GIS IN THE ROCKIES

Premier Geospatial Conference of the Rocky Mountain West 2017

September 20-21, 2017

Hilton Denver Inverness, Denver, Colorado

SQL Spatial with QGIS Visualization

Matthew Baker | Denver Public Schools

Dave Murray | City of Westminster

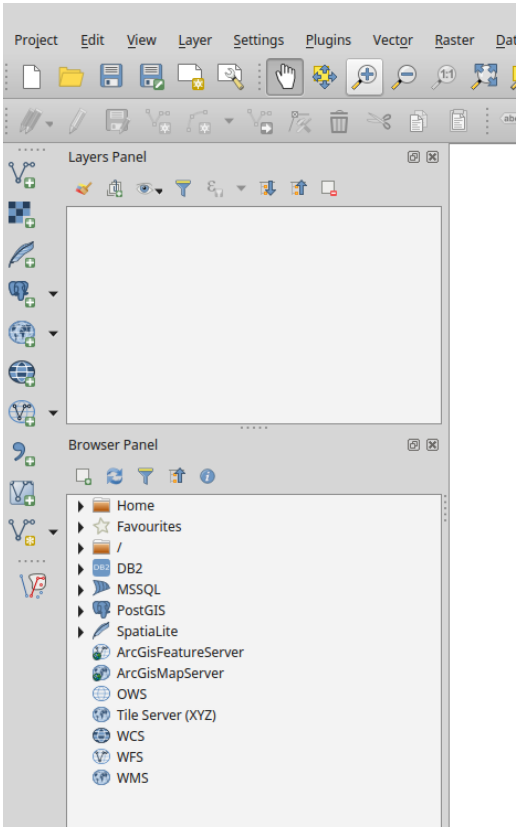
Table of Contents

Working with QGIS

Connect to PostGIS Database in QGIS

First launch QGIS Desktop 2.18 (not the QGIS Browser).

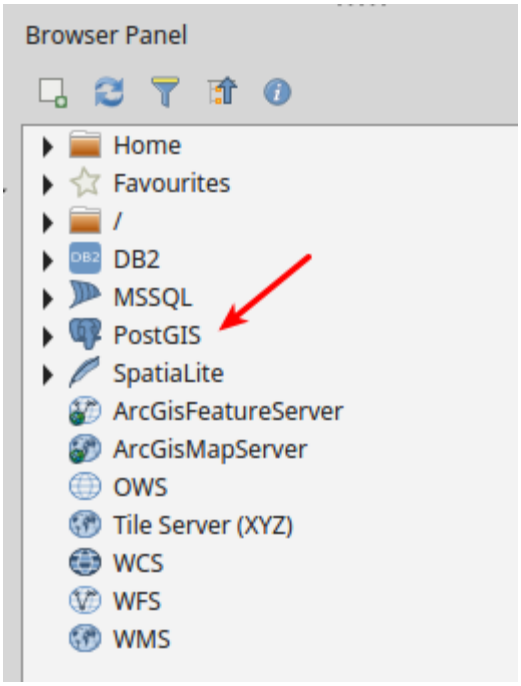
If not already visible, Enable the Browser Window from the menu View > Panels > Browser Panel



Dock the Browser panel below the Layers panel on the left-hand side of the QGIS window:

Note: The panels can be moved by dragging the tops of the panels, moving them to the desired location, and docking them at either side of the screen or within another panel.

In the Browser panel, find the PostGIS node, right-click, and choose ***New Connection***.

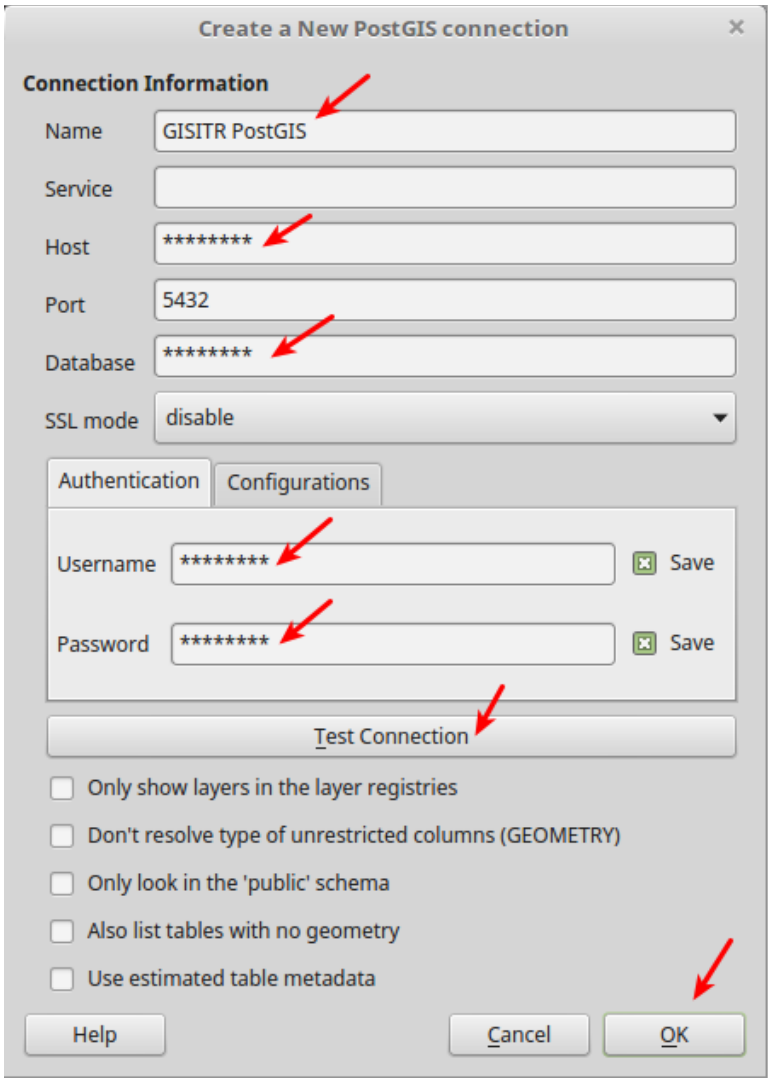


Enter the following options:

Name: GISITR PostGIS
Host: *(ask instructor)*
Database: *(ask instructor)*
Username: *(ask instructor)*
Password: *(ask instructor)*

Click the ***Test Connection*** button to make sure you've entered your credentials correctly.

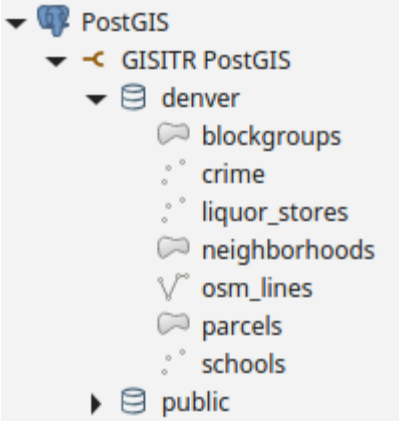
Click OK to add the PostGIS connection.



Explore PostGIS Data in QGIS

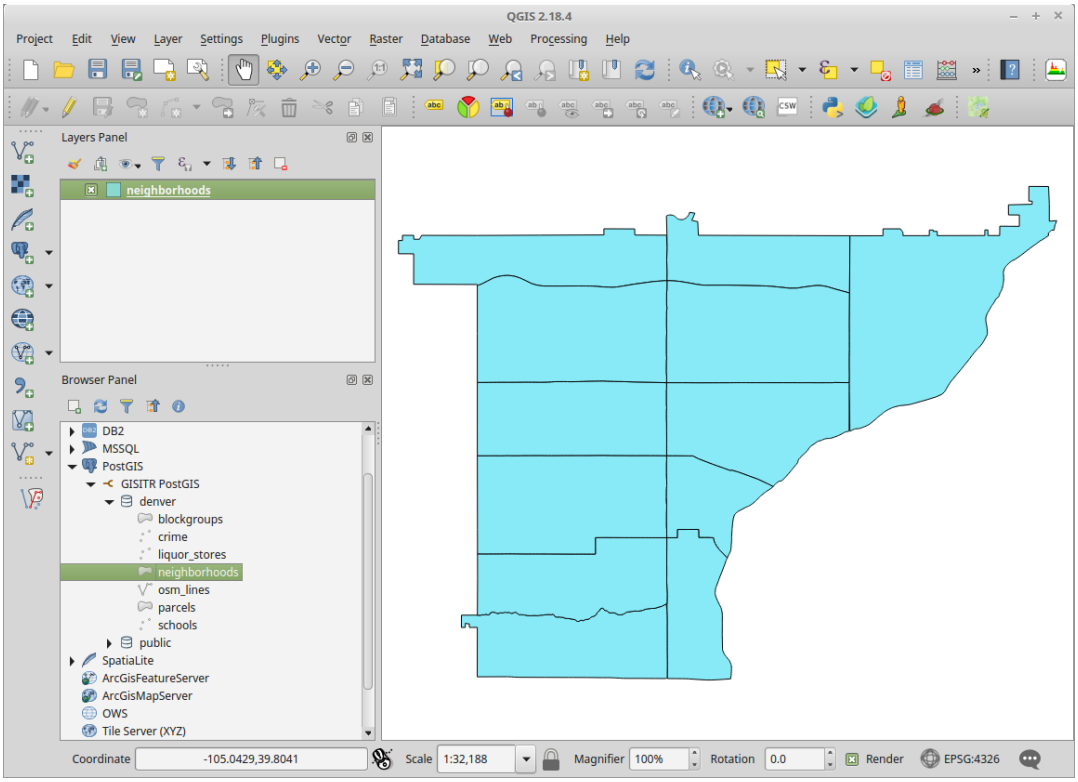
Expand the PostGIS node in the Browser Window to examine your PostGIS connection.

Expand the ***denver*** schema to display the spatial tables in the PostGIS database:

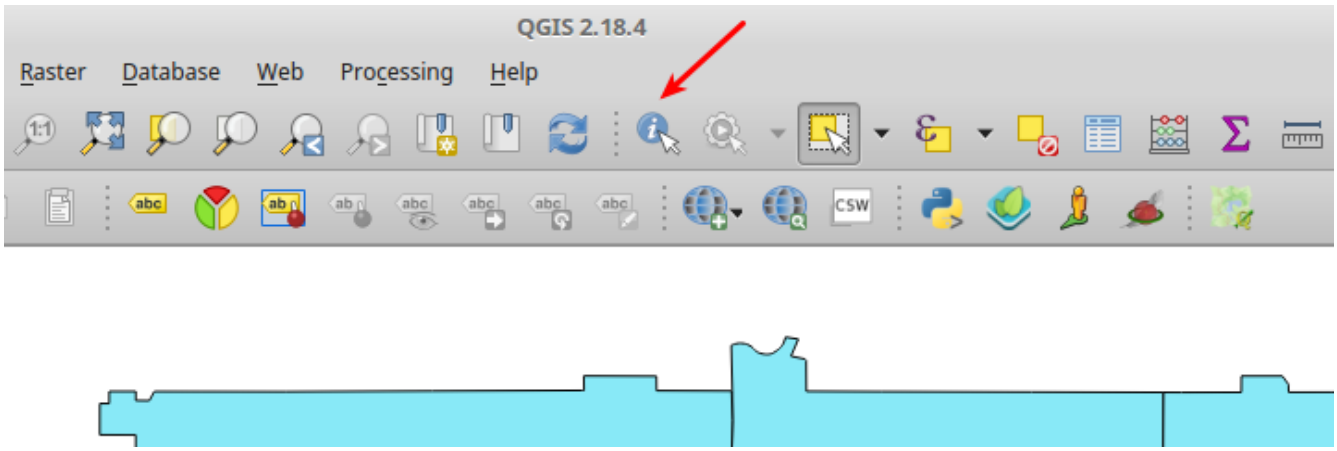


Double-click the neighborhoods layer to add it to the map canvas:

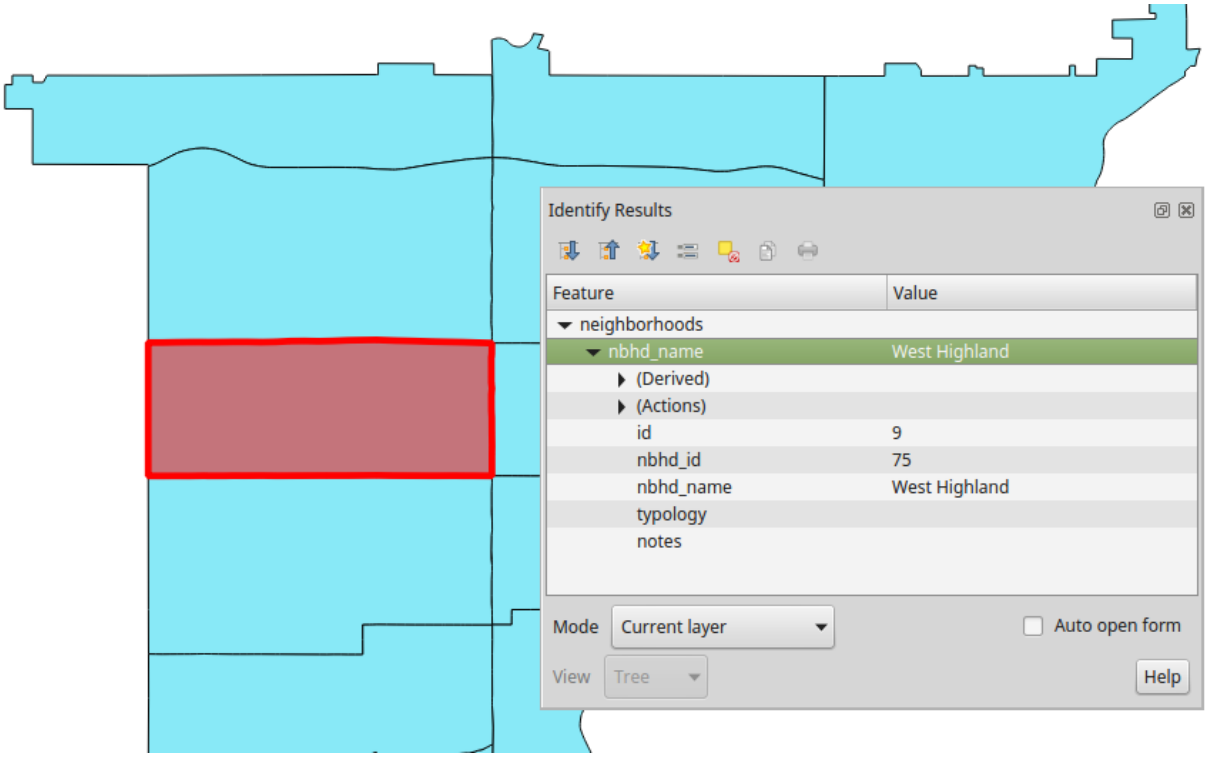
Note the neighborhoods layer appears in the Layers panel:



Use the Identify tool to examine the schema (table design) of the neighborhoods layer:



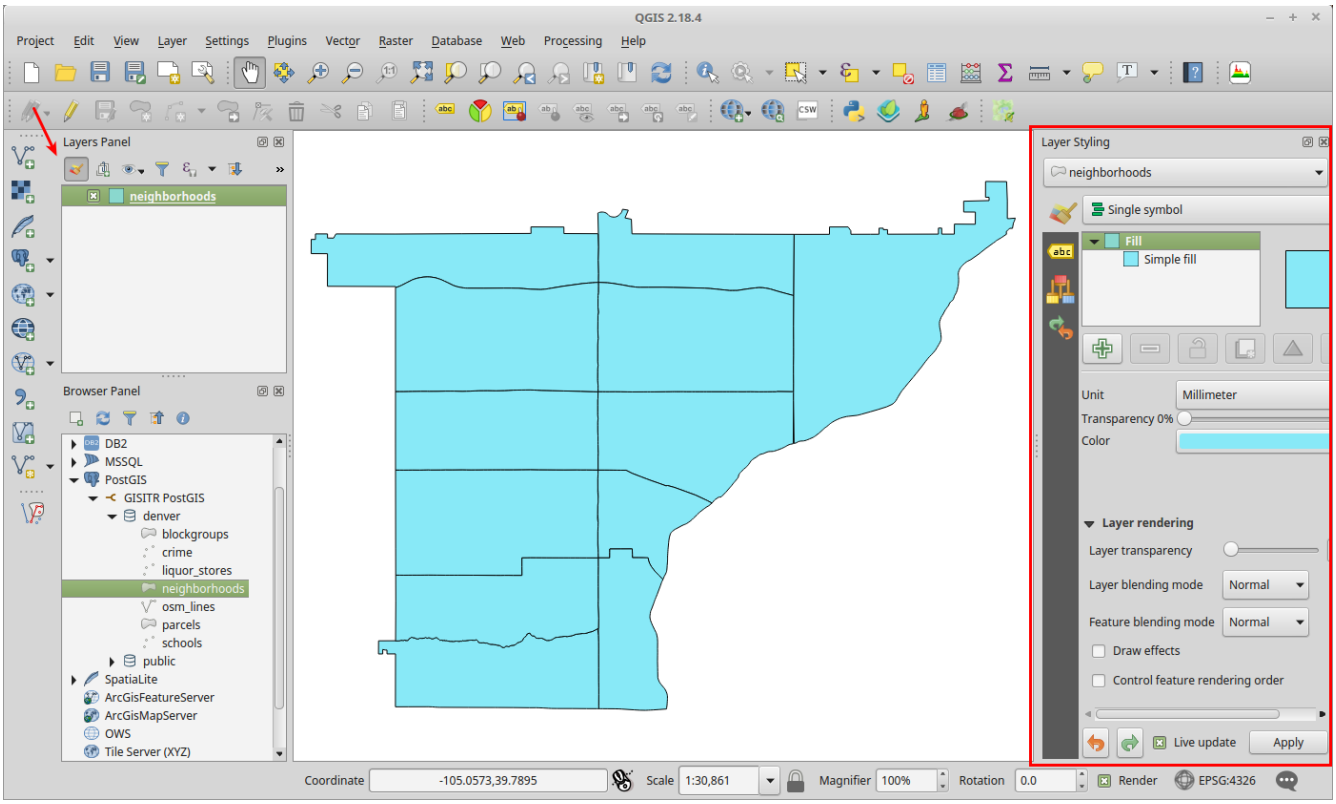
Click a feature in the neighborhoods layer to reveal the columns and values of the feature:



Note the ***nbhd_name*** column, which stores the name of the Denver Neighborhoods.

Style data in QGIS

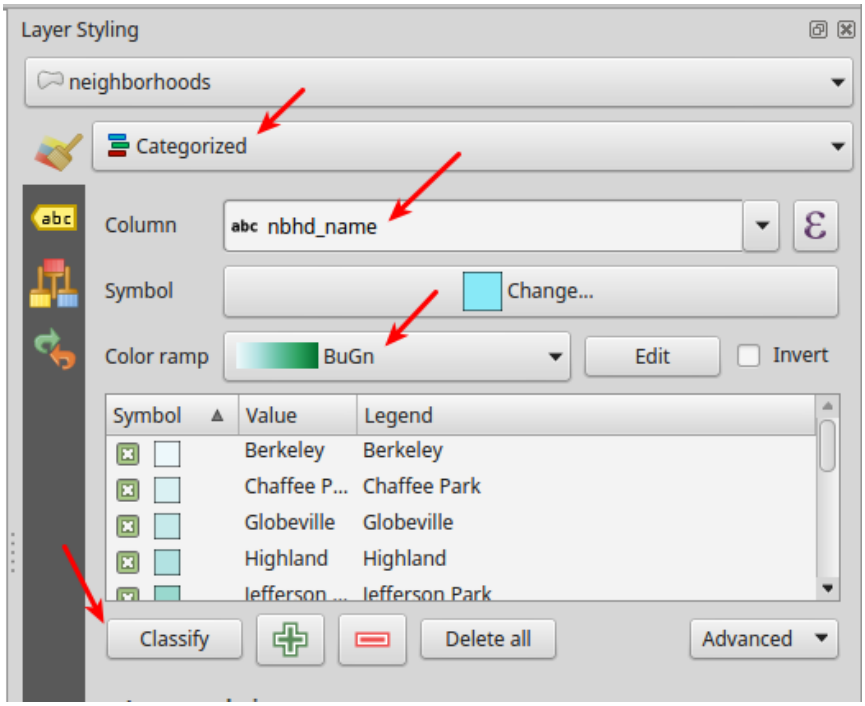
In the top left-hand side of the Layers panel, enable the Styling panel:



The ***Layer Styling panel*** is a dynamic version of the Layer Properties window. Changes made in the Layer Styling panel will be seen immediately on the map. The panel can be docked opposite the Layers/Browser panels, and hidden by toggling the same Icon used to launch the panel.

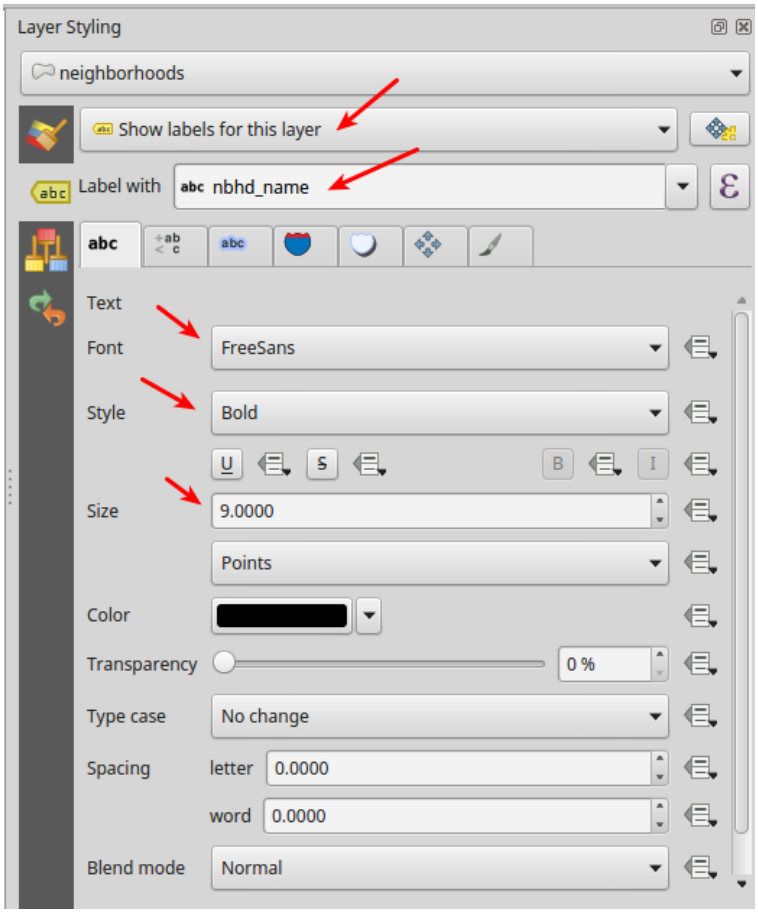
In order to make a basic Denver Neighborhood map, use the **Symbology** node to style the neighborhoods layer:

- Use the Categorized renderer
- Choose the ***nbhd_name*** column
- Assign the ***BuGn*** color ramp
- Click Classify to apply the renderer

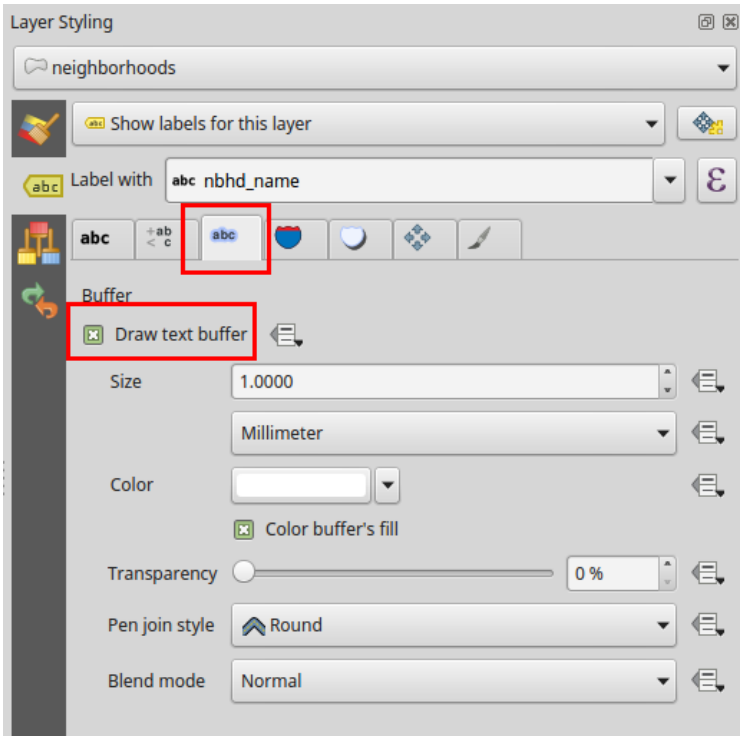


As you set the options, notice the map is updated with each new setting.

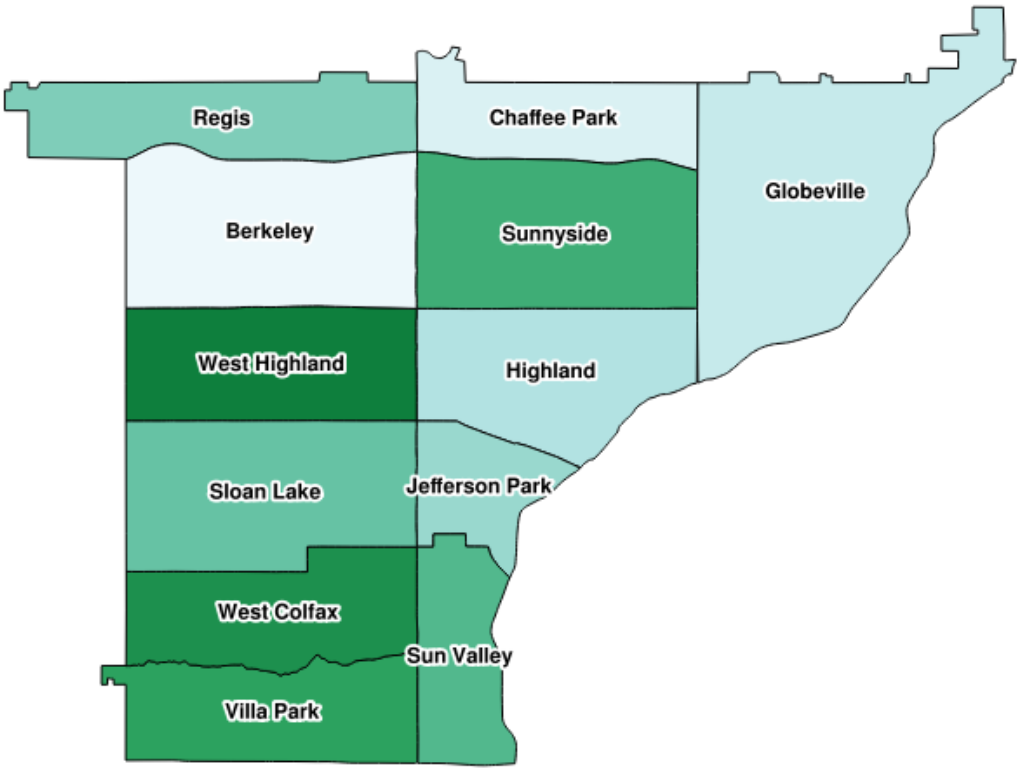
Next, toggle the **Labels** node and label the neighborhoods by their name:



Add a white halo around the labels by enabling the Buffer tab:

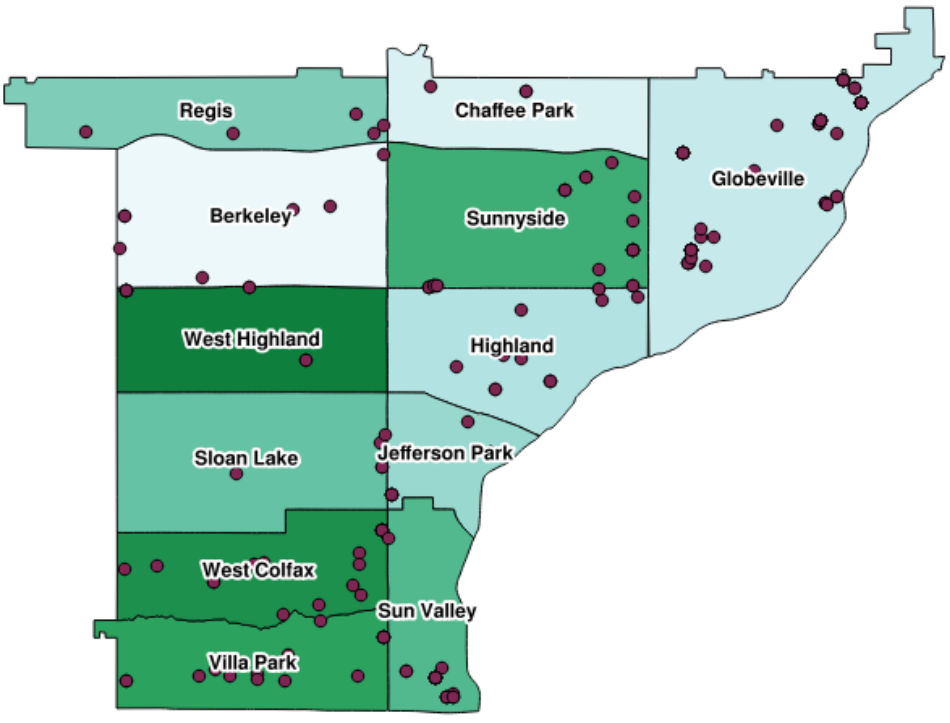


Note the changes on the map:



Create a dynamic heat map layer

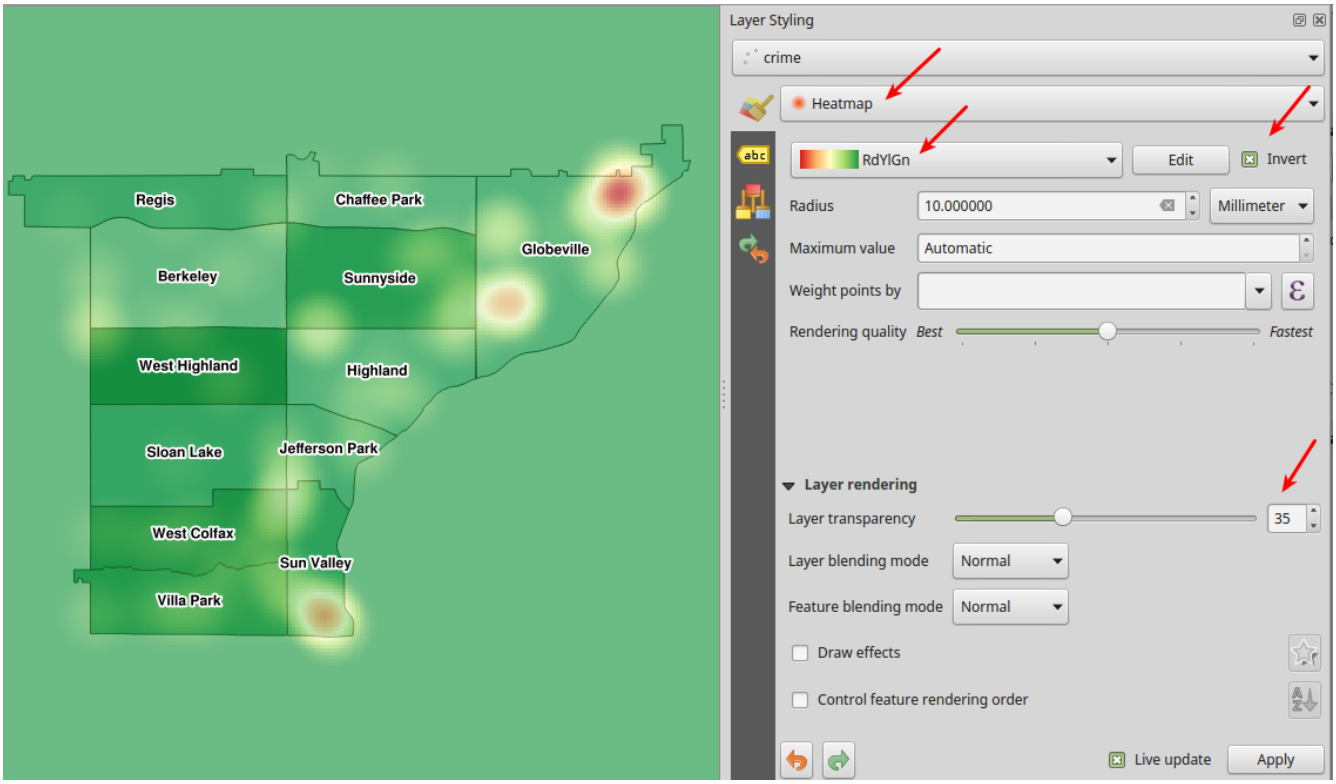
From the Browser window, add the ***denver.crime*** layer:



Use the Layer Styling panel to apply a heat map renderer to the crime layer.

- Choose the Heat Map renderer
- Choose the RdYlGn color ramp
- Invert the color ramp to use red as the highest value
- Set the layer transparency to 35%

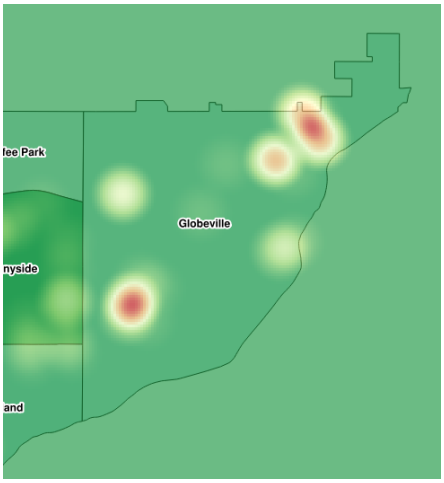
Note the result and layer settings:



Note the hotspots that show the high and low areas of crime in the NW part of Denver.

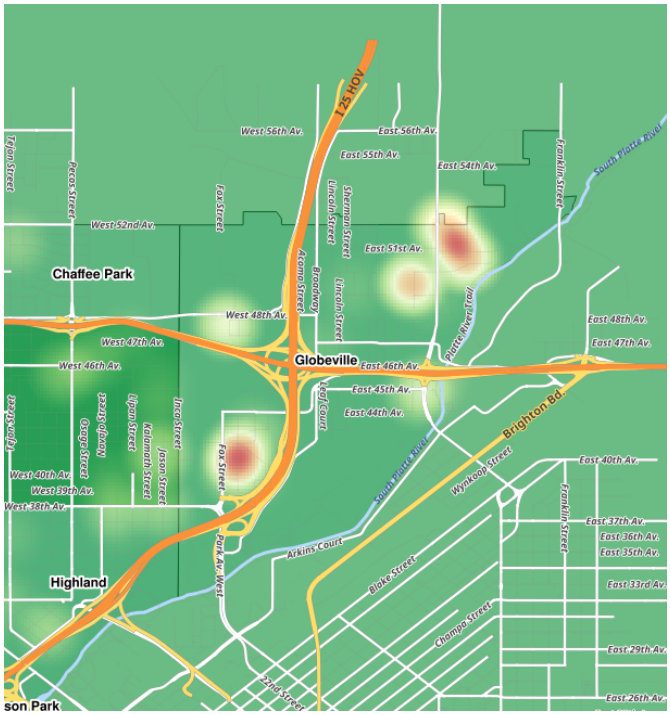
The Heatmap renderer is dynamic, and will re-render with scale (zoom).

Zoom into the Globeville neighborhood to get a refined heatmap:



Add OpenStreetMap Data

From the browser window, add the **denver.osm_lines** layer.



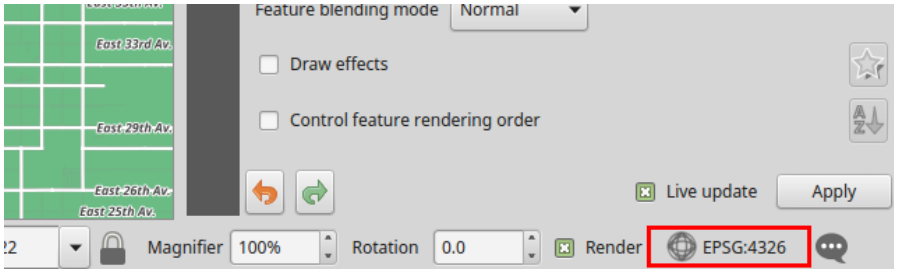
This is a subset of the Open Street Map dataset for the NW Denver region.

The data displays with a style that is saved in the PostGIS database.

***Optional**: Use the Layer Styling panel to examine the rule-based styling of the data. This is a somewhat complex set of rules, but demonstrate the detail to which data can be styled in a QGIS layer.*

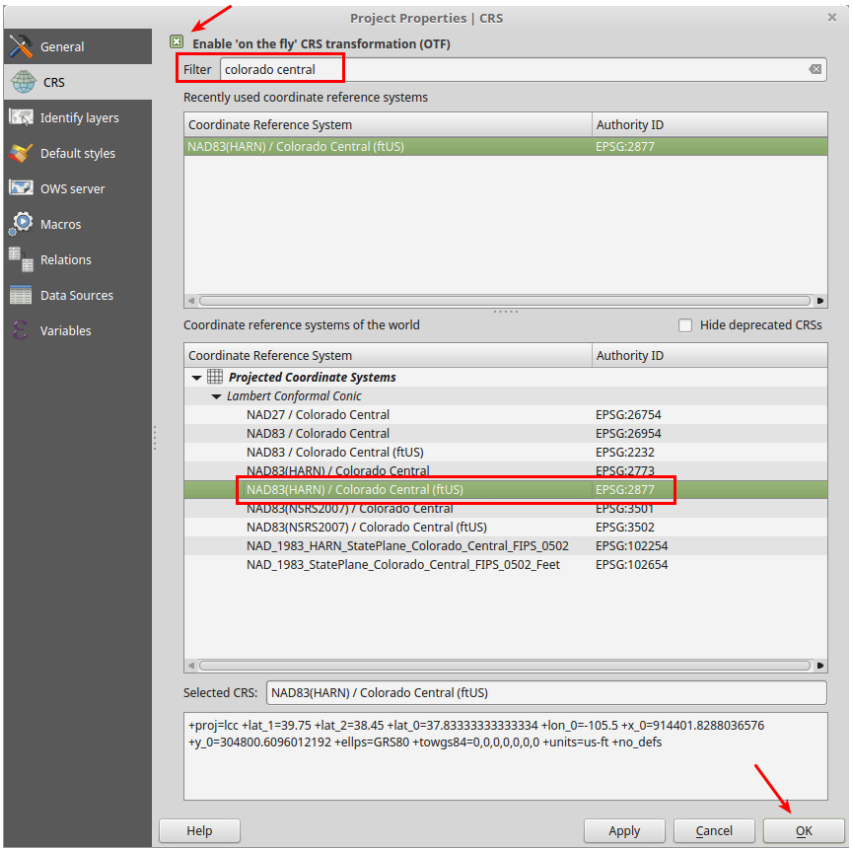
Change the display coordinate system of a QGIS map

Our QGIS map is currently displaying the data in the WGS84 (decimal degree or lat/lon) coordinate system, which we can see in the bottom right-hand corner of the QGIS window:



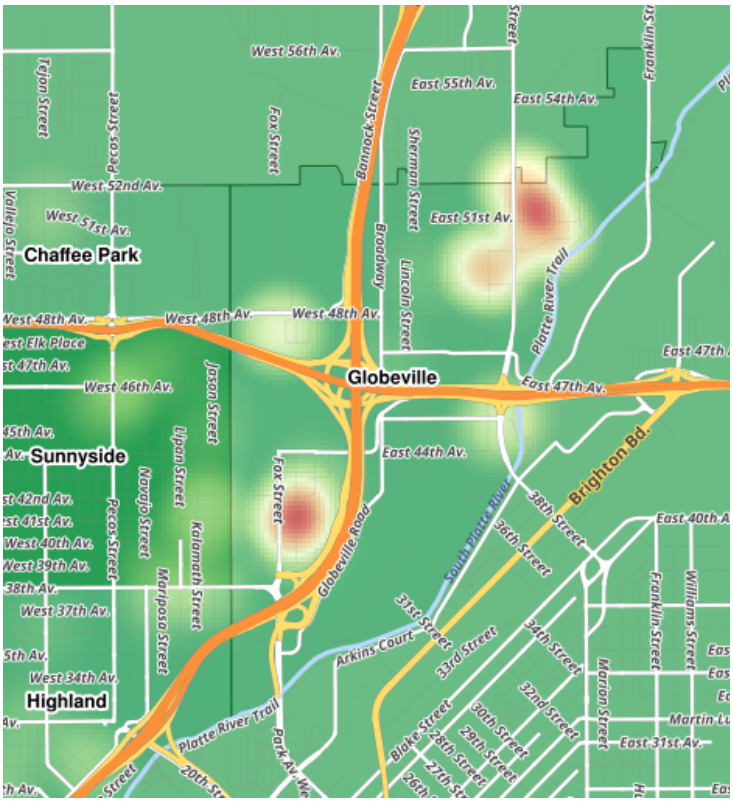
In order to display our data in the most regionally aesthetic coordinate system, click the SRS icon to display the Project Properties > CRS (Coordinate Reference System) window:

- Enable ‘on the fly’ CRS transformation
- filter on **colorado central** and note the options that appear in the bottom list
- Choose NAD83(HARN) Colorado Central (ftUS) **EPSG: 2877** from the list.
- This is the State Plane Central (feet) coordinate system
- Click OK to see the map re-projected to State Plane Colorado Central:



The difference at this scale is minimal, but it is important to understand the difference between the ***data*** *coordinates* and the ***display*** coordinates.

The underlying data in PostGIS is stored in WGS84. The units of this coordinate system are decimal degrees, which aren’t used for determining area, length, etc., and don’t look cartographically aesthetic. Since State Plane Central is a regional coordinate system, it not only looks correct, but stores data in feet – a regionally used system of measurement.



In the next section, we'll use SQL spatial functions to view and transform the coordinate system on-the-fly to determine area, length, etc.

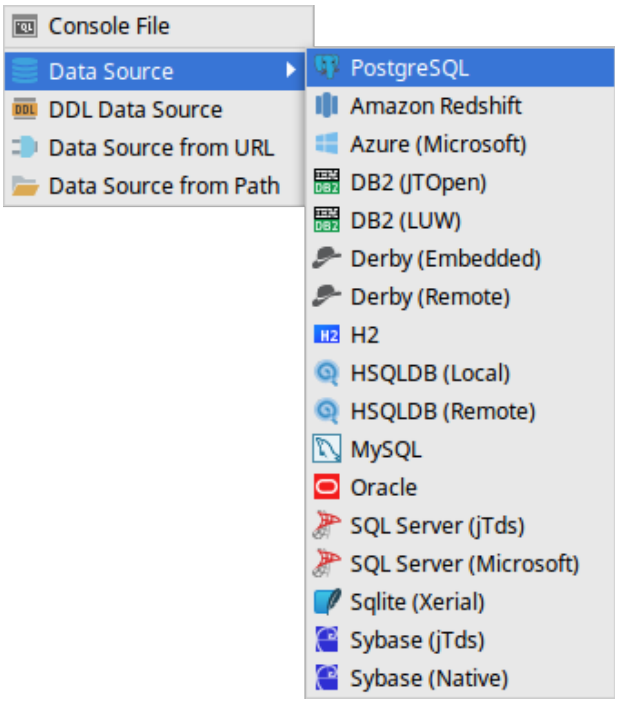
Introduction to SQL Queries

Connect to PostGIS database with DataGrip

Launch DataGrip

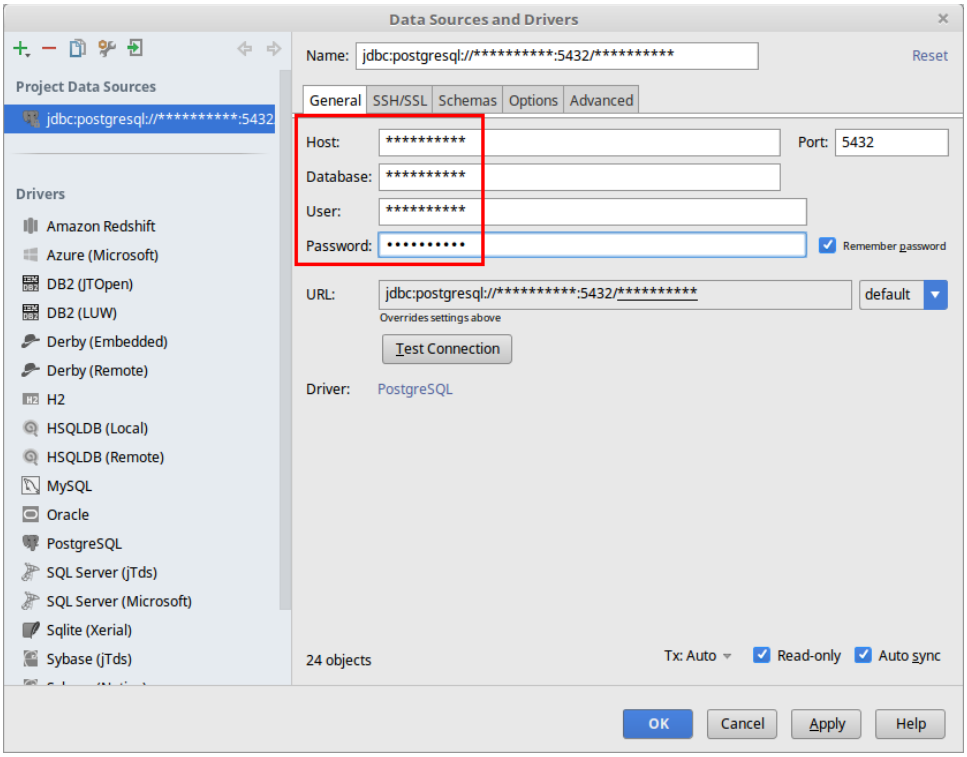
Enable the database window by clicking ***Alt + 1*** or by navigating to ***View > Tool Windows > Database***

Create a new connection to the PostGIS database by selecting a new PostgreSQL Data Source:



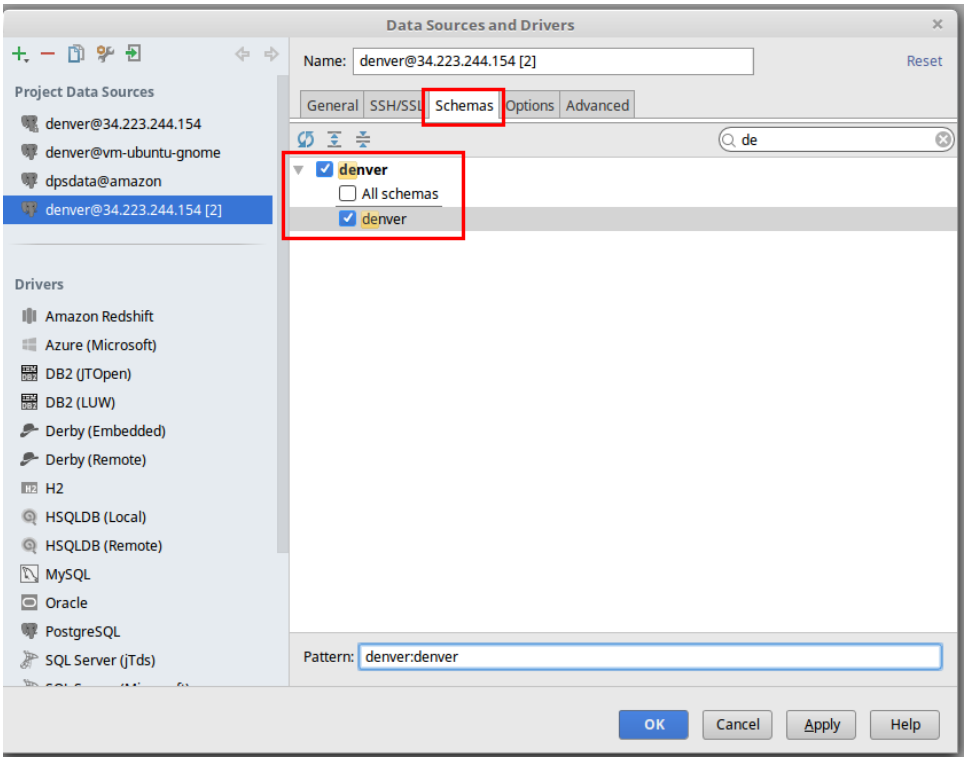
Under the General tab, enter the the same connection parameters you used to connect to the database in QGIS:

Note: *If you get a message in the bottom of the window about 'missing drivers', click the button to enable them.*

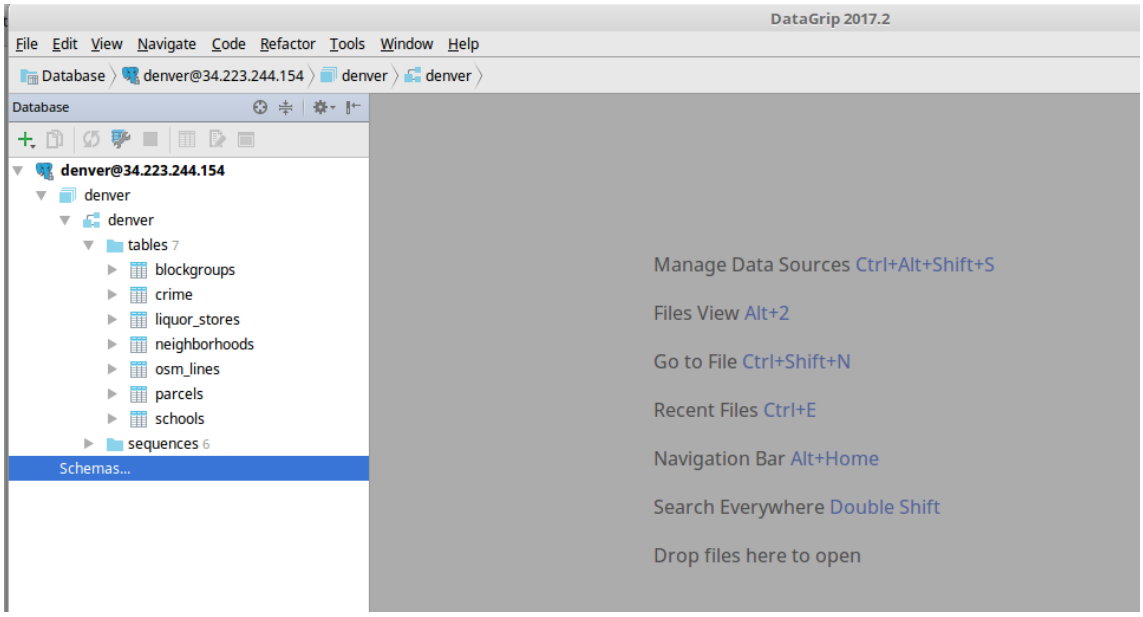


Under the Schema tab, enable the ***denver*** schema:

Click OK to create the connection to the PostGIS database.

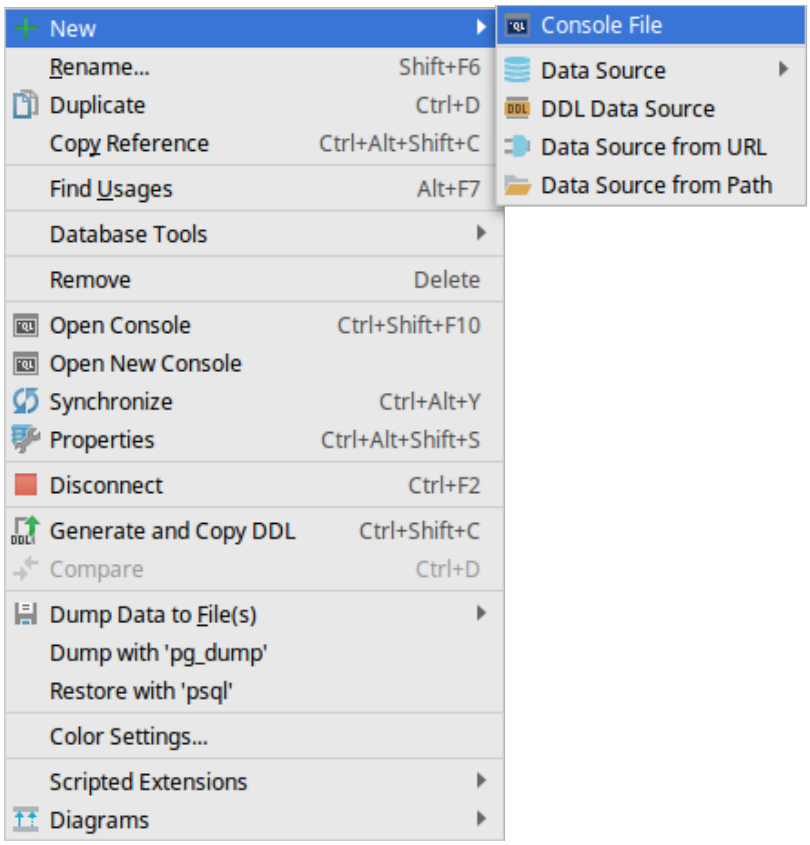


Expand your connection to expose the Database (denver), and schema (denver), and Tables nodes:



Write a SQL query

To begin writing SQL, create a new Console File by right-clicking your data connection and choosing ***New > Console File***:

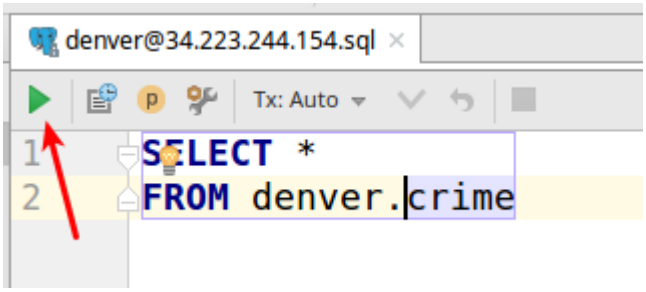


This creates a blank window that you can use to write SQL Queries.

Start by typing the following query - notice the feedback provided in the intellisense that shows the various objects within your database and tables:

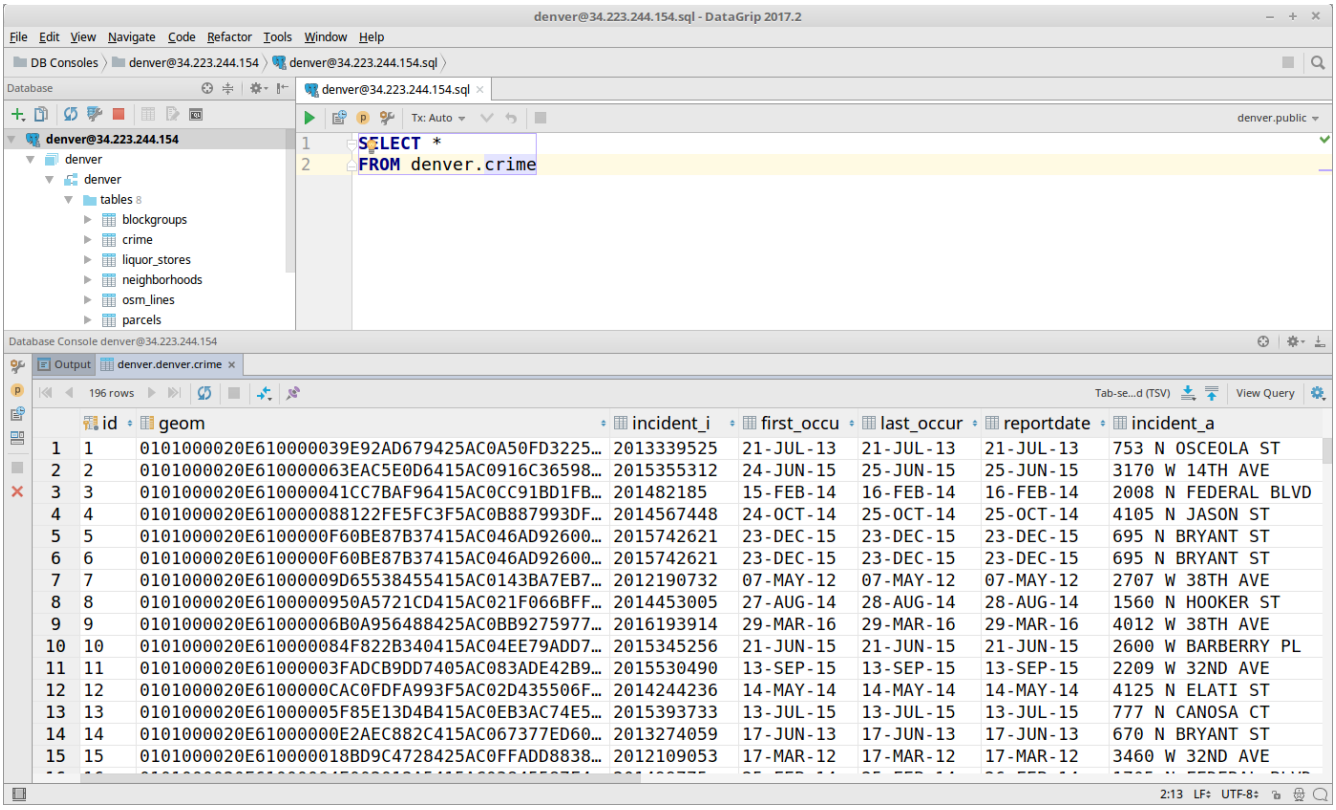
```
SELECT *
FROM denver.crime
```

This selects all fields from the crime table in the denver schema.



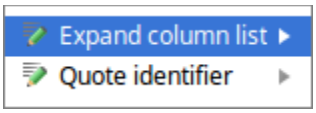
Click Control + Enter to run the query, or choose the Execute button from the DataGrip interface

The results of the query will display in the bottom of the DataGrip window:



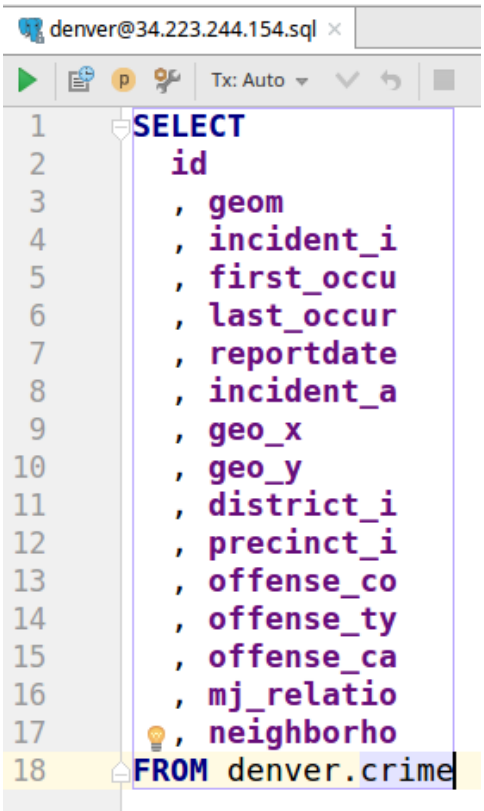
Notice the use of select * in your SQL, which states “select everything”. In order to see all the columns, put your cursor next to the * and click the Light-bulb icon that appears.

Click the option to ***Expand Column List***



This shows an expanded list of all the columns of this table, including the Geometry (geom) column.

This is useful for starting a query, and allows you to remove columns from your SQL that you won’t use.



Use COUNT and GROUP BY operators

Write a SQL query that will show the distinct values of the ***offense_ty*** (offense type) column and list them in alphabetical order:

```
SELECT
  DISTINCT offense_ty
FROM denver.crime
ORDER BY offense_ty ASC
```

Note the results:

Database Console denver@34.223.244.154

Output | denver.denver.crime

25 rows

	offense_ty
1	AGGRAVATED ASSAULT
2	ARSON - BUSINESS
3	ASSAULT - SIMPLE
4	BURGLARY - BUSINESS BY FORCE
5	BURGLARY - BUSINESS NO FORCE
6	BURGLARY - POSS. OF TOOLS
7	BURGLARY - RESIDENCE BY FORCE
8	BURGLARY - RESIDENCE NO FORCE
9	BURGLARY - SAFE
10	CRIMINAL MISCHIEF - OTHER
11	CRIMINAL TRESPASSING
12	DISTURBING THE PEACE
13	DRUG - MARIJUANA POSSESS
14	DRUG - MARIJUANA SELL

In order to count each type of crime, use the COUNT function, group by the offense type, and order by the count from highest to lowest (desc):

```
SELECT
  offense_ty
  , count(*)
FROM denver.crime
GROUP BY offense_ty
ORDER BY count(*) desc
```

Note the results with the added count column:

Database Console denver@34.223.244.154

Output | Result 5

25 rows

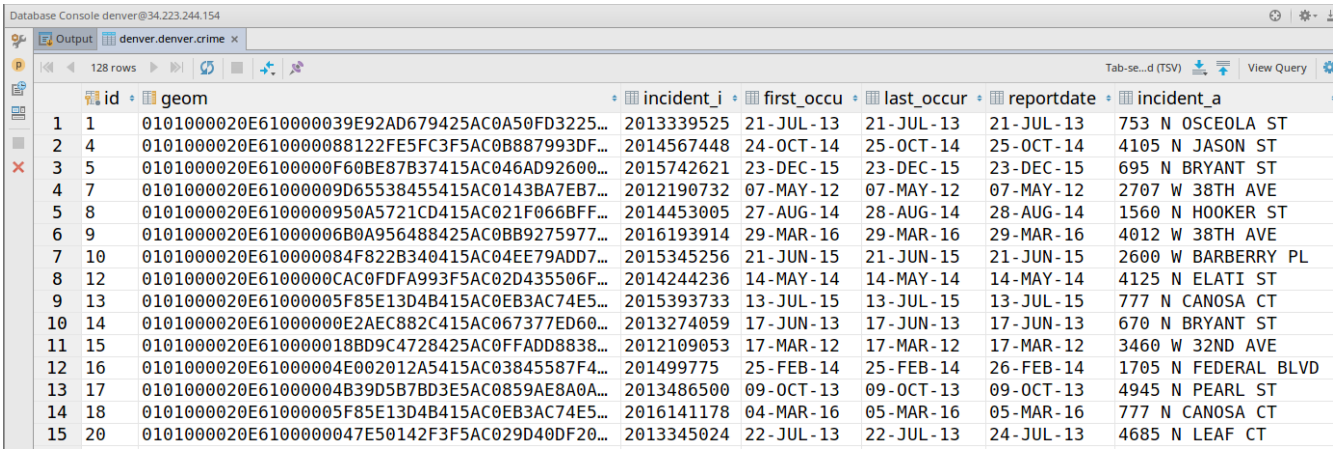
	offense_ty	count
1	BURGLARY - BUSINESS BY FORCE	99
2	CRIMINAL MISCHIEF - OTHER	18
3	BURGLARY - RESIDENCE BY FORCE	14
4	THEFT - OTHER	10
5	BURGLARY - RESIDENCE NO FORCE	9
6	ROBBERY - STREET	6
7	THEFT - ITEMS FROM VEHICLE	4
8	BURGLARY - BUSINESS NO FORCE	4
9	ROBBERY - RESIDENCE	4
10	CRIMINAL TRESPASSING	4
11	DRUG - MARIJUANA SELL	3
12	ROBBERY - BUSINESS	3
13	ASSAULT - SIMPLE	2
14	MENACING - FELONY W/WEAP	2
15	THEFT - SHOPLIFT	2

Use the LIKE operator

Write a query to see the crimes that begin with ‘burglary’ using the LIKE operator, noting the upper-case value used for BURGLARY crimes:

```
SELECT *
FROM denver.crime
WHERE offense_ty LIKE 'BURG%'
```

Note the results:



	id	geom	incident_i	first_occu	last_occur	reportdate	incident_a
1	1	0101000020E610000039E92AD679425AC0A50FD3225…	2013339525	21-JUL-13	21-JUL-13	21-JUL-13	753 N OSCEOLA ST
2	4	0101000020E610000088122FE5FC3F5AC0B887993DF…	2014567448	24-OCT-14	25-OCT-14	25-OCT-14	4105 N JASON ST
3	5	0101000020E6100000F60BE87B37415AC046AD92600…	2015742621	23-DEC-15	23-DEC-15	23-DEC-15	695 N BRYANT ST
4	7	0101000020E61000009D65538455415AC0143BA7EB7…	2012190732	07-MAY-12	07-MAY-12	07-MAY-12	2707 W 38TH AVE
5	8	0101000020E6100000950A5721CD415AC021F066BFF…	2014453005	27-AUG-14	28-AUG-14	28-AUG-14	1560 N HOOKER ST
6	9	0101000020E61000006B0A956488425AC0BB9275977…	2016193914	29-MAR-16	29-MAR-16	29-MAR-16	4012 W 38TH AVE
7	10	0101000020E610000084F822B340415AC04EE79ADD7…	2015345256	21-JUN-15	21-JUN-15	21-JUN-15	2600 W BARBERRY PL
8	12	0101000020E6100000CAC0FDFA993F5AC02D435506F…	2014244236	14-MAY-14	14-MAY-14	14-MAY-14	4125 N ELATI ST
9	13	0101000020E61000005F85E13D4B415AC0EB3AC74E5…	2015393733	13-JUL-15	13-JUL-15	13-JUL-15	777 N CANOSA CT
10	14	0101000020E61000000E2AEC882C415AC067377ED60…	2013274059	17-JUN-13	17-JUN-13	17-JUN-13	670 N BRYANT ST
11	15	0101000020E610000018BD9C4728425AC0FFADD8838…	2012109053	17-MAR-12	17-MAR-12	17-MAR-12	3460 W 32ND AVE
12	16	0101000020E61000004E002012A5415AC03845587F4…	201499775	25-FEB-14	25-FEB-14	26-FEB-14	1705 N FEDERAL BLVD
13	17	0101000020E61000004B39D5B7BD3E5AC0859AE8A0A…	2013486500	09-OCT-13	09-OCT-13	09-OCT-13	4945 N PEARL ST
14	18	0101000020E61000005F85E13D4B415AC0EB3AC74E5…	2016141178	04-MAR-16	05-MAR-16	05-MAR-16	777 N CANOSA CT
15	20	0101000020E6100000047E50142F3F5AC029D40DF20…	2013345024	22-JUL-13	22-JUL-13	24-JUL-13	4685 N LEAF CT

SQL Spatial Basics

View Geometry data using ST_AsText()

Write a query to select the id and geometry columns of the crime table:

```
select
  id
  , geom
from denver.crime
```

Use the SQL spatial **ST_AsText** function to show the geometry of the liquor stores as text:

```
select
  id
  , ST_AsText(geom)
from denver.crime
```

Run the query and view the result:

Database Console denver@34.223.244.154

Output | Result 1

5 rows

	id	st_astext
1	1	POINT(-105.036637474148 39.7761875423097)
2	2	POINT(-105.040759719134 39.7399912267202)
3	3	POINT(-105.025533229779 39.7767692274244)
4	4	POINT(-105.029658160775 39.76913483067)
5	5	POINT(-105.032772423096 39.7406186460011)

ST_AsText is a SQL function in the same way cast(), count(), avg(), etc. are SQL functions. The difference is the ST (spatial / temporal) functions operate on geometry, while mathematical functions operate on numbers.

The column labeled st_astext is the text output of the geometry of each of the points in the denver.crime table, displayed in the source coordinate system of the data, WGS 84 or SRID 4326. The numbers represent the coordinates of the LAT and LON of each point.

Transform Geometry using ST_Transform()

To see the coordinates of the data in State Plane, use the SQL spatial **ST_Transform(geom, srid)** function to project the geometry column to SRID:2877 as you did in QGIS to display the data in State Plane:

```
select
  id
  , ST_AsText(ST_Transform(geom, 2877))
from denver.crimes
```

Database Console denver@34.223.244.154

Output | Result 2

5 rows

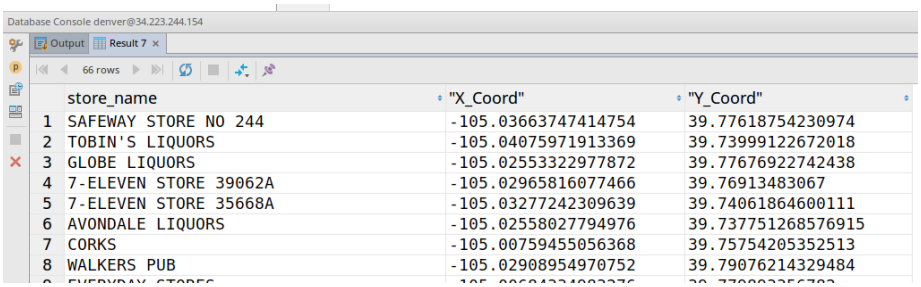
	id	st_astext
1	1	POINT(3130239.9999276 1707938.99986257)
2	2	POINT(3129147.99961699 1694747.99952422)
3	3	POINT(3133359.99976831 1708166.99986708)
4	4	POINT(3132214.99985319 1705379.99971762)
5	5	POINT(3131392.99972401 1694987.99950872)

Run the query and note the result coordinates are now in State Plane Colorado Central:

Write a query that will select the crime type, and show columns for the X and Y coordinates of each crime using the ST_X() and ST_Y() functions:

```
SELECT
    offense_ty as "Offense Type"
  , ST_X(geom) AS "X_Coord"
  , ST_Y(geom) AS "Y_Coord"
FROM denver.crime
```

Run the query to see the results:



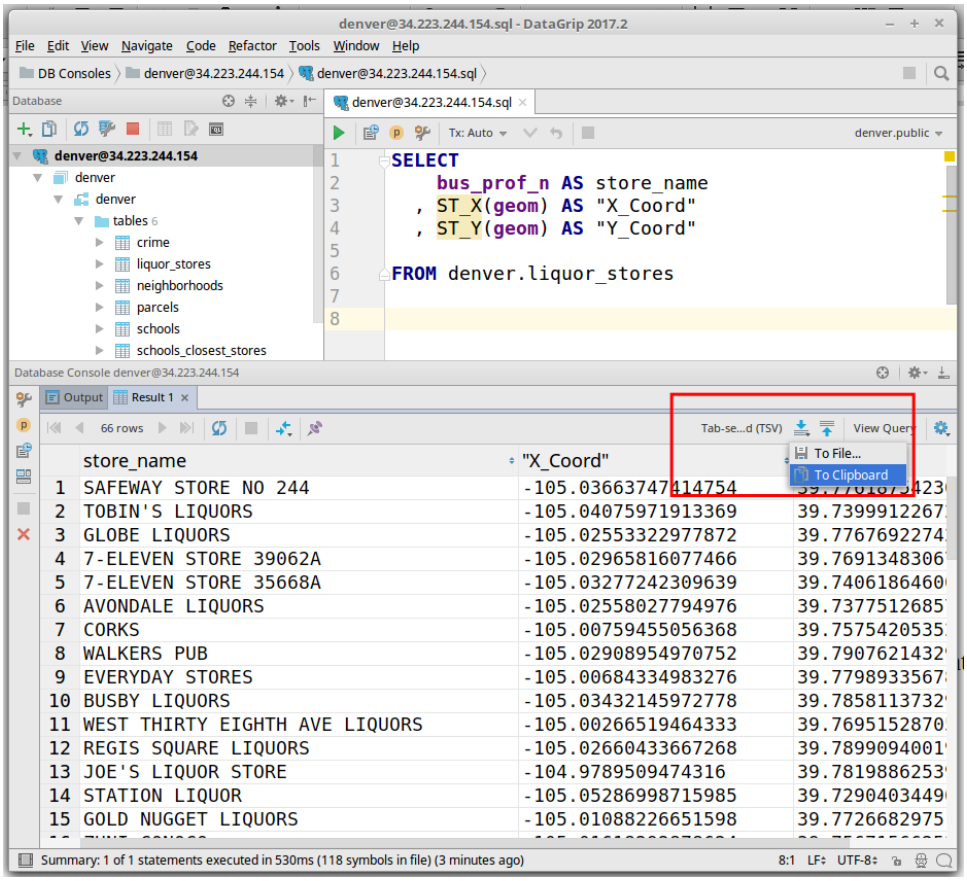
	store_name	"X_Coord"	"Y_Coord"
1	SAFEWAY STORE NO 244	-105.03663747414754	39.77618754230974
2	TOBIN'S LIQUORS	-105.04075971913369	39.73999122672018
3	GLOBE LIQUORS	-105.02553322977872	39.77676922742438
4	7-ELEVEN STORE 39062A	-105.02965816077466	39.76913483067
5	7-ELEVEN STORE 35668A	-105.03277242309639	39.74061864600111
6	AVONDALE LIQUORS	-105.02558027794976	39.737751268576915
7	CORKS	-105.00759455056368	39.75754205352513
8	WALKERS PUB	-105.02908954970752	39.79076214329484

Export Query Results to Spreadsheet

Export the results to a text file using the ***Copy > To Clipboard*** in the upper right-hand section of the results panel:

This loads the results of the query into the clipboard of your operating system.

Open a spreadsheet (Excel, LibreOffice Calc, OpenOffice), and paste the results into a new spreadsheet.

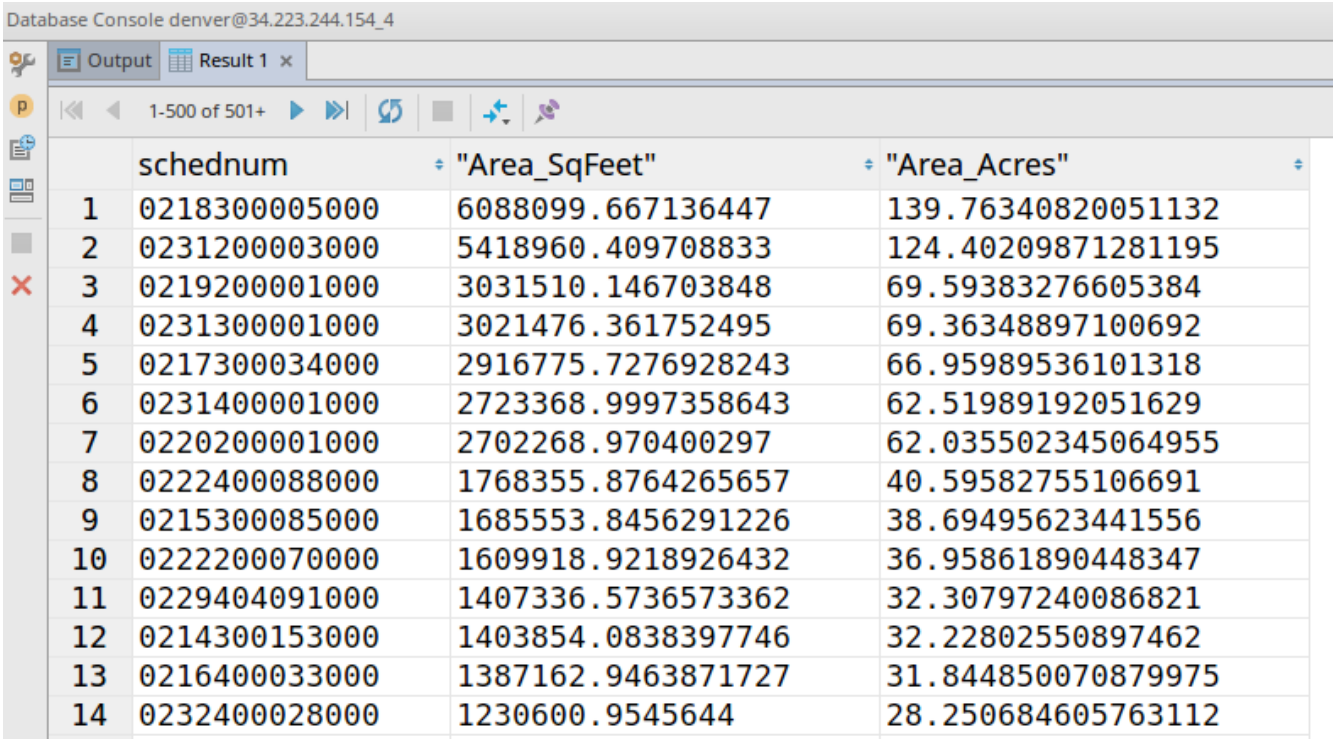


Calculate the area of polygons using ST_Area()

Use the ST_Area function to show the area of each denver.parcels table both in square feet and in acres, and order by area in descending order:

```
SELECT
  schednum --parcel ID
  , ST_Area(ST_Transform(geom, 2877)) as "Area_SqFeet"
  , ST_Area(ST_Transform(geom, 2877)) * 0.00002295682
         AS "Area_Acres"
FROM denver.parcels
order by ST_Area(geom) desc
```

Examine the results.



	schednum	"Area_SqFeet"	"Area_Acres"
1	0218300005000	6088099.667136447	139.76340820051132
2	0231200003000	5418960.409708833	124.40209871281195
3	0219200001000	3031510.146703848	69.59383276605384
4	0231300001000	3021476.361752495	69.36348897100692
5	0217300034000	2916775.7276928243	66.95989536101318
6	0231400001000	2723368.9997358643	62.51989192051629
7	0220200001000	2702268.970400297	62.035502345064955
8	0222400088000	1768355.8764265657	40.59582755106691
9	0215300085000	1685553.8456291226	38.69495623441556
10	0222200070000	1609918.9218926432	36.95861890448347
11	0229404091000	1407336.5736573362	32.30797240086821
12	0214300153000	1403854.0838397746	32.22802550897462
13	0216400033000	1387162.9463871727	31.844850070879975
14	0232400028000	1230600.9545644	28.250684605763112

Spatial Analysis with SQL Functions

Spatial Intersects with ST_Intersects()

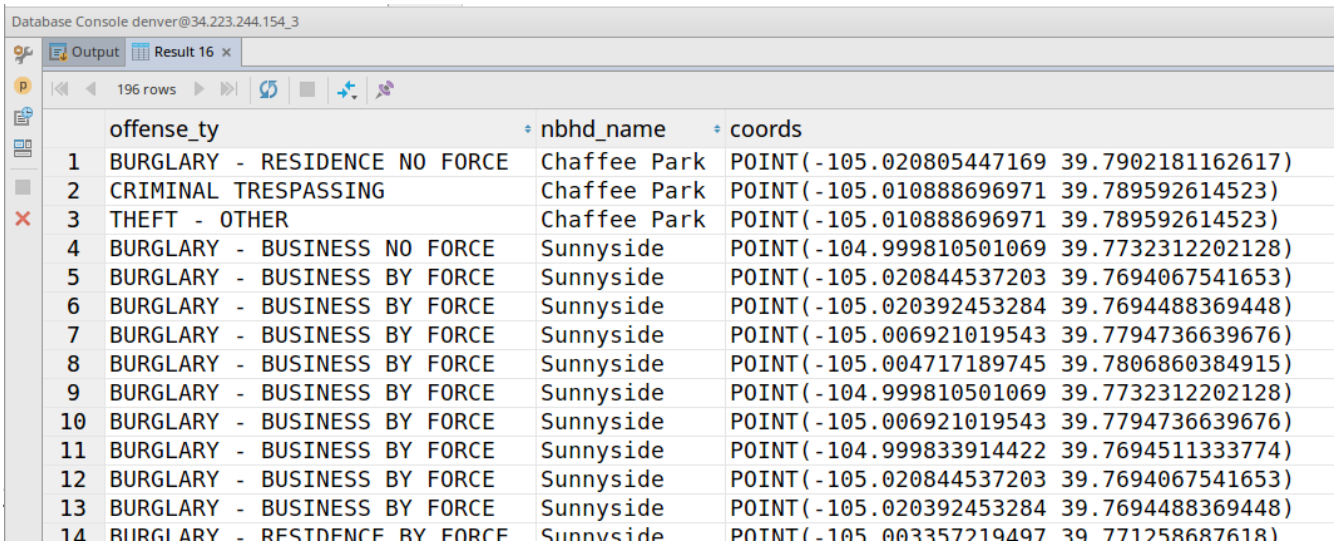
In order to link data spatially, the **ST_Intersects** function is used to spatially intersect points and polygons. This function uses the geometry from both tables as input parameters, and is primarily used in a JOIN condition on two tables

Write a query that will spatially intersects the crimes to the neighborhoods, selecting the crime type, neighborhood name, and geometry as text:

```
SELECT
  c.offense_ty
  , n.nbhd_name
  , ST_AsText(c.geom) as coords
FROM denver.crime AS c
  JOIN denver.neighborhoods AS n
    ON ST_Intersects(c.geom, n.geom)
```

Note: *When joining two tables together, alias each table with a short nickname in order to shorten references to columns of either table in SELECT statements. Below the denver.crimes table alias is* ***c*** *and the denver.neighborhoods table alias is* ***n****. You can see in the select statement the use of the aliases when selecting columns from each table*

And note the results:



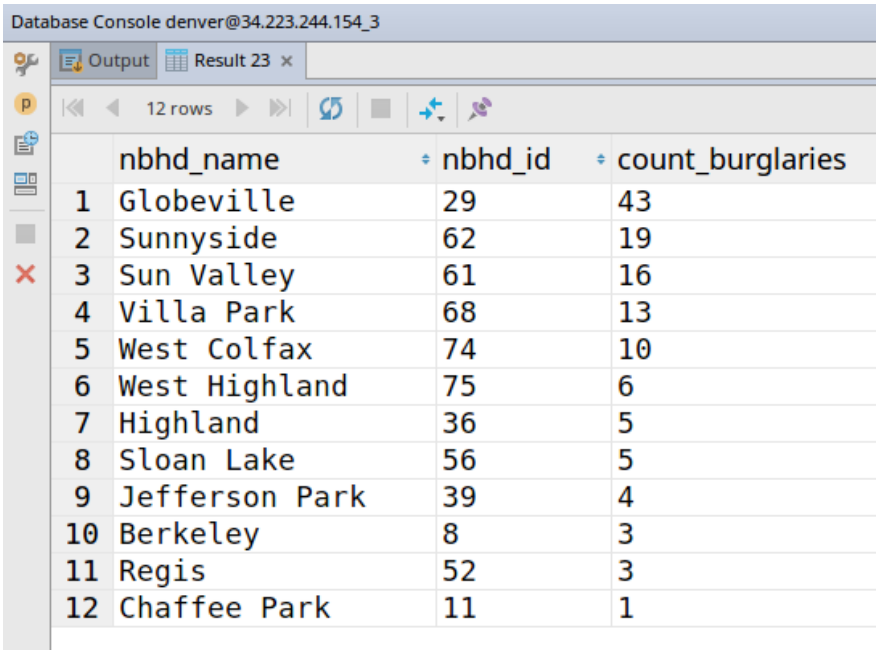
	offense_ty	nbhd_name	coords
1	BURGLARY - RESIDENCE NO FORCE	Chaffee Park	POINT(-105.020805447169 39.7902181162617)
2	CRIMINAL TRESPASSING	Chaffee Park	POINT(-105.010888696971 39.789592614523)
3	THEFT - OTHER	Chaffee Park	POINT(-105.010888696971 39.789592614523)
4	BURGLARY - BUSINESS NO FORCE	Sunnyside	POINT(-104.999810501069 39.7732312202128)
5	BURGLARY - BUSINESS BY FORCE	Sunnyside	POINT(-105.020844537203 39.7694067541653)
6	BURGLARY - BUSINESS BY FORCE	Sunnyside	POINT(-105.020392453284 39.7694488369448)
7	BURGLARY - BUSINESS BY FORCE	Sunnyside	POINT(-105.006921019543 39.7794736639676)
8	BURGLARY - BUSINESS BY FORCE	Sunnyside	POINT(-105.004717189745 39.7806860384915)
9	BURGLARY - BUSINESS BY FORCE	Sunnyside	POINT(-104.999810501069 39.7732312202128)
10	BURGLARY - BUSINESS BY FORCE	Sunnyside	POINT(-105.006921019543 39.7794736639676)
11	BURGLARY - BUSINESS BY FORCE	Sunnyside	POINT(-104.999833914422 39.7694511333774)
12	BURGLARY - BUSINESS BY FORCE	Sunnyside	POINT(-105.020844537203 39.7694067541653)
13	BURGLARY - BUSINESS BY FORCE	Sunnyside	POINT(-105.020392453284 39.7694488369448)
14	BURGLARY - RESIDENCE BY FORCE	Sunnyside	POINT(-105.003357219497 39.771258687618)

Count Points in Polygons

Using what we know about counting and grouping, and having both tables joined using the ST_Intersects function, we can now count the number of burglaries by neighborhood, and order the results from most to least:

```
SELECT
  n.nbhd_name
  , n.nbhd_id
  , count(*) AS count_burglaries
FROM denver.crime AS c
  JOIN denver.neighborhoods AS n
    ON ST_Intersects(c.geom, n.geom)
WHERE c.offense_ty LIKE 'BURG%'
GROUP BY n.nbhd_id, n.nbhd_name
ORDER BY count(*) DESC
```

Note the results:



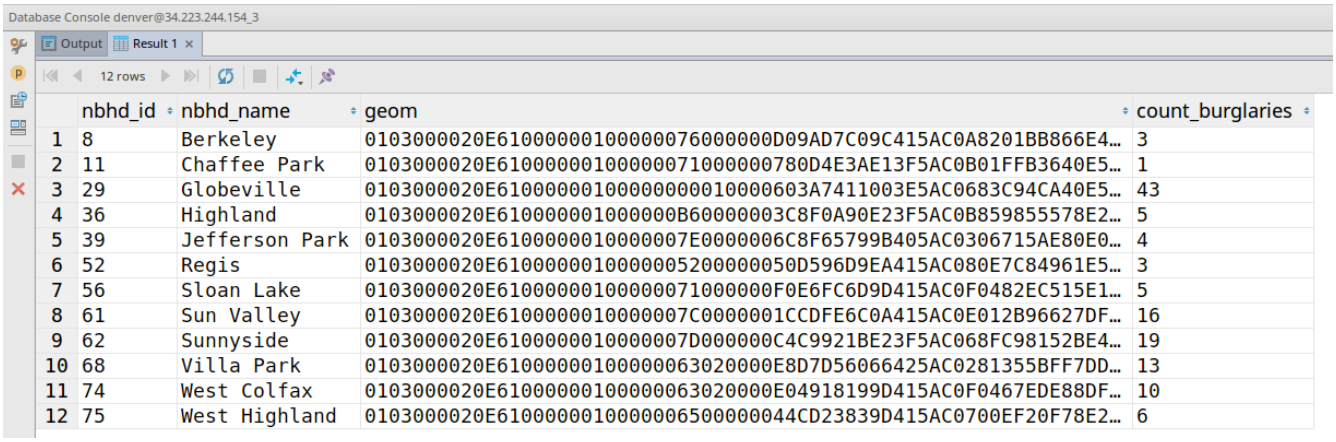
	nbhd_name	nbhd_id	count_burglaries
1	Globeville	29	43
2	Sunnyside	62	19
3	Sun Valley	61	16
4	Villa Park	68	13
5	West Colfax	74	10
6	West Highland	75	6
7	Highland	36	5
8	Sloan Lake	56	5
9	Jefferson Park	39	4
10	Berkeley	8	3
11	Regis	52	3
12	Chaffee Park	11	1

Create a table of the results using CREATE TABLE

In order to map the results, included the geom column in the select and group by statements:

```
SELECT
  n.nbhd_id
  , n.nbhd_name
  , n.geom
  , count(*) AS count_burglaries

FROM denver.crime AS c

  JOIN denver.neighborhoods AS n
    ON ST_Intersects(c.geom, n.geom)

WHERE c.offense_ty LIKE 'BURG%'


GROUP BY n.nbhd_id, nbhd_name, n.geom
```

Note the results:

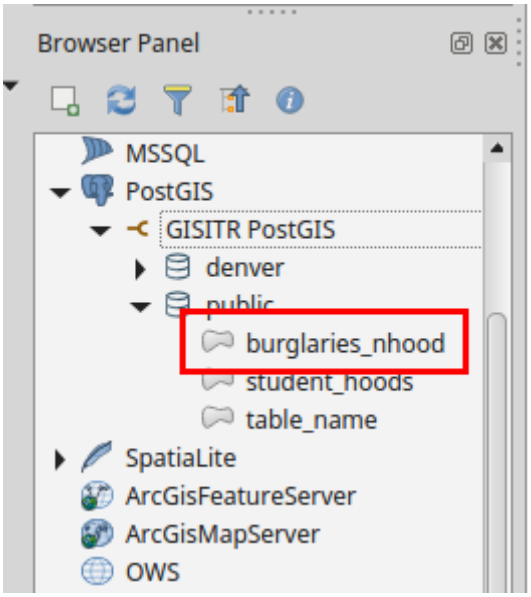


	nbhd_id	nbhd_name	geom	count_burglaries
1	8	Berkeley	0103000020E6100000010000007600000D09AD7C09C415AC0A8201BB866E4…	3
2	11	Chaffee Park	0103000020E610000001000000710000007 80D4E3AE13F5AC0B01FFB3640E5…	1
3	29	Globeville	0103000020E6100000010000000010000603A7411003E5AC0683C94CA40E5…	43
4	36	Highland	0103000020E6100000010000000B60000003C8F0A90E23F5AC0B859855578E2…	5
5	39	Jefferson Park	0103000020E610000001000000 7E0000006C8F65799B405AC0306715AE80E0…	4
6	52	Regis	0103000020E6100000010000005200000050D596D9EA415AC080E7C84961E5…	3
7	56	Sloan Lake	0103000020E610000001000000 71000000F0E6FC6D9D415AC0F0482EC515E1…	5
8	61	Sun Valley	0103000020E610000001000000 7C0000001CCDFE6C0A415AC0E012B96627DF…	16
9	62	Sunnyside	0103000020E610000001000000 7D000000C4C9921BE23F5AC068FC98152BE4…	19
10	68	Villa Park	0103000020E6100000010000006 3020000E8D7D56066425AC0281355BFF7DD…	13
11	74	West Colfax	0103000020E6100000010000006 3020000E04918199D415AC0F0467EDE88DF…	10
12	75	West Highland	0103000020E6100000010000006 500000044CD23839D415AC0700EF20F78E2…	6

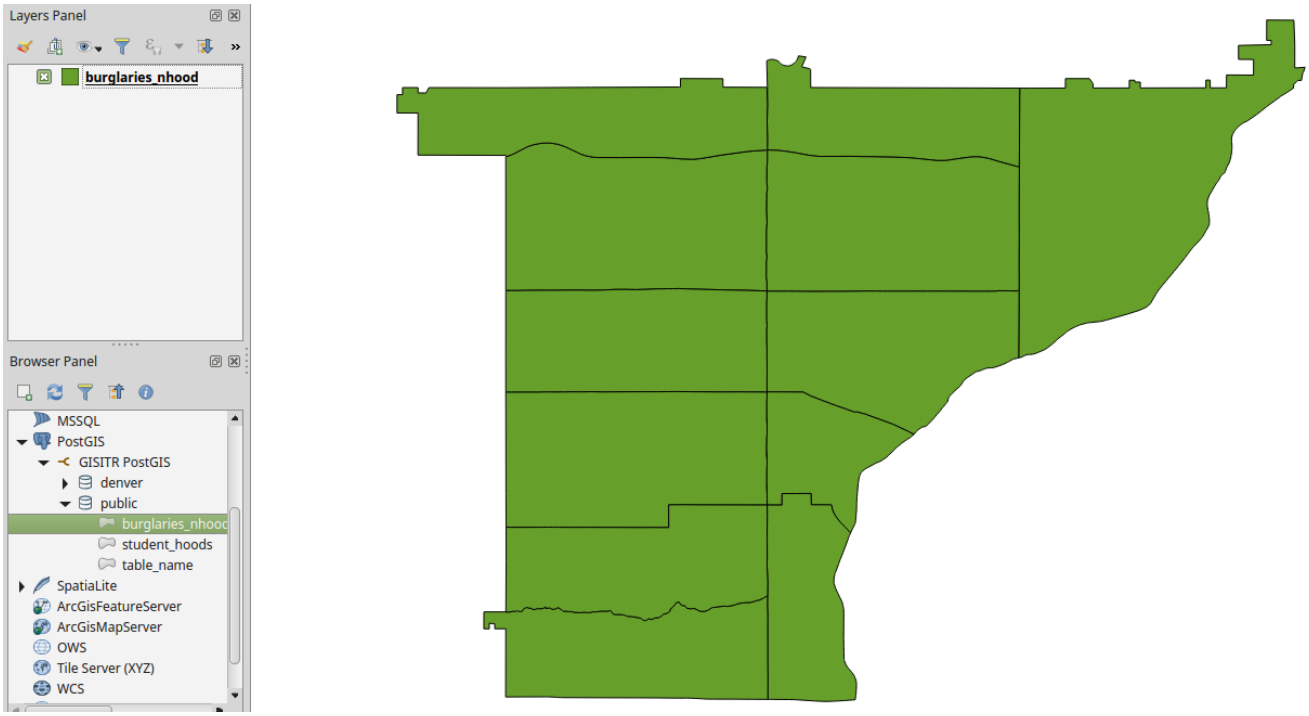
The results of this query can be stored as a table and viewed in QGIS. To create a table from the results, wrap a CREATE TABLE expression around the main query – note we are creating the table in the **public** schema (as noted in the schemaName.newTableName syntax):

```
CREATE TABLE public.burglaries_nhood AS (
  SELECT
    n.nbhd_id
    , n.nbhd_name
    , n.geom
    , count(*) AS count_burglaries
  FROM denver.crime AS c
    JOIN denver.neighborhoods AS n
      ON ST_Intersects(c.geom, n.geom)
  WHERE c.offense_ty LIKE 'BURG%'
  GROUP BY n.nbhd_id, nbhd_name, n.geom
)
```

In QGIS, refresh the PUBLIC schema to see the new table:

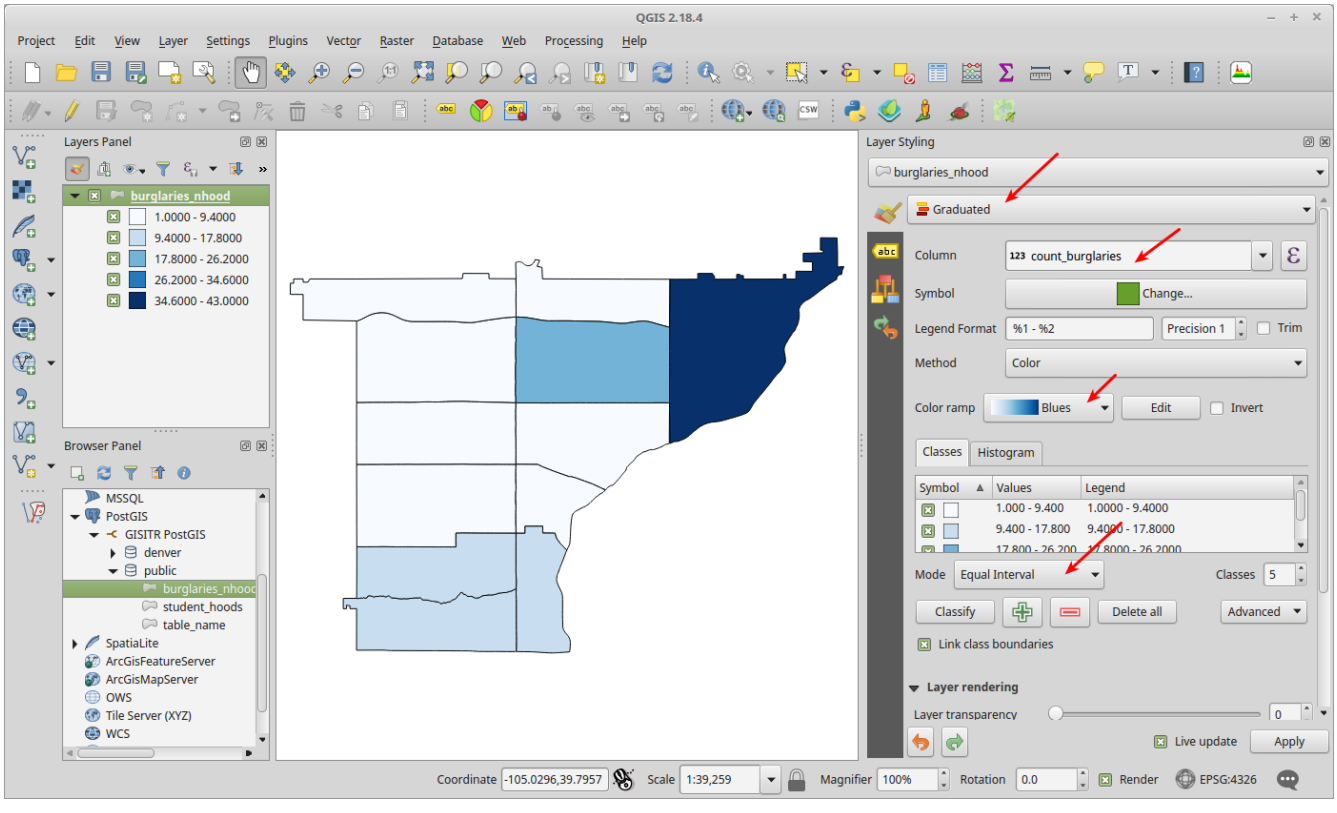


Double click the public.buglaries_nhood layer to add it to the map:



The layer contains all the geometry of the input denver.neighborhoods layer, but with the added column for *count_burglaries* as in the query that created it.

Use the layer styling panel to assign a graduated renderer to the layer using the ***count_burglaries*** column using an ***Equal Interval*** renderer and the ***blues*** color ramp:



Count features within a distance using ST_DWithin()

A common operation is to count features that are within a distance of a source feature. The PostGIS function ST_DWithin() is the best way to do this. Instead of creating a buffer around a source feature and using the buffer to count the features that intersect, the ST_DWithin() function skips the buffer, and uses the internal spatial intersect to make the query fast.

We'll use this function to calculate the number of crimes from each school, then determine which school has the highest number of crimes within ½ mile of each school.

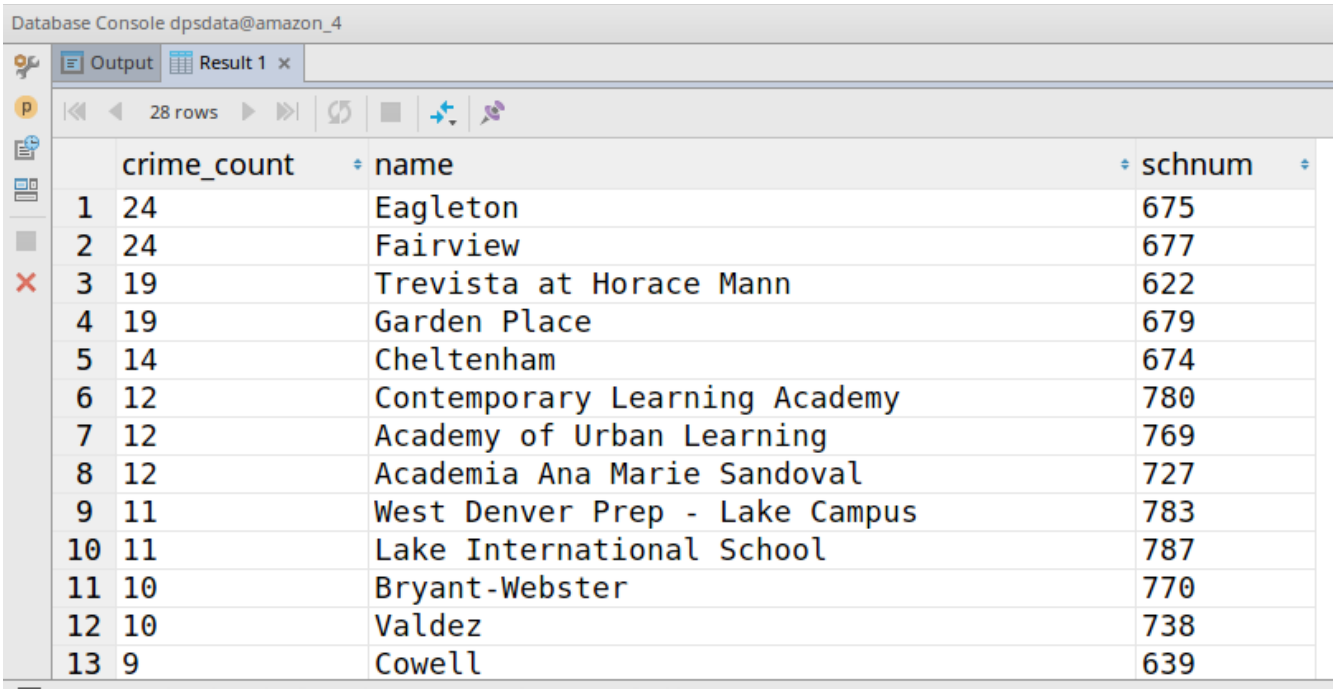
The full syntax of the function is ***ST_DWithin(geom1, geom2, distance)***

Remember that our geometry is in WGS84, and the units are decimal degrees. In order to use a 1/2mi, or 2640', we'll transform the geometry of each input (schools, crime) to State Plane feet on the fly.

Write a query to count the number of crimes within 2640' of each school, and note the use of ST_DWithin() in the ***where*** clause of the query as well as well as the ***ORDER BY*** using the count(*) expression in ***DESC*** order:

```
SELECT
   count(*)   AS crime_count
  , sch.name
  , sch.orig_fid AS schnum

FROM denver.schools AS sch
  , denver.crime AS cr

WHERE ST_DWithin(
   ST_Transform(cr.geom, 2877)
   , ST_Transform(sch.geom, 2877)
   , 2640)

GROUP BY sch.orig_fid, sch.name


ORDER BY count(*) DESC
```

And note the results:



	crime_count	name	schnum
1	24	Eagleton	675
2	24	Fairview	677
3	19	Trevista at Horace Mann	622
4	19	Garden Place	679
5	14	Cheltenham	674
6	12	Contemporary Learning Academy	780
7	12	Academy of Urban Learning	769
8	12	Academia Ana Marie Sandoval	727
9	11	West Denver Prep - Lake Campus	783
10	11	Lake International School	787
11	10	Bryant-Webster	770
12	10	Valdez	738
13	9	Cowell	639

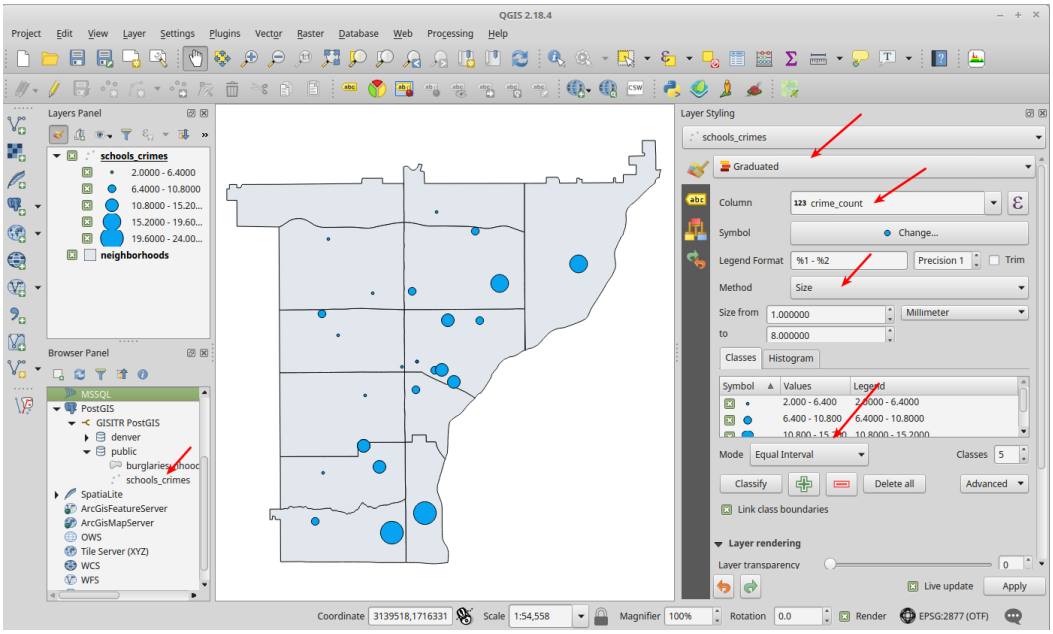
As with counting points in polygons, adding the GEOM column to the SELECT and GROUP BY, and wrapping the expression in a CREATE TABLE statement will create a new table with the results that can be mapped in QGIS.

Use the CREATE TABLE expression to map the results, including the GEOM column, noting that we removed the ORDER BY statement:

```
CREATE TABLE public.schools_crimes AS (
 SELECT
    count(*)      AS crime_count
   , sch.name
   , sch.orig_fid AS schnum
   , sch.geom

 FROM denver.schools AS sch
   , denver.crime AS cr

 WHERE ST_DWithin(
    ST_Transform(cr.geom, 2877)
    , ST_Transform(sch.geom, 2877)
    , 2640)
 GROUP BY sch.orig_fid, sch.name, sch.geom)
```

Open the new layer in QGIS.

Assign a ***graduated*** renderer on the ***crime_count*** column, using the ***size*** method (instead of color), and an ***equal interval*** renderer type:



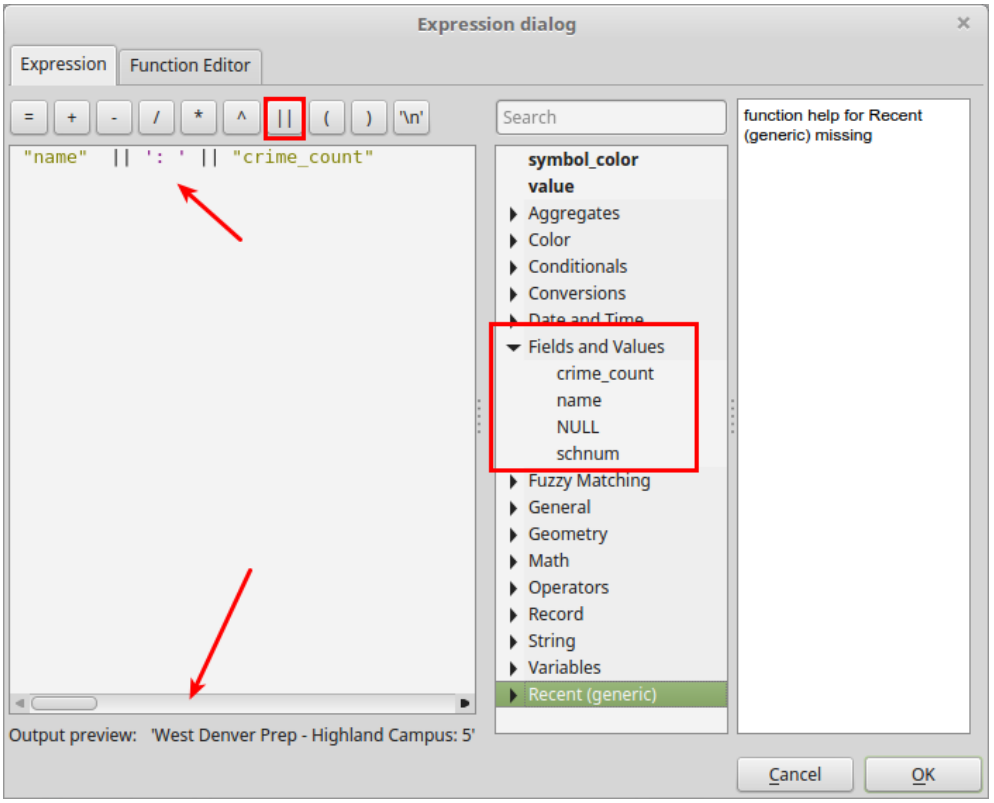
Next, assign a label to the points that shows the school name, and the number of crimes.

To do this, open the ***label expression*** box.

Write an expression that uses

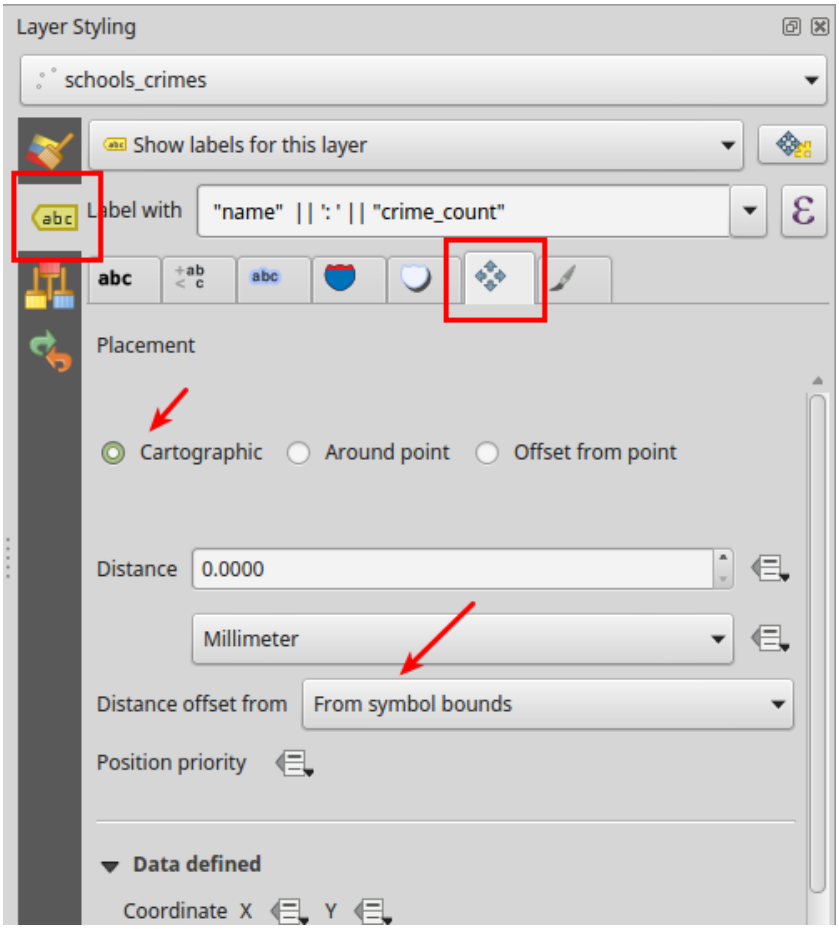
- the ***name*** field
- concatenate a colon and a space
- the ***crime_count*** field

Click OK when you see a valid output preview in the bottom of the dialogue box.

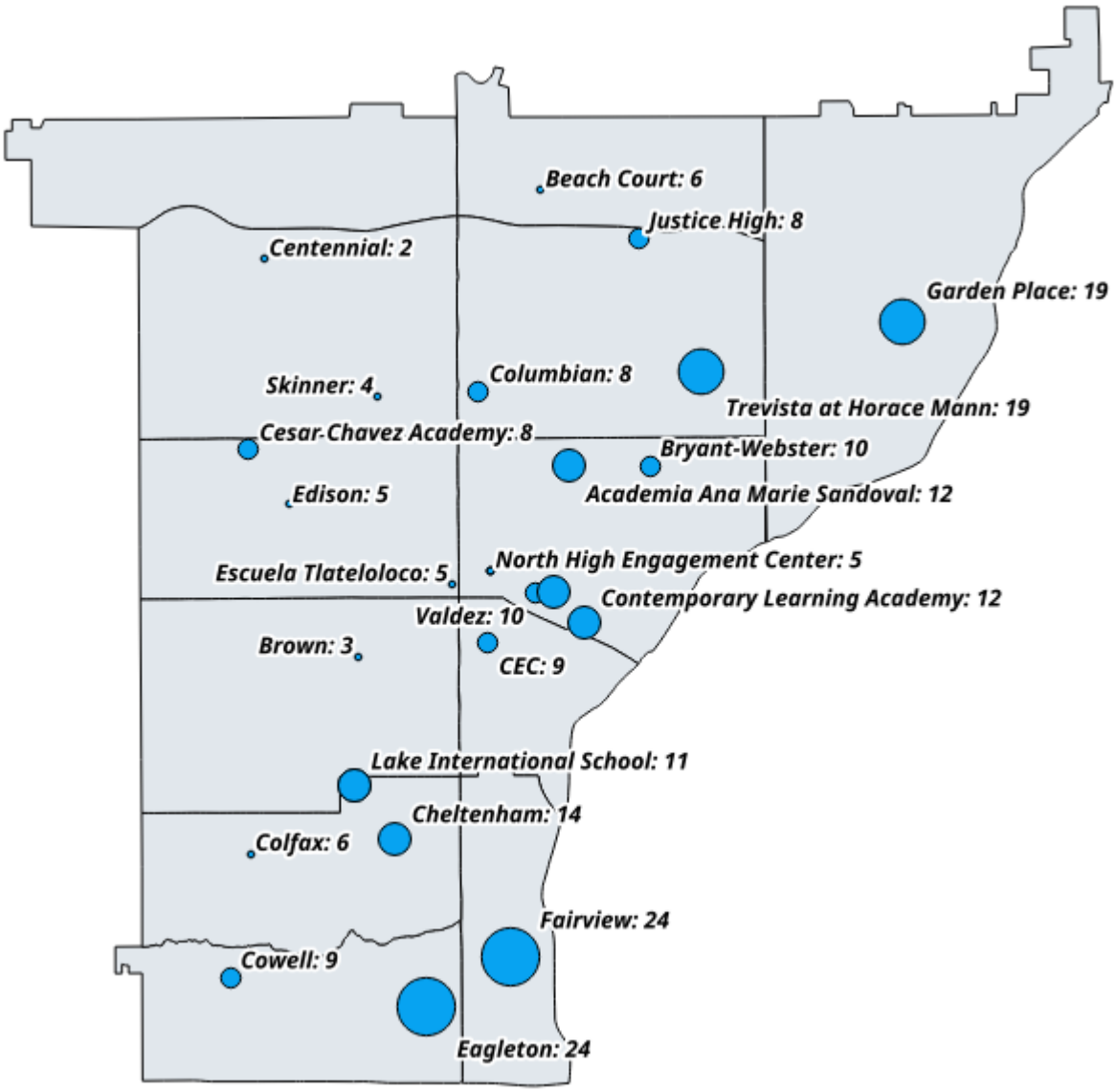


Additionally, use the ***label styling*** panel to

- add a halo around the labels, and
- ensure none of the labels overlap features by setting the placement option for Distance Offset from to ***from symbol bounds:***



Note the results on the map:



Calculate Distance using ST_Distance()

In the previous exercise we used ST_DWithin() to select target features (crimes) within a distance of source features (schools). That function uses distance under the hood, so in order to see the distances, we'll use the ST_Distance(geom1, geom2) function.

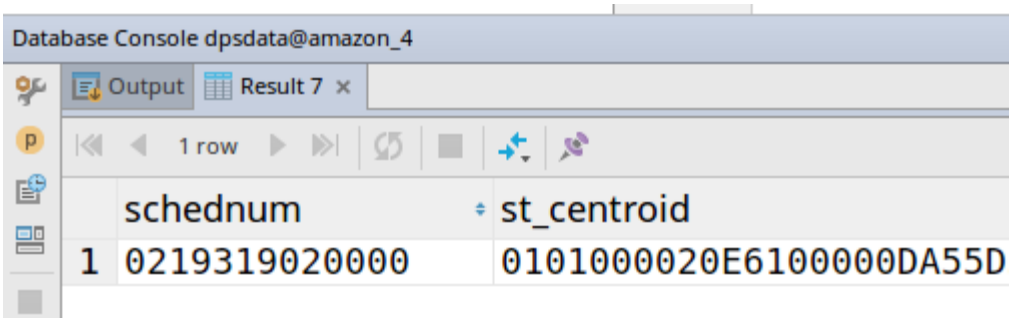
Use the ST_Centroid() function to convert polygons to point centroids

For the next exercise, we'll calculate the closest 10 schools to a particular parcel feature. To do this, we'll run the ST_Distance() function on the centroid of the parcel.

Write a query to select the centroid of 1 parcel by its SCHEDNUM:

```
SELECT p.schednum
, ST_Centroid(p.geom)
FROM denver.parcels AS p
WHERE p.schednum = '0219319020000'
```

Note the results:



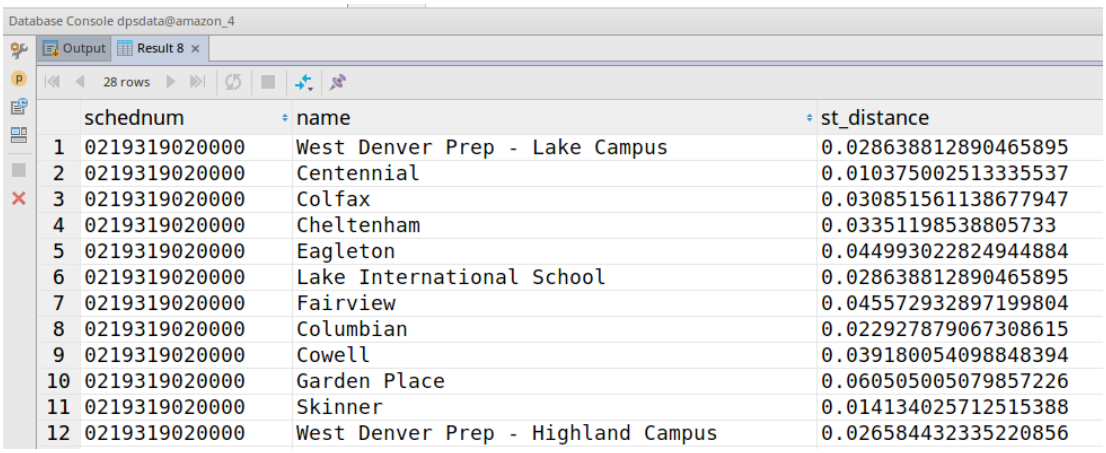
Calculate the distance from a parcel to each school

The ST_Distance() function can be used in a SELECT statement, and instead of joining two tables, we'll simply select multiple tables in our query.

Write a query that will select the distance each school is from a single parcel centroid:

```
SELECT
  p.schednum
  , sch.name
  , ST_Distance(ST_Centroid(p.geom), sch.geom)
FROM denver.parcels AS p
  , denver.schools AS sch
WHERE p.schednum = '0219319020000'
```

Note the results:



Database Console dpsdata@amazon_4

Output | Result 8 ×

28 rows

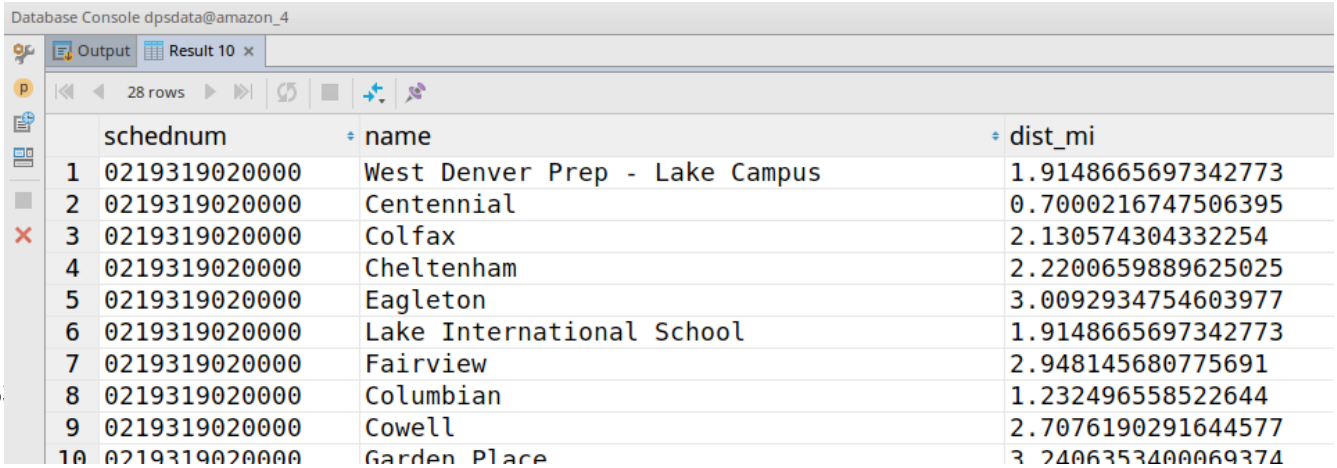
	schednum	name	st_distance
1	0219319020000	West Denver Prep - Lake Campus	0.028638812890465895
2	0219319020000	Centennial	0.010375002513335537
3	0219319020000	Colfax	0.030851561138677947
4	0219319020000	Cheltenham	0.03351198538805733
5	0219319020000	Eagleton	0.044993022824944884
6	0219319020000	Lake International School	0.028638812890465895
7	0219319020000	Fairview	0.045572932897199804
8	0219319020000	Columbian	0.022927879067308615
9	0219319020000	Cowell	0.039180054098848394
10	0219319020000	Garden Place	0.060505005079857226
11	0219319020000	Skinner	0.014134025712515388
12	0219319020000	West Denver Prep - Highland Campus	0.026584432335220856

The units of the ST_Distance() function are in decimal degrees, since our data is in WGS84.

Transform the geometry of both input tables to State Plane Colorado Central (SRID: 2877) in order to get the distance in feet, then divide the result by 5280 to get the distance in Miles:

```
SELECT
  p.schednum
  , sch.name
  , ST_Distance(
      ST_Transform(ST_Centroid(p.geom), 2877)
      , ST_Transform(sch.geom, 2877)
    ) / 5280 AS dist_mi
FROM denver.parcels AS p
  , denver.schools AS sch
WHERE p.schednum = '0219319020000'
```

And note the results:



Database Console dpsdata@amazon_4

Output | Result 10 ×

28 rows

	schednum	name	dist_mi
1	0219319020000	West Denver Prep - Lake Campus	1.9148665697342773
2	0219319020000	Centennial	0.7000216747506395
3	0219319020000	Colfax	2.130574304332254
4	0219319020000	Cheltenham	2.2200659889625025
5	0219319020000	Eagleton	3.0092934754603977
6	0219319020000	Lake International School	1.9148665697342773
7	0219319020000	Fairview	2.948145680775691
8	0219319020000	Columbian	1.232496558522644
9	0219319020000	Cowell	2.7076190291644577
10	0219319020000	Garden Place	3.2406353400069374

3

Create a line from a parcel to each school using ST_MakeLine()

If we want to visualize the location and distance from each school to a single parcel, we can add the ST_MakeLine() function to our selection.

The ***ST_MakeLine(geom1, geom2)*** function will generate a geometry between the input and target features.

Write a query to create a line between each school and the target parcel, and alias the line geometry ***geom_line***:

```
SELECT
  p.schednum
  , sch.name
  , ST_Distance(
      ST_Transform(ST_Centroid(p.geom), 2877)
      , ST_Transform(sch.geom, 2877)
    ) / 5280 AS dist_mi
  , ST_MakeLine(ST_Centroid(p.geom), sch.geom) as geom_line
FROM denver.parcels AS p
  , denver.schools AS sch
WHERE p.schednum = '0219319020000'
```

Note the results:

Database Console dpsdata@amazon_4

Output | Result 11

28 rows

	schednum	name	dist_mi	st_makeline
1	0219319020000	West Denver Prep - Lake Campus	1.9148665697342773	0102000020E610000002000000D/
2	0219319020000	Centennial	0.7000216747506395	0102000020E610000002000000D/
3	0219319020000	Colfax	2.130574304332254	0102000020E610000002000000D/
4	0219319020000	Cheltenham	2.2200659889625025	0102000020E610000002000000D/
5	0219319020000	Eagleton	3.0092934754603977	0102000020E610000002000000D/
6	0219319020000	Lake International School	1.9148665697342773	0102000020E610000002000000D/
7	0219319020000	Fairview	2.948145680775691	0102000020E610000002000000D/
8	0219319020000	Columbian	1.232496558522644	0102000020E610000002000000D/
9	0219319020000	Cowell	2.7076190291644577	0102000020E610000002000000D/
10	0219319020000	Garden Place	3.2406353400069374	0102000020E610000002000000D/
11	0219319020000	Skinner	0.7642401682219573	0102000020E610000002000000D/
12	0219319020000	West Denver Prep - Highland Campus	1.5158583979164957	0102000020E610000002000000D/

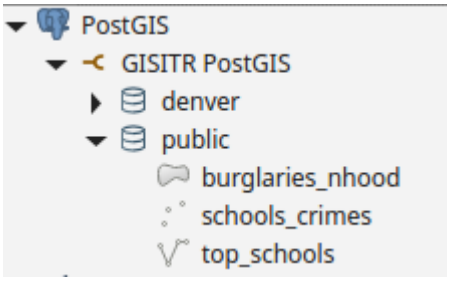
If we only wanted to see the top 10 schools, we can order the results by distance, and limit the query to return only the top 10:

```
SELECT
  p.schednum
  , sch.name
  , ST_Distance(
      ST_Transform(ST_Centroid(p.geom), 2877)
      , ST_Transform(sch.geom, 2877)
    ) / 5280 AS dist_mi
  , ST_MakeLine(ST_Centroid(p.geom), sch.geom) as geom_line
FROM denver.parcels AS p
  , denver.schools AS sch
WHERE p.schednum = '0219319020000'
 ORDER BY ST_Distance(ST_Centroid(p.geom), sch.geom) desc
 limit 10
```

Wrap a CREATE TABLE expression around the previous function to create a table of the results for display in QGIS:

```
create table public.top_schools as (
  SELECT
    p.schednum
    , sch.name
    , ST_Distance(
        ST_Transform(ST_Centroid(p.geom), 2877)
        , ST_Transform(sch.geom, 2877)
      ) / 5280                                    AS dist_mi
    , ST_MakeLine(ST_Centroid(p.geom), sch.geom) AS geom_line

  FROM denver.parcels AS p
    , denver.schools AS sch

  WHERE p.schednum = '0219319020000'

  ORDER BY ST_Distance(ST_Centroid(p.geom), sch.geom) DESC

  LIMIT 10
)
```

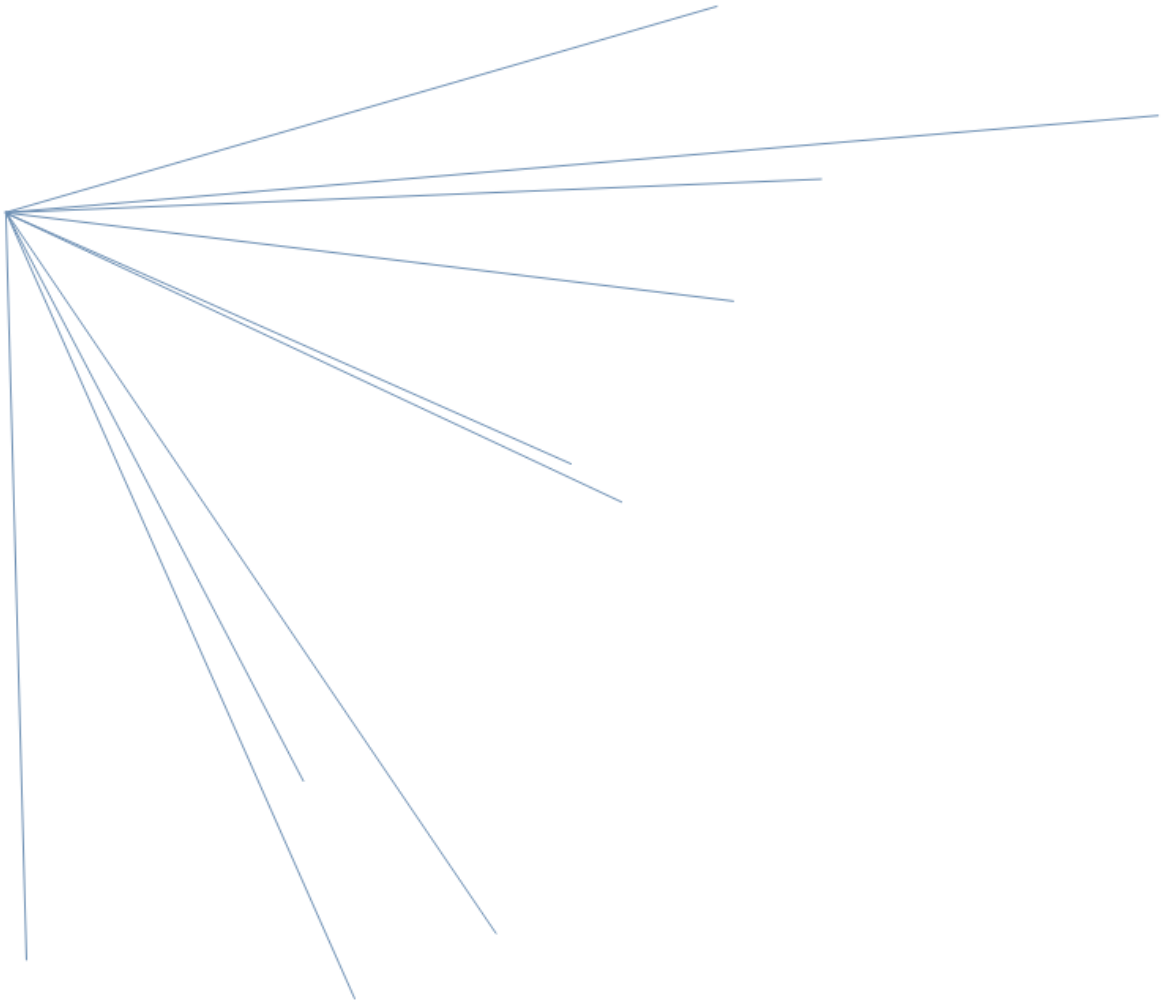
Refresh the ***public*** schema in QGIS and note the new table is read as having line geometry:



Since the only geometry we selected was from the ST_MakeLine() function, QGIS reads the result column as lines.

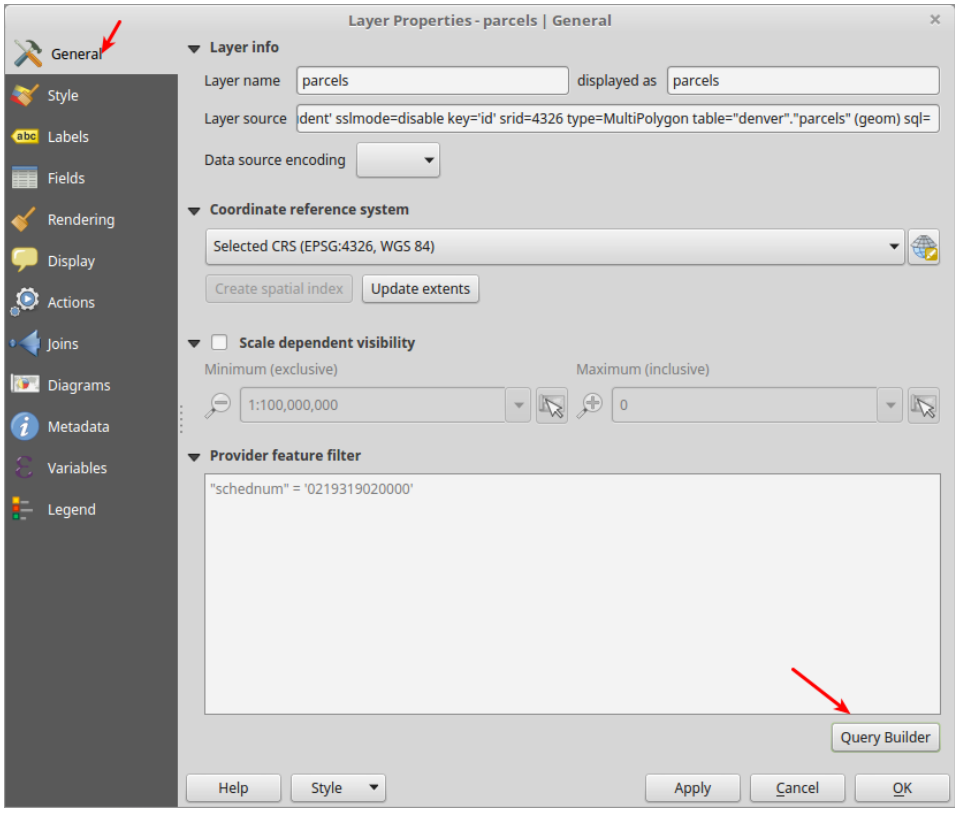
PostGIS allows the storage of multiple geometry columns, so in theory, you could store the point geometry of each school as well as the line geometry to the parcel.

Add the ***top_schools*** layer to your map:



Add the ***denver.parcels*** layer and filter on the SCHEDNUM we used above.

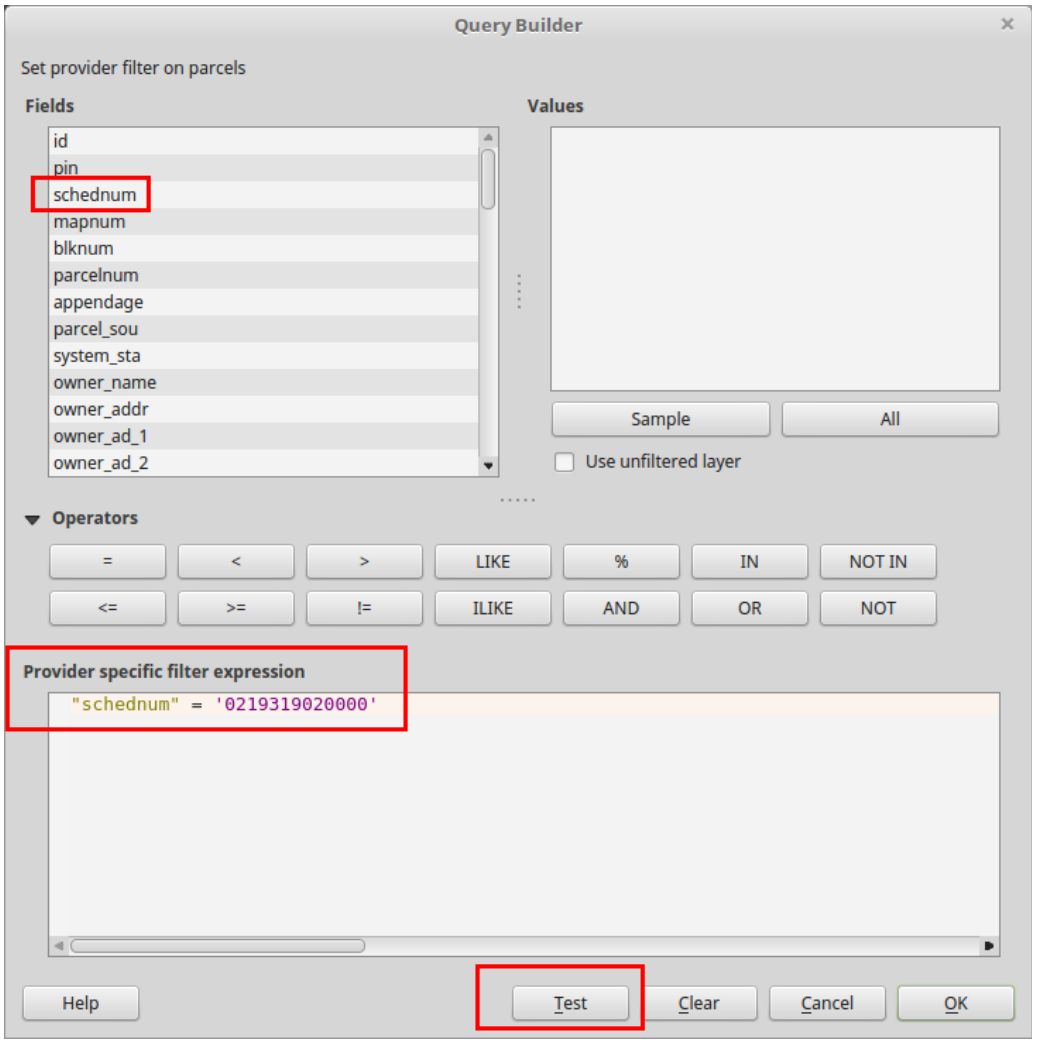
To filter the display of a layer, open the properties (right-click > Properties), and open the ***Query Builder*** window:



Write an expression to filter on the SCHEDNUM column:

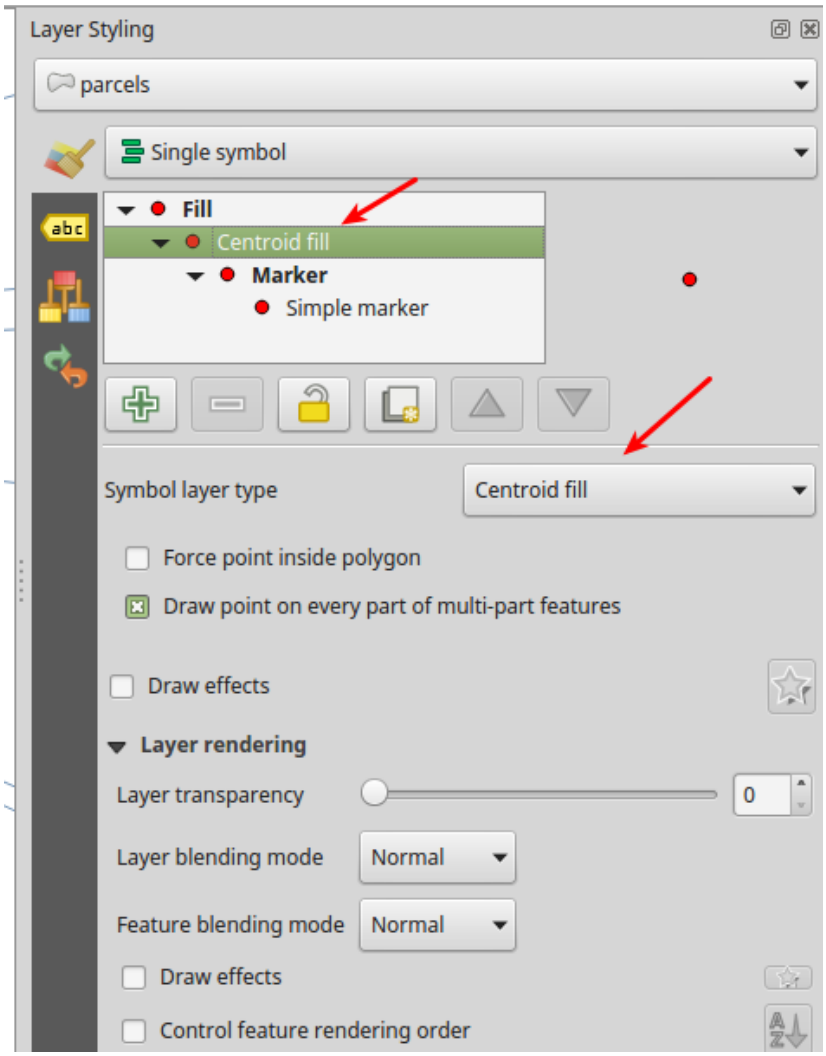
Click the Test button to ensure your expression works, click OK out of the Query Builder and Layer Properties dialogue boxes.

Next, style the single parcel as its centroid from the layer styling dialogue box:



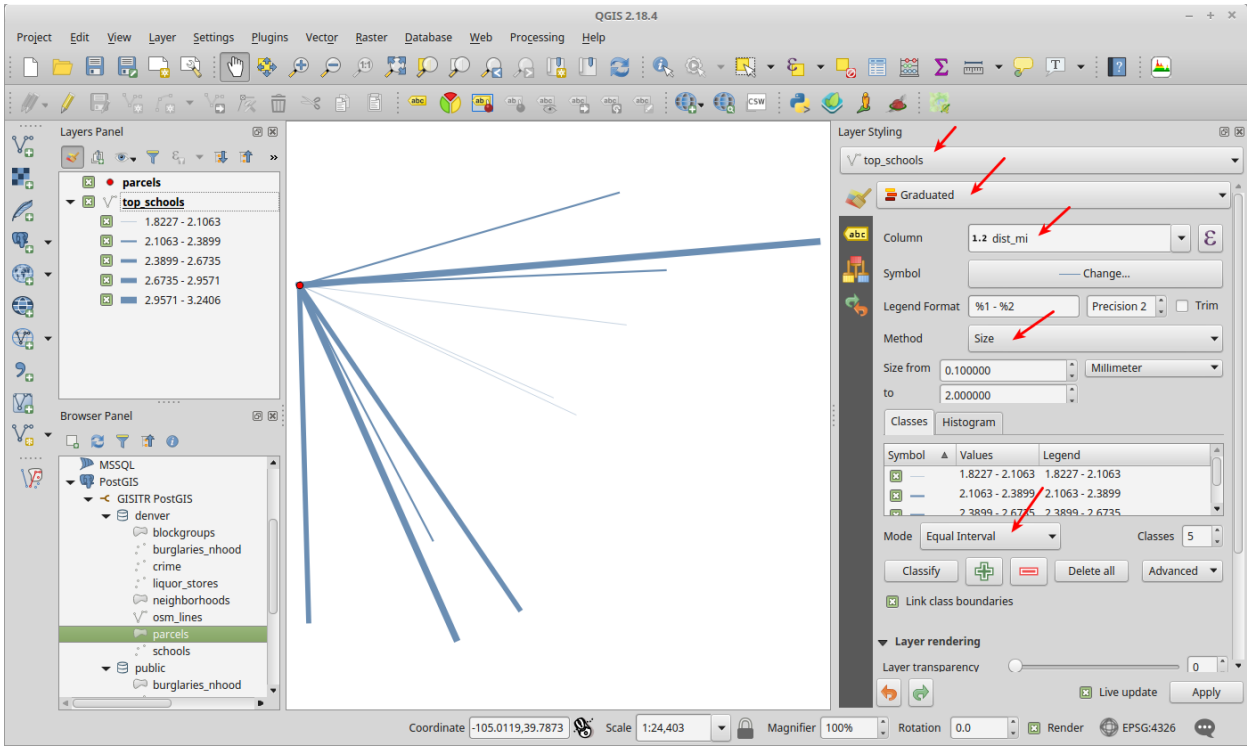
Next, symbolize the parcels as the centroid of the polygon, rather than the shape.

In the layer styling panel, change the ***Symbol Layer Type*** to ***Centroid Fill:***



Note the result on the map.

Next, style the lines by their distance values using a graduated renderer on the ***dist_mi*** column, and set the ***Method*** to ***size***:

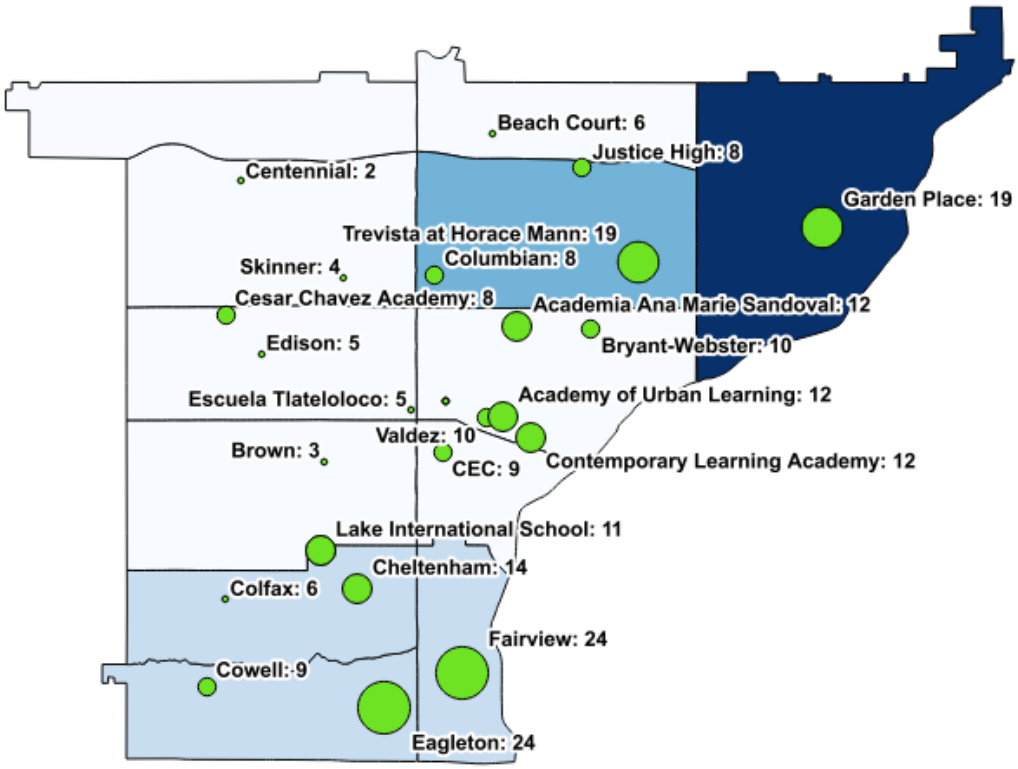


Create a final map in QGIS

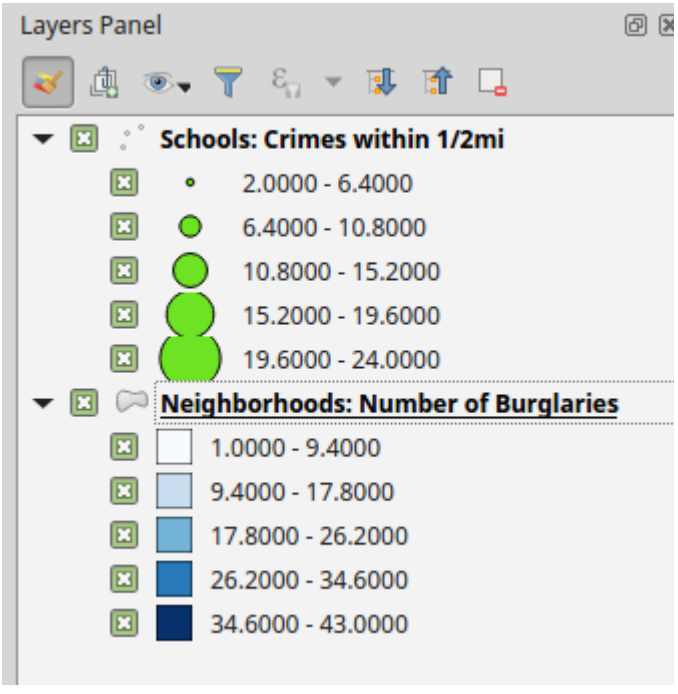
Style datasets for the map

Add both the ***burglaries_nhood*** and ***schools_crimes*** layers to a new QGIS map.

Style both layers as in the previous examples:

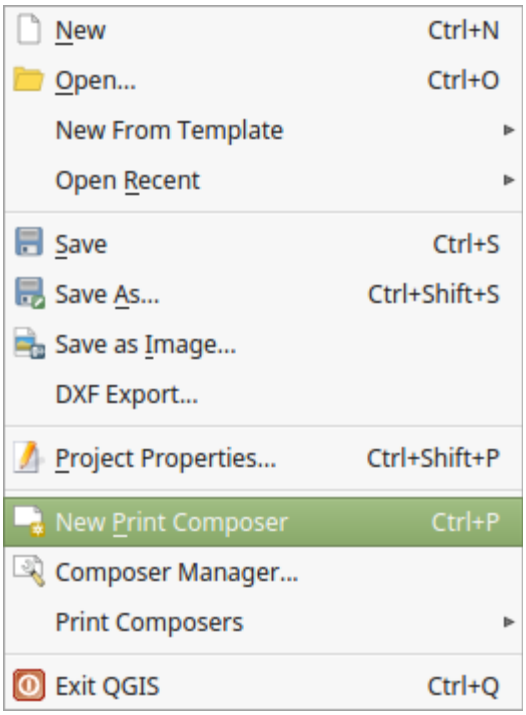


In the Layers panel, rename each layer to something more informative:

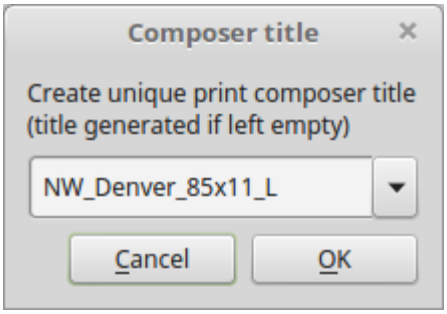


Create a Print Composer

From the QGIS Project menu, choose New Print Composer:



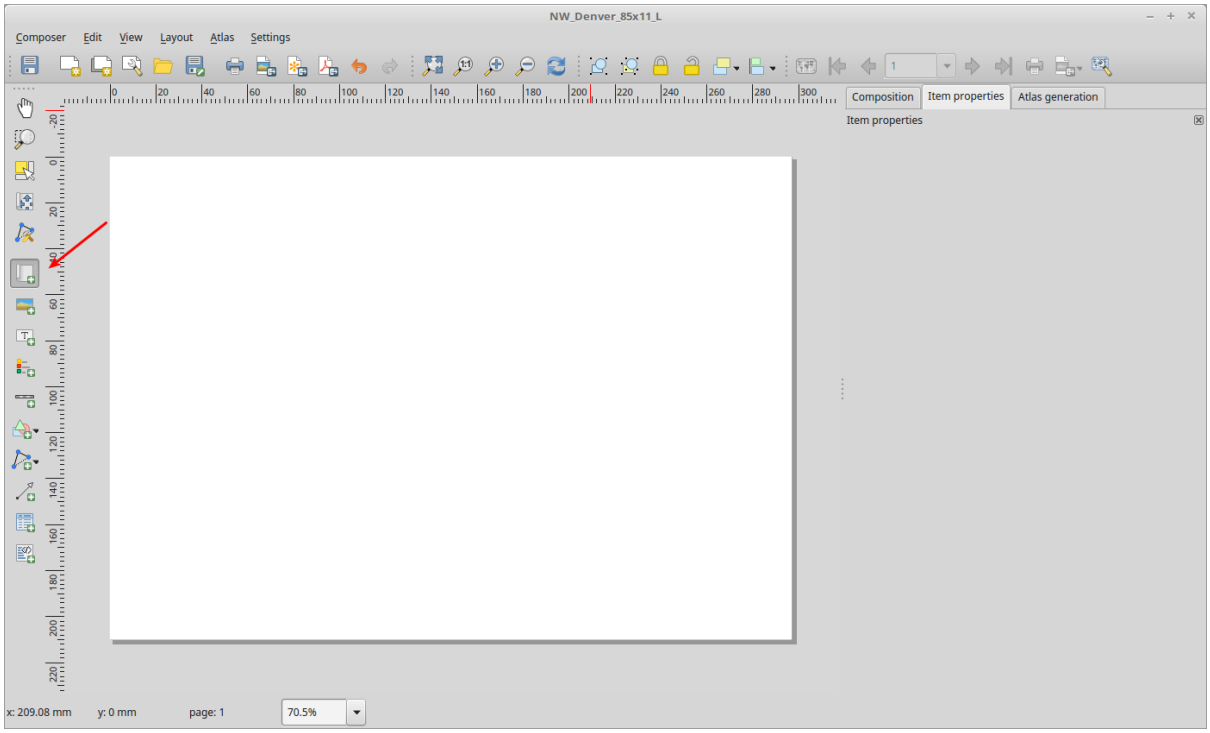
Give the Composer a title that you can use to recognize the type of map you're making – in this exercise, we'll create an 8.5x11 landscape plot, so name the composer: ***NW_Denver_85x11_L***



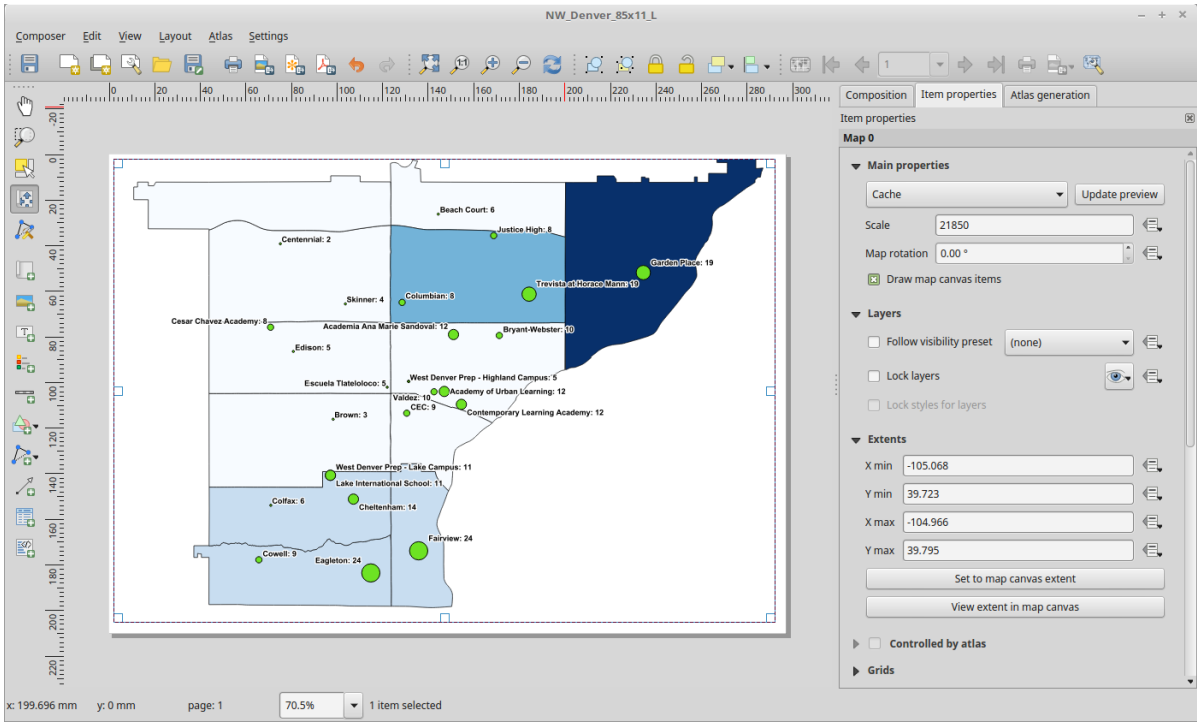
Add a map to the canvas

The Print Composer opens with an empty page.

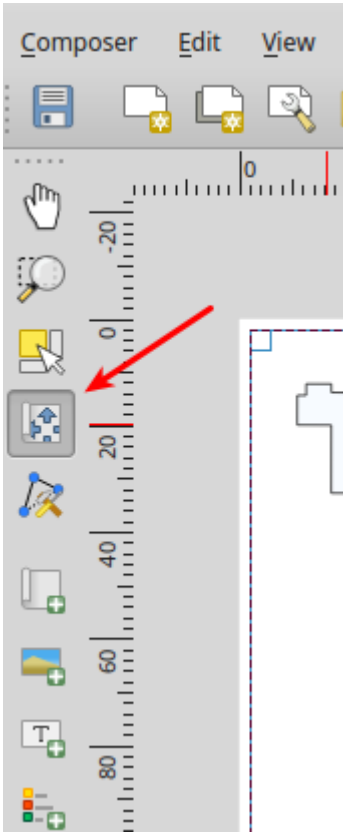
Click the ***Add new map*** icon, and draw a rectangle where the map will appear on the page:



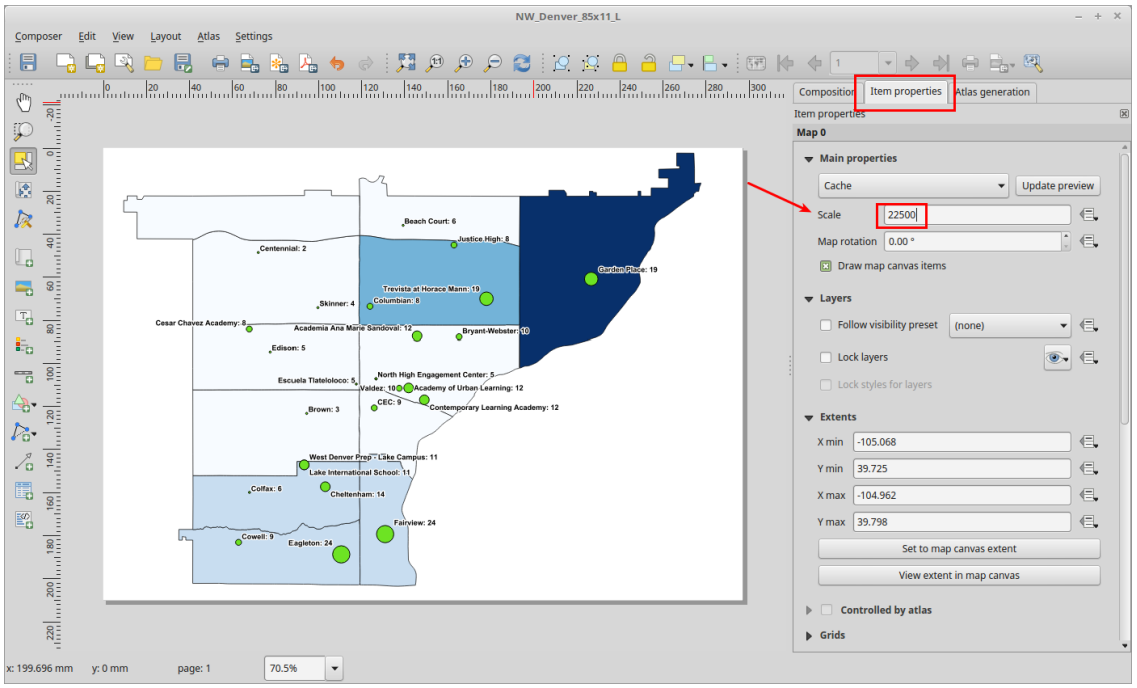
When the map is drawn, you'll see the contents of your QGIS layers panel:



If you need to re-orient the map, use the ***Move item content*** tool to pan the map contents to fit the map window.



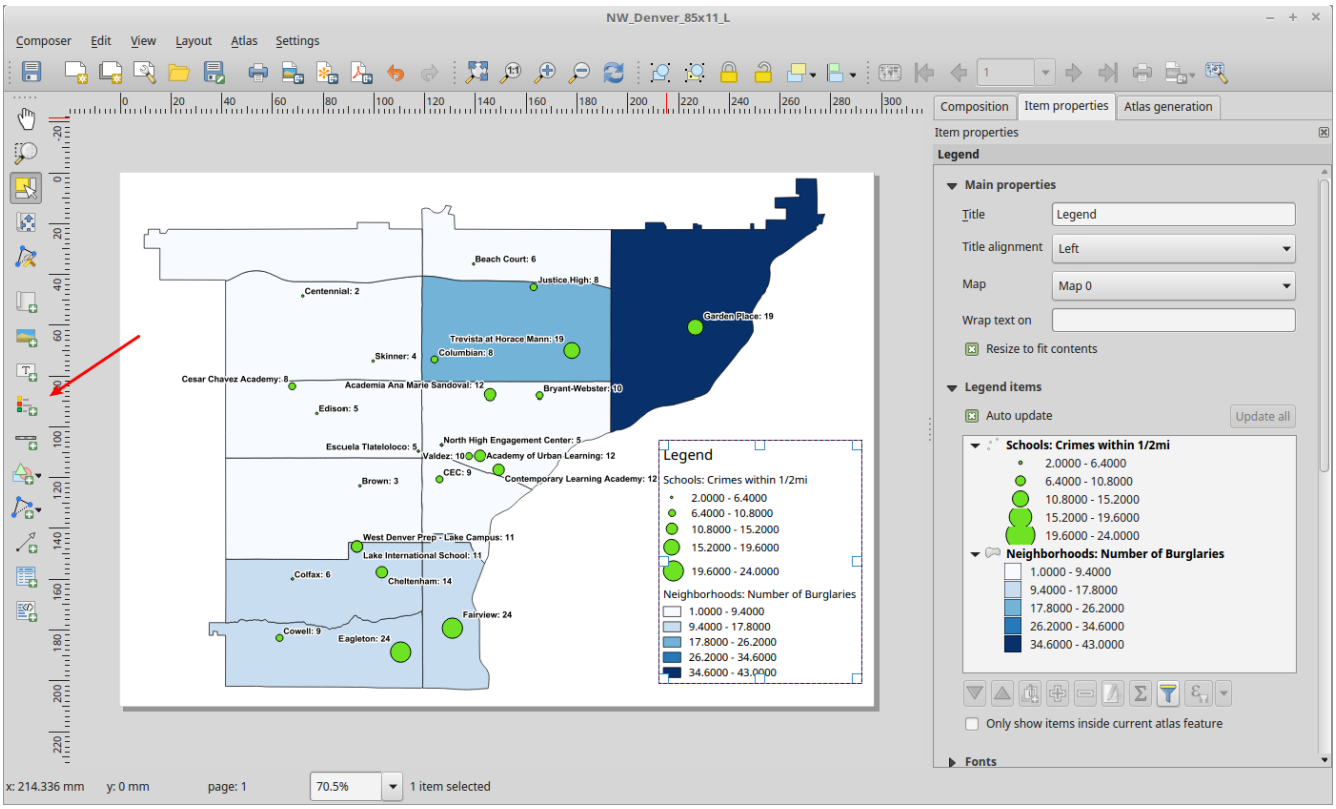
You can also change the ***scale*** in the ***Item Properties*** panel – here set to ***22500*** (or 1:22,500):



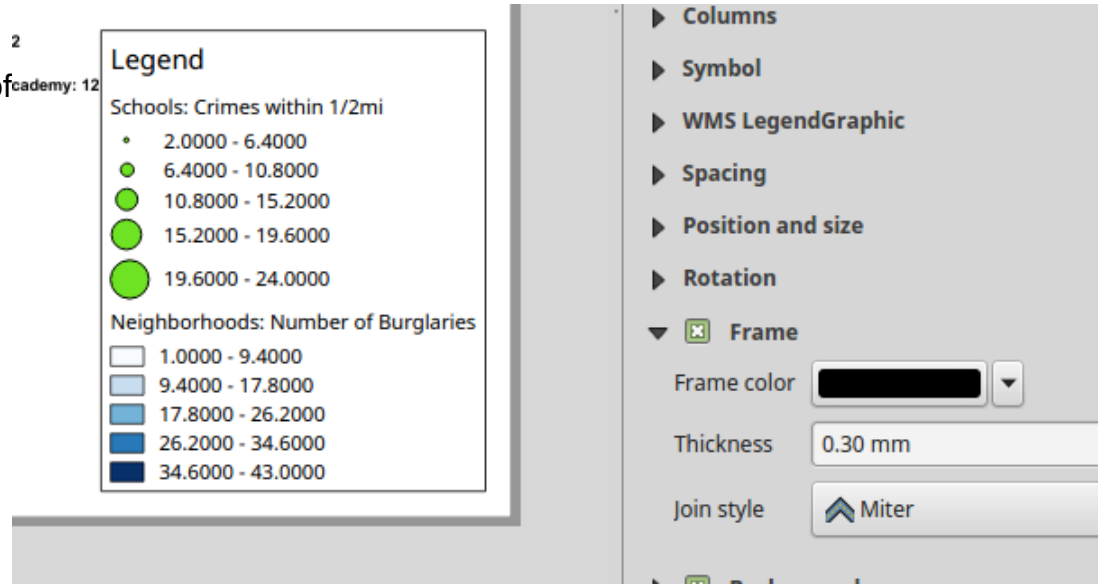
Add a legend to the map

Next, add a legend that describes the layers in your map.

Click the ***Add new legend*** botton from the left-hand side of the print composer window. This adds a legend to the map canvas. Adjust the placement of the legend to sit in the bottom right-hand corner of the map:

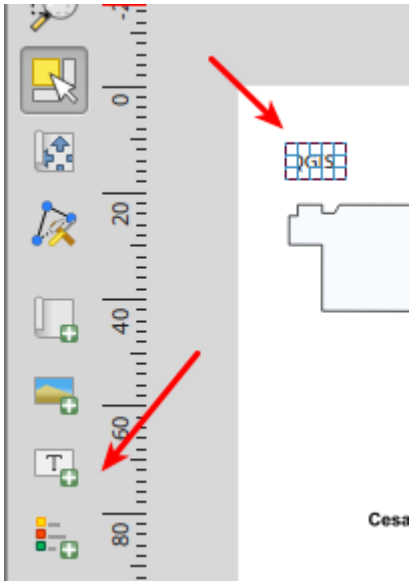


Check the ***Frame*** setting of the legend item to create a line frame around the legend:



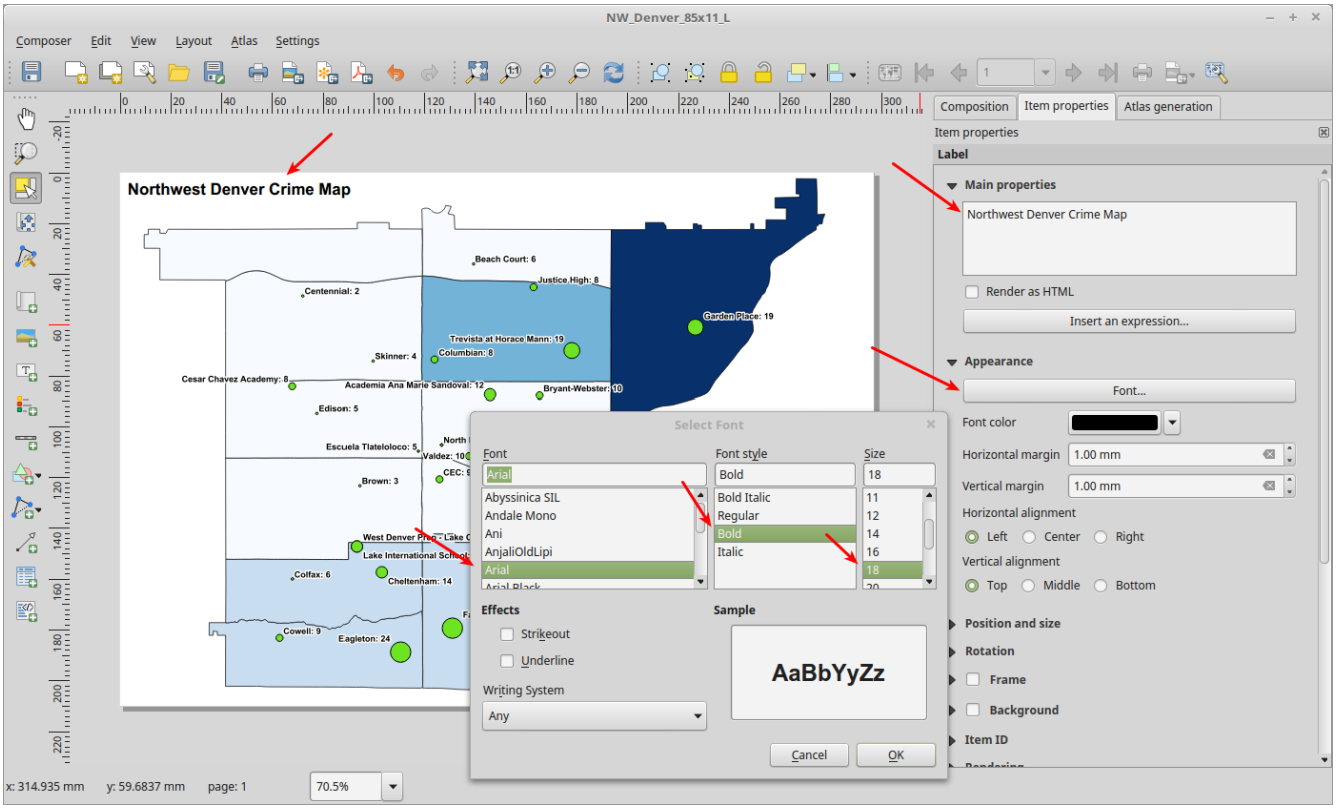
Add a title to the map

From the Print Composer side menu, click the ***Add new text*** button and click the map to add a new text item to the map:

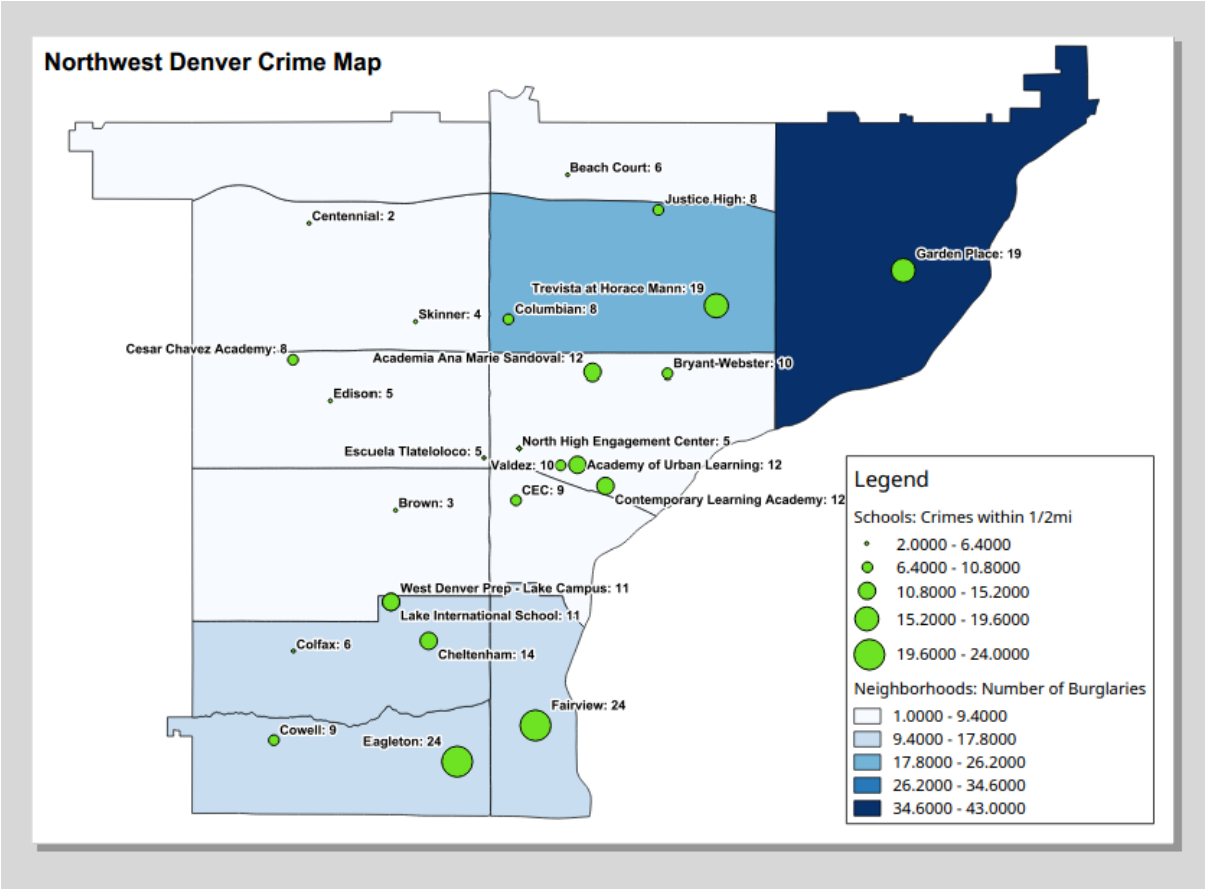


With the new text item selected, use the ***Item Properties*** panel to:

- change the text to read Northwest Denver Crime map
- change the font to Arial Bold 18pt



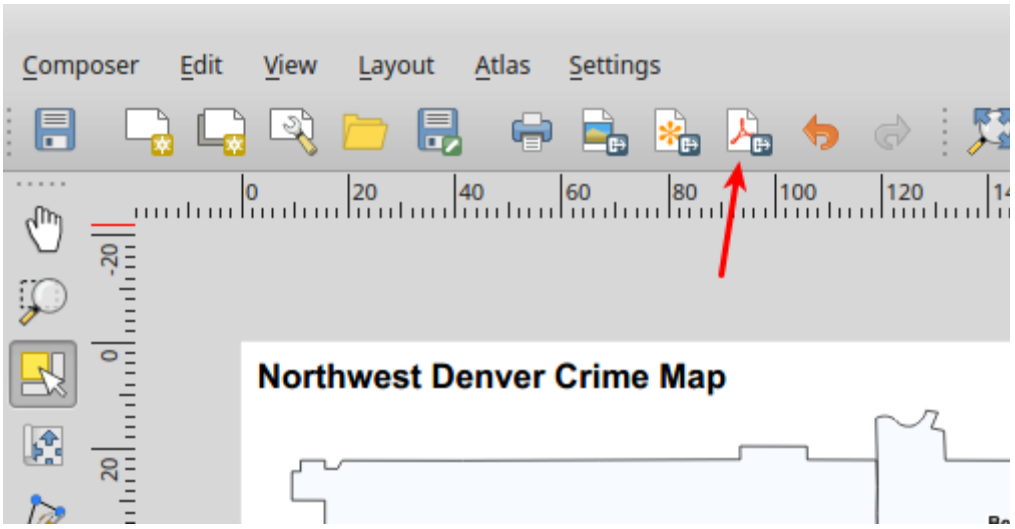
Click OK to see the final map:



Export map to PDF

Once you're satisfied with your map, export the composer to a PDF file.

From the top of the Print Composer window, click the Export as PDF button:



Follow the prompts to save the PDF to a location on disk.