Jerome Urioste

Mitchell Dahmer

Mariam Hassan

Chris Hart

Data Sources:

AirBnb CSV: <http://insideairbnb.com/get-the-data.html>

DarkSky Weather API: <https://darksky.net/dev>

Outline:

Possible uses for this information include:

* Finding times when a certain destination will be the most crowded/empty
* Showing when certain destinations will be the cheapest/most expensive
* Showing what the climate will be like in various peak and low times

Database Relationality:

This database will be relational. The final set of data will show multiple statistics related to both cost of renting and climate, and thus will require the use of a column table style set up. This database will examine the correlation between the cost of renting a space in a certain city and the climate in the area at the time.

Detailing the process of the extraction, transformation, and loading steps:

* Downloaded CSV’s from various cities for Airbnb
* Called two years worth of weather data from Dark Sky’s API for US cities (Chris\_Dark\_Sky - ETL project.ipynb)
* Cleaned and combined both city data and weather data using Pandas(Dark\_Sky\_Concat\_&\_ Clean.ipynb, updated\_airbnbcleaner.ipynb)
* Established database within SQL and began using SQLalchemy into the database.

What data sources you chose, and why:

We first each downloaded CSVs for two cities abroad from Airbnb’s website. We initially wanted to use openweathermap to analyze the weather but the platform only had data available in current, five day, and 16 day weather forecast form, which for the latter also needed to paid for. Consequently we moved over to DarkSky but then realized that the records for the cities outside of the USA was inconsistent causing api calls to halt or alternatively if an exception was included in the code, the 1000 api call limit per day would be wasted on row data that was incomplete. As a result we decided to only use DarkSky data from US cities.

Explication of why you have performed the types of transformations you did:

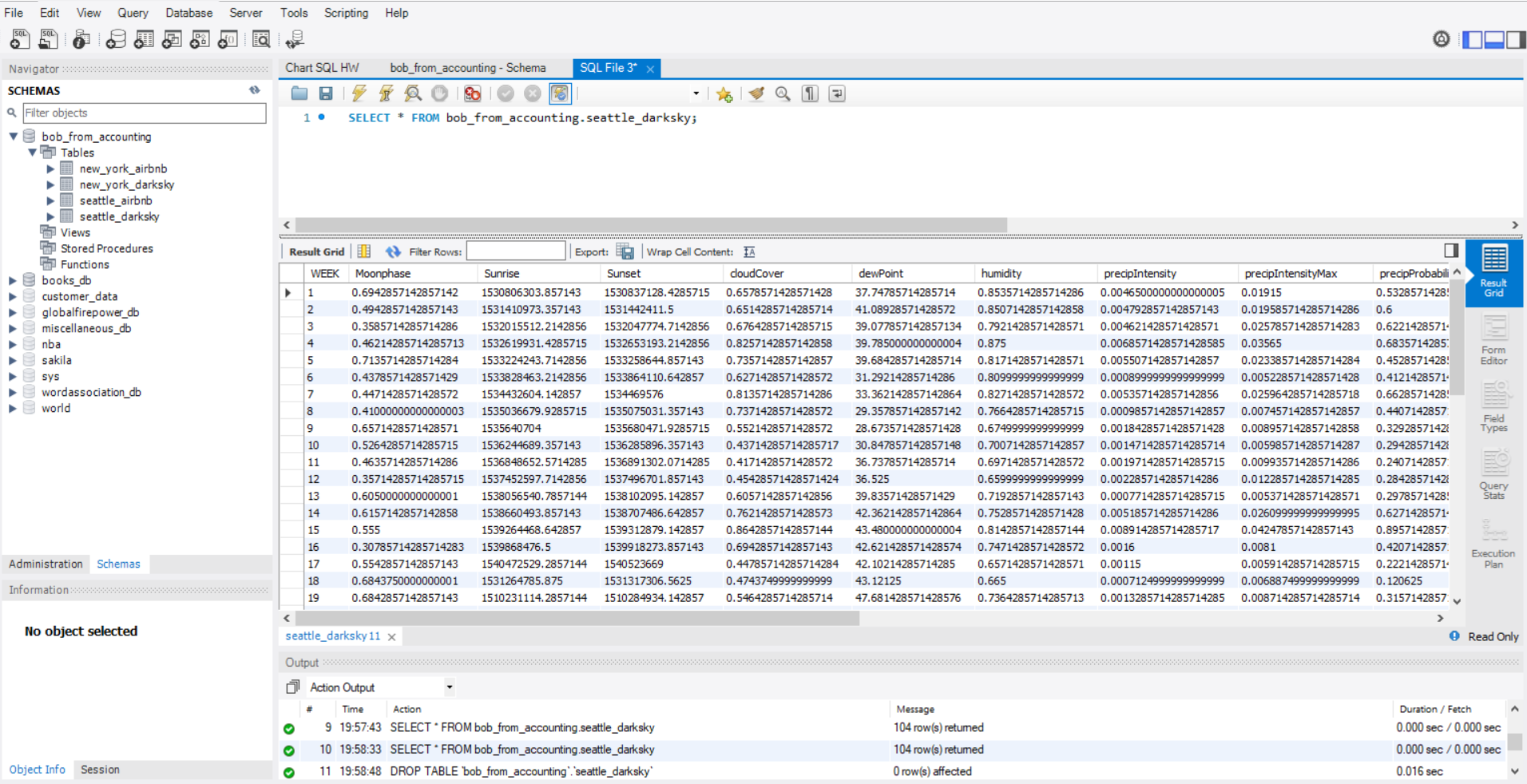
Converted to weather and airbnb dataframes to make them easier to work with, and dropped unnecessary columns to remove clutter then transformed the data into standard date format in order to extract the year,month,and week of each entry and then grouped the entire dataframe by week to get an accurate timeline of the weather in each city to see the changes in price overtime and the climate.

Why you chose the type of the final database:

We chose MySql as we decided the best option for this project was a relational database due to the direct comparison between average airbnb price and climate measurements. This, combines with the large volume of data we received, both in terms of sheer size and number of categories, MySql proved to be the best option for our purposes

Schema of the tables/collections in the final database:

* Load dependencies first
* *from sqlalchemy import create\_engine*
* *import pandas as pd*
* *from sqlalchemy.ext.declarative import declarative\_base*
* *Base = declarative\_base()*
* *from sqlalchemy import Column, Integer, String, Float*
* Create a new database with no tables, then insert into create\_engine path database your root username and password and database name. (create\_engine"mysql://root:root password@localhost/database name)
* Then you want to create a variable called con = engine.connect()
* Load load dataframe to mysql using .to\_sql and create a new table name then run code.
* Data\_frame name .to\_sql(name='new\_table',con=con,if\_exists='append')
* con.close().
* File with basic code to **connect** using python file sql\_connet.ipynb
* Can apply this code from any pandas dataframe.



Hypothetical use cases for your database

This data set has the potential to be very useful for identifying trends in tourism to various cities and how the correspond to different weather patterns. This is based on comparing the average price of the airbnb’s in the area (taken as an indicator of demand) to various statistics related to the weather, such as temperature, or number of daylight hours.