Data Visualization on Honey Production dataset using seaborn and matplotlib libraries.

SUBMITTED BY:

NITHISHWAR

1. Import required libraries and read the dataset.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df= pd.read_csv(r"C:\Users\Nithish\Desktop\GL\Python\week 4\week 4 Graded project\Datasets\honeyproduction.csv")
df
```

Out[3]:

	state	numcol	yieldpercol	totalprod	stocks	priceperlb	prodvalue	year
0	AL	16000.0	71	1136000.0	159000.0	0.72	818000.0	1998
1	AZ	55000.0	60	3300000.0	1485000.0	0.64	2112000.0	1998
2	AR	53000.0	65	3445000.0	1688000.0	0.59	2033000.0	1998
3	CA	450000.0	83	37350000.0	12326000.0	0.62	23157000.0	1998
4	CO	27000.0	72	1944000.0	1594000.0	0.70	1361000.0	1998
...	...	...	...	...	...	...	...	...
621	VA	4000.0	41	164000.0	23000.0	3.77	618000.0	2012
622	WA	62000.0	41	2542000.0	1017000.0	2.38	6050000.0	2012
623	WV	6000.0	48	288000.0	95000.0	2.91	838000.0	2012
624	WI	60000.0	69	4140000.0	1863000.0	2.05	8487000.0	2012
625	WY	50000.0	51	2550000.0	459000.0	1.87	4769000.0	2012

626 rows × 8 columns

2. Check the first few samples, shape, info of the data and try to familiarize yourself with different features.

In [4]:
```
df.head(10)  #---First few samples of data frame
```

Out[4]:

	state	numcol	yieldpercol	totalprod	stocks	priceperlb	prodvalue	year
0	AL	16000.0	71	1136000.0	159000.0	0.72	818000.0	1998
1	AZ	55000.0	60	3300000.0	1485000.0	0.64	2112000.0	1998
2	AR	53000.0	65	3445000.0	1688000.0	0.59	2033000.0	1998
3	CA	450000.0	83	37350000.0	12326000.0	0.62	23157000.0	1998
4	CO	27000.0	72	1944000.0	1594000.0	0.70	1361000.0	1998
5	FL	230000.0	98	22540000.0	4508000.0	0.64	14426000.0	1998
6	GA	75000.0	56	4200000.0	307000.0	0.69	2898000.0	1998
7	HI	8000.0	118	944000.0	66000.0	0.77	727000.0	1998
8	ID	120000.0	50	6000000.0	2220000.0	0.65	3900000.0	1998
9	IL	9000.0	71	639000.0	204000.0	1.19	760000.0	1998

In [5]:
```
df.shape   #---It has 626 rows and 8 columns
```

Out[5]:
```
(626, 8)
```

In [6]:
```
df.info()   #---Pulled all the info of the current Dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 626 entries, 0 to 625
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   state        626 non-null    object
 1   numcol       626 non-null    float64
 2   yieldpercol  626 non-null    int64
 3   totalprod    626 non-null    float64
 4   stocks       626 non-null    float64
 5   priceperlb   626 non-null    float64
 6   prodvalue    626 non-null    float64
 7   year         626 non-null    int64
dtypes: float64(5), int64(2), object(1)
memory usage: 39.3+ KB
```

In [7]:
```
df.columns   #---This dataframe has various type of features(columns), will analyze each and everything
```

Out[7]:
```
Index(['state', 'numcol', 'yieldpercol', 'totalprod', 'stocks', 'priceperlb',
       'prodvalue', 'year'],
      dtype='object')
```

3. Display the percentage distribution of the data in each year using the pie chart.

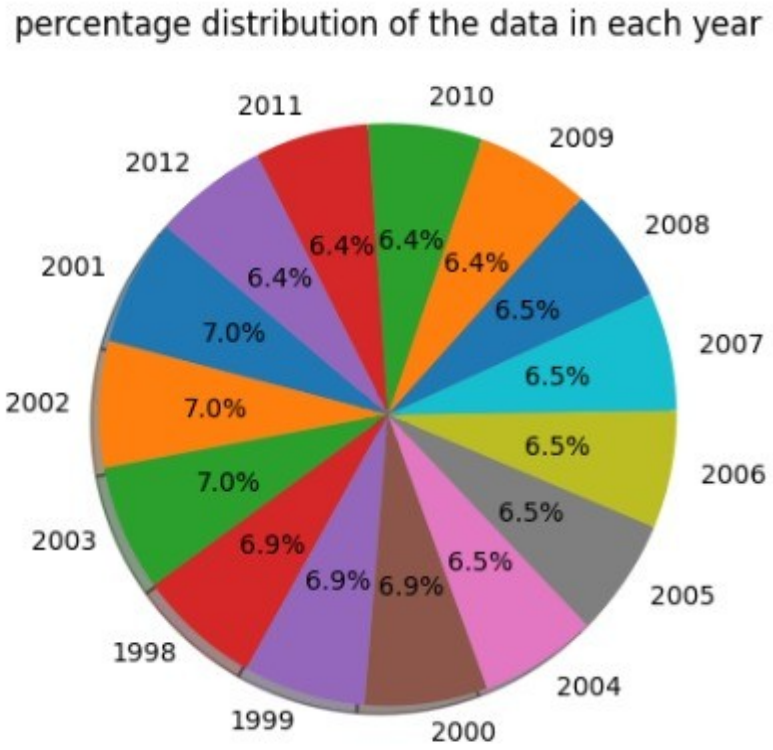
In [8]:
```
percentage_distribution_year = df['year'].value_counts(normalize=True)*100
percentage_distribution_year

#--Calculating the percentage distribution of the data in each year
```

Out[8]:
```
year
2001    7.028754
2002    7.028754
2003    7.028754
1998    6.869010
1999    6.869010
2000    6.869010
2004    6.549521
2005    6.549521
2006    6.549521
2007    6.549521
2008    6.549521
2009    6.389776
2010    6.389776
2011    6.389776
2012    6.389776
Name: proportion, dtype: float64
```

The normalize=True argument is passed to calculate the relative frequencies of each unique value. This means it calculates the proportion of each year in the total dataset. This results in a Series where the indices are the unique years and the values are the corresponding relative frequencies (in decimal form).

In [40]:

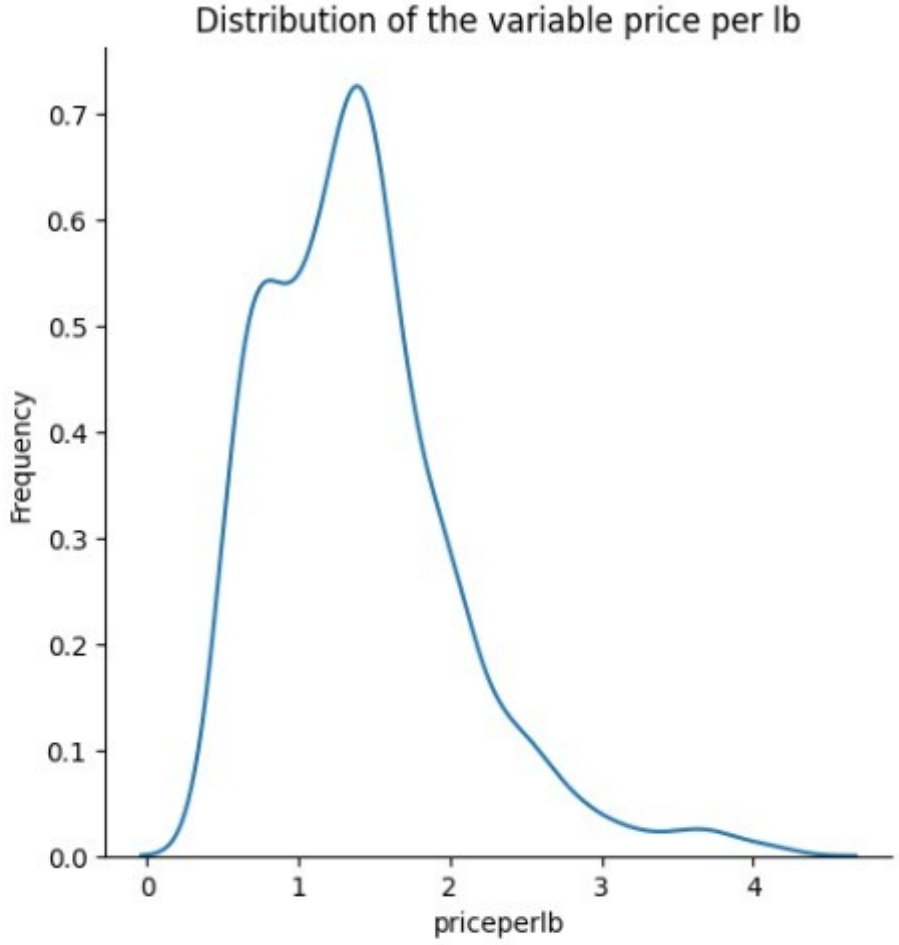
```
#---Plotting using pie chart

plt.figure(figsize=(10,5))
plt.pie(percentage_distribution_year, labels=percentage_distribution_year.index, autopct='%1.1f%%',startangle=140, shadow=Tr
plt.title('percentage distribution of the data in each year')
plt.show()
```



4. Plot and Understand the distribution of the variable "price per lb" using displot, and write your findings.

In [42]:
```
#---Plotting 'Distribution of the variable priceperlb' using displot

sns.displot(data=df, x=df['priceperlb'], kind='kde')
plt.title('Distribution of the variable priceperlb')
plt.xlabel('priceperlb')
plt.ylabel('Frequency')
plt.show()
```

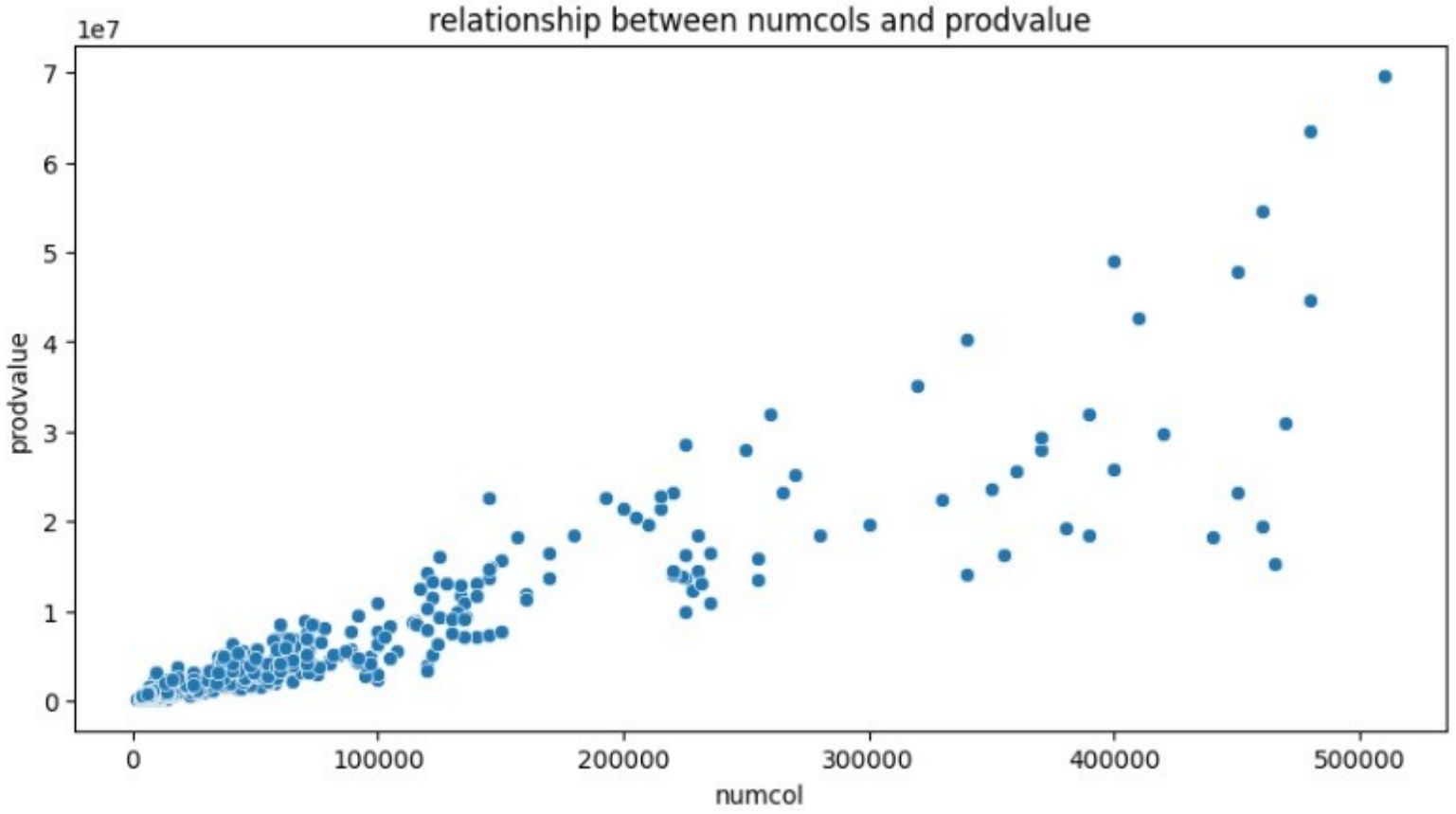
```
C:\Users\Nithish\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure lay
out has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [13]:
```
# - Findings:

#-- The distribution is right-skewed, indicating that there are fewer data points with higher prices per pound.
#-- The majority of the prices per pound are concentrated on the lower end of the scale.
#-- There are a few instances of relatively higher prices per pound, but they are not as common.


```

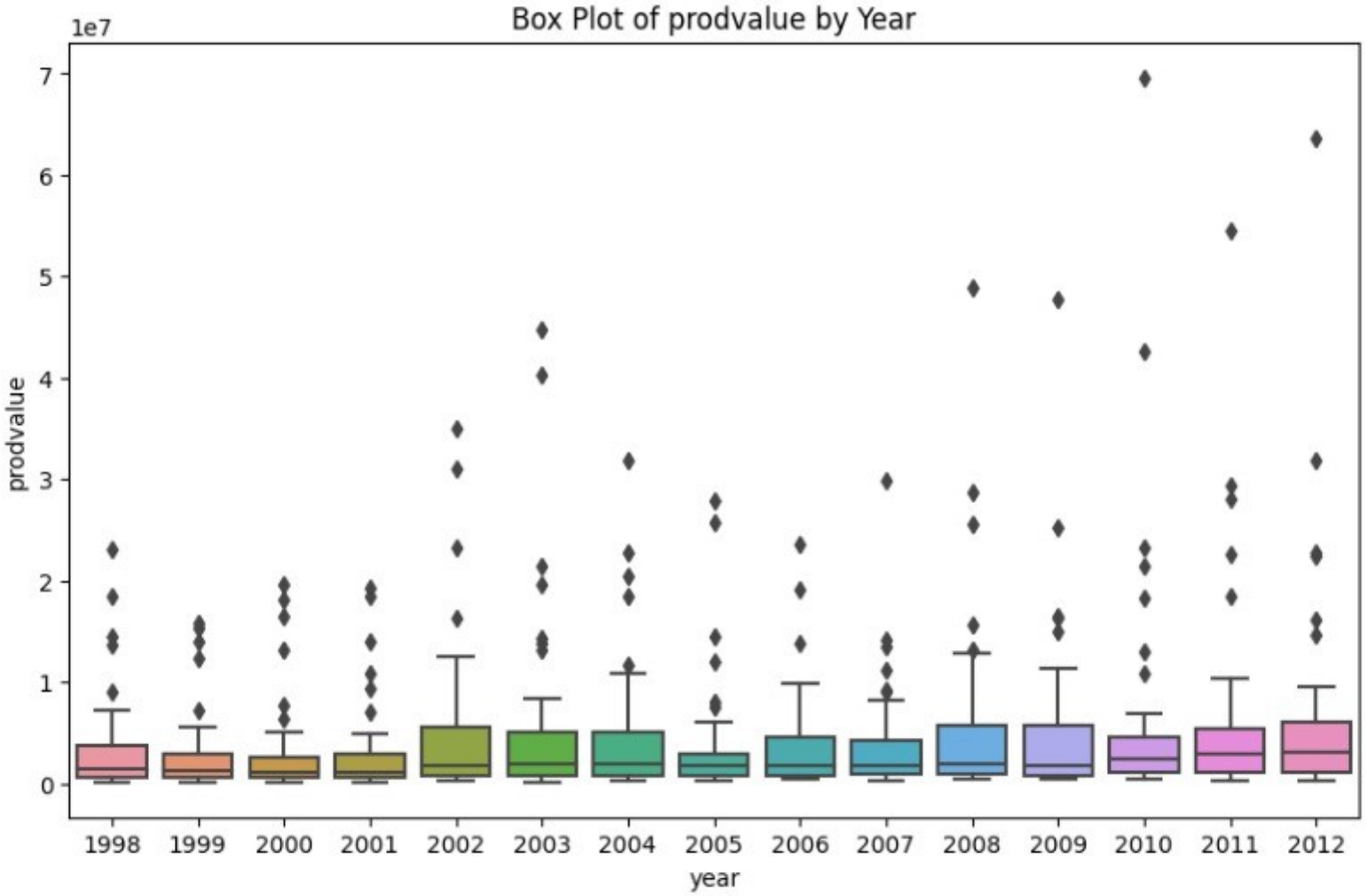
5. Plot and understand the relationship between the variables 'numcol' and 'prodval' through scatterplot, and write your findings.

In [14]:
```
#---Plotting the relationship between the variables 'numcol' and 'prodval' using scatterplot

plt.figure(figsize=(10,5))
sns.scatterplot(data=df, x=df['numcol'], y=df['prodvalue'])
plt.title('relationship between numcols and prodvalue')
plt.show()
```



6. Plot and understand the relationship between categorical variable 'year' and a numerical variable 'prodvalue' through boxplot, and write your findings.

In [16]:
```
#---Plotting the relationship between categorical variable 'year' and a numerical variable 'prodvalue' using boxplot

plt.figure(figsize=(10,6))
sns.boxplot(data=df, x=df['year'], y=df['prodvalue'])
plt.title('Box Plot of prodvalue by Year')
plt.xlabel('year')
plt.ylabel('prodvalue')
plt.show()
```



7. Visualize and understand the relationship between the multiple pairs of variables throughout different years using pairplot and add your inferences. (use columns 'numcol', 'yield percol', 'total prod', 'prodvalue','year')

In [18]:
```
df
```

Out[18]:

	state	numcol	yieldpercol	totalprod	stocks	priceperlb	prodvalue	year
0	AL	16000.0	71	1136000.0	159000.0	0.72	818000.0	1998
1	AZ	55000.0	60	3300000.0	1485000.0	0.64	2112000.0	1998
2	AR	53000.0	65	3445000.0	1688000.0	0.59	2033000.0	1998
3	CA	450000.0	83	37350000.0	12326000.0	0.62	23157000.0	1998
4	CO	27000.0	72	1944000.0	1594000.0	0.70	1361000.0	1998
...	...	...	...	...	...	...	...	...
621	VA	4000.0	41	164000.0	23000.0	3.77	618000.0	2012
622	WA	62000.0	41	2542000.0	1017000.0	2.38	6050000.0	2012
623	WV	6000.0	48	288000.0	95000.0	2.91	838000.0	2012
624	WI	60000.0	69	4140000.0	1863000.0	2.05	8487000.0	2012
625	WY	50000.0	51	2550000.0	459000.0	1.87	4769000.0	2012

626 rows × 8 columns

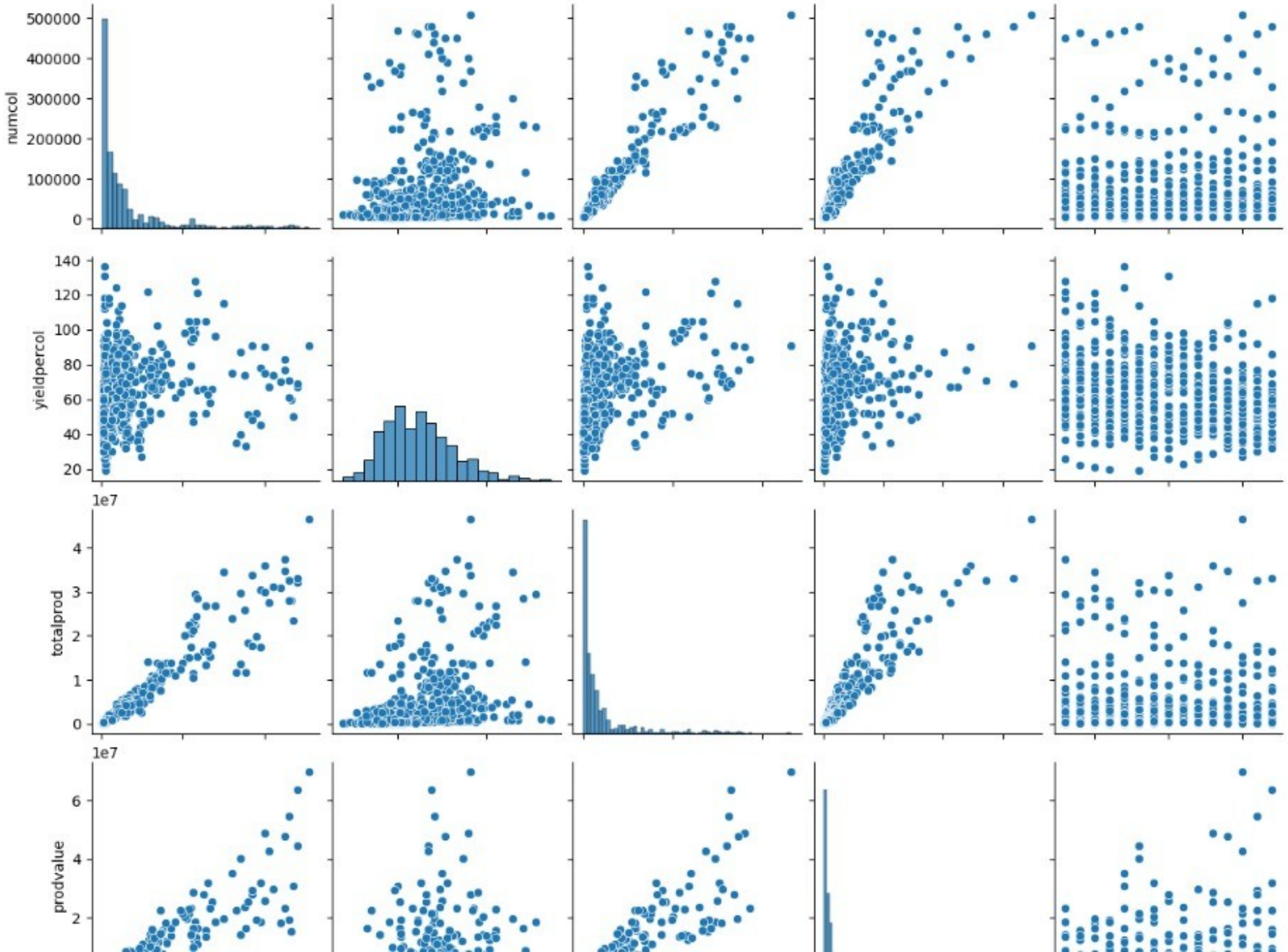
In [19]:
```
columns = ['numcol','yieldpercol','totalprod','prodvalue','year']
columns

#--Columns that needs to be paired
```

Out[19]:
```
['numcol', 'yieldpercol', 'totalprod', 'prodvalue', 'year']
```

In [20]:
```
1 sns.pairplot(df[columns])
```

```
C:\Users\Nithish\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure lay
out has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

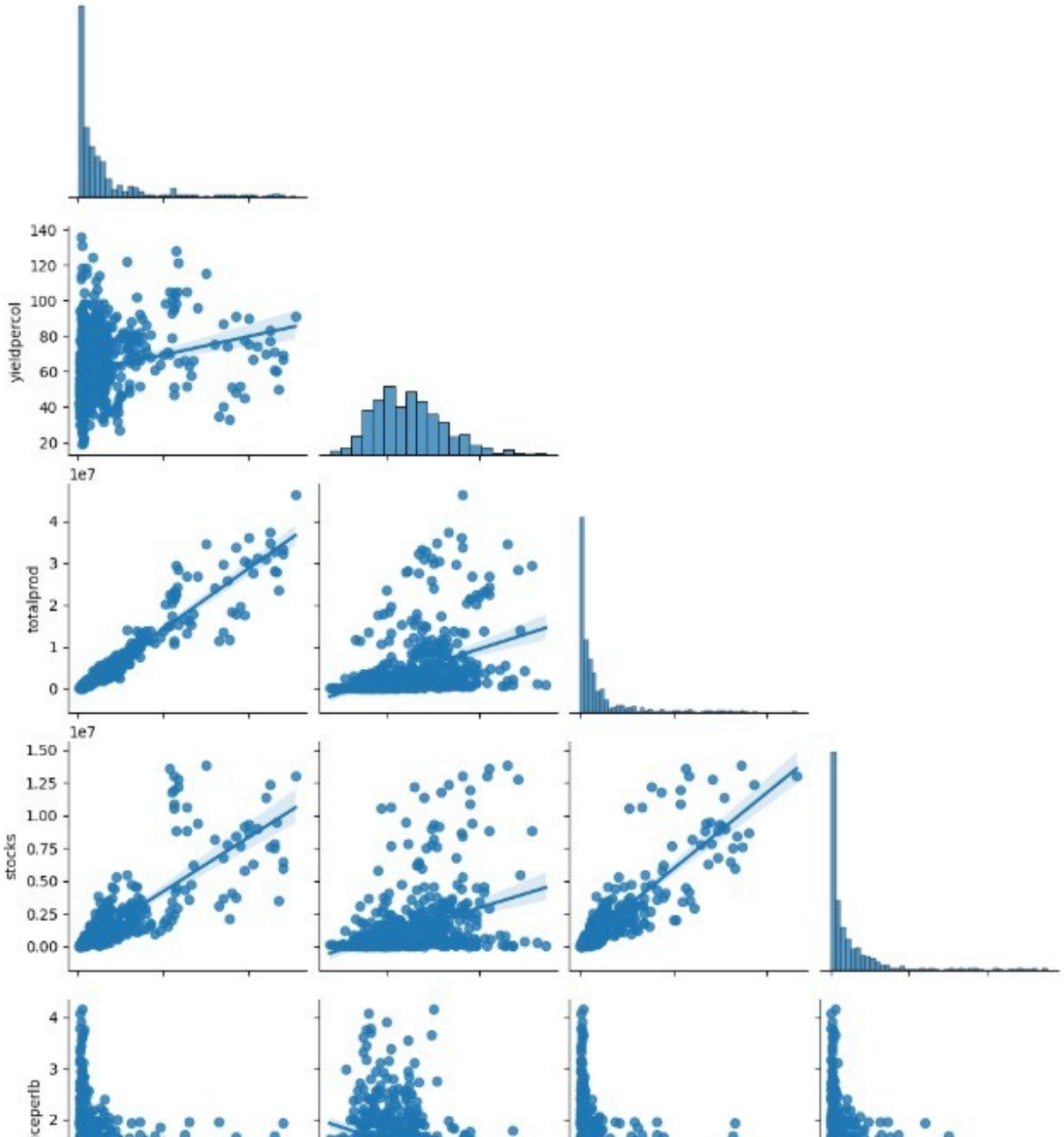
Out[20]: <seaborn.axisgrid.PairGrid at 0x1d531a80250>



In [38]:
```
1 sns.pairplot(df[columns], kind='reg', corner=True)    #---to get regression kind of plots
```

```
C:\Users\Nithish\AppData\Local\Programs\Python\Python311\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure lay
out has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[38]: <seaborn.axisgrid.PairGrid at 0x1d53cb84850>

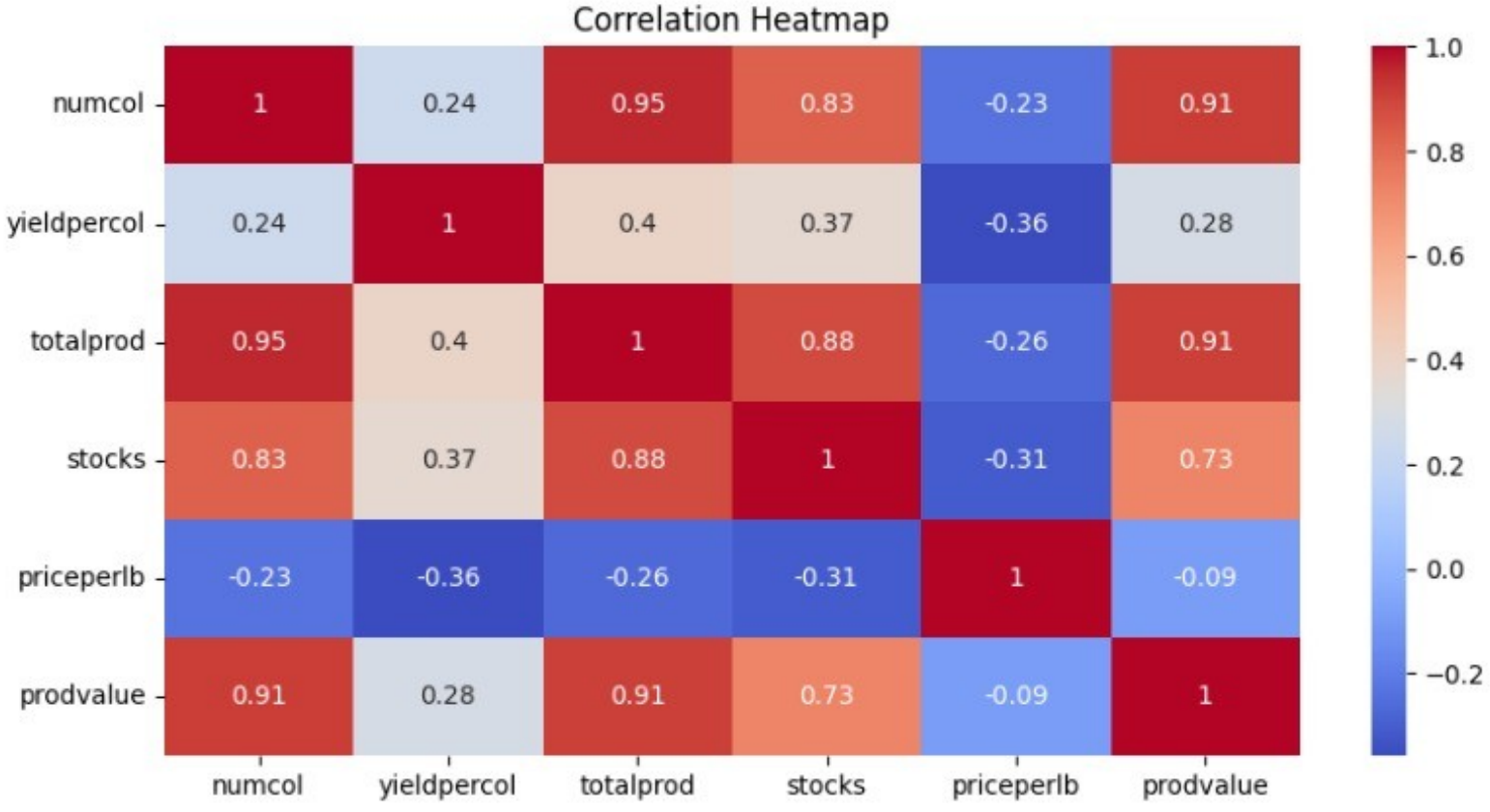


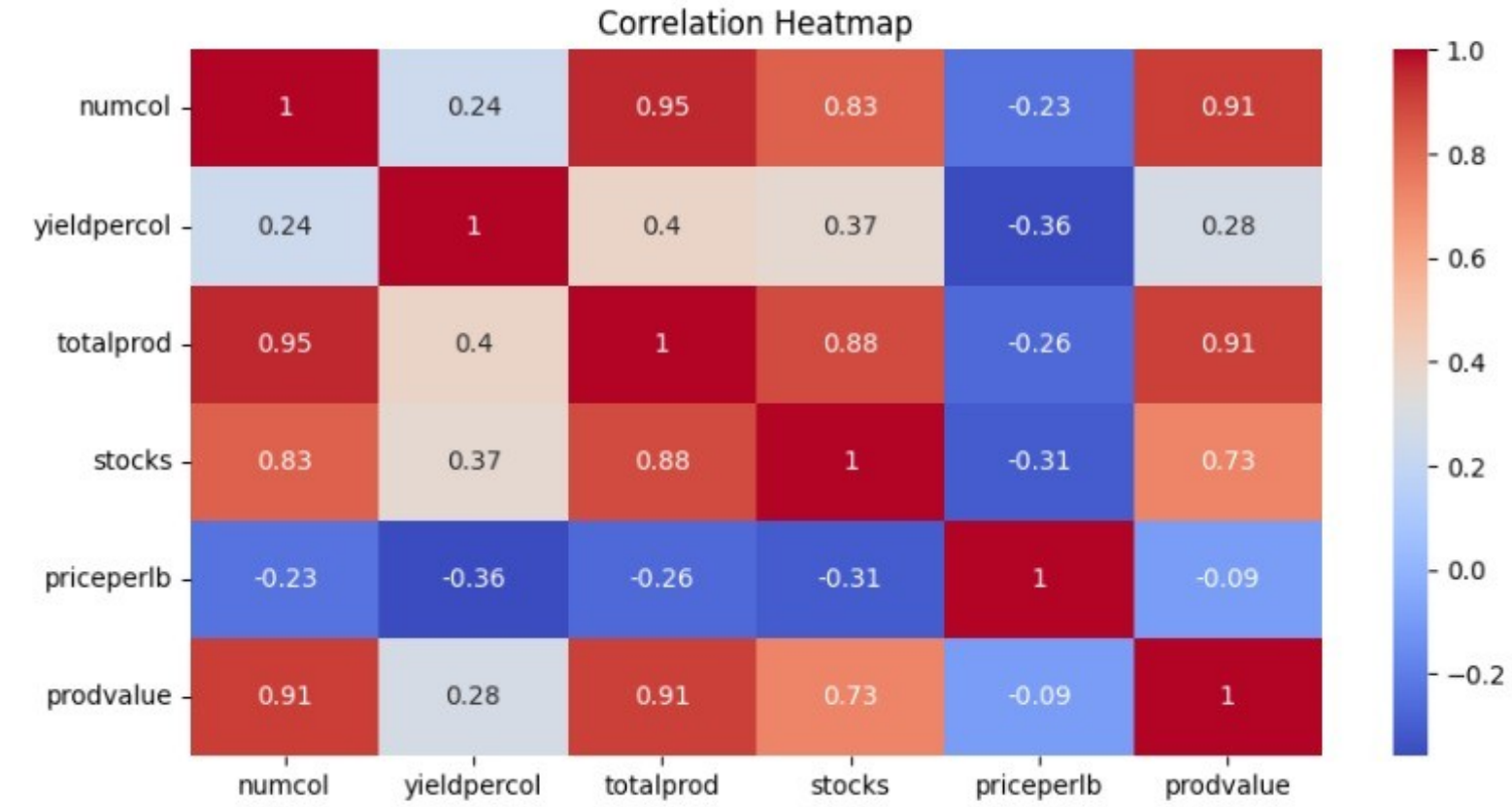
8. Display the correlation values using a plot and add your inferences. (use columns 'numcol', 'yield percol', 'total prod', 'stocks', 'price per lb', 'prodvalue')

In [22]:
```
columns = ['numcol', 'yieldpercol', 'totalprod', 'stocks', 'priceperlb', 'prodvalue']
columns

#---Columns that needs to be correlated
```

Out[22]:
```
['numcol', 'yieldpercol', 'totalprod', 'stocks', 'priceperlb', 'prodvalue']
```

In [36]:
```
#--Plotting a heatmap of the correlation matrix

plt.figure(figsize=(10,5))
sns.heatmap(data=df[columns].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```





In [ ]:

```
# --Inferences:

#-- The color scale in the heatmap indicates the strength and direction of correlations. Positive correlations are shown in

#-- Strong positive correlations are observed between 'totalprod' and 'numcol', as well as between 'prodvalue' and 'totalpro

#-- There is a moderate negative correlation between 'priceperlb' and 'prodvalue'. This indicates that as the price per poun

#-- Other correlations appear to be relatively weak, suggesting that the variables are less strongly related.
```

In [ ]:

```

```