# IT3071 – Machine Learning and Optimization Methods

## Lab Sheet 02

## Part A

01) Create a DataFrame with missing values and perform the following,

a) Identify missing values
b) Fill missing values with specified values
c) Drop rows with any missing values

02) Create a DataFrame with duplicate rows and,

a) Detect duplicate rows
b) Remove them

03) Create a DataFrame with incorrect data types and convert them to appropriate types.

04) Create a DataFrame with inconsistent capitalization and whitespaces.

a) Remove extra spaces
b) Convert all text to lowercase

05) Create a DataFrame with numeric outliers and,

a) Detect values greater than a threshold
b) Replace them with the mean

## Part B

Basic Image Processing

1. Handling Missing Data

In [9]:
```
import pandas as pd
import numpy as np

data1 = {
    'Name': ['Alice', 'Bob', np.nan, 'David'],
    'Age': [25, np.nan, 22, 28],
    'City': ['Colombo', 'Kandy', 'Galle', np.nan]
}
df1 = pd.DataFrame(data1)

df1.isnull()

df1_filled = df1.fillna({'Name': 'Unknown', 'Age': df1['Age'].mean(),
                         'City': 'Unknown'})
print(df1_filled)
print()
df1_dropped = df1.dropna()
print(df1_dropped)
```

```
      Name   Age     City
0    Alice  25.0  Colombo
1      Bob  25.0    Kandy
2  Unknown  22.0    Galle
3    David  28.0  Unknown

    Name   Age     City
0  Alice  25.0  Colombo
```

2. Removing Duplicates

In [5]:
```
data2 = {
    'ID': [1, 2, 3, 3, 4, 5, 5],
    'Name': ['A', 'B', 'C', 'C', 'D', 'E', 'E']
}
df2 = pd.DataFrame(data2)
print(df2)
print()
print(df2[df2.duplicated()])
print()
df2_unique = df2.drop_duplicates()
print(df2_unique)
```

```
   ID Name
0   1    A
1   2    B
2   3    C
3   3    C
4   4    D
5   5    E
6   5    E

   ID Name
3   3    C
6   5    E

   ID Name
0   1    A
1   2    B
2   3    C
4   4    D
5   5    E
```

3. Fixing Data Types

In [6]:
```
data3 = {
    'ID': ['101', '102', '103'],
    'JoinDate': ['2022-01-10', '2022-02-15',
                 '2022-03-01'],
    'Salary': ['50000', '60000', '70000']
}
df3 = pd.DataFrame(data3)
print(df3.dtypes)

print()
df3['ID'] = df3['ID'].astype(int)
df3['JoinDate'] = pd.to_datetime(df3['JoinDate'])
df3['Salary'] = df3['Salary'].astype(float)
print(df3.dtypes)
```

```
ID          object
JoinDate    object
Salary      object
dtype: object

ID                   int32
JoinDate    datetime64[ns]
Salary             float64
dtype: object
```

4. Standardizing Text Data

In [7]:
```
data4 = {
    'Product': [' Apple ', 'BaNaNa', '  Cherry', 'DaTE ']
}
df4 = pd.DataFrame(data4)
print(df4)
print()

df4['Product'] = df4['Product'].str.strip()
print(df4)
```

```
print()

df4['Product'] = df4['Product'].str.lower()
print(df4)
```

```
   Product
0    Apple
1    BaNaNa
2    Cherry
3     DaTE

  Product
0   Apple
1  BaNaNa
2  Cherry
3    DaTE

  Product
0   apple
1  banana
2  cherry
3    date
```

5. Outlier Detection and Handling

In [8]:
```
data5 = {
    'Score': [55, 60, 62, 90, 95, 200, 58, 59, 210]
}
df5 = pd.DataFrame(data5)
print(df5)
print()

outliers = df5[df5['Score'] > 100]
print(outliers)
print()

mean_score = df5[df5['Score'] <= 100]['Score'].mean()
df5['Score'] = df5['Score'].apply(lambda x: mean_score if x > 100 else x)
print(df5)
```

```
   Score
0     55
1     60
2     62
3     90
4     95
5    200
6     58
7     59
8    210

   Score
5    200
8    210

       Score
0  55.000000
1  60.000000
2  62.000000
3  90.000000
4  95.000000
5  68.428571
6  58.000000
7  59.000000
8  68.428571
```

Load the image

In [21]:
```
from PIL import Image
import matplotlib.pyplot as plt

# Open image
image = Image.open('panda.jpg')

# Display image using matplotlib
plt.imshow(image)
plt.axis('off')
plt.show()
```

Convert to Grayscale

In [22]:
```
gray_image = image.convert("L")
plt.imshow(gray_image, cmap='gray')
plt.axis('off')
plt.title("Grayscale Image")
plt.show()
```


Resize the Image

In [23]:
```
resized_image = image.resize((1000, 500))
plt.imshow(resized_image)
plt.axis('off')
plt.title("Resized Image")
plt.show()
```


Rotate the Image

In [24]:
```
rotated_image = image.rotate(45)  # Rotate 45 degrees
plt.imshow(rotated_image)
plt.axis('off')
plt.title("Rotated Image")
plt.show()
```


Flip the Image (Mirror)

In [25]:
```
flipped_image = image.transpose(Image.FLIP_LEFT_RIGHT)
plt.imshow(flipped_image)
plt.axis('off')
plt.title("Flipped Image")
plt.show()
```


Crop a Region

In [26]:
```
# Crop box: (left, upper, right, lower)
cropped_image = image.crop((50, 50, 250, 250))
plt.imshow(cropped_image)
plt.axis('off')
plt.title("Cropped Image")
plt.show()
```


Filtering

Blur Filter

In [27]:
```
from PIL import ImageFilter
```

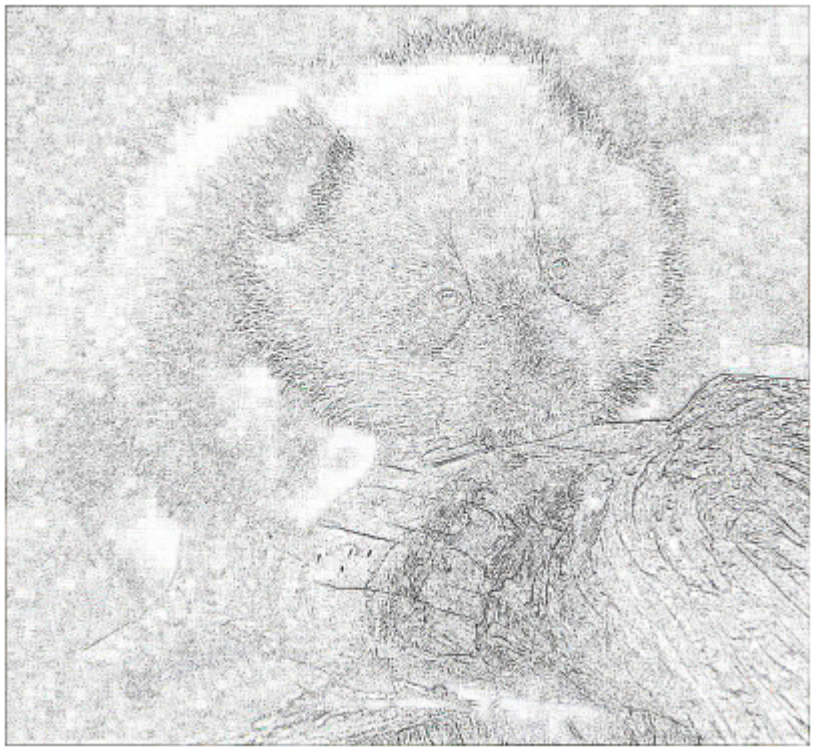
In [28]:
```
blurred = image.filter(ImageFilter.BLUR)
plt.imshow(blurred)
plt.axis('off')
plt.title("Blurred")
plt.show()
```

Blurred

Contour Filter

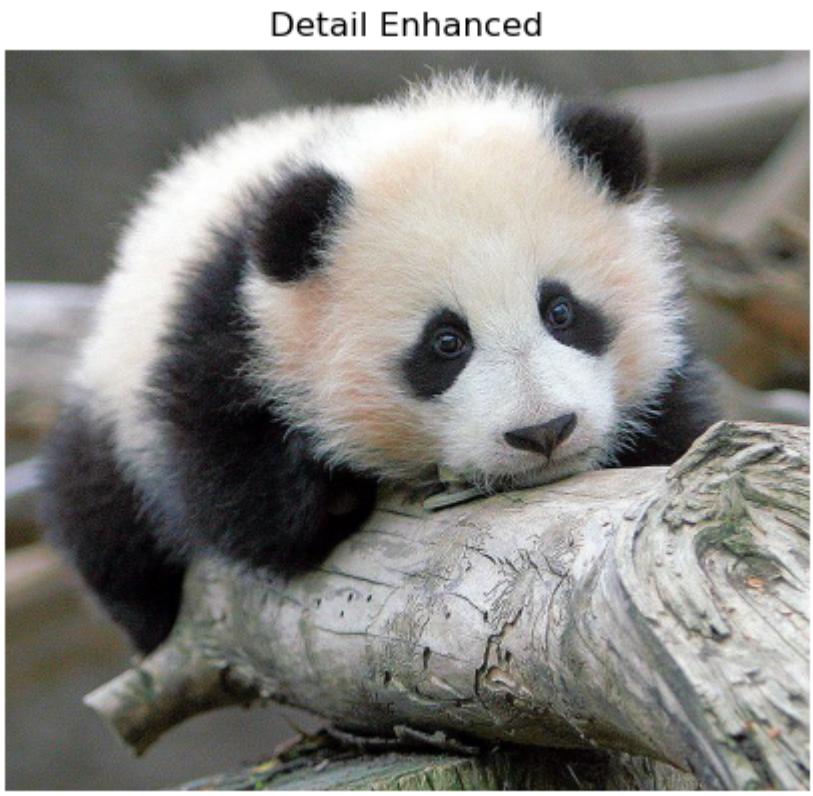
In [29]:
```
contour = image.filter(ImageFilter.CONTOUR)
plt.imshow(contour)
plt.axis('off')
plt.title("Contour")
plt.show()
```

Contour



Detail Enhancement

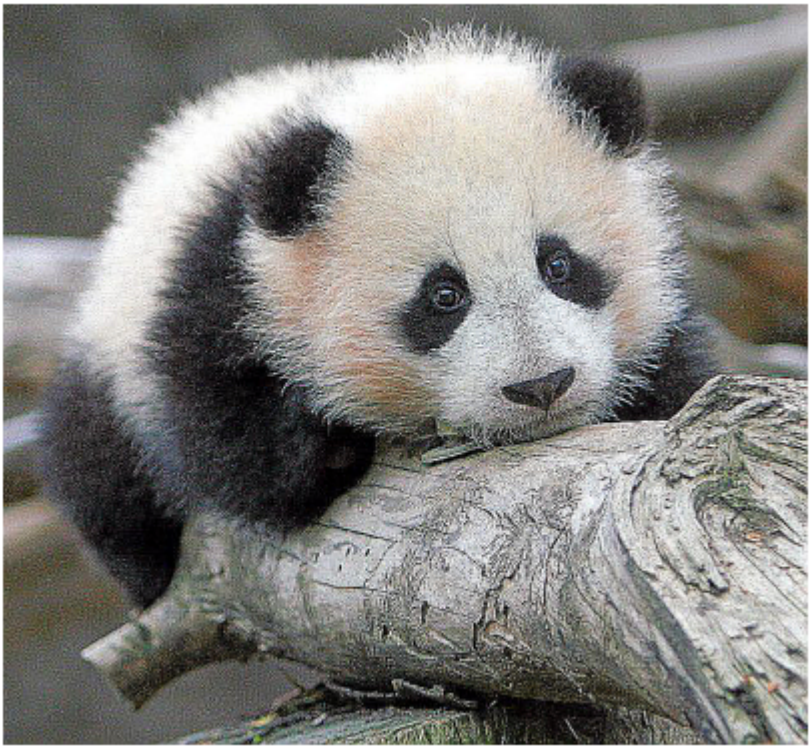
In [30]:
```
detailed = image.filter(ImageFilter.DETAIL)
plt.imshow(detailed)
plt.axis('off')
plt.title("Detail Enhanced")
plt.show()
```




Edge Enhancement

In [31]:
```
edges = image.filter(ImageFilter.EDGE_ENHANCE)
plt.imshow(edges)
plt.axis('off')
plt.title("Edge Enhanced")
plt.show()
```

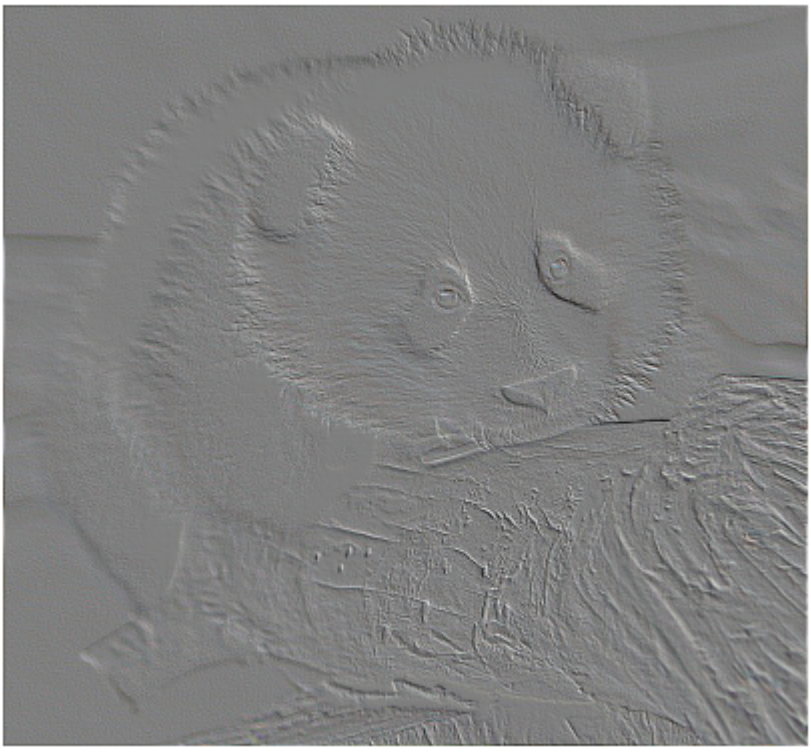
Edge Enhanced



Emboss Filter

In [32]:
```
emboss = image.filter(ImageFilter.EMBOSS)
plt.imshow(emboss)
plt.axis('off')
plt.title("Embossed")
plt.show()
```

Embossed



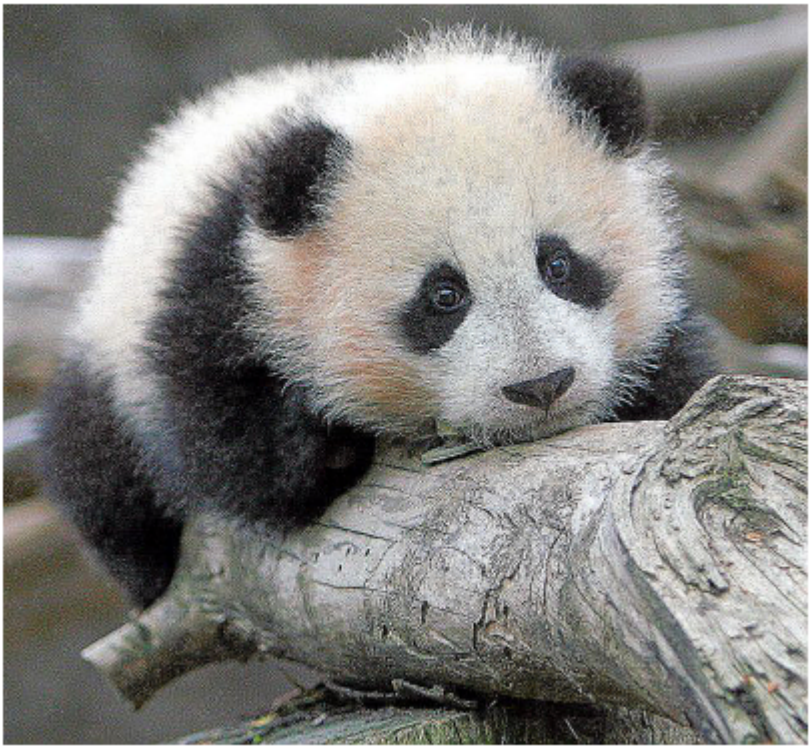
Sharpen Filter

In [33]:
```
sharpened = image.filter(ImageFilter.SHARPEN)
plt.imshow(sharpened)
plt.axis('off')
plt.title("Sharpened")
plt.show()
```


Custom Convolution Filter (Kernel)

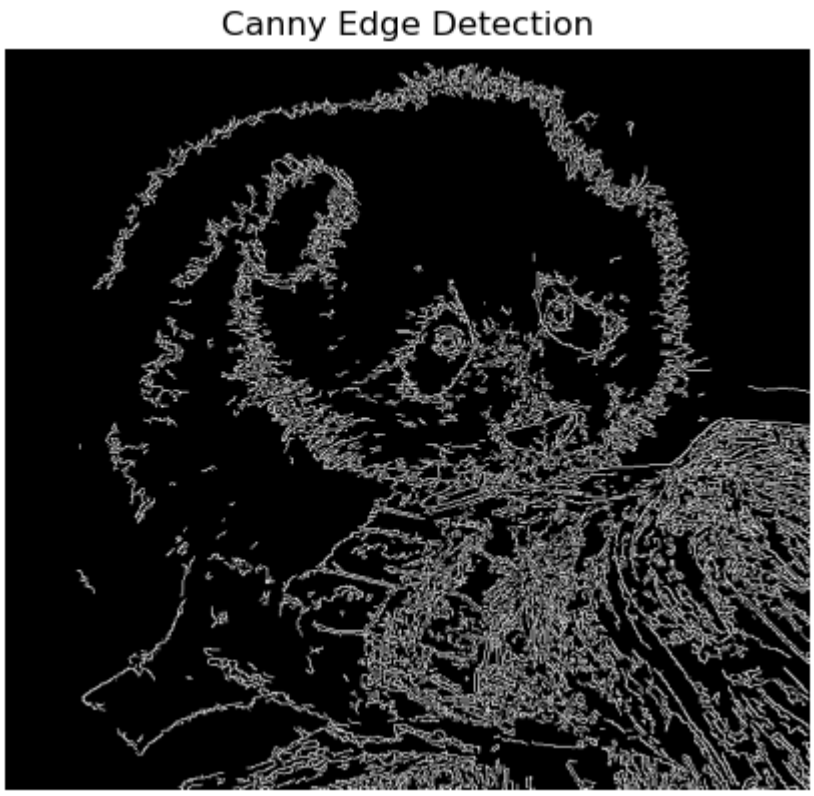
In [34]:
```
# Example 3x3 sharpening kernel
kernel = ImageFilter.Kernel(
    size=(3, 3),
    kernel=[0, -1, 0, -1, 5, -1, 0, -1, 0],
    scale=None,  # Automatically calculated
    offset=0
)
custom_filtered = image.filter(kernel)
plt.imshow(custom_filtered)
plt.axis('off')
plt.title("Custom Filtered")
plt.show()
```

Custom Filtered



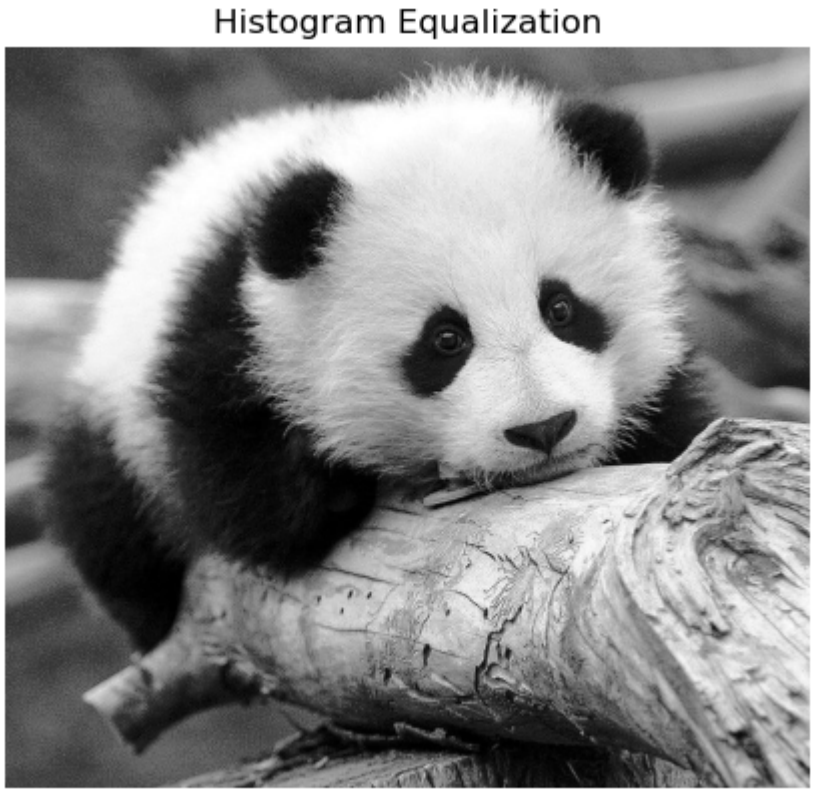
Edge Detection (Canny - OpenCV)

In [37]:
```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('panda.jpg', cv2.IMREAD_GRAYSCALE)
edges = cv2.Canny(img, 100, 200)

plt.imshow(edges, cmap='gray')
plt.title('Canny Edge Detection')
plt.axis('off')
plt.show()
```




Histogram Equalization

In [38]:
```
equalized = cv2.equalizeHist(img)
plt.imshow(equalized, cmap='gray')
plt.title("Histogram Equalization")
plt.axis('off')
plt.show()
```




Thresholding

In [39]:
```
_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
plt.imshow(binary, cmap='gray')
plt.title("Binary Threshold")
plt.axis('off')
plt.show()
```


Color Space Conversion (Ex- RGB to HSV)

In [40]:
```
color_img = cv2.imread('panda.jpg')
hsv = cv2.cvtColor(color_img, cv2.COLOR_BGR2HSV)
plt.imshow(cv2.cvtColor(hsv, cv2.COLOR_HSV2RGB))
plt.title("HSV Converted")
plt.axis('off')
plt.show()
```

HSV Converted

Save the Processed Image

In [35]:
```
gray_image.save("panda_gray.jpg")
```