

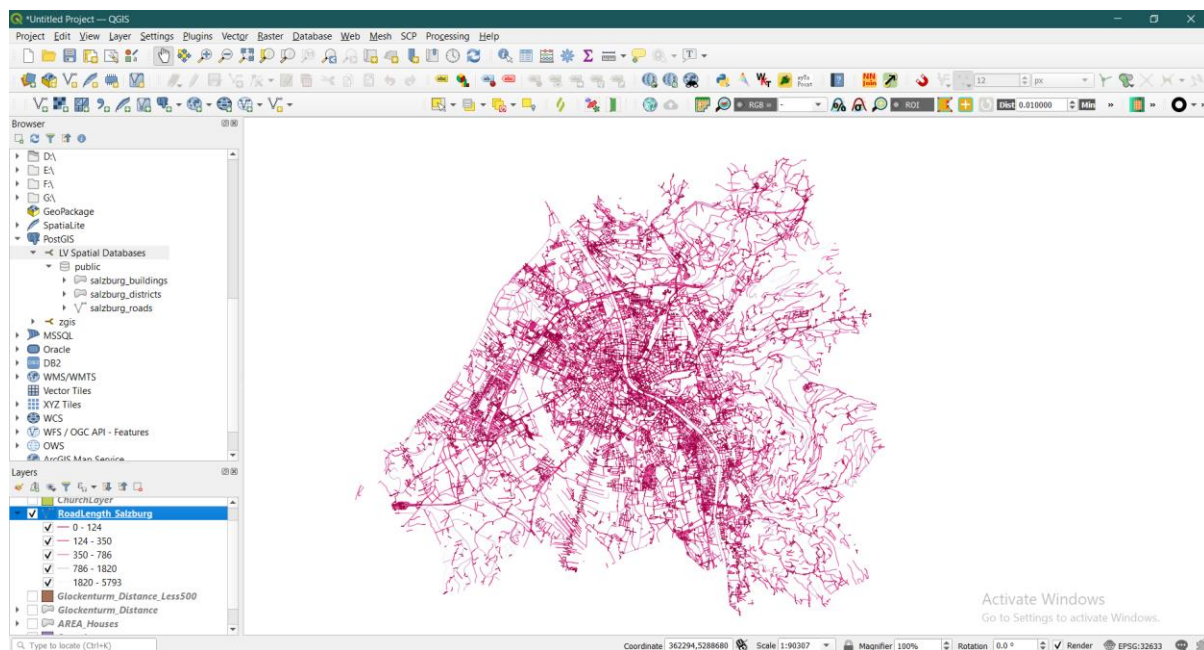
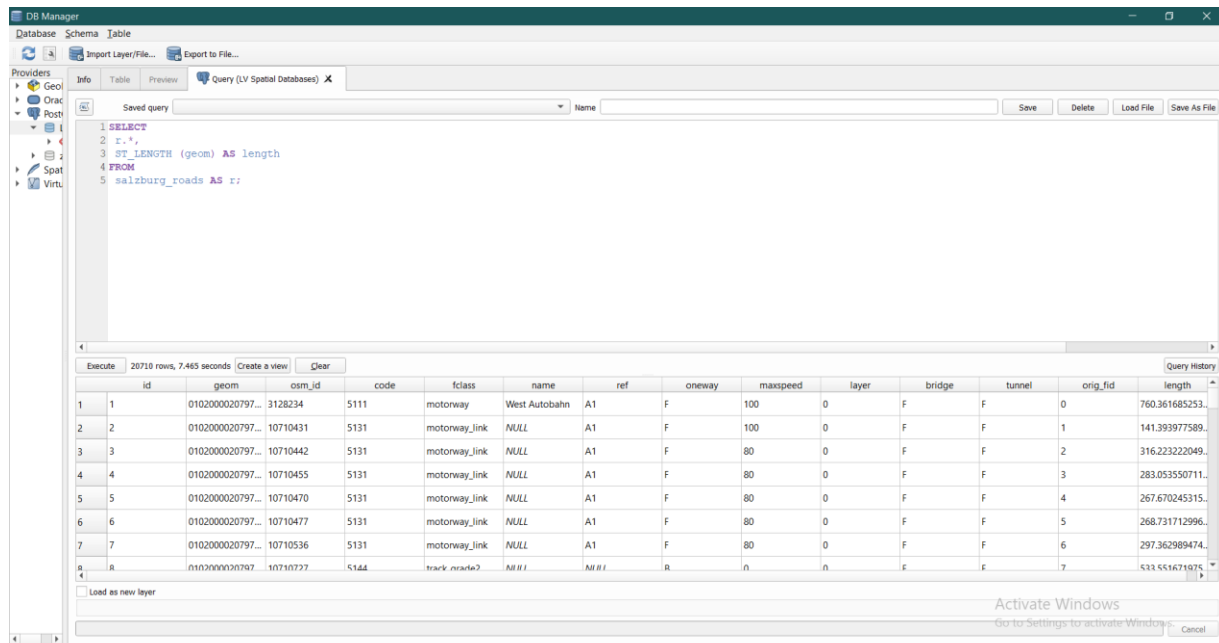
## Kiarash Pooladsaz-12118707

### Spatial DataBases-SQL\_III Assignment

- Assign categories based on spatial properties

“Use SQL to create categories and associate numeric values of table columns to them, particularly categorizing roads based on their length.”

We can categorize the road layer length in 2 ways; we can calculate the length of each attribute and do symbology in Qgis, as follow:



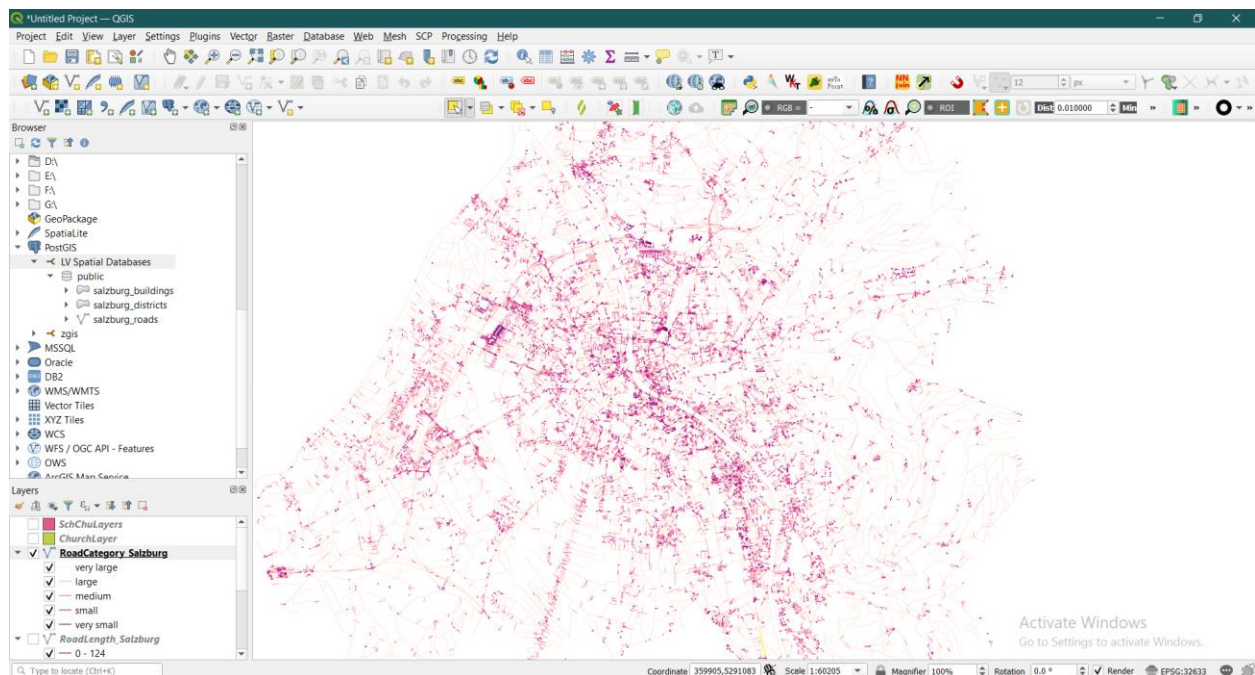
Or, we can determine which amounts of length are being described as which type, such as small:

The screenshot shows the QGIS DB Manager interface. The 'Query (LV Spatial Databases)' tab is active, displaying a SQL query that categorizes road lengths into five types: 'very small', 'small', 'medium', 'large', and 'very large'. The query results are shown in a table with 15 columns: id, geom, osm\_id, code, fclass, name, ref, oneway, maxspeed, layer, bridge, tunnel, orig\_fid, and road\_category. The results show 6 rows of data, all categorized as 'very large' or 'large'.

```
1 SELECT
2   *;
3 CASE
4   WHEN ST_LENGTH(geom) <= 15 THEN 'very small'
5   WHEN ST_LENGTH(geom) > 15 AND ST_LENGTH(geom) <= 40 THEN 'small'
6   WHEN ST_LENGTH(geom) > 40 AND ST_LENGTH(geom) <= 80 THEN 'medium'
7   WHEN ST_LENGTH(geom) > 80 AND ST_LENGTH(geom) <= 150 THEN 'large'
8   WHEN ST_LENGTH(geom) > 150 THEN 'very large'
9 END AS road_category
10 FROM salzburg_roads AS r;
```

	id	geom	osm_id	code	fclass	name	ref	oneway	maxspeed	layer	bridge	tunnel	orig_fid	road_category
1	1	0102000020797...	3128234	5111	motorway	West Autobahn	A1	F	100	0	F	F	0	very large
2	2	0102000020797...	10710431	5131	motorway_link		A1	F	100	0	F	F	1	large
3	3	0102000020797...	10710442	5131	motorway_link		A1	F	80	0	F	F	2	very large
4	4	0102000020797...	10710455	5131	motorway_link		A1	F	80	0	F	F	3	very large
5	5	0102000020797...	10710470	5131	motorway_link		A1	F	80	0	F	F	4	very large
6	6	0102000020797...	10710477	5131	motorway_link		A1	F	80	0	F	F	5	very large

Below the table, there are options to 'Load as new layer'. The 'Layer name (prefix)' is set to 'RoadCategory\_Salzburg'. The 'Geometry column' is set to 'geom'. The 'Avoid selecting by feature id' checkbox is unchecked.



- *Cross-products with distance calculation*

“Create a query to calculate the distance from every building to a motorway.”

We should determine that we want to calculate the distance of one building to the closest motorway. Therefore, ST\_UNION helps us here to avoid long-time waiting for other queries possibilities. It becomes the main measurement factor in this query instead of the default geom.

The screenshot shows the QGIS DB Manager interface. The SQL query is as follows:

```

1 SELECT
2 b.*
3 ST_DISTANCE(m.geom,b.geom) AS distance
4 FROM
5 salzburg_buildings AS b,
6 (SELECT
7 ST_UNION(geom) AS geom
8 FROM
9 salzburg_roads
10 WHERE
11 fclass='motorway' OR fclass='motorway_link') AS m;

```

The query results table is displayed below the query editor:

	id	geom	osm_id	code	fclass	name	type	orig_fid	distance
1	1	0103000020797...	24403125	1500	building	NULL	apartments	0	377.495067921...
2	2	0103000020797...	24403128	1500	building	Gummitechnik ...	commercial	1	401.637463303...
3	3	0103000020797...	24403133	1500	building	NULL	apartments	2	285.639145150...
4	4	0103000020797...	24403137	1500	building	NULL	apartments	3	403.682518806...
5	5	0103000020797...	24403141	1500	building	NULL	apartments	4	491.244664147...
6	6	0103000020797...	24403147	1500	building	NULL	detached	5	185.174264805...
7	7	0103000020797...	24403150	1500	building	Tapezierer Land...	retail	6	138.297201378...

Below the table, there are options to load the results as a new layer. The layer name is set to "DistanceToMotorway\_Salzburg".

We also make a query to add motorway layer to the Qgis:

The screenshot shows the QGIS DB Manager interface. The SQL query is as follows:

```

1 SELECT
2 r.*
3 FROM
4 salzburg_roads AS r
5 WHERE
6 fclass= 'motorway' OR fclass= 'motorway_link';

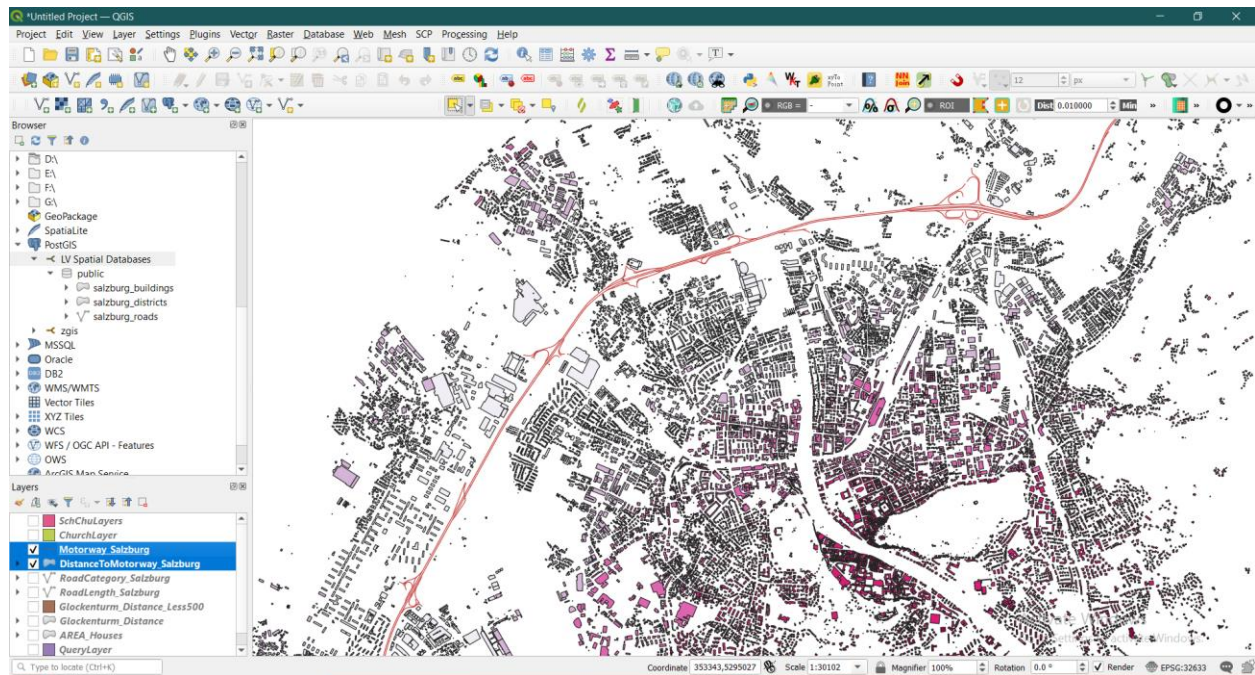
```

The query results table is displayed below the query editor:

	id	geom	osm_id	code	fclass	name	ref	oneway	maxspeed	layer	bridge	tunnel	orig_fid
1	1	0102000020797...	3128234	S111	motorway	West Autobahn	A1	F	100	0	F	F	0
2	2	0102000020797...	10710431	S131	motorway_link	NULL	A1	F	100	0	F	F	1
3	3	0102000020797...	10710442	S131	motorway_link	NULL	A1	F	80	0	F	F	2
4	4	0102000020797...	10710455	S131	motorway_link	NULL	A1	F	80	0	F	F	3
5	5	0102000020797...	10710470	S131	motorway_link	NULL	A1	F	80	0	F	F	4
6	6	0102000020797...	10710477	S131	motorway_link	NULL	A1	F	80	0	F	F	5
7	7	0102000020797...	10710536	S131	motorway_link	NULL	A1	F	80	0	F	F	6

Below the table, there are options to load the results as a new layer. The layer name is set to "Motorway\_Salzburg".

Therefore, the map shows what we were expecting☺:



- *The particularities of buffers, negation, and long queries*

“Create a query that selects all buildings with more than 20 meters distance to a road.”

As mentioned in the assignment, there are 2 ways (maybe at least) to select buildings which are located more than 20 meters from the road. You can make a query by ST\_BUFFER to create the potential noise buffer and then try to select buildings that their minimum distance to the 20 meter potential noise buffer is not true. Well, the buffer has been created first:

The screenshot shows the DB Manager window in QGIS. The SQL query is as follows:

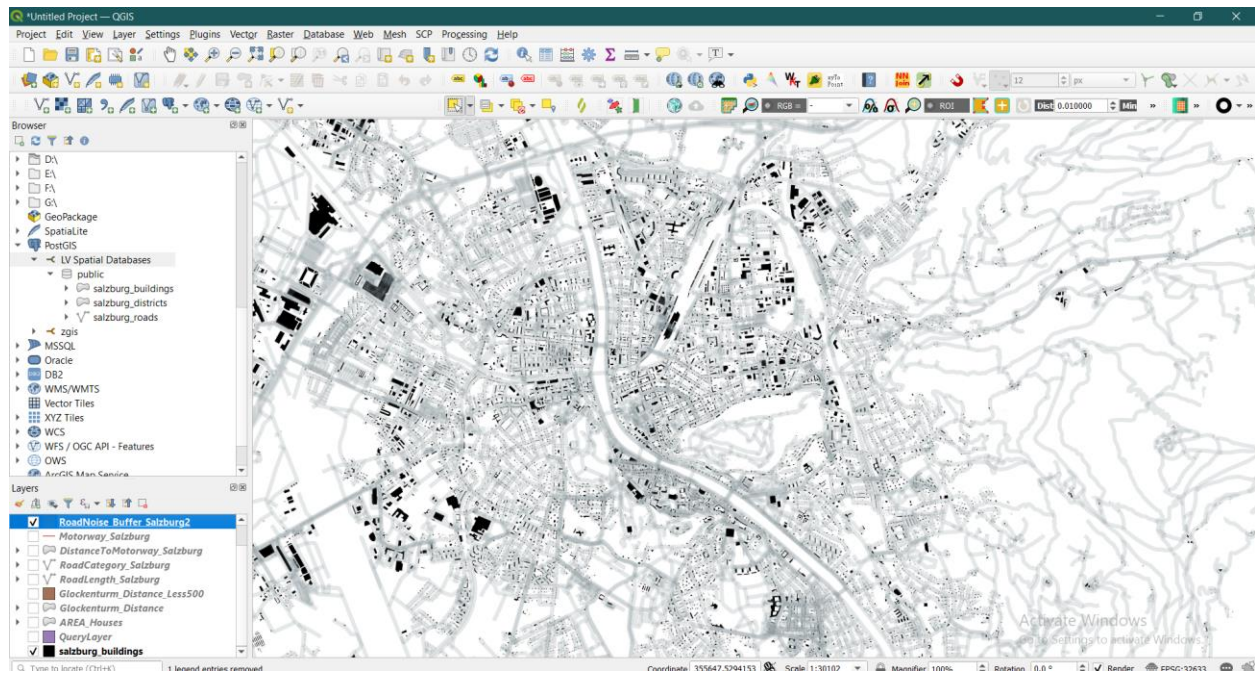
```
1 SELECT
2 *
3 ST_BUFFER(r.geom, 20, 'endcap=flat') AS potential_noise
4 FROM
5 salzburg_roads AS r;
```

The query results are displayed in a table with the following columns: id, geom, osm\_id, code, fclass, name, ref, oneway, maxspeed, layer, bridge, tunnel, orig\_fid, and potential\_noise. The table contains 6 rows of data.

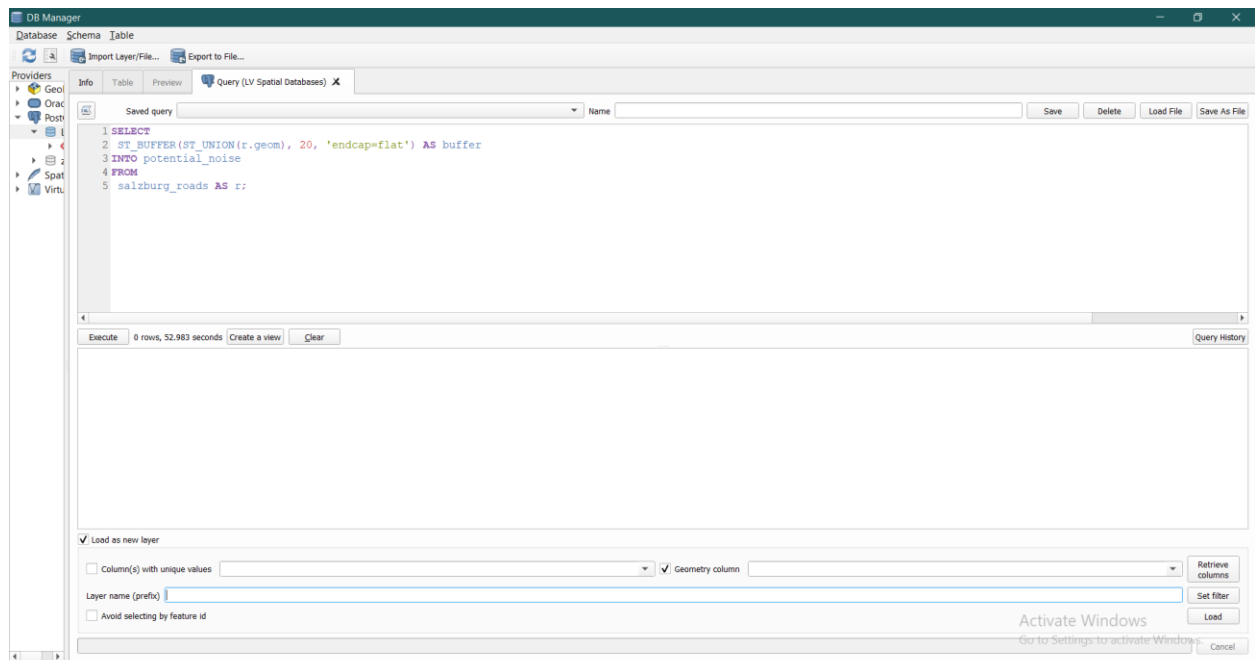
id	geom	osm_id	code	fclass	name	ref	oneway	maxspeed	layer	bridge	tunnel	orig_fid	potential_noise
1	0102000020797...	3128234	5111	motorway	West Autobahn	A1	F	100	0	F	F	0	0103000020797...
2	0102000020797...	10710431	5131	motorway_link	NULL	A1	F	100	0	F	F	1	0103000020797...
3	0102000020797...	10710442	5131	motorway_link	NULL	A1	F	80	0	F	F	2	0103000020797...
4	0102000020797...	10710455	5131	motorway_link	NULL	A1	F	80	0	F	F	3	0103000020797...
5	0102000020797...	10710470	5131	motorway_link	NULL	A1	F	80	0	F	F	4	0103000020797...
6	0102000020797...	10710477	5131	motorway_link	NULL	A1	F	80	0	F	F	5	0103000020797...

The query results are displayed in a table with the following columns: id, geom, osm\_id, code, fclass, name, ref, oneway, maxspeed, layer, bridge, tunnel, orig\_fid, and potential\_noise. The table contains 6 rows of data.

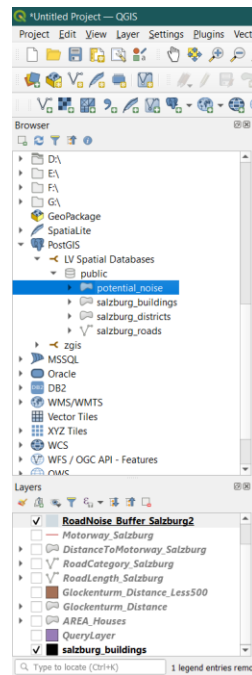




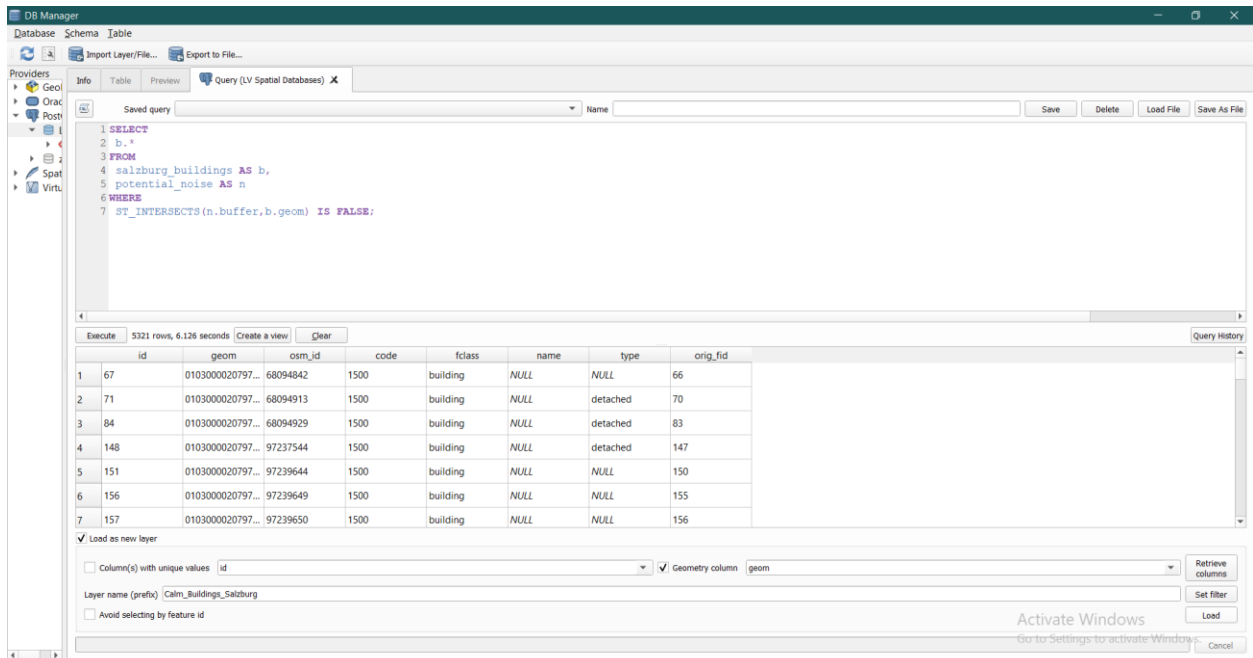
However, it seems that the performance of query would be so long for the next step! So, we can first create a table for the 20 meter potential noise buffer by SELECT.... INTO.... FROM.....

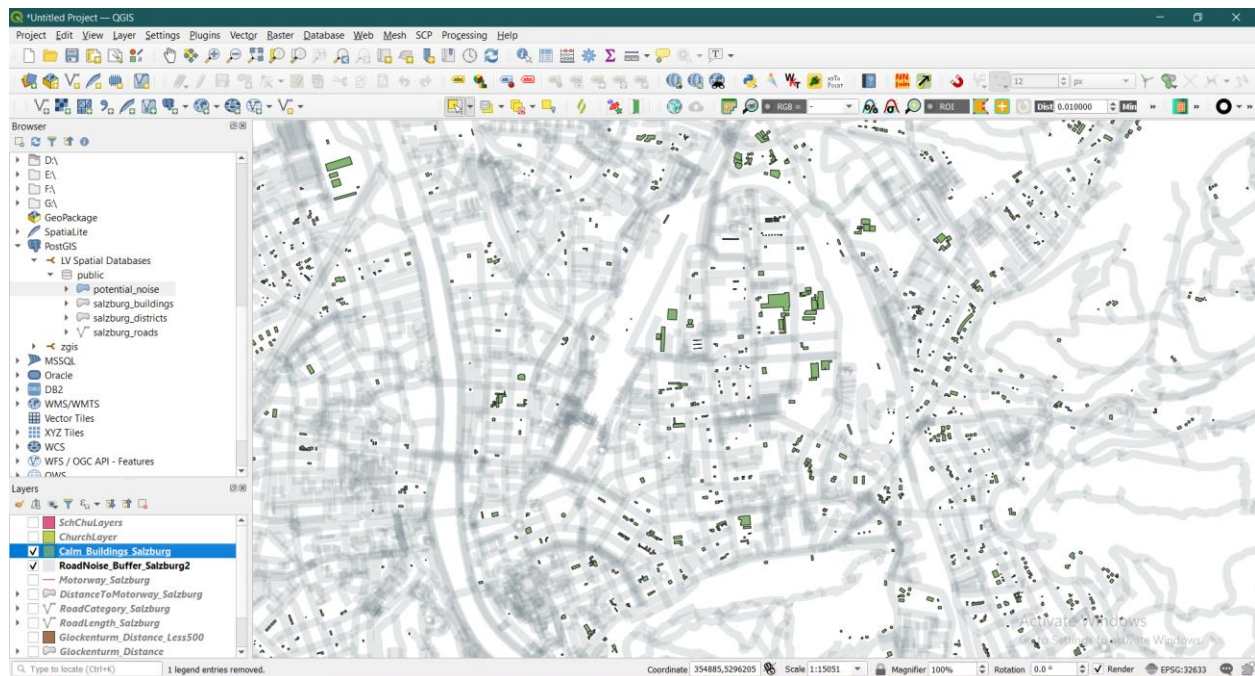


The created table is in the database now:



Then we intersect 2 tables, where the buildings with intersections with the buffer would be gone:





- *Think of three situations in which having GIS operations written as SQL queries can be beneficial and has advantages*

So, I can say if we imagine we have different type of shapefiles like point, polyline and polygon. Each could be like population and average income, road type, region. If you have a specific purpose, like the number of people in a certain region, living in 100-meter distance from the road, which consider all these 3 shapefiles together, SQL helps to decrease the computations for each shapefile and minimize the number of steps you need to pass.

Also, you can search easier for multiplied features in the layer with SQL, making distinct selection or skip them in some calculations. There may be some errors in the layer like some multiplied buildings attribute and you want to calculate the ratio of houses in each district. It certainly effect on the final value.