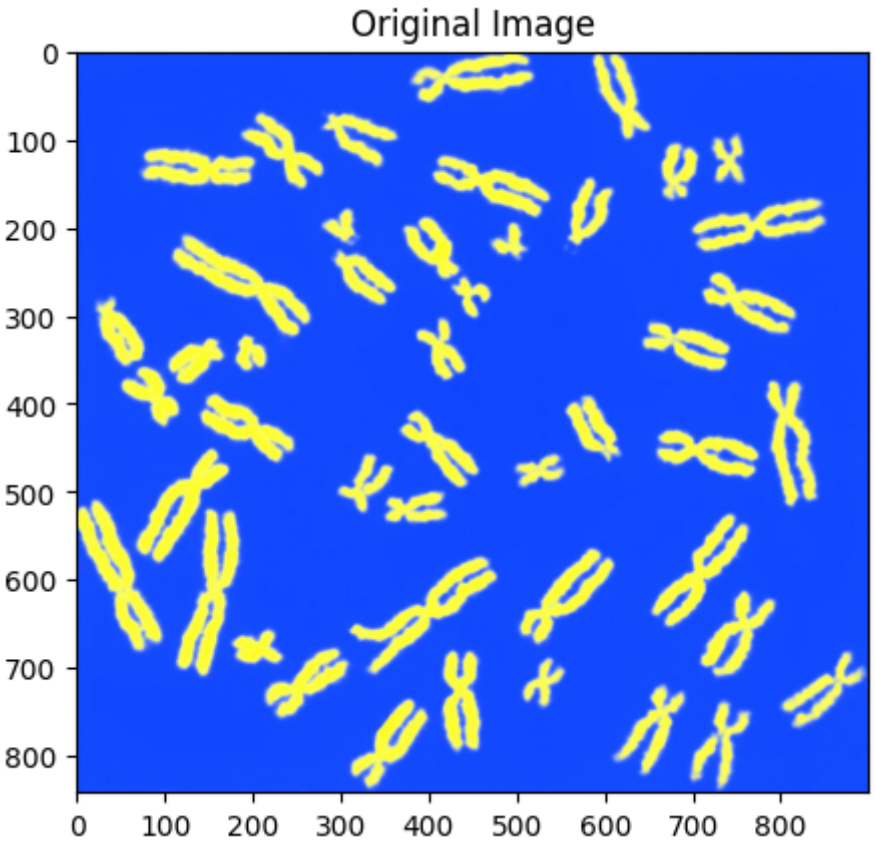
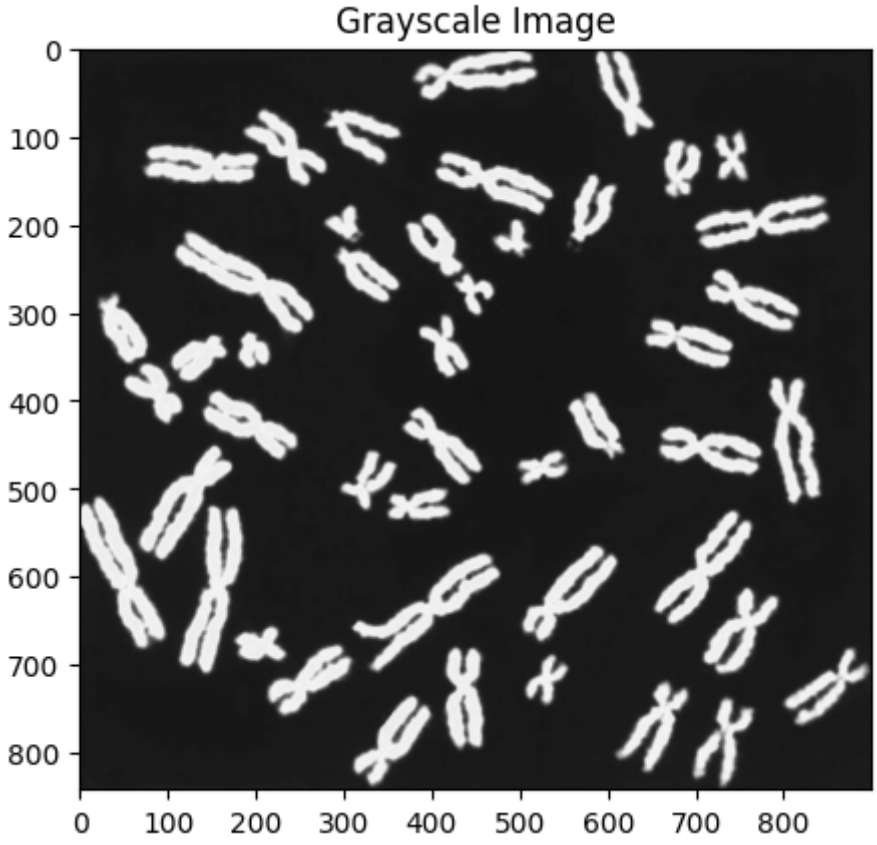
In [4]:
```
# Import necessary libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [6]:

```
image_path = r"C:\Users\navde\Desktop\chromosomes.jpg"  # Replace with the 
image = cv2.imread(image_path)

# Plot the original image
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.show()

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Plot the grayscale image
plt.imshow(gray_image, cmap="gray")
plt.title("Grayscale Image")
plt.show()
```

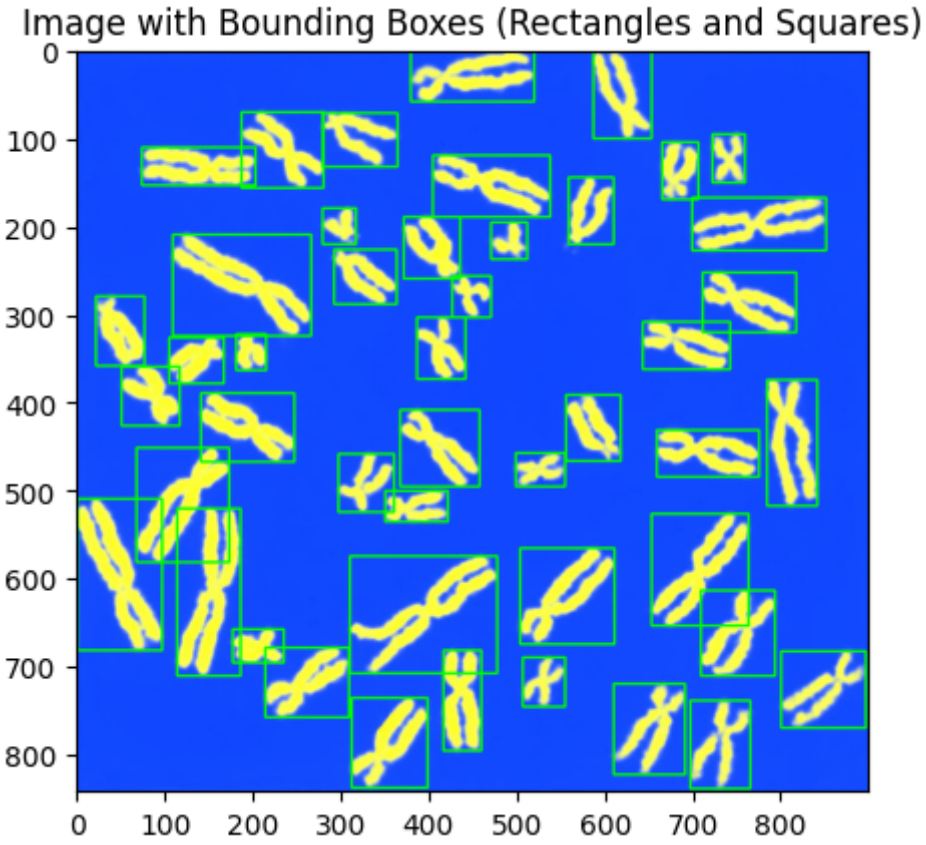



Grayscale Image
0
100
200
300
400
500
600
700
800
0
100
200
300
400
500
600
700
800

In [12]:

```
# Convert the grayscale image to binary using adaptive and global threshold
adaptive_binary = cv2.adaptiveThreshold(gray_image, 255, cv2.ADAPTIVE_THRESH
_, global_binary = cv2.threshold(gray_image, 128, 255, cv2.THRESH_BINARY)
binary_image = cv2.bitwise_and(adaptive_binary, global_binary)

# Apply morphological operations (erosion followed by dilation)
kernel = np.ones((3, 3), np.uint8)
binary_image = cv2.erode(binary_image, kernel, iterations=1)
binary_image = cv2.dilate(binary_image, kernel, iterations=1)

# Find contours in the binary image
contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL, cv2.CHAIN_AP
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Assuming 'gray_image' and 'image' are already defined

# Convert the grayscale image to binary using adaptive and global threshold
adaptive_binary = cv2.adaptiveThreshold(gray_image, 255, cv2.ADAPTIVE_THRESH
_, global_binary = cv2.threshold(gray_image, 128, 255, cv2.THRESH_BINARY)
binary_image = cv2.bitwise_and(adaptive_binary, global_binary)

# Apply morphological operations (erosion followed by dilation)
kernel = np.ones((3, 3), np.uint8)
binary_image = cv2.erode(binary_image, kernel, iterations=1)
binary_image = cv2.dilate(binary_image, kernel, iterations=1)

# Find contours in the binary image
contours, _ = cv2.findContours(binary_image, cv2.RETR_EXTERNAL, cv2.CHAIN_AP

# Create a list to store features and bounding boxes
data = []

# Create an image to draw bounding boxes
image_with_boxes = image.copy()

# Iterate through each contour and calculate features
for i, contour in enumerate(contours):
    area = cv2.contourArea(contour)
    perimeter = cv2.arcLength(contour, True)

    # Calculate circularity
    circularity = (4 * np.pi * area) / (perimeter ** 2)

    # Bounding box
    x, y, w, h = cv2.boundingRect(contour)
    bounding_box_area = w * h

    # Calculate aspect ratio
    aspect_ratio = float(w) / h

    # Determine the shape based on aspect ratio
    if aspect_ratio > 1.2:
        shape_label = "Rectangle"
    elif aspect_ratio < 0.8:
        shape_label = "Rectangle"  # Invert width and height for landscape r
    else:
        shape_label = "Square"
```

```
    # Append features to the list
    data.append({"Chromosome": f"Chromosome_{i + 1}", "Area": area, "Perime
                 "Bounding_Box_Area": bounding_box_area, "Circularity": cir
                 "X": x, "Y": y, "Width": w, "Height": h, "Shape": shape_la

    # Draw bounding box on the image
    cv2.rectangle(image_with_boxes, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Display the image with bounding boxes
plt.imshow(cv2.cvtColor(image_with_boxes, cv2.COLOR_BGR2RGB))
plt.title("Image with Bounding Boxes (Rectangles and Squares)")
plt.show()
```



In [13]:
```
chromosomes_data=pd.DataFrame(data)
chromosomes_data
```

Out[13]:

	Chromosome	Area	Perimeter	Bounding_Box_Area	Circularity	X	Y	Width	H
0	Chromosome_1	2113.5	460.090400	6800	0.125466	696	739	68	
1	Chromosome_2	2950.0	482.842707	8772	0.159009	312	736	86	
2	Chromosome_3	2435.5	499.144222	8343	0.122842	609	720	81	
3	Chromosome_4	1004.5	240.551296	2688	0.218145	506	690	48	
4	Chromosome_5	2172.0	497.386863	8352	0.110327	799	683	96	
5	Chromosome_6	3101.0	497.847760	4902	0.157224	416	682	43	
6	Chromosome_7	2712.0	455.244727	7505	0.164441	214	679	95	
7	Chromosome_8	1319.0	196.651801	2204	0.428607	177	658	58	
8	Chromosome_9	2795.5	495.972650	8148	0.142808	708	614	84	
9	Chromosome_10	4857.5	815.619398	22211	0.091759	310	575	167	
10	Chromosome_11	3460.5	571.511754	11663	0.133137	503	566	107	
11	Chromosome_12	3813.5	631.796023	13970	0.120055	652	527	110	
12	Chromosome_13	5286.5	820.742202	13680	0.098620	114	521	72	
13	Chromosome_14	5453.5	689.854899	16512	0.144002	1	510	96	
14	Chromosome_15	1434.5	313.379724	2485	0.183556	350	501	71	
15	Chromosome_16	1416.0	307.078207	4158	0.188701	297	459	63	
16	Chromosome_17	989.5	227.722869	2128	0.239780	498	458	56	
17	Chromosome_18	4210.5	579.854899	13650	0.157364	68	452	105	
18	Chromosome_19	2923.5	509.889390	6148	0.141306	658	432	116	
19	Chromosome_20	2425.5	419.830514	7830	0.172927	367	409	90	
20	Chromosome_21	1929.0	377.646750	4650	0.169969	555	392	62	
21	Chromosome_22	3259.5	396.232535	8268	0.260892	141	390	106	
22	Chromosome_23	3486.0	639.730009	8151	0.107039	783	375	57	
23	Chromosome_24	1915.0	266.107645	4422	0.339832	51	360	66	
24	Chromosome_25	1771.5	207.237588	3224	0.518339	105	327	62	
25	Chromosome_26	843.5	163.296463	1394	0.397504	181	323	34	
26	Chromosome_27	2309.5	464.918827	5346	0.134268	642	309	99	
27	Chromosome_28	1512.5	291.865005	3850	0.223121	386	304	55	
28	Chromosome_29	2191.5	236.208150	4345	0.493585	22	280	55	
29	Chromosome_30	838.0	194.509666	2068	0.278337	426	257	44	
30	Chromosome_31	2624.5	482.173661	7208	0.141857	710	253	106	
31	Chromosome_32	1876.0	331.019332	4402	0.215147	292	227	71	
32	Chromosome_33	5018.0	659.837653	18055	0.144833	109	210	157	
33	Chromosome_34	720.0	162.710676	1722	0.341751	470	196	41	
34	Chromosome_35	1952.5	303.119838	4480	0.267037	371	190	64	
35	Chromosome_36	731.0	163.539104	1558	0.343466	279	180	38	
36	Chromosome_37	3483.0	660.558436	9120	0.100309	698	168	152	
37	Chromosome_38	1736.0	328.391916	3876	0.202290	558	145	51	
38	Chromosome_39	3345.0	575.872145	9310	0.126752	404	120	133	

	Chromosome	Area	Perimeter	Bounding_Box_Area	Circularity	X	Y	Width	H
39	Chromosome_40	3303.0	553.847760	5547	0.135312	74	111	129	
40	Chromosome_41	1455.0	310.107645	2665	0.190129	664	105	41	
41	Chromosome_42	1031.0	255.480228	2035	0.198497	721	96	37	
42	Chromosome_43	1939.0	365.705624	5185	0.182190	279	72	85	
43	Chromosome_44	2549.5	426.232535	7998	0.176348	187	71	93	
44	Chromosome_45	2339.5	432.717817	6534	0.157009	586	2	66	
45	Chromosome_46	3348.0	618.759447	7980	0.109888	379	2	140	

In [14]:

```
# Select numeric columns for normalization
numeric_columns = df_features_result.select_dtypes(include=np.number).columı

# Manual z-scoring (standardization)
for col in numeric_columns:
    mean = df_features_result[col].mean()
    std_dev = df_features_result[col].std()
    df_features_result[col] = (df_features_result[col] - mean) / std_dev


# Detect outliers based on z-scores
z_threshold = 3.2 # Adjust this threshold based on your requirements

# Calculate z-scores for each numeric column
z_scores = (df_features_result[numeric_columns] - df_features_result[numeri

# Identify rows with at least one z-score beyond the threshold
outliers = (np.abs(z_scores) > z_threshold).any(axis=1)

# Display the rows with outliers
print("\nRows with Outliers:")
print(df_features_result[outliers])
```

```
Rows with Outliers:
      Chromosome      Area  Perimeter  Bounding_Box_Area  Circularity  \
9  Chromosome_10  1.944532   2.244969           3.329976    -1.063462

          X         Y    Width    Height      Shape
9 -0.382124  0.936989  2.51535  1.494362  Rectangle
```

In [ ]: