KaozhengPro

IT認證考試題庫

專業平臺

考證寶提供最新考古題與模擬試題
協助您高效通過認證考試

www.kaozhengpro.com

Exam : **GES-C01**

Title : SnowPro® Specialty: Gen AI Certification Exam

Version : DEMO

1.A data application developer is tasked with building a multi-turn conversational AI application using Streamlit in Snowflake (SiS) that leverages the COMPLETE (SNOWFLAKE. CORTEX) LLM function. To ensure the conversation flows naturally and the LLM maintains context from previous interactions, which of the following is the most appropriate method for handling and passing the conversation history?

○ The developer should store the entire conversation history in a temporary table in Snowflake and query it with each new turn, passing only the latest user message to the `COMPLETE` function.

○ Snowflake automatically manages conversational context for `COMPLETE` within the session, so the developer only needs to pass the current user prompt as a string.

○ The conversation history must be explicitly managed within the Streamlit application's state, typically by initializing `st.session_state.messages = []` and appending each user and assistant message as an object with `'role'` and `'content'` keys, then passing the full list to the `prompt_or_history` argument of `COMPLETE`.

○ The developer should concatenate all previous user prompts and assistant responses into a single, long string, and pass this as the `<prompt>` argument to `COMPLETE` for each turn.

○ The `COMPLETE` function has an optional `'conversation_id'` parameter that automatically retrieves and manages conversation history when provided.

A. Option A
B. Option B
C. Option C
D. Option D
E. Option E
Answer: C

2.A Streamlit application developer wants to use AI_COMPLETE (the latest version of COMPLETE (SNOWFLAKE. CORTEX)) to process customer feedback. The goal is to extract structured information, such as the customer's sentiment, product mentioned, and any specific issues, into a predictable JSON format for immediate database ingestion.
Which configuration of the AI_COMPLETE function call is essential for achieving this structured output requirement?

○ Including detailed instructions within the prompt string, such as `'Extract sentiment, product, and issues as a JSON object: { "sentiment": "...", "product": "...", "issues": "..." }'`.

○ Setting the `temperature` option to `0` and `max_tokens` to a high value, which implicitly guides the LLM to produce structured output.

○ Utilizing the `response_format` argument with a JSON schema object that precisely defines the expected structure, data types, and required fields for the output.

○ Using the `AI_EXTRACT` function multiple times, once for each piece of information (sentiment, product, issues) to be extracted, and then manually combining the results into a JSON object.

○ Enabling the `guardrails` option with a custom validation rule to ensure the LLM's raw text output conforms to a JSON pattern.

A. Option A
B. Option B
C. Option C
D. Option D
E. Option E
Answer: C
Explanation:
'AI_COMPLETE Structured Outputs' (and its predecessor ‘COMPLETE Structured Outputs’) specifically allows supplying a JSON schema as the 'response_format’ argument to ensure completion responses follow a predefined structure. This significantly reduces the need for post-processing in AI data pipelines and enables seamless integration with systems requiring deterministic responses. The JSON schema object defines the structure, data types, and constraints, including required fields. While prompting the model to 'Respond in JSON' can improve accuracy for complex tasks, the ‘response_format’ argument is the direct mechanism for enforcing the schema. Setting ‘temperature’ to 0 provides more consistent

results for structured output tasks.
Option A is a form of prompt engineering, which can help but does not guarantee strict adherence as response_format' does.
Option B controls randomness and length, not output structure.
Option D is less efficient for extracting multiple related fields compared to a single structured output call.
Option E's 'guardrails' are for filtering unsafe or harmful content, not for enforcing output format.

3.A Snowflake developer, AI_ENGINEER, is creating a Streamlit in Snowflake (SiS) application that will utilize a range of Snowflake Cortex LLM functions, including SNOwFLAKE.CORTEX.COMPLETE,SNOwFLAKE.CORTEX.CLASSIFY_TEXT, and SNOwFLAKE.CORTEX.EMBED_TEXT_768. The application also needs to access data from tables within a specific database and schema. AI_ENGINEER has created a custom role, app_dev_role, for the application to operate under.
Which of the following privileges or roles are absolutely necessary to grant to app_dev_role for the successful execution of these Cortex LLM functions and interaction with the specified database objects? (Select all that apply.)

- ☐ The `SNOWFLAKE.CORTEX_USER` database role, which provides the necessary permissions to call Snowflake Cortex AI functions.
- ☐ The `CREATE SNOWFLAKE.ML.DOCUMENT_INTELLIGENCE` privilege on the schema where the application resides.
- ☐ The `USAGE` privilege on the specific database and schema where the Streamlit application and its underlying data tables are located.
- ☐ The `ACCOUNTADMIN` role to ensure unrestricted access to all Snowflake Cortex features.
- ☐ The `CREATE COMPUTE POOL` privilege to provision resources for the Streamlit application.

A. Option A
B. Option B
C. Option C
D. Option D
E. Option E
Answer: A,C
Explanation:
To execute Snowflake cortex AI functions such as 'SNOWFLAKE.CORTEX.COMPLETE, 'SNOWFLAKE.CORTEX.CLASSIFY_TEXT, and 'EMBED_TEXT_768' (or their SAE prefixed counterparts), the role used by the application in this case) must be granted the 'SNOWFLAKE.CORTEX_USER database role. Additionally, for the Streamlit application to access any database or schema objects (like tables for data input/output, or for the Streamlit app itself if it is stored as a database object), the USAGE privilege must be granted on those specific database and schema objects.
Option B, 'CREATE SNOWFLAKE.ML.DOCUMENT_INTELLIGENCE, is a privilege specific to creating Document AI model builds and is not required for general Cortex LLM functions.
Option D, 'ACCOUNTADMIN', grants excessive privileges and is not a best practice for application roles.
Option E, 'CREATE COMPUTE POOL', is a privilege related to Snowpark Container Services for creating compute pools, which is not directly required for running a Streamlit in Snowflake application that consumes Cortex LLM functions.

4.A data application developer is tasked with building a multi-turn conversational AI application using Streamlit in Snowflake (SiS) that leverages the COMPLETE (SNOWFLAKE. CORTEX) LLM function. To

ensure the conversation flows naturally and the LLM maintains context from previous interactions, which of the following is the most appropriate method for handling and passing the conversation history?

○ The developer should store the entire conversation history in a temporary table in Snowflake and query it with each new turn, passing only the latest user message to the `COMPLETE` function.

○ Snowflake automatically manages conversational context for `COMPLETE` within the session, so the developer only needs to pass the current user prompt as a string.

○ The conversation history must be explicitly managed within the Streamlit application's state, typically by initializing `st.session_state.messages = []` and appending each user and assistant message as an object with `'role'` and `'content'` keys, then passing the full list to the `prompt_or_history` argument of `COMPLETE`.

○ The developer should concatenate all previous user prompts and assistant responses into a single, long string, and pass this as the `<prompt>` argument to `COMPLETE` for each turn.

○ The `COMPLETE` function has an optional `'conversation_id'` parameter that automatically retrieves and manages conversation history when provided.

A. Option A
B. Option B
C. Option C
D. Option D
E. Option E

Answer: C

Explanation:

To provide a stateful, conversational experience with the 'COMPLETE (SNOWFLAKE.CORTEX)' function (or its latest version, 'AI_COMPLETE'), all previous user prompts and model responses must be explicitly passed as part of the argument. This argument expects an array of objects, where each object represents a turn and contains a 'role' ('system', 'user', or 'assistant') and a ‘content key, presented in chronological order. In Streamlit, ‘st.session_state’ is the standard and recommended mechanism for storing and managing data across reruns of the application, making it ideal for maintaining chat history, by initializing 'st.session_state.messages = [l' and appending messages to it.

Option A is incorrect because 'COMPLETE does not inherently manage history from external tables.

Option B is incorrect as ‘COMPLETE does not retain state between calls; history must be explicitly managed.

Option D is a less effective form of prompt engineering compared to passing structured history, as it loses the semantic role distinction and can be less accurate for LLMs.

Option E describes a non- existent parameter for the 'COMPLETE function.

5.A Streamlit application developer wants to use AI_COMPLETE (the latest version of COMPLETE (SNOWFLAKE.CORTEX)) to process customer feedback. The goal is to extract structured information, such as the customer's sentiment, product mentioned, and any specific issues, into a predictable JSON format for immediate database ingestion.

Which configuration of the AI_COMPLETE function call is essential for achieving this structured output requirement?

○ Including detailed instructions within the prompt string, such as `'Extract sentiment, product, and issues as a JSON object: { "sentiment": "...", "product": "...", "issues": "..." }'`.

○ Setting the `temperature` option to `0` and `max_tokens` to a high value, which implicitly guides the LLM to produce structured output.

○ Utilizing the `response_format` argument with a JSON schema object that precisely defines the expected structure, data types, and required fields for the output.

○ Using the `AI_EXTRACT` function multiple times, once for each piece of information (sentiment, product, issues) to be extracted, and then manually combining the results into a JSON object.

○ Enabling the `guardrails` option with a custom validation rule to ensure the LLM's raw text output conforms to a JSON pattern.

A. Option A
B. Option B

C. Option C
D. Option D
E. Option E
Answer: C
Explanation:
'AI_COMPLETE Structured OutputS (and its predecessor ‘COMPLETE Structured OutputS) specifically allows supplying a JSON schema as the ‘response_format' argument to ensure completion responses follow a predefined structure. This significantly reduces the need for post-processing in AI data pipelines and enables seamless integration with systems requiring deterministic responses. The JSON schema object defines the structure, data types, and constraints, including required fields. For complex tasks, prompting the model to respond in JSON can improve accuracy, but the ‘response_format’ argument is the direct mechanism for enforcing the schema. Setting ‘temperature to 0 provides more consistent results for structured output tasks.
Option A is a form of prompt engineering, which can help but does not guarantee strict adherence as ‘response_format does.
Option B controls randomness and length, not output structure.
Option D, while 'AI_EXTRACT (or EXTRACT ANSWER) can extract information, using it multiple times and then manually combining results is less efficient and less robust than a single 'AI_COMPLETE call with a structured output schema for multiple related fields.
Option E's ‘guardrails' are for filtering unsafe or harmful content, not for enforcing output format.