DDL Commands

```
CREATE TABLE Users(username VARCHAR(255) PRIMARY KEY,
                   password VARCHAR(255));

CREATE TABLE Views( view_id INT PRIMARY KEY,
                    longitude REAL,
                    latitude REAL,
                    radius REAL);

CREATE TABLE WeaponType(weapon_code INT PRIMARY KEY,
                        weapon_desc VARCHAR(255));

CREATE TABLE PremisType(premis_code INT PRIMARY KEY,
                        premis_desc VARCHAR(255));

CREATE TABLE CrimeType(crime_code INT PRIMARY KEY,
                       crime_desc VARCHAR(255));

CREATE TABLE CrimeFilter( Username VARCHAR(255),
                          crime_code INT,
                          PRIMARY KEY(Username, crime_code),
                          FOREIGN KEY(Username) REFERENCES Users(Username),
                          FOREIGN KEY(crime_code) REFERENCES CrimeType(crime_code));

CREATE TABLE WeaponFilter( Username VARCHAR(255),
                           weapon_code INT,
                           PRIMARY KEY(Username, weapon_code),
                           FOREIGN KEY(Username) REFERENCES Users(Username),
                          FOREIGN KEY(weapon_code) REFERENCES
                 WeaponType(weapon_code));

CREATE TABLE PremisFilter( Username VARCHAR(255),
                           premis_code INT, PRIMARY KEY(Username, premis_code),
                           FOREIGN KEY(Username) REFERENCES Users(Username),
                           FOREIGN KEY(premis_code) REFERENCES PremisType(premis_code));

CREATE TABLE Crimes(dr_no INT PRIMARY KEY,
                    date_occ DATE,
                    time_occ TIME,
                    crime_code INT ,
                    weapon_code INT,
```

```
premis_code INT,
vict_age INT,
vict_sex VARCHAR(255),
vict_descent VARCHAR(255),
latitude REAL,
longitude REAL,
FOREIGN KEY(crime_code) REFERENCES CrimeType(crime_code),
FOREIGN KEY(weapon_code) REFERENCES WeaponType(weapon_code),
FOREIGN KEY(premis_code) REFERENCES PremisType(premis_code));
```

Importing Data into tables

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_la_crime |
+--------------------+
| CrimeFilter        |
| CrimeType          |
| Crimes             |
| PremisFilter       |
| PremisType         |
| Users              |
| Views              |
| WeaponFilter       |
| WeaponType         |
+--------------------+
9 rows in set (0.00 sec)
```

Crimes table

```
mysql> SELECT COUNT(*) FROM Crimes;
+----------+
| COUNT(*) |
+----------+
|   116477 |
+----------+
1 row in set (0.02 sec)
```

Users table

```
mysql> SELECT COUNT(*) FROM Users;
+----------+
| COUNT(*) |
+----------+
|     1500 |
+----------+
1 row in set (0.01 sec)
```

CrimeFilter table

```
mysql> SELECT COUNT(*) FROM CrimeFilter;
+----------+
| COUNT(*) |
+----------+
|     3263 |
+----------+
1 row in set (0.00 sec)
```

WeaponFilter table

```
mysql> SELECT COUNT(*) FROM WeaponFilter;
+----------+
| COUNT(*) |
+----------+
|     2955 |
+----------+
1 row in set (0.01 sec)
```

PremisFilter table

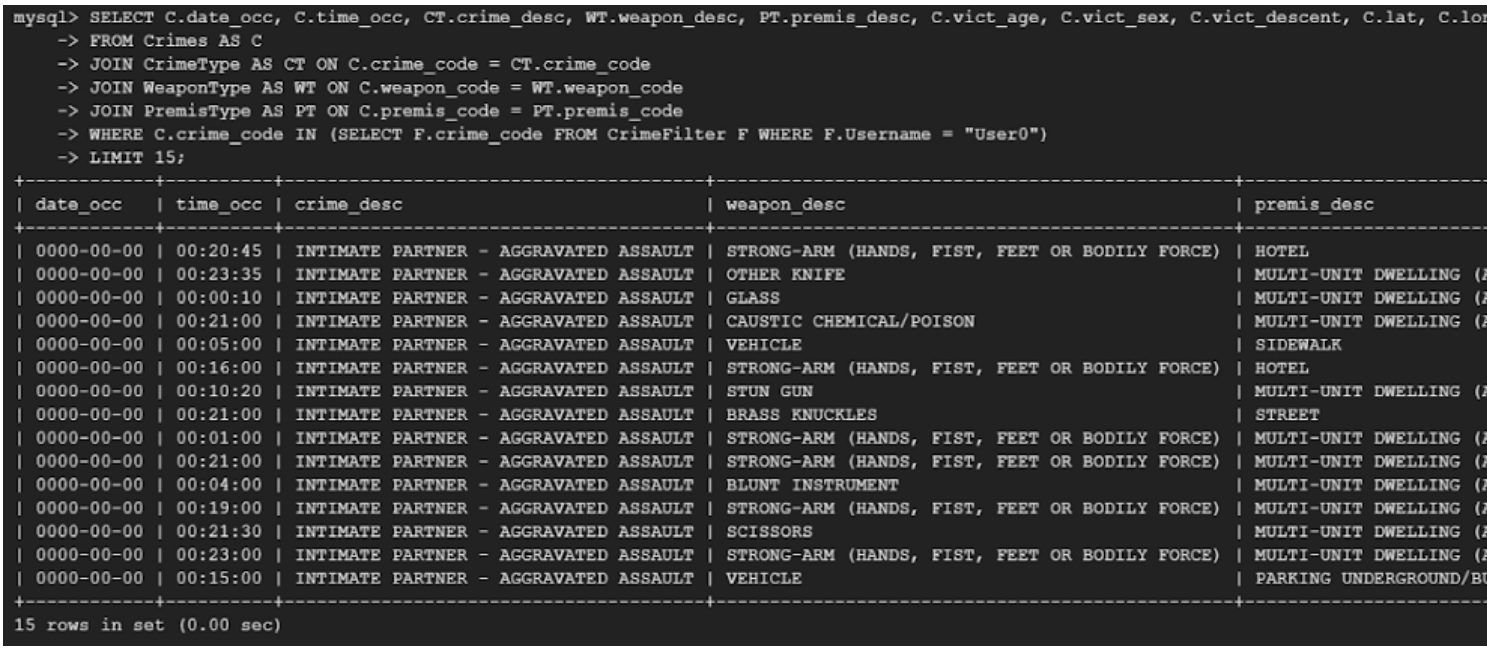
```
mysql> SELECT COUNT(*) FROM PremisFilter;
+----------+
| COUNT(*) |
+----------+
|     2991 |
+----------+
1 row in set (0.00 sec)
```

Advanced Queries

Query 1

Select crimes that fall under a user's ("User0") crime filters.

```
SELECT C.date_occ, C.time_occ, CT.crime_desc, WT.weapon_desc, PT.premis_desc, C.vict_age, C.vict_sex, C.vict_descent, C.lat, C.longitude
FROM Crimes AS C
JOIN CrimeType AS CT ON C.crime_code = CT.crime_code
JOIN WeaponType AS WT ON C.weapon_code = WT.weapon_code
JOIN PremisType AS PT ON C.premis_code = PT.premis_code
WHERE C.crime_code = 624
LIMIT 15;
```



```
mysql> SELECT C.date_occ, C.time_occ, CT.crime_desc, WT.weapon_desc, PT.premis_desc, C.vict_age, C.vict_sex, C.vict_descent, C.lat, C.lon
    -> FROM Crimes AS C
    -> JOIN CrimeType AS CT ON C.crime_code = CT.crime_code
    -> JOIN WeaponType AS WT ON C.weapon_code = WT.weapon_code
    -> JOIN PremisType AS PT ON C.premis_code = PT.premis_code
    -> WHERE C.crime_code IN (SELECT F.crime_code FROM CrimeFilter F WHERE F.Username = "User0")
    -> LIMIT 15;
+------------+----------+--------------------------------------+-----------------------------------------------+----------------------
| date_occ   | time_occ | crime_desc                           | weapon_desc                                   | premis_desc
+------------+----------+--------------------------------------+-----------------------------------------------+----------------------
| 0000-00-00 | 00:20:45 | INTIMATE PARTNER - AGGRAVATED ASSAULT | STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) | HOTEL
| 0000-00-00 | 00:23:35 | INTIMATE PARTNER - AGGRAVATED ASSAULT | OTHER KNIFE                                   | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:00:10 | INTIMATE PARTNER - AGGRAVATED ASSAULT | GLASS                                         | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:21:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | CAUSTIC CHEMICAL/POISON                       | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:05:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | VEHICLE                                       | SIDEWALK
| 0000-00-00 | 00:16:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) | HOTEL
| 0000-00-00 | 00:10:20 | INTIMATE PARTNER - AGGRAVATED ASSAULT | STUN GUN                                      | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:21:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | BRASS KNUCKLES                                | STREET
| 0000-00-00 | 00:01:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:21:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:04:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | BLUNT INSTRUMENT                              | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:19:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:21:30 | INTIMATE PARTNER - AGGRAVATED ASSAULT | SCISSORS                                      | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:23:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) | MULTI-UNIT DWELLING (
| 0000-00-00 | 00:15:00 | INTIMATE PARTNER - AGGRAVATED ASSAULT | VEHICLE                                       | PARKING UNDERGROUND/BU
+------------+----------+--------------------------------------+-----------------------------------------------+----------------------
15 rows in set (0.00 sec)
```

Query 2

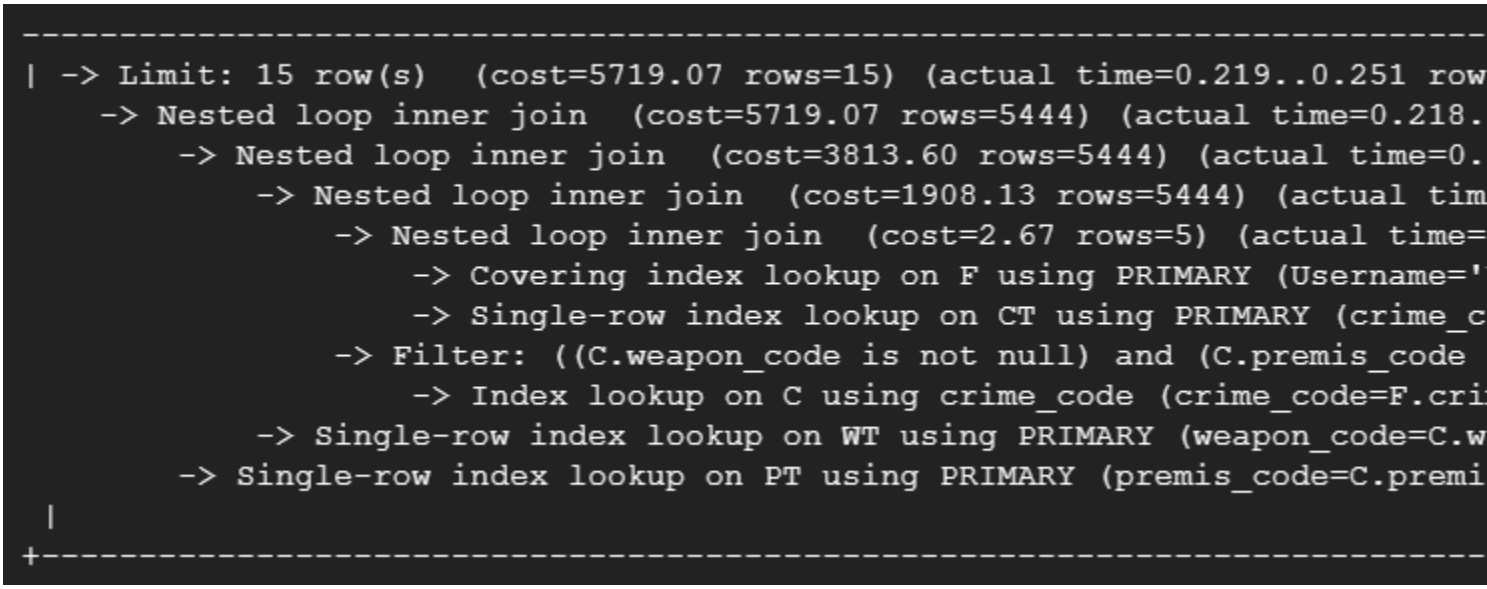
Select the most common crimes for each type of premise.

```
SELECT premis_code,
    (SELECT crime_code
    FROM Crimes
    WHERE premis_code = C.premis_code
    GROUP BY crime_code
    ORDER BY COUNT(*) DESC LIMIT 1) AS most_common_crime
```

FROM Crimes C
GROUP BY premis_code
ORDER BY premis_code ASC LIMIT 15;

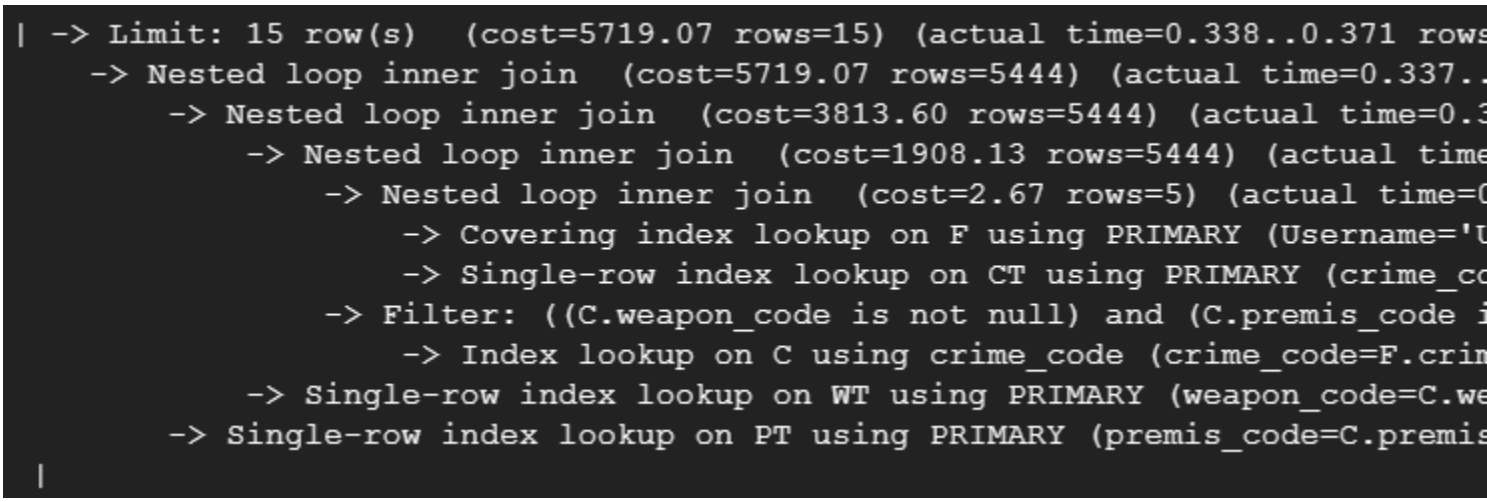
```
mysql> SELECT premis_code,
    -> (SELECT crime_code
    -> FROM Crimes
    -> WHERE premis_code = C.premis_code
    -> GROUP BY crime_code
    -> ORDER BY COUNT(*) DESC LIMIT 1) AS most_common_crime
    -> FROM Crimes C
    -> GROUP BY premis_code
    -> ORDER BY premis_code ASC LIMIT 15;
+-------------+--------------------+
| premis_code | most_common_crime |
+-------------+--------------------+
|         101 |                230 |
|         102 |                624 |
|         103 |                230 |
|         104 |                624 |
|         105 |                626 |
|         106 |                626 |
|         107 |                230 |
|         108 |                624 |
|         109 |                230 |
|         110 |                626 |
|         111 |                624 |
|         112 |                624 |
|         114 |                230 |
|         115 |                230 |
|         116 |                624 |
+-------------+--------------------+
15 rows in set, 1 warning (0.07 sec)
```

Query 1



This is the query1 with no indexes.

CREATE INDEX idx_crimefilter_usrname ON CrimeFilter (Username);



The cost is exactly the same because the User column is sorted, and User0 is always the first value in the column that is checked. The time is increased with the index because the query must now access the index and then get the value rather than go straight to the value.

CREATE INDEX crime_idx ON Crimes (crime_code, weapon_code, premis_code);

This index makes Query 1 have a lower cost since The index lookups do not have to sort through as many rows for crime_code, weapon_code, and premis_code. The query time did increase, probably due to the time spent indexing being greater than the time saved going through less rows.

```
------------------------------------------+
| -> Limit: 15 row(s)  (cost=4921.43 rows=15) (actual time=0.283..0.301 rows=15 loops=1)
    -> Nested loop inner join  (cost=4921.43 rows=4685) (actual time=0.282..0.299 rows=15 loops=1)
        -> Nested loop inner join  (cost=3281.84 rows=4685) (actual time=0.277..0.286 rows=15 loops=1)
            -> Nested loop inner join  (cost=1642.25 rows=4685) (actual time=0.273..0.276 rows=15 loops=1)
                -> Nested loop inner join  (cost=2.67 rows=5) (actual time=0.050..0.050 rows=1 loops=1)
                    -> Covering index lookup on F using PRIMARY (Username='User0')  (cost=0.92 rows=5) (actual time=0.040..0.040 rows=1 loops=1)
                    -> Single-row index lookup on CT using PRIMARY (crime_code=F.crime_code)  (cost=0.27 rows=1) (actual time=0.009..0.009 rows=1 loops=1)
                -> Index lookup on C using crime_idx (crime_code=F.crime_code), with index condition: ((C.weapon_code is not null) and (C.premis_code is not null))  (cost=252.96 rows=937) (actual time=0.222..0.225 rows=15 loops=1)
            -> Single-row index lookup on WT using PRIMARY (weapon_code=C.weapon_code)  (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=15)
        -> Single-row index lookup on PT using PRIMARY (premis_code=C.premis_code)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=15)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

CREATE INDEX crimetype_idx ON CrimeType (crime_code);

```
| -> Limit: 15 row(s)  (cost=4921.43 rows=15) (actual time=0.257..0.275 rows=15 loops=1)
    -> Nested loop inner join  (cost=4921.43 rows=4685) (actual time=0.256..0.273 rows=15 loops=1)
        -> Nested loop inner join  (cost=3281.84 rows=4685) (actual time=0.250..0.259 rows=15 loops=1)
            -> Nested loop inner join  (cost=1642.25 rows=4685) (actual time=0.245..0.249 rows=15 loops=1)
                -> Nested loop inner join  (cost=2.67 rows=5) (actual time=0.019..0.019 rows=1 loops=1)
                    -> Covering index lookup on F using PRIMARY (Username='User0')  (cost=0.92 rows=5) (actual time=0.010..0.010 rows
                    -> Single-row index lookup on CT using PRIMARY (crime_code=F.crime_code)  (cost=0.27 rows=1) (actual time=0.008..
                -> Index lookup on C using crime_idx (crime_code=F.crime_code), with index condition: ((C.weapon_code is not null) an
            -> Single-row index lookup on WT using PRIMARY (weapon_code=C.weapon_code)  (cost=0.25 rows=1) (actual time=0.000..0.000
        -> Single-row index lookup on PT using PRIMARY (premis_code=C.premis_code)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows
 |
+----------------------------------------------------------------------------------------------------------------------------------
```

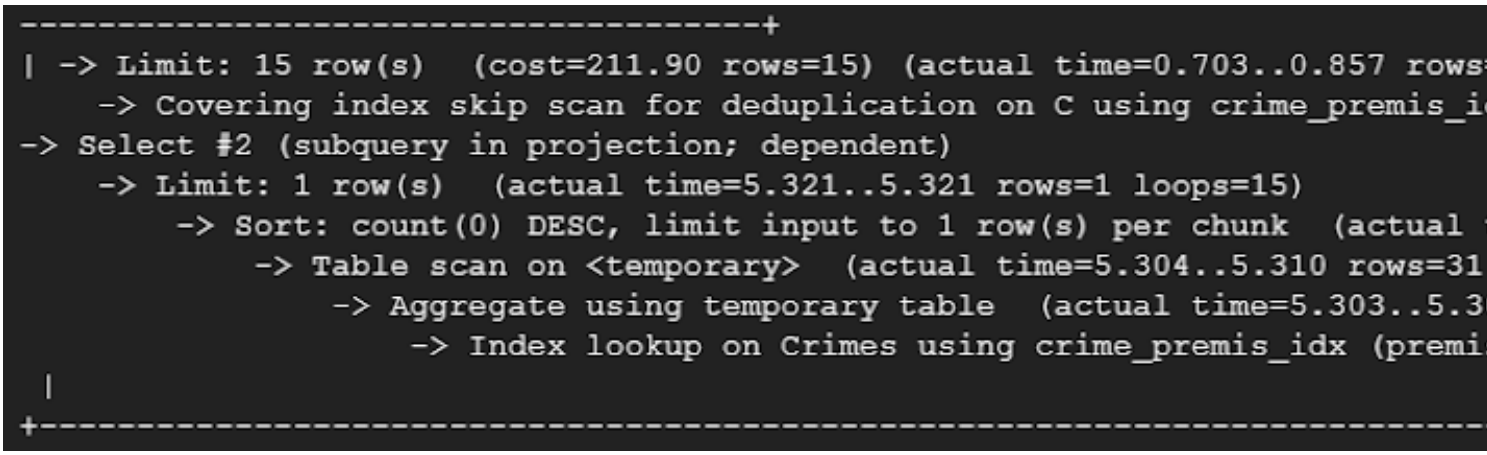
This index gives an improvement on time and cost and this could be due to crime_codes being an extremely varying value as there are many different variations of crime_codes and CrimeType crime_code is one of the filters that could be accessed by the user.

Query 2

```
-------------------------+
| -> Limit: 15 row(s)  (cost=92.65 rows=15) (actual time=0.030..0.088 rows=
    -> Covering index skip scan for deduplication on C using premis_code  (
-> Select #2 (subquery in projection; dependent)
    -> Limit: 1 row(s)  (actual time=3.954..3.954 rows=1 loops=15)
        -> Sort: count(0) DESC, limit input to 1 row(s) per chunk  (actual
            -> Table scan on <temporary>  (actual time=3.945..3.948 rows=31
                -> Aggregate using temporary table  (actual time=3.944..3.9
                    -> Index lookup on Crimes using premis_code (premis_cod
 |
+------------------------------------------------------------------------
```

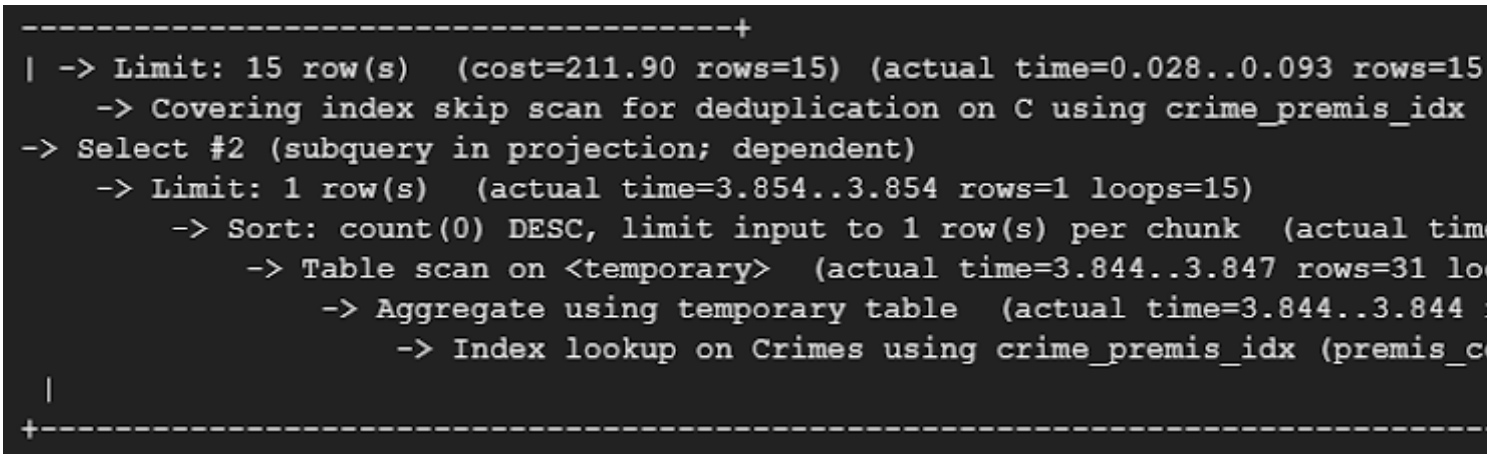
This is query2 with no indexes.

CREATE INDEX crime_premis_idx ON Crimes (premis_code);



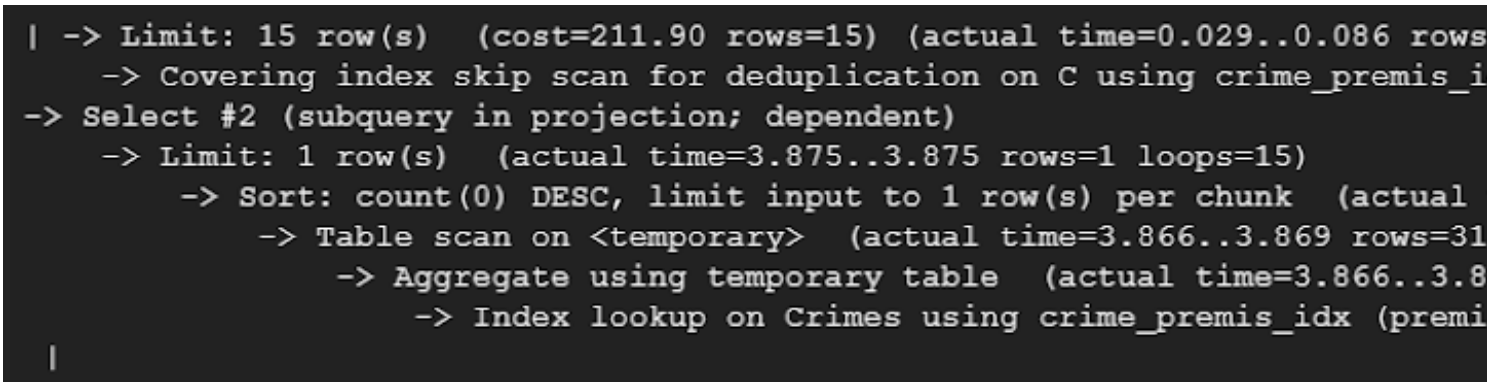
This index actually made no improvements to our time and cost and this could be due to us selecting premis_code in this query and if the index is on premis_code we are limiting the search causing it to be much slower.

CREATE INDEX crime_crime_idx ON Crimes (crime_code);



This index maintained the same cost, while decreasing time. The entire subquery had a significantly reduced time. This is because the index makes the group by and order by much faster.

CREATE INDEX crime_weapon_idx ON Crimes (weapon_code);



This index did not change anything from the previous index. This is because the query selects the crime code based off of the other columns, so it does not affect how quickly the crime_codes can be found.

REVISIONS

In our original upload of Stage 3, we unknowingly indexed the primary key, which is not valid because the indexing operation basically did nothing. The reasoning for this is because the primary keys are already indexed. To fix this, I drop the previous indexes and reindex the non primary keys

WITHOUT INDEX

```
+------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| -> Limit: 15 row(s)  (cost=5984.61 rows=15) (actual time=0.126..0.347 rows=15 loops=1)
    -> Nested loop inner join  (cost=5984.61 rows=8190) (actual time=0.125..0.344 rows=15 loops=1)
        -> Nested loop inner join  (cost=3117.97 rows=8190) (actual time=0.114..0.286 rows=15 loops=1)
            -> Table scan on WT  (cost=8.05 rows=78) (actual time=0.058..0.067 rows=32 loops=1)
            -> Index lookup on C using crime_idx (crime_code=624, weapon_code=WT.weapon_code), with index condition: (C.premis_code is not null)  (cost=29.50 rows=105)
        -> Single-row index lookup on PT using PRIMARY (premis_code=C.premis_code)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=15)
 |
```

WITH INDEX ON NON PK Crime_code

```
+------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| -> Limit: 15 row(s)  (cost=5984.61 rows=15) (actual time=0.113..0.270 rows=15 loops=1)
    -> Nested loop inner join  (cost=5984.61 rows=8190) (actual time=0.111..0.268 rows=15 loops=1)
        -> Nested loop inner join  (cost=3117.97 rows=8190) (actual time=0.101..0.219 rows=15 loops=1)
            -> Table scan on WT  (cost=8.05 rows=78) (actual time=0.044..0.048 rows=32 loops=1)
            -> Index lookup on C using crime_idx (crime_code=624, weapon_code=WT.weapon_code), with index condition: (C.premis_code is not null)  (cost=29.50 rows=105) (actual time=0.005..0.005
        -> Single-row index lookup on PT using PRIMARY (premis_code=C.premis_code)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=15)
 |
+------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
1 row in set (0.01 sec)
```

This yielded a better time as crime_code is directly what we are looking for and when indexing by crime_code, we were able to cut the runtime down because everything is sorted. However the lookup method is still the same which could be because the non-indexed version was coincidentally decently sorted.

WITHOUT INDEX:

```
mysql> EXPLAIN ANALYZE
esc, C.vict_age, C.vict_sex, C.vict_descent, C.lat, C.longitudedesc, PT.premis_d
    -> FROM Crimes AS C
    -> JOIN CrimeType AS CT ON C.crime_code = CT.crime_code
    -> JOIN WeaponType AS WT ON C.weapon_code = WT.weapon_code
    -> JOIN PremisType AS PT ON C.premis_code = PT.premis_code
    -> WHERE C.crime_code = 624
    -> LIMIT 15;
+-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| EXPLAIN


                                     |
+-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| -> Limit: 15 row(s)  (cost=5984.61 rows=15) (actual time=0.126..0.347 rows=15 loops=1)
    -> Nested loop inner join  (cost=5984.61 rows=8190) (actual time=0.125..0.344 rows=15 loops=1)
        -> Nested loop inner join  (cost=3117.97 rows=8190) (actual time=0.114..0.286 rows=15 loops=1)
            -> Table scan on WT  (cost=8.05 rows=78) (actual time=0.058..0.067 rows=32 loops=1)
            -> Index lookup on C using crime_idx (crime_code=624, weapon_code=WT.weapon_code), with index condition: (C.premis_code is not null)  (cost=29
        -> Single-row index lookup on PT using PRIMARY (premis_code=C.premis_code)  (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=15)
 |
+-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
1 row in set (0.00 sec)
```

WITH INDEX ON NON PRIMARY KEY TIME_OCC

```
| EXPLAIN


                                     |
+-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
| -> Limit: 15 row(s)  (cost=5984.61 rows=15) (actual time=0.100..0.302 rows=15 loops=1)
    -> Nested loop inner join  (cost=5984.61 rows=8190) (actual time=0.098..0.299 rows=15 loops=1)
        -> Nested loop inner join  (cost=3117.97 rows=8190) (actual time=0.076..0.258 rows=15 loops=1)
            -> Table scan on WT  (cost=8.05 rows=78) (actual time=0.034..0.039 rows=32 loops=1)
            -> Index lookup on C using crime_idx (crime_code=624, weapon_code=WT.weapon_code), with index condition: (C.premis_co
        -> Single-row index lookup on PT using PRIMARY (premis_code=C.premis_code)  (cost=0.25 rows=1) (actual time=0.002..0.002 r
 |
+-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-------------------------------------+
1 row in set (0.00 sec)
```

This improved our query time by a good amount because there are a variety of possible time occurrences but we believe that we can still index some other columns to have a greater improvement because there is only a finite number of possible time_occ values whereas something like longitude or latitude would have much greater variances.