

Oracle.1z0-1157-26.v2026-10-12.q24

Exam Code:	1z0-1157-26
Exam Name:	Agentic AI Foundations Associate
Certification Provider:	Oracle
Free Question Number:	24
Version:	v2026-10-12
# of views:	106
# of Questions views:	240
https://www.freepdfdumps.com/Oracle.1z0-1157-26.v2026-10-12.q24.html	

NEW QUESTION: 1

Which Python package is installed first for a simple OCI Responses API setup?

- A. boto3
- B. requests-html
- C. openai
- D. django

Answer: C (LEAVE A REPLY)

A basic Python implementation of the OCI Responses API uses the official OpenAI Python SDK, installed through the openai package. Oracle's Enterprise AI Agents quick-start documentation explicitly instructs developers to install it using `pip install openai` and further clarifies that the Responses API should be invoked using the OpenAI SDK rather than the OCI SDK.

This is possible because OCI's Responses API implements an OpenAI-compatible interface . Developers use familiar OpenAI request structures while configuring the base URL for OCI Generative AI and supplying OCI-compatible authentication. Oracle supports multiple OCI authentication approaches, including user principals, instance principals, and resource principals, while the client API retains the OpenAI-compatible programming model.

The other packages serve unrelated purposes. boto3 is the AWS SDK for Python; requests-html is an HTTP

/HTML processing library; and Django is a Python web application framework. None is the required client package for the documented OCI Responses API quick-start.

Consequently, C is the correct answer and directly matches both Oracle's implementation instructions and the answer identified in the supplied examination file.

Study Guide reference/topic: OCI Enterprise AI Agents - OCI Responses API, OpenAI compatibility, Python SDK setup, endpoints, and OCI authentication.

NEW QUESTION: 2

Which statement describes use cases for input guardrails in the OpenAI Agents SDK?

- A. Logging all user inputs to an audit trail for compliance reporting.
- B. Checking that the agent's final response meets quality and safety thresholds.
- C. Filtering which tools the agent is allowed to call based on user role.
- D. Validating input, detecting unsafe content, and blocking prompt-injection attempts.

Answer: D (LEAVE A REPLY)

Input guardrails are checks applied to the initial user input before or alongside execution of the primary agent workflow. Their purpose is to validate whether incoming content satisfies defined security, safety, relevance, or policy requirements and to interrupt execution when an unacceptable condition is detected.

The official OpenAI Agents SDK documentation states that input guardrails receive the same initial input supplied to the agent and can trigger a tripwire that stops execution.

Guardrails can therefore be used to detect malicious or otherwise disallowed user requests before they propagate through an expensive or action-capable agent workflow. Blocking execution is particularly important for security-sensitive cases because it can prevent the agent from consuming tokens or executing tools when the input fails validation.

Option B describes output guardrails, which run against the final agent output. Role-based tool authorization is a separate tool-access control problem, while audit logging is normally implemented through observability, tracing, or application-level compliance mechanisms rather than defining the primary input-guardrail function.

Therefore, D is the correct answer and agrees with the supplied course source.

Study Guide reference/topic: OpenAI Responses API and Agents SDK - input guardrails, output guardrails, tripwires, safety validation, and execution blocking.

NEW QUESTION: 3

Which four behaviors does every Select AI Agent perform?

- A. Retrieve, Rank, Generate, and Validate
- B. Plan, Use tools, Reflect, and Remember
- C. Plan, Execute, Monitor, and Scale
- D. Plan, Use tools, Summarize, and Terminate

Answer: B (LEAVE A REPLY)

Oracle Select AI Agent is architected around four foundational behaviors: Planning, Tool Use, Reflection, and Memory Management. Oracle documentation describes these as the framework's principal layers. Planning interprets the user's objective, decomposes it into ordered actions, and identifies appropriate capabilities. Tool Use invokes mechanisms such as NL2SQL, RAG, PL/SQL procedures, or external REST services. Reflection evaluates observations returned by those tools and determines whether the current plan should continue, be revised, or use another capability. Memory preserves context and useful information, supporting coherent multi-turn interactions and longer-term continuity.

Oracle explicitly states that Select AI Agent combines planning, tool use, reflection, and memory and implements a ReAct-style agentic pattern in which the agent reasons, acts through tools, evaluates observations, and continues toward the goal.

The alternative answer sets describe generic information-retrieval or operational lifecycle stages but do not correspond to Oracle's defined Select AI Agent architecture.

Consequently, B reproduces the four documented agent behaviors and is the correct answer in the supplied question set.

Study Guide reference/topic: Agentic AI for Oracle AI Database - Select AI Agent architecture, Planning, Tool Use, Reflection, Memory, and ReAct.

NEW QUESTION: 4

Which standard MCP transport supports remote or network-accessible deployments where multiple clients may connect?

- A.** Raw TCP sockets
- B.** STDIO
- C.** Streamable HTTP
- D.** Local Unix pipes only

Answer: C (LEAVE A REPLY)

Streamable HTTP is the standard MCP transport intended for remote or network-accessible client-server communication. Current MCP architecture documentation distinguishes it from STDIO by explaining that Streamable HTTP uses HTTP POST for client-to-server communication and can optionally use Server-Sent Events for streaming. It enables communication with remote MCP servers and can support standard HTTP authentication mechanisms.

The MCP transport specification further establishes two standard transport mechanisms: stdio and Streamable HTTP . With STDIO, the client launches an MCP server as a local subprocess and exchanges JSON-RPC messages through standard input and standard output. That pattern is therefore most appropriate for local process integration. By comparison, a Streamable HTTP server operates as an independent service and can handle multiple client connections, making it suitable for centralized or cloud-hosted MCP deployments.

Raw TCP sockets and local Unix pipes are not the standard remote MCP transport defined by the protocol.

Therefore, C is correct and matches the supplied source material.

Study Guide reference/topic: Model Context Protocol (MCP) Fundamentals - STDIO versus Streamable HTTP transport and remote MCP deployment.

NEW QUESTION: 5

Which prompt addition is used for zero-shot Chain-of-Thought prompting?

- A.** Set temperature equal to zero.
- B.** Upload a structured CSV first.

- C. Let's think step by step.
- D. Disable all external retrieval tools.

Answer: (SHOW ANSWER)

Zero-shot Chain-of-Thought prompting encourages a language model to decompose a problem into intermediate reasoning steps without supplying worked examples in the prompt. The classic prompting addition associated with this technique is "Let's think step by step." The technique is "zero-shot" because the user does not provide demonstrations showing how comparable problems should be solved; instead, a short natural-language instruction encourages stepwise decomposition.

OpenAI's published prompting guidance historically illustrates this exact technique, explaining that instructing a model to reason through a sequence of steps can improve performance on tasks requiring decomposition. The OpenAI Cookbook specifically gives "Let's think step by step" as an example of an instruction used to elicit a series of reasoning steps.

Setting temperature to zero affects sampling variability rather than creating Chain-of-Thought prompting.

Uploading structured data is unrelated to the reasoning technique, and disabling retrieval tools changes the model's information-access environment rather than prompting its problem-solving structure.

Accordingly, C is the intended and technically correct answer. The uploaded source also marks this exact phrase as correct.

Study Guide reference/topic: Introduction to AI Agents - prompting strategies, reasoning decomposition, zero-shot Chain-of-Thought, and agent reasoning patterns.

NEW QUESTION: 6

Which OCI capability is required for serving fine-tuned or imported custom models?

- A. Free Tier compute instances
- B. Dedicated AI Clusters
- C. Shared On-Demand inference
- D. OCI Object Storage buckets

Answer: (SHOW ANSWER)

OCI Generative AI uses Dedicated AI Clusters to provide the isolated compute infrastructure required for fine-tuning and hosting custom model workloads. Oracle defines Dedicated AI Clusters as compute resources dedicated to a customer's models rather than shared with other tenancies. They can be created specifically for fine-tuning or for hosting model endpoints.

Oracle's current model onboarding workflow confirms the requirement. For imported models, the process includes importing the model, creating a hosting Dedicated AI Cluster, creating an endpoint, and then invoking the model. Fine-tuned models similarly require dedicated clusters for fine-tuning and subsequent hosting.

Shared On-Demand inference is appropriate for supported Oracle-hosted pretrained models, but it does not provide the dedicated isolated serving environment required by these custom model workflows. Object Storage can be an input location for model artifacts or training data, but it is storage rather than model-serving infrastructure. General-purpose Free Tier compute is likewise not the managed Generative AI capability Oracle specifies for custom-model serving.

Thus, B is correct and agrees with the uploaded course material.

Study Guide reference/topic: OCI Enterprise AI Agents - Dedicated AI Clusters, imported models, fine- tuned custom models, hosting clusters, and endpoints.

NEW QUESTION: 7

How do On-Demand serving and Dedicated AI Clusters differ for OCI Enterprise AI Models?

- A.** Both options use identical infrastructure but are differentiated by pricing and branding only.
- B.** On-Demand uses shared infrastructure while Dedicated AI Clusters provide isolated GPUs required for custom models.
- C.** On-Demand doesn't use processors while Dedicated AI Clusters use isolated GPUs for all workloads.
- D.** On-Demand supports only embedding models while Dedicated AI Clusters support generative models.

Answer: B (LEAVE A REPLY)

OCI Generative AI provides both On-Demand and Dedicated serving modes. The fundamental architectural distinction is resource isolation. On-Demand model inference uses Oracle-managed shared infrastructure, making it suitable for straightforward consumption of supported pretrained models without reserving dedicated hardware. Dedicated mode assigns dedicated GPU resources to the customer's workload and provides predictable, isolated serving capacity.

Oracle states that Dedicated AI Clusters are compute resources dedicated to a customer's models and are not shared with other tenancies. They are used for hosting supported pretrained models and, critically, for fine- tuning and hosting custom or imported models. The course question correspondingly identifies B as correct.

Option A ignores the infrastructure-isolation difference. Option C is technically impossible because all model inference requires compute resources; On-Demand simply abstracts shared infrastructure. Option D is also incorrect because On-Demand is not limited to embedding models-OCI provides multiple supported model categories through managed inference.

Therefore, B accurately captures both shared-versus-dedicated serving and the importance of Dedicated AI Clusters for custom-model workloads.

Study Guide reference/topic: OCI Enterprise AI Agents - Enterprise AI Models, On-Demand serving, Dedicated AI Clusters, custom models, and GPU isolation.

NEW QUESTION: 8

Which authentication approach should be used for production-grade access to OCI Enterprise AI services?

- A. OCI IAM authentication with signed requests and IAM policies
- B. Browser session cookies stored with application code
- C. Anonymous access to tenancy-level resources
- D. Hardcoded credentials checked into source repositories

Answer: A (LEAVE A REPLY)

For production OCI Enterprise AI workloads, Oracle recommends OCI IAM-based authentication rather than long-lived development credentials. The uploaded source identifies OCI IAM authentication with signed requests and IAM policies as the correct production architecture.

Oracle's OCI Responses API authentication documentation distinguishes service API keys used for testing and early development from IAM authentication intended for production and OCI-managed environments.

IAM-based authentication uses OCI identity principals and request-signing mechanisms and allows authorization to be controlled through centralized IAM policies. Oracle specifically recommends IAM when applications execute in services such as OCI Functions or Oracle Kubernetes Engine, when long-lived API keys should be avoided, or when fine-grained centralized access control is required.

IAM policies also implement least privilege by defining exactly which users, groups, resource principals, or workloads can access individual Generative AI resource types. Browser cookies, anonymous tenancy access, and credentials committed into source repositories violate standard enterprise security practices and significantly increase credential-exposure risk.

Therefore, A is the only production-grade authentication approach among the choices. Study Guide reference/topic: OCI Enterprise AI Agents - OCI IAM, signed requests, policies, principals, least privilege, and production authentication.

NEW QUESTION: 9

Agent has multiply(a,b) and divide(a,b) . User: "What is 15 multiplied by 8, then divided by 3?" How does the OpenAI Agents SDK handle this?

- A. The Runner performs all arithmetic internally without involving the model or tools.
- B. The model calls every available arithmetic tool before answering.
- C. The SDK automatically combines all arithmetic operations into a single tool call.
- D. The model calls multiply(15, 8) , receives the result, then calls divide(120, 3) .

Answer: D (LEAVE A REPLY)

The OpenAI Agents SDK uses an iterative agent loop for multi-step tool execution. In this scenario, the model first determines that it needs the multiply tool and generates a call with the arguments 15 and 8 . The application executes that tool and returns 120 as a tool

result. The model receives the updated conversation state, recognizes that another operation remains, and subsequently requests `divide(120, 3)`. The resulting value is then available for the final response. The uploaded course source explicitly specifies this sequence.

OpenAI's Agents SDK documentation confirms that the Runner repeatedly calls the LLM, executes requested tools, appends their results, and runs the model again until final output is produced.

The Runner does not independently decide to calculate the arithmetic itself. Nor does the SDK automatically merge unrelated function calls into one synthetic operation. Likewise, an agent does not invoke every registered tool indiscriminately; the model selects the tools required by the current task.

Therefore, D accurately describes the sequential model/tool interaction.

Study Guide reference/topic: OpenAI Responses API and Agents SDK - agent loop, sequential tool calls, tool results, Runner orchestration, and multi-step execution.

NEW QUESTION: 10

Which responsibilities are handled by OCI Enterprise AI Agents?

- A. Chunking and indexing documents for retrieval-augmented generation.
- B. Defining the user prompt and instructions for the database.
- C. Managing hosted endpoints, runtime scaling, session management, and observability.
- D. Routing requests between OCI compute instances and storage buckets.

Answer: C (LEAVE A REPLY)

OCI Enterprise AI Agents provides the operational and orchestration capabilities required to run agentic applications at production scale. The uploaded course source identifies these responsibilities as hosted endpoints, runtime scaling, session management, and observability. Current Oracle documentation supports that architectural classification. OCI Generative AI Applications provide a managed runtime for agent workloads and centralize configuration for scaling, storage, networking, authentication, and deployments. Active deployments expose managed endpoints, while autoscaling controls can increase or decrease replicas according to workload metrics. OCI's Responses API also provides conversation state, Conversations, memory, and related context-management facilities for stateful agent interaction. Operational visibility is supported through OCI metrics, monitoring, endpoint telemetry, tracing, and hosted application logs integrated with OCI Observability and Management.

Document chunking/indexing is a retrieval-processing responsibility rather than the complete agent-platform role. Prompt definition remains application logic, and OCI network routing is handled by underlying OCI networking services.

Therefore, C best represents the production responsibilities of the Enterprise AI Agents layer.

Study Guide reference/topic: OCI Enterprise AI Agents - managed runtime, deployments, autoscaling, endpoints, conversations, memory, monitoring, and observability.

NEW QUESTION: 11

Why are docstrings especially important when defining LangChain tools with the `@tool` decorator?

- A. They determine which API provider LangChain uses.
- B. They improve Python execution speed.
- C. They encrypt tool arguments before execution.
- D. They tell the LLM when and how to use the tool.

Answer: D (LEAVE A REPLY)

When LangChain's `@tool` decorator is applied to a Python function, the function's documentation becomes part of the metadata shown to the language model. LangChain explicitly states that, by default, the function's docstring becomes the tool description, helping the model understand when the capability should be selected.

This metadata is operationally significant because tool selection is model-driven. A clear description communicates the tool's purpose, appropriate usage conditions, expected arguments, and semantic boundaries.

LangChain's context-engineering guidance emphasizes that tool names, descriptions, argument names, and argument descriptions guide the model's reasoning about when and how a particular tool should be invoked.

Docstrings therefore affect agent reliability, not Python runtime performance. They do not determine which underlying model provider is used, and they have no cryptographic role in protecting function parameters.

Poor or ambiguous descriptions can cause an LLM to choose an inappropriate tool or supply unsuitable arguments even when the Python implementation itself is technically correct.

Consequently, D precisely captures why docstrings are particularly important for `@tool` - defined LangChain functions. The uploaded source confirms the same answer.

Study Guide reference/topic: LangChain for AI Agents - `@tool` decorator, tool descriptions, docstrings, schemas, tool selection, and context engineering.

NEW QUESTION: 12

What occurs during the MCP initialization phase?

- A. The client and server exchange protocol versions and supported capabilities.
- B. The client completes user authentication as part of the protocol initialization exchange.
- C. The LLM is fine-tuned using the server's tool definitions.
- D. The server executes all tools once to warm internal caches.

Answer: (SHOW ANSWER)

MCP initialization establishes compatibility between the MCP client and server before normal protocol operations begin. During initialization, the parties establish a mutually supported protocol version and exchange their supported capabilities and implementation information. The client initiates the process with an initialize request containing its protocol

version, capabilities, and client information. The server responds with its supported protocol version, capabilities, and server information, after which the client signals that initialization has completed. Authentication is handled at the transport or deployment security layer rather than being the defining initialization exchange. MCP initialization also does not fine-tune the LLM or execute every registered tool. This capability-negotiation process enables OCI agent applications using MCP Calling to understand which remote capabilities can safely be used. Model Context Protocol

NEW QUESTION: 13

What is Oracle AI Database Private Agent Factory?

- A. A backup utility for embeddings
- B. A managed orchestration layer for deploying to Kubernetes
- C. A hardware appliance for training models on-premises
- D. A no-code platform for building agents

Answer: D (LEAVE A REPLY)

Oracle AI Database Private Agent Factory is a no-code platform for building, testing, and deploying intelligent AI agents . The uploaded question set marks D as correct, and current Oracle documentation independently confirms that definition.

Oracle describes Private Agent Factory as a platform intended for both business users and engineers. It provides an Agent Builder with visual and drag-and-drop capabilities, enabling users to construct intelligent assistants and workflows without writing conventional application code. The platform can combine pre-built agents, custom agents, reusable templates, enterprise data, LLMs, APIs, databases, and external tools.

The strategic purpose is to lower the engineering barrier for enterprise agent creation while retaining governance and integration with Oracle AI Database capabilities. Current releases include pre-built agents and workflow automation functionality for rapidly creating business-oriented agentic solutions.

It is not an embedding backup product, dedicated Kubernetes deployment manager, or physical training appliance. Those alternatives describe unrelated infrastructure or administration capabilities.

Therefore, D is directly supported by Oracle documentation.

Study Guide reference/topic: Agentic AI for Oracle AI Database - Private Agent Factory, no-code Agent Builder, pre-built agents, custom agents, workflows, and enterprise integration.

NEW QUESTION: 14

Which three model categories are available through OCI Enterprise AI Models?

- A. Robotics, Audio, and Vision models.
- B. Forecasting, Clustering, and Regression models.
- C. Chat models, Embed models, and Rerank models.
- D. SQL, NoSQL, and Graph models.

Answer: C (LEAVE A REPLY)

OCI Enterprise AI Models provides managed foundation-model capabilities oriented around three principal inference tasks: Chat, Embeddings, and Rerank . Chat models generate conversational or instructional responses and form the reasoning/generation foundation for many agentic applications. Embed models transform text or other supported content into numerical vector representations, enabling semantic search, recommendations, clustering, classification, and retrieval-augmented generation. Rerank models take an initial collection of retrieved candidates and reorder them according to relevance to a query, improving retrieval quality before selected context is passed to a generative model.

Oracle's current OCI Generative AI documentation explicitly identifies Chat, Embeddings, and Rerank as core Enterprise AI Model tasks. Robotics is not one of the defined Enterprise AI model categories, while clustering and classification are applications of embeddings rather than independent model categories. SQL, NoSQL, and Graph describe database technologies rather than generative-model classes.

The uploaded question source also identifies the Chat/Embed/Rerank combination as the correct selection.

Study Guide reference/topic: OCI Enterprise AI Agents - Enterprise AI Models, chat inference, embeddings, reranking, and model-supported agent workflows.

NEW QUESTION: 15

Which statement describes the OCI Responses API?

- A.** It requires a proprietary OCI-only SDK
- B.** It provides an OpenAI-compatible API
- C.** It supports a single model provider
- D.** It supports only read-only operations

Answer: B (LEAVE A REPLY)

The OCI Responses API is explicitly designed as an OpenAI-compatible API for interacting with supported models and constructing agentic workflows. Oracle documents it as both OpenAI-compatible and Open Responses-compliant, allowing applications to use familiar Responses API request structures and the OpenAI SDK while routing execution to OCI Generative AI. The uploaded examination source likewise marks B as correct.

Compatibility is important because developers can use established client patterns rather than adopting a proprietary OCI-only programming interface. Oracle specifically recommends the OpenAI SDK for calling the OCI Responses API. The OCI endpoint differs in its base URL, authentication model, available OCI- hosted models, and platform governance, but the request structure follows the compatible Responses API model. The interface is also not restricted to a single model provider. OCI supports multiple supported hosted models.

Nor is the API read-only: agent workflows can use Function Calling and MCP Calling, allowing applications to execute external actions through controlled tool implementations. It also supports File Search and Code Interpreter.

Therefore, the defining statement among the choices is B.

Study Guide reference/topic: OpenAI Responses API and Agents SDK - OCI Responses API, OpenAI compatibility, supported models, tools, and SDK interoperability.

NEW QUESTION: 16

In the OpenAI Agents SDK, what is the role of the Runner?

- A. It is the deployment platform for hosting agents in production.
- B. It handles authentication and rotates API keys for the agent.
- C. It executes the agent loop.
- D. It defines JSON schemas for function tools at compile time.

Answer: (SHOW ANSWER)

The Runner is responsible for executing the OpenAI Agents SDK agent loop. The uploaded course source identifies this directly as the Runner's role.

When `Runner.run()`, `Runner.run_sync()`, or `Runner.run_streamed()` is invoked, the Runner starts with an agent and user input, calls the configured model, evaluates the model output, and decides what happens next.

If the output is final, execution terminates. If the model requests a tool call, the Runner executes the tool, appends the result, and calls the model again. If the model produces a handoff, the Runner updates the active agent and continues the loop. OpenAI's official documentation describes precisely this lifecycle.

The Runner is therefore an orchestration/runtime component rather than an agent-hosting deployment service.

Authentication configuration exists separately, and function-tool JSON schemas are generated by the tool- definition mechanisms rather than being the Runner's primary responsibility.

This distinction is central to the SDK architecture: the Agent defines behavior and capabilities, while the Runner executes the iterative workflow.

Therefore, C is correct.

Study Guide reference/topic: OpenAI Responses API and Agents SDK - Runner, agent loop, tool execution, handoffs, final output, and runtime orchestration.

Valid 1z0-1157-26 Dumps shared by Actual4test.com for Helping Passing 1z0-1157-26 Exam! Actual4test.com now offer the **newest 1z0-1157-26 exam dumps**, the Actual4test.com 1z0-1157-26 exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com 1z0-1157-26 dumps with Test Engine

here: https://www.actual4test.com/1z0-1157-26_examcollection.html (60 Q&As Dumps, 30%OFF Special Discount: **Freepdfdumps**)

NEW QUESTION: 17

Which approaches can generate embeddings for Oracle AI Vector Search workflows?

- A. Use ONNX models in-database or external REST-based embedding providers
- B. Train a custom embedding model and manually insert vectors row by row
- C. Use external REST APIs exclusively, since in-database embedding is not supported
- D. Export data to a Python environment, generate embeddings, then reimport via SQL Loader

Answer: A (LEAVE A REPLY)

Oracle AI Vector Search supports both in-database embedding generation and external embedding providers , making A the correct answer. The supplied question source identifies the same combination.

For in-database processing, Oracle AI Database includes an ONNX runtime. Compatible ONNX embedding models can be imported as database objects and invoked directly through SQL using functions such as VECTOR_EMBEDDING . This allows vectorization to occur without moving source data outside the database.

Oracle also supports REST-based embedding generation.

DBMS_VECTOR.UTL_TO_EMBEDDING and related APIs can call external or local providers. Officially documented providers include OCI Generative AI, Cohere, OpenAI, Google AI, Hugging Face, Vertex AI, Mistral, Ollama, and Private AI, depending on the operation and configuration.

Manual Python export/import workflows are technically possible in custom architectures, but they are not the supported approaches being tested. Option C is explicitly false because Oracle supports in-database ONNX inference.

Therefore, A correctly captures Oracle's native and external embedding-generation strategies.

Study Guide reference/topic: Agentic AI for Oracle AI Database - ONNX Runtime, VECTOR_EMBEDDING, DBMS_VECTOR, REST embedding providers, and AI Vector Search.

NEW QUESTION: 18

What is the purpose of the Vector Stores API?

- A. Translating text between languages.
- B. Encrypting Object Storage buckets.
- C. Streaming video into AI agents.
- D. Indexing and retrieving data by meaning.

Answer: (SHOW ANSWER)

The Vector Stores API provides infrastructure for ingesting content and retrieving the portions that are semantically relevant to a query. Files associated with a vector store can

be chunked and prepared for vector- based retrieval, allowing applications and agents to locate content based on semantic similarity rather than only exact lexical matches .

OpenAI's official Vector Stores API supports searching a vector store with a natural-language query and returns relevant content chunks together with similarity scores. The API also supports attaching files to vector stores and configuring the chunking strategy used during ingestion. This architecture is foundational to retrieval-augmented generation and File Search workflows: source material is indexed, a user query retrieves semantically related chunks, and those chunks can then provide grounded context to a model.

Language translation is a generative-model task. Object Storage encryption is a cloud-storage security function, while video streaming is unrelated to the purpose of a vector store.

Accordingly, "Indexing and retrieving data by meaning" is the technically correct description and is also the answer specified by the uploaded question source.

Study Guide reference/topic: OpenAI Responses API and Agents SDK - Vector Stores, semantic retrieval, chunking, File Search, similarity ranking, and RAG.

NEW QUESTION: 19

What integration problem does MCP address?

- A.** GPU memory fragmentation during inference
- B.** Latency in cloud-hosted models
- C.** Token-limit problem in large language models
- D.** $N \times M$ custom-connector problem between AI apps and tools

Answer: D (LEAVE A REPLY)

MCP addresses the integration fragmentation created when multiple AI applications must independently connect to multiple external tools, services, and data sources. The uploaded course material characterizes this explicitly as the $N \times M$ custom-connector problem : without a common interoperability layer, every application-to-tool pairing can require a separate integration.

The official MCP architecture supports this framing by defining a standardized client-server protocol. MCP hosts establish clients that communicate with MCP servers, while servers expose reusable capabilities such as tools, resources, and prompts. A compatible AI application therefore consumes capabilities through the MCP protocol rather than requiring a completely bespoke protocol implementation for each downstream system.

MCP's tool-discovery mechanism further allows clients to obtain standardized names, descriptions, and schemas dynamically.

The problem is architectural interoperability, not GPU allocation, inference latency, or context-window limitations. Those issues require separate model, infrastructure, or prompt-management techniques. MCP instead standardizes the boundary between AI applications and external capabilities, reducing duplicated connector logic and enabling reusable integrations.

Study Guide reference/topic: Model Context Protocol (MCP) Fundamentals - interoperability, MCP client /server architecture, tool discovery, and the N × M integration problem.

NEW QUESTION: 20

Which OCI services are used for observability and auditing of deployed AI agents?

- A. OCI Logging, OCI Monitoring, and OCI Audit
- B. OCI Streaming, OCI Queue, and OCI Email Delivery
- C. OCI Events, OCI Functions, and OCI Vault
- D. OCI Data Catalog, OCI Search, and OCI DNS

Answer: [\(SHOW ANSWER\)](#)

OCI production AI architectures use the standard OCI observability and governance services to provide operational visibility and accountability. OCI Logging collects and centralizes service and application logs; OCI Generative AI hosted applications can expose deployment logs that open directly in OCI Logging and the Observability and Management service. OCI Monitoring supplies metrics and alarms for monitoring resource health and operational conditions. OCI Audit records calls made to supported OCI public API endpoints, providing an authoritative record of administrative and resource-management actions for investigation and compliance. Oracle's architecture guidance specifically recommends enabling OCI Logging, OCI Monitoring, and OCI Audit logs for critical AI-platform components. The services in the other options have legitimate OCI purposes, but they do not collectively represent the principal observability-and-auditing stack. Therefore, option A is the verified combination. Oracle Docs

NEW QUESTION: 21

Which behavior is NOT a characteristic of modern LLM-based AI agents?

- A. Adjusting behavior based on observations
- B. Pursuing goals across multiple steps
- C. Using external APIs or databases
- D. Requiring every execution path to be predefined

Answer: [D \(LEAVE A REPLY\)](#)

Modern LLM-based agents are specifically designed to avoid requiring every possible execution path to be predetermined. The uploaded course material therefore correctly identifies "Requiring every execution path to be predefined" as the behavior that is NOT characteristic of an agent.

OpenAI defines agents as systems capable of independently accomplishing workflows using an LLM to manage workflow execution and make decisions. An agent can determine when a workflow is complete, correct its actions after receiving observations, and dynamically select tools according to the current state.

This differs fundamentally from conventional deterministic automation in which developers encode every branch and execution path beforehand.

Agents commonly pursue objectives across multiple reasoning-and-action cycles. They can invoke external APIs, databases, search systems, or other tools; inspect the resulting observations; and choose subsequent actions. A typical agent loop continues until an exit condition is reached rather than following one permanently fixed sequence.

Predetermined rules may still be used for safety, permissions, and guardrails, but the complete path toward the goal does not need to be pre-scripted.

Therefore, D is the correct answer.

Study Guide reference/topic: Introduction to AI Agents - autonomy, agent loops, observations, dynamic tool use, multi-step goal execution, and deterministic workflows.

NEW QUESTION: 22

In the OpenAI Agents SDK, how does a Handoff differ from the Manager pattern?

- A. A Handoff calls tools directly; a Manager only routes messages.
- B. A Handoff uses async execution; a Manager uses sync execution.
- C. A Handoff keeps control; a Manager transfers control.
- D. A Handoff transfers control; a Manager keeps control.

Answer: D ([LEAVE A REPLY](#))

The distinction concerns ownership of the conversation and orchestration flow, not synchronous versus asynchronous execution. In the OpenAI Agents SDK's Manager pattern-also called agents-as-tools-a central manager remains the active user-facing agent. It invokes specialist agents as tools, receives their outputs, synthesizes them, and retains responsibility for the final response. The specialist supports the manager without taking ownership of the conversation.

A Handoff works differently. When the current agent hands the task to another agent, the selected specialist becomes the active agent and takes over the conversation for the remainder of that portion of the run.

OpenAI's official Agents SDK documentation explicitly describes the Manager pattern as retaining control and Handoffs as transferring control to a specialized agent.

This distinction allows architects to select centralized orchestration when a single agent must aggregate results or apply common controls, and decentralized handoffs when specialists should directly own particular interactions. Therefore, D states the relationship correctly and matches the uploaded question's answer key.

Study Guide reference/topic: OpenAI Responses API and Agents SDK - multi-agent orchestration, Agents as Tools/Manager pattern, and Handoffs.

NEW QUESTION: 23

How are tool calls handled between the LLM and the application?

- A. It performs the external operation directly during inference.
- B. It allows tools to run without application control.
- C. It generates a tool-call request for the application to validate and run.
- D. It grants the orchestration layer unrestricted system access.

Answer: (SHOW ANSWER)

For application-defined function tools, an LLM does not inherently execute the external operation itself.

Instead, the model produces a structured tool-call request identifying the selected function and supplying arguments. The application or agent runtime then interprets that request, applies appropriate validation or authorization, invokes the corresponding implementation, and returns the result to the model for subsequent reasoning.

The OpenAI Responses API defines function calls as custom tools supplied by the developer that enable the model to request execution of application code using typed arguments. The OpenAI Agents SDK makes the separation explicit: a `FunctionTool` contains the tool name, description, parameter JSON schema, and an `on_invoke_tool` implementation that actually executes when the runtime processes the model-generated arguments.

This boundary is critical for security. The model proposes an action; controlled application/runtime logic performs the action. The model is therefore not automatically granted direct database, filesystem, network, or operating-system privileges merely because a tool has been described to it.

Options A, B, and D incorrectly remove this enforcement boundary. Consequently, C is the correct architectural description and matches the supplied question source.

Study Guide reference/topic: OpenAI Responses API and Agents SDK - function calling, structured tool calls, application execution, validation, schemas, and security boundaries.

NEW QUESTION: 24

From the LLM's perspective, what is consistent between MCP-served tools and locally defined tools?

- A.** The LLM can only invoke MCP-served tools after explicit user approval.
- B.** The LLM interacts with both through the same tool-calling interface.
- C.** The LLM receives network paths and authentication credentials for MCP tools.
- D.** MCP-served tools always return richer outputs than local tools.

Answer: B (LEAVE A REPLY)

MCP standardizes how external systems expose capabilities to an AI application, but the model does not need to reason about the transport or deployment location of each capability. Once an MCP server's tools are discovered and incorporated into an agent's available tool set, they are represented to the model as callable tools with names, descriptions, and input schemas. Locally implemented function tools are presented through essentially the same model-facing tool abstraction. OpenAI's Agents SDK documentation explicitly states that tools obtained from configured MCP servers are added to the agent's list of available tools, alongside ordinary tools. Therefore, from the LLM's perspective, both are selected and invoked through the tool-calling mechanism rather than through separate network-specific interfaces.

Authentication, network connectivity, server lifecycle, authorization, and actual execution remain responsibilities of the application/MCP infrastructure. They are deliberately abstracted away from the model.

Therefore, option B precisely captures the architectural consistency described in the course question.

Study Guide reference/topic: Model Context Protocol (MCP) Fundamentals - MCP tools, tool discovery, agent tool abstraction, and client-server integration.

Valid 1z0-1157-26 Dumps shared by Actual4test.com for Helping Passing 1z0-1157-26 Exam! Actual4test.com now offer the **newest 1z0-1157-26 exam dumps**, the Actual4test.com 1z0-1157-26 exam **questions have been updated** and **answers have been corrected** get the **newest** Actual4test.com 1z0-1157-26 dumps with Test Engine here: https://www.actual4test.com/1z0-1157-26_examcollection.html (**60** Q&As Dumps, **30%OFF** Special Discount: **Freepdfdumps**)