databricks

Databricks Exam Guide

Databricks Certified Generative AI Engineer Associate

Important note: This exam will change on March 18, 2026. See notes immediately below.

Purpose of this Exam Guide

This exam guide gives you an overview of the exam and what it covers to help you determine your exam readiness. This document will be updated anytime there are any changes to an exam (and when those changes will take effect on an exam) so that you can be prepared. **This version covers the currently live version up to and including March 17, 2026. Please see below for the exam guide for the exam starting on March 18, 2026.**

Audience Description

The Databricks Certified Generative AI Engineer Associate certification exam assesses an individual's ability to design and implement LLM-enabled solutions using Databricks. This includes problem decomposition to break down complex requirements into manageable tasks as well as choosing appropriate models, tools, and approaches from the current generative AI landscape for developing comprehensive solutions. It also assesses Databricks-specific tools such as Vector Search for semantic similarity searches, Model Serving for deploying models and solutions, MLflow for managing solution lifecycle, and Unity Catalog for data governance. Individuals who pass this exam can be expected to build and deploy performant RAG applications and LLM chains that take full advantage of Databricks and its toolset.

About the Exam

- Number of scored questions: 45 multiple-choice or multiple-selection items*
- Time Limit: 90 minutes
- Registration fee: $200
- Delivery method: Online Proctored
- Prerequisite: None is required; related course attendance and six months of hands-on experience are highly recommended.
- Validity: 2 years.
- Recertification: Recertification is required every two years to maintain your certified status. To recertify, you must take the full exam that is currently live. Please review the "Getting Ready for the Exam" section on the exam webpage to prepare for taking the exam again.
- Unscored questions: Exams may include unscored questions to gather statistical information for future use. These questions are not identified on the form and do not

impact your score. Additional time is factored into the exams to account for these questions.

Recommended Preparation

- All current Databricks Academy ILT courses related to the Generative AI learner role, specifically, Generative AI Engineering with Databricks.
- Self-Paced (available in Databricks Academy): Generative AI Engineering with Databricks.

  This self-paced course will soon be replaced with the following four modules.
  - Generative AI Solution Development (RAG)
  - Generative AI Application Development (Agents)
  - Generative AI Application Evaluation and Governance
  - Generative AI Application Deployment and Monitoring
- Knowledge of current LLM's and their capabilities
- Knowledge of prompt engineering, prompt generation, and evaluation
- Knowledge of current related online tools and services like LangChain, Hugging Face Transformers, etc.
- Working knowledge of Python and its libraries that support RAG application and LLM chain development
- Working knowledge of current APIs for data preparation, model chaining, etc.
- Relevant Databricks Documentation resources

Exam outline

Section 1: Design Applications

- Design a prompt that elicits a specifically formatted response
- Select model tasks to accomplish a given business requirement
- Select chain components for a desired model input and output
- Translate business use case goals into a description of the desired inputs and outputs for the AI pipeline
- Define and order tools that gather knowledge or take actions for multi-stage reasoning

Section 2: Data Preparation

- Apply a chunking strategy for a given document structure and model constraints
- Filter extraneous content in source documents that degrades quality of a RAG application
- Choose the appropriate Python package to extract document content from provided source data and format.
- Define operations and sequence to write given chunked text into Delta Lake tables in Unity Catalog
- Identify needed source documents that provide necessary knowledge and quality for a given RAG application

- Identify prompt/response pairs that align with a given model task
- Use tools and metrics to evaluate retrieval performance
- Design retrieval systems using advanced chunking strategies.
- Explain the role of re-ranking in the information retrieval process.
- Apply chunking strategy for a given document structure

Section 3: Application Development

- Create tools needed to extract data for a given data retrieval need
- Select Langchain/similar tools for use in a Generative AI application.
- Identify how prompt formats can change model outputs and results
- Qualitatively assess responses to identify common issues such as quality and safety
- Select chunking strategy based on model & retrieval evaluation
- Augment a prompt with additional context from a user's input based on key fields, terms, and intents
- Create a prompt that adjusts an LLM's response from a baseline to a desired output
- Implement LLM guardrails to prevent negative outcomes
- Write metaprompts that minimize hallucinations or leaking private data
- Build agent prompt templates exposing available functions
- Select the best LLM based on the attributes of the application to be developed
- Select an embedding model context length based on source documents, expected queries, and optimization strategy
- Select a model for from a model hub or marketplace for a task based on model metadata/model cards
- Select the best model for a given task based on common metrics generated in experiments
- Utilize Agent Framework for developing agentic systems

Section 4: Assembling and Deploying Applications

- Code a chain using a pyfunc model with pre- and post-processing
- Control access to resources from model serving endpoints
- Code a simple chain according to requirements
- Code a simple chain using langchain
- Choose the basic elements needed to create a RAG application: model flavor, embedding model, retriever, dependencies, input examples, model signature
- Register the model to Unity Catalog using MLflow
- Sequence the steps needed to deploy an endpoint for a basic RAG application
- Create and query a Vector Search index
- Identify how to serve an LLM application that leverages Foundation Model APIs
- Identify resources needed to serve features for a RAG application
- Explain the key concepts and components of Mosaic AI Vector Search
- Identify batch inference workloads and apply `ai_query()` appropriately

Section 5: Governance

- Use masking techniques as guard rails to meet a performance objective
- Select guardrail techniques to protect against malicious user inputs to a Gen AI application
- Use legal/licensing requirements for data sources to avoid legal risk
- Recommend an alternative for problematic text mitigation in a data source feeding a GenAI application

Section 6: Evaluation and Monitoring

- Select an LLM choice (size and architecture) based on a set of quantitative evaluation metrics
- Select key metrics to monitor for a specific LLM deployment scenario
- Evaluate model performance in a RAG application using MLflow
- Use inference logging to assess deployed RAG application performance
- Use Databricks features to control LLM costs for RAG applications
- Use inference tables and Agent Monitoring to track a live LLM endpoint
- Identify evaluation judges that require ground truth
- Compare the evaluation and monitoring phases of the Gen AI application life cycle

Sample Questions

These questions are similar to actual question items and give you a general sense of how questions are asked on this exam. They include exam objectives as they are stated on the exam guide and give you a sample question that aligns to the objective. The exam guide lists all of the objectives that could be covered on an exam. The best way to prepare for a certification exam is to review the exam outline in the exam guide.

Question 1

Objective: Apply a chunking strategy for a given document structure and model constraints

A Generative AI Engineer is loading 150 million embeddings into a vector database that takes a maximum of 100 million.

Which TWO actions can they take to reduce the record count?

A. Increase the document chunk size
B. Decrease the overlap between chunks
C. Decrease the document chunk size
D. Increase the overlap between chunks
E. Use a smaller embedding model

Question 2

Objective: Identify needed source documents that provide necessary knowledge and quality for a given RAG application.

A Generative AI Engineer is assessing the responses from a customer-facing GenAI application that they are developing to assist in selling automotive parts. The application requires the customer to explicitly input `account_id` and `transaction_id` to answer questions. After initial launch, the customer feedback was that the application did well on answering order and billing details, but failed to accurately answer shipping and expected arrival date questions.

Which of the following receivers would improve the application's ability to answer these questions?

A. Create a vector store that includes the company shipping policies and payment terms for all automotive parts
B. Create a feature store table with `transaction_id` as primary key that is populated with invoice data and expected delivery date
C. Provide examples data for expected arrival dates as a tuning dataset, then periodically fine-tune the model so that it has updated shipping information
D. Amend the chat prompt to input when the order was placed and instruct the model to add 14 days to that as no shipping method is expected to exceed 14 days

Question 3

Objective: Choose the appropriate Python package to extract document content from provided source data and format.

A Generative AI Engineer is building a RAG application that will rely on context retrieved from source documents that have been scanned and saved as image files in formats like .jpeg or .png. They want to develop a solution using the least amount of lines of code.

Which Python package should be used to extract the text from the source documents?

A. beautifulsoup
B. scrapy
C. pytesseract
D. pyquery

Question 4

Objective: Select an embedding model context length based on source documents, expected queries, and optimization strategy

A Generative AI Engineer is creating a LLM-based application. The documents for its retriever have been chunked to a maximum of 512 tokens each. The Generative AI Engineer knows that cost and latency are more important than quality for this application. They have several context length levels to choose from.

Which will fulfill their need?

A. context length 512: smallest model is 0.13GB with and embedding dimension 384
B. context length 514: smallest model is 0.44GB and embedding dimension 768
C. context length 2048: smallest model is 11GB and embedding dimension 2560
D. context length 32768: smallest model is 14GB and embedding dimension 4096

Question 5

Objective: Select the best LLM based on the attributes of the application to be developed

A Generative AI Engineer would like to build an application that can update a memo field that is about a paragraph long to just a single sentence gist that shows intent of the memo field, but fits into their application front end.

With which Natural Language Processing task category should they evaluate potential LLMs for this application?

A. text2text Generation
B. Sentencizer
C. Text Classification
D. Summarization

Answers
Question 1: A, B
Question 2: B
Question 3: C
Question 4: A
Question 5: D

databricks

Databricks Exam Guide

Databricks Certified Generative AI Engineer Associate

Provide Exam Guide Feedback

Purpose of this Exam Guide

This exam guide gives you an overview of the exam and what it covers to help you determine your exam readiness. This document will get updated anytime there are any changes to an exam (and when those changes will take effect on an exam) so that you can be prepared. **This version covers the currently live version as of March 18, 2026. Please check back two weeks before you take your exam to make sure you have the most current version.**

Audience Description

The Databricks Certified Generative AI Engineer Associate certification exam assesses an individual's ability to design and implement LLM-enabled solutions using Databricks. This includes problem decomposition to break down complex requirements into manageable tasks as well as choosing appropriate models, tools, and approaches from the current generative AI landscape for developing comprehensive solutions. It also assesses Databricks-specific tools such as Vector Search for semantic similarity searches, Model Serving for deploying models and solutions, MLflow for managing solution lifecycle, and Unity Catalog for data governance. Individuals who pass this exam can be expected to build and deploy performant RAG applications and LLM chains that take full advantage of Databricks and its toolset.

About the Exam

- Number of scored questions: 45 multiple-choice or multiple-selection items*
- Time Limit: 90 minutes
- Registration fee: $200
- Delivery method: Online Proctored
- Prerequisite: None is required; related course attendance and six months of hands-on experience are highly recommended.
- Validity: 2 years.
- Recertification: Recertification is required every two years to maintain your certified status. To recertify, you must take the full exam that is currently live. Please review the "Getting Ready for the Exam" section on the exam webpage to prepare for taking the exam again.
- Unscored questions: Exams may include unscored questions to gather statistical information for future use. These questions are not identified on the form and do not

impact your score. Additional time is factored into the exams to account for these questions.

Recommended Preparation

- All current Databricks Academy ILT courses related to the Generative AI learner role, specifically, Generative AI Engineering with Databricks.
- Self-Paced (available in Databricks Academy): Generative AI Engineering with Databricks, with these courses:
    - Building Retrieval Agents On Databricks
    - Building Single-Agent Applications on Databricks
    - Generative AI Application Evaluation and Governance
    - Generative AI Application Deployment and Monitoring
- Knowledge of current LLM's and their capabilities
- Knowledge of prompt engineering, prompt generation, and evaluation
- Knowledge of current related online tools and services like LangChain, Hugging Face Transformers, etc.
- Working knowledge of Python and its libraries that support RAG application, Agent, and LLM chain development
- Working knowledge of current APIs for data preparation, model chaining, etc.
- Relevant Databricks Documentation resources

Exam outline

Section 1: Design Applications

- Design a prompt that elicits a specifically formatted response
- Select model tasks to accomplish a given business requirement
- Select chain components for a desired model input and output
- Translate business use case goals into a description of the desired inputs and outputs for the AI pipeline
- Define and order tools that gather knowledge or take actions for multi-stage reasoning
- Determine how and when to use Agent Bricks (Knowledge Assistant, Multiagent Supervisor, Information Extraction) to solve problems

Section 2: Data Preparation

- Apply a chunking strategy for a given document structure and model constraints
- Filter extraneous content in source documents that degrades quality of a RAG application
- Choose the appropriate Python package to extract document content from provided source data and format.
- Define operations and sequence to write given chunked text into Delta Lake tables in Unity Catalog

- Identify needed source documents that provide necessary knowledge and quality for a given RAG application
- Use tools and metrics to evaluate retrieval performance
- Design retrieval systems using advanced chunking strategies
- Explain the role of re-ranking in the information retrieval process

Section 3: Application Development

- Select Langchain/similar tools for use in a Generative AI application.
- Qualitatively assess responses to identify common issues such as quality and safety
- Select chunking strategy based on model & retrieval evaluation
- Augment a prompt with additional context from a user's input based on key fields, terms, and intents
- Create a prompt that adjusts an LLM's response from a baseline to a desired output
- Implement LLM guardrails to prevent negative outcomes
- Select the best LLM based on the attributes of the application to be developed
- Select an embedding model context length based on source documents, expected queries, and optimization strategy
- Select a model for from a model hub or marketplace for a task based on model metadata/model cards
- Select the best model for a given task based on common metrics generated in experiments
- Utilize MLflow and Agent Framework for developing agentic systems
- Compare the evaluation and monitoring phases of the Gen AI application life cycle
- Enable multi-agent systems to leverage Genie Spaces or conversational API to retrieve data

Section 4: Assembling and Deploying Applications

- Code a chain using a pyfunc model with pre- and post-processing
- Control access to resources from model serving endpoints
- Code a simple chain according to requirements
- Choose the basic elements needed to create a RAG application: model flavor, embedding model, retriever, dependencies, input examples, model signature
- Register the model to Unity Catalog using MLflow
- Create and query a Vector Search index
- Identify how to serve an LLM application that leverages Foundation Model APIs
- Explain the key concepts and components of Mosaic AI Vector Search
- Identify batch inference workloads and apply `ai_query()` appropriately
- Configure vector search for a particular solution based on number of embeddings, update frequency, latency, and cost requirements.
- Configure a persistent datastore to store and retrieve intermediate memory or structured information.

- Apply CI/CD best practices such as updating a Vector Search index, promoting prompts across environments, and testing individual components of an agent.
- Integrate managed, external, and custom MCP servers based on a given application requirements
- Apply prompt version control and manage prompt lifecycle
- Develop an appropriate interactive user facing interface for an agent usage scenario (Apps, Slack, Teams, etc.)

Section 5: Governance

- Use masking techniques as guard rails to meet a performance objective
- Select guardrail techniques to protect against malicious user inputs to a Gen AI application
- Use legal/licensing requirements for data sources to avoid legal risk
- Recommend an alternative for problematic text mitigation in a data source feeding a GenAI application

Section 6: Evaluation and Monitoring

- Select an LLM choice (size and architecture) based on a set of quantitative evaluation metrics
- Select key metrics to monitor for a specific LLM deployment scenario
- Evaluate agent performance using MLflow scoring and tracing
- Use inference logging to assess deployed RAG application performance
- Use Databricks features to control LLM costs
- Use inference tables and Agent Monitoring to track a live LLM endpoint
- Identify evaluation judges that require ground truth
- Use AI Gateway (Inference Tables, Usage Tables, and rate limiting) to track an LLM or agent deployed via Agent Framework.
- Use Databricks custom Scorers for evaluating agents and LLMs
- Incorporate SME feedback to improve agent performance

Sample Questions

These questions are similar to actual question items and give you a general sense of how questions are asked on this exam. They include exam objectives as they are stated on the exam guide and give you a sample question that aligns to the objective. The exam guide lists all of the objectives that could be covered on an exam. The best way to prepare for a certification exam is to review the exam outline in the exam guide.

Question 1

Objective: Apply a chunking strategy for a given document structure and model constraints

A Generative AI Engineer is loading 150 million embeddings into a vector database that takes a maximum of 100 million.

Which TWO actions can they take to reduce the record count?

F. Increase the document chunk size
G. Decrease the overlap between chunks
H. Decrease the document chunk size
I. Increase the overlap between chunks
J. Use a smaller embedding model

Question 2

Objective: Identify needed source documents that provide necessary knowledge and quality for a given RAG application.

A Generative AI Engineer is assessing the responses from a customer-facing GenAI application that they are developing to assist in selling automotive parts. The application requires the customer to explicitly input `account_id` and `transaction_id` to answer questions. After initial launch, the customer feedback was that the application did well on answering order and billing details, but failed to accurately answer shipping and expected arrival date questions.

Which of the following receivers would improve the application's ability to answer these questions?

E. Create a vector store that includes the company shipping policies and payment terms for all automotive parts
F. Create a feature store table with `transaction_id` as primary key that is populated with invoice data and expected delivery date
G. Provide examples data for expected arrival dates as a tuning dataset, then periodically fine-tune the model so that it has updated shipping information
H. Amend the chat prompt to input when the order was placed and instruct the model to add 14 days to that as no shipping method is expected to exceed 14 days

Question 3

Objective: Choose the appropriate Python package to extract document content from provided source data and format.

A Generative AI Engineer is building a RAG application that will rely on context retrieved from source documents that have been scanned and saved as image files in formats like .jpeg or .png. They want to develop a solution using the least amount of lines of code.

Which Python package should be used to extract the text from the source documents?

E. beautifulsoup
F. scrapy
G. pytesseract
H. pyquery

Question 4

Objective: Select an embedding model context length based on source documents, expected queries, and optimization strategy

A Generative AI Engineer is creating a LLM-based application. The documents for its retriever have been chunked to a maximum of 512 tokens each. The Generative AI Engineer knows that cost and latency are more important than quality for this application. They have several context length levels to choose from.

Which will fulfill their need?

E. context length 512: smallest model is 0.13GB with and embedding dimension 384
F. context length 514: smallest model is 0.44GB and embedding dimension 768
G. context length 2048: smallest model is 11GB and embedding dimension 2560
H. context length 32768: smallest model is 14GB and embedding dimension 4096

Question 5

Objective: Select the best LLM based on the attributes of the application to be developed

A Generative AI Engineer would like to build an application that can update a memo field that is about a paragraph long to just a single sentence gist that shows intent of the memo field, but fits into their application front end.

With which Natural Language Processing task category should they evaluate potential LLMs for this application?

E. text2text Generation
F. Sentencizer
G. Text Classification
H. Summarization

Answers
Question 1: A, B
Question 2: B
Question 3: C
Question 4: A
Question 5: D