K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: B - 1 **Roll No.: 16014022050**

Experiment No. 6

TITLE: To perform time series analysis on health care

AIM: To perform forecasting using time series analysis
Expected OUTCOME of Experiment:
CO4: Perform Time series Analytics and forecasting

Books/ Journals/ Websites referred: EXCEL SHEET, GOOGLE COLAB

Pre Lab/ Prior Concepts:

Students should have a basic understanding of: Time series Analytics and forecasting

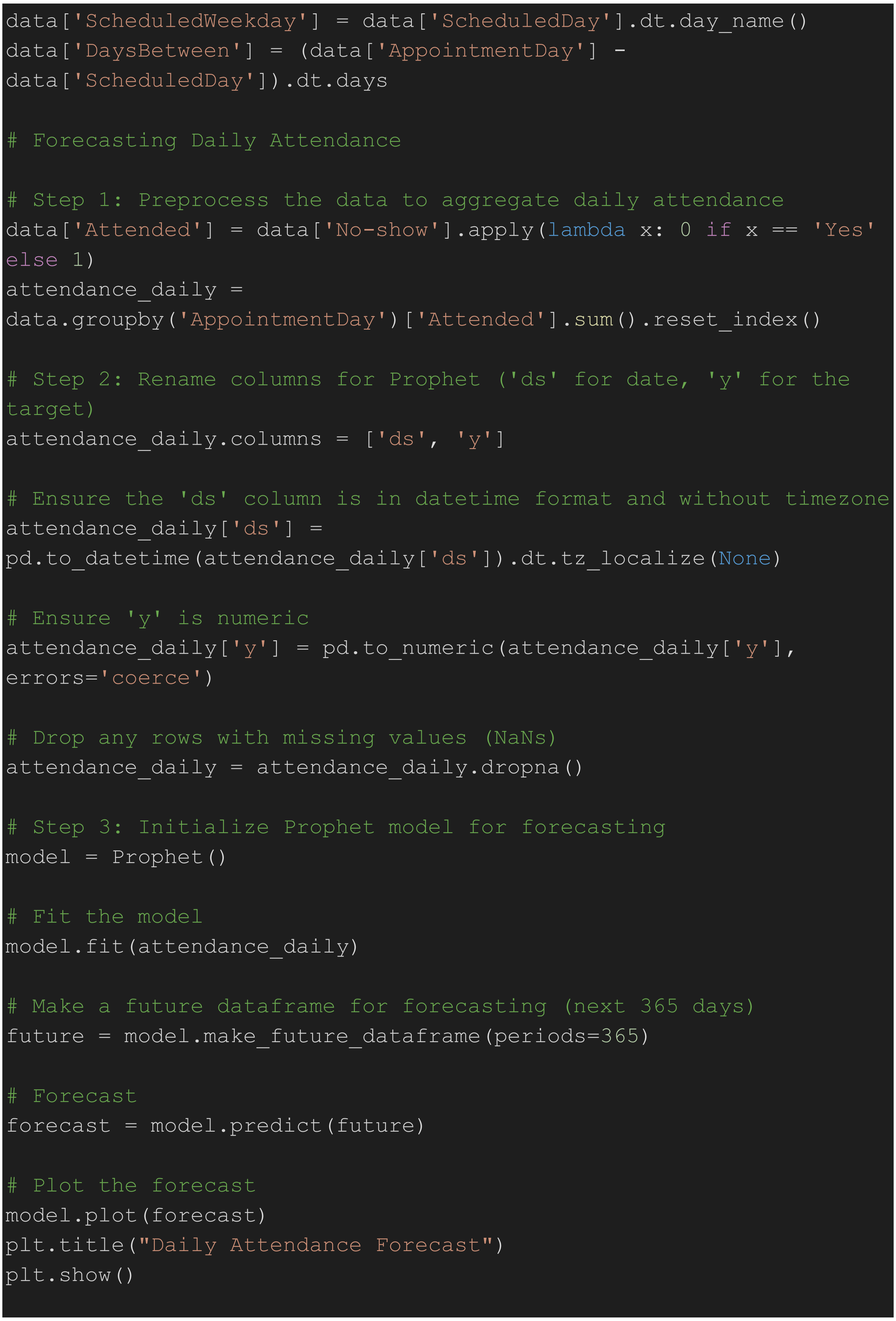
Procedure:

Data set Used: Hospital_patients_datasets

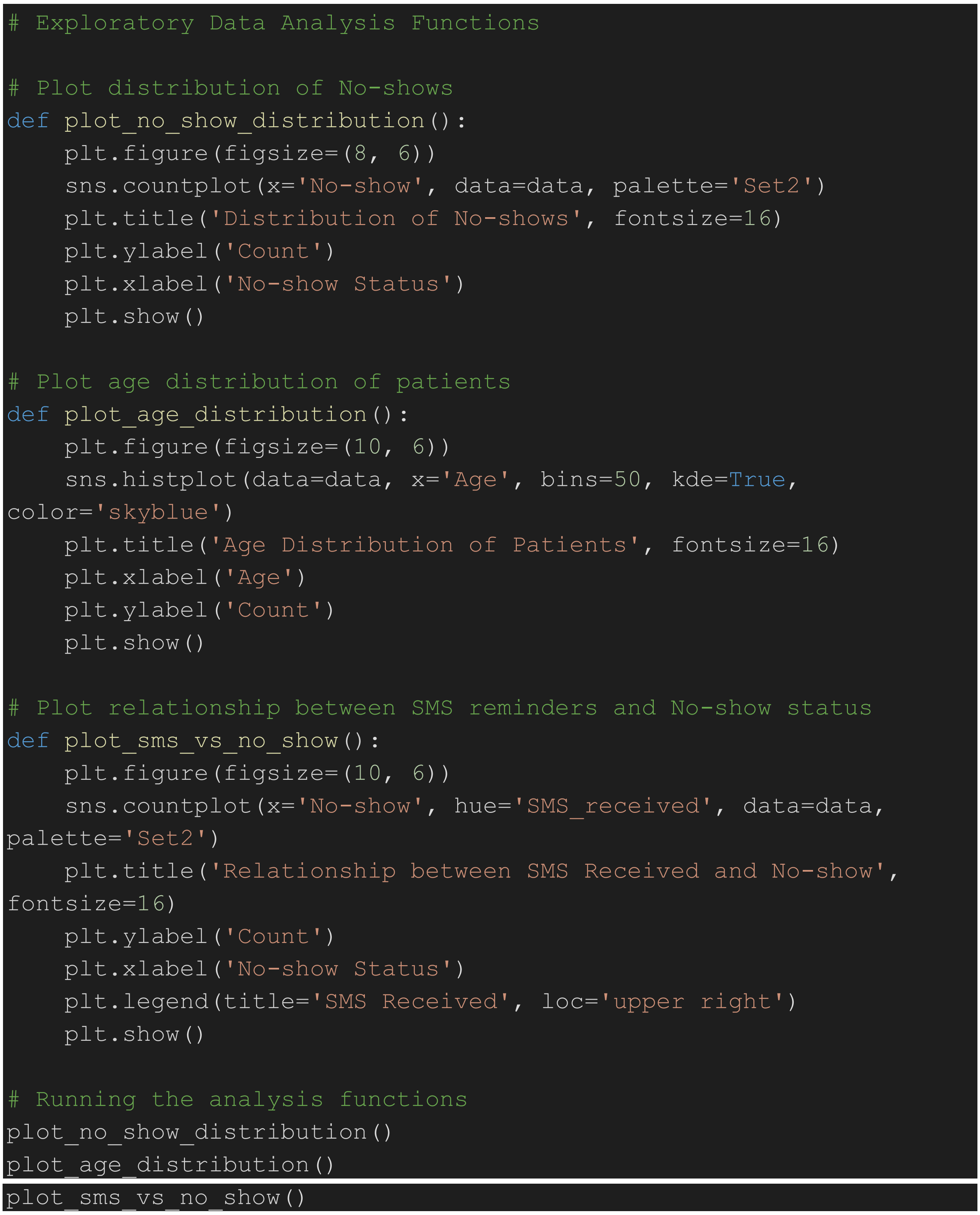
Step1: Select and Load the dataset

pip install prophet

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from prophet import Prophet

# Load the dataset
file_path = '/content/Hospital_patients_datasets (1) -
Hospital_patients_datasets (1).csv'  # Replace with actual path
data = pd.read_csv(file_path)

# Convert 'ScheduledDay' and 'AppointmentDay' to datetime format
data['ScheduledDay'] = pd.to_datetime(data['ScheduledDay'])
data['AppointmentDay'] = pd.to_datetime(data['AppointmentDay'])

# Create new features: Appointment Weekday, Scheduled Weekday, Days
Between Scheduling and Appointment
data['AppointmentWeekday'] = data['AppointmentDay'].dt.day_name()
```

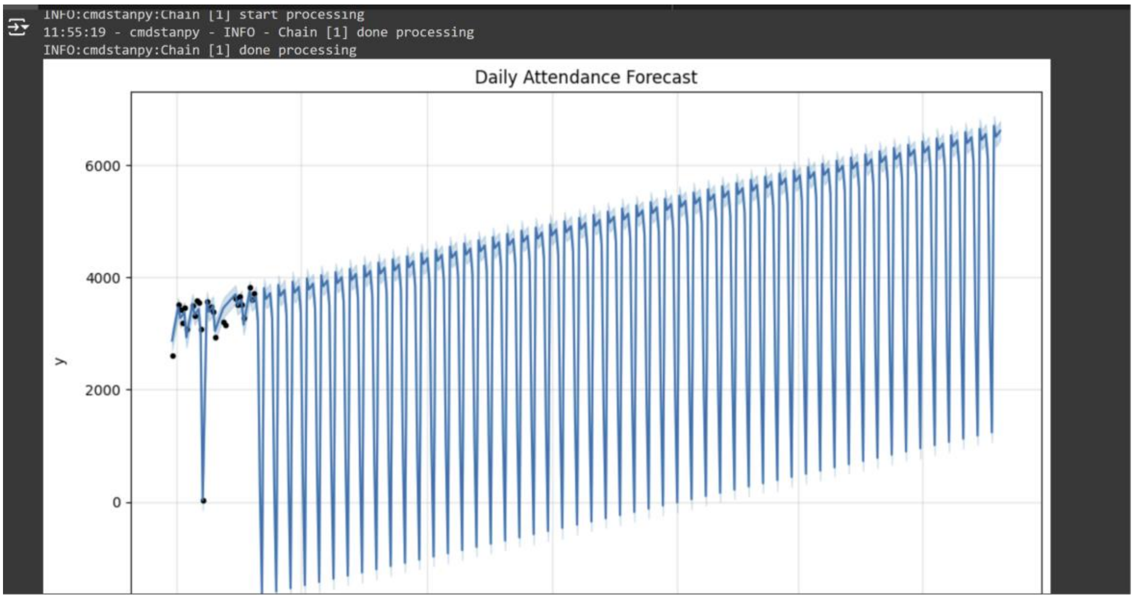


```
data['ScheduledWeekday'] = data['ScheduledDay'].dt.day_name()
data['DaysBetween'] = (data['AppointmentDay'] -
data['ScheduledDay']).dt.days

# Forecasting Daily Attendance

# Step 1: Preprocess the data to aggregate daily attendance
data['Attended'] = data['No-show'].apply(lambda x: 0 if x == 'Yes'
else 1)
attendance_daily =
data.groupby('AppointmentDay')['Attended'].sum().reset_index()

# Step 2: Rename columns for Prophet ('ds' for date, 'y' for the
target)
attendance_daily.columns = ['ds', 'y']

# Ensure the 'ds' column is in datetime format and without timezone
attendance_daily['ds'] =
pd.to_datetime(attendance_daily['ds']).dt.tz_localize(None)

# Ensure 'y' is numeric
attendance_daily['y'] = pd.to_numeric(attendance_daily['y'],
errors='coerce')

# Drop any rows with missing values (NaNs)
attendance_daily = attendance_daily.dropna()

# Step 3: Initialize Prophet model for forecasting
model = Prophet()

# Fit the model
model.fit(attendance_daily)

# Make a future dataframe for forecasting (next 365 days)
future = model.make_future_dataframe(periods=365)

# Forecast
forecast = model.predict(future)

# Plot the forecast
model.plot(forecast)
plt.title("Daily Attendance Forecast")
plt.show()
```



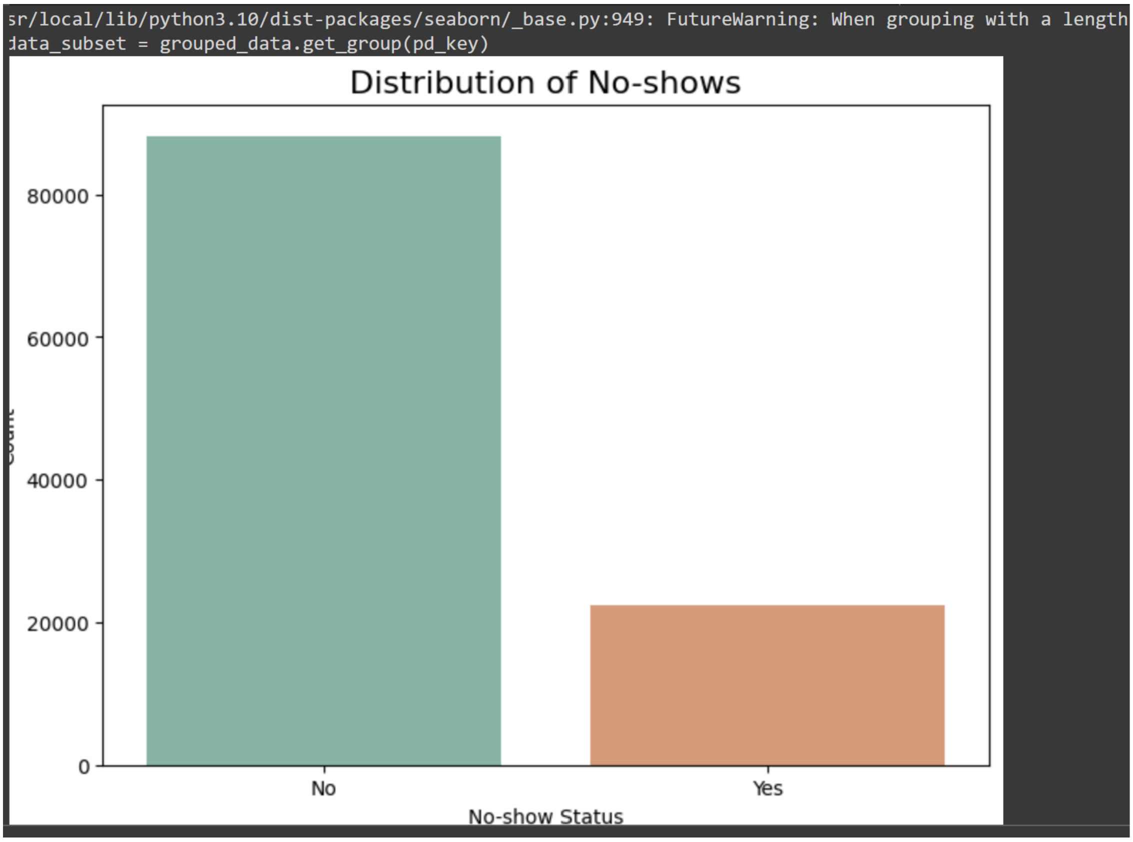
```
# Exploratory Data Analysis Functions

# Plot distribution of No-shows
def plot_no_show_distribution():
    plt.figure(figsize=(8, 6))
    sns.countplot(x='No-show', data=data, palette='Set2')
    plt.title('Distribution of No-shows', fontsize=16)
    plt.ylabel('Count')
    plt.xlabel('No-show Status')
    plt.show()

# Plot age distribution of patients
def plot_age_distribution():
    plt.figure(figsize=(10, 6))
    sns.histplot(data=data, x='Age', bins=50, kde=True,
color='skyblue')
    plt.title('Age Distribution of Patients', fontsize=16)
    plt.xlabel('Age')
    plt.ylabel('Count')
    plt.show()

# Plot relationship between SMS reminders and No-show status
def plot_sms_vs_no_show():
    plt.figure(figsize=(10, 6))
    sns.countplot(x='No-show', hue='SMS_received', data=data,
palette='Set2')
    plt.title('Relationship between SMS Received and No-show',
fontsize=16)
    plt.ylabel('Count')
    plt.xlabel('No-show Status')
    plt.legend(title='SMS Received', loc='upper right')
    plt.show()

# Running the analysis functions
plot_no_show_distribution()
plot_age_distribution()
plot_sms_vs_no_show()
```

Step2: Visualize the data

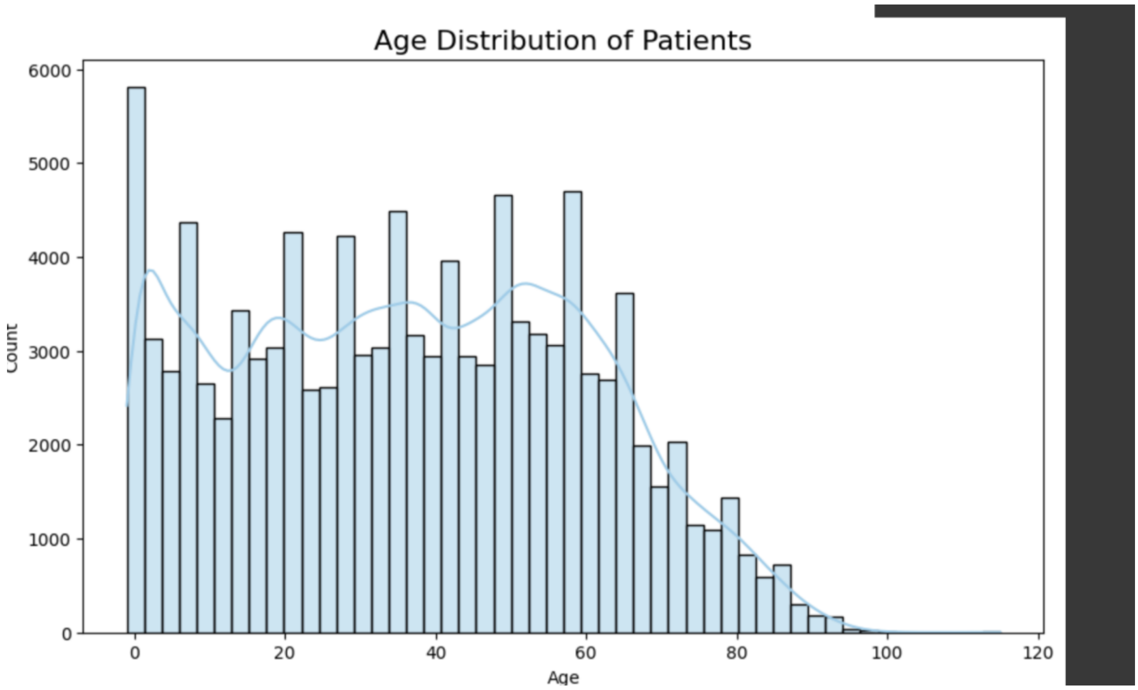


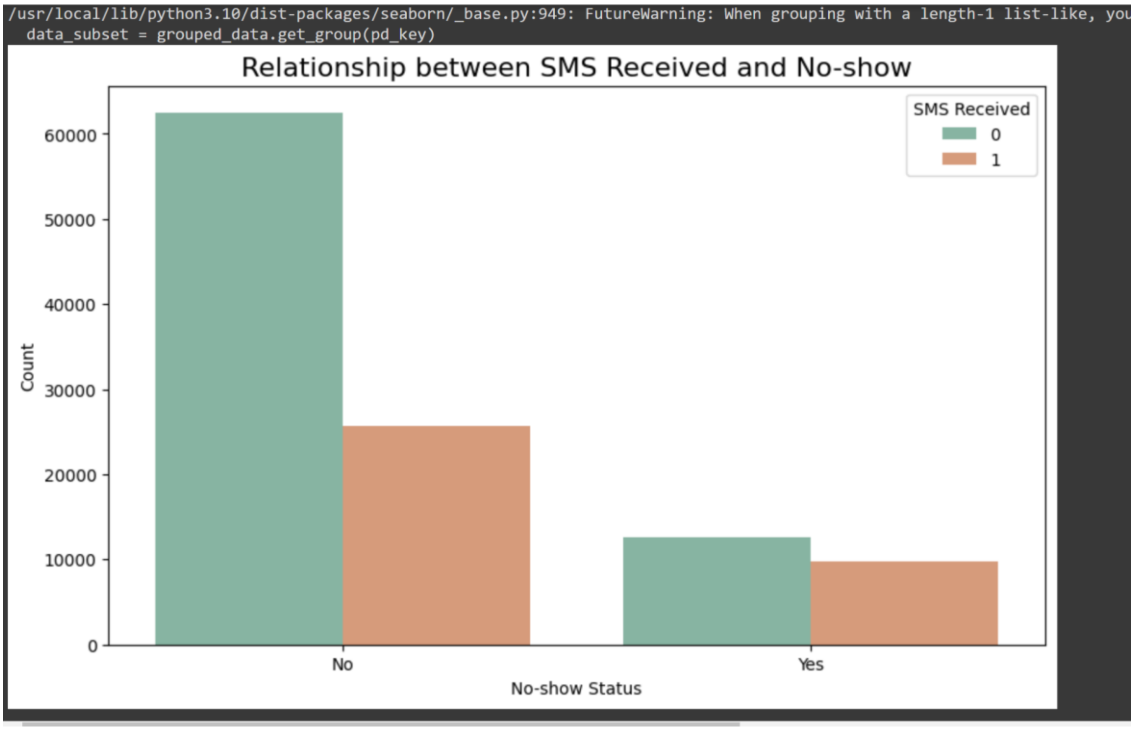
```
INFO:cmdstanpy:Chain [1] start processing
11:55:19 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

Step 3: Fit the model (ARIMA Model is Used)



```
sr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length
data_subset = grouped_data.get_group(pd_key)
```





```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you
  data_subset = grouped_data.get_group(pd_key)
```

Implementation Details:

1. Loading and Preparing the Data:

- The dataset is read from a CSV file that contains information about hospital appointments.
- Two columns, `ScheduledDay` (when the appointment was scheduled) and `AppointmentDay` (when the appointment took place), are converted to dates so that the code can work with them more easily.
- Three new pieces of information are created from the dates:
  - **AppointmentWeekday**: The day of the week when the appointment happened.
  - **ScheduledWeekday**: The day of the week when the appointment was scheduled.
  - **DaysBetween**: The number of days between scheduling the appointment and the actual appointment day.

2. Forecasting Attendance:

- Patients who didn't show up for their appointments are marked as `0` (no-show) and those who attended are marked as `1`.
- The code adds up the number of people who attended each day.
- Using the **Prophet** library, it predicts future attendance for the next year (365 days).
- Finally, it creates a graph to show this prediction of how many people will attend hospital appointments each day.

Date: 09/10/24 **Signature of faculty in-charge**

Post Lab Descriptive Questions:

1. What are the key components of a time series, and how do they affect the analysis?

   The key components of a time series are **trend, seasonality, and residuals (noise)**. The **trend** represents the long-term movement or direction of the data (upward, downward, or stable). **Seasonality** refers to patterns that repeat at regular intervals, such as daily, monthly, or yearly cycles. For instance, a retail store might experience more sales during holiday seasons. **Residuals** or **noise** capture the random fluctuations in the data that can't be explained by trends or seasonality. Understanding these components helps in developing models that can predict future behavior by isolating each factor's effect on the time series.

2. What is the purpose of decomposing a time series into trend, seasonal, and residual components?

   Decomposing a time series into **trend**, **seasonal**, and **residual** components helps analysts better understand and interpret the data. The **trend** shows the general movement over time, the **seasonality** reveals cyclical patterns, and the **residuals** represent unpredictable variations. By breaking down the time series, analysts can identify underlying patterns that affect the behavior of the data. This makes it easier to forecast future trends and address each component individually, leading to more accurate predictions and insights.

3. Explain how the ARIMA model works and what the terms (p, d, q) represent.

The **ARIMA (AutoRegressive Integrated Moving Average)** model is a popular time series forecasting method. It combines three aspects: **AutoRegression (AR)**, which uses past values to predict future ones; **Integration (I)**, which involves differencing the data to make it stationary (i.e., removing trends); and **Moving Average (MA)**, which models the relationship between an observation and a residual from a previous observation. The parameters **(p, d, q)** refer to:

- **p**: The number of lagged observations in the autoregressive model (how many past values to consider).

- **d**: The number of differencing steps needed to make the data stationary.
- **q**: The number of lagged forecast errors in the moving average model (how many past errors to consider). These parameters together define how the ARIMA model behaves to forecast future values based on past data.