

## 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 IN (SELECT F.crime_code FROM CrimeFilter F WHERE
F.Username = "User0")
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.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 IN (SELECT F.crime_code FROM CrimeFilter F WHERE F.Username = "User0")
-> LIMIT 15;
```

date_occ	time_occ	crime_desc	weapon_desc	premis_desc	vict_age	vict_sex	vict_descent	lat	longitude
0000-00-00	00:20:45	INTIMATE PARTNER - AGGRAVATED ASSAULT	STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	HOTEL	46	M	H	34.0516	-118.2561
0000-00-00	00:23:35	INTIMATE PARTNER - AGGRAVATED ASSAULT	OTHER KNIFE	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	44	F	H	34.0446	-118.245
0000-00-00	00:00:10	INTIMATE PARTNER - AGGRAVATED ASSAULT	GLASS	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	26	M	B	34.0397	-118.2637
0000-00-00	00:21:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	CAUSTIC CHEMICAL/POISON	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	29	M	B	34.0385	-118.2656
0000-00-00	00:05:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	VEHICLE	SIDEWALK	26	F	W	34.0627	-118.2463
0000-00-00	00:16:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	HOTEL	21	F	B	34.0533	-118.2561
0000-00-00	00:10:20	INTIMATE PARTNER - AGGRAVATED ASSAULT	STUN GUN	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	40	F	W	34.044	-118.2467
0000-00-00	00:21:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	BRASS KNUCKLES	STREET	40	M	B	34.0493	-118.2416
0000-00-00	00:01:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	38	F	W	34.039	-118.24
0000-00-00	00:21:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	26	F	H	34.0423	-118.2666
0000-00-00	00:04:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	BLUNT INSTRUMENT	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	38	M	B	34.0474	-118.2496
0000-00-00	00:19:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	45	F	B	34.0449	-118.2496
0000-00-00	00:21:30	INTIMATE PARTNER - AGGRAVATED ASSAULT	SCISSORS	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	38	M	B	34.0503	-118.2433
0000-00-00	00:23:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC)	46	F	W	34.0428	-118.2461
0000-00-00	00:15:00	INTIMATE PARTNER - AGGRAVATED ASSAULT	VEHICLE	PARKING UNDERGROUND/BUILDING	28	M	W	34.0466	-118.252

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

```
-----+
| -> Limit: 15 row(s) (cost=5719.07 rows=15) (actual time=0.219..0.251 rows=15 loops=1)
|   -> Nested loop inner join (cost=5719.07 rows=5444) (actual time=0.218..0.249 rows=15 loops=1)
|     -> Nested loop inner join (cost=3813.60 rows=5444) (actual time=0.214..0.236 rows=15 loops=1)
|       -> Nested loop inner join (cost=1908.13 rows=5444) (actual time=0.209..0.215 rows=15 loops=1)
|         -> Nested loop inner join (cost=2.67 rows=5) (actual time=0.030..0.030 rows=1 loops=1)
|           -> Covering index lookup on F using PRIMARY (Username='User0') (cost=0.92 rows=5) (actual time=0.019..0.019 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)
|         -> Filter: ((C.weapon_code is not null) and (C.premis_code is not null)) (cost=293.99 rows=1089) (actual time=0.179..0.184 rows=15 loops=1)
|           -> Index lookup on C using crime_code (crime_code=F.crime_code) (cost=293.99 rows=1089) (actual time=0.178..0.181 rows=15 loops=1)
|           -> Single-row index lookup on WT using PRIMARY (weapon_code=C.weapon_code) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1)
|         -> 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=1)
|
|-----+
```

This is the query1 with no indexes.

CREATE INDEX idx\_crimefilter\_username ON CrimeFilter (Username);

```
-----+
| -> Limit: 15 row(s) (cost=5719.07 rows=15) (actual time=0.338..0.371 rows=15 loops=1)
|   -> Nested loop inner join (cost=5719.07 rows=5444) (actual time=0.337..0.369 rows=15 loops=1)
|     -> Nested loop inner join (cost=3813.60 rows=5444) (actual time=0.332..0.354 rows=15 loops=1)
|       -> Nested loop inner join (cost=1908.13 rows=5444) (actual time=0.326..0.332 rows=15 loops=1)
|         -> Nested loop inner join (cost=2.67 rows=5) (actual time=0.027..0.027 rows=1 loops=1)
|           -> Covering index lookup on F using PRIMARY (Username='User0') (cost=0.92 rows=5) (actual time=0.017..0.017 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)
|         -> Filter: ((C.weapon_code is not null) and (C.premis_code is not null)) (cost=293.99 rows=1089) (actual time=0.299..0.304 rows=15 loops=1)
|           -> Index lookup on C using crime_code (crime_code=F.crime_code) (cost=293.99 rows=1089) (actual time=0.297..0.301 rows=15 loops=1)
|           -> Single-row index lookup on WT using PRIMARY (weapon_code=C.weapon_code) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1)
|         -> 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=1)
|
|-----+
```

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);

```
-----+
| -> 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=3321.84 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=1)
|         -> 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=1)
|
|-----+
```

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.

CREATE INDEX crimetype\_idx ON CrimeType (crime\_code);

```
-----+
| -> Limit: 15 row(s) (cost=4921.43 rows=15) (actual time=0.257..0.279 rows=15 loops=1)
|   -> Nested loop inner join (cost=4921.43 rows=4685) (actual time=0.256..0.278 rows=15 loops=1)
|     -> Nested loop inner join (cost=3321.84 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=1 loops=1)
|           -> Single-row index lookup on CT using PRIMARY (crime_code=F.crime_code) (cost=0.27 rows=1) (actual time=0.008..0.008 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=0.225..0.228 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=1)
|         -> 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=1)
|
|-----+
```

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=15 loops=1)
|   -> Covering index skip scan for deduplication on C using premis_code (cost=92.65 rows=198) (actual time=0.029..0.086 rows=15 loops=1)
| -> 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 time=3.954..3.954 rows=1 loops=15)
|       -> Table scan on <temporary> (actual time=3.945..3.948 rows=31 loops=15)
|         -> Aggregate using temporary table (actual time=3.944..3.944 rows=31 loops=15)
|           -> Index lookup on Crimes using premis_code (premis_code=C.premis_code) (cost=214.73 rows=614) (actual time=0.073..3.341 rows=3208 loops=15)
|
+-----+
```

This is query2 with no indexes.

CREATE INDEX crime\_premis\_idx ON Crimes (premis\_code);

```
-----+
| -> Limit: 15 row(s) (cost=211.90 rows=15) (actual time=0.703..0.857 rows=15 loops=1)
|   -> Covering index skip scan for deduplication on C using crime_premis_idx (cost=211.90 rows=263) (actual time=0.701..0.851 rows=15 loops=1)
| -> Select #2 (subquery in projection; dependent)
|   -> Limit: 1 row(s) (actual time=5.321..5.321 rows=1 loops=15)
|     -> Sort: count(0) DESC, limit input to 1 row(s) per chunk (actual time=5.320..5.320 rows=1 loops=15)
|       -> Table scan on <temporary> (actual time=5.304..5.310 rows=31 loops=15)
|         -> Aggregate using temporary table (actual time=5.303..5.303 rows=31 loops=15)
|           -> Index lookup on Crimes using crime_premis_idx (premis_code=C.premis_code) (cost=161.46 rows=461) (actual time=0.088..4.687 rows=3208 loops=15)
|
+-----+
```

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);

```
-----+
| -> Limit: 15 row(s) (cost=211.90 rows=15) (actual time=0.028..0.093 rows=15 loops=1)
|   -> Covering index skip scan for deduplication on C using crime_premis_idx (cost=211.90 rows=263) (actual time=0.027..0.089 rows=15 loops=1)
| -> Select #2 (subquery in projection; dependent)
|   -> Limit: 1 row(s) (actual time=3.854..3.854 rows=1 loops=15)
|     -> Sort: count(0) DESC, limit input to 1 row(s) per chunk (actual time=3.854..3.854 rows=1 loops=15)
|       -> Table scan on <temporary> (actual time=3.844..3.847 rows=31 loops=15)
|         -> Aggregate using temporary table (actual time=3.844..3.844 rows=31 loops=15)
|           -> Index lookup on Crimes using crime_premis_idx (premis_code=C.premis_code) (cost=161.46 rows=461) (actual time=0.073..3.240 rows=3208 loops=15)
|
+-----+
```

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);

```
-----+
| -> Limit: 15 row(s) (cost=211.90 rows=15) (actual time=0.029..0.086 rows=15 loops=1)
|   -> Covering index skip scan for deduplication on C using crime_premis_idx (cost=211.90 rows=263) (actual time=0.028..0.082 rows=15 loops=1)
| -> Select #2 (subquery in projection; dependent)
|   -> Limit: 1 row(s) (actual time=3.875..3.875 rows=1 loops=15)
|     -> Sort: count(0) DESC, limit input to 1 row(s) per chunk (actual time=3.875..3.875 rows=1 loops=15)
|       -> Table scan on <temporary> (actual time=3.866..3.869 rows=31 loops=15)
|         -> Aggregate using temporary table (actual time=3.866..3.866 rows=31 loops=15)
|           -> Index lookup on Crimes using crime_premis_idx (premis_code=C.premis_code) (cost=161.46 rows=461) (actual time=0.069..3.266 rows=3208 loops=15)
|
+-----+
```

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.





