where(ii)

W205 Final Project

Prepared by: Lisa Barcelo, Melanie Costello, Priya Gupta

November 22, 2016

EXECUTIVE SUMMARY

Objective

As wages fail to keep pace with the rapidly increasing cost of living in U.S. cities, many young professionals and families often find their paychecks don't quite go far enough once the bills are paid. Many are forced to make tradeoffs, like enduring 90-minute commutes each way, in order to afford the housing or lifestyle they want. Others must consider moving. The where ii project aims to help people improve their quality of life by identifying locations in which they can maximize their income relative to the cost of rent, factoring in time spent in traffic and rental trends over time.

Goals

While a number of companies independently offer great data on housing prices, salaries and traffic, there is no go-to resource that brings together these important elements to help people see the complete picture when considering a move. The goal of where ii is to fill this void.

Project Outline

The where ii project consumes data from a variety of sources, links it together and allows users to identify the best places to move via an interactive Tableau dashboard. This document outlines the technical components of where ii - from data acquisition to user consumption. It will also address key achievements and challenges throughout the following sections:

- Architecture
- Data Acquisition
- Data Cleansing & Transformation
- Application
- Next Steps

ARCHITECTURE

Initial Assessment

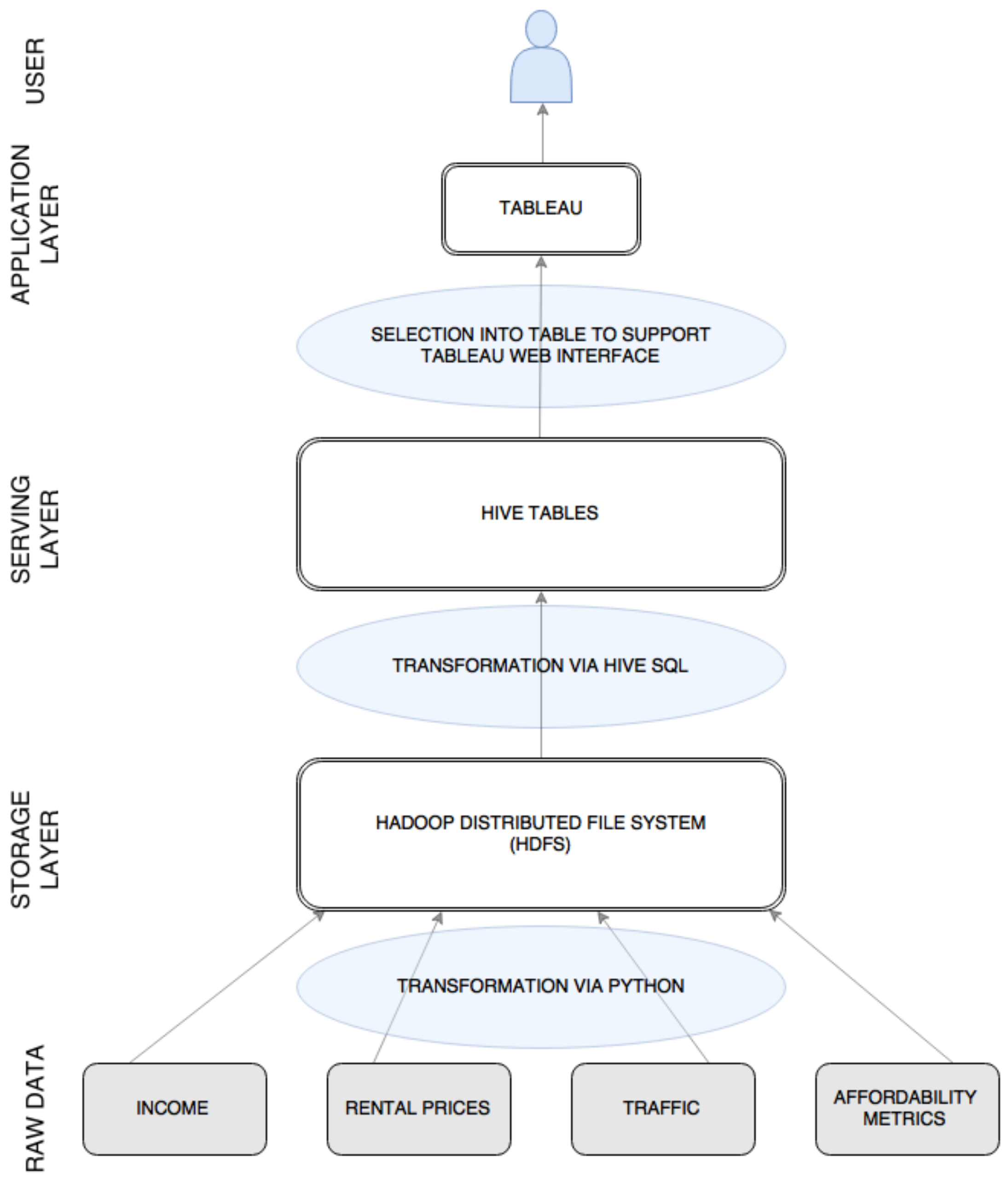
In assessing the best structure for our data storage and retrieval system, we considered the 3 Vs of data science - volume, velocity and variety. Our goals clearly dictated a need for a variety of data with the understanding that the volume would grow over time, likely at a manageable pace.

Those needs, in conjunction with an evaluation of the strengths and previous experience of the team members, resulted in the use of the following tools to build the data storage and retrieval system outlined below.

Tools & Languages

- HDFS
- Hive
- Tableau
- Python/pandas
- SQL

Storage & Retrieval


USER
APPLICATION LAYER
TABLEAU
SELECTION INTO TABLE TO SUPPORT TABLEAU WEB INTERFACE
SERVING LAYER
HIVE TABLES
TRANSFORMATION VIA HIVE SQL
STORAGE LAYER
HADOOP DISTRIBUTED FILE SYSTEM (HDFS)
TRANSFORMATION VIA PYTHON
RAW DATA
INCOME
RENTAL PRICES
TRAFFIC
AFFORDABILITY METRICS

RAW DATA

The where ii project compiles data regarding income, traffic, rental pricing and affordability from the following sources:

Description	Source
Income	U.S. Census
Traffic	Texas A&M University
Rental Prices	Zillow
Housing Affordability	Zillow

STORAGE LAYER

All data sources are updated on a monthly or quarterly basis, so we felt the Hadoop Distributed File System operating on a 100GB volume attached to an m3.large instance on Amazon Web Services would be an appropriate storage solution. Eventually, we will need to add storage capacity if we do not want to purge data.

SERVING LAYER

A traditional RDBMS approach suits the needs of this project for several reasons. All data sources, except for traffic, already contained unique identifiers that could be used as join keys with the help of a mapping table. The raw data sources were also already topic-specific, so the transition of each source into its own table was a natural move.

We elected to use Hive to create tables. Our shared comfortability with Hive and SQL gave us confidence that we would be able to effectively troubleshoot any problems. We also felt we would be well-positioned to try new techniques and functions we hadn't previously implemented in an academic or professional setting.

APPLICATION LAYER

One of the most important pieces of this project is the user experience. We wanted to equip the user with a fun, interactive way to explore and engage with the insights derived from our data. The user needs to easily be able to modify queries without having to write code. Tableau fit the bill.

With the installation of an ODBC driver, connecting Tableau to Hive is relatively straightforward. (LISA, CAN YOU WRITE SOMETHING HERE THAT TALKS ABOUT THE TABLE WE CREATED IN TABLEAU?)

DATA ACQUISITION

Income

Acquiring a robust set of data on income specific to occupation *and* specific to location proved challenging. The website glassdoor.com, which allows users to submit their position, company and salary, came to mind first. However, the rules governing Glassdoor's API specifically say income data cannot be scraped and utilized for other purposes. As we looked for other public datasets, we found many that aggregated average salaries by occupation, but they were not location specific. We also found datasets containing average salaries for different metro areas, but they were not broken down by occupation.

We settled on a dataset provided by the U.S. Census Bureau that groups salary- and employment-related data by Core Based Statistical Area (CBSA). Each CBSA is assigned a numeric code as an identifier.

(INSERT SAMPLE RECORD HERE)

Rental Prices

The real estate website zillow.com provides abundant data on rental and housing prices, broken down by metro area. Each metro area is assigned a numeric code as an identifier. These identifiers are unique to Zillow and are not the same as the CBSA codes found in the income data.

We decided to work with data specific to the cost of rent in each metro area, with the thought that our end user was likely to be a young professional or recent graduate who likely has not purchased a home. The Zillow rental data is updated quarterly, and it is summarized by metro area and by the following housing types:

- Studio
- 1 Bedroom
- 2 Bedroom
- 3 Bedroom
- 4 Bedroom
- 5+ Bedroom

The available data stretches back to the 1970s, but the fields are noticeably more populated in the last 5-7 years. We chose to incorporate all data, so we could compute changes in rent over time and identify cities in which rent is rapidly increasing.

Affordability Metrics

This dataset also comes from Zillow and contains quarterly data stretching back several decades. It contains estimates for the affordability of each metro region, based on median rent prices as they relate to median household income. Zillow's household income data is also provided by the U.S. Census Bureau, so we reasoned it would integrate nicely with the census income data we acquired.

Traffic

In assessing the key factors that influence quality of life, we felt traffic is becoming an increasingly important factor in one's decisions about where to work and live. In addition to maximizing income relative to the cost of housing, the opportunity to reduce one's commute time could strongly affect a decision about where to move.

We reviewed several sources of traffic data and settled on the dataset provided by Texas A&M University. While this dataset does not have unique identifiers that align with our other data sources, it was broken down by cities in a way that would allow us to link it with the rest of the data.

DATA CLEANSING & TRANSFORMATION

Objective

Our data cleansing efforts needed to achieve the following objectives:

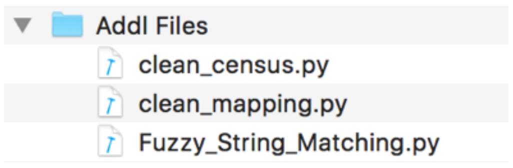
- Connect data sources with different unique identifiers
- Cleanse each data source, removing “dirty data” - such as the use of “#” instead of blank values
- Reshape some raw data into a structure more useful for our RDBMS design
- Normalize data to reduce redundancy
- Add columns to support table partitioning and query optimization

Implementation

After evaluating our objectives, we determined that we needed to use both Python and SQL to reach the desired result. Thus, we split the cleansing and transformation into two layers. As outlined in the architectural diagram on page 3 of this document, the first layer of transformation happens via Python and SQL in the DDL process. The second happens via SQL only in the ETL process.

DDL/PYTHON LAYER

Python Code References: Final_Proj_MLP/Addl_Files
REPLACE THIS SCREENGRAB



```
Addl Files
    clean_census.py
    clean_mapping.py
    Fuzzy_String_Matching.py
```

The Python scripts execute after raw data is retrieved, and the output of those Python scripts is what gets loaded to HDFS. These scripts do basic cleaning (removing header rows, etc.) and some more advanced transformation. Below is an outline of some notable achievements of these scripts:

Script	Achievement
clean_mapping.py	Custom string comparison function to connect CBSA codes from income data to metro ID from Zillow data sources
traffic_clean_transform.py	Implements edit distance function to align cities in Texas A&M traffic data with Zillow metro IDs.
zillow_process.py	Transposes raw data from Zillow, merging multiple columns of rental prices into one column with an additional column to reference the date period.

Example

For example, the zillow_process.py file takes a record that looks something like this:

metro_id	metro_name	beds	2015_Q 1	2015_Q 2	2015_Q 3	2015_Q 4	…
337464	New York, NY	1bed	2895	3011	3058	3100	

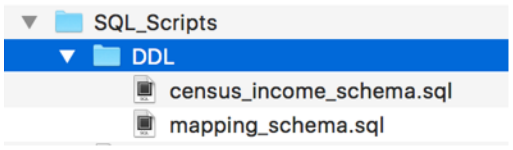
And transforms it to look like this:

metro_id	beds	date	price
337464	1bed	2015_Q1	2895
337464	1bed	2015_Q2	3011
337464	1bed	2015_Q3	3058
337464	1bed	2015_Q4	3100

After these transformations via Python, the SQL for the DDL process is relatively straightforward. We specify data types and pull in all columns to create base tables from which we will implement the ETL.

DDL SQL Code References: Final_Proj_MLP/SQL_Scripts/DDL

REPLACE THIS SCREENGRAB



```
SQL_Scripts
  DDL
    census_income_schema.sql
    mapping_schema.sql
```

ETL/SQL LAYER

ETL SQL Code References: Final_Proj_MLP/SQL_Scripts/ETL

INSERT SCREENGRAB HERE

The ETL process incorporates some additional cleaning steps and the implementation of additional columns and functionality to support the queries we know will run in the application layer. Notable achievements in this process include the following:

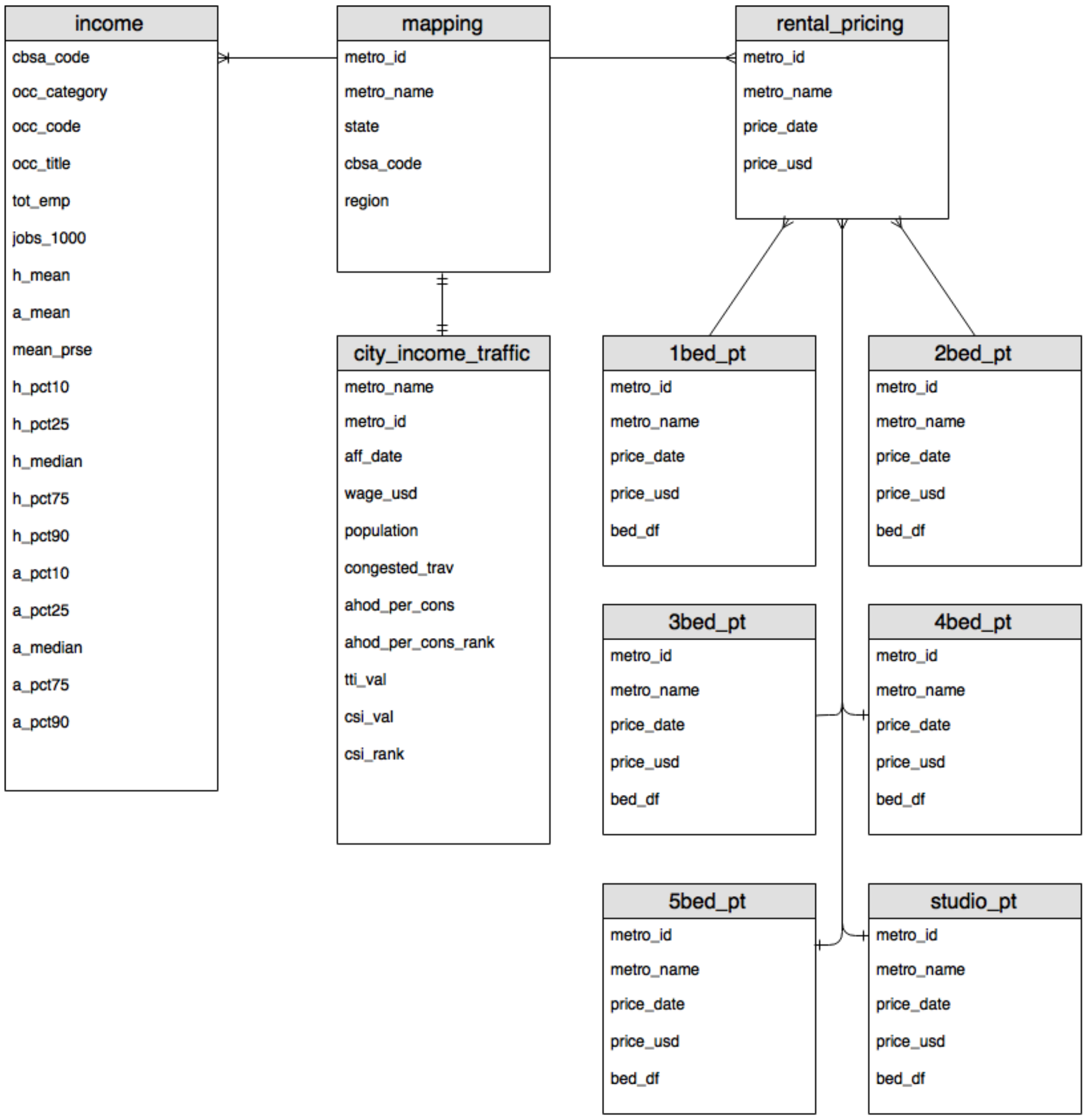
- Partitioning the rental price data by number of bedrooms.
  - One of the first prompts in the application layer will ask the user for the number of bedrooms needed. Partitioning on this field will allow us to speed up the query of the rental data.

- Adding a region column to the mapping table.
  - Creating supersets of the metro IDs will allow our queries to accommodate users looking to move to a specific part of the country (Northeast, Southwest, etc.).

- Standardizing missing values across data sources.
  - Some data sources used "**" or "#" instead of NULL for fields with missing data, so we standardized this.

- Paring down the number of columns in each table.
  - We pulled in only columns we felt would be useful in queries performed by the end user.

The result of our ETL process is a relational database that implements the following entity relationship diagram:


income
cbsa_code
occ_category
occ_code
occ_title
tot_emp
jobs_1000
h_mean
a_mean
mean_prse
h_pct10
h_pct25
h_median
h_pct75
h_pct90
a_pct10
a_pct25
a_median
a_pct75
a_pct90
mapping
metro_id
metro_name
state
cbsa_code
region
rental_pricing
metro_id
metro_name
price_date
price_usd
city_income_traffic
metro_name
metro_id
aff_date
wage_usd
population
congested_trav
ahod_per_cons
ahod_per_cons_rank
tti_val
csi_val
csi_rank
1bed_pt
metro_id
metro_name
price_date
price_usd
bed_df
2bed_pt
metro_id
metro_name
price_date
price_usd
bed_df
3bed_pt
metro_id
metro_name
price_date
price_usd
bed_df
4bed_pt
metro_id
metro_name
price_date
price_usd
bed_df
5bed_pt
metro_id
metro_name
price_date
price_usd
bed_df
studio_pt
metro_id
metro_name
price_date
price_usd
bed_df

APPLICATION

(NEED TO COMPLETE THIS ONCE TABLEAU COMPONENT IS UP & RUNNING - SHOULD INCLUDE A COUPLE TABLEAU SCREENGRABS)

NEXT STEPS

Batch Updates

Based on continued use of existing data sources, the where ii database will not require real time updates, only batch updates executed on a quarterly basis.

The Zillow files are updated quarterly and each update by Zillow appends new data to the existing data file. Simply put, Zillow does the hard part. A simply wget of the file link and re-running the python and hive DDL scripts will pull the new data into the existing table. The python scripts created for pre-processing are designed to take inputs instead of calling specific files, therefore, this method helps scale the processing process. As the Zillow files scale in the future, we need to create scripts to pull the new data out of the Zillow files and update the current zillow tables.

The Bureau of Labor and Statistics census income data is updated annually in March. Similar to how we processed the current income data, we will wget the latest file link and run it against the existing DDL scripts. We will add YEAR partition to the income table and will use a max function to ensure we are utilizing the latest data while still maintaining historical data for reference.

The traffic data from Texas A&M is updated annually and will also require a partition to update the current traffic table. Fortunately, utilizing the traffic_clean.py to clean the data and parsing it can be done at the command line by inputting the updated file.

The current process to clean, load, and create tables from external files is scalable. In the future, there may need to be few changes to compensate for additional partitions and improve the scalability of the process, but we do not foresee the current architecture as being an issue in the coming year. Even so, we have planned carefully to incorporate these changes when such changes become necessary.

Future Opportunities

Where(ii) is a model that bridges together several data sources on income by occupation, rental prices by metro region, and even traffic to better inform consumers of all their options. Where(ii)'s scalable implementation and architecture creates a baseline model that can be further expanded on. Here are some suggestions on future opportunities where(ii) can be expanded to address the following:

1) Incorporate data from sources that could include:

- FBI.gov : Crime Data
- National Center of Education Statistics: School Data
- Census: Population demographics (diversity etc)
- National Centers for Environmental Information: Weather/Climate data

2) Implement machine learning to predicting income based on gender, age, ethnicity, education, and occupation type. This data is available through the U.S. Bureau of Labor Statistics

3) Deep diving into the cost of living in specific cities of a metropolitan areas. Zillow data provides a more granular breakdown by city. For example, those wanting to move the SF Bay Area, this granularity will help them decide which city they get the most for their money.

4) For those wishing to travel outside of the country, we could incorporate statistics and information regarding other countries that compare to the US. This includes country GDP, average price per square foot, average cost of living, and even happiness -- data presented by worldhappiness.report.