Overview

Project Goal and Problem Statement

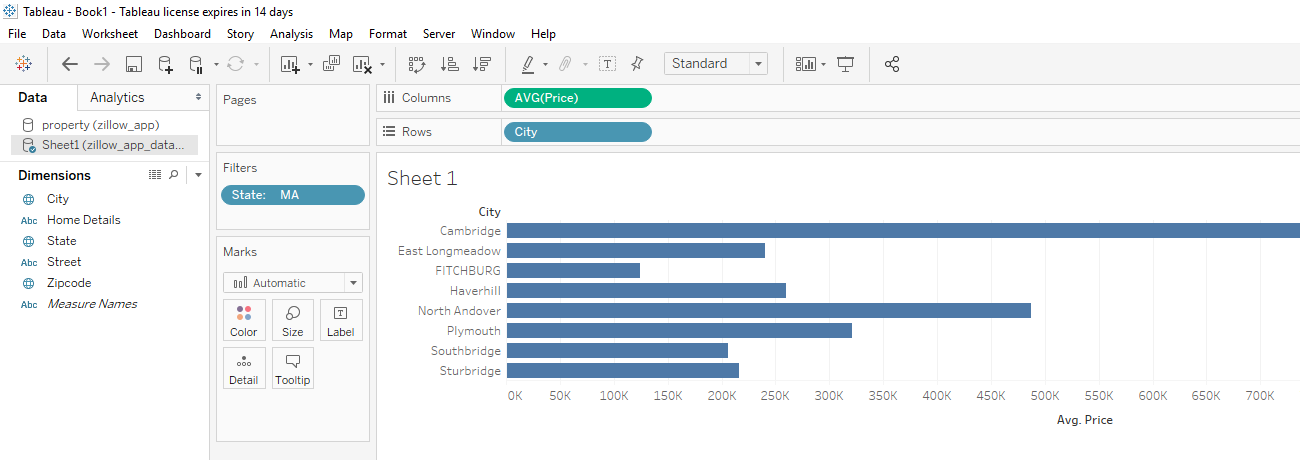
This project’s goal is to demonstrate how one can build a system that collects Zillow’s real estate data and performs further analysis via Tableau.

Big Data Source:

Zillow Real Estate API named Zestimate: <https://www.zillow.com/howto/api/GetZestimate.htm>

Results:

As a result of my processing, a user can see the average price for any city or state. Such information may be helpful when one finds themselves in the market to purchase real estate.



Youtube URL:

<https://www.youtube.com/watch?v=LVFEl47MlPw&feature=youtu.be>

## Processing Pipeline:

|  |
| --- |
| **Data Source** |
|  |

|  |
| --- |
| **Collection Tier** |
|  |

|  |
| --- |
| **Messaging Tier** |
|  |

|  |
| --- |
| **Storage Tier** |
| Image result for cassandra database icon |

|  |
| --- |
| **Visualization Tier** |
| Description: Image result for tableau |

## Pipeline Overview and Technologies used:

* Collection Tier – Flume with experimental Zillow data source
  + Flume will be ingesting real estate data from the Zillow API filtered by property that is within the state of Massachusetts.
* Messaging Tier: Kafka
  + Flume will push events into Kafka for further processing
* Storage Tier – Cassandra DB
  + A Kafka Consumer will pull the data and persist it into Cassandra.
* Visualization Tier – Tableau
  + Tableau will provide data visualization and dashboards by pointing to the Cassandra database as its source thereby allowing users to discover the average price of a home within any city of Massachusetts. Note: tableau can easily be configured for any data source. However since I came to know tableau wasn’t available on unix (where I built the platform), I extracted the dataset from Cassandra and configured tableau to use the extracted dataset.

## New Technology/Framework used:

* Zillow API Data Source
* Tableau

The goal of my project was to collect Zillow’s real estate data and make it available via Tableau for further user analysis. For this to work, I had to execute the following tasks.

1. Install Cassandra and start service.
2. Install flume, configure it, and start agent.
3. Install Kafka and start kafka server.
4. Create application to make API calls, pass the response to a kafka client, and persist it into Cassandra.
5. Configure tableau as a user visualization to view the data which shows the results

***Installing Cassandra and starting the service.***

I ran the following commands to install Cassandra:

* Wget <http://mirrors.gigenet.com/apache/cassandra/3.0.15/apache-cassandra-3.0.15.bin.tar.gz>
* tar zxvf apache-cassandra-3.0.15-bin.tar.gz
* Mkdir Cassandra
* Mv apache-cassandra-3.0.15/\* casssandra/

Then I modified the Cassandra.yaml by uncommenting:

* commitlog\_directory
* data\_file\_directories
* saved\_caches\_directory

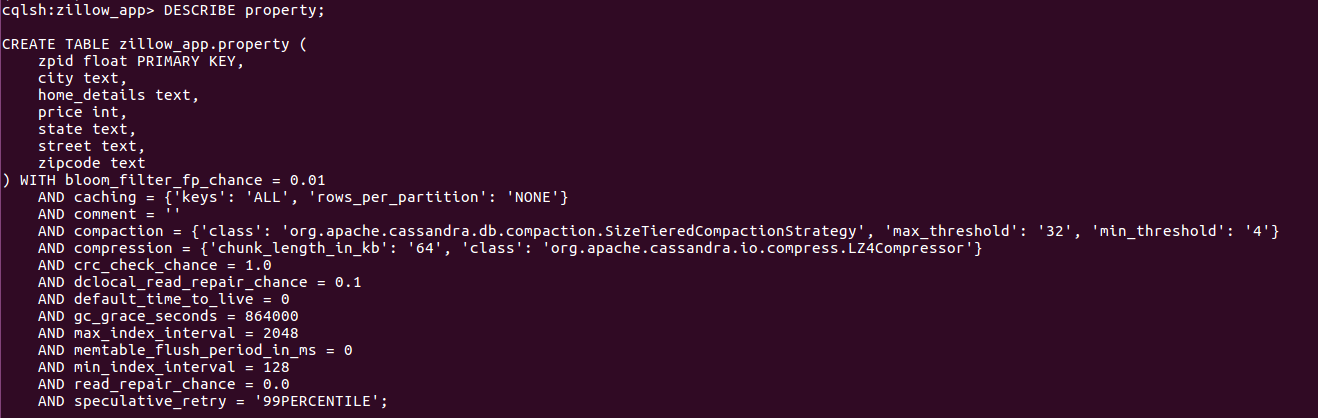
Then I start the Cassandra service with the following command:

* ./bin/Cassandra -f

Next, I create a new keyspace for the property data I wanted to store. I also add a table named, property with the fields I needed.

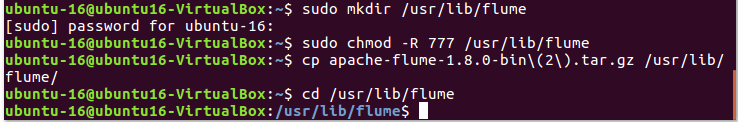
* CREATE KEYSPACE IF NOT EXISTS zillow\_app WITH replication = {'class':'SimpleStrategy','replication\_factor':1};
* USE zillow\_app;
* CREATE TABLE property (zpid float, street text, city text, state text, zipcode text, price int, home\_details text, PRIMARY KEY(zpid));

The table description can be seen in the image below.



***Installing flume, configuring it, and starting agent***

I ran the following commands to install flume.

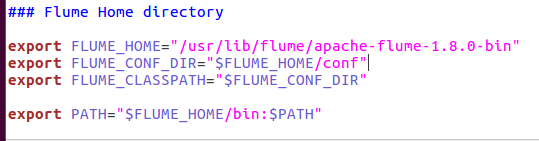




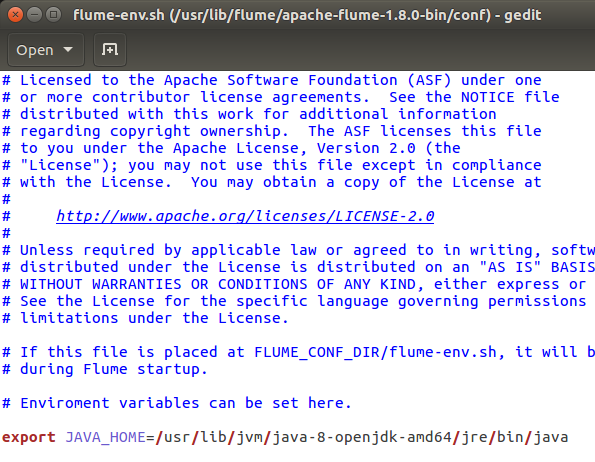
I then updated the ./bashrc file on my virtual machine.



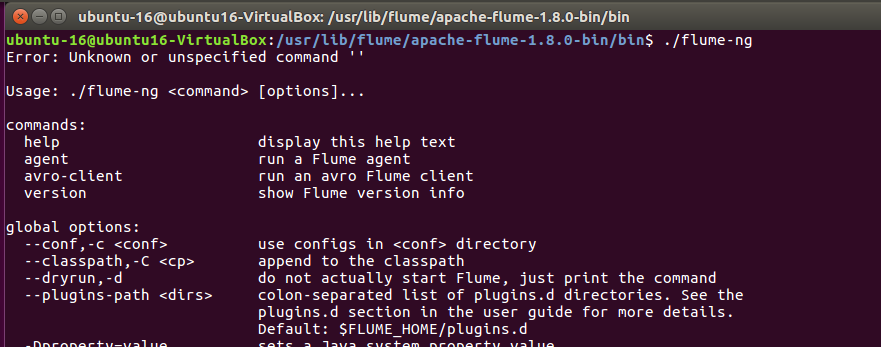
I set my flume home and flume path within the ./bashrc file.



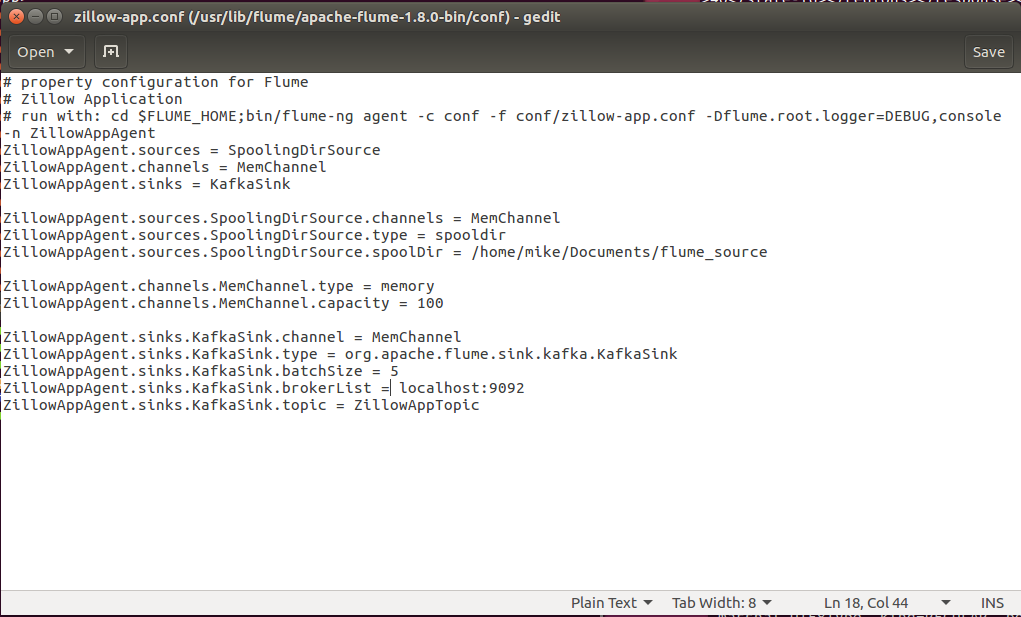
I then updated then Java Home in the configuration so that flume can identify the java on my machine.

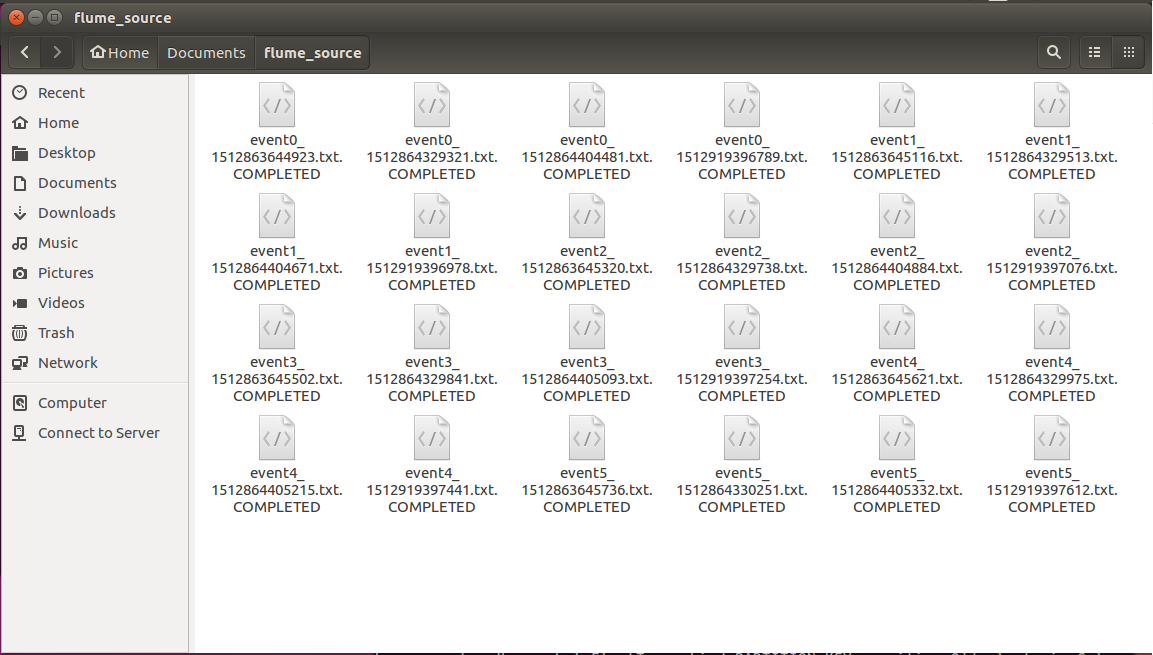


Then flume was successfully installed.



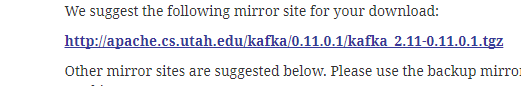
I then configured "Spooling Directory" Source - to read data from some specified directory. I use a Memory Channel and a Kafka Sink can my destination to land the data. The screen shot of my Zillow-App.conf flume configuration can been seen below. I also included it in my folder of attachments.



The image below shows what the flume source looked like when running the application.

***Installing Kafka and starting the kafka server***

I downloaded kafka from the following source:



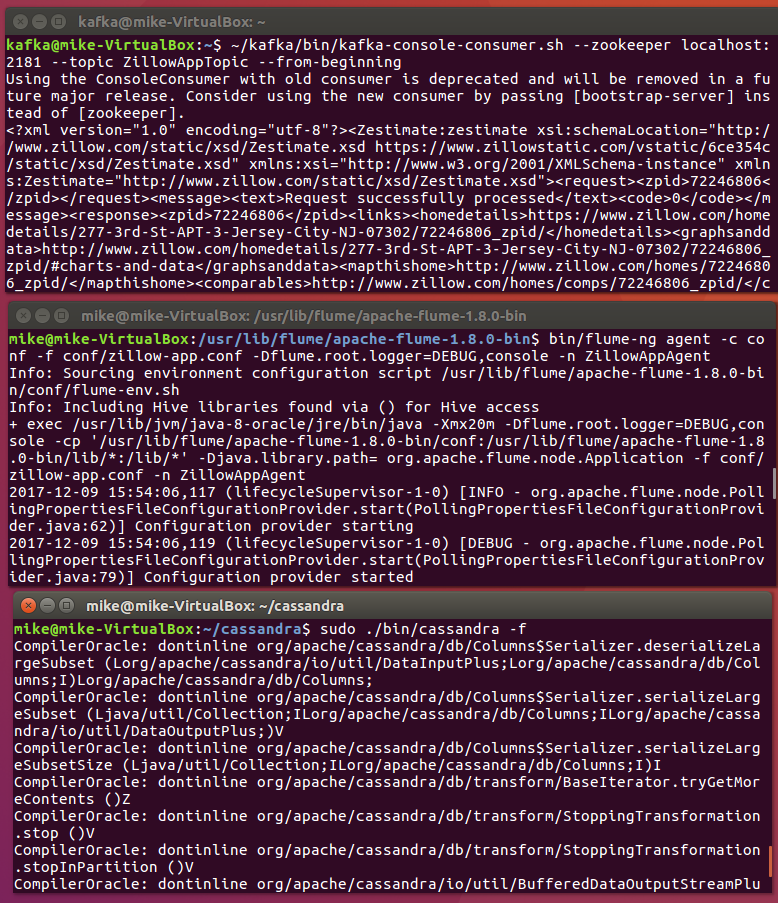
I then continued to download it with the following commands.

* Tar –xzf kafka\_2.11-0.11.0.1tgz
* Cd kafka\_2.11-0.11.01
* Bin/zookeeper-server-start.sh config/zookeeper.properties

I then created a kafka topic named ZillowAppTopic with the following command:

* Bin/kafka-topics.sh –create –zookeeper localhost:2181 –replication-factor 1 –partitions 1 –topic ZillowAppTopic

The image below shows all three running: Kafka, Flume, and Cassandra.



***Creating application to make API calls, pass the response to a kafka client, and persist it into Cassandra.***

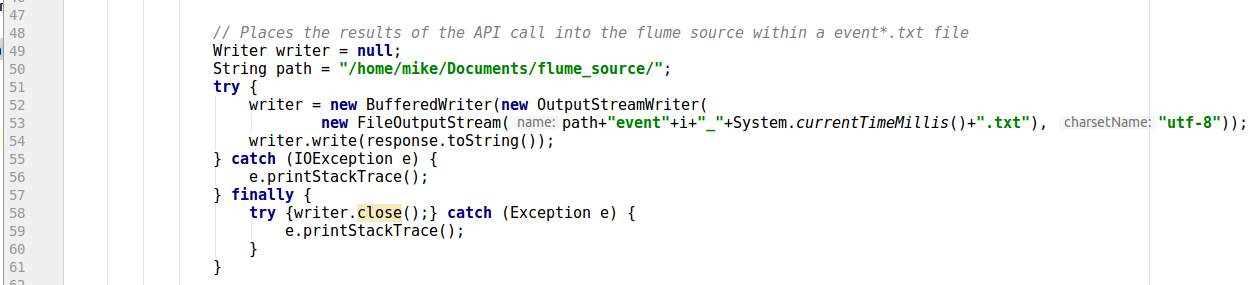
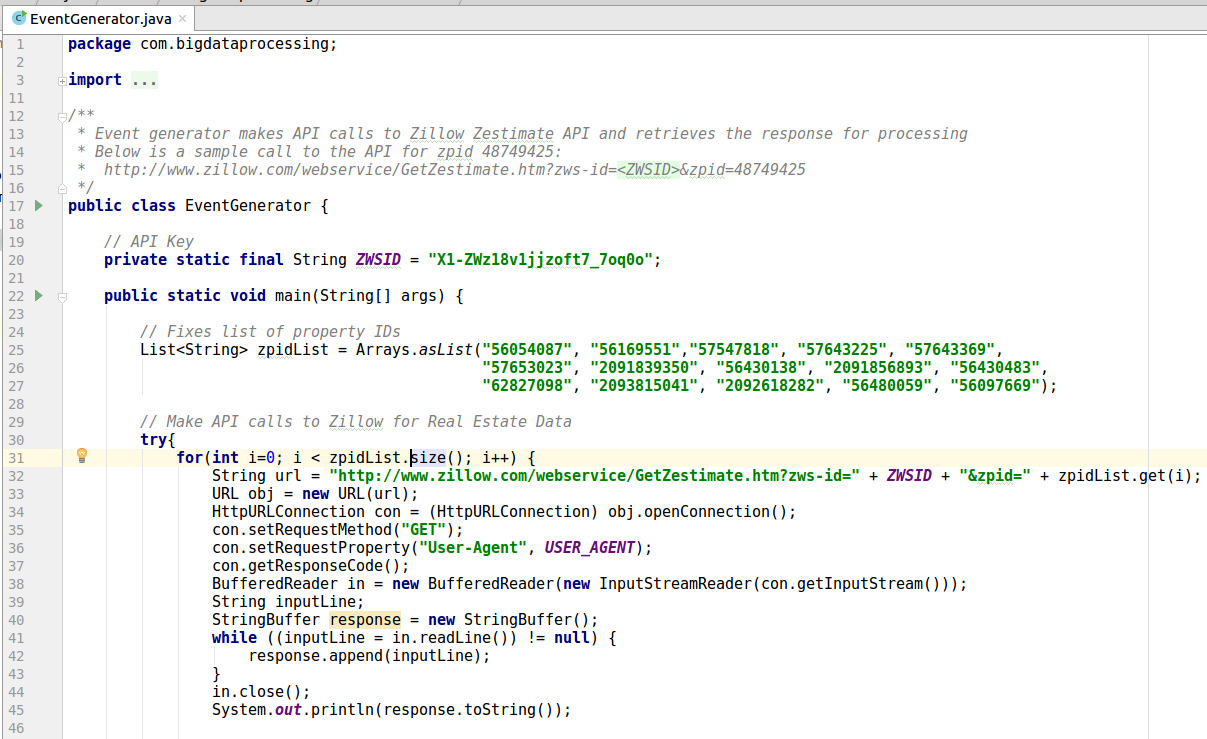
I created a simple application to make the API calls and persist the responses. The application is broken up into three parts:

* Consumer
* EventGenerator
* Persistence

Please see more about each below.

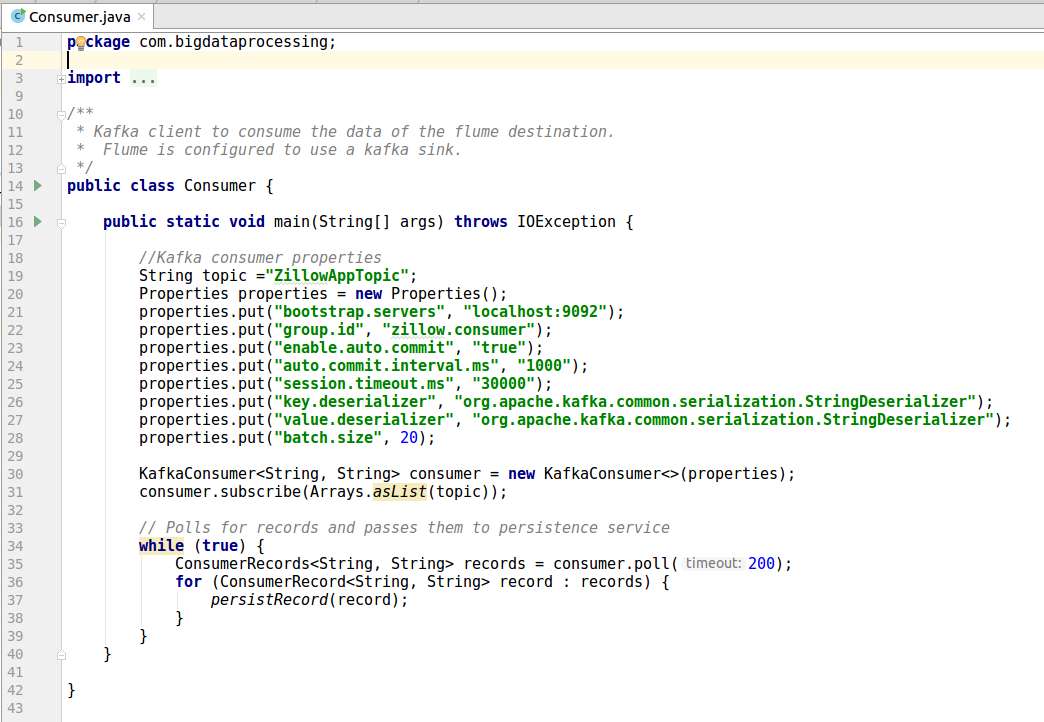
EventGenerator –

1. The event generator make the API calls to Zillow.com. On line 25 there are Zillow property IDs that I leverage to make the API call. On line 32 I form the url of the API to call. The second part of EventGenerator which starts o line 49 to 60, takes the response of the API call and places it into the source directory - /home/mike/documents/flume\_source/. Note that since flume is running with a kafka sink. The files are processed and placed onto the kafka topic to be consumed.



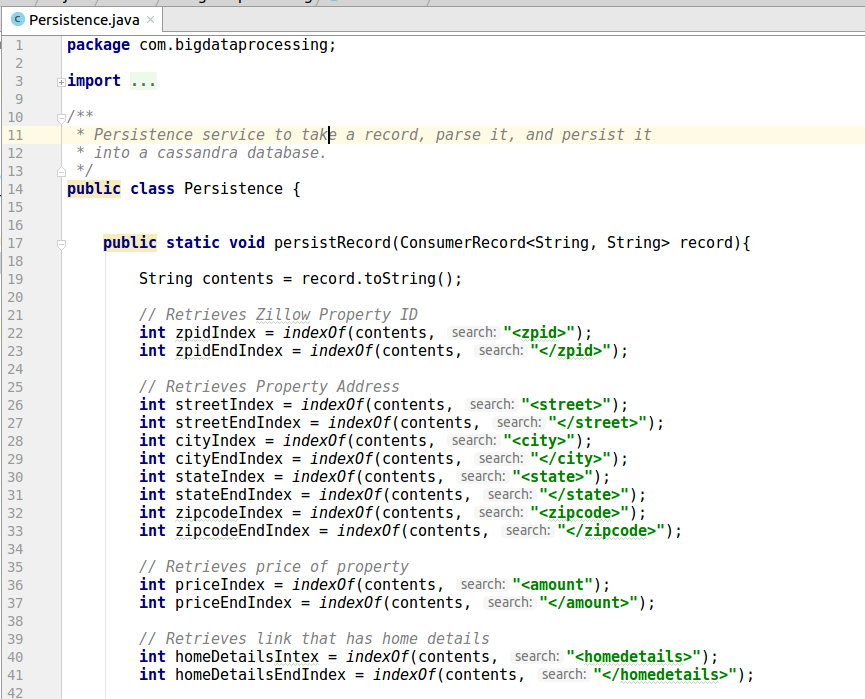
Consumer –

1. The Consumer is a simple kafka client. As seen below – the properties of the kafka consumer are first set. The consumer continues to poll for records from the flume source and as it gets them – passes them to the persistence service to save it in Cassandra.



Persistence –

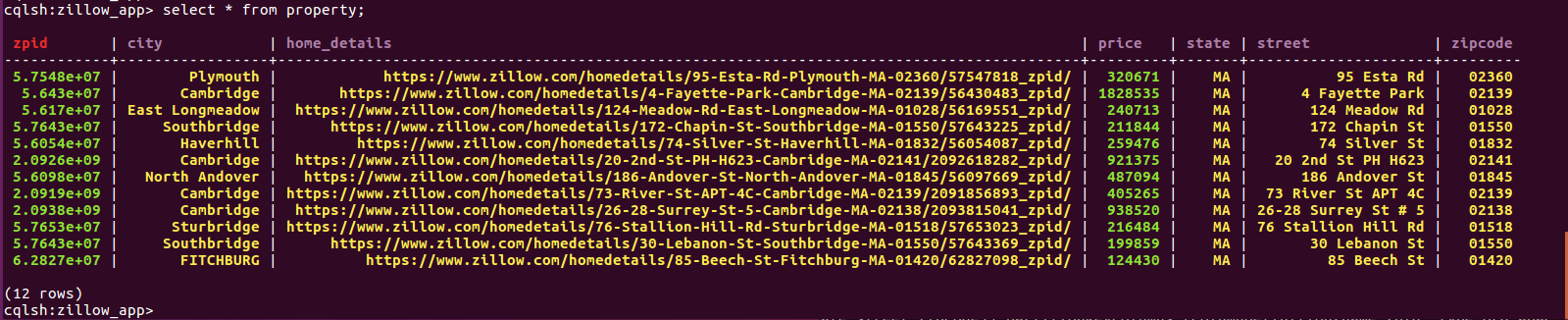
1. Once a kafka record is passed to persistence, it parses it to look for certain data. On line 21 to 40, you can see I am looking for the property’s address, price, and details.



The second part of Persistence builds an INSERT query and executes it. This saves the data in the Cassandra table that I built above.



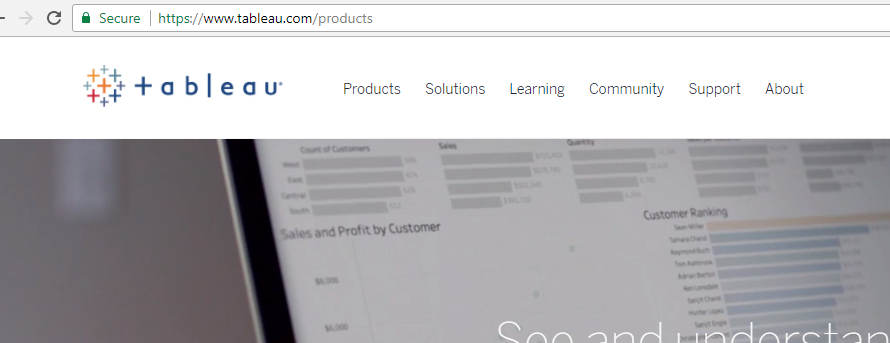
When this application is run – the data persistence in the Cassandra Property table look like this:



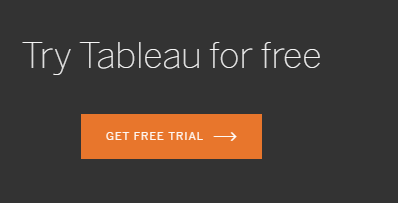
***Configuring tableau as a user visualization to view the data.***

I visited https:www.tableau.com.

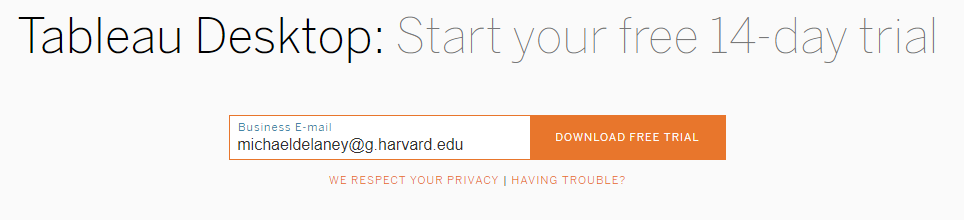
I navigated to Products > Desktop solutions.



Next, I opted to download their free trial.

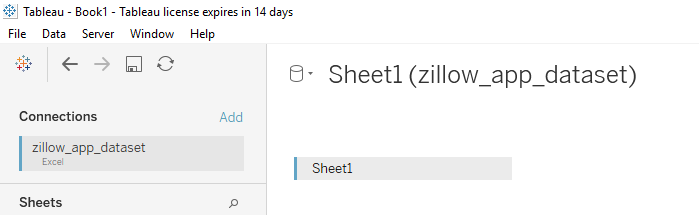


Once I supplied my information. I received my executable to install tableau locally.

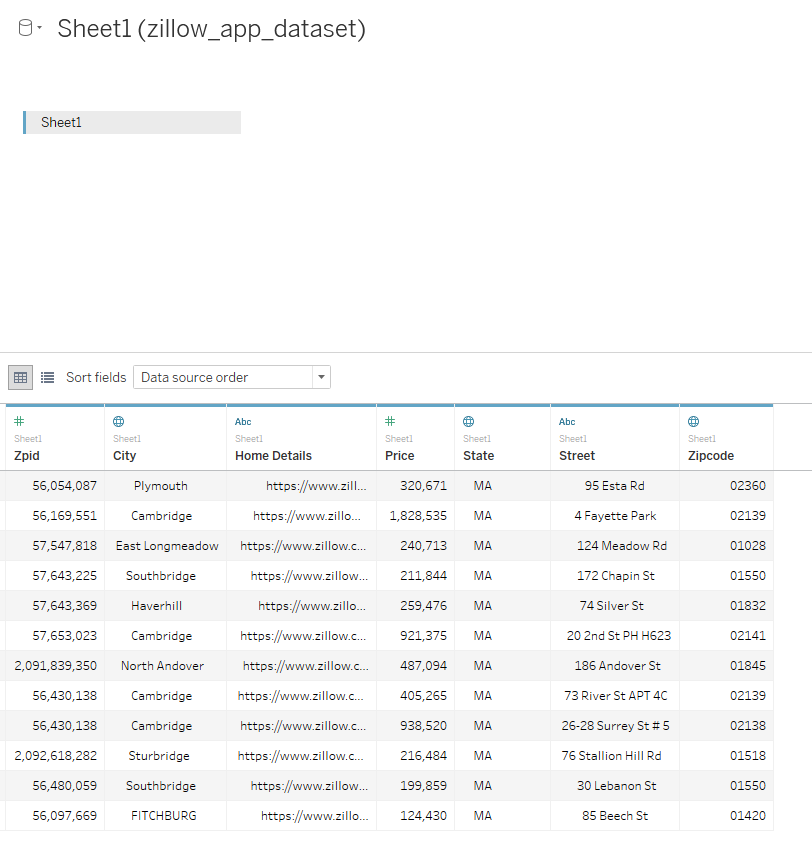




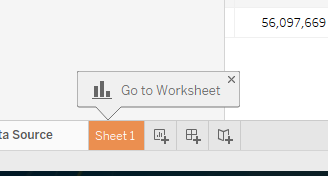
Once I installed the software. I opened up the GUI and connected to a dataset extract of the Zillow data I aggregated.



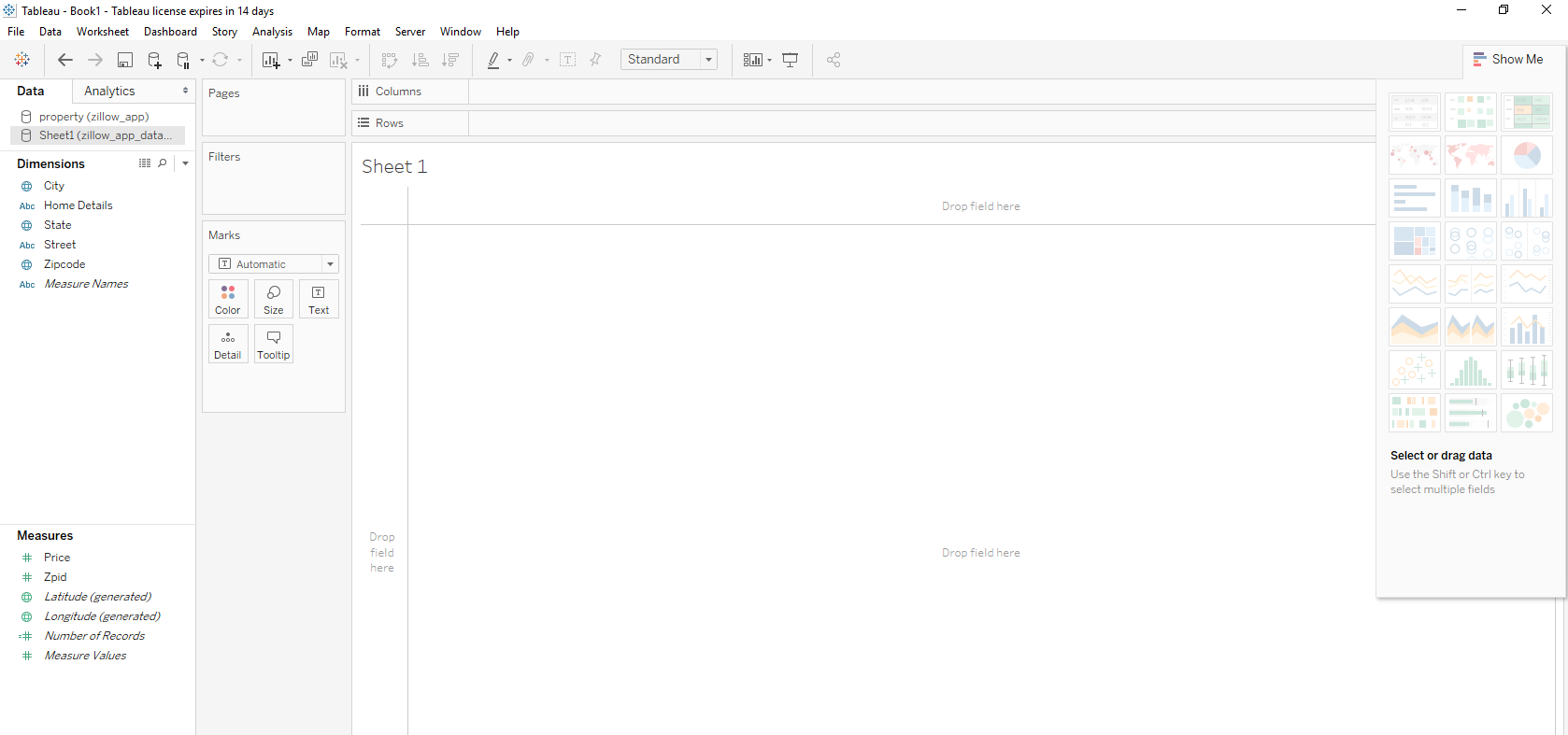
This created a new worksheet.



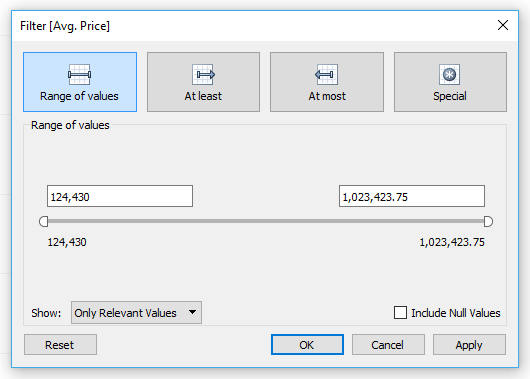
Next I clicked Sheet 1 for visualization. This allows us to play with the data to learn new things.



The image below shows a blank worksheet.



Next I configured the work sheet to compute the average price per city. I also added a filter to show only properties in the MA state.



As a result, the sheet looked like this:

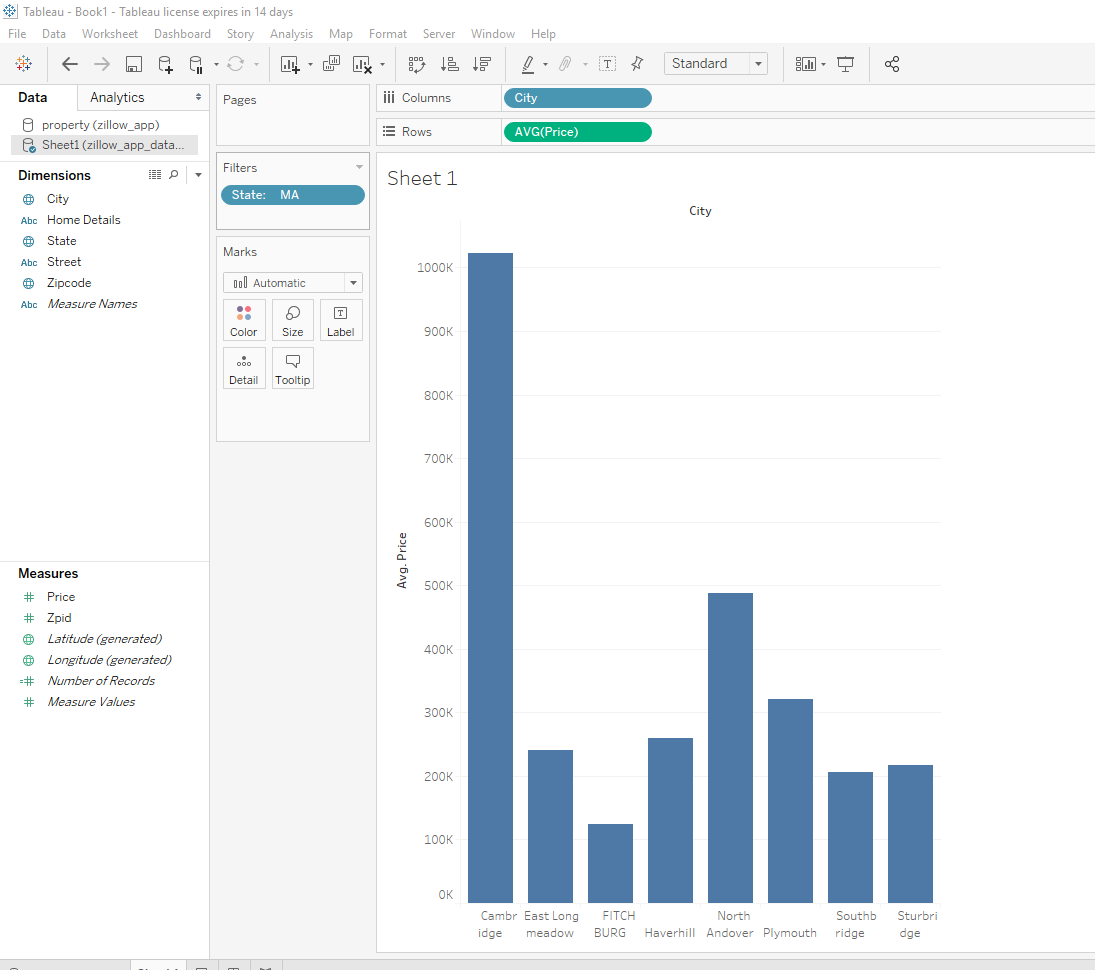
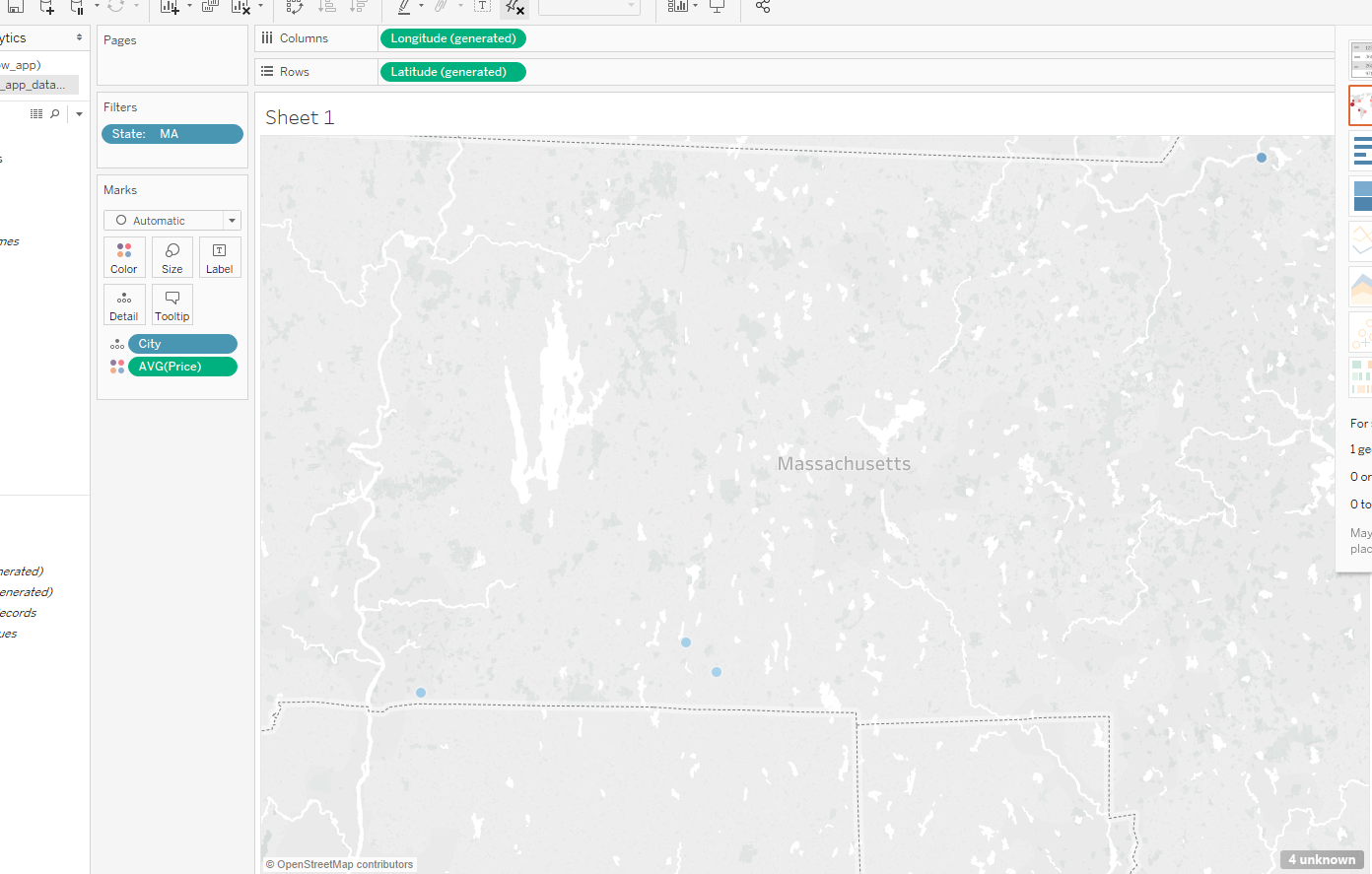
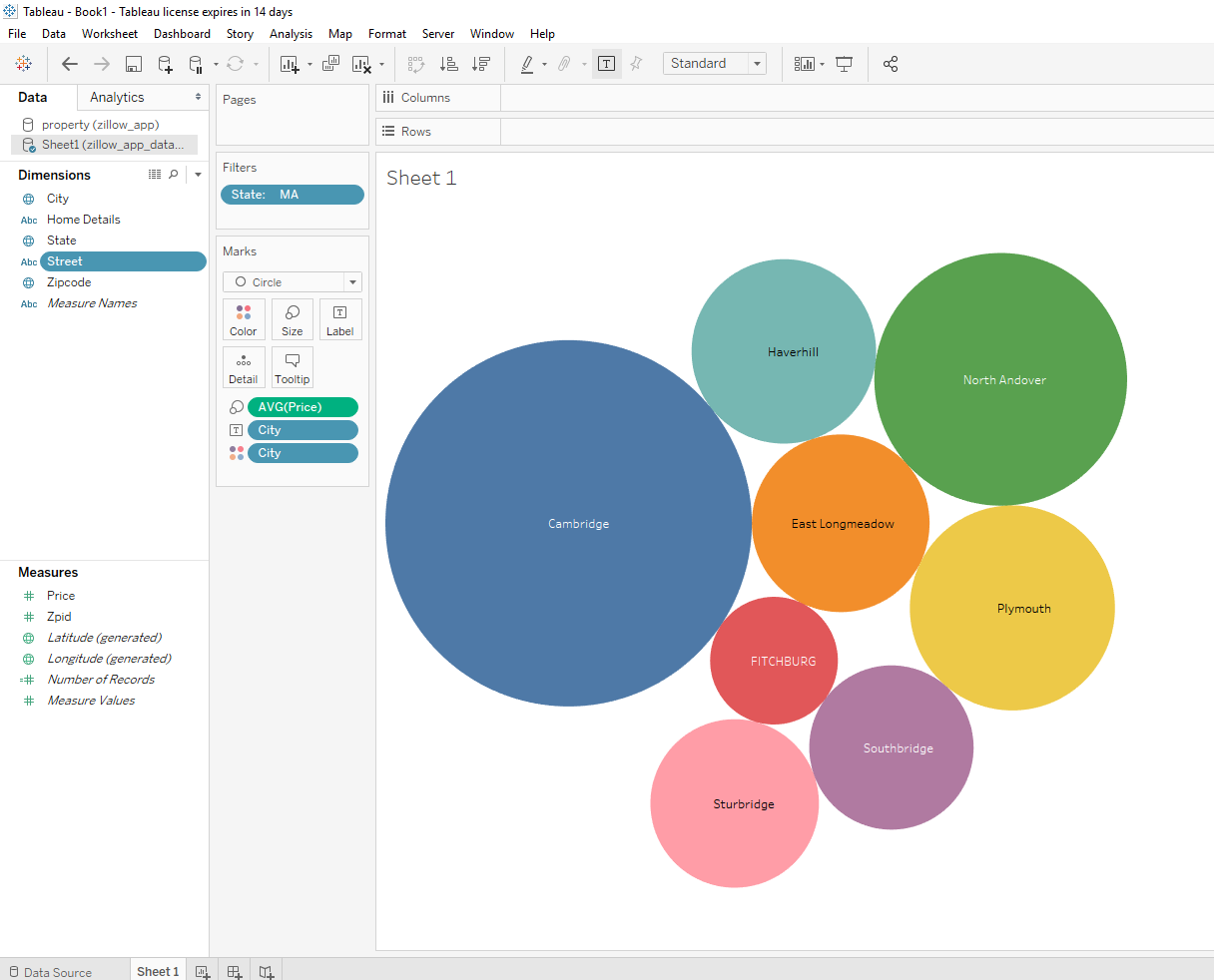


Tableau also allows user to change the visualization such as using a map which will display dots of where all the properties are – when you click on one it will display the price.



Other chart types can if a good picture of which regions have the highest priced real estate.



As a result of project, I was able to view the data in Tableau for further analysis. One lesson learned I can take away from this project was that I hadn’t known Tableau wasn’t available for the Unix platform. As such, my workaround was to extract the dataset and upload into tableau. The plus of this is that it doesn’t consume resources that may be needed, there will be reduced latency, and if the architecture goes down – users won’t be impacted. However the downside is that have the connectivity to the database would have been ideal for data refreshes.