In [4]:
```
!pip install geopandas
```

```
Collecting geopandas
  Downloading geopandas-1.1.0-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: numpy>=1.24 in c:\users\user\anaconda3\lib\site-packa
ges (from geopandas) (1.26.4)
Collecting pyogrio>=0.7.2 (from geopandas)
  Downloading pyogrio-0.11.0-cp312-cp312-win_amd64.whl.metadata (5.4 kB)
Requirement already satisfied: packaging in c:\users\user\anaconda3\lib\site-package
s (from geopandas) (24.1)
Requirement already satisfied: pandas>=2.0.0 in c:\users\user\anaconda3\lib\site-pac
kages (from geopandas) (2.2.2)
Collecting pyproj>=3.5.0 (from geopandas)
  Downloading pyproj-3.7.1-cp312-cp312-win_amd64.whl.metadata (31 kB)
Collecting shapely>=2.0.0 (from geopandas)
  Downloading shapely-2.1.1-cp312-cp312-win_amd64.whl.metadata (7.0 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\user\anaconda3\lib
\site-packages (from pandas>=2.0.0->geopandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\anaconda3\lib\site-pack
ages (from pandas>=2.0.0->geopandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\user\anaconda3\lib\site-pa
ckages (from pandas>=2.0.0->geopandas) (2023.3)
Requirement already satisfied: certifi in c:\users\user\anaconda3\lib\site-packages
(from pyogrio>=0.7.2->geopandas) (2024.8.30)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-packages
(from python-dateutil>=2.8.2->pandas>=2.0.0->geopandas) (1.16.0)
Downloading geopandas-1.1.0-py3-none-any.whl (338 kB)
Downloading pyogrio-0.11.0-cp312-cp312-win_amd64.whl (19.2 MB)
   ---------------------------------------- 0.0/19.2 MB ? eta -:--:--
   - -------------------------------------- 0.8/19.2 MB 4.2 MB/s eta 0:00:05
   --- ------------------------------------ 1.6/19.2 MB 4.0 MB/s eta 0:00:05
   ----- ---------------------------------- 2.6/19.2 MB 4.2 MB/s eta 0:00:04
   ------ --------------------------------- 3.1/19.2 MB 4.0 MB/s eta 0:00:04
   -------- ------------------------------- 3.9/19.2 MB 3.9 MB/s eta 0:00:04
   ---------- ----------------------------- 5.0/19.2 MB 3.9 MB/s eta 0:00:04
   ------------ --------------------------- 5.8/19.2 MB 3.9 MB/s eta 0:00:04
   ------------- -------------------------- 6.3/19.2 MB 3.9 MB/s eta 0:00:04
   -------------- ------------------------- 7.1/19.2 MB 3.8 MB/s eta 0:00:04
   ---------------- ----------------------- 8.1/19.2 MB 3.8 MB/s eta 0:00:03
   ------------------ --------------------- 8.9/19.2 MB 3.8 MB/s eta 0:00:03
   -------------------- ------------------- 9.7/19.2 MB 3.9 MB/s eta 0:00:03
   ---------------------- ----------------- 10.7/19.2 MB 3.9 MB/s eta 0:00:03
   ------------------------ --------------- 11.8/19.2 MB 4.0 MB/s eta 0:00:02
   -------------------------- ------------- 12.8/19.2 MB 4.0 MB/s eta 0:00:02
   ---------------------------- ----------- 13.9/19.2 MB 4.1 MB/s eta 0:00:02
   ------------------------------ --------- 14.7/19.2 MB 4.1 MB/s eta 0:00:02
   -------------------------------- ------- 15.5/19.2 MB 4.1 MB/s eta 0:00:01
   ---------------------------------- ----- 16.5/19.2 MB 4.1 MB/s eta 0:00:01
   ------------------------------------ --- 17.3/19.2 MB 4.1 MB/s eta 0:00:01
   -------------------------------------- - 18.6/19.2 MB 4.1 MB/s eta 0:00:01
   ---------------------------------------  19.1/19.2 MB 4.1 MB/s eta 0:00:01
   ---------------------------------------- 19.2/19.2 MB 3.9 MB/s eta 0:00:00
Downloading pyproj-3.7.1-cp312-cp312-win_amd64.whl (6.3 MB)
   ---------------------------------------- 0.0/6.3 MB ? eta -:--:--
   ------ --------------------------------- 1.0/6.3 MB 5.6 MB/s eta 0:00:01
   ------------- -------------------------- 2.1/6.3 MB 5.3 MB/s eta 0:00:01
   -------------------- ------------------- 3.1/6.3 MB 5.1 MB/s eta 0:00:01
   -------------------------- ------------- 4.2/6.3 MB 4.9 MB/s eta 0:00:01
```

```
   -------------------------------- ------ 5.2/6.3 MB 5.0 MB/s eta 0:00:01
   ------------------------------------- - 6.0/6.3 MB 4.7 MB/s eta 0:00:01
   --------------------------------------- 6.3/6.3 MB 4.5 MB/s eta 0:00:00
Downloading shapely-2.1.1-cp312-cp312-win_amd64.whl (1.7 MB)
   --------------------------------------- 0.0/1.7 MB ? eta -:--:--
   ------------------ -------------------- 0.8/1.7 MB 4.8 MB/s eta 0:00:01
   ----------------------------------- --- 1.6/1.7 MB 4.6 MB/s eta 0:00:01
   --------------------------------------- 1.7/1.7 MB 3.0 MB/s eta 0:00:00
Installing collected packages: shapely, pyproj, pyogrio, geopandas
Successfully installed geopandas-1.1.0 pyogrio-0.11.0 pyproj-3.7.1 shapely-2.1.1
```

In [5]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import geopandas as gpd
from matplotlib.colors import LinearSegmentedColormap
import matplotlib.ticker as ticker
from scipy import stats
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import warnings
warnings.filterwarnings('ignore')
```

In [6]:

```
#Set plot stying
plt.style.use('seaborn-v0_8-whitegrid')
plt.rcParams['figure.figsize'] = (14, 8)
plt.rcParams['axes.titlesize'] = 18
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12
```

In [7]:

```
#Load the data
maize=pd.read_csv("Maize-Production-2012-2018.csv")
```

In [8]:

```
maize.head()
```

Out[8]:

	County	Harvested Area (2012)	Production Metric Tonnes (2012)	Yield as Production over Harvest (2012)	Harvested Area (2013)	Production Metric Tonnes (2013)	Yield as Production over Harvest (2013)	Harv (
0	Baringo	39753.0	71867.0	1.81	29117.0	55805.0	1.92	34
1	Bomet	32697.0	73278.0	2.24	30620.0	72236.0	2.36	30
2	Bungoma	96209.0	262381.0	2.73	92705.0	221586.0	2.39	98
3	Busia	41990.0	50102.0	1.19	45898.0	63230.0	1.38	50
4	Elgeyo Marakwet	31533.0	91964.0	2.92	32015.0	101336.0	3.17	27

5 rows × 22 columns

In [13]:

```
#Extract year from columns to get real column name
#Here we are extracting year from every column for and returning none to the column
def extract_year(col_name):
    if "(" in col_name and ")" in col_name:
        return col_name.split("(")[1].split(")")[0]
    return None
    #Create Lists to hold the different types of data/act like storage of data. In
harvested_cols = [col for col in maize.columns if "Harvested Area" in col]
production_cols = [col for col in maize.columns if "Production Metric Tonnes" in co
yield_cols = [col for col in maize.columns if "Yield" in col]

# Create melted dataframes for each data type
#melt is used to reshape a data frame from a wide format to a long format. id_vars
#to keep columns are they are.value_vars are the columns to be melted,that is unpiv
#colums that will contain the values of the melted columns.
harvested_df = pd.melt(maize, id_vars=['County'], value_vars=harvested_cols,
                       var_name='Year_Column', value_name='Harvested_Area')
harvested_df['Year'] = harvested_df['Year_Column'].apply(extract_year)
harvested_df = harvested_df.drop('Year_Column', axis=1)

production_df = pd.melt(maize, id_vars=['County'], value_vars=production_cols,
                        var_name='Year_Column', value_name='Production_MT')
production_df['Year'] = production_df['Year_Column'].apply(extract_year)
production_df = production_df.drop('Year_Column', axis=1)

yield_df = pd.melt(maize, id_vars=['County'], value_vars=yield_cols,
                   var_name='Year_Column', value_name='Yield')
yield_df['Year'] = yield_df['Year_Column'].apply(extract_year)
yield_df = yield_df.drop('Year_Column', axis=1)
```

In [14]:

```
# Merge the three dataframes
maize_data = pd.merge(harvested_df, production_df, on=['County', 'Year'])
maize_data = pd.merge(maize_data, yield_df, on=['County', 'Year'])
```

```
# Convert Year to integer for easier sorting
maize_data['Year'] = maize_data['Year'].astype(int)

# Sort by County and Year
maize_data = maize_data.sort_values(['County', 'Year'])

# Check the resulting dataframe
print("\nReshaped Dataset:")
print(f"Number of records: {maize_data.shape[0]}")
print(f"Number of columns: {maize_data.shape[1]}")
maize_data.head()
```

```
Reshaped Dataset:
Number of records: 329
Number of columns: 5
```

Out[14]:

	County	Harvested_Area	Year	Production_MT	Yield
0	Baringo	39753.0	2012	71867.0	1.81
47	Baringo	29117.0	2013	55805.0	1.92
94	Baringo	34960.0	2014	34959.0	1.00
141	Baringo	44159.0	2015	83313.0	1.89
188	Baringo	33163.0	2016	72495.0	2.19

In [15]:

```
#EDA
# Check for any data quality issues
print("\nChecking for missing values:")
print(maize_data.isnull().sum())

# Check for potential outliers or inconsistencies
print("\nSummary statistics:")
print(maize_data.describe())

# Find any very unusual values that may need investigation
print("\nPotential data issues (checking for negative values):")
print("Negative harvested area:", (maize_data['Harvested_Area'] < 0).sum())
print("Negative production:", (maize_data['Production_MT'] < 0).sum())
print("Negative yield:", (maize_data['Yield'] < 0).sum())

print("\nVery large values that might be errors:")
print("Harvested area > 200,000 hectares:", (maize_data['Harvested_Area'] > 200000)
print("Production > 500,000 MT:", (maize_data['Production_MT'] > 500000).sum())
print("Yield > 5:", (maize_data['Yield'] > 5).sum())

# Let's also check for zeros that might indicate missing data
print("\nZero values:")
print("Zero harvested area:", (maize_data['Harvested_Area'] == 0).sum())
print("Zero production:", (maize_data['Production_MT'] == 0).sum())
print("Zero yield:", (maize_data['Yield'] == 0).sum())
```

```
Checking for missing values:
County            0
Harvested_Area    0
Year              0
Production_MT     0
Yield             0
dtype: int64

Summary statistics:
       Harvested_Area         Year  Production_MT       Yield
count      329.000000   329.000000     329.000000  329.000000
mean     46377.701337  2015.000000   77929.065167    1.449696
std      39396.485062     2.003046   96222.204826    0.925565
min          0.000000  2012.000000       0.000000    0.000000
25%      16663.000000  2013.000000   12508.000000    0.790000
50%      35549.000000  2015.000000   48121.000000    1.240000
75%      73191.000000  2017.000000   97513.000000    1.890000
max     273056.000000  2018.000000  548196.010000    5.090000

Potential data issues (checking for negative values):
Negative harvested area: 0
Negative production: 0
Negative yield: 0

Very large values that might be errors:
Harvested area > 200,000 hectares: 1
Production > 500,000 MT: 1
Yield > 5: 1

Zero values:
Zero harvested area: 1
Zero production: 1
Zero yield: 2
```

In [16]:

```
# Calculate total production per year
yearly_production = maize_data.groupby('Year')['Production_MT'].sum()
print("\nTotal Kenya Maize Production by Year (Metric Tonnes):")
print(yearly_production)
```

```
Total Kenya Maize Production by Year (Metric Tonnes):
Year
2012    3751205.00
2013    3592692.00
2014    3496079.00
2015    3972614.00
2016    3339182.00
2017    3473113.00
2018    4013777.44
Name: Production_MT, dtype: float64
```

In [19]:

```
# Calculate total harvested area per year
yearly_harvested = maize_data.groupby('Year')['Harvested_Area'].sum()
print("\nTotal Kenya Maize Harvested Area by Year (Hectares):")
print(yearly_harvested)
```

```
Total Kenya Maize Harvested Area by Year (Hectares):
Year
2012    2159321.00
2013    2123140.00
2014    2116142.00
2015    2267152.00
2016    2337586.00
2017    2113180.00
2018    2141742.74
Name: Harvested_Area, dtype: float64
```

In [20]:
```
# Calculate average yield per year
yearly_yield = yearly_production / yearly_harvested
print("\nAverage Kenya Maize Yield by Year (MT/Hectare):")
print(yearly_yield)
```

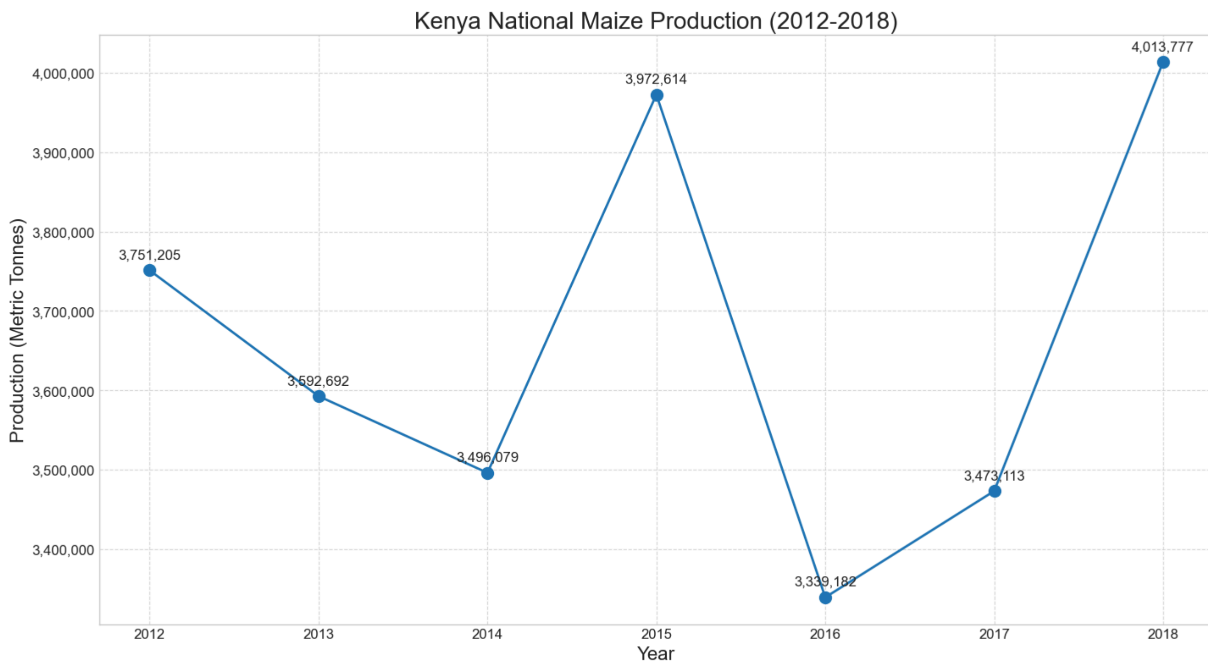
```
Average Kenya Maize Yield by Year (MT/Hectare):
Year
2012    1.737215
2013    1.692160
2014    1.652100
2015    1.752249
2016    1.428475
2017    1.643548
2018    1.874071
dtype: float64
```

In [21]:
```
# Analyze the top maize-producing counties
top_counties_2018 = maize_data[maize_data['Year'] == 2018].sort_values('Production_
print("\nTop 10 Maize-Producing Counties in 2018:")
print(top_counties_2018[['County', 'Harvested_Area', 'Production_MT', 'Yield']])
```

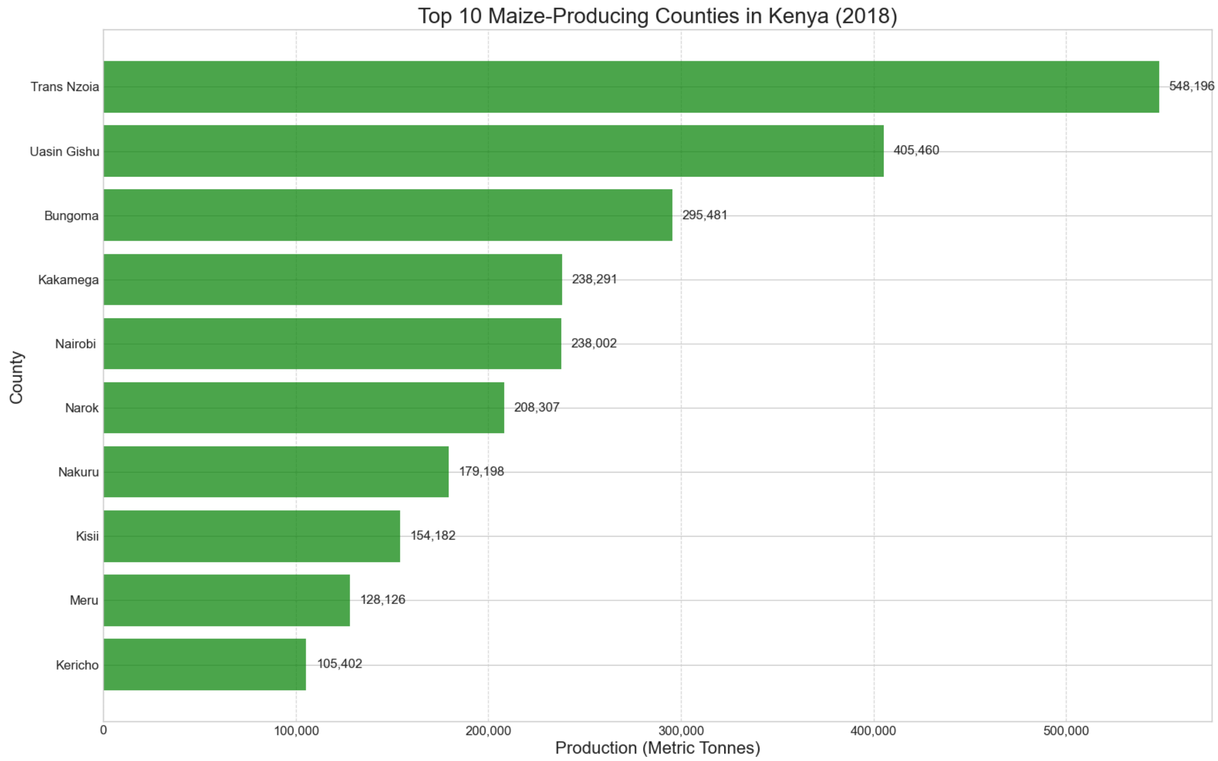
```
Top 10 Maize-Producing Counties in 2018:
          County  Harvested_Area  Production_MT  Yield
323  Trans Nzoia        107681.0      548196.01   5.09
325  Uasin Gishu         95209.0      405459.68   4.26
284      Bungoma         93484.0      295481.10   3.16
292     Kakamega         95387.0      238290.55   2.50
311     Nairobi          86102.0      238002.40   2.76
314        Narok         91602.0      208306.85   2.27
312       Nakuru         47830.0      179198.25   3.75
297        Kisii         74162.0      154181.90   2.08
307         Meru         82153.0      128126.34   1.56
293      Kericho         33461.0      105402.15   3.15
```

In [22]:
```
# 1. National Production Trends Over Time
plt.figure(figsize=(14, 8))
plt.plot(yearly_production.index, yearly_production.values, marker='o', linestyle='
plt.title('Kenya National Maize Production (2012-2018)', fontsize=20)
plt.xlabel('Year', fontsize=16)
plt.ylabel('Production (Metric Tonnes)', fontsize=16)
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(yearly_production.index)
plt.tight_layout()

# Format y-axis with commas for thousands
plt.gca().yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))
```

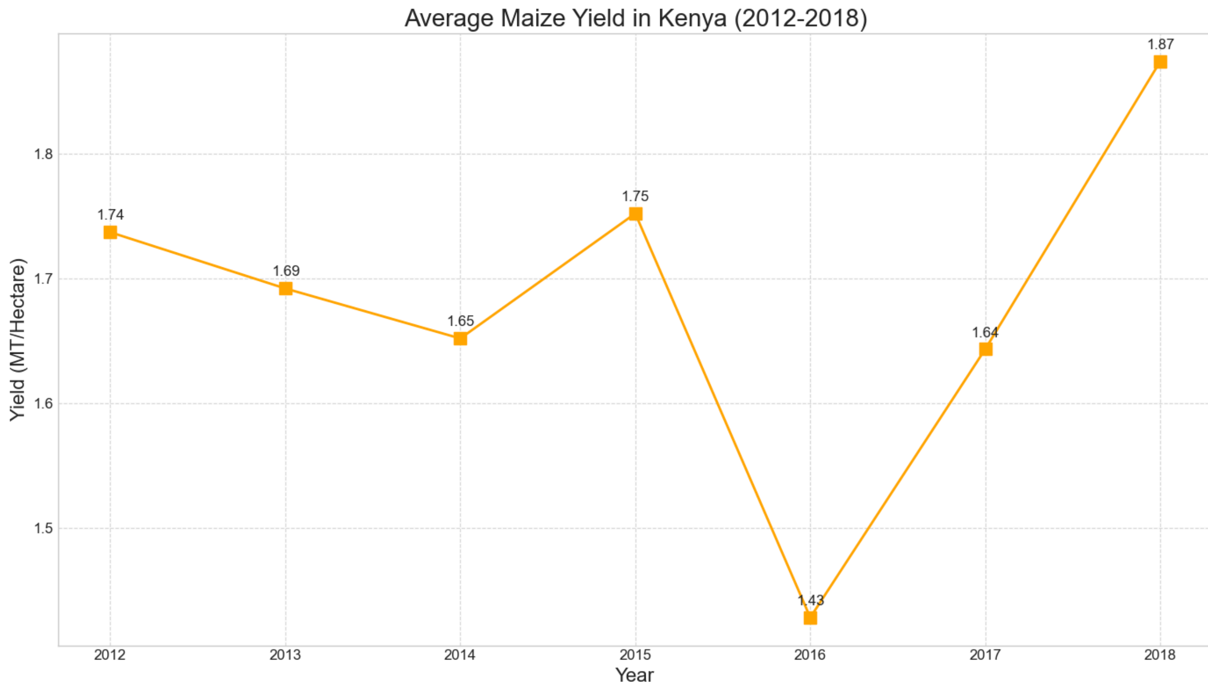
```
# Add data labels above each point
for year, prod in zip(yearly_production.index, yearly_production.values):
    plt.annotate(f'{prod:,.0f}',
                 (year, prod),
                 textcoords="offset points",
                 xytext=(0,10),
                 ha='center',
                 fontsize=12)
plt.savefig('kenya_maize_production_trend.png', dpi=300, bbox_inches='tight')
plt.show()
```



In [23]:
```
# 2. Top Producing Counties Comparison (2018)
plt.figure(figsize=(16, 10))
top_counties = top_counties_2018.sort_values('Production_MT')

# Create horizontal bar chart
plt.barh(top_counties['County'], top_counties['Production_MT'], color='green', alph
plt.title('Top 10 Maize-Producing Counties in Kenya (2018)', fontsize=20)
plt.xlabel('Production (Metric Tonnes)', fontsize=16)
plt.ylabel('County', fontsize=16)
plt.grid(True, axis='x', linestyle='--', alpha=0.7)

# Format x-axis with commas for thousands
plt.gca().xaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

# Add data labels to the end of each bar
for i, prod in enumerate(top_counties['Production_MT']):
    plt.text(prod + 5000, i, f'{prod:,.0f}', va='center', fontsize=12)

plt.tight_layout()
plt.savefig('top_counties_maize_production.png', dpi=300, bbox_inches='tight')
plt.show()
```



In [24]:
```
# 3. Yield Comparison Across Years
plt.figure(figsize=(14, 8))
plt.plot(yearly_yield.index, yearly_yield.values, marker='s', linestyle='-', linewi
plt.title('Average Maize Yield in Kenya (2012-2018)', fontsize=20)
plt.xlabel('Year', fontsize=16)
plt.ylabel('Yield (MT/Hectare)', fontsize=16)
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(yearly_yield.index)

# Add data labels above each point
for year, yld in zip(yearly_yield.index, yearly_yield.values):
    plt.annotate(f'{yld:.2f}',
                (year, yld),
                textcoords="offset points",
                xytext=(0,10),
                ha='center',
                fontsize=12)

plt.tight_layout()
plt.savefig('kenya_maize_yield_trend.png', dpi=300, bbox_inches='tight')
plt.show()
```



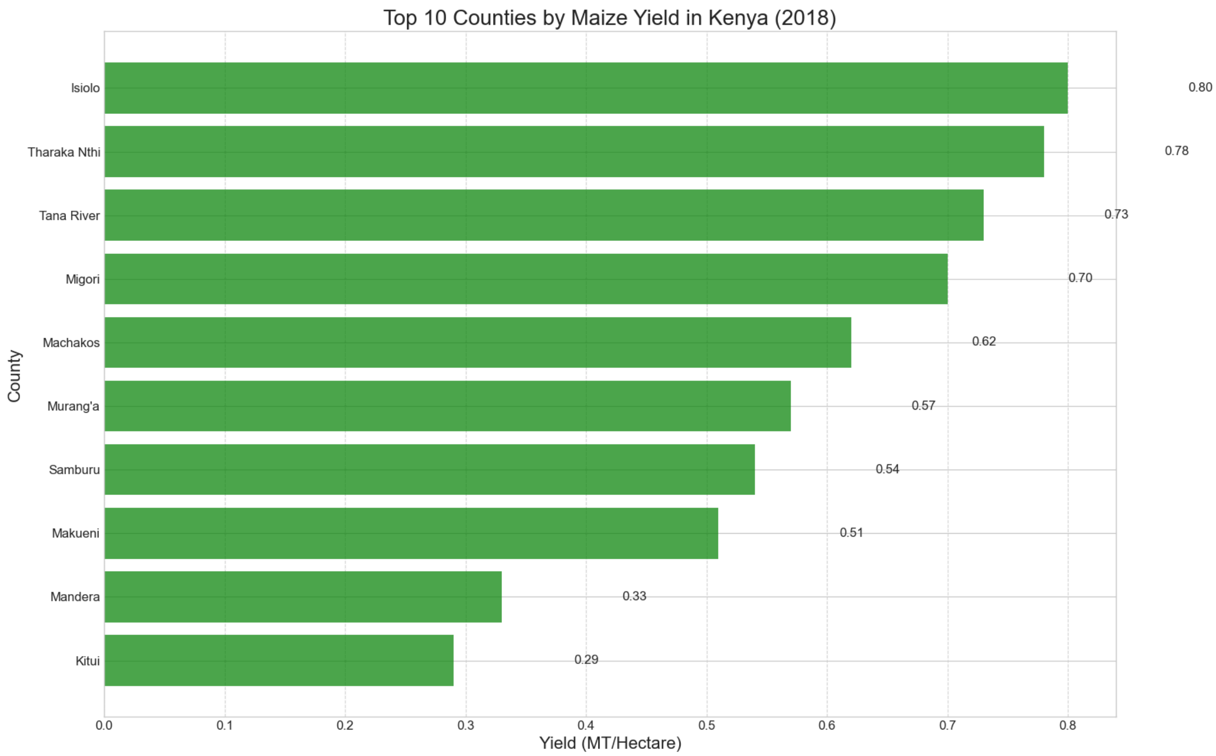
In [30]:

```
# 4. Production and Area Harvested Relationship (2018)
plt.figure(figsize=(16, 12))
data_2018 = maize_data[maize_data['Year'] == 2018]

# Filter out extreme outliers for better visualization
data_2018_filtered = data_2018[
    (data_2018['Harvested_Area'] <= data_2018['Harvested_Area'].quantile(0.95)) &
    (data_2018['Production_MT'] <= data_2018['Production_MT'].quantile(0.95))
]

# Create a scatter plot
plt.scatter(data_2018_filtered['Harvested_Area'],
           data_2018_filtered['Production_MT'],
           alpha=0.7,
           s=100,
           c=data_2018_filtered['Yield'],
           cmap='viridis')

plt.colorbar(label='Yield (MT/Hectare)')
plt.title('Relationship between Harvested Area and Production (2018)', fontsize=20)
plt.xlabel('Harvested Area (Hectares)', fontsize=16)
plt.ylabel('Production (Metric Tonnes)', fontsize=16)
plt.grid(True, linestyle='--', alpha=0.7)

# Add county labels to points
for idx, row in data_2018_filtered.iterrows():
    plt.annotate(row['County'],
                (row['Harvested_Area'], row['Production_MT']),
                fontsize=9,
                xytext=(5, 5),
                textcoords='offset points')

# Add a best fit line
x = data_2018_filtered['Harvested_Area']
```

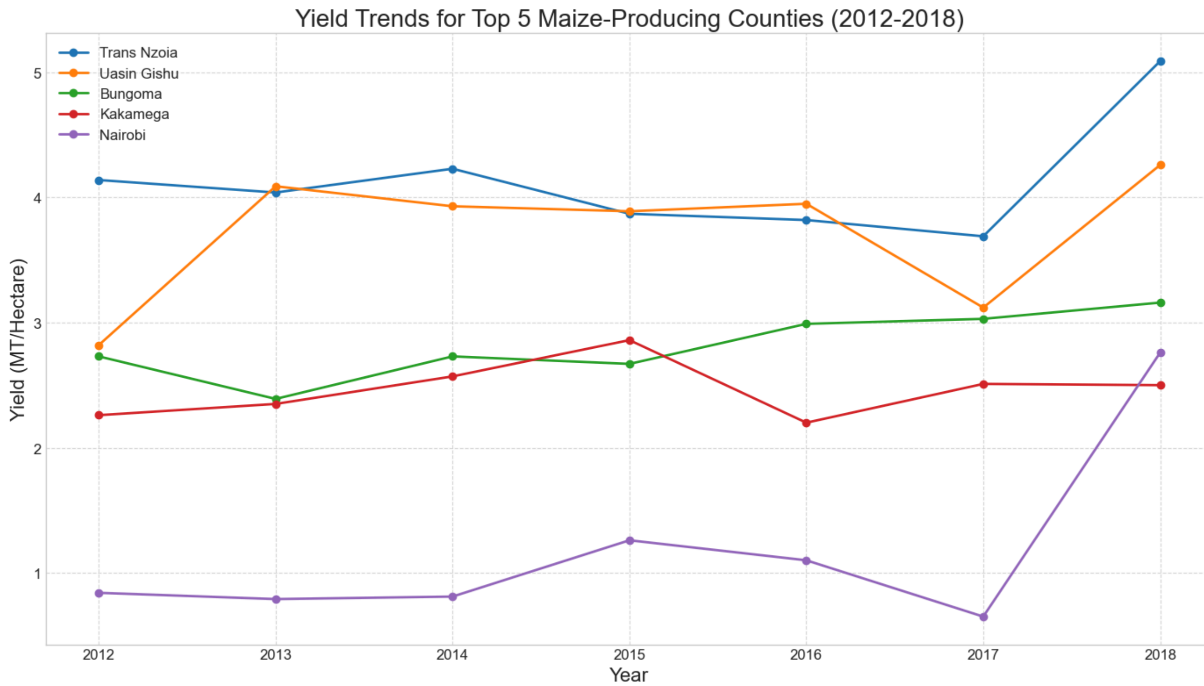
```
y = data_2018_filtered['Production_MT']
z = np.polyfit(x, y, 1)
p = np.poly1d(z)
plt.plot(x, p(x), "r--", alpha=0.7)

plt.tight_layout()
plt.savefig('area_production_relationship.png', dpi=300, bbox_inches='tight')
plt.show()
```

In [31]:
```
# 5. Top counties comparison by Yield (2018)
plt.figure(figsize=(16, 10))
top_yield_counties = maize_data[maize_data['Year'] == 2018].sort_values('Yield', as

# Create horizontal bar chart for yields
plt.barh(top_yield_counties['County'], top_yield_counties['Yield'], color='green',
plt.title('Top 10 Counties by Maize Yield in Kenya (2018)', fontsize=20)
plt.xlabel('Yield (MT/Hectare)', fontsize=16)
plt.ylabel('County', fontsize=16)
plt.grid(True, axis='x', linestyle='--', alpha=0.7)

# Add data labels
for i, yld in enumerate(top_yield_counties['Yield']):
    plt.text(yld + 0.1, i, f'{yld:.2f}', va='center', fontsize=12)

plt.tight_layout()
```

```
plt.savefig('top_counties_by_yield.png', dpi=300, bbox_inches='tight')
plt.show()
```



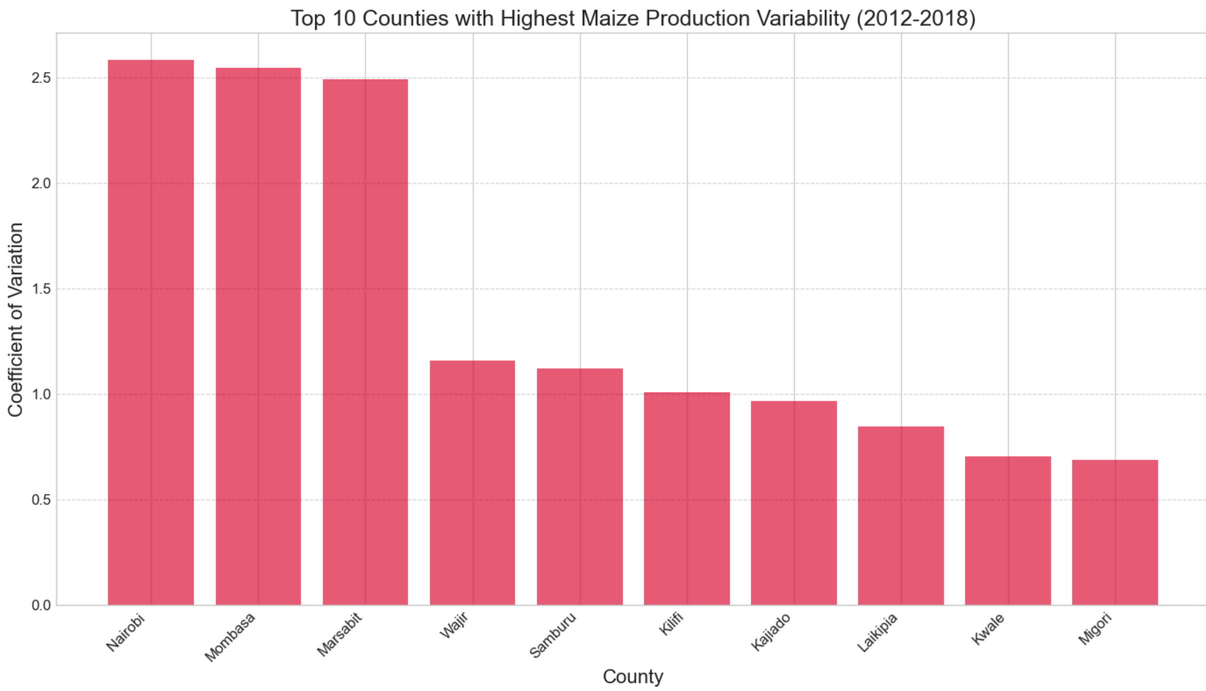
In [32]:
```
# 6. Historical yield trends for top 5 producing counties
top5_counties = top_counties_2018['County'].head(5).tolist()
top5_data = maize_data[maize_data['County'].isin(top5_counties)]

plt.figure(figsize=(14, 8))
for county in top5_counties:
    county_data = top5_data[top5_data['County'] == county]
    plt.plot(county_data['Year'], county_data['Yield'], marker='o', linewidth=2, la

plt.title('Yield Trends for Top 5 Maize-Producing Counties (2012-2018)', fontsize=2
plt.xlabel('Year', fontsize=16)
plt.ylabel('Yield (MT/Hectare)', fontsize=16)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=12)
plt.xticks(maize_data['Year'].unique())

plt.tight_layout()
plt.savefig('top5_counties_yield_trends.png', dpi=300, bbox_inches='tight')
plt.show()
```



In [33]:

```
# 8. Production Variability Analysis
# Calculate the coefficient of variation for each county to identify areas with hig
county_stats = maize_data.groupby('County').agg({
    'Production_MT': ['mean', 'std', 'min', 'max'],
    'Harvested_Area': ['mean', 'std', 'min', 'max'],
    'Yield': ['mean', 'std', 'min', 'max']
})

# Calculate coefficient of variation (CV = std / mean)
county_stats['Production_CV'] = county_stats[('Production_MT', 'std')] / county_sta
county_stats['Yield_CV'] = county_stats[('Yield', 'std')] / county_stats[('Yield',

# Sort by production variability
high_variability_counties = county_stats.sort_values('Production_CV', ascending=Fal
print("\nTop 10 Counties with Highest Production Variability:")
print(high_variability_counties['Production_CV'])

# Plot the counties with high variability
plt.figure(figsize=(14, 8))
counties = high_variability_counties.index
cv_values = high_variability_counties['Production_CV'].values

plt.bar(counties, cv_values, color='crimson', alpha=0.7)
plt.title('Top 10 Counties with Highest Maize Production Variability (2012-2018)',
plt.xlabel('County', fontsize=16)
plt.ylabel('Coefficient of Variation', fontsize=16)
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.savefig('production_variability_counties.png', dpi=300, bbox_inches='tight')
plt.show()
```

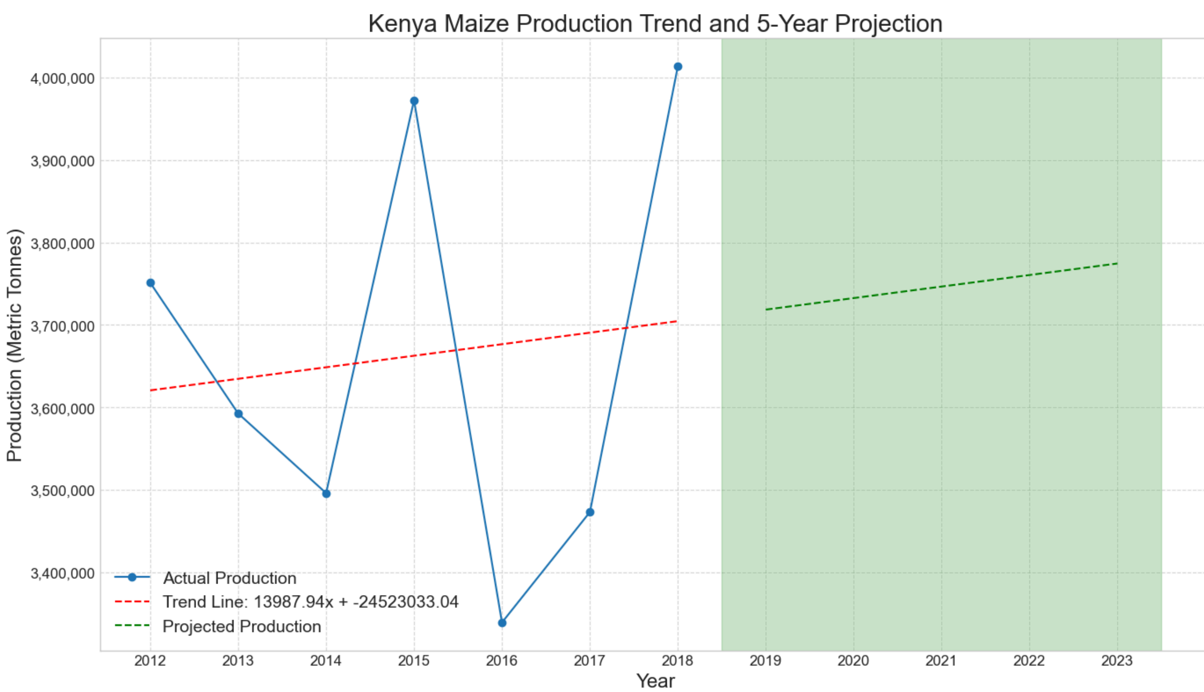
```
Top 10 Counties with Highest Production Variability:
County
Nairobi     2.580368
Mombasa     2.542622
Marsabit    2.489524
Wajir       1.159469
Samburu     1.120351
Kilifi      1.006939
Kajiado     0.966897
Laikipia    0.846451
Kwale       0.703715
Migori      0.687729
Name: Production_CV, dtype: float64
```



In [34]:

```python
# 9. Time Series Decomposition for National Production
# This helps identify trend, seasonality, and residual components
from statsmodels.tsa.seasonal import seasonal_decompose

# Fit a linear trend line to the production data
x = np.array(yearly_production.index)
y = np.array(yearly_production.values)
z = np.polyfit(x, y, 1)
p = np.poly1d(z)

# Plot the trend
plt.figure(figsize=(14, 8))
plt.plot(x, y, 'o-', label='Actual Production')
plt.plot(x, p(x), 'r--', label=f'Trend Line: {z[0]:.2f}x + {z[1]:.2f}')

# Project forward 5 years
future_years = np.array(range(2019, 2024))
projected_values = p(future_years)

plt.plot(future_years, projected_values, 'g--', label='Projected Production')
```

```python
# Highlight the projected region
plt.axvspan(2018.5, 2023.5, alpha=0.2, color='green')

# Add labels and formatting
plt.title('Kenya Maize Production Trend and 5-Year Projection', fontsize=20)
plt.xlabel('Year', fontsize=16)
plt.ylabel('Production (Metric Tonnes)', fontsize=16)
plt.grid(True, linestyle='--', alpha=0.7)
plt.xticks(np.concatenate([x, future_years]))
plt.legend(fontsize=14)

# Format y-axis with commas for thousands
plt.gca().yaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))

plt.tight_layout()
plt.savefig('production_trend_projection.png', dpi=300, bbox_inches='tight')
plt.show()

# Calculate and display the projected values
print("\nProjected Maize Production for Next 5 Years:")
for year, proj in zip(future_years, projected_values):
    print(f"{year}: {proj:,.0f} MT")
```




```
Projected Maize Production for Next 5 Years:
2019: 3,718,618 MT
2020: 3,732,606 MT
2021: 3,746,594 MT
2022: 3,760,582 MT
2023: 3,774,570 MT
```

In [35]:
```python
maize_data.to_csv(r"C:\Users\user\Downloads\Updated Maize-Production-2012-2018.csv"
```

In [ ]: