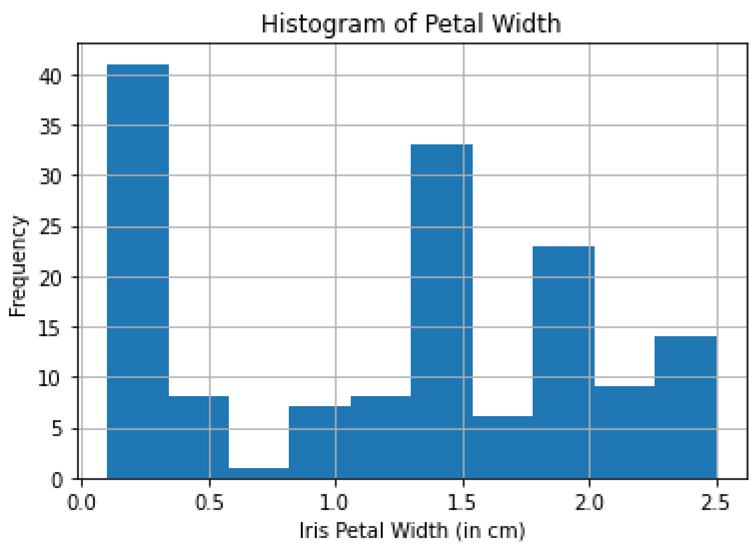
In [11]:

```
#1a
%matplotlib inline

plot = data['petal width'].hist(bins=10)
plot.set_xlabel('Iris Petal Width (in cm)')
plot.set_ylabel('Frequency')
plot.set_title('Histogram of Petal Width')
```

Out[11]:

```
Text(0.5, 1.0, 'Histogram of Petal Width')
```



1b) Based on the scatter of all pairs of attributes, it looks like petal width and petal length have the strongest correlation

In [12]:
```
#1c
import matplotlib.pyplot as plt

length = 'petal length'
width  = 'petal width'

fig, ax = plt.subplots(1, 1, figsize=(6, 6))

ax.scatter(data[length], data[width], color='red')
ax.set_xlabel(length)
ax.set_ylabel(width)
_ = ax.set_title('Petal Width vs petal Length')

print()
```

1d) The petal width and petal length have the closest to 1 correlation coefficient, r

In [13]:

```
print('Correlation:')
data.corr()
```

```
Correlation:
```

Out[13]:

	sepal length	sepal width	petal length	petal width
sepal length	1.000000	-0.109369	0.871754	0.817954
sepal width	-0.109369	1.000000	-0.420516	-0.356544
petal length	0.871754	-0.420516	1.000000	0.962757
petal width	0.817954	-0.356544	0.962757	1.000000

Question 2

In [14]:

```
import pandas as pd

googleData = pd.read_csv('/kaggle/input/google-trends-csv/google_trends.csv', header = 0)
googleData.columns = ['Week', 'Labubu', 'Lafufu', 'Bison']

googleData.head()
```

Out[14]:

	Week	Labubu	Lafufu	Bison
0	2024-12-08	0	0	0
1	2024-12-15	43	0	0
2	2024-12-22	0	0	0
3	2024-12-29	0	0	0
4	2025-01-05	0	0	0

In [15]:
```
print('Correlation:')
googleData.corr()
```

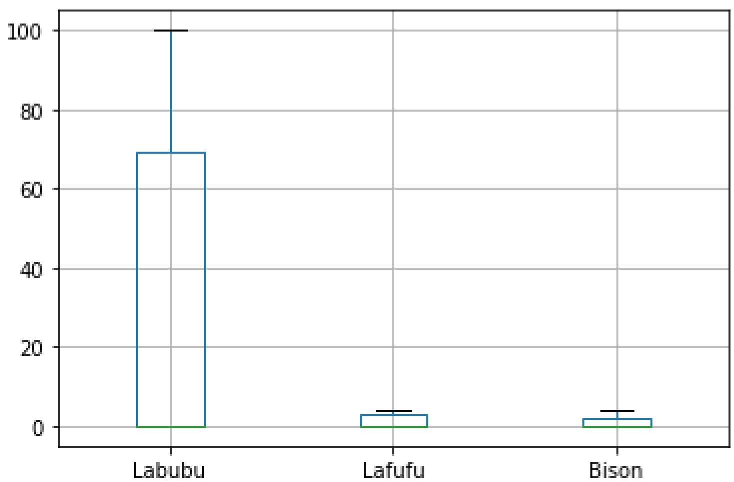
```
Correlation:
```

Out[15]:

	Labubu	Lafufu	Bison
Labubu	1.000000	0.910767	0.838075
Lafufu	0.910767	1.000000	0.796217
Bison	0.838075	0.796217	1.000000

In [16]:
```
#2a
googleData.boxplot()
```

Out[16]:
```
<AxesSubplot:>
```



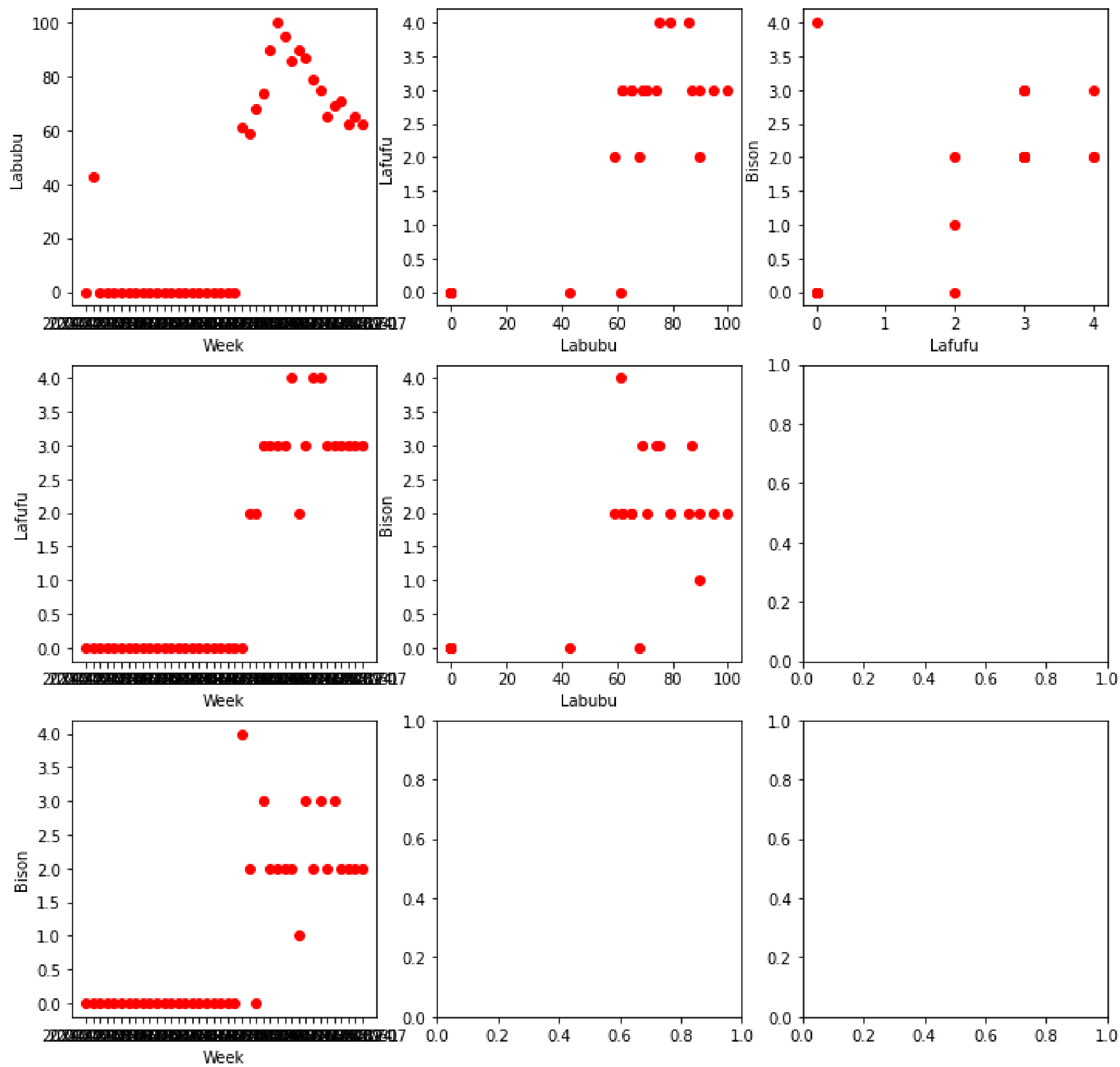
In [17]:

```
#2b
import pandas as pd

# keep date separate
date_col = 'Week'
val_cols = googleData.columns.difference([date_col])

vals = googleData[val_cols].apply(pd.to_numeric, errors='coerce')

# min-max per column
mins = vals.min()
ranges = vals.max() - mins

# avoid divide-by-zero for constant columns
ranges_safe = ranges.replace(0, 1)
norm = (vals - mins) / ranges_safe

# set constant columns to 0 (all values equal)
norm.loc[:, ranges == 0] = 0

# put date back for one DataFrame
googleData_n = pd.concat([googleData[[date_col]], norm], axis=1)
```

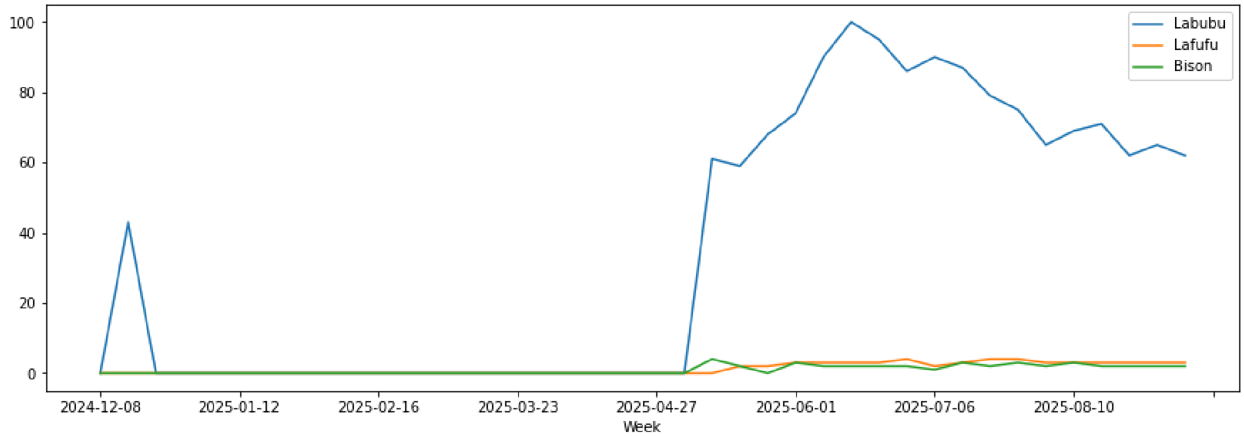
In [18]:

```
import matplotlib.pyplot as plt

print('Raw data:')

fig, axes = plt.subplots(3, 3, figsize=(12,12))
index = 0
for i in range(3):
    for j in range(i+1,4):
        ax2 = i
        ax1 = j - i - 1
        axes[ax1][ax2].scatter(googleData[googleData.columns[i]], googleD
ata[googleData.columns[j]], color='red')
        axes[ax1][ax2].set_xlabel(googleData.columns[i])
        axes[ax1][ax2].set_ylabel(googleData.columns[j])

print()
```

Raw data:

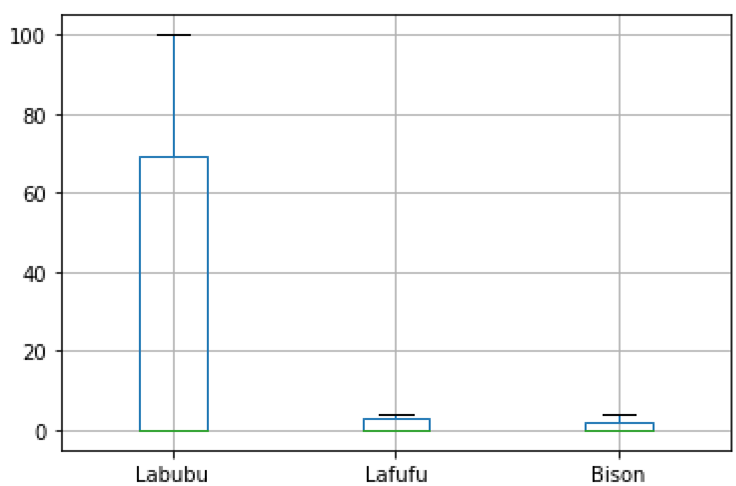

Labubu
Week
Lafufu
Labubu
Bison
Lafufu
Lafufu
Week
Bison
Labubu
Bison
Week

In [19]:
```
df = googleData.set_index('Week')[['Labubu','Lafufu','Bison']]
df.plot(figsize=(15,5))
plt.show()
```



In [20]:
```
googleData.boxplot()
```

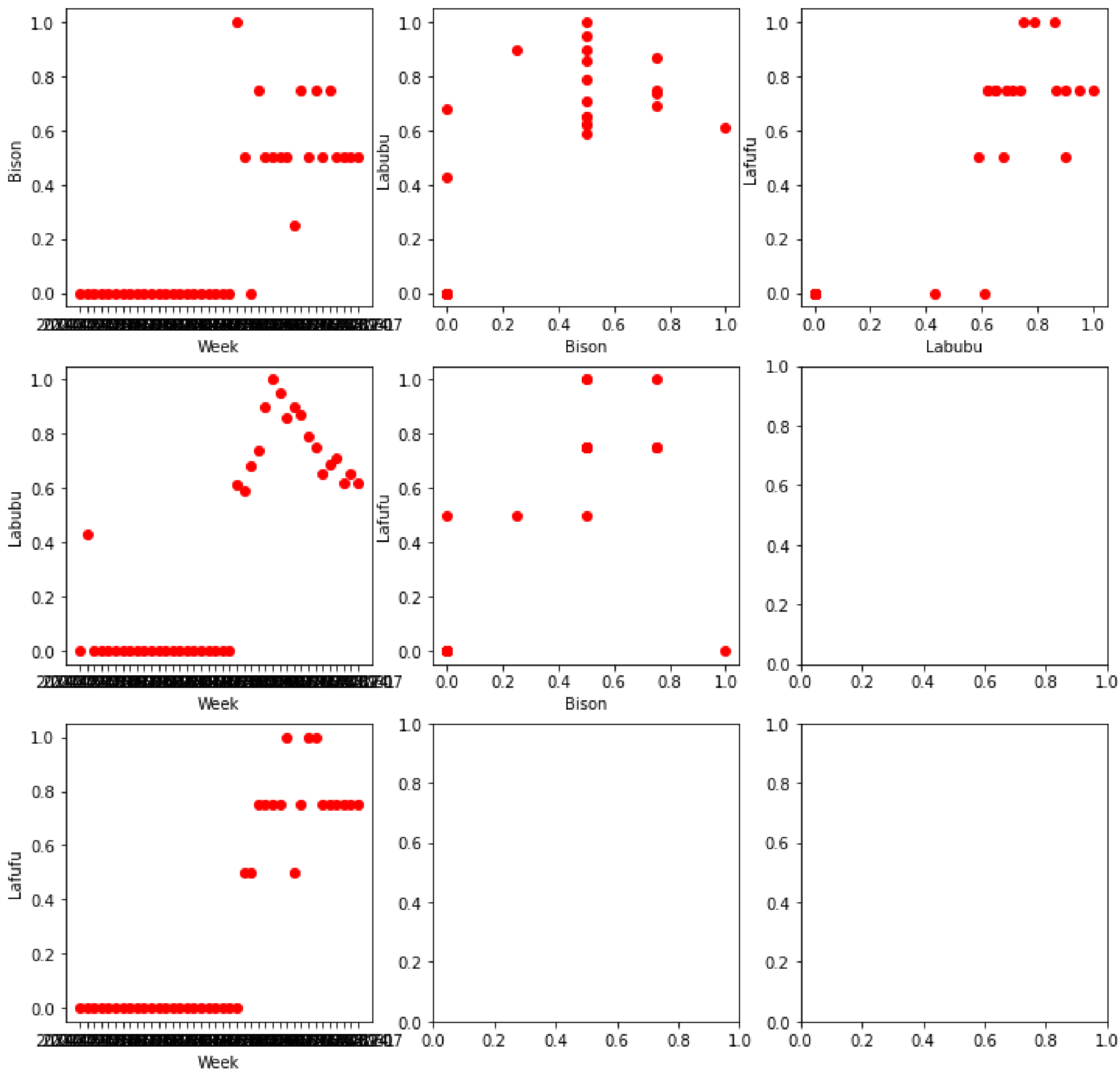
Out[20]:
```
<AxesSubplot:>
```



In [21]:

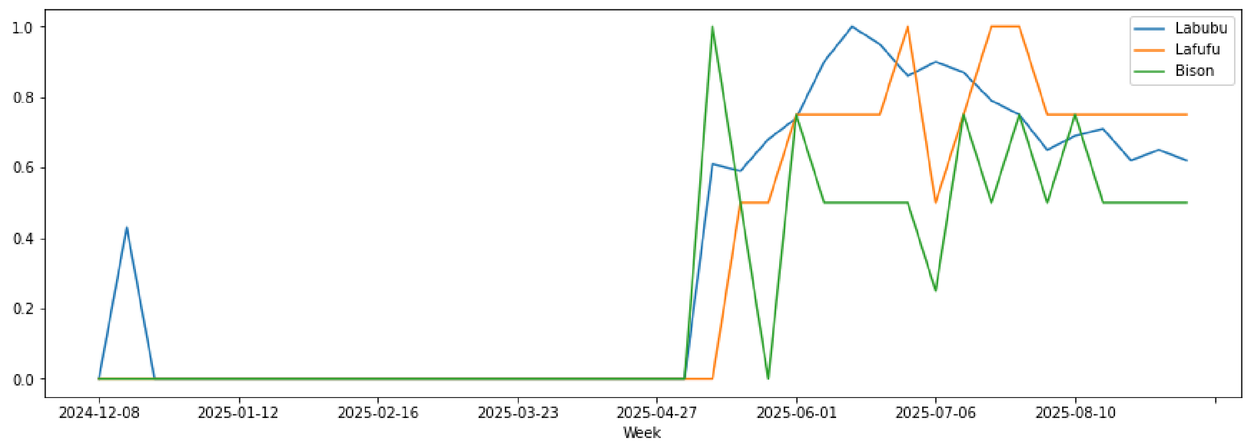
```
import matplotlib.pyplot as plt

print('min-max normalization:')

fig, axes = plt.subplots(3, 3, figsize=(12,12))
index = 0
for i in range(3):
    for j in range(i+1,4):
        ax2 = i
        ax1 = j - i - 1
        axes[ax1][ax2].scatter(googleData_n[googleData_n.columns[i]], googleData_n[googleData_n.columns[j]], color='red')
        axes[ax1][ax2].set_xlabel(googleData_n.columns[i])
        axes[ax1][ax2].set_ylabel(googleData_n.columns[j])

print()
```

min-max normalization:



In [22]:

```
df = googleData_n.set_index('Week')[['Labubu','Lafufu','Bison']]
df.plot(figsize=(15,5))
plt.show()
```

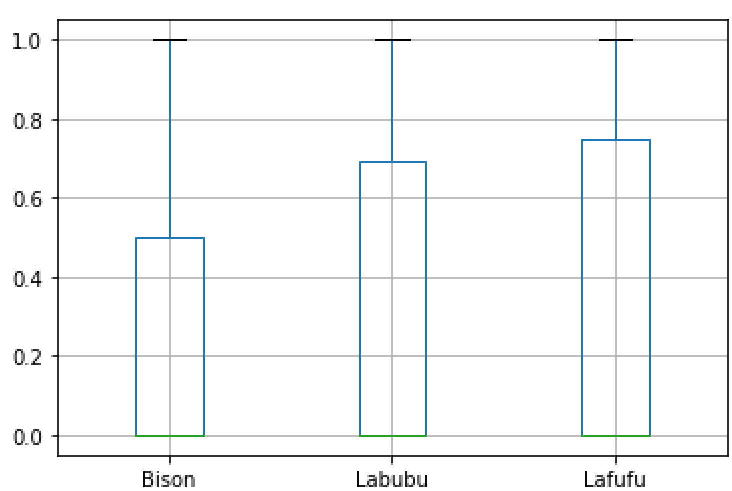


In [23]:

```
googleData_n.boxplot()
```

Out[23]:

```
<AxesSubplot:>
```



2c) min-max normalization is useful for this data. It significantly affects the time series and boxplot visualization of the data.