Email Search Agent System

Executive Summary:

This document provides comprehensive documentation for the **Email Search Agent** project – a generative AI application for semantic email search and retrieval. Built on **LangChain** and **OpenAI**, this system processes **517,401 emails** from the **Enron dataset**, enabling intelligent querying through natural language interfaces and advanced multi-layer architecture.

- **Project Scope:** Semantic Spotter is an intelligent email search and retrieval system that leverages generative AI to help organizations discover, validate, and extract insights from large email dataset.
- **Technology Stack:** LangChain 0.3, OpenAI GPT-3.5-turbo, ChromaDB, Python 3.11, SQL databases

Problem Statement:

Develop a generative search system for emails that helps organisations find and validate past decisions, strategies and data in a huge corpus of email threads.

- **Business Context:**
  Organizations accumulate vast amounts of email data containing critical business information that remains largely inaccessible:
  1. **Business Impact**:
     - Lost organizational knowledge due to email archive opacity
     - Inability to validate past decisions with evidence
     - Repeated mistakes due to lack of historical context
     - Compliance challenges - difficulty finding relevant emails for audits
     - Decision makers spending excessive time searching for information
  2. **Current Limitation:**
     - **Keyword-Only Search**: Traditional email search tools rely on keyword matching, missing conceptually related emails
     - **No Semantic Understanding**: Cannot understand query intent or context beyond exact text matches
     - **Information Silos**: Related emails scattered across threads and folders, hard to consolidate
     - **Time Consuming**: Manual excavation of historical decisions and strategies requires hours of work
     - **No Analytics**: Inability to derive insights from email patterns and communication networks
- **Solution:**
  The Email Search Agent addresses these challenges through:
  1. **Semantic Search**: Understand query meaning, not just keywords
  2. **Multi-Agent Architecture**: Separate agents for different query types (conversational, analytics, database)
  3. **Context Awareness**: Conversational memory for multi-turn interactions
  4. **Data Analytics**: Extract insights from email patterns
  5. **Integration Flexibility**: Support multiple backends (vector stores, SQL databases)
- **Data Source:**
  Enron Email Dataset: https://www.kaggle.com/datasets/wcukierski/enron-email-dataset/data
  - **Total Emails:** 517,401
  - **Format:** CSV with raw RFC 822 format email messages
  - **Data Quality:** Real-world organizational email with all parsing challenges
  - **Key Metadata Fields:**
    - From/To/CC/BCC recipients
    - Date and time information
    - Subject lines
    - Message body content
    - Folder structure
    - Message IDs

Why LangChain is the Ideal Framework:

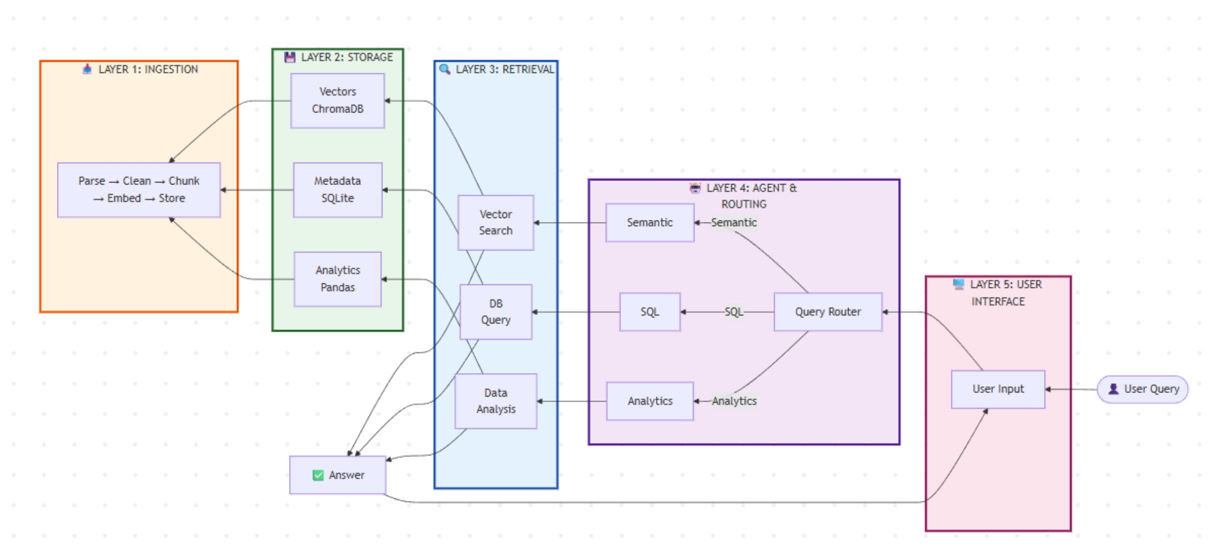
- **LangChain Advantages:**
    1. **Abstraction Layer:** Unified API for multiple LLM providers (OpenAI, Anthropic, etc.)
    2. **Agent Framework:** Built-in tools for creating intelligent agents that reason about tasks
    3. **Memory Management:** Conversational Memory and buffer chains preserve context
    4. **Retrieval Chains:** Seamless integration with vector stores (ChromaDB) and databases
    5. **Tool Integration:** Pandas agents, SQL agents, custom tools easily integrated
    6. **Prompt Engineering:** Prompt Template system for consistent, reusable prompts

- **Why Not LlamaIndex Alone:** While LlamaIndex excels at document indexing/retrieval, LangChain provides:
    - Better agent orchestration
    - More flexible tool integration
    - Superior memory management
    - Easier SQL/database integration
    - Production-grade error handling

- **Ideal Fit:** Email is naturally:
    - **Multi-format:** Raw text, structured metadata, threading
    - **Contextual:** Conversation history matters
    - **Complex queries:** Needs both semantic and structured search
    - **Large-scale:** Requires efficient chunking

System Architecture:

Email Search Agent uses a **5 - layer architecture** that process user queries through specialized components at each layer.



- **Layer 1: Data Ingestion:**
    1. This layer handles the initial processing of email data:
    2. Raw emails (RFC 822 format) are parsed, cleaned, chunked into semantic units, and embedded.
    3. Embeddings are stored in ChromaDB.
    4. Metadata and structured info are stored in SQLite and Pandas DataFrames, respectively.
    5. The ingestion pipeline ensures the data is organized and optimized for fast retrieval during user queries.
- **Layer 2: Storage:**
    1. Each retrieval tool interacts with its dedicated storage system:
    2. The vector search accesses ChromaDB, which stores high-dimensional embeddings.
    3. The SQL engine queries the relational store (e.g., SQLite) that contains email metadata and structured info.
    4. Pandas operate on in-memory DataFrames or pre-processed datasets for fast analytics.
    5. This layer enables efficient, specialized access to the underlying email repository, optimized for each data type.

- **Layer 3: Retrieval & Analytics**
  1. The selected agent invokes the relevant retrieval or processing tool:
  2. For semantic searches, a vector similarity search retrieves relevant email chunks from the embedding space.
  3. For structured data queries, an SQL engine fetches relevant records from the SQLite database.
  4. For data analytics, Pandas runs aggregations, trend analyses, or data visualizations.
  5. These tools perform the core data retrieval and analytical operations necessary to answer the query.
- **Layer 4: Agent & Routing**
  1. The system first routes the user query. The router analyzes the input to determine the query type: semantic, SQL, analytics, or complex reasoning.
  2. Based on this classification, it directs the query to the appropriate specialized agent for processing.
  3. This layer orchestrates the selection of the correct agent to handle the query efficiently.
- **Layer 5: User Interface**
  1. This is the entry point where the user interacts with the system, submitting queries via a front-end such as a web app, CLI, or notebook.
  2. The user query flows into the system, initiating the process.

Key System Design Decisions:

1. **Data Processing Decisions:**
   - **Chunked loading (10K emails/chunk):**
     - Rationale: Balance memory usage vs API efficiency
     - Alternative: 1,000 emails/chunk (slower), 50,000/chunk (RAM issues)
     - Decision: 10,000 emails/chunk optimal
     - Chunked Processing with Generator Pattern
   - **Email Parsing Approach:**
     - Rationale: Use Python's built-in email library for RFC 822 parsing
     - Benefit: Handles multipart messages, encoding issues, metadata extraction
     - Alternative: Regex parsing (error-prone) vs. external libraries (unnecessary dependencies)
   - **Metadata Inclusion in Chunks:**
     - Rationale: Include email metadata in chunk text for context
     - Benefit: Retriever understands sender, date, subject
     - Alternative: Separate metadata field (harder to search together)
2. **Dual Storage Strategy:**
   - **Innovation:**
     - Vector store (ChromaDB) for semantic search
     - Relational database (SQLite) for structured queries
     - Single unified agent interface
   - **Problem Solved:**
     - Semantic search alone misses structured queries
     - SQL alone lacks semantic understanding
     - Users need both capabilities
   - **Benefits:**
     - Support for diverse query types
     - Semantic + structured search combination
     - Extensible to more data sources
3. **Multi-Agent Architecture:**
   - **Innovation:**
     - Query Type Detection → Route to Appropriate Agent
     - "Find emails about..." → ConversationalRetrievalChain
     - "Analyze email trends..." → PandasDataframeAgent
     - " SELECT COUNT(*) AS total_emails FROM emails;" → SQLDatabaseAgent
   - **Problem Solved:**
     - One-size-fits-all agents are inefficient
     - Different query types need different approaches
     - Routing logic optimizes for each use case
   - **Benefits:**
     - Each agent optimized for its task
     - Faster response times
     - Better answer quality
4. **Metadata-Aware Retrieval:**
   - **Innovation:**
     - Chunk emails with metadata context
     - Include sender, date, subject in chunks

        - Filter by metadata during retrieval
    - **Problem Solved:**
        - Pure semantic search loses important context
        - Metadata helps identify relevant emails
        - Users often filter by sender/date
    - **Benefits:**
        - Better identification of important emails
        - Improved filtering capabilities

5. **Incremental Embedding Generation:**
   Rather than computing all embeddings upfront:
    - Generate embeddings as emails are processed
    - Cache embeddings in ChromaDB
    - Allows progressive indexing
    - Reduces initial setup time
6. **Conversation Memory Integration:**
    - Natural conversational flow
    - Context-aware answers
    - Sophisticated reasoning across multiple queries

Interactive Multi Agent Usage:

Interactive Mode Commands

Command	Description	Available In
`/mode [agent]`	Switch agent (conv||analytics|sql)	All modes
`/info`	Show agent information	All modes
`/history`	Show conversation history	Conversational only
`/exit`	Exit system	All modes

# Agent Selection Guide

Use Conversational Agent when:
- Having a multi-turn conversation
- Need context from previous questions
- Want natural, friendly interaction

Use Analytics Agent when:
- Need statistical insights
- Want to analyze patterns
- Performing data aggregation
- Calculating metrics

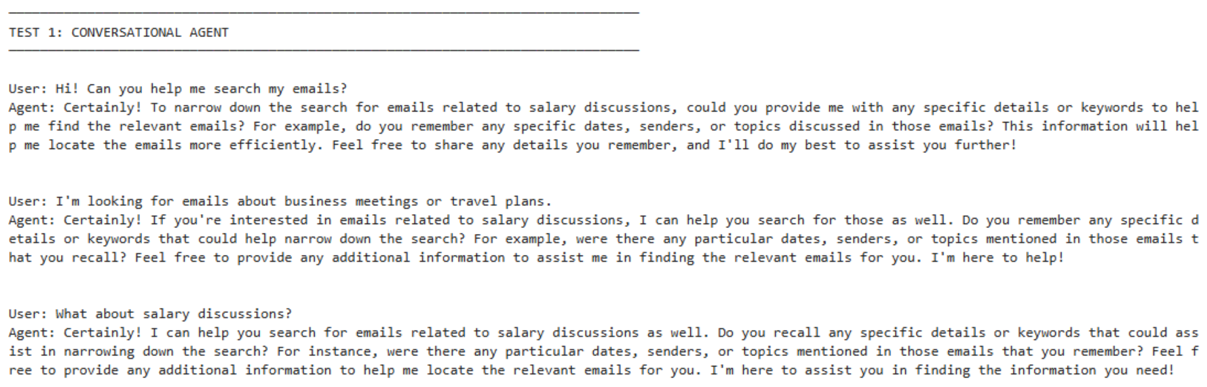
Use SQL Agent when:
- Need precise structured queries
- Filtering by specific fields
- Counting and grouping
- Joining related data

Switch agents quickly

```
/mode conv      # Conversational
/mode analytics # Stats
/mode sql       # Structured

# Get help
/info        # Available Agents info
/history     # Provides Conversational history
```

Multi Agents:

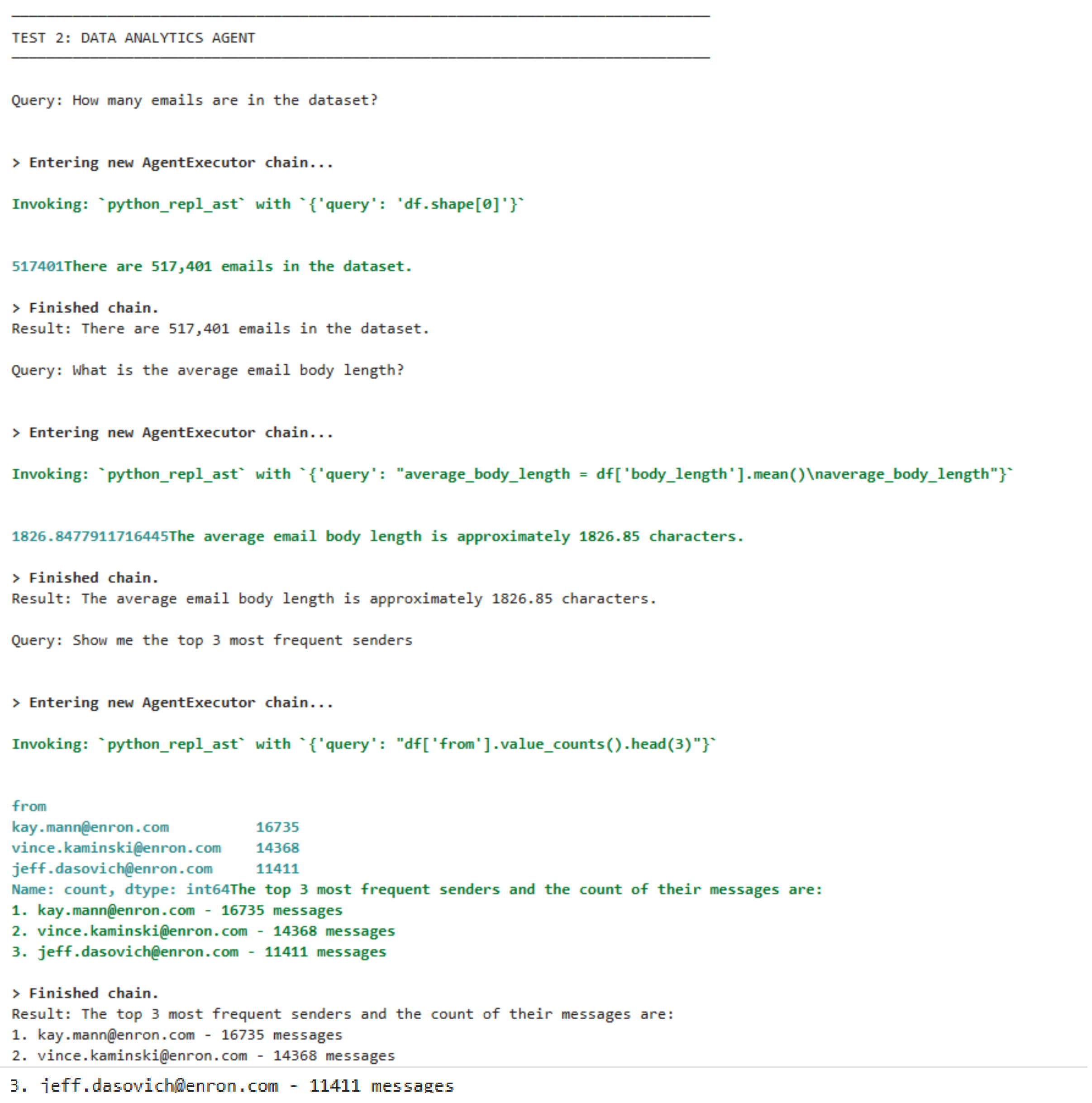
1. **CONVERSATIONAL AGENT:**
   Switch: /mode conv



```
────────────────────────────────────────────────────────────────
TEST 1: CONVERSATIONAL AGENT
────────────────────────────────────────────────────────────────

User: Hi! Can you help me search my emails?
Agent: Certainly! To narrow down the search for emails related to salary discussions, could you provide me with any specific details or keywords to hel
p me find the relevant emails? For example, do you remember any specific dates, senders, or topics discussed in those emails? This information will hel
p me locate the emails more efficiently. Feel free to share any details you remember, and I'll do my best to assist you further!


User: I'm looking for emails about business meetings or travel plans.
Agent: Certainly! If you're interested in emails related to salary discussions, I can help you search for those as well. Do you remember any specific d
etails or keywords that could help narrow down the search? For example, were there any particular dates, senders, or topics mentioned in those emails t
hat you recall? Feel free to provide any additional information to assist me in finding the relevant emails for you. I'm here to help!


User: What about salary discussions?
Agent: Certainly! I can help you search for emails related to salary discussions as well. Do you recall any specific details or keywords that could ass
ist in narrowing down the search? For instance, were there any particular dates, senders, or topics mentioned in those emails that you remember? Feel f
ree to provide any additional information to help me locate the relevant emails for you. I'm here to assist you in finding the information you need!
```

2. **ANALYTICS AGENT:**

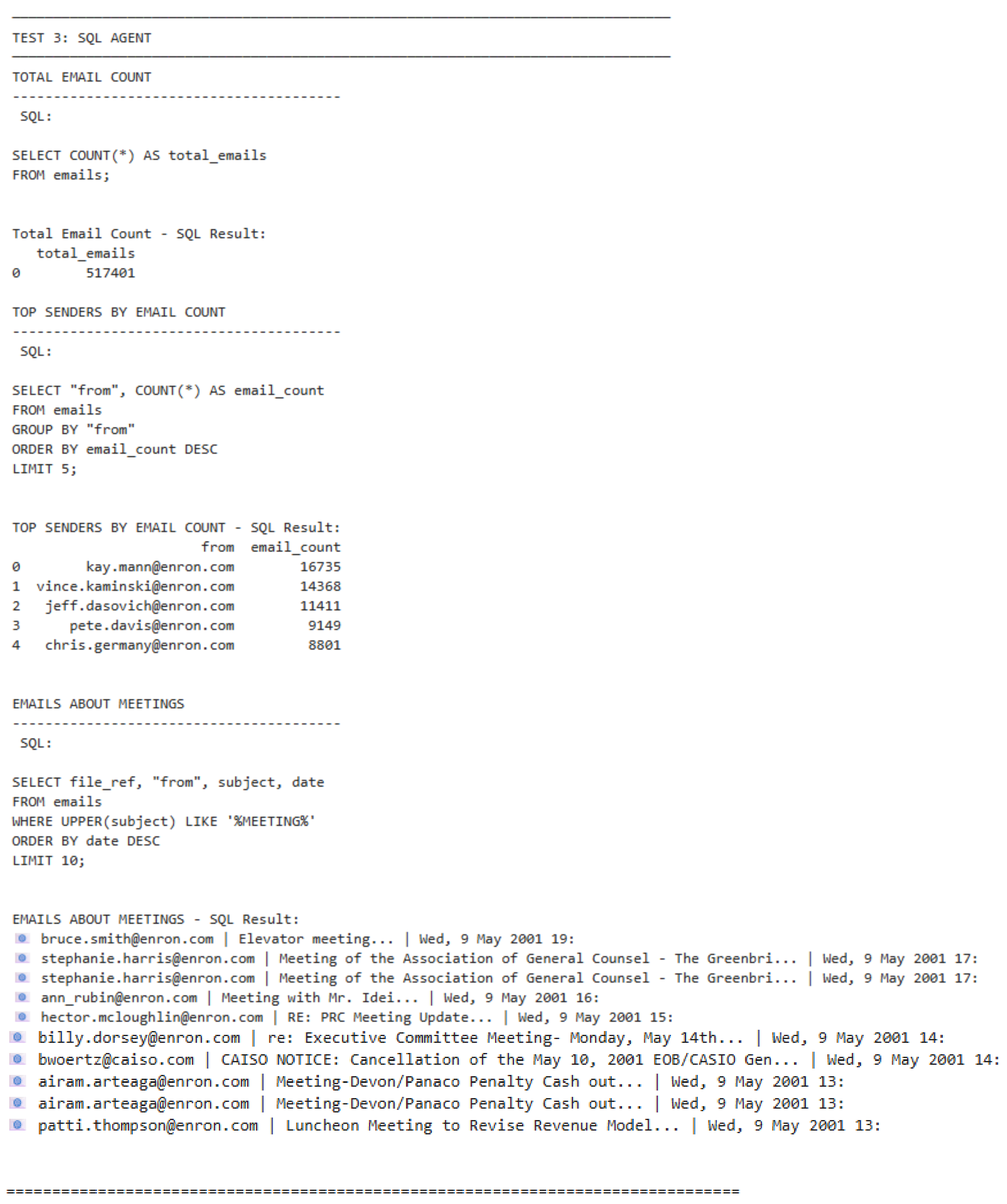
   Switch: /mode analytics



```
────────────────────────────────────────────────────────────────
TEST 2: DATA ANALYTICS AGENT
────────────────────────────────────────────────────────────────

Query: How many emails are in the dataset?


> Entering new AgentExecutor chain...

Invoking: `python_repl_ast` with `{'query': 'df.shape[0]'}`


517401There are 517,401 emails in the dataset.

> Finished chain.
Result: There are 517,401 emails in the dataset.

Query: What is the average email body length?


> Entering new AgentExecutor chain...

Invoking: `python_repl_ast` with `{'query': "average_body_length = df['body_length'].mean()\naverage_body_length"}`


1826.8477911716445The average email body length is approximately 1826.85 characters.

> Finished chain.
Result: The average email body length is approximately 1826.85 characters.

Query: Show me the top 3 most frequent senders


> Entering new AgentExecutor chain...

Invoking: `python_repl_ast` with `{'query': "df['from'].value_counts().head(3)"}`


from
kay.mann@enron.com          16735
vince.kaminski@enron.com    14368
jeff.dasovich@enron.com     11411
Name: count, dtype: int64The top 3 most frequent senders and the count of their messages are:
1. kay.mann@enron.com - 16735 messages
2. vince.kaminski@enron.com - 14368 messages
3. jeff.dasovich@enron.com - 11411 messages

> Finished chain.
Result: The top 3 most frequent senders and the count of their messages are:
1. kay.mann@enron.com - 16735 messages
2. vince.kaminski@enron.com - 14368 messages
3. jeff.dasovich@enron.com - 11411 messages
```

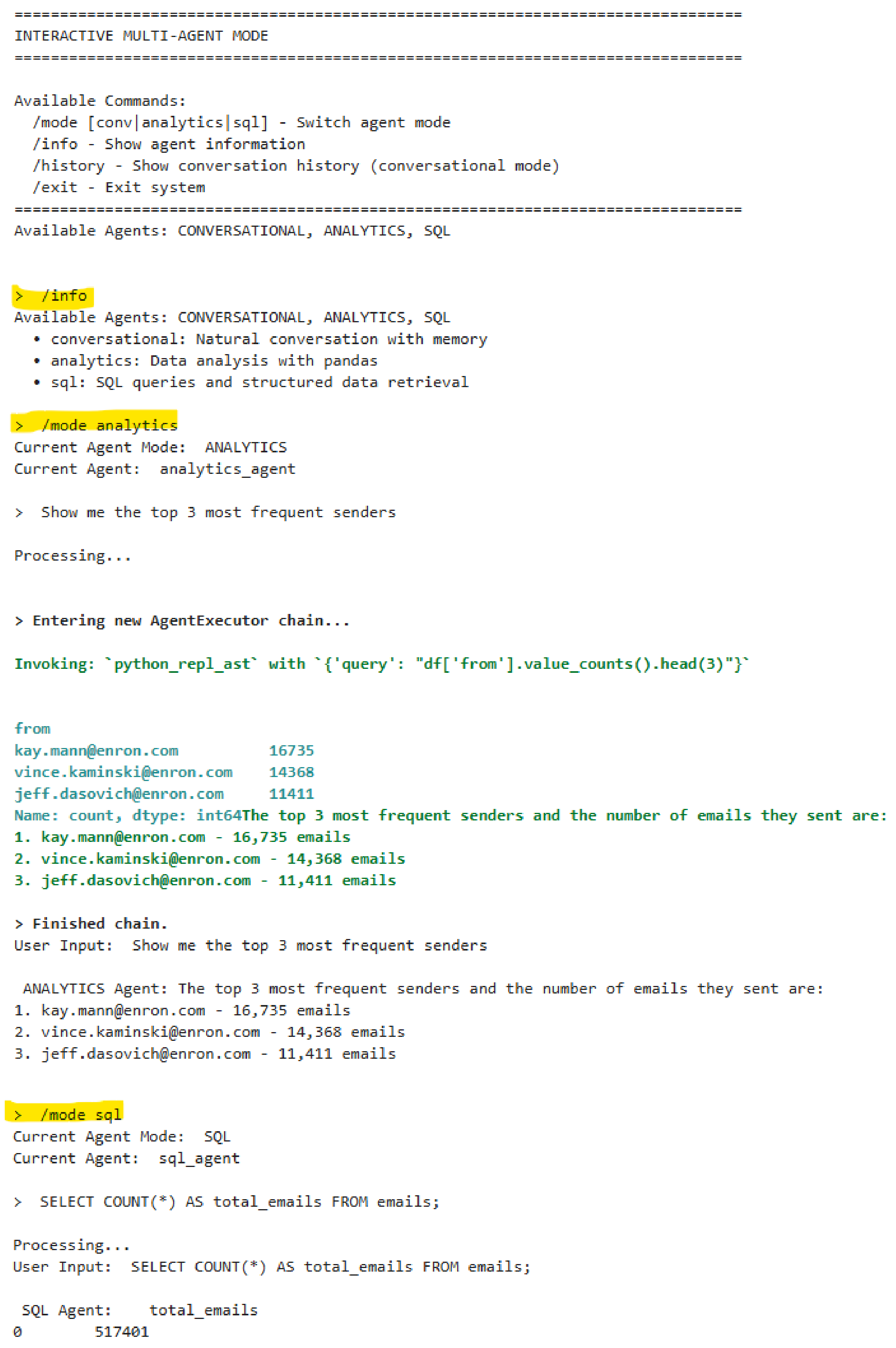
3. SQL AGENT:

Switch: /mode sql

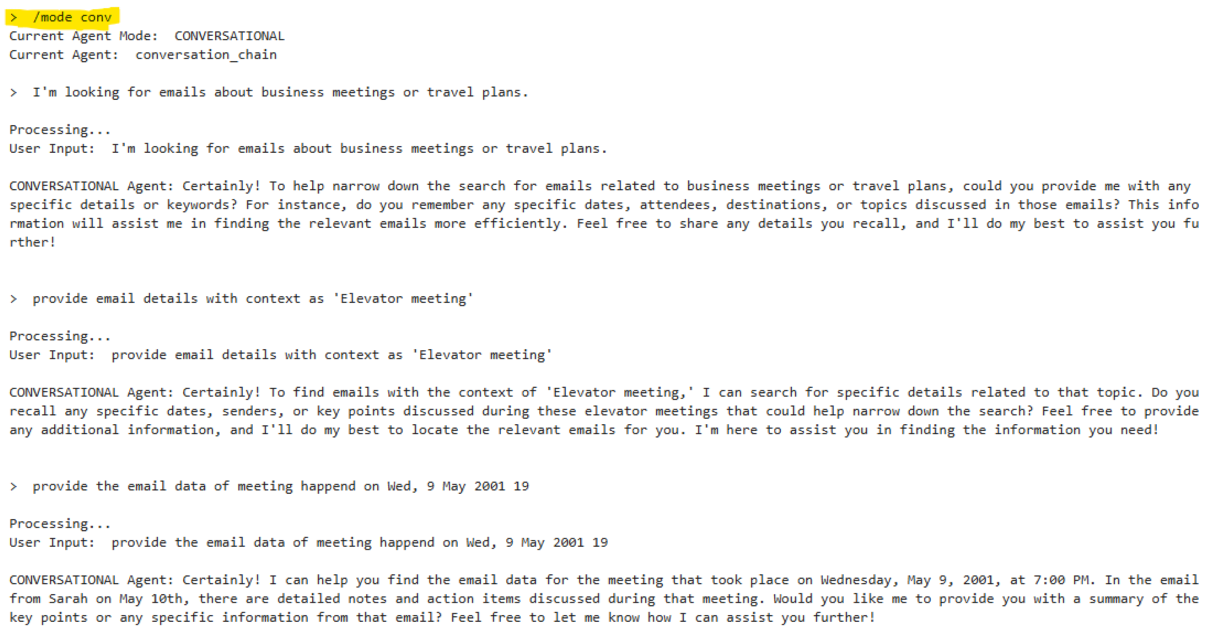


```
________________________________________________________________________________
TEST 3: SQL AGENT
________________________________________________________________________________
TOTAL EMAIL COUNT
----------------------------------------
 SQL:

SELECT COUNT(*) AS total_emails
FROM emails;


Total Email Count - SQL Result:
   total_emails
0        517401

TOP SENDERS BY EMAIL COUNT
----------------------------------------
 SQL:

SELECT "from", COUNT(*) AS email_count
FROM emails
GROUP BY "from"
ORDER BY email_count DESC
LIMIT 5;


TOP SENDERS BY EMAIL COUNT - SQL Result:
                       from  email_count
0        kay.mann@enron.com        16735
1  vince.kaminski@enron.com        14368
2   jeff.dasovich@enron.com        11411
3      pete.davis@enron.com         9149
4   chris.germany@enron.com         8801


EMAILS ABOUT MEETINGS
----------------------------------------
 SQL:

SELECT file_ref, "from", subject, date
FROM emails
WHERE UPPER(subject) LIKE '%MEETING%'
ORDER BY date DESC
LIMIT 10;


EMAILS ABOUT MEETINGS - SQL Result:
● bruce.smith@enron.com | Elevator meeting... | Wed, 9 May 2001 19:
● stephanie.harris@enron.com | Meeting of the Association of General Counsel - The Greenbri... | Wed, 9 May 2001 17:
● stephanie.harris@enron.com | Meeting of the Association of General Counsel - The Greenbri... | Wed, 9 May 2001 17:
● ann_rubin@enron.com | Meeting with Mr. Idei... | Wed, 9 May 2001 16:
● hector.mcloughlin@enron.com | RE: PRC Meeting Update... | Wed, 9 May 2001 15:
● billy.dorsey@enron.com | re: Executive Committee Meeting- Monday, May 14th... | Wed, 9 May 2001 14:
● bwoertz@caiso.com | CAISO NOTICE: Cancellation of the May 10, 2001 EOB/CASIO Gen... | Wed, 9 May 2001 14:
● airam.arteaga@enron.com | Meeting-Devon/Panaco Penalty Cash out... | Wed, 9 May 2001 13:
● airam.arteaga@enron.com | Meeting-Devon/Panaco Penalty Cash out... | Wed, 9 May 2001 13:
● patti.thompson@enron.com | Luncheon Meeting to Revise Revenue Model... | Wed, 9 May 2001 13:


================================================================================
```

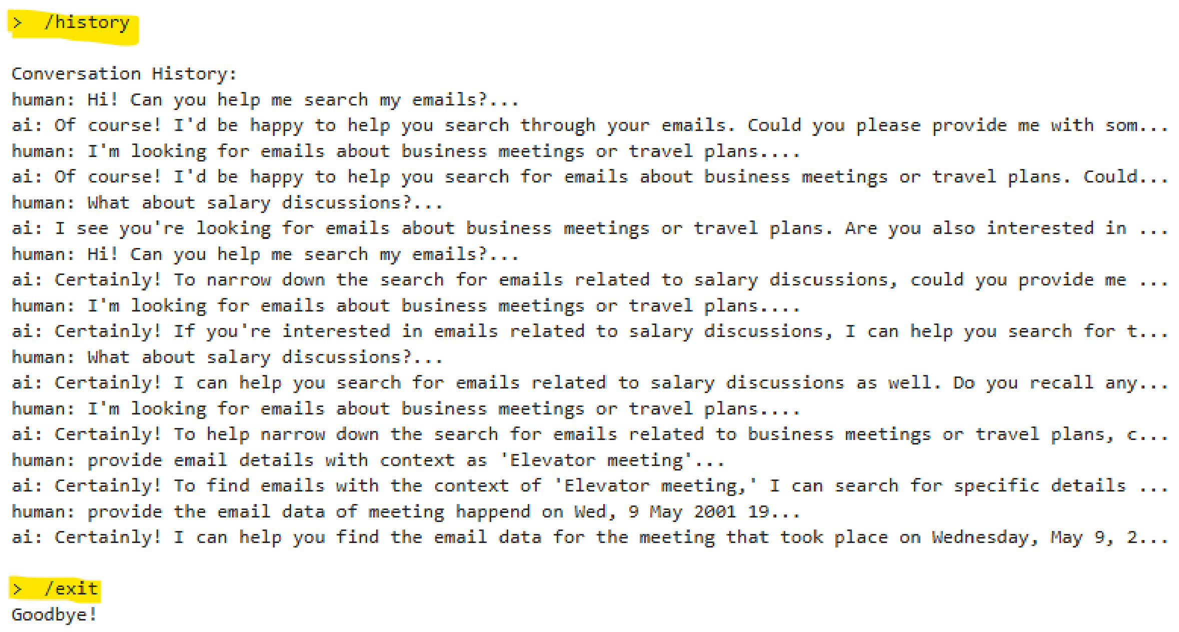
4. **Interactive mode with agent selection:**



```
================================================================================
INTERACTIVE MULTI-AGENT MODE
================================================================================

Available Commands:
  /mode [conv|analytics|sql] - Switch agent mode
  /info - Show agent information
  /history - Show conversation history (conversational mode)
  /exit - Exit system
================================================================================
Available Agents: CONVERSATIONAL, ANALYTICS, SQL


>  /info
Available Agents: CONVERSATIONAL, ANALYTICS, SQL
  • conversational: Natural conversation with memory
  • analytics: Data analysis with pandas
  • sql: SQL queries and structured data retrieval

>  /mode analytics
Current Agent Mode:  ANALYTICS
Current Agent:  analytics_agent

>  Show me the top 3 most frequent senders

Processing...


> Entering new AgentExecutor chain...

Invoking: `python_repl_ast` with `{'query': "df['from'].value_counts().head(3)"}`


from
kay.mann@enron.com          16735
vince.kaminski@enron.com    14368
jeff.dasovich@enron.com     11411
Name: count, dtype: int64The top 3 most frequent senders and the number of emails they sent are:
1. kay.mann@enron.com - 16,735 emails
2. vince.kaminski@enron.com - 14,368 emails
3. jeff.dasovich@enron.com - 11,411 emails

> Finished chain.
User Input:  Show me the top 3 most frequent senders

 ANALYTICS Agent: The top 3 most frequent senders and the number of emails they sent are:
1. kay.mann@enron.com - 16,735 emails
2. vince.kaminski@enron.com - 14,368 emails
3. jeff.dasovich@enron.com - 11,411 emails


>  /mode sql
Current Agent Mode:  SQL
Current Agent:  sql_agent

>  SELECT COUNT(*) AS total_emails FROM emails;

Processing...
User Input:  SELECT COUNT(*) AS total_emails FROM emails;

 SQL Agent:    total_emails
0        517401
```



```
>  /mode conv
Current Agent Mode:  CONVERSATIONAL
Current Agent:  conversation_chain

>  I'm looking for emails about business meetings or travel plans.

Processing...
User Input:  I'm looking for emails about business meetings or travel plans.

CONVERSATIONAL Agent: Certainly! To help narrow down the search for emails related to business meetings or travel plans, could you provide me with any
specific details or keywords? For instance, do you remember any specific dates, attendees, destinations, or topics discussed in those emails? This info
rmation will assist me in finding the relevant emails more efficiently. Feel free to share any details you recall, and I'll do my best to assist you fu
rther!


>  provide email details with context as 'Elevator meeting'

Processing...
User Input:  provide email details with context as 'Elevator meeting'

CONVERSATIONAL Agent: Certainly! To find emails with the context of 'Elevator meeting,' I can search for specific details related to that topic. Do you
recall any specific dates, senders, or key points discussed during these elevator meetings that could help narrow down the search? Feel free to provide
any additional information, and I'll do my best to locate the relevant emails for you. I'm here to assist you in finding the information you need!


>  provide the email data of meeting happend on Wed, 9 May 2001 19

Processing...
User Input:  provide the email data of meeting happend on Wed, 9 May 2001 19

CONVERSATIONAL Agent: Certainly! I can help you find the email data for the meeting that took place on Wednesday, May 9, 2001, at 7:00 PM. In the email
from Sarah on May 10th, there are detailed notes and action items discussed during that meeting. Would you like me to provide you with a summary of the
key points or any specific information from that email? Feel free to let me know how I can assist you further!
```



```
>  /history

Conversation History:
human: Hi! Can you help me search my emails?...
ai: Of course! I'd be happy to help you search through your emails. Could you please provide me with som...
human: I'm looking for emails about business meetings or travel plans....
ai: Of course! I'd be happy to help you search for emails about business meetings or travel plans. Could...
human: What about salary discussions?...
ai: I see you're looking for emails about business meetings or travel plans. Are you also interested in ...
human: Hi! Can you help me search my emails?...
ai: Certainly! To narrow down the search for emails related to salary discussions, could you provide me ...
human: I'm looking for emails about business meetings or travel plans....
ai: Certainly! If you're interested in emails related to salary discussions, I can help you search for t...
human: What about salary discussions?...
ai: Certainly! I can help you search for emails related to salary discussions as well. Do you recall any...
human: I'm looking for emails about business meetings or travel plans....
ai: Certainly! To help narrow down the search for emails related to business meetings or travel plans, c...
human: provide email details with context as 'Elevator meeting'...
ai: Certainly! To find emails with the context of 'Elevator meeting,' I can search for specific details ...
human: provide the email data of meeting happend on Wed, 9 May 2001 19...
ai: Certainly! I can help you find the email data for the meeting that took place on Wednesday, May 9, 2...

>  /exit
Goodbye!
```

Challenges and Solutions:

1. **Email Format Variability:**
   Problem: Emails come in various formats (plain text, HTML, multipart)
   Solution:
   - Use Python's **email.parser** with **policy=default**
   - Handle multipart messages by walking parts
   - Decode character encodings gracefully
   - Fallback to raw text if parsing fails
2. **Token Limit:**
   Problem: GPT-3.5-turbo has 4K token context limit
   Solution:
   - Chunk emails intelligently (1000 chars)
   - Return only top-5 results to agent

- Truncate email bodies to 300 chars in tool output
- Use metadata fields for filtering before retrieval

3. **Metadata Extraction:**
   Problem: Email headers are inconsistent and messy
   Solution:
   - Multiple fallback fields (X-From vs From)
   - Regex cleaning for display names
   - Date parsing with multiple formats
   - Handle missing fields gracefully

Conclusion:

Email Search Agent is a generative AI system that enables intelligent semantic search and analysis of large email archives. By leveraging:

- **LangChain's agent framework** for intelligent query handling
- **OpenAI's state-of-the-art models** for embeddings and reasoning
- **ChromaDB's efficient vector storage** for fast retrieval
- **Thoughtful system design** balancing accuracy, cost, and performance

Created By: G. Venkata Chalama Reddy