VBA Macro Analysis and Transformation Tool Documentation

Objective

The objective of the VBA Macro Analysis and Transformation Tool is to automate the documentation and understanding of legacy VBA macros used in critical processes across DFIN. The tool aims to facilitate the transformation of these macros into modern technologies and IT platforms by providing comprehensive analysis and documentation.

Expected Outcome

The desired outcome is to develop a prototype of a macro analysis tool that effectively documents and explains the logic behind VBA macros. This tool will enable organizations to understand legacy systems better and transition them to more modern, efficient solutions through automated documentation.

Technical Overview

Technologies Used

RAG (Retrieval-Augmented Generation):

- **Purpose**: Enhances the response generation process by combining retrieval mechanisms with generative models.
- **Functionality**: Retrieves relevant information from stored logs and generates accurate VBA code based on user queries.

LLM (Large Language Models):

- **Purpose**: Employs large language models to understand and interpret VBA code.

- **Functionality**: Utilizes Cohere embeddings to generate embeddings for VBA code and queries, enhancing semantic understanding and response accuracy.

Vector Databases:

- **Purpose**: Stores embeddings of VBA operations and logs for efficient retrieval and storage.
- **Implementation**: Initial VBA operations are stored with index "sg1". Logs and responses are stored with index "logs", facilitating structured data management and scalability.

Implementation Steps

Initialization and Setup

1. **Vector Database Setup**:
    - Configure the vector database to store embeddings and logs efficiently.
    - Establish index "sg1" to store initial VBA operations as a reference for interpretation and code generation.

User Interface

2. **User Interface Development**:
    - Develop an intuitive interface allowing users to:
        - Upload Excel files (.xlsm) containing VBA macros.
        - Input queries describing required VBA functionality.
    - Ensure simplicity and user-friendliness to facilitate seamless interaction with the tool.

VBA Code Extraction

3. **VBA Code Extraction from Excel Files**:
    - Extract VBA code from uploaded Excel files (.xlsm) using parsing techniques.

- Utilize the Cohere model and embeddings to enhance extraction accuracy and semantic understanding.
- Store extracted code and analysis as logs in the vector database under index "logs".

Generating VBA Code from Descriptions

4. **Description Generation**:
    - Generate descriptions for VBA code based on user queries using RAG, LLM, and historical chat data.
    - Store generated descriptions and corresponding VBA code snippets as logs in the vector database under a new index for structured data management.

Query Processing and VBA Code Generation

5. **Query Processing and Response Generation**:
    - Process user queries describing desired VBA functionality.
    - Retrieve relevant information from stored logs (using Cohere embeddings and index "sg1").
    - Generate accurate VBA code corresponding to the user query based on retrieved information.
    - Return generated VBA code to the user for further use or modification.

Validation and Testing

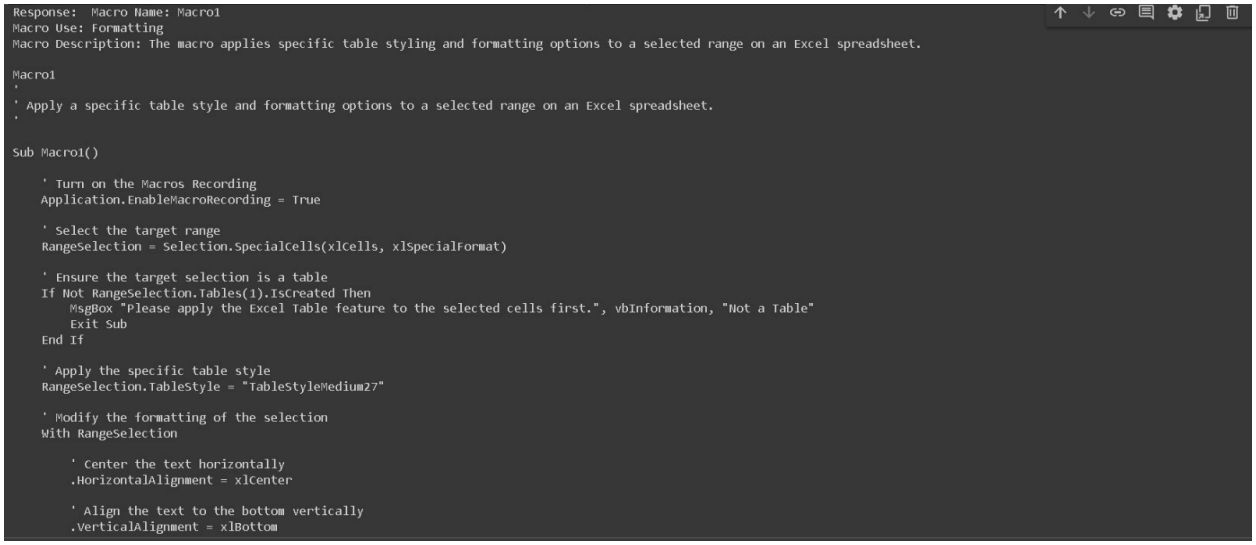
6. **Validation and Testing**:
    - Validate the accuracy and reliability of generated VBA code across various scenarios and complexities.
    - Conduct thorough testing to ensure the tool effectively handles different types of VBA macros and user queries.

Working

This model first gets VBA macros from the file which is then fed to an LLM which generates a response summarizing its working. This summary along with the code and function name are inserted into the ‘logs’. Whenever a user queries for a specific functionality, it retrieves the relevant code if it exists, if not it adds it to the ‘logs’.

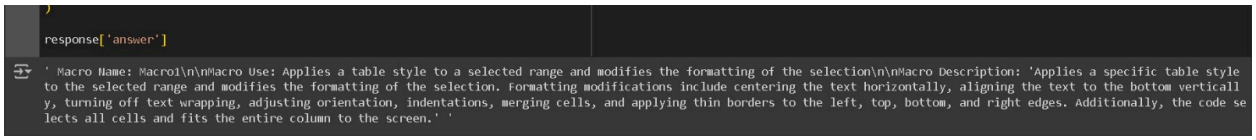
Outputs

Retrieved for asking about a code with functionality: ‘format tables’



```
Response:  Macro Name: Macro1
Macro Use: Formatting
Macro Description: The macro applies specific table styling and formatting options to a selected range on an Excel spreadsheet.

Macro1
'
' Apply a specific table style and formatting options to a selected range on an Excel spreadsheet.
'

Sub Macro1()

    ' Turn on the Macros Recording
    Application.EnableMacroRecording = True

    ' Select the target range
    RangeSelection = Selection.SpecialCells(xlCells, xlSpecialFormat)

    ' Ensure the target selection is a table
    If Not RangeSelection.Tables(1).IsCreated Then
        MsgBox "Please apply the Excel Table feature to the selected cells first.", vbInformation, "Not a Table"
        Exit Sub
    End If

    ' Apply the specific table style
    RangeSelection.TableStyle = "TableStyleMedium27"

    ' Modify the formatting of the selection
    With RangeSelection

        ' Center the text horizontally
        .HorizontalAlignment = xlCenter

        ' Align the text to the bottom vertically
        .VerticalAlignment = xlBottom
```

Summary of code generated when inputting a code which ‘formats tables’



```
)
response['answer']

' Macro Name: Macro1\n\nMacro Use: Applies a table style to a selected range and modifies the formatting of the selection\n\nMacro Description: 'Applies a specific table style to the selected range and modifies the formatting of the selection. Formatting modifications include centering the text horizontally, aligning the text to the bottom vertically, turning off text wrapping, adjusting orientation, indentations, merging cells, and applying thin borders to the left, top, bottom, and right edges. Additionally, the code selects all cells and fits the entire column to the screen.' '
```

Conclusion

This documentation provides a comprehensive overview of the VBA Macro Analysis and Transformation Tool. By leveraging advanced technologies such as RAG, LLM, and vector databases, the tool automates the documentation and understanding of legacy VBA macros. It facilitates their transformation into

modern technologies and IT platforms by ensuring reliability, accuracy, and user-friendliness in documenting and generating VBA code. This makes it an effective solution for organizations looking to transition from legacy systems to more efficient and modern solutions.