In [2]:

```
import pandas as pd
import numpy as np
import os
import sys
import matplotlib.pylab as plt
import plotly.graph_objects as go
import plotly.express as px
```

In [3]:

```
# Useful parameters
DATA_DIR = 'datasets/'
prod_estimates_dataset = 'PRODUCTION ESTIMATES.csv'
wholesale_prices_dataset = 'NATIONAL WHOLESALE PRICE OF
SOME SELECTED COMMODITIES.csv'
rainfall_dataset = 'RAINFALL.csv'
path_to_prod_est_dataset = os.path.join(DATA_DIR,
prod_estimates_dataset)
path_to_ws_prices_dataset = os.path.join(DATA_DIR,
wholesale_prices_dataset)
path_to_rainfall_dataset = os.path.join(DATA_DIR,
rainfall_dataset)
print('Production estimates dataset is found in:',
path_to_prod_est_dataset)
print('Wholesale prices dataset is found in:',
path_to_ws_prices_dataset)
print('Rainfall dataset is found in:',
path_to_rainfall_dataset)
```

```
Production estimates dataset is found in:
datasets/PRODUCTION ESTIMATES.csv
Wholesale prices dataset is found in: datasets/
NATIONAL WHOLESALE PRICE OF SOME SELECTED
COMMODITIES.csv
Rainfall dataset is found in: datasets/
RAINFALL.csv
```

In [4]:

```
data_rain= pd.read_csv('datasets/RAINFALL.csv')
data_rain.dtypes
```

Out[4]:

```
YEAR                    int64
REGION                 object
TOTAL RAINFALL(MM)      int64
dtype: object
```

In [5]:

```
data_production = pd.read_csv('datasets/production.csv')
data_production.dtypes
```

Out[5]:
```
REGION              object
DISTRICT            object
YEAR                 int64
CROP                object
AREA (HA)           object
YIELD (MT/HA)       object
PRODUCTION (MT)     object
dtype: object
```

In [6]:
```
df_whole= pd.read_csv('datasets/wholesale.csv',
encoding="ISO-8859-1")
dw= df_whole
```

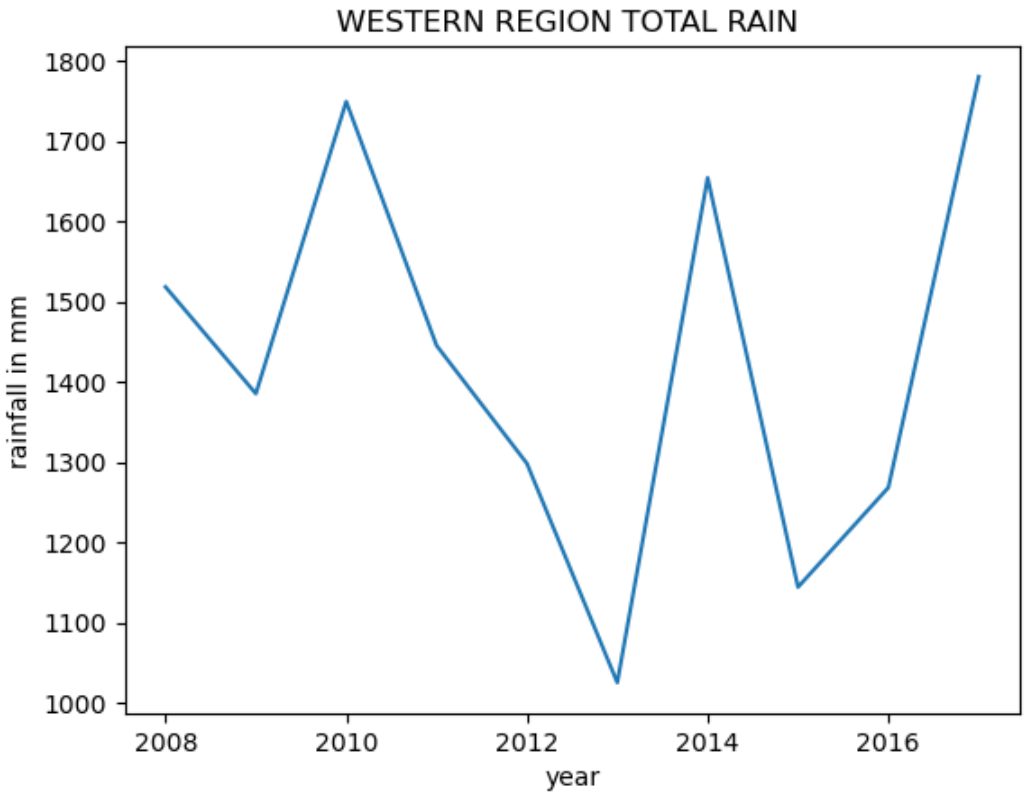
In [7]:
```
data_production.rename(columns={'YIELD (MT/HA)':'Yield',
'PRODUCTION (MT)': 'Production'}, inplace=True)
data_production= data_production.dropna(axis=0)

dp=
data_production.drop(data_production[data_production.Yiel
d.str.contains(r'[- ]')].index)
dp= dp.drop(dp[dp.Production.str.contains(r'[,]')].index)
dp['Yield'] = dp.Yield.astype(float)
dp['Production'] = dp.Production.astype(float)
```

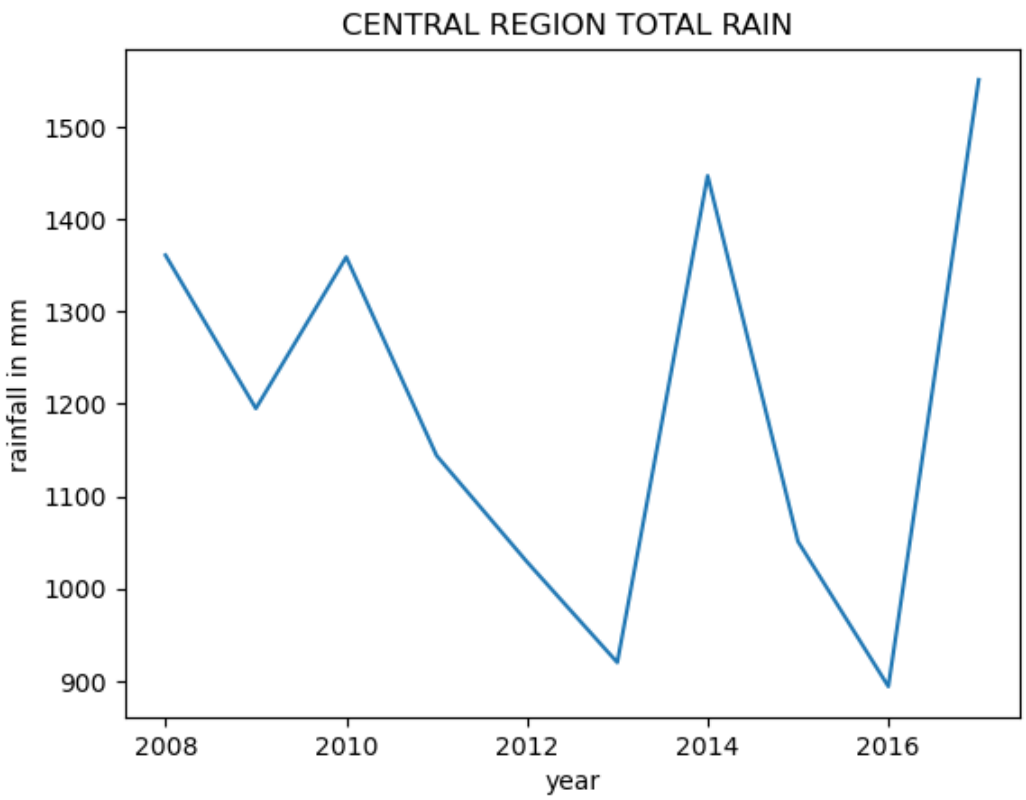
In [8]:
```
# SET SIMPLE NAMES FOR DATA SETS. dr=rain data,
dw=wholesale data, dp= production data

dr=data_rain
dw=df_whole
```

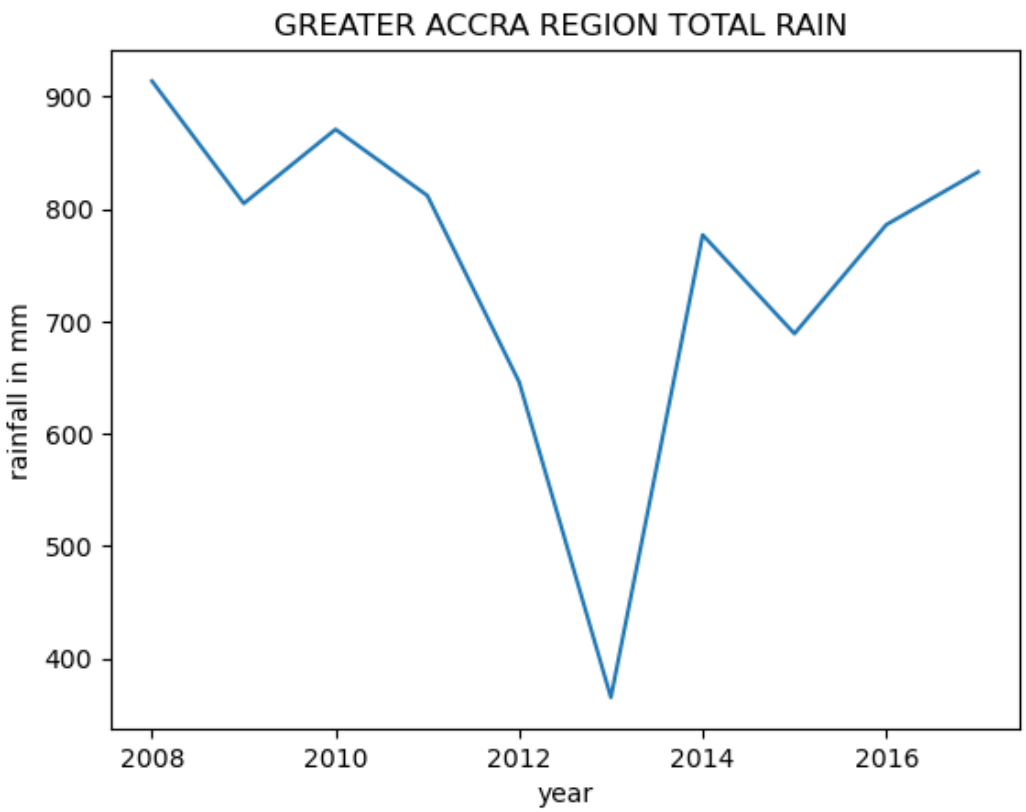
In [17]:
```
dw=dw.rename(columns={" PRICE, ¢ GH":'Price'})
```

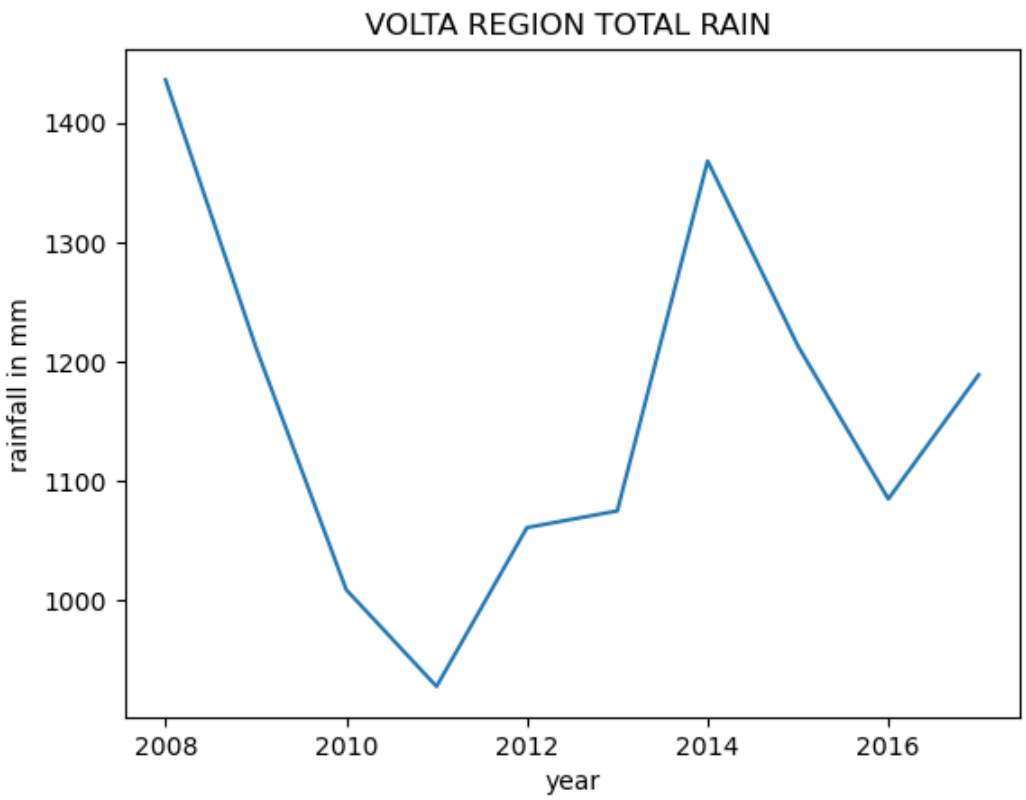
In [15]:
```
# Check for how rainfall changed for each region for each
month as base to later compare production

regions= dr['REGION'].unique()
year = dp['YEAR'].unique()
crops = dp['CROP'].unique()

for x in range(0,len(regions)):
    R = dr[dr['REGION']== regions[x]].reset_index()
    plt.plot(R['YEAR'],R['TOTAL RAINFALL(MM)'])
    plt.title(regions[x]+ ' REGION TOTAL RAIN ')
    plt.xlabel('year')
```

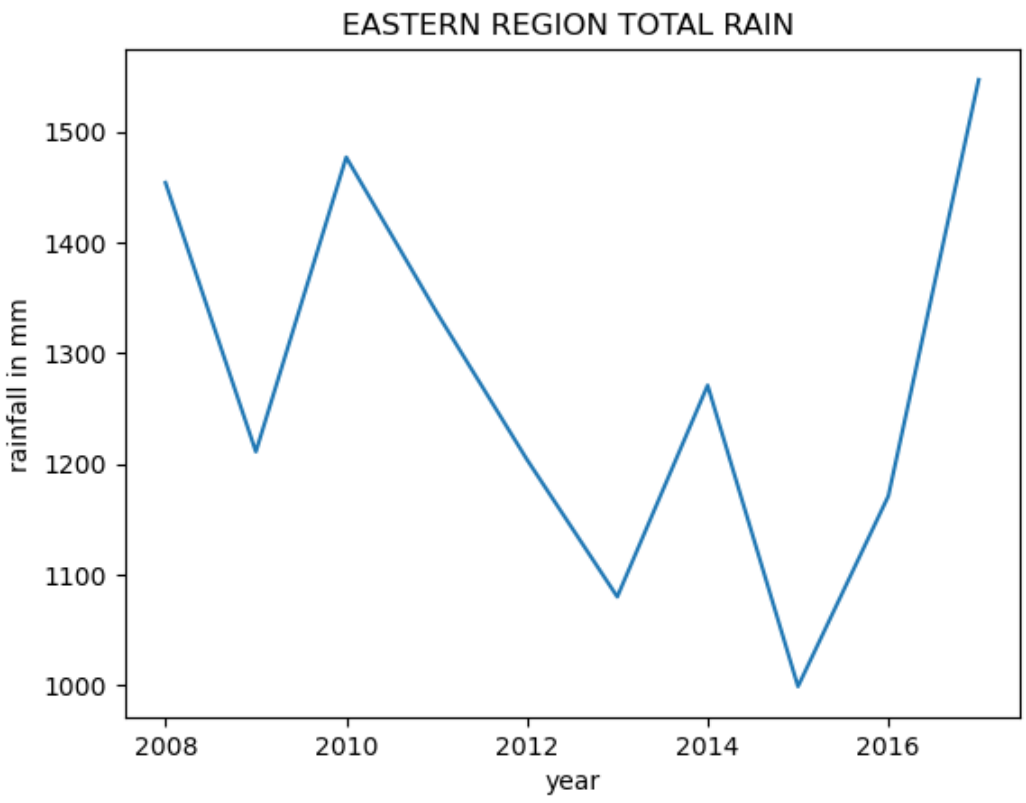
```
    plt.ylabel('rainfall in mm')
    plt.show()



"""compare rainfall directly with production"""
```

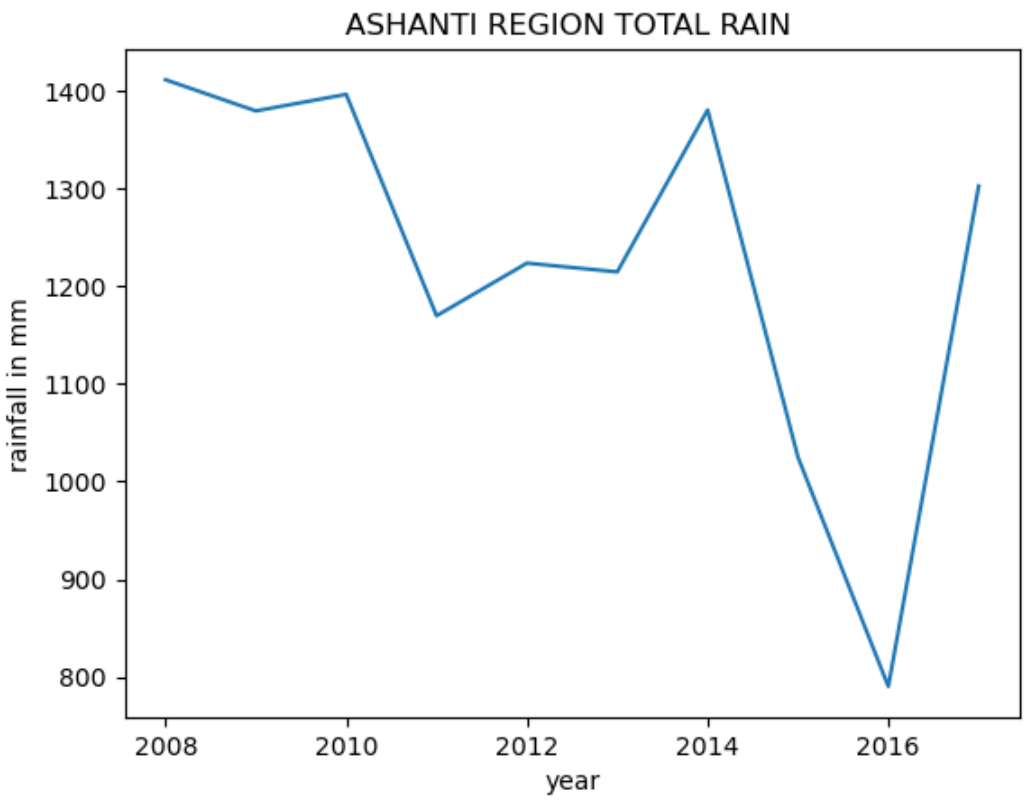


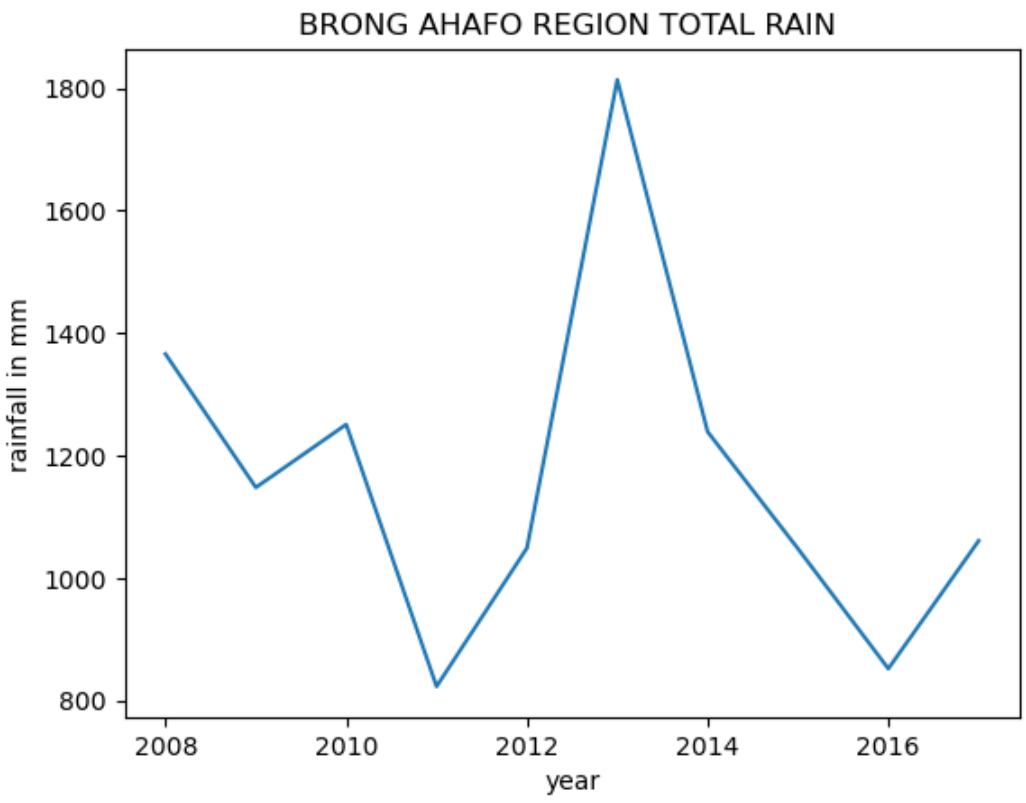

CENTRAL REGION TOTAL RAIN
rainfall in mm
1500
1400
1300
1200
1100
1000
900
2008
2010
2012
2014
2016
year

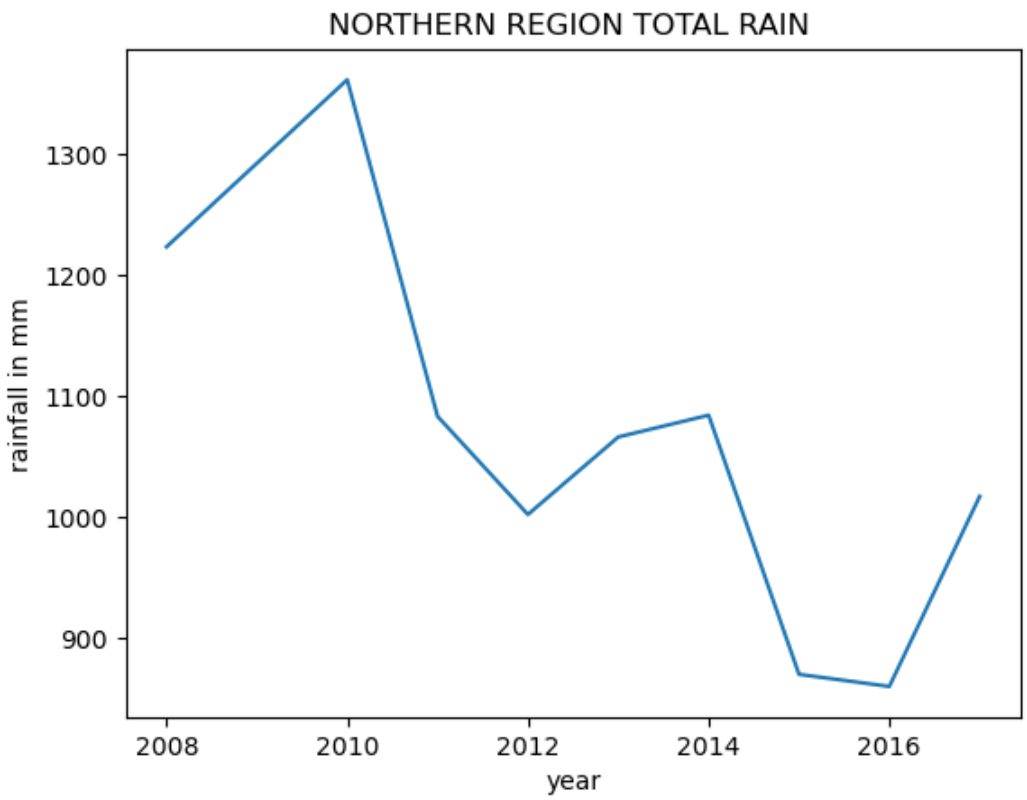

GREATER ACCRA REGION TOTAL RAIN
rainfall in mm
900
800
700
600
500
400
2008
2010
2012
2014
2016
year

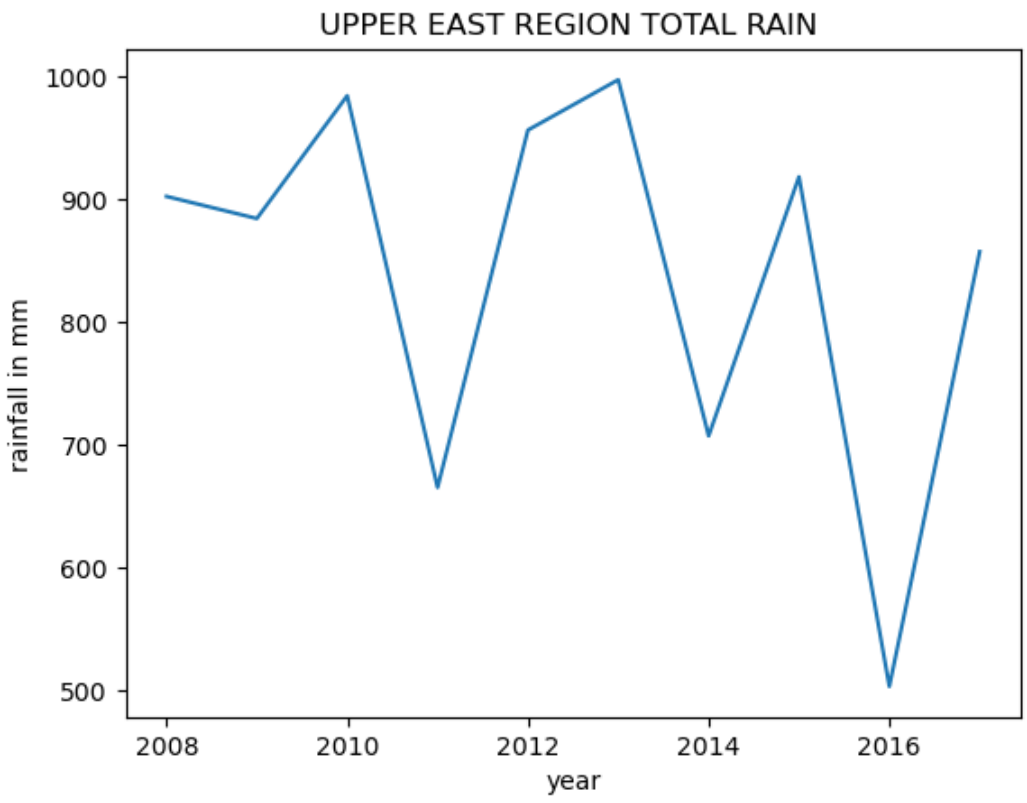

VOLTA REGION TOTAL RAIN
rainfall in mm
1400
1300
1200
1100
1000
2008
2010
2012
2014
2016
year

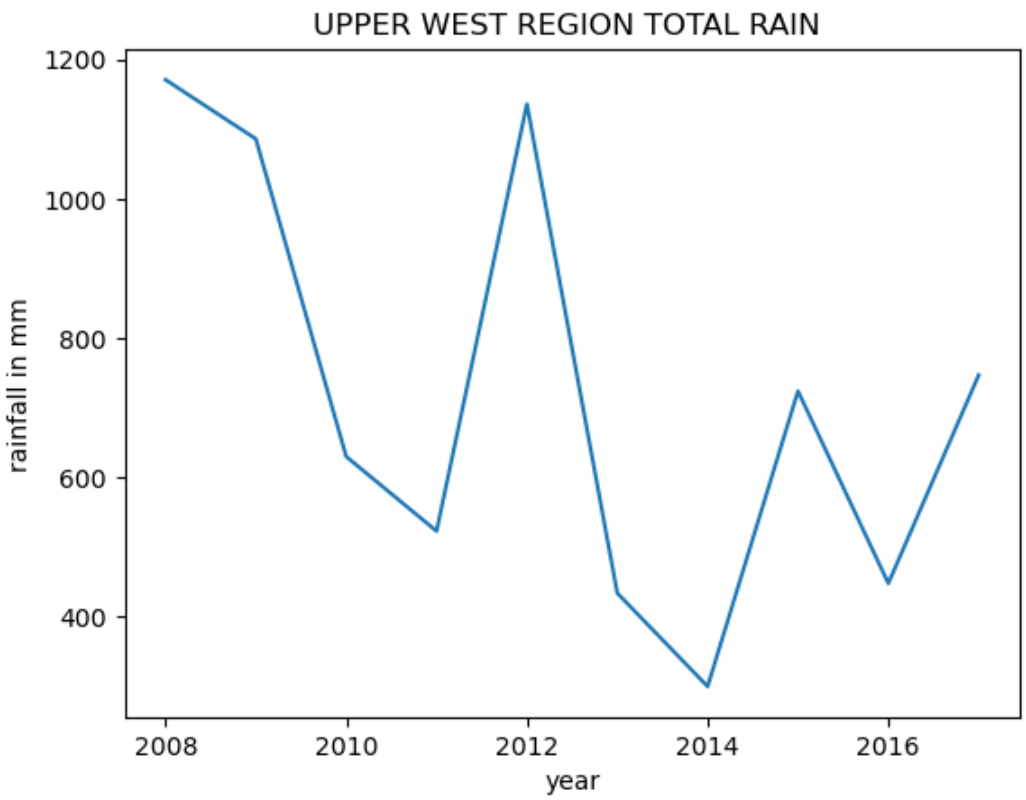

EASTERN REGION TOTAL RAIN
rainfall in mm
1500
1400
1300
1200
1100
1000
2008
2010
2012
2014
2016
year


ASHANTI REGION TOTAL RAIN
rainfall in mm
1400
1300
1200
1100
1000
900
800
2008
2010
2012
2014
2016
year


BRONG AHAFO REGION TOTAL RAIN
rainfall in mm
1800
1600
1400
1200
1000
800
2008
2010
2012
2014
2016
year


NORTHERN REGION TOTAL RAIN
rainfall in mm
1300
1200
1100
1000
900
2008
2010
2012
2014
2016
year


UPPER EAST REGION TOTAL RAIN
rainfall in mm
1000
900
800
700
600
500
2008
2010
2012
2014
2016
year



Out[15]: 'compare rainfall directly with production'

In [18]:

```
crops= dw.COMMODITY.unique()
h={}
for i in range(len(year)):
    for n in range (len(crops)):
        x2=dw[dw.COMMODITY==crops[n]]
        x1=x2[x2.YEAR==year[i]]
        x=x1["Price"].mean()
        h[crops[n]]={year[i]:x }


print(h)

"""xaxis = list(h.keys())
yaxis = list(h.values())
plt.plot(xaxis,yaxis)
plt.xlabel('year')
plt.ylabel('average price')
plt.show() """
```

```
{'MAIZE': {2017: 104.47209166666666}, 'MILLET':
{2017: 19.230500000000003}, 'LOCAL RICE': {2017:
209.91091666666668}, 'SORGHUM': {2017:
0.032499999999999994}, 'YAM': {2017:
516.4208333333333}, 'COCOYAM': {2017: 204.38}}
```

Out[18]:
```
"xaxis = list(h.keys())\nyaxis = list(h.values())
\nplt.plot(xaxis,yaxis)\nplt.xlabel('year')
\nplt.ylabel('average price')\nplt.show() "
```

In [9]:
```
month_switch={'JANUARY': 1,'FEBRUARY': 2,'MARCH':
3,'APRIL': 4,'MAY': 5,'JUNE': 6,'JULY': 7,'AUGUST':
8,'SEPTEMBER': 9,'OCTOBER': 10,'NOVEMBER': 11,'DECEMBER':
12}
```

In [20]:
```
dw['monthnum']= (pd.Series(dw.MONTH)).map(month_switch)
crops=dp.CROP.unique()
```

In [11]:
```
dateframe=pd.DataFrame({'yy':dw.YEAR, 'mm':dw.monthnum,
'dd': np.ones(len(dw.YEAR))})
dateframe.columns=['year','month','day']
dw['Date']=pd.to_datetime(dateframe)
```

In [21]:
```
data_crops=[]
for i in range(len(crops)):
    data_crops
```

Normalize yield and find most produced plant and the change in region and years

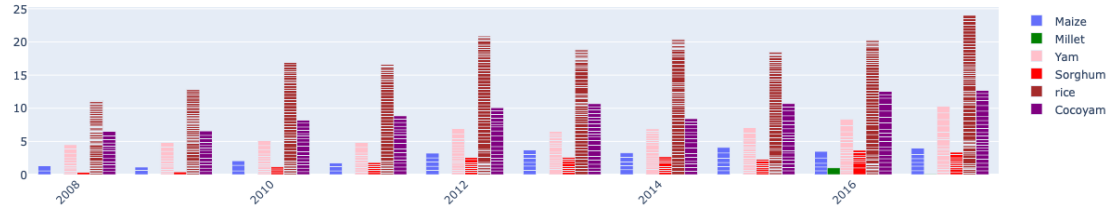
In [22]:
```
dp_maize=dp[dp['CROP']=='MAIZE']
m=dp_maize.Yield.max()
dp_maize['Yield']=dp_maize['Yield'].div(m).round(2)
dp_millet=dp[dp['CROP']=='MILLET']
mi=dp_millet.Yield.max()
dp_millet['Yield']=dp_millet['Yield'].div(mi).round(2)
dp_rice=dp[dp['CROP']=='RICE']
r=dp_rice.Yield.max()
dp_rice['Yield']=dp_rice['Yield'].div(r).round(2)
dp_sorghum=dp[dp['CROP']=='SORGHUM']
```

```
s=dp_sorghum.Yield.max()
dp_sorghum['Yield']=dp_sorghum['Yield'].div(s).round(2)
dp_yam=dp[dp['CROP']=='YAM']
y=dp_yam.Yield.max()
dp_yam['Yield']=dp_yam['Yield'].div(y).round(2)
dp_cocoyam=dp[dp['CROP']=='COCOYAM']
c=dp_cocoyam.Yield.max()
dp_cocoyam['Yield']=dp_cocoyam['Yield'].div(c)
```

In [23]:

```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=dp_maize["YEAR"],
    y=dp_maize['Yield'],
    name='Maize'
))
fig.add_trace(go.Bar(
    x=dp_millet["YEAR"],
    y=dp_millet["Yield"],
    name='Millet',
    marker_color='green'
))
fig.add_trace(go.Bar(
    x=dp_yam["YEAR"],
    y=dp_yam["Yield"],
    name='Yam',
    marker_color='pink'
))
fig.add_trace(go.Bar(
    x=dp_sorghum["YEAR"],
    y=dp_sorghum["Yield"],
    name='Sorghum',
    marker_color='red'))

fig.add_trace(go.Bar(
    x=dp_rice["YEAR"],
    y=dp_rice["Yield"],
    name='rice',
    marker_color='brown'))

fig.add_trace(go.Bar(
    x=dp_cocoyam["YEAR"],
    y=dp_cocoyam["Yield"],
```

```
    name='Cocoyam',
    marker_color='purple'))

fig.update_layout(barmode='group', xaxis_tickangle=-45)
fig.show()
```



In [24]:

```
df_merged = pd.merge(dr, dp)
```

In [25]:

```
Data= df_merged.drop('DISTRICT', axis=1)
```

Use merged data in pivot

In [28]:

```
#pip install pivottablejs

import pandas as pd

from pivottablejs import pivot_ui

pivot_ui(df_merged)
```

Row Heatmap | Sum | Yield

Yield | REGION | TOTAL RAINFALL(MM) | DISTRICT | AREA (HA) | Production

YEAR

CROP

CROP \ YEAR	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	Totals
CASSAVA										91.35	91.35
COCOYAM	55.57	56.19	70.28	75.61	86.13	91.26	71.98	91.36	107.13	108.79	814.30
COWPEA			9.46	11.55	19.72	25.10	31.06	34.30	41.62	45.56	218.37
GROUNDNUT			8.10	7.53	12.89	15.89	14.78	13.62	20.10	21.54	114.45
MAIZE	2.32	2.01	3.70	3.16	5.76	6.73	5.80	7.27	6.25	7.20	50.20
MAIZE										19.37	19.37
MILLET									1.00	0.07	1.07
PLANTAIN	38.14	32.40	40.20	40.44	79.50	93.15	89.85	88.87	68.61	135.40	706.56
RICE	61.54	70.72	92.97	91.16	114.98	103.81	112.23	101.42	111.33	131.94	992.10
RICE										4.50	4.50
SORGHUM	1.40	1.44	4.30	6.70	9.68	9.54	10.00	8.70	13.65	12.67	78.08
SOYABEAN	7.55	8.88	11.60	9.02	9.46	9.70	9.46	9.47	13.19	15.10	103.43
YAM	60.87	65.95	69.16	65.93	93.75	88.25	93.39	95.71	113.18	139.82	886.01
Totals	227.39	237.59	309.77	311.10	431.87	443.43	438.55	450.72	496.06	733.31	4,079.79

In [27]:

```
import seaborn as sns

# Create heatmap using seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(df_merged.corr(), cmap='coolwarm',
annot=True, fmt=".2g", linewidths=0.5)
```

Out[27]: <AxesSubplot:>