**Technical Report:**

**Extract, Transform, & Load Project**

Rainy with a Chance of Crime

Team: J3AC

Julian Freeman

Jason Ree

Jack Lindsay

Alex Wolf

Chris Holt

**Table of Contents**

1. Introduction
   1. Summary
   2. Scope
   3. Technologies & Resources
   4. Definitions and Abbreviations
2. ETL Process
   1. Crime Data
      1. Dates
      2. Address/Latitude & Longitude
      3. Crimes & Crime Mapping
   2. Weather Data
   3. Joining
3. Data Integrity
   1. Crime Data
   2. Weather Data
4. Data Quality
   1. Conclusions

**1. Introduction**

**a. Summary**

The goal is to determine whether the crimes committed are affected by weather conditions. The outcome of the project will be combining both crime and weather data to be implemented for analysis including crime-weather correlations and heat-mapping.

**b. Scope**

The data includes 5 major U.S. cities (but is available for expansion) with a range of weather conditions. For this project, the following city were selected:

* Atlanta
* Boston
* Chicago
* Los Angeles
* Denver

These cities were selected based on their physical distances from one another and thus, allows for a variety of weather data types to be collected including snow, rain, overcast, etc. This would be matched against all crime-data pulled from the cities over the past 5 years, from the beginning of 2014 to the end of 2018. The only exception to this range was Boston, where the data procured only ranged from August of 2015 to present. However, Boston still contained enough data points that its data could still prove significant, despite this minor difference. To achieve the purposes of this proposal, the data selected for the five cities was based on the availability of the following criteria:

* Duration/years of data
* Locational (latitude/longitude/address) aspects
* Crime codes/descriptions
* Dates and times of the crimes
* Significant number of records

**c. Technologies & Resources**

Python was implemented in order to: pull the weather data, clean the crime data, and verify the crime-code mapping. The team investigated a myriad of different weather data sources, such as Open Weather Map API, however, decided on utilizing the Dark Sky API due to cost-benefit calculations, data structure, ease of use and its detailed documentation. The weather data was pulled from this API for each city over the last 5 years, for 24 hours each date. Each one was outputted into an individual CSV and then combined the files into a single CSV file, in Python.

The crime data was cleaned and made more concise using Python. The data was shortened to include only the necessary columns and to cover only the past 5 years. The column outputs include city, crime codes, date/time, and addresses (latitude/longitude). The sources of the crime data included: Kaggle, Police Departments, and City government websites.

Lastly, for the crime-code mapping, these were standardized and mapped in Python and manual inputted. The codes and mapping were then verified in Python utilizing Pandas. Afterwards, all data was combined and saved within PostgreSQL.

The data sources, data statistics, and tools/modules used for each relevant city is summarized in the table below:

Table 1. Data Overview

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Datasets | Sources | Data Statistics | Tools / Modules Used |
| Crime Data | Atlanta | <http://www.atlantapd.org/i-want-to/crime-data-downloads> | 5,458,515  Data Points | * Python * Pandas * GeoPy.Geocoders * AST * NumPy * Time * Datetime |
| Boston | <https://data.boston.gov/dataset/crime-incident-reports-august-2015-to-date-source-new-system> | 6,629,480  Data Points |
| Chicago | <https://www.kaggle.com/chicago/chicago-crime> | 31,261,892  Data Points |
| Denver | <https://www.kaggle.com/paultimothymooney/denver-crime-data/downloads/denver-crime-data.zip/39> | 9,251,709  Data Points |
| Los Angeles | <https://www.kaggle.com/cityofLA/los-angeles-crime-arrest-data> | 41,079,624  Data Points |
| Weather Data | Cities Above | <https://darksky.net/dev> | ~43,800 Data Points per City | * Python Datetime * NumPy * Pandas |
| Combined Data | All  Above |  |  | * PostgreSQL * pgAdmin4 |

**d. Definitions & Abbreviations**

The city abbreviations that were used for convenience, for ‘save files’ and Python code, were as follows:

* CA – Chicago
* LA – Los Angeles
* ATL – Atlanta
* DNV (or DV) – Denver
* BS – Boston

The following are definitions/explanations of certain column headers and frequently used terms:

* Time:

Hour (approx.) for which the crime is estimated to have occurred and/or the weather event is estimated to have taken place, recorded on a range from 0 to 23 (0 being 12:00 A.M. and 23 being 11:00 P.M.)

* Date:

Year/Month/Day for which the crime was estimated to have occurred and/or the weather event is estimated to have taken place.

* Mapping:

One/Two-word description of the type of crime that occurred in order to log the results in a more uniform manner.

* Code:

The code used for each type of crime regarding the local relevant police department.

**2. ETL Process**

**a. Crime Data**

The crime data was available as CSV files from the list of sources. The process of obtaining the required data for this proposal did not require acquiring permissions or licensing.

**i. Dates**

Preliminary observations revealed that the dates and times of each city were recorded in different formats. Initially, the completeness of the dates and times for each city were checked for NA/NULL/NaNs, and fortunately, none existed. The next step was to ensure consistency of this important column between the five cities’ data, and their weather data, by utilizing the Python Datetime module. The dates and times were parsed and converted from string to datetime data type, then divided into separate columns. The dates were converted to YYYY/MM/DD format and the times ranged from 0 to 23 hours. The reasoning for only including the hours for the time was due to the weather data being hourly records.

**ii. Latitude & Longitude**

First, the latitude and longitudes for each crime, in each city, were inspected for missing values using Python NumPy. While Los Angeles and Atlanta datasets had complete data, Denver, Chicago, and Boston had missing latitudes and longitudes ranging from 0.08% to 6.35% of the data. Many efforts were dedicated to satiating these missing data points by using geolocational Python modules, such as GeoPy.Geocoders and Google Maps. Despite such endeavors, many of the latitudes/longitudes were unable to be found and the missing data percentages were not substantially alleviated (roughly 26% of NaNs were retrieved). Thus, after reaching a conclusion that the net effect of omitting such data from the dataset was negligible, the NaN data rows were excluded.

The next step included parsing and dividing latitude/longitude from addresses, when appropriate. Some datasets had strings of address with latitudes/longitudes, while some were organized in separate columns. For uniformity, the latitudes and longitudes for all the data were organized in their respective rows.

**iii. Crimes & Crime Mapping**

Due to non-uniform reporting and recording systems for crime data, each city’s data had to be mapped. In each dataset, a code and description was assigned to each incident and additionally, there could be several hundred codes in a single dataset. For the purpose of analysis, these codes needed to become standardized and their descriptions succinct so a list of common crime indicators could be referenced across all datasets.

This process involved using Python, Pandas, and Excel. First, utilizing Python and Pandas, a list of unique codes and descriptions were pulled from each dataset. These would also be given a reference column to the city they originated from. This step was crucial, since some codes were non-unique to that city and represented different types of crimes in different cities. After each table was created, it was then saved to a csv file.

The next step was going into each city’s file, in Excel, and mapping the codes. Due to the distinctiveness of each crime’s description, this step was completed manually. The mapping took place in a new column, where a very concise (one/two word) description of that crime was given. This process also included removing any duplicates that were not found and dropped in Python. These previously un-found duplicates were primarily caused by small formatting difference in crime descriptions (e.g. spaces and hyphens). After all crimes were mapped in Excel, Python was utilized again so all mapped datasets could be verified; the number of distinct codes in the files were equal to the number of distinct codes found in their original crime-data file. Once completed and verified, all mapped datasets were then combined to form one table and saved to a CSV, which served as the crime-key-lookup table for all datasets.

**b. Weather Data**

The weather data was acquired using Dark Sky API and Python packages. The API calls were for daily historical data, and each pull included an hour-by-hour weather report for that entire day. This was the best choice for a weather API, as it offered a robust range of historical data. We were able to pull hourly weather descriptions from 2014 to 2018, from all five cities, for only a dollar and some change. Due to some hours not being retrieved, there was some data loss in the API pull, but only approximately 20 data points at most in one city’s hourly weather reports. As each city’s dataset consists of approximately 43,800 data points, the few points lost still entail a 99.95% data retention rate.

The data we retrieved was exceptionally clean and consistent. Since a single API was used to pull this data, all the weather descriptions were similar in type, so there was no need to develop keys or codes in order to join the different cities. Once the data from the API was saved, the clean-up process consisted of dropping empty values, renaming columns (so the data can be easily joined with the crime data), merging all of the weather data into one Data frame, and exporting it all as a CSV.

**c. Joining**

The joining process was rather straight forward but required a bit of finessing when dealing with the column headers. Due to the various sources of data, we had to choose a format method to adhere to. When approaching this project, we realized we had two feasible options. The first option was to merge all the data into a singular Pandas Data frame, export it to a CSV, and upload that file to the SQL database. Then the second option was to build a schema in SQL, import each CSV file, and manually merge the tables in SQL. Fortunately, the latter option was chosen as we would later come to discover the merged data file exceeded 31.6 million datapoints. Uploading a CSV file of that magnitude is problematic and often leads to timeout errors, even on local servers. Implementing the second option instead, we created 3 tables:

1. Crime Data
2. Weather Data
3. Crime Keys Table (mapping)

The Crime and Weather data were then merged on the following column headers: ‘dates’, ‘time’, and ‘cities’. Simultaneously, we merged the crime-keys to that dataset on the following column headers: ‘codes’ and ‘cities’. Once the merging was completed, we created a new table in the SQL Database to hold the newly combined data.

The final dataset contained the following data categories listed by their respective column headers:

* City
* Date
* Time
* Code
* Mapping
* Longitude
* Latitude
* Weather

**3. Data Integrity**

**a. Crime Data**

As the data was cleaned, some missing records in both crime and weather data made itself known. In the crime data, this data was missing only in the latitudinal and longitudinal records. Since the addresses of the crime scenes were available, manual observations of a handful of these addresses were searched via Google Maps to better understand what may be causing the absent information. Interestingly, many of these addresses were isolated and desolate areas, e.g., a back alley of a row of restaurants or an open field near an airport, which is assumed to explain the initial omissions in the latitude/longitudes due to the difficultly of identifying and recording such locations. There were solutions to remedy such missing data points via the use of address records and GeoPy.Geocoders, Google Maps or other geolocational Python modules. However, even after numerous attempts to gather the latitudes and longitudes of the missing data using such tools, there were many instances where the geolocational module was not able to provide suitable and enough results. The ratios of missing data to the total size of the datasets were calculated to yield 1.33%, 0.08%, and 6.35% for Chicago, Denver, and Boston, respectively. When all data was accounted for in one combined dataset, these ratios combined drop to approximately 1.2% of all crime-data, giving crime data locations an approximate consistency of 98.8%. Given the large size of the total size of all crime data, it is safe to assume that the omission of these missing values would not substantially impact the analysis. It should also be noted that this missing data will only affect any future attempts to geographically map the crime data. The crime data has 100% reliability, regarding the mapping of crimes in relation to the dates and code of those crimes.

**b. Weather Data**

The weather data was considerably clean across the date ranges that were selected, as mentioned above. Because this data was not free, the data had to be pull efficiently as possible. After the data was pulled, almost every hour of every day in our range had a description of the weather at that time. This forecasting data was approximated to the city from which the readings were taken. We can’t pinpoint weather differences across different areas of these major cities, but we reasoned that weather differences across any city wouldn’t stray far from the forecasted description, if it strayed at all.

**4. Data Quality**

**a. Conclusions**

Overall, the result of the project met the desired objectives. Despite the challenges the diverse systems of crime records created, those records and weather data were cleaned and standardized with ~99% data reliability and over 31.6 million data records. The data has been combined into one superset, which is easily loadable in pgAdmin4, making it readily accessible despite the enormous size of the file. In conclusion, the data is ready for further correlations and geographic analysis.