**UCDavis-Data Analytics Bootcamp**

**Project 2 Outline**

Steps:

1. Determined data source
2. Wireframed what html site would look like
3. Started basic HTML code
4. Extracted data source to csv
5. Manipulated data source in pandas and loaded to SQL database
6. Created Flask app
7. Started crimemap.js file to build map layers
8. Downloaded Oakland neighborhood geojson
9. Created new virtual environment and installed geopandas
10. Conducted geo-spatial join to add neighborhood names to crime data and crime summary data to neighborhood geojson
11. Re-wrote crime data with neighborhood information to SQL database
12. Set up user filters (via .js) for the site: users can filter on year (2019 or 2020) and crime category (property crime, violent crime, other crime)
13. Created choropleth map
14. Created bar chart ranking crime types within given year and crime category
15. Created pie chart showing top 10 neighborhoods where most crime incidents occurred for given year and crime category
16. Created data table giving case number, crime description, and address where crime occurred for the selected year and crime category (with ability to search anything in table, and utilize various pagination options)

**Data ETL Process**

**Data Sources (Extract):**

<https://www.oaklandca.gov/resources/police-incident-data>

* Originally pulled past 90-day
* Changed scope and determined we would analyze 2019 and 2020 historical data
* Ultimately went to: <https://data.oaklandca.gov/Public-Safety/Download-Oakland-Crime-Heatmap/56xf-w7yc>

Oakland Neighborhoods Geojson:

* <https://github.com/maxogden/oakland-boundaries-geojson/blob/master/neighborhoods.geojson>

**Data Manipulation (Transform):**

* Original Dataset
  + Pulled and manipulated the columns in pandas (via Jupyter Notebook)
  + Created columns for Lat/Long by splitting column where they were combined with other text
  + Wanted this to be reproduceable, so you could consistently refresh
  + During data analysis, we realized that the past 90-day data was wonky.
  + Included data from 2018 and 2019
  + Wouldn’t provide enough data points for a good choropleth map
* New Dataset
  + From the “Export”, downloaded CSV
  + Data set included over 260,000 rows of data
  + Did data manipulation in Excel to limit to 2019 and 2020 data, tease out lat/long into unique columns, and remove rows that didn’t have key data fields (lat/long or crime types)
* Geo-Spatial Join
  + Created new virtual environment (geo\_env) to install geo pandas and limited other items so as not to accidentally mess anything up in our PythonData environment
  + Read in crime data into Jupyter notebook pandas data frame via our custom API endpoint
  + Read in neighborhoods geojson file into a geo-pandas geo-data frame
  + Converted crime data frame into geo-data frame
  + *Add other notes by following rest of notes in “Geopandas Spatial Join.ipynb” file*

**Data Loading:**

As mentioned, originally via pandas only. Changed scope and then did some manipulation in Excel as outlined above.

Jupyter Notebook

* Original Dataset
  + Read CSV into pandas data frame
  + Set up database in Postgres via PGAdmin
  + Created connection to SQL (Postgres) via Jupyter Notebook
  + Wrote data from data frame to SQL
* Geo-Spatial Joined Dataset
  + Re-wrote to SQL
* Load data to site via custom API endpoint

**Other Processes** (Fill out detail, maybe highlight challenges)

* Wireframing Website: <https://layoutit.com/>
  + Visually outline site components
  + Able to then download basic HTML code to get started
* Creating Flask API
  + Various Routes (default route for main landing page, route for API data, route that does SQL query to summarize data)
* JavaScript Coding
  + User filters on landing site to filter data
  + Bar Chart
  + Pie Chart
  + Data Table