Bayes the Databa(s)e

Alyssa Jones

May 4, 2018

Abstract

Databases and Data Warehouses are useful tools in almost every modern industry. Data can be extracted from them via SQL queries, thus allowing the user to visualize and analyze metrics of interest. Here I present a Data Warehouse and the SQL code and visualizations used to analyze its data.

1 Introduction

For the final project we were required to maintain a data warehouse on Google BigQuery. I named my data warehouse Bayes-the-Databae because I really like Bayes' Theorem. The bulk of Bayes-the-Databae contained databases of Airbnb data from Austin, Portland, and Nashville. These databases were populated from already existing data tables that did not have to be manipulated in any way. Also included in Bayes-the-Databae was a Zillow database. Populating the Zillow database required transforming a dataset via Map Reduce and then ingesting the data using Google Cloud Dataflow.

Once Bayes-the-Databae was up and running, it could then be used to address a scenario in which the city of Austin desired an analysis of the available Airbnb listings and their impact on local housing affordability.

2 Methods

2.1 BigQuery

Bayes-the-Databae was hosted on Google BigQuery. Advantages of BigQuery are its columnar storage, lack of keys, and its ability to divide queries into sub-queries to be run in parallel. BigQuery was an optimal choice for this project because we did not have a need to update and/or delete data, which is its major weakness.

2.2 SQL

To extract data from Bayes-the-Databae, I had to compose SQL queries. In BigQuery I used Standard SQL. Most queries were of the form:

```
SELECT field_1, ..., field_n
FROM table_a
JOIN table_b
ON key_a = key_b
WHERE condition;
```

I will go over my queries in more detail in the Discussion.

2.3 Tableau

To visualize interesting queries I considered both Matplotlib and Tableau, but ultimately went with Tableau. I had three main reasons for choosing Tableau.

1. Connecting to my database was as simple as clicking the Google BigQuery option when adding a new data source, and then selecting my table of choice. I could have also queried directly from Tableau, but I had already saved queries of interest as views.
2. I wanted to visualize clustering of listings around geographic markers and Tableau has a wonderful built-in Geographic Symbol Map option.
3. I wanted to learn something new and Tableau is free while I'm still a student!

2.4 Dataflow and Apache Beam

The data used to populate my Zillow database was not in a queryable format so it needed to be transformed before ingesting it into BigQuery. To do this efficiently, I utilized Google Cloud Dataflow and Apache Beam.

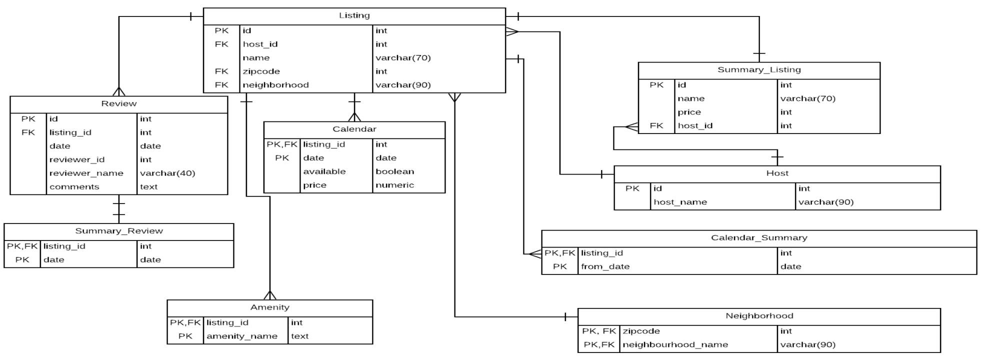
Dataflow and Beam were used to construct a pipeline job which can be broken down into three parts: reading from a data source (in my case, a Google Cloud bucket), transforming the data into BigQuery table rows, and finally writing the table rows into the specified BigQuery table.

Pseudocode for a typical pipeline job:

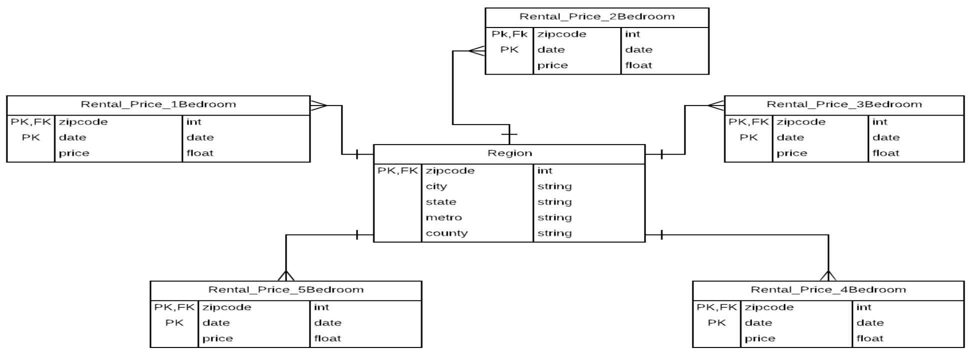
```
# initialize table schema
# create BigQuery record
# parse line
# parse record
# run pipeline job with beam
```

2.5 Lucid Chart

I used Lucid Chart to create Entity Relationship Diagrams to represent my data warehouse. ERDs are useful because each table is represented as a smaller table broken up such that one column represents the fields, another represents the data type of each field, and another column is reserved for marking whether a field is a primary and/or foreign key. When there exists some relationship between two tables in the database, their ERD representations are joined together by a line representing the type of relationship (e.g. one-to-one, one-to-many). See Figure 1 for ERDs used in this report.



(a) Airbnb ERD



(b) Zillow ERD

Figure 1: ERDs used for report. While these represent separate databases, they could be joined on Zillow.Region.zipcode = Airbnb.Listing.zipcode.

3 Discussion

3.1 Schema

My schema used for Bayes-the-Databae is reflected in the ERDs shown in Figure 1. While BigQuery itself does not require primary or foreign keys, it is helpful to keep them in the ERD in order to know which field to join two tables on.

3.2 Occupancy Rate

The occupancy rate of existing Airbnb listings is important information to have, particularly when setting the price for a new listing. I used the following query to pull this data from my database.

```
select c.date, t.num_all_listings, count(c.available) as num_booked_listings
from `spooky-data-seance.austin.Calendar` c
join `spooky-data-seance.tableau.atx_num_all_listings` t
on c.date = t.date
where available = False
group by date, t.num_all_listings
order by date;
```

The number of listings and occupancy rate spiked in March, 2017, as seen in Figure 2. This coincides with South by Southwest (SXSW), which draws many out of town tech professionals, musicians, and music fans to Austin. It's interesting to note that both metrics remained higher than before, even after SXSW 2017 ended.

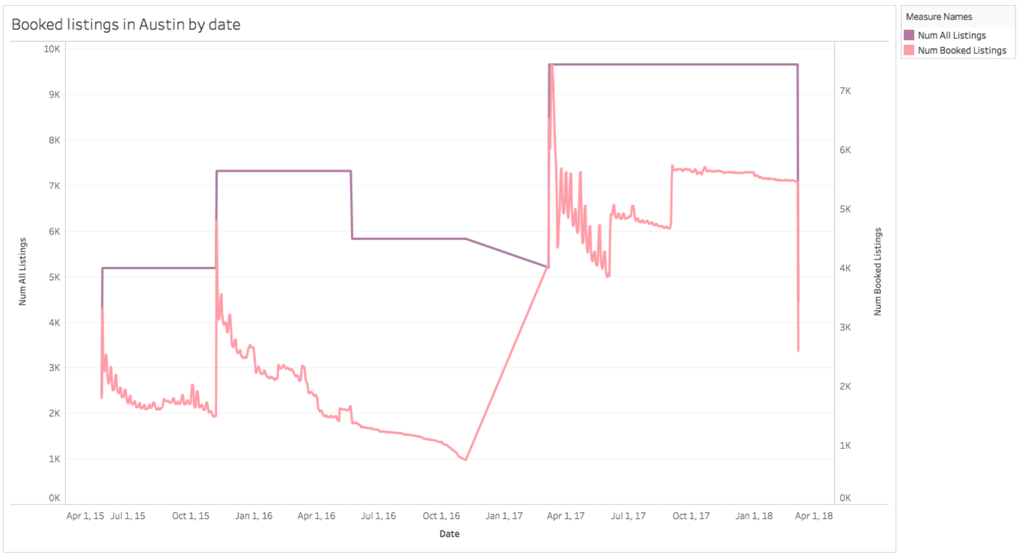


Figure 2: Booked listings in Austin by date.

3.3 Revenue Crossover

One metric of interest was the revenue crossover point, or how many days an individual would need to rent our their Airbnb listing to match the median monthly rental price for their zipcode.

I first created a view called austin.price_per_day_helper by selecting the (truncated) data, zipcode, listing id, and price of the listing from the calendar table, when available and the listing table otherwise.

```
select date_trunc(c.date, MONTH) as date, l.zipcode, l.id,
        case
                when c.price is null then l.price
                else c.price
        end as airbnb_price_day
from `spooky-data-seance.austin.Calendar` c join `spooky-data-seance.austin.Listing`
```

Then I created a view called v_atx_price_per_day by selecting the data, zipcode, number of bedrooms, and median daily rental price partitioned over date, zipcode, and bedrooms.

```
select h.date as date, h.zipcode as zipcode,
        case
                when l.bedrooms = 1 then 1
                when l.bedrooms = 2 then 2
                when l.bedrooms = 3 then 3
                when l.bedrooms = 4 then 4
                when l.bedrooms = 5 then 5
        end as bedrooms,
        percentile_cont(h.airbnb_price_day, 0.5) over(partition by h.date, h.zipcode,
from `spooky-data-seance.austin.price_per_day_helper` h join `spooky-data-seance.austi
on h.id = l.id
where l.bedrooms is not null;
```

After wrangling the Airbnb data, I then had to wrangle the Zillow data into one nice view with the following fields: zipcode, date, price (monthly), and bedrooms.

```
select br.zipcode, br.date, br.price as zillow_price_month, 1 as bedrooms
from `spooky-data-seance.zillow.Rental_Price_1Bedroom` br
join `spooky-data-seance.zillow.Region` r
on br.zipcode = r.zipcode
where r.city = 'Austin'
UNION ALL
select br.zipcode, br.date, br.price as zillow_price_month, 2 as bedrooms
from `spooky-data-seance.zillow.Rental_Price_2Bedroom` br
join `spooky-data-seance.zillow.Region` r
on br.zipcode = r.zipcode
where r.city = 'Austin'
UNION ALL
select br.zipcode, br.date, br.price as zillow_price_month, 3 as bedrooms
from `spooky-data-seance.zillow.Rental_Price_3Bedroom` br
join `spooky-data-seance.zillow.Region` r
on br.zipcode = r.zipcode
where city = 'Austin'
```

```
UNION ALL
select br.zipcode, br.date, br.price as zillow_price_month, 4 as bedrooms
from `spooky-data-seance.zillow.Rental_Price_4Bedroom` br
join `spooky-data-seance.zillow.Region` r
on br.zipcode = r.zipcode
where city = 'Austin'
UNION ALL
select br.zipcode, br.date, br.price as zillow_price_month, 5 as bedrooms
from `spooky-data-seance.zillow.Rental_Price_5Bedroom` br
join `spooky-data-seance.zillow.Region` r
on br.zipcode = r.zipcode
where city = 'Austin';
```

Upon completion of the above steps, I could then join the Austin Airbnb table to my Zillow table.

```
select a.date as date, a.zipcode as zipcode, a.bedrooms as bedrooms, a.airbnb_price_da
from `spooky-data-seance.austin.v_atx_price_day` a
inner join `spooky-data-seance.zillow.austin_price_month_helper` z
on a.zipcode = z.zipcode and a.date = z.date and a.bedrooms = z.bedrooms
where z.zillow_price_month is not null
group by date, zipcode, bedrooms, airbnb_price_day, zillow_price_month;
```

The Revenue Crossover Point over time for selected zipcodes can be seen in Figure 3. Dips in the Revenue Crossover Point can be seen in the months of March and October, which will be explained more in Section 3.4.

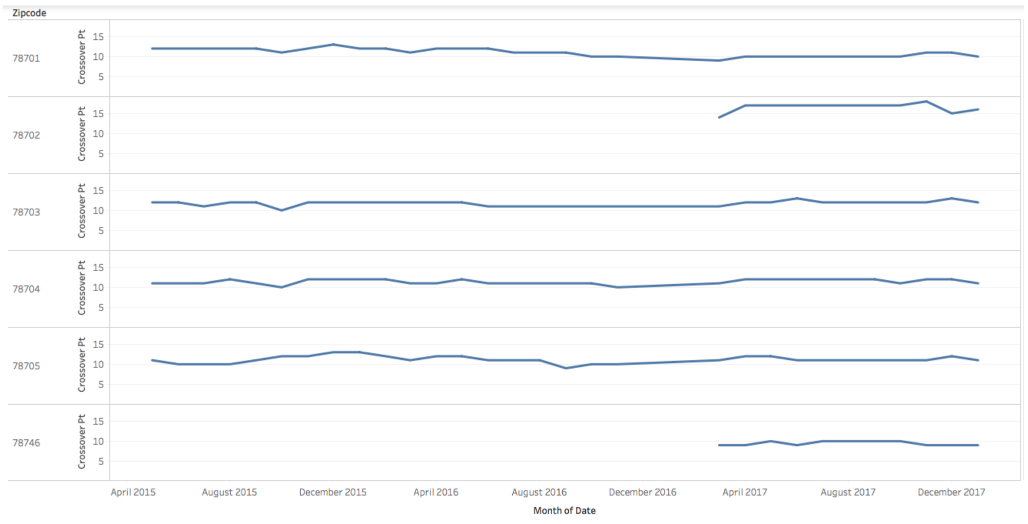


Figure 3: Revenue Crossover Point over time for selected Austin zipcodes.

3.4 Seasonal price variations

Given that Austin hosts several major festivals which draw in a large number of out-of-towners, I suspected that there may be some price variations that reflect the festival dates. The festivals I considered were SXSW, in March, and Austin City Limits (ACL), in October.

Using the data from my v_atx_price_per_day view, I obtained Figure 4. This confirmed my suspicion that, at least in the zipcodes near the downtown and Zilker Park areas, the average price of an Airbnb listing would increase during festival season.

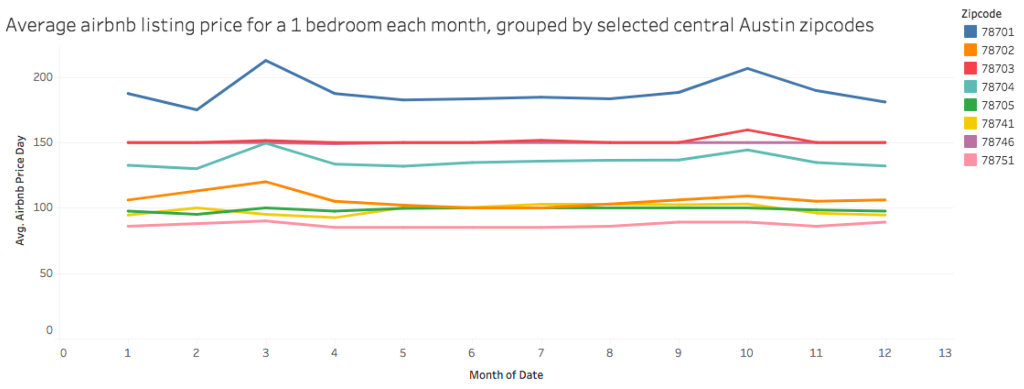


Figure 4: Seasonal price fluctuations during major events.

3.5 Listing Densities

Since Tableau has an extremely user-friendly symbol map feature, I wanted to visualize the density of Airbnb listings by zipcode.

This required the following simple query, which I then used to create Figure 5.

```
SELECT zipcode, count(id) from austin.Listing
group by zipcode;
```

This map reveals that the zipcodes surrounding the intersection of 1-35 and the Colorado River, or essentially the *cool* parts of Austin, contain the bulk of the city's Airbnb listings.

As someone who has lived in Austin for a decade, my visualization of where the bulk of Austin's Airbnb listings are located did not surprise me at all, but it was still a worthwhile analysis given that "local instinct" is not a statistically sound argument.

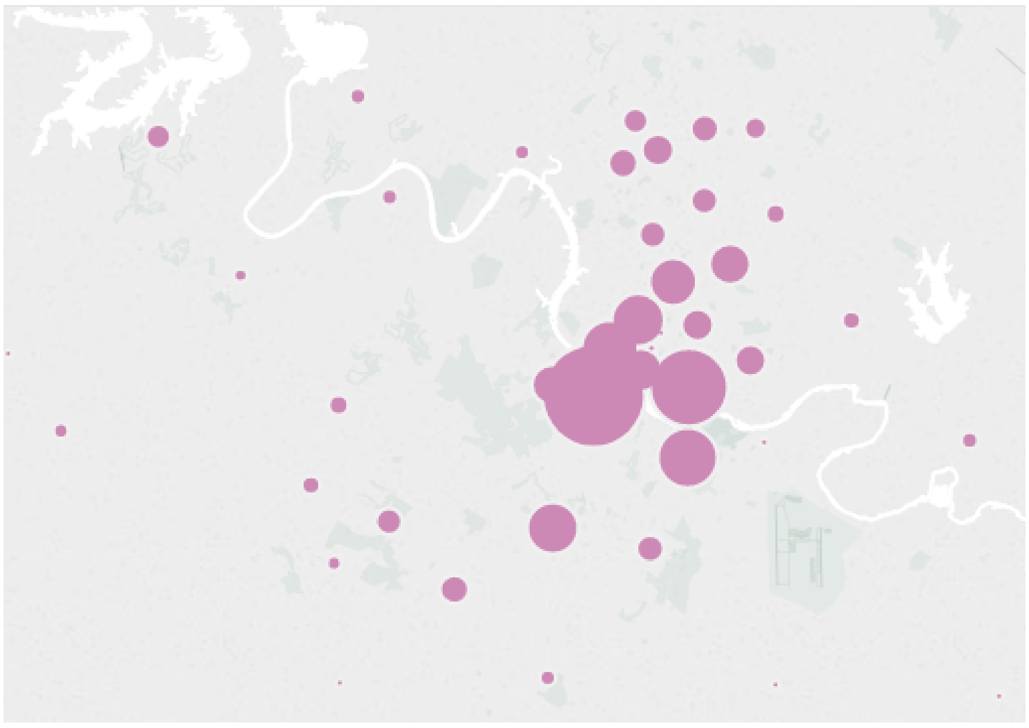


Figure 5: Circles represent zipcodes, size represents the number of listings per zipcode.

4 Conclusion

4.1 Results

Over the last year, Airbnb appears to have been gaining traction, at least in the city of Austin. This can be seen in Figure 2. Due to the increases in both demand and price during the festival seasons of March and October that we see in Figures 4, 3, and 2, it is advisable for an aspiring Host to list their domicile at these times for maximum profit.

4.2 Future Work

One thing I would have liked to explore had I had more time and more scikitlearn knowledge is whether there exists some definable relationship between amenities offered and the price that a listing goes for, the frequency that a particular listing is booked, or whether the host of the listing is likely to be labeled a Superhost or not. Since my Google Cloud free trial has 262 days remaining, I will probably take some time over the summer to explore the data in Bayes-the-Databae further.