Fundamental_multi-agent_llamaIndex_framework_webinar_CFA_UK_v1

June 13, 2025

1 Fundamental Analysis using Muli-Agent System and Agentic RAG

1. Learn how to build your own **Multi-Agent** Fundamental Analysis Workflow with **LlamaIndex:**:

- Built a multi-agent system using LlamaIndex's AgentWorkflow to automate fundamental financial analysis.
- Agents include fundamental, profitability, liquidity, and supervisor agents — each analyzing specific financial ratios and passing results downstream.
- Each agent has a clear goal, prompt, tool, and a defined handoff to the next agent.
- Tools are used to gather and evaluate data against predefined thresholds, updating the system state.
- LLMs used include GPT-4 variants and Claude Sonnet, leveraging on function calling.

2. Learn how to build an **Agentic RAG System** to chat with Financial Reports such as 10k using **ReAct Agent**

- Built a ReAct agent using LlamaIndex's ReActAgent to answer financial comparison questions using retrieval + reasoning.
- Two Query Engines are used as tools: one for Apple's 2024 financials, the other for Nvidia's — each tied to their respective document indexes.
- Agent uses OpenAI's GPT-4o-mini, with tool-calling enabled via the QueryEngineTool, allowing dynamic retrieval before reasoning.
- Asked complex financial questions, like comparing 2024 vs. 2023 revenues, and Apple's vs. Nvidia's revenues.
- Agent follows ReAct paradigm: identifies tool use if needed, fetches relevant context, and responds with calculated insights (e.g., % increase, ratio comparisons).

2 Technical Stack

- LlamaIndex: https://docs.llamaindex.ai/en/stable/examples/agent/agent_workflow_basic/
- FinanceToolKit: https://github.com/JerBouma/FinanceToolkit

3 What Will you learn?

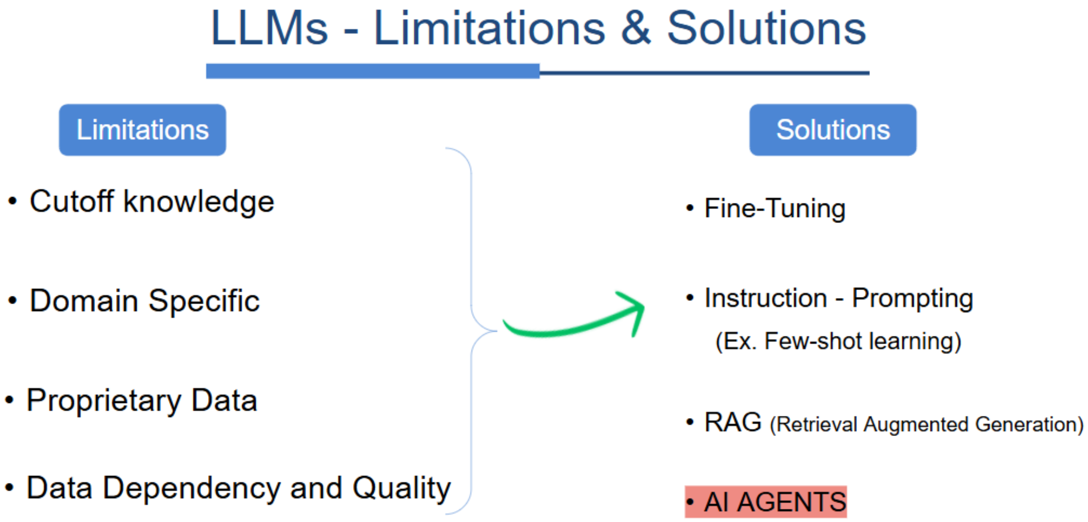
- LLMs limitations and solution
- Non Agentic vs Agentic Workflow
- Design Patterns
- Agents Architectures
- Reasoning Paradigm: ReAct
- Design a multi-agent system by calling external APIs
- Build a RAG ReAct agent to parse Financial reports and extract useful information.

4 Before You Practice – A Bit of Theory

4.1 LLMs limitations and solution

[10]:

```python
from IPython.display import Image, display
display(Image(filename=path+'/LLM_limitations_solutions.png'))
```



LLMs - Limitations & Solutions

Limitations

- Cutoff knowledge
- Domain Specific
- Proprietary Data
- Data Dependency and Quality

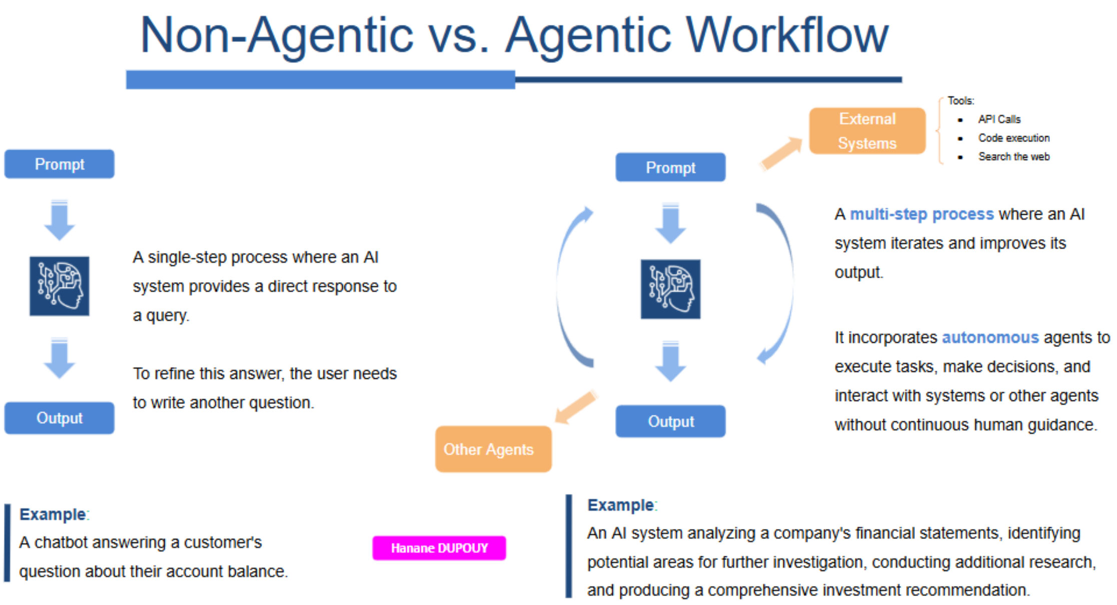
Solutions

- Fine-Tuning
- Instruction - Prompting
  (Ex. Few-shot learning)
- RAG (Retrieval Augmented Generation)
- AI AGENTS

4.2 Non Agentic vs Agentic Workflow

[11]:

```python
display(Image(filename=path+'/non_agentic_vs_agentic_workflow.png'))
```



Non-Agentic vs. Agentic Workflow

A single-step process where an AI system provides a direct response to a query.

To refine this answer, the user needs to write another question.

A **multi-step process** where an AI system iterates and improves its output.

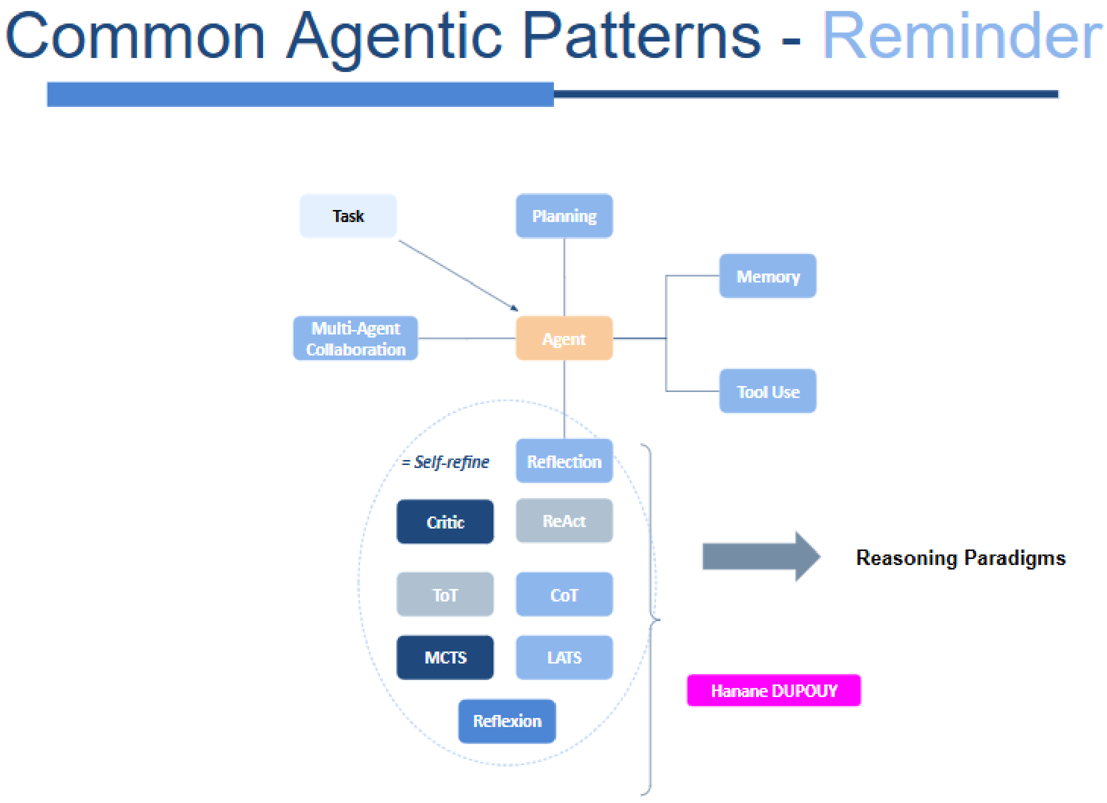
It incorporates **autonomous** agents to execute tasks, make decisions, and interact with systems or other agents without continuous human guidance.

Example:
A chatbot answering a customer's question about their account balance.

Example:
An AI system analyzing a company's financial statements, identifying potential areas for further investigation, conducting additional research, and producing a comprehensive investment recommendation.

4.3 Design Patterns

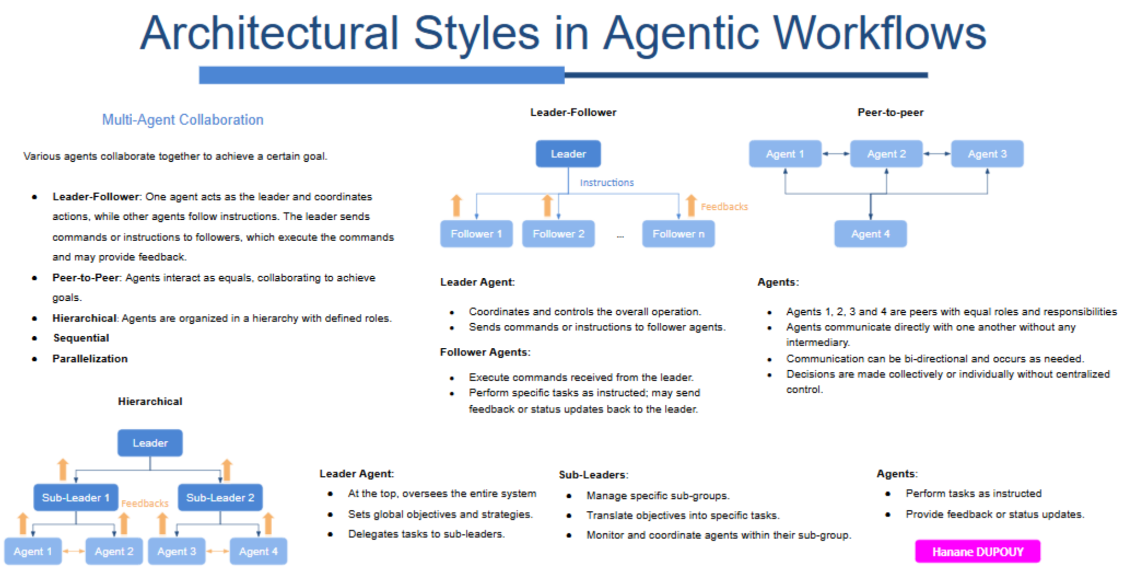
```
[12]: display(Image(filename=path+'/common_agentic_patterns.png'))
```



Common Agentic Patterns - Reminder

4.4 Agents Architectures

[13]:
```
display(Image(filename=path+'/architectural_styles.png'))
```



Architectural Styles in Agentic Workflows

Multi-Agent Collaboration

Various agents collaborate together to achieve a certain goal.

- **Leader-Follower**: One agent acts as the leader and coordinates actions, while other agents follow instructions. The leader sends commands or instructions to followers, which execute the commands and may provide feedback.
- **Peer-to-Peer**: Agents interact as equals, collaborating to achieve goals.
- **Hierarchical**: Agents are organized in a hierarchy with defined roles.
- **Sequential**
- **Parallelization**

Leader-Follower

Leader Agent:

- Coordinates and controls the overall operation.
- Sends commands or instructions to follower agents.

Follower Agents:

- Execute commands received from the leader.
- Perform specific tasks as instructed; may send feedback or status updates back to the leader.

Peer-to-peer

Agents:

- Agents 1, 2, 3 and 4 are peers with equal roles and responsibilities
- Agents communicate directly with one another without any intermediary.
- Communication can be bi-directional and occurs as needed.
- Decisions are made collectively or individually without centralized control.

Hierarchical

Leader Agent:

- At the top, oversees the entire system
- Sets global objectives and strategies.
- Delegates tasks to sub-leaders.

Sub-Leaders:

- Manage specific sub-groups.
- Translate objectives into specific tasks.
- Monitor and coordinate agents within their sub-group.

Agents:

- Perform tasks as instructed
- Provide feedback or status updates.

Hanane DUPOUY

5 1- Multi Agent System

5.1 Learn how to build your own Multi-Agent Fundamental Analysis Workflow with LlamaIndex

5.1.1 Multi-Agent

By leveraging on AgentWorflow class from LlamaIndex, we'll create a multi-agent system composed of:

- A fundamental_agent: Responsible for collecting ratios for a given ticker using the Finance-Toolkit package
- A profitability_agent: Responsible for gathering profitability ratios (ROA, ROE, Net Profit Margin, and Gross Margin) and analyzing their strengths or weaknesses compared to given thresholds. For example, if ROA > 5%, the firm is in a healthy range.
- A liquidity_agent: This agent collects liquidity ratios for a given ticker (Current Ratio, Quick Ratio, Debt-to-Equity Ratio, Interest Coverage Ratio) and comments on these ratios given a set of threshold values. For example if Current Ratio is between 1.5-3.0 the firm is in a healthy range.
- A supervisor_agent: This agent provides a final comment on the overall health of the firm based on the different comments coming from the different agents.

I've limited my agents to 2 specialized ones, you can add as much as you need.

5.1.2 Agent's Breakdown:

Then for each agent:

- A clear description of the goal of the agent was provided
- A system prompt of the agent's instructions
- A Tool (you can also provide several tools) that helps gathering the data needed and compare it to the set of the pre-defined thresholds.
- The next agent to take the handoff.

5.1.3 Tools

There are 4 tools, each agent has a tool. Each tool can act and modify the State of the multi-agent system.

5.1.4 AgentWorkflow

Finally, the AgentWorkflow class puts all these together:

- You list the agents to use
- You specify the agent root
- You initiate your state

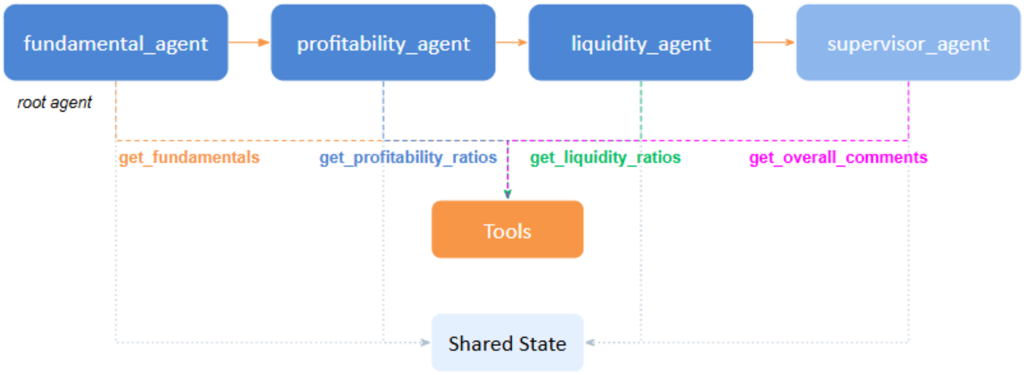
5.1.5 LLMs :

I used GPT-4o, GPT-4.1 and GPT-4.5-Preview. I also used Claude Sonnet 4 and 3.7. (I've used function calling LLMs because these agents supports natively tool calling).

5.1.6 Key Takeaways :

- The workflow is easy to put in place
- The agent's breakdown is well-defined, from its description to the system prompt and the next agents for handoff.
- I called LLMs inside of the tools to provide insights on the ratios compared to the thresholds. Otherwise, I faced sometimes some hallucination…
- The final answer from GPT-4.5-preview is well-structured compared to GPT-4o, providing key ratios, values, comments, and overall insights.
- Claude 3.7 Sonnet provided even better well designed answers thant GPTs.

[4]:
```
from IPython.display import Image, display
display(Image(filename=path+'/fundamental_multi_agent_system.png'))
```



5.2 Set Up

[ ]:
```
%pip install llama-index -q
%pip install financetoolkit -q
%pip install llama-index-llms-anthropic -q
```

[ ]:
```
import warnings
warnings.filterwarnings("ignore")
```

[ ]:
```
from google.colab import userdata
FINANCIAL_MODELING_PREP_API_KEY = userdata.
  ↪get('FINANCIAL_MODELING_PREP_API_KEY')
OPENAI_API_KEY = userdata.get('OPENAI_API_KEY')

from financetoolkit import Toolkit
import pandas as pd

from llama_index.llms.openai import OpenAI

LLM_4o = OpenAI(model="gpt-4o", api_key=OPENAI_API_KEY)
LLM_41 = OpenAI(model="gpt-4.1", api_key=OPENAI_API_KEY)
LLM_45 = OpenAI(model="gpt-4.5-preview", api_key=OPENAI_API_KEY)
```

[ ]:
```
from llama_index.llms.anthropic import Anthropic

CLAUDE_API_KEY = userdata.get('CLAUDE_API_KEY')
LLM_CLAUDE4 = Anthropic(model="claude-sonnet-4-0", api_key=CLAUDE_API_KEY)
```

[ ]:
```
LLM_CLAUDE37 = Anthropic(model="claude-3-7-sonnet-latest",␣
  ↪api_key=CLAUDE_API_KEY)
```

[ ]:
```
LLM_CLAUDE37_TOK = Anthropic(model="claude-3-7-sonnet-latest", max_tokens =␣
 ↪64000, api_key=CLAUDE_API_KEY)
```

5.3 MULTI-AGENT SYSTEM: How to build it?

5.3.1 TOOLS

Prompts and general functions

[ ]:
```
PROMPT_THRESHOLD_VERIFICATION = """
  Analyze the financial health of the firm {ticker} based on its profitability␣
 ↪ratios.

  ### Given:
  - Profitability or Liquidity Ratios for {ticker}:
    {dico_ratios}

  - Thresholds for Financial Health Evaluation:
    {thresholds_to_respect}

  ### Task:
  For each ratio, follow these steps:
  1  Assign a score from 1 to 10 (where 1 = very unhealthy, 10 = very healthy).
  2  Provide a justification explaining why the ratio received that score.
  3  Give an overall insight on the firm’s financial health, summarizing␣
 ↪strengths and weaknesses based on the individual ratio scores.

  Ensure the analysis is detailed, data-driven, and easy to interpret.
  """

THRESHOLDS_TO_RESPECT_PROFITABILITY = """
  ## Thresholds to respect for firm's financial health:
  ## Profitability Ratios (Assessing Earnings & Efficiency)
  |Ratio|        Healthy|        Moderate|        Weak|
  |-----|        -------|        --------|        ----|
  |Return on Assets (ROA)         |> 5%|        2% - 5%        |< 2%|
  |Return on Equity (ROE)|        > 15%        |8% - 15%        |< 8%|
  |Net Profit Margin        |> 10%        |5% - 10%        |< 5%|
  |Gross Profit Margin        |> 40%        |20% - 40%        |< 20%|
  """

THRESHOLDS_TO_RESPECT_LIQUIDITY = """
  ## Thresholds to respect for firm's financial health:
  ## Liquidity & Solvency Ratios (Assessing FinancAial Stability)
  | Ratio        | Healthy Range          | Warning Zone          | Risky/
 ↪Dangerous   |
  | ----        | ------------          | -----------          | --------------- ␣
 ↪|
 | Current Ratio          |  1.5 - 3.0          |  < 1.0          | > 3.0 (excess␣
↪cash)   |
 | Quick Ratio          | > 1.0          | < 1.0 |          - |
 | Debt-to-Equity (D/E) | 0.3 - 1.5          | > 2.0          | < 0.3␣
↪(under-leveraged) |
 | Interest Coverage  | > 3.0          | 1.5 - 3.0          | < 1.5 (high risk) |
 """

def get_completion_profitability(ticker, dico_ratios_profitability,␣
↪llm_for_complete):
  prompt = PROMPT_THRESHOLD_VERIFICATION.format(ticker=ticker,␣
↪dico_ratios=dico_ratios_profitability,␣
↪thresholds_to_respect=THRESHOLDS_TO_RESPECT_PROFITABILITY)
  resp = llm_for_complete.complete(prompt)
  return resp

def get_completion_liquidity(ticker, dico_ratios_liquidity, llm_for_complete):
  prompt = PROMPT_THRESHOLD_VERIFICATION.format(ticker=ticker,␣
↪dico_ratios=dico_ratios_liquidity,␣
↪thresholds_to_respect=THRESHOLDS_TO_RESPECT_LIQUIDITY)
  resp = llm_for_complete.complete(prompt)
  return resp
```

Tools

[ ]:

```
from llama_index.core.workflow import Context

async def get_fundamentals(ctx: Context, ticker: str) ->pd.DataFrame():
  """ Get the different fundamental ratios for a given ticker. """
  companies = Toolkit(
      [ticker], api_key=FINANCIAL_MODELING_PREP_API_KEY, start_date="2022-01-01"
  )
  ratios = companies.ratios.collect_all_ratios()

  print("ratios", ratios.loc['Return on Assets'])

  current_state = await ctx.get("state")

  if current_state["ticker"] == "":
    current_state["ticker"] = ticker

  if current_state["ratios"].empty:
    current_state["ratios"] = ratios

  await ctx.set("state", current_state)

  return f"Ratios extracted for {ticker}."
```

```
async def get_profitability_ratios(ctx: Context, llm_for_complete=LLM_4o):
  """Get profitability ratios for a given ticker: ROA, ROE, Net Profit Margin␣
 ↪and Gross Margin, and comments on these ratios given a set of threshold␣
 ↪values."""

  current_state = await ctx.get("state")
  ratios = current_state['ratios']
  ticker = current_state['ticker']

  ROA = ratios.loc['Return on Assets']
  ROE = ratios.loc['Return on Equity']
  net_profit_margin = ratios.loc['Net Profit Margin']
  gross_margin = ratios.loc['Gross Margin']

  # print("## Profitability Ratios (Assessing Earnings & Efficiency)")
  # print("Return on Assets (ROA):", ROA.index[-1], ROA.iloc[-1])
  # print("Return on Equity (ROE):", ROE.index[-1], ROE.iloc[-1])
  # print("Net Profit Margin:", net_profit_margin.index[-1], net_profit_margin.
 ↪iloc[-1])
  # print("Gross Margin:", gross_margin.index[-1], gross_margin.iloc[-1])

  roa_values = [ROA.index[-1], ROA.iloc[-1]]
  roe_values = [ROE.index[-1], ROE.iloc[-1]]
  net_profit_margin_values = [net_profit_margin.index[-1], net_profit_margin.
 ↪iloc[-1]]
  gross_margin_values = [gross_margin.index[-1], gross_margin.iloc[-1]]

  dico_ratios_profitability = {
      "Return on Assets (ROA)": roa_values,
      "Return on Equity (ROE)": roe_values,
      "Net Profit Margin": net_profit_margin_values,
      "Gross Margin": gross_margin_values
  }

  #Need to add conditions whether ratios, roa, roe, net profit margin, gross␣
 ↪margin are empty or not
  if current_state["profitability_ratios"] == {}:
    current_state["profitability_ratios"] = dico_ratios_profitability

  resp = get_completion_profitability(ticker, dico_ratios_profitability,␣
 ↪llm_for_complete)
  current_state['threshold_profitability_comments'] = resp

  await ctx.set("state", current_state)
  return "Profitability ratios extracted and Comments performed: " + str(resp)


async def get_liquidity_ratios(ctx: Context, llm_for_complete=LLM_4o):
  """Get liquidity ratios for a given ticker: Current Ratio, Quick Ratio,␣
 ↪Debt-to-Equity Ratio, Interest Coverage Ratio and comments on these ratios␣
 ↪given a set of threshold values."""

  current_state = await ctx.get("state")
  ratios = current_state['ratios']
  ticker = current_state['ticker']

  current_ratio = ratios.loc['Current Ratio']
  quick_ratio = ratios.loc['Quick Ratio']
  debt_to_equity_ratio = ratios.loc['Debt-to-Equity Ratio']
  interest_coverage_ratio = ratios.loc['Interest Coverage Ratio']

  # print("## Profitability Ratios (Assessing Earnings & Efficiency)")
  # print("Current Ratio:", current_ratio.index[-1], current_ratio.iloc[-1])
  # print("Quick Ratio:", quick_ratio.index[-1], quick_ratio.iloc[-1])
  # print("Debt to Equity Ratio:", debt_to_equity_ratio.index[-1],␣
 ↪debt_to_equity_ratio.iloc[-1])
  # print("Interest Coverage Ratio:", interest_coverage_ratio.index[-1],␣
 ↪interest_coverage_ratio.iloc[-1])

  current_ratio_values = [current_ratio.index[-1], current_ratio.iloc[-1]]
  quick_ratio_values = [quick_ratio.index[-1], quick_ratio.iloc[-1]]
  debt_to_equity_ratio_values = [debt_to_equity_ratio.index[-1],␣
 ↪debt_to_equity_ratio.iloc[-1]]
  interest_coverage_ratio_values = [interest_coverage_ratio.index[-1],␣
 ↪interest_coverage_ratio.iloc[-1]]

  dico_ratios_liquidity = {
      "Current Ratio": current_ratio_values,
      "Quick Ratio": quick_ratio_values,
      "Debt to Equity Ratio": debt_to_equity_ratio_values,
      "Interest Coverage Ratio": interest_coverage_ratio_values
  }

  #Need to add conditions whether ratios, roa, roe, net profit margin, gross␣
 ↪margin are empty or not
  if current_state["liquidity_ratios"] == {}:
    current_state["liquidity_ratios"] = dico_ratios_liquidity

  resp = get_completion_liquidity(ticker, dico_ratios_liquidity,␣
 ↪llm_for_complete)
```

```
  current_state['threshold_liquidity_comments'] = resp

  await ctx.set("state", current_state)
  return "Liquidity ratios extracted and Comments performed: " + str(resp)

async def get_overall_comments(ctx: Context):
  """Get comments on diffrent type of ratios. The overall_comments are given by␣
 ↪the SupervisedAgent based on threshold_profitability_comments and␣
 ↪threshold_liquidity_comments."""

  current_state = await ctx.get("state")
  profitability_comments = current_state['threshold_profitability_comments']
  liquidity_comments = current_state['threshold_liquidity_comments']

  if profitability_comments is None:
    return "No profitability comments found."

  if liquidity_comments is None:
    return "No liquidity comments found."

  if ~ profitability_comments is None and ~ liquidity_comments is None:
    overall_comments = profitability_comments + "\n" + liquidity_comments
    current_state['overall_comments'] = overall_comments

  return "Overall comments done."
```

5.3.2 AGENTS

[ ]:

```
from llama_index.core.agent.workflow import FunctionAgent

fundamental_agent = FunctionAgent(
    name="FundamentalAgent",
    description="Get various fundamental ratios for a given ticker.",
    system_prompt=(
        "You are the fundament analyst that can extract different fundamental␣
 ↪ratios for a given ticker. "
        "Once you have extracted the fundamental financial ratios, you should␣
 ↪hand off control to the ProfitabilityAgent to extract the profitability␣
 ↪ratios or/and the LiquidityAgent to extract the liquidity ratios."

    ),
    llm=LLM_4o,
    tools=[get_fundamentals],
    can_handoff_to=["ProfitabilityAgent"],
)

profitability_agent = FunctionAgent(
    name="ProfitabilityAgent",
    description="Collect profitability ratios for a given ticker: ROA, ROE, Net␣
 ↪Profit Margin and Gross Margin and Comment on the results given a set of␣
 ↪threshold values.",
    system_prompt=(
        """You are the ProfitabilityAgent that can collect profitability ratios␣
 ↪(profitability_ratios) on a given ticker.
        You collect these ratios from the FundamentalAgent.
        Once these ratios are collected in profitability_ratios, you should␣
 ↪comment on these ratios based on the thresholds values provided in␣
 ↪get_profitability_ratios.
        These comments must be included in threshold_profitability_comments. At␣
 ↪the end provide ONLY these comments included in␣
 ↪threshold_profitability_comments. DO NOT ADD anything else.
        Once the comments are done, you should hand off control to the␣
 ↪LiquidityAgent.
        """
    ),
    llm=LLM_4o,
    tools=[get_profitability_ratios],
    can_handoff_to=["LiquidityAgent"],
)

liquidity_agent = FunctionAgent(
    name="LiquidityAgent",
    description="Collect liquidity ratios for a given ticker: Current Ratio,␣
 ↪Quick Ratio, Debt-to-Equity Ratio, Interest Coverage Ratio and comments on␣
 ↪these ratios given a set of threshold values.",
    system_prompt=(
        """You are the LiquidityAgent that can collect liquidity ratios␣
 ↪(liquidity_ratios) on a given ticker.
        You collect these ratios from the FundamentalAgent.
        Once these ratios are collected in liquidity_ratios, you should comment␣
 ↪on these ratios based on the thresholds values provided in␣
 ↪get_liquidity_ratios.
        These comments must be included in threshold_liquidity_comments. At the␣
 ↪end provide ONLY these comments included in threshold_liquidity_comments. DO␣
 ↪NOT ADD anything else.
        Once the comments are done, you should hand off control to the␣
 ↪SupervisorAgent.
        """
    ),
    llm=LLM_4o,
    tools=[get_liquidity_ratios],
    can_handoff_to=["SupervisorAgent"],
)
```

```
supervisor_agent = FunctionAgent(
    name="SupervisorAgent",
    description="Provide an overall comment based on the comments coming from␣
 ↪the ProfitabilityAgent and LiquidityAgent and stored on overall_comments.",
    system_prompt=(
        "You are an fundament analyst expert and supervisor. You collect␣
 ↪comments coming from various agents such as ProfitabilityAgent and␣
 ↪LiquidityAgent. "
        "If you collect only one or no comments, go back to ProfitabilityAgent␣
 ↪and LiquidityAgent until you get the desired feedbacks on the ratios."
        "If you don't collect threshold_liquidity_comments, you should hand off␣
 ↪control to the LiquidityAgent to gather these comments."
        "If you don't collect threshold_profitability_comments, you should hand␣
 ↪off control to the ProfitabilityAgent to gather these comments."
        "Based on the results in the comments coming from the␣
 ↪ProfitabilityAgent and LiquidityAgent, provide an overall comment on the␣
 ↪health of the firm."
        "Justify your comment with details and data. "
    ),
    llm=LLM_4o,
    tools=[get_overall_comments],
    can_handoff_to=["FundamentalAgent","ProfitabilityAgent","LiquidityAgent"],
)
```

5.3.3 AGENT WORKFLOW

[ ]:

```
from llama_index.core.agent.workflow import AgentWorkflow

agent_workflow = AgentWorkflow(
    agents=[fundamental_agent, profitability_agent,liquidity_agent,␣
 ↪supervisor_agent],
    root_agent=fundamental_agent.name,
    initial_state={
        "ratios": pd.DataFrame(),
        "profitability_ratios": {},
        "liquidity_ratios": {},
        "threshold_profitability_comments": None,
        "threshold_liquidity_comments": None,
        "overall_comments": None,
        "ticker": "",
    },

)
```

5.4 LLM: GPT-4o

5.4.1 Calling the Mult-Agent System for a specific task

Run 1

[ ]:

```
from llama_index.core.agent.workflow import (
    AgentInput,
    AgentOutput,
    ToolCall,
    ToolCallResult,
    AgentStream,
)

handler = agent_workflow.run(
    user_msg=(
        "Provide the fundamental analysis of Apple and comments on the␣
  ↪financial health of the company."
    )
)

current_agent = None
current_tool_calls = ""
async for event in handler.stream_events():
    if (
        hasattr(event, "current_agent_name")
        and event.current_agent_name != current_agent
    ):
        current_agent = event.current_agent_name
        print(f"\n{'='*50}")
        print(f"  Agent: {current_agent}")
        print(f"{'='*50}\n")

    elif isinstance(event, AgentOutput):
        if event.response.content:
            print("  Output:", event.response.content)
        if event.tool_calls:
            print(
                "   Planning to use tools:",
                [call.tool_name for call in event.tool_calls],
            )
    elif isinstance(event, ToolCallResult):
        print(f"  Tool Result ({event.tool_name}):")
        print(f"   Arguments: {event.tool_kwargs}")
        print(f"   Output: {event.tool_output}")
    elif isinstance(event, ToolCall):
        print(f"  Calling Tool: {event.tool_name}")
        print(f"   With arguments: {event.tool_kwargs}")
```

```
==================================================
 Agent: FundamentalAgent
==================================================

  Planning to use tools: ['get_fundamentals']

Obtaining financial statements: 100%|      | 3/3 [00:01<00:00,  2.60it/s]
Obtaining historical data: 100%|     | 2/2 [00:00<00:00,  9.64it/s]

ratios 2022      NaN
2023    0.275
2024   0.2613
Freq: Y-DEC, Name: Return on Assets, dtype: float64
 Calling Tool: get_fundamentals
  With arguments: {'ticker': 'AAPL'}
 Tool Result (get_fundamentals):
  Arguments: {'ticker': 'AAPL'}
  Output: Ratios extracted for AAPL.
   Planning to use tools: ['handoff']
 Calling Tool: handoff
  With arguments: {'to_agent': 'ProfitabilityAgent', 'reason': 'Collect
profitability ratios for AAPL and comment on the results given a set of
threshold values.'}
 Tool Result (handoff):
  Arguments: {'to_agent': 'ProfitabilityAgent', 'reason': 'Collect profitability
ratios for AAPL and comment on the results given a set of threshold values.'}
  Output: Agent ProfitabilityAgent is now handling the request due to the
following reason: Collect profitability ratios for AAPL and comment on the
results given a set of threshold values..
Please continue with the current request.

==================================================
 Agent: ProfitabilityAgent
==================================================

  Planning to use tools: ['get_profitability_ratios']
 Calling Tool: get_profitability_ratios
  With arguments: {}
 Tool Result (get_profitability_ratios):
  Arguments: {}
  Output: Profitability ratios extracted and Comments performed: To analyze the
financial health of Apple Inc. (AAPL) based on its profitability ratios, we will
evaluate each ratio against the provided thresholds, assign a score from 1 to
10, and provide justifications for each score. Finally, we will summarize the
overall financial health of the firm.

### 1 Return on Assets (ROA)
```

```
- **Value:** 26.13%
- **Thresholds:**
  - Healthy: > 5%
  - Moderate: 2% - 5%
  - Weak: < 2%

**Score:** 10/10

**Justification:** Apple's ROA of 26.13% is significantly above the healthy
threshold of 5%. This indicates that the company is highly efficient in using
its assets to generate earnings. A high ROA suggests strong operational
efficiency and effective asset management, which is a positive indicator of
financial health.

### 2  Return on Equity (ROE)

- **Value:** 157.41%
- **Thresholds:**
  - Healthy: > 15%
  - Moderate: 8% - 15%
  - Weak: < 8%

**Score:** 10/10

**Justification:** With an ROE of 157.41%, Apple far exceeds the healthy
threshold of 15%. This exceptionally high ROE indicates that the company is
generating substantial profits from its shareholders' equity, reflecting strong
financial performance and effective use of equity capital. Such a high ROE is
indicative of a company with a competitive advantage and strong profitability.

### 3  Net Profit Margin

- **Value:** 23.97%
- **Thresholds:**
  - Healthy: > 10%
  - Moderate: 5% - 10%
  - Weak: < 5%

**Score:** 10/10

**Justification:** Apple's net profit margin of 23.97% is well above the healthy
threshold of 10%. This high margin indicates that the company retains a
significant portion of its revenue as profit after all expenses, showcasing
strong cost management and pricing power. A high net profit margin is a sign of
a financially robust company.

### 4  Gross Margin
```

```
- **Value:** 46.21%
- **Thresholds:**
  - Healthy: > 40%
  - Moderate: 20% - 40%
  - Weak: < 20%

**Score:** 10/10

**Justification:** The gross margin of 46.21% is above the healthy threshold of
40%, indicating that Apple is effectively managing its production costs relative
to its sales. A high gross margin suggests that the company has a strong
competitive position, allowing it to maintain pricing power and cost efficiency.

### Overall Insight on Financial Health

Apple Inc. demonstrates exceptional financial health based on its profitability
ratios. Each ratio significantly exceeds the healthy thresholds, indicating
strong operational efficiency, effective asset and equity management, and robust
cost control. The high scores across all ratios reflect a company with a strong
competitive advantage, capable of generating substantial profits and maintaining
financial stability. Apple's financial strengths include its ability to
efficiently utilize assets and equity, maintain high profit margins, and manage
production costs effectively. There are no apparent weaknesses in the
profitability ratios, suggesting that Apple is in a very healthy financial
position.
   Planning to use tools: ['handoff']
  Calling Tool: handoff
  With arguments: {'to_agent': 'LiquidityAgent', 'reason': "Profitability
analysis for AAPL is complete. Now, it's time to collect liquidity ratios and
provide comments on them."}
  Tool Result (handoff):
  Arguments: {'to_agent': 'LiquidityAgent', 'reason': "Profitability analysis
for AAPL is complete. Now, it's time to collect liquidity ratios and provide
comments on them."}
  Output: Agent LiquidityAgent is now handling the request due to the following
reason: Profitability analysis for AAPL is complete. Now, it's time to collect
liquidity ratios and provide comments on them..
Please continue with the current request.

==================================================
  Agent: LiquidityAgent
==================================================

   Planning to use tools: ['get_liquidity_ratios']
  Calling Tool: get_liquidity_ratios
  With arguments: {}
  Tool Result (get_liquidity_ratios):
  Arguments: {}
  Output: Liquidity ratios extracted and Comments performed: To analyze the
financial health of Apple Inc. (AAPL) based on the provided profitability
ratios, we will evaluate each ratio against the given thresholds, assign a score
from 1 to 10, and provide a justification for each score. Finally, we will
summarize the overall financial health of the firm.

### 1. Current Ratio
- **Value:** 0.8673
- **Healthy Range:** 1.5 - 3.0
- **Evaluation:** The current ratio of 0.8673 is below the healthy range,
indicating that Apple may not have enough current assets to cover its current
liabilities. This places the company in the warning zone, suggesting potential
liquidity issues.
- **Score:** 3
- **Justification:** The score reflects the fact that the current ratio is below
1.0, indicating a potential liquidity concern. However, Apple is a large,
established company with significant cash flows, which may mitigate some of the
risks associated with a low current ratio.

### 2. Quick Ratio
- **Value:** 0.5589
- **Healthy Range:** > 1.0
- **Evaluation:** The quick ratio is also below the healthy threshold,
indicating that Apple may not have sufficient liquid assets to meet its short-
term obligations without relying on inventory sales.
- **Score:** 2
- **Justification:** A quick ratio below 1.0 suggests that Apple might face
challenges in meeting its short-term liabilities with its most liquid assets.
The score reflects the potential risk, although Apple's strong market position
and cash flow generation capabilities may provide some buffer.

### 3. Debt-to-Equity Ratio
- **Value:** 2.0906
- **Healthy Range:** 0.3 - 1.5
- **Evaluation:** The debt-to-equity ratio is above the healthy range and falls
into the warning zone. This indicates that Apple is using a higher level of debt
financing compared to equity, which could increase financial risk.
- **Score:** 4
- **Justification:** The score reflects the elevated financial leverage, which
could pose risks if not managed properly. However, Apple's ability to generate
substantial earnings and cash flow may help mitigate these risks.

### 4. Interest Coverage Ratio
- **Value:** Infinity (inf)
- **Healthy Range:** > 3.0
- **Evaluation:** An infinite interest coverage ratio suggests that Apple has no
difficulty covering its interest expenses, indicating strong financial health in
terms of its ability to meet debt obligations.
```

```
- **Score:** 10
- **Justification:** The infinite interest coverage ratio is a strong indicator
of Apple's ability to comfortably cover its interest expenses, reflecting
excellent financial health in this aspect.

### Overall Insight on Apple’s Financial Health

**Strengths:**
- **Interest Coverage:** Apple demonstrates exceptional strength in its ability
to cover interest expenses, indicating robust earnings relative to its interest
obligations.

**Weaknesses:**
- **Liquidity Ratios (Current and Quick Ratios):** Both ratios are below the
healthy thresholds, suggesting potential liquidity concerns. However, Apple's
strong cash flow and market position may alleviate some of these concerns.
- **Debt-to-Equity Ratio:** The elevated ratio indicates higher financial
leverage, which could pose risks if not managed carefully.

**Summary:** While Apple shows strong financial health in terms of its ability
to cover interest expenses, the low liquidity ratios and high debt-to-equity
ratio suggest areas of potential concern. However, given Apple's strong market
position, brand strength, and cash flow generation capabilities, the company is
likely well-positioned to manage these risks effectively. Overall, Apple's
financial health appears solid, but monitoring liquidity and leverage levels
will be important moving forward.
   Planning to use tools: ['handoff']
  Calling Tool: handoff
  With arguments: {'to_agent': 'SupervisorAgent', 'reason': "Liquidity analysis
for AAPL is complete. Now, it's time to provide an overall comment based on the
profitability and liquidity comments."}
  Tool Result (handoff):
  Arguments: {'to_agent': 'SupervisorAgent', 'reason': "Liquidity analysis for
AAPL is complete. Now, it's time to provide an overall comment based on the
profitability and liquidity comments."}
  Output: Agent SupervisorAgent is now handling the request due to the following
reason: Liquidity analysis for AAPL is complete. Now, it's time to provide an
overall comment based on the profitability and liquidity comments..
Please continue with the current request.

==================================================
  Agent: SupervisorAgent
==================================================

   Planning to use tools: ['get_overall_comments']
  Calling Tool: get_overall_comments
  With arguments: {'overall_comments': "Apple Inc. (AAPL) demonstrates
exceptional financial health based on its profitability ratios, with all metrics
significantly exceeding healthy thresholds. This indicates strong operational
efficiency, effective asset and equity management, and robust cost control.
However, liquidity ratios reveal potential concerns, with both current and quick
ratios below healthy levels, suggesting possible liquidity issues. The debt-to-
equity ratio is also elevated, indicating higher financial leverage. Despite
these concerns, Apple's strong cash flow, market position, and infinite interest
coverage ratio suggest the company is well-positioned to manage these risks
effectively. Overall, Apple's financial health is solid, but monitoring
liquidity and leverage will be important moving forward."}
 Tool Result (get_overall_comments):
  Arguments: {'overall_comments': "Apple Inc. (AAPL) demonstrates exceptional
financial health based on its profitability ratios, with all metrics
significantly exceeding healthy thresholds. This indicates strong operational
efficiency, effective asset and equity management, and robust cost control.
However, liquidity ratios reveal potential concerns, with both current and quick
ratios below healthy levels, suggesting possible liquidity issues. The debt-to-
equity ratio is also elevated, indicating higher financial leverage. Despite
these concerns, Apple's strong cash flow, market position, and infinite interest
coverage ratio suggest the company is well-positioned to manage these risks
effectively. Overall, Apple's financial health is solid, but monitoring
liquidity and leverage will be important moving forward."}
  Output: bad operand type for unary ~: 'CompletionResponse'
 Output: Based on the analysis of Apple Inc.'s (AAPL) financial health, the
company exhibits exceptional profitability, with all profitability ratios
significantly exceeding healthy thresholds. This indicates strong operational
efficiency, effective asset and equity management, and robust cost control.
However, there are potential concerns regarding liquidity, as both the current
and quick ratios are below healthy levels, suggesting possible liquidity issues.
Additionally, the debt-to-equity ratio is elevated, indicating higher financial
leverage.

Despite these concerns, Apple's strong cash flow, market position, and infinite
interest coverage ratio suggest that the company is well-positioned to manage
these risks effectively. Overall, Apple's financial health is solid, but it will
be important to monitor liquidity and leverage levels moving forward to ensure
continued financial stability.
```

Output: Based on the analysis of Apple Inc.'s (AAPL) financial health, the company exhibits exceptional profitability, with all profitability ratios significantly exceeding healthy thresholds. This indicates strong operational efficiency, effective asset and equity management, and robust cost control. However, there are potential concerns regarding liquidity, as both the current and quick ratios are below healthy levels, suggesting possible liquidity issues. Additionally, the debt-to-equity ratio is elevated, indicating higher financial leverage.

Despite these concerns, Apple's strong cash flow, market position, and infinite interest coverage ratio suggest that the company is well-positioned to manage these risks effectively. Overall, Apple's financial health is solid, but it will be important to monitor liquidity and leverage levels moving forward to ensure continued financial stability.