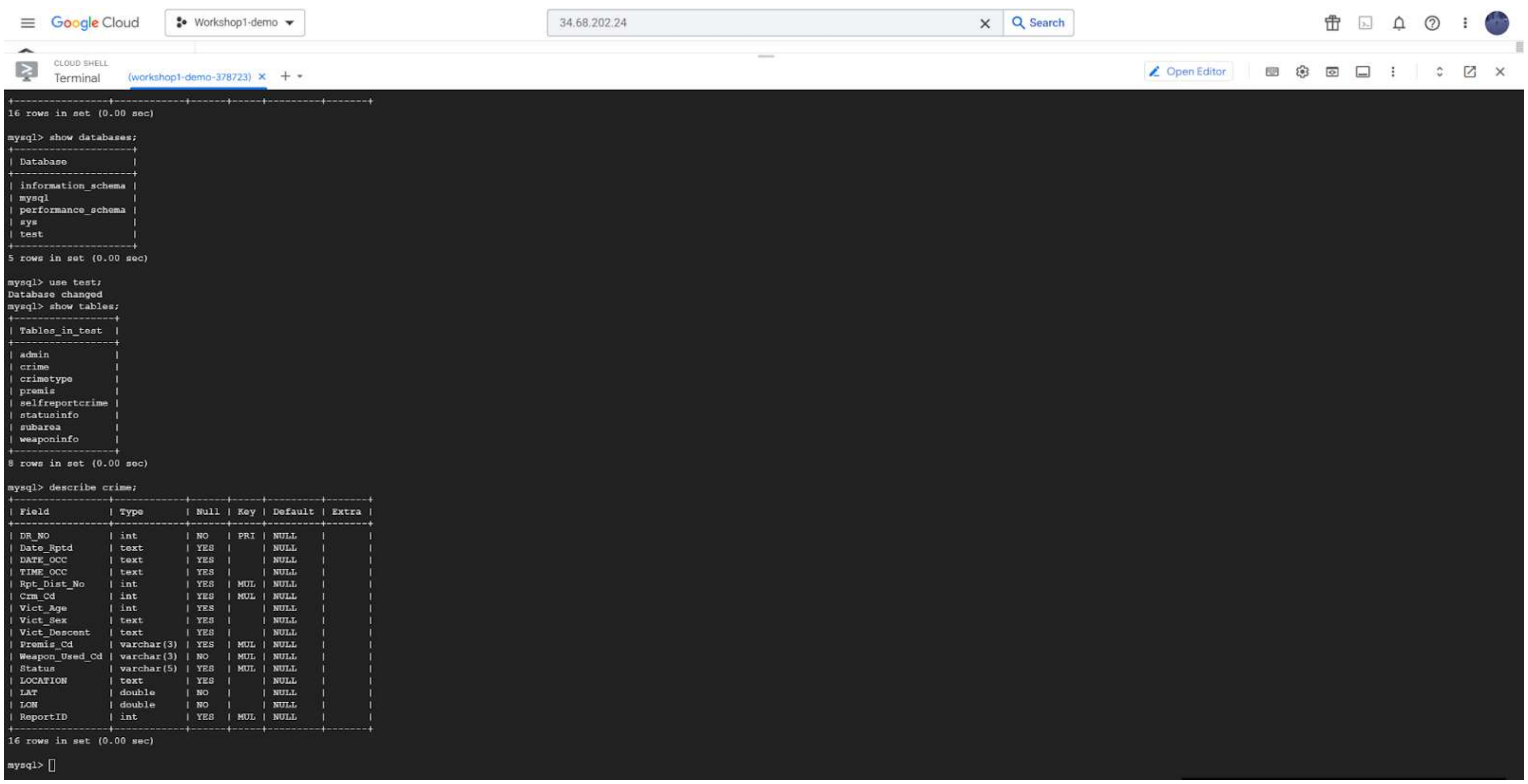
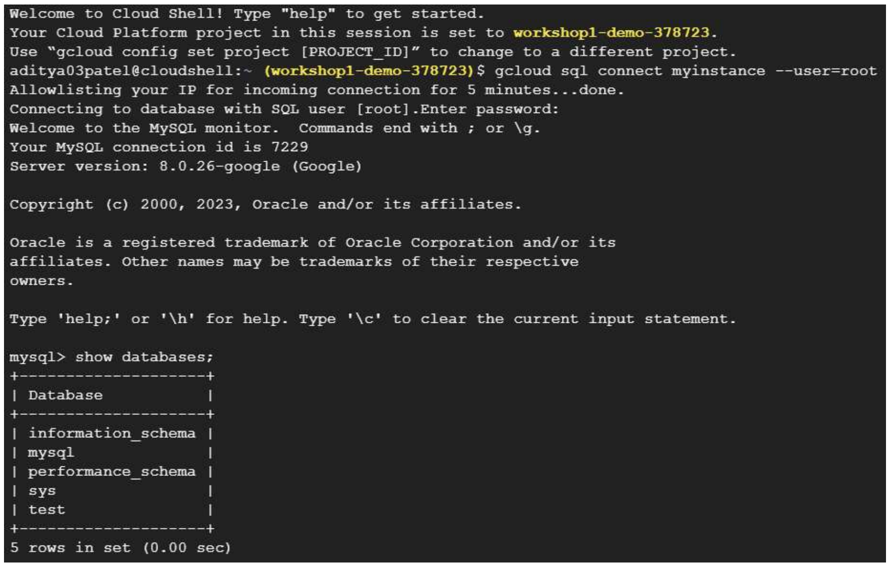
Connection To GCP Using Terminal



```
+----------------+-------------+------+-----+---------+-------+
16 rows in set (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+
5 rows in set (0.00 sec)

mysql> use test;
Database changed
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| admin          |
| crime          |
| crimetype      |
| premis         |
| selfreportcrime |
| statusinfo     |
| subarea        |
| weaponinfo     |
+----------------+
8 rows in set (0.00 sec)

mysql> describe crime;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| DR_NO          | int         | NO   | PRI | NULL    |       |
| Date_Rptd      | text        | YES  |     | NULL    |       |
| DATE_OCC       | text        | YES  |     | NULL    |       |
| TIME_OCC       | text        | YES  |     | NULL    |       |
| Rpt_Dist_No    | int         | YES  | MUL | NULL    |       |
| Crm_Cd         | int         | YES  | MUL | NULL    |       |
| Vict_Age       | int         | YES  |     | NULL    |       |
| Vict_Sex       | text        | YES  |     | NULL    |       |
| Vict_Descent   | text        | YES  |     | NULL    |       |
| Premis_Cd      | varchar(3)  | YES  | MUL | NULL    |       |
| Weapon_Used_Cd | varchar(3)  | NO   | MUL | NULL    |       |
| Status         | varchar(5)  | YES  | MUL | NULL    |       |
| LOCATION       | text        | YES  |     | NULL    |       |
| LAT            | double      | NO   |     | NULL    |       |
| LON            | double      | NO   |     | NULL    |       |
| ReportID       | int         | YES  | MUL | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
16 rows in set (0.00 sec)

mysql> 
```

This is a picture of the terminal on my GCP where I show all the databases alongside the tables present in the test database. I then go into the crime table to show all the columns. This should prove that our database is located on the GCP.



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to workshop1-demo-378723.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
aditya03patel@cloudshell:~ (workshop1-demo-378723)$ gcloud sql connect myinstance --user=root
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7229
Server version: 8.0.26-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

DDL Commands For Creating The Table

```
CREATE TABLE `crime` (
  `DR_NO` int NOT NULL,
  `Date_Rptd` text,
  `DATE_OCC` text,
  `TIME_OCC` text,
  `Rpt_Dist_No` int DEFAULT NULL,
  `Crm_Cd` int DEFAULT NULL,
  `Vict_Age` int DEFAULT NULL,
  `Vict_Sex` text,
  `Vict_Descent` text,
  `Premis_Cd` varchar(3) DEFAULT NULL,
  `Weapon_Used_Cd` varchar(3) NOT NULL,
  `Status` varchar(5) DEFAULT NULL,
  `LOCATION` text,
  `LAT` double NOT NULL,
  `LON` double NOT NULL,
  `ReportID` int DEFAULT NULL,
  PRIMARY KEY (`DR_NO`),
  KEY `crime_ibfk_5_idx` (`Weapon_Used_Cd`),
  KEY `ReportID_idx` (`ReportID`),
  KEY `At` (`Rpt_Dist_No`),
  KEY `Premis_Is` (`Premis_Cd`),
  KEY `Commit` (`Crm_Cd`),
  KEY `Status_Is` (`Status`),
  KEY `Weapon_Used_Cd_idx` (`Weapon_Used_Cd`),
  KEY `Weapon_Used_Cd1_idx` (`Weapon_Used_Cd`(2)),
  CONSTRAINT `At` FOREIGN KEY (`Rpt_Dist_No`) REFERENCES `subarea` (`Rpt_Dist_No`)
ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Commit` FOREIGN KEY (`Crm_Cd`) REFERENCES `crimetype` (`Crm_Cd`)
ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `In` FOREIGN KEY (`ReportID`) REFERENCES `selfreportcrime` (`ReportID`),
  CONSTRAINT `Premis_Is` FOREIGN KEY (`Premis_Cd`) REFERENCES `premis`
(`Premis_Cd`)
ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Status_Is` FOREIGN KEY (`Status`) REFERENCES `statusinfo` (`Status`)
ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Using` FOREIGN KEY (`Weapon_Used_Cd`) REFERENCES `weaponinfo`
(`Weapon_Used_Cd`)
);
```

```
CREATE TABLE `crimetype` (
  `Crm_Cd` int NOT NULL,
  `Crm_Cd_Desc` text,
  PRIMARY KEY (`Crm_Cd`)
);

CREATE TABLE `premis` (
  `Premis_Cd` varchar(3) NOT NULL,
  `Premis_Desc` text,
  PRIMARY KEY (`Premis_Cd`)
);

CREATE TABLE `selfreportcrime` (
  `ReportID` int NOT NULL,
  `Date_Rptd` text,
  `DATE_OCC` text,
  `TIME_OCC` text,
  `Vict_Age` int DEFAULT NULL,
  `Vict_Sex` text,
  `Vict_Descent` text,
  `LOCATION` text,
  `LAT` double DEFAULT NULL,
  `LON` double DEFAULT NULL,
  `Username` varchar(6) DEFAULT NULL,
  PRIMARY KEY (`ReportID`),
  KEY `Username_idx` (`Username`),
  CONSTRAINT `Username` FOREIGN KEY (`Username`) REFERENCES `admin`
(`Username`)
);

CREATE TABLE `statusinfo` (
  `Status` varchar(5) NOT NULL,
  `Status_Desc` text,
  PRIMARY KEY (`Status`)
);

CREATE TABLE `subarea` (
  `AREA` text NOT NULL,
  `AREA_NAME` text NOT NULL,
  `Rpt_Dist_No` int NOT NULL,
  PRIMARY KEY (`Rpt_Dist_No`)
);
```

```
CREATE TABLE `weaponinfo` (
  `Weapon_Used_Cd` varchar(3) NOT NULL,
  `Weapon_Desc` text,
  PRIMARY KEY (`Weapon_Used_Cd`)
);

CREATE TABLE `admin` (
  `Username` varchar(6) NOT NULL,
  `Password` text,
  PRIMARY KEY (`Username`)
);
```

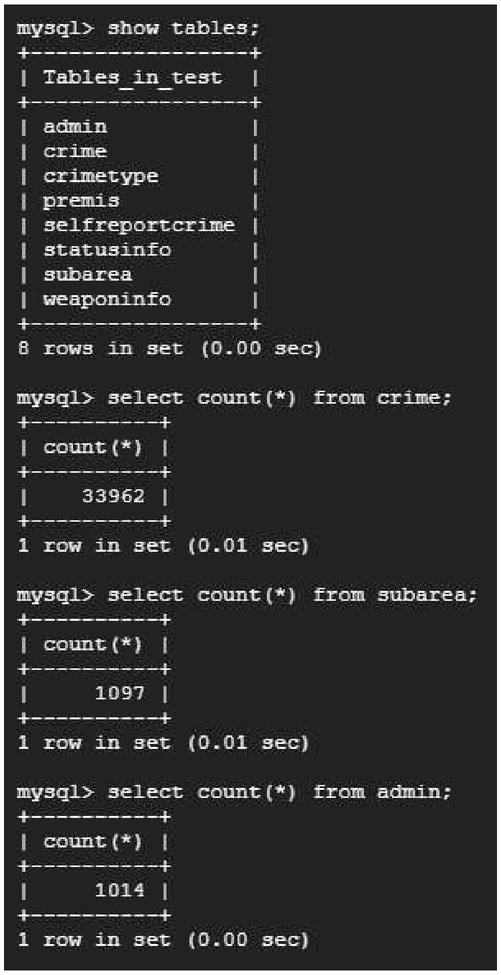
Entry Insertion Into Tables

The three tables we decided to have at least 3000 entries for were the tables Admin, Crime, and Subarea.

Admin stores the user login information of the people who have access to edit the database. This includes insertion to the crime database from the selfreportcrime.

Crime stores all the important information regarding the incident alongside a unique key for each incident.

Subarea stores all the different sub areas in the city of LA which are a part of a major area i.e. multiple sub areas are consisted within an area.



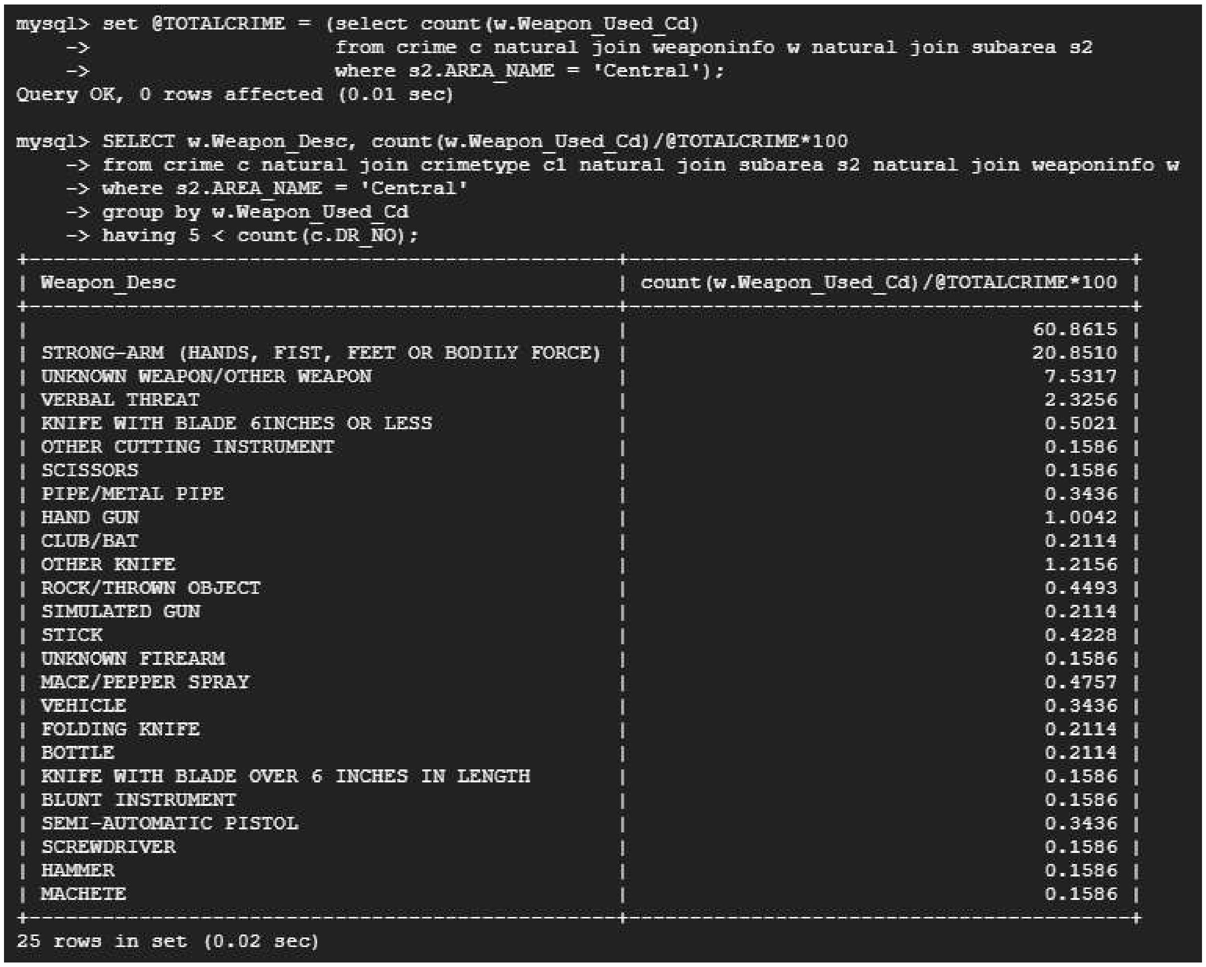
```
mysql> show tables;
+-----------------+
| Tables_in_test  |
+-----------------+
| admin           |
| crime           |
| crimetype       |
| premis          |
| selfreportcrime |
| statusinfo      |
| subarea         |
| weaponinfo      |
+-----------------+
8 rows in set (0.00 sec)

mysql> select count(*) from crime;
+----------+
| count(*) |
+----------+
|    33962 |
+----------+
1 row in set (0.01 sec)

mysql> select count(*) from subarea;
+----------+
| count(*) |
+----------+
|     1097 |
+----------+
1 row in set (0.01 sec)

mysql> select count(*) from admin;
+----------+
| count(*) |
+----------+
|     1014 |
+----------+
1 row in set (0.00 sec)
```

Advanced Queries

Query 1

```
set @TOTALCRIME = (select count(w.Weapon_Used_Cd)
                from crime c natural join weaponinfo w natural join subarea s2
                where s2.AREA_NAME = 'Central');

SELECT w.Weapon_Desc, count(w.Weapon_Used_Cd)/@TOTALCRIME*100
from crime c natural join crimetype c1 natural join subarea s2 natural join weaponinfo w
where s2.AREA_NAME = 'Central'
group by w.Weapon_Used_Cd
having 5 < count(c.DR_NO);
```

Output



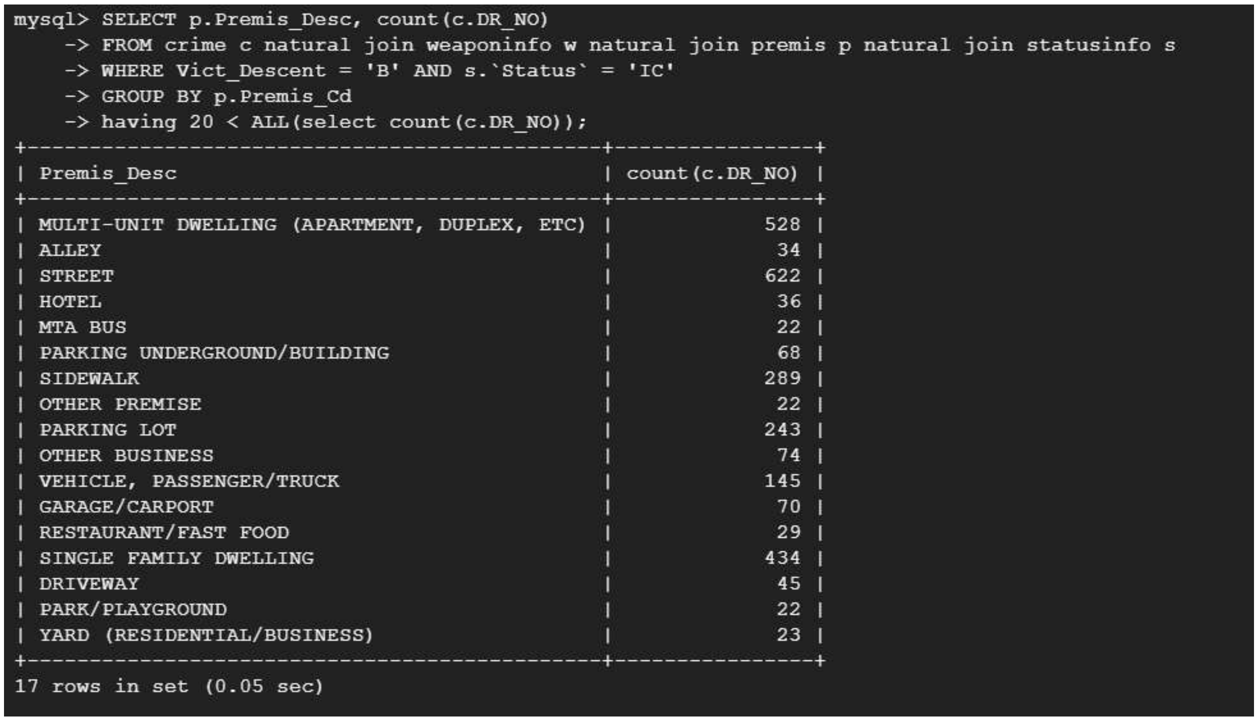
```
mysql> set @TOTALCRIME = (select count(w.Weapon_Used_Cd)
    ->                    from crime c natural join weaponinfo w natural join subarea s2
    ->                    where s2.AREA_NAME = 'Central');
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT w.Weapon_Desc, count(w.Weapon_Used_Cd)/@TOTALCRIME*100
    -> from crime c natural join crimetype c1 natural join subarea s2 natural join weaponinfo w
    -> where s2.AREA_NAME = 'Central'
    -> group by w.Weapon_Used_Cd
    -> having 5 < count(c.DR_NO);
+------------------------------------------------+----------------------------------------+
| Weapon_Desc                                    | count(w.Weapon_Used_Cd)/@TOTALCRIME*100 |
+------------------------------------------------+----------------------------------------+
|                                                |                                60.8615 |
| STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE) |                                20.8510 |
| UNKNOWN WEAPON/OTHER WEAPON                    |                                 7.5317 |
| VERBAL THREAT                                  |                                 2.3256 |
| KNIFE WITH BLADE 6INCHES OR LESS               |                                 0.5021 |
| OTHER CUTTING INSTRUMENT                       |                                 0.1586 |
| SCISSORS                                       |                                 0.1586 |
| PIPE/METAL PIPE                                |                                 0.3436 |
| HAND GUN                                       |                                 1.0042 |
| CLUB/BAT                                       |                                 0.2114 |
| OTHER KNIFE                                    |                                 1.2156 |
| ROCK/THROWN OBJECT                             |                                 0.4493 |
| SIMULATED GUN                                  |                                 0.2114 |
| STICK                                          |                                 0.4228 |
| UNKNOWN FIREARM                                |                                 0.1586 |
| MACE/PEPPER SPRAY                              |                                 0.4757 |
| VEHICLE                                        |                                 0.3436 |
| FOLDING KNIFE                                  |                                 0.2114 |
| BOTTLE                                         |                                 0.2114 |
| KNIFE WITH BLADE OVER 6 INCHES IN LENGTH       |                                 0.1586 |
| BLUNT INSTRUMENT                               |                                 0.1586 |
| SEMI-AUTOMATIC PISTOL                          |                                 0.3436 |
| SCREWDRIVER                                    |                                 0.1586 |
| HAMMER                                         |                                 0.1586 |
| MACHETE                                        |                                 0.1586 |
+------------------------------------------------+----------------------------------------+
25 rows in set (0.02 sec)
```

Basically what this query does is that it calculates the percentage of a certain weapon used for a crime in the central area and group by is done using weapon type. We also check that we have at least 5 crimes committed using that weapon in the central area before constructing the table.

Query 2

SELECT p.Premis_Desc, count(c.DR_NO)
FROM crime c natural join weaponinfo w natural join premis p natural join statusinfo s
WHERE Vict_Descent = 'B' AND s.`Status` = 'IC'
GROUP BY p.Premis_Cd
having 20 < ALL(select count(c.DR_NO))

Output



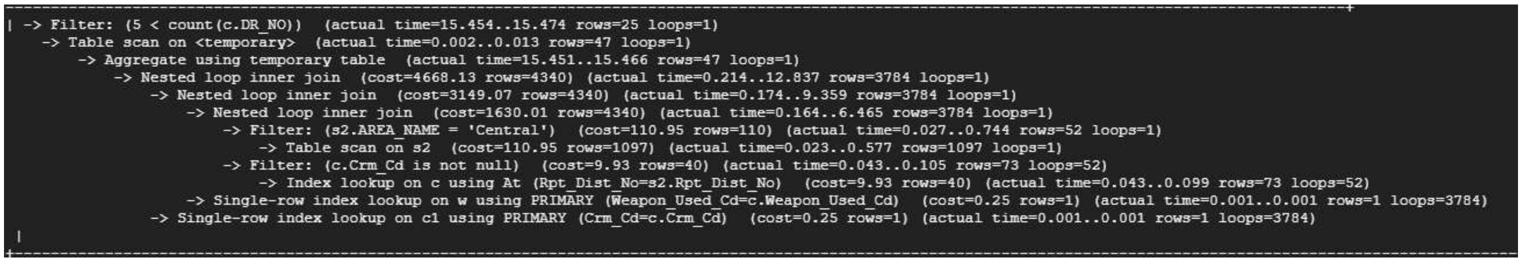
```
mysql> SELECT p.Premis_Desc, count(c.DR_NO)
    -> FROM crime c natural join weaponinfo w natural join premis p natural join statusinfo s
    -> WHERE Vict_Descent = 'B' AND s.`Status` = 'IC'
    -> GROUP BY p.Premis_Cd
    -> having 20 < ALL(select count(c.DR_NO));
+----------------------------------------------+----------------+
| Premis_Desc                                  | count(c.DR_NO) |
+----------------------------------------------+----------------+
| MULTI-UNIT DWELLING (APARTMENT, DUPLEX, ETC) |            528 |
| ALLEY                                        |             34 |
| STREET                                       |            622 |
| HOTEL                                        |             36 |
| MTA BUS                                      |             22 |
| PARKING UNDERGROUND/BUILDING                 |             68 |
| SIDEWALK                                     |            289 |
| OTHER PREMISE                                |             22 |
| PARKING LOT                                  |            243 |
| OTHER BUSINESS                               |             74 |
| VEHICLE, PASSENGER/TRUCK                     |            145 |
| GARAGE/CARPORT                               |             70 |
| RESTAURANT/FAST FOOD                         |             29 |
| SINGLE FAMILY DWELLING                       |            434 |
| DRIVEWAY                                     |             45 |
| PARK/PLAYGROUND                              |             22 |
| YARD (RESIDENTIAL/BUSINESS)                  |             23 |
+----------------------------------------------+----------------+
17 rows in set (0.05 sec)
```

This query basically goes over all the different premises and finds the current number of cases that are still under investigation for crimes committed by people with the decent ‘Black’. We also check and make sure that there are at least 20 crimes in that premis before getting the final table.

Indexing

Query 1

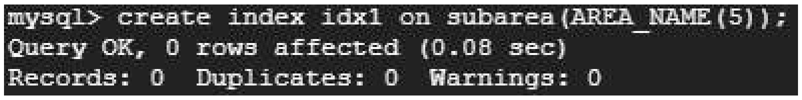
Explain analyze before adding indexes:



```
+-----------------------------------------------------------------------------------------------------------------------+
| -> Filter: (5 < count(c.DR_NO))  (actual time=15.454..15.474 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.013 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=15.451..15.466 rows=47 loops=1)
            -> Nested loop inner join  (cost=4668.13 rows=4340) (actual time=0.214..12.837 rows=3784 loops=1)
                -> Nested loop inner join  (cost=3149.07 rows=4340) (actual time=0.174..9.359 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=1630.01 rows=4340) (actual time=0.164..6.465 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=110.95 rows=110) (actual time=0.027..0.744 rows=52 loops=1)
                            -> Table scan on s2  (cost=110.95 rows=1097) (actual time=0.023..0.577 rows=1097 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.93 rows=40) (actual time=0.043..0.105 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)  (cost=9.93 rows=40) (actual time=0.043..0.099 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
+-----------------------------------------------------------------------------------------------------------------------+
```

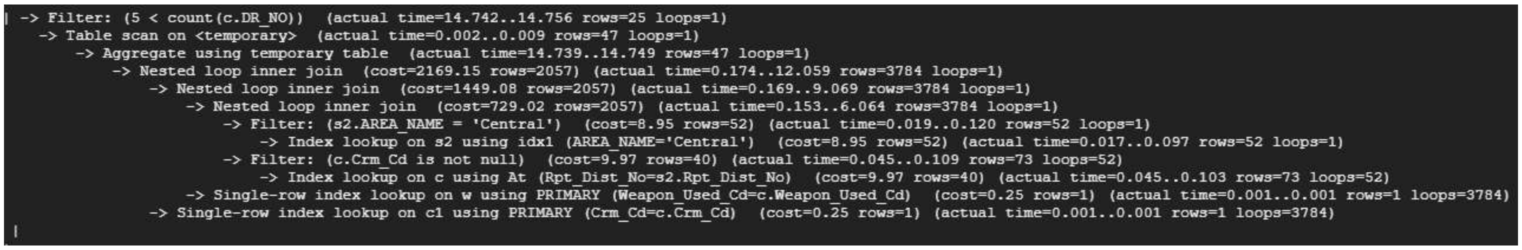
Here, we have multiple costs and times based off of the numerous loops used by the query. This means that there are multiple loops with multiple different costs and runtimes.

Now, I added the index called idx1, which basically truncates the size of the column AREA_NAME, which is located in the subarea entity. The reason I did this is because of the fact that the area we are focusing on is 'Central', so we only need limited characters to define the word, and upon further investigation, I found that only 5 characters from the word are required to successfully be able to run the query as intended.



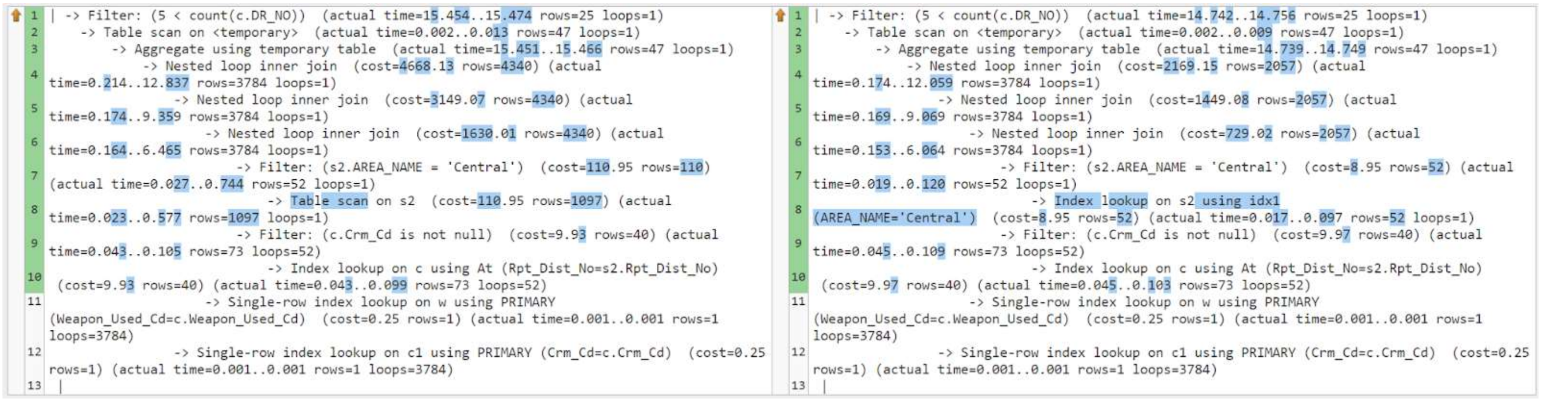
```
mysql> create index idx1 on subarea(AREA_NAME(5));
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

This is the implementation of idx1. After running the query once again with this index, here are the results >



```
| -> Filter: (5 < count(c.DR_NO))  (actual time=14.742..14.756 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.009 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=14.739..14.749 rows=47 loops=1)
            -> Nested loop inner join  (cost=2169.15 rows=2057) (actual time=0.174..12.059 rows=3784 loops=1)
                -> Nested loop inner join  (cost=1449.08 rows=2057) (actual time=0.169..9.069 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=729.02 rows=2057) (actual time=0.153..6.064 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=8.95 rows=52) (actual time=0.019..0.120 rows=52 loops=1)
                            -> Index lookup on s2 using idx1 (AREA_NAME='Central')  (cost=8.95 rows=52) (actual time=0.017..0.097 rows=52 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.97 rows=40) (actual time=0.045..0.109 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)  (cost=9.97 rows=40) (actual time=0.045..0.103 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
```

Right off the bat, I noticed a lot of changes in the runtimes and costs generally going down, but some exceptions were present. Below is the image of a comparison between the two (the left is the original while the right is the one with the added index.

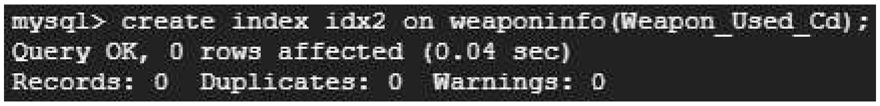


```
| -> Filter: (5 < count(c.DR_NO))  (actual time=15.454..15.474 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.013 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=15.451..15.466 rows=47 loops=1)
            -> Nested loop inner join  (cost=4668.13 rows=4340) (actual
time=0.214..12.837 rows=3784 loops=1)
                -> Nested loop inner join  (cost=3149.07 rows=4340) (actual
time=0.174..9.359 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=1630.01 rows=4340) (actual
time=0.164..6.465 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=110.95 rows=110)
(actual time=0.027..0.744 rows=52 loops=1)
                            -> Table scan on s2  (cost=110.95 rows=1097) (actual
time=0.023..0.577 rows=1097 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.93 rows=40) (actual
time=0.043..0.105 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)
 (cost=9.93 rows=40) (actual time=0.043..0.099 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY
(Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1
loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25
rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
```

```
| -> Filter: (5 < count(c.DR_NO))  (actual time=14.742..14.756 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.009 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=14.739..14.749 rows=47 loops=1)
            -> Nested loop inner join  (cost=2169.15 rows=2057) (actual
time=0.174..12.059 rows=3784 loops=1)
                -> Nested loop inner join  (cost=1449.08 rows=2057) (actual
time=0.169..9.069 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=729.02 rows=2057) (actual
time=0.153..6.064 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=8.95 rows=52) (actual
time=0.019..0.120 rows=52 loops=1)
                            -> Index lookup on s2 using idx1
(AREA_NAME='Central')  (cost=8.95 rows=52) (actual time=0.017..0.097 rows=52 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.97 rows=40) (actual
time=0.045..0.109 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)
 (cost=9.97 rows=40) (actual time=0.045..0.103 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY
(Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1
loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25
rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
```

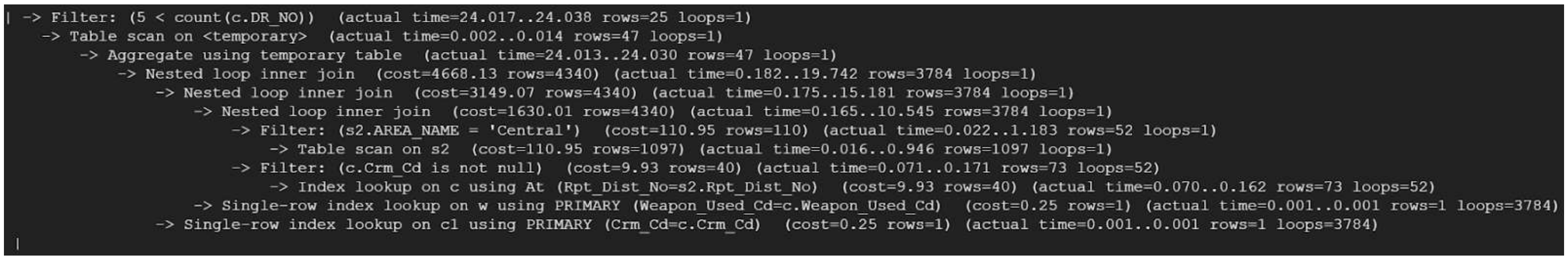
All the costs went down for the query except the one for Rpt_Dist_No, which could be due to the effects of the index on the crime entity. As explained earlier, the reason the other costs wen down was because of the redundant characters in the column in question.

The second indexing I used (idx2) was through the Weapon_Used_Cd column, which is a part of the table weaponinfo. This relates to the output we are trying to get because of the fact that we are measuring the ratios of the weapons used in a certain predefined area.



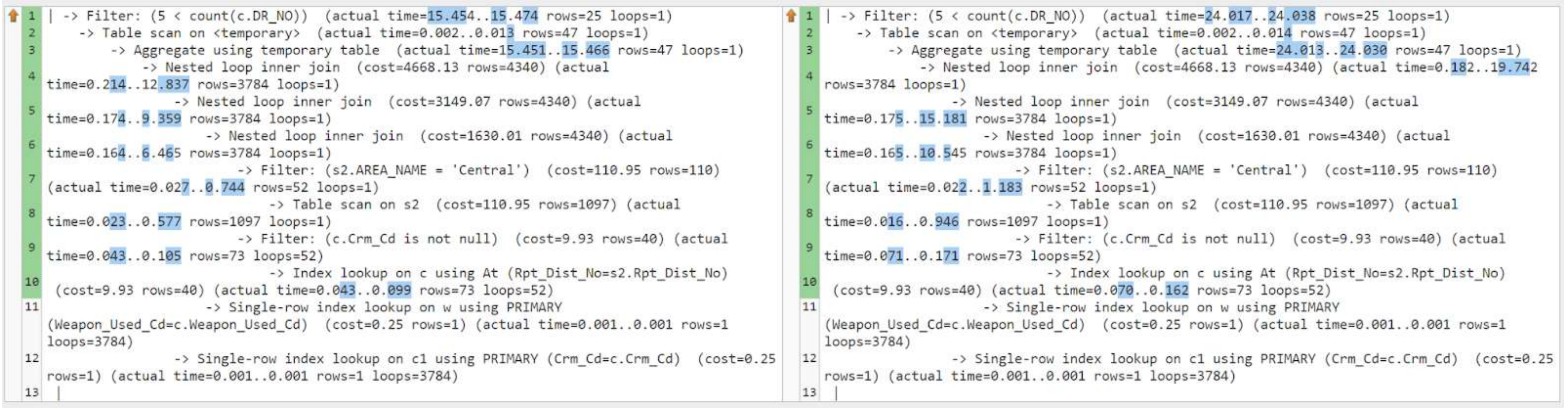
```
mysql> create index idx2 on weaponinfo(Weapon_Used_Cd);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

The explain analysis after applying the index is as follows:



```
| -> Filter: (5 < count(c.DR_NO))  (actual time=24.017..24.038 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.014 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=24.013..24.030 rows=47 loops=1)
            -> Nested loop inner join  (cost=4668.13 rows=4340) (actual time=0.182..19.742 rows=3784 loops=1)
                -> Nested loop inner join  (cost=3149.07 rows=4340) (actual time=0.175..15.181 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=1630.01 rows=4340) (actual time=0.165..10.545 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=110.95 rows=110) (actual time=0.022..1.183 rows=52 loops=1)
                            -> Table scan on s2  (cost=110.95 rows=1097) (actual time=0.016..0.946 rows=1097 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.93 rows=40) (actual time=0.071..0.171 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)  (cost=9.93 rows=40) (actual time=0.070..0.162 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
```

Once again, we can make a direct comparison with the original as shown here as follows:

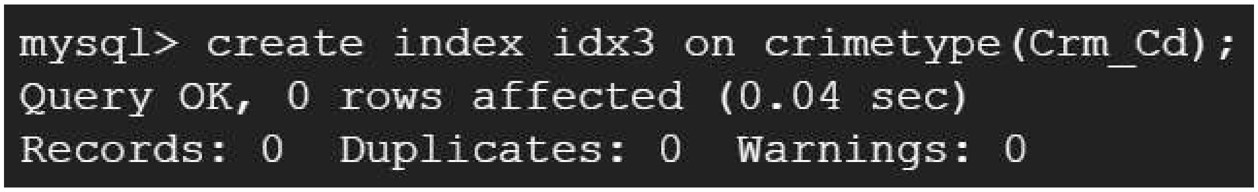


```
| -> Filter: (5 < count(c.DR_NO))  (actual time=15.454..15.474 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.013 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=15.451..15.466 rows=47 loops=1)
            -> Nested loop inner join  (cost=4668.13 rows=4340) (actual time=0.214..12.837 rows=3784 loops=1)
                -> Nested loop inner join  (cost=3149.07 rows=4340) (actual time=0.174..9.359 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=1630.01 rows=4340) (actual time=0.164..6.465 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=110.95 rows=110) (actual time=0.027..0.744 rows=52 loops=1)
                            -> Table scan on s2  (cost=110.95 rows=1097) (actual time=0.023..0.577 rows=1097 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.93 rows=40) (actual time=0.043..0.105 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)  (cost=9.93 rows=40) (actual time=0.043..0.099 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
```

```
| -> Filter: (5 < count(c.DR_NO))  (actual time=24.017..24.038 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.014 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=24.013..24.030 rows=47 loops=1)
            -> Nested loop inner join  (cost=4668.13 rows=4340) (actual time=0.182..19.742 rows=3784 loops=1)
                -> Nested loop inner join  (cost=3149.07 rows=4340) (actual time=0.175..15.181 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=1630.01 rows=4340) (actual time=0.165..10.545 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=110.95 rows=110) (actual time=0.022..1.183 rows=52 loops=1)
                            -> Table scan on s2  (cost=110.95 rows=1097) (actual time=0.016..0.946 rows=1097 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.93 rows=40) (actual time=0.071..0.171 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)  (cost=9.93 rows=40) (actual time=0.070..0.162 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
```

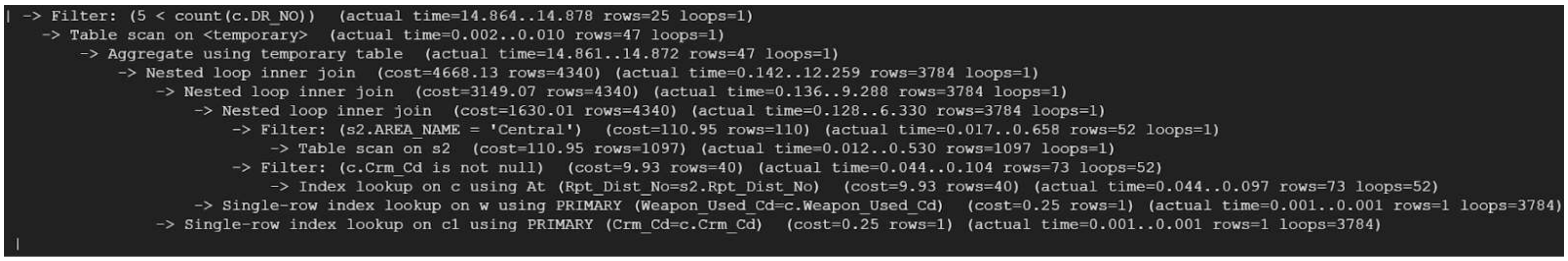
Here, the actual time for everything seems to have gone up, but the interesting observation is the fact that the cost seems to have stayed the exact same throughout the two different analyses. The weapon info is trivial to the query because of the fact that we are getting the count on weapons.

The third indexing I used (idx3) was on the Crm_Cd, which is located in the crimetype entity. This is basically indexing the table that is a part of the joining between multiple tables, which is located in the from part of the query.



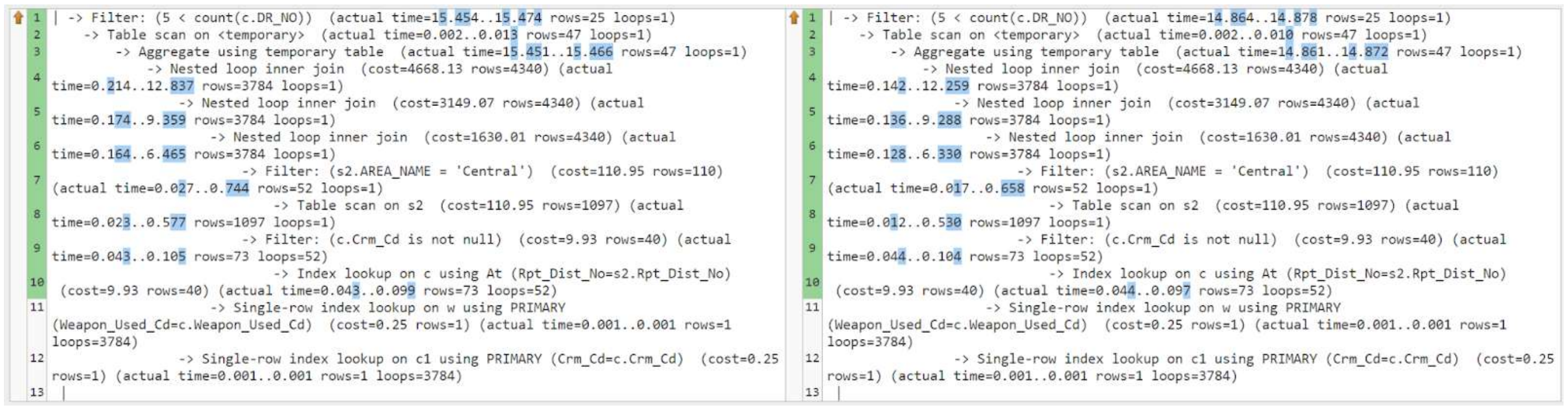
```
mysql> create index idx3 on crimetype(Crm_Cd);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Now, once we run the explain analyze, we get the result as follows >



```
| -> Filter: (5 < count(c.DR_NO))  (actual time=14.864..14.878 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.010 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=14.861..14.872 rows=47 loops=1)
            -> Nested loop inner join  (cost=4668.13 rows=4340) (actual time=0.142..12.259 rows=3784 loops=1)
                -> Nested loop inner join  (cost=3149.07 rows=4340) (actual time=0.136..9.288 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=1630.01 rows=4340) (actual time=0.128..6.330 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=110.95 rows=110) (actual time=0.017..0.658 rows=52 loops=1)
                            -> Table scan on s2  (cost=110.95 rows=1097) (actual time=0.012..0.530 rows=1097 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.93 rows=40) (actual time=0.044..0.104 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)  (cost=9.93 rows=40) (actual time=0.044..0.097 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
```

Now, we can run it through the comparer and see the exact changes in the costs and times >



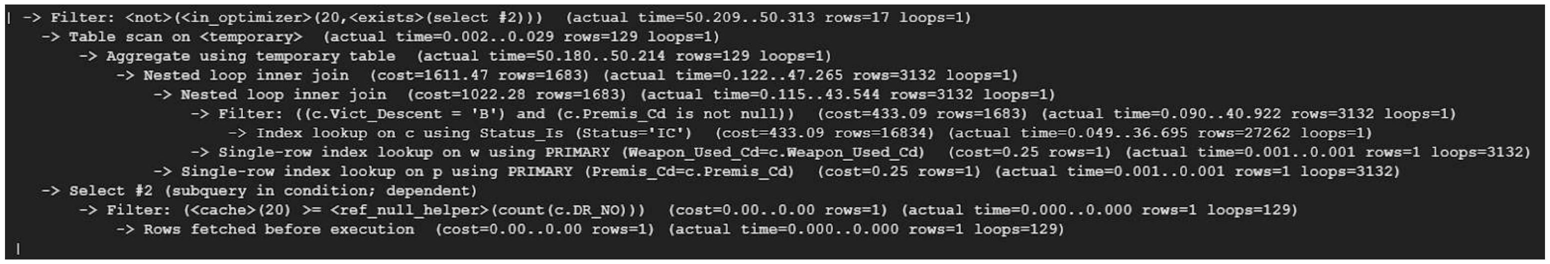
```
| -> Filter: (5 < count(c.DR_NO))  (actual time=15.454..15.474 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.013 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=15.451..15.466 rows=47 loops=1)
            -> Nested loop inner join  (cost=4668.13 rows=4340) (actual
time=0.214..12.837 rows=3784 loops=1)
                -> Nested loop inner join  (cost=3149.07 rows=4340) (actual
time=0.174..9.359 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=1630.01 rows=4340) (actual
time=0.164..6.465 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=110.95 rows=110)
(actual time=0.027..0.744 rows=52 loops=1)
                            -> Table scan on s2  (cost=110.95 rows=1097) (actual
time=0.023..0.577 rows=1097 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.93 rows=40) (actual
time=0.043..0.105 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)
 (cost=9.93 rows=40) (actual time=0.043..0.099 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY
(Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1
loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25
rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
```

```
| -> Filter: (5 < count(c.DR_NO))  (actual time=14.864..14.878 rows=25 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.010 rows=47 loops=1)
        -> Aggregate using temporary table  (actual time=14.861..14.872 rows=47 loops=1)
            -> Nested loop inner join  (cost=4668.13 rows=4340) (actual
time=0.142..12.259 rows=3784 loops=1)
                -> Nested loop inner join  (cost=3149.07 rows=4340) (actual
time=0.136..9.288 rows=3784 loops=1)
                    -> Nested loop inner join  (cost=1630.01 rows=4340) (actual
time=0.128..6.330 rows=3784 loops=1)
                        -> Filter: (s2.AREA_NAME = 'Central')  (cost=110.95 rows=110)
(actual time=0.017..0.658 rows=52 loops=1)
                            -> Table scan on s2  (cost=110.95 rows=1097) (actual
time=0.012..0.530 rows=1097 loops=1)
                        -> Filter: (c.Crm_Cd is not null)  (cost=9.93 rows=40) (actual
time=0.044..0.104 rows=73 loops=52)
                            -> Index lookup on c using At (Rpt_Dist_No=s2.Rpt_Dist_No)
 (cost=9.93 rows=40) (actual time=0.044..0.097 rows=73 loops=52)
                    -> Single-row index lookup on w using PRIMARY
(Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1
loops=3784)
                -> Single-row index lookup on c1 using PRIMARY (Crm_Cd=c.Crm_Cd)  (cost=0.25
rows=1) (actual time=0.001..0.001 rows=1 loops=3784)
 |
```

Now, we can make the observation that the time has gone down in all the loops and table scans, but this didn't end up affecting the actual cost of anything, which is surprising because the only reason the entity is added on the query is because of the fact that it's needed to get the actual accurate count on the number of entities that actually make sense to add based off of the presence of a crime type. In other words, it affected the time on the query a lot more than I expected considering the fact that it doesn't play too trivial of a role here.

Query 2

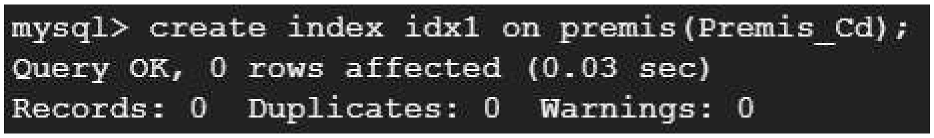
Explain analyze before adding indexes:



```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual time=50.209..50.313 rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.029 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=50.180..50.214 rows=129 loops=1)
            -> Nested loop inner join  (cost=1611.47 rows=1683) (actual time=0.122..47.265 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1022.28 rows=1683) (actual time=0.115..43.544 rows=3132 loops=1)
                    -> Filter: ((c.Vict_Descent = 'B') and (c.Premis_Cd is not null))  (cost=433.09 rows=1683) (actual time=0.090..40.922 rows=3132 loops=1)
                        -> Index lookup on c using Status_Is (Status='IC')  (cost=433.09 rows=16834) (actual time=0.049..36.695 rows=27262 loops=1)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
 |
```

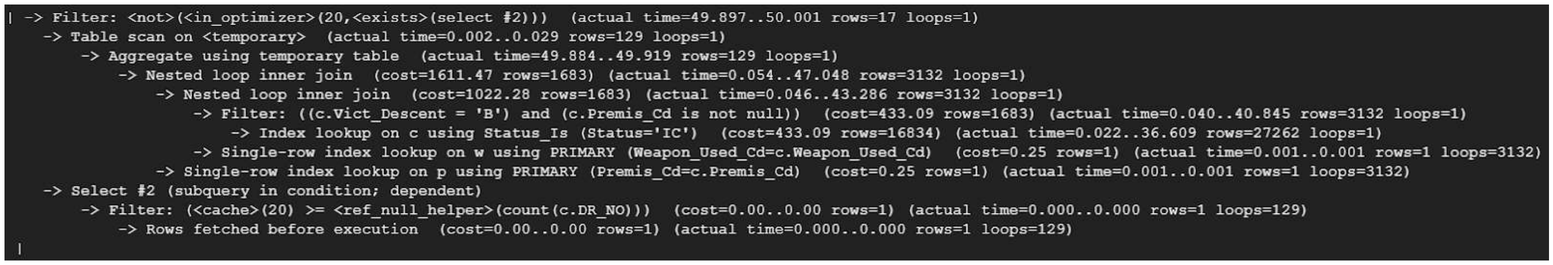
Here, we have multiple costs and times based off of the numerous loops used by the query. This means that there are multiple loops with multiple different costs and runtimes.

Now, I added the index called idx1, which is basically indexing my Premis_Cd, which is located in the Premis entity. Here is the command line for that >



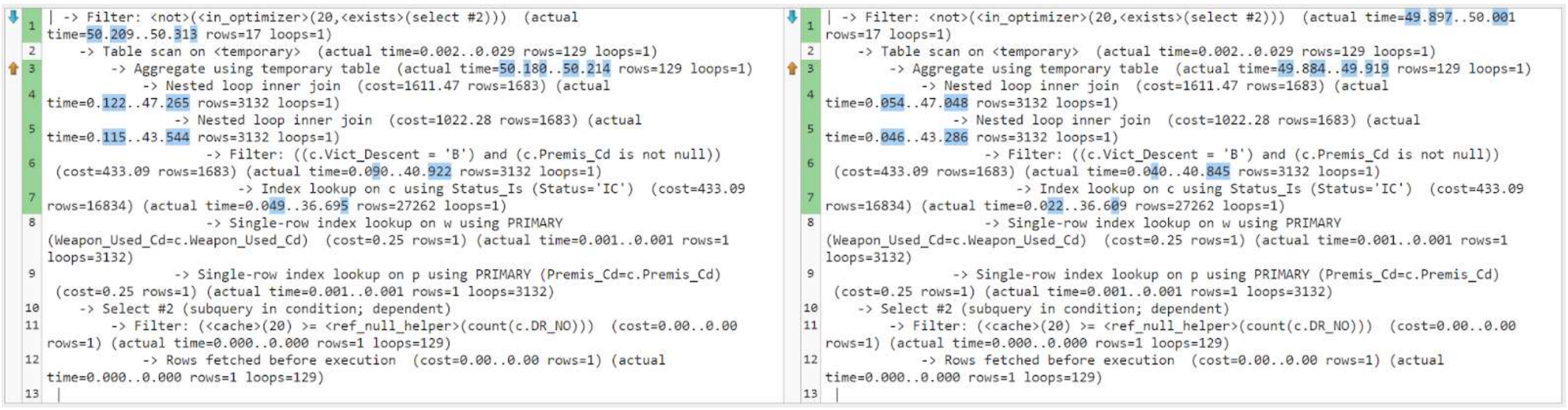
```
mysql> create index idx1 on premis(Premis_Cd);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

After implementing the index, we can now run explain analyze and see how it has changed >



```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual time=49.897..50.001 rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.029 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=49.884..49.919 rows=129 loops=1)
            -> Nested loop inner join  (cost=1611.47 rows=1683) (actual time=0.054..47.048 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1022.28 rows=1683) (actual time=0.046..43.286 rows=3132 loops=1)
                    -> Filter: ((c.Vict_Descent = 'B') and (c.Premis_Cd is not null))  (cost=433.09 rows=1683) (actual time=0.040..40.845 rows=3132 loops=1)
                        -> Index lookup on c using Status_Is (Status='IC')  (cost=433.09 rows=16834) (actual time=0.022..36.609 rows=27262 loops=1)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
 |
```

Now, we can track the exact changes by putting it through a comparer >

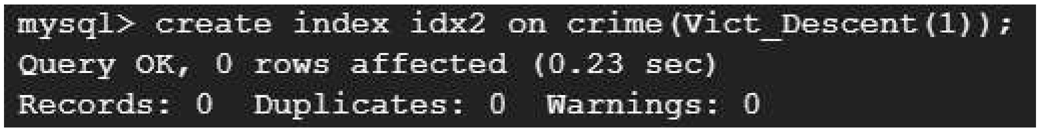


```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual time=50.209..50.313 rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.029 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=50.180..50.214 rows=129 loops=1)
            -> Nested loop inner join  (cost=1611.47 rows=1683) (actual time=0.122..47.265 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1022.28 rows=1683) (actual time=0.115..43.544 rows=3132 loops=1)
                    -> Filter: ((c.Vict_Descent = 'B') and (c.Premis_Cd is not null))  (cost=433.09 rows=1683) (actual time=0.090..40.922 rows=3132 loops=1)
                        -> Index lookup on c using Status_Is (Status='IC')  (cost=433.09 rows=16834) (actual time=0.049..36.695 rows=27262 loops=1)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
 |
```

```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual time=49.897..50.001 rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.029 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=49.884..49.919 rows=129 loops=1)
            -> Nested loop inner join  (cost=1611.47 rows=1683) (actual time=0.054..47.048 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1022.28 rows=1683) (actual time=0.046..43.286 rows=3132 loops=1)
                    -> Filter: ((c.Vict_Descent = 'B') and (c.Premis_Cd is not null))  (cost=433.09 rows=1683) (actual time=0.040..40.845 rows=3132 loops=1)
                        -> Index lookup on c using Status_Is (Status='IC')  (cost=433.09 rows=16834) (actual time=0.022..36.609 rows=27262 loops=1)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
 |
```

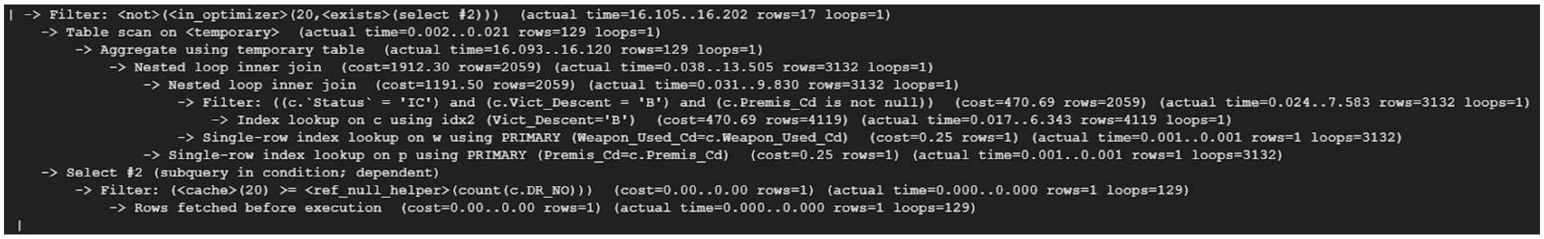
The reason we use Premis_Cd for indexing is that we are using that inside the group by. I would assume that this bring about some sort of the change in the cost or the time of the query, but it really didn't affect it by a lot except for a minor drop in time. Now, I believe that the reason this could be the case is because of the fact that it might not be as significant to the output. This is a bit contradictory to the last query because of the fact that weapon used gave us a bit of a larger difference.

For the second index, we used the column Vict_Descent, which is located in the crime entity. The command line for that is >



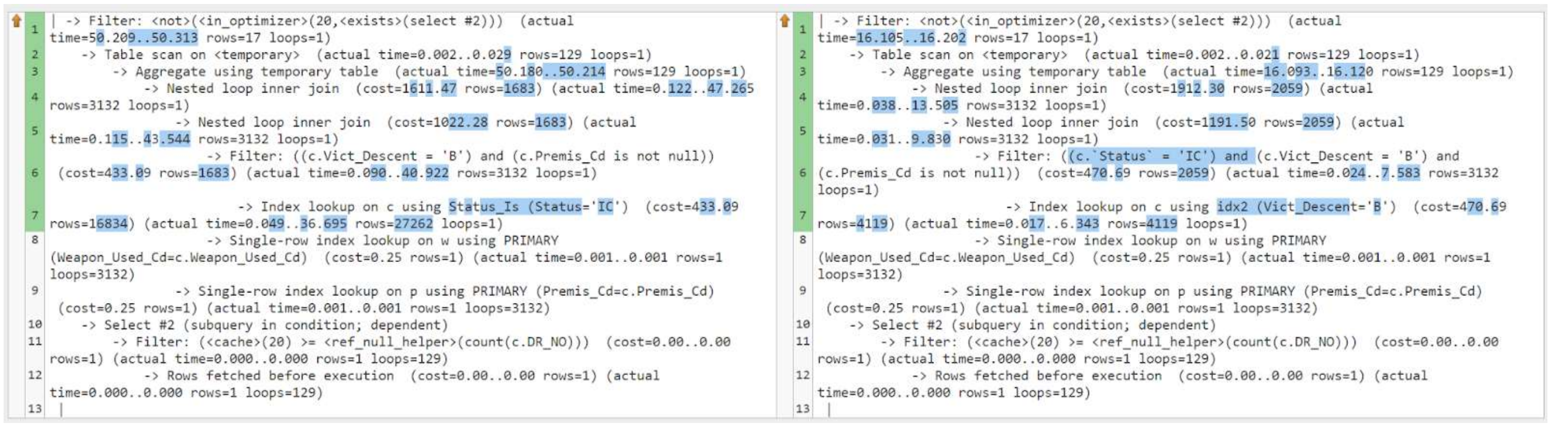
```
mysql> create index idx2 on crime(Vict_Descent(1));
Query OK, 0 rows affected (0.23 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

After creating the index, here is the output of explain analyze >



```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual time=16.105..16.202 rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.021 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=16.093..16.120 rows=129 loops=1)
            -> Nested loop inner join  (cost=1912.30 rows=2059) (actual time=0.038..13.505 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1191.50 rows=2059) (actual time=0.031..9.830 rows=3132 loops=1)
                    -> Filter: ((c.`Status` = 'IC') and (c.Vict_Descent = 'B') and (c.Premis_Cd is not null))  (cost=470.69 rows=2059) (actual time=0.024..7.583 rows=3132 loops=1)
                        -> Index lookup on c using idx2 (Vict_Descent='B')  (cost=470.69 rows=4119) (actual time=0.017..6.343 rows=4119 loops=1)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
 |
```

Now, we can run it through the comparator to spot the differences >

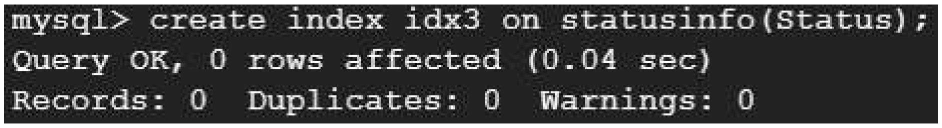


```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual time=50.209..50.313 rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.029 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=50.180..50.214 rows=129 loops=1)
            -> Nested loop inner join  (cost=1611.47 rows=1683) (actual time=0.122..47.265 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1022.28 rows=1683) (actual time=0.115..43.544 rows=3132 loops=1)
                    -> Filter: ((c.Vict_Descent = 'B') and (c.Premis_Cd is not null))  (cost=433.09 rows=1683) (actual time=0.090..40.922 rows=3132 loops=1)
                        -> Index lookup on c using Status_Is (Status='IC')  (cost=433.09 rows=16834) (actual time=0.049..36.695 rows=27262 loops=1)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
 |
```

```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual time=16.105..16.202 rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.021 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=16.093..16.120 rows=129 loops=1)
            -> Nested loop inner join  (cost=1912.30 rows=2059) (actual time=0.038..13.505 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1191.50 rows=2059) (actual time=0.031..9.830 rows=3132 loops=1)
                    -> Filter: ((c.`Status` = 'IC') and (c.Vict_Descent = 'B') and (c.Premis_Cd is not null))  (cost=470.69 rows=2059) (actual time=0.024..7.583 rows=3132 loops=1)
                        -> Index lookup on c using idx2 (Vict_Descent='B')  (cost=470.69 rows=4119) (actual time=0.017..6.343 rows=4119 loops=1)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
 |
```

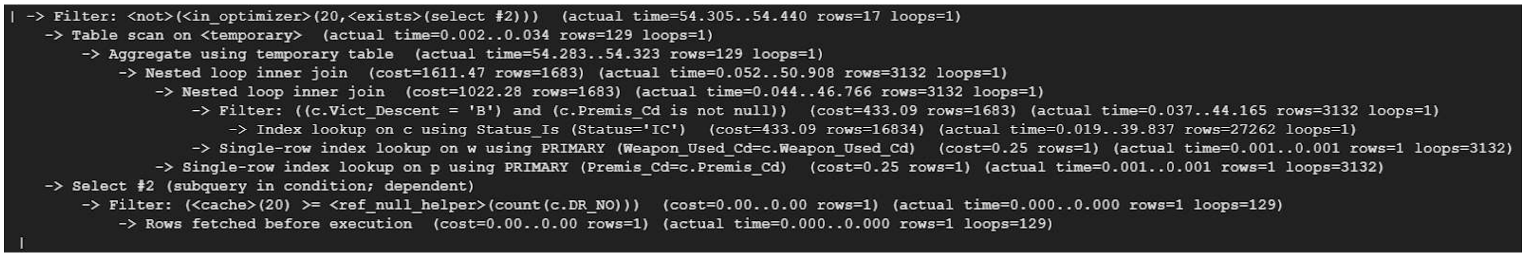
The reason we picked victim descent as our index was because of the fact that it was a text and we could easily truncate it to still be usable. Now when we look at the exact changes that happened in the explain analyze, we can see that we actually have a higher cost instead of a lower one. This is certainly unexpected because this should have made it easier to lookup. The reason this could be the case is because it could be inefficient.

For the third index, we used the Status column from the statusinfo entity. The reason we did this is because of the fact that is it used in the where statement to validify the status of the investigation. The command line for it is as follows >



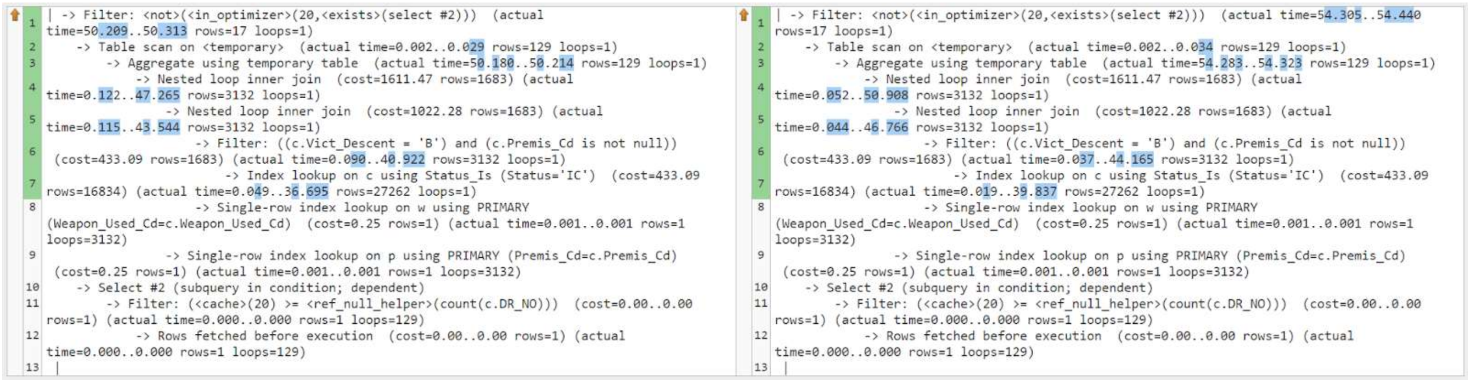
```
mysql> create index idx3 on statusinfo(Status);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

After creating the index, here is the output of explain analyze >



```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual time=54.305..54.440 rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.034 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=54.283..54.323 rows=129 loops=1)
            -> Nested loop inner join  (cost=1611.47 rows=1683) (actual time=0.052..50.908 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1022.28 rows=1683) (actual time=0.044..46.766 rows=3132 loops=1)
                    -> Filter: ((c.Vict_Descent = 'B') and (c.Premis_Cd is not null))  (cost=433.09 rows=1683) (actual time=0.037..44.165 rows=3132 loops=1)
                        -> Index lookup on c using Status_Is (Status='IC')  (cost=433.09 rows=16834) (actual time=0.019..39.837 rows=27262 loops=1)
                    -> Single-row index lookup on w using PRIMARY (Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual time=0.000..0.000 rows=1 loops=129)
 |
```

Now we can put it through a comparator >



```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual
time=50.209..50.313 rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.029 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=50.180..50.214 rows=129 loops=1)
            -> Nested loop inner join  (cost=1611.47 rows=1683) (actual
time=0.122..47.265 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1022.28 rows=1683) (actual
time=0.115..43.544 rows=3132 loops=1)
                    -> Filter: ((c.Vict_Descent = 'B') and (c.Premis_Cd is not null))
 (cost=433.09 rows=1683) (actual time=0.090..40.922 rows=3132 loops=1)
                        -> Index lookup on c using Status_Is (Status='IC')  (cost=433.09
rows=16834) (actual time=0.049..36.695 rows=27262 loops=1)
                    -> Single-row index lookup on w using PRIMARY
(Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1
loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)
 (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00
rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual
time=0.000..0.000 rows=1 loops=129)
 |
```

```
| -> Filter: <not>(<in_optimizer>(20,<exists>(select #2)))  (actual time=54.305..54.440
rows=17 loops=1)
    -> Table scan on <temporary>  (actual time=0.002..0.034 rows=129 loops=1)
        -> Aggregate using temporary table  (actual time=54.283..54.323 rows=129 loops=1)
            -> Nested loop inner join  (cost=1611.47 rows=1683) (actual
time=0.052..50.908 rows=3132 loops=1)
                -> Nested loop inner join  (cost=1022.28 rows=1683) (actual
time=0.044..46.766 rows=3132 loops=1)
                    -> Filter: ((c.Vict_Descent = 'B') and (c.Premis_Cd is not null))
 (cost=433.09 rows=1683) (actual time=0.037..44.165 rows=3132 loops=1)
                        -> Index lookup on c using Status_Is (Status='IC')  (cost=433.09
rows=16834) (actual time=0.019..39.837 rows=27262 loops=1)
                    -> Single-row index lookup on w using PRIMARY
(Weapon_Used_Cd=c.Weapon_Used_Cd)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1
loops=3132)
                -> Single-row index lookup on p using PRIMARY (Premis_Cd=c.Premis_Cd)
 (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=3132)
    -> Select #2 (subquery in condition; dependent)
        -> Filter: (<cache>(20) >= <ref_null_helper>(count(c.DR_NO)))  (cost=0.00..0.00
rows=1) (actual time=0.000..0.000 rows=1 loops=129)
            -> Rows fetched before execution  (cost=0.00..0.00 rows=1) (actual
time=0.000..0.000 rows=1 loops=129)
 |
```

This is surprising because of the fact that we are using this as an argument inside the where statement. This could be because of the fact that it could have a lot of duplicates.