Project Final Report

by
Snehitha Mamidi

for

Comp 851 Spring 2021

GitHub Link :

https://github.com/Snehitha98/COMP851-Final-Project.git

Project Topic :

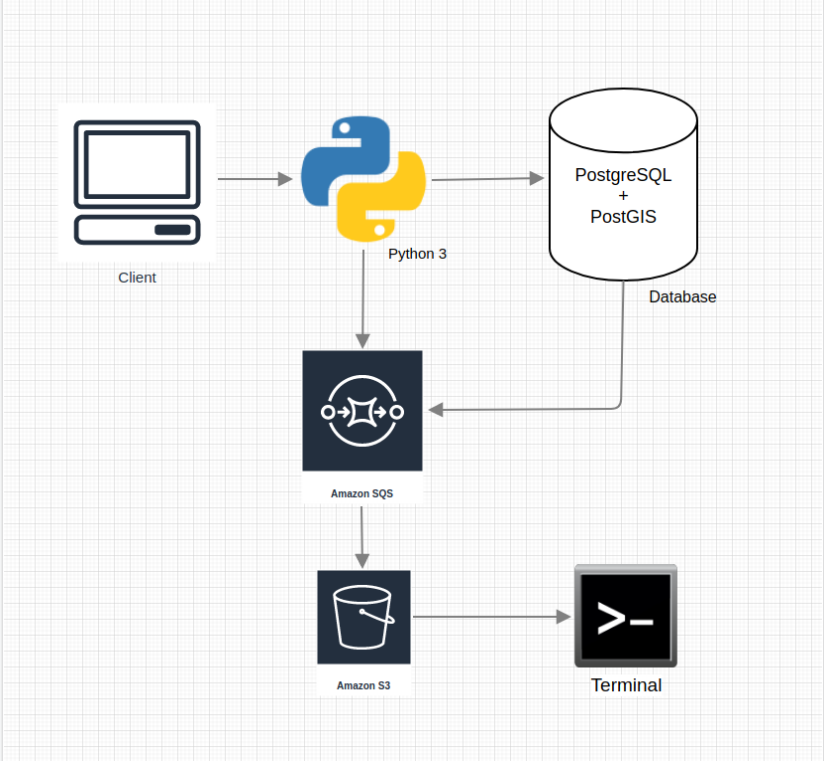
I have chosen the second part for the project.

Data Stream 2

The PTWC Widgets will be deployed into the field and communicate their GPS position. In order to prepare field operations, we will need to establish a database which can determine the proximity of Widgets to county and township locations where field operators may be stationed, or sent. In order to do this, we must deploy a GIS database, called PostGIS and ingest the city lat / long positions. In addition, we must notify and record the ingest of these positions in preparation for the location of the field operators and Widget positions. We must do so using the AWS SQS, and SNS/ SES interfaces in order to send the notifications and emails, and finally deposit log entries on s3.

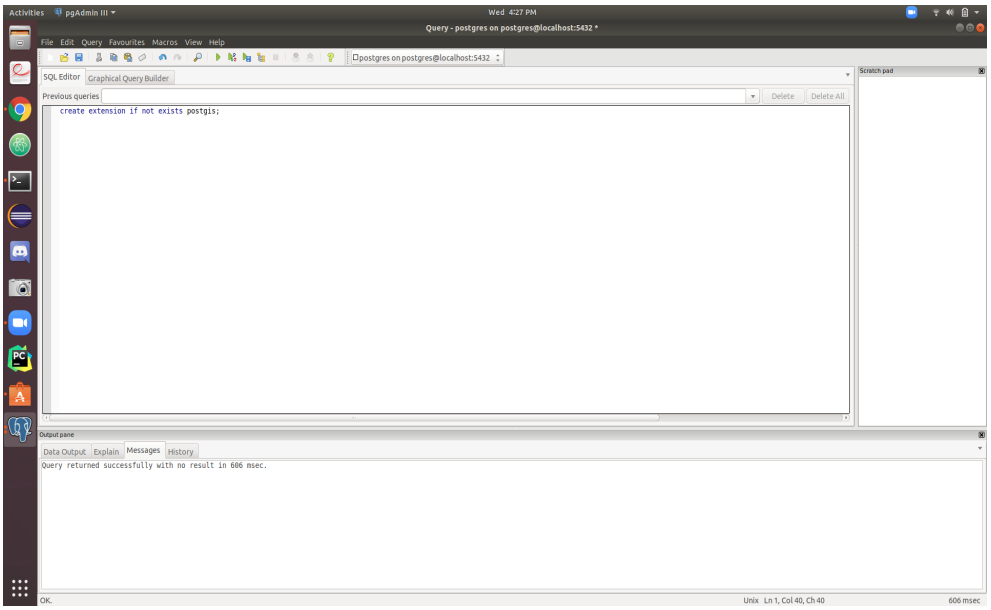
I implemented using Python, version 3.

Design of the solution using draw.io :

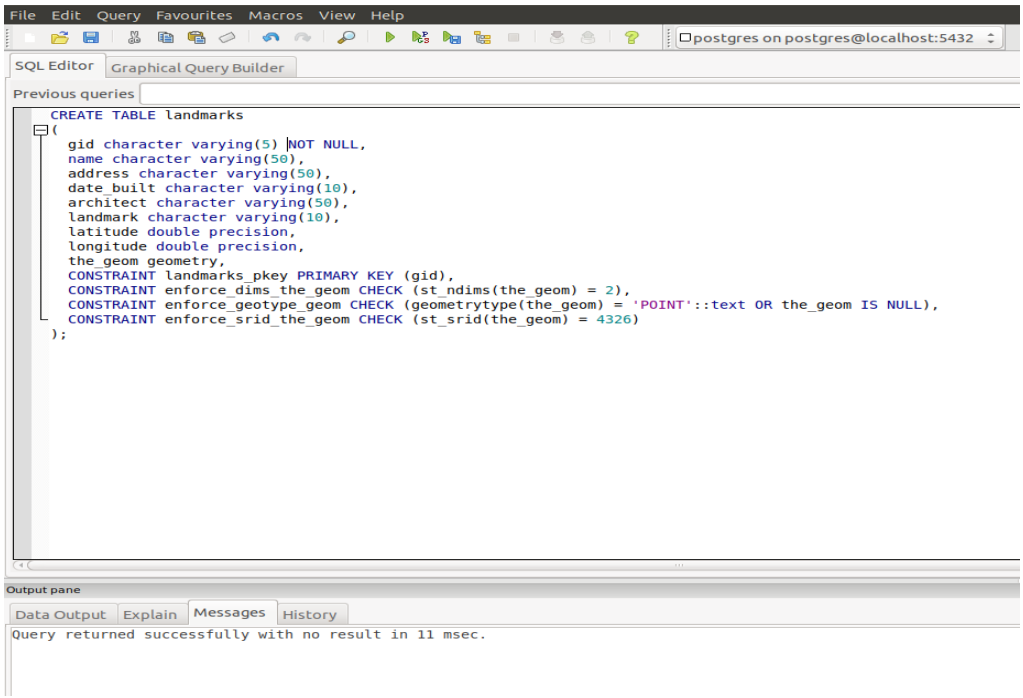


Implementation of the project using PostGres GUI:

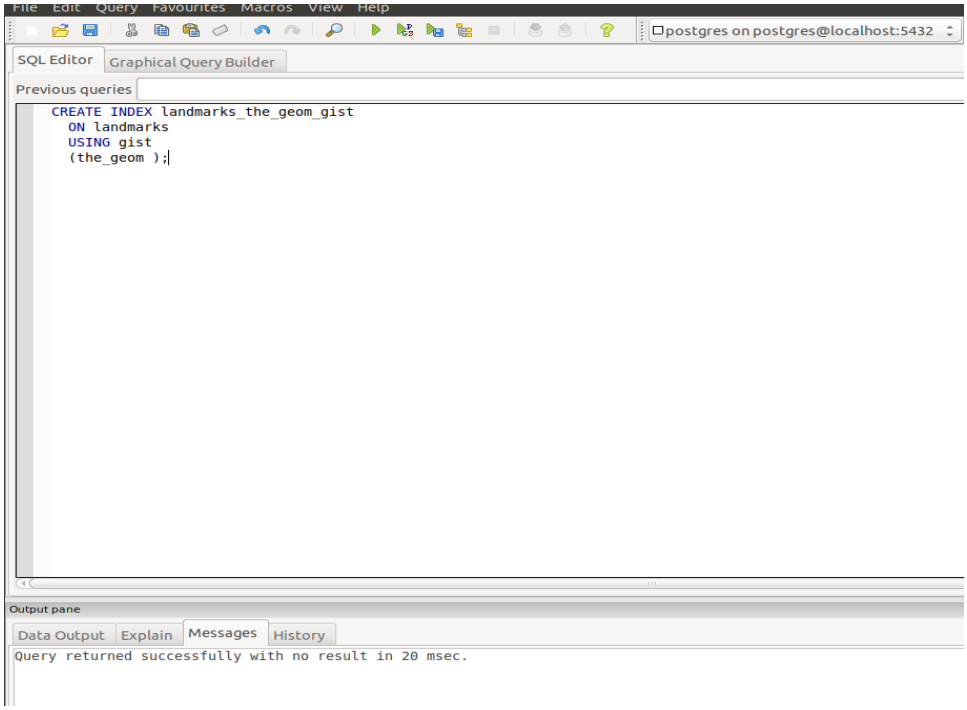
Step 1: Create PostGIS Extension



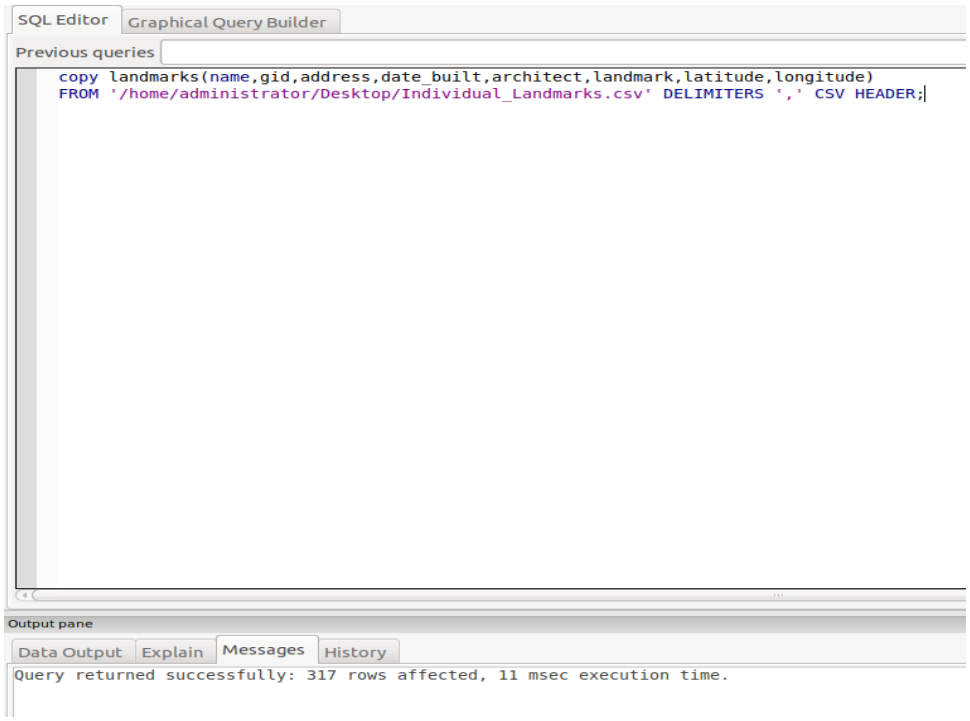
Step 2: Create the table in the database



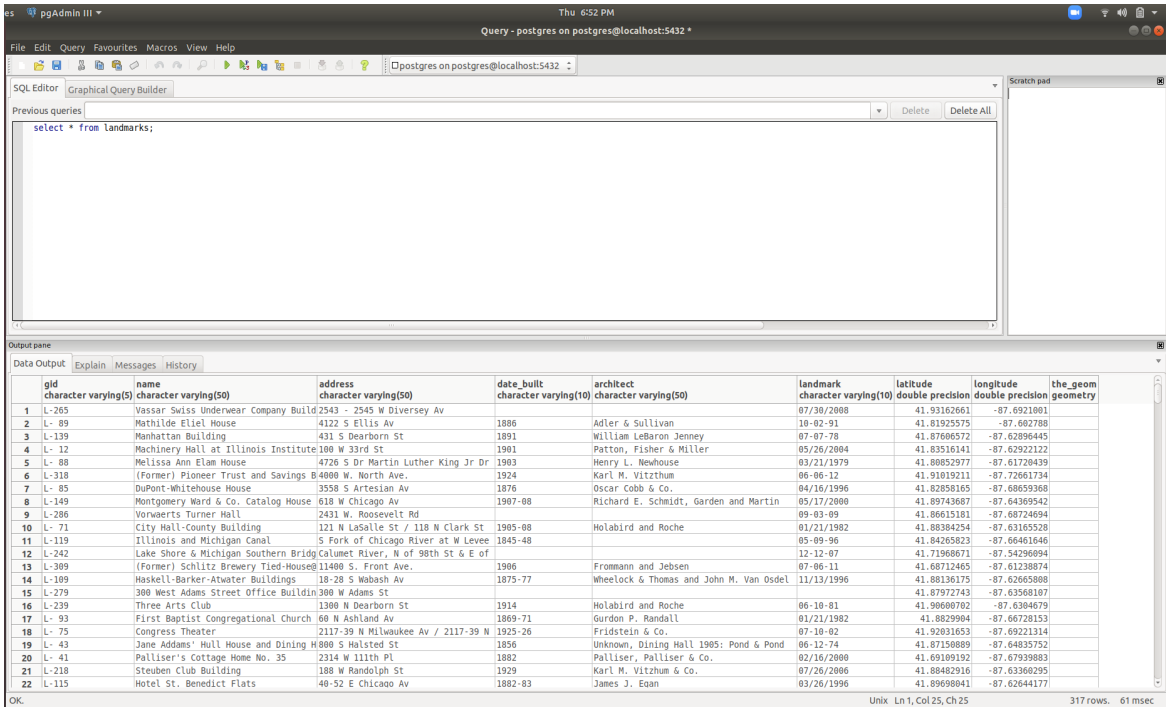
Step 3: Create Index



Step 4: Copy the CSV data into the database

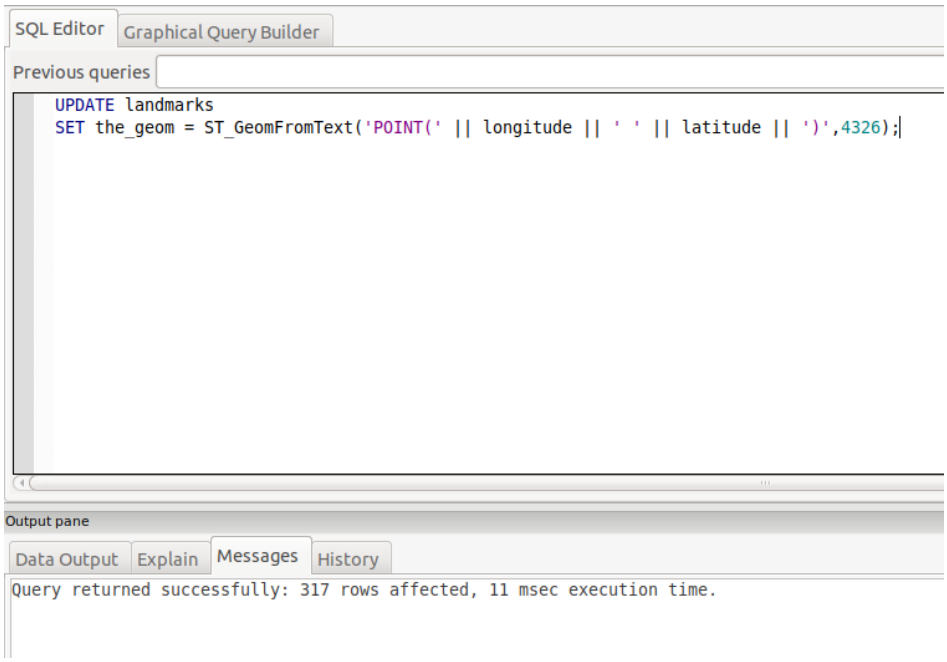


Below is the screenshot of the csv data imported :



	gid character varying(5)	name character varying(50)	address character varying(50)	date_built character varying(10)	architect character varying(50)	landmark character varying(10)	latitude double precision	longitude double precision	the_geom geometry
1	L-265	Vassar Swiss Underwear Company Build	2543 - 2545 W Diversey Av			07/30/2008	41.93162661	-87.6921001	
2	L- 89	Mathilde Eliel House	4122 S Ellis Av	1886	Adler & Sullivan	10-02-91	41.81925575	-87.602788	
3	L-139	Manhattan Building	431 S Dearborn St	1891	William LeBaron Jenney	07-07-78	41.87606572	-87.62896445	
4	L- 12	Machinery Hall at Illinois Institute	100 W 33rd St	1901	Patton, Fisher & Miller	05/26/2004	41.83516141	-87.62922122	
5	L- 88	Melissa Ann Elam House	4726 S Dr Martin Luther King Jr Dr	1903	Henry L. Newhouse	03/21/1979	41.80852977	-87.61720439	
6	L-318	(Former) Pioneer Trust and Savings B	4000 W. North Ave.	1924	Karl M. Vitzthum	06-06-12	41.91019211	-87.72661734	
7	L- 85	DuPont-Whitehouse House	3558 S Artesian Av	1876	Oscar Cobb & Co.	04/16/1996	41.82858165	-87.68659368	
8	L-149	Montgomery Ward & Co. Catalog House	618 W Chicago Av	1907-08	Richard E. Schmidt, Garden and Martin	05/17/2000	41.89743687	-87.64369542	
9	L-286	Vorwaerts Turner Hall	2431 W. Roosevelt Rd			09-03-09	41.86615181	-87.68724694	
10	L- 71	City Hall-County Building	121 N LaSalle St / 118 N Clark St	1905-08	Holabird and Roche	01/21/1982	41.88384254	-87.63165528	
11	L-119	Illinois and Michigan Canal	S Fork of Chicago River at W Levee	1845-48		05-09-96	41.84265823	-87.66461646	
12	L-242	Lake Shore & Michigan Southern Bridg	Calumet River, N of 98th St & E of			12-12-07	41.71968671	-87.54296094	
13	L-309	(Former) Schlitz Brewery Tied-House@	11400 S. Front Ave.	1906	Frommann and Jebsen	07-06-11	41.68712465	-87.61238874	
14	L-109	Haskell-Barker-Atwater Buildings	18-28 S Wabash Av	1875-77	Wheelock & Thomas and John M. Van Osdel	11/13/1996	41.88136175	-87.62665808	
15	L-279	300 West Adams Street Office Buildin	300 W Adams St				41.87972743	-87.63568107	
16	L-239	Three Arts Club	1300 N Dearborn St	1914	Holabird and Roche	06-10-81	41.90600702	-87.6304679	
17	L- 93	First Baptist Congregational Church	60 N Ashland Av	1869-71	Gurdon P. Randall	01/21/1982	41.8829904	-87.66728153	
18	L- 75	Congress Theater	2117-39 N Milwaukee Av / 2117-39 N	1925-26	Fridstein & Co.	07-10-02	41.92031653	-87.69221314	
19	L- 43	Jane Addams' Hull House and Dining H	800 S Halsted St	1856	Unknown, Dining Hall 1905: Pond & Pond	06-12-74	41.87150889	-87.64835752	
20	L- 41	Palliser's Cottage Home No. 35	2314 W 111th Pl	1882	Palliser, Palliser & Co.	02/16/2000	41.69109192	-87.67939883	
21	L-218	Steuben Club Building	188 W Randolph St	1929	Karl M. Vitzhum & Co.	07/26/2006	41.88482916	-87.63360295	
22	L-115	Hotel St. Benedict Flats	40-52 E Chicago Av	1882-83	James J. Egan	03/26/1996	41.89698041	-87.62644177	

Step 5: Translate latitude and longitude into PostGIS POINT geometry



Step 6: Write a PostGIS query to display the closest 5 landmarks for the given latitude and longitude

SQL Editor | Graphical Query Builder

Previous queries

```
-- distance units in planar degrees. 4326 is WGS 84 and the long lat units are degrees
SELECT
ST_Distance(ST_GeomFromText('POINT(-87.6348345 41.8786207)', 4326), landmarks.the_geom) AS planar_degrees,
name,
architect
FROM landmarks
ORDER BY planar_degrees ASC
LIMIT 5;
```

Output pane

Data Output | Explain | Messages | History

	planar_degrees double precision	name character varying(50)	architect character varying(50)
1	0.0007463848779255456	Brooks Building	Holabird & Roche
2	0.0013933886958850995	300 West Adams Street Office Buildin	
3	0.0018793811696624153	Continental And Commercial National	
4	0.002712009853856664	Chicago Board of Trade Building	Holabird & Root
5	0.00307102762438656	Rookery Building	Burnham & Root

Implementation of the project using Python3:

```
import psycopg2
import boto3
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT



try:
   # boto3
   sqs = boto3.resource('sqs',aws_access_key_id =
'xxxxxxxxxxxxxxxx',
                        aws_secret_access_key =
'xxxxxxxxxxxxxxxxxxxxxxx')
   queue = sqs.create_queue(QueueName='sm1552_pwtc',
Attributes={'DelaySeconds': '3'})
```

```
   # connecting to postgis
   connection = psycopg2.connect(user="postgres",
                                 password="CT13root",
                                 host="localhost")
   connection.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT);
   cursor = connection.cursor()

   # create database
   cursor.execute("drop database if exists pwtc;")

   create_database = """create database pwtc;"""
   cursor.execute(create_database)
   connection.commit()

   # create extension postgis if not exists
   create_extension_query = """create extension if not exists
postgis;"""
   cursor.execute(create_extension_query)
   connection.commit()

   # Create the table landmarks in the database
   create_tables_landmarks = """  CREATE TABLE landmarks
(
  gid character varying(5) NOT NULL,
  name character varying(50),
  address character varying(50),
  date_built character varying(10),
  architect character varying(50),
  landmark character varying(10),
  latitude double precision,
  longitude double precision,
  the_geom geometry,
  CONSTRAINT landmarks_pkey PRIMARY KEY (gid),
  CONSTRAINT enforce_dims_the_geom CHECK (st_ndims(the_geom) = 2),
  CONSTRAINT enforce_geotype_geom CHECK (geometrytype(the_geom) =
'POINT'::text OR the_geom IS NULL),
  CONSTRAINT enforce_srid_the_geom CHECK (st_srid(the_geom) = 4326)
);
"""
```

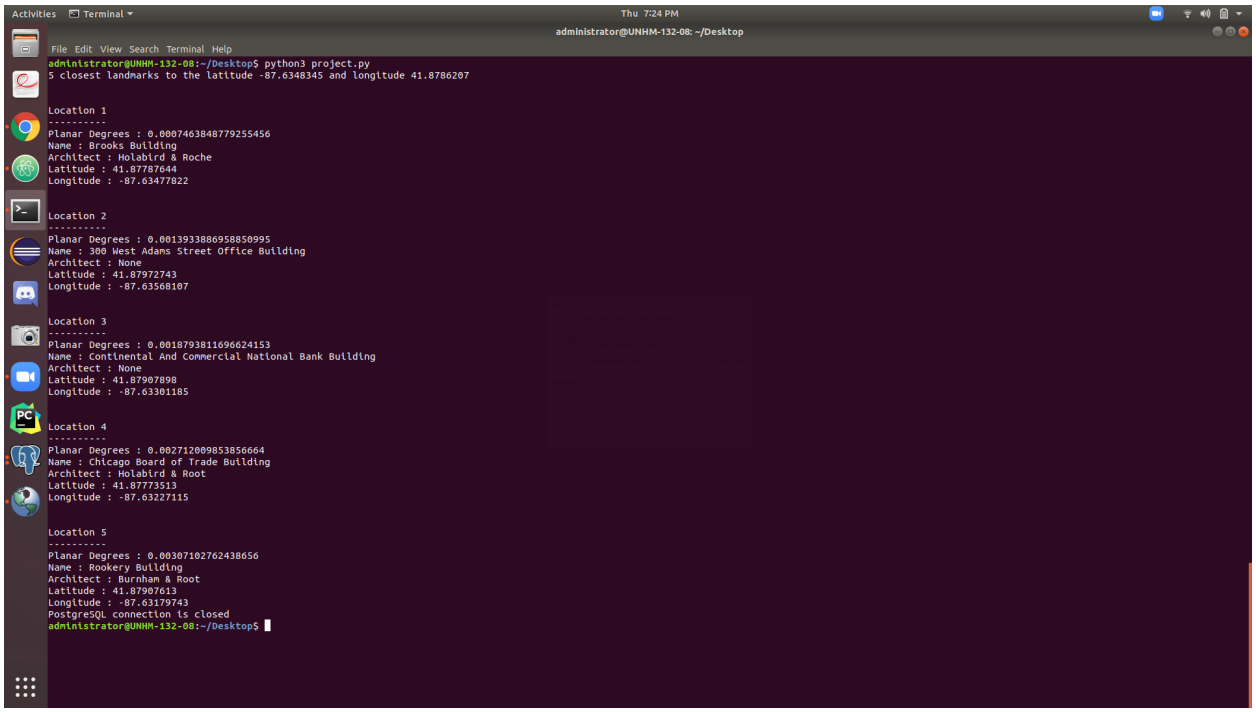
```
   cursor.execute(create_tables_landmarks)
   connection.commit()

   # create index
   create_index_landmarks = """ CREATE INDEX if not exists
landmarks_the_geom_gist ON landmarks USING gist (the_geom )"""
   cursor.execute(create_index_landmarks)
   connection.commit()

   # Copy the CSV data into the database
   insert_data = """ copy
landmarks(name,gid,address,date_built,architect,landmark,latitude,lon
gitude) FROM '/home/administrator/Desktop/Individual_Landmarks.csv'
DELIMITERS ',' CSV HEADER """
   cursor.execute(insert_data)
   connection.commit()

   # sending insertion info to queue
   response =
queue.send_message(MessageBody='Landmarks',MessageAttributes={
      'Insertion':{
         'StringValue':'Data Uploaded Successfully!!!',
         'DataType':'String'
         }})


   queue = sqs.get_queue_by_name(QueueName='sm1552_pwtc')

   # Translate latitude and longitude into POINT geometry
   update_table = """UPDATE landmarks SET the_geom =
ST_GeomFromText('POINT(' || longitude || ' ' || latitude || ')',4326)
"""
   cursor.execute(update_table)
   connection.commit()

   # This query returns the 5 closest landmarks to a given latitude
and longitude
   select_statement = """SELECT distinct
ST_Distance(ST_GeomFromText('POINT(-87.6348345 41.8786207)', 4326),
```

```
landmarks.the_geom) AS planar_degrees,
name,
architect, latitude, longitude
FROM landmarks
ORDER BY planar_degrees ASC
LIMIT 5 """
   count = 1
   cursor.execute(select_statement)
   connection.commit()
   location_details=[]
   records = cursor.fetchall()
   print("\n")
   print(f'5 closest landmarks to the latitude -87.6348345 and
longitude 41.8786207')

   for row in records:
       print("\n")
       print("Location " + str(count))
       print("----------")
       print("Planar Degrees : " + str(row[0]))
       print("Name : " + str(row[1]))
       print("Architect : " + str(row[2]))
       print("Latitude : "+ str(row[3]))
       print("Longitude : "+ str(row[4]))

       count +=1
       location_details.append(str(row[0]))
       location_details.append(str(row[1]))
       location_details.append(str(row[2]))
       location_details.append(str(row[3]))
       location_details.append(str(row[4]))

   # sending location data to the queue
   response =
queue.send_message(MessageBody='Landmarks',MessageAttributes={
      'Locations':{
         'StringValue':",".join(location_details),
         'DataType':'String'
         }})
```

```
    connection.commit()


except (Exception, psycopg2.Error) as error :
    if(connection):
        print(error)

finally:
    # closing database connection.
    if(connection):
        cursor.close()
        connection.close()
        print("\n")
        print("PostgreSQL connection is closed")
```

Output:

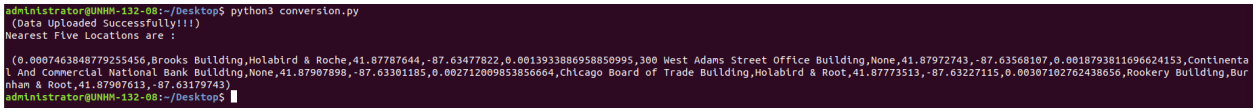
1. **Displaying 5 closest landmarks to a given latitude and longitude**



2. **Processing messages from queues :**

```
import boto3

sqs = boto3.resource('sqs')

# Get the queue
queue = sqs.get_queue_by_name(QueueName='sm1552_pwtc')

for message in
queue.receive_messages(MessageAttributeNames=['Insertion']):
    # Get the custom author message attribute if it was set
    author_text = ''
    if message.message_attributes is not None:
        author_name =
message.message_attributes.get('Insertion').get('StringValue')
        if author_name:
            author_text = ' ({0})'.format(author_name)

    # Print out the body and author (if set)
    print('{0}'.format(author_text))
print("Nearest Five Locations are : \n")
# Process messages by printing out body and optional author name
for message in
queue.receive_messages(MessageAttributeNames=['Locations']):
    # Get the custom author message attribute if it was set
    author_text = ''
    if message.message_attributes is not None:
        author_name =
message.message_attributes.get('Locations').get('StringValue')
        if author_name:
            author_text = ' ({0})'.format(author_name)

    # Print out the body and author (if set)
    print('{0}'.format(author_text))
```

Output :



Challenges :

After sending 5 locations to the queues, the locations are generated in a tuple and then I tried sending to the queue in list format. To achieve this I had issues, as we need to convert these lists into string to display as messages through queues.