A few things are not working as expected. Here are the changes we need to make:   
  
1. (Current process) We are fetching new connected accounts every time to the DB when the script is run. \*we will load these separately in the DB not through the API call every time.

**Queries: In the current process we fetch connected accounts (sender account) from close api every time.**

(What i requested) a table of all connected accounts, this can be manually maintained, i.e Later this table will hold both close and gmail accounts with a boolean value i.e send from close YES/NO, so when the value is “NO” it sends from GMAIL and when the value is yes, it send via CLOSE email.

**Queries: I create a table where you can add all the connected accounts manually with a flag of `send\_via\_close`.**

**Addition: If you want i can write a separate script, which automatically fetches connected accounts every time you run the script & store all the newly available accounts to DB. Also, via command line args you will be able to add other connected accounts to DB, switch the flag (close to gmail or gmail to close), inactive or delete the account. (Need extra payment)**

2. This section  
  
MAX\_LIMIT\_PER\_DAY = 30 # maximum amount of email a sender can sent per day

WAITING\_TIME = 50 # wait time for next email sending in seconds

Implements a global wait time which is the wrong approach.  
  
What we need is: When a single email i.e [emailone@mail.com](mailto:emailone@mail.com) sends an email it waits x minutes before sending another email from this account.  
  
Meanwhile the script goes to the next email in the database and sends another email and the process repeats each being allocated x minutes to send the emails.

**Feedback: Understood**

**\*NEW modification**   
This value should be closer to 16 minutes **per email** account if the goal is to spread out 30 emails over a 480 minutes (28,800 seconds) sending period each day (call it 9am-5pm Eastern Time)

If this limit is reached then we need to stop the script and run it the next day.

**Queries: So each time an email sent from a sender that sender email will wait 16 minutes to send the next email till in total 30 emails per day.**

We need to create a DB field that tracks whether or not an email has been sent to the lead, so that the next day the script skips all leads that have received their first email. (First Email Sent - Boolean YES/NO

**Feedback: Understood**  
  
If the list in the DB is short and the emails need to repeat from the first email we need to implement an indexing mechanism where it goes back to the first email in the list.

**Feedback: Not clear. Does it mean, suppose you have 1000 receivers & 25 senders, then**   
**The 1st sender sent email to the 1st receiver and the 25th sender sent email to the 25th receiver by continuing the process. Then again the 1st sender sent an email to the 26th receiver.**  
What we need to watch out for is the rate limits imposed by close, i.e implement a rate limit mechanism so we don’t surpass it and cause exceptions in the script.  
<https://developer.close.com/topics/rate-limits/>

  
  
For this loop we need to have a try/catch clause **within the loop** so in case any one item in the loop causes an exception, it doesn't affect other items.  
  
For example, see line 146. If indexing into contact['emails'][0]['email'] results in an IndexException, it'll force the whole thing to break.

**Feedback: Understood, But what will the system do if it halt for rate limit issues.**

I guess this is related to the wait time for each email:  
  
 if counter >= len(contacts):

exit()

logging.info(f'--> waiting for {WAITING\_TIME}s for next email sent..')

time.sleep(WAITING\_TIME)

Instead of putting an if-clause towards the end to control flow, why not make it the condition of the loop?

**Feedback: It does the same thing, will change it as you want.**