DUMPSARENA

Operationalizing Machine Learning and Generative AI Solutions

Microsoft AI-300

Version Demo

Total Demo Questions: 11

Total Premium Questions: 110

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Implement machine learning operations (MLOps)	52
Topic 2, Implement generative AI operations (GenAIOps)	58
Total	110

QUESTION NO: 1

A company plans to deploy a generative AI application that handles customer requests. The company must reduce the risk of prompt injection and must prevent harmful generated content from being returned to users.

You need to implement appropriate safety controls. Which two actions should you perform? Each correct answer presents part of the solution. Choose two. NOTE: Each correct selection is worth one point.

A. Configure prompt shield protections.

B. Increase the model temperature.

C. Configure content filters.

D. Disable trace collection.

ANSWER: A C

Explanation:

Configure prompt shield protections because prompt shields are designed to detect prompt injection attacks, including malicious instructions that can be embedded in user input or retrieved documents. This helps reduce the likelihood that untrusted content overrides the application instructions.

Configure content filters because content filtering evaluates prompts and model responses for harmful content categories. The application can block or otherwise handle content that violates the configured safety policy before it reaches users.

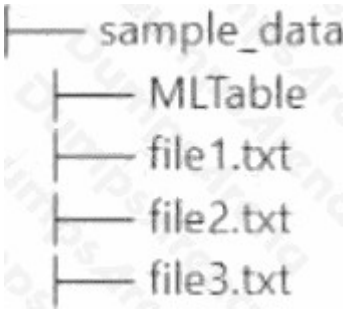
Increasing temperature generally makes responses less deterministic and does not provide a safety control. Disabling trace collection reduces operational visibility and does not mitigate prompt injection or harmful content. For more information, see Prompt Shields in Azure AI Content Safety and Content filtering in Azure OpenAI.

QUESTION NO: 2 - (SIMULATION)

You manage an Azure Machine Learning workspace named workspace1 by using the Python SDK v2.

The default datastore of workspace1 contains a folder named sample_data.

The folder structure contains the following content:

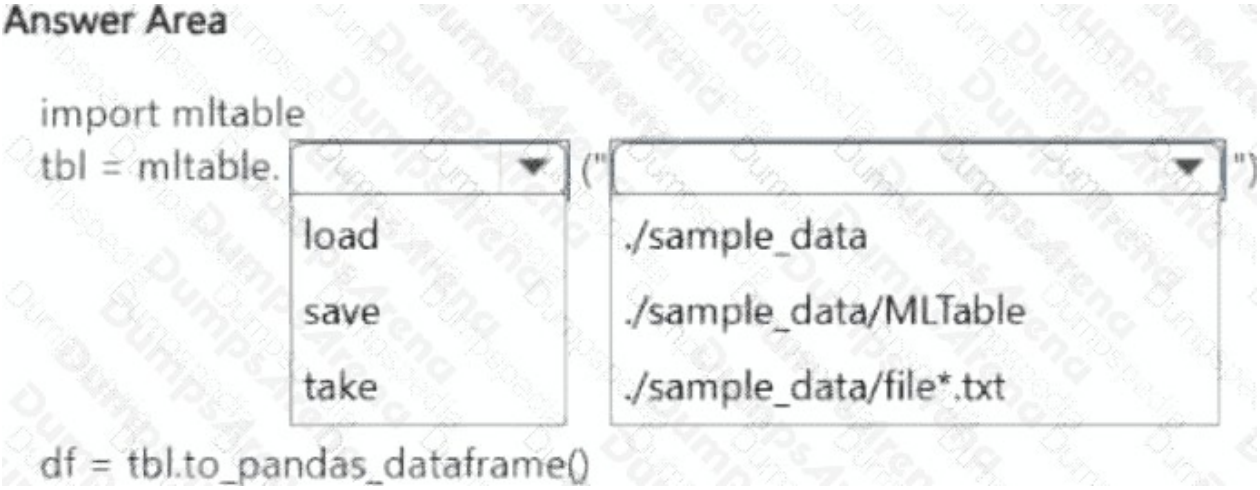


You write Python SDK v2 code to materialize the data from the files in the sample_data folder into a Pandas data frame.

You need to complete the Python SDK v2 code to use the MLTable folder as the materialization blueprint.

How should you complete the code? To answer, select the appropriate options in the answer area . NOTE: Each correct selection is worth one point

Answer Area



ANSWER: See the explanation for the answer

Explanation:

Select:

`load`
`./sample_data`

The completed code is:

```
import mltable

tbl = mltable.load("./sample_data")
df = tbl.to_pandas_dataframe()
```

`mltable.load()` must receive the path to the directory that contains the MLTable definition file. Here, the `MLTable` file is in the `sample_data` directory, so the correct path is `./sample_data`, not the MLTable file itself or the individual text files.

QUESTION NO: 3

An organization uses Azure Machine Learning to run a pipeline after new training data is approved. The pipeline must run automatically without requiring a data scientist to manually submit a job.

You need to automate the recurring pipeline execution.

What should you create?

A. Azure Machine Learning schedule

B. Managed online endpoint

C. Compute instance

D. MLflow experiment

ANSWER: A

Explanation:

An **Azure Machine Learning schedule** is the appropriate solution because it can trigger a pipeline job on a recurring time-based schedule. The schedule stores the job definition and execution frequency, enabling automated, repeatable pipeline runs.

A compute cluster provides compute capacity but does not initiate pipeline jobs. An online endpoint provides inference, and an experiment records job runs rather than scheduling them. Azure Machine Learning schedules support recurring execution of pipeline jobs for operational machine learning workflows. See Schedule machine learning jobs.

QUESTION NO: 4

A team develops multiple AI applications in Microsoft Foundry that rely on shared prompt templates. The team requires a centralized way to track, version, and reuse prompt content across projects.

You need to recommend a solution to track and reuse prompt content.

Which approach should you recommend?

A. Store prompts as versioned files in a Git repository.

B. Register prompts as datasets in the Azure Machine Learning workspace.

C. Embed prompts directly in application configuration files.

D. Persist prompts in Azure Blob Storage with folder-level organization.

ANSWER: A

Explanation:

Store prompts as versioned files in a Git repository. Prompt templates are application artifacts: they influence model behavior, require review before release, and commonly change as an application is tested and refined. A shared Git repository provides a central location from which multiple Microsoft Foundry applications and teams can consume the same prompt assets. It also supplies a complete, immutable change history, meaningful diffs between prompt revisions, branches for parallel development, and pull-request workflows for peer review and approval. Teams can tag approved prompt versions and link a particular application deployment or experiment to the exact prompt revision it used, improving reproducibility and auditability. Git-based storage also supports treating prompts as code in CI/CD processes: automated tests or evaluations can validate a changed prompt before it is merged and promoted for shared use. Microsoft documents Git repositories as a collaboration and version-control mechanism for source files, with commits recording changes and branches enabling isolated work. Microsoft's DevOps guidance further supports source control, repeatable delivery, and collaboration for solution artifacts. See Azure Repos Git documentation and Microsoft DevOps guidance.

QUESTION NO: 5

An Azure Machine Learning workspace contains multiple registered versions of a model that is used in production.

An older model version must no longer be deployable, but it must remain available for compliance review and potential rollback. You need to change the state of the model version to meet the requirements.

What should you do?

A. Archive the training dataset for the model version.

B. Delete the model version.

C. Archive the model version.

D. Unregister the model version.

ANSWER: C

Explanation:

Archive the model version. Azure Machine Learning model assets support lifecycle management through an archive state. Archiving changes the version from an active asset to an archived asset, preventing it from being selected for deployment

while retaining the registered model, its files, metadata, tags, and lineage in the workspace. This provides the required operational safeguard: routine production deployment processes cannot use the older version after it has been archived.

Because the version remains registered rather than being removed, authorized reviewers can retain it as evidence for compliance, audit, reproducibility, and governance activities. The archive operation is also reversible. If an investigation or rollback requires that model version again, it can be restored (unarchived) to make it active and eligible for use. This preserves a controlled rollback path without keeping an obsolete model version deployable during normal operations.

Microsoft documents model asset management, including the ability to archive and restore registered model versions, in Manage models in Azure Machine Learning. The available Azure Machine Learning model lifecycle commands are also documented in the Azure CLI model command reference.

QUESTION NO: 6 - (DRAG DROP)

A team deploys a classification model to production and monitors performance and data changes.

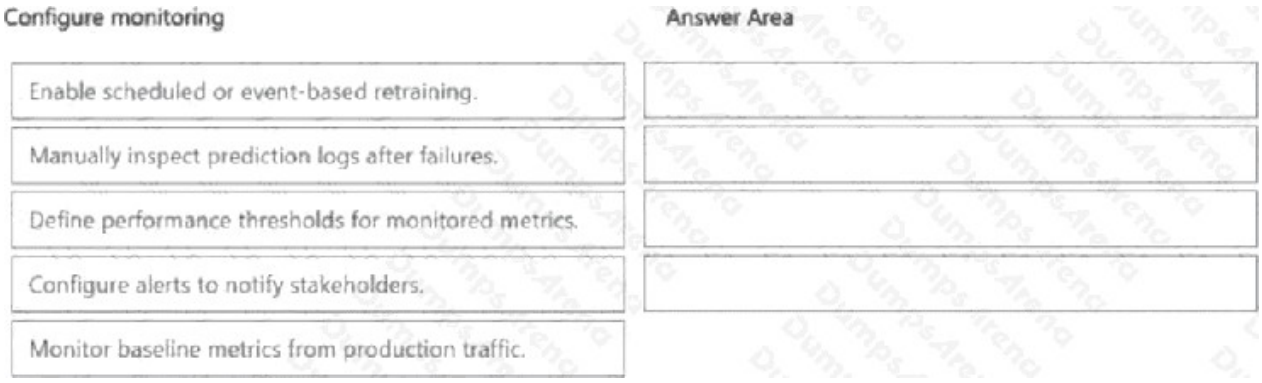
The team wants to ensure that significant drops in prediction accuracy automatically trigger the following:

Stakeholders must be notified of the drops.

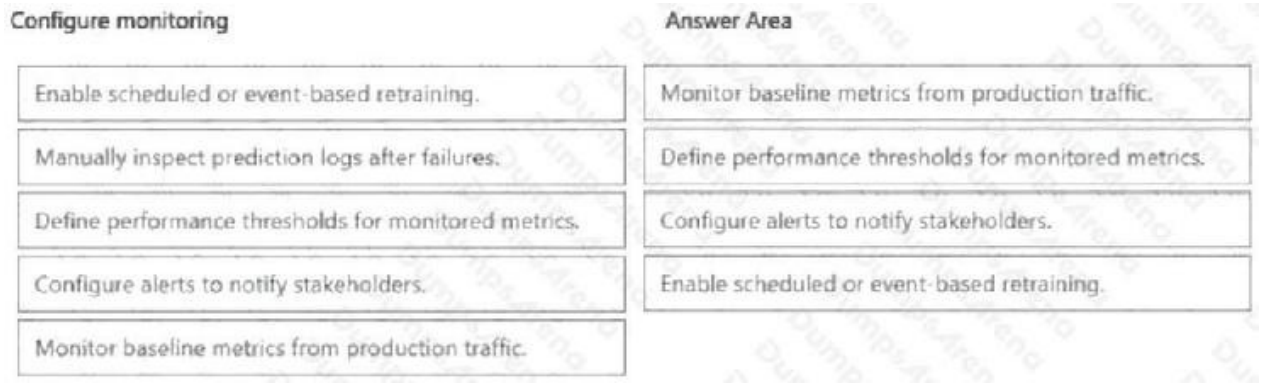
Retraining must be initiated when thresholds are exceeded

You need to configure monitoring to meet the requirements.

Which four actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



ANSWER:



Explanation:

Production model monitoring starts by monitoring baseline metrics from production traffic. A deployed classification model needs actual inference and performance telemetry so the team can continuously observe the metric that represents quality, such as prediction accuracy. This creates the operational monitoring signal on which the automated process is based. Azure Machine Learning model monitoring supports collecting production data and calculating model-quality or data-change signals over a configured monitoring schedule. See Monitor model performance in Azure Machine Learning.

After the model metric is being monitored, define the performance threshold that represents unacceptable degradation. The threshold converts the business requirement of a “significant drop” into an objective, measurable condition. Azure Monitor can evaluate a metric or monitoring result against that condition through an alert rule. When the observed accuracy crosses the configured limit, the alert is activated. The alert is then configured with an action group, which delivers the required notification to the relevant stakeholders. Action groups support operational notification channels and automated actions; details are documented in Azure Monitor action groups.

Finally, enable event-based retraining. The threshold breach and resulting alert provide the event that starts the remediation workflow, such as invoking a published Azure Machine Learning pipeline or another automation endpoint that launches retraining. This preserves an automated lifecycle: observe production behavior, evaluate it against the defined quality requirement, notify the people responsible, and initiate the retraining process when the condition is met. The final sequence therefore directly fulfills both required outcomes: stakeholder notification and automatic retraining after a material accuracy decline.

QUESTION NO: 7 - (HOTSPOT)

HOTSPOT -

A company is creating an internal tool that summarizes long meeting transcripts and extracts action items. The model must:

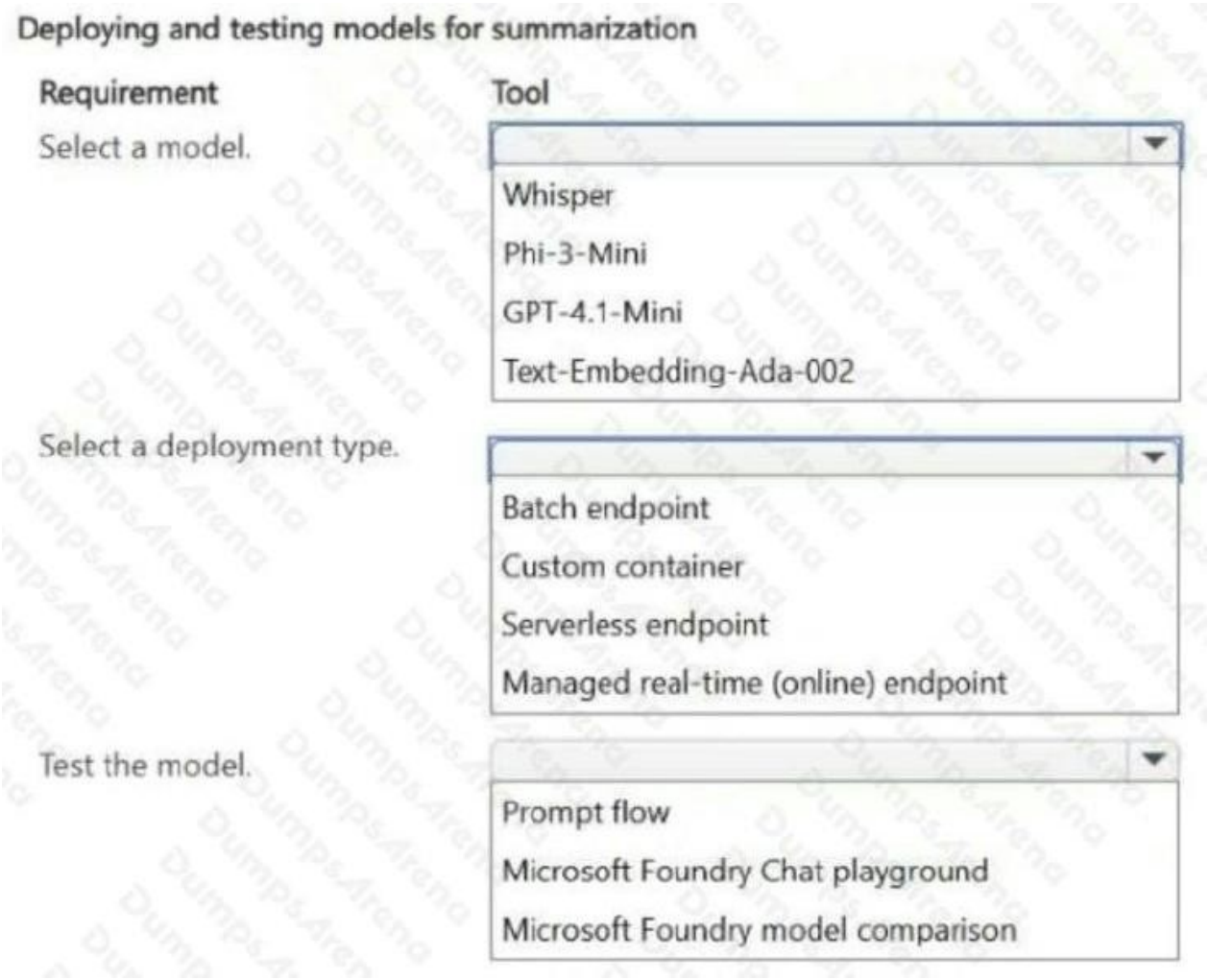
Process text inputs up to 200k tokens long.

Generate concise summaries in seconds.

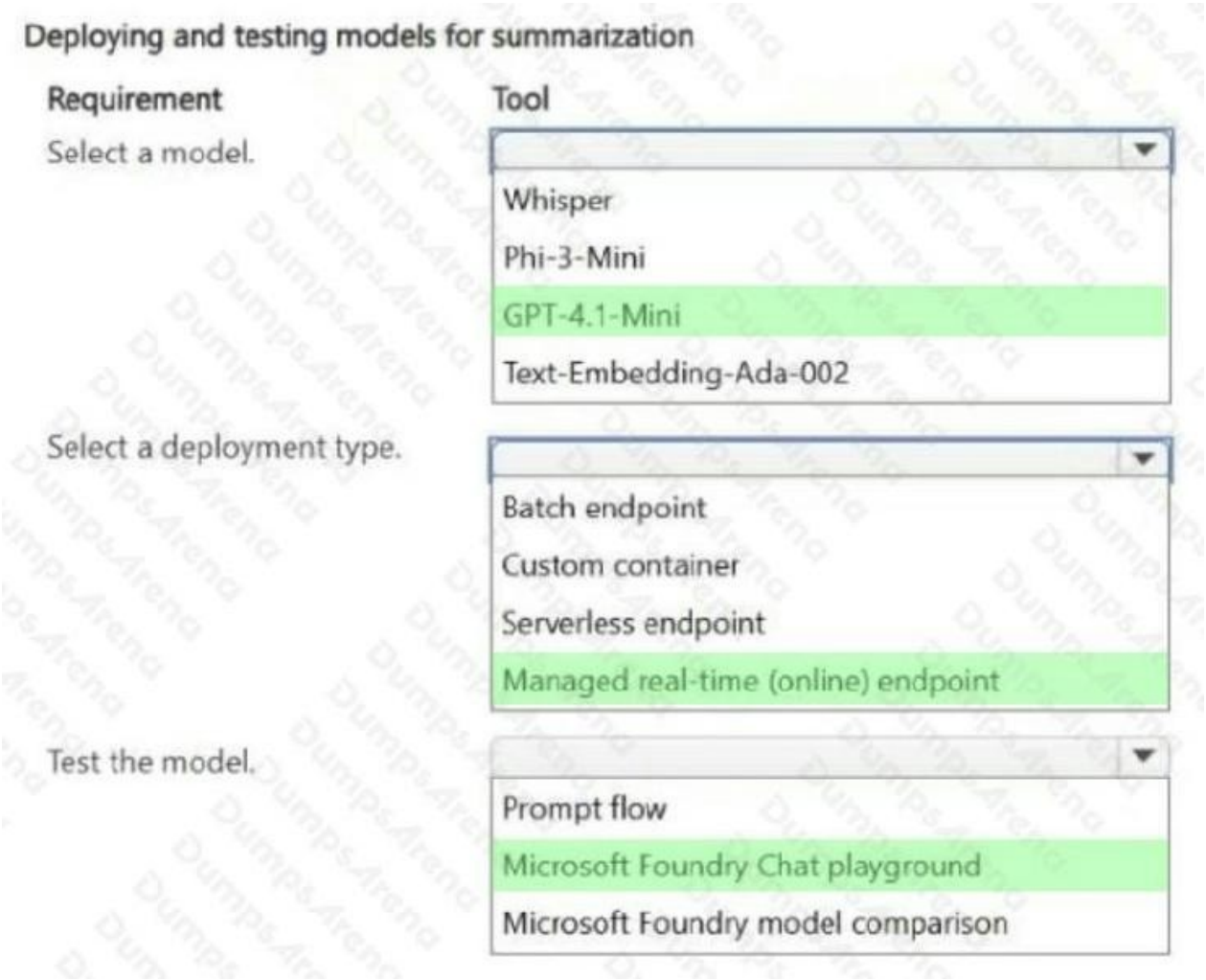
Support interactive testing before integration into the app.

You need to select, deploy, and test a model that supports summarization with low latency.

How should you complete the configuration plan? To answer, select the appropriate options in the answer area.



ANSWER:



Explanation:

The correct configuration uses **GPT-4.1-Mini**, a **Managed real-time (online) endpoint**, and the **Microsoft Foundry Chat playground**. GPT-4.1-Mini is appropriate because this is a text-generation workload: the tool must read a very long meeting transcript, identify the important information, and return a concise natural-language summary with action items. GPT-4.1 models support long-context processing, and GPT-4.1 Mini is designed to provide responsive, efficient generation for workloads where speed and cost efficiency matter. Its context window is substantially larger than the required 200,000-token input, allowing the transcript and a well-designed summarization instruction to be supplied in the same request. Microsoft documents the GPT-4.1 family's long-context capabilities at Azure OpenAI GPT-4.1 series models.

A managed real-time online endpoint is the right deployment approach because the internal tool needs an immediate request/response experience. The application can submit a transcript when a user requests a summary, receive generated text promptly, and display the result without waiting for an offline processing cycle. Online endpoints are intended for low-latency inference and synchronous application integration; their purpose and behavior are described in Azure Machine Learning online endpoints.

Before integrating the model call into the internal application, the Microsoft Foundry Chat playground provides the suitable interactive environment. A developer can test instructions, vary summarization formats, include action-item extraction rules, and inspect responses directly in a chat-style interface. This allows the team to validate output quality and refine prompts using representative transcript content before implementing the production workflow. Microsoft describes this interactive prompt-testing capability in the Microsoft Foundry chat playground documentation.

QUESTION NO: 8 - (HOTSPOT)

You are reviewing a dataset that will be used for an advanced fine-tuning job in Microsoft Foundry.

The fine-tuning job uses preference comparison data.

You review the following dataset excerpt.

```
Training record 1:
01 {"input":[{"role":"system","content":"You are a policy assistant."},
02          {"role":"user","content":"Summarize Policy X for managers."}],
03  "preferred_output":[{"role":"assistant","content":"Policy X applies to all managers and outlines approval
requirements."}],
04  "non_preferred_output":[{"role":"assistant","content":"Policy X applies to all managers and outlines approval
requirements."}]}

Training record 2:
01
02 {"input":[{"role":"system","content":"You are a policy assistant."},
03          {"role":"user","content":"What is the retention period?"}],
04  "preferred_output":[{"role":"assistant","content":"The retention period is seven years."}],
05  "non_preferred_output":[{"role":"assistant","content":"The retention period is five years."}]}
```

For each of the following statements, select Yes if the statement is true. Otherwise, select No . NOTE: Each correct selection is worth one point.

Preference-based fine-tuning dataset validation

Statement	Yes	No
The first record provides a usable preference signal for preference-based fine-tuning.	○	○
The second record provides a usable preference signal for preference-based fine-tuning.	○	○
Each preferred and non-preferred output in the dataset contains a valid assistant message as required for direct preference optimization.	○	○
Both records must contain different preferred and non-preferred outputs to produce usable training signals.	○	○

ANSWER:

Preference-based fine-tuning dataset validation

Statement	Yes	No
The first record provides a usable preference signal for preference-based fine-tuning.	○	●
The second record provides a usable preference signal for preference-based fine-tuning.	●	○
Each preferred and non-preferred output in the dataset contains a valid assistant message as required for direct preference optimization.	●	○
Both records must contain different preferred and non-preferred outputs to produce usable training signals.	●	○

Explanation:

Preference-based fine-tuning trains a model from comparisons rather than from a single target completion. Each example supplies the conversation context together with one assistant response that should be preferred and another that should receive a lower ranking. In the displayed records, the values in both `preferred_output` and `non_preferred_output` are correctly represented as message arrays, and each array contains a message whose role is `assistant`. This is the required message structure for the response portions of a preference comparison record. Consequently, the statement that each preferred and non-preferred output shown contains a valid assistant message is true.

A usable comparison also requires a meaningful preference distinction. The first record repeats precisely the same assistant content in its preferred and non-preferred outputs, so it cannot teach the model a ranking. The second record has competing retention periods and explicitly marks the seven-year response as preferred, which supplies a usable ranking signal. For a collection of records to provide usable preference signals, every individual comparison must similarly distinguish its preferred

response from its non-preferred response. Microsoft Foundry documents preference optimization as training on preferred and non-preferred responses, where the model learns to favor the selected response for the same input context. See Direct Preference Optimization fine-tuning in Microsoft Foundry and Fine-tuning in Microsoft Foundry.

QUESTION NO: 9 - (DRAG DROP)

A team manages prompts that are used by a generative AI application built on Microsoft Foundry. Multiple developers contribute prompt updates, and changes must be reviewed and tracked over time.

The team requires that:

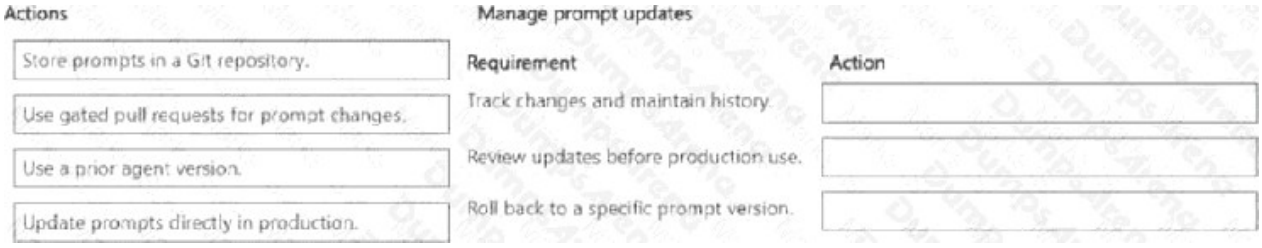
Prompt changes are reviewed before being applied to the version in production.

Previous prompt versions can be restored if issues occur.

Prompt updates follow the same governance practices as the application code.

You need to implement a controlled process for managing and updating prompts in production.

How should you manage prompt updates to meet the requirements? To answer, move the appropriate actions to the correct requirements. You may use each action once, more than once, or not at all. You may need to move the split bar between panes or scroll to view content . NOTE: Each correct selection is worth one point.



ANSWER:

Explanation:

A controlled prompt-update process should treat prompts as versioned engineering artifacts. Store prompts in a Git repository to track changes and maintain history. Each commit captures the prompt content at a particular point in time, together with the associated author, timestamp, and change record. This gives the team a dependable historical record of prompt evolution and allows the prompt assets to be governed alongside the application source. Git-based repositories also support the same branch policies, access controls, audit practices, and automated delivery processes used for production code. Microsoft's guidance for Git repositories and source control is available in Azure Repos Git documentation.

Use gated pull requests for prompt changes to review updates before production use. Developers can work on proposed prompt revisions in branches, then submit those changes for review before they are merged into the branch or release path that feeds production. Required reviewers and validation policies create an explicit approval checkpoint, ensuring that the production prompt is not updated until the team has reviewed the proposed change. This directly aligns prompt changes with a standard code-review workflow. Details on enforcing reviews and completion policies are provided in Azure DevOps pull request documentation.

Use a prior agent version to roll back to a specific prompt version. When a known-good version is retained, the team can restore that prior version quickly if a newly deployed prompt introduces incorrect behavior, quality regressions, or operational issues. Versioned deployment makes recovery deliberate and traceable because the restored state is a specific previously established version rather than an ad hoc rewrite. Together, Git history, gated review, and prior-version restoration provide the required governed lifecycle for prompts in production.

QUESTION NO: 10

A data science team trains a classification model that predicts loan approval outcomes.

Before registering the model, the team must ensure the following:

Predictions must not disproportionately impact protected groups.

Prediction errors can be evaluated across different data segments.

You need to assess whether the model meets Responsible AI expectations.

Which two approaches should you use? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point. Choose two .

A. Analyze error rates across the global cohort.

B. Measure endpoint latency under load.

C. Validate inference schema compatibility.

D. Evaluate feature importance for prediction transparency.

E. Analyze error rates across defined demographic cohorts.

ANSWER: D E

Explanation:

Evaluate feature importance for prediction transparency and analyze error rates across defined demographic cohorts. Together, these approaches address key Responsible AI requirements for a high-impact classification scenario such as loan approval. Feature importance and model interpretability help the team understand which input features have the greatest influence on predictions. This provides transparency into model behavior, supports review of potentially sensitive or proxy features, and enables stakeholders to investigate individual and aggregate prediction drivers.

Error analysis across defined demographic cohorts enables the team to slice model performance by meaningful data segments, including protected groups where appropriate and legally permitted. The team can compare error patterns and performance metrics across those cohorts to identify whether the model's mistakes, such as incorrect approvals or denials, are disproportionately experienced by a particular group. This directly supports fairness assessment before the model is registered or deployed.

Azure Machine Learning's Responsible AI dashboard brings model interpretability and error analysis capabilities together for model assessment. It allows teams to inspect feature importance and identify cohorts with concentrated errors, helping them make evidence-based decisions about whether further mitigation is needed. See Responsible AI dashboard in Azure Machine Learning and Responsible AI insights SDK.

QUESTION NO: 11

A Retrieval-Augmented Generation (RAG) solution returns incomplete answers because relevant content is inconsistently retrieved from the knowledge source.

You need to improve RAG accuracy without changing the embedding model currently in use. You need to achieve this goal while minimizing operational costs.

Which two actions should you perform? Each correct answer presents part of the solution. Choose two. NOTE: Each correct selection is worth one point.

A. Tune chunk size and overlap to match content structure.

B. Implement an optimized re-ranker.

C. Increase token limits for all requests.

D. Optimize the length of embedding vectors.

ANSWER: A B

Explanation:

Tune chunk size and overlap to match content structure improves retrieval recall without requiring a new embedding model. RAG retrieval depends on the passages that are created and indexed. Chunks that separate headings from their supporting text, split a logical procedure across boundaries, or contain too much unrelated content can prevent relevant evidence from being returned. Choosing chunk boundaries that preserve semantic units and applying a measured overlap makes related context available in adjacent chunks, increasing the likelihood that the relevant passage is retrieved. This is an indexing and content-preparation improvement, so it can deliver better answers without embedding-model replacement or increased generation-token use.

Implement an optimized re-ranker improves the quality of the evidence passed to the generative model. Initial vector retrieval efficiently produces a candidate set, while reranking applies a more precise relevance evaluation to order that limited set. The highest-value passages are therefore placed first and can be selected for the model context, improving answer completeness and grounding. Operational cost remains controlled by retrieving a limited number of candidates and reranking only those candidates rather than increasing context windows or processing every document. Azure AI Search supports semantic ranking to improve relevance over initial search results, and Microsoft RAG guidance emphasizes retrieval quality and grounding data preparation. See Semantic ranking in Azure AI Search and Azure OpenAI on your data.