

Microsoft

Exam Questions AI-200

Developing AI Cloud Solutions on Azure



NEW QUESTION 1

You are investigating high latency in an AI search application that processes millions of requests daily. Telemetry is stored in Azure Monitor Logs.

You must create a KQL query that correlates information from the AppRequests table and the AppDependencies table. The query must meet the following requirements:

Include only data from the last 24 hours.

Filter for failed requests only.

Calculate the average duration of dependencies, grouped by operation.

The query must follow best practice to optimize the performance by minimizing the initial data scan.

You need to create the query.

In which order should perform the actions? To answer, move all actions from the list of actions to the answer area and arrange them in the correct order.

Actions

Answer Area

⋮ Summarize average dependency duration by operation.

⋮ Join the dependencies table.

⋮ Select the requests table.

⋮ Filter the failed requests.

⋮ Apply a time filter.

A. Mastered

B. Not Mastered

Answer: A

Explanation:

Verified Answer:1) Select the requests table; 2) apply the 24-hour time filter; 3) filter failed requests; 4) join the dependencies table; 5) summarize average dependency duration by operation.

Detailed Explanation:KQL performs best when high-selectivity filters are applied as early as possible, especially the time predicate that limits the amount of data scanned. Starting from AppRequests, restricting to the last 24 hours, and then filtering failures reduces the left-side dataset before the join. The dependency table is then correlated with those requests, and aggregation is performed last to calculate average dependency duration by operation. Joining or summarizing before the time and failure filters would process more data than necessary.

Study Guide Alignment:Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.

Official Microsoft Learn References:AI-200 Study Guide | Optimize log queries in Azure Monitor

NEW QUESTION 2

You develop an ASP.NET Core app that uses Azure App Configuration. You also create an App Configuration containing 100 settings. The app must meet the following requirements:

- Ensure the consistency of all configuration data when changes to individual settings occur.
- Handle configuration data changes dynamically without causing the application to restart.
- Reduce the overall number of requests made to App Configuration APIs

You must implement dynamic configuration updates in the app.

What are two ways to achieve this goal? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Create and configure Azure Key Vault

B. Implement the Azure Key Vault configuration provider

C. Register all keys in the App Configuration store Set the refreshAll parameter of the Register method to false.

D. Create and register a sentinel key in the App Configuration stor

E. Set the refreshAll parameter of the Register method to true

F. Decrease the App Configuration cache expiration from the default value.

G. Increase the App Configuration cache expiration from the default value.

H. Create and implement environment variables for each App Configuration store setting

Answer: CE

Explanation:

Detailed Explanation:A sentinel key allows the application to check one value and refresh all related configuration only when that sentinel changes, preserving a consistent set of settings instead of refreshing 100 keys independently. Increasing the refresh/cache interval reduces how frequently the application contacts App Configuration APIs. Together these measures support dynamic refresh without application restart while controlling request volume. Registering every key separately or decreasing the cache interval would increase API traffic, and Key Vault/environment variables do not solve coordinated App Configuration refresh.

Study Guide Alignment:Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.

Official Microsoft Learn References:AI-200 Study Guide | Dynamic configuration in ASP.NET Core

NEW QUESTION 3

You need to implement trace correlation according to the business requirements.

Which three 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.

NOTE: More than one order of answer choices is correct. You will receive credit for any of the correct orders you select.

Actions

⋮ Redeploy the instrumented services.

⋮ Configure a trace exporter in the OpenTelemetry SDK.

⋮ Instrument the application code by using the OpenTelemetry SDK.

⋮ Call TelemetryClient.TrackEvent() within service methods.

⋮ Implement a view in the OpenTelemetry SDK.

Answer Area

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:
Trace correlation requires the application to create OpenTelemetry spans and an exporter to send those spans to Azure Monitor. Instrumentation must therefore be present before the updated service is deployed. The previous Application Insights SDK instrumentation does not satisfy the case-study requirement that all tracing use OpenTelemetry. Configuring an OpenTelemetry view is unrelated to basic distributed-trace export, and calling TrackEvent is an Application Insights SDK pattern rather than the required OpenTelemetry approach.
Study Guide Alignment: Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.
Official Microsoft Learn References: AI-200 Study Guide | Enable Azure Monitor OpenTelemetry

NEW QUESTION 4

You are developing an AI application that retrieves database credentials from Key Vault by using the Azure SDK for Python. The application must use managed identity for authentication.

You review the following code segment that retrieves a secret from Key Vault:

```
01 from azure.identity import DefaultAzureCredential
02 from azure.keyvault.secrets import SecretClient
03
04 credential = DefaultAzureCredential()
05 client = SecretClient(
06     vault_url="https://vault.vault.azure.net/",
07     credential=credential
08 )
09
10 secret = client.get_secret("dbPassword", version="123")
```

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

Statement	Yes	No
The code retrieves a specific version of the dbPassword secret.	☐	☐
The code authenticates to Key Vault without storing client secrets in the application when deployed to an Azure hosted environment that has a managed identity enabled.	☐	☐
If dbPassword is rotated and a new version is created, subsequent calls to this code will retrieve the new version.	☐	☐

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

The `get_secret` call supplies an explicit version value, so it retrieves that particular version of `dbPassword`. `DefaultAzureCredential` can authenticate to Key Vault by using the Azure-hosted application's managed identity without embedding a client secret in the application. Because the code requests a fixed version, later secret rotation creates a newer version but subsequent executions of this exact code continue to request the specified old version. To follow the latest version automatically, the version parameter must be omitted.

Study Guide Alignment: Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.
Official Microsoft Learn References: AI-200 Study Guide | Managed identities for Azure resources | Use Key Vault references for App Service and Functions

NEW QUESTION 5

You deploy an API to Azure Container Apps.

The solution must provide the following functionality:

- Support the concurrent activation of multiple application versions.
- Allocate a specific percentage of incoming requests to a secondary version

You need to configure revision behavior.

Which configurations should you use? To answer, move the appropriate configurations to the correct requirements. You may use each configuration 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.



- A. Mastered
B. Not Mastered

Answer: A

Explanation:

Azure Container Apps supports side-by-side application versions through multiple revision mode. Once multiple revisions can be active, ingress traffic can be distributed by percentage across those revisions. This is the required combination for canary or controlled rollout scenarios. Single revision mode deactivates the previous revision as a new one becomes active, while traffic splitting is the specific feature that assigns a percentage of incoming requests to each active revision. Study Guide Alignment: Containerized Azure workloads: registry builds, App Service containers, Container Apps revision/scaling behavior, and AKS deployment choices.

Official Microsoft Learn References: AI-200 Study Guide | Azure Container Apps revisions

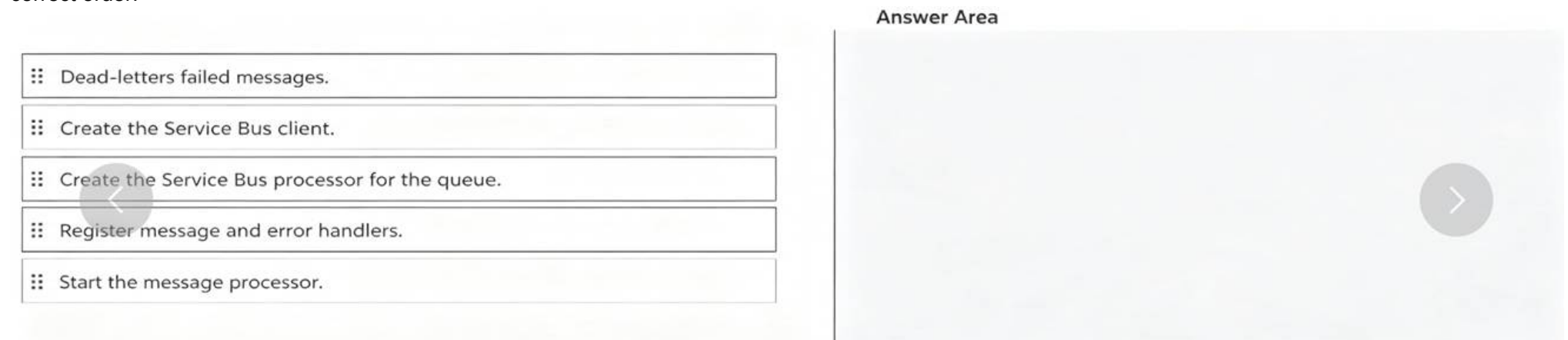
NEW QUESTION 6

You are provisioning and configuring a Service Bus processor for AI batch jobs.

The processor must connect to an existing queue, register handlers for message and error processing, and then begin receiving messages.

You need to provision and configure the Service Bus processor for message and error handling.

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.



- A. Mastered
B. Not Mastered

Answer: A

Explanation:

The processor depends on a namespace client/connection, so the client is created first. Next, the queue-specific processor or receiver is created from that client. Message and error callbacks must be registered before processing starts so incoming messages and failures have defined handlers. Starting the processor is therefore the final step. Dead-lettering failed messages is an optional application settlement decision and is not a prerequisite for constructing and starting the processor.

Study Guide Alignment: Azure service integration: Service Bus, Event Grid, Azure Functions triggers /bindings, and event-driven processing.

Official Microsoft Learn References: AI-200 Study Guide | Service Bus message transfers, locks, and settlement

NEW QUESTION 7

A Python API running in ACA must send distributed traces to Azure Monitor. The API creates spans. However, no traces appear in Azure Monitor.

You need to configure the OpenTelemetry SDK pipeline to export traces to Azure Monitor.

What should you do? 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.

Actions

Enable log sampling.

Call `tracer.start_as_current_span()`

Initialize the application's TracerProvider for tracing.

Configure a span processor to send spans to the exporter.

Create the Azure Monitor component that sends trace data.

OpenTelemetry configuration mapping

Requirement

Register a global tracer provider.

Export traces to Azure Monitor.

Connect the exporter to the provider.

Generate spans in the application code.

Action

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:
OpenTelemetry tracing is a pipeline: the application obtains a tracer from a configured provider, creates spans, passes ended spans through a span processor, and exports them using the Azure Monitor exporter. Creating spans alone is insufficient; without the provider/processor/exporter chain, no telemetry reaches Azure Monitor. Log sampling and legacy Application Insights TrackEvent calls are unrelated to the required OpenTelemetry trace-export path.
Study Guide Alignment: Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.
Official Microsoft Learn References: AI-200 Study Guide | Enable Azure Monitor OpenTelemetry

NEW QUESTION 8

You need to redesign the configuration approach so that:
configuration supports dynamic refresh;
secrets are stored outside App Configuration;
the application can securely access both App Configuration and Key Vault.
You need to enable secure dynamic configuration management for the platform.
Which three actions should you perform? Each correct answer presents part of the solution. Choose three.
NOTE: Each correct selection is worth one point.

- A. Store configuration values in environment variables.
- B. Use a service principal secret for both App Configuration and Key Vault access.
- C. Store API keys in Key Vault.
- D. Allow configuration updates with a polling interval.
- E. Use managed identity for both App Configuration and Key Vault access.

Answer: CDE

Explanation:
Explanation
The original answer omitted the secret-storage requirement. API keys and other secrets should be held in Key Vault, while App Configuration should retain non-secret configuration and Key Vault references. Dynamic refresh is implemented through the App Configuration refresh/polling mechanism rather than requiring an application restart. Managed identity provides credential-free access to both services in Azure. Environment variables do not provide centralized dynamic refresh, and a stored service-principal secret creates exactly the credential-management problem the design is intended to eliminate.
Study Guide Alignment: Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.
Official Microsoft Learn References: AI-200 Study Guide | Dynamic configuration in ASP.NET Core | Use Key Vault references in App Configuration | Managed identities for Azure resources

NEW QUESTION 9

You are reviewing the message processing code in a backend worker service.
You review the following Python code that initializes and starts a Service Bus processor

```
from azure.servicebus import ServiceBusClient, ServiceBusReceiveMode

01 import json
02 from azure.servicebus import ServiceBusClient, ServiceBusReceiveMode
03 from azure.identity import DefaultAzureCredential
04 credential = DefaultAzureCredential()
05 with client.get_queue_receiver(
06     queue_name="inference-requests",
```

Service Bus processor behavior and SDK defaults			
Statement	Yes	No	
Messages are removed from the queue as soon as they are received by the processor.	☐	☐	
If message processing fails due to a transient service error, the message can be retried by another consumer.	☐	☐	
Messages with invalid JSON payloads are retried until the maximum delivery count is reached.	☐	☐	

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

The default Service Bus receive behavior is PeekLock rather than receive-and-delete, so merely receiving a message does not remove it from the queue; it must be settled successfully. If processing fails and the message remains unsettled or the lock is lost, it can become available for another delivery. Invalid JSON, however, is an application-level content problem: Service Bus does not inspect the payload as JSON and automatically retry it to MaxDeliveryCount solely because deserialization fails. The application's settlement/error path determines what happens.

Study Guide Alignment: Azure service integration: Service Bus, Event Grid, Azure Functions triggers/bindings, and event-driven processing.

Official Microsoft Learn References: AI-200 Study Guide | Service Bus message transfers, locks, and settlement | Service Bus messages, routing, and correlation

NEW QUESTION 10

You process Azure Service Bus messages that require a dependent external API call.

If the API is temporarily unavailable, you must delay processing of the message without incrementing the delivery count. You need to find a way to process the message when the API is available while keeping the message accessible. Which message action should you perform?

- A. Defer
- B. Complete
- C. Dead-letter
- D. Abandon

Answer: A

Explanation:

Deferral is intended for a message that cannot be processed now because a particular dependency is temporarily unavailable, while the application wants to keep the message in the main queue for later retrieval by sequence number. Complete would permanently remove it, dead-letter would move it aside as a failed message, and abandon immediately releases the lock for redelivery. Deferral therefore best matches the requirement to postpone the individual message while keeping it available for deliberate later processing.

Study Guide Alignment: Azure service integration: Service Bus, Event Grid, Azure Functions triggers/bindings, and event-driven processing.

Official Microsoft Learn References: AI-200 Study Guide | Service Bus message deferral | Service Bus message transfers, locks, and settlement

NEW QUESTION 10

You are designing an Azure Function app that exposes a public API. The solution must:

Validate incoming request data and return results immediately to the caller. Support Microsoft Entra ID authentication.

Scale automatically under variable load. You need to implement a trigger. Which trigger should you implement?

- A. HTTP
- B. Azure Queue storage
- C. Azure Event Grid
- D. Service Bus topic

Answer: A

Explanation:

An HTTP trigger is the only option that directly accepts a synchronous external request and can return a response immediately to the caller. Azure Functions uses HTTP triggers to implement serverless APIs and webhooks, while Microsoft Entra authentication can be applied to the Function App or API surface. Queue, Event Grid, and Service Bus triggers are asynchronous event-processing mechanisms and do not provide the request/response semantics required by this public API scenario.

Study Guide Alignment: Azure service integration: Service Bus, Event Grid, Azure Functions triggers

/bindings, and event-driven processing.

Official Microsoft Learn References: AI-200 Study Guide | Azure Functions HTTP trigger

NEW QUESTION 11

You deploy a production Azure Function app that connects to an Azure SQL Database. The solution must provide the following functionality:

- Prevent secrets from being exposed in source control.
- Support secret rotation without redeploying the function app.
- Avoid downtime during credential updates.

You need to configure secure and maintainable secret management. What should you configure?

- A. Application settings with Key Vault references
- B. Environment variables in local.settings.json
- C. Parameter file stored in source control
- D. Hard coded connection string in the startup class

Answer: A

Explanation:

Key Vault references in Function/App Service application settings keep the connection secret outside source control while allowing the platform to resolve it at runtime by identity. When a referenced secret version is not fixed, rotation can be picked up without rebuilding or redeploying the function code.

`local.settings.json` is intended for local development and must not be used as a production secret store. A source-controlled parameter file or hard-coded connection string directly violates the security requirement.

Study Guide Alignment: Security and operations: Key Vault, App Configuration, managed identity, OpenTelemetry, Azure Monitor, and KQL-based troubleshooting.

Official Microsoft Learn References: AI-200 Study Guide | Use Key Vault references for App Service and Functions

NEW QUESTION 15

You develop a message-processing service deployed to Azure Container Apps. The service reads messages from an Azure Service Bus queue. The solution must minimize costs by ensuring NO compute resources are consumed when the queue is empty. You need to configure scaling for the service. Which two actions should you perform? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.

- A. A.Increase the scaling rule to allow for the maximum running replica count.
- B.Configure the scaling rule to allow for the termination of all active replicas.
- C.Configure a Kubernetes Event-driven Autoscaler rule that monitors queue length.
- D.Enable HTTP ingress concurrency scaling.

Answer: BC

NEW QUESTION 19

Drag and Drop

You are developing a new page for a website that uses Azure Cosmos DB for data storage. The feature uses documents that have the following format:

```
{
  "name": "John",
  "city": "Seattle"
}
```

You must display data for the new page in a specific order. You create the following query for the page:

```
SELECT *
FROM People p
ORDER BY p.name, p.city DESC
```

You need to configure an Azure Cosmos DB policy to support the query.

How should you configure the policy? To answer, drag the appropriate JSON segments to the correct locations. Each JSON segment may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.

NOTE: Each correct selection is worth one point.

JSON segments

orderBy

sortOrder

ascending

descending

compositeIndexes

Answer Area

```
{
  "automatic": true,
  "ngMode": "Consistent",
  "includedPaths": [
    {
      "path": "/"
    }
  ],
  "excludedPaths": [ ],
  "": [
    {
      "path": "/name", "order": "descending"
    },
    {
      "path": "/city", "order": "
    }
  ]
}
```

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

Box 1: compositeIndexes

Azure Cosmos DB requires a defined composite index for any query utilizing an ORDER BY clause with multiple properties.

Box 2: descending

In an ORDER BY clause, properties without an explicit direction default to ascending order. Your query ORDER BY p.name, p.city DESC translates to an ascending sort on /name and a descending sort on /city.

Reference:

<https://docs.azure.cn/en-us/cosmos-db/index-policy>

NEW QUESTION 21

You have an Event Grid subscription that triggers an Azure Function.

You need to prevent loss of events in case the endpoint returns an HTTP 400 response.

Which action should you perform?

- A. Implement optimistic batchin
- B. Implement asynchronous handshake validatio
- C. Configure a dead-letter destinatio
- D. Configure a retry policy.

Answer: C

Explanation:

Configuring a dead-letter destination is exactly the right step to take.

By default, when an endpoint returns an HTTP 400 (Bad Request) or HTTP 413 (Payload Too Large) response, Azure Event Grid immediately stops delivery attempts and drops the message. It treats these specific errors as non-transient, client-side issues, meaning it skips its standard 24-hour retry policy.

Configuring a dead-letter destination ensures that these dropped events are safely preserved for future troubleshooting and reprocessing.

Reference:

<https://turbo360.com/blog/azure-event-grid-dead-letter-monitoring>

NEW QUESTION 23

You plan to deploy a container to an Azure App Service API app named api1. You host the source code for api1 in a GitHub repository. The container uses the API key at runtime to connect to a backend service.

The container must be able to retrieve the API key at runtime without exposing it in the source repository or Git commit history.

You need to ensure that the API key remains outside of Git commit history and is available to the container at runtime.

Solution: Store the API key as a GitHub repository secret.

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

Correct:

* Store the API key in Azure Key Vault and reference it from an App Service application setting.

Storing the API key in Azure Key Vault and referencing it via App Service application settings is the recommended, secure approach. This strategy completely removes sensitive credentials from your GitHub repository and Git commit history while injecting them safely into your container environment at runtime.

Incorrect:

* Embed the API key as a hardcoded environment variable in the Dockerfile.

* Store the API key as a GitHub repository secret.

Reference:

<https://www.qservicesit.com/full-stack-applications-on-azure>

NEW QUESTION 28

You plan to deploy a container to an Azure App Service API app named api1. You host the source code for api1 in a GitHub repository. The container uses the API key at runtime to connect to a backend service.

The container must be able to retrieve the API key at runtime without exposing it in the source repository or Git commit history.

You need to ensure that the API key remains outside of Git commit history and is available to the container at runtime.

Solution: Embed the API key as a hardcoded environment variable in the Dockerfile.

Does the solution meet the goal?

- A. Yes
- B. No

Answer: B

Explanation:

Correct:

* Store the API key in Azure Key Vault and reference it from an App Service application setting.

Storing the API key in Azure Key Vault and referencing it via App Service application settings is the recommended, secure approach. This strategy completely removes sensitive credentials from your GitHub repository and Git commit history while injecting them safely into your container environment at runtime.

Incorrect:

* Embed the API key as a hardcoded environment variable in the Dockerfile.

* Store the API key as a GitHub repository secret.

Reference:

<https://www.qservicesit.com/full-stack-applications-on-azure>

NEW QUESTION 29

You deploy multiple instances of a change feed processor to handle a high ingestion rate within Azure Cosmos DB for NoSQL.

Each processor instance must process a different subset of partitions.

You need to ensure the workload is load-balanced across all processor instances.

What should you configure?

- A. indexing precision
- B. strong consistency
- C. lease container
- D. autoscale throughput

Answer: C

Explanation:

To handle a high ingestion rate in Azure Cosmos DB for NoSQL, you must deploy multiple instances of the change feed processor using the same workflow name

and same lease container.

Reference:

<https://github.com/Azure/azure-cosmos-dotnet-v3/wiki/Comprehensive-Test-Suite-for-Azure-Cosmos-DB-Change-Feed-Processor:-All-Versions-and-Deletes>

NEW QUESTION 33

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

AI-200 Practice Exam Features:

- * AI-200 Questions and Answers Updated Frequently
- * AI-200 Practice Questions Verified by Expert Senior Certified Staff
- * AI-200 Most Realistic Questions that Guarantee you a Pass on Your First Try
- * AI-200 Practice Test Questions in Multiple Choice Formats and Updates for 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The AI-200 Practice Test Here](#)