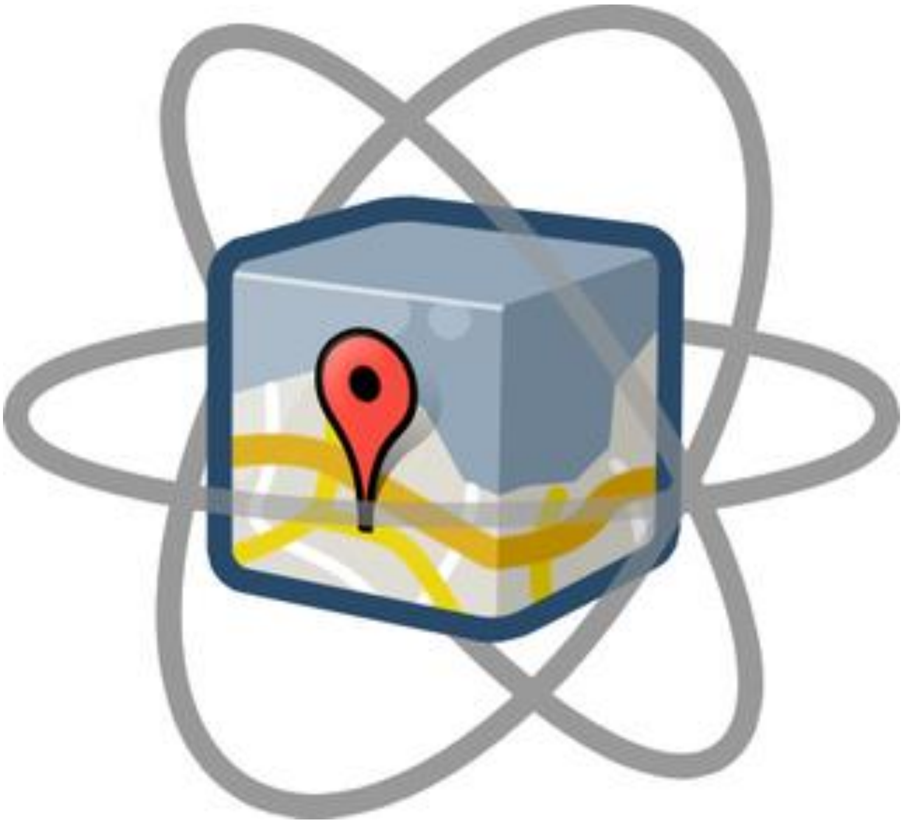
COPERNICUS MASTER
IN DIGITAL EARTH

SPATIAL DATABASES



EXERCISE 06: SQL FOR SPATIAL QUERIES

KHIZER ZAKIR

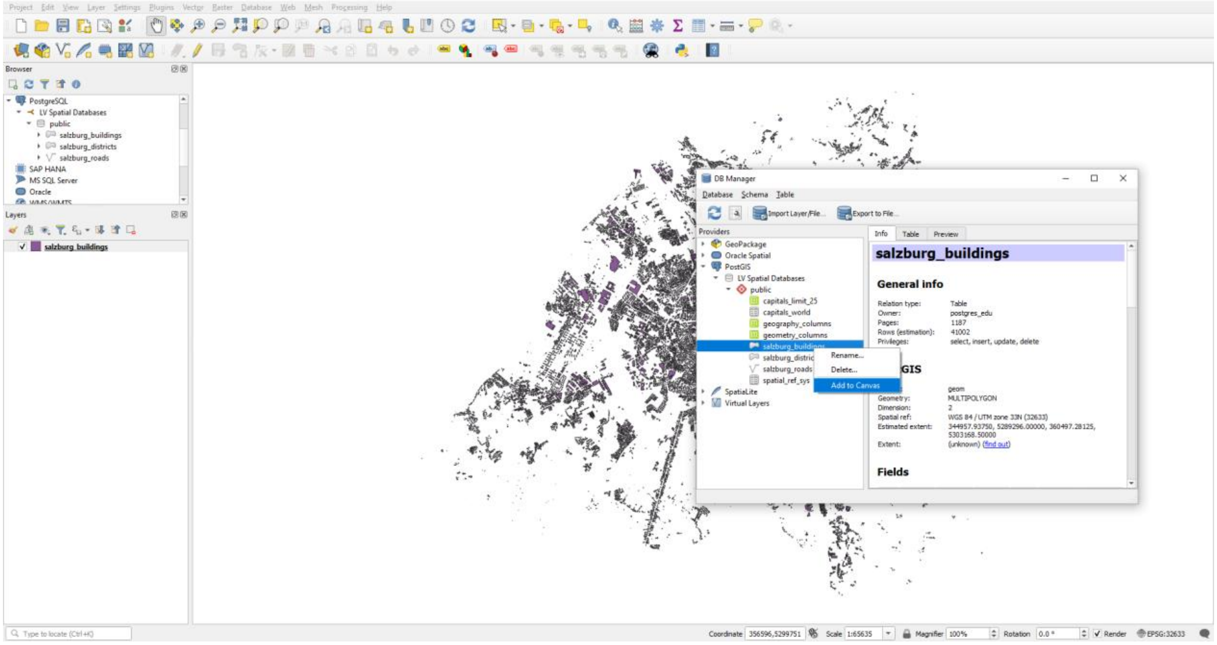
UNIVERSITY OF SALZBURG

13/12/2022

Note: The zip file contains all the ".SQL" files; The files are named according to the list of tasks in this document. For example, "task1.sql" corresponds to the screenshots for task1 in this document. The QGIS project has also been zipped in the folder. The queries were performed several times for practice; hence, the SS may not be coherent.

Task 1:

Use a PostGIS table for visualisation as layer in QGIS; Add to Canvas:



salzburg_buildings

General info

Relation type: Table
Owner: postgres_edu
Pages: 1187
Rows (estimation): 41002
Privileges: select, insert, update, delete

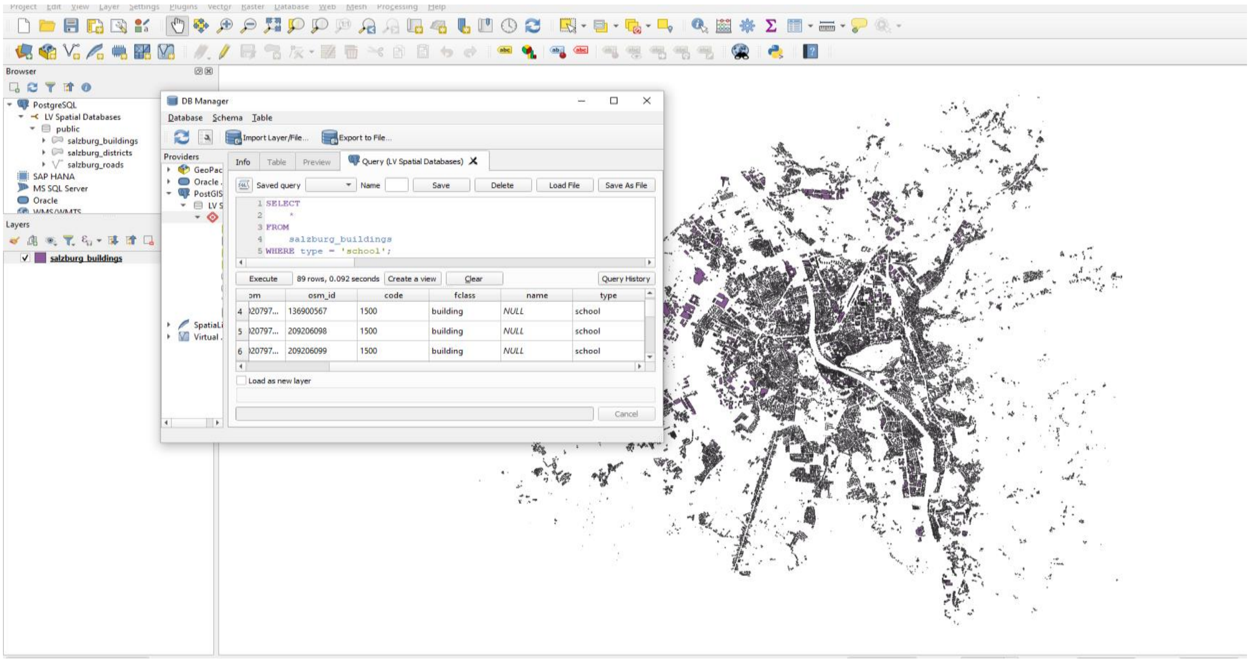
GIS

geom
Geometry: MULTIPOLYGON
Dimension: 2
Spatial ref: WGS 84 / UTM zone 33N (32633)
Estimated extent: 344957.93750, 5289296.00000, 360497.28125, 5303168.50000
Extent: (unknown) (find out)

Fields

Task2:

Selecting Data by Attribute value and creating new layer(s) - Selecting 'School' from *salzburg_buildings*:



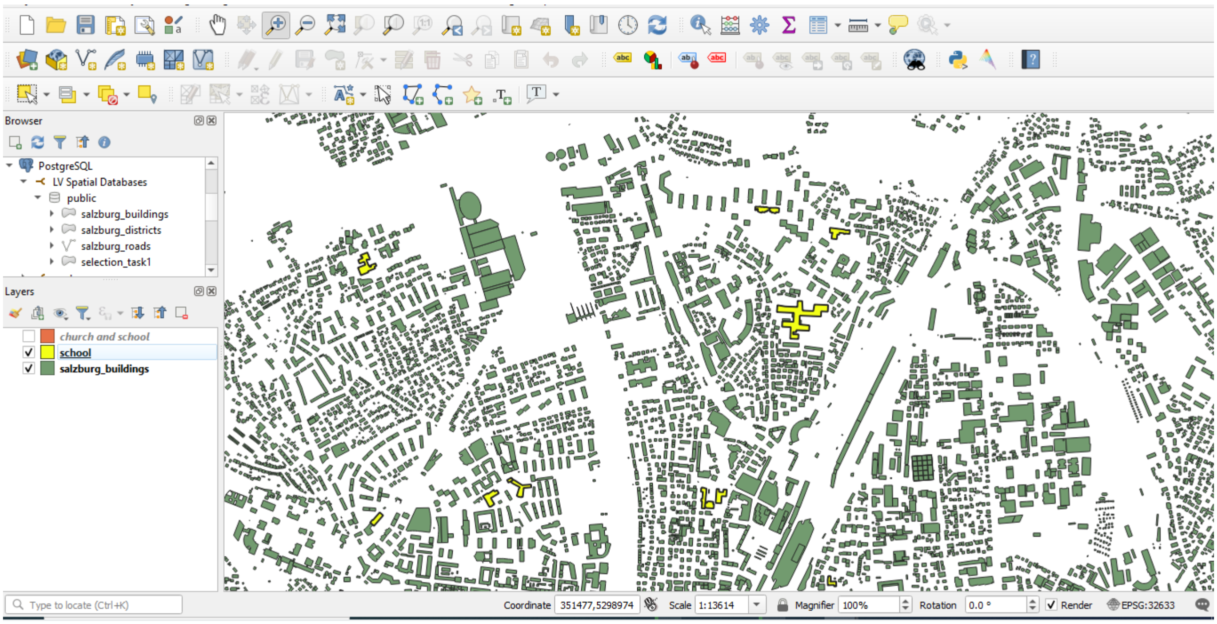
```
SELECT
    *
FROM
    salzburg_buildings
WHERE type = 'school';
```

89 rows, 0.092 seconds

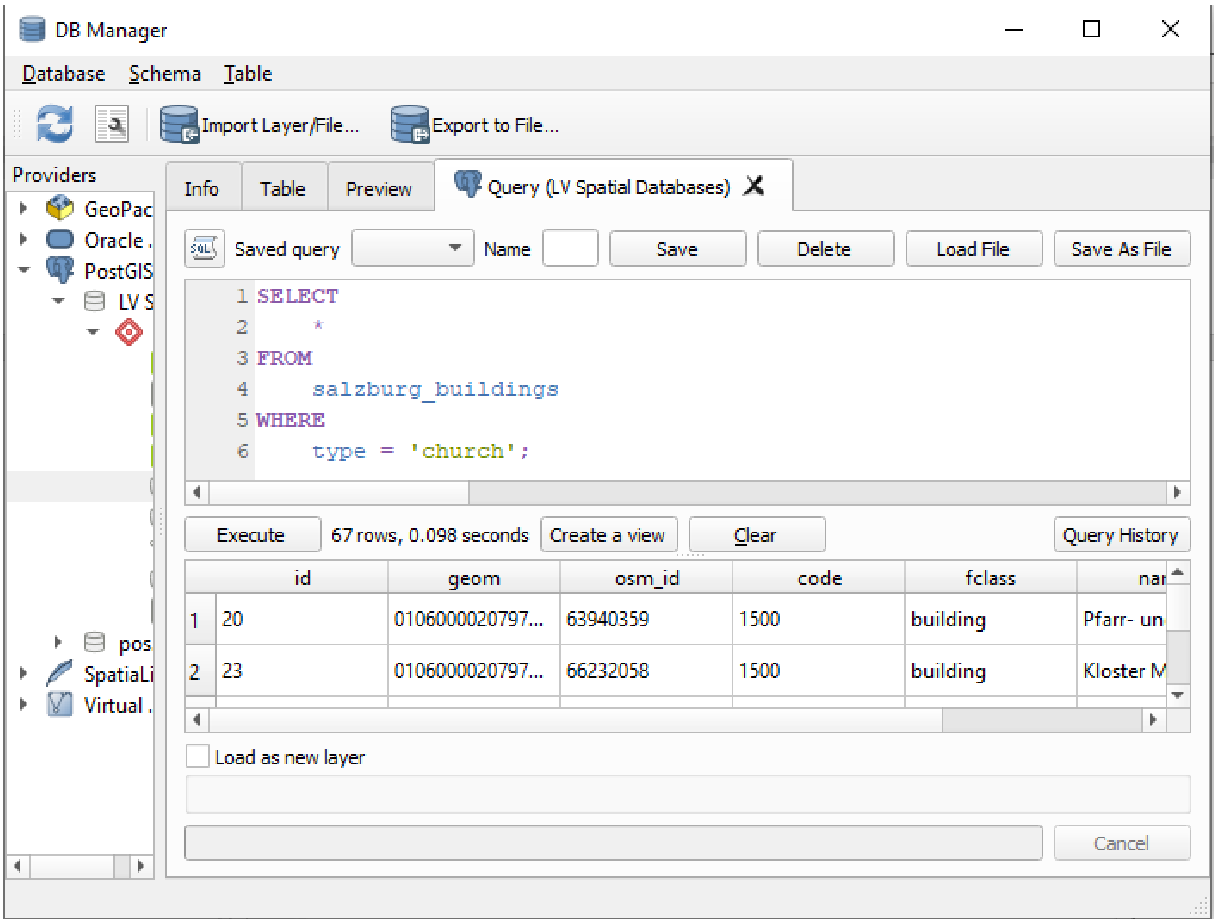
	om	osm_id	code	fclass	name	type
4	20797...	136900567	1500	building	NULL	school
5	20797...	209206098	1500	building	NULL	school
6	20797...	209206099	1500	building	NULL	school

□ Load as new layer

Loading schools and creating new layer in QGIS:



Query and Layer for type = ‘church’

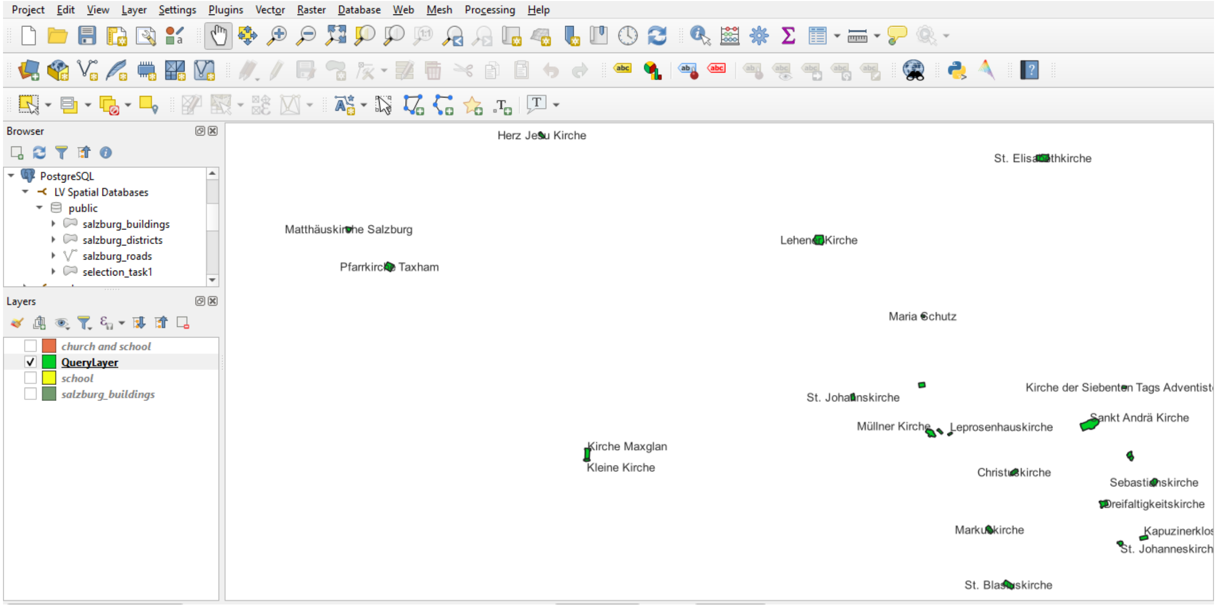


```
SELECT
    *
FROM
    salzburg_buildings
WHERE
    type = 'church';
```

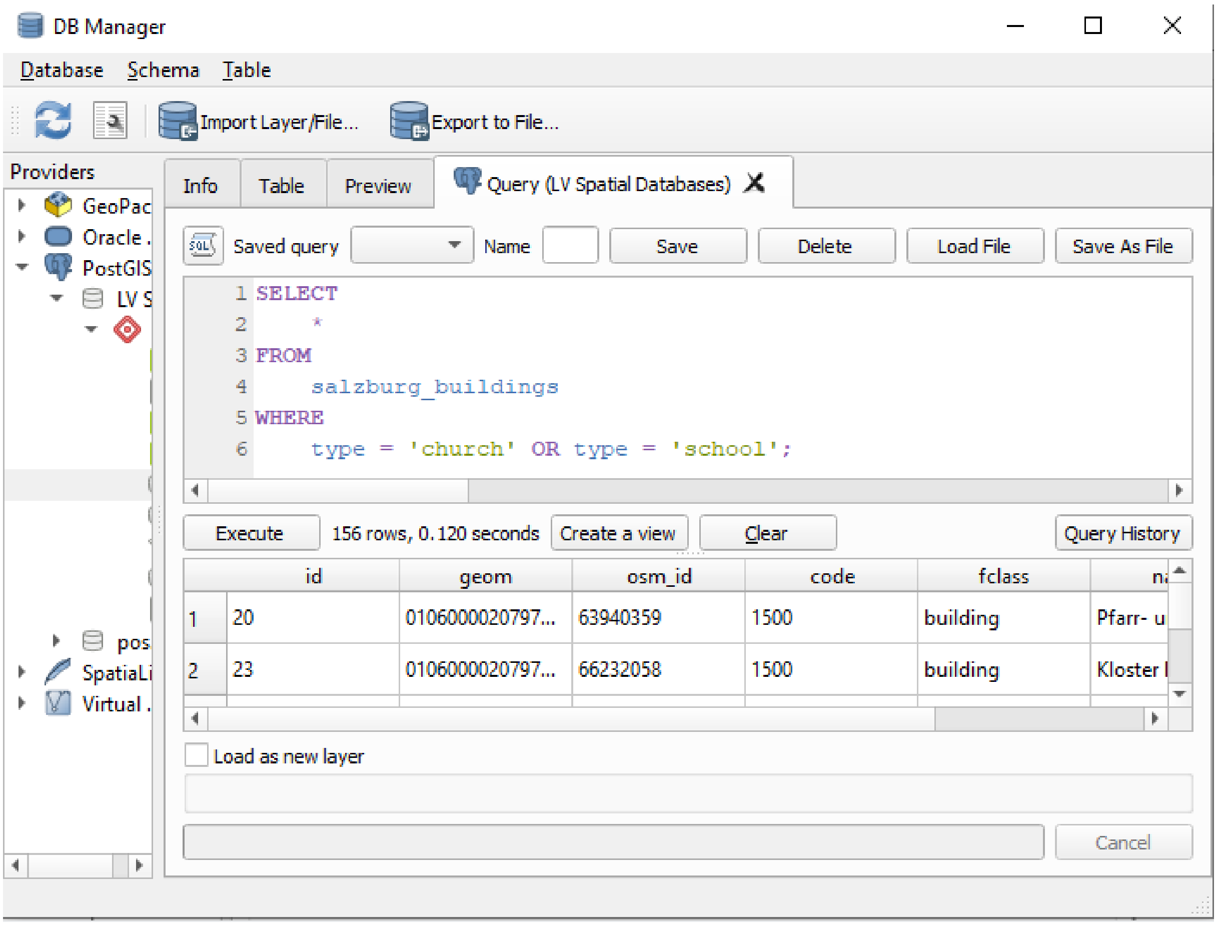
67 rows, 0.098 seconds

	id	geom	osm_id	code	fclass	nar
1	20	0106000020797...	63940359	1500	building	Pfarr- un
2	23	0106000020797...	66232058	1500	building	Kloster M

☐ Load as new layer



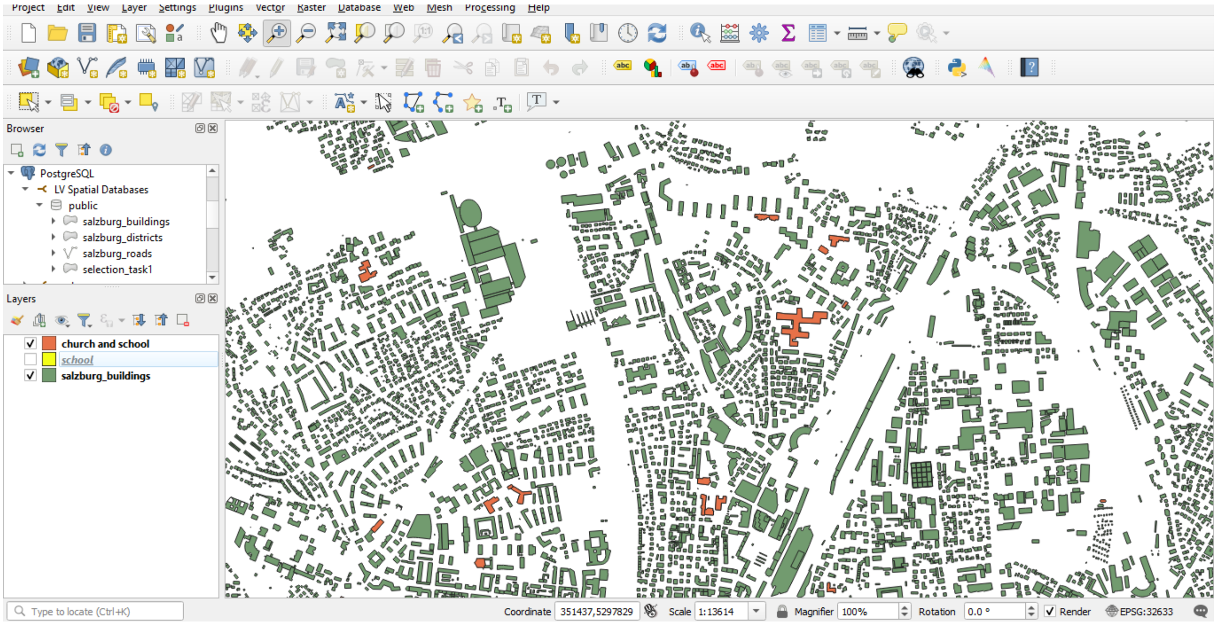
Church and School together:



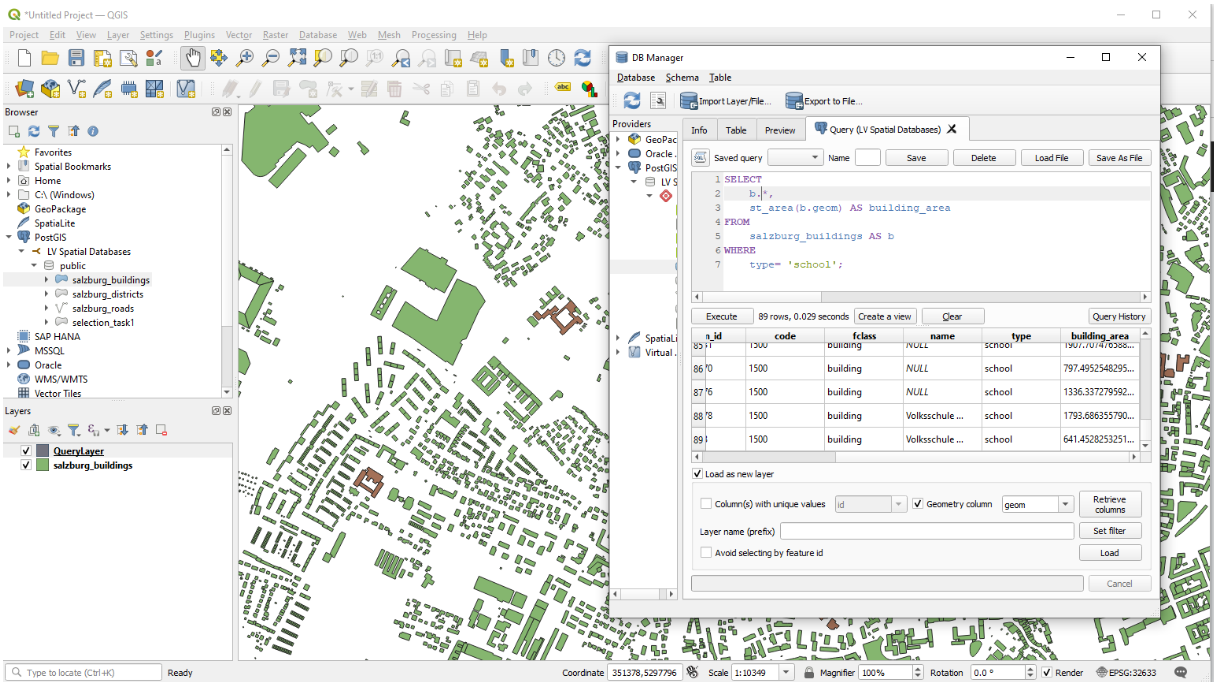
```
SELECT
    *
FROM
    salzburg_buildings
WHERE
    type = 'church' OR type = 'school';
```

156 rows, 0.120 seconds

	id	geom	osm_id	code	fclass	na
1	20	0106000020797...	63940359	1500	building	Pfarr- u
2	23	0106000020797...	66232058	1500	building	Kloster I



Area of School:



```
SELECT
    b.*,
    st_area(b.geom) AS building_area
FROM
    salzburg_buildings AS b
WHERE
    type= 'school';
```

89 rows, 0.029 seconds

	n_id	code	fclass	name	type	building_area
85	[illegible]	1500	building	NULL	school	1907.707476588...
86	70	1500	building	NULL	school	797.4952548295...
87	76	1500	building	NULL	school	1336.337279592...
88	78	1500	building	Volksschule ...	school	1793.686355790...
89	[illegible]	1500	building	Volksschule ...	school	641.4528253251...

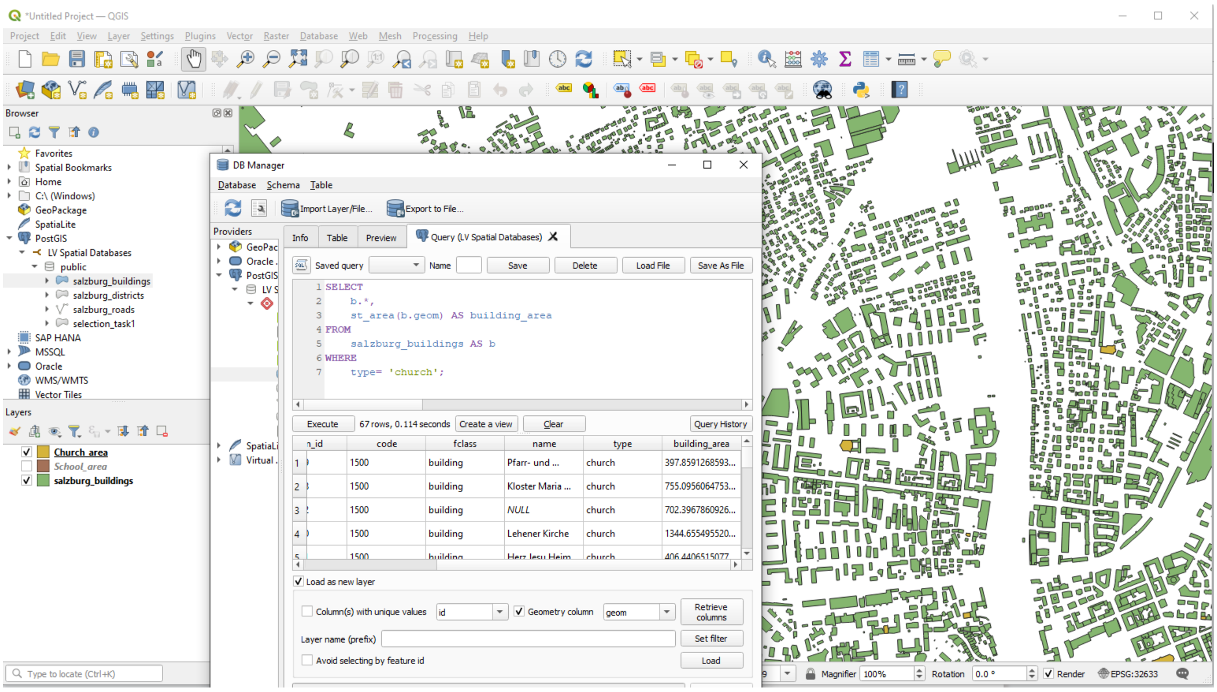
☑ Load as new layer

☐ Column(s) with unique values: id ☑ Geometry column: geom

Layer name (prefix):

☐ Avoid selecting by feature id

Area of church:



```
SELECT
    b.*,
    st_area(b.geom) AS building_area
FROM
    salzburg_buildings AS b
WHERE
    type= 'church';
```

Execute | 67 rows, 0.114 seconds | Create a view | Clear | Query History

_id	code	fclass	name	type	building_area
1	1500	building	Pfarr- und ...	church	397.8591268593...
2	1500	building	Kloster Maria ...	church	755.0956064753...
3	1500	building	*NULL*	church	702.3967860926...
4	1500	building	Lehener Kirche	church	1344.655495520...
5	1500	building	Herz Jesu Heim	church	406.4406515077

☑ Load as new layer

☐ Column(s) with unique values: id ☑ Geometry column: geom

Layer name (prefix):

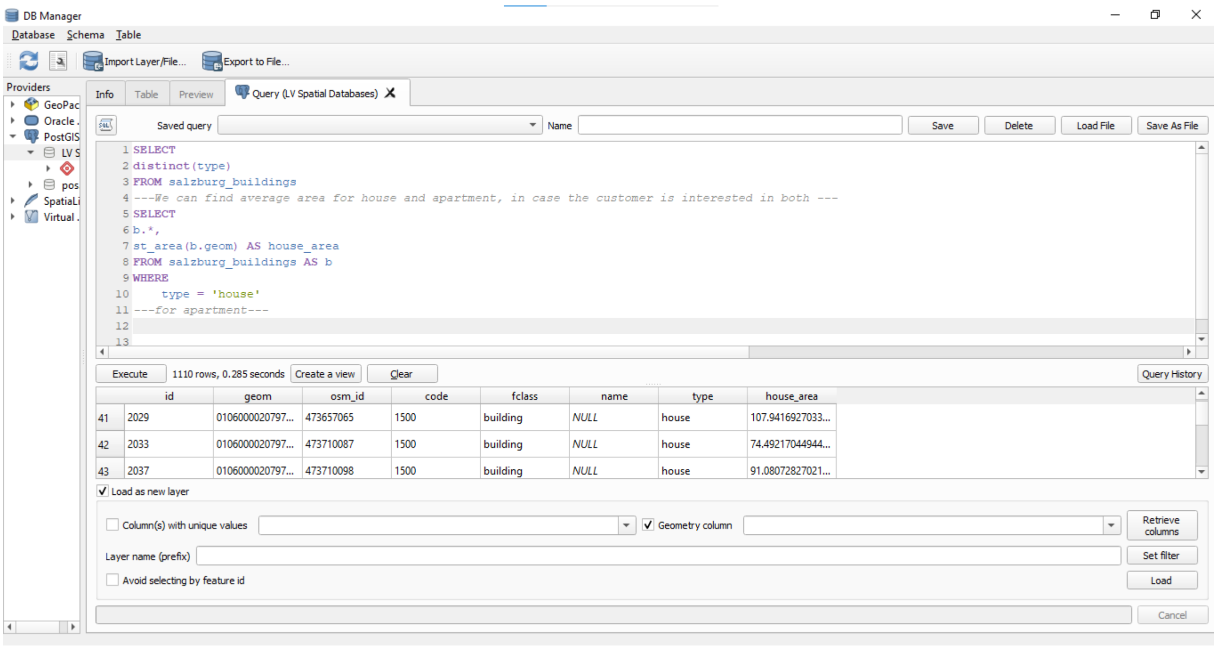
☐ Avoid selecting by feature id

Task 3:

Selecting data by spatial properties: Select type = ‘houses’ by area

Area for the type: ‘house’ and/or ‘apartment’

Query to estimate area for the type: ‘house’ and ‘apartment’



```
SELECT
distinct(type)
FROM salzburg_buildings
---We can find average area for house and apartment, in case the customer is interested in both ---
SELECT
b.*,
st_area(b.geom) AS house_area
FROM salzburg_buildings AS b
WHERE
    type = 'house'
---for apartment---
```

Execute | 1110 rows, 0.285 seconds | Create a view | Clear | Query History

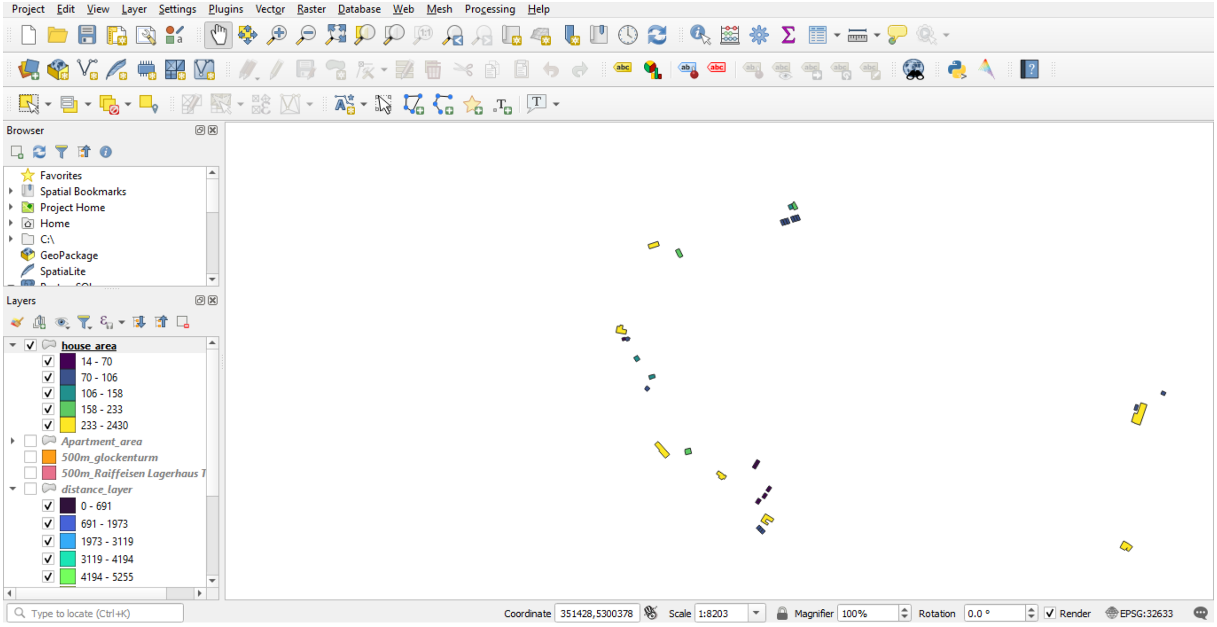
	id	geom	osm_id	code	fclass	name	type	house_area
41	2029	0106000020797...	473657065	1500	building	*NULL*	house	107.9416927033...
42	2033	0106000020797...	473710087	1500	building	*NULL*	house	74.49217044944...
43	2037	0106000020797...	473710098	1500	building	*NULL*	house	91.08072827021...

☑ Load as new layer

☐ Column(s) with unique values ☑ Geometry column

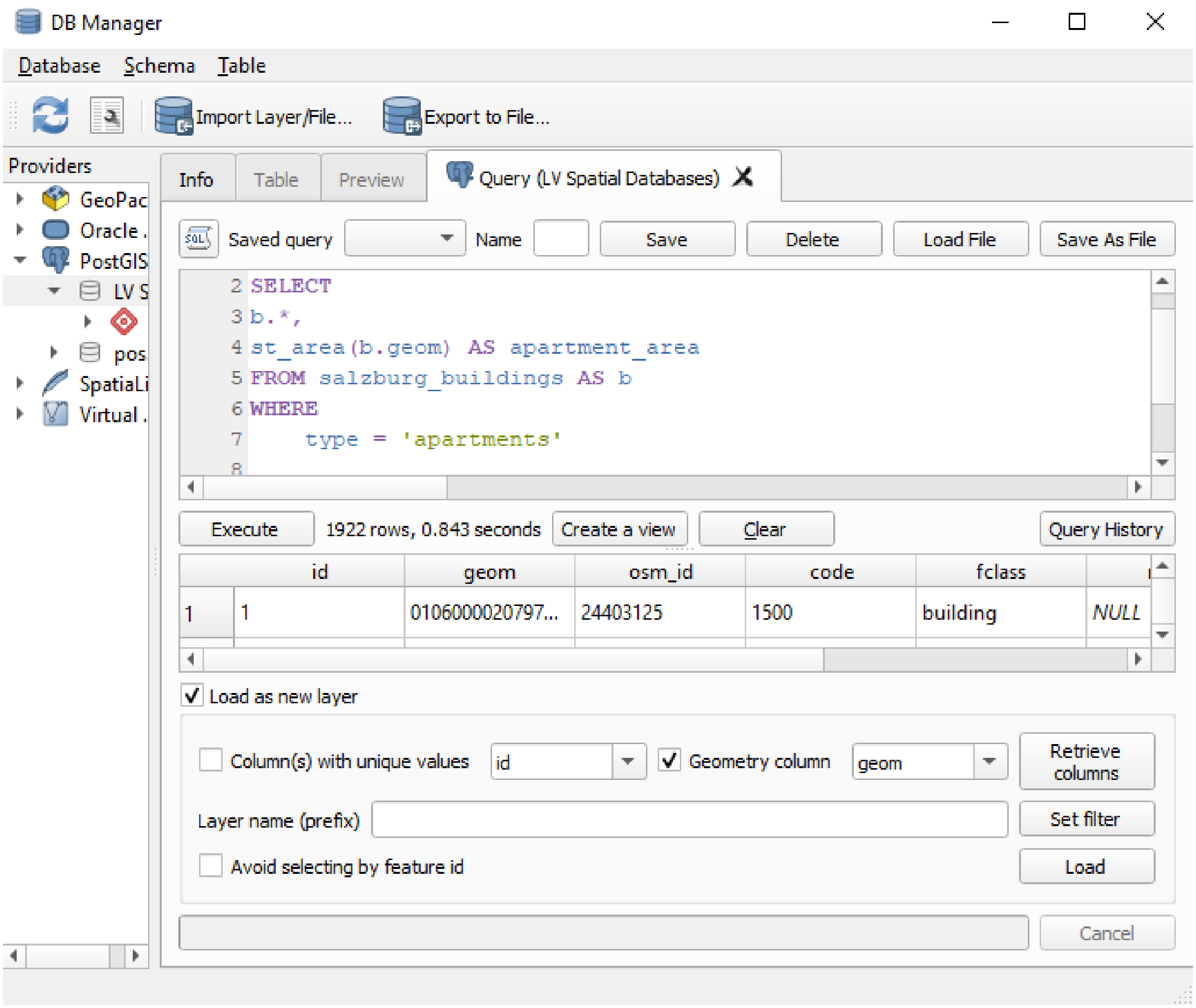
Layer name (prefix):

☐ Avoid selecting by feature id



- ☑ house_area
  - ☑ 14 - 70
  - ☑ 70 - 106
  - ☑ 106 - 158
  - ☑ 158 - 233
  - ☑ 233 - 2430
- ☐ *Apartment_area*
- ☐ *500m_glockenturm*
- ☐ *500m_Raiffeisen Lagerhaus T*
- ☐ *distance_layer*
  - ☑ 0 - 691
  - ☑ 691 - 1973
  - ☑ 1973 - 3119
  - ☑ 3119 - 4194
  - ☑ 4194 - 5255

Coordinate 351428,5300378 Scale 1:8203 Magnifier 100% Rotation 0.0 ° ☑ Render EPSG:32633



```
SELECT
b.*,
st_area(b.geom) AS apartment_area
FROM salzburg_buildings AS b
WHERE
    type = 'apartments'
```

1922 rows, 0.843 seconds

	id	geom	osm_id	code	fclass
1	1	0106000020797...	24403125	1500	building

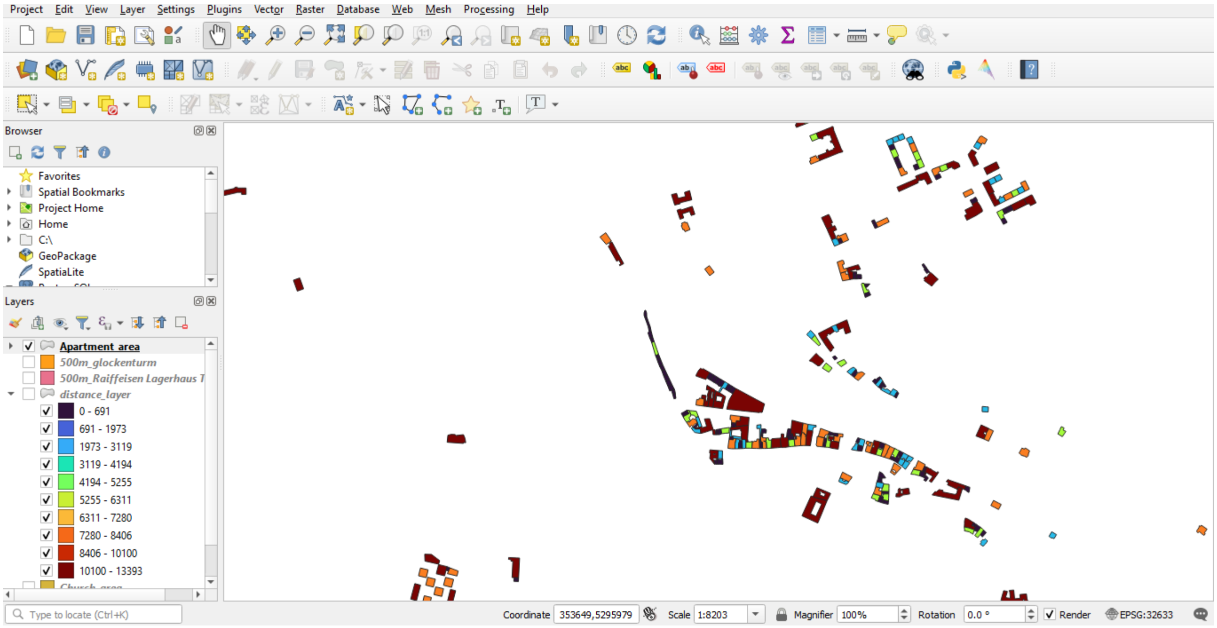
☑ Load as new layer

☐ Column(s) with unique values: id

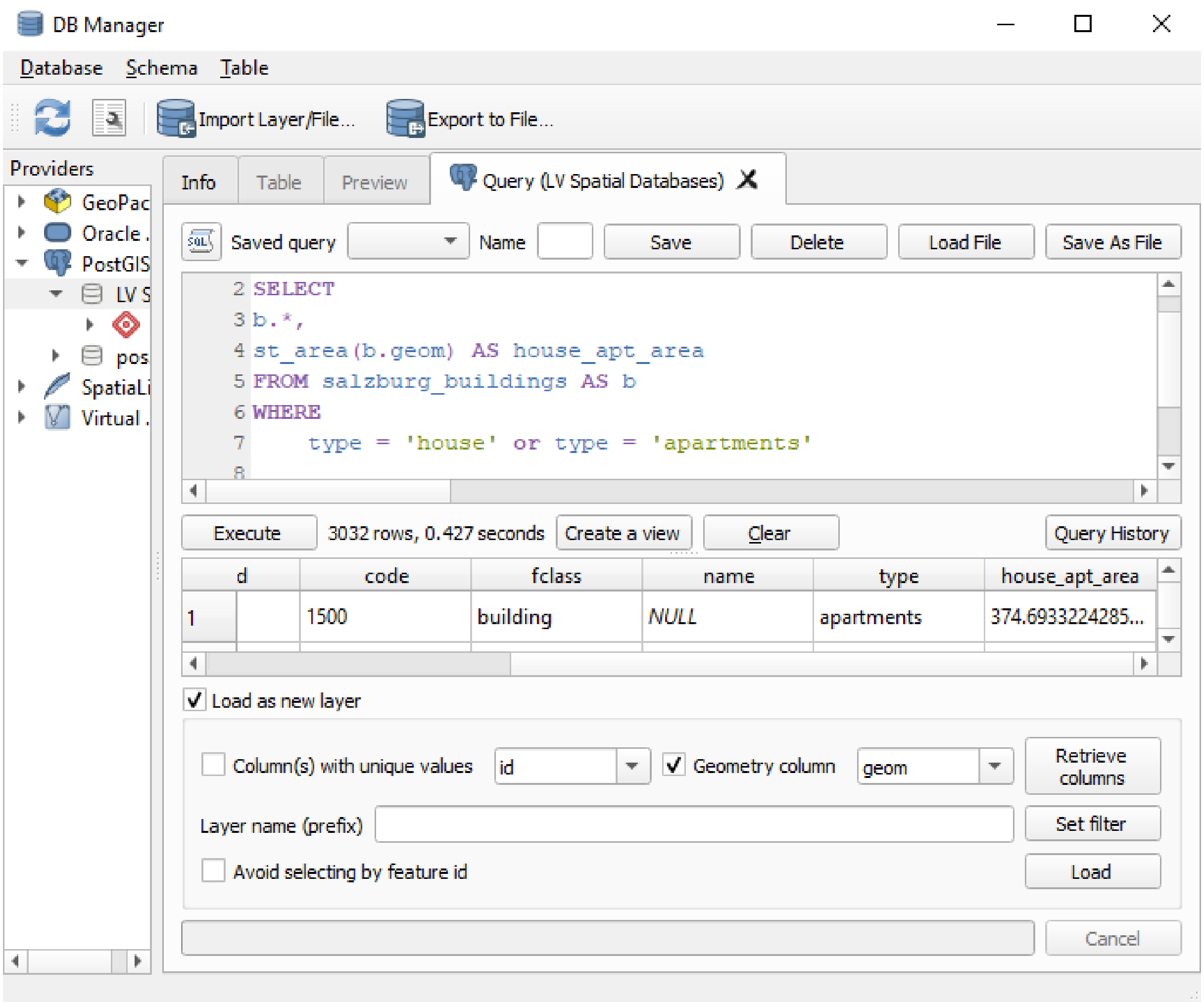
☑ Geometry column: geom

Layer name (prefix):

☐ Avoid selecting by feature id



Query for “house” and “apartment” together:



```
SELECT
b.*,
st_area(b.geom) AS house_apt_area
FROM salzburg_buildings AS b
WHERE
    type = 'house' or type = 'apartments'
```

3032 rows, 0.427 seconds

	d	code	fclass	name	type	house_apt_area
1		1500	building	NULL	apartments	374.6933224285...

☑ Load as new layer

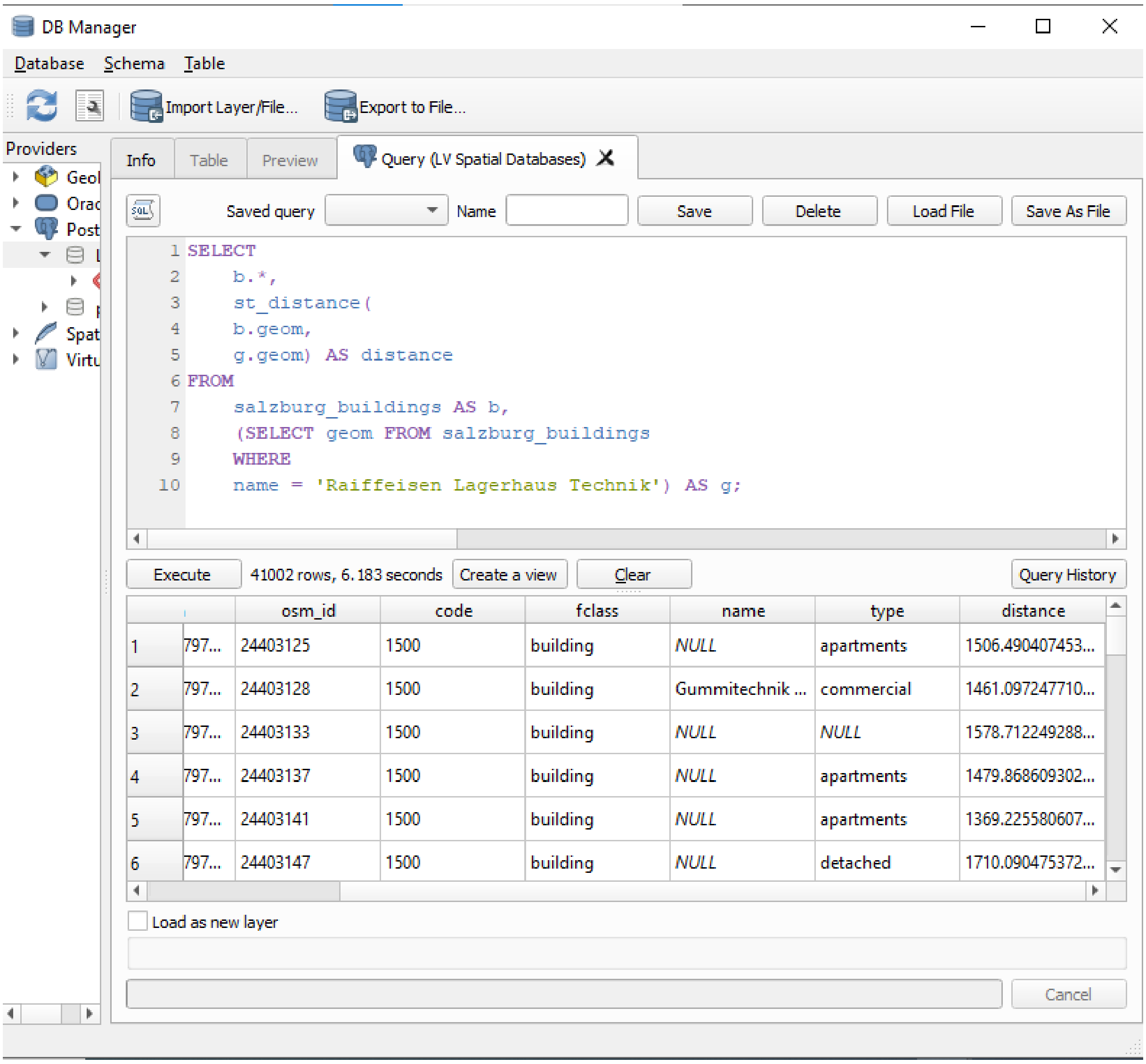
□ Column(s) with unique values: id ☑ Geometry column: geom

Layer name (prefix)

□ Avoid selecting by feature id

Task 4:

Selecting data by location

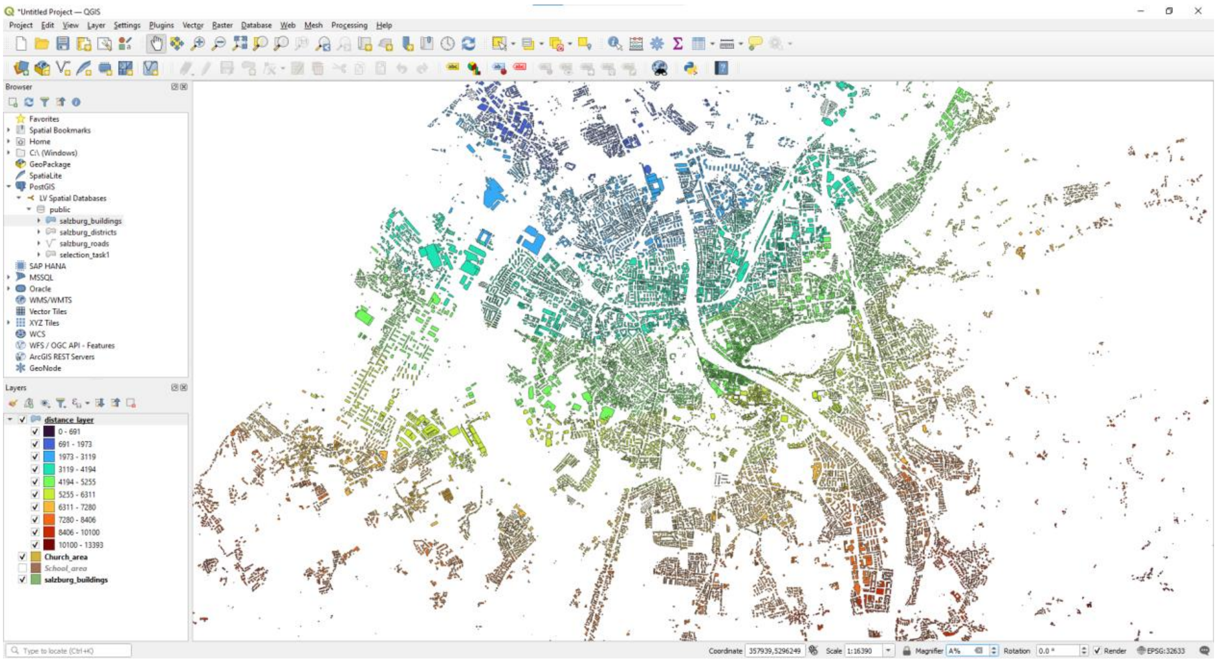


```
SELECT
    b.*,
    st_distance(
    b.geom,
    g.geom) AS distance
FROM
    salzburg_buildings AS b,
    (SELECT geom FROM salzburg_buildings
    WHERE
    name = 'Raiffeisen Lagerhaus Technik') AS g;
```

41002 rows, 6.183 seconds

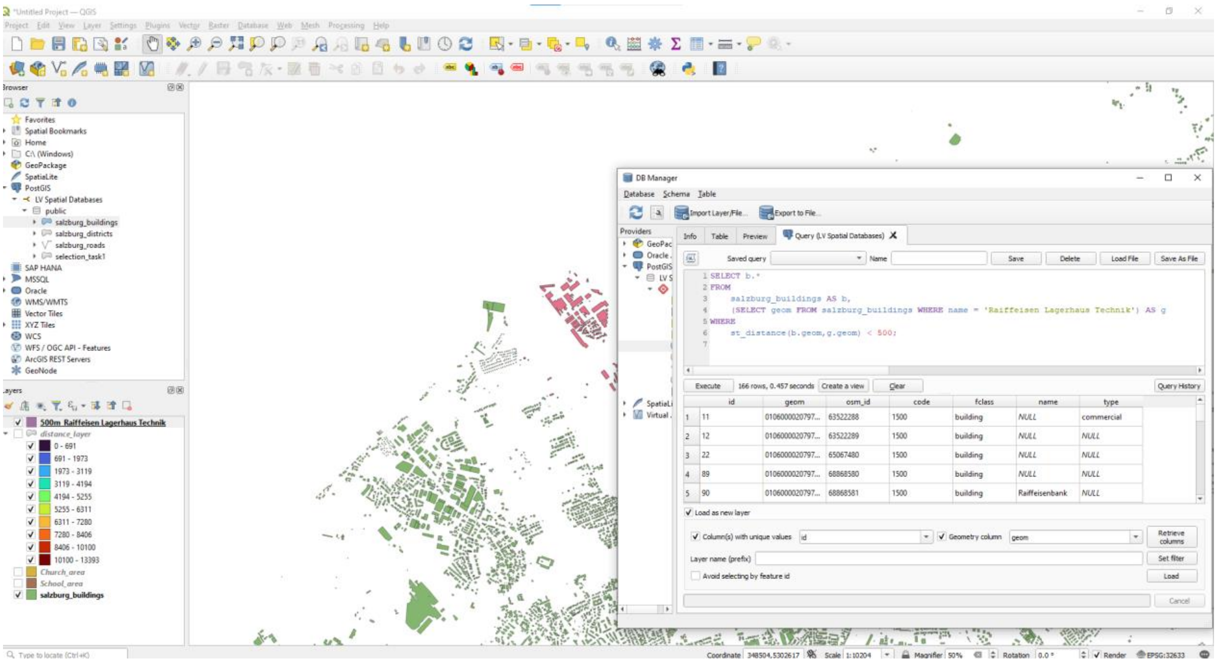
		osm_id	code	fclass	name	type	distance
1	797...	24403125	1500	building	NULL	apartments	1506.490407453...
2	797...	24403128	1500	building	Gummitechnik ...	commercial	1461.097247710...
3	797...	24403133	1500	building	NULL	NULL	1578.712249288...
4	797...	24403137	1500	building	NULL	apartments	1479.868609302...
5	797...	24403141	1500	building	NULL	apartments	1369.225580607...
6	797...	24403147	1500	building	NULL	detached	1710.090475372...

layers with natural breaks: 10 classes:



Task 5:

Layer and query for buildings at less than 500m distance from the chosen location.



```
SELECT b.*
FROM
    salzburg_buildings AS b,
    (SELECT geom FROM salzburg_buildings WHERE name = 'Raiffeisen Lagerhaus Technik') AS g
WHERE
    st_distance(b.geom,g.geom) < 500;
```

166 rows, 0.457 seconds

	id	geom	osm_id	code	fclass	name	type
1	11	0106000020797...	63522288	1500	building	NULL	commercial
2	12	0106000020797...	63522289	1500	building	NULL	NULL
3	22	0106000020797...	65067480	1500	building	NULL	NULL
4	89	0106000020797...	68868580	1500	building	NULL	NULL
5	90	0106000020797...	68868581	1500	building	Raiffeisenbank	NULL

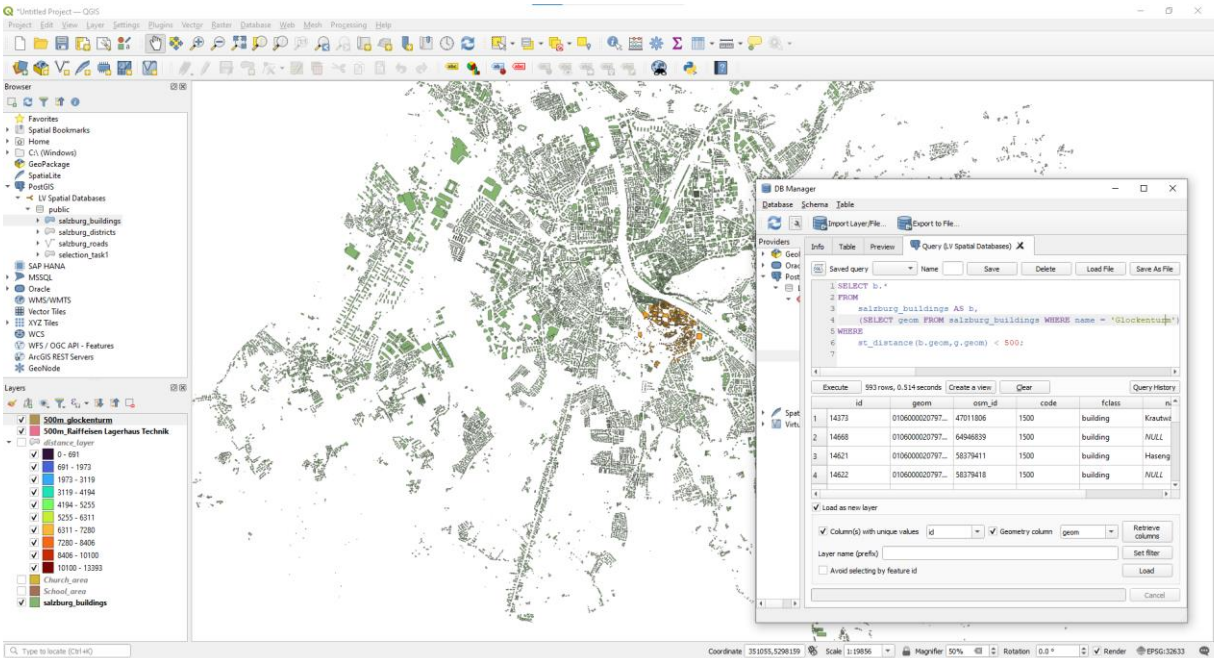
☑ Load as new layer

☑ Column(s) with unique values: id ☑ Geometry column: geom

Layer name (prefix):

□ Avoid selecting by feature id

At less than 500m distance from the city centre:



```
SELECT b.*
FROM
    salzburg_buildings AS b,
    (SELECT geom FROM salzburg_buildings WHERE name = 'Glockenturm')
WHERE
    st_distance(b.geom,g.geom) < 500;
```

593 rows, 0.514 seconds

	id	geom	osm_id	code	fclass	n
1	14373	0106000020797...	47011806	1500	building	Krautw
2	14668	0106000020797...	64946839	1500	building	NULL
3	14621	0106000020797...	58379411	1500	building	Haseng
4	14622	0106000020797...	58379418	1500	building	NULL

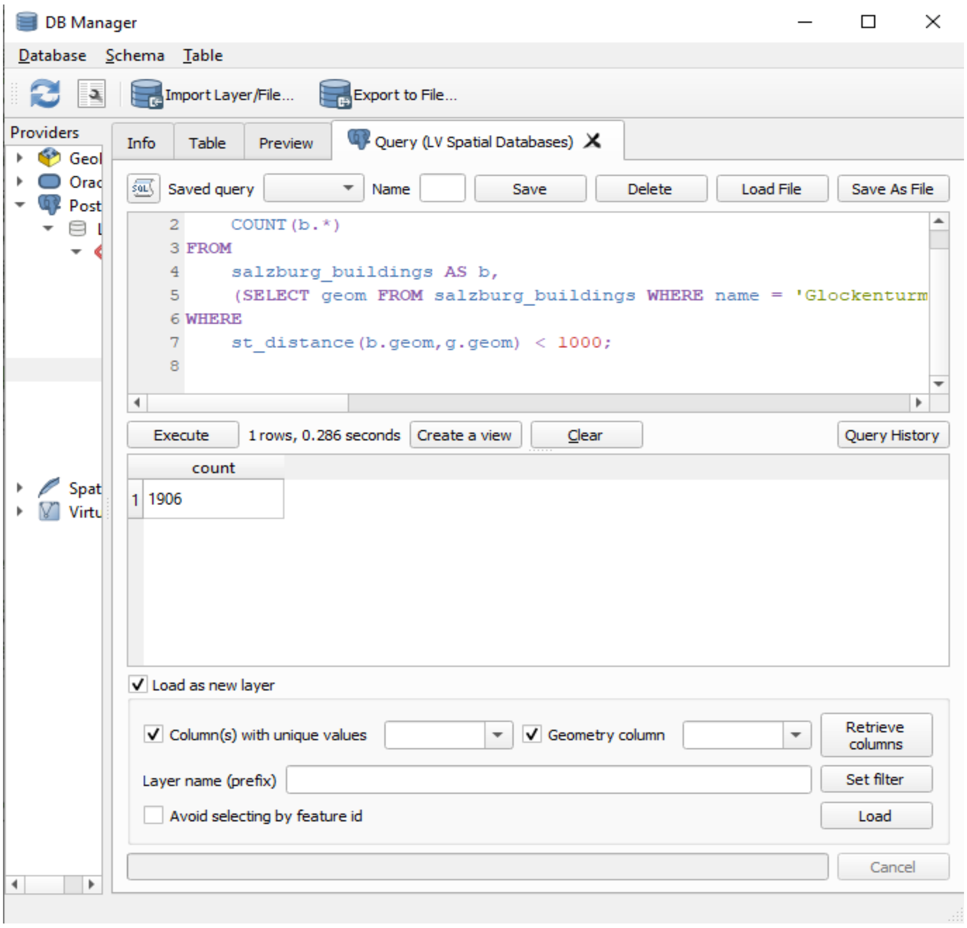
☑ Load as new layer

☑ Column(s) with unique values: id ☑ Geometry column: geom

☐ Avoid selecting by feature id

Task 6:

Count the number of buildings at less than 1000m or 1 km distance from the city centre.



```
    COUNT(b.*)
FROM
    salzburg_buildings AS b,
    (SELECT geom FROM salzburg_buildings WHERE name = 'Glockenturm
WHERE
    st_distance(b.geom,g.geom) < 1000;
```

1 rows, 0.286 seconds

	count
1	1906

☑ Load as new layer

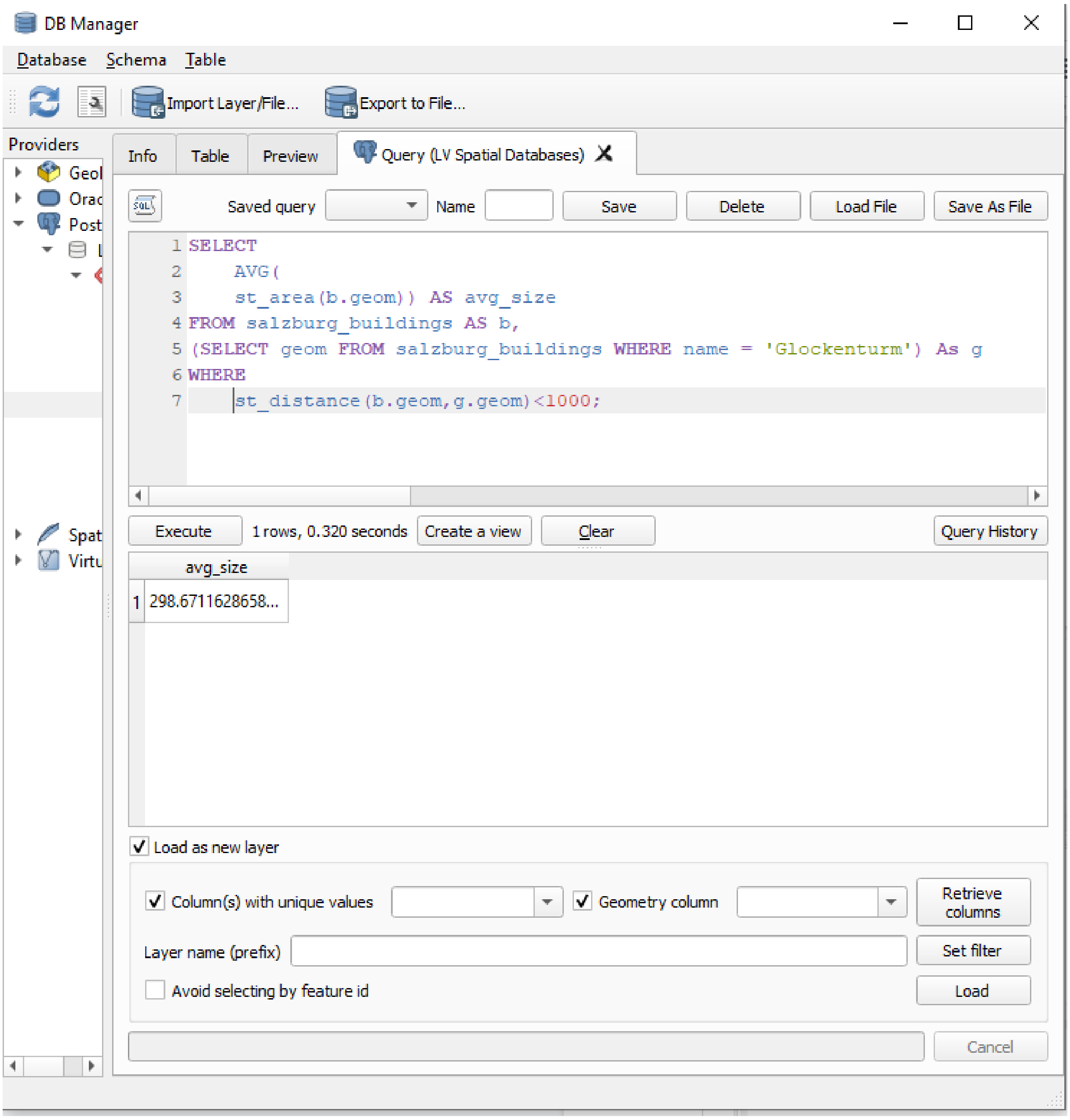
☑ Column(s) with unique values ☑ Geometry column

Layer name (prefix)

☐ Avoid selecting by feature id

Task 7:

Average area/size of buildings that are at less than 1000m distance from city centre and the chosen location.



```
SELECT
    AVG(
    st_area(b.geom)) AS avg_size
FROM salzburg_buildings AS b,
(SELECT geom FROM salzburg_buildings WHERE name = 'Glockenturm') As g
WHERE
    st_distance(b.geom,g.geom)<1000;
```

1 rows, 0.320 seconds

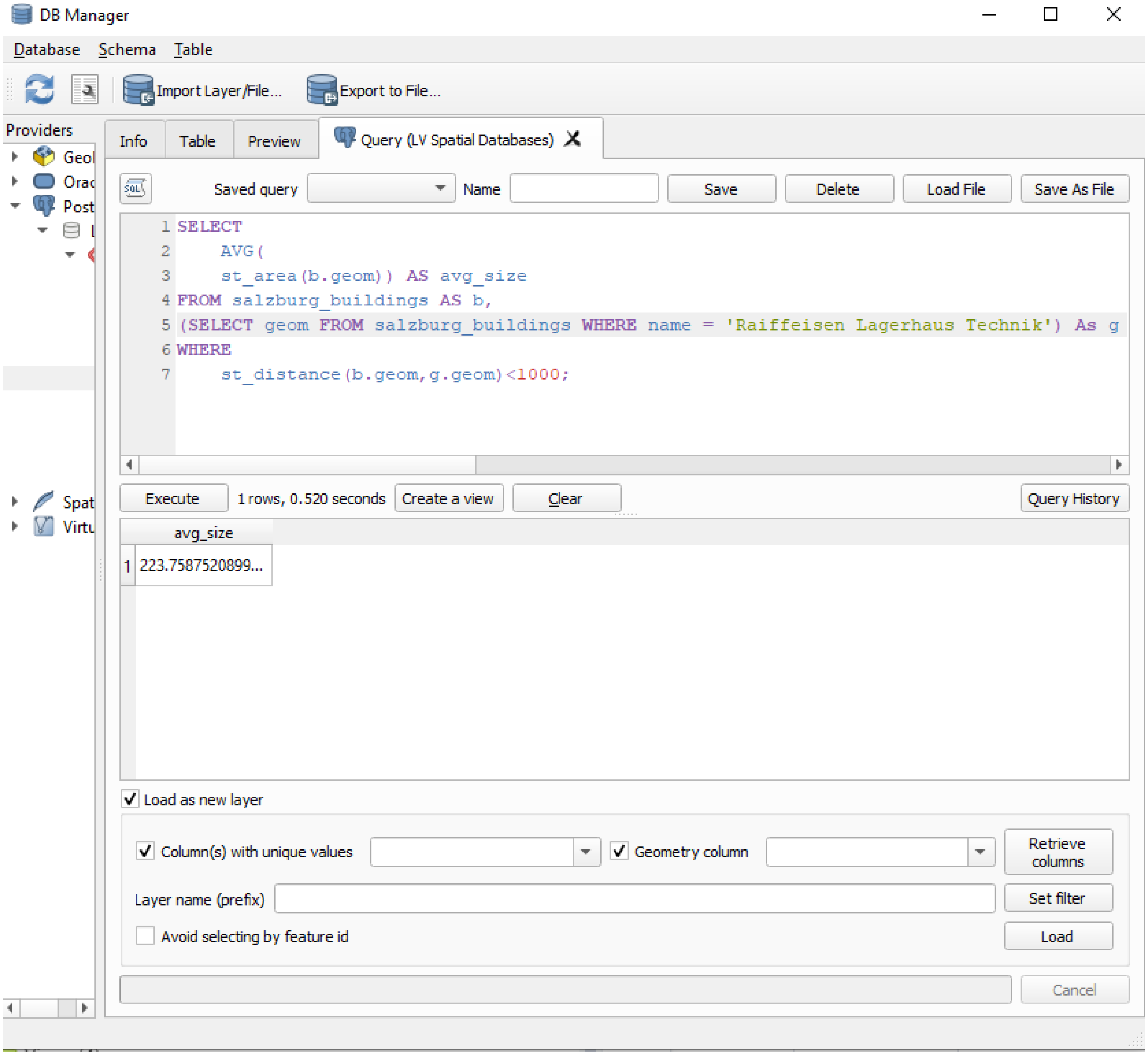
	avg_size
1	298.6711628658...

☑ Load as new layer

☑ Column(s) with unique values ☑ Geometry column

Layer name (prefix)

□ Avoid selecting by feature id



```
SELECT
    AVG(
    st_area(b.geom)) AS avg_size
FROM salzburg_buildings AS b,
(SELECT geom FROM salzburg_buildings WHERE name = 'Raiffeisen Lagerhaus Technik') As g
WHERE
    st_distance(b.geom,g.geom)<1000;
```

1 rows, 0.520 seconds

	avg_size
1	223.7587520899...

☑ Load as new layer

☑ Column(s) with unique values ☑ Geometry column

Layer name (prefix)

□ Avoid selecting by feature id

Explanation:

As soon as you run the query, the system filters all the buildings within 1000m distance from the “location” (st_distance(b.geom, g.geom)). Next, st_area gives you the area of the filtered buildings; however, the AVG – arithmetic function calculates the average for all the values instantaneously.

This is one form of spatial sub-query or filtering in SQL. Noticeably, the arithmetic operations are used to filter the spatial properties from the database, while another spatial operation was introduced as a sub-query to complete the task. This query has a logical flow as well that is easily readable for any non-SQL users.