The Real Estate Project

March 4th, 20223

By Craig Schlachter

Step 1: Ask

In this step we define the problem and objectives of our case study and its desired outcome

1.1 Background

The Schlachters are making a big move in 2023. Craig and his wife, Jihyeon, are moving from South Korea to Georgia, USA. After using data analysis for over a year, Craig wants to use his SQL and visualization skills to understand the local housing market better and run linear regression models to predict housing prices.

Craig is confident that an analysis of credible housing data will provide useful insights for budgeting and scouting potential cities for his family to settle in.

1.2 Business Task

Analyze a comprehensive housing dataset to gain insights into how variables affect housing prices and **discover trends** of the Georgia housing market.

1.3 Business Objectives

- What trends have been identified?
- How do different variables affect home prices?
- How can these trends and the variables' differing price effects help the Schlachters form their house-hunting strategy?

1.4 Deliverables

- A clear summary of the business task
- A description of all data sources used
- Documentation of any data cleaning and manipulation
- A summary of the analysis
- Supporting visualizations and key findings
- Useful recommendations based on the analysis

1.5 Key Stakeholders

- Craig Schlachter, the husband of the family moving to Georgia.
- Jihyeon Kim, the wife of the family moving to Georgia.

. . .

Step 2: Prepare

In the prepare phase, we identify the data being used and its limitations.

2.1 Information on Data Source

1. Data is publicly available on Kaggle: Real Estate Georgia and stored in 1 csv file containing 39 columns.
2. This dataset was co-produced by Guenter Roehrich and Jordan, who produced a dataset of real estate listings for Georgia for the first 6 months of 2021.
3. Data collected includes address, price, living area, city, county, house description, hometype, bedrooms, bathrooms, year built and much more.

2.2 Limitations of Data Set

- This dataset is not highly cited or used widely in other people's projects from what we can see on Kaggle.
- This dataset only provides the first 6 months of data of 2021, for one state, Georgia.
- As data was collected through web-scraping, we are unable to know its integrity or exact accuracy due to no website source being listed.

2.3 Is Data ROCCC?

A good data source is ROCCC which stands for Reliable, Original, Comprehensive, Current, and Cited.

- Reliable - HIGH - Dataset has 13,800 records for 39 different fields
- Original - LOW - Third party provider (Kaggle, Jordan & Geunter web-scrapped)
- Comprehensive - HIGH - 39 different parameters match need of business task
- Current - MED - Data is under two years old, but may not be as relevant
- Cited - LOW - Data collected by third party who didn't give their source

Overall, this dataset is considered medium quality and is recommended only as rough guideline for the Schlachters to use for budgeting and house-hunting.

2.4 Data Selection

The following file is downloaded and then imported into our created SQL table ‘g_housing’.

2.5 Tools

We are using SQL for data-wrangling and exploratory data analysis. Finally, we are using Tableau for visualizations.

. . .

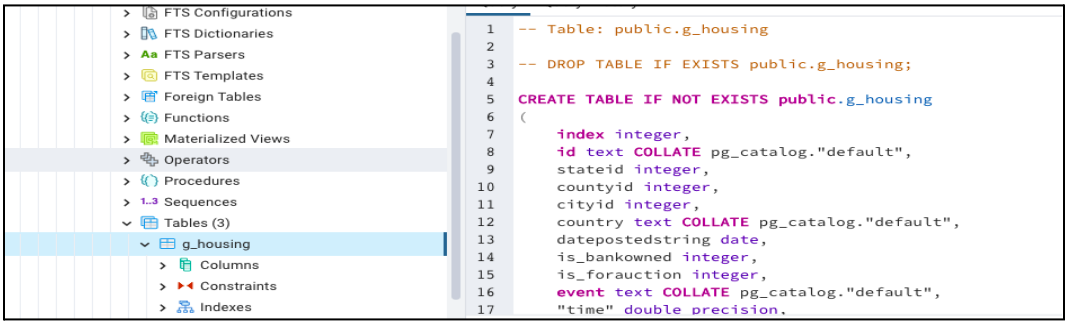
Step 3: Process

Here, we will process the data to ensure it is clean, correct, relevant, complete and free of errors and outliers by performing:

- Explore and observe our dataset
- Check for and treat missing or null values
- Transform data - format data type
- Validate the data is processed with preliminary statistical analysis.

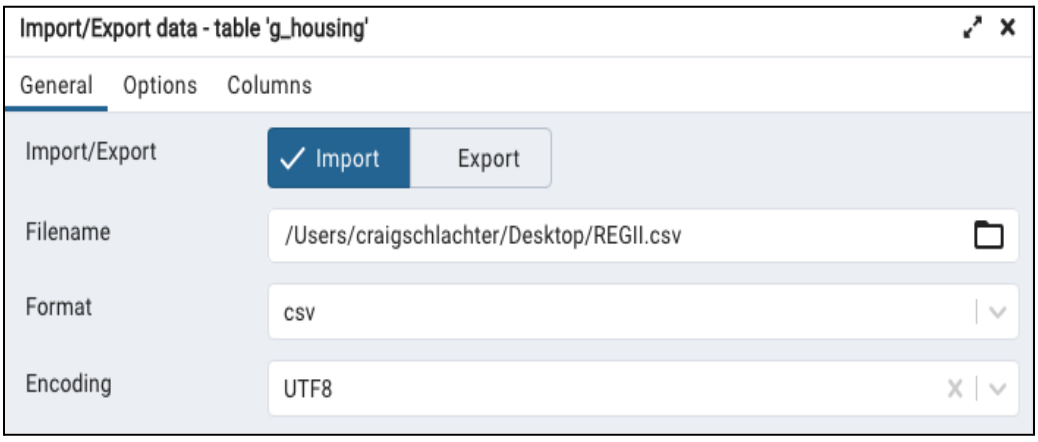
3.1 Preparing the environment

The SQL table is created, columns are named, and data types are set.



3.2 Importing Dataset

Reading in the selected file.



3.2 Data Cleaning and Manipulation

1. Observe and familiarize with data
2. Check for nulls or missing values
3. Perform validation checks of data

Previewing the first 100 rows to familiarize with the data.

```sql
SELECT * FROM public.g_housing
LIMIT 100
```

Data Output | Messages | Notifications

	index integer	id text	stateid integer	countyid integer	cityid integer	country text	datepostedstring date	is_bankowned integer	is_forauction integer	event text	time double precision	price double precision	pricepers double pr
1	9374	30002-14475947	16	312677	0	USA	2021-07-09	0	0	Listed for sale	1625788800000	592500	
2	9375	30002-14481404	16	312678	14956	USA	2021-07-08	0	0	Listed for sale	1625702400000	499900	
3	9378	30002-97971641	16	312681	14956	USA	2021-06-29	0	0	Listed for sale	1624924800000	614900	
4	9383	30002-14508955	16	312686	14956	USA	2021-06-03	0	0	Listed for sale	1622678400000	648900	
5	9384	30002-14508960	16	312687	14956	USA	2021-06-03	0	0	Listed for sale	1625529600000	614900	
6	9385	30002-14509095	16	312688	14956	USA	2021-05-14	0	0	Price change	1625184000000	384900	
7	9376	30002-14481330	16	312679	14956	USA	2021-07-02	0	0	Listed for sale	1625184000000	350000	177.147
8	9386	30002-14475917	16	312689	14956	USA	2021-04-15	0	0	Price change	1625529600000	549000	277.86
9	9387	30002-2078041517	16	312690	14956	USA	2021-03-20	0	0	Price change	1624060800000	169000	85.537
10	9388	30002-2088745111	16	312691	14956	USA	2021-01-30	0	0	Price change	1624147200000	179500	90.851

Then I'm going to find out if there are any nulls or missing values in the dataset.

```sql
/* pricepersquarefoot has 5432 missing values. Updating by imputing livingarea averages first, then updating
by imputing pricepersquarfoot derived from price & livingarea. */
SELECT pricepersquarefoot
FROM g_housing
WHERE pricepersquarefoot = 0
OR pricepersquarefoot IS NULL
;


UPDATE g_housing
SET livingarea = avg_liv_area.avg_living_area
FROM(
    SELECT city, AVG(livingarea) AS avg_living_area
    FROM g_housing
    WHERE livingarea <> 0
    GROUP BY city
    ) AS avg_liv_area
WHERE g_housing.city = avg_liv_area.city
AND livingarea = 0


/* pricepersquarefoot has been imputed. Update again because i need to calculate living area based on city
and hometype for more accurate results.*/

UPDATE g_housing
SET pricepersquarefoot = price/livingarea
WHERE pricepersquarefoot = 0
;
```

Imputing the **5,432 missing values** of the **pricepersquarefoot** column by taking the **average livingarea** for each unique city and then **updating the table** to set pricepersquarefoot = to price divided by livingarea for the 5,432 missing values.

```sql
--5183 values of 0
SELECT livingareavalue
FROM g_housing
WHERE livingareavalue IS NULL
OR livingareavalue <= 0;

ALTER TABLE g_housing           -- Removed whole column because it's a copy of livingarea
DROP COLUMN livingareavalue;
```

5,183 missing values for the **livingareavalue** were found. After comparing it to **livingarea** column, they proved identical. We **dropped livingareavalue** to avoid future confusion.

```sql
-- 7138 values of 0. livingarea and building area are copies of each other.
SELECT livingarea, buildingarea, livingarea - buildingarea
FROM g_housing
WHERE buildingarea <> 0
AND livingarea - buildingarea > 0;

ALTER TABLE g_housing           --Dropped the buildingarea column
DROP COLUMN buildingarea;
```

7,138 missing values for the **buildingarea** were found. After comparing it to **livingarea** column, they proved identical. We **dropped buildingarea** to avoid future confusion.

Above were a summation of the significant missing values we found and how we remedied them. This isn't an exhaustive list. Please read the SQL transcript for complete details.

Next, I'm going to handle the duplicates by **identifying and removing duplicate** records.

```sql
-- Have 1250 duplicates that I need to remove. Use MIN function on index to single out lowest value duplicates.
SELECT id, COUNT(*)
FROM g_housing
GROUP BY id
HAVING COUNT(*) > 1;


DELETE FROM g_housing           -- The 1250 duplicate ids have been removed
WHERE index NOT IN (
                    SELECT MIN(index)
                    FROM g_housing
                    GROUP BY id)

```

We discovered **1,250 duplicate ids**. To remove them, we used the **MIN() function** to find the lowest index value among the ids. Then we deleted index numbers that weren't minimum values.

Next, we are going to look for **outliers** and correct these.

```sql
/* Have a min. price of 10 then ascending prices of 795,895,975,1000, and etc. 177 values lower than 10,000.
May have to filter these low values so they don't skew my data.*/

SELECT price
FROM g_housing
ORDER BY price ASC;

-- The low value of 0.005 pricepersqft are tied to the low values above in price.
SELECT MAX(pricepersquarefoot), MIN(pricepersquarefoot)
FROM g_housing;
```

From the above query we observed that the price column has some **extremely low values** for house prices. We made a note of these values where in the future we **filter these out with the N_TILE function**.

Next, we found a number of the same city names being used twice. With further analysis we **uncovered inconsistent case usage** for the city inputs.(e.g. All uppercase/lowercase)

```
/* Some cities are UPPER CASE. Cross check for duplicates of UPPER and normal case.(Avondale Est,Avondale Estates,
Ball ground, Ball Ground, Blue ridge, Blue Ridge, Trenton, Trenton, JESUP, Jesup, LAGRANGE, Lagrange, JOHNS CREEK,
Johns Creek, Mc Caysville, McCaysville, Mc Rae, Mc Rae Helena, McRae Helena, Merritt St, Hawkinsville, Norman park,
Norman Park, Pt Wentworth, Port Wentworth, Rock spring, Rock Spring, Rocky face, Rocky Face, Sandy springs,
Sandy Springs, Talking rock, Talking Rock, Tunnel hill, Tunnel Hill, Union Pt, Union Point, Warner robins,
Warner Robins, West Pt, West Point) */

SELECT city, COUNT(*)
FROM g_housing
GROUP BY city
ORDER BY city ;

-- Updated the cities above and made them uniform.
UPDATE g_housing
SET city = REPLACE(city, 'West Pt', 'West Point')
WHERE city LIKE '%West Pt%';
```

By running the above query and **ordering it alphabetically**, we're able to know all the unique city names and check for duplicates written in all uppercase or lowercase. After completing a list of the duplicates, we **updated the table to replace** the affected cities with camel case style.

When looking for outliers, we stumbled upon **another set of duplicates** in the **streetaddress column** that needed to be handled.

```
-- DELETE  id '31401-317206349', '31401-317206359','30363-82610574', streetaddress were exact same.
SELECT streetaddress, COUNT(*), hometype, city, price
FROM g_housing
GROUP BY streetaddress, hometype, city, price
ORDER BY COUNT(*) DESC;

-- DELETING 3 Duplicate streetaddresses that were the same but slightly different id
DELETE FROM g_housing
WHERE id IN('30510-2076566105','31401-317206349', '31401-317206359','30363-82610574')
RETURNING *;
```

We ran a simple **DELETE statement** to delete all rows in the list of ids we gave it. The duplicates have been successfully removed.

This **concludes** our task of detecting and correcting outliers.

Moving on, we are combing through the dataset **looking for inconsistencies and inaccurate data**.

We found **instances of inconsistent data** for the **bedrooms and bathrooms columns**. Using **Regular Expressions**, we found the relevant bedroom and bathroom data in the description column that we used to populate the fields.

```
-- 1.4: Handle Inconsistent Data: Correct inconsistent or inaccurate data(bed/baths columns)


/* Using REGEXP to find listings with 0 bed or bath that have descriptions of bed/bath so that we can update them.
127 records found for Multi_Family hometypes */

SELECT id,
       bathrooms,
       bedrooms,
       description,
       regexp_matches(description, '(\d+)[\s-]*(bed(room)?|bath(room)?(s)?|full\s+bath(s)?)[\s,-]*(\d+)?','gi')
FROM g_housing
WHERE hometype = 'MULTI_FAMILY' AND (bedrooms = 0 OR bathrooms = 0)
ORDER BY bedrooms DESC, bathrooms DESC;

/* ids of Multi_Family to update: "31639-2072223671","31602-248201096","31602-2071770776","31601-217674146",
"31602-248201097","31602-248201057","30108-71746930","30415-76364920",  "30520-2069567562","30017-14721246"
"31027-2070051057", "30093-14776437","30071-2069710884","30417-247166169","30046-14720333","30046-14724390"
"30314-35834054","30096-14786700","30417-2087070435","30228-69916037","30318-69382789","30012-14998190"
"31811-299036383","31811-299036388","30344-35853078","30165-2069653573","30014-2082533786","30014-69860893"
"30014-69887141","30310-35842739","30312-65441115","30017-14721251","30017-14721237","30305-35900793"
"30240-2125751331","31537-240254700","30017-14721243","30017-2069927215","30017-14722251","30017-14722256"
"30017-14722252","30501-2071106315","30224-69895285","30236-35790794","30121-2069810607","30315-35892457"
"31794-227347040","30058-70839448","31794-227350438","31794-227346845","31794-227345449","30161-217338552"
"30317-14466278","30525-2069576184","30401-2073114878","30291-2070335389","30663-252314697","30337-55016988"
"30012-2069621595","30308-71752019","31204-210204723","30017-14721236","30046-14723744","31023-105215604" */
```

After uncovering the **127 inconsistent listings for Multi_Family hometypes**, we next checked the Single_Family homes.

```
-- 8 records found for SINGLE_FAMILY hometypes

SELECT id,
       bathrooms,
       bedrooms,
       description,
       regexp_matches(description, '(\d+)[\s-]*(bed(room)?|bath(room)?(s)?|full\s+bath(s)?)[\s,-]*(\d+)?','gi')
FROM g_housing
WHERE hometype = 'SINGLE_FAMILY' AND (bedrooms = 0 OR bathrooms = 0)
ORDER BY bedrooms DESC, bathrooms DESC;


/* ids of Single_Family to update:"30184-55491257","31069-70827021"
"30607-54366335","30747-2071169210"*/


-- 1 records found for CONDO hometypes

SELECT id,
       bathrooms,
       bedrooms,
       description,
       regexp_matches(description, '(\d+)[\s-]*(bed(room)?|bath(room)?(s)?|full\s+bath(s)?)[\s,-]*(\d+)?','gi')
FROM g_housing
WHERE hometype = 'CONDO' AND (bedrooms = 0 OR bathrooms = 0)
ORDER BY bedrooms DESC, bathrooms DESC;

/* ids of CONDO to update:"30605-2069710143"*/
```

Above, we found **9 records** that needed further analysis for **Single_Family and Condo** homes. After further investigation, we found **5 records needed updating**.

Finally, we analyzed the final hometype, **LOT, which yielded 29 records that needed updating**. After all records were flagged, we ran the **UPDATE** statement and **DELETE statement** to make our data consistent and accurate.

```
568  -- 186 records found for LOT hometypes
569
570  SELECT id,
571         bathrooms,
572         bedrooms,
573         description,
574         regexp_matches(description, '(\d+)[\s-]*(bed(room)?|bath(room)?(s)?|full\s+bath(s)?)[\s,-]*(\d+)?','gi')
575  FROM g_housing
576  WHERE hometype = 'LOT' AND (bedrooms = 0 OR bathrooms = 0)
577  ORDER BY bedrooms DESC, bathrooms DESC;
578
579  /* ids of LOTS to update:"31632-76508777","31064-2075414403","31063-111810006","31546-105433918","31005-49861887"
580  "30094-14994046","30564-105323006","30721-214754566","30241-2072211750","31032-2079673115","30240-2097965622"
581  "30547-2072716008","31076-2071593083","30630-232367700","30442-2071053141","31791-242687198","30635-105241420"
582  "30543-101327579","30525-2077488869","30628-89890606","31714-193667520","30531-89880673","30728-58519842"
583  "30629-230148045","30176-2095958273","31021-243909060","30045-14722414","31037-2069653151","30669-2074037854"  */
584
585
586  --Updating all the records for the different hometypes that I flagged above. /* All Records Updated */
587
588  UPDATE g_housing
589  SET bedrooms = 1,
590      bathrooms = 1
591  WHERE id = '30669-2074037854';
592
593  -- Deleting records that didn't have enough info to impute. /* Deleted */
594  DELETE FROM g_housing
595  WHERE id = '31601-217674146'
596  RETURNING *;
```

This **concludes** our task. All data is now considered consistent and accurate.

Next, we were **tasked with checking if our data types were correct and standardized**. After looking through the dataset, **all data types were ready** and standardized for the analysis and visualization phases.

Lastly, **we need to validate** (check for nulls/missing values, duplicates, outliers) our data before saving the cleaned data to a new file.

As mentioned earlier, extreme outliers were found in the price column. Remembering this, we need to update our table to make sure it is validated for the analysis phase of our project.

```sql
--Finding outliers using the N_tile function. Filtering the outliers in the bottom 1 and top 100 percentiles.

WITH q1 AS (
            SELECT price, NTILE(100) OVER (ORDER BY price) AS percentile
            FROM g_housing)

SELECT price
FROM q1
WHERE percentile IN (1, 100);

-- Deleting the 255 price outliers to  make the data more representative

DELETE FROM g_housing
WHERE price IN(
            WITH q1 AS (
            SELECT price, NTILE(100) OVER (ORDER BY price) AS percentile
            FROM g_housing)

SELECT price
FROM q1
WHERE percentile IN (1, 100))
RETURNING *;
```

We used the **NTILE function** to order the prices into **percentiles from 1-100**. We put this query into a CTE, then selected the prices that were only in the bottom 1 and top 100 percentiles, our outliers. Finally, we put this query into a DELETE statement to take the 255 price outliers from our dataset.

This **concluded our validation so we saved the cleaned data to a new file** to prep it for the analysis phase.

. . .

Step 4: Analyze

4.1 Performing Calculations

Pulling statistics for analysis:

1. Count number of listings for top city and hometype. Find average home price, square footage, bedrooms and bathrooms.

2. Count the top 10 most popular cities measured by a count of home listings.
3. Calculate the average home price by city.
4. Calculate the correlation coefficient of price to square feet, price to bedrooms, price to bathrooms, and price to year built.

```sql
--Finding the Average Listing in Georgia(city, hometype, price, livingarea, bed, bath)
SELECT city,
       COUNT(*) AS city_count,
       hometype,
       COUNT(*) AS hometype_count,
       ROUND(AVG(price::numeric)) AS avg_price,
       ROUND(AVG(livingarea::numeric)) AS avg_sqft,
       ROUND(AVG(bedrooms::numeric),2) AS avg_beds,
       ROUND(AVG(bathrooms::numeric),2) AS avg_baths
FROM g_housing_cleaned
GROUP BY city, hometype
ORDER BY 2 DESC, 4 DESC
LIMIT 1;
```

Data Output | Messages | Notifications

	city text	city_count bigint	hometype text	hometype_count bigint	avg_price numeric	avg_sqft numeric	avg_beds numeric	avg_baths numeric
1	Atlanta	433	SINGLE_FAMILY	433	548230	2353	3.73	2.91

Step 1's Results

```sql
-- Number of homes in each location

SELECT city,
       COUNT(*) AS num_of_homes
FROM g_housing_cleaned
WHERE bathrooms <>0 AND bedrooms <>0
GROUP BY city
ORDER BY COUNT(*) DESC, city ASC
LIMIT 10;
```

Data Output | Messages | Notifications

	city text	num_of_homes bigint
1	Atlanta	716
2	Savannah	221
3	Marietta	209
4	Macon	172
5	Columbus	144
6	Augusta	140
7	Albany	122
8	Decatur	117
9	Lawrenceville	115
10	Athens	111

Step 2's results

```sql
-- Avg Price per location
SELECT city,
       ROUND(AVG(price::numeric)) AS avg_price
FROM g_housing_cleaned
WHERE bedrooms <> 0 AND bathrooms <> 0
GROUP BY city
ORDER BY ROUND(AVG(price::numeric)) DESC;
```

Data Output | Messages | Notificatior

	city text	avg_price numeric
1	Juliette	1600000
2	Plainville	1112500
3	Woodville	1055000
4	Mount Airy	949000
5	Berkeley Lake	940000
6	Lyerly	879500
7	Tybee Island	863804
8	Fairmount	806362
9	Bishop	779557
10	Cherry Log	777692
11	Roopville	769083
12	Saint Simons Island	760317
13	Greensboro	754379
14	Brookhaven	751761
15	Menlo	750000

Step 3's results(truncated for visual purposes)

```sql
-- Checking correlation between livingarea and price
SELECT ROUND(corr_price_livarea::numeric,2) AS corr_price_livarea
FROM(SELECT CORR(livingarea,price) AS corr_price_livarea
    FROM g_housing_cleaned
    WHERE bathrooms <> 0 AND bedrooms <> 0) t;
```

Data Output | Messages | Notifications

	corr_price_livarea numeric
1	0.52

Step 4's Price to Sqft. Correlation

```sql
-- Checking correlation between bedrooms and price
SELECT ROUND(correlation::numeric,2) AS corr_price_bed
FROM(SELECT CORR(bedrooms,price) AS correlation
FROM g_housing_cleaned
WHERE bathrooms <> 0 AND bedrooms <> 0) t;
```

Data Output | Messages | Notifications

	corr_price_bed numeric
1	0.40

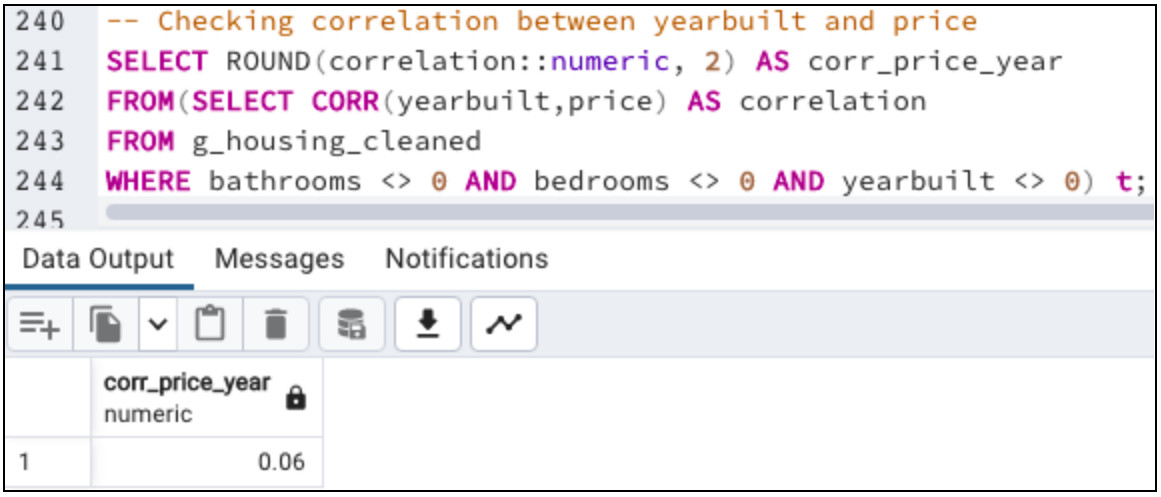
Step 4's Price to Beds Correlation

```sql
-- Checking correlation between bathrooms and price
SELECT ROUND(correlation::numeric, 2) AS corr_price_bath
FROM(SELECT CORR(bathrooms,price) AS correlation
FROM g_housing_cleaned
WHERE bathrooms <> 0 AND bedrooms <> 0) t;
```

Data Output | Messages | Notifications

	corr_price_bath numeric
1	0.64

Step 4's Price to Baths Correlation



Step 4's Price to Year Built Correlation

Interpreting statistical findings:

1. The **average home** you will find **in Georgia** is a **single family** home located in **Atlanta** with a price of **$548,230** giving you **2353 sqft., 3.73 bedrooms** and **2.91 bathrooms**.
2. **Atlanta is the most popular destination** for home listings followed by Savannah, Marietta and eventually Athens rounding out the top 10.
3. Average home **prices per city can be affected by the sample of listings per city.** The top 10 most popular cities for listings will have a very accurate representation compared to tiny cities.
4. The **price correlation coefficient** is **0.52** for sqft., **0.40** for bedrooms, **0.64** for bathrooms and **0.06** for year built.

. . .

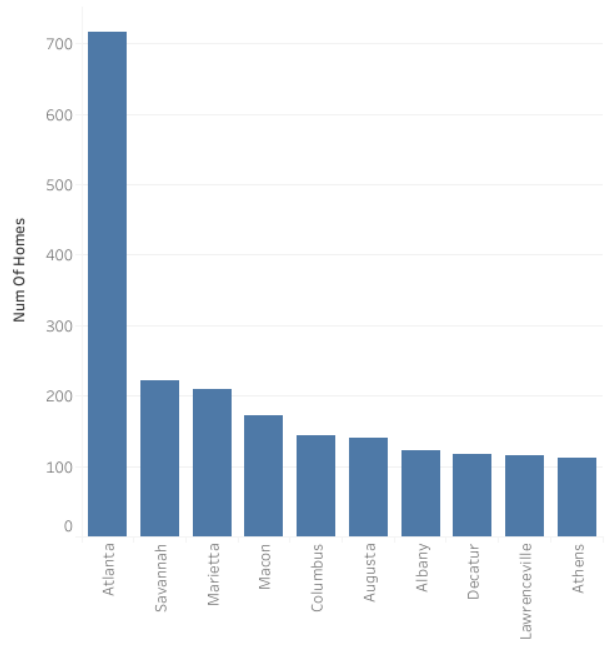
Step 5: Share

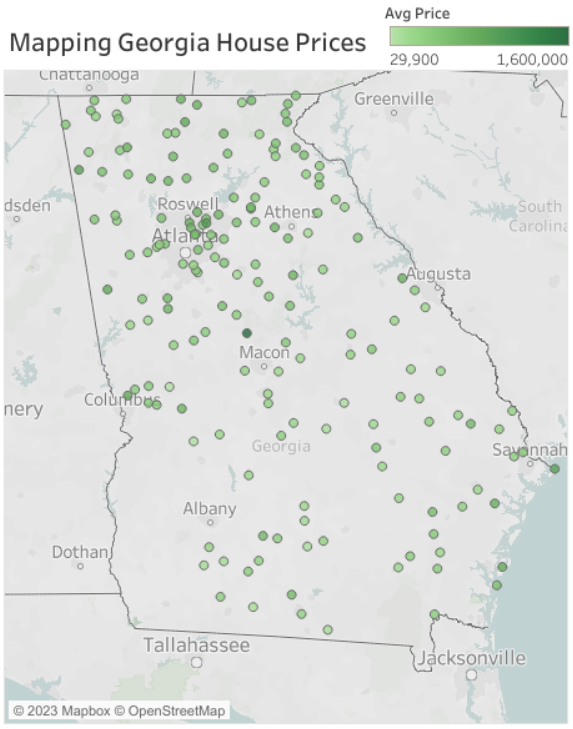
In this step, we are creating visualizations and communicating our findings based on our analysis.

4.1 Data Visualizations and Findings

Average Listing in Georgia

City	Hometype	Price	Square Footage	Bedrooms	Bathrooms
Atlanta	Single Family	$548,230	2,353	3.73	2.91

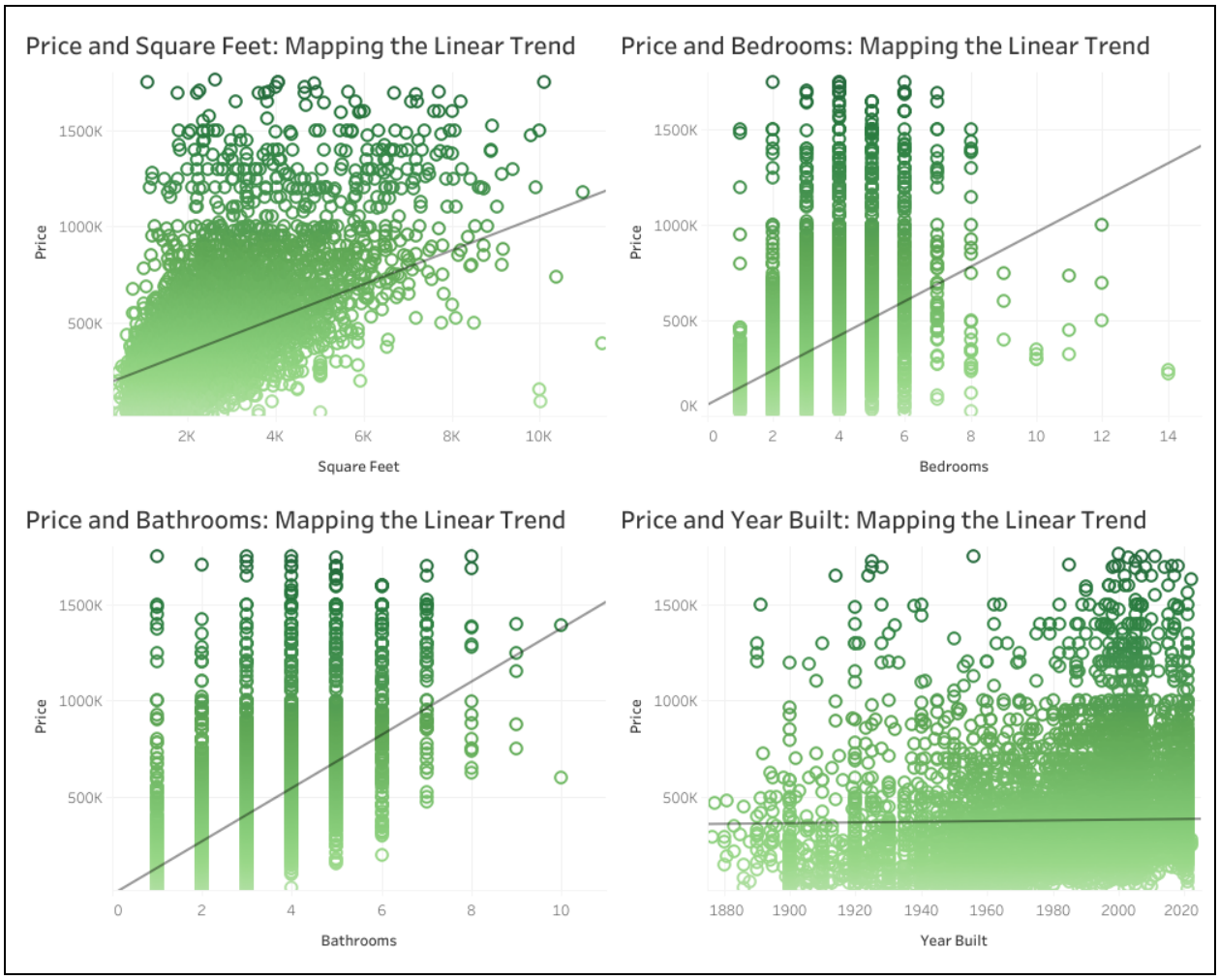




Average Home Listings in Georgia

In this dashboard, we are looking at the typical average Georgia home listing, the top 10 cities by house listings and a map of average house prices in Georgia.

1. We discovered your typical listing in Georgia will be found in Atlanta, cost you $550K, give you 2,353 sqft, 3.71 bedrooms on average and 2.91 bathrooms.
2. People prefer to live in Atlanta where there are a lot of career opportunities and entertainment or on the coastline in Savannah where there is a lot of beauty.
3. Average home prices increase as we geographically move closer to Atlanta's economic sphere of influence.



Performing Linear Regression on Home Prices

In this dashboard, we are looking at the breakdown of linear regression for price correlated to the variables of square feet, bedrooms, bathrooms and year built.

1. We discovered that these four variables all have positive correlations to an increasing price.
2. The strongest correlation was exhibited in the amount of bathrooms the home has.
3. We discovered that after the amount of bathrooms, square footage and bedrooms had the most influence on price
4. We noted that the year built doesn't have a particularly strong correlation to an increasing price.

Step 6: Act

In the final step, we will be delivering our insights and providing recommendations based on our analysis.

Here, we revisit our business questions and share with you our high-level business recommendations.

1. **What trends have been identified?**
    - The majority of people prefer to live in single family homes in the Atlanta metro area.
    - House prices are higher on average in and around Atlanta. This signals there is more desire to live near economically thriving areas with a lot of entertainment.

2. **How do different variables affect home prices?**
    - The strongest indicator of a home price is the amount of bathrooms(0.64 coefficient). After that the most influential variables are square footage(0.54 coefficient) and bedrooms(0.46) in sequential order. Surprisingly, the year a home is built (0.06) barely affects the price of it.

3. **How can these trends and the variables' differing price effects help the Schlachters form their house-hunting strategy?**
    - By understanding these trends and knowledge of how different variables affect home prices, the Schlachters can feel encouraged to know that they will be able to find a house in the metro area of Atlanta and remain in their budget by reducing the amount of bathrooms and square footage of the home.
    - The best home recommendation for the Schlachters is a newly constructed Atlanta area Condo with 1 bathroom, 1 bedroom, and less than 1,000 sqft.

The dataset and complete code can be found here.