암기할 함수 정리

🧪 빅데이터 분석기사 작업형 1유형 - 핵심 코드 정리

각 함수와 코드는 **실전 출제 빈도순 + 사용 목적** 기준으로 정리함

📂 1. 데이터 로딩 및 구조 파악

```
df = pd.read_excel("data/basic1.xlsx")  # 엑셀 파일 로딩
df.head()  # 상위 5개 행 출력
df.info()  # 데이터 타입, 결측치 등 구조 확인
df.shape  # 행/열 수 확인
```

🖌️ 2. 결측치 처리

```
df.isnull().sum()  # 결측치 개수 확인
df['f1'] = df['f1'].fillna(0)  # 단일값으로 대체
df['f1'] = df['f1'].fillna(df['f1'].mean())  # 전체 평균으로 대체
df['f1'] = df['f1'].fillna(df.groupby('city')['f1'].transform('mean'))  # 그룹별 평균으로 대체
```

📊 3. 조건 필터링 및 집계

```
df[df['f2'] >= 1]  # 조건에 맞는 행 필터링
df.loc[(df['city'] == '서울') & (df['f2'] >= 1), 'f1'].mean()  # 복합 조건 평균
df[df['f4'] == 'ENFJ']['f5'].mean()  # 특정 그룹 평균
```

🔁 4. 그룹화 및 집계 (`groupby`, `agg`, `transform`)

```
df.groupby('city')['f1'].mean()  # city별 f1 평균
df.groupby('f1_grade')['f2'].agg(lambda x: x.value_counts().idxmax())  # 그룹별 최빈값
df['f1_mean'] = df.groupby('city')['f1'].transform('mean')  # 그룹 평균을 각 행에 확장
```

🧬 5. 파생변수 생성

```
df['f1_grade'] = np.where(df['f1'] >= 70, 'high', 'low')  # 조건 기반 파생 컬럼 생성
```

📈 6. 정렬 및 상위 추출

```
df.sort_values('f5', ascending=False)  # 단일 정렬 (내림차순)
df.sort_values(['city', 'f5'], ascending=[True, False])  # 이중 정렬
df.sort_values('f5', ascending=False).head(3)  # 상위 3개 추출
```

📉 7. 이상치 탐지 및 처리 (IQR 기반)

```
q1 = df['f5'].quantile(0.25)
q3 = df['f5'].quantile(0.75)
iqr = q3 - q1
condition = (df['f5'] < q1 - 1.5 * iqr) | (df['f5'] > q3 + 1.5 * iqr)
df.loc[condition, 'f5'] = df['f5'].median()  # 이상치를 중앙값으로 대체
```

🔤 8. 문자열 처리

```
df[df['f4'].str.contains('E')]  # 'E'가 포함된 값 필터링
df['f4'].str.len()  # 문자열 길이 계산
```

🔧 9. 데이터 타입 변환

```
df['f5'] = df['f5'].astype(int)  # 정수형으로 변환
```

📌 10. 기타 유용한 코드

```
df.drop_duplicates(subset=['f4', 'f2'])  # 특정 컬럼 조합 기준 중복 제거
df['f2'].value_counts()  # 값 빈도수 집계
df['f2'].mode()  # 최빈값 (여러 개일 수 있음)
```

Tip:

- `groupby + agg/transform + lambda + idxmax` → 실기 전처리 패턴 핵심
- `sort_values + head`, `fillna + groupby.transform()` → 거의 매회 등장
- 불린 인덱싱 & 조건부 파생변수 생성은 실무 감각 요구

```
# 최빈값이 여러 개 있을 경우 첫 번째만 사용
df['f2'].mode().iloc[0]
```

중요하지만 위에서 빠진 함수 리스트

함수	설명	비고
`df.duplicated()`	중복 여부를 불린 값으로 반환	`drop_duplicates()` 와 함께 사용
`df.reset_index(drop=True)`	정렬 후 인덱스 초기화	`sort_values()` 이후 자주 사용
`df['col'].nunique()`	고유값 개수 반환	`value_counts()` 와 함께 자주 쓰임
`df['col'].unique()`	고유값 리스트 반환	범주형 탐색용
`df['col'].map()`	딕셔너리 기반 값 변환	카테고리 매핑 시 유용
`df.replace()`	값 치환	결측치 외 다른 값 대체 시 사용
`df.corr()`	수치형 간 상관관계 분석	간단한 탐색에 유용
`df['col'].rank()`	순위 계산	정렬/비율 분석에 활용
`df.sample(n=5)`	무작위 샘플 추출	검증용/테스트 데이터 추출 시
`df.columns.str.strip()`	컬럼명 공백 제거	엑셀 import 후 자주 필요
`pd.cut()`	연속형 수치 → 구간 범주형으로 변환	구간 나누기 문제에서 출제 가능

함수	설명	비고
`pd.qcut()`	분위수 기준 구간화	균등 분할 문제 시 유리
`df.eval()`	문자열로 연산 수행	고급 활용. 가독성 좋음