1. Setup & Load Data

```
# Imports
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from scipy import stats

# Load
train = pd.read_csv("/content/train.csv", index_col="Id")
test = pd.read_csv("/content/test.csv", index_col="Id")
all_data = pd.concat([train.drop("SalePrice", axis=1), test])
y = train["SalePrice"]

print(all_data.shape)
```

```
(2919, 79)
```

2. Handling Missing Values

```
# Features mapped by strategy
fill_none = ["Alley","FireplaceQu","PoolQC","Fence","MiscFeature","GarageType",
             "GarageFinish","GarageQual","GarageCond","BsmtQual","BsmtCond",
             "BsmtExposure","BsmtFinType1","BsmtFinType2","MasVnrType"]
fill_zero = ["GarageYrBlt","GarageArea","GarageCars","BsmtFinSF1","BsmtFinSF2",
             "BsmtUnfSF","TotalBsmtSF","BsmtFullBath","BsmtHalfBath","MasVnrArea"]
fill_mode = ["MSZoning","Functional","Utilities","Electrical","KitchenQual","Exterior1st","Exterior2nd","SaleType","SaleCondition"]

# Fill
all_data[fill_none] = all_data[fill_none].fillna("None")
all_data[fill_zero] = all_data[fill_zero].fillna(0)
imp = SimpleImputer(strategy="most_frequent")
all_data[fill_mode] = imp.fit_transform(all_data[fill_mode])
print(all_data.isnull().sum().sort_values(ascending=False).head(10))
```

```
LotFrontage    486
MSSubClass       0
MSZoning         0
LotArea          0
Street           0
Alley            0
LotShape         0
LandContour      0
Utilities        0
LotConfig        0
dtype: int64
```

3. Feature Engineering

- a) Combine Areas & Age
- b) Transform Skewed Numericals
- c) Encode Ordinals & Categoricals
- d) One-Hot Encoding

```
#a) Combine Areas & Age
all_data["TotalSF"] = (all_data["TotalBsmtSF"] + all_data["1stFlrSF"] +
                       all_data["2ndFlrSF"])
all_data["HouseAge"] = all_data["YrSold"] - all_data["YearBuilt"]
all_data["RemodAge"] = all_data["YrSold"] - all_data["YearRemodAdd"]
all_data["TotalBath"] = (all_data["FullBath"] + 0.5*all_data["HalfBath"] +
                         all_data["BsmtFullBath"] + 0.5*all_data["BsmtHalfBath"])
```

```
#b) Transform Skewed Numericals
numeric_feats = all_data.dtypes[all_data.dtypes != "object"].index
skewness = all_data[numeric_feats].apply(lambda x: stats.skew(x.dropna()))
skewed = skewness[abs(skewness) > 0.75].index
from scipy.special import boxcox1p
for feat in skewed:
    all_data[feat] = boxcox1p(all_data[feat], 0.15)
```

```
#c) Encode Ordinals & Categoricals
# Example ordinal mapping
qual_map = {"None":0, "Po":1, "Fa":2, "TA":3, "Gd":4, "Ex":5}
all_data["ExterQual"] = all_data["ExterQual"].map(qual_map)
# Add more similarly...

# Label encode neighborhood
lbl = LabelEncoder()
all_data["Neighborhood"] = lbl.fit_transform(all_data["Neighborhood"])



#d) One-Hot Encoding
all_data = pd.get_dummies(all_data)
print("Final features:", all_data.shape)
```

```
Final features: (2919, 279)
```

4. Train/Test Split

```
X = all_data.loc[train.index]
X_test = all_data.loc[test.index]
print(X.shape, X_test.shape)
# Distribution ✓
print("Target skewness:", stats.skew(y))
```

```
(1460, 279) (1459, 279)
Target skewness: 1.880940746034036
```

Outlier Removal

```
# Remove extreme TotalSF vs SalePrice
idx = X[(X["TotalSF"] > 5e4) & (y < np.percentile(y, 75))].index
X.drop(idx, inplace=True); y.drop(idx, inplace=True)
print("After outlier removal:", X.shape)
```

```
After outlier removal: (1460, 279)
```