In [ ]:
```
# 🔹 PRACTICAL 3 – Outlier Detection & Feature Transformation
```

In [1]:
```
#Name: Ritika Junekar
#Subject: PD
#Roll_No:29
```

In [3]:
```
#Part 1: IQR Outlier detection
```

In [5]:
```
import pandas as pd
import numpy as np
```

In [7]:
```
df = pd.DataFrame({
    'val': np.append(np.random.normal(50, 5, 200), [200, 220, 300])
})
```

In [9]:
```
Q1 = df['val'].quantile(0.25)
Q3 = df['val'].quantile(0.75)
IQR = Q3 - Q1
```

In [11]:
```
outliers = df[(df['val'] < Q1 - 1.5 * IQR) | (df['val'] > Q3 + 1.5 * IQR)]
print("Outliers:\n", outliers)
```

```
Outliers:
             val
22     66.693659
200   200.000000
201   220.000000
202   300.000000
```

In [ ]:
```
#Part 2: Capping + Power Transform + Scaling
```

In [13]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, PowerTransformer
```

In [15]:
```
rng = np.random.RandomState(0)
x = np.concatenate([rng.normal(50, 5, 200), np.array([200, 220, 300])])
df = pd.DataFrame({'value': x})
```

In [17]:
```
display(df.describe())
```

	value
count	203.000000
mean	53.157195
std	24.078891
min	37.235051
25%	46.539706
50%	50.332586
75%	54.252984
max	300.000000

In [19]:
```
# Detect outliers using IQR
Q1 = df['value'].quantile(0.25)
Q3 = df['value'].quantile(0.75)
IQR = Q3 - Q1
```

In [21]:
```
mask_out = (df['value'] < (Q1 - 1.5 * IQR)) | (df['value'] > (Q3 + 1.5 * IQR))
print('Outliers index:', df[mask_out].index.tolist())
```

```
Outliers index: [200, 201, 202]
```

In [23]:
```
# Capping (winsorizing)
lower = Q1 - 1.5 * IQR
upper = Q3 + 1.5 * IQR
df['value_capped'] = df['value'].clip(lower, upper)
```

In [25]:
```
# Power transform (Yeo-Johnson)
pt = PowerTransformer(method='yeo-johnson')
df['value_trans'] = pt.fit_transform(df[['value_capped']])
```

In [27]:
```
# Standard scaling
scaler = StandardScaler()
df['value_scaled'] = scaler.fit_transform(df[['value_trans']])
```

In [29]:
```
display(df.head())
```

	value	value_capped	value_trans	value_scaled
0	58.820262	58.820262	1.462508	1.462508
1	52.000786	52.000786	0.320758	0.320758
2	54.893690	54.893690	0.825138	0.825138
3	61.204466	61.204466	1.825890	1.825890
4	59.337790	59.337790	1.542821	1.542821

In [ ]: