BPD Implementations

Swapnil
Mona

Github Link

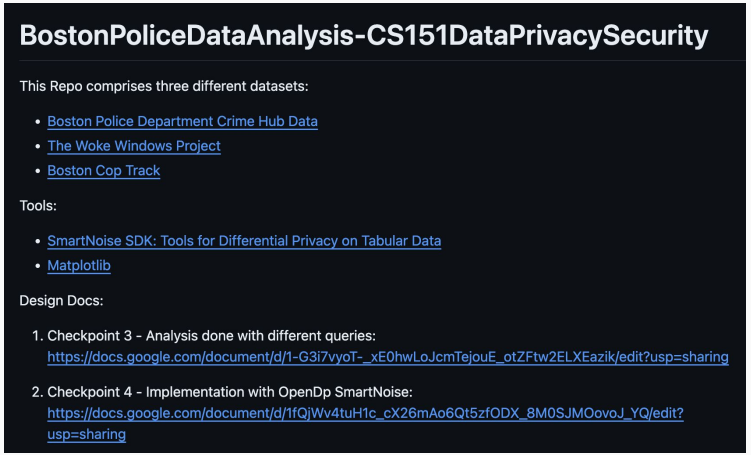
Github link - https://github.com/MonaMaNotAvailable/BostonPoliceDataAnalysis-CS151DataPrivacySecurity

😇 Exact count = 4139 😈 Noisy Count = 4133 with epsilon 4.0

HighestNeighAllegations

neighborhood	frequency
Dorchester	1051
Hyde Park	848
West Roxbury	703
Roxbury	306
Roslindale	263
Mattapan	225
South Boston	212
East Boston	133
Jamaica Plain	109
Allston/Brighton	105
Charlestown	63
South End	41
Central Boston	32
Fenway/Kenmore	32
Back Bay/Beacon Hill	16

```
+----------------------+-----------+
|     neighborhood     | frequency |
+----------------------+-----------+
|      Dorchester      |    1051   |
|      Hyde Park       |    847    |
|     West Roxbury     |    703    |
|       Roxbury        |    306    |
|      Roslindale      |    262    |
|       Mattapan       |    225    |
|     South Boston     |    211    |
|     East Boston      |    132    |
|    Jamaica Plain     |    109    |
|   Allston/Brighton   |    105    |
|     Charlestown      |     63    |
|      South End       |     41    |
|    Central Boston    |     31    |
|    Fenway/Kenmore    |     31    |
| Back Bay/Beacon Hill |     16    |
+----------------------+-----------+
```



5 Different SQL Queries

Query | Query History

```
1  SELECT allegation, COUNT(*) AS frequency
2  FROM bpd_allegations
3  GROUP BY allegation
4  ORDER BY frequency DESC;
```

Figure 1. Count of Allegations by Type

Query | Query History

```
1  SELECT neighborhood, COUNT(*) AS frequency
2  FROM bpd_allegations
3  WHERE neighborhood IS NOT NULL
4  GROUP BY neighborhood
5  ORDER BY frequency DESC;
```

Figure 2. Allegation Counts by Neighborhood

Query | Query History

```
1  SELECT
2      first_name,
3      last_name,
4      COUNT(allegation)/2 AS allegation_count
5  FROM
6      AllegationCountOnOfficers.AllegationCountOnOfficers
7  WHERE
8      active = TRUE
9  GROUP BY
10     first_name,
11     last_name
12 ORDER BY allegation_count DESC
```

Figure 3. Adjusted Allegation Counts per Officer

Query | Query History

```
1  SELECT title, AVG(total) AS avgSalaries
2  FROM officers
3  WHERE title IS NOT NULL
4  GROUP BY title
5  HAVING AVG(total) IS NOT NULL
6  ORDER BY avgSalaries DESC;
```

Figure 4. Average Salaries of Officers by Title or Rank

Query | Query History

```
1  SELECT NEIGHBORHOOD, COUNT(*) AS totalVictims
2  FROM PersonShot.PersonShot
3  WHERE NEIGHBORHOOD IS NOT NULL
4  GROUP BY NEIGHBORHOOD
5  ORDER BY totalVictims DESC;
```

Figure 5. Shootings Counts by Neighborhood

Usage of Privacy Budget

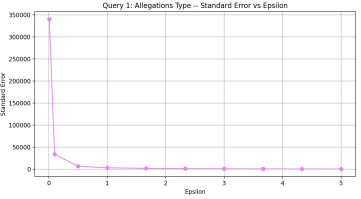
- First, total epsilon is divided in a sequential manner between all these queries, i.e. Total epsilon 50 ⇒ 50/5 ⇒ 10 per query
- High epsilon is chosen to provide more accuracy.
- Low epsilon is chosen to provide more noise.

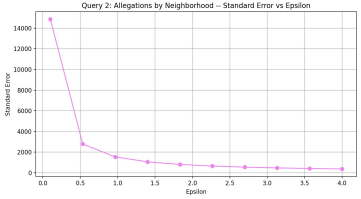
	Q1	Q2	Q3	Q4	Q5	Total
Lower Bound	0.01	0.1	0.05	9.0	1.0	10.16
Upper Bound	5.0	4.0	3.0	60.0	2.2	74.2
Assigned Budget	3.37	2.70	2.02	40.43	1.48	50

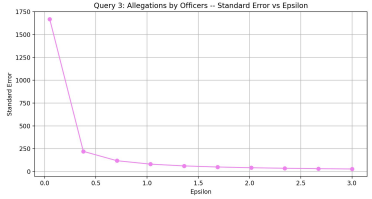
Table 1: Range of epsilon for different queries

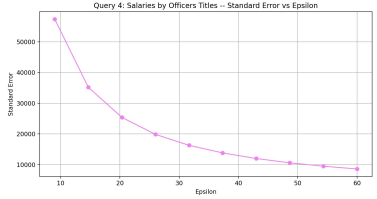
Epsilon vs Error
&
Epsilon vs Runtime

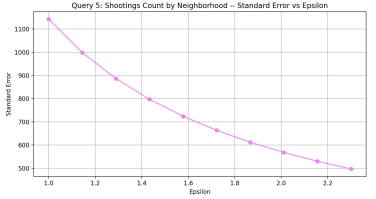
Q1 Q2 Q3 Q4 Q5

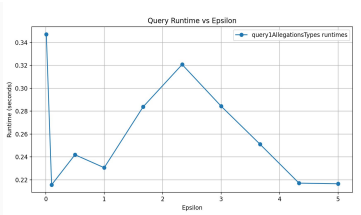


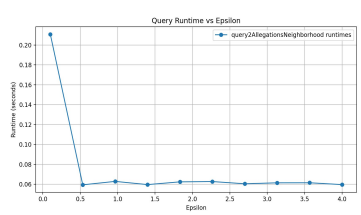


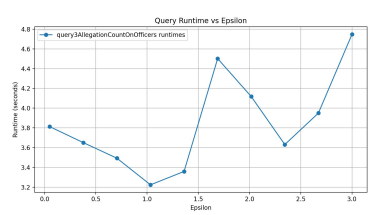


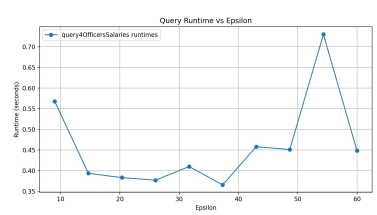


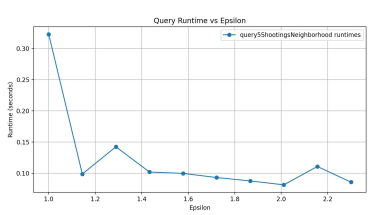






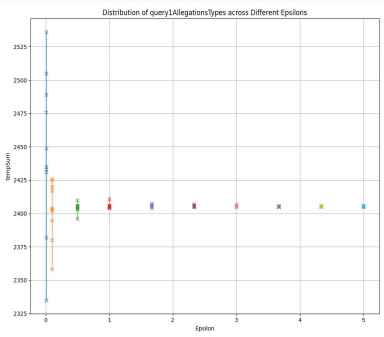




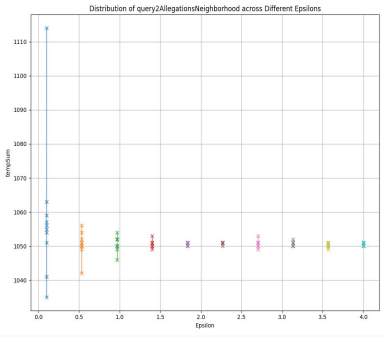


Distribution of Results after 10 Executions

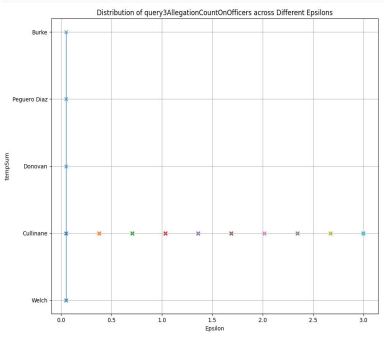
Q1



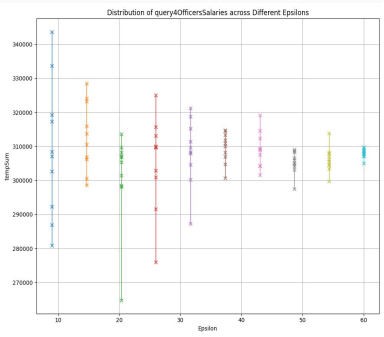
Q2



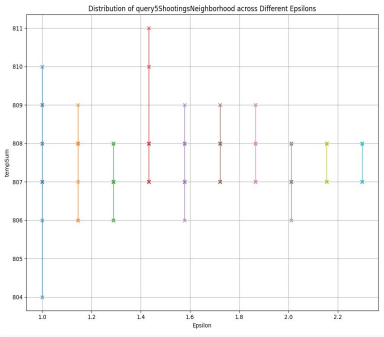
Q3



Q4

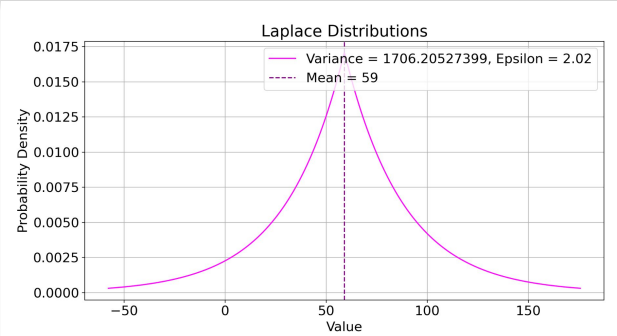


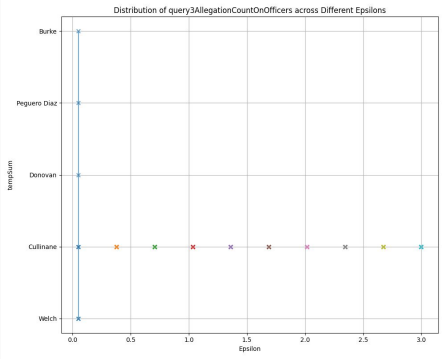
Q5



Comparing the analytical result from checkpoint 3

- Query 3: Number of allegation count per police officer
- Empirical Result:
  - Error: ~**2.09 (e=40)**, ~ **8.34 (e=10)** -> final count in the range of (57,61) and (51,67)
- Actual Result:
  - Error: **27.81 (e=3), 1668.77 (e=0.05)** -> as precise as possible but with noise added
  - Actual privacy budget allocated ⇒ **Error: 41.31 (e = 2.02)**





Thank you for listening!

Any question?