Department of Data Science

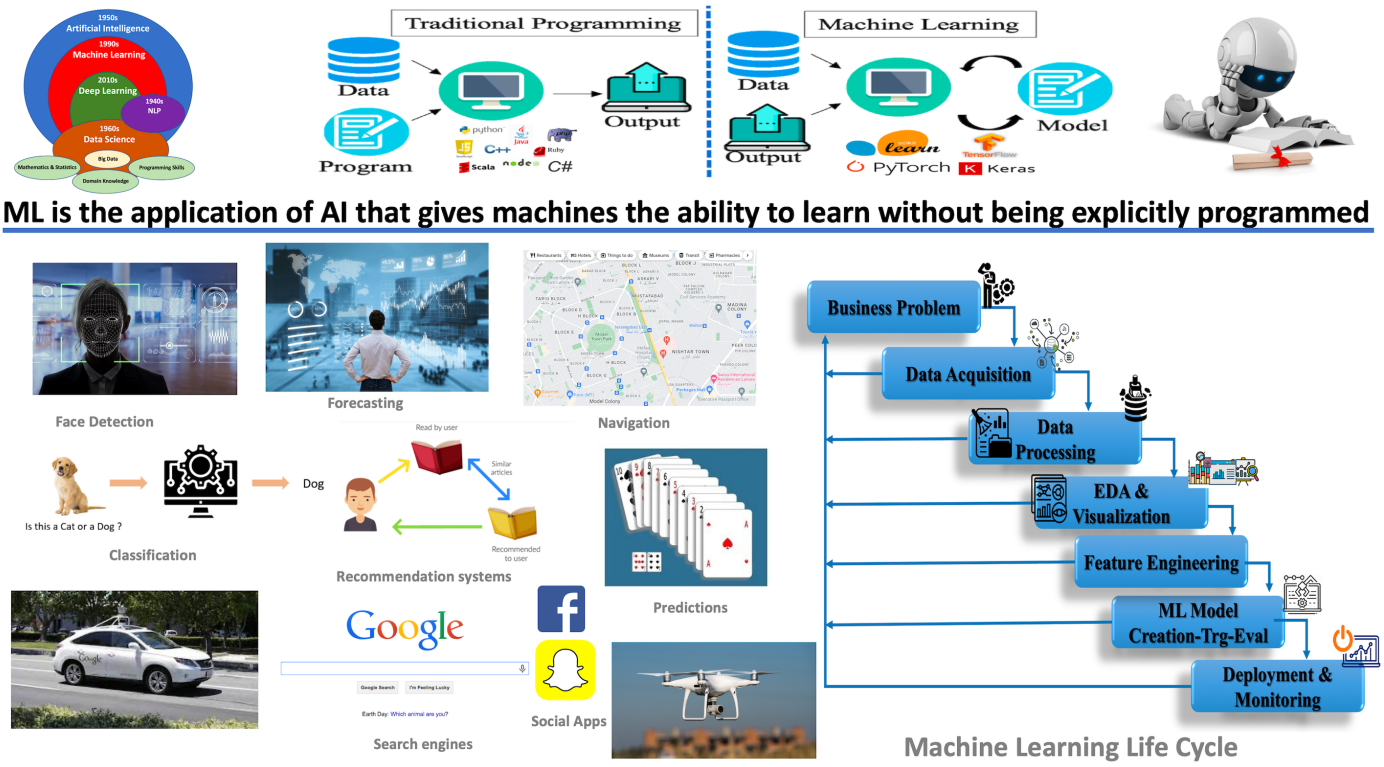
Course: Tools and Techniques for Data Science

Instructor: Muhammad Arif Butt, Ph.D.

Lecture 6.7 (Preprocessing/Feature Engineering - Handling Outliers)

Open in Colab

(https://colab.research.google.com/github/arifpucit/data-science/blob/master/Section-4-Mathematics-for-Data-Science/Lec-4.1(Descriptive-Statistics).ipynb)



Learning agenda of this notebook

1. Overview of Data Pre-Processing and Feature Engineering
2. Detecting and Handling Outliers
    - Univariate Analysis
        - Z-Score Method

    - IQR Method
    - Percentile Method
- Multivariate Analysis

Overview of Data Pre-Processing and Feature Engineering

- `Data Preprocessing` involves actions that we need to perform on the dataset in order to make it ready to be fed to the machine learning model.
- `Feature Engineering` is the process of using domain knowledge to extract features from raw data via data mining techniques.

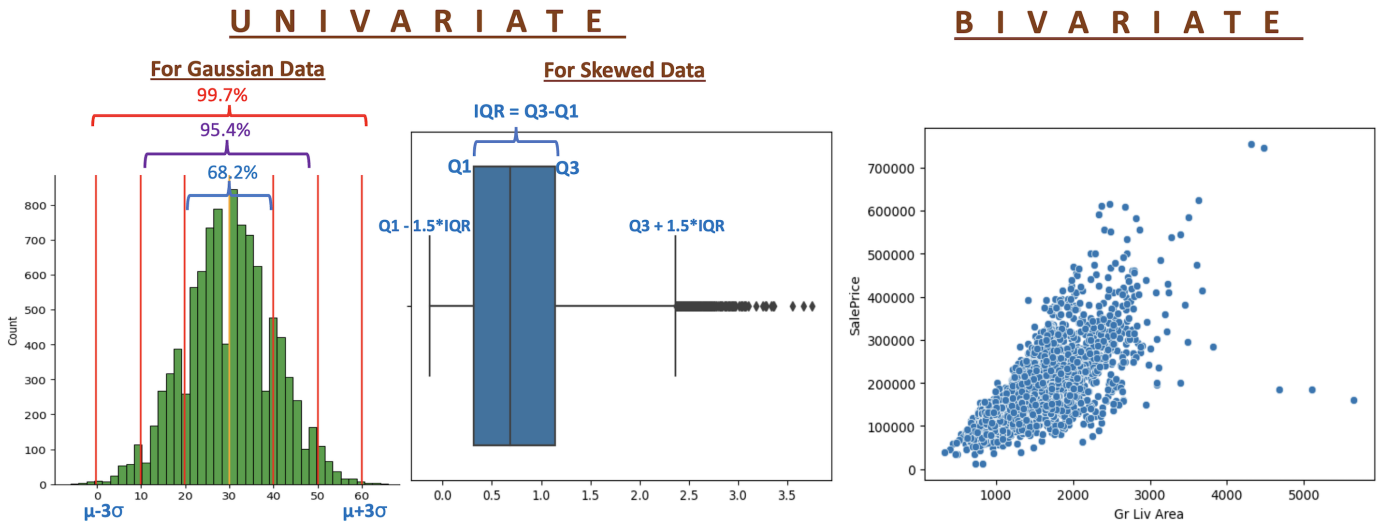
City	Size	Covered Area	No of bedrooms	Trees near by	No of bathrooms	Schools near by	Construction Date	Price
Lahore	2000	3500	3	1	3	1	25/10/2001	20.5 M
Karachi	2600	3000	2	0	4	1	16/05/1990	18 M
Islamabad	1800	2000	3	1	3	2	25/11/1995	20 M
Shaikhupura	1600	2600	1	2	NaN	0	08/06/2020	5 M
Lahore	2600	2000	3	3	1	1	03/09/2016	4 M
Karachi	3000	1000	2	2	1	NaN	19/01/1980	6 M
Islamabad	2000	3600	44	4	3	3	21/07/1999	30 M
Lahore	1000	2000	3	NaN	1	2	12/04/2015	10 M

- Pre-processing package of sklearn provides a bundle of utility functions and transformer classes for data preprocessing (will cover later).
    - **Detecting and handling outliers**
        - Univariate (Z-Score, IQR, Percentiles)
        - Multivariate Analysis (Depth-based, Distance-based, Density-based methods)
        - Trimming, Capping/Winsorization, Discritization
    - **Missing values Imputation**
        - Univariate Imputation (Panda's `fillna()` method, Sklearn's `SimpleImputer()` transformer)
        - Multivariate Imputation (Sklearn's `IterativeImputer()` and `KNNImputer()` transformers)
    - **Encoding Categorical Features**
        - Encode Nominal i/p features using Pandas `get_dummies()` and Scikit-Learn's `OneHotEncoder()`
        - Encode Ordinal i/p features using Scikit-Learn's `OrdinalEncoder()`
        - Encode categorical o/p label using Scikit-Learn's `LabelEncoder()`
    - **Feature Scaling**
        - Use numPy to perform maxabs, minmax, standard and robust scaling
        - Use Sklearn's `MaxAbsScaler` , `MinMaxScaler` , `StandardScaler` , `RobustScalar` transformers
    - **Extracting Information**
        - Use Sklearn's `CountVectorizer` , `DictVectorizer` , `TfidfVectorizer` , and `TfidfTransformer`
    - **Combining Information**
        - Use `FeatureUnion` , `Pipeline` , `PCA`

What are Outliers and How to Deal with them?

An outlier is a data point that differs significantly from other observations

- Every data science project starts with collection of data and that is where outliers are introduced in your dataset. An outlier can be a result of a mistake during data collection in which case it must be handled appropriately. An outlier can be a real exception, which is just an indication of variance in your data. For example, if your machine learning model is going to detect anomalies in a banking transaction, then you must not remove the outliers because this is what the model will be looking for later. So, remember that even if a data point is an outlier, its still a data point! Carefully consider your data, its sources, and your goals whenver deciding to remove an outlier. Each case is different! All distance based models are sensitive to outliers, while tree based algorithms are insensitive to outliers.
- **How to identify outliers?**
  - Z Score Method
  - IQR Method
  - Percentiles Method
  - Multivariate Analysis using Scatter Plot

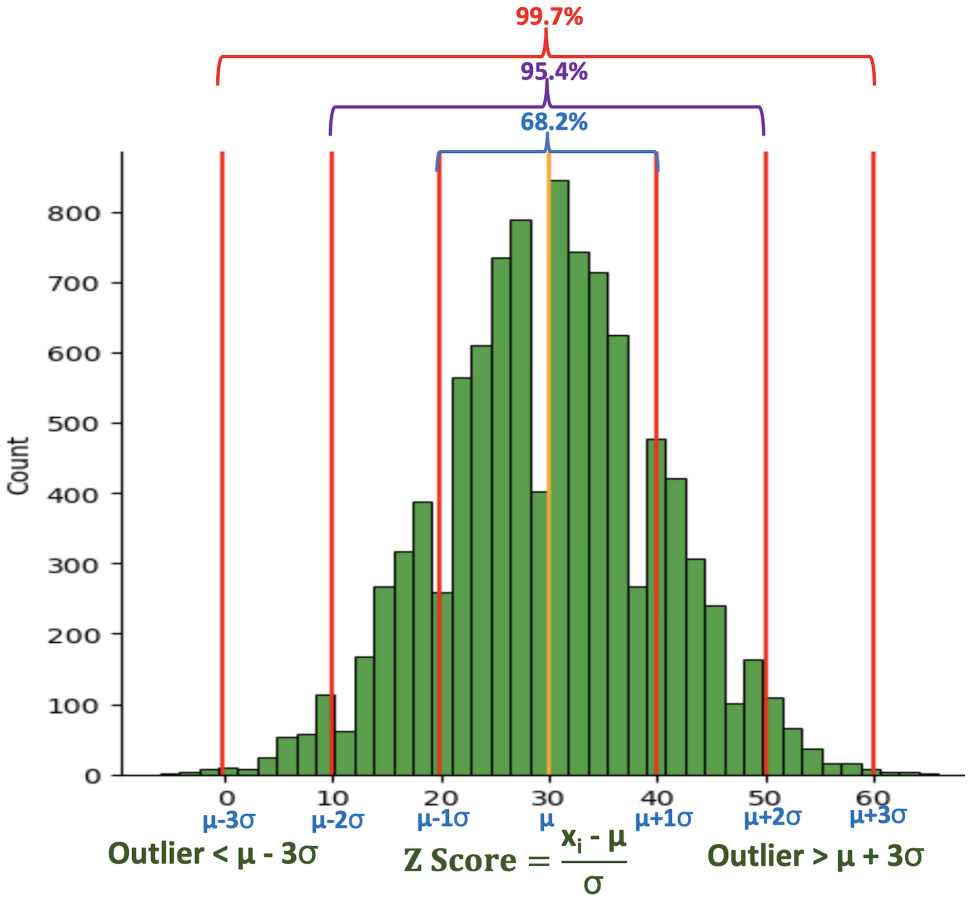


- **How to treat outliers?**
  - **Trimming**
  - **Capping/Winsorization**
  - **Discretization**:

1. Z-Score Method to Indentify Outliers

- The Z-score method should be applied only on columns whose data is almost normally distributed.
- You can calculate the upper and lower limits using any of the following formulas:

$$x_i < \mu - 3\sigma \qquad \text{OR} \qquad x_i > \mu + 3\sigma$$



$$\frac{x_i - \mu}{\sigma} < -3 \quad \text{OR} \quad \frac{x_i - \mu}{\sigma} > 3$$

- After you have identified the outliers under a column/feature:

a. Load Dataset

In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
df = pd.read_csv('datasets/outliers.csv')
df
```

Out[1]:

	x1	x2	y
0	11.0	0.140452	11.0
1	19.0	0.473867	15.0
2	31.0	1.713352	20.0
3	16.0	0.888953	13.0
4	36.0	0.118194	23.0
...	...	...	...
9995	30.0	1.705223	20.0
9996	28.0	2.444101	19.0
9997	37.0	1.089873	24.0
9998	36.0	0.776210	23.0
9999	31.0	0.817765	20.0

10000 rows × 3 columns

b. Check the Distributions

In [2]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   x1      10000 non-null  float64
 1   x2      10000 non-null  float64
 2   y       10000 non-null  float64
dtypes: float64(3)
memory usage: 234.5 KB
```

In [3]:

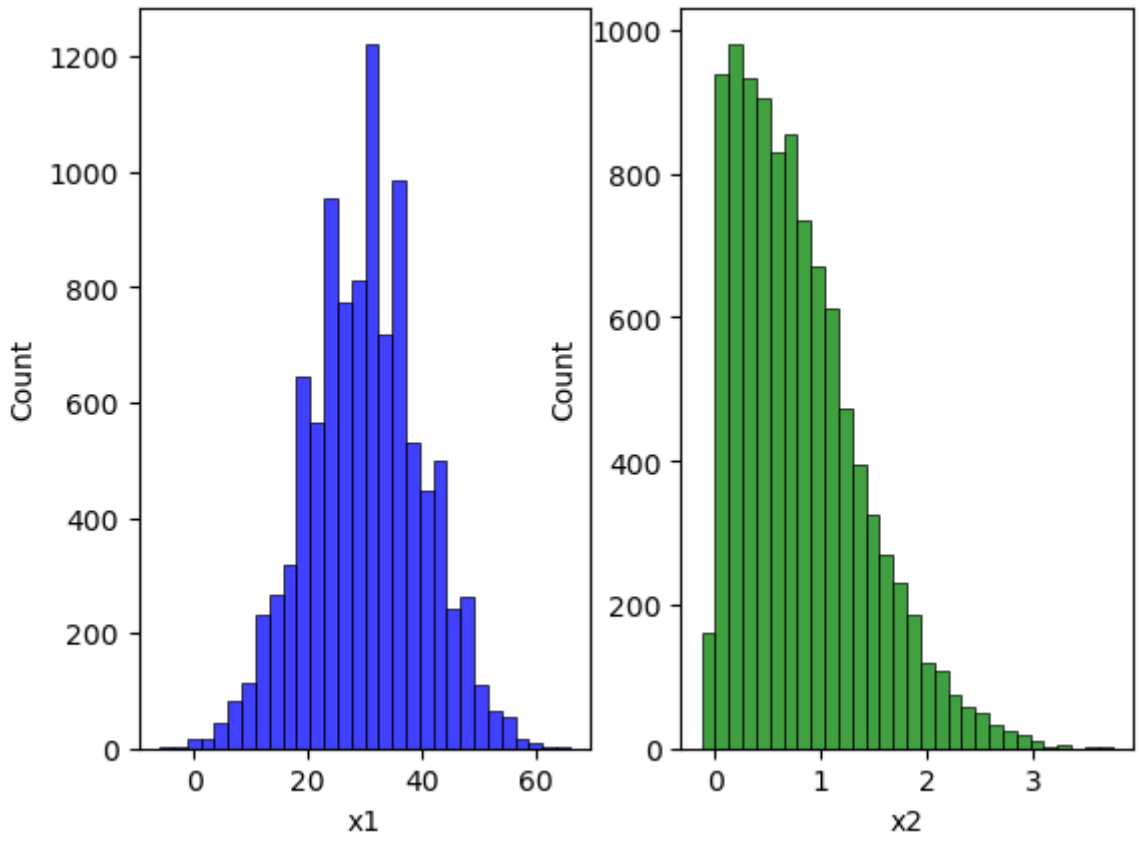
```
1 df.describe()
```

Out[3]:

	x1	x2	y
count	10000.0000	10000.000000	10000.000000
mean	29.9199	0.795030	19.959300
std	10.0421	0.599598	5.034697
min	-6.0000	-0.127513	2.000000
25%	23.0000	0.318087	17.000000
50%	30.0000	0.688281	20.000000
75%	37.0000	1.137827	23.000000
max	66.0000	3.748642	38.000000

In [4]:

```
1 fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2)
2 sns.histplot(x='x1', data = df, bins=30, color='blue', ax=ax1)
3 sns.histplot(x='x2', data=df, bins=30, color='green', ax=ax2)
4 plt.show();
```

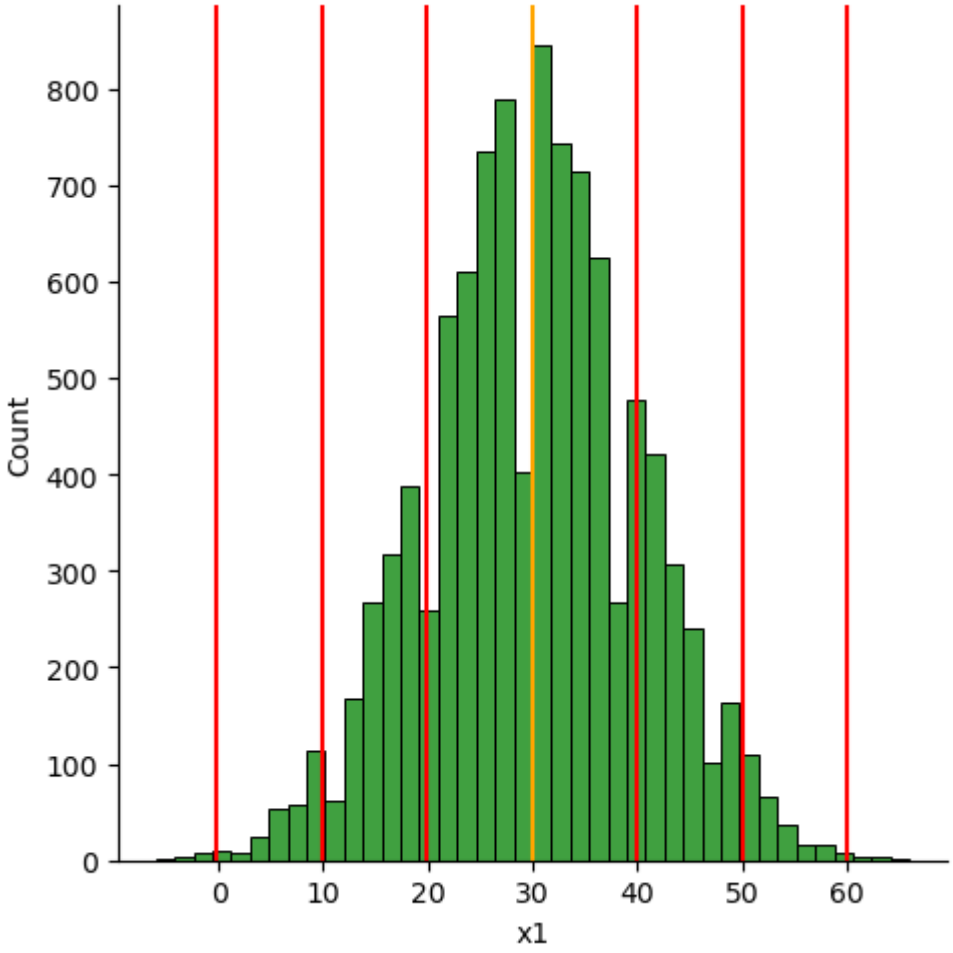


In [5]:

```
mu = df.x1.mean()
sigma = df.x1.std()
minimum = df.x1.min()
maximum = df.x1.max()
print("Mean: ", mu)
print("Std Dev: ", sigma)
print("Min: ", minimum)
print("Max: ", maximum)

sns.displot(x='x1',data=df, kind='hist', bins=40, color='green')
plt.axvline(df.x1.mean(), color='orange')
for i in [-3, -2, -1, 1, 2, 3]:
    plt.axvline(mu+i*sigma, color='red')
plt.show();
```

```
Mean:  29.9199
Std Dev:  10.042099798689334
Min:  -6.0
Max:  66.0
```



c. Identify Outliers

Option 1: Find values which are below $\mu - 3\sigma$ or above $\mu + 3\sigma$

In [6]:

```
#Finding boundry values
lower_limit = mu - 3*sigma
upper_limit = mu + 3*sigma
print("Lower Limit", lower_limit)
print("Upper Limit", upper_limit)
```

```
Lower Limit -0.20639939606800084
Upper Limit 60.046199396068
```

In [7]:

```
df[df.x1>upper_limit]
```

Out[7]:

	x1	x2	y
1322	63.0	1.351085	36.0
2195	64.0	1.069022	37.0
2411	63.0	0.032994	36.0
5544	62.0	1.907387	36.0
6171	61.0	0.694187	35.0
6668	66.0	0.920489	38.0
9500	61.0	1.635512	35.0

In [8]:

```
df[df.x1 < lower_limit]
```

Out[8]:

	x1	x2	y
765	-1.0	0.158337	5.0
825	-1.0	1.095189	5.0
1138	-2.0	0.044623	4.0
3493	-1.0	-0.001617	5.0
6639	-4.0	2.498395	3.0
6707	-1.0	1.630756	4.0
7116	-3.0	0.292362	4.0
7825	-4.0	0.053568	3.0
8832	-6.0	0.885744	2.0
9443	-2.0	0.263597	4.0
9704	-1.0	0.742121	5.0

In [9]:

```
1 df[(df.x1 > upper_limit)  |  (df.x1 < lower_limit)]
```

Out[9]:

	x1	x2	y
765	-1.0	0.158337	5.0
825	-1.0	1.095189	5.0
1138	-2.0	0.044623	4.0
1322	63.0	1.351085	36.0
2195	64.0	1.069022	37.0
2411	63.0	0.032994	36.0
3493	-1.0	-0.001617	5.0
5544	62.0	1.907387	36.0
6171	61.0	0.694187	35.0
6639	-4.0	2.498395	3.0
6668	66.0	0.920489	38.0
6707	-1.0	1.630756	4.0
7116	-3.0	0.292362	4.0
7825	-4.0	0.053568	3.0
8832	-6.0	0.885744	2.0
9443	-2.0	0.263597	4.0
9500	61.0	1.635512	35.0
9704	-1.0	0.742121	5.0

Option 2: Find $Z = \frac{x_i - \mu}{\sigma}$ of corresponding `x1` values and ouliers will be values where $Z < -3 \quad \text{OR} \quad Z > 3$

In [10]:

```
df1 = df.copy()
df1['z_x1'] = (df['x1'] - df['x1'].mean()) / df['x1'].std()
df1
```

Out[10]:

	x1	x2	y	z_x1
0	11.0	0.140452	11.0	-1.884058
1	19.0	0.473867	15.0	-1.087412
2	31.0	1.713352	20.0	0.107557
3	16.0	0.888953	13.0	-1.386154
4	36.0	0.118194	23.0	0.605461
...	...	...	...	...
9995	30.0	1.705223	20.0	0.007976
9996	28.0	2.444101	19.0	-0.191185
9997	37.0	1.089873	24.0	0.705042
9998	36.0	0.776210	23.0	0.605461
9999	31.0	0.817765	20.0	0.107557

10000 rows × 4 columns

In [11]:

```
df1[(df1.z_x1 < -3)  |  (df1.z_x1 > 3)]
```

Out[11]:

	x1	x2	y	z_x1
765	-1.0	0.158337	5.0	-3.079027
825	-1.0	1.095189	5.0	-3.079027
1138	-2.0	0.044623	4.0	-3.178608
1322	63.0	1.351085	36.0	3.294142
2195	64.0	1.069022	37.0	3.393722
2411	63.0	0.032994	36.0	3.294142
3493	-1.0	-0.001617	5.0	-3.079027
5544	62.0	1.907387	36.0	3.194561
6171	61.0	0.694187	35.0	3.094980
6639	-4.0	2.498395	3.0	-3.377770
6668	66.0	0.920489	38.0	3.592884
6707	-1.0	1.630756	4.0	-3.079027
7116	-3.0	0.292362	4.0	-3.278189
7825	-4.0	0.053568	3.0	-3.377770
8832	-6.0	0.885744	2.0	-3.576931
9443	-2.0	0.263597	4.0	-3.178608
9500	61.0	1.635512	35.0	3.094980
9704	-1.0	0.742121	5.0	-3.079027

d. What to do with the Outliers

- **Trimming**
- **Capping (Winsorization)**
- **Discretization:**

(i) Remove the outliers (Trimming)

Option 1: Remove values which are below $\mu - 3\sigma$ or above $\mu + 3\sigma$

In [12]:

```
new_df = df[(df.x1 <= upper_limit)  &  (df.x1 >= lower_limit)]
new_df
```

Out[12]:

	x1	x2	y
0	11.0	0.140452	11.0
1	19.0	0.473867	15.0
2	31.0	1.713352	20.0
3	16.0	0.888953	13.0
4	36.0	0.118194	23.0
...	...	...	...
9995	30.0	1.705223	20.0
9996	28.0	2.444101	19.0
9997	37.0	1.089873	24.0
9998	36.0	0.776210	23.0
9999	31.0	0.817765	20.0

9982 rows × 3 columns

Option 2: Remove values where $Z < -3 \quad \text{OR} \quad Z > 3$

In [13]:

```
df1
```

Out[13]:

	x1	x2	y	z_x1
0	11.0	0.140452	11.0	-1.884058
1	19.0	0.473867	15.0	-1.087412
2	31.0	1.713352	20.0	0.107557
3	16.0	0.888953	13.0	-1.386154
4	36.0	0.118194	23.0	0.605461
...	...	...	...	...
9995	30.0	1.705223	20.0	0.007976
9996	28.0	2.444101	19.0	-0.191185
9997	37.0	1.089873	24.0	0.705042
9998	36.0	0.776210	23.0	0.605461
9999	31.0	0.817765	20.0	0.107557

10000 rows × 4 columns

In [14]:

```
new_df1 = df1[(df1.z_x1 <= 3)  &  (df1.z_x1 >= -3)]
new_df1
```

Out[14]:

	x1	x2	y	z_x1
0	11.0	0.140452	11.0	-1.884058
1	19.0	0.473867	15.0	-1.087412
2	31.0	1.713352	20.0	0.107557
3	16.0	0.888953	13.0	-1.386154
4	36.0	0.118194	23.0	0.605461
...	...	...	...	...
9995	30.0	1.705223	20.0	0.007976
9996	28.0	2.444101	19.0	-0.191185
9997	37.0	1.089873	24.0	0.705042
9998	36.0	0.776210	23.0	0.605461
9999	31.0	0.817765	20.0	0.107557

9982 rows × 4 columns

In [15]:

```
new_df1 = new_df1.drop('z_x1',axis=1)
new_df1
```

Out[15]:

	x1	x2	y
0	11.0	0.140452	11.0
1	19.0	0.473867	15.0
2	31.0	1.713352	20.0
3	16.0	0.888953	13.0
4	36.0	0.118194	23.0
...	...	...	...
9995	30.0	1.705223	20.0
9996	28.0	2.444101	19.0
9997	37.0	1.089873	24.0
9998	36.0	0.776210	23.0
9999	31.0	0.817765	20.0

9982 rows × 3 columns

(ii) Replace outlier with upper value (Capping/Winsorization)

- `np.where(condition,a,b)`
- If condition is true then return `a` else return `b`

In [16]:

```
capped_df = df.copy()
capped_df['x1'] = np.where(df['x1'] < lower_limit, lower_limit,
                    (np.where(df['x1'] > upper_limit, upper_limit, df['x1']))
                    )
capped_df
```

Out[16]:

	x1	x2	y
0	11.0	0.140452	11.0
1	19.0	0.473867	15.0
2	31.0	1.713352	20.0
3	16.0	0.888953	13.0
4	36.0	0.118194	23.0
...	...	...	...
9995	30.0	1.705223	20.0
9996	28.0	2.444101	19.0
9997	37.0	1.089873	24.0
9998	36.0	0.776210	23.0
9999	31.0	0.817765	20.0

10000 rows × 3 columns

Verify

In [17]:

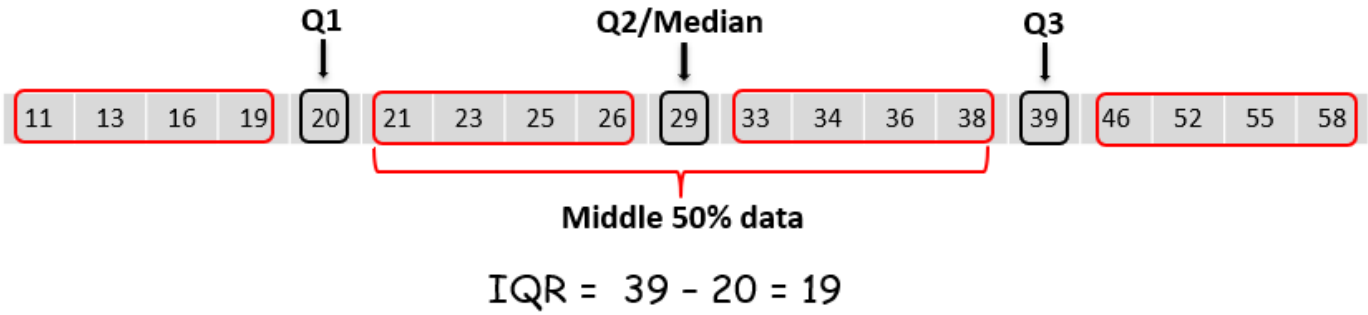
```
1 capped_df.describe()
```

Out[17]:

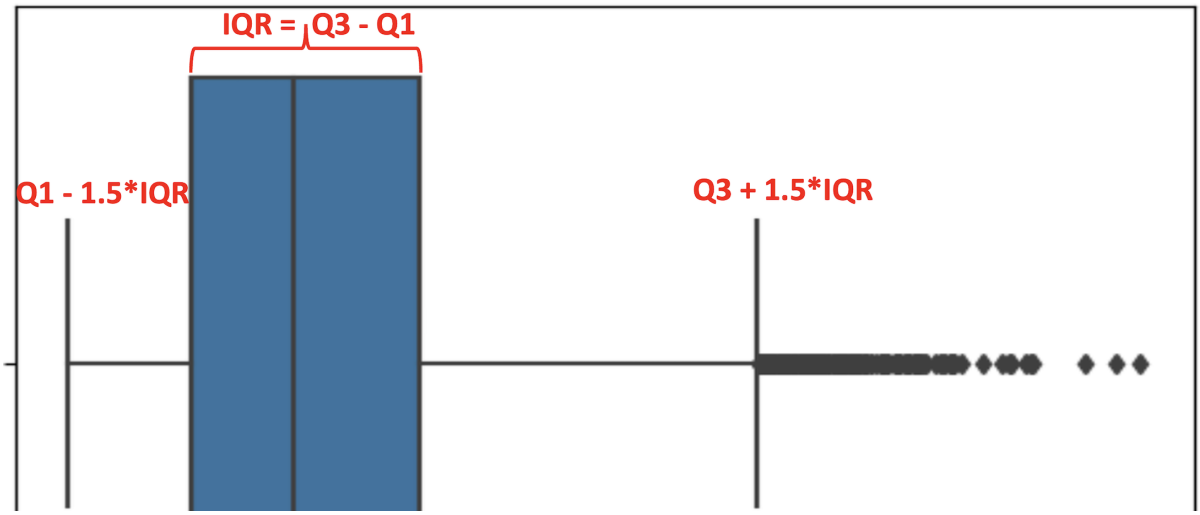
	x1	x2	y
count	10000.000000	10000.000000	10000.000000
mean	29.920305	0.795030	19.959300
std	10.028301	0.599598	5.034697
min	-0.206399	-0.127513	2.000000
25%	23.000000	0.318087	17.000000
50%	30.000000	0.688281	20.000000
75%	37.000000	1.137827	23.000000
max	60.046199	3.748642	38.000000

2. IQR Method to Identify Outliers

- Quantiles divides a distribution and the most common are `median` , `quartiles` , `deciles` , and `percentiles` .
- - **Quartiles:**, as their name suggests, are quantiles that divide a distribution into quarters by splitting a rank-ordered dataset into four equal parts.
    - 1st Quartile Q1 is the same as the 25th percentile.
    - 2nd Quartile Q2 is the same as 50th percentile.
    - 3rd Quratile Q3 is same as 75th percentile
- Inter Quartile Range is the difference between the third quartile(Q3) and the first Quartile (Q1)



- A boxplot is a method that is used to graphically represent numerical data through their quartiles.
- A boxplot is a very simple but effective way to visualize outliers. Think about the lower and upper whiskers as the boundaries of the data distribution. Any data points that show above or below the whiskers, can be considered outliers or anomalous.
- Outliers in this case are defined as the observations that are below (Q1 − 1.5x IQR) or boxplot lower whisker or above (Q3 + 1.5x IQR) or boxplot upper whisker.



a. Load Dataset

In [18]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
df = pd.read_csv('datasets/outliers.csv')
df
```

Out[18]:

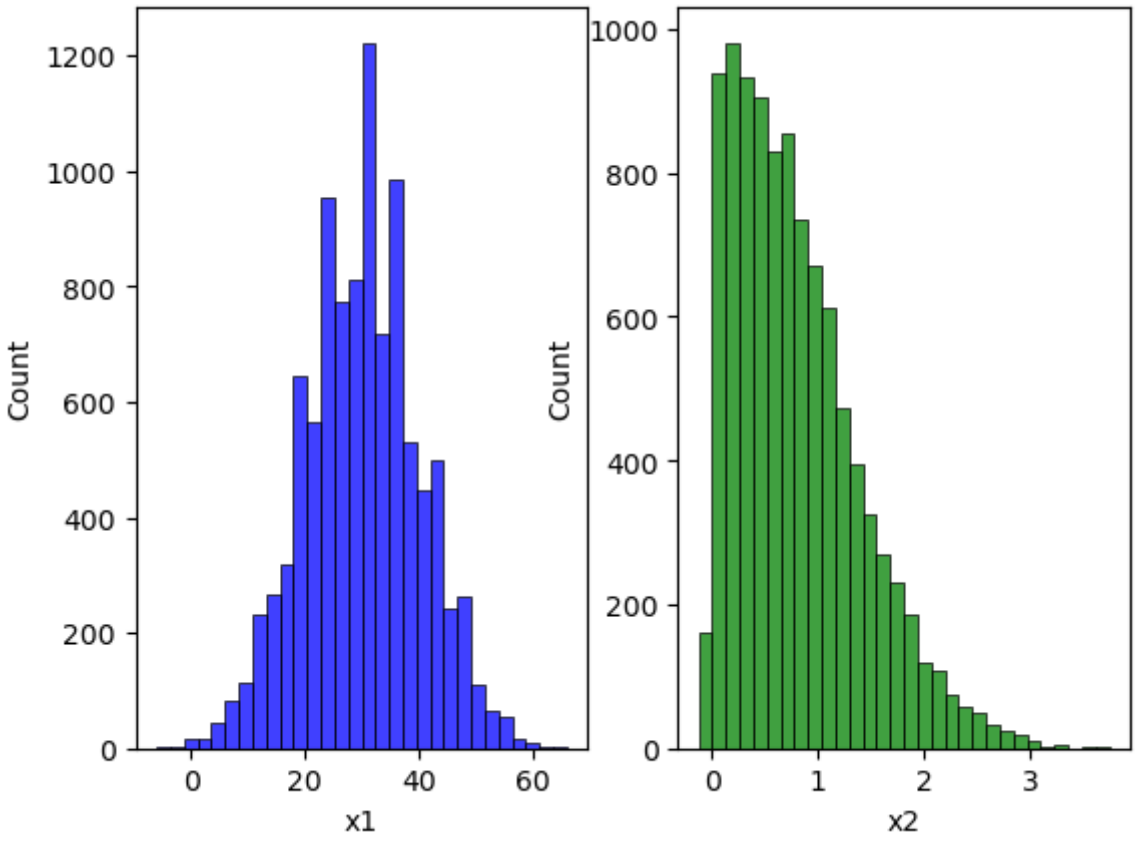
	x1	x2	y
0	11.0	0.140452	11.0
1	19.0	0.473867	15.0
2	31.0	1.713352	20.0
3	16.0	0.888953	13.0
4	36.0	0.118194	23.0
...	...	...	...
9995	30.0	1.705223	20.0
9996	28.0	2.444101	19.0
9997	37.0	1.089873	24.0
9998	36.0	0.776210	23.0
9999	31.0	0.817765	20.0

10000 rows × 3 columns

b. Check the Distributions

In [19]:

```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2)
sns.histplot(x='x1', data = df, bins=30, color='blue', ax=ax1)
sns.histplot(x='x2', data=df, bins=30, color='green', ax=ax2)
plt.show();
```

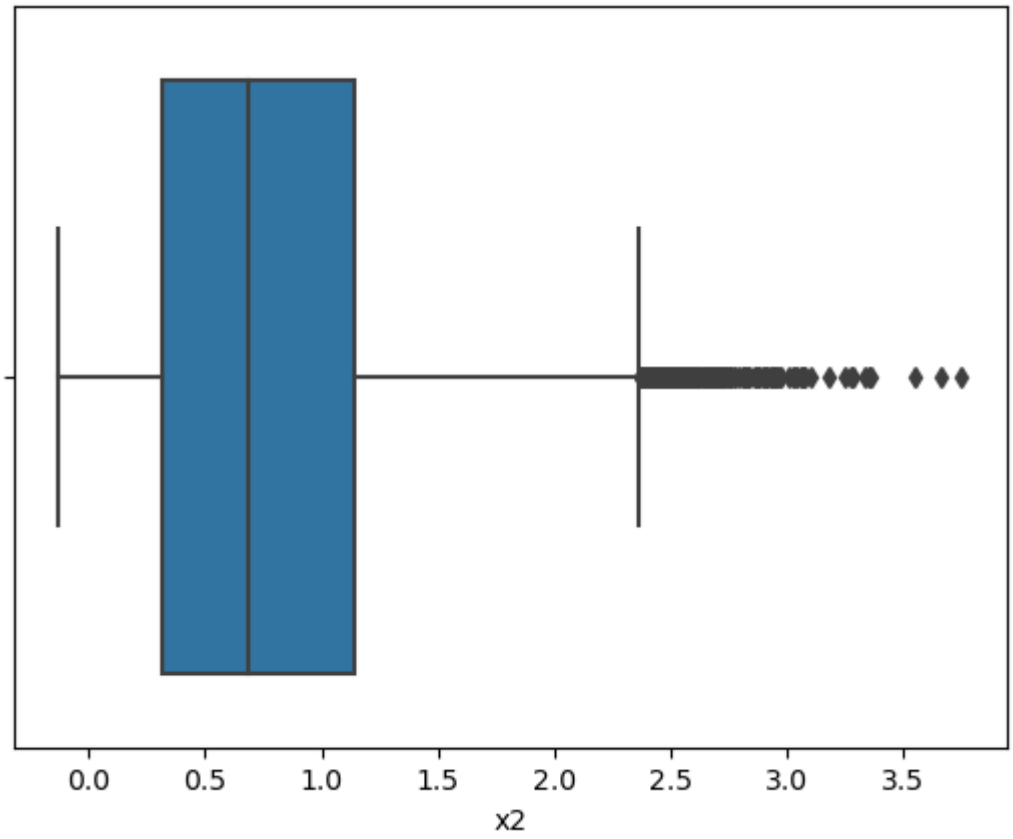


c. Identify Outliers

Draw the BoxPlot:

In [20]:

```
1 import seaborn as sns
2 sns.boxplot(x=df['x2'], data=df);
```



Calculate IQR:

In [21]:

```
1 q1 = df['x2'].quantile(0.25)
2 q3 = df['x2'].quantile(0.75)
3 iqr = q3 - q1
4 iqr
```

Out[21]:

```
0.8197399004172614
```

Calculate Lower and Upper Limit:

In [22]:

```
1 lower_limit = q1 - 1.5*iqr
2 upper_limit = q3  + 1.5*iqr
3 lower_limit, upper_limit
```

Out[22]:

```
(-0.9115229048744358, 2.36743669679461)
```

Find Outliers: Below $Q1 - 1.5 \times \text{IQR}$ or above $Q3 + 1.5 * \text{IQR})$

In [23]:

```
df[df['x2'] < lower_limit]
```

Out[23]:

x1	x2	y

In [24]:

```
df[df['x2'] > upper_limit]
```

Out[24]:

	x1	x2	y
11	32.0	2.674464	21.0
35	25.0	2.856379	18.0
80	36.0	2.579868	23.0
147	18.0	2.976488	14.0
148	39.0	2.547117	24.0
...	...	...	...
9716	36.0	2.548617	23.0
9784	32.0	2.467331	21.0
9836	47.0	2.735268	29.0
9982	41.0	2.579395	26.0
9996	28.0	2.444101	19.0

180 rows × 3 columns

d. What to do with the Outliers

(i) Remove the outliers (Trimming)

In [25]:

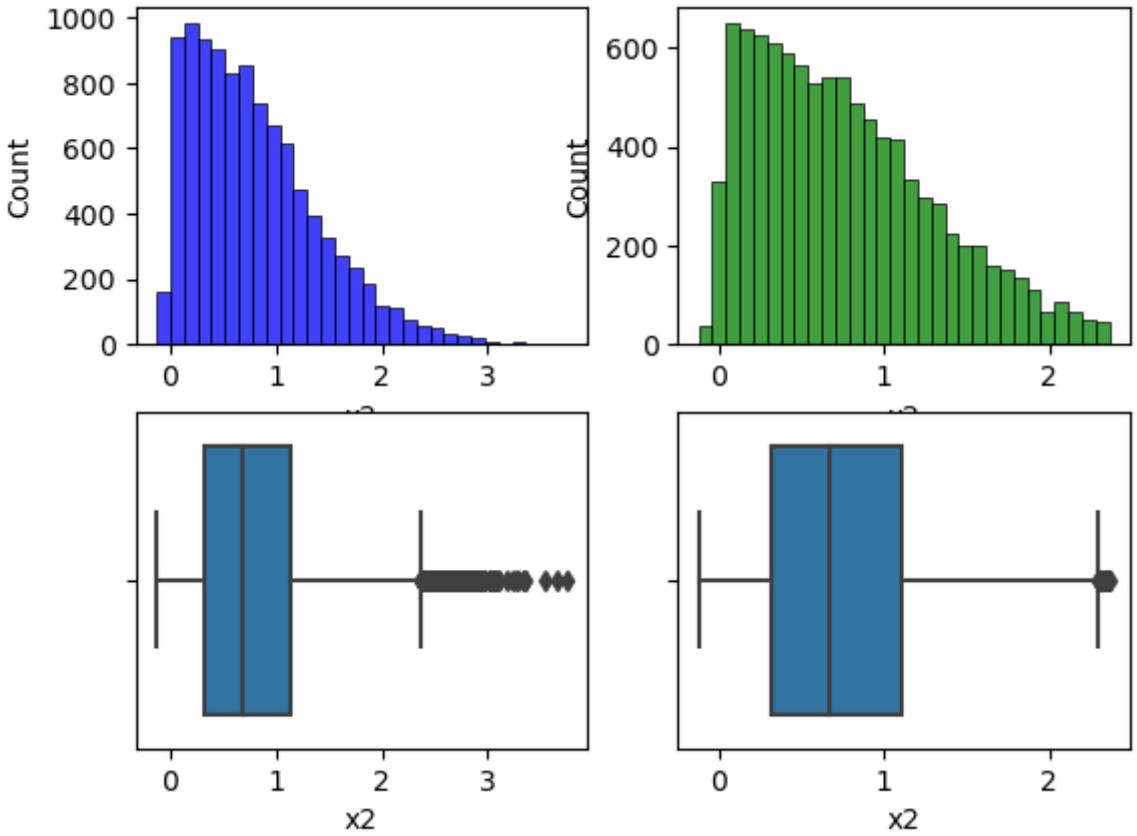
```
new_df = df[(df.x2 >= lower_limit)  &  (df.x2 <= upper_limit)]
new_df.shape
```

Out[25]:

```
(9820, 3)
```

In [26]:

```
# Compare the results
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2)
sns.histplot(x='x2', data=df, bins=30, color='blue', ax=ax1)
sns.histplot(x='x2', data=new_df, bins=30, color='green', ax=ax2)
sns.boxplot(x='x2', data=df, ax=ax3)
sns.boxplot(x='x2', data=new_df, ax=ax4);
```



(ii) Replace outlier with upper value (Capping/Winsorization)

- `np.where(condition,a,b)`
- If condition is true then return `a` else return `b`

In [27]:

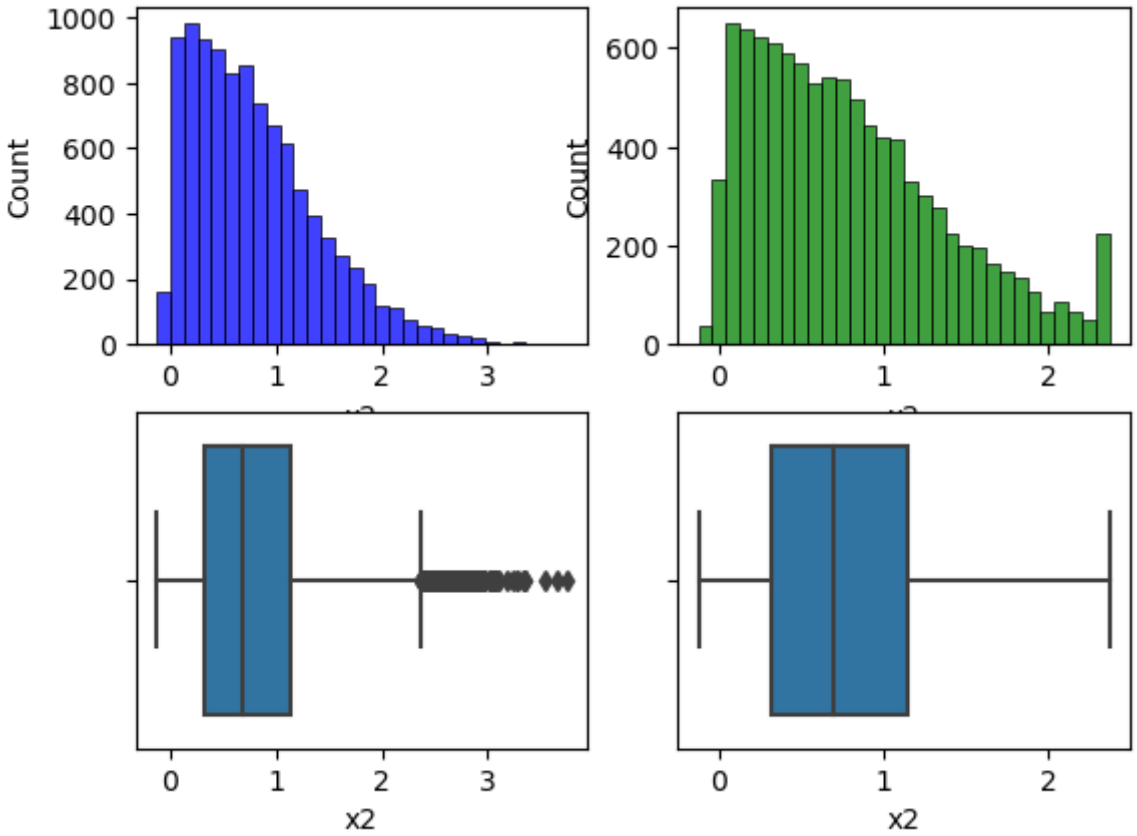
```
capped_df = df.copy()
capped_df['x2'] = np.where(df['x2'] < lower_limit, lower_limit,
                    (np.where(df['x2'] > upper_limit, upper_limit, df['x2']))
                    )
capped_df.shape
```

Out[27]:

```
(10000, 3)
```

In [28]:

```
# Compare the results
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2)
sns.histplot(x='x2', data=df, bins=30, color='blue', ax=ax1)
sns.histplot(x='x2', data=capped_df, bins=30, color='green', ax=ax2)
sns.boxplot(x='x2', data=df, ax=ax3)
sns.boxplot(x='x2', data=capped_df, ax=ax4);
```



3. Percentile Method to Identify Outliers

- For `Normally distributed` data we have used empirical relations of Normal distribution, which says that the data points which fall below $\mu - 3\sigma$ or above $\mu + 3\sigma$ are outliers.
- For `Skewed distributions` , we use Inter-Quartile Range (IQR) proximity rule, which says that the data points which fall below $Q1 - 1.5 \times IQR$ or above $Q3 + 1.5 \times IQR$ are outliers.
- For `Other distributions` , we use **Percentile-based approach**, in which you specify a particular threshold value (percentile) below and above which all data points are considered as outliers.

a. Load Dataset

In [29]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
df = pd.read_csv('datasets/outliers.csv')
df
```

Out[29]:

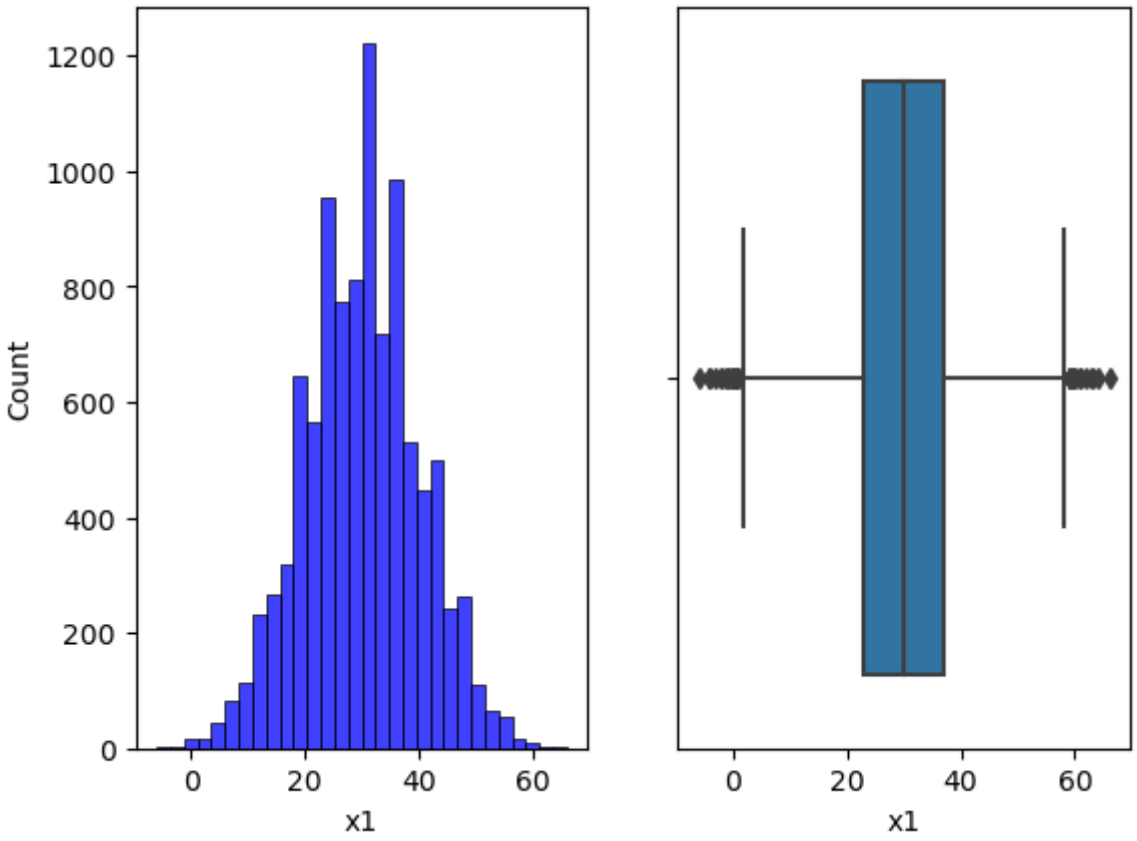
	x1	x2	y
0	11.0	0.140452	11.0
1	19.0	0.473867	15.0
2	31.0	1.713352	20.0
3	16.0	0.888953	13.0
4	36.0	0.118194	23.0
...	...	...	...
9995	30.0	1.705223	20.0
9996	28.0	2.444101	19.0
9997	37.0	1.089873	24.0
9998	36.0	0.776210	23.0
9999	31.0	0.817765	20.0

10000 rows × 3 columns

b. Check the Distributions

In [30]:

```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2)
sns.histplot(x='x1', data = df, bins=30, color='blue', ax=ax1)
sns.boxplot(x=df['x1'], ax=ax2)
plt.show();
```



c. Identify Outliers

- Assume that all values below 1% and 99% are outliers

In [31]:

```
lower_limit = df['x1'].quantile(0.01)
upper_limit = df['x1'].quantile(0.99)

lower_limit, upper_limit
```

Out[31]:

```
(6.0, 53.0)
```

d. What to do with the Outliers

(i) Remove the outliers (Trimming)

In [32]:

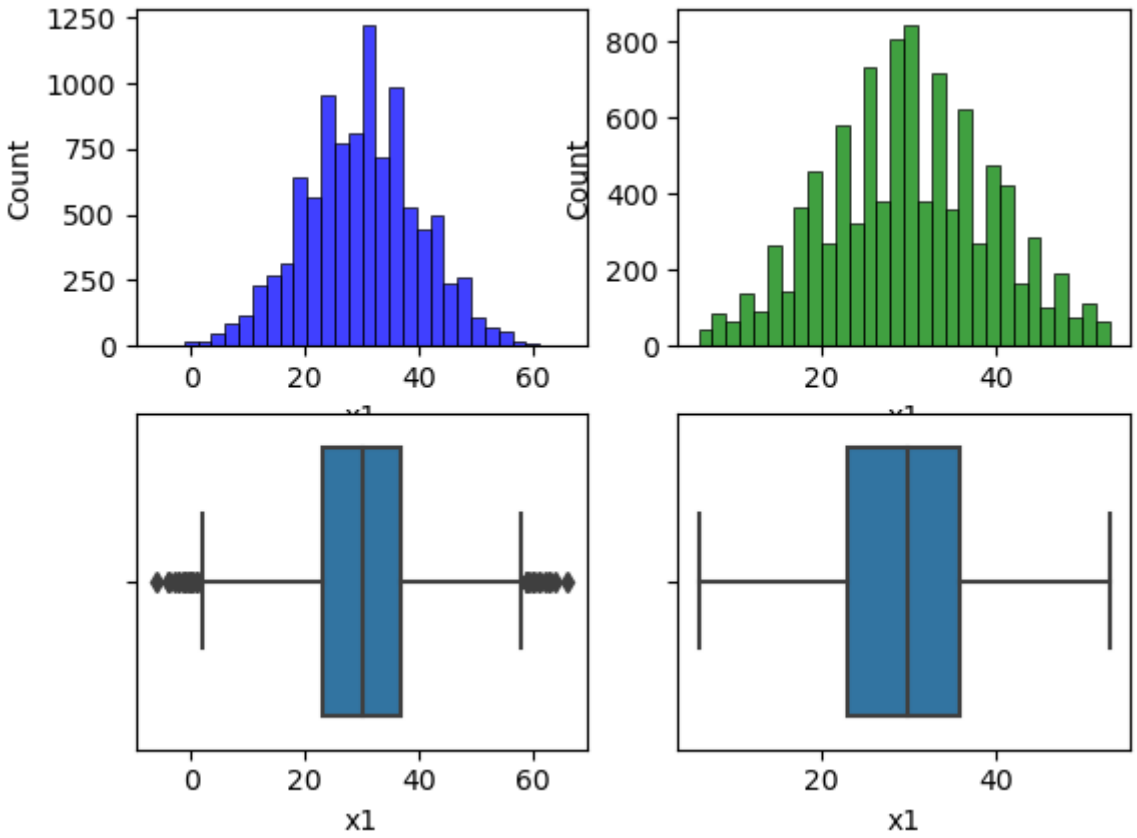
```
new_df = df[(df.x1 >= lower_limit)  &  (df.x1 <= upper_limit)]
new_df.shape
```

Out[32]:

```
(9835, 3)
```

In [33]:

```
# Compare the results
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2)
sns.histplot(x='x1', data=df, bins=30, color='blue', ax=ax1)
sns.histplot(x='x1', data=new_df, bins=30, color='green', ax=ax2)
sns.boxplot(x='x1', data=df, ax=ax3)
sns.boxplot(x='x1', data=new_df, ax=ax4);
```



(ii) Replace outlier with upper value (Capping/Winsorization)

- `np.where(condition,a,b)`
- If condition is true then return `a` else return `b`

In [34]:

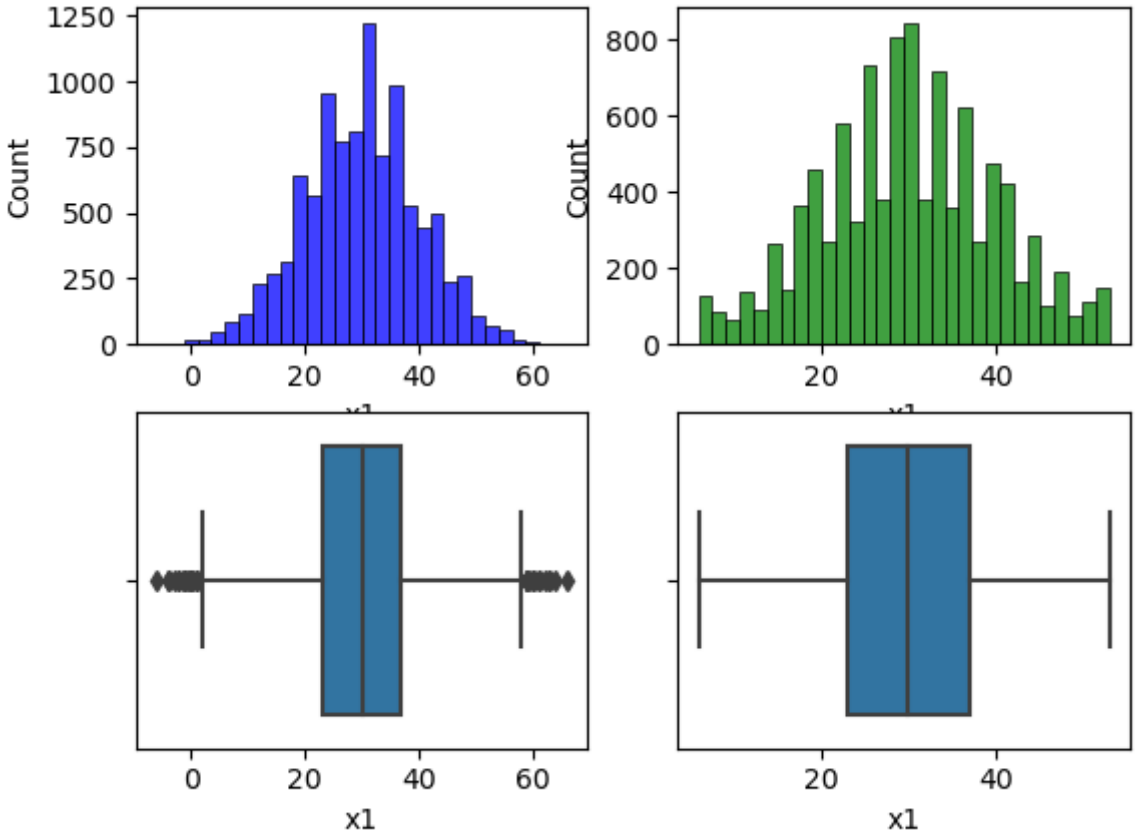
```
capped_df = df.copy()
capped_df['x1'] = np.where(df['x1'] < lower_limit, lower_limit,
                    (np.where(df['x1'] > upper_limit, upper_limit, df['x1']))
                    )
capped_df.shape
```

Out[34]:

```
(10000, 3)
```

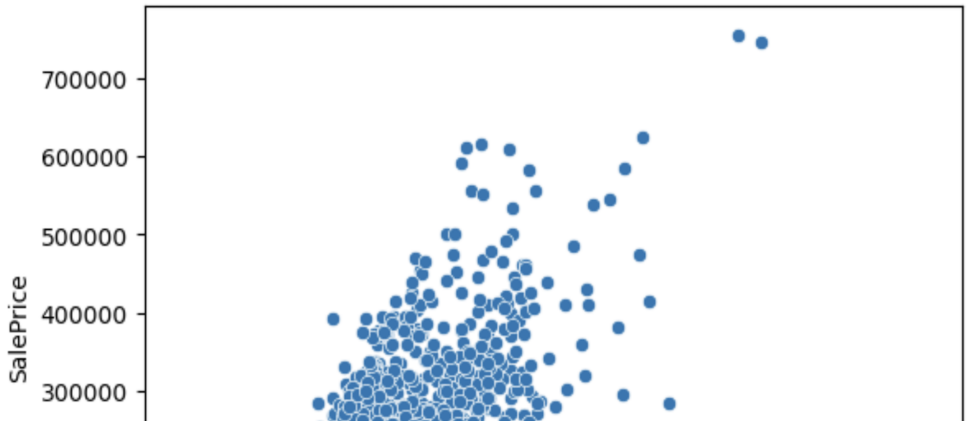
In [35]:

```
# Compare the results
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2)
sns.histplot(x='x1', data=df, bins=30, color='blue', ax=ax1)
sns.histplot(x='x1', data=capped_df, bins=30, color='green', ax=ax2)
sns.boxplot(x='x1', data=df, ax=ax3)
sns.boxplot(x='x1', data=capped_df, ax=ax4);
```



4. Multi-Variate Analysis to Handle Outliers

- We use multivariate methods to find the outliers that only show up within combinations of observations from two or more different variables.
  - Scatterplot matrices
  - Boxplots
- There are a bundle of methods that you may come accross in literature like:
  - Depth-based methods
  - Distance-based methods
  - Density-based methods
  - Mahalanobis distance based methods



a. Use Scatterplot

Load Dataset

In [36]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
df = pd.read_csv('datasets/Ames_Housing_Data.csv')
df
```

Out[36]:

	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilities	...
0	526301100	20	RL	141.0	31770	Pave	NaN	IR1	Lvl	AllPub	...
1	526350040	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	...
2	526351010	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	...
3	526353030	20	RL	93.0	11160	Pave	NaN	Reg	Lvl	AllPub	...
4	527105010	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	...
...	...	...	...	...	...	...	...	...	...	...	...
2925	923275080	80	RL	37.0	7937	Pave	NaN	IR1	Lvl	AllPub	...
2926	923276100	20	RL	NaN	8885	Pave	NaN	IR1	Low	AllPub	...
2927	923400125	85	RL	62.0	10441	Pave	NaN	Reg	Lvl	AllPub	...
2928	924100070	20	RL	77.0	10010	Pave	NaN	Reg	Lvl	AllPub	...
2929	924151050	60	RL	74.0	9627	Pave	NaN	Reg	Lvl	AllPub	...

2930 rows × 81 columns

In [47]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 81 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   PID              2930 non-null   int64
 1   MS SubClass      2930 non-null   int64
 2   MS Zoning        2930 non-null   object
 3   Lot Frontage     2440 non-null   float64
 4   Lot Area         2930 non-null   int64
 5   Street           2930 non-null   object
 6   Alley            198 non-null    object
 7   Lot Shape        2930 non-null   object
 8   Land Contour     2930 non-null   object
 9   Utilities        2930 non-null   object
 10  Lot Config       2930 non-null   object
 11  Land Slope       2930 non-null   object
 12  Neighborhood     2930 non-null   object
 13  Condition 1      2930 non-null   object
 14  Condition 2      2930 non-null   object
 15  Bldg Type        2930 non-null   object
 16  House Style      2930 non-null   object
 17  Overall Qual     2930 non-null   int64
 18  Overall Cond     2930 non-null   int64
 19  Year Built       2930 non-null   int64
 20  Year Remod/Add   2930 non-null   int64
 21  Roof Style       2930 non-null   object
 22  Roof Matl        2930 non-null   object
 23  Exterior 1st     2930 non-null   object
 24  Exterior 2nd     2930 non-null   object
 25  Mas Vnr Type     2907 non-null   object
 26  Mas Vnr Area     2907 non-null   float64
 27  Exter Qual       2930 non-null   object
 28  Exter Cond       2930 non-null   object
 29  Foundation       2930 non-null   object
 30  Bsmt Qual        2850 non-null   object
 31  Bsmt Cond        2850 non-null   object
 32  Bsmt Exposure    2847 non-null   object
 33  BsmtFin Type 1   2850 non-null   object
 34  BsmtFin SF 1     2929 non-null   float64
 35  BsmtFin Type 2   2849 non-null   object
 36  BsmtFin SF 2     2929 non-null   float64
 37  Bsmt Unf SF      2929 non-null   float64
 38  Total Bsmt SF    2929 non-null   float64
 39  Heating          2930 non-null   object
 40  Heating QC       2930 non-null   object
 41  Central Air      2930 non-null   object
 42  Electrical       2929 non-null   object
 43  1st Flr SF       2930 non-null   int64
 44  2nd Flr SF       2930 non-null   int64
 45  Low Qual Fin SF  2930 non-null   int64
 46  Gr Liv Area      2930 non-null   int64
 47  Bsmt Full Bath   2928 non-null   float64
 48  Bsmt Half Bath   2928 non-null   float64
 49  Full Bath        2930 non-null   int64
 50  Half Bath        2930 non-null   int64
 51  Bedroom AbvGr    2930 non-null   int64
 52  Kitchen AbvGr    2930 non-null   int64
 53  Kitchen Qual     2930 non-null   object
 54  TotRms AbvGrd    2930 non-null   int64
 55  Functional       2930 non-null   object
```

```
 56  Fireplaces       2930 non-null   int64
 57  Fireplace Qu     1508 non-null   object
 58  Garage Type      2773 non-null   object
 59  Garage Yr Blt    2771 non-null   float64
 60  Garage Finish    2771 non-null   object
 61  Garage Cars      2929 non-null   float64
 62  Garage Area      2929 non-null   float64
 63  Garage Qual      2771 non-null   object
 64  Garage Cond      2771 non-null   object
 65  Paved Drive      2930 non-null   object
 66  Wood Deck SF     2930 non-null   int64
 67  Open Porch SF    2930 non-null   int64
 68  Enclosed Porch   2930 non-null   int64
 69  3Ssn Porch       2930 non-null   int64
 70  Screen Porch     2930 non-null   int64
 71  Pool Area        2930 non-null   int64
 72  Pool QC          13 non-null     object
 73  Fence            572 non-null    object
 74  Misc Feature     106 non-null    object
 75  Misc Val         2930 non-null   int64
 76  Mo Sold          2930 non-null   int64
 77  Yr Sold          2930 non-null   int64
 78  Sale Type        2930 non-null   object
 79  Sale Condition   2930 non-null   object
 80  SalePrice        2930 non-null   int64
dtypes: float64(11), int64(27), object(43)
memory usage: 1.8+ MB
```

In [37]:

```
1 df.corr(numeric_only=True)
```

Out[37]:

	PID	MS SubClass	Lot Frontage	Lot Area	Overall Qual	Overall Cond	Year Built	Y Remod/A
PID	1.000000	-0.001281	-0.096918	0.034868	-0.263147	0.104451	-0.343388	-0.157
MS SubClass	-0.001281	1.000000	-0.420135	-0.204613	0.039419	-0.067349	0.036579	0.043
Lot Frontage	-0.096918	-0.420135	1.000000	0.491313	0.212042	-0.074448	0.121562	0.091
Lot Area	0.034868	-0.204613	0.491313	1.000000	0.097188	-0.034759	0.023258	0.021
Overall Qual	-0.263147	0.039419	0.212042	0.097188	1.000000	-0.094812	0.597027	0.569
Overall Cond	0.104451	-0.067349	-0.074448	-0.034759	-0.094812	1.000000	-0.368773	0.047
Year Built	-0.343388	0.036579	0.121562	0.023258	0.597027	-0.368773	1.000000	0.612
Year Remod/Add	-0.157111	0.043397	0.091712	0.021682	0.569609	0.047680	0.612095	1.000
Mas Vnr Area	-0.229283	0.002730	0.222407	0.126830	0.429418	-0.135340	0.313292	0.196
BsmtFin SF 1	-0.098375	-0.060075	0.215583	0.191555	0.284118	-0.050935	0.279870	0.151
BsmtFin SF 2	-0.001145	-0.070946	0.045999	0.083150	-0.041287	0.041134	-0.027415	-0.062
Bsmt Unf SF	-0.087707	-0.130421	0.116743	0.023658	0.270058	-0.136819	0.128998	0.164
Total Bsmt SF	-0.189642	-0.219445	0.353773	0.253589	0.547294	-0.173344	0.407526	0.297
1st Flr SF	-0.141902	-0.247828	0.457391	0.332235	0.477837	-0.157052	0.310463	0.242
2nd Flr SF	-0.003289	0.304237	0.029187	0.032996	0.241402	0.006218	0.016828	0.158
Low Qual Fin SF	0.056940	0.025765	0.005249	0.000812	-0.048680	0.009175	-0.144282	-0.060
Gr Liv Area	-0.107579	0.068061	0.383822	0.285599	0.570556	-0.115643	0.241726	0.316
Bsmt Full Bath	-0.037759	0.013701	0.108915	0.125877	0.167858	-0.042766	0.211849	0.134
Bsmt Half Bath	0.004328	-0.003329	-0.024724	0.026903	-0.041647	0.084455	-0.030626	-0.046
Full Bath	-0.171431	0.134631	0.184521	0.127433	0.522263	-0.214316	0.469406	0.457
Half Bath	-0.166636	0.175879	0.041880	0.035497	0.268853	-0.088127	0.269268	0.211
Bedroom AbvGr	0.006345	-0.019208	0.240442	0.136569	0.063291	-0.006137	-0.055093	-0.021
Kitchen AbvGr	0.076470	0.257698	0.005407	-0.020301	-0.159744	-0.086386	-0.137852	-0.142
TotRms AbvGrd	-0.068981	0.031898	0.353137	0.216597	0.380693	-0.089816	0.111919	0.197
Fireplaces	-0.108056	-0.049955	0.257255	0.256989	0.393007	-0.031702	0.170672	0.133
Garage Yr Blt	-0.256829	0.088754	0.076306	-0.008952	0.570569	-0.326017	0.834849	0.652

	PID	MS SubClass	Lot Frontage	Lot Area	Overall Qual	Overall Cond	Year Built	Y Remod/A
Garage Cars	-0.237484	-0.045883	0.308706	0.179512	0.599545	-0.181557	0.537443	0.425
Garage Area	-0.210606	-0.103239	0.358505	0.212822	0.563503	-0.153754	0.480131	0.376
Wood Deck SF	-0.051135	-0.017310	0.120084	0.157212	0.255663	0.020344	0.228964	0.217
Open Porch SF	-0.071311	-0.014823	0.163040	0.103760	0.298412	-0.068934	0.198365	0.241
Enclosed Porch	0.162519	-0.022866	0.012758	0.021868	-0.140332	0.071459	-0.374364	-0.220
3Ssn Porch	-0.024894	-0.037956	0.028564	0.016243	0.018240	0.043852	0.015803	0.037
Screen Porch	-0.025735	-0.050614	0.076666	0.055044	0.041615	0.044055	-0.041436	-0.046
Pool Area	-0.002845	-0.003434	0.173947	0.093775	0.030399	-0.016787	0.002213	-0.011
Misc Val	-0.008260	-0.029254	0.044476	0.069188	0.005179	0.034056	-0.011011	-0.003
Mo Sold	-0.050455	0.000350	0.011085	0.003859	0.031103	-0.007295	0.014577	0.018
Yr Sold	0.009579	-0.017905	-0.007547	-0.023085	-0.020719	0.031207	-0.013197	0.032
SalePrice	-0.246521	-0.085092	0.357318	0.266549	0.799262	-0.101697	0.558426	0.532

38 rows × 38 columns

In [38]:

```
1 df.corr(numeric_only=True)['SalePrice']
```

Out[38]:

```
PID               -0.246521
MS SubClass       -0.085092
Lot Frontage       0.357318
Lot Area           0.266549
Overall Qual       0.799262
Overall Cond      -0.101697
Year Built         0.558426
Year Remod/Add     0.532974
Mas Vnr Area       0.508285
BsmtFin SF 1       0.432914
BsmtFin SF 2       0.005891
Bsmt Unf SF        0.182855
Total Bsmt SF      0.632280
1st Flr SF         0.621676
2nd Flr SF         0.269373
Low Qual Fin SF   -0.037660
Gr Liv Area        0.706780
Bsmt Full Bath     0.276050
Bsmt Half Bath    -0.035835
Full Bath          0.545604
Half Bath          0.285056
Bedroom AbvGr      0.143913
Kitchen AbvGr     -0.119814
TotRms AbvGrd      0.495474
Fireplaces         0.474558
Garage Yr Blt      0.526965
Garage Cars        0.647877
Garage Area        0.640401
Wood Deck SF       0.327143
Open Porch SF      0.312951
Enclosed Porch    -0.128787
3Ssn Porch         0.032225
Screen Porch       0.112151
Pool Area          0.068403
Misc Val          -0.015691
Mo Sold            0.035259
Yr Sold           -0.030569
SalePrice          1.000000
Name: SalePrice, dtype: float64
```

In [39]:

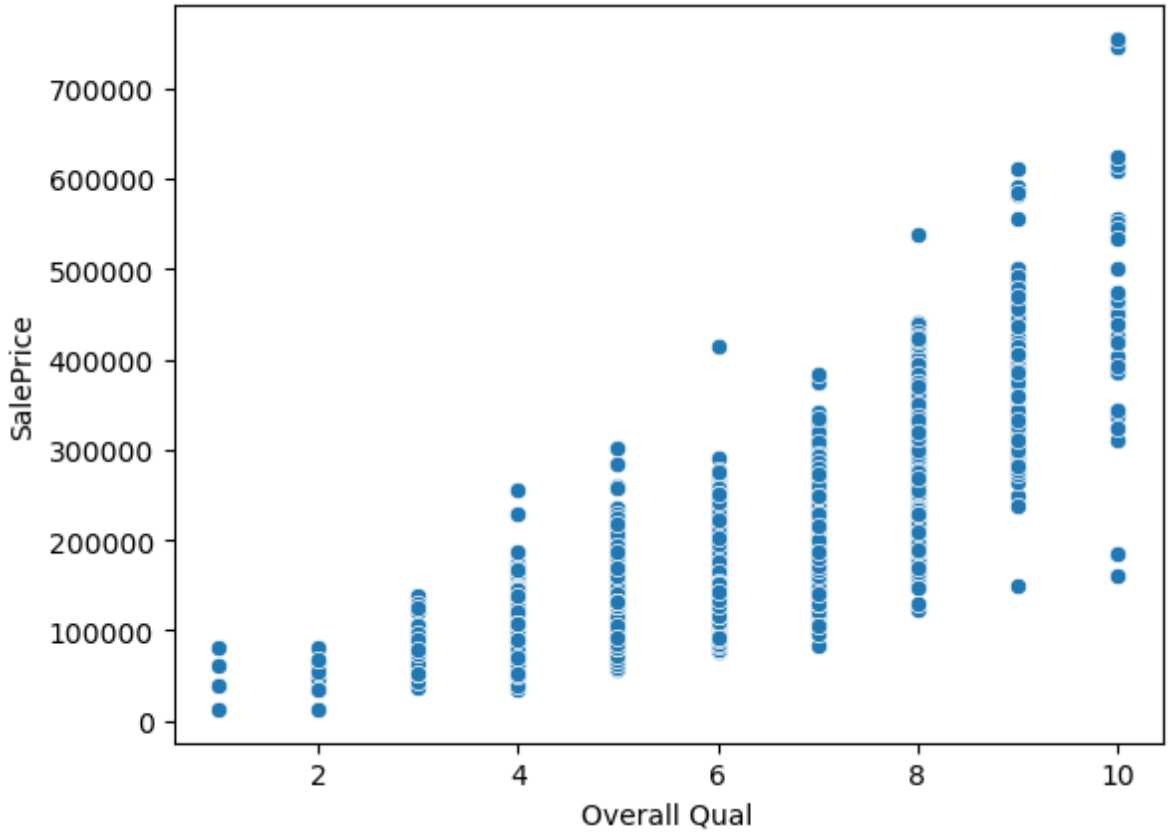
```
1 df.corr(numeric_only=True)['SalePrice'].sort_values()
```

Out[39]:

```
PID               -0.246521
Enclosed Porch    -0.128787
Kitchen AbvGr     -0.119814
Overall Cond      -0.101697
MS SubClass       -0.085092
Low Qual Fin SF   -0.037660
Bsmt Half Bath    -0.035835
Yr Sold           -0.030569
Misc Val          -0.015691
BsmtFin SF 2       0.005891
3Ssn Porch         0.032225
Mo Sold            0.035259
Pool Area          0.068403
Screen Porch       0.112151
Bedroom AbvGr      0.143913
Bsmt Unf SF        0.182855
Lot Area           0.266549
2nd Flr SF         0.269373
Bsmt Full Bath     0.276050
Half Bath          0.285056
Open Porch SF      0.312951
Wood Deck SF       0.327143
Lot Frontage       0.357318
BsmtFin SF 1       0.432914
Fireplaces         0.474558
TotRms AbvGrd      0.495474
Mas Vnr Area       0.508285
Garage Yr Blt      0.526965
Year Remod/Add     0.532974
Full Bath          0.545604
Year Built         0.558426
1st Flr SF         0.621676
Total Bsmt SF      0.632280
Garage Area        0.640401
Garage Cars        0.647877
Gr Liv Area        0.706780
Overall Qual       0.799262
SalePrice          1.000000
Name: SalePrice, dtype: float64
```

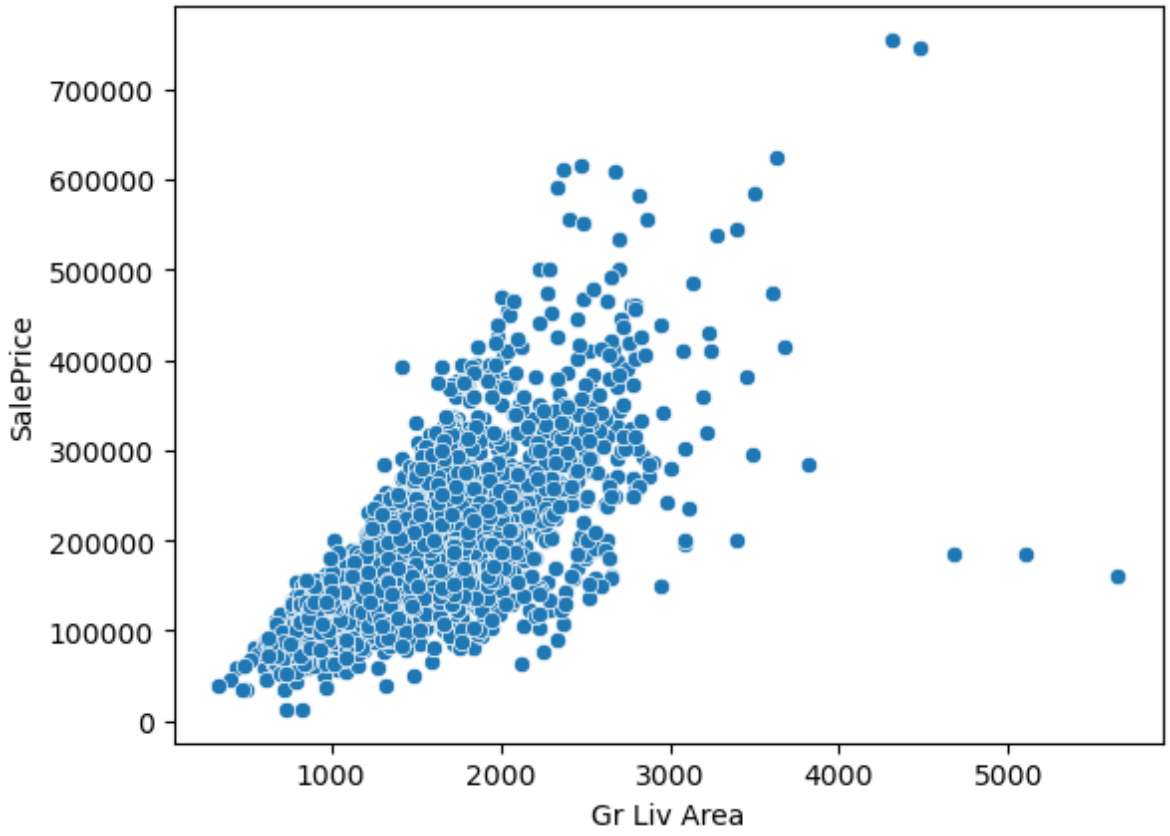
In [40]:

```
1 sns.scatterplot(x='Overall Qual', y='SalePrice', data=df);
```



In [41]:

```
1 sns.scatterplot(x='Gr Liv Area', y='SalePrice', data=df);
```



In [42]:

```
df[(df['Gr Liv Area']>4000) & (df['SalePrice']<200000)]
```

Out[42]:

	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour	Utilities	...
1498	908154235	60	RL	313.0	63887	Pave	NaN	IR3	Bnk	AllPub	...
2180	908154195	20	RL	128.0	39290	Pave	NaN	IR1	Bnk	AllPub	...
2181	908154205	60	RL	130.0	40094	Pave	NaN	IR1	Bnk	AllPub	...

3 rows × 81 columns

In [43]:

```
indices = df[(df['Gr Liv Area']>4000) & (df['SalePrice']<200000)].index
indices
```

Out[43]:

```
Int64Index([1498, 2180, 2181], dtype='int64')
```

In [44]:

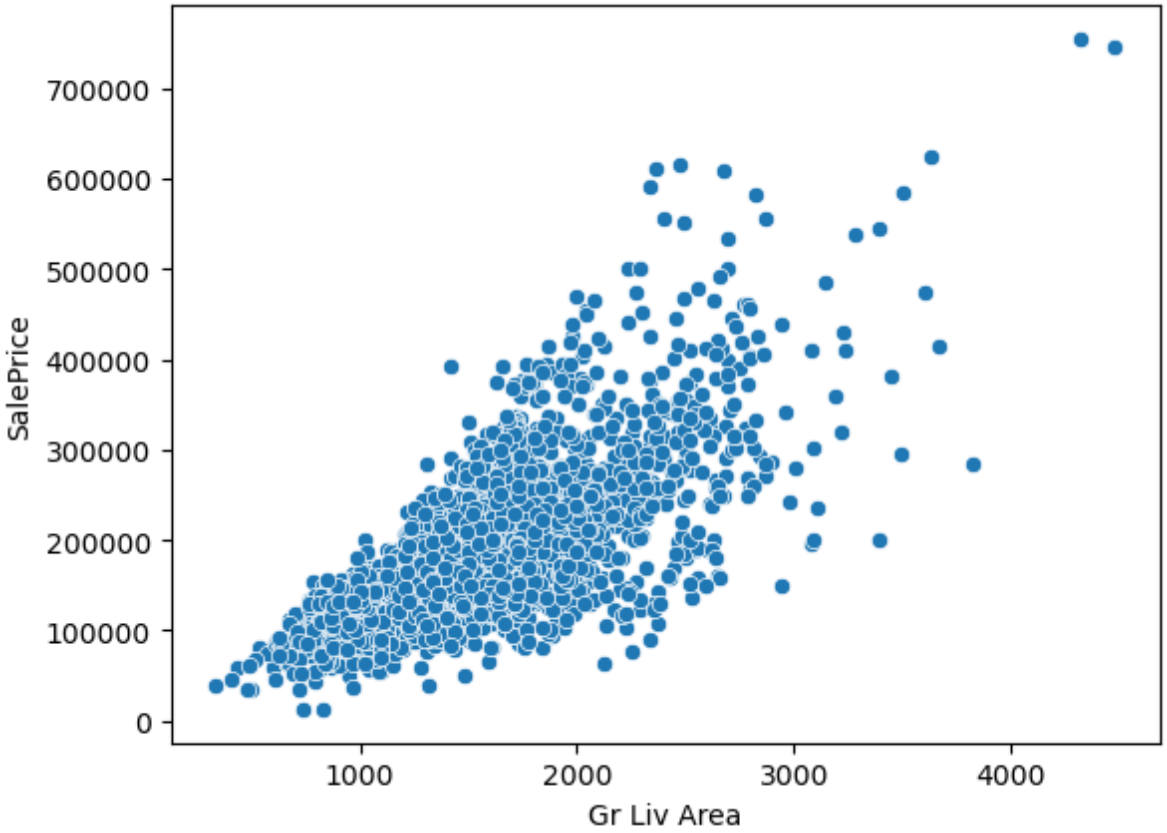
```
new_df = df.drop(indices,axis=0)
new_df.shape
```

Out[44]:

```
(2927, 81)
```

In [45]:

```
1 sns.scatterplot(x='Gr Liv Area', y='SalePrice', data=new_df);
```



In [46]:

```
1 df.to_csv("datasets/Ames_outliers_removed.csv",index=False)
```