to streamline the data upload?

- Create a before insert trigger to correctly map the records.
- Create a Process Builder on the Candidate_c object to map the records.
- **Update the PrimaryId__c field definition to mark it as an External Id.**
- Upload the CSV into a custom object related to Candidate_c.

Unattempted
External ID in Salesforce is a custom field that has the 'External ID' attribute checked meaning that it contains unique record identifiers from a system outside of Salesforce. When we select this option the

import wizard will detect existing records in Salesforce that have the same External Identification.
The fields with the below data types can only be external Id:
 Number
 Text
 Email
The correct answer is Update the PrimaryId__c field definition to mark it as an External Id because when importing custom objects, solutions, or person accounts, you can use external IDs to prevent the import from creating duplicate records and to map the records correctly.
Reference:
https://help.salesforce.com/articleView?id=sf.faq_import_general_what_is_an_external.htm&type=5
The rest of the answers are not convincing:
Upload the CSV into a custom object related to Candidate__c is NOT correct because this will only work when the field is marked as External ID.
Reference:
https://help.salesforce.com/articleView?id=000320964&type=1&mode=1
Create a Process Builder on the Candidate_c object to map the records is NOT correct because we don't need any custom solution for this requirement.
Create a before insert trigger to correctly map the records is NOT correct because we don't need any custom solution for this requirement.

49. Question

Which two settings must be defined in order to update a record of a junction object?
Choose 2 answers.

- [ ] Read/Write access on the junction object.
- [ ] Read access on the primary relationship.
- [ ] **Read/Write access on the secondary relationship.**
- [ ] **Read/Write access on the primary relationship.**

Unattempted
In order to understand the answer let us first gather the understanding of Junction Object.
Junction objects help associate two objects. They are custom objects in Salesforce that allow building a master-detail relationship established between two different data objects.
You can use master-detail relationships to model many-to-many relationships between any two objects. A many-to-many relationship allows each record of one object to be linked to multiple records from another object and vice versa.
Junction objects establish many-to-many relationships between two objects and are created with two master-detail relationships.
Now to understand the answer selection please follow the below steps.
1) Create a user and assign it a custom profile.
2) Create three custom objects.
i) PrimaryRelationshipObj

ii) SecondaryRelationshipObj

iii) JunctionObj

3) Create first MD relationship on JunctionObj with PrimaryRelationshipObj (That will become primary relationship)

4) Create a second MD relationship on JunctionObj with SecondaryRelationshipObj (That will become primary relationship)

5) Make sure to create a tab for all three objects and make sure to assign Read, Create, Edit, and Delete.

6) From Sharing settings first mark PrimaryRelationshipObj and SecondaryRelationshipObj with Private "Default Internal access". (You will not observe JunctionObj there because its access is controlled by Primary and Secondary objects. (And hence, "Read/Write access on the junction object." is totally incorrect option.

7) Now through Admin, a user creates 2-3 records under junction object.

8) Login with the user created in the first step.

Observation: The user will not be able to even view any records under the JunctionObj tab.

9) Now change the sharing setting options with the other three options.

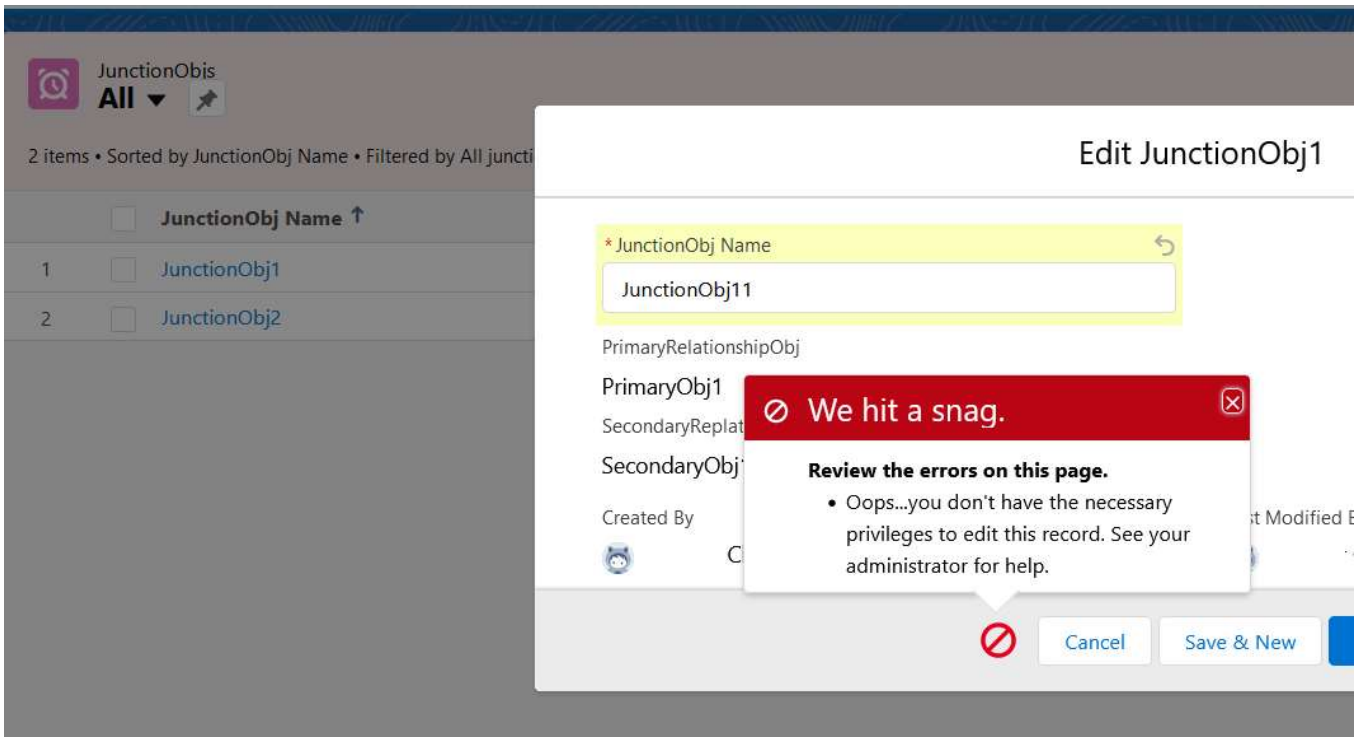
Observation: The user will be able to edit the records under JunctionObj only when "Read/Write access on the primary relationship." and "Read/Write access on the secondary relationship."

PrimaryRelationshipObj	Public Read/Write
SecondaryReplationshipObj	Public Read/Write

The rest of the answer options are incorrect:

"Read/Write access on the junction object" is incorrect as already explained above.

If you keep "Read access on the primary relationship" and try to modify the records of JunctionObj then it will show the below error.



50. Question

A developer created a weather app that contains multiple Lightning web components.
One of the components, called Toggle, has a toggle for Fahrenheit or Celsius units. Another component, called Temperature, displays the current temperature in the unit selected in the Toggle component.
When a user toggles from Fahrenheit to Celsius or vice versa in the Toggle component, the information must be sent to the Temperature so the temperature can be converted and displayed.
What is the recommended way to accomplish this?

- Use an application event to communicate between the components.
- **The Toggle component should call a method in the Temperature component.**
- Use Lightning Message Service to communicate between the components.
- Create a custom event to handler the communication between components.

Unattempted

This question is a scenario-based question and it is related to Lightning Web Component.
Let's understand the scenario step by step:
1. Component 1 named "Toggle" is used to toggle between Fahrenheit or Celsius and vice versa.
2. Component 2 named "Temperature" is used to show the current temperature in the selected unit.
3. The important point is When a user toggles from Fahrenheit to Celsius or vice versa in the Toggle component, the information must be sent to the Temperature so the temperature can be converted and displayed.
4. As per the scenario, the Lightning Components can be placed as below:

- Toggle component can act as the parent component
- Temperature component can act as a child component
- When the toggle button is used, the selected unit information should be passed to the Temperature (child) component.

So, here the point to focus on is: When the toggle button is used, the selected unit information should be passed from Toggle (Parent) component to the Temperature (child) component.
Please Note: To communicate down the containment hierarchy, owner and parent components can call JavaScript methods on child components. To expose a public method, decorate it with @api.
That means in our scenario, we can create a method in the child component which can be called from the parent component.
With this explanation, the correct answer is
"The Toggle component should call a method in the Temperature component."
Let's see below code example:
Screenshot – Temperature (Child Component) – html file

```
<!-- Temperature Component
     (Child Component)
-->

<template>
    <h1>This is the Temperature Component</h1>
    <p>The Current Temerature is: {currentTemp} {defaultUnit}</p>
</template>
```

Screenshot Temperature (Child Component) – javascript file

```
import { LightningElement, api } from 'lwc';

export default class Temperature extends LightningElement {
defaultUnit = 'Celsius';
currentTemp = 25;
    @api
    handleValueChange(inputUnit) {
        this.defaultUnit = inputUnit;

        if(this.defaultUnit == 'Fahrenheit'){
            this.currentTemp = this.currentTemp * 9/5 + 32;
        }
        else{
            this.currentTemp = 25;
        }
    }

}
```

This method will be called from parent component

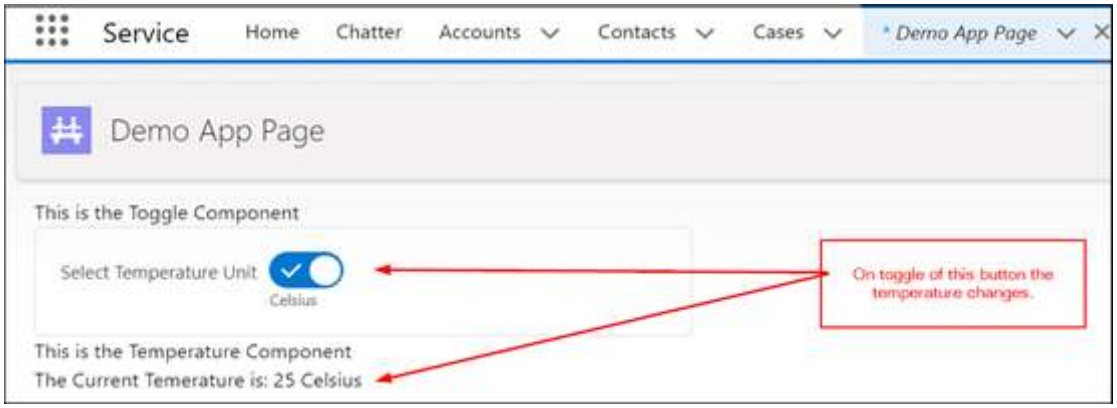
Screenshot Toggle (Parent Component) – html file

```
<!-- Toggle Component
     (Parent Component)
-->
<template>
    <h1>This is the Toggle Component</h1>

    <div class="slds-box">
        <lightning-input type="toggle"
        label="Select Temperature Unit"
        name="input1" checked={checked}
        onchange={toggleTemperature}
        message-toggle-active="Celsius"
        message-toggle-inactive="Fahrenheit" ></lightning-input>
    </div>
    <c-temperature></c-temperature>
</template>
```

Screenshot Toggle (Parent Component) – javascript file

```
import { LightningElement } from 'lwc';

export default class Toggle extends LightningElement {

    checked = true;
    toggleTemperature(event){

        this.checked = !this.checked;
        console.log(this.checked);

        if(this.checked){
            this.template.querySelector("c-temperature").handleValueChange('Celsius');
        }
        else{
            this.template.querySelector("c-temperature").handleValueChange('Fahrenheit');
        }
    }
}
```

Screenshot – UI Showing Toggle and Temperature



References:

https://developer.salesforce.com/docs/component-library/documentation/en/lwc/create_javascript_methods

https://salesforcediaries.com/2019/08/21/lwc-communication-part-3-call-methods-on-child-components/

Use Page numbers below to navigate to other practice tests

Pages: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

We help you to succeed in your certification exams

We have helped over thousands of working professionals to achieve their certification goals with our practice tests.

Skillcertpro

Quick Links

ABOUT US
FAQ
BROWSE ALL PRACTICE TESTS
CONTACT FORM

Important Links

REFUND POLICY
REFUND REQUEST
TERMS & CONDITIONS
PRIVACY POLICY

Privacy Policy