[베이직 반] 시각화

데이터 시각화 (matplotlib, seaborn)

0) 용어정리

▸ 수치형(Numeric)

- **이산형(Discrete)**: 셀 수 있는 정수 값(0,1,2...). 예) 형제/배우자 수(`sibsp`), 자녀/부모 수(`parch`)
- **연속형(Continuous)**: 두 값 사이에 항상 또 다른 값(소수 포함), 실수 범위(소수 포함). 예) 나이(`age`), 요금(`fare`)

> ✔️ 평균·표준편차 계산이 의미 있고, 크기 비교가 자연스러움

▸ 범주형(Categorical)

- **명목형(Nominal)**: 순서 없음. 예) 성별(`sex`), 탑승항(`embarked`)
- **순서형(Ordinal)**: 순서 있음. 예) 객실등급(`pclass`=1/2/3, `class`=First/Second/Third)

> ✔ 평균보다는 빈도/비율 비교가 자연스러움

▸ 분석 차원(변수 개수)

- **단변량**: 변수 1개(분포/요약)
- **이변량**: 변수 2개(관계/차이)
- **다변량**: 3개 이상(층화 비교, 상호작용)

---

1) 그래프 선택 가이드

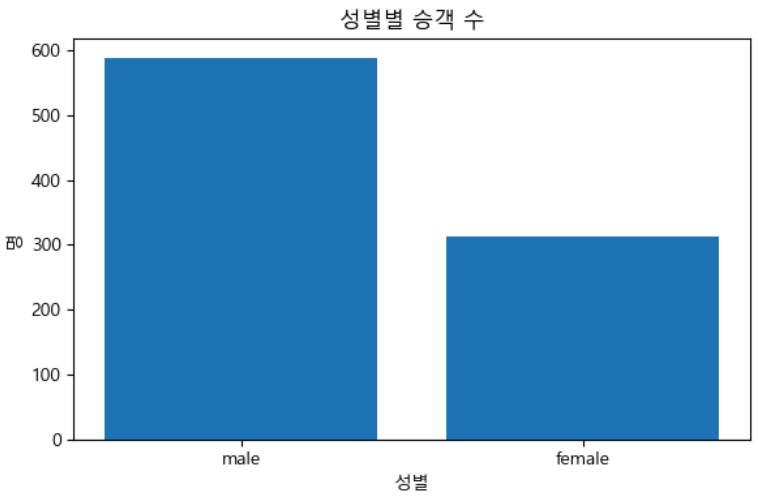
▼ **요약표**

질문(목적)	변수 타입	추천 그래프(Seaborn 중심)	왜 이걸 쓰나?
범주 **빈도/비율**	단변량 **범주형**	`countplot`(개수), `barplot`(평균/비율)	막대 길이로 비교가 가장 직관적
수치 **분포 모양**	단변량 **수치형**	`histplot`(+ `kde=True`)	몰림·꼬리·다봉성 파악
중앙값/이상치	단변량 **수치형**	`boxplot`	중앙값·사분위·이상치 요약
범주별 수치 분포	이변량 **범주×수치**	`boxplot` / `violinplot`	그룹 간 분포·변동 차이 한 번에
범주별 평균/비율	이변량 **범주×수치(0/1)**	`barplot`	0/1 평균=**비율**
수치×수치 관계	이변량 **수치×수치**	`scatterplot`	점 구름으로 상관·패턴·이상치
범주×범주 연관	이변량 **범주×범주**	`heatmap`	색으로 비율/연관 강도 직관적
여러 수치 쌍관계	다변량 **수치 다수**	`pairplot`	전체 구조·이상치 빠르게 파악
여러 범주 쌍관계	다변량 **범주 다수**	`FacetGrid`	여러 범주 조건으로 잘라서 비교

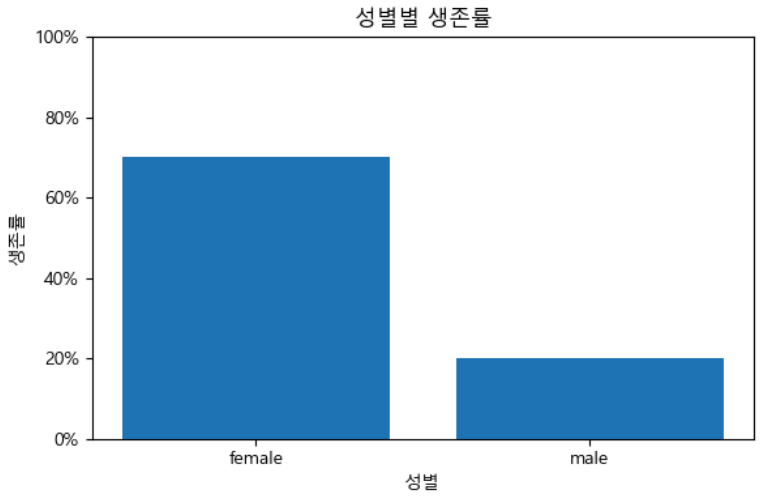
1. 범주 빈도/비율

- 변수 타입 : 단변량 **범주형**
- 추천 그래프(Seaborn) : `countplot`(개수), `barplot`(평균/비율)
- 왜? 막대 길이로 비교가 가장 직관적이다.
- 예시 그래프

▼ `countplot`**(개수)**



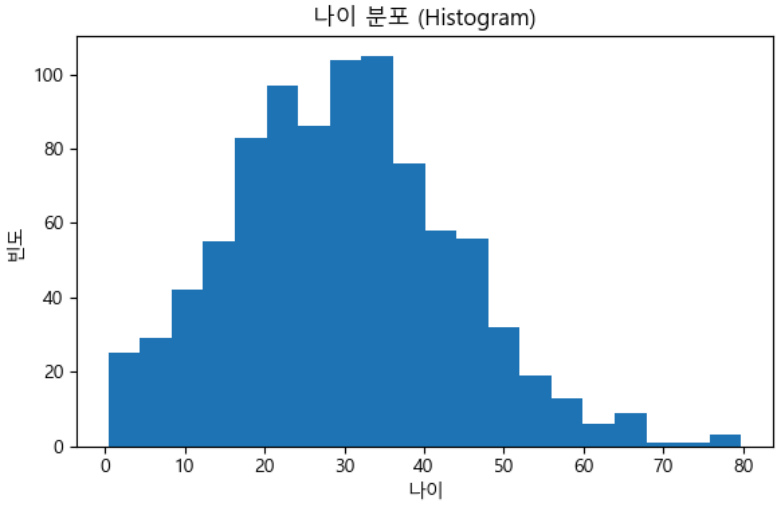
▼ `barplot`**(평균/비율)**



2. 수치 분양 모양

- 변수 타입 : 단변량 **수치형**
- 추천 그래프(Seaborn) : `histplot` (+ `kde=True`)
- 왜? 몰림·꼬리·다봉성 파악하기 쉽다.
- 예시 그래프

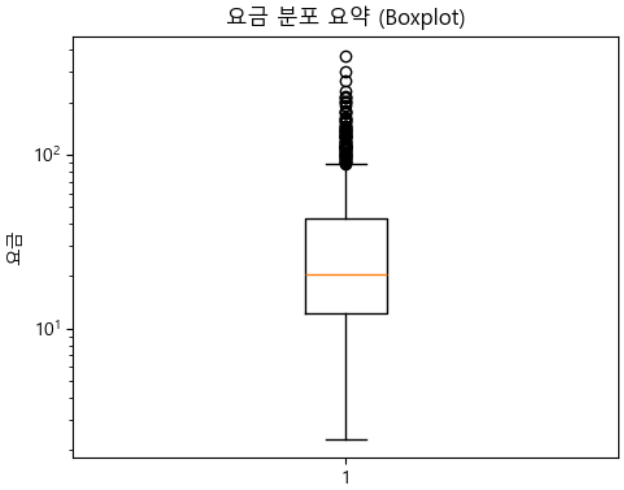
▼ `histplot`



3. 중앙값/이상치

- 변수 타입 : 단변량 **수치형**
- 추천 그래프(Seaborn) : `boxplot`
- 왜? 중앙값·사분위·이상치 요약이 가장 쉽게 알아볼 수 있다.
- 예시 그래프

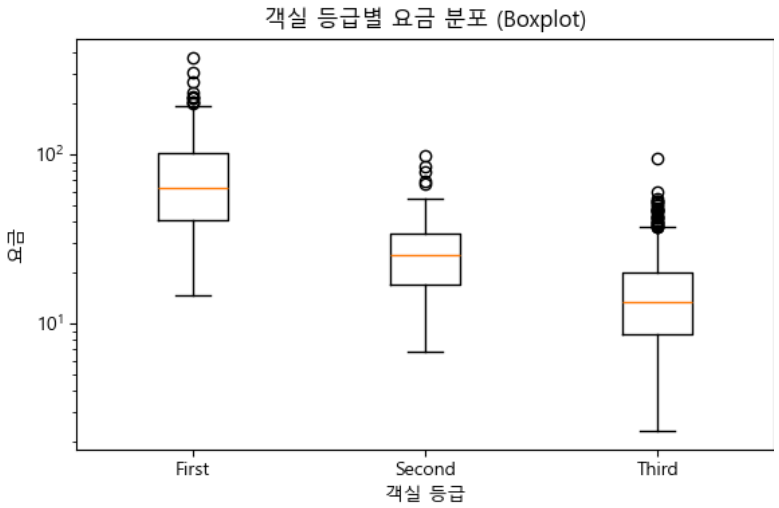
▼ `boxplot`



4. 범주별 수치 분포

- 변수 타입 : 이변량 **범주×수치**
- 추천 그래프(Seaborn) : `boxplot` / `violinplot`
- 왜? 그룹 간 분포·변동 차이 한 번에 확인 가능하다.
- 예시 그래프

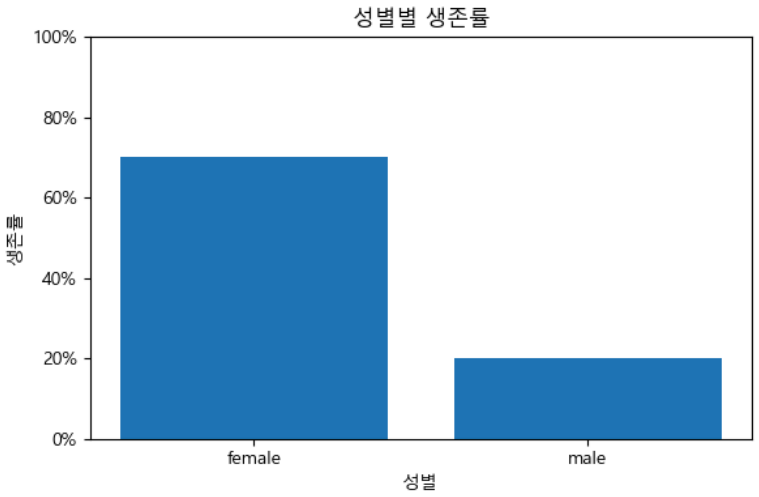
▼ `boxplot`



5. 범주별 평균/비율

- 변수 타입 : 이변량 **범주×수치(0/1)**
- 추천 그래프(Seaborn) : countplot(개수), barplot(평균/비율)
- 왜? 0/1 평균=**비율**이기에 barplot으로 비교가 쉽다.
- 예시 그래프

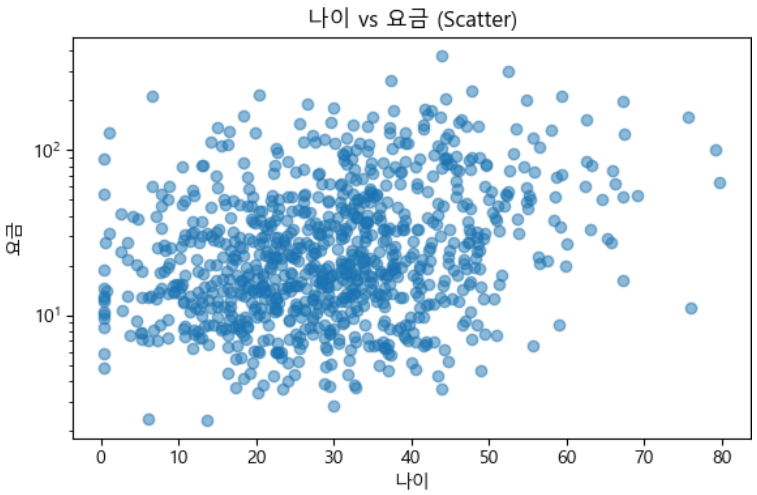
▼ `barplot` **(평균/비율)**



6. 수치×수치 관계

- 변수 타입 : 이변량 **수치×수치**
- 추천 그래프(Seaborn) : scatterplot
- 왜? 점 구름으로 상관·패턴·이상치의 분포를 확인할 수 있다.
- 예시 그래프

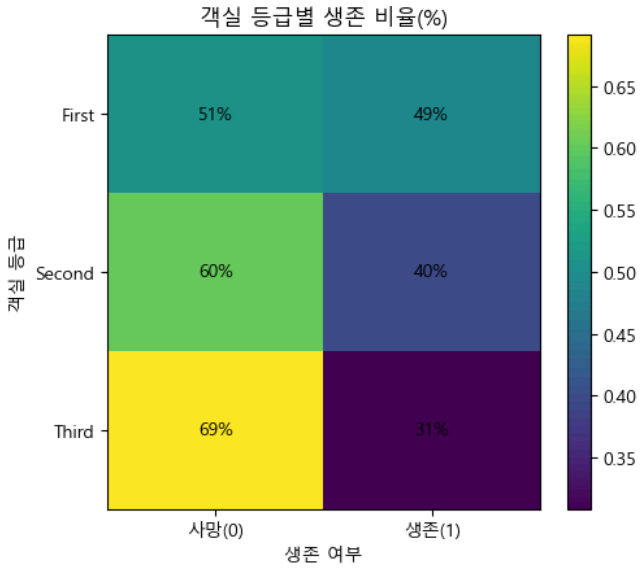
▼ scatterplot



7. 범주×범주 연관

- 변수 타입 : 이변량 **범주×범주**
- 추천 그래프(Seaborn) : heatmap
- 왜? 색으로 비율/연관 강도 직관적이다.
- 예시 그래프

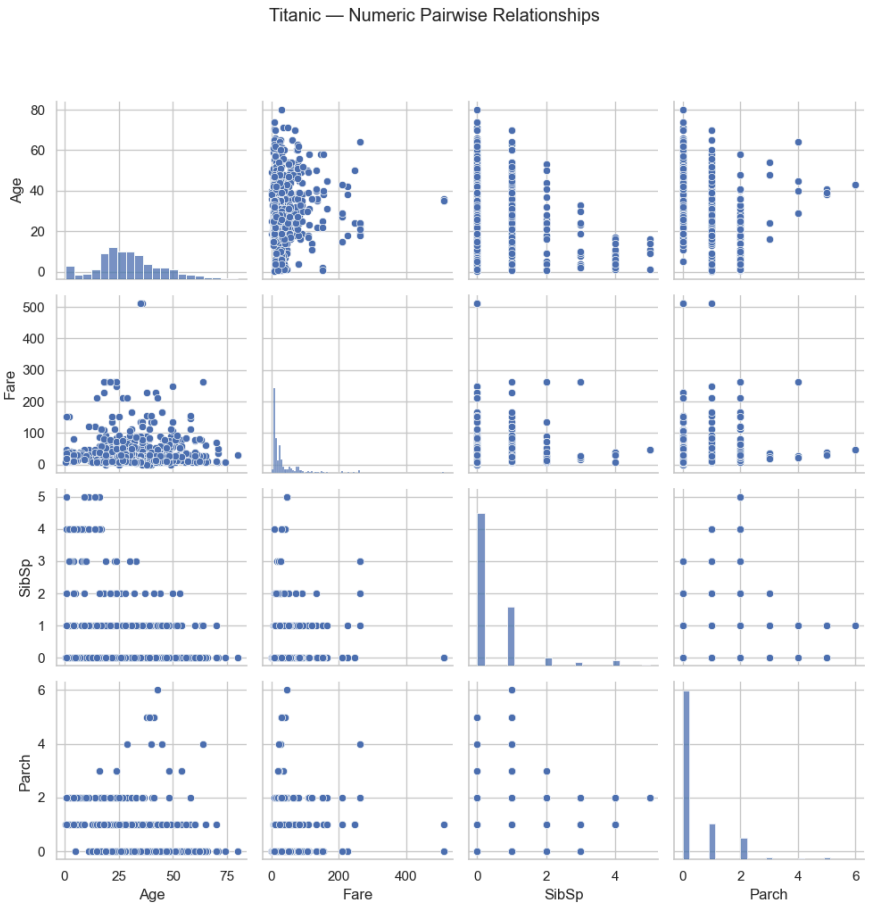
▼ heatmap



8. 여러 수치 쌍관계 (1)

- 변수 타입 : 다변량 **수치 다수**
- 추천 그래프(Seaborn) : pairplot
- 왜? 전체 구조·이상치 빠르게 파악할 수 있다.
- 예시 그래프

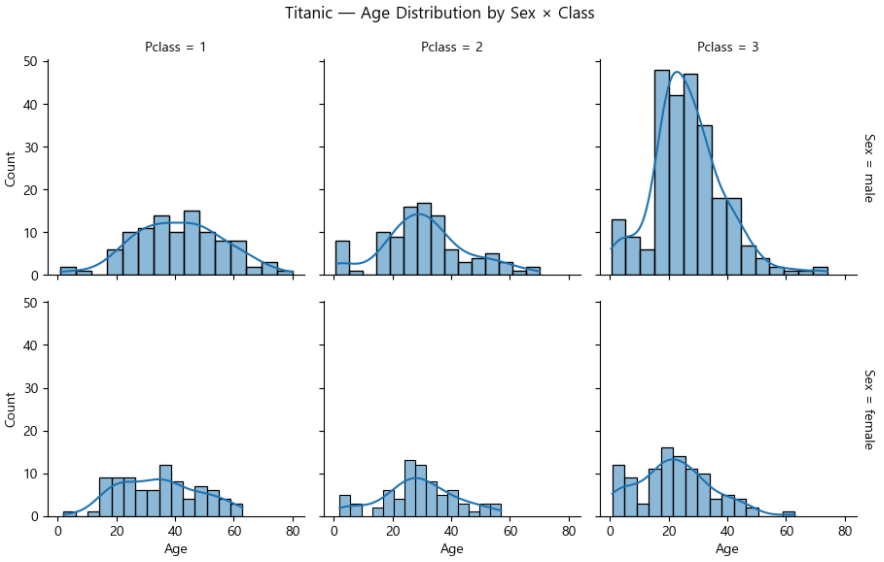
▼ pairplot



9. 여러 범주 쌍관계 (2)

- 변수 타입 : 다변량 **범주 다수**
- 추천 그래프(Seaborn) : `FacetGrid`
- 왜? 여러 범주 조건으로 잘라서 비교할 수 있다.
- 예시 그래프

▼ `FacetGrid`



---

2) 간단 의사결정트리

1. **비교**가 목표인가?
   - 평균/비율 → **barplot** ｜ 분포 → **box/violin**
2. **분포**가 궁금한가?
   - 연속 1개 → **histplot**(+KDE)
3. **관계**가 궁금한가?
   - 연속×연속 → **scatter** ｜ 범주×범주 → **heatmap**
4. **변수 3개 이상**인가?
   - **FacetGrid(범주)** / **pairplot(수치)** 사용

---

3) 왜 Matplotlib & Seaborn인가?

- **Matplotlib**
  - 파이썬에서 가장 기본이 되는 **그래프 라이브러리**
  - 여러 종류의 차트(선형 차트, 막대 차트, 산점도 등)를 직접 그릴 수 있고, 많은 커스터마이징이 가능
- **Seaborn**
  - Matplotlib 기반으로 만들어진 **좀 더 쉬운** 시각화 라이브러리
  - **데이터프레임(Pandas)**과 잘 연결되고, **기본 디자인**도 깔끔하게 설정되어 있어서 **적은 코드**로 **멋진 그래프** 생성

Tip: Matplotlib와 Seaborn은 서로 보완 관계입니다. 처음엔 Seaborn으로 간단히 시작하고, 더 깊은 커스터마이징이 필요할 때 Matplotlib를 함께 쓰면 됩니다.

---

4) 준비하기

1. **파이썬 설치**: 미니콘다(Miniconda)나 아나콘다(Anaconda) 설치를 추천합니다.
2. **Jupyter Notebook(또는 JupyterLab)**: 코드 실행 결과를 바로바로 확인하면서 그래프를 띄울 수 있어 매우 편리합니다.
3. **라이브러리 설치**:

   ```
   pip install matplotlib seaborn pandas
   ```

4. **Notebook에서 그래프 보이기**: 보통

   ```
   %matplotlib inline
   ```

   이 설정을 노트북 맨 위에 써주면, 그림이 노트북 셀 안에 바로 표시됩니다.
5. **한글 깨짐 방지**

```
import matplotlib.pyplot as plt

plt.rcParams['font.family'] = 'Malgun Gothic' # For Windows
# plt.rcParams['font.family'] = 'AppleGothic' # For MacOS
```

- 한글 폰트 설정 문제 → 위와 같이 설정 후 시도

---

5) 첫 그래프 그려보기

1. Matplotlib 예제

```
import matplotlib.pyplot as plt

# 예시 데이터
x = [1, 2, 3, 4]
y = [10, 20, 15, 25]

plt.plot(x, y, marker='o')
plt.title("기본 라인 그래프")
plt.xlabel("X축")
plt.ylabel("Y축")
plt.show()
```

- `plt.plot(x, y)`: 기본 선형 그래프를 그립니다.
- `marker='o'`: 각 점에 동그라미 표시를 추가합니다.
- `plt.title()`, `plt.xlabel()`, `plt.ylabel()`: 그래프 제목과 축 레이블을 설정합니다.
- `plt.show()`: 그래프를 화면에 표시합니다.

2. Seaborn 예제

```
import seaborn as sns
import pandas as pd

# 샘플 데이터프레임
```

```
df = pd.DataFrame({
    'month': ['Jan','Feb','Mar','Apr'],
    'sales': [10, 20, 15, 25]
})

sns.lineplot(x='month', y='sales', data=df)
plt.title("Seaborn 라인 그래프")
plt.show()
```

- `sns.lineplot()` : Seaborn에서 제공하는 선형 그래프 함수.
- `x='month', y='sales'` : 데이터프레임 `df` 의 `month` 컬럼을 x축, `sales` 컬럼을 y축으로 사용.
- **장점**: `data=` 에 데이터프레임을 바로 연결하면 축 레이블도 자동 처리되고, 격자와 색상 등 기본 스타일이 예쁘게 설정됩니다.

---

6) 주요 그래프 유형 맛보기

▼ **요약표**

그래프	언제/목적	변수 타입	Seaborn 권장 함수	Matplotlib 대안	왜 이걸 쓰나	핵심 팁
선형(Line)	시간/순서에 따른 변화 추적	순서형/시계열 vs 수치	`sns.lineplot`	`plt.plot`	추세·증감 확인에 최적	x축 순서(날짜/월) 정렬 필수
막대(빈도)	범주 **개수** 비교	범주	`sns.countplot`	`plt.bar` (집계치로)	막대 길이가 개수를 직관적으로 표현	범주 많으면 상위 N만, 정렬
막대(평균/비율)	그룹 **평균/비율** 비교	범주×수치(0/1 포함)	`sns.barplot`	`plt.bar` (집계 결과로)	0/1 평균=비율 → 막대 = 비율/평균	`%` 포맷, 오차막대는 상황에 따라
산점도	두 수치 간 관계/상관/이상치	수치×수치	`sns.scatterplot`	`plt.scatter`	점 구름으로 패턴·상관·클러스터 파악	과포개 시 `alpha` , 로그 스케일 고려
히스토그램	수치의 분포 모양	수치(연속)	`sns.histplot(kde=True)`	`plt.hist`	몰림·꼬리·다봉성 파악	bin 수 적절히(데이터 크기별 조정)
박스/바이올린	그룹별 분포·중앙값·이상치	범주×수치	`sns.boxplot` / `sns.violinplot`	`plt.boxplot`	평균 막대가 감추는 분포/이상치 확인	꼬리 길면 로그 스케일 고려

1. 선형 그래프 (Line Plot)

언제? 시간(혹은 순서)에 따른 **추세/변화**를 보고 싶을 때.

데이터

```
days  = ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"]
sales = [10, 12,  9,   13,   15,   14,   11]
df = pd.DataFrame({"day": days, "sales": sales})
```

Seaborn

```
sns.lineplot(data=df, x="day", y="sales", marker="o")
plt.title("Daily Sales (Seaborn)")
plt.xlabel("Day")
plt.ylabel("Sales")
plt.tight_layout()
plt.show()
```

Matplotlib

```
plt.plot(days, sales, marker="o")
plt.title("Daily Sales")
plt.xlabel("Day")
```

```
plt.ylabel("Sales")
plt.tight_layout()
plt.show()
```

팁: 날짜형이면 `pd.to_datetime` 후 x축 정렬을 보장하세요.

---

2. 막대 — 빈도(개수) 비교 (countplot)

언제? 범주별 **건수/빈도** 비교.

데이터

```
basket = pd.DataFrame({"fruit": ["Apple","Banana","Apple","Cherry","Apple","Banana"]})
```

Seaborn (권장)

```
order = basket['fruit'].value_counts().index  # 내림차순
sns.countplot(data=basket, x="fruit", order=order)
plt.title("Fruit Count")
plt.xlabel("Fruit")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```

Matplotlib (집계 후)

```
counts = basket["fruit"].value_counts().sort_values(ascending=False)
plt.bar(counts.index, counts.values)
plt.title("Fruit Count")
plt.xlabel("Fruit")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```

주의: **개수**를 볼 땐 `countplot`, **평균/비율**을 볼 땐 `barplot`!

---

3. 막대 — 평균/비율 비교 (barplot)

언제? 그룹 **평균**(연속형) 또는 **비율**(0/1) 비교.

예 A: 평균 비교

```
df = pd.DataFrame({
    "clazz": ["A","A","A","B","B","B","C","C"],
    "score": [80, 75, 70, 88, 85, 90, 78, 76]
})
# errorbar 옵션은 오차막대이고, 95% 신뢰구간를 가진다. None을 줄 경우 값 숨김
sns.barplot(data=df, x="clazz", y="score", errorbar=None)  # 기본: 평균
plt.title("Average Score by Class")
plt.xlabel("Class")
plt.ylabel("Average Score")
plt.tight_layout()
plt.show()
```

예 B: 비율(0/1 평균=비율)

```
df = pd.DataFrame({
    "group": ["G1","G1","G1","G2","G2","G2","G2"],
    "passed": [1, 0, 1, 1, 1, 0, 0]   # 0/1
})
```

```
ax = sns.barplot(data=df, x="group", y="passed", errorbar=None)
plt.title("Pass Rate by Group")
plt.xlabel("Group")
plt.ylabel("Pass Rate")
plt.ylim(0,1)
plt.tight_layout()
plt.show()
```

Matplotlib 대안: 그룹별 평균/비율을 **직접 집계**한 뒤 `plt.bar`로 그리기.

---

4. 산점도 (Scatter Plot)

언제? 두 수치형 변수 사이의 **관계/상관/이상치**.

데이터

```
df = pd.DataFrame({
    "height": [160,165,170,175,180,168,172],
    "weight": [ 55, 60, 65, 72, 80, 58, 67],
    "sex":    ["F","M","M","M","M","F","F"]
})
```

Seaborn

```
sns.scatterplot(data=df, x="height", y="weight", hue="sex")
plt.title("Height vs Weight by Sex")
plt.xlabel("Height (cm)")
plt.ylabel("Weight (kg)")
plt.tight_layout()
plt.show()
```

Matplotlib

- 색을 구분하고 싶다면.. 색 매핑(dict) 해야함.. Seaborn 씁시다

```
plt.scatter(df["height"], df["weight"])
plt.title("Height vs Weight")
plt.xlabel("Height (cm)")
plt.ylabel("Weight (kg)")
plt.tight_layout()
plt.show()
```

팁: 과포개(overplotting) 시 `alpha=0.4`, 점 크기 `s=`, 또는 로그 스케일/hexbin 고려.

---

5. 히스토그램 (Histogram)

언제? 수치형 변수의 **분포 모양**(몰림·꼬리·다봉성).

데이터

```
scores = [55,60,62,65,70,72,75,78,80,82,85,88]
df = pd.DataFrame({"score": scores})
```

Seaborn

```
sns.histplot(data=df, x="score", bins=5, kde=True)
plt.title("Score Distribution (Seaborn)")
plt.xlabel("Score")
plt.ylabel("Count")
```

```
plt.tight_layout()
plt.show()
```

Matplotlib

```
plt.hist(scores, bins=5)
plt.title("Score Distribution")
plt.xlabel("Score")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```

팁: bin 수에 따라 모양이 달라집니다(너무 적거나 많으면 왜곡).

---

6. 박스플롯 & 바이올린 (Box / Violin)

언제? 그룹별 분포의 **중앙값·사분위·이상치** 확인(박스), **분포 형태**까지(바이올린).

데이터

```
df = pd.DataFrame({
    "clazz": ["A","A","A","B","B","B","C","C","C","C"],
    "score": [ 70, 75, 80, 65, 68, 78, 60, 62, 64, 66]
})
```

Seaborn (Box)

```
sns.boxplot(data=df, x="clazz", y="score")
plt.title("Scores by Class (Boxplot)")
plt.xlabel("Class")
plt.ylabel("Score")
plt.tight_layout()
plt.show()
```

Seaborn (Violin)

```
sns.violinplot(data=df, x="clazz", y="score", inner="quartile")
plt.title("Scores by Class (Violin)")
plt.xlabel("Class")
plt.ylabel("Score")
plt.tight_layout()
plt.show()
```

---

7) Titanic 문제 세트

> **공통 준비**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

plt.rcParams["font.family"] = "Malgun Gothic"  # 또는 "AppleGothic", "NanumGothic"
plt.rcParams["axes.unicode_minus"] = False
%matplotlib inline
titanic = pd.read_csv("train.csv")
```

문제 1.

- 질문 : 성별(`sex`)에 따라 **생존률(`survived`)**이 달랐는가?
- 문제 : 성별별 생존률 비교
- 권장 그래프 : 범주×연속[0/1] → 막대그래프 → `barplot`
- **왜 이 그래프?** 이진변수 평균이 곧 **비율**이므로 막대 높이가 바로 생존률을 의미.

```
ax = sns.barplot(data=titanic, x="sex", y="survived", errorbar=None)
ax.set_title("성별별 생존률")
ax.set_xlabel("성별")
ax.set_ylabel("생존률")
plt.ylim(0, 1)
plt.show()
```

- **해석 포인트**: 표본 수(`titanic["Sex"].value_counts()`)도 함께 확인해 신뢰도 판단.

문제 2.

- 질문 : 객실 등급(`class`)에 따라 **요금(`fare`)** 분포는 어떻게 다른가?
- 문제 : 객실 등급별 요금 분포
- 권장 그래프 : 범주×연속 → 박스/바이올린 → `boxplot` 또는 `violinplot`
- **왜 이 그래프?** 평균 막대만 보이면 outlier 영향이 큼. **중앙값·사분위·이상치**까지 한 번에 확인 가능.

```
# botplot
sns.boxplot(data=titanic, x="Pclass", y="Fare")
plt.title("객실 등급별 요금 분포 (Box)")
plt.xlabel("객실 등급")
plt.ylabel("요금")
plt.show()

# violinplot
sns.violinplot(data=titanic, x="Pclass", y="Fare", inner="quartile")
plt.title("객실 등급별 요금 분포 (Violin)")
plt.show()
```

문제 3.

- 질문 : 승객 **나이(`age`)**는 어떤 분포인가?
- 문제 : 나이 분포 파악
- 권장 그래프 : 연속 단변량 → 히스토그램 → `histplot` (+KDE)
- **왜 이 그래프?** 연속형 1개 변수의 **분포 모양**(봉우리, 꼬리, 다봉성) 파악에 최적

```
sns.histplot(data=titanic, x="Age", bins=20, kde=True)
plt.title("나이 분포")
plt.xlabel("나이")
plt.ylabel("빈도")
plt.show()
```

문제 4.

- 질문 : **나이(** `age` **)**와 **요금(** `fare` **)** 사이에 어떤 패턴이 있는가?
- 문제 : **나이 vs 요금** 관계
- 권장 그래프 : 연속×연속 → 산점도 → `scatterplot` (+ `hue="survived"`)
- **왜 이 그래프?** 두 연속형의 **상관/클러스터/이상치**를 점 구름으로 쉽게 확인 가능

```
sns.scatterplot(data=titanic, x="Age", y="Fare", hue="Survived")
plt.title("나이 vs 요금 (생존 여부 색상)")
plt.xlabel("나이")
plt.ylabel("요금")
plt.show()
```

문제 5.

- 질문 : 객실 등급(`class`)에 따라 **생존률**이 어떻게 달랐는가?
- 문제 : 등급×생존 비율
- 권장 그래프 : 범주×범주 → 히트맵 → 교차표(`crosstab`) **행 기준 비율** → `heatmap`
- **왜 이 그래프? 비율**을 색으로 표시하면 등급 간 차이를 **한눈에** 비교 가능.

```
ct = pd.crosstab(titanic["Pclass"], titanic["Survived"], normalize="index") * 100
sns.heatmap(ct, annot=True, fmt=".0f", cmap="Blues")
plt.title("객실 등급별 생존 비율(%)")
plt.xlabel("생존(0=사망, 1=생존)"); plt.ylabel("객실 등급")
plt.show()
```

문제 6.

- 질문 : 탑승항(`embarked`)별 **승객 수**와 **생존률**을 각각 어떻게 보여줄까?
- 문제 : 탑승항별 인원 & 생존률
- 권장 그래프 : 두 시각화 비교 → `countplot` (개수) + `barplot(y='survived')` (비율)
- **왜 이 그래프? 개수**와 **비율**은 다른 질문. 각각 최적 그래프가 다름.

```
# (1) 인원
sns.countplot(data=titanic, x="Embarked")
plt.title("탑승항별 승객 수")
plt.show()

# (2) 생존률
ax = sns.barplot(data=titanic, x="Embarked", y="Survived", errorbar=None)
plt.title("탑승항별 생존률")
plt.ylim(0,1)
plt.show()
```

문제 7.

- 질문 : Titanic의 **수치형 변수들끼리 관계(상관)**를 한눈에 파악하려면 어떻게 보여줄까?
- 문제 : 수치형 컬럼만 골라 **피어슨 상관행렬** 계산 후 **히트맵**으로 시각화
- 권장 그래프 : 상관행렬 → `heatmap`

- **왜 이 그래프?** 여러 수치 변수의 **쌍별 상관(-1~1)**을 **색과 수치**로 직관적으로 비교할 수 있음.

```
num = titanic.select_dtypes(include="number")
corr = num.corr()  # 기본: Pearson → 스피어만 : num.corr(method="spearman")
plt.figure(figsize=(5.6,4.6))
sns.heatmap(corr, annot=True, fmt=".2f", vmin=-1, vmax=1, center=0, cmap="coolwarm")
plt.title("Pearson Correlation (numeric only)")
plt.tight_layout()
plt.show()
```

📎 미니 과제

1. H&M
    a. 데이터 :
    b. 실습 파일 :
    c. 정답 :
2. 서울시 부동산 데이터
    a. 데이터 :
    b. 실습 파일 :
    c. 정답 :

📎 자료

1. 자료 PDF :
2. 이전 전처리 라이브 세션 자료 :