```
import os

from swarms.utils.pandas_utils import (
    display_agents_info,
    dict_to_dataframe,
    pydantic_model_to_dataframe,
)
from swarms import Agent
from swarm_models import OpenAIChat

# Create an instance of the OpenAIChat class
llm = OpenAIChat(
    api_key=os.getenv("OPENAI_API_KEY"),
    model_name="gpt-4o-mini",
    temperature=0.1,
)

# Initialize the director agent
# Initialize the director agent
director = Agent(
    agent_name="Director",
    system_prompt="Directs the tasks for the accountants",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
```

```
    saved_state_path="director.json",
)

# Initialize accountant 1
accountant1 = Agent(
    agent_name="Accountant1",
    system_prompt="Prepares financial statements",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
    saved_state_path="accountant1.json",
)

# Initialize accountant 2
accountant2 = Agent(
    agent_name="Accountant2",
    system_prompt="Audits financial records",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
    saved_state_path="accountant2.json",
)

# Initialize 8 more specialized agents
```

```
balance_sheet_analyzer = Agent(
    agent_name="BalanceSheetAnalyzer",
    system_prompt="Analyzes balance sheets",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
    saved_state_path="balance_sheet_analyzer.json",
)

income_statement_analyzer = Agent(
    agent_name="IncomeStatementAnalyzer",
    system_prompt="Analyzes income statements",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
    saved_state_path="income_statement_analyzer.json",
)

cash_flow_analyzer = Agent(
    agent_name="CashFlowAnalyzer",
    system_prompt="Analyzes cash flow statements",
    llm=llm,
    max_loops=1,
    dashboard=False,
```

```
    state_save_file_type="json",
    saved_state_path="cash_flow_analyzer.json",
)

financial_ratio_calculator = Agent(
    agent_name="FinancialRatioCalculator",
    system_prompt="Calculates financial ratios",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
    saved_state_path="financial_ratio_calculator.json",
)

tax_preparer = Agent(
    agent_name="TaxPreparer",
    system_prompt="Prepares tax returns",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
    saved_state_path="tax_preparer.json",
)

payroll_processor = Agent(
    agent_name="PayrollProcessor",
```

```
    system_prompt="Processes payroll",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
    saved_state_path="payroll_processor.json",
)

inventory_manager = Agent(
    agent_name="InventoryManager",
    system_prompt="Manages inventory",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
    saved_state_path="inventory_manager.json",
)

budget_planner = Agent(
    agent_name="BudgetPlanner",
    system_prompt="Plans budgets",
    llm=llm,
    max_loops=1,
    dashboard=False,
    state_save_file_type="json",
    saved_state_path="budget_planner.json",
```

```
)



agents = [
    director,
    accountant1,
    accountant2,
    balance_sheet_analyzer,
    income_statement_analyzer,
    cash_flow_analyzer,
    financial_ratio_calculator,
    tax_preparer,
    payroll_processor,
    inventory_manager,
    budget_planner,
]

out = display_agents_info(agents)
print(out)



# Dict to DataFrame
data_dict = director.agent_output.model_dump()
print(data_dict)

df = dict_to_dataframe(data_dict)
```

```
print(df)


# Pydantic Model to DataFrame

df = pydantic_model_to_dataframe(director.agent_output)

print(df)
```