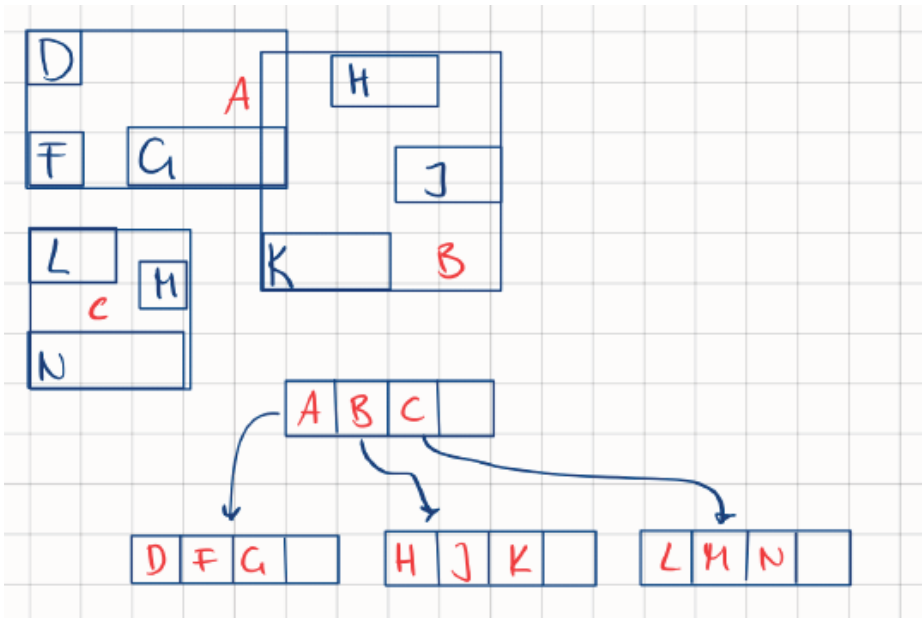
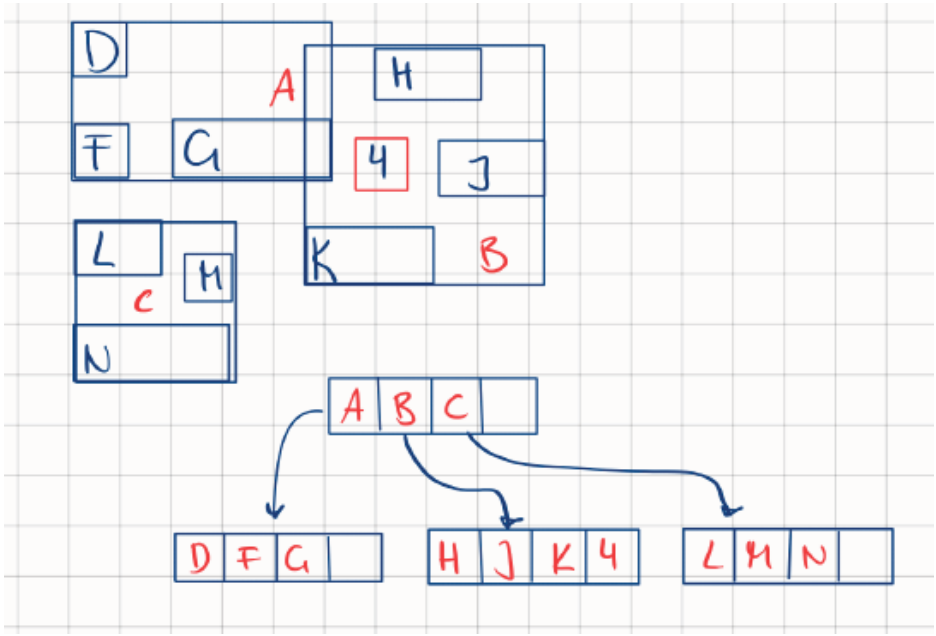
Group 6: Luc Jonker, 4836111, Ming-Chieh Hu, 6186416, Sophie Lipskaya 6306055

1. *Describe (and illustrate with drawings) the 'insert' algorithm of the R-tree. Assume that the objects D, F, G, H, J, K, L, M and N already in the R-tree and that subsequently (and in the given order) the following objects are added in this order 4, 3, 2 and 1 to the R-tree (show both the scene with the rectangle groupings and the R-tree after every step).*

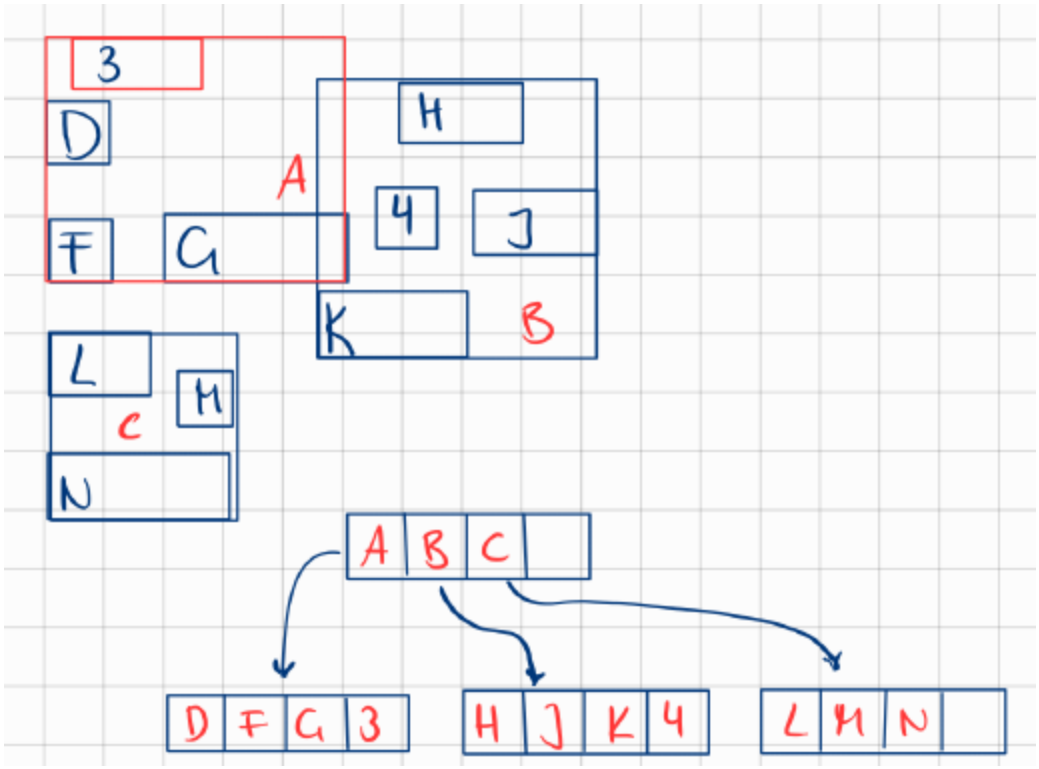
To begin:



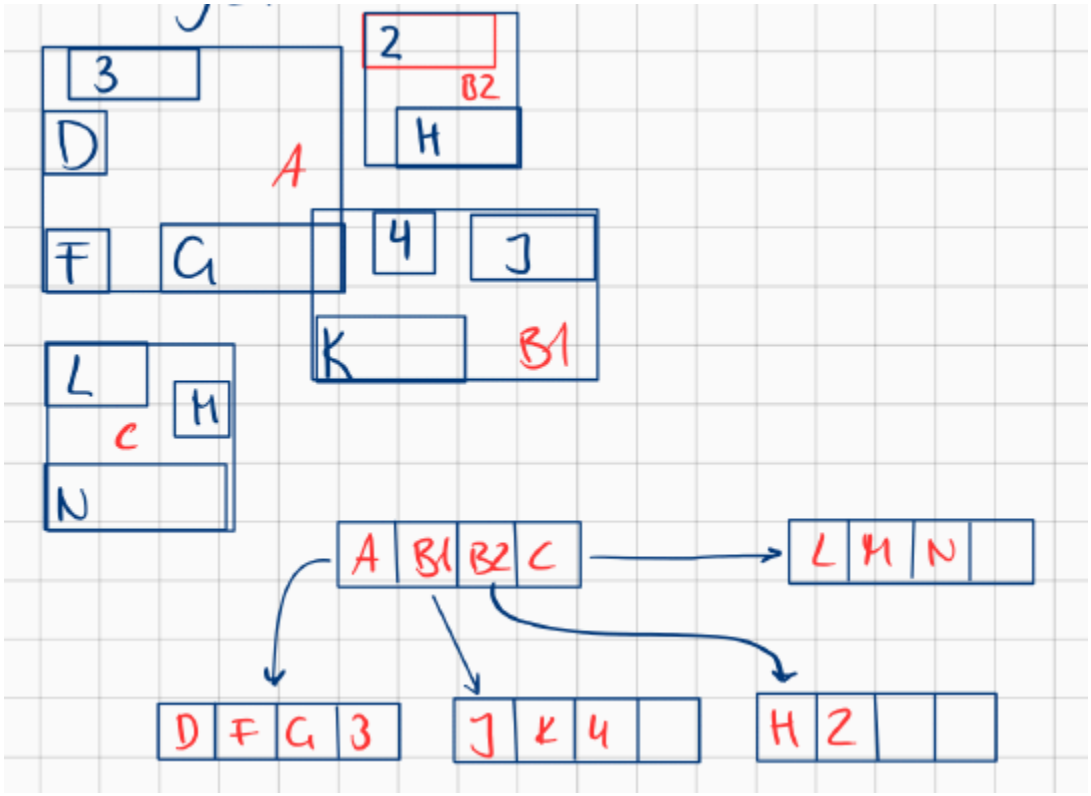
Insert item 4:



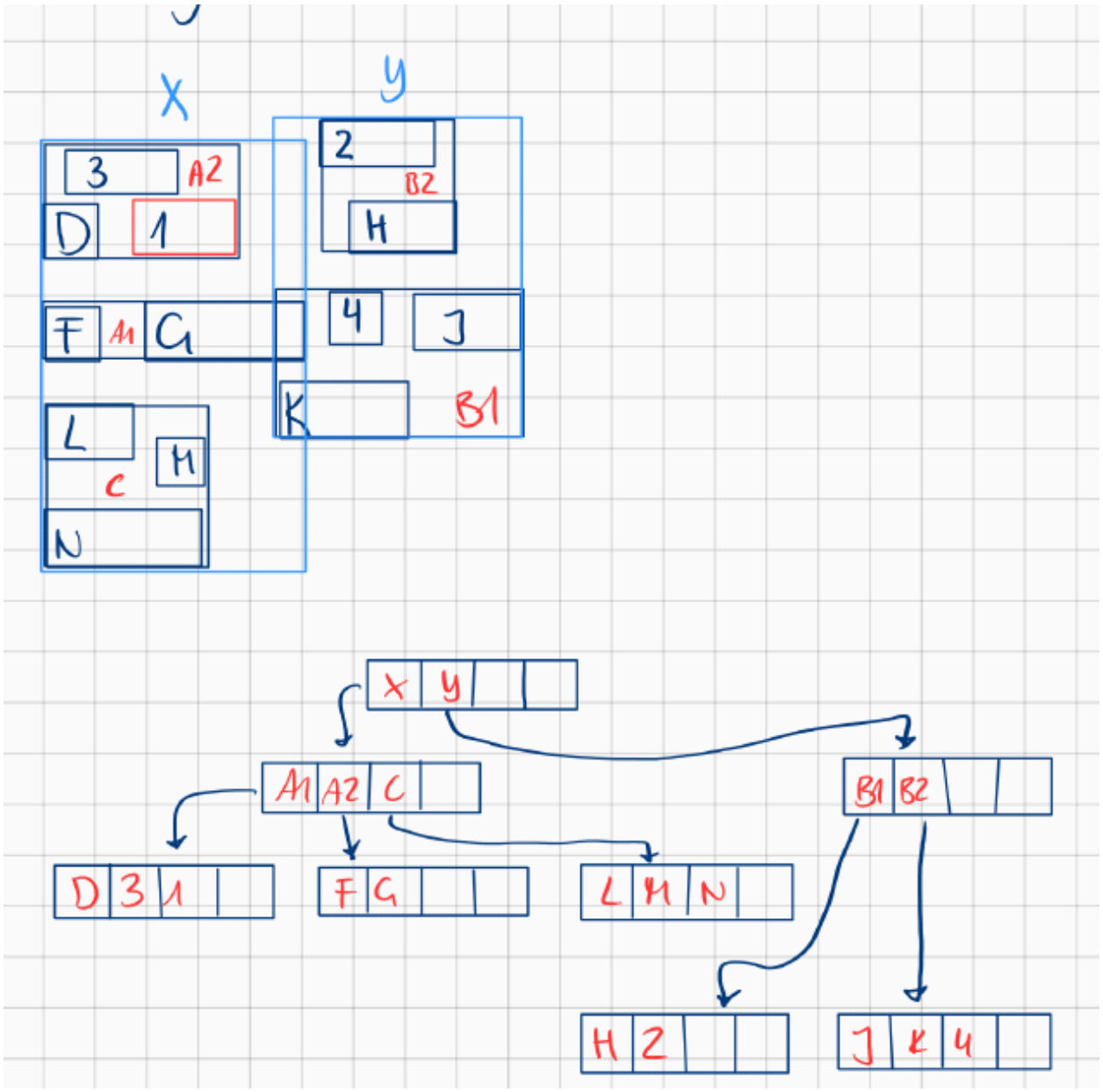
Insert item 3:



Insert item 2:



Insert item 1:



2. *Load some cadastral data (planar partition) in PostGIS with script from topomap_v3.sql: C:\Program Files\PostgreSQL\9.0\bin>psql -f \peter\doc\geoDBMS\topomap_v3.sql -U postgres test Note: the script also defin*

3. *es the tables, before actual data loading starts (see last page). Note 2: de not use pgadmin to load the data, but the command line tool psql (as shown above)*

```
PS C:\Program Files\PostgreSQL\17\bin> .\psql -f C:\Users\Chizzy\Documents\Geomatics\Database_mgmt\topomap_v3.sql -U postgres test
Password for user postgres:

SET
SET
BEGIN
psql:C:/Users/Chizzy/Documents/Geomatics/Database_mgmt/topomap_v3.sql:8: NOTICE:  table "kadnl050k_node" does not exist, skipping
DROP TABLE
psql:C:/Users/Chizzy/Documents/Geomatics/Database_mgmt/topomap_v3.sql:9: NOTICE:  table "kadnl050k_edge" does not exist, skipping
DROP TABLE
psql:C:/Users/Chizzy/Documents/Geomatics/Database_mgmt/topomap_v3.sql:10: NOTICE:  table "kadnl050k_face" does not exist, skipping
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
COMMIT
BEGIN
COPY 129441
COMMIT
BEGIN
COPY 178814
COMMIT
BEGIN
COPY 50238
COMMIT
VACUUM
VACUUM
VACUUM
SET
SET
BEGIN
psql:C:/Users/Chizzy/Documents/Geomatics/Database_mgmt/topomap_v3.sql:358560: NOTICE:  table "kadnl050k_face_geometry" does not exist, skipping
DROP TABLE
CREATE TABLE
COMMIT
BEGIN
COPY 50238
COMMIT
```

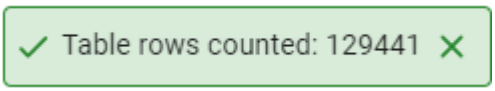
4. *What are functions st_buildarea and st_collect doing, check the PostGIS manual. Describe the expected input and explain what will be the output*

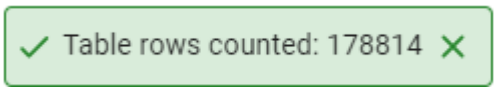
   ST_BuildArea:
   This function can take several separate line segments / polygons and merge them into 1 (or multiple) areal polygons. The merge process includes detecting and making inner rings into holes. Use ST_Collect to form a collection if needed.

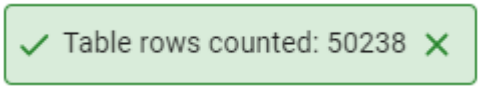
   Input: LineString, MultiLineString, Polygon, MultiPolygon or GeometryCollection
   Output: Either a Polygon or a MultiPolygon, NULL if no polygon is formed

   ST_Collect:
   Collects distinct geometries and collects them into one geometry collection.
   Input: Accepts two, an array of, or an aggregate rowset function of geometries.
   Output: Returns either a Multi* geometry or a GeometryCollection depending on if the input geometries are of the same or different types respectively.

5. *Count the number of nodes, edges, and faces*

Nodes: 

Edges: 

Faces: 

6. *Do some simple SQL statistics as specified below:*
    a. *find smallest polygon, largest polygon, average polygon area, total polygon area.*
    b. *similar for longest edge, shortest edge, average edge length, total edge length*
    c. *count the edges longer than 1000m, count the polygons larger 10000m2 .*

A:

Smallest:

```
SELECT face_id, ST_Area(geometry) as area, geometry
FROM kadnl050k_face_geometry
ORDER BY area asc
LIMIT 1;
```

	face_id integer	area double precision	geometry geometry
1	140299786	0.34225799998606254	0103000020407100000100000004000000894160E578A90141EC51B89E51021A418195438B82A901418D976E9250021A411D5A643B7CA901410C022B8753021A41894160E578A90141EC51B89E51021A41

Largest:

```
SELECT face_id, ST_Area(geometry) as area, geometry
FROM kadnl050k_face_geometry
ORDER BY area desc
LIMIT 1;
```

	face_id integer	area double precision	geometry geometry
1	140073148	1275974.345765506	0103000020407100000200000C101000096438B6CD4B6014123DBF9FED9B61A411904560E2AB70141B072681181B61A413E0AD7A352B701414E62105857B61A41F0A7C64B71B70141FCA9F1D235B61A4183C0CAA191

Average:

```
SELECT AVG(ST_Area(geometry)) as area
FROM kadnl050k_face_geometry;
```

	area double precision
1	6816.979198347352

Total:

```
SELECT SUM(ST_Area(geometry)) as area
FROM kadnl050k_face_geometry;
```

	area double precision
1	342471400.9665743

B:

Longest:

```
SELECT edge_id, ST_Length(geometry) as len, geometry
FROM kadnl050k_edge
ORDER BY len DESC
LIMIT 1;
```

	edge_id integer	len double precision	geometry geometry
1	141367849	2737.3955145413297	0102000020407100001D00000017D9CEF7EB6000415839B4C830BD19415839B4C8CC600041B29DEFA714BD19414A0C022B6D5F00417B14AE4784BB19418D976E12795D004154E3A51BF4B91941AC1C5A64415A004179E9

Shortest:

```
SELECT edge_id, ST_Length(geometry) as len, geometry
FROM kadnl050k_edge
ORDER BY len asc
LIMIT 1;
```

	edge_id integer	len double precision	geometry geometry
1	143999109	0.00583095188230724	0102000020407100000200000DD2406810D08024160E5D0224E7F19418195438B0D080241C976BE1F4E7F1…

Average:

```
SELECT AVG(ST_Length(geometry)) as len
FROM kadnl050k_edge;
```

	len double precision
1	46.62322635002013

Total:

```
SELECT SUM(ST_Length(geometry)) as len
FROM kadnl050k_edge;
```

	len double precision
1	8336885.5965525

C:

Edges longer than 1000m:

```
SELECT COUNT(len)
FROM (SELECT ST_Length(geometry) as len
FROM kadnl050k_edge
WHERE ST_Length(geometry) > 1000)
```

	count bigint
1	84

Polygons larger than 10000m2:

```
SELECT COUNT(area)
FROM (SELECT ST_Area(geometry) as area
FROM kadnl050k_face_geometry
WHERE ST_Area(geometry) > 10000)
```

	count bigint
1	8215

7. *Do two spatial joins (without spatial index):*

a. *of edge table with itself to check if there are crossing edges (and in order not to wait too long add condition that one of the crossing edges must at least be 500m); use: ST_Crosses*

```
SELECT kke.geometry, kke2.geometry
FROM kadnl050k_edge kke
      JOIN kadnl050k_edge kke2
      ON ST_Crosses(kke.geometry, kke2.geometry)
WHERE ST_Length(kke.geometry) > 500;
```

We waited over 45 min and still didn't get a result, so we moved on.

b. *of face table with polygon table to find the polygon pip (polygon reference point) in different face bbox (add condition: area of bbox must be at least 100m2); use ST_Contains Report how long these queries take, use in psql the \timing option (probably better to do something else in between).*

```
SELECT kkf.face_id, kkf.mbr_geometry as mbr, kkf2.face_id, kkf2.pip_geometry
FROM kadnl050k_face as kkf, kadnl050k_face_geometry as kkf2
WHERE kkf.face_id != kkf2.face_id
AND ST_Contains(kkf.mbr_geometry, kkf2.pip_geometry)
```

AND ST_Area(kkf.mbr_geometry) > 100;

	face_id integer	mbr geometry
1	140378112	0103000020407100000100000050000006891
2	140378112	0103000020407100000100000050000006891
3	140378112	0103000020407100000100000050000006891
4	140115974	0103000020407100000100000050000006210
5	140115974	0103000020407100000100000050000006210
6	140115974	0103000020407100000100000050000006210
7	140115978	0103000020407100000100000050000000D7A3
8	140115978	0103000020407100000100000050000000D7A3
9	140115981	0103000020407100000100000050000000DF4F
10	140115982	0103000020407100000100000050000001B2F
11	140115984	0103000020407100000100000050000000E17A
12	140115984	0103000020407100000100000050000000E17A
13	140640292	0103000020407100000100000050000000986E
14	140640292	0103000020407100000100000050000000986E
15	140640293	0103000020407100000100000050000000508D
16	140640300	0103000020407100000100000050000000CFF7
17	140640300	0103000020407100000100000050000000CFF7
18	140640300	0103000020407100000100000050000000CFF7
19	140640300	0103000020407100000100000050000000CFF7
20	140640300	0103000020407100000100000050000000CFF7

Total rows: 1000 of 174411 Query complete 00:05:24.423

Query took approx 5 minutes 24 seconds to produce output

8. *Create spatial index (in PostgGIS this is gist index, r-tree like structure) on the relevant tables, repeat queries (especially the spatial joins).*

Query 1

```
CREATE INDEX kadnl050k_edge_geom_idx ON kadnl050k_edge USING GIST(geometry)

SELECT e1.edge_id, e2.edge_id
FROM kadnl050k_edge as e1, kadnl050k_edge as e2
WHERE ST_Crosses(e1.geometry, e1.geometry)
AND (ST_Length(e1.geometry) > 500 OR ST_Length(e2.geometry) > 500);
```

Note, generates empty output (Query complete 00:00:01.078)

	edge_id integer	edge_id integer

Query 2

```
CREATE INDEX kadnl050k_face_mbr_idx ON kadnl050k_face USING GIST(mbr_geometry);
CREATE INDEX kadnl050k_face_pip_idx ON kadnl050k_face_geometry USING
GIST(pip_geometry);

SELECT kkf.face_id, kkf.mbr_geometry as mbr, kkf2.face_id, kkf2.pip_geometry
FROM kadnl050k_face as kkf, kadnl050k_face_geometry as kkf2
WHERE kkf.face_id != kkf2.face_id
AND ST_Contains(kkf.mbr_geometry, kkf2.pip_geometry)
AND ST_Area(kkf.mbr_geometry) > 100;
```

	face_id integer	mbr geometry
1	140378112	0103000020407100000100000005000000689
2	140378112	0103000020407100000100000005000000689
3	140378112	0103000020407100000100000005000000689
4	140115974	0103000020407100000100000005000000621
5	140115974	0103000020407100000100000005000000621
6	140115974	0103000020407100000100000005000000621
7	140115978	0103000020407100000100000005000000D7A
8	140115978	0103000020407100000100000005000000D7A
9	140115981	0103000020407100000100000005000000DF4
10	140115982	0103000020407100000100000005000001B2
11	140115984	0103000020407100000100000005000000E17
12	140115984	0103000020407100000100000005000000E17
13	140640292	0103000020407100000100000005000000986
14	140640292	0103000020407100000100000005000000986
15	140640293	0103000020407100000100000005000000508
16	140640300	0103000020407100000100000005000000CFF
17	140640300	0103000020407100000100000005000000CFF
18	140640300	0103000020407100000100000005000000CFF
19	140640300	0103000020407100000100000005000000CFF
20	140640300	0103000020407100000100000005000000CFF

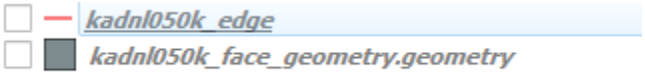
Total rows: 1000 of 174411 Query complete 00:00:01.013

Query took 1.013 seconds

9. *If not already done so, install Quantum GIS (qgis) standalone installer after download from http://qgis.org/en/site/forusers/download.html*

Done :)

10. *Make a connection to the (test) database and add PostGIS table/layer after set-up connection to database.*



11. *Display the edges longer than 1000m and the polygons larger than 10000m2*

Edges longer than 1000m

Polygons larger than 10000m2