1.군집분석

- 데이터 경로 /kaggle/input/adp-p7/problem1.csv

데이터 설명

- InvoiceNo: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.
- StockCode: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
- Description: Product (item) name. Nominal.
- Quantity: The quantities of each product (item) per transaction. Numeric.
- InvoiceDate: Invice Date and time. Numeric, the day and time when each transaction was generated.
- UnitPrice: Unit price. Numeric, Product price per unit in sterling.
- CustomerID: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
- Country: Country name. Nominal, the name of the country where each customer resides.

데이터 출처

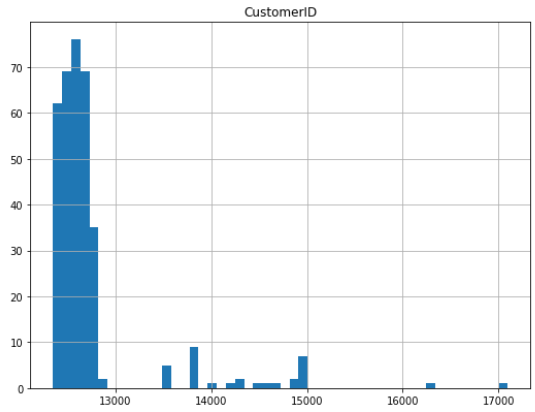
- https://archive.ics.uci.edu/ml/index.php

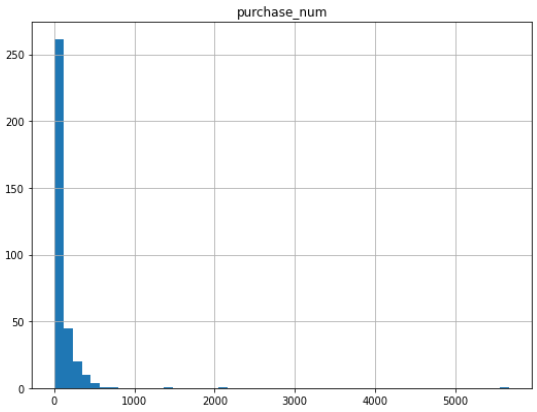
1.1 F(소비자별 구매빈도), M(소비자별 총 구매액) feature를 새로 생성해서 그 결과값으로 탐색적 분석 실시

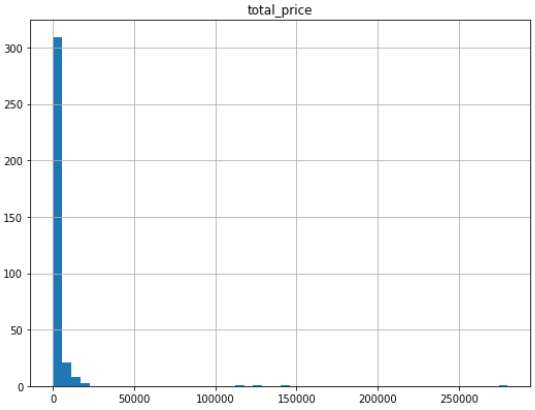
In [6]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns


data = pd.read_csv('problem1.csv')

data = data.loc[data.Quantity > 0] # Quantity값이 0보다 작을 경우 반품이라 가정하여

data['total_price'] = data.Quantity * data.UnitPrice # 총 구매액 생성

data_cust = pd.DataFrame(data.CustomerID.unique(), columns=['CustomerID']) # Custome

data_num = pd.DataFrame(data.CustomerID.value_counts().reset_index()) # CustomerID별
data_num.columns = ['CustomerID', 'purchase_num']

data_group = data.groupby('CustomerID').sum().reset_index() # 고객번호별 총 구매액 (
data_group.drop(['Quantity', 'UnitPrice'], axis = 1, inplace = True) # 불필요 컬럼
```

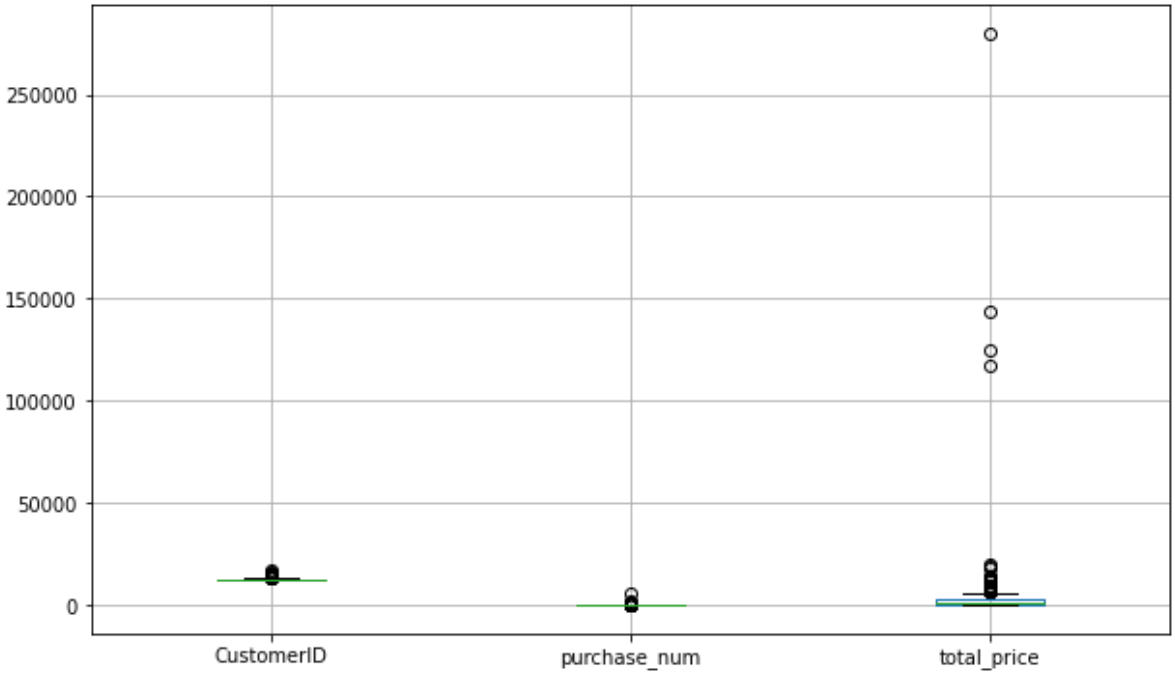
```
data_final = pd.merge(data_cust, data_num, how = 'left', on = 'CustomerID') # Custom
data_final = pd.merge(data_final, data_group, how = 'left', on = 'CustomerID') # 선

# EDA 시행

print(data_final.info())
print(data_final.describe())
print(data_final.isnull().sum())

# 마법의 matplotlib 명령
%matplotlib inline

# 수치형 변수 히스토그램 그려보기
import matplotlib.pyplot as plt
data_final.hist(bins = 50, figsize = (20,15))
plt.show()

data_final.boxplot(figsize = (10,6)) # numeric 변수만 그려짐
plt.show()
data_final.boxplot(column = 'purchase_num') # 컬럼별로 하나씩
plt.show()
data_final.boxplot(column = 'total_price') # 컬럼별로 하나씩
plt.show()
```

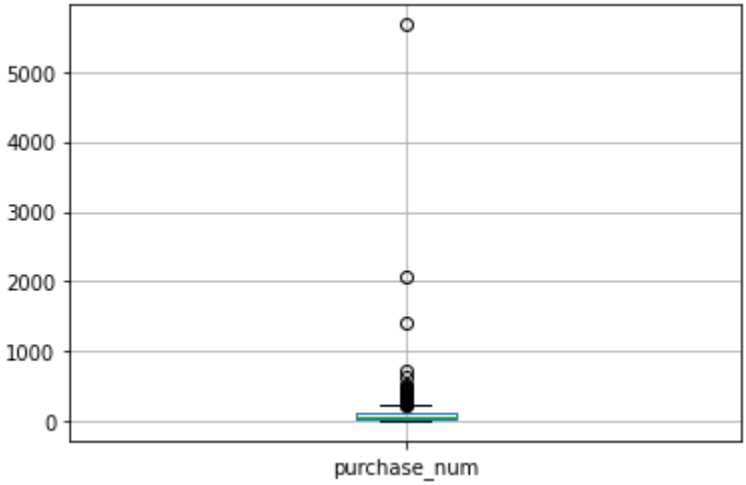
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 345 entries, 0 to 344
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   CustomerID    345 non-null    float64
 1   purchase_num  345 non-null    int64  
 2   total_price   345 non-null    float64
dtypes: float64(2), int64(1)
memory usage: 10.8 KB
None
         CustomerID  purchase_num    total_price
count    345.000000    345.000000     345.000000
mean   12738.921739    115.556522    4145.790174
std      591.750760    343.590427   19255.241859
min    12348.000000      2.000000      63.000000
25%    12476.000000     23.000000     460.890000
50%    12591.000000     50.000000    1035.800000
75%    12707.000000    108.000000    2794.510000
max    17097.000000   5677.000000  280206.020000
CustomerID      0
purchase_num    0
total_price     0
dtype: int64
```

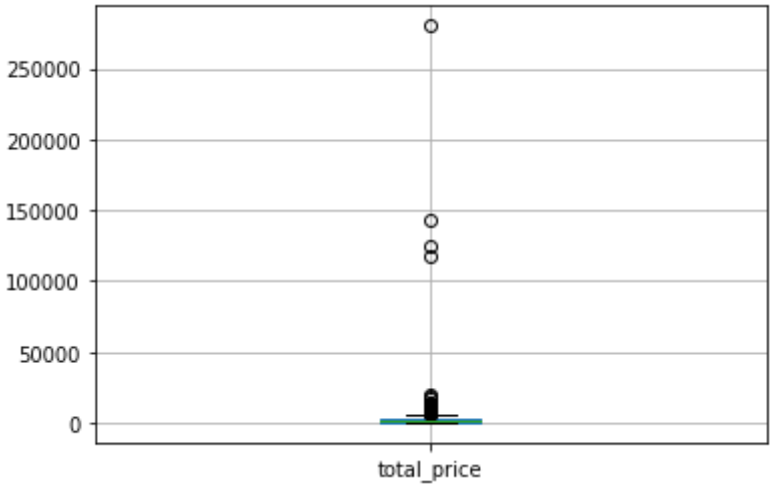

CustomerID
0
10
20
30
40
50
60
70
13000
14000
15000
16000
17000


purchase_num
0
50
100
150
200
250
0
1000
2000
3000
4000
5000


total_price
0
50
100
150
200
250
300
0
50000
100000
150000
200000
250000


250000
200000
150000
100000
50000
0
CustomerID
purchase_num
total_price





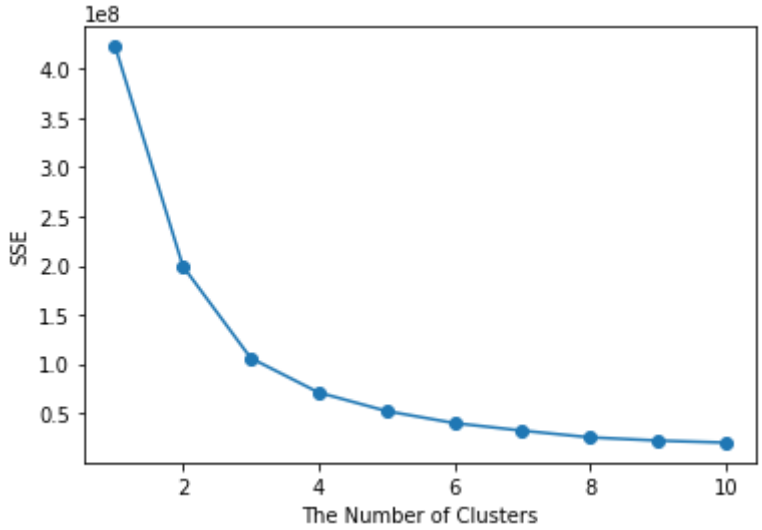
답안

- 문제에서 요구한대로 고객번호별 구매 빈도와 총 소비 금액에 대한 파생변수를 추출하였다.
- 고객번호별 고유값에 해당 파생변수를 merge를 통해 붙여 final 데이터를 생성하였음
- F와 M 컬럼은 수치형 데이터이며, 구매횟수와 총 소비 금액에 대해 histogram을 그려보았다.
- 구매 빈도의 경우 max값 5677로 확인되며 특정 고객이 구매를 매우 많이 한것으로 추정된다.
- 대부분의 구매빈도의 경우 1~2회에 몰려있으며 오른쪽으로 긴 고리를 갖는 그래프가 확인된다.
- 총 구매액 역시 구매 빈도와 비슷한 형태의 그래프를 띄고 있으며 일부 25만이 넘는 구매액을 가진 고객도 존재한다.

1.2 F, M feature 기반으로 군집분석 실시, 필요시 이상값 보정

In [18]:
```
# 이상치 보정 시행
# def detect_oulier(df = None, column = None, weight = 1.5) :
#     Q1 = data_final[column].quantile(0.25)
#     Q3 = data_final[column].quantile(0.75)

#     IQR = Q3 - Q1
#     IQR_weight = IQR * weight
```

```
#     out_idx = data_final[(data_final[column] < Q1 - IQR_weight) | (data_final[colu

#     return out_idx


# # 컬럼별 이상치들에 대해 각 컬럼의 중위수로 보정 시행
# print(detect_oulier(data_final, 'purchase_num'))
# data_final.loc[detect_oulier(data_final, 'purchase_num'), 'purchase_num'] = data_f
# print(detect_oulier(data_final, 'total_price'))
# data_final.loc[detect_oulier(data_final, 'total_price'), 'total_price'] = data_fin


# K-means를 활용한 군집분석 시행
from sklearn.cluster import KMeans
km = KMeans(n_clusters = 3, random_state=1)
km.fit(data_final)

# elbow 그림 그리는 함수
import matplotlib.pyplot as plt
def elbow(data_fianl):
    sse=[]
    for i in range(1, 11):
        km=KMeans(n_clusters=i, random_state=1)
        km.fit(data_fianl)
        sse.append(km.inertia_)

    plt.plot(range(1,11), sse, marker='o')
    plt.xlabel('The Number of Clusters')
    plt.ylabel('SSE')
    plt.show()
    print(sse)

elbow(data_final) # 3에서 4로 가는 구간에서 기울기 소실이 급격하게 이루어 짐.

# 클러스터별 콜린스키 하라바츠 결과 비교 : 군집이 3개일때 콜린스키 하라바츠 score가
# 따라서 군집은 2개로 묶도록 한다.
from sklearn.metrics import calinski_harabasz_score
for k in range(2,4):
    kmeans_model = KMeans(n_clusters=k, random_state=1).fit(data_final)
    labels = kmeans_model.labels_
    print(k, ':', calinski_harabasz_score(data_final, labels))


# K-means를 활용한 군집분석 시행
from sklearn.cluster import KMeans

km = KMeans(n_clusters = 3, random_state=1)
km.fit(data_final)
new_labels = km.labels_
data_final['cluster'] = new_labels
data_final
```




```
[422299739.27822477, 199220683.69048336, 105997464.27875607, 71066684.22370614, 52231926.619048, 40025441.30728433, 32343474.71380206, 25494280.665596683, 22235780.832377748, 20127644.83297242]
2 : 384.0771683399779
3 : 510.2734239251927
```

Out[18]:

	CustomerID	purchase_num	total_price	cluster	coef
0	14911.0	50	1035.80	0	0.594948
1	12682.0	50	1035.80	1	0.737074
2	12705.0	50	1035.80	1	0.733185
3	12727.0	139	3980.70	2	0.601180
4	12645.0	73	1775.18	2	0.033760
...	...	...	...	...	...
340	12665.0	3	63.00	1	0.678708
341	12504.0	9	482.05	1	0.801835
342	12509.0	7	176.50	1	0.734572
343	12791.0	2	192.60	1	0.686268
344	12789.0	4	91.85	1	0.655778

345 rows × 5 columns

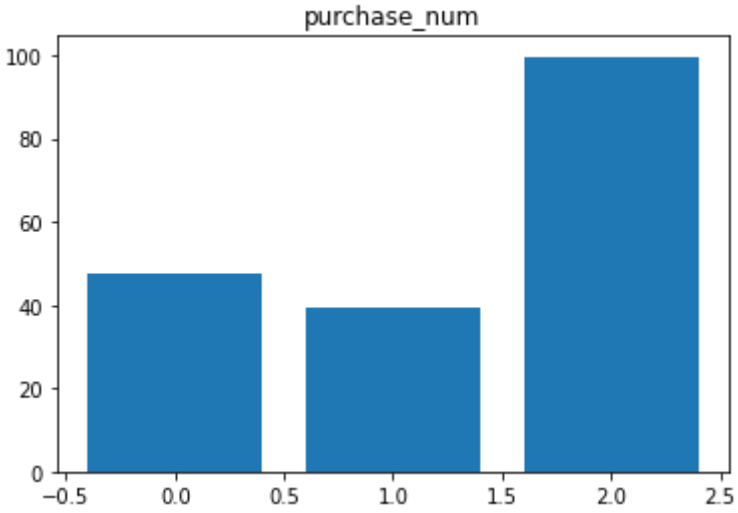
답안

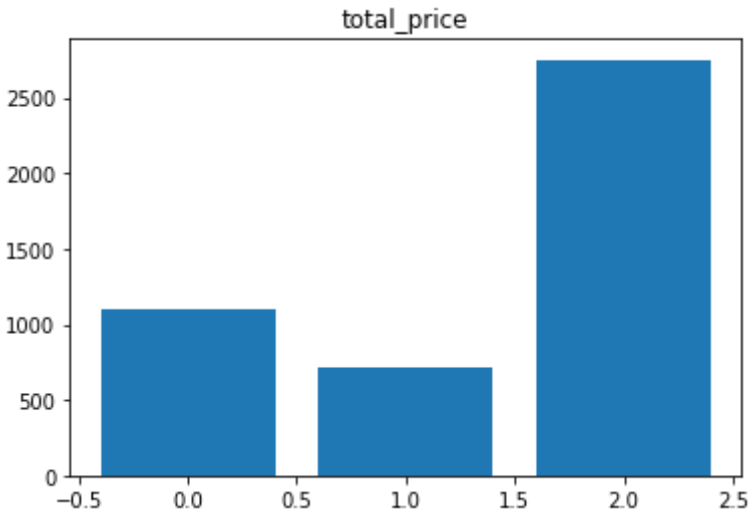
- 이상치 확인 결과 새로 생성한 파생변수에서 이상치로 인식할만한 값들이 발견되었다.
- 그러나 고객의 실제 구매 데이터 이기에 이상치 처리를 따로 하지는 않겠다.
- 만약 진행한다면 전통적인 IQR 방법을 사용하여 이상치를 각 컬럼의 중위수로 대치해 줄 수 있다.
- 이후 KMeans 군집분석을 활용하여 최초 군집을 생성하였고,
- elbow 그림을 그려 확인한 결과 3~4 구간에서 기울기의 급격한 소실이 확인된다.
- 콜린스키 하라바츠를 활용하여 군집이 3일때와 4일때의 score를 확인한 결과 3일때의 점수가 더 높기에 군집수는 3으로 설정한다.
- 이후 각 행에 군집 label을 할당 해 주었다.

1.3 군집 결과의 적합성을 군집 내 응집도, 군집 간 분리도의 개념을 사용해서 서술

In [19]:
```
from sklearn.metrics import silhouette_samples, silhouette_score
score_samples = silhouette_samples(data_final, data_final.cluster)
data_final['coef'] = score_samples
average_score = silhouette_score(data_final, data_final.cluster)
# 실루엣 계수가 1에 가까울수록 군집이 잘되었다고 판단할 수 있다.
print("실루엣 계수 : ", average_score) # 실루엣 계수는 0.56으로 확인됨.

performance = data_final.groupby('cluster').mean()
print(performance)

plt.bar(performance.index, performance.purchase_num.values)
plt.title('purchase_num')
plt.show()
plt.bar(performance.index, performance.total_price.values)
plt.title('total_price')
plt.show()
# 2번 군집이 가장 잘 군집되었다 판단되며 1번 군집이 가장 많이 흩어져있게 되었다.
# 군집별 구매 횟수와 총 지출금액은 확연한 차이를 보이며, 각 군집의 특성이 잘 나타나
```

```
실루엣 계수 :  0.6526465603053981
            CustomerID  purchase_num  total_price      coef
cluster
0         14455.655172     47.344828  1094.708621  0.368332
1         12580.230769     39.558704   719.588381  0.720662
2         12585.463768     99.681159  2751.477246  0.528664
```





답안

- 군집내 응집도와 군집간 분리도를 측정할 수 있는 계수가 바로 실루엣 계수이다.
- 실루엣 계수는 -1에서 1까지의 범위를 가지고 있으며, 1에 가까울수록 군집이 잘 이루어진것이다.
- 위에서 실시한 군집들의 실루엣 계수를 구해본 결과 0.634로 확인된다.
- 그리고 각 군집별 coef값을 확인해본 결과 1번 군집이 가장 밀도 있게 군집이 잘 되었으며0번이 가장 약하다.
- 구매빈도와 총 구매액관점에서 봤을 때 2번 군집이 구매빈도와 구매액이 가장 많은 군집이다.

1.4 적합된 군집 별 특성에 대한 의견과 비즈니스적 판단 제시

답안

- 2번 군집은 구매빈도도 많고 총 구매액도 많은 집단이다.
- 이러한 집단은 일종의 충성 집단으로 판단되기에 오랬동안 우리 회사(?)를 이용할 수 있는 프로그램을 운영하겠다.
- 예 : VIP 대상 일정 금액 이상 구매시 포인트 or 경품 선물
- 0번 군집은 2번 군집에 비해 구매 빈도와 총 구매액 모두가 적으나 2번 군집 다음으로 구매빈도와 총 구매액이 높다.
- 따라서 이러한 집단에는 더 많은 구매빈도를 유도하기 위해 그동안 구매했던 상품과 유사한 상품을 더 노출시킨다.
- 마지막 1번 군집은 구매 빈도도 낮고 총 구매액도 낮은 집단이다. 이러한 고객들은 우리 회사(?) 뿐만 아니라
- 다양한 회사를 이용하며 같은 상품에 대해 더 싼 가격을 찾아다니는 고객으로 판단할 수 있다.
- 따라서 현재 우리 회사(?)의 상품들의 가격이 타사 대비 싼 가격인지를 확인하고,
- 이벤트 등을 통해 아주 싼 가격으로 마케팅을 진행하여 우리 회사에 오래 머무를 수 있는 기회를 제공하겠다.

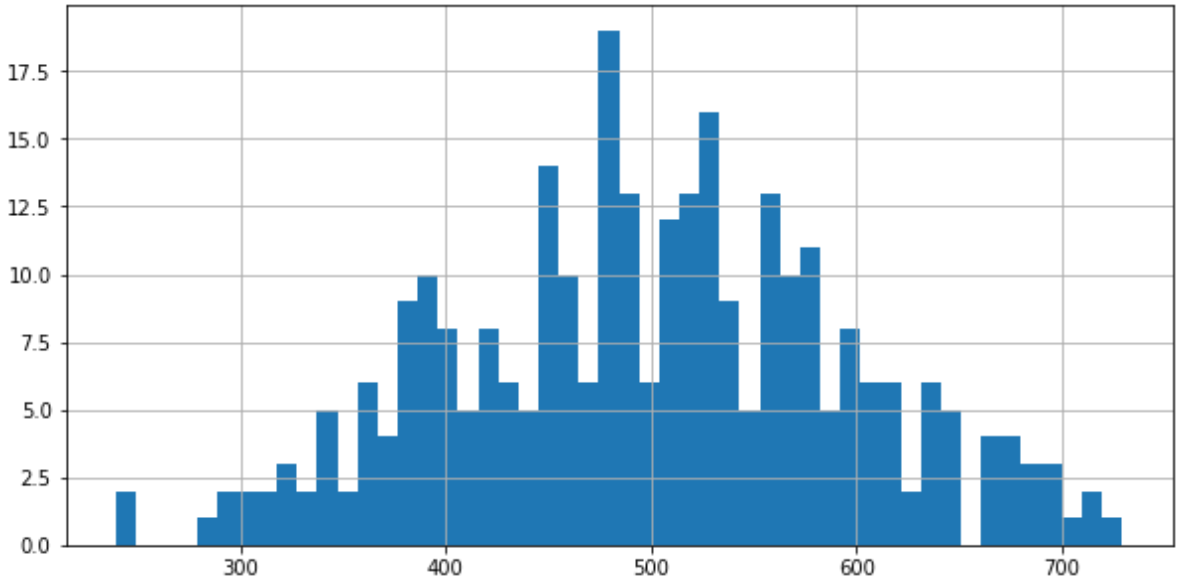
2번 시계열분석

- 경로 : /kaggle/input/adp-p7/problem2.csv
- 데이터 설명
  각 raw는 관광지 A의 1990년 1월 부터 25년동안의 매달 평균 이용객 숫자이다.

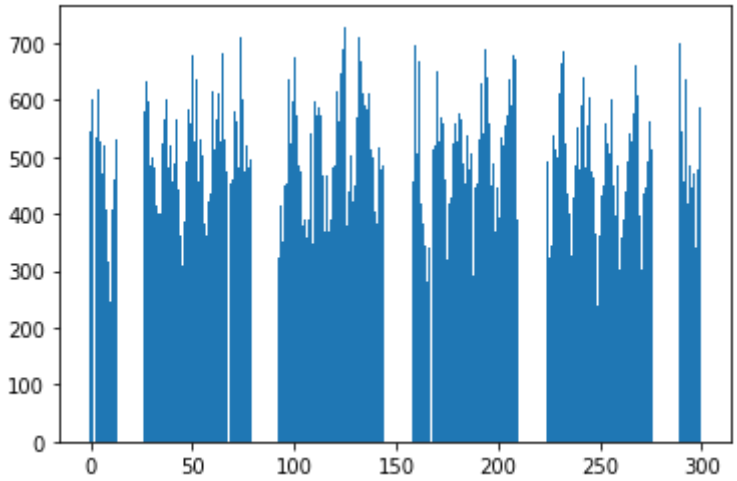
2-1 EDA와 시각화를 진행하라

In [63]:
```
data = pd.read_csv('problem2.csv')

# 마법의 matplotlib 명령
%matplotlib inline

# 수치형 변수 히스토그램 그려보기
import matplotlib.pyplot as plt
```

```
data.counts.hist(bins = 50, figsize = (10,5))
plt.show()

print(data.info())
print(data.isnull().sum())

plt.bar(data.index, data.counts.values)
plt.show()
```




```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  300 non-null    int64  
 1   counts      295 non-null    float64
dtypes: float64(1), int64(1)
memory usage: 4.8 KB
None
Unnamed: 0    0
counts        5
dtype: int64
```



답안

- 데이터는 총 300행으로 확인된다. datetime을 의미하는 컬럼은 보이지 않기에 추후 추가해야 할 것으로 판단된다.
- 주어진 데이터의 histogram을 그려본 결과 정규분포의 모양으로 확인된다.

- counts 컬럼은 총 300행 중 5개의 결측값이 있는 것으로 확인된다.
- plt.plot을 활용하여 시간 축에 따른 방문객의 숫자를 그려보았을 때 정상성이 확인되지는 않는다.
- seasonal decompose를 활용하여 정상성을 확인할 수 있으나, 아직 컬럼에 null값이 있기에 해당기능은 동작하지 않는다.
- 이후 결측치를 보간한 후 decompose 진행하겠으며, 이 데이터들은 실 측정 데이터이며 시계열 데이터 이기에..
- 이상치 보정을 하지 않고 모델링을 진행하겠다.

2-2 결측치 처리와 해당 결측치 처리 방식에 대한 논리적 근거를 제시하라

In [64]:
```
data.counts = data.counts.interpolate(method='polynomial', order = 2)
data.isnull().sum() # 결측치 대치 완료
```

Out[64]:
```
Unnamed: 0    0
counts        0
dtype: int64
```

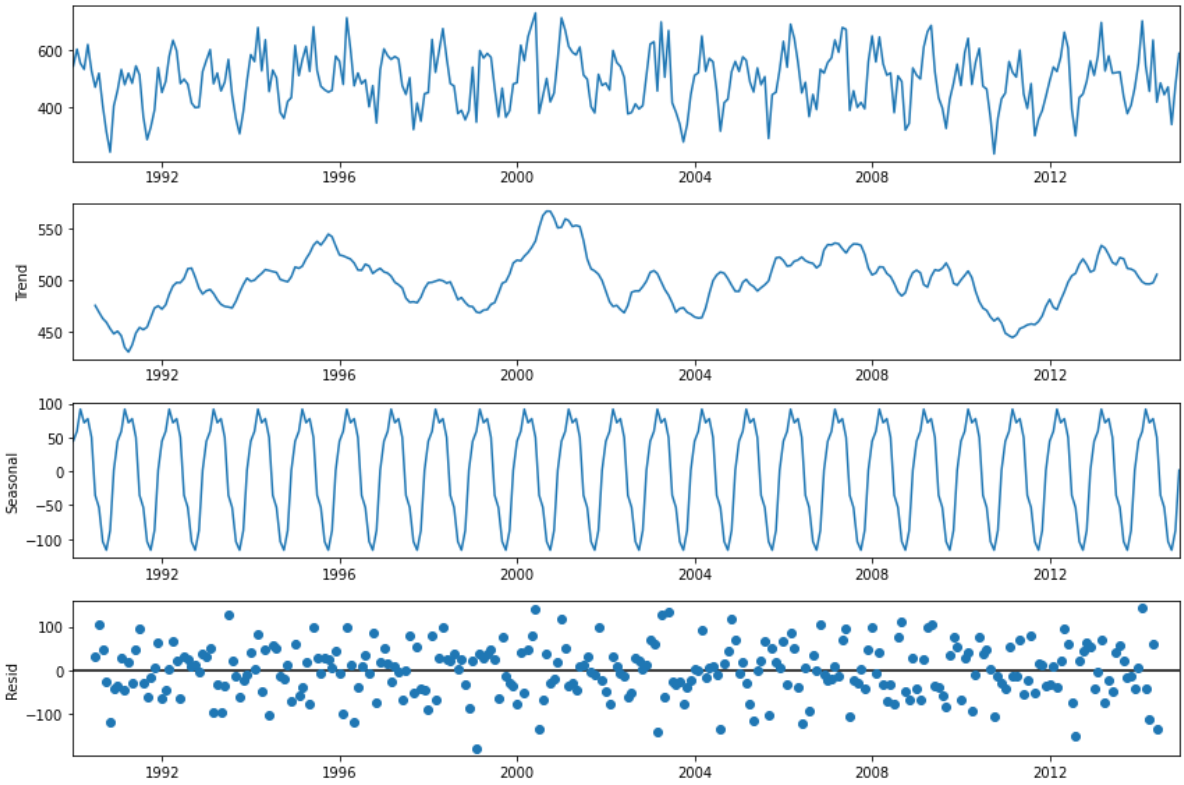
답안

- 결측치를 처리하는 방법에는 많은 방법이 있다.
- 대표적인 방법이 바로 결측치 자체를 삭제하는 방법이다. 또한 중위수, 평균으로 대치해주는 방법이 있다.
- 그리고 bfill이나 ffill 등 앞과 뒤의 숫자로 대치를 해주는 방법이 있다.
- 시계열 데이터의 경우 결측치가 있을 경우 앞이나 뒤의 숫자로 보간을 해주는 것이 오차를 줄일수 있다.
- 그러나 이번 데이터에 있는 결측치는 interpolate방법을 통해 보간하고자 한다.
- 앞과 뒤의 숫자를 고려하여 degree = 2로 설정하여 대치할 것이며, 두 숫자 사이의 관계를 고려한 대치 방법이기에
- 시간 축에 따른 데이터들의 흐름을 깨지 않은 상태에서 보간이 될 것으로 판단된다.

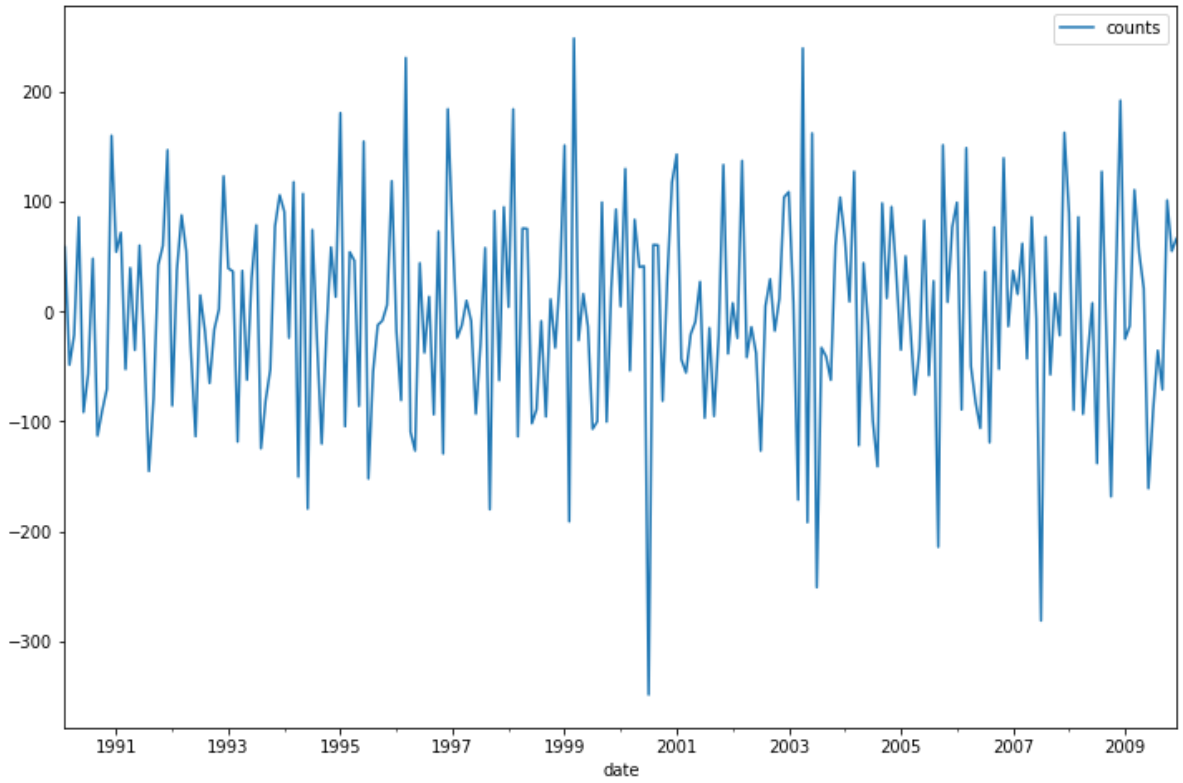
2-3 계절성을 반영한 시계열 모델을 제시하고 정확도 측면에서 모델 성능 평가 할 것

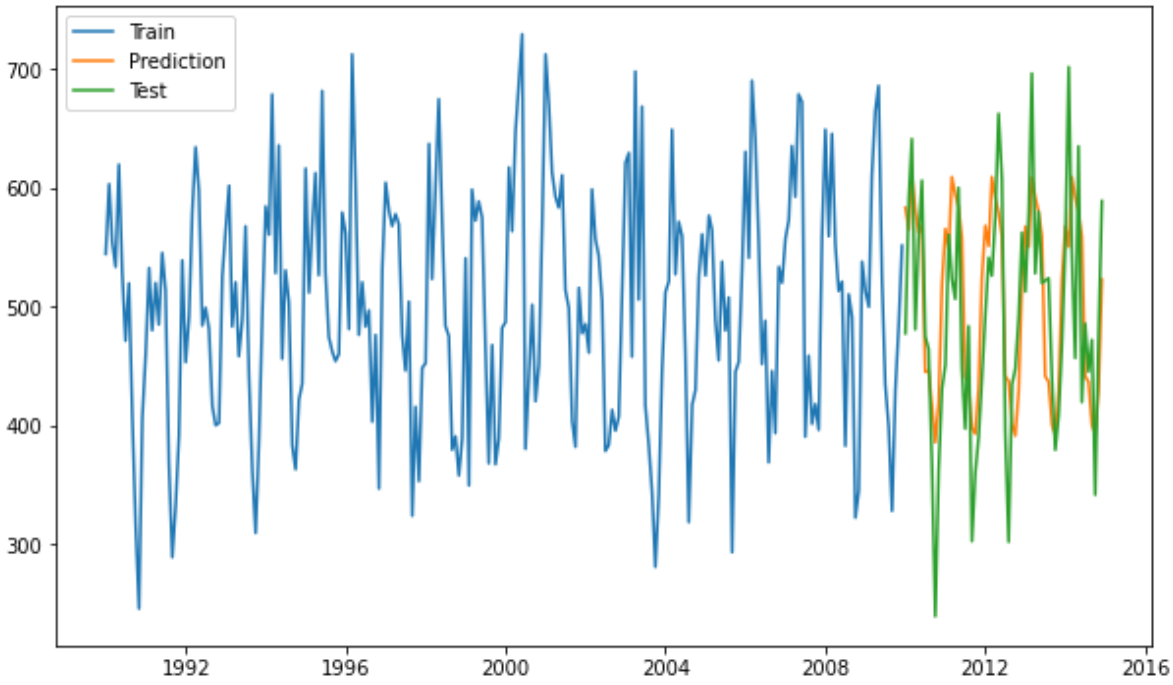
In [65]:
```
data.drop('Unnamed: 0', axis = 1, inplace = True)

data['date'] = pd.date_range('1990-01-01', '2014-12-01', freq = 'MS')
data.set_index(data.date, inplace = True)
data.drop('date', axis = 1, inplace = True)

from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.pyplot as plt

ts = data
result = seasonal_decompose(ts, model='additive')

plt.rcParams['figure.figsize'] = [12, 8]
result.plot()
plt.show()

train_len = int(len(data) * 0.8)
```

```
training = data[:train_len]
test = data.drop(training.index)

## 귀무가설 : 데이터가 정상성(stationary)을 갖지 않는다. (변동이 있다.)
## 대립가설 : 데이터가 정상성(stationary)을 갖는다

from statsmodels.tsa.stattools import adfuller

adf = adfuller(training, regression='ct')

print('ADF Statistic: {}'.format(adf[0]))
print('p-value : {}'.format(adf[1]))
# p-value가 0.05보다 크므로 데이터가 정상성을 갖지 않는다는 귀무가설 채택. 따라서 차

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

diff_data = training.diff(1)
diff_data = diff_data.dropna()

diff_data.plot()

from statsmodels.tsa.stattools import adfuller

adf = adfuller(diff_data)

print('ADF Statistic: {}'.format(adf[0]))
print('p-value : {}'.format(adf[1]))
# 1차 차분 후 정상성 검정 결과 p-value가 매우 작아져 정상성이 확보되었다.

from pmdarima import auto_arima

auto_model = auto_arima(training, start_P=1, D=1 , Q=1,
                        max_P=2, max_D=2, max_Q=2, m=12,
                        seasonal=True, information_criterion='aic',
                        stepwise=True)

auto_model.summary()

# 학습 데이터셋으로부터 test 데이터 길이만큼 예측
prediction= pd.DataFrame(auto_model.predict(n_periods=len(test)),index=test.index)
prediction.columns = ['predicted_counts']

plt.figure(figsize=(10,6))
plt.plot(training, label="Train") # Train 데이터
plt.plot(prediction, label="Prediction") # 모델이 예상한 그래프
plt.plot(test, label="Test") # 실제 그래프

plt.legend(loc='upper left')
plt.show()

from sklearn.metrics import r2_score,mean_squared_error
test['predicted_counts'] = prediction
r2_score(test['counts'], test['predicted_counts'])
```



ADF Statistic: -2.9023457794184204
p-value : 0.16145777240955378
ADF Statistic: -10.774444937917826
p-value : 2.3391802421181215e-19





Out[65]: 0.4632609709251949

답안

- 불필요 컬럼 Unnamed를 삭제하고, 시계열 데이터를 위한 datetime 컬럼을 생성한 후 index로 설정하였다.
- 이후 시간에 따른 데이터들의 정상성을 확인해보기 위해 decompose를 진행하였고,
- 정확한 숫자를 위해 adfuller를 활용하여 정상성 검정을 시행하였다. 1차 검정결과 정상성 미확보 판단되어
- 차분을 시행한 후 adfuller로 다시 한번 검정 결과 정상성이 확보됨을 확인하였다.
- 이후 auto_model을 활용하여 최적 모델리을 진행하였다.(시간관계상 max_P,D,Q값을 2로 설정하였다.)
- 이후 모델을 통헤 prediction을 생성하였고 실제 test값과 prediction의 R2 Score 확인 결과 정확도는 0.46으로 확인된다.

2-4 분석 결과 활용 가능 여부에 대한 분석 전문가로서의 제안

답안

- 이 모델의 최종적인 예측값이 실제 데이터를 설명하는 R2 Score는 46.3%이다.
- 이정도의 정확도는 그리 높다고 생각할 수 없는 데이터 이다.
- 적어도 70~80% 이상의 설명률이 나왔을 때야 해당 모델을 사용할 수 있다 판단된다.
- 현재의 정확도가 이렇게 낮은 이유는, Input으로 쓰일 데이터에 대한 깔끔한 전처리가 완전히 이루어지지 않았고,
- 모델링을 하는 과정에서 다양한 케이스에 대한 auto_arima를 시행하지 못하였기 때문이라 판단된다.
- 따라서 향후 분석결과를 활요하기 위해서는 정확도를 더 높일 필요가 있다 판단되며,
- 이를위해 P,D,Q값을 조금 더 다양하게 준 다음 최적 모델을 찾을 경우 정확도는 더 올라갈 것으로 생각한다.

3번

3-1 서울에서 영동까지 100km/h로 가고 영동에서 서울까지 80km/로 돌아왔을 때, 평균 속도는?

In [69]:
```
# 조화평균 문제
sokdo = 2/((1/100)+(1/80))
print("서울에서 영동까지 갔다 올 때의 평균 속도 : ", sokdo, 'Km/h')
```

```
서울에서 영동까지 갔다 올 때의 평균 속도 :  88.88888888888889 Km/h
```

3-2 연매출이 3000, 4000, 5000이었다면 연평균 몇배가 증가한 것인가?

In [72]:
```
# 기하평균 문제
a = 3000
b = 4000
c = 5000

ab = (b-a) / a
bc = (c-b) / b

r = (ab * bc) ** (1/2)*100

print("연 평균 증가율은 ", round(r,2), "%이다.")
```

```
연 평균 증가율은  28.87 %이다.
```

3-3 남성, 여성의 등산, 수영에 대한 취미 선호도 빈도표(2x2)를 보고, 남성 중에서 등산을 좋아할 확률을 구하시오

	등산	수영
남자	20	10
여자	15	30

In [74]:
```
a = 20/30
print("남성 중 등산을 조항할 확률은",round(a,2),"이다.")
```

남성 중 등산을 조항할 확률은 0.67 이다.

3-4 표본 10개의 분산이 90일 때 신뢰도 95%로 모분산의 신뢰구간을 추정

In [79]:
```
# 모분산 신뢰구간 공식 : [(n-1) * Var / (카이제곱a/2, n-1), (n-1) * Var / (카이제곱1
n = 10
std = np.sqrt(90)
chi_0_025 = 19.02 # (a/2)
chi_0_975 = 2.7 # 1-(a/2)

under = (n-1) * (std**2) / chi_0_025
upper = (n-1) * (std**2) / chi_0_975

print('모분산의 신뢰구간 :' , under, '~', upper)
```

모분산의 신뢰구간 : 42.586750788643535 ~ 300.0

4번 임상 대상 20명에 대해 혈압약 투약 이전과 이후의 차이가 24, 표준편차 9 신뢰구간 95%, 차이가 존재하는지 확인하려한다.

4-1 귀무가설과 연구가설을 제시하시오

답안

- 같은 표본에 대해 전,후의 평균을 비교하는 검정이므로 paired t-test를 시행한다.
- 귀무가설 : 혈압약 투약 이전과 이후의 차이는 없다. mu(이전) = mu(이후)
- 대립가설 : 혈압약 투약 이전과 이후의 혈압은 차이가 있다. mu(이전) != mu(이후)

4-2 검정 후 귀무가설 기각 여부 제시

In [5]:
```
# 표본의 수가 30개 미만이므로 t검정을 시행해야 한다.
# 95% 신뢰수준의 t 통계량 : 2.069
# t통계량 구하는 공식 : t = (표본평균 - 모평균) / (std / np.sqrt(n))
# (표본평균 - 모평균) = 24

n = 20
std = 9
t = 2.069

under = 24 - t * (std / np.sqrt(n))
upper = 24 + t * (std / np.sqrt(n))
t_statistic = 24 / (std / np.sqrt(n))

print("모평균의 95% 신뢰구간 : ", under, '~ mu~ ', upper)
print("표본평균의 t통계량 : ", t_statistic)
print("95% 신뢰수준의 t 통계량은 2.069이나 평균의 차이 24에 대한 t통계량은",t_statis
print("유의확률 p-value는 기각역에 속한다. 따라서 귀무가설을 기각하고 대립가설을 채택
print("즉 혈압약 투약 이전과 이후의 혈압 차이는 유의하다는 결론을 내린다.")
```

```
모평균의 95% 신뢰구간 :  19.83621781909764 ~ mu~  28.16378218090236
표본평균의 t통계량 :  11.925695879998878
95% 신뢰수준의 t 통계량은 2.069이나 평균의 차이 24에 대한 t통계량은 11.9256958799988
78 이므로
유의확률 p-value는 기각역에 속한다. 따라서 귀무가설을 기각하고 대립가설을 채택한다.
즉 혈압약 투약 이전과 이후의 혈압 차이는 유의하다는 결론을 내린다.
```

5번 공장 X,Y,Z의 평균 출하 소요시간을 여러 일자에 거쳐 측정한 데이터이다. 각 공장의 중위수의 차이가 존재하는지 확인하려 한다.

- problem5.csv

5-1. 연구가설과 귀무가설을 설정하시오

답안

- 순위형 데이터 이므로 비모수적 검정 방법인 크루스컬 왈리스 검정을 시행한다.
- 귀무가설 : 각 공장의 중위수는 차이가 없다.
- 대립가설 : 각 공장의 중위수는 차이가 있다.

5-2. 검정통계량을 구하고 가설을 채택하시오

In [33]:
```
data = pd.read_csv('problem5.csv', encoding='cp949')
data.columns = ['time', 'name', 'rank']
data_x = data.loc[data['name'] == 'X'].reset_index(drop = True)
data_y = data.loc[data['name'] == 'Y'].reset_index(drop = True)
```

```
data_z = data.loc[data['name'] == 'Z'].reset_index(drop = True)

from scipy.stats import kruskal
kruskal(data_x['rank'], data_y['rank'], data_z['rank'])
```

Out[33]: KruskalResult(statistic=0.1249463430741201, pvalue=0.9394382661600987)

답안

- 크루스컬 왈리스 검정 결과 통계량은 0.125, p-value는 0.939로 확인된다.
- 유의확률 p-value가 유의수준 5%인 0.05보다 크기에 귀무가설을 채택한다.
- 즉, 각 공장의 중위수는 차이가 없다는 결론을 내린다.

6번 1개년 50억, 2개년 60억, 3개년 70억의 예산을 가지고 NPV(순현재가치)가 가장 높아지는 안을 제시하시오.

	1개년	2개년	3개년
1안	10	20	15
2안	15	14	19
3안	12	11	30
4안	13	25	20
5안	16	30	24

이 문제 역시 틀리라고 낸 문제이므로.. 과감하게 틀리렵니다.

구글링 해보니 이자율이 필요한것 같은데.. 주어진 자료에서 이자율은 확인이 안됩니다.ㅠ

구글링을 통해 얻어 걸린 파이썬 코드를 확인해봤습니다만, 이해가 가질 않습니다.ㅠ

아래는 구글링으로 확인한 파이썬 풀이 방법

In [34]:
```
from itertools import combinations
import pandas as pd

investment_df = pd.DataFrame([[10, 20, 15],
                              [15, 14, 19],
                              [12, 11, 30],
                              [13, 25, 20],
                              [16, 30, 24]], index=[1, 2, 3, 4, 5], columns=['yr1',
investment_df
```

Out[34]:

	yr1	yr2	yr3
1	10	20	15
2	15	14	19
3	12	11	30
4	13	25	20
5	16	30	24

In [35]:
```
items = [1, 2, 3, 4, 5]

combination_list = list(combinations(items, 2))+list(combinations(items, 3))

result_df = pd.DataFrame()

for c in combination_list:

    temp_df = pd.DataFrame()
    for item in items:
        if item in c:
            temp_df = pd.concat([temp_df, investment_df.loc[item]], axis=1)

    result_df = pd.concat([result_df, temp_df.sum(axis=1)], axis=1)

result_df.columns = combination_list

df_t = result_df.T
df_t = df_t[(df_t['yr1'] < 50) & (df_t['yr2'] < 60) & (df_t['yr3'] < 70)]
df_t['total'] = df_t.sum(axis=1)
df_t.sort_values(by='total', ascending=False)
```

Out[35]:

	yr1	yr2	yr3	total
(2, 3, 4)	40	50	69	159
(1, 3, 4)	35	56	65	156
(1, 2, 4)	38	59	54	151
(1, 2, 3)	37	45	64	146
(4, 5)	29	55	44	128
(3, 5)	28	41	54	123
(2, 5)	31	44	43	118
(1, 5)	26	50	39	115
(3, 4)	25	36	50	111
(2, 4)	28	39	39	106
(1, 4)	23	45	35	103
(2, 3)	27	25	49	101
(1, 3)	22	31	45	98
(1, 2)	25	34	34	93

In [ ]: