TODB -CSE532 _ASSIGNMENT 3 : SPATIAL QUERIES

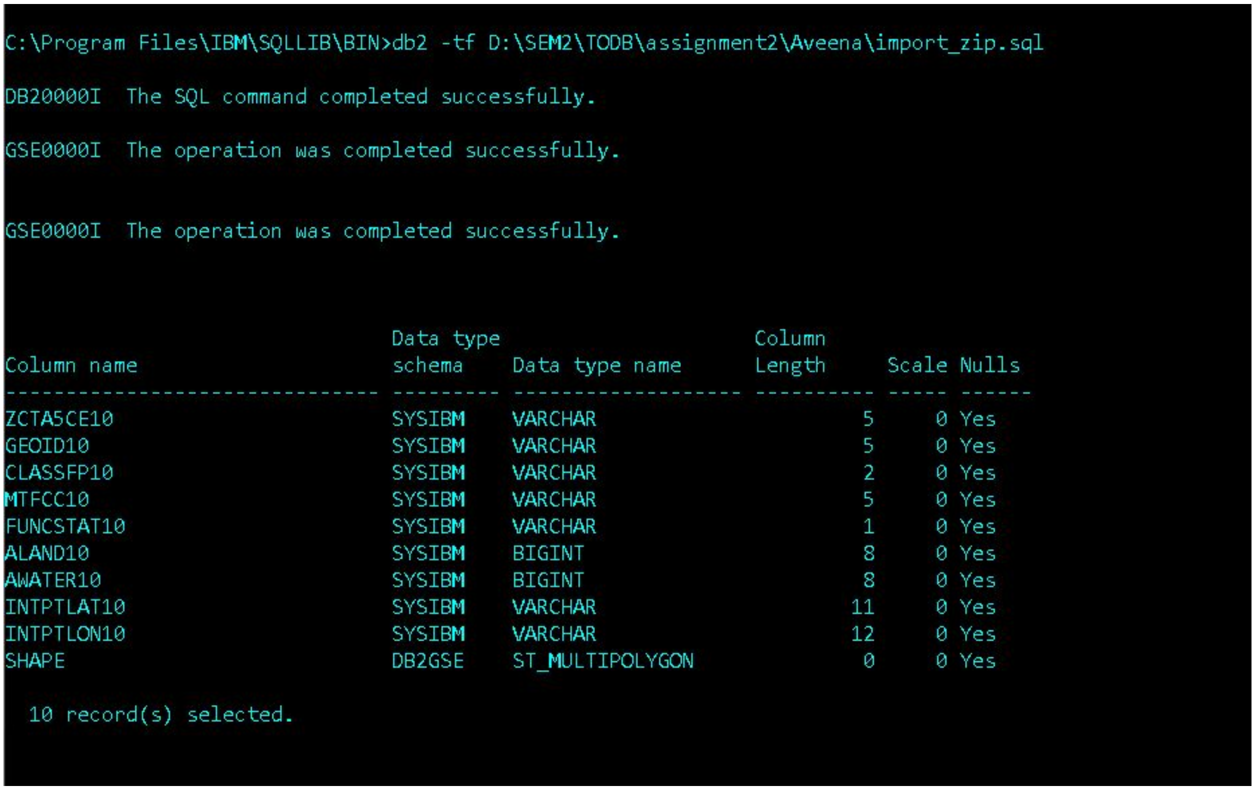
Question 1.

- a. Enable the sample database for spatial support:

→ db2se enable_db sample

- b. Load the zip code area dataset using the import SQL file:

→ db2 -tf import_zip.sql



```
C:\Program Files\IBM\SQLLIB\BIN>db2 -tf D:\SEM2\TODB\assignment2\Aveena\import_zip.sql

DB20000I  The SQL command completed successfully.

GSE0000I  The operation was completed successfully.


GSE0000I  The operation was completed successfully.



                                Data type                     Column
Column name                     schema    Data type name      Length     Scale Nulls
------------------------------- --------- ------------------- ---------- ----- ------
ZCTA5CE10                       SYSIBM    VARCHAR                      5     0 Yes
GEOID10                         SYSIBM    VARCHAR                      5     0 Yes
CLASSFP10                       SYSIBM    VARCHAR                      2     0 Yes
MTFCC10                         SYSIBM    VARCHAR                      5     0 Yes
FUNCSTAT10                      SYSIBM    VARCHAR                      1     0 Yes
ALAND10                         SYSIBM    BIGINT                       8     0 Yes
AWATER10                        SYSIBM    BIGINT                       8     0 Yes
INTPTLAT10                      SYSIBM    VARCHAR                     11     0 Yes
INTPTLON10                      SYSIBM    VARCHAR                     12     0 Yes
SHAPE                           DB2GSE    ST_MULTIPOLYGON              0     0 Yes

  10 record(s) selected.
```

- c. Create two tables for facilities using the createfacilititytable.sql (we create two tables, cse532.facilityoriginal for original data, and cse532.facility with a spatial column).

→ db2 -tf createfacilititytable.sql

- d. Load Health_Facility_General_Information.csv into cse532.facilityoriginal using script:

→ db2 load from "C:\myfolder\Health_Facility_General_Information.csv" of del MESSAGES load.msg INSERT INTO cse532.facilityoriginal

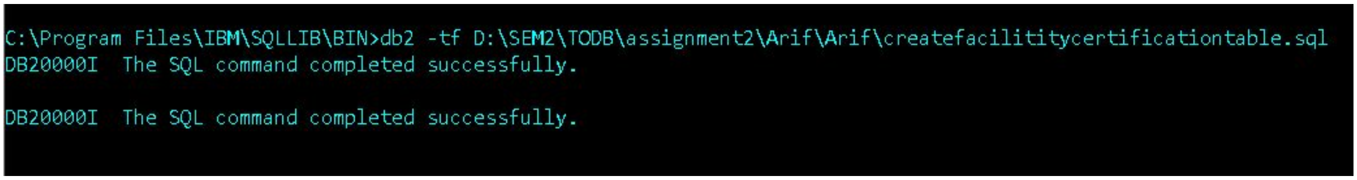
- e. Write a SQL script **facilityinsert.sql** to insert data into cse532.facility by selecting data from cse532.facilityoriginal table and converting (*Latitude, Longitude*) attributes into DB2GSE.ST_POINT type with srs_id 1 for *geolocation* attribute in cse532.facility.

f. Create a SQL script **createfacilititycertificationtable.sql** to create a table:

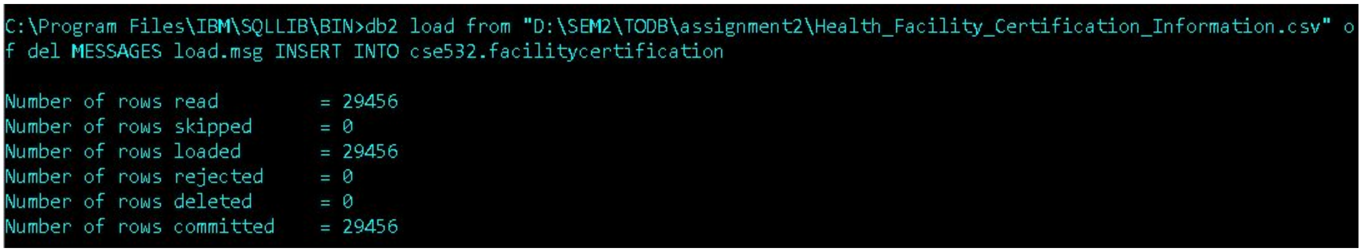
cse532.facilitycertification (FacilityID, FacilityName, Description, AttributeType, AttributeValue, MeasureValue, County)

and load the csv file into the table:

db2 load from "C:\yourpath\Health_Facility_Certification_Information.csv" of del MESSAGES load.msg INSERT INTO cse532.facilitycertification



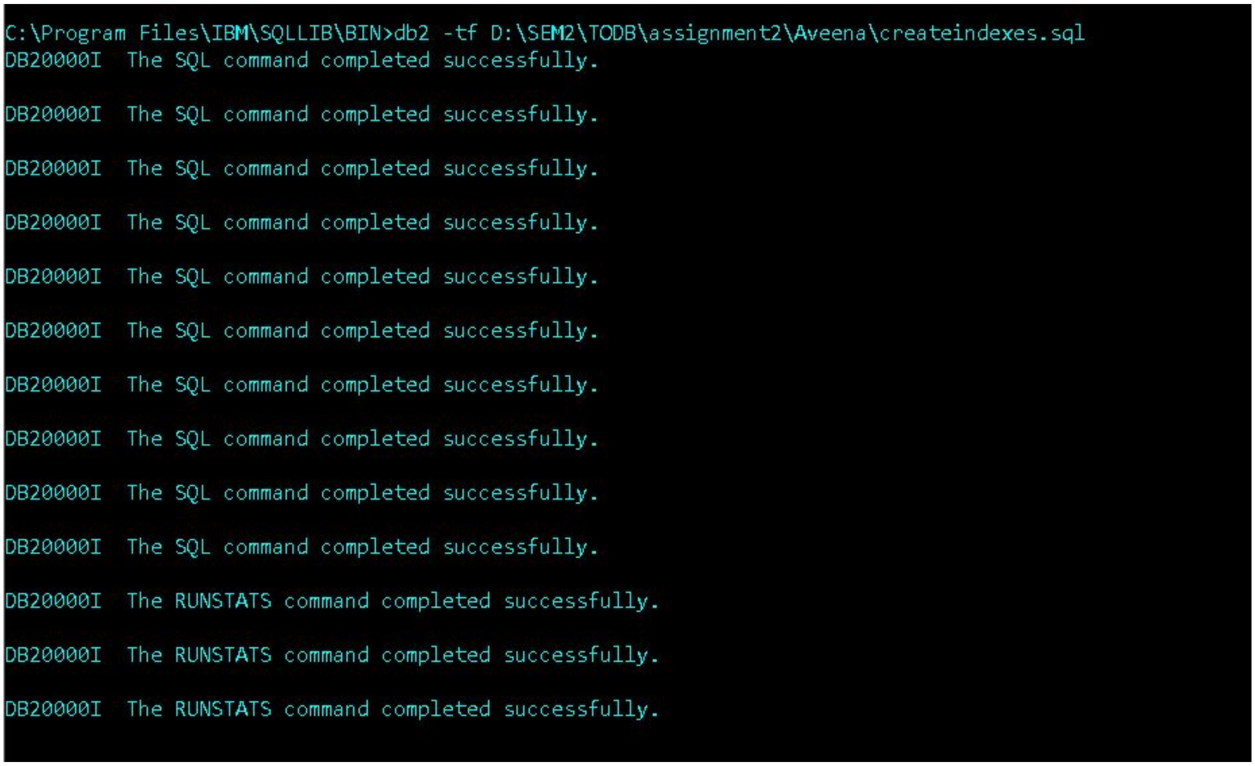
```
C:\Program Files\IBM\SQLLIB\BIN>db2 -tf D:\SEM2\TODB\assignment2\Arif\Arif\createfacilititycertificationtable.sql
DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.
```



```
C:\Program Files\IBM\SQLLIB\BIN>db2 load from "D:\SEM2\TODB\assignment2\Health_Facility_Certification_Information.csv" o
f del MESSAGES load.msg INSERT INTO cse532.facilitycertification

Number of rows read         = 29456
Number of rows skipped      = 0
Number of rows loaded       = 29456
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 29456
```

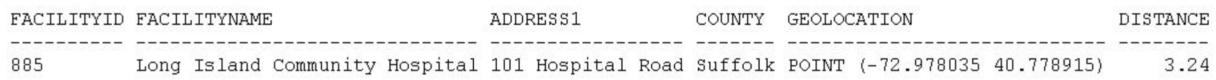
g. Update the createindexes.sql to add additional indexes besides spatial indexes for the queries below.

→ db2 -tf createindexes.sql

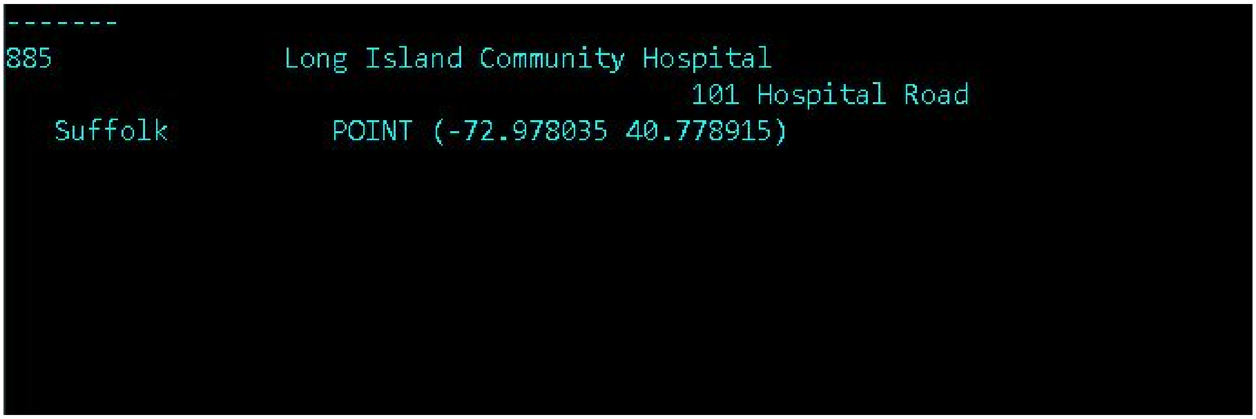


```
C:\Program Files\IBM\SQLLIB\BIN>db2 -tf D:\SEM2\TODB\assignment2\Aveena\createindexes.sql
DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The SQL command completed successfully.

DB20000I  The RUNSTATS command completed successfully.

DB20000I  The RUNSTATS command completed successfully.

DB20000I  The RUNSTATS command completed successfully.
```

Question 2. **Write a query nearester.sql to find closest healthcare facility with an ER room (AttributeValue = 'Emergency Department') from "2799 Horseblock Road Medford, NY 11763"(40.824369, -72.993983) (latitude, longitude)**. Please return location and distance in your result. You can use unit 'KILOMETER', 'METER', or 'STATUTE MILE' for distance measurement.
Nearest neighbor search is not directed supported by DB2. You can use ST_BUFFER to create a buffered area (polygon/circle) from a point within a certain distance and search only stores within the buffer. Note that 0.25 degree is roughly 10 miles. For all the datasets, we use spatial reference nad83_srs_1 with srs ID as 1.
You can find information here on functions such as ST_POINT, ST_BUFFER, ST_WITHIN or ST_CONTAINS, and ST_DISTANCE.



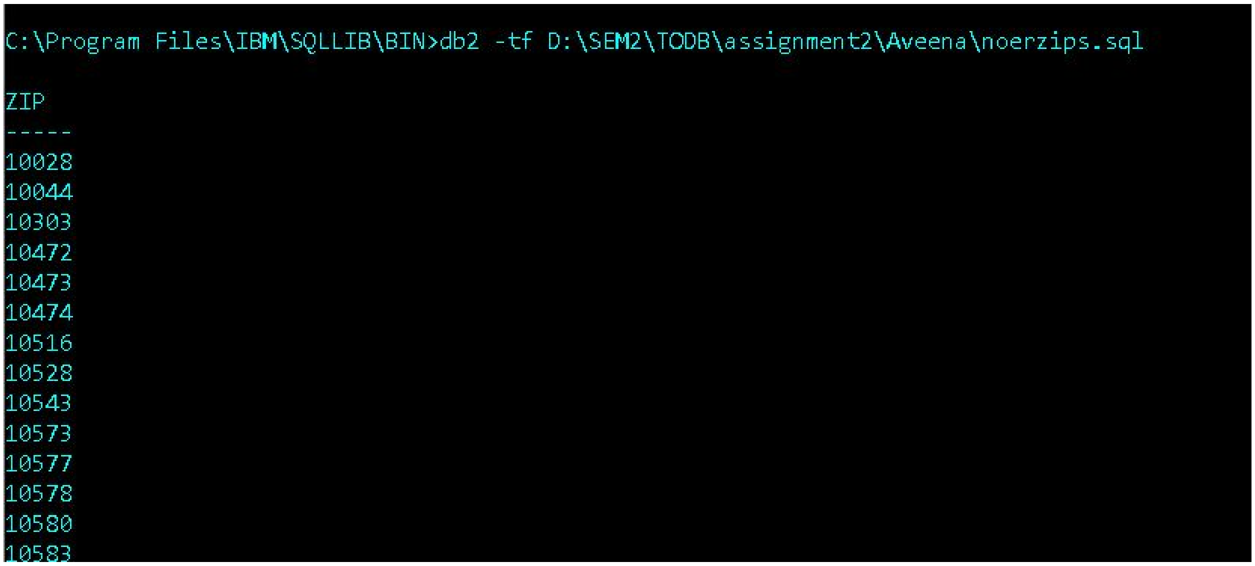
```
FACILITYID FACILITYNAME                   ADDRESS1          COUNTY  GEOLOCATION                  DISTANCE
---------- ------------------------------ ----------------- ------- ---------------------------- --------
885        Long Island Community Hospital 101 Hospital Road Suffolk POINT (-72.978035 40.778915)     3.24
```



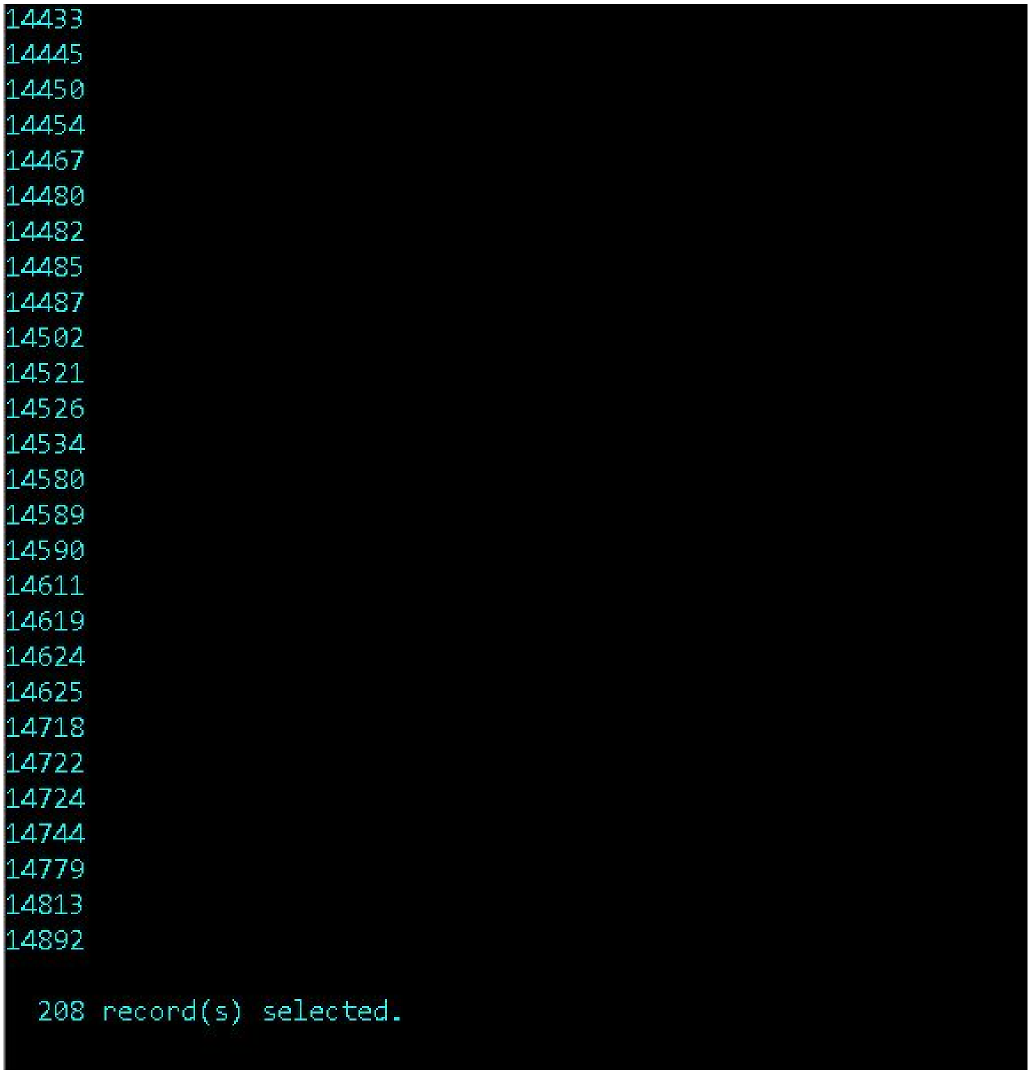
```
-------
885              Long Island Community Hospital
                                         101 Hospital Road
   Suffolk          POINT (-72.978035 40.778915)
```

Question 3. EXPLANATION

1. **Here we are ignoring all the zipcodes that are not present in both uszip table and facility table. Even though there might zipcodes with no shape and neither a neighbour of ER department or ER department itself, we have not considered it**
2. **We fetch all the zipcodes first. Then fetch all the ER-Department-zip codes union neighbours of ER-Department zipcodes**
3. **Subtract 2nd from 1st and that would be our output**



```
C:\Program Files\IBM\SQLLIB\BIN>db2 -tf D:\SEM2\TODB\assignment2\Aveena\noerzips.sql

ZIP
-----
10028
10044
10303
10472
10473
10474
10516
10528
10543
10573
10577
10578
10580
10583
```



```
14433
14445
14450
14454
14467
14480
14482
14485
14487
14502
14521
14526
14534
14580
14589
14590
14611
14619
14624
14625
14718
14722
14724
14744
14779
14813
14892

  208 record(s) selected.
```

Question 4. (1 point) **Drop all indexes and perform the two queries again, and compare the query performance in terms of execution time for above two queries.**
Show your time difference with and without indexes in your **README** file.

Query No.	**Procedure SQL statement**	**Before indexes**	**After indexes**	**Before (timestamp units)**	**After(timestamp units)**
1	nearester.sql	1sec	1sec	1907000	1854000
2	noerzips.sql	8sec	7sec	23427000	20034270
3	mergezip.sql	-	50mins	-	-

Question 5. Write SQL queries or stored procedure mergezip.sql to merge zip code areas into large ones with neighboring zip code areas, so that the new population in each merged region (combined zipcodes) is large than the current average population, using the zip code population table in Homework1. For simplicity, you can remove the duplicates from the population table.

Fetch ZIPCODE, ZIPPOP from ZIPPOP table and GEOID10, SHAPE from USZIP table for this procedure. Find the average population from ZIPPOP. Find all the zipcodes which have ZIPcodes in both the tables. Find the neighbouring zipcode for each zipcode and find their sum of population. If this sum is greater than average population, then create a new cluster id. All the unique cluster id would give the total number of distinct regions after the merge.