**Final Project**

**For**

**System Integration Fall 2020**

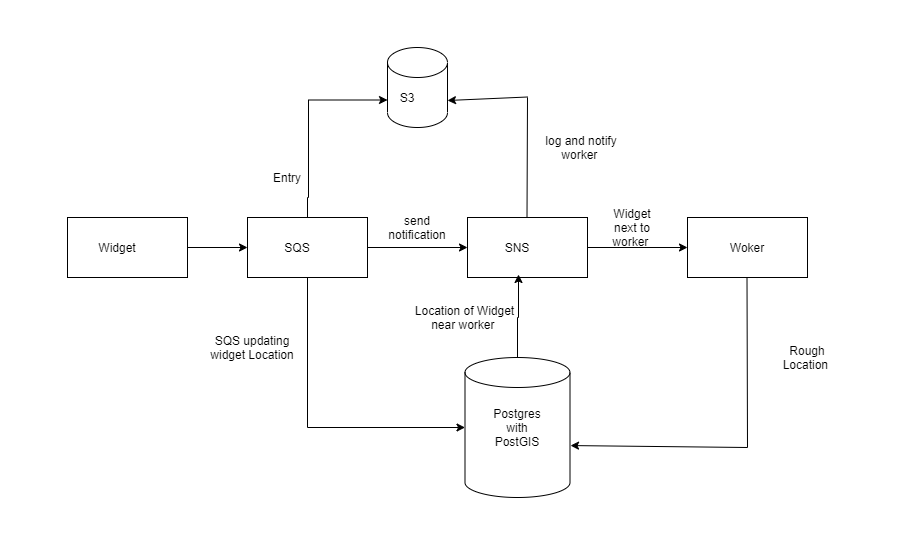
**By**

**Bishnu Poudyal**

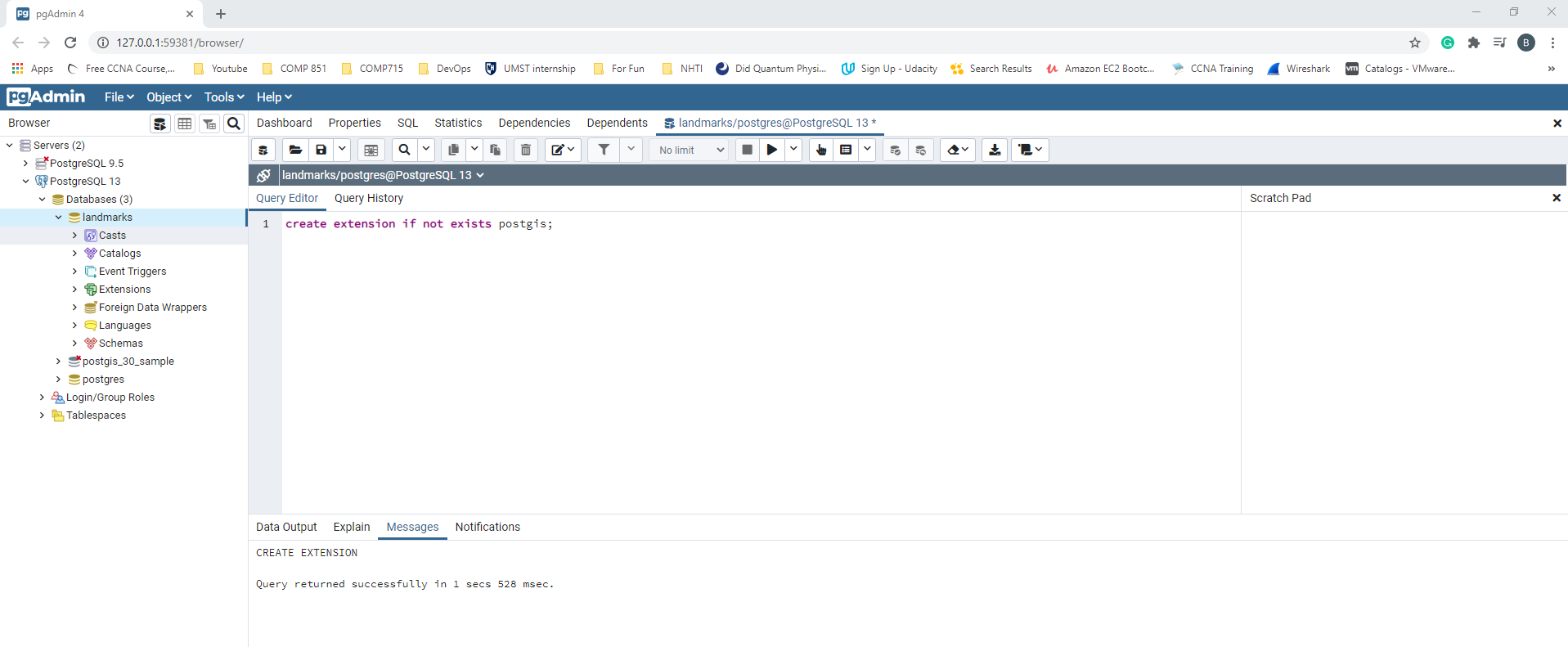
**GitHub Link:** <https://github.com/bipoudyal16/SI_Final_Project.git>

**Project Topic:**

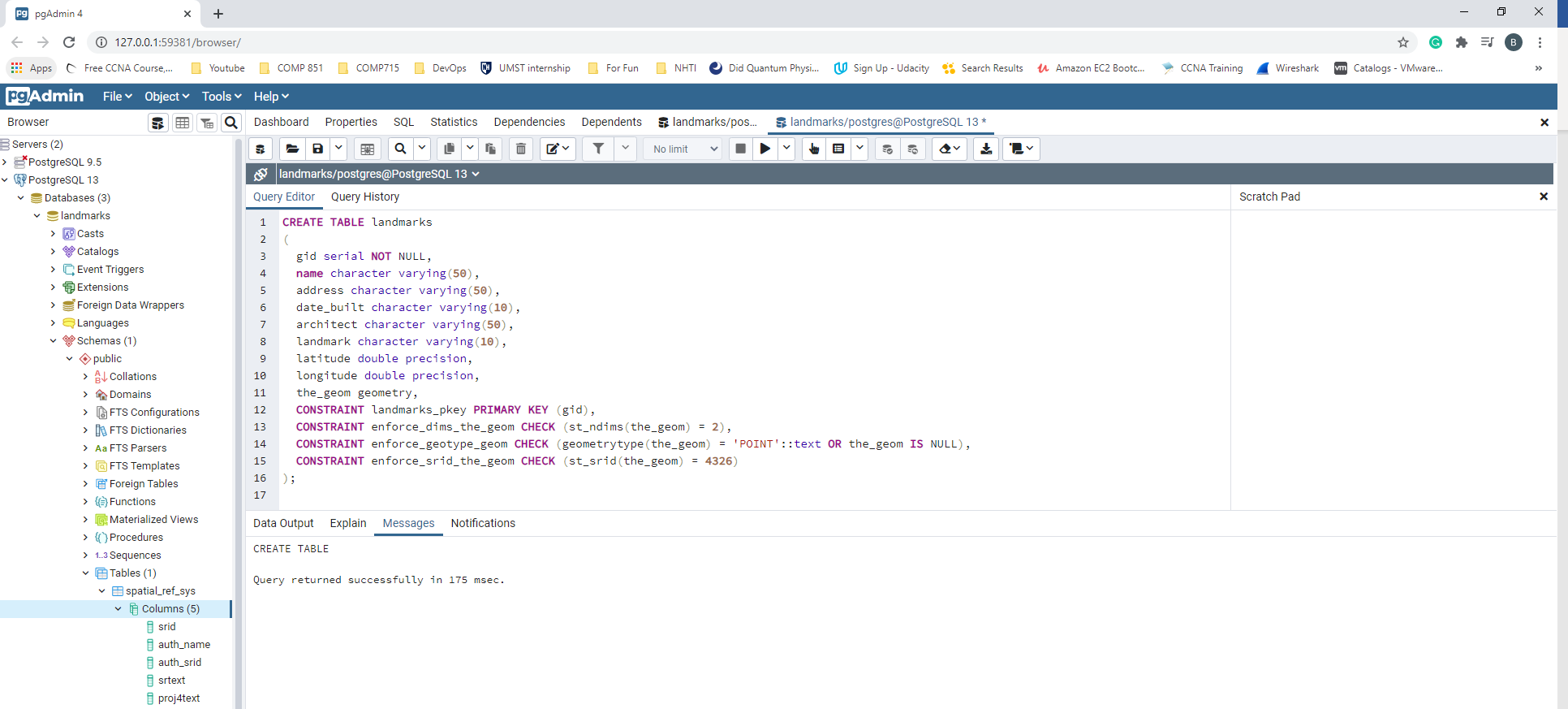
I have chosen option two for my project. In this project I have deployed PTWC Widgets as a communication GPS position for field operations. I am implementing database which can findout the widgets in the locations where worker is located. This will use PostGIS database and using latitude and longitude to provide site info.



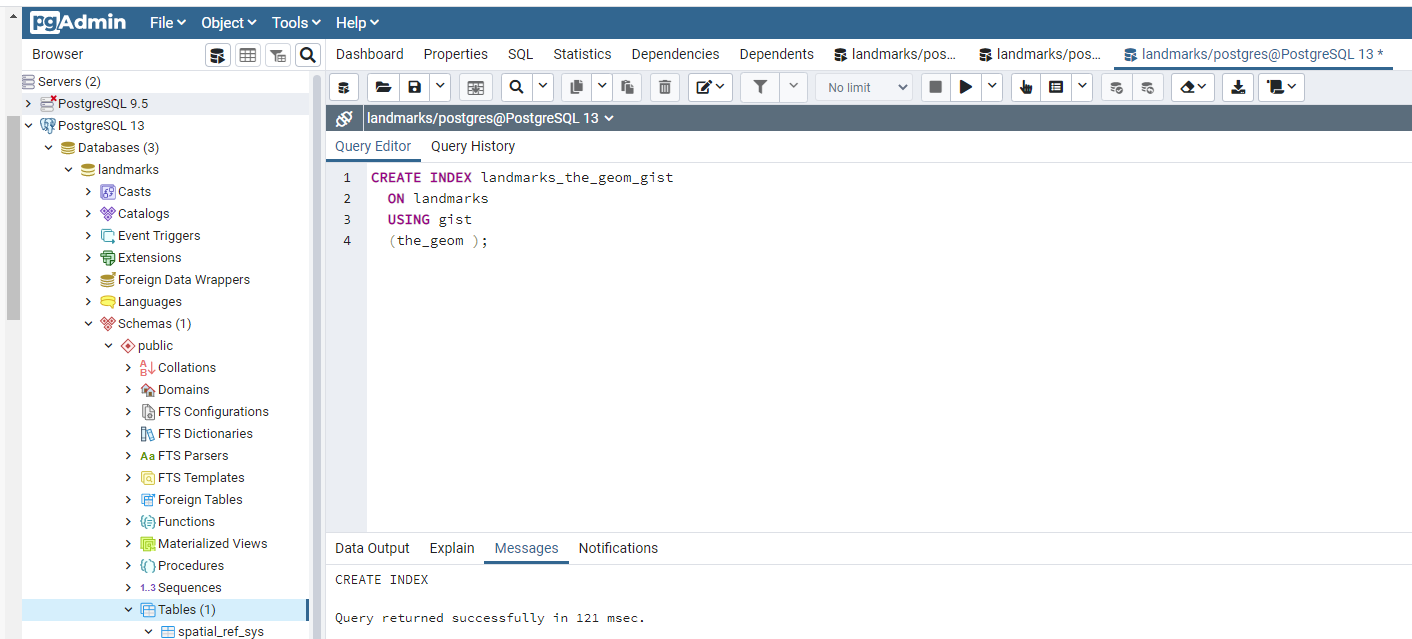
**Step 1:** Creating postgis



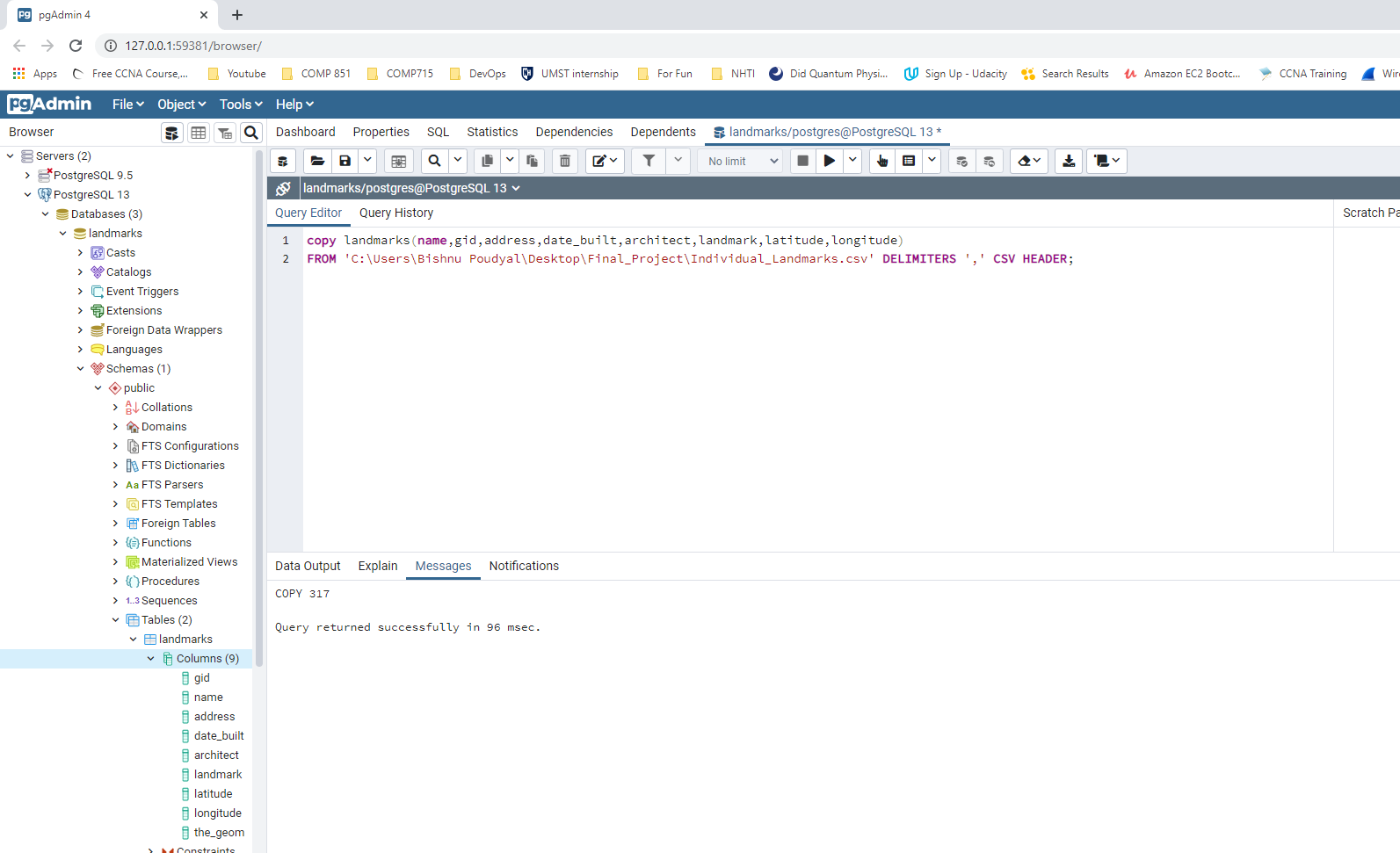
**Step 2:** Create the table in database

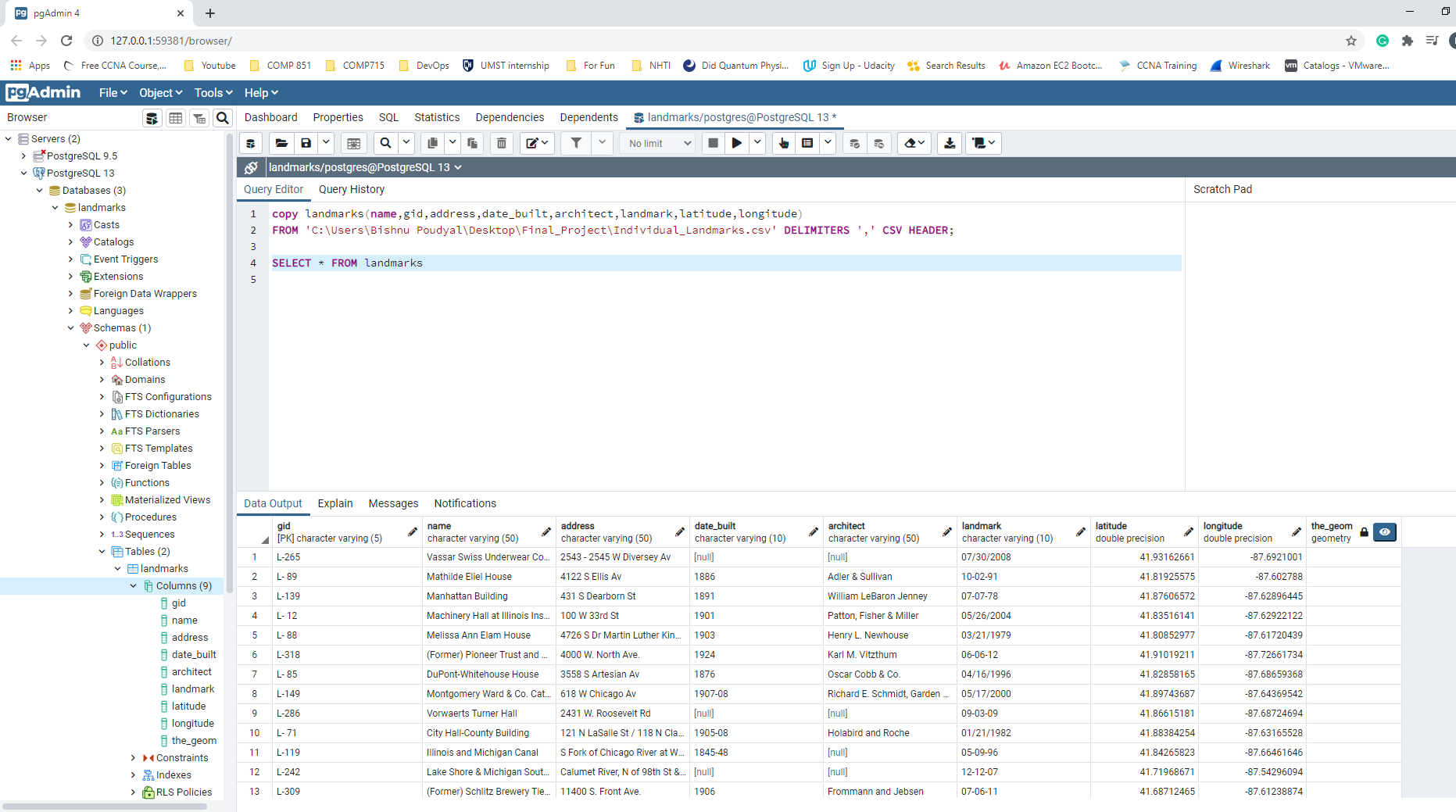


**Step 3:** Create Index

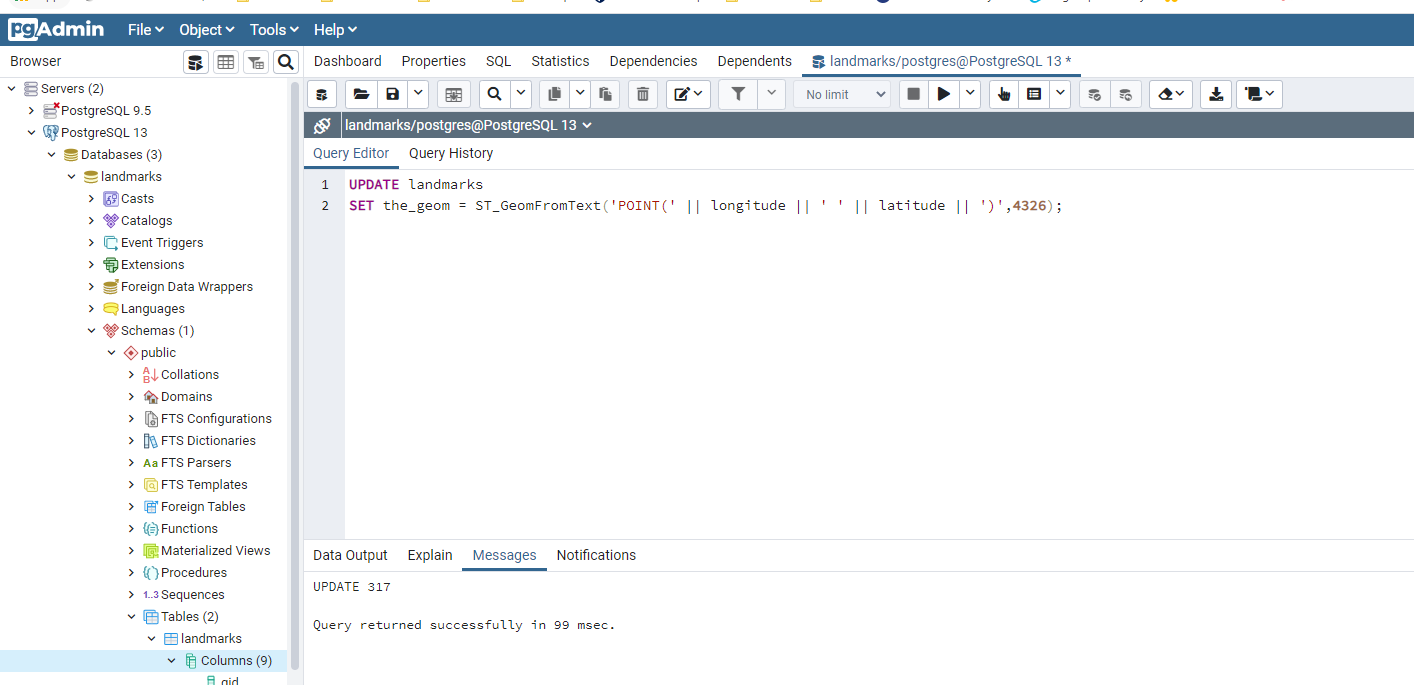


**Step 4:** Copy the CSV data into the Database

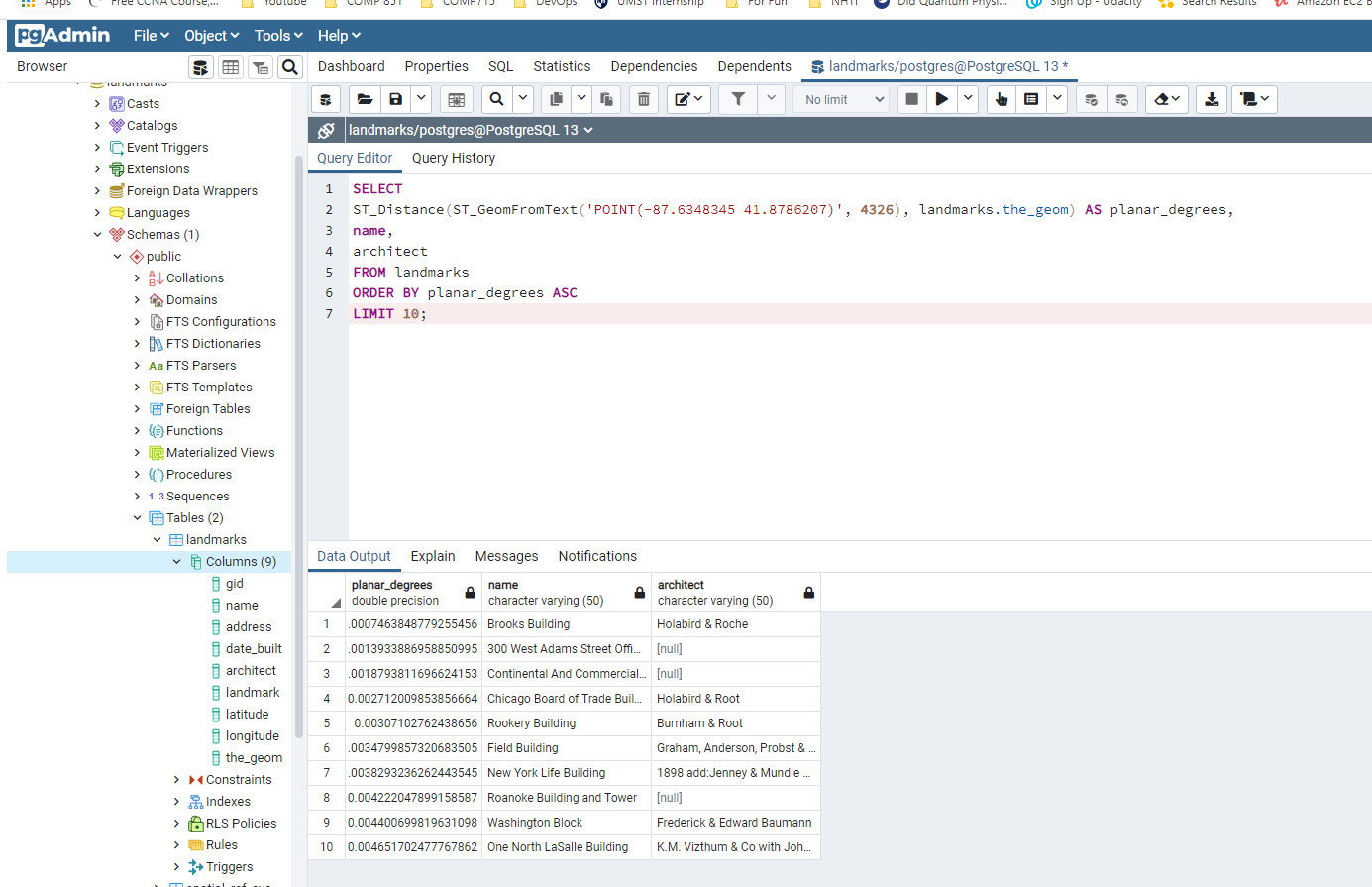




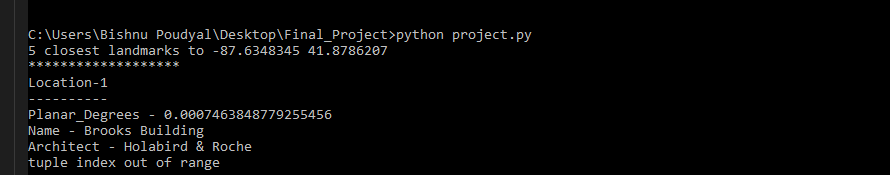
**Step 5:** Translate latitude and longitude into POINT geometry

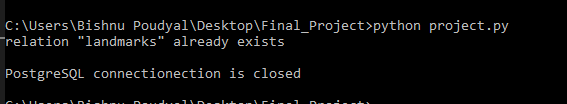


**Step 6:** Writing PostGIS queries to display 10 location for this latitude and longitude



**Using python for implementation:**





import psycopg2

import boto3

from psycopg2.extensions import ISOLATION\_LEVEL\_AUTOCOMMIT

import connection

try:

   #boto3 Got informtion from https://boto3.amazonaws.com/v1/documentation/api/latest/guide/sqs.html

   sqs = boto3.resource('sqs',aws\_access\_key\_id =  '',

                        aws\_secret\_access\_key= '', region\_name='us-west-2')

   queue = sqs.create\_queue(QueueName='landmarks', Attributes={'DelaySeconds': '5'})

   #connecting to postgis https://www.postgresqltutorial.com/postgresql-python/connect/

   connection = psycopg2.connect(user="postgres",

                            password="admin",

                            host="127.0.0.1")

   connection.set\_isolation\_level(ISOLATION\_LEVEL\_AUTOCOMMIT);

   cursor = connection.cursor()

   cursor.execute("drop database if exists landmarks;")

   createdatabase = """create database landmarks; """

   cursor.execute(createdatabase)

   connection.commit()

   #create extension postgis

   create\_extension\_query\_postgis = """create extension if not exists postgis;"""

   cursor.execute(create\_extension\_query\_postgis)

   connection.commit()

   #create tables and indexes in the databse called Landmark: using same SQL statements used in PGAdmin

   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\_the\_geom\_gist ON landmarks USING gist (the\_geom )"""

   cursor.execute(create\_index)

   connection.commit()

   #Copy the CSV data into the Database

   insert\_data = """copy landmarks(name,gid,address,date\_built,architect,landmark,latitude,longitude) FROM 'C:\\Users\\Bishnu Poudyal\\Desktop\\Final\_Project\\Individual\_Landmarks.csv' DELIMITERS ',' CSV HEADER """

   cursor.execute(insert\_data)

   connection.commit()

   #creating new message https://boto3.amazonaws.com/v1/documentation/api/latest/guide/sqs.html

   response = queue.send\_message(MessageBody='landmarks',MessageAttributes={

      'uploadmessage':{

         'StringValue':'Uploaded Successfully!!!',

         'DataType':'String'

         }})

   queue = sqs.get\_queue\_by\_name(QueueName='landmarks')

   #Translate latitude and longitude into POINT geometry

   table\_update = """UPDATE landmarks SET the\_geom = ST\_GeomFromText('POINT(' || longitude || ' ' || latitude || ')',4326) """

   cursor.execute(table\_update)

   connection.commit()

   #Writing PostGIS queries to display 10 location for this latitude and longitude https://www.postgresqltutorial.com/postgresql-python/query/

   selece\_queries = """SELECT distinct 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 10 """

   count = 1

   cursor.execute(selece\_queries)

   connection.commit()

   location\_details=[]

   records = cursor.fetchall()

   print("5 closest landmarks to -87.6348345 41.8786207")

   print("............................................")

   for row in records:

       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]))

       print("...................................")

       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]))

       location\_details.append(str(row[5]))

       location\_details.append(str(row[6]))

       location\_details.append(str(row[7]))

       location\_details.append(str(row[8]))

       location\_details.append(str(row[9]))

   #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 connectionection.

    if(connection):

        cursor.close()

        connection.close()

        print("PostgreSQL connectionection is closed")