```
import subprocess
from typing import Any, Dict, List, Callable

from loguru import logger
from pydantic import BaseModel

# Setting up loguru logging
logger.add("agent_metadata.log", rotation="10 MB")

try:
    import pandas as pd
except ImportError:
    logger.error("Failed to import pandas")
    subprocess.run(["pip", "install", "pandas"])
    import pandas as pd


def display_agents_info(agents: List[Callable]) -> pd.DataFrame:
    """
    Displays information about all agents in a list using a DataFrame.

    :param agents: List of callable functions that return Agent instances.
    """
    # Extracting relevant information from each agent
    agent_data = []
    for agent_callable in agents:
```

```
    try:
        agent = agent_callable()
        agent_info = {
            "ID": agent.id,
            "Name": agent.agent_name,
            "Description": agent.description,
            "max_loops": agent.max_loops,
            # "Docs": agent.docs,
            "System Prompt": agent.system_prompt,
            "LLM Model": agent.llm.model_name,  # type: ignore
        }
        agent_data.append(agent_info)
    except AttributeError as e:
        logger.error(
            f"Failed to extract information from agent: {e}"
        )
        continue

# Creating a DataFrame to display the data
try:
    df = pd.DataFrame(agent_data)
except Exception as e:
    logger.error(f"Failed to create DataFrame: {e}")
    return

# Displaying the DataFrame
```

```
    try:
        print(df)

        return df
    except Exception as e:
        logger.error(f"Failed to print DataFrame: {e}")


def dict_to_dataframe(data: Dict[str, Any]) -> pd.DataFrame:
    """
    Converts a dictionary into a pandas DataFrame.

    :param data: Dictionary to convert.
    :return: A pandas DataFrame representation of the dictionary.
    """
    # Convert dictionary to DataFrame
    df = pd.json_normalize(data)
    return df


def pydantic_model_to_dataframe(model: BaseModel) -> pd.DataFrame:
    """
    Converts a Pydantic Base Model into a pandas DataFrame.

    :param model: Pydantic Base Model to convert.
    :return: A pandas DataFrame representation of the Pydantic model.
```

"""

Convert Pydantic model to dictionary

model_dict = model.dict()

Convert dictionary to DataFrame

df = dict_to_dataframe(model_dict)

return df