

# PDT ZADANIE 3

## PROTOKOL

Filip Vida

30.10. 2022

GITHUB: [https://github.com/xvidaf/PDT3\\_Vida](https://github.com/xvidaf/PDT3_Vida)

Queries sa tiež nachádzajú v súbore dpt3\_queries.txt

2.

Keď chceme získať koordináty polygonu, vieme tu urobiť zo stĺpca way. Priložené sú 2 riešenia, druhé vypíše koordinát centrálneho bodu v polygóne, zatiaľ čo prvé vypíše body z ktorých sa polygón skladá. Way je potrebné preformátovať na srid 4326, aby tieto koordináty boli reálne skontrolovateľné, napr. na google maps.

```
SELECT osm_id,name, way, st_astext(st_transform(way,4326)) FROM
planet_osm_polygon
WHERE admin_level = '4'
```


```
SELECT osm_id,name, way, st_x(st_centroid(st_transform(way,4326))) AS "Longitude", st_y(st_centroid(st_transform(way,4326))) AS "Latitude" FROM
planet_osm_polygon
WHERE admin_level = '4'
```

Oba ale obsahujú rovnaký výstup pre way, ktorý si môžeme nakresliť na mape.

Data outputMessagesGeometry Viewer xNotifications

+

-



Total rows: 8 of 8Query complete 00:00:00.416

Data outputMessagesGeometry Viewer xNotifications

osm\_id

bigint

name

text

way

geometry

1	-388269	Žilinský kraj	0103000020110F000001000000E13500005CD2000F6F1F3F4191CA038B2C2458414D36D9D9761F3F41EA783604292458419E1D41F4AD1F3F41CEE8DD441424584107F2CF37C41F3F41ACF73
2	-388267	Trenčiansky kraj	0103000020110F00000100000048390000F0230C19D3793D41916CAC1103CB5741F04A72B3DA793D4146DD978300CB5741208659B5377A3D41D6F3132DE3CA574104EF6018667A3D416B5A
3	-388266	Trnavský kraj	0103000020110F000001000000684200003509362C6FC33C41D5CAC96BE1AD5741C6C5339E6FC33C416221681DCFAD57419B88206773C33C414E469AF7BBAD574146FA016A79C33C417DAE
4	-388268	Nitriansky kraj	0103000020110F00000100000004370000D95D343A0E133E41EA72DFF1A82457412560F51918133E4128888363A5245741717F76D9E3133E4105F78CAC65245741B94638FF6D143E41A5EE0D
5	-388270	Banskobystrický kr...	0103000020110F000001000000644D00009F8709784C633F41F68D0B3455A557410CDF03954D633F411743C03154A55741CFEBF9E681633F41B44C70864AA5574131BAB177C2633F412921F4
6	-388265	Bratislavský kraj	0103000020110F0000010000008C1C0000C573FE09C6973C4198F047E5C68957414390A066C9973C4179E0F3ABB8957419B5D516DD1973C41B072E9D8AD8957416D673297E1973C410A99C
7	-388272	Košický kraj	0103000020110F00000100000098470000CDEBA5ADD12341412E3E96EEFCC457419408EF79DC234141335CB006EAC4574177D09F0FEC234141A029DA65CFC45741FA821B51E0234141E3FB4

Total rows: 8 of 8Query complete 00:00:00.472Ln 5, Col 1

Veľkosť plochy polygónu vieme zistiť pomocou funkcie `st_area` z stĺpca `way`. Takisto ako pri predošlej úlohe si `way` transformujeme na `srid 4326`, čím získame rozlohu v metroch. Následne  $m^2$  prekonvertujeme na  $km^2$ . Výsledky zoradíme.

Data output Messages Geometry Viewer x Notifications

4.

```
INSERT INTO planet_osm_polygon
(name, way, planet_id)
VALUES ('Filipov dom', ST_MakePolygon(st_transform(ST_GeomFromText
('LINESTRING(18.270149786880466 47.868199134993915, 18.270286579457487 47.8683061493461, 18.270431418736955 47.86821532894873,
18.270295967188563 47.8681127671495, 18.270149786880466 47.868199134993915)'),4326,3857)),420420);
```

Data output Messages Geometry Viewer x Notifications

[illegible]

```

22 SELECT *
23 FROM planet_osm_polygon
24 WHERE name LIKE 'Filipov dom';

```

Data output Messages Geometry Viewer × Notifications

5.

Kraj v ktorom sa dom nachádza zistíme tak, že si vyselektujeme kraje, a pridáme podmienku, že nami vytvorený polygon sa v danom polygóne kraja nachádza pomocou funkcie ST\_contains.

```

26 SELECT osm_id,name, way FROM
27 planet_osm_polygon
28 WHERE admin_level = '4' and ST_Contains(way, (SELECT way as filip_way
29 FROM planet_osm_polygon
30 WHERE name = 'Filipov dom'))

```

Data output Messages Geometry Viewer × Notifications

	osm_id bigint	name text	way geometry
1	-388268	Nitriansky...	0103000201110F0000010000004370000D95D343A0E133E41EA72DFF1A82457412560F5191B133E412888363A5245741717F76D9E3133E4105F78C65245741B94638FF6D143E41A5EE0D714924574150DDC49B02153E41355B05C53...

6.

Point pridáme veľmi podobne ako polygón, ale použijeme len jedni koordináty a funkciu st\_pointFromText.

```

INSERT INTO planet_osm_point
(name, way, osm_id)
VALUES ('Filipova lokacia',st_transform(ST_PointFromText('POINT(17.04151397658695 48.15542176441618)', 4326),3857) ,420420);

```

```

36 SELECT *
37 FROM planet_osm_point
38 WHERE name = 'Filipova lokacia'

```

Data output Messages Geometry Viewer × Notifications

	osm_id bigint	access text	addr:house name text	addr:house number text	addr:interpolation text	admin_level text	aerialway text	aeroway text	amenity text	area text	barrier text	bicycle text	brand text	bridge text	boundary text	building text	capital text	const text
1	420420	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]

Data output Messages Geometry Viewer × Notifications

Total rows: 1 of 1 Query complete 00:00:00.284

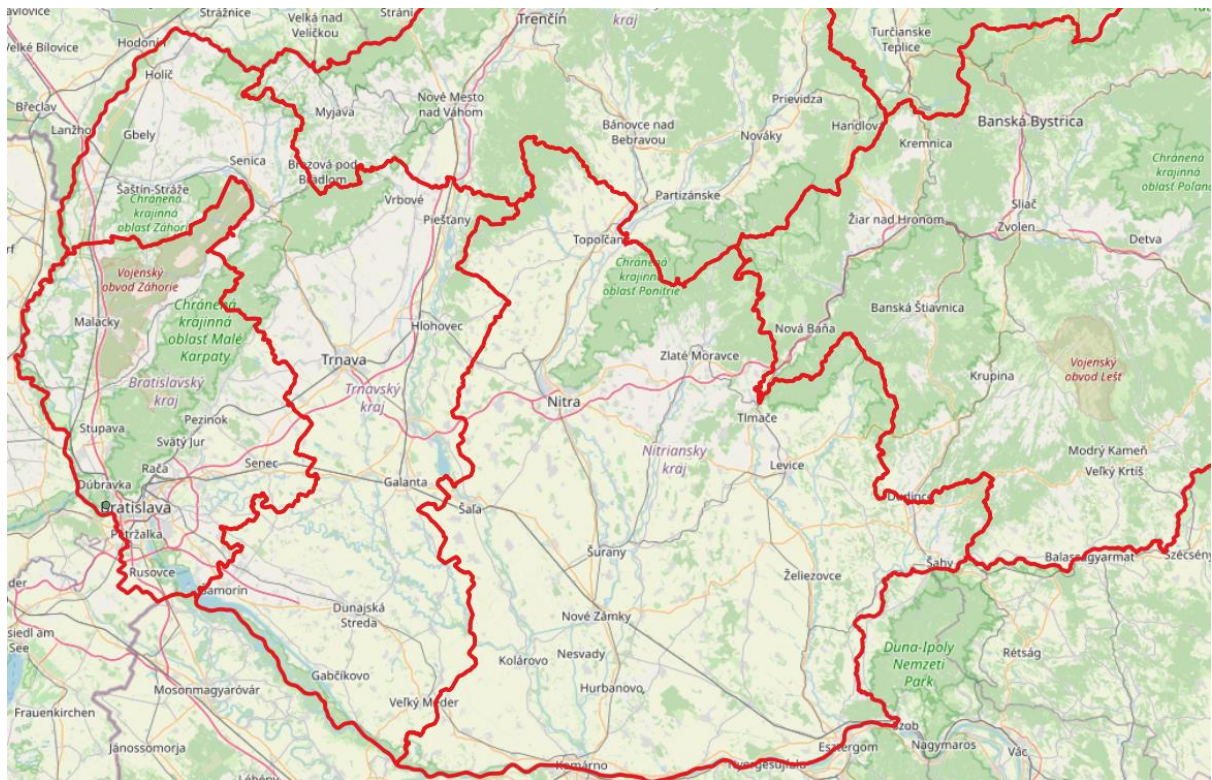
Na toto vieme tiež využiť už použitý `st_contains`, pomocou ktorého vidíme že nie, niesom akurát doma.



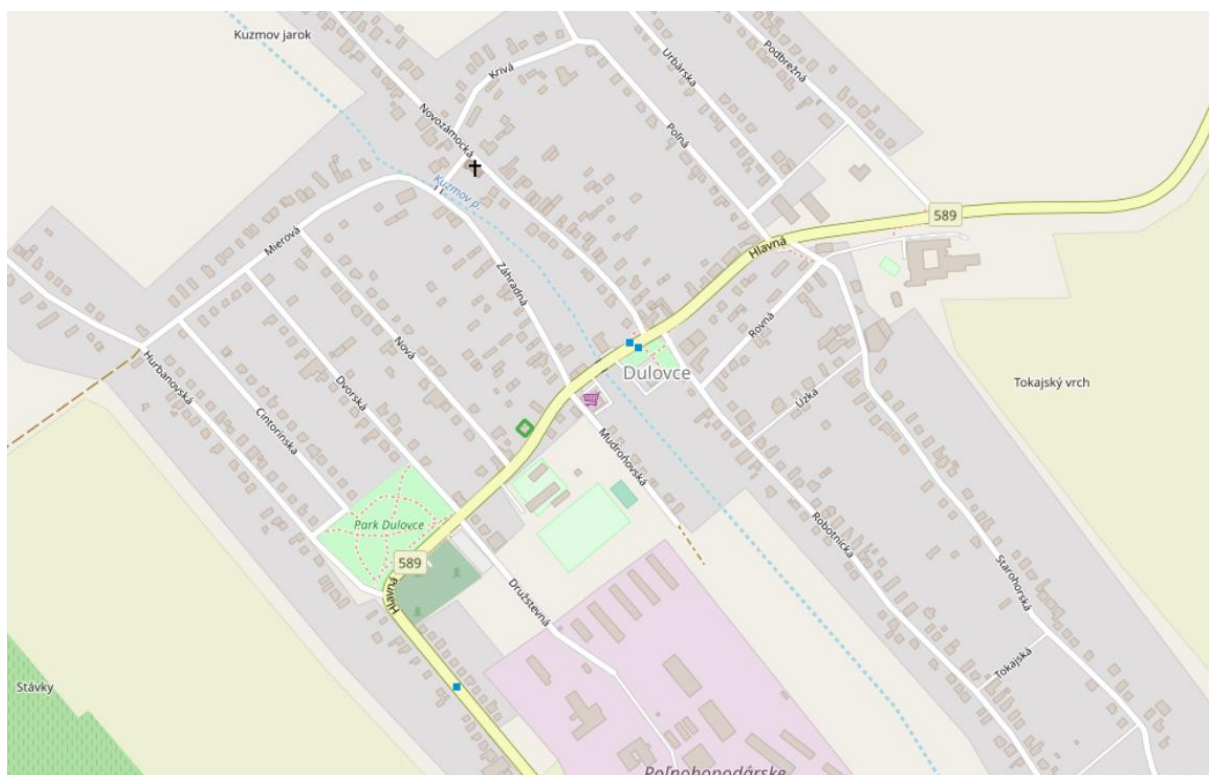
Vidíme, že sa nachádzam okolo 2 kilometrov od fakulty.



S mapou ako podklad (dosť neprehľadné)

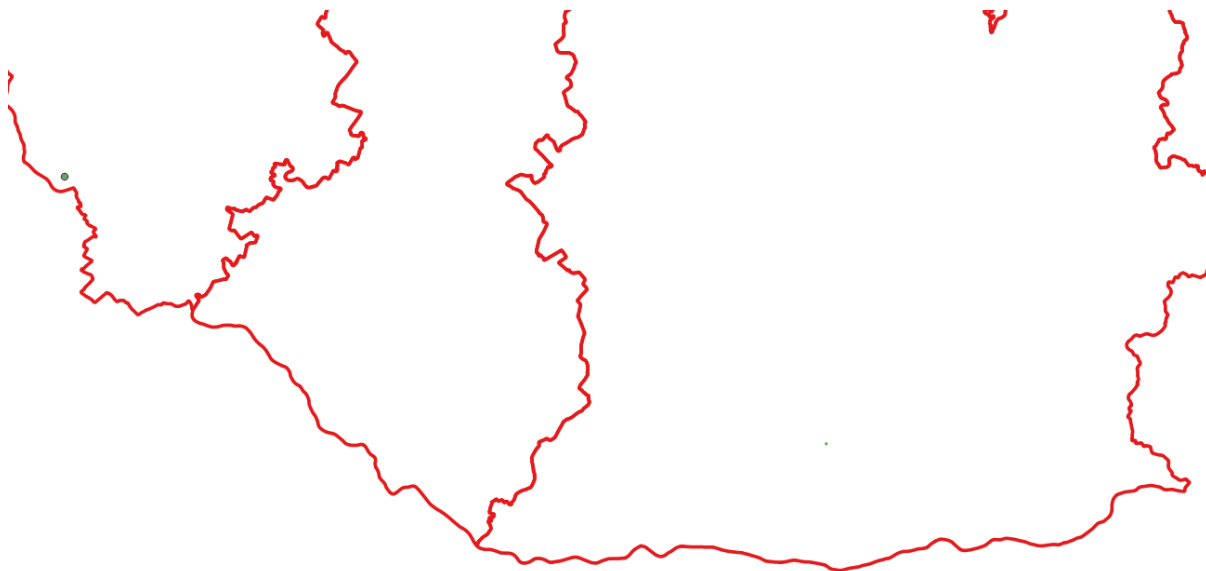


Dom s mapou ako podklad





Tu vidíme bližší náhľad, zelený point je moja aktuálna lokácia, zelený polygón je moje miesto bydliska.



10.

Plošne najmenší okres is zistíme pomocou 2 subqueries. Najprv si vyberieme všetky polygóny, ktoré majú admin\_level = 8, a tiež obsahujú v mene okres, aby sa vyfiltrovali nežiadané záznamy. Z týchto výsledkov následne vypíšeme ich rozmer, pomocou st\_area, do ktorého pošleme transformovaný way, toto nám vráti rozlohu v m<sup>2</sup>, tak to ešte prekonvertujeme na km<sup>2</sup> a usporiadame ich, a vynerieme najmenší. Z tohto následne vyberieme súradnice ako už raz bolo spomínané.

```
30 SELECT name, way, size, st_x(st_centroid(st_transform(way,4326))) AS "Longitude", st_y(st_centroid(st_transform(way,4326))) AS "Latitude", ST_SRID(st_transform(way,4326)) as srid
31 FROM
32 (SELECT name, way, st_area(st_transform(way,4326), False)/1000000 as size FROM (
33 SELECT name, way
34 FROM planet_osm_polygon
35 WHERE admin_level = '8') as okresy_a_spol
36 where name like '%okres%')
37 ORDER by size ASC
38 LIMIT 1) as smallest
39
```

name	way	size	Longitude	Latitude	srid
okres Bra...	0103000020110...	9.56871229618822	17.0994108837330	48.1509790762363	4326

11.

Do tabuľky vložíme všetky cesty, ktorých aspoň jeden bod je menej ako 10km od Hranice, ktorú sme získali pomocou st\_intersection.

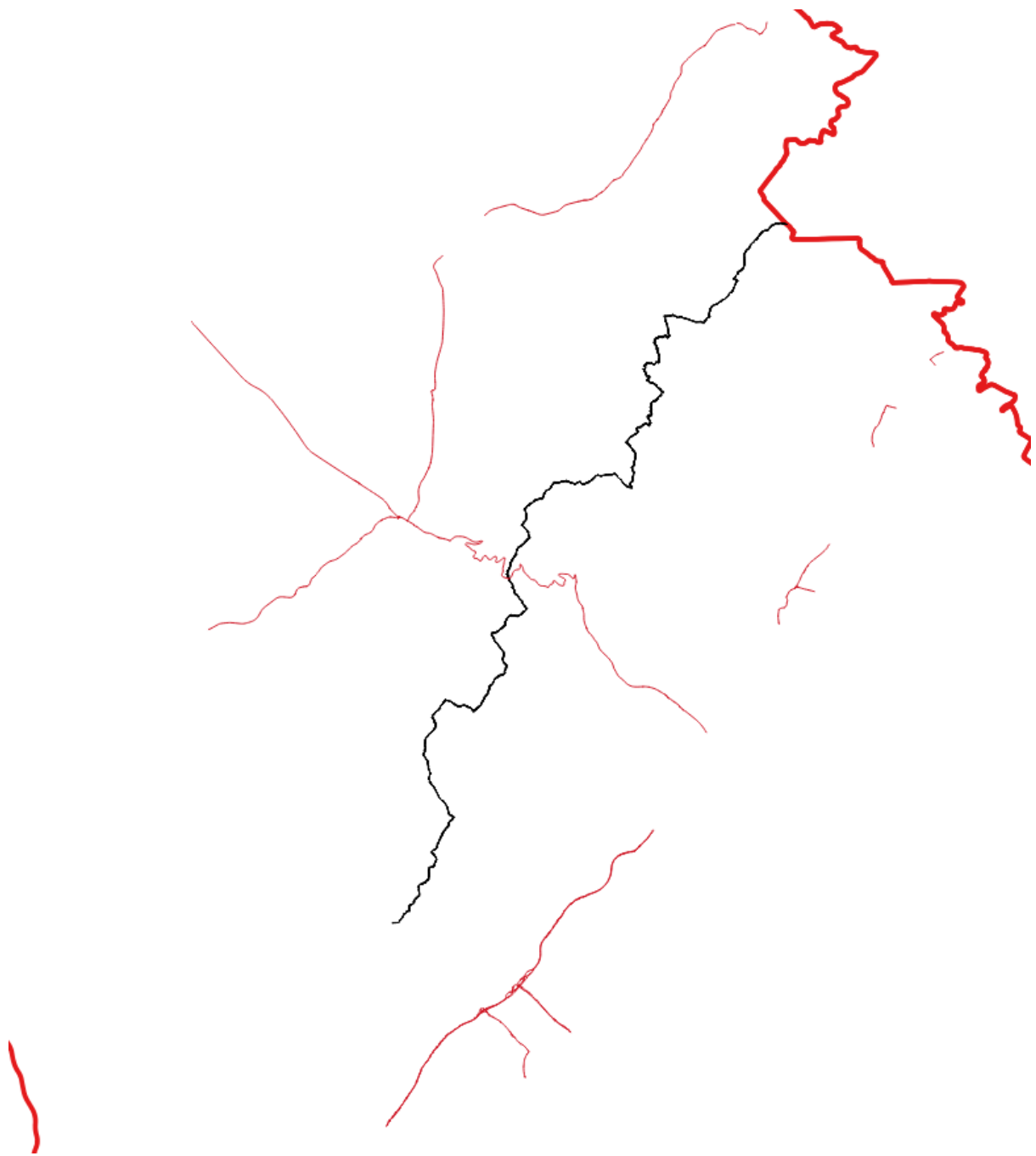
Hranicu si nájdeme pomocou ST\_Intersection, následne hľadáme všetky cesty, ktoré spĺňajú podmienku, že sú vzdialené menej ako 10km. Cesty vyfiltrujeme, aby neobsahovali Hranice a chodníky.

```
1 CREATE TABLE roads_near_hranica (LIKE planet_osm_roads INCLUDING ALL);
2
3 INSERT INTO roads_near_hranica
4 SELECT * FROM planet_osm_roads
5 WHERE ST_Distance((SELECT ST_Intersection((select way
6 FROM planet_osm_polygon
7 WHERE name = 'okres Malacky'),(select way
8 FROM planet_osm_polygon
9 WHERE name = 'okres Pezinok'))),planet_osm_roads.way) <= 10000 AND boundary IS NULL and highway NOT LIKE ALL(ARRAY['path','footpath','track']);
```

INSERT 0 365

Query returned successfully in 9 secs 750 msec.

Aspoň jeden bod cesty je menej ako 10km od hranice.



12.

Katastrálne územia boli nainportované do tabuľky test. Najprv si vyberieme všetky katastre a cesty, ktoré sa nachádzajú v komárňanskom okrese. Následne si z týchto dvoch tabuľiek spájame tie, ktoré sa intersektujú pomocou ST\_Intersects. Z takto povyberaných dvojíc si zistíme dĺžku cesty v katastrálnom území, a vyberieme najdlhšie.

Priradené sú dve možnosti riešenia, keďže sa mi zdalo že v tabuľke roads niesu len roads, a pokúsil som sa to vyfiltrovať.



Vyfiltrované cesty:

Ak necháme, že highway môže byť null, výsledok sa nachádza v katastri Čalovec, no nevieme presne o čo ide.

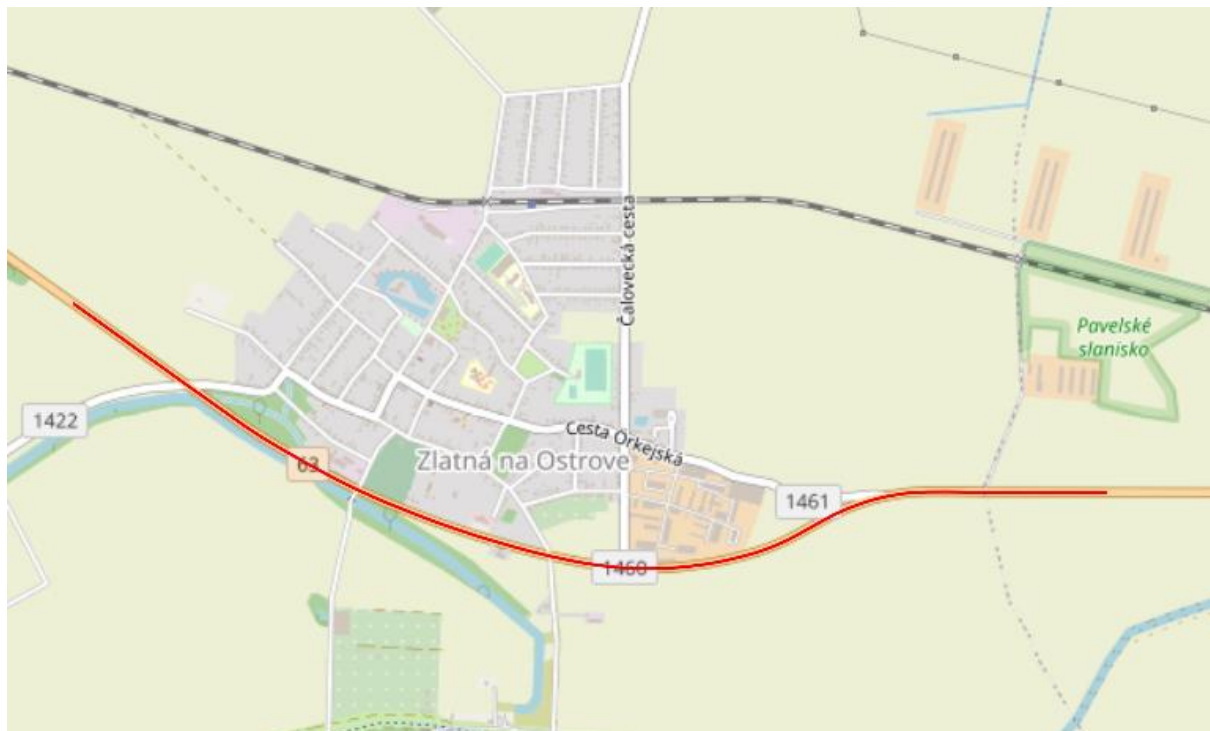
```
14 SELECT ogc_fid, objectid, name, nm5 as meno_kataster, ST_Length(ST_Intersection(ST_Transform(way,3857),ST_Transform(wkb_geometry,3857))) as dlzka_cesty, way as road_way
15 FROM
16 (SELECT +
17 FROM planet_osm_roads
18 WHERE ST_Intersects(way, (SELECT way
19 FROM planet_osm_polygon
20 WHERE admin_level = '8' and name = 'okres Komárno')))) as cesty,
21 (select +
22 FROM test
23 WHERE ST_contains((SELECT way
24 FROM planet_osm_polygon
25 WHERE admin_level = '8' and name = 'okres Komárno'),ST_Transform(wkb_geometry,3857))) as katastre
26 WHERE ST_Intersects(ST_Transform(wkb_geometry,3857),ST_Transform(way,3857)) AND boundary is NULL
27 ORDER BY dlzka_cesty DESC
28 LIMIT 1
```

ogc_fid	objectid	name	meno_kataster	dlzka_cesty	road_way
integer	numeric (20)	text	character varying (50)	double precision	geometry
1	359	3918	[null]	Čalovec	7610.843917646591 0102000020110F00007E000000C30235453F943E4157BFE9D24130574111AF50FB09943E41D8522D0337305741983E23E5C9933E41D463F3F22A305741328463E

Ak si povieme, že highway musí mať hodnotu, výsledok dostaneme v zemianskej oľči.

```
28 SELECT name, nm5 as meno_kataster, ST_Length(ST_Intersection(ST_Transform(way,3857),ST_Transform(wkb_geometry,3857))) as dlzka_cesty, way as road_way, wkb_geometry as kataster_way, highway, boundary
29 FROM
30 (SELECT +
31 FROM planet_osm_roads
32 WHERE ST_Intersects(way, (SELECT way
33 FROM planet_osm_polygon
34 WHERE admin_level = '8' and name = 'okres Komárno')))) as cesty,
35 (select +
36 FROM test
37 WHERE ST_contains((SELECT way
38 FROM planet_osm_polygon
39 WHERE admin_level = '8' and name = 'okres Komárno'),ST_Transform(wkb_geometry,3857))) as katastre
40 WHERE ST_Intersects(ST_Transform(way,3857),ST_Transform(wkb_geometry,3857)) AND boundary is NULL and highway IS NOT NULL
41 ORDER BY dlzka_cesty DESC
42 LIMIT 1
```

name	meno_kataster	dlzka_cesty	road_way
text	character varying (50)	double precision	geometry
1	[null]	Zemianska Oľča	4512.904939913218 0102000020110F00000040100001F814BCC2B9A3E416E40317EEB2557418FADC39CC8993E41C02BA885E925574161B1F70648993E41DFAED137E82557416B9E18C71B993E414F04DA0E52557413A5A61EEB963E41417E6C33E4255741D0EBAE1



Prikladám aj druhý výsledok, ale ten má highway NULL, a nieje úplne jasné či sa jedná o cestu pojazdnú autom. Táto cesta je ale dlhšia ako predošlí výsledok.



13.

Najprv si nájdeme polygón, ktorý predstavuje bratislavu pomocou %Bratislava% a admin level = 6. Vytvoríme okolo nej 20km buffer, ktorý ale limitujeme len na slovensko pomocou ST\_Intersection. Následne od tohto polygónu zmažeme časť, ktorú pokrýva Bratislava samotná pomocou ST\_Difference.

```

12 SELECT ST_DIFFERENCE((SELECT ST_INTERSECTION((SELECT ST_BUFFER(
13 (SELECT way
14 FROM planet_osm_polygon
15 WHERE NAME LIKE '%Bratislava%' AND admin_level = '6'),20000)),
16 (SELECT way
17 FROM planet_osm_polygon
18 WHERE admin_level = '2'))),(SELECT way
19 FROM planet_osm_polygon
20 WHERE NAME LIKE '%Bratislava%' AND admin_level = '6'))

```

Data output Messages Geometry Viewer × Notifications



Total rows: 1 of 1 Query complete 00:00:00.583

Leaflet  
Ln 11, Col 1

Následne po vložení do tabuľky zistíme rozmer.

```

56 SELECT ST_AREA(ST_TRANSFORM(way,4326),FALSE)/1000000 as sizekm
57 FROM planet_osm_polygon
58 WHERE osm_id = 241212

```

Data output Messages Geometry Viewer × Notifications



sizekm  
double precision

1	883.6050290981573
---	-------------------