🏠 [3주차] Ames 집값 EDA & 전처리 의사결정 파이프라인

과제 설명

미국 아이오와주 에임스市에서 거래된 주택 **1,460건**의 데이터를 가지고, 강의에서 배운 EDA·기술통계를 **직접 적용**하는 작은 프로젝트다. 목표는 "모델을 만드는 것"이 아니라, 데이터를 관찰하고 **"그래서 어떻게 전처리할 것인가"를 스스로 결정**하는 것이다.

어떻게 진행되나? (모든 파트가 같은 3단계 반복)

> **① 실습 → ② 해석 → ③ 의사결정**

`_____` 빈칸 또는 `# TODO`로 남겨둔 *빈칸 몇 개와, 그보다 훨씬 중요한 "✍️ 서술형" 답** 를 채워주세요.

⚠️ 제출 규칙

1. 모든 `_____` 빈칸과 `# TODO`를 채운다.
2. **"✍️ 서술형"** 답은 반드시 **본인 데이터의 실제 수치를 인용**해 작성한다. (예: "SalePrice 왜도가 1.88이므로...")
3. **맨 마지막 셀**에 생성형 AI(GPT/Claude) 사용 내역을 붙여넣는다. (안 썼으면 "사용 안 함")

---

0. 환경 설정 & 데이터 로드

아래 셀들을 순서대로 실행한다. 라이브러리를 불러오고, 인터넷에서 데이터를 바로 읽어온다. (별도 파일 업로드 불필요)

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

plt.rcParams['axes.unicode_minus'] = False
sns.set_theme(style="whitegrid")
pd.set_option('display.max_columns', 100)
print("라이브러리 준비 완료 ✅")
```

```
라이브러리 준비 완료 ✅
```

In [2]:
```
# 데이터 불러오기: Ames Housing (1460행 × 81열)
URL = "https://raw.githubusercontent.com/hjhuney/Data/master/AmesHousing/
df = pd.read_csv(URL)
```

```
print("데이터 크기 (행, 열):", df.shape)
df.head()
```

```
데이터 크기 (행, 열): (1460, 81)
```

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotSha
0	1	60	RL	65.0	8450	Pave	NaN	R
1	2	20	RL	80.0	9600	Pave	NaN	R
2	3	60	RL	68.0	11250	Pave	NaN	I
3	4	70	RL	60.0	9550	Pave	NaN	I
4	5	60	RL	84.0	14260	Pave	NaN	I

> **데이터 설명**: 한 행(row)이 주택 1채의 거래 기록이다. 예측 대상(타깃)은 `SalePrice`(판매가, 달러)이고, 나머지 79개 컬럼이 그 집의 특징(면적·품질·동네·차고 등)이다.

Part 1. 데이터 타입의 정렬 — dtype vs 실제 의미

> ① 실습: `info()` · `describe()` 로 전체를 보고, `MSSubClass` 의 기술통계를 확인 ② 해석: 컴퓨터가 '숫자'로 읽은 변수가 정말 사칙연산이 되는 변수인지 비판적으로 검토 ③ 의사결정: 인코딩 전략을 결정

In [3]:
```
# ① 전체 형태 확인
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       1460 non-null   object 
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64  
 5   Street         1460 non-null   object 
 6   Alley          91 non-null     object 
 7   LotShape       1460 non-null   object 
 8   LandContour    1460 non-null   object 
 9   Utilities      1460 non-null   object 
 10  LotConfig      1460 non-null   object 
 11  LandSlope      1460 non-null   object 
 12  Neighborhood   1460 non-null   object 
 13  Condition1     1460 non-null   object 
 14  Condition2     1460 non-null   object 
 15  BldgType       1460 non-null   object 
 16  HouseStyle     1460 non-null   object 
 17  OverallQual    1460 non-null   int64  
 18  OverallCond    1460 non-null   int64  
 19  YearBuilt      1460 non-null   int64  
 20  YearRemodAdd   1460 non-null   int64  
 21  RoofStyle      1460 non-null   object 
 22  RoofMatl       1460 non-null   object 
 23  Exterior1st    1460 non-null   object 
 24  Exterior2nd    1460 non-null   object 
 25  MasVnrType     588 non-null    object 
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object 
 28  ExterCond      1460 non-null   object 
 29  Foundation     1460 non-null   object 
 30  BsmtQual       1423 non-null   object 
 31  BsmtCond       1423 non-null   object 
 32  BsmtExposure   1422 non-null   object 
 33  BsmtFinType1   1423 non-null   object 
 34  BsmtFinSF1     1460 non-null   int64  
 35  BsmtFinType2   1422 non-null   object 
 36  BsmtFinSF2     1460 non-null   int64  
 37  BsmtUnfSF      1460 non-null   int64  
 38  TotalBsmtSF    1460 non-null   int64  
 39  Heating        1460 non-null   object 
 40  HeatingQC      1460 non-null   object 
 41  CentralAir     1460 non-null   object 
 42  Electrical     1459 non-null   object 
 43  1stFlrSF       1460 non-null   int64  
 44  2ndFlrSF       1460 non-null   int64  
 45  LowQualFinSF   1460 non-null   int64  
 46  GrLivArea      1460 non-null   int64  
 47  BsmtFullBath   1460 non-null   int64  
 48  BsmtHalfBath   1460 non-null   int64  
 49  FullBath       1460 non-null   int64  
 50  HalfBath       1460 non-null   int64  
 51  BedroomAbvGr   1460 non-null   int64  
 52  KitchenAbvGr   1460 non-null   int64  
 53  KitchenQual    1460 non-null   object 
 54  TotRmsAbvGrd   1460 non-null   int64  
```

```
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

In [4]:
```
# 수치형 변수들의 요약 통계
df.describe()
```

Out[4]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315
std	421.610009	42.300571	24.284752	9981.264932	1.382997
min	1.000000	20.000000	21.000000	1300.000000	1.000000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000

In [5]:
```
# MSSubClass(주택 유형 코드)는 숫자로 저장돼 있다. 직접 확인해보자.

# TODO: dtype, 평균, 고유값을 출력하라
print("MSSubClass dtype :", df['MSSubClass']._____)           # 저장 타입
print("MSSubClass 평균  :", round(df['MSSubClass']._____(), 2))  # 평균 계
print("MSSubClass 고유값:", sorted(df['MSSubClass']._____()))    # 고유값
# 출력 결과를 보고 이 변수의 평균값이 의미 있는 숫자인지 생각해보자
```

```
MSSubClass dtype : int64
MSSubClass 평균  : 56.9
MSSubClass 고유값: [np.int64(20), np.int64(30), np.int64(40), np.int64(45),
np.int64(50), np.int64(60), np.int64(70), np.int64(75), np.int64(80), np.i
nt64(85), np.int64(90), np.int64(120), np.int64(160), np.int64(180), np.in
t64(190)]
```

✍️ **서술형 Part 1**

Q1. `MSSubClass`의 평균값(위 출력)이 통계적으로 무의미한 이유를, 변수의 척도(명목/순서/등간/비율) 개념과 연결해 설명하세요. 그리고 모델이 이 변수를 크기가 있는 숫자로 오해하지 않도록 **어떤 인코딩**을 적용할지 결정하세요. → **답:** _____

Q2. 두 범주형 변수 `ExterQual`(외관 품질: Ex>Gd>TA>Fa>Po, 순서 있음)과 `Neighborhood`(동네 이름, 순서 없음)를 숫자로 바꾸려 합니다. 각각 **Ordinal 인코딩**과 **One-Hot 인코딩** 중 무엇을 써야 하며 그 이유는 무엇인가요? (힌트: 순서 정보가 있는가?) → **답:** _____

A1.

A2.

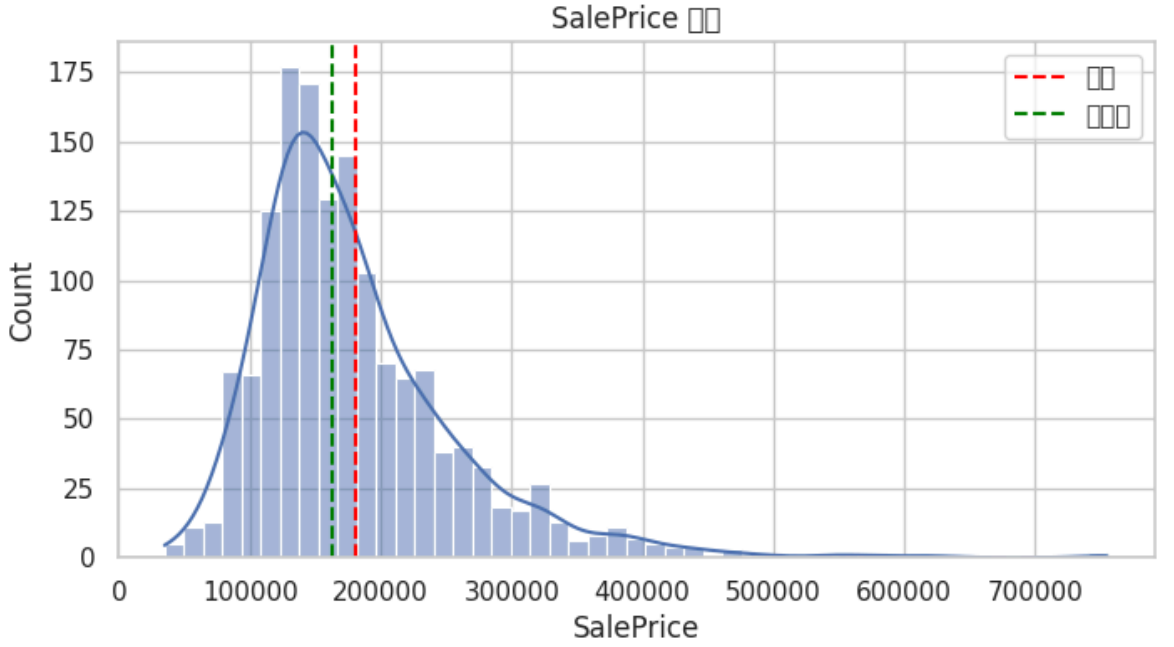
Part 2. Target 변수의 분포와 변환 전략

> ① 실습: `SalePrice`의 히스토그램·왜도·첨도, 그리고 `LotArea`의 평균 vs 중앙값 ② 해석: 꼬리가 어디로 치우쳤는지, 평균-중앙값 격차가 무엇을 뜻하는지 파악 ③ 의사결정: 로그 변환 필요 여부, 대표값 선택

In [6]:
```python
# ① SalePrice 분포 진단
sp = df['SalePrice']

# TODO: 왜도(skewness)와 첨도(kurtosis)를 구하라  (힌트: 시리즈의 .skew() , .k
skew_ = _______
kurt_ = _______

print(f"평균  : {sp.mean():,.0f}")
print(f"중앙값: {sp.median():,.0f}")
print(f"왜도  : {skew_:.2f}")
print(f"첨도  : {kurt_:.2f}")

plt.figure(figsize=(8,4))
sns.histplot(sp, kde=True)
plt.axvline(sp.mean(),   color='red',   ls='--', label='평균')
plt.axvline(sp.median(), color='green', ls='--', label='중앙값')
plt.title('SalePrice 분포'); plt.legend(); plt.show()
```

```
평균  : 180,921
중앙값: 163,000
왜도  : 1.88
첨도  : 6.54
```

```
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 48516 (\N{HANGUL SYLLABLE BUN}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 54252 (\N{HANGUL SYLLABLE PO}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 54217 (\N{HANGUL SYLLABLE PYEONG}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 44512 (\N{HANGUL SYLLABLE GYUN}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 51473 (\N{HANGUL SYLLABLE JUNG}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 50521 (\N{HANGUL SYLLABLE ANG}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 44050 (\N{HANGUL SYLLABLE GABS}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [7]:
```
# LotArea(부지 면적)의 평균 vs 중앙값 비교
print(f"LotArea 평균  : {df['LotArea'].mean():,.0f}")
print(f"LotArea 중앙값: {df['LotArea'].median():,.0f}")
print(f"LotArea 최댓값: {df['LotArea'].max():,.0f}  <- 극단값 확인")
```

```
LotArea 평균  : 10,517
LotArea 중앙값: 9,478
LotArea 최댓값: 215,245  <- 극단값 확인
```

✍️ 서술형 **Part 2**

Q1. `SalePrice`의 왜도 부호와 분포 꼬리 방향(좌왜/우왜)을 쓰고, 이 분포를 선형 모델에 넣을 때 어떤 변환(예: 로그 변환)이 필요한지 결정하세요. 변환한다면 왜도 수치가 어떻게 변할지도 예측해보세요. → **답:** _____

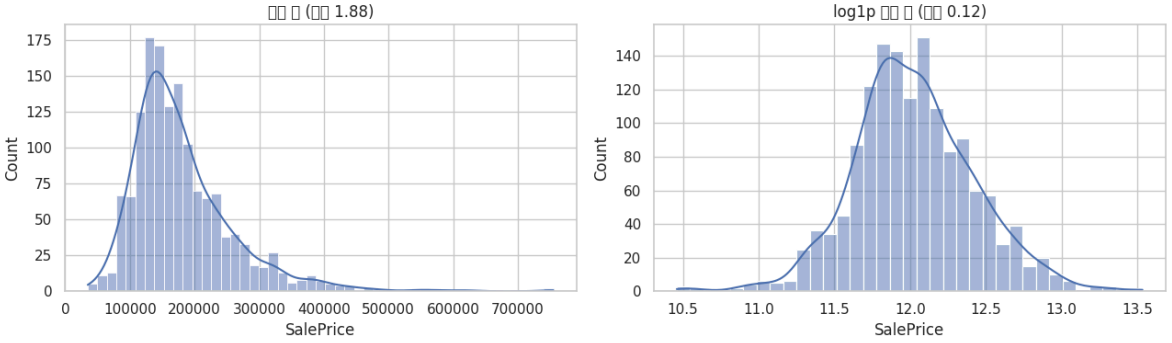
Q2. `LotArea`의 평균과 중앙값 중, 현재 데이터의 '대표값'으로 더 적절한 것은? 극단적 이상치가 통계량에 미치는 영향을 근거로 방어하세요. → **답:** _____

A1.

A2.

In [8]:
```
# ③ 로그 변환을 직접 적용해 예측이 맞았는지 확인
sp_log = np.log1p(sp)   # log1p = log(1+x), 0이 있어도 안전

fig, ax = plt.subplots(1, 2, figsize=(13,4))
sns.histplot(sp,     kde=True, ax=ax[0]); ax[0].set_title(f'변환 전 (왜도 {
sns.histplot(sp_log, kde=True, ax=ax[1]); ax[1].set_title(f'log1p 변환 후
plt.tight_layout(); plt.show()
print("변환 후 왜도가 0에 가까워졌는가?", round(sp_log.skew(), 3))
```

```
/tmp/ipykernel_735/2563761168.py:7: UserWarning: Glyph 48320 (\N{HANGUL SY
LLABLE BYEON}) missing from font(s) DejaVu Sans.
  plt.tight_layout(); plt.show()
/tmp/ipykernel_735/2563761168.py:7: UserWarning: Glyph 54872 (\N{HANGUL SY
LLABLE HWAN}) missing from font(s) DejaVu Sans.
  plt.tight_layout(); plt.show()
/tmp/ipykernel_735/2563761168.py:7: UserWarning: Glyph 51204 (\N{HANGUL SY
LLABLE JEON}) missing from font(s) DejaVu Sans.
  plt.tight_layout(); plt.show()
/tmp/ipykernel_735/2563761168.py:7: UserWarning: Glyph 50780 (\N{HANGUL SY
LLABLE WAE}) missing from font(s) DejaVu Sans.
  plt.tight_layout(); plt.show()
/tmp/ipykernel_735/2563761168.py:7: UserWarning: Glyph 46020 (\N{HANGUL SY
LLABLE DO}) missing from font(s) DejaVu Sans.
  plt.tight_layout(); plt.show()
/tmp/ipykernel_735/2563761168.py:7: UserWarning: Glyph 54980 (\N{HANGUL SY
LLABLE HU}) missing from font(s) DejaVu Sans.
  plt.tight_layout(); plt.show()
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 48320 (\N{HANGUL SYLLABLE BYEON}) missing from font(s) De
jaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 54872 (\N{HANGUL SYLLABLE HWAN}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 51204 (\N{HANGUL SYLLABLE JEON}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 50780 (\N{HANGUL SYLLABLE WAE}) missing from font(s) Deja
Vu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 46020 (\N{HANGUL SYLLABLE DO}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 54980 (\N{HANGUL SYLLABLE HU}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



변환 후 왜도가 0에 가까워졌는가? 0.121

Part 3. 결측치 비율에 따른 입체적 대체 전략 🔥

> ① 실습: 결측 컬럼과 비율을 내림차순 테이블로 ② 해석: 높은 결측(FireplaceQu)·애매한 결측(LotFrontage)·0 밀집(MasVnrArea)의 맥락 유추 ③ 의사결정: 컬럼마다 다른 처방을 설계

이 파트가 이번 과제의 **변별력 구간**입니다. "결측이면 무조건 채운다/버린다"가 아니라, 결측이 왜 생겼는지에 따라 처방이 달라집니다.

In [9]:
```
# ① 결측 비율(%) 내림차순
miss = (df.isnull().mean()*100).sort_values(ascending=False)
miss = miss[miss > 0].round(1)
print("결측이 있는 컬럼 수:", len(miss))
miss.head(20)
```

```
결측이 있는 컬럼 수: 19
```

Out[9]:

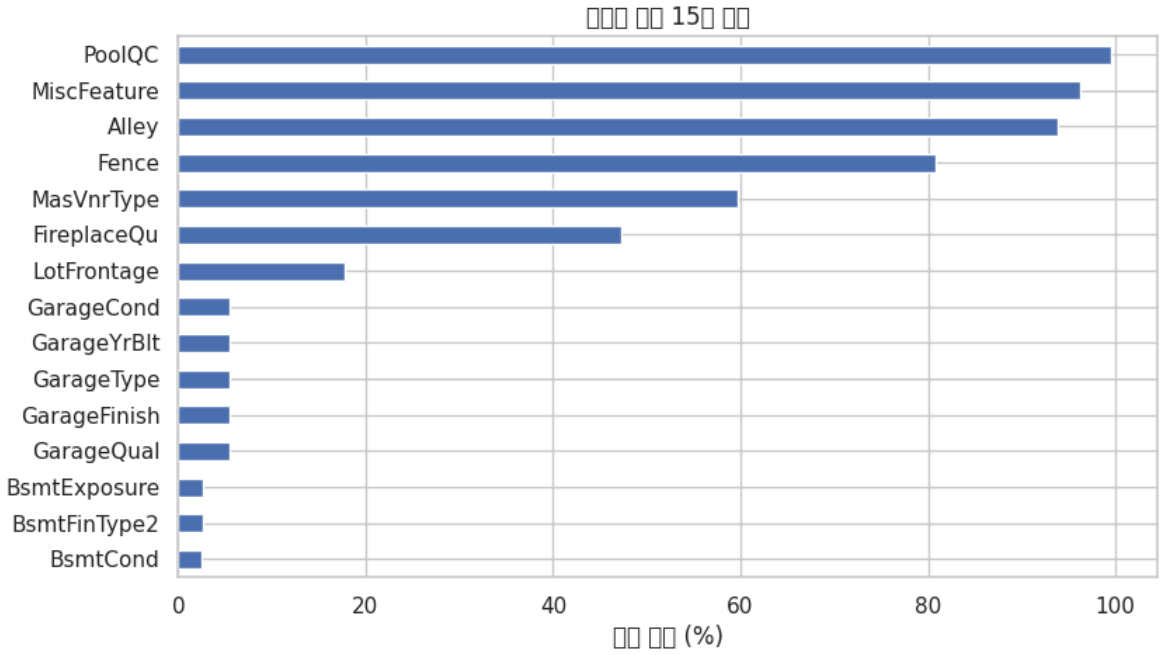
	0
PoolQC	99.5
MiscFeature	96.3
Alley	93.8
Fence	80.8
MasVnrType	59.7
FireplaceQu	47.3
LotFrontage	17.7
GarageQual	5.5
GarageFinish	5.5
GarageType	5.5
GarageYrBlt	5.5
GarageCond	5.5
BsmtFinType2	2.6
BsmtExposure	2.6
BsmtCond	2.5
BsmtQual	2.5
BsmtFinType1	2.5
MasVnrArea	0.5
Electrical	0.1

dtype: float64

In [10]:
```
# 결측 비율 상위 시각화
plt.figure(figsize=(9,5))
```

```
miss.head(15).sort_values().plot(kind='barh')
plt.xlabel('결측 비율 (%)'); plt.title('결측치 상위 15개 컬럼'); plt.show()
```

```
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 44208 (\N{HANGUL SYLLABLE GYEOL}) missing from font(s) De
jaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 52769 (\N{HANGUL SYLLABLE CEUG}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 52824 (\N{HANGUL SYLLABLE CI}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 49345 (\N{HANGUL SYLLABLE SANG}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 50948 (\N{HANGUL SYLLABLE WI}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 44060 (\N{HANGUL SYLLABLE GAE}) missing from font(s) Deja
Vu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 52972 (\N{HANGUL SYLLABLE KEOL}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 47100 (\N{HANGUL SYLLABLE REOM}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 48708 (\N{HANGUL SYLLABLE BI}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 50984 (\N{HANGUL SYLLABLE YUL}) missing from font(s) Deja
Vu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [11]:
```
# ② 세 변수의 맥락을 조사해보자
# (a) FireplaceQu(벽난로 품질)가 결측인 집의 Fireplaces(벽난로 개수)는?
print("[FireplaceQu 결측 행의 벽난로 개수]")
print(df.loc[df['FireplaceQu'].isnull(), 'Fireplaces'].value_counts(), "\

# (b) MasVnrArea(조적 베니어 면적)의 값 분포 — 0이 얼마나 몰려있나?
# 힌트: 0인지 아닌지를 True/False(=1/0)로 변환 → 합산하면 0의 개수, 평균내면 0의
print("[MasVnrArea] 결측 개수:", df['MasVnrArea'].______().sum(),
      "| 값이 0인 비율: %.1f%%" % ((df['MasVnrArea'] == 0).______() * 100))

# (c) LotFrontage 결측을 전체 중앙값 하나로 채우면 동네마다 도로 폭이 다른데도
# 모든 집에 같은 값이 들어간다. 동네별로 중앙값이 실제로 다른지 확인해보자. >> 도구
# TODO: 동네(Neighborhood)를 기준으로 묶어 LotFrontage의 중앙값을 구하라
print("[동네별 LotFrontage 중앙값 (일부)]")
print(df.______('Neighborhood')['LotFrontage'].______().dropna().head(6))
```

```
[FireplaceQu 결측 행의 벽난로 개수]
Fireplaces
0    690
Name: count, dtype: int64

[MasVnrArea] 결측 개수: 8 | 값이 0인 비율: 59.0%

[동네별 LotFrontage 중앙값 (일부)]
Neighborhood
Blmngtn    43.0
Blueste    24.0
BrDale     21.0
BrkSide    52.0
ClearCr    80.0
CollgCr    70.0
Name: LotFrontage, dtype: float64
```

✍️ **서술형 Part 3**

Q1 [구조적 결측]. `FireplaceQu`는 결측률이 약 47%지만, 위 (a)에서 보듯 결측 행은 전부 벽난로 개수가 0이다. 즉 결측이 아니라 **'벽난로 없음'이라는 정보**다. 이 컬럼을

그냥 '열 삭제'하면 왜 위험한가요? 데이터를 보존하는 본인만의 처리법(예: `'None'` 범주 부여)을 제시하세요. → **답:** _____

Q2 [0 밀집 분포]. `MasVnrArea`는 값의 약 59%가 0(=베니어 없음)이다. 이 소수의 결측(8건)을 **'0을 포함한 전체 평균값'**으로 채우면 어떤 왜곡이 생기는가? 더 안전한 대안은? → **답:** _____

A1.

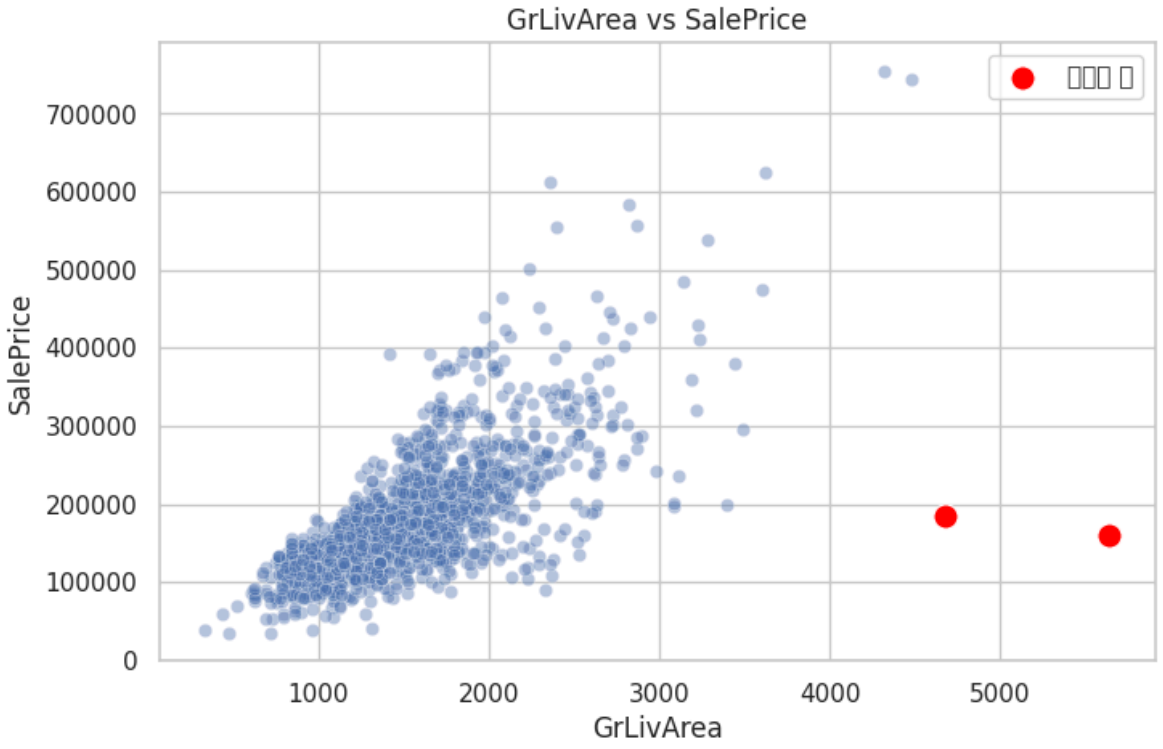
A2.

Part 4. 의미 있는 이상치와 파생 변수의 부작용

> ① 실습: `GrLivArea` vs `SalePrice` 산점도에서 이상 패턴 포착, `Total_Area` 파생 변수 생성 ② 해석: 포착된 점이 수집 오류(Noise)인지 실제 거래(의미 있는 이상치)인지 추론 ③ 의사결정: 제거할지 유지할지, 파생 변수의 부작용 대응

In [12]:
```
# ① 산점도로 이상 패턴 찾기
plt.figure(figsize=(8,5))
# TODO: 두 변수의 관계를 산점도로 그려라
sns._____(data=df, x='GrLivArea', y='SalePrice', alpha=0.4)

# 면적은 아주 큰데(>4000) 가격은 낮은(<200000) 수상한 점을 빨간색으로
weird = df[(df['GrLivArea'] > 4000) & (df['SalePrice'] < 200000)]
# TODO: 위에서 걸러낸 이상점을 빨간색으로 덧그려라
sns._____(data=weird, x='GrLivArea', y='SalePrice', color='red', s=120,
plt.title('GrLivArea vs SalePrice'); plt.legend(); plt.show()

print(f"수상한 점 개수: {len(weird)}")
print(weird[['GrLivArea','SalePrice','OverallQual','SaleCondition']])
```

```
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 49688 (\N{HANGUL SYLLABLE SU}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 49345 (\N{HANGUL SYLLABLE SANG}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 54620 (\N{HANGUL SYLLABLE HAN}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 51216 (\N{HANGUL SYLLABLE JEOM}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```




```
수상한 점 개수: 2
      GrLivArea  SalePrice  OverallQual SaleCondition
523        4676     184750           10       Partial
1298       5642     160000           10       Partial
```

In [13]:
```
# ③ 여러 면적 변수를 더해 Total_Area 파생 변수 생성
df['Total_Area'] = df['1stFlrSF'] + df['2ndFlrSF'] + df['TotalBsmtSF']

# 새 변수와 기존 변수들의 상관을 확인
print(df[['Total_Area','1stFlrSF','2ndFlrSF','TotalBsmtSF','SalePrice']].
```

```
Total_Area     0.782
1stFlrSF       0.606
2ndFlrSF       0.319
TotalBsmtSF    0.614
SalePrice      1.000
Name: SalePrice, dtype: float64
```

✍️ **서술형 Part 4**

Q1 [이상치 판단]. 빨간 점 2개는 `SaleCondition`이 `Partial`(미완공 상태 거래)이다. 이들을 **노이즈로 보고 완전히 제거**할 것인가, 아니면 **유지하되 이상치에 강건한 트리 계열 모델**을 쓸 것인가? 본인의 결정 논리를 서술하라. → **답:** _____

Q2 [다중공선성]. `Total_Area`는 `1stFlrSF + 2ndFlrSF + TotalBsmtSF`의 합이라, 이 셋과 거의 완벽하게 겹친다(위 상관 확인). 파생 변수와 원본 3개를 **모두** 모델에 넣으면 왜 문제가 되는가? (힌트: 서로 강하게 상관된 변수를 중복 투입 = 다중공선성) 원본을 유지할지 삭제할지 결정하라. → **답:** _____

A1.

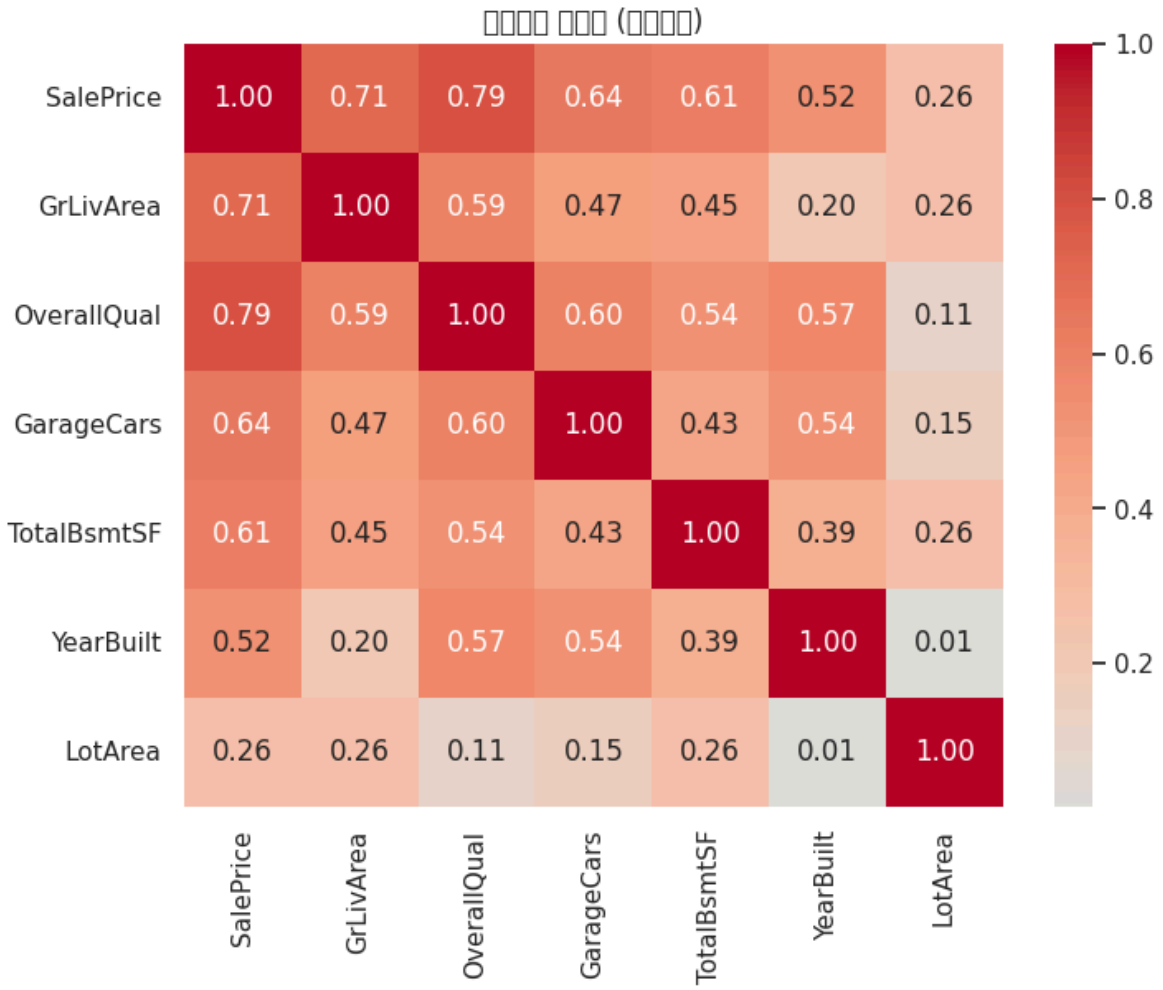
A2.

Part 5. 다변량 관계의 함정과 Target Leakage

> ① 실습: 수치형 상관 히트맵 + 범주형은 Boxplot으로 ② 해석: 히트맵에 범주형(동네 등)의 영향력이 누락됨을 인지 ③ 의사결정: 분석의 맹점, Target Leakage 필터링

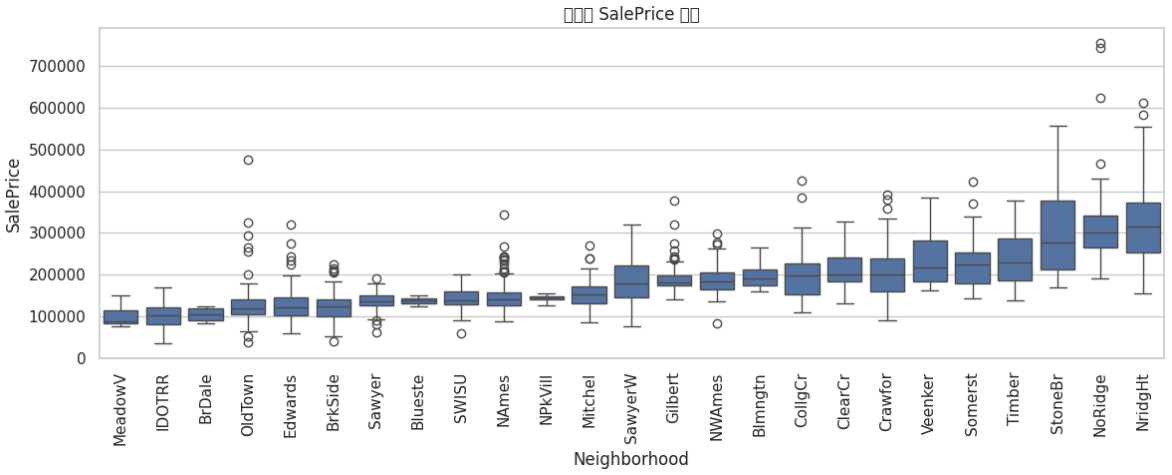
In [14]:
```
# ① 수치형 상관 히트맵
key = ['SalePrice','GrLivArea','OverallQual','GarageCars','TotalBsmtSF','
plt.figure(figsize=(8,6))
# TODO: 상관계수 행렬을 구하고 히트맵으로 그려라
sns._____(df[key]._____(), annot=True, fmt='.2f', cmap='coolwarm', cent
plt.title('상관계수 히트맵 (수치형만)'); plt.show()
```

```
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 49345 (\N{HANGUL SYLLABLE SANG}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 44288 (\N{HANGUL SYLLABLE GWAN}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 44228 (\N{HANGUL SYLLABLE GYE}) missing from font(s) Deja
Vu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 49688 (\N{HANGUL SYLLABLE SU}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 55176 (\N{HANGUL SYLLABLE HI}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 53944 (\N{HANGUL SYLLABLE TEU}) missing from font(s) Deja
Vu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 47605 (\N{HANGUL SYLLABLE MAEB}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 52824 (\N{HANGUL SYLLABLE CI}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 54805 (\N{HANGUL SYLLABLE HYEONG}) missing from font(s) D
ejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 47564 (\N{HANGUL SYLLABLE MAN}) missing from font(s) Deja
Vu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [15]:
```
# 히트맵엔 없는 범주형 Neighborhood(동네)의 영향력을 Boxplot으로
order = df.groupby('Neighborhood')['SalePrice'].median().sort_values().in
plt.figure(figsize=(12,5))
# TODO: 동네별 SalePrice 분포를 박스플롯으로 그려라
sns._____(data=df, x='Neighborhood', y='SalePrice', order=order)
plt.xticks(rotation=90); plt.title('동네별 SalePrice 분포'); plt.tight_layc

nb = df.groupby('Neighborhood')['SalePrice'].median()
print(f"가장 싼 동네 중앙값 {nb.min():,.0f} vs 가장 비싼 동네 {nb.max():,.0f}
```

```
/tmp/ipykernel_735/1827557070.py:5: UserWarning: Glyph 46041 (\N{HANGUL SY
LLABLE DONG}) missing from font(s) DejaVu Sans.
  plt.xticks(rotation=90); plt.title('동네별 SalePrice 분포'); plt.tight_la
yout(); plt.show()
/tmp/ipykernel_735/1827557070.py:5: UserWarning: Glyph 45348 (\N{HANGUL SY
LLABLE NE}) missing from font(s) DejaVu Sans.
  plt.xticks(rotation=90); plt.title('동네별 SalePrice 분포'); plt.tight_la
yout(); plt.show()
/tmp/ipykernel_735/1827557070.py:5: UserWarning: Glyph 48324 (\N{HANGUL SY
LLABLE BYEOL}) missing from font(s) DejaVu Sans.
  plt.xticks(rotation=90); plt.title('동네별 SalePrice 분포'); plt.tight_la
yout(); plt.show()
/tmp/ipykernel_735/1827557070.py:5: UserWarning: Glyph 48516 (\N{HANGUL SY
LLABLE BUN}) missing from font(s) DejaVu Sans.
  plt.xticks(rotation=90); plt.title('동네별 SalePrice 분포'); plt.tight_la
yout(); plt.show()
/tmp/ipykernel_735/1827557070.py:5: UserWarning: Glyph 54252 (\N{HANGUL SY
LLABLE PO}) missing from font(s) DejaVu Sans.
  plt.xticks(rotation=90); plt.title('동네별 SalePrice 분포'); plt.tight_la
yout(); plt.show()
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 46041 (\N{HANGUL SYLLABLE DONG}) missing from font(s) Dej
aVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 45348 (\N{HANGUL SYLLABLE NE}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 48324 (\N{HANGUL SYLLABLE BYEOL}) missing from font(s) De
jaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 48516 (\N{HANGUL SYLLABLE BUN}) missing from font(s) Deja
Vu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: Us
erWarning: Glyph 54252 (\N{HANGUL SYLLABLE PO}) missing from font(s) DejaV
u Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



가장 싼 동네 중앙값 88,000 vs 가장 비싼 동네 315,000 (3.6배)

✍️ 서술형 Part 5

Q1 [분석의 맹점]. 위 수치형 히트맵만 보고 "집값에 영향을 주는 건 이 수치형 변수들뿐"이라고 결론 내리면 무엇을 놓치는가? Boxplot에서 드러난 `Neighborhood`의 영향력을 근거로, 강의에서 배운 **히트맵의 3가지 맹점 중 어느 것**에 해당하는지 짚어라. → **답:** _____

Q2 [Target Leakage]. 데이터에 '판매 직전 납부한 취등록세액' 변수가 있고 `SalePrice`와 상관이 0.99라고 하자. 취등록세는 보통 판매가에 비례해 매겨진다. 이 변수를 피처에서 **제외해야 하는 이유**를 Target Leakage 관점에서 설명하라. (힌트: 집을 팔기 *전에* 알 수 있는 값인가?) → **답:** _____

A1.

A2.

Part 6. 데이터 누수 없는 파이프라인 (심화·선택)

> ① 실습: Train/Test 분할 후, 대체·스케일링 코드의 TODO를 채워 올바르게 작동시키기 ② 해석: "Train 통계로 Test를 채운다"는 원리와 정보 유출의 차이 ③ 의사결정: 분할을 '가장 먼저' 해야 하는 원칙 선언

In [17]:
```
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler

# 간단히 수치형 변수 몇 개만 사용
feats = ['GrLivArea','OverallQual','TotalBsmtSF','GarageCars','LotFrontag
X = df[feats].copy()
y = np.log1p(df['SalePrice'])

# 1) 가장 먼저 Train/Test 분할  (random_state는 본인 학번 뒤 4자리로!)
# TODO: 아래 ______ 에 본인 학번 앞 4자리 (예: 2021학번이면 2021)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

# 2) 결측 대체: Train 에서 fit, Test 에는 transform 만
imp = SimpleImputer(strategy='median')
X_train_imp = imp.fit_transform(X_train)          # fit_transform: Train
# TODO: Test 는 새로 fit 하지 말고, Train 이 학습한 값으로 transform 만 하라
X_test_imp  = imp.______(X_test)

# 3) 스케일링도 동일 원칙 (Train 에서만 fit)
sc = StandardScaler()
X_train_sc = sc.fit_transform(X_train_imp)
# TODO: Test 스케일링 (transform 만)
X_test_sc  = sc.______(X_test_imp)

print("완료 — Train:", X_train_sc.shape, "/ Test:", X_test_sc.shape)
print("Train 중앙값으로 채운 LotFrontage 대체값:", round(imp.statistics_[fea
```

```
완료 — Train: (1168, 5) / Test: (292, 5)
Train 중앙값으로 채운 LotFrontage 대체값: 69.0
```

✍️ 서술형 **Part 6**

Q1. Test 데이터에 `fit`(또는 `fit_transform`)을 하면 안 되는 이유를, "Test는 미래에 들어올 미지의 데이터"라는 관점에서 설명하세요. Test의 통계(평균·중앙값)를 미리 들여다보면 무엇이 새어나갈까요? → **답:** _____

Q2. 결측 대체·스케일링 같은 전처리보다 **Train/Test 분할을 반드시 먼저** 해야 하는 본인의 행동 원칙을, 데이터 누수 방지 측면에서 말해주세요. → **답:** _____

A1.

A2.

📌 [필수] 생성형 AI 활용 방법

이 과제를 풀며 GPT/Claude 등을 **어디에, 어떻게** 썼는지 기록한다. (안 썼으면 "사용 안 함")

답변

- **활용한 부분:**
- **대화 내역:**

🪞 회고

- 가장 어려웠던 부분과 이유:

A.

- 강의 땐 이해했다고 생각했지만 직접 해보며 새로 깨달은 점:

A.