In [2]:

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

In [3]:

```
### import data
data = pd.read_excel(r"C:\Users\shubham lokare\Downloads\Data\Data.xlsx")
```

In [4]:

```
data
```

Out[4]:

	Billing date	Variant	Economic Index	Industry Growth Rate (%)	Seasonality Factor
0	2022-08-18	XXX11	87.45	8.45	High
1	2022-08-19	XXX11	145.07	14.54	Medium
2	2022-08-20	XXX17	123.20	-2.96	Medium
3	2022-08-21	XXXV1	109.87	-4.83	Low
4	2022-08-22	XXX11	65.60	3.67	Low
...	...	...	...	...	...
3090	2023-03-30	XXXV5	95.03	11.05	Medium
3091	2023-07-23	XXX12	145.76	10.76	Low
3092	2023-07-25	XXX12	89.90	0.94	Low
3093	2024-04-30	XXXV2	133.98	-2.29	Low
3094	2024-04-30	XXXV2	68.85	-2.09	Medium

3095 rows × 5 columns

In [5]:

```
data.head(10)
```

Out[5]:

	Billing date	Variant	Economic Index	Industry Growth Rate (%)	Seasonality Factor
0	2022-08-18	XXX11	87.45	8.45	High
1	2022-08-19	XXX11	145.07	14.54	Medium
2	2022-08-20	XXX17	123.20	-2.96	Medium
3	2022-08-21	XXXV1	109.87	-4.83	Low
4	2022-08-22	XXX11	65.60	3.67	Low
5	2022-08-22	XXX11	65.60	-3.15	Low
6	2022-08-22	XXX11	55.81	9.97	Medium
7	2022-08-22	XXX11	136.62	13.29	Medium
8	2022-08-23	XXXV1	110.11	3.68	High
9	2022-08-23	XXXV1	120.81	0.17	Medium

In [6]:

```
### 1st movement business decision
data.describe()
```

Out[6]:

	Billing date	Economic Index	Industry Growth Rate (%)
count	3095	3095.000000	3095.00000
mean	2023-07-05 18:49:40.032310016	99.936892	4.86347
min	2022-08-18 00:00:00	50.000000	-5.00000
25%	2023-02-13 00:00:00	74.690000	-0.17000
50%	2023-06-27 00:00:00	100.480000	4.76000
75%	2023-11-30 00:00:00	125.130000	9.83000
max	2024-04-30 00:00:00	149.970000	14.99000
std	NaN	29.131357	5.75851

In [9]:

```
### mean for Economic Index
data['Economic Index'].mean()
```

Out[9]:

```
99.93689176090469
```

In [10]:

```
## mean for Industry Growth Rate (%)
data['Industry Growth Rate (%)'].mean()
```

Out[10]:

```
4.863470113085621
```

In [11]:

```python
### mode for Industry Growth Rate (%)
data['Industry Growth Rate (%)'].mode()
```

Out[11]:

```
0    3.22
Name: Industry Growth Rate (%), dtype: float64
```

In [14]:

```python
### mode for Industry Growth Rate (%)
data['Economic Index'].mode()
```

Out[14]:

```
0    126.1
Name: Economic Index, dtype: float64
```

In [15]:

```python
### median for Economic Index
data['Economic Index'].median()
```

Out[15]:

```
100.48
```

In [16]:

```python
### median for Industry Growth Rate (%)
data['Industry Growth Rate (%)'].median()
```

Out[16]:

```
4.76
```

In [17]:

```python
## 2nd movement business decision
### variance for Economic Index
data['Economic Index'].var()
```

Out[17]:

```
848.6359589847249
```

In [18]:

```python
### variance for Industry Growth Rate (%)
data['Industry Growth Rate (%)'].var()
```

Out[18]:

```
33.16043436683429
```

In [19]:

```python
### std for Economic Index
data['Economic Index'].std()
```

Out[19]:

```
29.13135697122132
```

In [20]:

```python
### std for Industry Growth Rate (%)
data['Industry Growth Rate (%)'].std()
```

Out[20]:

```
5.758509734890989
```

In [21]:

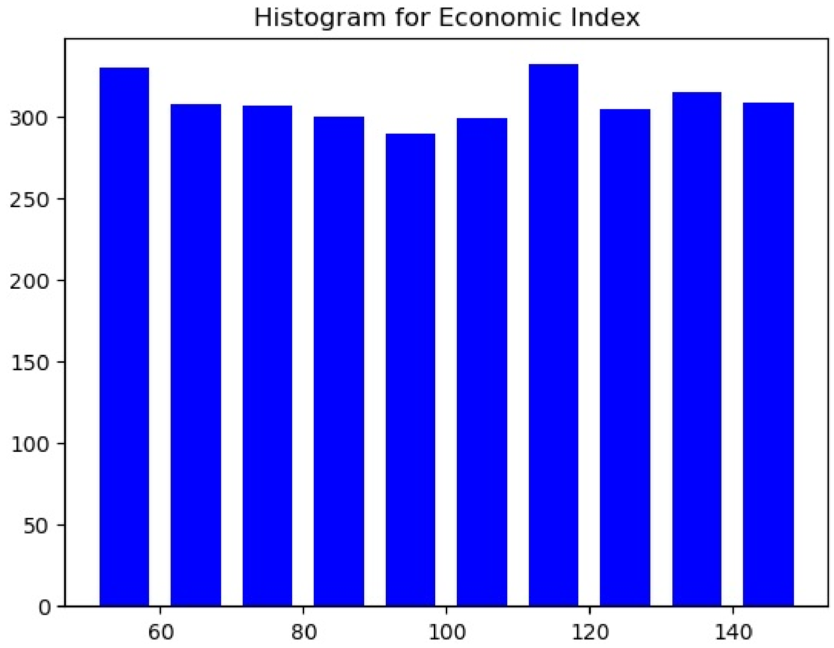
```python
### 3rd movement business decision
### here we find out the skewness of data and there shape to check weather is left skewed or right skewed and a
## skew for Economic Index
data['Economic Index'].skew()
```

Out[21]:

```
-0.010509100931631269
```

In [22]:

```python
#### histogram for Economic Index
plt.title('Histogram for Economic Index')
plt.hist(data['Economic Index'] ,histtype='bar' ,rwidth=0.7 ,color = 'blue')
plt.show()

### based on the skewness and shape of the histogram it symmetric skewed or well shape curve or normal data
```



In [23]:

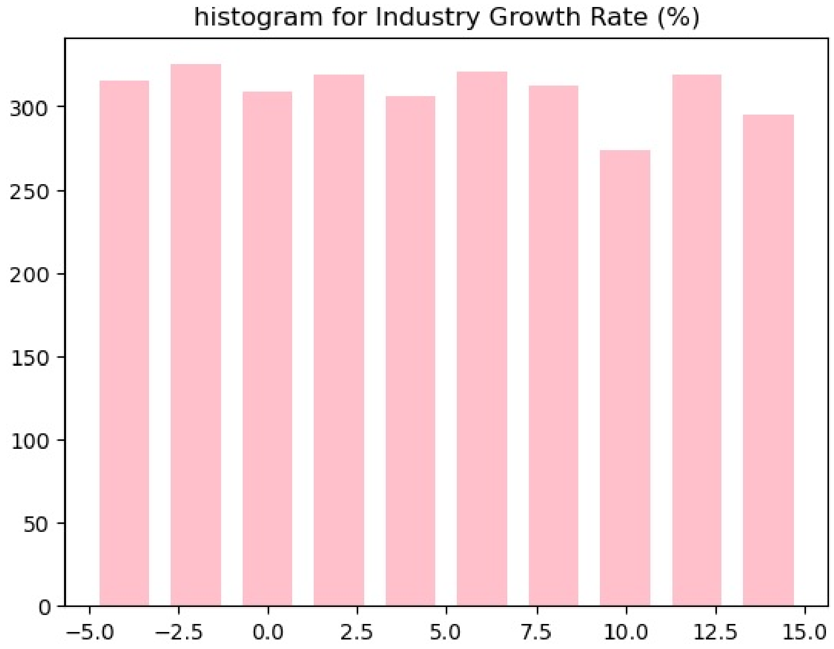
```
### skewness for Industry Growth Rate (%)
data['Industry Growth Rate (%)'].skew()
```

Out[23]:

```
0.03723079867303332
```

In [24]:

```
### histogram for Industry Growth Rate (%)
plt.title('histogram for Industry Growth Rate (%)')
plt.hist(data['Industry Growth Rate (%)'] ,histtype='bar' ,rwidth=0.7 ,color = 'pink')
plt.show()

### based on the skewness and the shape of an histogram it is well shape cure and normal distrubuted data with
```



In [25]:

```
### 4th movement business decision
#### in 4th movement business decision we find out the kurtosis
### kurtosis for Industry Growth Rate (%)
data['Industry Growth Rate (%)'].kurt()

### the Industry Growth Rate (%) has a negative kurtosis  and (k<3) so it playtokurtosis distrubution which have
```

Out[25]:

```
-1.198996184782077
```

In [26]:

```
### kurtosis for Economic Index
data['Economic Index'].kurt()

###  Economic Index has a negative kurtosis  and (k<3) so it playtokurtosis distrubution which have  wide peak
```

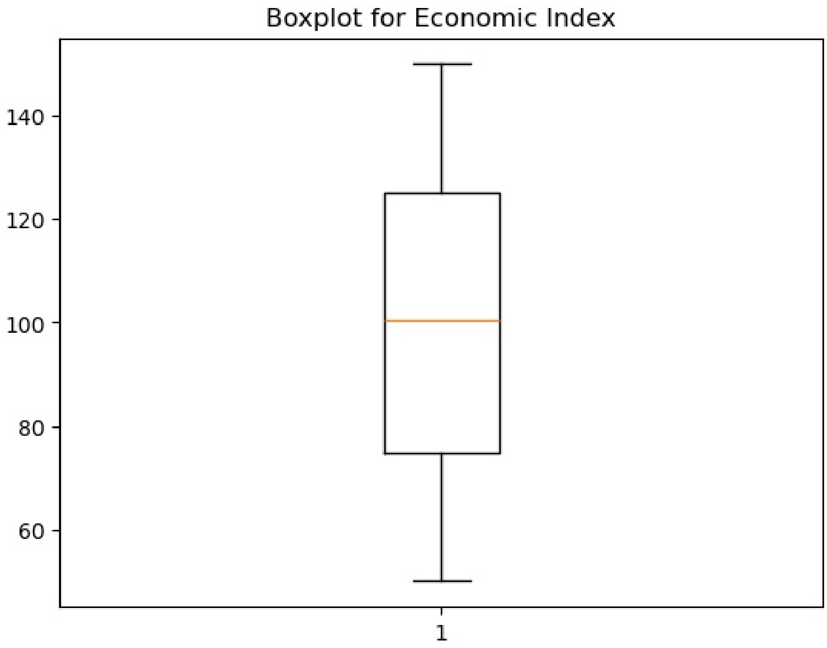
Out[26]:

```
-1.212746828575326
```

In [27]:

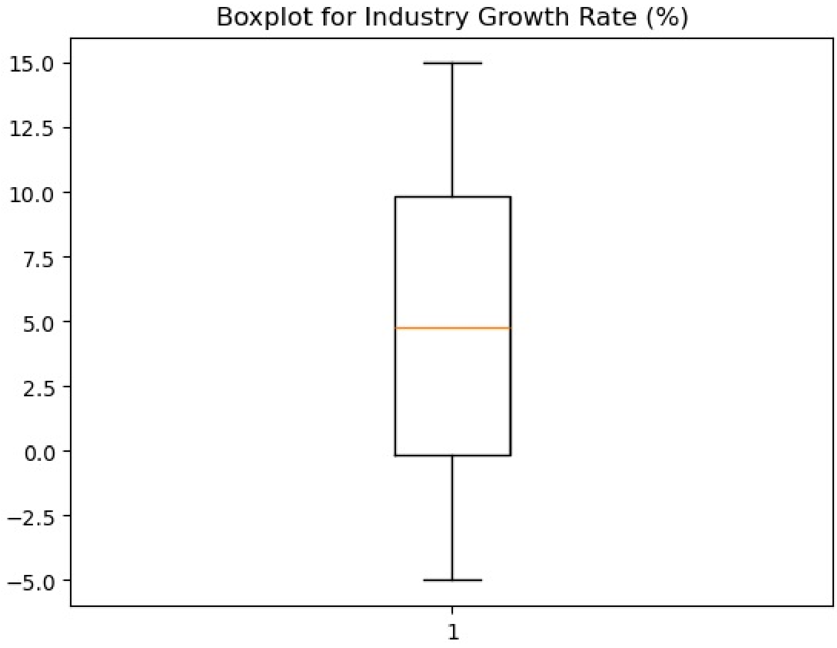
```
#### boxlot for Economic Index
```

```
plt.title('Boxplot for Economic Index')
plt.boxplot(data['Economic Index'])
plt.show()

### based on the boxplot the Economic Index don't have any outliers in data
```



In [28]:

```
#### boxplot for Industry Growth Rate (%)
plt.title('Boxplot for Industry Growth Rate (%)')
plt.boxplot(data['Industry Growth Rate (%)'])
plt.show()

### based on the boxplot the Industry Growth Rate (%) don't have any outliers in data
```



In [30]:

```
#### ckeck the data having any missing values

data.isna().sum()
### the data don't have any missing values
```

Out[30]:

```
Billing date                0
Variant                     0
Economic Index              0
Industry Growth Rate (%)    0
Seasonality Factor          0
dtype: int64
```

In [32]:

```
### data having any duplicate
print("Duplicates:", data.duplicated().sum())
```

```
Duplicates: 0
```

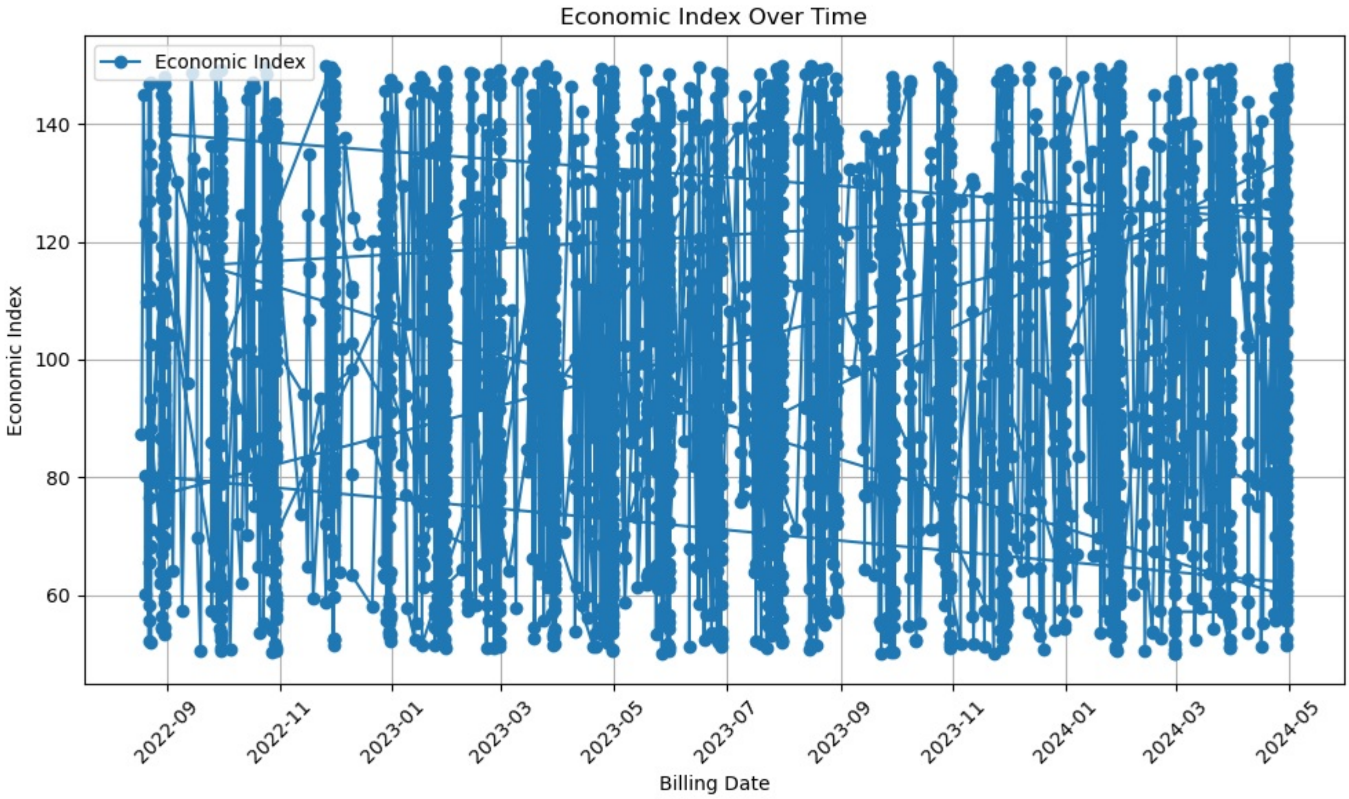
In [36]:

```
# Strip any spaces from column names
```

```
data.columns = data.columns.str.strip()

# Convert 'Billing date' to datetime format
data['Billing date'] = pd.to_datetime(data['Billing date'])
```

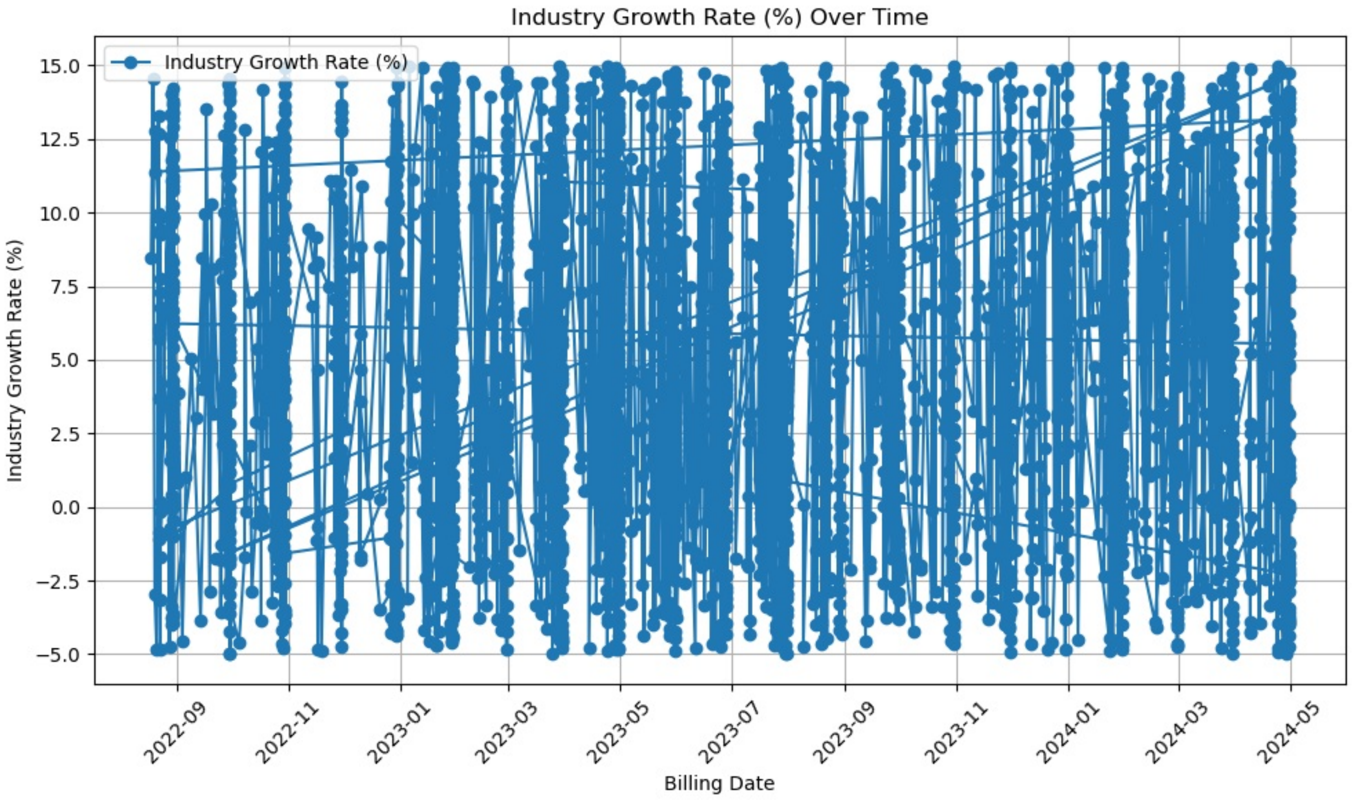
In [37]:

```
# Plotting the time series (Economic Index over Billing Date)
plt.figure(figsize=(10, 6))
plt.plot(data['Billing date'], data['Economic Index'], marker='o', label='Economic Index')
plt.xlabel('Billing Date')
plt.ylabel('Economic Index')
plt.title('Economic Index Over Time')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.show()
```



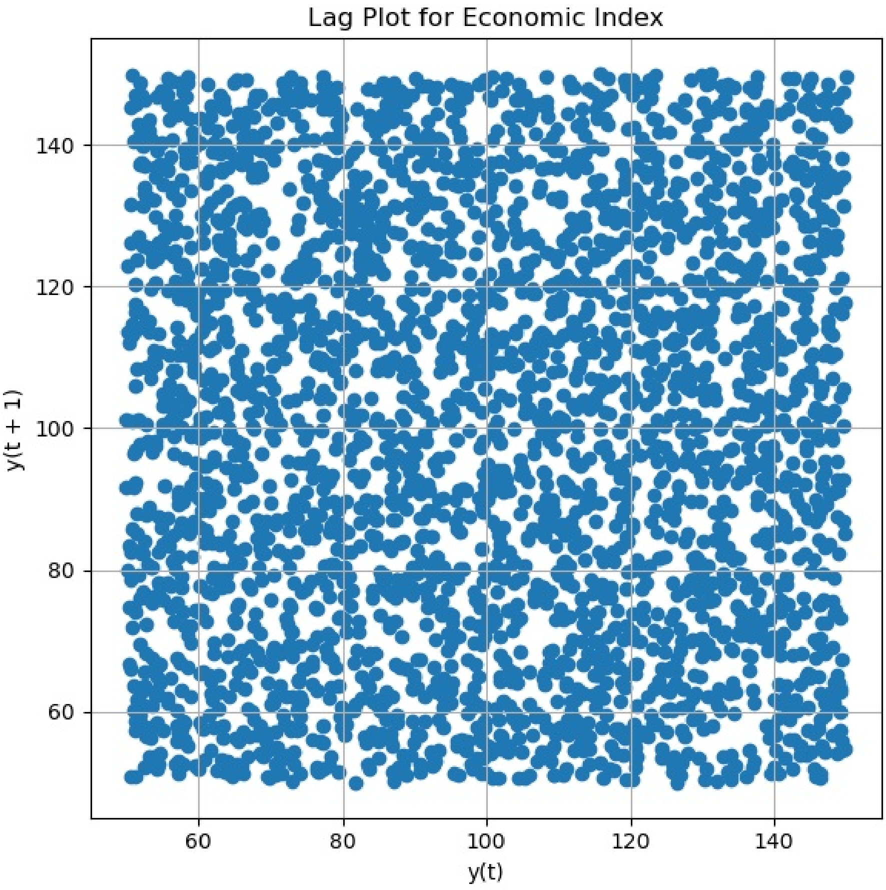
In [39]:

```
# Plotting the time series (Industry Growth Rate over Billing Date)
plt.figure(figsize=(10, 6))
plt.plot(data['Billing date'], data['Industry Growth Rate (%)'], marker='o', label='Industry Growth Rate (%)')
plt.xlabel('Billing Date')
plt.ylabel('Industry Growth Rate (%)')
plt.title('Industry Growth Rate (%) Over Time')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend()
plt.show()
```



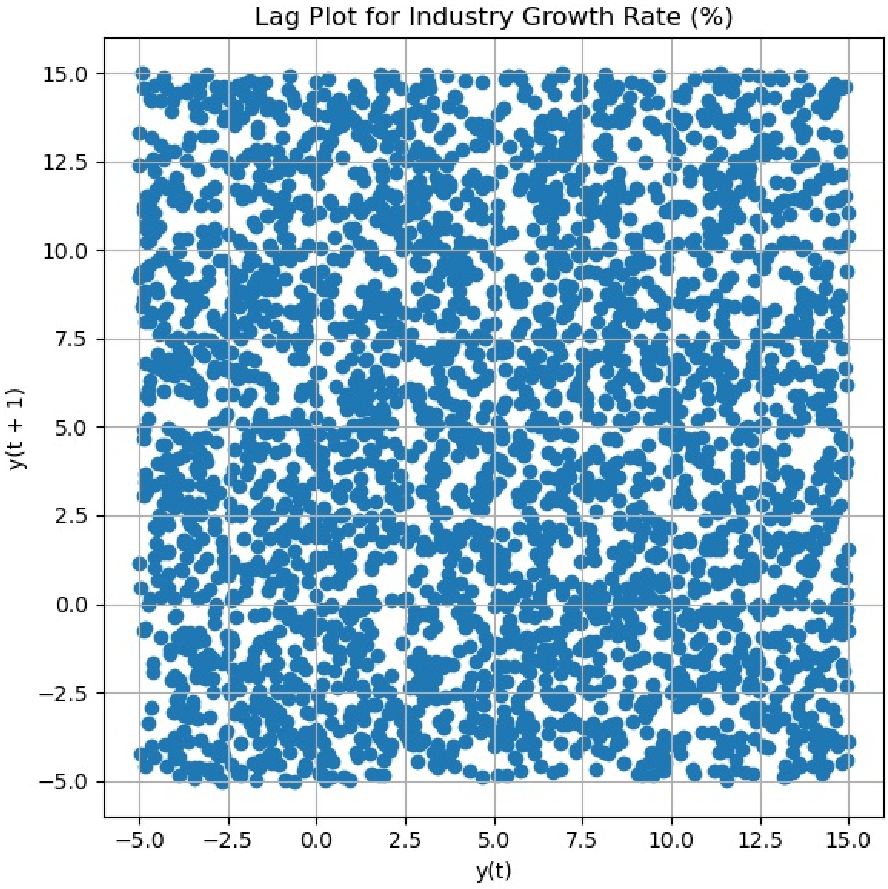
In [40]:

```
#### lag plot Economic Index
from pandas.plotting import lag_plot
# Lag plot for Economic Index
plt.figure(figsize=(6, 6))
lag_plot(data['Economic Index'])
plt.title('Lag Plot for Economic Index')
plt.grid(True)
plt.tight_layout()
plt.show()
```



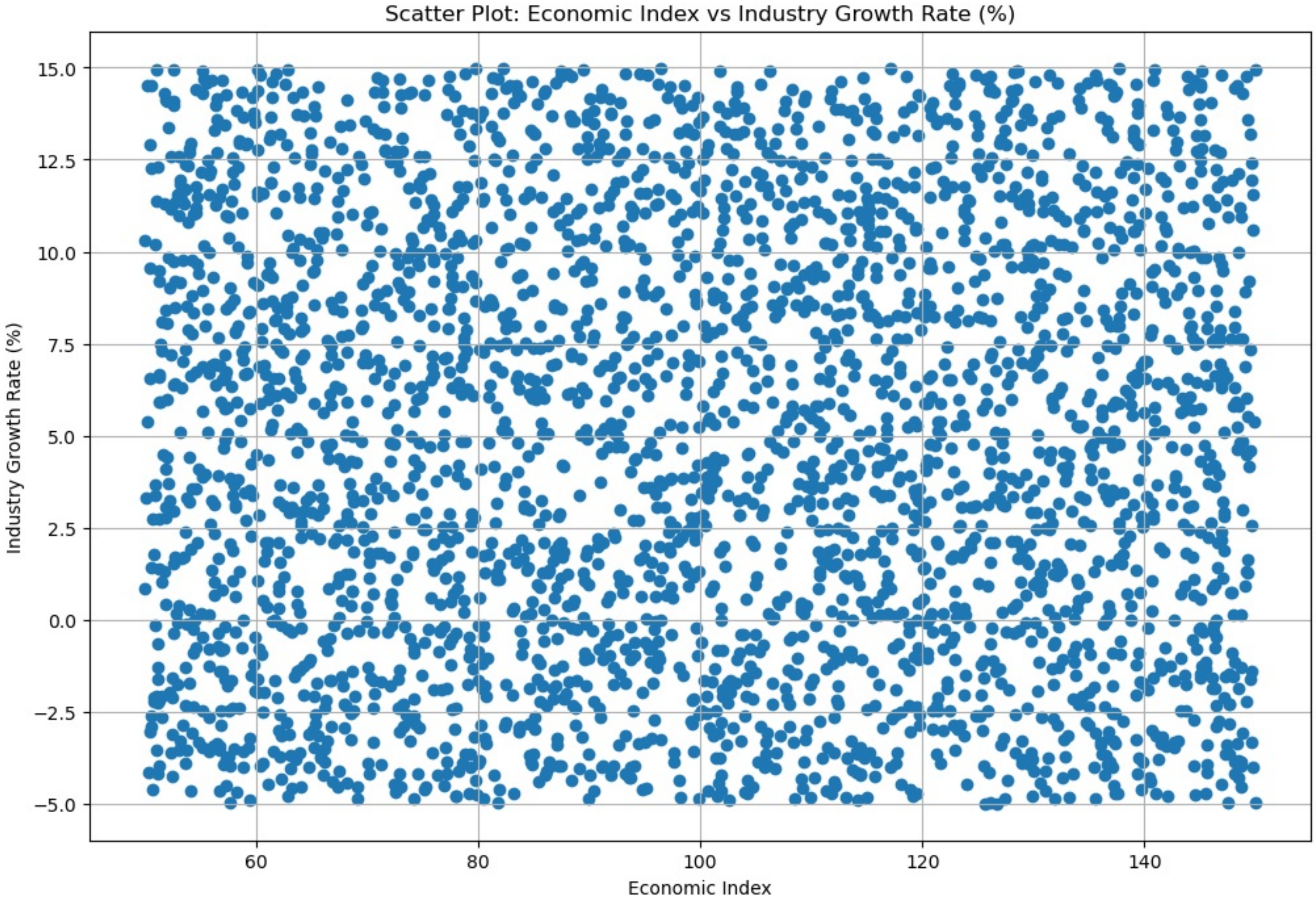
In [41]:

```
### lag plot for Industry Growth Rate (%)
plt.figure(figsize=(6, 6))
lag_plot(data['Industry Growth Rate (%)'])
plt.title('Lag Plot for Industry Growth Rate (%)')
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [44]:

```
# Scatter plot between Economic Index and Industry Growth Rate (%)
plt.figure(figsize=(12, 8))

# Plot the actual column values, not the list of column names
plt.scatter(x=data['Economic Index'], y=data['Industry Growth Rate (%)'])

# Labeling the axes and setting a title
plt.xlabel('Economic Index')
plt.ylabel('Industry Growth Rate (%)')
plt.title('Scatter Plot: Economic Index vs Industry Growth Rate (%)')

# Display the plot
plt.grid(True)
plt.show()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js