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

Out[1]:

	뚜레쥬르 지점	뚜레쥬르 강릉교동	뚜레쥬르 강릉입암	뚜레쥬르 동해	뚜레쥬르 원주봉화산	뚜레쥬르 원주단구	뚜레쥬르 원주중앙	뚜레쥬르 원주봉산	뚜레쥬르 춘천퇴계중앙	뚜레쥬르 춘천후평	뚜레쥬르 홍천연봉
0	마늘 단짝 고구마	4900.0	4900.0	NaN	4900	4900.0	4900.0	4900.0	4900.0	4900.0	4900.0
1	깊은 밤 뱅스위스	4300.0	4300.0	NaN	4300	4300.0	4300.0	4300.0	4300.0	4300.0	4300.0
2	BELT 샌드위치	NaN	NaN	NaN	6900	6900.0	6900.0	6900.0	NaN	NaN	6900.0
3	BLT콥 샐러드	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	쉬림프 에그 샐러드	NaN	NaN	NaN	NaN	10500.0	NaN	NaN	NaN	NaN	NaN

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 간석백산', '뚜레쥬르 인하대역', '뚜레쥬르 제물포역',
       '뚜레쥬르 간석역', '뚜레쥬르 인천동수역', '뚜레쥬르 부평청천', '뚜레쥬르 부평신일해피트리', '뚜레쥬르 카페석남',
       '뚜레쥬르 나은병원', '뚜레쥬르 송도더테라스', '뚜레쥬르 송도그린워크', '뚜레쥬르 연수무지개', '인천 영종운서점점',
       '뚜레쥬르 수원천천푸르지오', '뚜레쥬르 매교솔밭사거리', '뚜레쥬르 수원역로데오', '뚜레쥬르 매교힐스테이트',
       '뚜레쥬르 배곧한라비발디', '뚜레쥬르 카페시흥정왕', '뚜레쥬르 시화센종', '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 오산벌원', '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 [2]:
```
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 엘그레이 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','꾀배기','옛날 단팥 도넛',r'^단팥빵$','단팥소보로빵'],
        '신제품': ['마구마구', '단짝', '뱅스위스']
    }

    # 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/gangwon-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'강원특별자치도\s+\w+시', address) or
            re.match(r'강원특별자치도\s+\w+군', address) or
            re.match(r'강원도\s+\w+구', address) or
            re.match(r'강원도\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/gangwon-pricedata.csv',
                                 './adress_process/gangwon_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 강릉교동	강원특별자치도 강릉시	강원도	3340.0	4346.9	7216.7	8300.0	5080.0	4600.0	4125.0	2900.0	3100.0
1	뚜레쥬르 강릉입암	강원특별자치도 강릉시	강원도	2866.7	4455.2	7100.0	NaN	5344.4	4600.0	3220.0	2400.0	3025.0
2	뚜레쥬르 동해	강원특별자치도 동해시	강원도	2657.1	4231.6	NaN	NaN	5344.4	NaN	3220.0	2933.3	3025.0
3	뚜레쥬르 원주단구	강원특별자치도 원주시	강원도	3000.0	4250.0	7120.0	10500.0	5150.0	4600.0	3250.0	3100.0	3116.7
4	뚜레쥬르 원주봉산	강원특별자치도 원주시	강원도	3183.3	4213.8	7177.8	NaN	5171.4	4600.0	2466.7	2933.3	2866.7
5	뚜레쥬르 원주봉화산	강원특별자치도 원주시	강원도	3000.0	4277.8	7128.6	NaN	5141.7	4600.0	3683.3	2400.0	3120.0
6	뚜레쥬르 원주중앙	강원특별자치도 원주시	강원도	2980.0	4597.4	7185.7	NaN	5250.0	4600.0	3220.0	2933.3	3200.0
7	뚜레쥬르 춘천퇴계중앙	강원특별자치도 춘천시	강원도	2800.0	4128.1	6000.0	NaN	5120.0	4600.0	3650.0	3300.0	3233.3
8	뚜레쥬르 춘천후평	강원특별자치도 춘천시	강원도	3040.0	4180.5	7250.0	8300.0	4838.5	4600.0	3100.0	2800.0	3083.3
9	뚜레쥬르 홍천연봉	강원특별자치도 홍천군	강원도	3214.3	4170.6	6950.0	NaN	4955.6	4600.0	2983.3	3100.0	3200.0

In [3]:
```
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_gangwon/average_allbread_gangwon.csv', index=False, encoding='utf-8-sig')
grouped_df
```

Out[3]:

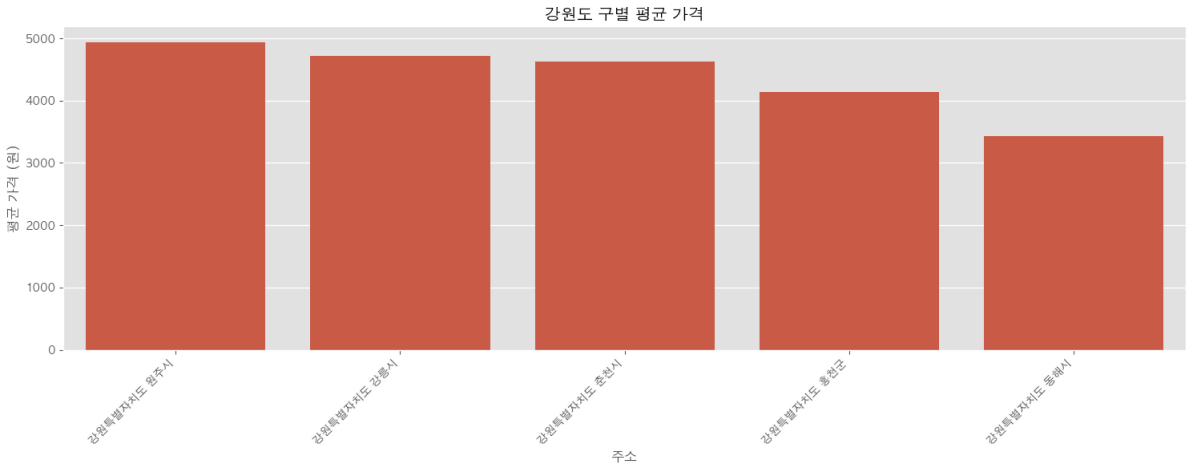
	주소	평균가격
0	강원특별자치도 원주시	4943.078125
1	강원특별자치도 강릉시	4719.862500
2	강원특별자치도 춘천시	4625.943750
3	강원특별자치도 홍천군	4143.314286
4	강원특별자치도 동해시	3435.960000

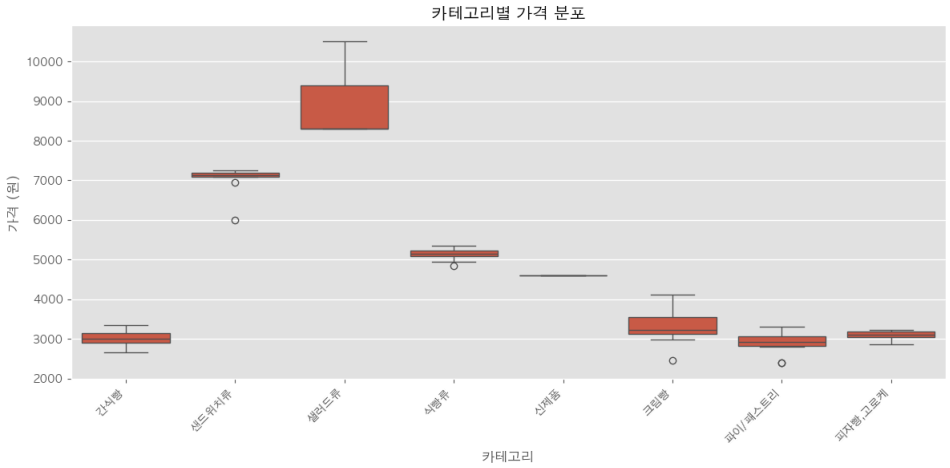
In [4]:
```
categories = ['간식빵', '샌드위치류', '샐러드류', '식빵류', '신제품', '크림빵', '파이/패스트리', '피자빵,고로케']

# 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_gangwon/average_categorized_gangwonshop.csv', encoding='utf-8-sig')
grouped_df
```

Out[4]:

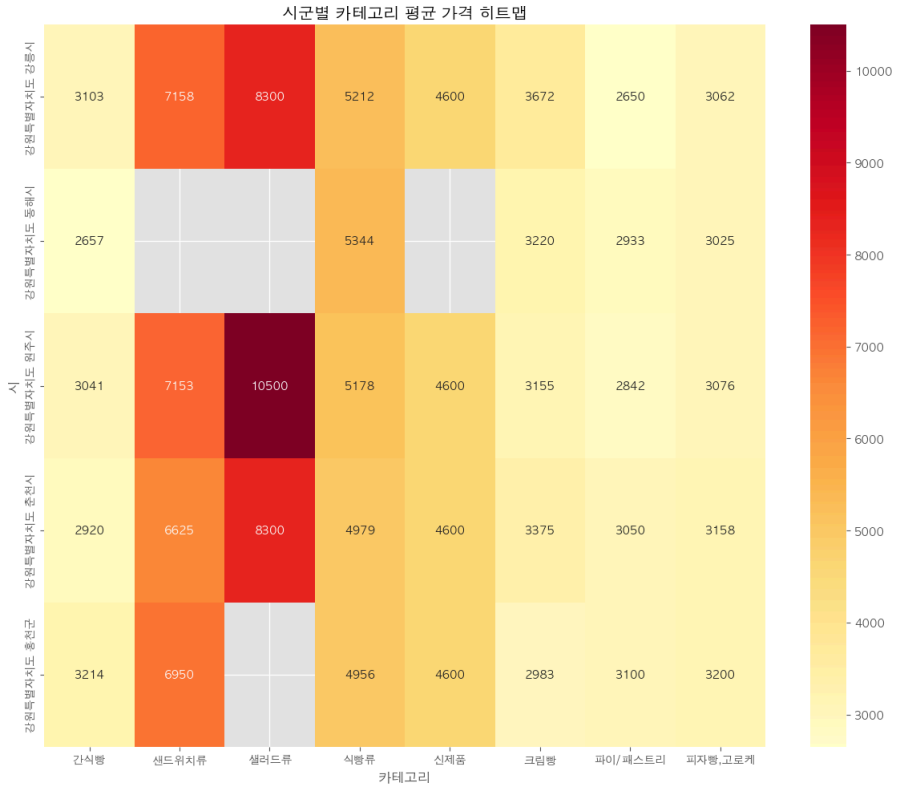
주소	간식빵	샌드위치류	샐러드류	식빵류	신제품	크림빵	파이/패스트리	피자빵,고로케
강원특별자치도 강릉시	3103.35	7158.35	8300.0	5212.20	4600.0	3672.5	2650.00	3062.50
강원특별자치도 동해시	2657.10	NaN	NaN	5344.40	NaN	3220.0	2933.30	3025.00
강원특별자치도 원주시	3040.82	7153.02	10500.0	5178.27	4600.0	3155.0	2841.65	3075.85
강원특별자치도 춘천시	2920.00	6625.00	8300.0	4979.25	4600.0	3375.0	3050.00	3158.30
강원특별자치도 홍천군	3214.30	6950.00	NaN	4955.60	4600.0	2983.3	3100.00	3200.00

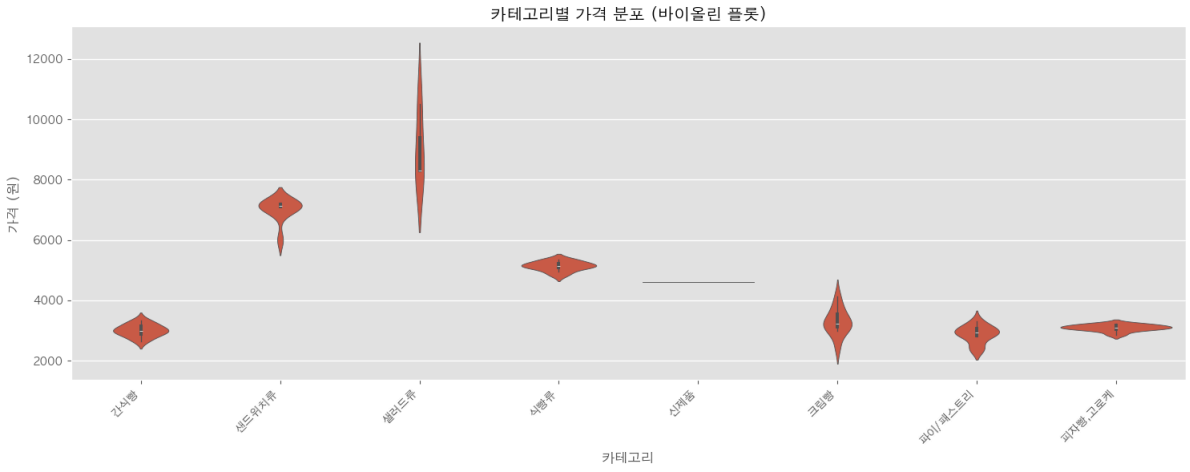
In [5]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from matplotlib import font_manager, rc

# 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('강원도 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()
```





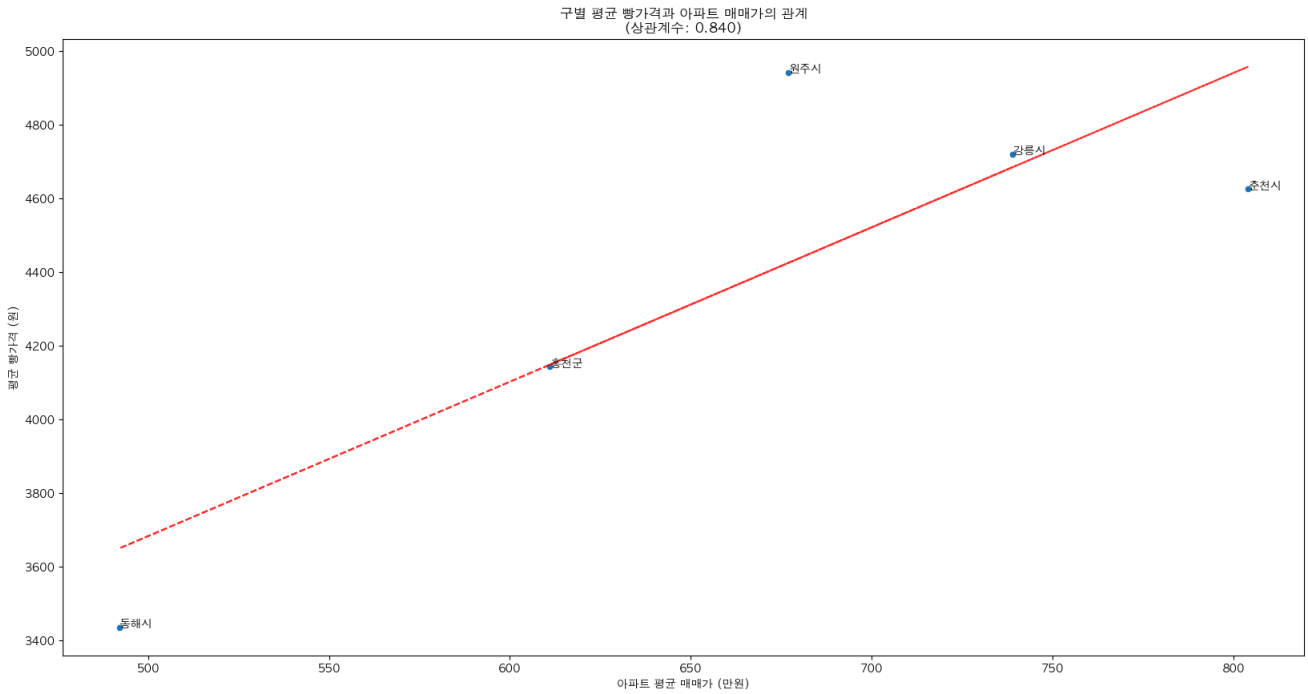
In [6]:
```
# 3. 구별/카테고리별 평균 가격 히트맵
plt.figure(figsize=(12, 10))
pivot_data = result.groupby('주소')[categories].mean()
sns.heatmap(pivot_data, annot=True, fmt='.0f', cmap='YlOrRd')
plt.title('시군별 카테고리 평균 가격 히트맵')
plt.ylabel('시')
plt.xlabel('카테고리')
plt.tight_layout()
plt.show()

# 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()
```





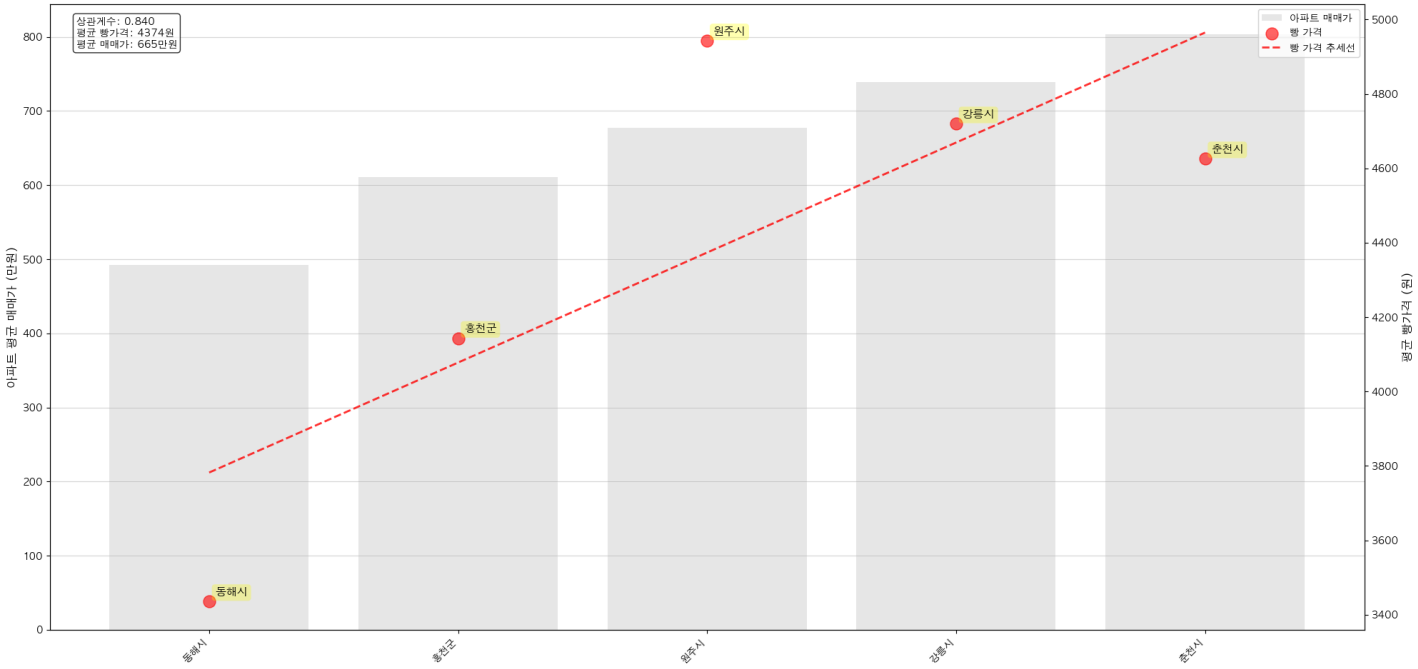
In [14]:
```python
# 1. 구별 평균 빵가격 계산
categories = ['간식빵', '샌드위치류', '샐러드류', '식빵류', '신제품', '크림빵', '파이/패스트리', '피자빵,고로케']
bread_price_by_district = result.groupby('주소')[categories].mean().mean(axis=1).reset_index()
bread_price_by_district.columns = ['구분', '평균_빵가격']
# '충청남도'제거

bread_price_by_district['구분'] = bread_price_by_district['구분'].str.replace('강원특별자치도', '').str.strip()

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

apt_price['구분'] = apt_price['구분'].str.replace('강원도', '').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  강릉시  4719.862500
1  동해시  3435.960000
2  원주시  4943.078125
3  춘천시  4625.943750
4  홍천군  4143.314286

전처리 후 아파트 가격 데이터:
     구분     매매     전세
0   강릉시  739.0  579.0
1   고성군  604.0  418.0
2   동해시  492.0  360.0
3   삼척시  458.0  338.0
4   속초시  717.0  525.0
5   양구군  450.0  277.0
6   양양군  690.0  488.0
7   영월군  406.0  253.0
8   원주시  677.0  526.0
9   인제군  345.0  203.0
10  정선군  436.0  232.0
11  철원군  402.0  247.0
12  춘천시  804.0  635.0
13  태백시  490.0  313.0
14  평창군  351.0  227.0
15  홍천군  611.0  459.0
16  횡성군  392.0  242.0

병합된 데이터:
    구분        평균_빵가격     매매
0  강릉시  4719.862500  739.0
1  동해시  3435.960000  492.0
2  원주시  4943.078125  677.0
3  춘천시  4625.943750  804.0
4  홍천군  4143.314286  611.0

상관계수: 0.840
양의 상관관계가 있습니다: 아파트 가격이 높은 구일수록 빵 가격도 높은 경향이 있습니다.
```



In [15]:

```python
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('강원도 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.")
```

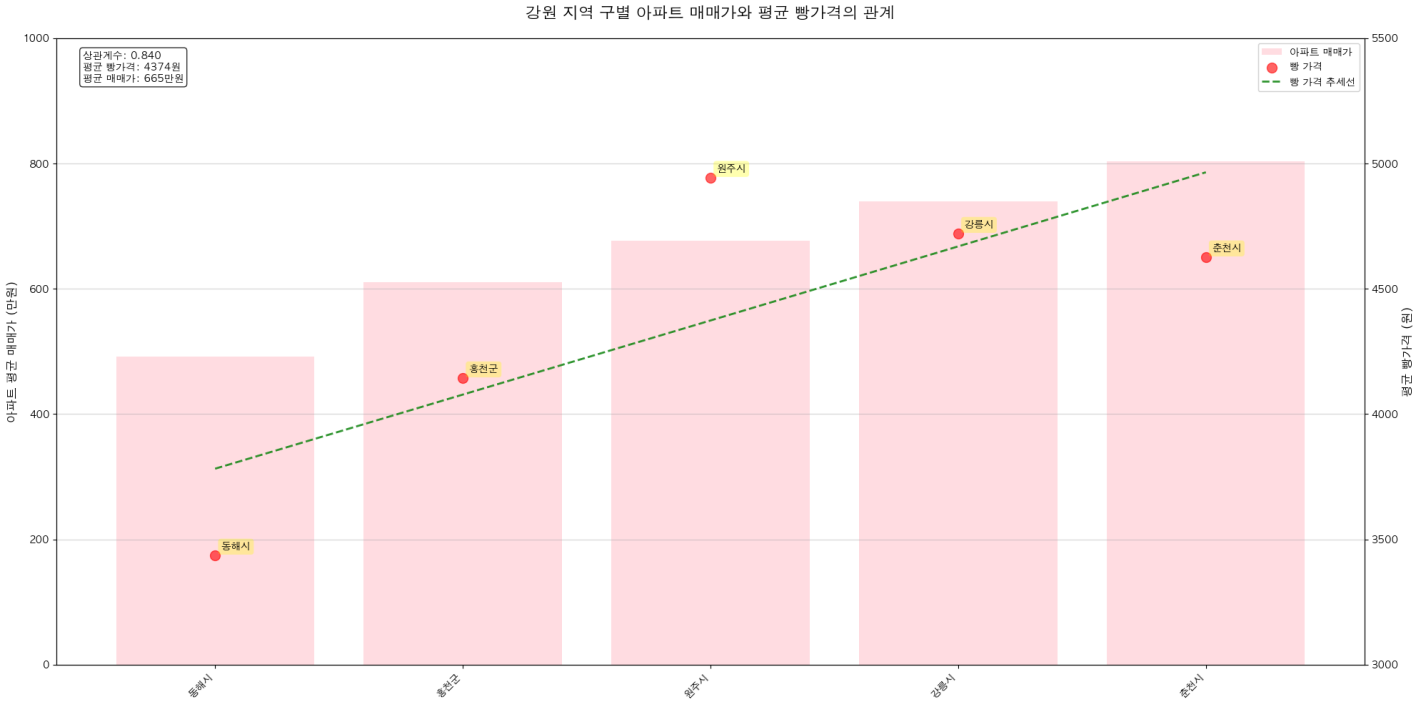



```
=== 지역별 상세 데이터 ===

아파트 가격 상위 5개 지역:
    구분     매매       평균_빵가격
3  춘천시  804.0  4625.943750
0  강릉시  739.0  4719.862500
2  원주시  677.0  4943.078125
4  홍천군  611.0  4143.314286
1  동해시  492.0  3435.960000

빵 가격 상위 5개 지역:
    구분     매매       평균_빵가격
2  원주시  677.0  4943.078125
0  강릉시  739.0  4719.862500
3  춘천시  804.0  4625.943750
4  홍천군  611.0  4143.314286
1  동해시  492.0  3435.960000

순위 차이가 가장 큰 5개 지역 (불일치도가 높은 지역):
    구분     매매       평균_빵가격  순위_차이
2  원주시  677.0  4943.078125    2.0
3  춘천시  804.0  4625.943750    2.0
0  강릉시  739.0  4719.862500    0.0
1  동해시  492.0  3435.960000    0.0
4  홍천군  611.0  4143.314286    0.0

상관계수: 0.840
양의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격도 높은 경향이 있습니다.
```

In [20]:

```python
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='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=100, 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))), "g--",
             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=10,
                     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('강원 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')
```

```python
    ax1.set_ylim(0, 1000)
    ax2.set_ylim(3000, 5500)

    # 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
3  춘천시  804.0  4625.943750
0  강릉시  739.0  4719.862500
2  원주시  677.0  4943.078125
4  홍천군  611.0  4143.314286
1  동해시  492.0  3435.960000

빵 가격 상위 5개 지역:
    구분     매매       평균_빵가격
2  원주시  677.0  4943.078125
0  강릉시  739.0  4719.862500
3  춘천시  804.0  4625.943750
4  홍천군  611.0  4143.314286
1  동해시  492.0  3435.960000

순위 차이가 가장 큰 5개 지역 (불일치도가 높은 지역):
    구분     매매       평균_빵가격  순위_차이
2  원주시  677.0  4943.078125    2.0
3  춘천시  804.0  4625.943750    2.0
0  강릉시  739.0  4719.862500    0.0
1  동해시  492.0  3435.960000    0.0
4  홍천군  611.0  4143.314286    0.0

상관계수: 0.840
양의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격도 높은 경향이 있습니다.
```