1. 파일 불러오기
2. 데이터 확인
3. 데이터 전처리
4. 연령별 데이터 분할
5. 각 등급을 위한 추천운동 딕셔너리(연령별로)
   5-1. 추천 운동 랜덤 반환
6. 연령만 고려한 추천운동 리스트
   6-1. 추천 운동 랜덤 반환
7. 이름 및 나이 입력 받고 조건에 맞는 추천 운동 반환
8. 운동 순서 고려 시 사용할 코드

In [ ]:
```
# 모듈 임포트
import pandas as pd

#파일 불러오기
df = pd.read_csv('c:/mid_project/국민연령별추천운동정보/KS_MRFN_AGE_ACCTO_RECOMMEND_SPORTS_INFO_202303.csv')

df
```

Out[ ]:

	AGRDE_FLAG_NM	BMI_IDEX_GRAD_NM	MBER_SEXDSTN_FLAG_CD	COAW_FLAG_NM	SPORTS_STEP_NM	FLAG_ACCTO_R
0	10대	정상	F	참가증	본운동	
1	10대	정상	F	참가증	본운동	
2	10대	정상	F	참가증	본운동	
3	10대	정상	F	참가증	본운동	
4	10대	정상	F	참가증	본운동	
...	...	...	...	...	...	
5035	70대 이상	3단계비만	M	3등급	마무리운동	
5036	70대 이상	3단계비만	M	3등급	마무리운동	
5037	70대 이상	3단계비만	M	3등급	마무리운동	
5038	70대 이상	3단계비만	M	3등급	마무리운동	
5039	70대 이상	3단계비만	M	3등급	마무리운동	

5040 rows × 7 columns

데이터 확인

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

Out[ ]:

	AGRDE_FLAG_NM	BMI_IDEX_GRAD_NM	MBER_SEXDSTN_FLAG_CD	COAW_FLAG_NM	SPORTS_STEP_NM	FLAG_ACCTO_REC
0	10대	정상	F	참가증	본운동	
1	10대	정상	F	참가증	본운동	
2	10대	정상	F	참가증	본운동	
3	10대	정상	F	참가증	본운동	
4	10대	정상	F	참가증	본운동	

In [ ]:
```
df.isna().sum()  # 결측치 확인
```

Out[ ]:
```
AGRDE_FLAG_NM                  0
BMI_IDEX_GRAD_NM               0
MBER_SEXDSTN_FLAG_CD           0
COAW_FLAG_NM                   0
SPORTS_STEP_NM                 0
FLAG_ACCTO_RECOMEND_MVM_RANK_CO    0
RECOMEND_MVM_NM                0
dtype: int64
```

데이터 전처리

In [ ]:
```
# 등급, 운동순서, 추천운동 컬럼 제외 전부 삭제
df.drop(['FLAG_ACCTO_RECOMEND_MVM_RANK_CO','BMI_IDEX_GRAD_NM', 'MBER_SEXDSTN_FLAG_CD'], axis = 1, inplace = Tru
```

In [ ]:
```
# 등급표 인코딩하기
df['COAW_FLAG_NM'] = df['COAW_FLAG_NM'].map({'참가증':4, '3등급':3, '2등급':2, '1등급':1})
```

연령별 데이터 분할

나이 나누기 기준

- 유소년: 10대 데이터 사용
- 청소년: 10대
- 성인: 20대~50대
- 노인: 원 데이터 기준 노인 데이터의 최소 나이가 61세임을 고려 ==> 60대, 70대 이상

In [ ]:
```
child = df[df['AGRDE_FLAG_NM'] == '10대']
teens = df[df['AGRDE_FLAG_NM'] == '10대']
adult = df[df['AGRDE_FLAG_NM'].str.contains('20대|30대|40대|50대')]
elders = df[df['AGRDE_FLAG_NM'].str.contains('60대|70대 이상')]
```

In [ ]:
```
# 연령대별 준비/본/마무리 데이터 분할
child_pre = child[child['SPORTS_STEP_NM'] == '준비운동']
child_main = child[child['SPORTS_STEP_NM'] == '본운동']
child_last = child[child['SPORTS_STEP_NM'] == '마무리운동']

teens_pre = teens[teens['SPORTS_STEP_NM'] == '준비운동']
teens_main = teens[teens['SPORTS_STEP_NM'] == '본운동']
teens_last = teens[teens['SPORTS_STEP_NM'] == '마무리운동']

adult_pre = adult[adult['SPORTS_STEP_NM'] == '준비운동']
adult_main = adult[adult['SPORTS_STEP_NM'] == '본운동']
adult_last = adult[adult['SPORTS_STEP_NM'] == '마무리운동']

elders_pre = elders[elders['SPORTS_STEP_NM'] == '준비운동']
elders_main = elders[elders['SPORTS_STEP_NM'] == '본운동']
elders_last = elders[elders['SPORTS_STEP_NM'] == '마무리운동']
```

어떤 기준으로 운동을 추천할 것인가?

- `연령 데이터별 등급 기준으로 추천운동 최다 빈도 상위 n개 추천`
- 추천운동 빈도 수를 내림차순 정렬했을 때 상위 2개 요소는 빈도수가 다음 순위보다 높음
  `상위 2개는 고정 출력/나머지 n개 랜덤 출력`
- 준비운동-본운동-마무리운동 나눠서?

각 등급을 위한 추천운동 딕셔너리

- 연령별 데이터 사용
- 추천 운동 중 빈도수가 높은 상위 20개

In [ ]:
```
# 등급별로 추천 운동 상위 20개 종목 리스트로 반환하는 함수
def top_recommend_movements(data, coaw_flag_num, top_n=20):
    filtered_data = data[data['COAW_FLAG_NM'] == coaw_flag_num]
    top_movements = filtered_data['RECOMEND_MVM_NM'].value_counts().sort_values(ascending=False).head(top_n)
    return top_movements.index.tolist()
```

In [ ]:
```
# 데이터에 top_recommend_movements 함수를 적용해 딕셔너리로 저장하는 함수
def grade_top_recommend_movements(data, grade_name):
    grade_data = {}
    for i in range(1, 5):
        grade_data[f'{grade_name}_{i}'] = top_recommend_movements(data, i)
    return grade_data
```

In [ ]:
```
# 등급별로 추천 운동 상위 20개 종목 리스트로 반환하는 함수
def top_recommend_movements(data, coaw_flag_num, top_n=20):
    filtered_data = data[data['COAW_FLAG_NM'] == coaw_flag_num]
    top_movements = filtered_data['RECOMEND_MVM_NM'].value_counts().sort_values(ascending=False).head(top_n)
    return top_movements.index.tolist()

# 데이터에 top_recommend_movements 함수를 적용해 딕셔너리로 저장하는 함수
def grade_top_recommend_movements(data, grade_name):
    grade_data = {}
    for i in range(1, 5):
        grade_data[f'{grade_name}_{i}'] = top_recommend_movements(data, i)
    return grade_data
```

```
# 각 등급을 위한 추천운동 딕셔너리(연령 데이터별)
child_grade_movements = grade_top_recommend_movements(child, 'child')
teens_grade_movements = grade_top_recommend_movements(teens, 'teens')
adult_grade_movements = grade_top_recommend_movements(adult, 'adult')
elders_grade_movements = grade_top_recommend_movements(elders, 'elders')
```

In [ ]:
```
# child_grade_movements
child_grade_movements['child_1']  # 유소년 1등급을 위한 추천 운동 20개
```

Out[ ]:
```
['정적 스트레칭  루틴프로그램',
 '동적 스트레칭 루틴프로그램',
 '줄넘기 운동',
 '몸통 비틀기',
 '복식호흡',
 '맨몸운동  루틴프로그램',
 '온 몸 뻗어주기',
 '버피 테스트',
 '짝 스트레칭  루틴프로그램',
 '목 돌리기',
 '조깅',
 '팔 벌려 뛰기',
 '자전거타기',
 '양발 벌려 무릎 밀어내기',
 '시소우',
 '전완대고 버티기',
 '앉았다 일어서기',
 '팔꿈치 원 그리기',
 '모관운동',
 '요가 및 필라테스  루틴프로그램']
```

등급별 추천운동 5개 반환(상위 2개 고정, 나머지 3개 랜덤)

- 연령별 데이터 사용

In [ ]:
```
# 무작위로 추천운동을 반환하는 함수(상위 2개 고정, 나머지 3개 랜덤)
import random

def get_random_elements(original_list, num_fixed=2, num_random=3):
    # 첫 번째와 두 번째 요소를 추출하여 고정
    fixed_elements = original_list[:num_fixed]

    # 나머지 요소 중에서 랜덤하게 선택
    random_elements = random.sample(original_list[num_fixed:], num_random)

    # 고정된 요소와 랜덤한 요소를 결합하여 새로운 리스트 생성
    result_list = fixed_elements + random_elements

    return result_list
```

In [ ]:
```
# 무작위로 추천운동을 반환하는 함수(상위 2개 고정, 나머지 3개 랜덤) 사용 예
# 유소년 데이터 사용

# 1등급
child_grade1_random = get_random_elements(child_grade_movements['child_1'])
print('유소년 1등급 랜덤 운동추천')
print(child_grade1_random)
print('')

# 2등급
child_grade2_random = get_random_elements(child_grade_movements['child_2'])
print('유소년 2등급 랜덤 운동추천')
print(child_grade2_random)
print('')

# 3등급
child_grade3_random = get_random_elements(child_grade_movements['child_3'])
print('유소년 3등급 랜덤 운동추천')
print(child_grade3_random)
print('')

# 4등급
child_grade4_random = get_random_elements(child_grade_movements['child_4'])
print('유소년 4등급 랜덤 운동추천')
print(child_grade4_random)
print('')
```

```
유소년 1등급 랜덤 운동추천
['정적 스트레칭  루틴프로그램', '동적 스트레칭 루틴프로그램', '전완대고 버티기', '짝 스트레칭  루틴프로그램', '모관운동']

유소년 2등급 랜덤 운동추천
['정적 스트레칭  루틴프로그램', '동적 스트레칭 루틴프로그램', '조깅', '목 돌리기', '짝 스트레칭  루틴프로그램']

유소년 3등급 랜덤 운동추천
['정적 스트레칭  루틴프로그램', '동적 스트레칭 루틴프로그램', '몸통 비틀기', '버피 테스트', '목 돌리기']

유소년 4등급 랜덤 운동추천
['동적 스트레칭 루틴프로그램', '정적 스트레칭  루틴프로그램', '양발 벌려 무릎 밀어내기', '조깅', '버피 테스트']
```

In [ ]:
```python
# 무작위로 추천운동을 반환하는 함수(상위 2개 고정, 나머지 3개 랜덤) 사용 예

# 1등급
child_grade1_random = get_random_elements(child_grade_movements['child_1'])
teens_grade1_random = get_random_elements(teens_grade_movements['teens_1'])
adult_grade1_random = get_random_elements(adult_grade_movements['adult_1'])
elders_grade1_random = get_random_elements(elders_grade_movements['elders_1'])
# 2등급
child_grade2_random = get_random_elements(child_grade_movements['child_2'])
teens_grade2_random = get_random_elements(teens_grade_movements['teens_2'])
adult_grade2_random = get_random_elements(adult_grade_movements['adult_2'])
elders_grade2_random = get_random_elements(elders_grade_movements['elders_2'])

# 3등급
child_grade3_random = get_random_elements(child_grade_movements['child_3'])
teens_grade3_random = get_random_elements(teens_grade_movements['teens_3'])
adult_grade3_random = get_random_elements(adult_grade_movements['adult_3'])
elders_grade3_random = get_random_elements(elders_grade_movements['elders_3'])

# 4등급
child_grade4_random = get_random_elements(child_grade_movements['child_4'])
teens_grade4_random = get_random_elements(teens_grade_movements['teens_4'])
adult_grade4_random = get_random_elements(adult_grade_movements['adult_4'])
elders_grade4_random = get_random_elements(elders_grade_movements['elders_4'])
```

최종

In [ ]:
```python
import random

# 사용자 정보 입력받기
name = input('이름을 입력하세요: ')
age = int(input('나이를 입력하세요: '))
grade = int(input('등급을 입력하세요(참가증은 4를 입력해주세요.): '))

# 연령 및 등급에 따라 적절한 운동 추천 리스트 선택
if 11 <= age <= 12:
        selected_recommendations = child_grade_movements[f'child_{grade}']

if 13 <= age <= 19:
        selected_recommendations = teens_grade_movements[f'teens_{grade}']

if 20 <= age <= 59:
        selected_recommendations = adult_grade_movements[f'adult_{grade}']


if age >= 60:
        selected_recommendations = elders_grade_movements[f'elders_{grade}']


# 첫 번째와 두 번째 요소 고정 추출 및 나머지 세 개의 요소 랜덤 선택하는 함수
def get_random_elements(recommendations):
    fixed_elements = recommendations[:2]
    random_elements = random.sample(recommendations[2:], 3)
    return fixed_elements + random_elements

# 랜덤 추천 생성 및 출력
random_result = get_random_elements(selected_recommendations)
print(f'{name}님({age}세/{grade}등급) 추천 운동 리스트')
print(random_result)
```

In [ ]:
```python
#import random

# 사용자 정보 입력받기
#name = input('이름을 입력하세요: ')
#age = int(input('나이를 입력하세요: '))
#grade = int(input('등급을 입력하세요(참가증은 4를 입력해주세요.): '))

# 연령 및 등급에 따라 적절한 운동 추천 리스트 선택
```

```
#if 11 <= age <= 12:
    #if grade == 1:
        #selected_recommendations = child_grade_movements['child_1']
    #elif grade == 2:
        #selected_recommendations = child_grade_movements['child_2']
    #elif grade == 3:
        #selected_recommendations = child_grade_movements['child_3']
    #else:
        #selected_recommendations = child_grade_movements['child_4']

#if 13 <= age <= 19:
    #if grade == 1:
        #selected_recommendations = teens_grade_movements['teens_1']
    #elif grade == 2:
        #selected_recommendations = teens_grade_movements['teens_2']
    #elif grade == 3:
        #selected_recommendations = teens_grade_movements['teens_3']
    #else:
        #selected_recommendations = teens_grade_movements['teens_2']

#if 20 <= age <= 59:
    #if grade == 1:
        #selected_recommendations = adult_grade_movements['adult_1']
    #elif grade == 2:
        #selected_recommendations = adult_grade_movements['adult_2']
    #elif grade == 3:
        #selected_recommendations = adult_grade_movements['adult_3']
    #else:
        #selected_recommendations = adult_grade_movements['adult_4']

#if age >= 60:
    #if grade == 1:
        #selected_recommendations = elders_grade_movements['elders_1']
    #elif grade == 2:
        #selected_recommendations = elders_grade_movements['elders_2']
    #elif grade == 3:
        #selected_recommendations = elders_grade_movements['elders_3']
    #else:
        #selected_recommendations = elders_grade_movements['elders_4']

# 첫 번째와 두 번째 요소 고정 추출 및 나머지 세 개의 요소 랜덤 선택하는 함수
#def get_random_elements(recommendations):
    #fixed_elements = recommendations[:2]
    #random_elements = random.sample(recommendations[2:], 3)
    #return fixed_elements + random_elements

# 랜덤 추천 생성 및 출력
#random_result = get_random_elements(selected_recommendations)
#print(f'{name}님({age}세/{grade}등급) 추천 운동 리스트')
#print(random_result)
```

```
김멋사님(25세/3등급) 추천 운동 리스트
['짐볼을 이용한 동적 루틴 스트레칭', '유산소 운동 전 동적 루틴 스트레칭', '옆구리 스트레칭', '다리 벌려 앞으로 상체 숙이기', '한 발 앞으로 내밀고 앉았다 일어서기']
```

연령만 고려한 추천운동 리스트

- 등급, 운동순서 고려 X

In [ ]:

```
# 유소년 전체 추천운동 상위 20개 리스트(등급 및 운동순서 고려 X)
child_recommend = child['RECOMEND_MVM_NM'].explode().value_counts().sort_values(ascending=False).head(20).index
print('유소년 전체 추천운동 상위 20개 리스트')
print(child_recommend)
print('')

# 청소년 전체 추천운동 상위 20개 리스트(등급 및 운동순서 고려 X)
teens_recommend =  teens['RECOMEND_MVM_NM'].explode().value_counts().sort_values(ascending=False).head(20).inde
print('청소년 전체 추천운동 상위 20개 리스트')
print(teens_recommend)
print('')

# 성인 전체 추천운동 상위 20개 리스트(등급 및 운동순서 고려 X)
adult_recommend =  adult['RECOMEND_MVM_NM'].explode().value_counts().sort_values(ascending=False).head(20).inde
print('성인 전체 추천운동 상위 20개 리스트')
print(adult_recommend)
print('')

# 노인 전체 추천운동 상위 20개 리스트(등급 및 운동순서 고려 X)
elders_recommend =  elders['RECOMEND_MVM_NM'].explode().value_counts().sort_values(ascending=False).head(20).in
print('노인 전체 추천운동 상위 20개 리스트')
```

```
print(elders_recommend)
```

```
유소년 전체 추천운동 상위 20개 리스트
['정적 스트레칭  루틴프로그램', '동적 스트레칭 루틴프로그램', '온 몸 뻗어주기', '복식호흡', '줄넘기 운동', '맨몸운동  루틴프로그램', '목 돌리기', '버피 테스트', '조깅', '몸통 비틀기', '짝 스트레칭  루틴프로그램', '자전거타기', '양발 벌려 무릎 밀어내기', '팔 벌려 뛰기', '시소우', '모관운동', '팔꿈치 원 그리기', '전완대고 버티기', '다리모아 무릎 누르기', '수영']

청소년 전체 추천운동 상위 20개 리스트
['정적 스트레칭  루틴프로그램', '동적 스트레칭 루틴프로그램', '온 몸 뻗어주기', '복식호흡', '줄넘기 운동', '맨몸운동  루틴프로그램', '목 돌리기', '버피 테스트', '조깅', '몸통 비틀기', '짝 스트레칭  루틴프로그램', '자전거타기', '양발 벌려 무릎 밀어내기', '팔 벌려 뛰기', '시소우', '모관운동', '팔꿈치 원 그리기', '전완대고 버티기', '다리모아 무릎 누르기', '수영']

성인 전체 추천운동 상위 20개 리스트
['유산소 운동 전 동적 루틴 스트레칭', '버피운동', '달리기', '실내 자전거타기', '다리 벌려 앞으로 상체 숙이기', '배스트레칭', '좌식생활자를 위한 동적 루틴 스트레칭', '앉았다 일어서기', '발목 엎고 다리 잡아당기기', '자가근막이완술 루틴 스트레칭', '트레드밀에서 걷기', '다리 벌려 옆으로 상체 숙이기', '한발 앞으로 내밀고 앉았다 일어서기', '하지 루틴 스트레칭1', '옆구리 스트레칭', '엎드려 버티기', '다리 모아 상체 숙이기', '손 뻗어 윗몸 일으키기', '실외 자전거타기', '계단 뛰어 오르기']

노인 전체 추천운동 상위 20개 리스트
['엉덩이 스트레칭', '넙다리 안쪽 스트레칭', '넙다리 앞쪽 스트레칭', '엎드려 양팔 및 다리 들어올리기', '몸통 들어올리기', '등/어깨 뒤쪽 스트레칭', '깍지 끼고 상체 숙이기', '벽에서 팔굽혀 펴기', '누워서 전신 뻗기', '고정식 자전거 타기', '물통으로 양팔 들어올리기', '옆구리 스트레칭', '대퇴사두근 스트레칭', '내전근 스트레칭', '발가락 폄근 스트레칭', '배 스트레칭', '몸통 비틀기', '고정식 트레드밀에서 걷기', '실내 자전거타기', '골반 스트레칭2']
```

연령별 추천운동 5개 반환(상위 2개 고정, 나머지 3개 랜덤)

In [ ]:
```
def get_random_elements(original_list, num_fixed=2, num_random=3):
    # 첫 번째와 두 번째 요소를 추출하여 고정
    fixed_elements = original_list[:num_fixed]

    # 나머지 요소 중에서 랜덤하게 선택
    random_elements = random.sample(original_list[num_fixed:], num_random)

    # 고정된 요소와 랜덤한 요소를 결합하여 새로운 리스트 생성
    result_list = fixed_elements + random_elements

    return result_list

# 연령별 랜덤 운동 추천(상위 2개 고정, 나머지 3개 랜덤)
child_random_result = get_random_elements(child_recommend)
print('유소년 랜덤 운동추천')
print(child_random_result)
print('')

teens_random_result = get_random_elements(teens_recommend)
print('청소년 랜덤 운동추천')
print(teens_random_result)
print('')

adult_random_result = get_random_elements(adult_recommend)
print('성인 랜덤 운동추천')
print(adult_random_result)
print('')

elders_random_result = get_random_elements(elders_recommend)
print('노인 랜덤 운동추천')
print(elders_random_result)
print('')
```

```
유소년 랜덤 운동추천
['정적 스트레칭  루틴프로그램', '동적 스트레칭 루틴프로그램', '수영', '맨몸운동  루틴프로그램', '온 몸 뻗어주기']

청소년 랜덤 운동추천
['정적 스트레칭  루틴프로그램', '동적 스트레칭 루틴프로그램', '팔 벌려 뛰기', '양발 벌려 무릎 밀어내기', '모관운동']

성인 랜덤 운동추천
['유산소 운동 전 동적 루틴 스트레칭', '버피운동', '달리기', '배스트레칭', '트레드밀에서 걷기']

노인 랜덤 운동추천
['엉덩이 스트레칭', '넙다리 안쪽 스트레칭', '엎드려 양팔 및 다리 들어올리기', '옆구리 스트레칭', '고정식 자전거 타기']
```

위 코드 반복문으로 간소화

In [ ]:
```
import random

# 첫 번째와 두 번째 요소 고정 추출 및 나머지 세 개의 요소 랜덤 선택하는 함수
def get_random_elements(recommendations):
    fixed_elements = recommendations[:2]
    random_elements = random.sample(recommendations[2:], 3)
    return fixed_elements + random_elements

# 데이터를 담고 있는 딕셔너리
```

```
recommendation_dict = {
    '유소년': child_recommend,
    '청소년': teens_recommend,
    '성인': adult_recommend,
    '노인': elders_recommend
}

# 반복문을 통한 랜덤 추천 생성 및 출력
for group, recommendations in recommendation_dict.items():
    random_result = get_random_elements(recommendations)
    print(f'{group} 랜덤 운동추천')
    print(random_result)
    print('')
```

```
유소년 랜덤 운동추천
['정적 스트레칭  루틴프로그램', '동적 스트레칭 루틴프로그램', '자전거타기', '복식호흡', '수영']

청소년 랜덤 운동추천
['정적 스트레칭  루틴프로그램', '동적 스트레칭 루틴프로그램', '온 몸 뻗어주기', '조깅', '모관운동']

성인 랜덤 운동추천
['유산소 운동 전 동적 루틴 스트레칭', '버피운동', '배스트레칭', '다리 벌려 앞으로 상체 숙이기', '엎드려 버티기']

노인 랜덤 운동추천
['엉덩이 스트레칭', '넙다리 안쪽 스트레칭', '대퇴사두근 스트레칭', '물통으로 양팔 들어올리기', '엎드려 양팔 및 다리 들어올리기']
```

이름 및 나이 입력 받아서 추천 운동 반환

In [ ]:
```
import random

# 첫 번째와 두 번째 요소 고정 추출 및 나머지 세 개의 요소 랜덤 선택하는 함수
def get_random_elements(recommendations):
    fixed_elements = recommendations[:2]
    random_elements = random.sample(recommendations[2:], 3)
    return fixed_elements + random_elements

# 데이터를 담고 있는 딕셔너리
recommendation_dict = {
    '유소년': child_recommend,
    '청소년': teens_recommend,
    '성인': adult_recommend,
    '노인': elders_recommend
}

# 나이 입력 받기
name = input('이름을 입력하세요: ')
age = int(input('나이를 입력하세요: '))

# 나이에 따라 적절한 운동 추천 리스트 선택
if 11 <= age <= 12:
    selected_recommendations = child_recommend
elif 13 <= age <= 19:
    selected_recommendations = teens_recommend
elif 20 <= age <= 59:
    selected_recommendations = adult_recommend
else:
    selected_recommendations = elders_recommend

# 랜덤 추천 생성 및 출력
random_result = get_random_elements(selected_recommendations)
print(f'{name}님({age}세)을 위한 추천 운동 리스트')
print(random_result)
```

```
AI스쿨님(25세)을 위한 추천 운동 리스트
['유산소 운동 전 동적 루틴 스트레칭', '버피운동', '다리 벌려 옆으로 상체 숙이기', '계단 뛰어 오르기', '앉았다 일어서기']
```

운동 순서도 고려할 경우 아래 코드 사용

In [ ]:
```
# 등급 별 추천 운동 상위 20개 종목 리스트로 반환하는 함수
def top_recommend_movements(data, coaw_flag_num, top_n=20):
    filtered_data = data[data['COAW_FLAG_NM'] == coaw_flag_num]
    top_movements = filtered_data['RECOMEND_MVM_NM'].value_counts().sort_values(ascending=False).head(top_n)
    return top_movements.index.tolist()

# 유소년 등급 별 추천 운동 상위 20개 종목 리스트(준비-본-마무리)
child_pre_1st = top_recommend_movements(child_pre, 1)
child_pre_2nd = top_recommend_movements(child_pre, 2)
```

```
child_pre_3rd = top_recommend_movements(child_pre, 3)
child_pre_4th = top_recommend_movements(child_pre, 4)

child_main_1st = top_recommend_movements(child_main, 1)
child_main_2nd = top_recommend_movements(child_main, 2)
child_main_3rd = top_recommend_movements(child_main, 3)
child_main_4th = top_recommend_movements(child_main, 4)

child_pre_1st = top_recommend_movements(child_last, 1)
child_pre_2nd = top_recommend_movements(child_last, 2)
child_pre_3rd = top_recommend_movements(child_last, 3)
child_pre_4th = top_recommend_movements(child_last, 4)

# 청소년 등급 별 추천 운동 상위 20개 종목 리스트(준비-본-마무리)
teens_pre_1st = top_recommend_movements(teens_pre, 1)
teens_pre_2nd = top_recommend_movements(teens_pre, 2)
teens_pre_3rd = top_recommend_movements(teens_pre, 3)
teens_pre_4th = top_recommend_movements(teens_pre, 4)

teens_main_1st = top_recommend_movements(teens_main, 1)
teens_main_2nd = top_recommend_movements(teens_main, 2)
teens_main_3rd = top_recommend_movements(teens_main, 3)
teens_main_4th = top_recommend_movements(teens_main, 4)

teens_last_1st = top_recommend_movements(teens_last, 1)
teens_last_2nd = top_recommend_movements(teens_last, 2)
teens_last_3rd = top_recommend_movements(teens_last, 3)
teens_last_4th = top_recommend_movements(teens_last, 4)

# 성인 등급 별 추천 운동 상위 20개 종목 리스트(준비-본-마무리)
adult_pre_1st = top_recommend_movements(adult_pre, 1)
adult_pre_2nd = top_recommend_movements(adult_pre, 2)
adult_pre_3rd = top_recommend_movements(adult_pre, 3)
adult_pre_4th = top_recommend_movements(adult_pre, 4)

adult_main_1st = top_recommend_movements(adult_main, 1)
adult_main_2nd = top_recommend_movements(adult_main, 2)
adult_main_3rd = top_recommend_movements(adult_main, 3)
adult_main_4th = top_recommend_movements(adult_main, 4)

adult_last_1st = top_recommend_movements(adult_last, 1)
adult_last_2nd = top_recommend_movements(adult_last, 2)
adult_last_3rd = top_recommend_movements(adult_last, 3)
adult_last_4th = top_recommend_movements(adult_last, 4)

# 노인 등급 별 추천 운동 상위 20개 종목 리스트(준비-본-마무리)
elders_pre_1st = top_recommend_movements(elders_pre, 1)
elders_pre_2nd = top_recommend_movements(elders_pre, 2)
elders_pre_3rd = top_recommend_movements(elders_pre, 3)
elders_pre_4th = top_recommend_movements(elders_pre, 4)

elders_main_1st = top_recommend_movements(elders_main, 1)
elders_main_2nd = top_recommend_movements(elders_main, 2)
elders_main_3rd = top_recommend_movements(elders_main, 3)
elders_main_4th = top_recommend_movements(elders_main, 4)

elders_last_1st = top_recommend_movements(elders_last, 1)
elders_last_2nd = top_recommend_movements(elders_last, 2)
elders_last_3rd = top_recommend_movements(elders_last, 3)
elders_last_4th = top_recommend_movements(elders_last, 4)
```

In [ ]:

```
# 위 코드 함수로 고치다가 실패한 코드
#def get_top_recomend_movements_by_grade(data, age_group_col, sports_step_col, coaw_flag_col, top_n=20):
    #results = []

    #for coaw_flag_num in range(1, 5):
        #filtered_data = data[(data[coaw_flag_col] == coaw_flag_num) & (data[age_group_col] == age_group) & (da
        #top_movements = filtered_data['RECOMEND_MVM_NM'].value_counts().head(top_n).index.tolist()
        #results.extend(top_movements)  # extend를 사용하여 이중 리스트가 아닌 단일 리스트에 추가

    #return results

# Example usage
#age_group_col = 'AGRDE_FLAG_NM'
#sports_step_col = 'SPORTS_STEP_NM'
#coaw_flag_col = 'COAW_FLAG_NM'

#age_groups = ['10대']  # 실제 데이터에 따라 수정 필요
#sports_steps = ['본운동']  # 실제 데이터에 따라 수정 필요

#for age_group in age_groups:
    #for sports_step in sports_steps:
        #results = get_top_recomend_movements_by_grade(child, age_group_col, sports_step_col, coaw_flag_col)
```

```
        #print(results)
```