Import Libraries

In [ ]:
```
import pandas as pd #for df manipulation
from datetime import datetime, timedelta, date #DateTime for date conversions and logic on who to email.

#for the logic behind the email functions
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

#Loading data from excel
df = pd.read_excel("Your\\file\\path")
```

Excel does the date conversion but I have to manually do it in Python. Im targeting users who's appointments are within the next 4 days. Using the

.today() function to get todays date and making a subset variable that adds 4 days to todays date.

from there a boolean mask that catches all the requirements I need,

1. User's appointment has to be within the next 4 days
2. They shouldn't already be confirmed(0) but this is optional it could just be a simple reminder so i'll probably remove that moving forward.
3. The contact column isn't empty
4. Their contact contains a @, some users may type a number so this filters for just emails.

In [ ]:
```
#Function to find appointments within 4 days, that have contact information and haven't confirmed
def find_upcoming_appointments_with_email(dataframe):
  dataframe['Date'] = pd.to_datetime(dataframe['Date']).dt.date #Manually convert date type to be a date
  today = date.today() #getting current date
  future_date = today + timedelta(days = 4) #Adding 4 days to current date

  #Mask to filter appointments within 4 days, that have contact information and haven't confirmed
  upcoming_appointment = df[(dataframe['Date'] < future_date) &
                            (dataframe['Confirmed'] == 0) &
                            (dataframe['Contact'].notna()) &
                            (df['Contact'].str.contains("@", na = False)) &
                            (df['Contacted']==0)&
                            (df['Cancelled/Missed']==0)
                            ]

  return(upcoming_appointment)

upcoming_appointments = find_upcoming_appointments_with_email(df)
```

In [ ]:
```
print(upcoming_appointments)
```

Function to send emails to those identified in the earlier Functions

In [ ]:
```
def send_emails_for_upcoming_appointments(upcoming_appointments):
    sender_email = "CompanyEmail" #Company email address
    sender_password = "CompanyPassword" #Email address app key
    smtp_server = "smtp.gmail.com" #Hosting site
    port = 587

    #working through newly created df upcoming_appointments to get user information
    for index, appointment in upcoming_appointments.iterrows():
        receiver_email = appointment['Contact'] #should iterate through everyone in new df contact section
        name = appointment['Name']
        appointment_date = appointment['Date'].strftime('%m/%d/%Y')
        appointment_time = appointment['Time']
        appointment_ID = appointment['Appointment_ID']
      #Prepping email
        message = MIMEMultipart("alternative")
        message["Subject"] = "Appointment Reminder" #Subject
        message["From"] = sender_email #Who will be receiving the email \ Both defined earlier in this block
        message["To"] = receiver_email #Who will be sending the email   /
  #Body of email
        text = "Hi {},\nThis is a reminder for your upcoming appointment on {} at {}.".format(name, appointment_
        html = f"""\
            <html>
              <body>
                <p>Hi {name},<br>
                This is a reminder for your upcoming appointment on <b>{appointment_date}</b> at {appointment_t
                if you need to make a change to your appointment, reach out or re-visit the terminal and we can
```

```
                get you rescheduled. Thank you, have a wonderful day.
                </p>
              </body>
            </html>
            """
        part1 = MIMEText(text, "plain")
        part2 = MIMEText(html, "html")
        message.attach(part1)
        message.attach(part2)
      #Attempting to send the email.
        try:
            with smtplib.SMTP(smtp_server, port) as server:
                server.starttls()  # Secure the connection
                server.login(sender_email, sender_password)
                server.sendmail(sender_email, receiver_email, message.as_string())
                print("Email sent to ID {}: {}".format(appointment_ID, receiver_email))
        except Exception as e:
            print("Error sending email to {}: {}".format(receiver_email, e))
```

In [ ]:
```
send_emails_for_upcoming_appointments(upcoming_appointments)
```

In [ ]:
```
index_for_upcoming = upcoming_appointments.index
print(index_for_upcoming)
df.loc[index_for_upcoming, 'Contacted'] = 1

print(df.loc[5])
```

Earlier function was for upcoming appointments, this is for past appointments

In [ ]:
```
def get_past_appointments(dataframe):

    #repeating the conversion process
    dataframe['Date'] = pd.to_datetime(dataframe['Date']).dt.date
    today = date.today()

    #instead of adding days, subtracting days to make sure appointments were in the past before sending
    yesterday = today - timedelta(days=1)

    past_appointments = dataframe[(dataframe['Date'] >= yesterday) & (dataframe['Date'] < today) & (dataframe['
    past_appointments = past_appointments[past_appointments['Contact'].str.contains("@", na=False)]

    return past_appointments

past_appointments = get_past_appointments(df)
print(len(past_appointments))
```

In [ ]:
```
def send_email_for_previous_appointments_feedback(past_appointments):
    sender_email = "CompanyEmail@example.com"
    sender_password = "Company-Password"  # Use the correct password or app-specific password
    smtp_server = "smtp.Company.com"
    port = 587

    for index, appointment in past_appointments.iterrows():
        receiver_email = appointment['Contact']
        name = appointment['Name']
        appointment_date = appointment['Date'].strftime('%m/%d/%Y')
        appointment_service = appointment['Service']
        appointment_ID = appointment['Appointment_ID']

        message = MIMEMultipart("alternative")
        message["Subject"] = "Feedback Request"
        message["From"] = sender_email
        message["To"] = receiver_email

        text = "Hi {},\n You recently had an appointment on {} for a {}.".format(name, appointment_date, appoin
        html = f"""\
            <html>
              <body>
                <p>Hi {name},<br>
                This is in regards to your previous appointment on {appointment_date}, first I want to thank yo
                appreciate you taking the time. We'd love to get your feedback if you're available.  Thank you
                choosing to be serviced by our incredible staff and we look forward to seeing you again soon!
                </p>
              </body>
            </html>
            """

        part1 = MIMEText(text, "plain")
        part2 = MIMEText(html, "html")
```

```
        message.attach(part1)
        message.attach(part2)

        try:
            with smtplib.SMTP(smtp_server, port) as server:
                server.starttls()  # Secure the connection
                server.login(sender_email, sender_password)
                server.sendmail(sender_email, receiver_email, message.as_string())
                print("Email sent to ID {}: {}".format(appointment_ID, receiver_email))
        except Exception as e:
            print("Error sending email to {}: {}".format(receiver_email, e))
```

In [ ]:
```
send_email_for_previous_appointments_feedback(past_appointments)
```