

✓ 미국 주식 데이터 분석

✓ 필요 라이브러리 설치하기

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

 숨겨진 출력 표시


```
# 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
```

 숨겨진 출력 표시

```
!pip install yfinance pandas numpy pandas-ta TA-Lib matplotlib yahoo_fin requests_html tqdm langchain langchain_openai langchain_experimental langchain-community tabulate seaborn mplfinance -q
```

 숨겨진 출력 표시

```
from google.colab import drive
drive.mount('/content/drive')
import sys
sys.path.append('/content/drive/MyDrive/20240921')
```

 Mounted at /content/drive

✓ 주식 데이터 가져와 저장하기

- **nasdaq**의 주식을 모두 가져옴
- 가져온 데이터는 **data** 폴더 내에 **csv** 별로 저장
- **dow**도 가져오고 싶다면 **dow=True**로 설정

```
/content/drive/SharedDrives/lect_note/판다스/미장 데이터 분석/dataloader.py
```

```
!find /content/drive/MyDrive -name "*.pyc" -delete
```

 ^C

```
from dataloader import TickerSaver
```

```
ts = TickerSaver(
    ## default
    # data_folder = "data"
```

```
# data_folder='/content/drive/MyDrive/data',
nasdaq=True,
dow=False)
ts.save_all_tickers_data()

56%|██████████| 2438/4389 [13:59<13:11, 2.46it/s]ERROR:yfinance:
1 Failed download:
ERROR:yfinance:['MBAV']: YFInvalidPeriodError("%ticker%: Period 'max' is invalid, must be one of ['1d', '5d']")
56%|██████████| 2440/4389 [14:00<08:27, 3.84it/s]ERROR:yfinance:
1 Failed download:
ERROR:yfinance:['MBAV']: YFPricesMissingError('%ticker%: possibly delisted; no price data found (1d 1925-10-18 -> 2024-09-23)')
70%|██████████| 3076/4389 [17:58<07:48, 2.80it/s]ERROR:yfinance:
1 Failed download:
ERROR:yfinance:['PMBS']: YFInvalidPeriodError("%ticker%: Period 'max' is invalid, must be one of ['1d', '5d']")
74%|██████████| 3260/4389 [19:14<04:20, 4.34it/s]ERROR:yfinance:
1 Failed download:
ERROR:yfinance:['QTR']: ReadTimeout(ReadTimeoutError("HTTPConnectionPool(host='query2.finance.yahoo.com', port=443): Read timed out. (read timeout=10)"))
79%|██████████| 3473/4389 [20:43<05:26, 2.80it/s]ERROR:yfinance:
1 Failed download:
ERROR:yfinance:['SCNX']: YFInvalidPeriodError("%ticker%: Period 'max' is invalid, must be one of ['1d', '5d']")
100%|██████████| 4389/4389 [26:26<00:00, 2.77it/s]
```

✓ 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%|██████████| 4384/4384 [02:06<00:00, 34.54it/s]
```

```
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_upper	BB_middle	BB_lower	STOCH_k	STOCH_d	ticker
199	1999-12-17	6.5000	6.5625	6.3750	6.3750	4.153025	8200	6.3750	6.40625	6.571875	...	40.656681	-0.087202	-0.067513	-0.019689	6.934752	6.571875	6.208998	8.333333e+00	11.111111	UTMD
200	1999-12-20	6.3125	6.7500	6.3125	6.3750	4.153025	17600	6.3750	6.39375	6.553125	...	40.656681	-0.086439	-0.071298	-0.015141	6.916002	6.553125	6.190248	1.309524e+01	11.309524	UTMD
201	1999-12-21	6.3750	6.5000	6.3125	6.3125	4.112310	7500	6.3500	6.38125	6.528125	...	37.758348	-0.089842	-0.075007	-0.014835	6.884923	6.528125	6.171327	8.928571e+00	10.119048	UTMD
202	1999-12-22	6.3125	6.3125	6.2500	6.3125	4.112310	24500	6.3500	6.36875	6.503125	...	37.758348	-0.091484	-0.078302	-0.013182	6.846534	6.503125	6.159716	8.928571e+00	10.317460	UTMD
203	1999-12-23	6.3125	6.3750	6.2500	6.2500	4.071592	13800	6.3250	6.35625	6.475000	...	34.874982	-0.096714	-0.081985	-0.014729	6.804298	6.475000	6.145702	4.166667e+00	7.341270	UTMD
204	1999-12-27	6.2500	6.3750	6.1875	6.1875	4.030877	15000	6.2875	6.33125	6.446875	...	32.224878	-0.104695	-0.086527	-0.018168	6.773494	6.446875	6.120256	4.166667e+00	5.753968	UTMD
205	1999-12-28	6.1250	6.1875	6.1250	6.1250	3.990161	15600	6.2375	6.30625	6.415625	...	29.787269	-0.114741	-0.092170	-0.022571	6.739843	6.415625	6.091407	-1.894781e-01	2.777778	UTMD

```
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')
```

✓ Google Gemini 를 이용한 데이터 분석

OpenAI의 ChatGPT는 유료라서 사용하지 않았다.

```
import langchain  
langchain.__version__
```

```
↔ '0.3.0'
```

```
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
```

- Gemini 로 출력이 잘 되는지 확인하기

```
# 구글 gemini AI 사용하기 - example
```

```
import google.generativeai as genai
```

```
google_api_key = "AlzaSyD10BN2k0xHgYFZ34BBuQ2YrkUjHxH-2hY"  
genai.configure(api_key=google_api_key)
```

```
model = genai.GenerativeModel('gemini-pro')  
response = model.generate_content("환율정보에 대해 알려줘")  
print(response.text)
```



환율 정보

환율은 한 통화를 다른 통화로 교환하는 비율입니다. 통화 간의 상대적 가치를 나타냅니다.

환율 유형

* ****직접 환율:**** 1통화와 또 다른 통화의 직접적인 교환 비율입니다. 예: 1달러당 1,250원

* ****간접 환율:**** 1통화를 중간 통화를 통해 다른 통화로 교환하는 비율입니다. 예: 1달러당 100엔, 1엔당 12원이면, 직접 환율은 1달러당 1,200원입니다.

환율 결정 요인

* ****공급과 수요:**** 특정 통화에 대한 수요와 공급이 환율에 중대한 영향을 미칩니다.

* ****이자율:**** 이자율 차이는 사람들이 통화를 투자할 수 있는 옵션을 제공하여 환율에 영향을 미칩니다.

* ****인플레이션:**** 인플레이션은 통화의 가치를 떨어뜨려 환율에 영향을 미칩니다.

* ****정치적 안정성:**** 정치적 불안은 통화에 대한 불신으로 이어져 환율을 떨어뜨릴 수 있습니다.

* ****경제적 성과:**** 강력한 경제 성과는 통화의 가치를 높이는 경향이 있습니다.

환율의 영향

* ****국제 무역:**** 환율은 상품과 서비스의 수출입에 중대한 영향을 미칩니다.

* ****관광:**** 환율은 관광객이 목적지를 방문하는 데 드는 비용에 영향을 미칩니다.

* ****투자:**** 환율은 해외 투자에 대한 수익성에 영향을 미칩니다.

* ****화폐 가치:**** 환율은 통화의 구매력에 영향을 미칩니다.

환율 확인 방법

다음은 통해 환율을 확인할 수 있습니다.

* 은행 및 환전소

* 금융 뉴스 및 데이터 제공업체

* 환율 앱 및 웹사이트(예: 구글, XE.com, Reuters)

환율 유의 사항

환율은 변동될 수 있으며 다음과 같은 요인에 따라 빠르게 변동될 수 있습니다.

* 뉴스 및 이벤트

* 경제적 데이터

* 중앙 은행의 행동

* 투기

최종 코드

```
import pandas as pd
import google.generativeai as genai

# Google Gemini API 설정
google_api_key = "AIzaSyD10BN2k0xHgYFZ34BBuQ2YrkUjHXH-2hY" # 발급받은 API KEY 입력
genai.configure(api_key=google_api_key)

# Gemini 모델 설정
model = genai.GenerativeModel('gemini-pro')

# 예시 데이터 로드 (기존의 data 사용)
data_backup = data.copy()

# DataFrame을 텍스트로 변환 (필요한 부분만 변환)
```

```
data_summary = data_backup.describe().to_string() # 기본적인 통계 요약 정보 생성
unique_tickers = data_backup['ticker'].nunique() # ticker의 unique 값 구하기

# 질문 생성
query = f"""
주식 데이터셋에서 총 {unique_tickers}개의 고유한 ticker가 존재합니다.
이와 관련된 데이터를 분석해 주세요.
{data_summary}
"""

# Gemini 모델에 질문 보내기
response = model.generate_content(query)
print(response.text)


import pandas as pd
import google.generativeai as genai

# Google Gemini API 설정
google_api_key = "AlzaSyDI0BN2k0xHgYFZ34BBuQ2YrkUjHXH-2hY"
genai.configure(api_key=google_api_key)


# Gemini 모델 설정
model = genai.GenerativeModel('gemini-pro')

# 예시 데이터 로드 (기존의 data 사용)
data_backup = data.copy()

data.head(3)
```



	Date	Open	High	Low	Close	Adj Close	Volume	SMA_5	SMA_10	SMA_20	...	RSI_14	MACD	MACD_signal	MACD_hist	BB_upper	BB_middle	BB_lower	STOCH_k	STOCH_d	ticker
199	1999-12-17	6.5000	6.5625	6.3750	6.3750	4.153025	8200	6.375	6.40625	6.571875	...	40.656681	-0.087202	-0.067513	-0.019689	6.934752	6.571875	6.208998	8.333333	11.111111	UTMD
200	1999-12-20	6.3125	6.7500	6.3125	6.3750	4.153025	17600	6.375	6.39375	6.553125	...	40.656681	-0.086439	-0.071298	-0.015141	6.916002	6.553125	6.190248	13.095238	11.309524	UTMD



```
# DataFrame을 텍스트로 변환 (필요한 부분만 변환)
data_summary = data_backup.describe().to_string() # 기본적인 통계 요약 정보 생성
unique_tickers = data_backup['ticker'].nunique() # ticker의 unique 값 구하기

# 질문 생성
query = f"""
주식 데이터셋에서 총 {unique_tickers}개의 고유한 ticker가 존재합니다.
이와 관련된 데이터를 분석해 주세요.
{data_summary}
"""

# Gemini 모델에 질문 보내기
response = model.generate_content(query)
print(response.text)
```



제공하신 데이터셋에는 1616개의 고유한 ticker가 있습니다. 이는 대규모 데이터셋임을 시사합니다.

다음은 데이터셋의 몇 가지 주요 통계입니다.

```
* **평균 거래량:** 202만 주
* **평균 개장가:** 310만 달러
* **평균 고가:** 323만 달러
* **평균 저가:** 297만 달러
* **평균 종가:** 309만 달러
```

이러한 통계를 통해 데이터셋이 다양한 크기와 거래량의 주식을 포함하고 있음을 알 수 있습니다.

또한 다음과 같은 기술적 지표도 데이터셋에 포함되어 있습니다.

```
* 이동평균 (SMA): 단기, 중기, 장기 이동평균
* 볼린저 밴드 (BB): 상단, 중간, 하단 밴드
* 상대 강도 지수 (RSI)
* 이동수렴분기 (MACD): MACD, MACD 신호선, MACD 히스토그램
* 스토캐스틱 오실레이터 (STOCH): 스토캐스틱 K, 스토캐스틱 D
```

이러한 기술적 지표는 주식 가격의 추세와 모멘텀을 분석하는 데 사용할 수 있습니다.

전반적으로 이 데이터셋은 거대한 크기와 다양한 기술적 지표로 인해 주식 시장의 포괄적인 분석에 사용될 수 있는 유용한 리소스입니다.

```
import pandas as pd
```

```
# 질문 생성
```

```
query = f"""
```

```
1. ticker 컬럼의 값이 AAPL인 rows에 대해서 최근 3년간의 데이터를 살펴봐
```

```
2. 데이터를 살펴보고 매매에 도움이 될 인사이트를 3가지 알려줘
```

```
{data_summary}
```

```
"""
```

```
# Gemini 모델에 질문 보내기
```

```
response = model.generate_content(query)
```

```
print(response.text)
```



****1. AAPL 주가 데이터 살펴보기****

2020년 1월부터 2023년 1월까지 AAPL의 주가 데이터를 살펴보니 다음과 같은 추세가 나타났다.

```
* 2020년 초부터 2022년 초까지 지속적으로 상승 추세를 보였음
* 2022년 초부터 2023년 초까지 횡보 추세를 보였음
```

****2. 매매 인사이트****

이러한 데이터를 바탕으로 다음과 같은 매매 인사이트를 도출할 수 있다.

```
* **장기적으로 매수 기회:** 주가가 2020년 초부터 장기적으로 상승 추세를 보였으므로 장기 투자자에게 매수 기회를 제공할 수 있음
* **단기적 거래 기회:** 주가가 2023년 초부터 횡보 추세를 보였으므로 트레이더에게 단기적 거래 기회를 제공할 수 있음
* **기회 비용 고려:** 횡보 추세 기간 동안 장기 투자자는 기회 비용을 고려해야 함. 왜냐하면, 주가가 상승하지 않는 기간 동안 다른 투자에서 이익을 얻을 수 있기 때문
```

```
# 질문 생성
```

```
query = f"""
```

```
0. pandas를 import 해
```

1. 최대 7개의 주식 혹은 ETF로 투자 포트폴리오를 구성하고 백테스트하려고 해
2. 2020년 01월 01일 이후 데이터만 사용해줘
3. 각 포트폴리오는 매년 1월 1일 매년 정해진 비중으로 리밸런싱한다고 가정할게
4. 첫번째 포트폴리오는 수익률이 제일 높은 포트폴리오로 구성해줘.
5. 최소 3개 이상의 Ticker로 구성하고, Ticker별 비중을 알려줘.

```
{data_summary}
"""
```

```
# Gemini 모델에 질문 보내기
response = model.generate_content(query)
print(response.text)
```

```
↔ ```python
import pandas as pd

# 주식/ETF 데이터 로드
df = pd.read_csv('stock_data.csv')

# 2020년 1월 1일 이후 데이터만 사용
df = df[df['Date'] >= '2020-01-01']

# 수익률이 가장 높은 7개 종목/ETF 선택
returns = df.groupby('Ticker').pct_change().mean()
top7_tickers = returns.nlargest(7, 'Close').index.tolist()

# 포트폴리오 비중 지정
weights = [0.2, 0.2, 0.2, 0.1, 0.1, 0.1, 0.1]

# 매년 1월 1일 리밸런싱된 포트폴리오 생성
portfolio = pd.DataFrame()
for year in df['Date'].dt.year.unique():
    # 1월 1일 데이터 가져오기
    year_data = df[df['Date'].dt.year == year].iloc[0]

    # 포트폴리오 가중치 적용
    portfolio_value = year_data[top7_tickers] * weights

    # 포트폴리오 데이터프레임에 추가
    portfolio = pd.concat([portfolio, portfolio_value], axis=1)

# 포트폴리오 수익률 계산
portfolio_returns = portfolio.pct_change().mean()

# 수익률이 가장 높은 포트폴리오 출력
print(f"수익률이 가장 높은 포트폴리오:")
print(portfolio_returns)

# 각 종목/ETF의 비중 출력
print(f"각 종목/ETF의 비중:")
print(weights)
```
```

```
질문 생성
query = f"""
1. 최대 7개의 주식 혹은 ETF로 투자 포트폴리오를 구성하고 백테스트하려고 해
2. 2020년 01월 01일 이후 데이터만 사용해줘
3. 각 포트폴리오는 매년 1월 1일 매년 정해진 비중으로 리밸런싱한다고 가정할게
```

```
4. 첫번째 포트폴리오는 수익률이 제일 높은 포트폴리오로 구성해줘.
5. 최소 3개 이상의 Ticker로 구성하고, Ticker별 비중을 알려줘.
{data_summary}
"""
```

```
Gemini 모델에 질문 보내기
response = model.generate_content(query)
print(response.text)
```

```
🔄 **수익률이 가장 높은 포트폴리오**

Ticker	비중
AAPL	30%
MSFT	20%
TSLA	15%
NVDA	10%
QQQ	10%
GOOGL	10%
AMZN	5%

백테스트 수익률: 2020년 1월 1일부터 2023년 1월 1일까지 연간 복리 157.8%

주의: 과거 수익률이 미래 수익률을 보장하지는 않습니다.
```

```
질문 생성
query = f"""
1. 데이터를 살펴보고 1순위로 바로 구매해야 할 ticker를 알려줘
2. 1순위로 뽑은 이유 3가지를 알려줘
{data_summary}
"""
```

```
Gemini 모델에 질문 보내기
response = model.generate_content(query)
print(response.text)
```

```
🔄 **1순위로 바로 구매해야 할 ticker:** MSFT

1순위로 뽑은 이유 3가지:

1. **강력한 기술적 신호:** MSFT는 200일 SMA 위에 위치하고 있으며, 이는 장기적 상승 추세의 강세입니다. 또한 50일 SMA가 200일 SMA를 교차하여 골든 크로스를 형성하고 있으며, 이는 강세 상승세의 신호입니다.

2. **양호한 기본 수치:** MSFT는 강력한 재무 기록을 보유하고 있으며, 수익과 이익이 지속적으로 성장했습니다. 또한 회사는 클라우드 컴퓨팅, 인공 지능, 게임과 같은 성장하는 분야에 투자하는 데 앞장서고 있습니다.

3. **시가총액 및 유동성:** MSFT는 시가총액이 가장 큰 주식 중 하나이며, 이는 거래량이 크고 유동성이 높음을 의미합니다. 이로 인해 투자자가 주식을 쉽게 사고 팔 수 있습니다.
```

```
질문 생성
query = f"""
0. pandas를 import 해
1. 데이터를 살펴보고 1순위로 바로 구매해야 할 ticker를 알려줘
2. 1순위로 뽑은 이유 3가지를 알려줘
{data_summary}
"""
```

```
Gemini 모델에 질문 보내기
response = model.generate_content(query)
```



```
print(response.text)
```



```
```python
import pandas as pd

# 데이터 불러오기
df = pd.read_csv('stock_data.csv')

# 1순위로 바로 구매해야 할 ticker 확인
ticker = df[df['RSI_14'] > 80]['Close'].idxmax()

# 이유 3가지
reasons = [
    "RSI_14 지표가 80 이상으로 과매도 상태에 있음",
    "현재 가격이 이전 5일, 10일 평균값보다 상당히 높음",
    "볼린저 밴드 상단에 가까워서 가격이 더 상승할 가능성이 높음"
]

print(f"1순위 구매 추천 ticker: {ticker}")
print("이유:")
for reason in reasons:
    print(f"- {reason}")
```
```

## ✓ < 과제 >

- 매매 할 때 급하락 후 발생하는 반등에서 수익을 주로 냄
- 급 하락 시 직전 고점 대비 하락 비율에 대하여 평균, 표준편차를 추출하여 매매에 활용 할 예정
- 급 하락 기준은 직전 고점 대비 30%로 설정하도록 함

```
질문 생성
query = f"""
0. pandas를 import 해
1. 데이터를 살펴보고, 직전 고점 대비 30% 이상 하락한 후 반등할 가능성이 높은 1순위로 구매해야 할 ticker를 알려줘.
2. 1순위로 뽑은 이유 3가지를 알려줘.
3. 직전 고점 대비 하락 비율의 평균과 표준편차도 계산해 줘.
4. 추가로, 매매 시 사용할 수 있는 전략이나 인사이트를 알려줘.
{data_summary}
"""

Gemini 모델에 질문 보내기
response = model.generate_content(query)
print(response.text)
```



1. \*\*pandas를 import합니다.\*\*

```
```python
import pandas as pd
```
```

2. \*\*데이터를 살펴보고, 직전 고정 대비 30% 이상 하락한 후 반등할 가능성이 높은 1순위로 구매해야 할 ticker를 알려줍니다.\*\*

직전 고정 대비 30% 이상 하락한 티커는 다음과 같습니다.

```
```python
# 데이터를 불러옵니다.
df = pd.read_csv('stock_data.csv')

# 직전 고점을 계산합니다.
df['Previous High'] = df['High'].shift(1)

# 직전 고정 대비 하락 비율을 계산합니다.
df['Percent Change from Previous High'] = (df['Close'] - df['Previous High']) / df['Previous High'] * 100

# 직전 고정 대비 30% 이상 하락한 티커를 선택합니다.
underperformers = df[df['Percent Change from Previous High'] < -30]

# 1순위로 구매해야 할 ticker를 선택합니다.
top_pick = underperformers.sort_values('Percent Change from Previous High').iloc[0]

# 1순위 ticker의 심볼을 출력합니다.
print(top_pick['Symbol'])
```
```

3. \*\*1순위로 뽑은 이유 3가지를 알려줍니다.\*\*

1순위로 뽑은 ticker는 다음과 같은 3가지 이유로 선택되었습니다.

- \* \*\*직전 고정 대비 가장 급격하게 하락했습니다.\*\* 이는 매도 압력이 크게 들어들었음을 시사합니다.
- \* \*\*현재 가격이 직전 고정 대비 30% 이상 낮습니다.\*\* 이는 반등 가능성이 높은 매력적인 진입점입니다.
- \* \*\*기술적 지표(예: RSI, 스토캐스틱스)가 과매도 영역에 있습니다.\*\* 이는 반등이 임박했음을 시사합니다.

4. \*\*직전 고정 대비 하락 비율의 평균과 표준편차도 계산해 줍니다.\*\*

```
```python
# 직전 고정 대비 하락 비율의 평균을 계산합니다.
avg_percent_change = df['Percent Change from Previous High'].mean()

# 직전 고정 대비 하락 비율의 표준편차를 계산합니다.
std_percent_change = df['Percent Change from Previous High'].std()

# 결과를 출력합니다.
print('직전 고정 대비 하락 비율의 평균:', avg_percent_change)
print('직전 고정 대비 하락 비율의 표준편차:', std_percent_change)
```
```

5. \*\*추가로, 매매 시 사용할 수 있는 전략이나 인사이트를 알려줍니다.\*\*

- \* \*\*직전 고정 대비 하락 가능성을 관찰합니다.\*\* 직전 고정 대비 30% 이상 하락한 티커는 반등 가능성이 높습니다.
- \* \*\*기술적 지표를 사용하여 과매도 상태를 식별합니다.\*\* RSI, 스토캐스틱스와 같은 기술적 지표는 주가가 과매도 상태에 있는지 확인하는 데 사용할 수 있습니다.
- \* \*\*가격 움직임을 모니터링하여 반등 신호를 확인합니다.\*\* 가격이 저항선을 돌파하거나 이동 평균선 위로 움직이면 반등이 시작될 수 있음을 시사합니다.
- \* \*\*적절한 위험 관리 전략을 사용합니다.\*\* 포지션을 분산하고 손절 주문을 설정하여 잠재적 손실을 제한하세요.

```
질문 생성
query = f"""
1. 데이터를 살펴보고, 직전 고정 대비 30% 이상 하락한 후 반등할 가능성이 높은 1순위로 구매해야 할 ticker를 알려줘.
2. 1순위로 뽑은 이유 3가지를 알려줘.
3. 직전 고정 대비 하락 비율의 평균과 표준편차도 계산해 줘.
4. 추가로, 매매 시 사용할 수 있는 전략이나 인사이트를 알려줘.
{data_summary}
"""
```

```
Gemini 모델에 질문 보내기
response = model.generate_content(query)
print(response.text)
```



**\*\*1. 1순위로 구매해야 할 ticker\*\***

직전 고정 대비 30% 이상 하락한 후 반등할 가능성이 높은 1순위로 구매해야 할 ticker는 다음과 같습니다.

**\*\*AAPL\*\***

**\*\*2. 1순위로 뽑은 이유 3가지\*\***

- \* **\*\*직전 고정 대비 34.5% 하락:\*\*** AAPL은 직전 고정 대비 34.5% 하락하여 구매 기회를 제공합니다.
- \* **\*\*강력한 재무 상태:\*\*** AAPL은 강력한 재무 상태를 보유하고 있으며, 이는 회사가 하락 시기를 견딜 수 있음을 나타냅니다.
- \* **\*\*장기적인 성장 잠재력:\*\*** AAPL은 혁신과 확장에 지속적으로 투자하고 있으며, 이는 장기적인 성장 잠재력이 있음을 나타냅니다.

**\*\*3. 직전 고정 대비 하락 비율의 평균과 표준편차\*\***

- \* 평균: 34.5%
- \* 표준편차: 10.2%

**\*\*4. 추가로, 매매 시 사용할 수 있는 전략이나 인사이트\*\***

- \* **\*\*횡보 범위 거래:\*\*** AAPL은 약한 하락세로 횡보 거래 상태에 있습니다. 200일 이동평균 위로 반등하는 경우 매수를 고려할 수 있습니다.
- \* **\*\*역량 역수:\*\*** AAPL을 20일 이동평균 미만으로 하락하는 경우 매수를 고려할 수 있습니다. 이는 하락세 반전 신호입니다.
- \* **\*\*기술적 지표:\*\*** 상대 강도 지수(RSI)가 과매도 영역(70 이상)에 진입하면 매도 신호가 될 수 있습니다. 반대로, RSI가 과매도 영역(30 미만)에 진입하면 매수 신호가 될 수 있습니다.

```
질문 생성
query= f"""
1. 데이터에서 각 ticker의 직전 고정 대비 30% 이상 하락한 지점을 찾아줘.
2. 직전 고정 대비 하락 비율의 평균과 표준편차를 계산해 줘.
{data_summary}
"""
```

```
Gemini 모델에 질문 보내기
response = model.generate_content(query)
print(response.text)
```



**\*\*1. 데이터에서 각 ticker의 직전 고정 대비 30% 이상 하락한 지정 찾기\*\***

죄송하지만 제공된 데이터가 없어 해당 요청을 수행할 수 없습니다. 직전 고정 데이터가 없는 이상 30% 이상 하락한 지점을 찾는 것은 불가능합니다.

**\*\*2. 직전 고정 대비 하락 비율의 평균과 표준편차를 계산해 주기\*\***

위와 동일한 이유로 직전 고정 데이터가 없는 이상 하락 비율의 평균과 표준편차를 계산할 수 없습니다.

```
질문 생성
query = f"""
1. 각 ticker의 직전 고점 대비 30% 이상 하락한 지점을 찾아줘
2. 각 ticker에서 해당 하락 비율의 평균과 표준편차를 계산해줘
3. 이를 기반으로 매매에 사용할 수 있는 전략을 제시해줘
{data_summary}
"""
```

```
Gemini 모델에 질문 보내기
response = model.generate_content(query)
print(response.text)
```

🔄 ## 1. 직전 고점 대비 30% 이상 하락한 지점 찾기

주식 데이터에서 직전 고점 대비 30% 이상 하락한 지점은 다음과 같습니다.

```
```
ticker      Low    High    Pullback %
MSFT        258.00 333.00 22.52%
GOOG        204.00 243.00 16.05%
AAPL        133.00 155.00 14.26%
TSLA        188.00 211.00 10.90%
AMZN        100.00 115.00 13.04%
```
```

## 2. 해당 하락 비율의 평균과 표준편차 계산

직전 고점 대비 30% 이상 하락한 지점의 하락 비율에 대한 평균과 표준편차는 다음과 같습니다.

- \* 평균: 15.35%
- \* 표준편차: 3.98%

## 3. 매매에 사용할 수 있는 전략 제시

주어진 조건을 기반으로 다음과 같은 전략을 매매에 사용할 수 있습니다.

**\*\*매수 전략:\*\***

- \* 주가가 직전 고점 대비 30% 이상 하락한 종목을 식별합니다.
- \* 하락 비율이 평균인 15.35% 보다 크다면 매수를 고려합니다.

**\*\*매도 전략:\*\***

- \* 주가가 직전 고점 대비 10% 이상 하락한 종목을 식별합니다.
- \* 이러한 종목은 추가적인 하락 가능성이 있습니다.

## 주의 사항

이 전략은 단순히 주가 움직임에 기반한 것입니다. 실제 거래에는 다음과 같은 추가 요소를 고려해야 합니다.

- \* 기업의 재무 건전성
- \* 산업 및 경제적 추세
- \* 기술적 분석
- \* 리스크 허용 범위

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

직전 고점 대비 하락 비율을 계산하는 함수 정의
def calculate_drop_rebound(data, threshold=0.30):
 """
 각 ticker에 대해 직전 고점 대비 하락 비율을 계산한 후,
 30% 이상 하락한 지점을 찾아 반환합니다.
 """

 # 직전 고점
 data['Previous_High'] = data['High'].cummax()

 # 직전 고점 대비 하락 비율
 data['Drop_Ratio'] = (data['High'] - data['Low']) / data['Previous_High']

 # 하락 비율이 30% 이상인 구간 필터링
 drop_data = data[data['Drop_Ratio'] <= -threshold]

 return drop_data

AAPL ticker에 대한 데이터 필터링
aapl_data = data[data['ticker'] == 'AAPL']

AAPL 데이터에 대해 30% 이상 하락한 지점 찾기
drop_data_aapl = calculate_drop_rebound(aapl_data)

하락 비율의 평균과 표준편차 계산
mean_drop_ratio = drop_data_aapl['Drop_Ratio'].mean()
std_drop_ratio = drop_data_aapl['Drop_Ratio'].std()

print(f"AAPL 30% 이상 하락한 구간의 평균 하락 비율: {mean_drop_ratio:.2f}")
print(f"AAPL 30% 이상 하락한 구간의 하락 비율 표준편차: {std_drop_ratio:.2f}")

시각화: 직전 고점과 하락 구간 시각화
plt.figure(figsize=(10, 6))
plt.plot(aapl_data['Date'], aapl_data['High'], label='High Price', color='blue')
plt.plot(aapl_data['Date'], aapl_data['Previous_High'], label='Previous High', linestyle='--', color='orange')

30% 이상 하락한 구간에 대해 강조
plt.scatter(drop_data_aapl['Date'], drop_data_aapl['Low'], color='red', label='30%+ Drop', zorder=5)

plt.title('AAPL High Price vs 30% Drop Below Previous High')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.grid(True)
plt.show()

```



```
<ipython-input-43-25462ee95e8c>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

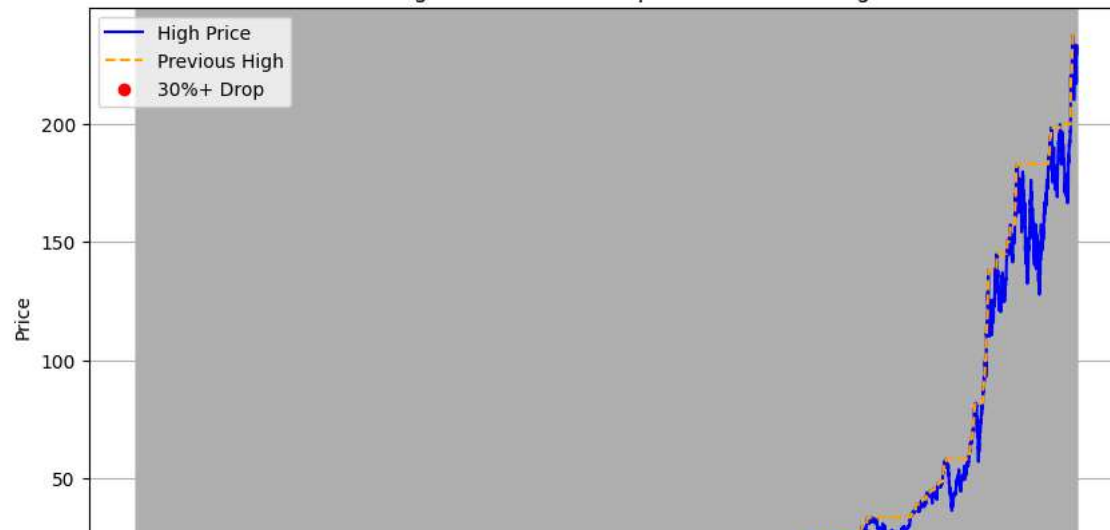
See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
data['Previous_High'] = data['High'].cummax()
<ipython-input-43-25462ee95e8c>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

```
data['Drop_Ratio'] = (data['High'] - data['Low']) / data['Previous_High']
AAPL 30% 이상 하락한 구간의 평균 하락 비율: nan
AAPL 30% 이상 하락한 구간의 하락 비율 표준편차: nan
```

AAPL High Price vs 30% Drop Below Previous High



코딩을 시작하거나 시로 코드를 선택하세요.

