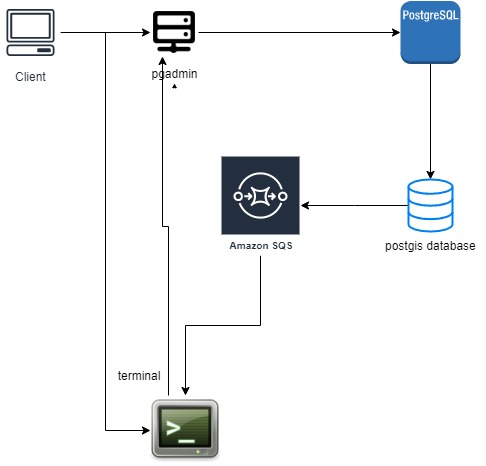
Draw.io:



CREATE TABLE 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)

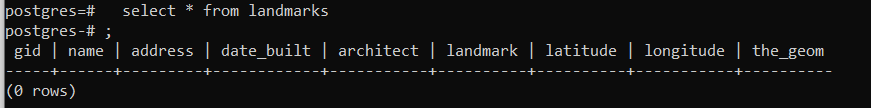
);

CREATE INDEX landmarks\_the\_geom\_gist

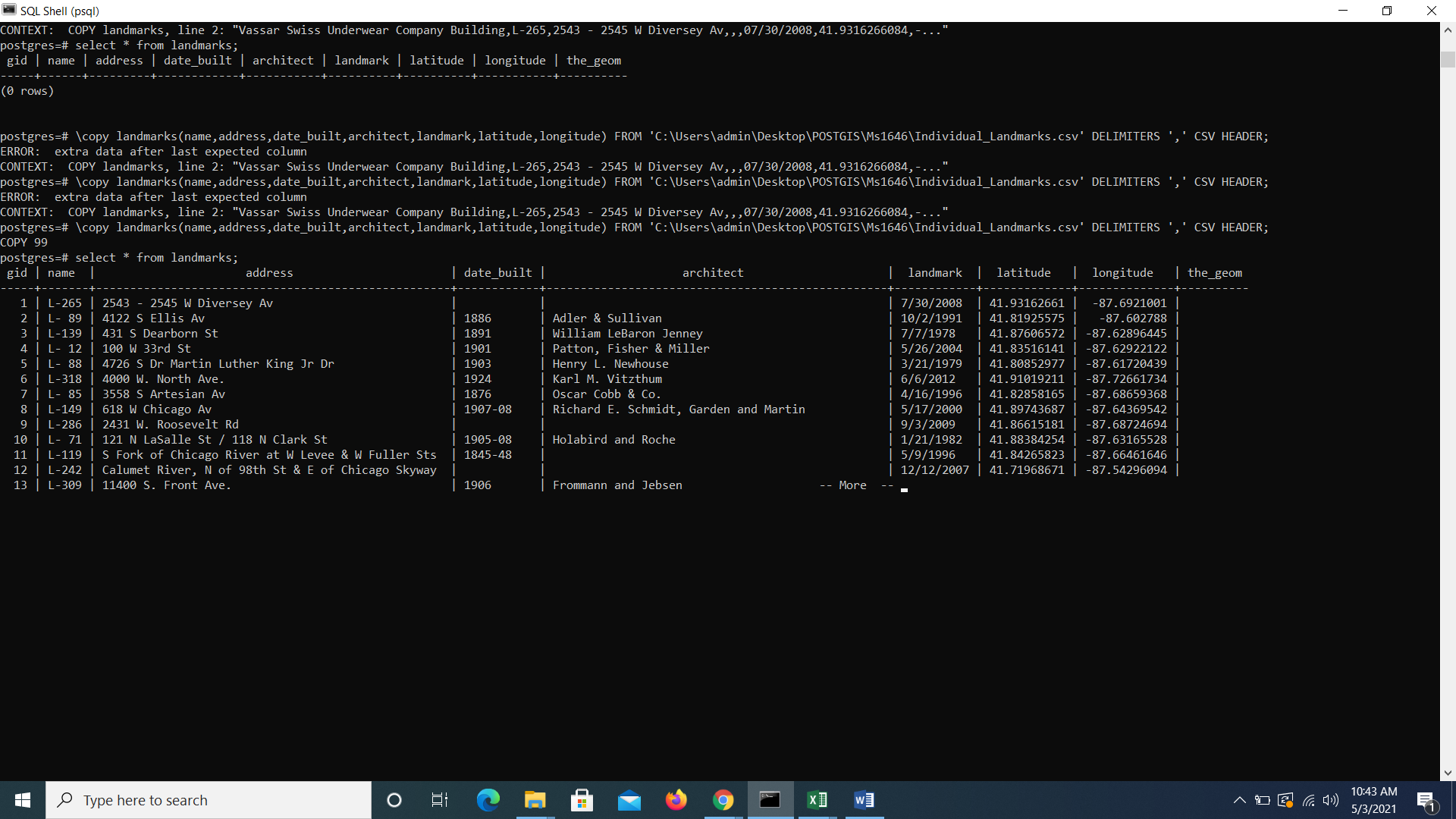
ON landmarks

USING gist

(the\_geom );

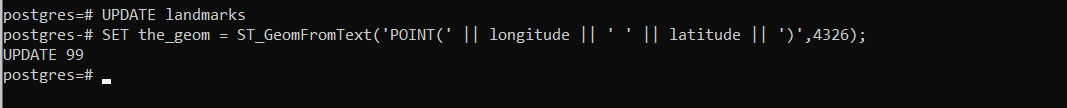


\copy landmarks(name,address,date\_built,architect,landmark,latitude,longitude) FROM 'C:\Users\admin\Desktop\POSTGIS\Ms1646\Individual\_Landmarks.csv' DELIMITERS ',' CSV HEADER;



UPDATE landmarks

SET the\_geom = ST\_GeomFromText('POINT(' || longitude || ' ' || latitude || ')',4326);



SELECT distinct

ST\_Distance(ST\_GeomFromText('POINT(-87.6348345 41.8786207)', 4326), landmarks.the\_geom) AS planar\_degrees,

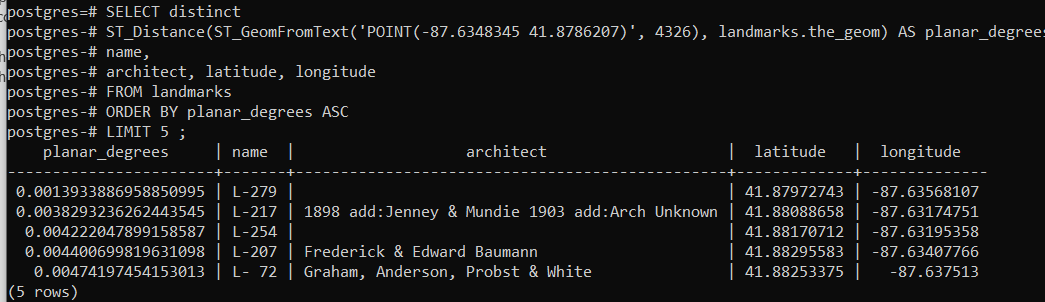
name,

architect, latitude, longitude

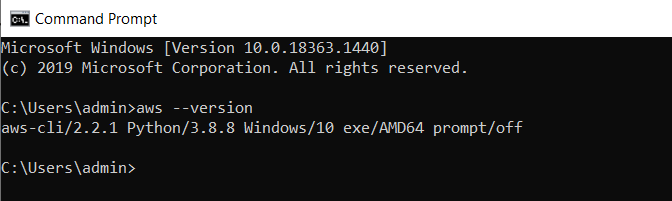
FROM landmarks

ORDER BY planar\_degrees ASC

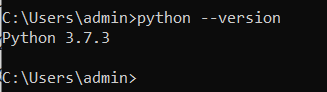
LIMIT 5;



AWS Version check:



Python version:



Pwtc.py:

import psycopg2

import boto3

from psycopg2.extensions import ISOLATION\_LEVEL\_AUTOCOMMIT

try:

#boto3

sqs = boto3.resource('sqs',aws\_access\_key\_id = 'AKIAWNKTUM4AQA7BEMV6',

aws\_secret\_access\_key = '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')

queue = sqs.create\_queue(QueueName='ms1646', Attributes={'DelaySeconds': '5'})

#connecting to postgis

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

password="CT13root",

host="127.0.0.1")

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

cursor = connection.cursor()

#create extension postgis

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

cursor.execute(create\_extension\_query)

connection.commit()

#create tables and indexes

create\_tables\_landmarks = """ CREATE TABLE if not exists landmarks

(

gid character varying(50) 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\_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,gid,address,date\_built,architect,landmark,latitude,longitude) FROM '/home/administrator/Desktop/si/project/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')

#updation of table for POINT

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

cursor.execute(update\_table)

connection.commit()

#Display near 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()

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")

GIT Hub link : https://github.com/ms1646/MS1646Project.git