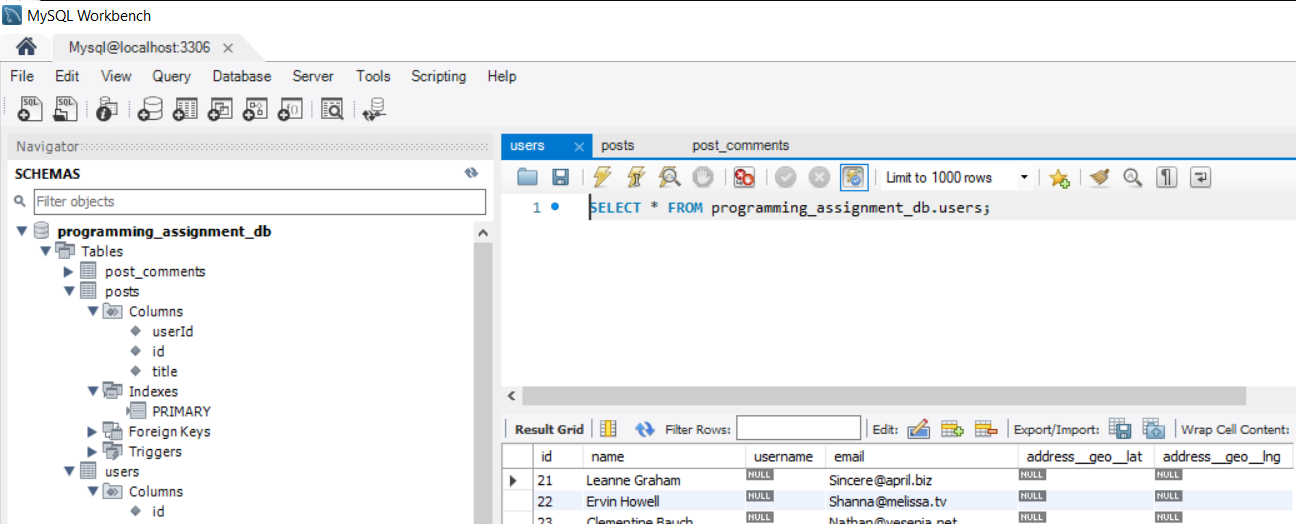
**Part 1.**

1. Create programming\_assignment\_db schema in MySQL
2. Run testDatabase.sql script provided to load data into tables



**Part 2.**

*Files: fill\_users\_table.py*

1. Call https://jsonplaceholder.typicode.com/users using Postman to view user data. User Json data (below) provides full user profile. However, we only need to extract three fields to update missing user table data:

- username

- address\_\_geo\_\_lat

- address\_\_geo\_\_lng

{

    "id": 1,

    "name": "Leanne Graham",

    "username": "Bret",

    "email": "Sincere@april.biz", Will be used to find correct user in database

    "address": {

      "street": "Kulas Light",

      "suite": "Apt. 556",

      "city": "Gwenborough",

      "zipcode": "92998-3874",

      "geo": {

        "lat": "-37.3159",

        "lng": "81.1496"

      }

    },

    "phone": "1-770-736-8031 x56442",

    "website": "hildegard.org",

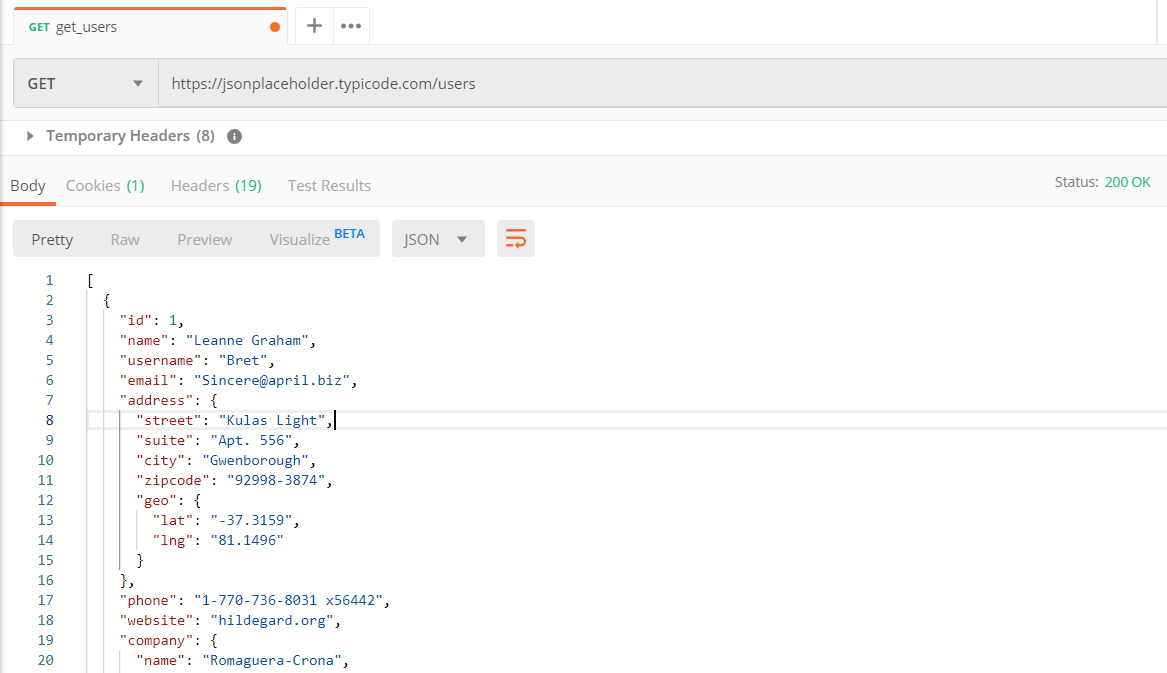
    "company": {

      "name": "Romaguera-Crona",

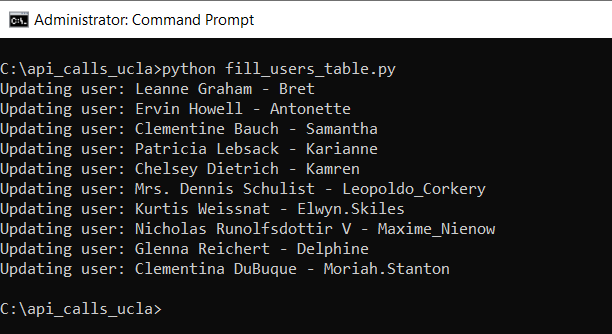
      "catchPhrase": "Multi-layered client-server neural-net",

      "bs": "harness real-time e-markets"

    }

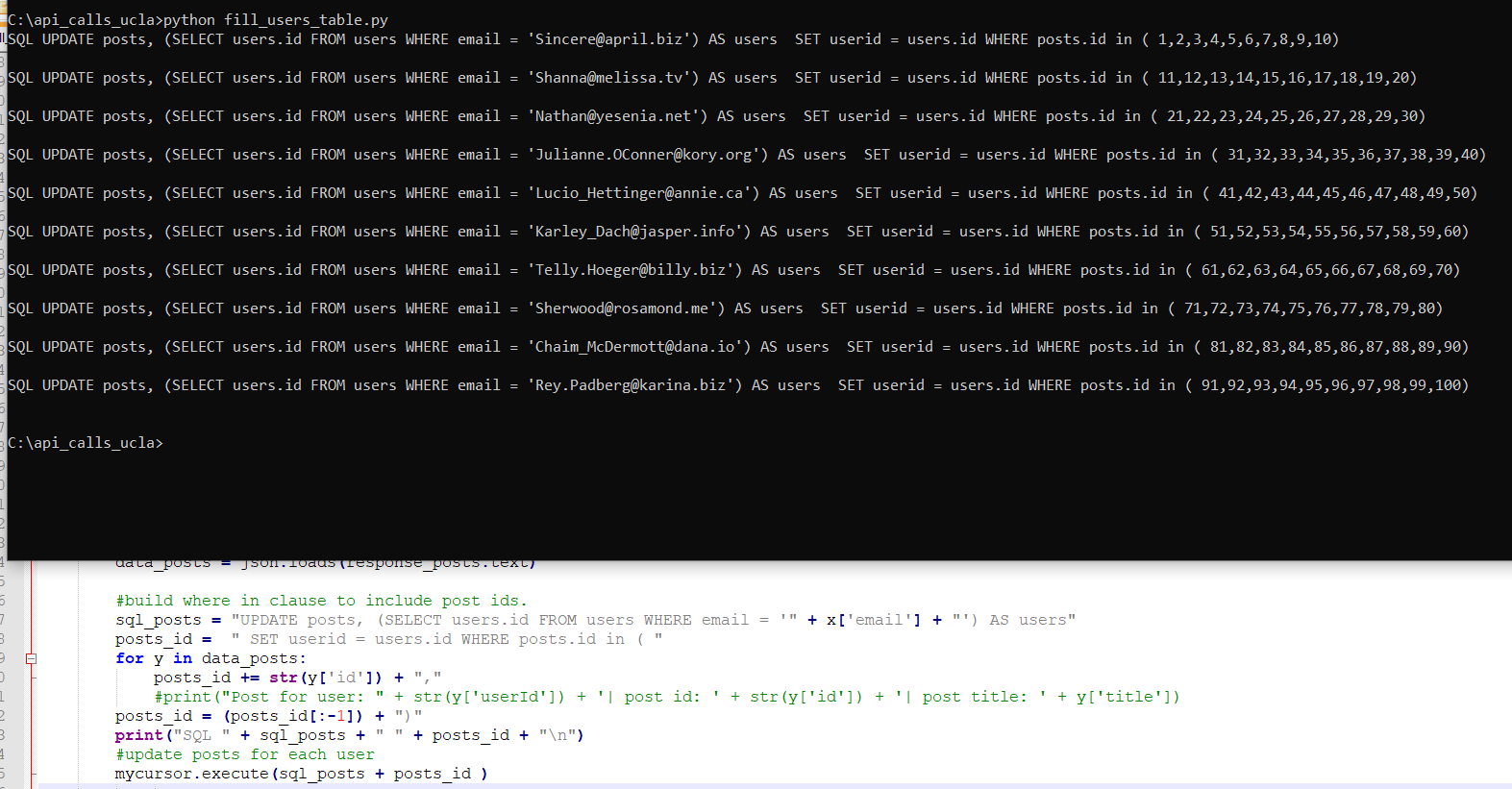


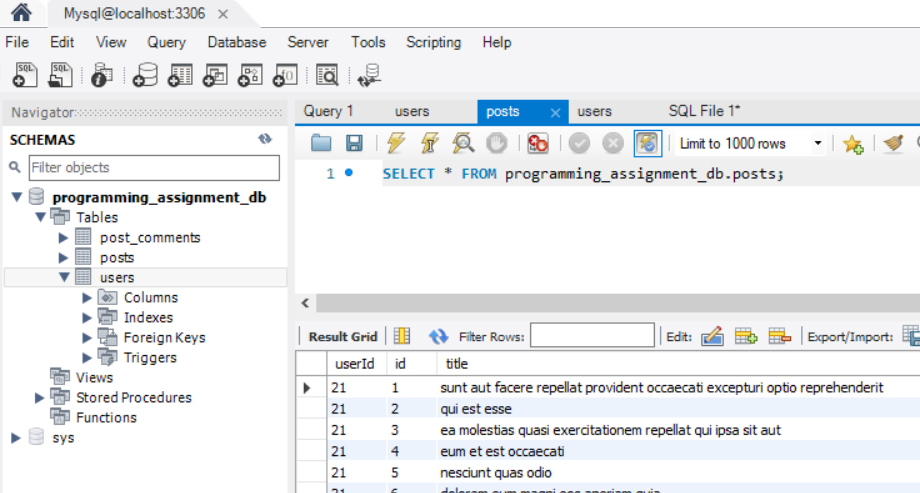
Run script to loop through users returned by api, then update users table.



While we have Json for each user, use user email to pull that user’s id (PK from users table) and use that id to update posts table. This is done by calling <https://jsonplaceholder.typicode.com/posts?userId=4> for each user inside for loop and building SQL (in clause for posts table that contains all the post ids for that user) for update.

**SQL and program run results.**



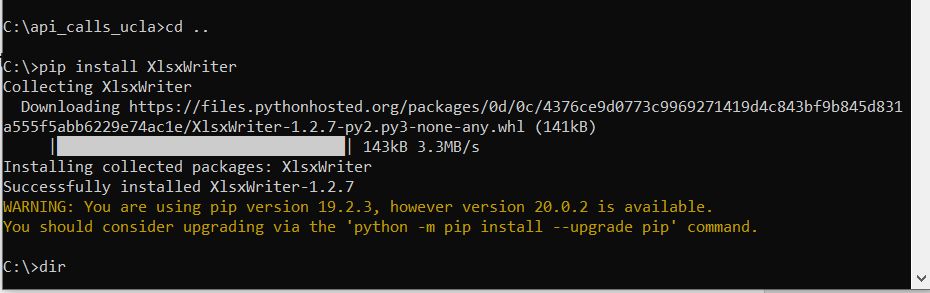


**Part 3.**

*Files: build\_commenters\_report.py*

1. Run script to create distance function f\_distance.sql
2. Run script to create view that supplies the report’s data v\_users\_commenters.sql
3. Execute *build\_commenters\_report.py*

To build excel report xlsxwriter package needs to be installed.



To create excel report, I decided to use a view that can be reused to accomplish task 4 and 5.

The view used function distance to calculate differences between latitude and longitude sets.

View used to pull frequent commenters

CREATE VIEW V\_users\_commenters

AS

select dt.\* , uu.username commenter\_username, uu.id commenter\_id,

-- calculate distance

uu.address\_\_geo\_\_lat commenter\_lat, uu.address\_\_geo\_\_lng commenter\_lng,

distance(poster\_lat, poster\_lng, uu.address\_\_geo\_\_lat, uu.address\_\_geo\_\_lng) distance

from users uu inner join

(

SELECT p.userid poster\_id, u.username poster\_username, u.address\_\_geo\_\_lat poster\_lat, u.address\_\_geo\_\_lng poster\_lng,

pc.email commenter\_email, count(0) nr\_of\_times\_commented

FROM posts p, post\_comments pc, users u

where p.id = pc.postId

and p.userId = u.id

group by p.userid , pc.email

having count(0) >= 3

) as dt

on uu.email = dt.commenter\_email

order by nr\_of\_times\_commented desc

Function used to calculate distance

DELIMITER $$

CREATE FUNCTION `Distance`(

lat1 FLOAT, lon1 FLOAT,

lat2 FLOAT, lon2 FLOAT

) RETURNS float(10,3)

NO SQL

DETERMINISTIC

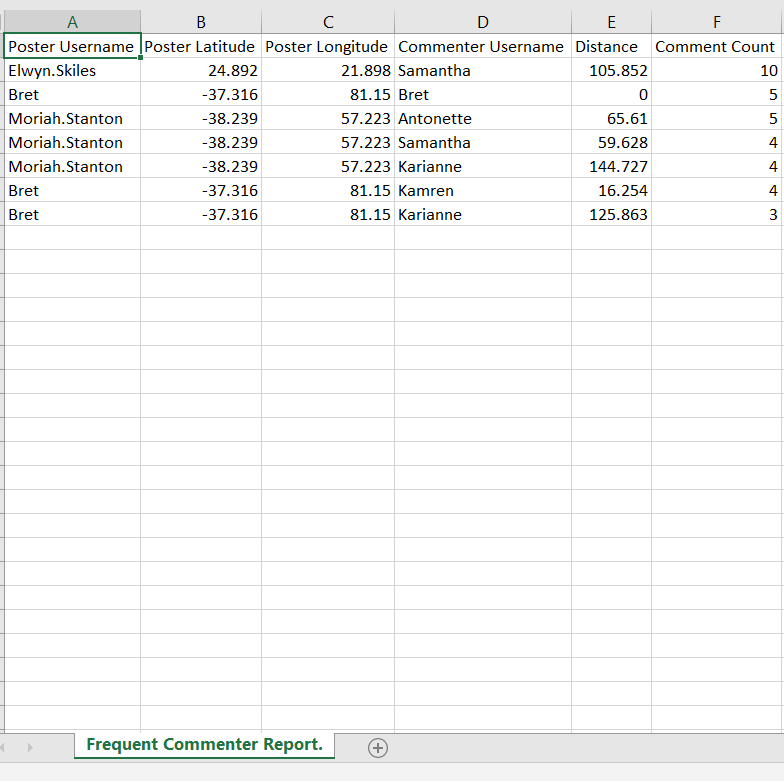
BEGIN

RETURN DEGREES(ACOS( COS(RADIANS(lat1)) \* COS(RADIANS(lat2)) \* COS(RADIANS(lon2) - RADIANS(lon1)) + SIN(RADIANS(lat1)) \* SIN(RADIANS(lat2)) ));

END$$

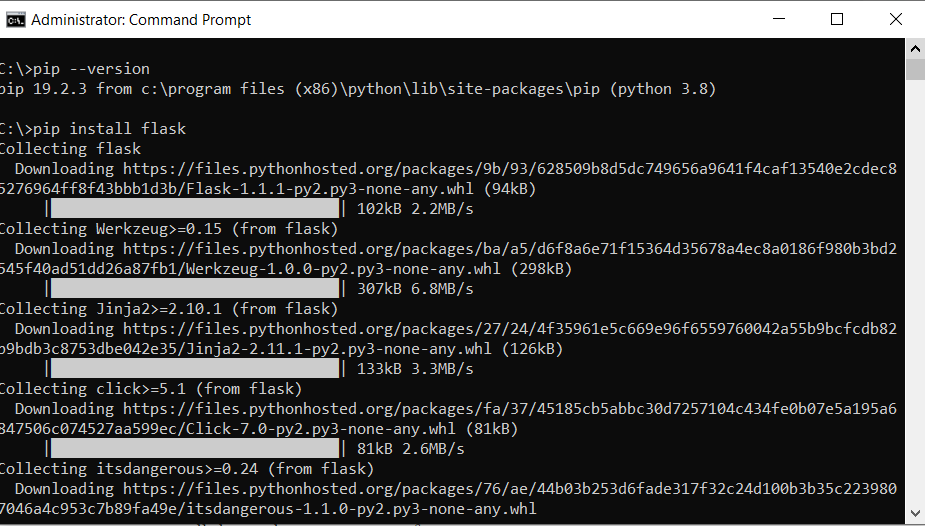
DELIMITER ;

Generated report.

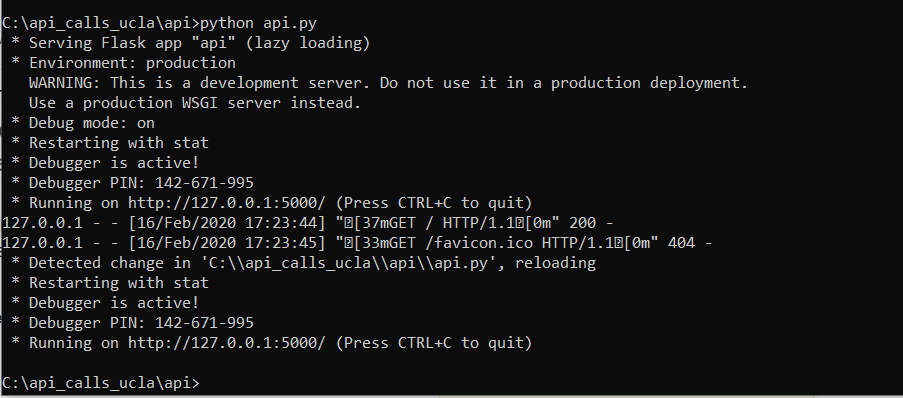


**Part 4.**

Install flask to build API



API run page



Testing API using PostMan

http://127.0.0.1:5000/api/v1/posters?id=21

Note: there 3 frequent users records – json needs to be modified to loop through rest of frequent users and add elements – struggled with formatting here (for this report user Bret has 3 frequent users).

