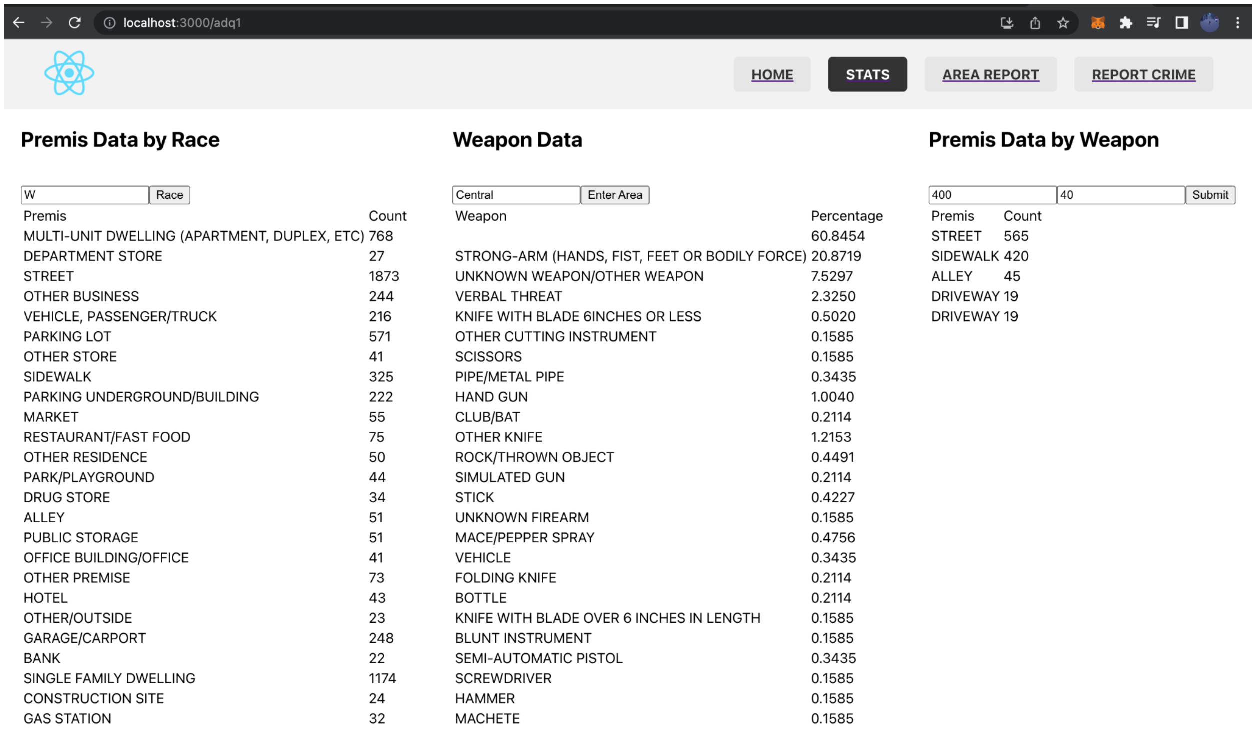
1. I believe that the direction of our project is mostly aligned with our final product. We used the same database as initially proposed, and the very same CRUD functionality as initially proposed. The only difference we have is the absence of the proposed graph feature, which we omitted due to difficulty reasons.

2. The application achieved almost everything we set out to achieve, which includes the advanced queries and the stored procedure that all have different functionalities regarding aiding the user to get specific information on specific categories whether that be sorted by premis, weaponID, or race.



Premis Data by Race

W | Race

Premis	Count
MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	768
DEPARTMENT STORE	27
STREET	1873
OTHER BUSINESS	244
VEHICLE, PASSENGER/TRUCK	216
PARKING LOT	571
OTHER STORE	41
SIDEWALK	325
PARKING UNDERGROUND/BUILDING	222
MARKET	55
RESTAURANT/FAST FOOD	75
OTHER RESIDENCE	50
PARK/PLAYGROUND	44
DRUG STORE	34
ALLEY	51
PUBLIC STORAGE	51
OFFICE BUILDING/OFFICE	41
OTHER PREMISE	73
HOTEL	43
OTHER/OUTSIDE	23
GARAGE/CARPORT	248
BANK	22
SINGLE FAMILY DWELLING	1174
CONSTRUCTION SITE	24
GAS STATION	32

Weapon Data

Central | Enter Area

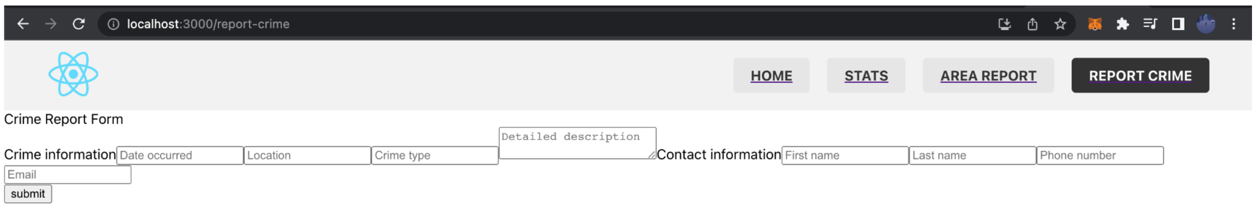
Weapon	Percentage
	60.8454
STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	20.8719
UNKNOWN WEAPON/OTHER WEAPON	7.5297
VERBAL THREAT	2.3250
KNIFE WITH BLADE 6INCHES OR LESS	0.5020
OTHER CUTTING INSTRUMENT	0.1585
SCISSORS	0.1585
PIPE/METAL PIPE	0.3435
HAND GUN	1.0040
CLUB/BAT	0.2114
OTHER KNIFE	1.2153
ROCK/THROWN OBJECT	0.4491
SIMULATED GUN	0.2114
STICK	0.4227
UNKNOWN FIREARM	0.1585
MACE/PEPPER SPRAY	0.4756
VEHICLE	0.3435
FOLDING KNIFE	0.2114
BOTTLE	0.2114
KNIFE WITH BLADE OVER 6 INCHES IN LENGTH	0.1585
BLUNT INSTRUMENT	0.1585
SEMI-AUTOMATIC PISTOL	0.3435
SCREWDRIVER	0.1585
HAMMER	0.1585
MACHETE	0.1585

Premis Data by Weapon

400 | 40 | Submit

Premis	Count
STREET	565
SIDEWALK	420
ALLEY	45
DRIVEWAY	19
DRIVEWAY	19

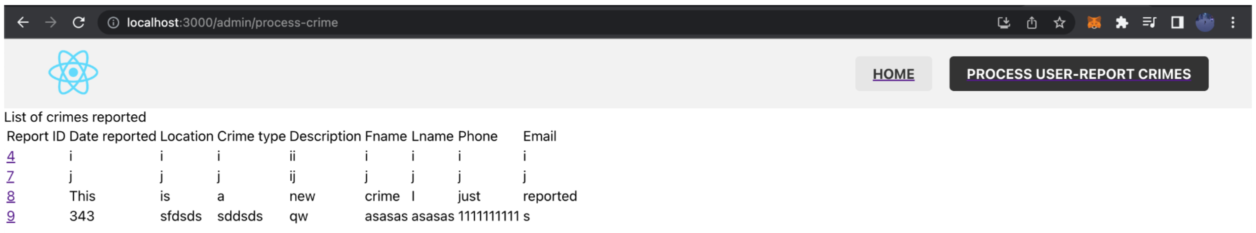
Additionally, the report crime feature adds the element of the user being able to interact and make an impact on the website/database. They can do this through the report crime tab built into the website, where they can input what they saw/heard and provide contact info such that the authorities can reach them for further questions.



Crime Report Form

Crime information Date occurred | Location | Crime type | Detailed description Contact information First name | Last name | Phone number
Email
submit

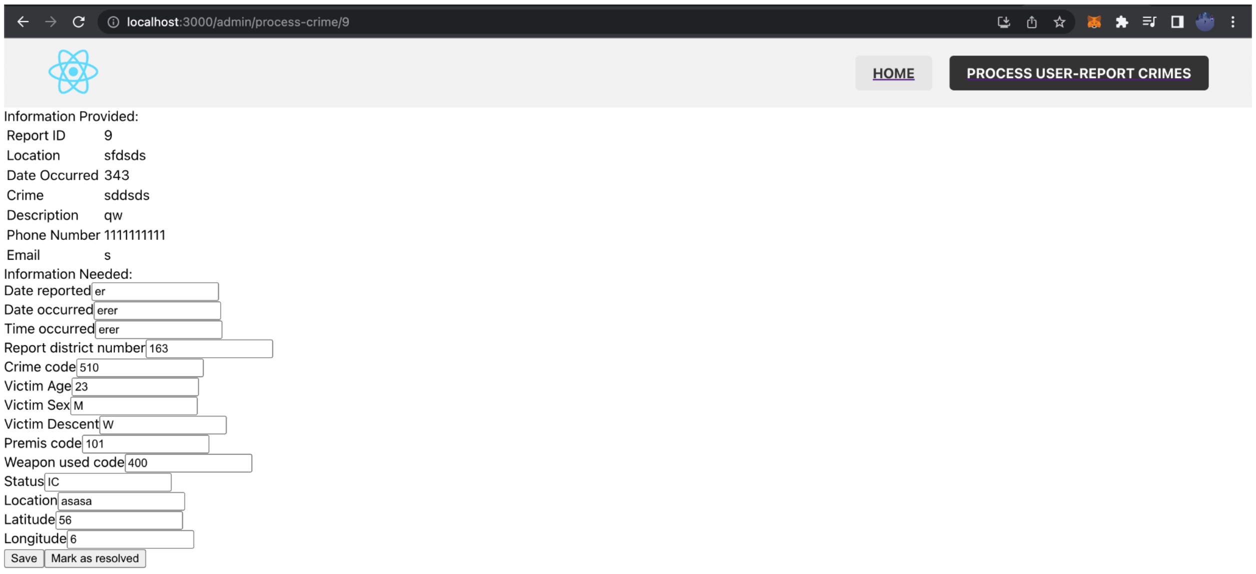
Now, the admin side of the website was a success as well because of the fact that we achieved exactly what we initially proposed, which includes the ability for the admin/police to see all the reports submitted in a row format.


localhost:3000/admin/process-crime
HOME
PROCESS USER-REPORT CRIMES

List of crimes reported

Report ID	Date reported	Location	Crime type	Description	Fname	Lname	Phone	Email
4	i	i	i	ii	i	i	i	i
7	j	j	j	ij	j	j	j	j
8	This	is	a	new	crime	I	just	reported
9	343	sfdsds	sddsds	qw	asasas	asasas	1111111111	s

They are also able to press on a specific reportID and make edits to the actual crime database based on the research they did. They also have the ability to mark a reportID as resolved, which deletes that specific reportID from the database.


localhost:3000/admin/process-crime/9
HOME
PROCESS USER-REPORT CRIMES

Information Provided:

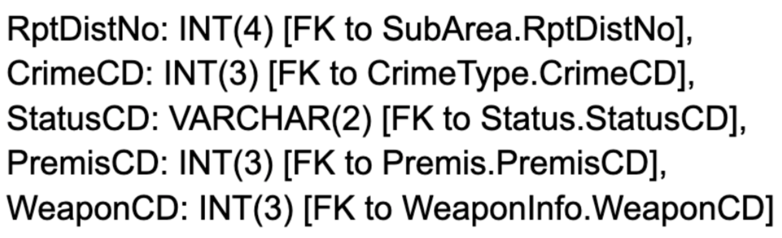
Report ID	9
Location	sfdsds
Date Occurred	343
Crime	sddsds
Description	qw
Phone Number	1111111111
Email	s

Information Needed:

Date reported	er
Date occurred	erer
Time occurred	erer
Report district number	163
Crime code	510
Victim Age	23
Victim Sex	M
Victim Descent	W
Premis code	101
Weapon used code	400
Status	IC
Location	asasa
Latitude	56
Longitude	6

Now, the only feature we were not able to implement is the cool feature, which was a pie chart for us that provided important information regarding the characteristics of the specific components of the database.

3. We used the proposed dataset from the link provided to us by the TAs with no changes other than splitting the data up into several entities for clarity's sake. The only change we made to the schema was the fact that we changed almost all the foreign keys inside the crime table from string, which it was set to by default, to int or varchar. Specifically, this includes the attributes shown below.



RptDistNo: INT(4) [FK to SubArea.RptDistNo],
CrimeCD: INT(3) [FK to CrimeType.CrimeCD],
StatusCD: VARCHAR(2) [FK to Status.StatusCD],
PremisCD: INT(3) [FK to Premis.PremisCD],
WeaponCD: INT(3) [FK to WeaponInfo.WeaponCD]

Next, we also ended up merging two tables - Area and Subarea because of redundancy in the existence of both. This resulted in the subarea entity having three attributes.

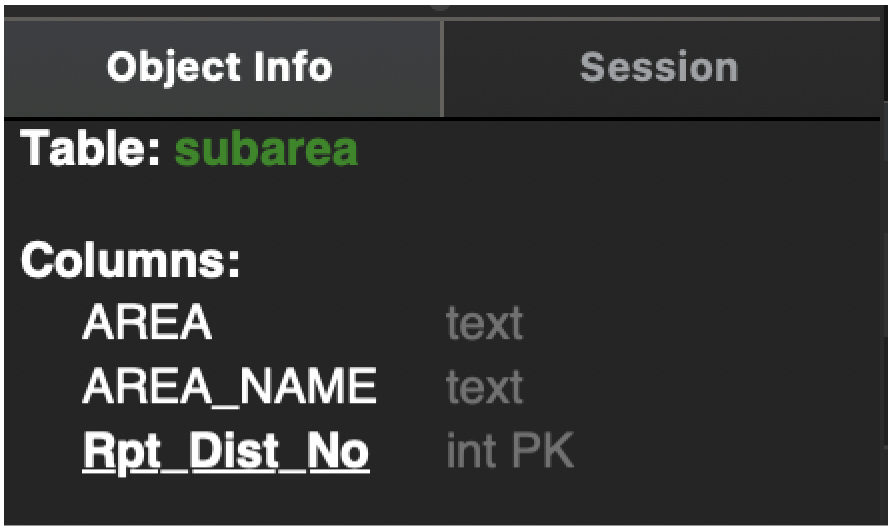


Table: subarea

Columns:

AREA	text
AREA_NAME	text
Rpt_Dist_No	int PK

Lastly, we changed up some of the data types for the self-report crime alongside the removal and addition of other attributes that might've been more/less relevant to our design.

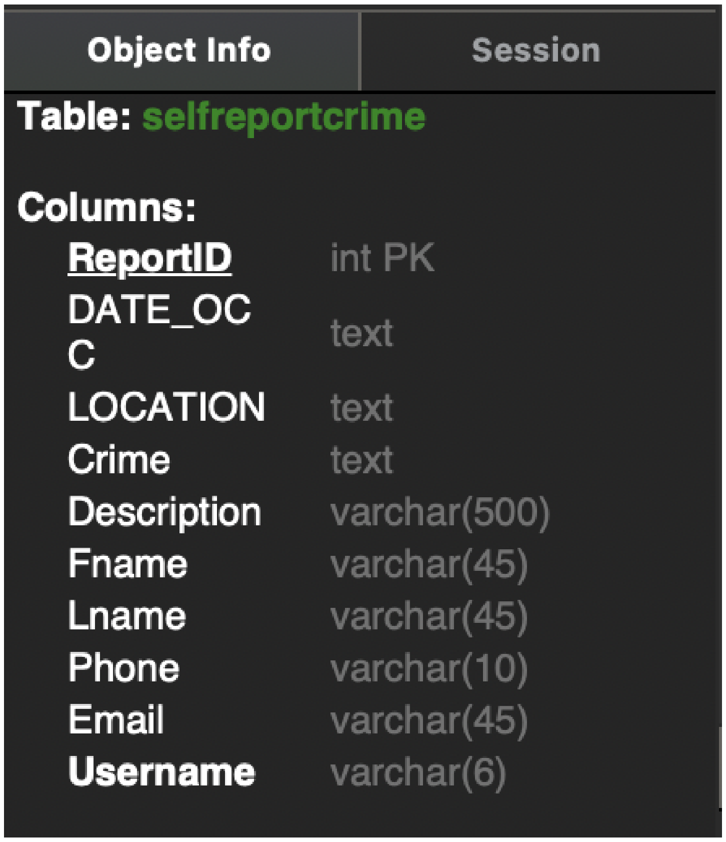
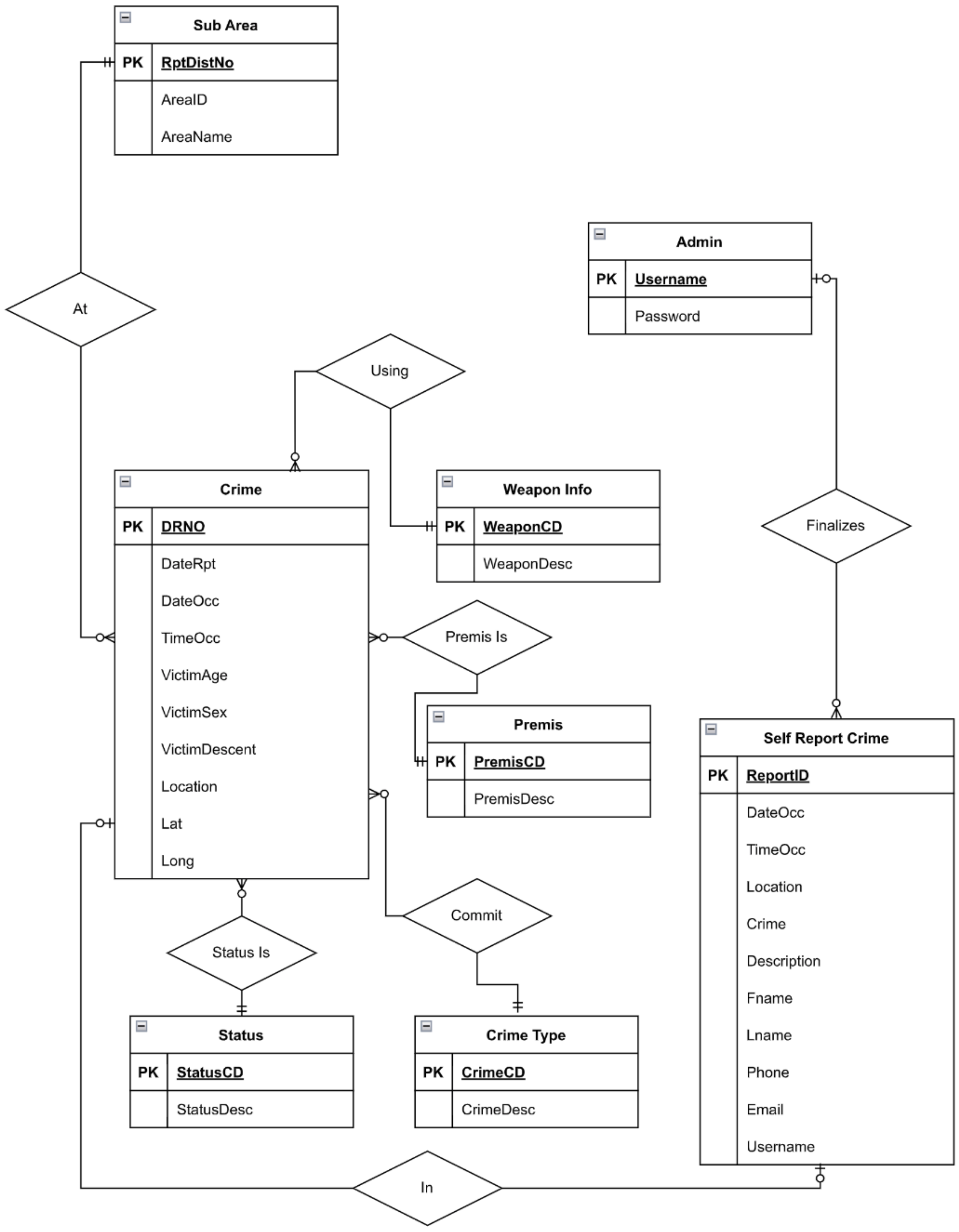


Table: selfreportcrime

Columns:

ReportID	int PK
DATE_OCC	text
LOCATION	text
Crime	text
Description	varchar(500)
Fname	varchar(45)
Lname	varchar(45)
Phone	varchar(10)
Email	varchar(45)
Username	varchar(6)

4. Shown below is our ER diagram -



Now, some of the primary changes we made were the removal of the area table and integration of it within the subarea table, which is why there are three attributes there. Another significant change we made was the fact that the self-report crime table only connects to the crime entity.

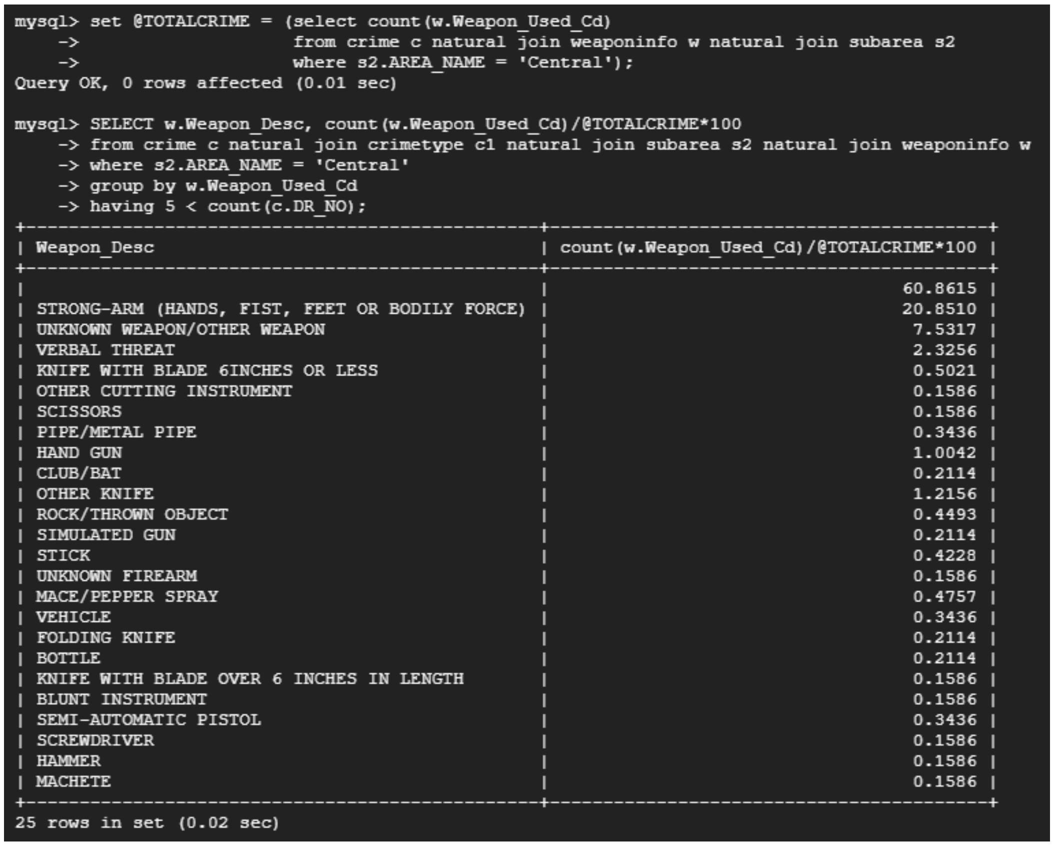
In our initial design, the self-report crime had a connection to every other entity on the database, which was unnecessary. That table specifically went through some changes as well because of the fact that some of the information stored inside of it wasn't necessarily relevant to the database as a whole. I believe that the final design is a lot more suitable because of the fact that it gets rid of a lot of redundancies and make the database more efficient as a whole due to the fact that we got rid of a lot of foreign keys that might've deemed to be redundant/hard to implement.

5. The functionalities we added include the two advanced queries -

Query 1

```
set @TOTALCRIME = (select count(w.Weapon_Used_Cd)
                   from crime c natural join weaponinfo w natural join
subarea s2
                   where s2.AREA_NAME = 'Central');

SELECT w.Weapon_Desc, count(w.Weapon_Used_Cd)/@TOTALCRIME*100
from crime c natural join crimetype c1 natural join subarea s2 natural join
weaponinfo w
where s2.AREA_NAME = 'Central'
group by w.Weapon_Used_Cd
having 5 < count(c.DR_NO);
```

Output



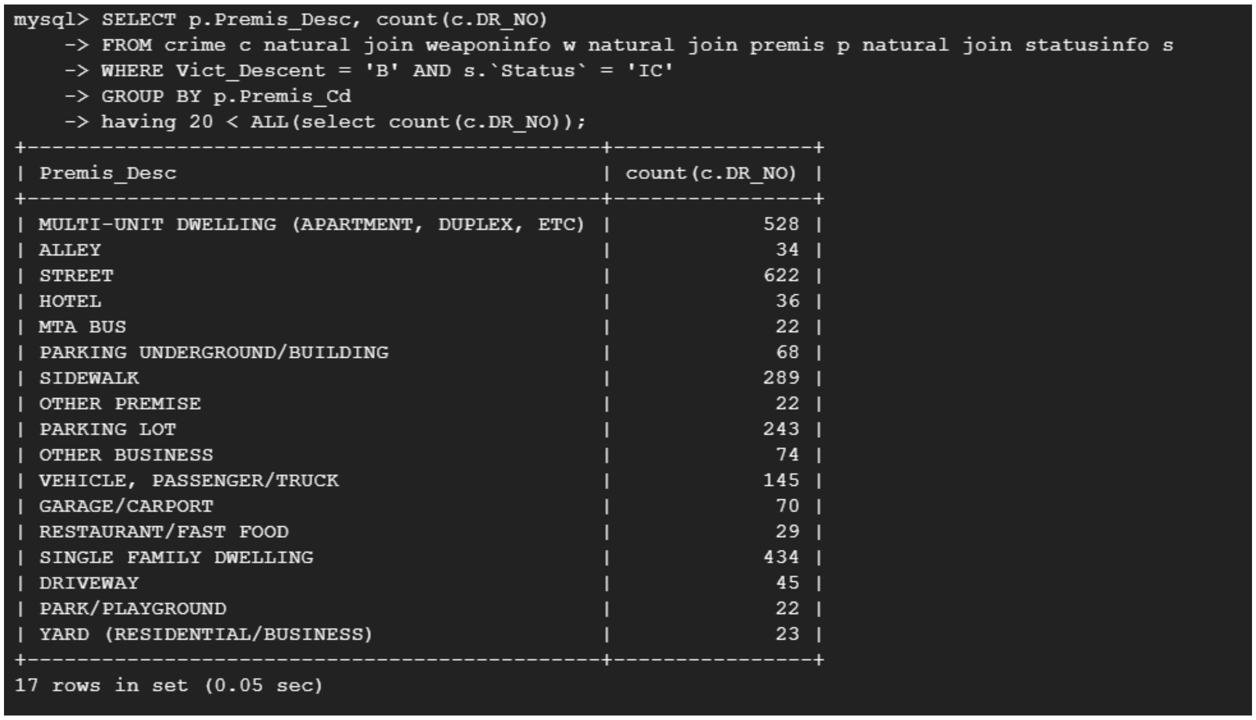
```
mysql> set @TOTALCRIME = (select count(w.Weapon_Used_Cd)
    ->                    from crime c natural join weaponinfo w natural join subarea s2
    ->                    where s2.AREA_NAME = 'Central');
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT w.Weapon_Desc, count(w.Weapon_Used_Cd)/@TOTALCRIME*100
    -> from crime c natural join crimetype c1 natural join subarea s2 natural join weaponinfo w
    -> where s2.AREA_NAME = 'Central'
    -> group by w.Weapon_Used_Cd
    -> having 5 < count(c.DR_NO);
+--------------------------------------------+------------------------------------------+
| Weapon_Desc                                | count(w.Weapon_Used_Cd)/@TOTALCRIME*100  |
+--------------------------------------------+------------------------------------------+
|                                            |                                  60.8615 |
| STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) |                              20.8510 |
| UNKNOWN WEAPON/OTHER WEAPON                |                                   7.5317 |
| VERBAL THREAT                              |                                   2.3256 |
| KNIFE WITH BLADE 6INCHES OR LESS           |                                   0.5021 |
| OTHER CUTTING INSTRUMENT                   |                                   0.1586 |
| SCISSORS                                   |                                   0.1586 |
| PIPE/METAL PIPE                            |                                   0.3436 |
| HAND GUN                                   |                                   1.0042 |
| CLUB/BAT                                   |                                   0.2114 |
| OTHER KNIFE                                |                                   1.2156 |
| ROCK/THROWN OBJECT                         |                                   0.4493 |
| SIMULATED GUN                              |                                   0.2114 |
| STICK                                      |                                   0.4228 |
| UNKNOWN FIREARM                            |                                   0.1586 |
| MACE/PEPPER SPRAY                          |                                   0.4757 |
| VEHICLE                                    |                                   0.3436 |
| FOLDING KNIFE                              |                                   0.2114 |
| BOTTLE                                     |                                   0.2114 |
| KNIFE WITH BLADE OVER 6 INCHES IN LENGTH   |                                   0.1586 |
| BLUNT INSTRUMENT                           |                                   0.1586 |
| SEMI-AUTOMATIC PISTOL                      |                                   0.3436 |
| SCREWDRIVER                                |                                   0.1586 |
| HAMMER                                     |                                   0.1586 |
| MACHETE                                    |                                   0.1586 |
+--------------------------------------------+------------------------------------------+
25 rows in set (0.02 sec)
```

Basically what this query does is that it calculates the percentage of a certain weapon used for a crime in the central area and group by is done using weapon type. We also check that we have at least 5 crimes committed using that weapon in the central area before constructing the table.

Query 2

```
SELECT p.Premis_Desc, count(c.DR_NO)
FROM crime c natural join weaponinfo w natural join premis p natural join
statusinfo s
WHERE Vict_Descent = 'B' AND s.`Status` = 'IC'
GROUP BY p.Premis_Cd
having 20 < ALL(select count(c.DR_NO))
```

Output



```
mysql> SELECT p.Premis_Desc, count(c.DR_NO)
    -> FROM crime c natural join weaponinfo w natural join premis p natural join statusinfo s
    -> WHERE Vict_Descent = 'B' AND s.`Status` = 'IC'
    -> GROUP BY p.Premis_Cd
    -> having 20 < ALL(select count(c.DR_NO));
+----------------------------------------------+----------------+
| Premis_Desc                                  | count(c.DR_NO) |
+----------------------------------------------+----------------+
| MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC) |            528 |
| ALLEY                                        |             34 |
| STREET                                       |            622 |
| HOTEL                                        |             36 |
| MTA BUS                                      |             22 |
| PARKING UNDERGROUND/BUILDING                 |             68 |
| SIDEWALK                                     |            289 |
| OTHER PREMISE                                |             22 |
| PARKING LOT                                  |            243 |
| OTHER BUSINESS                               |             74 |
| VEHICLE, PASSENGER/TRUCK                     |            145 |
| GARAGE/CARPORT                               |             70 |
| RESTAURANT/FAST FOOD                         |             29 |
| SINGLE FAMILY DWELLING                       |            434 |
| DRIVEWAY                                     |             45 |
| PARK/PLAYGROUND                              |             22 |
| YARD (RESIDENTIAL/BUSINESS)                  |             23 |
+----------------------------------------------+----------------+
17 rows in set (0.05 sec)
```

This query basically goes over all the different premises and finds the current number of cases that are still under investigation for crimes committed by people with the decent ‘White’. We also check and make sure that there are at least 20 crimes in that premis before getting the final table.

We also implemented the following stored procedure -

```
CREATE DEFINER=`root`@`%` PROCEDURE `ProcPremis`(IN weapontype int, IN
agenew int)
BEGIN
    DECLARE Premis_Cd_Var INT;
    DECLARE Premis_Desc_Var VARCHAR(100);
    DECLARE Num_Crimes_Var INT;
    DECLARE exitLoop BOOLEAN DEFAULT FALSE;
    DECLARE premisCur CURSOR FOR (SELECT Premis_Cd FROM premis);
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET exitLoop=TRUE;

    DROP TABLE IF EXISTS premis_count;
    CREATE TABLE premis_count (
        Premis_Cd INT PRIMARY KEY,
        Premis_Desc VARCHAR(100),
        Num_Crimes INT
    );

    OPEN premisCur;

    loop_0: LOOP
        FETCH premisCur INTO Premis_Cd_Var;
        IF exitLoop THEN
            LEAVE loop_0;
        END IF;

        IF (weapontype = 106 or weapontype = 107) THEN
            SELECT count(DR_NO), Premis_Desc
            INTO Num_Crimes_Var, Premis_Desc_Var
            from crime c natural join premis p
            where (c.Weapon_Used_Cd BETWEEN 100 AND 199)
                and Vict_Age <= agenew and Premis_Cd = Premis_Cd_Var
            group by c.Premis_Cd;
        ELSE
            SELECT count(DR_NO), Premis_Desc
            INTO Num_Crimes_Var, Premis_Desc_Var
            from crime c natural join premis p
            where (c.Weapon_Used_Cd = weapontype)
                and Vict_Age <= agenew and c.Premis_Cd = Premis_Cd_Var
            group by c.Premis_Cd;
        END IF;
        INSERT INTO premis_count VALUES (Premis_Cd_Var, Premis_Desc_Var, 
Num_Crimes_Var);
    END LOOP loop_0;

    CLOSE premisCur;

    SELECT * FROM premis_count ORDER BY Num_Crimes DESC;
END
```

Description - This procedure filters all crime records using the specified weapon and victim under a certain age, and then counts the number of crimes for each different premis. In the code above, a cursor is defined to iterate through all premises. Then it counts the number of crimes with the corresponding premis that also satisfies the weapon and age conditions. Finally, it inserts the information retrieved into a new table. The contents of the table are returned eventually.

We also implemented the following trigger -

```
BEGIN
    DECLARE err VARCHAR(10);
    IF LENGTH(NEW.Description) <= 1 THEN
        SET err = ''INVALID'';
        SIGNAL SQLSTATE ''45000'' SET MESSAGE_TEXT = ''Invalid 
Description'';
    END IF;

    IF LENGTH(NEW.Phone) != 10 THEN
        SET err = ''INVALID'';
        SIGNAL SQLSTATE ''45000'' SET MESSAGE_TEXT = ''Invalid Phone 
Number'';
    END IF;
END;
```

Description - This trigger is called when new crime data is entered, and it removes entries with data that essentially don’t make sense. For example, if the phone number isn’t 10 integers, it won’t put that data into the database. The same goes for the description. If a solid description is not given, the report will not be put on the database for consideration.

6. MySQL is a great open-source rdms application that can be used to provide a good backend to store and manage data. This really helped with managing our entire crime database, as well as make it efficient to execute advanced queries, our trigger and stored procedure. Google Cloud Platform provided a way to connect the MySQL database to the application, which helped us with running our application efficiently.

7. Technical challenges -

Aditya Patel - The hardest challenge was for me to learn and implement the entire backend for the application. Coming into CS 411, I had no base on how to SQL or backend/frontend. Learning Node.js and implementing it within the span of three weeks was the hardest thing for me not just in the project, but the entire class as well. Now, specifically within the actual backend, the most challenging part for me was implementing the stored procedure because of the fact that it had a loop and it returned multiple outputs. Upon researching it further and further, I was not able to find a solution on how to format the entire thing because the number of outputs returned depended directly on the number of inputs. In the end, we just ended up changing the entire stored procedure such that it returned a singular output that was sort of a combination of the outputs.

Rohan Gudipaty - The hardest challenge for me was learning and interacting with MySQL. I had experience working with SQL, but I never applied it to a database project that interacted with a website. At first, some of the queries I came up with towards the beginning of the project weren't executing, and causing the application to close completely. After some time debugging and researching, I had to format SQL queries in specific ways to work with the database (to elaborate, there were no errors in how my queries were originally formatted, but certain solutions can cause your program not to execute).

Tianyue Cao - This is not my first time making a website with an SQL back end. However, I never used GCP before and it is very challenging for me to learn how to set up and connect to a database on GCP remotely. Another challenge I encountered during this project is to work together with my teammates. When we first combined the front end and back end together, our interfaces did not match. Then we spent a lot of time modifying our code to match them. Finally, designing the database structure is also hard because relational models are complex and we have to meet so many project requirements.

8. Something we changed that wasn't in the original proposal was that we didn't include the pie chart, which gave a visual representation of some key details. We figured this would be important to add more functionality to the website, as well as add more use for the users. That way, if we were to expand on this website and share it with officials, people would be able to report real-time crimes happening, thereby updating the database and updating the information live.

9. One feature we wanted to implement was a graph or a map to better visualize the data. For a graph, we could implement filters and use time-series analysis to show the number of crimes

committed in a particular area or by a particular weapon over time, especially since we had data going back to 2010. For a map, we could again implement filters to show where most crimes were committed during a particular year, or for a particular weapon show which area it was used the most. We could also make this website more public so users can start to input data, thereby showing more real-time crime data with this application.

Diego Saavedra - One of the most significant challenges I faced during this project was working on the front-end design, specifically creating the structure within the divs and handling the CSS. While my previous experience with React Native was helpful, transitioning to React presented its unique hurdles. Ensuring the pages were responsive to different screen sizes to prevent any overlapping of elements required a deeper understanding of CSS and responsive design techniques. Another challenge I faced was organizing the code to make it more readable for my teammates. I realized that some parts of my initial code were difficult to understand, so I had to rewrite sections to improve clarity and maintainability. This process not only improved the overall quality of the code but also facilitated better collaboration with my team.

10. Contributions:

Aditya Patel - Backend/Database Implementation/Advanced Queries
Rohan Gudipaty - Worked on Backend and implemented trigger.
Tianyue Cao - Frontend/Stored Procedure
Diego Saavedra - DivStructure/CSS Files