Carlson School of Management
MSBA 6320: Data Management, Databases and Data Warehousing
Professor De Liu

Location, Location, Location

Dimensional data warehouse designed for conducting analysis on the best locations to live for data scientists

---

December 8, 2014

Prepared By Team 1:

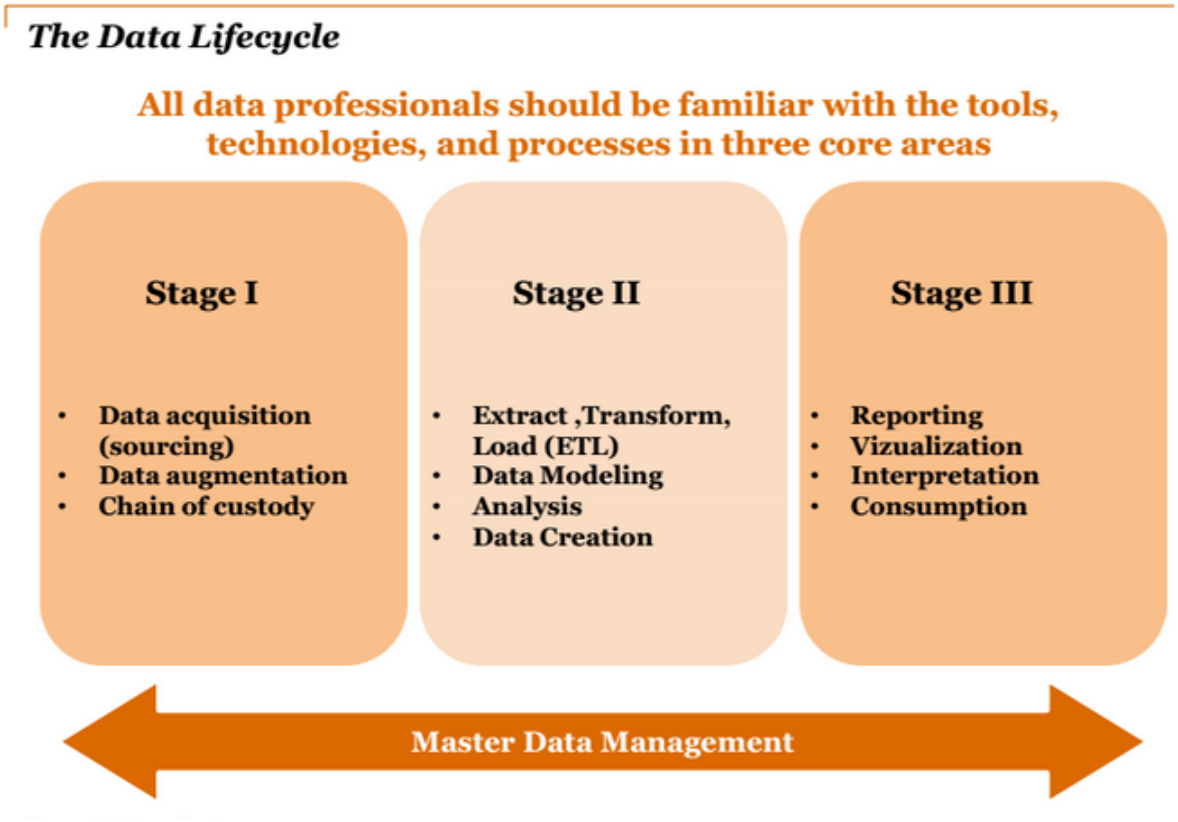
Bharadhwaj Anand
Hamza Farooq
Shravan Shetty
Yang Shi
Kyle Kelly

Table of Contents

Introduction

Project Background

Business analytics can be described most simply as the process of turning data into useful information, and drawing insights from that information. In Tim Michael's guest lecture to our MSBA 6320 Data Management course, he presented a simple Master Data Management spectrum that encapsulates most of the business analytics data lifecycle:



Source: Tim Michaels, PwC Director on Data Warehousing, U of M Guest Lecture for MSBA 6320 on 17 Nov. 2014

Often times, much of the focus is placed on Stage III of the data management spectrum. Businesses care more about the actual use of the data or information--through reporting, visualization, interpretation, or consumption--to create additional value, and rightly so. However, it is also important to consider Stage I and Stage II of the data management spectrum, which focuses on cleaning, structuring and organizing data. Solid data structure and organization will lead to more robust analytical capabilities, which can create far superior value.

This project focuses more Stage I and Stage II of the data management spectrum displayed above. Our team was tasked with designing and implementing a data warehouse that

considers all of the concepts learned in the MSBA 6320 Data Management course. While we considered all Stages of the data management spectrum in our report, our overall goal was to design and implement a data warehouse flexible enough to handle different visualization, reporting, and interpretation methods that future data scientists may wish to employ.

Objectives and Business Requirements

Given the nature of this course and the general makeup of the students in the MSBA program, we decided to focus our objectives around property listing information that can be more useful for young professionals looking for careers in the tech industry. Therefore, our business requirements necessitated designing a flexible yet robust data warehouse capable of handling queries that can provide insights into how property listings differ throughout the United States.

More specifically, our business requirements were characterized by questions that the customers of our data warehouse would ask. As stated before, we defined our project's customer to be a young professional that has recently graduated and is either beginning or continuing a career in the technology industry. This individual has likely been offered work by a particular company located in one of the major tech centers in the United States. As such, our customer is now evaluating which locations represent the best value for more permanent residence, and would like to use our data warehouse to research their options. We decided to focus specifically on six states in the United States that have a high number of technology-related companies including: California, Washington, Minnesota, Illinois, New York, and Massachusetts.

Given our defined customer segment, our data warehouse seeks to provide useful information on the most current property listings in the given states. Since we assumed that our customer is now in the market for buying or renting a property, we only considered posted listings that are currently on the market. Our customer will likely ask questions that involve descriptions of the property and how that property relates to other similar properties in terms of price, location, size, and features. However, we also assumed that the most useful information for a particular property listing will be the listing price, and how the property or location scores in terms of availability of public transportation, schools, and the ability to walk to many different popular community areas.

With these objectives and business requirements in mind, we worked to develop a data warehouse that was capable of storing property listing information provided publicly on the web. While there are many different realtor and property listing sites available to both the private and public sectors, we decided to design our data warehouse to reflect data from the popular listing site Zillow.com to ensure consistency and simplicity.

Data Warehouse Design

Source Data

Before designing our data warehouse, we decided on a singular source that would provide the initial dataset that would be used to both guide our design and to populate our data warehouse. As stated previously, given our customer definition and business requirements, we determined that Zillow.com both offered the most accurate and relevant source data and also provided the easiest access to data. Given that the data offered on Zillow.com is displayed through a well-developed and user-friendly UI, we determined that the best way to capture the source data would be through scraping relevant pages with a customized python script.

We used the BeautifulSoup python package to easily traverse the web pages and collect relevant data. Since Zillow structures their site through a series of pagination-based hierarchies, we developed a simple looping function that fetches 1000 listings from each search result by state. For additional simplicity, we only scraped the top 1000 pages for each of the six states we determined in our initial business requirements. The following is the snippet of the python code that paginates through each search result to fetch the data:

```
#Search Result link
link='http://www.zillow.com/homes/for_sale/IL/fsba,fsbo,fore_lt/pmf,pf_pt/21_ri
d/44.229457,-81.309814,35.065973,-97.701416_rect/5_zm/{}_p/'
pgnum=1
while pgnum:
    #Scrape the search results
    if scrape_data(init_soup(link.format(pgnum))) is not None:
        pgnum+=1
        print pgnum
        my_sleep()
```

```
    else:
        break
    #scrape till 40 pages
    if pgnum==40:
        break
```

We were left with over 5000 records in our sample dataset that represented the most current property listings in the six states previously mentioned.

The Grain and Facts

After collecting our initial data set from Zillow.com, we carefully examined the source data keeping in mind our business requirements and questions for analysis to design our dimensional model. The first step was to use the business requirements to determine the facts and grain of the model.

As our business requirements were centered on the property listing price and scores associated with the relative location, we determined that the grain of the model would represent a single new listing on Zillow.com. This assumed that our project's customers would be more likely to make a higher number of queries surrounding individual property listings, or aggregated queries on a number of current listings.

GRAIN: A single instance of a current property, apartment, or house listing on Zillow.com.

With the grain of our model representing an instance of an individual property, apartment, or house listing, our facts needed to reflect the most important and relevant attributes that would describe that particular instance. We determined that listing price would present itself as the most valuable fact, and that three attributes (as defined by Zillow) would also represent facts that further describe the value of the listing. These attributes--walk score, transit score, and school score--are derived entirely by proprietary equations on Zillow.com that represent the additional value of a property in relation to its location at the time of the listing.

FACTS: Listing price, Walk Score, Transit Score, School Score

The Dimensions

After determining the grain and the facts that characterize our dimensional model, we needed to decide what dimensions would help describe an individual instance of a property listing. We considered the data carefully and drew out three specific dimensions necessary to intersect with our fact table. Each individual listing on Zillow.com is characterized by a location, realtor, and a property or house itself.

DIMENSION 1: Location

The location dimension holds all information associated with a listings exact location in the world. The source data provides a listing's address according to address standards in the United States, so we broke the address into several useful attributes including Street, City, Zip Code, and State, to allow for drilling down into property listings according to location.

DIMENSION 2: Realtor

The realtor dimension solely provides data on an individual realty company, when provided. Given the nature of our customer and business requirements, we assumed that there would not be much analysis on how specific realtors may relate to property listings. We decided that realty company name would be sufficient for any further analysis. Our data suggests that realty companies, rather than individual realtors, are more likely to make use of Zillow's functionality for posting information.

DIMENSION 3: House

Our final and most robust dimension describes all the major tangible characteristics associated with the property itself. We immediately recognized this dimension as a slowly changing dimension (SCD) because properties may be listed more than once on Zillow and they can occasionally undergo changes or remodels that would need to be recorded historically. We designed this dimension as a Type 1 and Type 2 SCD, with most of the historical attributes representing the tangible characteristics of the property itself such as number of bedrooms or bathrooms, square footage etc. Type 1 changes would be recorded as error changes, and do not require historical tracking. However, Type 2 changes require historical records, so we added

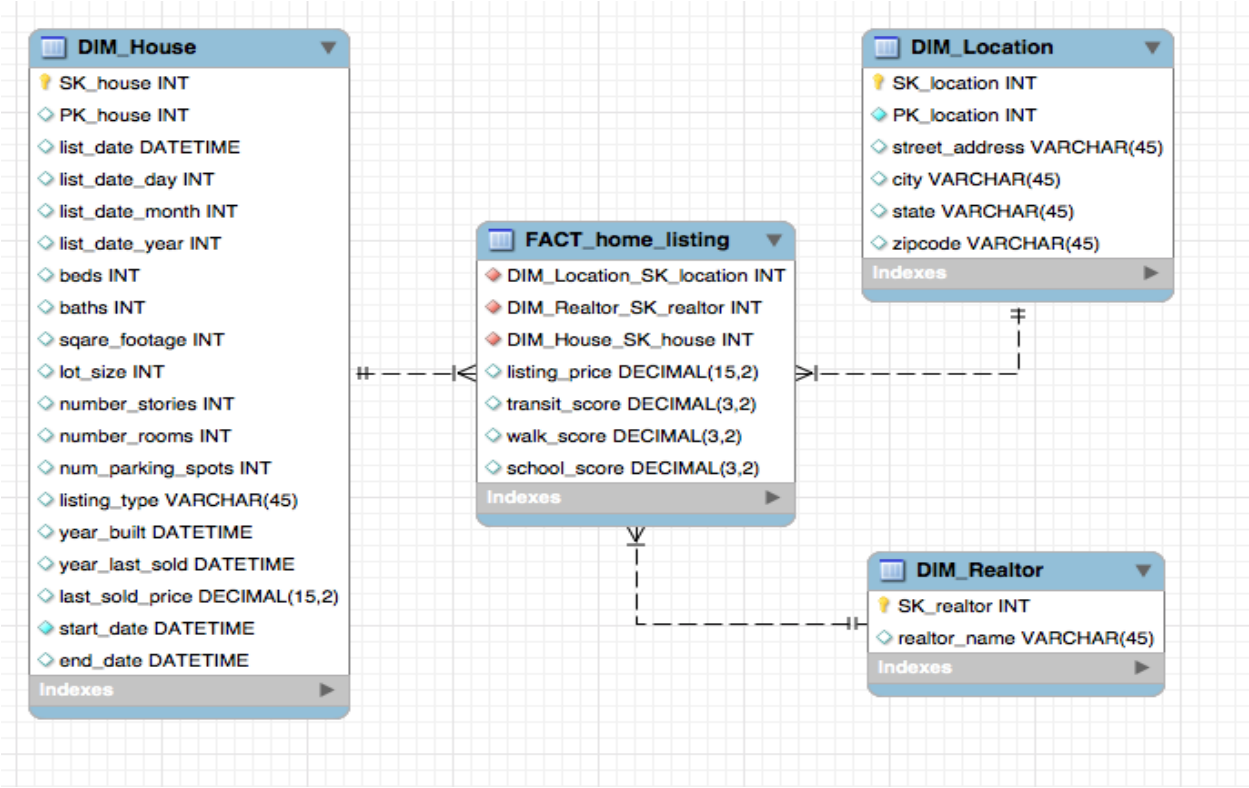
additional time columns in the form of "Start Date" and "End Date," with the most recent record having a NULL value in the "End Date" field.

Since our initial data set represents all current listings on Zillow relevant to our target area, there are no instances of historical changes to this dimension at the moment. But we have ensured the design to be capable of logging these changes in the future, and they will be accurately represented by the combination of Primary and Surrogate Keys with the Type 2 historical fields and additional "Start Date" and "End Date" fields.

This dimension also contains all relevant data associated with the point in time when the property is listed on Zillow. Unfortunately, the only data provided on Zillow.com that relates to time was presented as "Days on Zillow," an attribute that represnts the number of days that a property has been listed on Zillow's webiste. We needed to work backwards to determine the actual list date and derive the time dimension from that datum. So it is important to note that the temporal attributes may not accurately represent *exactly* how long a property has been listed on the market, but rather, the time that a property has been listed on Zillow.com.

The Dimensional Model

Our dimensional model is outlined below, and coincides with the previous definitions of our Grain, Fact, and Dimension Tables:



Design Issues

While it can be relatively easy to design a theoretical dimensional model, there are challenges associated with translating technical theory to practice. Many of the stumbling blocks we encountered were often simple data type inconsistencies or problems with the clarity and completeness of the source data, but we eventually discovered a few design issues that were previously unforeseen.

Initially, we had designed our Location dimension to be a slowly changing dimension as well. We had incorporated several attributes into the Location dimension that are now in the fact table including walk score, transit score, and school score. As we carried out our implementation of the design, we quickly realized that these proprietary scores were not associated with a broader location so much as they were determined at the exact instance of a property listing. As such, we quickly re-designed our model and added those listing characteristics as facts, and removed them from the location dimension. We then determined that there would be no reason to keep historical information on location attributes, since those attributes should theoretically never change.

We also encountered a second fairly significant design issue when trying to implement one of our first dimensional models. We had initially planned on creating a fourth dimension to capture elements of time associated with when a property was listed on Zillow. However, this dimension was derived entirely from a single field in the source dataset, and proved to be less than useful. We found it more appropriate to apply derived temporal attributes to our house dimension. Addressing this design issue eliminated many of the logical errors we encountered when attempting to execute data flow tasks and link a dimension to non-existent base table attributes.

Apart from those design changes, most of our problems proved to be logical errors that occurred when implementing the actual ETL process. The next section of the report details our process for extracting, transforming, and loading the data, and some of the challenges we faced during this portion of the project.

Implementation

ETL Process Overview

We scraped the data from zillow website where in we got necessary information from the search results. Since Zillow.com has pagination, we had to develop script that will scrape each every list item in that page and get following features:

House Features: cooling, heater, deck, porch, security, dishwasher, laundry etc

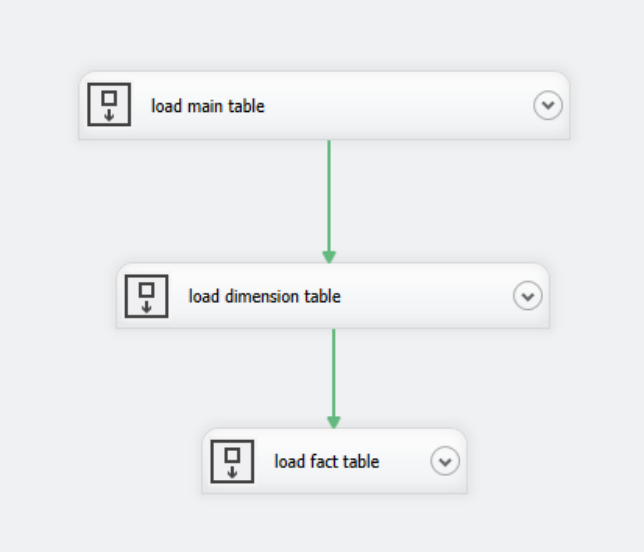
House Size: lot size, area (sq ft), number of bedroom, and bathroom, parking

Price, Last sold and z-estimate

Location Factors: Transit Score, Walk score, School Score

All these data were stored in the csv file using our previously described python script.

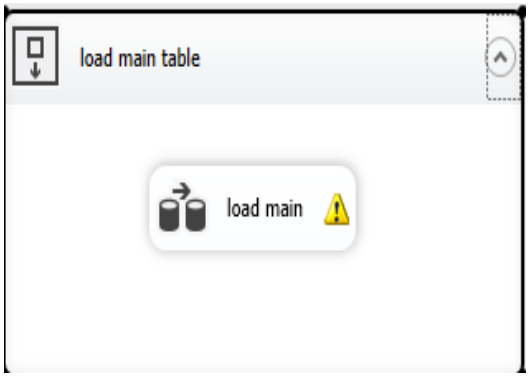
At a high level, our entire ETL process flow follows a step wise structure starting with loading the base table, followed by loading the dimension tables from the base table, and finally loading the fact table from the dimension tables and the base table. The following diagram outlines this process:

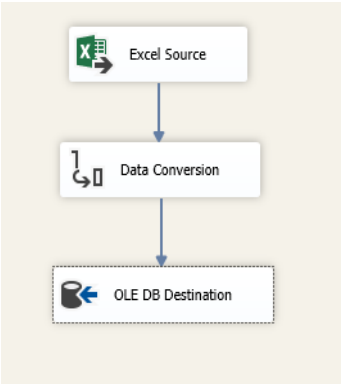


Step 1: Creating and Populating the Base Table

After dumping our scraped data into a simple csv file in excel, we decided to create a large base table as a staging area for the SQL Server Integration Services (SSIS) platform ETL process. We structured the base table to look exactly like the flat csv file and set up a simple data flow procedure that would convert the data into necessary formats.

To populate base table we dumped the data from csv and converted it into one standard table by copying the csv file into single table called base_main. We performed necessary data conversion to maintain proper size constraints and mitigate truncation errors. Following is the diagram of the procedure we used:





We encountered difficulty in storing the data from csv to SQL server tables, as we had to make sure that the size of each data type directly sourced from the excel file was similar to that of the tables. In doing so, we needed to conduct several different unicode conversions to comply with SQL server datatype protocols.

Step 2: Creating the Fact Table and Dimension Tables

Structuring Non SCD Dimensions: The fact table is mainly the combination of the surrogate keys of each dimension table and the facts that incorporate the business question that quantifies each dimension. We made sure to use surrogate key for each dimension table to both incorporate for the possibility of historical records in slowly changing dimensions, and to insure the integrity of the original data through its true primary key. Each dimension table is essentially a mapping of a portion of the associated Base Table.

Load realtor dimension ***Load location dimension***

Structuring SCD dimensions: In this section, we determined the "house" dimension to be a slowly changing dimension table as it denotes the important features of the property that, in turn, directly affect the price of the house. Therefore, it is best to keep track of each and every change made to the house table. To account for these potential changes, we added two columns called "start_date" and "end_date" to keep track of historical instances of a particular house by keeping records in accordance with the properties of a Type 2 SCD. By default the start_date is the day when price is listed on zillow, and end_date is NULL. Any user that wishes to find the most current historical record of a particular house should search for the record that contains a NULL end_date.

Structuring facts: Finally we accounted for the facts: listing price, walk score, transit score, and school score. We determined previously that these facts provide a more complete picture of the overall value of a particular listing. Price is an important factor when choosing house in that it indicates to the buyers the potential range of monetary value needed to make a purchase. In the end, sale prices very rarely match initial list prices, as the actual sale transaction is typically characterized by stages of negotiation in which both sides can redefine value. But for the purposes of this project, our facts were grounded solely in posted listings data, leaving the end user of our data warehouse to make further inferences themselves. These facts do not change often and they contribute to the location and attractiveness of the society.

Step 3: Populating the Dimension Tables

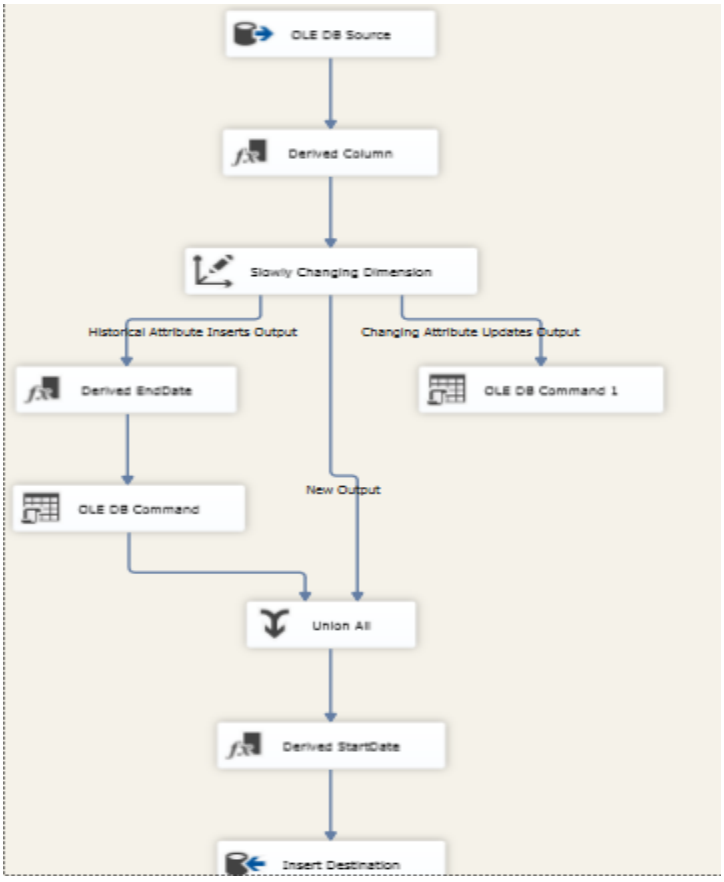
Loading the base table was completed previously in Step 1, so we used the data from that table to populate our dimension tables first. We loaded dimension tables in this generic way for

location, house, and realtor. This process involved mapping each column of the base table with the corresponding columns in our dimension tables.

Loading the House Dimension Table: For the house dimension table we created derived columns from "Days on Zillow" using the DATEADD function:

Derived Column Name	Derived Column	Expression	Data Type	L
day	<add as new column>	DAY(DATEADD("day",-Days_On_Zillow,GETDATE()))	four-byte signed integ...	
month	<add as new column>	MONTH(DATEADD("day",-Days_On_Zillow,GETDAT...	four-byte signed integ...	
year	<add as new column>	YEAR(DATEADD("day",-Days_On_Zillow,GETDATE()))	four-byte signed integ...	
list_date	<add as new column>	DATEADD("day",-Days_On_Zillow,GETDATE())	database timestamp [D...	

As the house dimension is a slowly changing dimension, we needed to incorporate some of the more advanced data flow functionality in SSIS to properly link relevant historical attributes.



As we can see from the diagram above, we used derived column to populate "start_date" column using the list_date from the base table which we created when loading the house

dimension, and then applied to SCD. Following is the list of actions we selected when there is a change in the house record:

Dimension Columns	Change Type
beds	Changing attribute
cooling	Changing attribute
deck	Changing attribute
dishwash	Changing attribute
heating	Changing attribute
last_sold	Historical attribute
laundry	Changing attribute
listing_type	Changing attribute
lot_size	Changing attribute
num_rooms	Historical attribute
num_stories	Historical attribute
parking_spots	Historical attribute
porch	Changing attribute
security	Changing attribute
sq_ft	Changing attribute
year	Fixed attribute

Finally, for any changes in historical attributes, a new record will be created with the old record's end_date column being new_start_date and these values will change automatically. To achieve this, we made sure to do some transformation on the start and end dates. Following is the transformation we did for end_date :

Derived Column Name	Derived Column	Expression
endDate_scd	<add as new ...	(DT_DBDATE)(DATEADD("SECOND",-1,DATEADD("day",-Days_On_Zillow,GETDATE())))

For next start_date:

Derived Column Name	Derived Column	Expression
startDate_scd	<add as new column>	(DT_DBDATE)(DATEADD("day",-Days_On_Zillow,GETDATE()))

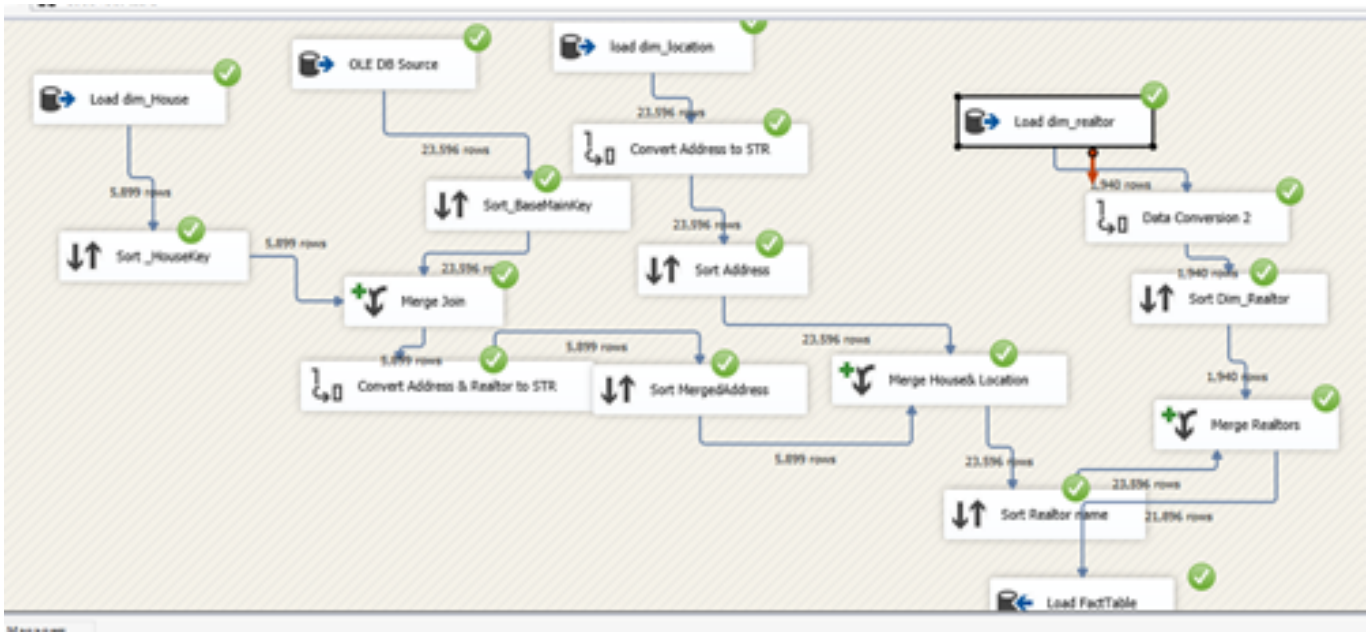
Loading the Location and Realtor Dimension Tables: Since both the realtor and location dimension tables are not slowly changing dimensions, the loading process was relatively

simple. We mapped relevant attributes from the base table to their corresponding dimension tables and completed this process with ease.

Step 4: Populating the Fact Table

Once we had finished loading our dimension tables, we were then able to properly link data and load our fact tables. Since the fact table essentially represents the intersection of our three dimension tables, we needed to implement a fairly complex merging data flow to link all instances that populate the dimension tables with the facts still located within the base table. This proved to be significantly more complex than we anticipated, and required several steps.

Merging Facts: In the fact table, we had to consider merging the base table (which stores all attributes of the fact table) with the dimension tables that store the corresponding surrogate keys.



From the above diagram, house and location have appropriate fact attributes that must loaded into the fact table. Hence we had to do necessary merge by sorting the business attributes of each dimension and base table respectively.

House Dimension Merge: In this process we sorted the house key from the dim_house table and base_main key from the Base Table. Since we loaded house key with base_main key values during dimension loading, each property listing in both tables are unique. We merged the two tables using these keys.

Location Dimension Merge: Before beginning the merge process, we needed to complete a quick data type conversion to ensure that the sorting function would run properly. After converting the address key from DT_Text to DT_STR, we then sorted based on the address key in order to properly merge. We chose address key as our business key as it is unique for both tables and it made the merging process relatively simple.

Realtor Dimension Merge: We needed to perform a similar data conversion task to comply with the sorting function. As the Realtor table had distinct values of realtor, this merge was a simple task.

Methods of Analysis and Findings

Data Visualization

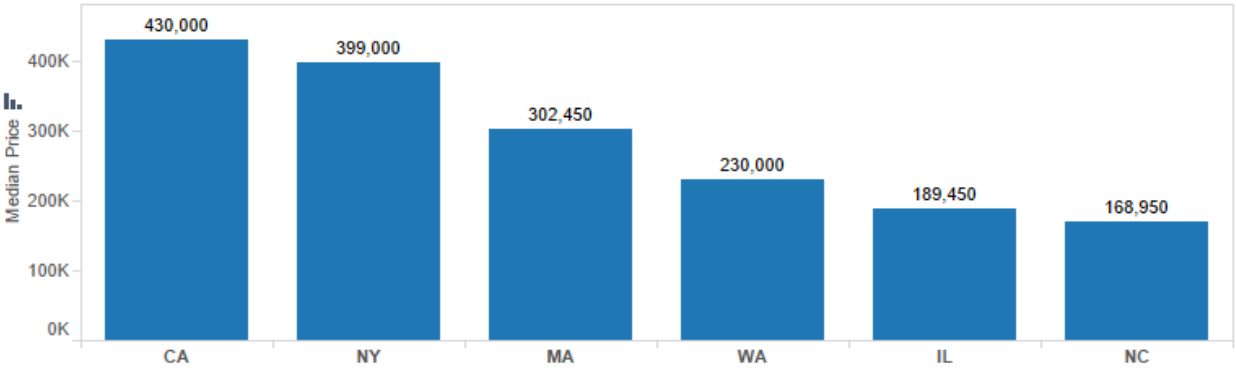
According to the instructions outlined in the project description, we used other tools to conduct data visualization. We completed data visualization tasks throughout the entire course of the project to ensure the validity of the data that we collected, and check results from the data warehouse itself with results produced by the relevant visualization and reporting tools, including Tableau. At the end of this project, we would recommend users to use a more robust visualization and analytics tool to parse through data and answer analytical questions, similar to the dynamic dashboard that we developed through Tableau.

We divided the visualization part in two major areas: first was to delve in to the insights that the data provided and the second was to provide a dashboard for the users to search for houses based on their own requirements.

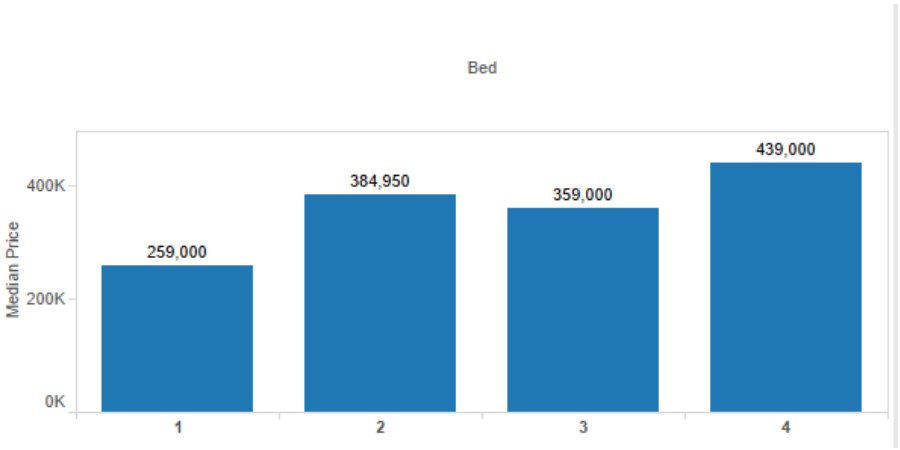
Insights

Some of the interesting facts and insights we found from our data are as follows:

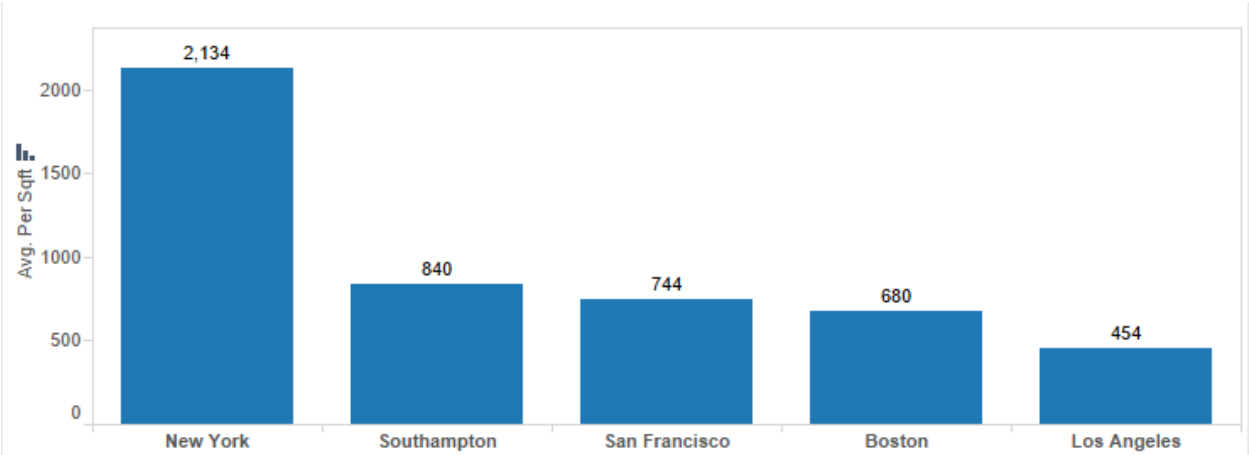
1. California has the most expensive houses for sale



2. In New York the demand for 2 bedrooms is higher than 3 bedrooms. Two bedrooms houses are listed to be $300,000 more than the three bedroom ones.

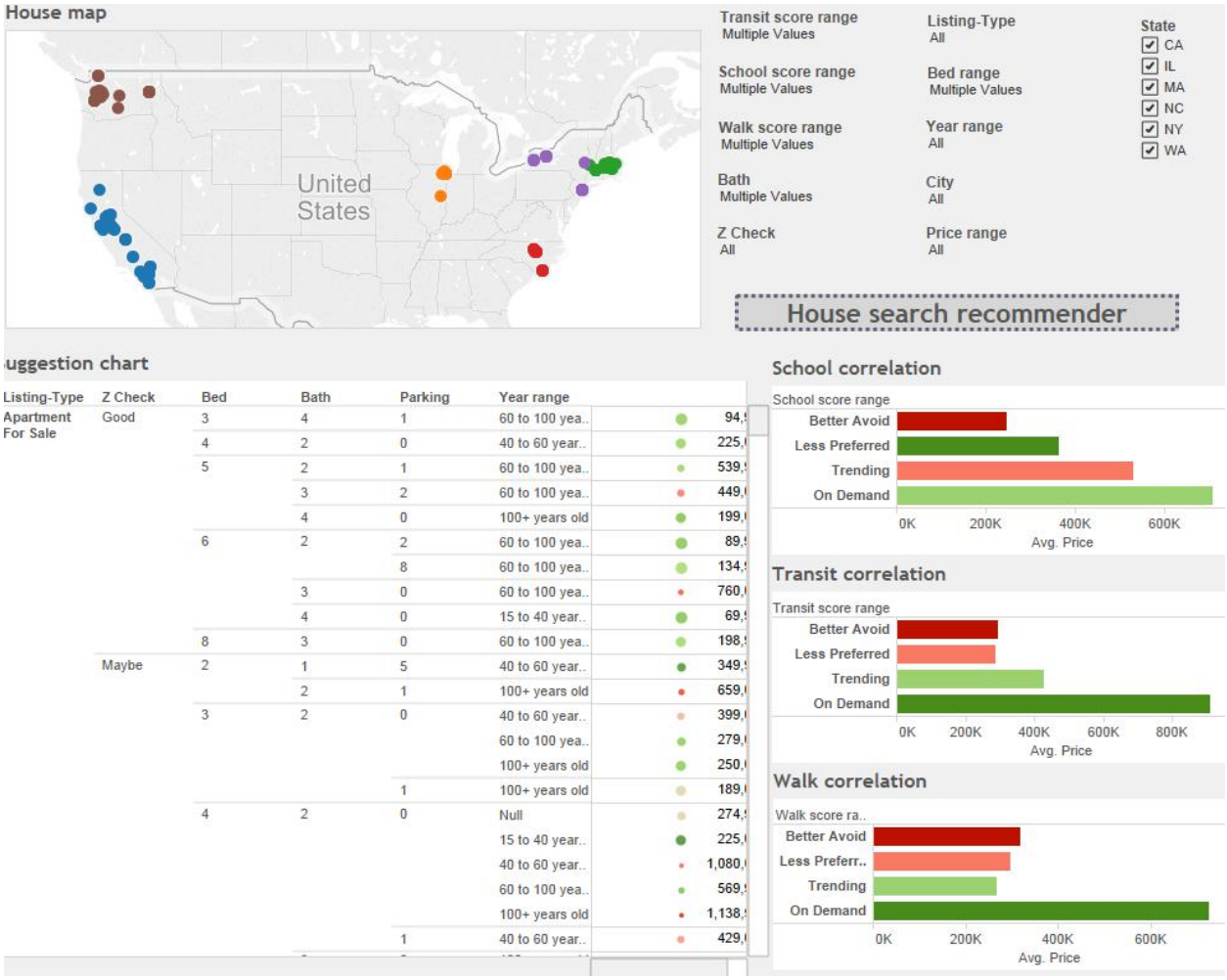


3. Walk Score is so important in New York that median price jumps from 0.5 million to 3.2 million from a walk score of 90 to 99
4. A huge price driver for Massachusetts is transit score, at 70, the average cost is 1.2 million whereas at 90 it's 2.9 million
5. New York has the oldest houses (avg 74) whereas the newest ones are in North Carolina (avg 21)
6. New York city has the highest cost per sq ft and it is more than twice than any other listed city. The top 5 most expensive cities on the basis of Sq ft are as follows:



Dashboard

Users wanting to search for a home according to desired specifications can use this 'House search recommender' dynamic dashboard to easily filter data. The granularity of the details relates directly to the dimensions involved in building the warehouse like House attributes, Demographic filters, Time selections (Realtors information made less business sense for a dashboard reporting). Filters involve every significant attribute for a house (like bed range, price range, built year range, Z check apartment type and baths and parking space available - explanation follows below), demographic selections and other influencing correlated attributes which majorly impacts the price of an house like school correlations where houses were given rating to 10 higher the rating closer the house is to a very good school, transit correlation is given ratings to 100 specifies the accessibility the house has to public transportation, similarly the walk correlation signifies the proximity to popular and on demand areas (like parks, hospitals, downtown, markets, mall etc). The user gets the flexibility to drill down to every available details in helping one finding that perfect home.



<u>Additional info -</u>

New derived attributes are used for the ease of use. Bed range contains options to selecting a range for beds like '1-2', '3-5', '6 and above' giving users options and facilitating comparison at any level. Price range defines the average price of an house '75000+ $' will have houses in range between 50000-100000$ giving budget specific filters options. Year range specifies the age of the house, users may choose between to choose whether they want '21st century' house or '100+ years old house', even if not used as a filter values will be seen available in 'Suggestion chart' to help users make decisions. 'Walk, Transit & School score range' help select preferences for users if one has a family he will be looking forward to selecting 'On demand' ranges from 'School' and 'Walk'. If one doesn't own a car he will more look into 'Transit score' etc.

<u>Dashboard guide:</u>

One rule follows the dashboard - More the Greener more the greener, density of reddishness shows less preferability

Suggestion chart rule - Same rule as above applies, also the size of the circle directly correlates to the goodness of the house. The bigger and greener it is, the better it is.

Conclusion

Learnings

This project gave our group the opportunity to practice almost every aspect of the course curriculum. We were able to successfully piece together many of the different learnings in order to produce a small yet scalable data warehouse much in the same way that we would do so in a traditional business setting. While many of those learnings make sense in theory, there are often unforeseen problems with implementing them together in a cohesive manner. This project brought many of these issues to light and provided our group with the chance to solve additional problems.

One of the biggest learnings from this project is that designing and implementing a data warehouse is an iterative process. We often had to revisit previous designs when we encountered runtime errors in the ETL process. Some of those errors uncovered design flaws that resulted in reworking our dimensional model several times to account for future logical data flows. Likewise, many of our individual data flow tasks would often require the group to revisit previous data flows and evaluate alternative methods to populate tables.

What We Would Do Differently Next Time

If given the opportunity to rework the project from the beginning, the group would have spent much more time developing the business requirements and capturing a better dataset. Business requirements are often somewhat overlooked in data management, but in reality, they can represent the most important aspect of any project. They drive the entire design and implementation process, and if we spent more time fleshing them out, we may have been able to create a more robust dataset.

While datasets are not the sole drivers in dimensional modeling and implementation, they do represent a significant portion of the data management process. Given that we were able to create our own dataset using a scraping script, we could have spent more time using our business requirements to discover additional features. If we worked with a larger and more informative dataset, we may have been able to develop an entirely different dimensional model capable of answering several different business questions.

References

Allen, Annette. "SSIS Basics: Using the Merge Join Transformation." Simple Talk: A Technical Journal and Community Hub from Red Gate. Red Gate, 16 Oct. 2013. Web. 6 Dec. 2014. <https://www.simple-talk.com/sql/ssis/ssis-basics-using-the-merge-join-transformation/>.

Michaels, Tim. “PwC U of M Talk.” Guest Lecture for MSBA 6320: Data Management, Databases, and Data Warehousing. University of Minnesota, Carlson School of Management, Minneapolis. 17 November 2014. In-class lecture.

"What Are Slowly Changing Dimensions?"What Are Slowly Changing Dimensions?Web. 5 Dec. 2014. <http://datawarehouse4u.info/SCD-Slowly-Changing-Dimensions.html>.