PassQueen
a.k.a dumpsqueen.com

Pega Certified System Architect (PCSA) 87V1

Pegasystems PEGAPCSA87V1

Version Demo

Total Demo Questions: 28

Total Premium Questions: 287

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Application Design	41
Topic 2, Case Management	73
Topic 3, Data Modeling	58
Topic 4, User Experience	55
Topic 5, Process Automation	28
Topic 6, Security	6
Topic 7, Reporting	18
Topic 8, Data Integration	7
Topic 9, Mix Questions	1
Total	287

QUESTION NO: 1

A reservation process allows customers to reserve a flight, hotel room, and rental car as part of a travel itinerary. Which configuration displays a Select flight insurance check box only when the itinerary includes a flight?

A. A disable condition (when rule) that is applied to the section with the flight insurance information.

B. A required condition (when rule) that is applied to the Select flight insurance check box.

C. A visibility condition (when rule) that is applied to the Select flight insurance check box.

D. A disable condition (when rule) that is applied to the Select flight insurance check box.

ANSWER: C

Explanation:

A visibility condition (when rule) that is applied to the Select flight insurance check box is correct because a visibility condition controls whether a UI control is rendered and shown to the user. The when rule evaluates the current itinerary data—for example, whether a flight reservation exists—and Pega displays the checkbox only when that evaluation is true. This keeps the form relevant: customers whose itinerary contains a flight can choose insurance, while customers booking only a hotel or rental car do not see a flight-specific control. In Pega sections and views, visibility is configured as a condition based on a when rule or other expression that evaluates case data. The condition is evaluated at run time as the UI is displayed or refreshed, enabling the interface to adapt to the selections and data in the travel itinerary. This is distinct from configurations that affect whether a control can be edited or whether its value must be supplied; the stated requirement is specifically to display the control only for applicable itineraries. For more information, see Pega documentation on configuring visibility conditions and creating when rules.

QUESTION NO: 2

A process routes loan requests to a specific loan officer based on the type of loan. - If the loan is a mortgage, it is routed to Adam Ross. If the loan is for an automobile, it is routed to Julia Samuels. - If the loan is an equity line, the case is routed to Don Smith. How do you configure a router to ensure that case advances to the correct loan officer?

A. Route the case to a worklist using a skilled router.

B. Route the case to a work queue using a When condition.

C. Route the case to a work queue using a skilled router.

D. Route the case to a worklist using a When condition.

ANSWER: D

Explanation:

Route the case to a worklist using a When condition. A worklist is the appropriate destination because Adam Ross, Julia Samuels, and Don Smith are individually identified operators. In Pega, an operator’s worklist holds assignments addressed directly to that operator, allowing the named loan officer to receive and process the relevant case assignment.

The routing logic must evaluate the loan type and select the corresponding destination: the mortgage condition identifies Adam Ross, the automobile-loan condition identifies Julia Samuels, and the equity-line condition identifies Don Smith. Configuring conditional logic with When rules makes the routing decision explicit, maintainable, and based on case data. Each condition can evaluate the property that stores the requested loan type, and the matching route sends the assignment to the selected operator’s worklist. This supports deterministic routing whenever the business requirement specifies a particular person rather than a general pool of qualified users. Pega’s routing configuration supports directing assignments to worklists and using business logic to determine the appropriate destination. For further details, see Pega Academy: Routing work and Pega Platform documentation: Routing assignments.

QUESTION NO: 3

A business architect has developed a new process for a case type. To verify that the UI elements collect the expected results, you want to test the process and the fields. Which two configurations, when used together, allow you to record a set of interactions and save the test results to verify process functionality? (Choose Two)

A. Add explicit assertions on the UI elements

B. Add validations on the UI elements

C. Create a unit test for the case type

D. Create a scenario test for the case type

ANSWER: A D

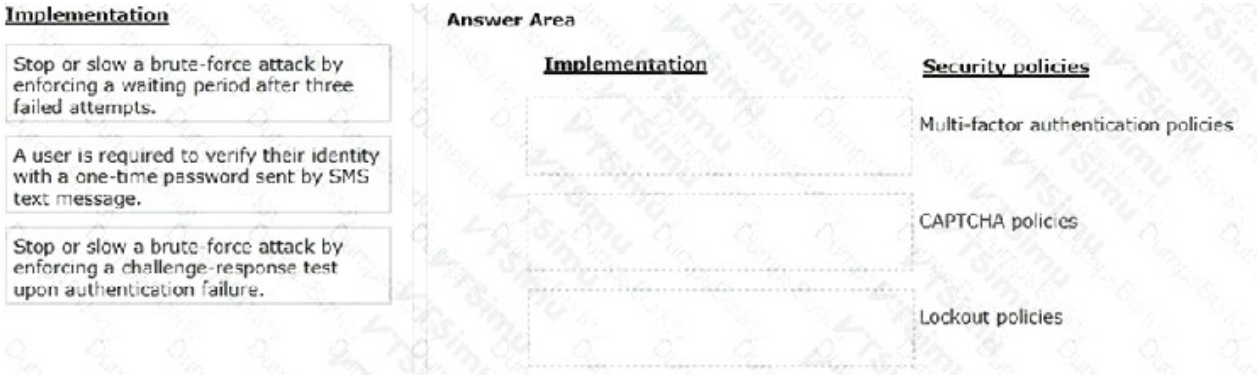
Explanation:

Create a scenario test for the case type is correct because scenario tests are Pega's no-code, end-to-end functional testing capability. A test author records the interactions a user performs while progressing through a case, such as entering values, selecting controls, submitting assignments, and navigating the case workflow. Pega saves those recorded interactions as a reusable test case, which can then be run to verify that the case process continues to behave as expected after rule changes.

Add explicit assertions on the UI elements is the complementary configuration because a recorded interaction alone confirms the path taken, whereas assertions verify the expected outcome at specific points in that path. An assertion can check that a UI element is present or that it displays the expected value. This makes the scenario test suitable for confirming both that the process executes and that the fields collect, display, and retain the intended data. When the scenario test runs, Pega records the execution results, including whether its assertions pass, providing evidence of process functionality and helping identify regressions. See Pega's Scenario testing documentation and Scenario tests learning topic.

QUESTION NO: 4 - (SIMULATION)

Select each security implementation on the left and drag it to the corresponding security policies.



ANSWER: See the explanation for the answer

Explanation:

Correct mapping:

MFA adds an additional identity-verification factor, such as an SMS one-time password. CAPTCHA presents a challenge-response test, while lockout policies restrict or delay further login attempts after repeated failures.

QUESTION NO: 5

How do you guide users through an application form without requiring user training?

A. Add the corresponding step to an appropriate stage.

B. Send a notification to the assigned user.

C. Add an instruction to the assignment.

D. Add an optional action to the case to explain the task.

ANSWER: C

Explanation:

Add an instruction to the assignment. is correct because assignment instructions provide contextual, just-in-time guidance directly where a user performs work. In a case type, an assignment represents a task routed to a user or work queue. Configuring an instruction on that assignment lets the application display concise information about what the user needs to do, the business purpose of the task, and any relevant details or expectations. This embedded guidance helps users complete an application form correctly without needing separate classroom training, external documentation, or prior familiarity with the workflow.

Instructions are particularly useful when a task has rules that may not be obvious from the form fields alone, such as the conditions for collecting supporting information or the expected outcome before the user submits the assignment. Because the guidance is delivered at the point of work, it supports a consistent user experience and can reduce errors and rework. Pega's case-management capabilities are designed to present work in context, including assignment-level information that helps users process cases effectively. For additional platform and learning resources, see Pega Documentation and Pega Academy.

QUESTION NO: 6

Which two statement are true about insights? (Choose two.)

A. You can search for an select the fields that you want to include in an insight.

B. You can transform data queries into sharable visualizations.

C. You can transform sharable visualizations into data queries.

D. You can edit application data directly in an insight.

ANSWER: A B

Explanation:

Insights provide a self-service reporting experience for exploring application data. When creating an insight, users choose the data source and then locate the properties that are meaningful for the analysis. The field-selection experience supports finding and selecting the fields to display, group, filter, or otherwise use in the insight, so "You can search for an select the fields that you want to include in an insight." describes a core capability. Insights also let users turn the resulting data query into a presentation that is useful to other users, such as a table or chart, and save or share that visualization. Therefore, "You can transform data queries into sharable visualizations." is also correct. This capability helps business users move from raw operational data to an understandable, reusable view without building a separate report definition in Dev Studio. The insight remains based on the selected data and query configuration, while its visualization makes the results easier to interpret and distribute. For Pega learning material on reporting and Insights, consult Pega Academy and the official Pega Documentation portal.

QUESTION NO: 7

Which two scenarios require you to configure conditional processing within the case type? (Choose Two.)

A. A scholarship eligibility application requires students to enter standardized test scores, Students with qualifying test scores can schedule and interview. Students without qualifying test scores receives a rejection email.

B. A catering booking application requires customers to enter information about the expected party size, event date, and event time. When customers submit the information the catering company sends a confirmation email.

C. An application requires customer to select the type of request in a drop-down list. The system routes the request to the appropriate department work queue. A user with access to the work queue processes the case through fulfillment.

D. A shopping application requires a guest to fill out payment information. A user who enters a membership number skips the payment information step.

ANSWER: A D

Explanation:

Conditional processing is appropriate when the case flow must evaluate information and then determine whether work should occur, be skipped, or follow a different path. In the scholarship eligibility scenario, the student's standardized test score determines the next case outcome. Qualifying scores allow the case to proceed to interview scheduling, while nonqualifying scores cause the case to send a rejection email. This requires a condition, typically implemented with a decision shape or conditional logic that evaluates the score and directs the case to the applicable outcome.

In the shopping scenario, entry of a membership number changes whether the payment-information step is needed. The payment collection assignment should therefore be configured to run only when the applicable condition is met; when a membership number is entered, the step is skipped. This is a standard use of conditional processing in a Pega case life cycle: a process or assignment is made available based on a condition evaluated from case data. Pega's case processing features support decision-based paths and conditional execution so that users see and complete only the work relevant to their situation. See Pega documentation on creating case types and Pega Academy: Conditional processing.

QUESTION NO: 8

Customers can log their own product support requests using an online portal. Once logged in, the portal displays the list of products purchased by the customer. The customer can initiate one or more support requests for each product. What is the appropriate scope for a data page that sources the list of products purchased by the customer?

A. Thread

B. System

C. Requestor

D. Node

ANSWER: C

Explanation:

Requestor is the appropriate scope because the list of purchased products is specific to the customer currently using the portal. A requestor-scoped data page is cached separately for each requestor session, so the signed-in customer can reuse their product list while creating and viewing multiple support requests without repeatedly sourcing the same data. This supports efficient processing across the customer's portal interaction while maintaining isolation between different users' data.

In Pega, a requestor represents a user session or connection to the application. Requestor scope is therefore well suited to data that should remain available for the duration of that user's session but must not be shared with other customers. The product list can be loaded once for the logged-in customer and referenced by each support-request case they initiate. If the application uses customer context as an input parameter, that context should also be supplied consistently when the data page is referenced. Pega's data page guidance explains that scope determines the lifetime and sharing of a data page's cached contents; selecting Requestor scope provides session-level reuse for user-specific information. See Pega Academy: Data pages and Pega Documentation: Data pages.

QUESTION NO: 9

What two pieces of information comprise a data element? (Choose Two)

A. The name of the referencing user view

B. The name of the data element

C. The name of the clipboard page

D. The value of the data element

ANSWER: B D

Explanation:

In Pega's data model, a data element is the basic unit used to hold a specific piece of case, user, or application information. It consists of **the name of the data element** and **the value of the data element**. The name identifies what the information represents and is used when rules reference the property, such as a customer name, requested amount, or status. The value is the actual information currently stored for that named element in a particular context, for example, "Jordan Lee," "5000," or "Open."

This name-and-value structure lets Pega rules consistently read, set, validate, and display business data. During case processing, values are typically held on the clipboard in properties; however, the clipboard page supplies context and organization for those properties rather than forming part of the definition of an individual data element. Pega's data modeling guidance describes properties as the mechanism for defining and storing application data, with a property name identifying the data and its value representing the stored content. See Pega Platform data modeling documentation and Pega Academy: Data model.

QUESTION NO: 10

In a claims application customers can file home insurance claims. Each claims contains a list of items of loss. Depending on the situation, some claims… investigated for potential fraud in parallel to the actual claim process.

Which two case types do you create to support this scenario? (Choose two.)

A. Items of loss

B. Customer

C. Claim

D. Fraud Investigation

ANSWER: C D

Explanation:

Claim is a case type because it represents the primary business transaction that the application must process from submission through resolution. A claim has its own lifecycle, including the collection and assessment of loss information, evaluation of coverage, decisioning, and settlement. The list of items of loss is information managed in the context of that claim, rather than a separate case lifecycle in this scenario.

Fraud Investigation is also a case type because a potential-fraud review is a distinct unit of work with its own process, assignments, participants, and outcome. Creating it as a case enables the investigation to run independently and in parallel with the claim process when the relevant business conditions apply. The fraud case can be associated with the claim while allowing specialist work to be tracked, routed, and resolved separately. Pega case types are intended to model these meaningful business processes and their lifecycles; related work can be implemented as separate cases when it needs independent processing. See Pega Academy's Case management topic and the Pega documentation on case types.

QUESTION NO: 11

Select each Application Design Requirement on the left and drag it to the appropriate Design Approach on the right

Application design requirement

1. Display only relevant fields for a given task

2. Help the user perform an expected task

3. Record the justification for an action taken on case

4. Alert the user of a pending assignment

A. Assignment Instruction

B. Intent Driver UI

C. Assignment Notification

D. Audit Note
• 1-b,2-a,3-d,4-c
• 1-a,2-b,3-d,4-c
• 1-b,2-c,3-d,4-a
• 1-c,2-d,3-b,4-a

E. 1-Intent-Driven UI, 2-Assignment Instruction, 3-Audit Note, 4-Assignment Notification

ANSWER: E

Explanation:

The correct mapping is “1-Intent-Driven UI, 2-Assignment Instruction, 3-Audit Note, 4-Assignment Notification.” An intent-driven UI tailors the fields and controls presented to a user according to the context of the work, helping ensure that a person sees only the information relevant to the task being performed. Assignment instructions provide task-specific guidance directly to the user who receives an assignment, supporting the expected action and helping the user complete it correctly. An audit note captures the business justification or contextual comments associated with an action on a case, preserving that information in the case history for later review. Assignment notifications alert users about work that requires their attention, including pending assignments. These features apply Pega’s case-management and user-experience design approach: present contextual work, guide users where they perform that work, preserve an accountable record of decisions, and notify users of actionable assignments. Pega training materials cover these case and assignment capabilities in the Pega Academy, while implementation documentation is available through the Pega Documentation portal.

QUESTION NO: 12

A flow action calls a pre-processing data transform to initiate values. There are several flow actions available for the assignment. You want to make sure that the values are only initiated once for each flow action. How do you implement a solution?

A. Do nothing. The pre-processing data transform is only called once for each assignment.

B. Make sure that the flow action does not have the highest likelihood since it will always be invoked.

C. Configure the data transform as post-processing instead of pre-processing.

D. Add logic to the pre-processing data transform to test if values were already initiated.

ANSWER: D

Explanation:

Add logic to the pre-processing data transform to test if values were already initiated. A flow action’s pre-processing data transform runs when the user opens that action, so it can be invoked again if the user revisits the action, switches between available actions, or reopens the assignment. Initialization that unconditionally sets properties can therefore overwrite user-entered values or repeat an operation that should occur only on the first opening of that particular action.

The data transform should be designed to be idempotent: first evaluate a persisted indicator or the relevant target property values, then perform initialization only when that indicator shows the action has not yet been initialized. After setting the initial values, set the indicator so later executions of the same pre-processing transform preserve the existing values. The indicator should be scoped appropriately to distinguish one flow action from another when the assignment offers multiple actions. This approach makes the behavior explicit and reliable regardless of how often the user navigates to a flow action.

Pega data transforms support conditional logic for precisely this kind of property initialization and update behavior. See Pega documentation on creating data transforms and configuring flow actions.

QUESTION NO: 13

While running a process, you notice that a read-only field on a form contains a value.

Which tool allows you to determine if a declare expression was used to calculate the value?

A. Declarative network

B. Clipboard tool

C. The Tracer

D. Live UI

ANSWER: A

Explanation:

Declarative network is the correct tool because it provides a visual map of declarative relationships between properties and declarative rules in an application. When investigating a value that appears in a read-only field, you can use the network to locate the target property and see whether a Declare Expression rule calculates or derives that property's value. The network shows dependencies, including the input properties that can trigger recalculation and the declarative rule that supplies the result. This is particularly useful when the value is not explicitly set by a flow action, data transform, or activity, because declarative processing can update a property automatically when its source values change. By tracing the property relationship in the Declarative Network, a developer can confirm the calculation path and identify the relevant Declare Expression rule for further review. Pega describes declarative processing as behavior that automatically maintains values based on defined dependencies, and the Declarative Network is intended to help analyze those dependencies. See Pega Academy's Declarative Network topic and its overview of declarative processing.

QUESTION NO: 14

A transaction dispute case type allows customers to dispute a bank card transaction for one of three reasons:

* Stolen card

* Duplicate charge

* Fulfillment error

Most dispute for the stolen cards occur at two vendors. MegaMart and Maxvalu. The fraud analyst requests a report that displays stolen card dispute meeting the following thresholds:

Given the following filter conditions, which logic statement returns the correct subnet of transaction disputes?

A. F1 AND F2 AND F3 AND F4 AND F5

B. (F1 AND F3) OR (F3 AND F4) OR F5

C. ((F1 AND F3) OR (F2 AND F4)) AND F5

D. (F1 AND F3) OR (F2 AND F4) AND F5

ANSWER: C

Explanation:

((F1 AND F3) OR (F2 AND F4)) AND F5 correctly models the requested report population by grouping each vendor with its applicable threshold, then requiring the stolen-card dispute condition for every returned record. The first grouped condition, *F1 AND F3*, identifies disputes associated with MegaMart that satisfy MegaMart's threshold. The second group, *F2 AND F4*, identifies disputes associated with Maxvalu that satisfy that vendor's threshold. Using *OR* between these complete groups allows a dispute from either vendor to qualify, without mixing a vendor condition with the other vendor's threshold.

The final *AND F5* is outside the vendor groups, so it applies to both possible vendor-and-threshold combinations. This is essential because the analyst wants only stolen-card disputes, not all transaction disputes that happen to meet a vendor threshold. Parentheses make the intended evaluation order explicit: first evaluate each vendor-specific combination, then combine the eligible vendor paths, and finally restrict the result to the stolen-card dispute reason. This follows the Pega report-definition approach of combining filter conditions with explicit logical grouping to form the intended result set. See Pega documentation on report definitions and Pega documentation on defining filter conditions.

QUESTION NO: 15

In the first step in a case type, the user compares data on a form to the data on a customer account. If the data matches, the case is resolved. If the data does not match, the user advances the case to update the account. Management only wants a record of the cases that update an account. What two configuration options do you use to implement this requirement ? (Choose Two)

A. Add a persist case shape after the first step.

B. Configure the starting flow to instantiate the case type as a temporary case.

C. Apply a when condition to the first step to persist only cases requiring updates.

D. Configure the first step to instantiate the case type as a temporary case.

ANSWER: A D

Explanation:

Configure the first step to instantiate the case type as a temporary case so that Pega does not create a persisted case record while the user performs the initial comparison. A temporary case is appropriate for work that might be completed without needing a record in the system of record. Therefore, when the form data matches the customer account data and the user resolves the case at that point, the temporary case ends without producing the unwanted case record.

Add a persist case shape after the first step on the path that continues to the account-update work. The Persist Case shape converts the temporary case into a normal, persisted case. From that point, Pega stores the case and its subsequent processing history, providing management with a record for every case in which an account update is required. Together, temporary instantiation for the comparison and explicit persistence before update processing ensure that only update cases are recorded. Pega documents that temporary cases are not persisted unless the flow explicitly persists them; see Pega documentation on temporary cases and Pega Academy: Case processing.

QUESTION NO: 16

In designing your application, you want to apply consistent visual styles to all parts of the application. How do you meet this requirement?

A. Specify a skin in the application rule.

B. Apply styles to the screen layout.

C. Use the Live UI tool to select the skin rule.

D. Specify a skin in the harness rule.

ANSWER: A

Explanation:

Specify a skin in the application rule. In Pega, a skin is the centralized collection of presentation settings that controls the visual design of an application, including typography, colors, borders, spacing, controls, layouts, and other user-interface elements. Associating the skin with the application rule makes that styling available consistently throughout the application's user interface, rather than requiring developers to configure individual styling choices repeatedly in each screen or layout. This supports a maintainable design system: updates to the skin can be applied centrally and then reflected wherever the application uses the corresponding UI components. Pega's application rule is therefore the appropriate place to establish the application-level skin relationship. A skin can also inherit from another skin, enabling an application to use a base design while making controlled application-specific branding or styling changes. This approach provides both visual consistency and efficient maintenance as the application evolves. For further details, see Pega documentation on skins and styling and the application rule.

QUESTION NO: 17

Users create Insurance Coverage Request Cases to authorize insurance payments. Users

enter information that includes the name of the patient, the date of the procedure and the type of the procedure. After entering the information, user submits the request for a review of the patient's insurance policy. Because multiple users enter requests, duplicate request can occur. A request is considered a duplicate if the patient name, procedure type, procedure date match an existing request. You have been given two requirements:

- Ensure that users can identify duplicate requests.

- If a case is duplicate, it is not written to the database. Otherwise, write the case to the database.

Which two options configure application so that users can identify duplicate requests? (Choose two)

A. Add a duplicate search step to the case life cycle design.

B. Configure a duplicate search decision table and add it to a Decision shape

C. Configure weighted conditions.

D. Configure a validate rule to validate matching cases.

ANSWER: A C

Explanation:

Add a duplicate search step to the case life cycle design. is correct because Pega provides a dedicated duplicate-case-search step that is configured in the case life cycle. The step runs when the relevant request data has been entered, searches existing cases, and presents potential matches to the user. This supports identifying a possible duplicate before the new request proceeds to case creation and persistence.

Configure weighted conditions. is also correct because duplicate searching uses weighted conditions to define how closely a new request must match an existing case. The application can evaluate the patient name, procedure type, and procedure date as matching criteria, assign each criterion an appropriate weight, and set the matching threshold. For this requirement, the three fields can be configured so that a request is identified as a duplicate only when the required matching combination is met. The duplicate-search experience then lets the user recognize the existing request and prevents the duplicate request from continuing to database creation.

Using the specialized duplicate-search capability keeps the matching logic declarative, visible in the case design, and aligned with Pega's case-processing model. See Pega Academy's Duplicate case search topic and the Pega Platform documentation on duplicate case search.

QUESTION NO: 18

A development team plans to enhance functionality of an existing application by changing several user interface rules. The team would like to pilot the enhancements to a small group of users before rolling the changes out to the entire user base. What approach maximizes reuse and maintainability?

A. Place the updated rules into a new minor version of the ruleset and include the new ruleset version in a new application.

B. Place the updated rules into a new ruleset and include the new ruleset in a new application.

C. Place the updated rules into a new ruleset and include the new ruleset in a new version of the application.

D. Place the updated rules into a new minor version of the ruleset and include the new ruleset version in a new version of the application.

ANSWER: D

Explanation:

Place the updated rules into a new minor version of the ruleset and include the new ruleset version in a new version of the application. This approach uses Pega's intended versioning model to isolate the pilot while preserving reuse of the existing application and ruleset structure. The new minor ruleset version contains the UI rule changes and can override the prior version's rules where needed, without copying unchanged rules. A new application version can then reference that newer ruleset version for pilot users, while the existing application version remains available to users who should continue using the current UI behavior.

After the pilot is validated, the newer application version can be promoted and made available more broadly. This creates a clear, maintainable record of which application version uses which ruleset version, supports controlled deployment, and avoids unnecessary duplication of the application or its rules. Pega applications are assembled from rulesets and ruleset versions, so aligning an application version with the revised minor ruleset version provides controlled evolution of the application. See Pega's application versioning guidance and the ruleset versioning guidance.

QUESTION NO: 19

Which two configurations do you use to validate the minimum age of a new potential customer in the Collect Account Information assignment step? (Choose Two)

A. Create and Edit Validate rule to check the customer age.

B. Reference the Edit Validate rule on the Collect Account Information flow action.

C. Reference the Validate rule on the Collect Account Information flow action.

D. Reference the Validate rule on the Collect Account Information assignment.

E. Reference the Edit validate rule on the Collect Account Information assignment.

F. Create a Validate rule to check the customer age.

ANSWER: C F

Explanation:

To enforce a minimum-age requirement when a user submits the Collect Account Information form, create a Validate rule to check the customer age and reference the Validate rule on the Collect Account Information flow action. A Validate rule is the Pega rule type intended for business validations that can compare entered data with a required condition, such as confirming that the customer's age meets a defined threshold. The rule can add an appropriate message to the relevant property when the condition is not met, preventing the flow action from completing successfully until valid information is supplied.

The flow action is the appropriate place to associate this validation because it controls what happens when a user submits the assignment form. Configuring the Validate rule on the flow action ensures that Pega runs the age validation as part of processing that user action. This supports reusable, declarative validation logic while keeping the assignment focused on routing work and presenting the task. Pega training material describes validating user-entered data through validation rules and flow-action configuration; see Pega Academy: Validating user input and Pega documentation: Creating a Validate rule.

QUESTION NO: 20

An accident claim case creates a vehicle claim case for each vehicle involved in an accident. Which two configurations prevent the accident claim case from resolving before all vehicle claims are resolved? (Choose Two)

A. Add each vehicle claim as a child case of the accident claim.

B. Add a manual approval step to the accident claim case.

C. Add an optional process to pause the accident case until the vehicle claims are paid.

D. Add a wait step to the accident claim case to wait until ail vehicle claims have a status of Resolved.

ANSWER: A D

Explanation:

Adding each vehicle claim as a child case of the accident claim establishes a parent-child case relationship. Pega manages this relationship as part of case processing, so the parent accident claim remains open while its child vehicle claims require completion. This models the business structure directly: the accident is the overall claim, and each vehicle claim is work that belongs to that overall claim. Pega's case-management features support creating and tracking related child cases from a parent case; see Pega Case management documentation.

Adding a wait step to the accident claim case to wait until ail vehicle claims have a status of Resolved also prevents premature resolution by explicitly pausing the parent's processing until the required status condition is met for every vehicle claim. A wait shape can suspend a case at a defined point and resume its flow only after its configured condition becomes true. Therefore, the accident claim cannot advance to its resolution processing until all associated vehicle claims are resolved. This is appropriate when the parent workflow must enforce a specific completion condition rather than merely continue independently. Pega Academy's system architect learning resources cover case lifecycle and process orchestration concepts: Pega Academy.

QUESTION NO: 21

In an application that sells office supplies, the Payment view displays order items and collects payment information. In the Payment section rule, the order items are grouped in a dynamic layout. You find out later that the Order Summary view must also display the order items.

How do you configure the UI so that the order items display is shared between the Payment

view and Order Summary view?

A. Convert the Payment section layout that contains order items to a section, and embed this section in the Order Summary section.

B. Build the Order Summary section with a layout inside to group the order items, similar to the Payment section.

C. Embed the Payment section in the Order Summary section.

D. Reuse the Payment section in the Order Summary view and use a disable when condition to disable payment information on the Payment section rule.

ANSWER: A

Explanation:

Convert the Payment section layout that contains order items to a section, and embed this section in the Order Summary section. This creates a reusable UI component for the common order-item display rather than duplicating its fields, presentation, and configuration. In Pega, a section is a reusable rule that can be included in other sections, views, or harnesses. By placing the order-items dynamic layout in its own section, both the Payment UI and the Order Summary UI can render the same maintained definition.

The extracted section should contain only the shared order-item content, such as the repeating data display, labels, visibility logic, and layout configuration that are applicable in both contexts. The Payment section retains its payment-specific fields and embeds the shared order-items section; the Order Summary section also embeds that same shared section. Consequently, a later update to the item display is made once in the reusable section and is automatically reflected wherever it is included. This supports Pega's rule-reuse approach, improves consistency between views, and reduces the risk that duplicated UI definitions diverge over time. See Pega Academy's guidance on sections and its overview of views.

QUESTION NO: 22

An organization has two lines of business: selling books for children and reselling college textbooks. The division selling books for children can use the same basic user interface (UI) as the division reselling textbooks with the exception of the payment methods. How do you apply the Situational Layer CakeTM in this scenario ?

A. Place the UI rules in the base layer, and create a new layer for the payment rules for both lines of business.

B. Place the UI rules and generic payment method rules in the base layer, and create a new layer for the division-specific payment rules.

C. Place the UI rules in the base layer, and create a parallel base layer for the payments rules.

D. Place the UI rules in the base layer, and create a new layer for the payment rule for each division.

ANSWER: D

Explanation:

Place the UI rules in the base layer, and create a new layer for the payment rule for each division. This applies the core principle of Pega's Situational Layer Cake: put reusable, shared capabilities in a lower layer and put business-unit-specific behavior in higher, specialized layers. Because both divisions use the same basic user interface, the UI rules belong in the base layer so that they are defined once and can be inherited by both lines of business. Payment behavior is the stated point of variation, so each division needs its own specialized layer containing the payment rule appropriate to that division. This design keeps the application modular, supports reuse, and isolates each division's payment implementation from the common UI. It also makes future change safer: a UI enhancement made in the base layer is available to both divisions, while a change to a payment method can be made in the relevant division layer without unnecessarily affecting the other line of business. Pega application layering is intended to organize reusable enterprise functionality and progressively specialize it for divisions or implementation contexts. See Pega documentation on application layering and Pega's Application layering learning material.

QUESTION NO: 23

On the Case Designer, who two component can users tag with a Minimum Lovable Product (MLP) release for project sizing purposes? (Choose Two.)

A. A user Mobile App channel

B. An external system of record (SOR)

C. An automation

D. An approval decision

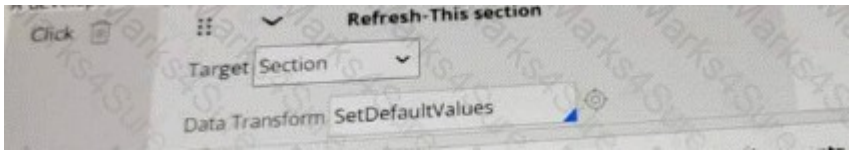
ANSWER: A D

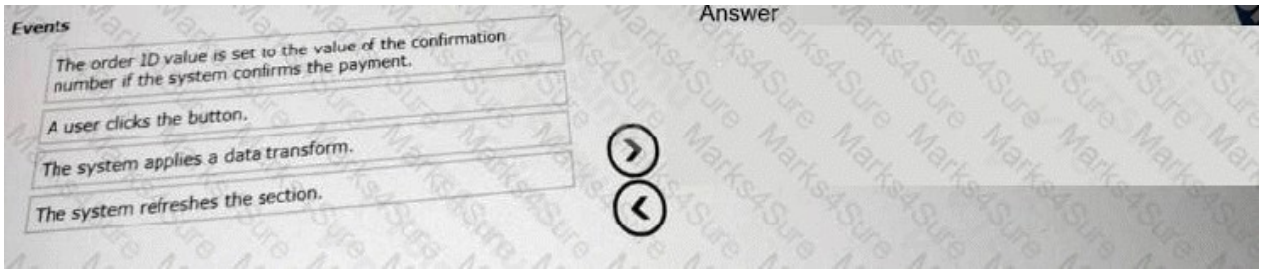
Explanation:

An MLP release tag helps a team organize the application backlog into incrementally deliverable slices and estimate the work associated with each planned release. In Case Designer, a user can apply this planning information to a Mobile App channel, allowing the team to identify when a mobile experience is intended to be included in the delivery roadmap. A user can also tag an approval decision, so that the approval capability can be associated with the appropriate MLP and considered during release sizing. These tags support the Pega Express approach of defining a minimum lovable product: a meaningful, usable initial release that delivers prioritized business value, followed by subsequent enhancements in later releases. Release tagging therefore provides a design-time planning aid across relevant case-design components rather than changing the underlying runtime behavior of the channel or approval. For background on Pega's iterative delivery methodology and MLP planning, see Pega Academy: Pega Express. Pega's product documentation portal also provides current platform documentation and version-specific guidance at Pega Documentation.

QUESTION NO: 24 - (SIMULATION)

A developer configures a button with the action set as shown in the following image.





ANSWER: See the explanation for the answer

Explanation:

Correct event sequence:

The *Refresh-This section* action runs its configured data transform before refreshing the section. Therefore, the updated order ID value is available when the section is rendered again.

QUESTION NO: 25

Which two statements demonstrate the role of a report? (Choose Two)

A. Reports are used to assess process performance.

B. Reports are used to update data in a database.

C. Reports are used to select items from a list while working in an assignment.

D. Reports are used to source a list of selectable items while working in an assignment

ANSWER: A C

Explanation:

Reports are used to assess process performance because Pega report definitions retrieve and organize persisted application data into useful views. For example, a report can summarize work volume, assignment completion, resolution time, aging, SLA-related information, or operator workload. These results help users and managers monitor operational performance and identify trends that require attention. Reports are also used to select items from a list while working in an assignment. A report definition can supply the data shown by reporting controls in a user interface, allowing an application to present relevant records or values to an operator as part of completing work. This lets a case participant make an informed selection using current application data without needing to manually enter the value. Pega's reporting capability is therefore both an analytical feature for reviewing business and process data and a reusable way to present queried data within an application experience. See Pega's documentation on reports and report definitions.

QUESTION NO: 26

An insurance claim case type is defined as follows: If the Review claim step is configured to set

the status to Pending-Investigation, when is the status of the case set to Pending-Investigation?

A. When Investigate claim step completes

B. When the Process claim stage starts

C. When the Review claim step completes

D. When the Review claim step starts

ANSWER: D

Explanation:

When the Review claim step starts, Pega sets the case status to *Pending-Investigation*. A status configured on a case-life-cycle step represents the current point or condition of the case while that step is active. Therefore, Pega applies the configured status as the Review claim step begins and its work is made available, allowing users, reports, queues, and integrations to identify immediately that the claim is awaiting investigation-related review activity.

Case statuses provide meaningful business visibility beyond the standard workflow progression. They can communicate the case's current condition, such as pending review, awaiting information, or resolved, and are available for use in views and reporting. Configuring the status on the Review claim step ensures that the status changes at the transition into that work, rather than requiring completion of the step or of a later step. This makes the case status accurately reflect the work currently underway. See Pega Academy's Case status topic and the Pega documentation for case status management.

QUESTION NO: 27

Several Development teams work on different enhancements.

The release date for each enhancement is uncertain. Which two options allow each team to keep its work separate? (Choose Two)

A. Create a new ruleset version for each team.

B. Set up a branch ruleset for each team

C. Create a new application for each team

D. Create a production ruleset for each team.

ANSWER: A B

Explanation:

Creating a new ruleset version for each team keeps rule changes segregated because Pega stores rules by ruleset and version. Each team can develop its enhancement in its own version rather than directly changing a shared ruleset version. This provides a distinct rule layer for the team's work and supports controlled promotion when that enhancement is ready for release. Pega's ruleset versioning model is designed to preserve and manage rule changes across application development and deployment.

Setting up a branch ruleset for each team is also appropriate, particularly when release timing is independent or uncertain. A branch is an isolated development area based on an application ruleset stack. Teams can create and test rules in their own branches without exposing incomplete changes to the shared application. When an enhancement is approved and ready, the team can merge its branch rules into the target ruleset version. This workflow allows separate development streams to progress independently while retaining a managed path for integrating changes. Pega documents branch rulesets as a mechanism for isolating work and merging completed changes into the development ruleset stack.

See Pega documentation on branch rulesets and Pega Academy: Rulesets.

QUESTION NO: 28

Which two rules do you localize by using the localization wizard? (Choose Two)

A. Work Parties

B. Paragraph

C. Correspondence Fragment

D. Field Value

ANSWER: B D

Explanation:

The Localization wizard is used to prepare application text for translation by creating localized rule instances for supported rule types. It can localize **Paragraph** rules, which hold reusable blocks of user-facing text that can appear in sections, correspondence, and other UI content. This makes the same paragraph available in the language selected by an operator or requestor without requiring duplicate application design.

The wizard also localizes **Field Value** rules. Field values are Pega's standard mechanism for storing short, reusable text such as labels, button captions, prompts, messages, and other UI strings. A localized field value supplies the appropriately translated text according to the active locale, allowing the application interface to adapt consistently across languages.

Both rule types therefore contain translatable, user-visible text and are supported by Pega's localization workflow. Pega localization uses language-specific rule instances and rule resolution to select the content matching the current locale. For background on Pega localization and language support, see Pega Documentation: Localizing an application and Pega Documentation: Field Value rules.