Database for Community Resource Accessibility of Older Adult Homes

SA8902: Assignment 2

Ellie Maclennan

Assignment 2: Creating a Spatial Database

a. **Definition of the research questions**

My research interests are related to the wellbeing of older adults (those 65 years and older) in Canada, which based on my prior research experience, varies considerably across regions and more local communities. As mobility and transportation are a key consideration for access and quality of life for older adults, I hope to create a database to help assess how well-resourced and accessible neighborhoods are in Toronto in terms of meeting the basic needs of older adults. Further, I want to explore how living in one of the 10 Long-Term Care (LTC) homes operated by the City of Toronto may compare to living in one of the Retirement Homes located in Toronto. From an equity and access standpoint, comparing these two different housing options for older adults is also of great interest. Therefore, I will be looking at what resources are in each neighborhood, as well as how far residences are from resources in their neighborhood, and if these two different types of homes are close to each other. The following five research questions could be answered using this database:

1) How many hospitals are there in each neighbourhood? Which neighborhoods have no hospitals?
2) Which neighborhoods have the most LTCS and Retirement Homes? Which have the least?
3) Are grocery stores located in close proximity to LTC homes? Which LTC homes have the nearest grocery stores?
4) How much green space (i.e., park space) is there in each neighborhood?
5) How far are LTCS from retirement homes? Do they seem to cluster together? Are there any in near proximity to each other?

b. **Stages of data modeling and database design**

First, I sketched rudimentary ER diagrams to understand what type of entities I wanted to have to answer my research questions, and what data I would need to find. After multiple iterations of this, I was able to locate data that was publicly available.

After looking into my data, I looked at my datafiles to ensure they all fell under the same theme, and would therefore be in the same table. I also looked to identify Candidate keys, which are those columns that are eligible to be Primary keys (unique identifiers). Importantly, this stage also involved searching for functional dependencies (relationships between attributes where an attribute(s) determines the value of another). Here, I identified no functional dependencies that would require further subdivision of my tables. For the purpose of this assignment, there are some attributes that have repeating values (e.g., values for “City” are all “Toronto”), but I wanted to keep this to show how this database could be scaled up and expanded beyond the boundaries of Toronto. There were also some additional descriptive attributes that were not used in answering my research questions for this assignment, but I saw as pertinent to other related questions. Once again, I returned to my ER diagram to plan tables and ensure

relationship and cardinality made sense. Six tables still seemed optimal, and thus I moved onto implementing the database.

I ended up creating a database with the following six entities:

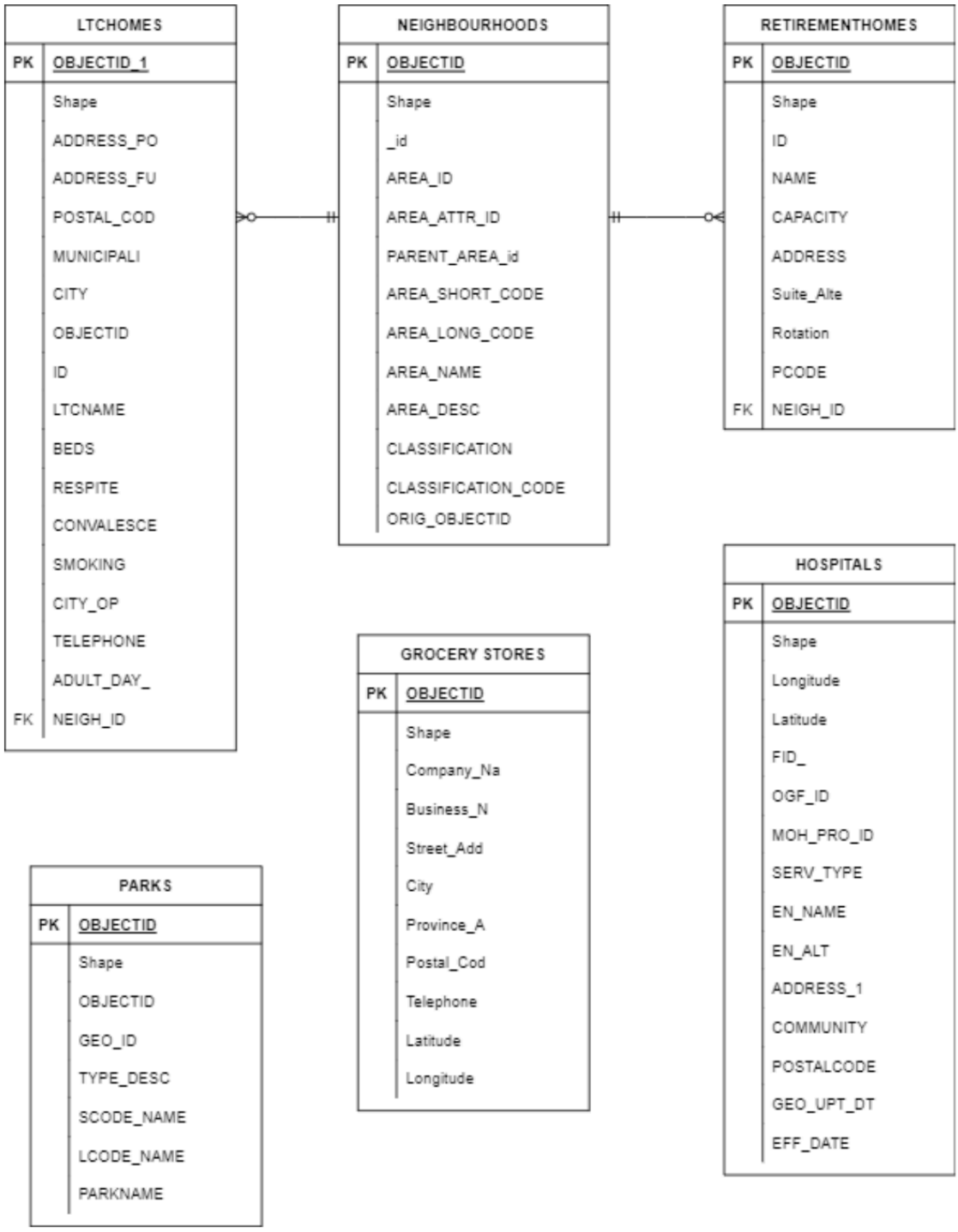
- NEIGHBORHOODS
- LONG-TERM CARE (LTC) HOMES
- RETIREMENT HOMES
- GROCERY STORES
- HOSPITALS
- PARKS

All entity data came from separate data files. Some were already in Shapefile format, and some I converted to Shapefiles using ARCGIS (purely for simplicity and consistency as a designer) and imported the files to SQL server using ARC Catalog. All data was imported using a consistent SRID (32617) and spatial data as the geometry data type (explained why this form below). After this, I used queries (describe below) to set my NEIGHBOURHOOD foreign keys for the LTCHOMES and RETIREMENTHOMES tables, and I set my spatial index for my GROCERYSTORES table, as this seemed to be one of the tables that would be most benefited by having a spatial index.

c. **ER diagram and discussion describing the data model**

As can be seen in the ER diagram below, my database made use of six tables. NEIGHBORHOODS, LTCHOMES, RETIREMENTHOMES, GROCERY STORES, HOSPITALS, and PARKS.

Here, I conceptualize NEIGHBOURHOODS uniquely, as they are the communitary boundaries that contain other tables. However, because of the use of spatial queries, the other tables do not need to have defined relationships. Because my research questions are focused around Retirement homes and LTC homes, and specifically how they are positioned in communities, I did set relationships with these entities and neighborhoods, where each RETIREMENTHOME or LTCHOME must belong to a NEIGHBOURHOOD, and each NEIGHBOURHOOD may have none, one, or multiple LTCHOMES and/or RETIREMENTHOMES. For all other resources, such as PARKS, GROCERYSTORES, and HOSPITALS, I did not conceptualize these as belonging to one NEIGHBOURHOOD per se, as these resources may be shared among NEIGHBOURHOODS, and in the case of PARKS, may even span the boundary lines of multiple NEIGHBOURHOODS. Because of these, I did not think it made sense conceptually to define set relationships between NEIGHBOURHOODS and these resources, and instead will make use of spatial queries to understand resources within the boundaries of NEIGHBOURHOODS, as well as explore questions related to distances from LTCHOMES OR RETIREMENTHOMES to resources.



d. A **data dictionary**

NEIGHBOURHOODS (OBJECT_ID, Shape, _id, AREA_ID, AREA_ATTR_ID, PARENT_AREA_ID, AREA_SHORT_CODE, AREA_LONG_CODE, AREA_NAME, AREA_DESC, CLASSIFICATION, CLASSIFICATION_CODE, ORIG_OBJECTID)

- A neighbourhood instance includes all official neighbourhoods in the City of Toronto.

Column Name	**Data Type (Length)**	**Key**	**Null Status**	**Default Value**	**Remarks**	**Description**
OBJECT_ID	Integer	Primary Key	Not Null	DBMS supplied	Surrogate Key; Initial value=1; Increment= 1	Unique identifier
Shape	Geometry	No	Not Null	Geometry	None	Holds spatial information
_id	Numeric(18,0)	No	Null	None		System identifier
AREA_ID	Numeric(23,15)	No	Null	None		Area identifier
AREA_ATTR _ID	Numeric(18,0)	No	Null	None		Attribute identifier
PARENT_AR EA_ID	Numeric(18,0)	No	Null	None		Parent area identifier
AREA_SHO RT_CODE	Numeric(18,0)	No	Null	None		Short name of neighborho od
AREA_LON G_CODE	Numeric(18,0)	No	Null	None		Condensed name of area
AREA_NAM E	Numeric(80)	No	Null	None		Full name of area
AREA_DES C	Numeric(80)	No	Null	None		Area name with region number

CLASSIFICATION	Numeric(80)	No	Null	None		Classification of specified area
ORIG_OBJECTID	Numeric(18, 0)	No	Null	None		Original ID from file

LTCHOMES (OBJECTID_1, Shape, ADDRESS_PO, ADDRESS_FU, POSTAL_COD, MUNICIPALITY, CITY, OBJECTID, ID, LTCNAME, BEDS, RESPITE, CONVALESCE, SMOKING, CITY_OP, TELEPHONE, ADULT_DAY, NEIGH_ID)

- An LTCHOMES instance is any one of the 10 City-operated LTC Homes in Toronto.

ColumnName	**Data Type (Length)**	**Key**	**Null Status**	**Default Value**	**Remarks**	**Description**
OBJECTID_1	Integer	Primary Key	Not Null	DBMS supplied	Surrogate Key; Initial value=1; Increment=1	Unique identifier
Shape	Geometry	No	Not Null	Geometry		Holds spatial information
ADDRESS_PO	Integer	No	Null	None		Feature ID
ADDRESS_FU	VarChar(131)	No	Null	None		Feature type
POSTAL_COD	VarChar(7)	No	Null	None		Postal code
MUNICIPALITY	VarChar(40)	No	Null	None		City of Toronto Municipality
CITY	VarChar(7)	No	Null	None		Name of City
LTCNAME	VarChar(254)	No	Null	None		Home name

BEDS	Integer	No	Null	None		Number of beds
RESPITE	Integer	No	Null	None		Number of respite beds
SMOKING	BOOL (or bit)	No	Null	False	False for No, True for Yes	If smoking rooms available
CITY_OP	BOOL (or bit)	No	Null	False	False for No, True for Yes	City operated
TELEPHONE	VarChar(30)	No	Null	None		Phone number
ADULT_DAY	BOOL (or bit)	No	Null	False	False for No, True for Yes	If adult care available
NEIGH_ID	Integer	Foreign Key	Not Null	None	REF: NEIGHBOU RHOODS	Foreign Key to NEIGHBO URHOODS

RETIREMENTHOMES (OBJECTID, Shape, ID, HOMENAME, CAPACITY, ADDRESS, SUITE_ALT, Rotation, PCODE, NEIGH_ID)

- A RETIREMENTHOMES instance is any of the listed retirement homes in Toronto.

Column Name	**Data Type (Length)**	**Key**	**Null Status**	**Default Value**	**Remarks**	**Descriptio n**
OBJECTID	Integer	Primary Key	Not Null	DBMS supplied	Surrogate Key; Initial value=1; Increment= 1	Unique identifier
Shape	Geometry	No	Not Null	Geometry	None	Holds spatial information
ID	VarChar(1 0)	No	Null	None		System identifier
HOMENAME	VarChar(2	No	Null	None		Name of

	54)					home
CAPACITY	Integer	No	Null	None		Number of beds
ADDRESS	VarChar(50)	No	Null	None		Address of home
SUITE_ALT	Integer	No	Null	None		Number of suites
Rotation	Integer	No	Null	None		Types of care
PCODE	VarChar(50)	No	Null	None		Postal Code
NEIGH_ID	Integer	Foreign Key	Not Null	None	REF: NEIGHBOURHOODS	Foreign Key to NEIGHBOURHOODS

GROCERYSTORES (OBJECTID, Shape, Company_Na, Business_N, Street_Add, City, Province_A, Postal_Cod, Telephone, Latitude, Longitude)

- A grocery store instance is any one that is a grocery store in Toronto boundaries listed on SimplyAnalytics.

Column Name	**Data Type (Length)**	**Key**	**Null Status**	**Default Value**	**Remarks**	**Description**
OBJECTID	Integer	Primary Key	Not Null	DBMS supplied	Surrogate Key; Initial value=1; Increment=1	Unique identifier
Shape	Geometry	No	Not Null	Geometry	None	Holds spatial information
Company_Na	VarChar(254)	No	Null	None		Name of owner organization
Business_N	VarChar(254)	No	Null	None		Name of business

Street_Add	VarChar(254)	No	Null	None		Street Address
City	VarChar(50)	No	Null	None		City Name
Province_A	VarChar(20)	No	Null	None		Province Name
Postal_Cod	VarChar(8)	No	Null	None		Postal Code
Telephone	Numeric(38,8)	No	Null	None		Phone number
Latitude	Numeric(38,8)	No	Null	None		Latitude
Longitude	Numeric(38,8)	No	Null	None		Longitude

HOSPITALS (OBJECTID, Shape Longitude, Latitude, FID_, OGF_ID, MOH_PRO_ID, SERV_TYPE, EN_NAME, EN_ALT, ADDRESS_1, COMMUNITY, POSTALCODE, GEO_UPT_DT, EFF_DATE)

- A HOSPITALS instance is one of the official general hospitals listed by Ontario Open Data in the city of Toronto.

Column Name	**Data Type (Length)**	**Key**	**Null Status**	**Default Value**	**Remarks**	**Description**
OBJECTID	Integer	Primary Key	Not Null	DBMS supplied	Surrogate Key; Initial value=1; Increment=1	Unique identifier
Shape	Geometry	No	Not Null	Geometry	None	Holds spatial information
Longitude	Numeric(38,8)	No	Null	None		Longitude
Latitude	Numeric(38,8)	No	Null	None		Latitude
FID_	Integer	No	Null	None		

OGF_ID	Integer	No	Null	None		Ontario Geospatial Feature identifier (OGF ID).
MOH_PRO_ID	Numeric(38,8)	No	Null	None		Ministry of Health service provider identifier
SERV_TYPE	VarChar(254)	No	Null	None		Type of health services provided
EN_NAME	VarChar(254)	No	Null	None		Name of hospital in English
EN_ALT	VarChar(254)	No	Null	None		Alternative name to link with operator
ADDRESS_1	VarChar(254)	No	Null	None		Address
COMMUNITY	VarChar(254)	No	Null	None		Identifies census subdivision or municipality
POSTALCODE	VarChar(8)	No	Null	None		Postal Code
GEO_UPT_DT	Numeric(38,8)	No	Null	None		Time data set was last updated for LIO editor.
EFF_DATE	Numeric(38,8)	No	Null	None		Date record was created.

PARKS (OBJECTID, Shape Longitude, Latitude, FID_, OGF_ID, MOH_PRO_ID, SERV_TYPE, EN_NAME, EN_ALT, ADDRESS_1, COMMUNITY, POSTALCODE, GEO_UPT_DT, EFF_DATE,)

- A PARKS instance is one of thel parks listed by Ontario Open Data in the city of Toronto.

Column Name	**Data Type (Length)**	**Key**	**Null Status**	**Default Value**	**Remarks**	**Description**
OBJECTID_1	Integer	Primary Key	Not Null	DBMS supplied	Surrogate Key; Initial value=1; Increment= 1	Unique identifier
Shape	Geometry	No	Not Null	Geometry	None	Holds spatial information
GEO_ID	Numeric(38,8)	No	Null	None		Geographi c identifier
TYPE_DESC	VarChar(30)	No	Null	None		Area type
SCODE_NA ME	VarChar(10)	No	Null	None		Area short code
LCODE_NAM E	VarChar(20)	No	Null	None		Park number
PARKNAME	VarChar(200)	No	Null	None		Park name

Relationships

Parent	**Child**	**Referential Integrity Constraint**	**On Update**	**On Delete**
NEIGHBOURHO ODS	RETIREME NTHOMES	NEIGHID in RETIREMENTHOME S must exist in OBJECTID in NEIGHBOURHOODS	Yes	No
NEIGHBOURHO ODS	LTCHOMES	NEIGHID in LTCHOMES must	Yes	No

		exist in OBJECTID in NEIGHBOURHOODS		

e. **Spatial data type justification**
I opted to create a very spatial database, in order to better answer my spatial research questions. All entities made use of spatial data, as I wanted to understand the location of resources, homes for older adults, and neighbourhoods relative to each other. I used the geometry data type (two-dimensional) to store my projected spatial data (see a more detailed about geometric versus geographic data below). I used the WGS 84 / UTM zone 17N (SRID: 32617) for my spatial reference systems in order to make meaning out of my coordinate pairs. This is the standard system for the Toronto region.

I chose the geometry data type for a few reasons, but for this database, processing performance was the most important consideration in making this selection. Because this database is designed to be used to approximate spatial relationships, and not serve as an exact measure of space, minor inaccuracies in representations are permissible. For example, I am using points for representing many buildings (e.g., hospitals) when in the real world these are better represented by polygons. Likewise, using a projection to represent space comes with distortion, but because of the purpose of the database, efficient querying is more important to optimize than precise accuracy. The geometric data type requires less processing power and time to run these queries. Thus, apart from the need for consistency in spatial data types for spatial queries, was the most important factor in making this decision. I highlight other differences in spatial data types in the following question.

More specifically, I used polygons (connected sequence of coordinate pairs) for my NEIGHBOURHOODS and PARKS tables, and points for my RETIREMENTHOMES, LTCHOMES, GROCERYSTORES, AND HOSPITALS tables. This was because my NEIGHBOURHOODS represent real-world areas, and need to be able to contain other features within their boundaries. Similarly, the PARKS table represents green space areal data, so I wanted to be able to capture these areas, which could be approximated using polygons.
For the other tables, I simply wanted to capture their approximate locations, and the size of these buildings was not relevant to my research questions, so point representations were suitable and efficient to store this information.

f. **More on the general differences between the geography and geometry data types**
The geography and geometry data types are the two spatial data types offered by SQL server, and they help connect and represent the world as various geometries. These data types both require selection of a coordinate system and datum (representation of size and shape of the Earth) in order to consistently tie locations to places. Likewise, these spatial reference systems must account for units, where the Prime Meridian is drawn, and when needed, which projection will be used.

The geography data type accounts for the shape of the Earth. Consequently, this data type is based on a chosen 3D model of the Earth, and sets the positions of objects using angles of coordinates. In contrast, geometric data assigns spatial data as flat or planar. This type uses projections and Cartesian coordinates, which measure x and y of a point of origin.

Both data types are useful, but in different contexts. Both models appear different when visualized, so this is an important factor in selecting which datatype to use. In terms of accuracy, the geographic data type is good for cases where precision is very important. Likewise, for map making, projections can cause issues for how to visualize the Earth as a flat surface, as we have to make arbitrary decisions about where to make these splits. In contrast, the geometry data type offers more simplicity and efficiency in calculations, and less computational power is an important advantage.

g. **Spatial indexes**
A spatial index is specifically (and can only be used) for geography or geometry data types, and they help deal with the computational load and complexity of working with spatial data. As we can see in this database, measuring distances between vertices (e.g., finding the shortest distance between objects) quickly requires many computations.

In order to understand spatial indexes, it is important to understand that spatial queries are executed in SQL in two stages. First, SQL uses a primary filter, a quick way to approximate a set of candidate results, though this first filter is not sufficiently stringent to determine which records should not be included. Second, SQL then applies a secondary filter that is more stringent and refines the results further, ultimately producing the final results. Thus, spatial indexes work as a primary filter for spatial queries by identifying the starting set of results. Spatial indexes implement a grid system to index objects in grid cells at various levels of resolution, helping identify objects and areas of relevance to the query. Thus, spatial indexes help minimize the number of calculations necessary, improving query performance.

There are three key rules for creating spatial indexes: the covering rule, the deepest-cell rule, and the cells-per-object rule. The covering rule posits if any grid cell of any level is completely covered by a geometry, the index for that geometry should not contain any cells that further divide this cell into lower grid levels. The deepest-cell rule says that when a partially covered cell is subdivided, only the cell(s) at the deepest nested grid of intersection need to be added to the index. The final rule can be applied after the first two, and it aims to mitigate the problem of spatial indexes becoming too large to be useful. Thus, a limit should be set to avoid too much subdivision even when technically required, despite the deepest-cell rule, and the cell of the current grid level should be used instead in these instances.

Below is the SQL code I used to create the aforementioned spatial index:

```
/* Making a spatial index for grocery stores -
got extent from biggest map layer - properties - source - extent */

Create SPATIAL INDEX idxGeometry ON GROCERYSTORES (Shape)
```

```
USING Geometry_GRID
WITH(
Bounding_Box = (-79.639265, 43.580996, -79.115274, 43.855457),
GRIDS = (
        LEVEL_1 = MEDIUM,
        LEVEL_2 = MEDIUM,
        LEVEL_3 = MEDIUM,
        LEVEL_4 = MEDIUM),
        CELLS_PER_OBJECT = 16
        )
GO
```

h & i. **Spatial SQL Statements to Answer Research Questions**
Here I walk through my SQL queries, which questions they were used to answer, and provide a preview of the results they yield.

Query 0: I used this query to find which neighbourhood each home belonged to in order to help make foreign keys in RETIREMENTHOMES AND LTCHOMES tables.

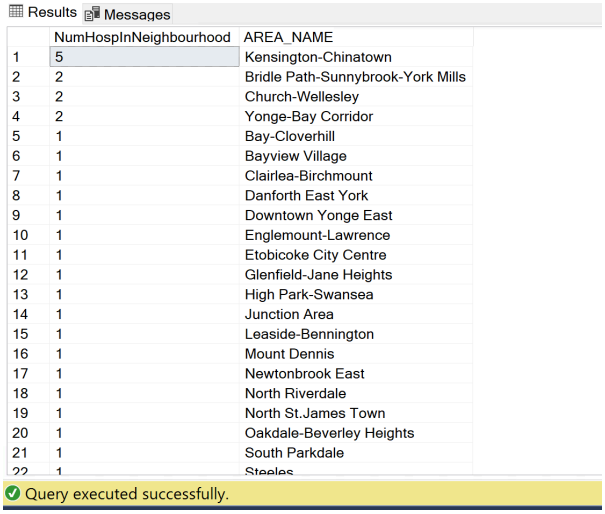
```
/* This query is used to find which NEIGHBOURHOODS each LTCHOME belongs to in order to set the foreign keys. Please note: this same query (changing names) was used to find the NEIGHBOURHOOD.OBJECTID for the RETIREMENTHOMES table as well. */

SELECT LTCHOMES.LTCNAME, NEIGHBOURHOODS.OBJECTID
FROM LTCHOMES, NEIGHBOURHOODS
WHERE (NEIGHBOURHOODS.Shape).STContains(LTCHOMES.Shape) = 1;
```

Query 1: “How many hospitals are there in each neighbourhood? Which neighborhoods have no hospitals?”

This query counts the number of hospitals per neighbourhood. I needed to use aggregate functions to do this, and I could use the GROUP BY clause to ensure the counts were performed and displayed by neighbourhood. I used the select and subtracted the count of neighbourhoods containing hospitals by the total number of hospitals, as I wanted the results to print all the neighbourhood names that do not have hospitals in them, along with their corresponding hospital counts of zero.

```
/* Finding number of hospitals per neighbourhood */
SELECT 31 - COUNT(NEIGHBOURHOODS.AREA_NAME) AS NumHospInNeighbourhood,
NEIGHBOURHOODS.AREA_NAME
FROM NEIGHBOURHOODS, HOSPITALS
```

```
WHERE (NEIGHBOURHOODS.Shape).STContains(HOSPITALS.Shape) = 0
GROUP BY NEIGHBOURHOODS.AREA_NAME
ORDER BY COUNT(NEIGHBOURHOODS.AREA_NAME) ASC,
NEIGHBOURHOODS.AREA_NAME ASC;
```



Results Messages

	NumHospInNeighbourhood	AREA_NAME
1	5	Kensington-Chinatown
2	2	Bridle Path-Sunnybrook-York Mills
3	2	Church-Wellesley
4	2	Yonge-Bay Corridor
5	1	Bay-Cloverhill
6	1	Bayview Village
7	1	Clairlea-Birchmount
8	1	Danforth East York
9	1	Downtown Yonge East
10	1	Englemount-Lawrence
11	1	Etobicoke City Centre
12	1	Glenfield-Jane Heights
13	1	High Park-Swansea
14	1	Junction Area
15	1	Leaside-Bennington
16	1	Mount Dennis
17	1	Newtonbrook East
18	1	North Riverdale
19	1	North St.James Town
20	1	Oakdale-Beverley Heights
21	1	South Parkdale
22	1	Steeles

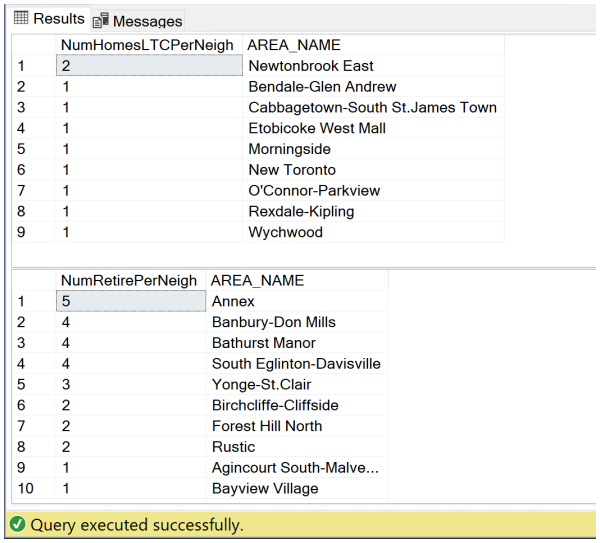
Query executed successfully.

Query 2 (technically two queries): “Which neighborhoods have the most LTCS and Retirement Homes?”

This query helps find the number of LTCHOMES versus RETIREMENT homes in the NEIGHBOURHOODS. I wrote these as two separate queries, and they can be used to compare across these two types of homes. These make use of the .STContains() method, where a value that equals 1 means there is a home in said neighbourhood.

```
/* FIND NUMBER LTC HOMES PER NEIGHBOURHOOD */
SELECT COUNT(NEIGHBOURHOODS.AREA_NAME) AS NumHomesLTCPerNeigh,
NEIGHBOURHOODS.AREA_NAME
FROM NEIGHBOURHOODS,LTCHOMES
WHERE (NEIGHBOURHOODS.Shape).STContains(LTCHOMES.Shape) = 1
GROUP BY NEIGHBOURHOODS.AREA_NAME
ORDER BY COUNT(NEIGHBOURHOODS.AREA_NAME) DESC,
NEIGHBOURHOODS.AREA_NAME ASC;

/*FIND NUMBER RETIREMENT HOMES PER NEIGHBOURHOOD */
SELECT COUNT(NEIGHBOURHOODS.AREA_NAME) AS NumRetirePerNeigh,
NEIGHBOURHOODS.AREA_NAME
FROM NEIGHBOURHOODS, RETIREMENTHOMES
WHERE (NEIGHBOURHOODS.Shape).STContains(RETIREMENTHOMES.Shape) = 1
GROUP BY NEIGHBOURHOODS.AREA_NAME
```

ORDER BY COUNT(NEIGHBOURHOODS.AREA_NAME) DESC,
NEIGHBOURHOODS.AREA_NAME ASC;



Results Messages

	NumHomesLTCPerNeigh	AREA_NAME
1	2	Newtonbrook East
2	1	Bendale-Glen Andrew
3	1	Cabbagetown-South St.James Town
4	1	Etobicoke West Mall
5	1	Morningside
6	1	New Toronto
7	1	O'Connor-Parkview
8	1	Rexdale-Kipling
9	1	Wychwood

	NumRetirePerNeigh	AREA_NAME
1	5	Annex
2	4	Banbury-Don Mills
3	4	Bathurst Manor
4	4	South Eglinton-Davisville
5	3	Yonge-St.Clair
6	2	Birchcliffe-Cliffside
7	2	Forest Hill North
8	2	Rustic
9	1	Agincourt South-Malve...
10	1	Bayview Village

Query executed successfully.

Query 3: “Are grocery stores located in close proximity to LTC homes? Which LTC homes have the nearest grocery stores?”

For this query, I used a loop to try to find the closest grocery store for every LTCHOMES. As in other programming languages, I was able to create a counter variable to look through all the LTC Homes and get the name and distance to the nearest Grocery Store and stop after running through the 10 LTCHOMES. I will note, I did not run this to compare for the RETIREMENTHOMES, as this would be computationally laborious, though possible. I then output the distance of the nearest grocery store to every LTC Home in metres.

```
/* Loop outputs the distance of the closest grocery store for every LTC home - loop goes to 1-11 because there are 10 LTC homes */
DECLARE @Counter INT
SET @Counter=1
WHILE ( @Counter <= 11)
BEGIN
   SELECT TOP 1 LTCHOMES.LTCNAME,
ROUND((LTCHOMES.Shape.STDistance(GROCERYSTORES.Shape)/100), 0) as ClosestStoreDistInMetres
         FROM GROCERYSTORES, LTCHOMES
         WHERE LTCHOMES.OBJECTID_1 = @Counter
   SET @Counter  = @Counter  + 1
END
```

Results Messages

	LTCNAME	ClosestStoreDistInMetres
1	Bendale Acres	242

	LTCNAME	ClosestStoreDistInMetres
1	Carefree Lodge	133

	LTCNAME	ClosestStoreDistInMetres
1	Castleview Wychwood Towers	137

	LTCNAME	ClosestStoreDistInMetres
1	Cummer Lodge	124

	LTCNAME	ClosestStoreDistInMetres
1	Fudger House	172

	LTCNAME	ClosestStoreDistInMetres
1	Kipling Acres	33

	LTCNAME	ClosestStoreDistInMetres
1	Lakeshore Lodge	173

	LTCNAME	ClosestStoreDistInMetres
1	True Davidson Acres	214

	LTCNAME	ClosestStoreDistInMetres
1	Wesburn Manor	120

	LTCNAME	ClosestStoreDistInMetres
1	Seven Oaks	283

	LTCNAME	ClosestStoreDistInMetres

Query 4: “How much green space (here being parks) are present in each neighborhood?”

This query is used to sum the park area in each neighborhood. It also combines spatial methods, and it sums the .STArea() method to calculate the areas. Again, the ORDER BY statement is used.

```
SELECT NEIGHBOURHOODS.AREA_NAME, SUM(PARKS.Shape.STArea()) AS
TotalParkSpace
FROM PARKS, NEIGHBOURHOODS
WHERE (NEIGHBOURHOODS.Shape).STContains(PARKS.Shape) = 1
GROUP BY NEIGHBOURHOODS.AREA_NAME
ORDER BY SUM(PARKS.Shape.STArea()) DESC;
```

Results Messages

	AREA_NAME	ParkSpaceInKilometres
1	Morningside Heights	10844.7914973145
2	West Humber-Clairville	3125.31704150391
3	St Lawrence-East Bayfront-The Islands	1751.89078100586
4	Rockcliffe-Smythe	1690.99710449219
5	High Park-Swansea	1583.71007250977
6	Oakdale-Beverley Heights	1257.75199829102
7	Downsview	1224.43629858398
8	Milliken	1064.88125366211
9	York University Heights	1055.6181237793
10	Stonegate-Queensway	1036.56698779297
11	Edenbridge-Humber Valley	1012.51057983398
12	Wexford/Maryvale	975.487017578125
13	Golfdale-Cedarbrae-Woburn	960.600188720703
14	West Rouge	943.152211181641
15	Tam O'Shanter-Sullivan	893.861148193359
16	Lansing-Westgate	893.848574707031
17	Eringate-Centennial-West Deane	814.385922851563
18	Kennedy Park	804.470802734375
19	Woburn North	749.232126708984
20	Cliffcrest	718.154658691406
21	Banbury-Don Mills	696.718065185547
22	West Hill	664.530601806641

Query executed successfully.

Query 5: “Are there any Retirement homes within 1km of LTC homes?”
/* If any retirement homes and LTC homes within 1km of each other*/

This query was used to help see if any LTC homes are located nearby the Retirement Homes. I used one kilometre as that is usually considered to be an approximate 10 minute walk (though depending on walking ability, etc.). I used the .STBuffer function and the STWithin methods to do this.

```
SELECT  LTCHOMES.LTCNAME as LTCHomeName, RETIREMENTHOMES.HOMENAME as
RetireHomeName
FROM LTCHOMES, RETIREMENTHOMES
WHERE LTCHOMES.Shape.STWithin(RETIREMENTHOMES.Shape.STBuffer(1000)) = 1;
```

Results Messages

	LTCHomeName	RetireHomeName
1	Castleview Wychwood Towers	The O'Neill Centre
2	Castleview Wychwood Towers	Briar Crest Retirement Home