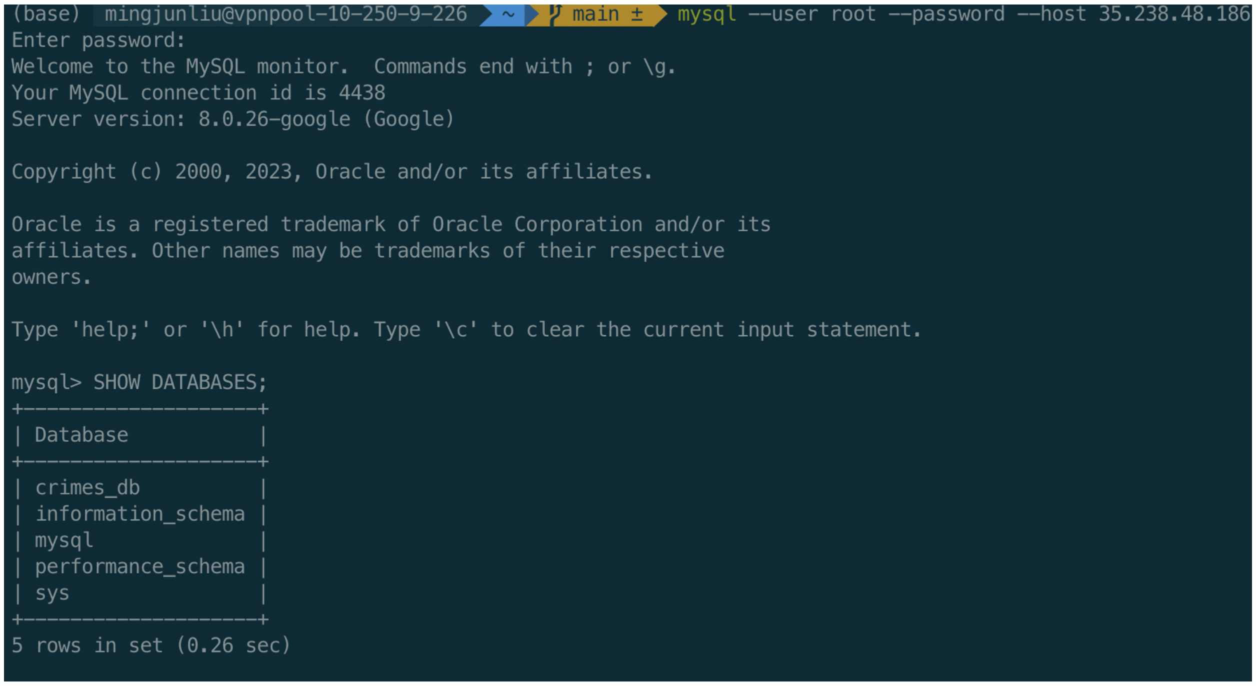
CS411 Stage 3

Implemented the database tables on GCP

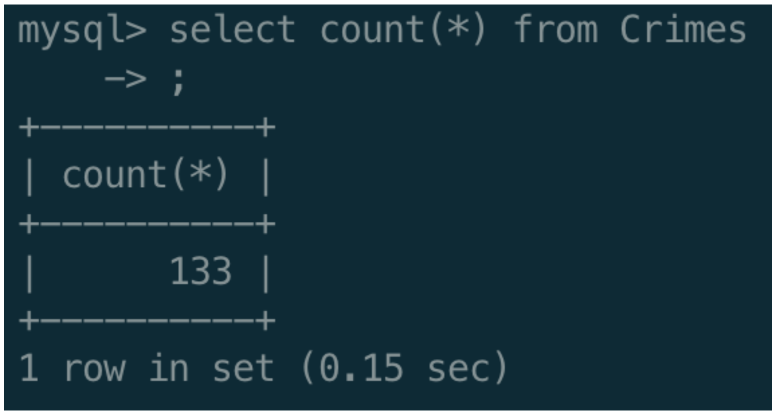


```
(base)  mingjunliu@vpnpool-10-250-9-226  ~  ᚴ main ±  mysql --user root --password --host 35.238.48.186
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4438
Server version: 8.0.26-google (Google)

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           |
+--------------------+
| crimes_db          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.26 sec)
```

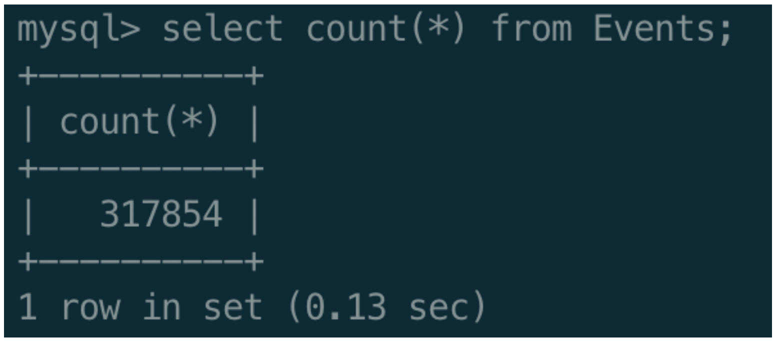
Provided the DDL commands for your tables

```
CREATE TABLE Locations (
AreaName VARCHAR(50) NOT NULL,
StreetName VARCHAR(50) NOT NULL,
Latitude FLOAT(10, 6) NOT NULL,
Longtitude FLOAT(10, 6) NOT NULL,
PRIMARY KEY (StreetName, AreaName)
);

CREATE TABLE Victims (
EventID INT NOT NULL,
VictimID VARCHAR(255) NOT NULL,
Age INT,
Sex ENUM('M', 'F'),
PRIMARY KEY (EventID, VictimID),
FOREIGN KEY (EventID) REFERENCES Events(EventID)
);

CREATE TABLE Crimes (
CrimeType INT NOT NULL,
CrimeDesc VARCHAR(100) NOT NULL,
Mocodes INT NOT NULL,
Risk FLOAT(8, 2) NOT NULL,
PRIMARY KEY (CrimeType)
);

CREATE TABLE Events (
EventID INT NOT NULL,
DateReported DATE NOT NULL,
DateOccur DATE NOT NULL,
AreaName VARCHAR(255) NOT NULL,
StreetName VARCHAR(255) NOT NULL,
CrimeType INT NOT NULL,
PRIMARY KEY (EventID),
FOREIGN KEY (StreetName, AreaName) REFERENCES Locations(StreetName, AreaName)
);
```

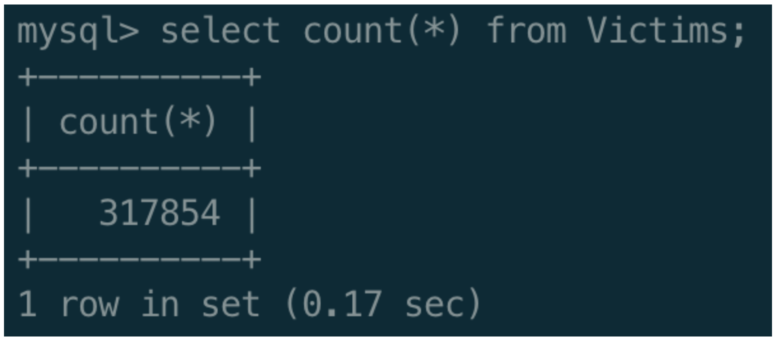
Inserted at least 1000 rows in the tables



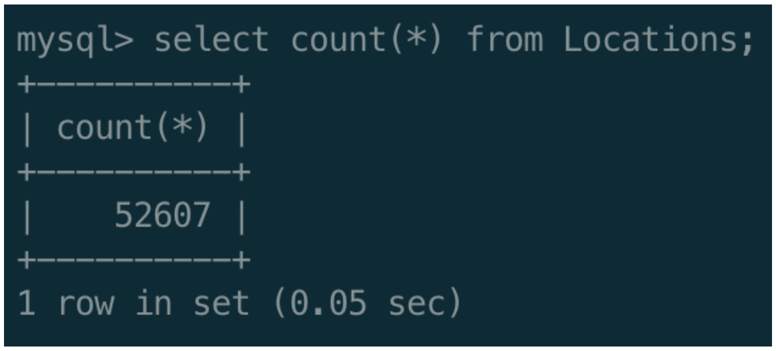
```
mysql> select count(*) from Crimes
    -> ;
+----------+
| count(*) |
+----------+
|      133 |
+----------+
1 row in set (0.15 sec)
```



```
mysql> select count(*) from Events;
+----------+
| count(*) |
+----------+
|   317854 |
+----------+
1 row in set (0.13 sec)
```



```
mysql> select count(*) from Victims;
+----------+
| count(*) |
+----------+
|   317854 |
+----------+
1 row in set (0.17 sec)
```



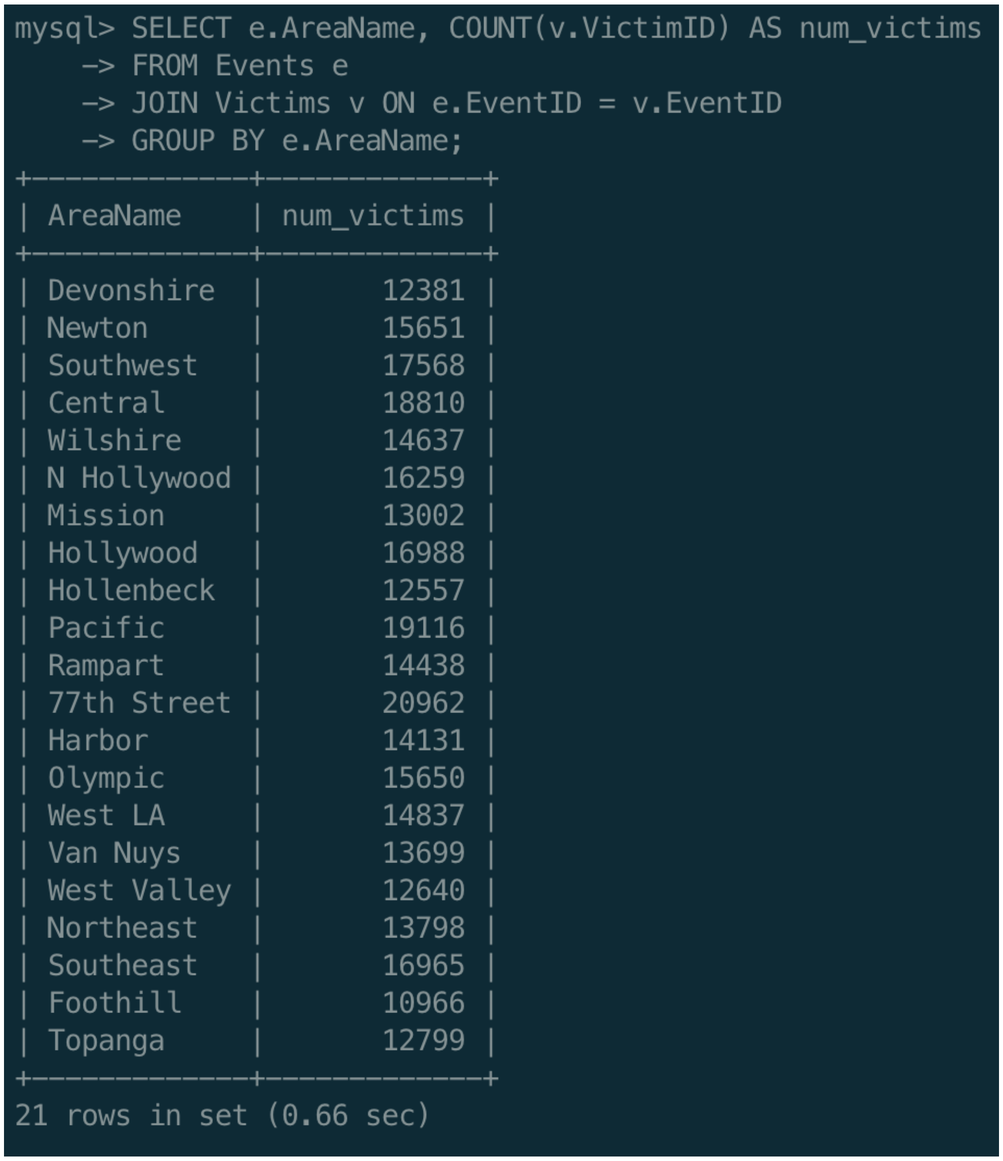
```
mysql> select count(*) from Locations;
+----------+
| count(*) |
+----------+
|    52607 |
+----------+
1 row in set (0.05 sec)
```

Note: The CrimeType and Location could be duplicate, so the number of these two table is much less than Events and Victims.

Two advanced queries

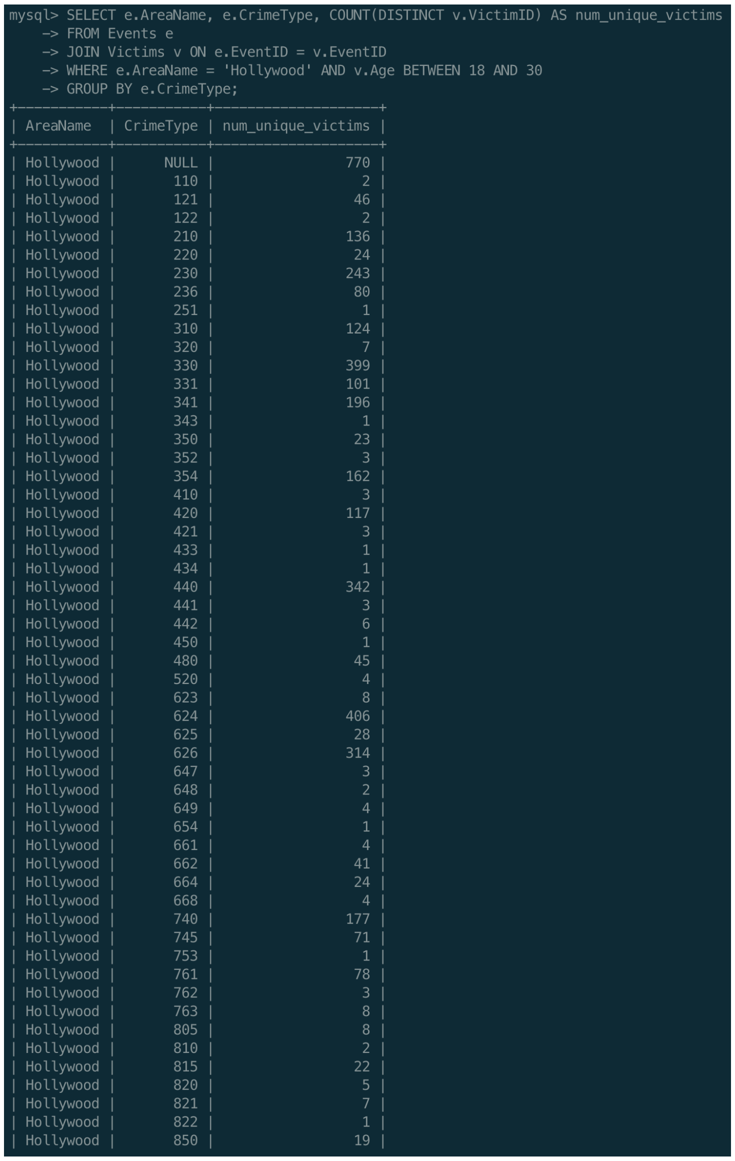
Query A: Give the number of victims in every district.

```
SELECT e.AreaName, COUNT(v.VictimID) AS num_victims
FROM Events e
JOIN Victims v ON e.EventID = v.EventID
GROUP BY e.AreaName;
```


```
mysql> SELECT e.AreaName, COUNT(v.VictimID) AS num_victims
    -> FROM Events e
    -> JOIN Victims v ON e.EventID = v.EventID
    -> GROUP BY e.AreaName;
+-------------+-------------+
| AreaName    | num_victims |
+-------------+-------------+
| Devonshire  |       12381 |
| Newton      |       15651 |
| Southwest   |       17568 |
| Central     |       18810 |
| Wilshire    |       14637 |
| N Hollywood |       16259 |
| Mission     |       13002 |
| Hollywood   |       16988 |
| Hollenbeck  |       12557 |
| Pacific     |       19116 |
| Rampart     |       14438 |
| 77th Street |       20962 |
| Harbor      |       14131 |
| Olympic     |       15650 |
| West LA     |       14837 |
| Van Nuys    |       13699 |
| West Valley |       12640 |
| Northeast   |       13798 |
| Southeast   |       16965 |
| Foothill    |       10966 |
| Topanga     |       12799 |
+-------------+-------------+
21 rows in set (0.66 sec)
```

Query B: Give the number of every CrimeType in Hollywood.

```
SELECT e.AreaName, e.CrimeType, COUNT(DISTINCT v.VictimID) AS num_unique_victims
FROM Events e
JOIN Victims v ON e.EventID = v.EventID
WHERE e.AreaName = 'Hollywood' AND v.Age BETWEEN 18 AND 30
GROUP BY e.CrimeType;
```

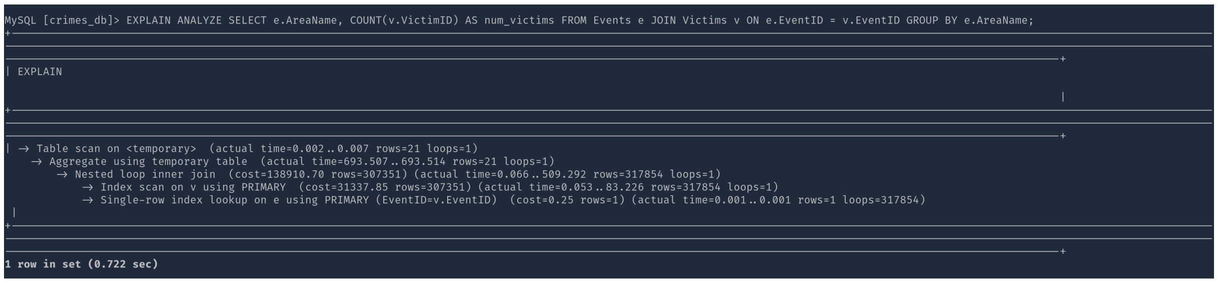


```
mysql> SELECT e.AreaName, e.CrimeType, COUNT(DISTINCT v.VictimID) AS num_unique_victims
    -> FROM Events e
    -> JOIN Victims v ON e.EventID = v.EventID
    -> WHERE e.AreaName = 'Hollywood' AND v.Age BETWEEN 18 AND 30
    -> GROUP BY e.CrimeType;
+-----------+-----------+--------------------+
| AreaName  | CrimeType | num_unique_victims |
+-----------+-----------+--------------------+
| Hollywood |      NULL |                770 |
| Hollywood |       110 |                  2 |
| Hollywood |       121 |                 46 |
| Hollywood |       122 |                  2 |
| Hollywood |       210 |                136 |
| Hollywood |       220 |                 24 |
| Hollywood |       230 |                243 |
| Hollywood |       236 |                 80 |
| Hollywood |       251 |                  1 |
| Hollywood |       310 |                124 |
| Hollywood |       320 |                  7 |
| Hollywood |       330 |                399 |
| Hollywood |       331 |                101 |
| Hollywood |       341 |                196 |
| Hollywood |       343 |                  1 |
| Hollywood |       350 |                 23 |
| Hollywood |       352 |                  3 |
| Hollywood |       354 |                162 |
| Hollywood |       410 |                  3 |
| Hollywood |       420 |                117 |
| Hollywood |       421 |                  3 |
| Hollywood |       433 |                  1 |
| Hollywood |       434 |                  1 |
| Hollywood |       440 |                342 |
| Hollywood |       441 |                  3 |
| Hollywood |       442 |                  6 |
| Hollywood |       450 |                  1 |
| Hollywood |       480 |                 45 |
| Hollywood |       520 |                  4 |
| Hollywood |       623 |                  8 |
| Hollywood |       624 |                406 |
| Hollywood |       625 |                 28 |
| Hollywood |       626 |                314 |
| Hollywood |       647 |                  3 |
| Hollywood |       648 |                  2 |
| Hollywood |       649 |                  4 |
| Hollywood |       654 |                  1 |
| Hollywood |       661 |                  4 |
| Hollywood |       662 |                 41 |
| Hollywood |       664 |                 24 |
| Hollywood |       668 |                  4 |
| Hollywood |       740 |                177 |
| Hollywood |       745 |                 71 |
| Hollywood |       753 |                  1 |
| Hollywood |       761 |                 78 |
| Hollywood |       762 |                  3 |
| Hollywood |       763 |                  8 |
| Hollywood |       805 |                  8 |
| Hollywood |       810 |                  2 |
| Hollywood |       815 |                 22 |
| Hollywood |       820 |                  5 |
| Hollywood |       821 |                  7 |
| Hollywood |       822 |                  1 |
| Hollywood |       850 |                 19 |
```

Indexing Analysis

- **Query A**

```
EXPLAIN ANALYZE
SELECT e.AreaName, COUNT(v.VictimID) AS num_victims
FROM Events e JOIN Victims v ON e.EventID = v.EventID
GROUP BY e.AreaName;
```



```
MySQL [crimes_db]> EXPLAIN ANALYZE SELECT e.AreaName, COUNT(v.VictimID) AS num_victims FROM Events e JOIN Victims v ON e.EventID = v.EventID GROUP BY e.AreaName;
| EXPLAIN
| -> Table scan on <temporary>  (actual time=0.002..0.007 rows=21 loops=1)
    -> Aggregate using temporary table  (actual time=693.507..693.514 rows=21 loops=1)
        -> Nested loop inner join  (cost=138910.70 rows=307351) (actual time=0.066..509.292 rows=317854 loops=1)
            -> Index scan on v using PRIMARY  (cost=31337.85 rows=307351) (actual time=0.053..83.226 rows=317854 loops=1)
            -> Single-row index lookup on e using PRIMARY (EventID=v.EventID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=317854)
 |
1 row in set (0.722 sec)
```

We can find that the aggregation takes the most time in the procedure.

Default index of `Events`:

```
MySQL [crimes_db]> SHOW INDEX FROM Events;
+--------+------------+---------------------+--------------+-------------+-----------+-------------+----------+--------+------+-------
| Table  | Non_unique | Key_name            | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_
+--------+------------+---------------------+--------------+-------------+-----------+-------------+----------+--------+------+-------
| Events |          0 | PRIMARY             |            1 | EventID     | A         |      315062 |     NULL |   NULL |      | BTREE
| Events |          1 | AreaName            |            1 | AreaName    | A         |          22 |     NULL |   NULL | YES  | BTREE
| Events |          1 | AreaName            |            2 | StreetName  | A         |       53164 |     NULL |   NULL | YES  | BTREE
+--------+------------+---------------------+--------------+-------------+-----------+-------------+----------+--------+------+-------
5 rows in set (0.045 sec)
```

Default index of `Victims`:

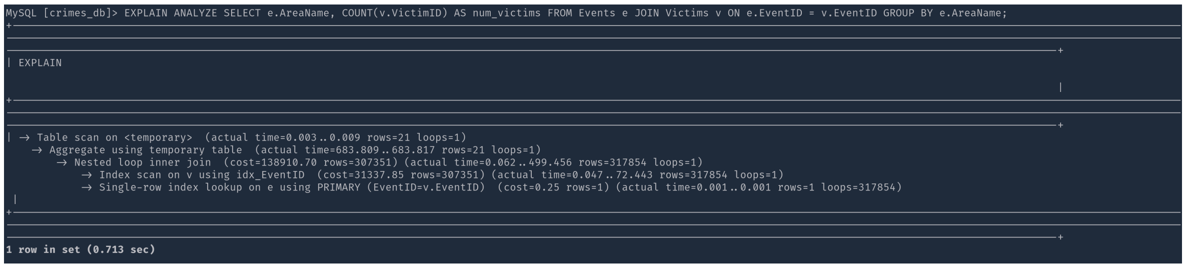
```
MySQL [crimes_db]> SHOW INDEX FROM Victims;
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+-----
| Table   | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comm
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+-----
| Victims |          0 | PRIMARY  |            1 | EventID     | A         |      282232 |     NULL |   NULL |      | BTREE      |
| Victims |          0 | PRIMARY  |            2 | VictimID    | A         |      307351 |     NULL |   NULL |      | BTREE      |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+-----
2 rows in set (0.030 sec)
```

1. Add index on `Victims` using `EventID`:

   Command:

   ```
   CREATE INDEX idx_EventID on Victims (EventID);
   ```

   Indexing analysis after adding new index:

   ```
   MySQL [crimes_db]> EXPLAIN ANALYZE SELECT e.AreaName, COUNT(v.VictimID) AS num_victims FROM Events e JOIN Victims v ON e.EventID = v.EventID GROUP BY e.AreaName;
   | EXPLAIN
   | -> Table scan on <temporary>  (actual time=0.003..0.009 rows=21 loops=1)
       -> Aggregate using temporary table  (actual time=683.809..683.817 rows=21 loops=1)
           -> Nested loop inner join  (cost=138910.70 rows=307351) (actual time=0.062..499.456 rows=317854 loops=1)
               -> Index scan on v using idx_EventID  (cost=31337.85 rows=307351) (actual time=0.047..72.443 rows=317854 loops=1)
               -> Single-row index lookup on e using PRIMARY (EventID=v.EventID)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=317854)
    |
   1 row in set (0.713 sec)
   ```

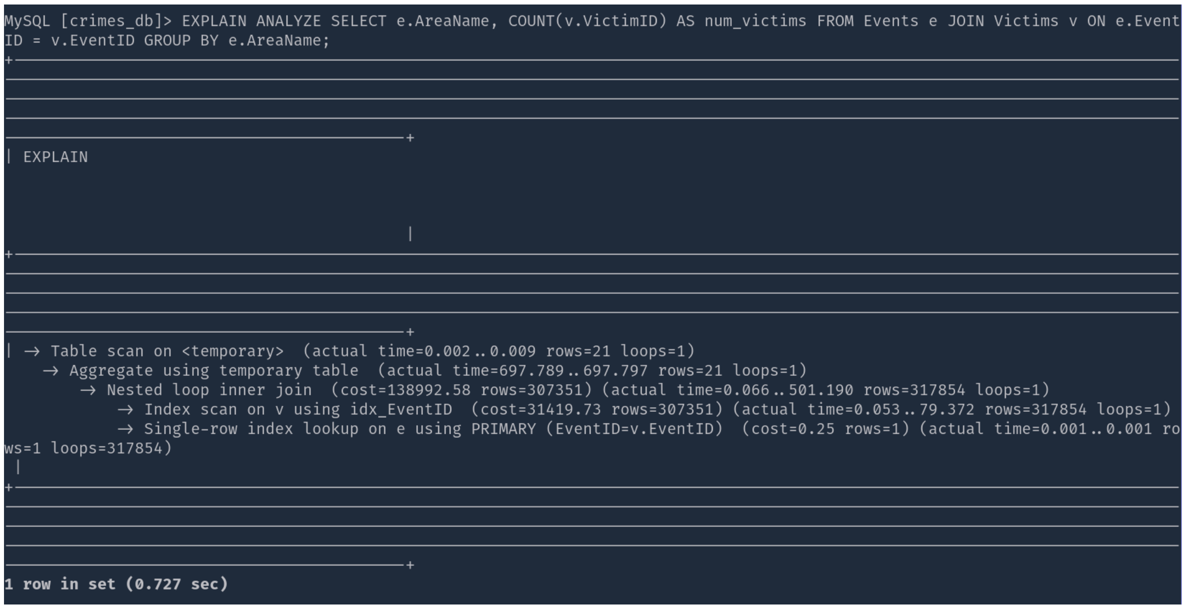
   We can see that the time spent on aggregating reduces from 693 → 683. No obvious improvement because `Victims` already use `EventID` as a key before we add it as the key.

2. Add index on `Events` using `EventID` :

   Command:

   ```
   CREATE INDEX idx_EventID on Events (EventID);
   ```

Indexing analysis after adding new index:

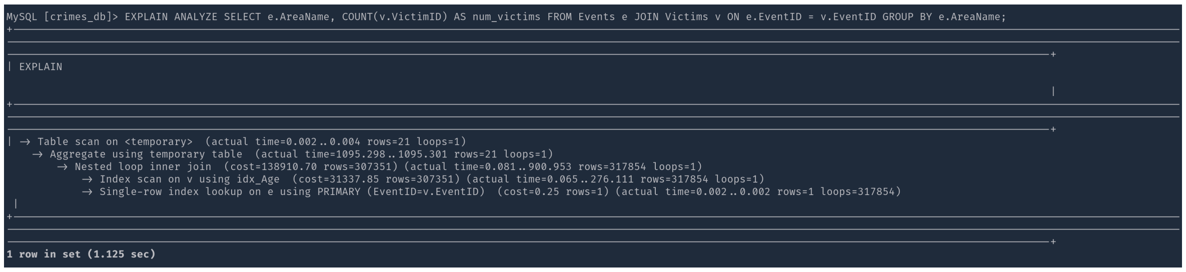


```
MySQL [crimes_db]> EXPLAIN ANALYZE SELECT e.AreaName, COUNT(v.VictimID) AS num_victims FROM Events e JOIN Victims v ON e.Event
ID = v.EventID GROUP BY e.AreaName;
| EXPLAIN
                                                                                                                              |
| -> Table scan on <temporary>  (actual time=0.002..0.009 rows=21 loops=1)
    -> Aggregate using temporary table  (actual time=697.789..697.797 rows=21 loops=1)
        -> Nested loop inner join  (cost=138992.58 rows=307351) (actual time=0.066..501.190 rows=317854 loops=1)
            -> Index scan on v using idx_EventID  (cost=31419.73 rows=307351) (actual time=0.053..79.372 rows=317854 loops=1)
            -> Single-row index lookup on e using PRIMARY (EventID=v.EventID)  (cost=0.25 rows=1) (actual time=0.001..0.001 ro
ws=1 loops=317854)
 |
1 row in set (0.727 sec)
```

No obvious improvement because `Events` already use `EventID` as a key before we add it as the key.

3. Remove new indexes added on `Victims` and `Events` in 1. and 2., and then add new index on `Victims` using `Age`:

```
DROP INDEX idx_EventID on Events;
DROP INDEX idx_EventID on Victims;
CREATE INDEX idx_Age on Victims (Age);
```

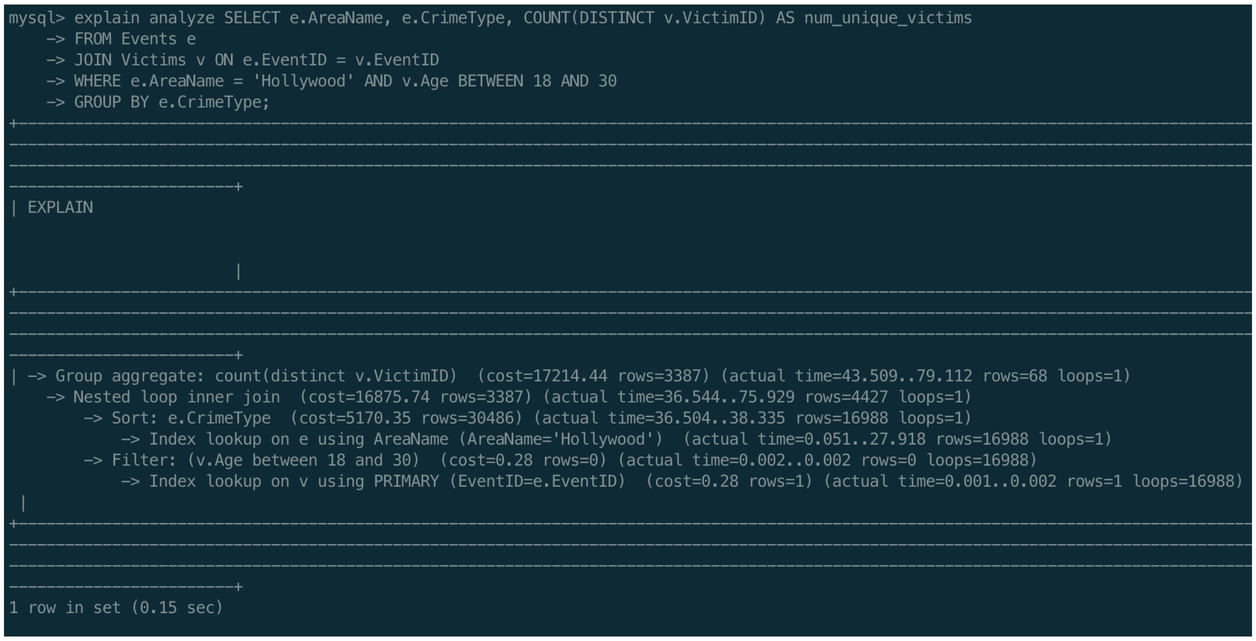


```
MySQL [crimes_db]> EXPLAIN ANALYZE SELECT e.AreaName, COUNT(v.VictimID) AS num_victims FROM Events e JOIN Victims v ON e.EventID = v.EventID GROUP BY e.AreaName;
| EXPLAIN
                                                                                                                              |
| -> Table scan on <temporary>  (actual time=0.002..0.004 rows=21 loops=1)
    -> Aggregate using temporary table  (actual time=1095.298..1095.301 rows=21 loops=1)
        -> Nested loop inner join  (cost=138910.70 rows=307351) (actual time=0.081..900.953 rows=317854 loops=1)
            -> Index scan on v using idx_Age  (cost=31337.85 rows=307351) (actual time=0.065..276.111 rows=317854 loops=1)
            -> Single-row index lookup on e using PRIMARY (EventID=v.EventID)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=317854)
 |
1 row in set (1.125 sec)
```

As we can observe, the aggregation time does not decrease, instead, it increases because the index on `Age` does not help in this query.

- **Query B:**

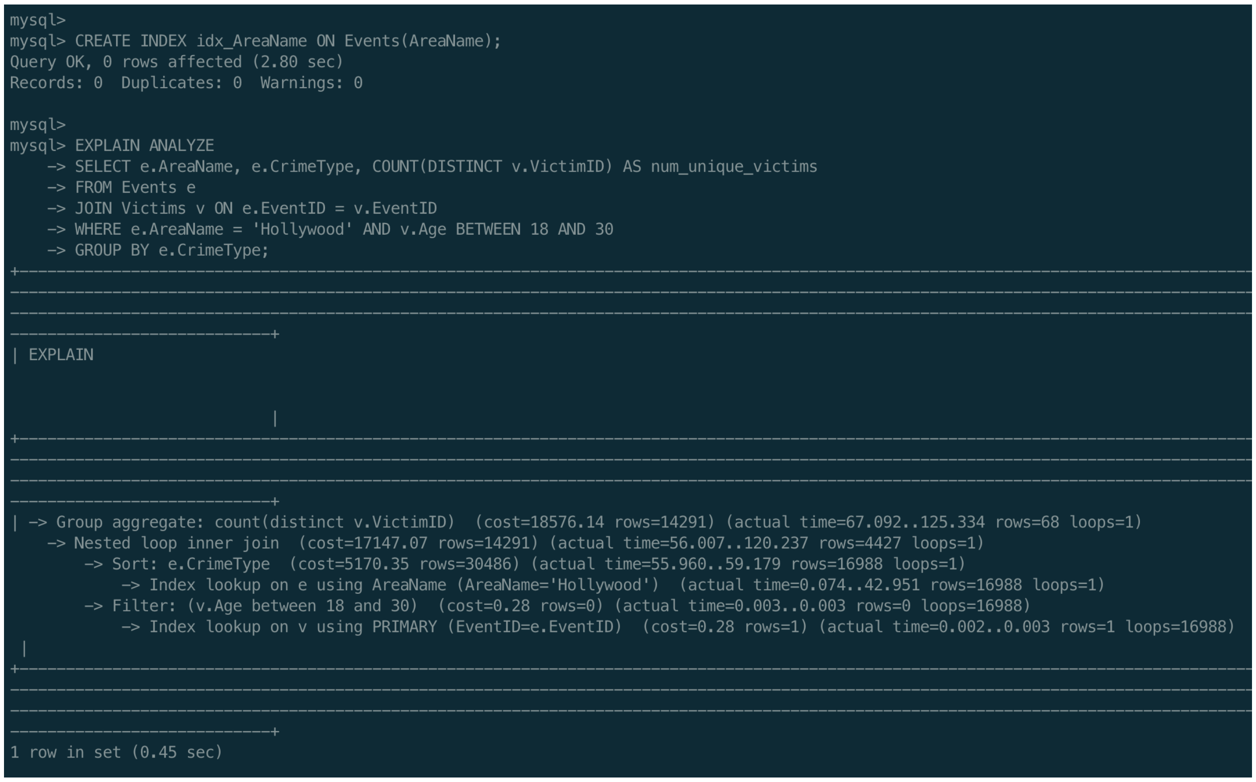
```
SELECT e.AreaName, e.CrimeType, COUNT(DISTINCT v.VictimID) AS num_unique_victims
FROM Events e
JOIN Victims v ON e.EventID = v.EventID
WHERE e.AreaName = 'Hollywood' AND v.Age BETWEEN 18 AND 30
GROUP BY e.CrimeType;
```



```
mysql> explain analyze SELECT e.AreaName, e.CrimeType, COUNT(DISTINCT v.VictimID) AS num_unique_victims
    -> FROM Events e
    -> JOIN Victims v ON e.EventID = v.EventID
    -> WHERE e.AreaName = 'Hollywood' AND v.Age BETWEEN 18 AND 30
    -> GROUP BY e.CrimeType;
+-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------+
| EXPLAIN


                        |
+-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------+
| -> Group aggregate: count(distinct v.VictimID)  (cost=17214.44 rows=3387) (actual time=43.509..79.112 rows=68 loops=1)
    -> Nested loop inner join  (cost=16875.74 rows=3387) (actual time=36.544..75.929 rows=4427 loops=1)
        -> Sort: e.CrimeType  (cost=5170.35 rows=30486) (actual time=36.504..38.335 rows=16988 loops=1)
            -> Index lookup on e using AreaName (AreaName='Hollywood')  (actual time=0.051..27.918 rows=16988 loops=1)
        -> Filter: (v.Age between 18 and 30)  (cost=0.28 rows=0) (actual time=0.002..0.002 rows=0 loops=16988)
            -> Index lookup on v using PRIMARY (EventID=e.EventID)  (cost=0.28 rows=1) (actual time=0.001..0.002 rows=1 loops=16988)
 |
+-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------+
1 row in set (0.15 sec)
```

1. Add index on `Events` using `AreaName`

```
CREATE INDEX idx_AreaName ON Events (AreaName);
```

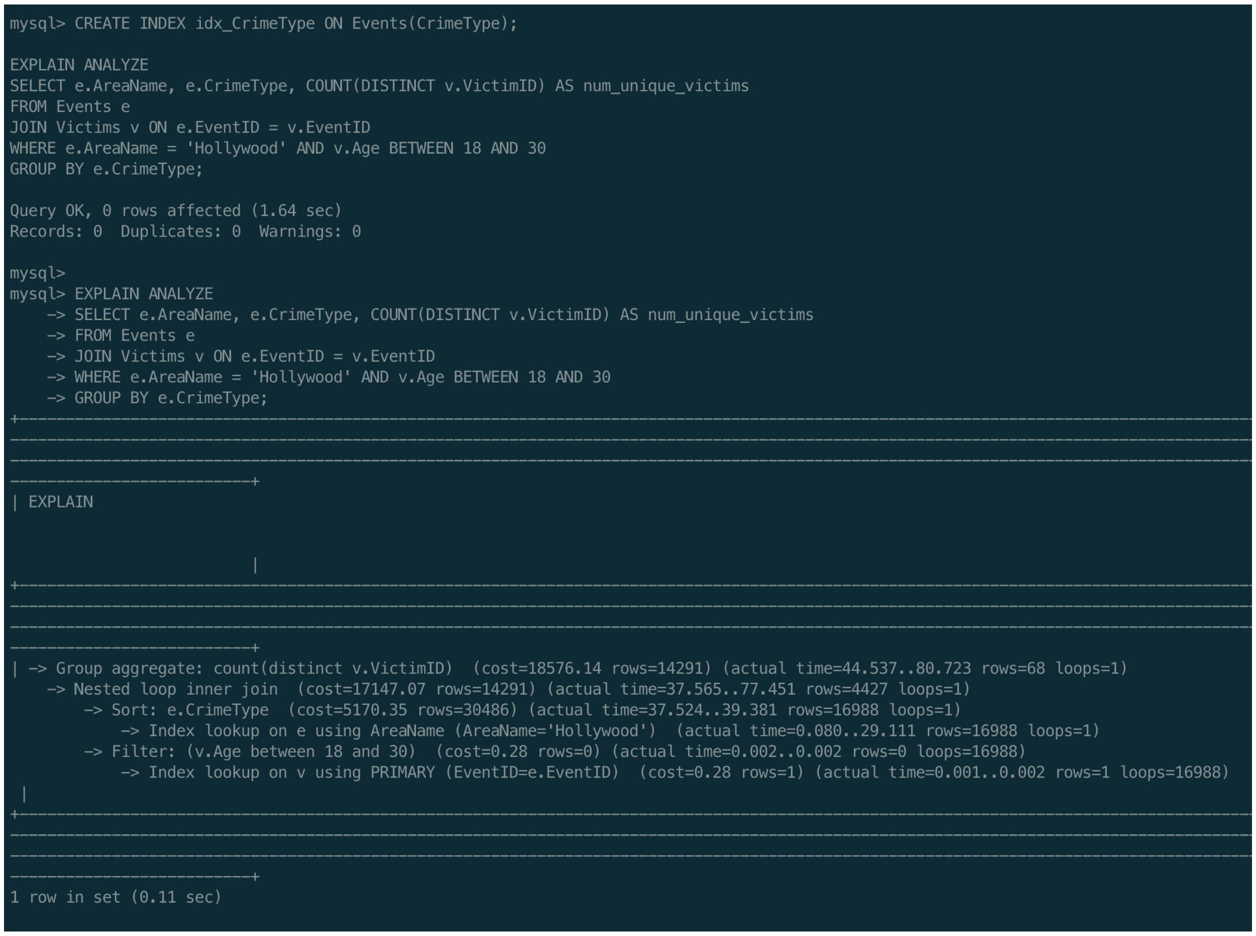


```
mysql>
mysql> CREATE INDEX idx_AreaName ON Events(AreaName);
Query OK, 0 rows affected (2.80 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> EXPLAIN ANALYZE
    -> SELECT e.AreaName, e.CrimeType, COUNT(DISTINCT v.VictimID) AS num_unique_victims
    -> FROM Events e
    -> JOIN Victims v ON e.EventID = v.EventID
    -> WHERE e.AreaName = 'Hollywood' AND v.Age BETWEEN 18 AND 30
    -> GROUP BY e.CrimeType;
+-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------+
| EXPLAIN


                            |
+-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------+
| -> Group aggregate: count(distinct v.VictimID)  (cost=18576.14 rows=14291) (actual time=67.092..125.334 rows=68 loops=1)
    -> Nested loop inner join  (cost=17147.07 rows=14291) (actual time=56.007..120.237 rows=4427 loops=1)
        -> Sort: e.CrimeType  (cost=5170.35 rows=30486) (actual time=55.960..59.179 rows=16988 loops=1)
            -> Index lookup on e using AreaName (AreaName='Hollywood')  (actual time=0.074..42.951 rows=16988 loops=1)
        -> Filter: (v.Age between 18 and 30)  (cost=0.28 rows=0) (actual time=0.003..0.003 rows=0 loops=16988)
            -> Index lookup on v using PRIMARY (EventID=e.EventID)  (cost=0.28 rows=1) (actual time=0.002..0.003 rows=1 loops=16988)
 |
+-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------
-----------------------------+
1 row in set (0.45 sec)
```

We could see the speed even slower.

2. Add index on `Events` using `CrimeType`

```
CREATE INDEX idx_CrimeType ON Events (CrimeType);
```

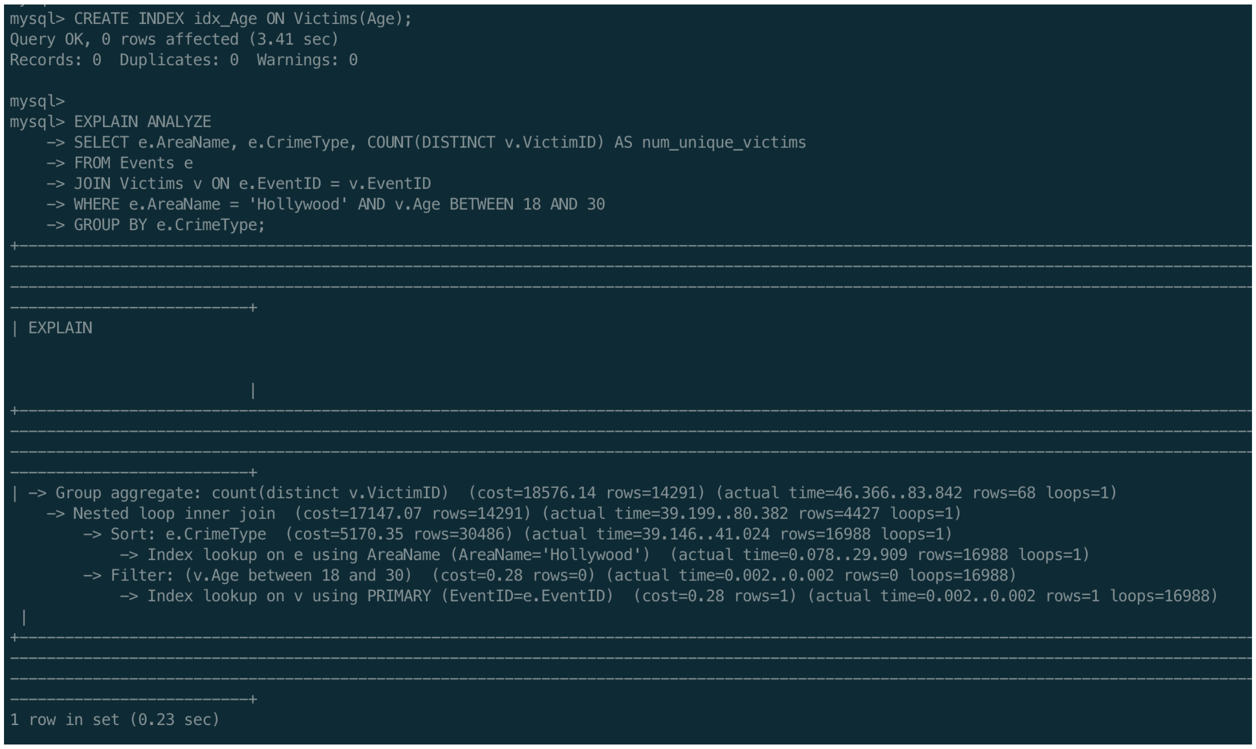


```
mysql> CREATE INDEX idx_CrimeType ON Events(CrimeType);

EXPLAIN ANALYZE
SELECT e.AreaName, e.CrimeType, COUNT(DISTINCT v.VictimID) AS num_unique_victims
FROM Events e
JOIN Victims v ON e.EventID = v.EventID
WHERE e.AreaName = 'Hollywood' AND v.Age BETWEEN 18 AND 30
GROUP BY e.CrimeType;

Query OK, 0 rows affected (1.64 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> EXPLAIN ANALYZE
    -> SELECT e.AreaName, e.CrimeType, COUNT(DISTINCT v.VictimID) AS num_unique_victims
    -> FROM Events e
    -> JOIN Victims v ON e.EventID = v.EventID
    -> WHERE e.AreaName = 'Hollywood' AND v.Age BETWEEN 18 AND 30
    -> GROUP BY e.CrimeType;
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------+
| EXPLAIN



                          |
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------+
| -> Group aggregate: count(distinct v.VictimID)  (cost=18576.14 rows=14291) (actual time=44.537..80.723 rows=68 loops=1)
    -> Nested loop inner join  (cost=17147.07 rows=14291) (actual time=37.565..77.451 rows=4427 loops=1)
        -> Sort: e.CrimeType  (cost=5170.35 rows=30486) (actual time=37.524..39.381 rows=16988 loops=1)
            -> Index lookup on e using AreaName (AreaName='Hollywood')  (actual time=0.080..29.111 rows=16988 loops=1)
        -> Filter: (v.Age between 18 and 30)  (cost=0.28 rows=0) (actual time=0.002..0.002 rows=0 loops=16988)
            -> Index lookup on v using PRIMARY (EventID=e.EventID)  (cost=0.28 rows=1) (actual time=0.001..0.002 rows=1 loops=16988)
 |
+--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------
---------------------------+
1 row in set (0.11 sec)
```

We could see the speed improved a lot.

3. Add index on `Victims` using `Age`

```
CREATE INDEX idx_Age ON Victims (Age);
```



```
mysql> CREATE INDEX idx_Age ON Victims(Age);
Query OK, 0 rows affected (3.41 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> EXPLAIN ANALYZE
    -> SELECT e.AreaName, e.CrimeType, COUNT(DISTINCT v.VictimID) AS num_unique_victims
    -> FROM Events e
    -> JOIN Victims v ON e.EventID = v.EventID
    -> WHERE e.AreaName = 'Hollywood' AND v.Age BETWEEN 18 AND 30
    -> GROUP BY e.CrimeType;
+----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
---------------------------+
| EXPLAIN


                          |
+----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
---------------------------+
| -> Group aggregate: count(distinct v.VictimID)  (cost=18576.14 rows=14291) (actual time=46.366..83.842 rows=68 loops=1)
    -> Nested loop inner join  (cost=17147.07 rows=14291) (actual time=39.199..80.382 rows=4427 loops=1)
        -> Sort: e.CrimeType  (cost=5170.35 rows=30486) (actual time=39.146..41.024 rows=16988 loops=1)
            -> Index lookup on e using AreaName (AreaName='Hollywood')  (actual time=0.078..29.909 rows=16988 loops=1)
        -> Filter: (v.Age between 18 and 30)  (cost=0.28 rows=0) (actual time=0.002..0.002 rows=0 loops=16988)
            -> Index lookup on v using PRIMARY (EventID=e.EventID)  (cost=0.28 rows=1) (actual time=0.002..0.002 rows=1 loops=16988)
 |
+----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------
---------------------------+
1 row in set (0.23 sec)
```

We could see there is not specific improvement.