

---

2022 소프트웨어특강1 기말프로젝트

**주가와 재무제표 수집,  
종목 추출 시스템 설계 및 구현**

18013171 김리아

2022.06.13 월요일

---

주가 및 재무제표  
수집 모듈

01. 수정 주가 데이터 수집 및 저장

02. 재무 데이터 수집 및 저장

03. 가치주 종목 추출

종목 추출 모듈

04. 우량주 종목 추출

05. 종목 추출 함수

---

01

## 수정 주가 데이터 수집 및 저장

---

## 01. 수정 주가 데이터 수집 및 저장 | 데이터베이스 테이블

stocks	type	column
code	varchar(10)	종목 코드
name	text	종목 명
exchange	text	시장 구분 (KOSPI/KOSDAQ/KONEX)
mktcap	bigint	시가총액
shares	bigint	상장주식수
date	date	날짜

prices	type	column
Code	varchar(10)	종목 코드
Date	date	날짜
Open	bigint(20)	시가
High	bigint(20)	고가
Low	bigint(20)	저가
Close	bigint(20)	종가
Volume	bigint(20)	거래량
Change	double	

[stocks] : KRX 전종목시세 데이터 중 종목 코드, 종목 이름, 시장 구분, 시가총액, 상장주식수 정보를 크롤링하여 저장할 테이블

[prices] : FinanceDataReader를 이용해 수정주가 정보를 받아와 저장할 테이블

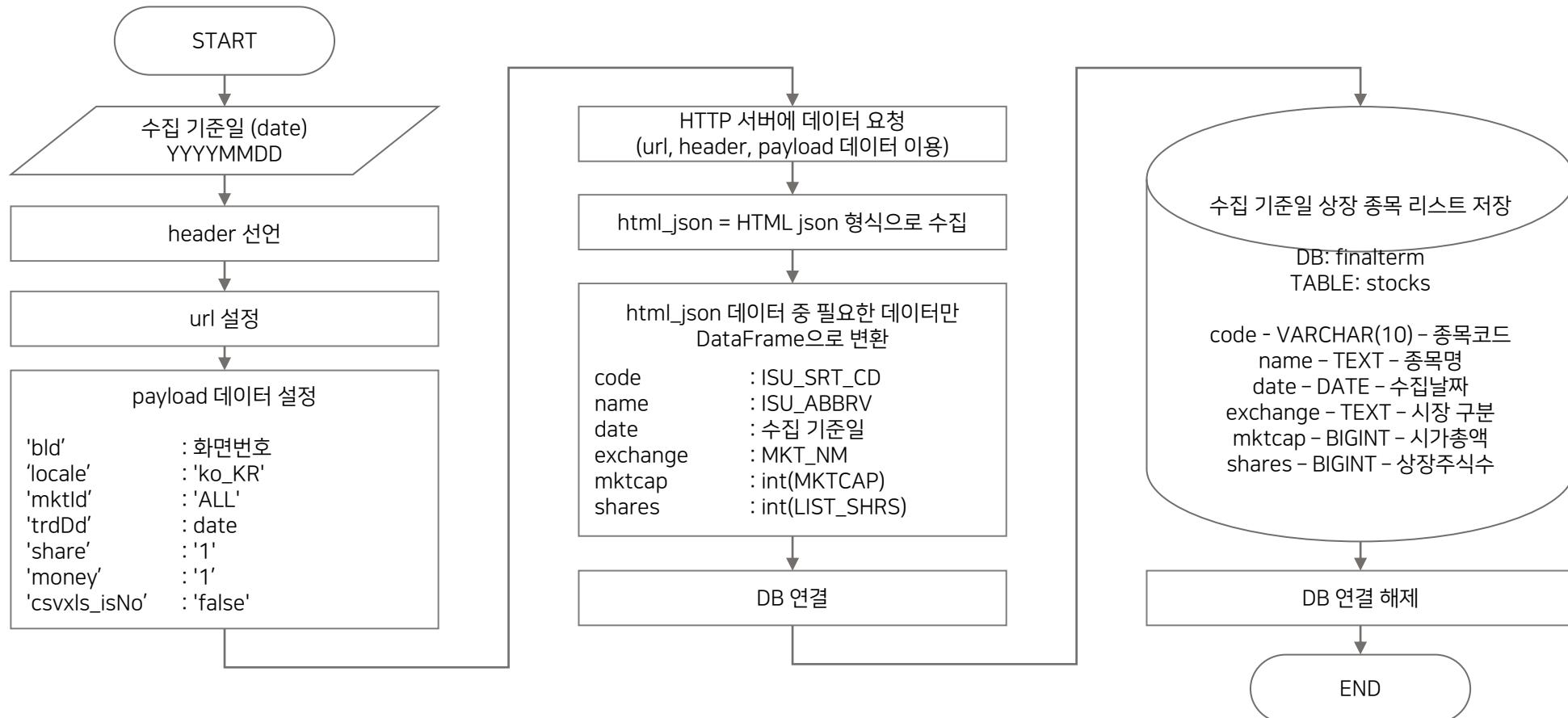
# 01. 수정 주가 데이터 수집 및 저장 1) STOCK 클래스

종목 정보와 관련된 변수, 함수를 담고 있는 클래스

STOCK
- stock_list : DataFrame
+ __init__(): void + setStocks(date): void + getAllStocks(): DataFrame + getStocks(exchange): DataFrame + getStockCode(): DataFrame

변수	type		설명
stock_list	DataFrame		stocks 테이블에서 불러온 종목 정보를 담을 DataFrame 변수
함수	parameter	return	설명
setStocks()	date	void	date 기준, KRX 전종목시세 데이터를 크롤링하여 stocks 테이블에 저장하는 함수
getAllStocks()	void	DataFrame	stocks 테이블에서 모든 종목 정보를 select한 후, DataFrame 형식으로 return하는 함수
getStocks()	exchange	DataFrame	stocks 테이블에서 시장 구분(exchange)에 맞는 데이터만 select한 후, DataFrame 형식으로 return하는 함수
getStockCode()	void	DataFrame	stocks 테이블에서 모든 종목 코드만 select한 후, DataFrame 형식으로 return 하는 함수

# 01. 수정 주가 데이터 수집 및 저장 1) STOCKS 수집 FlowChart



# 01. 수정 주가 데이터 수집 및 저장 1) STOCK 클래스 코드 (set)

```
class STOCK:
    stock_list = None # 종목리스트 저장을 변수

    # 종목리스트 초기화
    def __init__(self):
        self.stock_list = pd.DataFrame()

    # 종목 수집하기
    def setStocks(self, date):
        # header 선언
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36',
            'X-Requested-With': 'XMLHttpRequest'
        }

        # 데이터 url 설정
        url = 'http://data.krx.co.kr/comm/bldAttendant/getJsonData.cmd'

        # payload 데이터 설정 (데이터 속성)
        p_data = {
            'bld': 'dbms/MDC/STAT/standard/MDCSTAT01501', # 화면번호
            'locale': 'ko_KR',
            'mktId': 'ALL', # 전체 조회
            'trdDd': date, # 조회 날짜
            'share': '1',
            'money': '1', # 금액 단위
            'csvxls_isNo': 'false'
        }

        # HTTP 서버에 데이터 요청
        res = requests.post(url, headers=headers, data=p_data)
```

```
# 요청한 데이터 json 형식으로 html_json에 할당
html_json = json.loads(res.content)

# json 형태의 결과값 DataFrame 형식으로 df에 할당
df = pd.json_normalize(html_json['OutBlock_1'])

# df 중 필요한 데이터만 클래스의 stock_list에 할당
self.stock_list[['code', 'name']] = df[['ISU_SRT_CD', 'ISU_ABBRV']] # 종목 코드, 이름
self.stock_list['date'] = datetime.strptime(date[0:4] + "-" + date[4:6] + "-" + date[6:8], '%Y-%m-%d')
self.stock_list[['exchange', 'mktcap', 'shares']] = df[['MKT_NM', 'MKTCAP', 'LIST_SHRS']]

# 시가총액, 상장주식수 int 형변환
for i in range(0, len(self.stock_list)):
    self.stock_list['mktcap'][i] = int(self.stock_list['mktcap'][i].replace(',', ''))
    self.stock_list['shares'][i] = int(self.stock_list['shares'][i].replace(',', ''))

# sqlalchemy의 create_engine를 이용해 DB 연결
server = '127.0.0.1' # local server
user = 'root' # user name
password = '0000' # 개인 password
db = 'finalterm' # DB 이름
engine = create_engine('mysql+pymysql://{}:{}@{}/{}?charset=utf8'.format(user,password,server,db))

# 종목리스트 stocks DB에 저장
self.stock_list.to_sql(name='stocks', con=engine, if_exists='append', index=False,
                      dtype = {
                          'code' : sqlalchemy.types.VARCHAR(10), # 종목코드
                          'name' : sqlalchemy.types.TEXT(), # 종목명
                          'date' : sqlalchemy.types.DATE(), # 수집날짜
                          'exchange' : sqlalchemy.types.TEXT(), # KOSPI / KOSDAQ / KONEX
                          'mktcap' : sqlalchemy.types.BIGINT(), # 시가총액
                          'shares' : sqlalchemy.types.BIGINT() # 상장주식수
                      })
```

# 01. 수정 주가 데이터 수집 및 저장 1) STOCK 클래스 코드 (get)

```
# 모든 종목 불러오기
def getAllStocks(self):
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    sql = "SELECT * FROM stocks"
    cursor.execute(sql)

    self.stock_list = pd.DataFrame(cursor.fetchall())
    self.stock_list.columns = [col[0] for col in cursor.description]

    # DB 연결 해제
    cursor.close()
    conn.close()

    return self.stock_list

# KOSPI / KOSDAQ / KONEX 선택해 불러오기
def getStocks(self, exchange):
    # DB 연결 & 특정 exchange 종목만 가져오기
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    sql = "SELECT * FROM stocks WHERE exchange='{}'".format(exchange.upper())
    cursor.execute(sql)

    # stock_list에 DB 데이터 할당
    self.stock_list = pd.DataFrame(cursor.fetchall())
    self.stock_list.columns = [col[0] for col in cursor.description] # column명 지정

    # DB 연결 해제
    cursor.close()
    conn.close()

    return self.stock_list
```

```
# DB에 저장되어 있는 종목 코드 가져오기
def getStockCode(self):
    # DB 연결 & 특정 exchange 종목만 가져오기
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    sql = "SELECT code FROM stocks"
    cursor.execute(sql)

    # stock_list에 DB 데이터 할당
    self.stock_list = pd.DataFrame(cursor.fetchall())
    self.stock_list.columns = [col[0] for col in cursor.description] # column명 지정

    # DB 연결 해제
    cursor.close()
    conn.close()

    return self.stock_list
```

# 01. 수정 주가 데이터 수집 및 저장 1) STOCK 클래스 실행 예시

## STOCK 클래스 사용 예시

```
# 주식 종목 클래스 생성 & set
s = STOCK()

# 2022년 1분기 기준 종목 수집 > DB 저장
s.setStocks('20220331')

# 주식 종목 클래스 생성 & get
g = STOCK()

# DB에 저장된 주식 종목 불러오기
stocks = g.getAllStocks() # 전체
stocks = g.getStocks('KOSPI') # KOSPI 종목만
stocks = g.getStocks('KOSDAQ') # KOSDAQ 종목만
stocks = g.getStocks('KONEX') # KONEX 종목만
stocks = g.getStockCode() # 종목 코드 가져오기
stocks
```

	code	name	date	exchange	mktcap	shares
0	060310	3S	2022-03-31	KOSDAQ	162413010630	46271513
1	095570	AJ네트웍스	2022-03-31	KOSPI	289830006050	46822295
2	006840	AK홀딩스	2022-03-31	KOSPI	290783963950	13247561
3	054620	APS홀딩스	2022-03-31	KOSDAQ	248809496200	20394221
4	265520	AP시스템	2022-03-31	KOSDAQ	362933748750	15281421
...	...	...	...	...	...	...
2620	000547	흥국화재2우B	2022-03-31	KOSPI	4569600000	153600
2621	000545	흥국화재우	2022-03-31	KOSPI	6689280000	768000
2622	003280	흥아해운	2022-03-31	KOSPI	699636456090	240424899
2623	037440	희림	2022-03-31	KOSDAQ	132263512500	13922475
2624	238490	힘스	2022-03-31	KOSDAQ	103280714680	11312236

2625 rows × 6 columns

getAllStocks() 실행 결과

# 01. 수정 주가 데이터 수집 및 저장 1) STOCK 클래스 실행 예시

code	name		date	exchange	mktcap	shares
0	AJ네트웍스		2022-03-31	KOSPI	289830006050	46822295
1	AK홀딩스		2022-03-31	KOSPI	290783963950	13247561
2	BGF		2022-03-31	KOSPI	509213328120	95716791
3	BGF리테일		2022-03-31	KOSPI	3016041597000	17283906
4	BNK금융지주		2022-03-31	KOSPI	2594444558160	325935246
...	...	...	...	...	...	...
935	069260	휴켐스	2022-03-31	KOSPI	923856088800	40878588
936	000540	흥국화재	2022-03-31	KOSPI	260182712250	64242645
937	000547	흥국화재2우B	2022-03-31	KOSPI	4569600000	153600
938	000545	흥국화재우	2022-03-31	KOSPI	6689280000	768000
939	003280	흥아해운	2022-03-31	KOSPI	699636456090	240424899

940 rows × 6 columns

getStocks('KOSPI') 실행 결과

code	name		date	exchange	mktcap	shares
0	060310	3S	2022-03-31	KOSDAQ	162413010630	46271513
1	054620	APS홀딩스	2022-03-31	KOSDAQ	248809496200	20394221
2	265520	AP시스템	2022-03-31	KOSDAQ	362933748750	15281421
3	211270	AP위성	2022-03-31	KOSDAQ	230759251200	15082304
4	032790	BNGT	2022-03-31	KOSDAQ	120352455875	29534345
...	...	...	...	...	...	...
1551	024060	흥구석유	2022-03-31	KOSDAQ	121500000000	15000000
1552	010240	흥국	2022-03-31	KOSDAQ	87860822480	12322696
1553	189980	흥국에프엔비	2022-03-31	KOSDAQ	155791318045	39997771
1554	037440	희림	2022-03-31	KOSDAQ	132263512500	13922475
1555	238490	힘스	2022-03-31	KOSDAQ	103280714680	11312236

1556 rows × 6 columns

getStocks('KOSDAQ') 실행 결과

code
0
1
2
3
4
...
2620
2621
2622
2623
2624

2625 rows × 1 columns

getStockCode() 실행 결과

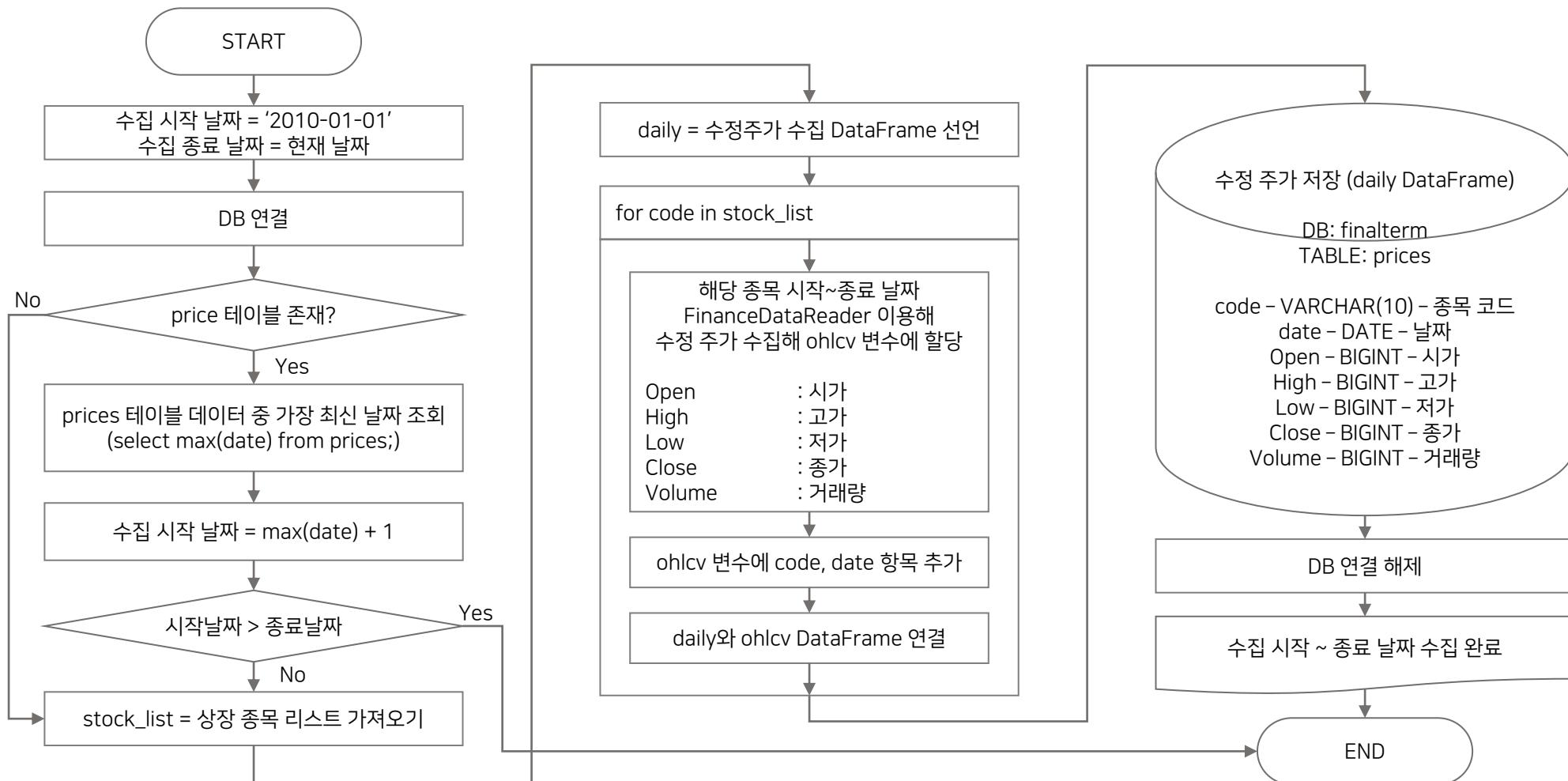
# 01. 수정 주가 데이터 수집 및 저장 2) PRICE 클래스

주가 정보와 관련된 변수, 함수를 담고 있는 클래스

PRICE
- price_list : DataFrame
+ __init__(): void + setPrices(): void + getPrices(): DataFrame + getTermPrices(term): DataFrame + getTermOHLCV(term) : DataFrame + termToDate(term): str, str

변수	type		설명
price_list	DataFrame		prices 테이블에서 불러온 수정주가 정보를 담을 DataFrame 변수
함수	parameter	return	설명
setPrices()	void	void	DB에 저장된 마지막 날짜 다음날부터 오늘까지 수정주가 정보를 FinanceDataReader를 이용해 가져와 prices 테이블에 저장하는 함수
getPrices()	void	DataFrame	prices 테이블에서 모든 수정주가 정보를 select한 후, DataFrame 형식으로 return하는 함수
getTermPrices()	term	DataFrame	prices 테이블에서 해당 분기의 데이터만 select한 후, DataFrame 형식으로 return하는 함수
getTermOHLCV()	term	DataFrame	prices 테이블에서 해당 분기의 데이터만 select한 후, 분기별 OHLCV를 계산해 DataFrame 형식으로 return 하는 함수
termToDate()	term	str, str	YYYYQN 형식의 분기 문자열을 해당 분기의 시작 날짜, 종료 날짜로 변환해 return 하는 함수

# 01. 수정 주가 데이터 수집 및 저장 2) PRICES 수집 FlowChart



# 01. 수정 주가 데이터 수집 및 저장 2) PRICE 클래스 코드 (set)

```
class PRICE:  
    price_list = None # 수정주가 리스트 저장할 변수  
  
    # 수정주가 리스트 초기화  
    def __init__(self):  
        self.price_list = pd.DataFrame()  
  
    # 수정주가 수집하기  
    def setPrices(self):  
        # 수정주가 수집 날짜 설정  
        start_date = datetime.strptime('2010-01-01', '%Y-%m-%d')  
        end_date = datetime.now() # 오늘까지 data 수집  
  
        # DB 연결하기  
        conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')  
        cursor = conn.cursor()  
  
        # prices 테이블 존재 여부 체크하기  
        cursor.execute("SHOW TABLES LIKE 'prices'")  
        check = cursor.fetchall()  
  
        # prices 테이블 존재하는 경우  
        if len(check) != 0:  
            sql = "SELECT max(date) from prices" # 가장 최신 날짜 조회하기  
            df = pd.read_sql(sql, conn)  
  
            # prices 테이블에 데이터 존재하는 경우  
            if df.loc[0][0] is not None:  
                max_date = df.loc[0][0].strftime('%Y-%m-%d')  
                max_date = datetime.strptime(max_date, '%Y-%m-%d')  
  
                # max_date까지는 수정주가 데이터 존재  
                print(max_date.strftime('%Y-%m-%d'), "exist ...")  
                start_date = max_date + timedelta(days=1) # 주가 수집 시작 날짜 변경 (max_date + 1)  
  
            # 이미 오늘 수정주가까지 수집 완료한 경우  
            if start_date > end_date:  
                print("Already Exist")  
                return  
  
    # 이미 오늘 수정주가까지 수집 완료한 경우
```

```
    # 상장 종목 리스트 가져오기  
    stocks = fdr.StockListing('KRX') # 코스피, 코스닥, 코넥스 전체  
    stock_list = fdr.StockListing('KRX').dropna()  
  
    # sqlalchemy의 create_engine을 이용해 DB 연결  
    server = '127.0.0.1' # local server  
    user = 'root' # user name  
    password = '0000' # 개인 password  
    db = 'finalterm' # DB 이름  
    engine = create_engine('mysql+pymysql://{}:{}@{}:{}/?charset=utf8'.format(user,password,server,db))  
  
    # 수정주가 수집해 저장할 DataFrame 변수  
    daily = pd.DataFrame()  
  
    # 모든 종목 코드에 대해 수정주가 수집  
    for code in stock_list['Symbol'].values:  
        ohlc = fdr.DataReader(code, start=start_date, end=end_date) # start~end 수정 주가 수집  
        ohlc['code'] = code # 종목코드 추가  
        daily = pd.concat([daily, ohlc]) # daily와 ohlc 연결  
        daily['date'] = daily.index # 날짜 추가  
  
        print("(+"+code+") ... OK") # 해당 종목 수집 완료  
  
    # 수집한 수정주가 DB에 저장  
    daily.to_sql(name='prices', con=engine, if_exists='append', index=False,  
                 dtype = {  
                     'code' : sqlalchemy.types.VARCHAR(10), # 종목코드  
                     'date' : sqlalchemy.types.DATE(), # 날짜  
                     'Open' : sqlalchemy.types.BIGINT(), # 시가  
                     'High' : sqlalchemy.types.BIGINT(), # 고가  
                     'Low' : sqlalchemy.types.BIGINT(), # 저가  
                     'Close' : sqlalchemy.types.BIGINT(), # 종가  
                     'Volume' : sqlalchemy.types.BIGINT(), # 거래량  
                 })  
  
    # DB 접속 해제  
    cursor.close()  
    conn.close()  
    engine.dispose()  
  
    print(start_date, " ~ ", end_date, " 수집 완료")
```

# 01. 수정 주가 데이터 수집 및 저장 2) PRICE 클래스 코드 (get)

```
# 수정주가 불러오기 (전체 날짜)
def getPrices(self):
    # DB 연결 & 전체 수정주가 불러오기
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    sql = "SELECT * FROM prices"
    cursor.execute(sql)

    # DB 불러온 데이터 price_list DataFrame에 할당
    self.price_list = pd.DataFrame(cursor.fetchall())
    self.price_list.columns = [col[0] for col in cursor.description]

    # DB 연결 해제
    cursor.close()
    conn.close()

    return self.price_list

# 해당 분기 수정주가 불러오기
def getTermPrices(self, term):
    # term > 분기별 시작/종료날짜 설정
    start_date, end_date = self.termToDate(term)

    # DB 연결 & 해당 분기 수정주가 불러오기
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    sql = "SELECT * FROM prices WHERE DATE(date) BETWEEN '{}' AND '{}'".format(start_date, end_date)
    cursor.execute(sql)

    # DB 불러온 데이터 price_list DataFrame에 할당
    self.price_list = pd.DataFrame(cursor.fetchall())
    self.price_list.columns = [col[0] for col in cursor.description]

    # DB 연결 해제
    cursor.close()
    conn.close()

    return self.price_list
```

```
# 분기별 OHLCV 구하기 (분기별 시가, 종가 등)
def getTermOHLCV(self, term):
    # term으로 일련의 값 period로 변환하기
    period = termToPeriod(term)

    # 해당 분기 주가 데이터 불러오기
    df_is = self.getTermPrices(term)
    df_is = df_is.sort_values(by=['Code', 'Date'], axis=0) # code, date 기준 정렬

    # 결측치 처리
    df_is[['Open', 'High', 'Low', 'Close']] = df_is[['Open', 'High', 'Low', 'Close']].replace(0, np.nan)
    df_is['Open'] = np.where(pd.notnull(df_is['Open']) == True, df_is['Open'], df_is['Close'])
    df_is['High'] = np.where(pd.notnull(df_is['High']) == True, df_is['High'], df_is['Close'])
    df_is['Low'] = np.where(pd.notnull(df_is['Low']) == True, df_is['Low'], df_is['Close'])
    df_is['Close'] = np.where(pd.notnull(df_is['Close']) == True, df_is['Close'], df_is['Close'])

    # 컬럼명 맞춰주기
    df_is.rename(columns={'Code': 'code', 'Date': 'date'}, inplace=True)

    # stock_code 별로 통계 모으기
    groups = df_is.groupby('code')

    # 분기별 OHLCV 구하기
    df_ohlcv = pd.DataFrame()

    df_ohlcv['High'] = groups.max()['High'] # 분기별 고가
    df_ohlcv['Low'] = groups.min()['Low'] # 분기별 저가
    df_ohlcv['Period'] = period # 분기 이름 설정
    df_ohlcv['Open'], df_ohlcv['Close'], df_ohlcv['Volume'] = np.nan, np.nan, np.nan # 분기별 시가, 종가, 거래량 (아래에서 구할 것)

    df_ohlcv['code'] = df_ohlcv.index # 종목 코드
    df_ohlcv = df_ohlcv.reset_index(drop=True) # index 재설정

    for i in range(len(df_ohlcv)):
        df_ohlcv['Open'][i] = float(df_is[df_is['code']==df_ohlcv['code']][i].head(1)['Open']) # 분기별 시가
        df_ohlcv['Close'][i] = float(df_is[df_is['code']==df_ohlcv['code']][i].tail(1)['Close']) # 분기별 저가
        df_ohlcv['Volume'][i] = float(df_is[df_is['code']==df_ohlcv['code']][i].tail(1)['Volume']) # 분기별 거래량

    # 분기별 시가총액, 상장주식수 구하기
    df_stock = STOCK().getAllStocks()

    # OHLCV와 시가총액, 상장주식수 결합하기
    df_ohlcv = pd.merge(df_ohlcv, df_stock, how='left', on=['code']) # 종목 코드 기준 결합

    return df_ohlcv
```

## 01. 수정 주가 데이터 수집 및 저장 2) PRICE 클래스 코드

```
# 분기별 날짜 계산 함수
def termToDate(self, term):
    if term[5] == '1': # 1분기 (작년 4월~3월)
        start_date = str(int(term[0:4])-1) + '-04-01'
        end_date = term[0:4] + '-03-31'

    elif term[5] == '2': # 2분기 (작년 7월~6월)
        start_date = str(int(term[0:4])-1) + '-07-01'
        end_date = term[0:4] + '-06-30'

    elif term[5] == '3': # 3분기 (작년 10월~9월)
        start_date = str(int(term[0:4])-1) + '-09-01'
        end_date = term[0:4] + '-09-30'

    elif term[5] == '4': # 4분기 (1월~12월)
        start_date = term[0:4] + '-01-01'
        end_date = term[0:4] + '-12-31'

    return start_date, end_date
```

# 01. 수정 주가 데이터 수집 및 저장 2) PRICE 클래스 실행 예시

## PRICE 클래스 사용 예시

```
# 수정주가 클래스 생성 & set  
s = PRICE()  
s.setPrices()
```

```
2022-06-10 exist ...  
(060310) ... OK  
(095570) ... OK  
(006840) ... OK  
(054620) ... OK  
(265520) ... OK  
(211270) ... OK  
(027410) ... OK  
(282330) ... OK  
(032790) ... OK  
(138930) ... OK  
(001460) ... OK  
(013720) ... OK  
(001040) ... OK  
(079160) ... OK  
(035760) ... OK  
(311690) ... OK  
(000120) ... OK  
(011150) ... OK  
(007350) ... OK
```

```
# 수정주가 클래스 생성 & set  
s = PRICE()  
s.setPrices()
```

```
(000000) ... OK  
(028080) ... OK  
(032860) ... OK  
(200670) ... OK  
(212310) ... OK  
(079980) ... OK  
(065510) ... OK  
(215090) ... OK  
(005010) ... OK  
(263920) ... OK  
(243070) ... OK  
(084110) ... OK  
(145020) ... OK  
(010240) ... OK  
(189980) ... OK  
(000540) ... OK  
(003280) ... OK  
(037440) ... OK  
(238490) ... OK  
2022-06-11 00:00:00 ~ 2022-06-12 17:46:05.259452 수집 완료
```

setPrices() 실행 결과

# 01. 수정 주가 데이터 수집 및 저장 2) PRICE 클래스 실행 예시

```
# 수정주가 클래스 생성 & get
g = PRICE()
list = g.getPrices() # 전체 수정주가 불러오기
list = g.getTermPrices('2022Q1') # 2022 1분기 수정주가만 불러오기
list = g.getTermOHLCV('2022Q1') # 2022 1분기 OHLCV 불러오기 (분기별 시가, 종가, 고가, 저가, 거래량 + 시가총액, 상장주식수)
list
```

	Open	High	Low	Close	Volume	Change	Code	Date
0	2430	2520	2395	2515	287995	0.050104	060310	2021-04-01
1	2530	2535	2490	2525	207049	0.003976	060310	2021-04-02
2	2535	2550	2505	2535	217511	0.003960	060310	2021-04-05
3	2535	2690	2485	2550	552344	0.005917	060310	2021-04-06
4	2500	2580	2500	2575	302490	0.009804	060310	2021-04-07
...	...	...	...	...	...	...	...	...
564352	9260	9260	9130	9180	15980	-0.008639	238490	2022-03-25
564353	9180	9180	9060	9140	14712	-0.004357	238490	2022-03-28
564354	9150	9150	9050	9100	11512	-0.004376	238490	2022-03-29
564355	9100	9190	9070	9100	11770	0.000000	238490	2022-03-30
564356	9180	9180	9030	9130	12115	0.003297	238490	2022-03-31

564357 rows × 8 columns

getTermPrices('2022Q1') 실행 결과

	High	Low	Period	Open	Close	Volume	code	name	date	exchange	mktcap	shares
0	21550.0	11250.0	2022/03	14200.0	13650.0	214548.0	000020	동화약품	2022-03-31	KOSPI	381264565500	27931470
1	1470.0	720.0	2022/03	1340.0	889.0	417614.0	000040	KR모터스	2022-03-31	KOSPI	85466120152	96137368
2	16000.0	12650.0	2022/03	13200.0	15950.0	42169.0	000050	경방	2022-03-31	KOSPI	437273556500	27415270
3	53500.0	17150.0	2022/03	19350.0	47450.0	298202.0	000060	메리츠화재	2022-03-31	KOSPI	5723656250000	120625000
4	147500.0	79900.0	2022/03	93800.0	86100.0	13667.0	000070	삼양홀딩스	2022-03-31	KOSPI	737383733100	8564271
...	...	...	...	...	...	...	...	...	...	...	...	
2326	6870.0	2700.0	2022/03	5150.0	4585.0	63761.0	950170	JTC	2022-03-31	KOSDAQ	160500295445	35005517
2327	26500.0	16950.0	2022/03	24300.0	20300.0	45327.0	950190	미투젠	2022-03-31	KOSDAQ	275671807600	13579892
2328	17300.0	6520.0	2022/03	12550.0	7290.0	23924.0	950200	소마젠	2022-03-31	KOSDAQ	139483601370	19133553
2329	40850.0	12850.0	2022/03	28100.0	20300.0	511003.0	950210	프레스티지바이오파마	2022-03-31	KOSPI	1219951946500	60096155
2330	14450.0	6190.0	2022/03	11150.0	7270.0	299156.0	950220	네오이뮨텍	2022-03-31	KOSDAQ	717850777700	98741510

2331 rows × 12 columns

getTermOHLCV('2022Q1') 실행 결과

---

02

## 재무 데이터 수집 및 저장

## 02. 재무 데이터 수집 및 저장 1) 손익계산서 테이블

incomes	type	column
code	varchar(10)	종목 코드
period	text	기간
rpt_type	float	손익계산서 종류
rev	float	매출액
cgs	float	매출원가
gross	float	매출총이익
sga	float	판매비와관리비
opr	float	영업이익
fininc	float	금융이익

incomes	type	column
fincost	float	금융원가
otherrev	float	기타수익
othercost	float	기타비용
otherpl	float	종속기업, 공동지배기업및관계기업관련손익
ebit	float	세전계속사업이익
tax	float	법인세비용
contop	float	계속영업이익
discontop	float	중단영업이익
netinc	float	당기순이익

[incomes] : fnguide.com에서 손익계산서를 크롤링하여 저장할 테이블

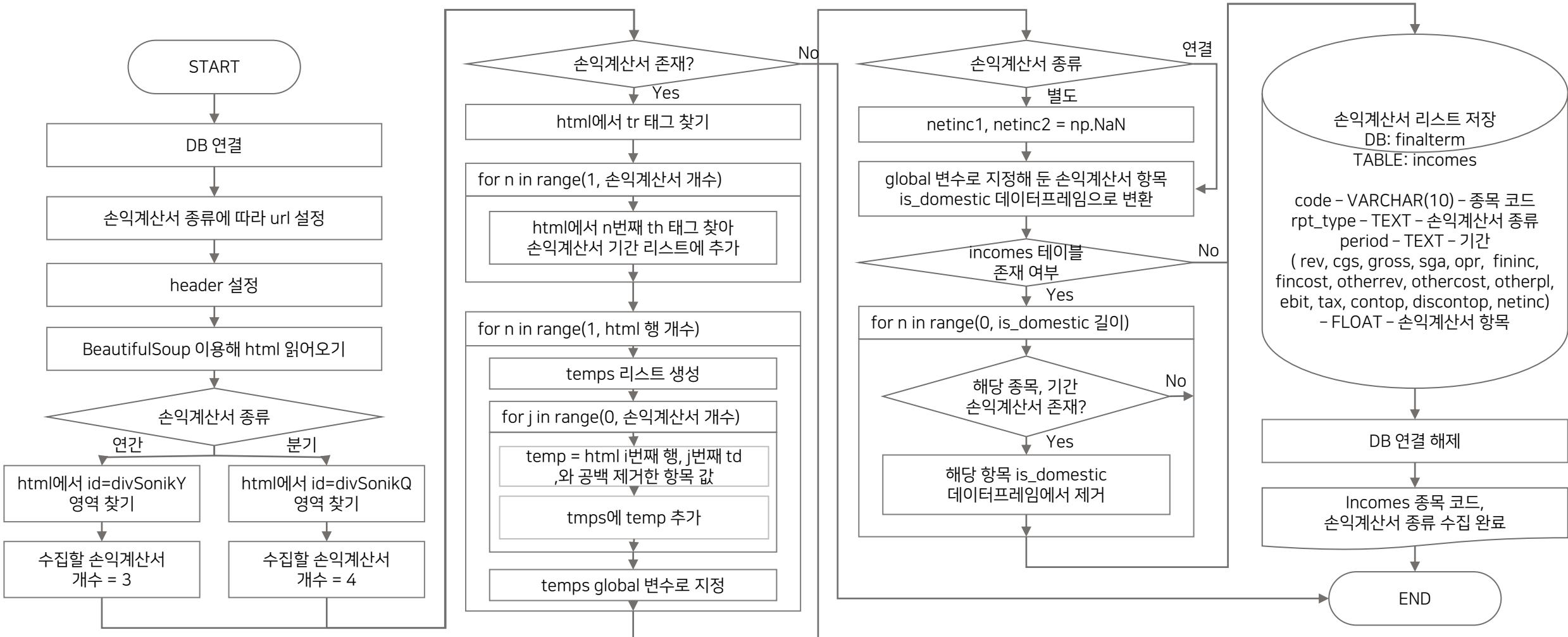
## 02. 재무 데이터 수집 및 저장 1) 손익계산서 클래스

손익계산서와 관련된 변수, 함수를 담고 있는 클래스

INCOMES
- is_list : DataFrame
+ __init__(): void + setIncomes(code, rpt_type, freq): void + getIncomes(): DataFrame + getTermIncomes(period): DataFrame + getStockIncomes(code, period): DataFrame + getTrailingIncomes(code, period): DataFrame

변수	type		설명
is_list	DataFrame		incomes 테이블에서 불러온 손익계산서 정보를 담을 DataFrame
함수	parameter	return	설명
setIncomes()	code, rpt_type, freq	void	fnguide에서 해당 종목, 종류의 손익계산서를 크롤링하고, 데이터베이스에 저장하는 함수
getIncomes()	void	DataFrame	incomes 테이블에서 모든 손익계산서 데이터를 select한 후, DataFrame 형식으로 return하는 함수
getTermIncomes()	period	DataFrame	incomes 테이블에서 해당 분기의 손익계산서만 select한 후, DataFrame 형식으로 return하는 함수
getStockIncomes()	code, period	DataFrame	incomes 테이블에서 해당 종목의 해당 분기 손익계산서만 select한 후, DataFrame 형식으로 return하는 함수
getTrailingIncomes()	code, period	DataFrame	incomes 테이블에서 해당 종목의 해당 분기 포함 1년 간의 손익계산서를 select한 후, 트레일링 데이터로 만들어 DataFrame 형식으로 return하는 함수

## 02. 재무 데이터 수집 및 저장 1) 손익계산서 수집 FlowChart



## 02. 재무 데이터 수집 및 저장 1) 손익계산서 클래스 코드 (set)

```
class INCOMES:
    is_list = None # 손익계산서 리스트 저장을 변수
    # 손익계산서 리스트 초기화
    def __init__(self):
        self.is_list = pd.DataFrame()

    # 손익계산서 수집하기
    def setIncomes(self, code, rpt_type, freq):
        # sqlalchemy의 create_engine를 이용해 DB 연결
        server = '127.0.0.1' # local server
        user = 'root' # user name
        password = '0000' # 개인 password
        db = 'finalterm' # DB 이름
        engine = create_engine('mysql+pymysql://{}:{}@{}/{}?charset=utf8'.format(user,password,server,db))

        items_en = ['rev', 'cgs', 'gross', 'sga', 'sga1', 'sga2', 'sga3', 'sga4', 'sga5', 'sga6', 'sga7', 'sga8', # 12
                   'opr', 'opr1', 'fininc', 'fininc1', 'fininc2', 'fininc3', 'fininc4', 'fininc5', 'fininc6', # 9
                   'fininc7', 'fininc8', 'fininc9', 'fininc10', 'fininc11', 'fincost', 'fincost1', 'fincost2', # 8
                   'fincost3', 'fincost4', 'fincost5', 'fincost6', 'fincost7', 'fincost8', 'fincost9', 'fincost10', # 8
                   'otherrev', 'otherrev1', 'otherrev2', 'otherrev3', 'otherrev4', 'otherrev5', 'otherrev6', # 7
                   'otherrev7', 'otherrev8', 'otherrev9', 'otherrev10', 'otherrev11', 'otherrev12', 'otherrev13', # 7
                   'otherrev14', 'otherrev15', 'otherrev16', 'othercost', 'othercost1', 'othercost2', 'othercost3', # 7
                   'othercost4', 'othercost5', 'othercost6', 'othercost7', 'othercost8', 'othercost9', 'othercost10', # 7
                   'othercost11', 'othercost12', 'otherpl', 'otherpl1', 'otherpl2', 'otherpl3', 'otherpl4', # 7
                   'ebit', 'tax', 'contop', 'discontop', 'netinc'] # 5

        # 손익계산서 종류
        if rpt_type.upper() == 'CONSOLIDATED': # 연결 손익계산서 (ReportGB=0)
            url = 'http://comp.fnguide.com/SV02/ASP/SVD_Finance.asp?pGB=1&gicode=A{}&cID=&MenuYn=Y&ReportGB=D&NewMenuID=103&stckB=701'.format(code)
            items_en = items_en + ['netinc1', 'netinc2']
        else: # 별도 손익계산서 (ReportGB=0)
            url = 'http://comp.fnguide.com/SV02/ASP/SVD_Finance.asp?pGB=1&gicode=A{}&cID=&MenuYn=Y&ReportGB=B&NewMenuID=103&stckB=701'.format(code)

        # Header 설정
        headers = {
            "User-Agent" : "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36"
        }
        req = Request(url=url, headers=headers)
        html = urlopen(req).read()
        soup = BeautifulSoup(html, 'html.parser')
```

```
# 손익계산서 종류
if freq.upper() == 'A': # 연간 손익계산서 영역 추출
    is_a = soup.find(id = 'divSonikY')
    num_col = 3 # 최근 3개년간 데이터
else: # 분기 손익계산서 영역 추출 (freq.upper() == 'Q')
    is_a = soup.find(id = 'divSonikQ')
    num_col = 4 # 최근 4개 분기 데이터

# 손익계산서 손색하시 않는 종목
if is_a is None:
    return

is_a = is_a.findall(['tr'])

# 손익계산서 자료 수집
period = [is_a[0].find_all('th')[n].get_text() for n in range(1, num_col+1)]

# 항목별 값 불러오기
for item, i in zip(items_en, range(1, len(is_a))):
    temps = []
    for j in range(0, num_col):
        temp = [float(is_a[i].find_all('td')[j]['title'].replace(',', '').replace('#xa0', ''))#
                if is_a[i].find_all('td')[j]['title'].replace(',', '').replace('#xa0', '') != ''#
                else (0 if is_a[i].find_all('td')[j]['title'].replace(',', '').replace('#xa0', '') == '-0' #
                      else 0)]
        temps.append(temp[0])
    temps.append(period[i])
    globals()[item] = temps # item_en 내 각 항목을 global 변수로 지정하고 값 저장

# 지배/비지배 항목 처리
if rpt_type.upper() == 'CONSOLIDATED': # 연결 연간 손익계산서는 아무 것도 하지 않음
    pass
else: # 별도 연간 손익계산서 해당 항목을 Null값으로 채움
    globals()['netinc1'], globals()['netinc2'] = [np.NaN]*num_col, [np.NaN]*num_col
```

## 02. 재무 데이터 수집 및 저장 1) 손익계산서 클래스 코드 (set)

```
# 데이터프레임으로 변환
is Domestic = pd.DataFrame({
    'code': code,
    'period': period,
    'rev': rev, # 매출액
    'cgs': cgs, # 매출원가
    'gross': gross, # 매출총이익
    'sga': sga, # 판매비와관리비
    'opr': opr, # 영업이익
    'fininc': fininc, # 금융이익
    'fincost': fincost, # 금융원가
    'otherrev': otherrev, # 기타수익
    'othercost': othercost, # 기타비용
    'otherpl': otherpl, # 종속기업, 공동지배기업및관계기업관련손익
    'ebit': ebit, # 세전계속사업이익
    'tax': tax, # 법인세비용
    'contop': contop, # 계속영업이익
    'discontop': discontop, # 종단영업이익
    'netinc': netinc, # 당기순이익
})
is Domestic['rpt_type'] = rpt_type+'_'+freq.upper() # 손익계산서 종류

# DB 중복 데이터 있는지 체크 (종목코드, 기간, 손익계산서 종류)
conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
cursor = conn.cursor()

# incomes 테이블 존재 여부 체크 (없는 경우 바로 DB에 저장)
cursor.execute("SHOW TABLES LIKE 'incomes'")
result = cursor.fetchall()
result
```

```
# incomes 테이블 존재하는 경우
if len(result)!=0:
    r = rpt_type+'_'+freq.upper() # 손익계산서 종류

    # 존재하는 data인지 체크
    for i in range(0, len(is Domestic)):
        sql = "SELECT * FROM incomes WHERE code='{}' and period='{}' and rpt_type='{}'.format(code, is Domestic.loc[i]['period'], r)
        df = pd.read_sql(sql, conn)

        if not df.empty: # db에 존재하는 경우
            print(code, is Domestic.loc[i]['period'], r, "exist ... ")
            is Domestic = is Domestic.drop([i], axis=0) # 삭제

# DB에 저장
is Domestic.to_sql(name='incomes', con=engine, if_exists='append', index=False,
dtype = {
    'code': sqlalchemy.types.VARCHAR(10),
    'rpt_type': sqlalchemy.types.TEXT(),
    'period': sqlalchemy.types.TEXT(),
    'rev': sqlalchemy.types.FLOAT(),
    'cgs': sqlalchemy.types.FLOAT(),
    'gross': sqlalchemy.types.FLOAT(),
    'sga': sqlalchemy.types.FLOAT(),
    'opr': sqlalchemy.types.FLOAT(),
    'fininc': sqlalchemy.types.FLOAT(),
    'fincost': sqlalchemy.types.FLOAT(),
    'otherrev': sqlalchemy.types.FLOAT(),
    'othercost': sqlalchemy.types.FLOAT(),
    'otherpl': sqlalchemy.types.FLOAT(),
    'ebit': sqlalchemy.types.FLOAT(),
    'tax': sqlalchemy.types.FLOAT(),
    'contop': sqlalchemy.types.FLOAT(),
    'discontop': sqlalchemy.types.FLOAT(),
    'netinc': sqlalchemy.types.FLOAT()
})

# DB 연결 해제
cursor.close()
conn.close()
engine.dispose()

print("(Incomes "+code+" "+rpt_type+"_"+freq.upper()+" ... OK")
```

## 02. 재무 데이터 수집 및 저장 1) 손익계산서 클래스 코드 (get)

```
# 손익계산서 불러오기
def getIncomes(self):
    # db에 연결
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    # 손익계산서 항목 불러와 is_list에 할당
    sql = "SELECT * FROM incomes"
    cursor.execute(sql)
    self.is_list = pd.DataFrame(cursor.fetchall())

    # DB에 손익계산서 존재하지 않는 경우
    if len(self.is_list) == 0:
        print("Empty Incomes")
        return

    # 손익계산서 존재하는 경우 column명 수정
    self.is_list.columns = [col[0] for col in cursor.description]

    # db 연결 해제
    cursor.close()
    conn.close()

    return self.is_list

# 해당 분기의 손익계산서 불러오기
def getTermIncomes(self, period):
    # db에 연결
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    # 분기 동일한 손익계산서 항목 불러와 is_list에 할당
    sql = "SELECT * FROM incomes WHERE period='{}' AND rpt_type='Consolidated_Q'".format(period)
    cursor.execute(sql)
    self.is_list = pd.DataFrame(cursor.fetchall())

    # DB에 해당 분기 손익계산서 존재하지 않는 경우
    if len(self.is_list) == 0:
        print(period, ": Empty Incomes")
        return

    # 손익계산서 존재하는 경우 column명 수정
    self.is_list.columns = [col[0] for col in cursor.description]

    # db 연결 해제
    cursor.close()
    conn.close()

    return self.is_list
```

```
# 해당 종목, 해당 분기의 손익계산서 불러오기
def getStockIncomes(self, code, period):
    # db에 연결
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    # 종목, 분기 동일한 손익계산서 항목 불러와 is_list에 할당
    sql = "SELECT * FROM incomes WHERE code='{}' AND period='{}' AND rpt_type='Consolidated_Q'".format(code, period)
    cursor.execute(sql)
    self.is_list = pd.DataFrame(cursor.fetchall())

    # DB에 해당 종목 손익계산서 존재하지 않는 경우
    if len(self.is_list) == 0:
        print(code, ": Empty Incomes")
        return

    # 손익계산서 존재하는 경우 column명 수정
    self.is_list.columns = [col[0] for col in cursor.description]

    # db 연결 해제
    cursor.close()
    conn.close()

    return self.is_list

# 트레일링 데이터 만들기
def getTrailingIncomes(self, code, period):
    df_quat = []
    period_quat = period # 기간 보정 변수

    # 4분기 데이터 불러오기
    for i in range(4):
        df_is = self.getStockIncomes(code, period_quat) # 포함손익계산서

        # 분기 데이터 존재하지 않는 경우 트레일링 데이터 만들지 않음
        if df_is is None:
            return

        df_quat[i] = df_is

        if int(period_quat[5:7])-3 > 0: # period 마지막 2개 : 06, 09, 12 (2~4분기)
            period_quat = period_quat[0:4] + '/' + str(int(period_quat[5:7])-3).zfill(2) # 이전 분기로 만들기
        else: # 03 (1분기)
            period_quat = str(int(period_quat[0:4])-1) + '/12' # 이전 분기로 만들기 (작년 12)

    df_trailing = pd.DataFrame(df_quat[0], columns=['code', 'period', 'rpt_type'])
    df_trailing[df_quat[0].columns[2:-1]] = 0

    # 모든 데이터 더해 트레일링 데이터로 만들기
    for i in range(len(df_quat)):
        df_trailing[df_quat[0].columns[2:-1]] += df_quat[i][df_quat[i].columns[2:-1]]

    return df_trailing
```

## 02. 재무 데이터 수집 및 저장 1) 손익계산서 클래스 실행 예시

getIncomes() 실행 결과

```
g = INCOMES() # 손익계산서 클래스 생성
list = g.getIncomes() # 전체 손익계산서 가져오기
list
```

	code	period	rev	cgs	gross	sga	opr	fininc	fincost	otherrev	othercost	otherpl	ebit	tax	contop	discontop	netinc
0	060310	2021/03	37.43	40.70	-3.28	4.14	-7.42	0.04	1.14	29.65	0.44	-3.53	17.16	0.33	16.82	0.00	16.82
1	060310	2021/06	37.76	32.66	5.10	7.91	-2.82	0.03	0.49	0.36	0.11	0.00	-3.03	0.60	-3.63	0.00	-3.63
2	060310	2021/09	76.44	55.76	20.69	7.31	13.38	0.02	0.59	2.05	0.02	0.00	14.84	0.02	14.83	0.00	14.83
3	060310	2021/12	66.42	48.80	17.62	7.97	9.66	0.04	0.62	1.20	0.06	0.00	10.22	0.00	10.22	0.00	10.22
4	095570	2021/06	2613.87	0.00	2613.87	2495.24	118.63	25.04	101.68	50.76	7.79	48.68	133.64	35.05	98.59	46.48	145.07
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
34981	037440	2022/03	477.22	423.23	53.99	32.86	21.13	5.05	5.12	1.18	4.59	0.00	17.64	5.43	12.21	0.00	12.21
34982	238490	2021/06	144.19	118.00	26.19	33.78	-7.58	10.40	10.06	0.18	0.08	0.37	-6.77	-2.89	-3.88	0.00	-3.88
34983	238490	2021/09	206.73	115.86	90.87	39.57	51.30	14.34	2.39	0.17	0.03	-0.06	63.33	10.93	52.40	0.00	52.40
34984	238490	2021/12	49.28	56.44	-7.15	60.91	-68.07	-3.03	-2.89	0.76	-1.25	-2.13	-68.33	-16.87	-51.46	0.00	-51.46
34985	238490	2022/03	90.17	86.60	3.57	25.53	-21.96	6.94	0.78	0.12	0.11	-0.04	-15.83	0.22	-16.05	0.00	-16.05

34986 rows × 18 columns

getTermIncomes('2022/03') 실행 결과

```
g = INCOMES() # 손익계산서 클래스 생성
list = g.getTermIncomes('2022/03') # 해당 분기의 모든 종목 손익계산서 가져오기
list
```

	code	period	rev	cgs	gross	sga	opr	fininc	fincost	otherrev	othercost	otherpl	ebit	tax	contop	discontop	netinc
0	095570	2022/03	2958.04	0.00	2958.04	2765.61	192.43	9.70	82.28	11.72	6.56	8.05	133.05	24.50	108.55	0.0	108.55
1	006840	2022/03	8198.40	6969.53	1228.87	1692.78	-463.92	69.98	233.90	82.96	151.62	106.88	-589.62	-109.23	-480.39	0.0	-480.39
2	054620	2022/03	60.28	31.87	28.40	101.11	-72.71	3.10	10.93	19.91	7.86	42.77	-25.70	-0.82	-24.88	0.0	-24.88
3	265520	2022/03	801.91	546.98	254.93	139.89	115.04	20.26	10.43	62.21	15.31	0.00	171.76	42.38	129.39	0.0	129.39
4	211270	2022/03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2325	189980	2022/03	192.95	127.09	65.86	52.41	13.45	0.85	1.92	1.02	0.85	-0.19	12.36	2.92	9.44	0.0	9.44
2326	000540	2022/03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0	0.00
2327	003280	2022/03	262.10	228.57	33.53	14.18	19.35	1.84	15.88	44.64	22.75	-30.91	-3.71	-17.78	14.07	0.0	14.07
2328	037440	2022/03	477.10	423.34	53.76	32.89	20.88	5.05	5.12	1.23	4.59	0.00	17.45	5.43	12.02	0.0	12.02
2329	238490	2022/03	90.17	86.60	3.58	25.56	-21.99	6.95	0.79	0.12	0.11	-0.02	-15.83	0.22	-16.05	0.0	-16.05

2330 rows × 18 columns

getStockIncomes('005930', '2022/03') 실행 결과

```
g = INCOMES() # 손익계산서 클래스 생성
list = g.getStockIncomes('005930', '2022/03') # 해당 종목코드, 분기의 손익계산서 가져오기
list
```

	code	period	rev	cgs	gross	sga	opr	fininc	fincost	otherrev	othercost	otherpl	ebit	tax	contop	discontop
0	005930	2022/03	777815.0	470721.0	307094.0	165880.0	141214.0	350219.9	303332.2	7001.93	4531.1	2324.77	150698.0	37452.1	113246.0	0.0

getTrailingIncomes('005930', '2022/03') 실행 결과

```
g = INCOMES() # 손익계산서 클래스 생성
list = g.getTrailingIncomes('005930', '2022/03') # 해당 종목코드, 분기의 트레일링 데이터 가져오기
list
```

	code	period	rpt_type	rev	cgs	gross	sga	opr	fininc	fincost	otherrev	othercost	otherpl	ebit	tax
0	005930	2022/03	Consolidated_Q	2919978.0	1719835.0	1200144.0	636420.0	563723.0	98067.4	87440.0	26061.46	21848.41	8145.72	586710.0	145806.3

## 02. 재무 데이터 수집 및 저장 2) 재무상태표 테이블

balances	type	column
code	varchar(10)	종목 코드
period	text	기간
rpt_type	float	재무상태표 종류
asset	float	자산
curasset	float	유동자산
inv	float	재고자산
rec	float	매출채권
cash	float	현금및현금성자산
longasset	float	비유동자산
otherassets	float	기타금융업자산
liab	float	부채
curliab	float	유동부채

balances	type	column
curdebt	float	단기차입금
longliab	float	비유동부채
longdebt	float	장기차입금
otherliab	float	기타금융업부채
equity	float	자본
equity2	float	자본금
equity3	float	신종자본증권
equity4	float	자본잉여금
equity5	float	기타자본
equity6	float	기타포괄손익누계액
equity7	float	이익잉여금(결손금)

[balances] : fnguide.com에서 재무상태표를 크롤링하여 저장할 테이블

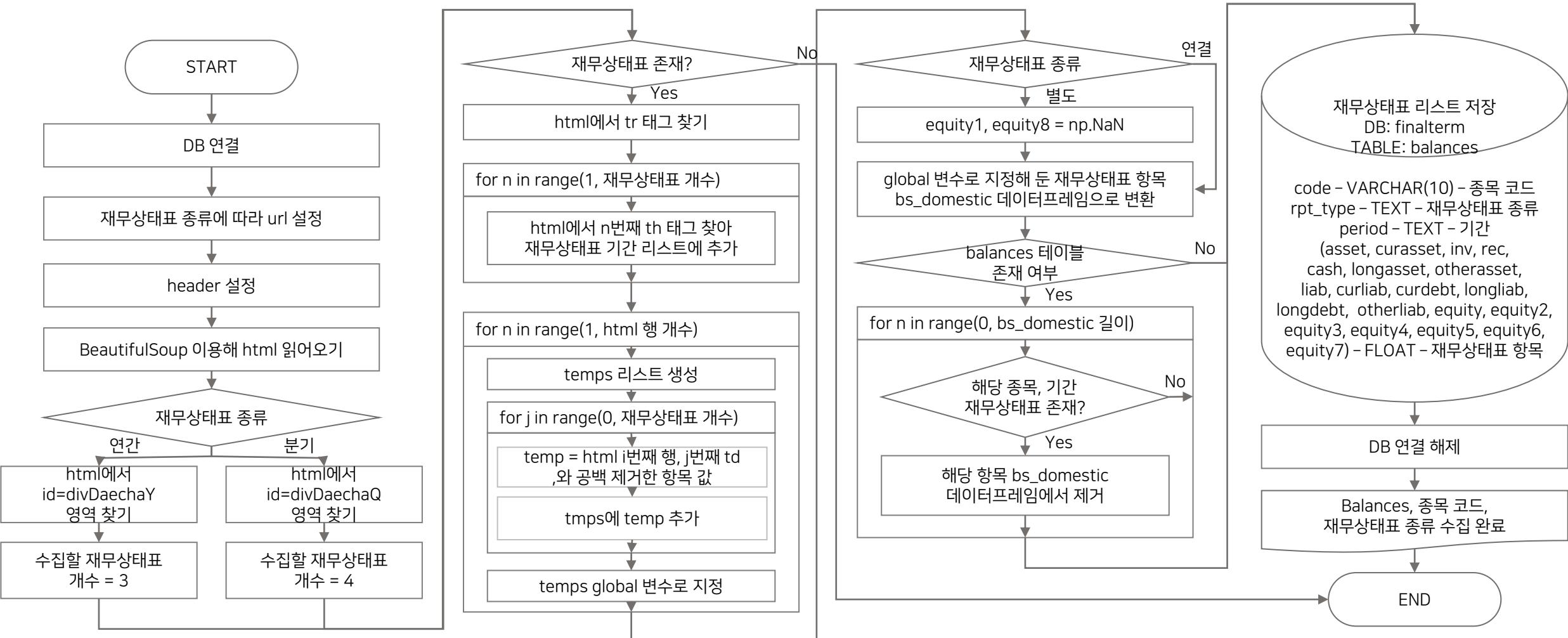
## 02. 재무 데이터 수집 및 저장 2) 재무상태표 클래스

재무상태표와 관련된 변수, 함수를 담고 있는 클래스

BALANCES
- bs_list : DataFrame
+ __init__(): void + setBalances(code, rpt_type, freq): void + getBalances(): DataFrame + getTermBalances(period): DataFrame + getStockBalances(code, period): DataFrame + getTrailingBalances(code, period): DataFrame

변수	type		설명
bs_list	DataFrame		balances 테이블에서 불러온 재무상태표 정보를 담을 DataFrame
함수	parameter	return	설명
setBalances()	code, rpt_type, freq	void	fnguide에서 해당 종목, 종류의 재무상태표를 크롤링하고, 데이터베이스에 저장하는 함수
getBalances()	void	DataFrame	balances 테이블에서 모든 재무상태표 데이터를 select한 후, DataFrame 형식으로 return하는 함수
getTermBalances()	period	DataFrame	balances 테이블에서 해당 분기의 재무상태표만 select한 후, DataFrame 형식으로 return하는 함수
getStockBalances()	code, period	DataFrame	balances 테이블에서 해당 종목의 해당 분기 재무상태표만 select한 후, DataFrame 형식으로 return하는 함수
getTrailingBalances()	code, period	DataFrame	balances 테이블에서 해당 종목의 해당 분기 포함 1년 간의 재무상태표를 select한 후, 트레일링 데이터로 만들어 DataFrame 형식으로 return하는 함수

## 02. 재무 데이터 수집 및 저장 2) 재무상태표 클래스 FlowChart



## 02. 재무 데이터 수집 및 저장 2) 재무상태표 클래스 코드 (set)

```
class BALANCES:
    bs_list = None # 재무상태표 리스트 저장할 변수

    # 재무상태표 리스트 초기화
    def __init__(self):
        self.bs_list = pd.DataFrame()

    # 재무상태표 수집하기
    def setBalances(self, code, rpt_type, freq):
        # sqlalchemy의 create_engine를 이용해 DB 연결
        server = '127.0.0.1' # local server
        user = 'root' # user name
        password = '0000' # 개인 password
        db = 'finalterm' # DB 이름

        engine = create_engine('mysql+pymysql://{}:{}@{}?charset=utf8'.format(user,password,server,db))

        items_en = ['asset', 'c_asset', 'inv', 'curassets2', 'curassets3', 'rec', 'curassets5', # 7
                   'curassets6', 'curassets7', 'curassets8', 'curassets9', 'cash', 'curassets11', # 6
                   'l_asset', 'ltassets1', 'ltassets2', 'ltassets3', 'ltassets4', 'ltassets5', 'ltassets6', 'ltassets7', # 8
                   'ltassets8', 'ltassets9', 'ltassets10', 'ltassets11', 'ltassets12', 'ltassets13', 'finassets', # 7
                   'liab', 'cl_liab', 'curlia1', 'curlia2', 'curlia3', 'curlia4', 'curlia5', # 7
                   'curlia6', 'curlia7', 'curlia8', 'curlia9', 'curlia10', 'curlia11', 'curlia12', 'curlia13', # 8
                   'li_liab', 'lt_liab', 'l_debt', 'ltliab3', 'ltliab4', 'ltliab5', 'ltliab6', # 7
                   'ltliab7', 'ltliab8', 'ltliab9', 'ltliab10', 'ltliab11', 'ltliab12', 'finliab', # 7
                   'equity', 'equity1', 'equity2', 'equity3', 'equity4', 'equity5', 'equity6', 'equity7', 'equity8'] # 9

        # 재무상태표 종류 (연간/분기)
        if freq.upper() == 'A': # 연간
            num_col = 3 # 최근 3개 연간 데이터
        else: # 분기 (freq.upper() == 'Q')
            num_col = 4 # 최근 4개 분기 데이터

        # 재무상태표 존재하지 않는 종목
        if bs_a is None:
            return

        bs_a = bs_a.findall(['tr'])

        # 최근 재무상태표 자료 수집
        period = [bs_a[0].find_all('th')[n].get_text() for n in range(1, num_col+1)]

        # 항목별 값 불러오기
        for item, i in zip(items_en, range(1, len(bs_a))):
            temps = []
            for j in range(0, num_col):
                temp = [float(bs_a[i].find_all('td')[j]['title'].replace(',', ''))replace('#xa0', ''))]
                if bs_a[i].find_all('td')[j]['title'].replace(',', '').replace('#xa0', '') != '':
                    else (0 if bs_a[i].find_all('td')[j]['title'].replace(',', '').replace('#xa0', '') == '-0' else 0)
                temps.append(temp[0])

            globals()[item] = temps # item_en 내 각 항목을 global 변수로 지정하고 값 저장
```

```
# 지배/비지배 항목 처리
if rpt_type.upper() == 'CONSOLIDATED': # 연간 재무상태표는 아무 것도 하지 않음
    pass
else: # 별도 연간 재무상태표는 연간에만 존재하는 항목을 Null값으로 채움
    globals()['equity1'], globals()['equity8'] = [np.NaN]*num_col, [np.NaN]*num_col
```

## 02. 재무 데이터 수집 및 저장 2) 재무상태표 클래스 코드 (set)

```
bs Domestic = pd.DataFrame({  
    'code':code,  
    'period':period,  
    'asset':asset, # 자산  
    'curasset':c_asset, # 유동자산  
    'inv':inv, # 재고자산  
    'rec':rec, # 매출채권  
    'cash':cash, # 현금 및 현금성자산  
    'longasset':l_asset, # 비유동자산  
    'otherasset':finassets, # 기타금융자산  
    'liab':liab, # 부채  
    'curliab':c_liab, # 유동부채  
    'curdebt':c_debt, # 단기차입금  
    'longliab':l_liab, # 비유동부채  
    'longdebt':l_debt, # 장기차입금  
    'otherliab':itliab6, # 기타금융업부채  
    'equity': equity, # 자본  
    'equity2': equity2, # 자본금  
    'equity3': equity3, # 신종자본증권  
    'equity4': equity4, # 자본잉여금  
    'equity5': equity5, # 기타자본  
    'equity6': equity6, # 기타포괄손익누계액  
    'equity7': equity7 # 이익잉여금(결손금)  
})  
  
# 재무제표 종료 컬럼 추가  
bs Domestic[rpt_type] = rpt_type+'_'+freq.upper()  
  
# DB 종목 데이터 있는지 체크 (종목코드, 기간, 재무상태표 종류)  
conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')  
cursor = conn.cursor()  
  
# balances 테이블 존재 여부 체크 (없는 경우 바로 DB에 저장)  
cursor.execute("SHOW TABLES LIKE 'balances'")  
check = cursor.fetchall()  
  
if len(check)!=0: # balances 테이블이 존재하는 경우 : 존재하는 데이터인지 체크  
    c = code # 종목코드  
    r = rpt_type+'_'+freq.upper() # 재무상태표 종류  
  
    for i in range(0, len(bs Domestic)): # 재무상태표 기간  
        sql = "SELECT * from balances where code='{}' and period='{}' and rpt_type='{}'".format(c, bs Domestic.loc[i]['period'], r)  
        df = pd.read_sql(sql, conn)  
        if not df.empty: # db에 존재하는 경우  
            print(code, bs Domestic.loc[i]['period'], r, "exist ... ")  
            bs Domestic = bs Domestic.drop([i], axis=0) # 삭제
```

```
# DB에 저장  
bs Domestic.to_sql(name='balances', con=engine, if_exists='append', index=False,  
dtype = {  
    'code' : sqlalchemy.types.VARCHAR(10),  
    'name' : sqlalchemy.types.TEXT(),  
    'rpt_type': sqlalchemy.types.TEXT(),  
    'period' : sqlalchemy.types.TEXT(),  
    'asset': sqlalchemy.types.FLOAT(),  
    'curasset': sqlalchemy.types.FLOAT(),  
    'inv':sqlalchemy.types.FLOAT(),  
    'rec':sqlalchemy.types.FLOAT(),  
    'cash':sqlalchemy.types.FLOAT(),  
    'longasset':sqlalchemy.types.FLOAT(),  
    'otherasset': sqlalchemy.types.FLOAT(),  
    'liab': sqlalchemy.types.FLOAT(),  
    'curliab': sqlalchemy.types.FLOAT(),  
    'curdebt': sqlalchemy.types.FLOAT(),  
    'longliab': sqlalchemy.types.FLOAT(),  
    'longdebt': sqlalchemy.types.FLOAT(),  
    'otherliab': sqlalchemy.types.FLOAT(),  
    'equity': sqlalchemy.types.FLOAT(),  
    'equity2': sqlalchemy.types.FLOAT(),  
    'equity3': sqlalchemy.types.FLOAT(),  
    'equity4': sqlalchemy.types.FLOAT(),  
    'equity5': sqlalchemy.types.FLOAT(),  
    'equity6': sqlalchemy.types.FLOAT(),  
    'equity7': sqlalchemy.types.FLOAT(),  
})  
  
# DB 접속 해제  
cursor.close()  
conn.close()  
engine.dispose()  
  
print("(Balances "+code+" "+rpt_type+"_"+freq.upper()+" ... OK")
```

## 02. 재무 데이터 수집 및 저장 2) 재무상태표 클래스 코드 (get)

```
# 재무상태표 불러오기
def getBalances(self):
    # db에 연결
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    # 전체 재무상태표 항목 불러와 bs_list에 할당
    sql = "SELECT * FROM balances"
    cursor.execute(sql)
    self.bs_list = pd.DataFrame(cursor.fetchall())

    # DB에 재무상태표 존재하지 않는 경우
    if len(self.bs_list) == 0:
        print("Empty Balances")
        return

    # 재무상태표 존재하는 경우 column명 수정
    self.bs_list.columns = [col[0] for col in cursor.description]

    # db 연결 해제
    cursor.close()
    conn.close()

    return self.bs_list

# 해당 분기의 재무상태표 불러오기
def getTermBalances(self, period):
    # db에 연결
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    # 분기 동일한 재무상태표 항목 불러와 bs_list에 할당
    sql = "SELECT * FROM balances WHERE period='{}' AND rpt_type='Consolidated_Q'".format(period)
    cursor.execute(sql)
    self.bs_list = pd.DataFrame(cursor.fetchall())

    # DB에 해당 분기 재무상태표 존재하지 않는 경우
    if len(self.bs_list) == 0:
        print(period, ": Empty Balances")
        return

    # 재무상태표 존재하는 경우 column명 수정
    self.bs_list.columns = [col[0] for col in cursor.description]

    # db 연결 해제
    cursor.close()
    conn.close()

    return self.bs_list
```

```
# 해당 종목, 해당 분기의 재무상태표 불러오기
def getStockBalances(self, code, period):
    # db에 연결
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    # 종목, 분기 동일한 재무상태표 항목 불러와 bs_list에 할당
    sql = "SELECT * FROM balances WHERE code='{}' AND period='{}' AND rpt_type='Consolidated_Q'".format(code, period)
    cursor.execute(sql)
    self.bs_list = pd.DataFrame(cursor.fetchall())

    # DB에 해당 종목 재무상태표 존재하지 않는 경우
    if len(self.bs_list) == 0:
        print(code, ": Empty Balances")
        return

    # 재무상태표 존재하는 경우 column명 수정
    self.bs_list.columns = [col[0] for col in cursor.description]

    # db 연결 해제
    cursor.close()
    conn.close()

    return self.bs_list

# 트레일링 데이터 만들기
def getTrailingBalances(self, code, period):
    df_quat = {}
    period_quat = period # 기간 보정 변수

    # 4분기 데이터 불러오기
    for i in range(4):
        df_bs = self.getStockBalances(code, period_quat) # 재무상태표

        # 분기 데이터 존재하지 않는 경우 트레일링 데이터 만들지 않음
        if df_bs is None:
            return

        df_quat[i] = df_bs

        if int(period_quat[5:7])-3 > 0: # period 마지막 2개 : 06, 08, 12 (2~4분기)
            period_quat = period_quat[0:4] + '/' + str(int(period_quat[5:7])-3).zfill(2) # 이전 분기로 만들기
        else: # 03 (1분기)
            period_quat = str(int(period_quat[0:4])-1) + '/12' # 이전 분기로 만들기 (작년 12)

    df_trailing = pd.DataFrame(df_quat[0], columns=['code', 'period', 'rpt_type'])
    df_trailing[df_quat[0].columns[2:-1]] = 0

    # 모든 데이터 더해 트레일링 데이터로 만들기
    for i in range(len(df_quat)):
        df_trailing[df_quat[0].columns[2:-1]] += df_quat[i][df_quat[i].columns[2:-1]]

    return df_trailing
```

## 02. 재무 데이터 수집 및 저장 2) 재무상태표 클래스 실행 예시

### getBalances() 실행 결과

```
g = BALANCES() # 재무상태표 클래스 생성
list = g.getBalances() # 전체 재무상태표 가져오기
list
```

	code	period	asset	curasset	inv	rec	cash	longasset	otherasset	liab	...	longdebt	otherliab	equity	equity2	equity3	equity4
0	060310	2019/03	510.63	161.40	15.30	52.60	67.10	349.24	0.0	180.10	...	16.80	0.00	330.53	223.86	0.0	438.64
1	060310	2020/03	482.53	180.23	34.50	35.72	53.92	302.30	0.0	158.71	...	25.00	0.00	323.82	224.01	0.0	439.11
2	060310	2021/03	518.31	160.21	26.13	43.33	52.03	358.10	0.0	169.24	...	20.00	0.00	349.07	228.38	0.0	452.44
3	095570	2019/12	12315.70	2738.84	162.73	523.63	447.21	9576.81	0.0	9211.17	...	1720.82	0.00	3104.49	468.22	0.0	1037.25
4	095570	2020/12	12041.40	1964.45	219.02	579.28	132.01	10076.90	0.0	9056.93	...	1504.36	4.38	2984.45	468.22	0.0	1037.25
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
34981	238490	2022/03	818.52	311.68	81.50	87.53	90.36	506.84	0.0	117.54	...	0.00	14.54	700.98	56.56	0.0	188.67
34982	238490	2021/06	943.26	466.60	117.21	88.78	205.83	476.66	0.0	208.07	...	0.00	15.60	735.19	56.56	0.0	195.07
34983	238490	2021/09	974.96	450.61	62.95	35.91	278.00	524.35	0.0	187.31	...	0.00	16.20	787.65	56.56	0.0	195.07
34984	238490	2021/12	842.93	336.14	93.60	43.62	154.81	506.80	0.0	103.77	...	0.00	14.57	739.16	56.56	0.0	188.67
34985	238490	2022/03	818.33	311.32	81.50	87.53	90.00	507.02	0.0	117.35	...	0.00	14.54	700.98	56.56	0.0	188.67

34986 rows × 23 columns

### getTermBalances('2022/03') 실행 결과

```
g = BALANCES() # 재무상태표 클래스 생성
list = g.getTermBalances('2022/03') # 해당 분기의 모든 종목 재무상태표 가져오기
list
```

	code	period	asset	curasset	inv	rec	cash	longasset	otherasset	liab	...	longdebt	otherliab	equity	equity2	equity3	equity4
0	095570	2022/03	13672.30	2618.73	289.26	925.59	705.72	11053.60	0.0	10021.90	...	2027.77	0.00	3650.41	468.22	0.0	11053.60
1	006840	2022/03	45365.00	13653.80	4098.77	4068.07	4536.19	31711.20	0.0	33975.30	...	10622.70	1542.21	11389.70	662.38	0.0	28500.00
2	054620	2022/03	3363.22	971.42	16.50	75.41	372.37	2391.80	0.0	1533.15	...	619.50	0.09	1830.07	108.96	0.0	1619.50
3	265520	2022/03	4668.89	3228.15	822.58	527.18	1524.00	1440.74	0.0	2659.40	...	730.00	26.04	2009.48	76.41	0.0	619.50
4	211270	2022/03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2325	189980	2022/03	1337.03	538.04	171.79	86.16	176.49	798.99	0.0	525.31	...	6.50	1.00	811.72	40.00	0.0	33975.30
2326	000540	2022/03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2327	003280	2022/03	2930.33	817.14	97.48	118.09	487.58	2113.19	0.0	1827.88	...	1.20	0.32	1102.45	1202.12	0.0	974.96
2328	037440	2022/03	1666.31	1257.95	0.00	799.93	260.45	408.36	0.0	1043.92	...	0.00	6.94	622.39	69.61	0.0	452.44
2329	238490	2022/03	818.52	311.68	81.50	87.53	90.36	506.84	0.0	117.54	...	0.00	14.54	700.98	56.56	0.0	117.54

2330 rows × 23 columns

### getStockBalances('352820', '2022/03') 실행 결과

```
g = BALANCES() # 재무상태표 클래스 생성
list = g.getStockBalances('352820', '2022/03') # 해당 종목코드, 분기의 재무상태표 가져오기
list
```

	code	period	asset	curasset	inv	rec	cash	longasset	otherasset	liab	...	longdebt	otherliab	equity	equity2	equity3	equity4
0	352820	2022/03	50007.4	20276.5	781.81	1894.58	4963.18	29730.8	0.0	18570.0	...	5919.45	193.85	31437.4	206.77	0.0	26208.7

1 rows × 23 columns

### getTrailingBalances('352820', '2022/03') 실행 결과

```
g = BALANCES() # 재무상태표 클래스 생성
list = g.getTrailingBalances('352820', '2022/03') # 해당 종목코드, 분기의 트레일링 데이터 가져오기
list
```

	code	period	rpt_type	asset	curasset	inv	rec	cash	longasset	otherasset	...	longliab	longdebt	otherliab	equity	equity2
0	352820	2022/03	Consolidated_Q	166596.5	68139.9	3421.1	7088.64	25019.82	98456.5	0.0	...	42184.01	23851.91	636.4	100722.5	803.8

1 rows × 17 columns

## 02. 재무 데이터 수집 및 저장 3) 현금흐름표 테이블

cashflow	type	column
code	varchar(10)	종목 코드
period	text	기간
rpt_type	float	현금흐름표 종류
cfo	float	영업현금흐름
cfo1	float	당기순손익
cfo2	float	법인세비용차감전계속사업이익
cfo3	float	현금유출이없는비용등가산
cfo4	float	감가상각비
cfo5	float	현금유입이없는수익등차감
cfo6	float	영업활동으로인한자산부채변동
cfo7	float	영업에서창출된현금흐름
cfo8	float	기타영업활동으로인한현금흐름
cfi	float	투자활동으로인한현금흐름

balances	type	column
cfi1	float	투자활동으로인한현금유입액
cfi2	float	투자활동으로인한현금유출액
cfi3	float	기타투자활동으로인한현금흐름
cff	float	재무활동으로인한현금흐름
cff1	float	재무활동으로인한현금유입액
cff2	float	재무활동으로인한현금유출액
cff3	float	기타재무활동으로인한현금흐름
cff4	float	영업투자재무활동기타현금흐름
cff5	float	연결범위변동으로인한현금의증가
cff6	float	환율변동증가
cff7	float	현금및현금성자산의증가
cff8	float	기초현금및현금성자산
cff9	float	기말현금및현금성자산

[cashflow] : fnguide.com 에서 현금흐름표를 크롤링하여 저장할 테이블

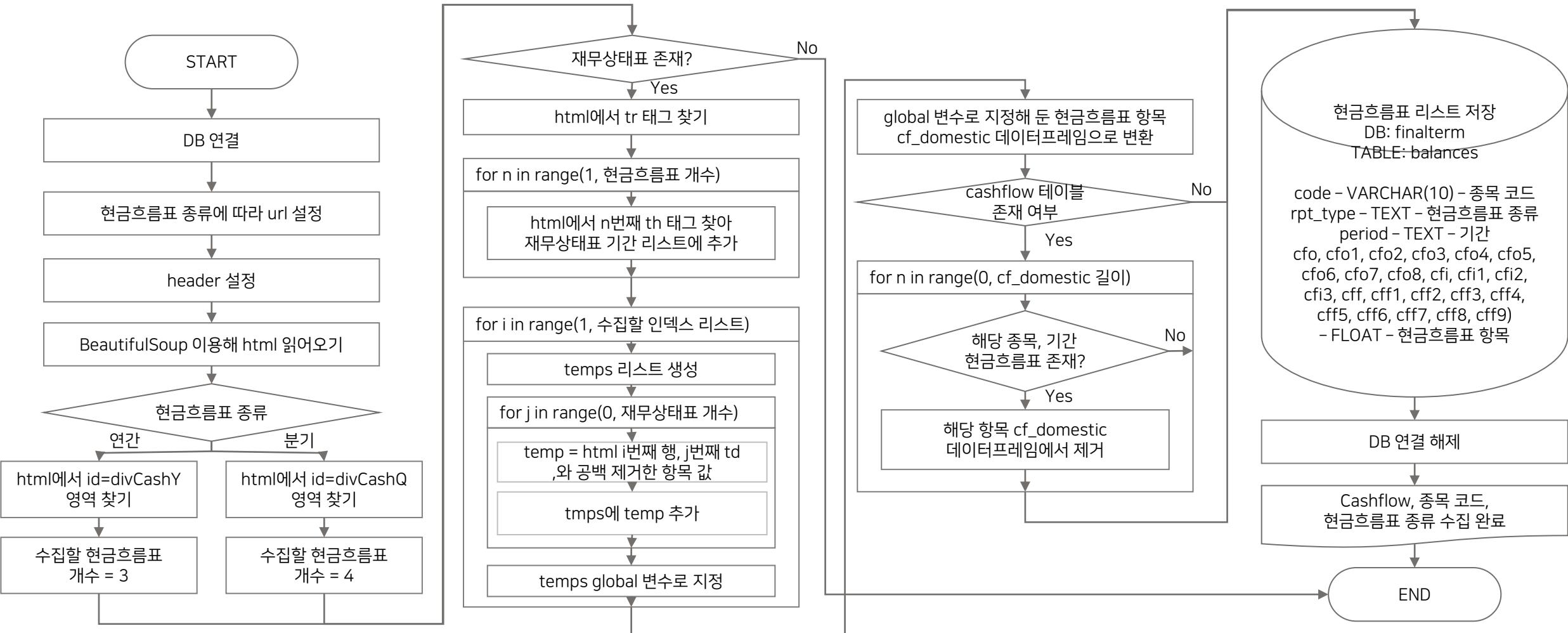
## 02. 재무 데이터 수집 및 저장 3) 현금흐름표 클래스

현금흐름표와 관련된 변수, 함수를 담고 있는 클래스

CASHFLOW
- cf_list : DataFrame
+ __init__(): void + setCashflow(code, rpt_type, freq): void + getCashflow(): DataFrame + getTermCashflow(period): DataFrame + getStockCashflow(code, period): DataFrame + getTrailingCashflow(code, period): DataFrame

변수	type		설명
cf_list	DataFrame		cashflow 테이블에서 불러온 현금흐름표 정보를 담을 DataFrame
함수	parameter	return	설명
setCashflow()	code, rpt_type, freq	void	fnguide에서 해당 종목, 종류의 현금흐름표를 크롤링하고, 데이터베이스에 저장하는 함수
getCashflow()	void	DataFrame	cashflow 테이블에서 모든 현금흐름표 데이터를 select한 후, DataFrame 형식으로 return하는 함수
getTermCashflow()	period	DataFrame	cashflow 테이블에서 해당 분기의 현금흐름표만 select한 후, DataFrame 형식으로 return하는 함수
getStockCashflow()	code, period	DataFrame	cashflow 테이블에서 해당 종목의 해당 분기 현금흐름표만 select한 후, DataFrame 형식으로 return하는 함수
getTrailingCashflow()	code, period	DataFrame	cashflow 테이블에서 해당 종목의 해당 분기 포함 1년 간의 현금흐름표를 select한 후, 트레일링 데이터로 만들어 DataFrame 형식으로 return하는 함수

## 02. 재무 데이터 수집 및 저장 3) 현금흐름표 클래스 FlowChart



## 02. 재무 데이터 수집 및 저장 3) 현금흐름표 클래스 코드 (set)

```
class CASHFLOW:  
    cf_list = None # 현금흐름표 리스트 저장을 변수  
  
    # 현금흐름표 리스트 초기화  
    def __init__(self):  
        self.cf_list = pd.DataFrame()  
  
    # 현금흐름표 수집하기  
    def setCashflow(self, code, rpt_type, freq):  
        # sqlalchemy의 create_engine를 이용해 DB 연결  
        server = '127.0.0.1' # local server  
        user = 'root' # user name  
        password = '0000' # 개인 password  
        db = 'finalterm' # DB 이름  
  
        engine = create_engine('mysql+pymysql://{}:{}@{}:{}/?charset=utf8'.format(user,password,server,db))  
  
        items_en = ['cfo', 'cfo1', 'cfo2', 'cfo3', 'cfo4', 'cfo5', 'cfo6', 'cfo7', 'cfo8', '#9  
                   'cff1', 'cff11', 'cff2', 'cff3', 'cff4', 'cff5', 'cff6', 'cff7', 'cff8', 'cff9'] #8  
  
        # 현금흐름표 종류  
        if rpt_type.upper() == 'CONSOLIDATED': # 연결 현금흐름표 (ReportGB=0)  
            url = 'http://comp.fnguide.com/SVO2/ASP/SVD_Finance.asp?pGB=1&gicode=A{}&cID=&MenuYn=Y&ReportGB=D&NewMenuID=103&stkGb=701'.format(code)  
            items_en = items_en + ['netinc1', 'netinc2']  
        else: # 별도 현금흐름표 (ReportGB=8)  
            url = 'http://comp.fnguide.com/SVO2/ASP/SVD_Finance.asp?pGB=1&gicode=A{}&cID=&MenuYn=Y&ReportGB=B&NewMenuID=103&stkGb=701'.format(code)  
  
        # Header 설정  
        headers = {  
            "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36  
        }  
        req = Request(url=url, headers=headers)  
        html = urlopen(req).read()  
        soup = BeautifulSoup(html, 'html.parser')
```

```
# 현금흐름표 종류  
if freq.upper() == 'A': # 연간 현금흐름표 영역 추출  
    cf_a = soup.find(id = 'divCashY')  
    num_col = 3 # 최근 3개년간 데이터  
else: # 분기 현금흐름표 영역 추출 (freq.upper() == 'Q')  
    cf_a = soup.find(id = 'divCashQ')  
    num_col = 4 # 최근 4개 분기 데이터  
  
# 현금흐름표 존재하지 않는 종목코드 체크  
if cf_a is None:  
    return  
  
cf_a = cf_a.find_all(['tr'])  
  
# 현금흐름표 항목 개수 맞지 않는 종목코드 체크  
if len(cf_a) < 158:  
    return  
  
# 현금흐름표 자료 수집  
period = [cf_a[0].find_all('th')[n].get_text() for n in range(1, num_col+1)]  
  
# 항목별 값 불러오기 (수집할 인덱스 값 미리 설정)  
idx = [1,2,3,4,9,39,70,75,76,84,85,99,113,121,122,134,145,153,154,155,156,157,158]  
for item, i in zip(items_en, idx):  
    temps = []  
    for j in range(0, num_col):  
        temp = [float(cf_a[i].find_all('td')[j]['title'].replace(',', ''))  
                .replace('\xa0', '')]#  
        if cf_a[i].find_all('td')[j]['title'].replace(',', '').replace('\xa0', '') != '#  
        else (0 if cf_a[i].find_all('td')[j]['title'].replace(',', '').replace('\xa0', '') == '-0' #  
              else 0)]  
  
    temps.append(temp[0])  
  
globals()[item] = temps # item_en 내 각 항목을 global 변수로 지정하고 값 저장  
  
items_en = ['cfo', 'cfo1', 'cfo2', 'cfo3', 'cfo4', 'cfo5', 'cfo6', 'cfo7', 'cfo8', '#9  
                   'cff1', 'cff11', 'cff2', 'cff3', 'cff4', 'cff5', 'cff6', 'cff7', 'cff8', 'cff9'] #8
```

## 02. 재무 데이터 수집 및 저장 3) 현금흐름표 클래스 코드 (set)

```
# 현금흐름표 주요컬럼값만 추출하여 데이터프레임으로 변환
cf Domestic = pd.DataFrame({
    'code':code,
    'period':period,
    'cto':cto, # 영업현금흐름
    'cfo1':cfo1, # 당기순손익
    'cto2':cto2, # 법인세비용차감전계속사업이익
    'cto3':cto3, # 현금유출이없는비용등가산
    'cto4':cto4, # 강가상각비
    'cto5':cto5, # 현금유입이없는수익등차감
    'cto6':cto6, # 영업활동으로인한자산부채변동(운전자본변동)
    'cto7':cto7, # 영업에서창출된현금흐름
    'cto8':cto8, # 기타영업활동으로인한현금흐름
    'ctf1':ctf1, # 투자활동으로인한현금흐름
    'ctf11':ctf11, # 투자활동으로인한현금유입액
    'ctf12':ctf12, # 투자활동으로인한현금유출액
    'ctf13':ctf13, # 기타투자활동으로인한현금흐름
    'ctf1f':ctf1f, # 재무활동으로인한현금흐름
    'ctf111':ctf111, # 재무활동으로인한현금유입액
    'ctf112':ctf112, # 재무활동으로인한현금유출액
    'ctf131':ctf131, # 기타재무활동으로인한현금흐름
    'ctf4':ctf4, # 영업투자재무활동기타현금흐름
    'ctf5':ctf5, # 연결법변동으로인한현금의증가
    'ctf6':ctf6, # 환율변동효과
    'ctf7':ctf7, # 현금및현금성자산의증가
    'ctf8':ctf8, # 기초현금및현금성자산
    'ctf9':ctf9 # 기말현금및현금성자산
})

cf Domestic['rpt_type'] = rpt_type+'_'+freq.upper() # 손익계산서 종류

# DB 종목 데이터 있는지 체크 (종목코드, 기간, 현금흐름표 종류)
conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
cursor = conn.cursor()

# cashflow 테이블 존재 여부 체크 (없는 경우 바로 DB에 저장)
cursor.execute('SHOW TABLES LIKE "cashflow"')
result = cursor.fetchall()

if len(result)!=0: # cashflow 테이블이 존재하는 경우 : 존재하는 데이터인지 체크
    r = rpt_type+'_'+freq.upper() # 현금흐름표 종류

    for i in range(0, len(cf Domestic)): # 현금흐름표 기간
        sql = "SELECT * from cashflow where code='{}' and period='{}' and rpt_type='{}'".format(code, cf Domestic.loc[i]['period'], r)
        df = pd.read_sql(sql, conn)
        if not df.empty: # db에 존재하는 경우
            print(code, cf Domestic.loc[i]['period'], r, "exist ... ")
            cf Domestic = cf Domestic.drop([i], axis=0) # 삭제
```

```
# DB에 저장
cf Domestic.to_sql(name='cashflow', con=engine, if_exists='append', index=False,
dtype = {
    'code' : sqlalchemy.types.VARCHAR(10),
    'name' : sqlalchemy.types.TEXT(),
    'rpt_type': sqlalchemy.types.TEXT(),
    'period' : sqlalchemy.types.TEXT(),
    'cto' : sqlalchemy.types.FLOAT(),
    'cfo1': sqlalchemy.types.FLOAT(),
    'cto2': sqlalchemy.types.FLOAT(),
    'cto3': sqlalchemy.types.FLOAT(),
    'cto4': sqlalchemy.types.FLOAT(),
    'cto5': sqlalchemy.types.FLOAT(),
    'cto6': sqlalchemy.types.FLOAT(),
    'cto7': sqlalchemy.types.FLOAT(),
    'cto8': sqlalchemy.types.FLOAT(),
    'ctf1': sqlalchemy.types.FLOAT(),
    'ctf11': sqlalchemy.types.FLOAT(),
    'ctf12': sqlalchemy.types.FLOAT(),
    'ctf13': sqlalchemy.types.FLOAT(),
    'ctf1f': sqlalchemy.types.FLOAT(),
    'ctf111': sqlalchemy.types.FLOAT(),
    'ctf112': sqlalchemy.types.FLOAT(),
    'ctf131': sqlalchemy.types.FLOAT(),
    'ctf4': sqlalchemy.types.FLOAT(),
    'ctf5': sqlalchemy.types.FLOAT(),
    'ctf6': sqlalchemy.types.FLOAT(),
    'ctf7': sqlalchemy.types.FLOAT(),
    'ctf8': sqlalchemy.types.FLOAT(),
    'ctf9': sqlalchemy.types.FLOAT()
})

# DB 접속 해제
cursor.close()
conn.close()
engine.dispose()

print("(Cashflow "+code+" "+rpt_type+"_"+freq.upper()+" ... OK")
```

## 02. 재무 데이터 수집 및 저장 3) 현금흐름표 클래스 코드 (get)

```
# 현금흐름표 불러오기
def getCashflow(self):
    # db에 연결
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    # 모든 현금흐름표 항목 불러와 cf_list에 할당
    sql = "SELECT * FROM cashflow"
    cursor.execute(sql)
    self.cf_list = pd.DataFrame(cursor.fetchall())

    # DB에 현금흐름표 존재하지 않는 경우
    if len(self.cf_list) == 0:
        print("Empty Cashflow")
        return

    # 현금흐름표 존재하는 경우 column명 설정
    self.cf_list.columns = [col[0] for col in cursor.description]

    # db 연결 해제
    cursor.close()
    conn.close()

    return self.cf_list

# 해당 분기의 현금흐름표 불러오기
def getTermCashflow(self, period):
    # db에 연결
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    # 분기 동일한 현금흐름표 항목 불러와 cf_list에 할당
    sql = "SELECT * FROM cashflow WHERE period='{}' AND rpt_type='Consolidated_Q'".format(period)
    cursor.execute(sql)
    self.cf_list = pd.DataFrame(cursor.fetchall())

    # DB에 해당 분기 현금흐름표 존재하지 않는 경우
    if len(self.cf_list) == 0:
        print(period, ": Empty Cashflow")
        return

    # 현금흐름표 존재하는 경우 column명 설정
    self.cf_list.columns = [col[0] for col in cursor.description]

    # db 연결 해제
    cursor.close()
    conn.close()

    return self.cf_list
```

```
# 해당 종목, 해당 분기의 현금흐름표 불러오기
def getStockCashflow(self, code, period):
    # db에 연결
    conn = pymysql.connect(host='localhost', user='root', password='0000', db='finalterm', charset='utf8')
    cursor = conn.cursor()

    # 종목, 분기 동일한 현금흐름표 항목 불러와 cf_list에 할당
    sql = "SELECT * FROM cashflow WHERE code='{}' AND period='{}' AND rpt_type='Consolidated_Q'".format(code, period)
    cursor.execute(sql)
    self.cf_list = pd.DataFrame(cursor.fetchall())

    # DB에 해당 종목 현금흐름표 존재하지 않는 경우
    if len(self.cf_list) == 0:
        print(code, ": Empty Cashflow")
        return

    # 현금흐름표 존재하는 경우 column명 설정
    self.cf_list.columns = [col[0] for col in cursor.description]

    # db 연결 해제
    cursor.close()
    conn.close()

    return self.cf_list

# 트레일링 데이터 만들기
def getTrailingCashflow(self, code, period):
    df_quat = {}
    period_quat = period # 기간 보정 변수

    # 4분기 데이터 불러오기
    for i in range(4):
        df_cf = self.getStockCashflow(code, period_quat) # 현금흐름표

        # 분기 데이터 존재하지 않는 경우 트레일링 데이터 만들지 않음
        if df_cf is None:
            return

        df_quat[i] = df_cf

        if int(period_quat[5:7])-3 > 0: # period 마지막 2자 : 06, 09, 12 (2-4분기)
            period_quat = period_quat[0:4] + '/' + str(int(period_quat[5:7])-3).zfill(2) # 이전 분기로 만들기
        else: # 03 (1분기)
            period_quat = str(int(period_quat[0:4])-1) + '/12' # 이전 분기로 만들기 (작년 12)

    df_trailing = pd.DataFrame(df_quat[0], columns=['code', 'period', 'rpt_type'])
    df_trailing[df_quat[0].columns[2:-1]] = 0

    # 모든 데이터 더해 트레일링 데이터로 만들기
    for i in range(len(df_quat)):
        df_trailing[df_quat[0].columns[2:-1]] += df_quat[i][df_quat[i].columns[2:-1]]

    return df_trailing
```

## 02. 재무 데이터 수집 및 저장 3) 현금흐름표 클래스 실행 예시

getCashflow () 실행 결과

```
g = CASHFLOW() # 현금흐름표 클래스 생성
list = g.getCashflow() # 전체 현금흐름표 가져오기
list
```

	code	period	cfo	cfo1	cfo2	cfo3	cfo4	cfo5	cfo6	cfo7	...	cff1	cff2	cff3	cff4	cff5	cff6	cff7	cff8	cff9
0	060310	2021/03	10.72	16.82	0.00	9.94	5.28	29.76	13.75	10.75	...	33.78	30.11	0.00	0.00	0.0	0.42	33.55	25.20	58.71
1	060310	2021/06	-0.40	-3.63	0.00	6.94	5.40	0.10	-1.80	1.41	...	20.78	8.69	0.00	0.00	0.0	0.10	-4.00	58.75	54.71
2	060310	2021/09	-2.77	14.83	0.00	7.18	5.61	1.73	-23.24	-2.96	...	0.04	3.27	0.00	0.00	0.0	1.43	-4.95	54.76	49.81
3	060310	2021/12	48.85	10.22	0.00	6.10	5.15	-0.75	32.30	49.38	...	0.05	13.17	0.00	0.00	0.0	-0.81	10.39	49.81	60.21
4	095570	2021/06	153.95	145.07	0.00	563.90	393.92	78.03	-433.25	197.69	...	1521.06	2302.78	-95.63	-16.47	0.0	-2.35	78.31	795.39	873.70
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
34967	037440	2022/03	-27.44	12.21	0.00	31.16	13.17	12.79	-49.94	-19.36	...	0.00	22.83	0.00	0.00	0.0	0.65	-110.62	366.30	255.61
34968	238490	2021/06	-17.99	0.00	-6.77	6.25	2.96	3.54	-9.20	-13.27	...	51.01	1.61	-0.06	0.00	0.0	0.20	25.04	180.80	205.81
34969	238490	2021/09	111.79	0.00	63.33	4.29	3.96	15.00	58.63	111.25	...	0.01	30.51	0.00	0.00	0.0	4.13	72.17	205.83	278.01
34970	238490	2021/12	-63.74	0.00	-68.33	36.96	3.74	-0.33	-49.21	-80.25	...	0.00	51.16	0.00	0.00	0.0	-0.07	-123.19	278.00	154.81
34971	238490	2022/03	-49.47	0.00	-15.83	5.41	3.60	13.98	-24.51	-48.92	...	20.01	11.39	-11.16	0.00	0.0	1.26	-64.80	154.81	90.00

34972 rows × 26 columns

getTermCashflow ('2022/03') 실행 결과

```
g = CASHFLOW() # 현금흐름표 클래스 생성
list = g.getTermCashflow('2022/03') # 해당 분기의 모든 종목 현금흐름표 가져오기
list
```

	code	period	cfo	cfo1	cfo2	cfo3	cfo4	cfo5	cfo6	cfo7	...	cff1	cff2	cff3	cff4	cff5	cff6	cff7	cff8	
0	095570	2022/03	29.86	108.55	0.00	524.40	389.26	24.39	-455.14	153.42	...	1837.53	1849.77	0.00	0.0	0.0	3.68	-146.60	852.32	7
1	006840	2022/03	-949.01	-480.39	0.00	1160.21	578.26	207.47	-1162.16	-689.81	...	2624.46	2181.29	-6.93	0.0	0.0	3.32	-616.14	5152.34	45
2	054620	2022/03	-12.20	-24.88	0.00	37.88	11.63	65.30	45.43	-6.87	...	10.27	31.82	0.00	0.0	0.0	0.21	-104.63	477.00	3
3	265520	2022/03	68.46	129.39	0.00	86.07	17.00	59.05	-25.43	130.96	...	102.68	5.76	0.00	0.0	0.0	17.06	167.26	1356.74	15
4	211270	2022/03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2325	189980	2022/03	-4.31	9.44	0.00	26.98	13.37	4.71	-28.39	3.31	...	2.85	3.61	0.00	0.0	0.0	-0.31	64.93	111.56	1
2326	000540	2022/03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	0.00	0.00	0.00	0.0	0.0	0.00	0.00	0.00	0.00
2327	003280	2022/03	47.01	14.07	0.00	111.38	32.09	69.73	-14.46	41.27	...	0.00	35.44	0.00	0.0	0.0	-6.23	105.70	381.89	4
2328	037440	2022/03	-27.48	12.02	0.00	31.18	13.17	12.99	-49.61	-19.39	...	1.20	22.83	0.00	0.0	0.0	0.70	-108.77	369.22	2
2329	238490	2022/03	-49.40	0.00	-15.83	5.39	3.61	13.98	-24.42	-48.85	...	20.01	11.39	-11.16	0.0	0.0	1.27	-64.73	155.09	0.00

2330 rows × 26 columns

getStockCashflow ('035720', '2022/03') 실행 결과

```
g = CASHFLOW() # 현금흐름표 클래스 생성
list = g.getStockCashflow('035720', '2022/03') # 해당 종목코드, 분기의 현금흐름표 가져오기
list
```

	code	period	cfo	cfo1	cfo2	cfo3	cfo4	cfo5	cfo6	cfo7	...	cff1	cff2	cff3	cff4	cff5	cff6	cff7	cff8	
0	035720	2022/03	372.57	0.0	17595.9	3611.92	806.28	17220.7	-3319.1	668.05	...	1867.84	1266.5	0.0	0.0	0.0	-183.31	-5072.55	53257.0	48184.4

1 rows × 26 columns

getTrailingCashflow ('035720', '2022/03') 실행 결과

```
g = CASHFLOW() # 현금흐름표 클래스 생성
list = g.getTrailingCashflow('035720', '2022/03') # 해당 종목코드, 분기의 트레일링 데이터 가져오기
list
```

	code	period	rpt_type	cfo	cfo1	cfo2	cfo3	cfo4	cfo5	cfo6	...	cff	cff1	cff2	cff3	cff4	cff5	cff6
0	035720	2022/03	Consolidated_Q	11157.88	0.0	37345.85	19569.88	2695.24	38765.02	-5238.37	...	38788.26	46874.44	7398.61	-687.43	0.0	0.0	-82.4

1 rows × 26 columns

---

03

## 가치주 종목 추출

---

## 03. 가치주 종목 추출 | 가치주 클래스

가치주 종목 추출을 위한 지표 계산과 관련된 변수, 함수를 담고 있는 클래스

변수	type	설명
period	str	지표 계산 기간
stocks	DataFrame	stocks 테이블에 저장된 모든 종목 정보
ohlcv	DataFrame	prices 테이블에 저장된 해당 분기 수정 주가 데이터
fin_unit	int	재무제표 단위 맞추기 위한 변수 (단위: 억원)

함수	parameter	return	설명
getPER()	void	DataFrame	DB에 저장된 모든 종목에 대해 PER 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getPBR()	void	DataFrame	PBR 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getPSR()	void	DataFrame	PSR 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getPCR()	void	DataFrame	PCR 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getEVEBITDA()	void	DataFrame	EV/EBITDA 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getEVSales()	void	DataFrame	EV/Sales 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getNCAV()	void	DataFrame	안전마진 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getPEG()	void	DataFrame	PEG 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수

## 03. 가치주 종목 추출 | 가치주 클래스 생성

### # 4. 가치주 종목 추출

#### ## 가치주 지표 계산 클래스

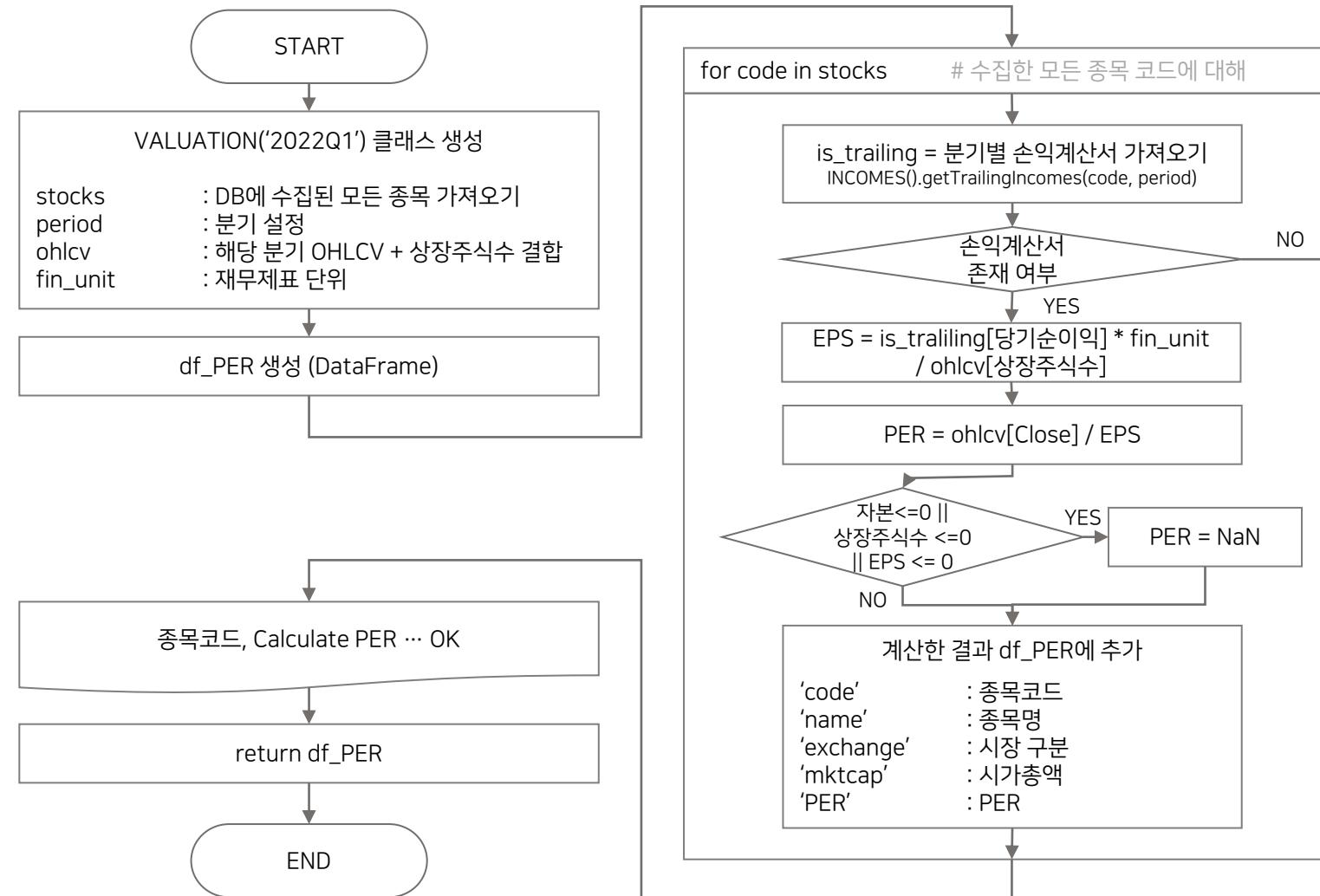
```
class VALUATION:
    period = None # 분기 할당 (str)
    stocks = None # 수집된 종목 할당 (DataFrame)
    ohlcv = None # 분기별 수정주가 할당 (DataFrame)
    fin_unit = 0 # 재무제표 단위 설정

    def __init__(self, term):
        print("init 실행 중", datetime.now())

        self.stocks = STOCK().getAllStocks() # 수집된 종목 가져오기
        self.period = termToPeriod(term) # 분기 설정
        self.ohlcv = PRICE().getTermOHLCV(term) # 해당 분기 수정주가 가져오기
        self.ohlcv = self.ohlcv.set_index('code') # 종목 코드를 인덱스로 사용
        self.fin_unit = 100000000 # 재무제표 단위 : 억 원

        print("init 실행 완료", datetime.now())
```

## 03. 가치주 종목 추출 1) 이익대비 저평가 종목 (PER)



## 03. 가치주 종목 추출 1) 이익대비 저평가 종목 (PER)

```
# 이익 대비 저평가 종목 (주가 수익 비율)
def getPER(self):
    # PER 계산 후 저장할 DataFrame 생성
    df_PER = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 손익계산서 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)

        # 손익계산서 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or code not in self.ohlcvs.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. EPS = 당기순이익/상장주식수
        # 당기순이익: incomes > netinc
        EPS = float((is_trailing['netinc'] * self.fin_unit) / self.ohlcvs.loc[code]['shares']) # ohlcvs : 해당 종목 코드의 상장주식수

        # 3. PER = 주가/EPS 계산
        PER = float(self.ohlcvs.loc[code]['Close'] / EPS)

        # 4. 결측치 처리
        if float(is_trailing['netinc']) == 0 or self.ohlcvs.loc[code]['shares'] == 0 or EPS == 0: # 0으로 나누는 경우
            PER = np.nan # np.nan 처리
        if PER < 0: # PER 값이 음수인 경우
            PER = 0 # 0으로 처리

        # 5. 계산한 결과 DataFrame에 추가
        df_PER = df_PER.append({
            'code': code,
            'name': name,
            'exchange': exc,
            'period': self.period, # 기간
            'mktcap': self.ohlcvs.loc[code]['mktcap'], # 시가총액
            'PER': PER }, ignore_index=True)

        print(code, ": Calculate PER ... OK")

    return df_PER
```

## 03. 가치주 종목 추출 1) 이익대비 저평가 종목 (PER)

getPER() 실행 결과: 모든 종목에 대해 PER 값 계산

```
v = VALUATION('2022Q1')
```

```
init 실행 중 2022-06-12 15:12:00.417741  
init 실행 완료 2022-06-12 15:18:19.434332
```

```
# PER 구하기
```

```
per_list = v.getPER()  
per_list
```

```
Empty DB Incomes
```

```
060310 : Not enough data to calculate ...  
095570 : Calculate PER ... OK  
006840 : Calculate PER ... OK  
054620 : Calculate PER ... OK  
265520 : Calculate PER ... OK  
211270 : Calculate PER ... OK  
027410 : Calculate PER ... OK  
282330 : Calculate PER ... OK  
032790 : Calculate PER ... OK  
138930 : Calculate PER ... OK  
001460 : Calculate PER ... OK  
Empty DB Incomes  
001465 : Not enough data to calculate ...
```

```
# PER 구하기
```

```
per_list = v.getPER()  
per_list
```

0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	3.499855
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	0.000000
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	12.243959
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	6.467678
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	NaN
...	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	27.084721
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	NaN
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	38.218970
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	19.427660
2181	238490	힘스	KOSDAQ	2022/03	1.032807e+11	0.000000

2182 rows × 6 columns

# 03. 가치주 종목 추출 1) 이익대비 저평가 종목 (PER)

```
# PER 종목 선정
df_factor = per_list.copy()
factor_list = ['PER']
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # PER, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	PER	score
1 023590	다우기술	1.251256	100.000000
2 001530	DI동일	1.279302	99.999974
3 000880	한화	1.443846	99.999820
4 024110	기업은행	1.697331	99.999584
5 139480	이마트	1.714299	99.999568
6 011200	HMM	1.714468	99.999568
7 950130	엑세스바이오	1.850054	99.999441
8 003380	하림지주	1.945864	99.999352
9 000210	DL	1.947041	99.999351
10 001230	동국제강	1.989159	99.999312
11 001120	LX인터내셔널	2.005910	99.999296
12 205470	휴마시스	2.058360	99.999247
13 298020	효성티엔씨	2.156513	99.999155
14 006120	SK디스커버리	2.187466	99.999127
15 034730	SK	2.203551	99.999112
16 011780	금호석유	2.520174	99.998816
17 031430	신세계인터내셔날	2.655080	99.998690
18 005880	대한해운	2.689223	99.998659
19 042670	현대두산인프라코어	2.882788	99.998478
20 001040	CJ	2.927732	99.998436
21 036460	한국가스공사	3.083438	99.998201
22 000070	삼양홀딩스	3.092482	99.998282
23 004020	현대제철	3.126576	99.998251
24 018670	SK가스	3.161570	99.998218
25 005490	POSCO홀딩스	3.207656	99.998175
26 010060	OCI	3.354670	99.998038
27 383220	F&F	3.476459	99.997924
28 004000	롯데정밀화학	3.556533	99.997849
29 001430	세아베스틸	3.636138	99.997775
30 004800	효성	3.655593	99.997757

## 1) PER 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
PER 1이상, 낮은 순 30개 종목 선정

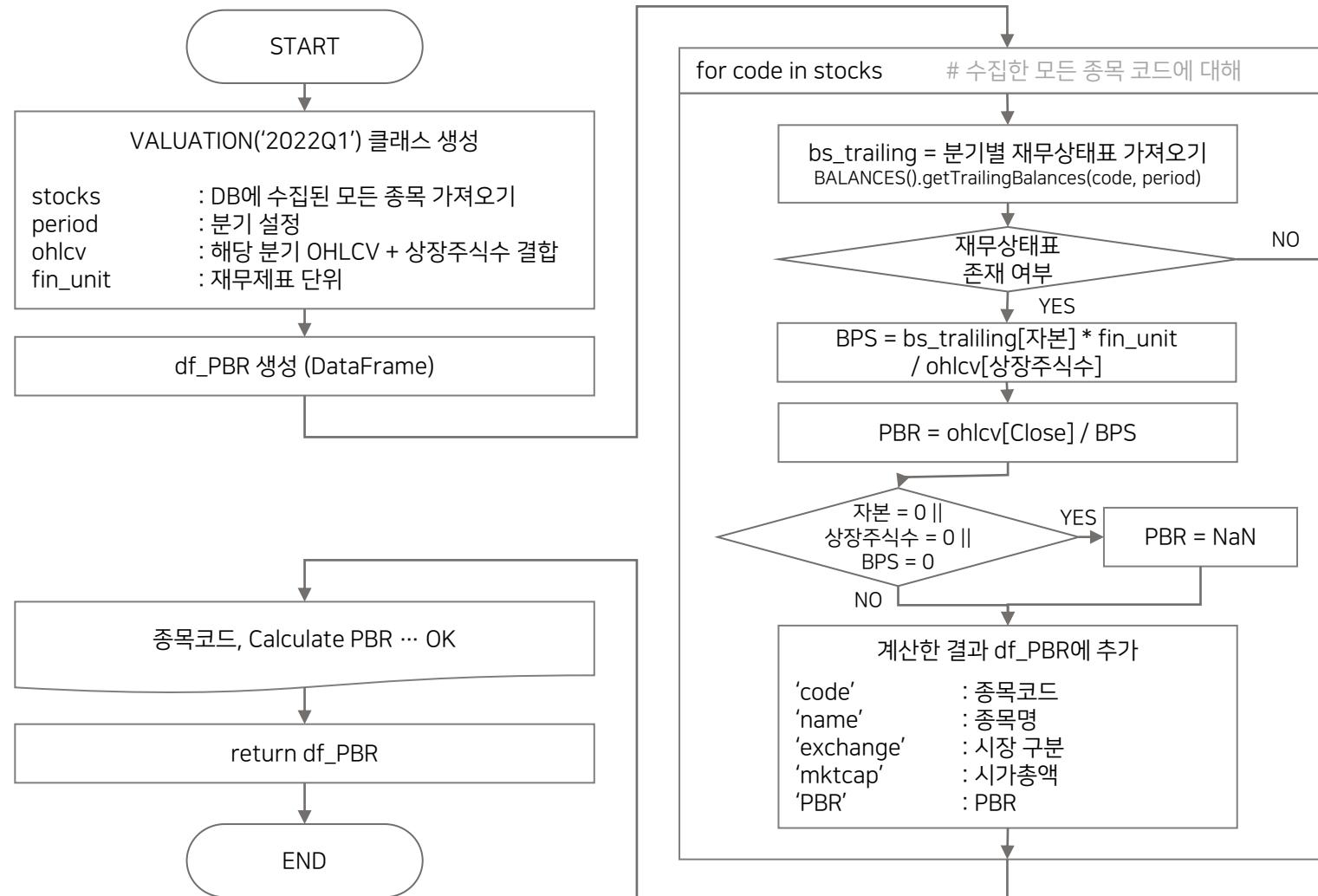
```
# PER 종목 선정
df_factor = per_list.copy()
factor_list = ['PER']
result = stock_select(df_factor, 0.2, 'KOSDAQ', 30, factor_list) # PER, 시가총액 상위 20%, KOSDAQ 종목 중 30개 선택
result
```

code	name	PER	score
1 950130	엑세스바이오	1.850054	100.000000
2 003380	하림지주	1.945864	99.998036
3 205470	휴마시스	2.058360	99.997305
4 096530	씨젠	4.951057	99.364409
5 049070	인탑스	5.454860	99.261148
6 090460	비에이치	6.851420	98.974904
7 108320	LX세미콘	6.892517	98.966481
8 086520	에코프로	7.108488	98.922215
9 121800	비덴트	7.802206	98.780028
10 036930	주성엔지니어링	8.104694	98.718029
11 091700	파트론	8.375534	98.662517
12 178320	서진시스템	8.701677	98.595670
13 095610	테스	8.838782	98.567568
14 056190	에스에프에이	9.765314	98.377663
15 074600	원익QnC	9.814246	98.367634
16 067310	하나마이크론	9.989650	98.331683
17 222800	심텍	10.335902	98.260714
18 319660	피에스케이	10.453699	98.236570
19 033290	코월패션	10.497714	98.227548
20 106240	파인테크닉스	10.509669	98.225098
21 215000	골프존	10.568552	98.213029
22 043150	바텍	10.708485	98.184348
23 272290	이녹스첨단소재	11.173492	98.089039
24 035600	KG이니시스	11.219807	98.079546
25 095660	네오위즈	11.332485	98.056451
26 293490	카카오게임즈	11.507073	98.020667
27 069080	월전	11.709526	97.979172
28 357780	솔브레인	11.986233	97.922457
29 183300	코미코	12.133537	97.892265
30 112040	웨메이드	12.218704	97.874809

## 2) PER 기준 종목 선정 결과

시가총액 상위 20% KOSDAQ 종목에 대해  
PER 1이상, 낮은 순 30개 종목 선정

## 03. 가치주 종목 추출 2) 장부가치대비 저평가 종목 (PBR)



## 03. 가치주 종목 추출 2) 장부가치대비 저평가 종목 (PBR)

```
# 장부가치 대비 저평가 종목 (주가 순자산 비율)
def getPBR(self):
    # PBR 계산 후 저장할 DataFrame 생성
    df_PBR = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 끝기별 재무상태표 가져오기
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)

        # 재무상태표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if bs_trailing is None or code not in self.ohlcvs.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. BPS = 자본/상장주식수 계산
        # 자본: balances > equity
        BPS = float((bs_trailing['equity']*self.fin_unit) / self.ohlcvs.loc[code]['shares']) # ohlcvs : 해당 종목 코드의 상장주식수

        # 3. PBR = 주가/BPS 계산
        PBR = float(self.ohlcvs.loc[code]['Close'] / BPS)

        # 4. 결측치 처리
        if float(bs_trailing['equity']) == 0 or self.ohlcvs.loc[code]['shares'] == 0 or BPS == 0: # 0으로 나누는 경우
            PBR = np.nan # np.nan 처리
        if PBR < 0: # PBR 값이 음수인 경우
            PBR = 0 # 0으로 처리

        # 5. 계산한 결과 DataFrame에 추가
        df_PBR = df_PBR.append({
            'code': code,
            'name': name,
            'exchange': exc,
            'period': self.period,
            'mktcap': self.ohlcvs.loc[code]['mktcap'],
            'PBR': PBR }, ignore_index=True)

        print(code, ": Calculate PBR ... OK")

    return df_PBR
```

## 03. 가치주 종목 추출 2) 장부가치대비 저평가 종목 (PBR)

getPBR() 실행 결과: 모든 종목에 대해 PBR 값 계산

```
# PBR 구하기  
pbr_list = v.getPBR()  
pbr_list
```

```
100220 : calculate PBR ... OK  
337930 : Calculate PBR ... OK  
099390 : Calculate PBR ... OK  
064480 : Calculate PBR ... OK  
288330 : Calculate PBR ... OK  
365900 : Calculate PBR ... OK  
251630 : Calculate PBR ... OK  
018290 : Calculate PBR ... OK  
044480 : Calculate PBR ... OK  
033560 : Calculate PBR ... OK  
Empty DB Balances  
191600 : Not enough data to calculate ...  
369370 : Not enough data to calculate ...  
126340 : Calculate PBR ... OK  
121800 : Calculate PBR ... OK  
148140 : Calculate PBR ... OK  
082800 : Calculate PBR ... OK  
318410 : Calculate PBR ... OK  
002070 : Calculate PBR ... OK  
100220 : Calculate PBR ... OK
```

```
# PBR 구하기  
pbr_list = v.getPBR()  
pbr_list
```

0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	0.207604
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	0.063523
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	0.348534
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	0.496519
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	2.845191
...	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	0.496487
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	0.829170
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	1.538758
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	0.545216
2181	238490	힐스	KOSDAQ	2022/03	1.032807e+11	0.348570

2182 rows × 6 columns

## 03. 가치주 종목 추출 2) 장부가치대비 저평가 종목 (PBR)

```
# PBR 종목 선정
df_factor = pbr_list.copy()
factor_list = ['PBR']
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # PBR, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	PBR	score
1 036460	한국가스공사	0.103828	100.000000
2 267270	현대건설기계	0.108750	99.999995
3 004170	신세계	0.112386	99.999992
4 005380	현대차	0.117388	99.999987
5 011170	롯데케미칼	0.117520	99.999987
6 005490	POSCO홀딩스	0.118080	99.999986
7 011210	현대위아	0.118881	99.999985
8 003470	유안타증권	0.120237	99.999984
9 001740	SK네트웍스	0.123329	99.999981
10 034310	NICE	0.123571	99.999981
11 018670	SK가스	0.125308	99.999979
12 029780	삼성카드	0.126229	99.999978
13 009540	한국조선해양	0.126809	99.999978
14 006800	미래에셋증권	0.127412	99.999977
15 034220	LG디스플레이	0.127632	99.999977
16 161390	한국타이어앤테크놀로지	0.128108	99.999977
17 097950	CJ제일제당	0.129005	99.999976
18 003550	LG	0.132706	99.999972
19 006650	대한유화	0.134723	99.999970
20 002790	아모레G	0.138439	99.999967
21 002380	KCC	0.138566	99.999966
22 298040	효성중공업	0.139740	99.999965
23 280360	롯데제과	0.142406	99.999963
24 005940	NH투자증권	0.143418	99.999962
25 030200	KT	0.143582	99.999962
26 036830	솔브레인홀딩스	0.143742	99.999961
27 000720	현대건설	0.143751	99.999961
28 012330	현대모비스	0.145167	99.999960
29 001120	LX인터넷서울	0.146550	99.999959
30 001230	동국제강	0.147859	99.999957

### 1) PBR 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
PBR 0.1이상, 낮은 순 30개 종목 선정

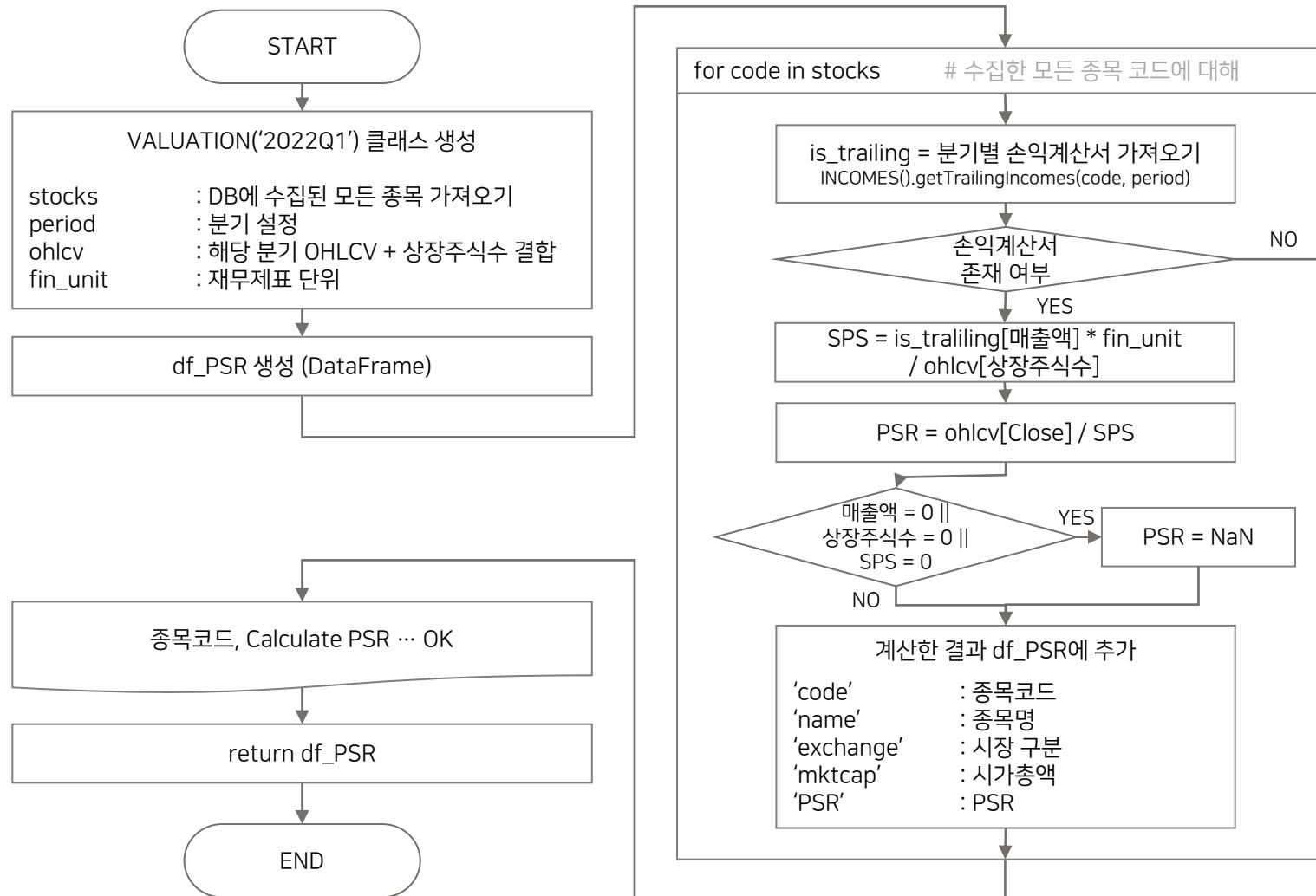
```
# PBR 종목 선정
df_factor = pbr_list.copy()
factor_list = ['PBR']
result = stock_select(df_factor, 0.2, 'KOSDAQ', 30, factor_list) # PBR, 시가총액 상위 20%, KOSDAQ 종목 중 30개 선택
result
```

code	name	PBR	score
1 036830	솔브레인홀딩스	0.143742	100.000000
2 035760	CJ ENM	0.173449	99.870179
3 038500	삼표시멘트	0.218696	99.672447
4 178320	서진시스템	0.241380	99.573319
5 046890	서울반도체	0.243720	99.563091
6 049070	인탑스	0.247395	99.547031
7 034230	파라다이스	0.253088	99.522154
8 056190	에스에프에이	0.271509	99.441651
9 030520	한글과컴퓨터	0.272344	99.438005
10 078340	컴투스	0.274395	99.429040
11 195940	HK이노엔	0.308005	99.282165
12 095660	네오워즈	0.308024	99.282081
13 035600	KG이니시스	0.311811	99.265530
14 091700	파트론	0.346713	99.113009
15 085660	차바이오텍	0.356004	99.072408
16 041190	우리기술투자	0.381323	98.961763
17 033290	코렐파션	0.403351	98.865498
18 060150	인선이엔티	0.405822	98.854698
19 217270	넵튠	0.412775	98.824315
20 069080	월전	0.435142	98.726568
21 086520	에코프로	0.458470	98.624625
22 074600	원익QnC	0.466403	98.589956
23 095610	테스	0.470403	98.572479
24 086900	메디톡스	0.483584	98.514875
25 025980	아남티	0.490799	98.483346
26 095700	제넥신	0.498883	98.448019
27 025900	동화기업	0.501117	98.438257
28 086450	동국제약	0.506204	98.416025
29 950130	엑세스바이오	0.510067	98.399142
30 145020	휴젤	0.518958	98.360289

### 2) PBR 기준 종목 선정 결과

시가총액 상위 20% KOSDAQ 종목에 대해  
PBR 0.1이상, 낮은 순 30개 종목 선정

## 03. 가치주 종목 추출 3) 매출대비 저평가 종목 (PSR)



## 03. 가치주 종목 추출 3) 매출대비 저평가 종목 (PSR)

```
# 매출 대비 저평가 종목 (주가 매출 비율)
def getPSR(self):
    # PSR 계산 후 저장할 DataFrame 생성
    df_PSR = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 손익계산서 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)

        # 손익계산서 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or code not in self.ohlcvs.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. SPS = 매출액/상장주식수 계산
        # 매출액: incomes > rev
        SPS = float((is_trailing['rev'] * self.fin_unit) / self.ohlcvs.loc[code]['shares']) # ohlcvs : 해당 종목 코드의 상장주식수

        # 3. PSR = 주가/SPS 계산
        PSR = float(self.ohlcvs.loc[code]['Close'] / SPS)

        # 4. 결측치 처리
        if float(is_trailing['rev']) == 0 or self.ohlcvs.loc[code]['shares'] == 0 or SPS == 0: # 0으로 나누는 경우
            PSR = np.nan # np.nan 처리
        if PSR < 0: # PSR 값이 음수인 경우
            PSR = 0 # 0으로 처리

        # 5. 계산한 결과 DataFrame에 추가
        df_PSR = df_PSR.append({
            'code': code,
            'name': name,
            'exchange': exc,
            'period': self.period,
            'mktcap': self.ohlcvs.loc[code]['mktcap'],
            'PSR': PSR }, ignore_index=True)

    print(code, ": Calculate PSR ... OK")

return df_PSR
```

## 03. 가치주 종목 추출 3) 매출대비 저평가 종목 (PSR)

getPSR() 실행 결과: 모든 종목에 대해 PSR 값 계산

```
# PSR 구하기  
psr_list = v.getPSR()  
psr_list
```

```
137950 : Calculate PSR ... OK  
033320 : Calculate PSR ... OK  
348950 : Calculate PSR ... OK  
204270 : Not enough data to calculate ...  
194370 : Calculate PSR ... OK  
026040 : Calculate PSR ... OK  
126880 : Calculate PSR ... OK  
322510 : Calculate PSR ... OK  
Empty DB Incomes  
254160 : Not enough data to calculate ...  
033050 : Calculate PSR ... OK  
094970 : Not enough data to calculate ...  
058420 : Calculate PSR ... OK  
025620 : Calculate PSR ... OK  
036420 : Calculate PSR ... OK  
089790 : Calculate PSR ... OK  
030000 : Calculate PSR ... OK  
052670 : Calculate PSR ... OK  
271980 : Calculate PSR ... OK  
001560 : Calculate PSR ... OK
```

```
# PSR 구하기  
psr_list = v.getPSR()  
psr_list
```

0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	0.267959
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	0.087992
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	6.429352
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	0.731407
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	NaN
...	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	2.017709
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	NaN
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	7.794003
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	0.616329
2181	238490	힐스	KOSDAQ	2022/03	1.032807e+11	2.106179

2182 rows × 6 columns

## 03. 가치주 종목 추출 3) 매출대비 저평가 종목 (PSR)

```
# PSR 종목 선정
df_factor = psr_list.copy()
factor_list = ['PSR']
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # PSR 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	PSR	score
1 006120	SK디스커버리	0.104689	100.000000
2 003690	코리안리	0.104787	99.999996
3 036460	한국가스공사	0.108189	99.999865
4 001740	SK네트웍스	0.108819	99.999841
5 003070	코오롱글로벌	0.116367	99.999549
6 006260	LS	0.122370	99.999318
7 000150	두산	0.125492	99.999197
8 267250	현대중공업지주	0.127317	99.999127
9 018670	SK가스	0.143112	99.998518
10 001450	현대해상	0.150041	99.998250
11 023590	다우기술	0.150923	99.998216
12 139480	이마트	0.150938	99.998216
13 031430	신세계인터내셔날	0.155893	99.998025
14 108670	LX하우시스	0.161860	99.997794
15 034730	SK	0.167583	99.997574
16 001430	세아베스틸	0.170261	99.997470
17 023530	롯데쇼핑	0.175246	99.997278
18 298040	효성중공업	0.183432	99.996962
19 049770	동원F&B	0.192383	99.996617
20 001230	동국제강	0.200418	99.996307
21 097950	CJ제일제당	0.205327	99.996117
22 085620	미래에셋생명	0.206990	99.996053
23 298020	효성티엔씨	0.215951	99.995707
24 145990	상양사	0.218437	99.995612
25 267270	현대건설기계	0.219870	99.995556
26 001680	대상	0.221778	99.995483
27 004020	현대제철	0.222658	99.995449
28 000070	삼양홀딩스	0.228367	99.995228
29 011210	현대워아	0.230698	99.995138
30 015760	한국전력	0.234722	99.994983

### 1) PSR 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
PSR 0.1이상, 낮은 순 30개 종목 선정

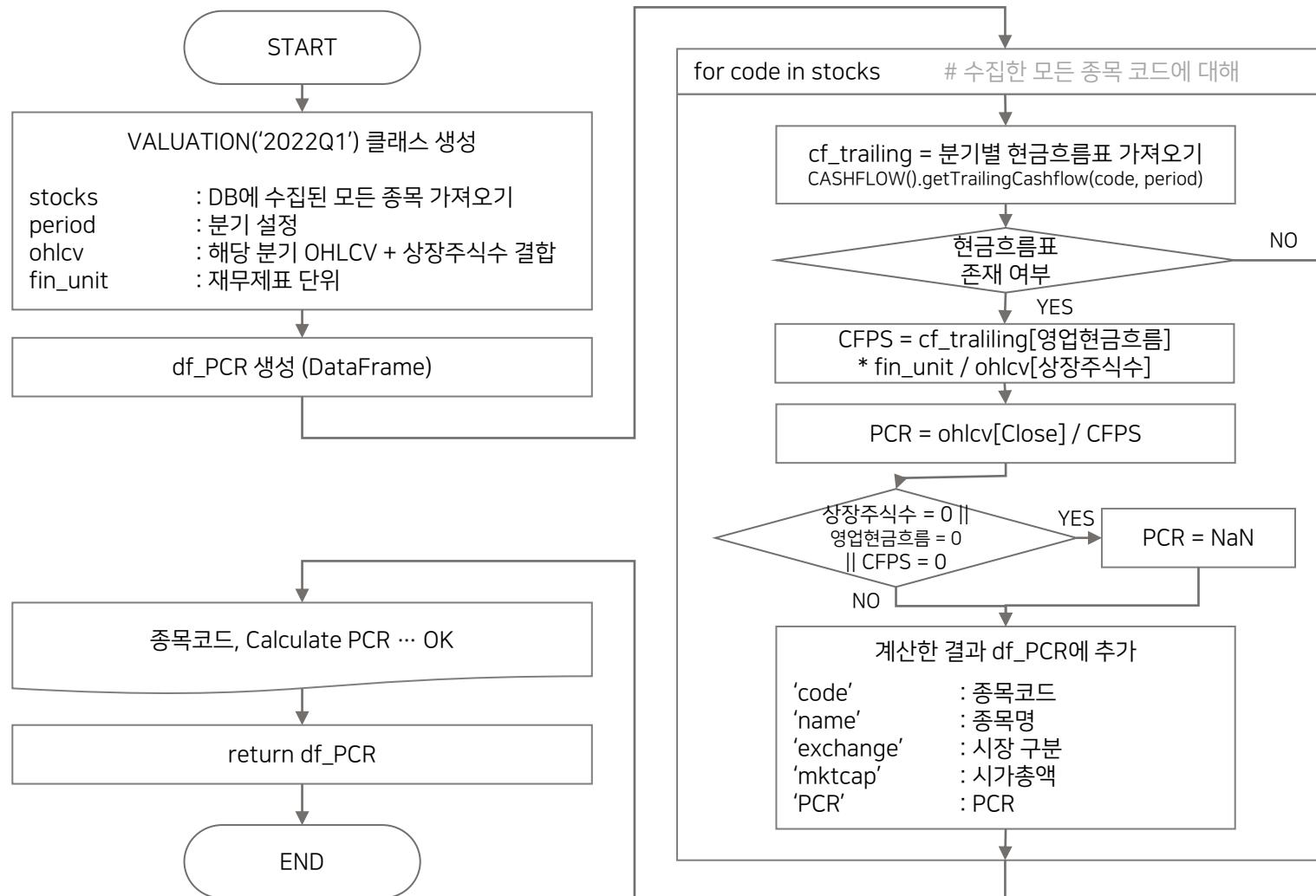
```
# PSR 종목 선정
df_factor = psr_list.copy()
factor_list = ['PSR']
result = stock_select(df_factor, 0.2, 'KOSDAQ', 30, factor_list) # PSR 시가총액 상위 20%, KOSDAQ 종목 중 30개 선택
result
```

code	name	PSR	score
1 091700	파트론	0.477121	100.000000
2 049070	인탑스	0.531317	99.997909
3 950130	엑세스바이오	0.602210	99.995173
4 178320	서진시스템	0.605168	99.995059
5 035600	KG이니시스	0.629129	99.994135
6 046890	서울반도체	0.643174	99.993593
7 090460	비에이치	0.652457	99.993234
8 035760	CJ ENM	0.775059	99.988504
9 033290	코월패션	0.825607	99.986553
10 056190	에스에프에이	0.909761	99.983306
11 038500	삼표시멘트	0.928283	99.982591
12 205470	휴마시스	0.986499	99.980345
13 022100	포스코 ICT	1.035379	99.978459
14 074600	원익QnC	1.042388	99.978189
15 108320	LX세미콘	1.137081	99.974535
16 086520	에코프로	1.148806	99.974082
17 222800	실텍	1.188882	99.972536
18 067310	하나마이크론	1.201757	99.972039
19 106240	파인테크닉스	1.231067	99.970908
20 100090	삼강엘앤티	1.243003	99.970448
21 060250	NHN한국사이버결제	1.251600	99.970116
22 036830	솔브레인홀딩스	1.428432	99.963293
23 085660	차바이오텍	1.429716	99.963243
24 084850	아이티엠반도체	1.489850	99.960923
25 033640	네페스	1.517848	99.959842
26 030520	한글과컴퓨터	1.523531	99.959623
27 091120	이엠텍	1.531287	99.959324
28 086450	동국제약	1.551837	99.958531
29 240810	원익IPS	1.576224	99.957590
30 025900	동화기업	1.605504	99.956460

### 2) PSR 기준 종목 선정 결과

시가총액 상위 20% KOSDAQ 종목에 대해  
PSR 0.1이상, 낮은 순 30개 종목 선정

## 03. 가치주 종목 추출 4) 현금흐름대비 저평가 종목 (PCR)



## 03. 가치주 종목 추출 4) 현금흐름대비 저평가 종목 (PCR)

```
# 현금흐름 대비 저평가 종목 (주가 현금흐름 비율)
def getPCR(self):
    # PCR 계산 후 저장할 DataFrame 생성
    df_PCR = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 현금흐름표 가져오기
        cf_trailing = CASHFLOW().getTrailingCashflow(code, self.period)

        # 현금흐름표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if cf_trailing is None or code not in self.ohlcvs.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. CFPS = 영업현금흐름/상장주식수 계산
        # 영업현금흐름: cashflow > cfo
        CFPS = float((cf_trailing['cfo'] * self.fin_unit) / self.ohlcvs.loc[code]['shares']) # ohlcvs : 해당 종목 코드의 상장주식수

        # 3. PCR = 주가/CFPS 계산
        PCR = float(self.ohlcvs.loc[code]['Close'] / CFPS)

        # 4. 결측치 처리
        if float(cf_trailing['cfo']) == 0 or self.ohlcvs.loc[code]['shares'] == 0 or CFPS == 0: # 0으로 나누는 경우
            PCR = np.nan # np.nan 처리
        if PCR < 0: # PCR 값이 음수인 경우
            PCR = 0 # 0으로 처리

        # 5. 계산한 결과 DataFrame에 추가
        df_PCR = df_PCR.append({
            'code': code,
            'name': name,
            'exchange': exc,
            'period': self.period,
            'mktcap': self.ohlcvs.loc[code]['mktcap'],
            'PCR': PCR }, ignore_index=True)

        print(code, ": Calculate PCR ... OK")

    return df_PCR
```

## 03. 가치주 종목 추출 4) 현금흐름대비 저평가 종목 (PCR)

getPCR() 실행 결과: 모든 종목에 대해 PCR 값 계산

# PCR 구하기

```
pcr_list = v.getPCR()  
pcr_list  
  
016920 : Calculate PCR ... OK  
284620 : Calculate PCR ... OK  
035720 : Calculate PCR ... OK  
293490 : Calculate PCR ... OK  
323410 : Not enough data to calculate ...  
377300 : Calculate PCR ... OK  
042000 : Calculate PCR ... OK  
006380 : Calculate PCR ... OK  
317530 : Calculate PCR ... OK  
071850 : Calculate PCR ... OK  
050110 : Calculate PCR ... OK  
223310 : Calculate PCR ... OK  
109070 : Calculate PCR ... OK  
900310 : Calculate PCR ... OK  
078340 : Calculate PCR ... OK  
063080 : Calculate PCR ... OK  
307930 : Calculate PCR ... OK  
263700 : Calculate PCR ... OK  
214370 : Calculate PCR ... OK  
221980 : Calculate PCR ... OK
```

# PCR 구하기

```
pcr_list = v.getPCR()  
pcr_list  
  
0 095570 AJ네트웍스 KOSPI 2022/03 2.898300e+11 0.000000  
1 006840 AK홀딩스 KOSPI 2022/03 2.907840e+11 0.000000  
2 054620 APS홀딩스 KOSDAQ 2022/03 2.488095e+11 0.000000  
3 265520 AP시스템 KOSDAQ 2022/03 3.629337e+11 5.332948  
4 211270 AP위성 KOSDAQ 2022/03 2.307593e+11 NaN  
... ... ... ... ... ...  
2177 189980 흥국에프엔비 KOSDAQ 2022/03 1.557913e+11 7.892564  
2178 000540 흥국화재 KOSPI 2022/03 2.601827e+11 NaN  
2179 003280 흥아해운 KOSPI 2022/03 6.996365e+11 0.000000  
2180 037440 희림 KOSDAQ 2022/03 1.322635e+11 7.803157  
2181 238490 힘스 KOSDAQ 2022/03 1.032807e+11 0.000000
```

2182 rows × 6 columns

# 03. 가치주 종목 추출 4) 현금흐름대비 저평가 종목 (PCR)

```
# PCR 종목 선정
df_factor = pcr_list.copy()
factor_list = ['PCR']
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # PCR 시가총액 상위 20%, 전체 종목 중 30개 선정
result
```

code	name	PCR	score
1 000880	한화	0.484545	100.000000
2 000370	한화손해보험	0.570086	99.996033
3 088350	한화생명	0.758490	99.987295
4 183190	아세아시멘트	0.866372	99.982292
5 950130	엑세스바이오	0.933140	99.979195
6 001040	CJ	0.933964	99.979157
7 003380	하림지주	1.188721	99.967342
8 003690	코리안리	1.287323	99.962769
9 020560	아시아나항공	1.289471	99.962669
10 034220	LG디스플레이	1.346730	99.960014
11 011200	HMM	1.488586	99.953435
12 031430	신세계인터넷내셔널	1.571906	99.949571
13 023530	롯데쇼핑	1.675421	99.944770
14 001800	오리온홀딩스	1.729085	99.942281
15 032640	LG유플러스	1.779967	99.939921
16 030200	KT	1.780783	99.939883
17 085620	미래에셋생명	1.782491	99.939804
18 086790	하나금융지주	1.858610	99.936274
19 003070	코오롱글로벌	1.858922	99.936259
20 205470	휴마시스	1.900331	99.934339
21 001740	SK네트웍스	2.216611	99.919671
22 017670	SK텔레콤	2.291059	99.916218
23 006800	미래에셋증권	2.344916	99.913720
24 011780	금호석유	2.444376	99.909107
25 005880	대한해운	2.524615	99.905386
26 003490	대한항공	2.547746	99.904313
27 004170	신세계	2.612402	99.901315
28 383220	F&F	2.714107	99.896598
29 055550	신한지주	2.744568	99.895185
30 034310	NICE	2.815003	99.891919

## 1) PCR 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
PCR 0.1이상, 낮은 순 30개 종목 선정

```
# PCR 종목 선정
df_factor = pcr_list.copy()
factor_list = ['PCR']
result = stock_select(df_factor, 0.2, 'KOSDAQ', 30, factor_list) # PCR 시가총액 상위 20%, KOSDAQ 종목 중 30개 선정
result
```

code	name	PCR	score
1 950130	엑세스바이오	0.933140	100.000000
2 003380	하림지주	1.188721	99.980509
3 205470	휴마시스	1.900331	99.926239
4 035760	CJ ENM	3.335476	99.816791
5 100090	삼강엔엔티	3.826490	99.779345
6 035600	KG이니시스	3.963334	99.768909
7 025980	아난티	4.418105	99.734227
8 096530	씨엔	4.647169	99.716758
9 091700	파트론	4.736752	99.709926
10 108320	LX세미콘	5.805411	99.628427
11 074600	원익QnC	6.088883	99.606809
12 215200	메가스터디교육	6.433266	99.580545
13 215000	골프존	6.782689	99.553897
14 067310	하나마이크론	6.887186	99.545928
15 049070	인탑스	6.971379	99.539507
16 033290	코월패션	7.753475	99.479863
17 319660	피에스케이	7.927950	99.466557
18 036930	주성엔지니어링	7.930287	99.466378
19 183300	코미코	8.165231	99.448461
20 089970	에이피티씨	8.385446	99.431667
21 095610	테스	8.524408	99.421069
22 046890	서울반도체	8.824195	99.398207
23 048260	오스템임플란트	9.781028	99.325236
24 222800	실텍	10.050211	99.304707
25 033640	네페스	10.720390	99.253597
26 090460	비에이치	10.916000	99.238680
27 069080	월천	11.236340	99.214250
28 036540	SFA반도체	11.474806	99.196064
29 060150	인선이엔티	12.517389	99.116553
30 067160	아프리카TV	12.594394	99.110681

## 2) PCR 기준 종목 선정 결과

시가총액 상위 20% KOSDAQ 종목에 대해  
PCR 0.1이상, 낮은 순 30개 종목 선정

## 03. 가치주 종목 추출 5) 가치지표 결합하기 (PER+PBR)



# 03. 가치주 종목 추출 5) 가치지표 결합하기 (PER+PBR)

```
# PER, PBR 결합 종목 선정
factor_list = ['PER', 'PBR']
df_factor = pd.merge(per_list, pbr_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # PER+PBR, 시가총액 상위 20%, 전체 종목 중 30개 선정
result
```

code	name	PER	PBR	score
1 001230	동국제강	1.989159	0.147859	99.999796
2 001120	LX인터넷서널	2.005910	0.146550	99.999791
3 011200	HMM	1.714468	0.388620	99.999512
4 005880	대한해운	2.689223	0.156941	99.999454
5 298020	효성티엔씨	2.156513	0.327424	99.999411
6 011780	금호석유	2.520174	0.244289	99.999383
7 036460	한국가스공사	3.083438	0.103828	99.999361
8 042670	현대두산인프라코어	2.882788	0.184595	99.999317
9 018670	SK가스	3.161570	0.125308	99.999288
10 005490	POSCO홀딩스	3.207656	0.118080	99.999279
11 950130	엑세스바이오	1.850054	0.510067	99.999240
12 010060	OCI	3.354670	0.212682	99.999048
13 004800	효성	3.655593	0.158142	99.999001
14 004000	롯데정밀화학	3.556533	0.252095	99.998887
15 003550	LG	4.044242	0.132706	99.998864
16 007070	GS리테일	3.844619	0.188247	99.998862
17 003070	코오롱글로벌	3.844197	0.231416	99.998788
18 003530	한화투자증권	4.434461	0.160119	99.998635
19 139130	DGB금융지주	3.693192	0.374404	99.998613
20 192400	쿠쿠홀딩스	4.422754	0.181465	99.998604
21 205470	휴마시스	2.058360	0.827538	99.998599
22 210980	SKC(엔디)	4.384572	0.270029	99.998470
23 009160	SIMPAC	4.508442	0.253539	99.998440
24 383220	F&F	3.476459	0.592136	99.998341
25 006040	동원산업	5.147754	0.156952	99.998307
26 014830	유니드	4.861785	0.236200	99.998305
27 034120	SBS	4.765497	0.272380	99.998288
28 016380	KG동부제철	4.870247	0.248427	99.998280
29 004170	신세계	5.604642	0.112386	99.998171
30 004430	송원산업	5.131220	0.255916	99.998146

## 1) PER+PBR 결합 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
PER 1 이상, PBR 0.1이상,  
낮은 순 30개 종목 선정

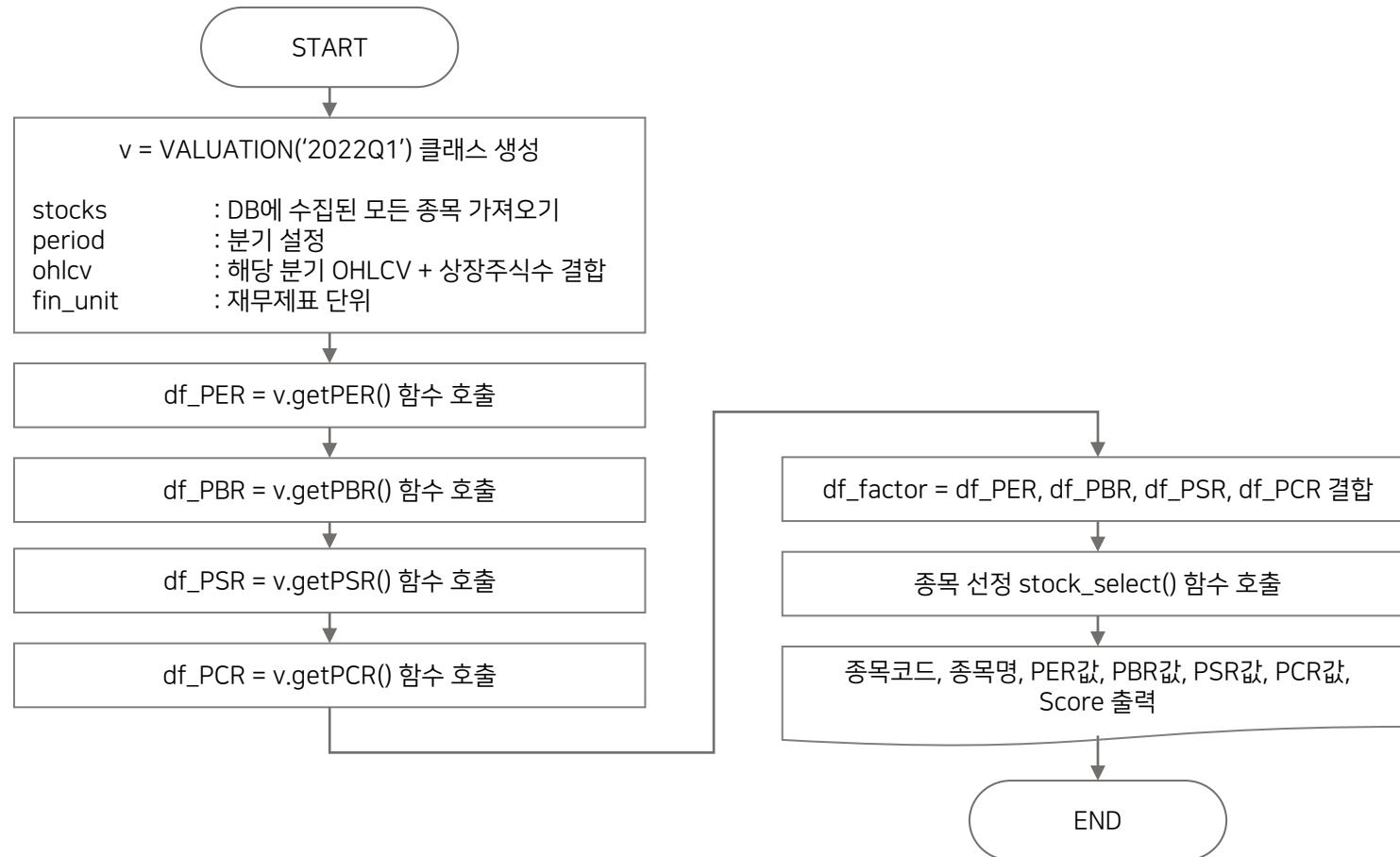
```
# PER, PBR 결합 종목 선정
factor_list = ['PER', 'PBR']
df_factor = pd.merge(per_list, pbr_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])
result = stock_select(df_factor, 0.2, 'KOSPI', 30, factor_list) # PER+PBR, 시가총액 상위 20%, KOSPI 종목 중 30개 선정
result
```

code	name	PER	PBR	score
1 001230	동국제강	1.989159	0.147859	99.999796
2 001120	LX인터넷서널	2.005910	0.146550	99.999791
3 011200	HMM	1.714468	0.388620	99.999512
4 005880	대한해운	2.689223	0.156941	99.999454
5 298020	효성티엔씨	2.156513	0.327424	99.999411
6 011780	금호석유	2.520174	0.244289	99.999383
7 036460	한국가스공사	3.083438	0.103828	99.999361
8 042670	현대두산인프라코어	2.882788	0.184595	99.999317
9 018670	SK가스	3.161570	0.125308	99.999288
10 005490	POSCO홀딩스	3.207656	0.118080	99.999279
11 010060	OCI	3.354670	0.212682	99.999048
12 004800	효성	3.655593	0.158142	99.999001
13 004000	롯데정밀화학	3.556533	0.252095	99.998887
14 003550	LG	4.044242	0.132706	99.998864
15 007070	GS리테일	3.844619	0.188247	99.998862
16 003070	코오롱글로벌	3.844197	0.231416	99.998788
17 003530	한화투자증권	4.434461	0.160119	99.998635
18 139130	DGB금융지주	3.693192	0.374404	99.998613
19 192400	쿠쿠홀딩스	4.422754	0.181465	99.998604
20 210980	SKC(엔디)	4.384572	0.270029	99.998470
21 009160	SIMPAC	4.508442	0.253539	99.998440
22 383220	F&F	3.476459	0.592136	99.998341
23 006040	동원산업	5.147754	0.156952	99.998307
24 014830	유니드	4.861785	0.236200	99.998305
25 034120	SBS	4.765497	0.272380	99.998288
26 016380	KG동부제철	4.870247	0.248427	99.998280
27 004170	신세계	5.604642	0.112386	99.998171
28 004430	송원산업	5.131220	0.255916	99.998146
29 284740	쿠쿠홈시스	5.025914	0.292738	99.998132
30 030200	KT	5.868919	0.143582	99.997994

## 2) PER+PBR 결합 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
PER 1이상, PBR 0.1이상,  
낮은 순 30개 종목 선정

## 03. 가치주 종목 추출 6) 4대장 결합하기 (PER+PBR+PSR+PCR)



# 03. 가치주 종목 추출 6) 4대장 결합하기 (PER+PBR+PSR+PCR)

```
# 4대장 결합 종목 선정
factor_list = ['PER', 'PBR', 'PSR', 'PCR']
# 결합하기
df_factor = pd.merge(per_list, pbr_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])
df_factor = pd.merge(df_factor, psr_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])
df_factor = pd.merge(df_factor, pcr_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])

result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # PER&PBR, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

	code	name	PER	PBR	PSR	PCR	score
1	003070	코오롱글로벌	3.844197	0.231416	0.116367	1.858922	99.985723
2	001740	SK네트웍스	17.357580	0.123329	0.108819	2.216611	99.981458
3	034220	LG디스플레이	6.587205	0.127632	0.250753	1.346730	99.938731
4	001230	동국제강	1.988159	0.147859	0.200418	4.020328	99.928401
5	049770	동원F&B	11.508513	0.218263	0.192383	4.445341	99.892003
6	034310	NICE	7.328878	0.123571	0.252046	2.815003	99.921031
7	298020	효성티엔씨	2.156513	0.327424	0.215951	4.484545	99.916770
8	097950	CJ제일제당	6.036377	0.129005	0.205327	5.739855	99.905618
9	298040	효성중공업	11.413018	0.139740	0.183432	7.211524	99.895823
10	007070	GS리테일	3.844619	0.188247	0.289239	4.006598	99.893472
11	006040	동원산업	5.147754	0.156952	0.316190	3.390002	99.889841
12	047040	대우건설	5.696973	0.228686	0.323416	3.351544	99.887281
13	030200	KT	5.868919	0.143582	0.370699	1.780783	99.887100
14	005490	POSCO홀딩스	3.207656	0.118080	0.313054	4.483349	99.878870
15	004170	신세계	5.604642	0.112386	0.373398	2.612402	99.876492
16	280360	롯데제과	44.343013	0.142406	0.351794	3.915762	99.860737
17	032640	LG유플러스	8.811074	0.192898	0.441520	1.779967	99.858776
18	011210	현대위아	37.321122	0.118881	0.230698	8.609429	99.855164
19	086280	현대글로비스	7.634720	0.317958	0.313747	7.352761	99.844119
20	204320	만도	14.787376	0.294903	0.372913	5.547163	99.840349
21	006360	GS건설	9.313987	0.203512	0.422520	4.111074	99.839022
22	012450	한화에어로스페이스	10.731922	0.170289	0.401603	4.913660	99.837156
23	005090	SGC에너지	5.583782	0.256736	0.350369	6.844778	99.836267
24	000270	기아	6.304702	0.220799	0.418729	4.654571	99.834884
25	005850	에스엘	14.322274	0.194672	0.384163	6.096723	99.829785
26	011170	롯데케미칼	7.149728	0.117520	0.367516	6.983412	99.827730
27	000720	현대건설	9.822174	0.143751	0.295325	9.377633	99.827597
28	006650	대한유화	13.306161	0.134723	0.411006	5.660755	99.824666
29	282330	BGF리테일	19.117911	0.980324	0.432584	5.247425	99.818188
30	011780	금호석유	2.520174	0.244289	0.533227	2.444376	99.816749

## 1) 4대장 결합 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
PER 1 이상, PBR/PSR/PCR 0.1 이상  
낮은 순 30개 종목 선정

```
# 4대장 결합 종목 선정
factor_list = ['PER', 'PBR', 'PSR', 'PCR']
# 결합하기
df_factor = pd.merge(per_list, pbr_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])
df_factor = pd.merge(df_factor, psr_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])
df_factor = pd.merge(df_factor, pcr_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])

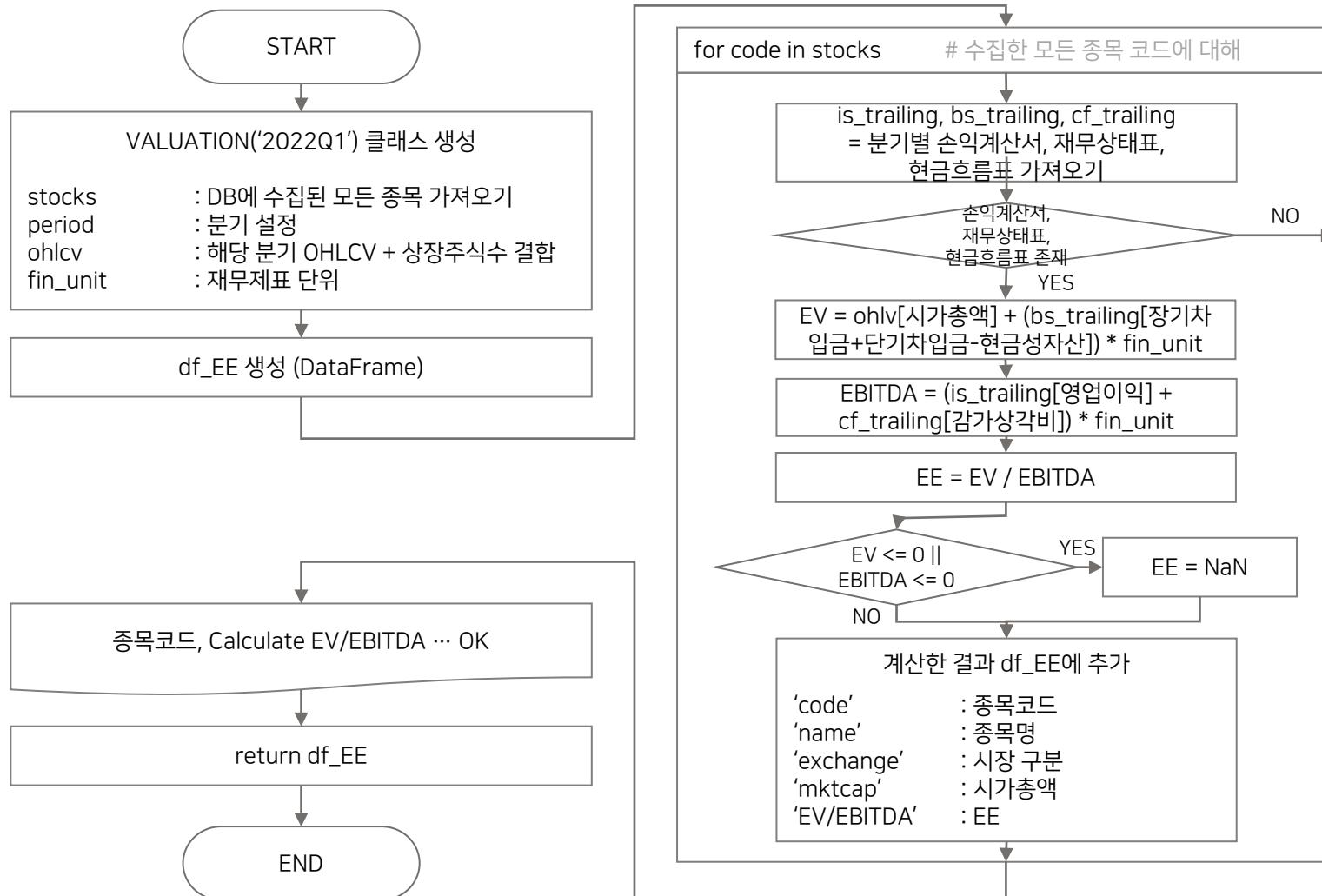
result = stock_select(df_factor, 0.2, 'KOSPI', 30, factor_list) # PER&PBR, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

	code	name	PER	PBR	PSR	PCR	score
1	003070	코오롱글로벌	3.844197	0.231416	0.116367	1.858922	99.987166
2	001740	SK네트웍스	17.357580	0.123329	0.108819	2.216611	99.986252
3	001230	동국제강	1.988159	0.147859	0.200418	4.020328	99.982514
4	049770	동원F&B	11.508513	0.218263	0.192383	4.445341	99.892003
5	034220	LG디스플레이	6.587205	0.127632	0.250753	1.346730	99.880496
6	298020	효성티엔씨	2.156513	0.327424	0.215951	4.484545	99.873984
7	097950	CJ제일제당	6.036377	0.129005	0.205327	5.739855	99.867547
8	298040	효성중공업	11.413018	0.139740	0.183432	7.211524	99.867471
9	034310	NICE	7.328878	0.123571	0.252046	2.815003	99.862218
10	007070	GS리테일	3.844619	0.188247	0.289239	4.006598	99.818141
11	011210	현대위아	37.321122	0.118881	0.230698	8.609429	99.805819
12	006040	동원산업	5.147754	0.156952	0.316190	3.390002	99.802542
13	047040	대우건설	5.696973	0.228686	0.323416	3.351544	99.796773
14	005490	POSCO홀딩스	3.207656	0.118080	0.313054	4.483349	99.792961
15	030200	KT	5.868919	0.143582	0.370699	1.780783	99.775597
16	004170	신세계	5.604642	0.112386	0.373398	2.612402	99.763789
17	086280	현대글로비스	7.634720	0.317958	0.313747	7.352761	99.757897
18	280360	롯데제과	44.343013	0.142406	0.351794	3.915762	99.757625
19	000720	현대건설	9.822174	0.143751	0.295325	9.377633	99.749410
20	000120	CJ대한통운	17.258554	0.185856	0.256708	12.477081	99.743890
21	005090	SGC에너지	5.583782	0.256736	0.350369	6.844778	99.733781
22	204320	만도	14.787376	0.294903	0.372913	5.547163	99.731555
23	011170	롯데케미칼	7.149728	0.117520	0.367516	6.983412	99.711629
24	032640	LG유플러스	8.811074	0.192898	0.441520	1.779967	99.715823
25	012450	한화에어로스페이스	10.731922	0.170289	0.401603	4.913660	99.712322
26	005850	에스엘	14.322274	0.194672	0.384163	6.096723	99.712294
27	006650	LG전자	11.974276	0.243496	0.252748	15.918046	99.711454
28	006360	GS건설	9.313987	0.203512	0.422520	4.111074	99.704501
29	000270	기아	6.304702	0.220799	0.418729	4.654571	99.702401
30	006650	대한유화	13.306161	0.134723	0.411006	5.660755	99.695255

## 2) 4대장 결합 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
PER 1 이상, PBR/PSR/PCR 0.1 이상  
낮은 순 30개 종목 선정

## 03. 가치주 종목 추출 7) 실적대비 기업 가치 (EV/EBITDA)



## 03. 가치주 종목 추출 7) 실적대비 기업 가치 (EV/EBITDA)

```
# 실적 대비 기업 가치 (EV/EBITDA)
def getEVEBITDA(self):
    # EV/EBITDA 계산 후 저장할 DataFrame 생성
    df_EE = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 손익계산서 & 재무상태표 & 현금흐름표 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)
        cf_trailing = CASHFLOW().getTrailingCashflow(code, self.period)

        # 재무제표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or bs_trailing is None or cf_trailing is None or code not in self.ohlcvs.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. EV=(시가총액 + 차입금 - 현금성자산) 계산
        # 시가총액: ohlcvs > mktcap
        # 차입금: balances > longdebt + curdebt (장기차입금 + 단기차입금)
        # 현금성자산: balances > cash
        EV = float(self.ohlcvs.loc[code]['mktcap'] + (bs_trailing['longdebt'] + bs_trailing['curdebt']) - bs_trailing['cash'])*self.fin_unit

        # 3. EBITDA=(영업이익 + 감가상각비) 계산
        # 영업이익: incomes > opr
        # 감가상각비: cashflow > cfo4
        EBITDA = float(is_trailing['opr'] + cf_trailing['cfo4']*self.fin_unit)

        # 4. EV/EBITDA 계산
        if EV <= 0 or EBITDA <= 0: # EV 음수 or EBITDA 음수인 경우
            EE = np.Nan
        else: # 계산 가능한 경우
            EE = float(EV / EBITDA)

        # 5. 계산한 결과 DataFrame에 추가
        df_EE = df_EE.append({
            'code': code,
            'name': name,
            'exchange': exc,
            'period': self.period,
            'mktcap': self.ohlcvs.loc[code]['mktcap'],
            'EV/EBITDA': EE }, ignore_index=True)

    print(code, ": Calculate EV/EBITDA ... OK")

return df_EE
```

## 03. 가치주 종목 추출 7) 실적대비 기업 가치 (EV/EBITDA)

getEVEBITDA() 실행 결과: 모든 종목에 대해 EV/EBITDA 값 계산

```
# EV/EBITDA 구하기
EE_list = v.getEVEBITDA()
EE_list
000000 : Calculate EV/EBITDA ... OK
011810 : Calculate EV/EBITDA ... OK
077970 : Calculate EV/EBITDA ... OK
071970 : Calculate EV/EBITDA ... OK
002820 : Calculate EV/EBITDA ... OK
Empty DB Incomes
Empty DB Balances
Empty DB Cashflow
289080 : Not enough data to calculate ...
084870 : Not enough data to calculate ...
002710 : Calculate EV/EBITDA ... OK
089230 : Calculate EV/EBITDA ... OK
161570 : Calculate EV/EBITDA ... OK
192410 : Calculate EV/EBITDA ... OK
032540 : Calculate EV/EBITDA ... OK
048770 : Calculate EV/EBITDA ... OK
246690 : Calculate EV/EBITDA ... OK
317240 : Not enough data to calculate ...
002900 : Calculate EV/EBITDA ... OK
038340 : Calculate EV/EBITDA ... OK
```

	code	name	exchange	period	mktcap	EV/EBITDA
0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	8.879163
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	24.892799
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	91.825158
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	27.008583
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	NaN
...	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	42.306290
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	NaN
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	42.677096
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	26.261981
2181	238490	힘스	KOSDAQ	2022/03	1.032807e+11	30.261006

# 03. 가치주 종목 추출

## 7) 실적대비 기업 가치 (EV/EBITDA)

```
# EV/EBITDA 종목 선정
df_factor = EE_list.copy()
factor_list = ['EV/EBITDA']
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # EV/EBITDA, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	EV/EBITDA	score
1 030200	KT	0.115601	100.0
2 001680	대상	1.892233	100.0
3 000270	기아	2.253027	100.0
4 015760	한국전력	2.396871	100.0
5 002790	아모레G	2.658021	100.0
6 280360	롯데제과	2.776622	100.0
7 017670	SK텔레콤	3.299110	100.0
8 006650	대한유화	3.738337	100.0
9 066570	LG전자	4.325422	100.0
10 001740	SK네트웍스	4.331573	100.0
11 089860	롯데렌탈	4.393584	100.0
12 004170	신세계	4.548300	100.0
13 032640	LG유플러스	4.592411	100.0
14 161390	한국타이어앤테크놀로지	5.051833	100.0
15 079160	CJ CGV	5.212454	100.0
16 034310	NICE	5.269102	100.0
17 007070	GS리테일	5.287920	100.0
18 030520	한글과컴퓨터	5.594059	100.0
19 034220	LG디스플레이	5.886021	100.0
20 139480	이마트	6.605574	100.0
21 204320	만도	6.820125	100.0
22 108670	LX하우시스	6.845710	100.0
23 023530	롯데쇼핑	6.947745	100.0
24 004990	롯데지주	6.978453	100.0
25 282330	BGF리테일	7.592365	100.0
26 020560	아시아나항공	7.819109	100.0
27 034730	SK	7.867272	100.0
28 001040	CJ	8.355163	100.0
29 011200	HMM	8.731570	100.0
30 000660	SK하이닉스	8.971889	100.0
	SK하이닉스	8.971889	100.0

### 1) EV/EBITDA 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
EV/EBITDA 0이상,  
낮은 순 30개 종목 선정

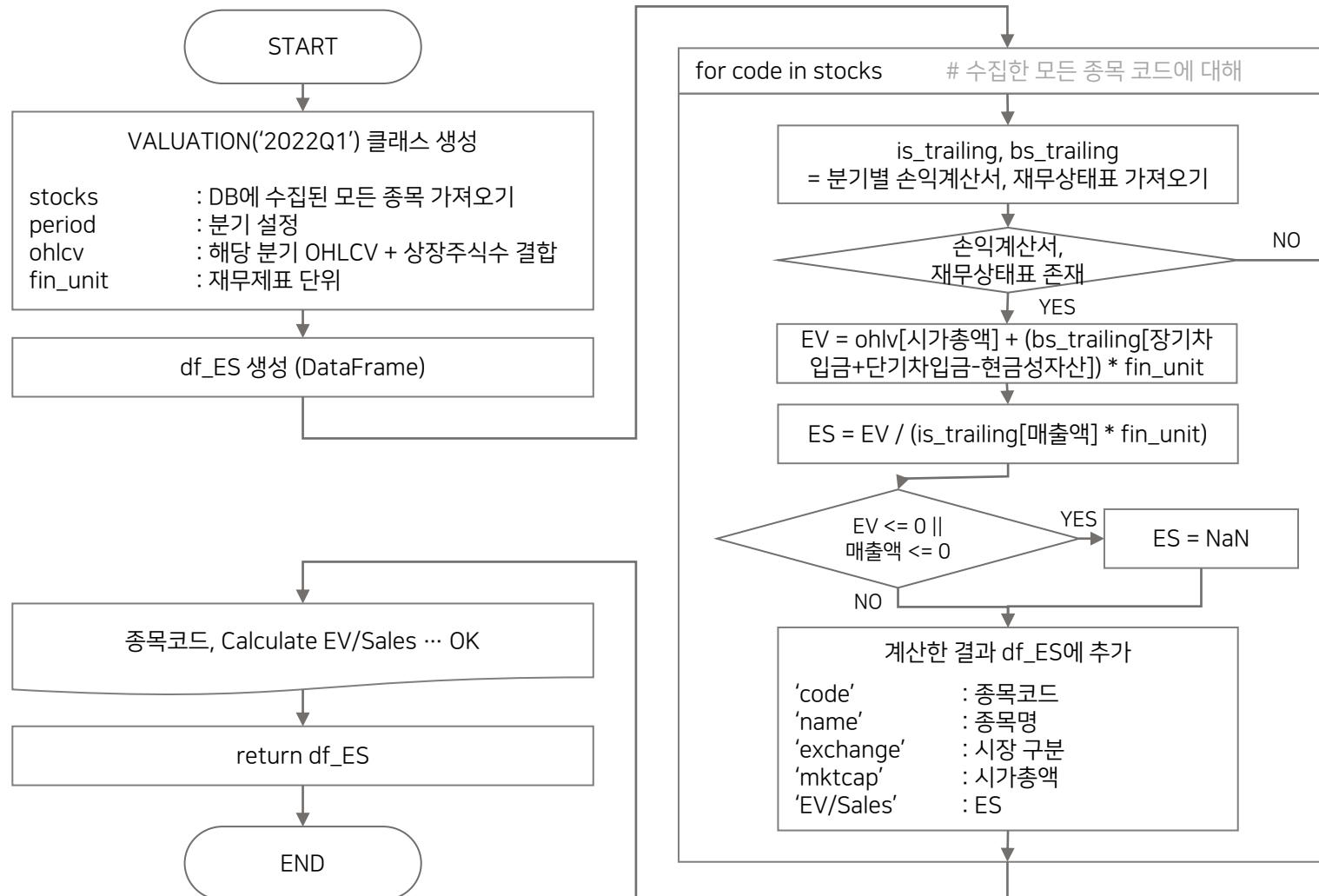
```
# EV/EBITDA 종목 선정
df_factor = EE_list.copy()
factor_list = ['EV/EBITDA']
result = stock_select(df_factor, 0.2, 'KOSPI', 30, factor_list) # EV/EBITDA, 시가총액 상위 20%, KOSPI 종목 중 30개 선택
result
```

code	name	EV/EBITDA	score
1 030200	KT	0.115601	100.0
2 001680	대상	1.892233	100.0
3 000270	기아	2.253027	100.0
4 015760	한국전력	2.396871	100.0
5 002790	아모레G	2.658021	100.0
6 280360	롯데제과	2.776622	100.0
7 017670	SK텔레콤	3.299110	100.0
8 006650	대한유화	3.738337	100.0
9 066570	LG전자	4.325422	100.0
10 001740	SK네트웍스	4.331573	100.0
11 089860	롯데렌탈	4.393584	100.0
12 004170	신세계	4.548300	100.0
13 032640	LG유플러스	4.592411	100.0
14 161390	한국타이어앤테크놀로지	5.051833	100.0
15 079160	CJ CGV	5.212454	100.0
16 034310	NICE	5.269102	100.0
17 007070	GS리테일	5.287920	100.0
18 034220	LG디스플레이	5.886021	100.0
19 139480	이마트	6.605574	100.0
20 204320	만도	6.820125	100.0
21 108670	LX하우시스	6.845710	100.0
22 023530	롯데쇼핑	6.947745	100.0
23 004990	롯데지주	6.978453	100.0
24 282330	BGF리테일	7.592365	100.0
25 020560	아시아나항공	7.819109	100.0
26 034730	SK	7.867272	100.0
27 001040	CJ	8.355163	100.0
28 011200	HMM	8.731570	100.0
29 000660	SK하이닉스	8.971889	100.0
30 003490	대한항공	9.360930	100.0

### 2) EV/EBITDA 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
EV/EBITDA 0이상,  
낮은 순 30개 종목 선정

## 03. 가치주 종목 추출 8) 실적대비 기업 가치 (EV/Sales)



## 03. 가치주 종목 추출 8) 실적대비 기업 가치 (EV/Sales)

```
# 실적 대비 기업가치 (EV/Sales)
def getEVSales(self):
    # EV/Sales 계산 후 저장할 DataFrame 생성
    df_ES = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 본기별 손익계산서 & 재무상태표 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)

        # 재무제표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or bs_trailing is None or code not in self.ohlcvs.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. EV=(시가총액 + 차입금 - 현금성자산) 계산
        # 시가총액: stocks > mktcap
        # 차입금: balances > longdebt (장기차입금 + 단기차입금)
        # 현금성자산: balances > cash
        EV = float(self.ohlcvs.loc[code]['mktcap'] + (bs_trailing['longdebt'] + bs_trailing['curdebt']) - bs_trailing['cash'])*self.fin_unit

        # 3. EV/Sales(매출액) 계산
        # 매출액: incomes > rev
        ES = float(EV / (is_trailing['rev']*self.fin_unit))

        # 4. 결측치 처리
        if EV <= 0 or float(is_trailing['rev']) <= 0: # EV 음수 or 매출액 음수인 경우
            ES = np.NaN

        # 5. 계산한 결과 DataFrame에 추가
        df_ES = df_ES.append({
            'code':code,
            'name':name,
            'exchange': exc,
            'period': self.period,
            'mktcap': self.ohlcvs.loc[code]['mktcap'],
            'EV/Sales': ES }, ignore_index=True)

        print(code, ": Calculate EV/Sales ... OK")

    return df_ES
```

## 03. 가치주 종목 추출 8) 실적대비 기업 가치 (EV/Sales)

getEVSales() 실행 결과: 모든 종목에 대해 EV/Sales 값 계산

# EV/Sales 구하기

```
ES_list = v.getEVSales()
```

```
ES_list
```

```
123750 : Calculate EV/Sales ... OK
085810 : Calculate EV/Sales ... OK
117670 : Calculate EV/Sales ... OK
291650 : Calculate EV/Sales ... OK
293780 : Calculate EV/Sales ... OK
Empty DB Incomes
Empty DB Balances
267810 : Not enough data to calculate ...
018250 : Calculate EV/Sales ... OK
161000 : Calculate EV/Sales ... OK
196300 : Calculate EV/Sales ... OK
310200 : Calculate EV/Sales ... OK
179530 : Not enough data to calculate ...
900100 : Calculate EV/Sales ... OK
205500 : Calculate EV/Sales ... OK
052790 : Calculate EV/Sales ... OK
290740 : Calculate EV/Sales ... OK
238090 : Calculate EV/Sales ... OK
092600 : Calculate EV/Sales ... OK
129890 : Calculate EV/Sales ... OK
```

# EV/Sales 구하기

```
ES_list = v.getEVSales()
```

```
ES_list
```

	code	name	exchange	period	mktcap	EV/Sales
0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	1.242540
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	1.703735
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	11.365733
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	0.416820
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	Nan
...	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	2.971939
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	Nan
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	5.871036
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	0.630853
2181	238490	힘스	KOSDAQ	2022/03	1.032807e+11	0.881227

## 03. 가치주 종목 추출 8) 실적대비 기업 가치 (EV/Sales)

```
# EV/Sales 종목 선정
df_factor = ES_list.copy()
factor_list = ['EV/Sales']
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # EV/Sales, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	EV/Sales	score
1 030200	KT	0.013974	100.000000
2 000270	기아	0.054150	99.999619
3 001680	대상	0.056798	99.999594
4 088350	한화생명	0.057526	99.999588
5 000370	한화손해보험	0.066686	99.999501
6 001120	LX인터넷서널	0.095009	99.999233
7 003690	코리안리	0.107240	99.999117
8 085620	미래에셋생명	0.129194	99.998909
9 066570	LG전자	0.132867	99.998874
10 001450	현대해상	0.132876	99.998874
11 280360	롯데제과	0.135714	99.998847
12 049070	인탑스	0.159845	99.998618
13 006360	GS건설	0.172043	99.998503
14 005830	DB손해보험	0.202121	99.998218
15 030520	한글과컴퓨터	0.205659	99.998185
16 032830	삼성생명	0.207201	99.998170
17 205470	휴마시스	0.211803	99.998126
18 002790	아모레G	0.211848	99.998126
19 006650	대한유화	0.225872	99.997993
20 047040	대우건설	0.231976	99.997935
21 030000	제일기획	0.239663	99.997862
22 005440	현대그린푸드	0.254438	99.997723
23 028050	삼성엔지니어링	0.264864	99.997624
24 047050	포스코인터넷서널	0.266314	99.997610
25 034310	NICE	0.276230	99.997516
26 000880	한화	0.285906	99.997425
27 139480	이마트	0.290091	99.997385
28 108670	LX하우시스	0.304800	99.997246
29 042660	대우조선해양	0.305414	99.997240
30 003070	코오롱글로벌	0.306098	99.997233

### 1) EV/Sales 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
EV/Sales 0이상,  
낮은 순 30개 종목 선정

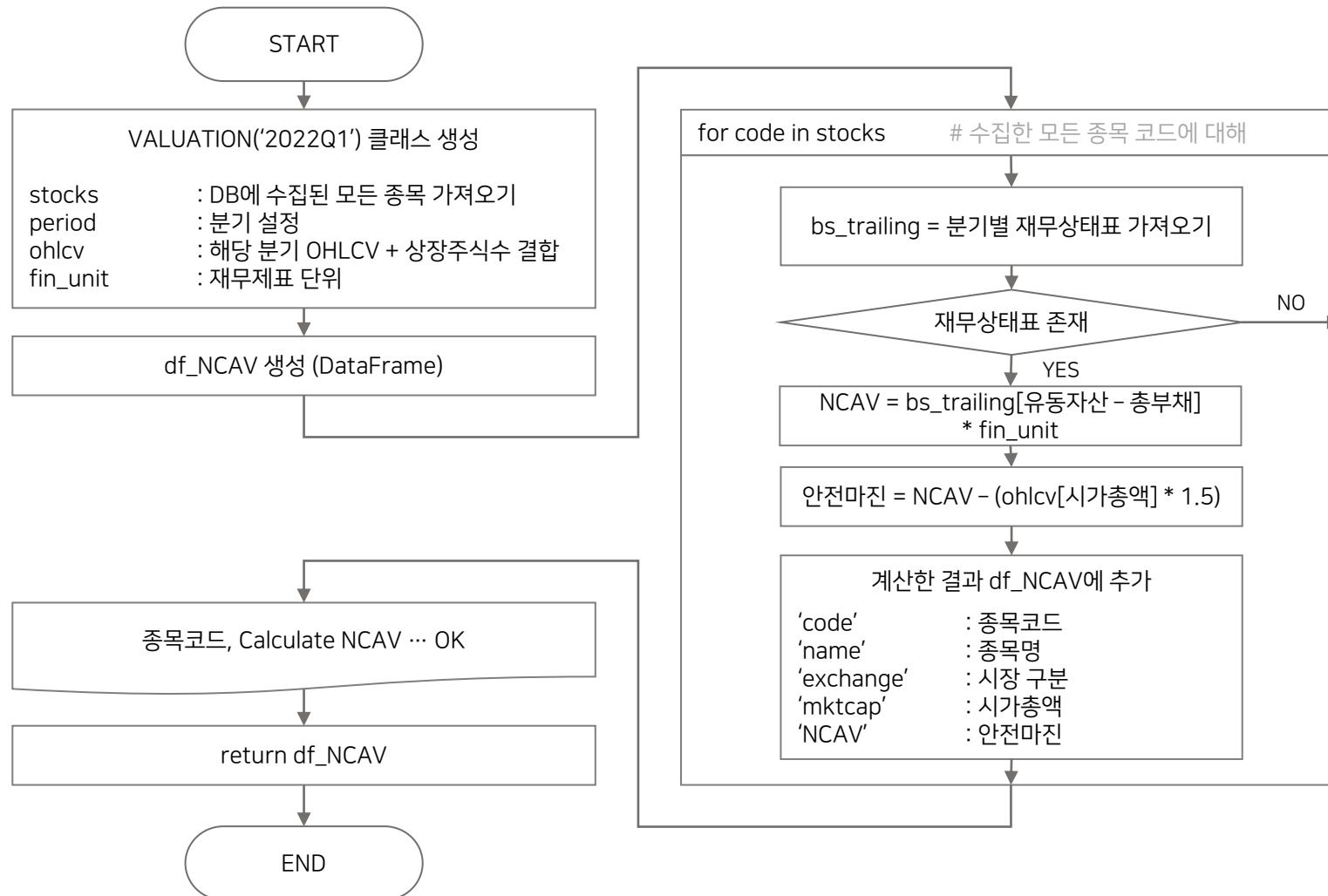
```
# EV/Sales 종목 선정
df_factor = ES_list.copy()
factor_list = ['EV/Sales']
result = stock_select(df_factor, 0.2, 'KOSPI', 30, factor_list) # EV/Sales, 시가총액 상위 20%, KOSPI 종목 중 30개 선택
result
```

code	name	EV/Sales	score
1 030200	KT	0.013974	100.000000
2 000270	기아	0.054150	99.988529
3 001680	대상	0.056798	99.987773
4 088350	한화생명	0.057526	99.987565
5 000370	한화손해보험	0.066686	99.984950
6 001120	LX인터넷서널	0.095009	99.976863
7 003690	코리안리	0.107240	99.973370
8 085620	미래에셋생명	0.129194	99.967102
9 066570	LG전자	0.132867	99.966053
10 001450	현대해상	0.132876	99.966051
11 280360	롯데제과	0.135714	99.965240
12 006360	GS건설	0.172043	99.954867
13 005830	DB손해보험	0.202121	99.946280
14 032830	삼성생명	0.207201	99.944829
15 002790	아모레G	0.211848	99.943502
16 006650	대한유화	0.225872	99.939498
17 047040	대우건설	0.231976	99.937755
18 030000	제일기획	0.239663	99.935560
19 005440	현대그린푸드	0.254438	99.931342
20 028050	삼성엔지니어링	0.264864	99.928365
21 047050	포스코인터넷서널	0.266314	99.927951
22 034310	NICE	0.276230	99.925119
23 000880	한화	0.285906	99.922357
24 139480	이마트	0.290091	99.921162
25 108670	LX하우시스	0.304800	99.916962
26 042660	대우조선해양	0.305414	99.916787
27 003070	코오롱글로벌	0.306098	99.916591
28 012330	현대모비스	0.308171	99.916000
29 204320	만도	0.308665	99.915858
30 007070	GS리테일	0.320124	99.910017

### 2) EV/Sales 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
EV/Sales 0이상,  
낮은 순 30개 종목 선정

## 03. 가치주 종목 추출 9) 안전마진이 있는 그레이엄의 NCAV



## 03. 가치주 종목 추출 9) 안전마진이 있는 그레이엄의 NCAV

```
# 안전마진이 있는 그레이엄의 NCAV(청산가치) 투자법
def getNCAV(self):
    # NCAV 계산 후 저장할 DataFrame 생성
    df_NCAV = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 재무상태표 가져오기
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)

        # 재무제표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlc 데이터가 존재하지 않는 종목인 경우
        if bs_trailing is None or code not in self.ohlcv.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. NCAV (유동자산 - 종부채) 계산
        # 유동자산: balances > curasset
        # 종 부채: balances > liab
        NCAV = float(bs_trailing['curasset'] - bs_trailing['liab']) * self.fin_unit

        # 3. 안전마진(NCAV - (시가총액*1.5)) 계산
        # 시가총액: ohlc > mktcap
        SafetyMargin = NCAV - (self.ohlcv.loc[code]['mktcap'] * 1.5)

        # 4. 계산한 결과 DataFrame에 추가
        df_NCAV = df_NCAV.append({
            'code':code,
            'name':name,
            'exchange': exc,
            'period': self.period,
            'mktcap': self.ohlcv.loc[code]['mktcap'],
            'NCAV': SafetyMargin }, ignore_index=True)

    print(code, ": Calculate NCAV ... OK")

    return df_NCAV
```

## 03. 가치주 종목 추출 9) 안전마진이 있는 그레이엄의 NCAV

getNCAV() 실행 결과: 모든 종목에 대해 안전마진 값 계산

### ### NCAV

# NCAV 구하기

```
NCAV_list = v.getNCAV()  
NCAV_list
```

```
241590 : Not enough data to calculate ...  
006060 : Calculate NCAV ... OK  
013520 : Calculate NCAV ... OK  
010690 : Calculate NCAV ... OK  
126640 : Calculate NCAV ... OK  
133820 : Calculate NCAV ... OK  
061250 : Calculate NCAV ... OK  
010660 : Calculate NCAV ... OK  
000850 : Calculate NCAV ... OK  
016580 : Calculate NCAV ... OK  
032560 : Calculate NCAV ... OK  
004800 : Calculate NCAV ... OK  
094280 : Calculate NCAV ... OK  
097870 : Not enough data to calculate ...  
298040 : Calculate NCAV ... OK  
298050 : Calculate NCAV ... OK  
298020 : Calculate NCAV ... OK  
298000 : Calculate NCAV ... OK  
093370 : Calculate NCAV ... OK  
050090 : Calculate NCAV ... OK
```

# NCAV 구하기

```
NCAV_list = v.getNCAV()  
NCAV_list
```

	code	name	exchange	period	mktcap	NCAV
0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	-3.334038e+12
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	-8.340486e+12
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	-6.059542e+11
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	-3.852046e+11
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	-2.754479e+11
...	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	-2.138500e+11
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	-1.337159e+13
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	-1.417359e+12
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	-1.198883e+11
2181	238490	힘스	KOSDAQ	2022/03	1.032807e+11	-5.985407e+10

## 03. 가치주 종목 추출 9) 안전마진이 있는 그레이엄의 NCAV

```
# NCAV 종목 선정
df_factor = NCAV_list.copy()

factor_list = ['NCAV']
result = stock_select2(df_factor, 0.3, 'ALL', 30, factor_list) # NCAV, 시가총액 상위 30%, 전체 종목 중 30개 선택
result
```

code	name	NCAV	score
1 006800	미래에셋증권	7.244344e+13	100.000000
2 016360	삼성증권	6.727760e+13	92.869001
3 039490	키움증권	6.033157e+13	83.280611
4 005940	NH투자증권	3.210410e+13	44.315034
5 003540	대신증권	2.444576e+13	33.743354
6 003470	유안타증권	1.994691e+13	27.533079
7 008560	메리츠증권	1.148073e+13	15.846254
8 000720	현대건설	9.469998e+12	13.070609
9 030610	교보증권	5.032844e+12	6.945500
10 078020	이베스트투자증권	4.562230e+12	6.295859
11 001510	SK증권	4.203789e+12	5.801062
12 294870	HDC현대산업개발	3.740799e+12	5.161944
13 003240	태광산업	2.949000e+12	4.068934
14 003530	한화투자증권	2.938918e+12	4.055016
15 001500	현대차증권	2.218751e+12	3.060888
16 018260	삼성에스디에스	1.926802e+12	2.657878
17 161390	한국타이어앤티크놀로지	1.748306e+12	2.411480
18 285130	SK케미칼	1.619641e+12	2.233869
19 012630	HDC	1.405091e+12	1.937701
20 267270	현대건설기계	1.249416e+12	1.722805
21 013120	동원개발	1.144053e+12	1.577361
22 002790	아모레G	1.011479e+12	1.394354
23 004490	세방전지	9.122860e+11	1.257426
24 020000	한섬	9.106465e+11	1.255163
25 003060	에이프로젝트	8.372299e+11	1.153818
26 003030	세아제강지주	7.942217e+11	1.094449
27 081000	일진다이아	7.421612e+11	1.022584
28 126560	현대유처넷	7.300897e+11	1.005920
29 226320	잇즈한글	6.130455e+11	0.844351
30 000580	에스엘	4.816053e+11	0.662909

### 1) NCAV 기준 종목 선정 결과

시가총액 상위 30% 모든 종목에 대해  
NCAV 0이상,  
높은 순 30개 종목 선정

```
# NCAV 종목 선정
df_factor = NCAV_list.copy()

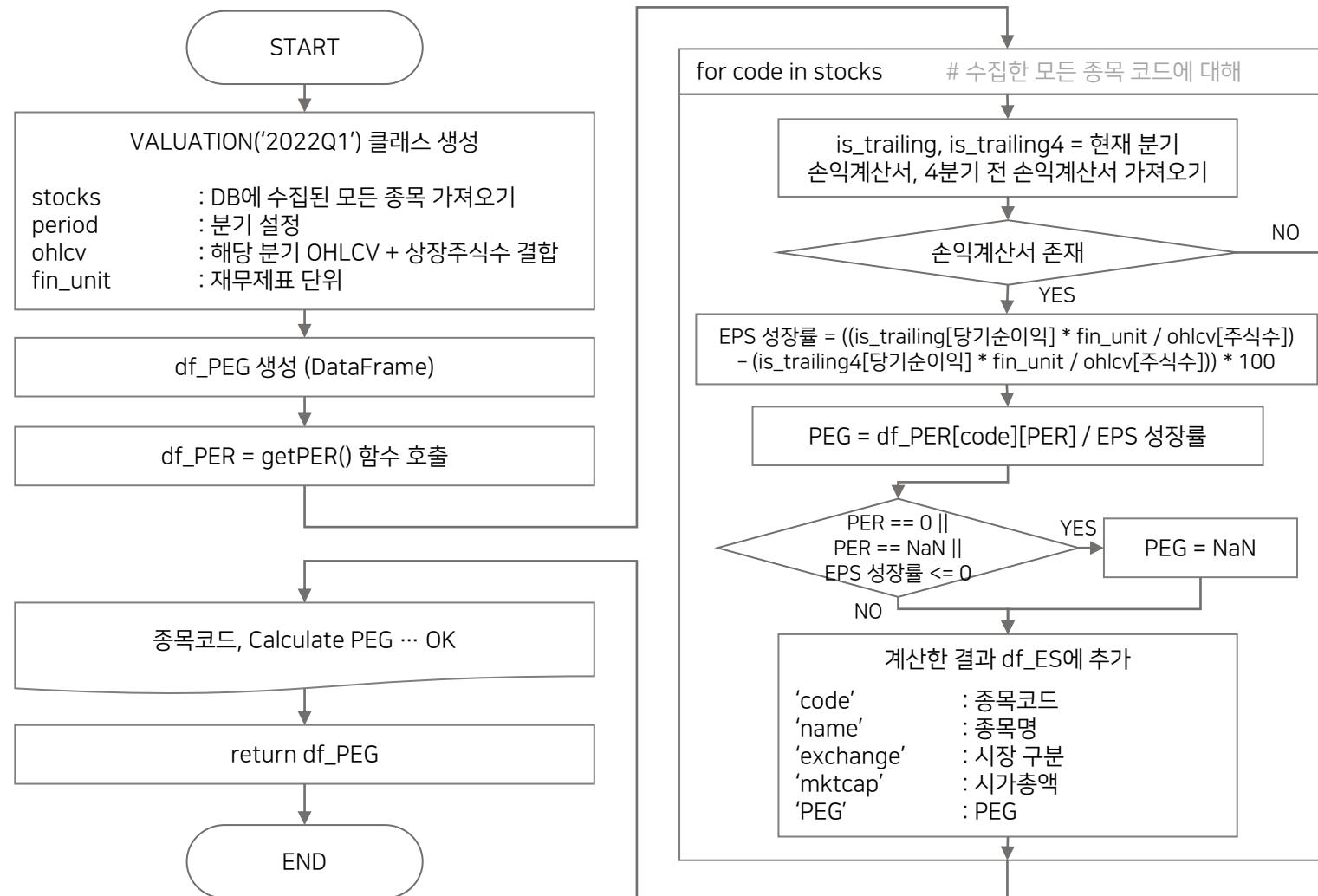
factor_list = ['NCAV']
result = stock_select2(df_factor, 0.3, 'KOSPI', 30, factor_list) # NCAV, 시가총액 상위 30%, KOSPI 종목 중 30개 선택
result
```

code	name	NCAV	score
1 006800	미래에셋증권	7.244344e+13	100.000000
2 016360	삼성증권	6.727760e+13	92.863020
3 039490	키움증권	6.033157e+13	83.266587
4 005940	NH투자증권	3.210410e+13	44.266325
5 003540	대신증권	2.444576e+13	33.687777
6 003470	유안타증권	1.994691e+13	27.472293
7 008560	메리츠증권	1.148073e+13	15.775665
8 000720	현대건설	9.469998e+12	12.997692
9 030610	교보증권	5.032844e+12	6.867445
10 001510	SK증권	4.203789e+12	5.722047
11 294870	HDC현대산업개발	3.740799e+12	5.082393
12 003240	태광산업	2.949000e+12	3.988466
13 003530	한화투자증권	2.938918e+12	3.974537
14 001500	현대차증권	2.218751e+12	2.979575
15 018260	삼성에스디에스	1.926802e+12	2.576227
16 161390	한국타이어앤티크놀로지	1.748306e+12	2.329622
17 285130	SK케미칼	1.619641e+12	2.151862
18 012630	HDC	1.405091e+12	1.855445
19 267270	현대건설기계	1.249416e+12	1.640370
20 002790	아모레G	1.011479e+12	1.311642
21 004490	세방전지	9.122860e+11	1.174600
22 020000	한섬	9.106465e+11	1.172335
23 003060	에이프로젝트	8.372299e+11	1.070905
24 003030	세아제강지주	7.942217e+11	1.011486
25 081000	일진다이아	7.421612e+11	0.939561
26 126560	현대유처넷	7.300897e+11	0.922883
27 226320	잇즈한글	6.130455e+11	0.761178
28 005850	에스엘	4.816053e+11	0.579584
29 064960	SNT모티브	4.448879e+11	0.528857
30 000650	대한유화	4.317910e+11	0.510762

### 2) NCAV 기준 종목 선정 결과

시가총액 상위 30% KOSPI 종목에 대해  
NCAV 0이상,  
높은 순 30개 종목 선정

## 03. 가치주 종목 추출 10) 주가수익 성장비율 (PEG)



---

### 03. 가치주 종목 추출 10) 주가수익 성장비율 (PEG)

현재 EPS와 4분기 전 EPS로 EPS 성장률을 구해야 하지만,  
4분기 재무제표 데이터만 존재하므로  
4분기 전의 트레일링 데이터를 만들 수 없어 구하지 못하였다.

---

04

## 우량주 종목 추출

---

## 04. 우량주 종목 추출 | 우량주 클래스

우량주 종목 추출을 위한 지표 계산과 관련된 변수, 함수를 담고 있는 클래스

변수	type	설명
period	str	지표 계산 기간
stocks	DataFrame	stocks 테이블에 저장된 모든 종목 정보
ohlcv	DataFrame	prices 테이블에 저장된 해당 분기 수정 주가 데이터
fin_unit	int	재무제표 단위 맞추기 위한 변수 (단위: 억원)

함수	parameter	return	설명
getROA()	void	DataFrame	DB에 저장된 모든 종목에 대해 ROA 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getROE()	void	DataFrame	ROE 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getRIM()	rate	DataFrame	목표 수익률 rate를 파라미터로 받아 RIM 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getGP/A()	void	DataFrame	GP/A 값을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getStability()	void	DataFrame	부채비율, 차입금비율을 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getGrowthRate()	void	DataFrame	성장을 지표를 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getTurnover()	void	DataFrame	회전률 지표를 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수
getGrossMargin()	void	DataFrame	이익률 지표를 계산하고, 코드, 이름, 시장 구분, 시가총액과 함께 DataFrame에 담아 return 하는 함수

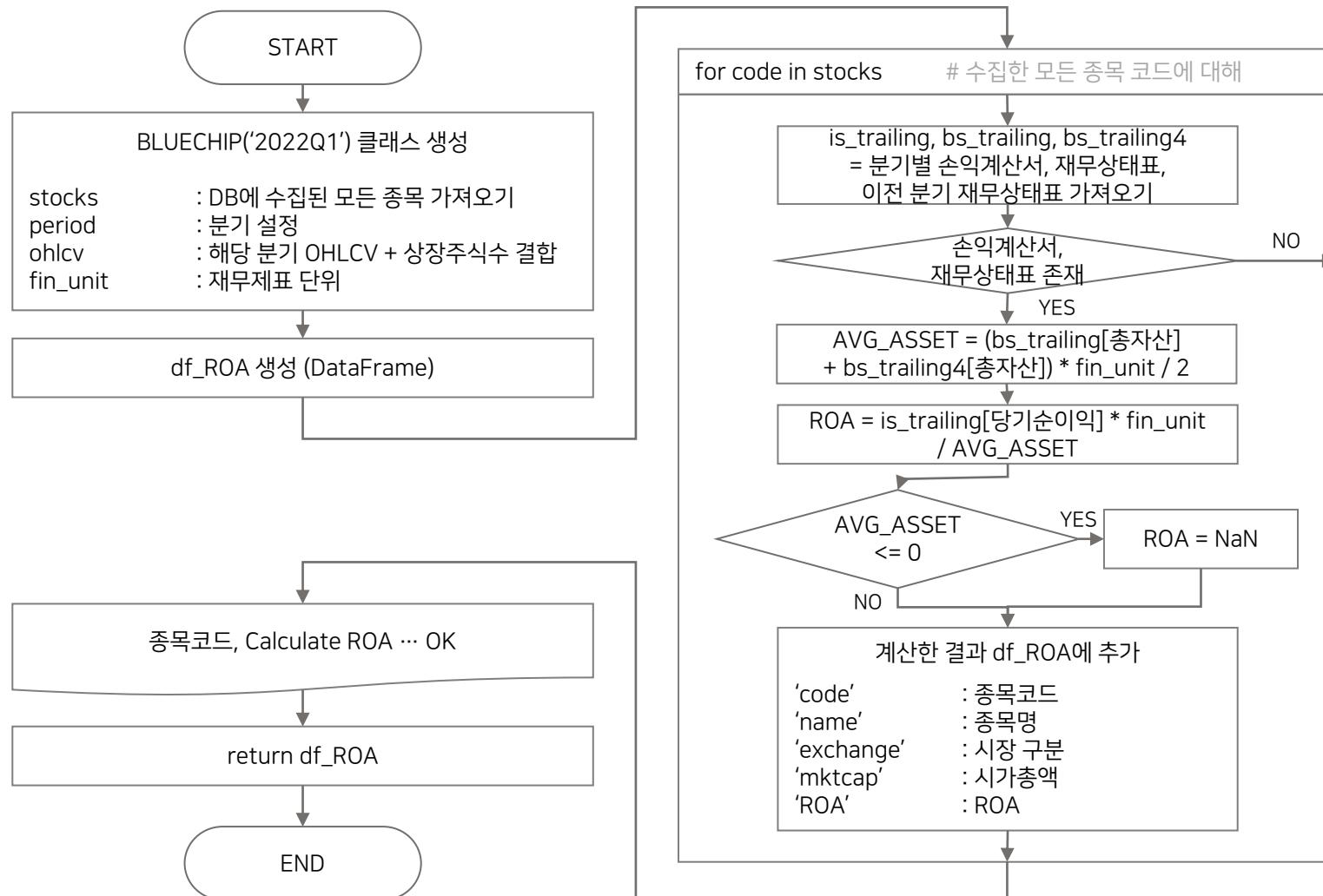
## 04. 우량주 종목 추출 | 우량주 클래스

### # 5. 우량주 종목 추출

#### ## 우량주 지표 계산 클래스

```
class BLUECHIP:  
    period = None # 분기 할당 (str)  
    stocks = None # 수집된 종목 할당 (DataFrame)  
    ohlcv = None # 분기별 수정주가 할당 (DataFrame)  
    fin_unit = 0 # 재무제표 단위 설정  
  
    def __init__(self, term):  
        print("init 실행 중", datetime.now())  
  
        self.stocks = STOCK().getAllStocks() # 수집된 종목 가져오기  
        self.period = termToPeriod(term) # 분기 설정  
        self.ohlcv = PRICE().getTermOHLCV(term) # 해당 분기 수정주가 가져오기  
        self.ohlcv = self.ohlcv.set_index('code') # 종목 코드를 인덱스로 사용  
        self.fin_unit = 100000000 # 재무제표 단위 : 억원  
  
        print("init 실행 완료", datetime.now())
```

## 04. 우량주 종목 추출 1) 자산대비 이익 (ROA)



## 04. 우량주 종목 추출 1) 자산대비 이익 (ROA)

```
# 투자 효율이 좋은 기업 (자산 대비 이익)
def getROA(self):
    # ROA 계산 후 저장할 DataFrame 생성
    df_ROA = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 불가별 손익계산서, 재무상태표 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)

        # 재무제표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or bs_trailing is None or code not in self.ohlcvs.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. 자산 평균 구하기 : (기초총자산 + 기말총자산) / 2
        # 트레일링 데이터 부족하므로 현재 자산만으로 계산 (balances > asset)
        AVG_ASSET = float(bs_trailing['asset'] * self.fin_unit)

        # 3. ROA = 당기순이익/자산평균 계산
        # 당기순이익: incomes > netinc
        if AVG_ASSET <= 0:
            ROA = np.NaN
        else:
            ROA = float((is_trailing['netinc'] * self.fin_unit) / AVG_ASSET)

        # 4. 계산한 결과 DataFrame에 추가
        df_ROA = df_ROA.append({
            'code': code,
            'name': name,
            'exchange': exc,
            'period': self.period,
            'mktpcap': self.ohlcvs.loc[code]['mktpcap'],
            'ROA': ROA }, ignore_index=True)

        print(code, ": Calculate ROA ... OK")

    return df_ROA
```

(기초총자산+기말총자산) / 2로 자산평균을 구해야 하지만,  
DB에 4분기 데이터만 존재하므로  
이전 분기 트레일링 데이터를 구할 수 없다.  
따라서 자산 평균 대신 현재 자산 값만으로 ROA를 계산했다.

## 04. 우량주 종목 추출 1) 자산대비 이익 (ROA)

getROA() 실행 결과: 모든 종목에 대해 ROA 계산

### BLUECHIP 클래스 사용 예시

```
b = BLUECHIP('2022Q1')
```

```
init 실행 중 2022-06-16 18:15:29.167604  
init 실행 완료 2022-06-16 18:23:01.524275
```

### ### ROA

```
# ROA 구하기
```

```
ROA_list = b.getROA()  
ROA_list
```

```
140910 : Not enough data to calculate ...  
078520 : Calculate ROA ... OK  
298380 : Calculate ROA ... OK  
203400 : Calculate ROA ... OK  
195990 : Calculate ROA ... OK  
003800 : Calculate ROA ... OK  
088800 : Calculate ROA ... OK  
241840 : Calculate ROA ... OK  
312610 : Calculate ROA ... OK  
015260 : Calculate ROA ... OK  
234070 : Empty Incomes  
234070 : Empty Balances  
234070 : Not enough data to calculate ...  
072990 : Calculate ROA ... OK  
227100 : Calculate ROA ... OK  
044990 : Empty Incomes  
044990 : Empty Balances  
044990 : Not enough data to calculate ...  
176440 : Calculate ROA ... OK  
042290 : Calculate ROA ... OK
```

```
# ROA 구하기
```

```
ROA_list = b.getROA()  
ROA_list
```

	code	name	exchange	period	mktcap	ROA
0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	0.015228
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	-0.011198
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	0.016370
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	0.032554
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	0.000000
	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	0.011092
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	0.000000
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	0.016718
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	0.009988
2181	238490	힐스	KOSDAQ	2022/03	1.032807e+11	-0.005304

## 04. 우량주 종목 추출 1) 자산대비 이익 (ROA)

```
# ROA 종목 선정  
df_factor = ROA_list.copy()  
factor_list = ['ROA']  
  
result = stock_select2(df_factor, 0.2, 'ALL', 30, factor_list) # ROA, 시가총액 상위 20%, 전체 종목 중 30개 선택  
result
```

code	name	ROA	score
1 205470	휴마시스	0.258004	100.000000
2 007700	F&F홀딩스	0.222149	86.102861
3 950130	엑세스바이오	0.166796	64.648605
4 011200	HMM	0.123698	47.944320
5 383800	LX홀딩스	0.090937	38.385935
6 287410	제이시스메디칼	0.098082	38.015787
7 096530	씨전	0.095140	36.875304
8 137310	에스디바이오센서	0.094227	36.521518
9 112040	위메이드	0.087798	34.029896
10 383220	F&F	0.077311	29.964948
11 108320	LX세미콘	0.075499	29.262726
12 000990	DB하이텍	0.073159	28.355908
13 348210	넥스汀	0.071924	27.877020
14 089970	에이피티씨	0.069239	26.836576
15 042700	한미반도체	0.063949	24.786135
16 035900	JYP Ent.	0.061419	23.805523
17 011780	금호석유	0.060207	23.335894
18 004000	롯데정밀화학	0.058249	22.576846
19 036930	주성엔지니어링	0.057817	22.409466
20 121800	비엔트	0.055578	21.541401
21 214150	클래시스	0.055160	21.379674
22 195870	해성디에스	0.053885	20.885549
23 298020	효성티앤씨	0.053866	20.877816
24 272290	이녹스첨단소재	0.053405	20.699424
25 357780	솔브레인	0.051053	19.787882
26 194480	데브시스터즈	0.049642	19.240652
27 215000	골프존	0.049504	19.187396
28 266280	미원에스씨	0.048888	18.948409
29 067160	아프리카TV	0.048764	18.900482
30 106240	파인테크닉스	0.048620	18.844620

### 1) ROA 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
ROA 높은 순 30개 종목 선정

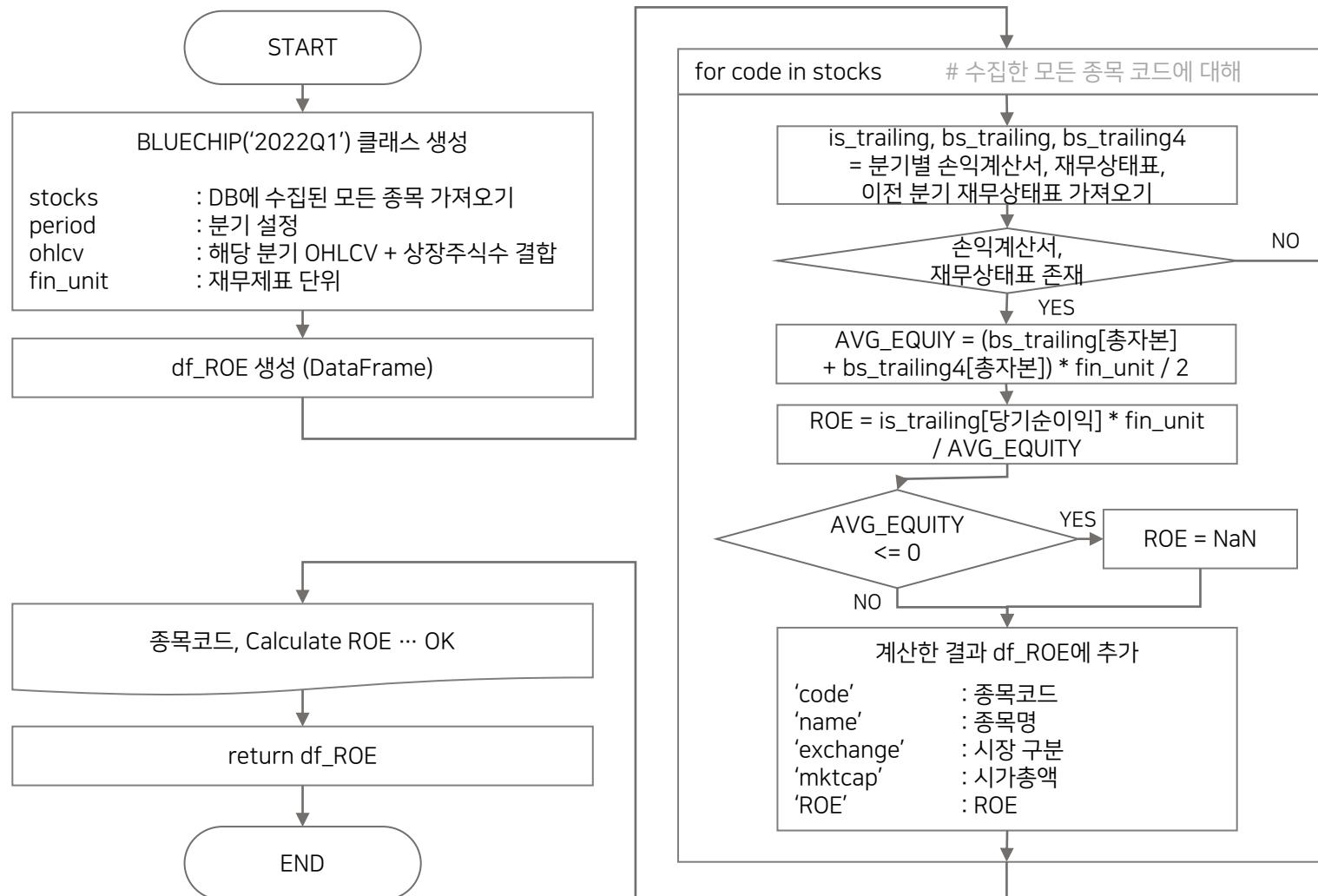
```
# ROA 종목 선정  
df_factor = ROA_list.copy()  
factor_list = ['ROA']  
  
result = stock_select2(df_factor, 0.2, 'KOSPI', 30, factor_list) # ROA, 시가총액 상위 20%, KOSPI 종목 중 30개 선택  
result
```

code	name	ROA	score
1 007700	F&F홀딩스	0.222149	100.000000
2 011200	HMM	0.123698	55.682610
3 383800	LX홀딩스	0.099037	44.581486
4 137310	에스디바이오센서	0.094227	42.416150
5 383220	F&F	0.077311	34.801338
6 000990	DB하이텍	0.073159	32.932597
7 042700	한미반도체	0.063949	28.786657
8 011780	금호석유	0.060207	27.102344
9 004000	롯데정밀화학	0.058249	26.220785
10 195870	해성디에스	0.053885	24.256509
11 298020	효성티앤씨	0.053866	24.247528
12 268280	미원에스씨	0.048888	22.006712
13 284740	쿠쿠홈시스	0.043117	19.409218
14 002840	미원상사	0.039175	17.634662
15 014830	유니드	0.036093	16.247160
16 093370	후성	0.035300	15.890219
17 010060	OCI	0.034411	15.490276
18 014680	한솔케미칼	0.034153	15.373837
19 192400	쿠쿠홀딩스	0.034070	15.336496
20 035720	카카오	0.033849	15.237216
21 009160	SIMPAC	0.033268	14.975426
22 001230	동국제강	0.032274	14.528283
23 192650	드림텍	0.031434	14.150131
24 021240	코웨이	0.031179	14.035095
25 298050	효성첨단소재	0.030496	13.727825
26 259960	크래프톤	0.030058	13.530570
27 000660	SK하이닉스	0.029723	13.379949
28 011070	LG이노텍	0.029400	13.234304
29 034120	SBS	0.029205	13.146768
30 003550	LG	0.028749	12.941562

### 2) ROA 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
ROA 높은 순 30개 종목 선정

## 04. 우량주 종목 추출 2) 자본대비 이익 (ROE)



## 04. 우량주 종목 추출 2) 자본대비 이익 (ROE)

```
# 투자 효율이 좋은 기업 (자본 대비 이익)
def getROE(self):
    # ROE 계산 후 저장할 DataFrame 생성
    df_ROE = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 눈기별 손익계산서, 재무상태표 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)

        # 재무제표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or bs_trailing is None or code not in self.ohlcvs.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. 자본 평균 구하기 : (기초총자본 + 기말총자본) / 2
        # 트레일링 데이터 부족하므로 현재 자본만으로 계산 (balances > equity)
        AVG_EQUITY = float(bs_trailing['equity'] * self.fin_unit)

        # 3. ROE = 당기순이익/자본평균 계산
        # 당기순이익: incomes > netinc
        if AVG_EQUITY <= 0:
            ROE = np.NaN
        else:
            ROE = float((is_trailing['netinc'] * self.fin_unit) / AVG_EQUITY)

        # 4. 계산한 결과 DataFrame에 추가
        df_ROE = df_ROE.append({
            'code':code,
            'name':name,
            'exchange': exc,
            'period': self.period,
            'mktpcap': self.ohlcvs.loc[code]['mktpcap'],
            'ROE': ROE }, ignore_index=True)

        print(code, ": Calculate ROE ... OK")

    return df_ROE
```

(기초총자본+기말총자본) / 2로 자본평균을 구해야 하지만,  
DB에 4분기 데이터만 존재하므로  
이전 분기 트레일링 데이터를 구할 수 없다.  
따라서 자본 평균 대신 현재 자본 값만으로 ROE를 계산했다.

## 04. 우량주 종목 추출 2) 자본대비 이익 (ROE)

getROE() 실행 결과: 모든 종목에 대해 ROE 계산

### ROE

# ROE 구하기

```
ROE_list = b.getROE()  
ROE_list
```

```
...  
005305 : Empty Incomes  
005305 : Empty Balances  
005305 : Not enough data to calculate ...  
011170 : Calculate ROE ... OK  
002270 : Calculate ROE ... OK  
071840 : Calculate ROE ... OK  
038060 : Calculate ROE ... OK  
162120 : Empty Incomes  
162120 : Empty Balances  
162120 : Not enough data to calculate ...  
253610 : Empty Incomes  
253610 : Empty Balances  
253610 : Not enough data to calculate ...  
085370 : Calculate ROE ... OK  
08537M : Empty Incomes  
08537M : Empty Balances  
08537M : Not enough data to calculate ...  
060240 : Calculate ROE ... OK  
058470 : Calculate ROE ... OK  
...
```

# ROE 구하기

```
ROE_list = b.getROE()  
ROE_list
```

	code	name	exchange	period	mktcap	ROE
0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	0.059318
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	-0.043428
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	0.028466
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	0.076769
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	0.000000
	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	0.018331
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	0.000000
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	0.040262
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	0.028064
2181	238490	힘스	KOSDAQ	2022/03	1.032807e+11	-0.006409

## 04. 우량주 종목 추출 2) 자본대비 이익 (ROE)

```
# ROE 종목 선정
df_factor = ROE_list.copy()
factor_list = ['ROE']

result = stock_select2(df_factor, 0.2, 'ALL', 30, factor_list) # ROA, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	ROE	score
1 205470	흥마시스	0.402038	100.000000
2 007700	F&F풀링스	0.286295	71.211076
3 950130	엑세스바이오	0.275704	68.576711
4 055550	신한지주	0.272109	67.682445
5 011200	HMM	0.226671	56.380485
6 383220	F&F	0.170327	42.366019
7 287410	제이시스템디컬	0.158077	39.318940
8 298020	효성티엔씨	0.151830	37.765157
9 112040	위메이드	0.143906	35.794254
10 298050	효성첨단소재	0.132016	32.836794
11 137310	에스디바이오센서	0.131165	32.624960
12 096530	씨젠	0.130827	32.540986
13 036930	주성엔지니어링	0.119420	29.703599
14 222800	실험	0.117104	29.127687
15 108320	LX세미콘	0.114927	28.586160
16 106240	파인테크닉스	0.104701	26.042623
17 139130	DGB금융지주	0.101377	25.215742
18 000990	DB하이텍	0.101377	25.215742
19 383800	LX풀링스	0.100989	25.119277
20 011780	금호석유	0.096933	24.110507
21 067160	아프리카TV	0.094589	23.527436
22 089970	에이피티씨	0.093956	23.369944
23 195870	해성디에스	0.088950	22.124735
24 348210	넥스틴	0.085855	21.354912
25 272290	이녹스첨단소재	0.081679	20.316138
26 215000	골프존	0.081257	20.211419
27 137400	피언티	0.080822	20.103177
28 042700	한미반도체	0.080218	19.952847
29 090460	비에이치	0.079502	19.774691
30 035900	JYP Ent.	0.078217	19.455251

### 1) ROE 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
ROE 높은 순 30개 종목 선정

```
# ROE 종목 선정
df_factor = ROE_list.copy()
factor_list = ['ROE']

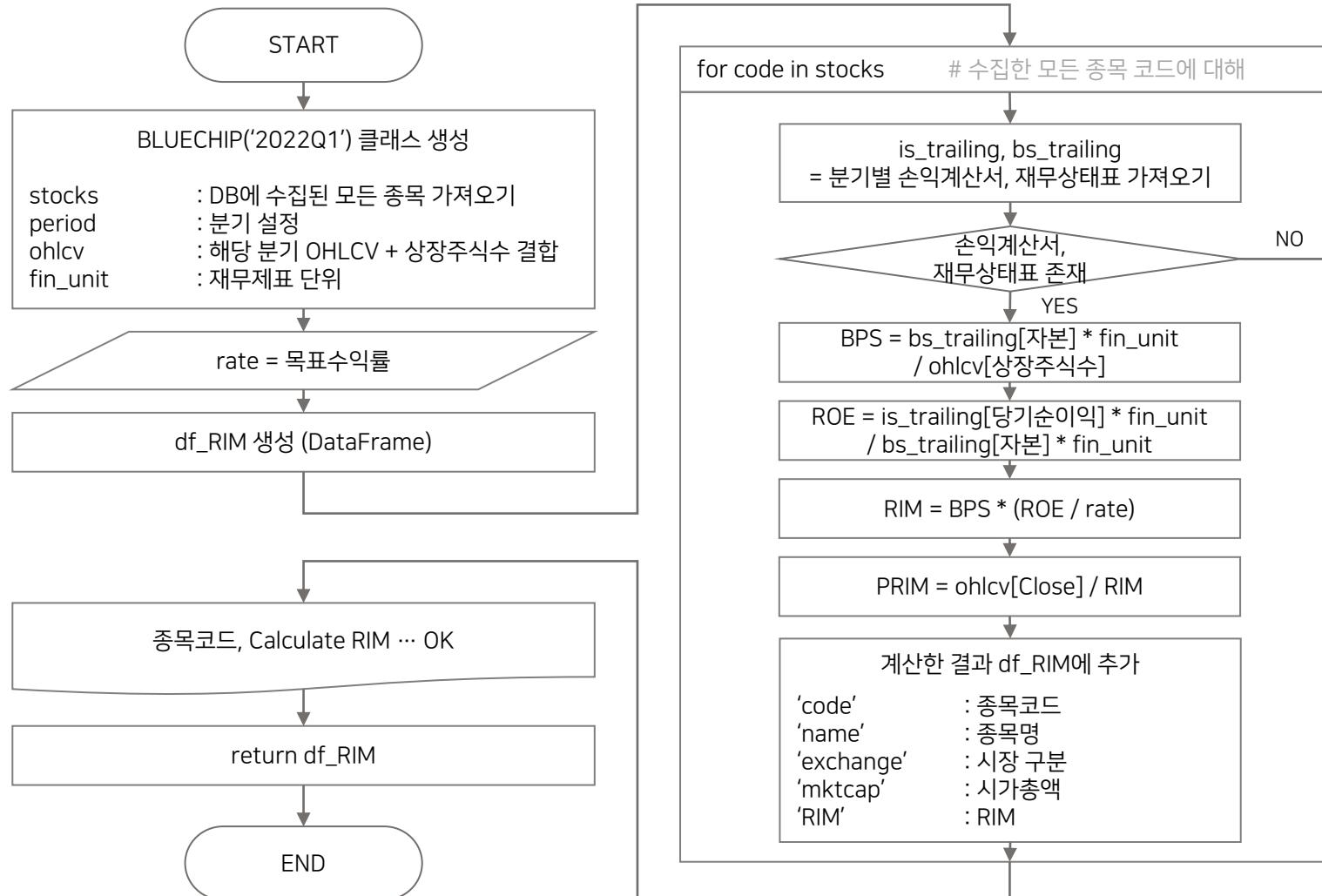
result = stock_select2(df_factor, 0.2, 'KOSPI', 30, factor_list) # ROA, 시가총액 상위 20%, KOSPI 종목 중 30개 선택
result
```

code	name	ROE	score
1 007700	F&F풀링스	0.286295	100.000000
2 055550	신한지주	0.272109	95.044830
3 011200	HMM	0.226671	79.173759
4 383220	F&F	0.170327	59.493581
5 298020	효성티엔씨	0.151830	53.032701
6 298050	효성첨단소재	0.132016	46.111920
7 137310	에스디바이오센서	0.131165	45.814447
8 000990	DB하이텍	0.101377	35.409860
9 139130	DGB금융지주	0.101377	35.409860
10 383800	LX풀링스	0.100989	35.274396
11 011780	금호석유	0.096933	33.857805
12 195870	해성디에스	0.088950	31.069234
13 042700	한미반도체	0.080218	28.019303
14 093370	후성	0.075367	26.324858
15 001230	동국제강	0.074332	25.963534
16 001120	LX인터내셔널	0.073059	25.518838
17 004000	롯데정밀화학	0.070882	24.758453
18 011070	LG이노텍	0.070549	24.641883
19 010950	S-Oil	0.068972	24.091173
20 268280	미원에스씨	0.065109	22.742053
21 021240	코웨이	0.064933	22.680443
22 042670	현대두산인프라코어	0.064034	22.366249
23 010060	OCI	0.063399	22.144566
24 210980	SK디엔디	0.061586	21.511410
25 003070	코오롱글로벌	0.060199	21.026818
26 192650	드림텍	0.060049	20.974337
27 005880	대한해운	0.058359	20.384249
28 284740	쿠쿠홀시스	0.058246	20.344635
29 034120	SBS	0.057157	19.964235
30 009160	SIMPAC	0.056236	19.642832

### 2) ROE 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
ROE 높은 순 30개 종목 선정

## 04. 우량주 종목 추출 3) 잔여이익모델 (RIM)



## 04. 우량주 종목 추출 3) 잔여이익모델 (RIM)

```
# 회계사가 유행시킨 장기 투자 전략 (잔여 이익 모델)
def getRIM(self, rate): # rate: 목표수익률 입력받기
    # RIM 계산 후 저장할 DataFrame 생성
    df_RIM = pd.DataFrame()

    # 모든 종목에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 손익계산서, 재무상태표 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)

        # 재무제표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlc 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or bs_trailing is None or code not in self.ohlcv.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. BPS = 자본/상장주식수 계산하기
        # 자본: balances > equity
        # 상장주식수: stocks > shares
        BPS = float((bs_trailing['equity'] * self.fin_unit) / self.ohlcv.loc[code]['shares'])

        # 3. ROE = 당기순이익/자본 계산하기
        # 당기순이익: incomes > netinc
        # 자본: balances > equity
        ROE = float((is_trailing['netinc'] * self.fin_unit) / (bs_trailing['equity'] * self.fin_unit))

        # 4. RIM = BPS * (ROE / 목표수익률) 계산하기
        RIM = float(BPS * (ROE / rate))

        # 5. P/RIM = 주가 / RIM 계산하기
        if self.ohlcv.loc[code]['shares'] <= 0 or float(bs_trailing['equity']) <= 0 or RIM <= 0:
            PRIM = np.NaN
        else:
            PRIM = float(self.ohlcv.loc[code]['Close'] / RIM)

        # 4. 계산한 결과 DataFrame에 추가
        df_RIM = df_RIM.append({
            'code':code,
            'name':name,
            'exchange':exc,
            'period':self.period,
            'mktcap':self.ohlcv.loc[code]['mktcap'],
            'RIM':PRIM}, ignore_index=True)

    print(code, ": Calculate RIM ... OK")

return df_RIM
```

ROE 계산 시,  
(기초총자본+기말총자본) / 2로 자본평균을 구해야 하지만,  
DB에 4분기 데이터만 존재하므로  
이전 분기 트레일링 데이터를 구할 수 없다.  
따라서 자본 평균 대신 현재 자본 값만으로 ROE를 계산했다.

## 04. 우량주 종목 추출 3) 잔여이익모델 (RIM)

getRIM() 실행 결과: 모든 종목에 대해 RIM 계산

### ### RIM

# RIM 구하기

```
RIM_list = b.getRIM(0.1) # 목표 수익률 10%
RIM_list
```

```
272550 : Calculate RIM ... OK
000070 : Calculate RIM ... OK
000075 : Empty Incomes
000075 : Empty Balances
000075 : Not enough data to calculate ...
002810 : Calculate RIM ... OK
361670 : Calculate RIM ... OK
054540 : Calculate RIM ... OK
065570 : Calculate RIM ... OK
005680 : Calculate RIM ... OK
003720 : Calculate RIM ... OK
023000 : Calculate RIM ... OK
004380 : Calculate RIM ... OK
002450 : Calculate RIM ... OK
032280 : Calculate RIM ... OK
002290 : Calculate RIM ... OK
004440 : Calculate RIM ... OK
000520 : Calculate RIM ... OK
009770 : Calculate RIM ... OK
037460 : Calculate RIM ... OK
```

# RIM 구하기

```
RIM_list = b.getRIM(0.1) # 목표 수익률 10%
RIM_list
```

	code	name	exchange	period	mktcap	RIM
0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	0.349986
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	NaN
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	1.224396
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	0.646768
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	NaN
	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	2.708472
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	NaN
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	3.821897
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	1.942766
2181	238490	힘스	KOSDAQ	2022/03	1.032807e+11	NaN

## 04. 우량주 종목 추출 3) 잔여이익모델 (RIM)

```
# RIM 종목 선정  
df_factor = RIM_list.copy()  
factor_list = ['RIM']  
  
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # RIM, 시가총액 상위 20%, 전체 종목 중 30개 선택  
result
```

code	name	RIM	score
1 007700	F&F풀딩스	0.052015	100.000000
2 183190	아세아시멘트	0.070121	99.999831
3 032190	다우데이타	0.071406	99.999819
4 023590	다우기술	0.125126	99.999318
5 001530	DI동일	0.127930	99.999292
6 000880	한화	0.144385	99.999138
7 024110	기업은행	0.169733	99.998902
8 139480	이마트	0.171430	99.998886
9 011200	HMM	0.171447	99.998886
10 950130	액세스바이오	0.185005	99.998759
11 003380	하림지주	0.194586	99.998670
12 000210	DL	0.194704	99.998669
13 001230	동국제강	0.198916	99.998630
14 001120	LX인터내셔널	0.200591	99.998614
15 205470	휴마시스	0.205836	99.998565
16 298020	효성티앤씨	0.215651	99.998473
17 006120	SK디스커버리	0.218747	99.998445
18 034730	SK	0.220355	99.998430
19 011780	금호석유	0.252017	99.998134
20 031430	신세계인터내셔날	0.265508	99.998008
21 005880	대한해운	0.268922	99.997976
22 042670	현대두산인프라코어	0.288279	99.997796
23 001040	CJ	0.292773	99.997754
24 036460	한국가스공사	0.308344	99.997609
25 000070	삼양풀딩스	0.309248	99.997600
26 004020	현대제철	0.312658	99.997568
27 018670	SK가스	0.316157	99.997536
28 005490	POSCO풀딩스	0.320766	99.997493
29 010060	OCI	0.335467	99.997356
30 383220	F&F	0.347646	99.997242
27 004000	롯데정밀화학	0.355653	99.997167
28 001430	세아베스틸	0.363614	99.997093
29 004800	효성	0.365559	99.997075
30 139130	DGB금융지주	0.369319	99.997040

### 1) RIM 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
목표 수익률 10%,  
RIM 0이상 낮은 순 30개 종목 선정

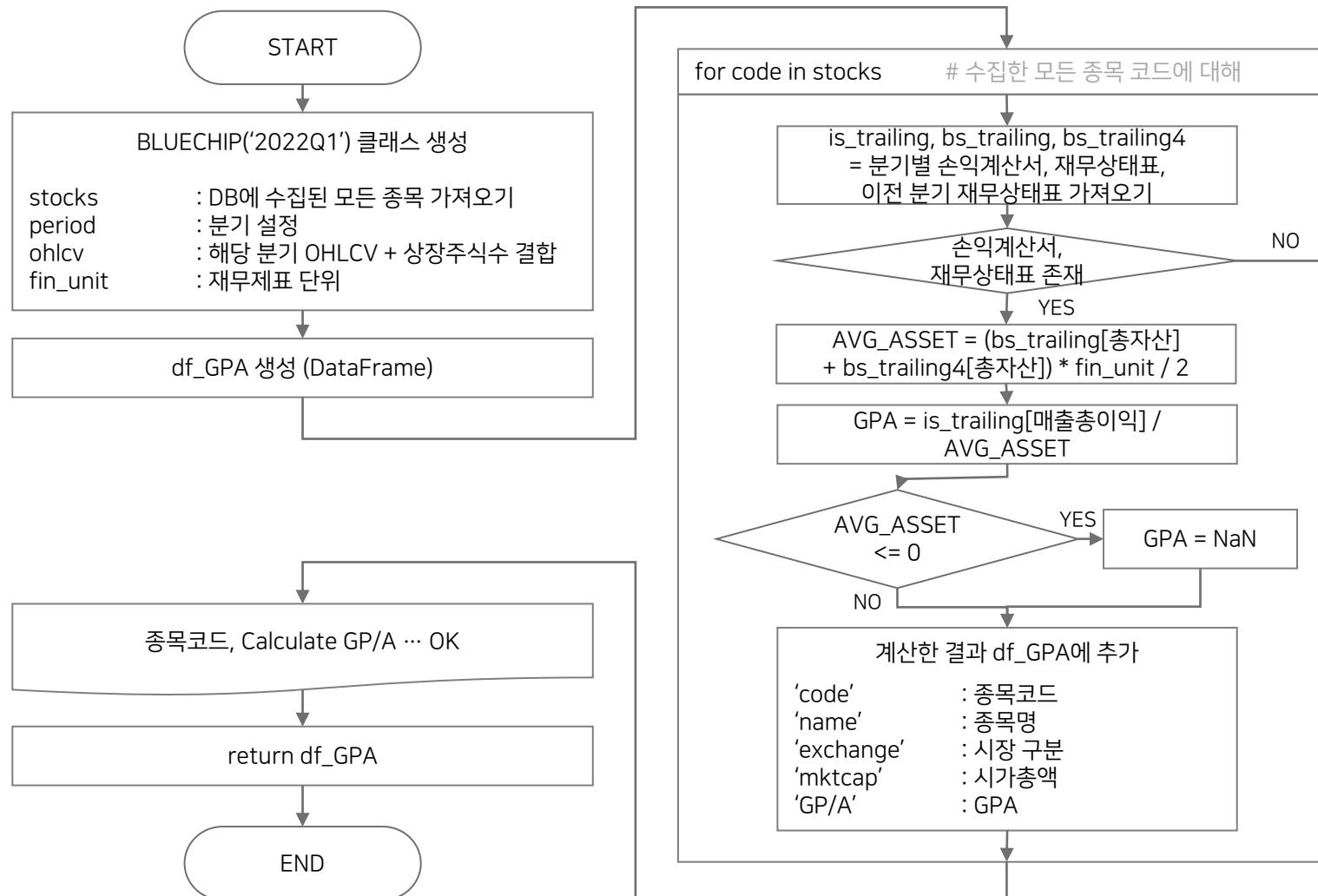
```
# RIM 종목 선정  
df_factor = RIM_list.copy()  
factor_list = ['RIM']  
  
result = stock_select(df_factor, 0.2, 'KOSPI', 30, factor_list) # RIM, 시가총액 상위 20%, KOSPI 종목 중 30개 선택  
result
```

code	name	RIM	score
1 007700	F&F풀딩스	0.052015	100.000000
2 183190	아세아시멘트	0.070121	99.999831
3 023590	다우기술	0.125126	99.999318
4 001530	DI동일	0.127930	99.999292
5 000880	한화	0.144385	99.999138
6 024110	기업은행	0.169733	99.998902
7 139480	이마트	0.171430	99.998886
8 011200	HMM	0.171447	99.998886
9 000210	DL	0.194704	99.998669
10 001230	동국제강	0.198916	99.998630
11 001120	LX인터내셔널	0.200591	99.998614
12 298020	효성티앤씨	0.215651	99.998473
13 006120	SK디스커버리	0.218747	99.998445
14 034730	SK	0.220355	99.998430
15 011780	금호석유	0.252017	99.998134
16 031430	신세계인터내셔날	0.265508	99.998008
17 005880	대한해운	0.268922	99.997976
18 042670	현대두산인프라코어	0.288279	99.997796
19 001040	CJ	0.292773	99.997754
20 036460	한국가스공사	0.308344	99.997609
21 000070	삼양풀딩스	0.309248	99.997600
22 004020	현대제철	0.312658	99.997568
23 018670	SK가스	0.316157	99.997536
24 005490	POSCO풀딩스	0.320766	99.997493
25 010060	OCI	0.335467	99.997356
26 383220	F&F	0.347646	99.997242
27 004000	롯데정밀화학	0.355653	99.997167
28 001430	세아베스틸	0.363614	99.997093
29 004800	효성	0.365559	99.997075
30 139130	DGB금융지주	0.369319	99.997040

### 2) RIM 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
목표 수익률 10%,  
RIM 0이상 낮은 순 30개 종목 선정

## 04. 우량주 종목 추출 4) GP/A (영업 효율이 좋은 기업)



## 04. 우량주 종목 추출 4) GP/A (영업 효율이 좋은 기업)

```
# 영업 효율이 좋은 기업 (GP/A)
def getGPA(self):
    # GP/A 계산 후 저장할 DataFrame 생성
    df_GPA = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 손익계산서, 재무상태표 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)

        # 재무제표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or bs_trailing is None or code not in self.ohlcvs.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. GP/A = 매출총이익 / 자산평균 계산하기
        # 매출총이익: incomes > gross
        # 자산 평균 = (기초자산 + 기말자산)/2 but 트레일링 데이터 1년치만 존재하므로, 총자산 이용
        if float(bs_trailing['asset']) <= 0:
            GPA = np.NaN
        else:
            GPA = (is_trailing['gross'] * self.fin_unit) / (bs_trailing['asset'] * self.fin_unit)

        # 4. 계산한 결과 DataFrame에 추가
        df_GPA = df_GPA.append({
            'code': code,
            'name': name,
            'exchange': exc,
            'period': self.period,
            'mktcap': self.ohlcvs.loc[code]['mktcap'],
            'GP/A': GPA }, ignore_index=True)

    print(code, ": Calculate GP/A ... OK")

return df_GPA
```

(기초총자산+기말총자산) / 2로 자산평균을 구해야 하지만,  
DB에 4분기 데이터만 존재하므로  
이전 분기 트레일링 데이터를 구할 수 없다.  
따라서 자산 평균 대신 현재 자산 값만으로 GP/A를 계산했다.

## 04. 우량주 종목 추출 4) GP/A (영업 효율이 좋은 기업)

getGPA() 실행 결과: 모든 종목에 대해 GP/A 계산

### #### GP/A

```
# GP/A 구하기
GPA_List = b.getGPA()
GPA_List
01150 : Calculate GP/A ... OK
00104K : Empty Incomes
00104K : Empty Balances
00104K : Not enough data to calculate ...
000120 : Calculate GP/A ... OK
011150 : Calculate GP/A ... OK
011155 : Empty Incomes
011155 : Empty Balances
011155 : Not enough data to calculate ...
001045 : Empty Incomes
001045 : Empty Balances
001045 : Not enough data to calculate ...
097950 : Calculate GP/A ... OK
097955 : Empty Incomes
097955 : Empty Balances
097955 : Not enough data to calculate ...
051500 : Calculate GP/A ... OK
058820 : Calculate GP/A ... OK
023460 : Calculate GP/A ... OK
056730 : Calculate GP/A ... OK
```

```
# GP/A 구하기
GPA_List = b.getGPA()
GPA_List
```

	code	name	exchange	period	mktcap	GP/A
0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	0.198893
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	0.028616
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	0.015375
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	0.067810
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	0.000000
	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	0.051224
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	0.000000
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	0.007344
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	0.035921
2181	238490	힘스	KOSDAQ	2022/03	1.032807e+11	0.031704

## 04. 우량주 종목 추출 4) GP/A (영업 효율이 좋은 기업)

```
# GP/A 종목 선정  
df_factor = GPA_list.copy()  
factor_list = ['GP/A']
```

```
result = stock_select2(df_factor, 0.2, 'ALL', 30, factor_list) # ROA, 시가총액 상위 20%, 전체 종목 중 30개 선택  
result
```

code	name	GP/A	score
1 950130	엑세스바이오	0.396320	100.000000
2 008770	호텔신라	0.383870	96.858462
3 205470	휴마시스	0.350303	88.388839
4 194480	데브시스터즈	0.334512	84.404361
5 030190	NICE평가정보	0.300702	75.873559
6 383220	F&F	0.261818	66.062291
7 008930	한미사이언스	0.252650	63.748942
8 287410	제이시스메디칼	0.230406	58.136185
9 067000	조이시티	0.225498	56.897897
10 034310	NICE	0.210311	53.065931
11 230360	에코마케팅	0.188562	47.578057
12 101730	위메이드엑스	0.187661	47.350892
13 067160	아프리카TV	0.186337	47.016859
14 031430	신세계인터넷내셔널	0.185833	46.889525
15 096530	씨젠	0.182530	46.056238
16 032640	LG유플러스	0.182517	46.052974
17 064550	바이오니아	0.176996	44.659822
18 030200	KT	0.174106	43.930696
19 181710	NHN	0.171778	43.343227
20 215000	골프존	0.168651	42.554289
21 108320	LX세미콘	0.165636	41.793470
22 053800	안랩	0.163221	41.184143
23 085370	루트로닉	0.163024	41.134469
24 051900	LG생활건강	0.158708	40.045462
25 021240	코웨이	0.158217	39.921596
26 284740	쿠쿠홈시스	0.157184	39.660730
27 215200	메가스터디교육	0.152653	38.517591
28 206560	덱스터	0.149856	37.811916
29 011200	HMM	0.147231	37.149462
30 036570	엔씨소프트	0.146701	37.015851

### 1) GP/A 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
GP/A 0이상,  
높은 순 30개 종목 선정

```
# GP/A 종목 선정  
df_factor = GPA_list.copy()  
factor_list = ['GP/A']
```

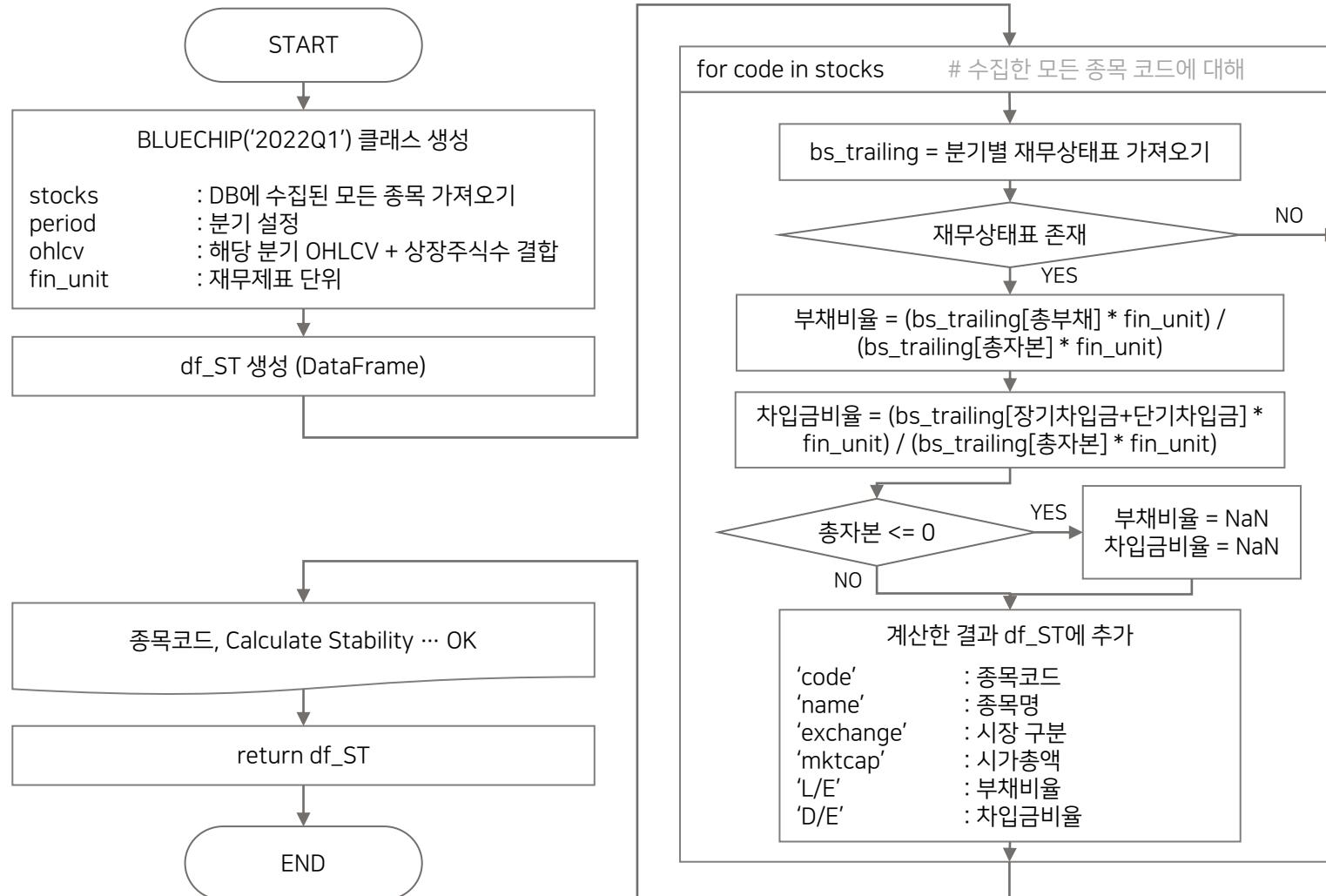
```
result = stock_select2(df_factor, 0.2, 'KOSPI', 30, factor_list) # ROA, 시가총액 상위 20%, KOSPI 종목 중 30개 선택  
result
```

code	name	GP/A	score
1 008770	호텔신라	0.383870	100.000000
2 383220	F&F	0.261818	68.204976
3 008930	한미사이언스	0.252650	65.816595
4 034310	NICE	0.210311	54.787088
5 031430	신세계인터넷내셔널	0.185833	48.410355
6 032640	LG유플러스	0.182517	47.546671
7 030200	KT	0.174106	45.355558
8 181710	NHN	0.171778	44.749035
9 051900	LG생활건강	0.158708	41.344309
10 021240	코웨이	0.158217	41.216426
11 284740	쿠쿠홈시스	0.157184	40.947098
12 011200	HMM	0.147231	38.354380
13 036570	엔씨소프트	0.146701	38.216435
14 005390	신성통상	0.143342	37.341258
15 090430	아모레퍼시픽	0.140980	36.725899
16 020000	한섬	0.140975	36.724751
17 137310	에스디바이오센서	0.140010	36.473313
18 030000	제일기획	0.137298	35.766897
19 009450	경동나비엔	0.124556	32.447494
20 383800	LX홀딩스	0.122979	32.036755
21 185750	종근당	0.122674	31.957259
22 282330	BGF리테일	0.117345	30.568892
23 009240	한샘	0.116914	30.456765
24 002790	아모레G	0.116568	30.366502
25 042700	한미반도체	0.113162	29.479296
26 000990	DB하이텍	0.113000	29.437164
27 081660	릴라홀딩스	0.110815	28.867810
28 049770	동원F&B	0.106554	27.757891
29 017670	SK텔레콤	0.103618	26.993092
30 003090	대웅	0.103347	26.922427

### 2) GP/A 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
GP/A 0이상,  
높은 순 30개 종목 선정

## 04. 우량주 종목 추출 5) 안정성 지표



## 04. 우량주 종목 추출 5) 안정성 지표

```
# 안정성 지표
def getStability(self): # LE, DE 낮을 수록 좋다?
    # Stability 계산 후 저장할 DataFrame 생성
    df_ST = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 재무상태표 가져오기
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)

        # 재무상태표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if bs_trailing is None or code not in self.ohlcv.index:
            print(code, ": Data Not Enough!")
            continue # 해당 종목은 pass

        # 2. 부채비율 = (총부채/총자본) 계산
        # 부채비율 Liability/Equity
        # 총부채: balances > liab
        # 총자본: balances > equity
        if float(bs_trailing['equity']) == 0: # 0으로 나누는 경우
            LE = np.NaN
        else:
            LE = float((bs_trailing['liab']*self.fin_unit) / (bs_trailing['equity']*self.fin_unit))

        # 3. 차입금비율 = (장기차입금+단기차입금)/총자본 계산
        # 차입금비율 Debt/Equity
        # 장기/단기차입금: balances > longdebt, curdebt
        # 총자본: balances > equity
        if float(bs_trailing['equity']) == 0: # 0으로 나누는 경우
            DE = np.NaN
        else:
            DE = float(((bs_trailing['longdebt']*self.fin_unit) + (bs_trailing['curdebt']*self.fin_unit)) / (bs_trailing['equity']*self.fin_unit))

        # 4. 계산한 결과 DataFrame에 추가
        df_ST = df_ST.append({
            'code':code,
            'name':name,
            'exchange':exc,
            'period':self.period,
            'mktcap':self.ohlcv.loc[code]['mktcap'],
            'L/E':LE, # 부채비율
            'D/E':DE}, ignore_index=True) # 차입금비율

        print(code, ": Calculate Stability ... OK")

    return df_ST
```

## 04. 우량주 종목 추출 5) 안정성 지표

getStability() 실행 결과: 모든 종목에 대해 부채비율(L/E), 차입금비율(D/E) 계산

```
b = BLUECHIP('2022Q1')
```

```
init 실행 중 2022-06-13 01:06:04.171814  
init 실행 완료 2022-06-13 01:15:47.212663
```

#### 안정성 지표

```
# 안정성지표 구하기  
Id_list = b.getStability()  
Id_list
```

```
093050 : Calculate Stability ... OK  
003550 : Calculate Stability ... OK  
034220 : Calculate Stability ... OK  
051900 : Calculate Stability ... OK  
051905 : Empty Balances  
051905 : Data Not Enough!  
373220 : Data Not Enough!  
003555 : Empty Balances  
003555 : Data Not Enough!  
032640 : Calculate Stability ... OK  
011070 : Calculate Stability ... OK  
066570 : Calculate Stability ... OK  
066575 : Empty Balances  
066575 : Data Not Enough!  
037560 : Calculate Stability ... OK  
051910 : Calculate Stability ... OK  
051915 : Empty Balances  
051915 : Data Not Enough!  
079550 : Calculate Stability ... OK  
006260 : Calculate Stability ... OK
```

```
# 안정성지표 구하기  
Id_list = b.getStability()  
Id_list
```

	code	name	exchange	period	mktcap	L/E	D/E
0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	2.895370	1.011022
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	2.878094	1.551008
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	0.738874	0.424143
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	1.358210	0.490054
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	0.370914	0.000000
...	...	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	0.652682	0.471724
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	41.657558	0.000000
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	1.408216	0.006598
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	1.809843	0.527872
2181	238490	힘스	KOSDAQ	2022/03	1.032807e+11	0.208250	0.043878

## 04. 우량주 종목 추출 5) 안정성 지표

```
# 안정성지표 종목 선정 (부채비율)
df_factor = Id_list.copy()
factor_list = ['L/E'] # 부채비율
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # 부채비율, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	L/E	score
1 003470	유안타증권	0.000000	100.000000
2 003540	대신증권	0.000000	100.000000
3 003530	한화투자증권	0.000000	100.000000
4 008560	메리츠증권	0.000000	100.000000
5 010050	우리종금	0.000000	100.000000
6 041190	우리기술투자	0.000000	100.000000
7 034830	한국토지신탁	0.000000	100.000000
8 039490	키움증권	0.000000	100.000000
9 005940	NH투자증권	0.000000	100.000000
10 006800	미래에셋증권	0.000000	100.000000
11 016360	삼성증권	0.000000	100.000000
12 029780	삼성카드	0.000050	100.000000
13 383800	LX홀딩스	0.019706	100.000000
14 199800	풀젠	0.035553	100.000000
15 323990	박셀바이오	0.039761	100.000000
16 376300	디어유	0.080760	99.999999
17 026960	동서	0.092474	99.999999
18 214370	케어젠	0.094721	99.999999
19 000240	한국엔컴퍼니	0.094954	99.999999
20 950210	프레스티지바이오파마	0.096818	99.999999
21 067630	HLB생명과학	0.109883	99.999999
22 082270	젬백스	0.115339	99.999999
23 058470	리노공업	0.116513	99.999999
24 271940	일진하이솔루스	0.118457	99.999999
25 019170	신풍제약	0.129594	99.999999
26 950220	네오아이뮨텍	0.132905	99.999999
27 141080	레고켐바이오	0.133162	99.999999
28 003550	LG	0.141360	99.999999
29 007390	네이처셀	0.144712	99.999999
30 064760	티씨케이	0.147842	99.999999

### 1) 부채비율 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
부채비율 0이상,  
낮은 순 30개 종목 선정

```
# 안정성지표 종목 선정 (부채비율)
df_factor = Id_list.copy()
factor_list = ['L/E'] # 부채비율
result = stock_select(df_factor, 0.2, 'KOSPI', 30, factor_list) # 부채비율, 시가총액 상위 20%, KOSPI 종목 중 30개 선택
result
```

code	name	L/E	score
1 003530	한화투자증권	0.000000	100.000000
2 034830	한국토지신탁	0.000000	100.000000
3 039490	키움증권	0.000000	100.000000
4 006800	미래에셋증권	0.000000	100.000000
5 003540	대신증권	0.000000	100.000000
6 010050	우리종금	0.000000	100.000000
7 016360	삼성증권	0.000000	100.000000
8 005940	NH투자증권	0.000000	100.000000
9 008560	메리츠증권	0.000000	100.000000
10 003470	유안타증권	0.000000	100.000000
11 029780	삼성카드	0.000050	100.000000
12 383800	LX홀딩스	0.019706	100.000000
13 026960	동서	0.092474	99.999999
14 000240	한국엔컴퍼니	0.094954	99.999999
15 950210	프레스티지바이오파마	0.096818	99.999999
16 271940	일진하이솔루스	0.118457	99.999999
17 019170	신풍제약	0.129594	99.999999
18 003550	LG	0.141360	99.999999
19 035250	강원랜드	0.157301	99.999999
20 192400	쿠쿠홀딩스	0.204284	99.999998
21 006650	대한유화	0.207912	99.999998
22 004000	롯데정밀화학	0.216882	99.999998
23 259960	크래프톤	0.222258	99.999998
24 002790	아모레G	0.225212	99.999998
25 009420	한울바이오파마	0.233621	99.999998
26 003240	태광산업	0.238320	99.999998
27 002840	미원상사	0.245048	99.999998
28 042700	한미반도체	0.254402	99.999998
29 020150	일진마티리얼즈	0.261539	99.999998
30 033270	유나이티드제약	0.263790	99.999998

### 2) 부채비율 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
부채비율 0이상,  
낮은 순 30개 종목 선정

## 04. 우량주 종목 추출 5) 안정성 지표

```
# 안정성지표 종목 선정 (차입금비율)
df_factor = Id_list.copy()
factor_list = ['D/E'] # 차입금비율
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # 차입금비율, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	D/E	score
1 298380	에이비엘바이오	0.0	100.0
2 950220	네오이뮨텍	0.0	100.0
3 235980	메드락트	0.0	100.0
4 048530	인트론바이오	0.0	100.0
5 101730	위메이드맥스	0.0	100.0
6 069080	웰젠	0.0	100.0
7 215000	골프존	0.0	100.0
8 009420	한울바이오파마	0.0	100.0
9 082640	동양생명	0.0	100.0
10 214370	케어젠	0.0	100.0
11 007390	네이처셀	0.0	100.0
12 030190	NICE평가정보	0.0	100.0
13 053800	안랩	0.0	100.0
14 215600	신라젠	0.0	100.0
15 376300	디어유	0.0	100.0
16 029780	삼성카드	0.0	100.0
17 064760	티씨케이	0.0	100.0
18 271940	일진하이솔루스	0.0	100.0
19 145020	휴젤	0.0	100.0
20 240810	원익IPS	0.0	100.0
21 051600	한전KPS	0.0	100.0
22 035900	JYP Ent.	0.0	100.0
23 058470	리노공업	0.0	100.0
24 108320	LX세미콘	0.0	100.0
25 012750	에스원	0.0	100.0
26 383800	LX홀딩스	0.0	100.0
27 336260	두산퓨얼셀	0.0	100.0
28 071840	롯데하이마트	0.0	100.0
29 089970	에이피티씨	0.0	100.0
30 199800	풀젠	0.0	100.0

### 1) 차입금비율 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
차입금비율 0이상,  
낮은 순 30개 종목 선정

```
# 안정성지표 종목 선정 (차입금비율)
df_factor = Id_list.copy()
factor_list = ['D/E'] # 차입금비율
result = stock_select(df_factor, 0.2, 'KOSPI', 30, factor_list) # 차입금비율, 시가총액 상위 20%, KOSPI 종목 중 30개 선택
result
```

code	name	D/E	score
1 383800	LX홀딩스	0.000000	100.0
2 082640	동양생명	0.000000	100.0
3 012750	에스원	0.000000	100.0
4 071840	롯데하이마트	0.000000	100.0
5 336260	두산퓨얼셀	0.000000	100.0
6 009420	한울바이오파마	0.000000	100.0
7 005690	파미셀	0.000000	100.0
8 051600	한전KPS	0.000000	100.0
9 064960	SNT모티브	0.000000	100.0
10 271940	일진하이솔루스	0.000000	100.0
11 029780	삼성카드	0.000000	100.0
12 018260	삼성에스디에스	0.000066	100.0
13 259960	크래프톤	0.000070	100.0
14 114090	GKL	0.000130	100.0
15 377300	카카오페이지	0.000260	100.0
16 000370	한화손해보험	0.000308	100.0
17 192400	루크홀딩스	0.000350	100.0
18 137310	에스디바이오센서	0.000461	100.0
19 307950	현대오토에버	0.000573	100.0
20 214320	이노션	0.000595	100.0
21 020150	일진머티리얼즈	0.000653	100.0
22 004000	롯데정밀화학	0.000668	100.0
23 035250	강원랜드	0.000767	100.0
24 026960	동서	0.001106	100.0
25 950210	프레스티지바이오파마	0.001276	100.0
26 085620	미래에셋생명	0.001777	100.0
27 003550	LG	0.001911	100.0
28 088350	한화생명	0.002563	100.0
29 271560	오리온	0.002664	100.0
30 002840	미원상사	0.002891	100.0

### 2) 차입금비율 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
차입금비율 0이상,  
낮은 순 30개 종목 선정

# 04. 우량주 종목 추출 5) 안정성 지표

```
# 안정성지표 종목 선정 (PER + 차입금비율 합성)
df_factor = pd.merge(per_list, ld_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])
factor_list = ['PER', 'D/E']
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # PER+차입금비율, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	PER	D/E	score
1 001530	DI동일	1.279302	0.456876	99.999898
2 000880	한화	1.443846	0.395390	99.999833
3 011200	HMM	1.714468	0.026868	99.999779
4 139480	이마트	1.714299	0.151358	99.999754
5 950130	엑세스바이오	1.850054	0.051344	99.999711
6 024110	기업은행	1.697331	0.791234	99.999637
7 205470	휴마시스	2.058360	0.012023	99.999621
8 001120	LX인터넷서널	2.005910	0.386508	99.999572
9 000210	DL	1.947041	0.573978	99.999563
10 001230	동국제강	1.989159	0.696803	99.999520
11 003380	하림지주	1.945864	0.872017	99.999506
12 006120	SK디스커버리	2.187466	0.364186	99.999492
13 034730	SK	2.203551	0.336127	99.999490
14 298020	효성티엔씨	2.156513	0.810922	99.999419
15 023590	다우기술	1.251256	3.083671	99.999397
16 011780	금호석유	2.520174	0.114690	99.999386
17 031430	신세계인터넷내셔널	2.655080	0.094516	99.999327
18 005880	대한해운	2.689223	0.453638	99.999241
19 042670	현대두산인프라코어	2.882786	0.389641	99.999163
20 001040	CJ	2.927732	0.343057	99.999151
21 000070	삼양홀딩스	3.092482	0.152336	99.999111
22 004020	현대제철	3.126576	0.300965	99.999066
23 005490	POSCO홀딩스	3.207656	0.179439	99.999052
24 018670	SK가스	3.161570	0.365726	99.999037
25 036460	한국가스공사	3.083438	0.673967	99.999014
26 010060	OCI	3.354670	0.418636	99.998937
27 004000	롯데정밀화학	3.556533	0.000668	99.998925
28 383220	F&F	3.476459	0.441123	99.998876
29 001430	세아베스틸	3.636138	0.218204	99.998845
30 004800	효성	3.655593	0.316835	99.998817

## 1) PER+차입금비율 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
PER 1 이상, 차입금비율 0이상,  
낮은 순 30개 종목 선정

```
# 안정성지표 종목 선정 (PBR + 차입금비율 합성)
df_factor = pd.merge(pbr_list, ld_list, how='left', on=['code', 'name', 'exchange', 'period', 'mktcap'])
factor_list = ['PBR', 'D/E']
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # PBR+차입금비율, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	PBR	D/E	score
1 004170	신세계	0.112386	0.119755	99.999990
2 029780	삼성카드	0.126229	0.000000	99.999989
3 011170	롯데케미칼	0.117520	0.107647	99.999988
4 003550	LG	0.132706	0.001911	99.999986
5 006650	대한유화	0.134723	0.025555	99.999984
6 005490	POSCO홀딩스	0.118080	0.179439	99.999984
7 161390	한국타이어앤팩크놀로지	0.128108	0.089454	99.999984
8 002790	아모레G	0.136439	0.042340	99.999981
9 000720	현대건설	0.143751	0.034465	99.999979
10 030200	KT	0.143582	0.044937	99.999978
11 005380	현대차	0.117388	0.292862	99.999978
12 009540	한국조선해양	0.126809	0.230331	99.999977
13 011210	현대위아	0.118881	0.305892	99.999977
14 012330	현대모비스	0.145167	0.068919	99.999976
15 267270	현대건설기계	0.108750	0.427197	99.999975
16 280360	롯데제과	0.142406	0.121398	99.999975
17 007700	F&F홀딩스	0.148917	0.117903	99.999972
18 018670	SK가스	0.125308	0.365726	99.999970
19 017670	SK텔레콤	0.159237	0.072030	99.999969
20 097950	CJ제일제당	0.129005	0.354215	99.999969
21 036830	솔브레이인홀딩스	0.143742	0.227631	99.999969
22 028260	삼성물산	0.163145	0.054595	99.999968
23 181710	NHN	0.168573	0.032324	99.999967
24 034310	NICE	0.123571	0.488392	99.999965
25 036460	한국가스공사	0.103828	0.673967	99.999965
26 005250	녹십자홀딩스	0.154523	0.210129	99.999964
27 192400	쿠루홀딩스	0.181465	0.000350	99.999962
28 006040	동원산업	0.156952	0.236223	99.999962
29 020000	한섬	0.180580	0.034554	99.999961
30 001120	LX인터넷서널	0.146550	0.386508	99.999959

## 2) PBR+차입금비율 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
PBR 0.1 이상, 차입금비율 0이상,  
낮은 순 30개 종목 선정

---

## 04. 우량주 종목 추출 6) 성장률 지표

3년 간의 재무제표 데이터를 이용해야 하지만,  
4분기 재무제표 데이터만 존재하므로  
3년의 트레일링 데이터를 만들 수 없어 구하지 못하였다.

## 04. 우량주 종목 추출 7) 회전률 지표

```
# 부지런한 기업 (회전율 지표)
def getTurnover(self):
    # Turnover 계산 후 저장할 DataFrame 생성
    df_T0 = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 손익계산서, 재무상태표 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)
        bs_trailing = BALANCES().getTrailingBalances(code, self.period)

        # 손익계산서, 재무상태표 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or bs_trailing is None or code not in self.ohlcv.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. 총자산 회전율 계산
        # 평균 총자산 = (총자산 + 4분기 전 총자산) / 2
        # 총자산: balances > asset
        # 4분기 전 트레일링 데이터 존재하지 않으므로 현재 총자산만 계산에 이용
        AVG_Asset = float(bs_trailing['asset'] * self.fin_unit)

        # ASS_T0 = 매출액(incomes > rev) / AVG_Asset
        if AVG_Asset <= 0:
            ASS_T0 = np.NaN
        else:
            ASS_T0 = float((is_trailing['rev'] * self.fin_unit) / AVG_Asset)

        # 3. 매출채권 회전율 계산
        # 평균 매출채권 = (매출채권 + 4분기 전 매출채권) / 2
        # 매출채권: balances > rec
        # 4분기 전 트레일링 데이터 존재하지 않으므로 현재 매출채권만 계산에 이용
        AVG_Rec = float(bs_trailing['rec'] * self.fin_unit)

        # REC_T0 = 매출액(incomes > rev) / AVG_Receivables
        if AVG_Rec <= 0:
            REC_T0 = np.NaN
        else:
            REC_T0 = float((is_trailing['rev'] * self.fin_unit) / AVG_Rec)
```

```
# 4. 재고자산 회전율 계산
# 평균 재고자산 = (재고자산 + 4분기 전 재고자산) / 2
# 재고자산: balances > inv
# 4분기 전 트레일링 데이터 존재하지 않으므로 현재 재고자산만 계산에 이용
AVG_Inv = float(bs_trailing['inv'] * self.fin_unit)

# INV_T0 = 매출원가(incomes cgs) / 평균 재고자산
if AVG_Inv <= 0:
    INV_T0 = np.NaN
else:
    INV_T0 = float((is_trailing['cgs'] * self.fin_unit) / AVG_Inv)

# 3. 계산한 결과 DataFrame에 추가
df_T0 = df_T0.append({
    'code': code,
    'name': name,
    'exchange': exc,
    'period': self.period,
    'mktcap': self.ohlcv.loc[code]['mktcap'],
    'ASS_turnover': ASS_T0,
    'REC_turnover': REC_T0,
    'INV_turnover': INV_T0 }, ignore_index=True)

print(code, ": Calculate Turnover ... OK")

return df_T0
```

(총자산 + 4분기 전 총자산) / 2  
 (매출채권 + 4분기 전 매출채권) / 2  
 (재고자산 + 4분기 전 재고자산) / 2로 평균 값을 이용해야 하지만,  
 DB에 4분기 데이터만 존재하므로  
 이전 분기 트레일링 데이터를 구할 수 없다.  
 따라서 평균값 대신 현재 총자산, 매출채권, 재고자산만으로  
 회전률 지표를 계산했다.

## 04. 우량주 종목 추출 7) 회전률 지표

getTurnover() 실행 결과: 모든 종목에 대해 총자산회전율, 매출채권 회전율, 재고자산 회전율 계산

### 회전률 지표								
# Turnover 구하기								
to_list = b.getTurnover() to_list								
000200 : calculate turnover ... OK 038530 : Calculate Turnover ... OK 900280 : Calculate Turnover ... OK 215000 : Calculate Turnover ... OK 121440 : Calculate Turnover ... OK 183410 : Empty Incomes 183410 : Empty Balances 183410 : Not enough data to calculate ... 366030 : Calculate Turnover ... OK 076340 : Empty Incomes 076340 : Empty Balances 076340 : Not enough data to calculate ... 009290 : Calculate Turnover ... OK 014200 : Calculate Turnover ... OK 017040 : Calculate Turnover ... OK 029480 : Calculate Turnover ... OK 017900 : Calculate Turnover ... OK 037710 : Calculate Turnover ... OK 026910 : Calculate Turnover ... OK 355150 : Not enough data to calculate ... 007000 : Not enough data to calculate ...								
code	name	exchange	period	mktcap	ASS turnover	REC turnover	INV turnover	
0	095570	AJ네트웍스	KOSPI	2022/03	2.898300e+11	0.198893	3.003746	0.000000
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	0.186153	2.158318	1.951752
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	0.031175	1.319794	2.448134
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	0.287868	2.076579	1.490167
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	0.000000	0.000000	0.000000
...	...	...	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	0.148889	2.261761	0.879840
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	0.000000	0.000000	NaN
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	0.081981	3.490803	2.358013
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	0.314829	0.687308	NaN
2181	238490	힘스	KOSDAQ	2022/03	1.032807e+11	0.136974	1.915732	1.059098

## 04. 우량주 종목 추출 7) 회전률 지표

```
# Turnover 종목 선정
df_factor = to_list.copy()
factor_list = ['ASS turnover'] # 총자산 회전율
```

```
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # 총자산 회전율, 시가총액 상위 20%, 모든 종목 중 30개 선택
result
```

code	name	ASS turnover	score
1	047050	포스코인터내셔널	0.800875 100.000000
2	282330	BGF리테일	0.668493 83.470313
3	001120	LX인터내셔널	0.649872 81.145345
4	005610	SPC삼립	0.622233 77.694243
5	205470	휴마시스	0.538333 67.218097
6	298020	효성티엔씨	0.537908 67.165116
7	011070	LG이노텍	0.514506 64.242974
8	950130	엑세스바이오	0.512415 63.981886
9	091700	파트론	0.491963 61.428253
10	097520	엠씨넥스	0.491214 61.334730
11	049770	동원F&B	0.477799 59.659637
12	003070	코오롱글로벌	0.471305 58.848734
13	086280	현대글로비스	0.469469 58.619548
14	060250	NHN한국사이버결제	0.464780 58.034100
15	108320	LX세미콘	0.457643 57.142926
16	192650	드림텍	0.455083 56.823258
17	009240	한샘	0.451533 56.379930
18	248070	솔루엠	0.435943 54.433309
19	090460	비에이치	0.432419 53.993374
20	009900	명신산업	0.430065 53.699380
21	222800	삼택	0.422613 52.768969
22	106240	파인테크닉스	0.415069 51.827016
23	010950	S-Oil	0.414820 51.795912
24	105630	한세실업	0.390523 48.762004
25	008770	호텔신라	0.383870 47.931331
26	195870	해성디에스	0.380905 47.561179
27	007810	코리아써키트	0.378339 47.240693
28	022100	포스코 ICT	0.376703 47.036490
29	001440	대한전선	0.370573 46.270980
30	030000	제일기획	0.365503 45.638018

### 1) 총자산 회전율 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
총자산 회전율 0이상,  
높은 순 30개 종목 선정

```
# Turnover 종목 선정
df_factor = to_list.copy()
factor_list = ['ASS turnover'] # 총자산 회전율
```

```
result = stock_select(df_factor, 0.2, 'KOSPI', 30, factor_list) # 총자산 회전율, 시가총액 상위 20%, KOSPI 종목 중 30개 선택
result
```

code	name	ASS turnover	score
1	047050	포스코인터내셔널	0.800875 100.000000
2	282330	BGF리테일	0.668493 83.470313
3	001120	LX인터내셔널	0.649872 81.145345
4	005610	SPC삼립	0.622233 77.694243
5	298020	효성티엔씨	0.537908 67.165116
6	011070	LG이노텍	0.514506 64.242974
7	097520	엠씨넥스	0.491214 61.334730
8	049770	동원F&B	0.477799 59.659637
9	003070	코오롱글로벌	0.471305 58.848734
10	086280	현대글로비스	0.469469 58.619548
11	192650	드림텍	0.455083 56.823258
12	009240	한샘	0.451533 56.379930
13	248070	솔루엠	0.435943 54.433309
14	009000	명신산업	0.430065 53.699380
15	010950	S-Oil	0.414820 51.795912
16	105630	한세실업	0.390523 48.762004
17	008770	호텔신라	0.383870 47.931331
18	195870	해성디에스	0.380905 47.561179
19	007810	코리아써키트	0.378339 47.240693
20	001440	대한전선	0.370573 46.270980
21	030000	제일기획	0.365503 45.638018
22	066570	LG전자	0.364994 45.574373
23	383220	F&F	0.359352 44.869997
24	018260	삼성에스디에스	0.356150 44.470115
25	025860	남해화학	0.352224 43.979891
26	108670	LX하우시스	0.351901 43.939622
27	018670	SK가스	0.348007 43.453419
28	028050	삼성엔지니어링	0.346832 43.306715
29	286940	롯데정보통신	0.346712 43.291644
30	005390	신성통상	0.345551 43.146702

### 2) 총자산 회전율 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
총자산 회전율 0이상,  
높은 순 30개 종목 선정

## 04. 우량주 종목 추출 7) 회전률 지표

```
# Turnover 종목 선정
df_factor = to_list.copy()
factor_list = ['REC turnover'] # 매출채권 회전률

result = stock_select2(df_factor, 0.2, 'ALL', 30, factor_list) # 매출채권, 시가총액 상위 20%, 모든 종목 중 30개 선택
result
```

code	name	REC turnover	score
1 003690	코리안리	1.614743e+06	100.000000
2 000060	메리츠화재	6.602758e+04	4.089046
3 001450	현대해상	1.950733e+04	1.208077
4 005830	DB손해보험	1.029182e+04	0.637366
5 000810	삼성화재	2.293216e+03	0.142017
6 082640	동양생명	2.043524e+02	0.012655
7 032830	삼성생명	1.838877e+02	0.011388
8 023590	다우기술	1.558393e+02	0.009651
9 383800	LX플딩스	1.542008e+02	0.009550
10 088350	한화생명	5.115037e+01	0.003168
11 085620	미래에셋생명	4.336836e+01	0.002686
12 035250	강원랜드	2.493477e+01	0.001544
13 395400	SK리즈	2.262500e+01	0.001401
14 282330	BGF리테일	2.064580e+01	0.001279
15 032190	다오데이터	2.038294e+01	0.001262
16 016360	삼성증권	1.309630e+01	0.000811
17 034230	파라다이스	1.287226e+01	0.000797
18 382840	원준	1.261452e+01	0.000781
19 010140	삼성증권	1.232392e+01	0.000763
20 003530	한화투자증권	1.141997e+01	0.000707
21 025980	아남티	1.003548e+01	0.000621
22 032350	롯데관광개발	9.398107e+00	0.000582
23 007070	GS리테일	7.848414e+00	0.000486
24 214150	클레시스	7.241253e+00	0.000448
25 010620	현대미포조선	6.731161e+00	0.000417
26 086790	하나금융지주	6.268297e+00	0.000388
27 023530	롯데쇼핑	6.204793e+00	0.000384
28 114090	GKL	6.067289e+00	0.000376
29 008770	호텔신라	5.957799e+00	0.000369
30 138040	메리츠금융지주	5.708709e+00	0.000354

### 1) 매출채권 회전율 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
매출채권 회전율 0이상,  
높은 순 30개 종목 선정

```
# Turnover 종목 선정
df_factor = to_list.copy()
factor_list = ['REC turnover'] # 매출채권 회전률

result = stock_select2(df_factor, 0.2, 'KOSPI', 30, factor_list) # 매출채권, 시가총액 상위 20%, KOSPI 종목 중 30개 선택
result
```

code	name	REC turnover	score
1 003690	코리안리	1.614743e+06	100.000000
2 000060	메리츠화재	6.602758e+04	4.089046
3 001450	현대해상	1.950733e+04	1.208077
4 005830	DB손해보험	1.029182e+04	0.637366
5 000810	삼성화재	2.293216e+03	0.142017
6 082640	동양생명	2.043524e+02	0.012655
7 032830	삼성생명	1.838877e+02	0.011388
8 023590	다우기술	1.558393e+02	0.009651
9 383800	LX플딩스	1.542008e+02	0.009550
10 088350	한화생명	5.115037e+01	0.003168
11 085620	미래에셋생명	4.336836e+01	0.002686
12 035250	강원랜드	2.493477e+01	0.001544
13 395400	SK리즈	2.262500e+01	0.001401
14 282330	BGF리테일	2.064580e+01	0.001279
15 016360	삼성증권	1.309630e+01	0.000811
16 010140	삼성증권	1.232392e+01	0.000763
17 034230	파라다이스	1.287226e+01	0.000797
18 382840	원준	1.261452e+01	0.000781
19 010140	삼성증권	1.232392e+01	0.000763
20 003530	한화투자증권	1.141997e+01	0.000707
21 025980	아남티	1.003548e+01	0.000621
22 032350	롯데관광개발	9.398107e+00	0.000582
23 007070	GS리테일	7.848414e+00	0.000486
24 010620	현대미포조선	6.731161e+00	0.000417
25 086790	하나금융지주	6.268297e+00	0.000388
26 023530	롯데쇼핑	6.204793e+00	0.000384
27 114090	GKL	6.067289e+00	0.000376
28 008770	호텔신라	5.957799e+00	0.000369
29 138040	메리츠금융지주	5.708709e+00	0.000354
30 009240	한샘	5.399066e+00	0.000334
31 139480	이마트	5.334802e+00	0.000330
32 180640	한진칼	5.253684e+00	0.000325
33 210980	SK디앤디	4.968219e+00	0.000308
34 028670	판오션	4.949152e+00	0.000306

### 2) 매출채권 회전율 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
매출채권 회전율 0이상,  
높은 순 30개 종목 선정

## 04. 우량주 종목 추출 7) 회전률 지표

```
# Turnover 종목 선정
df_factor = to_list.copy()
factor_list = ['INV turnover'] # 재고자산 회전율

result = stock_select2(df_factor, 0.2, 'ALL', 30, factor_list) # 재고자산, 시가총액 상위 20%, 모든 종목 중 30개 선택
result
```

code	name	INV turnover	score
1 023590	다우기술	30659.756250	100.000000
2 088350	한화생명	2252.750238	7.347580
3 032190	다우데이타	316.941266	1.033737
4 005830	DB손해보험	276.409822	0.901540
5 017390	서울가스	222.782121	0.726627
6 307950	현대오로에버	124.445747	0.405893
7 016710	대성플링스	112.883663	0.368182
8 000120	CJ대한통운	98.280595	0.320552
9 018260	삼성에스디에스	88.035079	0.287136
10 034120	SBS	85.203550	0.277900
11 035250	강원랜드	64.406229	0.210068
12 022100	포스코 ICT	60.972780	0.198869
13 180640	한진칼	49.715072	0.162151
14 060250	NHN한국사이버결제	44.029262	0.143606
15 035600	KG이니시스	29.680548	0.096806
16 286940	롯데정보통신	22.223743	0.072485
17 035900	JYP Ent.	19.893698	0.064885
18 012510	더존비즈온	19.462585	0.063479
19 363280	티와이홀딩스	18.298985	0.059684
20 114090	GKL	17.634796	0.057518
21 034230	파라다이스	16.559646	0.054011
22 012750	에스원	14.980814	0.048861
23 030000	제일기획	14.343999	0.046784
24 003550	LG	13.316911	0.043434
25 051600	한진KPS	12.151819	0.039634
26 028670	팬오션	11.352535	0.037027
27 282330	BGF리테일	11.330165	0.036955
28 001450	현대해상	11.049672	0.036040
29 078340	컴투스	10.194475	0.033250
30 089590	제주항공	9.893254	0.032268

### 1) 재고자산 회전율 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
재고자산 회전율 0이상,  
높은 순 30개 종목 선정

```
# Turnover 종목 선정
df_factor = to_list.copy()
factor_list = ['INV turnover'] # 재고자산 회전율

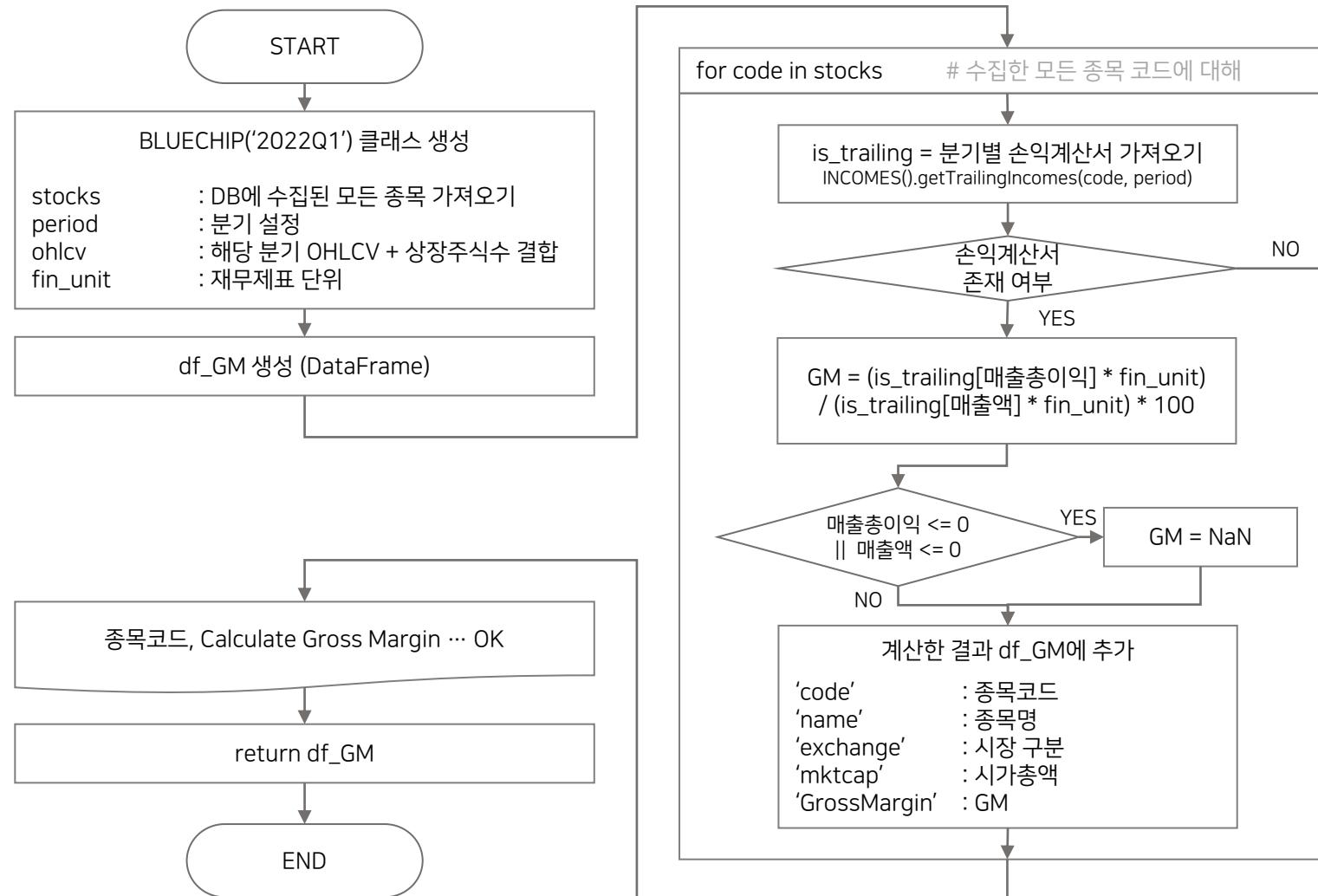
result = stock_select2(df_factor, 0.2, 'KOSPI', 30, factor_list) # 재고자산, 시가총액 상위 20%, KOSPI 종목 중 30개 선택
result
```

code	name	INV turnover	score
1 023590	다우기술	30659.756250	100.000000
2 088350	한화생명	2252.750238	7.347580
3 005830	DB손해보험	276.409822	0.901540
4 017390	서울가스	222.782121	0.726627
5 307950	현대오토에버	124.445747	0.405893
6 016710	대성플링스	112.883663	0.368182
7 000120	CJ대한통운	98.280595	0.320552
8 018260	삼성에스디에스	88.035079	0.287136
9 034120	SBS	85.203550	0.277900
10 035250	강원랜드	64.406229	0.210068
11 180640	한진칼	49.715072	0.162151
12 286940	롯데정보통신	22.223743	0.072485
13 012510	더존비즈온	19.462585	0.063479
14 363280	티와이홀딩스	18.298985	0.059684
15 114090	GKL	17.634796	0.057518
16 012750	에스원	14.980814	0.048861
17 030000	제일기획	14.343999	0.046784
18 003550	LG	13.316911	0.043434
19 051600	한진KPS	12.151819	0.039634
20 028670	팬오션	11.332535	0.037027
21 282330	BGF리테일	11.330165	0.036955
22 001450	현대해상	11.049672	0.036040
23 089590	제주항공	9.893254	0.032268
24 007070	GS리테일	9.517508	0.031042
25 003070	코오롱글로벌	6.848035	0.022336
26 017800	현대엘리베이	6.399009	0.020871
27 011200	HMM	6.134847	0.020009
28 000720	현대건설	5.357044	0.017473
29 020560	아시아나항공	5.180745	0.016898
30 047050	포스코인터내셔널	5.002246	0.016315

### 2) 재고자산 회전율 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
재고자산 회전율 0이상,  
높은 순 30개 종목 선정

## 04. 우량주 종목 추출 8) 이익률 지표 (해자가 있는 기업)



## 04. 우량주 종목 추출 8) 이익률 지표 (해자가 있는 기업)

```
# 해자가 있는 기업 (이익률 지표)
def getGrossMargin(self):
    # Gross Margin 계산 후 저장할 DataFrame 생성
    df_GM = pd.DataFrame()

    # 모든 종목 코드에 대해
    for code, name, exc in zip(self.stocks['code'], self.stocks['name'], self.stocks['exchange']):
        # 1. 분기별 손익계산서 가져오기
        is_trailing = INCOMES().getTrailingIncomes(code, self.period)

        # 손익계산서 trailing 데이터 만들지 못하는 경우
        # 또는 ohlcv 데이터가 존재하지 않는 종목인 경우
        if is_trailing is None or code not in self.ohlcv.index:
            print(code, ": Not enough data to calculate ...")
            continue # 해당 종목은 pass

        # 2. 매출총이익(gross margin) = (매출총이익/매출액)*100 계산
        # 매출총이익: incomes > gross
        # 매출액: incomes > rev
        if float(is_trailing['gross']) <= 0 or float(is_trailing['rev']) <= 0: # 매출총이익 음수 or 매출액 음수인 경우
            GM = np.NaN
        else:
            GM = float((is_trailing['gross']*self.fin_unit) / (is_trailing['rev']*self.fin_unit) * 100)

        # 3. 계산한 결과 DataFrame에 추가
        df_GM = df_GM.append({
            'code':code,
            'name':name,
            'exchange':exc,
            'period':self.period,
            'mktcap':self.ohlcv.loc[code]['mktcap'],
            'GrossMargin':GM }, ignore_index=True)

    print(code, ": Calculate Gross Margin ... OK")

return df_GM
```

## 04. 우량주 종목 추출 8) 이익률 지표 (해자가 있는 기업)

getGrossMargin() 실행 결과: 모든 종목에 대해 매출총이익률 값 계산

#### 이익률 지표

```
# GrossMargin 구하기
gm_list = b.getGrossMargin()
gm_list

INFO : Calculate Gross Margin ... OK
291650 : Calculate Gross Margin ... OK
293780 : Calculate Gross Margin ... OK
Empty DB Incomes
267810 : Not enough data to calculate ...
018250 : Calculate Gross Margin ... OK
161000 : Calculate Gross Margin ... OK
196300 : Calculate Gross Margin ... OK
310200 : Calculate Gross Margin ... OK
179530 : Not enough data to calculate ...
900100 : Calculate Gross Margin ... OK
205500 : Calculate Gross Margin ... OK
052790 : Calculate Gross Margin ... OK
290740 : Calculate Gross Margin ... OK
238090 : Calculate Gross Margin ... OK
092600 : Calculate Gross Margin ... OK
129890 : Calculate Gross Margin ... OK
174900 : Calculate Gross Margin ... OK
255440 : Calculate Gross Margin ... OK
060230 : Calculate Gross Margin ... OK
```

```
# GrossMargin 구하기
gm_list = b.getGrossMargin()
gm_list
```

	code	name	exchange	period	mktcap	GrossMargin
0	095570	AJ네트웍스	KOSPI	2022/03	2.398300e+11	100.000000
1	006840	AK홀딩스	KOSPI	2022/03	2.907840e+11	15.372372
2	054620	APS홀딩스	KOSDAQ	2022/03	2.488095e+11	49.319104
3	265520	AP시스템	KOSDAQ	2022/03	3.629337e+11	23.556013
4	211270	AP위성	KOSDAQ	2022/03	2.307593e+11	NaN
	...	...	...	...	...	...
2177	189980	흥국에프엔비	KOSDAQ	2022/03	1.557913e+11	34.403979
2178	000540	흥국화재	KOSPI	2022/03	2.601827e+11	NaN
2179	003280	흥아해운	KOSPI	2022/03	6.996365e+11	8.957735
2180	037440	희림	KOSDAQ	2022/03	1.322635e+11	11.409652
2181	238490	힐스	KOSDAQ	2022/03	1.032807e+11	23.145788

## 04. 우량주 종목 추출 8) 이익률 지표 (해자가 있는 기업)

```
# GrossMargin 종목 선정
df_factor = gm_list.copy()
factor_list = ['GrossMargin']
result = stock_select(df_factor, 0.2, 'ALL', 30, factor_list) # 이익률지표, 시가총액 상위 20%, 전체 종목 중 30개 선택
result
```

code	name	GrossMargin	score
1 097230	HJ증권업	0.406900	100.000000
2 068240	다원시스	0.486916	99.919658
3 082640	동양생명	1.257922	99.145501
4 032830	삼성생명	1.534628	98.867665
5 085620	미래에셋생명	2.049218	98.350973
6 047050	포스코인터내셔널	3.410383	96.984247
7 003470	유안타증권	3.869809	96.522944
8 138930	BNK금융지주	4.103920	96.287877
9 138040	메리츠금융지주	4.386256	96.004387
10 000670	영풍	4.501193	95.888981
11 088350	한화생명	4.938273	95.450115
12 006650	대한유화	5.238044	95.149120
13 036460	한국가스공사	5.355515	95.031169
14 175330	JB금융지주	5.734038	94.651100
15 018670	SK가스	6.025182	94.358766
16 298000	효성화학	6.054780	94.329048
17 006120	SK디스커버리	6.063807	94.319984
18 105560	KB금융	6.288422	94.094451
19 001440	대한전선	6.355871	94.026726
20 016360	삼성증권	6.554926	93.826858
21 011210	현대위아	6.902600	93.477763
22 086280	현대글로비스	7.534949	92.842831
23 003540	대신증권	7.820311	92.556304
24 286940	롯데정보통신	8.045892	92.329801
25 086790	하나금융지주	8.096770	92.278716
26 001120	LX인터내셔널	8.458279	91.915729
27 097520	엠씨넥스	8.575347	91.798184
28 100090	삼강업엔티	8.642441	91.730815
29 064350	현대로템	8.646517	91.726723
30 316140	우리금융지주	8.686665	91.686411

### 1) 매출총이익률 기준 종목 선정 결과

시가총액 상위 20% 모든 종목에 대해  
GrossMargin 0이상,  
낮은 순 30개 종목 선정

```
# GrossMargin 종목 선정
df_factor = gm_list.copy()
factor_list = ['GrossMargin']
result = stock_select(df_factor, 0.2, 'KOSPI', 30, factor_list) # 이익률지표, 시가총액 상위 20%, KOSPI 종목 중 30개 선택
result
```

code	name	GrossMargin	score
1 097230	HJ증권업	0.406900	100.000000
2 082640	동양생명	1.257922	99.145501
3 032830	삼성생명	1.534628	98.867665
4 085620	미래에셋생명	2.049218	98.350973
5 047050	포스코인터내셔널	3.410383	96.984247
6 003470	유안타증권	3.869809	96.522944
7 138930	BNK금융지주	4.103920	96.287877
8 138040	메리츠금융지주	4.386256	96.004387
9 000670	영풍	4.501193	95.888981
10 088350	한화생명	4.938273	95.450115
11 006650	대한유화	5.238044	95.149120
12 036460	한국가스공사	5.355515	95.031169
13 175330	JB금융지주	5.734038	94.651100
14 018670	SK가스	6.025182	94.358766
15 298000	효성화학	6.054780	94.329048
16 006120	SK디스커버리	6.063807	94.319984
17 105560	KB금융	6.288422	94.094451
18 001440	대한전선	6.355871	94.026726
19 016360	삼성증권	6.554926	93.826858
20 011210	현대위아	6.902600	93.477763
21 086280	현대글로비스	7.534949	92.842831
22 003540	대신증권	7.820311	92.556304
23 286940	롯데정보통신	8.045892	92.329801
24 086790	하나금융지주	8.096770	92.278716
25 001120	LX인터내셔널	8.458279	91.915729
26 097520	엠씨넥스	8.575347	91.798184
27 064350	현대로템	8.646517	91.726723
28 316140	우리금융지주	8.686665	91.686411
29 294870	HDC현대산업개발	8.714558	91.658403
30 008730	율촌화학	8.779793	91.592902

### 2) 매출총이익률 기준 종목 선정 결과

시가총액 상위 20% KOSPI 종목에 대해  
GrossMargin 0이상,  
낮은 순 30개 종목 선정

---

05

## 종목 추출 함수

---

## 03. 가치주 종목 추출 | 종목 추출 함수

```
def stock_select(df_factor, mktcap_top, exchange, n, factor_list):
    basic_list = ['code', 'name', 'exchange', 'period', 'mktcap'] # 종목 코드, 이름, 주식 시장, 기간, 시가총액
    basic_list.extend(factor_list) # basic_list에 factor_list 요소 넣기

    df_select = df_factor.copy() # df_select에 df_factor 복사하기
    df_select = df_select[basic_list] # df_select에 basic_list 요소 넣기

    df_select['score'] = 0 # df_select에 score 항목 추가 & 0 초기화

    # 시가총액 기준 내림차순 정렬 후, 상위 mktcap_top% 산출
    df_select = df_select.sort_values(by=['mktcap'], ascending=False).head(int(len(df_select) * mktcap_top))
    df_select = df_select.dropna()

    # 해당 exchange 종목만 골라내기
    if exchange.upper() != 'ALL':
        df_select = df_select[df_select['exchange'] == exchange]

    # 최저값 이상 종목 추출
    floor = 0.1
    df_range = df_select.copy()

    for i in range(len(factor_list)):
        # 슬기로운 컨트 투자 기준, 각 factor 최저값 설정
        if factor_list[i] == 'PER': # PER = 1
            floor = 1
        elif factor_list[i] == 'PBR' or factor_list[i] == 'PSR' or factor_list[i] == 'PCR': # PBR, PSR, PCR = 0.1
            floor = 0.1
        else: # 나머지 = 0
            floor = 0

        # 최저값 이상 종목만 추출
        df_range = df_range[df_range[factor_list[i]] >= floor]

    df_select = df_range.copy()

    return df_select
```

```
# 팩터간의 점수 계산
sort_desc = ['NCAV', 'ROA', 'ROE', 'GP/A', 'ASS turnover', 'REC turnover', 'INV turnover'] # 절수를 높은 점수 팩터 리스트

for i in range(len(factor_list)):
    if factor_list[i] in sort_desc: # 절수를 높은 점수 팩터인 경우
        # (factor 값 - factor 최소값) / factor 최대값
        df_select[factor_list[i] + '_score'] = (df_select[factor_list[i]] - min(df_select[factor_list[i]])) / max(df_select[factor_list[i]] - min(df_select[factor_list[i]]))

    else: # 절수를 높은 점수
        # (factor 값 - factor 최대값) / factor 최소값
        df_select[factor_list[i] + '_score'] = (df_select[factor_list[i]] - max(df_select[factor_list[i]])) / min(df_select[factor_list[i]] - max(df_select[factor_list[i]]))

    # 점수 = 위 계산 값 / factor 개수 * 100 > (0~100점으로 맞추기)
    df_select['score'] += (df_select[factor_list[i] + '_score'] / len(factor_list)) * 100

# 상위 n개 종목 추출
df_select = df_select.sort_values(by=['score'], ascending=False).head(n)

# 종목 선택
stock_select = df_select[['code', 'name']]

# 각 팩터 점수 추가
for i in range(len(factor_list)):
    stock_select[factor_list[i]] = df_select[factor_list[i]]

# 점수 합계
stock_select['score'] = df_select['score']

stock_select.reset_index(drop=True, inplace=True) # index 재설정
stock_select.index = stock_select.index + 1 # index 1부터 시작 (~30)

return stock_select
```

---

감사합니다

---