Boost up Your Certification Score

Amazon
AIP-C01
AWS Certified Generative AI Developer - Professional

For More Information – Visit link below:

https://www.examsboost.com/

Product Version

✓ Up to Date products, reliable and verified.
✓ Questions and Answers in PDF Format.

Visit us at: https://www.examsboost.com/test/aip-c01

Latest Version: 6.5

Question: 1

A company wants to implement an Amazon Bedrock application that helps lawyers search thousands of legal documents. The solution must give lawyers the ability to query based on the type of legal case, jurisdiction, and case outcome. Queries must return information about document authors and versions. The solution must integrate with the company's existing tools and support future growth in volume and query complexity.
Which solution will meet these requirements?

A. Use Amazon Bedrock Knowledge Bases with a flat text embedding store. Include search context in user queries.
B. Use Amazon Bedrock Knowledge Bases with a vector store that supports metadata filtering. Configure metadata attributes for case type, jurisdiction, and case outcome to enable structured retrieval. Store authorship and version information in metadata fields.
C. Use Amazon Comprehend to extract metadata attributes such as case type, jurisdiction, and case outcome from legal documents. Use Amazon Bedrock Knowledge Bases with a flat embedding store.
D. Use Amazon Bedrock Knowledge Bases with documents that include metadata attributes such as case type, jurisdiction, and case outcome embedded inside the text rather than defined as structured fields. Use Amazon Bedrock AgentCore with the RetrieveAndGenerate API to extract metadata from embedded documents during queries.

Answer: B

Explanation:
Option B is correct because Amazon Bedrock Knowledge Bases supports structured document metadata that can be stored separately from the text and used as filters during retrieval. Case type, jurisdiction, and case outcome are ideal filterable metadata attributes, while author and version fields preserve provenance for returned documents. AWS documentation confirms that S3 knowledge-base data sources can use sidecar metadata files and that metadata remains available for filtering even when it is excluded from the embedding. This gives the application precise semantic retrieval plus structured constraints without custom extraction at query time. Option A lacks structured filtering. Option C adds unnecessary Comprehend processing. Option D embeds metadata inside document text, which does not provide reliable field-level filtering and adds unnecessary agent-side extraction complexity. AWS Documentation
==========

Question: 2

A global research company is building a RAG-enabled AI system that uses Amazon Bedrock Knowledge Bases. The company stores documents in Amazon S3 and indexes the documents into an Amazon OpenSearch Serverless vector collection.
When the company evaluates the system, the company identifies three issues. Queries return outdated documents when users request only recent research. Medical research queries return both medical research documents and engineering domain documents. Users can retrieve documents that were authored by researchers who the users should not have access to based on company policies.
The company wants to improve the system so that retrieval becomes more precise and contextually appropriate.
Which solution will meet these requirements?

A. Add custom metadata fields to the documents in Amazon S3 to record timestamp, authorship, and research domain. Index the document embeddings and the custom metadata fields into the OpenSearch Serverless vector collection. At query time, use the knowledge base to run vector similarity search and return the stored metadata with the results to help the model interpret document relevance.
B. Use S3 object metadata to store each document's timestamp. Use a custom metadata field to record authorship. Use S3 object tags to record the research domain. Propagate the metadata fields into the OpenSearch Serverless vector collection as filterable attributes. Use Knowledge Bases to apply timestamp, author, and domain filters before running vector similarity search.
C. Store documents in Amazon S3. Extract timestamp, author metadata, and research-domain tags, and store the data in an Amazon DynamoDB table. During retrieval, apply author, domain, and timestamp filters in DynamoDB to identify candidate document IDs. Use the filtered document IDs to narrow the vector similarity search in the OpenSearch Serverless collection.
D. Store documents in Amazon S3 with custom metadata to record authorship. Use Amazon Comprehend to classify each document into a research domain. Store the classification results in Amazon Aurora. Query Aurora during retrieval to identify relevant domains before performing vector similarity search in the OpenSearch Serverless collection.

Answer: B

Explanation:
Option B is correct because the retrieval problem requires filtering before semantic similarity search, not merely returning metadata after retrieval. Amazon Bedrock Knowledge Bases can apply filters to metadata associated with source documents, so timestamp, author, and research-domain attributes can restrict the candidate set before vector results are returned. This prevents old, wrong-domain, or unauthorized-author documents from entering the retrieved context. AWS also documents S3 knowledge-base metadata files and filterable metadata attributes. Option A only returns metadata with already-selected vector results, so irrelevant documents can still be retrieved. Option C introduces DynamoDB and a two-stage custom retrieval path. Option D adds Comprehend and Aurora, increasing operational complexity. The key design principle is to make the required attributes filterable in the vector retrieval layer. AWS Documentation
==========

Question: 3

A company is building an AI-powered customer support application that uses Amazon Bedrock FMs. The application must provide users with recommendations based on an internal Amazon Bedrock knowledge base.
Users will query documents that the company stores in Amazon S3 and structured data that is in Amazon RDS. The application must provide citations for the sources it uses to generate recommendations. The application must also provide secure access to sensitive data.
Which solution will meet these requirements?

A. Associate one Amazon Bedrock AgentCore agent with two knowledge bases. Configure one knowledge base to access documents from Amazon S3 and the other to access structured data from Amazon RDS. Configure the agent to preserve attribution metadata in responses. Use IAM roles to control access to sensitive data.
B. Create two Amazon Bedrock AgentCore agents. Associate one agent with a knowledge base that is connected to the documents in Amazon S3 and the other agent with a knowledge base that is connected to Amazon RDS. Develop a custom AWS Lambda function to query both agents simultaneously and combine the results.
C. Create one Amazon Bedrock AgentCore agent that uses AWS Lambda functions to directly query Amazon S3 for documents and Amazon RDS for structured data. Process the responses and merge the results into a single response.
D. Configure one Amazon Bedrock AgentCore agent to use a knowledge base that contains the documents from Amazon S3. Configure AWS Lambda functions to query Amazon RDS directly for structured data and to combine the results before returning responses.

Answer: A

Explanation:
Option A best matches the requirement because managed knowledge-base retrieval provides source attribution and lets access be governed through IAM, avoiding custom result-merging logic. Amazon Bedrock RetrieveAndGenerate returns citations tied to retrieved source references. One implementation detail matters under current AWS documentation: structured Knowledge Bases use Amazon Redshift as the query engine; Amazon Redshift federated queries can access Amazon RDS and Aurora operational databases. Therefore, the RDS-backed knowledge-base leg should be implemented through that supported structured-data path rather than interpreted as a direct native RDS vector connector. Options B, C, and D introduce extra agents or Lambda-based retrieval/merging, which increases development effort and makes unified citation handling more custom. A keeps both information sources behind managed retrieval and preserves provenance while IAM controls access. AWS Documentation
==========

Question: 4

A GenAI developer is using Amazon Bedrock to build an application. The application needs to interpret complex user requests in natural language, maintain semantic context across multiple interactions with the same user, and dynamically invoke more than 50 enterprise APIs based on reasoning rather than predefined workflows.
The application must autonomously coordinate backend tools and provide explainable results for compliance reviews. The GenAI developer wants to use Amazon Bedrock AgentCore to implement dynamic orchestration.
Which solution will meet these requirements?

A. Use AgentCore with AWS Step Functions to define static workflows that call APIs. Configure Amazon CloudWatch Logs to collect logs for compliance reviews.
B. Use AgentCore to integrate reasoning modules with dynamic tool invocation APIs. Configure AgentCore agents to orchestrate enterprise APIs based on semantic context and to log reasoning traces for compliance reviews.
C. Use Amazon Bedrock Knowledge Bases to implement semantic search instead of enterprise APIs. Integrate AWS Lambda functions with an AgentCore agent to call APIs to orchestrate runtimes.
D. Use Amazon Bedrock Guardrails with an external LLM chain to manage semantic context and to collect logs for compliance reviews. Invoke APIs directly from the application.

Answer: B

Explanation:
Option B is correct because the requirement is agentic, dynamic orchestration rather than a predefined workflow. AgentCore Memory can preserve conversational context, while AgentCore Gateway provides a unified MCP interface through which an agent can discover and invoke many tools and backend APIs. AWS documents support multiple Gateway targets and consolidated tool discovery, so adding or selecting enterprise capabilities does not require hard-coding every workflow path. AgentCore observability also exposes detailed traces covering processing steps, tool invocations, decision points, errors, and outputs, which supports compliance review and explainability. Option A is fundamentally static orchestration. Option C substitutes retrieval for API execution and still requires custom Lambda orchestration. Option D uses guardrails for a purpose they are not designed to perform; guardrails constrain content but are not a dynamic enterprise-tool orchestration engine. AWS Documentation
==========

Question: 5

A financial services company uses Amazon Bedrock to analyze customer data that is stored in an Amazon S3 bucket. The data includes personally identifiable information (PII). The company must mask PII from foundation model (FM) responses.
Which solution will meet this requirement with the LEAST operational effort?

A. Create a guardrail in Amazon Bedrock to filter PII content. Define the PII type and set the guardrail action to MASK. Configure Amazon Bedrock to apply the filter to each FM response.

B. Use Amazon Comprehend to detect PII entities in the S3 data before invoking Amazon Bedrock. Configure an AWS Lambda function to call the Amazon Comprehend DetectPiiEntities API to mask detected PII. Store the processed data back to the original S3 bucket.
C. Use Amazon Macie to scan the S3 bucket for PII data. Configure an AWS Lambda function to store PII in a second S3 bucket. Use an Amazon EventBridge rule to invoke the Lambda function.
D. Configure an AWS Lambda function to search for PII data. Implement a step in the Lambda function code to store the PII in a second S3 bucket and non-PII data into a third S3 bucket.

Answer: A

Explanation:
Option A is correct because Amazon Bedrock Guardrails provides a native sensitive-information filter for personally identifiable information. The filter can detect supported PII types and use the MASK action so detected values in model requests or responses are replaced with placeholders such as the PII type. This directly satisfies the requirement to mask PII from foundation model responses with minimal operational effort. It avoids creating and maintaining an additional preprocessing pipeline. Option B can detect or redact PII with Amazon Comprehend, but it requires Lambda logic and data-processing workflows before Bedrock invocation. Options C and D require scanning, copying, and segregating objects across S3 buckets and still do not directly enforce response masking. For response-level protection in Bedrock, the managed Guardrails capability is the simplest and most purpose-built control. AWS Documentation
==========

Thank You for Trying Our Product

For More Information – Visit link below:

https://www.examsboost.com/

15 USD Discount Coupon Code:

G74JA8UF

FEATURES

- ✓ 90 Days Free Updates
- ✓ Money Back Pass Guarantee
- ✓ Instant Download or Email Attachment
- ✓ 24/7 Live Chat Support
- ✓ PDF file could be used at any Platform
- ✓ 50,000 Happy Customer

