### **LinkedIn Search Agent (Search/Linkedin) Script Analysis**

This table breaks down the files and functions responsible for finding LinkedIn profiles and inferring sales-related insights about them.

| **File** | **Function/Method Name** | **Input Parameters** | **Output [Datatype]** | **Comments** |
| --- | --- | --- | --- | --- |
| **linkedin\_agent.py** | search\_agent (variable) | model, name, description, instruction, tools | Agent instance | Defines the core AI agent. It is configured with a detailed prompt instructing the Gemini model on how to use Google Search to find LinkedIn profiles and infer the top 3 job priorities for a sales context. |
| **linkedin\_serp.py** | infer\_priorities | title: str | list | A placeholder function that returns a hardcoded list of priorities. This is likely for testing or as a fallback. |
| **linkedin\_serp.py** | search\_linkedin\_serpapi | name: str | dict | **(Alternative Method)** A non-agent function that directly calls the SerpApi service to get Google search results for LinkedIn profiles. It formats the results but does not use a generative model to infer priorities beyond the placeholder. |
| **linkedin\_agent\_runner.py** | get\_linkedin\_p | user\_name: str, runner, user\_id, session\_id | dict or list | An asynchronous function that executes the search\_agent. It takes a person's name, runs the agent, captures the final text response, cleans it of any markdown formatting, and parses it into a Python object. |
| **linkedin\_agent\_runner.py** | get\_linkedin | user\_name: str | dict, list, or None | **The main entry point for the agent.** This is a synchronous wrapper that makes it easy to call the async get\_linkedin\_p function from a standard Python script. This is the function that the Streamlit UI would call. |