City of Boston Crime Database Project

By Fayzulla Abdurakhimov, Garima Gupta, Charu Mangla,

Edgar Martinez, and Akshitha Morusu

BUAN 6320 Database Foundations

Professor Hossein Kamalzadeh

University of Texas at Dallas

TABLE OF CONTENT

Part 2 Business Understanding

Background on the dataset

Police departments operate similar to for-profit businesses with regards to decision-making. They must allocate resources, construct budgets, and develop business plans to ensure the financial stability of the department. The city of Boston is home to one of the largest and most robust police departments in the country. For the fiscal year 2021, the Boston Police Department (BPD) had an operating budget of $404.18 million[1] and over 2,100 police officers[2]. While the goal is not like most businesses that wish to maximize revenue, the BPD still has to meet its goals and maximize the utility of its resources. With ongoing threats from municipal governments to reduce spending, police departments must have leaner operations to ensure it meets their community demands.

One of the most important metrics in a police department is the response time to emergency calls. These response times are largely influenced by the size of the police force and how well a working shift of officers is dispersed among a district. A small police force may operate as effectively as a large one if officers are strategically placed in areas that will receive the most calls. Poorly managed police forces could result in an excessive number of required officers or officers working longer hours to cover more areas. Before data analytics became the norm, police departments used intuition and call volume to guess where the next emergency call would take place. With the introduction of data in decision-making, more police departments are able to make strong predictions of where the next crime will take place in order to have a police unit ready to respond.

Since 2015, BPD has made available its crime incident reports that document details from the incidents BPD officers respond to. The database is updated annually and includes all pertinent details of each police incident. The data is collected in an effort to make better decisions within BPD's current strategy to maximize its utility while providing transparency to the community. The database is available through the website data.boston.gov[3].

Business application

As a government-funded organization, BPD must focus on business decisions that maximize its utility and efficient use of its resources. The dataset can help uncover some key information that would allow for more strategic decision-making. Information such as which districts face the most crime, which time of the year, month, week, or day are most likely to face crime, and an understanding of the different types of incidents can allow us to identify the areas where BPD can improve. Strategic decisions, such as where to place our patrol units, how to schedule officer shifts, and what resources we need to address the type of incidents could all be answered with a robust database and querying abilities. Therefore, to test the functionality of our database, our guiding questions for this project will be as follows.

- What districts and reporting areas are facing the most crime (refer to step 5)?
- What months and days of the week face the most crime (refer to step 3)?
- Which offenses are committed the most (refer to step 5)?

---

[1] Source: https://data.aclum.org/2021/05/06/more-of-the-same-unpacking-the-2022-boston-police-budget/
[2] Source: https://www.policedatainitiative.org/participating-agencies/boston-massachusetts-police/
[3] Crime Incident Reports (August 2015 - To Date) is provided by the Boston Police Department at https://data.boston.gov/dataset/crime-incident-reports-august-2015-to-date-source-new-system

Part 3 Data Understanding

About the Dataset

There are 17 attributes (columns) recorded for each incident, as shown in the exhibit below. Summary statistics and a description of each attribute are also included in further below. The dataset measures about 60 MB and contains over 300,000 records. We have run the following query to check the data types of the dataset:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'project'
AND TABLE_NAME = 'crime';
```

COLUMN_NAME	DATA_TYPE	COLUMN_NAME	DATA_TYPE
DAY_OF_WEEK	text	OFFENSE_CODE	text
DISTRICT	text	OFFENSE_CODE_GROUP	text
HOUR	int	OFFENSE_DESCRIPTION	text
INCIDENT_NUMBER	text	REPORTING_AREA	int
Lat	double	SHOOTING	text
Location	text	STREET	text
Long	double	UCR_PART	text
MONTH	int	YEAR	int
OCCURRED_ON_DATE	text		

Data Quality

Like most big data, this dataset suffered from some poor data entries that needed some cleaning. The main issue was that a significant portion of the rows did not contain any location information. While this could be related to incidents that do not take place in a physical location, or a characteristic that only domain knowledge could verify, we believe for the purposes of our analysis it is best to delete these rows. Fortunately, it was only a small portion of the dataset (6%), and we still had plenty to work with. Aside from this, we found no other major issues simply by using the sort and filter functions in excel to preview the data.

Relationships in the data

The first relationship in the dataset is the one between offense code, offense code group, and offense description. To make the name clearer, we will change “offense code group” to “offense group”. Offense group is a categorical description of the offense code in a given incident. Essentially, offense group is a broader grouping of offense codes. For example, several different offense codes appear under the same larceny offense group to reflect the

different larceny types. The offense description describes the specific offense code, and not the offense group. Therefore, there is a 1-to-1 relationship with offense code and offense description.

The second relationship in the dataset is between latitude, longitude, and location. Latitude and longitude show the specific point where the incident occurred. Location combines these two to provide the coordinates of the location. Because we see this as a duplication of data and entries, we will only keep location. While not directly related, street provides a more commonly understood way to locate an incident. The location column serves a more precise description because it points to a specific location within the street, whereas street could refer to any point along the street that spans across several districts or reporting areas.

The last relationship is between date occurred on, year, month, day of week, and hour. The date occurred on records the year, day, month, hour, and minutes of the incident. Therefore, year, month and hour are duplicate or redundant data because the date occurred on column already records these. Day of week records the day of the week, which is not recorded in the date occurred on field. Therefore, we will keep this one to preserve the data. We will go into further detail with these three relationships in part 4 when we normalize our schema.

Summary of the data

Attached below is Figure 1, which shows each attribute with a brief description and summary statistics to uncover information on that attribute. The code to obtain the summary statistics is located in the appendix of this report.

Figure 1: Summary information

Column name	Description	Summary Statistics (ex = exhibit)
INCIDENT_NUMBER *(categorical data)*	Internal Boston police department report number	**Total count:** 306,412 (ex 1) **Count(Unique):** 271,754 (ex 1) **Percentage of Missing Values:** 0% (ex 23)
OFFENSE_CODE *(categorical data)*	Numerical code of offense description	**Count(Unique):** 263 (ex 1) **Mode:** 03006 (ex 25) **Percentage of Missing Values:** 0% (ex 23)
OFFENSE_CODE_GROUP *(categorical data)*	Internal categorization of offense	**Count(Unique):** 66 (ex 1) **Mode:** Medical Assistance (ex 19) **Percentage of Missing Values:** 0% (ex 23)
OFFENSE_DESCRIPTION *(categorical data)*	Primary descriptor of incident	**Count(Unique):** 244 (ex 1) **Mode:** SICK/INJURED/MEDICAL - PERSON (ex 20) **Percentage of Missing Values:** 0% (ex 23)
DISTRICT *(categorical data)*	What district the crime was reported in	**Count(Unique):** 13 (ex 1) **Mode:** B2 (ex 21) **Percentage of Missing Values:** 0.5% (ex 23)
REPORTING_AREA *(categorical data)*	Reporting area number where the crime was reported from	**Count(Unique):** 879 (ex 1) **Mode:** 111 (ex 22) **Percentage of Missing Values:** 6.3% (ex 23)
SHOOTING *(categorical data)*	Binary variable of whether shooting took place	**Count of Y:** 1055 (ex 1) **Count of Null (presumed no):** 326,765 (ex 18) **Percentage of Missing Values:** 99.67% (ex 23)
OCCURRED_ON_DATE *(categorical data)*	Earliest date and time the incident could have taken place in (YYYY/MM/DD HH:MM:SS) format	**Earliest date (min):** 6/15/15 0:00 (ex 2) **Latest date (max):** 10/3/18 20:49 (ex 2) **Percentage of Missing Values:** 0% (ex 23)
YEAR	Year when the incident took place	**Percentage of 2018**: 74,356 = 22.68%

(categorical data)		**Percentage of 2017:** 100,938 = 39.79% **Percentage of 2016:** 99,134 = 30.24% **Percentage of 2015:** 53,392 = 16.28% **Range:** 2015-2018 **Percentage of Missing Values:** 0% (ex 23)
MONTH *(categorical data)*	Month when the incident took place (number)	**Mode:** August (08) - 35,137 times (ex 4) **Count(Unique):** 12 months (ex 1) **Range**: January (01) - December (12) (ex 5) **Percentage of Missing Values:** 0% (ex 23)
DAY_OF_WEEK *(categorical data)*	Day of the week when the incident took place	**Mode:** Friday (5) (ex 6) **Count(Unique):** 7 days (ex 7) **Range:** Monday (1) - Sunday (7) (ex 9) **Percentage of Missing Values:** 0% (ex 23)
HOUR	Hour when the incident took place (24 hour format)	**Mode:** 13 (ex 9) **Count(Unique):** 24 (ex 8) **Range:** 0-23 **Percentage of Missing Values:** 4,73% (ex 23)
UCR_PART *(categorical data)*	Universal Crime Reporting Part number	**Percentage of Part One:** 63,231 = 19,28% **Percentage of Part Two:** 100,283 = 30,59% **Percentage of Part Three:** 162,928 Other: 1,285 = 49,7% **Percentage of Missing Values:** 93 = 0% (ex 23)
STREET *(categorical data)*	Street name where the incident took place	**Mode:** Washington street (ex 11) **Count(Unique):** 4,685 (ex 1) **Percentage of Missing Values:** 3.34% (ex 23)
Lat *(categorical data)*	Latitude where the incident took place	**Count(Unique):** 18,240 (ex 12) **Mode:** 42.34862382 (ex 13) **Percentage of Missing Values:** 6.29% (ex 23)
Long	Longitude where the incident took place	**Count(Unique):** 18,240 (ex 14)

		Mode: -71.08277637 (ex 15) **Percentage of Missing Values:** 6.29% (ex 23)
Location	Latitude and longitude where the incident took place	**Count(Unique):** 18,255 (ex 16) **Mode:** (42.34862382, -71.08277637) (ex 17) **Percentage of Missing Values:** 0% (ex 23)

Commentary on the summary statistics

Interesting point is that most accidents happen in Washington street around 1pm. Moreover, the most occurring days and months for crimes were Friday and August in turn. Additionally, only in 1055 times of accidents shooting cases occurred which is equal to 0.3% of all cases. This helps us answer some of our guiding questions in part 2 above.

All the queries are available in the appendix.

Part 4 Design a Database

Database Goals

The primary goals of designing this database are to eliminate redundancies in the original table and allow for querying that will answer the guiding questions in part 2.

Schema Design

From the base table of 17 attributes, we believe the first step in designing a more efficient schema is to create 4 entities. The ER diagram below (figure 2) shows the 4 entities, the attributes included in each, and the relationship between each entity.

The first entity should record all the data about offenses. Because there are only 263 different offenses, it is redundant to repeat this data entry for 300,000 rows. This offense table should have the offense code, offense code group, and offense description. The primary key should be the offense code as it is unique to every offense description.

Next, a location entity should house all of the data related to location. Because there are only 18,255 locations for 300,000 rows, it is also redundant to have so many duplicate entries. This incident table should have district, reporting area, longitude, latitude, and location variables. The primary key should be the location variable as it is unique for every different location.

Next, a data entity should house all of the data related to the date of the crime. This table should have the occured on date, year, month, day of week, and hour attributes. The primary key should be the occurred on date, as it is the most unique of the variables.

The last entity in our proposed schema design is an incident table that combines all of the previous entities together. This table would also include any other variables not included in the other entities. The incident number and shooting attributes would be included in the table, as well as the primary keys from the other entities (foreign keys in this table). In total, we include incident number, shooting, offense code, location, and occured on date in the entity.

Constraints

Primary keys have been assigned to each entity. The Location table will use location as its primary key, the offense table will use the offense code as its primary key, and date will use occurred on date as its primary key. The incident table will use the incident number as its primary key, and will use the other entity's primary keys as foreign keys to connect the tables together.

Relationships

Relationships between the entities are shown in the ER diagram (figure 2) and explained below.

Incident to offense: Each incident must at a minimum have 1 offense, and it can have many offenses

Offense to incident: Each offense must at a minimum have 1 incident (to appear in the table), and it can appear under many incidents.

Incident to location: Each incident must have one location associated with it, and only one location. Incidents may not span across locations.

Location to incident: Each location must appear in at least 1 incident, and it can appear under many incidents.

Incident to date: Each incident must have one date associated with it, and only one date. Incidents that have more than one date appear as a brand new incident.

Date to incident: Each date must at a minimum appear in 1 incident, and it can appear under many incidents.

Schema Normalization

In order to effectively normalize our schema, we will use our proposed design and normalize until we reach Boyce Codd normal form (BCNF). The first step in the normalization process is to bring the schema into the first normal form. We found after our initial design that the primary keys were duplicated across some rows, therefore violating the unique requirement of first normal form. To correct this, we replace the primary keys with generated unique IDs. These are incident ID to replace incident number, offense ID to replace offense code, location ID to replace location, and date ID to replace occurred on date.

Finding Functional Dependencies

A = {Incident ID, incident number, shooting, Offense ID, offense code, offense group,offense description, Location ID, location, longitude, latitude, district, reporting area, street, Date ID, occurred on date, year, month, day of week, hour}

F = {Location ID} -> {location, longitude, latitude, district, reporting area, street}
Drop longitude and latitude for data redundancy
Location ID, location, district, reporting area, street are in the same table,
Location ID is a key.

{Offense ID} -> {offense code, offense group, offense description}
Offense ID, offense code, offense group, offense description are in the same table,
Offense ID is a key

{Date ID} -> {occured on date, year, month, day of week, hour}
Drop year, month, hour for data redundancy
Date ID, occurred on date, day of week are in the same table, Date ID is a key

{incident ID} -> {offense ID, incident number, shooting, offense ID, location ID, date ID}
Incident ID, incident number, shooting are in the same table, Incident ID is a key
Incident ID, offense ID, location ID, date ID are in the same table, Incident ID is a key
while offense, location, and date are foreign keys

Boyce Codd Normal Form

To verify we have reached BCNF, we will list out all of the functional dependencies above. Because all of the left hand sides are candidate keys, we have reached BCNF.

Incident table (Incident ID, incident number, shooting, offense ID, location ID, date ID)

Offense table (offense ID, offense code, offense group, offense description)

Location table (location ID, location, district, reporting area, street)

Date table (date ID, occurred on date, day of week)

See our finalized schema in the updated ER diagram in Figure 3 below.

Data Import

After finalizing our schema following our normalization, we imported the data into MySQL using the code as shown in Exhibit 24. Screenshots of the tables are also included.

With our import, we ran into a few issues. The first was that the import time for the original table took around 4 hours. Fortunately, we did this early on in the project timeline so we just allowed the import to take place overnight. From this original table, we created our database, using the finalized schema above.

During our normalized table creation, we limited our dataset to 10,000 rows because we reached the processing limit of MySQL Workbench and it did not allow us to create our tables with the necessary attributes. Therefore, in our finalized schema (the 4 tables) we have only 10,000 rows of data from the original dataset. Fortunately, this limitation allowed the tables to work property. This should show the schema as a proof of concept rather than a functional database to obtain answers from queries. If we had additional processing power and a way to unlock MySQL Workbench's processing limit, we could apply the full data from the dataset to the finalized schema.

We also ran into some errors that conflicted with MySQL's default method of generating foreign keys. As a result, we found a workaround to add foreign keys in the "create table" step, instead of the "alter table" step.

Figure 2
ER Diagram
(pre-normalization)

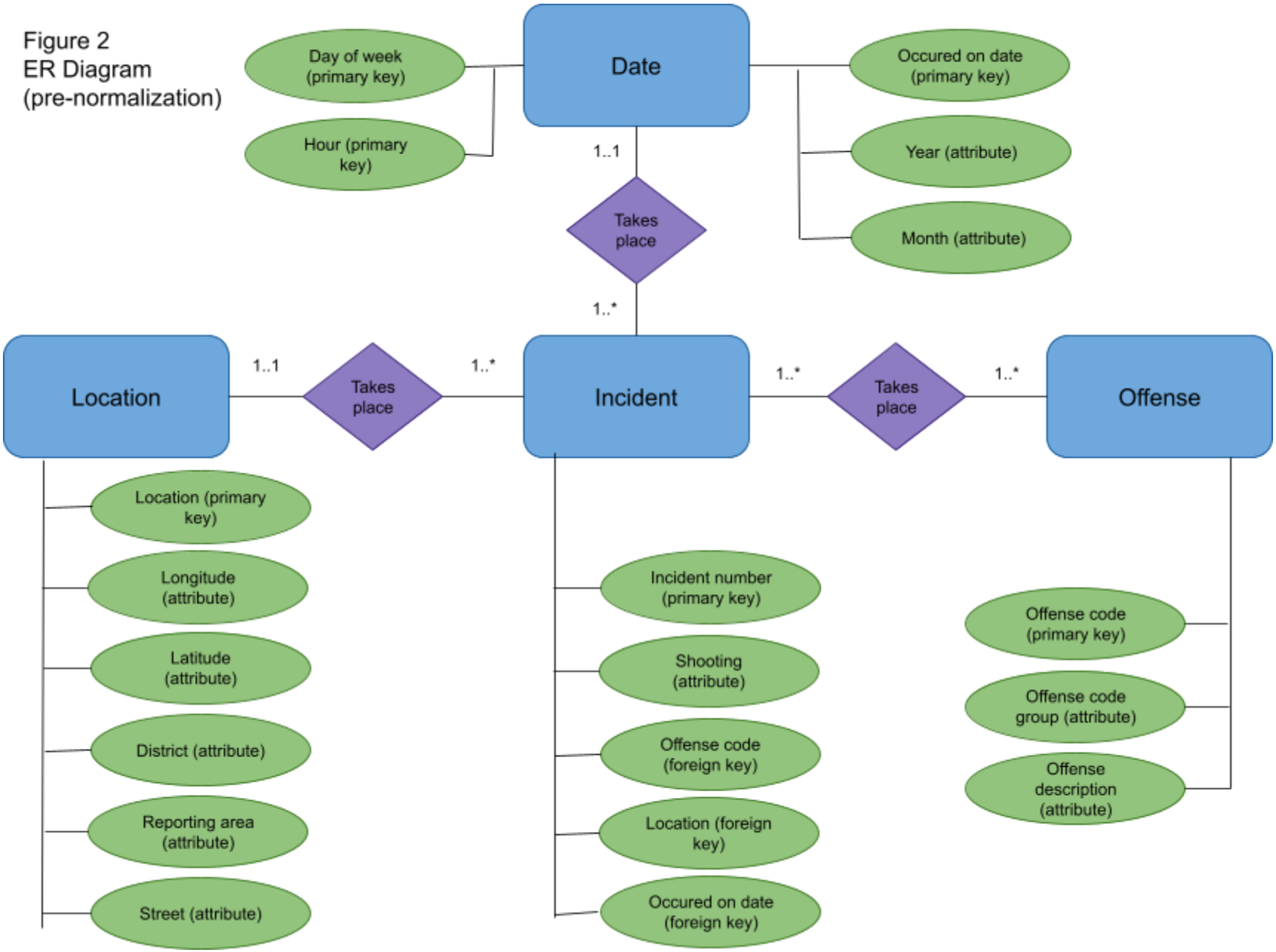
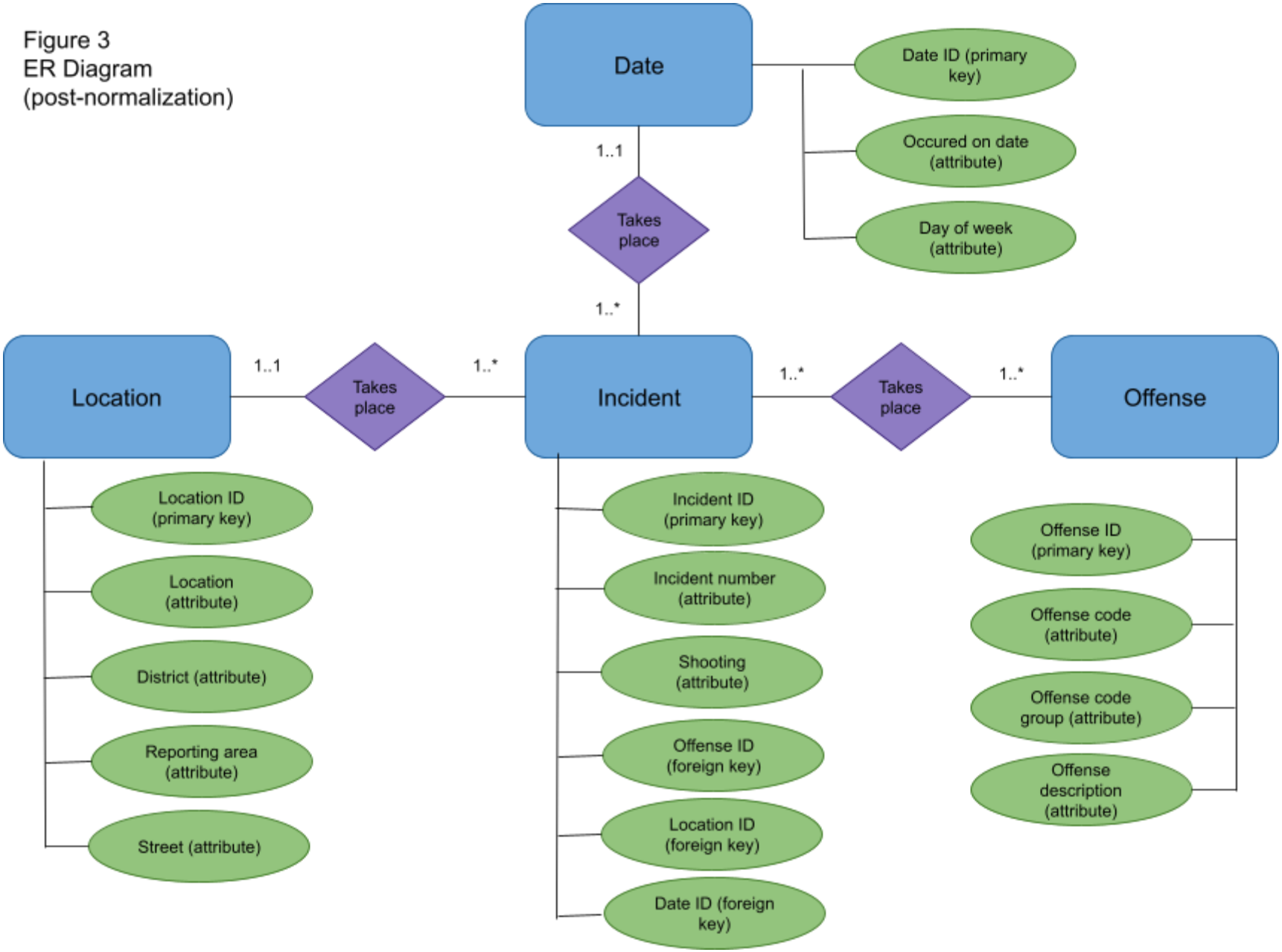


Figure 3
ER Diagram
(post-normalization)



Part 5 Data Cleaning and Database Testing

Checking the data

Our findings in step 3 show the summary statistics of our table, following the import process in step 4.

As far as numeric columns, here are our findings. We discovered that we had few numeric columns in our dataset. It was primarily the unique ID rows, offense code, and reporting area that had numeric data, and these did not contain missing values or outliers.

As for character columns, the shooting column had 99% null values (see step 3). However, we interpreted these null values as "no" (no shooting occurred) and converted blank values to "No". See the updated table with the code attached.

```
UPDATE incident2
SET shooting= "N"
WHERE shooting IS NULL or shooting = "";
SET sql_safe_updates=0;
```

INCIDENT_NUMBER	SHOOTING	OFFENCE_ID	Date_ID	LOCATION_ID	INCIDENT_ID
I182080058	N	28	36	1	76
I182080058	N	27	36	1	77
I182080058	N	26	36	1	78
I182080058	N	25	36	1	79
I182080058	N	24	36	1	80

Joining data

We tested our database by joining the entities with each other. Fortunately, we did not face any problems. To do so we queried district, reporting area and street from the location table and joined it with shooting, occured on date, and offense description from incident, date, offense tables respectively. See the updated table with the code attached.

```
Select l.DISTRICT, l.REPORTING_AREA, l.STREET, i.SHOOTING, d.OCCURRED_ON_DATE,
o.OFFENSE_DESCRIPTION
from incident2 i
left join location2 l
on l.LOCATION_ID = i.LOCATION_ID
left join Date1 d
on i.Date_id = d.Date_id
left join offence1 o
on i.OFFENCE_ID = o.OFFENCE_ID;
```

DISTRICT	REPORTING_AREA	STREET	SHOOTING	OCCURRED_ON_DATE	OFFENSE_DESCRIPTION
E18	957	CLIFFMONT ST	N	2018-10-03 08:00:00	LIQUOR - DRINKING IN PUBLIC
E18	957	CLIFFMONT ST	N	2018-10-03 11:34:00	PROSTITUTION - ASSISTING OR PROMOTING
E18	957	CLIFFMONT ST	N	2018-10-03 08:00:00	CRIMINAL HARASSMENT
E18	957	CLIFFMONT ST	N	2018-10-03 11:34:00	FRAUD - WIRE
E18	957	CLIFFMONT ST	N	2018-10-03 11:34:00	CONSPIRACY EXCEPT DRUG LAW

Answers to Guiding Questions

For sake of interest we answered our questions that we asked previously in part 2. The first question was what districts and reporting areas are facing the most crime? We find out that Washington street is the most popular area for incidents and B2 is also notorious for the same reason. Refer to the query and output below.

```
Select l.DISTRICT, l.REPORTING_AREA, l.STREET, count(STREET) freq
from location2 l
group by l.STREET
order by freq desc
```

DISTRICT	REPORTING_AREA	STREET	freq
B2	287	WASHINGTON ST	318

For the next question we found that the most common offense was Medical Assistance. The query and output can be seen below:

```
select OFFENSE_CODE_GROUP, count(OFFENSE_CODE_GROUP) c
from crime w
group by OFFENSE_CODE
order by  c desc;
```

OFFENSE_CODE_GROUP	c
Medical Assistance	19360
Investigate Person	19172
Motor Vehicle Accident Response	16730
Vandalism	15540
Simple Assault	15191
Verbal Disputes	13478
Towed	11632

Overall, we can say that our database is working properly. By using this dataset, the Boston Police Department can better optimize its decision making to target police incidents more efficiently.

Appendix

Exhibit 1

```
#finding unique values of columns
SELECT "DAY_OF_WEEK", COUNT(distinct DAY_OF_WEEK)
from crime;
SELECT "OFFENSE_CODE_GROUP", COUNT(distinct OFFENSE_CODE_GROUP)
from crime;
SELECT "OFFENSE_CODE", COUNT(distinct OFFENSE_CODE)
from crime;
SELECT "DISTRICT", COUNT(distinct DISTRICT)
from crime;
SELECT "REPORTING_AREA", COUNT(distinct REPORTING_AREA)
from crime;
SELECT "OFFENSE_DESCRIPTION", COUNT(distinct OFFENSE_DESCRIPTION)
from crime;
SELECT "SHOOTING", COUNT(distinct SHOOTING)
from crime;
SELECT "MONTH", COUNT(distinct MONTH)
from crime;
SELECT "STREET", COUNT(distinct STREET)
from crime;
```

Exhibit 2

```
#finding range of date column
select min(date1), max(date1) from (
SELECT STR_TO_DATE(date,'%Y-%m-%d') as date1 from(
SELECT left(OCCURRED_ON_DATE,10) date
from crime) t1) t2;
```

Exhibit 3

```
#finding unique values of columns
select YEAR, count(YEAR) c
from crime
group by YEAR;
```

Exhibit 4

```
#finding the month when most incidents occur
select months, count(months) freq from (
SELECT EXTRACT(MONTH FROM date1) months from(
SELECT STR_TO_DATE(date,'%Y-%m-%d') as date1 from(
SELECT left(OCCURRED_ON_DATE,10) date
from crime) t1) t2) t3
group by months
order by freq desc;
```

Exhibit 5

```
#finding the range of months column
select min(months), max(months) from (
select months, count(months) freq from (
SELECT EXTRACT(MONTH FROM date1) months from(
```

```
SELECT STR_TO_DATE(date,'%Y-%m-%d') as date1 from(
SELECT left(OCCURRED_ON_DATE,10) date
from crime) t1) t2) t3
group by months
order by freq desc) t4;
```

Exhibit 6

```
#finding the day of week when most accidents occur
select DAY_OF_WEEK, count(DAY_OF_WEEK) c
from crime
group by day_of_week
order by c desc;
```

Exhibit 7

```
#finding unique values of columns
select count(distinct DAY_OF_WEEK)
from crime;
```

Exhibit 8

```
#finding unique values of columns
select count(distinct HOUR)
from crime;
```

Exhibit 9

```
#finding hour of week when most accidents occur
select HOUR, count(HOUR) c
from crime
group by HOUR
order by  HOUR;
```

Exhibit 10

```
#finding UCR_PART when most accidents occur
select UCR_PART, count(UCR_PART) c
from crime
group by UCR_PART;
```

Exhibit 11

```
#finding STREET where most accidents occur
select STREET, count(STREET) c
from crime
group by STREET
order by  c desc;
```

Exhibit 12

```
#finding unique values of latitude
select count(distinct Lat)
from crime;
```

Exhibit 13

```
#finding Latitude where most accidents occur
```

```
select Lat, count(Lat) c
from crime
group by Lat
order by  c desc;
```

Exhibit 14

```
#finding unique values of longitude
select count(distinct c.Long)
from crime c;
```

Exhibit 15

```
#finding Longitude where most accidents occur
select w.Long, count(w.Long) c
from crime w
group by w.Long
order by  c desc;
```

Exhibit 16

```
#finding unique values of columns
select count(distinct Location)
from crime c;
```

Exhibit 17

```
#finding Location where  most incidents occur
select Location, count(Location) c
from crime w
group by Location
order by  c desc;
```

Exhibit 18

```
#finding Shooting where most accidents occur
select SHOOTING, count(SHOOTING) c
from crime w
group by SHOOTING
order by  c desc;
```

Exhibit 19

```
#finding Offense group where most incidents occur
select OFFENSE_CODE_GROUP, count(OFFENSE_CODE_GROUP) c
from crime w
group by OFFENSE_CODE
order by  c desc;
```

Exhibit 20

```
#finding Offence_Descripttion where most incidents occur
select OFFENSE_DESCRIPTION, count(OFFENSE_DESCRIPTION) c
from crime w
group by OFFENSE_DESCRIPTION
order by  c desc;
```

Exhibit 21

#finding *District* where most incidents occur

```
select DISTRICT, count(DISTRICT) c
from crime w
group by DISTRICT
order by  c desc;
```

Exhibit 22

#finding *Reporting area* where most incidents occur

```
select REPORTING_AREA, count(REPORTING_AREA) c
from crime w
group by REPORTING_AREA
order by  c desc;
```

Exhibit 23

#finding the missing values of our columns:

```
select 'INCIDENT_NUMBER' as INCIDENT_NUMBER,
  1.0*count(case when INCIDENT_NUMBER is null or INCIDENT_NUMBER = "" then 1
end)/count(*) as PctMissing,
  count(distinct INCIDENT_NUMBER) as UniqueCount
from crime
union all
select 'DAY_OF_WEEK' as DAY_OF_WEEK,
  1.0*count(case when DAY_OF_WEEK is null or DAY_OF_WEEK = "" then 1 end)/count(*) as
PctMissing,
  count(distinct DAY_OF_WEEK) as UniqueCount
from crime
union all
select 'DISTRICT' as DISTRICT,
  1.0*count(case when DISTRICT is null or DISTRICT = "" then 1 end)/count(*) as PctMissing,
  count(distinct DISTRICT) as UniqueCount
from crime
union all
select 'Lat' as HOUR,
  1.0*count(case when Lat is null or Lat = "" then 1 end)/count(*) as PctMissing,
  count(distinct Lat) as UniqueCount
from crime
union all
select 'Location' as Location,
  1.0*count(case when Location is null or Location = "" then 1 end)/count(*) as PctMissing,
  count(distinct Location) as UniqueCount
from crime
union all
select 'Long' as Long1,
  1.0*count(case when c.Long is null or c.Long = "" then 1 end)/count(*) as PctMissing,
  count(distinct c.Long) as UniqueCount
from crime c
union all
select 'MONTH' as MONTH,
  1.0*count(case when MONTH is null or MONTH = "" then 1 end)/count(*) as PctMissing,
  count(distinct MONTH) as UniqueCount
from crime
```

```
union all
select 'OCCURRED_ON_DATE' as OCCURRED_ON_DATE,
  1.0*count(case when OCCURRED_ON_DATE is null or OCCURRED_ON_DATE = "" then 1
end)/count(*) as PctMissing,
  count(distinct OCCURRED_ON_DATE) as UniqueCount
from crime
union all
select 'OFFENSE_CODE' as OFFENSE_CODE,
  1.0*count(case when OFFENSE_CODE is null or OFFENSE_CODE = "" then 1 end)/count(*)
as PctMissing,
  count(distinct OFFENSE_CODE) as UniqueCount
from crime
union all
select 'OFFENSE_CODE_GROUP' as OFFENSE_CODE_GROUP,
  1.0*count(case when OFFENSE_CODE_GROUP is null or OFFENSE_CODE_GROUP = ""
then 1 end)/count(*) as PctMissing,
  count(distinct OFFENSE_CODE_GROUP) as UniqueCount
from crime
union all
select 'OFFENSE_DESCRIPTION' as OFFENSE_DESCRIPTION,
  1.0*count(case when OFFENSE_DESCRIPTION is null or OFFENSE_DESCRIPTION = ""
then 1 end)/count(*) as PctMissing,
  count(distinct OFFENSE_DESCRIPTION) as UniqueCount
from crime
union all
select 'REPORTING_AREA' as REPORTING_AREA,
  1.0*count(case when REPORTING_AREA is null or REPORTING_AREA = "" then 1
end)/count(*) as PctMissing,
  count(distinct REPORTING_AREA) as UniqueCount
from crime
union all
select 'SHOOTING' as SHOOTING,
  1.0*count(case when SHOOTING is null or SHOOTING = "" then 1 end)/count(*) as
PctMissing,
  count(distinct SHOOTING) as UniqueCount
from crime
union all
select 'STREET' as STREET,
  1.0*count(case when STREET is null or STREET = "" then 1 end)/count(*) as PctMissing,
  count(distinct STREET) as UniqueCount
from crime
union all
select 'UCR_PART' as UCR_PART,
  1.0*count(case when UCR_PART is null or UCR_PART = "" then 1 end)/count(*) as
PctMissing,
  count(distinct UCR_PART) as UniqueCount
from crime
union all
select 'YEAR' as YEAR,
  1.0*count(case when YEAR is null or YEAR = "" then 1 end)/count(*) as PctMissing,
  count(distinct YEAR) as UniqueCount
```

```
from crime
union all
select 'HOUR' as HOUR,
  1.0*count(case when HOUR is null or HOUR = "" then 1 end)/count(*) as PctMissing,
  count(distinct HOUR) as UniqueCount
from crime;
```

Exhibit 24
Below is our import code for the final schema and screenshots of the results screens

```
#create table LOCATION2
CREATE TABLE LOCATION2
(Location varchar(45), DISTRICT varchar(45), REPORTING_AREA varchar(45), STREET
varchar(45));

INSERT INTO LOCATION2
SELECT DISTINCT Location, DISTRICT, REPORTING_AREA, STREET
FROM crime
limit 10000;
alter table LOCATION2 add LOCATION_ID INT NOT NULL auto_increment primary key;
```

Location	DISTRICT	REPORTING_AREA	STREET	LOCATION_ID
(42.27668221, -71.12065268)	E18	957	CLIFFMONT ST	1
(42.26260773, -71.12118637)	E18	495	ARLINGTON ST	2
(42.27810693, -71.11636523)	E18	499	AMERICAN LEGION HWY	3
(42.35211146, -71.13531147)	D14	795	ALLSTON ST	4
(42.33367922, -71.09187755)	B2	289	SCHROEDER PLZ	5
(42.30812619, -71.07692974)	B2	329	DEVON ST	6
(42.33044113, -71.08373481)	B2	287	WASHINGTON ST	7

```
#create table Offence1
CREATE TABLE Offence1
(OFFENSE_CODE int, OFFENSE_CODE_GROUP varchar(100), OFFENSE_DESCRIPTION
varchar(100));

INSERT INTO Offence1
SELECT DISTINCT OFFENSE_CODE, OFFENSE_CODE_GROUP,
OFFENSE_DESCRIPTION
FROM crime
limit 10000;

alter table Offence1 add OFFENCE_ID INT NOT NULL auto_increment primary key;
```

OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIPTION	OFFENCE_ID
727	Auto Theft	AUTO THEFT - LEASED/RENTED VEHICLE	1
2403	Disorderly Conduct	DISTURBING THE PEACE	2
542	Commercial Burglary	BURGLARY - COMMERICAL - NO FORCE	3
3201	Property Lost	PROPERTY - LOST	4
1831	Drug Violation	DRUGS - SICK ASSIST - OTHER NARCOTIC	5
2647	Other	THREATS TO DO BODILY HARM	6
1825	Drug Violation	DRUGS - POSSESSION OF DRUG PARAPHANALIA	7

```
#create table Date1
CREATE TABLE Date1
(OCCURRED_ON_DATE varchar(45), DAY_OF_WEEK varchar(45));

INSERT INTO Date1
SELECT DISTINCT OCCURRED_ON_DATE, DAY_OF_WEEK
FROM crime
limit 10000;

alter table Date1 add Date_ID INT NOT NULL auto_increment primary key;
```

OCCURRED_ON_DATE	DAY_OF_WEEK	Date_ID
2018-10-01 23:30:00	Monday	1
2018-10-03 20:13:00	Wednesday	2
2018-10-03 11:42:00	Wednesday	3
2018-09-29 16:10:00	Saturday	4
2018-08-30 20:00:00	Thursday	5
2018-10-02 00:18:00	Tuesday	6
2018-09-30 18:08:00	Sunday	7

```
#create table Incident2
CREATE TABLE Incident2
(INCIDENT_NUMBER varchar(45), SHOOTING varchar(45), OFFENCE_ID int, Date_ID int,
LOCATION_ID int,
constraint of_fk FOREIGN KEY (OFFENCE_ID)
REFERENCES Offence1(OFFENCE_ID),
constraint d_fk FOREIGN KEY (Date_ID)
REFERENCES Date1(Date_ID),
constraint l_fk FOREIGN KEY (LOCATION_ID)
REFERENCES Location2(LOCATION_ID));

INSERT INTO Incident2
SELECT DISTINCT INCIDENT_NUMBER, SHOOTING, OFFENCE_ID, Date_ID,
LOCATION_ID
FROM crime, Offence1, Location2, Date1
limit 10000;

alter table incident2 add INCIDENT_ID INT NOT NULL auto_increment primary key;
```

INCIDENT_NUMBER	SHOOTING	OFFENCE_ID	Date_ID	LOCATION_ID	INCIDENT_ID
I182080058		53	36	1	1
I182080058		97	35	1	2
I182080058		52	36	1	3
I182080058		96	35	1	4
I182080058		95	35	1	5
I182080058		94	35	1	6
I182080058		93	35	1	7

Exhibit 25

```
select OFFENSE_CODE, count(OFFENSE_CODE) c
from crime w
group by OFFENSE_CODE
order by  c desc;
```