PostGIS :
An Introduction to Data Normalization &
Tutorials on the Spatial Analysis of Parcel Data

:: Leonardo Hughes

As accurate and up to date data becomes increasingly accessible to everyday citizens, tutorials on the aggregation and normalization of datasets can empower you, the user, to implement POSTGIS as a tool in which to analyze these datasets with the added functionality of spatial commands. This tutorial is designed to step the user through the Extraction, Transformation and Loading process of creating ones very own spatial database using data provided by the City of Detroit.

Data

All data gathered within this tutorial has been made publicly available at:
https://data.detroitmi.gov/

Datasets:

- Parcel Map
- Building Demolition
- Zoning
- Neighborhood
- Owner
- Rental

The data compiled for this tutorial has been chosen in order provide the user with commonly formatted data structures released by government organizations. Such datasets are often used within Geographic Information Systems, Community & Economic Development Organizations, Business Analytics as well as a host of other fields.

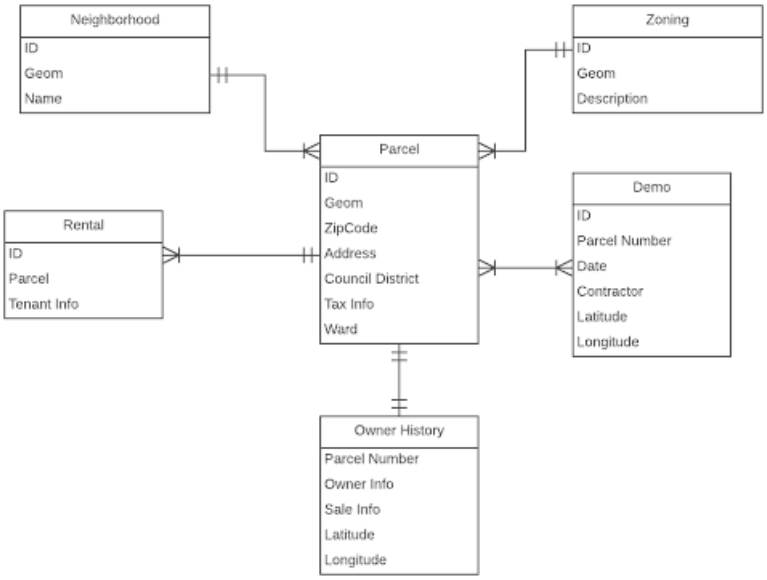
The appendix of the tutorial includes the data normalization script in its entirety.

Data Structure and the Normalization Process:: ERD

In order to create a functional Spatial Database using PostGIS, the process of data normalization can serve as guiding pillars towards optimizing your very own scripts. Normalization is a the hierarchical process of constructing a relational database free of duplicated data, contradictory data, or dependencies which may lead to the loss of data. This process is pivotal in the organizing of data designed to inform strategic planning which relies heavily on up to date and computationally accurate information. Normalization takes 3 forms:

- 1NF - The attribute of a table cannot hold multiple values
- 2NF - No non-prime attribute is dependent on the proper subset of any candidate key of table.
- 3NF - Transitive functional dependency of non-prime attribute on any super key should be removed.

The 3 Normal Forms of a Database can be visually interpreted using an Entity Relationship Diagram, or ERD. Below is the ERD associated with the Spatial Database created in this tutorial.



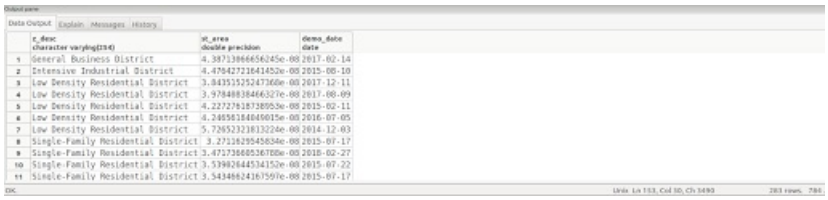
One example of why the relational dependencies across table is so crucial to the functionality of the database can be expressed through the Parcel to Zoning relationship. In our schema, we have set it up so that the zoning designation is dropped from the Parcel table (see Appendix for how to DROP COLUMN). This is done so that if a parcel lies within a region which undergoes a change in zoning law, the data stored in the Parcel table will no longer yield inaccurate data. Rather than zoning designation depend on the Parcel value, zoning classification is applied to the parcels which lay within the geometry, or shape, of the polygon area found in the data.

Getting to know your data is crucial in deciding which information may be superfluous and can be removed. Consider which relational dependencies make sense for the purposes of *your* spatial analysis! Happy scripting!

Analytical Queries!

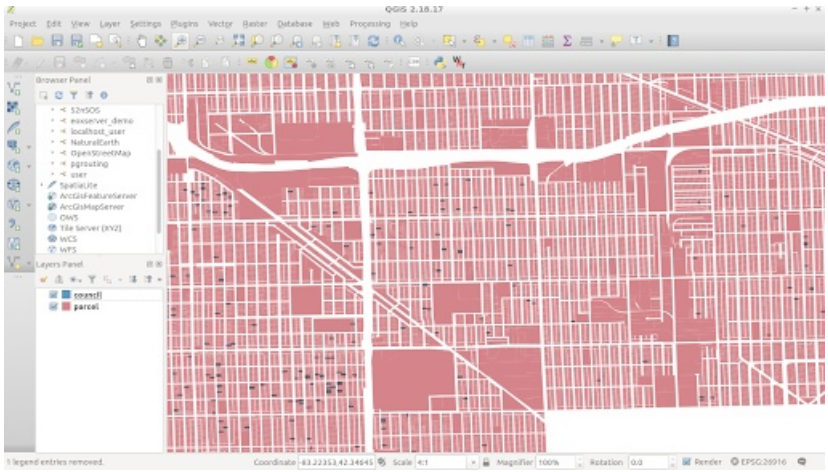
1. What is the acreage of demolished parcels in Detroit?
   Let's widdle this query down into one that may be visually displayed in QGIS. For this purpose, we will focus on parcels located within the 7th Council District. These types of queries may be beneficial in understanding the spatial patterns found in the changing physical landscape of communities in urban settings at the political level. Boundaries in representation do change and we can organize our datasets to reflect these changes and provide accountability by local government.

```
SELECT z.z_desc, ST_Area(p.geom)/4047, d.demo_date
FROM zoning z
JOIN parcel1 p
ON ST_CONTAINS(z.geom, p.geom)
JOIN demo1 d
USING(parcelnum)
WHERE p.council_district = '7'
GROUP BY z.z_desc, ST_AREA(p.geom)/4047, d.demo_date
ORDER BY z.z_desc
;
```



z_desc character varying(254)	st_area double precision	demo_date date
General Business District	4.387138666656245e-08	2017-02-14
Intensive Industrial District	4.47642721641452e-08	2015-08-10
Low Density Residential District	3.84351525247360e-08	2017-12-11
Low Density Residential District	3.97448836466327e-08	2017-08-09
Low Density Residential District	4.22727618738953e-08	2015-02-11
Low Density Residential District	4.24656184046015e-08	2016-07-05
Low Density Residential District	5.72652321813224e-08	2014-12-03
Single-Family Residential District	3.27116295458634e-08	2015-07-17
Single-Family Residential District	3.47173660536780e-08	2018-02-27
Single-Family Residential District	3.53902644534152e-08	2015-07-22
Single-Family Residential District	3.54348624167597e-08	2015-07-17

QGIS

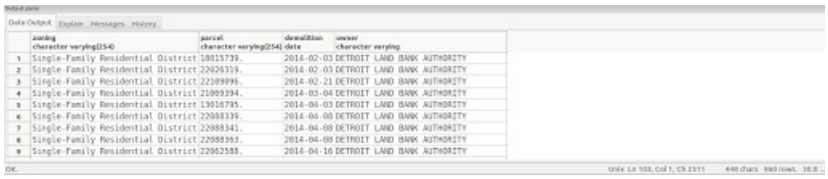


Here we are introduced to several operations commonly seen in SQL, but with added functionality made possible by POSTGRES and spatial data. The query seen above relies on two JOINs in order to link parcel to zoning attributes based on their geometries, and then demolition attributes to parcel numbers based on whether or not they match.
Each table has important information as to the characteristics of land use in District 7. Having organized the data according to our Entity Relation Diagram, we preserve data integrity while allowing for updates/deletions to occur without the breaking of these relationships.

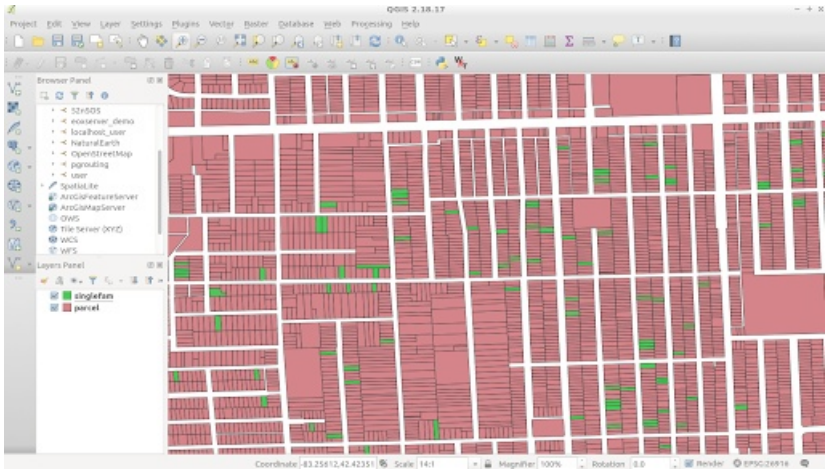
The output contains 283 rows.

2. How many parcels zoned as ‘Single- Family Residential’ were demolished by the City of Detroit since January 1st, 2014? The frequency of parcel demolition in urban settings such as Detroit is a unique characteristic of a City whose physical changes have lead to many forms of revitalization. Although the output may be best displayed as a table due to the scale of parcel to City, the distribution of demolished parcels by zoning districts is a compelling research task for any planner.

```
SELECT l.z_desc, l.parcelnum, d.demo_date
FROM (
 SELECT z.zdesc, p.parcelnum
 FROM zoning as z
 JOIN parcel1 as p
 ON ST_CONTAINS(z.geom, p.geom)
 GROUP BY z.zdesc, p.parcelnum) as l
JOIN demo1 d
ON l.parcelnum = d.parcelnum
WHERE l.z_desc = 'Single-Family Residential District'
GROUP BY l.z_desc, l.parcelnum, d.demo_date
ORDER BY d.demo_date
;
```



QGIS



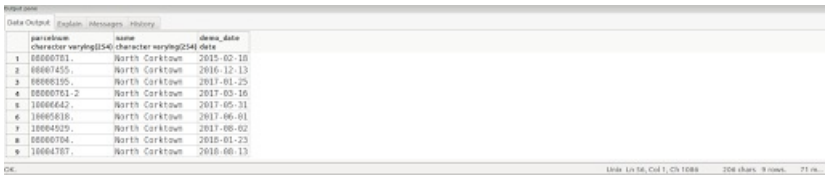
The first is the SUBQUERY, or the lines following the FROM clause: Aliased as `l` , the Parcel and Zoning table are joined based on parcel geometries which lies completely within a zone. A second JOIN is then followed using the aliased table (Parcel and Zoning) with the Building Demolition table, JOINed on matching values of parcelnum, which refers to the parcel number. The WHERE clause is then applied to a double-joined table to restrict output to parcels within the 'Single Family Residential District' which have a demolition date. In essence, parcels are matched to zones based on their geometry and demolitions are matched to their parcel based on their parcel numbers.

The output contains 960 rows.

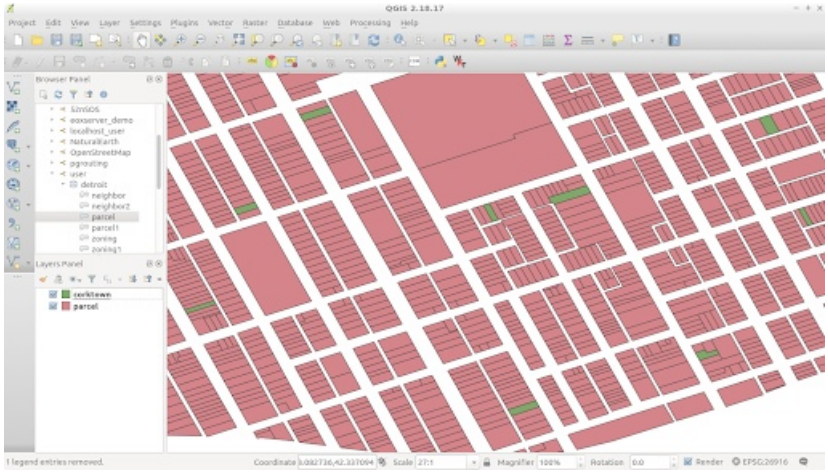
3. Using this database, the user can delve into topics of urban policy and planning in real time. Development within communities is initiated and sustained at the neighborhood level. Therefore, our database includes the geometry of Neighborhoods located in the Detroit in order to perform similar queries as seen in example Query 2. Below is a similar query structure but relies on the MultiPolygon geometry field within the Neighborhood geom table.

Through the use of the ST_CONTAINS function in the JOIN field, the parcels whose geometries are completely within the boundaries of the polygon whose attribute name is similar to "Corktown" will be placed into the output data. Take note that there are no demolitions that have taken place within this neighborhood but do take place in the area recognized as "North Corktown."

```
SELECT b.parcelnum, n.name, b.demo_date
FROM (
 SELECT *
 FROM parcel1 p
 JOIN demo1 d
 USING(parcelnum)) AS b
JOIN neighbor n
ON ST_Contains(n.geom, b.geom)
WHERE n.name ILIKE '%Corktown%'
ORDER BY b.demo_date
;
```



QGIS



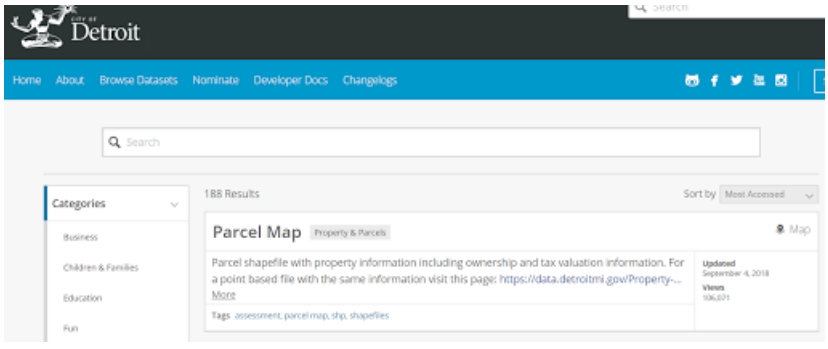
The database includes the geometry of Neighborhoods outlined in Detroit in order to perform similar queries as seen in example Query 2. The Above relies on the MultiPolygon geometry field within the Neighborhood table to restrict the output to parcels contained in a fuzzy match.

The third argument in the SELECT statement provides an additional restriction on parcel data to yield output for parcels whose attribute data contain information within the demolition date field after a JOIN has occurred.

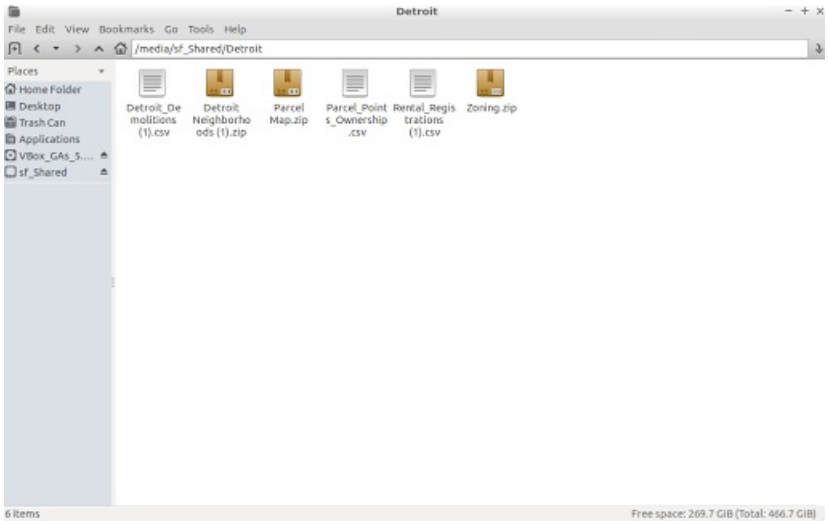
The output contains 9 rows.

Where to find this data?

Information included in this analysis has been gathered at https://data.detroitmi.gov/



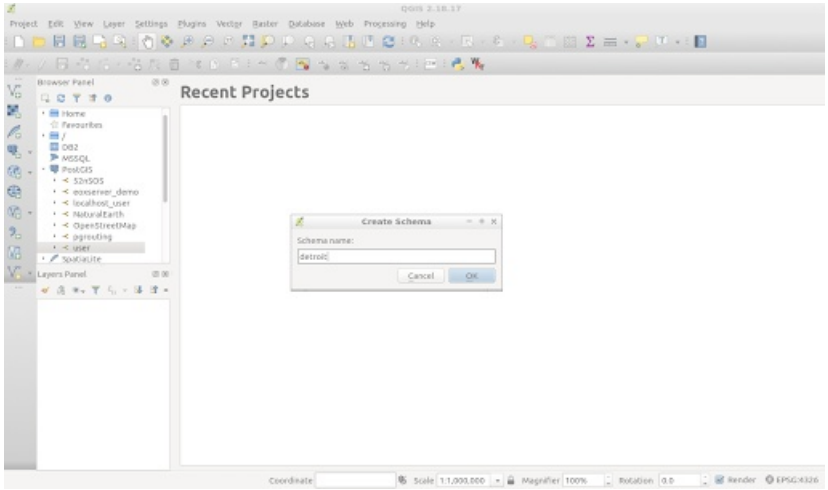
Place all of your data files into the same folder in order to keep track of the original files used in creating your database. You may find this helpful if you accidentally drop a column by mistake and must load the original data back in. Mistakes do happen.



The next step in loading your data will be done through QGIS. Please note that there are several ways of going loading data into PostGIS - such as command line scripting. This tutorial will use QGIS as a companion to PostGIS's spatial functions.

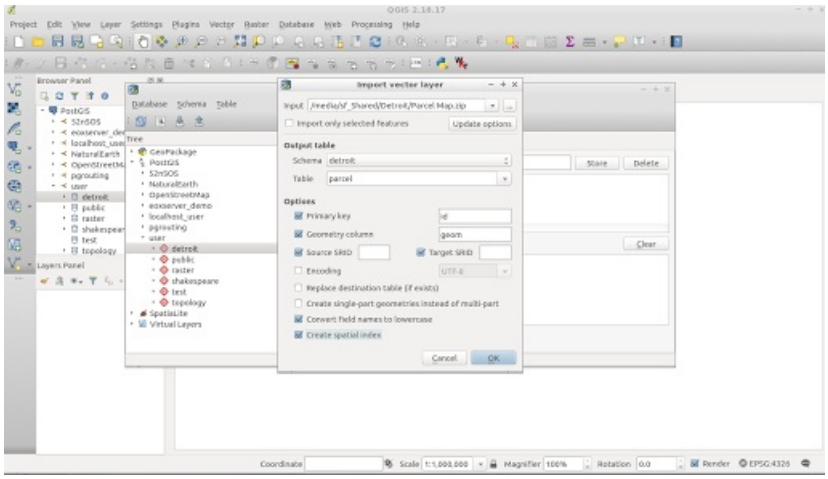
Add POSTGIS Spatial Layer within QGIS, name it "user" if a "user" layer is not already provided.

Right click on the "user" layer. Select "Create Schema" and name your schema!



Now that your schema has been created, we will begin adding data in as Tables. The Parcel Map shapefile will be the first in but order does not matter.

Navigate to the DB Manager within QGIS and select the drop down arrow for the "user" layer. Highlight the schema you have just created and select the "import layer/file" tab (displayed as a downward facing arrow).



Navigate to your resource folder where you placed the original datasets downloaded from https://data.detroitmi.gov/.

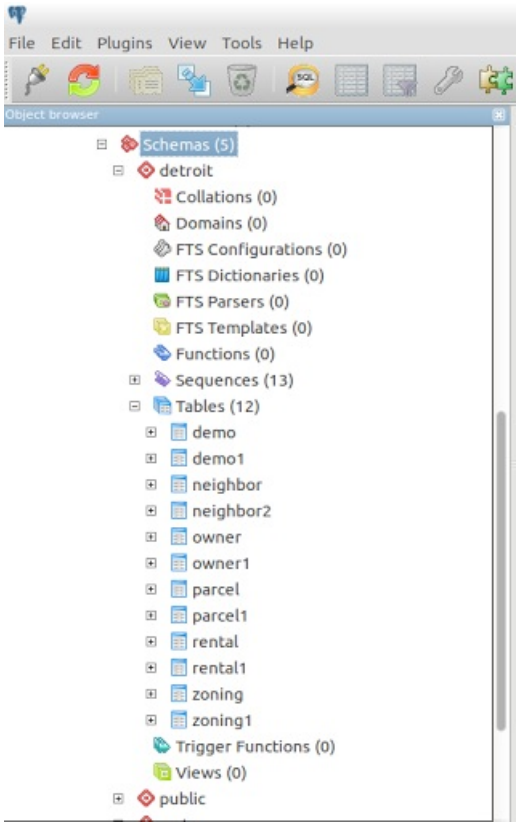
Give the data table an appropriate name- it is common practice to have all table names printed in lowercase.

In the options portion of the the Import Vector Layer tab, be sure to check:

1. Primary Key to the 'id' field
2. Geometry Column to 'geom' field
3. Source/Target SRID fields to the appropriate EPSG Code
4. Convert field names to lower case
5. Create the spatial index

Steps 2 and 3 do not apply to tables without geometry fields or latitude/longitude information and can therefore be skipped.

It is now time to view your tables in your PGAdmin! After the normalization script is ran and depending on how you name your tables, they should look something like …



Appendix

Data Dictionary

- VARCHAR (length)
  character strings of a specified length
- BOOLEAN
  True or False
- INTEGER or INT
  numeric values with an implied scale of zero
- DECIMAL
  numeric values, user defines precision & scale in the data
- NUMERIC
  point base
- DOUBLE PRECISION
  approximate numeric values, up to a precision of 64
- DATE
  YYYY-MM-DD
- SERIAL
  incrementing 4byte integer

Normalization Script

```
set search_path to detroit, public;
show search_path;
```

```
------------------
------Parcel------
------------------


select * from parcel limit 1;

ALTER TABLE parcel RENAME COLUMN "property_c" TO "prop_class";
ALTER TABLE parcel RENAME COLUMN "taxable_va" TO "tax_val";
ALTER TABLE parcel RENAME COLUMN "improved_v" TO "improved_val";
ALTER TABLE parcel RENAME COLUMN "homestead_" TO "homestead";
ALTER TABLE parcel RENAME COLUMN "num_buildi" TO "num_building";
ALTER TABLE parcel RENAME COLUMN "related_pa" TO "related_parcel";
ALTER TABLE parcel RENAME COLUMN "last_sal_1" TO "last_sale_date";
ALTER TABLE parcel RENAME COLUMN "taxable_st" TO "tax_status";
ALTER TABLE parcel RENAME COLUMN "legaldesc" TO "legal_desc";
ALTER TABLE parcel RENAME COLUMN "owner_stre" TO "owner_street";
ALTER TABLE parcel RENAME COLUMN "council_di" TO "council_district";
ALTER TABLE parcel RENAME COLUMN "building_s" TO "building";
ALTER TABLE parcel RENAME COLUMN "land_value" TO "land_val";
ALTER TABLE parcel RENAME COLUMN "owner_stat" TO "owner_state";
ALTER TABLE parcel RENAME COLUMN "last_sale_" TO "last_sale_val";
ALTER TABLE parcel RENAME COLUMN "owner_coun" TO "owner_country";


create table parcel1 (
    id integer PRIMARY KEY,
    geom geometry(MultiPolygon, 26916),
    prop_class varchar(45),
    acreage numeric,
    zip_code varchar(45),
    year_built integer,
    improved_val integer,
    homestead varchar(254),
    owner_zip varchar(45),
    owner2 varchar(254),
    depth varchar(45),
    last_terms varchar(254),
    objectid integer,
    num_building integer,
    related_parcel varchar(254),
    owner_city varchar(254),
    sqft integer,
    last_sale_date varchar(254),
    parcelnum varchar(254),
    tax_status varchar(254),
    owner1 varchar(254),
    legal_desc varchar(254),
    frontage integer,
    landmap varchar(254),
    nez varchar(45),
    taxpayer varchar(254),
    sev integer,
    owner_street varchar(254),
    ward varchar(254),
    council_district varchar(254),
```

```
    building varchar(45),
    land_val integer,
    address varchar(245),
    owner_state varchar(45),
    last_sale_val integer,
    owner_country varchar(254),
    floor_area integer,
    zoning varchar(45)
);

INSERT INTO parcel1(
    id,
    geom,
    prop_class,
    acreage,
    zip_code,
    year_built,
    improved_val,
    homestead,
    owner_zip,
    owner2,
    depth,
    last_terms,
    objectid,
    num_building ,
    related_parcel,
    owner_city,
    sqft,
    last_sale_date,
    parcelnum,
    tax_status,
    owner1,
    legal_desc,
    frontage,
    landmap,
    nez,
    taxpayer,
    sev,
    owner_street,
    ward,
    council_district ,
    building,
    land_val ,
    address,
    owner_state,
    last_sale_val,
    owner_country,
    floor_area,
    zoning)
(SELECT
    id,
    geom,
    prop_class,
    acreage,
    zip_code,
    year_built,
    improved_val,
```

```
    homestead,
    owner_zip,
    owner2,
    depth,
    last_terms,
    objectid,
    num_building ,
    related_parcel,
    owner_city,
    sqft,
    last_sale_date,
    parcelnum,
    tax_status,
    owner1,
    legal_desc,
    frontage,
    landmap,
    nez,
    taxpayer,
    sev,
    owner_street,
    ward,
    council_district ,
    building,
    land_val ,
    address,
    owner_state,
    last_sale_val,
    owner_country,
    floor_area,
    zoning
FROM parcel
);



ALTER TABLE parcel1 DROP COLUMN "owner_zip";
ALTER TABLE parcel1 DROP COLUMN "owner2";
ALTER TABLE parcel1 DROP COLUMN "owner_city";
ALTER TABLE parcel1 DROP COLUMN "owner1";
ALTER TABLE parcel1 DROP COLUMN "owner_street";
ALTER TABLE parcel1 DROP COLUMN "owner_state";
ALTER TABLE parcel1 DROP COLUMN "owner_country";
ALTER TABLE parcel1 DROP COLUMN "floor_area";
-- ALTER TABLE parcel1 DROP COLUMN "zoning";



ALTER TABLE parcel1 ALTER COLUMN zip_code
 TYPE NUMERIC USING zip_code::numeric;
ALTER TABLE parcel1 ALTER COLUMN last_sale_date
 TYPE DATE USING last_sale_date::date;


select * from parcel1 limit 1;
```

```
---------------------------
--- Demolition -----------
---------------------------

ALTER TABLE demo RENAME COLUMN "parcel id" TO "parcelnum";
ALTER TABLE demo RENAME COLUMN "contractor name" TO "contractor";
ALTER TABLE demo RENAME COLUMN "primary funding source" TO "funder";
ALTER TABLE demo RENAME COLUMN "demolition date" TO "demo_date";
ALTER TABLE demo RENAME COLUMN "commercial building" TO "commercial";



create table demo1(
    id serial PRIMARY KEY,
    address varchar(254),
    parcelnum varchar(254),
    contractor varchar(254),
    price varchar(254),
    funder varchar(254),
    demo_date varchar(254),
    commercial varchar(254),
    council_district varchar(254),
    neighborhood varchar(254),
    latitude varchar,
    longitude varchar,
    location varchar(254)
);

INSERT INTO demo1(
    id,
    address,
    parcelnum,
    contractor,
    price,
    funder,
    demo_date,
    commercial,
    council_district,
    neighborhood,
    latitude,
    longitude,
    location)
(SELECT
    id,
    address,
    parcelnum,
    contractor,
    price,
    funder,
    demo_date,
    commercial,
    council_district,
    neighborhood,
    latitude,
    longitude,
    location
FROM demo
```

```
);


select * from demo1 limit 1;

ALTER TABLE demo1 ALTER COLUMN price
 TYPE DECIMAL USING price::decimal;
ALTER TABLE demo1 ALTER COLUMN demo_date
 TYPE DATE USING demo_date::date;
ALTER TABLE demo1 ALTER COLUMN latitude
 TYPE DOUBLE PRECISION USING latitude::double precision;
ALTER TABLE demo1 ALTER COLUMN longitude
 TYPE DOUBLE PRECISION USING longitude::double precision;


---------------------------
-----Neighborhood----------
---------------------------


select * from neighbor limit 1;

ALTER TABLE neighbor RENAME COLUMN "district_n" TO "district";
ALTER TABLE neighbor RENAME COLUMN "neighborho" TO "neighb";
ALTER TABLE neighbor RENAME COLUMN "new_nhood" TO "new_name";
ALTER TABLE neighbor RENAME COLUMN "nhood_name" TO "name";
ALTER TABLE neighbor RENAME COLUMN "nhood_num" TO "number";
ALTER TABLE neighbor RENAME COLUMN "shape_area" TO "area";
ALTER TABLE neighbor RENAME COLUMN "shape_le_1" TO "length_1";
ALTER TABLE neighbor RENAME COLUMN "target_fid" TO "fid";


create table neighbor2 (
    id serial PRIMARY KEY,
    geom geometry(MultiPolygon, 26916),
    acres numeric,
    district numeric,
    join_count numeric,
    neighb numeric,
    new_name varchar,
    name varchar,
    number numeric,
    objectid numeric,
    area numeric,
    length_1 numeric,
    fid numeric);

INSERT INTO neighbor2 (
    id,
    geom,
    acres,
    district,
    join_count,
    neighb,
    new_name,
    name,
    number,
```

```
    objectid,
    area,
    length_1,
    fid)
(SELECT
    id,
    geom,
    acres,
    district,
    join_count,
    neighb,
    new_name,
    name,
    number,
    objectid,
    area,
    length_1,
    fid
FROM neighbor
);



select * from neighbor2 limit 1;


----------------------------
------ Owner History-------
----------------------------

select * from owner limit 1;



ALTER TABLE owner RENAME COLUMN "parcel number" TO "parcelnum";
ALTER TABLE owner RENAME COLUMN "zip code" TO "prop_zip_code";
ALTER TABLE owner RENAME COLUMN "council district" TO "council_district";
ALTER TABLE owner RENAME COLUMN "taxable status" TO "taxable_status";
ALTER TABLE owner RENAME COLUMN "owner 2" TO "owner_2";
ALTER TABLE owner RENAME COLUMN "owner street address" TO "owner_address";
ALTER TABLE owner RENAME COLUMN "owner city" TO "city";
ALTER TABLE owner RENAME COLUMN "owner state" TO "state";
ALTER TABLE owner RENAME COLUMN "owner zip code" TO "zip_code";
ALTER TABLE owner RENAME COLUMN "owner country" TO "country";
ALTER TABLE owner RENAME COLUMN "last sale price" TO "last_sale_price";
ALTER TABLE owner RENAME COLUMN "last terms of sale" TO "last_terms_of_sale";
ALTER TABLE owner RENAME COLUMN "land value" TO "land_val";
ALTER TABLE owner RENAME COLUMN "taxable value" TO "tax_val";
ALTER TABLE owner RENAME COLUMN "improved value" TO "improved_val";
ALTER TABLE owner RENAME COLUMN "state equalized value" TO "state_equal_val";
ALTER TABLE owner RENAME COLUMN "property class" TO "prop_class";
ALTER TABLE owner RENAME COLUMN "total acreage" TO "total_acreage";
ALTER TABLE owner RENAME COLUMN "principal residence exemption"
 TO "principal_residence_exemption";
ALTER TABLE owner RENAME COLUMN "floor area" TO "floor_area";
ALTER TABLE owner RENAME COLUMN "year built" TO "year_built";
ALTER TABLE owner RENAME COLUMN "building style" TO "building_style";
```

```
ALTER TABLE owner RENAME COLUMN "building style" TO "building_style";
ALTER TABLE owner RENAME COLUMN "number of buildings" TO "num_of_buildings";
ALTER TABLE owner RENAME COLUMN "related parcel" TO "related_parcel";
ALTER TABLE owner RENAME COLUMN "legal description" TO "legal_desc";

create table owner1(
    id serial PRIMARY KEY,
    objectid varchar,
    parcelnum varchar,
    address varchar,
    prop_zip_code varchar,
    council_district varchar,
    taxable_status varchar,
    owner varchar,
    owner_2 varchar,
    taxpayer varchar,
    owner_address varchar,
    city varchar,
    state varchar,
    zip_code varchar,
    zoning varchar,
    country varchar,
    saledate varchar,
    last_sale_price varchar,
    last_terms_of_sale varchar,
    land_val varchar,
    improved_val varchar,
    state_equal_val varchar,
    prop_class varchar,
    total_acreage varchar,
    frontage varchar,
    principal_residence_exemption varchar,
    nez varchar,
    depth varchar,
    sqft varchar,
    floor_area varchar,
    year_built varchar,
    building_style varchar,
    num_of_buildings varchar,
    ward varchar,
    landmap varchar,
    related_parcel varchar,
    longitude varchar,
    latitude varchar,
    legal_desc varchar,
    location varchar

);

INSERT INTO owner1(
    id,
    objectid,
    parcelnum,
    address,
    prop_zip_code,
    council_district,
    taxable_status,
    owner
```

```
    owner,
    owner_2,
    taxpayer,
    owner_address,
    city,
    state,
    zip_code,
    zoning,
    country,
    saledate,
    last_sale_price,
    last_terms_of_sale,
    land_val,
    improved_val,
    state_equal_val,
    prop_class,
    total_acreage,
    frontage,
    principal_residence_exemption,
    nez,
    depth,
    sqft,
    floor_area,
    year_built,
    building_style,
    num_of_buildings,
    ward,
    landmap,
    related_parcel,
    longitude,
    latitude,
    legal_desc,
    location)
(SELECT
    id,
    objectid,
    parcelnum,
    address,
    prop_zip_code,
    council_district,
    taxable_status,
    owner,
    owner_2,
    taxpayer,
    owner_address,
    city,
    state,
    zip_code,
    zoning,
    country,
    saledate,
    last_sale_price,
    last_terms_of_sale,
    land_val,
    improved_val,
    state_equal_val,
    prop_class,
```

```
    total_acreage,
    frontage,
    principal_residence_exemption,
    nez,
    depth,
    sqft,
    floor_area,
    year_built,
    building_style,
    num_of_buildings,
    ward,
    landmap,
    related_parcel,
    longitude,
    latitude,
    legal_desc,
    location
FROM owner
);


select * from owner1 limit 1;


ALTER TABLE owner1 ALTER COLUMN prop_zip_code
 TYPE NUMERIC USING prop_zip_code::numeric;
ALTER TABLE owner1 ALTER COLUMN zip_code
 TYPE DECIMAL USING zip_code::decimal;
ALTER TABLE owner1 ALTER COLUMN saledate
 TYPE DATE USING saledate::date;
ALTER TABLE owner1 ALTER COLUMN last_sale_price
 TYPE DECIMAL USING last_sale_price::decimal;
ALTER TABLE owner1 ALTER COLUMN land_val
 TYPE DECIMAL USING land_val::decimal;
ALTER TABLE owner1 ALTER COLUMN improved_val
 TYPE DECIMAL USING improved_val::decimal;
ALTER TABLE owner1 ALTER COLUMN state_equal_val
 TYPE DECIMAL USING state_equal_val::decimal;
ALTER TABLE owner1 ALTER COLUMN total_acreage
 TYPE DECIMAL USING total_acreage::decimal;
ALTER TABLE owner1 ALTER COLUMN frontage
 TYPE DECIMAL USING frontage::decimal;
ALTER TABLE owner1 ALTER COLUMN total_acreage
 TYPE DECIMAL USING total_acreage::decimal;
ALTER TABLE owner1 ALTER COLUMN principal_residence_exemption
 TYPE DECIMAL USING principal_residence_exemption::decimal;
ALTER TABLE owner1 ALTER COLUMN depth
 TYPE DECIMAL USING depth::decimal;
ALTER TABLE owner1 ALTER COLUMN sqft
 TYPE DECIMAL USING sqft::decimal;
ALTER TABLE owner1 ALTER COLUMN floor_area
 TYPE DECIMAL USING floor_area::decimal;
ALTER TABLE owner1 ALTER COLUMN year_built
 TYPE INTEGER USING year_built::integer;
ALTER TABLE owner1 ALTER COLUMN num_of_buildings
 TYPE DECIMAL USING num_of_buildings::decimal;
ALTER TABLE owner1 ALTER COLUMN longitude
```

```
 TYPE DOUBLE PRECISION USING longitude::double precision;
ALTER TABLE owner1 ALTER COLUMN latitude
 TYPE DOUBLE PRECISION USING latitude::double precision;


select * from owner1 limit 1;


-----------------------------
----- Rental-----------------
-----------------------------

select * from rental limit 1;

ALTER TABLE rental RENAME COLUMN "case number" TO "casenum";
ALTER TABLE rental RENAME COLUMN "parcel number" TO "parcelnum";
ALTER TABLE rental RENAME COLUMN "first name" TO "first_name";
ALTER TABLE rental RENAME COLUMN "last name" TO "last_name";
ALTER TABLE rental RENAME COLUMN "middle initial" TO "middle_initial";
ALTER TABLE rental RENAME COLUMN "received by" TO "received_by";
ALTER TABLE rental RENAME COLUMN "received date" TO "received_date";
ALTER TABLE rental RENAME COLUMN "csa creation date" TO "csa_creation_date";
ALTER TABLE rental RENAME COLUMN "action description" TO "action_description";
ALTER TABLE rental RENAME COLUMN "date issued" TO "date_issued";
ALTER TABLE rental RENAME COLUMN "case type" TO "case_type";
ALTER TABLE rental RENAME COLUMN "zip code" TO "zip_code";
ALTER TABLE rental RENAME COLUMN "council district" TO "council_district";


create table rental1(
    id serial PRIMARY KEY,
    casenum varchar(254),
    parcelnum varchar(254),
    address varchar(254),
    use varchar(254),
    units varchar(254),
    first_name varchar(254),
    last_name varchar(254),
    middle_initial varchar(45),
    received_by varchar(254),
    received_date varchar(254),
    csa_creation_date varchar(254),
    action_description varchar(254),
    date_issued varchar(254),
    disposition varchar(254),
    status varchar(254),
    case_type varchar(254),
    zip_code varchar(254),
    council_district varchar(254),
    location varchar(254)
);

INSERT INTO rental1(
    id,
    casenum,
    parcelnum,
    address,
```

```
    use,
    units,
    first_name,
    last_name,
    middle_initial,
    received_by,
    received_date,
    csa_creation_date,
    action_description,
    date_issued,
    disposition,
    status,
    case_type,
    zip_code,
    council_district,
    location)
(SELECT
    id,
    casenum,
    parcelnum,
    address,
    use,
    units,
    first_name,
    last_name,
    middle_initial,
    received_by,
    received_date,
    csa_creation_date,
    action_description,
    date_issued,
    disposition,
    status,
    case_type,
    zip_code,
    council_district,
    location
FROM rental
);

select * from rental1 limit 1;

ALTER TABLE rental1 ALTER COLUMN received_date
 TYPE DATE USING received_date::date;
ALTER TABLE rental1 ALTER COLUMN csa_creation_date
 TYPE DATE USING csa_creation_date::date;
ALTER TABLE rental1 ALTER COLUMN date_issued
 TYPE DATE USING date_issued::date;

ALTER TABLE rental1 DROP COLUMN "address";
ALTER TABLE rental1 DROP COLUMN "zip_code";
ALTER TABLE rental1 DROP COLUMN "council_district";



----------------------------
----- Zoning ---------------
----------------------------
```

```sql
select * from zoning limit 1;

ALTER TABLE zoning RENAME COLUMN "shape_leng" TO "shape_length";
ALTER TABLE zoning RENAME COLUMN "zcode_n" TO "zcode";
ALTER TABLE zoning RENAME COLUMN "zdescr_n" TO "z_desc";


create table zoning1 (
    id serial PRIMARY KEY,
    geom geometry(MultiPolygon, 26916),
    objectid integer,
    shape_area integer,
    shape_length integer,
    zcode integer,
    z_desc varchar(254),
    zoning_rev varchar(254)
);

INSERT INTO zoning1(
    id,
    geom,
    objectid,
    shape_area,
    shape_length,
    zcode,
    z_desc,
    zoning_rev)
(SELECT
    id,
    geom,
    objectid,
    shape_area,
    shape_length,
    zcode,
    z_desc,
    zoning_rev
FROM zoning
);
```