In [1]:
```
import pandas as pd
import numpy as np
file_path ='./cafedata/gyeonggi-pricedata.csv'
df = pd.read_csv(file_path)
df.head()
```

Out[1]:

	뚜레쥬르 지점	뚜레쥬르 화정중앙	뚜레쥬르 화정덕양구청	뚜레쥬르 마두역	뚜레쥬르 일산동국대병원점	뚜레쥬르 일산산들	뚜레쥬르 광명휴먼시아	뚜레쥬르 광명하안	뚜레쥬르 곤지암	뚜레쥬르 광주탄벌	...	뚜레쥬르 평택지제영신	뚜레쥬르 하남신장	뚜레쥬르 미사강변	뚜레쥬르 화성시청	뚜레쥬르 동탄지스타	뚜레쥬르 동탄메타폴리스	뚜레쥬르 동탄역	뚜레쥬르 향남2지구	뚜레쥬르 병점금강	뚜레쥬르 병점중심상가
0	마늘단짝 고구마	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0	...	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0
1	깊은 밤 뺑스위스	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0	...	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0
2	BELT 샌드위치	NaN	7500.0	6900.0	6900.0	NaN	7100.0	7100.0	NaN	6900.0	...	7300.0	6900.0	7300.0	6900.0	6900.0	6900.0	NaN	6900.0	7200.0	6900.0
3	BLT콥 샐러드	NaN	NaN	NaN	NaN	8500.0	NaN	NaN	8500.0	8500.0	...	8500.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8500.0
4	쉬림프 에그 샐러드	NaN	10500.0	NaN	10500.0	NaN	NaN	NaN	10500.0	NaN	...	NaN	NaN	NaN	10500.0	NaN	NaN	NaN	NaN	11000.0	NaN

5 rows × 81 columns

In [2]:
```
print(df.columns)
```

```
Index(['뚜레쥬르 지점', '뚜레쥬르 화정중앙', '뚜레쥬르 화정덕양구청', '뚜레쥬르 마두역', '뚜레쥬르 일산동국대병원점',
       '뚜레쥬르 일산산들', '뚜레쥬르 광명휴먼시아', '뚜레쥬르 광명하안', '뚜레쥬르 곤지암', '뚜레쥬르 광주탄벌',
       '뚜레쥬르 구리인창삼보', '뚜레쥬르 카페구리갈매', '뚜레쥬르 계양구청', '뚜레쥬르 박촌역', '뚜레쥬르 인천시청',
       '뚜레쥬르 만수3지구', '뚜레쥬르 소래포구역', '뚜레쥬르 간석벽산', '뚜레쥬르 인하대역', '뚜레쥬르 제물포역',
       '뚜레쥬르 간석역', '뚜레쥬르 인천동수역', '뚜레쥬르 부평청천', '뚜레쥬르 부평신일해피트리', '뚜레쥬르 카페석남',
       '뚜레쥬르 나은병원', '뚜레쥬르 송도더테라스', '뚜레쥬르 송도그린워크', '뚜레쥬르 연수무지개', '뚜레쥬르 영종운서',
       '뚜레쥬르 수원천천푸르지오', '뚜레쥬르 매교솔밭사거리', '뚜레쥬르 수원역로데오', '뚜레쥬르 매교힐스테이트',
       '뚜레쥬르 배곧한라비발디', '뚜레쥬르 카페시흥정왕', '뚜레쥬르 시화세종', '뚜레쥬르 카페안산중앙', '뚜레쥬르 안산다농',
       '뚜레쥬르 안산센트럴포레', '뚜레쥬르 안양호계현대', '뚜레쥬르 안양수촌마을', '뚜레쥬르 평촌학원가',
       '뚜레쥬르 양주옥정역', '뚜레쥬르 양주옥정고', '뚜레쥬르 옥정한신더휴', '뚜레쥬르 양평서종', '뚜레쥬르 여주역푸르지오',
       '뚜레쥬르 여주오학', '뚜레쥬르 오산법원', '뚜레쥬르 오산누읍', '뚜레쥬르 오산역', '뚜레쥬르 용인흥덕',
       '뚜레쥬르 동천동문', '뚜레쥬르 용인데이파크', '뚜레쥬르 수지구청역', '뚜레쥬르 용인죽전', '뚜레쥬르 용인파머스마켓',
       '뚜레쥬르 용인양지', '뚜레쥬르 용인둔전', '뚜레쥬르 의왕역', '뚜레쥬르 오전중앙', '뚜레쥬르 신곡초교',
       '뚜레쥬르 의정부신곡', '뚜레쥬르 의정부센트럴', '뚜레쥬르 이천SK하이닉스', '뚜레쥬르 이천창전사거리',
       '뚜레쥬르 파주운정가람', '뚜레쥬르 파주금촌역', '뚜레쥬르 평택에이스고덕', '뚜레쥬르 평택법원', '뚜레쥬르 평택지제영신',
       '뚜레쥬르 하남신장', '뚜레쥬르 미사강변', '뚜레쥬르 화성시청', '뚜레쥬르 동탄지스타', '뚜레쥬르 동탄메타폴리스',
       '뚜레쥬르 동탄역', '뚜레쥬르 향남2지구', '뚜레쥬르 병점금강', '뚜레쥬르 병점중심상가'],
      dtype='object')
```

In [3]:
```
import re

def categorize_menu(df):
    # 키워드 기반 카테고리 매핑 딕셔너리
    category_keywords = {
        '샌드위치류': ['샌드위치', 'BELT', 'BLT', 'V.E.L.T'],
        '샐러드류': ['샐러드'],
        '식빵류': ['식빵', '우유롤', '우유 브레드', '소버식빵'],
        '크림빵': ['크림가득 메론빵','마담 얼그레이 크림번','순진우유크림빵','겹겹이 연유 크림 데니쉬','사르르 고구마케이크빵','사르르 우유크림빵','빵속에리얼초코','카페모카크림빵','까까웨뜨'],
        '피자빵,고로케': ['고로케', '소시지브레드','피자토스트','NEW어니언소시지포카치아'],
        '파이/패스트리': ['바통쉬크레','크라상','애플파이','유자파이'],
        '간식빵': ['소금버터롤','치즈방앗간','깨찰빵','소보로빵','오리지널 커피번','카페모카빵','과배기','옛날 단팥 도넛',r'^단팥빵$','단팥소보로빵'],
        '신제품': ['마구마구', '단짝', '뺑스위스']
    }

    # 새로운 카테고리 컬럼 생성
    df['카테고리'] = '기타'  # 기본값

    # 각 메뉴명에 대해 카테고리 매핑
    for idx, menu_name in enumerate(df['뚜레쥬르 지점']):
        if pd.isna(menu_name):  # null 체크
            continue

        menu_name = str(menu_name).lower()  # 소문자 변환

        # 각 카테고리의 키워드 체크
        for category, keywords in category_keywords.items():
            if any(keyword.lower() in menu_name for keyword in keywords):
                df.loc[idx, '카테고리'] = category
                break

    return df

def analyze_categories_by_store(df):
    # 매장별 카테고리별 기본 통계
    stores = df.columns[1:-1]  # 첫 번째 열(메뉴명)과 마지막 열(카테고리) 제외

    # 카테고리별 기본 통계
    category_stats = pd.DataFrame()

    for store in stores:
        # 매장별 데이터 숫자로 변환 (오류 방지)
        df[store] = pd.to_numeric(df[store], errors='coerce')

        temp = df.groupby('카테고리').agg({store: 'mean'})
        temp.reset_index(inplace=True)
        temp.rename(columns={store: '평균 가격'}, inplace=True)
        temp['매장명'] = store
        category_stats = pd.concat([category_stats, temp], axis=0)

    return category_stats

def pivot_store_category(stats):
    # 피벗 테이블 생성
    pivot_table = stats.pivot_table(index='매장명', columns='카테고리', values='평균 가격', aggfunc='mean')
    pivot_table=pivot_table.round(1)
    pivot_table.reset_index(inplace=True)
    return pivot_table

# 데이터 로드 및 처리
def process_bakery_data(filepath):
    # CSV 파일 읽기
    df = pd.read_csv(filepath)

    # 카테고리 지정
    df = categorize_menu(df)

    # 매장별 카테고리별 분석
    stats = analyze_categories_by_store(df)
```

```
    # 피벗 테이블 생성
    pivot_table = pivot_store_category(stats)

    return df, pivot_table

# 파일 처리 및 결과 생성
df, pivot_table = process_bakery_data(file_path)

# 카테고리화된 데이터 및 매장별 통계 표시
from IPython.display import display

# print("카테고리화된 가격 데이터 (처음 5개 행)")
# display(df.head())
#########################################################################################################################

storeinfo_filepath='./adress_process/gyeonggi-adress.csv'

def process_address(address):
    try:
        # 수동 수정
        if address == '경기도 동탄지성로469번길 60 5단지 상가1동107호,108호,109호':
            return '경기도 화성시'
        elif address == '경기도 동탄지성로469번길 60 5단지 상가1동107호,108호,109호':
            return '경기도 화성시'

        # 정규표현식으로 '충청남도 XX시' 추출
        match = (
            re.match(r'경기도\s+\w+시', address) or
            re.match(r'경기도\s+\w+군', address) or
            re.match(r'인천광역시\s+\w+구', address) or
            re.match(r'인천광역시\s+\w+군', address)
        )

        if match:
            return match.group()

        # 기본값 반환
        return address
    except Exception as e:
        print(f"주소 처리 중 오류 발생: {address}, {e}")
        return address

def load_store_info(storeinfo_filepath):
    store_info = pd.read_csv(storeinfo_filepath)
    # 주소 컬럼 처리
    store_info['주소'] = store_info['주소'].apply(process_address)
    return store_info



def process_bakery_data(price_filepath, store_info_filepath):
    # 가격 데이터 로드
    df = pd.read_csv(price_filepath)

    # 매장 정보 데이터 로드
    store_info = load_store_info(store_info_filepath)

    # 카테고리 지정
    df = categorize_menu(df)

    # 매장별 카테고리별 분석
    stats = analyze_categories_by_store(df)

    # 피벗 테이블 생성 후 매장 정보 병합
    pivot_table = pivot_store_category(stats)
    result = pd.merge(pivot_table, store_info,
                      left_on='매장명',
                      right_on='매장',
                      how='left')

    # 컬럼 순서 재정렬
    columns = ['매장명', '주소', '지역'] + [col for col in result.columns
              if col not in ['매장명', '매장', '주소', '지역']]
    result = result[columns]

    return df, result

# 실제 파일 경로로 호출
df, result = process_bakery_data('./cafedata/gyeonggi-pricedata.csv',
                                 './adress_process/gyeonggi_adress.csv')

# 결과 출력
print("\n매장별 카테고리별 평균 가격 (주소 정보 포함)")
display(result)
```

매장별 카테고리별 평균 가격 (주소 정보 포함)

	매장명	주소	지역	간식빵	기타	샌드위치류	샐러드류	식빵류	신제품	크림빵	파이/패스트리	피자빵,고로케
0	뚜레쥬르 간석벽산	인천광역시 남동구	경기도	3350.0	4200.0	7128.6	8350.0	5000.0	4933.3	3728.6	2933.3	3040.0
1	뚜레쥬르 간석역	인천광역시 미추홀구	경기도	3155.6	4170.2	7183.3	8333.3	5206.7	4600.0	3737.5	2966.7	3166.7
2	뚜레쥬르 계양구청	인천광역시 계양구	경기도	3262.5	4221.6	NaN	NaN	5760.0	4600.0	3966.7	3000.0	3250.0
3	뚜레쥬르 곤지암	경기도 광주시	경기도	3256.2	4281.7	6570.6	9085.7	5072.2	4933.3	3120.0	3100.0	3142.9
4	뚜레쥬르 광명하안	경기도 광명시	경기도	3133.3	4474.1	7150.0	8300.0	5221.4	4600.0	3400.0	2933.3	3133.3
...	...	...	...	...	...	...	...	...	...	...	...	...
77	뚜레쥬르 하남신장	경기도 하남시	경기도	3260.0	3948.6	7100.0	8300.0	5091.7	4600.0	3333.3	2800.0	3180.0
78	뚜레쥬르 향남2지구	경기도 화성시	경기도	3500.0	4292.1	7171.4	8300.0	4727.3	4600.0	3622.2	2400.0	3116.7
79	뚜레쥬르 화성시청	경기도 화성시	경기도	3500.0	4252.6	6775.0	9400.0	4807.1	4600.0	3314.3	2800.0	2850.0
80	뚜레쥬르 화정덕양구청	경기도 고양시	경기도	3022.2	4582.4	7471.4	10500.0	5027.3	4600.0	2700.0	2933.3	3100.0
81	뚜레쥬르 화정중앙	경기도 고양시	경기도	3254.5	4252.5	7400.0	9400.0	5183.3	4600.0	3433.3	3133.3	2800.0

82 rows × 12 columns

In [4]:
```
grouped_data = result.groupby('주소')[['간식빵', '샌드위치류', '샐러드류', '식빵류', '신제품', '크림빵', '파이/패스트리', '피자빵,고로케']].mean().mean(axis=1).sort_values(ascending=False

# groupby 결과를 데이터프레임으로 변환
grouped_df = pd.DataFrame(grouped_data).reset_index()

# 컬럼명 변경
grouped_df.columns = ['주소', '평균가격']

# CSV 파일로 저장
grouped_df.to_csv('anal_gyeonggi/average_allbread_gyeonggi.csv', index=False, encoding='utf-8-sig')
grouped_df
```

Out[4]:

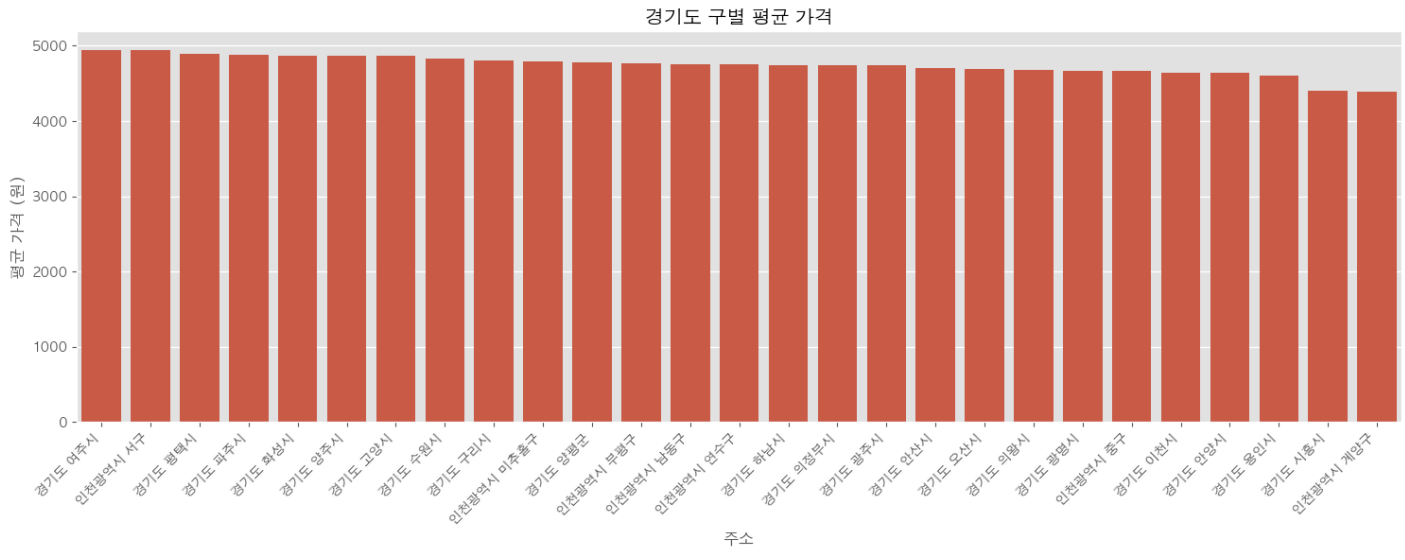
	주소	평균가격
0	경기도 여주시	4942.606250
1	인천광역시 서구	4941.862500
2	경기도 평택시	4897.681250
3	경기도 파주시	4879.025000
4	경기도 화성시	4870.460714
5	경기도 양주시	4863.558333
6	경기도 고양시	4863.261667
7	경기도 수원시	4825.803125
8	경기도 구리시	4808.837500
9	인천광역시 미추홀구	4788.033333
10	경기도 양평군	4780.125000
11	인천광역시 부평구	4766.170833
12	인천광역시 남동구	4759.312500
13	인천광역시 연수구	4756.520833
14	경기도 하남시	4747.012500
15	경기도 의정부시	4741.950000
16	경기도 광주시	4741.306250
17	경기도 안산시	4702.979167
18	경기도 오산시	4696.366667
19	경기도 의왕시	4683.993750
20	경기도 광명시	4668.518750
21	인천광역시 중구	4661.712500
22	경기도 이천시	4647.085714
23	경기도 안양시	4644.743750
24	경기도 용인시	4597.721562
25	경기도 시흥시	4397.733333
26	인천광역시 계양구	4388.271429

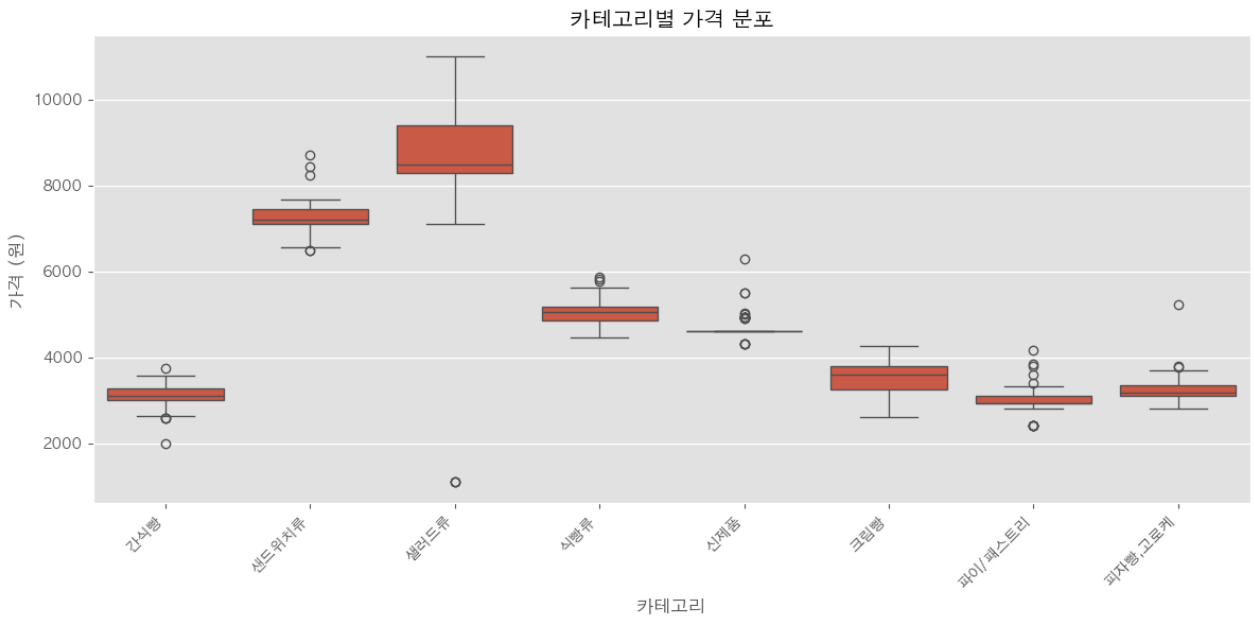
In [5]:
```
categories = ['간식빵', '샌드위치류', '샐러드류', '식빵류', '신제품', '크림빵', '파이/패스트리', '피자빵,고로케']

# 각 카테고리별로 구의 평균 가격 계산
grouped_data = {}
for category in categories:
    grouped_data[category] = result.groupby('주소')[category].mean().round(2)

# 데이터프레임 생성
grouped_df = pd.DataFrame(grouped_data)

# CSV 파일로 저장
grouped_df.to_csv('anal_gyeonggi/average_categorized_gyeonggishop.csv', encoding='utf-8-sig')
grouped_df
```

Out[5]:

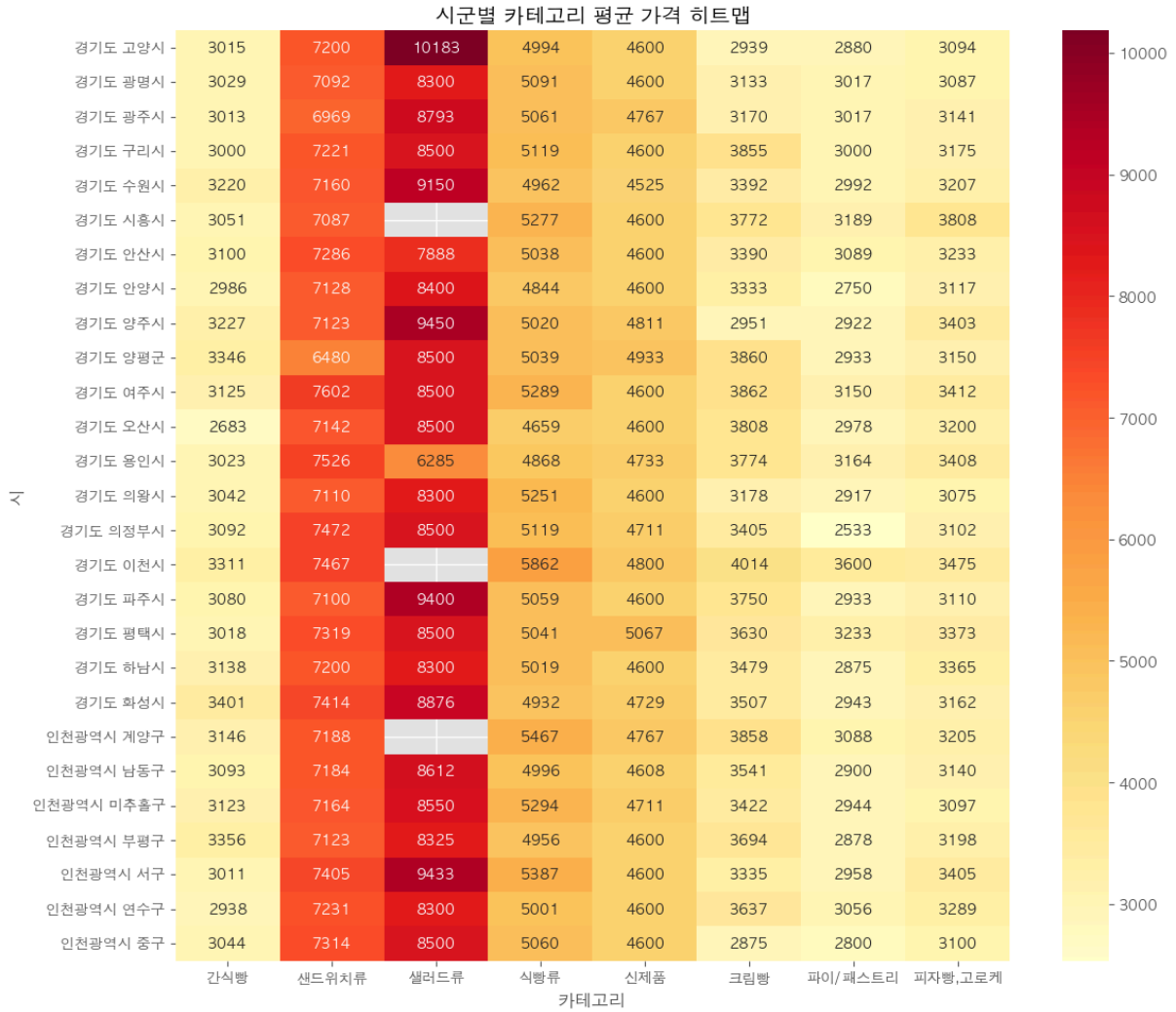
주소	간식빵	샌드위치류	샐러드류	식빵류	신제품	크림빵	파이/패스트리	피자빵,고로케
경기도 고양시	3015.38	7200.40	10183.33	4994.34	4600.00	2938.66	2879.98	3094.00
경기도 광명시	3029.15	7091.65	8300.00	5090.70	4600.00	3133.35	3016.65	3086.65
경기도 광주시	3013.10	6968.65	8792.85	5061.10	4766.65	3170.00	3016.65	3141.45
경기도 구리시	3000.00	7221.45	8500.00	5119.25	4600.00	3855.00	3000.00	3175.00
경기도 수원시	3219.62	7159.72	9150.00	4961.68	4525.00	3392.08	2991.65	3206.68
경기도 시흥시	3051.00	7086.53	NaN	5277.17	4600.00	3772.23	3188.87	3808.33
경기도 안산시	3099.93	7285.57	7887.50	5038.13	4600.00	3390.47	3088.90	3233.33
경기도 안양시	2986.00	7128.35	8400.00	4843.60	4600.00	3333.33	2750.00	3116.67
경기도 양주시	3227.20	7123.23	9450.00	5020.20	4811.10	2951.20	2922.20	3403.33
경기도 양평군	3345.50	6480.00	8500.00	5038.90	4933.30	3860.00	2933.30	3150.00
경기도 여주시	3125.00	7601.55	8500.00	5289.30	4600.00	3862.50	3150.00	3412.50
경기도 오산시	2683.33	7142.50	8500.00	4659.03	4600.00	3808.33	2977.77	3199.97
경기도 용인시	3023.20	7525.83	6285.41	4867.68	4733.33	3773.84	3164.15	3408.33
경기도 의왕시	3041.70	7110.00	8300.00	5251.10	4600.00	3177.50	2916.65	3075.00
경기도 의정부시	3092.03	7472.23	8500.00	5119.40	4711.10	3405.27	2533.33	3102.23
경기도 이천시	3311.10	7466.70	NaN	5862.50	4800.00	4014.30	3600.00	3475.00
경기도 파주시	3080.35	7100.00	9400.00	5058.55	4600.00	3750.00	2933.30	3110.00
경기도 평택시	3018.00	7319.45	8500.00	5040.90	5066.67	3629.77	3233.33	3373.33
경기도 하남시	3138.35	7200.00	8300.00	5018.60	4600.00	3479.15	2875.00	3365.00
경기도 화성시	3401.07	7414.46	8876.19	4931.64	4728.57	3507.00	2942.84	3161.91
인천광역시 계양구	3146.25	7187.50	NaN	5466.65	4766.65	3858.35	3087.50	3205.00
인천광역시 남동구	3092.88	7183.92	8612.50	4996.10	4608.32	3540.80	2899.98	3140.00
인천광역시 미추홀구	3122.70	7163.87	8550.00	5293.63	4711.10	3422.00	2944.43	3096.53
인천광역시 부평구	3355.70	7122.60	8325.00	4955.90	4600.00	3694.07	2877.77	3198.33
인천광역시 서구	3011.10	7405.00	9433.30	5387.35	4600.00	3335.00	2958.35	3404.80
인천광역시 연수구	2938.43	7231.47	8300.00	5001.30	4600.00	3636.53	3055.53	3288.90
인천광역시 중구	3044.40	7314.30	8500.00	5060.00	4600.00	2875.00	2800.00	3100.00

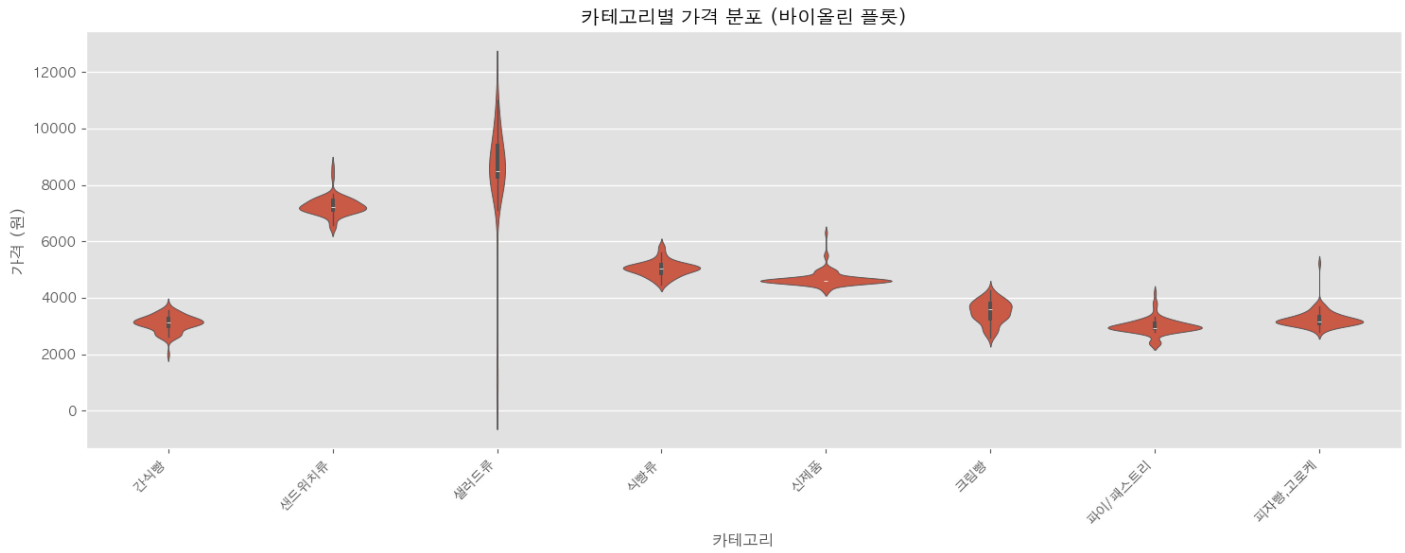
In [6]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from matplotlib import font_manager, rc

# Mac OS 용 폰트 설정
plt.rc('font', family='AppleGothic') # 맥용 폰트 설정

# 그래프 기본 설정
plt.rcParams['axes.unicode_minus'] = False
plt.style.use('ggplot')


# 1. 구별 전체 평균 가격 분석
plt.figure(figsize=(15, 6))
grouped_data = result.groupby('주소')[['간식빵', '샌드위치류', '샐러드류', '식빵류', '신제품', '크림빵', '파이/패스트리', '피자빵,고로케']].mean().mean(axis=1).sort_values(ascending=False

sns.barplot(x=grouped_data.index, y=grouped_data.values)
plt.title('경기도 구별 평균 가격')
plt.xticks(rotation=45, ha='right')
plt.ylabel('평균 가격 (원)')
plt.tight_layout()
plt.show()

# 2. 카테고리별 가격 분포 (박스플롯)
plt.figure(figsize=(12, 6))
categories = ['간식빵', '샌드위치류', '샐러드류', '식빵류', '신제품', '크림빵', '파이/패스트리', '피자빵,고로케']
data_melted = pd.melt(result, value_vars=categories)

sns.boxplot(x='variable', y='value', data=data_melted)
plt.title('카테고리별 가격 분포')
plt.xticks(rotation=45, ha='right')
plt.xlabel('카테고리')
plt.ylabel('가격 (원)')
plt.tight_layout()
plt.show()
```





In [7]:
```python
# 3. 구별/카테고리별 평균 가격 히트맵
plt.figure(figsize=(12, 10))
pivot_data = result.groupby('주소')[categories].mean()
sns.heatmap(pivot_data, annot=True, fmt='.0f', cmap='YlOrRd')
plt.title('시군별 카테고리 평균 가격 히트맵')
plt.ylabel('시')
plt.xlabel('카테고리')
plt.tight_layout()
plt.show()

# 5. 카테고리별 가격 분포 (바이올린 플롯)
plt.figure(figsize=(15, 6))
sns.violinplot(x='variable', y='value', data=data_melted)
plt.title('카테고리별 가격 분포 (바이올린 플롯)')
plt.xticks(rotation=45, ha='right')
plt.xlabel('카테고리')
plt.ylabel('가격 (원)')
plt.tight_layout()
plt.show()
```





In [8]:
```python
# 1. 구별 평균 빵가격 계산
categories = ['간식빵', '샌드위치류', '샐러드류', '식빵류', '신제품', '크림빵', '파이/패스트리', '피자빵,고로케']
bread_price_by_district = result.groupby('주소')[categories].mean().mean(axis=1).reset_index()
bread_price_by_district.columns = ['구분', '평균_빵가격']
# '충청남도' 제거

# bread_price_by_district['구분'] = bread_price_by_district['구분'].str.replace('인천광역시', '인천').str.strip()

# 아파트 가격 데이터 전처리
apt_price = pd.read_csv('anal_gyeonggi/gyeonggi_APT_PRICE.csv')
# '경기도'와 '구' 제거

apt_price['구분'] = apt_price['구분'].str.replace('인천광역시', '인천').str.strip()

apt_price['매매'] = pd.to_numeric(apt_price['매매'].str.replace(',', ''), errors='coerce')
apt_price = apt_price.dropna()  # 결측치 제거

# apt_price = apt_price[~apt_price['구분'].str.contains('인천광역시 계양구')]
# apt_price = apt_price[~apt_price['구분'].str.contains('인천광역시 남동구')]
# apt_price = apt_price[~apt_price['구분'].str.contains('인천광역시 미추홀구')]
# apt_price = apt_price[~apt_price['구분'].str.contains('인천광역시 부평구')]
# apt_price = apt_price[~apt_price['구분'].str.contains('인천광역시 서구')]
# apt_price = apt_price[~apt_price['구분'].str.contains('인천광역시 연수구')]
# apt_price = apt_price[~apt_price['구분'].str.contains('인천광역시 중구')]

# 데이터 확인
print("전처리 후 구별 빵가격 데이터:")
print(bread_price_by_district)
print("\n전처리 후 아파트 가격 데이터:")
print(apt_price)

# 데이터 병합
merged_df = pd.merge(bread_price_by_district, apt_price[['구분', '매매']], on='구분', how='inner')
print("\n병합된 데이터:")
print(merged_df)

# 시각화
if not merged_df.empty:
    plt.figure(figsize=(20, 10))
    sns.scatterplot(data=merged_df, x='매매', y='평균_빵가격')

    # 추세선 추가
    x = merged_df['매매'].values
    y = merged_df['평균_빵가격'].values
    z = np.polyfit(x, y, 1)
    p = np.poly1d(z)
    plt.plot(x, p(x), "r--", alpha=0.8)

    # 각 점에 구 이름 표시
    for idx, row in merged_df.iterrows():
        plt.annotate(row['구분'], (row['매매'], row['평균_빵가격']))

    correlation = merged_df['평균_빵가격'].corr(merged_df['매매'])
    plt.title(f'구별 평균 빵가격과 아파트 매매가의 관계\n(상관계수: {correlation:.3f})')
    plt.xlabel('아파트 평균 매매가 (만원)')
    plt.ylabel('평균 빵가격 (원)')

    print(f"\n상관계수: {correlation:.3f}")
    if correlation > 0:
        print("양의 상관관계가 있습니다: 아파트 가격이 높은 구일수록 빵 가격도 높은 경향이 있습니다.")
    else:
        print("음의 상관관계가 있습니다: 아파트 가격이 높은 구일수록 빵 가격이 낮은 경향이 있습니다.")
```

전처리 후 구별 빵가격 데이터:

	구분	평균_빵가격
0	경기도 고양시	4863.261667
1	경기도 광명시	4668.518750
2	경기도 광주시	4741.306250
3	경기도 구리시	4808.837500
4	경기도 수원시	4825.803125
5	경기도 시흥시	4397.733333
6	경기도 안산시	4702.979167
7	경기도 안양시	4644.743750
8	경기도 양주시	4863.558333
9	경기도 양평군	4780.125000
10	경기도 여주시	4942.606250
11	경기도 오산시	4696.366667
12	경기도 용인시	4597.721562
13	경기도 의왕시	4683.993750
14	경기도 의정부시	4741.950000
15	경기도 이천시	4647.085714
16	경기도 파주시	4879.025000
17	경기도 평택시	4897.681250
18	경기도 하남시	4747.012500
19	경기도 화성시	4870.460714
20	인천광역시 계양구	4388.271429
21	인천광역시 남동구	4759.312500
22	인천광역시 미추홀구	4788.033333
23	인천광역시 부평구	4766.170833
24	인천광역시 서구	4941.862500
25	인천광역시 연수구	4756.520833
26	인천광역시 중구	4661.712500

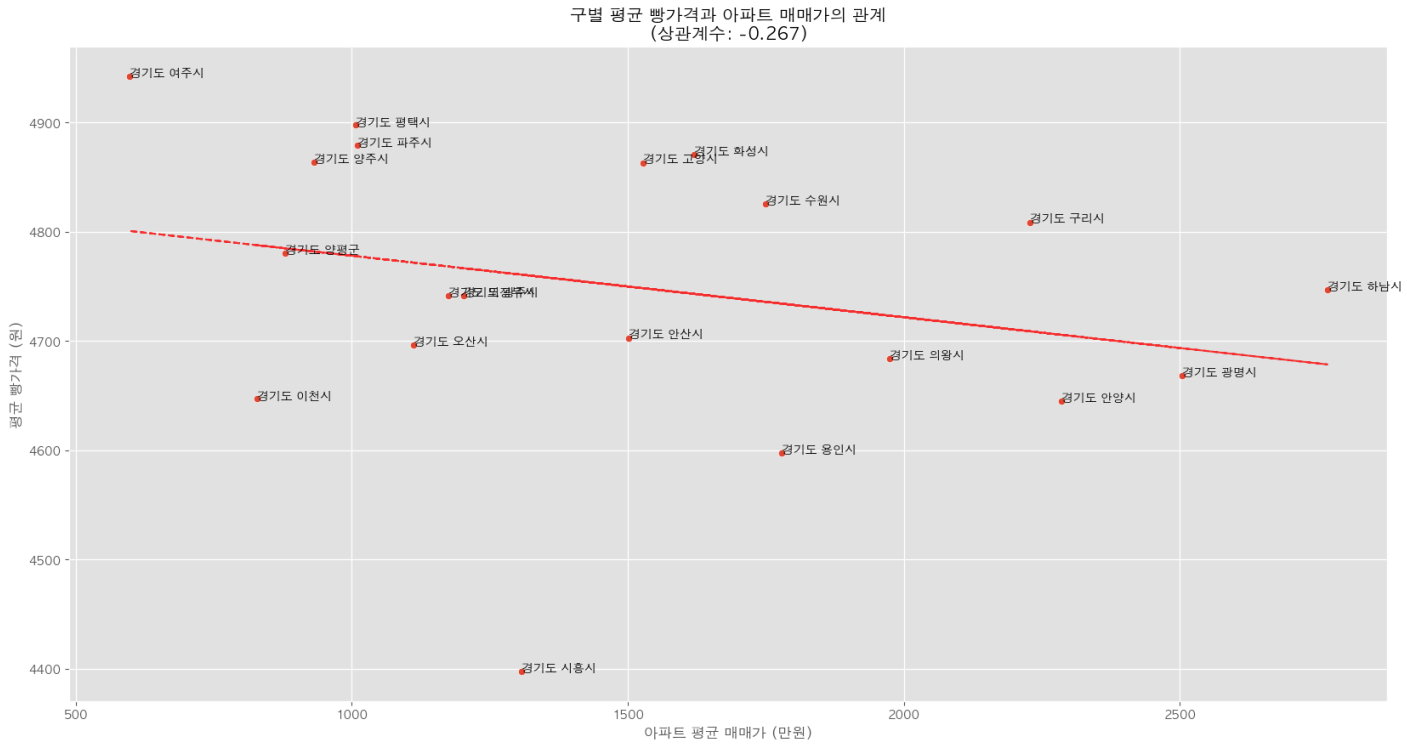
전처리 후 아파트 가격 데이터:

	구분	매매	전세
0	경기도 가평군	684	504
1	경기도 고양시	1527	1,041
2	경기도 과천시	5819	2,349
3	경기도 광명시	2504	1,456
4	경기도 광주시	1203	832
5	경기도 구리시	2229	1,234
6	경기도 군포시	1664	1,067
7	경기도 김포시	1316	889
8	경기도 남양주시	1370	940
9	경기도 동두천시	586	458
10	경기도 부천시	1731	1,141
11	경기도 성남시	3559	1,820
12	경기도 수원시	1749	1,092
13	경기도 시흥시	1308	875
14	경기도 안산시	1501	907
15	경기도 안성시	685	517
16	경기도 안양시	2286	1,349
17	경기도 양주시	932	583
18	경기도 양평군	879	633
19	경기도 여주시	597	486
20	경기도 연천군	553	358
21	경기도 오산시	1112	765
22	경기도 용인시	1779	1,152
23	경기도 의왕시	1974	1,208
24	경기도 의정부시	1175	798
25	경기도 이천시	828	653
26	경기도 파주시	1010	715
27	경기도 평택시	1007	670
28	경기도 포천시	550	400
29	경기도 하남시	2767	1,643
30	경기도 화성시	1620	1,001
31	인천 강화군	643	432
32	인천 계양구	1011	714
33	인천 남동구	1109	774
34	인천 동구	874	630
35	인천 미추홀구	1000	704
36	인천 부평구	1241	851
37	인천 서구	1231	793
38	인천 연수구	1526	903
39	인천 중구	1075	691

병합된 데이터:

	구분	평균_빵가격	매매
0	경기도 고양시	4863.261667	1527
1	경기도 광명시	4668.518750	2504
2	경기도 광주시	4741.306250	1203
3	경기도 구리시	4808.837500	2229
4	경기도 수원시	4825.803125	1749
5	경기도 시흥시	4397.733333	1308
6	경기도 안산시	4702.979167	1501
7	경기도 안양시	4644.743750	2286
8	경기도 양주시	4863.558333	932
9	경기도 양평군	4780.125000	879
10	경기도 여주시	4942.606250	597
11	경기도 오산시	4696.366667	1112
12	경기도 용인시	4597.721562	1779
13	경기도 의왕시	4683.993750	1974
14	경기도 의정부시	4741.950000	1175
15	경기도 이천시	4647.085714	828
16	경기도 파주시	4879.025000	1010
17	경기도 평택시	4897.681250	1007
18	경기도 하남시	4747.012500	2767
19	경기도 화성시	4870.460714	1620

상관계수: -0.267
음의 상관관계가 있습니다: 아파트 가격이 높은 구일수록 빵 가격이 낮은 경향이 있습니다.

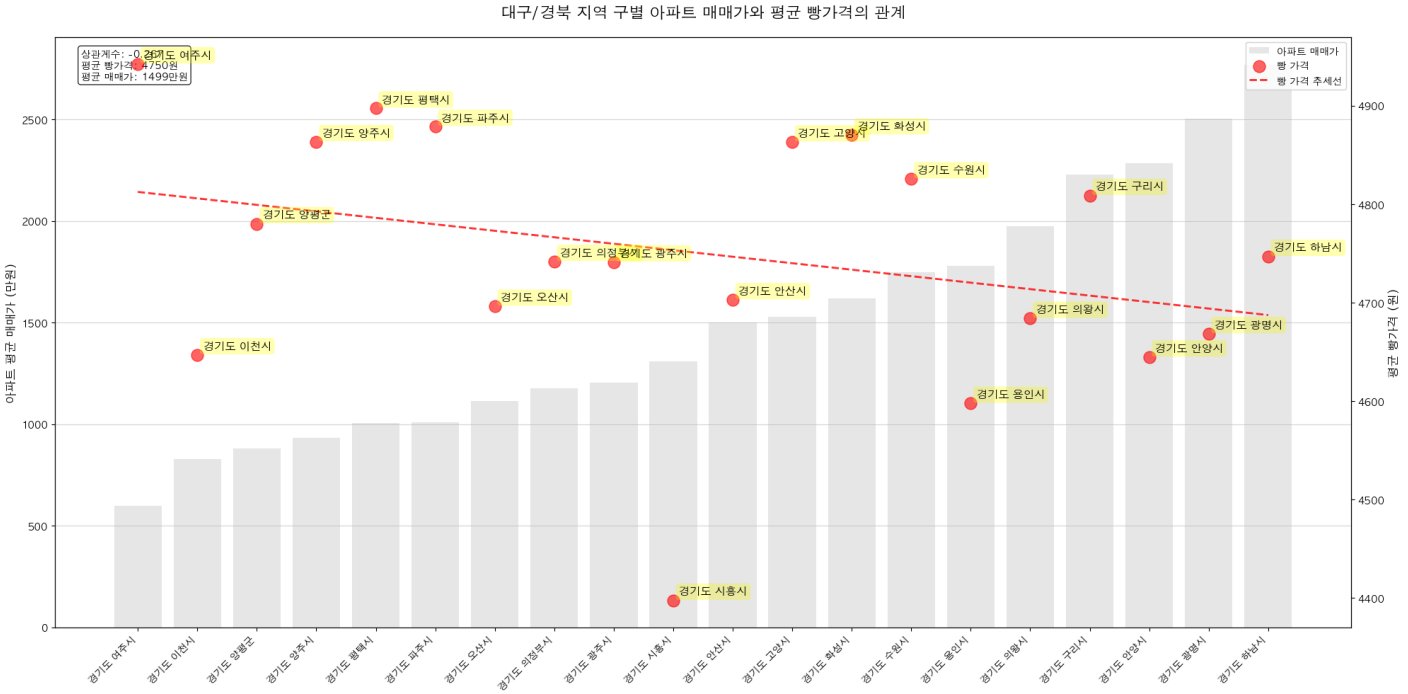


In [9]:
```
pip install adjustText
```

```
Requirement already satisfied: adjustText in /opt/anaconda3/lib/python3.12/site-packages (1.3.0)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.12/site-packages (from adjustText) (1.26.4)
Requirement already satisfied: matplotlib in /opt/anaconda3/lib/python3.12/site-packages (from adjustText) (3.8.4)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.12/site-packages (from adjustText) (1.13.1)
Requirement already satisfied: contourpy>=1.0.1 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->adjustText) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->adjustText) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->adjustText) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->adjustText) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->adjustText) (23.2)
Requirement already satisfied: pillow>=8 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->adjustText) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->adjustText) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /opt/anaconda3/lib/python3.12/site-packages (from matplotlib->adjustText) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.12/site-packages (from python-dateutil>=2.7->matplotlib->adjustText) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 시각화 기본 설정
plt.style.use('default')  # 기본 스타일 사용
plt.rcParams['figure.figsize'] = (20, 10)
plt.rcParams['font.family'] = 'AppleGothic'

if not merged_df.empty:
    # 데이터 정렬
    sorted_df = merged_df.sort_values(by='매매')

    # 메인 그래프 생성
    fig, ax1 = plt.subplots(figsize=(20, 10))

    # 기본 그리드 제거
    ax1.grid(False)

    # 바차트 배경 (매매가)
    ax1.bar(range(len(sorted_df)), sorted_df['매매'], alpha=0.5, color='lightgray', label='아파트 매매가')

    # 산점도를 위한 두 번째 y축 생성
    ax2 = ax1.twinx()
    ax2.grid(False)

    # 산점도 그리기
    scatter = ax2.scatter(range(len(sorted_df)), sorted_df['평균_빵가격'],
                          s=150, alpha=0.6, color='red', label='빵 가격')

    # 추세선 추가
    z = np.polyfit(range(len(sorted_df)), sorted_df['평균_빵가격'], 1)
    p = np.poly1d(z)
    ax2.plot(range(len(sorted_df)), p(range(len(sorted_df))), "r--",
             linewidth=2, alpha=0.8, label='빵 가격 추세선')

    # x축 레이블 설정 (45도 회전)
    ax1.set_xticks(range(len(sorted_df)))
    ax1.set_xticklabels(sorted_df['구분'], rotation=45, ha='right')

    # 각 점에 구 이름 표시
    for i, row in enumerate(sorted_df.itertuples()):
        ax2.annotate(row.구분,
                     (i, row.평균_빵가격),
                     xytext=(6, 6),
                     textcoords='offset points',
                     fontsize=11,
                     bbox=dict(boxstyle='round,pad=0.3', fc='yellow', alpha=0.3, ec='none'))
                     # arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=0'))

    # 평균선 추가
    # ax2.axhline(y=sorted_df['평균_빵가격'].mean(), color='g', linestyle='--', alpha=0.3, label='평균 빵가격')
    # ax1.axhline(y=sorted_df['매매'].mean(), color='b', linestyle='--', alpha=0.3, label='평균 매매가')

    # y축 그리드만 추가
    ax1.grid(True, axis='y', alpha=0.3, linestyle='-', color='gray')

    # 축 레이블 설정
    ax1.set_ylabel('아파트 평균 매매가 (만원)', fontsize=12)
    ax2.set_ylabel('평균 빵가격 (원)', fontsize=12)

    # 상관계수 계산
    correlation = sorted_df['평균_빵가격'].corr(sorted_df['매매'])
```

```
    # 그래프 제목 설정
    plt.title('대구/경북 지역 구별 아파트 매매가와 평균 빵가격의 관계', fontsize=16, pad=20)

    # 통계 정보 추가
    stats_text = f'상관계수: {correlation:.3f}\n'
    stats_text += f'평균 빵가격: {sorted_df["평균_빵가격"].mean():.0f}원\n'
    stats_text += f'평균 매매가: {sorted_df["매매"].mean():.0f}만원'
    ax1.text(0.02, 0.98, stats_text,
             transform=ax1.transAxes,
             verticalalignment='top',
             bbox=dict(boxstyle='round', facecolor='white', alpha=0.8))

    # 범례 추가
    lines1, labels1 = ax1.get_legend_handles_labels()
    lines2, labels2 = ax2.get_legend_handles_labels()
    ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper right')

    # 여백 조정
    plt.tight_layout()

    # 그래프 표시
    plt.show()

    # 추가 분석 출력
    print("\n=== 지역별 상세 데이터 ===")
    analysis_df = merged_df.copy()
    analysis_df['가격_차이_순위'] = analysis_df['매매'].rank(ascending=False)
    analysis_df['빵가격_순위'] = analysis_df['평균_빵가격'].rank(ascending=False)
    analysis_df['순위_차이'] = abs(analysis_df['가격_차이_순위'] - analysis_df['빵가격_순위'])

    print("\n아파트 가격 상위 5개 지역:")
    print(analysis_df.nlargest(5, '매매')[['구분', '매매', '평균_빵가격']])
    print("\n빵 가격 상위 5개 지역:")
    print(analysis_df.nlargest(5, '평균_빵가격')[['구분', '매매', '평균_빵가격']])
    print("\n순위 차이가 가장 큰 5개 지역 (불일치도가 높은 지역):")
    print(analysis_df.nlargest(5, '순위_차이')[['구분', '매매', '평균_빵가격', '순위_차이']])

    print(f"\n상관계수: {correlation:.3f}")
    if correlation > 0:
        print("양의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격도 높은 경향이 있습니다.")
    else:
        print("음의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격이 낮은 경향이 있습니다.")
```

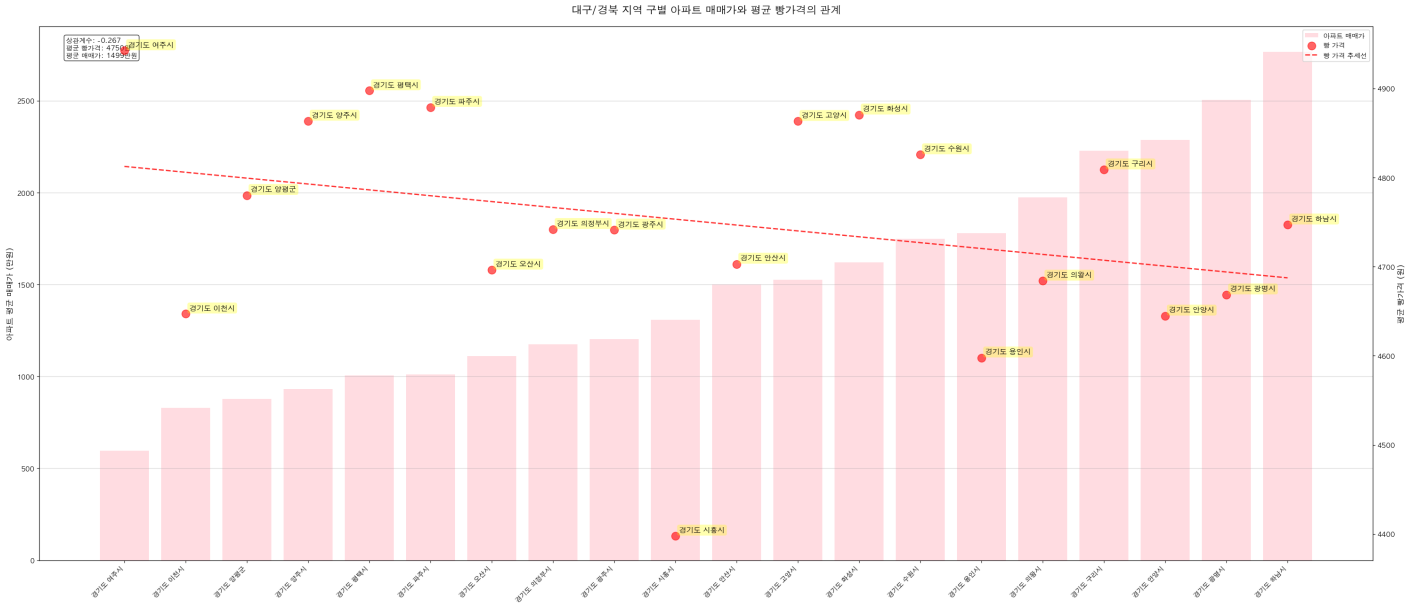



```
=== 지역별 상세 데이터 ===

아파트 가격 상위 5개 지역:
    구분    매매       평균_빵가격
18  경기도 하남시  2767  4747.01250
1   경기도 광명시  2504  4668.51875
7   경기도 안양시  2286  4644.74375
3   경기도 구리시  2229  4808.83750
13  경기도 의왕시  1974  4683.99375

빵 가격 상위 5개 지역:
    구분    매매       평균_빵가격
10  경기도 여주시   597  4942.606250
17  경기도 평택시  1007  4897.681250
16  경기도 파주시  1010  4879.025000
19  경기도 화성시  1620  4870.460714
8   경기도 양주시   932  4863.558333

순위 차이가 가장 큰 5개 지역 (불일치도가 높은 지역):
    구분    매매       평균_빵가격  순위_차이
10  경기도 여주시   597  4942.606250   19.0
7   경기도 안양시  2286  4644.743750   15.0
1   경기도 광명시  2504  4668.518750   14.0
17  경기도 평택시  1007  4897.681250   14.0
12  경기도 용인시  1779  4597.721562   13.0

상관계수: -0.267
음의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격이 낮은 경향이 있습니다.
```

In [11]:

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# 시각화 기본 설정
plt.style.use('default')  # 기본 스타일 사용
plt.rcParams['figure.figsize'] = (23, 13)
plt.rcParams['font.family'] = 'AppleGothic'

if not merged_df.empty:
    # 데이터 정렬
```

```python
sorted_df = merged_df.sort_values(by='매매')

# 메인 그래프 생성
fig, ax1 = plt.subplots(figsize=(30, 13))

# 기본 그리드 제거
ax1.grid(False)

# 바차트 배경 (매매가)
ax1.bar(range(len(sorted_df)), sorted_df['매매'], alpha=0.5, color='pink', label='아파트 매매가')

# 산점도를 위한 두 번째 y축 생성
ax2 = ax1.twinx()
ax2.grid(False)

# 산점도 그리기
scatter = ax2.scatter(range(len(sorted_df)), sorted_df['평균_빵가격'],
                      s=150, alpha=0.6, color='red', label='빵 가격')

# 추세선 추가
z = np.polyfit(range(len(sorted_df)), sorted_df['평균_빵가격'], 1)
p = np.poly1d(z)
ax2.plot(range(len(sorted_df)), p(range(len(sorted_df))), "r--",
         linewidth=2, alpha=0.8, label='빵 가격 추세선')

# x축 레이블 설정 (45도 회전)
ax1.set_xticks(range(len(sorted_df)))
ax1.set_xticklabels(sorted_df['구분'], rotation=45, ha='right')

# 각 점에 구 이름 표시
for i, row in enumerate(sorted_df.itertuples()):
    ax2.annotate(row.구분,
                 (i, row.평균_빵가격),
                 xytext=(6, 6),
                 textcoords='offset points',
                 fontsize=11,
                 bbox=dict(boxstyle='round,pad=0.3', fc='yellow', alpha=0.3, ec='none'))
                 # arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=0'))

# 평균선 추가
# ax2.axhline(y=sorted_df['평균_빵가격'].mean(), color='g', linestyle='--', alpha=0.3, label='평균 빵가격')
# ax1.axhline(y=sorted_df['매매'].mean(), color='b', linestyle='--', alpha=0.3, label='평균 매매가')

# y축 그리드만 추가
ax1.grid(True, axis='y', alpha=0.3, linestyle='-', color='gray')

# 축 레이블 설정
ax1.set_ylabel('아파트 평균 매매가 (만원)', fontsize=12)
ax2.set_ylabel('평균 빵가격 (원)', fontsize=12)

# 상관계수 계산
correlation = sorted_df['평균_빵가격'].corr(sorted_df['매매'])

# 그래프 제목 설정
plt.title('대구/경북 지역 구별 아파트 매매가와 평균 빵가격의 관계', fontsize=16, pad=20)

# 통계 정보 추가
stats_text = f'상관계수: {correlation:.3f}\n'
stats_text += f'평균 빵가격: {sorted_df["평균_빵가격"].mean():.0f}원\n'
stats_text += f'평균 매매가: {sorted_df["매매"].mean():.0f}만원'
ax1.text(0.02, 0.98, stats_text,
         transform=ax1.transAxes,
         verticalalignment='top',
         bbox=dict(boxstyle='round', facecolor='white', alpha=0.8))

# 범례 추가
lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper right')

# 여백 조정
plt.tight_layout()

# 그래프 표시
plt.show()

# 추가 분석 출력
print("\n=== 지역별 상세 데이터 ===")
analysis_df = merged_df.copy()
analysis_df['가격_차이_순위'] = analysis_df['매매'].rank(ascending=False)
analysis_df['빵가격_순위'] = analysis_df['평균_빵가격'].rank(ascending=False)
analysis_df['순위_차이'] = abs(analysis_df['가격_차이_순위'] - analysis_df['빵가격_순위'])

print("\n아파트 가격 상위 5개 지역:")
print(analysis_df.nlargest(5, '매매')[['구분', '매매', '평균_빵가격']])
print("\n빵 가격 상위 5개 지역:")
print(analysis_df.nlargest(5, '평균_빵가격')[['구분', '매매', '평균_빵가격']])
print("\n순위 차이가 가장 큰 5개 지역 (불일치도가 높은 지역):")
print(analysis_df.nlargest(5, '순위_차이')[['구분', '매매', '평균_빵가격', '순위_차이']])

print(f"\n상관계수: {correlation:.3f}")
if correlation > 0:
    print("양의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격도 높은 경향이 있습니다.")
else:
    print("음의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격이 낮은 경향이 있습니다.")
```



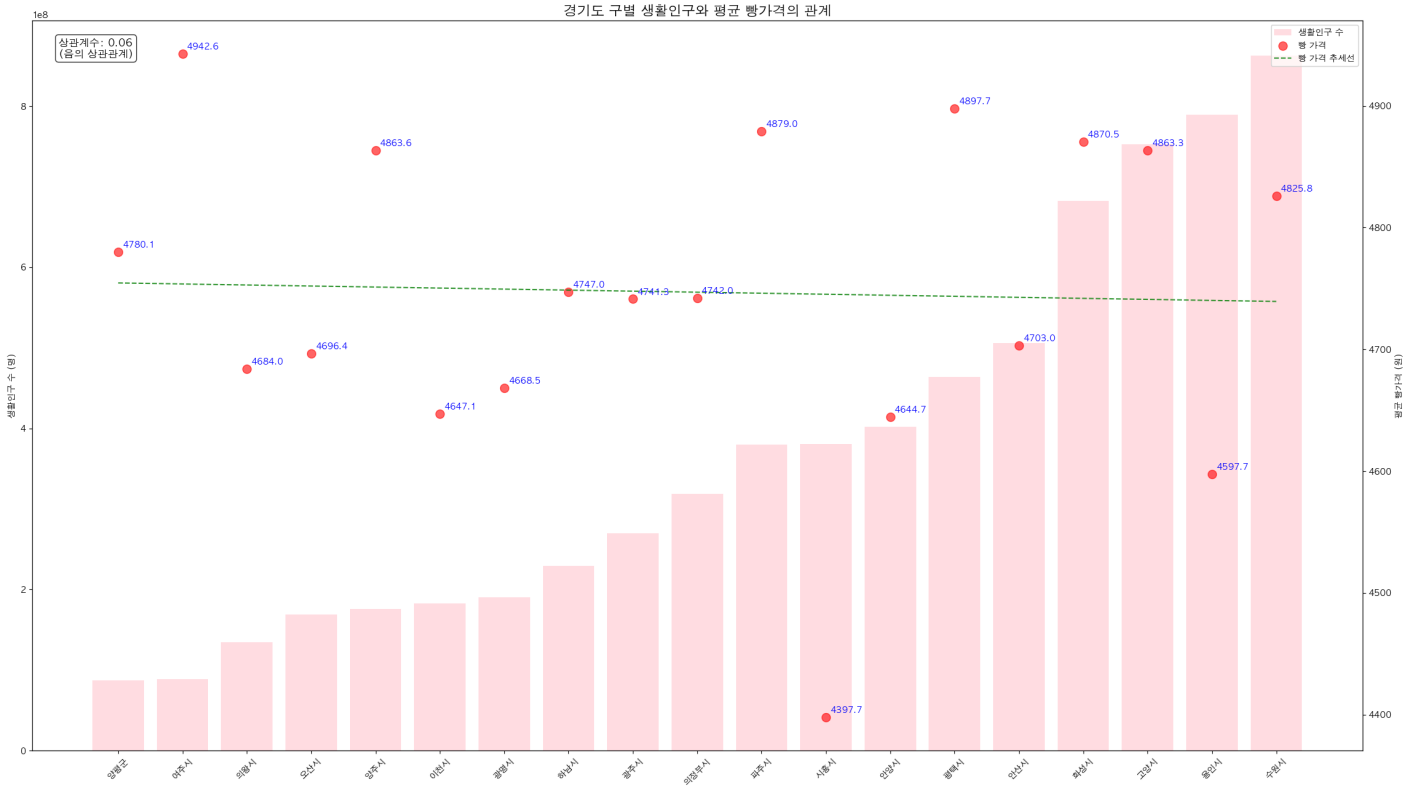

```
=== 지역별 상세 데이터 ===

아파트 가격 상위 5개 지역:
          구분    매매       평균_빵가격
18  경기도 하남시  2767  4747.01250
1   경기도 광명시  2504  4668.51875
7   경기도 안양시  2286  4644.74375
3   경기도 구리시  2229  4808.83750
13  경기도 의왕시  1974  4683.99375

빵 가격 상위 5개 지역:
          구분    매매       평균_빵가격
10  경기도 여주시   597  4942.606250
17  경기도 평택시  1007  4897.681250
16  경기도 파주시  1010  4879.025000
19  경기도 화성시  1620  4870.460714
8   경기도 양주시   932  4863.558333

순위 차이가 가장 큰 5개 지역 (불일치도가 높은 지역):
          구분    매매       평균_빵가격  순위_차이
10  경기도 여주시   597  4942.606250   19.0
7   경기도 안양시  2286  4644.743750   15.0
1   경기도 광명시  2504  4668.518750   14.0
17  경기도 평택시  1007  4897.681250   14.0
12  경기도 용인시  1779  4597.721562   13.0

상관계수: -0.267
음의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격이 낮은 경향이 있습니다.
```

In [12]:

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# 데이터 로드
people_df = pd.read_csv('./data_people/processed_gyeonggi_living_population.csv')
people_df['생활인구 수'] = people_df['생활인구 수'].astype(float)
people_df = people_df.rename(columns={'지역': '지역'})

# 'merged_df'에서 '구분' 처리 및 '지역'으로 열 이름 변경
merged_df['구분'] = merged_df['구분'].str.replace('경기도 ', '').str.strip()
merged_df = merged_df.rename(columns={'구분': '지역'})

# 병합
merged_df = pd.merge(
    merged_df,
    people_df,
    on='지역',
    how='inner',
    indicator=True
)

if merged_df.empty:
    raise ValueError("병합된 데이터프레임이 비어 있습니다.")
merged_df = merged_df.drop(columns=['_merge'])

# 데이터 정렬
sorted_df = merged_df.sort_values(by='생활인구 수', ascending=True).dropna()

# 시각화
plt.figure(figsize=(20, 10))
fig, ax1 = plt.subplots()

# 바차트: 생활인구
bars = ax1.bar(range(len(sorted_df)), sorted_df['생활인구 수'], color='pink', alpha=0.5, label='생활인구 수')

# 산점도: 빵 가격
ax2 = ax1.twinx()
scatter = ax2.scatter(range(len(sorted_df)), sorted_df['평균_빵가격'], color='red', s=100, alpha=0.6, label='빵 가격')

# 데이터 레이블 추가
for i, row in enumerate(sorted_df.itertuples()):
    ax2.annotate(f"{row.평균_빵가격:.1f}",
                 (i, row.평균_빵가격),
                 xytext=(5, 5),
                 textcoords='offset points',
                 fontsize=10, color='blue')

# 추세선 추가 (빵 가격)
if len(sorted_df) > 1:
    z = np.polyfit(range(len(sorted_df)), sorted_df['평균_빵가격'], 1)
    p = np.poly1d(z)
    ax2.plot(range(len(sorted_df)), p(range(len(sorted_df))), "g--", alpha=0.8, label='빵 가격 추세선')

# 상관계수 계산 및 그래프에 표시
correlation = sorted_df['평균_빵가격'].corr(sorted_df['생활인구 수'])
ax1.text(0.02, 0.95, f"상관계수: {correlation:.2f}\n(음의 상관관계)",
         transform=ax1.transAxes,
         fontsize=12,
         bbox=dict(boxstyle="round", facecolor="white", alpha=0.8))
```

```
# x축 설정
ax1.set_xticks(range(len(sorted_df)))
ax1.set_xticklabels(sorted_df['지역'], rotation=45)
ax1.set_ylabel('생활인구 수 (명)')
ax2.set_ylabel('평균 빵가격 (원)')

# 제목 설정
plt.title('경기도 구별 생활인구와 평균 빵가격의 관계', fontsize=16)
plt.tight_layout()

# 범례 추가
lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper right')

# 그래프 표시
plt.show()
```

```
<Figure size 2000x1000 with 0 Axes>
```



In [13]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# 데이터 로드
people_df = pd.read_csv('./data_people/processed_gyeonggi_living_population.csv')
people_df['생활인구 수'] = people_df['생활인구 수'].astype(float)  # '생활인구 수'가 이미 숫자로 저장되어 있다면 생략 가능
people_df = people_df.rename(columns={'지역': '지역'})  # '지역' 열 이름 확인 및 정리

# 'merged_df'에서 '구분' 처리 및 '지역'으로 열 이름 변경
merged_df['구분'] = merged_df['구분'].str.replace('경기도 ', '').str.strip()
merged_df = merged_df.rename(columns={'구분': '지역'})

# 병합
merged_df = pd.merge(
    merged_df,
    people_df,
    on='지역',  # 병합 키로 '지역' 열 사용
    how='inner',  # 교집합 병합
    indicator=True
)

# 병합 결과 확인
if merged_df.empty:
    raise ValueError("병합된 데이터프레임이 비어 있습니다. 데이터가 일치하지 않습니다.")
merged_df = merged_df.drop(columns=['_merge'])  # 병합 상태 열 삭제

# 데이터 정렬
sorted_df = merged_df.sort_values(by='생활인구 수', ascending=False).dropna()

# 상위 10개 데이터 확인
print("\n=== 생활인구 수 상위 10개 지역 ===")
print(sorted_df[['지역', '생활인구 수', '평균_빵가격']].head(10))  # 상위 10개 데이터 출력

# 상위 몇 개 데이터 확인
sorted_df = merged_df.sort_values(by='평균_빵가격', ascending=False).dropna()

print("\n=== 평균_빵가격 상위 10개  ===")
print(sorted_df[['지역', '생활인구 수', '평균_빵가격']].head(10))  # 상위 10개 데이터를 확인

# 시각화
plt.figure(figsize=(15, 10))
fig, ax1 = plt.subplots()

# 바차트: 생활인구
bars = ax1.bar(range(len(sorted_df)), sorted_df['생활인구 수'], color='pink', alpha=0.5, label='생활인구 수')

# 산점도: 빵 가격
ax2 = ax1.twinx()
scatter = ax2.scatter(range(len(sorted_df)), sorted_df['평균_빵가격'], color='red', s=100, alpha=0.6, label='빵 가격')

# 추세선 추가 (빵 가격)
if len(sorted_df) > 1:
    z = np.polyfit(range(len(sorted_df)), sorted_df['평균_빵가격'], 1)
    p = np.poly1d(z)
    ax2.plot(range(len(sorted_df)), p(range(len(sorted_df))), "g--", alpha=0.8, label='빵 가격 추세선')

# x축 설정
```

```
ax1.set_xticks(range(len(sorted_df)))
ax1.set_xticklabels(sorted_df['지역'], rotation=45)
ax1.set_ylabel('생활인구 수 (명)')
ax2.set_ylabel('평균 빵가격 (원)')

# 제목 설정
plt.title('경기도 구별 생활인구와 평균 빵가격의 관계', fontsize=16)

# 그래프 표시
plt.tight_layout()
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File /opt/anaconda3/lib/python3.12/site-packages/pandas/core/indexes/base.py:3805, in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: '구분'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[13], line 11
      8 people_df = people_df.rename(columns={'지역': '지역'})  # '지역' 열 이름 확인 및 정리
     10 # 'merged_df'에서 '구분' 처리 및 '지역'으로 열 이름 변경
---> 11 merged_df['구분'] = merged_df['구분'].str.replace('경기도 ', '').str.strip()
     12 merged_df = merged_df.rename(columns={'구분': '지역'})
     14 # 병합

File /opt/anaconda3/lib/python3.12/site-packages/pandas/core/frame.py:4102, in DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File /opt/anaconda3/lib/python3.12/site-packages/pandas/core/indexes/base.py:3812, in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: '구분'
```

In [ ]: