

Project ETL - Extract, Transform, Load

Team members

Swati Swain

Emi Rivera

Nicholas Prentkowski

Terrence Cummings

Minneapolis Police Incidents, Use of Force, and Demographics

Project Objective:

To create a database that will facilitate an analysis of the relationship between police incidents, the use of force by police in those incidents, and the demographic characteristics of the neighborhood in which the incidents occurred such as race and income.

The team will extract data from various sources, clean and normalize the data, and load the data into a Postgres database. Such data can then be used by applications to allow analysis of relationships and trends between these factors.

We chose relational database for this project for the below reasons:

1. We are working with complex queries and reports. With SQL we can build one script that retrieves and presents the data easily. Running queries in NoSQL is doable, but much slower.
2. We have a high transaction application. SQL databases are a better fit for heavy duty or complex transactions because it's more stable and ensure data integrity.
3. We need to ensure ACID compliance (Atomicity, Consistency, Isolation, Durability) or defining exactly how transactions interact with a database.
4. We don't anticipate a lot of changes or growth.

The target data, for the city of Minneapolis, includes:

1. Police incident data by neighborhood
2. Police "use of force" data by neighborhood
3. Income levels by neighborhood
4. Race distribution by neighborhood

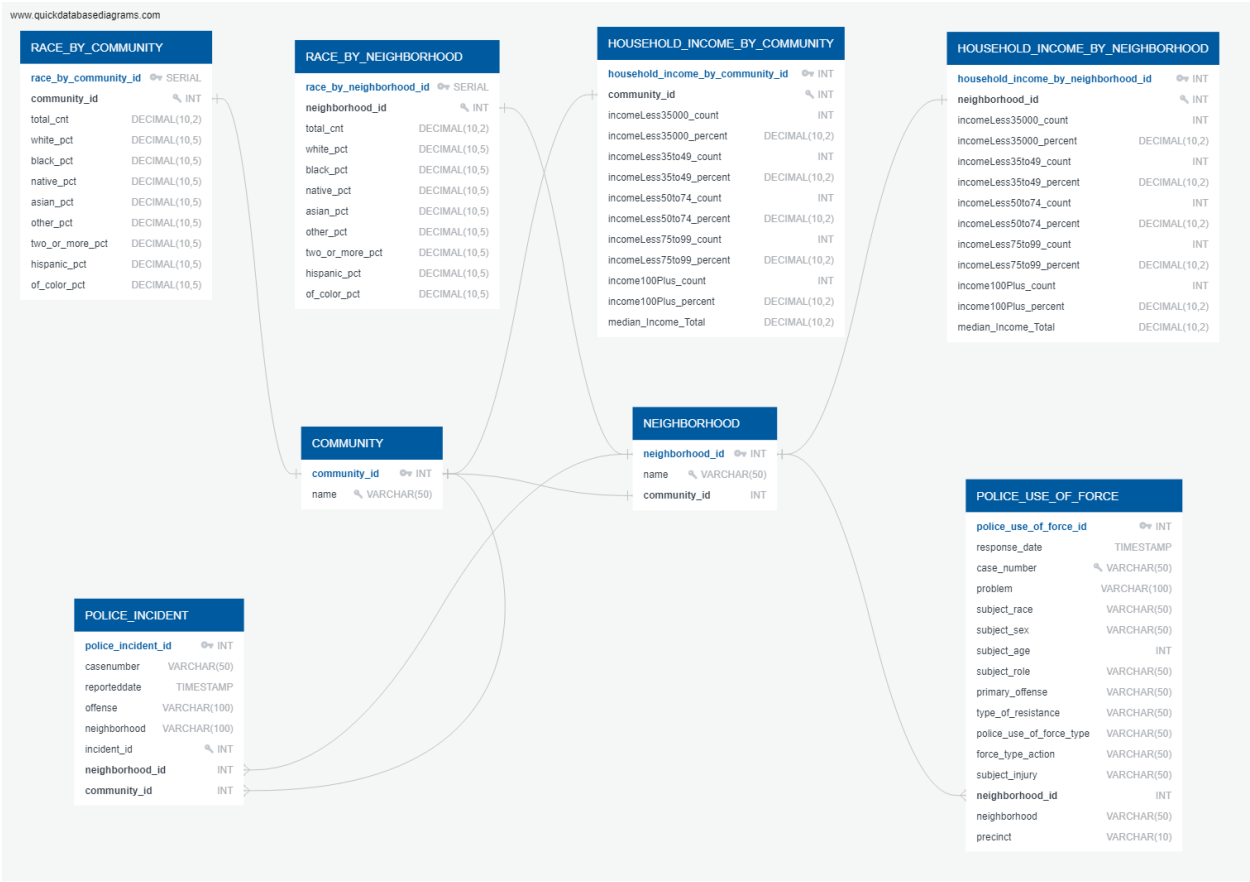
Please find following Table of Contents directing to ERD, SQL, and detailed ETL flow for each dataset.

Table of Contents

ENTITY RELATIONSHIP DIAGRAM (ERD).....	4
SQL FOR POSTGRESQL TABLES.....	5
MINNEAPOLIS COMMUNITIES & NEIGHBORHOODS	10
DESCRIPTION.....	10
DATA SOURCES	10
DATA EXTRACTION.....	10
DATA TRANSFORMATION	10
DATA LOADING	10
JUPYTER NOTEBOOKS	11
POLICE INCIDENTS.....	12
DESCRIPTION.....	12
DATA SOURCES	12
DATA EXTRACTION.....	12
DATA TRANSFORMATION	12
DATA LOADING	12
POLICE USE OF FORCE	13
DESCRIPTION.....	13
DATA SOURCES	13
DATA EXTRACTION.....	13
DATA TRANSFORMATION	13
DATA LOADING	14
NEIGHBORHOOD RACE DEMOGRAPHICS.....	15
DESCRIPTION.....	15
DATA SOURCES	15
DATA EXTRACTION.....	15
DATA TRANSFORMATION	15
DATA LOADING	16
JUPYTER NOTEBOOK	16
NEIGHBORHOOD INCOME DEMOGRAPHICS	17
DESCRIPTION.....	17

DATA SOURCES	17
DATA EXTRACTION.....	17
JUPYTER NOTEBOOK:	17
JUPYTER NOTEBOOK:	18
DATA TRANSFORMATION	18
JUPYTER NOTEBOOKS:	18
DATA LOADING	18
JUPYTER NOTEBOOKS:	19
 WEB APPLICATION	 20

Entity Relationship Diagram (ERD)



SQL for PostgreSQL tables

```
--  
COMMUNITY  
  
DROP TABLE IF EXISTS COMMUNITY CASCADE;  
CREATE TABLE COMMUNITY (  
    community_id INT NOT NULL,  
    name VARCHAR(50) NOT NULL,  
    CONSTRAINT pk_COMMUNITY PRIMARY KEY (  
        community_id  
    ),  
    CONSTRAINT uc_COMMUNITY_name UNIQUE (  
        name  
    )  
);  
  
--NEIGHBORHOOD  
  
DROP TABLE IF EXISTS NEIGHBORHOOD CASCADE;  
CREATE TABLE NEIGHBORHOOD (  
    neighborhood_id INT NOT NULL,  
    name VARCHAR(50) NOT NULL,  
    community_id INT NOT NULL,  
    CONSTRAINT pk_NEIGHBORHOOD PRIMARY KEY (  
        neighborhood_id  
    ),  
    CONSTRAINT uc_NEIGHBORHOOD_name UNIQUE (  
        name  
    )  
);  
  
DROP TABLE IF EXISTS HOUSEHOLD_INCOME_BY_NEIGHBORHOOD CASCADE;  
CREATE TABLE HOUSEHOLD_INCOME_BY_NEIGHBORHOOD (  
    household_income_by_neighborhood_id INT NOT NULL,  
    neighborhood_id INT NOT NULL,  
    IncomeLess35000_count INT DEFAULT NULL,  
    IncomeLess35000_percent DECIMAL(10,2) DEFAULT NULL,  
    IncomeLess35to49_count INT DEFAULT NULL,  
    IncomeLess35to49_percent DECIMAL(10,2) DEFAULT NULL,  
    IncomeLess50to74_count INT DEFAULT NULL,  
    IncomeLess50to74_percent DECIMAL(10,2) DEFAULT NULL,  
    IncomeLess75to99_count INT DEFAULT NULL,  
    IncomeLess75to99_percent DECIMAL(10,2) DEFAULT NULL,
```

```

Income100Plus_count INT DEFAULT NULL,
Income100Plus_percent DECIMAL(10,2) DEFAULT NULL,
Median_Income_Total DECIMAL(10,2) DEFAULT NULL,
CONSTRAINT pk_HOUSEHOLD_INCOME_BY_NEIGHBORHOOD PRIMARY KEY (
    household_income_by_neighborhood_id
),
CONSTRAINT uc_HOUSEHOLD_INCOME_BY_NEIGHBORHOOD_neighborhood_id UNIQUE (
    neighborhood_id
)
);

```

```

DROP TABLE IF EXISTS HOUSEHOLD_INCOME_BY_COMMUNITY CASCADE;

```

```

CREATE TABLE HOUSEHOLD_INCOME_BY_COMMUNITY (
    household_income_by_community_id INT NOT NULL,
    community_id INT NOT NULL,
    IncomeLess35000_count INT DEFAULT NULL,
    IncomeLess35000_percent DECIMAL(10,2) DEFAULT NULL,
    IncomeLess35to49_count INT DEFAULT NULL,
    IncomeLess35to49_percent DECIMAL(10,2) DEFAULT NULL,
    IncomeLess50to74_count INT DEFAULT NULL,
    IncomeLess50to74_percent DECIMAL(10,2) DEFAULT NULL,
    IncomeLess75to99_count INT DEFAULT NULL,
    IncomeLess75to99_percent DECIMAL(10,2) DEFAULT NULL,
    Income100Plus_count INT DEFAULT NULL,
    Income100Plus_percent DECIMAL(10,2) DEFAULT NULL,
    Median_Income_Total DECIMAL(10,2) DEFAULT NULL,
    CONSTRAINT pk_HOUSEHOLD_INCOME_BY_COMMUNITY PRIMARY KEY (
        household_income_by_community_id
    ),
    CONSTRAINT uc_HOUSEHOLD_INCOME_BY_COMMUNITY_community_id UNIQUE (
        community_id
    )
);

```

```

DROP TABLE IF EXISTS RACE_BY_NEIGHBORHOOD CASCADE;

```

```

CREATE TABLE RACE_BY_NEIGHBORHOOD (
    race_by_neighborhood_id SERIAL NOT NULL,
    neighborhood_id INT NOT NULL,
    total_cnt DECIMAL(10,2) DEFAULT NULL,
    white_pct DECIMAL(10,5) DEFAULT NULL,
    black_pct DECIMAL(10,5) DEFAULT NULL,
    native_pct DECIMAL(10,5) DEFAULT NULL,
    asian_pct DECIMAL(10,5) DEFAULT NULL,

```

```

other_pct DECIMAL(10,5)    DEFAULT NULL,
two_or_more_pct DECIMAL(10,5)    DEFAULT NULL,
hispanic_pct DECIMAL(10,5)    DEFAULT NULL,
of_color_pct DECIMAL(10,5)    DEFAULT NULL,
CONSTRAINT pk_RACE_BY_NEIGHBORHOOD PRIMARY KEY (
    race_by_neighborhood_id
),
CONSTRAINT uc_RACE_BY_NEIGHBORHOOD_neighborhood_id UNIQUE (
    neighborhood_id
)
);

```

```

DROP TABLE IF EXISTS RACE_BY_COMMUNITY CASCADE;

```

```

CREATE TABLE RACE_BY_COMMUNITY (
    race_by_community_id SERIAL NOT NULL,
    community_id INT NOT NULL,
    total_cnt DECIMAL(10,2)    DEFAULT NULL,
    white_pct DECIMAL(10,5)    DEFAULT NULL,
    black_pct DECIMAL(10,5)    DEFAULT NULL,
    native_pct DECIMAL(10,5)    DEFAULT NULL,
    asian_pct DECIMAL(10,5)    DEFAULT NULL,
    other_pct DECIMAL(10,5)    DEFAULT NULL,
    two_or_more_pct DECIMAL(10,5)    DEFAULT NULL,
    hispanic_pct DECIMAL(10,5)    DEFAULT NULL,
    of_color_pct DECIMAL(10,5)    DEFAULT NULL,
    CONSTRAINT pk_RACE_BY_COMMUNITY PRIMARY KEY (
        race_by_community_id
    ),
    CONSTRAINT uc_RACE_BY_COMMUNITY_community_id UNIQUE (
        community_id
    )
);

```

```

DROP TABLE IF EXISTS POLICE_USE_OF_FORCE CASCADE;

```

```

CREATE TABLE POLICE_USE_OF_FORCE (
    police_use_of_force_id SERIAL NOT NULL,
    response_date TIMESTAMP    DEFAULT NULL,
    case_number VARCHAR(50)    DEFAULT NULL,
    problem VARCHAR(100)    DEFAULT NULL,
    subject_race VARCHAR(50)    DEFAULT NULL,
    subject_sex VARCHAR(50)    DEFAULT NULL,
    subject_age INT    DEFAULT NULL,
    subject_role VARCHAR(50)    DEFAULT NULL,

```

```

primary_offense VARCHAR(50) DEFAULT NULL,
type_of_resistance VARCHAR(50) DEFAULT NULL,
police_use_of_force_type VARCHAR(50) DEFAULT NULL,
force_type_action VARCHAR(50) DEFAULT NULL,
subject_injury VARCHAR(50) DEFAULT NULL,
neighborhood_id INT DEFAULT NULL,
neighborhood VARCHAR(50) DEFAULT NULL,
precinct VARCHAR(10) DEFAULT NULL,
CONSTRAINT pk_POLICE_USE_OF_FORCE PRIMARY KEY (
    police_use_of_force_id
),
CONSTRAINT uc_POLICE_USE_OF_FORCE_case_number UNIQUE (
    case_number
)
);

```

```

DROP TABLE IF EXISTS POLICE_INCIDENT CASCADE;

```

```

CREATE TABLE POLICE_INCIDENT (
    police_incident_id SERIAL NOT NULL,
    casenumber VARCHAR(50) DEFAULT NULL,
    reporteddate TIMESTAMP DEFAULT NULL,
    offense VARCHAR(100) DEFAULT NULL,
    neighborhood VARCHAR(100) DEFAULT NULL,
    incident_id INT DEFAULT NULL,
    neighborhood_id INT DEFAULT NULL,
    community_id INT DEFAULT NULL,
    CONSTRAINT pk_POLICE_INCIDENT PRIMARY KEY (
        police_incident_id
    ),
    CONSTRAINT uc_POLICE_INCIDENT_incident_id UNIQUE (
        incident_id
    )
);

```

```

--Foreign Keys

```

```

ALTER TABLE NEIGHBORHOOD ADD CONSTRAINT fk_NEIGHBORHOOD_community_id FOREIGN
KEY(community_id)
REFERENCES COMMUNITY (community_id);

```



```
ALTER TABLE HOUSEHOLD_INCOME_BY_NEIGHBORHOOD ADD CONSTRAINT  
fk_HOUSEHOLD_INCOME_BY_NEIGHBORHOOD_neighborhood_id FOREIGN  
KEY(neighborhood_id)  
REFERENCES NEIGHBORHOOD (neighborhood_id);
```

```
ALTER TABLE HOUSEHOLD_INCOME_BY_COMMUNITY ADD CONSTRAINT  
fk_HOUSEHOLD_INCOME_BY_COMMUNITY_community_id FOREIGN KEY(community_id)  
REFERENCES COMMUNITY (community_id);
```

```
ALTER TABLE HOUSEHOLD_INCOME_BY_NEIGHBORHOOD ADD CONSTRAINT  
fk_HOUSEHOLD_INCOME_BY_NEIGHBORHOOD_neighborhood_id FOREIGN  
KEY(neighborhood_id)  
REFERENCES NEIGHBORHOOD (neighborhood_id);
```

```
ALTER TABLE HOUSEHOLD_INCOME_BY_COMMUNITY ADD CONSTRAINT  
fk_HOUSEHOLD_INCOME_BY_COMMUNITY_community_id FOREIGN KEY(community_id)  
REFERENCES COMMUNITY (community_id);
```

```
ALTER TABLE RACE_BY_COMMUNITY ADD CONSTRAINT fk_RACE_BY_COMMUNITY_community_id  
FOREIGN KEY(community_id)  
REFERENCES COMMUNITY (community_id);
```

```
ALTER TABLE POLICE_USE_OF_FORCE ADD CONSTRAINT  
fk_POLICE_USE_OF_FORCE_neighborhood_id FOREIGN KEY(neighborhood_id)  
REFERENCES NEIGHBORHOOD (neighborhood_id);
```

```
ALTER TABLE POLICE_INCIDENT ADD CONSTRAINT fk_POLICE_INCIDENT_neighborhood_id  
FOREIGN KEY(neighborhood_id)  
REFERENCES NEIGHBORHOOD (neighborhood_id);
```

```
ALTER TABLE POLICE_INCIDENT ADD CONSTRAINT fk_POLICE_INCIDENT_community_id  
FOREIGN KEY(community_id)  
REFERENCES COMMUNITY (community_id);
```

Minneapolis Communities & Neighborhoods

Description

The objective is to set up reference tables for Minneapolis Communities and Neighborhoods. The community_id/neighborhood_id(s) will be used as foreign keys in all tables to aggregate data at community and neighborhood level.

Data sources

Data is obtained from Open Data Minneapolis Police Incident reports in form of csv files.

Source file location in project folder:

- a. source_files\MLPS_Communities_raw.csv
- b. source_files\MLPS_Neighborhoods_raw.csv

Data extraction

Use the raw csv files from MINNESOTA COMPASS (mncompass.org) for Community and Neighborhood data.

Data transformation

- **Community Data transformation:**
 1. Import MLPS_Communities_raw.csv from source_files into a dataframe.
 2. Extract the final list of fields from the dataframe to match the database table.
 3. Rename fields in dataframe which don't match the SQL table.
 4. Start the index field from 1(increment by 1 as well) in the final df and rename index field to match the column in the table.
- **Neighborhood Data transformation:**
 1. Import MLPS_Neighborhoods_raw.csv from source_files into a dataframe.
 2. Import the community table from Postgres to get the community names and IDs.
 3. Join the neighborhood dataframe with the community dataframe on the community names to get the cmonnunity ID.
 4. Validate count of records before and after the join to ensure no records were dropped.
 5. Extract the final list of fields from the dataframe to match the database table.
 6. Rename fields in dataframe which don't match the SQL table.
 7. Start the index field from 1(increment by 1 as well) in the final df and rename index field to match the column in the table.

Data loading

Below steps were performed to load MLS neighborhoods and communities tables in Postgres:

1. Postgres tables: neighborhood & community were created using the SQL script generated from the ERD diagram.
2. Neighborhood and Community data was loaded to their respective tables using SQL Alchemy through Python/pandas.

Jupyter notebooks

- **Community data:** 1_MLS_Community_Data_ETL.ipnyb
- **Neighborhood data:** 2_MLS_Neighborhood_Data_ETL.ipnyb

Police Incidents

Description

Any time the police are dispatched to an emergency call or respond on their own, an officer must file an incident report. The police in Minneapolis (MPD) keep track of these incidents utilizing an electronic records management systems. Prior to 2018 the MPD were using a system called CAPRS, in mid 2018 they began using PIMS which offered more flexibility and technological ability.

Data sources

We used Open Minneapolis to obtain the Police Incident data. Open Minneapolis is a website that has a variety of Minneapolis data provided by the City of Minneapolis.

Data extraction

The data was extracted from Open Minneapolis via CSV's. We utilized data spanning 2015 to 2019. In total six data sets were utilized, one CSV per year. Additionally, the 2018 data was split into two CSV's due to the recording system changing halfway through the year (there was one data set for the beginning of the year and another once the system changed).

Data transformation

Data transformations were executed in Python.

- The first transformation to occur was a union of all of the six files into one police incident dataset spanning the four years. An additional challenge was posed due to the fact the system change also changed the formatting of the CSV's. The variables remained relatively similar but the names and order of the variables changed.
- Most of the fields were irrelevant to our analysis so they were dropped.
- The neighborhood field was a pivotal piece to our project, any police incident with a missing neighborhood name was dropped.
- The last major transformation was merging the final police incident dataset with a neighborhood dataset in order to obtain a unique neighborhood and community ID. There were transformations made to the neighborhood names in both datasets as these did not match across all of the neighborhoods.
- Once all neighborhood names matched these datasets were joined together.
- The ultimate file contained the case number, reported date, offense, neighborhood, incident ID, neighborhood ID, and a community ID.

Data loading

The ultimate file was exported as a CSV and was loaded into our PostgreSQL database.

Police Use of Force

Description

The objective is to obtain data regarding the incidents during which Minneapolis police officers deemed the use of force necessary.

Data sources

The primary source utilized to meet the specified objective was a data frame found on the webpage titled, *Police Use of Force*, which can be found on the website titled, Open Minneapolis. The page can be found via the following hyperlink:

<http://opendata.minneapolismn.gov/datasets/police-use-of-force/data?geometry=-103.617%2C-5.468%2C10.289%2C48.789&orderBy=ResponseDate&orderByAsc=false>

Data extraction

The source data was extracted via a comma-separated values file (.CSV) that was first downloaded locally and finally pushed onto our project team's repository. It is specifically stored in the folder titled, "source_file."

Data transformation

Transformation (cleaning) involved the following steps:

1. Declaring and assigning a variable to the CSV...
2. Reading in the CSV by using the read_csv function, which will produce and store a Pandas data frame...
3. Dropping unessential data fields...
4. Renaming remaining fields to match entity-relationship diagram (ERD)...
5. Using .dtypes code to determine type of value held in the response_date field...
6. Using astype function and Numpy to convert response_date field to datetime64...
7. Declaring and assigning a variable to the MLS_Neighborhoods CSV...
8. Changing the field name titled: name, in new data frame to, neighborhood...
9. Using the replace function to match the spelling and punctuation of the ten neighborhoods that conflict with one another when trying to merge the two data frames...
10. Merging the two data frames on the field, neighborhood and via a left join...
11. Use double brackets to rearrange the order of the fields of data frame to match the ERD...
12. Use .dtypes to check, or refer to the last time it was used to see what type of values can be found in the subject_age field...
13. Change all NaN(s) within subject_age field to 0 via fillna, which will allow for conversion to int64...
14. Convert subject_age field to int64 via astype function...
15. Change all NaN(s) within neighborhood_id field to 0 via fillna...
16. Convert values in neighborhood_id from float64 to int64 via .astype...
17. Rename final data frame to something more concise and clearer...
18. Export as CSV to the folder, target_files...

Data loading

The steps are as follows:

1. Create tables in PostgreSQL using the SQL script based on the ERD presented at the beginning of this document.
2. Use SQLAlchemy (from sqlalchemy import create_engine) to connect to PostgreSQL database.
3. Use Pandas df.to_sql to populate PostgreSQL tables with Pandas dataframe values.

Neighborhood Race Demographics

Description

The objective is to obtain data regarding the racial mix of Minneapolis neighborhoods and communities.

Data sources

Data is obtained from MINNESOTA COMPASS (mncompass.org). We need to scrape data from the following endpoints:

1. Scrape links to Minneapolis neighborhood-specific webpage on mncompass.org found on:
 - a. <http://www.mncompass.org/profiles/neighborhoods/minneapolis-saint-paul>
2. Scrape links to Minneapolis community-specific webpage on mncompass.org found on:
 - a. <http://www.mncompass.org/profiles/neighborhoods/minneapolis-saint-paul>
3. Scrape race data for each Minneapolis neighborhood links obtained in step 1. For example, Armatage neighborhood at:
 - a. <http://www.mncompass.org/profiles/neighborhoods/minneapolis/armatage>
4. Scrape race data for each Minneapolis community links obtained in step 2. For example, Camden at:
 - a. <http://www.mncompass.org/profiles/communities/minneapolis/camden>

Data extraction

Selenium webdriver (from selenium import webdriver) was used to scrape data at the URL. This is because the data is populated by Javascript and therefore not accessible by Splinter.

Extraction followed the following process:

1. Scrape the individual neighborhood and community links and store in lists of URLs.
2. Send the webdriver to each link in the lists and scrape the race data from each page.

The neighborhood and community race data is then stored in a Pandas dataframe and written to csv files.

Data transformation

Transformation (cleaning) involved the following steps:

1. Read in the csv's from extraction as Pandas dataframes.
2. The scraped data contained the word 'suppressed' in some table cells. Replace this with NaN so all missing data is represented by NaN.
3. Convert text-styled numbers into numeric type.
4. Add a 'total' column as the sum of the individual race columns.
5. Use pd.merge to bring in neighborhood and community ID's that will be used in PostgreSQL keys.
6. Delete extraneous columns.
7. Reorder columns for presentability.

Data loading

Steps:

4. Create tables in PostgreSQL using the SQL script based on the ERD presented at the beginning of this document.
5. Use SQLAlchemy (from sqlalchemy import create_engine) to connect to PostgreSQL database.
6. Use Pandas df.to_sql to populate PostgreSQL tables with Pandas dataframe values.

Jupyter Notebook

1. 6_MLS_Race_Data_Extract.ipynb
2. 6a_MLS_Race_Data_Transform.ipynb

Neighborhood Income Demographics

Description

The objective is to obtain data regarding the household income of Minneapolis neighborhoods and communities.

Data sources

Data is obtained from MINNESOTA COMPASS (mncompass.org). We need to scrape data from the following endpoints:

1. Scrape links to Minneapolis neighborhood and community specific webpage on mncompass.org from:
 - a. <http://www.mncompass.org/profiles/neighborhoods/minneapolis-saint-paul>
2. Scrape household income data for each Minneapolis neighborhood links obtained in step 1. For example, Downtown West neighborhood at:
 - a. <http://www.mncompass.org/profiles/neighborhoods/minneapolis/downtown-west>
3. Scrape household income data for each Minneapolis community links obtained in step 2. For example, Loring Park at:
 - a. <http://www.mncompass.org/profiles/neighborhoods/minneapolis/loring-park>

Data extraction

The webpages in MN Compass are populated via java script. Selenium package is used to automate web browser interaction from Python. Selenium will start a browser session. The `python_button.click()` is used to tell Selenium to click the JavaScript link on the page. After arriving at the individual MLS neighborhood page, Selenium hands off the page source to BeautifulSoup. BeautifulSoup will grab all the rendered data on the page that matches the required class and IDs.

Extraction of neighborhood household income data:

1. Scrape all individual neighborhood part-links from <http://www.mncompass.org/profiles/neighborhoods/minneapolis-saint-paul> page with **class_='minneapolis-neighborhoods-listing'**
2. Append '<http://www.mncompass.org/profiles/neighborhoods/>' to the part links from step 1 to create complete links for neighborhood pages.
3. Loop through the neighborhood links in step 2 to fetch household income data for all available MLS links using a combination of Selenium and BeautifulSoup.
4. Save the final data set as '**MLPS_Hsld_Income_by_Neighborhood.csv**' in resources folder.

Jupyter Notebook:

- 3a_MLS_Income_by_Neighborhood_Extract.ipnyb

Extraction of community household income data:

1. Scrape all individual neighborhood part-links from <http://www.mncompass.org/profiles/neighborhoods/minneapolis-saint-paul> page with **class_ = 'minneapolis-communities-listing'**
2. Append '<http://www.mncompass.org/profiles/neighborhoods/>' to the part links from step 1 to create complete links for neighborhood pages.
3. Loop through the neighborhood links in step 2 to fetch household income data for all available MLS links using a combination of Selenium and BeautifulSoup.
4. Save the final data set as '**MLPS_Hsld_Income_by_Community.csv**' in resources folder.

Jupyter Notebook:

- 4a_MLS_Income_by_Community_Extract.ipnyb

Data transformation

Below steps were performed to clean/transform household income data for MLS neighborhoods and communities:

1. Import the csvs saved in the 'Extract' steps into a dataframe.
2. Replace \n, commas(,), %, \$ in source data with "
3. Trim whitespace from all values across all fields in the dataframe.
4. Median income field: Extract the median income by removing '\$ ' and converting to float.
5. Replace 'suppressed' and blank fields with '**NaN**' across the dataframe.
6. Convert all household income count fields to numeric.
7. Convert all household percent fields to float.
8. Import the neighborhood table from Postgres to get the neighborhood names and IDs.
9. Join the household income dataframe with the neighborhood dataframe on the neighborhood names to get the neighborhood ID.
10. Validate count of records before and after the join to ensure no records were dropped.
11. Extract the final list of fields from the dataframe to match the database table.
12. Rename fields in dataframe which don't match the SQL table.
13. Start the index field from 1(increment by 1 as well) in the final df and rename index field to match the column in the table.

Jupyter notebooks:

- **Neighborhood data:** 3b_MLS_Income_by_Neighborhood_Transform_Load.ipnyb
- **Community data:** 4b_MLS_Income_by_Community_Transform_Load.ipnyb

Data loading

Below steps were performed to load household income data for MLS neighborhoods and communities:

3. Postgres tables: household_income_by_neighborhood & household_income_by_community were created using the SQL script generated from the ERD diagram.
4. Household income data for neighborhoods and communities were loaded to their respective tables using SQL Alchemy through Python/pandas.

Jupyter notebooks:

- **Neighborhood data:** 3b_MLS_Income_by_Neighborhood_Transform_Load.ipnyb
- **Community data:** 4b_MLS_Income_by_Community_Transform_Load.ipnyb

Web Application

The app summarizes police incidents/ use of force by police/ demographics (race and household income) for Minneapolis and as well as individual neighborhoods in Minneapolis by Year(>=2015). App file name: app.py

The app gets its data from views built in Postgres (script loc: SQL\View_Scripts_for_App) on top of the tables.

The app has two routes as below:

1. **Root path (\)** – This uses **index.html** in templates folder to display data.

Snapshot of Root (Home page) at a glance:


Minneapolis Crime Rates & Demographics

[Home](#)

Crime Rates and Demographics Stats

Year	Police Incidents Count	Use of Force Cases	% White Use of Force	% Of Color Use of Force	% White (Demographics)	% Of Color (Demographics)	Median Household Income
2015	19,971	2,135	26	68	59	41	58,025.14
2016	20,107	2,290	20	71	60	40	58,185.68
2017	21,922	2,017	24	68	61	39	59,049.65
2018	19,203	2,076	26	64	61	39	58,761.84
2019	22,978	2,204	24	65	61	39	58,182.53
2020	8,528	879	22	58	60	40	58,646.87

Neighborhood Stats

Armatage 

Search

Top 5 neighborhoods with most Use of Force (Year >= 2012)

Neighborhood	Use of Force Cases	% White - Use of Force	% Of Color - Use of Force	% Race Unknown - Use of Force	% White (Demographics)	% Of Color (Demographics)	Median Household Income	Income Group
Downtown West	4,520	23	71	6	60	40	60,383	Middle Income
Hawthorne	809	10	83	7	22	78	40,378	Lower Middle Income
Jordan	801	10	87	3	18	82	45,907	Lower Middle Income
Near - North	788	9	85	6	15	85	47,087	Lower Middle Income
Folwell	580	13	85	2	33	67	40,938	Lower Middle Income

Root page broken down into individual components:

1-a. Jumbotron component with Home button: Summary header with a link to Home page (root)

1-b. Table summarizing Total police incidents in Minneapolis by Year along with police use of force data split by percentages for use of force race and demographics data (race and median household income).

Underlying view: `vw_minneapolis_stats`

1-a

Minneapolis Crime Rates & Demographics

Home

Crime Rates and Demographics Stats

Neighborhood Stats

1-b

Year	Police Incidents Count	Use of Force Cases	% White Use of Force	% Of Color Use of Force	% White (Demographics)	% Of Color (Demographics)	Median Household Income
2015	19,971	2,135	26	68	59	41	58,025.14
2016	20,107	2,290	20	71	60	40	58,185.68
2017	21,922	2,017	24	68	61	39	59,049.65
2018	19,203	2,076	26	64	61	39	58,761.84
2019	22,978	2,204	24	65	61	39	58,182.53
2020	8,528	879	22	58	60	40	58,646.87

Armatage

Search

1-c: Table summarizing data for maximum police use of force in top 5 neighborhoods in Minneapolis along with Demographics data.

Underlying view: **vw_police_use_of_force_summary**

Top 5 neighborhoods with most Use of Force (Year >= 2012)

Neighborhood	Use of Force Cases	% White - Use of Force	% Of Color - Use of Force	% Race Unknown - Use of Force	% White (Demographics)	% Of Color (Demographics)	Median Household Income	Income Group
Downtown West	4,520	23	71	6	60	40	60,383	Middle Income
Hawthorne	809	10	83	7	22	78	40,378	Lower Middle Income
Jordan	801	10	87	3	18	82	45,907	Lower Middle Income
Near - North	788	9	85	6	15	85	47,087	Lower Middle Income
Folwell	580	13	85	2	33	67	40,938	Lower Middle Income

1-d: Dropdown menu listing out all neighborhoods in Minneapolis. Based on the neighborhood selected by the user, the app will redirect to another page summarizing police incidents and the demographics data for the selected neighborhoods in Minneapolis by Year (≥ 2015).

Underlying table: **neighborhood**

Section in app code redirecting the user input to the selected neighborhood dynamically:

```
# create route that renders index.html template
@app.route("/", methods=['GET', 'POST'])
def echo():
    if request.method == 'POST':
        newNeighborhood = request.form['neighborhood']

        return redirect("/") + newNeighborhood
```

Webpage with drop down menu:

1-d

Neighborhood Stats

ics)	% Of Color (Demographics)	Median Household Income
	41	58,025.14
	40	58,185.68
	39	59,049.65
	39	58,761.84
	39	58,182.53
	40	58,646.87

Armatage

Armatage

Audubon Park

Bancroft

Beltrami

Bottineau

Bryant

Bryn Mawr

Camden Industrial Area

Cedar - Isles - Dean

2. **Dynamic Path (to each neighborhood based on user selection in the dropdown in Home page)** - This uses **neighborhood.html** in templates folder to display data.

Snapshot of app route -

```
@app.route("/<neighborhood>")
def neighborhood_data(neighborhood):
```

Below snapshot showing sample webpage where **neighborhood = 'Downtown West'**

2-a: Name of the selected neighborhood (dynamically populated)

2-b: Home button: Button redirecting to the Home page

2-c: Table summarizing total police incidents in the **selected neighborhood** in Minneapolis by 'Year' along with police use of force data split by percentages for use of force race and demographics data (race and median household income).

Underlying view: vw_mls_neighborhood_stats

Downtown West

Home

Crime Rates and Demographics Stats

Year	Police Incidents Count	Use of Force Cases	% White Use of Force	% Of Color Use of Force	% White (Demographics)	% Of Color (Demographics)	Median Household Income	Income Group
2015	2,529	546	26	69	60	40	60,383	Middle Income
2016	2,347	770	18	73	60	40	60,383	Middle Income
2017	2,273	511	22	71	60	40	60,383	Middle Income
2018	1,882	528	21	73	60	40	60,383	Middle Income
2019	2,612	456	25	64	60	40	60,383	Middle Income
2020	593	140	29	62	60	40	60,383	Middle Income