databricks

Databricks Exam Guide

Databricks Certified Generative AI Engineer Associate

Provide Exam Guide Feedback

Purpose of this Exam Guide

The purpose of this exam guide is to give you an overview of the exam and what is covered on the exam 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 June 1, 2024. 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 items: 45 multiple-choice or multiple-selection questions
- Time Limit: 90 minutes
- Registration fee: $200
- Delivery method: Online Proctored
- Test aides: None allowed
- Prerequisite: None required; course attendance and six months of hands-on experience in Databricks is highly recommended. Also, see Recommended Preparation in this document.
- 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.

Recommended Preparation

- All current Databricks Academy courses related to the Generative AI learner role, specifically, Generative AI Engineering with Databricks
- 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

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 a 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

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

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
- Recommend an alternative for problematic text mitigation in a data source feeding a RAG application
- Use legal/licensing requirements for data sources to avoid legal risk

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

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 ordered 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 a 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