Project Stage 3: Database Implementation and Indexing

By: Sailaja Nallacheruvu, Qi Wu, Aghalya Narayanan, Matthew Chung

Project: SafeLA

1. Database Implementation

```
Database changed
mysql> DESCRIBE AreaInLA;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| AreaId   | int          | NO   | PRI | NULL    |       |
| AreaName | varchar(255) | YES  | MUL | NULL    |       |
+----------+--------------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> DESCRIBE Location;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| Address    | varchar(255) | NO   | PRI | NULL    |       |
| DistrictId | int          | YES  |     | NULL    |       |
| Latitude   | double       | YES  |     | NULL    |       |
| Longitude  | double       | YES  |     | NULL    |       |
| AreaId     | int          | NO   | PRI | NULL    |       |
+------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> DESCRIBE Crime;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| CrimeCd   | int          | NO   | PRI | NULL    |       |
| CrimeDesc | varchar(255) | YES  | MUL | NULL    |       |
| Severity  | varchar(45)  | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> DESCRIBE Report;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| ReportId   | int          | NO   | PRI | NULL    |       |
| CrimeCd    | int          | YES  | MUL | NULL    |       |
| Address    | varchar(255) | YES  | MUL | NULL    |       |
| Premise    | varchar(255) | YES  |     | NULL    |       |
| Date       | datetime     | YES  |     | NULL    |       |
| Time       | varchar(255) | YES  |     | NULL    |       |
| WeaponUsed | varchar(255) | YES  |     | NULL    |       |
| CaseStatus | varchar(255) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)

mysql> DESCRIBE VictimInfo;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| VictimId  | int          | NO   | PRI | NULL    | auto_increment |
| ReportId  | int          | NO   | PRI | NULL    |                |
| Gender    | varchar(255) | YES  |     | NULL    |                |
| Age       | int          | YES  |     | NULL    |                |
| Ethnicity | varchar(255) | YES  |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

CREATE DATABASE SafeLA;

CREATE TABLE Crime (
CrimeCd INT NOT NULL ,
CrimeDesc VARCHAR(255),
Severity VARCHAR(255),
PRIMARY KEY(CrimeCd));

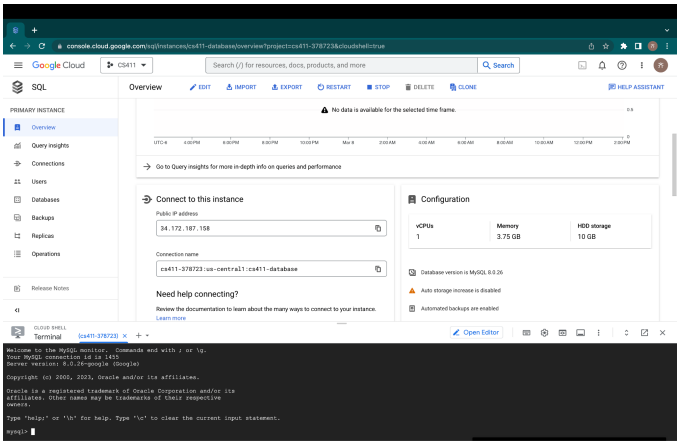
CREATE TABLE AreaInLA (
AreaId INT NOT NULL,
AreaName VARCHAR(255),
PRIMARY KEY(AreaId));

CREATE TABLE Location (
Address VARCHAR(255) NOT NULL,
DistrictId INT,
Latitude REAL,
Longitude REAL,
AreaId INT NOT NULL,
PRIMARY KEY (Address, AreaId),
FOREIGN KEY (AreaId)
REFERENCES AreaInLA(AreaId) ON
UPDATE CASCADE ON DELETE CASCADE);

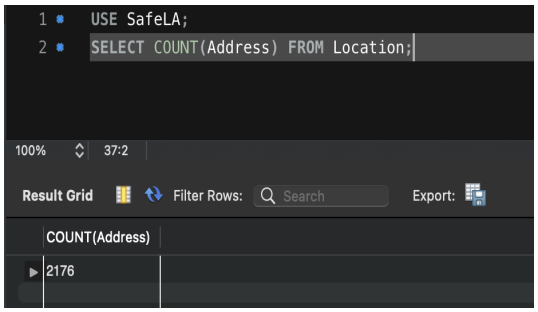
CREATE TABLE Report (
ReportId INT NOT NULL,
CrimeCd INT,
Address VARCHAR(255),
Premise VARCHAR(255),
Date DATE,
Time VARCHAR(255),
WeaponUsed VARCHAR(255),
CaseStatus VARCHAR(255),
PRIMARY KEY (ReportId),
FOREIGN KEY (CrimeCd)
REFERENCES Crime(CrimeCd) ON UPDATE
CASCADE ON DELETE SET NULL,
FOREIGN KEY (Address)
REFERENCES Location(Address) ON
UPDATE CASCADE ON DELETE SET NULL);

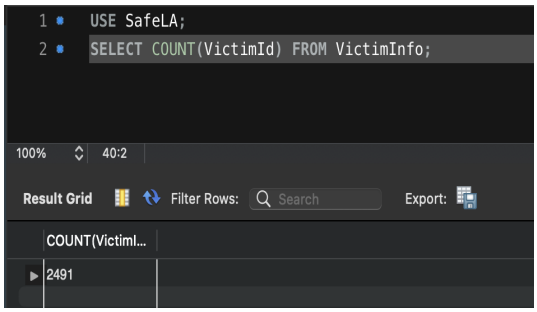
CREATE TABLE VictimInfo (
VictimId INT NOT NULL,
ReportId INT NOT NULL,
Gender VARCHAR(255),
Age INT,
Ethnicity VARCHAR(255),
PRIMARY KEY(VictimId, ReportId),
FOREIGN KEY (ReportId)
REFERENCES Report(ReportId) ON
UPDATE CASCADE ON DELETE
CASCADE)

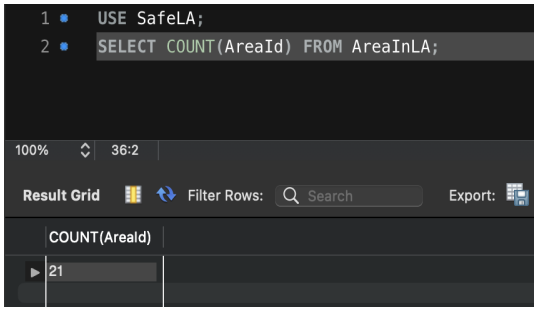
2. GCP Environment

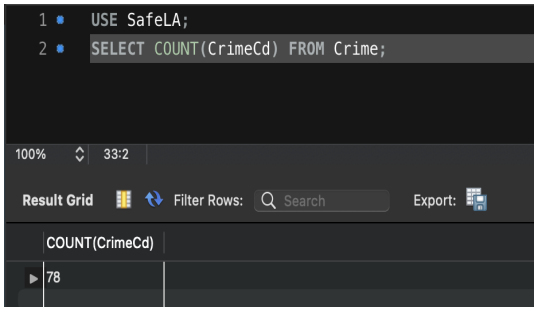


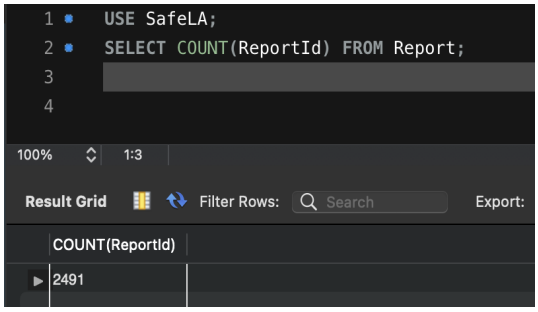
3. Data Insertion







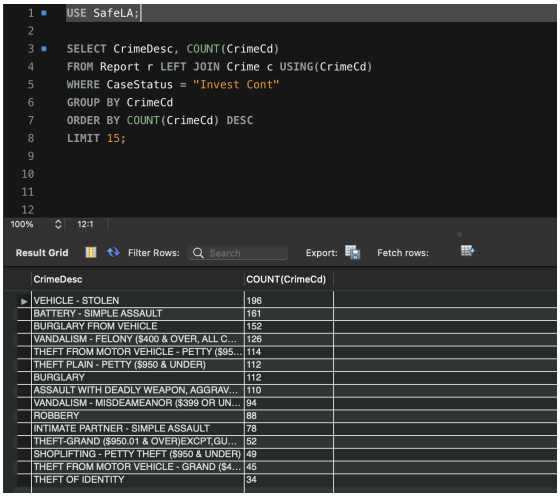




4. Advance Queries

#The total number of ongoing crime investigations categorized by type in LA

```
USE SafeLA;
SELECT CrimeDesc, COUNT(CrimeCd)
FROM Report r LEFT JOIN Crime c
USING(CrimeCd)
WHERE CaseStatus = "Invest Cont"
GROUP BY CrimeCd
ORDER BY COUNT(CrimeCd) DESC
LIMIT 15;
```



```
USE SafeLA;

SELECT CrimeDesc, COUNT(CrimeCd)
FROM Report r LEFT JOIN Crime c USING(CrimeCd)
WHERE CaseStatus = "Invest Cont"
GROUP BY CrimeCd
ORDER BY COUNT(CrimeCd) DESC
LIMIT 15;
```

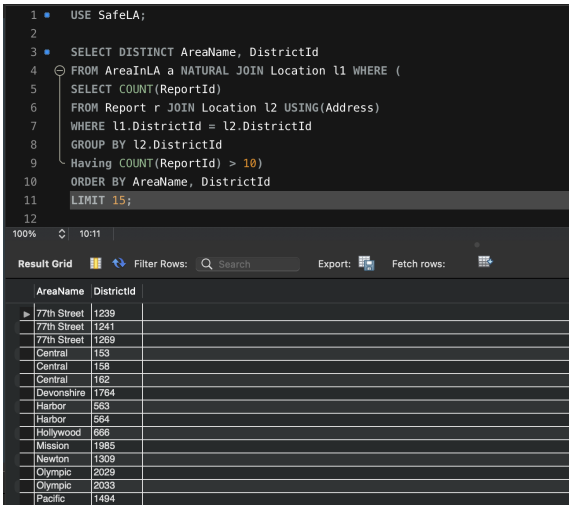
100% 12:1

Result Grid | Filter Rows: Search | Export: | Fetch rows:

CrimeDesc	COUNT(CrimeCd)
VEHICLE - STOLEN	196
BATTERY - SIMPLE ASSAULT	161
BURGLARY FROM VEHICLE	152
VANDALISM - FELONY ($400 & OVER, ALL C...	126
THEFT FROM MOTOR VEHICLE - PETTY ($95...	114
THEFT PLAIN - PETTY ($950 & UNDER)	112
BURGLARY	112
ASSAULT WITH DEADLY WEAPON, AGGRAV...	110
VANDALISM - MISDEAMEANOR ($399 OR UN...	94
ROBBERY	88
INTIMATE PARTNER - SIMPLE ASSAULT	78
THEFT-GRAND ($950.01 & OVER)EXCPT,GU...	52
SHOPLIFTING - PETTY THEFT ($950 & UNDER)	49
THEFT FROM MOTOR VEHICLE - GRAND ($4...	45
THEFT OF IDENTITY	34

#Show the AreaName and DistrictId where more than 10 crimes happened in this District.

```
USE SafeLA;
SELECT DISTINCT AreaName, DistrictId
FROM AreaInLA a NATURAL JOIN
Location l1 WHERE (
        SELECT COUNT(ReportId)
        FROM Report r JOIN Location l2
        USING(Address)
        WHERE l1.DistrictId = l2.DistrictId
        GROUP BY l2.DistrictId
        Having COUNT(ReportId) > 10)
ORDER BY DistrictId
LIMIT 15;
```



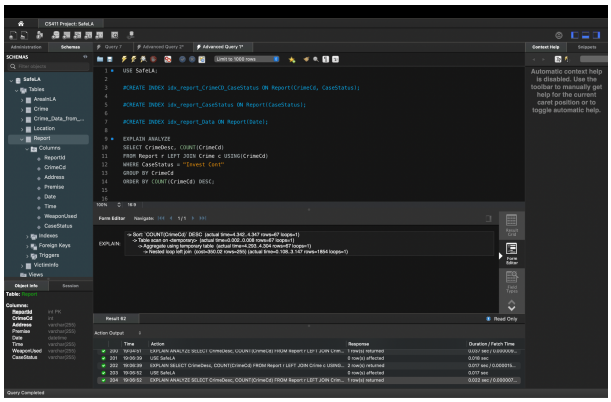
```
USE SafeLA;

SELECT DISTINCT AreaName, DistrictId
FROM AreaInLA a NATURAL JOIN Location l1 WHERE (
    SELECT COUNT(ReportId)
    FROM Report r JOIN Location l2 USING(Address)
    WHERE l1.DistrictId = l2.DistrictId
    GROUP BY l2.DistrictId
    Having COUNT(ReportId) > 10)
    ORDER BY AreaName, DistrictId
    LIMIT 15;
```

100% 10:11

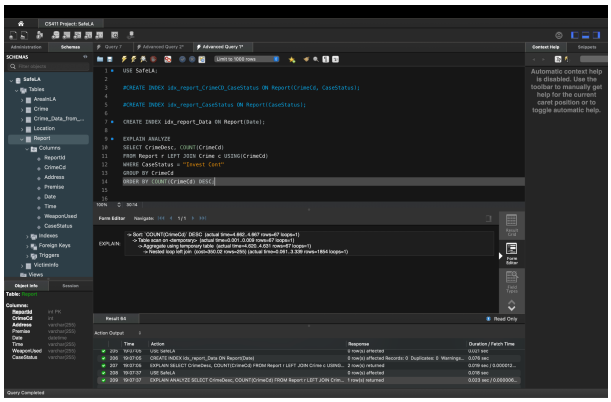
Result Grid | Filter Rows: Search | Export: | Fetch rows:

AreaName	DistrictId
77th Street	1239
77th Street	1241
77th Street	1269
Central	153
Central	158
Central	162
Devonshire	1764
Harbor	563
Harbor	564
Hollywood	666
Mission	1985
Newton	1309
Olympic	2029
Olympic	2033
Pacific	1494

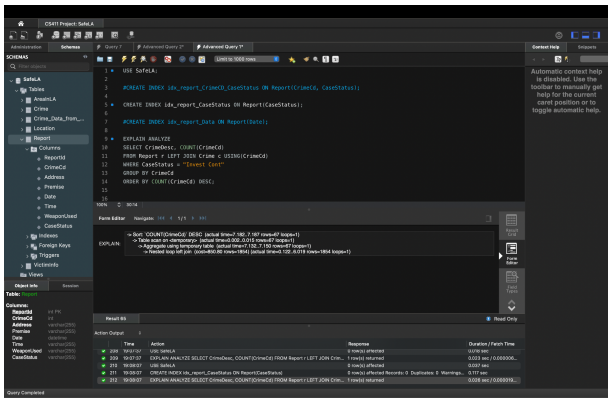
5. Indexing:
    a. Advanced Query 1



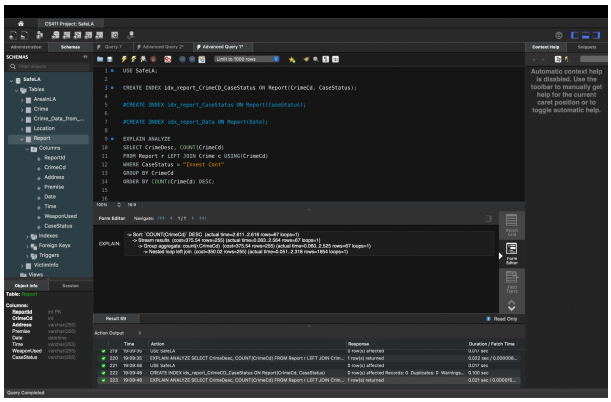
(a)



(b)



(c)



(d)

Figure (a) shows that using the default indexing, the query takes 4.3ms to run.

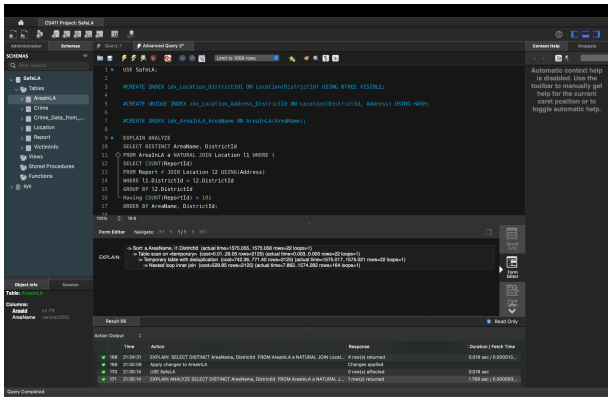
To determine whether an irrelevant index affects query performance, we add a new index using the attribute "Date" in the Report table. As expected in figure (b), the query takes about 4.6ms to run, which is roughly the same amount of time as with the default indexing. MySQL automatically chooses what it deems to be the best index for querying, and in this particular scenario, there is no way to improve performance beyond using the default index. Therefore, MySQL resorts to using the default index for the query.

Next, we attempted to improve query performance by using the "Case Status" attribute as our index in figure (c), given the relevance of our WHERE clause to this attribute. However, this approach proved to be counterproductive, with the query now taking 7.1ms to run. We also observed that the JOIN statement now takes significantly longer than before. The default index may be using "CrimeCd"

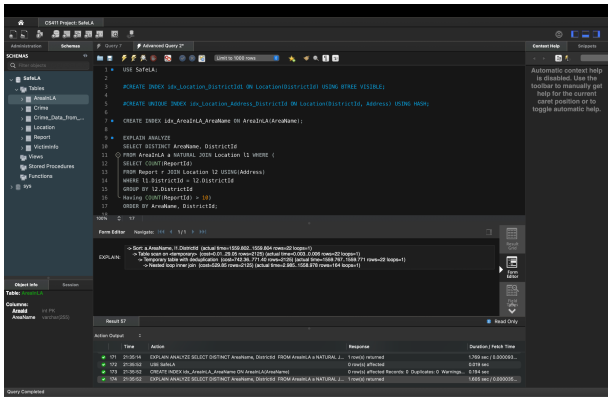
to join with the Crime table, which is why switching to the "Case Status" index takes much longer time, because it requires scanning through the entire table to find matches.

As a final attempt to improve performance, we use a composite index (CrimeCd, Case Status) for the query, as shown in Figure (d). This approach yields a query completion time of only about 2.6ms. We believe this improvement is due to the composite index enhancing the efficiency of both the WHERE and JOIN clauses.

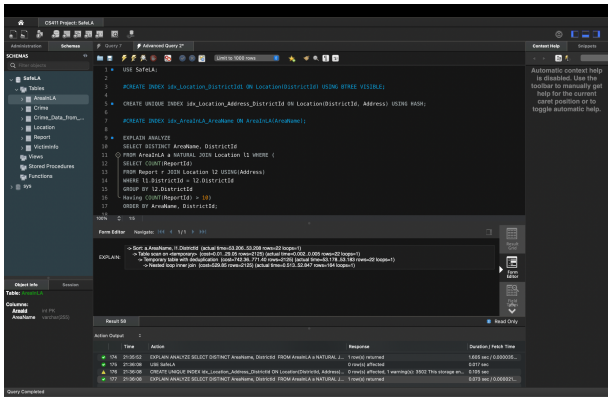
b. Advanced Query 2



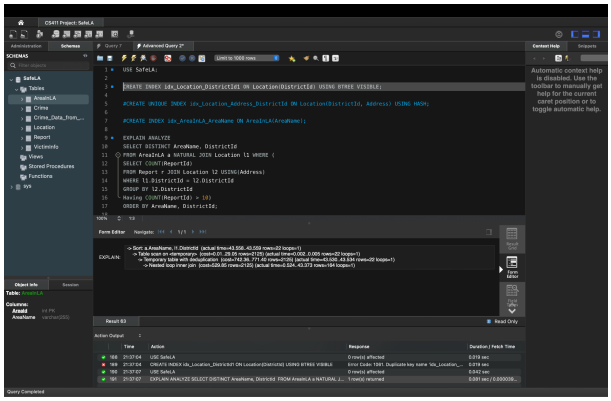
(a)



(b)



(c)



(d)

Figure (a) shows that using the default indexing, the query takes 1575.06 ms to run.

Initially, we attempted to enhance the performance of our query by using "AreaName" as the indexing for "AreaInLA", as shown in the figure(b). Our reasoning was that this could potentially increase the ordering speed. However,

this approach yielded similar results to using the default indexing, taking around 1559.80 ms. We believe that the limited number of areas in LA (21) contributed to the lack of significant improvement. Furthermore, the majority of the query's execution time was spent on joining tables, instead of sorting.

In an effort to improve the performance of our query, we attempted to use "DistrictId" as the indexing. We initially employed a hashing table to index this attribute, as shown in the figure (c), and we found that it yielded a significant improvement in the query's performance, with an execution time of only 53.21 ms. We attribute this enhancement to the extensive utilization of the "DistrictId" attribute in both the WHERE and GROUP clauses of the query.

As the final step in our performance optimization efforts, we compared the use of B-Tree indexing with the previously implemented hashing method for "DistrictId", as illustrated in Figure (d). The initial results demonstrated that B-Tree indexing led to a slightly faster execution time of 43.56 ms. However, we recognize that this could have been a coincidence. After conducting several additional runs, we found that the two indexing methods yielded similar execution time.

B-Tree indexing is good for range queries and sorting, while hashing indexing is faster for exact match lookups. In this case, the similarity in performance between the two indexing methods may be attributed to the size and distribution of our data.