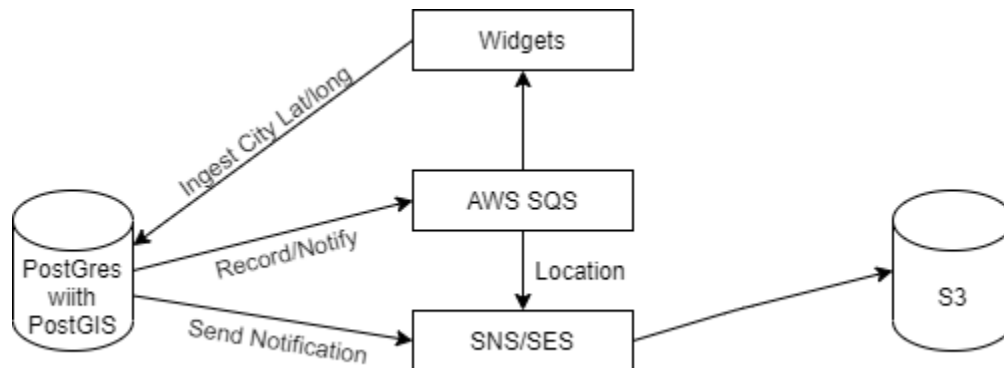


**Final Project  
For  
System Integration Fall 2020  
By  
Umamaheshwari Makkineni Ramnath(UM1008)**

## Project Topic:

For the final project, I have chosen option 2. I have deployed PTWC Widgets as a communication GPS position for field operations. I am implementing database which can find out 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

The screenshot shows the pgAdmin interface. On the left, the 'Servers' tree is expanded to show the 'pwtc' database. The 'Query Editor' is open, displaying the SQL command: `1 create extension if not exists postgis;`. The 'Messages' tab at the bottom shows the output: `CREATE EXTENSION` and a confirmation message: `Query returned successfully in 2 secs 841 msec.`

## Step 2: Create the table in database

The screenshot shows the pgAdmin interface with the 'public' schema selected in the left sidebar. The 'Query Editor' tab is active, displaying the following SQL code:

```
1 CREATE TABLE landmarks
2 (
3     gid serial NOT NULL,
4     name character varying(50),
5     address character varying(50),
6     date_built character varying(10),
7     architect character varying(50),
8     landmark character varying(10),
9     latitude double precision,
10    longitude double precision,
11    the_geom geometry,
12    CONSTRAINT landmarks_pkey PRIMARY KEY (gid),
13    CONSTRAINT enforce_dims_the_geom CHECK (st_ndims(the_geom) = 2),
14    CONSTRAINT enforce_geotype_geom CHECK (geometrytype(the_geom) = 'POINT'::text OR the_geom IS NULL),
15    CONSTRAINT enforce_srid_the_geom CHECK (st_srid(the_geom) = 4326)
16 );
```

The 'Messages' tab at the bottom shows the execution result: 'CREATE TABLE' and 'Query returned successfully in 52 msec.'

## Step 3: Create Index

The screenshot shows the pgAdmin interface with the 'landmarks' table selected in the left sidebar. The 'Query Editor' tab is active, displaying the following SQL code:

```
1 CREATE INDEX landmarks_the_geom_gist
2 ON landmarks
3 USING gist
4 (the_geom );
```

The 'Messages' tab at the bottom shows the execution result: 'CREATE INDEX' and 'Query returned successfully in 38 msec.'

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

The screenshot shows the PgAdmin interface with the 'Query Editor' tab active. The left sidebar displays the database structure, including 'Servers (2)', 'PostgreSQL', 'PostgreSQL 13', 'Databases (5)', and 'postgres'. The main query editor contains the following SQL command:

```
1 COPY landmarks(name,gid,address,date_built,architect,landmark,latitude,longitude)
2 FROM 'C:\Users\sunil\Downloads\123.csv' DELIMITERS ',' CSV HEADER;
3
```

The 'Messages' tab at the bottom shows the execution results:

```
COPY 317
Query returned successfully in 44 msec.
```

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

The screenshot shows the PgAdmin interface with the 'Query Editor' tab active. The left sidebar displays the database structure, including 'Servers (2)', 'PostgreSQL', 'PostgreSQL 13', 'Databases (5)', and 'postgres'. The main query editor contains the following SQL command:

```
1 UPDATE landmarks
2 SET the_geom = ST_GeomFromText('POINT(' || longitude || ' ' || latitude || ')',4326);
3
```

The 'Messages' tab at the bottom shows the execution results:

```
UPDATE 317
Query returned successfully in 42 msec.
```

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

The screenshot shows the PgAdmin interface with a PostgreSQL database named 'postgres' selected. The 'Query Editor' tab is active, displaying the following SQL query:

```
1 SELECT
2 ST_Distance(ST_GeomFromText('POINT(-87.6348345 41.8786287)', 4326), landmarks.the_geom) AS planar_degrees
3 name,
4 architect
5 FROM landmarks
6 ORDER BY planar_degrees ASC
7 LIMIT 5;
```

The 'Data Output' tab shows the results of the query, displaying 5 rows of data. The columns are 'planar\_degrees', 'name', and 'architect'.

planar_degrees	name	architect
0.0007463848779255456	Brooks Building	Holabird & Roche
0.0013933886958850995	300 West Adams Street Office Building	[null]
0.0018793811696624153	Continental And Commercial National Bank Building	[null]
0.002712009853856664	Chicago Board of Trade Building	Holabird & Root
0.00307102762438656	Rookery Building	Burnham & Root

## python implementation:

■ Anaconda Prompt (anaconda3)

```
(base) C:\Users\sunil>python 1.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

(base) C:\Users\sunil>
```