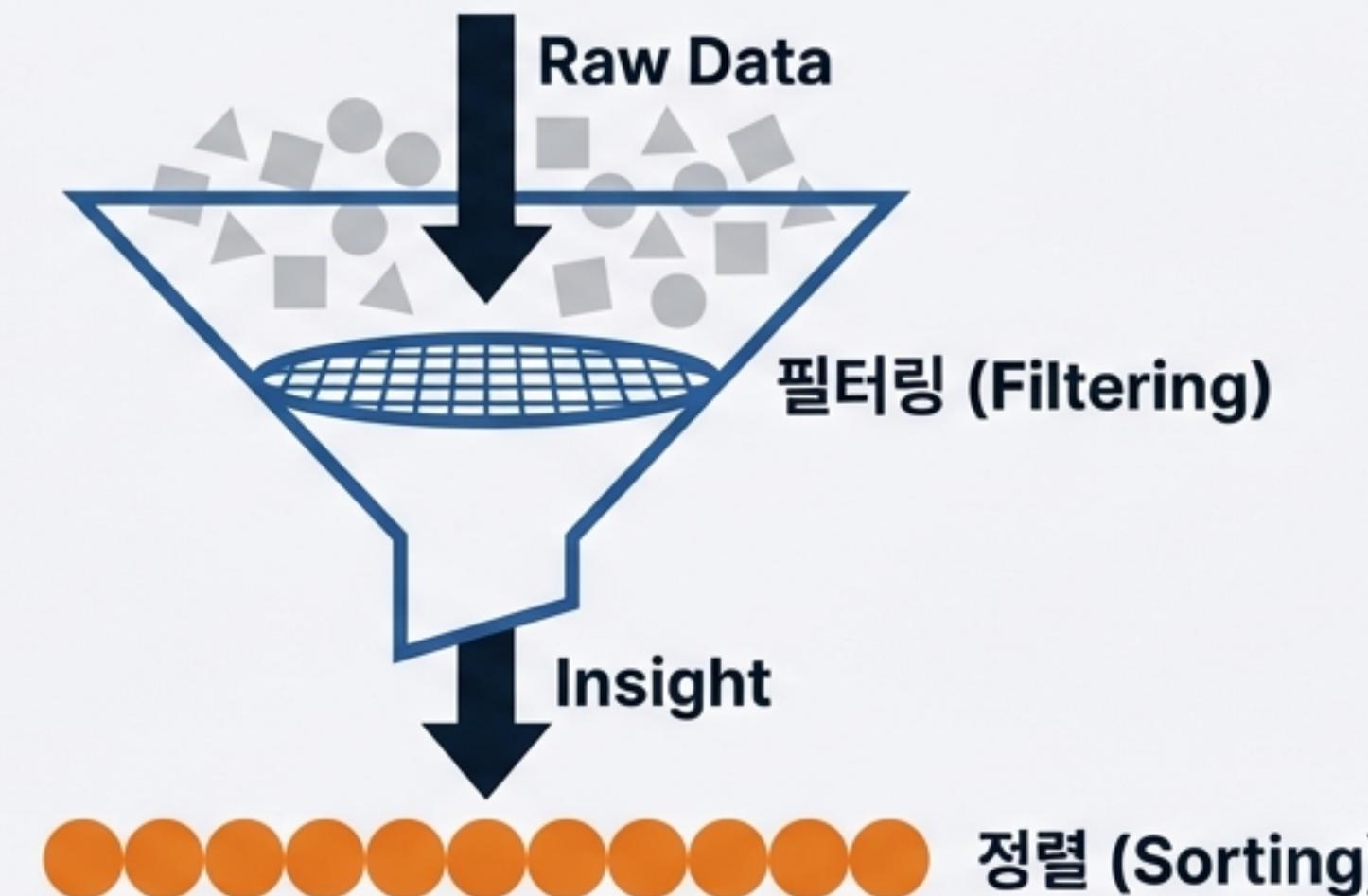


# AI활용프로그래밍 Week 12

# 데이터 필터링과 정렬

조건 검색부터 AI기반 EDA까지, 데이터 분석의 핵심 워크플로우



# 오늘의 학습 목표 (Today's Learning Objectives)



## Boolean Indexing

Pandas DataFrame에서 조건 필터링을 수행할 수 있다.



## Sorting & Ranking

정렬(**sort\_values**)과 상위/하위 추출(**nlargest**)을 사용할 수 있다.



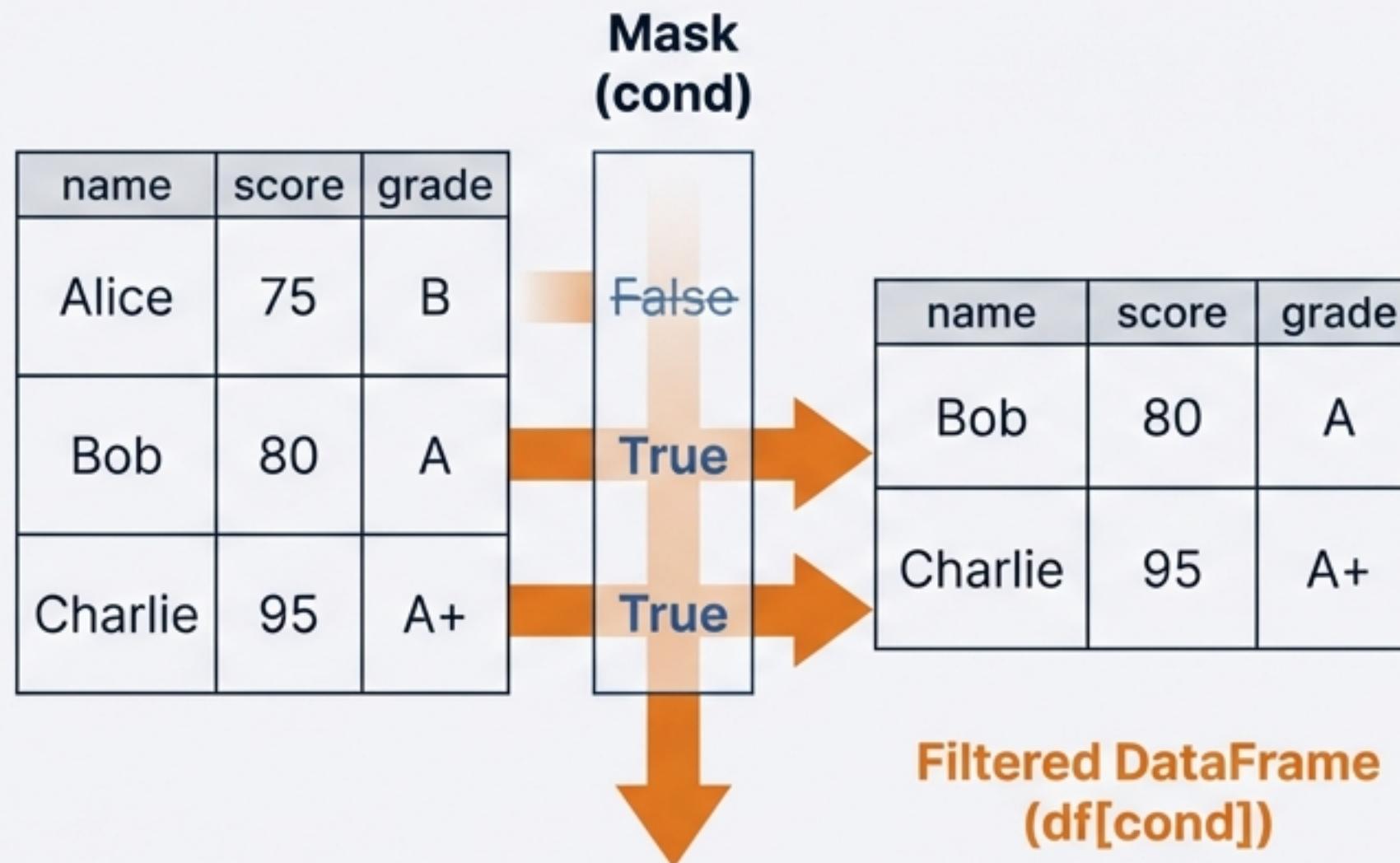
## AI-Assisted EDA

LLM을 활용해 탐색적 분석(EDA) 질문을 만들고 코드를 검증한다.

**Agenda:** Boolean indexing 기본 → 복합 조건(&, |) → 정렬과 Top-N → 결측치와 경고 처리 → **With AI 실습**

# 데이터 분석의 시작: 조건으로 골라내기

**Boolean Indexing이란?** 조건을 변수로 만들어 True인 행만 남기는 것.



```
import pandas as pd
```

```
df = pd.DataFrame({  
    'name': ['Alice', 'Bob', 'Charlie'],  
    'score': [75, 80, 95],  
    'grade': ['B', 'A', 'A+']  
})
```

# 1. 조건 생성

```
cond = df['score'] >= 80
```

1. 조건 생성  
(Create Condition)

# 2. 마스크 적용

```
filtered_df = df[cond]  
print(filtered_df)
```

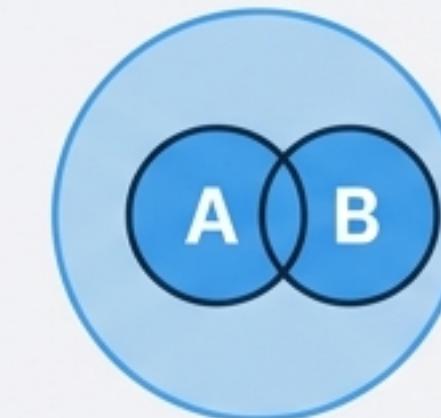
2. 마스크 적용  
(Apply Filter)

# 더 정교한 조건들: isin, between, 문자열

## Specific Group (isin)

```
df['region'].isin(['A', 'B'])
```

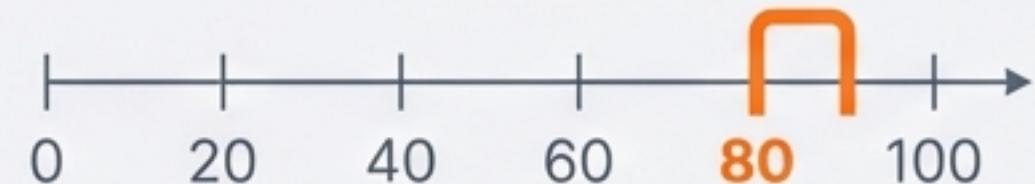
특정 집합에 포함되는 값 선택



## Range (between)

```
df['score'].between(80, 90)
```

구간(경계 포함) 선택



구간(경계 포함) 선택

## Text Handling (str)

```
df['name'].str.contains('k', na=False)
```

문자열 포함 여부 확인



문자열 포함 여부 확인

**Tip:** na=False는 결측치(NaN)가 있어도 에러가 나지 않게 방어해줍니다.

# 복합 조건의 기술: AND(&)와 OR(|)

Python의 기본 연산자와 Pandas의 비트 연산자는 다릅니다.



## ✗ WRONG

```
df['score'] >= 80 and df['score'] < 90
```

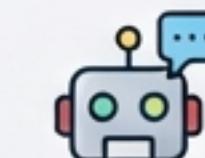
Python의 'and'는 Series(여러 값)를 동시에 처리할 수 없습니다.



## ✓ RIGHT

```
(df['score'] >= 80) & (df['score'] < 90)
```

연산자: & (AND), | (OR) 사용  
괄호 필수: 각 조건은 반드시 ()로 감싸야 함



AI에게 물어보기: '왜 and가 안 되는지  
Series의 True/False 관점에서 설명해줘'

# 데이터 줄세우기: sort\_values



```
df.sort_values('score', ascending=False)
```

- `ascending=False`: 내림차순 (큰 값이 위로)
- `ascending=True`: 오름차순 (작은 값이 위로 - Default)
- 활용: 정렬 후 `.head(n)`을 붙여 상위 데이터만 확인

Caution: `inplace=True`는 원본을 변경하므로 신중하게 사용하세요.

# 극단값 빠르게 추출하기: nlargest / nsmallest

전체 정렬(Sorting) 없이 상/하위 데이터만 빠르게 뽑아내는 효율적인 방법

## Top N (상위 추출)

```
df.nlargest(2, 'score')
```

name	score	group
Dave	90	A
Frank	85	B

가장 큰 값  
N개를 반환

## Bottom N (하위 추출)

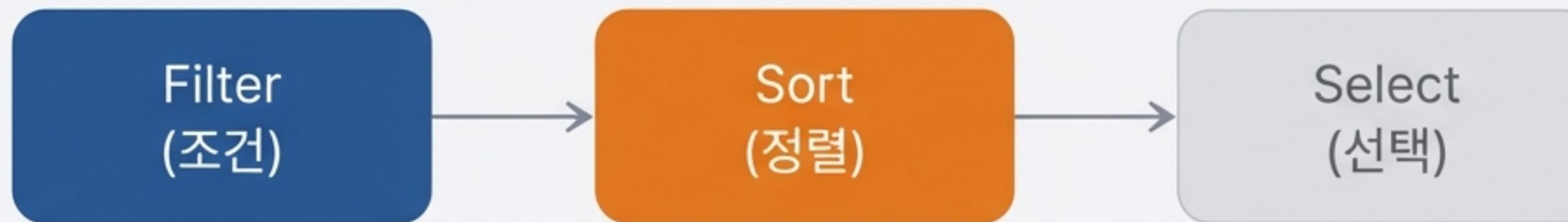
```
df.nsmallest(2, 'score')
```

name	score	group
Bob	70	B
Frank	85	B

가장 작은 값  
N개를 반환

**Why?** 데이터가 클 때 전체를 정렬하는 것보다 훨씬 빠르고 메모리를 적게 사용합니다.

# 필터링과 정렬의 결합: The Pipeline



```
result = (df[df['score'] >= 80]
          .sort_values('score', ascending=False)
          .head(3))
```

} Method Chaining



Line Break Safe Zone

가독성을 위해 괄호 `()`를 사용해 줄바꿈을 하고, 메서드를 체이닝(Chaining)합니다.

# 결측치(NaN) 다루기

데이터의 구멍을 찾고, 지우거나, 채우는 방법

## Detection (확인)

`df.isna()`

A	B	C
1	2	3
2	NaN	3
4	5	6

결측치 여부를  
True/False로 반환

## Removal (제거)

`df.dropna()`

A	B	C
1	2	3
2	NaN	3
4	5	6

결측치가 포함된 행을 삭제

## Replacement (대체)

`df.fillna(0)`

A	B	C
1	2	3
2	0	3
4	5	6

결측치를 특정 값  
(0, 평균 등)으로 채움

# 핀셋처럼 데이터 집어내기: loc, iloc & query

## `loc` [Label Based]

사람이 읽기 쉬운 라벨(이름) 기반

```
df.loc[df['score']>=80, ['name', 'score']]
```

	name	score
0.	Alice	85
1.	Bob	92
3.	David	95
4.	Eve	88

→ 85

## `iloc` [Position Based]

컴퓨터가 읽기 쉬운 정수 위치(Index) 기반

```
df.iloc[0, 1]
```

	0	1	2
0	85		
1			
2			
3			
4			

→ 85

## `query` [String Formula]

SQL처럼 문자열로 조건 작성 (가독성 우수)

```
df.query('score >= 80 and score < 90')
```

	name	score	grade
0	Alice	85	B
4	Eve	88	B

→ 85

# 데이터 무결성 지키기: SettingWithCopyWarning

필터링된 결과는 원본의 'View(보여주기)'일 수도, 'Copy(복사본)'일 수도 있어 모호합니다. 이때 값을 수정하면 경고가 발생합니다.



<ipython-input-1-e428e8377255>:1: SettingWithCopyWarning:  
A value is trying to be set on a copy of a slice from a DataFrame.  
Try using .loc[row\_indexer,col\_indexer] = value instead  
See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)

해결책: `.copy()`를 명시적으로 사용하세요.

```
# Safe
sub = df.loc[df['score'] >= 80].copy()
sub['level'] = 'high' # No warning
```



명확하게 복사본을 만들어 독립적인 DataFrame으로 다루세요.

# LLM을 활용한 스마트한 EDA

AI를 주니어 분석가로 활용하는 프롬프트 템플릿

## Prompt Template

- 역할:** 너는 데이터 분석가다.
- 데이터:** (Paste `df.head()` & `df.info()` here)
- 목표:** 성적 상위 10명을 뽑고 싶어.
- 요청:**
  - 필요한 필터 조건 제안해줘.
  - Pandas 코드를 작성해줘.
  - 결과 해석 및 데이터 품질 체크도 해줘.

## AI Response

네, 분석을 시작하겠습니다.

- 필터 조건:** `score` 컬럼을 기준으로 내림차순 정렬이 필요합니다.

- 코드:**

```
top_10 = df.nlargest(10, 'score')
```

- 해석:** 상위 10명의 평균 점수는...

# 고수들의 코딩 습관 (Pro Coding Habits)



## Check Shape (형상 확인)

필터링 전후에 `df.shape`를 찍어 데이터가 얼마나 줄어들었는지 반드시 확인하세요.



## Refactor to Functions (함수화)

반복되는 논리는 함수로 만들어 재사용성을 높이세요.



## Minimize Inplace (원본 보존)

원본 데이터 변경(`inplace=True`)은 꼭 필요한 경우에만 사용하세요.

Review with AI: ‘내 코드가 더 간결해질 수 있는지(readability)’ 리팩토링을 요청하세요.

# 실전 연습 & 퀴즈 (Practice & Quiz)

## Mini Exercise: Ranking Skeleton

조건(region='A')으로 필터 후, score 기준 내림차순 정렬, Top 10 출력 코드를 완성하세요.

```
# Skeleton Code
result = (df[_____]
           .sort_values(_____)
           .head(__))
```

## Quick Quiz

1. Pandas 복합 조건 연산자는?  
(A) and/or (B) &/|
2. 상위 5개를 뽑는 전용 함수는?  
(A) head (B) nlargest
3. `str.contains` 사용 시 결측치 에러 방지 옵션은?  
(na=False)
4. SettingWithCopyWarning 해결책은?  
(.copy() 사용)

# Week 12 핵심 요약 (Key Takeaways)

## Filter

```
df[cond],  
isin,  
between,  
str.contains,  
query
```

## Logic

& AND  
| OR  
~ NOT

Use Parentheses!

## Sort

```
sort_values(ascending=False)  
nlargest
```

## Select

**loc** Label  
**iloc** Position

## Safety

```
df.copy()  
isna()  
fillna()
```

## Workflow



“조건으로 거르고(Filter), 순서를 정하면(Sort), 인사이트가 보입니다.”