



# 파이썬과 DB로 주식 분석 시스템 만들기



- 08-1 실습 환경 구성하기
- 08-2 크롤러 및 저장 시스템 만들기
- 08-3 MySQL로 주식 분석하기

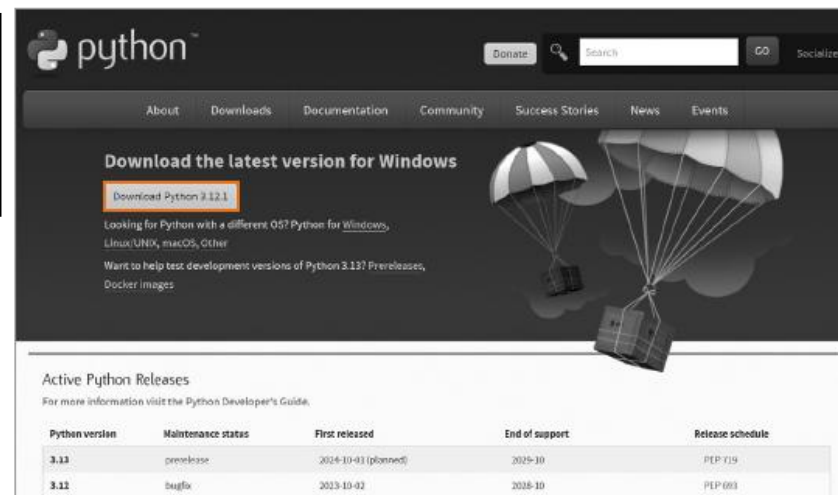
- 파이썬과 데이터베이스를 활용하면  
매일 변화하는 주식 데이터를 내 마음대로 활용하고 분석할 수 있음
- 좀 더 자세히 말하면 파이썬을 활용해 주식 관련 웹 크롤링을 하고  
크롤링 과정을 통해 얻은 데이터를 데이터베이스에 저장해  
이를 다양하게 활용할 수 있음
- 실생활에서 활용하기 좋은 이 실습을 구현하기 위해  
우선 파이썬과 DBMS가 설치되어 있어야 함
- DBMS인 MySQL은 이미 설치되어 있으므로  
파이썬 개발 환경을 구축해 보자

## ■ 파이썬 설치하기

- 파이썬 개발 환경을 구성하기 위해서 먼저 파이썬을 설치해야 함
1. 다음 링크에 접속해 내 컴퓨터의 운영체제에 맞게 선택해 파이썬을 설치해 보자.  
여기서는 윈도우 환경을 기준으로 설명하였고, macOS도 크게 다르지는 않음

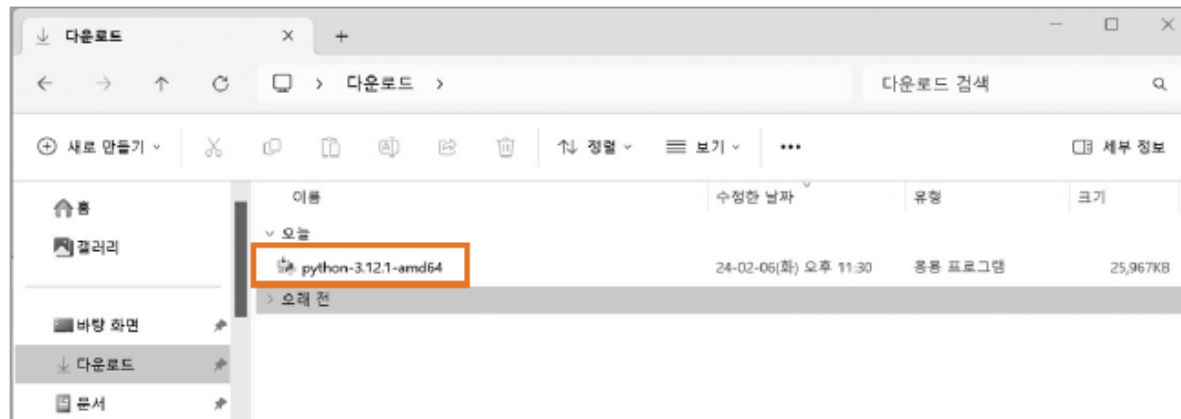
### 파이썬 설치 파일 내려받기 링크

- <https://www.python.org/downloads>



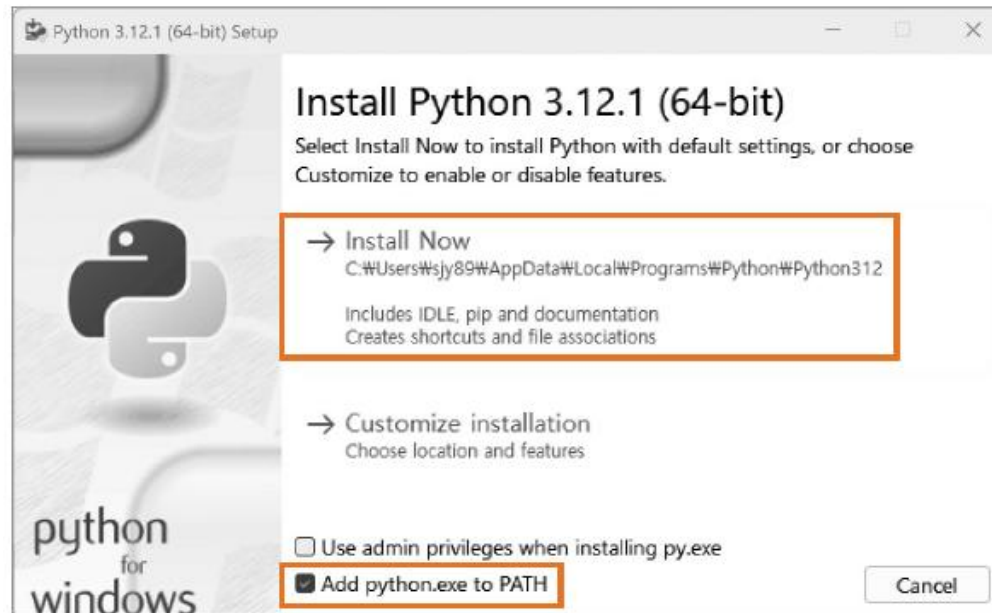
## ■ 파이썬 설치하기

2. 파이썬 설치 파일을 내려받은 폴더로 이동하자.  
내려받은 파이썬 설치 파일을 더블클릭해 설치를 시작함



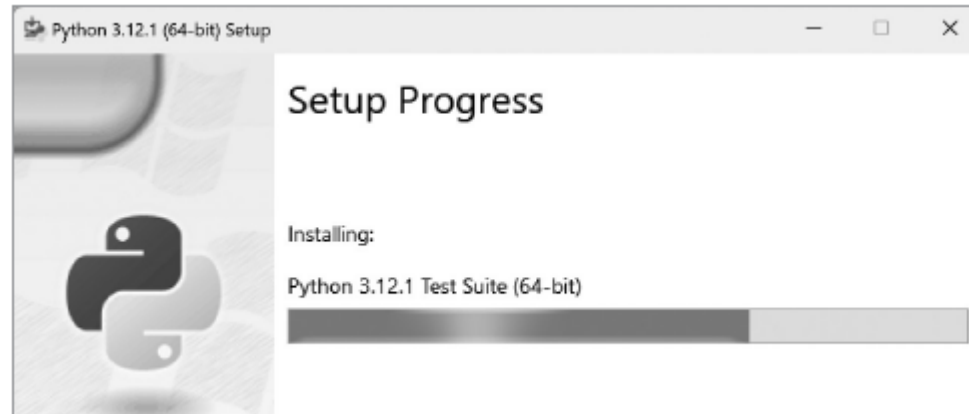
## ■ 파이썬 설치하기

3. 설치 프로그램의 첫 화면에서  
Add python.exe to PATH를 체크하고 [Install Now]를 클릭함



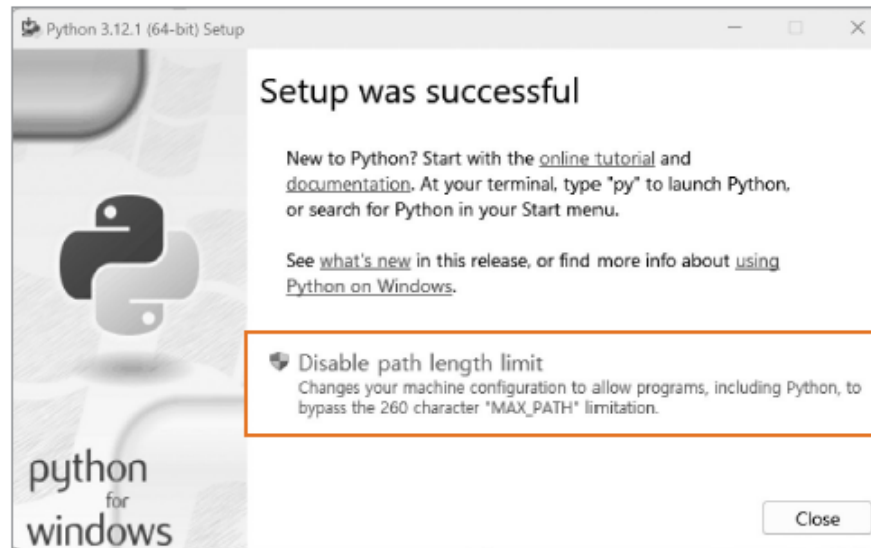
## ■ 파이썬 설치하기

### 4. Setup Progress 창이 등장하여 설치가 진행됨



## ■ 파이썬 설치하기

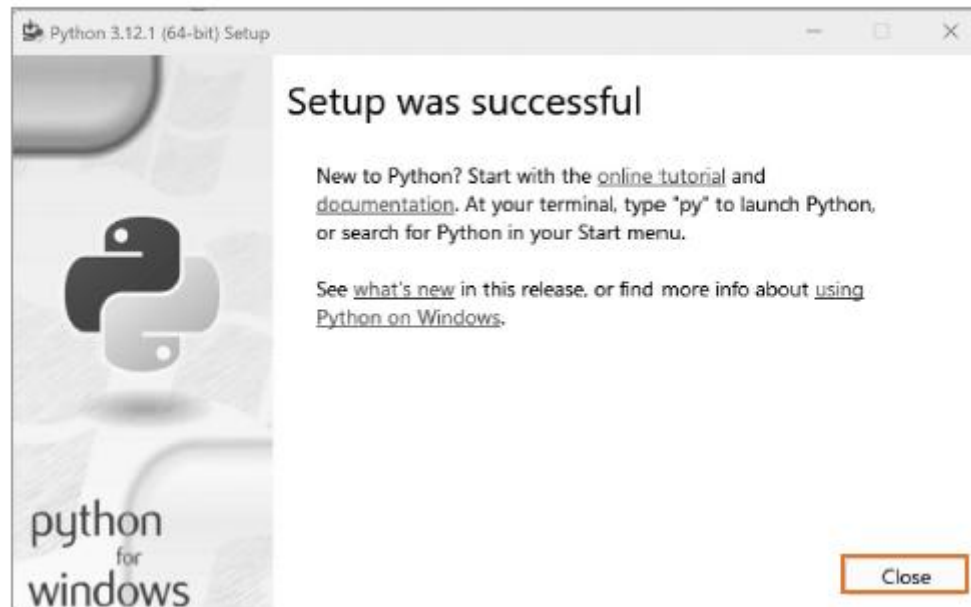
5. Setup was successful 창으로 바뀌면 [Disable path length limit]를 클릭한 뒤, '이 앱이 디바이스를 변경할 수 있도록 허용하시겠습니까?'라는 창이 등장하면 [예]를 클릭함



## 08-1 실습 환경 구성하기

### ■ 파이썬 설치하기

6. 다시 이 창으로 돌아오면 [Close]를 클릭하여 파이썬 설치를 마무리함





## ■ 파이썬 설치하기

7. 파이썬 설치가 완료되었으므로 시작 프로그램에서 Python 3.12 폴더와 그 하위에 설치된 프로그램 목록을 확인할 수 있음.  
여기에서 IDLE(Python 3.12 64-bit)를 클릭해 보자

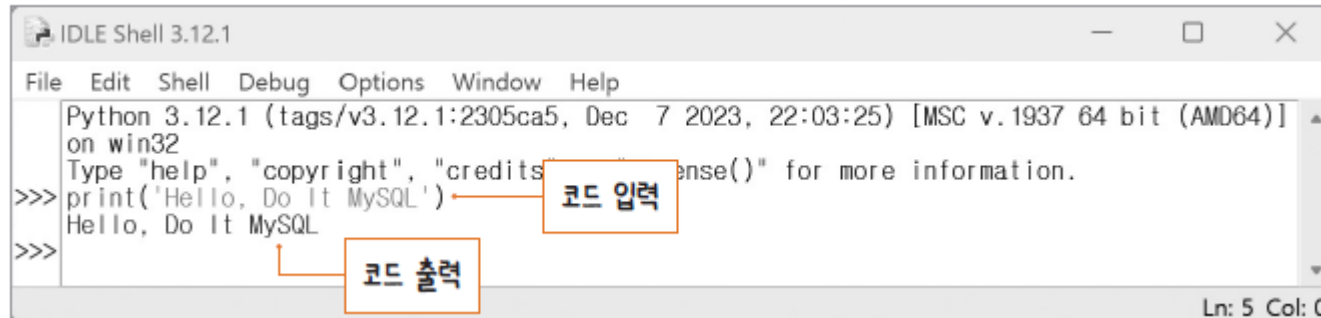


## ■ 파이썬 설치하기

8. IDLE Shell이 실행되면 파이썬이 제대로 실행되는지 확인하기 위해 간단히 다음 코드를 작성하고 실행해 보자

**Do it!** 파이썬 코드 입력

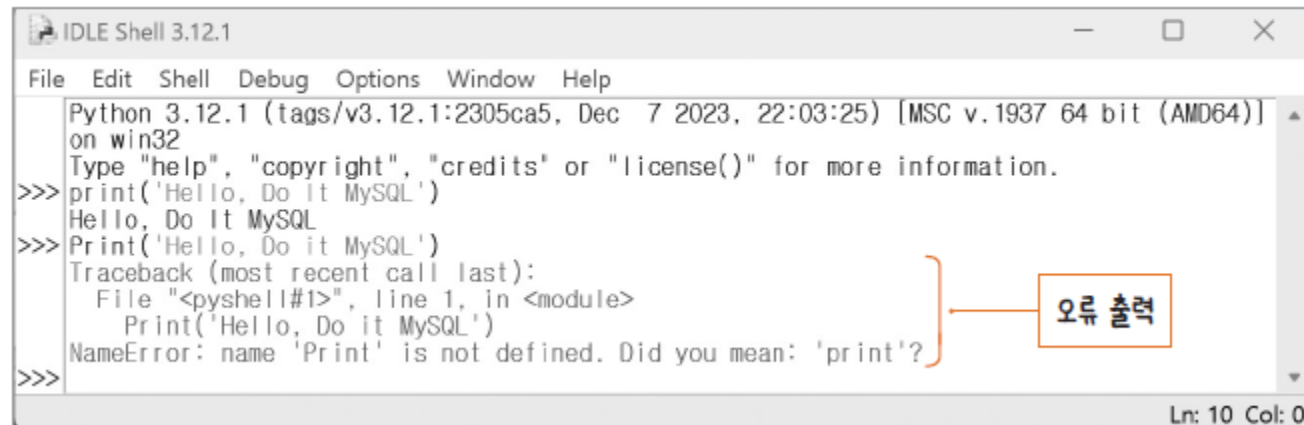
```
print('Hello, Do It MySQL')
```



```
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits", or "help()" for more information.
>>> print('Hello, Do It MySQL')
Hello, Do It MySQL
>>>
```

## ■ 파이썬 설치하기

9. 파이썬에서는 대소 문자, 띄어쓰기, 들여쓰기에 따라 코드가 다르게 해석되므로 실습 과정에서 반드시 주의해야 함. 다음과 같이 print라는 예약어를 'Print'로 작성했을 때 오류가 발생하는 것을 확인할 수 있음



```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('Hello, Do it MySQL')
Hello, Do it MySQL
>>> Print('Hello, Do it MySQL')
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    Print('Hello, Do it MySQL')
NameError: name 'Print' is not defined. Did you mean: 'print'?
>>>
```

오류 출력

## ■ 비주얼 스튜디오 코드 설치하기

- 파이썬을 설치할 때 함께 제공되는 IDLE Shell을 사용해 파이썬 프로그램을 구현할 수도 있지만,

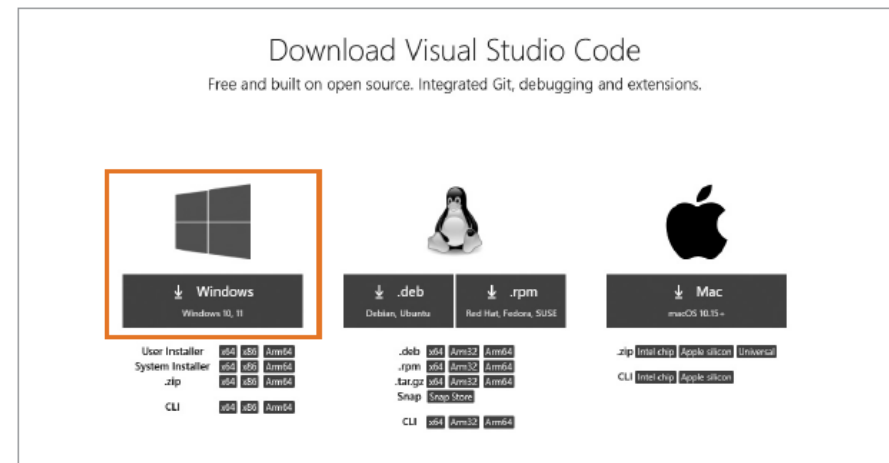
개발의 편의성을 위해 현업에서 많이 사용하는 무료 개발 도구인 비주얼 스튜디오 코드(Visual Studio Code, VS Code)를 설치해 보자

## ■ 비주얼 스튜디오 코드 설치하기

1. 비주얼 스튜디오 코드 설치 파일은 다음 링크를 통해 내려받을 수 있으며, 사용자 환경에 따라 알맞은 설치 파일을 선택함.  
여기서는 윈도우 환경을 기준으로 설명함

### 비주얼 스튜디오 코드 설치 파일 내려받기 링크

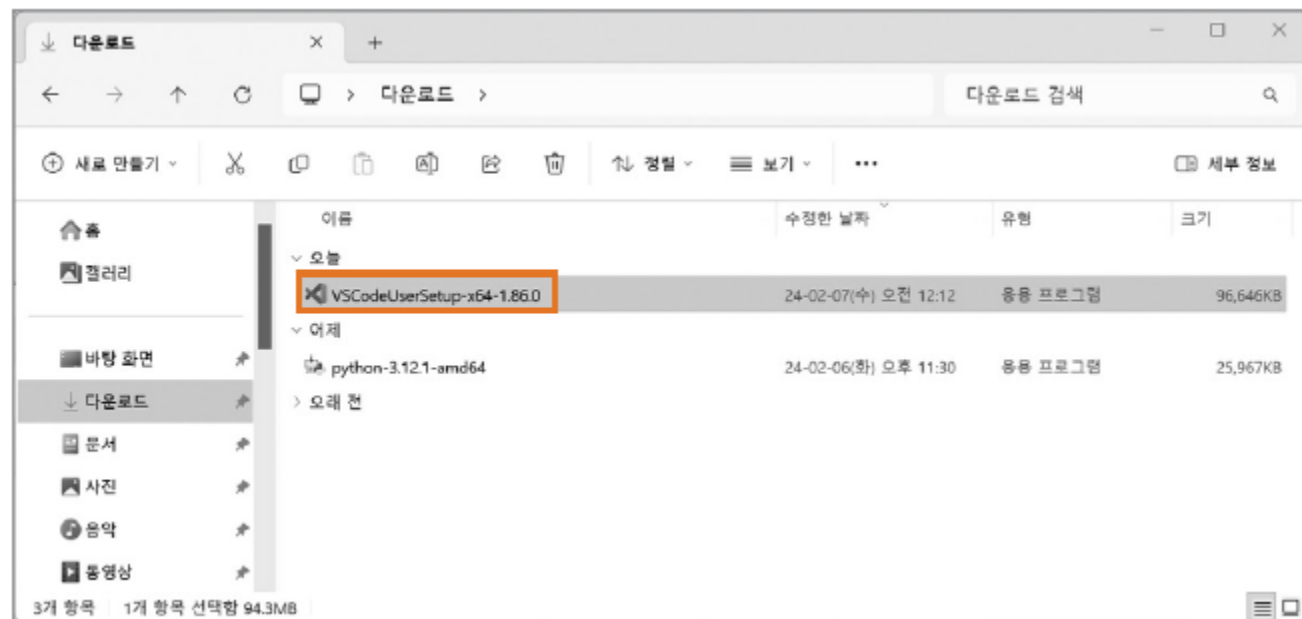
- <https://code.visualstudio.com/download>



## 08-1 실습 환경 구성하기

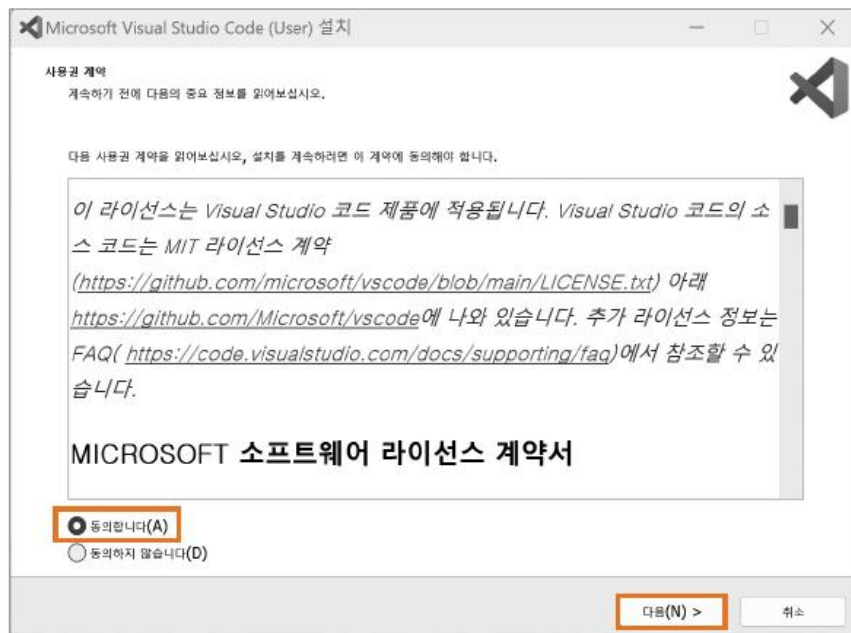
### ■ 비주얼 스튜디오 코드 설치하기

2. 내려받은 비주얼 스튜디오 코드 설치 파일을 더블클릭하여 설치를 시작함



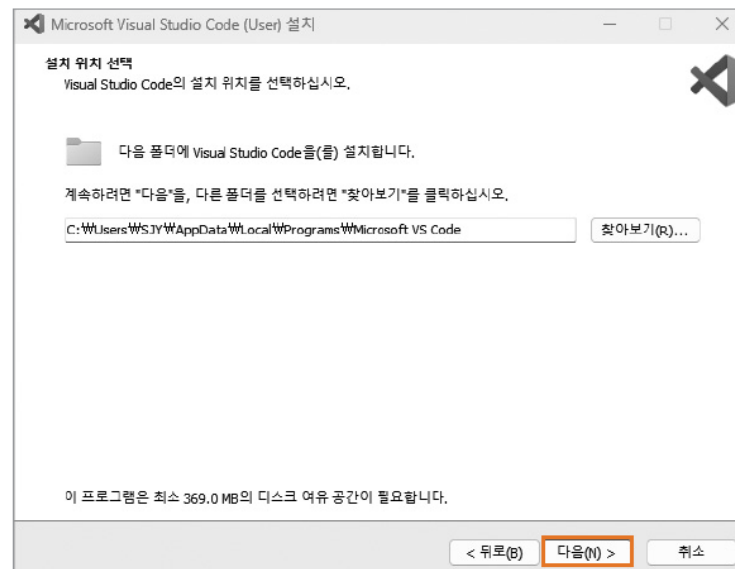
## ■ 비주얼 스튜디오 코드 설치하기

3. 사용권 계약 단계에서 [동의합니다(A)]를 선택한 후, [다음]을 클릭함



## ■ 비주얼 스튜디오 코드 설치하기

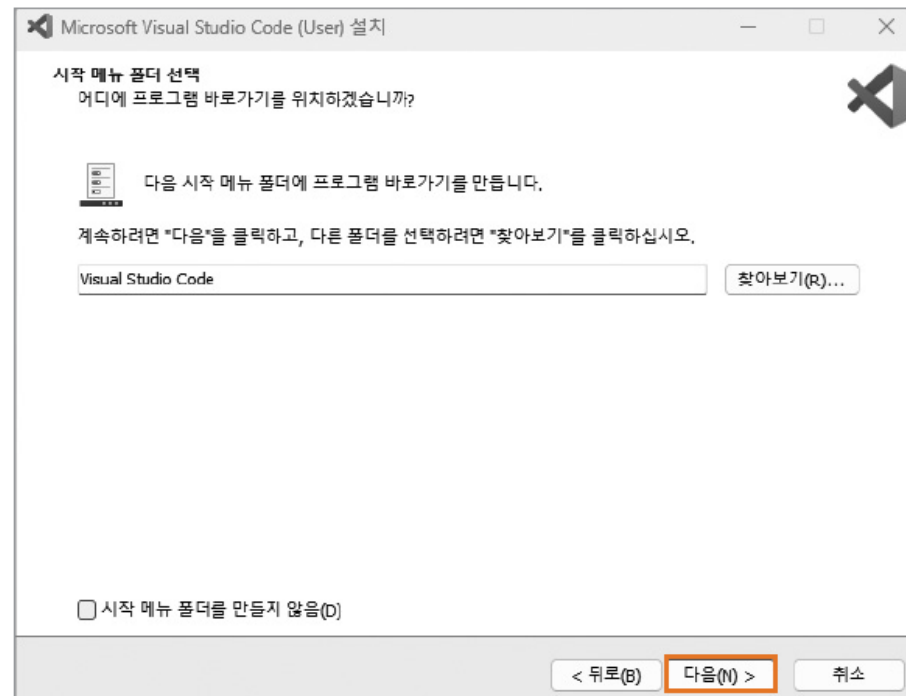
4. 설치 경로를 선택하는데 여기에서는 기본 경로를 그대로 사용함.  
[다음]을 클릭하여 설치를 계속 진행함





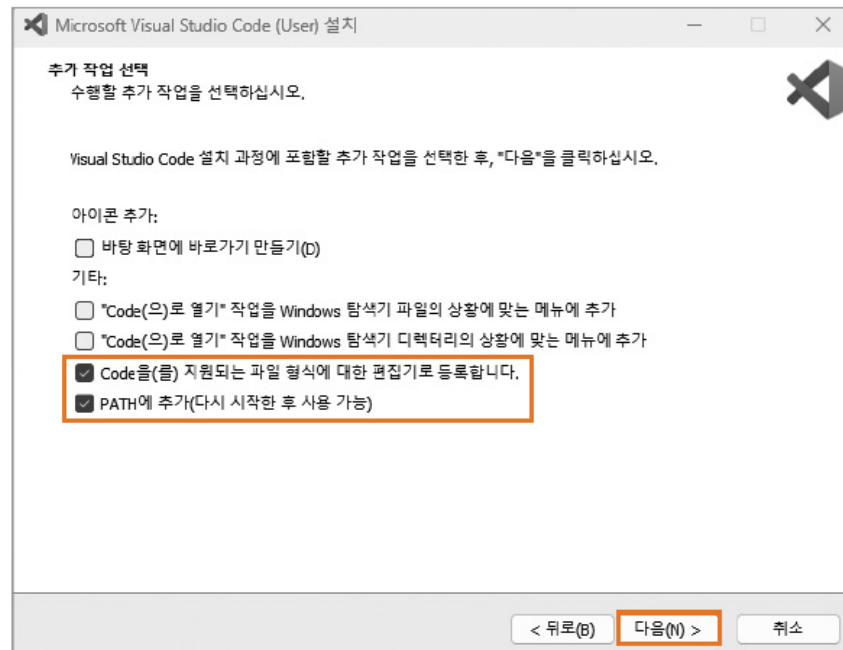
## ■ 비주얼 스튜디오 코드 설치하기

5. 시작 메뉴 폴더 선택 창에서도 기본 값을 그대로 유지하고 [다음]을 클릭함



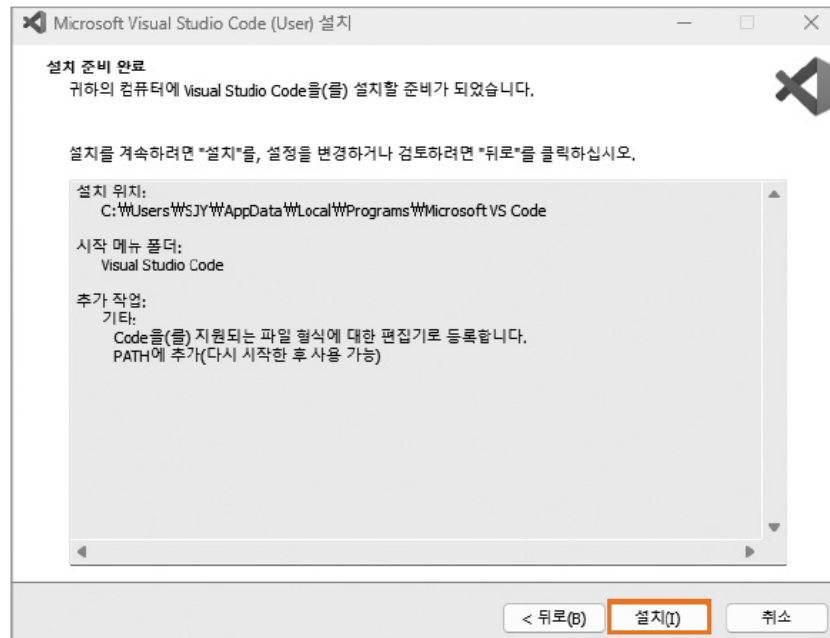
## ■ 비주얼 스튜디오 코드 설치하기

6. 추가 작업 선택 창에서는 다음과 같이 두 군데의 체크 박스를 선택하고 [다음]을 클릭 함



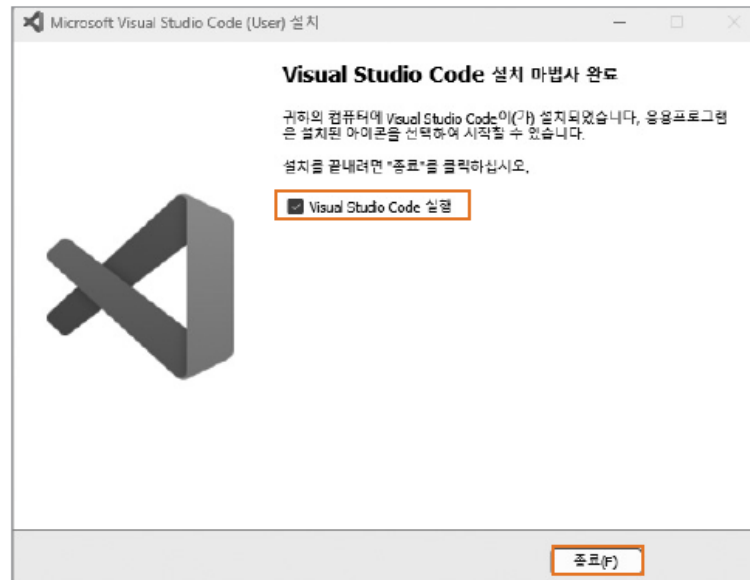
## ■ 비주얼 스튜디오 코드 설치하기

7. 여기까지 설치 준비가 완료되었으면 [설치] 버튼을 클릭함.  
이후 설치가 진행되는 창이 등장함



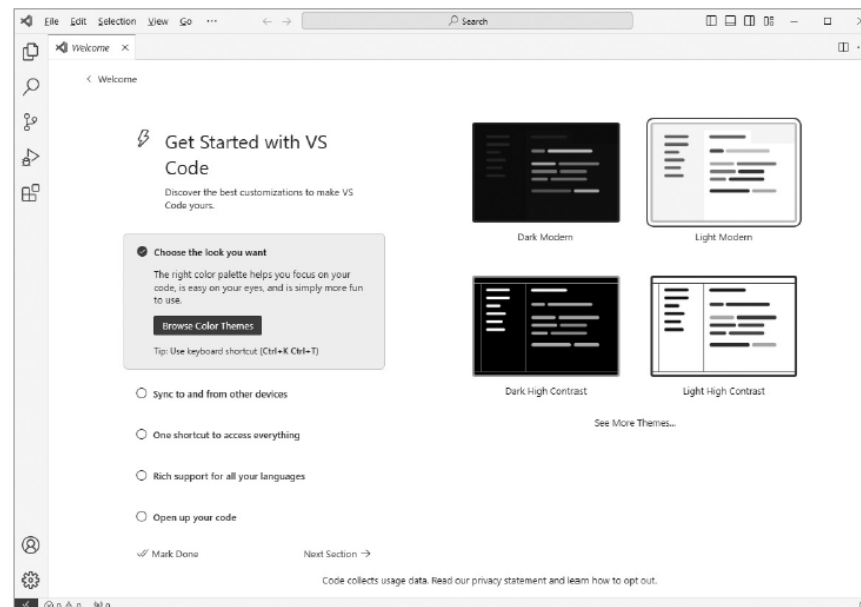
## ■ 비주얼 스튜디오 코드 설치하기

8. 프로그램을 정상으로 설치되면  
Visual Studio Code 설치 마법사 완료 창이 등장함.  
이 화면에서 [Visual Studio Code 실행]에 체크한 뒤, [종료]를 클릭함



## ■ 비주얼 스튜디오 코드 설치하기

9. 비주얼 스튜디오 코드가 실행되고,  
비주얼 스튜디오 코드의 테마를 선택하는 화면으로 넘어감



- 파이썬으로 주식 데이터를 수집하는 크롤러를 개발하고, MySQL에서는 크롤러를 통해 수집된 데이터를 저장하고 조작 및 가공할 수 있도록 시스템을 설계할 수 있음
- 하지만 이번 실습에서는 웹 크롤러가 아닌 야후 파이낸스(Yahoo Finance)에서 제공하는 API를 사용하여 데이터를 크롤링해 활용해 보자

## ▪ 주식 분석 시스템 구성하기

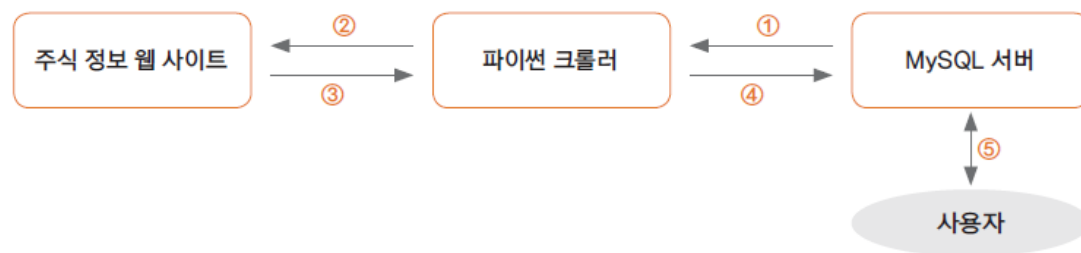
- 이번 실습에서 구성할 시스템의 구성도를 그려 보자.

우리가 구현해 볼 주식 분석 시스템은

**주식 정보가 담긴 웹 사이트,  
파이썬 크롤러,  
MySQL 서버**

이렇게 3가지 영역으로 나뉨

## ■ 주식 분석 시스템 구성하기



주식 분석 시스템의 데이터 흐름 구성도

- ① 파이썬 크롤러가 수집할 기업의 심벌 목록을 MySQL 서버에서 가져온다. 이 목록은 사용자가 원하는 기업의 이름을 미리 정의해 놓은 것이다. 주식에서는 여러 기업을 식별하기 위한 상징적인 문자로 심벌을 활용하는데 여기에서는 이러한 심벌로 수집할 기업의 목록을 구성한다.
- ② 파이썬 크롤러는 심벌 목록을 활용하여 주식 정보가 담긴 웹 사이트(여기서는 야후 파이낸스)의 정보를 크롤링한다.
- ③ 파이썬 크롤러가 요청한 데이터를 웹 사이트(여기서는 야후 파이낸스 API)로부터 전달받은 파이썬 크롤러는 DB 저장을 위해 데이터를 가공하고 코드를 실행한다.
- ④ 데이터베이스 테이블 구조에 맞춰 데이터를 저장한다.
- ⑤ 사용자는 데이터베이스에 접속하여 파이썬 크롤러를 통해 저장된 데이터를 다양하게 활용한다.



## ■ 파이썬 라이브러리 설치하기

- 주식 데이터를 크롤링하려면 수집한 데이터를 가공하는 라이브러리(패키지)와 데이터베이스 관련 라이브러리를 설치해야 함
- 파이썬 라이브러리는 다음 링크에서 검색하여 내려받아 사용할 수 있음

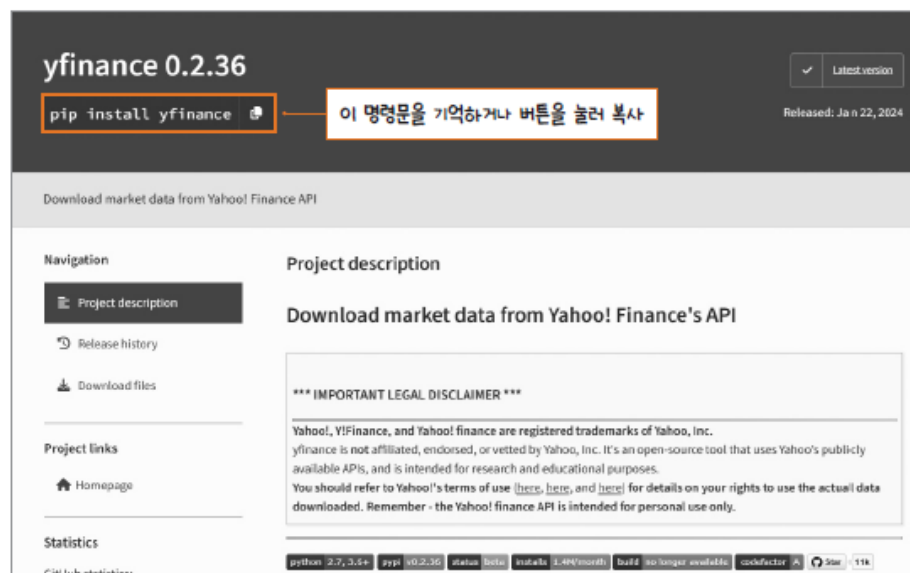
### 파이썬 라이브러리 내려받기 링크

• <https://pypi.org>

- 주식 정보가 담긴 웹 사이트로 야후 파이낸스(<https://finance.yahoo.com>)를 활용하려고 함
- 야후 파이낸스에서는 주식 데이터를 제공하는 파이썬 라이브러리를 배포하고 있어 이를 활용해 주식 데이터를 가져올 예정임

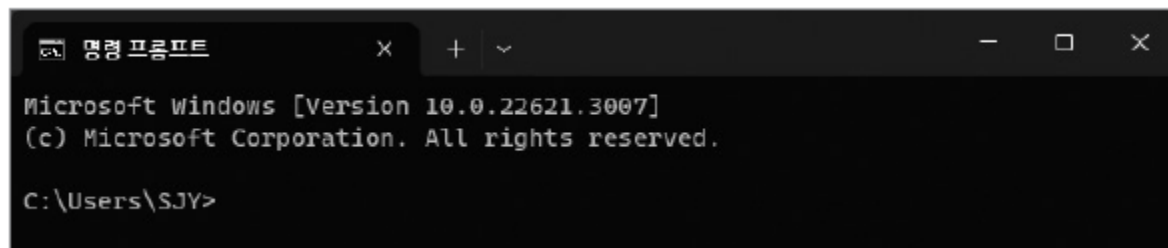
## ■ 파이썬 라이브러리 설치하기

1. 야후에서는 yfinance(<https://pypi.org/project/yfinance>)라는 라이브러리를 제공함.  
이 라이브러리는 앞에서 언급한 대로 야후 파이낸스의 주식 정보를 가져옴



## ■ 파이썬 라이브러리 설치하기

2. 파이썬 라이브러리는 필요할 때마다 따로 설치 작업을 진행해야 함.  
yfinance를 설치하기 위해서는 다음과 같이 명령 프롬프트를 실행함



```
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SJY>
```

2. 명령 프롬프트가 실행되면  
다음과 같은 라이브러리를 설치하기 위한 명령문을 입력해 보자.  
그러면 자동으로 패키지가 내려받아지며 설치되는 과정이 출력됨

**Do it!** yfinance 설치 명령

```
pip install yfinance
```

## ■ 파이썬 라이브러리 설치하기

3.

```
명령 프롬프트 - pip install yfmr x + v
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

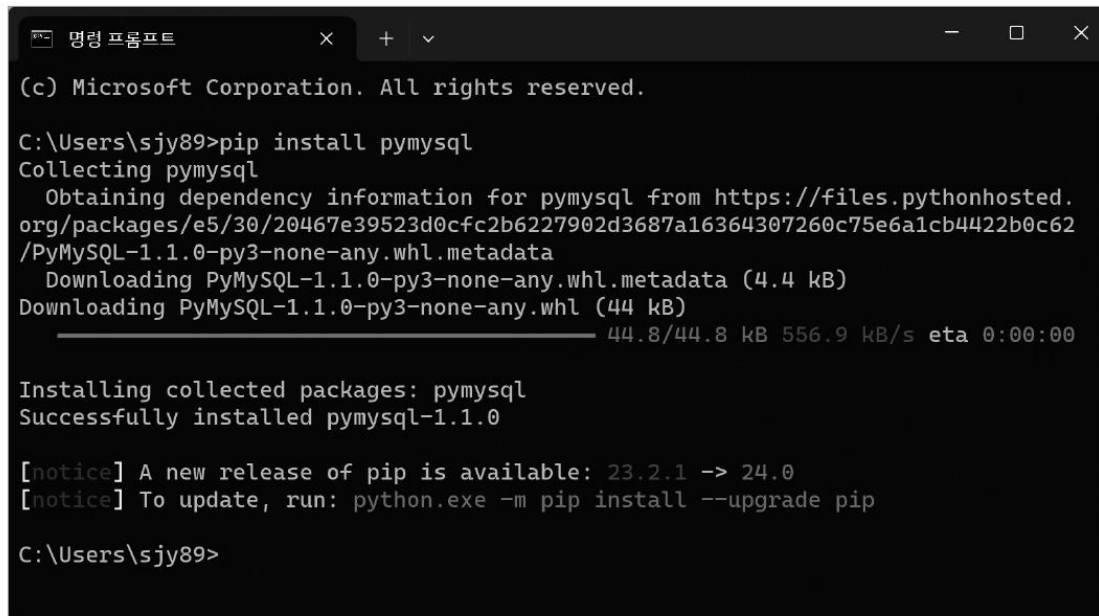
C:\Users\SJY>pip install yfinance
Collecting yfinance
  Downloading yfinance-0.2.36-py2.py3-none-any.whl.metadata (11 kB)
Collecting pandas>=1.3.0 (from yfinance)
  Downloading pandas-2.2.0-cp311-cp311-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.16.5 in c:\users\sjy\appdata\local\programs\python\python311\lib\site-packages (from yfinance) (1.26.4)
Collecting requests>=2.31 (from yfinance)
  Downloading requests-2.31.0-py3-none-any.whl.metadata (4.6 kB)
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Collecting lxml>=4.9.1 (from yfinance)
  Downloading lxml-5.1.0-cp311-cp311-win_amd64.whl.metadata (3.6 kB)
Collecting appdirs>=1.4.4 (from yfinance)
  Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Collecting pytz>=2022.5 (from yfinance)
  Downloading pytz-2024.1-py2.py3-none-any.whl.metadata (22 kB)
Collecting frozendict>=2.3.4 (from yfinance)
  Downloading frozendict-2.4.0.tar.gz (314 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 314.6/314.6 kB 19.0 MB/s eta 0:00:00
Installing build dependencies ... done
Getting requirements to build wheel ... done
Installing backend dependencies ... done
Preparing metadata (pyproject.toml) ... done
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.1.tar.gz (3.0 MB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.0/3.0 MB 94.9 MB/s eta 0:00:00
```

## ■ 파이썬 라이브러리 설치하기

4. 이번에는 파이썬에서 MySQL과 연동할 수 있도록 도와주는 pymysql 라이브러리를 설치해 보자.  
yfinance 패키지를 설치할 때와 마찬가지로 설치를 진행해 보자

**Do it!** pymysql 설치 명령

```
pip install pymysql
```



```
(c) Microsoft Corporation. All rights reserved.

C:\Users\sjsy89>pip install pymysql
Collecting pymysql
  Obtaining dependency information for pymysql from https://files.pythonhosted.org/packages/e5/30/20467e39523d0cfc2b6227902d3687a16364307260c75e6a1cb4422b0c62/PyMySQL-1.1.0-py3-none-any.whl.metadata
  Downloading PyMySQL-1.1.0-py3-none-any.whl.metadata (4.4 kB)
  Downloading PyMySQL-1.1.0-py3-none-any.whl (44 kB)
    44.8/44.8 kB 556.9 kB/s eta 0:00:00

Installing collected packages: pymysql
Successfully installed pymysql-1.1.0

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip

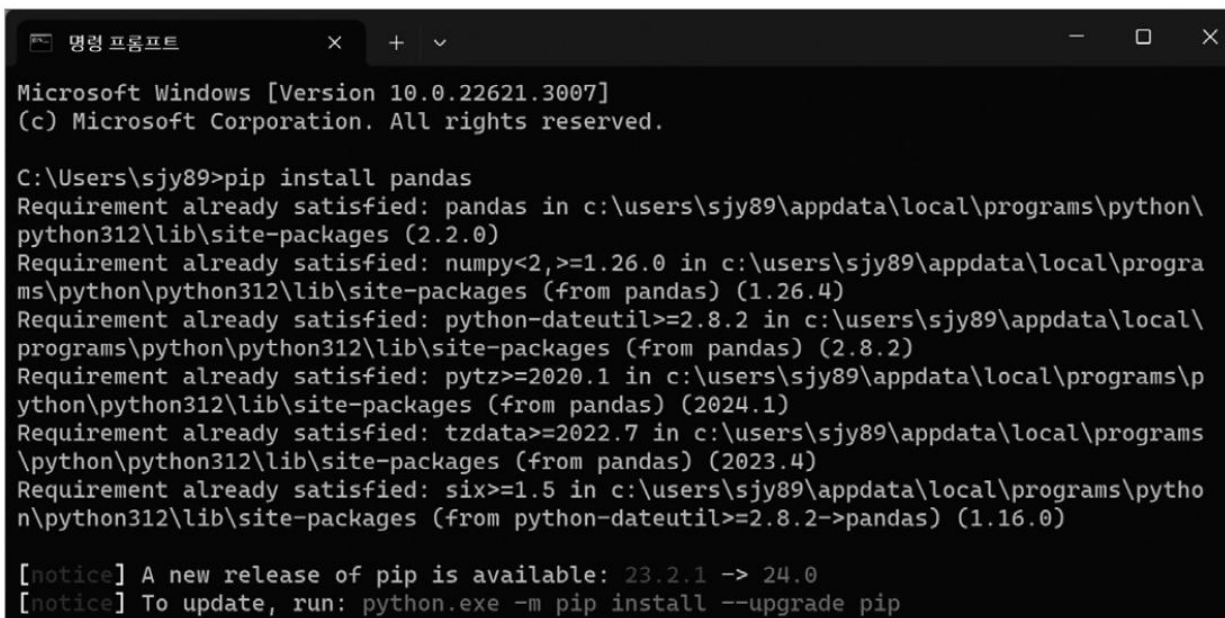
C:\Users\sjsy89>
```

## ■ 파이썬 라이브러리 설치하기

5. pandas는 관계형 또는 레이블이 지정된 데이터 작업을 쉽게 할 수 있도록 설계된 파이썬 라이브러리임.  
이 또한 yfinance 패키지를 설치 때와 마찬가지로 설치를 진행해 보자

**Do it!** pandas 설치 명령

```
pip install pandas
```



```
명령 프롬프트
Microsoft Windows [Version 10.0.22621.3007]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sjy89>pip install pandas
Requirement already satisfied: pandas in c:\users\sjy89\appdata\local\programs\python\python312\lib\site-packages (2.2.0)
Requirement already satisfied: numpy<2, >=1.26.0 in c:\users\sjy89\appdata\local\programs\python\python312\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\sjy89\appdata\local\programs\python\python312\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\sjy89\appdata\local\programs\python\python312\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\sjy89\appdata\local\programs\python\python312\lib\site-packages (from pandas) (2023.4)
Requirement already satisfied: six>=1.5 in c:\users\sjy89\appdata\local\programs\python\python312\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)

[notice] A new release of pip is available: 23.2.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

## ■ 데이터베이스 스키마와 기초 데이터 생성하기

- 파이썬 크롤러가 수집한 데이터를 저장할 수 있도록 테이블을 구성함

- 먼저, 실습에서 활용할 테이블의 ERD를 살펴보자

- 이 ERD는 기업 정보를 저장하는 nasdaq\_company 테이블과 1일 주식 정보를 저장하는 stock 테이블로 구성되어 있음

nasdaq_company	
symbol	심벌 이름
company_name	기업 이름
country	기업 국가
ipo_year	IPO 년도
sector	
industry	산업군
las_crawel_date_stock	마지막 크롤링한 날짜
open	시초가
high	일 최고가
low	일 최저가
close	종가
adj_close	시간 외 종가

stock	
date	주가 입력 날짜
symbol	심벌 이름
open	시초가
hign	일 최고가
low	일 최저가
close	종가
adj_close	시간 외 종가
volume	거래량

주식 분석 ERD

## ■ 데이터베이스 스키마와 기초 데이터 생성하기

- 실습에 사용할 데이터베이스, 테이블, 데이터를 차례로 생성해 보자

1. 먼저 데이터베이스를 생성해 보자.  
다음 쿼리를 통해 us\_stock라는 이름으로 데이터베이스를 생성함

**Do it!** 주식 분석을 위한 데이터베이스 생성

```
CREATE DATABASE us_stock;
```



## ■ 데이터베이스 스키마와 기초 데이터 생성하기

2. 테이블을 생성하는 DDL 문을 활용해  
테이블의 열 이름과 데이터 유형 그리고 인덱스 등을 설정함.  
이제 2개의 테이블을 만드는데, 먼저 nasdaq\_company 테이블을 생성하자.

### Do it! 주식 분석을 위한 nasdaq\_company 테이블 설정

```
USE us_stock;

CREATE TABLE nasdaq_company(
symbol VARCHAR(255),
company_name VARCHAR(255),
country VARCHAR (255),
ipo_year INT,
sector VARCHAR(255),
industry VARCHAR(255),
last_crawl_date_stock DATETIME,
is_delete VARCHAR(5),
open DECIMAL(18,2),
high DECIMAL(18,2),
low DECIMAL(18,2),
close DECIMAL(18,2),
adj_close DECIMAL(18,2),
volume BIGINT
```

## ■ 데이터베이스 스키마와 기초 데이터 생성하기

### 3. 이어서 stock 테이블을 생성해 보자

**Do it!** 주식 분석을 위한 stock 테이블 설정

```
USE us_stock;
```

```
CREATE TABLE stock(  
  date DATETIME,  
  symbol VARCHAR(255),  
  open DECIMAL(18,2),  
  high DECIMAL(18,2),  
  low DECIMAL(18,2),  
  close DECIMAL(18,2),  
  adj_close DECIMAL(18,2),  
  volume BIGINT  
);
```

```
CREATE INDEX ix_stock_1 ON stock(date,symbol);  
CREATE INDEX ix_stock_2 ON stock(symbol,date);
```

## ■ 데이터베이스 스키마와 기초 데이터 생성하기

### 4. 테이블을 모두 생성하였으니 이제 기초 데이터를 입력해 보자.

여기에서는 여건상 테슬라, 아마존, 엔디비아 등 12개의 기업 정보만 포함했지만  
야후 파이낸스에서는 7천 개가 넘는 기업의 정보를 제공함.

기업 수가 많으면 가져오는 데이터의 양도 많아지므로 주의해야 함

## ■ 데이터베이스 스키마와 기초 데이터 생성하기

4.

### Do it! 주식 분석을 위한 기초 데이터 삽입

```

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('TSLA', 'Tesla Inc. Common Stock', 2010, 'Capital Goods', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('MSFT', 'Microsoft Corporation Common Stock', 1986, 'Technology', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('AMZN', 'Amazon.com Inc. Common Stock', 1997, 'Consumer Services', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('AAPL', 'Apple Inc. Common Stock', 1980, 'Technology', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('INTC', 'Intel Corporation Common Stock', NULL, 'Technology', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('NVDA', 'NVIDIA Corporation Common Stock', 1999, 'Technology', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('AMD', 'Advanced Micro Devices Inc. Common Stock', NULL, 'Technology', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('META', 'Meta Platforms, Inc.', 2012, 'Technology', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('AMPG', 'AMPG, Inc.', 2012, '', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('CAR', 'CAR, Inc.', 2012, '', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('UAN', 'UAN, Inc.', 2012, '', 'K2');

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('BHR', 'BHR, Inc.', 2012, '', 'K2');

```

```

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)
VALUES ('TSLA', 'Tesla Inc. Common Stock', 2010, 'Capital Goods', 'K2');

```

```

INSERT INTO nasdaq_company (symbol, company_name, ipo_year, sector, industry)

```

## ■ 데이터베이스 스키마와 기초 데이터 생성하기

4.

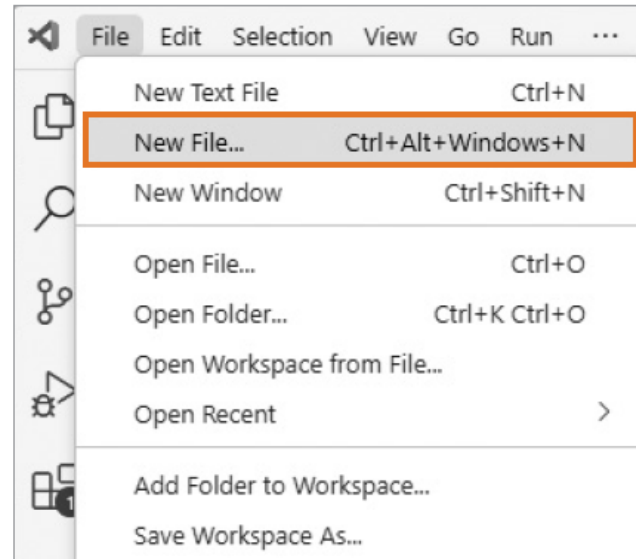
### 실행 결과

	symbol	company_name	country	ipo_year	sector	industry	last_crawel_date_stock	is_delete	open	high	low	dose	adj_dose	volume
▶	AAPL	Apple Inc. Common Stock	NULL	1980	Technology	K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	AMD	Advanced Micro Devices Inc. Common Stock	NULL	NULL	Technology	K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	AMPG	AMPG, Inc.	NULL	2012		K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	AMZN	Amazon.com Inc. Common Stock	NULL	1997	Consumer Services	K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	BHR	BHR, Inc.	NULL	2012		K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	CAR	CAR, Inc.	NULL	2012		K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	INTC	Intel Corporation Common Stock	NULL	NULL	Technology	K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	META	Meta Platforms, Inc.	NULL	2012	Technology	K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	MSFT	Microsoft Corporation Common Stock	NULL	1986	Technology	K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	NVDA	NVIDIA Corporation Common Stock	NULL	1999	Technology	K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	TSLA	Tesla Inc. Common Stock	NULL	2010	Capital Goods	K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
	UAN	UAN, Inc.	NULL	2012		K2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

## ■ 파이썬 크롤러 만들기

- 파이썬 크롤러는 앞에서 설치한 야후 파이낸스 API인 yfinance를 호출하여 데이터를 수집하고 가공하고, MySQL 서버로 데이터를 저장하는 역할을 함

1. 비주얼 스튜디오 코드를 실행하여 [File → New File]을 클릭함



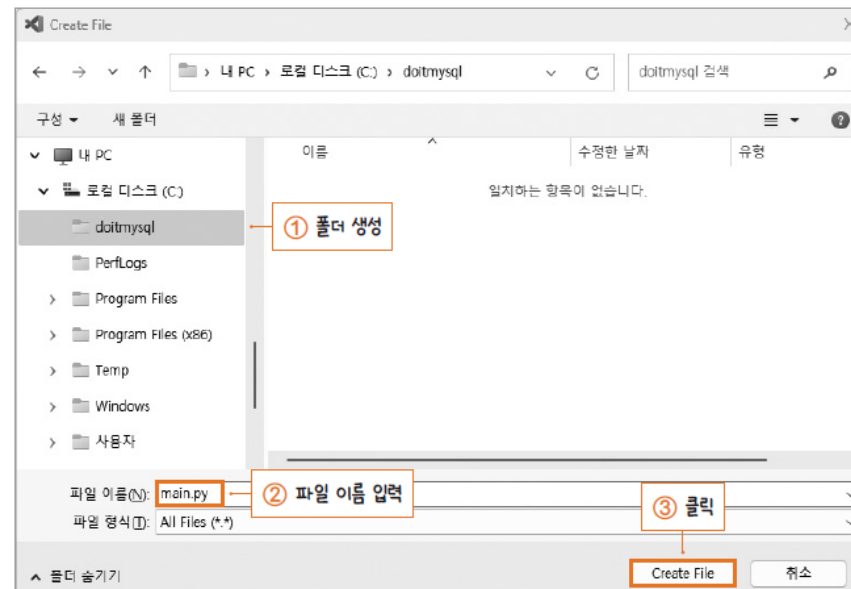
## ■ 파이썬 크롤러 만들기

2. 화면 상단에 New File 창이 나타나면 파일 이름으로 main.py를 입력함



## ■ 파이썬 크롤러 만들기

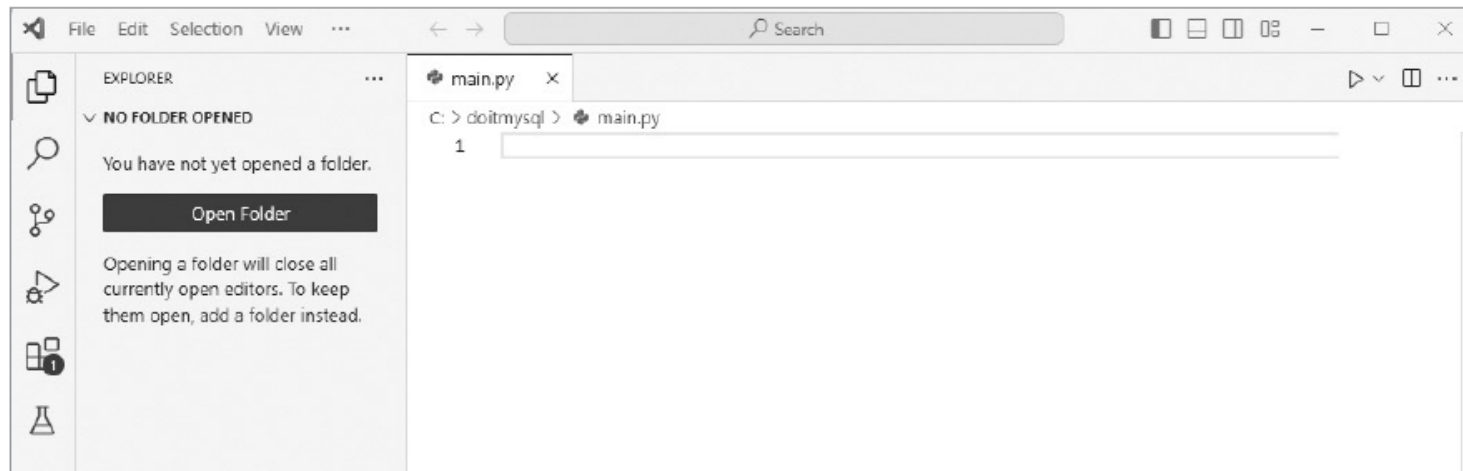
3. 파일을 저장할 곳을 물어보는 Create File 창이 나타남.  
C 드라이브에 'doitmysql'이라는 폴더를 생성한 뒤,  
main.py를 입력하고 [Create File] 버튼을 클릭함





## ■ 파이썬 크롤러 만들기

4. 파일이 생성되면 VS Code에 main.py 파일이 생성된 것을 확인할 수 있음.  
이로써 파이썬 크롤러를 만들기 위한 프로그래밍 준비 작업이 완료된 것임



## ■ 파이썬 크롤러 만들기

- 4. 앞으로 설명하는 파이썬 코드들은 모두 main.py 파일에 이어서 작성할 것임
- 4. 설치한 파이썬 패키지들을 импорт(import)해 보자

### Do it! 파이썬 패키지 импорт

```
1: from datetime import datetime, timedelta
2:
3: import pymysql
4: import pandas as pd
5: import yfinance as yf
```

## ■ 파이썬 크롤러 만들기

### 5. 파이썬에서 사용할 모듈을 정의하는 단계임.

앞에서 설치했던 모듈(라이브러리)을 여기에 불러 사용함.

import 문 뒤의 as는 SQL과 마찬가지로 별칭을 정의하는 명령문으로, 긴 모듈명을 간단하게 사용하기 위해서 이와 같이 정의함

## ■ 파이썬 크롤러 만들기

6. 이번에는 MySQL 접속 정보를 입력함.  
데이터베이스와 파이썬이 함께 실행될 파일이  
같은 서버에 위치하므로 여기서는 localhost를 입력함.

### Do it! MySQL 접속 정보 입력

```
11:     mysql_conn      =     pymysql.connect(host='localhost',      user='root',  
password='doitmysql',  
db='us_stock')
```

MySQL과의 연결 정보를 이와 같이 정의할 수 있음.  
여기서 mysql\_conn이라는 변수에 데이터를 저장하려는 MySQL 서버의 정보를 할당함

## ■ 파이썬 크롤러 만들기

### 7. 먼저 크롤링할 기업의 목록을 데이터베이스로 읽어 오는 함수를 생성해 보자

**Do it!** 크롤링한 데이터를 DB로 읽어오는 함수 생성

```
14: def getCompany():
15:
16:     mysql_cur = mysql_conn.cursor()
17:
18:     today = datetime.today() + timedelta(days=1)
19:
20:     try:
21:         mysql_cur.execute("select symbol, company_name, ipo_year, last_crawl_
date_stock from us_stock.nasdaq_company where is_delete is null;")
22:         results = mysql_cur.fetchall()
23:         print(results)
24:
25:
26:     for row in results:
27:         _symbol = row[0]
28:         _company_name = row[1]
29:
30:         if row[2] is None or row[2] == 0:
31:             _ipo_year = '1970'
32:         else:
33:             _ipo_year = row[2]
34:
35:         if row[3] is None:
36:             _last_crawl_date_stock = str(_ipo_year) + '-01-01'
37:         else:
```

## ■ 파이썬 크롤러 만들기

7. 14번째 줄의 def 문으로는 함수를 생성할 수 있음.

getCompany의 괄호 안에는 매개변수를 정의할 수 있지만  
현재 함수는 매개변수를 사용하지 않기 때문에 빈 괄호로 둬.

16번째 줄의 코드를 통해 mysql 데이터베이스와 연결하는 커서를 엮.

18번째 줄의 코드로는 날짜를 정의하는데,  
파이썬 크롤러가 실행될 때의 날짜에 + 1일로 하여 today라는 변수에 저장됨.

21번째 줄의 코드에서 mysql\_cur.execute() 함수는  
매개변수로 사용된 쿼리를 실행하는 함수임

## ■ 파이썬 크롤러 만들기

7. 22번째 줄의 코드에서 `mysql_cur.fetchall()` 함수는 21번째 줄에서 실행한 코드의 결과를 읽어오는 함수임.

26 ~ 47번째 줄의 코드는 `results` 변수에 저장된 데이터를 행 단위로 읽어 `_symbol`과 같이 열 이름으로 만든 각각의 변수에 값(데이터)을 할당함.

만약 새로 추가된 심벌로 `nasdaq_company` 테이블의 `last_crawl_date_stock` 열의 값이 `NULL`일 경우 크롤링했던 기록이 없다는 뜻이므로,

과거의 모든 데이터를 가져오기 위해 1970년부터 읽어 올 수 있도록 코드를 작성하고,

## ■ 파이썬 크롤러 만들기

7. 크롤링 기록이 있으면 마지막 크롤링한 날짜부터 최근 날짜까지의 데이터를 조회할 수 있도록 함.

47번째 줄의 코드는 더 자세히 살펴보자.

실제 주식을 가져오는 함수인 `getStock`을 호출하며,  
이때 매개변수로 변수를 전달함.

50 ~ 55번째 코드는 프로그램에 오류가 발생했을 때  
오류를 출력하고 이 코드 실행을 종료하기 위한 내용을 담고 있음



## ■ 파이썬 크롤러 만들기

### 8. 이어서 실제 주식 데이터를 가져오는 함수를 생성함

**Do it!** 주식 데이터를 가져오는 함수 생성

```
58: def getStock(_symbol, _start_date, _end_date):
59:
60:     mysql_cur = mysql_conn.cursor()
61:
62:     mysql_cur.execute("delete from us_stock.stock where date >= %s and date
        <= %s and symbol = %s", (_start_date, _end_date, _symbol))
63:     mysql_conn.commit()
64:
65:     try:
66:         stock_price = yf.download(_symbol, start=_start_date, end=_end_date)
67:         print(stock_price)
68:
69:         for index, row in stock_price.iterrows():
70:             _date = index.strftime("%Y-%m-%d")
71:             _open = str(row["Open"])
72:             _high = str(row["High"])
73:             _low = str(row["Low"])
74:             _close = str(row["Close"])
75:             _adj_close = str(row["Adj Close"])
76:             _volume = str(row["Volume"])
77:
78:             mysql_cur.execute("insert into us_stock.stock (date, symbol,
```

## ■ 파이썬 크롤러 만들기

8. 58번째 줄의 코드로는 함수를 정의함.  
이때, 3개의 매개변수값을 입력받을 수 있도록 정의함.

62번째 줄의 코드를 통해 중복된 값을 저장하지 않기 위해 크롤링하려는 날짜의 데이터가 존재하면 삭제함.

66번째 줄의 코드를 통해 yf.download라는 함수를 호출해 주식 데이터를 가져옴.

\_symbol은 심벌 이름을 할당하고  
\_start\_date와 \_end\_date는 수집할 날짜의 시작과 끝 날짜를 할당함.

함수를 호출한 결과는 stock\_price 변수에 저장함

## ■ 파이썬 크롤러 만들기

8. 69 ~ 76번째 줄의 코드는 stock\_price 변수에 저장된 데이터를 행 단위로 읽으면서 \_date, \_open과 같은 변수에 값을 할당함.

for index, row in stock\_price.iterrows():  
stock\_price라는 변수에 데이터프레임 형태로 저장된 결과를 한 행씩 읽으면서 각 변수에 값을 할당함.

\_date는 데이터의 날짜를 YYYY-MM-DD 형태로 값을 할당하고,  
이후 나머지 변수에 오른쪽의 결괏값인  
시초가, 최고가, 최저가, 종가 등의 값을 각각 할당함

### ■ 파이썬 크롤러 만들기

8. 78 ~ 79번째 줄의 코드로 stock 테이블에 크롤링한 데이터를 입력함.

82 ~ 83번째 줄의 코드로는 크롤링한 데이터의 마지막 날짜를 기록하여,  
다음 크롤링 시 참고할 수 있도록 nasdaq\_company 테이블에 업데이트함.

85 ~ 90번째 줄의 코드는 프로그램에 오류가 발생했을 때  
오류를 출력하고 로직을 종료함

## ■ 파이썬 크롤러 만들기

9. main.py 파일을 실행할 때 처음 실행될 함수로 getCompany()를 정의하는 코드를 마지막으로 작성해 보자.

**Do it!** main.py 파일을 실행

```
93: if __name__ == '__main__':  
94: getCompany()
```

이렇게 코드를 모두 작성하였으면  
**Ctrl + S** 또는 VSCode 상단의 메뉴에서  
[File → Save] 를 클릭하여 코드를 저장함

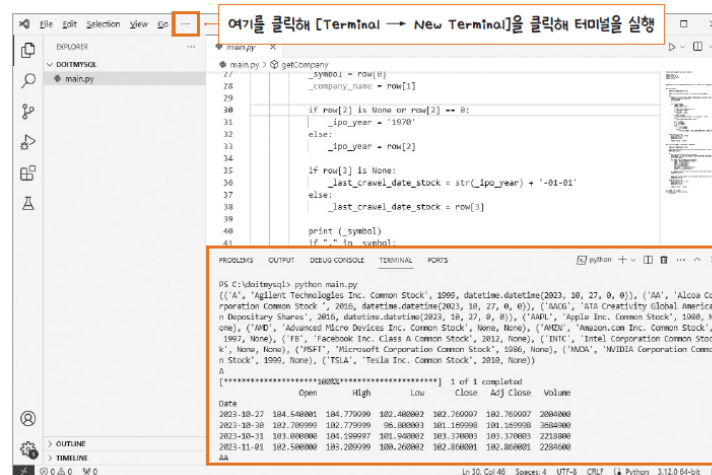
## ■ 파이썬 크롤러 만들기

10. 파이썬으로 제작한 크롤러인 main.py를 실행하여 주식 데이터를 데이터베이스에 저장해 보자.

VS Code 상단 메뉴에서 터미널을 실행하여 다음과 같이 터미널 환경에서 main.py를 실행함.

**Do it!** VS Code 터미널에서 크롤러 실행

python main.py



## ■ 파이썬 크롤러 만들기

10. 프로그램이 실행되면 각 기업의 목록을 하나씩 호출하며  
주식 데이터를 가져오는 화면을 볼 수 있음.  
이 데이터들은 모두 us\_stock 데이터베이스로 저장됨
10. 주식 데이터가 정상적으로 데이터베이스에 저장되었는지 확인함.  
이때 테이블 전체의 데이터를 조회하면 데이터양이 많을 수 있으므로  
상위 10개의 데이터만 조회하는 쿼리를 입력하여 데이터를 살펴보자

**Do it!** 수집된 주식 데이터 일부를 확인

```
SELECT * FROM stock LIMIT 10;
```

## ■ 파이썬 크롤러 만들기

11.

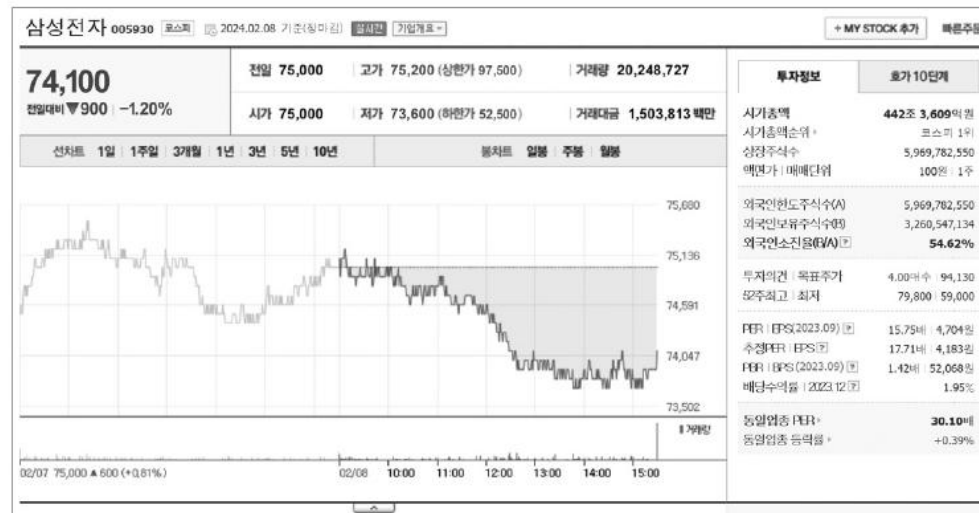
실행 결과

	date	symbol	open	high	low	dose	adj_dose	volume
▶	1980-12-12 00:00:00	AAPL	0.13	0.13	0.13	0.13	0.10	469033600
	1980-12-15 00:00:00	AAPL	0.12	0.12	0.12	0.12	0.09	175884800
	1980-12-16 00:00:00	AAPL	0.11	0.11	0.11	0.11	0.09	105728000
	1980-12-17 00:00:00	AAPL	0.12	0.12	0.12	0.12	0.09	86441600
	1980-12-18 00:00:00	AAPL	0.12	0.12	0.12	0.12	0.09	73449600
	1980-12-19 00:00:00	AAPL	0.13	0.13	0.13	0.13	0.10	48630400
	1980-12-22 00:00:00	AAPL	0.13	0.13	0.13	0.13	0.10	37363200
	1980-12-23 00:00:00	AAPL	0.14	0.14	0.14	0.14	0.11	46950400
	1980-12-24 00:00:00	AAPL	0.15	0.15	0.15	0.15	0.11	48003200
	1980-12-26 00:00:00	AAPL	0.16	0.16	0.16	0.16	0.12	55574400

주식 데이터의 날짜와 기업을 식별하기 위한 심벌,  
그리고 각 날짜의 시초가, 일 최고가, 일 최저가,  
종가, 시간 외 종가, 거래량 데이터를 확인할 수 있음



- 지금까지 공부한 내용과 직접 개발한 주식 데이터 크롤러를 활용해 주식 데이터를 분석해 보자
- 다음은 네이버에서 삼성전자를 검색하면 볼 수 있는 주식 정보 화면임



삼성전자 주식 정보

- 주식 정보 화면에는 가격 변동에 따른 차트와 함께 현재가, 전일 대비 등락가, 52주 최고가, 최저가 등 여러 가지 정보가 표시됨
- 그런데 이 화면은 기본적인 주식 정보만 나타냄
- 만약 다음 정보를 알고 싶다면 어떻게 해야 할까?

- 여러 주식 중에서 오늘 하루 상한가를 기록한 주식
- 여러 주식 중에서 52주 최고가를 기록한 상위 10개의 주식
- 등락 폭이 가장 큰 주식
- 기름값과 자동차 관련 주식의 상관 관계

- 이런 정보는 포털 사이트나 주식 정보 사이트에서 찾기 어려움
- 하지만 우리는 주식 데이터를 크롤링할 수도 있고  
이 데이터를 데이터베이스에 저장할 수도 있음
- 지금까지 배운 SQL을 활용하면 주식 정보를 얻을 수 있을 뿐 아니라  
내가 원하는 대로 주식을 분석할 수 있음

### ■ 52주 동안의 주가 분석하기

- 주식에서 '52주 최고가'란 1년 동안 가장 높은 가격을 갱신했다는 뜻이고, '52주 최저가'란 1년 동안 가장 낮은 가격을 갱신했다는 뜻임
- 52주 최저가, 최고가를 확인해 보고 최저가와 최고가 차이는 얼마나 나는지, 가격이 상승했다면 어느 정도인지 알아보자
- 다음 코드는 2023년 10월 04일 기준으로 52주간의 최저가 및 최고가를 구하고, 가격의 차이와 최저가 대비 최고가 상승 비율을 구함
- 데이터를 수집하는 시점에 따라 이 쿼리에서 날짜를 수정해서 사용해도 무방함

## ■ 52주 동안의 주가 분석하기

**Do it!** 52주간의 주식 최저가와 최고가 조회

```
SELECT
    symbol,
    CAST(MIN(close) AS DECIMAL(18,2)) AS w52_min,
    CAST(MAX(close) AS DECIMAL(18,2)) AS w52_max,
    CAST(MAX(close) - MIN(close) AS DECIMAL(18,2)) AS
`w52_diff_price($)` ,
    CAST((MAX(close) - MIN(close)) / MIN(close) * 100 AS DECIMAL(18,2))
AS `w52_
diff_ratio(%)`
FROM stock
WHERE date >= DATE_ADD('2023-10-04', INTERVAL -52 week) AND date <=
'2023-10-04'
```

■ stock 테이블에 저장된 데이터에서

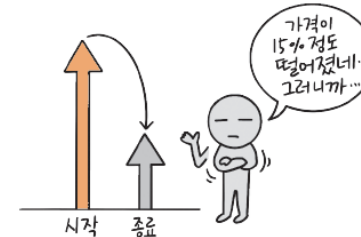
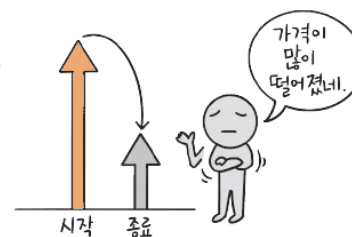
2023년 10월 4일 기준으로 앞의 52주 동안 데이터를 분석하여  
해당 기간 사이의 최고가와 최저가를 검색하고,  
최저가 대비 최고가 상승값과 상승 비율을 보여줌

실행 결과

	symbol	w52_min	w52_max	w52_diff_price(\$)	w52_diff_ratio(%)
▶	AAPL	125.02	196.45	71.43	57.13
	AMD	55.94	129.19	73.25	130.94
	AMZN	81.82	144.85	63.03	77.03
	INTC	24.90	38.86	13.96	56.06
	META	88.91	325.48	236.57	266.08
	MSFT	214.25	359.49	145.24	67.79
	NVDA	112.27	493.55	381.28	339.61
	TSLA	108.10	293.34	185.24	171.36

## ■ 하루 동안의 종목 변화 분석하기

- 주식 투자에 도전하기 전에 가장 먼저 할 일은 어떤 주식에 투자해야 큰 이익을 볼 수 있을지 알아보는 것임
- 하루 동안 가격이 상승한 종목과 하락한 종목이 무엇인지, 가격이 상승한 종목은 얼마나 올랐는지, 하락한 종목은 얼마나 떨어졌는지를 알고 있다면 다음 투자 전략을 세우기가 좋음
- 다음 그림처럼 추상적으로 주식을 분석하는 것보다는 구체적으로 얼마큼 주가가 하락했는지를 분석해야 함



## ■ 하루 동안의 종목 변화 분석하기

- 여기서는 하루 주식의 시초가, 종가, 거래 최대가, 거래 최소가 등을 조회하며 주식을 분석해 봄
- 참고로 우리나라 주식 시장은 하루 동안의 주가 변동 폭을 30%로 제한함
- 하지만 미국 주식 시장은 주가 변동 폭의 제한이 없음
- 우리가 활용하려는 실습 데이터는 미국 주식이므로 상승 종목과 하락 종목의 가격 차이가 매우 클 수 있음을 인지하자

## ▪ 하루 동안의 종목 변화 분석하기

1. 다음은 08-2에서 생성한 stock 테이블에서 하루 시초가와 종가를 비교해 상승한 금액, 비율과 하루 거래 중 최저 거래가와 최대 거래가의 차이를 구하는 쿼리임



## ■ 하루 동안의 종목 변화 분석하기

1.

**Do it!** 1일간의 시초가와 종가를 비교한 정보 조회

```
SELECT
    date,
    symbol,
    CAST(open AS DECIMAL(18,2)) AS open,
    CAST(close AS DECIMAL(18,2)) AS close,
    CAST((open - close) AS DECIMAL(18,2)) AS `diff_price($)`,
    CAST(((close - open) / open * 100) AS DECIMAL(18,2)) AS `diff_ratio(%)`,
    " AS '---',
    CAST(low AS DECIMAL(18,2)) AS low,
    CAST(high AS DECIMAL(18,2)) AS high,
    CAST((high - low) AS DECIMAL(18,2)) AS `diff_high_price($)`,
    CAST(((high - low) / low * 100) AS DECIMAL(18,2)) AS `diff_high_ratio(%)`
FROM stock
WHERE date = '2023-10-04';
```

## ■ 하루 동안의 종목 변화 분석하기

1.

실행 결과

	date	symbol	open	close	diff_price(\$)	diff_ratio(%)	---	low	high	diff_high_price(\$)	diff_high_ratio(%)
▶	2023-10-04 00:00:00	AAPL	171.09	173.66	-2.57	1.50		170.97	174.21	3.24	1.90
	2023-10-04 00:00:00	AMD	100.65	104.07	-3.42	3.40		100.34	104.40	4.06	4.05
	2023-10-04 00:00:00	AMZN	126.06	127.00	-0.94	0.75		125.68	127.36	1.68	1.34
	2023-10-04 00:00:00	INTC	36.52	35.93	0.59	-1.62		35.33	36.52	1.19	3.37
	2023-10-04 00:00:00	META	298.73	305.58	-6.85	2.29		298.50	306.90	8.40	2.81
	2023-10-04 00:00:00	MSFT	314.03	318.96	-4.93	1.57		314.00	320.04	6.04	1.92
	2023-10-04 00:00:00	NVDA	437.42	440.41	-2.99	0.68		432.92	441.43	8.51	1.97
	2023-10-04 00:00:00	TSLA	248.14	261.16	-13.02	5.25		247.60	261.86	14.26	5.76

2023년 10월 4일에 해당하는 주식 데이터에서  
일 최고가와 최저가 대한 금액을 조회하고,  
최저가 대비 최고가 상승 값과 상승 비율을 보여줌

## ■ 하루 동안의 종목 변화 분석하기

2. 이번에는 하루 동안 가격이 10% 이상 오른 종목들을 상승률 기준으로 내림차순으로 조회하는 쿼리임

**Do it!** 10% 이상 가격이 오른 종목 조회

```
SELECT
    date,
    symbol,
    CAST(open AS DECIMAL(18,2)) AS open,
    CAST(close AS DECIMAL(18,2)) AS close,
    CAST((open - close) AS DECIMAL(18,2)) AS `diff_price($)` ,
    CAST((((close - open) / open * 100) AS DECIMAL(18,2)) AS `diff_ratio(%)` ,
    " AS '---'",
    CAST(low AS DECIMAL(18,2)) AS low,
    CAST(high AS DECIMAL(18,2)) AS high,
    CAST((high - low) AS DECIMAL(18,2)) AS `diff_high_price($)` ,
    CAST((((high - low) / low * 100) AS DECIMAL(18,2)) AS `diff_high_ratio(%)`
FROM stock
WHERE date = '2022-02-24'
      AND CAST((((close - open) / open * 100) AS DECIMAL(18,2)) >= 10
ORDER BY CAST((((close - open) / open * 100) AS DECIMAL(18,2)) DESC;
```

## 08-3 MySQL로 주식 분석하기

### ■ 하루 동안의 종목 변화 분석하기

2.

실행 결과

	date	symbol	open	close	diff_price(\$)	diff_ratio(%)	---	low	high	diff_high_price(\$)	diff_high_ratio(%)
▶	2022-02-24 00:00:00	CAR	141.87	167.76	-25.89	18.25		141.79	169.33	27.54	19.42
	2022-02-24 00:00:00	TSLA	233.46	266.92	-33.46	14.33		233.33	267.49	34.16	14.64
	2022-02-24 00:00:00	NVDA	210.15	237.48	-27.33	13.00		208.90	238.00	29.10	13.93
	2022-02-24 00:00:00	AMD	104.56	116.61	-12.05	11.52		104.26	116.96	12.70	12.18

결과를 살펴보면 일 거래 가격에서  
최저가 대비 최고가가 10% 이상 오른 종목들을 확인할 수 있음

## ■ 전일 대비 종목의 변화 분석하기

- 이번에는 분석 범위를 한 단계만 넓혀서  
전일 대비 상승 또는 하락한 종목을 검색해 보자

어제		오늘	
회사 A	8만 원	회사 A	12만 원
회사 B	18만 원	회사 B	9만 원

- 전일 대비 종목을 분석하려면  
오늘 날짜의 행과 어제 날짜의 행을 일치시켜  
오늘 종목 가격에서 어제 종목 가격을  
빼는 계산을 수행해야 함

	2021-10-01
2021-10-01	← 2021-10-02
2021-10-02	2021-10-03
2021-10-03	2021-10-04
2021-10-04	2021-10-05
2021-10-05	2021-10-06
2021-10-06	2021-10-07
2021-10-07	2021-10-08
2021-10-08	2021-10-09
2021-10-09	2021-10-10
2021-10-10	

## ■ 전일 대비 종목의 변화 분석하기

- 같은 테이블 내에서 데이터를 비교해야 하므로  
SELF JOIN 또는 LAG, LEAD 함수를 사용해야 함. 여기서는 SELF JOIN을 사용함

**Do it!** SELF JOIN으로 전일 대비 증감과 증감률 조회

```
SELECT
    a.symbol,
    a.date AS a_date,
    CAST(a.close AS DECIMAL(18,2)) AS a_close,
    " AS '---',
    b.date AS b_date,
    CAST(b.close AS DECIMAL(18,2)) AS b_close,
    " AS '---',
    CAST((b.close - a.close) AS DECIMAL(18,2)) AS `diff_price($)` ,
    CAST(((b.close - a.close) / b.close * 100) AS DECIMAL(18,2)) AS `diff_ratio(%)`
FROM stock AS A
    INNER JOIN stock AS B ON a.symbol = b.symbol AND a.date = date_add(b.date,
INTERVAL -1 DAY)
WHERE a.date = '2023-10-04';
```

## ■ 전일 대비 종목의 변화 분석하기

### 실행 결과

	symbol	a_date	a_close	---	b_date	b_close	---	diff_price(\$)	diff_ratio(%)
▶	AAPL	2023-10-04 00:00:00	173.66		2023-10-05 00:00:00	174.91		1.25	0.71
	AMD	2023-10-04 00:00:00	104.07		2023-10-05 00:00:00	102.91		-1.16	-1.13
	AMZN	2023-10-04 00:00:00	127.00		2023-10-05 00:00:00	125.96		-1.04	-0.83
	INTC	2023-10-04 00:00:00	35.93		2023-10-05 00:00:00	35.89		-0.04	-0.11
	META	2023-10-04 00:00:00	305.58		2023-10-05 00:00:00	304.79		-0.79	-0.26
	MSFT	2023-10-04 00:00:00	318.96		2023-10-05 00:00:00	319.36		0.40	0.13
	NVDA	2023-10-04 00:00:00	440.41		2023-10-05 00:00:00	446.88		6.47	1.45
	TSLA	2023-10-04 00:00:00	261.16		2023-10-05 00:00:00	260.05		-1.11	-0.43

- 입력한 날짜와 다음 날짜의 종가를 비교하여  
전일 대비 증감값 및 증감 비율에 대한 결과를 확인할 수 있음

## ■ 주가가 연속 상승한 종목 분석하기

- 일정 기간 동안 연속으로 주가가 오른 종목을 분석해 보자
- 분석하려는 기간의 시작과 끝을 비교해 주가가 N% 이상 오른 종목 중에서 특히 연속으로 오른 종목을 조회할 것임
- 정교하게 분석하는 만큼 쿼리가 꽤 복잡함
- 쉽게 이해할 수 있도록 쿼리를 최대한 분리하고  
임시 테이블에 데이터를 저장하는 방식으로 실습을 진행함
- 이렇게 하면 쿼리 성능은 떨어질 수 있는데,  
지금은 성능보다 학습에 중점을 두어야 함



## ■ 주가가 연속 상승한 종목 분석하기

- 이번 실습은 stock 테이블에서  
2021년 2월 17일부터 2021년 2월 24일까지 일별 주식 데이터 중  
주가가 10% 이상 오른 종목을 찾아 그중에서도  
해당 기간 동안 주가가 한 번도 떨어지지 않은 종목만 조회해 보자
- 이때 임시 테이블을 사용해야 하므로  
세션은 유지하고 쿼리는 블록 단위로 실행해야 함

## ▪ 주가가 연속 상승한 종목 분석하기

1. 먼저 첫 날짜와 마지막 날짜의 종가(close)를 조회해  
증감 주가와 증감 주가율을 저장한 임시 테이블을 만들어 보자.

같은 symbol의 데이터를 비교하고자 FROM 문의 서브 쿼리를 사용해서  
2021년 2월 17일의 데이터 집합을 만들고,

2021년 2월 24일의 데이터 집합을 만든 다음,  
JOIN 조건으로 symbol 열을 비교함

## 주가가 연속 상승한 종목 분석하기

1.

**Do it!** 특정 기간 동안 종목별 등락을 저장하는 테이블 생성

```
CREATE TEMPORARY TABLE temp1
SELECT
    a.symbol,
    a.close AS a_close,
    b.close AS b_close,
    b.close - a.close AS close_diff,
    (b.close - a.close) / a.close * 100 AS ratio_diff
FROM (SELECT symbol, close FROM stock WHERE date = '2021-02-17' ) AS a
     INNER JOIN (SELECT symbol, close FROM stock WHERE date = '2021-02-24') AS b
      ON a.symbol = b.symbol;

SELECT * FROM temp1;
```

실행 결과

	symbol	a_close	b_close	close_diff	ratio_diff
▶	AAPL	130.84	125.35	-5.49	-4.195965
	AMD	89.94	86.94	-3.00	-3.335557
	AMPG	6.61	9.25	2.64	39.939486
	AMZN	165.43	157.98	-7.45	-4.503415
	BHR	6.02	7.28	1.26	20.930233
	CAR	42.82	55.90	13.08	30.546474
	INTC	61.85	63.19	1.34	2.166532
	META	273.57	264.31	-9.26	-3.384874
	MSFT	244.20	234.55	-9.65	-3.951679
	NVDA	149.06	144.99	-4.07	-2.730444
	TSLA	266.05	247.34	-18.71	-7.032513
	UAN	21.51	26.80	5.29	24.593212

temp1 테이블에서는 symbol 열의 데이터에 따라  
 각 2021-02-17의 가격과 2021-02-24의 가격  
 그리고 이 둘의 가격 차이, 증감률을 계산해 저장된 것을 확인할 수 있음

## ■ 주가가 연속 상승한 종목 분석하기

2. 앞서 생성한 데이터 집합 중에 주가 증가율이 10% 이상, 즉 `ratio_diff >= 10`인 일별 데이터 집합을 만들려고 `temp1` 테이블과 `stock` 테이블을 조인함.

이때 `ROW_NUMBER` 함수를 사용해 `symbol` 열을 `PARTITION BY`로 나눠 날짜 순서대로 순위를 부여함.

이렇게 순위를 부여한 이유는 전일 날짜를 비교할 때 주식 시장이 열리지 않는 주말이나 공휴일이 기간에 포함될 경우 공백이 생기므로 -1일 방식으로 비교할 수 없기 때문임

## ■ 주가가 연속 상승한 종목 분석하기

2. 여기서 생성한 데이터는 temp2이라는 이름의 임시 테이블에 저장함.

MySQL에서는 임시 테이블을 동시에 두 군데 이상에서 참조할 수 없기 때문에 SELF JOIN을 사용할 수 없음.

그래서 동일한 결과를 임시 테이블 2개에 저장하여 조인하여 SELF JOIN과 같은 효과를 만들어서 사용함

## 주가가 연속 상승한 종목 분석하기

**Do it!** 10% 상승한 종목들의 정보를 저장하는 테이블 생성

2.

```
CREATE TEMPORARY TABLE temp2
SELECT
    ROW_NUMBER() OVER (PARTITION BY a.symbol ORDER BY date ASC) AS
num,
    a.symbol,
    b.date,
    b.close
FROM temp1 AS a
    INNER JOIN stock AS b ON a.symbol = b.symbol
WHERE a.ratio_diff >= 10
    AND b.date >= '2021-02-17'
    AND b.date <= '2021-02-24';
CREATE TEMPORARY TABLE temp2_1
SELECT
    ROW_NUMBER() OVER (PARTITION BY a.symbol ORDER BY date ASC) AS num,
    a.symbol,
    b.date,
    b.close
FROM temp1 AS a
    INNER JOIN stock AS b ON a.symbol = b.symbol
WHERE a.ratio_diff >= 10
    AND b.date >= '2021-02-17'
    AND b.date <= '2021-02-24';
SELECT * FROM temp2;
```

실행 결과

	num	symbol	date	close
▶	1	AMPG	2021-02-17 00:00:00	6.61
	2	AMPG	2021-02-18 00:00:00	7.45
	3	AMPG	2021-02-19 00:00:00	8.36
	4	AMPG	2021-02-22 00:00:00	8.50
	5	AMPG	2021-02-23 00:00:00	8.78
	6	AMPG	2021-02-24 00:00:00	9.25
	1	BHR	2021-02-17 00:00:00	6.02
	2	BHR	2021-02-18 00:00:00	6.08
	3	BHR	2021-02-19 00:00:00	6.45
	4	BHR	2021-02-22 00:00:00	6.82
	5	BHR	2021-02-23 00:00:00	7.01
	6	BHR	2021-02-24 00:00:00	7.28
	1	CAR	2021-02-17 00:00:00	42.82
	2	CAR	2021-02-18 00:00:00	44.05
	3	CAR	2021-02-19 00:00:00	48.75
	4	CAR	2021-02-22 00:00:00	50.50
	5	CAR	2021-02-23 00:00:00	54.50
	6	CAR	2021-02-24 00:00:00	55.90

## ▪ 주가가 연속 상승한 종목 분석하기

2. temp2, temp2\_1 임시 테이블은  
temp1 임시 테이블의 데이터 중에

주가가 10% 이상 오른 종목의  
데이터의 중간 결과가 저장된 것을 확인할 수 있음

## ▪ 주가가 연속 상승한 종목 분석하기

3. 앞에서 저장한 temp2 테이블을 SELF JOIN해서 현재 순위보다 1만큼 높은 데이터를 찾아 비교함.

이전에 실습한 전일 데이터를 비교할 때와 같은 방법이지만 날짜가 아닌 ROW\_NUMBER 함수로 생성한 순위를 비교함.

이렇게 생성한 데이터는 temp3 임시 테이블에 저장함



## ■ 주가가 연속 상승한 종목 분석하기

3. **Do it!** symbol 열을 기준으로 전일 데이터 전일 대비 상승한 종목 데이터를 저장하는 테이블 생성

```
CREATE TEMPORARY TABLE temp3
SELECT
    b.symbol,
    a.date AS a_date,
    a.close AS a_close,
    b.date AS b_date,
    b.close AS b_close,
    b.close - a.close AS close_diff,
    ((b.close - a.close) / a.close) * 100 AS ratio_diff
FROM temp2 AS a
    INNER JOIN temp2_1 AS b ON a.symbol = b.symbol AND a.num = b.num - 1
ORDER BY b.symbol, b.date;
SELECT * FROM temp3;
```

## 주가가 연속 상승한 종목 분석하기

3.

실행 결과

	symbol	a_date	a_dose	b_date	b_dose	dose_diff	ratio_diff
▶	AMPG	2021-02-17 00:00:00	6.61	2021-02-18 00:00:00	7.45	0.84	12.708018
	AMPG	2021-02-18 00:00:00	7.45	2021-02-19 00:00:00	8.36	0.91	12.214765
	AMPG	2021-02-19 00:00:00	8.36	2021-02-22 00:00:00	8.50	0.14	1.674641
	AMPG	2021-02-22 00:00:00	8.50	2021-02-23 00:00:00	8.78	0.28	3.294118
	AMPG	2021-02-23 00:00:00	8.78	2021-02-24 00:00:00	9.25	0.47	5.353075
	BHR	2021-02-17 00:00:00	6.02	2021-02-18 00:00:00	6.08	0.06	0.996678
	BHR	2021-02-18 00:00:00	6.08	2021-02-19 00:00:00	6.45	0.37	6.085526
	BHR	2021-02-19 00:00:00	6.45	2021-02-22 00:00:00	6.82	0.37	5.736434
	BHR	2021-02-22 00:00:00	6.82	2021-02-23 00:00:00	7.01	0.19	2.785924
	BHR	2021-02-23 00:00:00	7.01	2021-02-24 00:00:00	7.28	0.27	3.851641
	CAR	2021-02-17 00:00:00	42.82	2021-02-18 00:00:00	44.05	1.23	2.872489
	CAR	2021-02-18 00:00:00	44.05	2021-02-19 00:00:00	48.75	4.70	10.669694
	CAR	2021-02-19 00:00:00	48.75	2021-02-22 00:00:00	50.50	1.75	3.589744
	CAR	2021-02-22 00:00:00	50.50	2021-02-23 00:00:00	54.50	4.00	7.920792
	CAR	2021-02-23 00:00:00	54.50	2021-02-24 00:00:00	55.90	1.40	2.568807
	UAN	2021-02-17 00:00:00	21.51	2021-02-18 00:00:00	22.49	0.98	4.556020

temp3 테이블에는 전일의 가격과 비교하여  
증감 여부 결과 데이터가 저장된 것을 확인할 수 있음

## ▪ 주가가 연속 상승한 종목 분석하기

4. 앞에서 생성한 temp3 테이블에서 전일과 비교해서 주가가 한 번이라도 하락한 종목이 있으면 해당 종목을 제외한 데이터를 temp4 임시 테이블에 저장함.

WHERE 문에 NOT IN을 사용해서  
symbol 열에서 하락한 종목이 있는지 검사한 다음,  
조건에 포함되지 않는 항목만 조회함

## ■ 주가가 연속 상승한 종목 분석하기

### 4. **Do it!** 주가가 한 번도 하락하지 않은 데이터를 저장하는 테이블 생성

```
CREATE TEMPORARY TABLE temp3_1
SELECT symbol FROM temp3 WHERE ratio_diff < 0 GROUP BY symbol;
CREATE TEMPORARY TABLE temp4
SELECT
    symbol,
    a_date,
    round(a_close, 2) AS a_close,
    b_date,
    round(b_close, 2) AS b_close,
    round(close_diff, 2) AS close_diff,
    round(ratio_diff, 2) AS ratio_diff
FROM temp3
WHERE symbol NOT IN (SELECT symbol FROM temp3_1);

SELECT * FROM temp4;
```

## 주가가 연속 상승한 종목 분석하기

4.

실행 결과							
	symbol	a_date	a_close	b_date	b_close	close_diff	ratio_diff
▶	AMPG	2021-02-17 00:00:00	6.61	2021-02-18 00:00:00	7.45	0.84	12.71
	AMPG	2021-02-18 00:00:00	7.45	2021-02-19 00:00:00	8.36	0.91	12.21
	AMPG	2021-02-19 00:00:00	8.36	2021-02-22 00:00:00	8.50	0.14	1.67
	AMPG	2021-02-22 00:00:00	8.50	2021-02-23 00:00:00	8.78	0.28	3.29
	AMPG	2021-02-23 00:00:00	8.78	2021-02-24 00:00:00	9.25	0.47	5.35
	BHR	2021-02-17 00:00:00	6.02	2021-02-18 00:00:00	6.08	0.06	1.00
	BHR	2021-02-18 00:00:00	6.08	2021-02-19 00:00:00	6.45	0.37	6.09
	BHR	2021-02-19 00:00:00	6.45	2021-02-22 00:00:00	6.82	0.37	5.74
	BHR	2021-02-22 00:00:00	6.82	2021-02-23 00:00:00	7.01	0.19	2.79
	BHR	2021-02-23 00:00:00	7.01	2021-02-24 00:00:00	7.28	0.27	3.85
	CAR	2021-02-17 00:00:00	42.82	2021-02-18 00:00:00	44.05	1.23	2.87
	CAR	2021-02-18 00:00:00	44.05	2021-02-19 00:00:00	48.75	4.70	10.67
	CAR	2021-02-19 00:00:00	48.75	2021-02-22 00:00:00	50.50	1.75	3.59
	CAR	2021-02-22 00:00:00	50.50	2021-02-23 00:00:00	54.50	4.00	7.92
	CAR	2021-02-23 00:00:00	54.50	2021-02-24 00:00:00	55.90	1.40	2.57
	UAN	2021-02-17 00:00:00	21.51	2021-02-18 00:00:00	22.49	0.98	4.56
	UAN	2021-02-18 00:00:00	22.49	2021-02-19 00:00:00	23.00	0.51	2.27

temp4 테이블에서는 한번이라도 하락한 종목이 있는 목록을 temp3\_1이라는 테이블에 저장하고, temp3 테이블의 결괏값이 temp3\_1에 포함되지 않는 종목 데이터만 저장된 것을 확인할 수 있음

## ■ 주가가 연속 상승한 종목 분석하기

5. 마지막으로 지금까지 생성한 모든 임시 테이블과 nasdaq\_company 테이블을 조인해 최종 데이터를 생성함

**Do it!** nasdaq\_company 테이블과 임시 테이블을 조인해 최종 정보 출력

```
SELECT
    a.symbol,
    d.company_name,
    d.industry,
    ROUND(a.a_close, 2) AS a_close,
    ROUND(a.b_close, 2) AS b_close,
    ROUND(a.close_diff, 2) AS diff_price,
    ROUND(a.ratio_diff, 2) AS diff_ratio
FROM temp1 AS a
    INNER JOIN (SELECT symbol FROM temp2 GROUP BY symbol) AS b ON
a.symbol =
b.symbol
    INNER JOIN (SELECT symbol FROM temp4 GROUP BY symbol) AS c ON
a.symbol =
c.symbol
    INNER JOIN nasdaq_company AS d ON a.symbol = d.symbol
ORDER BY ratio_diff DESC;
```

## 주가가 연속 상승한 종목 분석하기

### 5. 실행 결과

	symbol	company_name	industry	a_dose	b_dose	diff_price	diff_ratio
▶	AMPG	AMPG, Inc.	K2	6.61	9.25	2.64	39.94
	CAR	CAR, Inc.	K2	42.82	55.90	13.08	30.55
	UAN	UAN, Inc.	K2	21.51	26.80	5.29	24.59
	BHR	BHR, Inc.	K2	6.02	7.28	1.26	20.93

초기 설정한 날짜(2021-02-17~2021-02-24) 기간 동안  
연속으로 주식이 상승한 종목들을  
상승값을 기준으로 내림차순한 결과를 확인할 수 있음

## ■ 주가가 연속 상승한 종목 분석하기

### 6. 지금까지 생성한 임시 테이블을 삭제하자.

물론 현재 워크벤치나 세션을 닫아  
임시 테이블을 삭제할 수 있지만

다른 쿼리를 테스트하다가  
실수로 이와 관련된 쿼리를 다시 실행하거나  
같은 이름으로 다시 임시 테이블 생성을 시도하면 오류가 발생함.

#### Do it! 임시 테이블 삭제

```
DROP TEMPORARY TABLE temp1;  
DROP TEMPORARY TABLE temp2;  
DROP TEMPORARY TABLE temp2_1;  
DROP TEMPORARY TABLE temp3;  
DROP TEMPORARY TABLE temp3_1;  
DROP TEMPORARY TABLE temp4;
```