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

Out[1]:

| | 뚜레쥬르 지점 | 뚜레쥬르 전남강진 | 뚜레쥬르 나주혁신중앙 | 뚜레쥬르 순천선평 | 뚜레쥬르 순천왕지유심천 | 뚜레쥬르 전남고흥 | 뚜레쥬르 광양중마사랑 | 뚜레쥬르 전남구례 | 뚜레쥬르 목포하당중흥 | 뚜레쥬르 목포용해 | ... | 뚜레쥬르 광주수완대방 | 뚜레쥬르 광주월계 | 뚜레쥬르 광주첨단대라수 | 뚜레쥬르 광주백운 | 뚜레쥬르 광주백운DT | 뚜레쥬르 광주진월 | 뚜레쥬르 광주금남로 | 뚜레쥬르 광주서방 | 뚜레쥬르 상무 | 뚜레쥬르 광주동림 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 마늘 단짝 고구마 | 4900.0 | NaN | 4900.0 | 4900.0 | 4900.0 | 4900.0 | 4900.0 | 4900.0 | 4900 | ... | 4900.0 | NaN | 4900.0 | 4900.0 | 4900.0 | 4900 | 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 | NaN | ... | 4300.0 | NaN | 4300.0 | 4300.0 | 4300.0 | 4300 | 4300.0 | 4300.0 | 4300.0 | 4300.0 |
| 2 | BELT 샌드위치 | NaN | 6900.0 | 6900.0 | 7300.0 | 7000.0 | 7100.0 | 7000.0 | 7200.0 | 6900 | ... | 6900.0 | 6900.0 | 8500.0 | 6900.0 | 6900.0 | 6900 | NaN | 6900.0 | 6900.0 | 6900.0 |
| 3 | BLT콥 샐러드 | NaN | 8500.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 8500.0 | 8500.0 | NaN |
| 4 | 쉬림프 에그 샐러드 | NaN | NaN | NaN | 11000.0 | NaN | NaN | NaN | NaN | 10500 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

5 rows × 30 columns

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

```
Index(['뚜레쥬르 지점', '뚜레쥬르 전남강진', '뚜레쥬르 나주혁신중앙', '뚜레쥬르 순천선평', '뚜레쥬르 순천왕지유심천',
       '뚜레쥬르 전남고흥', '뚜레쥬르 광양중마사랑', '뚜레쥬르 전남구례', '뚜레쥬르 목포하당중흥', '뚜레쥬르 목포용해',
       '뚜레쥬르 목포남악부영', '뚜레쥬르 남악오룡', '뚜레쥬르 순천용당', '뚜레쥬르 순천조례명성', '뚜레쥬르 여수도원',
       '뚜레쥬르 여수죽림', '뚜레쥬르 여수쌍봉', '뚜레쥬르 여수엑스포타운', '뚜레쥬르 전남완도', '뚜레쥬르 진도',
       '뚜레쥬르 광주수완대방', '뚜레쥬르 광주월계', '뚜레쥬르 광주첨단대라수', '뚜레쥬르 광주백운', '뚜레쥬르 광주백운DT',
       '뚜레쥬르 광주진월', '뚜레쥬르 광주금남로', '뚜레쥬르 광주서방', '뚜레쥬르 상무', '뚜레쥬르 광주동림'],
      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','까까웨드'],
        '피자빵,고로케': ['고로케', '소시지브레드','피자토스트','NEW어니언소시지포카치아'],
        '파이/패스트리': ['바통쉬크레','크라상','애플파이','유자파이'],
        '간식빵': ['소금버터롤','치즈방앗간','깨찰빵','소보로빵','오리지널 커피번','카페모카빵','꽈배기','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/jeonnam_adress.csv'

def process_address(address):
    try:
        # 수동 수정
```

```
        if address == '경기도 동탄지성로469번길 60 5단지 상가1동107호,108호,109호':
            return '경기도 화성시'

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

        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/jeonnam-pricedata.csv',
                                 './adress_process/jeonnam_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 광양중마사랑 | 전라남도 광양시 | 전남 | 2960.0 | 4238.9 | 7100.0 | NaN | 5090.9 | 4600.0 | 3340.0 | 2600.0 | 3000.0 |
| 1 | 뚜레쥬르 광주금남로 | 광주광역시 동구 | 전남 | 2985.7 | 3944.0 | NaN | NaN | 4500.0 | 4600.0 | 2400.0 | 2933.3 | 2950.0 |
| 2 | 뚜레쥬르 광주동림 | 광주광역시 서구 | 전남 | 3284.6 | 4404.7 | 7137.5 | 8350.0 | 5100.0 | 4600.0 | 3516.7 | 2933.3 | 3220.0 |
| 3 | 뚜레쥬르 광주백운 | 광주광역시 남구 | 전남 | 3045.5 | 4147.1 | 7133.3 | NaN | 4762.5 | 4600.0 | 3337.5 | 2933.3 | 3180.0 |
| 4 | 뚜레쥬르 광주백운DT | 광주광역시 남구 | 전남 | 3045.5 | 4128.8 | 7133.3 | NaN | 4762.5 | 4600.0 | 3337.5 | 2933.3 | 3200.0 |
| 5 | 뚜레쥬르 광주서방 | 광주광역시 북구 | 전남 | 2671.4 | 3777.1 | 7340.0 | 8500.0 | 4866.7 | 4600.0 | 3514.3 | 3100.0 | 3025.0 |
| 6 | 뚜레쥬르 광주수완대방 | 광주광역시 광산구 | 전남 | 2990.0 | 4439.5 | 7150.0 | NaN | 5054.5 | 4600.0 | 3257.1 | 2400.0 | 3200.0 |
| 7 | 뚜레쥬르 광주월계 | 광주광역시 광산구 | 전남 | 3000.0 | 4295.6 | 7175.0 | NaN | 5375.0 | NaN | 3637.5 | 2933.3 | 3040.0 |
| 8 | 뚜레쥬르 광주진월 | 광주광역시 남구 | 전남 | 2790.0 | 4226.1 | 7111.1 | 8350.0 | 4933.3 | 4600.0 | 3333.3 | 2933.3 | 3025.0 |
| 9 | 뚜레쥬르 광주첨단대라수 | 광주광역시 광산구 | 전남 | 2600.0 | 4272.5 | 8500.0 | 8500.0 | 5254.5 | 4600.0 | 3460.0 | 2800.0 | 3116.7 |
| 10 | 뚜레쥬르 나주혁신중앙 | 전라남도 나주시 | 전남 | 2790.0 | 4166.0 | 7300.0 | 8500.0 | 5114.3 | 4300.0 | 3257.1 | 2933.3 | 3200.0 |
| 11 | 뚜레쥬르 남악오룡 | 전라남도 무안군 | 전남 | 2900.0 | 4478.6 | 7900.0 | 8500.0 | 4854.5 | 4600.0 | 2800.0 | 2900.0 | 3560.0 |
| 12 | 뚜레쥬르 목포남악부영 | 전라남도 무안군 | 전남 | 3125.0 | 3956.2 | 7220.0 | 9400.0 | 4660.0 | 4600.0 | 3400.0 | 3166.7 | 4100.0 |
| 13 | 뚜레쥬르 목포용해 | 전라남도 목포시 | 전남 | 2850.0 | 4091.2 | 7128.6 | 9400.0 | 5037.5 | 4900.0 | 3050.0 | 2933.3 | 3200.0 |
| 14 | 뚜레쥬르 목포하당중흥 | 전라남도 목포시 | 전남 | 2955.6 | 4197.9 | 7277.8 | NaN | 4900.0 | 4600.0 | 3300.0 | 3033.3 | 3233.3 |
| 15 | 뚜레쥬르 상무 | 광주광역시 서구 | 전남 | 2140.0 | 4188.6 | 7260.0 | 8500.0 | 5100.0 | 4600.0 | 3766.7 | 2933.3 | 3025.0 |
| 16 | 뚜레쥬르 순천선평 | 전라남도 순천시 | 전남 | 2650.0 | 4551.7 | 6050.0 | NaN | 4757.1 | 4600.0 | 4900.0 | 2933.3 | 3133.3 |
| 17 | 뚜레쥬르 순천왕지유심천 | 전라남도 순천시 | 전남 | 2744.4 | 4400.0 | 6200.0 | 11000.0 | 5180.0 | 5066.7 | 3525.0 | 2800.0 | 3100.0 |
| 18 | 뚜레쥬르 순천용당 | 전라남도 순천시 | 전남 | 2790.0 | 4444.1 | 7140.0 | NaN | 5135.7 | 4600.0 | 3314.3 | 3100.0 | 3200.0 |
| 19 | 뚜레쥬르 순천조례명성 | 전라남도 순천시 | 전남 | 2428.6 | 4362.9 | 7100.0 | NaN | 5000.0 | 4300.0 | 3562.5 | 2933.3 | 3025.0 |
| 20 | 뚜레쥬르 여수도원 | 전라남도 여수시 | 전남 | 3127.3 | 4382.9 | 7150.0 | NaN | 4822.2 | 4933.3 | 3587.5 | 3100.0 | 3200.0 |
| 21 | 뚜레쥬르 여수쌍봉 | 전라남도 여수시 | 전남 | 3116.7 | 4220.5 | 6966.7 | 8350.0 | 4954.5 | 4600.0 | 3580.0 | 2933.3 | 3200.0 |
| 22 | 뚜레쥬르 여수엑스포타운 | 전라남도 여수시 | 전남 | 2970.0 | 4193.9 | 7400.0 | 8500.0 | 4840.0 | 4600.0 | 3300.0 | 2933.3 | 3120.0 |
| 23 | 뚜레쥬르 여수죽림 | 전라남도 여수시 | 전남 | 2616.7 | 4235.1 | 7400.0 | NaN | 4925.0 | 4600.0 | 3660.0 | 3250.0 | 3266.7 |
| 24 | 뚜레쥬르 전남강진 | 전라남도 강진군 | 전남 | 3100.0 | 4297.4 | 7260.0 | NaN | 5750.0 | 4600.0 | 2960.0 | 3133.3 | 3333.3 |
| 25 | 뚜레쥬르 전남고흥 | 전라남도 고흥군 | 전남 | 2550.0 | 4323.9 | 7155.6 | NaN | 5191.7 | 4600.0 | 3160.0 | 2933.3 | 2950.0 |
| 26 | 뚜레쥬르 전남구례 | 전라남도 구례군 | 전남 | 2857.1 | 4097.4 | 7112.5 | NaN | 4720.0 | 4933.3 | 3475.0 | 3000.0 | 3100.0 |
| 27 | 뚜레쥬르 전남완도 | 전라남도 완도군 | 전남 | 3475.0 | 4376.0 | 7120.0 | 8300.0 | 5242.9 | 4600.0 | 3433.3 | 2800.0 | 3280.0 |
| 28 | 뚜레쥬르 진도 | 전라남도 진도군 | 전남 | 2616.7 | 4022.5 | 7180.0 | NaN | 4833.3 | 4600.0 | 3100.0 | 2933.3 | 2800.0 |

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_jeonla/average_allbread_jn.csv', index=False, encoding='utf-8-sig')
grouped_df
```

Out[4]:

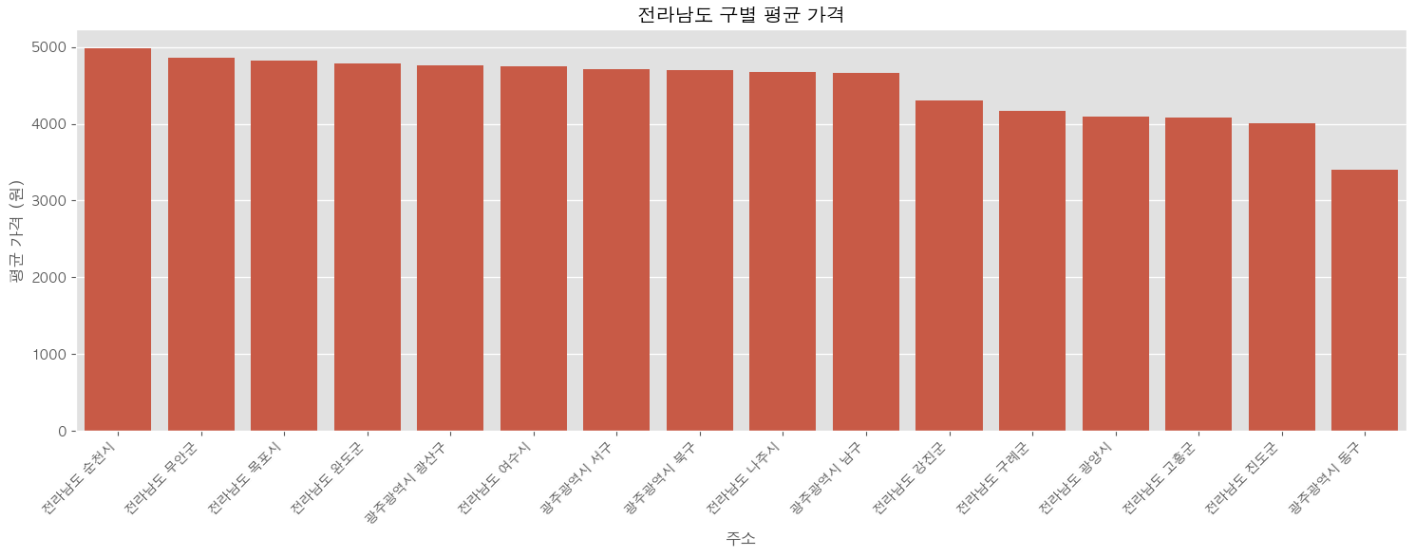
| | 주소 | 평균가격 |
|---|---|---|
| 0 | 전라남도 순천시 | 4977.162500 |
| 1 | 전라남도 무안군 | 4855.387500 |
| 2 | 전라남도 목포시 | 4824.962500 |
| 3 | 전라남도 완도군 | 4781.400000 |
| 4 | 광주광역시 광산구 | 4760.150000 |
| 5 | 전라남도 여수시 | 4745.412500 |
| 6 | 광주광역시 서구 | 4716.693750 |
| 7 | 광주광역시 북구 | 4702.175000 |
| 8 | 전라남도 나주시 | 4674.337500 |
| 9 | 광주광역시 남구 | 4657.508333 |
| 10 | 전라남도 강진군 | 4305.228571 |
| 11 | 전라남도 구례군 | 4171.128571 |
| 12 | 전라남도 광양시 | 4098.700000 |
| 13 | 전라남도 고흥군 | 4077.228571 |
| 14 | 전라남도 진도군 | 4009.042857 |
| 15 | 광주광역시 동구 | 3394.833333 |

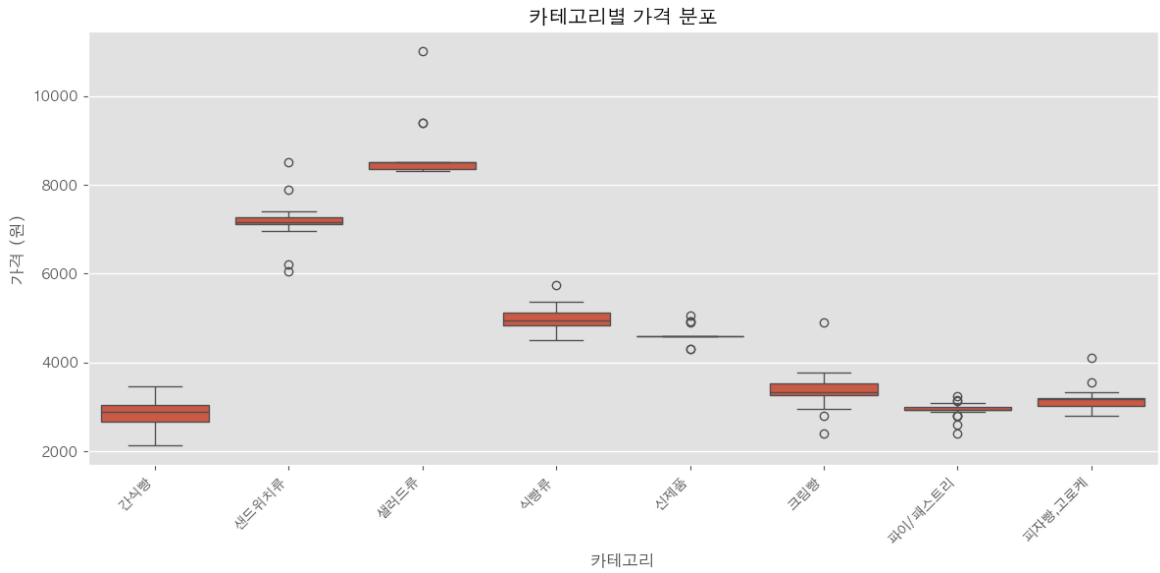
In [5]:
```
categories = ['간식빵', '샌드위치류', '샐러드류', '식빵류', '신제품', '크림빵', '파이/패스트리', '피자빵,고로케']

# 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_jeonla/average_categorized_shopjn.csv', encoding='utf-8-sig')
grouped_df
```

Out[5]:

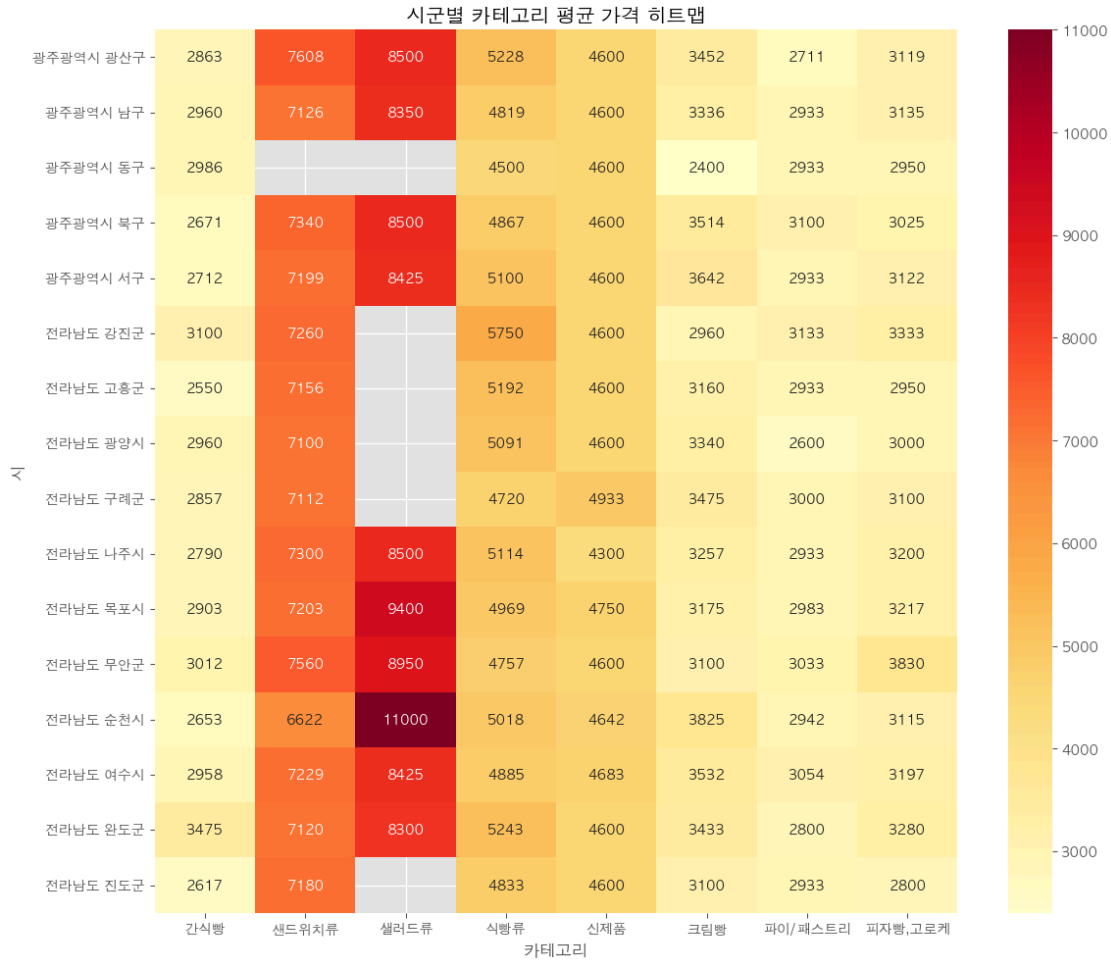
| 주소 | 간식빵 | 샌드위치류 | 샐러드류 | 식빵류 | 신제품 | 크림빵 | 파이/패스트리 | 피자빵,고로케 |
|---|---|---|---|---|---|---|---|---|
| 광주광역시 광산구 | 2863.33 | 7608.33 | 8500.0 | 5228.00 | 4600.00 | 3451.53 | 2711.10 | 3118.90 |
| 광주광역시 남구 | 2960.33 | 7125.90 | 8350.0 | 4819.43 | 4600.00 | 3336.10 | 2933.30 | 3135.00 |
| 광주광역시 동구 | 2985.70 | NaN | NaN | 4500.00 | 4600.00 | 2400.00 | 2933.30 | 2950.00 |
| 광주광역시 북구 | 2671.40 | 7340.00 | 8500.0 | 4866.70 | 4600.00 | 3514.30 | 3100.00 | 3025.00 |
| 광주광역시 서구 | 2712.30 | 7198.75 | 8425.0 | 5100.00 | 4600.00 | 3641.70 | 2933.30 | 3122.50 |
| 전라남도 강진군 | 3100.00 | 7260.00 | NaN | 5750.00 | 4600.00 | 2960.00 | 3133.30 | 3333.30 |
| 전라남도 고흥군 | 2550.00 | 7155.60 | NaN | 5191.70 | 4600.00 | 3160.00 | 2933.30 | 2950.00 |
| 전라남도 광양시 | 2960.00 | 7100.00 | NaN | 5090.90 | 4600.00 | 3340.00 | 2600.00 | 3000.00 |
| 전라남도 구례군 | 2857.10 | 7112.50 | NaN | 4720.00 | 4933.30 | 3475.00 | 3000.00 | 3100.00 |
| 전라남도 나주시 | 2790.00 | 7300.00 | 8500.0 | 5114.30 | 4300.00 | 3257.10 | 2933.30 | 3200.00 |
| 전라남도 목포시 | 2902.80 | 7203.20 | 9400.0 | 4968.75 | 4750.00 | 3175.00 | 2983.30 | 3216.65 |
| 전라남도 무안군 | 3012.50 | 7560.00 | 8950.0 | 4757.25 | 4600.00 | 3100.00 | 3033.35 | 3830.00 |
| 전라남도 순천시 | 2653.25 | 6622.50 | 11000.0 | 5018.20 | 4641.68 | 3825.45 | 2941.65 | 3114.58 |
| 전라남도 여수시 | 2957.68 | 7229.18 | 8425.0 | 4885.42 | 4683.32 | 3531.88 | 3054.15 | 3196.68 |
| 전라남도 완도군 | 3475.00 | 7120.00 | 8300.0 | 5242.90 | 4600.00 | 3433.30 | 2800.00 | 3280.00 |
| 전라남도 진도군 | 2616.70 | 7180.00 | NaN | 4833.30 | 4600.00 | 3100.00 | 2933.30 | 2800.00 |

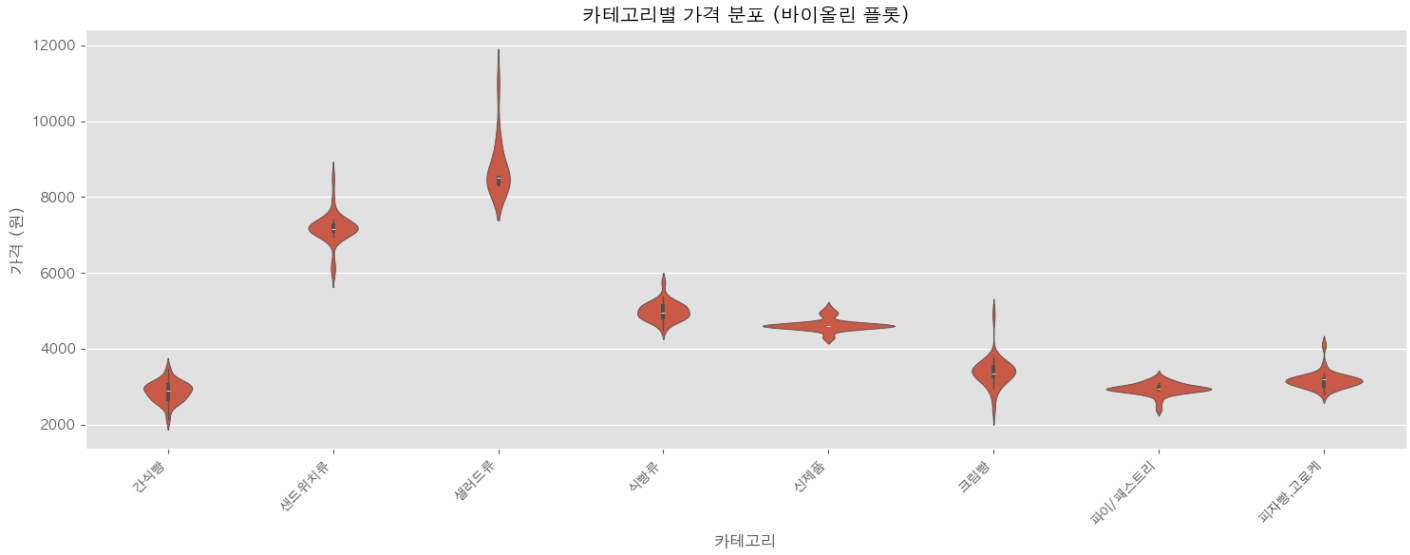
In [6]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from matplotlib import font_manager, rc

# 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('전라남도 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()
```



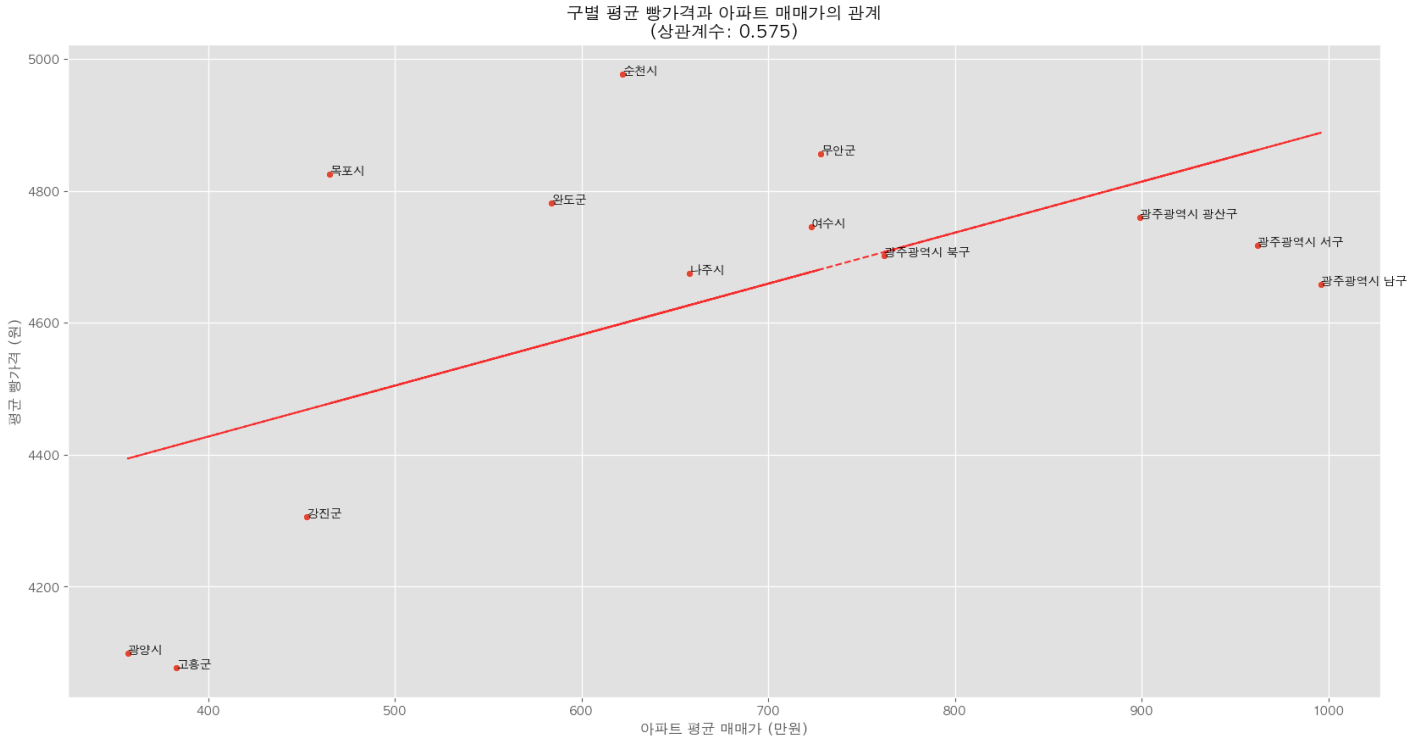




In [8]:

```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']
# '경상남도'제거
bread_price_by_district['구분'] = bread_price_by_district['구분'].str.replace('전라남도', '').str.strip()

# 아파트 가격 데이터 전처리
apt_price = pd.read_csv('anal_jeonla/jeonnam_apt_price.csv')
# '경상남도'와 '구' 제거
apt_price['구분'] = apt_price['구분'].str.replace('전라남도', '').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('광주광역시 동구')]

# 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)
```

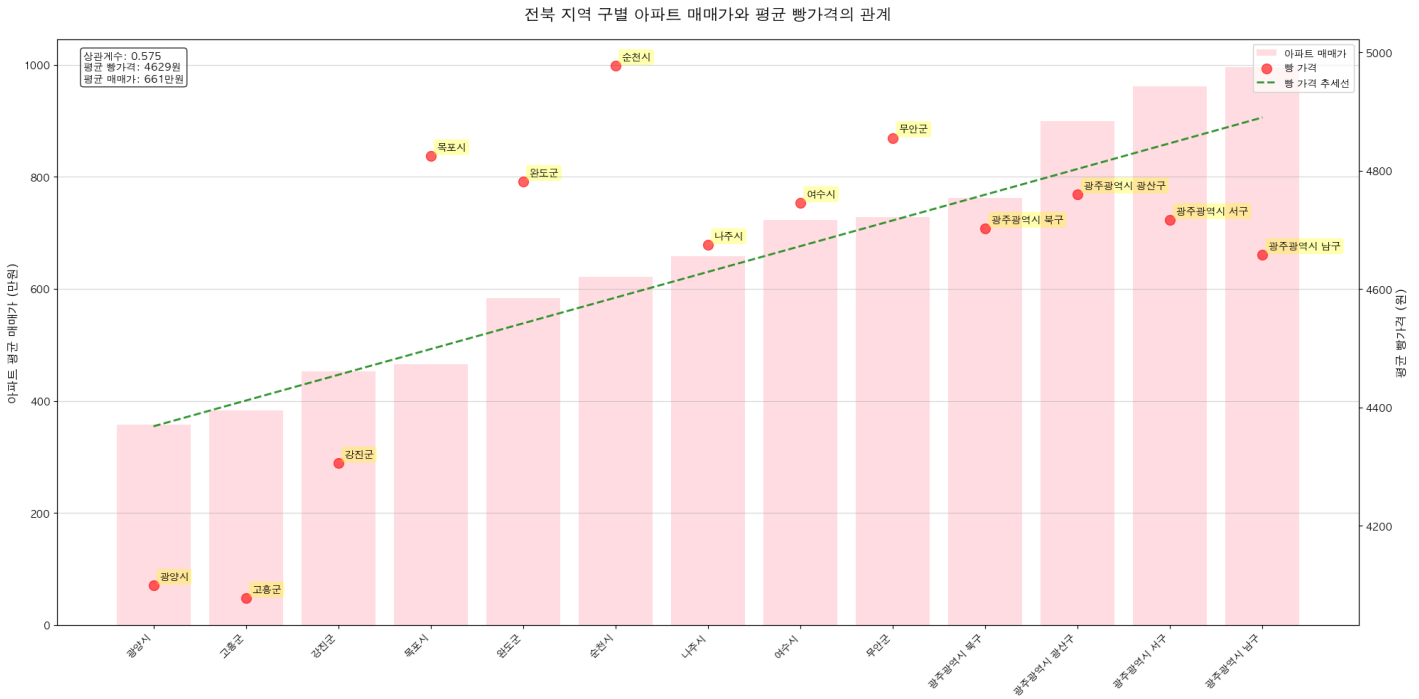
```
    # 각 점에 구 이름 표시
    for idx, row in merged_df.iterrows():
        plt.annotate(row['구분'], (row['매매'], row['평균_빵가격']))

    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   광주광역시 광산구  4760.150000
1    광주광역시 남구  4657.508333
2    광주광역시 동구  3394.833333
3    광주광역시 북구  4702.175000
4    광주광역시 서구  4716.693750
5         강진군  4305.228571
6         고흥군  4077.228571
7         광양시  4098.700000
8         구례군  4171.128571
9         나주시  4674.337500
10        목포시  4824.962500
11        무안군  4855.387500
12        순천시  4977.162500
13        여수시  4745.412500
14        완도군  4781.400000
15        진도군  4009.042857

전처리 후 아파트 가격 데이터:
          구분   매매   전세
0         강진군  453  230
1         고흥군  383  171
2         곡성군  370  213
3         광양시  357  296
4         나주시  658  473
5         담양군  460  267
6         목포시  465  369
7         무안군  728  570
8         보성군  417  300
9         순천시  622  499
10        여수시  723  519
11        영광군  553  395
12        영암군  300  205
13        완도군  584  310
14        장성군  303  200
15        장흥군  653  467
16        해남군  614  422
17        화순군  459  349
18  광주광역시 광산구  899  641
19   광주광역시 남구  996  653
21   광주광역시 북구  762  564
22   광주광역시 서구  962  676

병합된 데이터:
          구분        평균_빵가격   매매
0   광주광역시 광산구  4760.150000  899
1    광주광역시 남구  4657.508333  996
2    광주광역시 북구  4702.175000  762
3    광주광역시 서구  4716.693750  962
4         강진군  4305.228571  453
5         고흥군  4077.228571  383
6         광양시  4098.700000  357
7         나주시  4674.337500  658
8         목포시  4824.962500  465
9         무안군  4855.387500  728
10        순천시  4977.162500  622
11        여수시  4745.412500  723
12        완도군  4781.400000  584

상관계수: 0.575
양의 상관관계가 있습니다: 아파트 가격이 높은 구일수록 빵 가격도 높은 경향이 있습니다.
```




In [9]:
```
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'
```

```python
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('전북 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
1    광주광역시 남구   996  4657.508333
3    광주광역시 서구   962  4716.693750
0   광주광역시 광산구   899  4760.150000
2    광주광역시 북구   762  4702.175000
9          무안군   728  4855.387500

빵 가격 상위 5개 지역:
             구분    매매      평균_빵가격
10         순천시   622  4977.1625
9          무안군   728  4855.3875
8          목포시   465  4824.9625
12         완도군   584  4781.4000
0    광주광역시 광산구   899  4760.1500

순위 차이가 가장 큰 5개 지역 (불일치도가 높은 지역):
            구분    매매        평균_빵가격  순위_차이
1    광주광역시 남구   996  4657.508333    9.0
8          목포시   465  4824.962500    7.0
10         순천시   622  4977.162500    7.0
3    광주광역시 서구   962  4716.693750    5.0
12         완도군   584  4781.400000    5.0

상관계수: 0.575
양의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격도 높은 경향이 있습니다.
```

In [10]:

```python
import pandas as pd
import numpy as np
jbile_path ='./cafedata/jeonbuk-pricedata.csv'
jbdf = pd.read_csv(jbile_path)
jbdf.head()
```

Out[10]:

| | 뚜레쥬르 지점 | 뚜레쥬르 군산디오션 | 뚜레쥬르 군산터미널 | 뚜레쥬르 김제검산 | 뚜레쥬르 삼례농협하나로마트 | 뚜레쥬르 익산제일 | 뚜레쥬르 익산부송 | 뚜레쥬르 전주하가 | 뚜레쥬르 전주서신 | 뚜레쥬르 전주신일강변 | 뚜레쥬르 전주효자1가 | 뚜레쥬르 카페정읍수성 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 마늘 단짝 고구마 | 4900.0 | 4900.0 | 4900.0 | 4900.0 | NaN | 4900.0 | 4900.0 | 4900.0 | 4900.0 | 4900.0 | 4900.0 |
| 1 | 깊은 밤 뱅스위스 | 4300.0 | 4300.0 | 4300.0 | 4300.0 | NaN | 4300.0 | 4300.0 | 4300.0 | 4300.0 | 4300.0 | 4300.0 |
| 2 | BELT 샌드위치 | 6900.0 | NaN | 7100.0 | 6900.0 | 6900.0 | NaN | 6900.0 | 6900.0 | NaN | 6900.0 | 6900.0 |
| 3 | BLT콥 샐러드 | NaN | 8500.0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| 4 | 쉬림프 에그 샐러드 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 10500.0 |

In [11]:

```python
import re

def categorize_menu(jbdf):
    # 키워드 기반 카테고리 매핑 딕셔너리
    category_keywords = {
        '샌드위치류': ['샌드위치', 'BELT', 'BLT', 'V.E.L.T'],
        '샐러드류': ['샐러드'],
        '식빵류': ['식빵', '우유물', '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','꽈배기','옛날 단팥 도넛',r'^단팥빵$','단팥소보로빵'],
        '신제품': ['마구마구', '단짝', '뱅스위스']
    }

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

    # 각 메뉴명에 대해 카테고리 매핑
    for idx, menu_name in enumerate(jbdf['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):
                jbdf.loc[idx, '카테고리'] = category
                break

    return jbdf

def analyze_categories_by_store(jbdf):
    # 매장별 카테고리별 기본 통계
    stores = jbdf.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)
        jbdf[store] = pd.to_numeric(jbdf[store], errors='coerce')

        temp = jbdf.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(jbile_path):
    # CSV 파일 읽기
    jbdf = pd.read_csv(jbile_path)

    # 카테고리 지정
    jbdf = categorize_menu(jbdf)

    # 매장별 카테고리별 분석
    stats = analyze_categories_by_store(jbdf)

    # 피벗 테이블 생성
    pivot_table = pivot_store_category(stats)

    return jbdf, pivot_table

# 파일 처리 및 결과 생성
jbdf, pivot_table = process_bakery_data(jbile_path)

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

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

storeinfo_filepath='./adress_process/jeonbuk_adress.csv'

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

        # 정규표현식으로 '전라북도 XX시' 추출
        match = re.match(r'전라북도\s+\w+시', address) or re.match(r'전라북도\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
    jbdf = pd.read_csv(price_filepath)

    # 매장 정보 데이터 로드
    store_info = load_store_info(store_info_filepath)

    # 카테고리 지정
    jbdf = categorize_menu(jbdf)

    # 매장별 카테고리별 분석
    stats = analyze_categories_by_store(jbdf)

    # 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 jbdf, result

# 실제 파일 경로로 호출
jbdf, result = process_bakery_data('./cafedata/jeonbuk-pricedata.csv',
                                   './adress_process/jeonbuk_adress.csv')

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

# result.to_csv('./anal_gyeongsang/?시별_카테고리_평균가격.csv', encoding='utf-8-sig')
```

매장별 카테고리별 평균 가격 (주소 정보 포함)

| | 매장명 | 주소 | 지역 | 간식빵 | 기타 | 샌드위치류 | 샐러드류 | 식빵류 | 신제품 | 크림빵 | 파이/패스트리 | 피자빵,고로케 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 뚜레쥬르 군산디오션 | 전라북도 군산시 | 전북 | 2970.0 | 4245.7 | 6950.0 | 8500.0 | 4772.7 | 4600.0 | 3566.7 | 2933.3 | 3200.0 |
| 1 | 뚜레쥬르 군산터미널 | 전라북도 군산시 | 전북 | 2790.0 | 4184.4 | 7314.3 | NaN | 5137.5 | 4600.0 | 3375.0 | 2933.3 | 3300.0 |
| 2 | 뚜레쥬르 김제검산 | 전라북도 김제시 | 전북 | 2737.5 | 4275.0 | 7275.0 | NaN | 5163.6 | 4600.0 | 3166.7 | 3333.3 | 3460.0 |
| 3 | 뚜레쥬르 삼례농협하나로마트 | 전라북도 완주군 | 전북 | 2962.5 | 4343.9 | 7060.0 | 8350.0 | 4916.7 | 4600.0 | 3480.0 | 3100.0 | 3100.0 |
| 4 | 뚜레쥬르 익산부송 | 전라북도 익산시 | 전북 | 2900.0 | 4248.9 | 7150.0 | NaN | 5154.5 | 4600.0 | 3033.3 | 2800.0 | 3040.0 |
| 5 | 뚜레쥬르 익산제일 | 전라북도 익산시 | 전북 | 3083.3 | 4593.8 | 7066.7 | NaN | 5150.0 | NaN | 3100.0 | 3100.0 | 3133.3 |
| 6 | 뚜레쥬르 전주서신 | 전라북도 전주시 | 전북 | 3100.0 | 4042.9 | 7170.0 | NaN | 4688.9 | 4600.0 | 3333.3 | 2800.0 | 3150.0 |
| 7 | 뚜레쥬르 전주신일강변 | 전라북도 전주시 | 전북 | 2983.3 | 4020.0 | 7150.0 | NaN | 4983.3 | 4600.0 | 3233.3 | 3200.0 | 3166.7 |
| 8 | 뚜레쥬르 전주하가 | 전라북도 전주시 | 전북 | 3462.5 | 4036.7 | 7080.0 | NaN | 4700.0 | 4600.0 | 3100.0 | 2933.3 | 3120.0 |
| 9 | 뚜레쥬르 전주효자1가 | 전라북도 전주시 | 전북 | 3054.5 | 4227.1 | 6533.3 | NaN | 4908.3 | 5066.7 | 3622.2 | 2933.3 | 3100.0 |
| 10 | 뚜레쥬르 카페정읍수성 | 전라북도 정읍시 | 전북 | 2855.6 | 4441.9 | 6770.0 | 8800.0 | 5111.1 | 4600.0 | 3916.7 | 2966.7 | 3125.0 |

In [12]:
```
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_jeonla/average_allbread_jb.csv', index=False, encoding='utf-8-sig')
grouped_df
```

Out[12]:

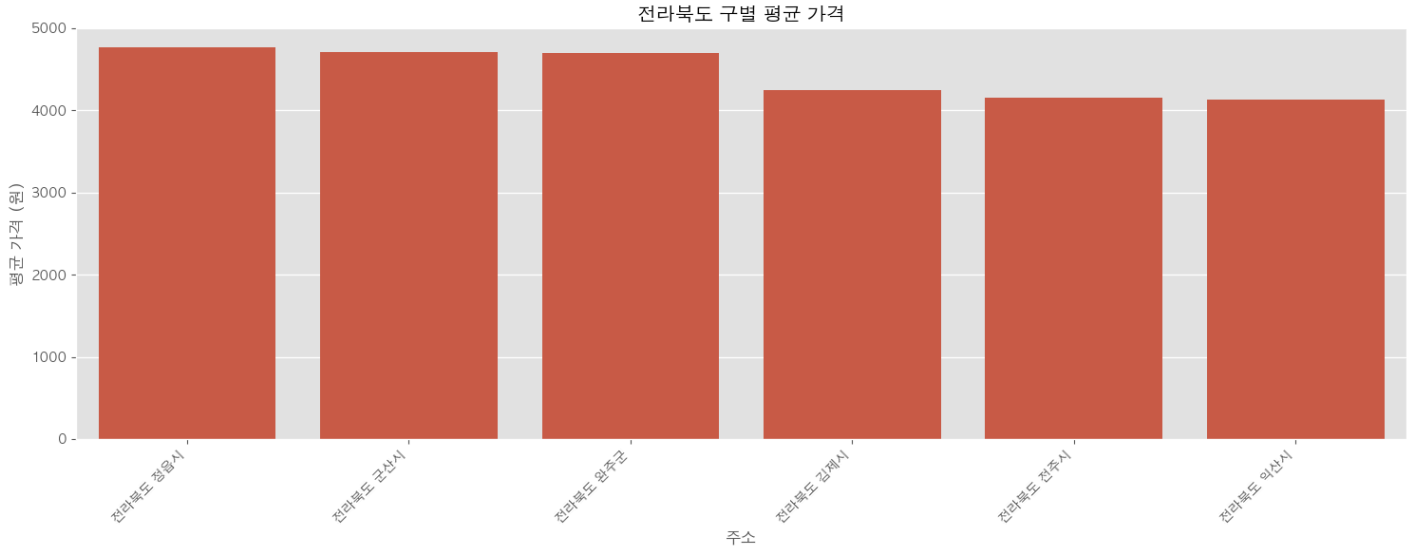
| | 주소 | 평균가격 |
|---|---|---|
| 0 | 전라북도 정읍시 | 4768.137500 |
| 1 | 전라북도 군산시 | 4715.175000 |
| 2 | 전라북도 완주군 | 4696.150000 |
| 3 | 전라북도 김제시 | 4248.014286 |
| 4 | 전라북도 전주시 | 4156.175000 |
| 5 | 전라북도 익산시 | 4136.507143 |

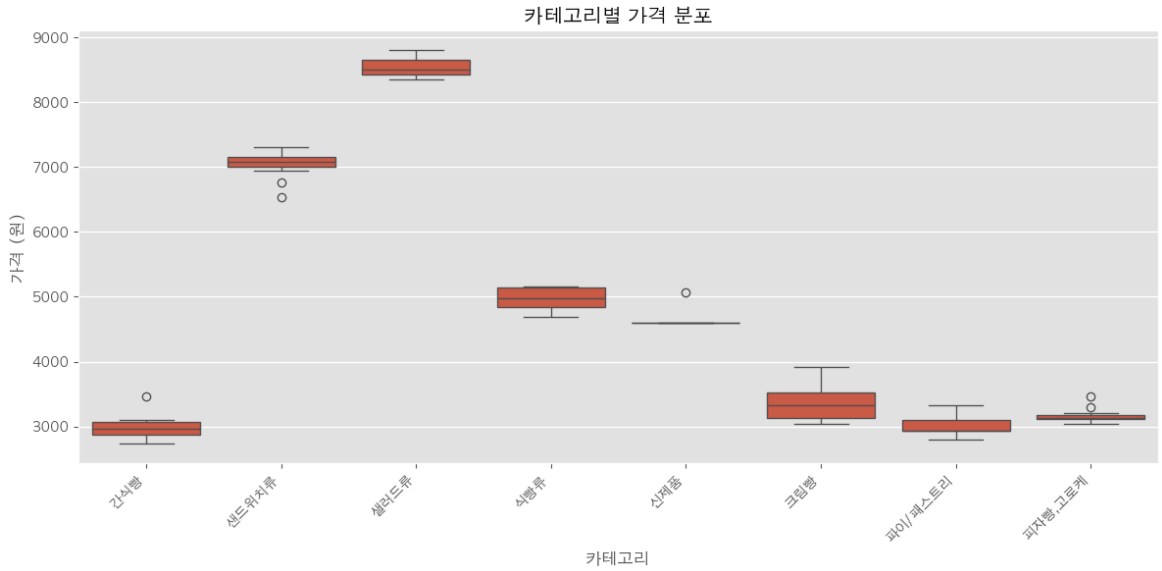
In [13]:
```
categories = ['간식빵', '샌드위치류', '샐러드류', '식빵류', '신제품', '크림빵', '파이/패스트리', '피자빵,고로케']

# 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_jeonla/average_categorized_jbshop.csv', encoding='utf-8-sig')
grouped_df
```

Out[13]:

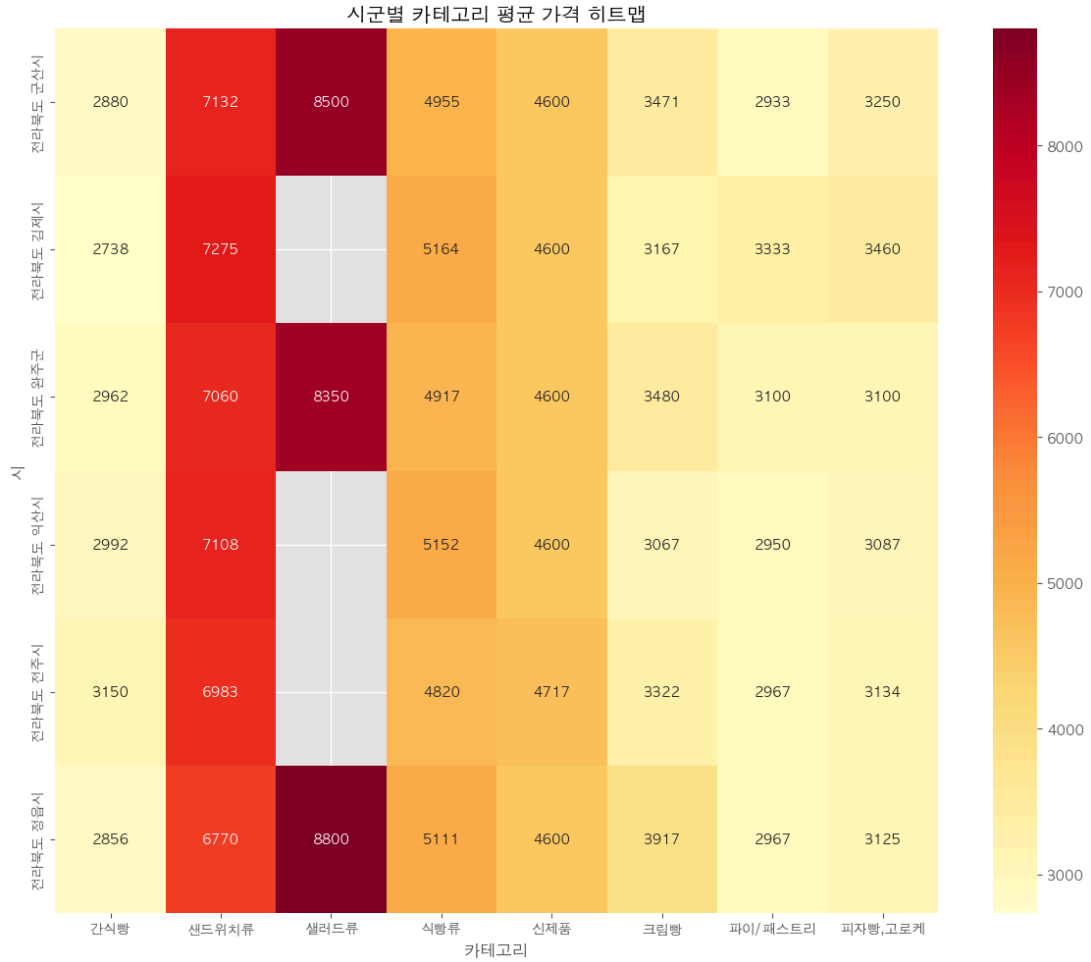
| 주소 | 간식빵 | 샌드위치류 | 샐러드류 | 식빵류 | 신제품 | 크림빵 | 파이/패스트리 | 피자빵,고로케 |
|---|---|---|---|---|---|---|---|---|
| 전라북도 군산시 | 2880.00 | 7132.15 | 8500.0 | 4955.10 | 4600.00 | 3470.85 | 2933.30 | 3250.00 |
| 전라북도 김제시 | 2737.50 | 7275.00 | NaN | 5163.60 | 4600.00 | 3166.70 | 3333.30 | 3460.00 |
| 전라북도 완주군 | 2962.50 | 7060.00 | 8350.0 | 4916.70 | 4600.00 | 3480.00 | 3100.00 | 3100.00 |
| 전라북도 익산시 | 2991.65 | 7108.35 | NaN | 5152.25 | 4600.00 | 3066.65 | 2950.00 | 3086.65 |
| 전라북도 전주시 | 3150.08 | 6983.32 | NaN | 4820.12 | 4716.68 | 3322.20 | 2966.65 | 3134.18 |
| 전라북도 정읍시 | 2855.60 | 6770.00 | 8800.0 | 5111.10 | 4600.00 | 3916.70 | 2966.70 | 3125.00 |

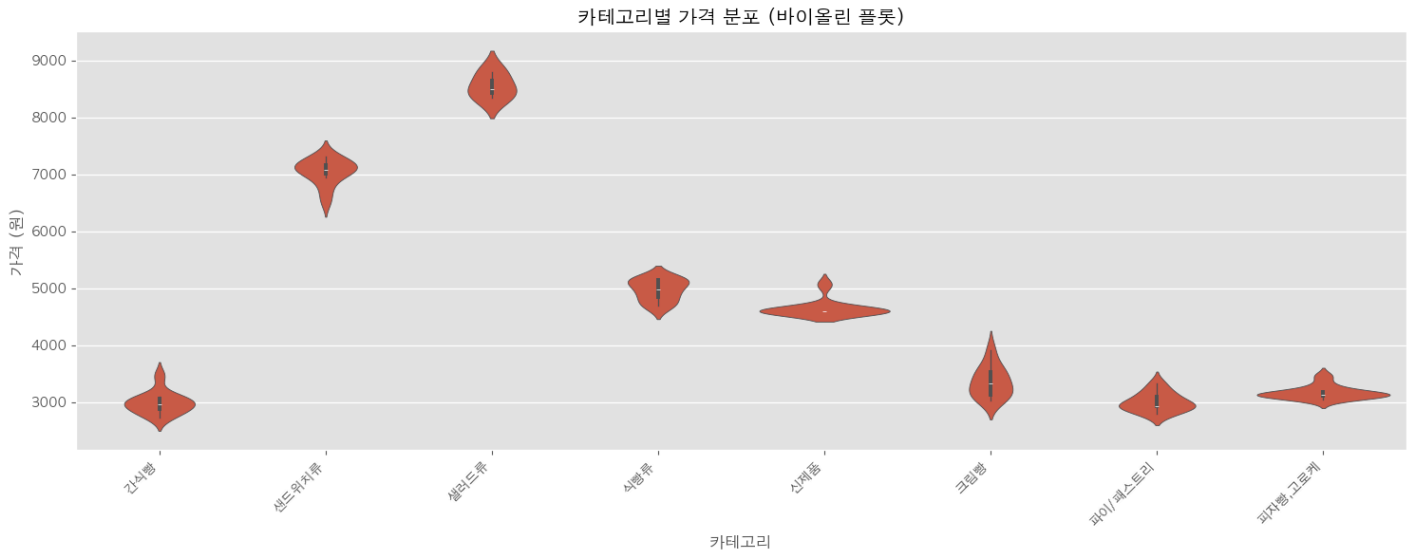
In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from matplotlib import font_manager, rc

# 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('전라북도 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()
```





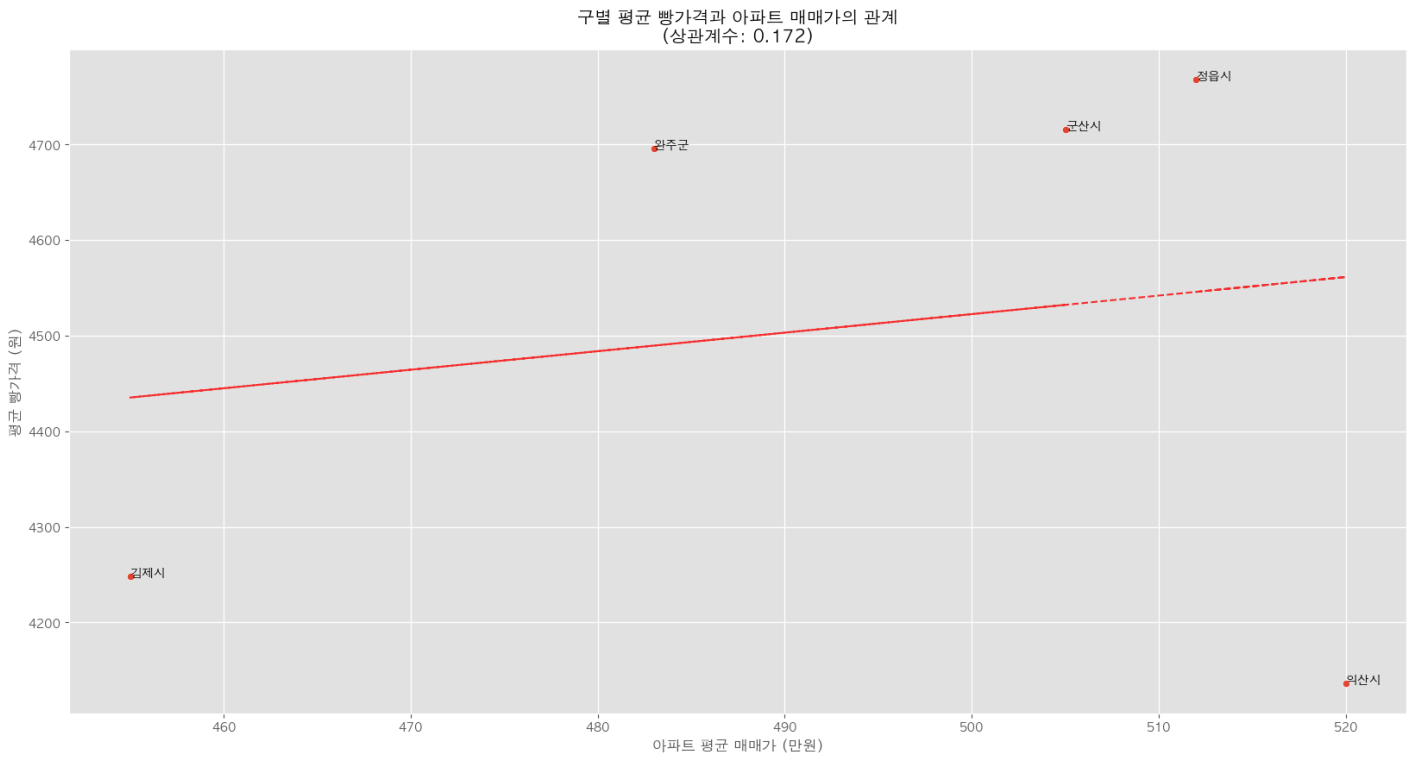

In [15]:
```python
# 3. 구별/카테고리별 평균 가격 히트맵
plt.figure(figsize=(12, 10))
pivot_data = result.groupby('주소')[categories].mean()
sns.heatmap(pivot_data, annot=True, fmt='.0f', cmap='YlOrRd')
plt.title('시군별 카테고리 평균 가격 히트맵')
plt.ylabel('시')
plt.xlabel('카테고리')
plt.tight_layout()
plt.show()

# 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 [16]:
```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']
# '전라북도' 제거
bread_price_by_district['구분'] = bread_price_by_district['구분'].str.replace('전라북도', '').str.strip()

# 아파트 가격 데이터 전처리
apt_price = pd.read_csv('anal_jeonla/jeonbuk_apt_price.csv')
# '경상남도'와 '구' 제거
apt_price['구분'] = apt_price['구분'].str.replace('전라북도', '').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('전주시')]


# 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.")
```

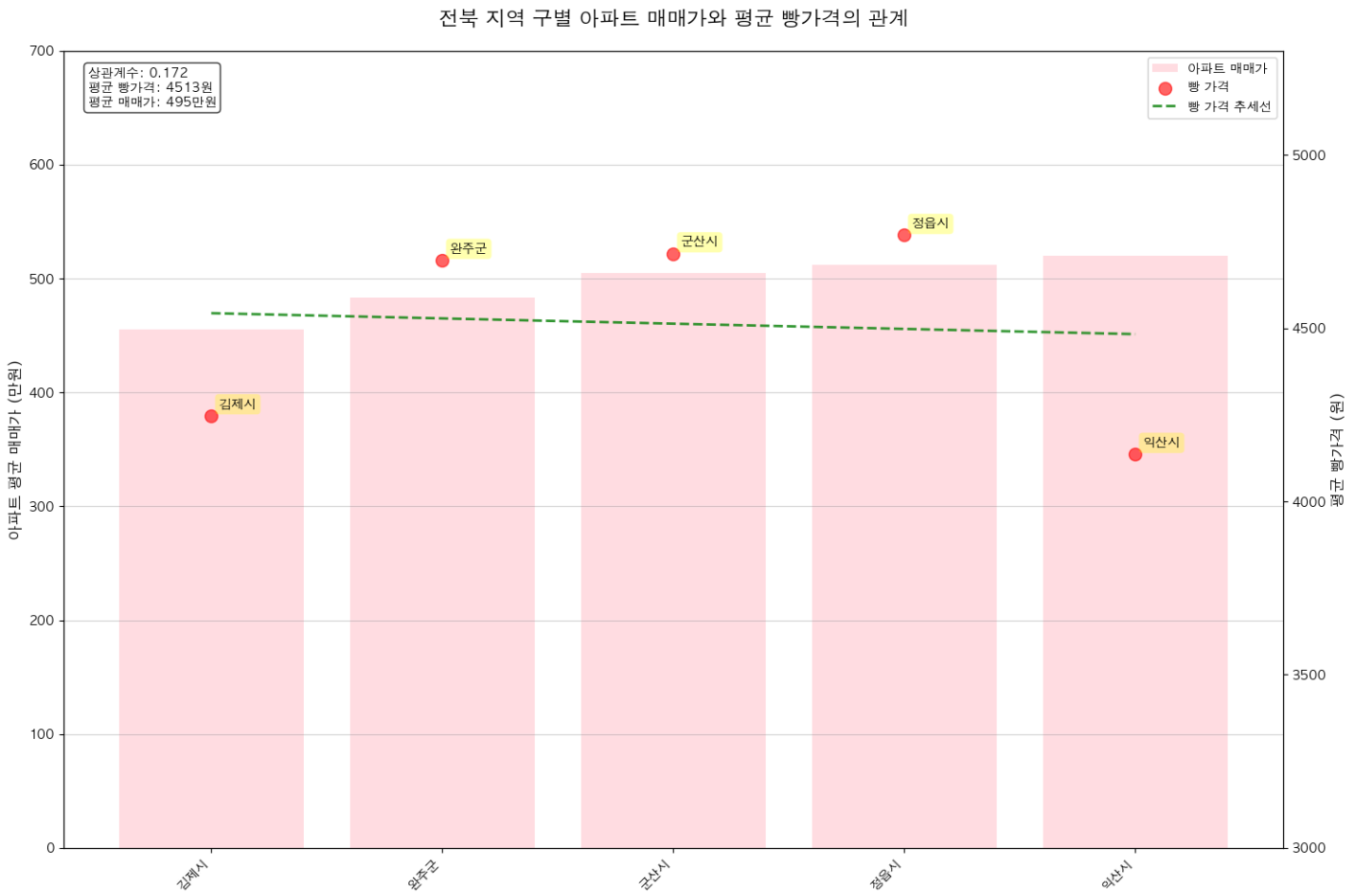
```
전처리 후 구별 빵가격 데이터:
    구분        평균_빵가격
0  군산시  4715.175000
1  김제시  4248.014286
2  완주군  4696.150000
3  익산시  4136.507143
4  전주시  4156.175000
5  정읍시  4768.137500

전처리 후 아파트 가격 데이터:
       구분   매매   전세
0    고창군  567  356
1    군산시  505  408
2    김제시  455  319
3    남원시  507  339
4    무주군  411  286
5    부안군  585  383
6    순창군  384  251
7    완주군  483  385
8    익산시  520  429
10   정읍시  512  355
11   진안군  377  219

병합된 데이터:
    구분        평균_빵가격   매매
0  군산시  4715.175000  505
1  김제시  4248.014286  455
2  완주군  4696.150000  483
3  익산시  4136.507143  520
4  정읍시  4768.137500  512

상관계수: 0.172
양의 상관관계가 있습니다: 아파트 가격이 높은 구일수록 빵 가격도 높은 경향이 있습니다.
```




In [22]:

```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'] = (15, 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=(15, 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('전북 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')
    ax1.set_ylim(0, 700)
    ax2.set_ylim(3000, 5300)

    # 여백 조정
    plt.tight_layout()
```

```python
    # 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  익산시  520  4136.507143
4  정읍시  512  4768.137500
0  군산시  505  4715.175000
2  완주군  483  4696.150000
1  김제시  455  4248.014286

빵 가격 상위 5개 지역:
    구분   매매       평균_빵가격
4  정읍시  512  4768.137500
0  군산시  505  4715.175000
2  완주군  483  4696.150000
1  김제시  455  4248.014286
3  익산시  520  4136.507143

순위 차이가 가장 큰 5개 지역 (불일치도가 높은 지역):
    구분   매매       평균_빵가격  순위_차이
3  익산시  520  4136.507143    4.0
0  군산시  505  4715.175000    1.0
1  김제시  455  4248.014286    1.0
2  완주군  483  4696.150000    1.0
4  정읍시  512  4768.137500    1.0

상관계수: 0.172
양의 상관관계가 있습니다: 아파트 가격이 높은 지역일수록 빵 가격도 높은 경향이 있습니다.
```