In [1]:
```
# Importing the libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

In [3]:
```
df=pd.read_csv(r"C:\Users\Prajakta Bose\Downloads\Crop Production data.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	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 [5]:
```
df.tail()
```

Out[5]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
246086	West Bengal	PURULIA	2014	Summer	Rice	306.0	801.0
246087	West Bengal	PURULIA	2014	Summer	Sesamum	627.0	463.0
246088	West Bengal	PURULIA	2014	Whole Year	Sugarcane	324.0	16250.0
246089	West Bengal	PURULIA	2014	Winter	Rice	279151.0	597899.0
246090	West Bengal	PURULIA	2014	Winter	Sesamum	175.0	88.0

In [6]:
```
df.shape
```

Out[6]:
```
(246091, 7)
```

In [7]:
```
df.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 [8]:

```
df.isnull().sum()
```

Out[8]:

```
State_Name          0
District_Name       0
Crop_Year           0
Season              0
Crop                0
Area                0
Production       3730
dtype: int64
```

In [9]:

```
#Dropping missing values
df.dropna(subset=["Production"],axis=0,inplace=True)
```

In [10]:

```
df.isnull().sum()
```

Out[10]:

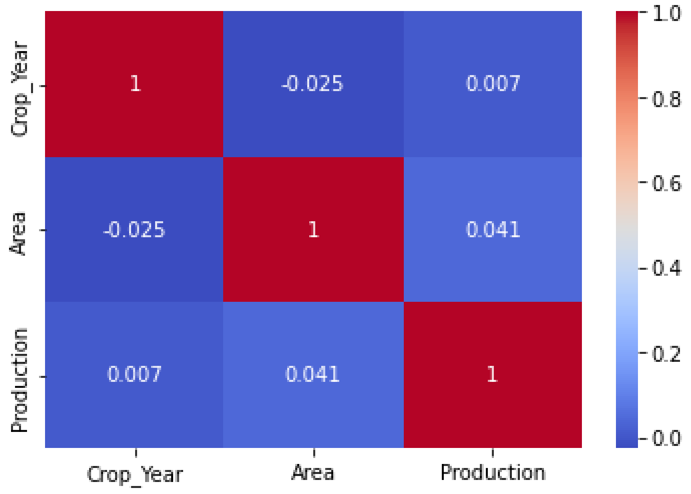
```
State_Name       0
District_Name    0
Crop_Year        0
Season           0
Crop             0
Area             0
Production       0
dtype: int64
```

In [12]:

```
# Correlation Matrix
corr_matrix = df.corr()
```

In [14]:

```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```



EXPLORATORY DATA ANALYSIS

UNIVARIATE ANALYSIS

There are 7 columns , we will explore each column one by one

In [16]:
```
#First we will see column 'State_Name'
df.State_Name.unique()
```

Out[16]:
```
array(['Andaman and Nicobar Islands', 'Andhra Pradesh',
       'Arunachal Pradesh', 'Assam', 'Bihar', 'Chandigarh',
       'Chhattisgarh', 'Dadra and Nagar Haveli', 'Goa', 'Gujarat',
       'Haryana', 'Himachal Pradesh', 'Jammu and Kashmir ', 'Jharkhand',
       'Karnataka', 'Kerala', 'Madhya Pradesh', 'Maharashtra', 'Manipur',
       'Meghalaya', 'Mizoram', 'Nagaland', 'Odisha', 'Puducherry',
       'Punjab', 'Rajasthan', 'Sikkim', 'Tamil Nadu', 'Telangana ',
       'Tripura', 'Uttar Pradesh', 'Uttarakhand', 'West Bengal'],
      dtype=object)
```

In [17]:
```
df.State_Name.nunique()
```

Out[17]:
```
33
```

Exploring column "District_Name"

In [18]:
```
df.District_Name.unique()
```

Out[18]:
```
array(['NICOBARS', 'NORTH AND MIDDLE ANDAMAN', 'SOUTH ANDAMANS',
       'ANANTAPUR', 'CHITTOOR', 'EAST GODAVARI', 'GUNTUR', 'KADAPA',
       'KRISHNA', 'KURNOOL', 'PRAKASAM', 'SPSR NELLORE', 'SRIKAKULAM',
       'VISAKHAPATANAM', 'VIZIANAGARAM', 'WEST GODAVARI', 'ANJAW',
       'CHANGLANG', 'DIBANG VALLEY', 'EAST KAMENG', 'EAST SIANG',
       'KURUNG KUMEY', 'LOHIT', 'LONGDING', 'LOWER DIBANG VALLEY',
       'LOWER SUBANSIRI', 'NAMSAI', 'PAPUM PARE', 'TAWANG', 'TIRAP',
       'UPPER SIANG', 'UPPER SUBANSIRI', 'WEST KAMENG', 'WEST SIANG',
       'BAKSA', 'BARPETA', 'BONGAIGAON', 'CACHAR', 'CHIRANG', 'DARRANG',
       'DHEMAJI', 'DHUBRI', 'DIBRUGARH', 'DIMA HASAO', 'GOALPARA',
       'GOLAGHAT', 'HAILAKANDI', 'JORHAT', 'KAMRUP', 'KAMRUP METRO',
       'KARBI ANGLONG', 'KARIMGANJ', 'KOKRAJHAR', 'LAKHIMPUR', 'MARIGAON',
       'NAGAON', 'NALBARI', 'SIVASAGAR', 'SONITPUR', 'TINSUKIA',
       'UDALGURI', 'ARARIA', 'ARWAL', 'AURANGABAD', 'BANKA', 'BEGUSARAI',
       'BHAGALPUR', 'BHOJPUR', 'BUXAR', 'DARBHANGA', 'GAYA', 'GOPALGANJ',
       'JAMUI', 'JEHANABAD', 'KAIMUR (BHABUA)', 'KATIHAR', 'KHAGARIA',
       'KISHANGANJ', 'LAKHISARAI', 'MADHEPURA', 'MADHUBANI', 'MUNGER',
       'MUZAFFARPUR', 'NALANDA', 'NAWADA', 'PASHCHIM CHAMPARAN', 'PATNA',
       'PURBI CHAMPARAN', 'PURNIA', 'ROHTAS', 'SAHARSA', 'SAMASTIPUR',
       'SARAN', 'SHEIKHPURA', 'SHEOHAR', 'SITAMARHI', 'SIWAN', 'SUPAUL',
       'VAISHALI', 'CHANDIGARH', 'BALOD', 'BALODA BAZAR', 'BALRAMPUR',
       'BASTAR', 'BEMETARA', 'BIJAPUR', 'BILASPUR', 'DANTEWADA',
       'DHAMTARI', 'DURG', 'GARIYABAND', 'JANJGIR-CHAMPA', 'JASHPUR',
       'KABIRDHAM', 'KANKER', 'KONDAGAON', 'KORBA', 'KOREA', 'MAHASAMUND',
       'MUNGELI', 'NARAYANPUR', 'RAIGARH', 'RAIPUR', 'RAJNANDGAON',
       'SUKMA', 'SURAJPUR', 'SURGUJA', 'DADRA AND NAGAR HAVELI',
       'NORTH GOA', 'SOUTH GOA', 'AHMADABAD', 'AMRELI', 'ANAND',
       'BANAS KANTHA', 'BHARUCH', 'BHAVNAGAR', 'DANG', 'DOHAD',
       'GANDHINAGAR', 'JAMNAGAR', 'JUNAGADH', 'KACHCHH', 'KHEDA',
       'MAHESANA', 'NARMADA', 'NAVSARI', 'PANCH MAHALS', 'PATAN',
       'PORBANDAR', 'RAJKOT', 'SABAR KANTHA', 'SURAT', 'SURENDRANAGAR',
       'TAPI', 'VADODARA', 'VALSAD', 'AMBALA', 'BHIWANI', 'FARIDABAD',
       'FATEHABAD', 'GURGAON', 'HISAR', 'JHAJJAR', 'JIND', 'KAITHAL',
       'KARNAL', 'KURUKSHETRA', 'MAHENDRAGARH', 'MEWAT', 'PALWAL',
       'PANCHKULA', 'PANIPAT', 'REWARI', 'ROHTAK', 'SIRSA', 'SONIPAT',
       'YAMUNANAGAR', 'CHAMBA', 'HAMIRPUR', 'KANGRA', 'KINNAUR', 'KULLU',
       'LAHUL AND SPITI', 'MANDI', 'SHIMLA', 'SIRMAUR', 'SOLAN', 'UNA',
       'ANANTNAG', 'BADGAM', 'BANDIPORA', 'BARAMULLA', 'DODA',
       'GANDERBAL', 'JAMMU', 'KARGIL', 'KATHUA', 'KISHTWAR', 'KULGAM',
       'KUPWARA', 'LEH LADAKH', 'POONCH', 'PULWAMA', 'RAJAURI', 'RAMBAN',
       'REASI', 'SAMBA', 'SHOPIAN', 'SRINAGAR', 'UDHAMPUR', 'BOKARO',
       'CHATRA', 'DEOGHAR', 'DHANBAD', 'DUMKA', 'EAST SINGHBUM', 'GARHWA',
       'GIRIDIH', 'GODDA', 'GUMLA', 'HAZARIBAGH', 'JAMTARA', 'KHUNTI',
       'KODERMA', 'LATEHAR', 'LOHARDAGA', 'PAKUR', 'PALAMU', 'RAMGARH',
       'RANCHI', 'SAHEBGANJ', 'SARAIKELA KHARSAWAN', 'SIMDEGA',
       'WEST SINGHBHUM', 'BAGALKOT', 'BANGALORE RURAL', 'BELGAUM',
       'BELLARY', 'BENGALURU URBAN', 'BIDAR', 'CHAMARAJANAGAR',
       'CHIKBALLAPUR', 'CHIKMAGALUR', 'CHITRADURGA', 'DAKSHIN KANNAD',
       'DAVANGERE', 'DHARWAD', 'GADAG', 'GULBARGA', 'HASSAN', 'HAVERI',
       'KODAGU', 'KOLAR', 'KOPPAL', 'MANDYA', 'MYSORE', 'RAICHUR',
       'RAMANAGARA', 'SHIMOGA', 'TUMKUR', 'UDUPI', 'UTTAR KANNAD',
       'YADGIR', 'ALAPPUZHA', 'ERNAKULAM', 'IDUKKI', 'KANNUR',
       'KASARAGOD', 'KOLLAM', 'KOTTAYAM', 'KOZHIKODE', 'MALAPPURAM',
       'PALAKKAD', 'PATHANAMTHITTA', 'THIRUVANANTHAPURAM', 'THRISSUR',
       'WAYANAD', 'AGAR MALWA', 'ALIRAJPUR', 'ANUPPUR', 'ASHOKNAGAR',
       'BALAGHAT', 'BARWANI', 'BETUL', 'BHIND', 'BHOPAL', 'BURHANPUR',
       'CHHATARPUR', 'CHHINDWARA', 'DAMOH', 'DATIA', 'DEWAS', 'DHAR',
       'DINDORI', 'GUNA', 'GWALIOR', 'HARDA', 'HOSHANGABAD', 'INDORE',
       'JABALPUR', 'JHABUA', 'KATNI', 'KHANDWA', 'KHARGONE', 'MANDLA',
       'MANDSAUR', 'MORENA', 'NARSINGHPUR', 'NEEMUCH', 'PANNA', 'RAISEN',
       'RAJGARH', 'RATLAM', 'REWA', 'SAGAR', 'SATNA', 'SEHORE', 'SEONI',
       'SHAHDOL', 'SHAJAPUR', 'SHEOPUR', 'SHIVPURI', 'SIDHI', 'SINGRAULI',
       'TIKAMGARH', 'UJJAIN', 'UMARIA', 'VIDISHA', 'AHMEDNAGAR', 'AKOLA',
       'AMRAVATI', 'BEED', 'BHANDARA', 'BULDHANA', 'CHANDRAPUR', 'DHULE',
       'GADCHIROLI', 'GONDIA', 'HINGOLI', 'JALGAON', 'JALNA', 'KOLHAPUR',
       'LATUR', 'MUMBAI', 'NAGPUR', 'NANDED', 'NANDURBAR', 'NASHIK',
       'OSMANABAD', 'PALGHAR', 'PARBHANI', 'PUNE', 'RAIGAD', 'RATNAGIRI',
       'SANGLI', 'SATARA', 'SINDHUDURG', 'SOLAPUR', 'THANE', 'WARDHA',
       'WASHIM', 'YAVATMAL', 'BISHNUPUR', 'CHANDEL', 'CHURACHANDPUR',
       'IMPHAL EAST', 'IMPHAL WEST', 'SENAPATI', 'TAMENGLONG', 'THOUBAL',
       'UKHRUL', 'EAST GARO HILLS', 'EAST JAINTIA HILLS',
       'EAST KHASI HILLS', 'NORTH GARO HILLS', 'RI BHOI',
       'SOUTH GARO HILLS', 'SOUTH WEST GARO HILLS',
       'SOUTH WEST KHASI HILLS', 'WEST GARO HILLS', 'WEST JAINTIA HILLS',
       'WEST KHASI HILLS', 'AIZAWL', 'CHAMPHAI', 'KOLASIB', 'LAWNGTLAI',
       'LUNGLEI', 'MAMIT', 'SAIHA', 'SERCHHIP', 'DIMAPUR', 'KIPHIRE',
       'KOHIMA', 'LONGLENG', 'MOKOKCHUNG', 'MON', 'PEREN', 'PHEK',
       'TUENSANG', 'WOKHA', 'ZUNHEBOTO', 'ANUGUL', 'BALANGIR',
       'BALESHWAR', 'BARGARH', 'BHADRAK', 'BOUDH', 'CUTTACK', 'DEOGARH',
       'DHENKANAL', 'GAJAPATI', 'GANJAM', 'JAGATSINGHAPUR', 'JAJAPUR',
       'JHARSUGUDA', 'KALAHANDI', 'KANDHAMAL', 'KENDRAPARA', 'KENDUJHAR',
       'KHORDHA', 'KORAPUT', 'MALKANGIRI', 'MAYURBHANJ', 'NABARANGPUR',
       'NAYAGARH', 'NUAPADA', 'PURI', 'RAYAGADA', 'SAMBALPUR', 'SONEPUR',
       'SUNDARGARH', 'KARAIKAL', 'MAHE', 'PONDICHERRY', 'YANAM',
       'AMRITSAR', 'BARNALA', 'BATHINDA', 'FARIDKOT', 'FATEHGARH SAHIB',
       'FAZILKA', 'FIROZEPUR', 'GURDASPUR', 'HOSHIARPUR', 'JALANDHAR',
       'KAPURTHALA', 'LUDHIANA', 'MANSA', 'MOGA', 'MUKTSAR', 'NAWANSHAHR',
       'PATHANKOT', 'PATIALA', 'RUPNAGAR', 'S.A.S NAGAR', 'SANGRUR',
       'TARN TARAN', 'AJMER', 'ALWAR', 'BANSWARA', 'BARAN', 'BARMER',
       'BHARATPUR', 'BHILWARA', 'BIKANER', 'BUNDI', 'CHITTORGARH',
       'CHURU', 'DAUSA', 'DHOLPUR', 'DUNGARPUR', 'GANGANAGAR',
       'HANUMANGARH', 'JAIPUR', 'JAISALMER', 'JALORE', 'JHALAWAR',
       'JHUNJHUNU', 'JODHPUR', 'KARAULI', 'KOTA', 'NAGAUR', 'PALI',
       'PRATAPGARH', 'RAJSAMAND', 'SAWAI MADHOPUR', 'SIKAR', 'SIROHI',
       'TONK', 'UDAIPUR', 'EAST DISTRICT', 'NORTH DISTRICT',
       'SOUTH DISTRICT', 'WEST DISTRICT', 'ARIYALUR', 'COIMBATORE',
       'CUDDALORE', 'DHARMAPURI', 'DINDIGUL', 'ERODE', 'KANCHIPURAM',
       'KANNIYAKUMARI', 'KARUR', 'KRISHNAGIRI', 'MADURAI', 'NAGAPATTINAM',
       'NAMAKKAL', 'PERAMBALUR', 'PUDUKKOTTAI', 'RAMANATHAPURAM', 'SALEM',
       'SIVAGANGA', 'THANJAVUR', 'THE NILGIRIS', 'THENI', 'THIRUVALLUR',
       'THIRUVARUR', 'TIRUCHIRAPPALLI', 'TIRUNELVELI', 'TIRUPPUR',
       'TIRUVANNAMALAI', 'TUTICORIN', 'VELLORE', 'VILLUPURAM',
       'VIRUDHUNAGAR', 'ADILABAD', 'HYDERABAD', 'KARIMNAGAR', 'KHAMMAM',
       'MAHBUBNAGAR', 'MEDAK', 'NALGONDA', 'NIZAMABAD', 'RANGAREDDI',
       'WARANGAL', 'DHALAI', 'GOMATI', 'KHOWAI', 'NORTH TRIPURA',
       'SEPAHIJALA', 'SOUTH TRIPURA', 'UNAKOTI', 'WEST TRIPURA', 'AGRA',
       'ALIGARH', 'ALLAHABAD', 'AMBEDKAR NAGAR', 'AMETHI', 'AMROHA',
       'AURAIYA', 'AZAMGARH', 'BAGHPAT', 'BAHRAICH', 'BALLIA', 'BANDA',
       'BARABANKI', 'BAREILLY', 'BASTI', 'BIJNOR', 'BUDAUN',
       'BULANDSHAHR', 'CHANDAULI', 'CHITRAKOOT', 'DEORIA', 'ETAH',
       'ETAWAH', 'FAIZABAD', 'FARRUKHABAD', 'FATEHPUR', 'FIROZABAD',
       'GAUTAM BUDDHA NAGAR', 'GHAZIABAD', 'GHAZIPUR', 'GONDA',
       'GORAKHPUR', 'HAPUR', 'HARDOI', 'HATHRAS', 'JALAUN', 'JAUNPUR',
       'JHANSI', 'KANNAUJ', 'KANPUR DEHAT', 'KANPUR NAGAR', 'KASGANJ',
       'KAUSHAMBI', 'KHERI', 'KUSHI NAGAR', 'LALITPUR', 'LUCKNOW',
       'MAHARAJGANJ', 'MAHOBA', 'MAINPURI', 'MATHURA', 'MAU', 'MEERUT',
       'MIRZAPUR', 'MORADABAD', 'MUZAFFARNAGAR', 'PILIBHIT', 'RAE BARELI',
       'RAMPUR', 'SAHARANPUR', 'SAMBHAL', 'SANT KABEER NAGAR',
       'SANT RAVIDAS NAGAR', 'SHAHJAHANPUR', 'SHAMLI', 'SHRAVASTI',
       'SIDDHARTH NAGAR', 'SITAPUR', 'SONBHADRA', 'SULTANPUR', 'UNNAO',
       'VARANASI', 'ALMORA', 'BAGESHWAR', 'CHAMOLI', 'CHAMPAWAT',
       'DEHRADUN', 'HARIDWAR', 'NAINITAL', 'PAURI GARHWAL', 'PITHORAGARH',
       'RUDRA PRAYAG', 'TEHRI GARHWAL', 'UDAM SINGH NAGAR', 'UTTAR KASHI',
       '24 PARAGANAS NORTH', '24 PARAGANAS SOUTH', 'BANKURA', 'BARDHAMAN',
       'BIRBHUM', 'COOCHBEHAR', 'DARJEELING', 'DINAJPUR DAKSHIN',
       'DINAJPUR UTTAR', 'HOOGHLY', 'HOWRAH', 'JALPAIGURI', 'MALDAH',
       'MEDINIPUR EAST', 'MEDINIPUR WEST', 'MURSHIDABAD', 'NADIA',
       'PURULIA'], dtype=object)
```

In [19]:

```
df.District_Name.nunique()
```

Out[19]:

```
646
```

In [20]:

```
#Crop year
df.Crop_Year.unique()
```

Out[20]:

```
array([2000, 2001, 2002, 2003, 2004, 2005, 2006, 2010, 1997, 1998, 1999,
       2007, 2008, 2009, 2011, 2012, 2013, 2014, 2015], dtype=int64)
```

In [22]:

```
# The dataset has data for 19 years fromm 1997 to 2015
# Season
df.Season .unique()
```

Out[22]:

```
array(['Kharif     ', 'Whole Year ', 'Autumn     ', 'Rabi       ',
       'Summer     ', 'Winter     '], dtype=object)
```

In [23]:

```
df.Season .value_counts()
```

Out[23]:

```
Kharif         94283
Rabi           66160
Whole Year     56127
Summer         14811
Winter          6050
Autumn          4930
Name: Season, dtype: int64
```

In [24]:

```
#Crop
df.Crop.unique()
```

Out[24]:

```
array(['Arecanut', 'Other Kharif pulses', 'Rice', 'Banana', 'Cashewnut',
       'Coconut ', 'Dry ginger', 'Sugarcane', 'Sweet potato', 'Tapioca',
       'Black pepper', 'Dry chillies', 'other oilseeds', 'Turmeric',
       'Maize', 'Moong(Green Gram)', 'Urad', 'Arhar/Tur', 'Groundnut',
       'Sunflower', 'Bajra', 'Castor seed', 'Cotton(lint)', 'Horse-gram',
       'Jowar', 'Korra', 'Ragi', 'Tobacco', 'Gram', 'Wheat', 'Masoor',
       'Sesamum', 'Linseed', 'Safflower', 'Onion', 'other misc. pulses',
       'Samai', 'Small millets', 'Coriander', 'Potato',
       'Other  Rabi pulses', 'Soyabean', 'Beans & Mutter(Vegetable)',
       'Bhindi', 'Brinjal', 'Citrus Fruit', 'Cucumber', 'Grapes', 'Mango',
       'Orange', 'other fibres', 'Other Fresh Fruits', 'Other Vegetables',
       'Papaya', 'Pome Fruit', 'Tomato', 'Mesta', 'Cowpea(Lobia)',
       'Lemon', 'Pome Granet', 'Sapota', 'Cabbage', 'Rapeseed &Mustard',
       'Peas  (vegetable)', 'Niger seed', 'Bottle Gourd', 'Varagu',
       'Garlic', 'Ginger', 'Oilseeds total', 'Pulses total', 'Jute',
       'Peas & beans (Pulses)', 'Blackgram', 'Paddy', 'Pineapple',
       'Barley', 'Sannhamp', 'Khesari', 'Guar seed', 'Moth',
       'Other Cereals & Millets', 'Cond-spcs other', 'Turnip', 'Carrot',
       'Redish', 'Arcanut (Processed)', 'Atcanut (Raw)',
       'Cashewnut Processed', 'Cashewnut Raw', 'Cardamom', 'Rubber',
       'Bitter Gourd', 'Drum Stick', 'Jack Fruit', 'Snak Guard', 'Tea',
       'Coffee', 'Cauliflower', 'Other Citrus Fruit', 'Water Melon',
       'Total foodgrain', 'Kapas', 'Colocosia', 'Lentil', 'Bean',
       'Jobster', 'Perilla', 'Rajmash Kholar', 'Ricebean (nagadal)',
       'Ash Gourd', 'Beet Root', 'Lab-Lab', 'Ribed Guard', 'Yam',
       'Pump Kin', 'Apple', 'Peach', 'Pear', 'Plums', 'Litchi', 'Ber',
       'Other Dry Fruit', 'Jute & mesta'], dtype=object)
```

In [25]:
```
df.Crop.nunique()
```

Out[25]:
```
124
```

In [27]:
```
# We get to know that top 3 crops are Rice 15082 #Maize 13787 #Moong(Green Gram) 101
df.head()
```

Out[27]:

	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

AREA Check the distribution of area using descriptive statistics

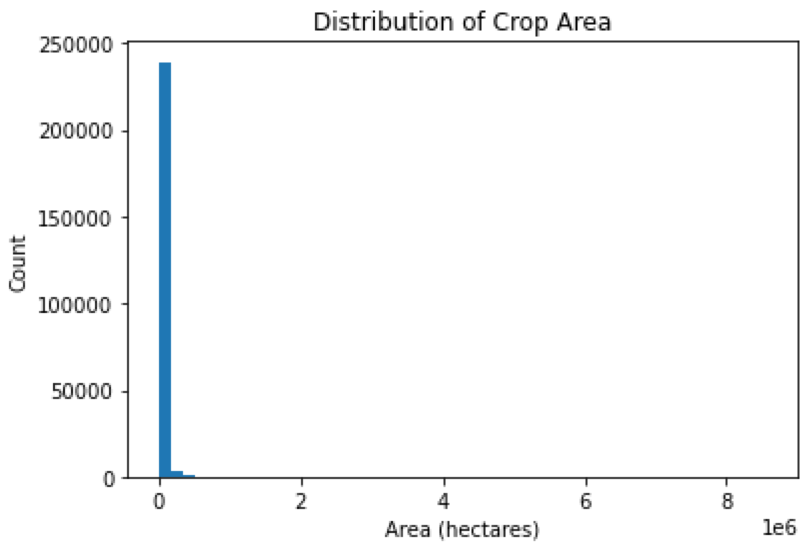
In [29]:
```
area_stats = df['Area'].describe()
print(area_stats)
```

```
count    2.423610e+05
mean     1.216741e+04
std      5.085744e+04
min      1.000000e-01
25%      8.700000e+01
50%      6.030000e+02
75%      4.545000e+03
```

```
max      8.580100e+06
Name: Area, dtype: float64
```

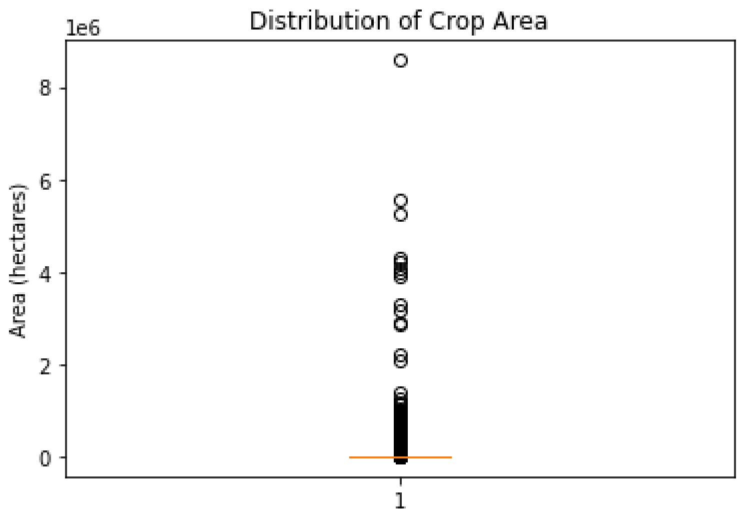
The "count" row shows that there are 242,361 entries in this column. The "mean" row shows that the average area of land used for cultivation is around 12,167 hectares. The "std" row shows that there is a lot of variation in the size of land used for cultivation, with a standard deviation of around 50,857 hectares. The "min" row shows that the smallest area used for cultivation in the dataset is 0.1 hectares. The "25%" row shows that 25% of the areas used for cultivation are less than or equal to 87 hectares. The "50%" row shows that 50% of the areas used for cultivation are less than or equal to 603 hectares. The "75%" row shows that 75% of the areas used for cultivation are less than or equal to 4,545 hectares. The "max" row shows that the largest area used for cultivation in the dataset is 8,580,100 hectares.

In [30]:
```
# Check the distribution of area using a histogram
plt.hist(df['Area'], bins=50)
plt.title('Distribution of Crop Area')
plt.xlabel('Area (hectares)')
plt.ylabel('Count')
plt.show()
```



The x-axis represents the area in hectares, and the y-axis represents the count of crops with that area. The histogram shows how the crop areas are distributed, with the majority of crops having a smaller area and fewer crops having larger areas.

In [31]:
```
# Check for outliers or extreme values in the data using a boxplot
plt.boxplot(df['Area'])
plt.title('Distribution of Crop Area')
plt.ylabel('Area (hectares)')
plt.show()
```



The boxplot shows a box with whiskers extending from the box, indicating the range of the data. The central box represents the middle 50% of the data, with the line inside the box representing the median value. The upper and lower whiskers represent the highest and lowest data points within a certain range of the median. Any data points outside of the whiskers are considered outliers.

In this particular boxplot, we can see that there are many outliers, as the whiskers extend very far from the box. This suggests that there are some crop areas that are much larger than the typical values in the dataset.

Production

In [34]:
```
# summary statistics for the "Production" column
production_stats = df['Production'].describe()
print(production_stats)
```

```
count    2.423610e+05
mean     5.825034e+05
std      1.706581e+07
min      0.000000e+00
25%      8.800000e+01
50%      7.290000e+02
75%      7.023000e+03
max      1.250800e+09
Name: Production, dtype: float64
```

1) count: The number of non-null values in the "Production" column. In this case, there are 242,361 non-null values.

2) mean: The arithmetic mean (average) of the "Production" column. In this case, the average production value is approximately 582,503.4 metric tons.

3) std: The standard deviation of the "Production" column, which measures the spread of the data around the mean. In this case, the standard deviation is approximately 17,065,810 metric tons.

4) min: The minimum value in the "Production" column. In this case, the minimum production value is 0 metric tons.

5) 25%: The first quartile (Q1), which is the value below which 25% of the data falls. In this case, 25% of the production values are below 88 metric tons.

6) 50%: The second quartile (Q2), which is the median or the value below which 50% of the data falls. In this case, the median production value is 729 metric tons.

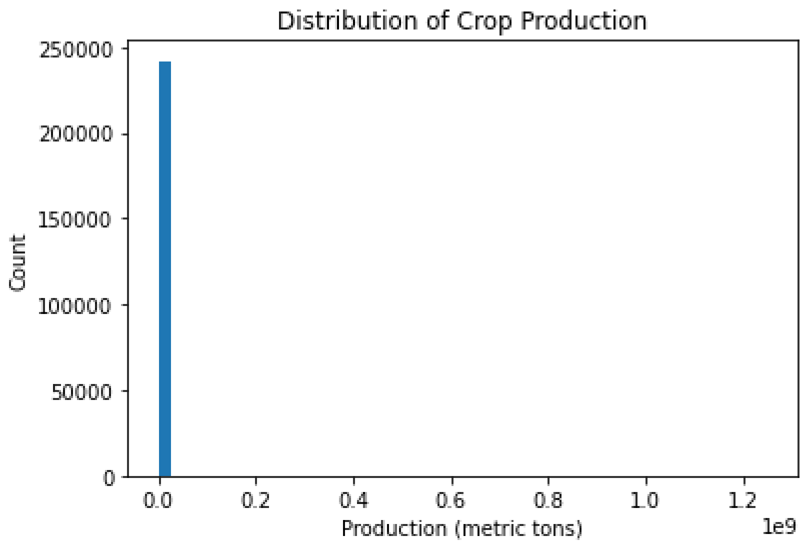
7) 75%: The third quartile (Q3), which is the value below which 75% of the data falls. In this case, 75% of the production values are below 7,023 metric tons.

9) max: The maximum value in the "Production" column. In this case, the maximum production value is 1,250,800,000 metric tons.

These statistics can provide insights into the central tendency, spread, and range of the "Production" column, which can help you understand the distribution of the data and identify any potential issues or outliers.

In [36]:

```
 #Histogram of the "Production" column
plt.hist(df['Production'], bins=50)
plt.title('Distribution of Crop Production')
plt.xlabel('Production (metric tons)')
plt.ylabel('Count')
plt.show()
```



Here's what the different parts of the histogram plot represent:

The x-axis represents the "Production" values, which are divided into different bins based on the specified bin size. In this case, the bin size is set to 50, so each bar in the histogram represents the count of production values falling within a range of 50 metric tons.

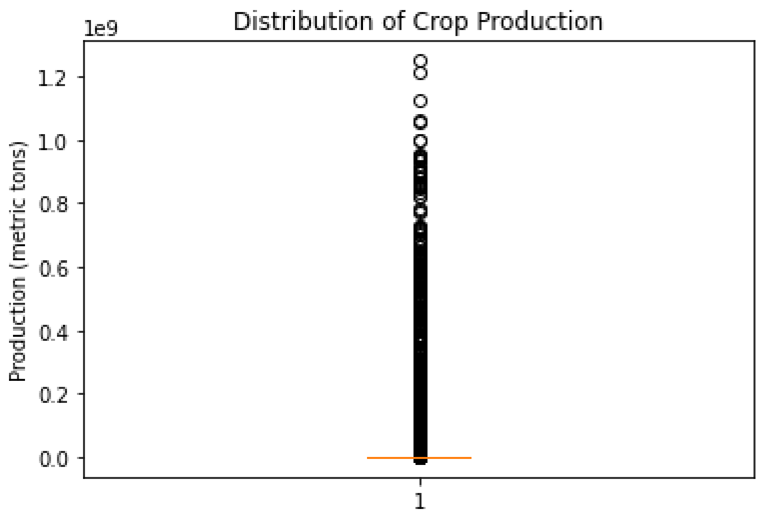
The y-axis represents the count of production values that fall within each bin range. The higher the bar, the more number of production values that fall within that bin range.

The title of the plot is "Distribution of Crop Production", which provides an overview of the plot.

The x-axis label is "Production (metric tons)", which provides a description of the x-axis.

The y-axis label is "Count", which provides a description of the y-axis.

In [38]:
```
plt.boxplot(df['Production'])
plt.title('Distribution of Crop Production')
plt.ylabel('Production (metric tons)')
plt.show()
```



Here's what the different parts of the boxplot represent:

The box represents the interquartile range (IQR), which contains the middle 50% of the "Production" values. The bottom of the box represents the first quartile (Q1), which is the 25th percentile of the "Production" values. The top of the box represents the third quartile (Q3), which is the 75th percentile of the "Production" values. The line inside the box represents the median, which is the 50th percentile of the "Production" values.

The whiskers extend from the box to the minimum and maximum values within 1.5 times the IQR. Any values beyond the whiskers are considered outliers and are plotted as individual points.

The title of the plot is "Distribution of Crop Production", which provides an overview of the plot.

The y-axis label is "Production (metric tons)", which provides a description of the y-axis.

By looking at the boxplot, we can quickly identify the median, IQR, and any potential outliers in the "Production" values. We can also determine whether the distribution is skewed or symmetrical. In this case, the boxplot suggests that the "Production" column contains a large number of outliers, and the distribution is highly skewed towards the lower values.

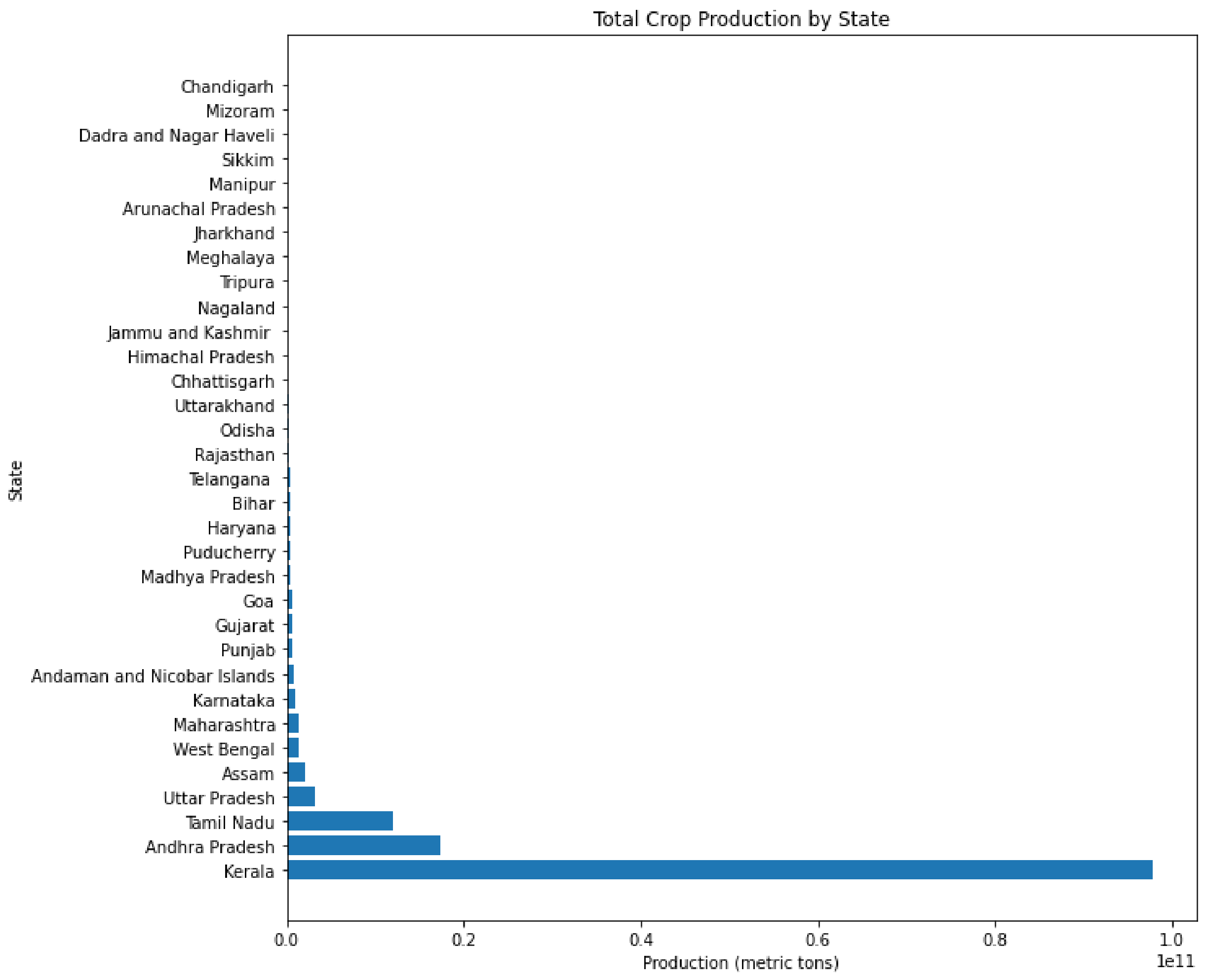
In [39]:
```
# Bivarate Analysis
state_production = df.groupby('State_Name')['Production'].sum().reset_index()
print(state_production)
```

```
                     State_Name    Production
0   Andaman and Nicobar Islands  7.182232e+08
1                Andhra Pradesh  1.732459e+10
2             Arunachal Pradesh  6.823913e+06
3                         Assam  2.111752e+09
4                         Bihar  3.664836e+08
5                    Chandigarh  6.395650e+04
6                  Chhattisgarh  1.009519e+08
7        Dadra and Nagar Haveli  1.847871e+06
8                           Goa  5.057558e+08
9                       Gujarat  5.242913e+08
```

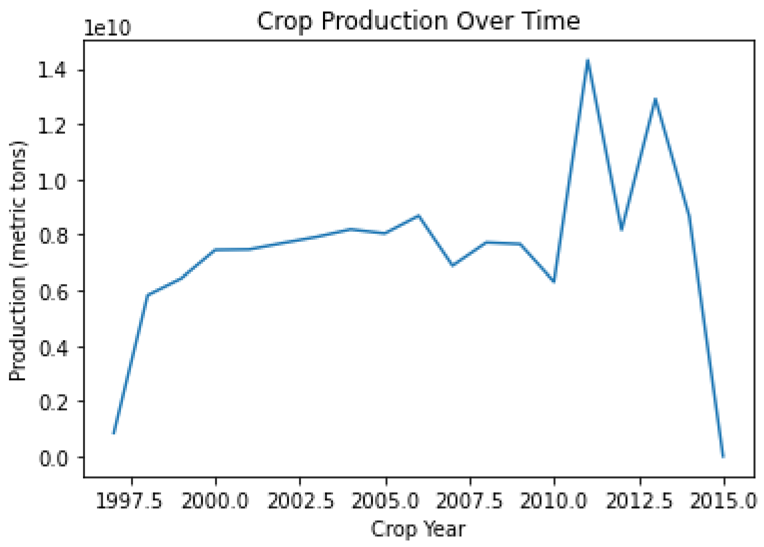
```
10                    Haryana  3.812739e+08
11           Himachal Pradesh  1.780517e+07
12          Jammu and Kashmir   1.329102e+07
13                  Jharkhand  1.077774e+07
14                  Karnataka  8.634298e+08
15                     Kerala  9.788005e+10
16             Madhya Pradesh  4.488407e+08
17                Maharashtra  1.263641e+09
18                    Manipur  5.230917e+06
19                  Meghalaya  1.211250e+07
20                    Mizoram  1.661540e+06
21                   Nagaland  1.276595e+07
22                     Odisha  1.609041e+08
23                 Puducherry  3.847245e+08
24                     Punjab  5.863850e+08
25                  Rajasthan  2.813203e+08
26                     Sikkim  2.435735e+06
27                 Tamil Nadu  1.207644e+10
28                 Telangana   3.351479e+08
29                    Tripura  1.252292e+07
30              Uttar Pradesh  3.234493e+09
31                Uttarakhand  1.321774e+08
32                West Bengal  1.397904e+09
```

In [44]:

```python
state_production = df.groupby('State_Name')['Production'].sum().reset_index()
# Sort the dataframe in descending order of production
state_production = state_production.sort_values(by='Production', ascending=False)
# Create the bar chart fig,
fig, ax = plt.subplots(figsize=(10, 10))
ax.barh(state_production['State_Name'], state_production['Production'])
ax.set_title('Total Crop Production by State')
ax.set_xlabel('Production (metric tons)')
ax.set_ylabel('State')
plt.show()
```

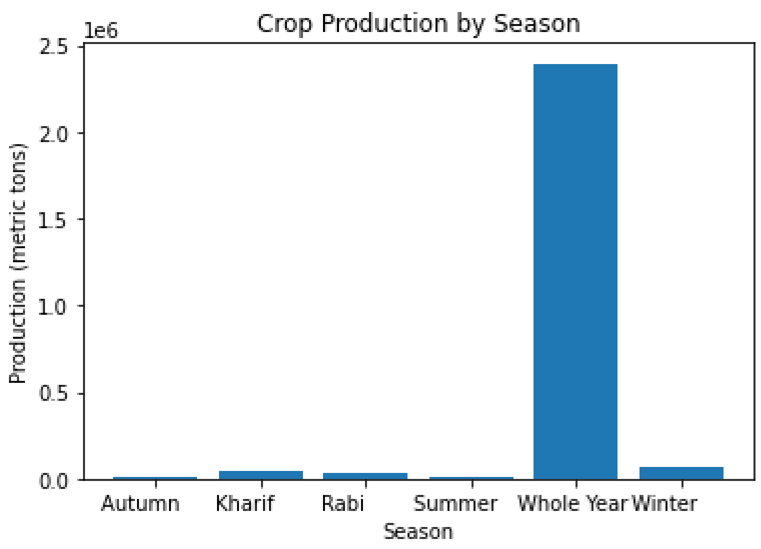


In [45]:
```
# Performing bivariate analysis on the "Crop_Year" and "Production" columns to see h
year_production = df.groupby('Crop_Year')['Production'].sum().reset_index()
# Create the line chart
plt.plot(year_production['Crop_Year'], year_production['Production'])
plt.title('Crop Production Over Time')
plt.xlabel('Crop Year')
plt.ylabel('Production (metric tons)')
plt.show()
```



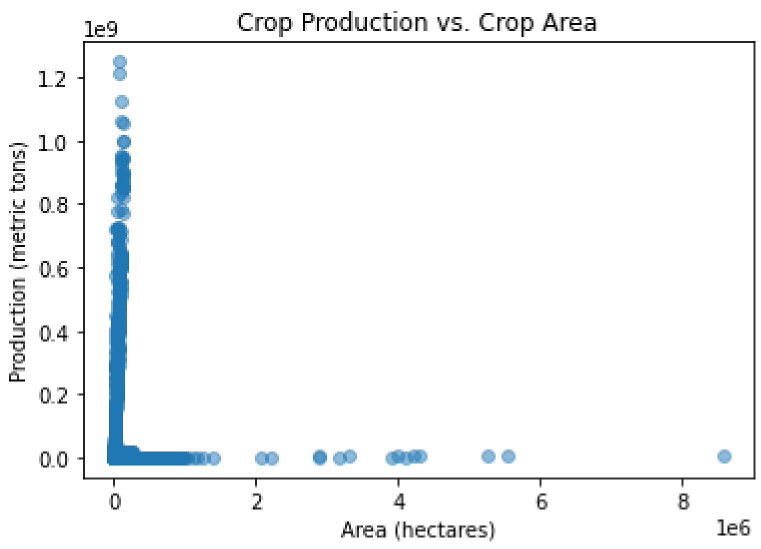
In [46]:
```
# Bivariate analysis on the "Season" and "Production" columns to see how the crop pr
# Group the data by season and calculate the mean production for each season
season_data = df.groupby('Season')['Production'].mean()
# Create a bar chart showing the production by season
```

```
plt.bar(season_data.index, season_data.values)
# Set the title and axis labels
plt.title('Crop Production by Season')
plt.xlabel('Season')
plt.ylabel('Production (metric tons)')
# Show the plot
plt.show()
```



This Histogram shows how the crop production varies across different seasons. The height of each bar represents the mean production for that season.

In [47]:
```
# Create a scatter plot showing the relationship between crop area and production
plt.scatter(df['Area'], df['Production'], alpha=0.5)
# Set the title and axis labels
plt.title('Crop Production vs. Crop Area')
plt.xlabel('Area (hectares)')
plt.ylabel('Production (metric tons)')
# Show the plot
plt.show()
```



This scatter plot shows how the size of the crop area affects the amount of crop produced. Each point on the plot represents a single crop, with the x-coordinate representing the crop area and the y-coordinate representing the crop production.

Productivity of different States To find the productivity of different states, we need to calculate the crop yield for each state. Crop yield is the amount of crop harvested per unit of land area, typically expressed in kilograms per hectare (kg/ha) or metric tons per hectare (t/ha).

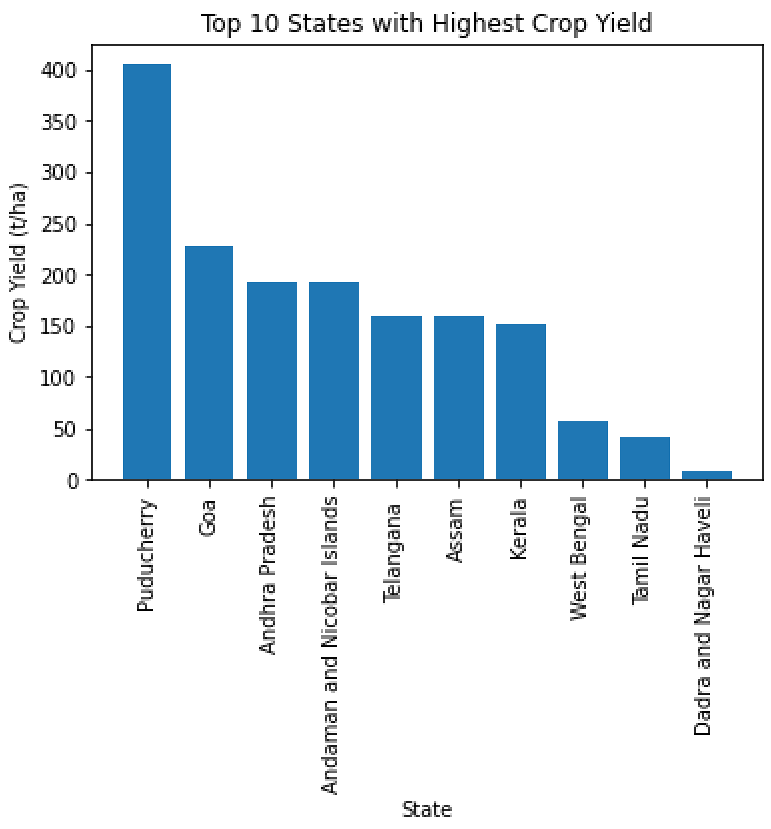
We can calculate the crop yield for each state by dividing the total production of each crop in a state by the total area of land used to grow that crop in the same state.

In [48]:
```
# Calculate crop yield for each state
df_yield = df.groupby(['State_Name', 'Crop']).agg({'Area': 'sum', 'Production': 'sum
df_yield['Yield'] = df_yield['Production'] / df_yield['Area']
df_yield = df_yield.reset_index()
```

Now the data has been grouped by state and crop. The above code sums the area and production for each group, and then calculates the yield by dividing production by area. The resulting dataframe, df_yield, contains the state, crop, area, production, and yield for each group.

Visualizing the dataframe to find the productivity of different states

In [50]:
```
# Create a bar chart of top 10 states with highest crop yield
top_states = df_yield.groupby('State_Name').agg({'Yield': 'mean'}).sort_values('Yiel
plt.bar(top_states.index, top_states['Yield'])
plt.title('Top 10 States with Highest Crop Yield')
plt.xlabel('State')
# Rotate the x-axis labels by 90 degrees
plt.xticks(rotation=90)
plt.ylabel('Crop Yield (t/ha)')
plt.show()
```



The bar chart of the top 10 states with state names on the x-axis and crop yield on the y-axis has been created. The resulting plot shows which states have the highest crop yield.

Most and Least Crop producing districts

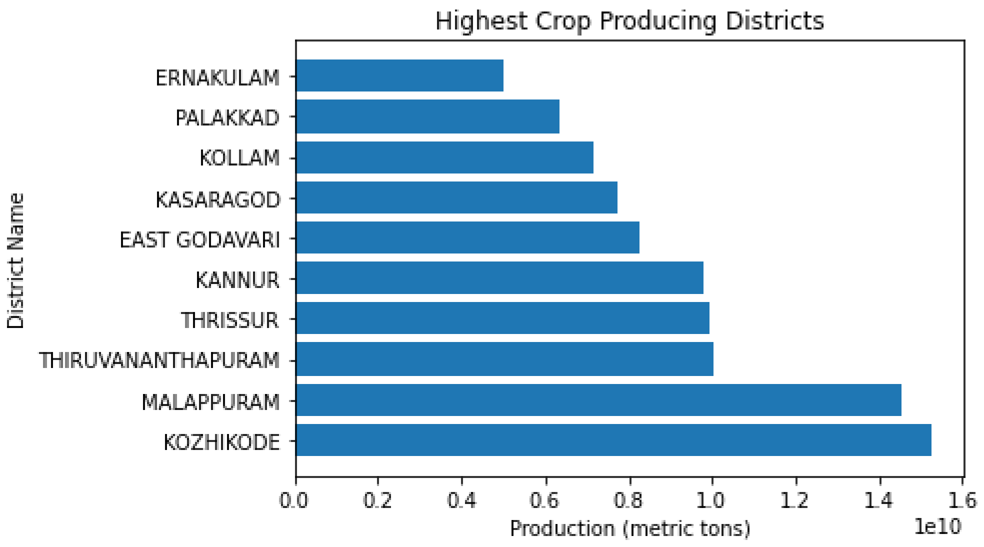
In [51]:
```
# Group the data by district and calculate the total production for each district
district_production = df.groupby('District_Name')['Production'].sum().reset_index()
```

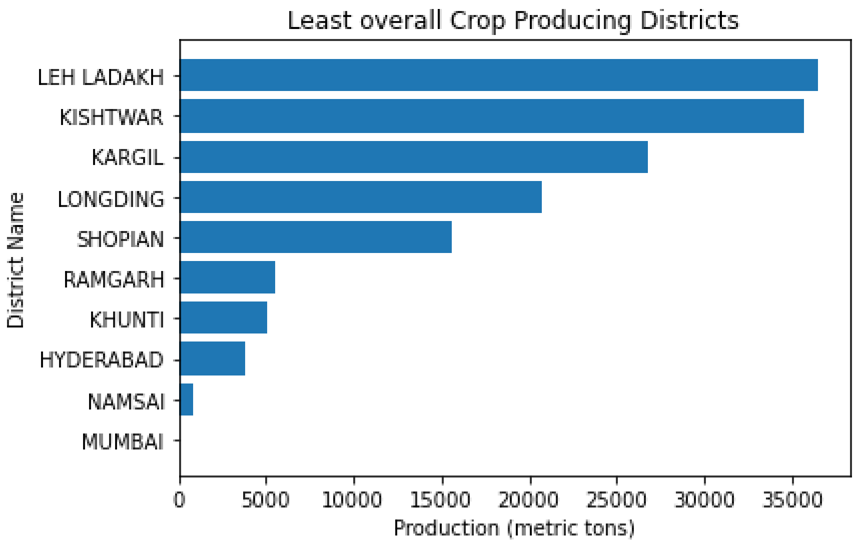
```
# Sort the dataframe in descending order to find the districts with the highest prod
top_districts = district_production.sort_values('Production', ascending=False).head(
# Sort the dataframe in ascending order to find the districts with the lowest produc
bottom_districts = district_production.sort_values('Production', ascending=True).hea
# Print the top and bottom districts
print('Highest Crop producing districts:')
print(top_districts)
print('\n Least overall Crop producing districts:')
print(bottom_districts)
```

```
Highest Crop producing districts:
          District_Name    Production
334           KOZHIKODE  1.528074e+10
372          MALAPPURAM  1.451840e+10
587  THIRUVANANTHAPURAM  1.002271e+10
590            THRISSUR  9.923508e+09
286              KANNUR  9.783432e+09
172       EAST GODAVARI  8.271057e+09
298           KASARAGOD  7.732217e+09
326              KOLLAM  7.151945e+09
437            PALAKKAD  6.369382e+09
178           ERNAKULAM  5.021649e+09

 Least overall Crop producing districts:
    District_Name  Production
397        MUMBAI         2.0
415        NAMSAI       794.0
238     HYDERABAD      3835.0
314        KHUNTI      5024.0
486       RAMGARH      5472.0
537       SHOPIAN     15614.0
354      LONGDING     20731.0
293        KARGIL     26793.8
318      KISHTWAR     35664.2
351    LEH LADAKH     36461.5
```

In [52]:

```
# Create a horizontal bar chart to visualize the top 10 crop producing districts
plt.barh(top_districts['District_Name'], top_districts['Production'])
plt.title('Highest Crop Producing Districts')
plt.xlabel('Production (metric tons)')
plt.ylabel('District Name')
plt.show()
# Create a horizontal bar chart to visualize the bottom 10 crop producing districts
plt.barh(bottom_districts['District_Name'], bottom_districts['Production'])
plt.title('Least overall Crop Producing Districts')
plt.xlabel('Production (metric tons)')
plt.ylabel('District Name')
plt.show()
```

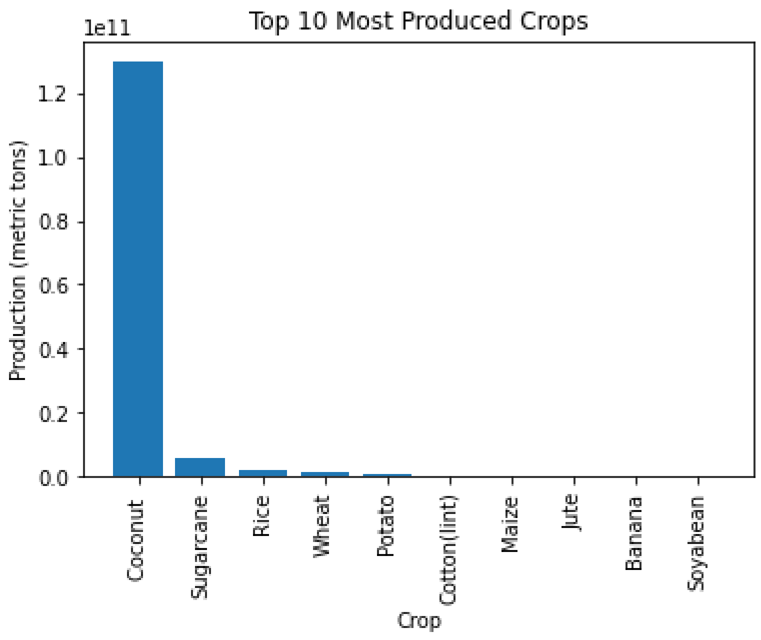




Top 10 "Most Produced" Crops

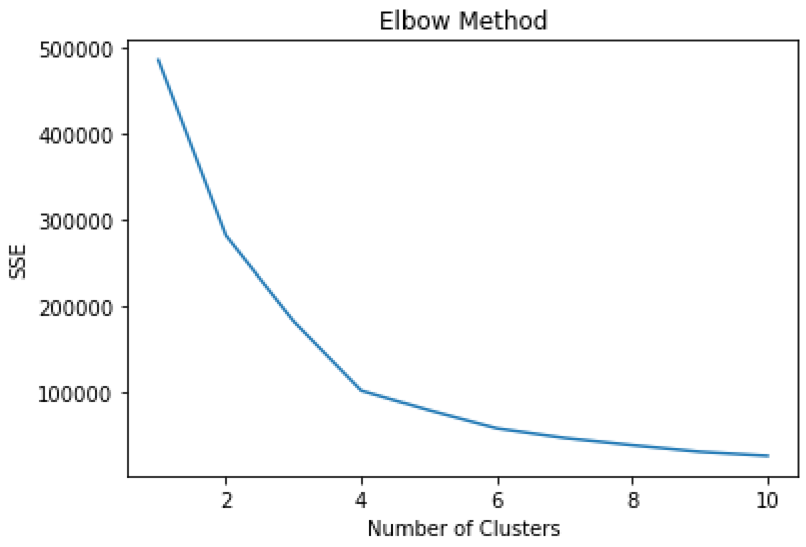
In [53]:

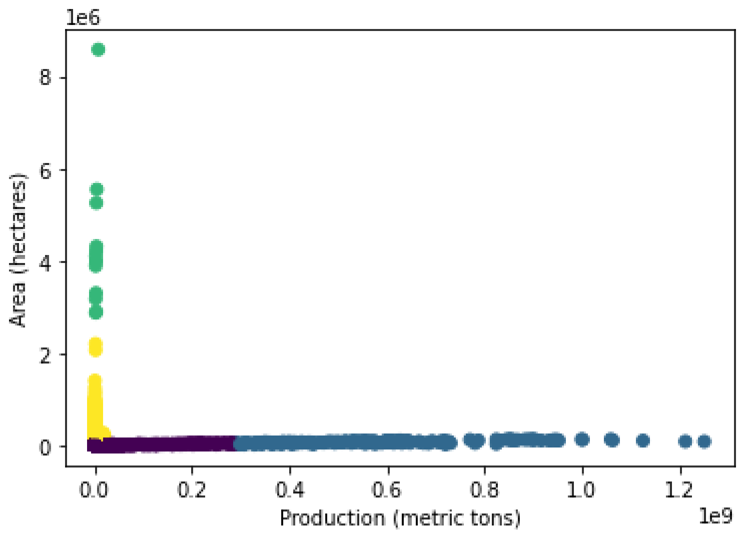
```
# Group the data by crop and sum the production
crop_production = df.groupby('Crop')['Production'].sum()
# Sort the results in descending order and take the top 10 crops
top_10_crops = crop_production.sort_values(ascending=False)[:10]
# Plot the top 10 crops
plt.bar(top_10_crops.index, top_10_crops.values)
plt.xticks(rotation=90)
plt.xlabel('Crop')
plt.ylabel('Production (metric tons)')
plt.title('Top 10 Most Produced Crops')
plt.show()
```



We can see from the above graph that Coconut,Sugarcane,Rice,Wheat and Potato are teh most produced crops. Coconut is at no. 1 among them.

In [84]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Drop rows with missing values
df.dropna(inplace=True)

# Select relevant features
X = df[['Production', 'Area']].values

# Normalize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Find the optimal number of clusters using elbow method
sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X_scaled)
    sse.append(kmeans.inertia_)

# Plot the elbow method
plt.plot(range(1, 11), sse)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('SSE')
plt.show()

# Choose the optimal number of clusters and fit the KMeans model
kmeans = KMeans(n_clusters=4)
kmeans.fit(X_scaled)

# Add cluster labels to the dataframe
df_clustered = df[['District_Name', 'Production', 'Area']].copy()
df_clustered['Cluster'] = kmeans.labels_

# Visualize the clusters
plt.scatter(df_clustered['Production'], df_clustered['Area'], c=df_clustered['Cluste
plt.xlabel('Production (metric tons)')
plt.ylabel('Area (hectares)')
plt.show()
```





The first plot shows the elbow method for determining the optimal number of clusters to use in the KMeans clustering algorithm. The x-axis represents the number of clusters, while the y-axis represents the sum of squared distances of data points from their assigned cluster centers. The idea behind the elbow method is to choose the number of clusters at the point where adding another cluster doesn't improve much more the overall quality of the clustering. In this case, the elbow point appears to be at around 3 or 4 clusters.

The second plot shows a scatter plot of district-wise crop production and area, with points colored based on their assigned cluster. The x-axis represents the total crop production for a district, while the y-axis represents the total area under cultivation. The different colors represent the different clusters that the districts were assigned to. This plot allows us to visually inspect how the districts are grouped together based on their crop production and area, and how the different clusters differ from each other.

```
In [ ]:
```

The above code performs clustering analysis on crop production data at the district level. Here is a step-by-step explanation of the code:

1) First, the necessary libraries are imported. These include pandas for data manipulation, numpy for numerical operations, seaborn and matplotlib for data visualization, and StandardScaler and KMeans for clustering analysis.

2) Next, the dataset is loaded into a pandas dataframe. The dataset contains information on crop production in various districts of India, including the district name, state, crop type, area under cultivation, and production.

3) The dataset is then filtered to include only the columns relevant for clustering analysis. These columns are 'District_Name', 'Crop', 'Area', and 'Production'.

4) The data is then grouped by district and crop, and the total area and production for each district-crop combination is calculated.

5) The resulting data is then pivoted such that each row represents a district, and the columns represent the crop types, with the values in the table indicating the total production of each crop in each district.

6) The data is then standardized using the StandardScaler() function to ensure that all features have similar ranges and are on the same scale.

7) The optimal number of clusters is determined by iterating over different values of k (the number of clusters), and calculating the sum of squared errors (SSE) for each value of k. The 'elbow method' is used to determine the optimal number of clusters, which is the point at which the decrease in SSE begins to level off.

8) The KMeans() function is then used to create a clustering model with the optimal number of clusters. The model is fitted to the standardized data.

9) Finally, the cluster labels are added to the original dataframe, and a heatmap is plotted to visualize the clusters. The heatmap shows the total production of each crop in each district, with each district colored according to its cluster label.

In summary, this code performs clustering analysis on crop production data at the district level, grouping districts based on their crop production patterns. This analysis can provide insights into the similarities and differences between districts in terms of crop production, which can be useful for policymakers, researchers, and other stakeholders.

In [62]:
```
# I am creating Different zones (Union Terr, South Zone, NE Zone, East Zone, North Z
north_india = ['Jammu and Kashmir', 'Punjab', 'Himachal Pradesh', 'Haryana', 'Uttara
east_india = ['Bihar', 'Odisha', 'Jharkhand', 'West Bengal']
south_india = ['Andhra Pradesh', 'Karnataka', 'Kerala' ,'Tamil Nadu', 'Telangana']
west_india = ['Rajasthan' , 'Gujarat', 'Goa','Maharashtra']
central_india = ['Madhya Pradesh', 'Chhattisgarh']
north_east_india = ['Assam', 'Sikkim', 'Nagaland', 'Meghalaya', 'Manipur', 'Mizoram'
ut_india = ['Andaman and Nicobar Islands', 'Dadra and Nagar Haveli', 'Puducherry']
```

In [66]:
```
def get_zonal_names(row):
    state_name = row['State_Name'].strip()
    if state_name in north_india:
        val = 'North Zone'
    elif state_name in south_india:
        val = 'South Zone'
    elif state_name in east_india:
        val = 'East Zone'
    elif state_name in west_india:
        val = 'West Zone'
    elif state_name in central_india:
        val = 'Central Zone'
    elif state_name in north_east_india:
        val = 'NE Zone'
    elif state_name in ut_india:
        val = 'Union Terr'
    else:
        val = 'No Value'
    return val

df['Zones'] = df.apply(get_zonal_names, axis=1)
df['Zones'].unique()
```

Out[66]:

```
array(['Union Terr', 'South Zone', 'NE Zone', 'East Zone', 'North Zone',
       'Central Zone', 'West Zone'], dtype=object)
```

The zones are defined based on the state names provided in the lists north_india, south_india, east_india, west_india, central_india, north_east_india, and ut_india. These lists contain the names of states that belong to each zone.

The function checks the State_Name column of the input row against each of the zone lists using the in operator and returns the corresponding zone name as a string value. If the state name does not belong to any of the zones, the function returns "No Value".

Finally, the Zones column of the DataFrame is created by applying the get_zonal_names function to each row of the DataFrame and storing the returned values. The unique values of the Zones column are then displayed using the unique() method. This provides a list of all the unique zone names that are present in the DataFrame.

In [67]:

```
df.Zones.value_counts()
```

Out[67]:

```
South Zone      53500
North Zone      49874
East Zone       43261
West Zone       33134
Central Zone    32972
NE Zone         28284
Union Terr       1336
Name: Zones, dtype: int64
```

Top 3 zones are South India, North India and East India.

In [68]:

```
# Zonal Distribution of crops
df.head()
```

Out[68]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Zones
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0	Union Terr
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0	Union Terr
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0	Union Terr

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Zones
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0	Union Terr
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0	Union Terr

In [70]:
```
def crop_category(crop):
    for i in ['Rice','Maize','Wheat','Barley','Varagu','Other Cereals & Millets','Ra
        if crop == i:
            return 'Cereal'
    for i in ['Moong','Urad','Arhar/Tur','Peas & beans','Masoor', 'Other Kharif puls
        if crop == i:
            return 'Pulses'
    for i in ['Peach','Apple','Litchi','Pear','Plums','Ber','Sapota','Lemon','Pome G
        if crop == i:
            return 'Fruits'
    for i in ['Bean','Lab-Lab','Moth','Guar seed','Soyabean','Horse-gram']:
        if crop == i:
            return 'Beans'
    for i in ['Turnip','Peas','Beet Root','Carrot','Yam','Ribed Guard','Ash Gourd ',
        if crop == i:
            return 'Vegetables'
    for i in ['Perilla','Ginger','Cardamom','Black pepper','Dry ginger','Garlic','Co
        if crop == i:
            return 'Spices'
    for i in ['other fibres','Kapas','Jute & mesta','Jute','Mesta','Cotton(lint)','S
        if crop == i:
            return 'Fibres'
    for i in ['Arcanut (Processed)','Atcanut (Raw)','Cashewnut Processed','Cashewnut
        if crop == i:
            return 'Nuts'
    for i in ['other oilseeds','Safflower','Niger seed','Castor seed','Linseed','Sun
        if crop == i:
            return 'Oilseeds'
    for i in ['Tobacco','Coffee','Tea','Sugarcane','Rubber']:
        if crop == i:
            return 'Commercial'
    return 'No Category'

df['crop_category'] = df['Crop'].apply(crop_category)
```

In [72]:
```
data_explore = df.copy()
data_explore.head()
```

Out[72]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Zones	crop_categ
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0	Union Terr	N
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0	Union Terr	Pu
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0	Union Terr	Ce
3	Andaman and Nicobar	NICOBARS	2000	Whole Year	Banana	176.0	641.0	Union Terr	Fr

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production	Zones	crop_categ
	Islands								
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0	Union Terr	N

This code creates a copy of the original dataframe df and assigns it to a new variable data_explore.

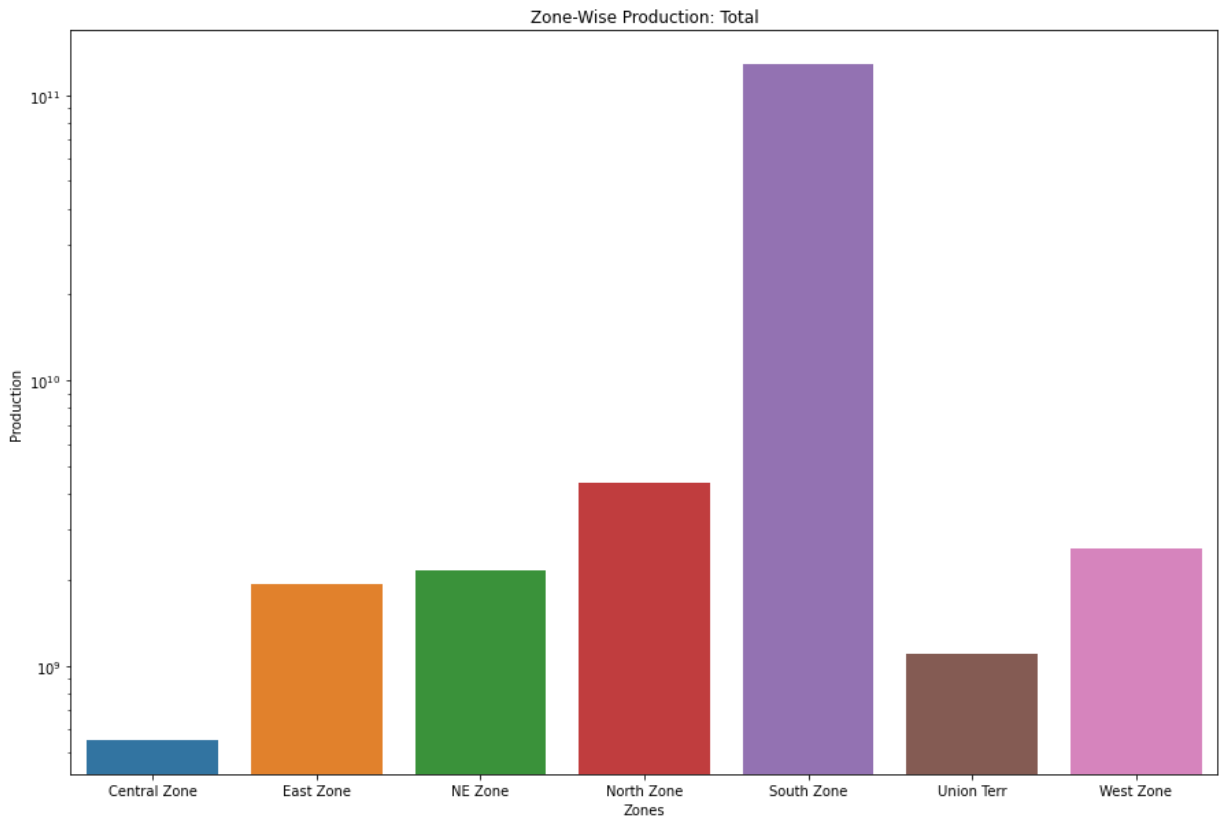
Creating a copy of the original dataframe is a good practice because it allows us to perform any exploratory data analysis or data manipulation without affecting the original dataset. This way, if we make any mistakes or want to start fresh, we can simply go back to the original dataset.

By doing this, any changes made to data_explore will not affect the original dataframe df. This helps to keep our analysis clean and organized.

Zonal distribution of crops

In [73]:
```
fig, ax = plt.subplots(figsize=(15,10))
sns.barplot(x='Zones', y='Production', data=data_explore.groupby('Zones').sum().rese
plt.yscale('log')
plt.title('Zone-Wise Production: Total')
```

Out[73]:
```
Text(0.5, 1.0, 'Zone-Wise Production: Total')
```

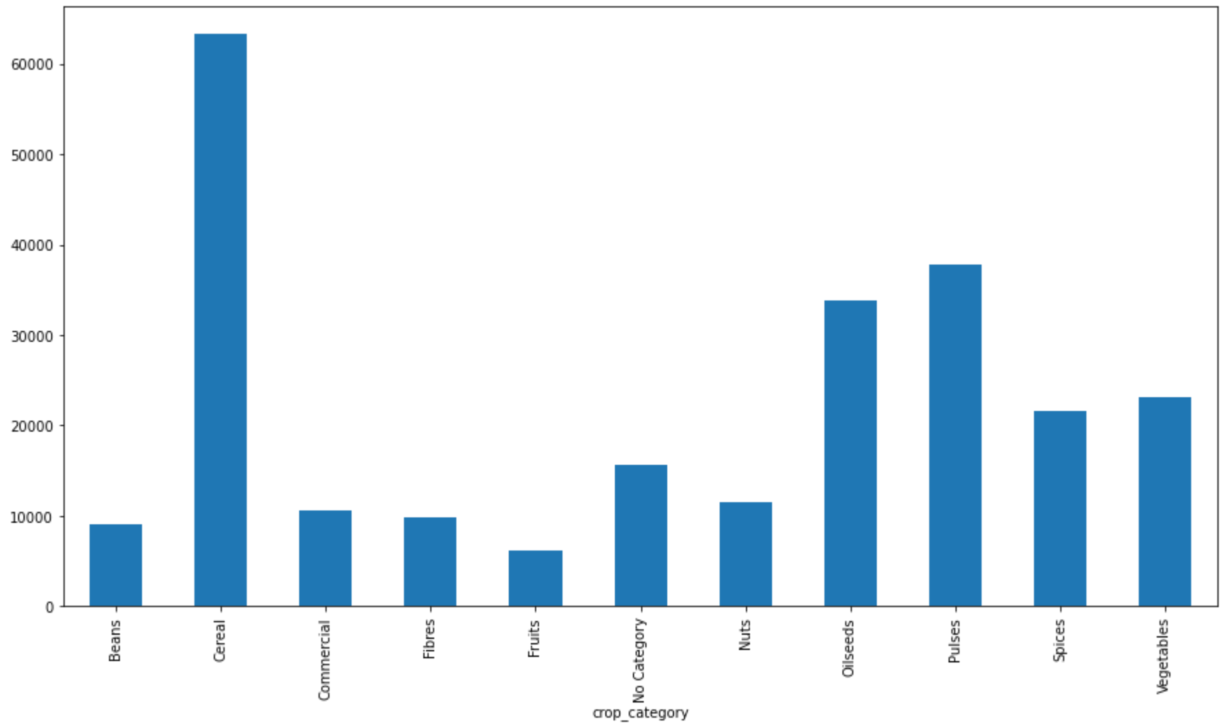


The x-axis shows the zones and the y-axis shows the production of crops in logarithmic scale. The bars represent the total production of crops in each zone. The title of the chart is "Zone-Wise Production: Total".

The chart shows that the North Zone has the highest production of crops, followed by the East and West zones. The Central and South zones have lower production compared to the other zones. The chart also shows that there are some outliers in the data, as some zones have significantly higher production compared to others.

Crop wise Production plot describing production values for all crop types.

In [74]:
```
plt.figure(figsize=(15,8))
plt.tick_params(labelsize=10)
data_explore.groupby("crop_category")["Production"].agg("count").plot.bar()
plt.show()
```

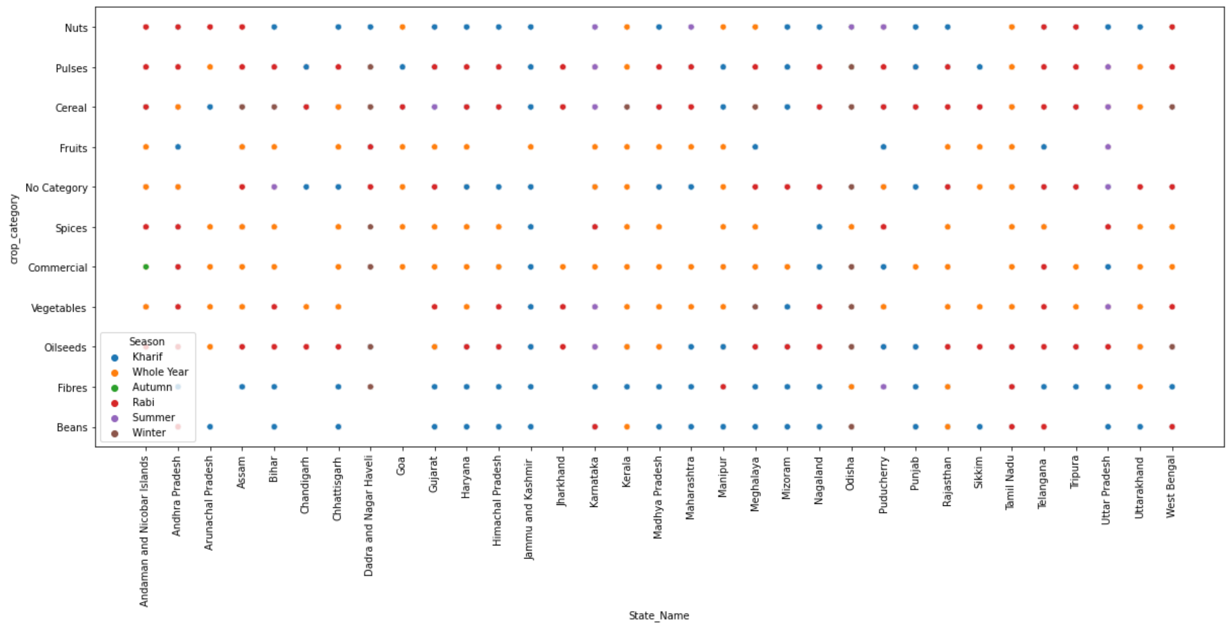


1) plt.figure(figsize=(15,8)) sets the size of the plot figure to (15,8). 2) plt.tick_params(labelsize=10) increases the font size of the axis ticks to 10. data_explore.groupby("crop_category")["Production"].agg("count") groups the dataset by crop_category and counts the number of production values for each group. 3) plot.bar() creates a bar plot for the grouped data. 4) plt.show() displays the plot. So, the plot shows the count of productions for each crop category. It gives an idea of how many productions are there for each category, which can be useful in understanding the distribution of data.

State versus Crop Category versus Season plot

In [75]:
```
plt.figure(figsize=(20,8))
sns.scatterplot(data=data_explore,x="State_Name",y="crop_category",hue="Season")
plt.xticks(rotation=90)
plt.show()
```

```
C:\Users\Prajakta Bose\anaconda3\lib\site-packages\IPython\core\pylabtools.py:151: Us
erWarning: Creating legend with loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



The scatter plot shows the distribution of different crop categories across different states, with different seasons represented by different colors. The x-axis represents the states, the y-axis represents the crop categories, and the colors represent different seasons. The plt.xticks(rotation=90) function is used to rotate the x-axis labels by 90 degrees to make them more readable. The plt.figure(figsize=(20,8)) function is used to adjust the size of the plot. Overall, this plot can be useful for identifying patterns in the distribution of different crop categories across different states and seasons.

In [ ]:

```
```

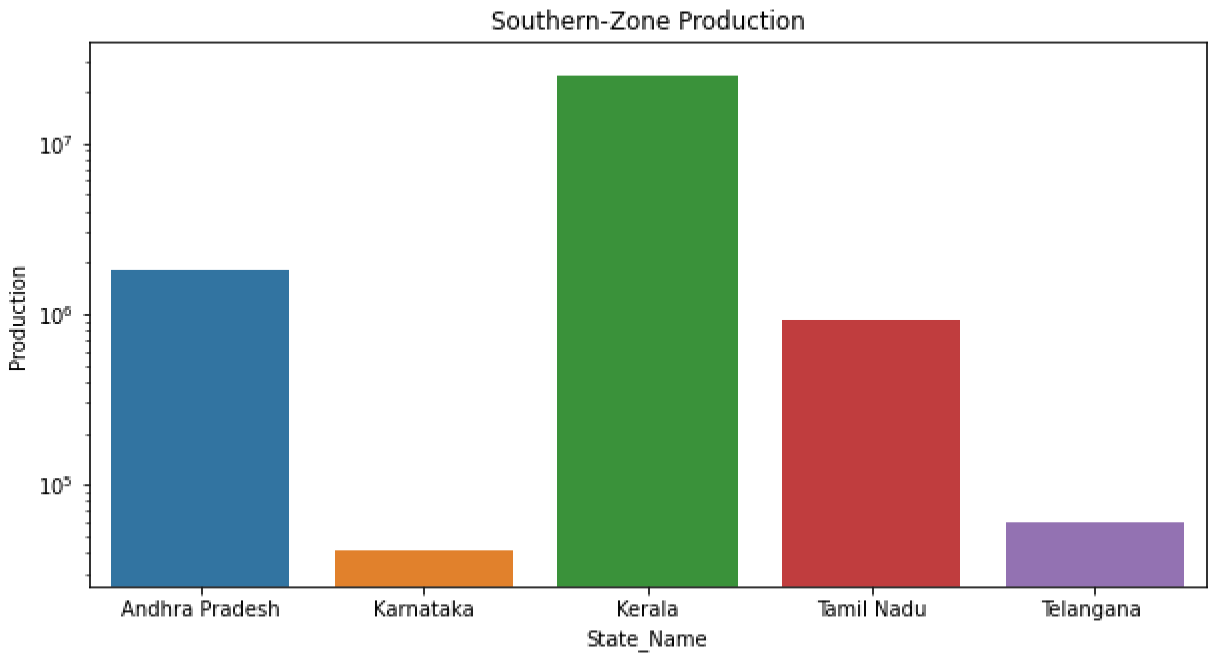
Production wise top zone is South India

In [76]:

```
south_zone = data_explore[data_explore["Zones"] == 'South Zone']
fig, ax = plt.subplots(figsize=(10,5))
sns.barplot(x=south_zone.State_Name, y=south_zone.Production, errwidth=0)
plt.yscale('log')
plt.title('Southern-Zone Production')
```

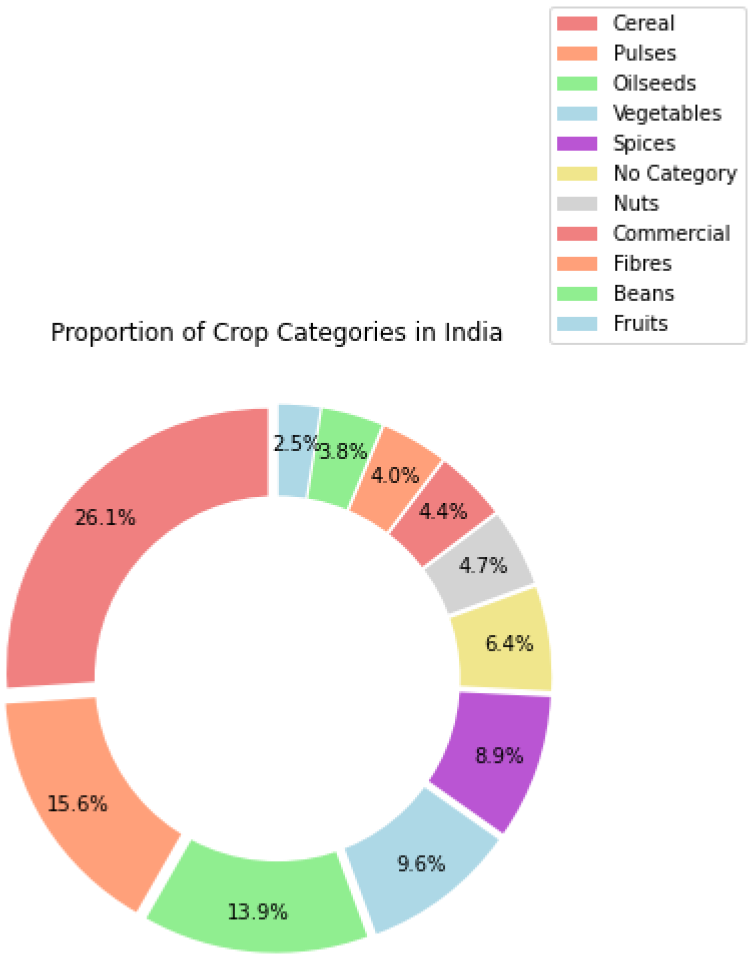
Out[76]:

```
Text(0.5, 1.0, 'Southern-Zone Production')
```



Different proportion of Crop Categories for India

```
In [78]:
# Create a new dataframe with the crop category counts
crop_counts = data_explore['crop_category'].value_counts()
# Plot a donut chart to show the proportion of each crop category
fig, ax = plt.subplots(figsize=(6, 6))
# Define colors for the chart
colors = ['#F08080', '#FFA07A', '#90EE90', '#ADD8E6', '#BA55D3', '#F0E68C', '#D3D3D3
# Create the chart
ax.pie(crop_counts, colors=colors, autopct='%1.1f%%', startangle=90, pctdistance=0.8
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig.gca().add_artist(centre_circle)
# Add a legend
ax.legend(labels=crop_counts.index, loc='best', bbox_to_anchor=(0.9, 1))
# Add a title
ax.set_title('Proportion of Crop Categories in India')
#Show the chart
plt.show()
```



Which State dominates in crop production with different categories of crops?

In [80]:
```
# Group by state and crop category and sum up the production values
state_crop_production = data_explore.groupby(['State_Name', 'crop_category'])['Produ

# Find the state with maximum production in each crop category
max_production_by_category = state_crop_production.groupby('crop_category').idxmax()

# Print the results for category
for category, (state, _) in max_production_by_category.iteritems():
    print(f"{state} dominates in {category} production")
```

```
Madhya Pradesh dominates in Beans production
Uttar Pradesh dominates in Cereal production
Uttar Pradesh dominates in Commercial production
West Bengal dominates in Fibres production
Tamil Nadu dominates in Fruits production
Kerala dominates in No Category production
Gujarat dominates in Nuts production
West Bengal dominates in Oilseeds production
Madhya Pradesh dominates in Pulses production
Karnataka dominates in Spices production
Uttar Pradesh dominates in Vegetables production
```

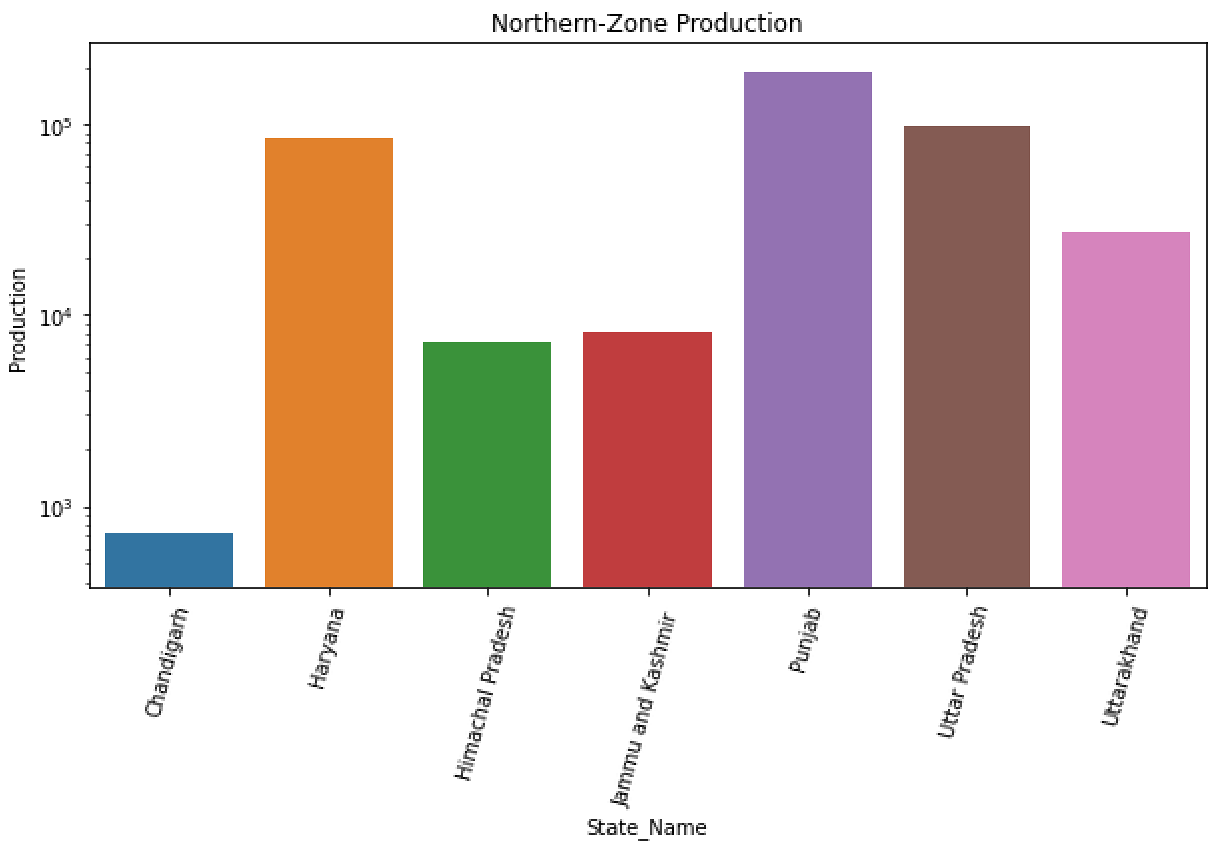
In [ ]:

Which are the top crops grown in Northern parts of India?

In [81]:
```
data_explore.Zones.value_counts()
```

Out[81]:
```
South Zone      53500
North Zone      49874
East Zone       43261
West Zone       33134
Central Zone    32972
NE Zone         28284
Union Terr       1336
Name: Zones, dtype: int64
```

In [82]:
```
# Filter data for North Zone
North_zone = data_explore[data_explore["Zones"] == 'North Zone']

# Create bar plot for production in North Zone
fig, ax = plt.subplots(figsize=(10, 5))
sns.barplot(x=North_zone['State_Name'], y=North_zone['Production'], errwidth=0)
plt.xticks(rotation=75)
plt.yscale('log')
plt.title('Northern-Zone Production')
plt.show()

# Group by state name and sum production values
top_crops = North_zone.groupby(by='State_Name')['Production'].sum().reset_index().so
print(top_crops)
```



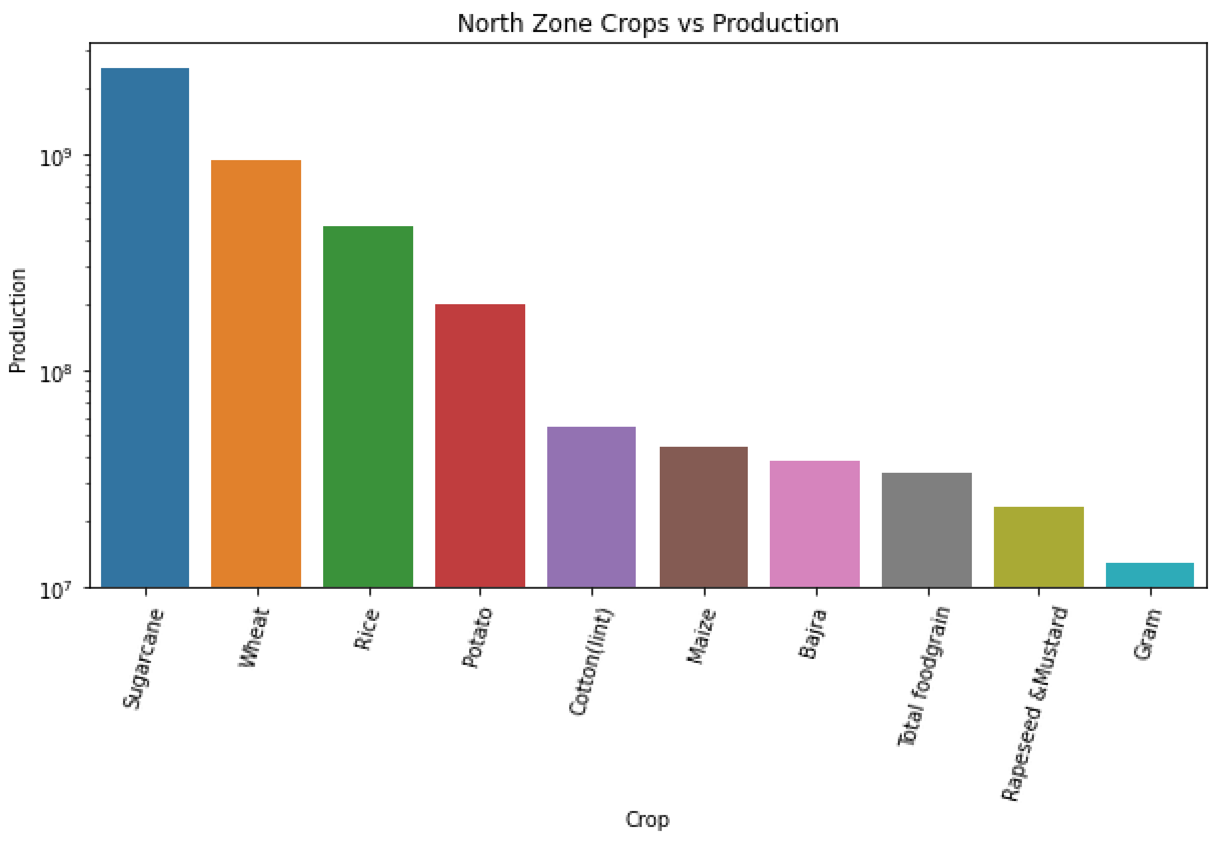
```
          State_Name    Production
5      Uttar Pradesh  3.234493e+09
4             Punjab  5.863850e+08
1            Haryana  3.812739e+08
6        Uttarakhand  1.321774e+08
2   Himachal Pradesh  1.780517e+07
3  Jammu and Kashmir  1.329102e+07
0         Chandigarh  6.395650e+04
```

In [83]:
```
df3 = North_zone.groupby(by='Crop')['Production'].sum().reset_index().sort_values(by
fig, ax = plt.subplots(figsize=(10,5))
sns.barplot(x=df3.Crop, y=df3.Production,errwidth=0)
```

```
plt.xticks(rotation=75); plt.yscale('log')
plt.title('North Zone Crops vs Production')
```

Out[83]:

```
Text(0.5, 1.0, 'North Zone Crops vs Production')
```



In [ ]: