EDA + DPL PROJECT BY: V KAVYASRI (22070126129) AHMED QASEM(22070126137)

Show code

Importing the required libraries.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the dataset

```
df=pd.read_excel('/content/Crop Yield Analysis.xls')
df
```

	States	Crops	Yield (Kg./Hectare) - 2017-18	Yield (Kg./Hectare) - 2018-19	Yield (Kg./Hectare) - 2019-20	Yield (Kg./He
0	All India	Rice	2576	2638	2722	
1	All India	Wheat	3368	3533	3440	
2	All India	Jowar	956	849	989	
3	All India	Bajra	1231	1219	1374	
4	All India	Maize	3065	3070	3006	
...	...	...	...	...	...	
202	West Bengal	Pulses	969	796	801	
203	West Bengal	Foodgrains	2839	2938	2904	
204	West Bengal	Oilseeds	1198	1255	1060	
205	West Bengal	Sugarcane	75000	84485	79657	
206	West Bengal	Jute & Mesta	2616	2644	2805	

207 rows × 7 columns

```
df.head()
```

	States	Crops	Yield (Kg./Hectare) - 2017-18	Yield (Kg./Hectare) - 2018-19	Yield (Kg./Hectare) - 2019-20	Yield (Kg./Hectare)
0	All India	Rice	2576	2638	2722	
1	All India	Wheat	3368	3533	3440	
2	All India	Jowar	956	849	989	
3	All India	Bajra	1231	1219	1374	
4	All India	Maize	3065	3070	3006	

Understand data and its features

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 207 entries, 0 to 206
Data columns (total 7 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   States                         207 non-null    object
 1   Crops                          207 non-null    object
 2   Yield (Kg./Hectare) - 2017-18  207 non-null    int64 
 3   Yield (Kg./Hectare) - 2018-19  207 non-null    int64 
 4   Yield (Kg./Hectare) - 2019-20  207 non-null    int64
 5   Yield (Kg./Hectare) - 2020-21  207 non-null    int64
 6   Yield (Kg./Hectare) - 2021-22  207 non-null    int64
dtypes: int64(5), object(2)
memory usage: 11.4+ KB
```

```
#Display column names
print(df.columns)
```

```
Index(['States', 'Crops', 'Yield (Kg./Hectare) - 2017-18',
       'Yield (Kg./Hectare) - 2018-19', 'Yield (Kg./Hectare) - 2019-20',
       'Yield (Kg./Hectare) - 2020-21', 'Yield (Kg./Hectare) - 2021-22'],
      dtype='object')
```

Describe the data

```
#Get statistical summary
print(df.describe())

#For non-numeric columns, you can get a summary using:
print(df.describe(include='object'))
```

```
       Yield (Kg./Hectare) - 2017-18  Yield (Kg./Hectare) - 2018-19  \
count                     207.000000                     207.000000   
mean                     8811.140097                    8930.685990   
std                     21904.132021                   22380.611941   
min                       238.000000                     266.000000   
25%                       955.000000                     921.500000   
50%                      1882.000000                    1835.000000   
75%                      2853.500000                    2915.000000   
max                    109840.000000                  105050.000000   

       Yield (Kg./Hectare) - 2019-20  Yield (Kg./Hectare) - 2020-21  \
count                     207.000000                     207.000000   
mean                     9033.043478                    9124.739130   
std                     22591.497893                   22768.063193   
min                       251.000000                     275.000000   
25%                       964.000000                     970.500000   
50%                      1886.000000                    1938.000000   
75%                      2901.500000                    3010.500000   
max                    127190.000000                  115810.000000   

       Yield (Kg./Hectare) - 2021-22  
count                     207.000000  
mean                     9196.328502  
std                     22961.286578  
min                       261.000000  
25%                      1000.500000  
50%                      2027.000000  
75%                      3019.500000  
max                    129950.000000  
           States Crops
count         207   207
unique         22    11
top     All India  Rice
freq           11    22
```

Performing Data Preprocessing

One Hot Encoding

```
import pandas as pd

# Perform one-hot encoding for the 'States' and 'Crops' columns
data_encoded = pd.get_dummies(df, columns=['States', 'Crops'])

# Display the first few rows of the encoded dataset
print(data_encoded.head())

# Save the encoded dataset to a new CSV file
data_encoded.to_csv('encoded_crop_yields.csv', index=False)
```

```
   Yield (Kg./Hectare) - 2017-18  Yield (Kg./Hectare) - 2018-19  \
0                           2576                           2638   
1                           3368                           3533   
2                            956                            849   
3                           1231                           1219   
4                           3065                           3070   

   Yield (Kg./Hectare) - 2019-20  Yield (Kg./Hectare) - 2020-21  \
0                           2722                           2717   
1                           3440                           3521   
```

```
2                            989                            1099
3                           1374                            1420
4                           3006                            3199

   Yield (Kg./Hectare) - 2021-22  States_All India  States_Andhra Pradesh  \
0                           2802                 1                      0
1                           3484                 1                      0
2                           1131                 1                      0
3                           1414                 1                      0
4                           3347                 1                      0

   States_Assam  States_Bihar  States_Chhattisgarh  ...  Crops_Cotton  \
0             0             0                    0  ...             0
1             0             0                    0  ...             0
2             0             0                    0  ...             0
3             0             0                    0  ...             0
4             0             0                    0  ...             0

   Crops_Foodgrains  Crops_Jowar  Crops_Jute & Mesta  Crops_Maize  \
0                 0            0                   0            0
1                 0            0                   0            0
2                 0            1                   0            0
3                 0            0                   0            0
4                 0            0                   0            1

   Crops_Oilseeds  Crops_Pulses  Crops_Rice  Crops_Sugarcane  Crops_Wheat
0               0             0           1                0            0
1               0             0           0                0            1
2               0             0           0                0            0
3               0             0           0                0            0
4               0             0           0                0            0

[5 rows x 38 columns]
```

Using Min Max Scaler

```
from sklearn.preprocessing import MinMaxScaler

# Select the columns you want to scale (exclude the categorical columns)
columns_to_scale = [
    'Yield (Kg./Hectare) - 2017-18',
    'Yield (Kg./Hectare) - 2018-19',
    'Yield (Kg./Hectare) - 2019-20',
    'Yield (Kg./Hectare) - 2020-21',
    'Yield (Kg./Hectare) - 2021-22'
]

# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Fit and transform the selected columns
df[columns_to_scale] = scaler.fit_transform(df[columns_to_scale])

# Display the first few rows of the scaled dataset
print(df.head())
```

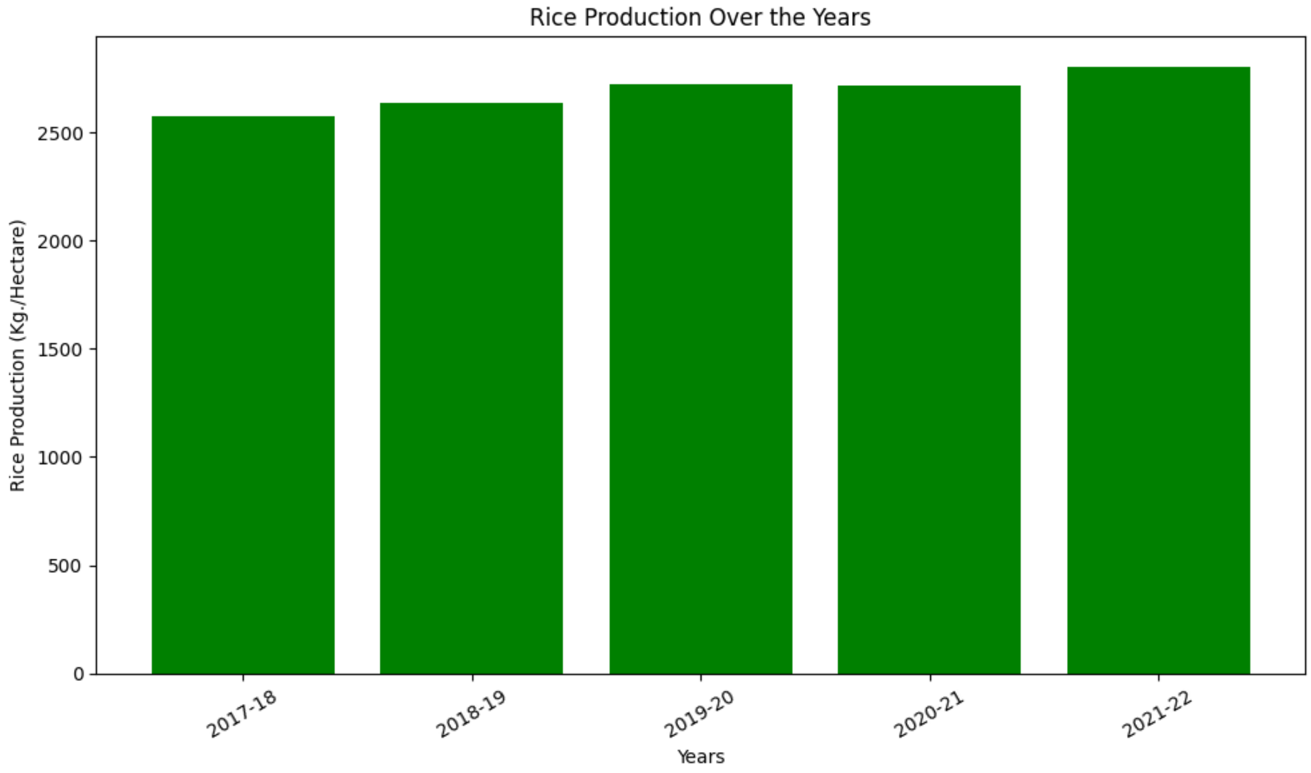
```
      States  Crops  Yield (Kg./Hectare) - 2017-18  \
0  All India   Rice                       0.021332
1  All India  Wheat                       0.028558
2  All India  Jowar                       0.006551
3  All India  Bajra                       0.009060
4  All India  Maize                       0.025793

   Yield (Kg./Hectare) - 2018-19  Yield (Kg./Hectare) - 2019-20  \
0                       0.022637                       0.019466
1                       0.031178                       0.025122
2                       0.005564                       0.005814
3                       0.009095                       0.008847
4                       0.026760                       0.021703

   Yield (Kg./Hectare) - 2020-21  Yield (Kg./Hectare) - 2021-22
0                       0.021136                       0.019593
1                       0.028095                       0.024852
2                       0.007132                       0.006708
3                       0.009910                       0.008890
4                       0.025308                       0.023795
```

Performing EDA

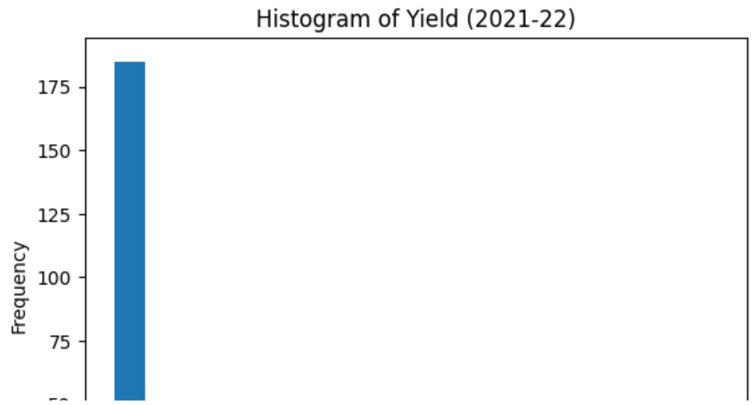
```
summary_stats = df.describe()
summary_stats
```

	Yield (Kg./Hectare) - 2017-18	Yield (Kg./Hectare) - 2018-19	Yield (Kg./Hectare) - 2019-20	Yield (Kg./Hectare) - 2020-21
count	207.000000	207.000000	207.000000	207.000000
mean	0.078221	0.082691	0.069183	0.076598
std	0.199852	0.213588	0.177971	0.197066
min	0.000000	0.000000	0.000000	0.000000
25%	0.006542	0.006256	0.005617	0.006020
50%	0.015000	0.014974	0.012880	0.014394
75%	0.023864	0.025281	0.020880	0.023677
max	1.000000	1.000000	1.000000	1.000000

```
# Box graph of rice yield for over the years
import matplotlib.pyplot as plt

# Data for rice production over the years
years = ["2017-18", "2018-19", "2019-20", "2020-21", "2021-22"]
rice_production = [2576, 2638, 2722, 2717, 2802]

# Creating a bar graph
plt.figure(figsize=(10, 6))
plt.bar(years, rice_production, color='green')
plt.xlabel('Years')
plt.ylabel('Rice Production (Kg./Hectare)')
plt.title('Rice Production Over the Years')
plt.xticks(rotation=30)
plt.tight_layout()

# Display
plt.show()
```

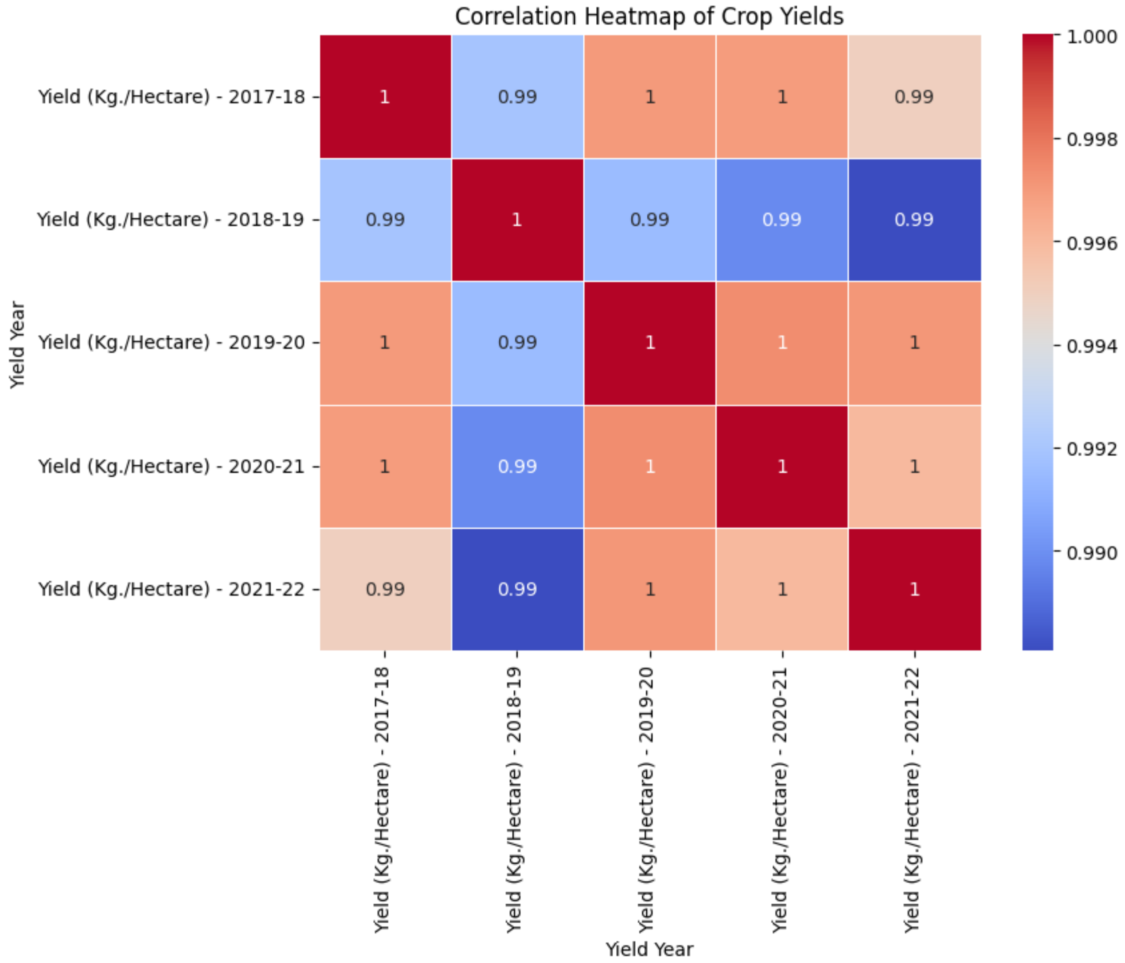


```
# Creating a histogram.
plt.hist(df['Yield (Kg./Hectare) - 2021-22'], bins=20)
plt.xlabel('Yield (Kg./Hectare) - 2021-22')
plt.ylabel('Frequency')
plt.title('Histogram of Yield (2021-22)')
plt.show()
```

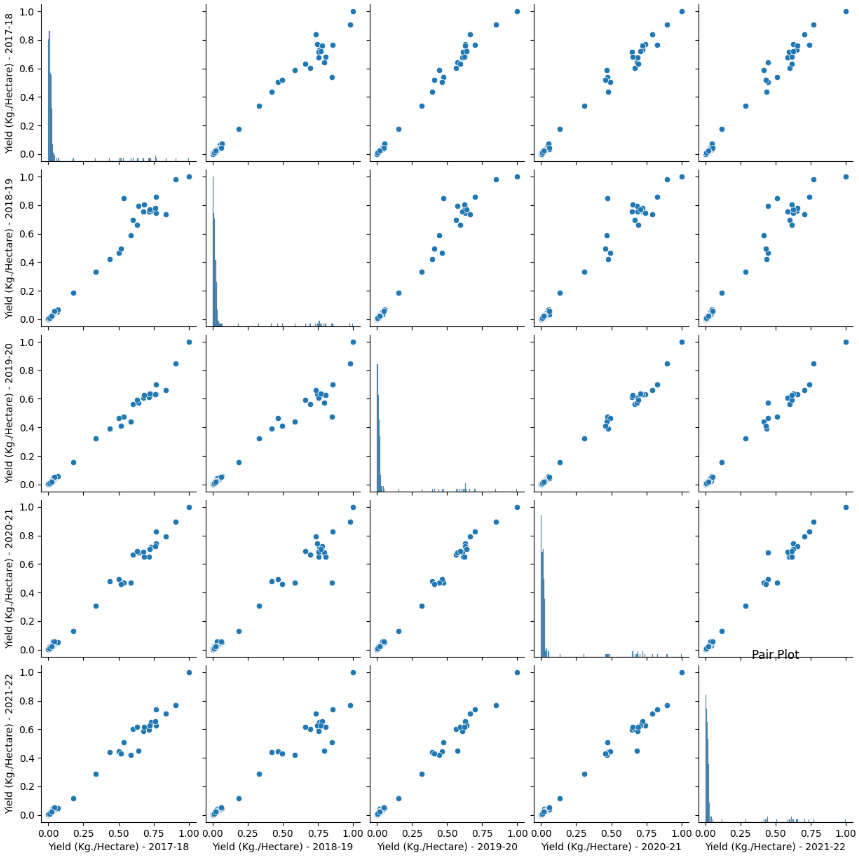


```
# Correlation matrix
correlation_matrix = df.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix,annot=True,cmap='coolwarm',linewidths=0.5)
plt.title('Correlation Heatmap of Crop Yields')
plt.xlabel('Yield Year')
plt.ylabel('Yield Year')
plt.show()
```

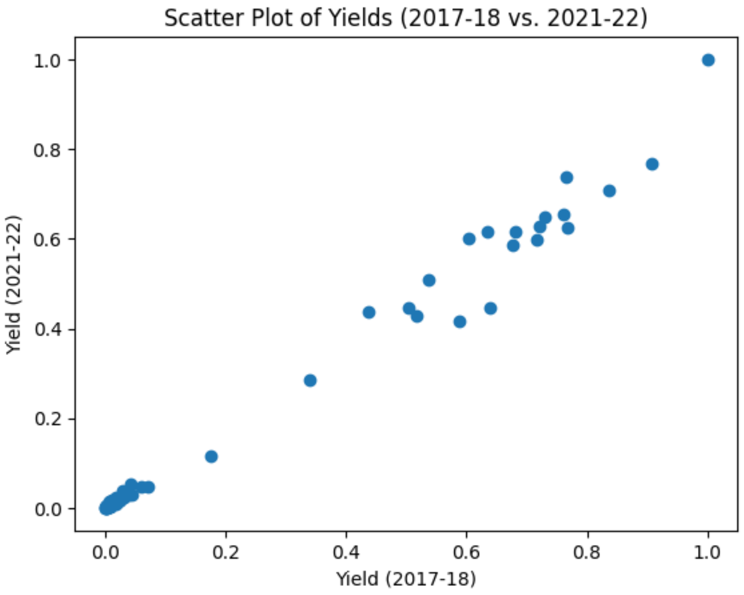
```
<ipython-input-13-c6a72e8492ac>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  correlation_matrix = df.corr()
```



```
# Pairplot
sns.pairplot(df)
plt.title('Pair Plot')
plt.show()
```



```
# Scatter Plot
plt.scatter(df['Yield (Kg./Hectare) - 2017-18'], df['Yield (Kg./Hectare) - 2021-22'])
plt.xlabel('Yield (2017-18)')
plt.ylabel('Yield (2021-22)')
plt.title('Scatter Plot of Yields (2017-18 vs. 2021-22)')
plt.show()
```



```
from matplotlib._api import define_aliases
# Calculate summary statistics
mean_yield = df['Yield (Kg./Hectare) - 2017-18'].mean()
median_yield = df['Yield (Kg./Hectare) - 2017-18'].median()
std_deviation = df['Yield (Kg./Hectare) - 2017-18'].std()
min_yield = df['Yield (Kg./Hectare) - 2017-18'].min()
max_yield = df['Yield (Kg./Hectare) - 2017-18'].max()

print(f"Mean: {mean_yield:.2f}")
print(f"Median: {median_yield:.2f}")
print(f"Standard Deviation: {std_deviation:.2f}")
print(f"Min: {min_yield:.2f}")
print(f"Max: {max_yield:.2f}")
```

```
Mean: 0.08
Median: 0.01
Standard Deviation: 0.20
Min: 0.00
Max: 1.00
```

```
from matplotlib._api import define_aliases
# Calculate summary statistics
mean_yield = df['Yield (Kg./Hectare) - 2021-22'].mean()
median_yield = df['Yield (Kg./Hectare) - 2021-22'].median()
std_deviation = df['Yield (Kg./Hectare) - 2021-22'].std()
min_yield = df['Yield (Kg./Hectare) - 2021-22'].min()
max_yield = df['Yield (Kg./Hectare) - 2021-22'].max()

print(f"Mean: {mean_yield:.2f}")
print(f"Median: {median_yield:.2f}")
print(f"Standard Deviation: {std_deviation:.2f}")
print(f"Min: {min_yield:.2f}")
print(f"Max: {max_yield:.2f}")
```

```
Mean: 0.07
Median: 0.01
Standard Deviation: 0.18
Min: 0.00
Max: 1.00
```