DUMPSARENA

Hitachi Vantara Certified Specialist - Pentaho Data Integration Implementation

Hitachi HCE-5920

Version Demo

Total Demo Questions: 10

Total Premium Questions: 103

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

QUESTION NO: 1

After monitoring a performance bottleneck in transformation you identified a user Defined Java Expression step as the problem.

Choose 2 answers

A. Replace the step with a Modified Java Script Value step

B. Increase the Change Number of Copies to Start for tie stop

C. Replace me step with a User Defined Java Class step

D. Split the step into multiple consecutive steps

ANSWER: C D

Explanation:

Replace me step with a User Defined Java Class step is appropriate because a User Defined Java Class step is designed for custom Java processing in a transformation and provides a more suitable implementation path when expression-based Java evaluation has been identified as the processing constraint. The custom class is initialized and compiled as transformation logic, allowing the developer to use explicit Java code, reusable helper methods, and efficient row-level processing patterns while retaining PDI's normal input/output row handling.

Split the step into multiple consecutive steps is also an effective tuning approach when the current expression contains independent or separable work. PDI transformations are executed as step pipelines: after the work is separated, downstream work can begin processing rows while upstream work continues. This overlap can improve throughput, particularly where a single complex step is preventing the transformation from using its pipeline efficiently. Splitting the logic also makes it possible to measure each stage independently with performance metrics and isolate any remaining costly calculation before further tuning.

Pentaho's transformation-step reference describes the User Defined Java Class step and its row-processing model: Pentaho User Defined Java Class documentation. General transformation performance and step-copy guidance is available in the Pentaho Data Integration Performance Tuning guide.

QUESTION NO: 2

You are using a Table Input step with parameterized SQL to retrieve rows for a selected date range.

Which two statements are correct about using parameters in the SQL statement? (Choose two.)

Choose 2 answers

A. Use question-mark placeholders for runtime data values.

B. Provide incoming fields in the order required by the SQL parameters.

C. Use a question-mark placeholder to supply a dynamic table name.

D. Use parameter placeholders only when the SQL returns no rows.

ANSWER: A B

Explanation:

A question-mark placeholder can be used for a runtime value in the SQL statement when the **Replace variables in script?** option is not being used for that value. The Table Input step binds incoming parameter fields to the placeholders through a prepared statement, which is appropriate for data values such as dates, numeric values, and codes.

The incoming parameter fields must be supplied to the Table Input step before it executes the query. For example, a Generate Rows step or an upstream input stream can provide the start-date and end-date fields that are bound to the SQL

placeholders. Table and column identifiers cannot be supplied as prepared-statement parameters; variable substitution is required when an identifier itself must be dynamic. See the Pentaho Table Input step documentation.

QUESTION NO: 3

You are developing the PDI solution for a customer. The customer requires that a set of SQL statements are executed on each connection to the database.

Which method will satisfy this requirement?

A. Use JNDI to specify the SQL statements.

B. Use the Advanced section in the Database Connection window to enter the SQL statements.

C. Use the Options section in the Database Connection windows and pass the SQL statements to the database as a parameter.

D. Use the Database Connection wizard and specify SQL statements.

ANSWER: B

Explanation:

Use the Advanced section in the Database Connection window to enter the SQL statements. Pentaho Data Integration database connections include an advanced connection setting specifically for SQL that must run immediately after PDI establishes a database session. This is appropriate when every newly created connection must initialize session-level behavior, such as setting a database session parameter, selecting a schema or role, configuring date or language behavior, or enabling database-specific execution settings. The SQL is associated with the reusable PDI database connection definition, so transformations and jobs that use that definition receive the required initialization when they open a connection.

This capability is distinct from ordinary connection properties: it is intended for executable initialization SQL, rather than merely supplying a driver or database parameter. Enter the statements in the post-connection SQL setting and ensure that they are valid for the selected database type and permitted for the account used by PDI. For implementation details on defining and maintaining PDI database connections, see the Pentaho Data Integration documentation and the Hitachi Vantara Data Integration documentation.

QUESTION NO: 4

A customer has an existing PDI job that calls a transformation. They want to execute the transformation through Spark on their Hadoop cluster.

Which change must be made to satisfy this requirement?

A. Change the Parameters of the transformation

B. Change the Run Configuration of the transformation

C. Change the Logging options of the transformation

D. Change the Arguments of the transformation

ANSWER: B

Explanation:

Change the Run Configuration of the transformation is correct because Pentaho Data Integration uses run configurations to determine the execution engine and target environment for a transformation. To run an existing transformation with Spark on a Hadoop cluster, the transformation must be associated with a Spark run configuration that contains the required Spark and cluster connection settings. This configuration tells PDI to submit the transformation for execution through the Spark engine rather than executing it locally with the default PDI engine. When a job invokes that transformation, the transformation's configured run environment controls how it is launched, allowing the job design and transformation logic to remain largely

unchanged. The Spark run configuration can define details such as the Spark connection, deployment mode, application settings, and Hadoop-related configuration needed to submit and run the workload on the cluster. This separation of transformation design from execution settings is a core PDI capability and makes it possible to use the same transformation in different runtime environments. See the Pentaho documentation on run configurations and the Spark run configuration documentation.

QUESTION NO: 5

You have a PDI input step that generates data within a transformation.

Which two statements are true about downstream steps in this scenario? (Choose two.)

Choose 2 answers

A. The steps will receive a stream of data from the input as soon as it is a available.

B. Only one step can receive data from the input step.

C. The steps will receive the data once the input step fully fetches it.

D. Multiple steps can receive data from the input step.

ANSWER: A D

Explanation:

In a Pentaho Data Integration transformation, steps are connected by row streams. When an input step generates rows, it places them into the transformation pipeline, allowing connected downstream steps to begin processing as rows become available rather than waiting for the input step to retrieve its entire result set. This streaming behavior supports pipeline parallelism: while one step continues reading or generating later rows, subsequent steps can transform, filter, write, or otherwise process earlier rows. Consequently, “The steps will receive a stream of data from the input as soon as it is a available.” is correct.

A single step can also have multiple outgoing hops. PDI sends the rows produced by that step to each connected downstream step, enabling branches in the transformation flow. This is useful when the same source rows must be processed in different ways or delivered to different targets without requiring separate input steps. Therefore, “Multiple steps can receive data from the input step.” is also correct. This behavior is fundamental to the transformation step-and-hop model described in the Pentaho Data Integration transformation steps documentation and the Hitachi Vantara Data Integration documentation.

QUESTION NO: 6

You are using a Transformation Executor step to invoke a child transformation for incoming rows from a parent transformation.

Which two configurations are used to exchange row data between the parent and child transformations? (Choose two.)

Choose 2 answers

A. Configure the input field mappings.

B. Configure the output field mappings.

C. Configure a Repository Explorer lock on the child transformation.

D. Configure a job hop from the parent transformation to the child transformation.

ANSWER: A B

Explanation:

The Transformation Executor step uses field mappings to send parent fields into the child transformation and to return selected child fields to the parent stream. The child transformation receives the supplied input through the Get rows from result step, then returns rows through the Copy rows to result step. Input mappings define the fields available to the child execution, and output mappings define the fields that are returned after the child transformation completes. This allows a reusable child transformation to process data while preserving the required parent-stream context. See the Transformation Executor documentation.

QUESTION NO: 7

A Big Data customer is experiencing failures on a Table input step when running a PDI transformation on AEL Spark against a large Oracle database.

What are two methods to resolve this issue? (Choose two.)

Choose 2 answers

A. Increase the maximum size of the message buffers for your AEL environment.

B. Load the data to HDFS before running the transform.

C. Add the Step ID to the Configuration File.

D. Increase the Spark driver memory configuration.

ANSWER: A B

Explanation:

Increasing the maximum size of the message buffers for the AEL environment is an appropriate remedy because AEL must transfer serialized PDI row data through its execution and messaging mechanisms. A large Oracle query can produce row volumes or individual row payloads that exceed the configured buffering capacity, resulting in failures before downstream processing can complete. Increasing the applicable buffer limit provides sufficient capacity for the AEL runtime to handle that data flow reliably.

Loading the data to HDFS before running the transformation is also a sound approach for large database extracts. It separates the Oracle extraction workload from the Spark transformation and places the source data in distributed storage that Spark can read in parallel. This avoids making a large JDBC Table Input extraction the bottleneck of an AEL Spark execution, improves scalability, and allows Spark to distribute processing across the cluster. Pentaho Data Integration is designed to integrate with Hadoop-compatible storage and execute transformations in big-data environments; Spark's guidance similarly describes distributed data-source processing and partition-aware reads. See the Pentaho Data Integration source and documentation project and the Apache Spark SQL data sources documentation.

QUESTION NO: 8

A client is developing a Web application to implement a wizard-like application used by many users. At several points in the workflow, the application needs to execute jobs and transformations that are stored in the Pentaho server. Execution will include user-specific parameters. Upon completion of the job or transformation, the Web application will continue to the next UI page.

What is the recommended approach to accomplish this task?

A. Use the Web service API to execute the jobs and transformations on the Pentaho server.

B. Use the scheduling tool of the OS to execute the job and transformation on the pentaho server.

C. Use the Kitchen and pan scripts to execute the jobs and transformations on the Pentaho server.

D. Use the Job and Transformation steps to execute the jobs and transformations on the Pentaho server.

ANSWER: C

Explanation:

Use the Kitchen and Pan scripts to execute the jobs and transformations on the Pentaho server. Kitchen is Pentaho Data Integration's command-line launcher for jobs, while Pan is the corresponding command-line launcher for transformations. Both utilities can execute content stored in a Pentaho Repository by supplying the repository connection and the job or transformation path. They also support runtime parameters, enabling the web application to pass values associated with the individual user or wizard session when it starts the execution.

This approach is appropriate when the application must know that processing has completed before advancing its interface. The web application can invoke the relevant command, provide the required parameter values, and wait for the process to return. The returned status and execution output can then be used by the application to determine whether it should proceed to the next page or present an execution error. This provides a direct, scriptable way to run repository-managed Data Integration content while preserving the synchronous workflow required by the wizard.

Hitachi's Pentaho documentation describes Kitchen and Pan as the command-line tools for running jobs and transformations: Pentaho Data Integration command-line tools. Parameter handling is documented in the Pentaho Data Integration parameters documentation.

QUESTION NO: 9

You have a PDI job with an Oozie JobExecutor entry and you want this job entry to finish before the nextjob entry starts.

How do you accomplish this task?

A. Use a Simple Evaluationentry

B. Add the timeout property in the Advanced Options pane.

C. Use a Wait For entry

D. Set the Enable Blocking option

ANSWER: D

Explanation:

Set the **Enable Blocking** option on the Oozie JobExecutor entry. Blocking causes PDI to submit the Oozie workflow and then wait for Oozie to report a terminal result before the PDI job continues along its outgoing hop. Consequently, the subsequent job entry does not begin while the submitted Oozie job is still running. This is the appropriate setting when a later PDI task depends on the workflow's completion, output, or success status. Without blocking, job submission can return control to PDI immediately, allowing the following entry to run independently of the remote workflow's state. The Oozie JobExecutor implementation includes a blocking configuration and uses it to control whether execution waits while the Oozie job is monitored. Pentaho's PDI documentation describes job entries as the control-flow components of a PDI job, while the Oozie integration source identifies the JobExecutor entry and its execution configuration. See the Pentaho Data Integration documentation and the Pentaho Kettle source repository.

QUESTION NO: 10

What are Job checkpoints?

A. Error handing hops that manage e-mail notifications.

B. Points in a job that have finished successfully and are automatically skipped after a previously failed run

C. Points in a job that log the step metrics during certain intervals

D. Points in a job that sends a heartbeat request to the Pentaho server to ensure connectivity.

ANSWER: B

Explanation:

Job checkpoints are designated points that enable a Pentaho Data Integration job to resume processing after a failure rather than repeating work that has already completed. When a job reaches a checkpoint successfully, PDI records that state in the job's logging information. If the job is later restarted following a failure, entries completed before the most recently successful checkpoint can be skipped, and execution resumes from that checkpoint. This is particularly useful for long-running, multi-stage jobs, where rerunning completed stages could be costly, duplicate data, or cause unwanted side effects. Checkpoints are configured through the job's settings and require an appropriate logging configuration so PDI can identify the last successfully reached checkpoint for a particular job execution. The checkpoint mechanism therefore supports reliable restartability and recovery by treating successfully completed portions of the job as already processed. Pentaho's documentation describes the Checkpoint job entry and its role in marking a recoverable point in job execution; see the Pentaho Checkpoint job-entry documentation. Additional job design and execution guidance is available in the Pentaho Data Integration documentation.