University of Tartu
Faculty of Science and Technology
Institute of Ecology and Earth Sciences
Department of Geography

LTOM.02.040 Spatial Databases

Individual project

Author: Petr Anashin
Supervisor: PhD Valentina Sagris

Tartu 2022

Table of contents.

Introduction

The main purpose of the project was to analyse accessibility of bike sharing stations (Tartu Smart Bike Share) in Tartu depending on spatial proximity of stations to the users and compare how does it differ in different locations across the city in order to offer a way to improve the service.

The project is valuable since cycling is one of the means of urban mobility, while bike sharing services provide convenient way to move within the city. Moreover, improvement of bike sharing network will help to reduce carbon emissions because it’s eco-friendly transportation type.

To estimate the accessibility of bike sharing network across the city, the following research question have been formulated:

1. How many bike sharing stations are located within the city and 600 meter distance to the city?
2. What is the number of bike sharing stations in each neighbourhood?
3. How many times each bike sharing station has been used for the given time period and what is the total number of usages for each district?
4. How accessible are bike sharing stations for the population in different locations of the city?

1 Database design and implementation

1.1 Main concepts and terms (conceptual model)

Bike sharing stations – network of electric bike stations that is maintained by Tartu Smart Bike Share.

Bike routes – information about each bike trip for particular period of time including start and end stations, trip length and time, membership status etc.

Population grid – 100 by 100 meter grid covering the city of Tartu. Originally it has been received by Statistics Estonia. Each cell displays population number that lived within its borders in 2015.

Neighbourhoods – boundaries of Tartu's neighbourhoods.

1.2 Logical model

Logical relationships between concepts are represented in Figure 1. Bike_share_docks and Bike_routes are connected by many-to-many relationship since one bike sharing station is related to many bike trips, while one bike route may start and finish at 2 different stations. The other two tables, Tartu_neighbourhoods and Tartu_pop_grid, will be connected with the help of spatial relationships.

Figure 1. Logical model

1.3 Data and tables

Spatial database was comprised of 4 tables. Bike routes were received from Tartu's Open Data as a csv file [1]. They represent trips that have been done in 4 days between the 1-st July 2019 and the 4-th July 2019. Also this table contains data on date and time when the bike has been locked and unlocked, start and end stations, length, cost and duration of the trip, membership status of the user and bike code.

Information about bike sharing stations that is stored in Bike_share_docks table displays location of stations on the 1-st of December 2022 and it is retrieved as csv file from Tartu linna geoHUB [2]. Besides coordinates of sharing stations, it also has column with number of bicycles at each station and station's name.

Population grid was derived from one of the previous courses, originally data by itself was received by Statistics Estonia. Each cell has a size 100 by 100 meters and information on population number living within this cell. Data was in vector format in csv file.

In the last table was collected data on spatial boundaries of Tartu's neighbourhoods. It was extracted from OpenStreetMap as a shapefile on the 1-st of December 2022. Neighbourhoods are displayed as multipolygons.

Table 1. Bike_share_docks table

Column name	**Type**	**Description**
X	*DOUBLE PRECISION*	X coordinate in Estonian CRS
Y	*DOUBLE PRECISION*	Y coordinate in Estonian CRS

Rataste_arv	*INTEGER*	Number of bicycles at the station
Nimi	*VARCHAR(256)*	Name of the station
Staatus	*INTEGER*	Status of the bike sharing station
Linnaosa	*INTEGER*	ID of the neighbourhood

Table 2. Bike_routes table

Column name	**Type**	**Description**
Unlock Date	*DATE*	Date when the bike has been unlocked
Unlock Time	*TIME*	Time when the bike has been unlocked
Lock Date	*DATE*	Date when the bike has been locked
Lock Time	*TIME*	Time when the bike has been locked
Membership	*VARCHAR(256)*	Subscription/membership type
Length	*DECIMAL(10,3)*	Length of the bike trip in km
Duration (Minutes)	*DECIMAL(10,3)*	Duration of the bike trip in minutes
Bike No	*INTEGER*	Bicycle digital code
Bike Type	*VARCHAR(256)*	Type of the bicycle (electric or usual one)
Cost	*DECIMAL(10,3)*	Price of the trip
Start Station	*VARCHAR(256)*	Name of the station where trip started
End Station	*VARCHAR(256)*	Name of the station where trip finished

Table 3. Tartu_pop_grid table

Column name	**Type**	**Description**
TOTAL_POPULATION	*INTEGER*	Population number in population's grid cell
geom	*GEOMETRY*	Multipolygon in WGS 84

Table 4. Tartu_neighbourhoods table

Column name	**Type**	**Description**
population	*INTEGER*	Population number of the neighbourhood
name	*VARCHAR(256)*	Neighbourhood's name
place	*VARCHAR(256)*	Unit of the administrative's division
type	*VARCHAR(256)*	Type of the geometry
geom	*GEOMETRY*	Multipolygon in WGS 84

1.4 Normalization, preprocessing and validation

1.4.1 Setting primary keys and removing extra columns

In order to guarantee uniqueness of ID for each table there were created additional columns “ID” with serial data type that were used as primary keys.

In the table “Bike_share_docks” columns X and Y have been removed after construction of geometry column, also there were removed attributes staatus and linnaosa because in the first column all values were filled with 1, while there were no corresponding names of neighbourhoods for the second column.

In the table “Tartu_neighbourhoods” attributes place, type and population were also removed. Columns place and type were filled only with 1 value and population column had only null values.

1.4.2 Construction of geometry columns, transformation of coordinate system and validation

Table “tartu_pop_grid” was filled with the help of corresponding csv file where geometry was stored in string format in the column “geom_text”. Therefore a new column “geom” has been added and via function ST_GeomFromText WKT format was changed to geometry column, while old column in string format was removed. As the CRS was WGS 84, coordinate system of the “geom” column was transformed to Estonian coordinate system via ST_Transform.

```
ALTER TABLE proj.tartu_pop_grid
ADD COLUMN geom geometry(MULTIPOLYGON,4326);


UPDATE proj.tartu_pop_grid SET
geom = ST_GeomFromText(geom_text, 4326);


ALTER TABLE proj.tartu_pop_grid
DROP COLUMN geom_text;


--change CRS to Estonian coord system
ALTER TABLE proj.tartu_pop_grid
ALTER COLUMN geom
TYPE Geometry(MULTIPOLYGON, 3301)
USING ST_Transform(geom, 3301);
```

In the table “tartu_neighbourhoods” CRS also was changed in the same way.

```
ALTER TABLE proj.tartu_neighbourhoods
```

```
ALTER COLUMN geom
TYPE Geometry(MULTIPOLYGON, 3301)
USING ST_Transform(geom, 3301);
```

Table “bike_share_docks” had coordinates of bike sharing stations (x and y) in Estonian coordinate system. For that table a new column “geom” was created with Estonian coordinate system and point type of geometry using AddGeometryColumn function. Then this column was filled with the help of ST_GeomFromText that took x and y coordinates and transformed it into geometry type from WKT.

```
SELECT AddGeometryColumn('proj', 'bike_share_docks', 'geom', 3301,
'POINT', 2);

UPDATE proj.bike_share_docks SET geom =
ST_GeomFromText('POINT('||x||' '||y||')', 3301);
```

All geometries have been validated based on the results of ST_IsValid function. There were no invalid geometries.

```
SELECT COUNT(ST_IsValid(geom)='true') AS valid, COUNT(id) AS
all_polygons FROM
proj.tartu_pop_grid;

SELECT COUNT(ST_IsValid(geom)='true') AS valid, COUNT(id) AS
all_polygons FROM
proj.tartu_neighbourhoods;
```

```
SELECT COUNT(ST_IsValid(geom)='true') AS valid, COUNT(id) AS
all_polygons FROM
proj.bike_share_docks;
```

1.5 Database structure

As it was written before, tables “bike_routes” and “bike_share_docs” have many-to-many relationship type, thus additional junction table “bike_docks_routes” was created to connect these two tables. In that table are stored all unique combinations of ID’s from “bike_routes” and “bike_share_docs”. Additionally, foreign keys have been assigned to columns “dock_id” and “route_id”.

```
--additional table to create many to many relationship between bike_routes and
bike_share_docks


CREATE TABLE proj.bike_docks_routes(
                    id serial PRIMARY KEY,
                    dock_id integer,
                    route_id integer
                    );


--create many to many relationship between bike_routes and bike_share_docks
SELECT proj.bike_share_docks.id as dock_id, proj.bike_routes.id as route_id
FROM proj.bike_share_docks, proj.bike_routes;


INSERT INTO proj.bike_docks_routes (dock_id, route_id)
```

```
SELECT proj.bike_share_docks.id as dock_id, proj.bike_routes.id as route_id
FROM proj.bike_share_docks, proj.bike_routes;

ALTER TABLE proj.bike_docks_routes ADD CONSTRAINT fk_dock_id
FOREIGN KEY (dock_id) REFERENCES proj.bike_share_docks (id);

ALTER TABLE proj.bike_docks_routes ADD CONSTRAINT fk_route_id
FOREIGN KEY (route_id) REFERENCES proj.bike_routes (id);
```

Final database structure is displayed in Figure 2. It has 2 one-to-many relationships between “bike_routes” – “bike_docks_routes” and “bike_docks_routes” – “bike_share_docks”. Tables “tartu_neighbourhoods” and “tartu_pop_grid” will be connected with the help of spatial joins.

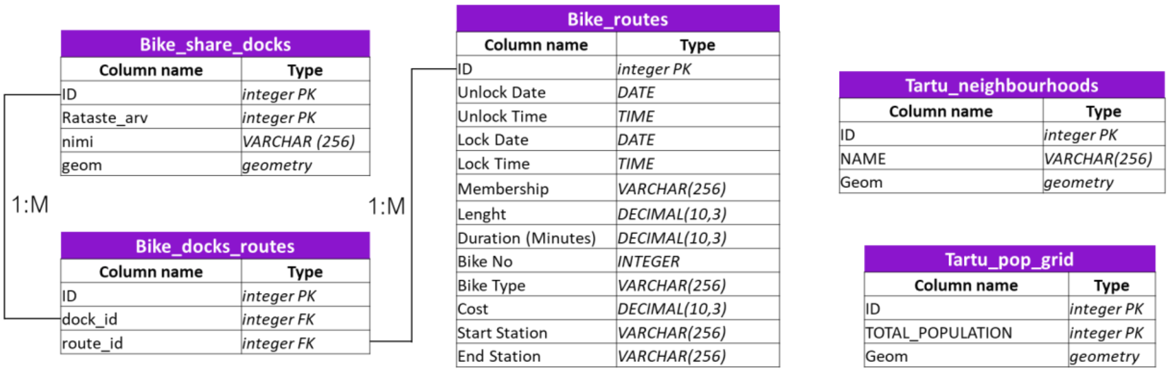

Figure 2. Database structure

2 Spatial analysis

2.1 Overview of the study objects and area

The city of Tartu was selected as a study area of the project. However, there are bike sharing stations that are located outside the city, but some of them also have impact on bike's network accessibility.

Overall, there are 99 stations in the table "bike_share_docs". Additionally I decided to check how many stations are within the city and less than 600 meters to its borders. Firstly, multipolygons of "tartu_neighbourhoods" have been united via ST_Union, then based on ST_DWithin was counted number of bike sharing stations that fall within the city and 600 meter distance around it. There were 88 such stations.

```
--how many bike stations are within the city and 600 meters to the city?
--Tartu borders
CREATE VIEW proj.tartu_borders AS
SELECT ST_Union(a.geom) AS geom FROM proj.tartu_neighbourhoods as a;

--number of bike sharing stations located in Tartu and 600m to the city
SELECT COUNT(b.*)
FROM proj.tartu_borders as a, proj.bike_share_docks as b
WHERE ST_DWithin(a.geom, b.geom, 600);
```

Figure 3 demonstrates how bike sharing stations are distributed across the city. Density is high in the city center and it's eastern and south-western parts, nevertheless it is lower in the north and south-east.

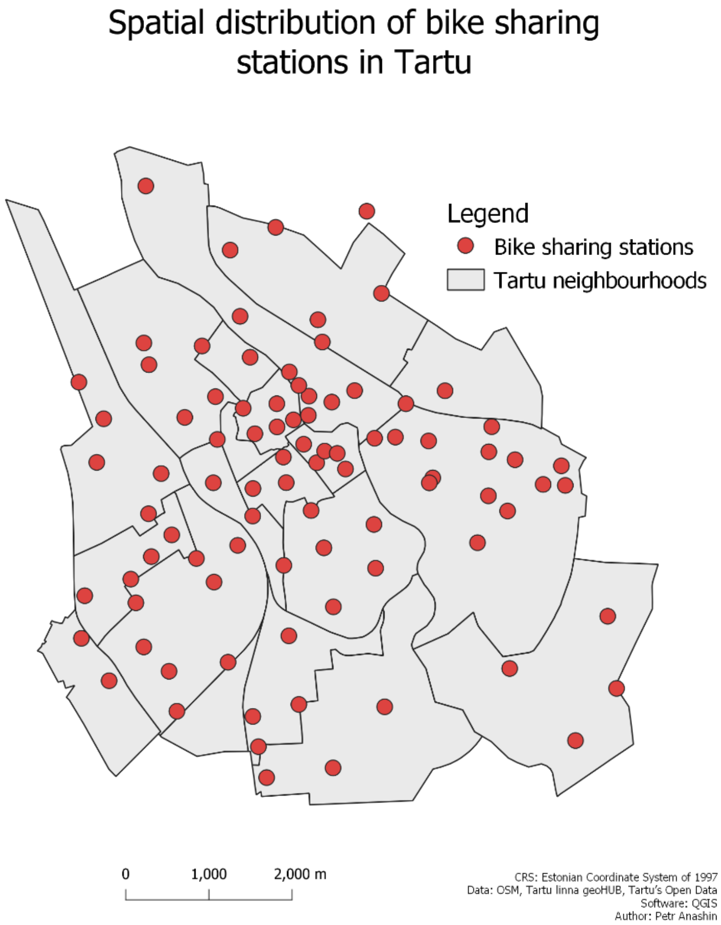

Figure 3. Spatial distribution of bike sharing station in Tartu

2.2 Spatial queries

Initially was calculated number of bike sharing stations in each neighbourhood. It has been done with the help of ST_Contains function that returned neighbourhoods from “tartu_neighbourhoods” table containing sharing stations from “bike_sharing_docks” table. Then data has been aggregated by GROUP BY. The results of visualization are represented in Figure 4.

```
CREATE VIEW proj.neighbourhood_docks_number AS
SELECT a.name, COUNT(a.name), a.geom
```

```
FROM proj.tartu_neighbourhoods as a, proj.bike_share_docks as b
WHERE ST_Contains(a.geom, b.geom) IS TRUE
GROUP BY a.name, a.geom ORDER BY COUNT(a.name) DESC;
```

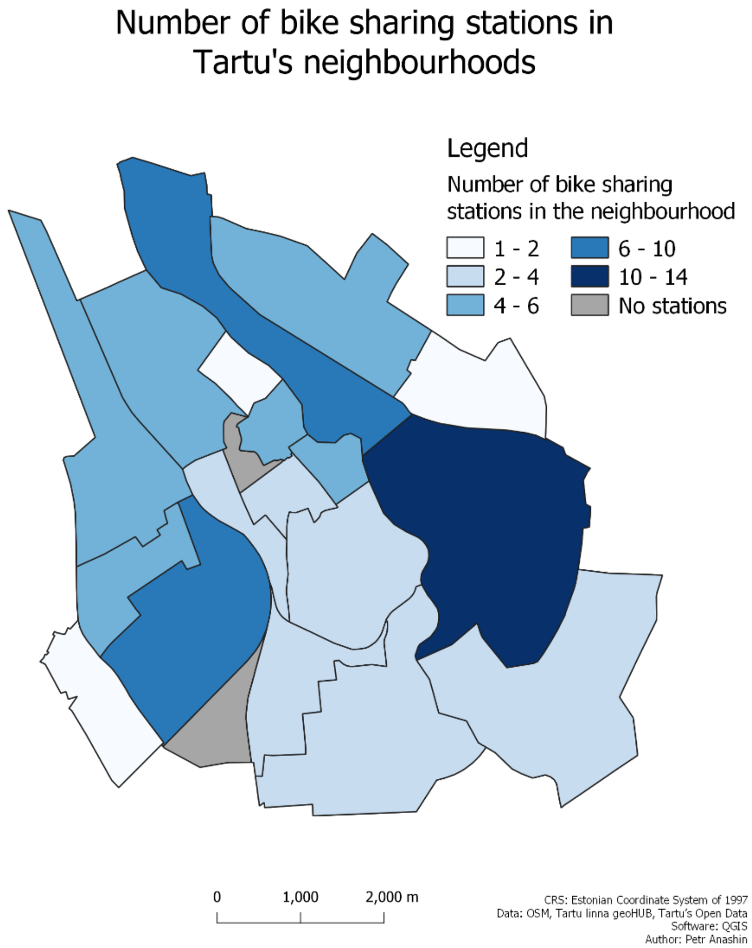

Figure 4. Number of bike sharing station in Tartu’s neighbourhoods

Afterwards have been written a spatial query to find out how many times bike sharing stations have been used in each neighbourhood. Firstly, it was measured how many times each station has been used. Junction table “bike_docks_routes” was joined to “bike_share_docks” using left join, then table “bike_routes” was joined to the junction table also with the help of left join. Final result was aggregated and the table with station names, geometries and number of usages was received.

After that was used the query similar to the previous one, but instead of counting was used sum. Map displaying the results is shown in Figure 5.

```
--count number of usages of each station
CREATE VIEW proj.dock_usages_number AS
SELECT bike_docks.nimi as dock_name, COUNT(bike_docks.nimi),
bike_docks.geom
FROM proj.bike_share_docks as bike_docks
LEFT JOIN proj.bike_docks_routes as docks_routes
ON bike_docks.id = docks_routes.dock_id
LEFT JOIN proj.bike_routes as bike_routes
ON docks_routes.route_id = bike_routes.id
WHERE bike_docks.nimi=bike_routes.start_station
GROUP BY bike_docks.nimi, bike_docks.geom ORDER BY
COUNT(bike_docks.nimi) DESC;

--count number of bike sharing usages per each neighbourhood
CREATE VIEW proj.neighbourhood_usages_number AS
SELECT a.name, SUM(b.count), a.geom
FROM proj.tartu_neighbourhoods as a, proj.dock_usages_number as b
WHERE ST_Contains(a.geom, b.geom) IS TRUE
GROUP BY a.name, a.geom ORDER BY SUM(b.count) DESC;
```

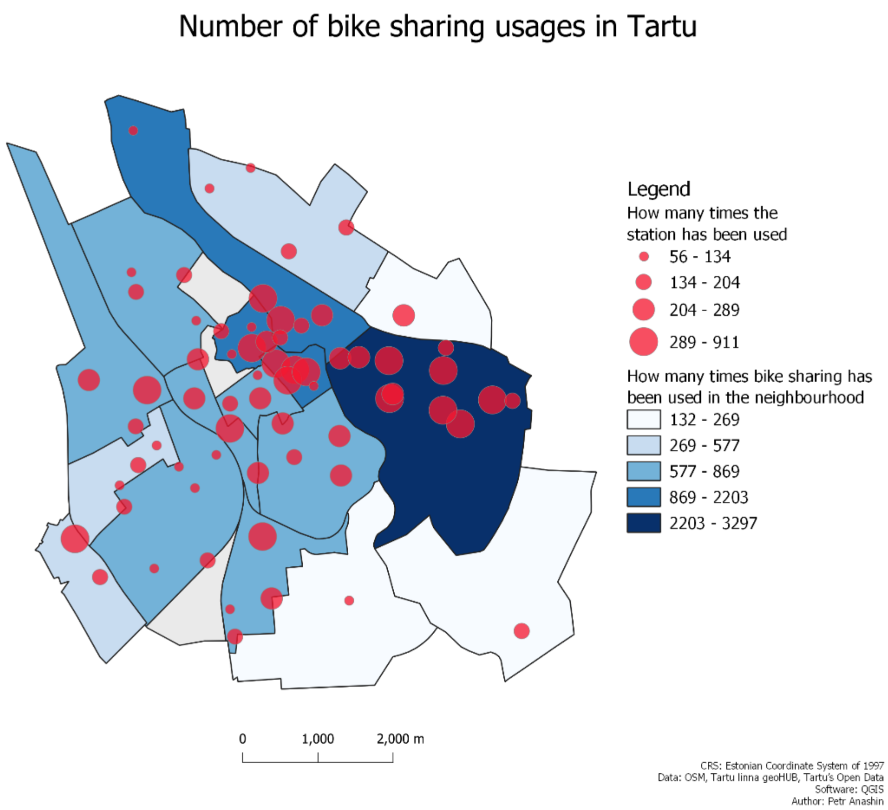

Figure 5. Number of bike sharing usages in Tartu

The last spatial query was related to the measuring of accessibility index for cells in population grid cell and aggregated accessibility index for each neighbourhood.

In the beginning there were created 600 meter buffers with the help of ST_Buffer corresponding to 10-minutes walking distance around centroids of population grid cells that were received using ST_Centroid in relation to geometry column of "tartu_pop_grid" table.

Then was estimated number of bicycles available within each buffer. Number of bikes at each station was provided in rataste_arv column of "bike_share_docks" table. All stations that were within particular buffer were captured using

ST_Contains and then the query was aggregated based on sum of number of bikes from rataste_arv column.

Applying similar query was received population number living within each buffer. Function ST_Contains captured population grid centroids that were within buffers and the query has been aggregated using total_population column corresponding to population number living in each grid cell.

As for the measuring of total population within each buffer was used separate view with buffers that was later converted into table to improve performance, in order to estimate accessibility index it was needed to join original population grid with this new table “table_pop_grid_population_buff”. It has been joined with the help of left join using ID of population grid cells. Then accessibility index was calculated as Standard floating catchment area:

$$A_i^{fca} = \frac{S_i}{D_i} \tag{1}$$

In (1) A_i^{fca} is floating catchment area accessibility index for unit i, S_i is aggregated supply within a distance circle or time-based buffer around location i, D_i aggregated demand (or population) at location i. As a result, total number of bicycles that fall within the buffer around grid cell was divided by total population living in this buffer.

Then additionally for each neighbourhood were assessed mean and median accessibility index. With the help of ST_Within there were identified centroids of population grid that fall within each neighbourhood and using aggregation functions based on AVG and PERCENTILE_CONT(0.5) there were received mean and median values for each district. Median values demonstrated better representativeness because mean values are not sustainable towards outliers.

The results may be reviewed in Figure 6.

```
--count accessibility index for each population grid cell

--create buffers around centroids of pop grid cells
CREATE VIEW proj.pop_grid_buffers AS
SELECT a.id AS cell_id, a.total_population,
ST_Buffer(ST_Centroid(a.geom), 600) AS geom FROM proj.tartu_pop_grid
AS a;

--number of bikes available within each buffer around centroids of population
grid
CREATE VIEW proj.bikes_number AS
SELECT a.cell_id, SUM(b.rataste_arv) AS bikes_number, a.geom
FROM proj.pop_grid_buffers as a, proj.bike_share_docks as b
WHERE ST_Contains(a.geom, b.geom) IS TRUE
GROUP BY a.cell_id, a.geom ORDER BY SUM(b.rataste_arv) DESC;

--population number within each buffer
CREATE VIEW proj.pop_grid_population_buff AS
SELECT a.cell_id, SUM(b.total_population) as total_population, a.geom
FROM proj.pop_grid_buffers as a, proj.tartu_pop_grid as b
WHERE ST_Contains(a.geom, ST_Centroid(b.geom)) IS TRUE
GROUP BY a.cell_id, a.geom ORDER BY SUM(b.total_population) DESC;

--create table from view to improve performance
CREATE TABLE proj.table_pop_grid_population_buff AS SELECT * FROM
proj.pop_grid_population_buff;
```

```
--join information about total population within 600m buffer around each cell and number of bikes
--calculate accessibility index as total number of bikes / total population
CREATE VIEW proj.accessibility_index AS
SELECT a.id, b.total_population, c.bikes_number, CAST(c.bikes_number AS float)/CAST(NULLIF(b.total_population,0) AS float) as accessibility_index, a.geom
FROM proj.tartu_pop_grid AS a
LEFT JOIN proj.table_pop_grid_population_buff AS b
ON a.id = b.cell_id
LEFT JOIN proj.bikes_number AS c
ON b.cell_id = c.cell_id;

--mean accessibility index in each neighbourhood
CREATE VIEW proj.neighbourhoods_accessibility AS
SELECT a.name, AVG(b.accessibility_index), a.geom
FROM proj.tartu_neighbourhoods as a, proj.accessibility_index as b
WHERE ST_Within(ST_Centroid(b.geom), a.geom) IS TRUE
GROUP BY a.name, a.geom ORDER BY AVG(b.accessibility_index) DESC;

--median accessibility index in each neighbourhood
CREATE VIEW proj.neighbourhoods_median_accessibility AS
SELECT a.name, PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY b.accessibility_index), a.geom
FROM proj.tartu_neighbourhoods as a, proj.accessibility_index as b
WHERE ST_Within(ST_Centroid(b.geom), a.geom) IS TRUE
GROUP BY a.name, a.geom;
```

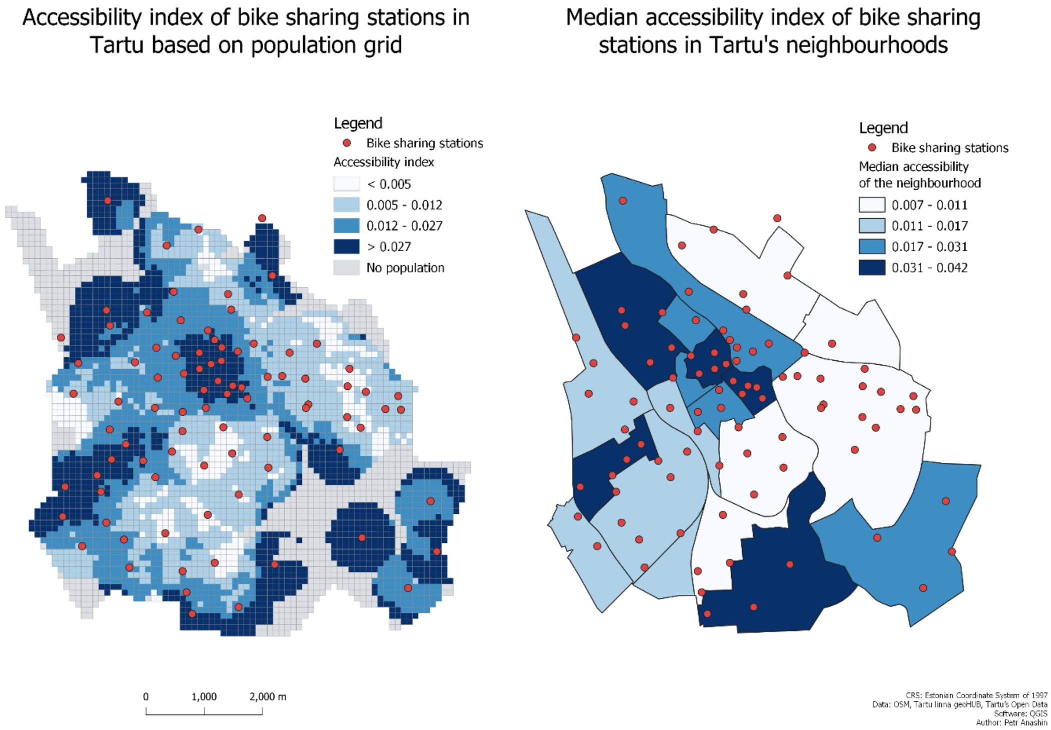

Figure 6. Accessibility index in the scope of population grid and neighbourhoods

The workflow of the spatial analysis is shown in Figure 7.

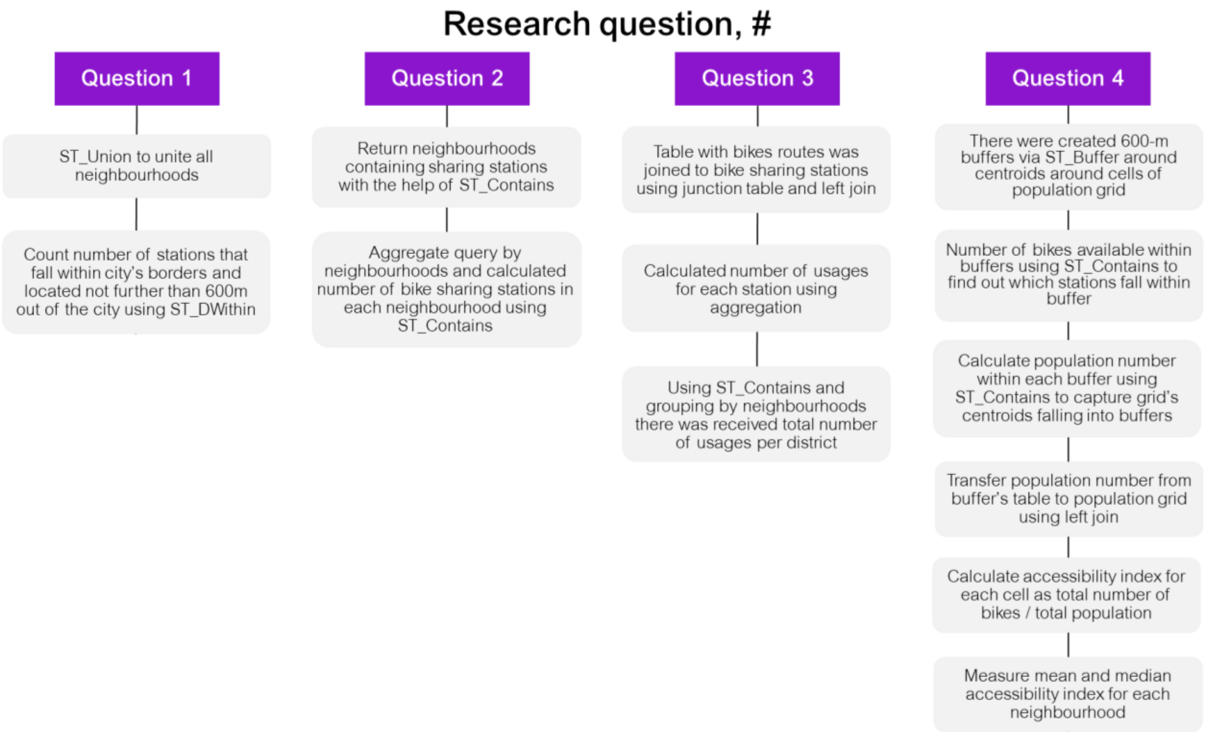

Figure 7. Workflow of the spatial analysis

2.3 Results

It was found out that Tartu Smart Bike Share owns 99 bike sharing stations, 88 of them are located within the city or not further than 600 meters away from its borders.

Number of bike sharing stations varies significantly among the neighbourhoods. For instance, Annelinn has 14 stations, while some neighbourhoods like Supilinn has only 1. They are more spread in the city center and it’s eastern and south-western parts, but as for the north and south-east the network is sparser there.

Talking about number of station’s usages, we may draw a conclusion from the Figure 5 that density of bike sharing network affect number of its usages because in the south and north residents use bike sharing relatively rare compared to the city center or eastern part of the city. However, despite the fact that Tammelinn and Maarjamõisa in the south-west have average number of station, it does not convert into frequent usage of bike sharing stations.

Accessibility index reveals that Annelinn, Ropka, Karlova and north-eastern neighbourhoods are the least accessible. Most probably it happens because demand for bike sharing service is higher than its supply due to high population density that do not match to the number of bike sharing stations. Therefore these destinations may be prospective in terms of future network’s expansion.

3 References

1) Tartu’s Open Data. IMO, Infotechnological Mobility Observatory. Retrieved December 1, 2022, from https://imo.ut.ee/en/infrastructure/estonian-open-government-data/tartus-open-data/?cn-reloaded=1.
2) LI rattaringluse parklad avaandmed. Tartu linna geoHUB. Retrieved December 1, 2022, from https://geohub.tartulv.ee/datasets/Tartu::li-rattaringluse-parklad-avaandmed/about.