In [1]:

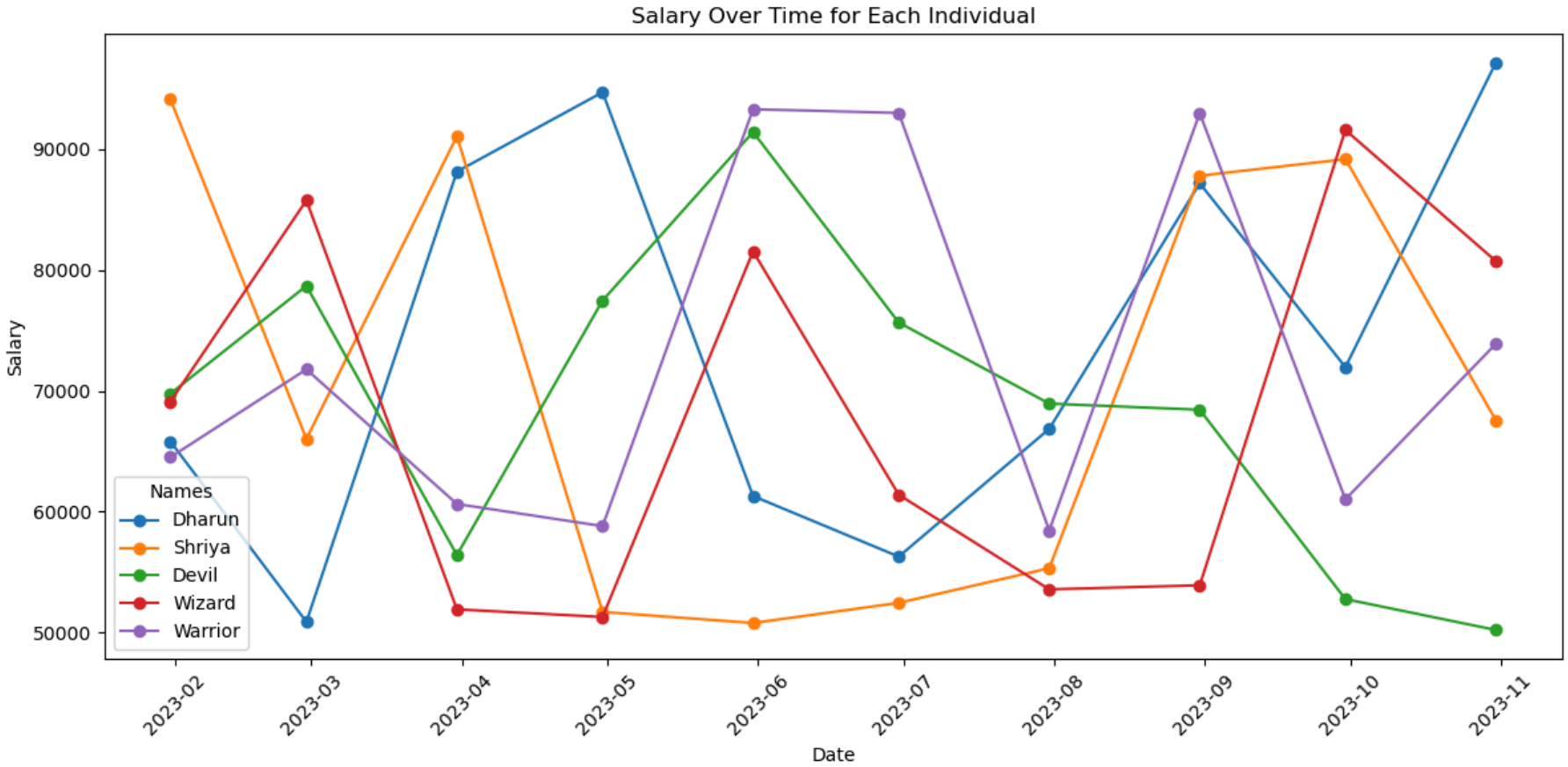
```python
#TIME SERIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set random seed for reproducibility
np.random.seed(42)

# Generate a time series data
dates = pd.date_range(start='2023-01-01', periods=10, freq='M')
names = ['Dharun', 'Shriya', 'Devil', 'Wizard', 'Warrior']
data = {
    'Date': dates
}

# Generate random salary data for each name over the date range
for name in names:
    data[name] = np.random.randint(50000, 100000, size=len(dates))

# Create a DataFrame
df_timeseries = pd.DataFrame(data)

# Set Date as the index
df_timeseries.set_index('Date', inplace=True)

# Display the DataFrame
print("Time Series DataFrame:")
print(df_timeseries)
print("\n")  # New line for better readability
```

Time Series DataFrame:

Date	Dharun	Shriya	Devil	Wizard	Warrior
2023-01-31	65795	94131	69769	69118	64502
2023-02-28	50860	66023	78693	85773	71777
2023-03-31	88158	91090	56396	51899	60627
2023-04-30	94732	51685	77480	51267	58792
2023-05-31	61284	50769	91434	81551	93323
2023-06-30	56265	52433	75658	61394	93021
2023-07-31	66850	55311	68942	53556	58433
2023-08-31	87194	87819	68431	53890	93001
2023-09-30	71962	89188	52747	91606	61016
2023-10-31	97191	67568	50189	80740	73897

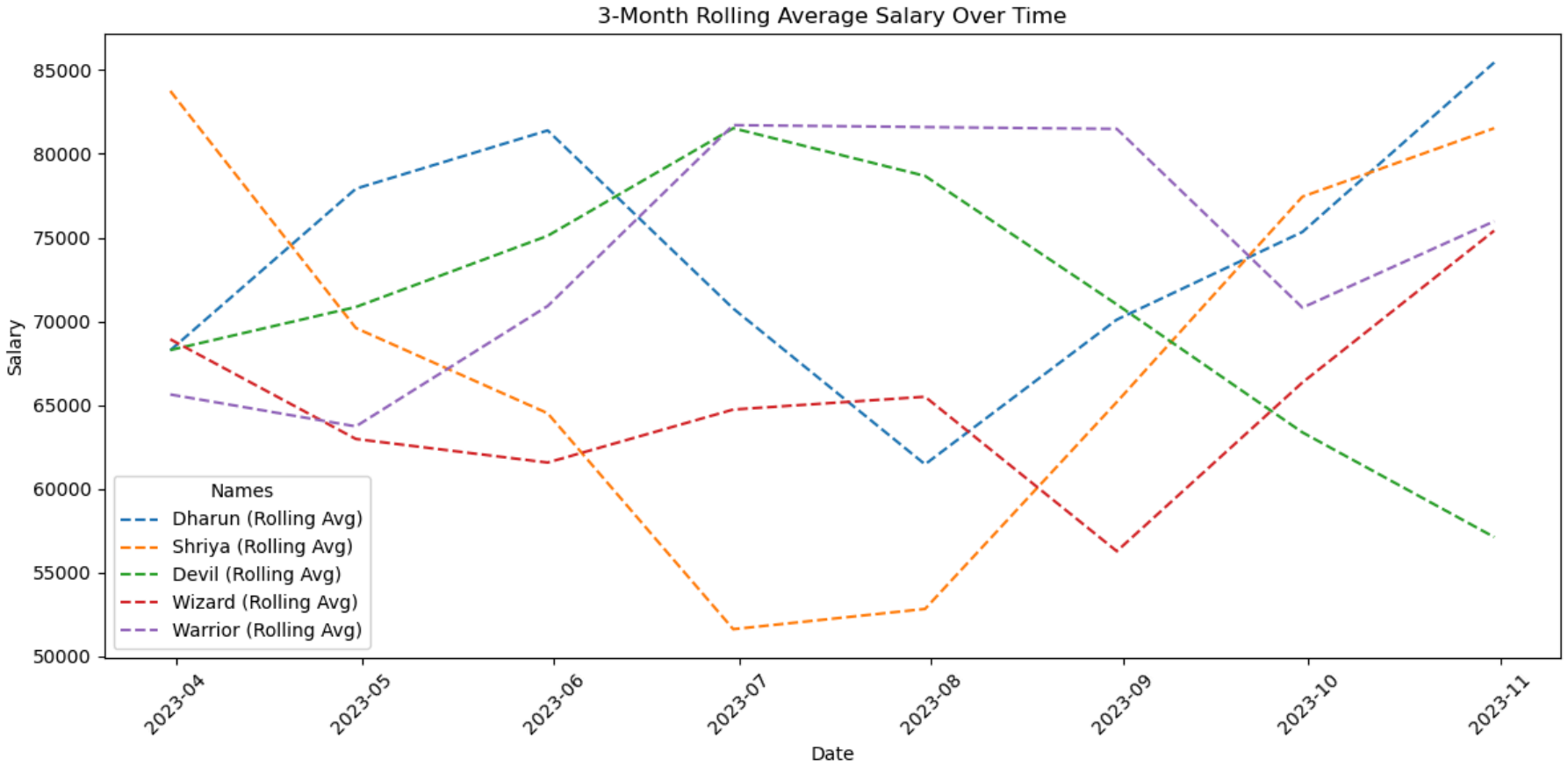
In [2]:
```python
# 1. Line Plot: Salary Over Time
print("1. Line Plot: Salary Over Time")
plt.figure(figsize=(12, 6))
for name in names:
    plt.plot(df_timeseries.index, df_timeseries[name], marker='o', label=name)

plt.title('Salary Over Time for Each Individual')
plt.xlabel('Date')
plt.ylabel('Salary')
plt.xticks(rotation=45)
plt.legend(title='Names')
plt.tight_layout()
plt.show()
```

```
1. Line Plot: Salary Over Time
```


Salary Over Time for Each Individual
Salary
90000
80000
70000
60000
50000
Names
Dharun
Shriya
Devil
Wizard
Warrior
2023-02
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
2023-09
2023-10
2023-11
Date

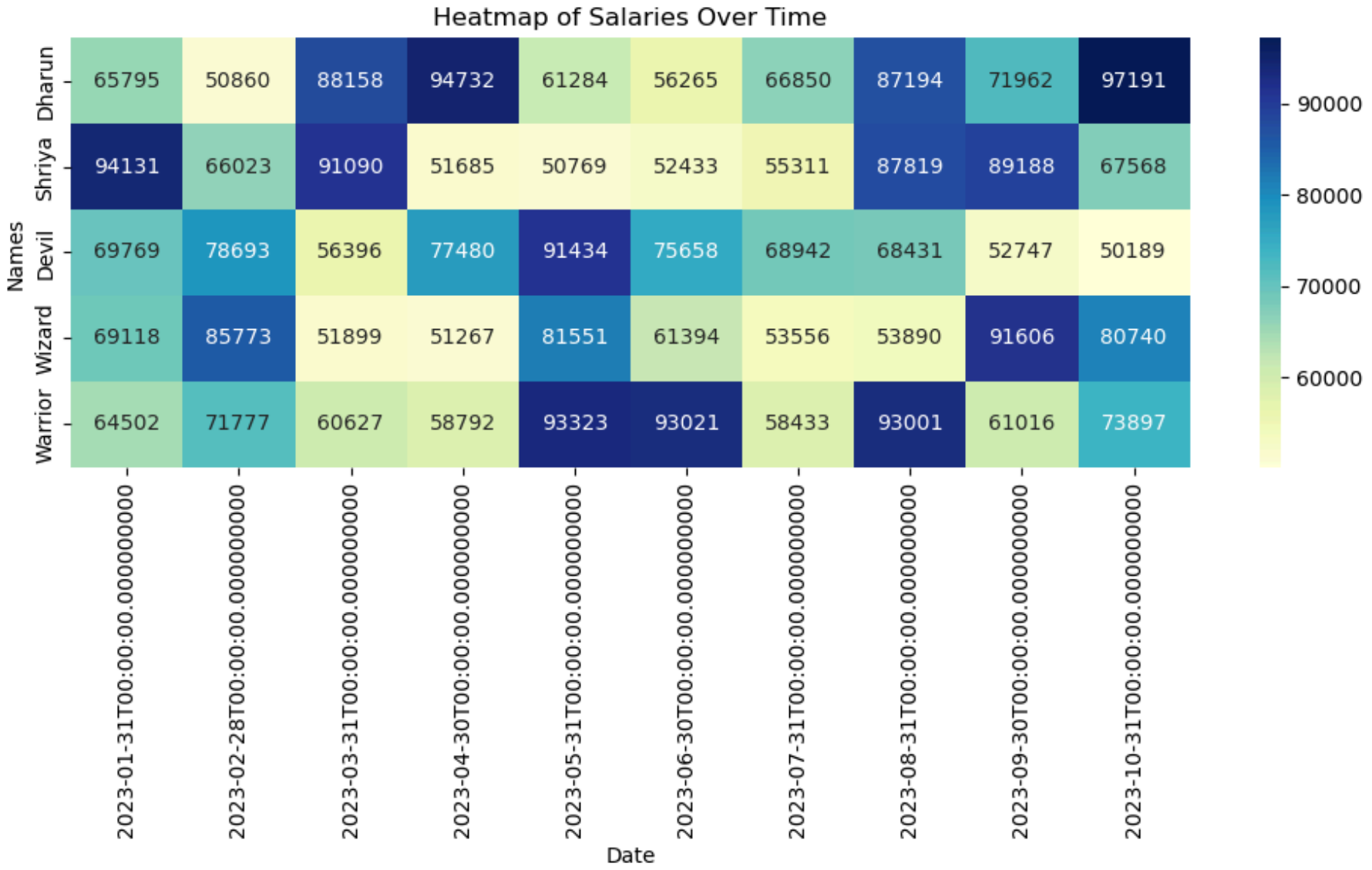
In [3]:
```
# 2. Rolling Average Plot
print("2. Rolling Average Plot")
plt.figure(figsize=(12, 6))
for name in names:
    rolling_avg = df_timeseries[name].rolling(window=3).mean()  # 3-month rolling average
    plt.plot(df_timeseries.index, rolling_avg, label=f'{name} (Rolling Avg)', linestyle='--')

plt.title('3-Month Rolling Average Salary Over Time')
plt.xlabel('Date')
plt.ylabel('Salary')
plt.xticks(rotation=45)
plt.legend(title='Names')
plt.tight_layout()
plt.show()
```

```
2. Rolling Average Plot
```


3-Month Rolling Average Salary Over Time
Salary
85000
80000
75000
70000
65000
60000
55000
50000
Names
Dharun (Rolling Avg)
Shriya (Rolling Avg)
Devil (Rolling Avg)
Wizard (Rolling Avg)
Warrior (Rolling Avg)
2023-04
2023-05
2023-06
2023-07
2023-08
2023-09
2023-10
2023-11
Date

In [4]:
```
# 3. Heatmap of Time Series Data
print("3. Heatmap of Time Series Data")
plt.figure(figsize=(10, 6))
sns.heatmap(df_timeseries.T, cmap='YlGnBu', annot=True, fmt=".0f")
plt.title('Heatmap of Salaries Over Time')
plt.xlabel('Date')
plt.ylabel('Names')
plt.tight_layout()
plt.show()
```

```
3. Heatmap of Time Series Data
```

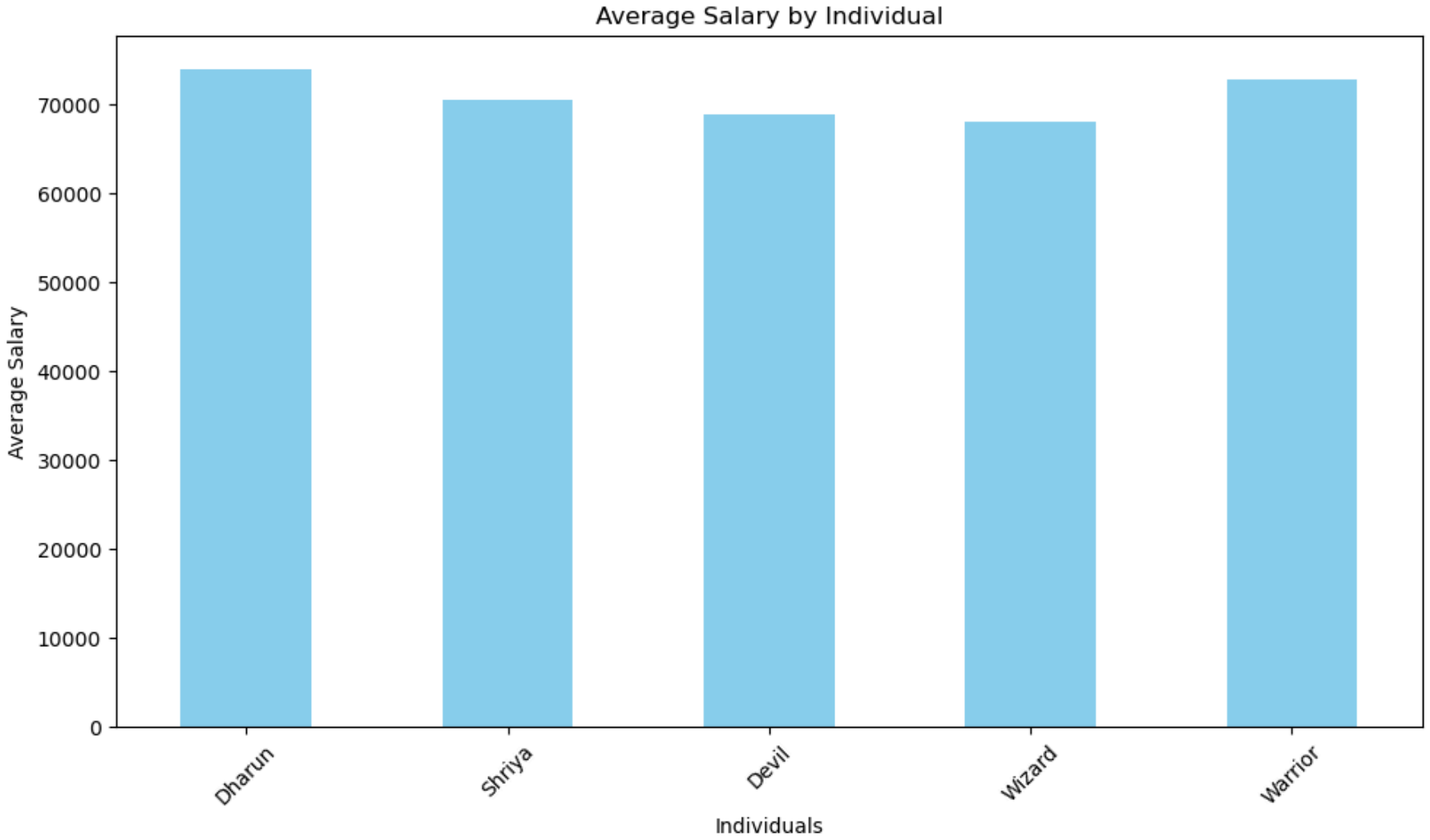

Heatmap of Salaries Over Time
Names
Dharun
Shriya
Devil
Wizard
Warrior
65795 50860 88158 94732 61284 56265 66850 87194 71962 97191
94131 66023 91090 51685 50769 52433 55311 87819 89188 67568
69769 78693 56396 77480 91434 75658 68942 68431 52747 50189
69118 85773 51899 51267 81551 61394 53556 53890 91606 80740
64502 71777 60627 58792 93323 93021 58433 93001 61016 73897
2023-01-31T00:00:00.000000000
2023-02-28T00:00:00.000000000
2023-03-31T00:00:00.000000000
2023-04-30T00:00:00.000000000
2023-05-31T00:00:00.000000000
2023-06-30T00:00:00.000000000
2023-07-31T00:00:00.000000000
2023-08-31T00:00:00.000000000
2023-09-30T00:00:00.000000000
2023-10-31T00:00:00.000000000
Date
90000
80000
70000
60000

In [5]:
```
# 4. Exporting Time Series Data to Excel
print("Exporting Time Series DataFrame to Excel")
df_timeseries.to_excel('time_series_output.xlsx', index=True)
print("Time series data exported successfully!")
```

```
Exporting Time Series DataFrame to Excel
Time series data exported successfully!
```

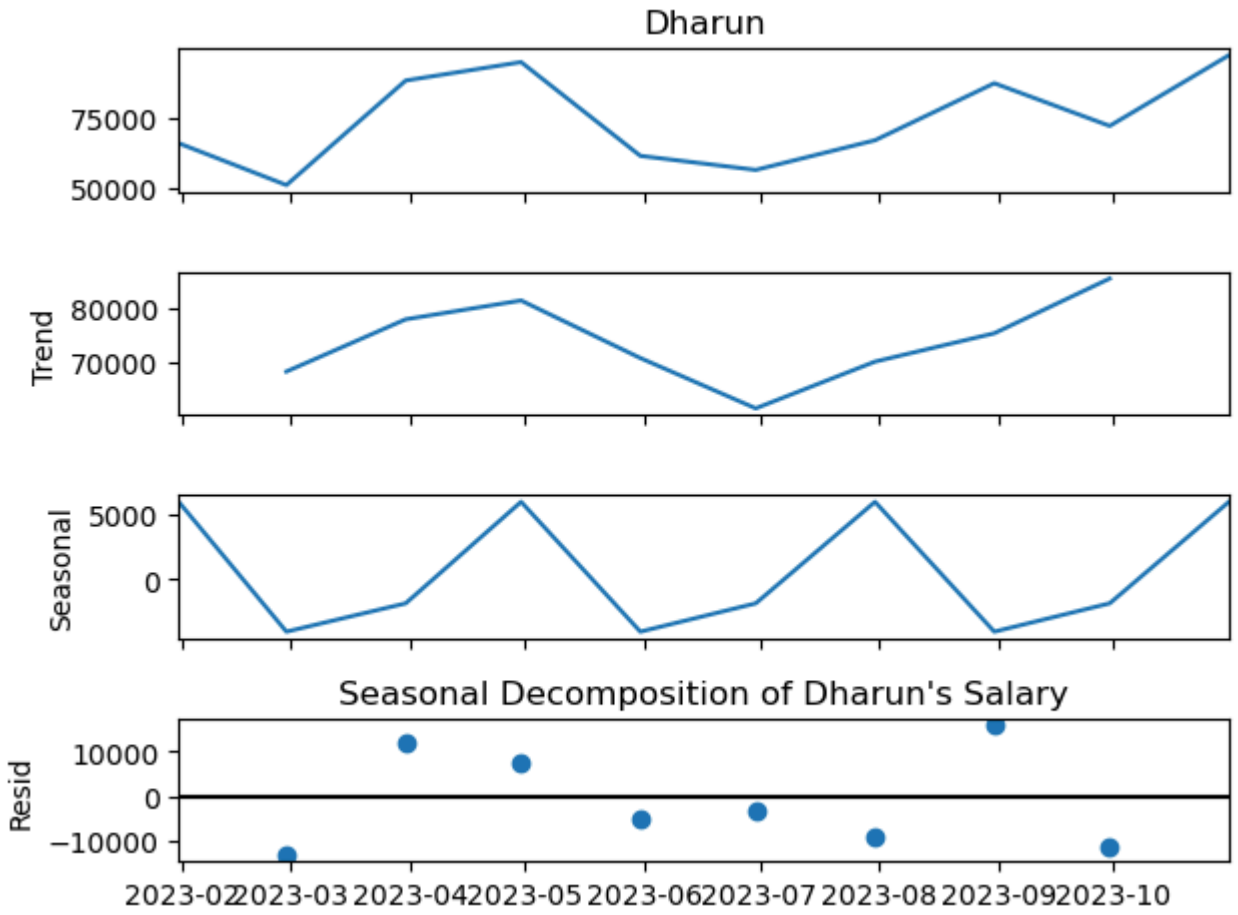
In [6]:
```
# Bar Plot: Average Salary by Individual
print("4. Bar Plot: Average Salary by Individual")
average_salaries = df_timeseries.mean()
plt.figure(figsize=(10, 6))
average_salaries.plot(kind='bar', color='skyblue')
plt.title('Average Salary by Individual')
plt.xlabel('Individuals')
plt.ylabel('Average Salary')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Expected Output for Bar Plot
# A bar plot will be displayed showing the average salary for each individual.
```

```
4. Bar Plot: Average Salary by Individual
```


Average Salary by Individual
Average Salary
70000
60000
50000
40000
30000
20000
10000
0
Dharun
Shriya
Devil
Wizard
Warrior
Individuals

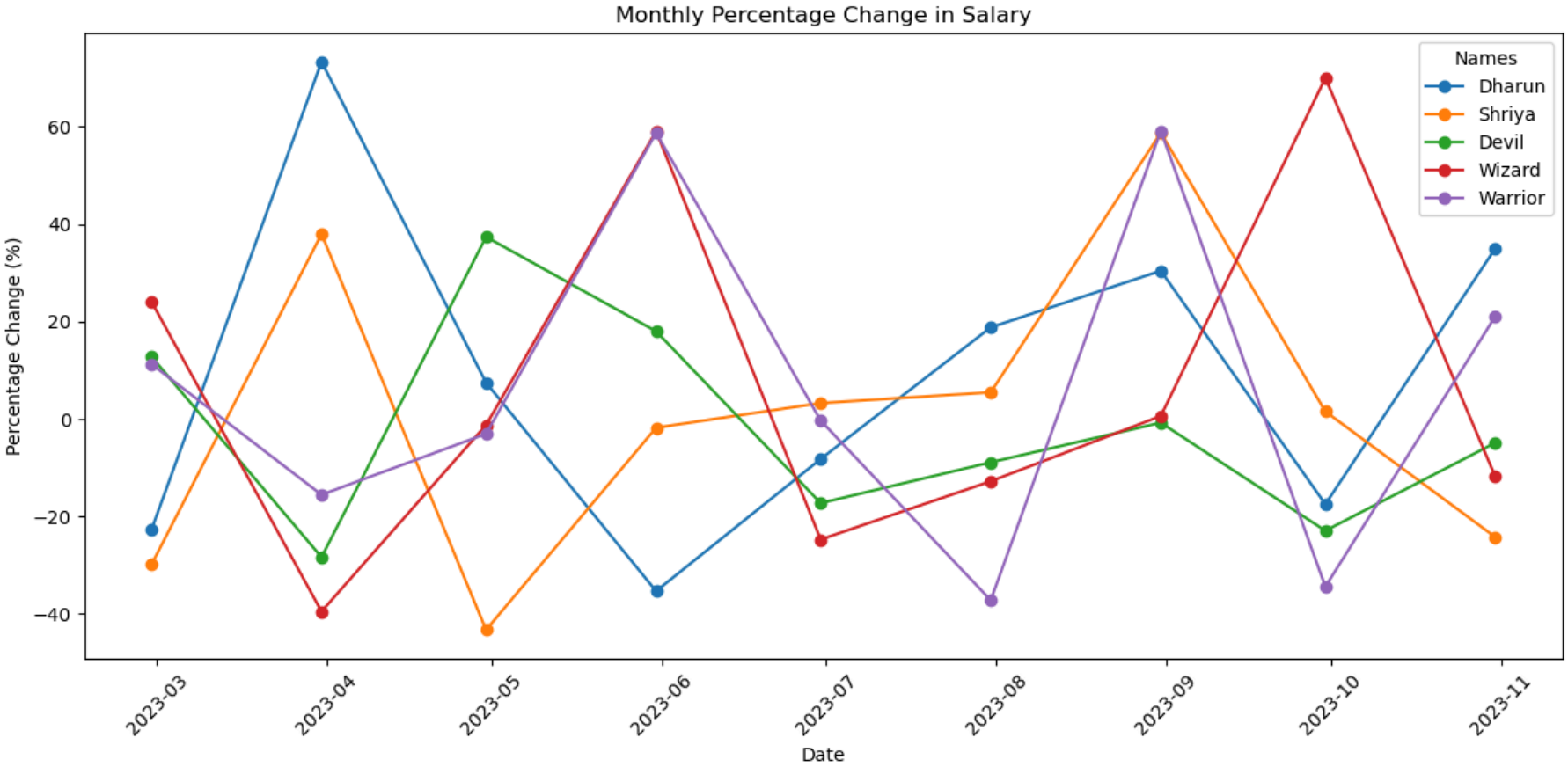
In [7]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

# Seasonal Decomposition
print("5. Seasonal Decomposition of Time Series")
result = seasonal_decompose(df_timeseries['Dharun'], model='additive', period=3)  # using Dharn's data
result.plot()
plt.title('Seasonal Decomposition of Dharun\'s Salary')
plt.tight_layout()
plt.show()
```

```
5. Seasonal Decomposition of Time Series

C:\Users\Mugunthan J\AppData\Local\Temp\ipykernel_9744\2752961917.py:8: UserWarning: The figure layout has changed t
o tight
  plt.tight_layout()
```


Dharun
75000
50000
Trend
80000
70000
Seasonal
5000
0
Seasonal Decomposition of Dharun's Salary
Resid
10000
0
−10000
2023-02
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
2023-09
2023-10

In [8]:
```
# Calculate Monthly Percentage Change
print("7. Monthly Percentage Change in Salary")
percentage_change = df_timeseries.pct_change() * 100
print("Monthly Percentage Change DataFrame:")
print(percentage_change)

# Plot Monthly Percentage Change
plt.figure(figsize=(12, 6))
for name in names:
    plt.plot(percentage_change.index, percentage_change[name], marker='o', label=name)

plt.title('Monthly Percentage Change in Salary')
plt.xlabel('Date')
plt.ylabel('Percentage Change (%)')
plt.xticks(rotation=45)
plt.legend(title='Names')
plt.tight_layout()
plt.show()
```

```
7. Monthly Percentage Change in Salary
Monthly Percentage Change DataFrame:
               Dharun     Shriya      Devil     Wizard    Warrior
Date
2023-01-31        NaN        NaN        NaN        NaN        NaN
2023-02-28 -22.699293 -29.860514  12.790781  24.096473  11.278720
2023-03-31  73.334644  37.967072 -28.334159 -39.492614 -15.534224
2023-04-30   7.457066 -43.259414  37.385630  -1.217750  -3.026704
2023-05-31 -35.308027  -1.772274  18.009809  59.071137  58.734182
2023-06-30  -8.189740   3.277591 -17.253976 -24.717048  -0.323607
2023-07-31  18.812761   5.488910  -8.876788 -12.766720 -37.183002
2023-08-31  30.432311  58.773119  -0.741203   0.623646  59.158352
2023-09-30 -17.469092   1.558888 -22.919437  69.987011 -34.392103
2023-10-31  35.058781 -24.240929  -4.849565 -11.861668  21.110856
```


Monthly Percentage Change in Salary
Percentage Change (%)
60
40
20
0
−20
−40
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
2023-09
2023-10
2023-11
Date
Names
Dharun
Shriya
Devil
Wizard
Warrior

In [9]:
```
print("Original Time Series DataFrame:")
print(df_timeseries)
print("\n")
```

```
Original Time Series DataFrame:
            Dharun  Shriya  Devil  Wizard  Warrior
Date
2023-01-31   65795   94131  69769   69118    64502
2023-02-28   50860   66023  78693   85773    71777
2023-03-31   88158   91090  56396   51899    60627
2023-04-30   94732   51685  77480   51267    58792
2023-05-31   61284   50769  91434   81551    93323
2023-06-30   56265   52433  75658   61394    93021
2023-07-31   66850   55311  68942   53556    58433
2023-08-31   87194   87819  68431   53890    93001
2023-09-30   71962   89188  52747   91606    61016
2023-10-31   97191   67568  50189   80740    73897
```

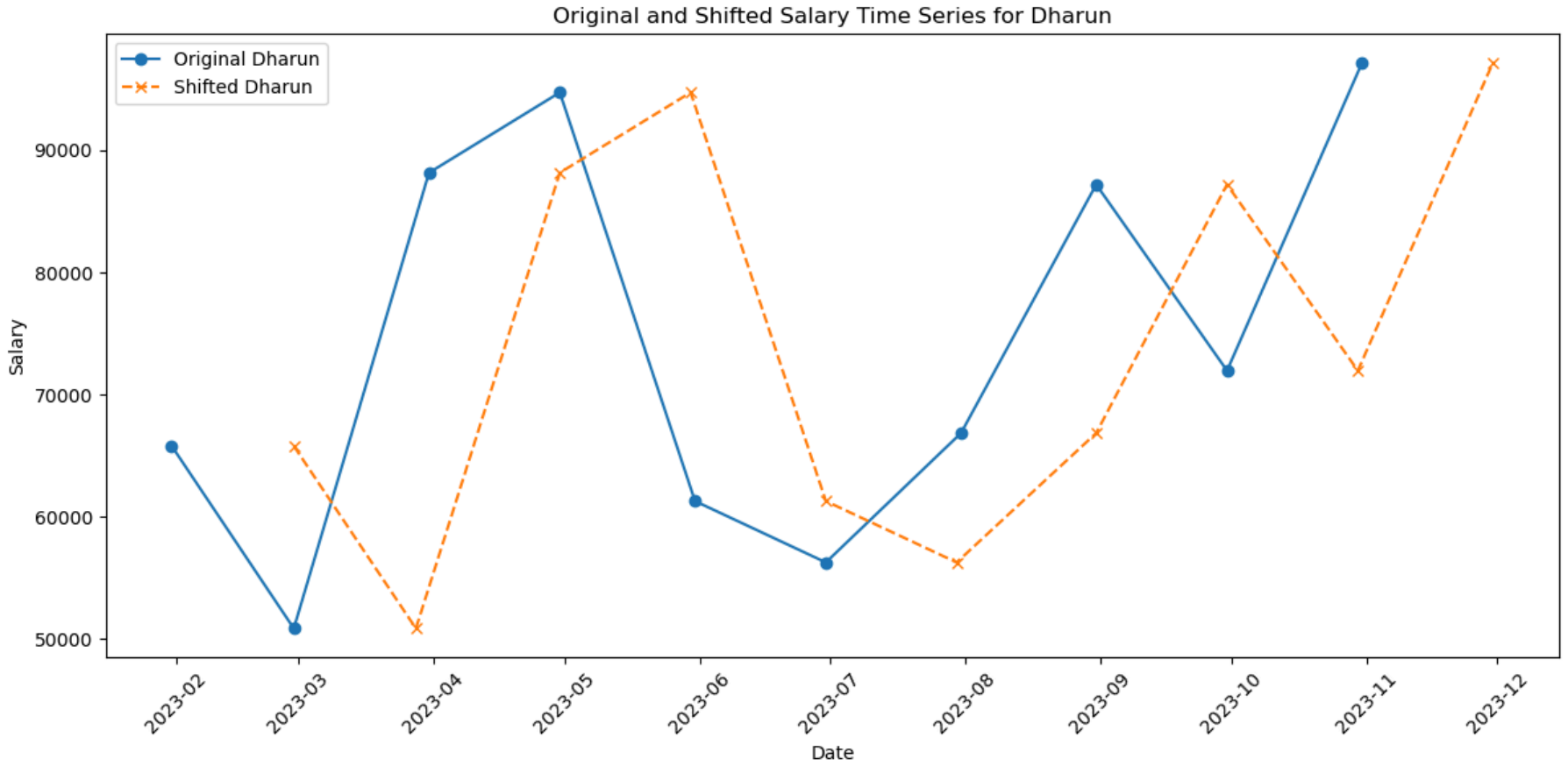
In [10]:
```
# 1. Shift Dates to the Next Month
df_shifted = df_timeseries.copy()
df_shifted.index = df_shifted.index + pd.DateOffset(months=1)
print("DataFrame After Shifting Dates by One Month:")
print(df_shifted)
print("\n")
```

```
DataFrame After Shifting Dates by One Month:
            Dharun  Shriya  Devil  Wizard  Warrior
Date
2023-02-28   65795   94131  69769   69118    64502
2023-03-28   50860   66023  78693   85773    71777
2023-04-30   88158   91090  56396   51899    60627
2023-05-30   94732   51685  77480   51267    58792
2023-06-30   61284   50769  91434   81551    93323
2023-07-30   56265   52433  75658   61394    93021
2023-08-31   66850   55311  68942   53556    58433
2023-09-30   87194   87819  68431   53890    93001
2023-10-30   71962   89188  52747   91606    61016
2023-11-30   97191   67568  50189   80740    73897
```

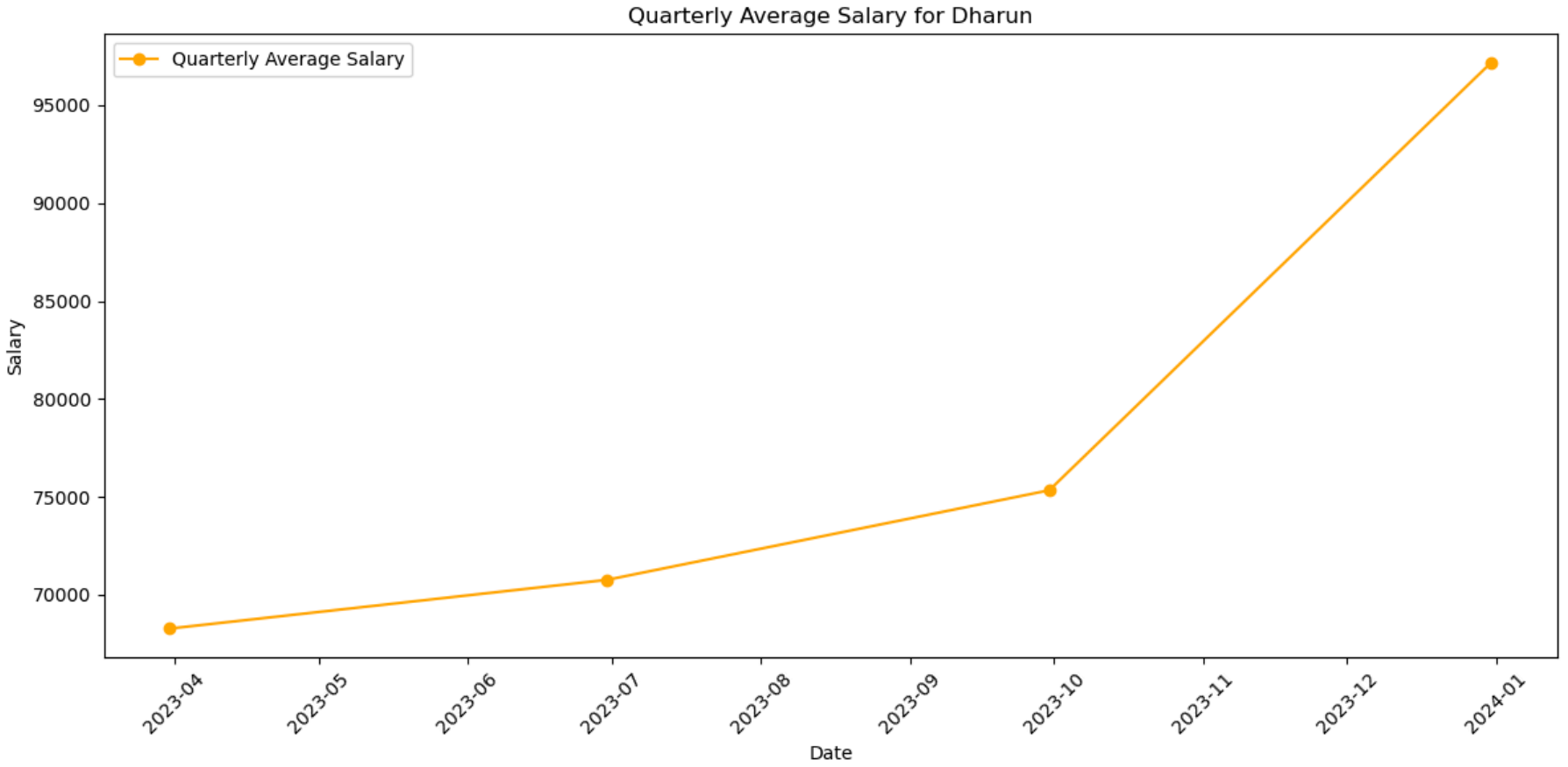
In [11]:
```
# 2. Changing Frequency to Quarterly
df_quarterly = df_timeseries.resample('Q').mean()
print("Quarterly Resampled DataFrame:")
print(df_quarterly)
print("\n")
```

```
Quarterly Resampled DataFrame:
                 Dharun        Shriya         Devil        Wizard  \
Date
2023-03-31  68271.000000  83748.000000  68286.000000  68930.000000
2023-06-30  70760.333333  51629.000000  81524.000000  64737.333333
2023-09-30  75335.333333  77439.333333  63373.333333  66350.666667
2023-12-31  97191.000000  67568.000000  50189.000000  80740.000000

                Warrior
Date
2023-03-31  65635.333333
2023-06-30  81712.000000
2023-09-30  70816.666667
2023-12-31  73897.000000
```

In [12]:
```
# 3. Visualize the Original and Shifted Time Series
plt.figure(figsize=(12, 6))
plt.plot(df_timeseries.index, df_timeseries['Dharun'], marker='o', label='Original Dharun')
plt.plot(df_shifted.index, df_shifted['Dharun'], marker='x', label='Shifted Dharun', linestyle='--')
plt.title('Original and Shifted Salary Time Series for Dharun')
plt.xlabel('Date')
plt.ylabel('Salary')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```



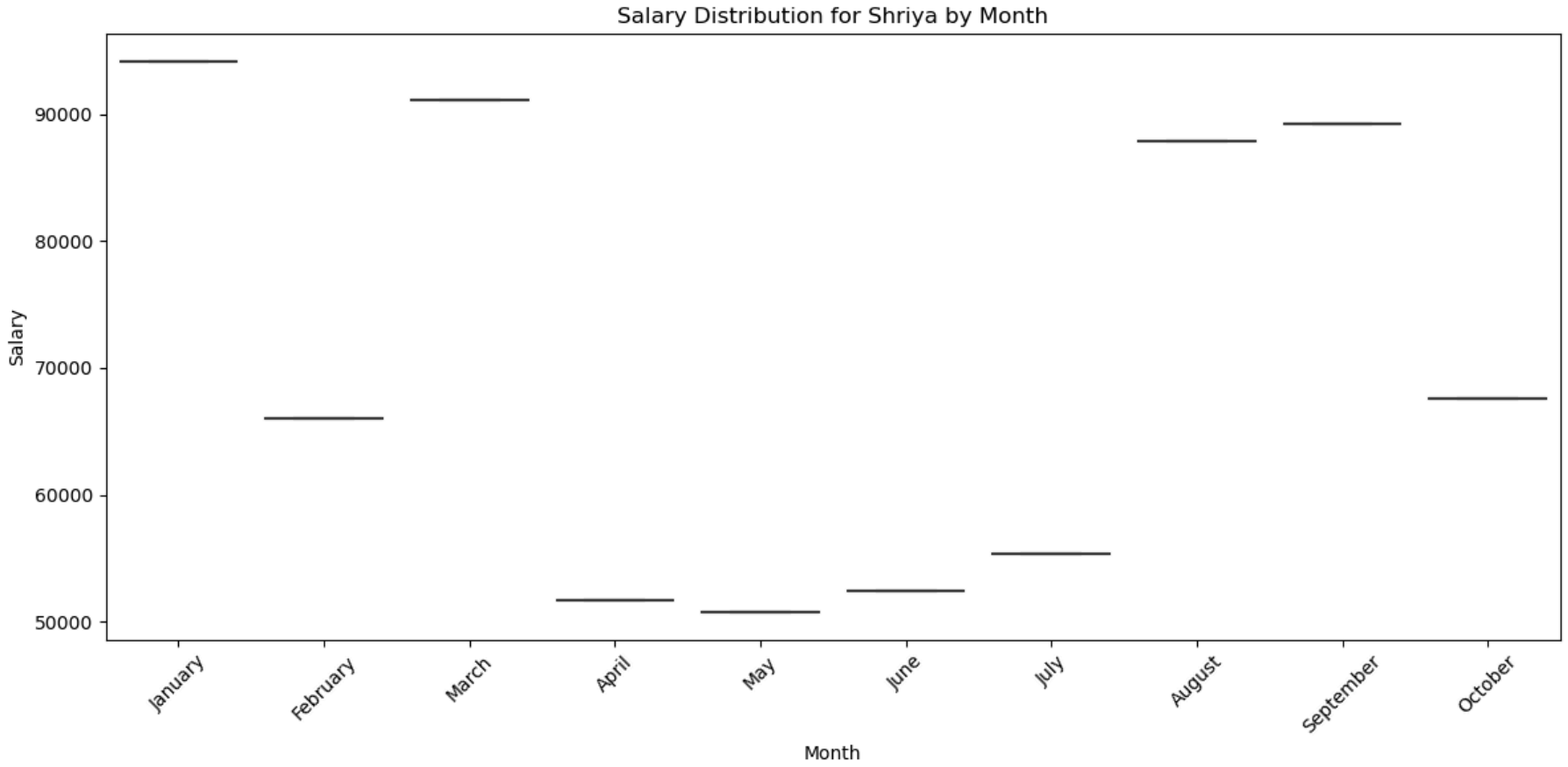
In [13]:
```
# 4. Visualize Quarterly Data
plt.figure(figsize=(12, 6))
plt.plot(df_quarterly.index, df_quarterly['Dharun'], marker='o', color='orange', label='Quarterly Average Salary')
plt.title('Quarterly Average Salary for Dharun')
plt.xlabel('Date')
plt.ylabel('Salary')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```



In [14]:
```
# 5. Adding a Column for Month Names
df_timeseries['Month'] = df_timeseries.index.month_name()
print("DataFrame with Month Names Added:")
print(df_timeseries)
print("\n")
```

```
DataFrame with Month Names Added:
            Dharun  Shriya  Devil  Wizard  Warrior      Month
Date
2023-01-31   65795   94131  69769   69118    64502    January
2023-02-28   50860   66023  78693   85773    71777   February
2023-03-31   88158   91090  56396   51899    60627      March
2023-04-30   94732   51685  77480   51267    58792      April
2023-05-31   61284   50769  91434   81551    93323        May
2023-06-30   56265   52433  75658   61394    93021       June
2023-07-31   66850   55311  68942   53556    58433       July
2023-08-31   87194   87819  68431   53890    93001     August
2023-09-30   71962   89188  52747   91606    61016  September
2023-10-31   97191   67568  50189   80740    73897    October
```

In [18]:
```
# 6. Visualizing Salary by Month
plt.figure(figsize=(12, 6))
sns.boxplot(x='Month', y='Shriya', data=df_timeseries.reset_index())
plt.title('Salary Distribution for Shriya by Month')
plt.xlabel('Month')
plt.ylabel('Salary')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [19]:
```
# 7. Save the Resampled Quarterly Data to Excel
print("Exporting Quarterly DataFrame to Excel")
df_quarterly.to_excel('quarterly_salary_output.xlsx', index=True)
print("Quarterly salary data exported successfully!")
```

```
Exporting Quarterly DataFrame to Excel
Quarterly salary data exported successfully!
```

In [ ]:
```
220901020 - DHARUN J
```