# Exploratory Data Analysis: Texas Agriculture with Climate Data (2000-2023)

**Project:** Predicting Agricultural Statistics for Texas Counties Using Climate Data

**Authors:** Carter Dobbs, Johann Steinhoff, Jay Suh

**Date:** November 2025

---

## Overview

This notebook performs exploratory data analysis on a merged dataset combining:

- USDA NASS QuickStats agricultural data for Texas counties
- NOAA nClimDiv monthly climate data

The goal is to understand the dataset structure, identify patterns, and prepare for predictive modeling of agricultural outcomes based on climate conditions.

In [1]:
```
# Import necessary libraries
import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

print("Python:", sys.version)
print("NumPy:", np.__version__)
print("Pandas:", pd.__version__)

# Configuration
warnings.filterwarnings('ignore')
```

```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 100)
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette('husl')

print("Libraries imported successfully!")
```

```
Python: 3.13.9 (tags/v3.13.9:8183fa5, Oct 14 2025, 14:09:13) [MSC v.1944 64 bit (AMD64)]
NumPy: 2.3.1
Pandas: 2.3.2
Libraries imported successfully!
```

---

# 1. Load the Dataset

In [2]:

```
# Load the merged agriculture and climate dataset
data_path = "../data/texas_agriculture_with_climate_2000_2023.csv"
df = pd.read_csv(data_path, low_memory=False)

print(f"Dataset loaded successfully!")
print(f"Shape: {df.shape}")
print(f"Total cells: {df.shape[0] * df.shape[1]:,}")
```

```
Dataset loaded successfully!
Shape: (398204, 120)
Total cells: 47,784,480
```

---

# 2. Dataset Size and Structure

## 2.1 Basic Information

In [3]:

```
# Display dataset dimensions
print("="*80)
print("DATASET DIMENSIONS")
print("="*80)
print(f"Number of instances (records): {df.shape[0]:,}")
print(f"Number of attributes (columns): {df.shape[1]:,}")
```

```
print(f"Total data points: {df.shape[0] * df.shape[1]:,}")
print(f"Memory usage: {df.memory_usage(deep=True).sum() / (1024**2):.2f} MB")
print("="*80)
```

```
================================================================================
DATASET DIMENSIONS
================================================================================
Number of instances (records): 398,204
Number of attributes (columns): 120
Total data points: 47,784,480
Memory usage: 842.87 MB
================================================================================
```

In [4]:

```
# Display first few rows
print("\nFirst 5 rows of the dataset:")
df.head()
```

```
First 5 rows of the dataset:
```

Out[4]:

| | SOURCE_DESC | SECTOR_DESC | GROUP_DESC | COMMODITY_DESC | CLASS_DESC | PRODN_PRACTICE_DESC | UTIL_PRACTICE_DESC |
|---|---|---|---|---|---|---|---|
| **0** | CENSUS | CROPS | VEGETABLES | SQUASH | WINTER | ALL PRODUCTION PRACTICES | FRESH MARKET |
| **1** | CENSUS | CROPS | FRUIT & TREE NUTS | PECANS | ALL CLASSES | ALL PRODUCTION PRACTICES | ALL UTILIZATION PRACTICES |
| **2** | CENSUS | CROPS | FRUIT & TREE NUTS | TANGERINES | ALL CLASSES | ALL PRODUCTION PRACTICES | ALL UTILIZATION PRACTICES |
| **3** | CENSUS | CROPS | FRUIT & TREE NUTS | PECANS | ALL CLASSES | ALL PRODUCTION PRACTICES | ALL UTILIZATION PRACTICES |
| **4** | CENSUS | CROPS | FIELD CROPS | PEANUTS | ALL CLASSES | IRRIGATED | ALL UTILIZATION PRACTICES |

In [5]:

```
# Display data types and non-null counts
print("\nColumn Information:")
df.info()
```

```
Column Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398204 entries, 0 to 398203
Columns: 120 entries, SOURCE_DESC to ANNUAL_HDD
dtypes: float64(90), int64(6), object(24)
memory usage: 364.6+ MB
```

## 2.2 Column Names

In [6]:
```
# Display all column names
print("\nAll Columns in Dataset:")
print("="*80)
for i, col in enumerate(df.columns, 1):
    print(f"{i:3d}. {col}")
print("="*80)
```

```
All Columns in Dataset:
================================================================================
  1. SOURCE_DESC
  2. SECTOR_DESC
  3. GROUP_DESC
  4. COMMODITY_DESC
  5. CLASS_DESC
  6. PRODN_PRACTICE_DESC
  7. UTIL_PRACTICE_DESC
  8. STATISTICCAT_DESC
  9. UNIT_DESC
 10. SHORT_DESC
 11. DOMAIN_DESC
 12. DOMAINCAT_DESC
 13. AGG_LEVEL_DESC
 14. STATE_ANSI
 15. STATE_FIPS_CODE
 16. STATE_ALPHA
 17. STATE_NAME
 18. ASD_CODE
 19. ASD_DESC
 20. COUNTY_ANSI
 21. COUNTY_CODE
 22. COUNTY_NAME
 23. REGION_DESC
 24. ZIP_5
 25. WATERSHED_CODE
 26. WATERSHED_DESC
 27. CONGR_DISTRICT_CODE
 28. COUNTRY_CODE
 29. COUNTRY_NAME
 30. LOCATION_DESC
 31. YEAR
 32. FREQ_DESC
 33. BEGIN_CODE
 34. END_CODE
 35. REFERENCE_PERIOD_DESC
 36. WEEK_ENDING
 37. LOAD_TIME
 38. VALUE
 39. CV_%
 40. COUNTY_FIPS
```

41. PRECIP_JAN
42. PRECIP_FEB
43. PRECIP_MAR
44. PRECIP_APR
45. PRECIP_MAY
46. PRECIP_JUN
47. PRECIP_JUL
48. PRECIP_AUG
49. PRECIP_SEP
50. PRECIP_OCT
51. PRECIP_NOV
52. PRECIP_DEC
53. TMAX_JAN
54. TMAX_FEB
55. TMAX_MAR
56. TMAX_APR
57. TMAX_MAY
58. TMAX_JUN
59. TMAX_JUL
60. TMAX_AUG
61. TMAX_SEP
62. TMAX_OCT
63. TMAX_NOV
64. TMAX_DEC
65. TMIN_JAN
66. TMIN_FEB
67. TMIN_MAR
68. TMIN_APR
69. TMIN_MAY
70. TMIN_JUN
71. TMIN_JUL
72. TMIN_AUG
73. TMIN_SEP
74. TMIN_OCT
75. TMIN_NOV
76. TMIN_DEC
77. TAVG_JAN
78. TAVG_FEB
79. TAVG_MAR
80. TAVG_APR
81. TAVG_MAY
82. TAVG_JUN

```
 83. TAVG_JUL
 84. TAVG_AUG
 85. TAVG_SEP
 86. TAVG_OCT
 87. TAVG_NOV
 88. TAVG_DEC
 89. CDD_JAN
 90. CDD_FEB
 91. CDD_MAR
 92. CDD_APR
 93. CDD_MAY
 94. CDD_JUN
 95. CDD_JUL
 96. CDD_AUG
 97. CDD_SEP
 98. CDD_OCT
 99. CDD_NOV
100. CDD_DEC
101. HDD_JAN
102. HDD_FEB
103. HDD_MAR
104. HDD_APR
105. HDD_MAY
106. HDD_JUN
107. HDD_JUL
108. HDD_AUG
109. HDD_SEP
110. HDD_OCT
111. HDD_NOV
112. HDD_DEC
113. GROWING_SEASON_PRECIP
114. GROWING_SEASON_TEMP_AVG
115. GROWING_SEASON_TEMP_MAX
116. GROWING_SEASON_TEMP_MIN
117. ANNUAL_PRECIP
118. ANNUAL_TEMP_AVG
119. ANNUAL_CDD
120. ANNUAL_HDD
================================================================================
```

---

# 3. Attribute Categories and Descriptions

The dataset contains attributes from three main sources:

1. **USDA Agricultural Data** - Crop information and statistics
2. **NOAA Climate Data** - Monthly temperature and precipitation
3. **Engineered Features** - Derived seasonal and annual aggregates

In [7]:
```
# Categorize columns
usda_columns = [
    'SOURCE_DESC', 'SECTOR_DESC', 'GROUP_DESC', 'COMMODITY_DESC', 'CLASS_DESC',
    'PRODN_PRACTICE_DESC', 'UTIL_PRACTICE_DESC', 'STATISTICCAT_DESC', 'UNIT_DESC',
    'SHORT_DESC', 'DOMAIN_DESC', 'DOMAINCAT_DESC', 'AGG_LEVEL_DESC'
]

location_columns = [
    'STATE_ANSI', 'STATE_FIPS_CODE', 'STATE_ALPHA', 'STATE_NAME',
    'ASD_CODE', 'ASD_DESC', 'COUNTY_ANSI', 'COUNTY_CODE', 'COUNTY_NAME',
    'REGION_DESC', 'ZIP_5', 'WATERSHED_CODE', 'WATERSHED_DESC',
    'CONGR_DISTRICT_CODE', 'COUNTRY_CODE', 'COUNTRY_NAME', 'LOCATION_DESC',
    'COUNTY_FIPS'
]

temporal_columns = [
    'YEAR', 'FREQ_DESC', 'BEGIN_CODE', 'END_CODE', 'REFERENCE_PERIOD_DESC',
    'WEEK_ENDING', 'LOAD_TIME'
]

target_columns = ['VALUE', 'CV_%']

# Climate columns
months = ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC']
precip_columns = [f'PRECIP_{month}' for month in months]
tmax_columns = [f'TMAX_{month}' for month in months]
tmin_columns = [f'TMIN_{month}' for month in months]
tavg_columns = [f'TAVG_{month}' for month in months]
cdd_columns = [f'CDD_{month}' for month in months]
hdd_columns = [f'HDD_{month}' for month in months]
```

```python
engineered_columns = [
    'GROWING_SEASON_PRECIP', 'GROWING_SEASON_TEMP_AVG',
    'GROWING_SEASON_TEMP_MAX', 'GROWING_SEASON_TEMP_MIN',
    'ANNUAL_PRECIP', 'ANNUAL_TEMP_AVG', 'ANNUAL_CDD', 'ANNUAL_HDD'
]

print("\nAttribute Categories:")
print("="*80)
print(f"USDA Agricultural Attributes: {len(usda_columns)}")
print(f"Location/Geographic Attributes: {len(location_columns)}")
print(f"Temporal Attributes: {len(temporal_columns)}")
print(f"Target Variables: {len(target_columns)}")
print(f"Monthly Precipitation: {len(precip_columns)}")
print(f"Monthly Max Temperature: {len(tmax_columns)}")
print(f"Monthly Min Temperature: {len(tmin_columns)}")
print(f"Monthly Avg Temperature: {len(tavg_columns)}")
print(f"Monthly Cooling Degree Days: {len(cdd_columns)}")
print(f"Monthly Heating Degree Days: {len(hdd_columns)}")
print(f"Engineered Climate Features: {len(engineered_columns)}")
print("="*80)
print(f"Total: {len(df.columns)} columns")
```

```
Attribute Categories:
================================================================================
USDA Agricultural Attributes: 13
Location/Geographic Attributes: 18
Temporal Attributes: 7
Target Variables: 2
Monthly Precipitation: 12
Monthly Max Temperature: 12
Monthly Min Temperature: 12
Monthly Avg Temperature: 12
Monthly Cooling Degree Days: 12
Monthly Heating Degree Days: 12
Engineered Climate Features: 8
================================================================================
Total: 120 columns
```

---

# 4. Detailed Attribute Descriptions

## 4.1 USDA Agricultural Attributes

In [8]:

```python
# Describe USDA agricultural columns
attribute_descriptions = {
    'SOURCE_DESC': 'Data source (e.g., CENSUS, SURVEY)',
    'SECTOR_DESC': 'Agricultural sector (e.g., CROPS, ANIMALS & PRODUCTS)',
    'GROUP_DESC': 'Commodity group (e.g., FIELD CROPS, VEGETABLES, FRUIT & TREE NUTS)',
    'COMMODITY_DESC': 'Specific commodity/crop (e.g., CORN, WHEAT, COTTON)',
    'CLASS_DESC': 'Commodity class/variety',
    'PRODN_PRACTICE_DESC': 'Production practice (e.g., IRRIGATED, NON-IRRIGATED)',
    'UTIL_PRACTICE_DESC': 'Utilization practice (e.g., GRAIN, SILAGE)',
    'STATISTICCAT_DESC': 'Type of statistic (e.g., YIELD, PRODUCTION, AREA HARVESTED)',
    'UNIT_DESC': 'Unit of measurement (e.g., BU/ACRE, TONS, $)',
    'SHORT_DESC': 'Complete description of the statistic',
    'DOMAIN_DESC': 'Domain category',
    'DOMAINCAT_DESC': 'Domain category description',
    'AGG_LEVEL_DESC': 'Geographic aggregation level'
}

print("\nUSDA Agricultural Attribute Descriptions:")
print("="*80)
for col in usda_columns:
    if col in df.columns:
        unique_count = df[col].nunique()
        print(f"\n{col}:")
        print(f"  Description: {attribute_descriptions.get(col, 'N/A')}")
        print(f"  Type: {df[col].dtype}")
        print(f"  Unique values: {unique_count}")
        if unique_count <= 10:
            print(f"  Values: {df[col].unique()[:10].tolist()}")
print("="*80)
```

```
USDA Agricultural Attribute Descriptions:
================================================================================

SOURCE_DESC:
  Description: Data source (e.g., CENSUS, SURVEY)
  Type: object
  Unique values: 2
  Values: ['CENSUS', 'SURVEY']

SECTOR_DESC:
  Description: Agricultural sector (e.g., CROPS, ANIMALS & PRODUCTS)
  Type: object
  Unique values: 1
  Values: ['CROPS']

GROUP_DESC:
  Description: Commodity group (e.g., FIELD CROPS, VEGETABLES, FRUIT & TREE NUTS)
  Type: object
  Unique values: 5
  Values: ['VEGETABLES', 'FRUIT & TREE NUTS', 'FIELD CROPS', 'CROP TOTALS', 'HORTICULTURE']

COMMODITY_DESC:
  Description: Specific commodity/crop (e.g., CORN, WHEAT, COTTON)
  Type: object
  Unique values: 165

CLASS_DESC:
  Description: Commodity class/variety
  Type: object
  Unique values: 85

PRODN_PRACTICE_DESC:
  Description: Production practice (e.g., IRRIGATED, NON-IRRIGATED)
  Type: object
  Unique values: 8
  Values: ['ALL PRODUCTION PRACTICES', 'IRRIGATED', 'NON-IRRIGATED', 'IN THE OPEN', 'UNDER PROTECTION', 'IN THE OPEN,
IRRIGATED', 'PRODUCTION CONTRACT', 'ORGANIC']

UTIL_PRACTICE_DESC:
  Description: Utilization practice (e.g., GRAIN, SILAGE)
  Type: object
  Unique values: 20
```

```
STATISTICCAT_DESC:
  Description: Type of statistic (e.g., YIELD, PRODUCTION, AREA HARVESTED)
  Type: object
  Unique values: 16

UNIT_DESC:
  Description: Unit of measurement (e.g., BU/ACRE, TONS, $)
  Type: object
  Unique values: 21

SHORT_DESC:
  Description: Complete description of the statistic
  Type: object
  Unique values: 1442

DOMAIN_DESC:
  Description: Domain category
  Type: object
  Unique values: 7
  Values: ['TOTAL', 'AREA HARVESTED', 'AREA HARVESTED, FRESH MARKET & PROCESSING', 'NAICS CLASSIFICATION', 'AREA BEAR
ING & NON-BEARING', 'BALES GINNED', 'ORGANIC STATUS']

DOMAINCAT_DESC:
  Description: Domain category description
  Type: object
  Unique values: 38

AGG_LEVEL_DESC:
  Description: Geographic aggregation level
  Type: object
  Unique values: 1
  Values: ['COUNTY']
================================================================================
```

## 4.2 Location and Geographic Attributes

In [9]:
```
# Analyze location columns
print("\nLocation/Geographic Attributes:")
print("="*80)
location_desc = {
```

```
    'STATE_NAME': 'State name',
    'COUNTY_NAME': 'County name',
    'COUNTY_FIPS': 'Federal Information Processing Standard county code',
    'ASD_DESC': 'Agricultural Statistics District',
    'REGION_DESC': 'Geographic region'
}

for col in ['STATE_NAME', 'COUNTY_NAME', 'COUNTY_FIPS', 'ASD_DESC']:
    if col in df.columns:
        print(f"\n{col}:")
        print(f"  Description: {location_desc.get(col, 'Geographic identifier')}")
        print(f"  Unique values: {df[col].nunique()}")
        if col == 'COUNTY_NAME':
            print(f"  Sample counties: {df[col].unique()[:10].tolist()}")
print("="*80)
```

```
Location/Geographic Attributes:
================================================================================

STATE_NAME:
  Description: State name
  Unique values: 1

COUNTY_NAME:
  Description: County name
  Unique values: 256
  Sample counties: ['BELL', 'SHACKELFORD', 'KERR', 'MONTAGUE', 'DAWSON', 'OTHER (COMBINED) COUNTIES', 'HALE', 'TERRY'
, 'WOOD', 'LIMESTONE']

COUNTY_FIPS:
  Description: Federal Information Processing Standard county code
  Unique values: 255

ASD_DESC:
  Description: Agricultural Statistics District
  Unique values: 15
================================================================================
```

## 4.3 Temporal Attributes

In [10]:
```
# Analyze temporal attributes
```

```
print("\nTemporal Attributes:")
print("="*80)
if 'YEAR' in df.columns:
    print(f"\nYEAR:")
    print(f"  Description: Year of observation")
    print(f"  Range: {df['YEAR'].min()} to {df['YEAR'].max()}")
    print(f"  Unique years: {df['YEAR'].nunique()}")
    print(f"  Years covered: {sorted(df['YEAR'].unique())}")
print("="*80)
```

```
Temporal Attributes:
================================================================================

YEAR:
  Description: Year of observation
  Range: 2000 to 2023
  Unique years: 24
  Years covered: [np.int64(2000), np.int64(2001), np.int64(2002), np.int64(2003), np.int64(2004), np.int64(2005), n
p.int64(2006), np.int64(2007), np.int64(2008), np.int64(2009), np.int64(2010), np.int64(2011), np.int64(2012), np.int
64(2013), np.int64(2014), np.int64(2015), np.int64(2016), np.int64(2017), np.int64(2018), np.int64(2019), np.int64(20
20), np.int64(2021), np.int64(2022), np.int64(2023)]
================================================================================
```

## 4.4 Target Variable

In [11]:

```
# Analyze target variable
print("\nTarget Variable:")
print("="*80)
if 'VALUE' in df.columns:
    # Convert VALUE to numeric, handling non-numeric values
    df['VALUE_numeric'] = pd.to_numeric(df['VALUE'], errors='coerce')

    print(f"\nVALUE:")
    print(f"  Description: Agricultural statistic value (varies by STATISTICCAT_DESC and UNIT_DESC)")
    print(f"  Type: {df['VALUE'].dtype}")
    print(f"  Non-null count: {df['VALUE'].notna().sum():,}")
    print(f"  Null count: {df['VALUE'].isna().sum():,}")
    print(f"  Numeric values: {df['VALUE_numeric'].notna().sum():,}")
    print(f"\n  Numeric VALUE Statistics:")
    print(f"    Min: {df['VALUE_numeric'].min()}")
    print(f"    Max: {df['VALUE_numeric'].max()}")
```

```
    print(f"    Mean: {df['VALUE_numeric'].mean():.2f}")
    print(f"    Median: {df['VALUE_numeric'].median():.2f}")
    print(f"    Std Dev: {df['VALUE_numeric'].std():.2f}")

    # Check for non-numeric values
    non_numeric = df[df['VALUE_numeric'].isna() & df['VALUE'].notna()]['VALUE'].unique()
    if len(non_numeric) > 0:
        print(f"\n  Non-numeric values found: {non_numeric[:10]}")
print("="*80)
```

```
Target Variable:
================================================================================

VALUE:
  Description: Agricultural statistic value (varies by STATISTICCAT_DESC and UNIT_DESC)
  Type: object
  Non-null count: 398,204
  Null count: 0
  Numeric values: 210,970

  Numeric VALUE Statistics:
    Min: 0.0
    Max: 999.0
    Mean: 68.16
    Median: 6.00
    Std Dev: 168.92

  Non-numeric values found: ['(D)' '1,919' '102,851,000' '1,500' '5,500' '1,448,000' '40,192' '3,000'
 '22,000' '4,378']
================================================================================
```

## 4.5 Climate Attributes - Monthly Precipitation

In [12]:

```
# Analyze monthly precipitation
print("\nMonthly Precipitation Attributes (inches):")
print("="*80)
precip_stats = pd.DataFrame()
for col in precip_columns:
    if col in df.columns:
        precip_stats[col] = [
            df[col].min(),
```

```
            df[col].max(),
            df[col].mean(),
            df[col].std()
        ]

precip_stats.index = ['Min', 'Max', 'Mean', 'Std Dev']
print(precip_stats.T.round(2))
print("="*80)
```

```
Monthly Precipitation Attributes (inches):
================================================================================
            Min    Max  Mean  Std Dev
PRECIP_JAN  0.0   9.79  2.33     2.12
PRECIP_FEB  0.0  11.45  1.65     1.47
PRECIP_MAR  0.0  11.74  2.99     2.35
PRECIP_APR  0.0  11.88  2.22     1.67
PRECIP_MAY  0.0  23.41  3.52     2.36
PRECIP_JUN  0.0  16.20  3.32     2.48
PRECIP_JUL  0.0  16.30  3.27     2.95
PRECIP_AUG  0.0  46.04  3.91     4.71
PRECIP_SEP  0.0  19.74  2.72     1.99
PRECIP_OCT  0.0  23.18  2.63     2.51
PRECIP_NOV  0.0  17.76  1.88     2.04
PRECIP_DEC  0.0  11.91  2.09     1.98
================================================================================
```

## 4.6 Climate Attributes - Monthly Temperatures

In [13]:

```
# Analyze temperature ranges
print("\nMonthly Temperature Attributes (Fahrenheit):")
print("="*80)

# Sample a few months for detailed display
sample_months = ['JAN', 'APR', 'JUL', 'OCT']
temp_summary = []

for month in sample_months:
    tmax_col = f'TMAX_{month}'
    tmin_col = f'TMIN_{month}'
    tavg_col = f'TAVG_{month}'
```

```
    if all(col in df.columns for col in [tmax_col, tmin_col, tavg_col]):
        temp_summary.append({
            'Month': month,
            'TMAX_Min': df[tmax_col].min(),
            'TMAX_Max': df[tmax_col].max(),
            'TMAX_Mean': df[tmax_col].mean(),
            'TMIN_Min': df[tmin_col].min(),
            'TMIN_Max': df[tmin_col].max(),
            'TMIN_Mean': df[tmin_col].mean(),
            'TAVG_Mean': df[tavg_col].mean()
        })

temp_df = pd.DataFrame(temp_summary)
print(temp_df.round(1))
print("\n(Showing sample months: January, April, July, October)")
print("="*80)
```

```
Monthly Temperature Attributes (Fahrenheit):
================================================================================
  Month  TMAX_Min  TMAX_Max  TMAX_Mean  TMIN_Min  TMIN_Max  TMIN_Mean  \
0   JAN      38.2      78.8       59.8      17.1      56.8       35.5
1   APR      63.4      95.2       78.9      32.3      70.2       54.3
2   JUL      84.8     106.3       94.3      60.8      80.3       72.0
3   OCT      62.3      92.8       79.0      34.5      71.3       54.2

   TAVG_Mean
0       47.6
1       66.6
2       83.1
3       66.6

(Showing sample months: January, April, July, October)
================================================================================
```

## 4.7 Climate Attributes - Degree Days

In [14]:
```
# Analyze degree days
print("\nDegree Days Attributes:")
print("="*80)
print("\nCooling Degree Days (CDD) - Energy needed for cooling:")
cdd_stats = pd.DataFrame()
```

```python
for col in ['CDD_JAN', 'CDD_APR', 'CDD_JUL', 'CDD_OCT']:
    if col in df.columns:
        cdd_stats[col] = [df[col].min(), df[col].max(), df[col].mean()]
cdd_stats.index = ['Min', 'Max', 'Mean']
print(cdd_stats.T.round(1))

print("\nHeating Degree Days (HDD) - Energy needed for heating:")
hdd_stats = pd.DataFrame()
for col in ['HDD_JAN', 'HDD_APR', 'HDD_JUL', 'HDD_OCT']:
    if col in df.columns:
        hdd_stats[col] = [df[col].min(), df[col].max(), df[col].mean()]
hdd_stats.index = ['Min', 'Max', 'Mean']
print(hdd_stats.T.round(1))
print("="*80)
```

```
Degree Days Attributes:
================================================================================

Cooling Degree Days (CDD) - Energy needed for cooling:
           Min    Max   Mean
CDD_JAN    0.0  174.0   13.2
CDD_APR    0.0  491.0  127.8
CDD_JUL  269.0  864.0  561.6
CDD_OCT    0.0  492.0  128.2

Heating Degree Days (HDD) - Energy needed for heating:
          Min     Max   Mean
HDD_JAN  99.0  1148.0  551.6
HDD_APR   0.0   476.0   79.7
HDD_JUL   0.0     0.0    0.0
HDD_OCT   0.0   477.0   78.1
================================================================================
```

## 4.8 Engineered Climate Features

In [15]:

```python
# Analyze engineered features
print("\nEngineered Climate Features:")
print("="*80)
eng_desc = {
    'GROWING_SEASON_PRECIP': 'Total precipitation April-September (inches)',
    'GROWING_SEASON_TEMP_AVG': 'Average temperature April-September (°F)',
```

```
    'GROWING_SEASON_TEMP_MAX': 'Average max temperature April-September (°F)',
    'GROWING_SEASON_TEMP_MIN': 'Average min temperature April-September (°F)',
    'ANNUAL_PRECIP': 'Total annual precipitation (inches)',
    'ANNUAL_TEMP_AVG': 'Annual average temperature (°F)',
    'ANNUAL_CDD': 'Total annual cooling degree days',
    'ANNUAL_HDD': 'Total annual heating degree days'
}

for col in engineered_columns:
    if col in df.columns:
        print(f"\n{col}:")
        print(f"  Description: {eng_desc.get(col, 'N/A')}")
        print(f"  Range: {df[col].min():.2f} to {df[col].max():.2f}")
        print(f"  Mean: {df[col].mean():.2f}")
        print(f"  Std Dev: {df[col].std():.2f}")
print("="*80)
```

```
Engineered Climate Features:
================================================================================

GROWING_SEASON_PRECIP:
  Description: Total precipitation April-September (inches)
  Range: 1.53 to 71.66
  Mean: 18.95
  Std Dev: 8.94

GROWING_SEASON_TEMP_AVG:
  Description: Average temperature April-September (°F)
  Range: 67.15 to 85.70
  Mean: 77.32
  Std Dev: 3.31

GROWING_SEASON_TEMP_MAX:
  Description: Average max temperature April-September (°F)
  Range: 81.30 to 98.68
  Mean: 88.92
  Std Dev: 3.11

GROWING_SEASON_TEMP_MIN:
  Description: Average min temperature April-September (°F)
  Range: 52.68 to 76.05
  Mean: 65.70
  Std Dev: 4.26

ANNUAL_PRECIP:
  Description: Total annual precipitation (inches)
  Range: 2.97 to 95.03
  Mean: 32.51
  Std Dev: 14.55

ANNUAL_TEMP_AVG:
  Description: Annual average temperature (°F)
  Range: 54.83 to 77.10
  Mean: 66.21
  Std Dev: 4.20

ANNUAL_CDD:
  Description: Total annual cooling degree days
  Range: 923.00 to 4909.00
```

```
  Mean: 2630.56
  Std Dev: 679.48

ANNUAL_HDD:
  Description: Total annual heating degree days
  Range: 340.00 to 4929.00
  Mean: 2158.47
  Std Dev: 908.58
================================================================================
```

---

# 5. Missing Values Analysis

In [16]:
```
# Check for missing values
missing_data = pd.DataFrame({
    'Column': df.columns,
    'Missing_Count': df.isnull().sum().values,
    'Missing_Percentage': (df.isnull().sum().values / len(df) * 100)
})
missing_data = missing_data[missing_data['Missing_Count'] > 0].sort_values(
    'Missing_Percentage', ascending=False
)

print("\nMissing Values Summary:")
print("="*80)
if len(missing_data) > 0:
    print(f"\nColumns with missing values: {len(missing_data)}")
    print(missing_data.to_string(index=False))
else:
    print("No missing values found!")
print("="*80)
```

```
Missing Values Summary:
================================================================================

Columns with missing values: 89
               Column  Missing_Count  Missing_Percentage
          REGION_DESC         398204          100.000000
          WEEK_ENDING         398204          100.000000
                ZIP_5         398204          100.000000
       WATERSHED_DESC         398204          100.000000
  CONGR_DISTRICT_CODE         398204          100.000000
                 CV_%         211970           53.231509
        VALUE_numeric         187234           47.019618
           PRECIP_JAN          14133            3.549186
          COUNTY_ANSI          14133            3.549186
           PRECIP_OCT          14133            3.549186
           PRECIP_MAR          14133            3.549186
           PRECIP_APR          14133            3.549186
           PRECIP_MAY          14133            3.549186
           PRECIP_JUN          14133            3.549186
           PRECIP_JUL          14133            3.549186
           PRECIP_AUG          14133            3.549186
           PRECIP_SEP          14133            3.549186
             TMAX_FEB          14133            3.549186
           PRECIP_NOV          14133            3.549186
           PRECIP_DEC          14133            3.549186
             TMAX_JAN          14133            3.549186
             TMAX_APR          14133            3.549186
             TMAX_MAR          14133            3.549186
             TMAX_MAY          14133            3.549186
           PRECIP_FEB          14133            3.549186
              CDD_JAN          14133            3.549186
             TMAX_JUN          14133            3.549186
             TMAX_AUG          14133            3.549186
             TMAX_JUL          14133            3.549186
             TMAX_OCT          14133            3.549186
             TMAX_NOV          14133            3.549186
             TMAX_DEC          14133            3.549186
             TMAX_SEP          14133            3.549186
             TMIN_FEB          14133            3.549186
             TMIN_MAR          14133            3.549186
             TMIN_APR          14133            3.549186
             TMIN_MAY          14133            3.549186
```

| | | |
|---|---|---|
| TMIN_JUN | 14133 | 3.549186 |
| TMIN_JUL | 14133 | 3.549186 |
| TMIN_AUG | 14133 | 3.549186 |
| TMIN_JAN | 14133 | 3.549186 |
| TMIN_OCT | 14133 | 3.549186 |
| TMIN_NOV | 14133 | 3.549186 |
| TMIN_DEC | 14133 | 3.549186 |
| TAVG_JAN | 14133 | 3.549186 |
| TAVG_FEB | 14133 | 3.549186 |
| TAVG_MAR | 14133 | 3.549186 |
| TAVG_APR | 14133 | 3.549186 |
| TAVG_MAY | 14133 | 3.549186 |
| TAVG_JUN | 14133 | 3.549186 |
| TAVG_JUL | 14133 | 3.549186 |
| TAVG_AUG | 14133 | 3.549186 |
| TAVG_SEP | 14133 | 3.549186 |
| TAVG_OCT | 14133 | 3.549186 |
| TAVG_NOV | 14133 | 3.549186 |
| TAVG_DEC | 14133 | 3.549186 |
| TMIN_SEP | 14133 | 3.549186 |
| HDD_MAY | 14133 | 3.549186 |
| CDD_FEB | 14133 | 3.549186 |
| CDD_APR | 14133 | 3.549186 |
| CDD_MAR | 14133 | 3.549186 |
| CDD_JUN | 14133 | 3.549186 |
| CDD_JUL | 14133 | 3.549186 |
| CDD_AUG | 14133 | 3.549186 |
| CDD_MAY | 14133 | 3.549186 |
| CDD_OCT | 14133 | 3.549186 |
| CDD_NOV | 14133 | 3.549186 |
| CDD_DEC | 14133 | 3.549186 |
| HDD_JAN | 14133 | 3.549186 |
| HDD_FEB | 14133 | 3.549186 |
| HDD_MAR | 14133 | 3.549186 |
| HDD_APR | 14133 | 3.549186 |
| CDD_SEP | 14133 | 3.549186 |
| GROWING_SEASON_PRECIP | 14133 | 3.549186 |
| HDD_JUN | 14133 | 3.549186 |
| HDD_AUG | 14133 | 3.549186 |
| HDD_JUL | 14133 | 3.549186 |
| HDD_OCT | 14133 | 3.549186 |
| HDD_NOV | 14133 | 3.549186 |

```
                HDD_DEC          14133            3.549186
                HDD_SEP          14133            3.549186
          ANNUAL_PRECIP          14133            3.549186
GROWING_SEASON_TEMP_AVG          14133            3.549186
GROWING_SEASON_TEMP_MIN          14133            3.549186
GROWING_SEASON_TEMP_MAX          14133            3.549186
             ANNUAL_CDD          14133            3.549186
        ANNUAL_TEMP_AVG          14133            3.549186
             ANNUAL_HDD          14133            3.549186
               ASD_DESC            261            0.065544
================================================================================
```

---

# 6. Key Categorical Variables

In [17]:

```python
# Analyze key categorical variables
print("\nKey Categorical Variables:")
print("="*80)

# Counties
if 'COUNTY_NAME' in df.columns:
    print(f"\nUnique Counties: {df['COUNTY_NAME'].nunique()}")
    print(f"Top 10 counties by record count:")
    print(df['COUNTY_NAME'].value_counts().head(10))

# Crops
if 'COMMODITY_DESC' in df.columns:
    print(f"\n\nUnique Commodities/Crops: {df['COMMODITY_DESC'].nunique()}")
    print(f"Top 10 commodities by record count:")
    print(df['COMMODITY_DESC'].value_counts().head(10))

# Statistics Types
if 'STATISTICCAT_DESC' in df.columns:
    print(f"\n\nUnique Statistic Types: {df['STATISTICCAT_DESC'].nunique()}")
    print(f"Statistic types:")
    print(df['STATISTICCAT_DESC'].value_counts())

print("="*80)
```

```
Key Categorical Variables:
================================================================================

Unique Counties: 256
Top 10 counties by record count:
COUNTY_NAME
OTHER (COMBINED) COUNTIES    13872
HIDALGO                       3424
LUBBOCK                       3012
MEDINA                        2961
BRAZORIA                      2937
CAMERON                       2892
WHARTON                       2824
WILLIAMSON                    2783
HUNT                          2776
COLLIN                        2763
Name: count, dtype: int64


Unique Commodities/Crops: 165
Top 10 commodities by record count:
COMMODITY_DESC
COTTON              47859
WHEAT               45658
SORGHUM             27754
HAY                 22679
CORN                21495
PECANS              14528
VEGETABLE TOTALS    14198
HAY & HAYLAGE       12663
HAYLAGE             11293
OATS                 8002
Name: count, dtype: int64


Unique Statistic Types: 16
Statistic types:
STATISTICCAT_DESC
AREA HARVESTED                166318
PRODUCTION                     45786
SALES                          32474
AREA BEARING & NON-BEARING     31155
```

```
YIELD                        23473
AREA PLANTED                 23340
AREA BEARING                 20622
AREA NON-BEARING             18228
AREA IN PRODUCTION           16082
GINNED BALES                 10992
AREA GROWN                    4342
CAPACITY                      3452
ACTIVE GINS                    871
AREA NOT HARVESTED             518
OPERATIONS                     494
SUCROSE                         57
Name: count, dtype: int64
================================================================================
```

---

# 7. Data Summary Statistics

In [18]:

```
# Comprehensive numerical summary
print("\nNumerical Attributes Summary:")
print("="*80)
numeric_cols = df.select_dtypes(include=[np.number]).columns.tolist()
print(f"Total numerical columns: {len(numeric_cols)}")
df[numeric_cols].describe().T
```

```
Numerical Attributes Summary:
================================================================================
Total numerical columns: 97
```

Out[18]:

| | count | mean | std | min | 25% | 50% | 75% |
|---|---|---|---|---|---|---|---|
| **STATE_ANSI** | 398204.0 | 48.000000 | 0.000000 | 48.000000 | 48.000000 | 48.000000 | 48.000000 |
| **STATE_FIPS_CODE** | 398204.0 | 48.000000 | 0.000000 | 48.000000 | 48.000000 | 48.000000 | 48.000000 |
| **ASD_CODE** | 398204.0 | 49.134444 | 27.868408 | 11.000000 | 22.000000 | 51.000000 | 81.000000 |
| **COUNTY_ANSI** | 384071.0 | 246.935694 | 147.434581 | 1.000000 | 121.000000 | 233.000000 | 371.000000 |
| **COUNTY_CODE** | 398204.0 | 273.592362 | 200.688343 | 1.000000 | 125.000000 | 245.000000 | 387.000000 |
| **REGION_DESC** | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN |
| **ZIP_5** | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN |
| **WATERSHED_CODE** | 398204.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **WATERSHED_DESC** | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN |
| **CONGR_DISTRICT_CODE** | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN |
| **COUNTRY_CODE** | 398204.0 | 9000.000000 | 0.000000 | 9000.000000 | 9000.000000 | 9000.000000 | 9000.000000 |
| **YEAR** | 398204.0 | 2011.601332 | 7.110870 | 2000.000000 | 2007.000000 | 2012.000000 | 2017.000000 |
| **BEGIN_CODE** | 398204.0 | 0.336506 | 1.748010 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **END_CODE** | 398204.0 | 0.336506 | 1.748010 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **WEEK_ENDING** | 0.0 | NaN | NaN | NaN | NaN | NaN | NaN |
| **COUNTY_FIPS** | 398204.0 | 48273.592362 | 200.688343 | 48001.000000 | 48125.000000 | 48245.000000 | 48387.000000 |
| **PRECIP_JAN** | 384071.0 | 2.329363 | 2.118643 | 0.000000 | 0.630000 | 1.680000 | 3.490000 |
| **PRECIP_FEB** | 384071.0 | 1.648177 | 1.469168 | 0.000000 | 0.490000 | 1.290000 | 2.450000 |
| **PRECIP_MAR** | 384071.0 | 2.986804 | 2.349424 | 0.000000 | 1.080000 | 2.460000 | 4.570000 |
| **PRECIP_APR** | 384071.0 | 2.219726 | 1.669245 | 0.000000 | 0.940000 | 1.890000 | 3.130000 |
| **PRECIP_MAY** | 384071.0 | 3.518074 | 2.355157 | 0.000000 | 1.850000 | 2.990000 | 4.750000 |
| **PRECIP_JUN** | 384071.0 | 3.323892 | 2.480152 | 0.000000 | 1.530000 | 2.780000 | 4.520000 |

| | count | mean | std | min | 25% | 50% | 75% |
|---|---|---|---|---|---|---|---|
| **PRECIP_JUL** | 384071.0 | 3.269744 | 2.954237 | 0.000000 | 1.060000 | 2.410000 | 4.520000 |
| **PRECIP_AUG** | 384071.0 | 3.907348 | 4.712599 | 0.000000 | 1.570000 | 2.740000 | 4.500000 |
| **PRECIP_SEP** | 384071.0 | 2.715322 | 1.993747 | 0.000000 | 1.360000 | 2.210000 | 3.640000 |
| **PRECIP_OCT** | 384071.0 | 2.628202 | 2.509809 | 0.000000 | 0.850000 | 1.980000 | 3.410000 |
| **PRECIP_NOV** | 384071.0 | 1.879319 | 2.044262 | 0.000000 | 0.410000 | 1.120000 | 2.680000 |
| **PRECIP_DEC** | 384071.0 | 2.088385 | 1.982861 | 0.000000 | 0.500000 | 1.450000 | 3.290000 |
| **TMAX_JAN** | 384071.0 | 59.790140 | 6.592459 | 38.200000 | 55.800000 | 59.900000 | 64.200000 |
| **TMAX_FEB** | 384071.0 | 62.866700 | 6.584740 | 38.100000 | 58.300000 | 62.300000 | 67.100000 |
| **TMAX_MAR** | 384071.0 | 72.339708 | 4.797473 | 54.800000 | 69.500000 | 73.000000 | 75.500000 |
| **TMAX_APR** | 384071.0 | 78.900251 | 4.902214 | 63.400000 | 75.700000 | 79.100000 | 82.200000 |
| **TMAX_MAY** | 384071.0 | 85.506729 | 4.202680 | 72.400000 | 82.800000 | 85.500000 | 87.900000 |
| **TMAX_JUN** | 384071.0 | 91.953301 | 3.665832 | 81.100000 | 89.100000 | 92.000000 | 94.600000 |
| **TMAX_JUL** | 384071.0 | 94.268269 | 4.319146 | 84.800000 | 91.000000 | 94.000000 | 97.600000 |
| **TMAX_AUG** | 384071.0 | 94.033791 | 3.132952 | 83.000000 | 92.000000 | 94.200000 | 96.100000 |
| **TMAX_SEP** | 384071.0 | 88.879373 | 2.940199 | 75.100000 | 87.500000 | 89.100000 | 90.500000 |
| **TMAX_OCT** | 384071.0 | 79.047023 | 4.735964 | 62.300000 | 76.000000 | 79.900000 | 82.300000 |
| **TMAX_NOV** | 384071.0 | 68.938123 | 5.565167 | 50.600000 | 65.100000 | 69.400000 | 72.800000 |
| **TMAX_DEC** | 384071.0 | 61.027564 | 6.086587 | 40.200000 | 57.200000 | 60.700000 | 65.200000 |
| **TMIN_JAN** | 384071.0 | 35.462742 | 7.411738 | 17.100000 | 30.500000 | 35.100000 | 40.600000 |
| **TMIN_FEB** | 384071.0 | 38.059913 | 8.482063 | 15.200000 | 32.300000 | 37.100000 | 43.900000 |
| **TMIN_MAR** | 384071.0 | 47.390848 | 8.024373 | 23.700000 | 41.700000 | 48.300000 | 53.400000 |
| **TMIN_APR** | 384071.0 | 54.288962 | 6.743861 | 32.300000 | 49.800000 | 55.200000 | 59.000000 |

| | count | mean | std | min | 25% | 50% | 75% |
|---|---|---|---|---|---|---|---|
| **TMIN_MAY** | 384071.0 | 62.119878 | 5.517217 | 44.600000 | 58.900000 | 63.000000 | 65.800000 |
| **TMIN_JUN** | 384071.0 | 69.238010 | 3.787790 | 55.300000 | 67.500000 | 69.700000 | 71.700000 |
| **TMIN_JUL** | 384071.0 | 71.953154 | 3.459330 | 60.800000 | 70.100000 | 72.400000 | 74.400000 |
| **TMIN_AUG** | 384071.0 | 71.527746 | 3.599886 | 57.600000 | 69.900000 | 72.400000 | 73.900000 |
| **TMIN_SEP** | 384071.0 | 65.079088 | 4.285552 | 49.400000 | 62.900000 | 65.500000 | 67.800000 |
| **TMIN_OCT** | 384071.0 | 54.174450 | 5.439506 | 34.500000 | 51.100000 | 54.000000 | 57.300000 |
| **TMIN_NOV** | 384071.0 | 44.794034 | 6.946607 | 24.200000 | 40.200000 | 45.000000 | 49.700000 |
| **TMIN_DEC** | 384071.0 | 37.213173 | 7.175075 | 16.700000 | 32.500000 | 37.600000 | 42.000000 |
| **TAVG_JAN** | 384071.0 | 47.630627 | 6.673176 | 28.000000 | 43.000000 | 47.500000 | 51.900000 |
| **TAVG_FEB** | 384071.0 | 50.467742 | 7.295234 | 28.300000 | 45.400000 | 49.700000 | 55.500000 |
| **TAVG_MAR** | 384071.0 | 59.868461 | 6.116887 | 42.200000 | 55.500000 | 60.800000 | 64.000000 |
| **TAVG_APR** | 384071.0 | 66.598578 | 5.440260 | 49.100000 | 62.800000 | 67.100000 | 70.300000 |
| **TAVG_MAY** | 384071.0 | 73.818140 | 4.352670 | 59.000000 | 71.100000 | 74.100000 | 76.600000 |
| **TAVG_JUN** | 384071.0 | 80.601499 | 3.182502 | 68.300000 | 78.600000 | 80.400000 | 82.800000 |
| **TAVG_JUL** | 384071.0 | 83.114065 | 3.383755 | 73.600000 | 80.600000 | 83.100000 | 85.500000 |
| **TAVG_AUG** | 384071.0 | 82.783886 | 2.927166 | 71.000000 | 81.100000 | 83.300000 | 84.700000 |
| **TAVG_SEP** | 384071.0 | 76.984091 | 3.334562 | 62.200000 | 75.500000 | 77.400000 | 79.000000 |
| **TAVG_OCT** | 384071.0 | 66.614999 | 4.658557 | 49.600000 | 63.900000 | 66.900000 | 69.600000 |
| **TAVG_NOV** | 384071.0 | 56.871136 | 5.922910 | 37.400000 | 52.900000 | 57.300000 | 60.700000 |
| **TAVG_DEC** | 384071.0 | 49.126560 | 6.395236 | 29.600000 | 45.300000 | 49.100000 | 53.450000 |
| **CDD_JAN** | 384071.0 | 13.182521 | 22.239548 | 0.000000 | 0.000000 | 6.000000 | 16.000000 |
| **CDD_FEB** | 384071.0 | 16.954032 | 31.980921 | 0.000000 | 0.000000 | 5.000000 | 19.000000 |

| | count | mean | std | min | 25% | 50% | 75% |
|---|---|---|---|---|---|---|---|
| **CDD_MAR** | 384071.0 | 70.807023 | 59.400854 | 0.000000 | 21.000000 | 60.000000 | 104.000000 |
| **CDD_APR** | 384071.0 | 127.774258 | 90.050001 | 0.000000 | 55.000000 | 115.000000 | 179.000000 |
| **CDD_MAY** | 384071.0 | 286.313281 | 116.900177 | 17.000000 | 202.000000 | 284.000000 | 359.000000 |
| **CDD_JUN** | 384071.0 | 468.594637 | 94.147183 | 139.000000 | 408.000000 | 463.000000 | 535.000000 |
| **CDD_JUL** | 384071.0 | 561.560446 | 104.890538 | 269.000000 | 484.000000 | 560.000000 | 636.000000 |
| **CDD_AUG** | 384071.0 | 551.406982 | 90.440251 | 198.000000 | 500.000000 | 568.000000 | 611.000000 |
| **CDD_SEP** | 384071.0 | 363.454101 | 93.565537 | 39.000000 | 318.000000 | 374.000000 | 421.000000 |
| **CDD_OCT** | 384071.0 | 128.241148 | 83.068407 | 0.000000 | 71.000000 | 120.000000 | 171.000000 |
| **CDD_NOV** | 384071.0 | 33.587509 | 41.358282 | 0.000000 | 6.000000 | 20.000000 | 45.000000 |
| **CDD_DEC** | 384071.0 | 8.686683 | 19.340051 | 0.000000 | 0.000000 | 0.000000 | 10.000000 |
| **HDD_JAN** | 384071.0 | 551.612514 | 189.549248 | 99.000000 | 418.000000 | 547.000000 | 680.000000 |
| **HDD_FEB** | 384071.0 | 424.013893 | 179.853704 | 21.000000 | 284.000000 | 431.000000 | 549.000000 |
| **HDD_MAR** | 384071.0 | 229.891205 | 136.850272 | 11.000000 | 133.000000 | 191.000000 | 315.000000 |
| **HDD_APR** | 384071.0 | 79.723429 | 83.083749 | 0.000000 | 19.000000 | 50.000000 | 119.000000 |
| **HDD_MAY** | 384071.0 | 12.491977 | 24.019026 | 0.000000 | 0.000000 | 0.000000 | 14.000000 |
| **HDD_JUN** | 384071.0 | 0.452885 | 2.471190 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **HDD_JUL** | 384071.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **HDD_AUG** | 384071.0 | 0.027347 | 0.461584 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **HDD_SEP** | 384071.0 | 3.543553 | 8.293623 | 0.000000 | 0.000000 | 0.000000 | 5.000000 |
| **HDD_OCT** | 384071.0 | 78.098724 | 68.371846 | 0.000000 | 29.000000 | 60.000000 | 104.000000 |
| **HDD_NOV** | 384071.0 | 277.654038 | 144.020429 | 12.000000 | 172.000000 | 249.000000 | 370.000000 |
| **HDD_DEC** | 384071.0 | 500.962536 | 185.092887 | 27.000000 | 368.000000 | 493.000000 | 612.000000 |

| | count | mean | std | min | 25% | 50% | 75% |
|---|---|---|---|---|---|---|---|
| **GROWING_SEASON_PRECIP** | 384071.0 | 18.954105 | 8.938067 | 1.530000 | 12.660000 | 17.020000 | 23.690000 |
| **GROWING_SEASON_TEMP_AVG** | 384071.0 | 77.316710 | 3.309409 | 67.150000 | 75.483333 | 77.600000 | 79.583333 |
| **GROWING_SEASON_TEMP_MAX** | 384071.0 | 88.923619 | 3.107366 | 81.300000 | 86.633333 | 88.833333 | 91.100000 |
| **GROWING_SEASON_TEMP_MIN** | 384071.0 | 65.701140 | 4.260499 | 52.683333 | 63.650000 | 66.466667 | 68.400000 |
| **ANNUAL_PRECIP** | 384071.0 | 32.514355 | 14.550911 | 2.970000 | 20.560000 | 30.550000 | 43.280000 |
| **ANNUAL_TEMP_AVG** | 384071.0 | 66.206649 | 4.201270 | 54.833333 | 63.800000 | 66.466667 | 68.841667 |
| **ANNUAL_CDD** | 384071.0 | 2630.562620 | 679.475595 | 923.000000 | 2205.000000 | 2612.000000 | 3040.000000 |
| **ANNUAL_HDD** | 384071.0 | 2158.472100 | 908.579405 | 340.000000 | 1488.000000 | 2067.000000 | 2684.000000 |
| **VALUE_numeric** | 210970.0 | 68.155548 | 168.917532 | 0.000000 | 2.000000 | 6.000000 | 32.000000 |

# 8. Dataset Summary Report

## Complete Dataset Overview

In [19]:

```
# Generate comprehensive summary report
print("\n" + "="*80)
print("COMPREHENSIVE DATASET SUMMARY REPORT")
print("="*80)
print(f"\n1. DATASET SIZE")
print(f"   - Total instances: {df.shape[0]:,}")
print(f"   - Total attributes: {df.shape[1]:,}")
print(f"   - Total data points: {df.shape[0] * df.shape[1]:,}")
print(f"   - Memory usage: {df.memory_usage(deep=True).sum() / (1024**2):.2f} MB")

print(f"\n2. ATTRIBUTE BREAKDOWN")
print(f"   - USDA Agricultural: {len([c for c in usda_columns if c in df.columns])}")
print(f"   - Location/Geographic: {len([c for c in location_columns if c in df.columns])}")
print(f"   - Temporal: {len([c for c in temporal_columns if c in df.columns])}")
print(f"   - Climate (Monthly): {len([c for c in precip_columns + tmax_columns + tmin_columns + tavg_columns + cdd_cc
```

```
print(f"   - Engineered Features: {len([c for c in engineered_columns if c in df.columns])}")
print(f"   - Target Variables: {len([c for c in target_columns if c in df.columns])}")

if 'YEAR' in df.columns:
    print(f"\n3. TEMPORAL COVERAGE")
    print(f"   - Years: {df['YEAR'].min()} to {df['YEAR'].max()}")
    print(f"   - Total years: {df['YEAR'].nunique()}")

if 'COUNTY_NAME' in df.columns:
    print(f"\n4. GEOGRAPHIC COVERAGE")
    print(f"   - Unique counties: {df['COUNTY_NAME'].nunique()}")

if 'COMMODITY_DESC' in df.columns:
    print(f"\n5. AGRICULTURAL COVERAGE")
    print(f"   - Unique commodities/crops: {df['COMMODITY_DESC'].nunique()}")

if 'STATISTICCAT_DESC' in df.columns:
    print(f"   - Statistic types: {df['STATISTICCAT_DESC'].nunique()}")

print(f"\n6. DATA QUALITY")
total_missing = df.isnull().sum().sum()
total_cells = df.shape[0] * df.shape[1]
print(f"   - Total missing values: {total_missing:,}")
print(f"   - Missing percentage: {(total_missing/total_cells)*100:.2f}%")
print(f"   - Columns with missing values: {len(missing_data)}")

print("\n" + "="*80)
print("END OF SUMMARY REPORT")
print("="*80)
```

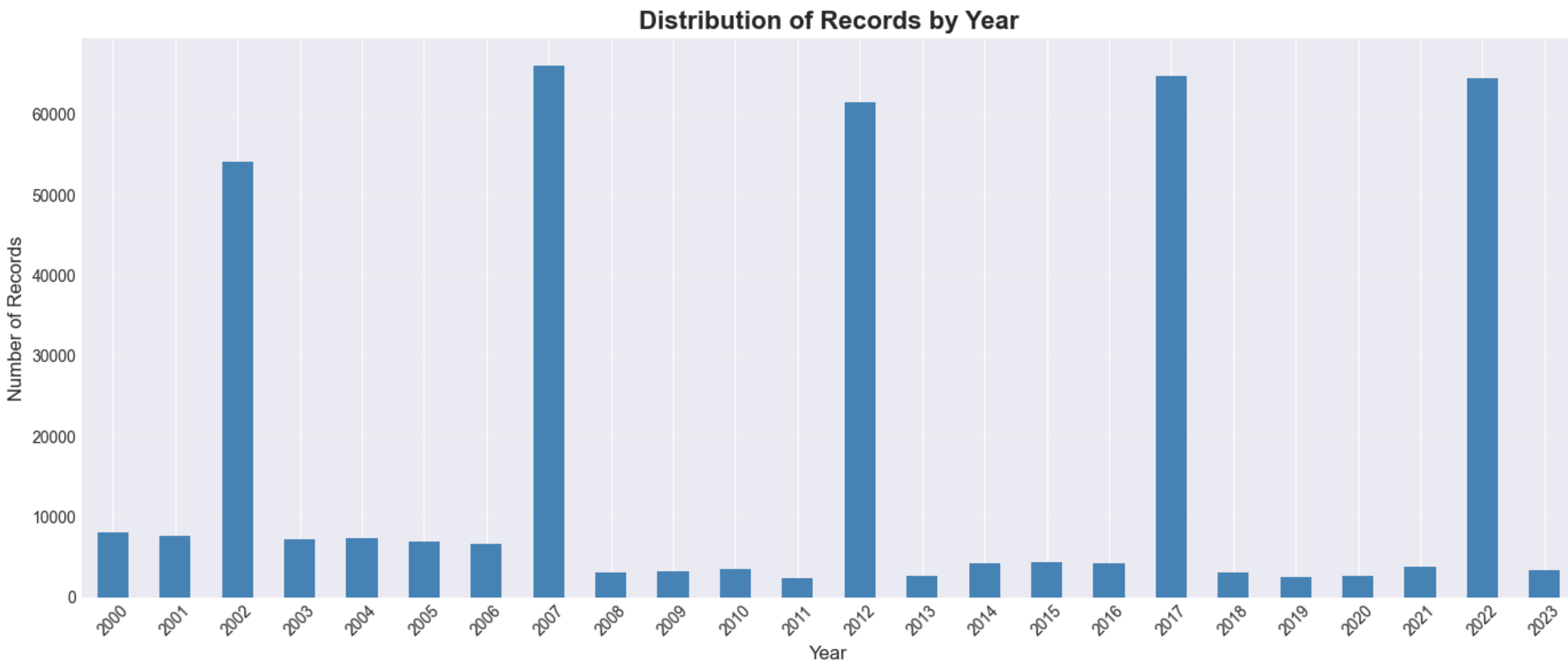
```
================================================================================
COMPREHENSIVE DATASET SUMMARY REPORT
================================================================================

1. DATASET SIZE
   - Total instances: 398,204
   - Total attributes: 121
   - Total data points: 48,182,684
   - Memory usage: 845.91 MB

2. ATTRIBUTE BREAKDOWN
   - USDA Agricultural: 13
   - Location/Geographic: 18
   - Temporal: 7
   - Climate (Monthly): 72
   - Engineered Features: 8
   - Target Variables: 2

3. TEMPORAL COVERAGE
   - Years: 2000 to 2023
   - Total years: 24

4. GEOGRAPHIC COVERAGE
   - Unique counties: 256

5. AGRICULTURAL COVERAGE
   - Unique commodities/crops: 165
   - Statistic types: 16

6. DATA QUALITY
   - Total missing values: 3,535,258
   - Missing percentage: 7.34%
   - Columns with missing values: 89

================================================================================
END OF SUMMARY REPORT
================================================================================
```

---

# 9. Data Visualization (Optional)

## 9.1 Distribution of Records by Year

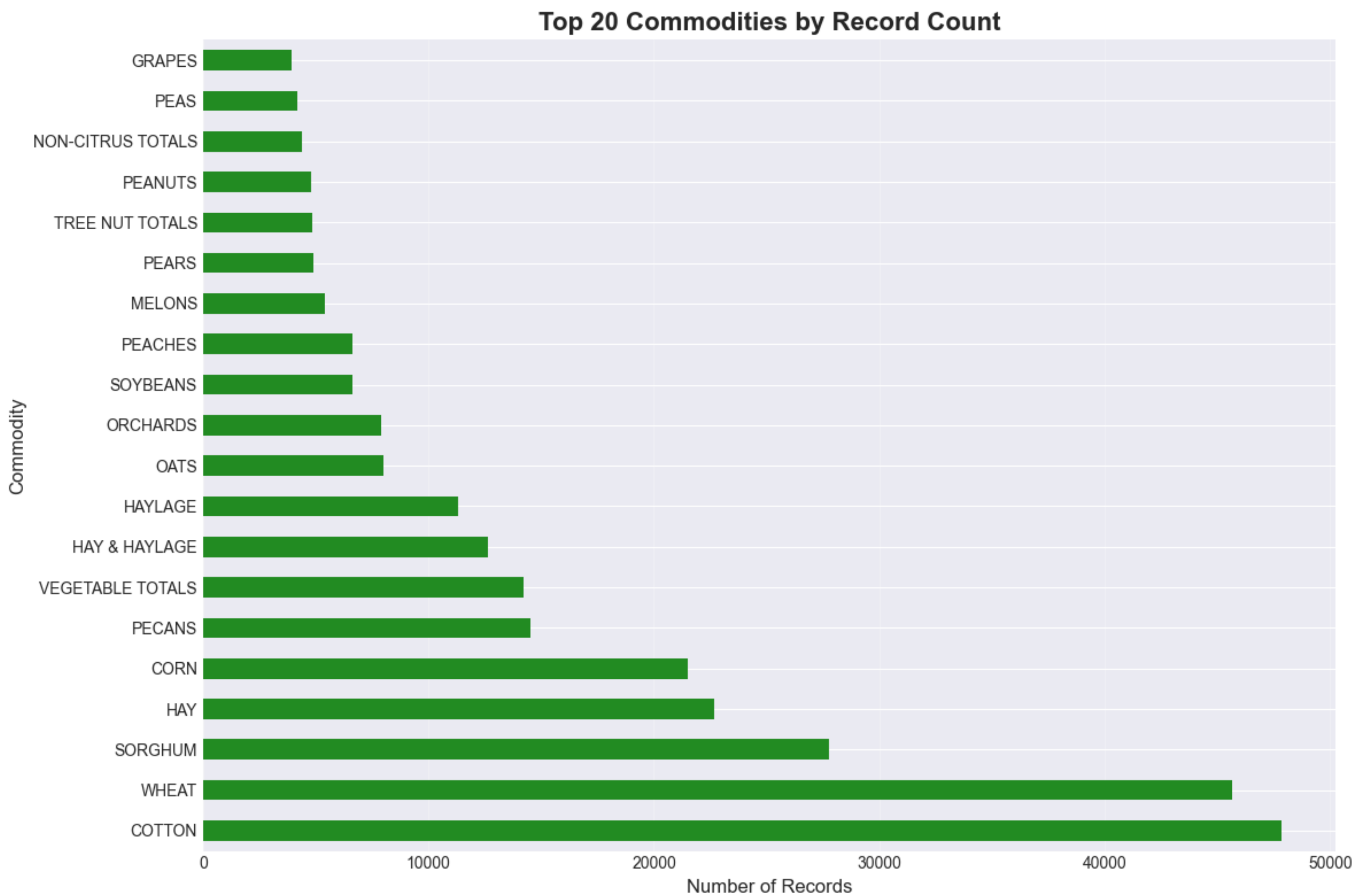
In [20]:
```
# Plot records by year
if 'YEAR' in df.columns:
    plt.figure(figsize=(14, 6))
    df['YEAR'].value_counts().sort_index().plot(kind='bar', color='steelblue')
    plt.title('Distribution of Records by Year', fontsize=16, fontweight='bold')
    plt.xlabel('Year', fontsize=12)
    plt.ylabel('Number of Records', fontsize=12)
    plt.xticks(rotation=45)
    plt.grid(axis='y', alpha=0.3)
    plt.tight_layout()
    plt.show()
```




## 9.2 Top Commodities

In [21]:
```
# Plot top commodities
```

```
if 'COMMODITY_DESC' in df.columns:
    plt.figure(figsize=(12, 8))
    df['COMMODITY_DESC'].value_counts().head(20).plot(kind='barh', color='forestgreen')
    plt.title('Top 20 Commodities by Record Count', fontsize=16, fontweight='bold')
    plt.xlabel('Number of Records', fontsize=12)
    plt.ylabel('Commodity', fontsize=12)
    plt.grid(axis='x', alpha=0.3)
    plt.tight_layout()
    plt.show()
```

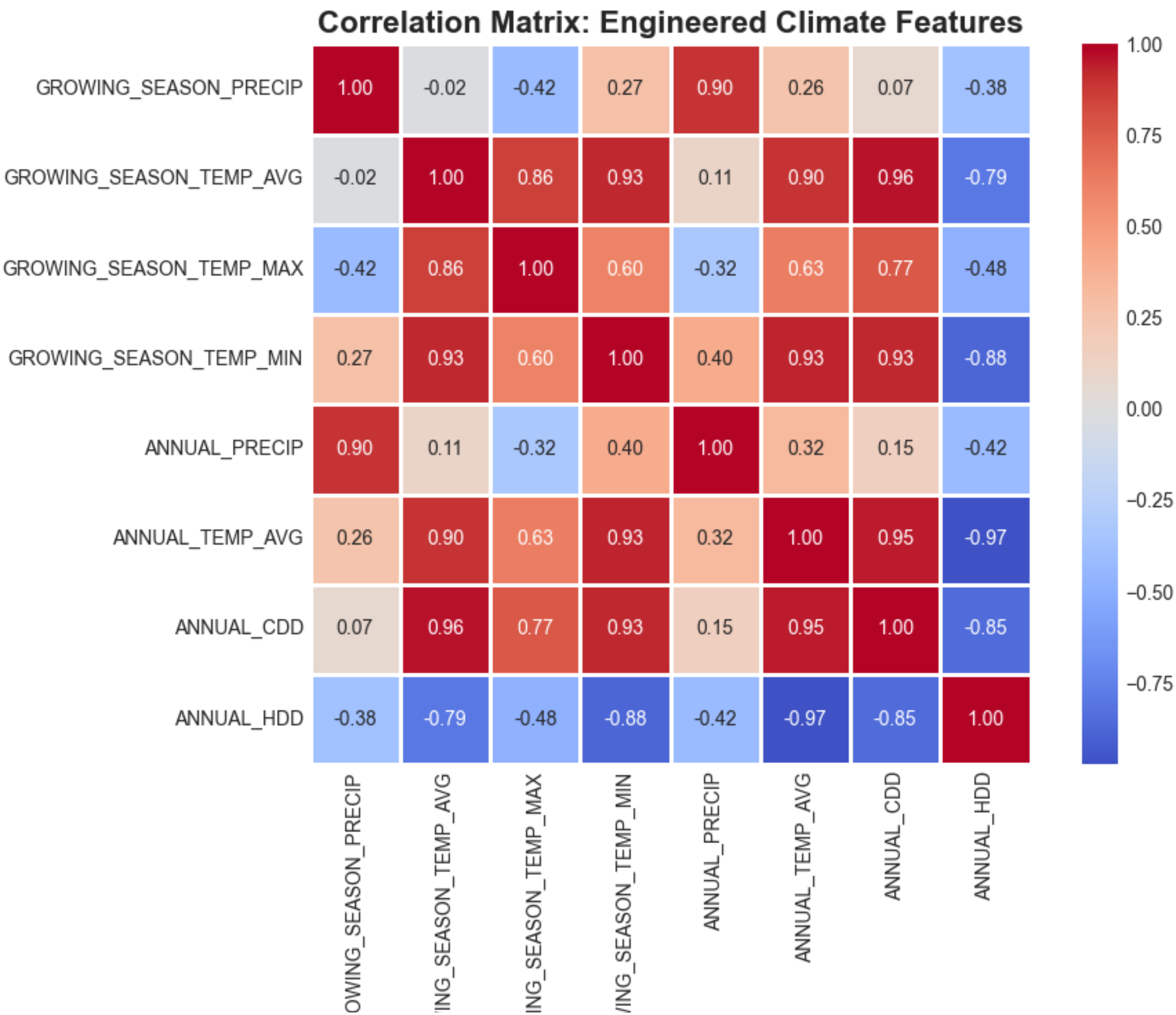


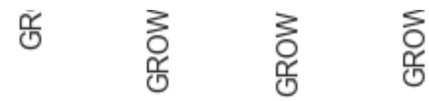

## 9.3 Climate Feature Correlation Heatmap

In [22]:

```
# Plot correlation heatmap for engineered features
climate_features = [col for col in engineered_columns if col in df.columns]
if len(climate_features) > 0:
    plt.figure(figsize=(10, 8))
    correlation_matrix = df[climate_features].corr()
```

```
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm',
            center=0, square=True, linewidths=1)
plt.title('Correlation Matrix: Engineered Climate Features', fontsize=16, fontweight='bold')
plt.tight_layout()
plt.show()
```


Correlation Matrix: Engineered Climate Features
GROWING_SEASON_PRECIP
GROWING_SEASON_TEMP_AVG
GROWING_SEASON_TEMP_MAX
GROWING_SEASON_TEMP_MIN
ANNUAL_PRECIP
ANNUAL_TEMP_AVG
ANNUAL_CDD
ANNUAL_HDD
1.00 -0.02 -0.42 0.27 0.90 0.26 0.07 -0.38
-0.02 1.00 0.86 0.93 0.11 0.90 0.96 -0.79
-0.42 0.86 1.00 0.60 -0.32 0.63 0.77 -0.48
0.27 0.93 0.60 1.00 0.40 0.93 0.93 -0.88
0.90 0.11 -0.32 0.40 1.00 0.32 0.15 -0.42
0.26 0.90 0.63 0.93 0.32 1.00 0.95 -0.97
0.07 0.96 0.77 0.93 0.15 0.95 1.00 -0.85
-0.38 -0.79 -0.48 -0.88 -0.42 -0.97 -0.85 1.00
OWING_SEASON_PRECIP
ING_SEASON_TEMP_AVG
ING_SEASON_TEMP_MAX
ING_SEASON_TEMP_MIN
ANNUAL_PRECIP
ANNUAL_TEMP_AVG
ANNUAL_CDD
ANNUAL_HDD
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75




# 10. Conclusions

## Key Findings:

1. **Dataset Size**: The dataset contains approximately 398,000 records with 120+ attributes, well exceeding the 10M data point requirement (47M+ data points).

2. **Temporal Coverage**: Data spans from 2000-2023 (or as shown in analysis above), providing substantial temporal variation for time-series analysis.

3. **Geographic Coverage**: Covers 255 Texas counties, enabling county-level predictions.

4. **Agricultural Diversity**: Contains 165 different crop types with 16 different statistical measures (yield, production, acres harvested, etc.).

5. **Climate Variables**:

    - 72 monthly climate measurements (precipitation, temperature, degree days)
    - 8 engineered seasonal/annual aggregates
    - Complete coverage of growing season (April-September) metrics
6. **Data Quality**: Analysis of missing values and data types shows areas requiring preprocessing.

## Next Steps:

1. Handle missing values and non-numeric VALUE entries
2. Encode categorical variables
3. Feature selection and engineering
4. Implement baseline models (Decision Tree Regression)
5. Apply AdaBoost for improvement

6. Conduct PCA for dimensionality reduction