In [13]:

```python
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt

def calculate_band_statistics(data, band_names=None, no_data_value=0):

    if band_names is None:
        band_names = [
            'B1_Coastal', 'B2_Blue', 'B3_Green', 'B4_Red',
            'B5_RedEdge1', 'B6_RedEdge2', 'B7_RedEdge3', 'B8_NIR',
            'B8A_RedEdge4', 'B9_WaterVapor', 'B11_SWIR1', 'B12_SWIR2'
        ]

    num_bands = data.shape[2] if len(data.shape) == 3 else len(band_names)

    # Initialize results dictionary
    results = {
        'Band': [],
        'Mean': [],
        'Std_Dev': [],
        'Minimum': [],
        'Maximum': [],
        'Q1': [],
        'Median': [],
        'Q3': [],
        'Skewness': [],
        'Valid_Pixels': [],
        'No_Data_Pixels': [],
        'Valid_Percentage': []
    }

    for i in range(num_bands):
        # Extract band data
        if len(data.shape) == 3:
            band = data[:, :, i].flatten()
        else:
            band = data[i].flatten() if hasattr(data[i], 'flatten') else np.array(d

        # Create mask for valid data
        if no_data_value is not None:
            valid_mask = (band != no_data_value) & (~np.isnan(band)) & (band >= 0)
        else:
            valid_mask = ~np.isnan(band)

        valid_data = band[valid_mask]

        # Store band name
        results['Band'].append(band_names[i] if i < len(band_names) else f'Band_{i+

        if len(valid_data) > 0:
            # Basic statistics
            results['Mean'].append(np.mean(valid_data))
            results['Std_Dev'].append(np.std(valid_data, ddof=1))
```

```
            results['Minimum'].append(np.min(valid_data))
            results['Maximum'].append(np.max(valid_data))


            quartiles = np.percentile(valid_data, [25, 50, 75])
            results['Q1'].append(quartiles[0])
            results['Median'].append(quartiles[1])
            results['Q3'].append(quartiles[2])


            results['Skewness'].append(stats.skew(valid_data))



            results['Valid_Pixels'].append(len(valid_data))
            results['No_Data_Pixels'].append(len(band) - len(valid_data))
            results['Valid_Percentage'].append(len(valid_data) / len(band) * 100)

        else:

            for key in ['Mean', 'Std_Dev', 'Minimum', 'Maximum', 'Q1', 'Median', 'Q
                results[key].append(np.nan)
            results['Valid_Pixels'].append(0)
            results['No_Data_Pixels'].append(len(band))
            results['Valid_Percentage'].append(0.0)

    # Create DataFrame
    stats_df = pd.DataFrame(results)

    return stats_df



def display_statistics_table(stats_df, precision=4):

    print("\n" + "=" * 120)
    print("SENTINEL-2 BAND STATISTICS - ROCHESTER SUMMER DATA")
    print("=" * 120)

    # Format numeric columns
    numeric_cols = ['Mean', 'Std_Dev', 'Minimum', 'Maximum', 'Q1', 'Median', 'Q3',
    display_df = stats_df.copy()

    for col in numeric_cols:
        display_df[col] = display_df[col].round(precision)

    display_df['Valid_Percentage'] = display_df['Valid_Percentage'].round(1)


    print(display_df.to_string(index=False,
                               col_space=10,
                               float_format=f'{{:.{precision}f}}'.format))

    print("\n" + "=" * 120)

def analyze_spectral_characteristics(stats_df):
```

```python
    print("\n" + "=" * 80)
    print("SPECTRAL ANALYSIS INSIGHTS")
    print("=" * 80)

    # Reflectance patterns
    print("\n1. REFLECTANCE PATTERNS:")
    vis_bands = ['B2_Blue', 'B3_Green', 'B4_Red']
    nir_bands = ['B8_NIR']
    swir_bands = ['B11_SWIR1', 'B12_SWIR2']

    vis_mean = stats_df[stats_df['Band'].isin(vis_bands)]['Mean'].mean()
    nir_mean = stats_df[stats_df['Band'].isin(nir_bands)]['Mean'].iloc[0]
    swir_mean = stats_df[stats_df['Band'].isin(swir_bands)]['Mean'].mean()

    print(f"   - Visible (RGB) average: {vis_mean:.3f}")
    print(f"   - NIR average: {nir_mean:.3f}")
    print(f"   - SWIR average: {swir_mean:.3f}")

    if nir_mean > vis_mean * 1.5:
        print("   → Strong vegetation signature detected (high NIR/Red ratio)")
    else:
        print("   → Mixed urban/vegetation landscape")

    # Variability analysis
    print("\n2. LANDSCAPE HETEROGENEITY:")
    high_var_bands = stats_df[stats_df['Std_Dev'] > stats_df['Std_Dev'].median()]['
    print(f"   - High variability bands: {', '.join(high_var_bands)}")
    print("   → Indicates diverse land cover types")

    # Skewness interpretation
    print("\n3. DISTRIBUTION CHARACTERISTICS:")
    highly_skewed = stats_df[stats_df['Skewness'] > 1.0]
    if not highly_skewed.empty:
        print("   - Highly positively skewed bands:")
        for _, row in highly_skewed.iterrows():
            print(f"     * {row['Band']}: {row['Skewness']:.2f} (few very bright pi

    # Data quality
    print("\n4. DATA QUALITY:")
    avg_valid = stats_df['Valid_Percentage'].mean()
    min_valid = stats_df['Valid_Percentage'].min()
    print(f"   - Average valid pixel percentage: {avg_valid:.1f}%")
    print(f"   - Minimum valid pixel percentage: {min_valid:.1f}%")

    if min_valid < 95:
        low_quality_bands = stats_df[stats_df['Valid_Percentage'] < 95]['Band'].tol
        print(f"   - Bands with potential quality issues: {', '.join(low_quality_ba


if __name__ == "__main__":

    print("Loading Sentinel-2 data...")
    data = np.load('sentinel2_rochester.npy')
```

```python
print("Calculating comprehensive band statistics...")

# Calculate statistics
stats = calculate_band_statistics(data, no_data_value=0)

# Display the statistics table
display_statistics_table(stats)

# Analyze spectral characteristics
analyze_spectral_characteristics(stats)

print(f"\nStatistics calculation complete!")
print(f"DataFrame shape: {stats.shape}")
print(f"Available columns: {list(stats.columns)}")
```

```
Loading Sentinel-2 data...
Calculating comprehensive band statistics...

====================================================================================
====================================
SENTINEL-2 BAND STATISTICS - ROCHESTER SUMMER DATA
====================================================================================
====================================
         Band       Mean    Std_Dev    Minimum    Maximum        Q1     Median
Q3   Skewness  Valid_Pixels  No_Data_Pixels  Valid_Percentage
   B1_Coastal     0.0887     0.0279     0.0333     0.6021     0.0709     0.0829
0.1007     2.9205        630024           53040          92.2000
      B2_Blue     0.0925     0.0350     0.0386     0.7542     0.0716     0.0853
0.1053     4.4789        630024           53040          92.2000
     B3_Green     0.1055     0.0344     0.0430     0.7484     0.0858     0.0987
0.1168     4.6922        630024           53040          92.2000
       B4_Red     0.0943     0.0445     0.0326     0.7728     0.0663     0.0850
0.1093     3.2314        630024           53040          92.2000
  B5_RedEdge1     0.1367     0.0409     0.0346     0.8159     0.1152     0.1310
0.1506     3.1080        630024           53040          92.2000
  B6_RedEdge2     0.2436     0.0610     0.0176     0.7830     0.2131     0.2472
0.2830    -0.7085        630024           53040          92.2000
  B7_RedEdge3     0.2858     0.0773     0.0161     0.7859     0.2444     0.2903
0.3377    -0.7101        630024           53040          92.2000
       B8_NIR     0.2914     0.0803     0.0188     0.9030     0.2475     0.2976
0.3464    -0.7161        630024           53040          92.2000
 B8A_RedEdge4     0.3035     0.0822     0.0157     0.7689     0.2605     0.3098
0.3595    -0.8215        630024           53040          92.2000
B9_WaterVapor     0.3451     0.0779     0.0653     0.6934     0.3052     0.3446
0.3882    -0.3957        630024           53040          92.2000
    B11_SWIR1     0.1917     0.0484     0.0218     0.8196     0.1710     0.1889
0.2086     0.5219        630024           53040          92.2000
    B12_SWIR2     0.1292     0.0486     0.0205     0.9295     0.1020     0.1204
0.1427     1.9499        630024           53040          92.2000

====================================================================================
====================================

================================================================================
SPECTRAL ANALYSIS INSIGHTS
================================================================================

1. REFLECTANCE PATTERNS:
   - Visible (RGB) average: 0.097
   - NIR average: 0.291
   - SWIR average: 0.160
   → Strong vegetation signature detected (high NIR/Red ratio)

2. LANDSCAPE HETEROGENEITY:
   - High variability bands: B6_RedEdge2, B7_RedEdge3, B8_NIR, B8A_RedEdge4, B9_Wate
rVapor, B12_SWIR2
   → Indicates diverse land cover types

3. DISTRIBUTION CHARACTERISTICS:
   - Highly positively skewed bands:
     * B1_Coastal: 2.92 (few very bright pixels)
```

```
    * B2_Blue: 4.48 (few very bright pixels)
    * B3_Green: 4.69 (few very bright pixels)
    * B4_Red: 3.23 (few very bright pixels)
    * B5_RedEdge1: 3.11 (few very bright pixels)
    * B12_SWIR2: 1.95 (few very bright pixels)

4. DATA QUALITY:
   - Average valid pixel percentage: 92.2%
   - Minimum valid pixel percentage: 92.2%
   - Bands with potential quality issues: B1_Coastal, B2_Blue, B3_Green, B4_Red, B5_
RedEdge1, B6_RedEdge2, B7_RedEdge3, B8_NIR, B8A_RedEdge4, B9_WaterVapor, B11_SWIR1,
B12_SWIR2

Statistics calculation complete!
DataFrame shape: (12, 12)
Available columns: ['Band', 'Mean', 'Std_Dev', 'Minimum', 'Maximum', 'Q1', 'Median',
'Q3', 'Skewness', 'Valid_Pixels', 'No_Data_Pixels', 'Valid_Percentage']
```

In [ ]:

```
###part 2
```

In [14]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd
from matplotlib.patches import Rectangle

def standardize(data, no_data_value=0, method='z_score'):

    # Initialize output array
    data_standardized = np.full_like(data, np.nan, dtype=np.float64)
    stats_info = {'band_means': [], 'band_stds': [], 'valid_pixels': []}

    print("Standardizing Sentinel-2 bands...")
    print("=" * 50)

    for band_idx in range(data.shape[2]):
        # Extract band
        band = data[:, :, band_idx].astype(np.float64)

        # Create mask for valid data
        if no_data_value is not None:
            valid_mask = (band != no_data_value) & (~np.isnan(band)) & (band >= 0)
        else:
            valid_mask = ~np.isnan(band)

        valid_data = band[valid_mask]

        if len(valid_data) > 1:  # Need at least 2 points for std calculation
            if method == 'z_score':
                # Standard z-score: (x - mean) / std
                band_mean = np.mean(valid_data)
                band_std = np.std(valid_data, ddof=1)

            elif method == 'robust_z_score':

                band_mean = np.median(valid_data)
```

```python
                mad = np.median(np.abs(valid_data - band_mean))
                band_std = mad * 1.4826  # Convert MAD to std equivalent

            # Avoid division by zero
            if band_std > 0:

                standardized_band = np.full_like(band, np.nan)
                standardized_band[valid_mask] = (band[valid_mask] - band_mean) / ba
                data_standardized[:, :, band_idx] = standardized_band

                # Store statistics
                stats_info['band_means'].append(band_mean)
                stats_info['band_stds'].append(band_std)
                stats_info['valid_pixels'].append(len(valid_data))

                print(f"Band {band_idx+1:2d}: Mean={band_mean:.4f}, Std={band_std:.
            else:
                print(f"Band {band_idx+1:2d}: Zero variance - cannot standardize")
                stats_info['band_means'].append(band_mean)
                stats_info['band_stds'].append(0)
                stats_info['valid_pixels'].append(len(valid_data))
        else:
            print(f"Band {band_idx+1:2d}: Insufficient valid data")
            stats_info['band_means'].append(np.nan)
            stats_info['band_stds'].append(np.nan)
            stats_info['valid_pixels'].append(0)

    print("=" * 50)
    print("Standardization complete!")

    return data_standardized, stats_info

def identify_outliers(standardized_data, threshold=3.0):


    outlier_masks = np.zeros_like(standardized_data, dtype=bool)
    outlier_stats = {'band_outlier_counts': [], 'band_outlier_percentages': []}

    for band_idx in range(standardized_data.shape[2]):
        band_z = standardized_data[:, :, band_idx]
        valid_mask = ~np.isnan(band_z)

        # Identify outliers
        outliers = (np.abs(band_z) > threshold) & valid_mask
        outlier_masks[:, :, band_idx] = outliers

        # Calculate statistics
        outlier_count = np.sum(outliers)
        total_valid = np.sum(valid_mask)
        outlier_percentage = (outlier_count / total_valid * 100) if total_valid > 0

        outlier_stats['band_outlier_counts'].append(outlier_count)
        outlier_stats['band_outlier_percentages'].append(outlier_percentage)

    return outlier_masks, outlier_stats
```

```python
def plot_histograms_with_outliers(original_data, standardized_data, band_names=None
                                  outlier_threshold=3.0, figsize=(20, 16), no_data_v


    if band_names is None:
        band_names = [
            'B1 Coastal', 'B2 Blue', 'B3 Green', 'B4 Red',
            'B5 RedEdge1', 'B6 RedEdge2', 'B7 RedEdge3', 'B8 NIR',
            'B8A RedEdge4', 'B9 WaterVapor', 'B11 SWIR1', 'B12 SWIR2'
        ]

    # Identify outliers
    outlier_masks, outlier_stats = identify_outliers(standardized_data, outlier_thr

    # Create subplots
    fig, axes = plt.subplots(4, 6, figsize=figsize)
    axes = axes.flatten()

    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b',
              '#e377c2', '#7f7f7f', '#bcbd22', '#17becf', '#ff9896', '#c5b0d5']

    for band_idx in range(12):
        # Extract original data
        original_band = original_data[:, :, band_idx].flatten()
        standardized_band = standardized_data[:, :, band_idx].flatten()

        # Create valid data mask
        if no_data_value is not None:
            valid_mask = (original_band != no_data_value) & (~np.isnan(original_ban
        else:
            valid_mask = ~np.isnan(original_band)

        valid_original = original_band[valid_mask]
        valid_standardized = standardized_band[valid_mask & ~np.isnan(standardized_

        # Plot original data histogram
        ax1 = axes[band_idx*2] if band_idx*2 < len(axes) else plt.subplot(4, 6, ban

        if len(valid_original) > 0:
            # Main histogram
            n, bins, patches = ax1.hist(valid_original, bins=50, alpha=0.7,
                                        color=colors[band_idx % len(colors)],
                                        density=True, edgecolor='black', linewidth=0

            # Mark outliers in original data
            outlier_mask_1d = outlier_masks[:, :, band_idx].flatten()[valid_mask]
            if np.any(outlier_mask_1d):
                outlier_values = valid_original[outlier_mask_1d[:len(valid_original
                if len(outlier_values) > 0:
                    ax1.scatter(outlier_values, np.zeros_like(outlier_values),
                                color='red', s=20, alpha=0.6, marker='^',
                                label=f'Outliers ({len(outlier_values)})')

            ax1.set_title(f'{band_names[band_idx]}\nOriginal Data', fontweight='bol
            ax1.set_xlabel('Surface Reflectance', fontsize=8)
            ax1.set_ylabel('Density', fontsize=8)
```

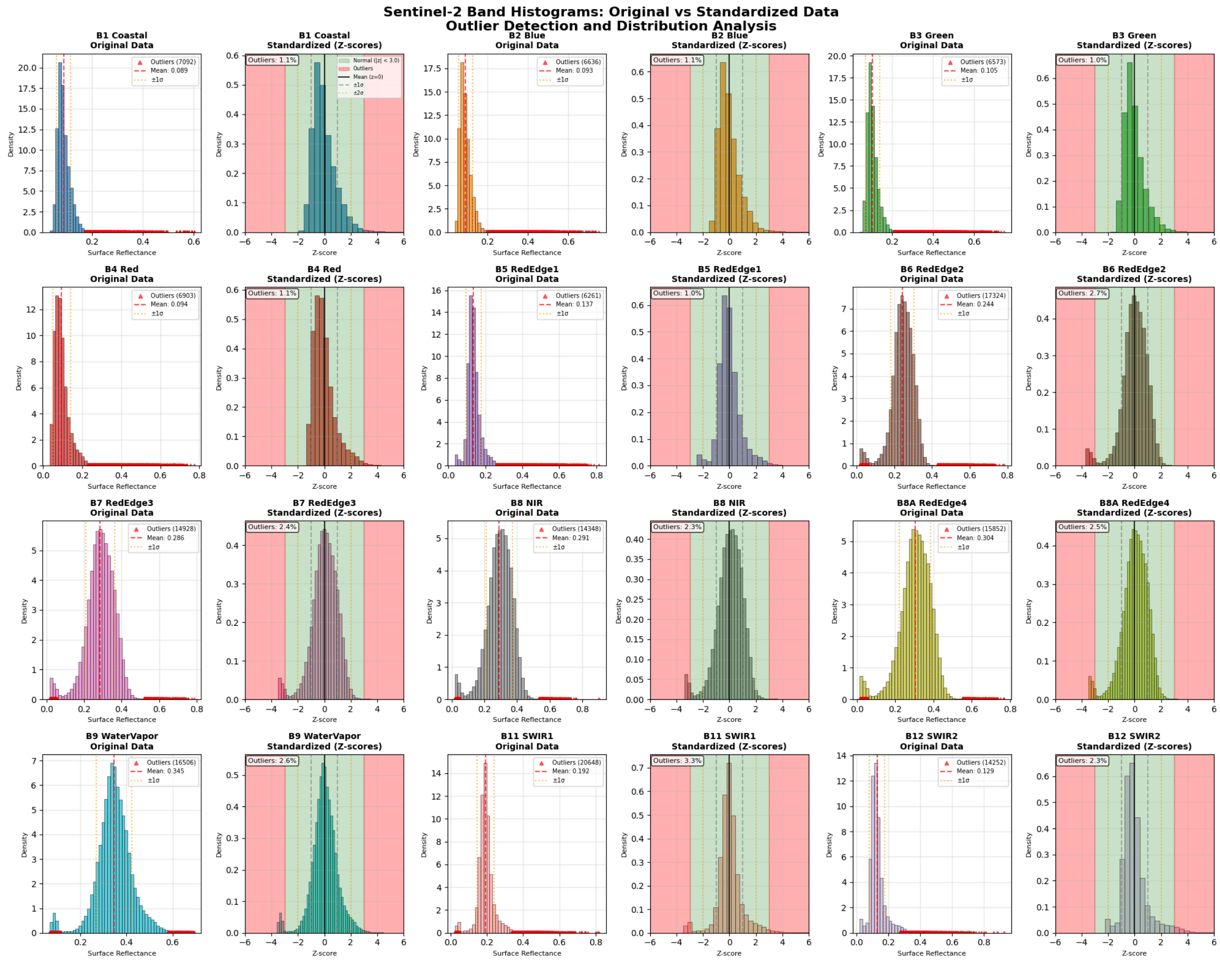
```
        ax1.grid(True, alpha=0.3)

        # Add statistics text
        mean_val = np.mean(valid_original)
        std_val = np.std(valid_original)
        ax1.axvline(mean_val, color='red', linestyle='--', alpha=0.7, label=f'M
        ax1.axvline(mean_val + std_val, color='orange', linestyle=':', alpha=0.
        ax1.axvline(mean_val - std_val, color='orange', linestyle=':', alpha=0.

        if np.any(outlier_mask_1d):
            ax1.legend(fontsize=7)

    # Plot standardized data histogram
    if band_idx*2+1 < len(axes):
        ax2 = axes[band_idx*2+1]
    else:
        continue

    if len(valid_standardized) > 0:
        # Main histogram
        n, bins, patches = ax2.hist(valid_standardized, bins=50, alpha=0.7,
                                    color=colors[band_idx % len(colors)],
                                    density=True, edgecolor='black', linewidth=0

        # Highlight outlier regions
        ax2.axvspan(-outlier_threshold, outlier_threshold, alpha=0.2, color='gr
                   label=f'Normal (|z| < {outlier_threshold})')
        ax2.axvspan(-10, -outlier_threshold, alpha=0.3, color='red', label='Out
        ax2.axvspan(outlier_threshold, 10, alpha=0.3, color='red')

        # Standard normal reference lines
        ax2.axvline(0, color='black', linestyle='-', alpha=0.8, label='Mean (z=
        ax2.axvline(-1, color='gray', linestyle='--', alpha=0.6, label='±1σ')
        ax2.axvline(1, color='gray', linestyle='--', alpha=0.6)
        ax2.axvline(-2, color='orange', linestyle=':', alpha=0.6, label='±2σ')
        ax2.axvline(2, color='orange', linestyle=':', alpha=0.6)

        ax2.set_title(f'{band_names[band_idx]}\nStandardized (Z-scores)', fontw
        ax2.set_xlabel('Z-score', fontsize=8)
        ax2.set_ylabel('Density', fontsize=8)
        ax2.set_xlim(-6, 6)
        ax2.grid(True, alpha=0.3)

        # Add outlier percentage
        outlier_pct = outlier_stats['band_outlier_percentages'][band_idx]
        ax2.text(0.02, 0.98, f'Outliers: {outlier_pct:.1f}%',
                transform=ax2.transAxes, verticalalignment='top',
                bbox=dict(boxstyle='round', facecolor='white', alpha=0.8),
                fontsize=8)

        if band_idx == 0:  # Add legend only to first subplot
            ax2.legend(fontsize=6, loc='upper right')


for idx in range(24, len(axes)):
    fig.delaxes(axes[idx])
```

```python
    plt.suptitle('Sentinel-2 Band Histograms: Original vs Standardized Data\nOutlie
                 fontsize=16, fontweight='bold', y=0.98)
    plt.tight_layout()
    return fig, outlier_stats

def explain_standardization():

    print("\n" + "=" * 80)
    print("WHAT STANDARDIZATION DOES TO YOUR DATA")
    print("=" * 80)

    explanations = {
        "Z-Score Formula": "z = (x - μ) / σ",
        "Purpose": [
            "Removes the units of measurement",
            "Centers data around zero (mean = 0)",
            "Scales data to unit variance (std = 1)",
            "Makes bands comparable despite different reflectance ranges"
        ],
        "Benefits": [
            "Enables comparison across spectral bands",
            "Highlights unusual pixels (outliers)",
            "Normalizes for machine learning algorithms",
            "Reveals distribution shapes more clearly"
        ],
        "Interpretation": [
            "z = 0: Average reflectance for that band",
            "z = +1: One standard deviation above average (brighter)",
            "z = -1: One standard deviation below average (darker)",
            "z > +3: Extreme outliers (very bright pixels - clouds, snow, metal)",
            "z < -3: Extreme outliers (very dark pixels - deep water, shadows)"
        ],
        "Remote Sensing Applications": [
            "Outlier detection: Clouds, shadows, water bodies",
            "Anomaly detection: Urban heat islands, pollution",
            "Change detection: Compare images from different dates",
            "Classification: Normalize features for ML algorithms"
        ]
    }

    for category, content in explanations.items():
        print(f"\n{category}:")
        print("-" * len(category))
        if isinstance(content, list):
            for item in content:
                print(f"  • {item}")
        else:
            print(f"  {content}")

def summarize_outlier_analysis(outlier_stats, band_names=None):

    if band_names is None:
        band_names = [
            'B1_Coastal', 'B2_Blue', 'B3_Green', 'B4_Red',
            'B5_RedEdge1', 'B6_RedEdge2', 'B7_RedEdge3', 'B8_NIR',
```

```
                'B8A_RedEdge4', 'B9_WaterVapor', 'B11_SWIR1', 'B12_SWIR2'
            ]

    print("\n" + "=" * 80)
    print("OUTLIER ANALYSIS SUMMARY")
    print("=" * 80)

    # Create summary table
    outlier_df = pd.DataFrame({
        'Band': band_names,
        'Outlier_Count': outlier_stats['band_outlier_counts'],
        'Outlier_Percentage': outlier_stats['band_outlier_percentages']
    })

    print(outlier_df.to_string(index=False, float_format='%.2f'))

    # Analysis
    print(f"\nOUTLIER INSIGHTS:")
    print("-" * 16)

    # Highest outlier percentage
    max_outlier_idx = np.argmax(outlier_stats['band_outlier_percentages'])
    max_outlier_pct = outlier_stats['band_outlier_percentages'][max_outlier_idx]
    print(f"• Highest outlier rate: {band_names[max_outlier_idx]} ({max_outlier_pct

    # Average outlier rate
    avg_outlier_pct = np.mean(outlier_stats['band_outlier_percentages'])
    print(f"• Average outlier rate: {avg_outlier_pct:.2f}%")

    # Bands with high outlier rates
    high_outlier_bands = [band_names[i] for i, pct in enumerate(outlier_stats['band
    if high_outlier_bands:
        print(f"• Bands with >2% outliers: {', '.join(high_outlier_bands)}")
        print("  → May indicate clouds, water bodies, or atmospheric artifacts")

    # Total outliers
    total_outliers = sum(outlier_stats['band_outlier_counts'])
    print(f"• Total outlier pixels detected: {total_outliers:,}")


if __name__ == "__main__":

    print("Demonstrating data standardization...")

    data = np.load('sentinel2_rochester.npy')


    explain_standardization()

    data_standardized, stats_info = standardize(data, no_data_value=0)


    fig, outlier_stats = plot_histograms_with_outliers(data, data_standardized)
    plt.show()
```

```
    summarize_outlier_analysis(outlier_stats)

    print(f"\nStandardization complete!")
    print(f"Original data range: {np.nanmin(data):.3f} to {np.nanmax(data):.3f}")
    print(f"Standardized data range: {np.nanmin(data_standardized):.3f} to {np.nanm
```

```
Demonstrating data standardization...

================================================================================
WHAT STANDARDIZATION DOES TO YOUR DATA
================================================================================

Z-Score Formula:
---------------
  z = (x - μ) / σ

Purpose:
-------
  • Removes the units of measurement
  • Centers data around zero (mean = 0)
  • Scales data to unit variance (std = 1)
  • Makes bands comparable despite different reflectance ranges

Benefits:
--------
  • Enables comparison across spectral bands
  • Highlights unusual pixels (outliers)
  • Normalizes for machine learning algorithms
  • Reveals distribution shapes more clearly

Interpretation:
--------------
  • z = 0: Average reflectance for that band
  • z = +1: One standard deviation above average (brighter)
  • z = -1: One standard deviation below average (darker)
  • z > +3: Extreme outliers (very bright pixels - clouds, snow, metal)
  • z < -3: Extreme outliers (very dark pixels - deep water, shadows)

Remote Sensing Applications:
---------------------------
  • Outlier detection: Clouds, shadows, water bodies
  • Anomaly detection: Urban heat islands, pollution
  • Change detection: Compare images from different dates
  • Classification: Normalize features for ML algorithms
Standardizing Sentinel-2 bands...
==================================================
Band  1: Mean=0.0887, Std=0.0279, Valid pixels=630,024
Band  2: Mean=0.0925, Std=0.0350, Valid pixels=630,024
Band  3: Mean=0.1055, Std=0.0344, Valid pixels=630,024
Band  4: Mean=0.0943, Std=0.0445, Valid pixels=630,024
Band  5: Mean=0.1367, Std=0.0409, Valid pixels=630,024
Band  6: Mean=0.2436, Std=0.0610, Valid pixels=630,024
Band  7: Mean=0.2858, Std=0.0773, Valid pixels=630,024
Band  8: Mean=0.2914, Std=0.0803, Valid pixels=630,024
Band  9: Mean=0.3035, Std=0.0822, Valid pixels=630,024
Band 10: Mean=0.3451, Std=0.0779, Valid pixels=630,024
Band 11: Mean=0.1917, Std=0.0484, Valid pixels=630,024
Band 12: Mean=0.1292, Std=0.0486, Valid pixels=630,024
==================================================
Standardization complete!
```


Sentinel-2 Band Histograms: Original vs Standardized Data
Outlier Detection and Distribution Analysis
B1 Coastal Original Data
Outliers (7092)
Mean: 0.089
B1 Coastal Standardized (Z-scores)
Outliers: 1.1%
B2 Blue Original Data
Outliers (6636)
Mean: 0.093
B2 Blue Standardized (Z-scores)
Outliers: 1.1%
B3 Green Original Data
Outliers (6573)
Mean: 0.105
B3 Green Standardized (Z-scores)
Outliers: 1.0%
B4 Red Original Data
Outliers (6903)
Mean: 0.094
B4 Red Standardized (Z-scores)
Outliers: 1.1%
B5 RedEdge1 Original Data
Outliers (6261)
Mean: 0.137
B5 RedEdge1 Standardized (Z-scores)
Outliers: 1.0%
B6 RedEdge2 Original Data
Outliers (17324)
Mean: 0.244
B6 RedEdge2 Standardized (Z-scores)
Outliers: 2.7%
B7 RedEdge3 Original Data
Outliers (14928)
Mean: 0.286
B7 RedEdge3 Standardized (Z-scores)
Outliers: 2.4%
B8 NIR Original Data
Outliers (14348)
Mean: 0.291
B8 NIR Standardized (Z-scores)
Outliers: 2.3%
B8A RedEdge4 Original Data
Outliers (15852)
Mean: 0.304
B8A RedEdge4 Standardized (Z-scores)
Outliers: 2.5%
B9 WaterVapor Original Data
Outliers (16506)
Mean: 0.345
B9 WaterVapor Standardized (Z-scores)
Outliers: 2.6%
B11 SWIR1 Original Data
Outliers (20648)
Mean: 0.192
B11 SWIR1 Standardized (Z-scores)
Outliers: 3.3%
B12 SWIR2 Original Data
Outliers (14252)
Mean: 0.129
B12 SWIR2 Standardized (Z-scores)
Outliers: 2.3%
Density
Surface Reflectance
Z-score

```
================================================================================
OUTLIER ANALYSIS SUMMARY
================================================================================
         Band  Outlier_Count  Outlier_Percentage
   B1_Coastal           7092                1.13
      B2_Blue           6636                1.05
     B3_Green           6573                1.04
       B4_Red           6903                1.10
  B5_RedEdge1           6261                0.99
  B6_RedEdge2          17324                2.75
  B7_RedEdge3          14928                2.37
       B8_NIR          14348                2.28
 B8A_RedEdge4          15852                2.52
B9_WaterVapor          16506                2.62
    B11_SWIR1          20648                3.28
    B12_SWIR2          14252                2.26

OUTLIER INSIGHTS:
----------------
• Highest outlier rate: B11_SWIR1 (3.28%)
• Average outlier rate: 1.95%
• Bands with >2% outliers: B6_RedEdge2, B7_RedEdge3, B8_NIR, B8A_RedEdge4, B9_WaterV
apor, B11_SWIR1, B12_SWIR2
  → May indicate clouds, water bodies, or atmospheric artifacts
• Total outlier pixels detected: 147,323

Standardization complete!
Original data range: 0.000 to 0.929
Standardized data range: -3.703 to 18.904
```

EXPLANATION

In addition to handling no-data by generating validity masks that omit pixels with values of 0, NaN, or negative numbers, this script conducts a thorough statistical analysis of the Sentinel-2 satellite imagery data, computing comprehensive descriptive statistics (mean, standard deviation, quartiles, and skewness) for each of the 12 spectral bands. By comparing reflectance patterns between visible, NIR, and SWIR bands to identify vegetation signatures, identifying high-variability bands that indicate diverse land cover, analyzing distribution skewness to identify bands with bright outlier pixels, and reporting the percentage of valid pixels per band, the code then performs an analysis of these statistics to provide insights about the characteristics of the landscape. In the end, the code provides an automated interpretation of the spectral characteristics and heterogeneity of the Rochester summer landscape.

Part2 - This script uses the formula $z=(x-\mu)/\sigma$ to standardize the data (z-score normalization) on the Sentinel-2 imagery, converting all spectral bands to a common scale with mean=0 and standard deviation=1. It handles no-data by removing pixels with values of 0, NaN, or negative numbers from the computations. The code then generates detailed histogram visualizations that compare the original vs. standardized data distributions for each band

and finds statistical outliers (pixels with $|z\text{-score}| > 3$), which usually represent extreme features like clouds, shadows, water bodies, or bright surfaces.