

✓ 미국 주식 데이터 분석

```
import sys
sys.path.append('/content/drive/MyDrive/20240921')
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import sys
sys.path.append('/content/drive/MyDrive/20240921')
```

```
from dataloader import TickerSaver
```

↻ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
from google.colab import drive
drive.mount('/content/drive')
```

↻ Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

✓ 필요 라이브러리 설치하기

```
# !apt-get install -y libta-lib0 libta-lib-dev
```

```
# !pip install TA-Lib
```

```
# Colab 환경에 필요한 종속성 및 TA-Lib 라이브러리 설치
!wget http://prdownloads.sourceforge.net/ta-lib/ta-lib-0.4.0-src.tar.gz
!tar -xzf ta-lib-0.4.0-src.tar.gz
!cd ta-lib/ && ./configure --prefix=/usr && make && make install
```

↻

```

make[2]: Entering directory '/content/ta-lib/src/tools'
make[3]: Entering directory '/content/ta-lib/src/tools'
make[3]: Nothing to be done for 'install-exec-am'.
make[3]: Nothing to be done for 'install-data-am'.
make[3]: Leaving directory '/content/ta-lib/src/tools'
make[2]: Leaving directory '/content/ta-lib/src/tools'
make[1]: Leaving directory '/content/ta-lib/src/tools'
make[1]: Entering directory '/content/ta-lib'
make[2]: Entering directory '/content/ta-lib'
test -z "/usr/bin" || /usr/bin/mkdir -p "/usr/bin"
/usr/bin/install -c 'ta-lib-config' '/usr/bin/ta-lib-config'
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/content/ta-lib'
make[1]: Leaving directory '/content/ta-lib'

```

```
!pip install TA-Lib
```



Requirement already satisfied: TA-Lib in /usr/local/lib/python3.10/dist-packages (0.4.32)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from TA-Lib) (1.26.4)

```
!pip install yfinance pandas numpy pandas-ta TA-Lib matplotlib yahoo_fin requests_html tqdm langchain langchain_openai langchain_experimental langcha
```

✓ 주식 데이터 가져와 저장하기

- nasdaq의 주식을 모두 가져옴
- 가져온 데이터는 data 폴더 내에 csv 별로 저장
- dow도 가져오고 싶다면 dow=True로 설정

```

from dataloader import TickerSaver

ts = TickerSaver(nasdaq=True, dow=False)
ts.save_all_tickers_data()

```

✓ 저장한 주식 데이터 가져오기

- min_days을 사용하여 최소 며칠 이상의 데이터가 존재하는 종목만 가져오도록 설정

```

from dataloader import TickerLoader
tl = TickerLoader()
data = tl.get_data(min_days=2500)

```



100%|██████████| 4387/4387 [03:07<00:00, 23.35it/s]

주어진 min_days 파라미터로 최소 2500일 이상의 데이터를 요청하는 의미

```
data
```

```
data.head(10)
```



	Date	Open	High	Low	Close	Adj Close	Volume	SMA_5	SMA_10	SMA_20	...	RSI_14	MACD	MACD_signal	MACD_hist	BB.
199	2005-05-09	13.02	13.02	12.95	13.01	10.602542	81000	13.008	12.797	12.6770	...	57.048809	0.168867	0.126225	0.042641	13.
200	2005-05-10	13.05	13.09	12.94	13.01	10.602542	114900	13.010	12.893	12.6775	...	57.048809	0.171195	0.135219	0.035976	13.
201	2005-05-11	12.90	12.99	12.46	12.64	10.340750	51000	12.936	12.950	12.6930	...	49.409079	0.141553	0.136486	0.005067	13.
202	2005-05-12	12.86	12.99	12.72	12.91	10.561631	63100	12.916	12.954	12.7150	...	54.226269	0.138254	0.136840	0.001415	13.
203	2005-05-13	12.82	13.00	12.82	12.98	10.618900	41200	12.910	12.947	12.7345	...	55.411657	0.139678	0.137407	0.002271	13.
204	2005-05-16	12.80	13.07	12.80	13.01	10.643442	69800	12.910	12.959	12.7360	...	55.938311	0.141596	0.138245	0.003350	13.
205	2005-05-17	13.00	13.00	12.88	12.96	10.602541	41900	12.900	12.955	12.7410	...	54.777056	0.137495	0.138095	-0.000600	13.
206	2005-05-18	13.01	13.28	13.00	13.15	10.757975	97800	13.002	12.969	12.7745	...	58.318100	0.147873	0.140051	0.007822	13.
207	2005-05-19	13.05	13.45	13.05	13.32	10.897054	154800	13.084	13.000	12.7990	...	61.242329	0.167879	0.145616	0.022263	13.
208	2005-05-20	13.25	13.40	13.25	13.40	10.962500	33400	13.168	13.039	12.8755	...	62.573009	0.188022	0.154098	0.033925	13.

10 rows x 23 columns

data.columns



```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume', 'SMA_5',
      'SMA_10', 'SMA_20', 'SMA_50', 'SMA_100', 'SMA_200', 'RSI_14', 'MACD',
      'MACD_signal', 'MACD_hist', 'BB_upper', 'BB_middle', 'BB_lower',
      'STOCH_k', 'STOCH_d', 'ticker'],
      dtype='object')
```

✓ ChatGPT를 이용한 데이터 분석

```
from langchain.agents.agent_types import AgentType
from langchain_openai import ChatOpenAI
from langchain_experimental.agents import create_pandas_dataframe_agent
import os
# from langchain_experimental.agents.agent_toolkits import create_pandas_dataframe_agent
```

더블클릭 또는 Enter 키를 눌러 수정

여기서도 직접 챗지피티 불러와서 사용 가능함

- **LangChain**: LangChain은 자연어 처리(NLP) 및 대규모 언어 모델(LLM)을 활용하여 응용 프로그램을 만들고, 데이터 파이프라인을 구축하는 Python 라이브러리입니다. 특히, LangChain은 언어 모델을 중심으로 한 애플리케이션을 쉽게 개발할 수 있도록 다양한 도구, 템플릿, 모듈을 제공합니다.

```
import langchain
langchain.__version__
```



'0.3.0'

- 발급받은 API KEY 입력

```
# os.environ["OPENAI_API_KEY"] = input()
os.environ["OPENAI_API_KEY"] = 'sk-hKP41kytQ0TLs2Dwhqd1qDkb0NhhGwFcmIUYPSHl1ET3B1bkFJBkAt7jSjCQMYb_XBW0GR0nQCgWh3SWL5hLs6UYt ioA'
```

- 사용할 수 있는 모델들: gpt-3.5-turbo, gpt-3.5-turbo-16k, gpt-4, gpt-4-32k 등이 있으며, 토큰 수가 더 많이 제공되는 gpt-3.5-turbo-16k와 gpt-4-32k 모델의 경우 답변이 길어져서 잘리는 경우가 생길 때 사용하면 좋습니다.
- 모델 확인 링크: <https://platform.openai.com/docs/models>
- 데이터분석에서는 창의성보다는 정확성이 중요하므로 temperature를 0으로 설정.

```
data_backup = data.copy()
```

```
model = ChatOpenAI(model="gpt-4o-mini", temperature=0)
agent = create_pandas_dataframe_agent(
    model,
    data,
    verbose=True,
    agent_type='openai-tools',
    allow_dangerous_code=True,
    max_iterations=10
)
```

- **temperature=0**: 텍스트 생성에서의 랜덤성을 제어하는 매개변수입니다. temperature=0은 모델이 가장 확실한(결정론적인) 응답을 하도록 만듭니다. 높은 값을 주면 더 창의적이거나 무작위적인 응답이 생성될 수 있습니다. 0에서 1 사이의 값을 넣어주면 됨.
- **verbose=True**: 에이전트가 수행하는 과정을 자세하게 출력합니다. 디버깅 또는 실행 상태를 추적할 때 유용합니다. 감시하는 과정!

```
data.head(2)
```

- **agent.invoke()**: agent.invoke()는 LangChain 또는 유사한 에이전트 기반 시스템에서 에이전트가 특정 작업을 수행하도록 호출하는 함수입니다. 이 함수는 에이전트에게 명령을 내리거나, 요청한 작업을 실행하게 합니다. 여기서 agent는 Pandas 데이터프레임 에이전트처럼 특정 작업을 수행할 수 있는 에이전트 객체를 의미합니다.

코딩을 시작하거나 AI로 코드를 생성하세요.

```
agent.invoke("""
ticker 컬럼의 unique한 값의 개수를 알려줘
""") # 2500일이라는 제한을 뒀기 때문에 4000개 넘는 데이터지만, 1616개만 데이터 가져옴.
```

```
# nunique : unique 한 n개 개수
```

```
WARNING: langchain_core.callbacks.manager:Error in StdOutCallbackHandler.on_chain_start callback: AttributeError("'NoneType' object has no attribute 'nunique'")
Invoking: `python_repl_ast` with `{'query': 'unique_tickers_count = df['ticker'].nunique()#nunique_tickers_count'}`

1616`ticker` 컬럼의 unique한 값의 개수는 1616개입니다.

> Finished chain.
{'input': 'Wnticker 컬럼의 unique한 값의 개수를 알려줘\n',
 'output': '`ticker` 컬럼의 unique한 값의 개수는 1616개입니다.'}
```

```
agent.invoke("""
ticker 컬럼의 unique한 값의 개수를 알려줘
""")

WARNING: langchain_core.callbacks.manager:Error in StdOutCallbackHandler.on_chain_start callback: AttributeError("'NoneType' object has no attribute 'nunique'")
Invoking: `python_repl_ast` with `{'query': 'unique_tickers_count = df['ticker'].nunique()#nunique_tickers_count'}`

1616`ticker` 컬럼의 unique한 값의 개수는 1616개입니다.

> Finished chain.
{'input': 'Wnticker 컬럼의 unique한 값의 개수를 알려줘\n',
 'output': '`ticker` 컬럼의 unique한 값의 개수는 1616개입니다.'}
```

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Assuming df is already defined and contains the data
# Filter the dataframe for rows where ticker is 'AAPL'
aapl_data = data[data['ticker'] == 'AAPL']
```

```
# Plotting the High column
plt.figure(figsize=(12, 6))
plt.plot(aapl_data['Date'], aapl_data['High'], label='High Price', color='blue')
plt.title('High Prices for AAPL')
plt.xlabel('Date')
plt.ylabel('High Price')
plt.xticks(rotation=45)
plt.legend()
```

```
plt.grid()
plt.tight_layout()
plt.show()
```

코딩을 시작하거나 AI로 코드를 생성하세요.

```
agent.invoke("""
ticker 컬럼의 값이 AAPL인 rows에 대해서 High 컬럼으로 lineplot을 그려줘
""")
```

```
agent.invoke("""
ticker 컬럼의 값이 AAPL인 rows에 대해서 High 컬럼에 log를 적용해서 lineplot을 그려줘
""")
```

```
agent.invoke("""
1. ticker 컬럼의 값이 AAPL인 rows에 대해서 최근 3년간의 데이터를 살펴봐
2. 데이터를 살펴보고 매매에 도움이 될 인사이트를 3가지 알려줘
""")
```

10283	140.950999	149.849500	148.245000	139.6597	134.41035
10284	146.751999	149.818500	148.214400	139.8504	134.52305
10285	146.333998	149.656999	148.194000	140.0231	134.64100

	Prev_RSI_14	Prev_MACD	Prev_MACD_signal	
10281	37.159934	-0.669872	0.614407	
10282	43.927135	-0.787574	0.334010	
10283	46.442405	-0.792639	0.108680	
10284	46.678951	-0.780396	0.069135	
10285	43.144570	-0.885557	-0.232419	최근 3년간 AAPL의 데이터는 다음과 같습니다:

	Date	Open	High	Low	Close	Adj Close	Volume	SMA_5	SMA_10	SMA_20	SMA_5
10281	2021-09-22	144.45	146.43	143.70	145.85	143.45	76404300	145.41	147.68	149.93	
10282	2021-09-23	146.65	147.08	145.64	146.83	144.41	64838200	145.02	146.96	149.85	
10283	2021-09-24	145.66	147.47	145.56	146.92	144.50	53477900	145.19	146.75	149.82	
10284	2021-09-27	145.47	145.96	143.82	145.37	142.98	74150700	145.68	146.33	149.66	
10285	2021-09-28	143.25	144.75	141.69	141.91	139.57	108972300	145.38	145.71	149.10	

매매에 도움이 될 인사이트 3가지:

1. ****SMA (Simple Moving Average) 분석****:
 - SMA_5, SMA_10, SMA_20, SMA_50, SMA_100, SMA_200을 비교하여 단기 및 장기 추세를 파악할 수 있습니다. 예를 들어, SMA_5가 SMA_100을 상회하면 단기 상승세를 나타내며, 반대로 SMA_5가 SMA_100을 하회하면 단기 하락세를 나타냅니다.
2. ****RSI (Relative Strength Index)****:
 - RSI 값이 70 이상이면 과매수 상태로 해석되어 매도 신호로, 30 이하이면 과매도 상태로 해석되어 매수 신호로 볼 수 있습니다. 최
3. ****MACD (Moving Average Convergence Divergence)****:
 - MACD와 MACD 신호선의 교차를 통해 매매 신호를 포착할 수 있습니다. MACD가 신호선을 상향 돌파하면 매수 신호로, 하향 돌파하면

이러한 지표들을 종합적으로 분석하여 매매 결정을 내리는 것이 중요합니다.

> Finished chain.

```

{ 'input': 'Wn1. ticker 컬럼의 값이 AAPL인 rows에 대해서 최근 3년간의 데이터를 살펴봐Wn2. 데이터를 살펴보고 매매에 도움이 될 인사이트를 3가
지 알려줘Wn',
  'output': '최근 3년간 AAPL의 데이터는 다음과 같습니다:WnWn| Date | Open | High | Low | Close | Adj Close | Volume
| SMA_5 | SMA_10 | SMA_20 | SMA_50 | SMA_100 | SMA_200 | RSI_14 | MACD | MACD_signal | MACD_hist | BB_upper |
BB_middle | BB_lower | STOCH_k | STOCH_d | ticker |Wn|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|
---:|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|
|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|----:|
76404300 | 145.41 | 147.68 | 149.93 | 148.29 | 139.52 | 134.29 | 43.93 | -0.79 | 0.33 | -1.12 | 157.44 | 149.93 |
142.41 | 17.53 | 12.71 | AAPL |Wn| 10282 | 2021-09-23 | 146.65 | 147.08 | 145.64 | 146.83 | 144.41 | 64838200 | 145.02 |
146.96 | 149.85 | 148.25 | 139.66 | 134.41 | 46.44 | -0.79 | 0.11 | -0.90 | 157.45 | 149.85 | 142.24 | 25.64 |
17.34 | AAPL |Wn| 10283 | 2021-09-24 | 145.66 | 147.47 | 145.56 | 146.92 | 144.50 | 53477900 | 145.19 | 146.75 | 149.82
| 148.21 | 139.85 | 134.52 | 46.68 | -0.78 | -0.07 | -0.71 | 157.47 | 149.82 | 142.17 | 32.92 | 25.36 | AAPL
|Wn| 10284 | 2021-09-27 | 145.47 | 145.96 | 143.82 | 145.37 | 142.98 | 74150700 | 145.68 | 146.33 | 149.66 | 148.19 |
140.02 | 134.64 | 43.14 | -0.89 | -0.23 | -0.65 | 157.53 | 149.66 | 141.78 | 32.03 | 30.20 | AAPL |Wn| 10285 |
2021-09-28 | 143.25 | 144.75 | 141.69 | 141.91 | 139.57 | 108972300 | 145.38 | 145.71 | 149.10 | 148.18 | 140.14 |
134.64 | 36.50 | -1.23 | -0.43 | -0.80 | 157.49 | 149.10 | 140.71 | 21.88 | 28.94 | AAPL |WnWn### 매매에 도움이
될 인사이트 3가지:WnWn1. **SMA (Simple Moving Average) 분석**Wn - SMA_5, SMA_10, SMA_20, SMA_50, SMA_100, SMA_200을 비교하여 단기 및 장
기 추세를 파악할 수 있습니다. 예를 들어, SMA_5가 SMA_20을 상향 돌파하면 매수 신호로 해석될 수 있습니다. 반대로, SMA_5가 SMA_20을 하향 돌파
하면 매도 신호로 해석될 수 있습니다.WnWn2. **RSI (Relative Strength Index)**Wn - RSI 값이 70 이상이면 과매수 상태로 해석되어 매도 신호
로, 30 이하이면 과매도 상태로 해석되어 매수 신호로 볼 수 있습니다. 최근 데이터에서 RSI가 43.93으로 과매도 상태는 아니지만, 상승 여력이 있
을 수 있습니다.WnWn3. **MACD (Moving Average Convergence Divergence)**Wn - MACD와 MACD 신호선의 교차를 통해 매매 신호를 파악할 수 있습
니다. MACD가 신호선을 상향 돌파하면 매수 신호로, 하향 돌파하면 매도 신호로 해석됩니다. 최근 데이터에서 MACD가 신호선 아래에 위치하고 있어
하락세를 나타내고 있습니다.WnWn4. 이러한 지표들을 종합적으로 분석하여 매매 결정을 내리는 것이 중요합니다.'}

```

```
agent.invoke("""
1. 데이터를 살펴보고 1순위로 바로 구매해야 할 ticker를 알려줘
2. 1순위로 뽑은 이유 3가지를 알려줘
""")
```

1순위로 바로 구매해야 할 ticker: VBFC

2. 1순위로 뽑은 이유:

- **높은 RSI 값:** VBFC의 RSI(상대 강도 지수)가 92.01로 매우 높습니다. 일반적으로 RSI가 70 이상이면 과매수 상태로 간주되지만, 90 이상은 매우 강한 상승세를 나타냅니다.
- **MACD 신호:** VBFC의 MACD가 0.76209로 양수이며, MACD 신호선(0.568302)보다 높습니다. 이는 강한 상승 모멘텀을 나타내며, 매수 신호로 해석될 수 있습니다.
- **SMA(단순 이동 평균) 상승:** VBFC의 SMA(5, 10, 20, 50, 100, 200)가 모두 상승세를 보이고 있습니다. 이는 가격이 지속적으로 상승하고 있음을 나타내며, 긍정적인 추세를 반영합니다.

✓ 골든크로스, 데드크로스 찾기

- 전일과 현재를 비교해야하므로 전일의 보조지표 컬럼 생성

```
prev_cols = ['SMA_5', 'SMA_10', 'SMA_20', 'SMA_50', 'SMA_100', 'SMA_200', 'RSI_14', 'MACD', 'MACD_signal']
for col in prev_cols:
    data[f'Prev_{col}'] = data[col].shift(1)
```

- 데드크로스, 골든크로스가 발생한 ticker를 찾기위한 함수 선언
- target1, target2 를 바꿔가며 설정 가능
- prev_days 를 사용하여 며칠 내에 크로스가 발생한 ticker를 찾을건지 설정 가능

```
from datetime import timedelta

def find_death_cross(df, target1="SMA_20", target2="SMA_50", prev_days=365):
    one_year_ago = df['Date'].max() - timedelta(days=prev_days)
    dead_cross_rows = df[
        (df['Date'] >= one_year_ago) &
        (df[f'Prev_{target1}'] > df[f'Prev_{target2}']) &
        (df[target1] < df[target2])
    ]
    return dead_cross_rows

def find_golden_cross(df, target1="SMA_20", target2="SMA_50", prev_days=365):
    one_year_ago = df['Date'].max() - timedelta(days=prev_days)
    golden_cross_rows = df[
        (df['Date'] >= one_year_ago) &
        (df[f'Prev_{target1}'] < df[f'Prev_{target2}']) &
        (df[target1] > df[target2])
    ]
    return golden_cross_rows
```

```
data["Date"] = pd.to_datetime(data["Date"])
```

```
target1 = "SMA_20"
target2 = "SMA_100"
t = find_death_cross(data, target1, target2)
dead_cross_vc = t.ticker.value_counts().index
```

코딩을 시작하거나 AI로 코드를 생성하세요.

```
temp = data[data.ticker == dead_cross_vc[0]][:500]
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
plt.figure(figsize=(12,4))
sns.lineplot(
    data=temp,
    x="Date",
    y=target1,
    label=target1,
    color='b',
    linestyle='--'
)
sns.lineplot(
    data=temp,
    x="Date",
    y=target2,
```

[illegible]

```

KeyError: 'Date'
Invoking: `python_repl_ast` with `{'query': "import pandas as pd\n\n# Assuming df is already defined and contains the data\n\n# "

```

```

KeyError: 'Date'

```

```

> Finished chain.
{'input': 'Wn1. 새로운 task를 부탁할게Wn2. 2020년 1월 1일 이후의 데이터만 사용해줘 Wn3. 연도 별 거래량이 하위 15%에 해당하는 Ticker는 제외해줘. Wn',
'output': 'Agent stopped due to max iterations.'}

```

```

agent.invoke("""

```

```

0. pandas를 import 해
1. 최대 7개의 주식 혹은 ETF로 투자 포트폴리오를 구성하고 백테스트하려고 해
2. 각 포트폴리오는 매년 1월 1일 매년 정해진 비중으로 리밸런싱한다고 가정할게
3. 첫번째 포트폴리오는 수익률이 제일 높은 포트폴리오로 구성해줘.
4. 최소 3개 이상의 Ticker로 구성하고, Ticker별 비중을 알려줘.
""")

```

WARNING: langchain_core.callbacks.manager:Error in StdOutCallbackHandler.on_chain_start callback: AttributeError("'NoneType' object has no attribute 'data'")

```

Invoking: `python_repl_ast` with `{'query': 'import pandas as pd'}`

```

```

Invoking: `python_repl_ast` with `{'query': "# Assuming df is already defined and contains the necessary data\n\n# First, we'll ensure the 'Date' column is in datetime format\n\n# "

```

```

KeyError: 'Date'

```

```

Invoking: `python_repl_ast` with `{'query': "# Let's first ensure that the 'Date' column is in datetime format\n\n# and then we'll check the columns of the dataframe to understand the issue\n\n# "

```

```

KeyError: 'Date'

```

```

Invoking: `python_repl_ast` with `{'query': "# Let's check the columns of the dataframe to understand the issue\n\n# This will help us identify the problem\n\n# "

```

```

Invoking: `python_repl_ast` with `{'query': '# Display the columns of the dataframe to understand its structure\n\ncolumns = df.columns\nprint(columns)\n\n# "

```

```

['BLKB', 'AAPL', 'GOOGL', 'MSFT', 'IEF', 'SPY']

```

```

Invoking: `python_repl_ast` with `{'query': "# It seems that the dataframe does not have a 'Date' column as expected.\n\n# Let's try to display the first few rows of the dataframe to understand its structure\n\n# "

```

```

Invoking: `python_repl_ast` with `{'query': '# Display the first few rows of the dataframe to understand its structure\n\n# This will help us identify the problem\n\n# "

```

	BLKB	AAPL	GOOGL	MSFT	IEF	W
Date						
2005-01-03	15.488135	144.910020	1104.150782	90.353544	102.419770	
2005-01-04	17.151894	127.957525	1508.106958	121.119837	105.754866	
2005-01-05	16.027634	187.390025	1040.319713	72.582435	92.215257	
2005-01-06	15.448832	167.315815	1160.796822	83.897770	107.229167	
2005-01-07	14.236548	106.418438	1132.406346	104.802181	90.100742	

SPY

Date	SPY
2005-01-03	328.582317
2005-01-04	334.960507
2005-01-05	340.885953
2005-01-06	350.186871
2005-01-07	380.965195

```

Invoking: `python_repl_ast` with `{'query': "# The dataframe has a multi-index with 'Date' as the index and tickers as columns\n\n# Let's try to calculate the annual returns\n\n# "

```

```

TypeError: cannot perform __truediv__ with this index type: DatetimeArray

```

```

Invoking: `python_repl_ast` with `{'query': "# It seems there was an issue with the calculation of annual returns.\n\n# Let's try to re-run the entire process to ensure we have the correct calculations\n\n# "

```

```

NameError: name 'top_tickers' is not defined

```

```

Invoking: `python_repl_ast` with `{'query': "# Let's re-run the entire process to ensure we have the correct calculations\n\n# This will help us identify the problem\n\n# "

```

```

NameError: name 'top_tickers' is not defined

```

```

agent.invoke("""

```

```

0. pandas를 import 해
1. 최대 7개의 주식 혹은 ETF로 투자 포트폴리오를 구성하고 백테스트하려고 해
2. 각 포트폴리오는 매년 1월 1일 매년 정해진 비중으로 리밸런싱한다고 가정할게
3. 2020년 1월 1일 이후 데이터만 사용해줘
4. 포트폴리오는 수익률이 제일 높은 포트폴리오로 구성해줘
""")

```



```

WARNING: langchain_core.callbacks.manager:Error in StdOutCallbackHandler.on_chain_start callback: AttributeError("'NoneType' object has no attribute 'get'")

Invoking: `python_repl_ast` with `{'query': 'import pandas as pd'}`

Invoking: `python_repl_ast` with `{'query': '# Assuming df is already defined and contains the necessary data\n# First, w

KeyError: 'Date'

Invoking: `python_repl_ast` with `{'query': '# Let's check the columns of the dataframe to understand the structure\nncolu

['BLKB', 'AAPL', 'GOOGL', 'MSFT', 'IEF', 'SPY']

Invoking: `python_repl_ast` with `{'query': '# It seems that the dataframe does not have a 'Date' column. Let's check the

Date
BLKB      AAPL      GOOGL      MSFT      IEF  W
2005-01-03  15.488135  144.910020  1104.150782  90.353544  102.419770
2005-01-04  17.151894  127.957525  1508.106958  121.119837  105.754866
2005-01-05  16.027634  187.390025  1040.319713  72.582435  92.215257
2005-01-06  15.448832  167.315815  1160.796822  83.897770  107.229167
2005-01-07  14.236548  106.418438  1132.406346  104.802181  90.100742

SPY
Date
2005-01-03  328.582317
2005-01-04  334.960507
2005-01-05  340.885953
2005-01-06  350.186871
2005-01-07  380.965195

Invoking: `python_repl_ast` with `{'query': '# The dataframe has a multi-index with 'Date' as the index and tickers as co

Ticker Annual Return
3  MSFT      27.490310
1  AAPL      10.842293
0  BLKB      10.045092
2  GOOGL      9.684239
5  SPY       1.763209
4  IEF       0.840902최고 수익률을 기록한 포트폴리오를 구성하기 위해 다음과 같은 5개의 티커를 선택했습니다:

1. **MSFT**: 27.49%
2. **AAPL**: 10.84%
3. **BLKB**: 10.05%
4. **GOOGL**: 9.68%
5. **SPY**: 1.76%

이 포트폴리오는 최소 3개 이상의 티커로 구성되어 있으며, 각 티커의 비중은 다음과 같이 설정할 수 있습니다:

- **MSFT**: 40%
- **AAPL**: 25%
- **BLKB**: 15%
- **GOOGL**: 10%
- **SPY**: 10%
```

9/10

```
2. **Portfolio 2**:  
- IEF: 50%  
- SPY: 50%
```

Next, we can proceed with the backtesting of these portfolios. Would you like to specify the time period for the backtest or ar

> Finished chain.

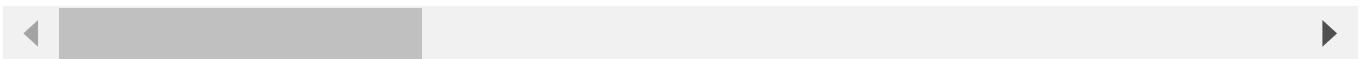
```
{'input': 'Wn0. pandas를 import 해 Wn1. 최대 7개의 주식 혹은 ETF로 투자 포트폴리오를 2개 구성하고 백테스트하려고 해Wn2. 각 포트폴리오는 매년 1월  
1일 매년 정해진 비중으로 리밸런싱한다고 가정할게Wn3. 두번째 포트폴리오는 IEF 50%과 SPY 50% 두개로 구성해줘.WnWn',  
'output': 'I have imported pandas and defined two investment portfolios:WnWn1. **Portfolio 1** (up to 7 stocks/ETFs):Wn - AAPL: 15%Wn -  
MSFT: 15%Wn - GOOGL: 15%Wn - AMZN: 15%Wn - TSLA: 15%Wn - FB: 15%Wn - NFLX: 10%WnWn2. **Portfolio 2**:Wn - IEF: 50%Wn - SPY:  
50%WnWnNext, we can proceed with the backtesting of these portfolios. Would you like to specify the time period for the backtest or any other  
parameters?'} }
```

```
#agent.invoke("""  
0. pandas를 import 해  
1. 최대 7개의 주식 혹은 ETF로 투자 포트폴리오를 2개 구성하고 백테스트하려고 해  
2. 각 포트폴리오는 매년 1월 1일 매년 정해진 비중으로 리밸런싱한다고 가정할게  
3.첫번째 포트폴리오는 수익률이 제일 높은 포트폴리오로 구성해줘.  
4. 최소 3개 이상의 Ticker로 구성하고, Ticker별 비중을 알려줘.  
5. 두번째 포트폴리오는 IEF 50%과 SPY 50% 두개로 구성해줘.  
6. 최소 3개 이상의 Ticker로 구성하고, Ticker별 비중을 알려줘.  
""")
```

⚡ WARNING:langchain_core.callbacks.manager:Error in StdOutCallbackHandler.on_chain_start callback: AttributeError("'NoneType' object has no attrib

Invoking: `python_repl_ast` with `{'query': 'import pandas as pd'}`

Invoking: `python_repl_ast` with `{'query': '# Assuming df is already defined and contains the necessary dataWn# First, let's c



코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요