<https://kanalservis.notion.site/kanalservis/Python-82f517c516d041b8aca227f0a44ed1f1>

**Регистрация в сервисах google и установка библиотек:**

Переходим на  [console.developers.google.com/cloud-resource-manager](https://console.developers.google.com/cloud-resource-manager)

1. Вводим учётные данные google аккаунта
2. Создаем новый проект – нажимаем на create project
3. Указываем project name и location и жмем create
4. В строке своего проекта нажимаем на три вертикальные точки и нажимаем settings
5. Слева в меню нажимаем Service Accounts и в открытом окне жмем Create Service Account. Вводим данные и нажимаем Create and continue. Далее выбираем роль – Owner и жмем Done
6. Нажимаем на email, переходим во вкладку Keys и нажимаем Add Key – Create new key. По умолчанию стоит JSON – жмем Create. Будет создан и сразу же скачан файл с ключами. Сохраните его, именно благодаря ему мы сможем получать доступ к сервисам Google.
7. Нажмите на кнопку с тремя горизонтальными штрихами, слева от надписи «Google APIs», выберите пункт «API & Services», а в нем подпункт «Enabled API and Services». Жмем Enable Apis And Services
8. В поиске пишем google drive – выбираем google drive api – жмем enable. Api устанавливается и перенаправляет нас.
9. Таким же методом установим Google таблицы - прописываем sheet и кликаем на сервисе «Google Sheets API» и устанавливаем его.
10. Переходим на вкладку Services Accounts и сохраняем email, чтобы в дальнейшем выдать себе доступ (newacc@nicetest-357207.iam.gserviceaccount.com)

**Инструкция по запуску скрипта:**

1. Переходим к установке необходимых библиотек в проекте– устанавливаем:  
     
   pip3 install --upgrade google-api-python-client oauth2client  
   pip install gspread sqlalchemy pandas
2. Открываем скрипт creategs.py. в CREDENTIALS\_FILE указываем наименование json файла, который скачался ранее.
3. Выполняем функцию createsheet(). В этой функции мы создаем таблицу с указанием необходимых параметров и даем права доступа (writer – редактирование) юзеру (указываем нужную почту).
4. Чтобы отслеживать изменения в гугл-таблице нам необходимо воспользоваться Apps Script (зайти туда можно из гугл-таблицы: «Расширения - Apps Script») и написать там функцию в редакторе, но чтобы запустить этот скрипт нужно передать права владения с сервисного аккаунта на пользовательский. Для этого отправим запрос на смену владельца (после выполнения функции update придет письмо на почту, указанную в переменной access, далее нужно открыть таблицу, перейти в настройки доступа и там принять запрос на смену владельца)
5. После выполнения функции createsheet в терминале будет ссылка, из этой ссылки копируем значение после "https://docs.google.com/spreadsheets/d/"и вставляем это в переменную spreadsheet\_id (она находится вначале этого файла).
6. Комментируем выполнение функции createsheet() и запускаем скрипт еще раз. Переходим по ссылке из терминала.
7. Открываем save\_df\_2postgres.py. Копируем значение из CREDENTIALS\_FILE в переменную credentials\_path
8. Чтобы подсоединится к базе данных postgres через sqlalchemy необходимо указать в переменной engine данные своей базы данных по следующей инструкции: ('СУБД//username://password@address/dbname')
9. Открываем в Postgres БД, переходим в Схемы – Таблицы – «название созданной таблицы». Нажимаем ПКМ – просмотр и редактирование данных – Все строки.

**Комментарий:**

На данный момент реализовал именно через цикл while и time.sleep(5) – запись через каждые 5 секунд.

Также есть другая реализация, которую я планирую сделать (она не бесконечная и скрипт будет выполняться только, когда проводились изменения): вы, наверное, увидели, что в 4 пункте я воспользовался Apps Script-ом и передал права доступа на пользовательский аккаунт (для того, чтобы писать скрипты), но дальше никак не развил идею. В этом варианте реализации необходимо перейти в Apps Script. В графе редактор написать следующую функцию:

function onEdit(event)

{

let ss = event.source.getActiveSheet();

let r = event.source.getActiveRange();

r.setComment("Last modified: " + (new Date()));

}

Она позволит отслеживать изменения в таблице.

Далее мне необходимо будет разместить все эти скрипты на хостинге, чтобы я мог взаимодействовать с ними из Apps Script.

Следующим шагом будет написать URL Fetch запрос, который будет отправлять GET-запрос на сервер. Когда на сервер приходит GET-запрос будет запускаться дополнительный скрипт на Python, со следующим сценарием: «Если приходит get-запрос, то мы запускаем остальные скрипты».

Задания на доп. баллы также попробую реализовать (не хватило времени), они будут залиты в этот репозиторий.