In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
# Load the data
file_path = '/Users/alanoudalturki/..'

data = pd.read_excel(file_path, sheet_name='Sheet')

# Display
data.head()
```

Out[2]:

	Key performance indicator	Jul, 2022	Aug, 2022	Sep, 2022	Oct, 2022	Nov, 2022	Dec, 2022	Jan, 2023	F 2(
0	Different % prescriptions filled per month (av...	0.920	0.9100	0.9500	0.9300	0.9400	0.9600	0.970	0.
1	Dispensing: error rate per 1000 items dispensed	0.015	0.0143	0.0145	0.0141	0.0145	0.0145	0.004	0.
2	Average patient wait time (min)	15.000	14.0000	14.0000	13.0000	14.0000	15.0000	10.000	11.
3	Pharmacist productivity (prescriptions dispens...	15.000	14.0000	11.0000	12.0000	13.0000	10.0000	22.000	15.
4	Patient interaction/counselling time (min)	2.000	3.0000	1.0000	2.0000	3.0000	1.0000	8.000	6.

In [3]:
```
# Convert all columns (excluding the first KPI column) to numeric, forcin
data_numeric = data.copy()

# Apply numeric conversion to all columns except the first (KPI column)
for col in data.columns[1:]:
    data_numeric[col] = pd.to_numeric(data[col], errors='coerce')

# Display the numeric data to confirm it has been correctly converted
data_numeric.head()
```

Out[3]:

	Key performance indicator	Jul, 2022	Aug, 2022	Sep, 2022	Oct, 2022	Nov, 2022	Dec, 2022	Jan, 2023	F 2(
0	Different % prescriptions filled per month (av...	0.920	0.9100	0.9500	0.9300	0.9400	0.9600	0.970	0.
1	Dispensing: error rate per 1000 items dispensed	0.015	0.0143	0.0145	0.0141	0.0145	0.0145	0.004	0.
2	Average patient wait time (min)	15.000	14.0000	14.0000	13.0000	14.0000	15.0000	10.000	11.
3	Pharmacist productivity (prescriptions dispens...	15.000	14.0000	11.0000	12.0000	13.0000	10.0000	22.000	15.
4	Patient interaction/counselling time (min)	2.000	3.0000	1.0000	2.0000	3.0000	1.0000	8.000	6.

In [4]:
```
data_numeric.fillna(method='ffill', inplace=True)
```

In [5]:
```
# non-numeric values
non_numeric_data = data_numeric.apply(lambda x: pd.to_numeric(x, errors='

# Show columns with non-numeric values
non_numeric_data_summary = non_numeric_data.sum()
print(non_numeric_data_summary[non_numeric_data_summary > 0])
```

```
Key performance indicator    5
dtype: int64
```

In [6]:
```
# Exclude the first column ('Key performance indicator') and calculate pc
data_numeric_no_kpi = data_numeric.iloc[:, 1:]

# Calculate percentage change across the months
data_pct_change = data_numeric_no_kpi.pct_change(axis=1) * 100

# Display
data_pct_change
```

Out[6]:

	Jul, 2022	Aug, 2022	Sep, 2022	Oct, 2022	Nov, 2022	Dec, 2022	Jan, 2023	Feb,
0	NaN	-1.086957	4.395604	-2.105263	1.075269	2.127660	1.041667	1.0
1	NaN	-4.666667	1.398601	-2.758621	2.836879	0.000000	-72.413793	-25.0
2	NaN	-6.666667	0.000000	-7.142857	7.692308	7.142857	-33.333333	10.0
3	NaN	-6.666667	-21.428571	9.090909	8.333333	-23.076923	120.000000	-31.8
4	NaN	50.000000	-66.666667	100.000000	50.000000	-66.666667	700.000000	-25.0

In [7]:
```
# Reattach the 'Key performance indicator' column to the percentage chang
data_pct_change_with_kpi = pd.concat([data['Key performance indicator'],

# Display the result with KPI
data_pct_change_with_kpi
```

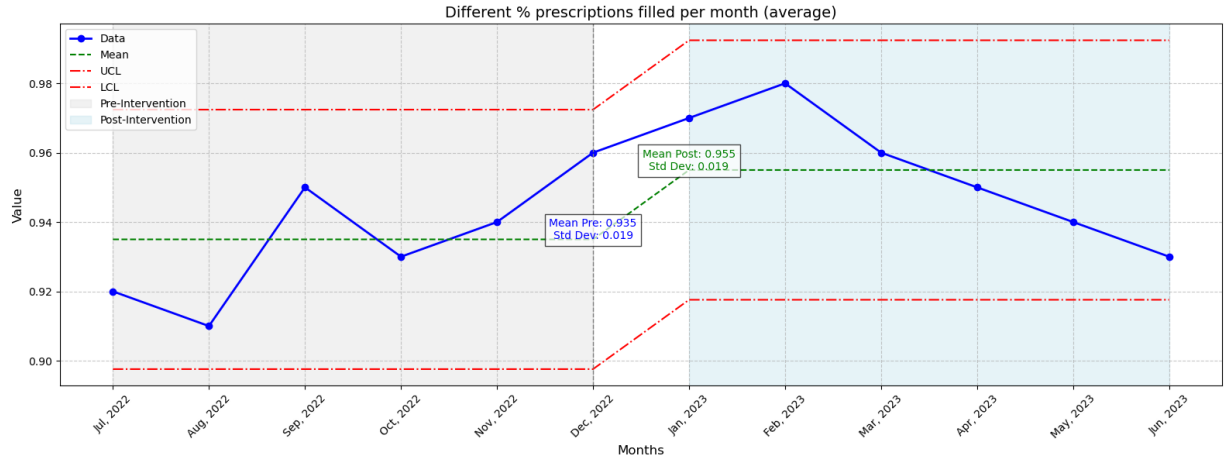
Out[7]:

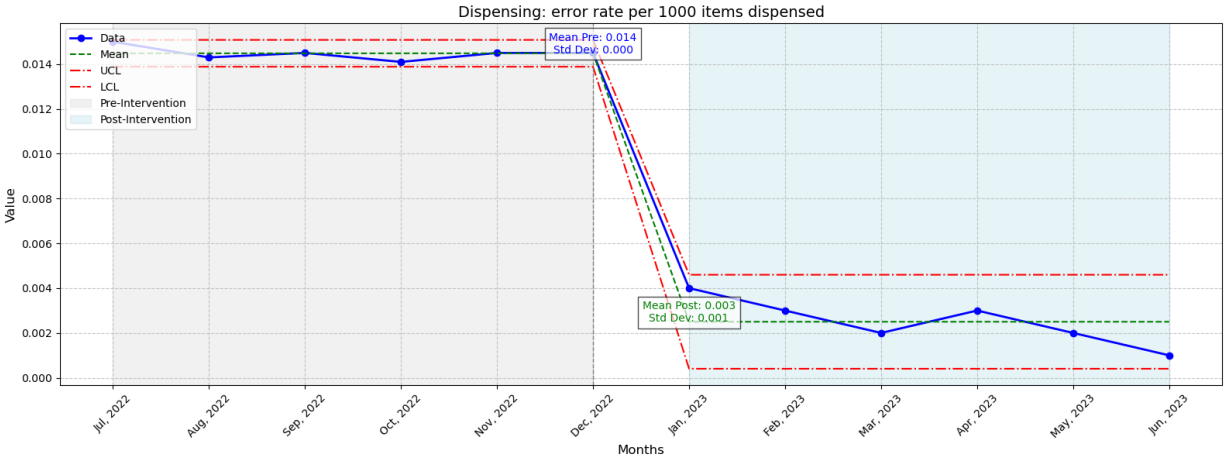
	Key performance indicator	Jul, 2022	Aug, 2022	Sep, 2022	Oct, 2022	Nov, 2022	Dec, 20
0	Different % prescriptions filled per month (av...	NaN	-1.086957	4.395604	-2.105263	1.075269	2.1276
1	Dispensing: error rate per 1000 items dispensed	NaN	-4.666667	1.398601	-2.758621	2.836879	0.0000
2	Average patient wait time (min)	NaN	-6.666667	0.000000	-7.142857	7.692308	7.1428
3	Pharmacist productivity (prescriptions dispens...	NaN	-6.666667	-21.428571	9.090909	8.333333	-23.0769
4	Patient interaction/counselling time (min)	NaN	50.000000	-66.666667	100.000000	50.000000	-66.6666

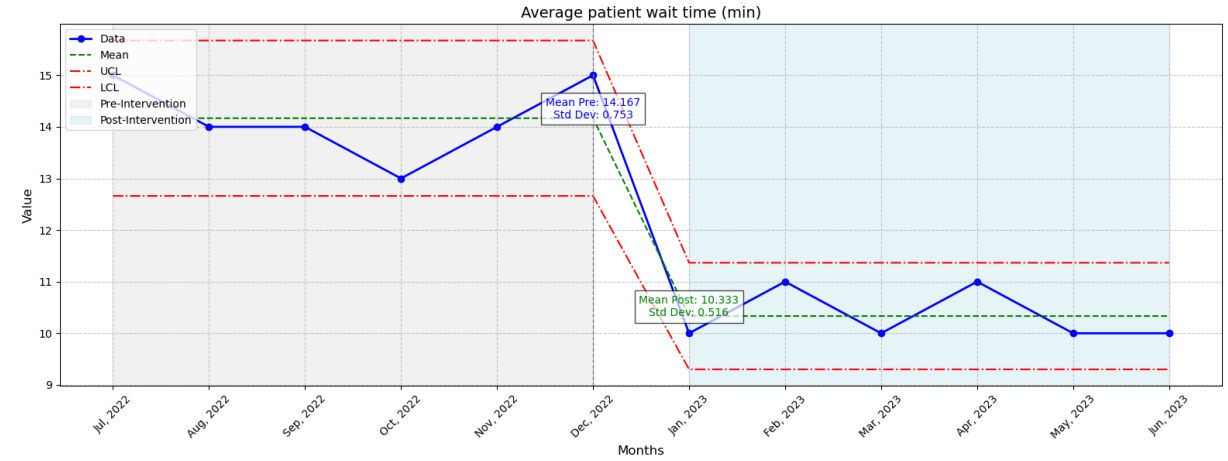
In [8]:
```
##control chart

# Load data
file_path = '/Users/alanoudalturki/Desktop/CQI_Project/Pharmacy Automatio
data = pd.read_excel(file_path, sheet_name='Sheet')

# Clean KPI names
data['Key performance indicator'] = data['Key performance indicator'].str

# Define pre- and post-robotic columns
columns_pre_robotic = ['Jul, 2022', 'Aug, 2022', 'Sep, 2022', 'Oct, 2022'
columns_post_robotic = ['Jan, 2023', 'Feb, 2023', 'Mar, 2023', 'Apr, 2023
all_columns = columns_pre_robotic + columns_post_robotic

# Function to create control charts with a single shifting line for pre a
def create_control_charts(data, kpi_list, pre_columns, post_columns, all_
    plt.figure(figsize=(16, len(kpi_list) * 6))

    for i, kpi in enumerate(kpi_list):
        # Extract KPI data
        kpi_data = data.loc[data['Key performance indicator'] == kpi]
        if kpi_data.empty:
            print(f"No data found for KPI: {kpi}")
            continue

        values_pre = kpi_data[pre_columns].values.flatten()
        values_post = kpi_data[post_columns].values.flatten()
```

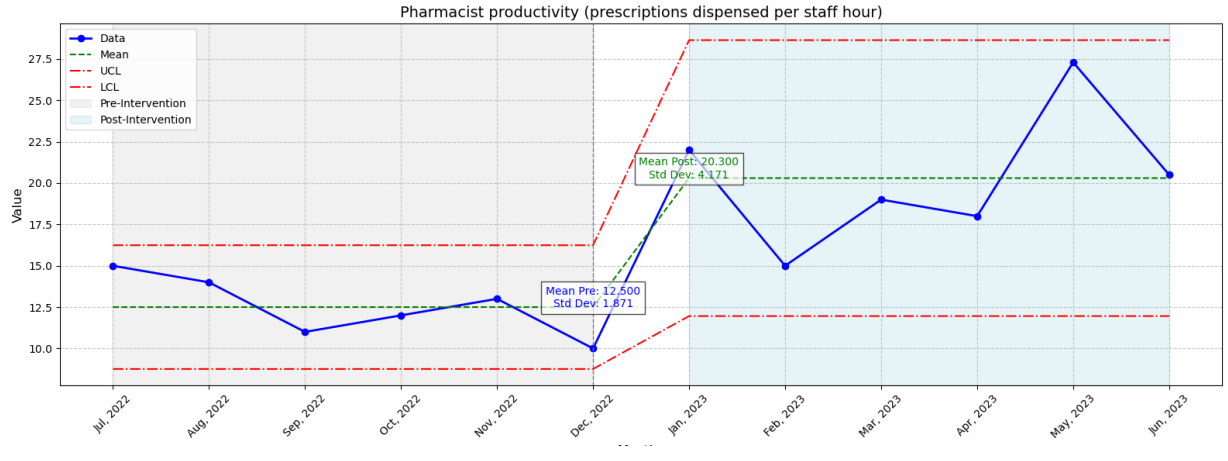
```
        # Skip KPIs with insufficient data
        if len(values_pre) == 0 or len(values_post) == 0:
            print(f"Insufficient data for KPI: {kpi}")
            continue

        # Combine pre and post values
        combined_values = np.concatenate([values_pre, values_post])

        # Calculate means and control limits
        mean_pre = np.mean(values_pre)
        std_pre = np.std(values_pre, ddof=1)
        mean_post = np.mean(values_post)
        std_post = np.std(values_post, ddof=1)

        combined_mean = [mean_pre] * len(pre_columns) + [mean_post] * len
        combined_ucl = [mean_pre + 2 * std_pre] * len(pre_columns) + [mea
        combined_lcl = [mean_pre - 2 * std_pre] * len(pre_columns) + [mea

        # Plot the control chart
        plt.subplot(len(kpi_list), 1, i + 1)
        plt.plot(all_columns, combined_values, 'o-', color='blue', label=
        plt.plot(all_columns, combined_mean, '--', color='green', label='
        plt.plot(all_columns, combined_ucl, '-.', color='red', label='UCL
        plt.plot(all_columns, combined_lcl, '-.', color='red', label='LCL

        # Annotate pre- and post-intervention means
        plt.text(pre_columns[-1], mean_pre, f"Mean Pre: {mean_pre:.3f}\nS
                 color='blue', fontsize=10, ha='center', bbox=dict(faceco
        plt.text(post_columns[0], mean_post, f"Mean Post: {mean_post:.3f}
                 color='green', fontsize=10, ha='center', bbox=dict(facec

        # Add a vertical line to indicate the transition from pre to post
        plt.axvline(x=pre_columns[-1], color='gray', linestyle='--', line

        # Highlight pre- and post-intervention periods
        plt.axvspan(0, len(pre_columns) - 1, color='lightgray', alpha=0.3
        plt.axvspan(len(pre_columns), len(all_columns) - 1, color='lightb

        # Add titles, labels, and legend
        plt.title(kpi, fontsize=14)
        plt.xlabel('Months', fontsize=12)
        plt.ylabel('Value', fontsize=12)
        plt.xticks(rotation=45)
        plt.legend(loc='upper left', fontsize=10)
        plt.grid(True, linestyle='--', alpha=0.7)

    plt.tight_layout()
    plt.show()

# List of KPIs
kpi_list = [
    'Different % prescriptions filled per month (average)',
    'Dispensing: error rate per 1000 items dispensed',
    'Average patient wait time (min)',
    'Pharmacist productivity (prescriptions dispensed per staff hour)',
    'Patient interaction/counselling time (min)'
]
```

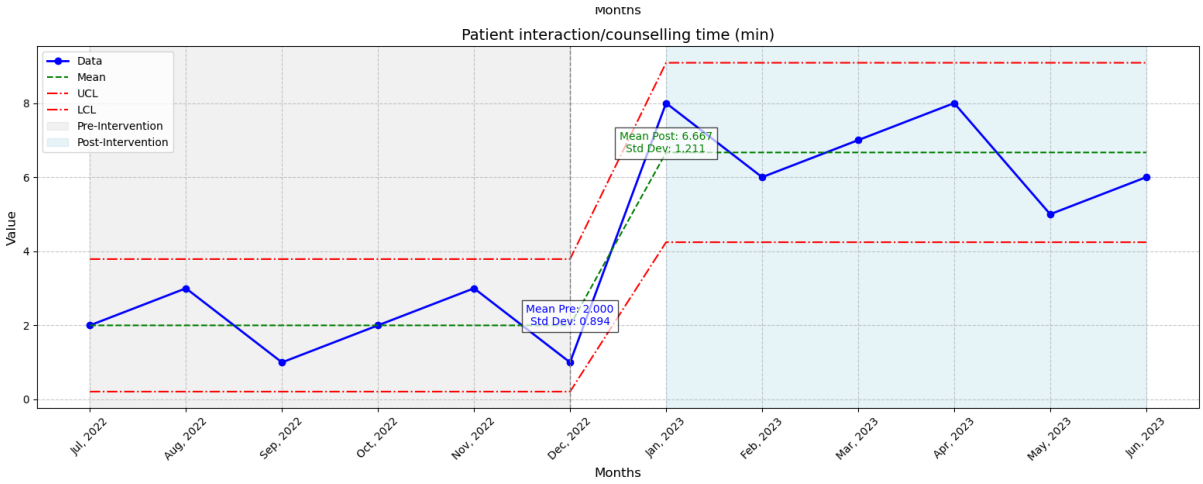
```
# Generate control charts
create_control_charts(data, kpi_list, columns_pre_robotic, columns_post_r
```










Months
Patient interaction/counselling time (min)
Data
Mean
UCL
LCL
Pre-Intervention
Post-Intervention
Mean Post: 6.667
Std Dev: 1.211
Mean Pre: 2.000
Std Dev: 0.894
Value
0
2
4
6
8
Jul, 2022
Aug, 2022
Sep, 2022
Oct, 2022
Nov, 2022
Dec, 2022
Jan, 2023
Feb, 2023
Mar, 2023
Apr, 2023
May, 2023
Jun, 2023
Months

In [9]:
```
# Extract UCL and LCL Pre and Post for each KPI

# Placeholder for results
ucl_lcl_values = []

# Function to calculate UCL and LCL for Pre and Post periods
def calculate_ucl_lcl(data, kpi_list, pre_columns, post_columns):
    for kpi in kpi_list:
        # Extract KPI data
        kpi_data = data.loc[data['Key performance indicator'] == kpi]
        if kpi_data.empty:
            continue

        # Get pre and post values
        values_pre = kpi_data[pre_columns].values.flatten()
        values_post = kpi_data[post_columns].values.flatten()

        # Calculate UCL and LCL for pre and post
        if len(values_pre) > 0:
            mean_pre = np.mean(values_pre)
            std_pre = np.std(values_pre, ddof=1)
            ucl_pre = mean_pre + 2 * std_pre
            lcl_pre = mean_pre - 2 * std_pre
        else:
            ucl_pre, lcl_pre = None, None

        if len(values_post) > 0:
            mean_post = np.mean(values_post)
            std_post = np.std(values_post, ddof=1)
            ucl_post = mean_post + 2 * std_post
            lcl_post = mean_post - 2 * std_post
        else:
            ucl_post, lcl_post = None, None

        # Append results
        ucl_lcl_values.append({
            "KPI": kpi,
            "UCL Pre": ucl_pre,
            "LCL Pre": lcl_pre,
            "UCL Post": ucl_post,
            "LCL Post": lcl_post
        })

# Call the function
calculate_ucl_lcl(data, kpi_list, columns_pre_robotic, columns_post_robot

# Convert results to a DataFrame
ucl_lcl_df = pd.DataFrame(ucl_lcl_values)

# Display the DataFrame
ucl_lcl_df
```

Out[9]:

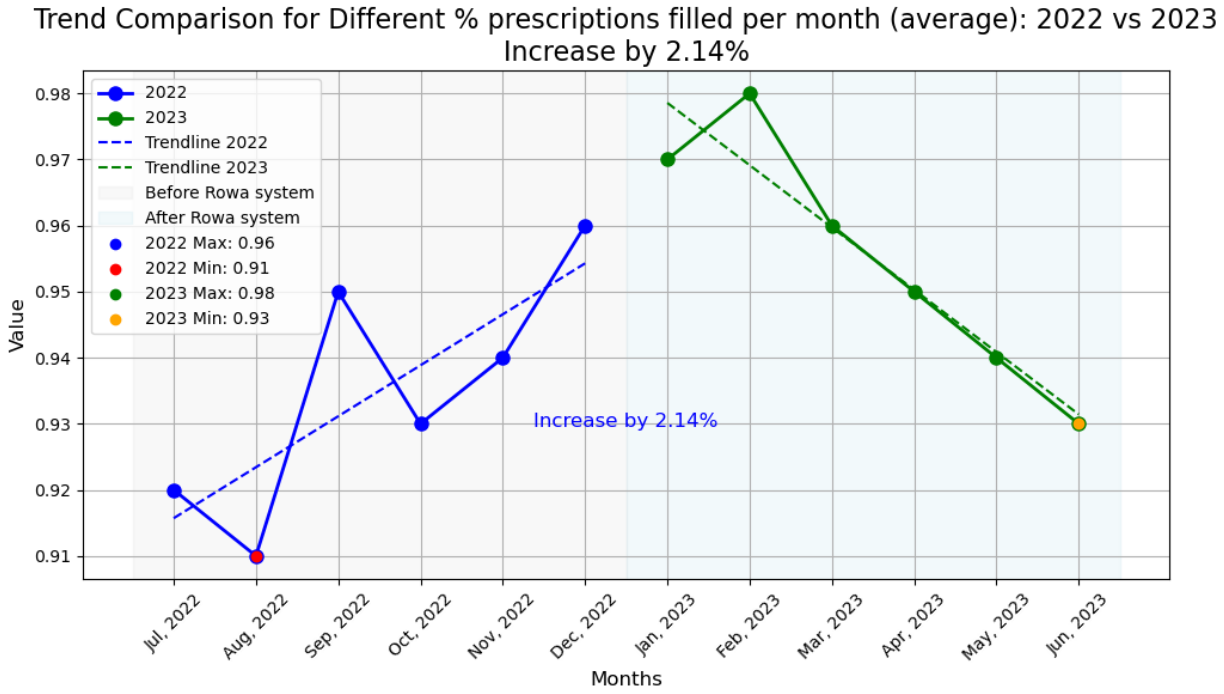
	KPI	UCL Pre	LCL Pre	UCL Post	LCL Post
0	Different % prescriptions filled per month (av...	0.972417	0.897583	0.992417	0.917583
1	Dispensing: error rate per 1000 items dispensed	0.015082	0.013884	0.004598	0.000402
2	Average patient wait time (min)	15.672212	12.661121	11.366129	9.300538
3	Pharmacist productivity (prescriptions dispens...	16.241657	8.758343	28.642661	11.957339
4	Patient interaction/counselling time (min)	3.788854	0.211146	9.088787	4.244546

In [10]:

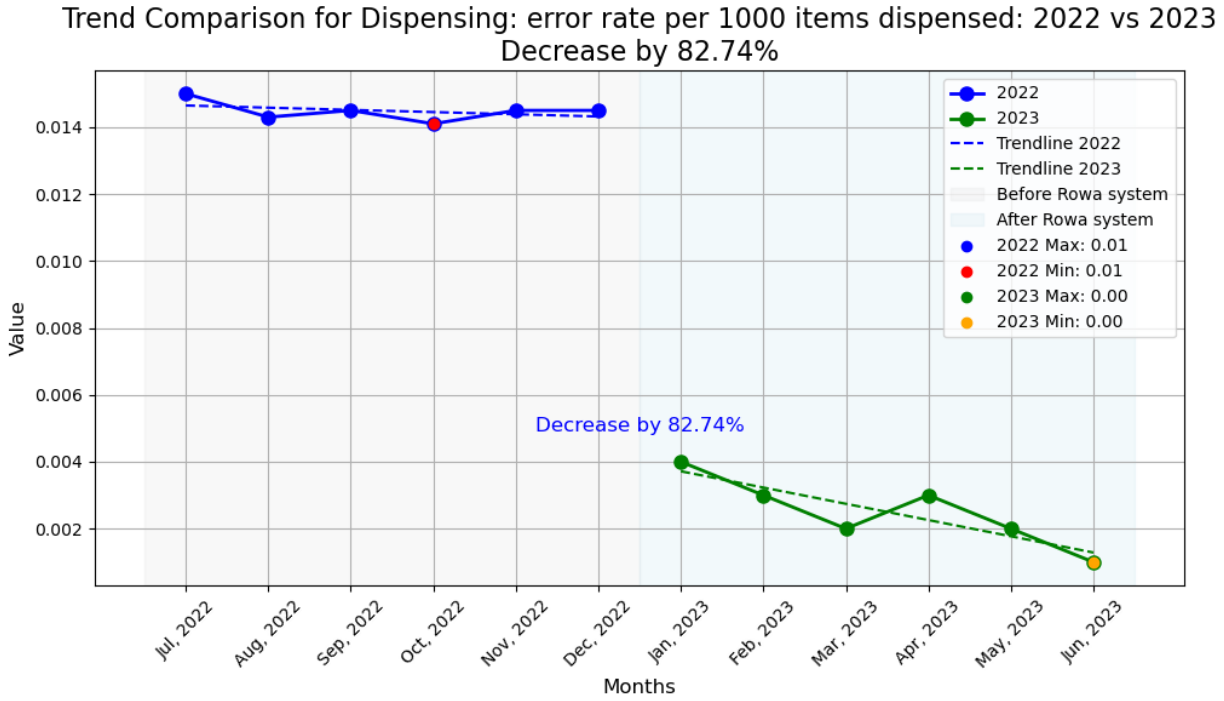
```
##Trend Comparison and Improvemnt Charts

# Define columns for 2022 and 2023
columns_2022 = ['Jul, 2022', 'Aug, 2022', 'Sep, 2022', 'Oct, 2022', 'Nov,
columns_2023 = ['Jan, 2023', 'Feb, 2023', 'Mar, 2023', 'Apr, 2023', 'May,

# Combine columns for labeling purposes
columns = columns_2022 + columns_2023

# Loop through each KPI and plot its values for both 2022 and 2023
for index, kpi in enumerate(data['Key performance indicator']):
    # Extract values for 2022 and 2023
    values_2022 = data_numeric_no_kpi.loc[index, columns_2022]
    values_2023 = data_numeric_no_kpi.loc[index, columns_2023]

    # Calculate averages for 2022 and 2023
    avg_2022 = values_2022.mean()
    avg_2023 = values_2023.mean()

    # Calculate percentage improvement
    if avg_2022 != 0 and avg_2022 != avg_2023:  # to Avoid division by ze
        percentage_improvement = ((avg_2023 - avg_2022) / avg_2022) * 100
    else:
        percentage_improvement = 0

    # Determine whether to show "Increase by X%" or "Decrease by X%"
    if percentage_improvement > 0:
        improvement_text = f'Increase by {percentage_improvement:.2f}%'
    elif percentage_improvement < 0:
        improvement_text = f'Decrease by {abs(percentage_improvement):.2f
    else:
        improvement_text = 'No Change'

    # Debugging: Print the calculated averages and percentage improvement
    print(f"KPI: {kpi}")
    print(f"Average 2022: {avg_2022:.2f}, Average 2023: {avg_2023:.2f}, {

    # Create the plot
    plt.figure(figsize=(10, 6))

    # Plot the values for 2022
    plt.plot(columns_2022, values_2022, marker='o', linestyle='-', color=
```

```
    # Plot the values for 2023
    plt.plot(columns_2023, values_2023, marker='o', linestyle='-', color=

    # Add trendline for 2022
    z_2022 = np.polyfit(range(len(values_2022)), values_2022, 1)
    p_2022 = np.poly1d(z_2022)
    plt.plot(columns_2022, p_2022(range(len(values_2022))), "b--", label=

    # Add trendline for 2023
    z_2023 = np.polyfit(range(len(values_2023)), values_2023, 1)
    p_2023 = np.poly1d(z_2023)
    plt.plot(columns_2023, p_2023(range(len(values_2023))), "g--", label=

    # Add lighter shaded background to indicate before and after "Rowa sy
    plt.axvspan(-0.5, 5.5, color='lightgray', alpha=0.15, label='Before R
    plt.axvspan(5.5, 11.5, color='lightblue', alpha=0.15, label='After Ro

    # Highlight max and min values for both years
    plt.scatter([columns_2022[np.argmax(values_2022)]], [max(values_2022)
    plt.scatter([columns_2022[np.argmin(values_2022)]], [min(values_2022)
    plt.scatter([columns_2023[np.argmax(values_2023)]], [max(values_2023)
    plt.scatter([columns_2023[np.argmin(values_2023)]], [min(values_2023)

    # Add titles and labels
    plt.title(f'Trend Comparison for {kpi}: 2022 vs 2023\n{improvement_te
    plt.xlabel('Months', fontsize=12)
    plt.ylabel('Value', fontsize=12)

    # Add gridlines and legend
    plt.grid(True)
    plt.legend()

    # Rotate x-axis labels for readability
    plt.xticks(rotation=45)

    # Adjust the position of the percentage improvement text
    plt.text(0.5, 0.3, f'{improvement_text}', transform=plt.gca().transAx

    # Show the plot
    plt.tight_layout()
    plt.show()

    # Print the percentage improvement for each KPI
    print(f'Percentage Improvement for {kpi} from 2022 to 2023: {improvem
```

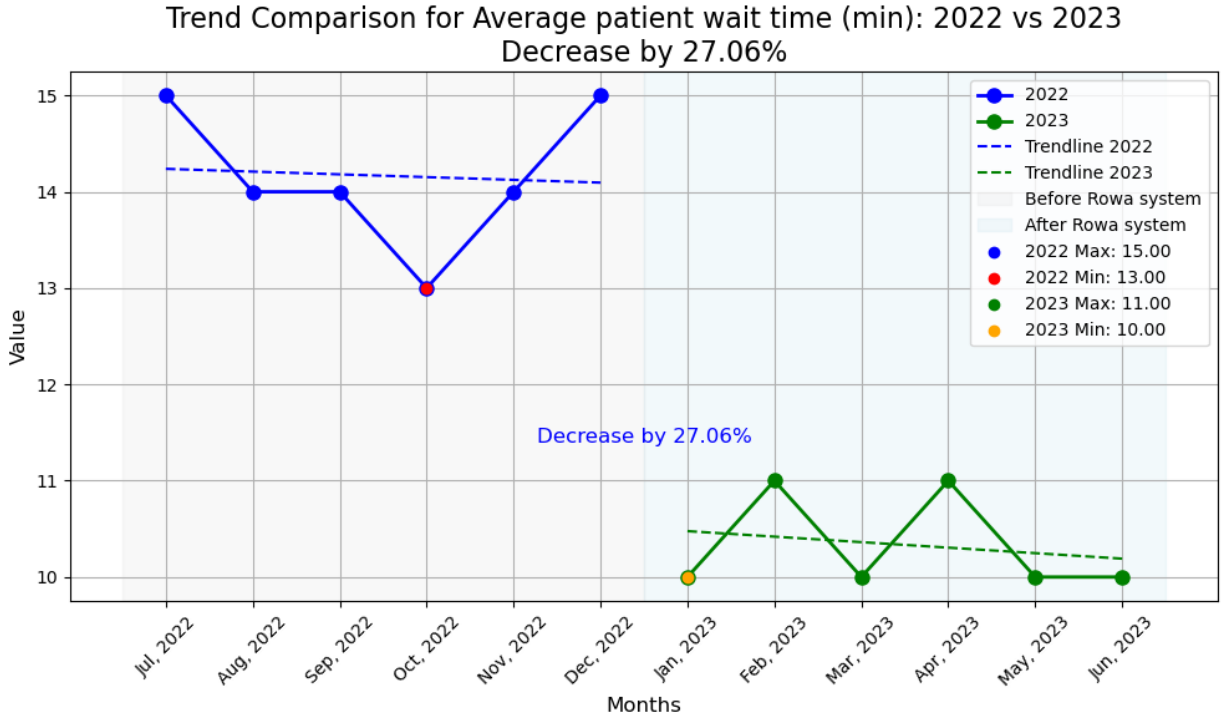
```
KPI: Different % prescriptions filled per month (average)
Average 2022: 0.94, Average 2023: 0.96, Increase by 2.14%
```



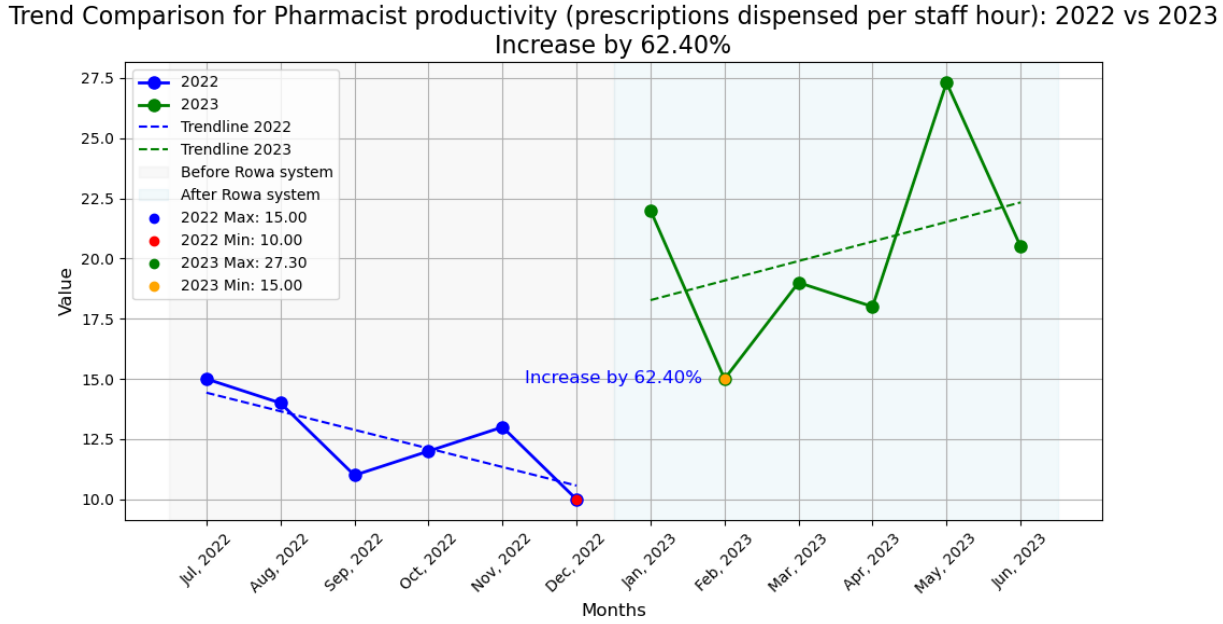
Percentage Improvement for Different % prescriptions filled per month (average) from 2022 to 2023: Increase by 2.14%
KPI: Dispensing: error rate per 1000 items dispensed
Average 2022: 0.01, Average 2023: 0.00, Decrease by 82.74%



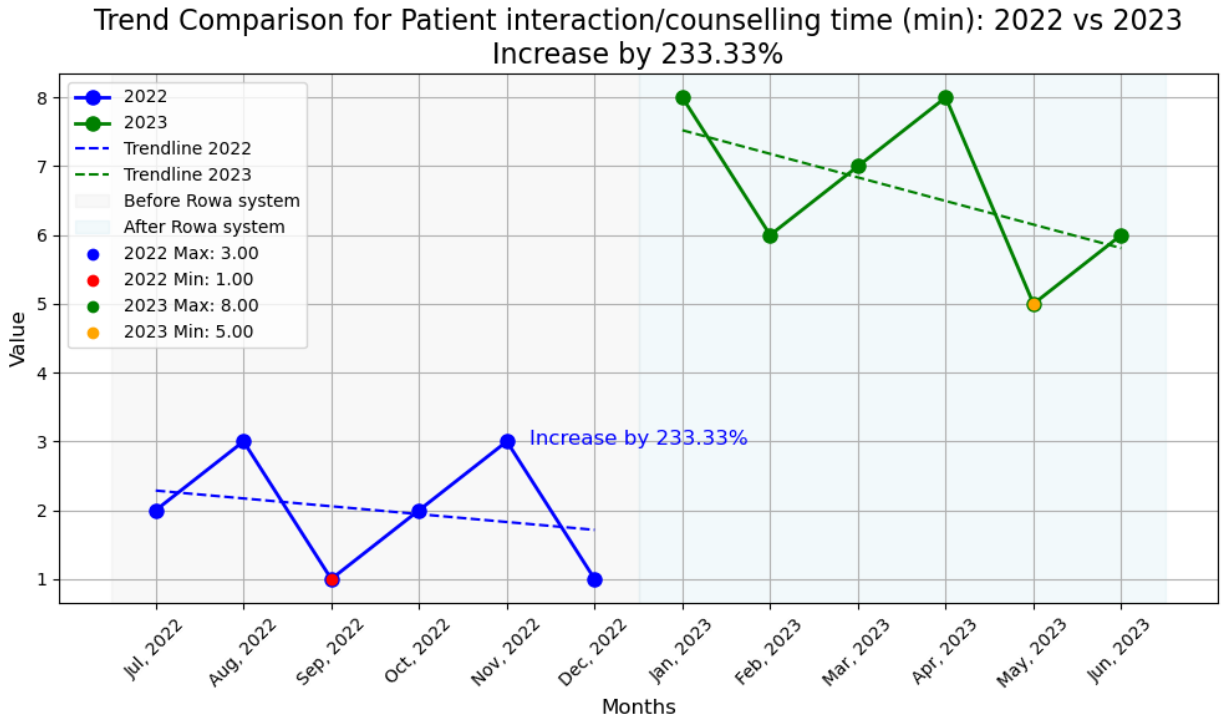
Percentage Improvement for Dispensing: error rate per 1000 items dispensed from 2022 to 2023: Decrease by 82.74%
KPI: Average patient wait time (min)
Average 2022: 14.17, Average 2023: 10.33, Decrease by 27.06%



Percentage Improvement for Average patient wait time (min) from 2022 to 2023: Decrease by 27.06%
KPI: Pharmacist productivity (prescriptions dispensed per staff hour)
Average 2022: 12.50, Average 2023: 20.30, Increase by 62.40%



Percentage Improvement for Pharmacist productivity (prescriptions dispensed per staff hour) from 2022 to 2023: Increase by 62.40%
KPI: Patient interaction/counselling time (min)
Average 2022: 2.00, Average 2023: 6.67, Increase by 233.33%




```
Percentage Improvement for Patient interaction/counselling time (min) fro
m 2022 to 2023: Increase by 233.33%
```

In [ ]: