PassQueen
a.k.a dumpsqueen.com

Certified Pega Lead System Architect (CPLSA) Exam 8.8

Pegasystems PEGACPLSA88V1

Version Demo

Total Demo Questions: 25

Total Premium Questions: 259

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Application Design	87
Topic 2, Case Management	28
Topic 3, Data and Integration	37
Topic 4, User Experience	25
Topic 5, Security	29
Topic 6, Reporting	17
Topic 7, Background Processing	34
Topic 8, Mix Questions	2
Total	259

QUESTION NO: 1

You oversee a medium-size development team, and some of the team members are new to

pega. What are the most efficient ways to ensure that the rules the team creates adhere to

best practices? (choose two)

A. Have new team members create and run Pega automated unit tests against their rules.

B. Use Pega Log Analyzer to identify exceptions associated with the new team members.

C. Run Tracer on each rule the new team members check in to identify an failures in rule execution.

D. Leverage the rule check-in approval process to review the new team member's changes first.

ANSWER: A D

Explanation:

Have new team members create and run Pega automated unit tests against their rules. Unit tests provide a repeatable, developer-owned way to validate a rule's expected behavior with defined test cases and data. Making unit testing part of normal development gives new team members prompt feedback as they build, helps establish reliable regression coverage, and lets the team verify behavior again after subsequent rule changes. Pega's unit-testing capabilities are designed to support this kind of rule-level validation; see Pega Academy: Unit testing.

Leverage the rule check-in approval process to review the new team member's changes first. A check-in approval workflow adds a targeted governance control before a new developer's changes become available to the wider application. Experienced reviewers can assess implementation choices, guardrail alignment, reuse of existing rules, naming, and completeness while also coaching the developer in the team's standards. This is efficient because review is focused on the actual change set and can be applied selectively to developers or rulesets that require additional oversight. Pega rule versioning and team-development practices support controlled collaboration and change management; see Pega Academy: Ruleset versioning.

QUESTION NO: 2

U+ Bank currently uses a Pega Platform™ application to automate its internal operations. The bank wants to add independent tax processing functionality to its application. This work should be routed to tax consultants in the application.

To support this new feature, the bank has appointed country-specific tax consultants because tax processing is handled differently in different countries.

Which one of the following options is the best approach to add tax processing to the bank's existing Pega Platform application?

A. Create a tax case type, then circumstance the required processes and routing utilities by using the country property.

B. Create a country-specific tax assessment application that is built on the existing Pega Platform application.

C. Create a tax case type and use case type specialization by country, each with its own process, to be routed and assigned to the appropriate tax consultant.

D. Create a country-specific tax process for tax assessment that is routed to the appropriate tax consultant.

ANSWER: A

Explanation:

Create a tax case type, then circumstance the required processes and routing utilities by using the country property. This approach introduces tax processing as an independent, reusable case type in the existing application while retaining a common case lifecycle and shared tax-processing capabilities. The country value is a natural business property on which to circumstance the rules that genuinely vary, such as the process flow, validations, calculations, correspondence, and assignment-routing behavior. At run time, Pega selects the most appropriate circumstanced rule for the country value,

allowing the application to apply country-specific tax handling without duplicating the entire case type or creating separate applications.

Routing can likewise use country-aware rule resolution or a decision-based routing utility to send each assignment to the appropriate tax-consultant work group, operator, or queue. This design keeps common behavior centralized, makes country-specific changes isolated and maintainable, and supports adding countries by adding only the needed circumstanced rules. Pega circumstancing is specifically intended to provide rule variations based on properties and other contextual conditions, while routing rules determine how assignments reach the correct users or work queues. See Pega Academy: Circumstancing and Pega Academy: Routing work.

QUESTION NO: 3

A development team uses branches for feature development. Before a feature branch is merged into the application ruleset, the team wants automated evidence that critical rule behavior has not regressed and a reviewable record of rule-quality issues.

Which two practices best support this requirement? (Choose Two)

A. Create and run PegaUnit test cases for the critical branch rules.

B. Review branch quality and address Guardrails warnings before the merge.

C. Run Tracer manually for every rule in the branch before the merge.

D. Use Pega Log Analyzer as the primary pre-merge regression test.

ANSWER: A B

Explanation:

Create and run PegaUnit test cases as part of the branch validation process. PegaUnit provides repeatable verification of rule behavior and can identify regressions before branch contents are merged.

Review branch quality for Guardrails warnings. Branch quality provides visibility into quality issues in the branch, allowing the team to correct or document concerns before introducing the rules into the shared application ruleset.

Tracer is useful for interactive diagnosis of a specific execution, but it is not a scalable regression-testing mechanism. Pega Log Analyzer is used to investigate log data and exceptions rather than to provide pre-merge rule-quality validation.

QUESTION NO: 4 - (SIMULATION)

The following figure depicts a hierarchy of applications that are built on other applications.

Specifically , the productionapp application is built on applications customerapp and

employeeapp. Each of these applications has additional built-on applications, including the duplicated MYEnterpriseApp application, which has two different versions in use. All applications are built on the same version of PegaRUUS.

ANSWER: See the explanation for the answer

Explanation:

Use the following application hierarchy/version path:

```
ProductionApp 01.01
  CustomerApp 07.01
    ServiceApp 01.01
      EmployeeApp 01.01
        MyEnterpriseApp 01.02
          PegaRULES 07.10
```

The shared enterprise application must resolve to `MyEnterpriseApp 01.02` in this hierarchy. The platform application at the base of the stack is `PegaRULES 07.10`.

QUESTION NO: 5

What are two valid reasons for defining a case type within a case type-specific ruleset? (Choose two)

A. Case-specific rulesets make it easier to rebase ruleset versions.

B. Each branch ruleset can be associated to case-specific user stories.

C. The ruleset can be added to the ruleset stack for multiple applications.

D. The case type might be converted to a component application in the future.

ANSWER: B D

Explanation:

Each branch ruleset can be associated to case-specific user stories because organizing rules by case type provides clear ownership and development boundaries. A branch can contain the rules needed for a specific enhancement to that case, making it easier for teams to associate work with the relevant backlog items, review the changes in context, and merge completed work while minimizing overlap with unrelated case processing. This structure is especially useful when several teams are delivering features for different case types in parallel. Pega's branch ruleset guidance describes branches as an isolated development mechanism that supports collaboration and controlled merging into the target ruleset.

The case type might be converted to a component application in the future is also a valid reason. A case-specific ruleset helps keep the case's rules cohesive and limits unnecessary dependencies on rules for other case types. That modularity makes subsequent extraction, packaging, and reuse of the case capability more manageable if the implementation is later promoted into a component application. Component applications are intended to encapsulate reusable functionality that can be built and maintained independently, so a focused ruleset organization supports that architectural goal from the start. See Pega Academy: Branch rulesets and Pega Academy: Component applications.

QUESTION NO: 6

A pega application has cases that represent customer accounts each with many members.

When a member of a customer account registers with the application through an offline

component, a related registration transaction is recorded. An advanced agent updates the

customer account cases with new members. The application is running in a multimode

system and advanced agents are enabled on all nodes. Which two elements are valid design

choices? (choose two)

A. Use the optimistic locking option on the case types.

B. Create a Registration subcase configured to run in offline mode.

C. Leverage the default object lock contention requeuing capability.

D. Override DetermineLockString to use .AccountID instead of .pyID as the lock string.

ANSWER: A D

Explanation:

Using the optimistic locking option on the case types is appropriate because registration activity can be processed asynchronously by advanced agents running on multiple nodes. Optimistic locking allows processing to proceed without

holding a database lock for the entire duration of the work. When the account case is saved, Pega verifies that the instance has not been changed by another requestor since it was opened. This design is particularly suitable where concurrent updates are possible and the application can handle a detected update conflict through retry or other case-processing logic.

Overriding DetermineLockString to use `.AccountID` instead of `.pyID` is also a valid design when the business account identifier is the resource that must be serialized. A consistent account-based lock string makes all updates for the same customer account contend for the same lock, regardless of the node on which an advanced agent executes. In a multimode deployment, Pega lock handling is coordinated across nodes, so this protects account-level consistency while registrations are being applied. See the Pega documentation search for optimistic locking and DetermineLockString.

QUESTION NO: 7

your application connects to two REST services that list details about bank offices.

One REST service uses a postal code as a GET parameter and returns a detailed list of bank officers in that postal code.

the other REST service uses a city name as a GET parameter and returns a detailed list of back offices in cities with that name. The application uses data pages to cache details about bank offices.

select two options to configure a data page to accept either a postal code or a city name as a parameter and call the appropriate REST connector.(choose TWO)

A. sourcing data from both rest connectors and using the response data transform to select the correct data

B. sourcing data from a single REST connector that is circumstanced based on the input parameter

C. sourcing data from each REST connector separately and using a when rule to select the appropriate data source at run time

D. sourcing data from an activity that evaluates the parameter to select the appropriate REST connector to call at run time

ANSWER: C D

Explanation:

Sourcing data from each REST connector separately and using a when rule to select the appropriate data source at run time is correct because a data page can define multiple data sources. A When rule on each source can determine whether that source applies for the supplied parameter values. For example, one source can be selected when a postal-code parameter is present, while the other is selected when a city-name parameter is present. Each selected source can pass its relevant parameter to its REST connector, and the parameterized data page cache keeps results distinct for different input values.

Sourcing data from an activity that evaluates the parameter to select the appropriate REST connector to call at run time is also correct. A data page may use an activity as its source. The activity can inspect the supplied data-page parameters, invoke the appropriate connector rule, and map the connector response into the data page's expected structure. This approach is useful when source-selection logic needs procedural handling or additional processing before or after the connector invocation. In either design, defining the postal code and city name as data page parameters provides the input context needed to retrieve and cache the appropriate bank-office data. See Pega's Data pages documentation and REST connectors documentation.

QUESTION NO: 8

An e-commerce application offers special discounts on a seasonal basis. A business manager is authorized to maintain these discounts.

Which of the following options is the best possible solution for maintaining the discounts?

A. Create a case that performs the data updates.

B. Use a list landing page without the default data sources.

C. Provide a Harness to update the discount details.

D. Use a list landing page and define the default data sources.

ANSWER: D

Explanation:

Use a list landing page and define the default data sources. A landing page is the appropriate user-facing pattern when an authorized business user must routinely view and maintain a collection of business records, such as seasonal discount definitions. A list landing page presents the available discount records in a consistent, navigable interface and supports opening an individual record for maintenance. This gives the business manager a purpose-built workspace rather than requiring the user to initiate a business process for each administrative change.

Defining default data sources is essential because it supplies the landing page with the data that it must display when the manager opens it. The data source configuration establishes how Pega retrieves the discount list, while the landing-page configuration makes that list available in the portal navigation. Together, these features separate the user experience for reference-data administration from the underlying persistence and integration details. This supports low-code maintenance, appropriate access control, and a straightforward operational experience for a nontechnical business manager. Pega documentation describes landing pages as portal-accessible interfaces for displaying application information and configuring their supporting data. See the Pega Platform documentation on landing pages and the Pega Academy for platform design guidance.

QUESTION NO: 9

A Citizen Developer working on a Claims case type wants to validate the BenefitType property value. This property is located on a Clipboard page named ClaimBenefit of the MyCo-Data-ClaimBenefit class, which is inherited from Data-.

As a Lead System Architect, what do you recommend to make this validation available to the Citizen Developer?

A. A System Architect should create a When rule in Dev Studio and reference the ClaimBenefit Clipboard page on the Pages it Classes tab. They define and implement the required criteria, because it is not possible to make it available for Citizen Developers.

B. A System Architect should create a property BenefitType in the work class and copies the BenefitType property value from the ClaimBenefit Clipboard page to pyWorkPage. Citizen Developers can check the criteria on work class BenefitType property.

C. A Citizen Developer uses any existing property and informs the system architect to implement a constraint rule. The System Architect marks the constraint rule as a relevant record, to make it visible to the Citizen Developer.

D. A System Architect should create a When rule in Dev Studio and reference the ClaimBenefit Clipboard page on the Pages & Classes tab. They mark the rule as a relevant record so that the rule becomes available to Citizen Developers.

ANSWER: D

Explanation:

A System Architect should create a When rule in Dev Studio and reference the ClaimBenefit Clipboard page on the Pages & Classes tab. They mark the rule as a relevant record so that the rule becomes available to Citizen Developers. A When rule is appropriate because it evaluates a Boolean condition and can be used as reusable validation or conditional logic. Since `BenefitType` resides on a named clipboard page rather than directly on the primary work page, declaring `ClaimBenefit` and its class on the rule's Pages & Classes tab gives the rule a properly typed reference to that page. The condition can then evaluate the property as `ClaimBenefit.BenefitType` without copying data into the work class merely to make it accessible.

Marking the rule as a relevant record is the key step for low-code authoring. Relevant records expose selected, approved rules for use in App Studio, allowing the Citizen Developer to select centrally governed logic while the System Architect retains responsibility for its implementation and rule context. This supports reuse and avoids duplicating validation expressions across the application. See Pega's guidance on creating When rules and its documentation for relevant records.

QUESTION NO: 10

MyHealth Corporation wants to use the age of the claim to increase the urgency of the assignment so that persons processing the claims work on the most urgent claims first. The claim assignment urgency increases by 1 each day the claim remains in an Unresolved status. At any time, MyHealth has up to 10,000 claims that are in process. Claims in the PendingProcessing workbasket are subject to this calculation. The application updates the claim urgency daily before the work day begins. All claims are processed within 30 days.

Which approach satisfies the claim urgency requirement and provides the best experience for the user who processes the claims?

A. Use the service-level agreement of the case to increase the value of pyUrgencyAssignAdjust by 1. Reset the goal date to the following day.

B. Use a declare expression to increment the value of pyUrgencyAssignAdjust by 1. When a user who processes claims opens the assignment, the urgency is increased by 1.

C. Use a job scheduler on a dedicated node to increase the value of pyUrgencyAssignAdjust by 1 for every assignment that matches the selection criteria.

D. Use the service-level agreement of the assignment to increase the value of pyUrgencyAssignAdjust by 1 every day the claim is in an Open status.

ANSWER: C

Explanation:

Use a job scheduler on a dedicated node to increase the value of pyUrgencyAssignAdjust by 1 for every assignment that matches the selection criteria. A job scheduler is designed to run background processing at a defined time and frequency, making it appropriate for a daily, pre-workday update. The scheduled job can identify the open assignments in the PendingProcessing workbasket whose parent claims remain unresolved, then apply the urgency adjustment without requiring a user to open any assignment.

This approach provides a consistent queue-ordering experience: when claim processors begin work, assignment urgency has already been calculated for the entire eligible population, including potentially 10,000 in-process claims. pyUrgencyAssignAdjust is the standard assignment urgency adjustment property, so incrementing it preserves Pega's normal assignment prioritization behavior while allowing the age-based business rule to influence work selection. Running the job on a dedicated background-processing node isolates the batch activity from interactive requestor processing and supports predictable daily execution. Pega documents job schedulers as the mechanism for recurring background tasks and describes node classification for allocating such processing appropriately. See Pega documentation on job schedulers and Pega documentation on node classification.

QUESTION NO: 11

One of the important requirements of Pega Web Mashup is to ensure that the same-origin policy is following.

From the following options, select two valid configurations. (Choose TWO)

A. Configuration A

B. Configuration B

C. Configuration C

D. Configuration D

ANSWER: A B

Explanation:

Configuration A and **Configuration B** are the valid selections indicated by the supplied configuration rationale. A Pega Web Mashup implementation embeds a Pega gadget in a host web page, and browser same-origin rules apply to the interaction between those resources. An origin is the combined value of the URL scheme, hostname, and port. Therefore, the host page and the gadget must use matching origin values when the mashup requires same-origin access. For example,

a host page at `https://www.pega.com/MyWebApp/home.html` and a gadget at `https://www.pega.com/MyWebApp/gadgetA.html` share the `https` scheme, the `www.pega.com` host, and the default HTTPS port, which is 443. Differences in the path portion do not change the origin, so separate application paths can still be used safely under the same-origin requirement. In deployment design, teams should explicitly confirm scheme, host, and port values, including any reverse proxies or nondefault ports, before publishing the mashup. Pega's Web Mashup documentation describes embedding Pega content in external web applications, while MDN provides the browser definition of origin and same-origin behavior. See Pega Web Mashup documentation and MDN Same-origin policy.

QUESTION NO: 12

You are implementing a paga customer service application and are integrating the system of record data for account information?

select the three tasks required to implement this integation

A. Specialize the pega customer service data pages that use account data.

B. Integrate with the system of record in the pega customer service account class

C. Rename the pega customer service data pages to match the system of record.

D. Integrate with the system of record to populate an Account data class.

E. Create a data transform to map the Account data class to the pega customer service Account class

ANSWER: A D E

Explanation:

Specialize the pega customer service data pages that use account data, integrate with the system of record to populate an Account data class, and create a data transform to map the Account data class to the pega customer service Account class are the required implementation tasks. Pega Customer Service uses data pages as the reusable, on-demand access layer for customer and account information. Specializing the relevant data pages lets the application preserve the Customer Service data-page contract while configuring source-specific loading behavior for the organization's system of record.

The integration should populate a separate Account data class. This provides a clear integration-layer data model that represents the response supplied by the external account system, rather than coupling the connector directly to the Customer Service application class. A data transform then maps that source-oriented Account structure into the Customer Service Account structure expected by the application. This separation makes field conversion, normalization, and maintenance explicit, and allows the customer-service features to consume account data consistently regardless of the external source format. Pega's documentation on data pages and the Pega Academy integration learning resources describe these reusable data-access and mapping patterns.

QUESTION NO: 13

An insurance company wants to extend a native mobile app to allow customers to create claims using a claims management application implemented on the pega platfrom. As a claim is processed, update are sent to the mobile app as push notifications.

How do you satisfy this requirement ?

A. Package the claims management application as a SDK mobile app

B. Configure a service to creat claim cases when called from the native mobile app

C. configure the native mobile app to creat a claim case using the page API

D. Embed the claims management application in the native mobile app using a pega web Mashup gadget

ANSWER: B

Explanation:

Configure a service to creat claim cases when called from the native mobile app is the appropriate integration approach because the existing native application needs a supported server-side interface through which it can request case creation in Pega Platform. A Pega REST service can receive the claim information submitted by the mobile client, authenticate and authorize the request, map the request payload to the required Pega data, and create a claim case using the application's normal case-processing logic. This keeps the claim lifecycle, validations, routing, service-level processing, and auditing in the claims management application rather than duplicating business logic in the mobile client.

The native app can call the service when a customer submits a claim, retain the case identifier returned by Pega, and use that identifier to correlate later status information. As the claim progresses, the Pega application can trigger the organization's configured mobile push-notification process for relevant case events. This provides a clean API boundary between the native app and Pega while allowing Pega to remain the system that owns and processes claim cases. Pega's integration documentation describes exposing application capabilities through REST services: Pega documentation: Creating REST services. See also the Pega Academy REST services topic.

QUESTION NO: 14 - (SIMULATION)

U+ Bank's marketing department wants to use the always-on outbound approach to send promotional emails about credit card offers to qualified customers. As a part of this promotion, the bank wants to identify the starting population by defining a few high-level criteria in a segment.

For each condition below, select which Two conditions should be defined in Segment and which three conditions should be defined in Engagement policy as best practice.

ANSWER: See the explanation for the answer

Explanation:

Define these conditions in the Segment because they are high-level customer attributes used to establish the initial eligible population:

Monthly income: monthly income is less than 20,000.
Credit score: credit score is greater than 200.

Define these conditions in the Engagement policy because they control eligibility, suitability, and communication/contact permissions for an offer:

Opt-in status: the customer has opted in to receive promotional emails.
Debt-to-income ratio: apply the debt-to-income eligibility/suitability condition.
Opt-out status: the customer has not opted out of credit-card promotions on mobile phones.

Segments should remain broad and identify the starting population. Engagement policies then refine that population using consent, contact preferences, and offer-suitability restrictions.

QUESTION NO: 15

A user report poor application performance at 5:00 pm yesterday. which two tools do you use to review the user's session to identify the cause of the poor performance ?

A. performance profiler

B. my performance Details report

C. Alerts log

D. tracer

E. My Performance Details report and Alerts log

ANSWER: E

Explanation:

My Performance Details report and Alerts log is the correct combination for investigating a performance problem reported after the affected session has already occurred. The My Performance Details report provides a retrospective view of performance measurements associated with a user's requestor session. This lets an administrator examine elapsed time and resource-related indicators for the relevant period, rather than attempting to reproduce the issue in a currently active session. Use the reported time and the affected user's identity to locate the appropriate session details.

The Alerts log complements those session metrics by recording threshold-based performance events generated while requests execute. Alerts include diagnostic context such as timing, requestor information, rule or interaction details, and database or external-service activity when applicable. Correlating alert timestamps with the user's reported 5:00 pm incident and the My Performance Details data helps identify the request or resource consumption responsible for the degraded experience. This is the appropriate diagnostic approach for an issue in a completed, historical user session. Pega's monitoring guidance is available in the Pega Platform documentation, including material on Pega alerts.

QUESTION NO: 16

Ar.mo Corporation is designing an Order Fulfillment application built on an Inventory application. Both applications reuse a section that displays Part details.

Where do you configure the PartOetails section?

A. In an Inventory ruleset within the Inventory application's work pool class

B. In an Order Fulfillment ruleset within the Order Fulfillment application's Parts data class

C. In an Enterprise ruleset within the Inventory application's Parts data class

D. In an Order Fulfillment ruleset within the Order Fulfillment application's work pool class

ANSWER: C

Explanation:

Configure the PartDetails section in an Enterprise ruleset within the Inventory application's Parts data class. The Inventory application is the built-on application for Order Fulfillment, so rules defined in the Inventory layer are available to the Order Fulfillment application through application layering and rule resolution. Placing the section in the Parts data class also correctly aligns the rule with the data it presents: the section displays details of a Part rather than information specific to an Order Fulfillment work object. This design creates a single, reusable definition for the shared UI, allowing both applications to present Part information consistently while avoiding duplicated section rules. An Enterprise ruleset is appropriate for reusable assets intended to be shared across application layers, while the Inventory application's class context ensures that the section is available from the inherited application layer. Pega application layering supports reuse by allowing an implementation application to access rules from its built-on applications, and ruleset organization helps keep reusable rules in the proper ownership layer. See Pega Academy's discussion of application layering and rulesets.

QUESTION NO: 17

Identify two benefits provided by using single sign-on (SSO) in a high availability (HA) application. (choose two)

A. To automatically authenticate users on a new server following a server crash

B. To redirect the user to a new server during a server quiesce (pause)

C. To allow the movement of stateful data between application servers

D. To allow for cookie persistence on the browser

ANSWER: A B

Explanation:

In a high-availability deployment, **To automatically authenticate users on a new server following a server crash** is a key SSO benefit. Authentication is delegated to a shared identity provider rather than being dependent on credentials or authentication state held by one Pega node. When a failed node is removed from service and the load balancer sends a subsequent request to another healthy node, the user can be recognized through the established SSO identity-provider session and authenticated without manually signing in again. This improves continuity during failover.

To redirect the user to a new server during a server quiesce (pause) also benefits from SSO. Quiescing takes a node out of normal request processing so that maintenance can occur without disrupting active users. Traffic management redirects new or resumed requests to an available node, while SSO lets the replacement node authenticate the user consistently through the centralized identity provider. Together, load balancing, node quiescing, and SSO help preserve a seamless user experience during planned maintenance as well as failures. Pega's security documentation describes the platform's SSO integration model and supported authentication approaches; see Pega documentation on single sign-on and the Pega Documentation portal.

QUESTION NO: 18

Your team is developing a Quoting application for BigCo. The team is developing a Policy Administration application. The Quoting application is built on the Policy Administration application. The Policy Administration application requires enhancements to support other areas of the business. These enhancement take place during the same time frame as the Quoting application development.

Select the two practices for your team to follow to minimize potential development conflicts. (Choose Two.)

A. Share rule sets between both application

B. Switch application when modifying rules.

C. Perform branch reviews when merging branches.

D. Periodically rebase the policy Administrator application

ANSWER: B C

Explanation:

Switch application when modifying rules is correct because the Quoting application is built on the Policy Administration application, and rules must be created or updated in the appropriate application layer. Before changing a Policy Administration rule, a developer should switch to the Policy Administration application context so that rule resolution, ruleset selection, and rule checkout occur in the intended layer. This keeps reusable policy functionality owned by its base application while allowing Quoting-specific work to remain in the Quoting layer. Clear layer ownership is essential when related applications are being enhanced concurrently.

Perform branch reviews when merging branches is also correct. Branch reviews provide a controlled collaboration point before changes are merged into the target rulesets. Reviewers can identify overlapping rule changes, validate that changes belong to the correct application layer, and confirm that the merged work is appropriate for the shared application architecture. Using branches and reviewing their contents before merge reduces the likelihood that concurrent development introduces unintended changes or integration issues. Pega documents branch rulesets as a mechanism for isolated development followed by managed merging; see Pega documentation on branch rulesets and Pega Academy branch development guidance.

QUESTION NO: 19

An application uses a property to process customer information that is stored in an external

database. some of the customer information changes infrequently.

how do you ensure the property contains the correct customer information at run time?

A. configure the property to refer to a data page using the SOR patten.

B. configure the user interface (UI) to refresh the information each time it is rebdered.

C. configure the property to copy the customer information from a data page using the snap shot patten.D, configure the property to use a report definition to retrieve the customer information directly.

ANSWER: A

Explanation:

Configure the property to refer to a data page using the SOR patten. The system-of-record pattern is appropriate when customer data is mastered outside the Pega application and the application needs to use the authoritative value at run time. Rather than persisting a separate case-level copy as the working source, the application retains the identifying context needed to obtain the customer record and refers to a data page for the external information. The data page's data source is configured to obtain the record from the external database, and its refresh behavior can be configured to match the required freshness of the information. This provides a controlled integration point, supports reuse of the customer lookup across the application, and allows Pega to cache data only for the configured period while still reloading it when the cache must be refreshed. As a result, customer information used by the property is based on the external system that owns the data. Pega describes data pages as reusable mechanisms for sourcing data from systems of record; see the Pega documentation on data pages and the Pega Academy data pages topic.

QUESTION NO: 20

A banking customer service application is in production that processes various customer complaint requests. After three years in production, the operations manager needs a historical report on resolved cases. This report is to receive in near real time.

Data warehouse has exposed a REST API to receive data. Reports are generated out of data warehouse.

What two of the following options to create an ideal design solution to post data to the data warehouse? (Choose Two)

A. Read data with data flows, which source data by using a dataset and output data to a utility that posts the data to the queue processor, which then posts data to data warehouse over REST. For in-flight cases, on resolution of a case, reuse a queue processor that you created.

B. Read data with data flows, which source data by using a dataset and output data to a utility that synchronously posts the data to data warehouse. For in-flight cases, on resolution of the case, configure the system to post the data to the data warehouse over REST.

C. Run a one-time utility browse all the resolved-cases data, and then post the data to data warehouse asynchronously. For in-flight cases, on resolution of a case, configure the system to synchronously post the data to the data warehouse over REST.

D. Prepare an extract rule and extract the data of already-resolved cases, and then load it into the data warehouse for reporting. For in-flight cases, on resolution of a case, configure the system to post the data to the data warehouse over REST.

ANSWER: A B

Explanation:

"Read data with data flows, which source data by using a dataset and output data to a utility that posts the data to the queue processor, which then posts data to data warehouse over REST. For in-flight cases, on resolution of a case, reuse a queue processor that you created." is correct because a data flow can process the substantial backlog of previously resolved cases from a dataset without requiring an interactive user request to wait. The queue processor then provides asynchronous, scalable delivery to the REST endpoint and supports resilient processing when the external data warehouse is temporarily unavailable. Reusing that queue processor for newly resolved cases gives the historical backfill and ongoing updates a consistent near-real-time integration mechanism.

"Read data with data flows, which source data by using a dataset and output data to a utility that synchronously posts the data to data warehouse. For in-flight cases, on resolution of the case, configure the system to post the data to the data warehouse over REST." is also correct. A data flow is appropriate for reading and processing the existing resolved-case population, while invoking the REST service when a case reaches resolution supplies the warehouse with current data immediately. This approach is suitable when immediate confirmation of the REST submission is required. Pega data flows are designed for processing and transforming data from sources such as datasets, and Pega queue processors provide

asynchronous background processing for integration workloads. See Pega documentation on data flows and Pega documentation on queue processors.

QUESTION NO: 21

An application for the U+ Vehicle Insurance company generates insurance quotes for vehicles specified by the customer. There can be various types of vehicles processed by the application, such as cars, motorcycles, trucks, and so on. The business specifications can also differ for each vehicle type in the quote process.

Which one of the following possibilities is the best data model design for the quote case type?

A. Create a separate data type by grouping similar types of vehicles based on the number of wheels. At runtime, parameterized data pages are used to identify the specific vehicle type.

B. Create a separate data type for each vehicle type, and then create a single vehicle page list in the quote case type. At runtime, dynamically identify the page class for each page in the vehicle page list.

C. Create a separate data type for each type of vehicle, and then create a separate page list in the quote case type. At runtime, a rule resolution algorithm will identify the specific vehicle type.

D. Create a single data type for all vehicle types and a single-page list in the quote case type. At runtime, circumstance will be used to identify the specific vehicle type.

ANSWER: B

Explanation:

Create a separate data type for each vehicle type, and then create a single vehicle page list in the quote case type. At runtime, dynamically identify the page class for each page in the vehicle page list. This design models the real business domain cleanly: a car, motorcycle, and truck can share a common role as a vehicle on a quote while retaining their own attributes and processing requirements. For example, a car might require seating capacity and safety features, whereas a truck can require load capacity and axle information. Separate data types allow each vehicle category to own the properties that are meaningful to it, rather than forcing unrelated fields into one overly broad structure.

The quote case can hold all quoted vehicles in one embedded vehicle page list, which supports quotes containing more than one vehicle and gives the case a single, consistent relationship for vehicle data. Identifying the appropriate page class at run time enables each list item to use the specialized vehicle data structure required for that item. This approach also supports maintainability: future vehicle categories can be introduced with their own data type and behavior without redesigning the quote case data model. Pega data types are intended to represent reusable business entities and their fields, while case data can use embedded page structures to represent related information. See Pega Academy: Data model and Pega documentation: Creating a data type.

QUESTION NO: 22

The U+ Corporation uses Pega software for mortgage underwriting tasks. After gathering the initial data, the mortgage case pauses and waits for a credit report. The credit report is a separate case, located in the same class group as the mortgage case, and typically resolves within 24 hours. After the credit report is complete, the mortgage case is routed to a specialist.

Which one of the following options is the best way to advance the mortgage case to the specialist?

A. Insert a credit report assignment step in the mortgage process containing an SLA to check the status of the credit report case and call the ResumeFile activity when the credit report case completes.

B. Include a split join shape in the mortgage process to include the credit report flow. Use the All option on the split join shape to ensure the credit report is complete before advancing to the step that routes to the specialist.

C. Add a wait shape to the mortgage process with a case dependency to advance the mortgage case to the next step when the credit report case is resolved.

D. Modify the credit report process to move the mortgage case to the specialist. The application routes the assignment to the specialist who has the least amount of work.

ANSWER: C

Explanation:

Add a wait shape to the mortgage process with a case dependency to advance the mortgage case to the next step when the credit report case is resolved. This is the appropriate Pega case-management pattern when one case must pause until a related, independently processed case reaches a resolved status. The Wait shape persists the mortgage case at the intended point in its life cycle and monitors the specified credit report case dependency. When that dependency is satisfied, Pega automatically resumes the paused flow and continues to the next configured step, which can route work to the underwriting specialist.

This approach keeps ownership of the mortgage workflow within the mortgage case while allowing the credit report case to complete through its own process. Because both cases are in the same class group, the mortgage case can establish the case dependency directly using the credit report case ID. It also avoids custom activities or periodic status-checking logic, while providing a declarative and maintainable design that reflects the real business condition: underwriting can proceed only after the required report is resolved. Pega documents the Wait shape as a mechanism for pausing a flow until a defined condition or case dependency is met; see Pega documentation: Wait shape and Pega Academy: Wait shape.

QUESTION NO: 23

You need to define a new LDAP authentication servlet because all of the standard LDAP servlets are

being used in authentication services

which two task do you perform to implement this requirement

A. Copy an existing LDAP servlet and rename it

B. Create a new mapping servlet and URL pattern

C. add a new URL pattern to the existing mapping servlet

D. rename an existing LDAP servlet

ANSWER: B

Explanation:

Create a new mapping servlet and URL pattern is correct because an LDAP authentication endpoint must have both a servlet definition and a unique URL mapping through which the application server routes authentication requests. When the available standard LDAP servlet mappings are already assigned to authentication services, defining another mapping provides an additional, independently addressable endpoint. The servlet mapping associates a servlet name with the LDAP authentication servlet implementation, while the URL pattern supplies the distinct request path used by the authentication service configuration. Together, these configuration elements allow the new authentication service to invoke the intended LDAP authentication flow without reusing an endpoint that is already in service. This is an application-server deployment configuration activity: after the mapping and URL pattern are added, the environment must use the newly defined servlet endpoint when configuring the relevant authentication service. Pega authentication services support external authentication mechanisms and are configured to direct requests through the applicable authentication processing endpoint. See Pega's Authentication services documentation and the Pega Academy authentication services topic for platform authentication configuration guidance.

QUESTION NO: 24

Which three actions do you recommend to address their performance issue and ensure optimal application performance in the future? (Choose Three.)

A. Dedicated a node to background processing.

B. Review application environment sizing.

C. Remove embedded dashboard reports from the manager portal.

D. Minimize the usage of robotic automations.

E. Review selection criteria of agent activities.

ANSWER: A B E

Explanation:

Dedicated a node to background processing., **Review application environment sizing.**, and **Review selection criteria of agent activities.** are the appropriate actions because they address the principal sources of sustained Pega Platform performance pressure: workload isolation, sufficient processing capacity, and efficient asynchronous work distribution. Assigning background processing to a dedicated node prevents agents, queue processors, and other non-interactive work from competing with requestors serving portal users. Reviewing environment sizing verifies that node count, CPU, memory, and capacity allocation match the actual transaction volume and workload profile, including expected growth. Reviewing agent activity selection criteria helps ensure that each background task retrieves only the appropriate work, uses selective queries, and avoids repeatedly scanning unnecessary instances. Together, these measures improve responsiveness for interactive users while making background execution more predictable and scalable. Pega node classification and dedicated node roles are described in Pega Academy: Node classification. Guidance on designing and operating background processing is available in Pega Academy: Background processing.

QUESTION NO: 25

A company is planning to build an application that can capture information about individuals in different roles, who work for different companies, in different parts of the world.

Future independent Pega Platform'* applications that can potentially use that same information are also being planned. Every application, including the first one, has a tight delivery schedule.

How do you design the data model?

A. Extend existing core Pega Platform classes such as Data-Party-Person, Data-Party-Com, and Data-Address-Postal, persisting instances of the extended classes in the PegaData schema.

B. Extend existing core Pega Platform classes such as PegaData-Contact, PegaData-Organization, and PegaData-Address, persisting instances of the extended classes in the CustomerData schema.

C. Create the minimum number of properties necessary to describe a person, organization, and address within the ORG-Data class. Share the ORG ruleset with other applications.

D. Define new Person, Organization, and Address Data classes that extend ORG-Data. Persist Person in the PegaData schema, and Organization and Address in the CustomerData schema.

ANSWER: A

Explanation:

Extend existing core Pega Platform classes such as Data-Party-Person, Data-Party-Com, and Data-Address-Postal, persisting instances of the extended classes in the PegaData schema. This approach uses Pega Platform’s established party and address model as the foundation for information that is inherently reusable: people, companies, and postal addresses. The supplied classes provide a consistent semantic model for these common business entities and allow the application to add organization-specific fields through inheritance rather than creating a parallel representation of the same concepts.

Persisting the extended class instances in the PegaData schema supports straightforward implementation using standard Pega Platform persistence while preserving a shared, platform-aligned model. Future applications can reuse the same class hierarchy and data definitions, reducing duplicate modeling effort and helping ensure that person, company, and address information retains the same meaning across applications. This is particularly suitable when the initial application has a short delivery timeline, because it starts from supported platform abstractions instead of requiring a separate enterprise data model to be designed and implemented first. Pega training discusses data modeling and reuse concepts at Pega Academy: Data model; the product documentation is available through Pega Documentation.