US Communities and Crime Database

Asta Adhira Anggono, Maria Pilar Bifaretti, Joyce Megumi Ishikawa,
Michael Popper, Valeria Roberto Vargas, Yijing Sun

Columbia University

APAN 5310: SQL
Professor Michael J. Coakley
April 25, 2023

Client Scenario:

As a team of database experts hired by the US Government to access the Communities and Crime Unnormalized Data Set (UCI machine learning repository, 1995), our mission is to provide valuable insights to our clients to help develop new government policies and allocate resources effectively to reduce crime rates across the country. We understand that crime has both short and long-term impacts on society, including hindering human capital development, reducing investment in communities, increasing corruption, and eroding trust in government. Hence, we recognize the importance of reducing crime rates as it leads to multiple benefits for society.

Our decision is supported by research that highlights the potential gains of reducing crime rates. For instance, according to Shapiro and Hassett (2012), a 10% decrease in homicides can increase housing values by 0.83%. With this motivation, we stay committed to leveraging the power of data to make our communities safer and more secure.

According to our client's needs, we are focusing on ensuring that the crime data is well-organized, easily accessible for analysis, stored and retrieved effectively, and that both analyst level and C-level audiences can access the data in different ways.

To support C-level decision-making processes, our project aims to answer critical questions related to crime such as the primary correlated factors, demographic effects (including age, income, education, and race), and the impact of the police structure on crime rates. This will be done by creating visualizations and interactive dashboards that update automatically with new data. We intend to leverage the geographical data to showcase the crime rate disparities across various US states using a geospatial graph, while utilizing the demographic data to present crime patterns alongside other variables in an interactive dashboard. Analysts will gain permissions to query the data stored in the PostgreSQL database through the Metabase graphical user interface. There they can load any additional relevant data to the Metabase dashboard, or view necessary tables for further analysis. Thus we have streamlined our data processing efficiency by focusing only on the factors of interest and reducing extraneous data processing. Our dashboard will make the data relationship more transparent and understandable, empowering our final users to raise potentially insightful queries for future analyses and policy recommendations.

Team Structure & Timeline:

When it comes to building an effective and efficient team, having a clear team contract and structure in place is essential. We started the project with creating a team contract that outlines the expectations and responsibilities of each team member to make sure that decision, communication flow, and tasks are delegated properly and equally.

We have divided our timeline into 5 different checkpoints, excluding our end product. Below is a snapshot of the team contract that we agreed on at the beginning of our project:

	Asta A	Joyce I	Maria P	Michael P	Valeria R	Yijing S	Total	Deadline
Checkpoint 1								**3/8/2023**
Datasets & Idea Brainstorming (Meet to decide)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%	100%	3/4/2023
Reasoning & motivation behind our choice		50.0%	50.0%				100%	3/7/2023
Datasets description	50.0%				50.0%		100%	3/7/2023
Initial action plan				50.0%		50.0%	100%	3/7/2023
Checkpoint 2: Formal dataset submission								**3/22/2023**
Team Contract (Meet to decide)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%	100%	3/15/2023
Preliminary draft of normalization design		33.3%	33.3%	33.3%			100%	3/20/2023
Description: source, type, and extent	33.3%				33.3%	33.3%	100%	3/20/2023
Checkpoint 3: Schema								**3/29/2023**
Team Meeting (review shcema)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%	100%	3/27/2023
Database Schema (constraints, schema)		33.3%	33.3%	33.3%			100%	3/27/2023
SQL Code	33.3%				33.3%	33.3%	100%	3/27/2023
Checkpoint 4: Submit the Data Plan for Review								**4/5/2023**
Team Meeting	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%	100%	4/3/2023
Python/R scripts/(SQL code)			33.3%		33.3%	33.3%	100%	4/3/2023
Plan explanation & reasoning	33.3%	33.3%		33.3%			100%	4/3/2023
Checkpoint 5: Submit the Customer Interaction Plan for Review								**4/12/2023**
Team Meeting	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%	100%	4/5/2023
Specific interaction Plan for analysts	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%	100%	4/3/2023
FINAL SUBMISSION - DUE 4/25/2022								
Project Report								4/25/2023
Problem statement		50.0%	50.0%				100%	4/23/2023
Proposal	50.0%				50.0%		100%	4/23/2023
Team structure and Timeline				50.0%		50.0%	100%	4/23/2023
Database Schema		33.3%	33.3%	33.3%			100%	4/23/2023
Analytics Applications				33.3%	33.3%	33.3%	100%	4/23/2023
Build Metabase Applications	33.3%	33.3%	33.3%				100%	4/23/2023
Presentation Slides								4/25/2023
State consultant/client scenario		50.0%	50.0%				100%	4/23/2023
Show a sample of the original data	33.3%		33.3%		33.3%		100%	4/23/2023
Overview and brief explanation of normalization plan		33.3%	33.3%	33.3%			100%	4/23/2023
Brief explanation of the ETL process				50%	50%		100%	4/23/2023
Process of interacting with data	50%					50%	100%	4/23/2023
Very brief demo of database interaction	50%					50%	100%	4/23/2023
PPT Design			100.0%				100%	4/24/2023

Though we have agreed to minimally divert from our team contract, we needed to make some significant changes that were necessary due to unforeseen circumstances. As the project progressed, we have received new information, changes in requirements, as well as clarifications that led us in modifying our roles and responsibilities to areas where we excel. Instances where we have diverted from our initial contract were in checkpoint 4 and the final submission of the report as well as slides. We initially divided the team into 2 to work on the ETL python code and the other team on the explanation & reasoning. Since we have 44 tables that we need to populate, it makes no sense to only let half the team members do the work. We decided that each member should take an equal amount of tables, write the ETL python script, and combine it at the end. Another time where we had to divert from the initial contract was the end product. Turns out the workload in making the dashboard was on the heavier side and that the data runs locally, so we appointed 2 members to work on the dashboard while the rest of us created the final report and presentation slides. Our whole team was flexible and adaptable enough to ensure that our objectives remain aligned and productive throughout the project's lifespan.

The most challenging part of the project was the normalization process of the data as well as creating the final metabase dashboard. The normalization process of our data was difficult due to the complexity and size of our data. Our data was diverse with different formats, units, and scales. It needed careful planning and attention to detail in order to ensure the data met the requirements of first, second, and third normal form. Moreover, normalizing data is often subjective (different types of attributes may be appropriate for different research questions), making it important to carefully consider which approach was best for us.

Creating the final Metabase dashboard was a challenging task since it was our first time using it. Though it is a powerful tool for data exploration and visualization, the interface is quite overwhelming and terminologies were foreign. We also needed knowledge of server configurations in order to give permission to our database. Overall, Metabase has a lot of features and functionalities that takes time to learn, resulting in a steep learning curve.

Our communication protocols as a team were very effective and every team member was proactive. We held weekly in-person meetings on Wednesdays and communicated actively through a WhatsApp chat for regular updates. Additionally, we emailed internal stakeholders whenever we were not confident to gauge how well we were progressing throughout the project.

Overall, each team member was very accountable and none of us failed to meet our deadlines and responsibilities that resulted in smooth decision making. In conclusion, the team contract and structure that we followed was a critical component in the success of our project by working efficiently, minimizing conflicts, and achieving our objectives.

Dataset:
Link to full dataset

Sample of the Original Data: US Communities & Crime

Êcommunityname	state	countyCode	communityCode	fold	population	householdsize	racepctblack	racePctWhite	racePctAsian
BerkeleyHeightstownship	NJ	39	5320	1	11980	3.1	1.37	91.78	6.5
Marpletownship	PA	45	47616	1	23123	2.82	0.8	95.57	3.44
Tigardcity	OR	?	?	1	29344	2.43	0.74	94.33	3.43
Gloversvillecity	NY	35	29443	1	16656	2.4	1.7	97.35	0.5
Bemidjicity	MN	7	5068	1	11245	2.76	0.53	89.16	1.17
Springfieldcity	MO	?	?	1	140494	2.45	2.51	95.65	0.9
Norwoodtown	MA	21	50250	1	28700	2.6	1.6	96.57	1.47
Andersoncity	IN	?	?	1	59459	2.45	14.2	84.87	0.4
Fargocity	ND	17	25700	1	74111	2.46	0.35	97.11	1.25
Wacocity	TX	?	?	1	103590	2.62	23.14	67.6	0.92
Shermancity	TX	?	?	1	31601	2.54	12.63	83.22	0.77
SanPablocity	CA	?	?	1	25158	2.89	21.34	49.42	17.21
BowlingGreencity	KY	?	?	1	40641	2.54	12.18	86.39	1.12

We have taken our data from the UCI Machine Learning Repository. Michael Redmond, a computer science member of La Salle University in Philadelphia, USA, compiled real data

related to crime and demographics in the USA to create the 'Communities and Crime Unnormalized Data Set'. This dataset is a collection of socio-economic and crime statistics for a selection of cities in the United States. Specifically, it includes community data, law enforcement data, and crime data sourced from the US Census of 1990, the US FBI Uniform Crime Report of 1995, and the US Law Enforcement Management and Administrative Statistics Survey of 1990.

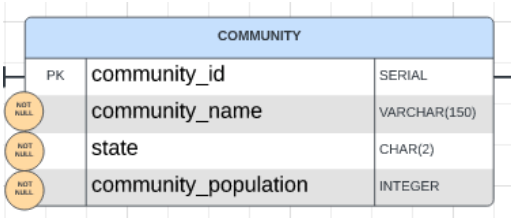
The dataset contains a total of 147 attributes, including 125 predictive attributes and 4 non-predictive attributes. The predictive attributes include demographic and socio-economic data such as age, education, income, employment, and race, as well as crime-related data such as the number of homicides, burglaries, and drug-related offenses. The non-predictive attributes include city and state identifiers, and a community name, and other geographic descriptors. There were two types of data, which were absolute and percentage values; for instance the black race attribute was expressed as blackPerCap and racepctblack. We kept the percentage values because it provided more context than an absolute value. In addition, absolute values that were not able to be expressed as a percentage such as median income were kept.

We have chosen this data due to a few number of reasons. Firstly, the relevance of this data. It contains a comprehensive collection of socioeconomic and crime statistics that is useful for analyzing crime trends and patterns, as well as identifying factors that contribute to crime. Secondly, the accessibility of this data. The data is publicly available through the UCI Machine Learning Repository making it easy for us as database experts to access and analyze. Thirdly, the scope of this data. It contains a large number of attributes that includes both demographic and crime related data which gives us a bigger room for analyzing crime trends and patterns. Lastly, the potential impact this dataset could give. With the help of our team, the dataset will provide useful insights that we can inform the Government to revise policies and allocate resources to address crime-related issues.

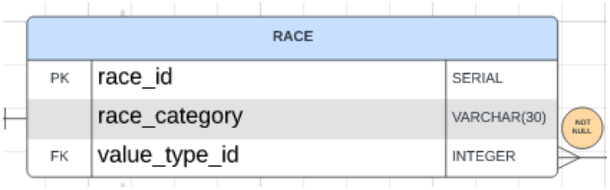
Normalization Plan:

We have normalized the dataset into the third normal form, from one table containing 147 attributes into **44 distinct tables** that are connected through various foreign key relationships.

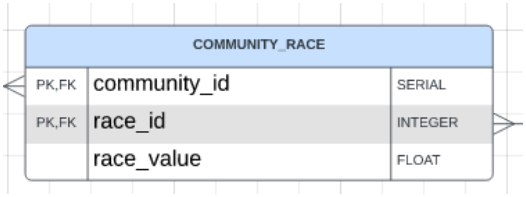
The **main table in this schema is the "community" table,** which contains information about each community, such as the community ID, name, state and population. We have decided to drop the community_code and county_code as we found it unrelated to the insights we intend to derive from the dataset.



Other tables in the schema contain information about demographic factors such as age, income, education, family, immigration, language, race, and employment. Each table will contain a unique *[factor]_id* (primary key), the corresponding *[factor]_category*, and the *value_type_id* (defined below).

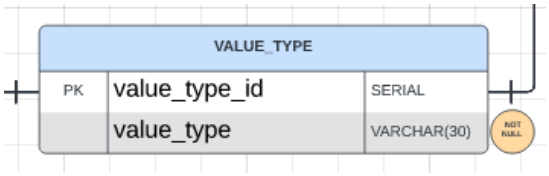


We then created an intermediary table "community_[factor]" for each demographic information table (for example, for the table "race", we create "community_race"). This table has information on each community's value for each factor in the demographic table. Continuing the "community_race" example, each *community_id* would have four observations in the "community_race" table, as there are 4 categories of race defined in the "race" table. These tables would have a combination primary key of the *community_id* and the *[factor]_id* as *race_id*. The final column will be *[factor]_value* as *race_value*.



In addition, a "value_type" table was created to store the value type of a value with *value_type_id* and *value_type*. This table was created because a table can have different forms of data types such as percent with two decimals and integer values. For example, in the "community" table, there is a category "pctWWage" which is the percentage of the population with a wage of 80.99. There is also a category "medIncome" as an integer type with a value of 89,000. These are two different data type values that need to be stored under the *"[factor]_value"* column in the "community_income" table. In order to differentiate between the percent and integer data types, the *value_type_id* is a foreign key in the "[factor]" tables ("income" table) to denote if the *[factor]_id* ("income_id") is a percent or an integer. This will

allow for efficient querying of the different value types using the *value_type_id* if we want to look at only percent values or vice versa.



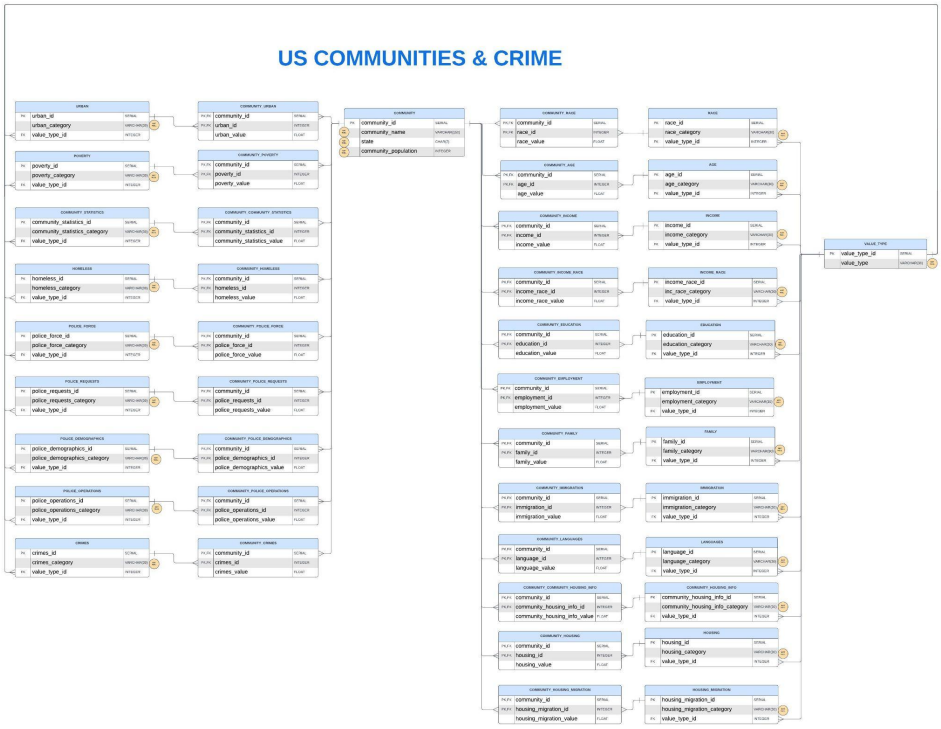
The **integrity constraints** in the schema are defined by the foreign key relationships between the tables, which ensure that data entered into each table is consistent with the values in the related tables.

There are **one-to-many relationships** between the “community” table and the other tables. For example, the “community_race” table links the “community” table with the “race” table, and a community can have multiple races, and a race value can only belong to a certain community.

There is a one-to-many relationship between the "community_[factor]" and the corresponding “[factor]” table. A demographic category can have multiple IDs but an ID can only belong to a specific demographic category. For example; in the case of “race” a community can have Black, Asian, Latino, or White. Each with a specific *race_id*, and a *race_id* can only belong to one race.

Overall, this schema is designed to **store a large amount of data** about crime and demographics in communities across the US and **provides a structure for efficiently organizing and querying this data.**

ER Diagram: Link to LucidChart & Link to ER diagram PDF



Extract Transform & Load (ETL):

The link to ETL Python Code is found here.

We began the ETL process by importing the necessary libraries on Python such as pandas, sqlalchemy, psycopg2 and numpy to ensure that we had all the tools we needed to proceed.

```
import pandas as pd
from sqlalchemy import create_engine, text
import psycopg2
import numpy as np
```

Then, we created the database by connecting to PgAdmin and created the tables based on the ER diagram, specifying the data types, integrity constraints, and other important details as shown below.

Create Database Tables

```
conn = psycopg2.connect(
    host="localhost",
    port='5432',
    database="postgres",
    user="postgres",
    password="123")

# Set autocommit to true to create database
conn.autocommit = True

# Create a new database
db_name = "crime_data_5310"
cur = conn.cursor()
cur.execute(f"CREATE DATABASE {db_name}")

print(str(db_name)+' database has been successfully created in PostgreSQL.')
```

```
crime_data_5310 database has been successfully created in PostgreSQL.
```

```
# Pass the connection string to a variable, conn_url
conn_url = 'postgresql://postgres:123@localhost:5432/crime_data_5310'

# Create an engine that connects to PostgreSQL server
engine = create_engine(conn_url)

# Establish a connection
connection = engine.connect()
```

```
# Create commands
stmt = (
"""
CREATE TABLE community (
  community_id serial PRIMARY KEY,
  community_name varchar(150) NOT NULL,
  state char(2) NOT NULL,
  community_population int NOT NULL
);

""",
"""
CREATE TABLE value_type (
  value_type_id serial PRIMARY KEY,
  value_type varchar(30) NOT NULL
);

""",
"""

CREATE TABLE race (
  race_id serial PRIMARY KEY,
  race_category varchar(30) NOT NULL,
  value_type_id int,
  FOREIGN KEY (value_type_id) REFERENCES value_type
);

""",
"""
CREATE TABLE community_race (
  community_id serial,
  race_id int,
  race_value float,
  PRIMARY KEY (community_id, race_id),
  FOREIGN KEY (race_id) REFERENCES race,
    FOREIGN KEY (community_id) REFERENCES community
);

"""
```

We loaded the crimedata.csv file into Python using pd.read_csv().

Loading Dataset

```
df1 = pd.read_csv('crimedata.csv', encoding='latin-1')
df1.head()
```

	Êcommunityname	state	countyCode	communityCode	fold	population	householdsize	racepctblack	racePctWhite	racePctAs
0	BerkeleyHeightstownship	NJ	39	5320	1	11980	3.10	1.37	91.78	6
1	Marpletownship	PA	45	47616	1	23123	2.82	0.80	95.57	1
2	Tigardcity	OR	?	?	1	29344	2.43	0.74	94.33	1
3	Gloversvillecity	NY	35	29443	1	16656	2.40	1.70	97.35	0

Then we cleaned the dataset by renaming column names that contained types or unnecessary symbols such as “Êcommunityname” and converting all the values into float type.

This step was important because we had both percentages and integers, and the goal of converting to float was to create a “value_type” table that could generate an ID for variables with percentage, categorical, and/or integer values.

Moreover, missing values (?) were converted into -1 for easy identification to avoid errors due to “?” and “Nan” values when inserting data into our SQL database.

Clean Dataset

```
df1 = df1.rename(columns = {"Êcommunityname":"communityname"})
```

Convert object type to float and replace ? with -1 as default value for null

```
[ ]  for col in df1.columns:
         if df1[col].dtype == object:
             if '?' in df1[col].unique():
                 df1[col] = df1[col].replace('?', -1)
             try:
                 df1[col] = df1[col].astype(float)
             except ValueError:
                 pass
         elif df1[col].dtype == int:
             df1[col] = df1[col].astype(float)
```

```
[ ]  df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2215 entries, 0 to 2214
Columns: 147 entries, communityname to nonViolPerPop
dtypes: float64(145), object(2)
memory usage: 2.5+ MB
```

This allowed us to easily filter out missing values by filtering -1 in the queries. The “Nan” values caused errors due to it being recognized as a string. In addition, we inserted a column named *community_id* to the dataset as the unique identifier of each community.

Since each row is a unique community, create a community_id column

```
[ ]  df1.insert(0, 'community_id', range(1, 1 + len(df1)))
```

This is because each row in the dataset represented a unique community. Also, this allowed the creation of foreign key constraints between the table *community* and the intermediary tables “community_[factor]” using *community_id*.

Once the dataset was cleaned and converted, we began the ETL process by creating the pandas dataframes based on the schema, creating the corresponding IDs, joining necessary tables, and loading the populated dataframes from Python into the SQL database tables.

Firstly the “community” dataframe was made by slicing columns related to community such as *community_id*, *communityname*, etc, and renaming columns from the cleaned dataset, as this was going to be needed for all of the intermediary tables “community_[factor]” dataframes we would create.

Slice necessary columns

```
[ ]  df_community = df1[['community_id','communityname', 'state', 'population']]
```

Rename columns

```
[ ]  df_community = df_community.rename(columns ={"communityname": "community_name",
                                                  "population": "community_population"})
```

Insert into df_community table in database

```
[ ]  df_community.to_sql(name='community', con=engine, if_exists='append', index=False)

     215
```

We created the "value_type" table from scratch, creating IDs for percentages and integers. To reference the two types of values that this dataset contains.

Create df_value_type

```
[ ]  df_value_type = pd.DataFrame({
       'value_type_id': [1, 2],
       'value_type': ['Percentage', 'Integer']})
```

Then, we moved onto a repeatable process for the 42 dataframes we needed to make based on the ER-diagram.

The following example is based on the tables "race" and "community_race".

First, we created a new race_df dataframe with a column *race_category* to contain the unique race related columns in our data such as *"racepctblack", "racePctwhite"*, etc, by slicing them using .iloc(). Then, we created a column for *race_id* which acted as the primary key.

Create df_race

```
[ ]  df_race = pd.DataFrame()
```

Slice necessary columns and convert to list as string for category column

```
[ ]  df1.iloc[0:5, 8:12]
     race_category = df1.iloc[0:5, 8:12].columns.tolist()
     print(race_category)

     ['racepctblack', 'racePctWhite', 'racePctAsian', 'racePctHisp']
```

```
[ ]  df_race['race_category'] = race_category
```

Insert ID column

```
[ ]  df_race.insert(0, 'race_id', range(1, 1 + len(df_race)))
```

Next, we created a *value_type_id* by matching strings that contain *"pct"* or *"Pct"* as a percentage (value_type_id: 1). Finally, we inserted the *df_race* dataframe into the corresponding SQL table "race" using the .to_sql() method.

Create a value_type_id. If category contains pct then value_type_id == 1, else 2 for integer

```
[ ] df_race['value_type_id'] = np.where(df_race['race_category'].str.contains('pct|Pct'), 1, 2)
```

```
[ ] df_race.head()
```

	race_id	race_category	value_type_id
0	1	racepctblack	1
1	2	racePctWhite	1
2	3	racePctAsian	1
3	4	racePctHisp	1

Insert into race table in database

```
[ ] df_race.to_sql(name='race', con=engine, if_exists='append', index=False)
```

4

Next, we created the intermediary table "community_race" dataframe by slicing the necessary columns from the cleaned dataset based on our normalization plan. We used .melt() to pivot longer the sliced columns so that there was only a column of *race_category* and their values.

Create df_community_race

Slice necessary columns

```
[ ] df_community_race = df1.iloc[:, [0,8,9,10,11]]
```

pd.melt() to pivot longer

```
[ ] df_community_race = pd.melt(df_community_race, id_vars = ['community_id'], value_vars=race_category)
```

```
[ ] df_community_race.head()
```

	community_id	variable	value
0	1	racepctblack	1.37
1	2	racepctblack	0.80
2	3	racepctblack	0.74
3	4	racepctblack	1.70
4	5	racepctblack	0.53

Next, we renamed the variable column to *race_category* and the *value* column to *race_value*.

Rename columns

```
[ ] df_community_race = df_community_race.rename(columns = {"variable": "race_category",
                                                            "value": "race_value"})
```

We joined the *race_df* dataframe with the *df_community_race* on *race_category* to ensure the *race_id* column was present in the *df_community_race* dataframe.

Join race_id by race_category

```
[ ] df_community_race = df_community_race.merge(df_race[['race_category','race_id']], on='race_category')
    df_community_race = df_community_race.drop(columns = ['race_category'])
```

```
[ ] df_community_race.head()
```

	community_id	race_value	race_id
0	1	1.37	1
1	2	0.80	1

We reordered the columns that match the schema and inserted the *df_community_race* into the corresponding SQL table "community_race" using the .to_sql() method

Reorder the column to be community_id, race_id, race_value

```
[ ] df_community_race = df_community_race.iloc[:, [0,2,1]]
```

```
[ ] df_community_race.head()
```

	community_id	race_id	race_value
0	1	1	1.37
1	2	1	0.80
2	3	1	0.74

Insert into community_race table in database

```
[ ] df_community_race.to_sql(name='community_race', con=engine, if_exists='append', index=False)
```

```
860
```

Finally, we managed to run the whole process on the remaining tables with no errors and closed the connection to the database, ensuring that there were no resource leaks and that all the data was safely stored.

Close Connection & Engine

```
[ ] #Close connection and engine to avoid resource leaks.

    # Close the connection
    connection.close()

    # Close the engine
    engine.dispose()
```

Analytics Application:

The following are the 20 analytical procedures queried using the PostgreSQL interface on Metabase. The double bracket syntax "{{}}" is to allow for the dynamic selection of a specific value in an attribute via Metabase. The *[factor]_value* attribute always filters out null values denoted as *-1* using a *where* clause *>= 0.*

1. What is the average number of crimes by crime category in each state?

```
select community.state, crimes.crimes_category, AVG(community_crimes.crimes_value)
from community
join community_crimes on community.community_id = community_crimes.community_id
join crimes on community_crimes.crimes_id = crimes.crimes_id
where {{crimes_category}}
where community_crimes.crimes_value >=0
group by community.state, crimes.crimes_category, community_crimes.crimes_value;
```

2. What is the percentage of the population of each race in each state?

```
select c.state, r.race_category, cr.race_value
from community as c
join community_race as cr on c.community_id = cr.community_id
join race as r on cr.race_id = r.race_id
where  cr.race_value >= 0
group by c.state, r.race_category, cr.race_value
```

3. What is the median household income in each state?

```
select community.state, income.income_category, community_income.income_value
from community
join community_income on community.community_id = community_income.community_id
join income on community_income.income_id = income.income_id
where {{state}}
where  income.income_category = 'medIncome'
where community_income.income_value >= 0
group by community.state, income.income_category, community_income.income_value
```

4. What is the per capita income for each race in each state?

```
select community.state, income_race.inc_race_category, community_income_race.income_race_value
from community
join community_income_race on community.community_id = community_income_race.community_id
join income_race on community_income_race.income_race_id = income_race.income_race_id
where {{state}}
where community_income_race.income_value >= 0
group by community.state, income_race.inc_race_category, community_income_race.income_race_value
```

5. What is the percentage of the population in each education category by state?

```
select community.state, education.education_category, community_education.education_value
```

```
from community
join community_education on community.community_id = community_education.community_id
join education on community_education.education_id = education.education_id
where {{state}}
where community_education.education_value >= 0
group by community.state, education.education_category, community_education.education_value
```

6. What is the percentage of the population who are unemployed and employed in each state?

```
select community.state, employment.employment_category, community_employment.employment_value
from community
join community_employment on community.community_id = community_employment.community_id
join employment
on community_employment.employment_id = employment.employment_id
where {{state}}
where employment.employment_category = PctUnemployed and employment.employment_category =PctEmploy
where community_employment.employment_value >= 0
group by
community.state, employment.employment_category, community_employment.employment_value
```

7. What is the average number of people per family in each state?

```
select community.state, family.family_category, community_family.family_value
from community
join community_family on community.community_id = community_family.community_id
join family on community_family.family_id = family.family_id
where {{state}}
where family.family_category = "PersPerFam"
where community_family.family_value >= 0
group by community.state, family.family_category, community_family.family_value
```

8. What is the percentage of the population that are foreign born in each state?

```
select community.state, immigration.immigration_category, community_immigration.immigration_value
from community
join community_immigration on community.community_id = community_immigration.community_id
join immigration on community_immigration.immigration_id = immigration.immigration_id
where {{state}}
where  immigration.immigration_category = "PctForeignBorn"
where community_immigration.immigration_value >=0
group by community.state, immigration.immigration_category, community_immigration.immigration_value
```

9. What is the percentage of the population that speaks English or a foreign language in each state?

```
select community.state, languages.language_category, community_languages.language_value
from community
join community_languages on community.community_id = community_languages.community_id
```

join languages on community_languages.language_id = languages.language_id
where {{state}}
where community_languages.language_value >= 0
group by community.state, languages.language_category, community_languages.language_value

10. What is the mean people per household in each state?

select community.state, community_housing_info.community_housing_category,
community_community_housing_info.community_housing_info_value
from community
join community_community_housing_info on community.community_id =
community_community_housing_info.community_id
join community_housing_info on community_community_housing_info.community_housing_info_id =
community_housing_info.community_housing_info_id
where {{state}}
where community_housing_info.community_housing_category = "householdsize"
where community_community_housing_info.community_housing_info_value >= 0
group by community.state, community_housing_info.community_housing_category,
community_community_housing_info.community_housing_info_value

11. What is the median gross rent in each state?

select community.state, housing.housing_category, community_housing.housing_value
from community
join community_housing on community.community_id = community_housing.community_id
join housing on community_housing.housing_id = housing.housing_id
where {{state}}
where housing.housing_category = "MedRent"
where community_housing.housing_value >= 0
group by community.state, housing.housing_category, community_housing.housing_value

12. What is the percent of the population born in the same state as currently living in each state?

select community.state, housing_migration.housing_migration_category,
community_housing_migration.housing_migration_value
from community
join community_housing_migration on community.community_id = community_housing_migration.community_id
join housing_migration on community_housing_migration.housing_migration_id =
housing_migration.housing_migration_id
where {{state}}
where housing_migration.housing_migration_category = PctBornSameState
where community_housing_migration.housing_migration_value >= 0
group by community.state, housing_migration.housing_migration_category,
community_housing_migration.housing_migration_value

13. What is the percentage of the population living in urban areas in each state?

select community.state, urban.urban_category, community_urban.urban_value

```
from community
join community_urban on community.community_id = community_urban.community_id
join urban on community_urban.urban_id = urban.urban_id
where {{state}}
group by community.state, urban.urban_category, community_urban.urban_value
```

14. What is the percentage of the population under the poverty level in each state?

```
select community.state, poverty.poverty_category, community_poverty.poverty_value
from community
join community_poverty on community.community_id = community_poverty.community_id
join poverty on community_poverty.poverty_id = poverty.poverty_id
where {{state}}
where  community_urban.urban_value >= 0
group by community.state, poverty.poverty_category, community_poverty.poverty_value
```

15. What is the population density in persons per square mile in each state?

```
select community.state, community_statistics.community_statistics_category,
community_community_statistics.community_statistics_value
from community
join community_community_statistics on community.community_id =
community_community_statistics.community_id
join community_statistics on community_community_statistics.community_statistics_id =
community_statistics.community_statistics_id
where {{state}}
where community_statistics.community_statistics_category = "PopDens"
where community_community_statistics.community_statistics_value >= 0
group by community.state, community_statistics.community_statistics_category,
community_community_statistics.community_statistics_value
```

16. What is the number of people who are homeless in shelters and homeless on the street in each state?

```
select community.state, homeless.homeless_category, community_homeless.homeless_value
from community
join community_homeless on community.community_id = community_homeless.community_id
join homeless on community_homeless.homeless_id = homeless.homeless_id
where {{state}}
where  community_homeless.homeless_value >= 0
group by community.state, homeless.homeless_category, community_homeless.homeless_value
```

17. What is the number of police forces in each police force category in each state?

```
select community.state, police_force.police_force_category, community_police_force.police_force_value
from community
join community_police_force on community.community_id = community_police_force.community_id
join police_force on community_police_force.police_force_id = police_force.police_force_id
where {{state}}
```

where community_police_force.police_force_value >= 0
group by community.state, police_force.police_force_category, community_police_force.police_force_value

18. What is the number of requests for police in each state?

select community.state, police_requests.police_requests_category,
community_police_requests.police_requests_value
from community
join community_police_requests on community.community_id = community_police_requests.community_id
join police_requests on community_police_requests.police_requests_id = police_requests.police_requests_id
where {{state}}
where community_police_requests.police_requests_value >= 0
group by community.state, police_requests.police_requests_category,
community_police_requests.police_requests_value

19. What is the percentage of the population that have a racial match between community and police force in each state?

select community.state, police_demographics.police_demographics_category,
community_police_demographics.police_demographics_value
from community
join community_police_demographics on community.community_id =
community_police_demographics.community_id
join police_demographics on community_police_demographics.police_demographics_id =
police_demographics.police_demographics_id
where {{state}}
where police_demographics.police_demographics_category = "RacialMatchCommPol"
where community_police_demographics.police_demographics_value >= 0
group by community.state, police_demographics.police_demographics_category,
community_police_demographics.police_demographics_value

20. What is the police operating budget per population in each state?

select community.state, police_operations.police_operations_category,
community_police_operations.police_operations_value
from community
join community_police_operations on community.community_id = community_police_operations.community_id
join police_operations on community_police_operations.police_operations_id =
police_operations.police_operations_id
where {{state}}
where police_operations.police_operations_category = "PolicBudgPerPop"
where community_police_operations.police_operations_value >= 0
group by community.state, police_operations.police_operations_category,
community_police_operations.police_operations_value

Customer Interaction Plan:

In order for our customers to interact with the data, we used Metabase to allow for C-level and analyst level interactions. The analytical procedures mentioned above are the queries for analysts that feed into a dashboard view intended for the C-level audience. By utilizing SQL queries, analysts can optimize the efficiency when processing a large scale of data, reviewing the previous work through documented queries, and integrating with different tools or platforms to meet specific needs of diverse audiences.

The technical audience such as the analyst level, will be provided appropriate access controls and permissions to interact and query the data. This will allow different groups of people access to the necessary tables. Secondly, they will be granted access to virtual relations or views via Metabase. To illustrate, this will grant the analysts access to data tables and interactive graphs, especially specific portions of the logical model, so that they do not have to see the entire model but just dynamically select different attributes. Also, the analysts will gain the permission to execute SQL queries in PostgreSQL and Metabase. It means that they will have direct access to the database through the SQL queries, which allows them to load the relevant data to the Metabase dashboard, or view necessary tables for further analysis. Lastly, limitations will be set on modifications or deletions of the data. Under this condition, the risk of potential modifications threatening the integrity of the information can be reduced. Apart from the accesses above, we will also provide the team with the proper documentation including an entity-relationship (ER) diagram, a database dictionary linked here, and query documentation linked here for better context of the database. Analysts will also be able to download the reports and 1 million rows of data directly from Metabase, allowing for easy sharing of insights with other stakeholders.

By designing the proper access structure for the analytical or technical audiences, our project will assist in answering relevant analytical questions. For example, we can focus on violent crimes, and visualize the raw values of different crime rates by state, and evaluate the correlation between demographic factors and crime numbers across the US. Thus, the analysts are able to identify potential trends or relationships that can be supportive in the decision-making process.

To cater to our C-suite audience, who might not be familiar with the raw data, our project will design a customized and interactive dashboard using Metabase. Our aim is to provide high-level summaries of key metrics related to crimes and other relevant data, so as to support C-level officers in their decision-making process. We will make the dashboard accessible to C-level managers through a secure web link to ensure appropriate access control and organizational permissions. This will enable even non-technical C-level managers to interact with the data in Metabase, view data patterns, and generate insights that can inform their decision-making process without having to execute any code.

The dashboard will tell a compelling story, answering key questions with clear and concise visualization that highlights patterns and correlations using intuitive visualizations such as maps, charts, and graphs to convey information in a visually appealing and easy-to-understand way. We expect this dashboard to effectively bring various insightful dimensions to policy-making that can reduce crime rates. For instance, it could help identify better ways to allocate resources, highlight the demographic factors most correlated with violent crimes for future research, and recognize potential societal reasons for committing violent crimes, and what other methods could be applied to cut the crime rate based on these societal reasons. As a result, the C-level officers might come up with not only the policies that relate to reallocation of police resources, but also those that have something to do with developing local economies to reduce violent crimes.

Redundancy/Performance Plan:

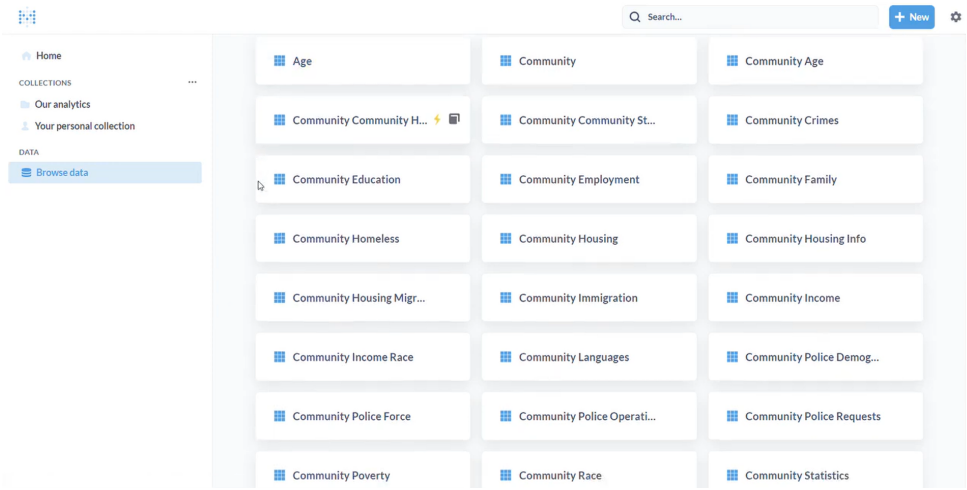
To ensure optimal performance we have optimized the database schema through normalization to the third normal form, created efficient queries, and set up indexing in our frequently queried columns (e.g. community table). In addition, since the database will currently be shared and accessed by only a relatively small number of people, we believe that the PostgreSQL database will be qualified to perform efficiently without heavy traffic.

Redundancy was minimized through the normalization of the relational database. By creating individual tables containing the column categories and data type to store the category name and data type we can prevent redundancy of the category name through foreign key relationships. In addition to minimizing data redundancy, we also planned to prevent data loss through backups. We plan to host the RDBMS on-premise with a commercial cloud solution such as Amazon Web Services (AWS). An on-premise cloud solution will provide us with controlled security of data access. In addition, it provides the benefits of cloud solutions which are scalability and data replication for backups to prevent data loss.

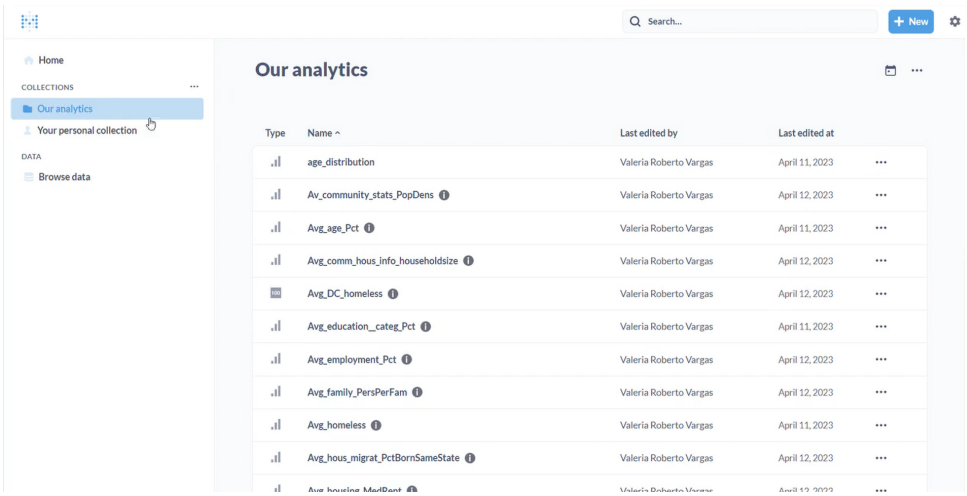
Dashboard Metabase:

The goal of our dashboard is to provide both analysts and C-suite employees with fast access to the data and recommended visualizations mentioned previously.

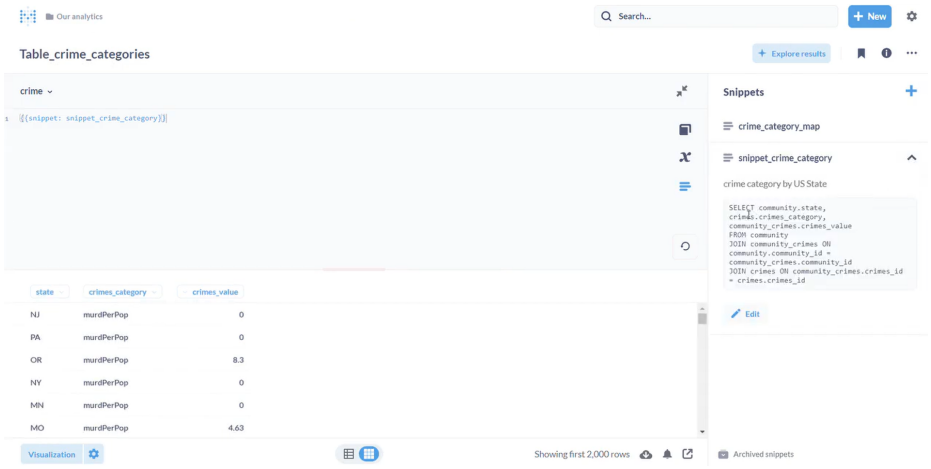
The next four screenshots will illustrate how analytical or technical teams can utilize their accessibility. As shown in the image below, analytical and technical teams can access the crime data via Metabase by navigating to the “Browse data” option in the menu and select the relational table of interest for further exploration.



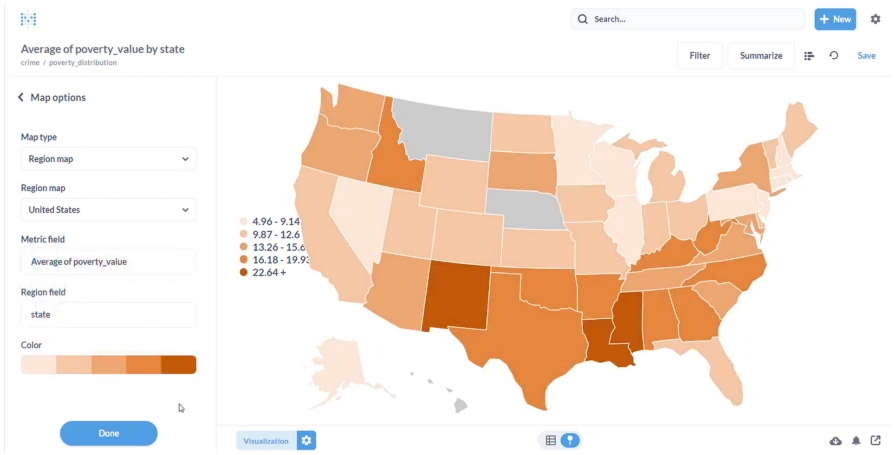
Once they want to dig deeper into specific relationship among several dimensions, analysts are able to create and manage their analysis process and data models in “Our analytics” page:



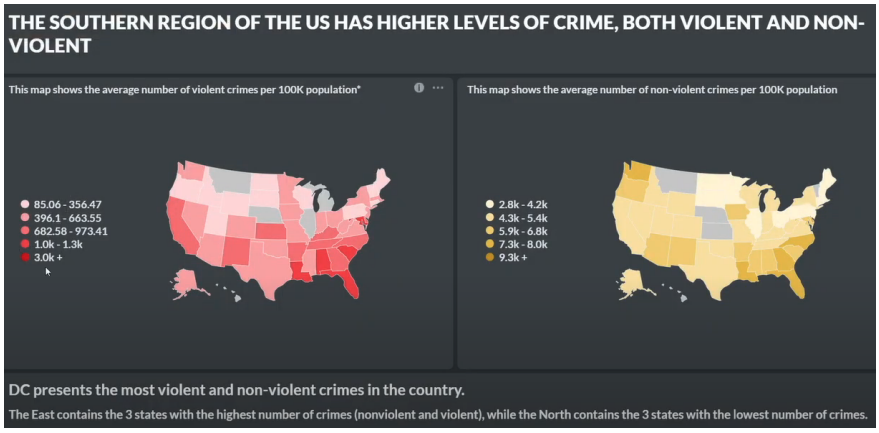
Next, they are able to execute their SQL queries after entering a specific table page by opening the editor shown in the picture below, going to SQL snippets, and then exploring results.



After running the SQL queries, the analytical team can then customize the visualizations to suit their specific analytical purposes by applying filters and visualizing effects in the menu. Although they are not able to manage the database, once satisfied with the results, they have access to download the reports and up to 1 million rows of data for further analysis or sharing insights with targeted stakeholders by clicking on the icons at the bottom right corner:

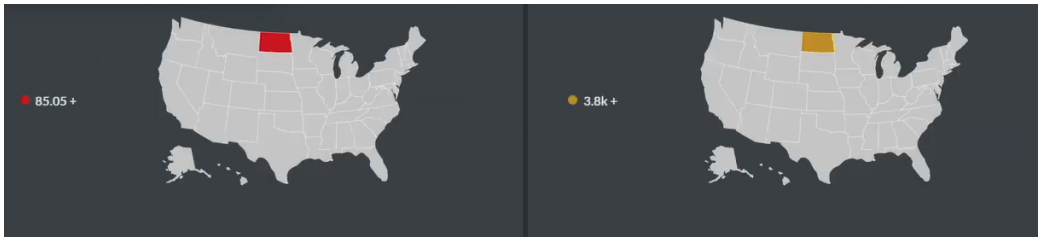


The intuitive dashboard begins with a comparison between violent and non-violent crime per capita across the US. The goal of this chart is to allow the user to notice discrepancies between states that have higher violent crime than non-violent crime in order to focus their efforts.

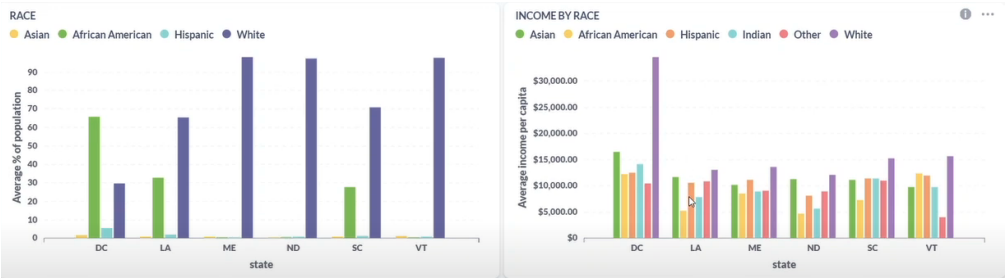


We wanted the higher level users to be directed to takeaways, so scrolling down the dashboard points to the 3 states with the highest and lowest violent/non-violent crimes, with background information:

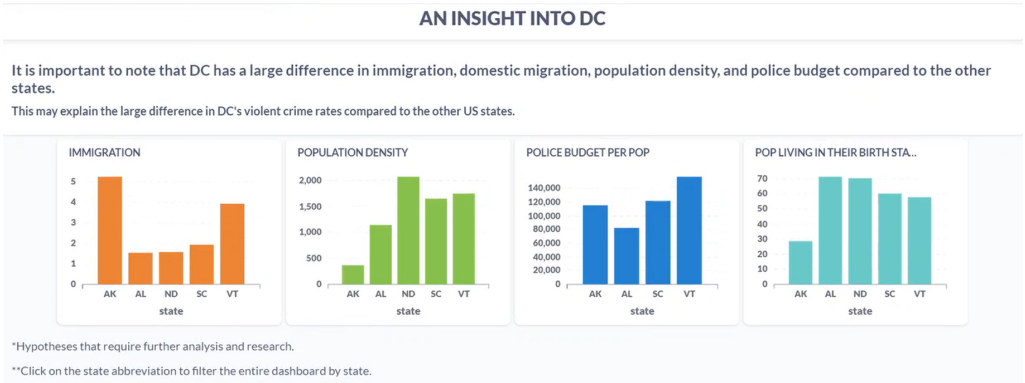
By clicking on an individual state abbreviation (ND in the example below), it is highlighted in the maps to allow for quick drill-downs:



Clicking on the violent-crime map brings you to a different dashboard, with more information on the demographic breakdown of the 3 highest and lowest violent crime states and their crimes. For example, race and income breakdown by state is provided, with the goal of giving a quick understanding of each states general demographic makeup:



After breaking down the demographic information, we then provide the user with pre-determined graphs that dive deeper into potential reasons behind the crime differences between states. Note that each of these graphs have downloadable CSVs that will allow the data analyst to perform any data analysis that they see fit. Two examples of analysis that we provide are below:



Finally, this page has a filter at the top, so that any combination of states can be viewed by the user. We wanted the user to control which states were provided so that they can make their own takeaways, rather than being forced to view the 6 states that we determined as important.

Conclusions:

Our goal throughout this project was to provide our client with access to an easy-to-understand, clean and effective database and visualization of crime in the United States so as to generate supportive insights for the policy-making process and reduce the crime rates across the US. We wanted to make sure that our interaction with all levels of users provided value. Finally, we wanted to make sure that our process was fast, repeatable and scalable.

We designed our RDMS and ETL steps to be extremely consistent. The schema of our database had 2 standalone tables ("community" and "value_type_id"), while the other 42 were repeated processes. This meant that we could divide and conquer the ETL process, cutting our time for development down immensely.

Our Metabase dashboard was laid out in a way that told a distinct story. We knew that our users would be a mix of technical or non-technical, and junior or senior employees, so having a predetermined analysis available meant that each user would derive value from the dashboard. Our technical data analysts have access to the database via PostgreSQL and Metabase, and can download the CSV files to run their own R or Python analyses on. Our senior C-suite users could follow along with the dashboard as we walked through the breakdown of violent or non-violent crime, and then drill down into potential causes of these discrepancies and revise relevant policies. Exemplary insights generated suggest a correlation between higher violent crime rates and higher unemployment rates or lower education levels. This could lead to the government launching financial assistance and affordable, high-quality education programs aimed at reducing violent crime rates in the US. Clearly our dashboard benefited our client by allowing for an effective story to be told through simple, easy-to-understand visualizations, while allowing for more data analysis to be done outside of the dashboard. Our RDMS and ETL process created a foundation for our dashboard to be built through 20 simple queries.

Overall, we were extremely satisfied with the result of our project. Our team worked seamlessly across multiple workflows. Each member of the team was dedicated to the project, taking part in any discussions, meeting deadlines and providing feedback on other's work.

References:

Shapiro, R., & Hassett, K. (2012, June 19). The Economic Benefits of Reducing Violent Crime. Retrieved from American Progress: https://www.americanprogress.org/article/the-economic-benefits-of-reducing-violent-crime/

UCI machine learning repository. (1995). Communities and Crime Unnormalized Data Set. Retrieved from UCI machine learning repository: http://archive.ics.uci.edu/ml/datasets/communities+and+crime+unnormalized#