

# Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

---

## PostGIS

Martin Schön , Bc.

AIS ID: 103121

E-mail: [xschon@stuba.sk](mailto:xschon@stuba.sk)

GitHub repozitár: <https://github.com/FIIT-DBS/zadanie-pdt-xSchon>

Predmet: Pokročilé databázové technológie

Zimný semester 2022/2023

## PostGIS

1. Import dát	3
2. Slovenské kraje	3
3. Kraje podľa veľkosti	4
4. Polygón domu	5
5. Kraj môjho domu	7
6. Aktuálna poloha v planet_osm_point	7
7. Som doma?	8
8. Vzdialenosť od FIIT	10
9. QGIS kraje a dom	11
10. Súradnice centroidu najmenšieho okresu	13
11. Úseky ciest bližšie ako 10km od hranice Pezinok – Malacky	14
12. Číslo katastrálneho územia s najdlhším úsekom cesty v mojom okrese	18
13. Oblasť Okolie_Bratislavy	20
Záver	22

## 1. Import dát

Dáta som si stiahol z uvedenej stránky a importoval do novej, lokálnej databázy pomocou knižnice [osm2pysql](#). Na vypracovanie častí zadania, pri ktorých nebolo treba použiť QGIS som použil DataGrip s jeho predinštalovanou funkcionalitou “GeoViewer”-a, ktorá dokáže zobrazit’ súradnice pre vybrané súradnicové systémy.

## 2. Slovenské kraje

Niekde sa uvádza  $x, y$  ako  $x=\text{longitude}$ ,  $y=\text{latitude}$  a niekde naopak - neexistuje jedno štandardizované poradie. Mení sa to medzi aplikáciami, zobrazeniami, je to rôzne na rôznych stránkach. Pri programovaní  $X$  zvyčajne predstavuje longitude a  $Y$  latitude (toto platí aj o nami použitých dátach). "Klasické" a teda dlhšie používané v geografii (dávno pred počítačmi) mapovanie ale bolo opačné. Preto napríklad aj na Google Maps  $X$  reprezentuje latitude a  $Y$  longitude. Toto predstavuje štandard pri zobrazovaní dát polohy na mapu - to znamená, že v počítači sú spravidla držané opačne, ako ich vo finále aplikácia zobrazuje. Pri prekreslovaní mojich výsledkov na mapu je teda (v priebehu celého tohto zadania) potrebné myslieť na túto skutočnosť.

V úlohe 2 máme transformovať všetky ways reprezentujúce hranice jednotlivých krajov do bodov. Vo finále mi tak ku každému kraju vznikne polygón bodov vo formáte [WKT](#), teda vo formáte POLYGON ((x1 y1, x2 y2, ... xn yn, x1 y1)). Polygón musí byť uzavretý cyklus, preto sa začiatočný bod opakuje na konci. Každá dvojica x y predstavuje jeden bod v súradnicovom systéme 4326, kde (ako je vyššie spomenuté) x predstavuje longitude a y latitude bodu kraja.

[illegible]

### 3. Kraje podľa veľkosti

Kraje vyberám rovnako, ako v úlohe 2, teda pomocou filtra hodnoty `admin_level` '4' z tabuľky `planet_osm_polygon`. Následne je tieto dáta potrebné pretransformovať z ich originálneho SRID 3857 na SRID 4326.

```
SELECT DISTINCT(ST_SRID(geog: way)) FROM planet_osm_polygon;
```

	st_srid
1	3857

Keby po tejto transformácii hneď vykonám `ST_Area` a získam tak veľkosť, mohla by byť skreslená až o 40%, keďže sa jedná o mercatorovo zobrazenie (ktoré spôsobuje, že rovnaká plocha sa javí väčšia, čím je bližšie k pólu). Preto je potrebné transformovať tento výsledok na `geography` type, ktorý namapuje body na guľové zobrazenie, čím získame presnejšie dáta o veľkosti. Tie sú trochu náročnejšie na vyrátanie (keďže rátajú vzdialenosť na guľi), no sú presné.

Finálny bod je predelenie veľkosti kraja miliónom, keďže prvotný výsledok je v m<sup>2</sup> a my chceme dostať hodnotu v km<sup>2</sup>.

```
SELECT name, ST_Area(geog: ST_Transform(way, 4326)::geography) / 1000000 as km2
FROM planet_osm_polygon
WHERE admin_level = '4'
ORDER BY km2;
```

Output -- S03

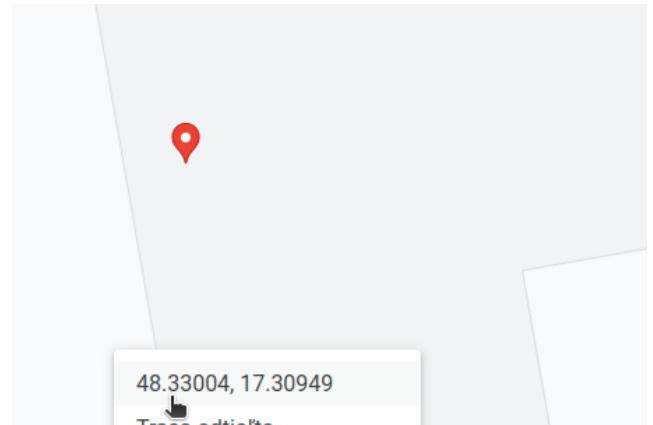
8 rows

name	km2
Bratislavský kraj	2051.660249334629
Trnavský kraj	4145.3500582742045
Trenčiansky kraj	4501.7955362039675
Nitriansky kraj	6341.240667757967
Košický kraj	6751.964233845798
Žilinský kraj	6806.886696527949
Prešovský kraj	8971.62395364117
Banskobystrický kraj	9454.546380914742

#### 4. Polygón domu

Na získanie súradníc domu som našiel dve použiteľné metódy - prvou je export súradníc z oficiálnych dát [open street map](#). Je možné exportovať dáta a z nich získať súradnice z súradnicovom systéme 4326, teda ako longitude latitude. Alternatívou je získať tieto dáta z google máp - v rovnakom formáte, po jednotlivých bodoch.

Ja som sa rozhodol pre druhú možnosť, kedy som si vybral jednotlivé body na základe google máp a potom som tieto body preniesol do databázy.



Prenesenie do databázy následne prebieha tak, že som z daných bodov vytvoril uzatvorený polygón. S pomocou funkcie ST\_GeomFromText a WKT polygónu vznikol (v mojom prípade) polygón zložený zo 6tich bodov na súradniciach približne 17.30 a 48.33. Netreba zabúdať na to, že aj v tomto prípade PostGIS číta longitude latitude opačne, ako je to v prípade Google máp.

```
ST_Transform(ST_SetSRID( geog: ST_GeomFromText('POLYGON((
17.309516058731177 48.3300009742030,
17.30967503789495 48.33002440991696,
17.30964503789495 48.330100328261695,
17.309725504163416 48.330110328261695,
17.309705504163416 48.330182479773576,
17.309471131731286 48.33015394805975,
17.309516058731177 48.3300009742030)))', srid: 4326), 3857));
```

Potom, čo som mal takto vytvorený polygón bolo potrebné uložiť ho v správnej súradnicovej sústave. V prípade way v databáze je to 3857. Samotný ST\_Transform ale nefunguje priamo - pretože pri vytváraní polygónu cez ST\_GeoFromText je súradnicová sústava nastavená na 0 - neznáma pre PostGIS. Preto som najskôr musel definovať, že sa jedná o polygón zo SRID 4326 a následne bola možná transformácia na formát vhodný pri vkladaní do DB - 3857. Celý import vyzerá takto (pridal som aj iné hodnoty pre ľahké hľadanie v DB):

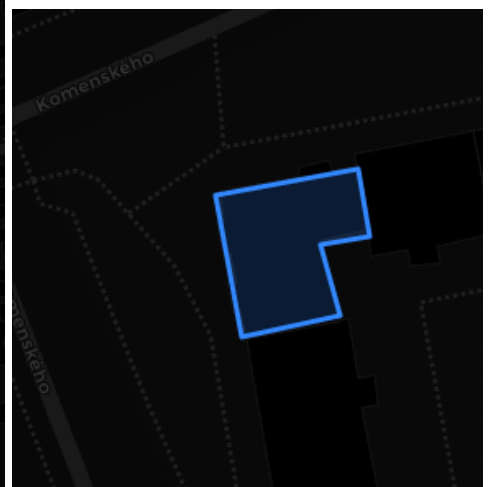
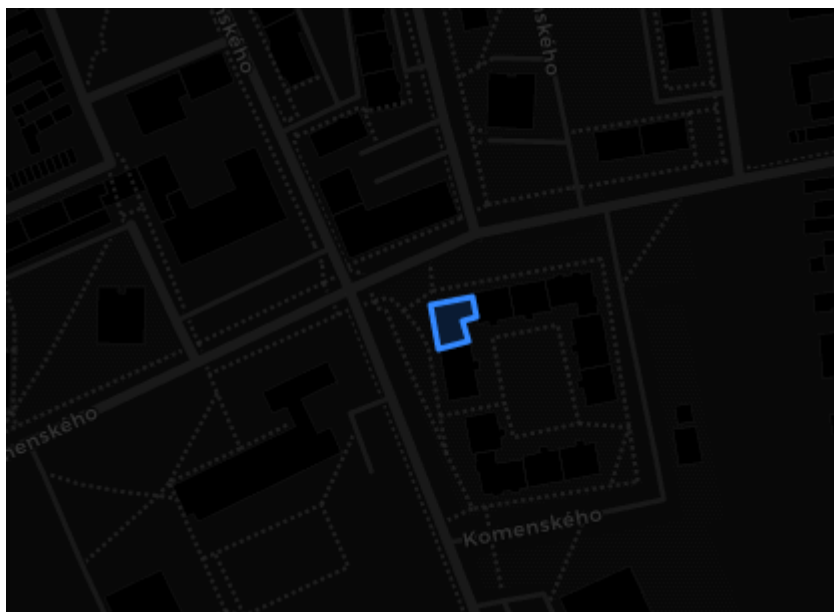
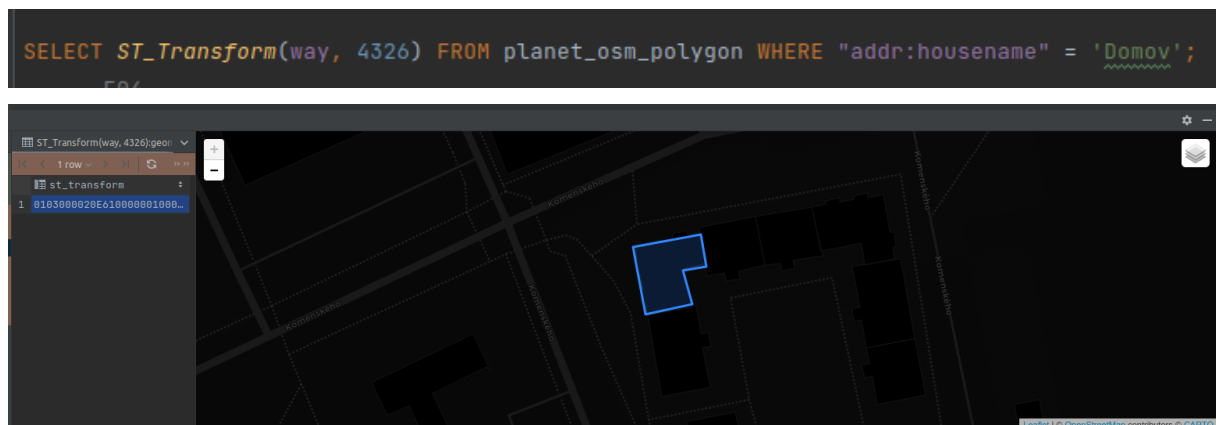
```

INSERT INTO planet_osm_polygon
  ("addr:house", "addr:house", name, "natural", z_order, tags, way)
VALUES ('Domov', 5, 'Martin Schön', 'grassland', 0, '',
  ST_Transform(ST_SetSRID( geog: ST_GeomFromText('POLYGON((
17.309516058731177 48.3300009742030,
17.30967503789495 48.33002440991696,
17.30964503789495 48.330100328261695,
17.309725504163416 48.330110328261695,
17.309705504163416 48.330182479773576,
17.309471131731286 48.33015394805975,
17.309516058731177 48.3300009742030))'), srid: 4326), 3857));

UPDATE planet_osm_polygon
SET way_area = ROUND(ST_Area( geog: way))
WHERE "addr:house" = 'Domov';

```

Pri následnom vyhľadávaní súradníc domu tak môžeme použiť tieto parametre. Pre vizualizáciu pomocou nástroja Geo Viewer v DataGripe je potrebné opäť transformovať way do súradnicového systému 4326. Potom je možné kliknutím na vrátený riadok vizualizovať polygón na mape



## 5. Kraj môjho domu

```

43  ---- S05  -----
44  ✓ SELECT kraj.name
45      FROM planet_osm_polygon AS dom
46  CROSS JOIN (SELECT way, name
47                FROM planet_osm_polygon
48                WHERE admin_level = '4') AS kraj
49      WHERE "addr:housename" = 'Domov'
50  AND ST_Within(geom1: dom.way, geom2: kraj.way);
51  ---- E05  -----
52

```

Output pdt\_postgis.public.planet\_osm\_polygon

name
1 Bratislavský kraj

Na získanie názvu kraja, v ktorom sa nachádza môj dom som použil funkciu `ST_Within`. Tá vráti `True`, ak je jeden celý objekt súčasťou druhého objektu - čo v mojom prípade platí. Vybral som si teda všetky kraje, cross joinom vytvoril všetky dvojice dom–kraj a následne v nich našiel ten, v ktorom sa nachádzam.

## 6. Aktuálna poloha v planet\_osm\_point

Cez `SELECT ST_Srid(way) FROM planet_osm_point;` som si potvrdil, že aj v `planet_osm_point` je použitá súradnicová sústava 3857. To znamená, že postup vkladania do tabuľky databázy bude podobný, ako pri vkladaní polygónu mojej domácnosti - ibaže namiesto polygónu budem vkladať bod. Cez `ST_GeometryType` som zistil, že naozaj sú geometrické dáta v tabuľke `planet_osm_point` typu `ST_Point`.

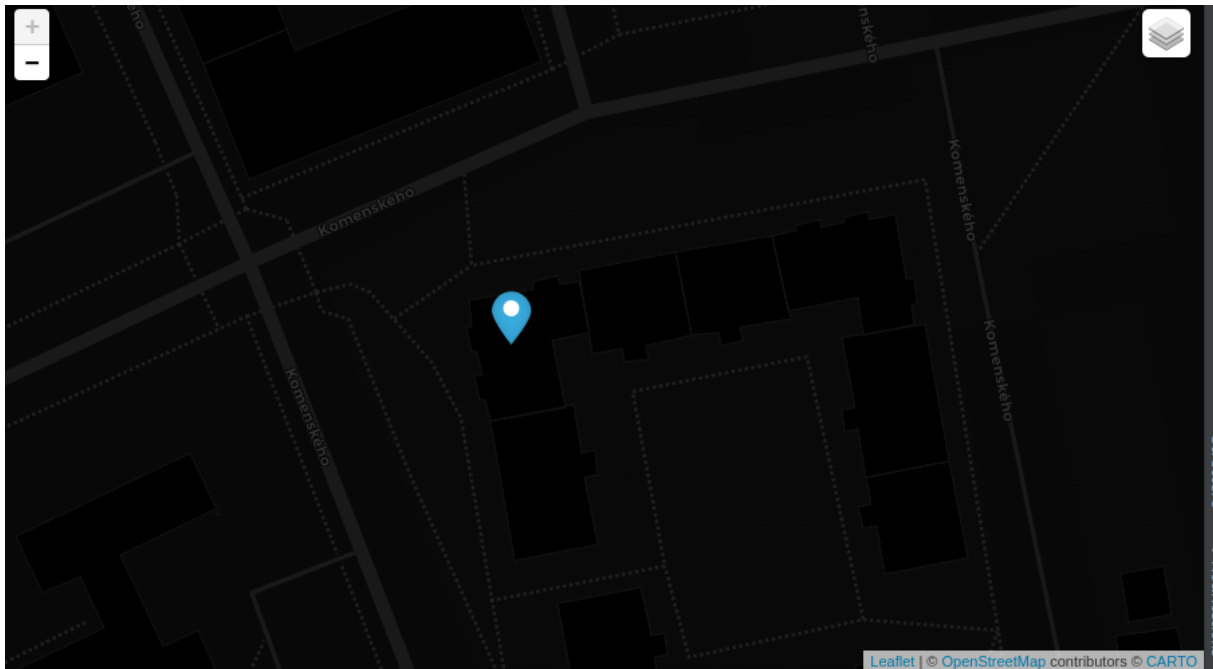
Následne som pomocou Google máp našiel svoju polohu a vložil ju do databázy veľmi podobným štýlom, ako polygón v úlohe 4 - vložil som bod z WKT, nastavil mu súradnicovú sústavu 3857 a pretransformoval do 4326.

```

INSERT INTO planet_osm_point
    (name, way)
VALUES ('Martin Schön',
        ST_Transform(ST_SetSRID(
            geog: ST_GeomFromText('POINT(17.309563464945274 48.33010486703582)'),
            srid: 4326), 3857));

```

Zobrazenie na mape:



## 7. Som doma?

```

62 ✓ SELECT home.way AS home,
63       me.way AS me,
64       ST_Within(geom1: me.way, geom2: home.way)
65 FROM planet_osm_polygon home
66 CROSS JOIN (SELECT way
67              FROM planet_osm_point
68              WHERE name = 'Martin Schön') AS me
69 WHERE name = 'Martin Schön';

```

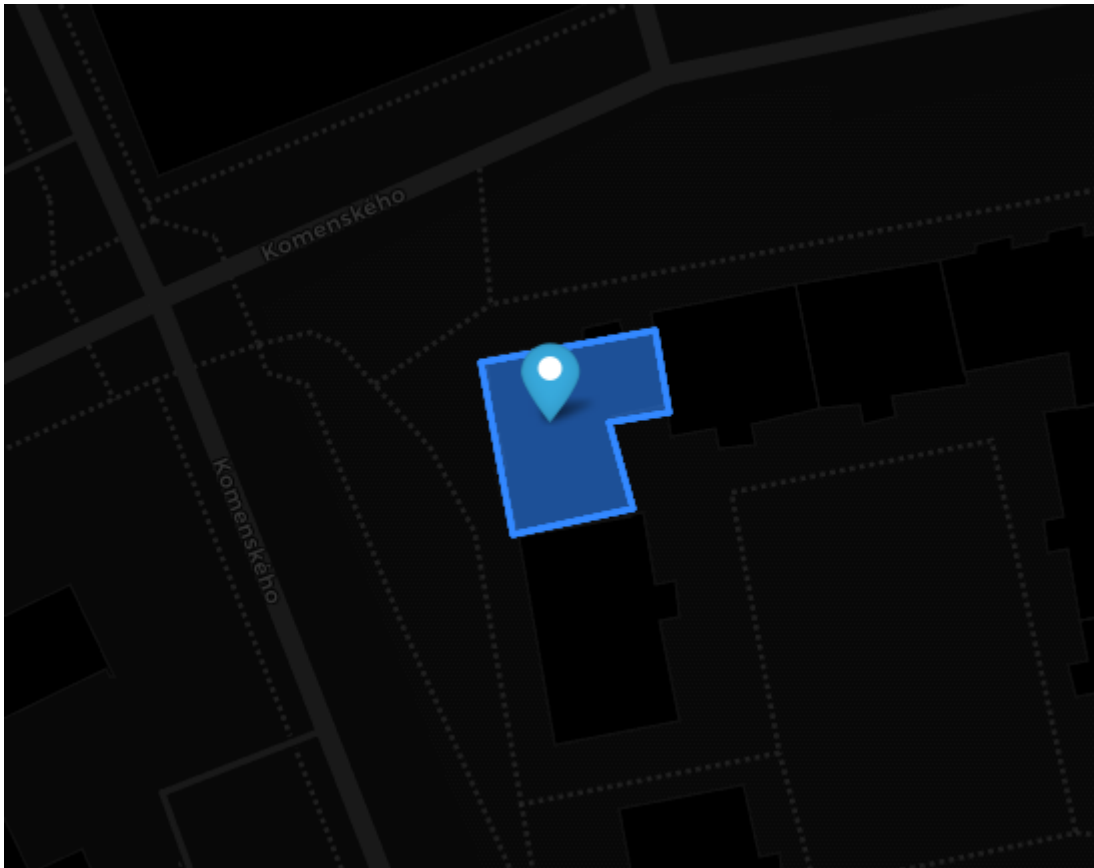
home	me	st_within
0103000020110F00000010000000...	0101000020110F000000450170CAE...	• true

Pomocou CROSS JOINu som si vytiahol súradnice mojej aktuálnej polohy a namapoval ich na súradnice môjho domu. Cez ST\_Within som následne zistil, či sa nachádzam doma - odpoveď na túto otázku je True, teda áno - nachádzam, keďže som lenivý a zadanie som začal vypracovávať až cez víkend :(.



Pre vizualizáciu v DataGripe je potrebné pri SELECTe transformovať home.way a me.way do súradnicového systému 4326.

```
SELECT ST_Transform(home.way, 4326) AS home,  
       ST_Transform(me.way, 4326) AS me,  
       ST_Within(geom(me.way), geom(home.way))
```



## 8. Vzdialenosť od FIIT

```
73 ✓ SELECT
74     ST_Distance(
75         geog1: ST_Transform(school.way, 4326)::geography,
76         geog2: ST_Transform(me.way, 4326)::geography) / 1000
77     AS distance_km
78 FROM planet_osm_polygon school
79 CROSS JOIN (SELECT way
80             FROM planet_osm_point
81             WHERE name = 'Martin Schön') AS me
82 WHERE name = 'Fakulta informatiky a informačných technológií STU';
```

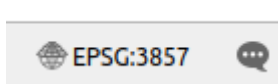
Output -- S08 -----...----- x

	distance_km ↕
1	26.31406597058

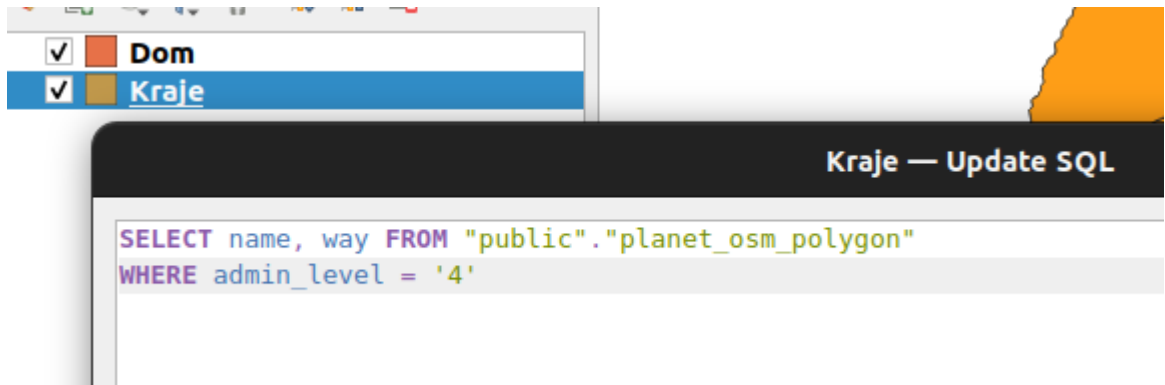
V tejto úlohe som využil vedomosti z predchádzajúcich úloh - CROSS JOINom som dostal svoj bod pri polygóne školy. Následne som pretransformoval dáta na geography, aby som získal reálnu vzdialenosť medzi nimi (podobne ako v úlohe 3). Vo finále som prerátal vzdialenosť na kilometre, aj keď to nebolo nevyhnutné - no ľahšie sa to číta.

## 9. QGIS kraje a dom

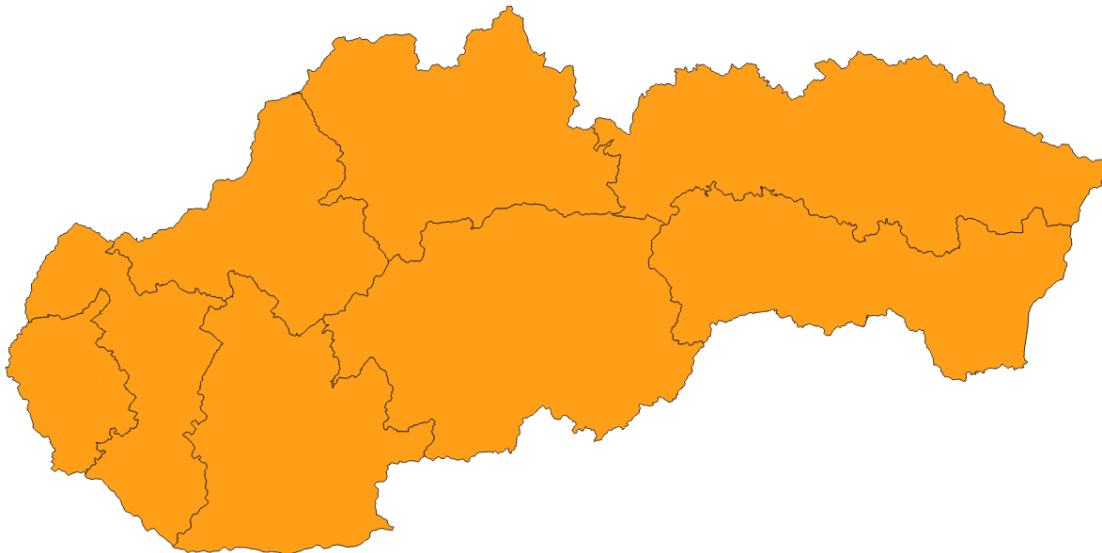
Stiahol som si QGIS, pripojil sa na lokálnu databázu cez DB Managera a dal som si vykresliť vrstvu všetkých polygónov. Potom som zistil, že si viem vytvárať a upravovať vrstvy, tak som pridal svoju vlastnú vrstvu reprezentujúcu kraje. Netreba vykonávať žiadnu transformáciu, je potrebné sa iba uistiť, že EPSG (v pravom dolnom rohu) je nastavené správne a reprezentuje SRID dát.



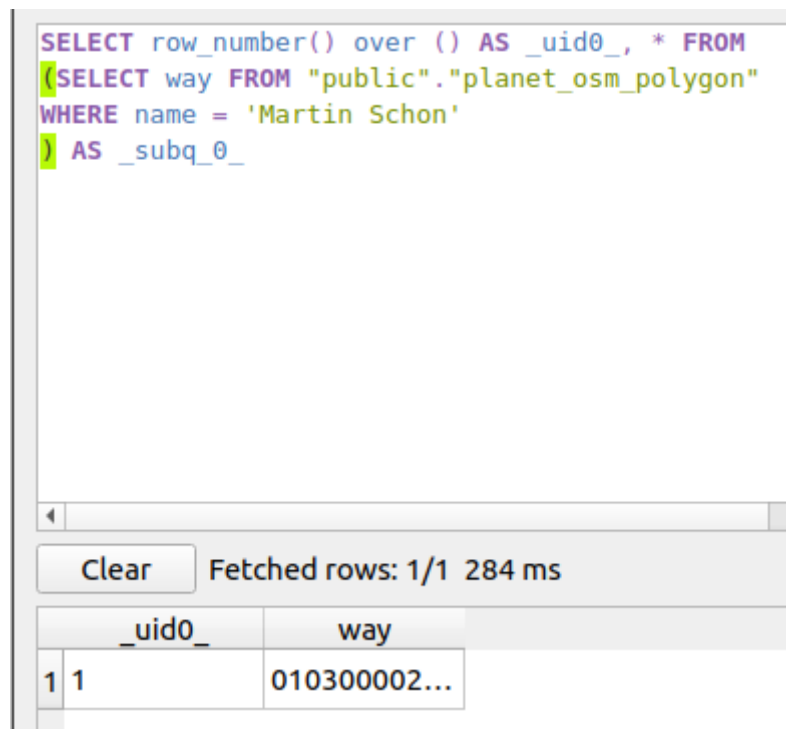
Na vykreslenie krajov som pridal vrstvu Kraje s jednoduchým SQL, ktoré ich vyberá ako v úlohe 2. Najväčší problém mi spôsobovalo to, že QGIS absolútne nezvláda, ak SQL príkaz končí bodkočiarkou, bez toho to ale mentálne ťažko prežívam ja. Inak nie je problém vykresliť si mapu krajov a obrázok. V prípade, ak sa stratím v prázdne je možné pravým kliknúť na vrstvu a zamerať sa na ňu.



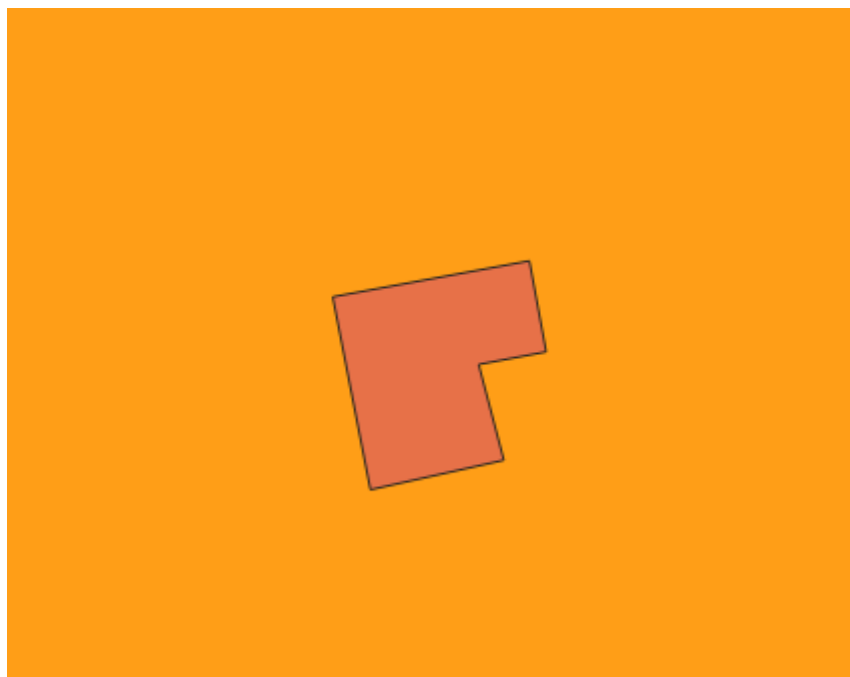
Mapa krajov:



Pri výbere môjho domu som použil query použitú aj v úlohe 4. Zvláštnosťou bolo, že QGIS si automaticky query upravil (ja som napísal iba tú v zátvorkách - SELECT way... Schon'). QGIS si moju query prečítal a zvládol ju vykresliť, ale pridal uid na unikátnu identifikáciu jednotlivých objektov (keďže môj dom neobsahuje id).



Následne som sa pokúsil nájsť svoj dom na mape Slovenska - je potrebné zoradiť layers v správnom poradí - konkrétne teda mať dom na vrchu krajov. To sa dá buď fyzickým potiahnutím vrstiev, alebo pravým klikom + Move To Top (prípadne move to bottom v prípade krajov). Každopádne potom som našiel dom na mape a tu je jeho screenshot:



## 10. Súradnice centroidu najmenšieho okresu

V tejto úlohe som opäť využil `ST_Area`, kedy som pomocou zoradenia `ORDER`om vybral najmenší kraj - Bratislavský. Následne som použil `ST_Centroid` funkciu, ktorá vyráta centroid polygónu. Tento výsledok som potom mohol transformovať na súradnicový systém 4326, z ktorého je možné vybrať `ST_Y` a `ST_X` reprezentujúci latitude a longitude súradnicu tohto bodu. Tým, že tieto dáta sú transformované, tak použitý SRID bol 4326.

```

97 ✓ SELECT name,
98         ST_Y(ST_Transform(ST_Centroid(way), 4326)) AS latitude,
99         ST_X(ST_Transform(ST_Centroid(way), 4326)) AS longitude,
100        ST_SRID( geog: (ST_Transform(ST_Centroid(way), 4326))) AS SRID
101 FROM planet_osm_polygon
102 WHERE admin_level = '4'
103 ORDER BY ST_Area( geog: way)
104 LIMIT 1;

```

Output -- S10 -----

name	latitude	longitude	srid
1 Bratislavský kraj	48.31800566738087	17.178987948813592	4326

Alternatívne by sa dalo použiť `ST_AsText`, ktoré vráti dáta vo formáte WKT Pointu, taktiež v 4326 SRID, nakoľko dáta boli transformované. Prípadne nemusím vôbec meniť dáta na AsText formát, ani transformovať - vždy dostanem centroid, hoci v poslednom prípade (bez transformácie) v SRID 3857 (takýto formát nemožno zobrazit' na GeoViewere DataGripu, ale údaj je správny).

```

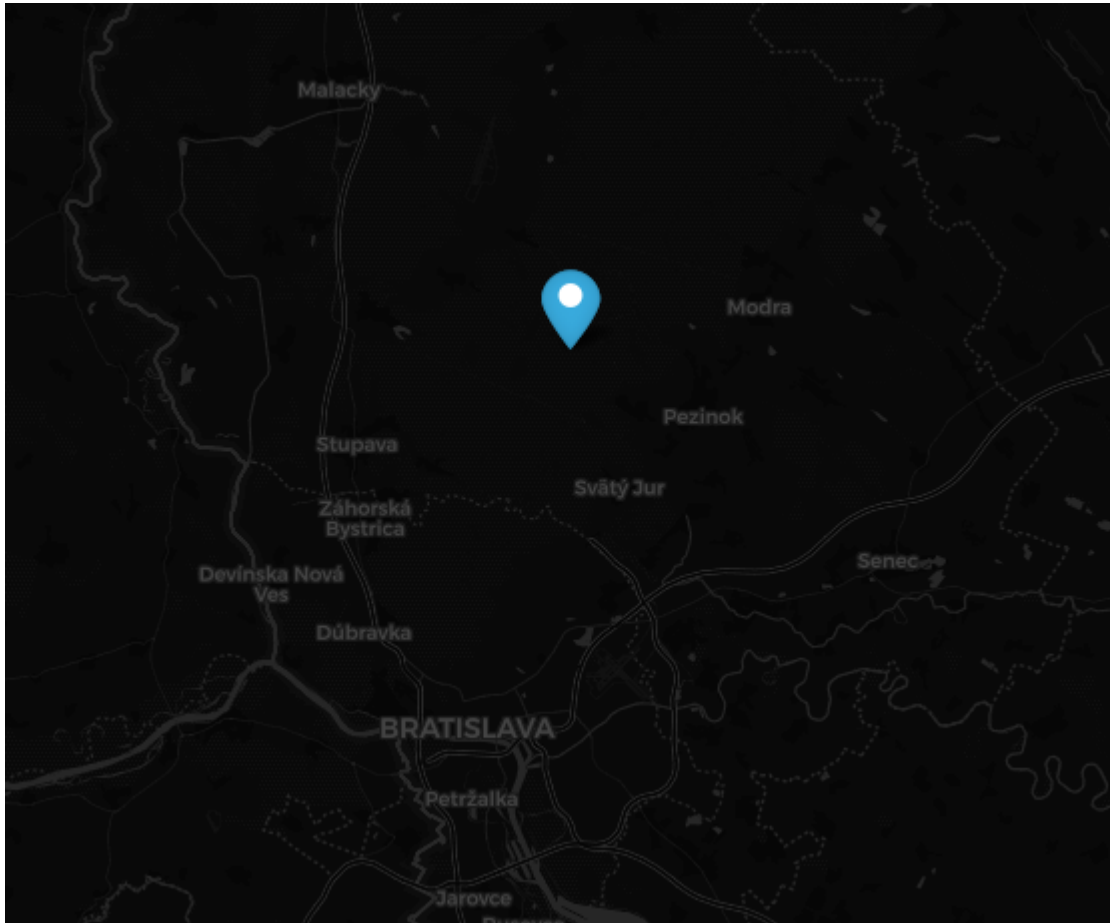
97 ✓ SELECT name,
98         ST_AsText(ST_Transform(ST_Centroid(way), 4326))
99 FROM planet_osm_polygon
100 WHERE admin_level = '4'
101 ORDER BY ST_Area( geog: way)
102 LIMIT 1;

```

Output -- S10 -----

name	st_astext
1 Bratislavský kraj	POINT(17.178987948813592 48.31800566738087)

Bod na mape sa pri rôznych prístupoch nijako nemení:



## 11. Úseky ciest bližšie ako 10km od hranice Pezinok – Malacky

```
CREATE TABLE roads_close_Pezinok_Malacky
(
  id serial PRIMARY KEY,
  osm_id BIGINT,
  way geometry(LineString, 3857),
  distance REAL,
  name VARCHAR(255),
  highway VARCHAR(255)
);
```

Najskôr som si vytvoril tabuľku `roads_close_Pezinok_Malacky`, v ktorej si budem držať výsledok tejto operácie. Bolo samozrejme možné pridať aj viac stĺpcov, či vytvárať tabuľku priamo zo `SELECTu`, ale ja do nej iba vyberám dané hodnoty - nepotrebujem všetky stĺpce. Založil som si teda tabuľku s vybranými stĺpcami.

```

INSERT INTO roads_close_Pezinok_Malacky (way, osm_id, distance, name, highway)
SELECT close_roads.way,
       close_roads.osm_id,
       ST_Distance(geog1: ST_Transform(close_roads.way, 4326)::geography, geog2: border.border::geography) / 1000 as km,
       close_roads.name,
       close_roads.highway
FROM planet_osm_roads close_roads,
     (SELECT
        ST_LineMerge(ST_Transform(ST_Intersection(malacky.way, pezinok.way), 4326)) as border
      FROM planet_osm_polygon AS malacky,
           (SELECT name, way
            FROM planet_osm_polygon
            WHERE name LIKE 'okres Pezinok') AS pezinok
           WHERE malacky.name LIKE 'okres Malacky') AS border
 WHERE close_roads.highway
 NOT IN ('proposed', 'footway', 'cycleway', 'platform', 'construction', 'path', 'service')
 AND close_roads.highway IS NOT NULL
 AND (close_roads.motorcar != 'no' OR close_roads.motorcar IS NULL)
 AND ST_Distance(geog1: ST_Transform(close_roads.way, 4326)::geography, geog2: border.border::geography) / 1000 < 10;

```

Následne túto tabuľku naplňam cestami. Vnoreným SELECTom vyberiem priesečok okresov Pezinok a Malacky - tým dostanem jednu líniu, ktorá reprezentuje hranicu (border) ktorá leží na hranici okresov Malacky a Pezinok. Používam ST\_LineMerge, aby mi z hraníc vznikol pekný jeden objekt.

```

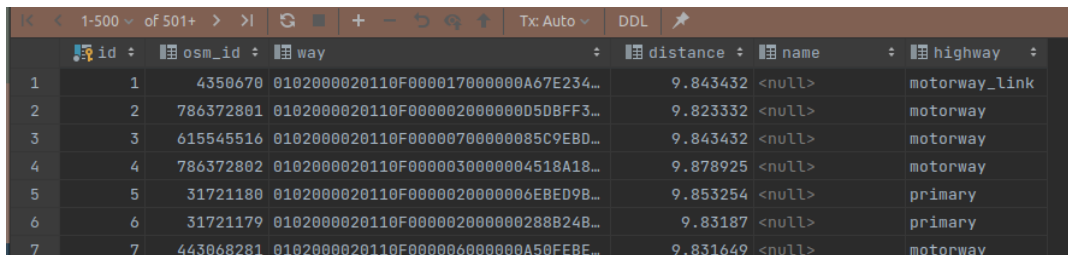
SELECT border.border
FROM (SELECT
      ST_LineMerge(ST_Transform(ST_Intersection(malacky.way, pezinok.way), 4326)) as border
    FROM planet_osm_polygon AS malacky,
         (SELECT name, way
          FROM planet_osm_polygon
          WHERE name LIKE 'okres Pezinok') AS pezinok
         WHERE malacky.name LIKE 'okres Malacky') as border;

```



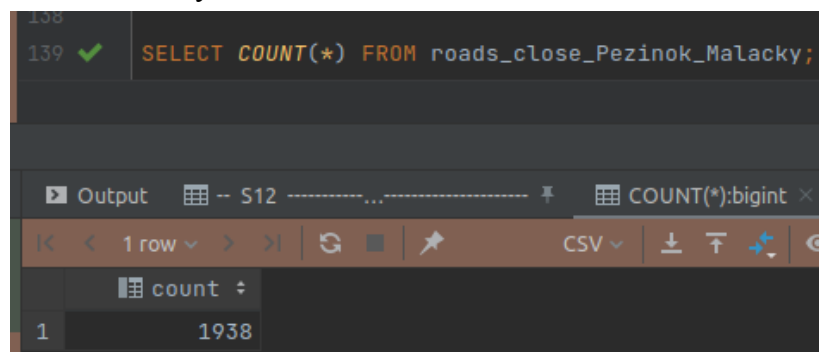
Následne z tabuľky `planet_osm_roads` vyberiem informácie o cestách. Keďže v tejto tabuľke sa nachádzajú aj iné čiary, ako tie, reprezentujúce cesty pre vozidlá, je potrebné ich filtrovať. To vykonávam pomocou stĺpca `highway` - ten v sebe drží informáciu o účele zapísanej geografickej hodnoty. V prípade `osm` existuje [highway dokumentácia](#), ktorá opisuje význam jednotlivých zameraní. Stačilo mi teda vymazať niektoré čiary - pre chodcov, vodu,... Následne vymažem aj tie, ktoré majú `highway` rovné hodnote `NULL`, ale nemajú `motorway` rovné **no** (`NULL` je v pohode). Nakoniec vyberiem vzdialenosť medzi borderom a cestami pomocou `ST_Distance` nad 4326 geography cestami a vyberiem tie, ktoré sú bližšie, ako 10km.

Do databázovej tabuľky potom vložím údaje o vzdialenosti od hranice, linku cesty (v SRID 3857 na vloženie do databázy) a doplnujúce informácie. Súbor s výsledkami je možné nájsť v prílohe pod menom `z03_11_schon.csv` vo formáte:



	id	osm_id	way	distance	name	highway
1	1	4350670	0102000020110F0000017000000A67E234...	9.843432	<null>	motorway_link
2	2	786372801	0102000020110F0000002000000D5DBFF3...	9.823332	<null>	motorway
3	3	615545516	0102000020110F000000700000085C9EBD...	9.843432	<null>	motorway
4	4	786372802	0102000020110F00000030000004518A18...	9.878925	<null>	motorway
5	5	31721180	0102000020110F00000020000006EBED9B...	9.853254	<null>	primary
6	6	31721179	0102000020110F0000002000000288B24B...	9.83187	<null>	primary
7	7	443068281	0102000020110F0000006000000A50FEBE...	9.831649	<null>	motorway

Vo finále som dostal 1938 takýchto ciest.



```
138
139 ✓ SELECT COUNT(*) FROM roads_close_Pezinok_Malacky;
```

Output S12 COUNT(\*):bigint

count
1938



Cesty vyzerajú nasledovne:



## 12. Číslo katastrálneho územia s najdlhším úsekom cesty v mojom okrese

Najskôr som si stiahol [dáta](#), ktoré som si nainportoval do databázy pomocou command line príkazu ogr2ogr. Tie mi vytvorili v databáze 5 nových tabuliek: obec\_0, okres\_0, sr\_0, kraj\_0, ku\_0. Pre mňa zaujímavá tabuľka je ku\_0, keďže reprezentuje katastrálne územia. Katastrálne územia môjho okresu (Pezinok) vyzerajú takto:



V tejto úlohe som mal vybrať názov a id katastra, v ktorom sa nachádza najdlhší úsek cesty. Cesty v tomto prípade vyberám rovnako, ako v predchádzajúcom príklade - odstránim nemotoristické čiary z tabuľky planet\_osm\_roads. Okrem týchto ciest potrebujem získať informácie o katastroch - z novo vlozenej tabuľky ku\_0 vyberiem katastre nachádzajúce sa v Pezinskom okrese. O katastroch si držím informáciu o id a mene, z id5 a nm5 keďže sa hodnoty v nich zhodujú s oficiálnym [zoznamom](#) katastrov na Slovensku a teda dobre reprezentujú jednotlivé katastrálne územia.

```
SELECT cadastre.idn5 AS cadastre_id,
       cadastre.nm5 AS cadastre_name,
       ref AS road_num,
       ST_Length( geog: ST_Intersection(ST_Transform(roads.way, 4326)::geography,
                                         ST_Transform(cadastre.way, 4326)::geography)) / 1000 as len_km,
       ST_Transform(ST_Intersection(ST_Transform(roads.way, 4326)::geography,
                                         ST_Transform(cadastre.way, 4326)::geography)::geometry, 4326) AS road_visualisation,
       ST_Transform(cadastre.way, 4326) AS cadastre_visualisation
FROM
  planet_osm_roads roads,
  (SELECT idn5, nm5, ST_Transform(shape, 3857) AS way FROM ku_0 WHERE lau1 = 'Pezinok') AS cadastre
WHERE ST_Intersects( geog1: roads.way, geog2: cadastre.way)
AND roads.highway
NOT IN ('proposed', 'footway', 'cycleway', 'platform', 'construction', 'path', 'service')
AND roads.highway IS NOT NULL
ORDER BY ST_Length( geog: ST_Intersection(roads.way, cadastre.way)) DESC
LIMIT 1;
```

Okrem toho si, samozrejme, vyberiem aj geografické znázornenie katastra, ktoré je potrebné pretransformovať, nakoľko je v SRID 5514. Potom môžem pomocou ST\_Length vybrať dĺžku prieseku (ST\_Intersection) jednotlivých ciest a katastrov. Z nich potom vyberiem najdlhší úsek. Používam aj podmienku ST\_Intersects medzi cestami a katastrami, keďže to pomôže zrýchliť proces automatickým vyradením dvojíc ciest a katastrov, ktoré nie sú pri sebe. Vo finále mi tak na porovnanie ostanú len nenulové dĺžky, vyberiem cez ORDER BY a LIMIT najdlhší priesek. Potom vyberiem dĺžku a geografiu na vizualizáciu.

Výsledok, ktorý dostanem z mojej (vyššie ukázanej) query a jeho vizualizácia:

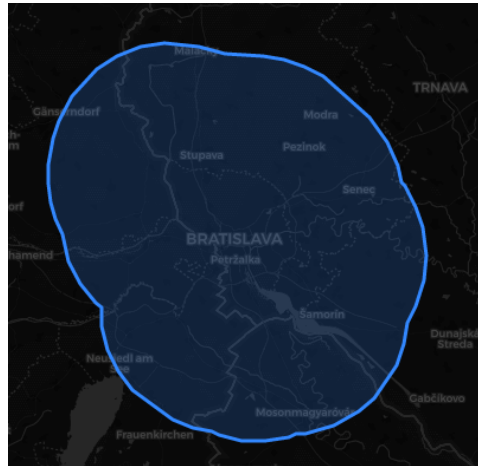
	cadastr_id	cadastr_name	road_num	len_km	road_vis...	cadastr_visualisation
1	846163	Pezinok	503	4.0004480605081865	01020000A0E6100...	01060000A0E6100000010000000103000...



### 13.Oblasť Okolie\_Bratislavy

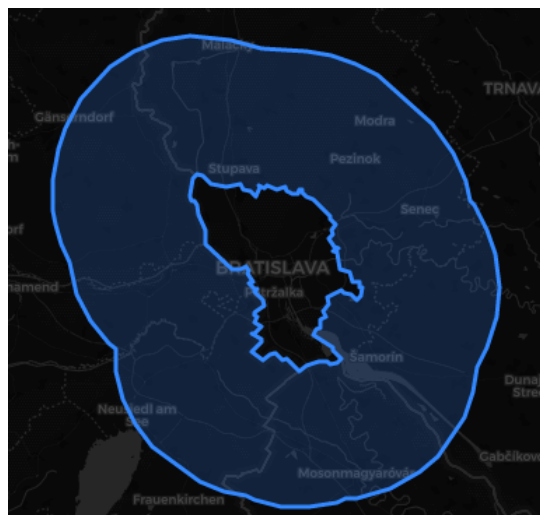
V tejto úlohe si najskôr vyberiem oblasť 20km okolo Bratislavy (od okraja mesta). Vykonám tak pomocou ST\_Buffer-u, ktorý v SRID 4326 prijíma argument metrov (20000m), okolo ktorých má vytvoriť polygón oblasti okolo daného objektu (v mojom prípade okolo Bratislavy). Všetky polygóny v tejto úlohe použité z databázy pochádzajú z tabuľky planet\_osm\_polygon

```
SELECT ST_Transform(ST_Buffer(ST_Transform(way, 4326)::geography, 20000)::geometry, 3857) AS way
FROM planet_osm_polygon
WHERE name = 'Bratislava' AND admin_level = '6') AS okolie,
```



Následne odstránim z tohto polygónu samotnú Bratislavu - pomocou ST\_Difference medzi okolím a polygónom Bratislavy. Tu je potrebné dbať na správne poradie polygónov.

```
SELECT
  ST_Transform(
    ST_Difference( geom1: okolie.way, geom2: bratislava.way), 4326) AS Bratislava_Okolie
FROM
  (SELECT ST_Transform(ST_Buffer(ST_Transform(way, 4326)::geography, 20000)::geometry, 3857) AS way
   FROM planet_osm_polygon
  WHERE name = 'Bratislava' AND admin_level = '6') AS okolie,
  (SELECT way
   FROM planet_osm_polygon
  WHERE name = 'Bratislava' AND admin_level = '6') AS bratislava;
```



A nakoniec vyberiem iba tú časť okolia, ktorá sa pretína so Slovenskom. Tú vyberiem z tabuľky a potom cez ST\_Intersection vyberiem iba tú časť, ktorá sa nachádza na Slovensku v okolí Bratislavy. Takáto intersection mi vráti Collection, ktorú rozbalím a vyberiem z nej polygón (3). Z tohto výsledného polygónu potom získam jeho rozlohu v km2, ako aj jeho vizualizáciu.

```
SELECT ST_Area( geog: ST_Transform(
  ST_CollectionExtract(
    ST_Intersection(slovakia.way,
      ST_Difference( geom1: okolie.way, geom2: bratislava.way)), 3), 4326
    )::geography) / 1000000 as km2,
  ST_Transform(
    ST_CollectionExtract(
      ST_Intersection(slovakia.way,
        ST_Difference( geom1: okolie.way, geom2: bratislava.way)), 3), 4326
    ) AS Bratislava_Okolie
FROM
  (SELECT ST_Transform(ST_Buffer(ST_Transform(way, 4326)::geography, 20000)::geometry, 3857) AS way
  FROM planet_osm_polygon
  WHERE name = 'Bratislava' AND admin_level = '6') AS okolie,
  (SELECT way
  FROM planet_osm_polygon
  WHERE name = 'Bratislava' AND admin_level = '6') AS bratislava,
  (SELECT way
  FROM planet_osm_polygon
  WHERE admin_level = '2'
  AND name = 'Slovensko') as slovakia;
```

Rozloha tejto oblasti je 1480 km2 a je znázornená na obrázku:

	km2	bratislava_okolie
1	1480.1274586081468	0106000020E61000000100000001...



## Záver

Zadanie som vypracoval samostatne, s využitím PostgreSQL verzie 15.0 a PostGIS verziou 3.3.1. V prílohe sa nachádza .csv súbor s výsledkami z úlohy 11, rovnako ako aj qgz súbor (spustiteľný v programe QGIS) z úlohy 10. Taktiež je priložený .sql súbor, v ktorom sa nachádzajú mnou použité SQL príkazy v ich finálnej verzii. Všetky tieto súbory je možné nájsť aj na [GitHube](#) uvedenom na titulnej strane. Ďakujem za čítanie dokumentácie mojej práce :).