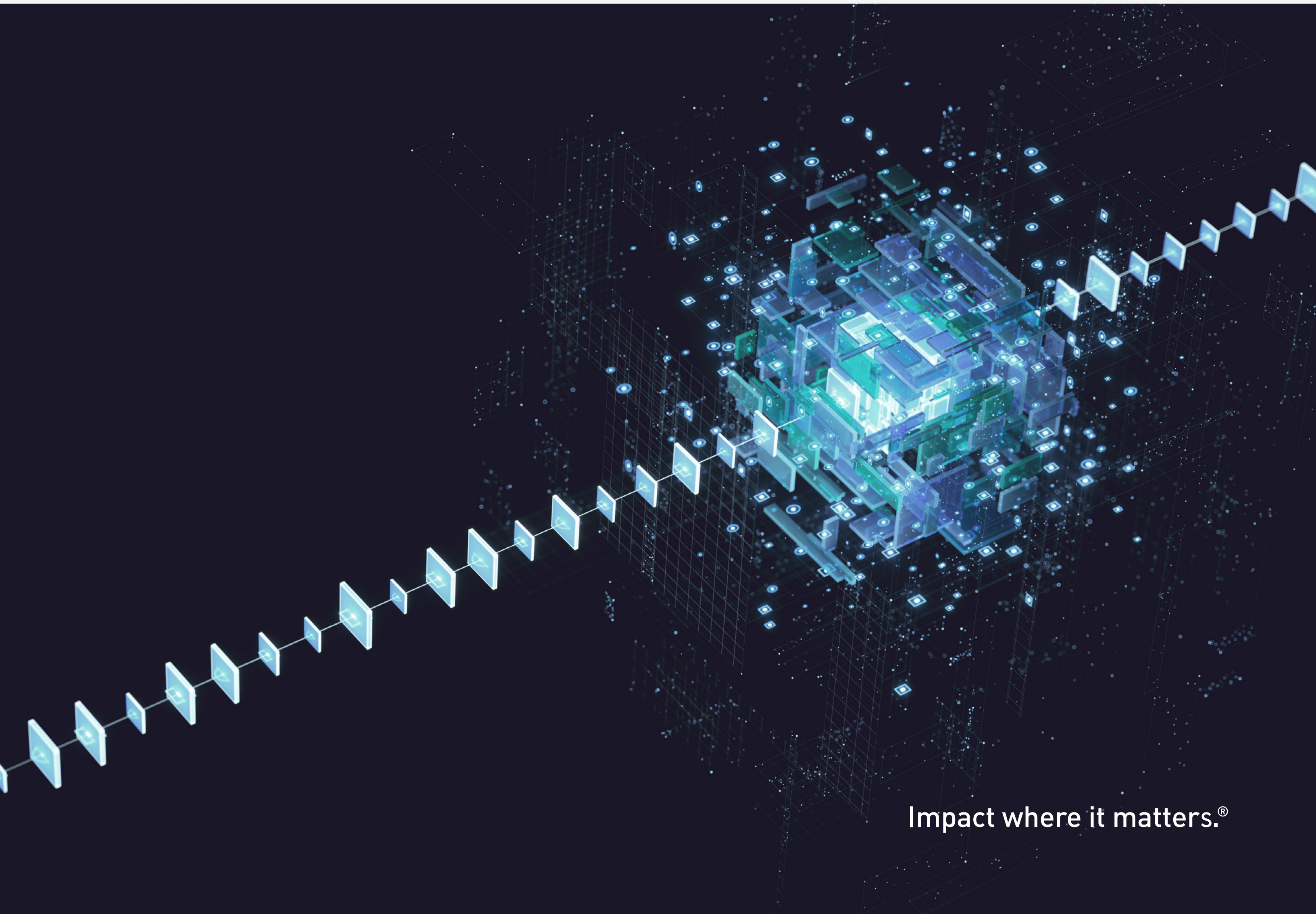
WHITEPAPER

Generative AI and Databricks
Building a gen AI assistant for brand analytics

Co-authored by:

Sandeep Varma, Principal
Abhinav Batra, Principal
Ankush Mantri, Business Technology Solutions Manager
Shivam Shivam, Enterprise Architect
Soham Das, Business Technology Solutions Associate Consultant
Shubham Kankane, Business Technology Solutions Consultant

March 2025



Contents

1. Background

In today's data-driven world, businesses rely heavily on actionable insights from vast datasets. Structured data forms the foundation of nearly every organization's operation. This data, typically organized into rows, columns and predefined formats, resides in relational databases, data warehouses or Lakehouse. It captures vital metrics such as sales performance and operational KPIs.

However, a large pharmaceutical company struggled to derive actionable insights from its structured data distributed across data warehouses. Data analysis became **time-consuming, infrequent** and inaccessible to **non-technical business users.** These users relied heavily on data analysts and IT teams, creating bottlenecks.

To bridge the gap between data and decision-making, there is a growing demand for generative AI-powered assistants that can extract, process and present insights in a user-friendly way. Developing such a tool required addressing several key challenges:

- **Diverse, complex datasets:** The company's data spanned numerous sources, each with its own schema and intricacies. Ensuring insights could be efficiently extracted without noise was a key concern.
- **Data accessibility for non-technical users:** Business stakeholders, such as brand managers and analysts who were **not fluent in structured query language (SQL) or data engineering**, relied on technical teams for every query. This introduced delays and resource strain. The challenge was to enable **natural language querying.** This required translating plain English into structured queries.
- **Real-time analytics needs:** The company needed **real-time or near-real-time analytics.** Traditional batch reports were outdated by the time decisions were made.
- **Adaptability and scalability:** The assistant needed to evolve alongside the organization, supporting future growth and additional use cases.

ZS developed a solution within the Databricks environment, using services such as Genie Spaces, Mosaic AI and MLFlow The Brand Assistant allows users to ask questions in natural language. It translates these questions into SQL using LLMs and LangGraph, enabling data exploration without deep technical knowledge.

The assistant's **real-time processing capabilities** ensure that answers are based on the latest data. It has transformed the pharmaceutical company's analytics landscape by making data **accessible, interpretable and immediately useful.**

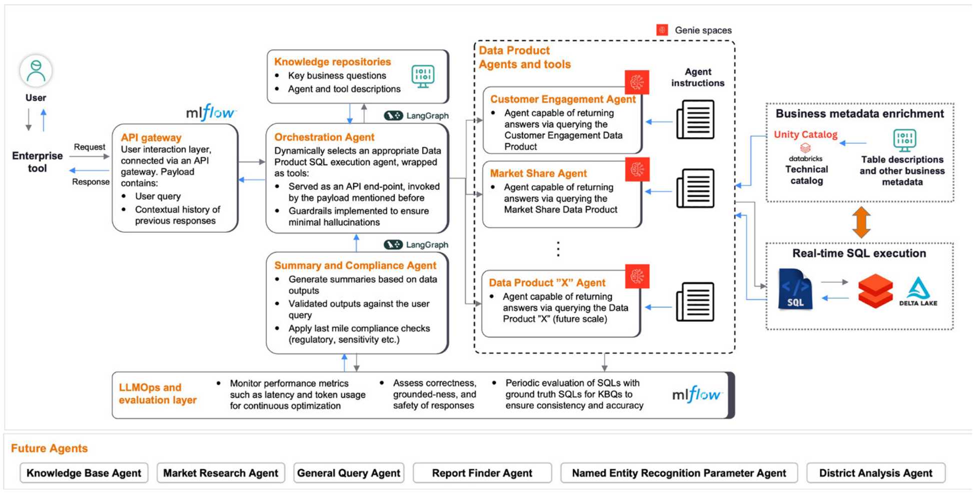
Technical users appreciate the advanced capabilities and integration, while business users appreciate the intuitive, fast answers. This blend of technical sophistication and user-friendly design has made the solution a showcase example of how AI and big data can deliver real business value.

2. Solution approach

To deliver actionable insights in real-time while ensuring precision, customization and scalability, the gen AI-powered Brand Assistant uses Databricks' native capabilities. Below is the solution workflow comprising of different modules and its description:

FIGURE 1:

Brand assistant solution workflow


Genie spaces
User
Enterprise tool
Request
Response
mlflow
API gateway
User interaction layer, connected via an API gateway. Payload contains:
• User query
• Contextual history of previous responses
Knowledge repositories
• Key business questions
• Agent and tool descriptions
LangGraph
Orchestration Agent
Dynamically selects an appropriate Data Product SQL execution agent, wrapped as tools:
• Served as an API end-point, invoked by the payload mentioned before
• Guardrails implemented to ensure minimal hallucinations
LangGraph
Summary and Compliance Agent
• Generate summaries based on data outputs
• Validated outputs against the user query
• Apply last mile compliance checks (regulatory, sensitivity etc.)
Data Product Agents and tools
Agent instructions
Customer Engagement Agent
• Agent capable of returning answers via querying the Customer Engagement Data Product
Market Share Agent
• Agent capable of returning answers via querying the Market Share Data Product
Data Product "X" Agent
• Agent capable of returning answers via querying the Data Product "X" (future scale)
Business metadata enrichment
Unity Catalog
databricks
Technical catalog
Table descriptions and other business metadata
Real-time SQL execution
SQL
DELTA LAKE
LLMOps and evaluation layer
• Monitor performance metrics such as latency and token usage for continuous optimization
• Assess correctness, grounded-ness, and safety of responses
• Periodic evaluation of SQLs with ground truth SQLs for KBQs to ensure consistency and accuracy
mlflow
Future Agents
Knowledge Base Agent
Market Research Agent
General Query Agent
Report Finder Agent
Named Entity Recognition Parameter Agent
District Analysis Agent

1. User interaction layer: Query submission and processing

i. Enterprise tool integration:

- Users interact with the system through an enterprise tool, which could be a web application, chatbot or analytics dashboard.
- The user submits a query related to customer engagement, market share or other
- The query is structured as a request payload, which includes:
  - The user's question
  - The contextual history of the past interactions

ii. API gateway

- The API gateway serves as a secure entry point into the system, ensuring proper authentication and request validation.
- The gateway receives the query and forwards it to the Orchestration Agent, enabling seamless communication between the user and backend data agents.

2. Intelligent query routing and orchestration

i. Orchestration Agent:

- The Orchestration Agent acts as the brain of the system, dynamically determining which Data Product Agent should handle the user's query.
- It is built on LangGraph, which allows for multi-step AI workflows, ensuring that queries are intelligently mapped to relevant datasets.
- The agent execution flow includes:
  - Analyzing user input and identifying the best-suited Data Product Agent to handle the query
  - Ensuring minimal AI hallucinations by applying guardrails before executing SQL queries
  - Executing query-routing logic, directing queries to the appropriate Genie Spaces

ii. Knowledge Repositories:

- The Orchestration Agent references a structured knowledge base to determine the correct data sources and execution paths.
- These repositories store:
  - Key business questions (KBQs)—standardized question formats and intent mapping
  - Descriptions of AI Agents & Tools—helping the Orchestration Agent determine which tool to use for each query

3. Data processing and query execution

i. Data Product Agents (Genie Spaces):

- Each Data Product Agent is responsible for querying a specific business dataset (e.g., customer engagement and market share).
- These agents retrieve real-time insights from structured data sources.
- Types of Data Product Agents:
  - Customer Engagement Agent—Queries Customer Engagement Data Products to analyze user behavior and interaction data.
  - Market Share Agent—retrieves market share data to compare competitive positioning and performance.
- Each agent executes AI-driven SQL queries on Delta Lake, ensuring that the retrieved insights are up-to-date and reliable.
- Architecture is defined and designed for future scalability, enabling expansion into additional analytics use cases by adding more agents in the future

ii. Business metadata enrichment (Unity Catalog)

- The Unity Catalog in Databricks serves as a centralized metadata repository, ensuring that every dataset used in AI-driven analytics is well-documented and structured.
- The system enriches query results with metadata by:
  - Providing table descriptions, column definitions, and business context
  - Ensuring governance and compliance by tracking data lineage
  - Improving query interpretation through AI-assisted metadata retrieval

iii. Real-time SQL execution on Delta Lake

- The Brand Assistant ensures fast and optimized SQL execution by leveraging tables referring to data in Delta Lake.
- SQL queries executed in Genie Spaces retrieve structured responses, which are then validated and enriched before being sent to users.

4. Compliance and response optimization

i. Summary and Compliance Agent

- Once the Data Product Agent fetches the data, the Summary and Compliance Agent performs final validation and enhancement before delivering results to the user.
- **Key responsibilities:**
  - Generates human-readable summaries based on SQL query outputs
  - Validates retrieved data against the original user query for accuracy
  - Applies last-mile compliance checks, ensuring that sensitive or regulated data adheres to governance policies.

5. LLM performance evaluation

i. LLMOps and evaluation layer:

- To ensure that the AI-powered responses remain consistent, reliable and optimized, the system incorporates an large language model operations (LLMOps) Evaluation Layer for continuous monitoring and fine-tuning. Below are the key functionalities:
  - Monitor performance metrics such as query latency, token usage and response times
  - Assess AI-generated SQL correctness by comparing AI-executed queries with ground-truth SQL templates
  - Evaluate AI safety by measuring groundedness, factual consistency and response alignment with business objectives
  - Perform periodic evaluations, ensuring that AI-driven SQL responses remain accurate over time.

ii. Performance dashboard:

- Capture performance of agent in universal column (UC) table against standard KBQs in continuous integration (CI) check.
- Use Databricks Jobs to continuously fetch details from above UC and compute the following:
    - Leveraged MLFlow to create metrics for capturing accuracy, precision and recall for each deployment
    - Using MLFlow Relevance Measure to compute relevance of summary against user query
    - Used LLM as a judge to categorize SQL response into degrees of correctness—good, bad, marked for review.
- These details are then used to populate Databricks Dashboard.

iii. Monitoring dashboard:

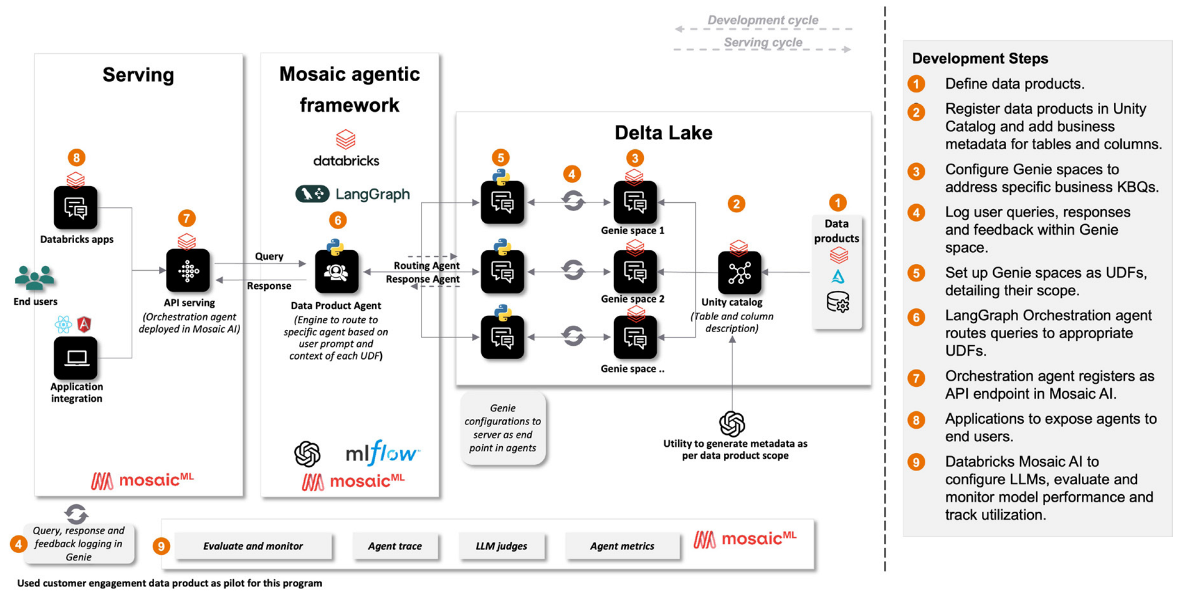
- The purpose of the monitoring dashboard is to capture user queries and responses and measure their relevance and latency by:
    - Using inference tables to capture traces for user queries
    - Use Databricks Jobs to continuously fetch details from inference table and compute the following:
        a. Using MLFlow Traces to capture latency for each step
        b. Using MLFlow Relevance Measure to compute relevance of summary against user query
    - These details are then used to populate Databricks Dashboard.

3. System architecture

3.1. Technical architecture

FIGURE 2:

Solution architecture for deploying agentic gen AI-powered chat assistants


Development cycle
Serving cycle
Serving
8
Databricks apps
End users
Application integration
7
API serving
(Orchestration agent deployed in Mosaic AI)
mosaicML
Mosaic agentic framework
databricks
LangGraph
6
Query
Response
Data Product Agent
(Engine to route to specific agent based on user prompt and context of each UDF)
Routing Agent
Response Agent
mlflow
mosaicML
Delta Lake
5
4
3
Genie space 1
Genie space 2
Genie space ..
2
Unity catalog
(Table and column description)
1
Data products
Genie configurations to server as end point in agents
Utility to generate metadata as per data product scope
4
Query, response and feedback logging in Genie
9
Evaluate and monitor
Agent trace
LLM judges
Agent metrics
mosaicML
Used customer engagement data product as pilot for this program

Development Steps

1. Define data products.
2. Register data products in Unity Catalog and add business metadata for tables and columns.
3. Configure Genie spaces to address specific business KBQs.
4. Log user queries, responses and feedback within Genie space.
5. Set up Genie spaces as UDFs, detailing their scope.
6. LangGraph Orchestration agent routes queries to appropriate UDFs.
7. Orchestration agent registers as API endpoint in Mosaic AI.
8. Applications to expose agents to end users.
9. Databricks Mosaic AI to configure LLMs, evaluate and monitor model performance and track utilization.

Below are the Databricks services which were used to build the Brand Assistant:

Databricks component	Description
Genie Spaces	Logical groupings of related datasets and query processing units that execute **SQL queries** based on structured metadata and business rules.
Unity Catalog	Provides **data governance, schema management, and metadata enrichment,** ensuring datasets are structured and queries are well-defined.
Delta Lake	Facilitates **real-time SQL execution**, enabling efficient, scalable, and structured data processing for AI-driven insights.
MLFlow	Monitors and evaluates **AI model performance, agent responses and accuracy metrics** for continuous optimization.
Mosaic ML	• Supports **large-scale AI inference, model deployment and LLM evaluation,** ensuring seamless AI-driven decision-making. • Deploys the Orchestration Agent as an API endpoint to handle user queries and serve responses in real-time.

Brand Assistant is designed to provide real-time, structured and highly accurate insights by leveraging Databricks' Mosaic AI, Genie Spaces and Unity Catalog. This architecture enables efficient data retrieval, intelligent query orchestration and scalable deployment across enterprise applications.

1. **Data product definition and metadata management**
    - Datasets are structured to address Key Business Questions
    - They are registered in Unity Catalog to maintain centralized governance.
    - Business metadata, including table and column descriptions, is added to enhance data discoverability and usability.
2. **Genie Spaces for query optimization**
    - Genie spaces group related tables that can answer similar business queries.
    - Tables used in aggregation and joins reside in the same Genie Space for efficient query resolution.
    - Domain-specific instructions can be configured to improve response quality and query accuracy.
    - Each Genie Space functions as a user defined function (UDF), making it easily callable viaapplication programming interfaces (APIs) or orchestration systems.
3. **Intelligent query orchestration using LangGraph**
    - The Orchestration Agent classifies query intent and maps metadata to route requests to the appropriate Genie Space.
    - It enables seamless and low-latency query execution, optimizing data retrieval.
    - The agent is deployed as an API endpoint within Databricks Mosaic AI for enabling secure, scalable access.
4. **Continuous monitoring and learning**
    - User interactions, responses and feedback are logged within Genie Spaces.
    - MLflow tracks performance metrics such as—response time, accuracy and relevance.
    - The insights support compliance and drive ongoing optimization.
5. **Scalable enterprise integration and end-user accessibility**
    - The assistant integrates with enterprise applications via APIs.
    - It supports multi-channel access, including web and chat interfaces, ensuring insights are easily accessible to business users.

4. Best practices and learnings

To ensure effective Genie Space and agent design,leverage the following best practices:

1. **Genie Space design**
    - Descriptions should clearly state purpose, the nature of tables included and the types of KBQs the Genies Space is expected to address.
    - If multiple Genie Spaces are used as tools, ensure their descriptions are distinct enough to guide accurate agent routing.
2. **Table and view configuration**
    - Limit each Genie Space to a maximum of five tables.
    - Avoid tables with more than 100 columns.
    - Remove unnecessary or housekeeping columns that don't contribute to answering KBQs.
3. **Use of SQL examples:**
    - Including SQL examples alongside general instructions has shown to significantly improve response consistency.
4. **Custom metric definitions**
    - Clearly define metrics using table and column references in the general instructions.
5. **LoVs column description**
    - Define lists of values (LoVs) for low cardinality columns in the column descriptions.
    - If abbreviations or aliases are used, include both the short form and full name in the column description.
6. **Custom Agent for routing**
    - Build custom agents for Genie Space routing based on specific use cases, instead of relying on Playground exports.
7. **Conversational history**
    - Enable users to ask follow up questions tracking user history, by passing the last n (set to five for current use case) user queries to the Genie Space Tool.
8. **Monitoring with inference tables**
    - Use inference tables registered in the Unity Catalog to monitor agent performance.
    - Logs are typically available within one hour of endpoint invocation. For more details, refer to the following: Inference tables for monitoring and debugging models

5. Business impact

The gen AI-powered Brand Assistant built on Databricks demonstrates a transformative approach to brand analytics, addressing the complexities of modern data environments while enabling faster, more informed decision-making.

1. Accelerated decision-making and quicker insights—**natural language processing allows any business user** (brand managers, marketing analysts and sales teams), to ask questions in plain English without needing technical expertise.
2. Democratization of data—provides **a self-service analytics** platform where business users can directly query structured data using **conversational AI.o SQL or coding is required.**
3. Scalable and future proof analytics—**enables enterprise-wide adoption** of AI-driven analytics and supports evolving use cases.

6. Conclusion

The gen AI-powered Brand Assistant, built on Databricks, represents a paradigm shift in how organizations interact with structured data.It addresses traditional barriers, like fragmented access and reliance on technical teams—by combining cutting-edge AI technologies with a user-friendly experience.

By leveraging natural language processing (NLP), LangGraph-based intelligent query routing and Unity Catalog-driven metadata management, the Brand Assistant empowers business users, regardless of technical expertise, to access insights independently and effortlessly. This fosters a self-service analytics culture, reducing reliance on IT teams and accelerating insight generation across the enterprise.

The ability to make strategic decisions in real time is no longer optional—it is imperative. The Brand Assistant's orchestration using Mosaic AI and UC enabled tables ensures that every data-driven insight is not only accurate but also up to date, enabling organizations to respond to market dynamics proactively. From predictive brand performance analysis to customer engagement insights, the platform is designed to support mission-critical business decisions with unparalleled speed and precision.

The Brand Assistant is a modular, future-ready solution designed to grow and adapt with the business. Unlike static BI tools, this AI-powered system scales with data volume, domain complexity and evolving analytical requirements.

By integrating with existing enterprise applications, the Brand Assistant bridges the gap between data engineering and strategic decision-making, effectively operationalizing data as a core business asset.

It is not merely a tool; it redefines how organizations interact with data, make informed decisions and sustain competitive advantage in an era of digital transformation.

7. References

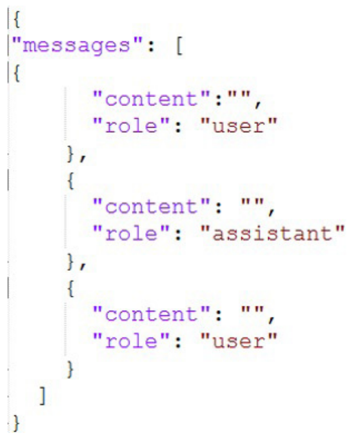
Sr. No.	Description	Source	Link
1	AI/BI Genie	Databricks	Link
2	Mosaic AI Documentation	Databricks	Link
3	Introduction to Unity Catalog	Databricks	Link

8. Appendix

8.1. Code snippets

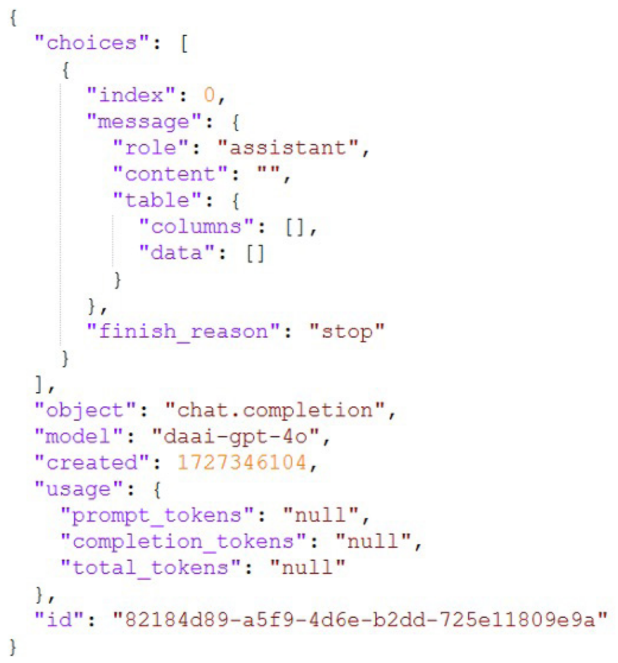
8.1.1. Input payload and response from Enterprise Tool

Sample code snippet to display dropdown and text box:



```
{
"messages": [
{
      "content":"",
      "role": "user"
    },
    {
      "content": "",
      "role": "assistant"
    },
    {
      "content": "",
      "role": "user"
    }
  ]
}
```

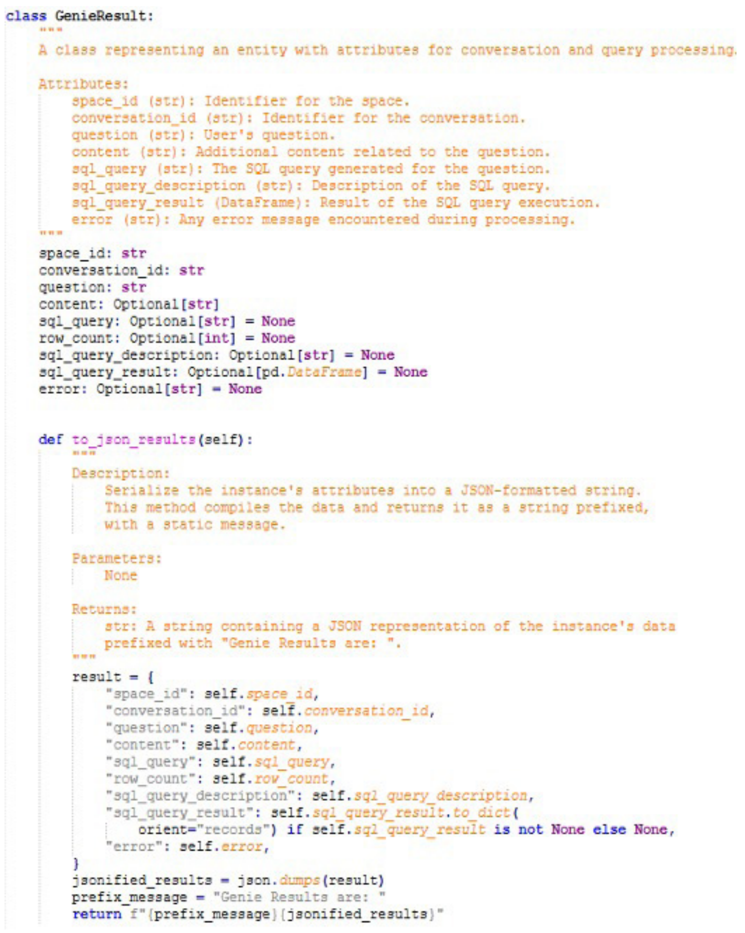
Input Payload



```
{
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "",
        "table": {
          "columns": [],
          "data": []
        }
      },
      "finish_reason": "stop"
    }
  ],
  "object": "chat.completion",
  "model": "daai-gpt-4o",
  "created": 1727346104,
  "usage": {
    "prompt_tokens": "null",
    "completion_tokens": "null",
    "total_tokens": "null"
  },
  "id": "82184d89-a5f9-4d6e-b2dd-725e11809e9a"
}
```

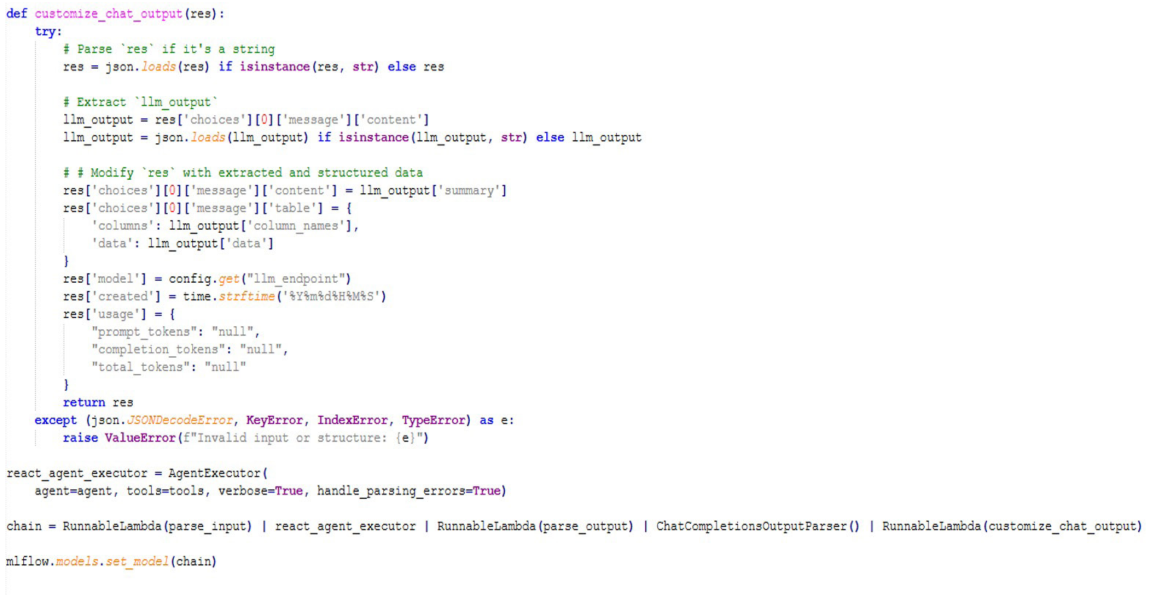
Output response

8.1.2 Defining Genie Rooms code snippet



```python
class GenieResult:
    """
    A class representing an entity with attributes for conversation and query processing.

    Attributes:
        space_id (str): Identifier for the space.
        conversation_id (str): Identifier for the conversation.
        question (str): User's question.
        content (str): Additional content related to the question.
        sql_query (str): The SQL query generated for the question.
        sql_query_description (str): Description of the SQL query.
        sql_query_result (DataFrame): Result of the SQL query execution.
        error (str): Any error message encountered during processing.
    """
    space_id: str
    conversation_id: str
    question: str
    content: Optional[str]
    sql_query: Optional[str] = None
    row_count: Optional[int] = None
    sql_query_description: Optional[str] = None
    sql_query_result: Optional[pd.DataFrame] = None
    error: Optional[str] = None


    def to_json_results(self):
        """
        Description:
            Serialize the instance's attributes into a JSON-formatted string.
            This method compiles the data and returns it as a string prefixed,
            with a static message.

        Parameters:
            None

        Returns:
            str: A string containing a JSON representation of the instance's data
            prefixed with "Genie Results are: ".
        """
        result = {
            "space_id": self.space_id,
            "conversation_id": self.conversation_id,
            "question": self.question,
            "content": self.content,
            "sql_query": self.sql_query,
            "row_count": self.row_count,
            "sql_query_description": self.sql_query_description,
            "sql_query_result": self.sql_query_result.to_dict(
                orient="records") if self.sql_query_result is not None else None,
            "error": self.error,
        }
        jsonified_results = json.dumps(result)
        prefix_message = "Genie Results are: "
        return f"{prefix_message}{jsonified_results}"
```

8.1.3 Agent code snippet



```python
def customize_chat_output(res):
    try:
        # Parse `res` if it's a string
        res = json.loads(res) if isinstance(res, str) else res

        # Extract `llm_output`
        llm_output = res['choices'][0]['message']['content']
        llm_output = json.loads(llm_output) if isinstance(llm_output, str) else llm_output

        # # Modify `res` with extracted and structured data
        res['choices'][0]['message']['content'] = llm_output['summary']
        res['choices'][0]['message']['table'] = {
            'columns': llm_output['column_names'],
            'data': llm_output['data']
        }
        res['model'] = config.get("llm_endpoint")
        res['created'] = time.strftime('%Y%m%d%H%M%S')
        res['usage'] = {
            "prompt_tokens": "null",
            "completion_tokens": "null",
            "total_tokens": "null"
        }
        return res
    except (json.JSONDecodeError, KeyError, IndexError, TypeError) as e:
        raise ValueError(f"Invalid input or structure: {e}")

react_agent_executor = AgentExecutor(
    agent=agent, tools=tools, verbose=True, handle_parsing_errors=True)

chain = RunnableLambda(parse_input) | react_agent_executor | RunnableLambda(parse_output) | ChatCompletionsOutputParser() | RunnableLambda(customize_chat_output)

mlflow.models.set_model(chain)
```

About ZS

ZS is a management consulting and technology firm that partners with companies to improve life and how we live it. We transform ideas into impact by bringing together data, science, technology and human ingenuity to deliver better outcomes for all. Founded in 1983, ZS has more than 13,000 employees in over 35 offices worldwide.

Learn more: zs.com