In [1]:

```
#!pip install finance-datareader
#!pip install yfinance
#!pip install pandas_datareader
```

In [2]:

```
import pandas as pd
import yfinance as yf
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import FinanceDataReader as fdr
from pandas_datareader import data as pdr
%matplotlib inline
```

국내 상장 ETF 데이터 분석

1. ETF 전 종목 기본 정보 불러오기

In [3]:

```
etf = pd.read_csv('./data_0319_20220307.csv',encoding='UTF-8')
etf.head(20)
# len(df) :547
```

0	KR7292340007	292340	티 200커버드콜ATM레버리지증권상장지수투자신탁[주식-파생형]	마이티200커버드콜ATM레버리지	DB Mighty KOSPI200 Covered Call ATM Leverage ETF	2018/03/20	코스피 200 커버드콜 ATM 지수	KRX	2X 레버리지 (2)	실물	국내	주식
1	KR7159800002	159800	DB마이티K100증권상장지수투자신탁(주식)	마이티코스피100	DB Mighty K100 ETF	2012/07/05	코스피 100	KRX	일반 (1)	실물	국내	주식

In [4]:

```
etf['기초자산분류'].value_counts()
```

Out[4]:

```
주식      429
채권       60
원자재      19
혼합자산     16
통화       11
부동산       7
기타        5
Name: 기초자산분류, dtype: int64
```

In [5]:

```
etf['기초시장분류'].value_counts()
```

Out[5]:

```
국내       372
해외       164
국내&해외     11
Name: 기초시장분류, dtype: int64
```

2. 종목별 1년간 주가 정보 불러오기

[etf 기본정보 테이블에서 단축코드 추출 -> 종목별 주가 정보 얻기]

In [6]:

```
code = etf['단축코드'].values
code
```

Out[6]:

```
array([292340, 159800, 361580, 285000, 287300, 287310, 290080, 284980,
       287320, 287330, 252400, 252420, 252410, 284990, 285010, 148020,
       285020, 315480, 105780, 290130, 368200, 367760, 367770, 388280,
       326240, 385560, 385550, 385540, 270800, 307010, 319870, 292050,
       403990, 234310, 241390, 401170, 300640, 266160, 282000, 114100,
       295020, 295000, 397410, 397420, 276650, 375270, 411720, 417450,
       399580, 336160, 326230, 272560, 196230, 315960, 252730, 252720,
       379780, 219390, 354240, 368590, 267490, 267500, 267450, 267440,
       388420, 140570, 140580, 379790, 183710, 310080, 174360, 136340,
       281990, 272570, 250730, 291680, 371150, 183700, 278240, 275750,
       270810, 361590, 302450, 334700, 334690, 253280, 253290, 225130,
       407310, 332930, 304780, 306520, 293180, 395290, 395280, 368190,
       402460, 395270, 367740, 407300, 381560, 381570, 346000, 304760,
       404470, 332940, 322400, 322410, 354350, 401590, 314700, 390950,
       419170, 306530, 304770, 375760, 140950, 152870, 192720, 176710,
       403790, 137930, 407160, 407170, 310960, 227550, 227560, 102110,
       243880, 252000, 267770, 252710, 243890, 315270, 289480, 227540,
       305540, 365040, 357870, 400970, 396500, 377990, 417630, 341850,
```

In [7]:

```
#FinanceDataReader 통해 전 종목 주가 불러오기 -> 딕셔너리 저장
etf_original ={}
for c in code.tolist():
    etf_original[c] = fdr.DataReader(symbol=str(c), start='2021-03-07')
```

In [8]:

```
etf_original
```

Out[8]:

```
{292340:              Open   High    Low  Close  Volume    Change
 Date
 2021-03-08  10090  10090   9840   9840       4 -0.004552
 2021-03-09   9650   9650   9650   9650      20 -0.019309
 2021-03-10   9845   9845   9805   9805     150  0.016062
 2021-03-11   9710  10020   9710  10020     177  0.021928
 2021-03-12  10095  10145  10095  10145       7  0.012475
 ...           ...    ...    ...    ...     ...       ...
 2022-03-03   9445   9445   9445   9445       0  0.031677
 2022-03-04   9200   9200   9195   9195       3 -0.026469
 2022-03-07   8835   8835   8835   8835     115 -0.039152
 2022-03-08   8650   8650   8650   8650       1 -0.020939
 2022-03-10   8650   8650   8650   8650       0  0.000000

 [250 rows x 6 columns],
 159800:              Open   High    Low  Close  Volume    Change
 Date
 2021-03-08  31674  31674  31674  31674      20  0.000000
```

[코스피, 코스닥 지수]

In [9]:

```

# kospi = fdr.DataReader('IXIC', '2019-03-07').rename(columns={'Close':'Kospi'})
# kosdaq = fdr.DataReader(symbol='KQ11', start='2019-03-07').rename(columns={'Close':'Kosdaq'})
# yf.download("KOS11", start="20121-03-07")
```

In [10]:

```
yf.pdr_override()
kospi = pdr.get_data_yahoo("^KS11", start="2021-03-07").rename(columns={'Close':'Kospi'})
kosdaq = pdr.get_data_yahoo("^KQ11", start="2021-03-07").rename(columns={'Close':'Kosdaq'})
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

In [11]:

```
kospi
```

Out[11]:

	Open	High	Low	Kospi	Adj Close	Volume
Date						
2021-03-08	3031.989990	3055.649902	2992.639893	2996.110107	2996.110107	1928300
2021-03-09	2989.959961	3000.489990	2929.360107	2976.120117	2976.120117	1534200
2021-03-10	2980.760010	3013.949951	2951.530029	2958.120117	2958.120117	905600
2021-03-11	2964.300049	3028.370117	2964.300049	3013.699951	3013.699951	1349200
2021-03-12	3030.729980	3061.429932	3030.729980	3054.389893	3054.389893	1669100
...	...	...	...	...	...	...
2022-03-03	2729.860107	2748.209961	2726.350098	2747.080078	2747.080078	614300
2022-03-04	2736.580078	2736.580078	2702.340088	2713.429932	2713.429932	765300
2022-03-07	2680.169922	2680.169922	2644.100098	2651.310059	2651.310059	571300
2022-03-08	2617.330078	2647.179932	2605.810059	2622.399902	2622.399902	540100
2022-03-10	2660.860107	2682.790039	2660.860107	2678.629883	2678.629883	602991

249 rows × 6 columns

3. 전처리

[종목별 주가 지수 데이터프레임 만들기 (종가 기준)]

In [12]:

```
etf_df = pd.concat(etf_original, axis=1)
etf_df
```

Out[12]:

	292340						159800				...	39167
	Open	High	Low	Close	Volume	Change	Open	High	Low	Close	...	Low
2021-03-08	10090	10090	9840	9840	4	-0.004552	31674	31674	31674	31674	...	NaN
2021-03-09	9650	9650	9650	9650	20	-0.019309	31221	31221	31221	31221	...	NaN
2021-03-10	9845	9845	9805	9805	150	0.016062	31615	31615	31221	31221	...	NaN
2021-03-11	9710	10020	9710	10020	177	0.021928	31492	31492	31492	31492	...	NaN
2021-03-12	10095	10145	10095	10145	7	0.012475	31797	32073	31797	31940	...	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
2022-03-03	9445	9445	9445	9445	0	0.031677	27885	27885	27885	27885	...	8765.0
2022-03-04	9200	9200	9195	9195	3	-0.026469	28030	28030	28030	28030	...	8635.0
2022-03-07	8835	8835	8835	8835	115	-0.039152	28030	28030	28030	28030	...	8485.0
2022-03-08	8650	8650	8650	8650	1	-0.020939	27005	27125	27005	27125	...	8395.0
2022-03-10	8650	8650	8650	8650	0	0.000000	27125	27125	27125	27125	...	8600.0

250 rows × 3228 columns

In [13]:

```
etf_df.columns
```

Out[13]:

```
MultiIndex([(292340,   'Open'),
            (292340,   'High'),
            (292340,    'Low'),
            (292340,  'Close'),
            (292340, 'Volume'),
            (292340, 'Change'),
            (159800,   'Open'),
            (159800,   'High'),
            (159800,    'Low'),
            (159800,  'Close'),
            ...
            (391670,    'Low'),
            (391670,  'Close'),
            (391670, 'Volume'),
            (391670, 'Change'),
            (391680,   'Open'),
            (391680,   'High'),
            (391680,    'Low'),
            (391680,  'Close'),
            (391680, 'Volume'),
            (391680, 'Change')],
           length=3228)
```

In [14]:

```
etf_close_list = list(filter(lambda x: 'Close' in x, etf_df.columns))
print(etf_close_list)
```

```
[(292340, 'Close'), (159800, 'Close'), (361580, 'Close'), (285000, 'Close'), (2873
00, 'Close'), (287310, 'Close'), (290080, 'Close'), (284980, 'Close'), (287320, 'C
lose'), (287330, 'Close'), (252400, 'Close'), (252420, 'Close'), (252410, 'Clos
e'), (284990, 'Close'), (285010, 'Close'), (148020, 'Close'), (285020, 'Close'),
(315480, 'Close'), (105780, 'Close'), (290130, 'Close'), (368200, 'Close'), (36776
0, 'Close'), (367770, 'Close'), (388280, 'Close'), (326240, 'Close'), (385560, 'Cl
ose'), (385550, 'Close'), (385540, 'Close'), (270800, 'Close'), (307010, 'Close'),
(319870, 'Close'), (292050, 'Close'), (403990, 'Close'), (234310, 'Close'), (24139
0, 'Close'), (401170, 'Close'), (300640, 'Close'), (266160, 'Close'), (282000, 'Cl
ose'), (114100, 'Close'), (295020, 'Close'), (295000, 'Close'), (397410, 'Close'),
(397420, 'Close'), (276650, 'Close'), (375270, 'Close'), (411720, 'Close'), (41745
0, 'Close'), (399580, 'Close'), (336160, 'Close'), (326230, 'Close'), (272560, 'Cl
ose'), (196230, 'Close'), (315960, 'Close'), (252730, 'Close'), (252720, 'Close'),
(379780, 'Close'), (219390, 'Close'), (354240, 'Close'), (368590, 'Close'), (26749
0, 'Close'), (267500, 'Close'), (267450, 'Close'), (267440, 'Close'), (388420, 'Cl
ose'), (140570, 'Close'), (140580, 'Close'), (379790, 'Close'), (183710, 'Close'),
(310080, 'Close'), (174360, 'Close'), (136340, 'Close'), (281990, 'Close'), (27257
0, 'Close'), (250730, 'Close'), (291680, 'Close'), (371150, 'Close'), (183700, 'Cl
ose'), (278240, 'Close'), (275750, 'Close'), (270810, 'Close'), (361590, 'Close'),
```

In [15]:

```
new_column_name = list(map(lambda x: x[0], etf_close_list))
print(new_column_name)
```

```
[292340, 159800, 361580, 285000, 287300, 287310, 290080, 284980, 287320, 287330, 252
400, 252420, 252410, 284990, 285010, 148020, 285020, 315480, 105780, 290130, 368200,
367760, 367770, 388280, 326240, 385560, 385550, 385540, 270800, 307010, 319870, 2920
50, 403990, 234310, 241390, 401170, 300640, 266160, 282000, 114100, 295020, 295000,
397410, 397420, 276650, 375270, 411720, 417450, 399580, 336160, 326230, 272560, 1962
30, 315960, 252730, 252720, 379780, 219390, 354240, 368590, 267490, 267500, 267450,
267440, 388420, 140570, 140580, 379790, 183710, 310080, 174360, 136340, 281990, 2725
70, 250730, 291680, 371150, 183700, 278240, 275750, 270810, 361590, 302450, 334700,
334690, 253280, 253290, 225130, 407310, 332930, 304780, 306520, 293180, 395290, 3952
80, 368190, 402460, 395270, 367740, 407300, 381560, 381570, 346000, 304760, 404470,
332940, 322400, 322410, 354350, 401590, 314700, 390950, 419170, 306530, 304770, 3757
60, 140950, 152870, 192720, 176710, 403790, 137930, 407160, 407170, 310960, 227550,
227560, 102110, 243880, 252000, 267770, 252710, 243890, 315270, 289480, 227540, 3055
40, 365040, 357870, 400970, 396500, 377990, 417630, 341850, 412560, 364960, 412570,
364980, 292160, 364990, 404540, 364970, 365000, 300610, 138530, 289260, 289250, 3109
70, 143850, 269370, 292150, 139280, 237440, 319640, 160580, 114820, 387270, 412770,
418670, 371450, 248270, 139310, 139320, 133690, 137610, 272580, 157450, 105010, 1233
20, 147970, 360750, 418660, 245340, 329750, 261110, 261120, 305080, 381170, 381180,
228810, 329200, 138520, 157490, 228800, 227570, 261140, 130680, 245350, 307510, 2363
50, 123310, 241180, 292560, 248260, 150460, 302190, 157500, 307520, 117690, 245360,
414780, 371470, 396520, 371460, 396510, 371160, 166400, 233160, 232080, 261060, 2610
70, 250780, 277640, 277650, 277630, 376410, 387280, 143860, 138540, 228790, 394670,
394660, 276000, 139260, 139220, 139290, 139270, 139250, 139230, 139240, 228820, 2250
60, 182480, 225040, 225030, 217790, 203780, 182490, 174350, 275980, 211560, 217770,
225050, 195930, 195920, 192090, 204480, 217780, 210780, 285690, 292730, 402520, 2785
30, 226980, 337160, 363580, 337150, 223190, 237350, 252650, 360140, 252670, 305720,
271060, 368680, 395170, 337120, 325010, 395160, 395150, 266370, 101280, 401470, 3855
20, 385510, 373490, 306950, 292190, 404260, 229720, 289040, 156080, 278540, 251350,
275280, 275290, 275300, 269420, 291890, 329650, 329660, 329670, 352540, 315930, 2710
50, 261220, 300950, 266390, 279530, 280940, 132030, 138910, 114260, 292770, 276990,
102960, 352560, 273140, 244620, 314250, 379800, 276970, 379810, 409820, 409810, 3049
40, 261250, 261260, 261270, 261240, 280930, 411420, 390390, 390400, 218420, 308620,
304670, 304660, 266360, 244580, 325020, 211900, 237370, 244670, 102780, 400570, 1446
00, 363570, 273130, 102970, 283580, 169950, 204450, 415340, 256750, 372330, 279540,
229200, 233740, 360150, 251340, 226490, 359210, 337140, 138920, 244660, 375770, 2664
10, 298770, 266420, 364690, 284430, 321410, 219480, 176950, 152380, 117700, 214980,
153130, 122630, 185680, 200030, 140700, 213610, 117460, 140710, 114800, 117680, 4115
40, 295040, 363510, 292500, 404650, 208470, 400590, 399110, 400580, 220130, 415760,
413220, 407830, 407820, 413930, 108590, 145850, 153270, 167860, 148070, 294400, 2532
50, 253230, 253240, 331910, 100910, 200250, 104530, 114470, 130730, 411860, 394340,
394350, 139660, 138230, 225800, 230480, 373790, 104520, 291630, 291620, 316670, 1222
60, 419890, 385710, 385720, 410870, 404120, 168300, 332500, 385590, 380340, 414270,
368470, 356540, 411060, 277540, 114460, 365780, 299070, 299080, 411050, 181480, 1906
20, 265690, 152500, 291130, 342140, 280320, 360200, 309230, 402970, 367380, 391590,
391600, 143460, 245710, 131890, 108450, 226380, 322130, 272220, 272230, 322120, 3221
50, 316300, 256440, 145670, 196030, 238720, 205720, 416090, 168580, 272910, 371870,
385600, 354500, 251890, 305050, 261920, 105190, 371130, 219900, 152100, 295820, 2531
50, 253160, 395750, 395760, 278420, 227830, 122090, 292750, 309210, 333940, 333950,
333960, 333970, 333980, 269540, 269530, 251590, 161510, 251600, 289670, 298340, 4196
50, 415920, 278620, 287180, 332610, 332620, 238670, 373530, 256450, 239660, 280920,
266550, 301410, 301400, 328370, 301440, 376250, 213630, 189400, 195970, 195980, 2156
20, 391670, 391680]
```

In [16]:

```
etf_close_price_df = pd.concat(list(map(lambda x:etf_df[x], etf_close_list)), axis=1)
etf_close_price_df
```

Out[16]:

```
            292340  159800  361580  285000  287300  287310  290080  284980  287320  287330  ...  328370  3014
             Close   Close   Close   Close   Close   Close   Close   Close   Close   Close  ...   Close  Clos
2021-03-08    9840   31674   20890   16852   10665   11544    7090    8149   10698    8622  ...   15000   111
2021-03-09    9650   31221   20760   16763   10729   11564    7153    8333   10473    8632  ...   14860   110
2021-03-10    9805   31221   20645   16580   10557   11377    7081    8246   10568    8627  ...   14825   110
2021-03-11   10020   31492   21080   17095   10813   11465    7144    8309   10743    8676  ...   15100   111
2021-03-12   10145   31940   21250   17367   11108   11668    7149    8250   10937    8730  ...   15310   113
...            ...     ...     ...     ...     ...     ...     ...     ...     ...     ...  ...     ...
```

In [17]:

```
etf_close_price_df.columns = new_column_name
etf_close_price_df = pd.concat([etf_close_price_df, kospi['Kospi'], kosdaq['Kosdaq']], axis=1)
etf_close_price_df
```

Out[17]:

```
            292340  159800  361580  285000  287300  287310  290080  284980  287320  287330  ...  376250  2136
2021-03-08    9840   31674   20890   16852   10665   11544    7090    8149   10698    8622  ...    9710   145
2021-03-09    9650   31221   20760   16763   10729   11564    7153    8333   10473    8632  ...    9670   149
2021-03-10    9805   31221   20645   16580   10557   11377    7081    8246   10568    8627  ...    9605   147
2021-03-11   10020   31492   21080   17095   10813   11465    7144    8309   10743    8676  ...    9795   148
2021-03-12   10145   31940   21250   17367   11108   11668    7149    8250   10937    8730  ...    9915   149
...            ...     ...     ...     ...     ...     ...     ...     ...     ...     ...  ...     ...
2022-         9445   27885   19210   15350   12315   10055    8255    9200   11470    7555  ...    9050   164
```

[수익률 데이터프레임 생성 및 결측치 처리]

In [18]:

```
rows = len(etf_close_price_df.index)
columns = etf_close_price_df.columns
cleaned_etf_df = etf_close_price_df.copy()
```

기준: 1년 전(2021년 3월 7일) 또는 2021년 3월 7일 이후 상장된 경우 상장일

In [19]:

```
#수익률 데이터 프레임 생성
for column in columns:
    base = etf_close_price_df.isna()[column].values.tolist().index(False)
    for i in range(base+1, rows):
        cleaned_etf_df[column].iloc[i] = (etf_close_price_df[column].iloc[i]/etf_close_price_df[colu
```

```
C:\Users\YJ\anaconda3\envs\test\lib\site-packages\pandas\core\indexing.py:1732: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  self._setitem_single_block(indexer, value, name)
```

In [20]:

```
cleaned_etf_df
```

Out[20]:

	292340	159800	361580	285000	287300	287310	290
2021-03-08	9840.000000	31674.000000	20890.000000	16852.000000	10665.000000	11544.000000	709
2021-03-09	-1.930894	-1.430195	-0.622307	-0.528127	0.600094	0.173250	
2021-03-10	-0.355691	-1.430195	-1.172810	-1.614052	-1.012658	-1.446639	-
2021-03-11	1.829268	-0.574604	0.909526	1.441965	1.387717	-0.684338	
2021-03-12	3.099593	0.839806	1.723313	3.056017	4.153774	1.074151	
...	...	...	...	...	...	...	
2022-03-03	-4.014228	-11.962493	-8.042125	-8.912889	15.471167	-12.898475	1
2022-03-04	-6.554878	-11.504704	-9.406415	-11.227154	17.065166	-13.808039	1
2022-03-07	-10.213415	-11.504704	-11.536620	-14.075481	16.690108	-16.190229	1
2022-03-08	-12.093496	-14.361937	-12.685495	-15.262283	13.689639	-17.143105	1
2022-03-10	-12.093496	-14.361937	-10.914313	-14.401851	17.440225	-16.190229	1

250 rows × 540 columns

In [21]:

```
#수익률 산정 기준값 행 삭제
cleaned_etf_df.drop(['2021-03-08'], inplace=True)
# for column in columns:
#     base = cleaned_etf_df.isna()[column].values.tolist().index(False)
#     cleaned_etf_df[column].iloc[base] =0

#결측치(NaN-상장 전 값 없음) 0으로 변환
cleaned_etf_df = cleaned_etf_df.fillna(0)
```

In [22]:

```
cleaned_etf_df
```

Out[22]:

	292340	159800	361580	285000	287300	287310	290080	284980	287320
2021-03-09	-1.930894	-1.430195	-0.622307	-0.528127	0.600094	0.173250	0.888575	2.257946	-2.103197
2021-03-10	-0.355691	-1.430195	-1.172810	-1.614052	-1.012658	-1.446639	-0.126939	1.190330	-1.215180
2021-03-11	1.829268	-0.574604	0.909526	1.441965	1.387717	-0.684338	0.761636	1.963431	0.420639
2021-03-12	3.099593	0.839806	1.723313	3.056017	4.153774	1.074151	0.832158	1.239416	2.234062
2021-03-15	2.997967	0.483046	1.220680	2.498220	6.282232	1.671864	1.918195	1.902074	2.514489
...	...	...	...	...	...	...	...	...	...
2022-	-4.014228	-11.962493	-8.042125	-8.912889	15.471167	-12.898475	16.431594	12.897288	7.216302

4. 분석 및 시각화

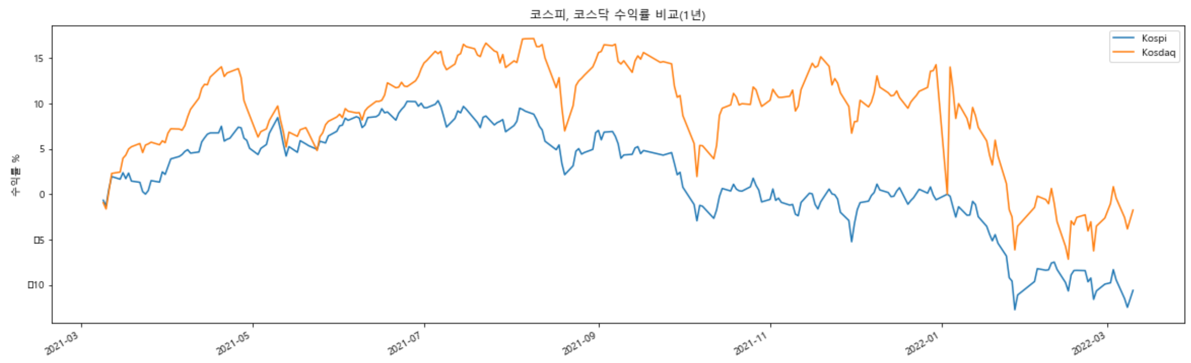
[코스피, 코스닥 1년전 기준 수익률 비교]

In [23]:

```
plt.rc('font', family='Malgun Gothic')

kospi_kosdaq = cleaned_etf_df[['Kospi','Kosdaq']]
# df3= pd.concat([etf_close_price3['Kospi'],etf_close_price3['Kosdaq']], axis=1)
kospi_kosdaq.plot(figsize=(20,6))
plt.ylabel('수익률 %')
plt.title('코스피, 코스닥 수익률 비교(1년)')
```

Out[23]:

```
Text(0.5, 1.0, '코스피, 코스닥 수익률 비교(1년)')

C:₩Users₩YJ₩anaconda3₩envs₩test₩lib₩site-packages₩IPython₩core₩pylabtools.py:151: Us
erWarning: Glyph 8722 (₩N{MINUS SIGN}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



[2022년 3월 8일 기준 수익률 top10 종목]

In [24]:

```
#etf 데이터프레임 인덱스 단축코드로 변경
etf2 = etf.set_index('단축코드')
etf2
```

Out[24]:

	표준코드	한글종목명	한글종목약명	영문종목명	상장일	기초지수명	지수산출기관	추적배수	복제방법	기초시장분류	기초자산분류	상장
단축코드												
292340	KR7292340007	DB 마이티 200커버드콜ATM레버리지증권상장지수	마이티 200커버드콜ATM 레버리지	DB Mighty KOSPI200 Covered Call ATM Leverage	2018/03/20	코스피 200 커버드콜	KRX	2X 레버리지	실물	국내	주식	600

In [25]:

```
# 수익률 column
last_index = len(cleaned_etf_df)-1
profit = cleaned_etf_df.iloc[last_index]
profit.pop('Kospi')
profit.pop('Kosdaq')
profit = pd.DataFrame(profit)
profit.columns = ['수익률']
profit.index = list(map(lambda x: int(x), profit.index))

profit
```

Out[25]:

	수익률
292340	-12.093496
159800	-14.361937
361580	-10.914313
285000	-14.401851
287300	17.440225
...	...
195970	-3.271441
195980	-17.776038
215620	11.479544
391670	-14.505713
391680	-16.542103

538 rows × 1 columns

In [26]:

```
#위 etf2 와 수익률 column 합치기
cleaned_etf = pd.concat([etf2, profit], axis=1)

#수익률 결측 행 삭제
cleaned_etf.dropna(inplace=True)
etf_name = cleaned_etf.sort_values(by='수익률', ascending=False)['기초지수명'].values
etf_rate = cleaned_etf.sort_values(by='수익률', ascending=False)['수익률'].values
```

In [27]:

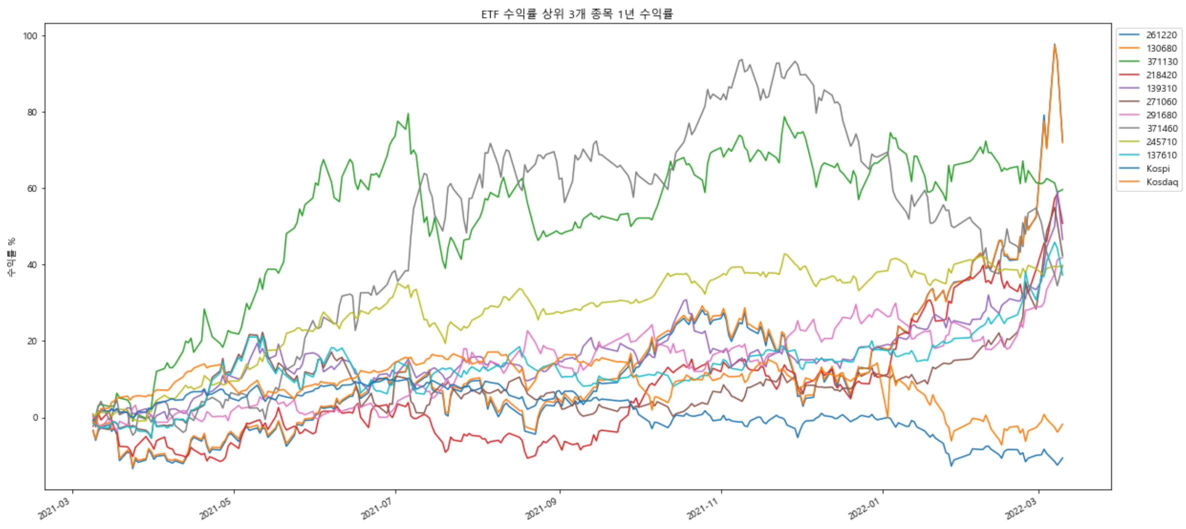
```
#top10
cleaned_etf[['한글종목명', '기초지수명', '수익률']].sort_values(by='수익률', ascending=False).head(
```

Out[27]:

	한글종목명	기초지수명	수익률
261220	삼성 KODEX WTI원유선물특별자산상장지수투자신탁[원유-파생형](H)	S&P GSCI Crude Oil Index ER	72.362556
130680	미래에셋 TIGER 원유선물 특별자산상장지수투자신탁(원유-파생형)	S&P GSCI Crude Oil Enhanced Index ER	71.947195
371130	한국투자KINDEX블룸버그베트남VN30선물레버리지증권상장지수투자신탁(주식-파생형)(H)	Bloomberg VN30 Futures Excess Return Index	59.631619
218420	삼성 KODEX 미국에너지 증권상장지수투자신탁[주식-파생형](합성)	S&P Select Sector Energy Index	50.781250
139310	미래에셋 TIGER 금속선물 특별자산상장지수투자신탁(금속-파생형)	S&P GSCI Industrial Metals Select Index(TR)	46.606705
271060	삼성 KODEX 3대농산물선물특별자산상장지수투자신탁[농산물-파생형](H)	S&P GSCI Grains Select Index ER	42.350656
291680	KB KBSTAR 차이나H선물인버스증권상장지수투자신탁(주식-파생형)(H)	Hang Seng China Enterprises Futures Index(Pric...	41.862955
371460	미래에셋 TIGER 차이나전기차SOLACTIVE증권상장지수투자신탁(주식-파생형)	Solactive China Electric Vehicle Index(Net Tot...	39.781887
245710	한국투자 KINDEX 베트남VN30증권상장지수투자신탁(주식-파생형)(합성)	VN30 Index(PR)	39.708940
137610	미래에셋 TIGER 농산물선물 특별자산상장지수투자신탁(농산물-파생형)	S&P GSCI Agriculture Enhanced Index(ER)	37.241379

In [28]:

```
top_ten = cleaned_etf.sort_values(by='수익률', ascending=False).index[:10]

top_ten2 = cleaned_etf_df[top_ten.tolist()+['Kospi','Kosdaq']]
top_ten2.plot(figsize=(20,10))
plt.legend(bbox_to_anchor=(1, 1))
plt.ylabel('수익률 %')
plt.title('ETF 수익률 상위 3개 종목 1년 수익률')
plt.show()
```

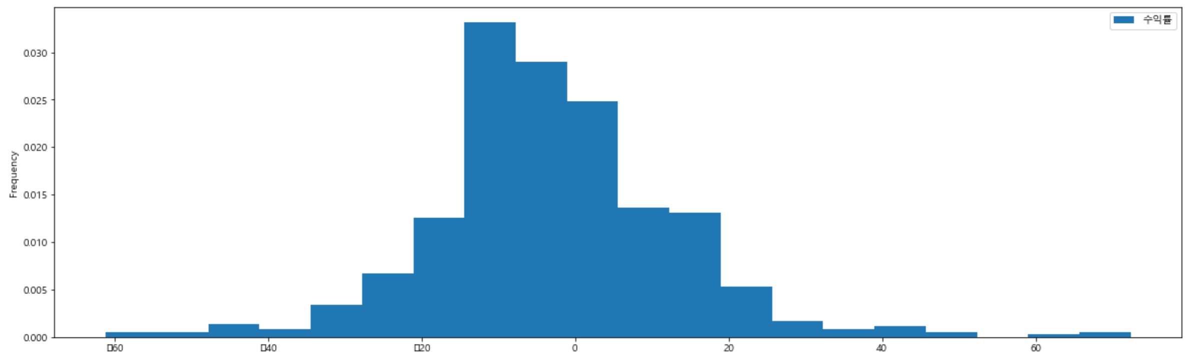


[전 종목 수익률 빈도 및 확률 분포]

In [29]:

```
#빈도수
fig, ax = plt.subplots(figsize=(20,6))
cleaned_etf.plot(kind='hist', y='수익률', bins=20, density=True, ax=ax)
plt.show()
```

```
C:\Users\YJ\anaconda3\envs\test\lib\site-packages\IPython\core\pylabtools.py:151: Us
erWarning: Glyph 8722 (\N{MINUS SIGN}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [30]:

```
mean = cleaned_etf['수익률'].mean()

kospi_yield = cleaned_etf_df['Kospi'].iloc[last_index]
kosdaq_yield = cleaned_etf_df['Kosdaq'].iloc[last_index]

mean
```

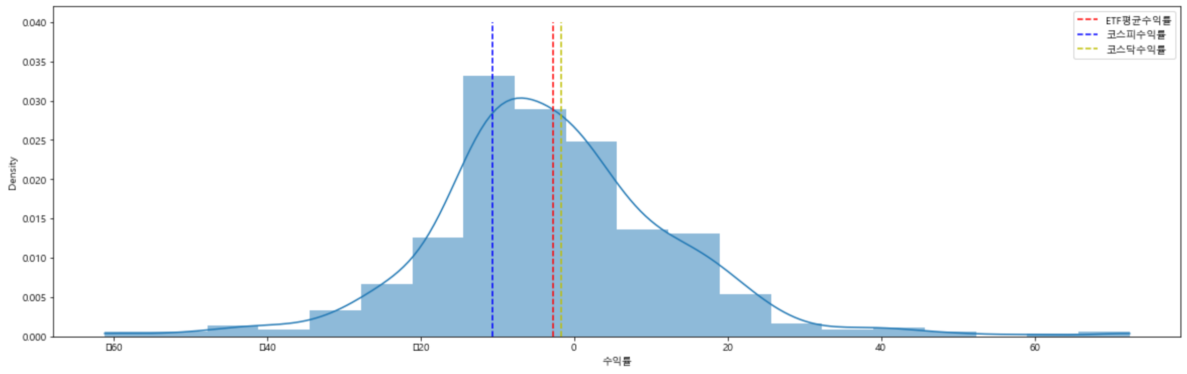
Out[30]:

```
-2.6961617177381663
```

In [31]:

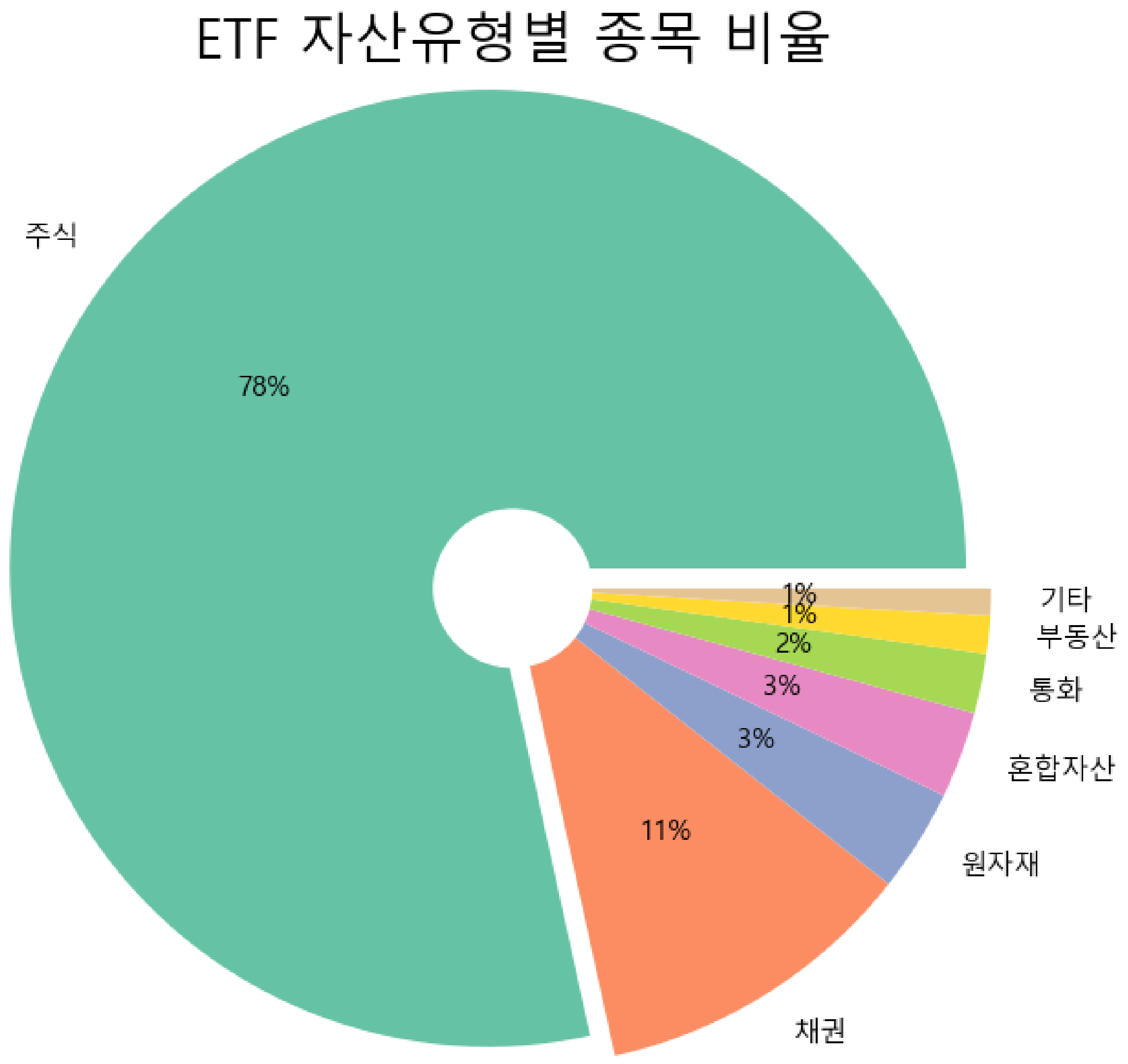
```
#확률 분포 (평균값, 코스피 수익률 비교)
fig, ax = plt.subplots(figsize=(20,6))
sns.histplot(cleaned_etf['수익률'], ax=ax, bins=20, kde=True, stat='density', linewidth=0)
plt.plot([mean, mean], [0,0.04], "r--", label="ETF평균수익률")
plt.plot([kospi_yield,kospi_yield], [0,0.04], "b--", label="코스피수익률")
plt.plot([kosdaq_yield,kosdaq_yield], [0,0.04], "y--", label="코스닥수익률")
plt.legend()
plt.show()
```

```
C:₩Users₩YJ₩anaconda3₩envs₩test₩lib₩site-packages₩IPython₩core₩pylabtools.py:151: Us
erWarning: Glyph 8722 (₩N{MINUS SIGN}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [32]:

```
cmap = plt.get_cmap('Set2')
colors = [cmap(i) for i in np.linspace(0, 1, 8)]

labels1 = etf['기초자산분류'].value_counts().index.tolist()
fracs1 = etf['기초자산분류'].value_counts().values.tolist()
explode1=(0.2,0,0,0,0,0,0)


plt.pie(fracs1, explode=explode1, labels = labels1, autopct = "%.0f%%", shadow= False, colors=colo
#donut
centre_circle = plt.Circle((0,0),0.5, color='black', fc='white',linewidth=0)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('ETF 자산유형별 종목 비율', fontsize=30,pad=180)

plt.show()

labels2 = etf['기초시장분류'].value_counts().index.tolist()
fracs2 = etf['기초시장분류'].value_counts().values.tolist()
explode2=(0.2,0,0)

plt.pie(fracs2, explode=explode2, labels = labels2, autopct = "%.0f%%", shadow= False, colors=colo
#donut
centre_circle = plt.Circle((0,0),0.5, color='black', fc='white',linewidth=0)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('ETF 시장별 종목 비율', fontsize=30,pad=180)

plt.show()
```


ETF 자산유형별 종목 비율
주식
78%
기타
1%
부동산
1%
통화
2%
혼합자산
3%
원자재
3%
채권
11%

In [33]:

```
sector = cleaned_etf.groupby('기초자산분류')['수익률'].mean().sort_values(ascending=False)
sector
```

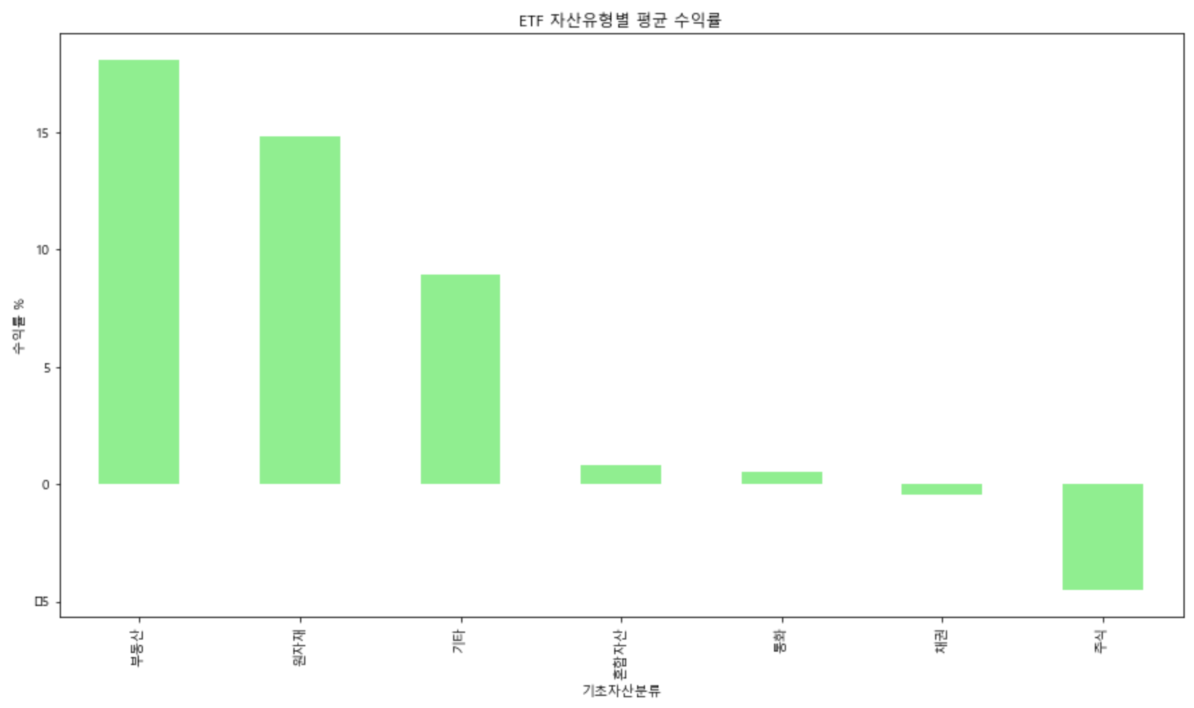
Out[33]:

```
기초자산분류
부동산      18.099826
원자재      14.812674
기타        8.923036
혼합자산     0.788318
통화        0.509333
채권       -0.444969
주식       -4.511446
Name: 수익률, dtype: float64
```

In [34]:

```
plt.figure(figsize=(15,8))
plt.title('ETF 자산유형별 평균 수익률')
sector.plot(kind='bar', color='lightgreen')
plt.ylabel('수익률 %')
plt.show()
```

```
C:₩Users₩YJ₩anaconda3₩envs₩test₩lib₩site-packages₩IPython₩core₩pylabtools.py:151: Us
erWarning: Glyph 8722 (₩N{MINUS SIGN}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [35]:

```
market = cleaned_etf.groupby('기초시장분류')['수익률'].mean().sort_values(ascending=False)
market
```

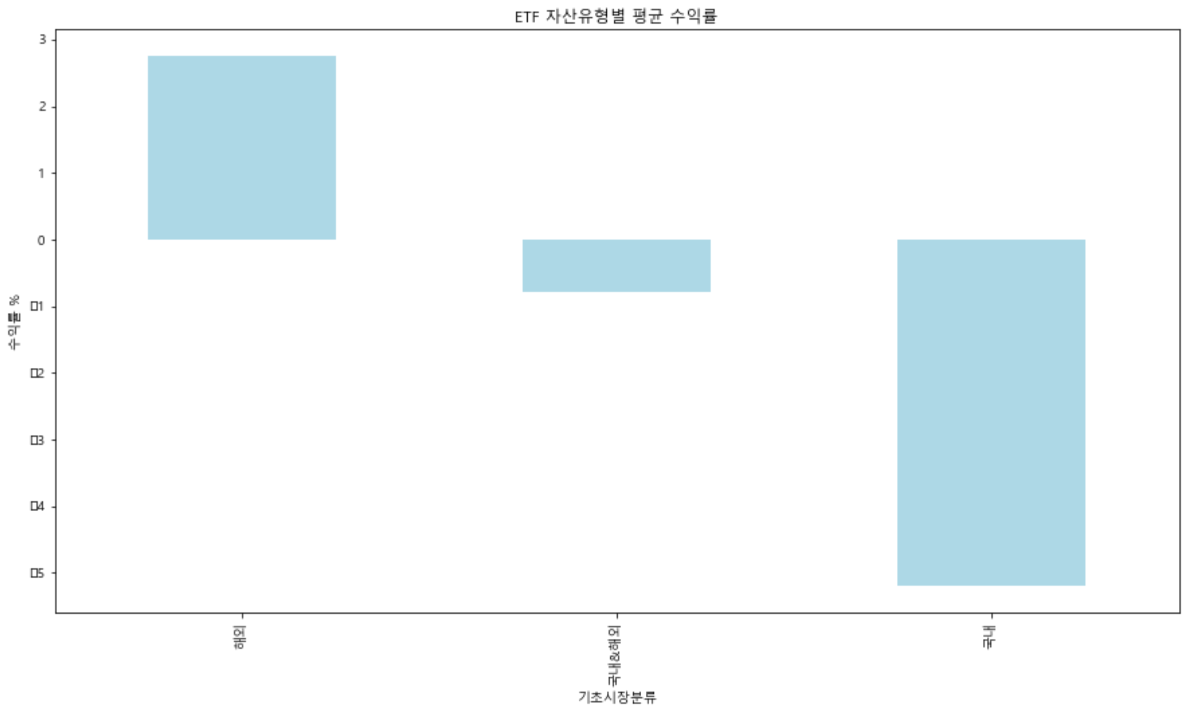
Out[35]:

```
기초시장분류
해외       2.758448
국내&해외   -0.781348
국내      -5.196613
Name: 수익률, dtype: float64
```

In [36]:

```
plt.figure(figsize=(15,8))
plt.title('ETF 자산유형별 평균 수익률')
plt.ylabel('수익률 %')
market.plot(kind='bar', color='lightblue')
plt.show()
```

```
C:₩Users₩YJ₩anaconda3₩envs₩test₩lib₩site-packages₩IPython₩core₩pylabtools.py:151: Us
erWarning: Glyph 8722 (₩N{MINUS SIGN}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```


ETF 자산유형별 평균 수익률
수익률 %
□1
□2
□3
□4
□5
해외
국내&해외
국내
기초시장분류