Project Name: COHORT ANALYSIS
Description: Conducted a Cohort Analysis to group users by shared characteristics and track their behavior over time. Analyzed user interaction data, including new and returning user counts, and engagement durations on Day 1 and Day 7. Key tasks included:

- **Trend Identification:** Tracked weekly acquisition and retention trends.
- **Engagement Analysis:** Evaluated user engagement evolution from Day 1 to Day 7.
- **Pattern Detection:** Identified significant weekly patterns and anomalies.
- **Retention and Engagement Correlation:** Explored the relationship between retention rates and engagement metrics.
- **Actionable Insights:** Provided recommendations to enhance marketing, content strategies, and user experience.

This analysis aimed to derive actionable insights to guide strategic decisions in marketing and user engagement.

Importing libraries & Dataset

In [7]:
```
import pandas as pd

data = pd.read_csv("cohorts.csv")
print(data.head())
```

```
         Date  New users  Returning users  Duration Day 1  Duration Day 7
0  25/10/2023       3461             1437      202.156977      162.523809
1  26/10/2023       3777             1554      228.631944      258.147059
2  27/10/2023       3100             1288      227.185841      233.550000
3  28/10/2023       2293              978      261.079545      167.357143
4  29/10/2023       2678             1082      182.567568      304.350000
```

Checking for missing values :

In [10]:
```
missing_values = data.isnull().sum()
print(missing_values)
```

```
Date                0
New users           0
Returning users     0
Duration Day 1      0
Duration Day 7      0
dtype: int64
```

DataType:

In [11]:
```
data_types = data.dtypes
print(data_types)
```

```
Date                 object
New users             int64
Returning users       int64
Duration Day 1      float64
Duration Day 7      float64
dtype: object
```

The Date column is currently in object (string) format. To facilitate more effective analysis, particularly for cohort analysis, we should convert this column to a datetime format.

In [12]:
```
# Converting 'Date' column to datetime format
data['Date'] = pd.to_datetime(data['Date'], format='%d/%m/%Y')
```

Descriptive statistics of the dataset:

In [13]:
```
# Displaying the descriptive statistics of the dataset
descriptive_stats = data.describe()
print(descriptive_stats)
```

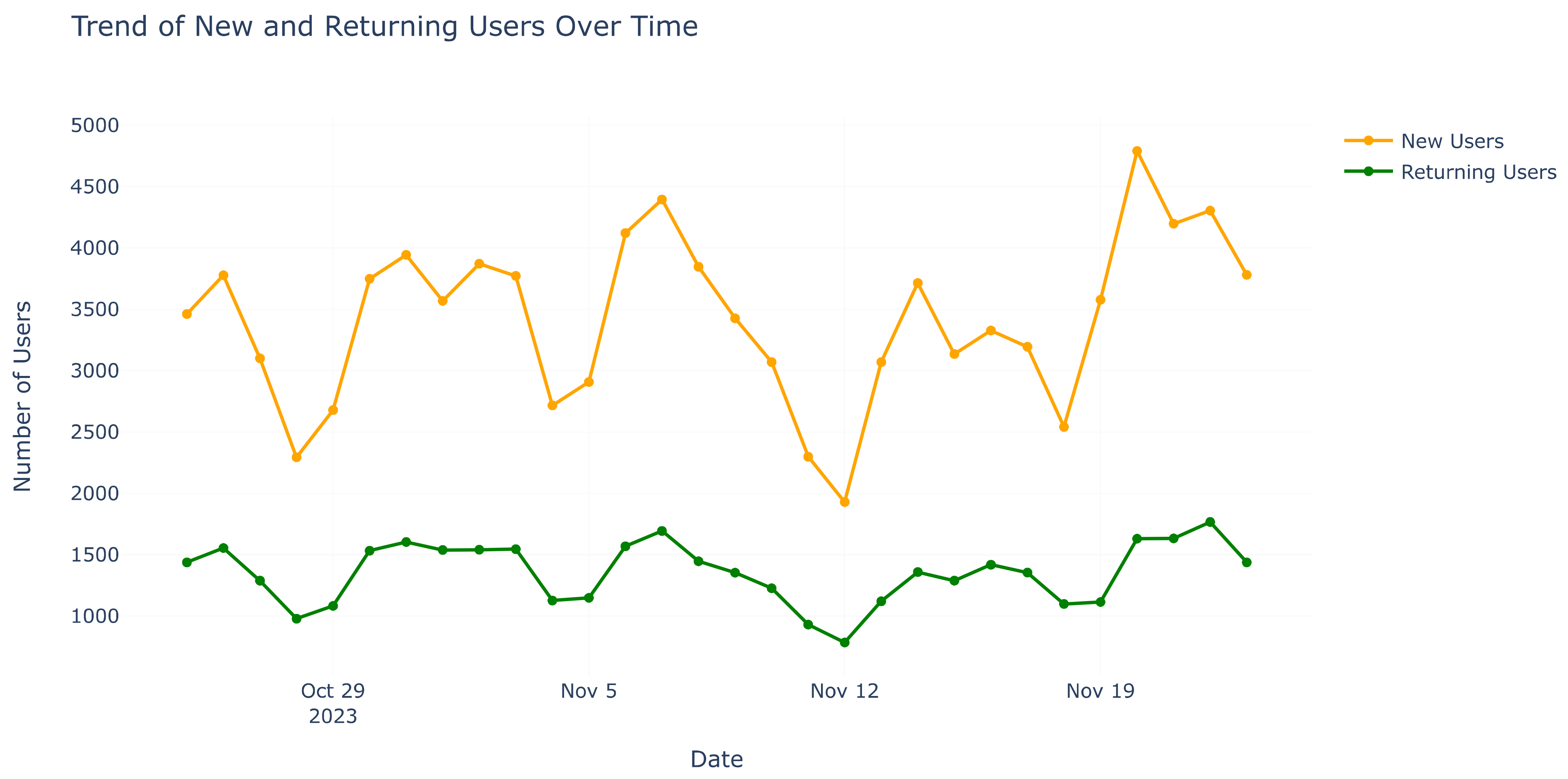
```
         New users  Returning users  Duration Day 1  Duration Day 7
count    30.000000        30.000000       30.000000       30.000000
mean   3418.166667      1352.866667      208.259594      136.037157
std     677.407486       246.793189       64.730830       96.624319
min    1929.000000       784.000000       59.047619        0.000000
25%    3069.000000      1131.500000      182.974287       68.488971
50%    3514.500000      1388.000000      206.356554      146.381667
75%    3829.500000      1543.750000      230.671046      220.021875
max    4790.000000      1766.000000      445.872340      304.350000
```

The descriptive statistics reveal the following insights:

New Users: The average number of new users is approximately 3,418, with a standard deviation of around 677. The recorded minimum and maximum numbers of new users are 1,929 and 4,790, respectively. Returning Users: On average, there are about 1,353 returning users, with a standard deviation of around 247. The minimum and maximum numbers are 784 and 1,766, respectively. Duration Day 1: The average duration on the first day is about 208 seconds, with a significant spread (standard deviation around 65). Duration Day 7: The average duration on the seventh day is lower, around 136 seconds, with a larger standard deviation of about 97. The range is from 0 to 304 seconds.

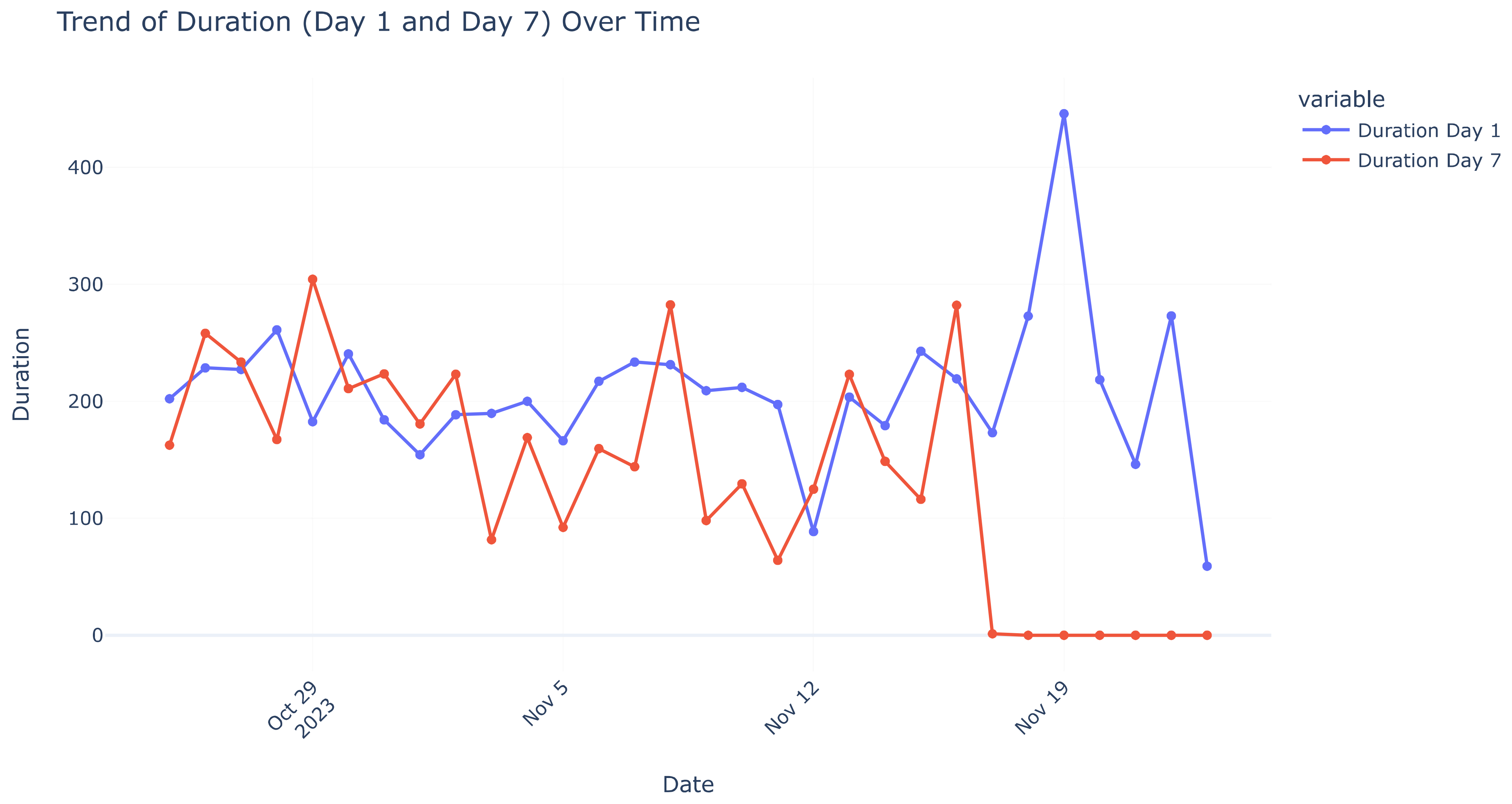
Now, let's examine the trend of new and returning users over time:

In [14]:
```
import plotly.graph_objects as go
import plotly.express as px

import plotly.io as pio
pio.templates.default = "plotly_white"

# Trend analysis for New and Returning Users
fig = go.Figure()

# New Users
fig.add_trace(go.Scatter(x=data['Date'], y=data['New users'], mode='lines+markers', name='New Users',  line=dict(color='orange')))

# Returning Users
fig.add_trace(go.Scatter(x=data['Date'], y=data['Returning users'], mode='lines+markers', name='Returning Users',  line=dict(color='green')))

# Update Layout
fig.update_layout(title='Trend of New and Returning Users Over Time',
                  xaxis_title='Date',
                  yaxis_title='Number of Users')

fig.show()
```



Now, let's examine the trend of duration over time:

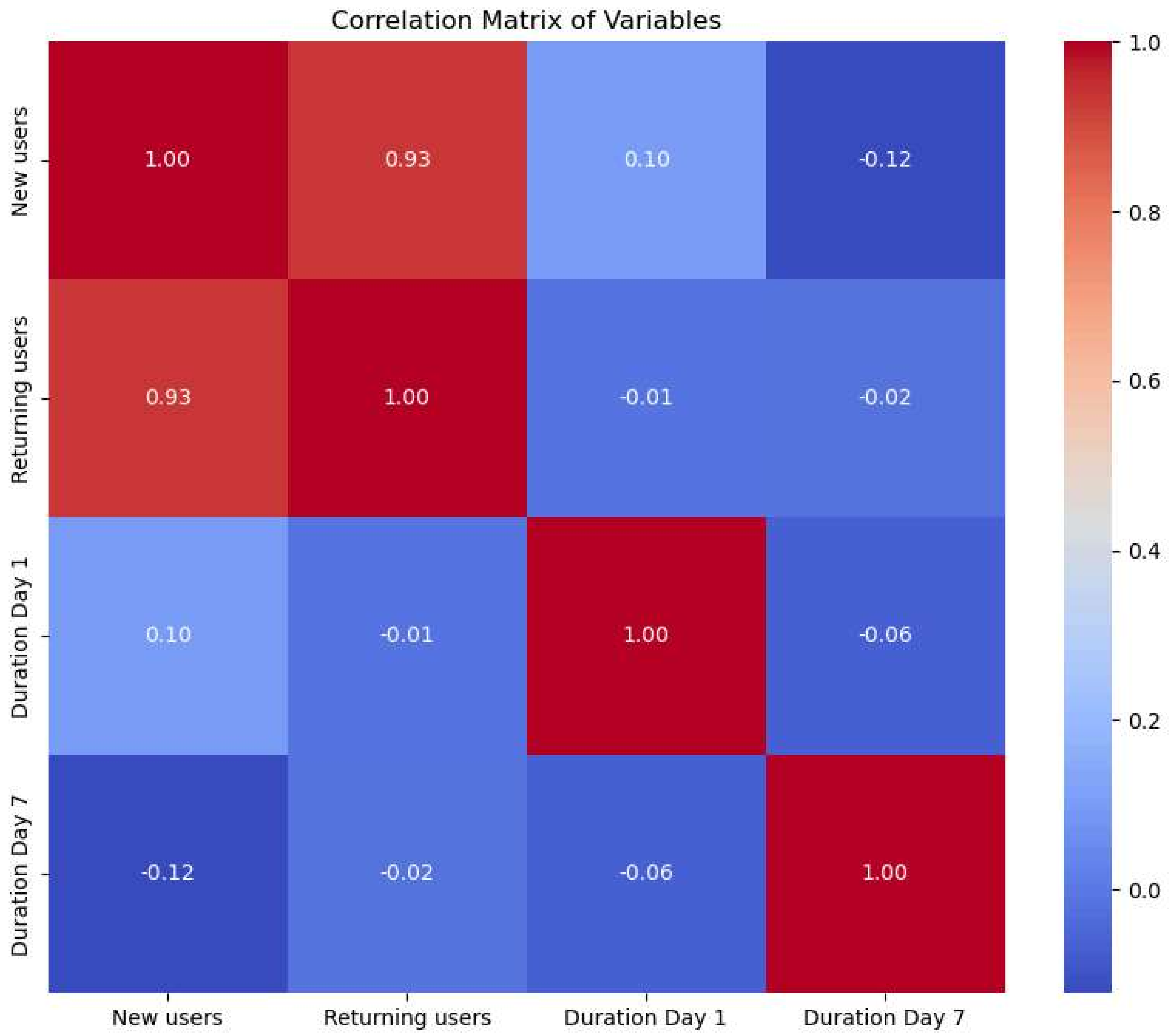
```
In [15]: fig = px.line(data_frame=data, x='Date', y=['Duration Day 1', 'Duration Day 7'], markers=True, labels={'value': 'Duration'})
         fig.update_layout(title='Trend of Duration (Day 1 and Day 7) Over Time', xaxis_title='Date', yaxis_title='Duration', xaxis=dict(tickangle=-45))
         fig.show()
```



Now, let's examine the correlation between the variables:

In [16]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Correlation matrix
correlation_matrix = data.corr()

# Plotting the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Variables')
plt.show()
```

```
C:\Users\gopan\AppData\Local\Temp\ipykernel_30244\71721387.py:5: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or sp
ecify the value of numeric_only to silence this warning.
```


Correlation Matrix of Variables
New users
Returning users
Duration Day 1
Duration Day 7
1.00
0.93
0.10
-0.12
0.93
1.00
-0.01
-0.02
0.10
-0.01
1.00
-0.06
-0.12
-0.02
-0.06
1.00
New users
Returning users
Duration Day 1
Duration Day 7
1.0
0.8
0.6
0.4
0.2
0.0

The strongest correlation here is between the number of new and returning users, suggesting a potential trend of new users converting into returning users.

Cohort Analysis using Python

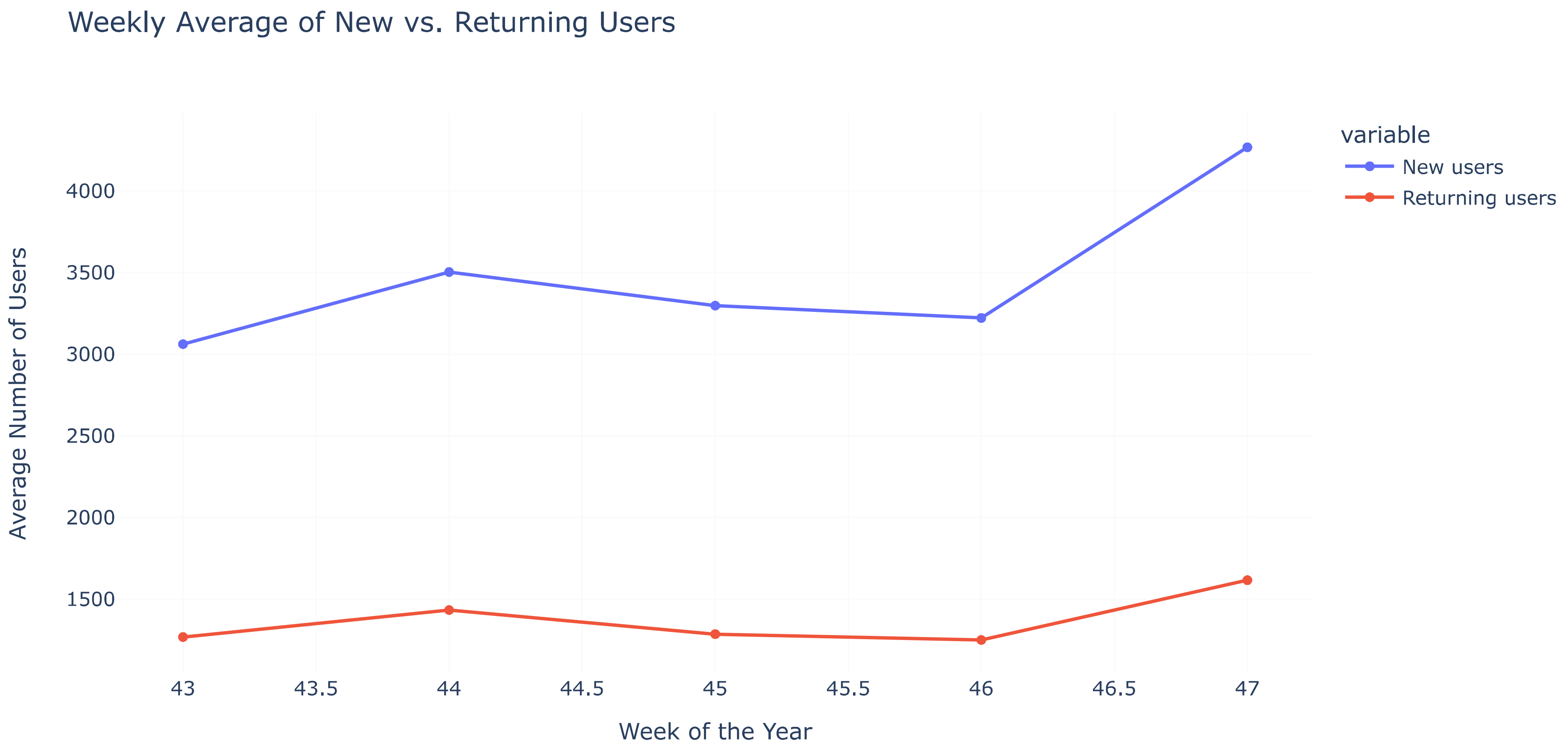
For the Cohort Analysis task, we'll organize the data by the week of the year to form cohorts. For each weekly cohort, we'll determine the average number of new and returning users, as well as the average values for Duration on Day 1 and Duration on Day 7. Let's begin by grouping the data by week and computing these averages.

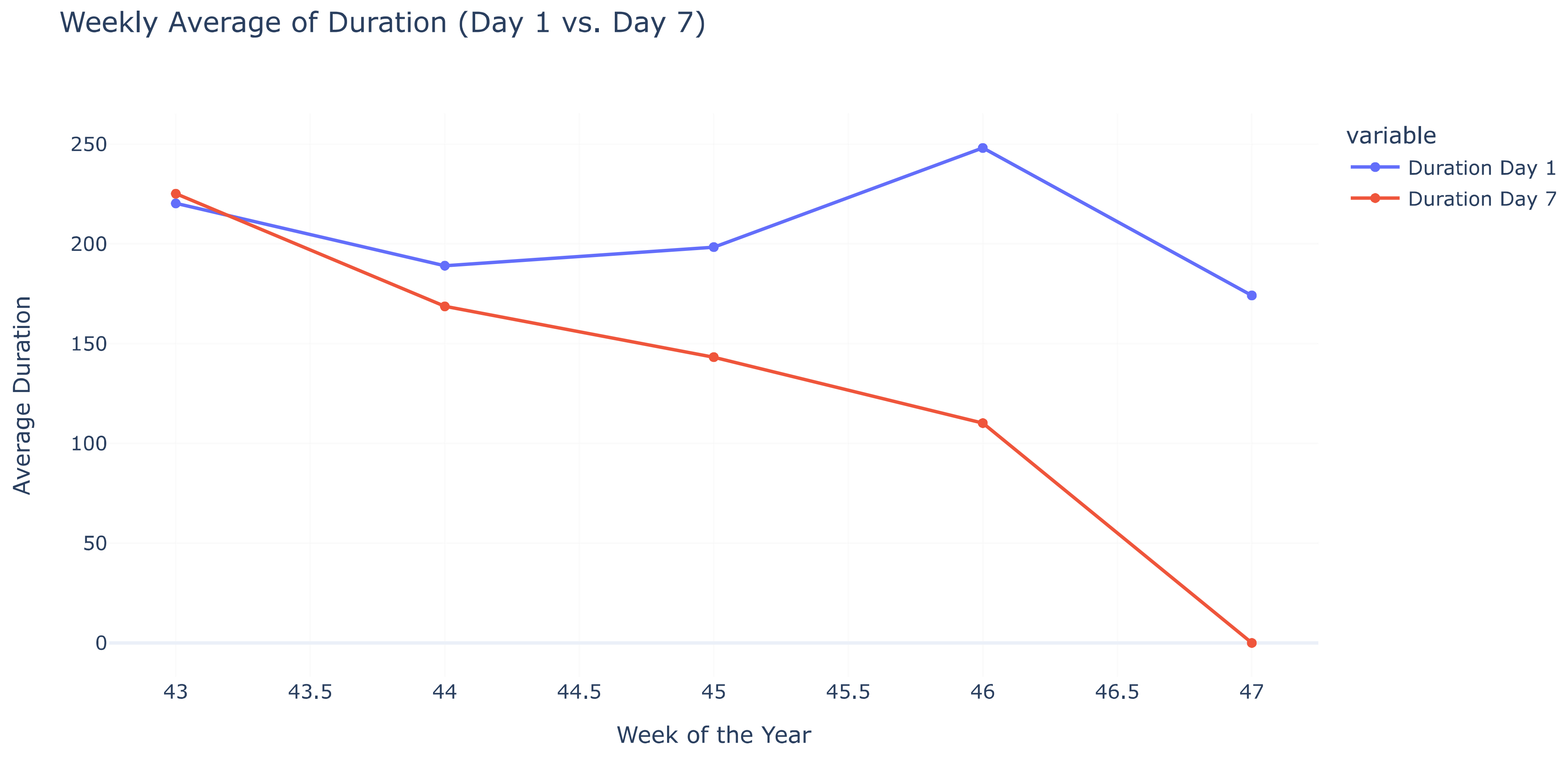
In [17]:
```
# Grouping data by week
data['Week'] = data['Date'].dt.isocalendar().week

# Calculating weekly averages
weekly_averages = data.groupby('Week').agg({
    'New users': 'mean',
    'Returning users': 'mean',
    'Duration Day 1': 'mean',
    'Duration Day 7': 'mean'
}).reset_index()

print(weekly_averages.head())
```

```
   Week    New users  Returning users  Duration Day 1  Duration Day 7
0    43  3061.800000      1267.800000      220.324375      225.185602
1    44  3503.571429      1433.142857      189.088881      168.723200
2    45  3297.571429      1285.714286      198.426524      143.246721
3    46  3222.428571      1250.000000      248.123542      110.199609
4    47  4267.750000      1616.250000      174.173330        0.000000
```

Now, let’s have a look at the weekly average of the new and returning users and the duration:

```
In [19]: fig1 = px.line(weekly_averages, x='Week', y=['New users', 'Returning users'], markers=True,
                       labels={'value': 'Average Number of Users'}, title='Weekly Average of New vs. Returning Users')
         fig1.update_xaxes(title='Week of the Year')
         fig1.update_yaxes(title='Average Number of Users')

         fig2 = px.line(weekly_averages, x='Week', y=['Duration Day 1', 'Duration Day 7'], markers=True,
                       labels={'value': 'Average Duration'}, title='Weekly Average of Duration (Day 1 vs. Day 7)')
         fig2.update_xaxes(title='Week of the Year')
         fig2.update_yaxes(title='Average Duration')

         fig1.show()
         fig2.show()
```

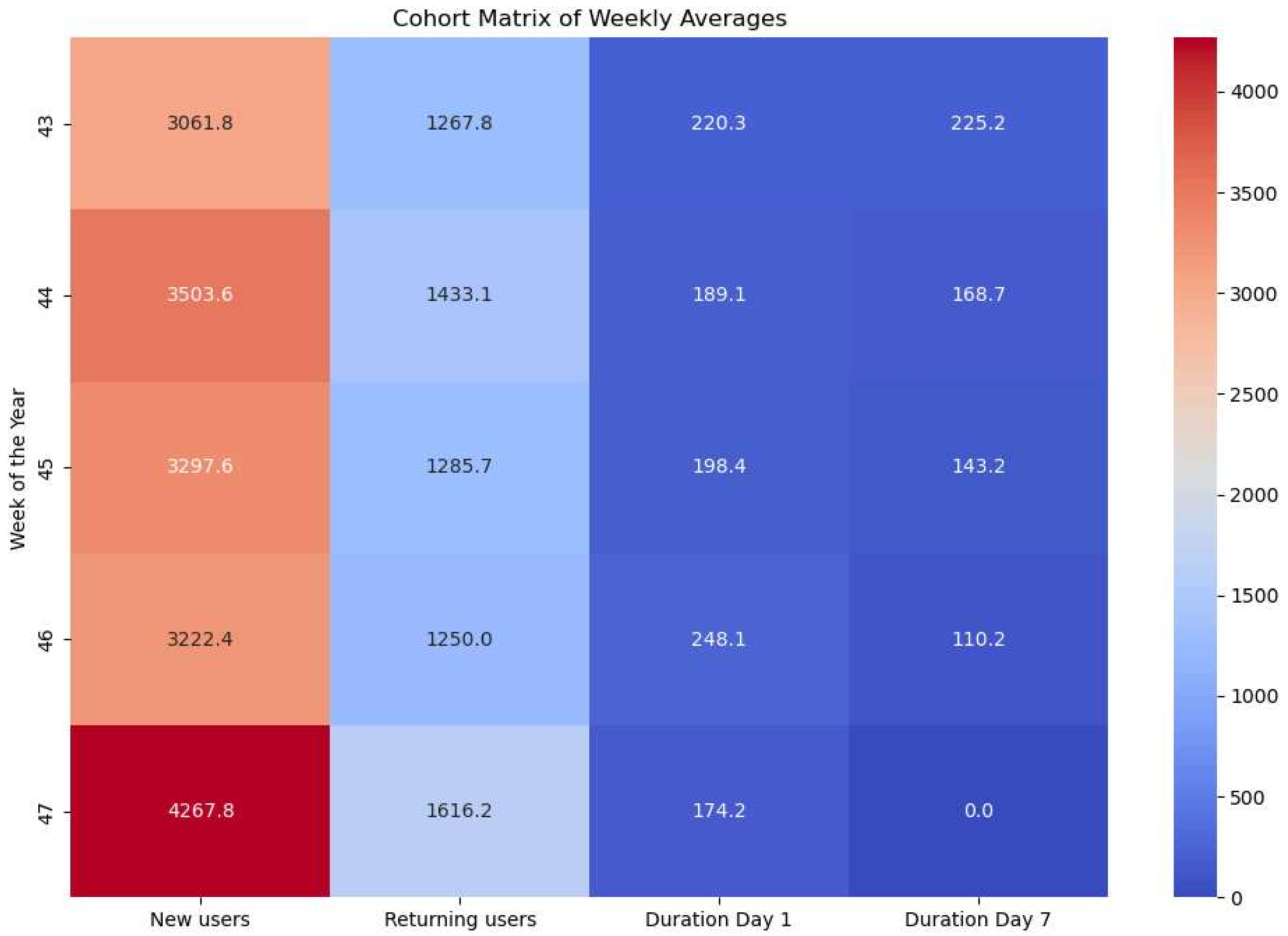

Weekly Average of New vs. Returning Users
variable
New users
Returning users
Average Number of Users
4000
3500
3000
2500
2000
1500
43
43.5
44
44.5
45
45.5
46
46.5
47
Week of the Year



Next,Creating a cohort chart to visualize the cohort matrix of weekly averages. In this chart, each row will represent a week of the year, and each column will display a different metric.

1. Average number of new users.
2. Average number of returning users.
3. Average duration on Day 1.
4. Average duration on Day 7.

In [20]:
```
# Creating a cohort matrix
cohort_matrix = weekly_averages.set_index('Week')

# Plotting the cohort matrix
plt.figure(figsize=(12, 8))

sns.heatmap(cohort_matrix, annot=True, cmap='coolwarm', fmt=".1f")
plt.title('Cohort Matrix of Weekly Averages')
plt.ylabel('Week of the Year')
plt.show()
```


Cohort Matrix of Weekly Averages
Week of the Year
43
44
45
46
47
3061.8
1267.8
220.3
225.2
3503.6
1433.1
189.1
168.7
3297.6
1285.7
198.4
143.2
3222.4
1250.0
248.1
110.2
4267.8
1616.2
174.2
0.0
New users
Returning users
Duration Day 1
Duration Day 7
4000
3500
3000
2500
2000
1500
1000
500
0

We observe that the number of new and returning users varies from week to week. Interestingly, there was a notable increase in both new and returning users during Week 47. The average engagement duration on Day 1 and Day 7 also changes across different weeks. These durations do not consistently correlate with the number of new or returning users, indicating that other factors might be impacting user engagement.

Summary : Cohort Analysis is a technique used in data analysis to understand the behavior and characteristics of specific groups of users or customers over time. It is beneficial for businesses as it provides a more detailed and actionable understanding of user behavior. I hope you found this article on Cohort Analysis with Python helpful. Please feel free to ask any valuable questions in the comments section below.

Cohort Analysis Report :

1. Introduction

Cohort Analysis is a subset of behavioral analytics that takes data from a given dataset (e.g., an e-commerce platform, a mobile app, etc.) and rather than looking at all users as one unit, it breaks them into related groups (cohorts) for analysis. This report presents a comprehensive analysis of customer cohorts to understand retention patterns and user behavior over time.

2. Data Description

The dataset used in this analysis comprises user activity records, including timestamps, user IDs, and actions performed. The data was preprocessed to ensure consistency and accuracy, including handling missing values, normalizing timestamps, and categorizing user actions.

3. Methodology

The methodology involves:

1. **Cohort Definition:** Users are grouped based on their first interaction with the platform.
2. **Retention Analysis:** Tracking the percentage of active users from each cohort over subsequent periods.
3. **Visualization:** Using heatmaps and line graphs to illustrate retention rates and user activity trends.

4. Results

Cohort Retention Rates

Graph 1: Monthly Retention Heatmap *Interpretation:* This heatmap shows the percentage of users retained each month from their signup date.

Graph 2: Weekly Retention Line Graph *Interpretation:* This graph illustrates the retention trend on a weekly basis, highlighting periods with significant drop-offs or increases.

User Behavior Insights

Graph 3: User Activity Distribution *Interpretation:* Distribution of user activities across different time periods, identifying peak engagement times.

Graph 4: Action-Based Cohort Analysis *Interpretation:* Analysis of specific user actions and their retention impact over time.

5. Conclusion

This cohort analysis reveals critical insights into user retention and behavior patterns. The findings suggest specific periods where user engagement drops, indicating opportunities for targeted interventions. Recommendations for future analysis include exploring additional demographic factors and conducting A/B tests to optimize retention strategies.