**Opening High-End Restaurant in Houston**

**(ETL Project Report)**

Diana Batten & Charles Dixon

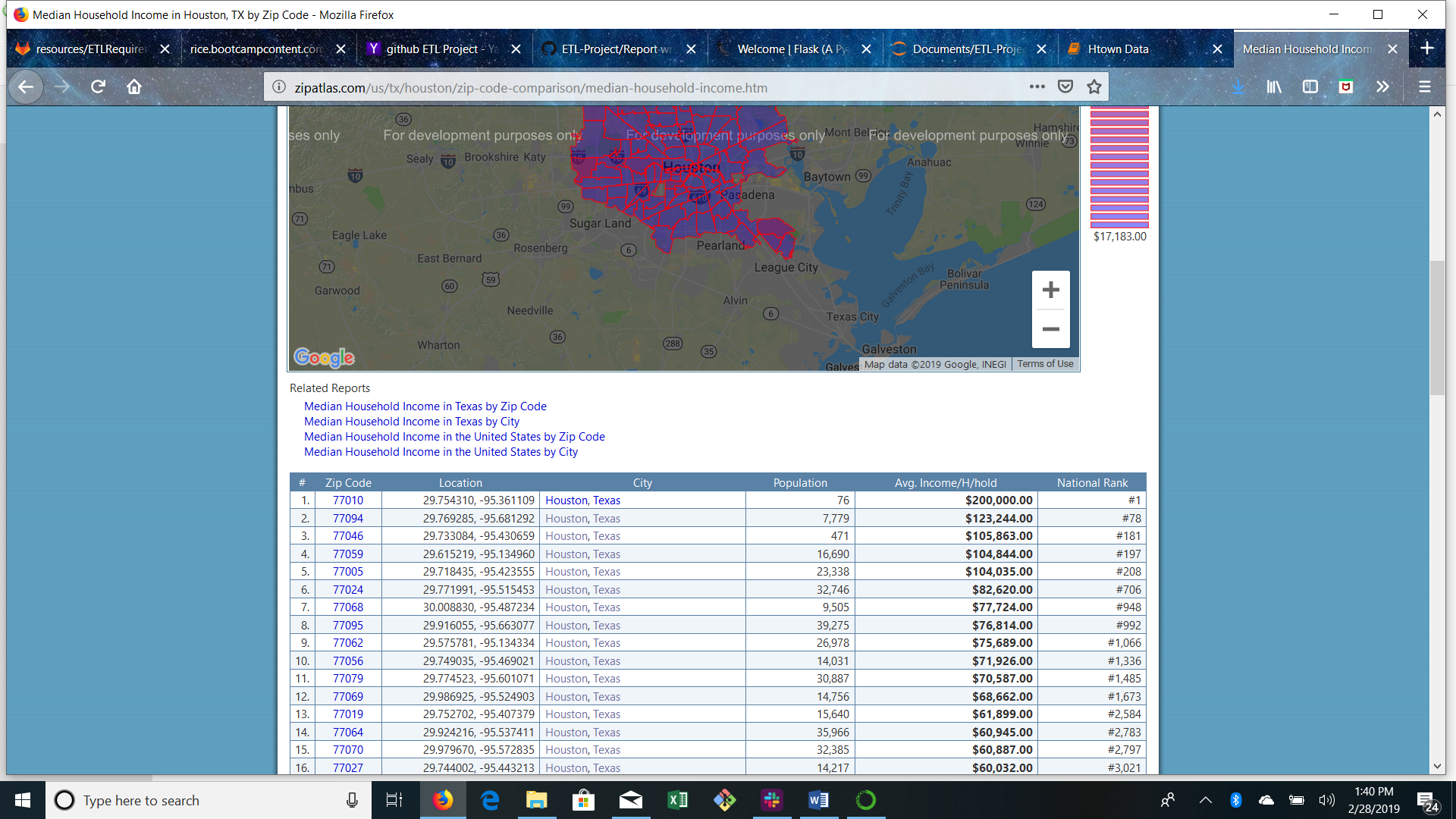
**The Data Sources and Why**

1. Yelp Developers Website (https://www.yelp.com/developers) -- We chose the website to build an **API** to gather data about Houston area restaurants.
2. Zip Atlas Website (<http://zipatlas.com/>) – We chose this website to obtain demographic data about the Houston, Texas area. The site presented us with the opportunity to **scrape** population, average salary, and zip code information from one spot.
3. Data.Gov “Real Estate Across the United States” (<https://catalog.data.gov/dataset/real-estate-across-the-united-states-rexus-lease> )– We chose the website to obtain commercial real estate data for the United States. The site allowed us to download a **CSV** file with commercial real estate properties, costs, zip codes and locations.

**The Process of Extraction and Transformation**

We used three data sources in our ETL Project.

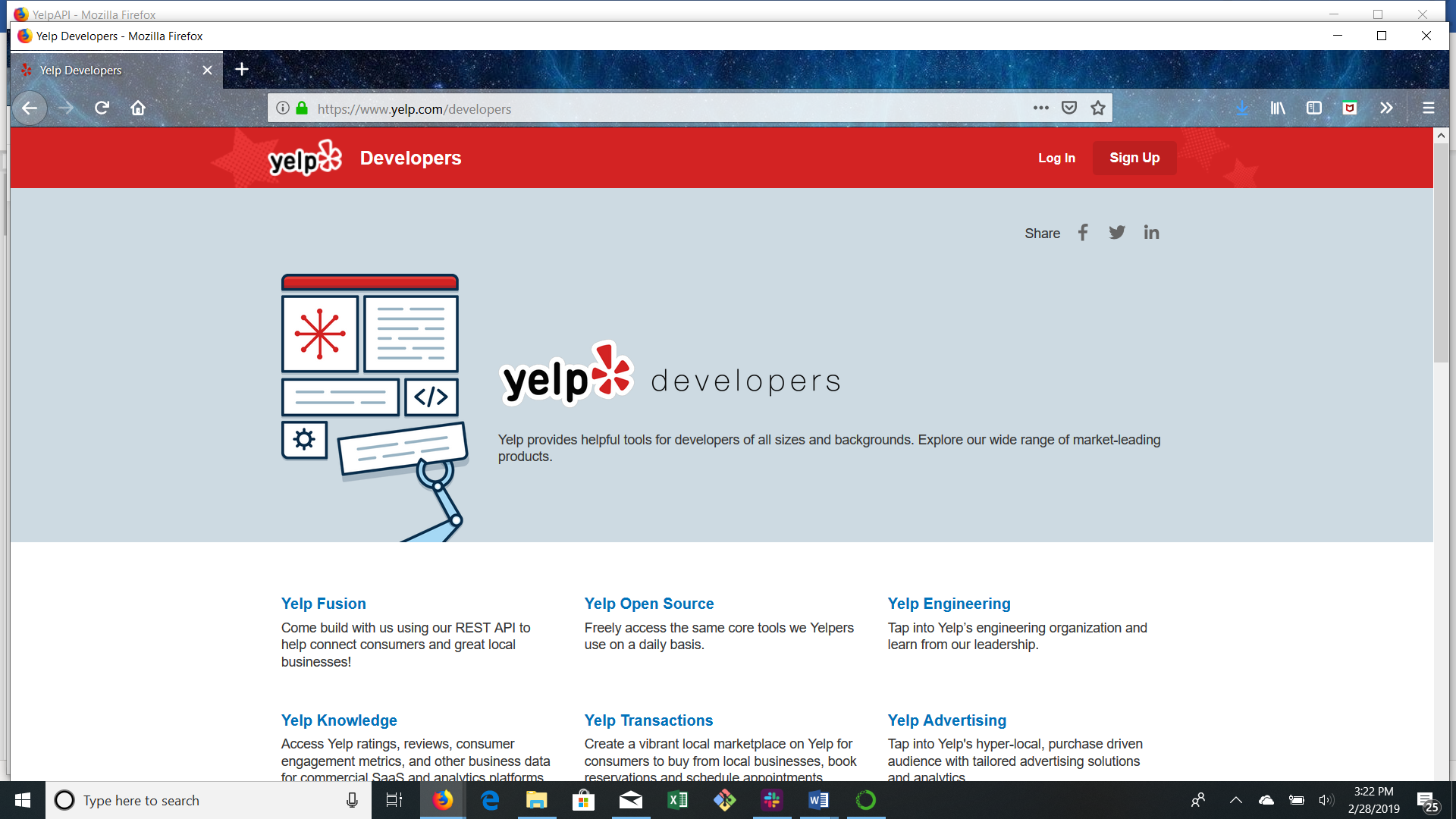
1. We did a web scraping of Houston, Texas fact table that has the zip code, average house hold income for the zip code, the population of the zip code, and the geographic location of the zip code.



The detailed process of the extraction and transformation entailed:

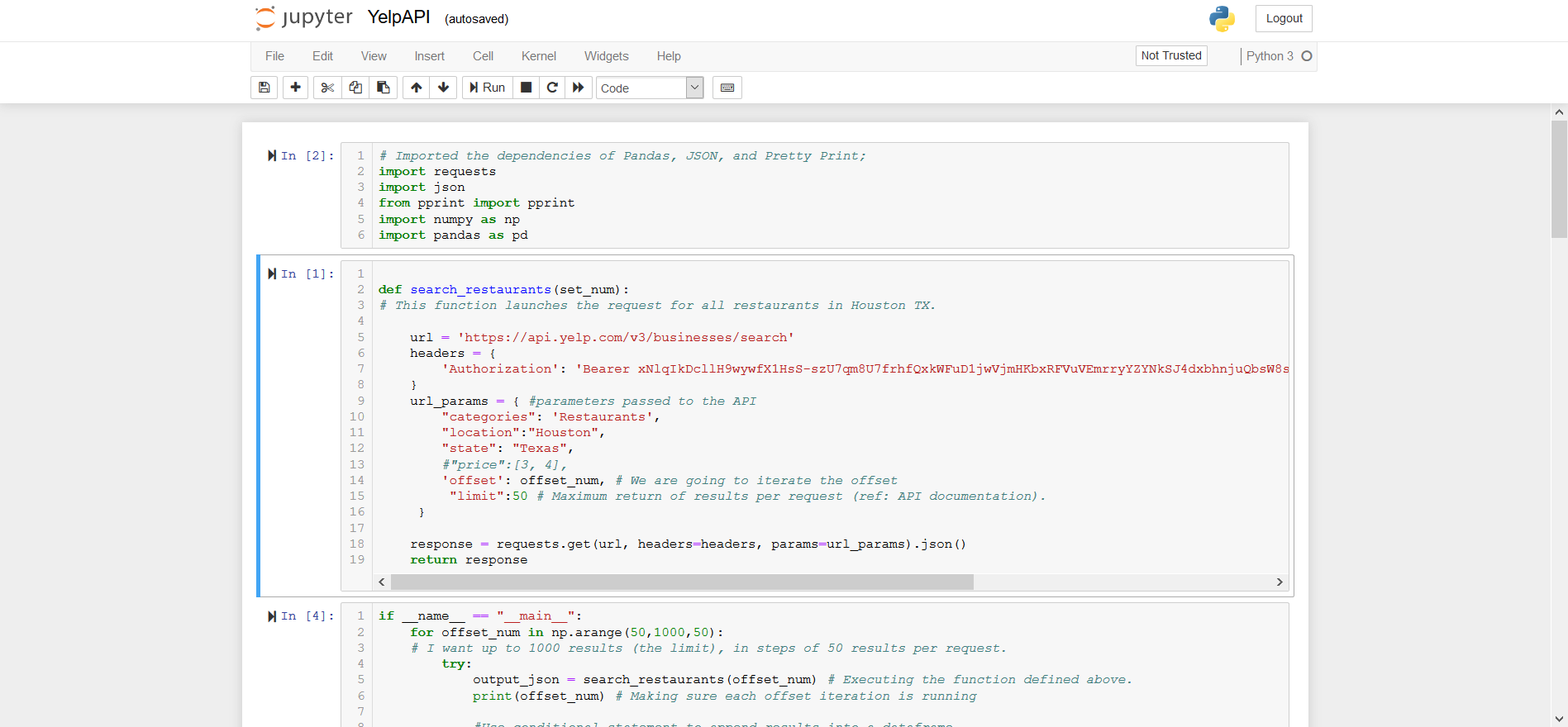
* Imported the dependencies of Pandas, Beautiful Soup, and Pretty Print;
* Setup the request link for the URL with the data;
* Retrieve the Houston zip code statistics with the request module;
* Develop the Beautiful Soup object by parsing with 'html.parser';
* Create a Beautiful Soup object using the text of the HTML string;
* Read in the URL for the data and identified the number of tables on the page;
* Since the table appeared to be the last on the page, started with 14 and worked backwards to identify the table we needed as the twelfth table;
* Created a Pandas data frame and printed the table;
* We **transformed** the data by eliminating the first row and making the second row the header. The first row was a count of columns and the first column was a count of rows and was redundant with the index;
* Renamed the first column from "#" to number to be able to delete the column; and
* Saved the table as a csv file.

1. We created a Yelp API to gather restaurant data for the Houston area.



The Detailing the process of the extraction and transformation entailed:

* Imported the dependencies of Pandas, JSON, and Pretty Print;
* Found the Yelp Developer’s webpage and researched how to query their data and wrote the function to launch the request for all Houston restaurants;



* Wrote the parameters to obtain up to 1,000 results at 50 results per request;
* Reviewed the first five rows of data returned to determine what additional transformation is needed;
* **Transformed** the data as zip code and other information was embedded in other strings of information;
* The zip code data was contained inside the location column as one of the key values in a dictionary so we looped through each row to pull the zipcode value and store in the new dataframe column at that same index;
* Created a Pandas data frame and printed the table; and
* Saved the table as a csv file.

1. We downloaded a CSV file commercial lease data from the Houston area.



The detailed process of the extraction and transformation entailed:

* Imported the dependencies of Pandas, Beautiful Soup, and Pretty Print;
* Setup the request link for the URL with the data;
* Retrieved the Houston commercial property statistics with the request module;
* Read in the URL for the data and identified the number of tables on the page;
* **Transformed** the data by creating a new column that only stored the first 5 characters of the zip-code string. This was the only dataset that used the 9-character zipcode
* Created a Pandas data frame and printed the table;
* Saved the table as a csv file.

**The Loading Process**

We used an online conversion tool <https://www.rebasedata.com/convert-csv-to-sqlite-online> to convert our CSV data files to SQLite data files.

* Searched for an online conversion tool;
* Selected the CSV file and downloaded the converted SQLite file;
* Uploaded the SQLite file to our github repo;

**Flask API**

def loadSession(file):

# Input the route to the data SQL file

path = "sqlite://Resources/"+location

engine = create\_engine(path)

# reflect an existing database into a new model

Base = automap\_base()

# reflect the tables

Base.prepare(engine, reflect=True)

session = Session(engine)

return session

file = leases.sqlite

loadSession(file)

**Why a Relational Database and Hypothetical Use Cases**

We chose a relational database for our project because the source data was relatively clean. We also realize that the information can be used with various other cities and various other industries to identify where to locate a business.