

The background features a dark grey field filled with a pattern of small, dark green dots. Overlaid on this are several white, dashed contour lines that meander across the space, primarily concentrated in the upper-left and lower-right corners. The text is centered in the middle of the image.

# Spatial Databases

+

Sample assignments

# Exercise 6: SQL-I

Mohamed Dhia TURKI

# Schools & Churches

The screenshot shows the DB Manager interface with a SQL query executed. The query filters for buildings that are either schools or churches. The results table shows 6 rows of data.

```
1 SELECT * FROM sbg_buildings
2 WHERE type='school' OR type='church';
```

	id	geom	osm_id	code	fclass	name	type
1	20	0106000020797...	63940359	1500	building	Pfarr- und ...	church
2	23	0106000020797...	66232058	1500	building	Kloster Maria ...	church
3	26	0106000020797...	66232062	1500	building	NULL	church
4	244	0106000020797...	112738384	1500	building	Neue ...	school
5	250	0106000020797...	112738399	1500	building	Volksschule ...	school
6	2780	0106000020797...	2811863	1500	building	NULL	school

Load as new layer

Column(s) with unique values:   Geometry column:

Layer name (prefix):

Avoid selecting by feature id

Buttons: Retrieve columns, Set filter, Load, Cancel

# Houses & area

The screenshot shows the DB Manager interface with a query window open. The query is:

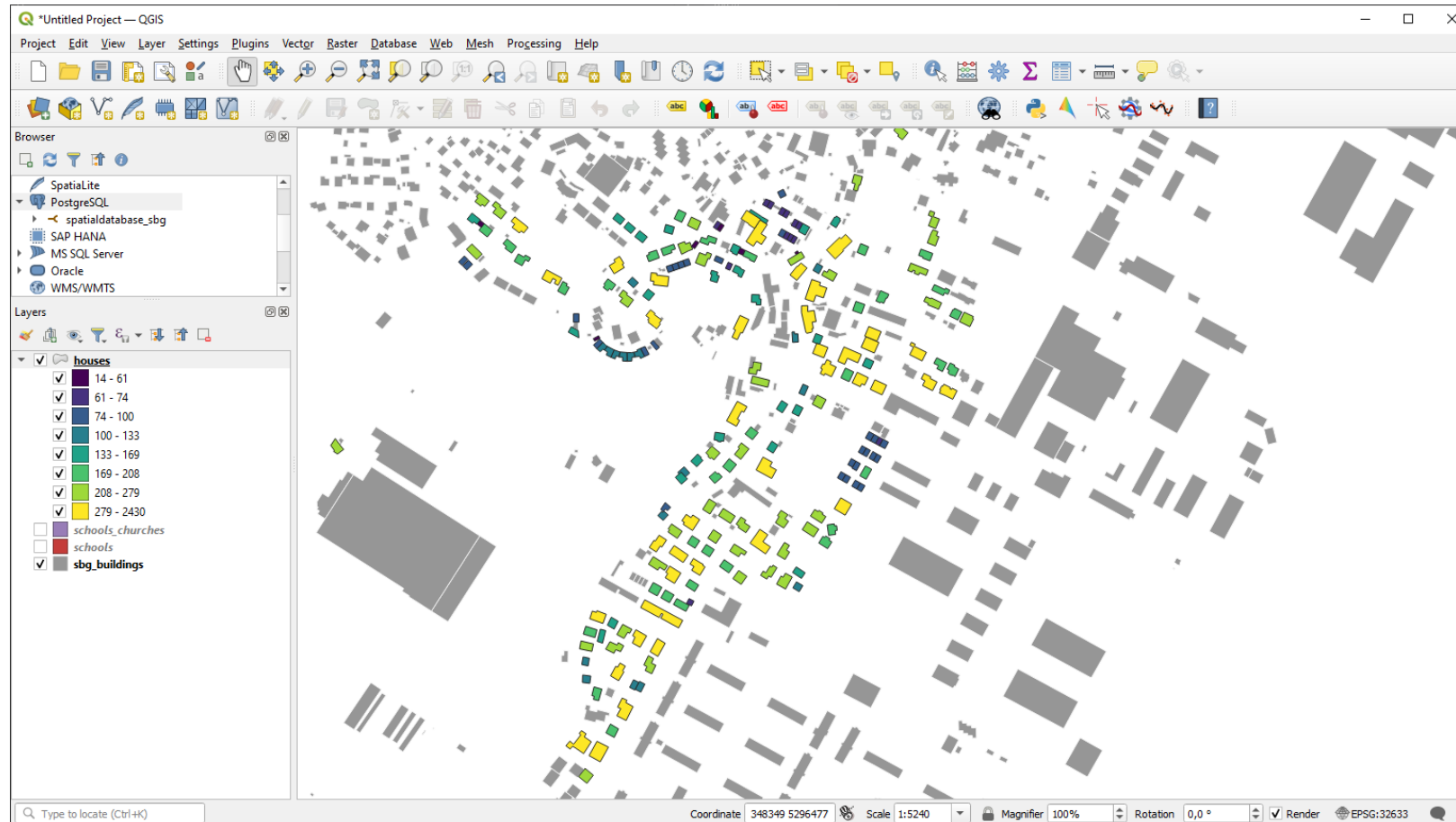
```
1 SELECT *, ST_Area (geom) AS area
2 FROM sbg_buildings
3 WHERE type='house'
```

The results table shows 1110 rows. The first 7 rows are as follows:

	id	geom	osm_id	code	fclass	name	type	area
1	27	0106000020797...	66232063	1500	building	Gasthof Maria ...	house	981.7600076948...
2	47	0106000020797...	68094654	1500	building	NULL	house	49.13625980060...
3	53	0106000020797...	68094660	1500	building	NULL	house	256.0368001466...
4	54	0106000020797...	68094661	1500	building	NULL	house	55.78037650085...
5	56	0106000020797...	68094663	1500	building	NULL	house	58.19001475075...
6	70	0106000020797...	68094912	1500	building	NULL	house	116.0320498834...
7	171	0106000020797...	97244975	1500	building	Zur Plainlinde	house	411.6395860083...

Below the table, there is a checkbox labeled "Load as new layer" which is currently unchecked. A "Cancel" button is visible at the bottom right of the results area.

# Houses & area



# Distance buildings to the castle

The screenshot shows the DB Manager interface with a SQL query window. The query is as follows:

```
1 SELECT
2 b.*,
3 st_distance (b.geom, g.geom) AS distance2castle
4 FROM
5 sbg_buildings AS b,
6 (SELECT geom FROM sbg_buildings WHERE name = 'Glockenturm') AS g;
```

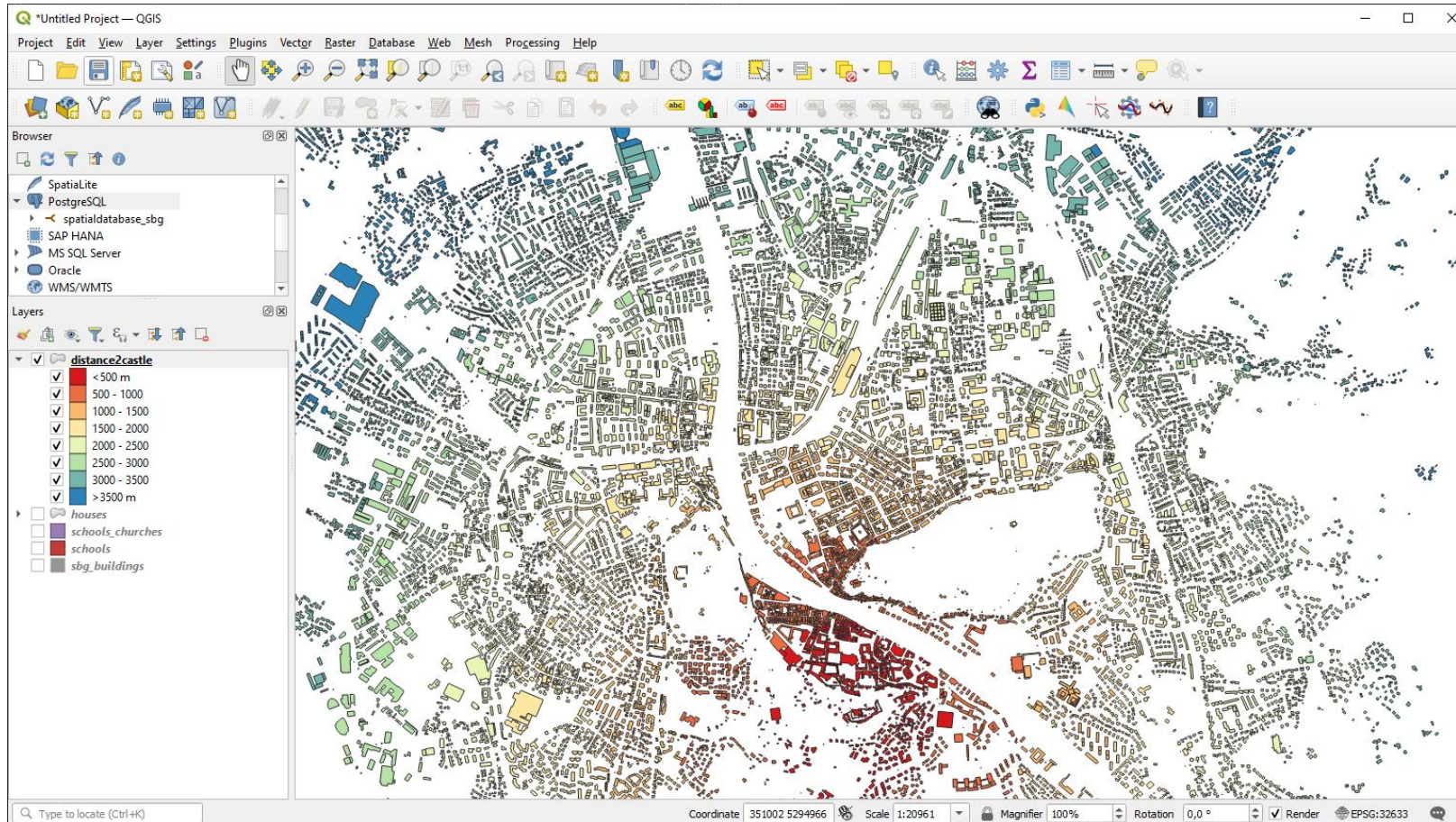
The query has been executed, returning 41002 rows in 4.189 seconds. The results are displayed in a table with the following columns: id, geom, osm\_id, code, fclass, name, type, and distance2castle. The first 9 rows are shown below:

	id	geom	osm_id	code	fclass	name	type	distance2castle
1	1	0106000020797...	24403125	1500	building	NULL	apartments	4242.446044197...
2	2	0106000020797...	24403128	1500	building	Gummittechnik ...	commercial	4262.69323172337
3	3	0106000020797...	24403133	1500	building	NULL	NULL	4147.697501015...
4	4	0106000020797...	24403137	1500	building	NULL	apartments	4266.636983008...
5	5	0106000020797...	24403141	1500	building	NULL	apartments	4351.204246243...
6	6	0106000020797...	24403147	1500	building	NULL	detached	4054.900872444...
7	7	0106000020797...	24403150	1500	building	Tapezierer ...	retail	4009.790554560...
8	8	0106000020797...	62643707	1500	building	NULL	NULL	5206.076898211...
9	9	0106000020797...	62643708	1500	building	Spar	NULL	5339.987552865...

Below the table, there are options to load the results as a new layer. The 'Load as new layer' checkbox is checked. The 'Column(s) with unique values' is set to 'id' and the 'Geometry column' is set to 'geom'. The layer name is 'distance2castle'. The 'Load' button is visible.



# Distance buildings to the castle



# Buildings within 500m of castle

The screenshot shows the DB Manager interface with a SQL query executed. The query is:

```
1 SELECT b.* FROM sbg_buildings AS b,  
2 (SELECT geom FROM sbg_buildings WHERE name = 'Glockenturm') AS g  
3 WHERE st_distance(b.geom,g.geom) < 500;  
4
```

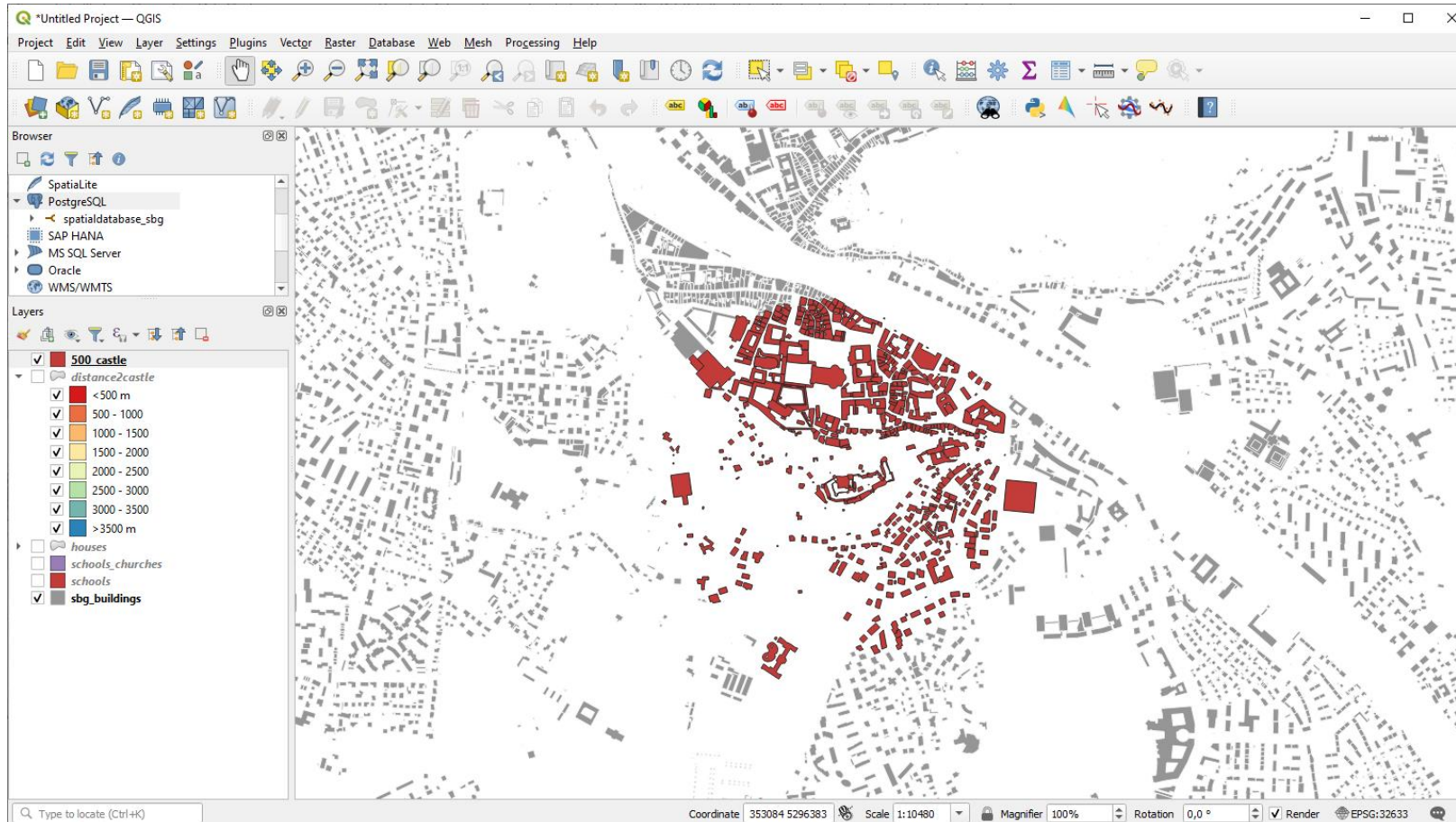
The results are displayed in a table with 9 rows. The columns are: id, geom, osm\_id, code, fclass, name, and type.

	id	geom	osm_id	code	fclass	name	type
1	14373	0106000020797...	47011806	1500	building	Krautwächterhä...	NULL
2	14668	0106000020797...	64946839	1500	building	NULL	NULL
3	14621	0106000020797...	58379411	1500	building	Hasengrabenze...	NULL
4	14622	0106000020797...	58379418	1500	building	NULL	NULL
5	14623	0106000020797...	58379419	1500	building	Sperrbogen ...	NULL
6	14624	0106000020797...	58379423	1500	building	Bianchi-Villa	house
7	14625	0106000020797...	58379424	1500	building	Altes Zeughaus	NULL
8	14626	0106000020797...	58379428	1500	building	NULL	NULL
9	14627	0106000020797...	58379429	1500	building	Großes Zeughaus	NULL

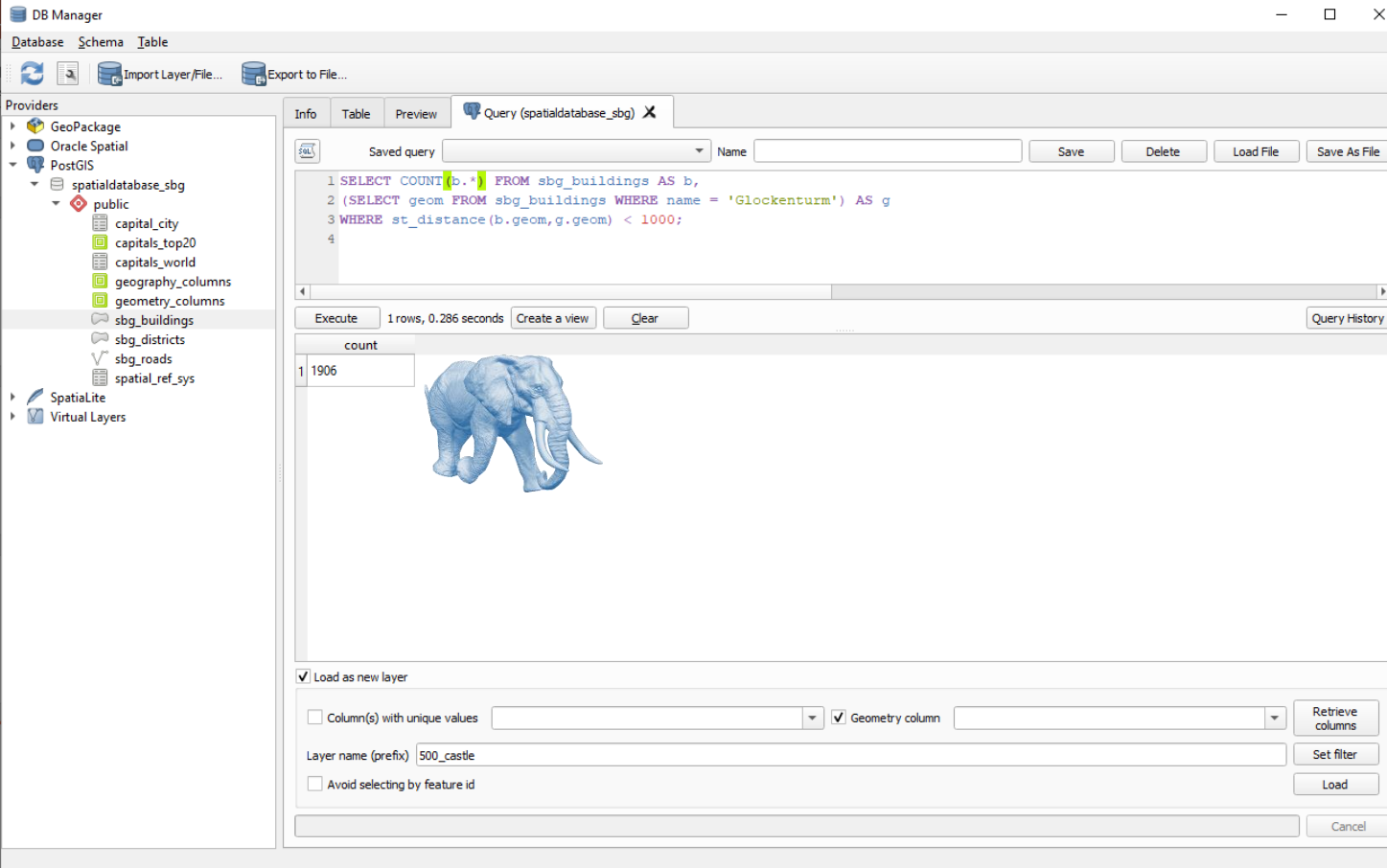
Below the table, there are options to load the results as a new layer. The "Load as new layer" checkbox is checked. The "Column(s) with unique values" is set to "id" and the "Geometry column" is set to "geom". The "Layer name (prefix)" is "500\_castle".



# Buildings within 500m of castle



# Number of buildings within 1km of castle



The screenshot shows the DB Manager interface with a query window open. The query is as follows:

```
1 SELECT COUNT(b.*) FROM sbg_buildings AS b,  
2 (SELECT geom FROM sbg_buildings WHERE name = 'Glockenturm') AS g  
3 WHERE st_distance(b.geom,g.geom) < 1000;  
4
```

The results pane shows a single row with the value 1906 under the column 'count'. A blue elephant icon is overlaid on the results table.

Below the results, the 'Load as new layer' section is visible with the following options:

- Load as new layer
- Column(s) with unique values
- Geometry column
- Layer name (prefix): 500\_castle
- Avoid selecting by feature id

Buttons for 'Retrieve columns', 'Set filter', 'Load', and 'Cancel' are also present.

# average size of all buildings, which are within 1km distance to the fortress

The screenshot shows the DB Manager interface with a query window open. The query is as follows:

```
1 SELECT AVG( ST_area(b.geom) ) AS avg_area FROM sbg_buildings AS b,  
2 (SELECT geom FROM sbg_buildings WHERE name = 'Glockenturm') AS g  
3 WHERE st_distance(b.geom,g.geom) < 1000;  
4
```

The query has been executed, resulting in 1 row and 0.505 seconds. The results are displayed in a table:

avg_area
1 298.6711628658...

Below the results, there are options to load the data as a new layer. The "Load as new layer" checkbox is checked. The "Layer name (prefix)" is set to "500\_castle".

# Explanation

- We use the previous selection of the buildings within 1km to select the buildings
- We add an area column for the buildings `ST_area(geom)`
- We wrap it all in an `AVG()` function

# E10 User Access & Security

Mohamed Dhia TURKI

# 4.3 Creating and configuring users

```
dynamic_salzburg_turkim/postgres_edu@dynamic_salzburg
Query Editor  Query History
1 CREATE ROLE carto_tur
2 LOGIN PASSWORD 'map';
3
4 CREATE ROLE intern_tur
5 LOGIN PASSWORD 'intern!';

Data Output  Explain  Messages  Notifications
ERROR: role "carto_tur" already exists
SQL state: 42710
```

The screenshot shows the DB Manager interface with the 'sbg\_buildings' table selected. The left pane shows a tree view of providers, including PostGIS, carto, and public. The right pane shows the 'Info' tab for the 'sbg\_buildings' table.

**General info**

Relation type:	Table
Owner:	postgres_edu
Pages:	4675
Rows (estimation):	41002
Privileges:	select

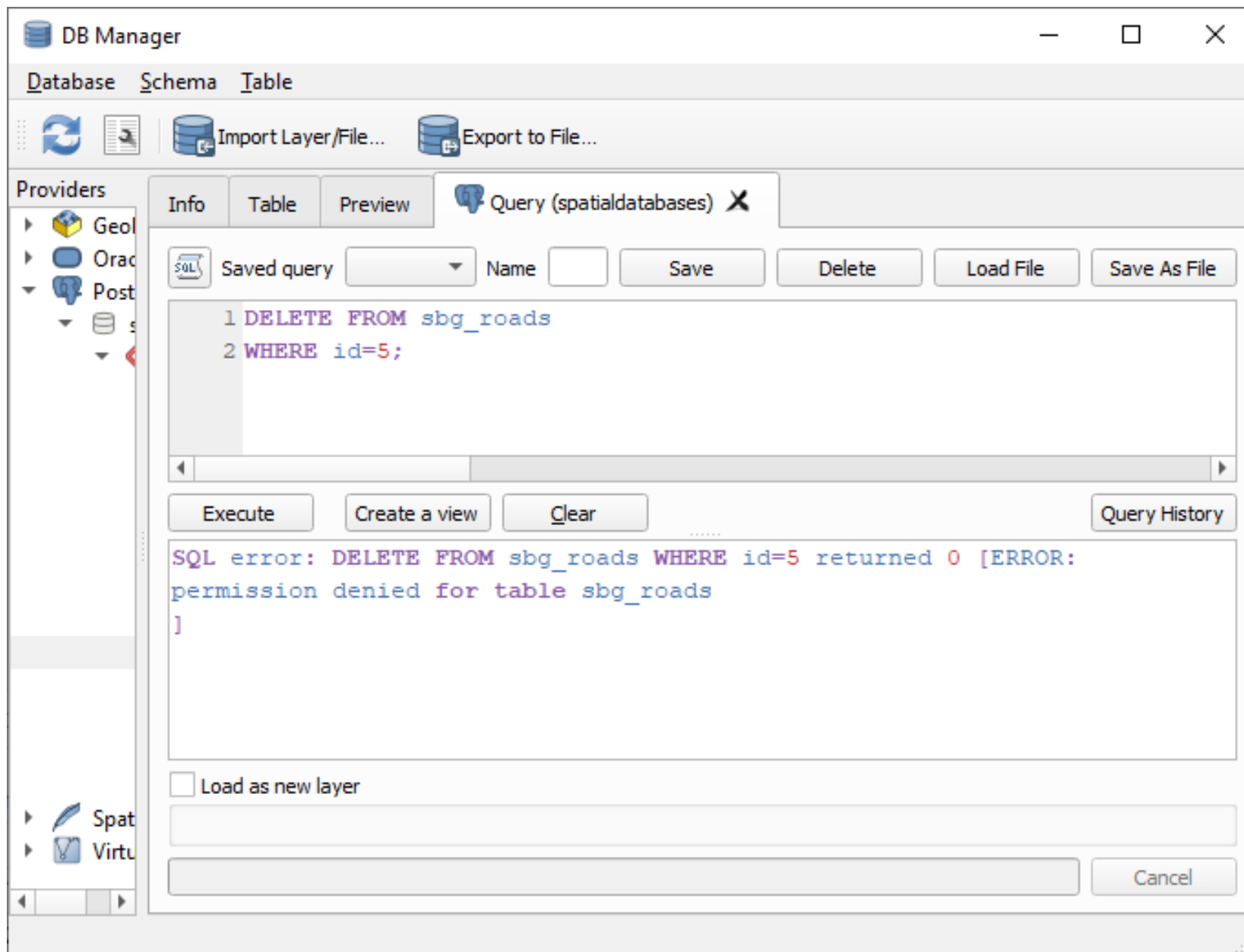
**PostGIS**

Column:	geom
Geometry:	MULTIPOLYGON
Dimension:	2
Spatial ref:	WGS 84 / UTM zone 33N (32633)
Estimated extent:	344880.25000, 5289227.00000, 360574.96875, 5303193.00000
Extent:	(unknown) <a href="#">find out</a>

**Fields**

Warnings: This user has read-only privileges. No spatial index defined [create it](#)





# TASK

- How to void passwords being known?
  - Asking the user to create a new password on first login
  - Using a sort of encryption to secure the passwords

# 4.6 Column-level security

The screenshot displays the QGIS DB Manager interface. On the left, the Query Editor shows the following SQL commands:

```
1 REVOKE ALL ON sbg_buildings FROM intern_tur;  
2 GRANT SELECT (id, name, geom, code, fclass, type) ON sbg_buildings TO intern_tur;  
3
```

The Messages panel at the bottom left shows the execution result: "GRANT Query returned successfully in 50 msec."

The main interface shows the "Providers" tree on the left, with "PostGIS" expanded to "intern" and "public". The "sbg\_buildings" table is selected. The right pane shows the "Info" tab for "sbg\_buildings":

- General info**
  - Relation type: Table
  - Owner: postgres\_edu
  - Pages: 4675
  - Rows (estimation): 41002
  - Privileges: This user has no privileges!
- PostGIS**
  - Column: geom
  - Geometry: geometry
  - Spatial ref: Undefined (-1)
  - Estimated extent: 344880.25000, 5289227.00000, 360574.96875, 5303193.00000
  - Extent: (unknown) [\(find out\)](#)
  - There is no entry in geometry\_columns!
  - No spatial index defined [\(create it\)](#)
- Fields**

DB Manager

Database Schema Table

Import Layer/File... Export to File...

Providers

- GeoPackage
- Oracle Spatial
- PostGIS
  - carto
  - intern
    - public
      - capital\_city
      - capitals\_top20
      - capitals\_world
      - geography\_columns
      - geometry\_columns
      - intern\_view
      - raster\_columns
      - raster\_overviews
      - salzburg\_dem
      - sbg\_buildings
      - sbg\_districts
      - sbg\_roads
      - security\_geofence
      - spatial\_ref\_sys
- SpatialLite
- Virtual Layers

Info Table Preview Query (intern) X

Saved query Name Save Delete Load File Save As File

```
1 SELECT id, name, geom, code, fclass, type FROM sbg_buildings;
```

Execute 41002 rows, 3.503 seconds Create a view Clear Query History

	id	name	geom	code	fclass	type
1	2903	NULL	0106000020797...	1500	building	NULL
2	6103	NULL	0106000020797...	1500	building	NULL
3	8741	NULL	0106000020797...	1500	building	NULL
4	12827	NULL	0106000020797...	1500	building	NULL
5	13781	NULL	0106000020797...	1500	building	NULL

Load as new layer

Cancel

# 4.7 Row-level security

The screenshot displays the QGIS interface with a PostgreSQL connection to 'dynamic\_salzburg\_turkim/postgres\_edu@dynamic\_salzburg'. The Query Editor shows the following SQL commands:

```
1 -- create policy
2 CREATE POLICY intern_shield ON sbg_buildings FOR ALL TO intern_tur
3 USING (security_level='public');
4 -- enable row-level security on table (only once)
5 ALTER TABLE sbg_buildings ENABLE ROW LEVEL SECURITY;
6
```

The Messages panel at the bottom left indicates: "ALTER TABLE" and "Query returned successfully in 67 msec."

The main map area shows a dense vector layer of buildings. A 'Row security' dialog box is open, showing a table with 11 rows. The first row is highlighted with a green border.

_uid_	id	name	code	fclass	type
1	34877	Perron	1500	building	NULL
2	34879	Studierendenhe...	1500	building	residential
3	34883	NULL	1500	building	NULL
4	34885	NULL	1500	building	NULL
5	34889	Blaue Gans	1500	building	hotel
6	34891	NULL	1500	building	NULL
7	34895	NULL	1500	building	shed
8	34897	NULL	1500	building	NULL
9	34901	SalzburgMuseu...	1500	building	public
10	34903	NULL	1500	building	NULL
11	34907	NULL	1500	building	NULL

# 4.7 Row-level security (with intersection)

dynamic\_salzburg\_turkim/postgres\_edu@dynamic\_salzburg

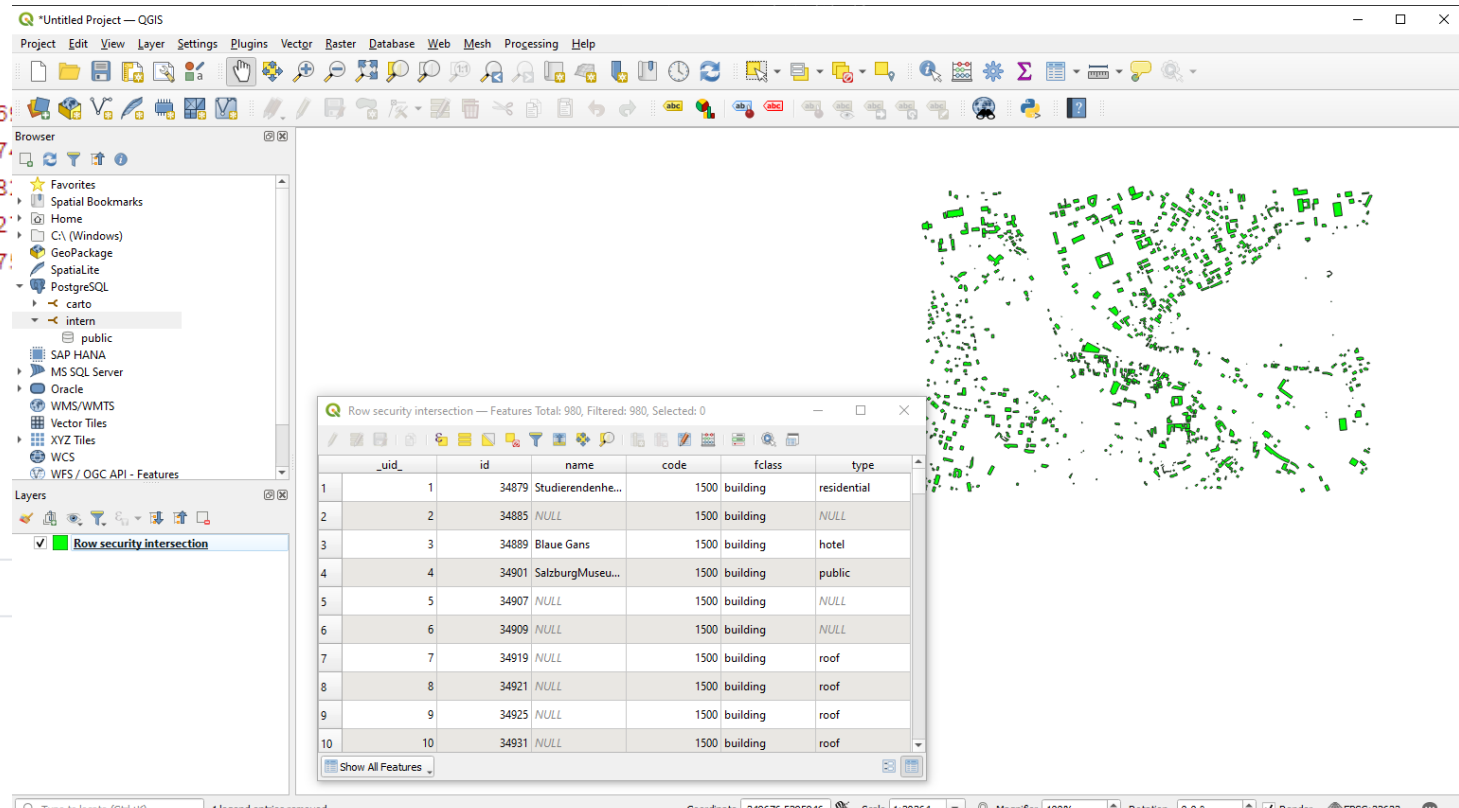
Query Editor Query History

```
1 DROP POLICY intern_shield ON sbg_buildings;
2 CREATE POLICY intern_shield
3 ON
4 sbg_buildings FOR ALL
5 TO
6 intern_tur USING ( st_intersects(
7   geom, ST_GeomFromText('Polygon ((352387.999041571340058 5296
8     354909.84093548177042976 5296931.33667
9     354909.84093548177042976 5295245.06758
10    352387.999041571340058 5295245.0675822
11    352387.999041571340058 5296931.3366747
12 AND security_level='public'));
13
```

Data Output Explain Messages Notifications

CREATE POLICY

Query returned successfully in 66 msec.



The screenshot shows the QGIS interface. The top panel is the Query Editor, displaying a SQL query to drop and create a row-level security policy named 'intern\_shield' on the 'sbg\_buildings' table. The query uses the 'st\_intersects' function to filter buildings based on a specific polygon geometry and a security level of 'public'. The middle panel is the Browser, showing a tree view of the database structure with the 'intern' schema selected. The bottom panel is the Table View, displaying the results of the query. The table has 10 rows and 6 columns: \_uid\_, id, name, code, fclass, and type. The results show various building features, including residential, hotel, public, and roof types.

_uid_	id	name	code	fclass	type
1	34879	Studierendenhe...	1500	building	residential
2	34885	NULL	1500	building	NULL
3	34889	Blaue Gans	1500	building	hotel
4	34901	SalzburgMuseu...	1500	building	public
5	34907	NULL	1500	building	NULL
6	34909	NULL	1500	building	NULL
7	34919	NULL	1500	building	roof
8	34921	NULL	1500	building	roof
9	34925	NULL	1500	building	roof
10	34931	NULL	1500	building	roof



# TASK

- Security risks
  - The password can be leaked somehow. Having the database only accessible through a local network or a VPN is more secure.
  - The data can be lost or damaged so backups should be made regularly. Also, the database server should be secured against physical damage.
  - Hacking and SQL injections. Use prepared statements.