

In [14]: #1-1 고유번호 별 가장 최신 연도 만 포함하는 데이터 생성

```
import pandas as pd
import numpy as np

df=pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/29_problem1.csv",encoding= "cp949" )
print(len(df['계약서고유번호'].unique()))
print(df.columns.tolist())
print(df[df['계약서고유번호']==15865])
```

10472

['순번', '계약구분', '재계약횟수', '거주개월', '아파트 이름', '아파트 ID', '아파트 평점', '호실고유번호', '층', '평형대', '계약자고유번호', '계약서고유번호', '입주연도', '퇴거연도', '거주연도', '월세(원)', '보증금(원)', '대표나이', '나이', '성별', '결혼여부', '거주자 수', '퇴거여부']

	순번	계약구분	재계약횟수	거주개월	아파트 이름	아파트 ID	아파트 평점	호실고유번호	층	평형대	...	퇴거연도	\
0	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
1	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
2	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
3	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
4	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
5	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
6	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
7	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
8	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
9	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
10	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
11	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	
12	1	유효	10	222	강남아파트	5	7.0	14520	1	12	...	NaN	

	거주연도	월세(원)	보증금(원)	대표나이	나이	성별	결혼여부	거주자 수	퇴거여부
0	2008	47100	3646000	46	33	남	미혼	3	미퇴거
1	2009	56500	4375000	46	34	남	미혼	3	미퇴거
2	2010	56500	4375000	46	35	남	미혼	3	미퇴거
3	2011	69900	5408000	46	36	남	미혼	3	미퇴거
4	2012	69900	5408000	46	37	남	미혼	3	미퇴거
5	2013	83800	6489000	46	38	남	미혼	3	미퇴거
6	2014	83800	6489000	46	39	남	미혼	3	미퇴거
7	2015	105600	8177000	46	40	남	미혼	3	미퇴거
8	2016	105600	8177000	46	41	남	미혼	3	미퇴거
9	2017	126700	9812000	46	42	남	미혼	3	미퇴거
10	2018	126700	9812000	46	43	남	미혼	3	미퇴거
11	2019	152040	11774400	46	44	남	미혼	3	미퇴거
12	2020	152040	11774400	46	45	남	미혼	3	미퇴거

[13 rows x 23 columns]

```
In [15]: data=df.sort_values(by=['계약서고유번호', '거주연도'], ascending=True).reset_index(drop=True)
data=data.drop_duplicates(subset=['계약서고유번호'], keep='last')
print(data.shape)
```

(10472, 23)

```
In [16]: #1-2 EDA 하고 결측치 처리
print(data.describe(exclude=np.number)) # 미혼, 현재 거주중이 좀 높음
print(data.describe()) # 연령이 max 와 75% 차이 큼 이사이에서 이상치 의심
```

	계약구분	아파트 이름	성별	결혼여부	퇴거여부
count	10410	10472	10472	10472	10472
unique	2	5	2	2	2
top	유효	비둘기아파트	여	미혼	미퇴거
freq	6219	4485	5963	9092	6258

	순번	재계약횟수	거주개월	아파트 ID	아파트 평점 \
count	10472.000000	10472.000000	10472.000000	10472.000000	10330.000000
mean	6419.261173	5.847689	137.096543	1.807678	6.363117
std	3728.707909	3.210563	77.470829	0.866103	1.284452
min	1.000000	1.000000	1.000000	1.000000	5.000000
25%	3169.750000	3.000000	70.000000	1.000000	5.000000
50%	6453.500000	6.000000	136.000000	2.000000	7.000000
75%	9635.250000	9.000000	222.000000	2.000000	7.000000
max	12883.000000	12.000000	323.000000	5.000000	10.000000

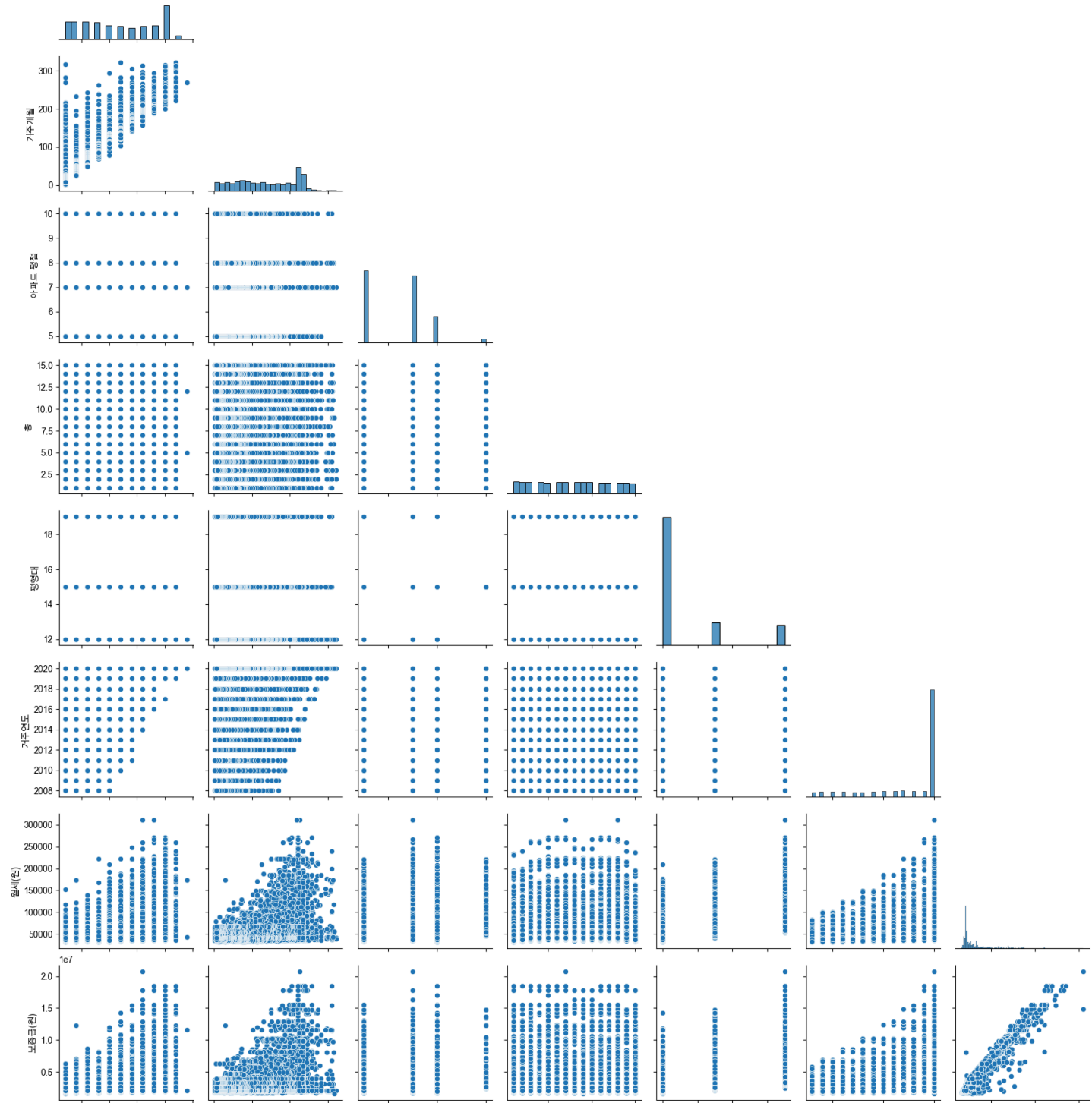
	호실고유번호	층	평형대	계약자고유번호	계약서고유번호 \
count	10472.000000	10472.000000	10472.000000	10472.000000	10472.000000
mean	42963.397154	7.850554	13.214572	48461.711803	46360.784759
std	25017.045008	4.284585	2.332189	26841.150582	26011.368152
min	1.000000	1.000000	12.000000	1.000000	1.000000
25%	21229.000000	4.000000	12.000000	24996.750000	23299.750000
50%	42920.000000	8.000000	12.000000	48563.500000	48062.500000
75%	64573.250000	12.000000	12.000000	73973.250000	69120.250000
max	86891.000000	15.000000	19.000000	86892.000000	86904.000000

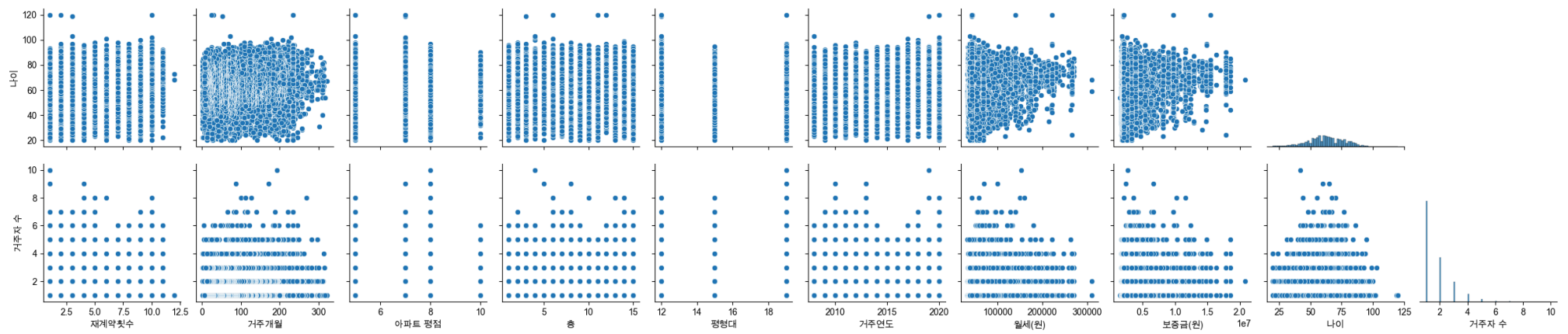
	입주연도	퇴거연도	거주연도	월세(원)	보증금(원) \
count	10472.000000	4214.000000	10472.000000	10472.000000	1.047200e+04
mean	2007.075917	2014.250119	2017.686211	63683.078686	3.787533e+06
std	6.204953	3.735257	3.683041	40101.986955	3.075739e+06
min	1994.000000	2008.000000	2008.000000	31300.000000	1.520000e+06
25%	2002.000000	2011.000000	2016.000000	42300.000000	2.052000e+06
50%	2004.000000	2015.000000	2020.000000	44600.000000	2.144000e+06
75%	2012.000000	2017.000000	2020.000000	66700.000000	3.930000e+06
max	2020.000000	2020.000000	2020.000000	311080.000000	2.078400e+07

	대표나이	나이	거주자 수
count	10472.000000	10472.000000	10472.000000
mean	65.805099	62.491310	1.681150
std	13.769877	14.203338	0.982725
min	21.000000	20.000000	1.000000
25%	57.000000	53.000000	1.000000
50%	65.000000	62.000000	1.000000
75%	75.000000	73.000000	2.000000
max	121.000000	120.000000	10.000000

```
In [17]: import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from matplotlib import font_manager, rc
font_path='/Library/Fonts/Arial Unicode.ttf'
font= font_manager.FontProperties(fname=font_path).get_name()
rc('font',family=font)
warnings.filterwarnings('ignore')
plt.figure(figsize=(20,20))
hist=['재계약횟수', '거주개월', '아파트 평점', '층', '평형대', '거주연도', '월세(원)', '보증금(원)', '나이', '거주자 수']
hist_hue=hist#+'결혼여부'
sns.pairplot(data[hist_hue],corner=True)
plt.show()
```

<Figure size 2000x2000 with 0 Axes>





```
In [18]: for val in ['평형대', '아파트 ID', '거주연도', '성별', '결혼여부', '아파트 평점']:
print(val, '비율')
dis=data[val].value_counts().sort_index()
print(dis/len(data)*100)
```

평형대 비율

평형대

12 75.085943

15 13.235294

19 11.678762

Name: count, dtype: float64

아파트 ID 비율

아파트 ID

1 42.828495

2 38.378533

3 15.326585

4 2.129488

5 1.336898

Name: count, dtype: float64

거주연도 비율

거주연도

2008 2.750191

2009 3.074866

2010 3.008021

2011 2.950726

2012 2.568755

2013 2.511459

2014 3.008021

2015 3.284950

2016 3.456837

2017 3.695569

2018 3.485485

2019 3.428189

2020 62.776929

Name: count, dtype: float64

성별 비율

성별

남 43.057678

여 56.942322

Name: count, dtype: float64

결혼여부 비율

결혼여부

기혼 13.177998

미혼 86.822002

Name: count, dtype: float64

아파트 평점 비율

아파트 평점

5.0 42.150497

7.0 39.237968

8.0 15.145149

10.0 2.110390
Name: count, dtype: float64

```
In [19]: data.isna().sum()
```

```
Out[19]:  순번            0
   계약구분        62
   재계약횟수       0
   거주개월        0
   아파트 이름      0
   아파트 ID        0
   아파트 평점      142
   호실고유번호     0
   층              0
   평형대          0
   계약자고유번호    0
   계약서고유번호    0
   입주연도         0
   퇴거연도        6258
   거주연도         0
   월세(원)         0
   보증금(원)       0
   대표나이        0
   나이            0
   성별            0
   결혼여부        0
   거주자 수        0
   퇴거여부        0
   dtype: int64
```

```
In [20]: # rating은 만족도를 나타내므로 최빈값, 퇴거연도는 퇴거여부를 나타내는 퇴거여부 거주기간을 나타내는 것으로 설명할 수 있다.
# 그래서 그냥 컬럼을 삭제 한다.
data=data.drop(columns=['퇴거연도','계약구분'])
data['아파트 평점']=data['아파트 평점'].fillna(data['아파트 평점'].mode()[0])
data.isna().sum()
print(data.shape)
```

(10472, 21)

```
In [21]: # 이상치처리
# age와 deposit에 이상치가 존재하는 것을 시각적으로 확인 가능
age_iqr=data.describe()['나이']['75%']-data.describe()['나이']['25%']
data=data[(data['나이']<=data.describe()['나이']['75%'] +age_iqr*1.5)&(data['나이']>=data.describe()['나이']['25%'] -age_iqr*1.5)]
deposit_iqr=age_iqr=data.describe()['보증금(원)']['75%']-data.describe()['보증금(원)']['25%']
data=data[(data['보증금(원)']<=data.describe()['보증금(원)']['75%'] +deposit_iqr*1.5 ) & (data['보증금(원)']>=data.describe()['보증금(원)']-deposit_iqr*1.5)]
print(data.shape)
```

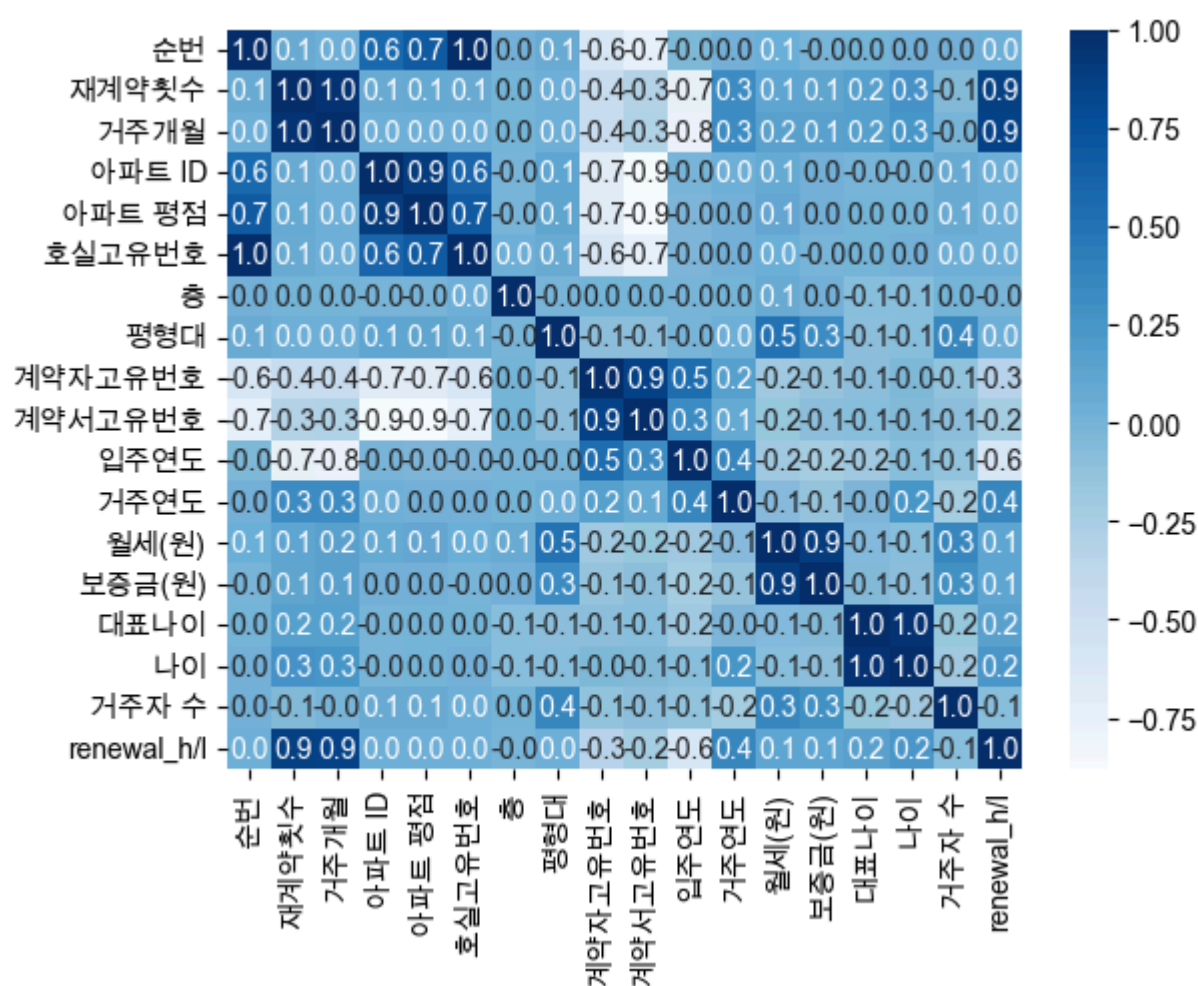

(9034, 21)

```
In [22]: # 재계약회수를 중앙값기준으로 중앙값보다크면 hiw 작으면 low인 renewal_h/l 컬럼 생성
data['renewal_h/l']=np.where(data['재계약횟수']>data['재계약횟수'].median(),'hi','low')
print(data['renewal_h/l'].value_counts())
data['renewal_h/l'].replace({'hi':1,'low':0},inplace=True)
```

```
renewal_h/l
low      4988
hi       4046
Name: count, dtype: int64
```

```
In [23]: # 데이터로 차원을 축소하고 축소가 필요한 이유를 논하시오 축소가 불필요한 경우 축소하지 않은 근거를 논하시오
numeric_features= data.select_dtypes(exclude=['object']).columns
sns.heatmap(data[numeric_features].corr(),annot=True,fmt='.1f',cmap='Blues')
# 난 필요하다고 생각함. 평수에 월세가 당연히 비례할 것이고, 나이와 결혼여부도 어느정도 상관관계가 있을 것임.
```

Out[23]: <Axes: >



```
In [46]: from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA

col=[
    '평형대', '아파트 ID', '성별', '결혼여부', '퇴거여부'
]
data2=data.drop(columns=['순번', '아파트 이름']).copy()
data2=pd.get_dummies(data2, columns=col, drop_first=True)# 공분산성 제거
data2=data2.reset_index(drop=True)

feature= data2.columns.difference(['재계약횟수', 'renewal_h/l']+col)
data2_scaled=StandardScaler().fit_transform(data2[feature])
pca=PCA(n_components=len(feature))
pca.fit(data2_scaled)
```

```
print(pca.explained_variance_ratio_)  
print(sum(pca.explained_variance_ratio_[:17])) # 17개로 80% 설명 가능
```

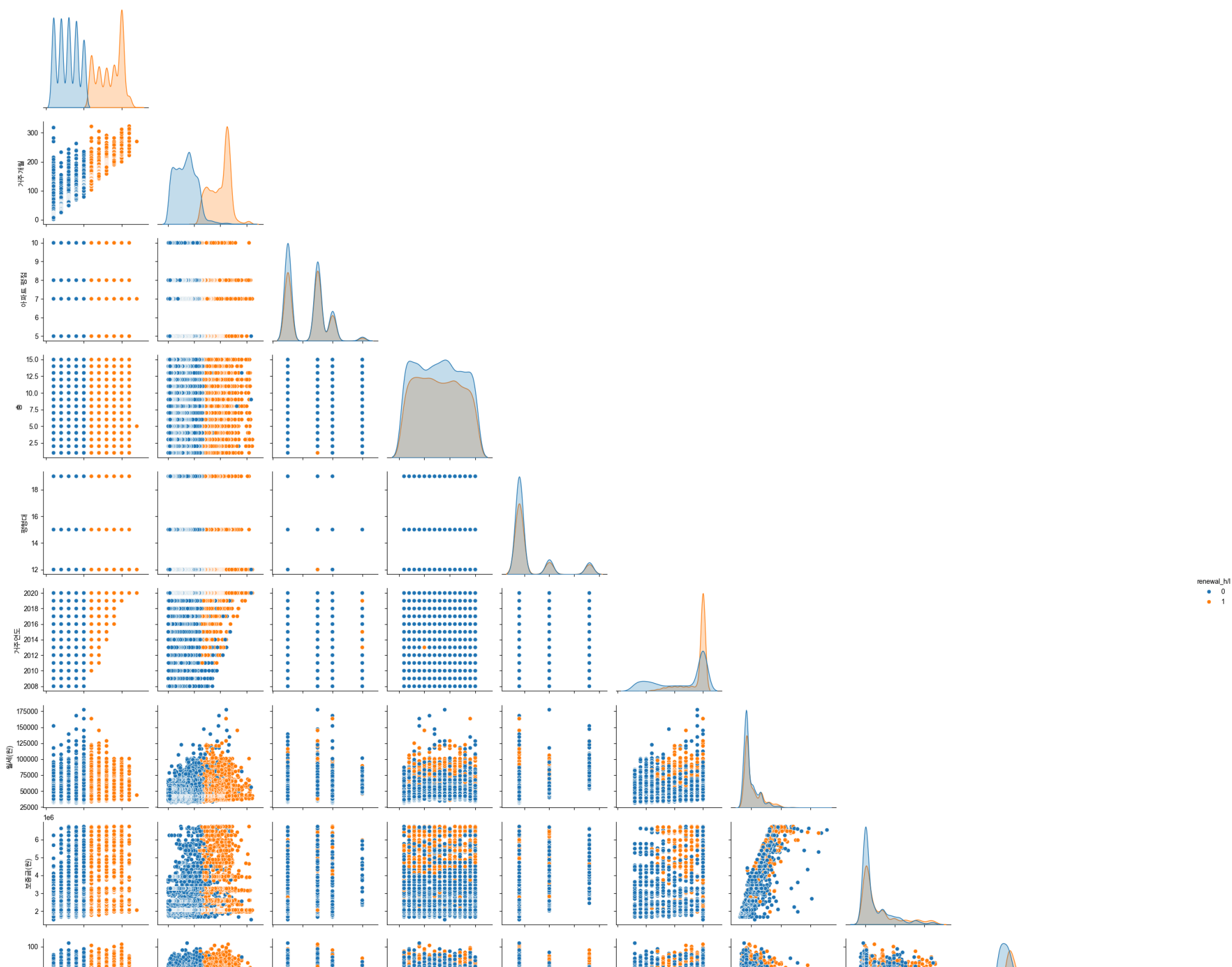
```
[1.88341123e-01 1.27614445e-01 1.04609000e-01 8.84149677e-02  
7.32047448e-02 6.49982194e-02 6.04775572e-02 5.62303607e-02  
4.64958242e-02 4.43846868e-02 3.80132562e-02 3.43373331e-02  
2.56160647e-02 1.85031412e-02 1.07535379e-02 7.14863409e-03  
5.61986904e-03 3.20419956e-03 1.64035229e-03 3.40896810e-04  
5.17867640e-05 0.00000000e+00]  
0.9947627645744784
```

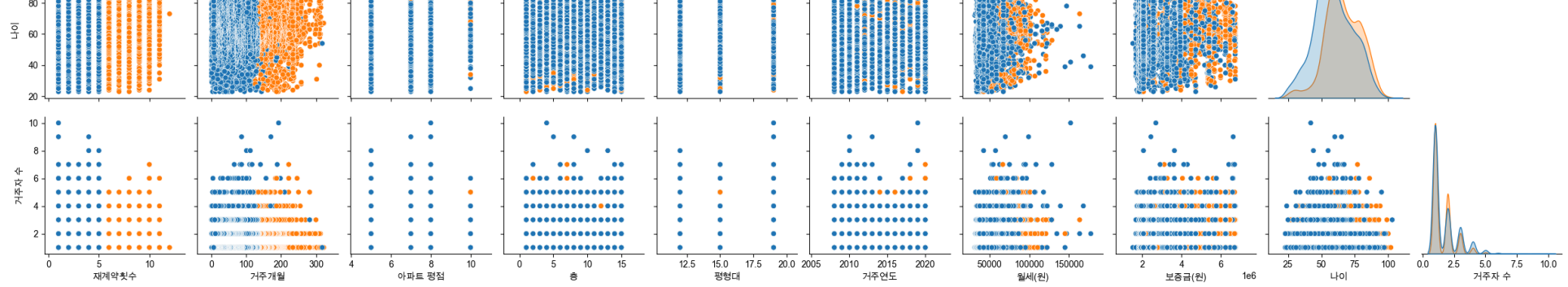
```
In [47]: pca_data=pca.transform(data2_scaled)  
pca_df=pd.DataFrame(data=pca_data[:, :17], columns=['pc'+str(i) for i in range(1,17+1)])  
pca_df['renewal_count']=data2['재계약횟수']
```

```
In [26]: #6 재계약 횟수 high와 low 특징
```

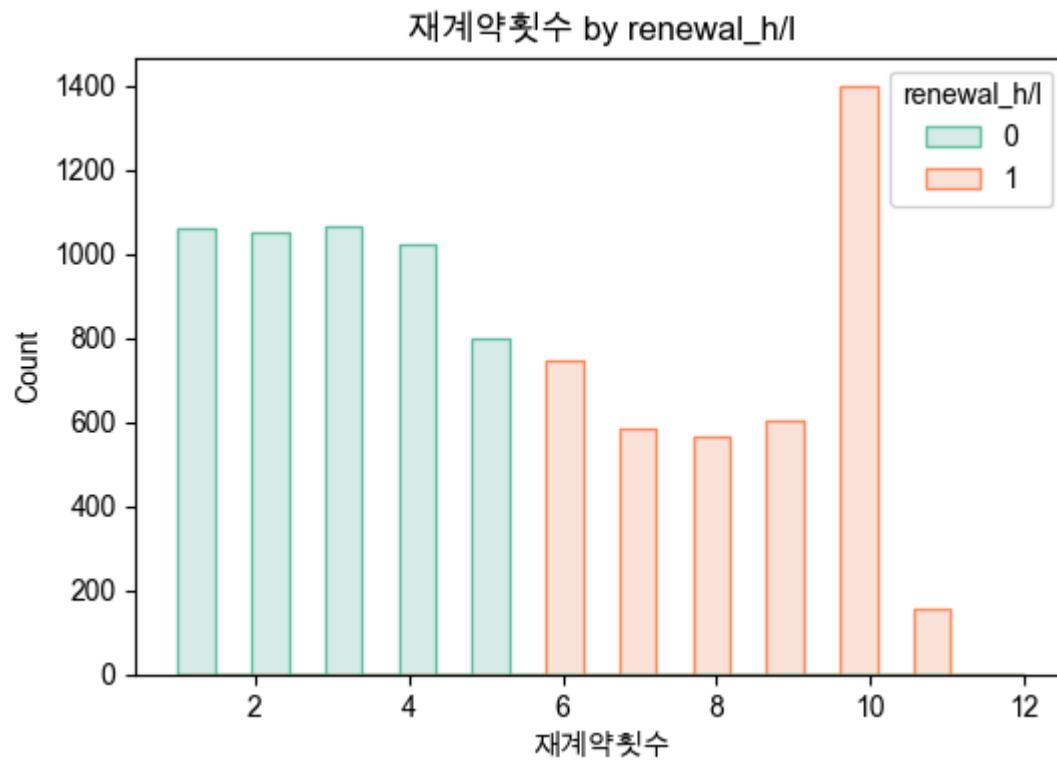
```
data_high=data[data['renewal_h/l']==1]  
data_low=data[data['renewal_h/l']==0]  
hist=['재계약횟수', '거주개월', '아파트 평점', '층', '평형대', '거주연도', '월세(원)', '보증금(원)', '나이', '거주자 수', '퇴거여부']  
hist_hue=hist+['renewal_h/l']  
sns.pairplot(data[hist_hue], hue='renewal_h/l', corner=True)  
# 재계약횟수와 거주개월이 훨씬 큰 high 집단에서,  
# 그리고 거주 년도가 높음.
```

```
Out[26]: <seaborn.axisgrid.PairGrid at 0x17f6dbbd0>
```

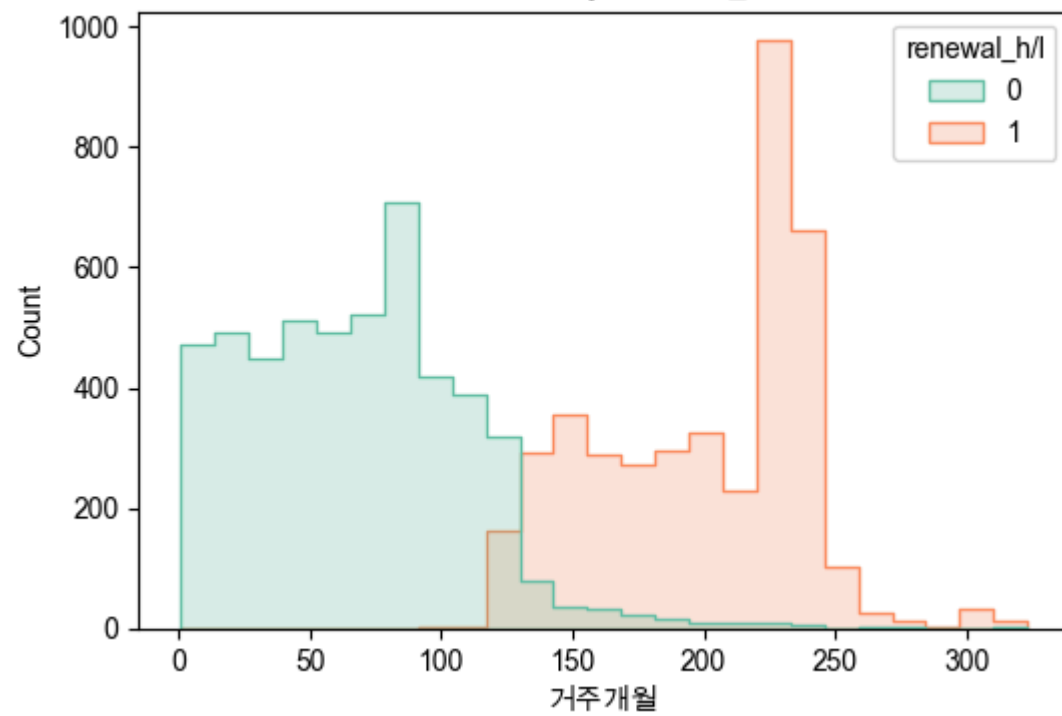




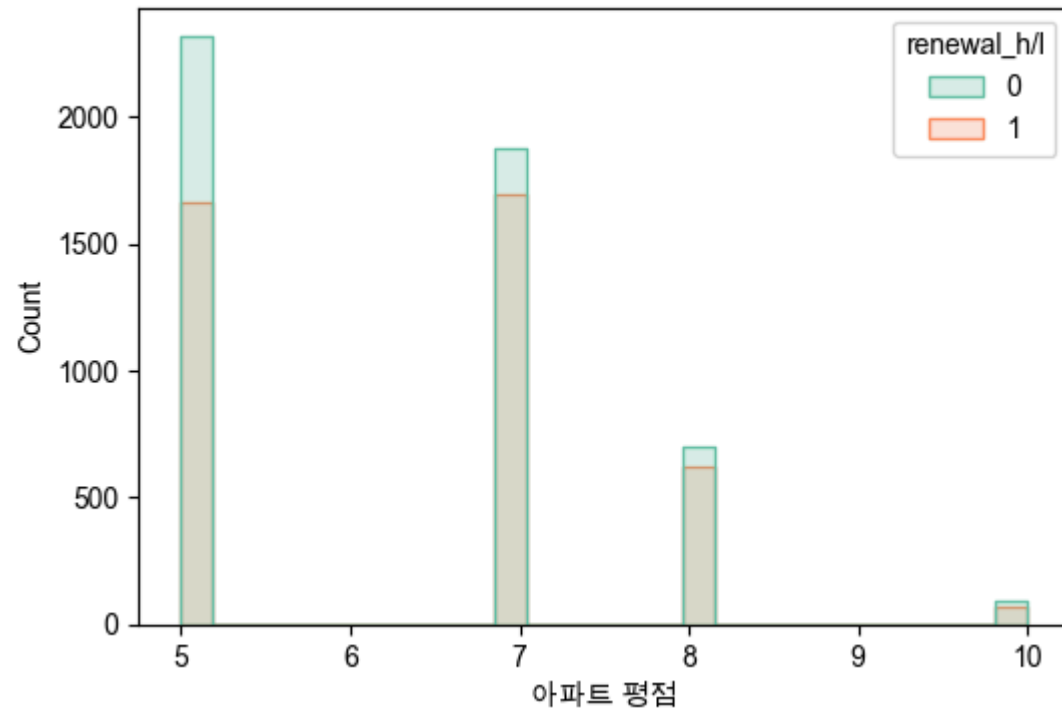
```
In [27]: for col in hist:
plt.figure(figsize=(6, 4))
sns.histplot(data=data, x=col, hue='renewal_h/l', kde=False, palette='Set2', element='step', stat='count')
plt.title(f'{col} by renewal_h/l')
plt.show()
# ...existing code...
```



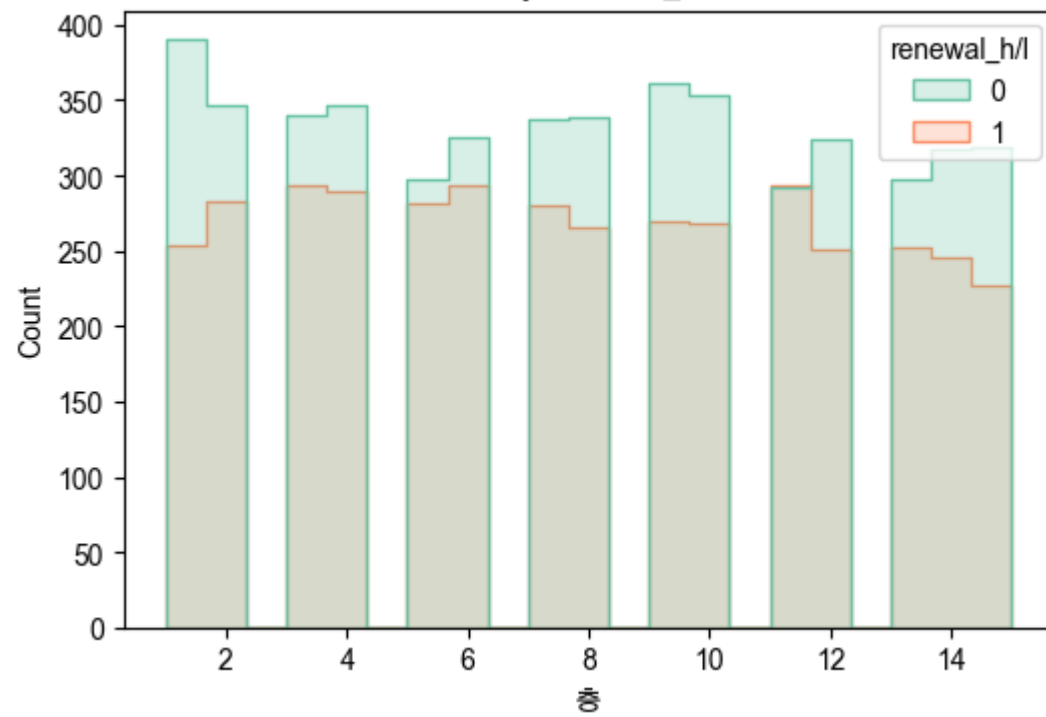
거주개월 by renewal_h/l



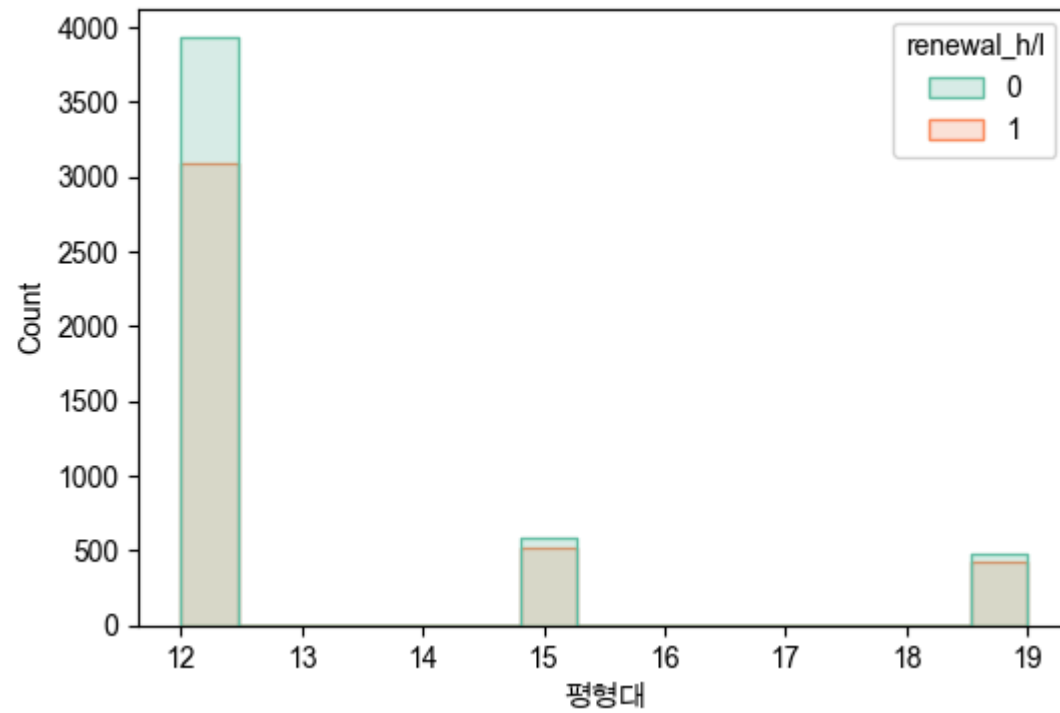
아파트 평점 by renewal_h/l



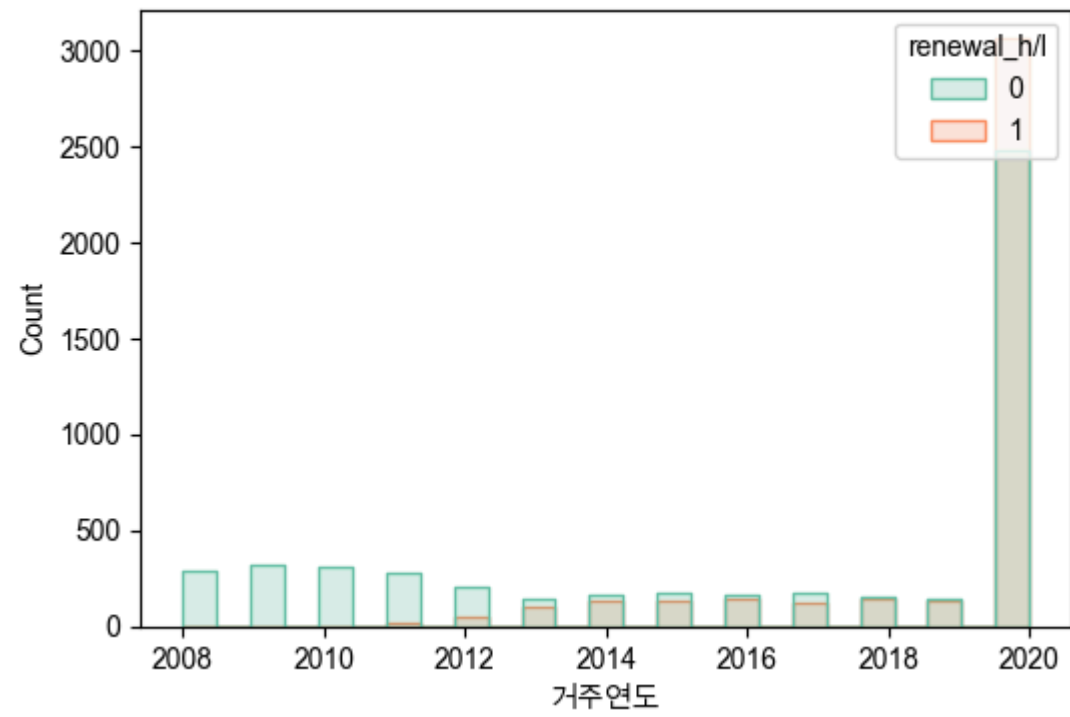
층 by renewal_h/l



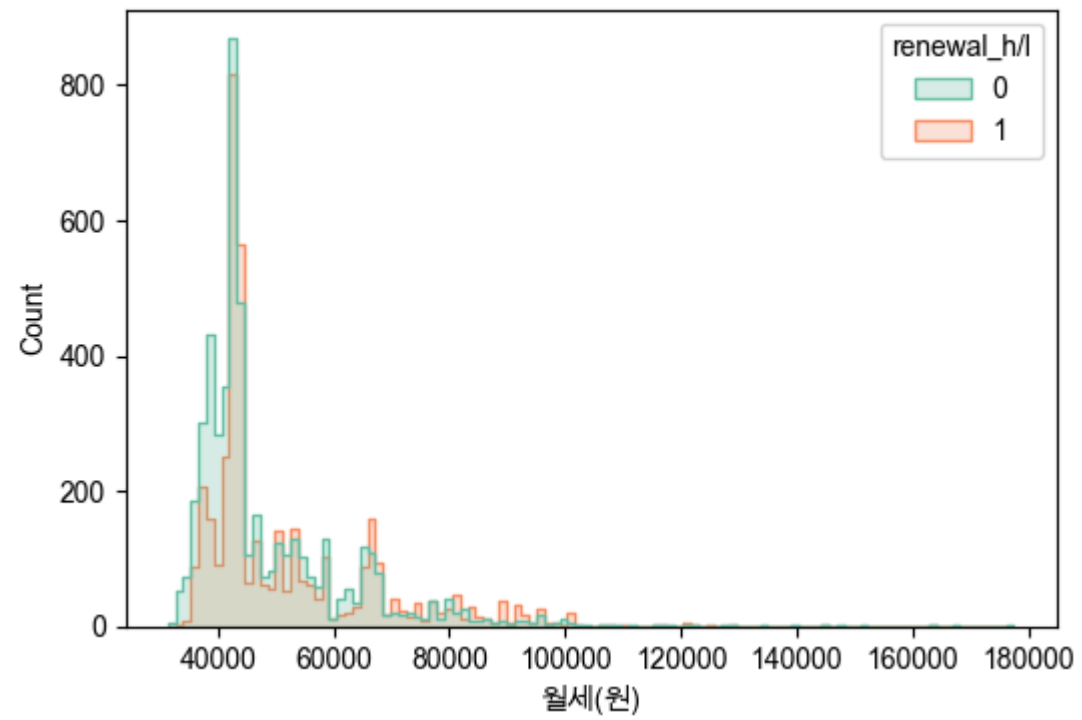
평형대 by renewal_h/l



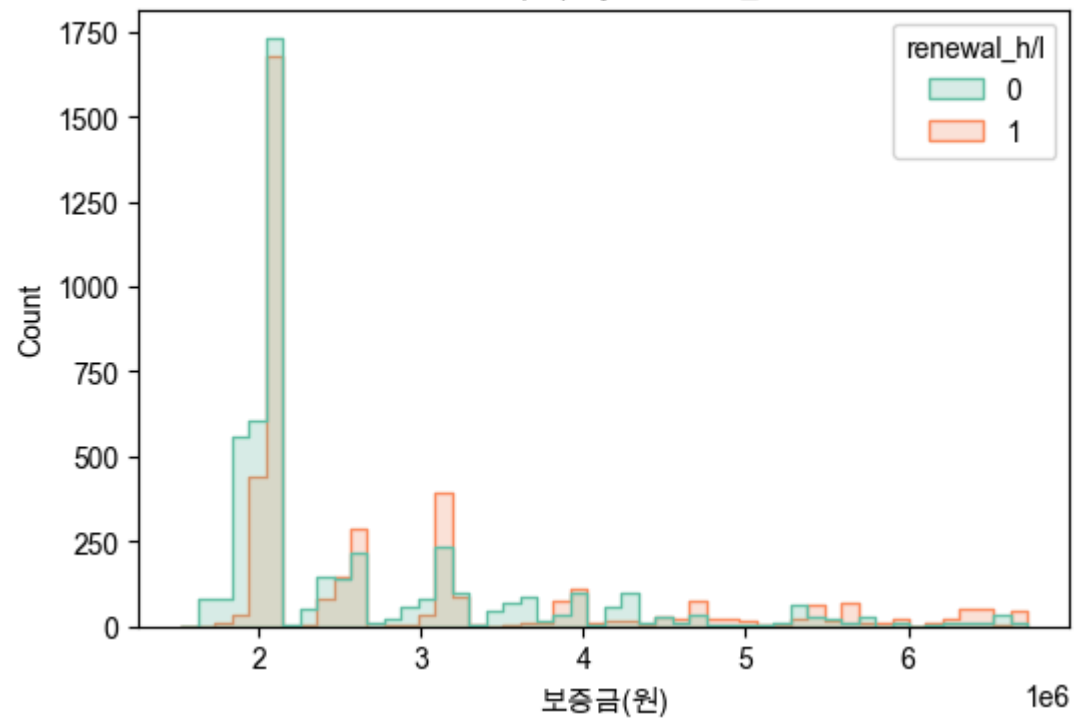
거주연도 by renewal_h/l



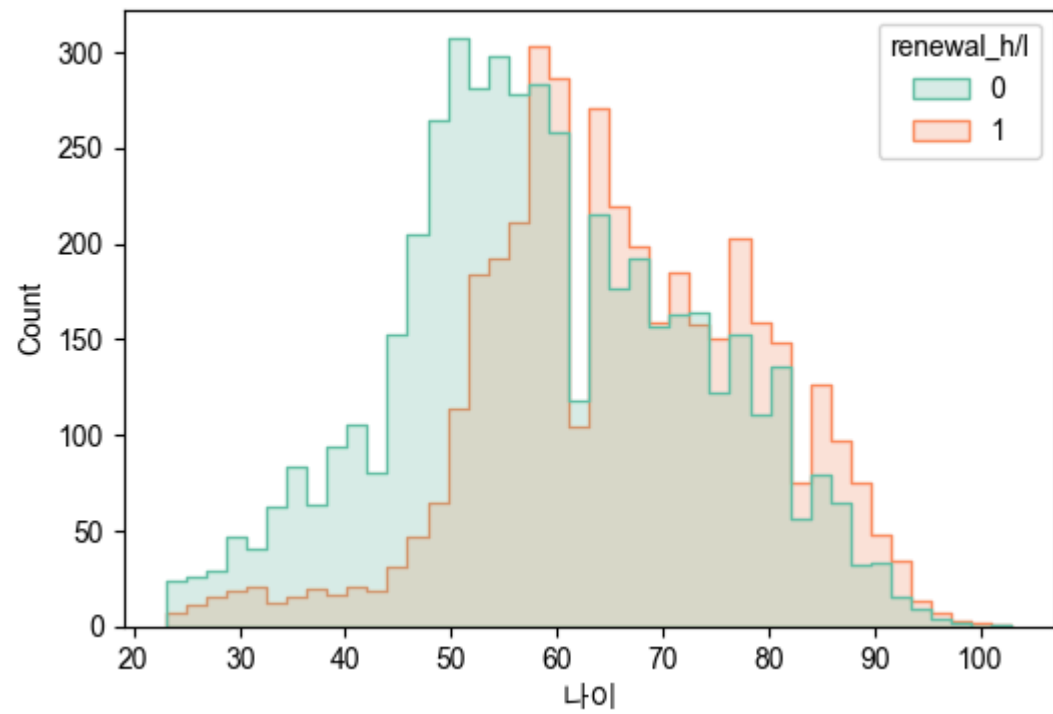
월세(원) by renewal_h/l



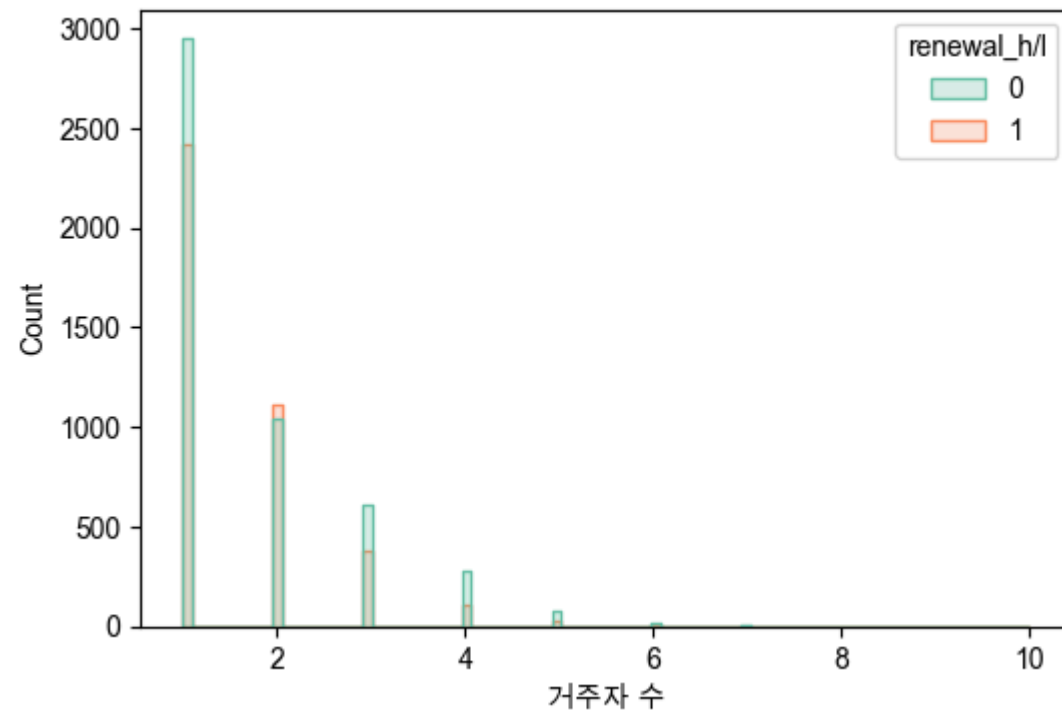
보증금(원) by renewal_h/l



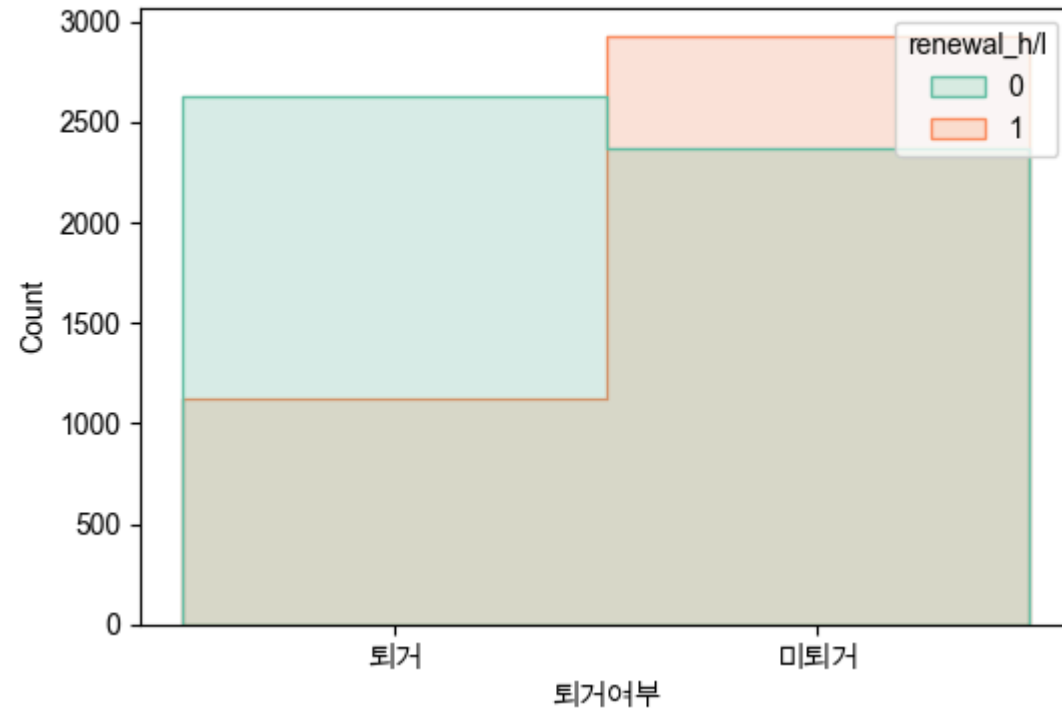
나이 by renewal_h/l



거주자 수 by renewal_h/l



퇴거여부 by renewal_h/l



```
In [28]: # 1-7 재계약 횟수를 종속 변수로 하는 회귀분석을 두가지 이상의 방법론을 통해 수행하고 최종 모델을 결정하시오
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error
numeric_features= pca_df.select_dtypes(exclude=['object']).columns.difference(['renewal_count','renewal_h/l'])
x=pca_df[numeric_features]
y=pca_df['renewal_count']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)

linear=LinearRegression()
linear.fit(x_train,y_train)
print('train score:',linear.score(x_train,y_train))
print('test score:',linear.score(x_test,y_test))
print('MSE:',mean_squared_error(y_test,linear.predict(x_test)))
ada=AdaBoostRegressor(n_estimators=1000,random_state=42)
ada.fit(x_train,y_train)
print('train score:',ada.score(x_train,y_train))
print('test score:',ada.score(x_test,y_test))
print('MSE:',mean_squared_error(y_test,ada.predict(x_test)))
# 선형회귀가 더 성능이 좋음.
```

```
train score: 0.9227503699347348
test score: 0.9304439557197028
MSE: 0.6944889566690504
train score: 0.7168602252400065
test score: 0.704027426036622
MSE: 2.955166387354563
```

```
In [29]: linear.coef_
```

```
Out[29]: array([-0.57966912, -0.24201347,  0.11489143,  0.01282186,  0.09165921,
        -0.0400977 ,  0.44378264,  0.3119945 , -0.20781081, -0.30239518,
         1.30953663,  0.30092028,  0.26826095, -1.28517483, -0.4247072 ,
         0.5851028 , -0.13512194])
```

```
In [ ]: maxidx=np.where(np.abs(linear.coef_)==np.max(np.abs(linear.coef_)))[0][0]
# pca 15 번째임
# pc 15와 상관성이 제일 높은 원차원 피쳐는 다음과 같다.
for i,col in enumerate(feature):
    print(col,np.corrcoef(data2_scaled[:,i],pca_df['pc'+str(maxidx+1)].values)[0,1])
```

거주개월 0.028857786170701404
거주연도 0.028413430415651024
거주자 수 0.4218746848592531
결혼여부_미혼 -0.0819517745905656
계약서고유번호 0.0047291285119923
계약자고유번호 -0.015728457767601373
나이 -0.012186699829817467
대표나이 -0.020442034962045354
보증금(원) -0.28915658427906
성별_여 0.5497104143875599
아파트 ID_2 -0.04572695469262352
아파트 ID_3 -0.0409168386054436
아파트 ID_4 0.21195605208899465
아파트 ID_5 0.003727038651516976
아파트 평점 0.04154406421755638
월세(원) -0.19476021757917114
입주연도 -0.008532257394050634
층 0.05327773774938121
퇴거여부_퇴거 -0.04908634705615579
평형대_15 -0.1615155731944286
평형대_19 0.3651272868492245
호실고유번호 -0.10119353589992722

```
In [49]: result={}
for k in range(1,len(linear.coef_)+1):
    k = 3 # 예: 두 번째로 큰 값
    abs_coef = np.abs(linear.coef_)
    sorted_indices = np.argsort(abs_coef)[::-1] # 내림차순 정렬 인덱스
    maxidx = sorted_indices[k-1] # k번째로 큰 값의 인덱스 (k=1이면 최대값)
    # maxidx=np.where(np.abs(linear.coef_)==np.max(np.abs(linear.coef_)))[0][0]
    # pca 15 번째임
    # pc 15와 상관성이 제일 높은 원차원 피쳐는 다음과 같다.

    for i,col in enumerate(feature):
        # print(col,np.corrcoef(data2_scaled[:,i],pca_df['pc'+str(maxidx+1)].values)[0,1])
        if col not in result:
            result[col]=np.abs(np.corrcoef(data2_scaled[:,i],pca_df['pc'+str(maxidx+1)].values)[0,1])
        else:
            result[col]+=np.abs(np.corrcoef(data2_scaled[:,i],pca_df['pc'+str(maxidx+1)].values)[0,1])
```

```
In [50]: result
```

```
Out [50]: {'거주개월': 1.0704085571428539,
          '거주연도': 0.9235973774278976,
          '거주자 수': 0.030268586243789315,
          '결혼여부_미혼': 0.07894439457670452,
          '계약서고유번호': 0.2713515489126836,
          '계약자고유번호': 4.754075512640511,
          '나이': 0.18732801075388075,
          '대표나이': 0.06272686813169977,
          '보증금(원)': 0.04374268180427143,
          '성별_여': 0.04670704641762157,
          '아파트 ID_2': 0.48723329917991554,
          '아파트 ID_3': 0.4398786363303873,
          '아파트 ID_4': 1.4504955528725867,
          '아파트 ID_5': 1.0119244538943555,
          '아파트 평점': 0.21659702183653504,
          '월세(원)': 0.04841539967620459,
          '입주연도': 1.5928197233476362,
          '층': 0.19461323184222248,
          '퇴거여부_퇴거': 0.24535510710817027,
          '평형대_15': 0.21935740433083148,
          '평형대_19': 0.07524067744476039,
          '호실고유번호': 3.7986932357629137}
```

```
In [32]: #인사이트 eda결과 재계약 횟수가 중앙값 보다 높은 집단은 그렇지 않은 집단보다 약 129개월 더 오래 거주하고
          # 계약자의 70% 가 미퇴거 상태이다.
          # 회귀분석을 통해 평점, 거주기간 이 재계약 횟수에 영향을 주는 인자로 파악된다.
```

```
In [52]: # 회차별로 1번 타자의 출루 가 있는 경우 에대해 득점이 발생했는지 확인하는 분석을 위한 전처리
df=pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/29_problem2.csv", )
```

```
In [53]: df.head(5)
```

```
Out [53]:
```

	game_id	a1_1	a1_2	a2_1	a2_2	a3_1	a3_2	a4_1	a4_2	a5_1	...	a9_2	b1	b2	b3	b4	b5	b6	b7	b8	b9
0	201900016	5	5	5	5	5	5	5	5	5	...	5	0	0	0	0	0	0	0	4	2
1	201900023	6	4	5	5	2	5	1	1	6	...	7	3	0	0	1	0	1	3	1	0
2	201900103	5	6	5	5	1	9	5	4	6	...	6	0	0	1	1	4	1	2	0	1
3	201900112	5	7	6	1	5	5	1	5	1	...	5	0	3	0	0	0	0	1	0	0
4	201900131	5	1	2	5	2	5	7	2	6	...	6	0	2	4	1	2	2	0	1	0

5 rows x 28 columns

```
In [34]: # 먼저 회차별로 1번 타자 혹은 두번째 타자가 홈런을 친경우 회차데이터 제외
# 타자 칼럼 a_col 에저장 data[data[a_col]!=4][a_col] 은 타자 행동에서 4인 경우에 해당하는 데이터 제외
# 1: 안타, 2: 2루타, 3: 3루타, 4: 홈런, 5: 볼넷, 6: 몸에 맞는 볼,
a_col=[x for x in df.columns if 'a' in x]
homerun_idx=df[(df[a_col]!=4)][a_col].dropna().index # 홈런이 아닌 경우의 인덱스
df=df.loc[homerun_idx,:].reset_index(drop=True)
a1_col=[x for x in df.columns if '_1' in x]
a2_col=[x for x in df.columns if '_2' in x]
b_col=[x for x in df.columns if 'b' in x]

score=[]
i=1
for a1,a2,b in zip(a1_col,a2_col,b_col):
    temp=df[['game_id',a1,a2,b]].copy()
    temp.columns=['game_id','1st','2nd','score']
    temp['inning']=i
    i+=1
    score.append(temp)
score=pd.concat(score).reset_index(drop=True)
score['score_label']=np.where(score['score']==0,False,True)
score
```

Out[34]:

	game_id	1st	2nd	score	inning	score_label
0	201900016	5	5	0	1	False
1	201900112	5	7	0	1	False
2	201900131	5	1	0	1	False
3	201900141	6	5	0	1	False
4	201900142	5	6	2	1	True
...
742	201902288	5	7	0	9	False
743	201902301	5	7	0	9	False
744	201902307	1	9	7	9	True
745	201902327	7	5	0	9	False
746	201902373	1	5	0	9	False

747 rows x 6 columns

```
In [35]: # 각 회차마다 1번 타자의 출루가 있는 경우 출루가 있는 경우 행동값이 1,2,3,6 인경우임.
```

```
score1=score[score['1st'].isin([1,2,3,6])].reset_index(drop=True)
```

```
In [36]: score1['score_label'].value_counts()
```

```
Out[36]: score_label
True      144
False      97
Name: count, dtype: int64
```

```
In [37]: # 1에 대해 로지스틱 리그레션 적용 2번 타자의 희생번트 여부에 대한 회귀계수에 대해 검정 적용
from sklearn.model_selection import train_test_split
score1['2nd_bunt']=np.where(score1['2nd']==9,1,0)# 희생번트 여부 9번이 희생번트임
X=score1[['1st','2nd','2nd_bunt']]
y=score1['score_label'].astype(int)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
# 회귀계수 검정
# 회귀계수는 0 , 회귀계수는 0이 아니다.
import statsmodels.api as sm

logit_model=sm.Logit(y_train,sm.add_constant(X_train)).fit()
print(logit_model.summary())
```

```
Optimization terminated successfully.
      Current function value: 0.651746
      Iterations 5
```

Logit Regression Results

```
=====
Dep. Variable:          score_label    No. Observations:          168
Model:                  Logit         Df Residuals:              164
Method:                  MLE          Df Model:                  3
Date:                   Wed, 08 Oct 2025    Pseudo R-squ.:          0.04792
Time:                   10:09:29           Log-Likelihood:         -109.49
converged:               True             LL-Null:              -115.00
Covariance Type:         nonrobust         LLR p-value:            0.01161
=====
```

	coef	std err	z	P> z	[0.025	0.975]
const	1.0056	0.516	1.950	0.051	-0.005	2.016
1st	0.0715	0.075	0.949	0.343	-0.076	0.219
2nd	-0.2295	0.089	-2.568	0.010	-0.405	-0.054
2nd_bunt	1.7165	0.585	2.936	0.003	0.571	2.862

```
=====
```



```
In [54]: X_train.shape
```

```
Out[54]: (168, 3)
```

```
In [38]: # SMOTE 적용
from imblearn.over_sampling import SMOTE
smote=SMOTE(sampling_strategy='minority',random_state=42)
x_train_res,y_train_res=smote.fit_resample(X_train,y_train)
print(y_train_res.value_counts())
```

```
score_label
0      95
1      95
Name: count, dtype: int64
```

```
In [39]: logit_model=sm.Logit(y_train_res,sm.add_constant(x_train_res)).fit()
print(logit_model.summary())
#번트가 유의미하다.
```

```
Optimization terminated successfully.
Current function value: 0.652728
Iterations 5
```

Logit Regression Results

```
=====
Dep. Variable:          score_label   No. Observations:          190
Model:                  Logit         Df Residuals:             186
Method:                 MLE           Df Model:                 3
Date:                  Wed, 08 Oct 2025   Pseudo R-squ.:           0.05831
Time:                  10:09:29          Log-Likelihood:          -124.02
converged:              True            LL-Null:                -131.70
Covariance Type:        nonrobust        LLR p-value:             0.001534
=====
```

	coef	std err	z	P> z	[0.025	0.975]
const	0.7924	0.487	1.629	0.103	-0.161	1.746
1st	0.0920	0.071	1.293	0.196	-0.047	0.231
2nd	-0.2517	0.085	-2.948	0.003	-0.419	-0.084
2nd_bunt	1.8826	0.553	3.403	0.001	0.798	2.967

```
=====
```

```
In [40]: #통계 분석 제품 A의 불량율은 0.03 -> 이항분포/ 베르누이 짐작하고 있어야함
#25개의 제품을 뽑았을 때 3개가 불량일 확율을 구하시오
import math
n=25
x=3
p=0.03
```

```
answer=math.factorial(n)/math.factorial(x)/math.factorial(n-x)*p**x*(1-p)**(n-x)
print(answer)
```

0.0317738436398321

Thought for 1m 21s

두 회사의 "양품 비율(율)" 차이 검정이므로 두 집단 비율 차이 검정을 쓰면 됩니다. 표본수가 충분히 커서(모든 기대도수 > 5)

- 이항비율 2표본 z-검정(=카이제곱 독립성 검정과 동치, $df=1$)을 쓰는 게 표준이고,
- 표본이 아주 작을 때만 Fisher의 정확 검정을 고려합니다.

데이터 요약

- C사: 1000개 중 600개 양품 $\rightarrow \hat{p}_C = 0.60$ (불량 400)
- D사: 500개 중 200개 양품 $\rightarrow \hat{p}_D = 0.40$ (불량 300)

1) 두 비율 z-검정

- 귀무가설 $H_0: p_C = p_D$
- 대립가설 $H_1: p_C \neq p_D$

$$\text{풀드 비율 } \hat{p} = \frac{600+200}{1000+500} = 0.5333$$

표준오차

$$SE = \sqrt{\hat{p}(1 - \hat{p}) \left(\frac{1}{1000} + \frac{1}{500} \right)} \approx 0.02733$$

검정통계량

$$z = \frac{0.60 - 0.40}{SE} \approx 7.32$$

양측 p값 $\approx 2.5 \times 10^{-13}$ (매우 유의)

결론: 유의수준 0.05(또는 0.01)에서 양품을 차이가 통계적⁵ 유의합니다. C사가 D사보다 높습니다.

좋아요. 표본이 작을 때 두 집단의 비율 차이를 검정하려면 피셔의 정확 검정(Fisher's exact test) 을 쓰면 됩니다. 2×2 분할표의 주어진 주변합을 고정하고, 관측된 표의 하이퍼지오메트릭 분포 확률(및 그보다 “같거나 더 극단적인” 표들의 확률)을 합산해 p값을 계산하는 방식이라, 표본이 작아도 정확합니다.

1) 문제 세팅 (2×2 표)

양품(성공)과 불량(실패)로 나누면:

	양품(성공)	불량(실패)	합계
C사	a	b	a+b
D사	c	d	c+d
합계	a+c	b+d	n

- 귀무가설 H_0 : 두 집단의 양품 비율이 같다 (독립).
- 대립가설 H_1 :
 - 양측: $p_C \neq p_D \rightarrow \text{alternative}=\text{"two-sided"}$
 - 단측(예: C사가 더 높다): $p_C > p_D \rightarrow \text{alternative}=\text{"greater"}$

(참고) 단일 표의 확률


주변합 고정 시, 관측된 a의 확률은 하이퍼지오메트릭:

$$P(A = a) = \frac{\binom{a+b}{a} \binom{c+d}{c}}{\binom{n}{a+c}}$$

양측 p값은 관측표보다 확률이 작거나 같은 모든 표의 확률을 합산(보수적).

2) Python 코드 (SciPy)

python

 Copy code

```
import numpy as np
from scipy.stats import fisher_exact

# 예시: C사 8개 중 양품 6, D사 7개 중 양품 2 (작은 표본 가정 예)
# 표는 [[a, b], [c, d]] 순서
table = np.array([[6, 2], [3, 2]]) # C: 양품 6, 불량 2, D: 양품 3, 불량 2
```

```

table = np.array([[0, 2], # C: 양품 0, 불량 2
                  [2, 5]]) # D: 양품 2, 불량 5

# 1) 양측 검정 (p_C != p_D)
oddsratio, p_two = fisher_exact(table, alternative="two-sided")
print("odds ratio:", oddsratio, "two-sided p:", p_two)

# 2) 단측 검정 (C사가 더 높다: p_C > p_D)
_, p_greater = fisher_exact(table, alternative="greater")
print("one-sided (C > D) p:", p_greater)

# 3) 단측 검정 (D사가 더 높다: p_C < p_D)
_, p_less = fisher_exact(table, alternative="less")
print("one-sided (C < D) p:", p_less)

```

- **oddsratio**: 오즈비(OR). 1보다 크면 C사의 양품 오↓ 더 큼.
- **p-value**: 설정한 대립가설에 맞춘 p값.

```

In [41]: #통계 2
# C 사 생산 제품중 1000개중 양품 600 개 D 사 제품중 500개중 양품이 200개  두회사의 양품율에 차이가 있는지 검정
# 귀무 : 두 양품률 차이 없다.
# 대립 : 차이있다.
# 표본이크니깐 정규분포 따를것
#
import numpy as np
import scipy.stats as stats
alpha=0.05
n1=1000
n2=500
p1_hat = 600/n1
p2_hat = 200/n2
z_score= ((p1_hat-p2_hat)-0)/((np.sqrt((p1_hat*(1-p1_hat)/n1)+(p2_hat*(1-p2_hat)/n2)))
rv=stats.norm()
print('0.05 일때 임계치',round(rv.ppf(0.95),4))
print(z_score)
print('pvalue',(1-rv.cdf(z_score)))

```

```

0.05 일때 임계치 1.6449
7.453559924999298
pvalue 4.54081217071689e-14

```

```

In [42]: import numpy as np
from statsmodels.stats.proportion import proportions_ztest

```

```
# 예시 데이터
# C사: 1000개 중 600개 양품
# D사: 500개 중 200개 양품
count = np.array([600, 200]) # 각 집단의 "성공(양품)" 개수
nobs = np.array([1000, 500]) # 각 집단의 전체 개수

# 양측 검정 (p1 != p2)
stat, pval = proportions_ztest(count, nobs, alternative="two-sided")

print("z 통계량 =", stat)
print("p-value =", pval)
```

z 통계량 = 7.319250547113997
p-value = 2.493596474326038e-13

```
In [43]: # 각 차종별 범퍼의 파손정도가 유의 한지
# 각 차종별 파손정도는 동일하다
# 동일하지 않다 -> 원웨이 아노바

df=pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/29_problem3.csv")

import scipy.stats as stats

data_a=df[df['name']=="A"]['ratio']
data_b=df[df['name']=="B"]['ratio']
data_c=df[df['name']=="C"]['ratio']
data_d=df[df['name']=="D"]['ratio']

print(stats.f_oneway(data_a,data_b,data_c,data_d))
```

F_onewayResult(statistic=24.97695307518529, pvalue=2.8174779556216382e-06)

```
In [44]: # 귀무가설을 채택하면 의미를 해석하고 사후 분석을
from statsmodels.stats.multicomp import MultiComparison

com=MultiComparison(df['ratio'],df['name'])
result=com.tukeyhsd()
print(result.summary())# a-d 제외하고 차이존재
```

Multiple Comparison of Means – Tukey HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
A	B	5.3934	0.001	2.1785	8.6083	True
A	C	-4.2156	0.0085	-7.4305	-1.0007	True
A	D	-0.7086	0.9207	-3.9235	2.5063	False
B	C	-9.609	0.0	-12.8239	-6.3941	True
B	D	-6.102	0.0003	-9.3169	-2.8871	True
C	D	3.507	0.0302	0.2921	6.7219	True

```
In [45]: # L1 L2 L3 생산라인에서 13% 37% 50% 생산하면 1.1% 2.1% 3.3% 불량 률을 갖는다. 불량 뺐을때 L1 일확율
A= 0.13*0.011 + 0.37*0.021 + 0.5 * 0.033
B= 0.13*0.011
print(B/A)

0.055642023346303505
```

```
In [ ]:
```