```
## 구글 드라이브 마운트
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/
```

```
from google.colab import output

# !cp 파일1 파일2 # 파일1을 파일2로 복사 붙여넣기
!cp "/content/drive/MyDrive/TAVE_빅콘테스트/제출데이터/RAW_DATA.zip" "data.zip"  ## 파일경로 예

# data.zip을 현재 디렉터리에 압축해제
!unzip "data.zip"
```

```
## 한글폰트 깨짐현상 해결
!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf


import matplotlib.pyplot as plt
plt.rc('font', family='NanumBarunGothic')

from google.colab import output
output.clear()
```

런타임 재시작

```
import pandas as pd
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
plt.rc('font', family='NanumBarunGothic')
import seaborn as sns
import statsmodels
import glob
from google.colab import drive
from google.colab import output
```

```
TRAIN_PATH = './제공데이터/2021 빅콘테스트_데이터분석분야_챔피언리그_수산Biz_문제데이터.xlsx'
SUBMISSION_PATH = './평가데이터/2021 빅콘테스트_데이터분석분야_챔피언리그_수산Biz_평가데이터.xls

ADDITIONAL_PATH = "./제공데이터/2021 빅콘테스트_데이터분석분야_챔피언리그_수산Biz_자율평가데이터
EXTERNAL_FUEL_PATH = './외부데이터/국제유가2021-09-09.csv'

df_train = pd.read_excel(TRAIN_PATH, )
df_external = pd.read_csv(EXTERNAL_FUEL_PATH)
df_additional = pd.read_excel(ADDITIONAL_PATH, )
```

```
10
11 # data loaded
12 DATA_PATH = '/외부데이터/*'
13 glob.glob(DATA_PATH)
```

```
[]
```

```
1 !ls 제공데이터
```

```
'2021 빅콘테스트_데이터분석분야_챔피언리그_수산Biz_평가데이터_update_210831.xlsx'
'2021 빅콘테스트_데이터분석분야_챔피언리그_수산Biz_문제데이터.xlsx'
'2021 빅콘테스트_데이터분석분야_챔피언리그_수산Biz_자율평가데이터.xlsx'
```

```
1 display(df_train)
```

	REG_DATE	P_TYPE	CTRY_1	CTRY_2	P_PURPOSE	CATEGORY_1	CATEGORY_2	P_NAME
0	2015-12-28	수산물	아르헨티나	아르헨티나	판매용	갑각류	새우	아르헨티나붉은새우
1	2015-12-28	수산물	바레인	바레인	판매용	갑각류	게	꽃게
2	2015-12-28	수산물	바레인	바레인	판매용	갑각류	게	꽃게
3	2015-12-28	수산물	칠레	칠레	판매용	패류 멍게류	해삼	해심
4	2015-12-28	수산물	중국	중국	판매용	어류	서대 박대 페루다	서대
...	...	...	...	...	...	...	...	..
42068	2019-12-30	수산물	러시아	러시아	판매용	갑각류	게	왕게
42069	2019-12-30	수산물	중국	중국	판매용	연체류 해물모듬	낙지	낙지

```
1 display(df_additional)
```

	REG_DATE	P_TYPE	CTRY_1	CTRY_2	P_PURPOSE	CATEGORY_1	CATEGORY_2	P_NAME
0	2020-01-06	수산물	호주	일본	판매용	어류	참치 새치류	남방참다랑어
1	2020-01-06	수산물	칠레	일본	자사제품제조용	어류	연어	은연어
2	2020-01-	수산물	중국	중국	판매용	어류	도다리	강도다

```
# 제공데이터 병합
df_train_copy = df_train.copy()
df_additional_copy = df_additional.copy()

df_train_copy = pd.concat([df_train_copy, df_additional_copy], axis = 0)
df_train = df_train_copy
```

	...	...	...	...	...	...	...	...

```
# EDA for 제공데이터
df_squid = df_train[df_train['P_NAME'] == '오징어']
df_salmon = df_train[df_train['P_NAME'] == '연어']
df_shrimp = df_train[df_train['P_NAME'] == '흰다리새우']
```

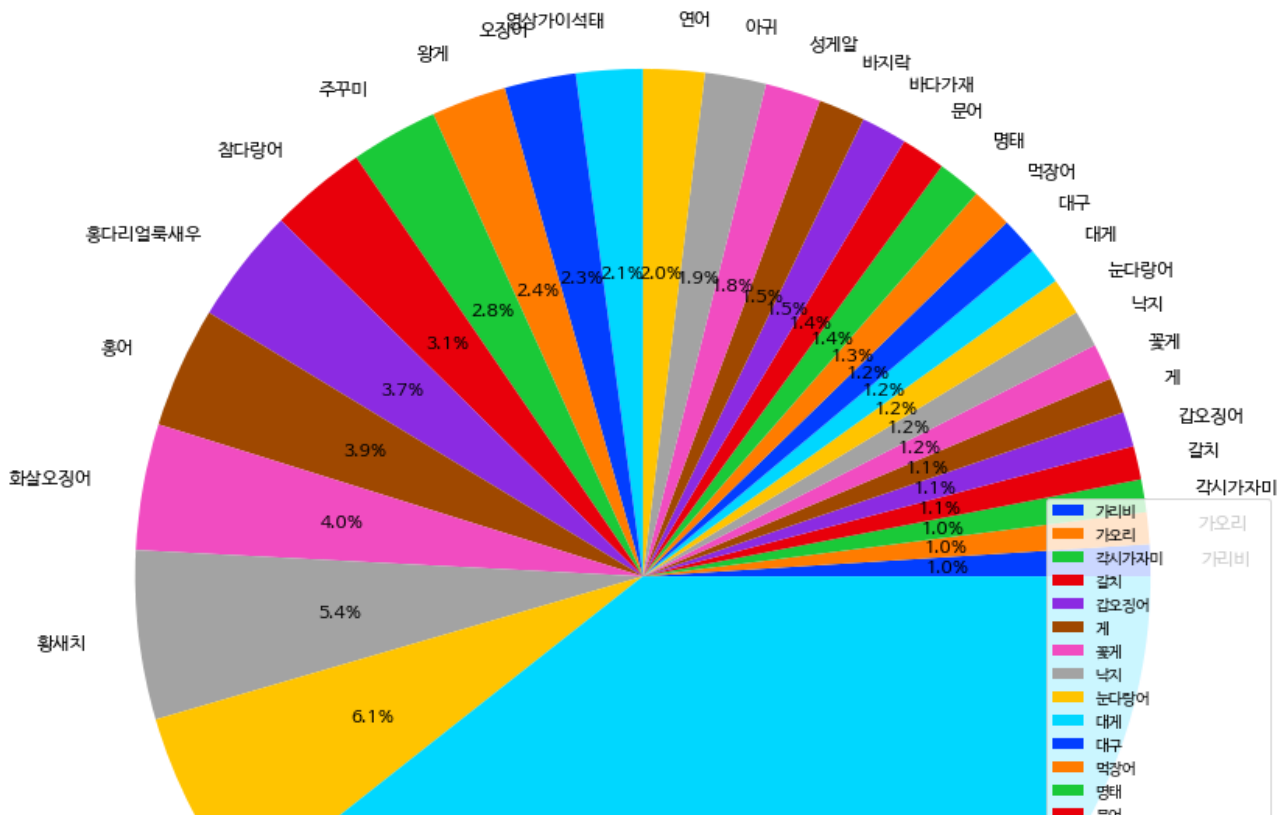
```
df_train['P_NAME'].value_counts()[0:10].values
```

```
array([3133, 2771, 2070, 1987, 1895, 1588, 1430, 1230, 1170, 1090])
```

```
# seaborn을 이용한 데이터 시각화(데이터 수량 상위 10개 품목)
plt.figure(figsize = (12,8))
sns.countplot(x = 'P_NAME', data = df_train, order = df_train['P_NAME'].value_counts().keys()[0
plt.xticks(rotation = 30)
plt.xlabel("품목", fontsize = 14)
plt.ylabel('데이터 수량', fontsize = 14)
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```

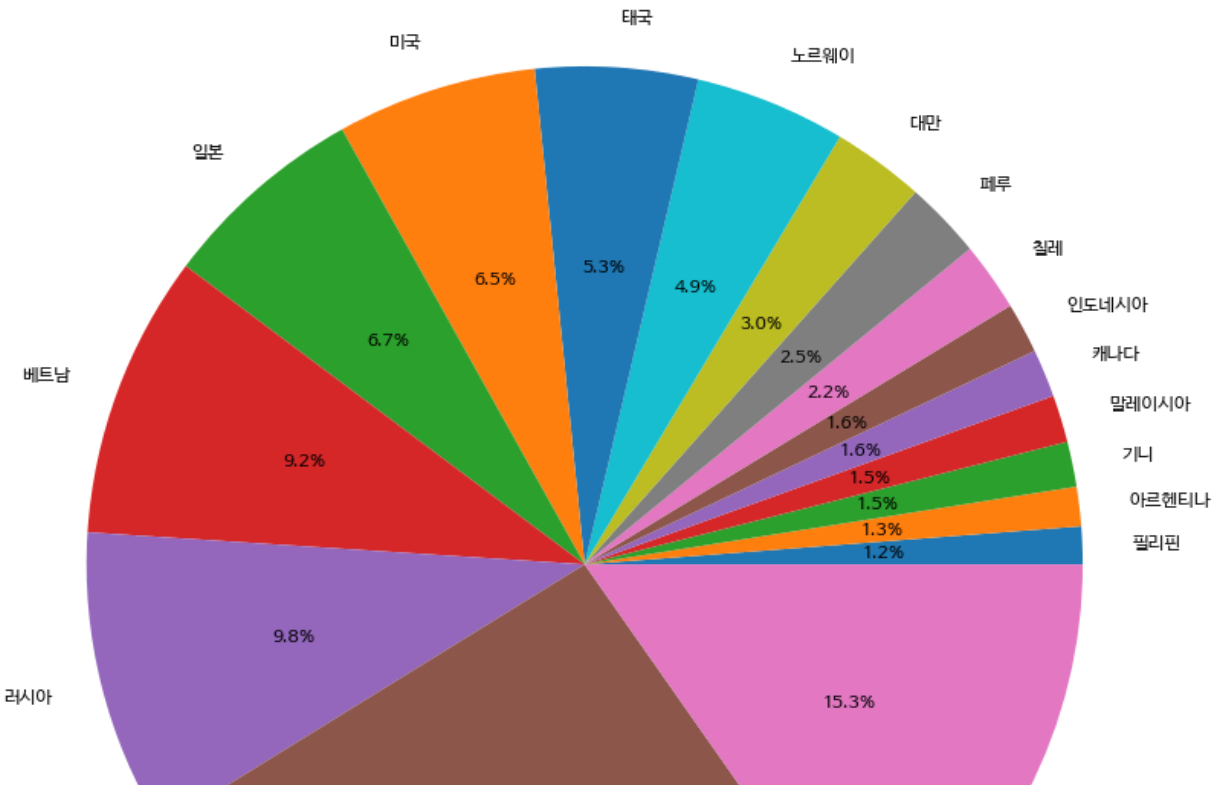

```
# pie chart를 이용한 시각화

p_name_total = np.sum(df_train['P_NAME'].value_counts().values)
p_name_ratio = df_train['P_NAME'].value_counts() / p_name_total
p_name_order = np.unique(df_train['P_NAME'].values).tolist()

count_rest = 0

for p_name, count in p_name_ratio.items():
    if count <= 0.01:
        count_rest += count
        p_name_order.remove(p_name)

rest_value = {"기타 수산물 품목(1%이하)" : count_rest}
p_name_order.append("기타 수산물 품목(1%이하)")
p_name_ratio = p_name_ratio.append(pd.Series(rest_value))

colors = sns.color_palette('bright')[0:len(p_name_order)]
plt.figure(figsize = (15,15))
plt.pie(p_name_ratio[p_name_order].sort_values(ascending=True), labels = p_name_order, colors =
plt.legend(p_name_order, loc = "lower right")
output.clear()
plt.show()
```




```
1 # seaborn을 이용한 데이터 시각화(데이터 수량 상위 10개 품목)
2 plt.figure(figsize = (12,8))
3 sns.countplot(x = 'CTRY_1', data = df_train, order = df_train['CTRY_1'].value_counts().keys()[0
4 plt.xticks(rotation = 30)
5 plt.xlabel("제조국가", fontsize = 14)
6 plt.ylabel('데이터 수량', fontsize = 14)
7 plt.legend()
8 plt.show()
```

```
No handles with labels found to put in legend.
```

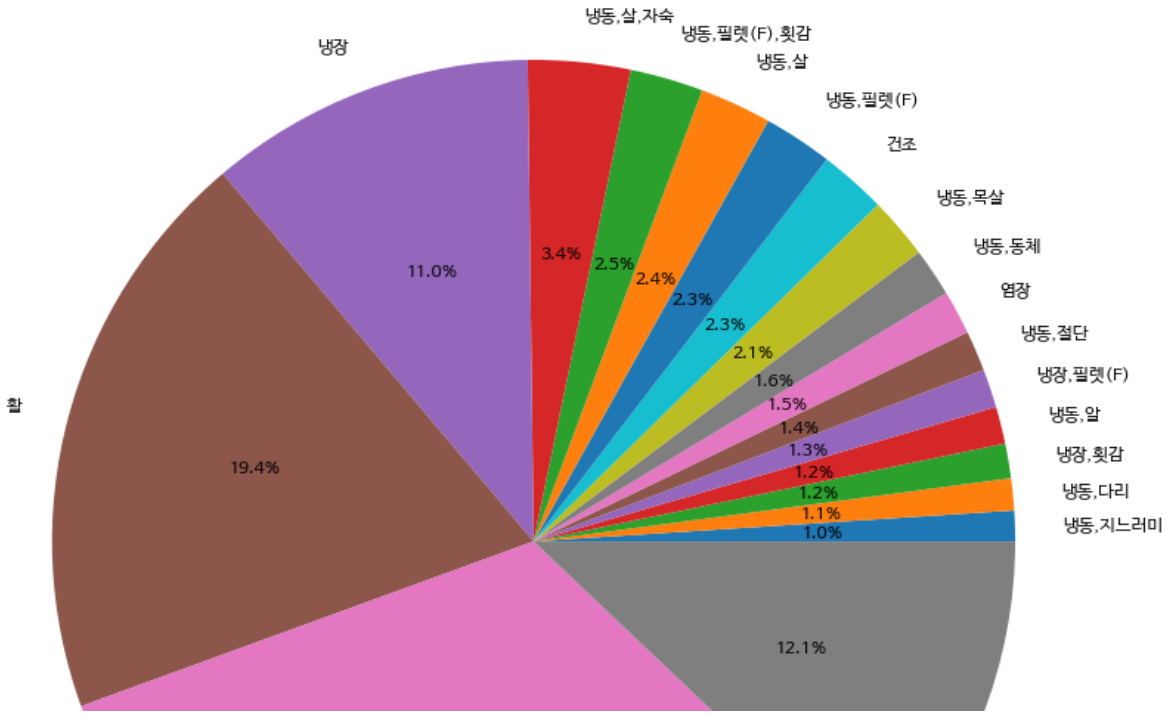

```
# pie chart를 이용한 시각화

p_name_total = np.sum(df_train['CTRY_1'].value_counts().values)
p_name_ratio = df_train['CTRY_1'].value_counts() / p_name_total
p_name_ratio = p_name_ratio.sort_values(ascending = True)
p_name_order = p_name_ratio.index.tolist()

count_rest = 0

for p_name, count in p_name_ratio.items():
    if count <= 0.01:
        count_rest += count
        p_name_order.remove(p_name)


rest_value = {"기타 제조국(1%이하)" : count_rest}
p_name_order.append("기타 제조국(1%이하)")
p_name_ratio = p_name_ratio.append(pd.Series(rest_value))

colors = sns.color_palette('tab10')[0:len(p_name_order)]
plt.figure(figsize = (15,15))
plt.pie(p_name_ratio[p_name_order], labels = p_name_order, colors = colors, autopct='%.1f%%', te
plt.legend(p_name_order, loc = "lower right")
output.clear()
plt.show()
```




```
1 plt.figure(figsize = (12,8))
2 sns.countplot(x = 'P_IMPORT_TYPE', data = df_train, order = df_train['P_IMPORT_TYPE'].value_cou
3 plt.xticks(rotation = 30)
4 plt.xlabel("수입 형태", fontsize = 14)
5 plt.ylabel('데이터 수량', fontsize = 14)
6 plt.legend()
7 plt.show()
```

```
No handles with labels found to put in legend.
```


```
# pie chart를 이용한 시각화

p_name_total = np.sum(df_train['P_IMPORT_TYPE'].value_counts().values)
p_name_ratio = df_train['P_IMPORT_TYPE'].value_counts() / p_name_total
p_name_ratio = p_name_ratio.sort_values(ascending = True)
p_name_order = p_name_ratio.index.tolist()

count_rest = 0

for p_name, count in p_name_ratio.items():
    if count <= 0.01:
        count_rest += count
        p_name_order.remove(p_name)

rest_value = {"기타 수입 형태(1%이하)" : count_rest}
p_name_order.append("기타 수입 형태(1%이하)")
p_name_ratio = p_name_ratio.append(pd.Series(rest_value))

colors = sns.color_palette('tab10')[0:len(p_name_order)]
plt.figure(figsize = (15,15))
plt.pie(p_name_ratio[p_name_order], labels = p_name_order, colors = colors, autopct='%.1f%%', te
plt.legend(p_name_order, loc = "lower right")
output.clear()
plt.show()
```

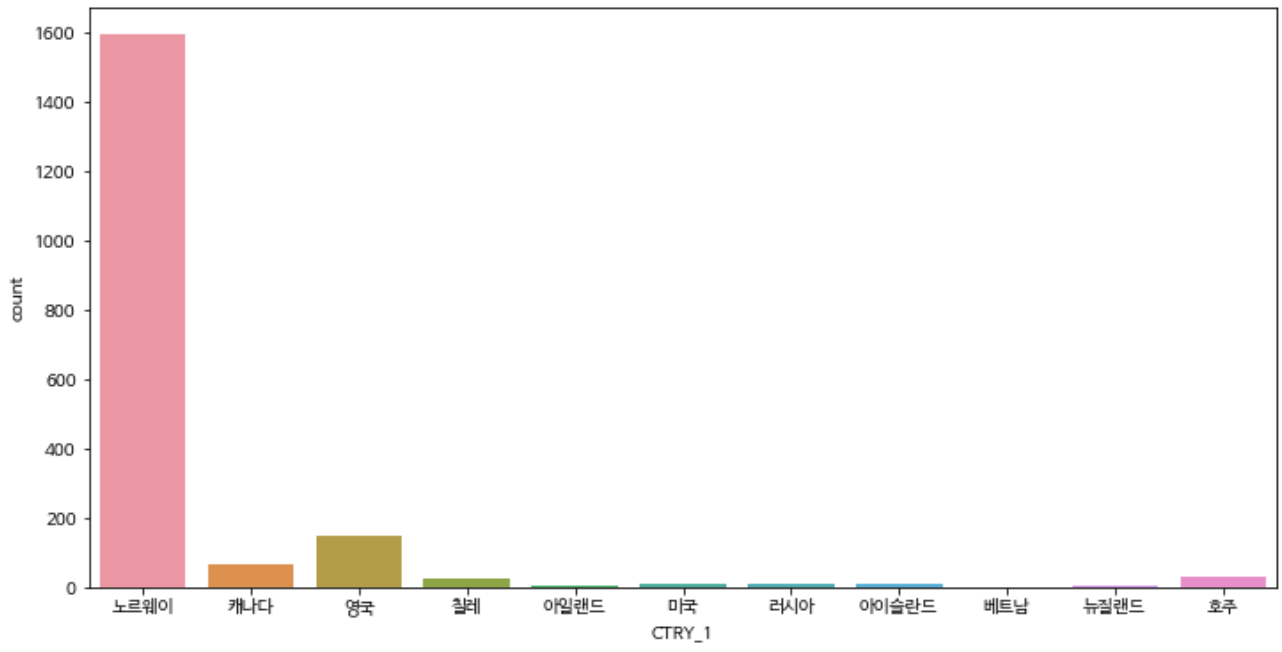



```
1 # 제조국별 분포
2
3 print(df_squid['CTRY_1'].value_counts().shape)
4 print(df_squid['CTRY_1'].value_counts())
5 plt.figure(figsize = (15,6))
6 sns.countplot(x = "CTRY_1", data = df_squid)
```

```
(18,)
페루          1046
중국           712
칠레           655
아르헨티나        91
대한민국          72
대만            71
뉴질랜드          61
미국             17
러시아            11
우루과이            8
인도네시아           7
캐나다             5
에콰도르            4
스페인             4
```

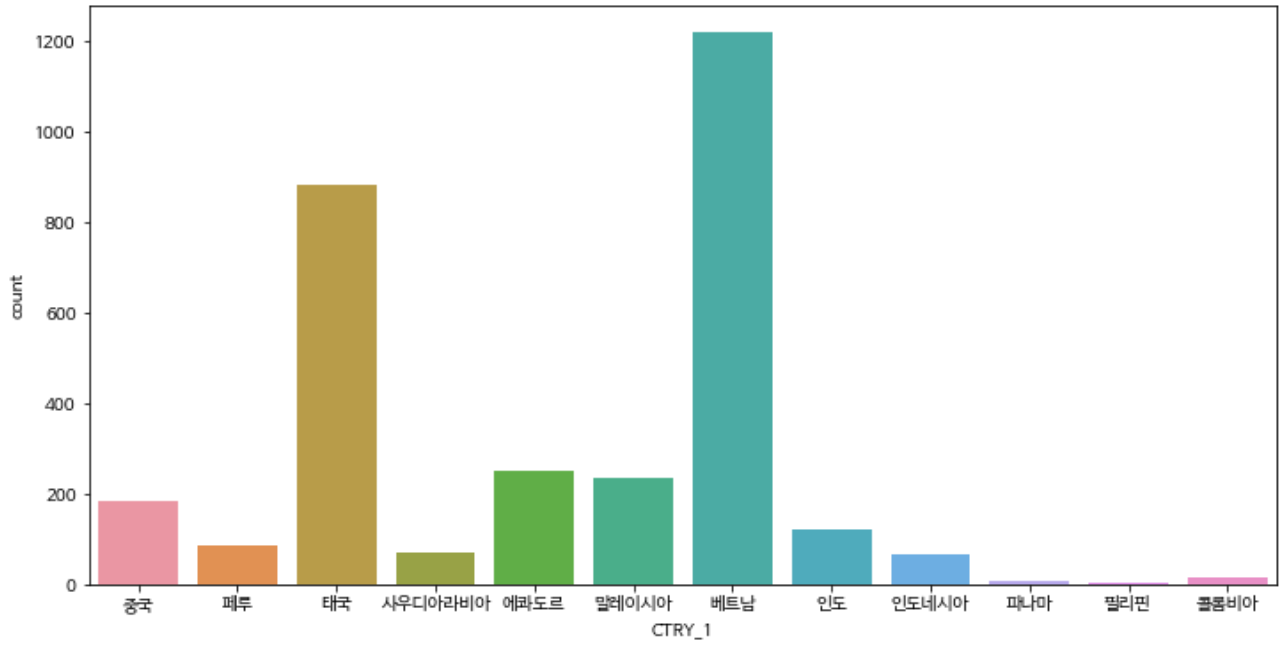
```
print('-----------------------------------')
print(df_salmon['CTRY_1'].value_counts().shape)
print(df_salmon['CTRY_1'].value_counts())
plt.figure(figsize = (12,6))
sns.countplot(x = "CTRY_1", data = df_salmon)
```

```
-----------------------------------
(11,)
노르웨이       1596
영국           147
캐나다           66
호주            31
칠레            23
러시아            11
미국             8
아이슬란드           7
뉴질랜드            3
아일랜드            2
베트남             1
Name: CTRY_1, dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7f83cd319990>
```



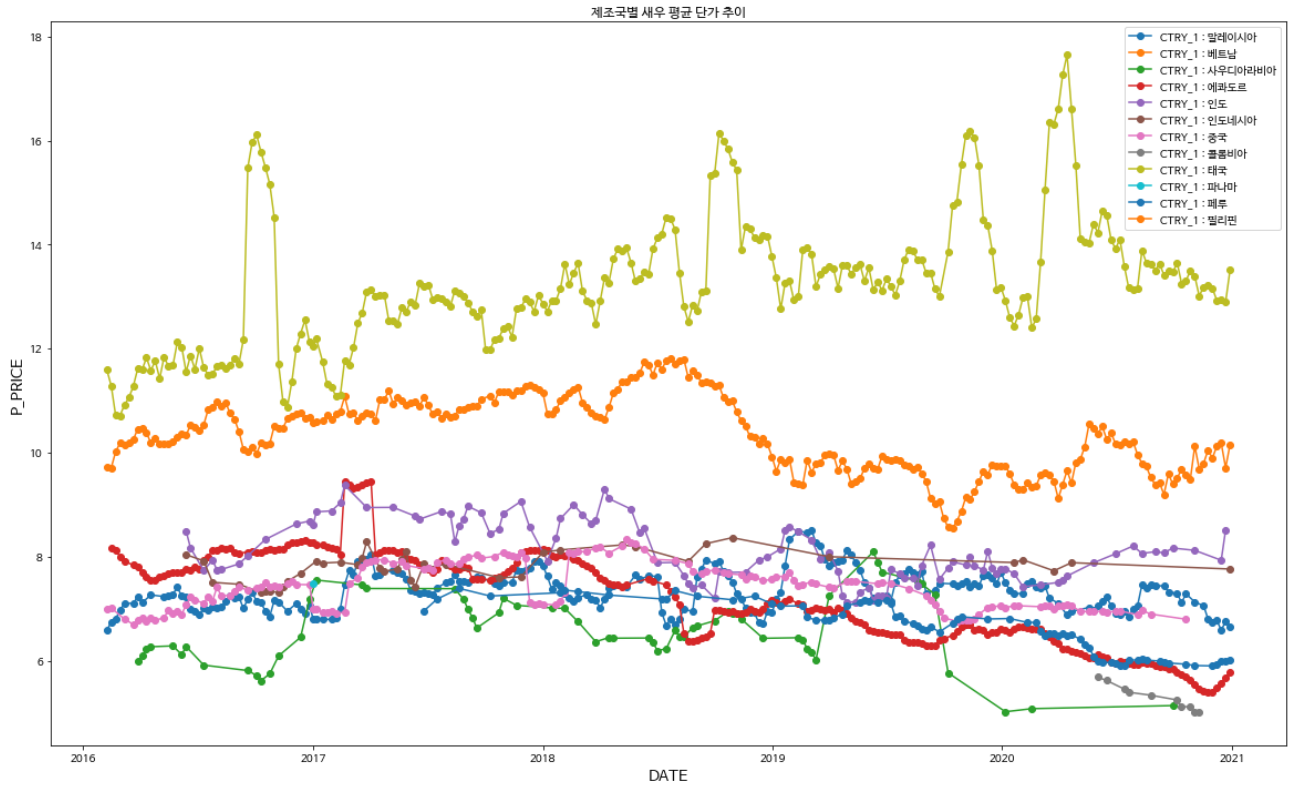
```
print('-----------------------------------')
print(df_shrimp['CTRY_1'].value_counts().shape)
print(df_shrimp['CTRY_1'].value_counts())
plt.figure(figsize = (12,6))
sns.countplot(x = "CTRY_1", data = df_shrimp)
```

```
---------------------------------
(12,)
베트남          1219
태국             883
에콰도르           250
말레이시아          233
중국             183
인도             120
페루              84
사우디아라비아         69
인도네시아           65
콜롬비아            16
파나마              7
필리핀              4
Name: CTRY_1, dtype: int64
<matplotlib.axes._subplots.AxesSubplot at 0x7f83cae31310>
```



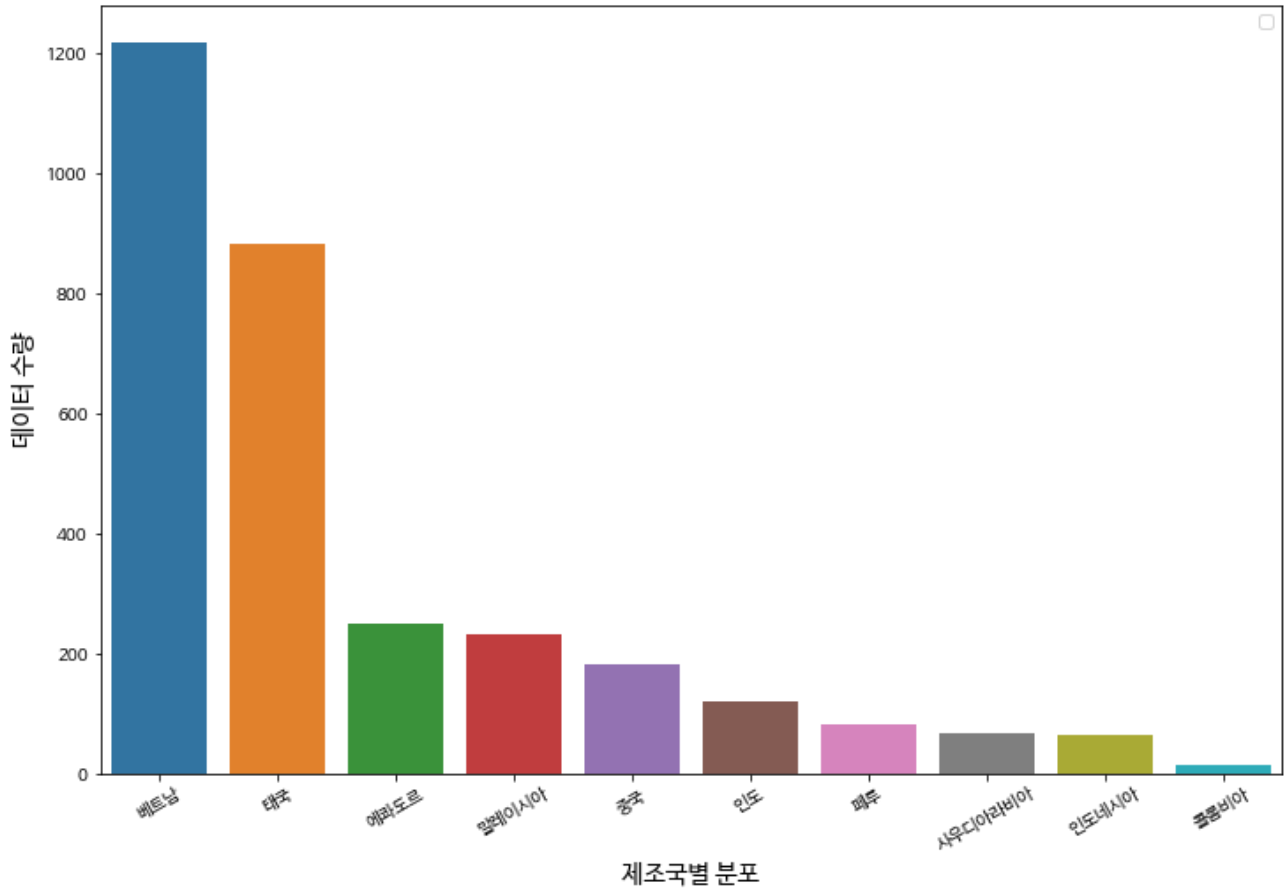

```
# 새우에 대한 제조국별 평균 단가의 차이 비교
df_shrimp_groupby = df_shrimp.groupby(by = 'CTRY_1')
plt.figure(1, figsize = (20,12))

for key, group in df_shrimp_groupby:
    plt.figure(1)
    group_series = group.groupby(by = 'REG_DATE').mean().rolling(7).mean()
    plt.plot(group_series['P_PRICE'], label = "CTRY_1 : " + key, marker = 'o')

plt.xlabel('DATE', fontsize = 14)
plt.ylabel('P_PRICE', fontsize = 14)
plt.title("제조국별 새우 평균 단가 추이")
```

```
plt.title('제조국별 새우 평균 단가 추이')
plt.xticks(rotation = 0)
plt.legend()
plt.show()
```

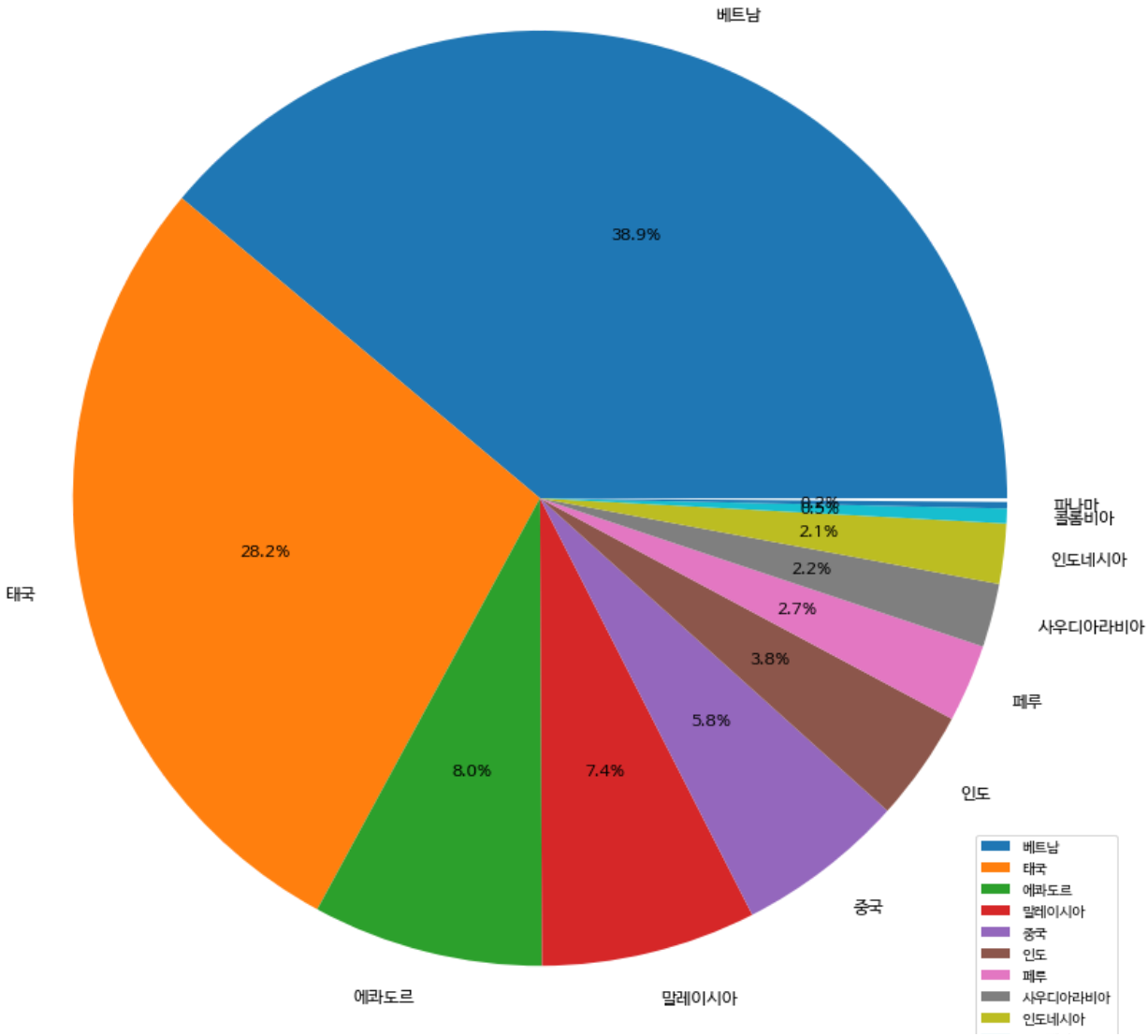



```
# 새우 데이터에 대한 제조국별 분포

plt.figure(figsize = (12,8))
sns.countplot(x = 'CTRY_1', data = df_shrimp, order = df_shrimp['CTRY_1'].value_counts().keys()
plt.xticks(rotation = 30)
plt.xlabel("제조국별 분포", fontsize = 14)
plt.ylabel('데이터 수량', fontsize = 14)
plt.legend()
plt.show()
```

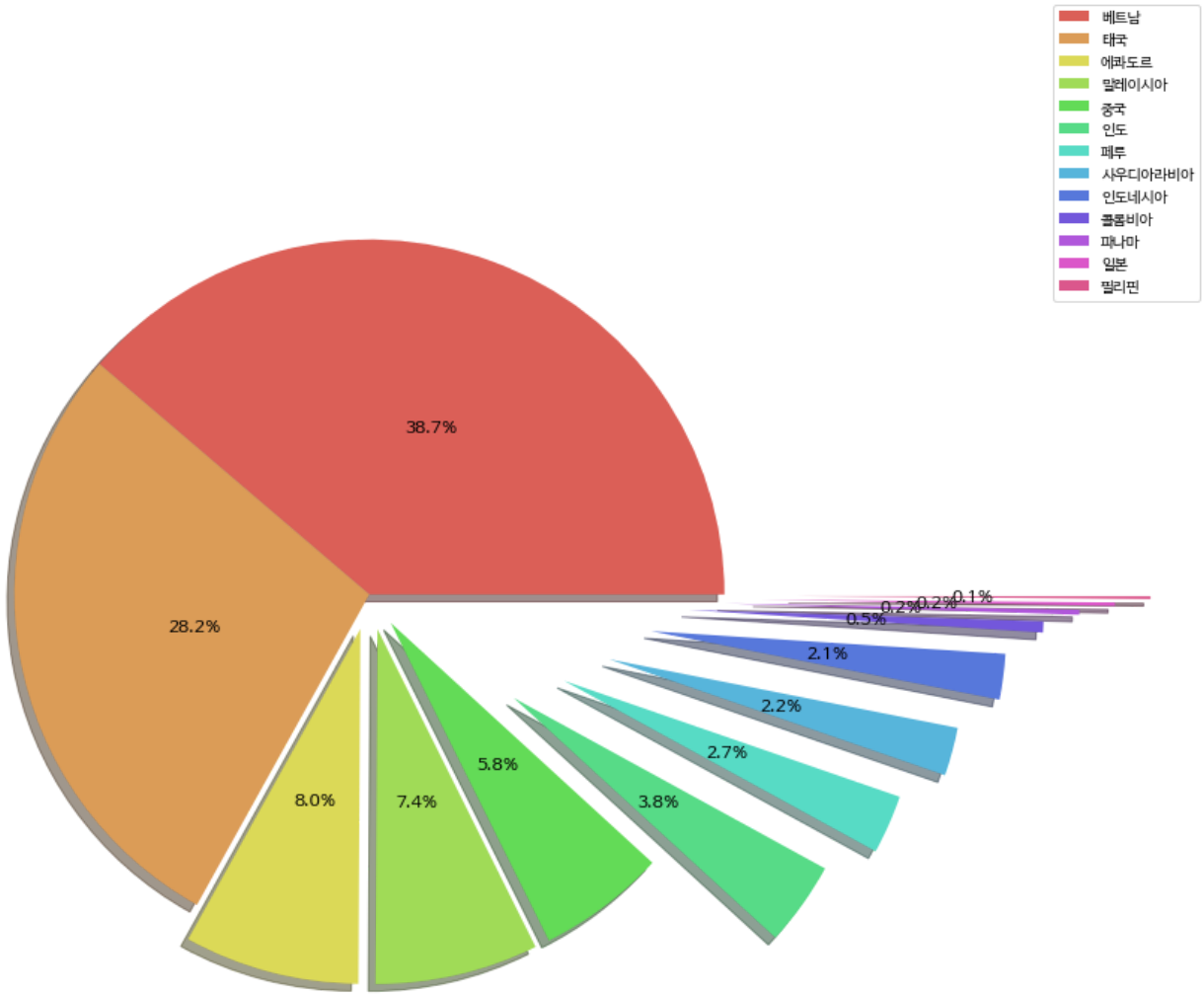
```
No handles with labels found to put in legend.
```




```
# pie chart를 이용한 시각화

p_name_total = np.sum(df_shrimp['CTRY_1'].value_counts().values)
p_name_ratio = df_shrimp['CTRY_1'].value_counts() / p_name_total
p_name_ratio = p_name_ratio.sort_values(ascending = False)[0:-1]
p_name_order = p_name_ratio.index.tolist()

colors = sns.color_palette('tab10')[0:len(p_name_order)]
plt.figure(figsize = (15,15))
plt.pie(p_name_ratio[p_name_order], labels = p_name_order, colors = colors, autopct='%.1f%%', te
plt.legend(p_name_order, loc = "lower right")
output.clear()
plt.show()
```




```
colors = sns.color_palette('hls',len(df_shrimp['CTRY_2'].value_counts().index))

plt.figure(figsize=(15,15))
plt.pie(df_shrimp['CTRY_2'].value_counts().values,
        autopct='%1.1f%%',
        pctdistance=0.5,
        labeldistance=1.1,
        explode = [0, 0, 0.1, 0.1, 0.1, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1,1.2],
        textprops={'fontsize': 12},
        colors = colors,
        shadow = True)

plt.axis('equal')
plt.legend(df_shrimp['CTRY_2'].value_counts().index)
plt.savefig('shrimp_country.png',dpi=200)
plt.show()
```




```
## 크루스칼 - 왈리스 순위합 검정(Kuscal-Wallis Rank Sum Test)
## k 집단중 하나라도 정규성 가정 깨질때 사용하는 검정방식
## 분석결과 통계적으로 유의한 차이있음
from scipy import stats
ctry_1_list = df_shrimp.CTRY_1.value_counts().sort_values(ascending = False).index[0:3]

pvalue_dict = {};
for ctry in ctry_1_list:
    lst = []
    for ctry2 in ctry_1_list:
        if ctry == ctry2:
            pvalue = 0
            lst.append(pvalue)
        else:
            pvalue = stats.kruskal(df_shrimp.loc[df_shrimp.CTRY_1==ctry,'P_PRICE'].values, df_sh
```

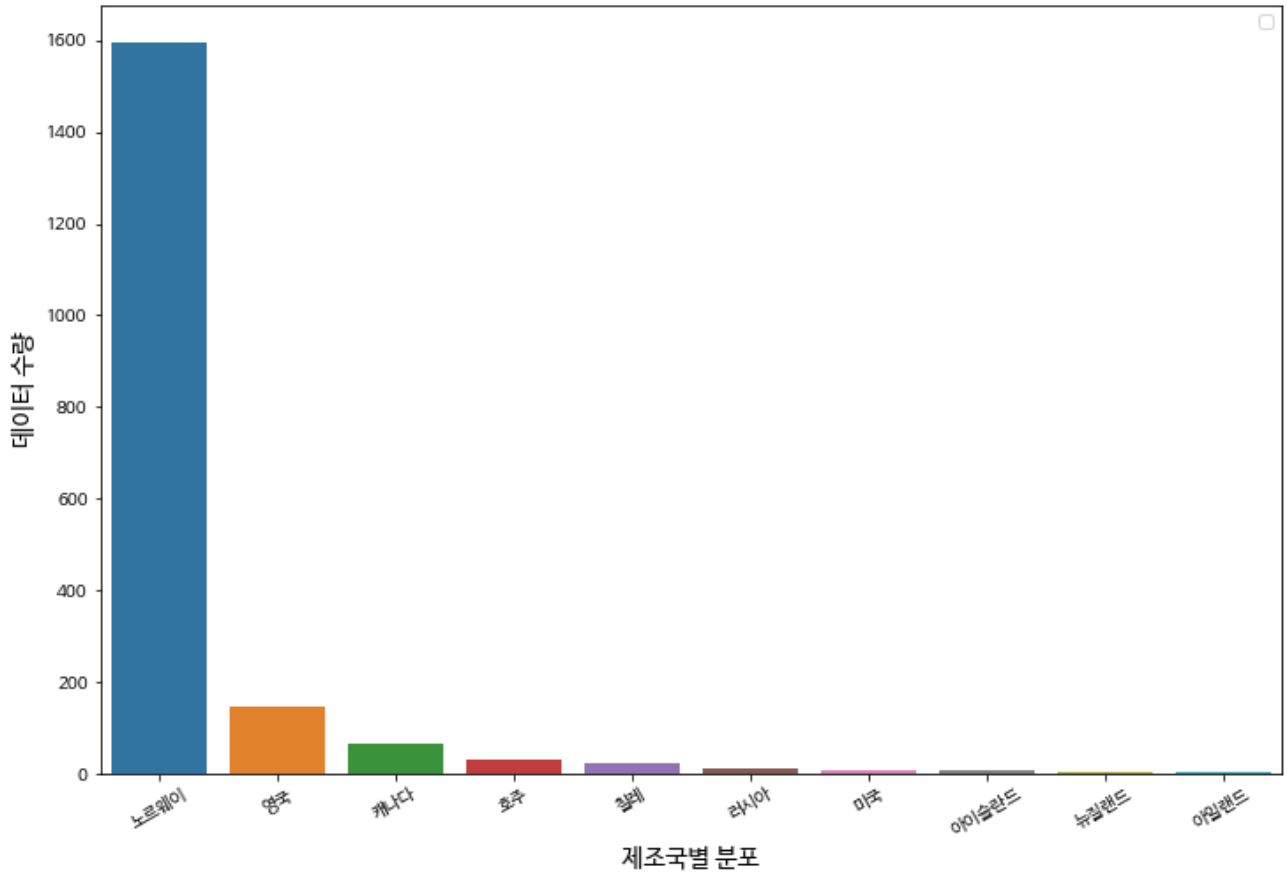
```
            lst.append(pvalue)

    pvalue_dict[ctry] = lst

df_pvalue = pd.DataFrame(data = pvalue_dict, index = ctry_1_list)
display(df_pvalue)


## 등분산 검정
## 귀무가설 : 등분산이다. / 대립가설 : 이분산이다.
## 오징어의 경우 이분산
## 유의확률이 0.000이므로 유의수준 0.05에서 이분산이다.
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='베트남','P_PRICE'],df_shrimp.lo
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='베트남','P_PRICE'],df_shrimp.lo
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='베트남','P_PRICE'],df_shrimp.lo
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='베트남','P_PRICE'],df_shrimp.lo
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='태국','P_PRICE'],df_shrimp.loc
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='태국','P_PRICE'],df_shrimp.loc
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='태국','P_PRICE'],df_shrimp.loc
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='에콰도르','P_PRICE'],df_shrimp
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='에콰도르','P_PRICE'],df_shrimp
print("p-value: ", stats.levene(df_shrimp.loc[df_shrimp.CTRY_1=='말레이시아','P_PRICE'],df_shrir

pvalue_dict = {};
for ctry in ctry_1_list:
    lst = []
    for ctry2 in ctry_1_list:
        if ctry == ctry2:
            pvalue = 0
            lst.append(pvalue)
        else:
            pvalue = stats.levene(df_shrimp.loc[df_shrimp.CTRY_1==ctry,'P_PRICE'].values, df_shr
            lst.append(pvalue)

    pvalue_dict[ctry] = lst

df_pvalue = pd.DataFrame(data = pvalue_dict, index = ctry_1_list)
display(df_pvalue)
```

	베트남	태국	에콰도르
베트남	0.000000e+00	4.180769e-72	1.153107e-72
태국	4.180769e-72	0.000000e+00	2.339415e-117
에콰도르	1.153107e-72	2.339415e-117	0.000000e+00

p-value: 1.9858079914720266e-05

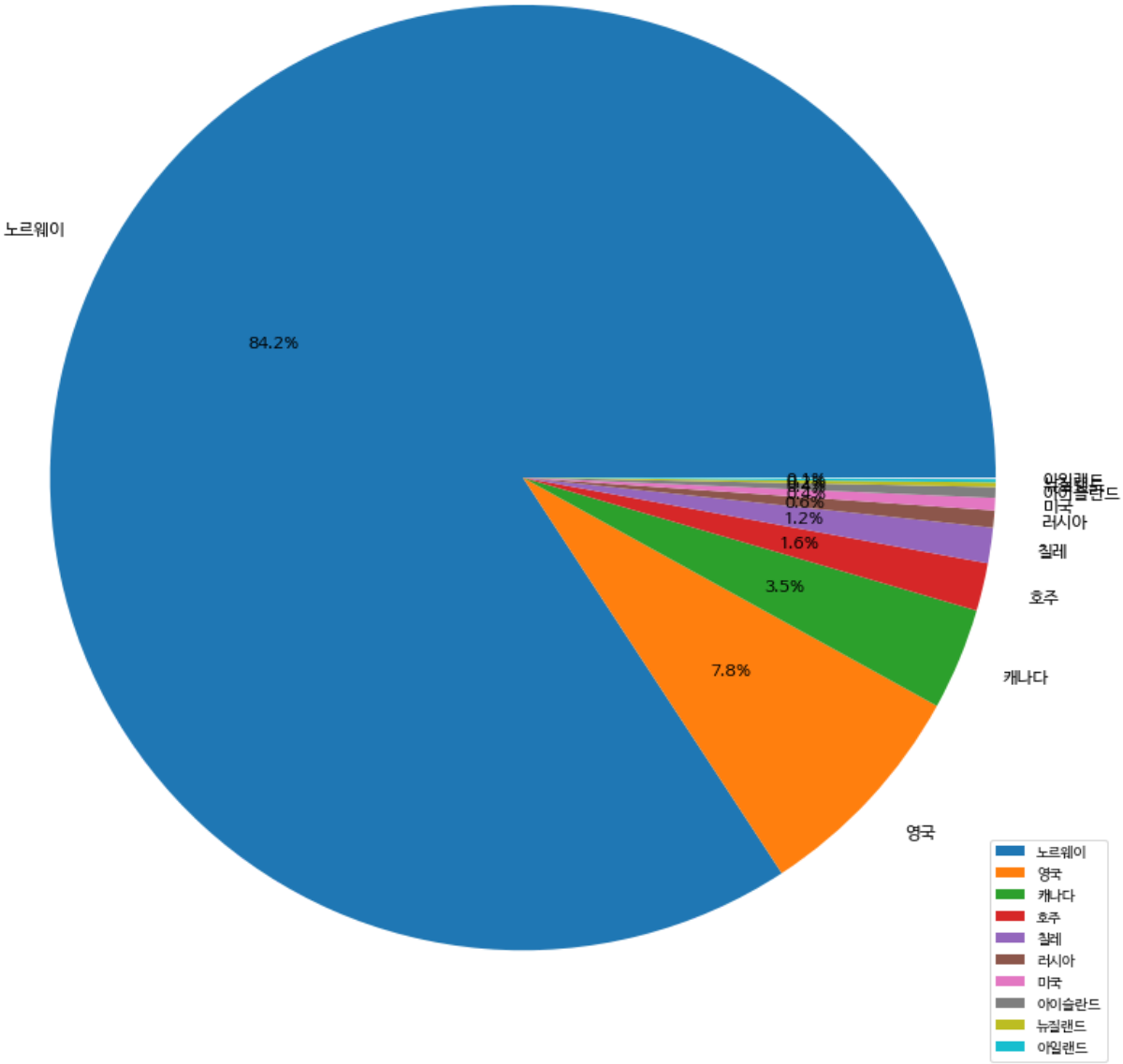
```
# 연어 데이터에 대한 제조국별 분포

plt.figure(figsize = (12,8))
sns.countplot(x = 'CTRY_1', data = df_salmon, order = df_salmon['CTRY_1'].value_counts().keys()
plt.xticks(rotation = 30)
plt.xlabel("제조국별 분포", fontsize = 14)
plt.ylabel('데이터 수량', fontsize = 14)
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```




```
# pie chart를 이용한 시각화
p_name_total = np.sum(df_salmon['CTRY_1'].value_counts().values)
p_name_ratio = df_salmon['CTRY_1'].value_counts() / p_name_total
p_name_ratio = p_name_ratio.sort_values(ascending = False)[0:-1]
p_name_order = p_name_ratio.index.tolist()

colors = sns.color_palette('tab10')[0:len(p_name_order)]
plt.figure(figsize = (15,15))
```

```
 9 plt.pie(p_name_ratio[p_name_order], labels = p_name_order, colors = colors, autopct='%.1f%%', te
10 plt.legend(p_name_order, loc = "lower right")
11 output.clear()
12 plt.show()
```




```
1 colors = sns.color_palette('hls',len(df_salmon['CTRY_2'].value_counts().index))
2
3 plt.figure(figsize = (15,15))
4 plt.pie(df_salmon['CTRY_2'].value_counts().values,
5          autopct='%.1f%%'
```

```
        autopct= %1.1f%% ,
        pctdistance=0.5,
        labeldistance=1.1,
        explode = [0.1, 0.1, 0.1, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1],
        textprops={'fontsize': 12},
        colors = colors,
        shadow = True)

plt.axis('equal')
plt.legend(df_salmon['CTRY_2'].value_counts().index)
plt.savefig('salmon_country.png',dpi=200)
plt.show()
```

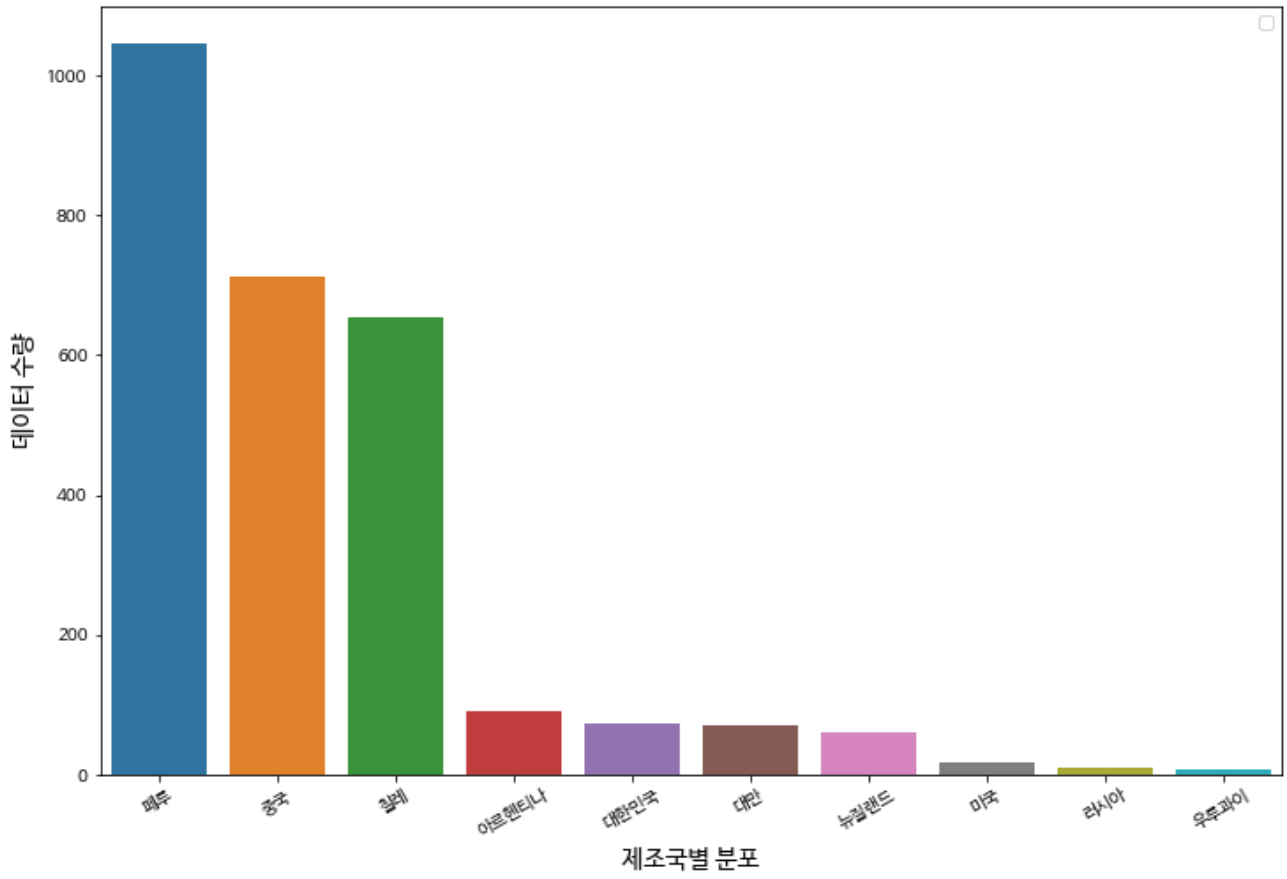

```
from scipy import stats
ctry_1_list = df_salmon.CTRY_1.value_counts().sort_values(ascending = False).index[0:3]

pvalue_dict = {};
for ctry in ctry_1_list:
    lst = []
    for ctry2 in ctry_1_list:
        if ctry == ctry2:
            pvalue = 0
            lst.append(pvalue)
        else:
            pvalue = stats.kruskal(df_salmon.loc[df_salmon.CTRY_1==ctry,'P_PRICE'].values, df_sa
            lst.append(pvalue)

    pvalue_dict[ctry] = lst

df_pvalue = pd.DataFrame(data = pvalue_dict, index = ctry_1_list)
display(df_pvalue)

## 등분산 검정
## 귀무가설 : 등분산이다. / 대립가설 : 이분산이다.
## 오징어의 경우 이분산
## 유의확률이 0.0000이므로 유의수준 0.05에서 이분산이다.
print("p-value: ", stats.levene(df_salmon.loc[df_salmon.CTRY_1=='노르웨이','P_PRICE'],df_salmon
print("p-value: ", stats.levene(df_salmon.loc[df_salmon.CTRY_1=='노르웨이','P_PRICE'],df_salmon
print("p-value: ", stats.levene(df_salmon.loc[df_salmon.CTRY_1=='영국','P_PRICE'],df_salmon.loc

pvalue_dict = {};
for ctry in ctry_1_list:
    lst = []
    for ctry2 in ctry_1_list:
        if ctry == ctry2:
            pvalue = 0
            lst.append(pvalue)
        else:
            pvalue = stats.levene(df_salmon.loc[df_salmon.CTRY_1==ctry,'P_PRICE'].values, df_sa
            lst.append(pvalue)

    pvalue_dict[ctry] = lst

df_pvalue = pd.DataFrame(data = pvalue_dict, index = ctry_1_list)
display(df_pvalue)
```

	노르웨이	영국	캐나다
노르웨이	0.000000e+00	1.316892e-40	1.470202e-20
영국	1.316892e-40	0.000000e+00	1.192667e-01
캐나다	1.470202e-20	1.192667e-01	0.000000e+00

p-value: 2.9889954350325333e-15

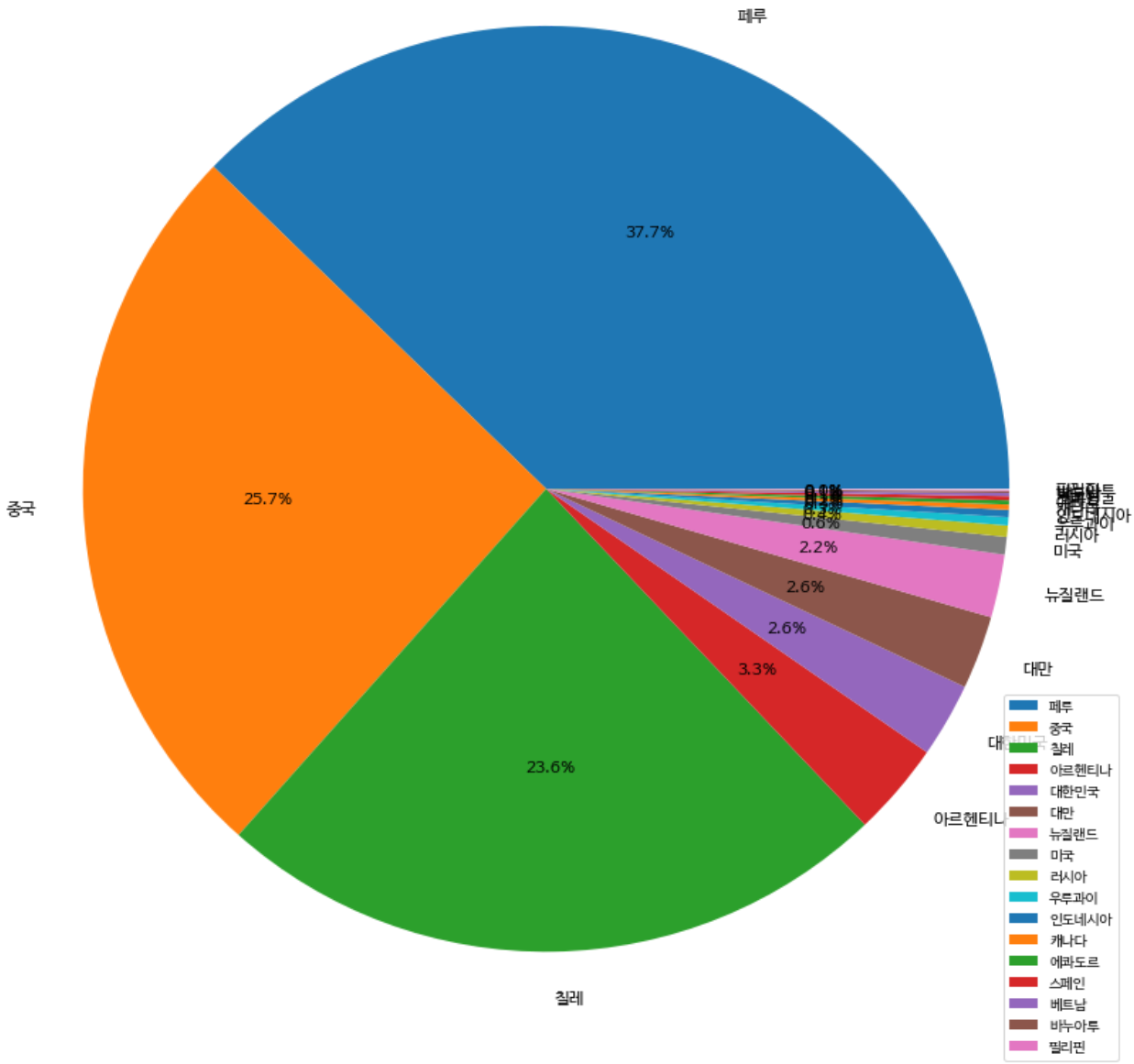
```
# 오징어 데이터에 대한 제조국별 분포

plt.figure(figsize = (12,8))
sns.countplot(x = 'CTRY_1', data = df_squid, order = df_squid['CTRY_1'].value_counts().keys()[0
plt.xticks(rotation = 30)
plt.xlabel("제조국별 분포", fontsize = 14)
plt.ylabel('데이터 수량', fontsize = 14)
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```




```
# pie chart를 이용한 시각화
p_name_total = np.sum(df_squid['CTRY_1'].value_counts().values)
p_name_ratio = df_squid['CTRY_1'].value_counts() / p_name_total
p_name_ratio = p_name_ratio.sort_values(ascending = False)[0:-1]
p_name_order = p_name_ratio.index.tolist()

colors = sns.color_palette('tab10')[0:len(p_name_order)]
plt.figure(figsize = (15,15))
```

```
plt.figure(figsize = (15,15))
plt.pie(p_name_ratio[p_name_order], labels = p_name_order, colors = colors, autopct='%.1f%%', te
plt.legend(p_name_order, loc = "lower right")
output.clear()
plt.show()
```




```
colors = sns.color_palette('hls',len(df_squid['CTRY_2'].value_counts().index))

plt.figure(figsize = (15,15))
plt.pie(df_squid['CTRY_2'].value_counts().values,
```

```
        autopct='%1.1f%%',
        pctdistance=0.5,
        explode = [0.1, 0.1, 0.1, 0.2, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1
        textprops={'fontsize': 12},
        colors = colors,
        shadow = True)

plt.axis('equal')
plt.legend(df_squid['CTRY_2'].value_counts().index, loc='upper right')
plt.savefig('squid_country.png',dpi=200)
plt.show()
```

페루

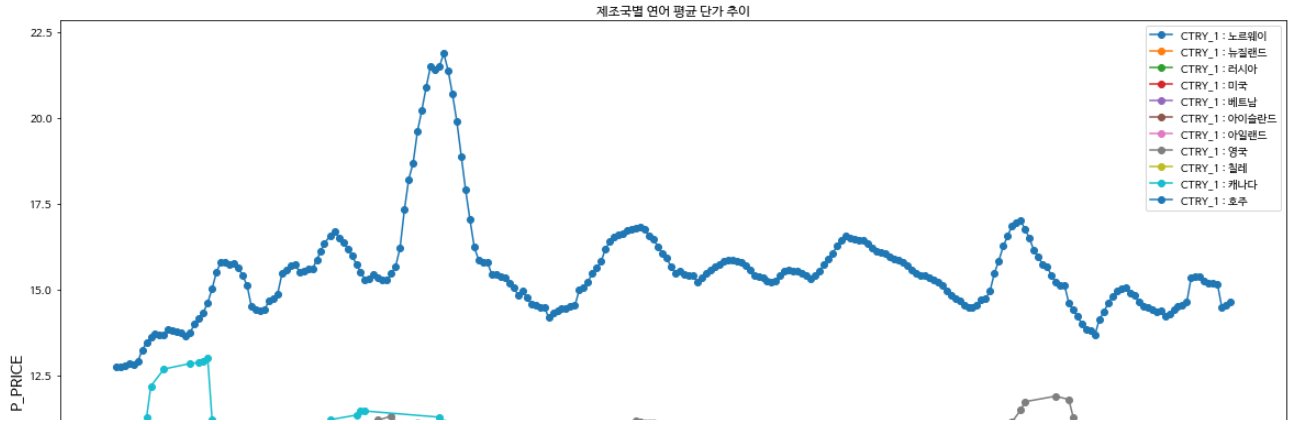
```
## 크루스칼 - 왈리스 순위합 검정(Kuscal-Wallis Rank Sum Test)
## k 집단중 하나라도 정규성 가정 깨질때 사용하는 검정방식
## 분석결과 통계적으로 유의한 차이있음
from scipy import stats
ctry_1_list = df_squid.CTRY_1.value_counts().sort_values(ascending = False).index[0:3]

pvalue_dict = {};
for ctry in ctry_1_list:
    lst = []
    for ctry2 in ctry_1_list:
        if ctry == ctry2:
            pvalue = 0
            lst.append(pvalue)
        else:
            pvalue = stats.kruskal(df_squid.loc[df_squid.CTRY_1==ctry,'P_PRICE'].values, df_squ
            lst.append(pvalue)

    pvalue_dict[ctry] = lst

df_pvalue = pd.DataFrame(data = pvalue_dict, index = ctry_1_list)
display(df_pvalue)


## 등분산 검정
## 귀무가설 : 등분산이다. / 대립가설 : 이분산이다.
## 오징어의 경우 이분산
## 유의확률이 0.0000이므로 유의수준 0.05에서 이분산이다.
print("p-value: ", stats.levene(df_squid.loc[df_squid.CTRY_1=='페루','P_PRICE'],df_squid.loc[df_
print("p-value: ", stats.levene(df_squid.loc[df_squid.CTRY_1=='페루','P_PRICE'],df_squid.loc[df_
print("p-value: ", stats.levene(df_squid.loc[df_squid.CTRY_1=='중국','P_PRICE'],df_squid.loc[df_

pvalue_dict = {};
for ctry in ctry_1_list:
    lst = []
    for ctry2 in ctry_1_list:
        if ctry == ctry2:
            pvalue = 0
            lst.append(pvalue)
        else:
            pvalue = stats.levene(df_squid.loc[df_squid.CTRY_1==ctry,'P_PRICE'].values, df_squi
            lst.append(pvalue)

    pvalue_dict[ctry] = lst

df_pvalue = pd.DataFrame(data = pvalue_dict, index = ctry_1_list)
display(df_pvalue)
```

	페루	중국	칠레
페루	0.000000e+00	3.463080e-01	1.115438e-47
중국	3.463080e-01	0.000000e+00	1.888171e-64
칠레	1.115438e-47	1.888171e-64	0.000000e+00

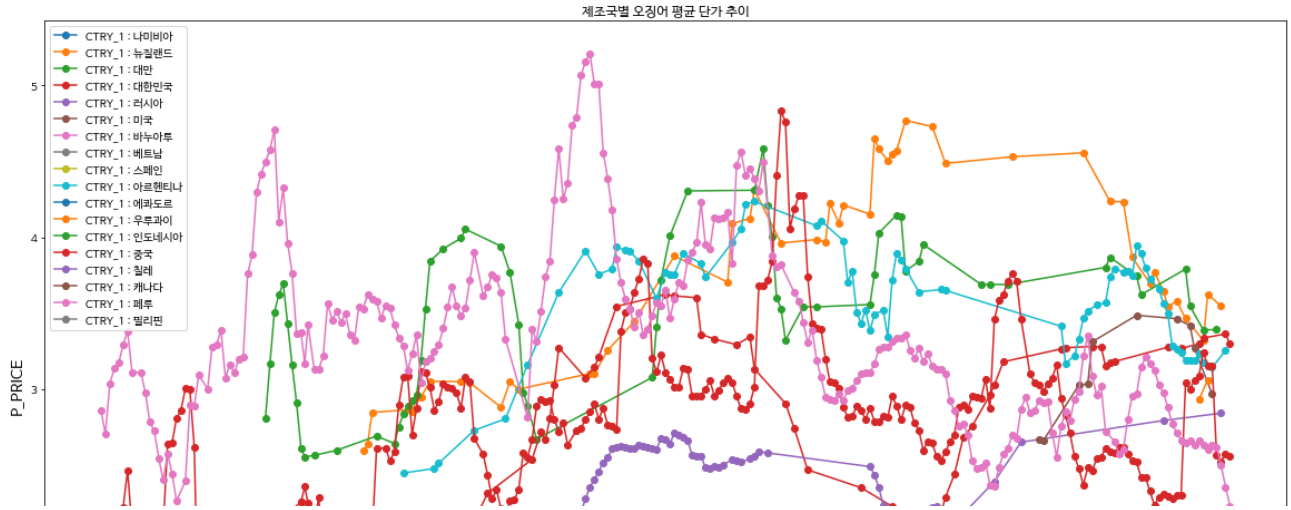
```
p-value:  9.41192405025777e-16
p-value:  2.5943164924289678e-54
p-value:  3.17089324524164e-10
```

```
페루
중국
칠레
```

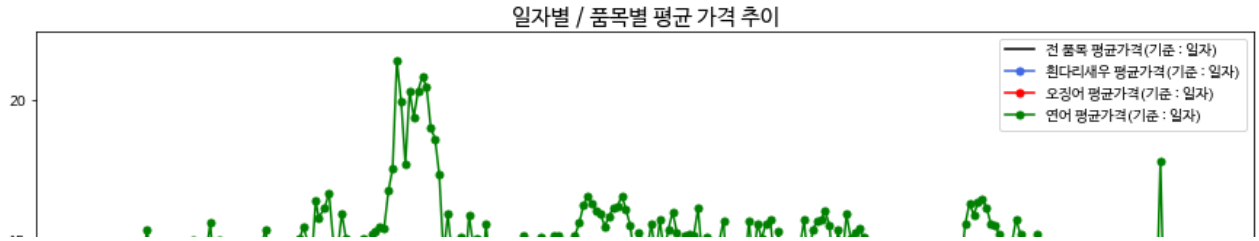
```
# 연어에 대한 제조국별 평균 단가의 차이 비교
df_salmon_groupby = df_salmon.groupby(by = 'CTRY_1')
plt.figure(1, figsize = (20,12))

for key, group in df_salmon_groupby:
    plt.figure(1)
    group_series = group.groupby(by = 'REG_DATE').mean().rolling(7).mean()
    plt.plot(group_series['P_PRICE'], label = "CTRY_1 : " + key, marker = 'o')

plt.xlabel('DATE', fontsize = 14)
plt.ylabel('P_PRICE', fontsize = 14)
plt.title("제조국별 연어 평균 단가 추이")
plt.xticks(rotation = 0)
plt.legend()
plt.show()
```




```
# 오징어에 대한 제조국별 평균 단가의 차이 비교
df_squid_groupby = df_squid.groupby(by = 'CTRY_1')
df_squid_grouped_lst = []
plt.figure(1, figsize = (20,12))

for key, group in df_squid_groupby:
    plt.figure(1)
    group_series = group.groupby(by = 'REG_DATE').mean().rolling(7).mean()
    df_squid_grouped_lst.append(group_series)
    plt.plot(group_series['P_PRICE'], label = "CTRY_1 : " + key, marker = 'o')

plt.xlabel('DATE', fontsize = 14)
plt.ylabel('P_PRICE', fontsize = 14)
plt.title("제조국별 오징어 평균 단가 추이")
plt.xticks(rotation = 0)
plt.legend()
plt.show()
```




```
# 전체적인 월별 평균 가격 흐름(새우, 오징어, 연어)
import matplotlib.dates as md
xfmt = md.DateFormatter('%Y-%m-%d')

fig, ax1 = plt.subplots(figsize=(15,8))
df_total_price_mean = df_train.groupby(by = ['REG_DATE']).mean().rolling(7).mean()
df_shrimp_price_mean = df_shrimp.groupby(by = ['REG_DATE']).mean()
df_squid_price_mean = df_squid.groupby(by = ['REG_DATE']).mean()
df_salmon_price_mean = df_salmon.groupby(by = ['REG_DATE']).mean()

plt.plot(df_total_price_mean['P_PRICE'], ms = 5, color = 'k', label = '전 품목 평균가격(기준 : '
plt.plot(df_shrimp_price_mean['P_PRICE'], marker = 'o', ms = 5, color = 'royalblue', label = '흰
plt.plot(df_squid_price_mean['P_PRICE'], marker = 'o', ms = 5, color = 'r', label = '오징어 평균
plt.plot(df_salmon_price_mean['P_PRICE'], marker = 'o', ms = 5, color = 'g', label = '연어 평균
plt.xlabel("Date", fontsize = 13)
plt.ylabel("Price", fontsize = 13)
plt.title("일자별 / 품목별 평균 가격 추이", fontsize = 15)
plt.legend(loc = 'upper right')
plt.show()
```



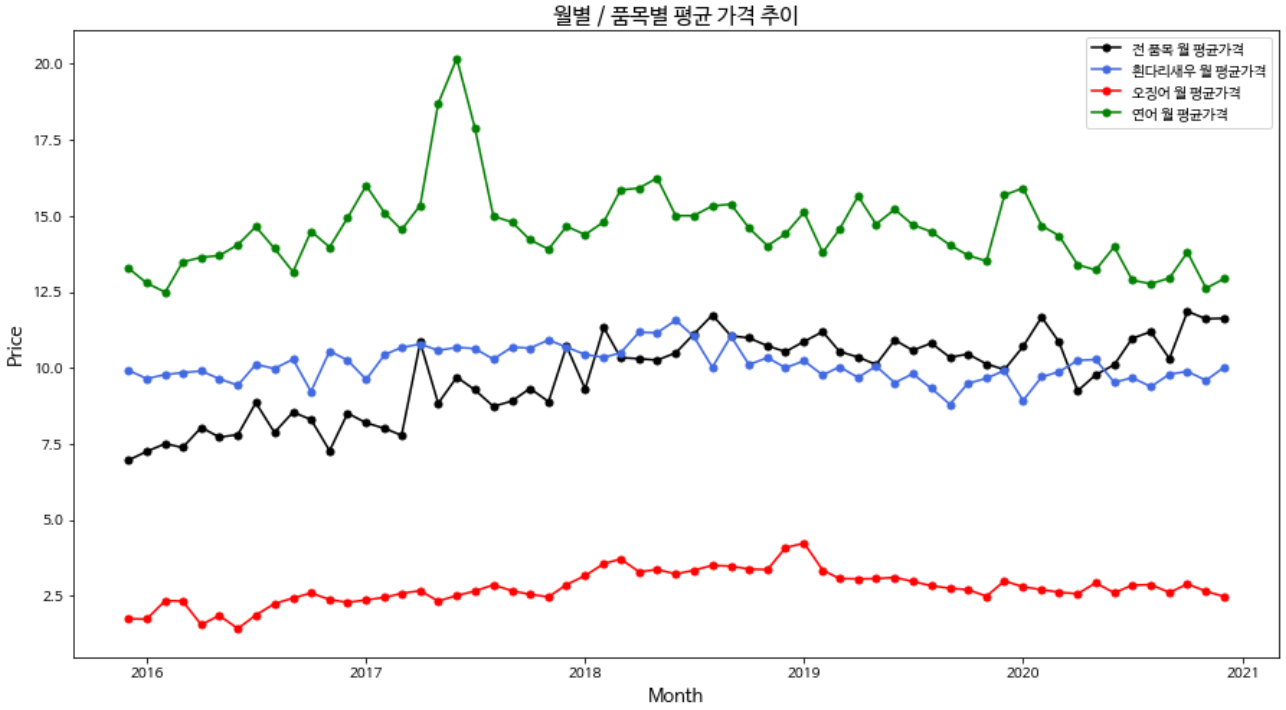

```
# 제조국별 평균 단가 추이를 비교해보자
```

```
# 월별 / 연도별 평균 가격
## 날짜별로 시각화 및 데이터 확인 하기 위해 변수 추가
date = pd.to_datetime(df_train.REG_DATE)
df_train['year'] = date.dt.year ## 몇년
df_train['month'] = date.dt.month ## 몇월
df_train['day'] = date.dt.weekday ## 몇요일 ## 월요일 = 0, 화요일 1, 일요일 = 6
df_train['week'] = date.dt.weekofyear ## 몇번째 주인지
df_train['holiday'] = df_train.apply(lambda x : 0 if x['day']<5 else 1, axis = 1) ## 주말이면 1

## 월별로 그룹화 하기 위해 변수 생성
df_train['year_month']=pd.to_datetime(df_train.REG_DATE).dt.to_period('1M')

df_squid = df_train[df_train['P_NAME'] == '오징어']
df_salmon = df_train[df_train['P_NAME'] == '연어']
df_shrimp = df_train[df_train['P_NAME'] == '흰다리새우']

## 월별 평균으로 그룹화
## 전체
df_train_group = df_train.groupby(by=['year_month']).agg({'P_PRICE':'mean'}).reset_index().sort_
df_train_group.index=df_train_group['year_month']

## 오징어
df_squid_group=df_squid.groupby(by=['year_month']).agg({'P_PRICE':'mean'}).reset_index().sort_va
df_squid_group.index = df_squid_group['year_month']

## 연어
df_salmon_group=df_salmon.groupby(by=['year_month']).agg({'P_PRICE':'mean'}).reset_index().sort_
df_salmon_group.index=df_salmon_group['year_month']

## 흰다리 새우
df_shrimp_group = df_shrimp.groupby(by=['year_month']).agg({'P_PRICE':'mean'}).reset_index().sor
df_shrimp_group.index=df_shrimp_group['year_month']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: FutureWarning: Series.dt.weel
  import sys
```

```
import matplotlib.dates as md
xfmt = md.DateFormatter('%Y-%m-%d')
fig, ax1 = plt.subplots(figsize=(15,8))

x_month = pd.to_datetime(df_train_group.index.strftime("%Y-%m"))

plt.plot(x_month, df_train_group['P_PRICE'].values, marker = 'o', ms = 5, color = 'k', label =
plt.plot(x_month, df_shrimp_group['P_PRICE'].values, marker = 'o', ms = 5, color = 'royalblue',
```

```
plt.plot(x_month, df_squid_group['P_PRICE'].values, marker = 'o', ms = 5, color = 'r', label =
plt.plot(x_month, df_salmon_group['P_PRICE'].values, marker = 'o', ms = 5, color = 'g', label =
plt.xlabel("Month", fontsize = 13)
plt.ylabel("Price", fontsize = 13)
plt.title("월별 / 품목별 평균 가격 추이", fontsize = 15)
plt.legend(loc = 'upper right')
plt.show()
```




```
## 표본 수가 50000이하이면 샤피로-윌크 테스트(Shapiro-Wilk Test) 진행  두번째값이 0.05보다 크면
## 정규성 만족하면 일표본 t-test/ 정규성 불만족이면 윌콕슨의 부호 순위검정
from scipy import stats

## 오징어 가격 정규성 검정
shapiro_test_squid = stats.shapiro(df_squid_price_mean['P_PRICE'].values)
shapiro_test_salmon = stats.shapiro(df_salmon_price_mean['P_PRICE'].values)
shapiro_test_shrimp = stats.shapiro(df_shrimp_price_mean['P_PRICE'].values)

print(shapiro_test_squid)
print(shapiro_test_salmon)
print(shapiro_test_shrimp)
```

```
(0.9440547823905945, 2.1036186126366374e-08)
(0.9132353067398071, 3.444335094915374e-11)
(0.9924631714820862, 0.204646036028862)
```

```
# t-test를 이용한 이상치 판별
## boxplot을 이용한 이상치 판별을 먼저 한다
```

```
## boxplot을 이용한 이상치 판별을 한지 한다.
```

```
# 수출국별 품목 이상치(새우)
comparison_data = df_shrimp

figure = plt.figure(figsize=(20, 8))
sns.boxplot(x='CTRY_1', y='P_PRICE',data=comparison_data,
            meanprops={"marker":"s","markerfacecolor":"black", "markeredgecolor":"black"},boxpro

sns.swarmplot(x='CTRY_1', y='P_PRICE',data=comparison_data)
plt.title("수출국가별 가격 분포 (항목 : 흰다리새우)", fontsize = 15)
plt.ylabel("P_PRICE")
plt.xlabel("수출국가", fontsize = 13)
plt.xticks(rotation = 0)
plt.show()
```

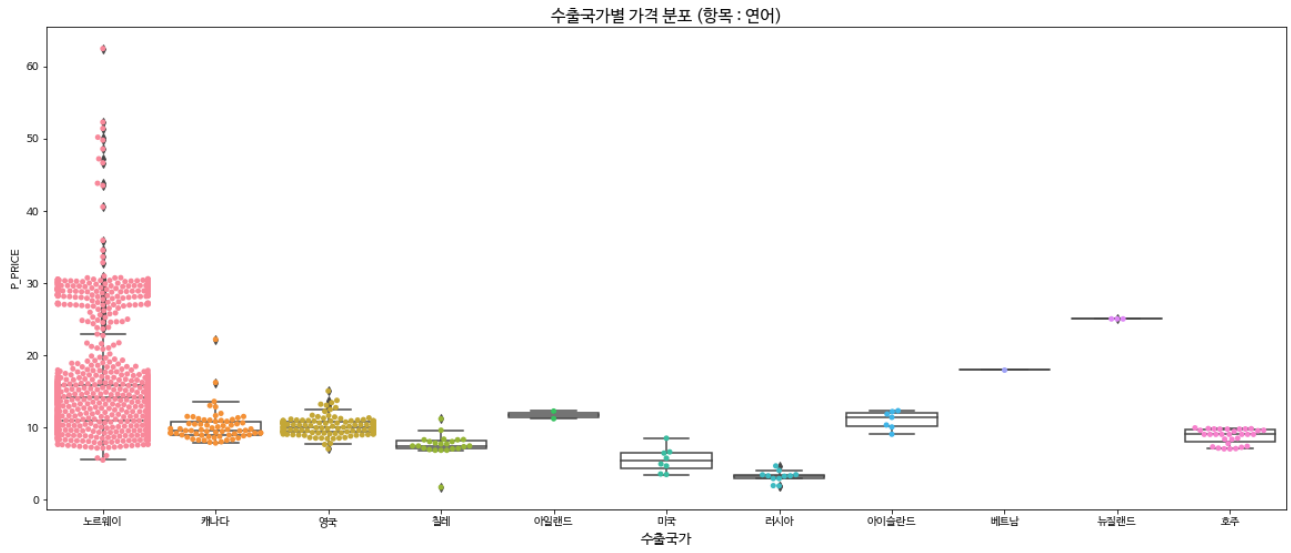
```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 53.6% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 26.2% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 50.5% of th
```

```
comparison_data = df_squid

figure = plt.figure(figsize=(20, 8))
sns.boxplot(x='CTRY_1', y='P_PRICE',data=comparison_data,
            meanprops={"marker":"s","markerfacecolor":"black", "markeredgecolor":"black"},boxpr

sns.swarmplot(x='CTRY_1', y='P_PRICE',data=comparison_data)
plt.title("수출국가별 가격 분포 (항목 : 오징어)", fontsize = 15)
plt.ylabel("P_PRICE")
plt.xlabel("수출국가", fontsize = 13)
plt.xticks(rotation = 0)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 7.0% of the
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 76.3% of th
```

```
comparison_data = df_salmon
figure = plt.figure(figsize=(20, 8))
sns.boxplot(x='CTRY_1', y='P_PRICE',data=comparison_data,
            meanprops={"marker":"s","markerfacecolor":"black", "markeredgecolor":"black"},boxpr

sns.swarmplot(x='CTRY_1', y='P_PRICE',data=comparison_data)
plt.title("수출국가별 가격 분포 (항목 : 연어)", fontsize = 15)
plt.ylabel("P_PRICE")
plt.xlabel("수출국가", fontsize = 13)
plt.xticks(rotation = 0)
plt.show()
```

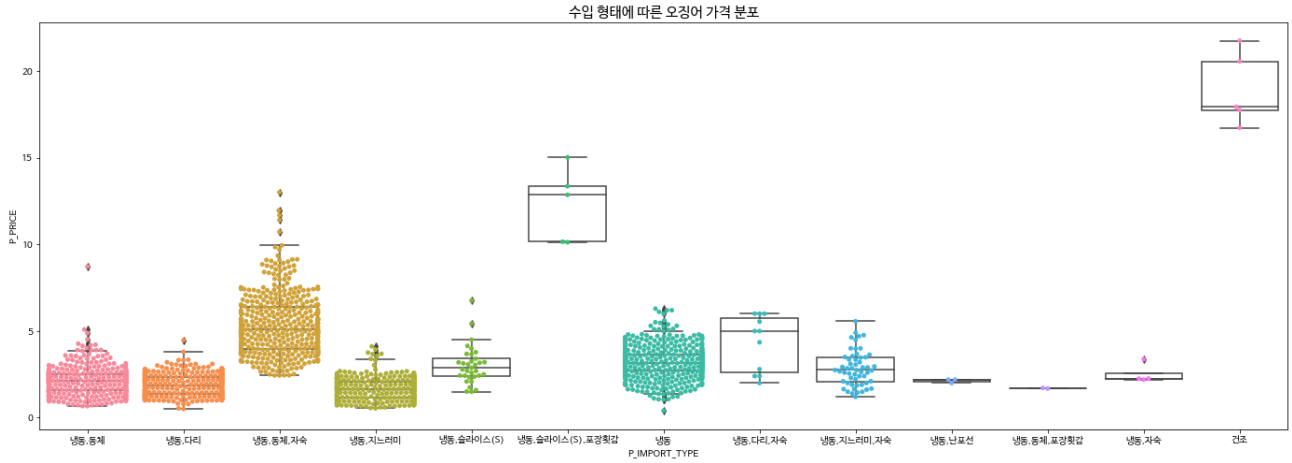
```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 75.6% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 6.1% of the
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 46.9% of th
  warnings.warn(msg, UserWarning)
```




```
# 오징어 가격 분포
figure = plt.figure(figsize=(24, 8))
sns.boxplot(x='P_IMPORT_TYPE', y='P_PRICE',data = df_squid,
            meanprops={"marker":"s","markerfacecolor":"black", "markeredgecolor":"black"},boxpr
```

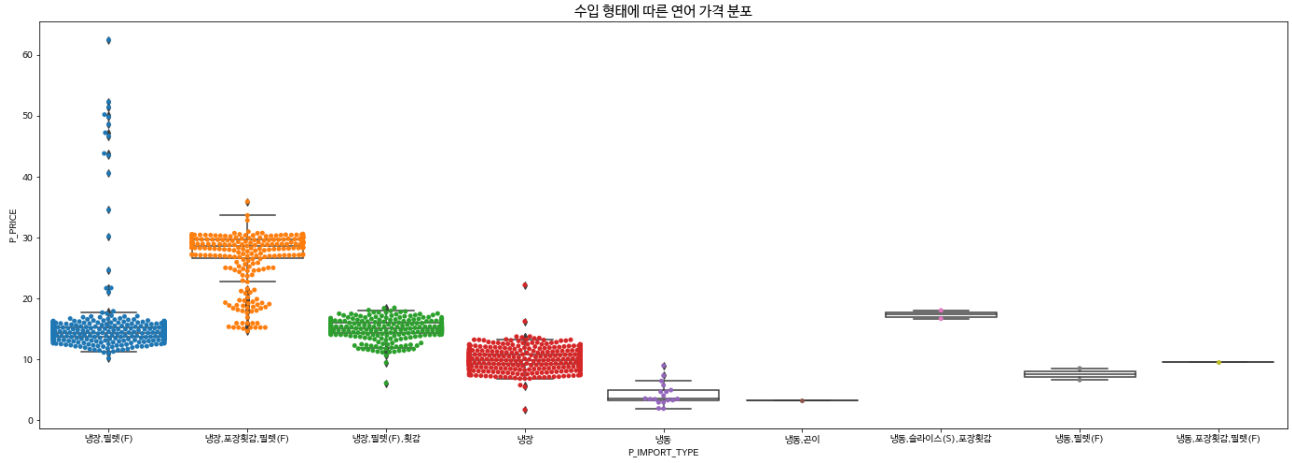
```
sns.swarmplot(x='P_IMPORT_TYPE', y='P_PRICE',data=df_squid)
plt.xticks(rotation = 0, fontsize = 10)
plt.title("수입 형태에 따른 오징어 가격 분포", fontsize = 15)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 67.3% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 72.7% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 29.0% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 72.3% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 58.9% of th
  warnings.warn(msg, UserWarning)
```



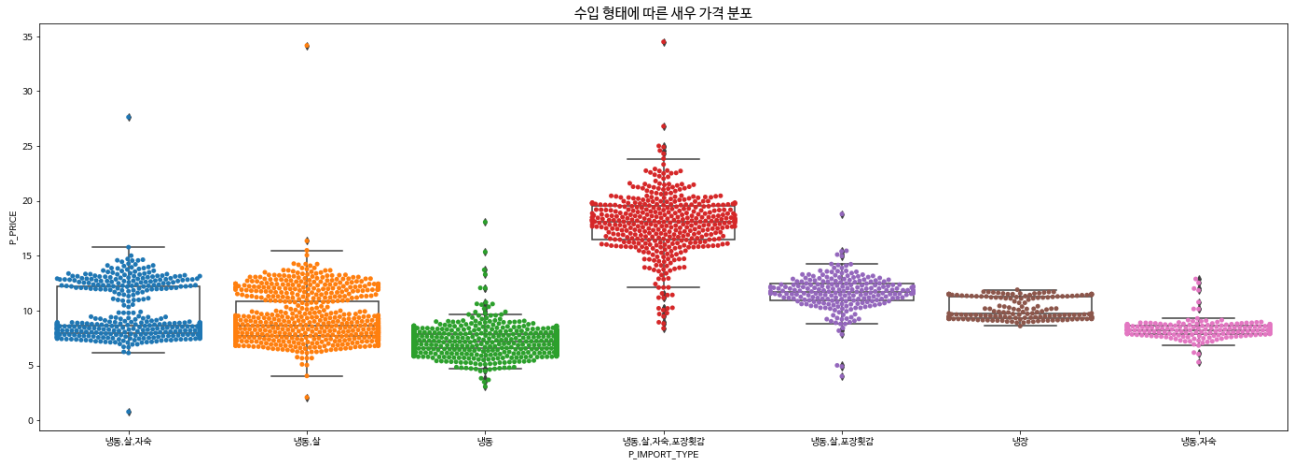

```
# 연어 가격 분포
figure = plt.figure(figsize=(24, 8))
sns.boxplot(x='P_IMPORT_TYPE', y='P_PRICE',data = df_salmon,
            meanprops={"marker":"s","markerfacecolor":"black", "markeredgecolor":"black"},boxpro
sns.swarmplot(x='P_IMPORT_TYPE', y='P_PRICE',data=df_salmon)
plt.xticks(rotation = 0, fontsize = 10)
plt.title("수입 형태에 따른 연어 가격 분포", fontsize = 15)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 64.2% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 31.4% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 40.4% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 72.7% of th
  warnings.warn(msg, UserWarning)
```




```
# 흰다리새우 가격 분포
figure = plt.figure(figsize=(24, 8))
sns.boxplot(x='P_IMPORT_TYPE', y='P_PRICE',data = df_shrimp,
            meanprops={"marker":"s","markerfacecolor":"black", "markeredgecolor":"black"},boxpro
sns.swarmplot(x='P_IMPORT_TYPE', y='P_PRICE',data=df_shrimp)
plt.title("수입 형태에 따른 새우 가격 분포", fontsize = 15)
plt.xticks(rotation = 0, fontsize = 10)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 27.4% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 38.3% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 64.8% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 6.2% of the
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 6.1% of the
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 24.4% of th
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:1296: UserWarning: 27.0% of th
  warnings.warn(msg, UserWarning)
```




```
path = glob.glob("./외부데이터/*")
path[1].split("/")[-1].split(" ")[0]
path
```

```
['./외부데이터/USD_THB 내역.csv',
 './외부데이터/USD_MYR 내역.csv',
 './외부데이터/USD_CAD 내역.csv',
 './외부데이터/국제유가2021-09-09.csv',
 './외부데이터/USD_PEN 내역.csv',
 './외부데이터/USD_VND 내역.csv',
 './외부데이터/USD_KRW 내역.csv',
 './외부데이터/USD_NOK 내역.csv',
 './외부데이터/USD_GBP 내역.csv',
 './외부데이터/USD_CLP 내역.csv',
 './외부데이터/USD_CNH 내역.csv']
```

```
# 환율 데이터 병합(사용 데이터 : )
import datetime
from tqdm import tqdm

df_rate = None
for i in tqdm(range(0,len(glob.glob("./외부데이터/*")))):
    path = glob.glob("./외부데이터/*")[i]
    df = pd.read_csv(path,)
    var_name = path.split("/")[-1].split(" ")[0]


    if var_name == "국제유가2021-09-09.csv":
        continue
```

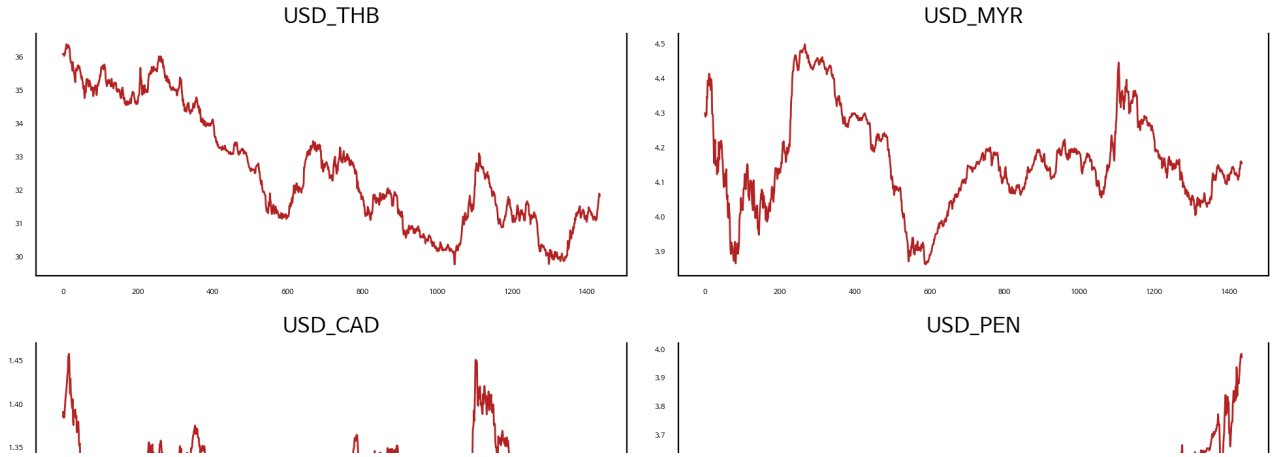
```
    # 헤더 삭제
    df =df.drop(axis = 0, index = 0)

    # rename columne
    df = df.rename(columns = {"날짜":"DATE", "종가": var_name})

    # sort
    df = df.sort_values(by = ['DATE'])[['DATE', var_name]].reset_index(drop = True)
    df.DATE = df.DATE.apply(lambda x : datetime.datetime.strptime(x,'%Y년 %m월 %d일'))

    def preprocessing(x):

        if type(x) == str:
            x_tmp = x.replace(",", "")
            x_tmp = float(x_tmp)
            return x_tmp
        else:
            return x

    df[var_name] = df[var_name].apply(lambda x : preprocessing(x))

    if df_rate is None:
        df_rate = df[["DATE", var_name]]
    else:
        df_rate_copy = df_rate.copy(...)
        df_rate = pd.merge(df_rate_copy, df[["DATE", var_name]], left_on = 'DATE', right_on = '[

    df_rate = df_rate.interpolate(method = "linear", axis = 0)

display(df_rate)
```

```
100%|██████████| 11/11 [00:00<00:00, 33.22it/s]
```

	DATE	USD_THB	USD_MYR	USD_CAD	USD_PEN	USD_VND	USD_KRW	USD_NOK	USD_GBI
0	2015-12-28	36.090	4.2985	1.3906	3.3978	22532.0	1168.85	8.7013	0.672
1	2015-12-29	36.080	4.2885	1.3842	3.4050	22480.0	1171.56	8.7193	0.675
2	2015-12-30	36.090	4.2920	1.3879	3.4095	22445.0	1177.51	8.7928	0.674
3	2015-12-31	36.030	4.2935	1.3841	3.4145	22485.0	1175.95	8.8423	0.678
4	2016-01-01	36.045	4.2935	1.3849	3.4095	22485.0	1175.45	8.8683	0.678
...	...	...	...	...	...	...	...	...	.
1430	2021-06-21	31.590	4.1450	1.2359	3.9518	23014.0	1131.40	8.5741	0.717
1431	2021-06-22	31.710	4.1595	1.2305	3.9788	23025.0	1133.57	8.5358	0.716
	2021								

```
# Plot
```

```
# Plot
fig, axes = plt.subplots(nrows=5, ncols=2, dpi=200, figsize=(10,12))
for i, ax in enumerate(axes.flatten()):
    data = df_rate[df_rate.columns[i+1]]
    ax.plot(data, color='firebrick', linewidth=1)
    # Decorations
    ax.set_title(df_rate.columns[i+1])
    ax.xaxis.set_ticks_position('none')
    ax.yaxis.set_ticks_position('none')
    ax.spines["top"].set_alpha(0)
    ax.tick_params(labelsize=4)

plt.tight_layout();
plt.savefig('International_rate.png',dpi=200)
```




```
1 display(df_external)
```

	Date	WTi	Brent	Dubai
0	2021-06-28	72.91	74.68	73.88
1	2021-06-25	74.05	76.18	73.46
2	2021-06-24	73.30	75.56	73.73
3	2021-06-23	73.08	75.19	73.43
4	2021-06-22	73.06	74.81	72.52
...	...	...	...	...
1406	2016-01-04	36.76	37.22	32.54
1407	2015-12-31	37.04	37.28	32.19
1408	2015-12-30	36.60	36.46	32.74
1409	2015-12-29	37.87	37.79	32.11
1410	2015-12-28	36.81	36.62	32.61

1411 rows × 4 columns

```
1 plt.figure(figsize = (12,8))
2 plt.plot(df_external.Date, df_external.WTi, color = 'firebrick', linewidth = 1.5, label = 'WTi'
3 plt.plot(df_external.Date, df_external.Brent, color = 'steelblue',linewidth = 1.5, label = 'Bre
4 plt.plot(df_external.Date, df_external.Dubai, color = 'forestgreen', linewidth = 1.5, label = '
5
6 plt.legend(loc = 'upper left',fontsize=12)
7 plt.grid(True, axis='y',linestyle='--')
8 plt.xticks(size=12)
9 plt.yticks(size=12)
10
11 plt.title("연간 국제 유가 동향", fontsize = 15)
12 plt.gca().spines['right'].set_visible(False)
13 plt.gca().spines['left'].set_visible(False)
14 plt.gca().spines['top'].set_visible(False)
15 plt.savefig('df_external_trend.png',dpi=200)
16 plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarnin
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarnin
  font.set_text(s, 0, flags=flags)
```


```
# 외부 데이터 병합
```

```
df_external = df_external.sort_values(by = ['Date'])[['Date', 'WTi', 'Brent', 'Dubai']].reset_i

import datetime

df_external.Date = df_external.Date.apply(lambda x : datetime.datetime.strptime(x, '%Y-%m-%d'))
display(df_external)
```

	Date	WTi	Brent	Dubai
0	2015-12-28	36.81	36.62	32.61
1	2015-12-29	37.87	37.79	32.11
2	2015-12-30	36.60	36.46	32.74
3	2015-12-31	37.04	37.28	32.19

```
# 변수 -> 정수 인덱스화
ctry_1_unique = np.unique(df_train['CTRY_1'].values)
ctry_2_unique = np.unique(df_train['CTRY_2'].values)

print("WnCTRY_1 카테고리 : ", ctry_1_unique)
print("WnCTRY_2 카테고리 : ", ctry_2_unique)

p_purpose = np.unique(df_train['P_PURPOSE'].values)
print("WnP_PURPOSE 카테고리 : ", p_purpose)

category_1 = np.unique(df_train['CATEGORY_1'].values)
category_2 = np.unique(df_train['CATEGORY_2'].values)

print('WnCATEGORY_1 카테고리 : ', category_1)
print('WnCATEGORY_2 카테고리 : ', category_2)

p_name = np.unique(df_train['P_NAME'].values)
print('WnP_NAME 카테고리 : ', p_name)

p_import_type = np.unique(df_train['P_IMPORT_TYPE'].values)
print('WnP_IMPORT_type 카테고리 : ', p_import_type)
```

```
CTRY_1 카테고리 :  ['가나' '감비아' '그리스' '그린란드' '기니' '기니비사우' '나미비아' '남
 '뉴질랜드' '니카라과' '대만' '대한민국' '덴마크' '라이베리아' '라트비아' '러시아' '루마니
 '마다가스카르' '말레이시아' '멕시코' '모로코' '모리타니' '모잠비크' '몰타' '미국' '미얀마
 '바누아투' '바레인' '방글라데시' '베네수엘라' '베트남' '불가리아' '브라질' '사우디아라비
 '세이셸' '세인트빈센트 그레나딘' '소말리아' '수리남' '스리랑카' '스페인' '시에라리온' '싱
 '아르헨티나' '아이슬란드' '아일랜드' '알제리' '앙골라' '에스토니아' '에콰도르' '영국' '예
 '우크라이나' '이란' '이집트' '이탈리아' '인도' '인도네시아' '일본' '중국' '칠레' '캐나다'
 '콜롬비아' '콩고 민주 공화국' '쿠바' '쿡 제도' '크로아티아' '키리바시' '태국' '터키' '튀니
 '파키스탄' '파푸아뉴기니' '팔라우' '페루' '포르투갈' '포클랜드 제도' '프랑스' '피지' '필리

CTRY_2 카테고리 :  ['가나' '감비아' '그리스' '기니' '기니비사우' '기타(ZZ)' '나미비아' '남
 '뉴질랜드' '대만' '덴마크' '독일' '라이베리아' '러시아' '루마니아' '마다가스카르' '말레이
 '모리셔스' '모리타니' '모잠비크' '몰타' '미국' '미얀마' '미크로네시아 연방' '바누아투' '바
 '베네수엘라' '베트남' '벨리즈' '북한' '불가리아' '브라질' '사모아' '사우디아라비아' '세네
 '솔로몬 제도' '수리남' '스리랑카' '스웨덴' '스페인' '시에라리온' '싱가포르' '아랍에미리트
 '아일랜드' '앙골라' '에콰도르' '영국' '예멘' '오만' '우루과이' '우크라이나' '이란' '이집트
 '인도네시아' '일본' '중국' '지부티' '칠레' '캐나다' '코트디부아르' '콜롬비아' '콩고 민주
 '크로아티아' '키리바시' '태국' '터키' '투발루' '튀니지' '파나마' '파키스탄' '파푸아뉴기니
 '포르투갈' '포클랜드 제도' '프랑스' '피지' '필리핀' '호주' '홍콩']

P_PURPOSE 카테고리 :  ['반송품(기타)' '반송품(외화획득용원료)' '외화획득용 원료' '외화획득

CATEGORY_1 카테고리 :  ['갑각류' '기타 수입식품' '알 곤이류' '어류' '연체류 해물모듬' '젓갈

CATEGORY_2 카테고리 :  ['가리비' '가물치' '가사리' '가오리' '가자미' '가재 랍스타' '갈치'
 '고등어' '고시래기' '골뱅이' '광어 넙치' '굴' '김' '꼬막' '꼴뚜기' '꽁치 학꽁치' '꽁치 확
```

```
'날치알' '남극빙어' '노래미' '농어' '능성어 붉바리 바리' '다시마' '달고기' '대구' '대구알
'도미 감성돔 돔류' '만세기' '망둑어' '멍게' '메기 동자개' '메로' '멸치' '명란(명태알)' '명
'물메기(곰치)' '미꾸라지' '미역' '민물붕어' '민어 점성어' '밀크피시' '바지락' '방어' '밴
'벤자리 알롱이' '병어' '보리멸' '복어' '부세' '붉평치(만다이 꽃돔)' '삼치' '상어 고래' '
'서대 박대 페루다' '성게알' '소라' '송어' '쏘가리' '아귀' '양미리 정어리' '양태' '어류 기
'열빙어(시샤모)' '열빙어(시샤모)알' '오징어' '옥돔' '우럭 볼락' '우렁 다슬기' '은민대구알
'자라' '장어' '재첩' '적어 눈볼대' '전갱기 매가리' '전갱이 매가리' '전복' '전어' '조개' '
'조기 보구치 강다리' '조기 보구치 강다리 ' '준치' '줄비늘치' '쥐치' '쭈꾸미' '참치 새치류
'청어알' '톳' '틸라피아(역돔)' '팡가시우스(홍메기)' '해물모둠' '해삼' '해초' '해파리' '호
'호키(새꼬리민태)알' '호키류' '홍어' '홍합']

P_NAME 카테고리 : ['PANGASIUS메기' 'Plagioscion squamosissimus' '가다랑어' '가라지' '가리
'가물치' '가시배새우' '가시투성왕게' '가오리' '가이석태속' '가자미' '각시가자미' '갈치' '
'강담돔' '강도다리' '개량조개' '개복치' '개불' '개조개' '갯고둥' '갯장어' '검복' '검정가
'검정볼락' '게' '게,Cancer pagurus' '고등어' '골뱅이' '곱사연어' '곱상어' '구라미' '굴' '
'금눈돔' '금색돔' '기름가자미' '기름치' '기타민어류' '기타병어류' '긴가라지' '긴가이석태'
'까지가자미속,Lepidopsetta polyxystra' '까치복' '까칠복' '깜장북방대합' '꼬리검정민태' '
'꼬시래기' '꼬치삼치' '꽁치' '꽃게' '낙지' '날개다랑어' '날치알' '남방걸장어' '남방대구' '
'녹새치' '논고둥' '농어' '눈다랑어' '능성어' '다슬기' '다시마' '달고기' '닭새우' '대게' '
'대두이석태' '대서양꼬마민어' '대서양먹장어' '대서양붉은볼락' '대서양연어' '대서양조기속'
'도화새우' '돌가사리' '돌가자미' '돌돔' '돔' '동갈돗돔' '동갈횟대' '동등이석태' '동자개' '
'두점박이민꽃게' '드렁허리' '등목어' '마설가자미' '마소치가자미' '만새기' '맛조개' '망둑어
'먹볼락' '먹장어' '멍게' '메기' '멸치' '명태' '명태알' '몽치다래' '문어' '물메기' '미꾸라
'민대구' '민들조개' '민물가재' '민물새우' '민밀복' '민어' '민태' '밀크피시' '바다가재' '바
'바리' '바리,교잡종' '바지락' '밤색무늬조개' '방어' '백합' '백합,MERCENARIA MERCENARIA' '
'뱀장어' '버들붕어' '버터플라이 킹피쉬' '벤자리' '벵에돔' '병어' '병치매가리' '보리멸' '볼
'볼락,Sebastes viviparus' '부세' '부시리' '북방대합' '북쪽분홍새우' '붉돔' '붉은대게' '붉
'붉은이석태' '붉평치' '붕어' '붕장어' '블루화이팅' '비너스백합' '비단조개' '뿔가자미' '사
'새꼬리민태알' '새꼬막' '새뱅이' '새우' '새조개' '샛돔' '샛돔류알' '서대' '성게알' '세네갈
'소주목탁가자미' '송어' '수조기' '스피노잠' '식용자라' '실꼬리돔' '쌍지붕어' '쏘가리' '아
'아르헨티나붉은새우' '양볼락' '양초선홍치' '양태' '어름돔' '얼룩볼락' '얼룩새우' '연어' '
```

```
1 # data merge
2 # df_train + df_external
3 df_train_join = pd.merge(df_train, df_external, left_on = 'REG_DATE', right_on = 'Date', how =
4 display(df_train_join)
5
6 df_train = df_train_join.copy()
```

	REG_DATE	P_TYPE	CTRY_1	CTRY_2	P_PURPOSE	CATEGORY_1	CATEGORY_2	P_NAME
0	2015-12-28	수산물	아르헨티나	아르헨티나	판매용	갑각류	새우	아르헨티나붉은새우
1	2015-12-28	수산물	바레인	바레인	판매용	갑각류	게	꽃게
2	2015-12-28	수산물	바레인	바레인	판매용	갑각류	게	꽃게
3	2015-12-28	수산물	칠레	칠레	판매용	패류 멍게류	해삼	해심
4	2015-12-28	수산물	중국	중국	판매용	어류	서대 박대 페루다	서대

```
1 df_train_join = pd.merge(df_train, df_rate, left_on = 'REG_DATE', right_on = 'DATE', how = 'lef
2 display(df_train_join)
3
4 df_train = df_train_join.copy()
```

	REG_DATE	P_TYPE	CTRY_1	CTRY_2	P_PURPOSE	CATEGORY_1	CATEGORY_2	P_NAME
0	2015-12-28	수산물	아르헨티나	아르헨티나	판매용	갑각류	새우	아르헨티나붉은새우
1	2015-12-28	수산물	바레인	바레인	판매용	갑각류	게	꽃게
2	2015-12-28	수산물	바레인	바레인	판매용	갑각류	게	꽃게
3	2015-12-28	수산물	칠레	칠레	판매용	패류 멍게류	해삼	해심
4	2015-12-28	수산물	중국	중국	판매용	어류	서대 박대 페루다	서대
...	...	...	...	...	...	...	...	..
49921	2020-12-28	수산물	중국	중국	판매용	연체류 해물모듬	낙지	낙지
49922	2020-12-28	수산물	일본	일본	판매용	어류	방어	방어
49923	2020-12-28	수산물	노르웨이	노르웨이	판매용	어류	고등어	고등어
49924	2020-12-28	수산물	중국	중국	판매용	연체류 해물모듬	낙지	낙지
49925	2020-12-28	수산물	노르웨이	노르웨이	판매용	어류	연어	연어

49926 rows × 29 columns

```
1 df_train = df_train.dropna()
```

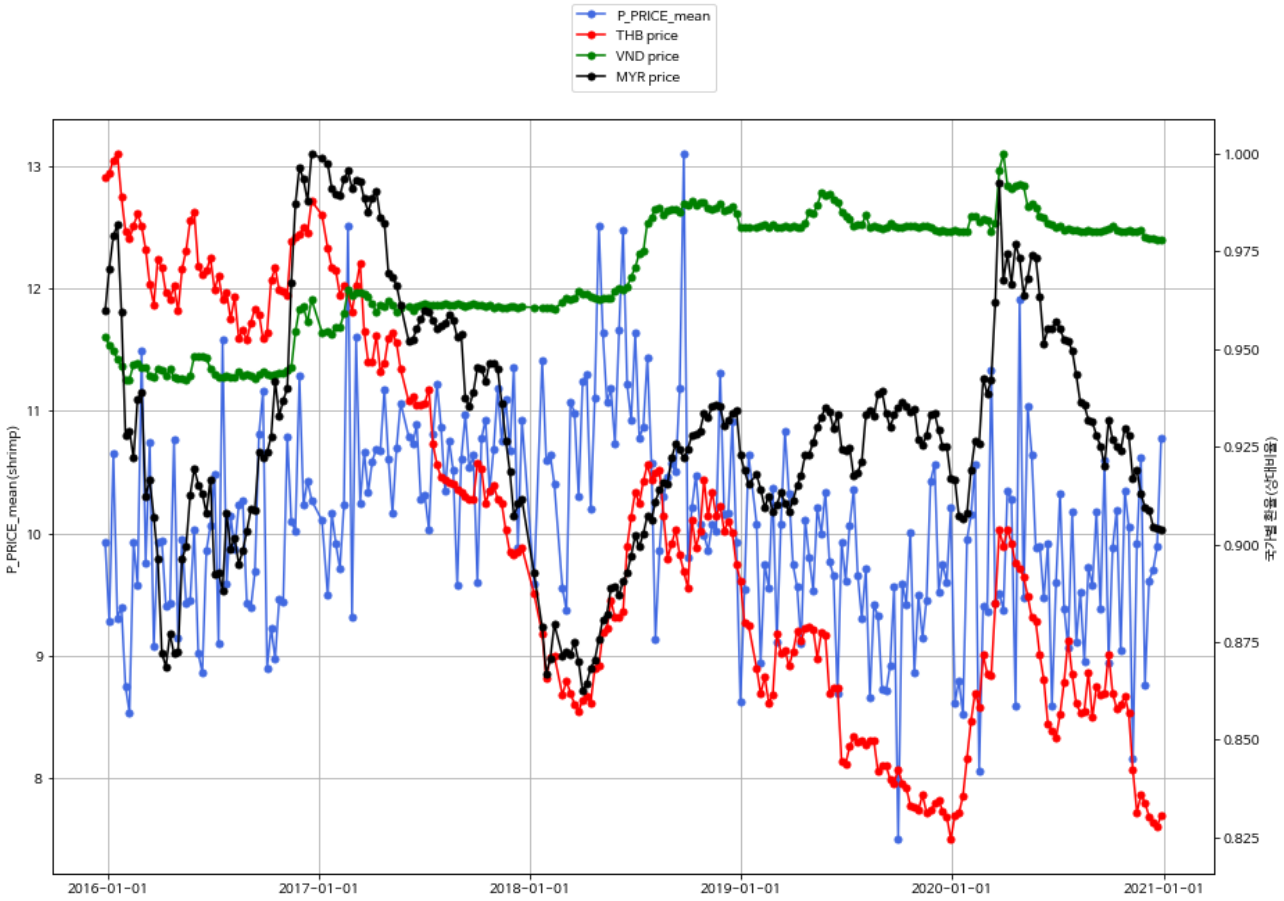
```
df_train = df_train.dropna()
```

```
# 예측값 : 오징어(상세어종 : 오징어), 연어(상세어종 : 연어), 새우(상세어종 : 흰다리새우)
df_squid = df_train[df_train['P_NAME'] == '오징어']
df_salmon = df_train[df_train['P_NAME'] == '연어']
df_shrimp = df_train[df_train['P_NAME'] == '흰다리새우']
```

```
# 환율에 대한 상관관계 분석
## 대상 : 새우
cols_for_corr = ['P_PRICE','USD_NOK', 'USD_CNH', 'USD_THB', 'USD_GBP', 'USD_MYR','USD_PEN', 'USD
df_shrimp_grouped = df_shrimp.groupby(by = "REG_DATE").mean()[cols_for_corr]
corr_shrimp = df_shrimp_grouped.corr()
plt.figure(figsize = (12,12))
sns.heatmap(corr_shrimp, annot = True, cmap = 'RdYlBu_r', vmin = -1, vmax = 1)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning
```

```
# 환율과 실제 가격 추이의 관계를 1) 데이터 시각화와 2) 교차 상관계수 분석으로 게산해보자.
# 항목 : 새우
# 1) 데이터 시각화
# price - > 변수 관계 없이 mean 값을 이용
import matplotlib.dates as md
xfmt = md.DateFormatter('%Y-%m-%d')

fig, ax1 = plt.subplots(figsize=(15,10))
df_shrimp_price_mean = df_shrimp.groupby(by = ['REG_DATE']).mean()

ax1.plot(df_shrimp_price_mean['P_PRICE'], marker='o', ms=5, color='royalblue', label='P_PRICE_me
ax1.xaxis.set_major_formatter(xfmt)
ax1.set_ylabel('P_PRICE_mean(shrimp)')
ax1.grid()

cols_for_corr = ['P_PRICE','USD_NOK', 'USD_CNH', 'USD_THB', 'USD_GBP', 'USD_MYR','USD_PEN', 'USD
# 환율 : VND 베트남, NOK : 노르웨이, CNH : 중국, TH : 태국, GBP : 영국, MYR : 말레이시아 PEN :


ax2 = ax1.twinx()
ax2.plot(df_shrimp_price_mean['USD_THB'] / np.max(df_shrimp_price_mean['USD_THB']), marker='o',
ax2.plot(df_shrimp_price_mean['USD_VND'] / np.max(df_shrimp_price_mean['USD_VND']), marker='o',
ax2.plot(df_shrimp_price_mean['USD_MYR'] / np.max(df_shrimp_price_mean['USD_MYR']), marker='o',

ax2.xaxis.set_major_formatter(xfmt)
ax2.set_xlabel('Date')
ax2.set_ylabel('국가별 환율(상대비율)')

fig.legend(loc = 'upper center')
plt.show()

# 2) 교차 상관계수 분석
shrimp_price_mean = df_shrimp_price_mean['P_PRICE'].values
USD_THB = df_shrimp_price_mean['USD_THB'].values
USD_VND = df_shrimp_price_mean['USD_VND'].values
USD_MYR = df_shrimp_price_mean['USD_MYR'].values

corr = np.corrcoef(np.array([shrimp_price_mean, USD_THB, USD_VND, USD_MYR]))
print("₩n₩n")
print("THB 교차 상관계수 : ", corr[0][1])
print("VND 교차 상관계수 : ", corr[0][2])
print("MYR 교차 상관계수 : ", corr[0][3])
print("₩n₩n")
corr_dict = {"THB_corr" : corr[0][1], "VND_corr" : corr[0][2], "MYR_corr":corr[0][3]}
corr_pd = pd.DataFrame(data = corr_dict, index = ["P_PRICE"])
display(corr_pd)
```

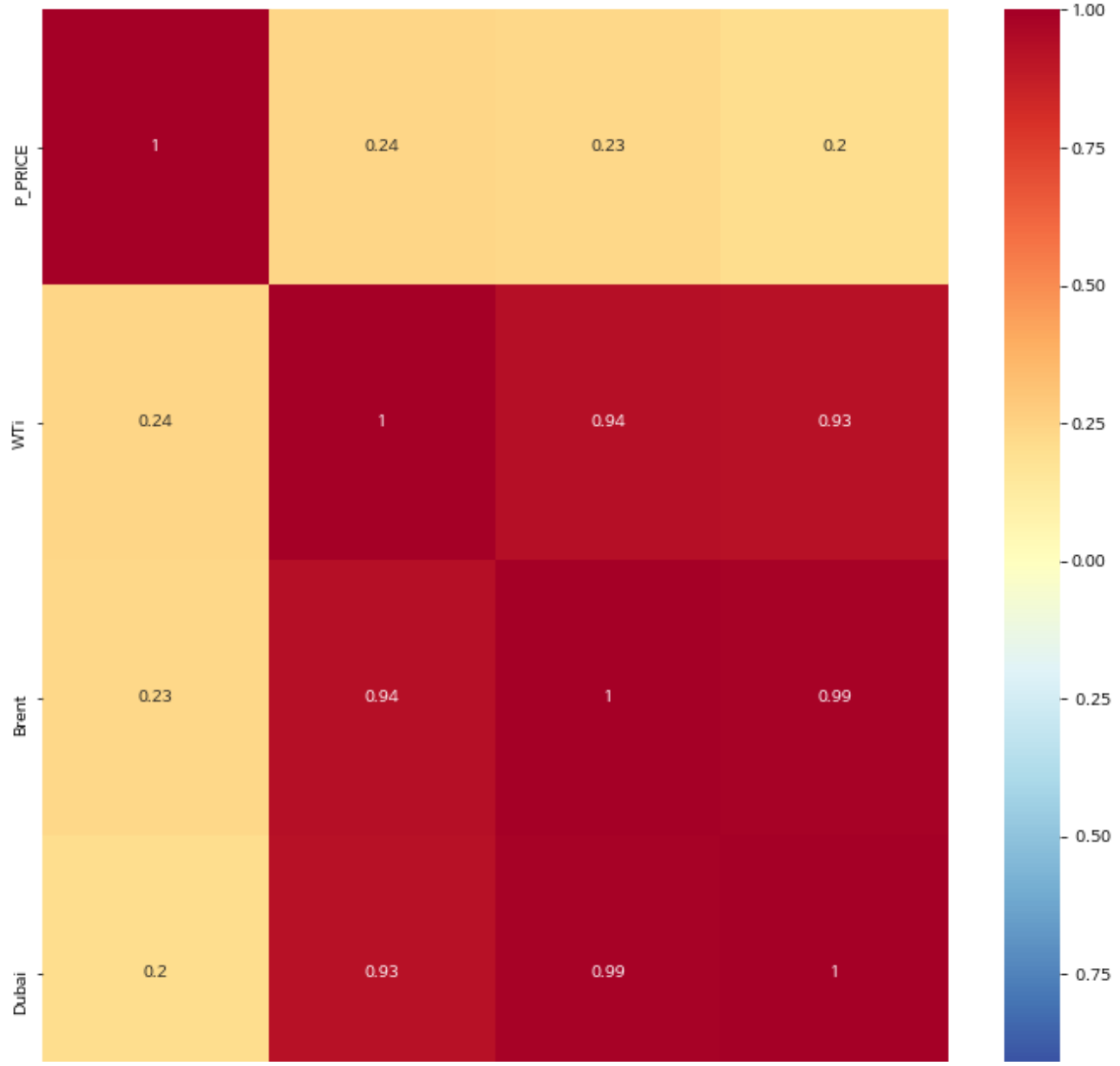



```
THB 교차 상관계수 :  0.1501764421197182
VND 교차 상관계수 :  -0.11297558966917051
MYR 교차 상관계수 :  -0.03138147369769412
```

	THB_corr	VND_corr	MYR_corr
P_PRICE	0.150176	-0.112976	-0.031381

```
1 # 유가에 대한 상관관계 분석
2 cols_for_corr = ['P_PRICE', 'WTi', 'Brent', 'Dubai']
3 df_shrimp_grouped = df_shrimp.groupby(by = "REG_DATE").mean()[cols_for_corr]
4 corr_shrimp = df_shrimp_grouped.corr()
5 plt.figure(figsize = (12,12))
6 sns.heatmap(corr_shrimp, annot = True, cmap = 'RdYlBu_r', vmin = -1, vmax = 1)
```

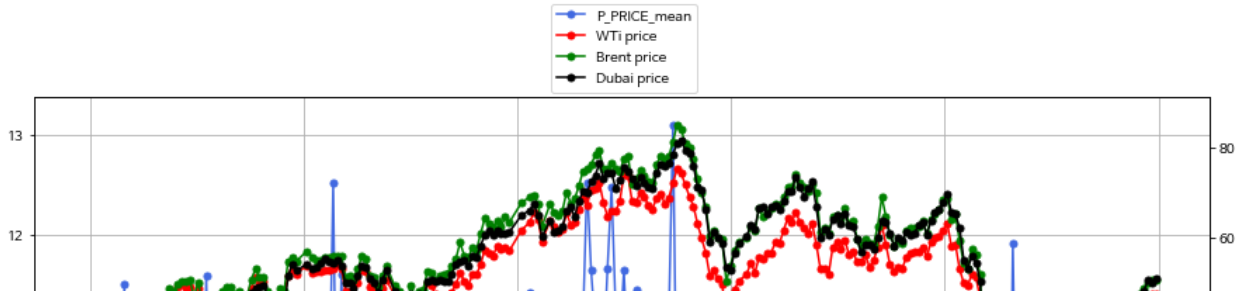
```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning
  font.set_text(s, 0, flags=flags)
<matplotlib.axes._subplots.AxesSubplot at 0x7f83c1355850>
```




```
# 국제유가와 실제 가격 추이의 관계를 1) 데이터 시각화와 2) 교차 상관계수 분석으로 계산해보자.
# 항목 : 새우
# 1) 데이터 시각화
# price - > 변수 관계 없이 mean 값을 이용
import matplotlib.dates as md
xfmt = md.DateFormatter('%Y-%m-%d')

fig, ax1 = plt.subplots(figsize=(15,8))
df_shrimp_price_mean = df_shrimp.groupby(by = ['REG_DATE']).mean()

ax1.plot(df_shrimp_price_mean['P_PRICE'], marker='o', ms=5, color='royalblue', label='P_PRICE_me
ax1.xaxis.set_major_formatter(xfmt)
ax1.set_ylabel('P_PRICE_mean(shrimp)')
ax1.grid()

ax2 = ax1.twinx()
ax2.plot(df_shrimp_price_mean['WTi'], marker='o', ms=5, color='r', label='WTi price')
```

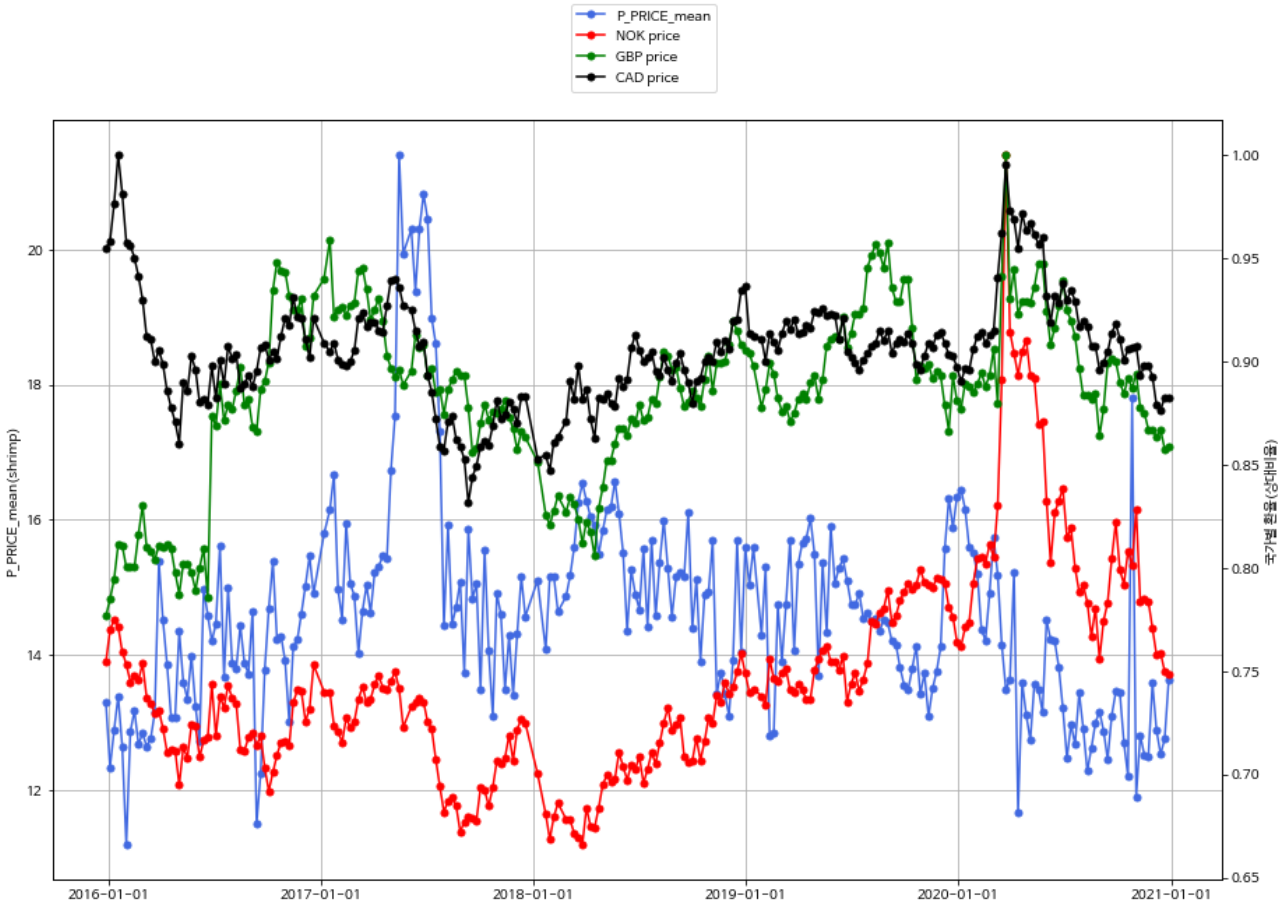
```
ax2.plot(df_shrimp_price_mean['Brent'], marker='o', ms=5, color='g', label='Brent price')
ax2.plot(df_shrimp_price_mean['Dubai'], marker='o', ms=5, color='k', label='Dubai price')
ax2.xaxis.set_major_formatter(xfmt)
ax2.set_xlabel('Date')
ax2.set_ylabel('International fuel price')

fig.legend(loc = 'upper center')
plt.show()

# 2) 교차 상관계수 분석
shrimp_price_mean = df_shrimp_price_mean['P_PRICE'].values
WTi_price = df_shrimp_price_mean['WTi'].values
Brent_price = df_shrimp_price_mean['Brent'].values
Dubai_price = df_shrimp_price_mean['Dubai'].values

corr = np.corrcoef(np.array([shrimp_price_mean, WTi_price, Brent_price, Dubai_price]))
print("₩n₩n")
print("WTi 교차 상관계수 : ", corr[0][1])
print("Brent 교차 상관계수 : ", corr[0][2])
print("Dubai 교차 상관계수 : ", corr[0][3])

corr_dict = {"WTi_corr" : corr[0][1], "Brent_corr" : corr[0][2], "Dubai_corr":corr[0][3]}
corr_pd = pd.DataFrame(data = corr_dict, index = ["P_PRICE"])
display(corr_pd)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarnin
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarnin
  font.set_text(s, 0, flags=flags)
```




```
# 연어에 대한 상관분석
# 환율
import matplotlib.dates as md
xfmt = md.DateFormatter('%Y-%m-%d')

fig, ax1 = plt.subplots(figsize=(15,10))
df_salmon_price_mean = df_salmon.groupby(by = ['REG_DATE']).mean()

ax1.plot(df_salmon_price_mean['P_PRICE'], marker='o', ms=5, color='royalblue', label='P_PRICE_me
ax1.xaxis.set_major_formatter(xfmt)
ax1.set_ylabel('P_PRICE_mean(shrimp)')
ax1.grid()

cols_for_corr = ['P_PRICE','USD_NOK', 'USD_CNH', 'USD_THB', 'USD_GBP', 'USD_MYR','USD_PEN', 'USD
# 환율 : VND 베트남, NOK : 노르웨이, CNH : 중국, TH : 태국, GBP : 영국, MYR : 말레이시아 PEN :


ax2 = ax1.twinx()
ax2.plot(df_salmon_price_mean['USD_NOK'] / np.max(df_salmon_price_mean['USD_NOK']), marker='o',
ax2.plot(df_salmon_price_mean['USD_GBP'] / np.max(df_salmon_price_mean['USD_GBP']), marker='o',
ax2.plot(df_salmon_price_mean['USD_CAD'] / np.max(df_salmon_price_mean['USD_CAD']), marker='o',

ax2.xaxis.set_major_formatter(xfmt)
ax2.set_xlabel('Date')
ax2.set_ylabel('국가별 환율(상대비율)')

fig.legend(loc = 'upper center')
plt.show()

# 2) 교차 상관계수 분석
shrimp_price_mean = df_salmon_price_mean['P_PRICE'].values
USD_THB = df_salmon_price_mean['USD_NOK'].values
USD_VND = df_salmon_price_mean['USD_GBP'].values
USD_MYR = df_salmon_price_mean['USD_CAD'].values

corr = np.corrcoef(np.array([shrimp_price_mean, USD_THB, USD_VND, USD_MYR]))
print("WnWn")
print("NOK 교차 상관계수 : ", corr[0][1])
print("GBP 교차 상관계수 : ", corr[0][2])
print("CAD 교차 상관계수 : ", corr[0][3])
print("WnWn")
corr_dict = {"NOK_corr" : corr[0][1], "GBP_corr" : corr[0][2], "CAD_corr":corr[0][3]}
corr_pd = pd.DataFrame(data = corr_dict, index = ["P_PRICE"])
```

```
44 display(corr_pd)
```



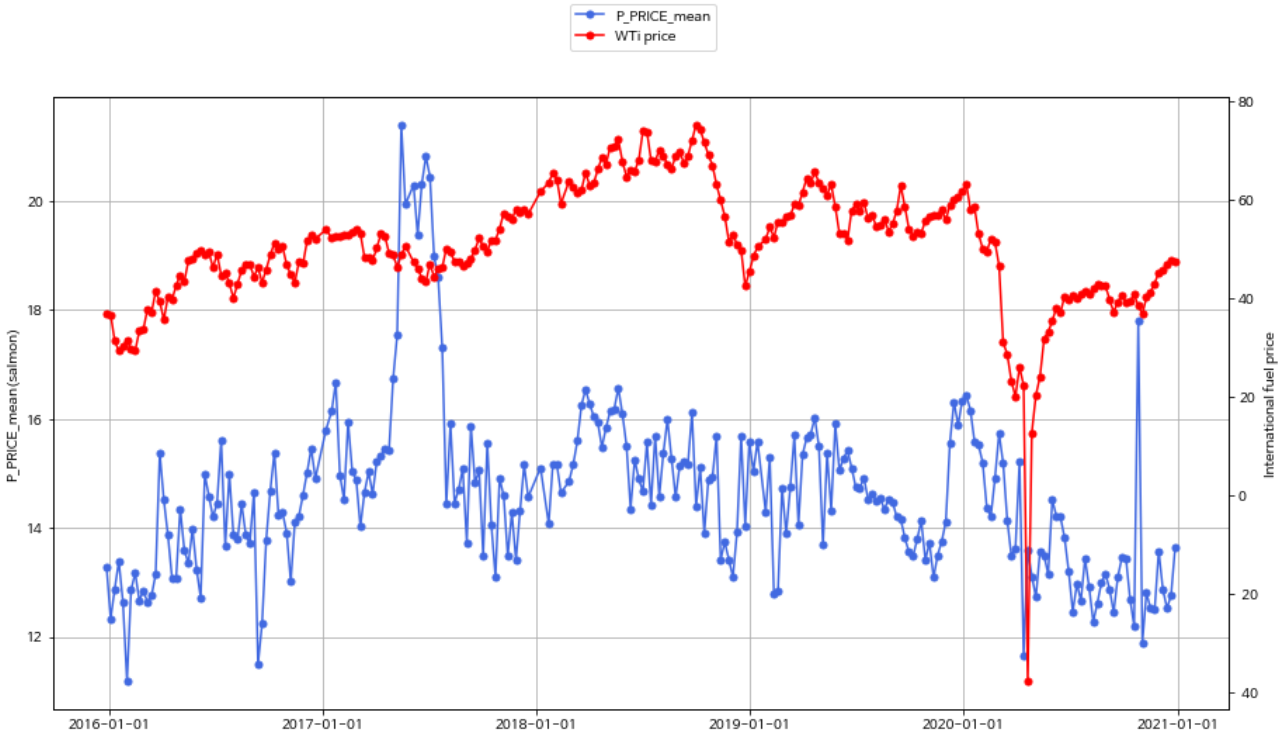

```
NOK 교차 상관계수 :  -0.2995932740617358
GBP 교차 상관계수 :  0.11010017685634266
CAD 교차 상관계수 :  -0.1721135843826741
```

	NOK_corr	GBP_corr	CAD_corr
P_PRICE	-0.299593	0.1101	-0.172114

```
1 # 국제유가와 실제 가격 추이의 관계를 1) 데이터 시각화와 2) 교차 상관계수 분석으로 게산해보자.
2 # 항목 : 연어
3 # 1) 데이터 시각화
4 # price - > 변수 관계 없이 mean 값을 이용
5 import matplotlib.dates as md
```

```
import matplotlib.dates as md
xfmt = md.DateFormatter('%Y-%m-%d')

fig, ax1 = plt.subplots(figsize=(15,8))
df_salmon_price_mean = df_salmon.groupby(by = ['REG_DATE']).mean()

ax1.plot(df_salmon_price_mean['P_PRICE'], marker='o', ms=5, color='royalblue', label='P_PRICE_me
ax1.xaxis.set_major_formatter(xfmt)
ax1.set_ylabel('P_PRICE_mean(salmon)')
ax1.grid()

ax2 = ax1.twinx()
ax2.plot(df_salmon_price_mean['WTi'], marker='o', ms=5, color='r', label='WTi price')
#ax2.plot(df_salmon_price_mean['Brent'], marker='o', ms=5, color='g', label='WTi price')
#ax2.plot(df_salmon_price_mean['Dubai'], marker='o', ms=5, color='k', label='WTi price')
ax2.xaxis.set_major_formatter(xfmt)
ax2.set_xlabel('Date')
ax2.set_ylabel('International fuel price')

fig.legend(loc = 'upper center')
plt.show()

# 2) 교차 상관계수 분석
salmon_price_mean = df_salmon_price_mean['P_PRICE'].values
WTi_price = df_salmon_price_mean['WTi'].values
Brent_price = df_salmon_price_mean['Brent'].values
Dubai_price = df_salmon_price_mean['Dubai'].values

corr = np.corrcoef(np.array([salmon_price_mean, WTi_price, Brent_price, Dubai_price]))
print("₩n₩n")
print("WTi 교차 상관계수 : ", corr[0][1])
print("Brent 교차 상관계수 : ", corr[0][2])
print("Dubai 교차 상관계수 : ", corr[0][3])

print("₩n₩n")
corr_dict = {"WTI_corr" : corr[0][1], "Brent_corr" : corr[0][2], "Dubai_corr":corr[0][3]}
corr_pd = pd.DataFrame(data = corr_dict, index = ["P_PRICE"])
display(corr_pd)
```

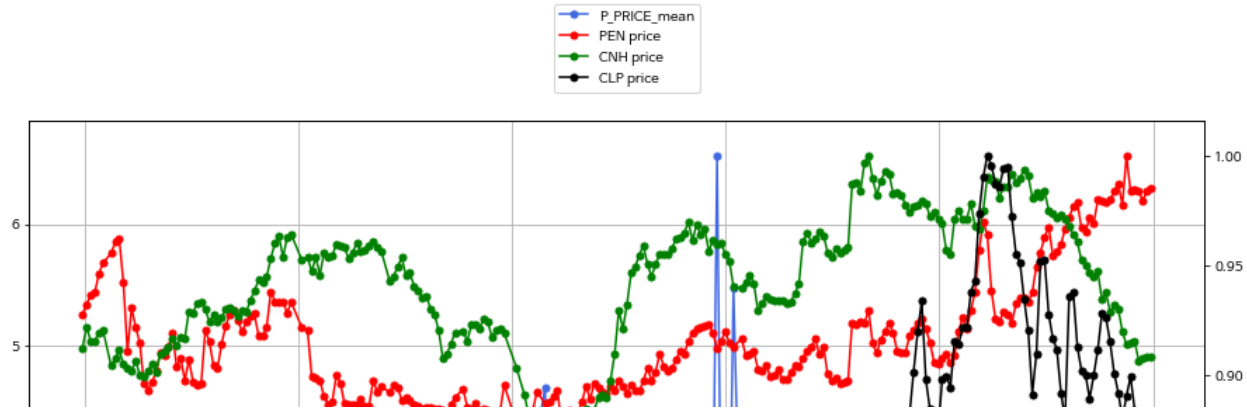
```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning
  font.set_text(s, 0, flags=flags)
```




```
WTi 교차 상관계수 :  0.3348211451474149
Brent 교차 상관계수 :  0.33522477794670474
```

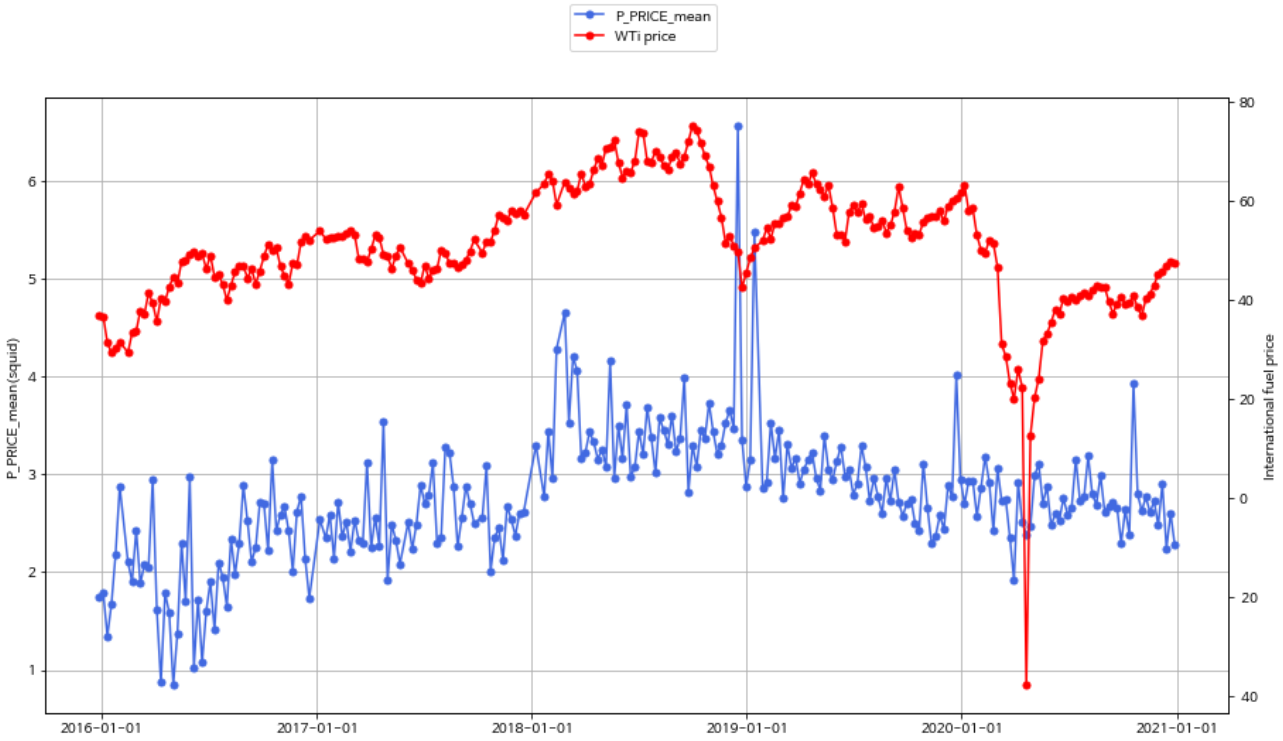
```
# 오징어에 대한 상관분석
# 환율
import matplotlib.dates as md
xfmt = md.DateFormatter('%Y-%m-%d')

fig, ax1 = plt.subplots(figsize=(15,10))
df_squid_price_mean = df_squid.groupby(by = ['REG_DATE']).mean()

ax1.plot(df_squid_price_mean['P_PRICE'], marker='o', ms=5, color='royalblue', label='P_PRICE_mea
ax1.xaxis.set_major_formatter(xfmt)
ax1.set_ylabel('P_PRICE_mean(shrimp)')
ax1.grid()

cols_for_corr = ['P_PRICE','USD_NOK', 'USD_CNH', 'USD_THB', 'USD_GBP', 'USD_MYR','USD_PEN', 'USD
# 환율 : VND 베트남, NOK : 노르웨이, CNH : 중국, TH : 태국, GBP : 영국, MYR : 말레이시아 PEN :


ax2 = ax1.twinx()
ax2.plot(df_squid_price_mean['USD_PEN'] / np.max(df_squid_price_mean['USD_PEN']), marker='o', ms
ax2.plot(df_squid_price_mean['USD_CNH'] / np.max(df_squid_price_mean['USD_CNH']), marker='o', ms
ax2.plot(df_squid_price_mean['USD_CLP'] / np.max(df_squid_price_mean['USD_CLP']), marker='o', ms

ax2.xaxis.set_major_formatter(xfmt)
ax2.set_xlabel('Date')
ax2.set_ylabel('국가별 환율(상대비율)')

fig.legend(loc='upper center')
```

```
fig.legend(loc = 'upper center')
plt.show()

# 2) 교차 상관계수 분석
shrimp_price_mean = df_squid_price_mean['P_PRICE'].values
USD_PEN = df_squid_price_mean['USD_PEN'].values
USD_CNH = df_squid_price_mean['USD_CNH'].values
USD_CLP = df_squid_price_mean['USD_CLP'].values

corr = np.corrcoef(np.array([shrimp_price_mean, USD_PEN, USD_CNH, USD_CLP]))
print("WnWn")
print("PEN 교차 상관계수 : ", corr[0][1])
print("CNH 교차 상관계수 : ", corr[0][2])
print("CLP 교차 상관계수 : ", corr[0][3])
print("WnWn")
corr_dict = {"PEN_corr" : corr[0][1], "CNH_corr" : corr[0][2], "CLP_corr":corr[0][3]}
corr_pd = pd.DataFrame(data = corr_dict, index = ["P_PRICE"])
display(corr_pd)
```




```
# 국제유가와 실제 가격 추이의 관계를 1) 데이터 시각화와 2) 교차 상관계수 분석으로 계산해보자.
# 항목 : 오징어
# 1) 데이터 시각화
# price - > 변수 관계 없이 mean 값을 이용
import matplotlib.dates as md
xfmt = md.DateFormatter('%Y-%m-%d')

fig, ax1 = plt.subplots(figsize=(15,8))
df_squid_price_mean = df_squid.groupby(by = ['REG_DATE']).mean()

ax1.plot(df_squid_price_mean['P_PRICE'], marker='o', ms=5, color='royalblue', label='P_PRICE_mea
ax1.xaxis.set_major_formatter(xfmt)
ax1.set_ylabel('P_PRICE_mean(squid)')
ax1.grid()

ax2 = ax1.twinx()
ax2.plot(df_squid_price_mean['WTi'], marker='o', ms=5, color='r', label='WTi price')
#ax2.plot(df_squid_price_mean['Brent'], marker='o', ms=5, color='g', label='WTi price')
#ax2.plot(df_squid_price_mean['Dubai'], marker='o', ms=5, color='k', label='WTi price')
ax2.xaxis.set_major_formatter(xfmt)
ax2.set_xlabel('Date')
ax2.set_ylabel('International fuel price')

fig.legend(loc = 'upper center')
plt.show()

# 2) 교차 상관계수 분석
squid_price_mean = df_squid_price_mean['P_PRICE'].values
WTi_price = df_squid_price_mean['WTi'].values
Brent_price = df_squid_price_mean['Brent'].values
Dubai_price = df_squid_price_mean['Dubai'].values

corr = np.corrcoef(np.array([squid_price_mean, WTi_price, Brent_price, Dubai_price]))
print("₩n₩n")
print("WTi 교차 상관계수 : ", corr[0][1])
print("Brent 교차 상관계수 : ", corr[0][2])
print("Dubai 교차 상관계수 : ", corr[0][3])
print("₩n₩n")

corr_dict = {"WTI_corr" : corr[0][1], "Brent_corr" : corr[0][2], "Dubai_corr":corr[0][3]}
corr_pd = pd.DataFrame(data = corr_dict, index = ["P_PRICE"])
display(corr_pd)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning
  font.set_text(s, 0, flags=flags)
```




```
WTi 교차 상관계수 :  0.41526230790712737
Brent 교차 상관계수 :  0.5041140542440752
Dubai 교차 상관계수 :  0.5195339447399241
```

	WTI_corr	Brent_corr	Dubai_corr
P_PRICE	0.415262	0.504114	0.519534

1

✓ 4초 오전 2:08에 완료됨