1. Popis projektu

Aplikácia by mala pomôcť šoférom alebo ľudom, ktorí si potrebujú načerpať palivo a nevedia, kde presne sa nachádza. Keďže dáta z druhého datasetu sú z Ameriky, budem sa zameriavať v projekte na štát Colorado.

Celá aplikácia sa skladá z troch častí a to databáza, kde mám uložené dva zdroje dát, ktoré som spomínal vyššie. Potom backend(FLASK), ktorý prijíma požiadavky z frontendu a zároveň spracováva a vytvára dopyty na databázu. Následne všetko konvertuje do formatu GEOJSON a odosiela na frontend. Frontend(MAPBOX-GL) spracováva a zobrazuje informácie cez formát geojson, filtruje a posiela requesty na backend.

Celkovo som pracoval s dátami na Mapboxe ako s klastrami, takže sa zobrazujú niektoré stanice až po priblížení. Na mape je pohyblivý bod, pomocou ktorého sa dajú vykonávať scenáre. Vytvoril som aj heatmapu, ktorá znázoruje hustotu benziniek na mape. Ikony na mape sú rozdelené podľa toho aké palivo je možné načerpať, teda palivo z ropy alebo na elektické palivo.

Najdôležitejšie scenáre:

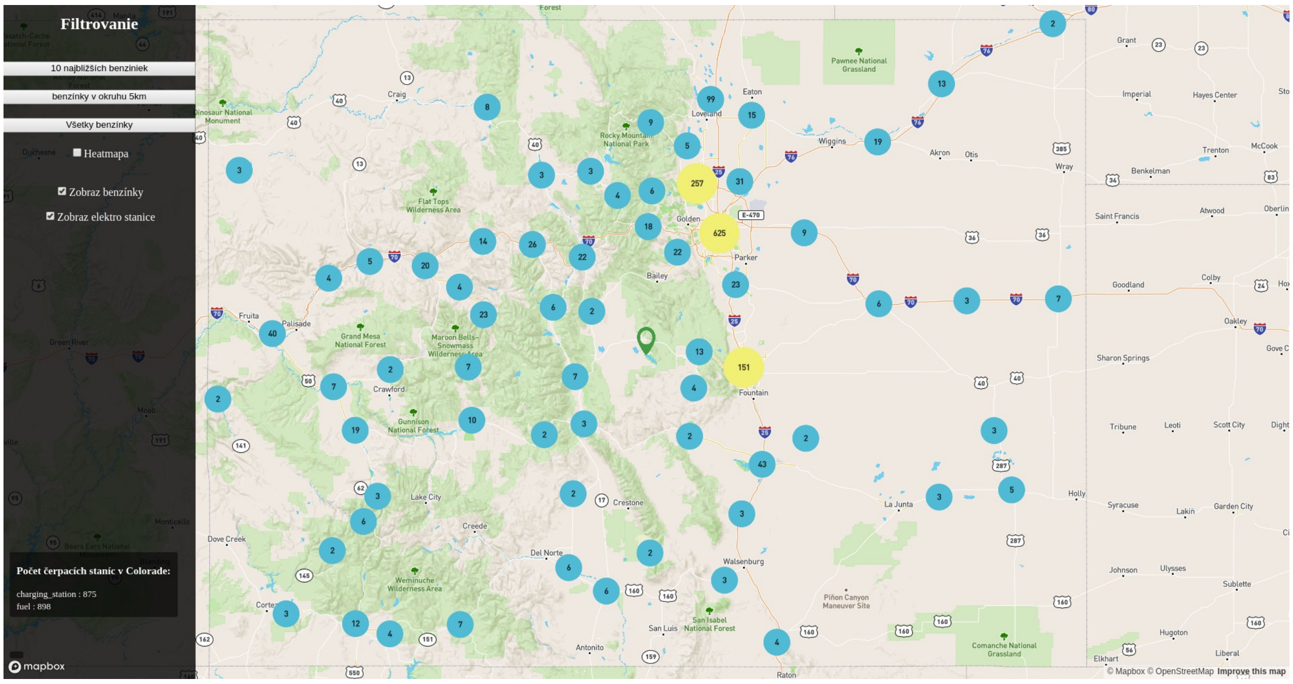
- Nájsť najbližších 10 benziniek
- Benzínky v okruhu 5 km
- Filtrovanie podľa elektrickej benzinky a elektrického paliva
- heatmapa benziniek
- zobrazenie počtu benziniek a elektrických čerpacích staníc

Zdroj dát: Openstreetmap.com, API na tankovacie
stanice: https://developer.nrel.gov/docs/transportation/alt-fuel-stations-v1/

Použité technológie: POSTGIS, PYTHON, MAPBOX-GL

2. Frontend

Ako frontend som použil Mapbox-GL, ktorý je novšia verzia mapboxjs. Aplikácia zobrazuje benzínky a elektro stanice na mape, ktoré sa zobrazujú v klastroch. Na frontende je možné filtrovanie dát, ktoré sa odosielajú na backend pomocou rôznych requestov. Na frontende je možné zobrazovať všetky scenáre popísané vyššie. Treba mať spustený plugin CORS, ktorý povoluje Access-Control-Allow-Origin.



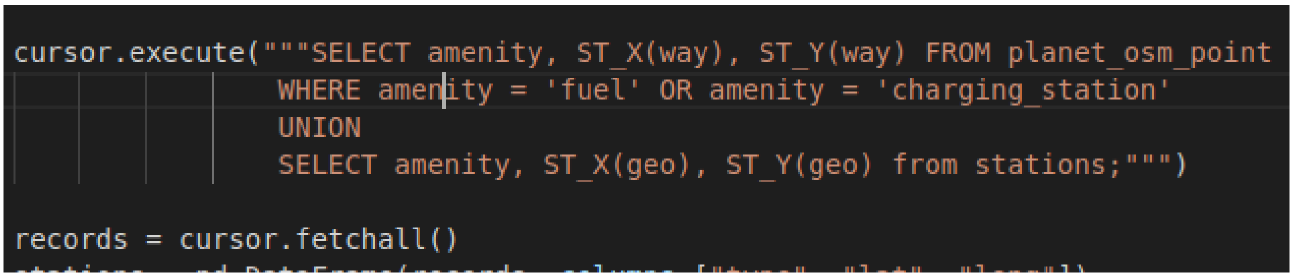
Počet čerpacích staníc v Colorade:

charging_station : 875
fuel : 898

Obr. 1: Ukážka systému

2. Backend

Ako backend som použil Flask, čo je pythonovský framework na routovanie. Tu som písal všetky query potrebné na databázu. Celkovo je práca s týmto frameworkom jednoduchá. Napríklad v porovnaní so SpringBootom od javy.



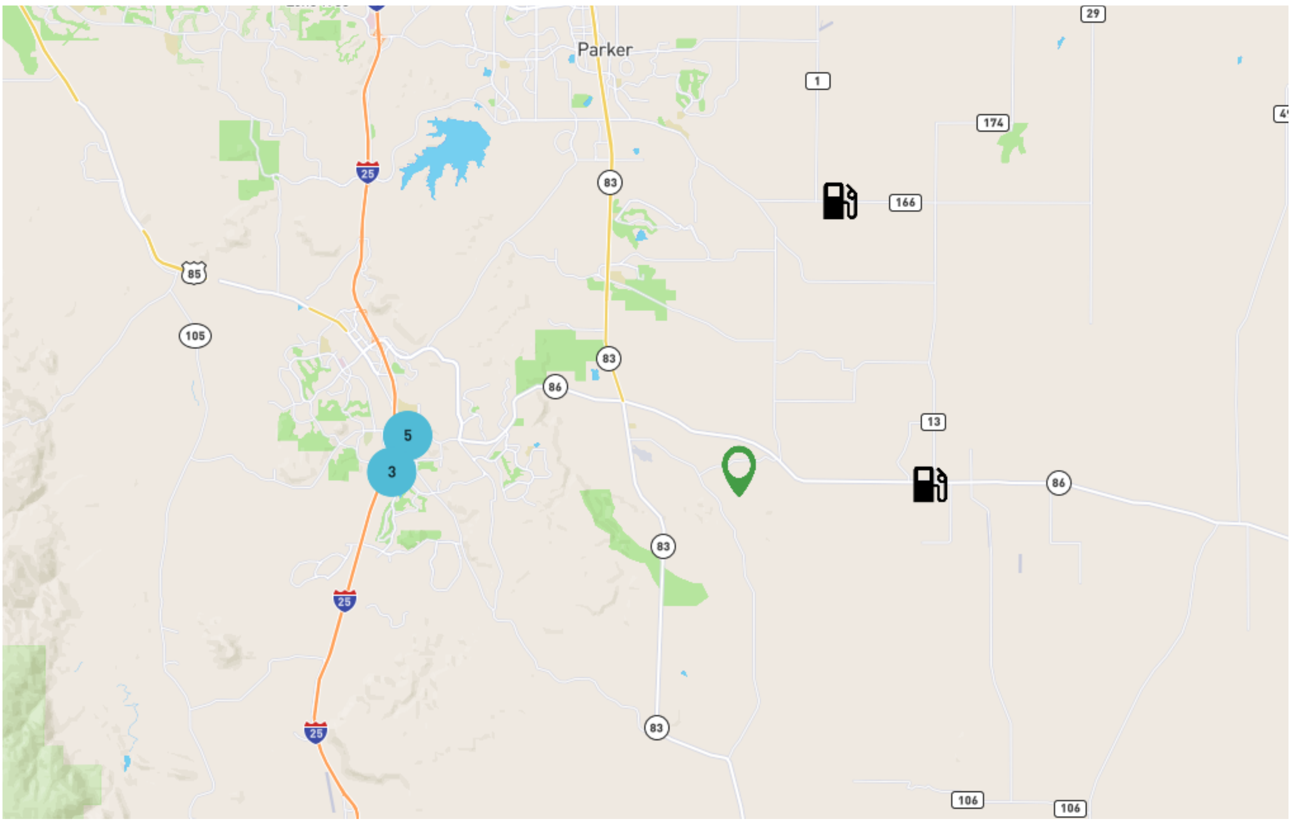
```
cursor.execute("""SELECT amenity, ST_X(way), ST_Y(way) FROM planet_osm_point
                WHERE amenity = 'fuel' OR amenity = 'charging_station'
                UNION
                SELECT amenity, ST_X(geo), ST_Y(geo) from stations;""")

records = cursor.fetchall()
```

Obr 2: Ukážka selectu na BE

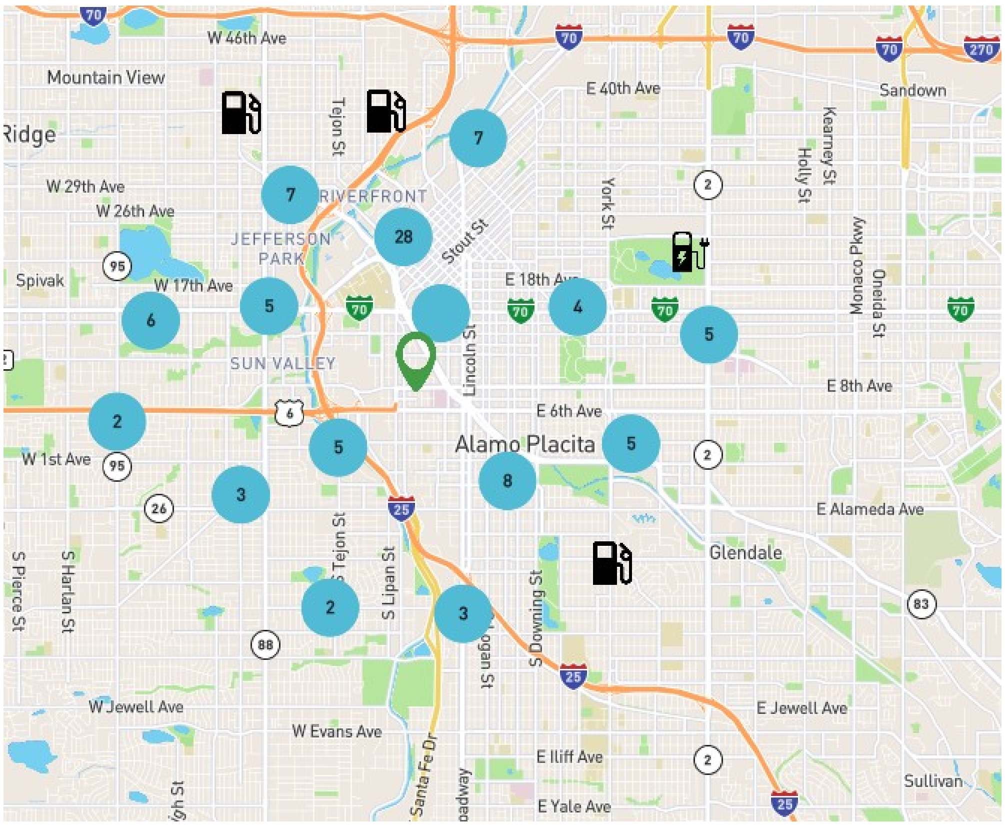
3. Ukážka scenárov

Na mape je možné hýbať pohyblivým bodom a tým neniť súradnice zadávania pri rôznych scenároch. K scenárom vždy pridávam query.

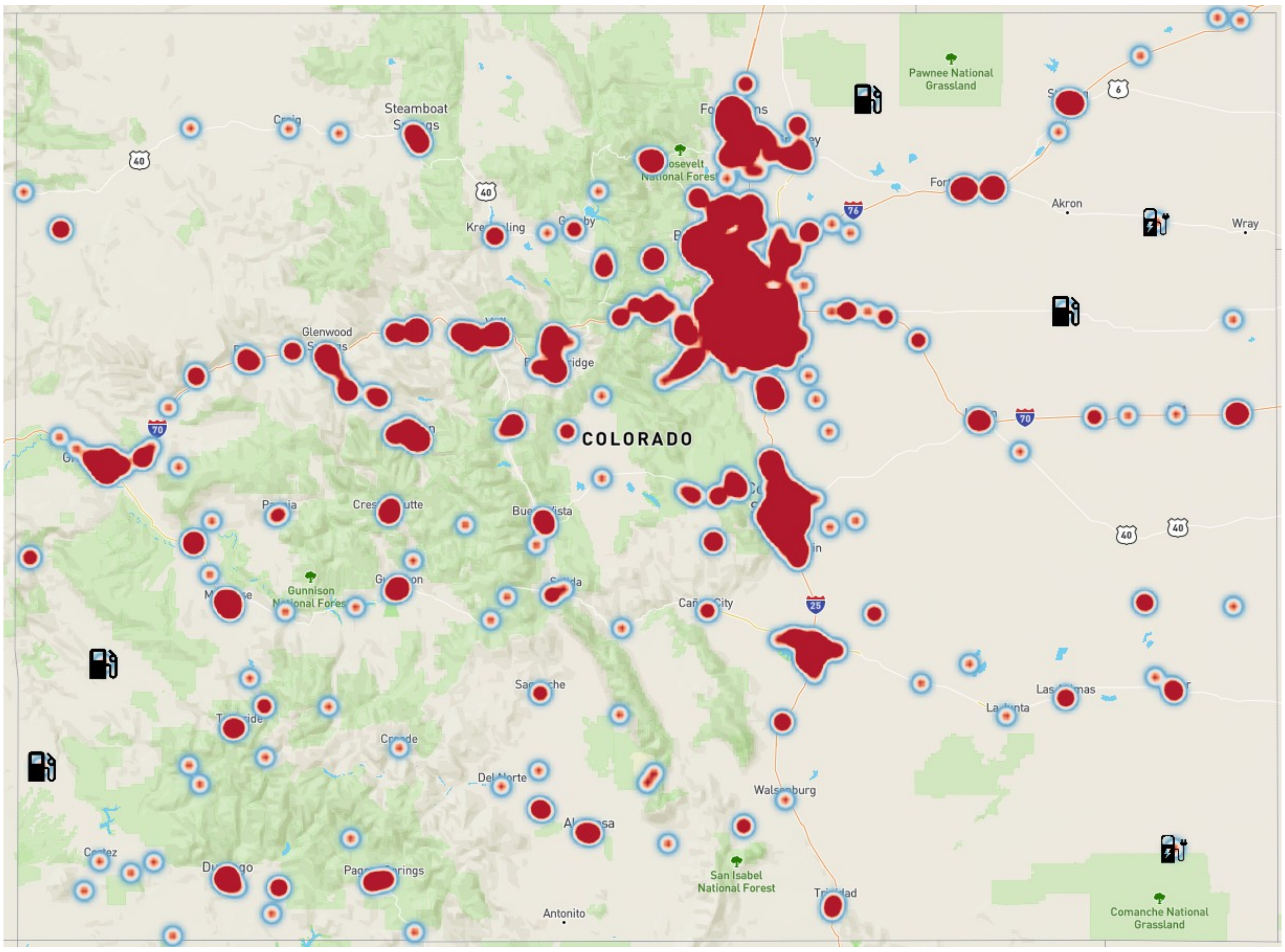
Nájdenie najbližších 10 benziniek:



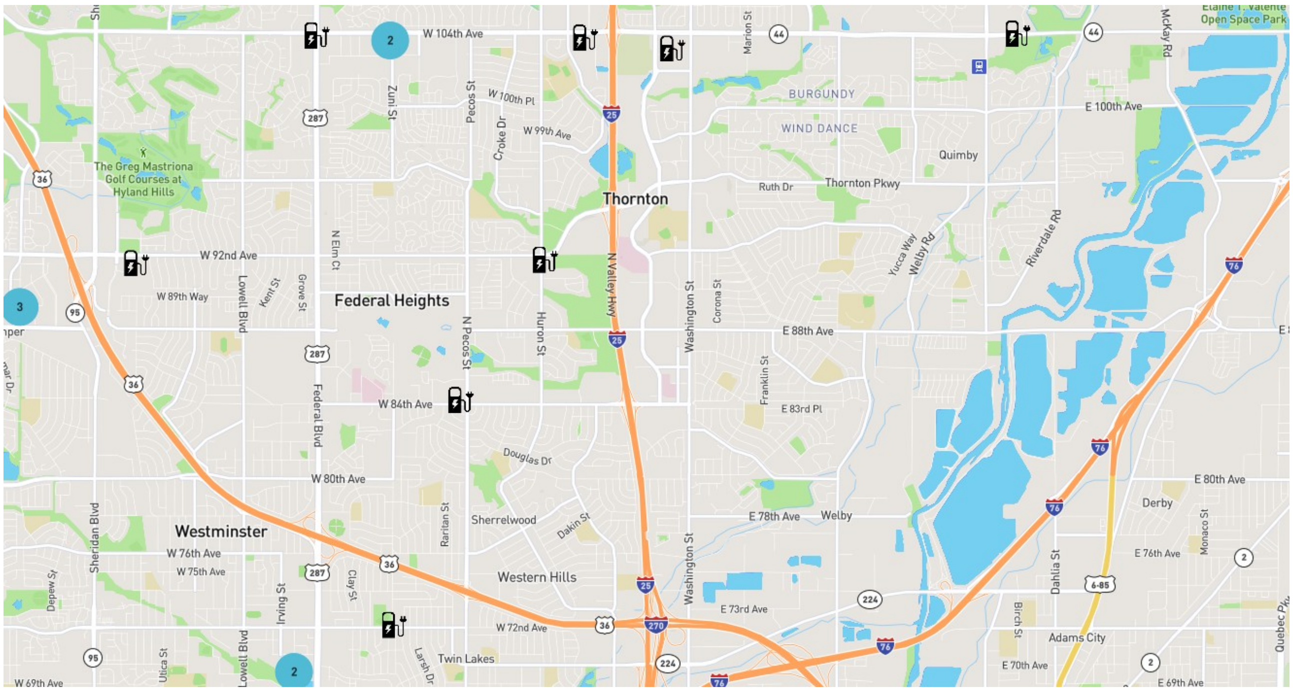
Nájdenie benzíniek v okruhu 5km:



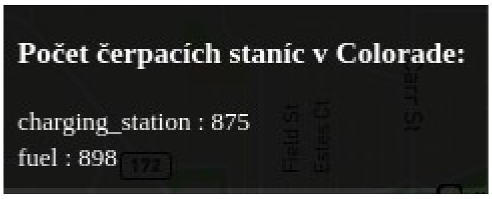
Heatmapa:



Filtrácia elektrostaníc a benziniek:



Zobrazenie počtu benziniek a elektrických čerpacích staníc:



Počet čerpacích staníc v Colorade:

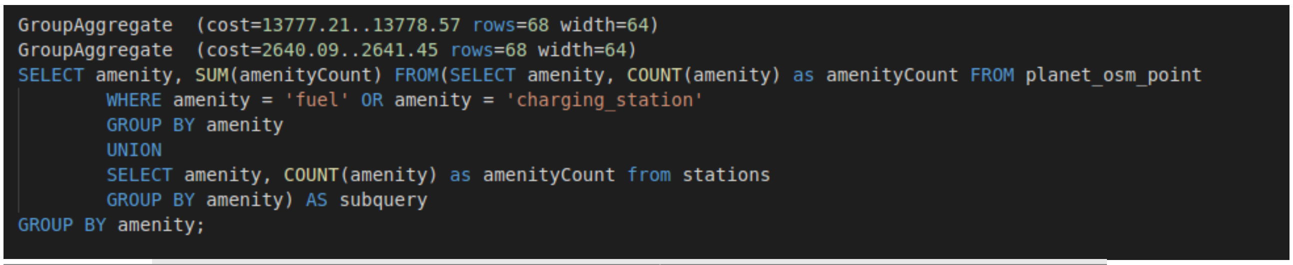
charging_station : 875
fuel : 898

4. Ukážka scenárov

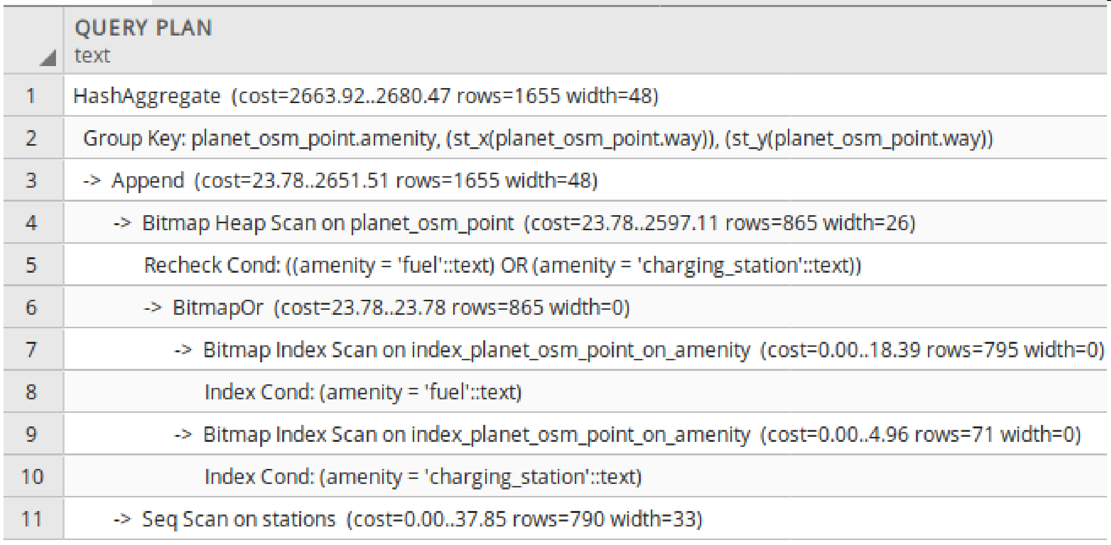
Na optimalizáciu query som použil indexovanie na základe parametru amenity. Čo mi v každom scenári zrýchlilo query v prepočte cca o 500 percent. V screenshote explainu je výsledok pred použitím indexu a po použití indexu. Vytvorenie indexu:

```
create index index_planet_osm_point_on_amenity on planet_osm_point(amenity);
```

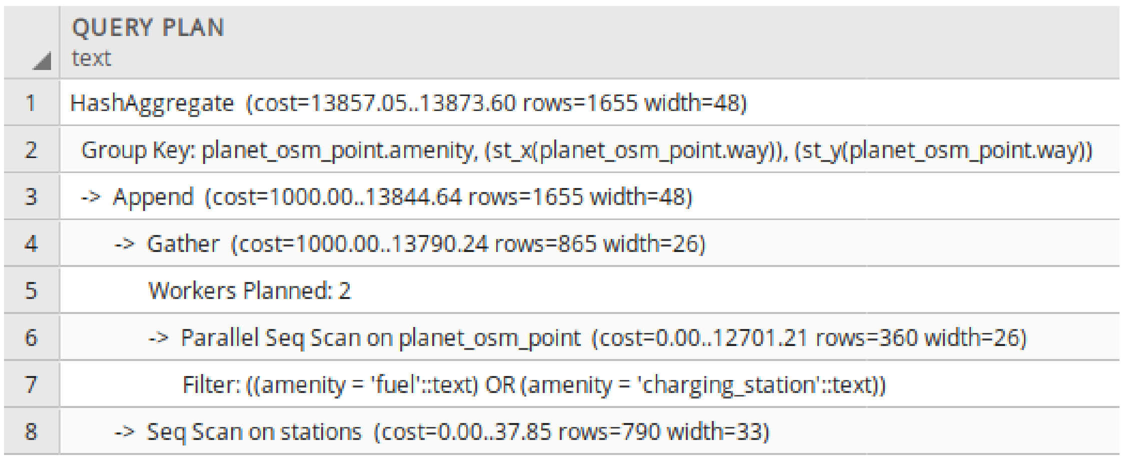
Všetky výsledky:



```
GroupAggregate  (cost=13777.21..13778.57 rows=68 width=64)
GroupAggregate  (cost=2640.09..2641.45 rows=68 width=64)
SELECT amenity, SUM(amenityCount) FROM(SELECT amenity, COUNT(amenity) as amenityCount FROM planet_osm_point
        WHERE amenity = 'fuel' OR amenity = 'charging_station'
        GROUP BY amenity
        UNION
        SELECT amenity, COUNT(amenity) as amenityCount from stations
        GROUP BY amenity) AS subquery
GROUP BY amenity;
```

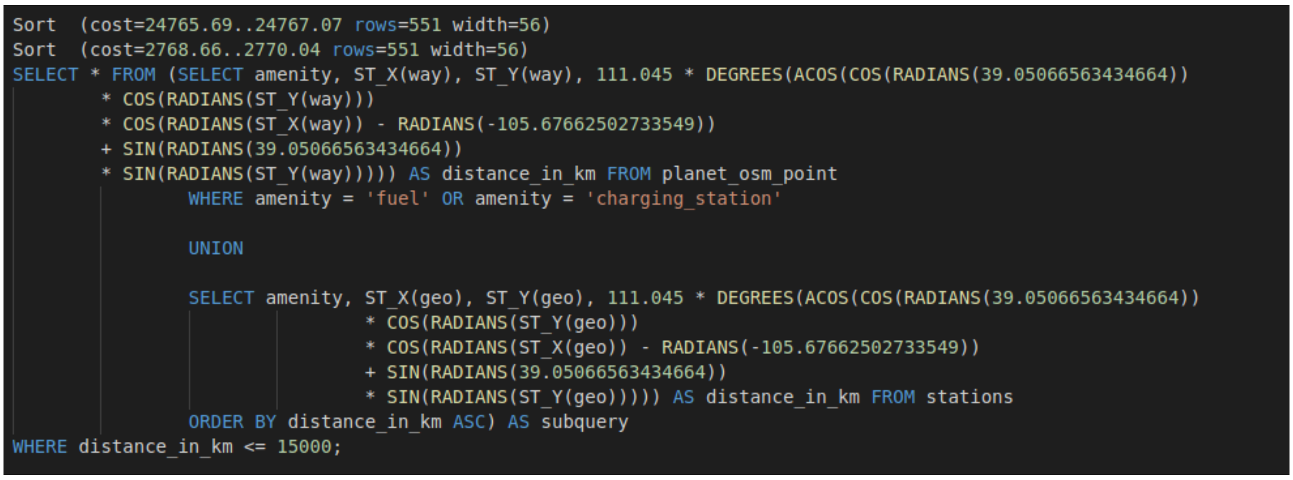


	QUERY PLAN text
1	HashAggregate (cost=2663.92..2680.47 rows=1655 width=48)
2	Group Key: planet_osm_point.amenity, (st_x(planet_osm_point.way)), (st_y(planet_osm_point.way))
3	-> Append (cost=23.78..2651.51 rows=1655 width=48)
4	-> Bitmap Heap Scan on planet_osm_point (cost=23.78..2597.11 rows=865 width=26)
5	Recheck Cond: ((amenity = 'fuel'::text) OR (amenity = 'charging_station'::text))
6	-> BitmapOr (cost=23.78..23.78 rows=865 width=0)
7	-> Bitmap Index Scan on index_planet_osm_point_on_amenity (cost=0.00..18.39 rows=795 width=0)
8	Index Cond: (amenity = 'fuel'::text)
9	-> Bitmap Index Scan on index_planet_osm_point_on_amenity (cost=0.00..4.96 rows=71 width=0)
10	Index Cond: (amenity = 'charging_station'::text)
11	-> Seq Scan on stations (cost=0.00..37.85 rows=790 width=33)

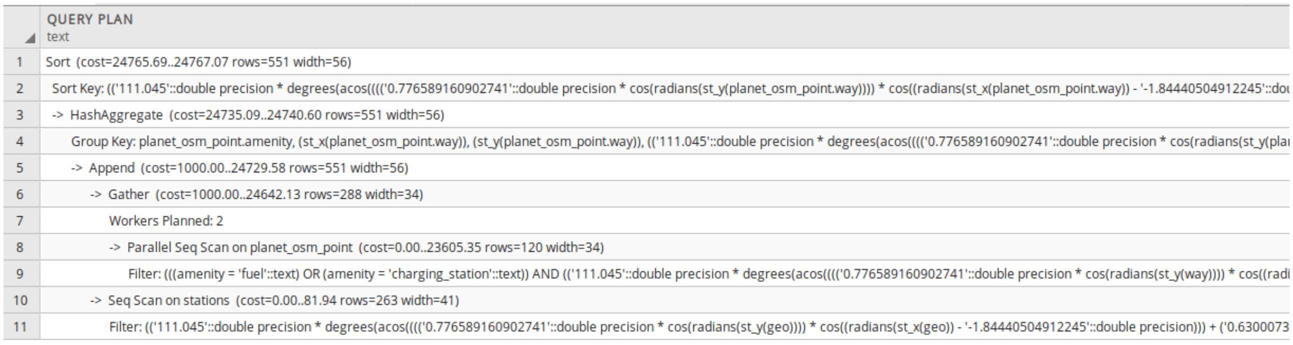


	QUERY PLAN text
1	HashAggregate (cost=13857.05..13873.60 rows=1655 width=48)
2	Group Key: planet_osm_point.amenity, (st_x(planet_osm_point.way)), (st_y(planet_osm_point.way))
3	-> Append (cost=1000.00..13844.64 rows=1655 width=48)
4	-> Gather (cost=1000.00..13790.24 rows=865 width=26)
5	Workers Planned: 2
6	-> Parallel Seq Scan on planet_osm_point (cost=0.00..12701.21 rows=360 width=26)
7	Filter: ((amenity = 'fuel'::text) OR (amenity = 'charging_station'::text))
8	-> Seq Scan on stations (cost=0.00..37.85 rows=790 width=33)

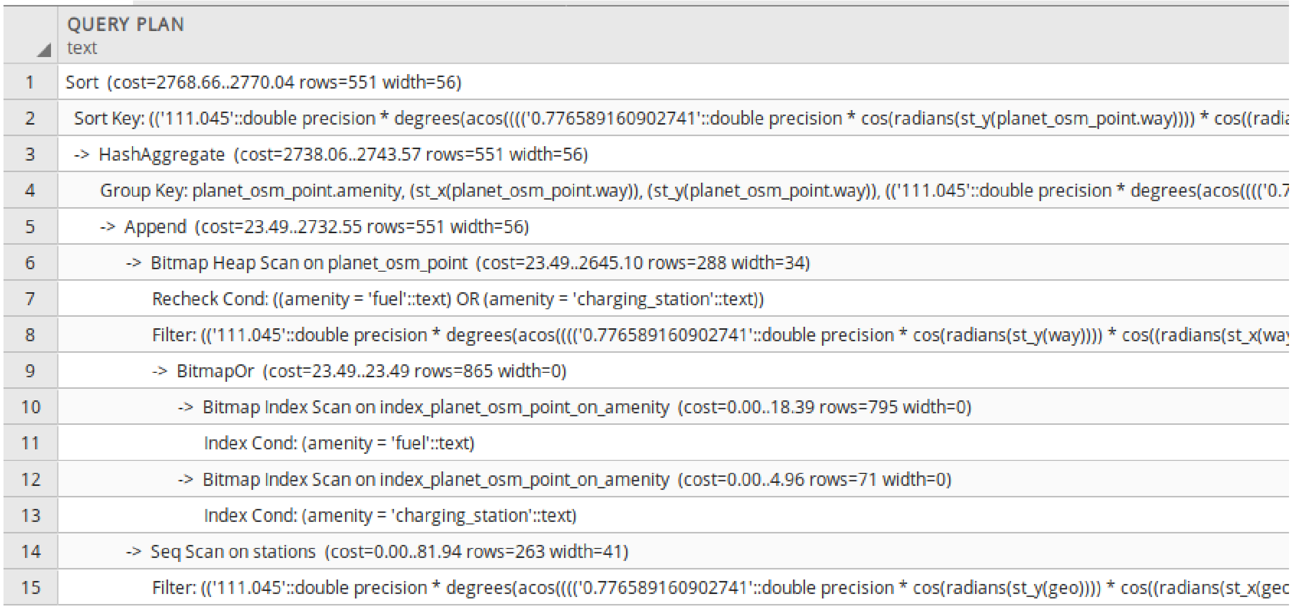
Nájdenie benzíniek v okruku 15 000 km:



```
Sort  (cost=24765.69..24767.07 rows=551 width=56)
Sort  (cost=2768.66..2770.04 rows=551 width=56)
SELECT * FROM (SELECT amenity, ST_X(way), ST_Y(way), 111.045 * DEGREES(ACOS(COS(RADIANS(39.05066563434664))
        * COS(RADIANS(ST_Y(way)))
        * COS(RADIANS(ST_X(way)) - RADIANS(-105.67662502733549))
        + SIN(RADIANS(39.05066563434664))
        * SIN(RADIANS(ST_Y(way))))) AS distance_in_km FROM planet_osm_point
                WHERE amenity = 'fuel' OR amenity = 'charging_station'

                UNION

                SELECT amenity, ST_X(geo), ST_Y(geo), 111.045 * DEGREES(ACOS(COS(RADIANS(39.05066563434664))
                        * COS(RADIANS(ST_Y(geo)))
                        * COS(RADIANS(ST_X(geo)) - RADIANS(-105.67662502733549))
                        + SIN(RADIANS(39.05066563434664))
                        * SIN(RADIANS(ST_Y(geo))))) AS distance_in_km FROM stations
                ORDER BY distance_in_km ASC) AS subquery
WHERE distance_in_km <= 15000;
```

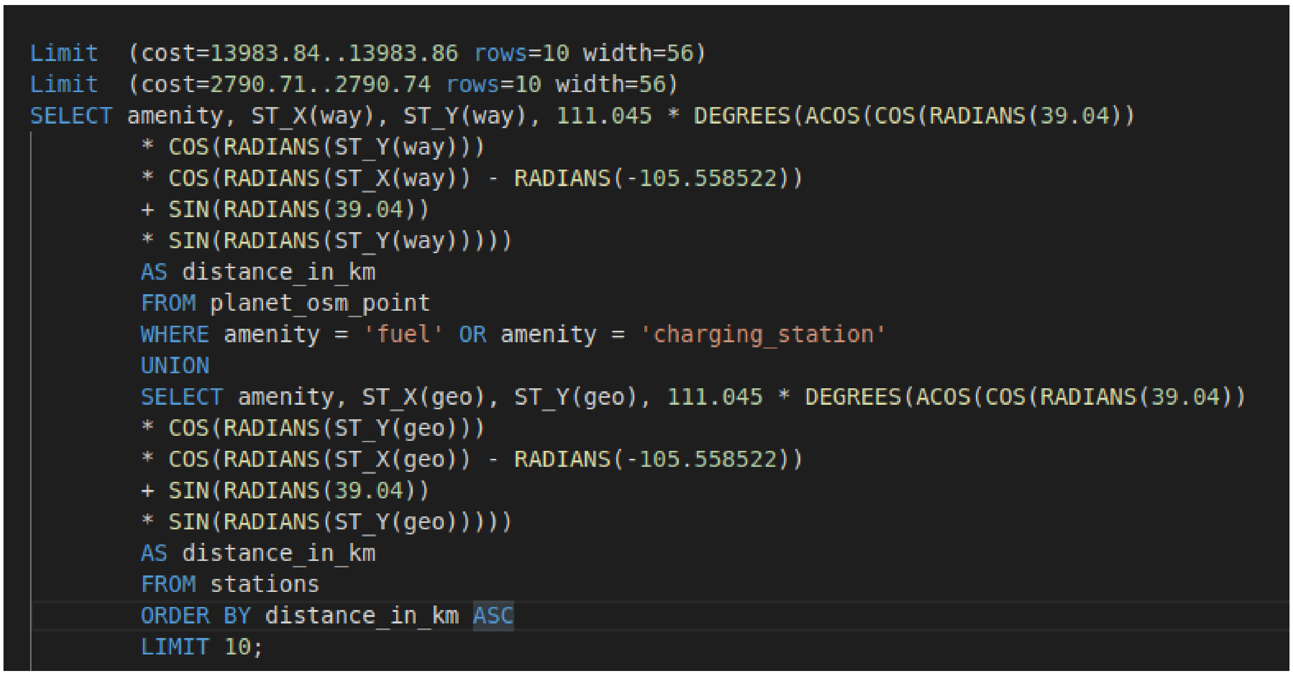


	QUERY PLAN text
1	Sort (cost=24765.69..24767.07 rows=551 width=56)
2	Sort Key: (('111.045'::double precision * degrees(acos(((('0.776589160902741'::double precision * cos(radians(st_y(planet_osm_point.way)))) * cos((radians(st_x(planet_osm_point.way)) - '-1.84440504912245'::dou
3	-> HashAggregate (cost=24735.09..24740.60 rows=551 width=56)
4	Group Key: planet_osm_point.amenity, (st_x(planet_osm_point.way)), (st_y(planet_osm_point.way)), (('111.045'::double precision * degrees(acos(((('0.776589160902741'::double precision * cos(radians(st_y(pla
5	-> Append (cost=1000.00..24729.58 rows=551 width=56)
6	-> Gather (cost=1000.00..24642.13 rows=288 width=34)
7	Workers Planned: 2
8	-> Parallel Seq Scan on planet_osm_point (cost=0.00..23605.35 rows=120 width=34)
9	Filter: (((amenity = 'fuel'::text) OR (amenity = 'charging_station'::text)) AND (('111.045'::double precision * degrees(acos(((('0.776589160902741'::double precision * cos(radians(st_y(way)))) * cos((radi
10	-> Seq Scan on stations (cost=0.00..81.94 rows=263 width=41)
11	Filter: (('111.045'::double precision * degrees(acos(((('0.776589160902741'::double precision * cos(radians(st_y(geo)))) * cos((radians(st_x(geo)) - '-1.84440504912245'::double precision))) + ('0.6300073

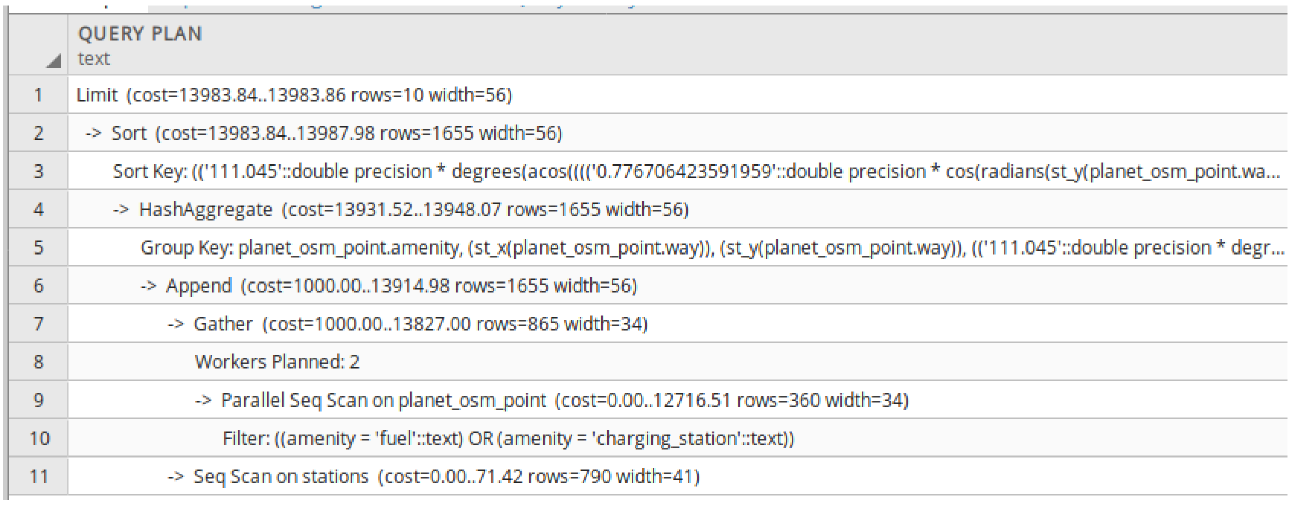


	QUERY PLAN text
1	Sort (cost=2768.66..2770.04 rows=551 width=56)
2	Sort Key: (('111.045'::double precision * degrees(acos(((('0.776589160902741'::double precision * cos(radians(st_y(planet_osm_point.way)))) * cos((radi
3	-> HashAggregate (cost=2738.06..2743.57 rows=551 width=56)
4	Group Key: planet_osm_point.amenity, (st_x(planet_osm_point.way)), (st_y(planet_osm_point.way)), (('111.045'::double precision * degrees(acos(((('0.7
5	-> Append (cost=23.49..2732.55 rows=551 width=56)
6	-> Bitmap Heap Scan on planet_osm_point (cost=23.49..2645.10 rows=288 width=34)
7	Recheck Cond: ((amenity = 'fuel'::text) OR (amenity = 'charging_station'::text))
8	Filter: (('111.045'::double precision * degrees(acos(((('0.776589160902741'::double precision * cos(radians(st_y(way)))) * cos((radians(st_x(way
9	-> BitmapOr (cost=23.49..23.49 rows=865 width=0)
10	-> Bitmap Index Scan on index_planet_osm_point_on_amenity (cost=0.00..18.39 rows=795 width=0)
11	Index Cond: (amenity = 'fuel'::text)
12	-> Bitmap Index Scan on index_planet_osm_point_on_amenity (cost=0.00..4.96 rows=71 width=0)
13	Index Cond: (amenity = 'charging_station'::text)
14	-> Seq Scan on stations (cost=0.00..81.94 rows=263 width=41)
15	Filter: (('111.045'::double precision * degrees(acos(((('0.776589160902741'::double precision * cos(radians(st_y(geo)))) * cos((radians(st_x(gec

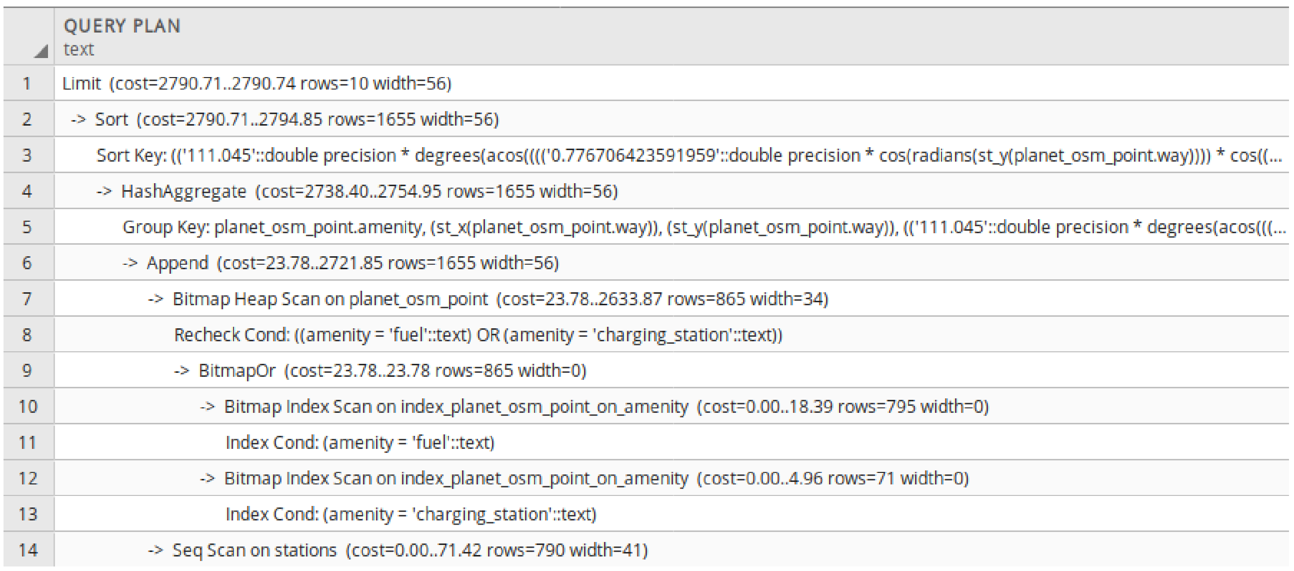
Najbližších 10 benziniek:



```
Limit  (cost=13983.84..13983.86 rows=10 width=56)
Limit  (cost=2790.71..2790.74 rows=10 width=56)
SELECT amenity, ST_X(way), ST_Y(way), 111.045 * DEGREES(ACOS(COS(RADIANS(39.04))
        * COS(RADIANS(ST_Y(way)))
        * COS(RADIANS(ST_X(way)) - RADIANS(-105.558522))
        + SIN(RADIANS(39.04))
        * SIN(RADIANS(ST_Y(way)))))
        AS distance_in_km
        FROM planet_osm_point
        WHERE amenity = 'fuel' OR amenity = 'charging_station'
        UNION
        SELECT amenity, ST_X(geo), ST_Y(geo), 111.045 * DEGREES(ACOS(COS(RADIANS(39.04))
        * COS(RADIANS(ST_Y(geo)))
        * COS(RADIANS(ST_X(geo)) - RADIANS(-105.558522))
        + SIN(RADIANS(39.04))
        * SIN(RADIANS(ST_Y(geo)))))
        AS distance_in_km
        FROM stations
        ORDER BY distance_in_km ASC
        LIMIT 10;
```

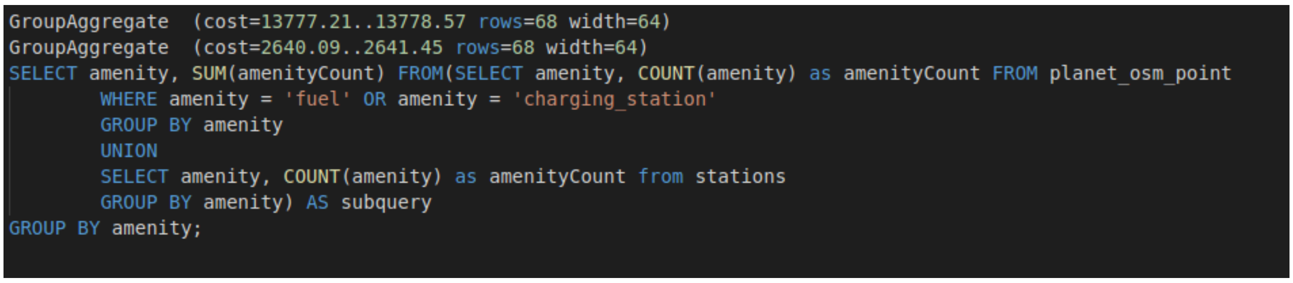


	QUERY PLAN text
1	Limit (cost=13983.84..13983.86 rows=10 width=56)
2	-> Sort (cost=13983.84..13987.98 rows=1655 width=56)
3	Sort Key: (('111.045'::double precision * degrees(acos(((('0.776706423591959'::double precision * cos(radians(st_y(planet_osm_point.wa...
4	-> HashAggregate (cost=13931.52..13948.07 rows=1655 width=56)
5	Group Key: planet_osm_point.amenity, (st_x(planet_osm_point.way)), (st_y(planet_osm_point.way)), (('111.045'::double precision * degr...
6	-> Append (cost=1000.00..13914.98 rows=1655 width=56)
7	-> Gather (cost=1000.00..13827.00 rows=865 width=34)
8	Workers Planned: 2
9	-> Parallel Seq Scan on planet_osm_point (cost=0.00..12716.51 rows=360 width=34)
10	Filter: ((amenity = 'fuel'::text) OR (amenity = 'charging_station'::text))
11	-> Seq Scan on stations (cost=0.00..71.42 rows=790 width=41)

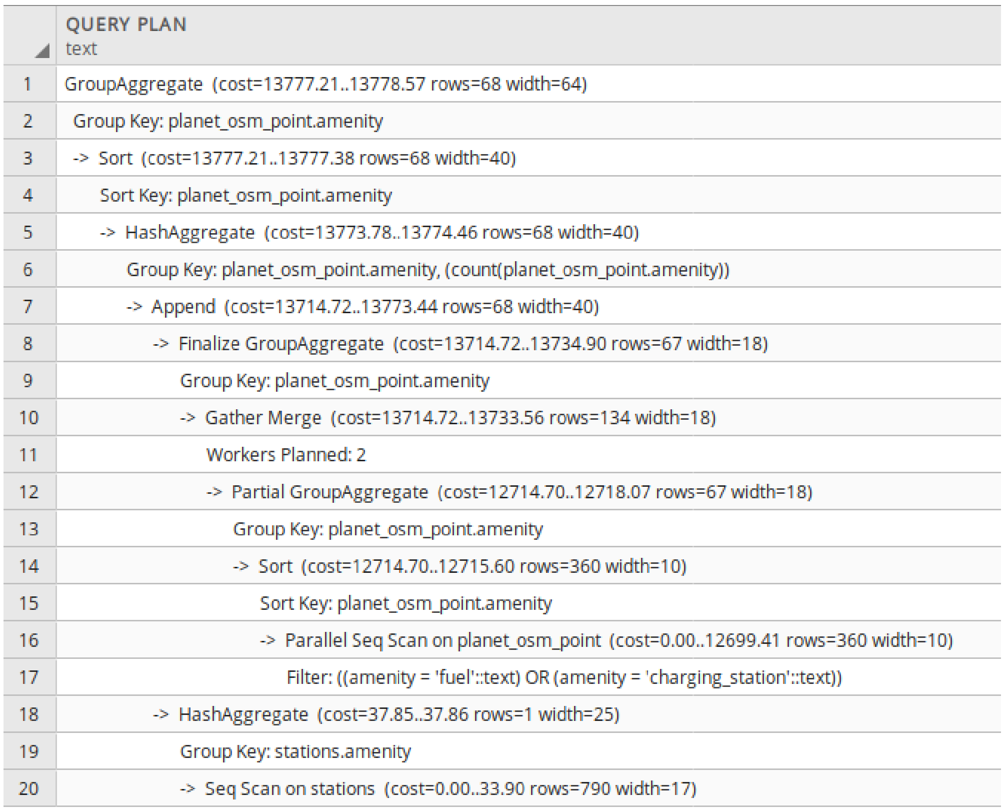


	QUERY PLAN text
1	Limit (cost=2790.71..2790.74 rows=10 width=56)
2	-> Sort (cost=2790.71..2794.85 rows=1655 width=56)
3	Sort Key: (('111.045'::double precision * degrees(acos(((('0.776706423591959'::double precision * cos(radians(st_y(planet_osm_point.way)))) * cos((...
4	-> HashAggregate (cost=2738.40..2754.95 rows=1655 width=56)
5	Group Key: planet_osm_point.amenity, (st_x(planet_osm_point.way)), (st_y(planet_osm_point.way)), (('111.045'::double precision * degrees(acos(((...
6	-> Append (cost=23.78..2721.85 rows=1655 width=56)
7	-> Bitmap Heap Scan on planet_osm_point (cost=23.78..2633.87 rows=865 width=34)
8	Recheck Cond: ((amenity = 'fuel'::text) OR (amenity = 'charging_station'::text))
9	-> BitmapOr (cost=23.78..23.78 rows=865 width=0)
10	-> Bitmap Index Scan on index_planet_osm_point_on_amenity (cost=0.00..18.39 rows=795 width=0)
11	Index Cond: (amenity = 'fuel'::text)
12	-> Bitmap Index Scan on index_planet_osm_point_on_amenity (cost=0.00..4.96 rows=71 width=0)
13	Index Cond: (amenity = 'charging_station'::text)
14	-> Seq Scan on stations (cost=0.00..71.42 rows=790 width=41)

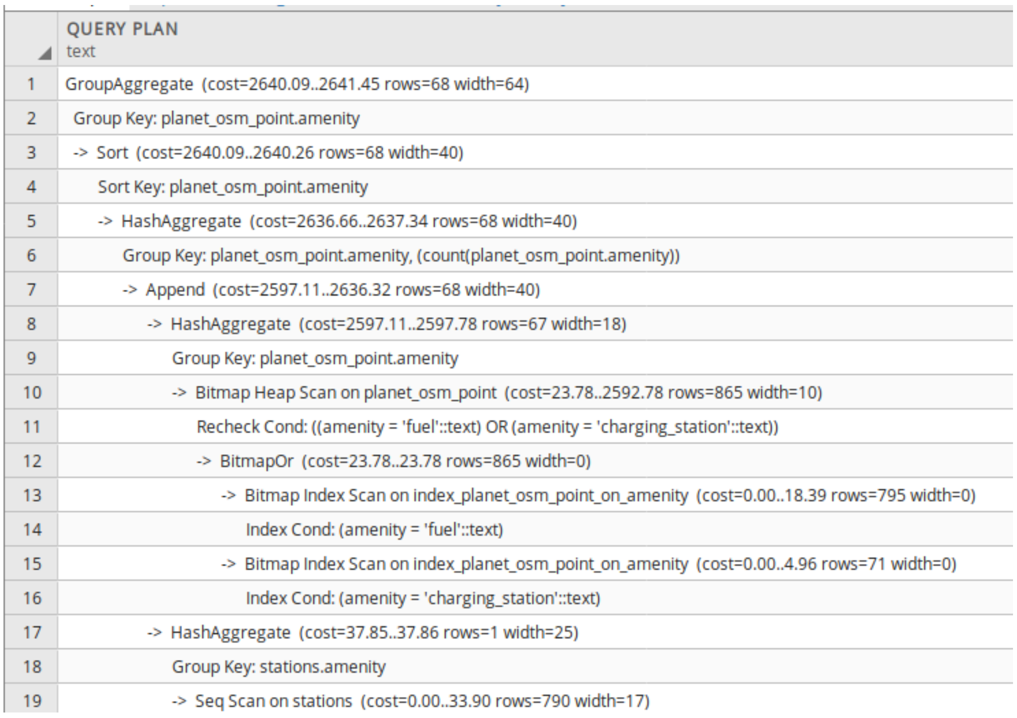
Zobrazenie počtu benziniek a elektrických čerpacích staníc(agregačná funkcia):



```
GroupAggregate  (cost=13777.21..13778.57 rows=68 width=64)
GroupAggregate  (cost=2640.09..2641.45 rows=68 width=64)
SELECT amenity, SUM(amenityCount) FROM(SELECT amenity, COUNT(amenity) as amenityCount FROM planet_osm_point
        WHERE amenity = 'fuel' OR amenity = 'charging_station'
        GROUP BY amenity
        UNION
        SELECT amenity, COUNT(amenity) as amenityCount from stations
        GROUP BY amenity) AS subquery
GROUP BY amenity;
```



	QUERY PLAN text
1	GroupAggregate (cost=13777.21..13778.57 rows=68 width=64)
2	Group Key: planet_osm_point.amenity
3	-> Sort (cost=13777.21..13777.38 rows=68 width=40)
4	Sort Key: planet_osm_point.amenity
5	-> HashAggregate (cost=13773.78..13774.46 rows=68 width=40)
6	Group Key: planet_osm_point.amenity, (count(planet_osm_point.amenity))
7	-> Append (cost=13714.72..13773.44 rows=68 width=40)
8	-> Finalize GroupAggregate (cost=13714.72..13734.90 rows=67 width=18)
9	Group Key: planet_osm_point.amenity
10	-> Gather Merge (cost=13714.72..13733.56 rows=134 width=18)
11	Workers Planned: 2
12	-> Partial GroupAggregate (cost=12714.70..12718.07 rows=67 width=18)
13	Group Key: planet_osm_point.amenity
14	-> Sort (cost=12714.70..12715.60 rows=360 width=10)
15	Sort Key: planet_osm_point.amenity
16	-> Parallel Seq Scan on planet_osm_point (cost=0.00..12699.41 rows=360 width=10)
17	Filter: ((amenity = 'fuel'::text) OR (amenity = 'charging_station'::text))
18	-> HashAggregate (cost=37.85..37.86 rows=1 width=25)
19	Group Key: stations.amenity
20	-> Seq Scan on stations (cost=0.00..33.90 rows=790 width=17)



	QUERY PLAN text
1	GroupAggregate (cost=2640.09..2641.45 rows=68 width=64)
2	Group Key: planet_osm_point.amenity
3	-> Sort (cost=2640.09..2640.26 rows=68 width=40)
4	Sort Key: planet_osm_point.amenity
5	-> HashAggregate (cost=2636.66..2637.34 rows=68 width=40)
6	Group Key: planet_osm_point.amenity, (count(planet_osm_point.amenity))
7	-> Append (cost=2597.11..2636.32 rows=68 width=40)
8	-> HashAggregate (cost=2597.11..2597.78 rows=67 width=18)
9	Group Key: planet_osm_point.amenity
10	-> Bitmap Heap Scan on planet_osm_point (cost=23.78..2592.78 rows=865 width=10)
11	Recheck Cond: ((amenity = 'fuel'::text) OR (amenity = 'charging_station'::text))
12	-> BitmapOr (cost=23.78..23.78 rows=865 width=0)
13	-> Bitmap Index Scan on index_planet_osm_point_on_amenity (cost=0.00..18.39 rows=795 width=0)
14	Index Cond: (amenity = 'fuel'::text)
15	-> Bitmap Index Scan on index_planet_osm_point_on_amenity (cost=0.00..4.96 rows=71 width=0)
16	Index Cond: (amenity = 'charging_station'::text)
17	-> HashAggregate (cost=37.85..37.86 rows=1 width=25)
18	Group Key: stations.amenity
19	-> Seq Scan on stations (cost=0.00..33.90 rows=790 width=17)