Project Document

For

System Integration

Comp851. Fall 2019

Members of the team:

- Usha
- Pradeep
- Roopesh
- Rahul
- Surya

The Project First Phase

GitHub Link for the project: https://github.com/shiva6162/ptwc

If we are to pick a name for our team, we would be ***Team Ranger***s (Just for the sake of picking a name) 😊

Work division:

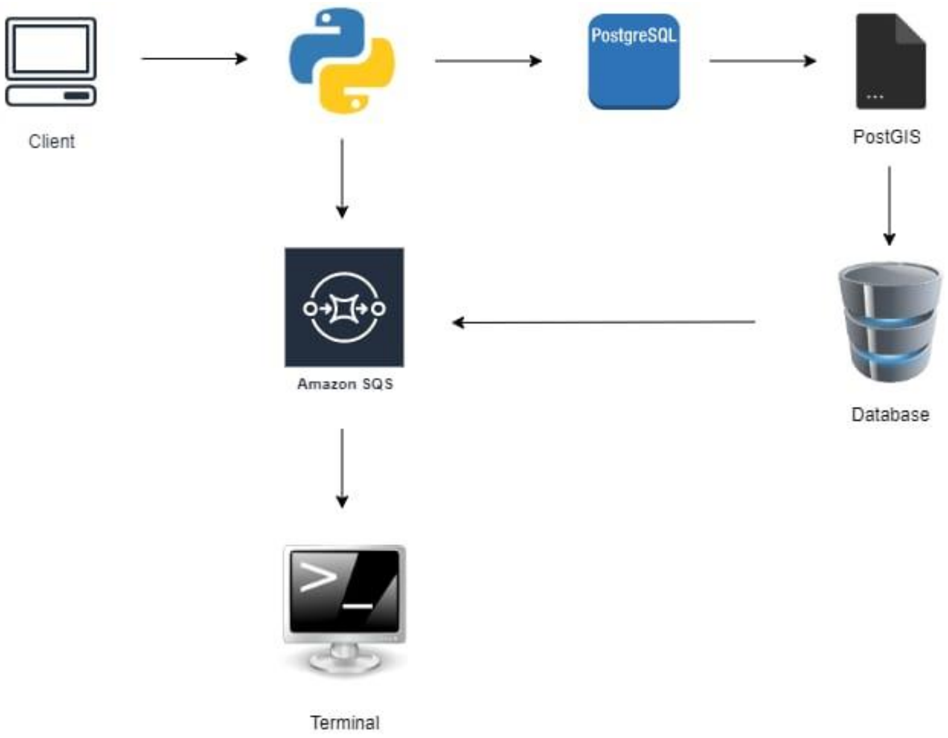
- Usha - Division of tasks and following up on task completion
- Pradeep - Creation of process flow diagram and downloading required manuals
- Roopesh - Preparing the system and coding
- Rahul - Researching online resources and help from various platforms
- Surya – PostGres implementation and Documentation

Project Topic:

We have chosen the ***second integration*** part for our project –

- The PTWC Widgets will be deployed into the field and communicate their GPS position. In order to prepare field operations, we would 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 would deploy a GIS database, called PostGIS and ingest the city latitude / longitude positions.
- In addition, we would notify and record the ingest of these positions in preparation for the location of the field operators and Widget positions. We would do so using the AWS SQS, and SNS/ SES interfaces in order to send the notifications and emails, and finally we would deposit log entries on s3.
- We would implement the same using Python3.

IO diagram explaining the process flow:



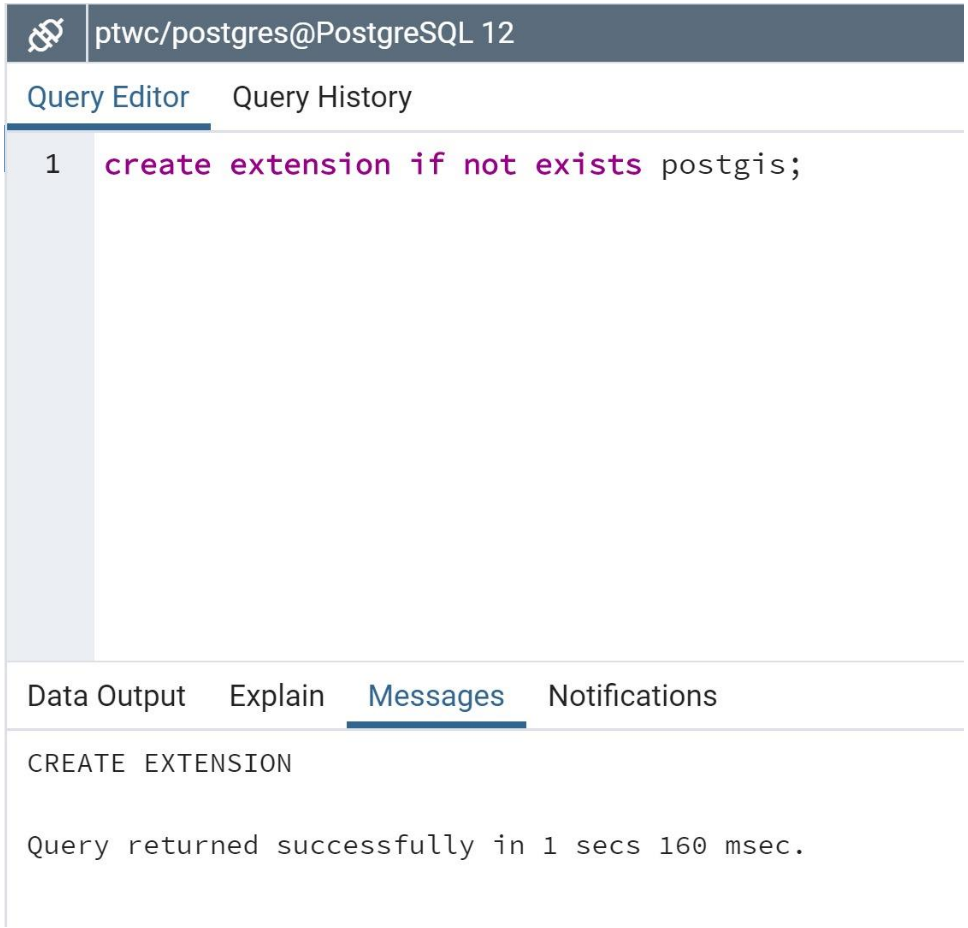
Project PoC Phase

Implementation of the project using PostGres GUI:

- Before implementing the project using python, the team has decided to implement it using the PostGres GUI.
- Our aim is to test the process flow agreed upon using the PostGres GUI and then on fetching the appropriate results we then want to use Python to implement the same.

Below is the Step-step execution flow in PostGres GUI:

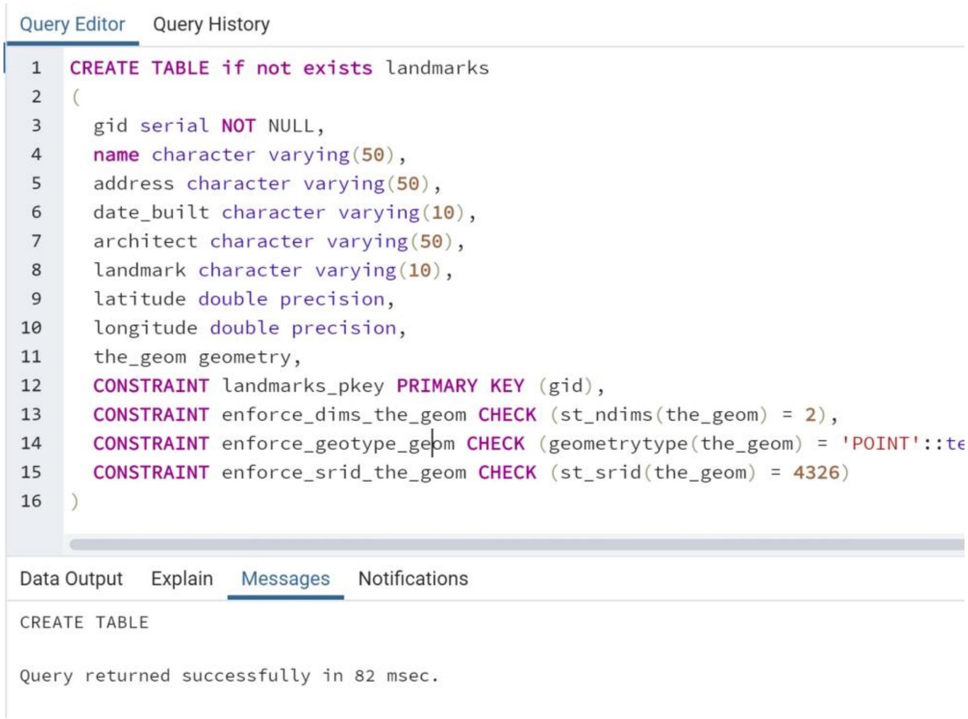
Step1: Create Post GIS Extension



```
create extension if not exists postgis;
```

```
CREATE EXTENSION

Query returned successfully in 1 secs 160 msec.
```

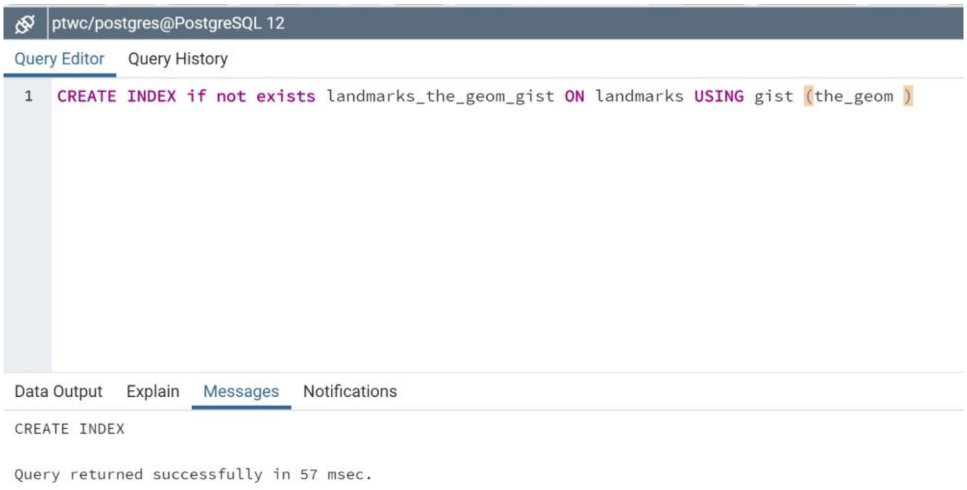
Step 2: Create Table



```sql
CREATE TABLE if not exists landmarks
(
  gid serial 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'::te
  CONSTRAINT enforce_srid_the_geom CHECK (st_srid(the_geom) = 4326)
)
```

```
CREATE TABLE

Query returned successfully in 82 msec.
```

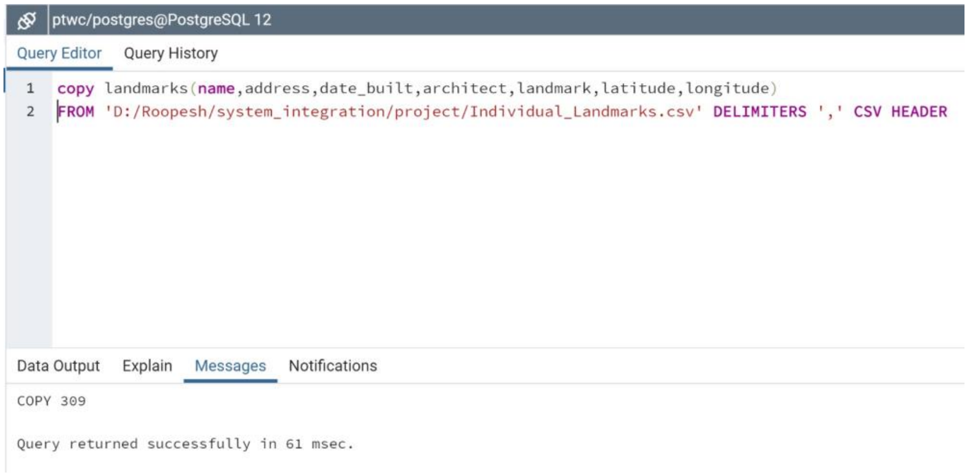
Step3: Create Index



```sql
CREATE INDEX if not exists landmarks_the_geom_gist ON landmarks USING gist (the_geom )
```

```
CREATE INDEX

Query returned successfully in 57 msec.
```

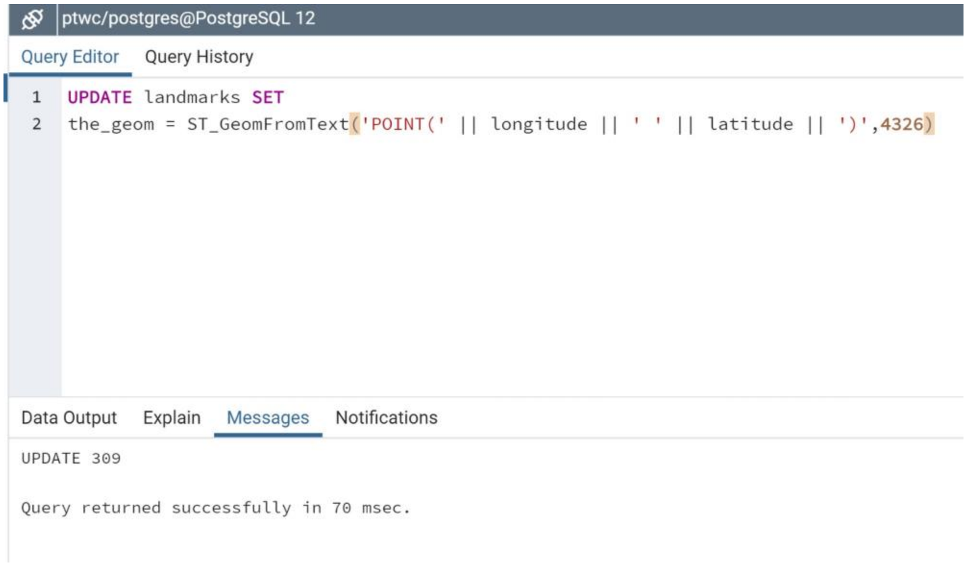
Step4: Import data from the CSV file



```
copy landmarks(name,address,date_built,architect,landmark,latitude,longitude)
FROM 'D:/Roopesh/system_integration/project/Individual_Landmarks.csv' DELIMITERS ',' CSV HEADER
```

```
COPY 309

Query returned successfully in 61 msec.
```

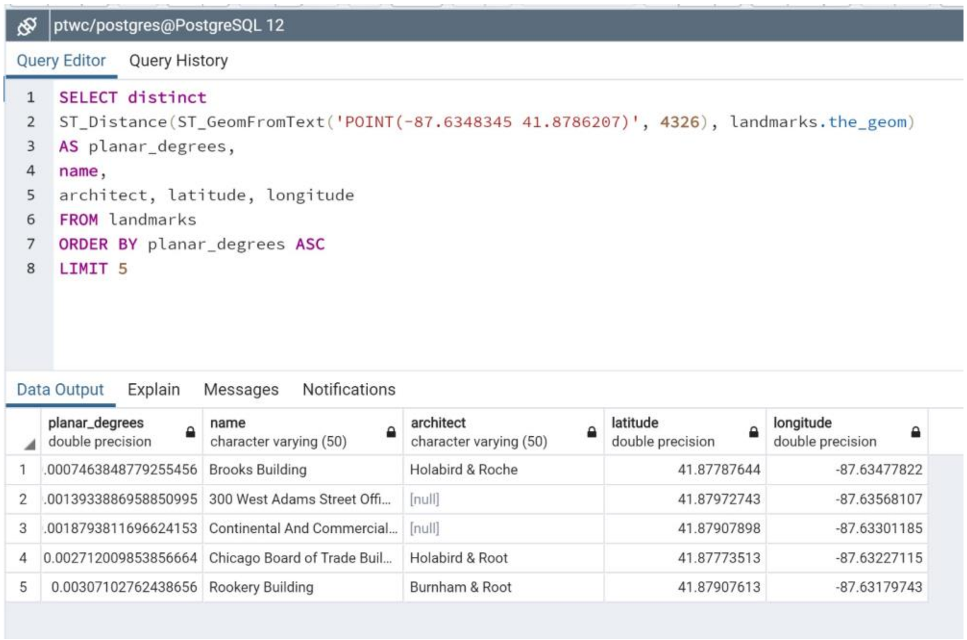
Step5: Convert Latitude and longitude coordinates to points that are readable by Post GIS



```
UPDATE landmarks SET
the_geom = ST_GeomFromText('POINT(' || longitude || ' ' || latitude || ')',4326)
```

```
UPDATE 309

Query returned successfully in 70 msec.
```

Step6: Write a Post GIS query to display the nearest 5 locations for the given latitude and longitude



```
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
```

	planar_degrees double precision	name character varying (50)	architect character varying (50)	latitude double precision	longitude double precision
1	.0007463848779255456	Brooks Building	Holabird & Roche	41.87787644	-87.63477822
2	.0013933886958850995	300 West Adams Street Offi...	[null]	41.87972743	-87.63568107
3	.0018793811696624153	Continental And Commercial...	[null]	41.87907898	-87.63301185
4	0.002712009853856664	Chicago Board of Trade Buil...	Holabird & Root	41.87773513	-87.63227115
5	0.00307102762438656	Rookery Building	Burnham & Root	41.87907613	-87.63179743

Implementation of the project using Python:

# Importing the required libraries

import psycopg2

import boto3

from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT

try:

  #connecting to AWS and creating a queue

  #boto3

  sqs = boto3.resource('sqs',aws_access_key_id = 'AKIAWNKTUM4AZYHDYFKQ',

                  aws_secret_access_key = '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')

  queue = sqs.create_queue(QueueName='pwtc-project', Attributes={'DelaySeconds': '5'})

```
    #connecting to postgis
    connection = psycopg2.connect(user="postgres",
                                  password="6162",
                                  host="127.0.0.1")
    connection.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT);
    cursor = connection.cursor()

    #create databse
    cursor.execute("drop database if exists pwtc;")
    create_database = """create database pwtc; """
    cursor.execute(create_database)
    connection.commit()

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

    #creating table
    create_tables_landmarks = """CREATE TABLE if not exists landmarks
(
  gid serial 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()

#insertion of data
insert_data = """ copy landmarks(name,address,date_built,architect,landmark,latitude,longitude) FROM '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':{
```

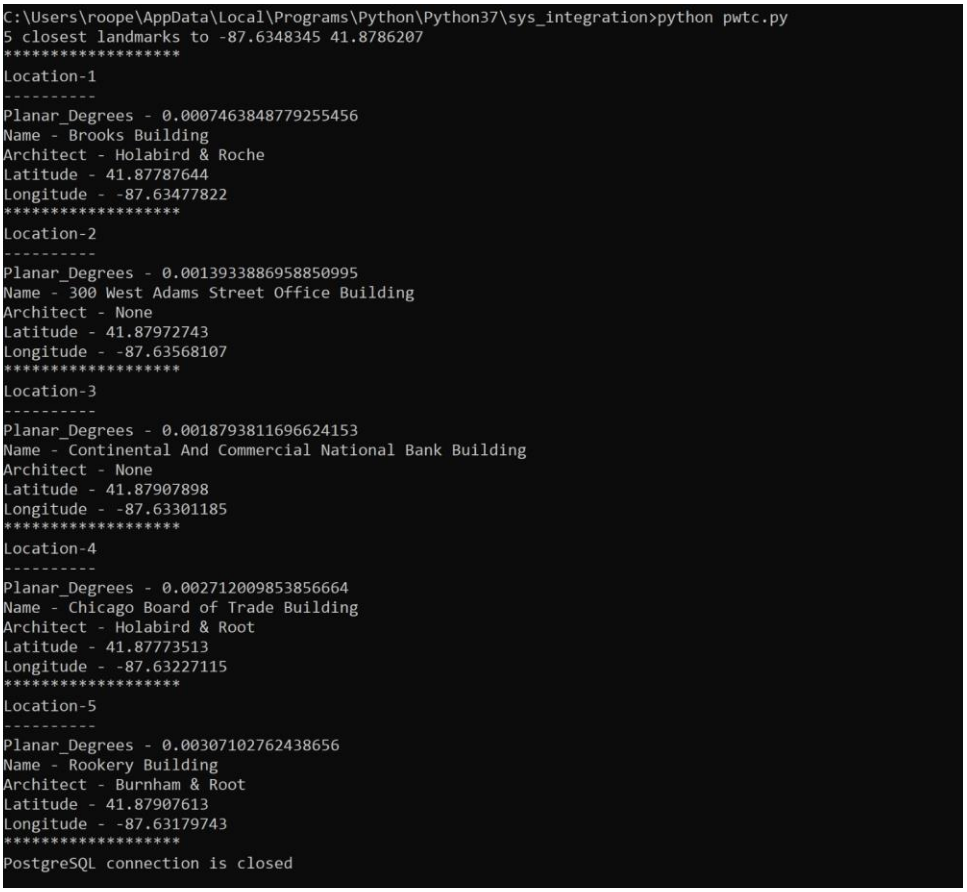
```
'StringValue':'Data Uploaded Successfully!!!',

'DataType':'String'

}})

queue = sqs.get_queue_by_name(QueueName='pwtc-project')


# Converting Latitude and longitude coordinates to points

update_table = """UPDATE landmarks SET the_geom = ST_GeomFromText('POINT(' ||
longitude || ' ' || latitude || ')',4326) """

cursor.execute(update_table)

connection.commit()


 #Displaying nearest locations

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("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]))

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

  #to handle error exceptions
  except (Exception, psycopg2.Error) as error :
    if(connection):
       print(error)

finally:
```

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

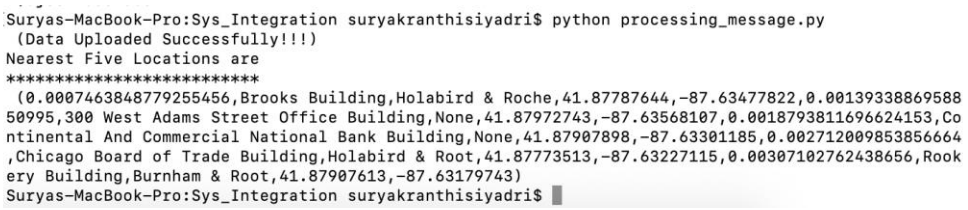
Output:

1. On execution of the above code, below is the output generated in the terminal



```
C:\Users\roope\AppData\Local\Programs\Python\Python37\sys_integration>python pwtc.py
5 closest landmarks to -87.6348345 41.8786207
*******************
Location-1
----------
Planar_Degrees - 0.0007463848779255456
Name - Brooks Building
Architect - Holabird & Roche
Latitude - 41.87787644
Longitude - -87.63477822
*******************
Location-2
----------
Planar_Degrees - 0.0013933886958850995
Name - 300 West Adams Street Office Building
Architect - None
Latitude - 41.87972743
Longitude - -87.63568107
*******************
Location-3
----------
Planar_Degrees - 0.0018793811696624153
Name - Continental And Commercial National Bank Building
Architect - None
Latitude - 41.87907898
Longitude - -87.63301185
*******************
Location-4
----------
Planar_Degrees - 0.002712009853856664
Name - Chicago Board of Trade Building
Architect - Holabird & Root
Latitude - 41.87773513
Longitude - -87.63227115
*******************
Location-5
----------
Planar_Degrees - 0.00307102762438656
Name - Rookery Building
Architect - Burnham & Root
Latitude - 41.87907613
Longitude - -87.63179743
*******************
PostgreSQL connection is closed
```

2. ***The output from the Queue(pwtc-project)***



```
Suryas-MacBook-Pro:Sys_Integration suryakranthisiyadri$ python processing_message.py
 (Data Uploaded Successfully!!!)
Nearest Five Locations are
**************************
 (0.0007463848779255456,Brooks Building,Holabird & Roche,41.87787644,-87.63477822,0.00139338869588
50995,300 West Adams Street Office Building,None,41.87972743,-87.63568107,0.0018793811696624153,Co
ntinental And Commercial National Bank Building,None,41.87907898,-87.63301185,0.002712009853856664
,Chicago Board of Trade Building,Holabird & Root,41.87773513,-87.63227115,0.00307102762438656,Rook
ery Building,Burnham & Root,41.87907613,-87.63179743)
Suryas-MacBook-Pro:Sys_Integration suryakranthisiyadri$ 
```

Challenges during the project execution:

- We attempted to send all the 5 locations to the queue, and we observed that the 5 locations data generated in Tuple and we have converted them into a list to send it into the Q and we had issues doing that. Hence, we had to convert each list into a string to achieve this.