Project Title: Crop Production Analysis in India

INTRODUCTION

The Crop Production Analysis in India project aims to leverage data science techniques to analyze and predict crop production trends in India. The agriculture domain plays a crucial role in the overall supply chain, and advancements in technology, particularly in the realm of the Future Internet, are expected to significantly impact this sector. This project focuses on developing a Business-to-Business collaboration platform within the agri-food sector to enhance collaboration among stakeholders.

Import necessary libraries

In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

Loading Dataset

In [2]:
```
crop_data = pd.read_csv('C:/Users/stati/OneDrive/Desktop/Crop Production Analysis in India.csv')

crop_data.shape
```

Out[2]:
```
(246091, 7)
```

EDA (Exploratory Data Analysis)

In [3]:
```
crop_data.head()
```

Out[3]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0

In [4]:
```
crop_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246091 entries, 0 to 246090
Data columns (total 7 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   State_Name     246091 non-null  object 
 1   District_Name  246091 non-null  object 
 2   Crop_Year      246091 non-null  int64  
 3   Season         246091 non-null  object 
 4   Crop           246091 non-null  object 
 5   Area           246091 non-null  float64
 6   Production     242361 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.1+ MB
```

In [5]:
```
crop_data.isnull().sum()
```

Out[5]:
```
State_Name          0
District_Name       0
Crop_Year           0
Season              0
Crop                0
Area                0
Production       3730
dtype: int64
```

In [6]:
```
data = crop_data.dropna()

data.shape
```

Out[6]:
```
(242361, 7)
```

In [7]:
```
data.columns
```

Out[7]:
```
Index(['State_Name', 'District_Name', 'Crop_Year', 'Season', 'Crop', 'Area',
       'Production'],
      dtype='object')
```

In [8]:
```
data.nunique()
```

Out[8]:
```
State_Name          33
District_Name      646
Crop_Year           19
Season               6
Crop               124
Area             38391
Production       51627
dtype: int64
```

In [9]:
```
sum_maxp = data["Production"].sum()

data["percent_of_production"] = data["Production"].map(lambda x:(x/sum_maxp)*100)
```

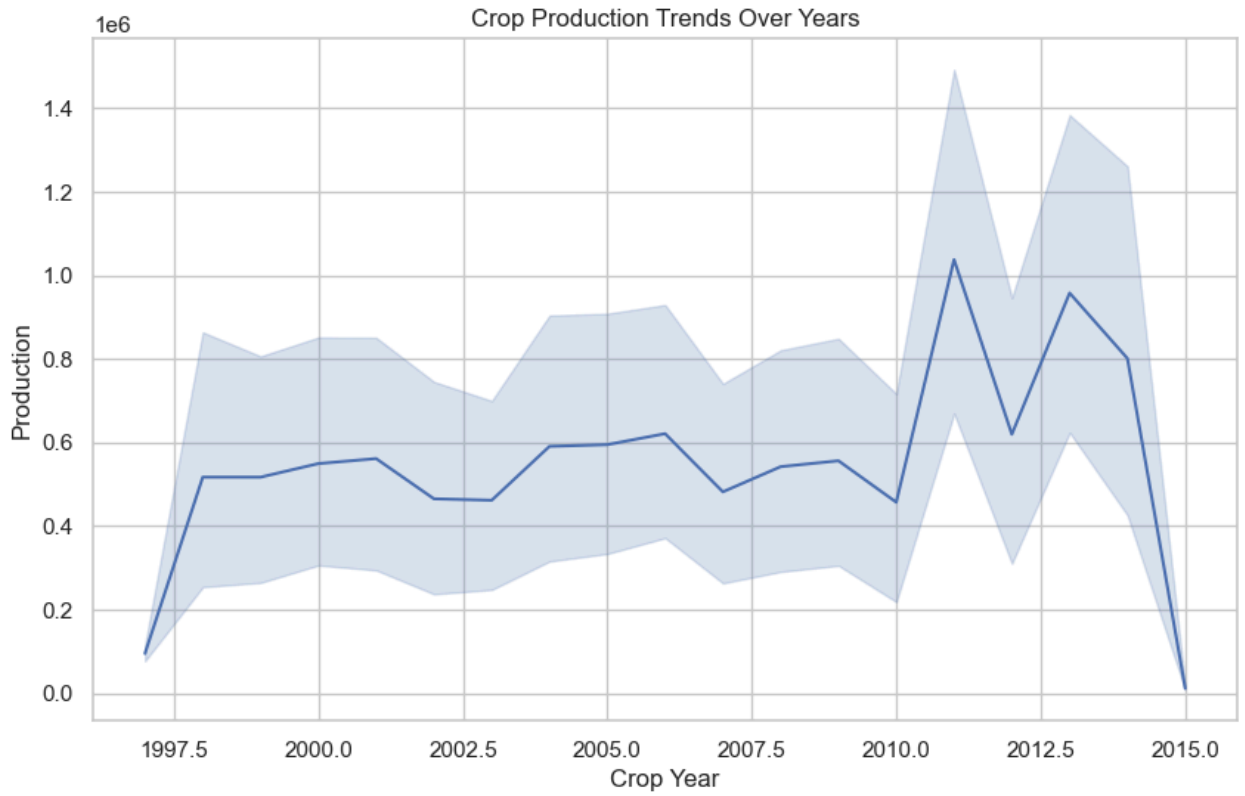
Visualization

Crop Production Trends Over Years

In [10]:
```
# Import visualization libraries

import matplotlib.pyplot as plt
import seaborn as sns
```

In [11]:
```
# Set the style for seaborn

sns.set(style="whitegrid")
```

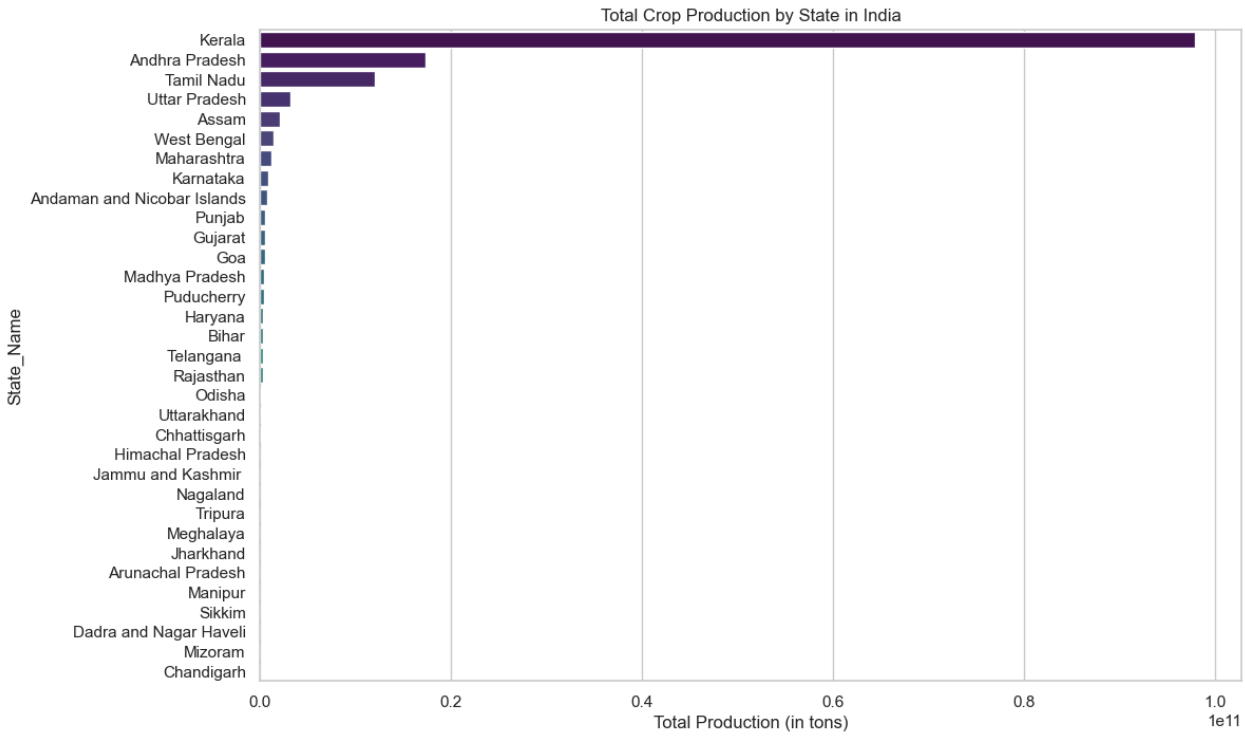
In [12]:
```
# Visualizing production trends over crop years using a line plot

plt.figure(figsize=(10, 6))
sns.lineplot(x=data["Crop_Year"], y=data["Production"])
plt.title("Crop Production Trends Over Years")
plt.xlabel("Crop Year")
plt.ylabel("Production")
plt.show()
```



Total Crop Production by State in India

In [13]:
```
# Grouping by 'State_Name' and summing 'Production' for each state

state_production = data.groupby('State_Name')['Production'].sum().sort_values(ascending=False)
```

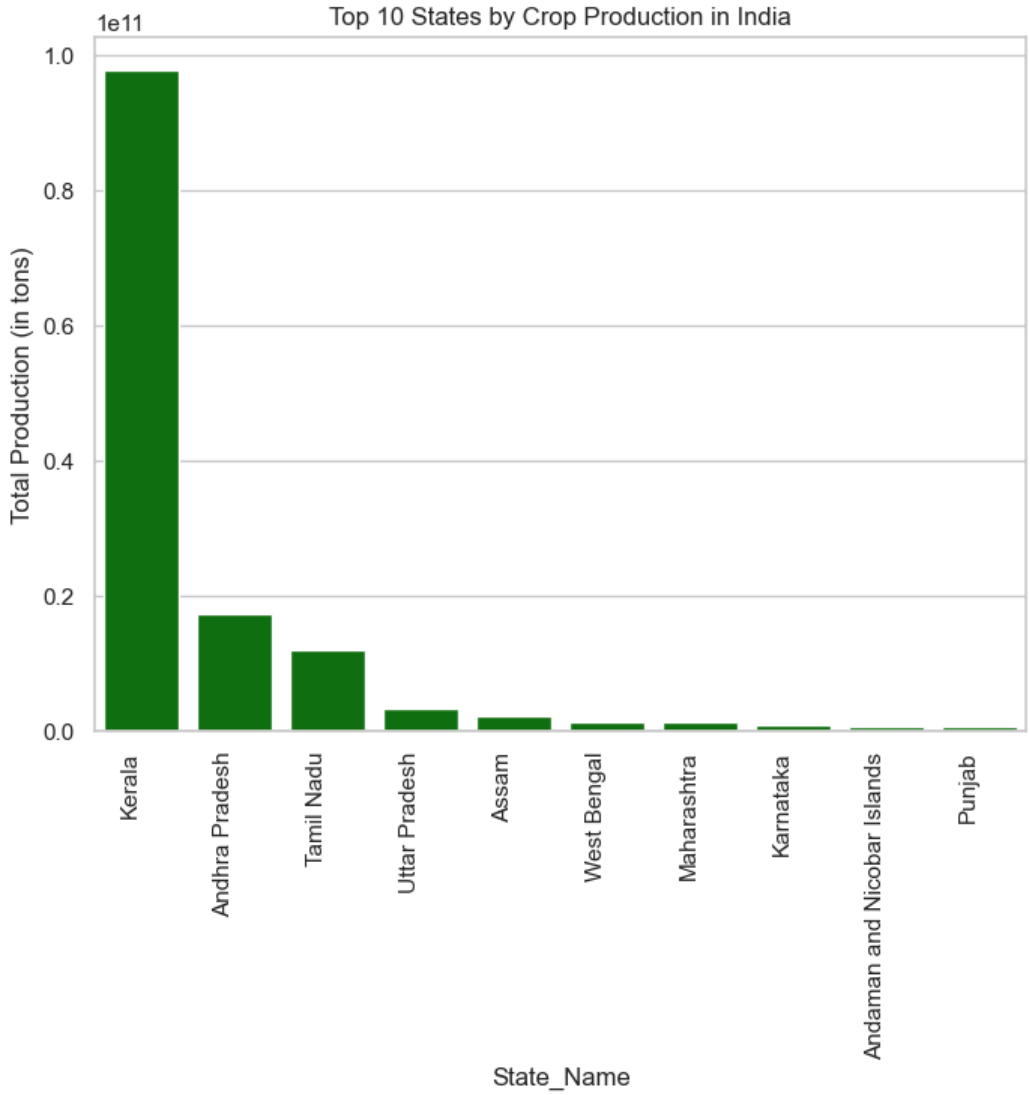
In [14]:
```
# Plotting the bar chart

plt.figure(figsize=(12, 8))
sns.barplot(x=state_production.values, y=state_production.index, palette='viridis')
plt.xlabel('Total Production (in tons)')
plt.title('Total Crop Production by State in India')
plt.show()
```



Top 10 States by Crop Production in India

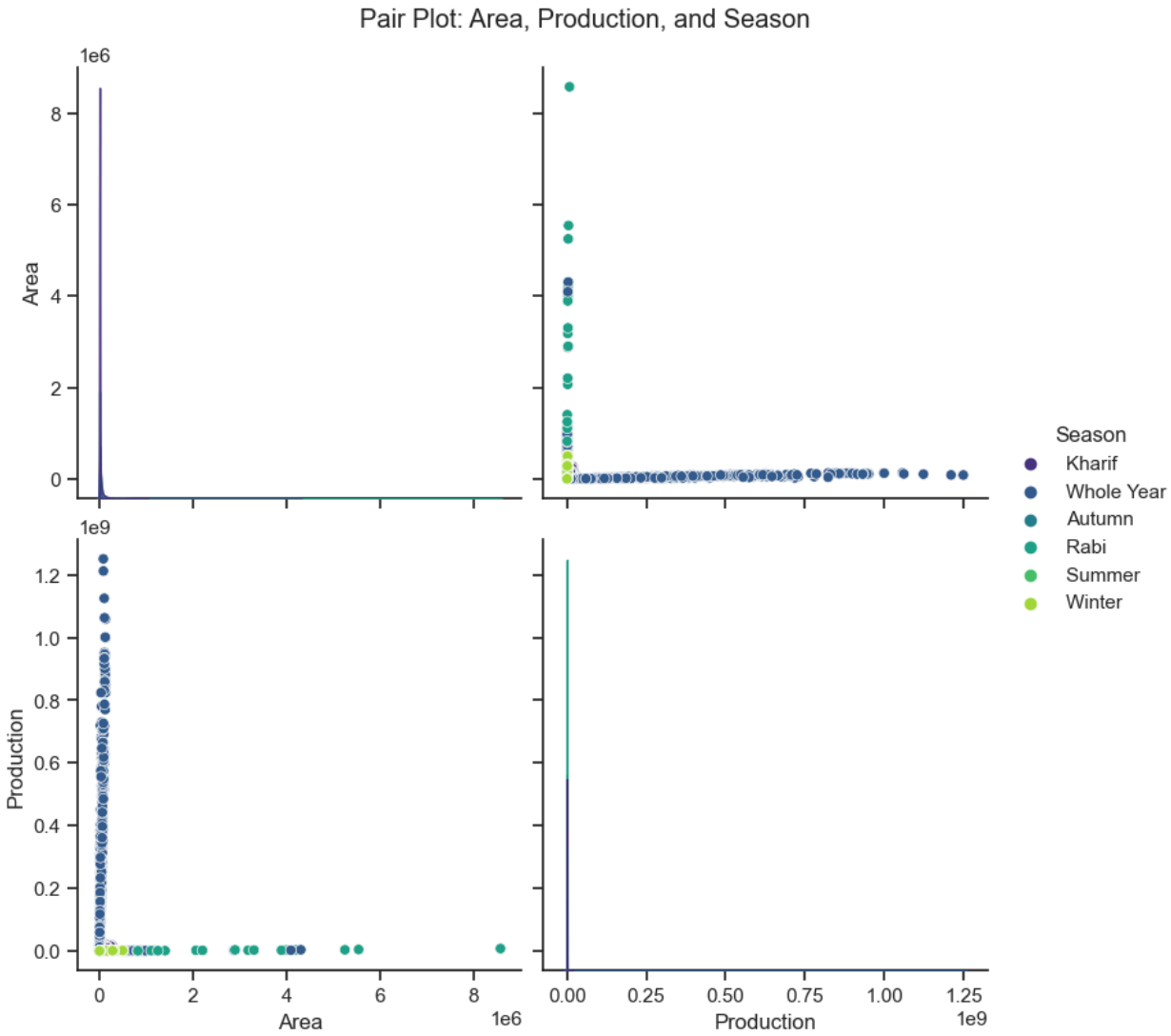
In [15]:
```
# Plotting the bar chart for the top 10 states
```

```
top_states_production = state_production.head(10)
plt.figure(figsize=(8, 6))
sns.barplot(x=top_states_production.index, y=top_states_production.values, color='green', orient='v')
plt.ylabel('Total Production (in tons)')
plt.title('Top 10 States by Crop Production in India')
plt.xticks(rotation=90, ha='right')
plt.show()
```



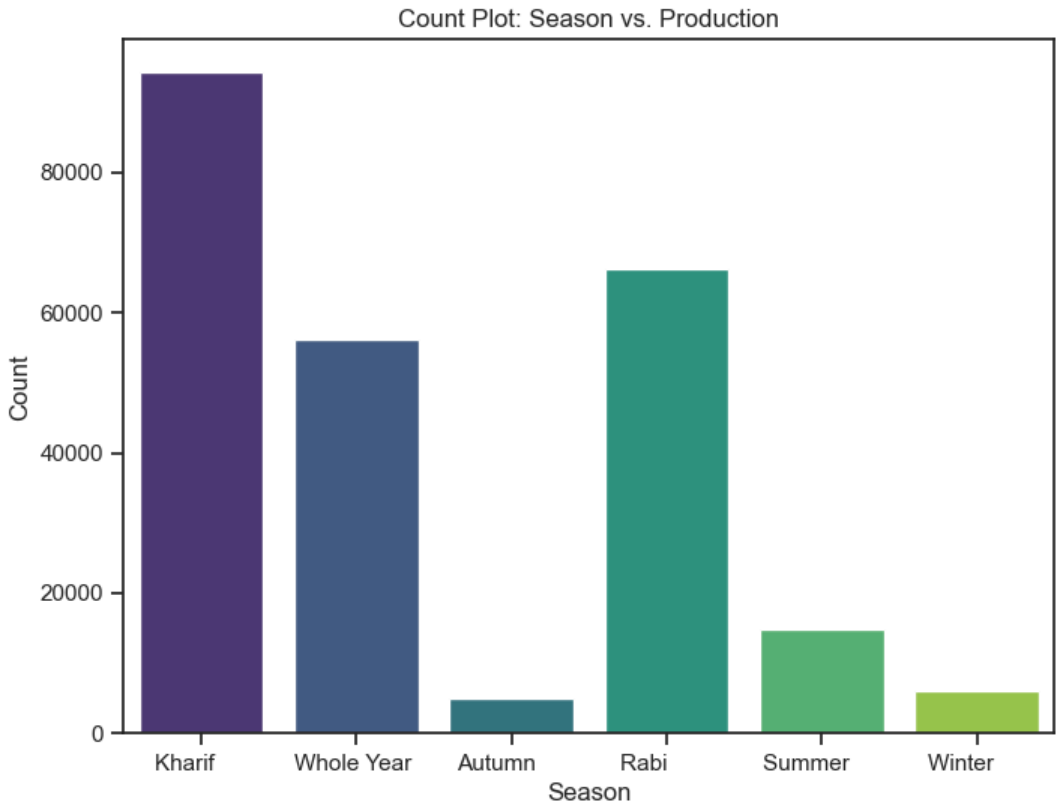
Pairplot: Area vs. Production

In [16]:
```
# Pair plot for selected variables

sns.set(style="ticks")
plot = sns.pairplot(data[['Area', 'Production', 'Season']], hue='Season', palette='viridis', height=4)
plot.fig.suptitle('Pair Plot: Area, Production, and Season', y=1.02)
plt.show()
```



Seasonal Crop Production Analysis

In [17]:
```
# Count plot for 'Season' vs. 'Production' with a different color palette

plt.figure(figsize=(8, 6))
sns.countplot(x="Season", data=data, palette='viridis')
plt.title('Count Plot: Season vs. Production')
plt.xlabel('Season')
plt.ylabel('Count')
plt.show()


# Grouping by 'Season' and summing 'Production'
data.groupby("Season")["Production"].sum().reset_index()
```



Out[17]:

	Season	Production
0	Autumn	6.441377e+07
1	Kharif	4.029970e+09
2	Rabi	2.051688e+09
3	Summer	1.706579e+08
4	Whole Year	1.344248e+11
5	Winter	4.345498e+08

Top Crops by Production

In [18]:
```
# Displaying the top 5 crops by production

data["Crop"].value_counts()[:5]
```

Out[18]:
```
Crop
Rice                15082
Maize               13787
Moong(Green Gram)   10106
Urad                 9710
Sesamum              8821
Name: count, dtype: int64
```

In [19]:
```
# Grouping by 'Crop' and summing 'Production'

top_crop_pro = data.groupby("Crop")["Production"].sum().reset_index().sort_values(by='Production', ascending=False)
top_crop_pro[:5]
```

Out[19]:

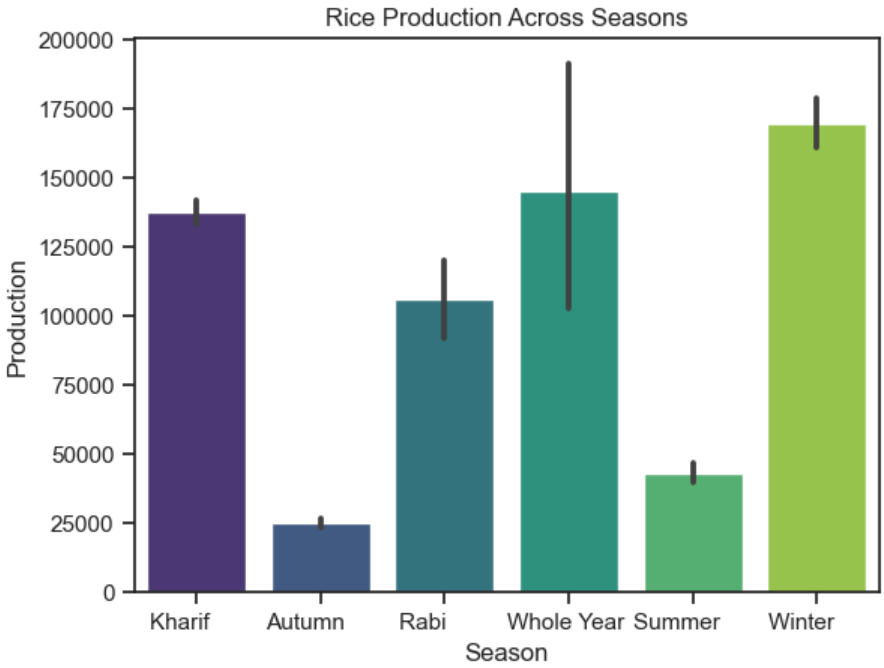
	Crop	Production
28	Coconut	1.299816e+11
106	Sugarcane	5.535682e+09
95	Rice	1.605470e+09
119	Wheat	1.332826e+09
87	Potato	4.248263e+08

Rice Production Analysis

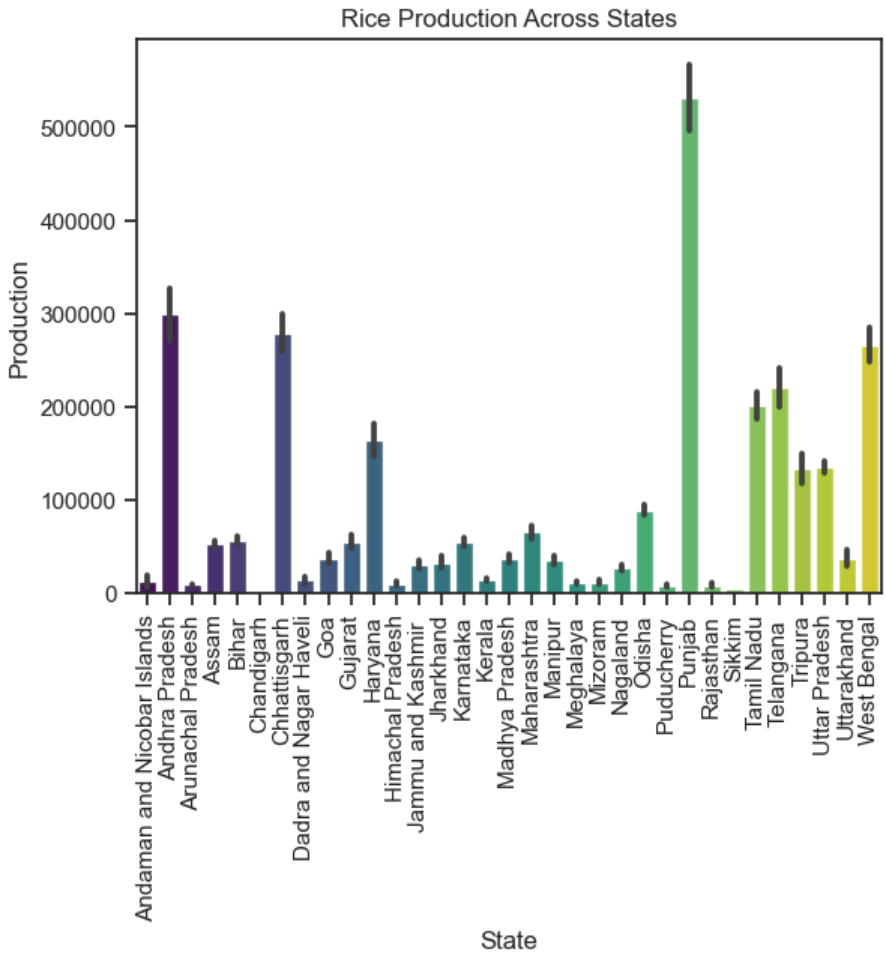
In [20]:
```
# Subset for Rice, Coconut, and Sugarcane production analysis
```

```
rice_df = data[data["Crop"] == "Rice"]
```

In [21]:
```
# Barplot for 'Season' vs. 'Production' for Rice

sns.barplot(x="Season", y="Production", data=rice_df, palette='viridis')
plt.title("Rice Production Across Seasons")
plt.xlabel("Season")
plt.ylabel("Production")
plt.show()
```



In [22]:
```
# Barplot for 'State_Name' vs. 'Production' for Rice

sns.barplot(x="State_Name", y="Production", data=rice_df, palette='viridis')
plt.title("Rice Production Across States")
plt.xlabel("State")
plt.ylabel("Production")
plt.xticks(rotation=90)
plt.show()
```

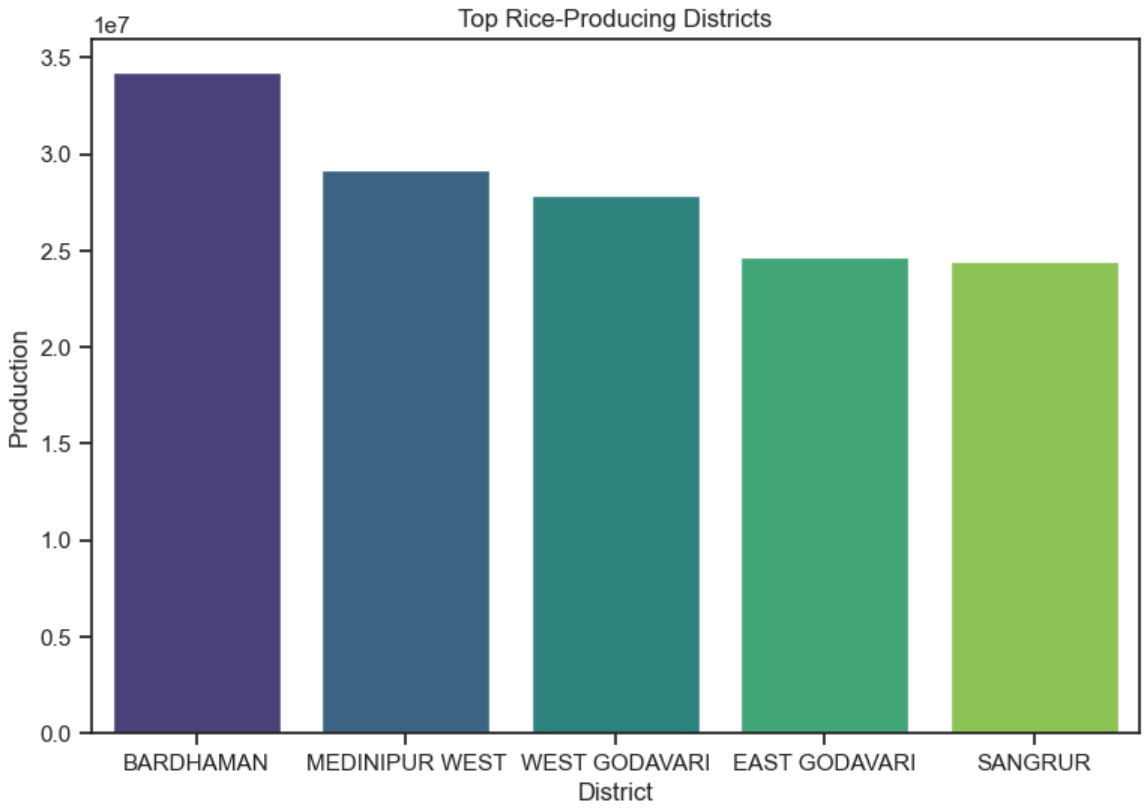


Top Rice Producing Districts

In [23]:
```
# Top rice-producing districts

top_rice_pro_dis = rice_df.groupby("District_Name")["Production"].sum().nlargest(5).reset_index()
```

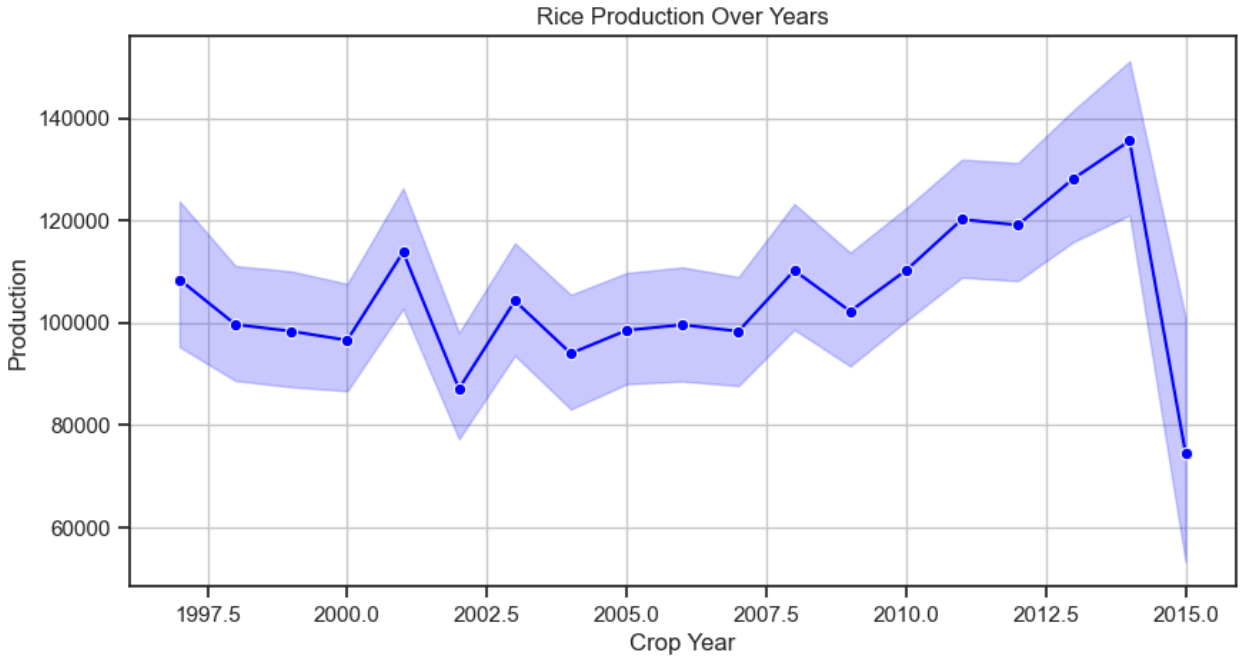
In [24]:
```
# Calculate the percentage of production for each district

top_rice_pro_dis["percent_of_pro"] = top_rice_pro_dis["Production"] / top_rice_pro_dis["Production"].sum() * 100
```

In [25]:
```
# Barplot for top rice-producing districts

plt.figure(figsize=(9, 6))
sns.barplot(x="District_Name", y="Production", data=top_rice_pro_dis, palette='viridis')
plt.title("Top Rice-Producing Districts")
plt.xlabel("District")
plt.ylabel("Production")
plt.show()
```



Rice Production Trends Over Years

In [26]:
```
# Line plot for 'Crop_Year' vs. 'Production' for Rice

plt.figure(figsize=(10, 5))
sns.lineplot(x="Crop_Year", y="Production", data=rice_df, marker='o', color='blue')
plt.title("Rice Production Over Years")
plt.xlabel("Crop Year")
plt.ylabel("Production")
plt.grid(True)
```



Data Preprocessing for Modeling

In [27]:
```
# Dropping unnecessary columns for modeling
```

```
data1 = data.drop(["District_Name", "Crop_Year"], axis=1)
```

In [28]:
```
# Creating dummy variables

data_dum = pd.get_dummies(data1)
data_dum[:5]
```

Out[28]:

	Area	Production	percent_of_production	State_Name_Andaman and Nicobar Islands	State_Name_Andhra Pradesh	State_Name_Arunachal Pradesh	State_Name_Assam	Stat
0	1254.0	2000.0	1.416670e-06	True	False	False	False	
1	2.0	1.0	7.083351e-10	True	False	False	False	
2	102.0	321.0	2.273756e-07	True	False	False	False	
3	176.0	641.0	4.540428e-07	True	False	False	False	
4	720.0	165.0	1.168753e-07	True	False	False	False	

5 rows × 166 columns

Modeling: Decision Tree Regressor

In [29]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [30]:
```
# Splitting the data into training and testing sets

from sklearn.model_selection import train_test_split

x = data_dum.drop("Production", axis=1)
y = data_dum[["Production"]]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

In [31]:
```
# Fitting the Decision Tree Regressor model

regressor = DecisionTreeRegressor(random_state=42)
regressor.fit(x_train, y_train)
```

Out[31]:
```
DecisionTreeRegressor(random_state=42)
```

In [32]:
```
# Making predictions

preds = regressor.predict(x_test)
```

In [33]:
```
# Evaluating the model

mse = mean_squared_error(y_test, preds)
r2 = r2_score(y_test, preds)
print('Mean Squared Error (MSE):', mse)
print('R-squared (R^2) score:', r2)
```

```
Mean Squared Error (MSE): 327520227162.98267
R-squared (R^2) score: 0.9989780499428523
```