Table Row Count

```
mysql> SELECT table_name, table_rows FROM information_schema.tables LIMIT 5;
+------------+------------+
| TABLE_NAME | TABLE_ROWS |
+------------+------------+
| Cases      |     750625 |
| CrimeCodes |        138 |
| Location   |     808963 |
| Victim     |     749749 |
| Weapons    |         79 |
+------------+------------+
5 rows in set (0.01 sec)
```

DDL Commands for Tables

```
CREATE TABLE CrimeCodes(
        Crime_Code INT NOT NULL PRIMARY KEY,
        Crime_Code_Desc VARCHAR(255) NOT NULL
);

CREATE TABLE Weapons(
        Weapon_Code INT NOT NULL PRIMARY KEY,
        Weapon_Desc VARCHAR(255) NOT NULL
);

CREATE TABLE Cases(
Record_Number INT NOT NULL PRIMARY KEY,
Weapon_Code INT,
Date_Occurred VARCHAR(22),
Time_Occurred VARCHAR(4),
Crime_Code INT,
Status_Desc VARCHAR(255),
FOREIGN KEY (Weapon_Code) REFERENCES Weapons(Weapon_Code)
        ON DELETE SET NULL
        ON UPDATE CASCADE,
FOREIGN KEY (Crime_Code) REFERENCES CrimeCodes(Crime_Code)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```

```
CREATE TABLE Victim(
        Record_Number INT NOT NULL PRIMARY KEY,
        Victim_Age INT,
        Victim_Sex VARCHAR(1),
        Descent_Code VARCHAR(1),
        FOREIGN KEY (Record_Number) REFERENCES Cases(Record_Number)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Location(
        Record_Number INT NOT NULL PRIMARY KEY,
        Area_Name VARCHAR(255),
        Reported_District INT,
        Longitude DEC,
        Latitude DEC,
        FOREIGN KEY (Record_Number) REFERENCES Cases(Record_Number)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Advanced Queries + Indexing

Advanced Query 1: Return the number of victims between a given age range and count the gender of victims in that age range for any given crime.

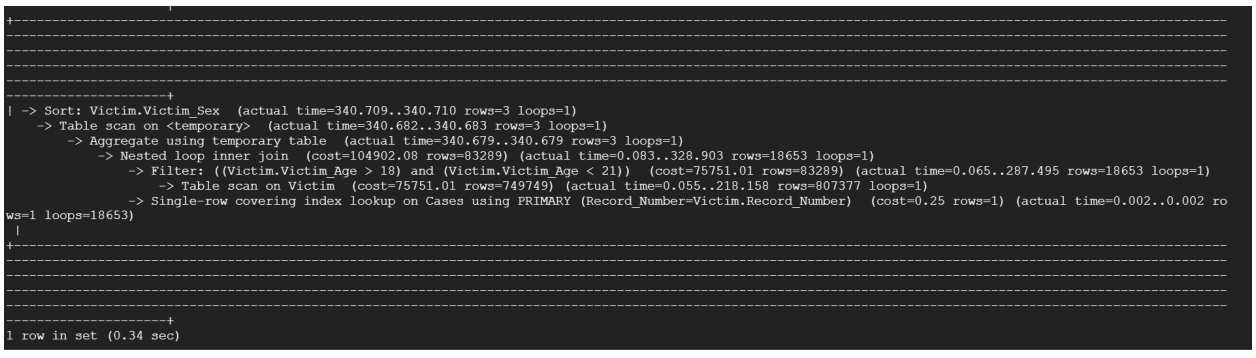
Query:

```
SELECT COUNT(*), Victim_Sex
FROM Cases NATURAL JOIN Victim
WHERE Victim_Age > 18 AND Victim_Age < 21
GROUP BY Victim_Sex
ORDER BY Victim_Sex
```

```
mysql> SELECT COUNT(*), Victim_Sex
    -> FROM Cases NATURAL JOIN Victim
    -> WHERE Victim_Age > 18 AND Victim_age < 21
    -> GROUP BY Victim_Sex
    -> ORDER BY Victim_Sex;
+----------+------------+
| COUNT(*) | Victim_Sex |
+----------+------------+
|     8806 | F          |
|     7497 | M          |
|     2350 | X          |
+----------+------------+
3 rows in set (0.28 sec)
```

NOTE: ONLY returns 3 rows since there are 3 genders in our dataset, X being undefined/unknown

Initial Query Explain Analyze



```
| -> Sort: Victim.Victim_Sex  (actual time=340.709..340.710 rows=3 loops=1)
    -> Table scan on <temporary>  (actual time=340.682..340.683 rows=3 loops=1)
        -> Aggregate using temporary table  (actual time=340.679..340.679 rows=3 loops=1)
            -> Nested loop inner join  (cost=104902.08 rows=83289) (actual time=0.083..328.903 rows=18653 loops=1)
                -> Filter: ((Victim.Victim_Age > 18) and (Victim.Victim_Age < 21))  (cost=75751.01 rows=83289) (actual time=0.065..287.495 rows=18653 loops=1)
                    -> Table scan on Victim  (cost=75751.01 rows=749749) (actual time=0.055..218.158 rows=807377 loops=1)
                -> Single-row covering index lookup on Cases using PRIMARY (Record_Number=Victim.Record_Number)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=18653)
 |
1 row in set (0.34 sec)
```

Index 1) CREATE INDEX age_index ON Victim(Victim_Age)

Explain Analyze After Index 1

```
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
| -> Sort: Victim.Victim_Sex  (actual time=179.906..179.907 rows=3 loops=1)
    -> Table scan on <temporary>  (actual time=179.882..179.882 rows=3 loops=1)
        -> Aggregate using temporary table  (actual time=179.879..179.879 rows=3 loops=1)
            -> Nested loop inner join  (cost=28879.72 rows=34594) (actual time=0.069..167.962 rows=18653 loops=1)
                -> Index range scan on Victim using age_index over (18 < Victim_Age < 21), with index condition: ((Victim.Victim_Age > 18) and (Victim.Victim_Ag
e < 21))  (cost=16771.82 rows=34594) (actual time=0.054..126.443 rows=18653 loops=1)
                -> Single-row covering index lookup on Cases using PRIMARY (Record_Number=Victim.Record_Number)  (cost=0.25 rows=1) (actual time=0.002..0.002 ro
ws=1 loops=18653)
 |
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------+
1 row in set (0.29 sec)
```

Explanation: We created an index on Victim_Age since we were filtering in the WHERE condition based on age before GROUPING BY. It makes sense that our time decreases by 0.05 seconds since our query makes use of filtering through age. Instead of a table scan, something like a B-Tree scan will be more usable and optimized.

Index 2) CREATE INDEX sex_index ON Victim_Sex

Explain Analyze After Index 2

```
+----------------------------------------------------------------------------------------------------
--------------------+
| -> Sort: Victim.Victim_Sex  (actual time=357.960..357.961 rows=3 loops=1)
    -> Table scan on <temporary>  (actual time=357.934..357.935 rows=3 loops=1)
        -> Aggregate using temporary table  (actual time=357.932..357.932 rows=3 loops=1)
            -> Nested loop inner join  (cost=104902.08 rows=83289) (actual time=0.086..345.097 rows=18653 loops=1)
                -> Filter: ((Victim.Victim_Age > 18) and (Victim.Victim_Age < 21))  (cost=75751.01 rows=83289) (actual time=0.067..303.068 rows=18653 loops=1)
                    -> Table scan on Victim  (cost=75751.01 rows=749749) (actual time=0.058..230.634 rows=807377 loops=1)
                -> Single-row covering index lookup on Cases using PRIMARY (Record_Number=Victim.Record_Number)  (cost=0.25 rows=1) (actual time=0.002..0.002 ro
ws=1 loops=18653)
 |
+----------------------------------------------------------------------------------------------------
--------------------+
1 row in set (0.36 sec)
```

Explanation: We created an index on Victim_Sex since we were GROUPING BY the sex of the victim. The time actually increased which makes sense since there are only three genders in our dataset and indexing them might be over-complicating in comparison to a table scan.

Explain Analyze After Index 3

Index 3)

CREATE INDEX age_index ON Victim(Victim_Age)

CREATE INDEX sex_index ON Victim_Sex

```
| -> Sort: Victim.Victim_Sex  (actual time=134.527..134.527 rows=3 loops=1)
    -> Table scan on <temporary>  (actual time=134.502..134.502 rows=3 loops=1)
        -> Aggregate using temporary table  (actual time=134.498..134.498 rows=3 loops=1)
            -> Nested loop inner join  (cost=28879.72 rows=34594) (actual time=0.072..122.444 rows=18653 loops=1)
                -> Index range scan on Victim using age_index over (18 < Victim_Age < 21), with index condition: ((Victim.Victim_Age > 18) and (Victim.Victim_Age < 21))  (cost=16771.82 rows=34594) (actual time=0.051..82.500 rows=18653 loops=1)
                -> Single-row covering index lookup on Cases using PRIMARY (Record_Number=Victim.Record_Number)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=18653)
 |
1 row in set (0.17 sec)
```

Explanation: We used both indexes and the time also significantly dropped. This makes sense since the sex index likely speeds up the group speed and the age index makes searching for the ages faster. In our implementation it will make sense to use both as an index.

Advanced Query 2: Return a query with the crimes in descending order by the count of victims greater than or equal to 18. Limited to the top 15 crimes.

Query:

```
SELECT Crime_Code, Crime_Code_Desc, COUNT(Victim.Record_Number)
FROM Victim JOIN Cases on Victim.Record_Number = Cases.Record_Number NATURAL JOIN CrimeCodes
WHERE Victim_Age >= 18
GROUP BY Crime_Code
ORDER BY COUNT(Victim.Record_Number) DESC
LIMIT 15;
```

```
mysql> SELECT Crime_Code, Crime_Code_Desc, COUNT(Victim.Record_Number) FROM Victim JOIN Cases on Victim.Record_Number = Cases.Record_Number NATURAL JOIN CrimeCodes WHERE Victim_Age >= 18 GROUP BY Crime_Code ORDER BY COUNT(Victim.Record_Number) DESC LIMIT 15;
+------------+-----------------------------------------------------+-----------------------------+
| Crime_Code | Crime_Code_Desc                                     | COUNT(Victim.Record_Number) |
+------------+-----------------------------------------------------+-----------------------------+
|        624 | BATTERY - SIMPLE ASSAULT                            |                       60122 |
|        354 | THEFT OF IDENTITY                                   |                       50236 |
|        330 | BURGLARY FROM VEHICLE                               |                       48033 |
|        230 | ASSAULT WITH DEADLY WEAPON, AGGRAVATED ASSAULT      |                       42691 |
|        626 | INTIMATE PARTNER - SIMPLE ASSAULT                   |                       39870 |
|        740 | VANDALISM - FELONY ($400 & OVER, ALL CHURCH VANDALISMS) |                   37080 |
|        440 | THEFT PLAIN - PETTY ($950 & UNDER)                  |                       35382 |
|        310 | BURGLARY                                            |                       33604 |
|        331 | THEFT FROM MOTOR VEHICLE - GRAND ($950.01 AND OVER) |                       27890 |
|        210 | ROBBERY                                             |                       21117 |
|        341 | THEFT-GRAND ($950.01 & OVER)EXCPT,GUNS,FOWL,LIVESTK,PROD |                   20927 |
|        745 | VANDALISM - MISDEAMEANOR ($399 OR UNDER)            |                       17177 |
|        930 | CRIMINAL THREATS - NO WEAPON DISPLAYED              |                       15471 |
|        420 | THEFT FROM MOTOR VEHICLE - PETTY ($950 & UNDER)     |                       13860 |
|        761 | BRANDISH WEAPON                                     |                       11802 |
+------------+-----------------------------------------------------+-----------------------------+
15 rows in set (2.90 sec)
```

Initial Query Explain Analyze

```
---------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=3423.643..3423.645 rows=15 loops=1)
    -> Sort: COUNT(Victim.Record_Number) DESC, limit input to 15 row(s) per chunk  (actual time=3423.641..3423.642 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=3423.562..3423.601 rows=134 loops=1)
            -> Aggregate using temporary table  (actual time=3423.554..3423.554 rows=134 loops=1)
                -> Nested loop inner join  (cost=250674.94 rows=249891) (actual time=0.132..2075.461 rows=587337 loops=1)
                    -> Nested loop inner join  (cost=163212.98 rows=249891) (actual time=0.124..1468.305 rows=587337 loops=1)
                        -> Filter: (Victim.Victim_Age >= 18)  (cost=75751.01 rows=249891) (actual time=0.107..321.734 rows=587337 loops=1)
                            -> Table scan on Victim  (cost=75751.01 rows=749749) (actual time=0.102..235.404 rows=807377 loops=1)
                        -> Filter: (Cases.Crime_Code is not null)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=587337)
                            -> Single-row index lookup on Cases using PRIMARY (Record_Number=Victim.Record_Number)  (cost=0.25 rows=1) (actual time=0.002..0.002
 rows=1 loops=587337)
                    -> Single-row index lookup on CrimeCodes using PRIMARY (Crime_Code=Cases.Crime_Code)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 lo
ops=587337)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------------------+
1 row in set (3.43 sec)
```

Index 1) CREATE INDEX on age_index on Victim(Victim_Age);

Explain Analyze After Index 1

```
----------+
| -> Limit: 15 row(s)  (actual time=4383.044..4383.046 rows=15 loops=1)
    -> Sort: COUNT(Victim.Record_Number) DESC, limit input to 15 row(s) per chunk  (actual time=4383.043..4383.044 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=4382.963..4383.003 rows=134 loops=1)
            -> Aggregate using temporary table  (actual time=4382.954..4382.954 rows=134 loops=1)
                -> Nested loop inner join  (cost=337799.11 rows=374874) (actual time=0.061..3009.824 rows=587337 loops=1)
                    -> Nested loop inner join  (cost=206593.21 rows=374874) (actual time=0.053..2335.388 rows=587337 loops=1)
                        -> Filter: (Victim.Victim_Age >= 18)  (cost=75387.31 rows=374874) (actual time=0.037..912.574 rows=587337 loops=1)
                            -> Covering index range scan on Victim using age_index over (18 <= Victim_Age)  (cost=75387.31 rows=374874) (actual time=0.035..839.
570 rows=587337 loops=1)
                        -> Filter: (Cases.Crime_Code is not null)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=587337)
                            -> Single-row index lookup on Cases using PRIMARY (Record_Number=Victim.Record_Number)  (cost=0.25 rows=1) (actual time=0.002..0.002
 rows=1 loops=587337)
                    -> Single-row index lookup on CrimeCodes using PRIMARY (Crime_Code=Cases.Crime_Code)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 lo
ops=587337)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------+
1 row in set (4.52 sec)
```

Explanation: Here, adding the index on the age of victims worsened the time performance of the query. Some reasons we can think of is that the indexed attribute is only part of the WHERE clause and the massive size of the query to scan through the entirety of a joined table of multiple relations.

Index 2) CREATE INDEX record_number_idx on Victim(Record_Number);

Explain Analyze After Index 2

```
----------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=3319.093..3319.095 rows=15 loops=1)
    -> Sort: COUNT(Victim.Record_Number) DESC, limit input to 15 row(s) per chunk  (actual time=3319.092..3319.093 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=3319.013..3319.053 rows=134 loops=1)
            -> Aggregate using temporary table  (actual time=3319.007..3319.007 rows=134 loops=1)
                -> Nested loop inner join  (cost=250674.94 rows=249891) (actual time=0.087..1993.327 rows=587337 loops=1)
                    -> Nested loop inner join  (cost=163212.98 rows=249891) (actual time=0.080..1391.300 rows=587337 loops=1)
                        -> Filter: (Victim.Victim_Age >= 18)  (cost=75751.01 rows=249891) (actual time=0.065..310.735 rows=587337 loops=1)
                            -> Table scan on Victim  (cost=75751.01 rows=749749) (actual time=0.061..229.470 rows=807377 loops=1)
                        -> Filter: (Cases.Crime_Code is not null)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=587337)
                            -> Single-row index lookup on Cases using PRIMARY (Record_Number=Victim.Record_Number)  (cost=0.25 rows=1) (actual time=0.001..0.001
 rows=1 loops=587337)
                    -> Single-row index lookup on CrimeCodes using PRIMARY (Crime_Code=Cases.Crime_Code)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 lo
ops=587337)
 |
+-------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------+
1 row in set (3.32 sec)
```

Explanation: Here, adding an index on the record number improved performance but not by a significant amount. This might be because record number is used as a common attribute for a join operation.

Index 3) CREATE INDEX crime_code_idx on Cases(Crime_Code);

Explain Analyze After Index 3

```
----------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (actual time=3423.643..3423.645 rows=15 loops=1)
    -> Sort: COUNT(Victim.Record_Number) DESC, limit input to 15 row(s) per chunk  (actual time=3423.641..3423.642 rows=15 loops=1)
        -> Table scan on <temporary>  (actual time=3423.562..3423.601 rows=134 loops=1)
            -> Aggregate using temporary table  (actual time=3423.554..3423.554 rows=134 loops=1)
                -> Nested loop inner join  (cost=250674.94 rows=249891) (actual time=0.132..2075.461 rows=587337 loops=1)
                    -> Nested loop inner join  (cost=163212.98 rows=249891) (actual time=0.124..1468.305 rows=587337 loops=1)
                        -> Filter: (Victim.Victim_Age >= 18)  (cost=75751.01 rows=249891) (actual time=0.107..321.734 rows=587337 loops=1)
                            -> Table scan on Victim  (cost=75751.01 rows=749749) (actual time=0.102..235.404 rows=807377 loops=1)
                        -> Filter: (Cases.Crime_Code is not null)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=587337)
                            -> Single-row index lookup on Cases using PRIMARY (Record_Number=Victim.Record_Number)  (cost=0.25 rows=1) (actual time=0.002..0.002
 rows=1 loops=587337)
                    -> Single-row index lookup on CrimeCodes using PRIMARY (Crime_Code=Cases.Crime_Code)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 lo
ops=587337)
 |
+-------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------+
1 row in set (3.43 sec)
```

Explanation: Here, adding an index on the crime code did not change the performance by a noticeable amount. Despite it being part of the GROUP BY clause, it is an index on a foreign key referencing a primary key, so that might be the reason why it did not improve performance.