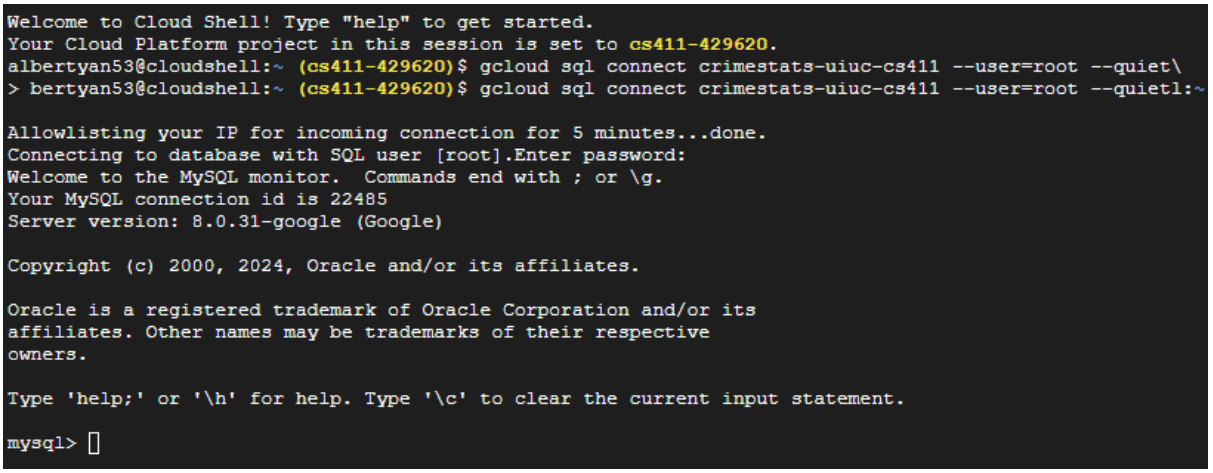
Connection to GCP:



Creation of Tables DDL Commands:

```
CREATE TABLE Universities(
UniversityId INT,
UniversityName VARCHAR(255),
PRIMARY KEY(UniversityId)
);

CREATE TABLE City(
cityName VARCHAR(255),
stateName VARCHAR(255),
population INT,
medianAge REAL,
povertyRate REAL,
medianIncome REAL,
medianProperty REAL,
PRIMARY KEY(cityName,stateName)
);
```

```
CREATE TABLE Crime(
crimeId INT,
universityID INT,
crimeType VARCHAR(255),
crimeAddress VARCHAR(255),
occurredDate VARCHAR(255),
occurredTime VARCHAR(255),
reportedDate VARCHAR(255),
reportedTime VARCHAR(255),
city VARCHAR(255),
state VARCHAR(255),
PRIMARY KEY(crimeId),
FOREIGN KEY(city) REFERENCES City(cityName),
FOREIGN KEY(universityId) REFERENCES Universities(universityId),
FOREIGN KEY(crimeType) REFERENCES CrimeStats(crimeType)
);

CREATE TABLE User(
username VARCHAR(255),
password VARCHAR(255),
address VARCHAR(255),
city VARCHAR(255),
state VARCHAR(255),
PRIMARY KEY(username),
FOREIGN KEY(city, state) REFERENCES City(cityName,stateName)
);

CREATE TABLE CrimeStats(
crimeType VARCHAR(255),
avgAgeVic REAL,
avgAgePerp REAL,
PRIMARY KEY(crimeType)
);
```

Three Tables with At Least 1000 Rows:

```
mysql> SELECT Count(*) FROM Crime;
+----------+
| Count(*) |
+----------+
|     1540 |
+----------+
1 row in set (0.04 sec)
```

```
mysql> SELECT Count(*) FROM User;
+----------+
| Count(*) |
+----------+
|     1011 |
+----------+
1 row in set (0.01 sec)
```

```
mysql> SELECT Count(*) FROM City;
+----------+
| Count(*) |
+----------+
|     1101 |
+----------+
1 row in set (0.00 sec)
```

Queries:

Query 1: Find Universities and reported time for crime instances of Assault or a Sexual Assault that were reported after 5 pm.

```
SELECT u.UniversityName AS UniversityName, reportedTime
FROM Universities u JOIN Crime c ON c.universityID = u.UniversityId
WHERE c.reportedTime > '17:00:00'
AND c.crimeType IN ('Assault', 'Sexual Assault')
AND EXISTS (
   SELECT 1
   FROM Crime c2
   WHERE c2.universityID = u.UniversityId
   AND c2.reportedTime > '17:00:00'
   AND c2.crimeType IN ('Assault', 'Sexual Assault')
);
```

```
+-----------------------+--------------+
| UniversityName        | reportedTime |
+-----------------------+--------------+
 | 18:53:00     |lorida
 | 17:19:00     |lorida
 | 20:48:00     |lorida
 | 21:36:00     |lorida
 | 20:30:00     |lorida
 | 19:26:00     |lorida
 | 22:55:00     |lorida
 | 18:19:00     |lorida
 | 20:59:00     |lorida
 | 20:59:00     |lorida
 | 17:34:00     |lorida
 | 17:03:00     |lorida
 | 20:42:00     |lorida
+-----------------------+--------------+
13 rows in set, 1 warning (0.01 sec)
```

Query 2: Return the number of instances of theft, trespassing, and vandalism for each hour of the day, but only for cities with median properties greater than 100000.

```
SELECT HOUR(occurredTime) AS Hour, COUNT(c.crimeId) AS CrimeCount
FROM Crime c JOIN City ON c.city = City.cityName
WHERE c.crimeType IN ('Theft', 'Trespassing', 'Vandalism') AND
City.medianProperty > 100000
GROUP BY Hour
ORDER BY Hour;
```

```
+------+------------+
| Hour | CrimeCount |
+------+------------+
|    0 |         32 |
|    1 |         22 |
|    2 |         11 |
|    3 |          8 |
|    4 |         11 |
|    5 |         10 |
|    6 |         16 |
|    7 |         16 |
|    8 |         45 |
|    9 |         37 |
|   10 |         45 |
|   11 |         44 |
|   12 |         65 |
|   13 |         41 |
|   14 |         45 |
|   15 |         54 |
|   16 |         75 |
|   17 |         59 |
|   18 |         46 |
|   19 |         44 |
|   20 |         45 |
|   21 |         36 |
|   22 |         33 |
|   23 |         27 |
+------+------------+
24 rows in set (0.01 sec)
```

Query 3: Return crime types, city names, and addresses where a crime was committed in a city that has a median age of over 26 and poverty rate is over 0.2.

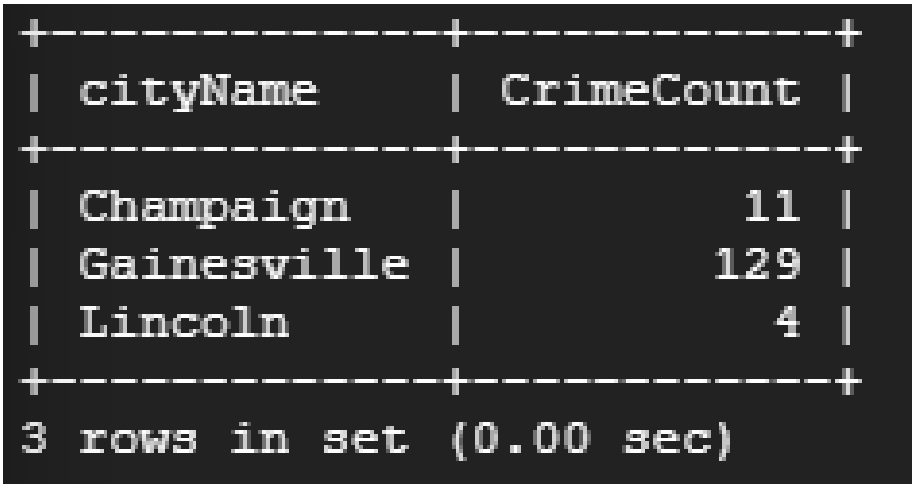
```
SELECT crimeId, crimeType
FROM Crime c JOIN City on c.city = City.cityName
WHERE c.city IN (SELECT cityName FROM City WHERE medianAge>26 AND
povertyRate >0.2)
LIMIT 15;
```

crimeType	cityName	crimeAddress
Theft	Champaign	59 E Green St Champaign
Theft	Champaign	512 S Third St Champaign
Fraud	Champaign	512 S Third St Champaign
Vandalism	Champaign	512 S Third St Champaign
Battery	Champaign	59 E Green St Champaign
Battery	Champaign	501 E Green St Champaign
Theft	Champaign	212 E Green St Champaign
Vandalism	Champaign	212 E Green St Champaign
Theft	Champaign	202 E Green St Champaign
Vandalism	Champaign	202 E Green St Champaign
Theft	Champaign	59 E Green St Champaign
Sexual Assault	Champaign	400 E Green St Champaign
Theft	Champaign	102 E Green St Champaign
Theft	Champaign	1117 S Oak St Champaign
Trespassing	Champaign	1117 S Oak St Champaign

15 rows in set (0.00 sec)

Query 4: Show how many Alcohol and Drug Crimes were committed before 4 pm in each city.

```
SELECT cityName, COUNT(c.crimeId) AS CrimeCount
FROM Crime c JOIN City ON City.cityName = c.city
WHERE c.crimeType IN ('Alcohol', 'Drugs') AND occurredTime < ‘16:00:00’
GROUP BY cityName;
```



```
+-------------+-------------+
| cityName    | CrimeCount  |
+-------------+-------------+
| Champaign   |          11 |
| Gainesville |         129 |
| Lincoln     |           4 |
+-------------+-------------+
3 rows in set (0.00 sec)
```

Indexing Analysis:

In this stage, we analyze the performance of four advanced SQL queries by experimenting with different indexing designs. We use the EXPLAIN ANALYZE command to measure the performance of each query before and after adding indices. We explore trade-offs of adding different indices to various attributes and report on the performance gains or degradations for each indexing configuration. Our analysis includes screenshots of the EXPLAIN ANALYZE commands and a thorough explanation of the results.

Query 1: Find Universities and Addresses for crime instances of Assault or a Sexual Assault that were reported after 5 pm.

```
EXPLAIN ANALYZE
SELECT u.UniversityName AS UniversityName, reportedTime
FROM Universities u JOIN Crime c ON c.universityID = u.UniversityId
WHERE c.reportedTime > '17:00:00'
AND c.crimeType IN ('Assault', 'Sexual Assault')
AND EXISTS (
   SELECT 1
   FROM Crime c2
   WHERE c2.universityID = u.UniversityId
   AND c2.reportedTime > '17:00:00'
   AND c2.crimeType IN ('Assault', 'Sexual Assault')
);
```

```
mysql> EXPLAIN ANALYZE
    -> SELECT u.UniversityName AS UniversityName, reportedTime
    -> FROM Universities u JOIN Crime c ON c.universityID = u.UniversityId
    -> WHERE c.reportedTime > '17:00:00'
    -> AND c.crimeType IN ('Assault', 'Sexual Assault')
    -> AND EXISTS (
    ->     SELECT 1
    ->     FROM Crime c2
    ->     WHERE c2.universityID = u.UniversityId
    ->     AND c2.reportedTime > '17:00:00'
    ->     AND c2.crimeType IN ('Assault', 'Sexual Assault')
    -> );
+------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                          |
+------------------------------------------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=63.51 rows=312) (actual time=3.933..4.065 rows=13 loops=1)
    -> Nested loop inner join  (cost=30.54 rows=18) (actual time=1.839..1.964 rows=13 loops=1)
        -> Filter: ((c.reportedTime > TIME'17:00:00') and (c.universityID is not null))  (cost=24.36 rows=18) (actual time=1.728..
1.847 rows=13 loops=1)
            -> Index range scan on c using crimeType over (crimeType = 'Assault') OR (crimeType = 'Sexual Assault'), with index co
ndition: (c.crimeType in ('Assault','Sexual Assault'))  (cost=24.36 rows=53) (actual time=1.709..1.824 rows=53 loops=1)
        -> Single-row index lookup on u using PRIMARY (UniversityId=c.universityID)  (cost=0.26 rows=1) (actual time=0.009..0.009
rows=1 loops=13)
    -> Single-row index lookup on <subquery2> using <auto_distinct_key> (universityID=c.universityID)  (actual time=0.161..0.161 r
ows=1 loops=13)
        -> Materialize with deduplication  (cost=26.13..26.13 rows=18) (actual time=2.083..2.083 rows=1 loops=1)
            -> Filter: (c2.universityID is not null)  (cost=24.36 rows=18) (actual time=1.570..2.060 rows=13 loops=1)
                -> Filter: (c2.reportedTime > TIME'17:00:00')  (cost=24.36 rows=18) (actual time=1.568..2.057 rows=13 loops=1)
                    -> Index range scan on c2 using crimeType over (crimeType = 'Assault') OR (crimeType = 'Sexual Assault'), with
 index condition: (c2.crimeType in ('Assault','Sexual Assault'))  (cost=24.36 rows=53) (actual time=1.559..2.046 rows=53 loops=1)
 |
```

We experimented with the following indexing designs:

1. CREATE INDEX idx_crime_reportedTime ON Crime (reportedTime);

```
mysql> CREATE INDEX idx_crime_reportedTime ON Crime (reportedTime);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE
    -> SELECT u.UniversityName AS UniversityName, reportedTime
    -> FROM Universities u JOIN Crime c ON c.universityID = u.UniversityId
    -> WHERE c.reportedTime > '17:00:00'
    -> AND c.crimeType IN ('Assault', 'Sexual Assault')
    -> AND EXISTS (
    ->     SELECT 1
    ->     FROM Crime c2
    ->     WHERE c2.universityID = u.UniversityId
    ->     AND c2.reportedTime > '17:00:00'
    ->     AND c2.crimeType IN ('Assault', 'Sexual Assault')
    -> );
+------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------+
| EXPLAIN
                                                                                                                   |
+------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=66.83 rows=342) (actual time=0.256..0.395 rows=13 loops=1)
    -> Nested loop inner join  (cost=30.83 rows=18) (actual time=0.091..0.222 rows=13 loops=1)
        -> Filter: ((c.reportedTime > TIME'17:00:00') and (c.universityID is not null))  (cost=24.36 rows=18) (actual time=0.081..
0.207 rows=13 loops=1)
            -> Index range scan on c using crimeType over (crimeType = 'Assault') OR (crimeType = 'Sexual Assault'), with index co
ndition: (c.crimeType in ('Assault','Sexual Assault'))  (cost=24.36 rows=53) (actual time=0.076..0.198 rows=53 loops=1)
        -> Single-row index lookup on u using PRIMARY (UniversityId=c.universityID)  (cost=0.26 rows=1) (actual time=0.001..0.001
rows=1 loops=13)
    -> Single-row index lookup on <subquery2> using <auto_distinct_key> (universityID=c.universityID)  (actual time=0.013..0.013 r
ows=1 loops=13)
        -> Materialize with deduplication  (cost=26.21..26.21 rows=18) (actual time=0.163..0.163 rows=1 loops=1)
            -> Filter: (c2.universityID is not null)  (cost=24.36 rows=18) (actual time=0.024..0.153 rows=13 loops=1)
                -> Filter: (c2.reportedTime > TIME'17:00:00')  (cost=24.36 rows=18) (actual time=0.024..0.152 rows=13 loops=1)
                    -> Index range scan on c2 using crimeType over (crimeType = 'Assault') OR (crimeType = 'Sexual Assault'), with
 index condition: (c2.crimeType in ('Assault','Sexual Assault'))  (cost=24.36 rows=53) (actual time=0.023..0.148 rows=53 loops=1)
 |
```

2. CREATE INDEX idx_crime_universityID ON Crime (universityID);

```
mysql> CREATE INDEX idx_crime_universityID ON Crime (universityID);
Query OK, 0 rows affected, 1 warning (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> EXPLAIN ANALYZE
    -> SELECT u.UniversityName AS UniversityName, reportedTime
    -> FROM Universities u JOIN Crime c ON c.universityID = u.UniversityId
    -> WHERE c.reportedTime > '17:00:00'
    -> AND c.crimeType IN ('Assault', 'Sexual Assault')
    -> AND EXISTS (
    ->     SELECT 1
    ->     FROM Crime c2
    ->     WHERE c2.universityID = u.UniversityId
    ->     AND c2.reportedTime > '17:00:00'
    ->     AND c2.crimeType IN ('Assault', 'Sexual Assault')
    -> );
+----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------+
| EXPLAIN







                                                                                                                     |
+----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=66.83 rows=342) (actual time=0.407..0.604 rows=13 loops=1)
    -> Nested loop inner join  (cost=30.83 rows=18) (actual time=0.146..0.333 rows=13 loops=1)
        -> Filter: ((c.reportedTime > TIME'17:00:00') and (c.universityID is not null))  (cost=24.36 rows=18) (actual time=0.132..
0.313 rows=13 loops=1)
            -> Index range scan on c using crimeType over (crimeType = 'Assault') OR (crimeType = 'Sexual Assault'), with index co
ndition: (c.crimeType in ('Assault','Sexual Assault'))  (cost=24.36 rows=53) (actual time=0.127..0.301 rows=53 loops=1)
        -> Single-row index lookup on u using PRIMARY (UniversityId=c.universityID)  (cost=0.26 rows=1) (actual time=0.001..0.001
rows=1 loops=13)
    -> Single-row index lookup on <subquery2> using <auto_distinct_key> (universityID=c.universityID)  (actual time=0.020..0.020 r
ows=1 loops=13)
        -> Materialize with deduplication  (cost=26.21..26.21 rows=18) (actual time=0.258..0.258 rows=1 loops=1)
            -> Filter: (c2.universityID is not null)  (cost=24.36 rows=18) (actual time=0.041..0.243 rows=13 loops=1)
                -> Filter: (c2.reportedTime > TIME'17:00:00')  (cost=24.36 rows=18) (actual time=0.040..0.241 rows=13 loops=1)
                    -> Index range scan on c2 using crimeType over (crimeType = 'Assault') OR (crimeType = 'Sexual Assault'), with
 index condition: (c2.crimeType in ('Assault','Sexual Assault'))  (cost=24.36 rows=53) (actual time=0.039..0.235 rows=53 loops=1)
 |
```

3. CREATE INDEX idx_crime_crimeType ON Crime (crimeType);

```
mysql> CREATE INDEX idx_crime_crimeType ON Crime (crimeType);
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE
    -> SELECT u.UniversityName AS UniversityName, reportedTime
    -> FROM Universities u JOIN Crime c ON c.universityID = u.UniversityId
    -> WHERE c.reportedTime > '17:00:00'
    -> AND c.crimeType IN ('Assault', 'Sexual Assault')
    -> AND EXISTS (
    ->     SELECT 1
    ->     FROM Crime c2
    ->     WHERE c2.universityID = u.UniversityId
    ->     AND c2.reportedTime > '17:00:00'
    ->     AND c2.crimeType IN ('Assault', 'Sexual Assault')
    -> );
+--------------------------------------------------------------------------------+
| EXPLAIN                                                                        |
+--------------------------------------------------------------------------------+
| -> Nested loop inner join  (cost=66.83 rows=342) (actual time=0.283..0.404 rows=13 loops=1)
    -> Nested loop inner join  (cost=30.83 rows=18) (actual time=0.091..0.206 rows=13 loops=1)
        -> Filter: ((c.reportedTime > TIME'17:00:00') and (c.universityID is not null))  (cost=24.36 rows=18) (actual time=0.080..
0.189 rows=13 loops=1)
            -> Index range scan on c using idx_crime_crimeType over (crimeType = 'Assault') OR (crimeType = 'Sexual Assault'), wit
h index condition: (c.crimeType in ('Assault','Sexual Assault'))  (cost=24.36 rows=53) (actual time=0.074..0.180 rows=53 loops=1)
        -> Single-row index lookup on u using PRIMARY (UniversityId=c.universityID)  (cost=0.26 rows=1) (actual time=0.001..0.001
rows=1 loops=13)
    -> Single-row index lookup on <subquery2> using <auto_distinct_key> (universityID=c.universityID)  (actual time=0.015..0.015 r
ows=1 loops=13)
        -> Materialize with deduplication  (cost=26.21..26.21 rows=18) (actual time=0.188..0.188 rows=1 loops=1)
            -> Filter: (c2.universityID is not null)  (cost=24.36 rows=18) (actual time=0.032..0.177 rows=13 loops=1)
                -> Filter: (c2.reportedTime > TIME'17:00:00')  (cost=24.36 rows=18) (actual time=0.032..0.175 rows=13 loops=1)
                    -> Index range scan on c2 using idx_crime_crimeType over (crimeType = 'Assault') OR (crimeType = 'Sexual Assau
lt'), with index condition: (c2.crimeType in ('Assault','Sexual Assault'))  (cost=24.36 rows=53) (actual time=0.031..0.171 rows=53
 loops=1)
```

The EXPLAIN ANALYZE command was used to measure the performance before and after adding indices. The query without any indexing had a cost of 63.51 and all three indexing attempts yielded a cost of 66.83. As a result, the cost increased by a small amount adding the indexes.

Query 2: Return the number of instances of theft, trespassing, and vandalism for each hour of the day, but only for cities with median properties greater than 100000.

SELECT HOUR(occurredTime) AS Hour, COUNT(c.crimeId) AS CrimeCount
FROM Crime c JOIN City ON c.city = City.cityName
WHERE c.crimeType IN ('Theft', 'Trespassing', 'Vandalism') AND
City.medianProperty > 100000
GROUP BY Hour
ORDER BY Hour;

```
mysql> EXPLAIN ANALYZE
    -> SELECT HOUR(occurredTime) AS Hour, COUNT(c.crimeId) AS CrimeCount
    -> FROM Crime c
    -> WHERE c.crimeType IN ('Theft', 'Trespassing', 'Vandalism')
    -> GROUP BY Hour
    -> ORDER BY Hour;
+------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------+
| EXPLAIN


                                                                                    |
+------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------+
| -> Sort: Hour  (actual time=1.853..1.854 rows=24 loops=1)
    -> Table scan on <temporary>  (actual time=1.830..1.832 rows=24 loops=1)
        -> Aggregate using temporary table  (actual time=1.828..1.828 rows=24 loops=1)
            -> Filter: (c.crimeType in ('Theft','Trespassing','Vandalism'))  (cost=156.75 rows=867) (actual time=0.622..1.549 rows
=867 loops=1)
                -> Table scan on c  (cost=156.75 rows=1540) (actual time=0.615..1.132 rows=1540 loops=1)
```

We experimented with the following indexing designs:

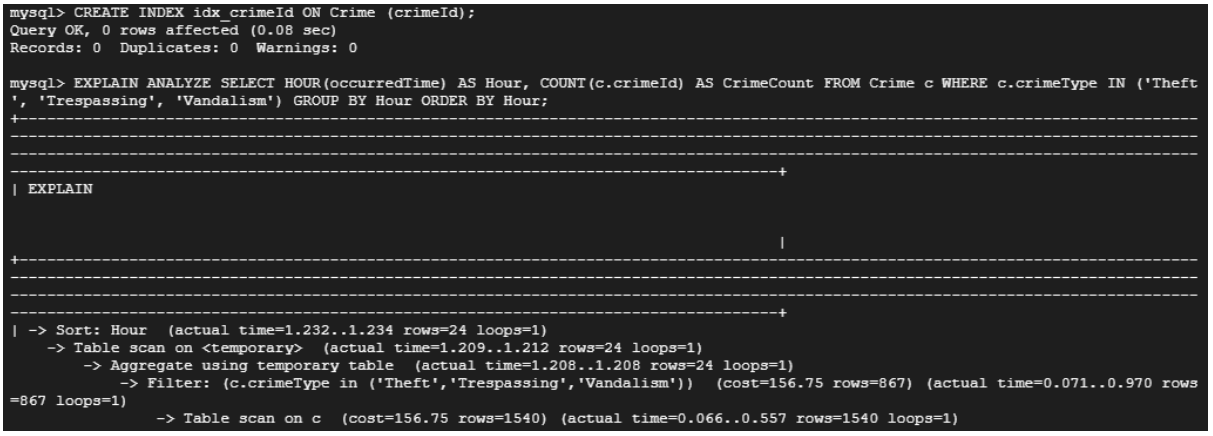
1. CREATE INDEX idx_crime_occurredTime ON Crime (occurredTime);

```
mysql> CREATE INDEX idx_crime_occurredTime ON Crime (occurredTime);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT HOUR(occurredTime) AS Hour, COUNT(c.crimeId) AS CrimeCount FROM Crime c WHERE c.crimeType IN ('Theft
', 'Trespassing', 'Vandalism') GROUP BY Hour ORDER BY Hour;
+------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------+
| EXPLAIN


                                                                                    |
+------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------+
| -> Sort: Hour  (actual time=1.336..1.337 rows=24 loops=1)
    -> Table scan on <temporary>  (actual time=1.308..1.311 rows=24 loops=1)
        -> Aggregate using temporary table  (actual time=1.307..1.307 rows=24 loops=1)
            -> Filter: (c.crimeType in ('Theft','Trespassing','Vandalism'))  (cost=156.75 rows=867) (actual time=0.070..1.058 rows
=867 loops=1)
                -> Table scan on c  (cost=156.75 rows=1540) (actual time=0.066..0.613 rows=1540 loops=1)
```

2. CREATE INDEX idx_crimeId ON Crime (crimeId);

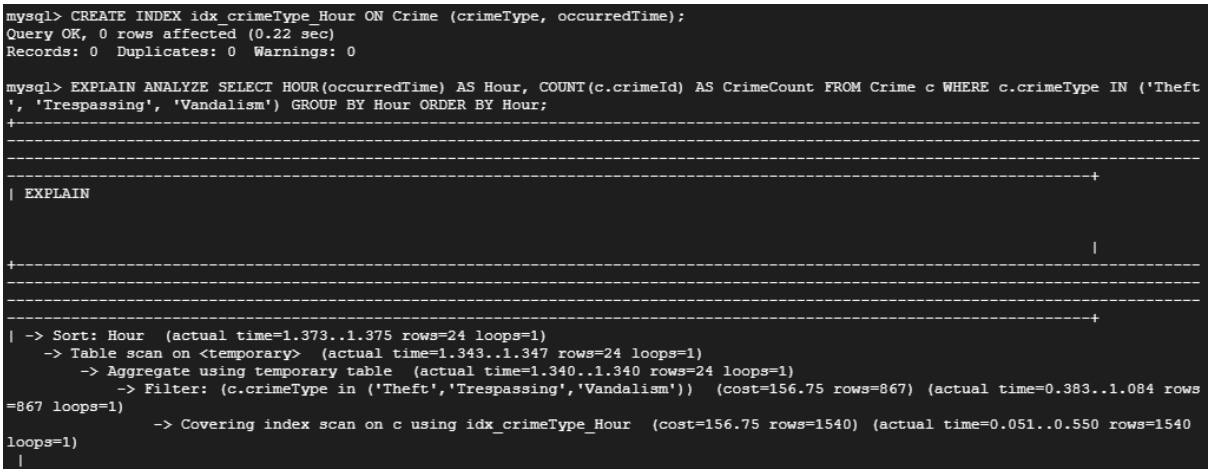


```
mysql> CREATE INDEX idx_crimeId ON Crime (crimeId);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT HOUR(occurredTime) AS Hour, COUNT(c.crimeId) AS CrimeCount FROM Crime c WHERE c.crimeType IN ('Theft
', 'Trespassing', 'Vandalism') GROUP BY Hour ORDER BY Hour;
+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------+
| EXPLAIN


                                                                                                                                  |
+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------+
| -> Sort: Hour  (actual time=1.232..1.234 rows=24 loops=1)
    -> Table scan on <temporary>  (actual time=1.209..1.212 rows=24 loops=1)
        -> Aggregate using temporary table  (actual time=1.208..1.208 rows=24 loops=1)
            -> Filter: (c.crimeType in ('Theft','Trespassing','Vandalism'))  (cost=156.75 rows=867) (actual time=0.071..0.970 rows
=867 loops=1)
                -> Table scan on c  (cost=156.75 rows=1540) (actual time=0.066..0.557 rows=1540 loops=1)
```

3. CREATE INDEX idx_crimeType_Hour ON Crime (crimeType, occurredTime);



```
mysql> CREATE INDEX idx_crimeType_Hour ON Crime (crimeType, occurredTime);
Query OK, 0 rows affected (0.22 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT HOUR(occurredTime) AS Hour, COUNT(c.crimeId) AS CrimeCount FROM Crime c WHERE c.crimeType IN ('Theft
', 'Trespassing', 'Vandalism') GROUP BY Hour ORDER BY Hour;
+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN


                                                                                                                                  |
+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------+
| -> Sort: Hour  (actual time=1.373..1.375 rows=24 loops=1)
    -> Table scan on <temporary>  (actual time=1.343..1.347 rows=24 loops=1)
        -> Aggregate using temporary table  (actual time=1.340..1.340 rows=24 loops=1)
            -> Filter: (c.crimeType in ('Theft','Trespassing','Vandalism'))  (cost=156.75 rows=867) (actual time=0.383..1.084 rows
=867 loops=1)
                -> Covering index scan on c using idx_crimeType_Hour  (cost=156.75 rows=1540) (actual time=0.051..0.550 rows=1540
loops=1)
 |
```

The EXPLAIN ANALYZE command was used to measure the performance before and after adding indices. The cost did not change before and after adding indices.

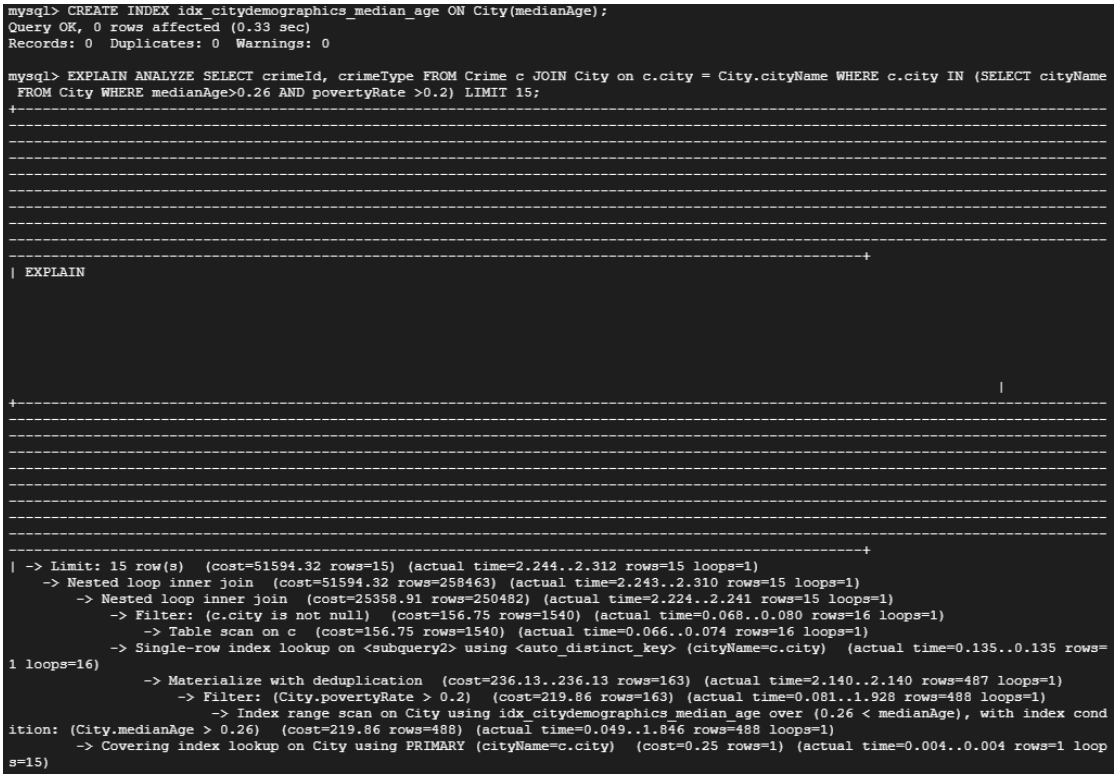
Query 3: Return crime types, city names, and addresses where a crime was committed in a city that has a median age of over 26 and poverty rate is over 0.2.

```
EXPLAIN ANALYZE
SELECT crimeId, crimeType
FROM Crime c JOIN City on c.city = City.cityName
WHERE c.city IN (SELECT cityName FROM City WHERE medianAge>26 AND
povertyRate >0.2)
LIMIT 15;
```

```
mysql> LIMIT 15;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the r
ight syntax to use near 'LIMIT 15' at line 1
mysql> EXPLAIN ANALYZE
    -> SELECT crimeId, crimeType
    -> FROM Crime c JOIN City on c.city = City.cityName
    -> WHERE c.city IN (SELECT cityName FROM City WHERE medianAge>0.26 AND povertyRate >0.2)
    -> LIMIT 15;
+----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
------------------+
| EXPLAIN
                                                                                                                                 |
+----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------
------------------+
| -> Limit: 15 row(s)  (cost=38971.16 rows=15) (actual time=1.217..1.298 rows=15 loops=1)
    -> Nested loop inner join  (cost=38971.16 rows=194358) (actual time=1.215..1.294 rows=15 loops=1)
        -> Nested loop inner join  (cost=19146.31 rows=188356) (actual time=1.185..1.208 rows=15 loops=1)
            -> Filter: (c.city is not null)  (cost=156.75 rows=1540) (actual time=0.078..0.088 rows=16 loops=1)
                -> Table scan on c  (cost=156.75 rows=1540) (actual time=0.076..0.084 rows=16 loops=1)
            -> Single-row index lookup on <subquery2> using <auto_distinct_key> (cityName=c.city)  (actual time=0.069..0.070 rows=
1 loops=16)
                -> Materialize with deduplication  (cost=124.08..124.08 rows=122) (actual time=1.091..1.091 rows=487 loops=1)
                    -> Filter: ((City.medianAge > 0.26) and (City.povertyRate > 0.2))  (cost=111.85 rows=122) (actual time=0.156..
0.852 rows=488 loops=1)
                        -> Table scan on City  (cost=111.85 rows=1101) (actual time=0.150..0.692 rows=1101 loops=1)
        -> Covering index lookup on City using PRIMARY (cityName=c.city)  (cost=0.25 rows=1) (actual time=0.004..0.005 rows=1 loop
s=15)
```

We experimented with the following indexing designs:

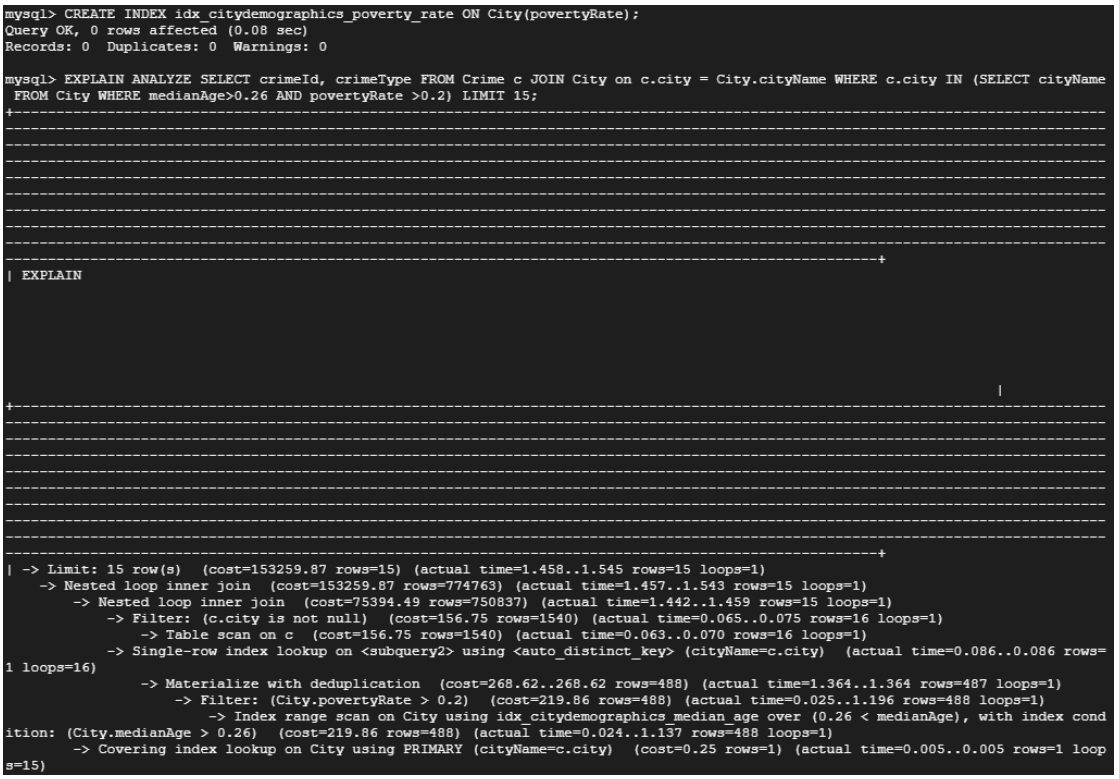
1. CREATE INDEX idx_citydemographics_median_age ON City(medianAge);



```
mysql> CREATE INDEX idx_citydemographics_median_age ON City(medianAge);
Query OK, 0 rows affected (0.33 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT crimeId, crimeType FROM Crime c JOIN City on c.city = City.cityName WHERE c.city IN (SELECT cityName
 FROM City WHERE medianAge>0.26 AND povertyRate >0.2) LIMIT 15;
+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                                                |
+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=51594.32 rows=15) (actual time=2.244..2.312 rows=15 loops=1)
    -> Nested loop inner join  (cost=51594.32 rows=258463) (actual time=2.243..2.310 rows=15 loops=1)
        -> Nested loop inner join  (cost=25358.91 rows=250482) (actual time=2.224..2.241 rows=15 loops=1)
            -> Filter: (c.city is not null)  (cost=156.75 rows=1540) (actual time=0.068..0.080 rows=16 loops=1)
                -> Table scan on c  (cost=156.75 rows=1540) (actual time=0.066..0.074 rows=16 loops=1)
            -> Single-row index lookup on <subquery2> using <auto_distinct_key> (cityName=c.city)  (actual time=0.135..0.135 rows=
1 loops=16)
                -> Materialize with deduplication  (cost=236.13..236.13 rows=163) (actual time=2.140..2.140 rows=487 loops=1)
                    -> Filter: (City.povertyRate > 0.2)  (cost=219.86 rows=163) (actual time=0.081..1.928 rows=488 loops=1)
                        -> Index range scan on City using idx_citydemographics_median_age over (0.26 < medianAge), with index cond
ition: (City.medianAge > 0.26)  (cost=219.86 rows=488) (actual time=0.049..1.846 rows=488 loops=1)
        -> Covering index lookup on City using PRIMARY (cityName=c.city)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loop
s=15)
```

2. CREATE INDEX idx_citydemographics_poverty_rate ON City(povertyRate);



```
mysql> CREATE INDEX idx_citydemographics_poverty_rate ON City(povertyRate);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT crimeId, crimeType FROM Crime c JOIN City on c.city = City.cityName WHERE c.city IN (SELECT cityName
 FROM City WHERE medianAge>0.26 AND povertyRate >0.2) LIMIT 15;
+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                                                |
+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=153259.87 rows=15) (actual time=1.458..1.545 rows=15 loops=1)
    -> Nested loop inner join  (cost=153259.87 rows=774763) (actual time=1.457..1.543 rows=15 loops=1)
        -> Nested loop inner join  (cost=75394.49 rows=750837) (actual time=1.442..1.459 rows=15 loops=1)
            -> Filter: (c.city is not null)  (cost=156.75 rows=1540) (actual time=0.065..0.075 rows=16 loops=1)
                -> Table scan on c  (cost=156.75 rows=1540) (actual time=0.063..0.070 rows=16 loops=1)
            -> Single-row index lookup on <subquery2> using <auto_distinct_key> (cityName=c.city)  (actual time=0.086..0.086 rows=
1 loops=16)
                -> Materialize with deduplication  (cost=268.62..268.62 rows=488) (actual time=1.364..1.364 rows=487 loops=1)
                    -> Filter: (City.povertyRate > 0.2)  (cost=219.86 rows=488) (actual time=0.025..1.196 rows=488 loops=1)
                        -> Index range scan on City using idx_citydemographics_median_age over (0.26 < medianAge), with index cond
ition: (City.medianAge > 0.26)  (cost=219.86 rows=488) (actual time=0.024..1.137 rows=488 loops=1)
        -> Covering index lookup on City using PRIMARY (cityName=c.city)  (cost=0.25 rows=1) (actual time=0.005..0.005 rows=1 loop
s=15)
```

3. CREATE INDEX idx_crime_city_name ON Crime(city);

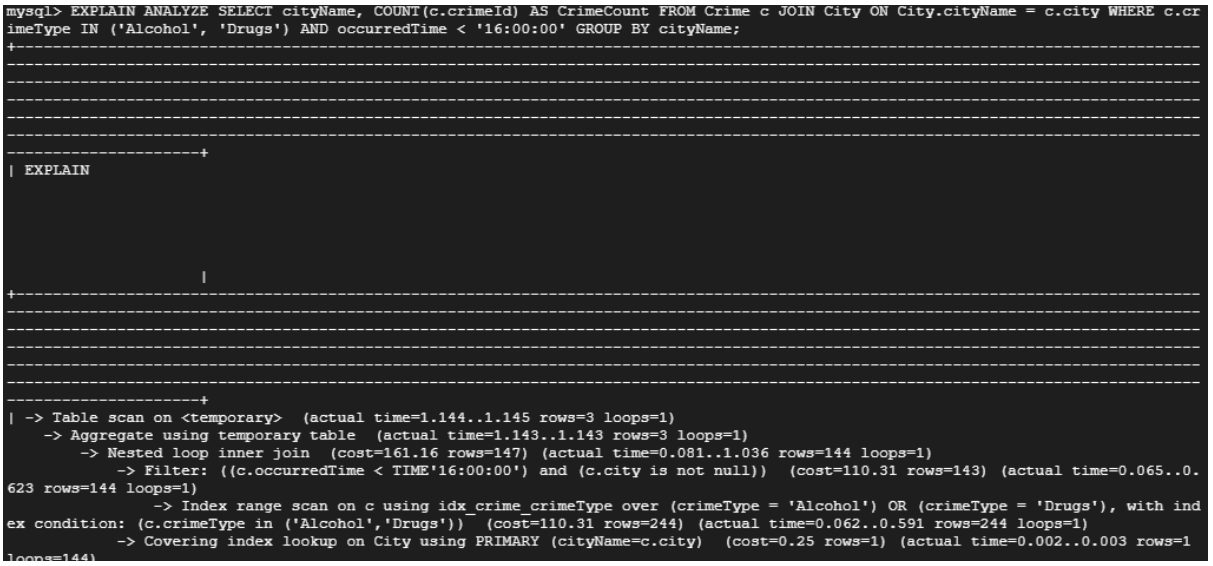
```
mysql> CREATE INDEX idx_crime_city_name ON Crime(city);
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN ANALYZE SELECT crimeId, crimeType FROM Crime c JOIN City on c.city = City.cityName WHERE c.city IN (SELECT cityName
 FROM City WHERE medianAge>0.26 AND povertyRate >0.2) LIMIT 15;
+--------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                      |
+--------------------------------------------------------------------------------------------------------------+
| -> Limit: 15 row(s)  (cost=153259.87 rows=15) (actual time=1.649..1.722 rows=15 loops=1)
    -> Nested loop inner join  (cost=153259.87 rows=774763) (actual time=1.647..1.719 rows=15 loops=1)
        -> Nested loop inner join  (cost=75394.49 rows=750837) (actual time=1.628..1.648 rows=15 loops=1)
            -> Filter: (c.city is not null)  (cost=156.75 rows=1540) (actual time=0.060..0.072 rows=16 loops=1)
                -> Table scan on c  (cost=156.75 rows=1540) (actual time=0.058..0.068 rows=16 loops=1)
            -> Single-row index lookup on <subquery2> using <auto_distinct_key> (cityName=c.city)  (actual time=0.098..0.098 rows=
1 loops=16)
                -> Materialize with deduplication  (cost=268.62..268.62 rows=488) (actual time=1.551..1.551 rows=487 loops=1)
                    -> Filter: (City.povertyRate > 0.2)  (cost=219.86 rows=488) (actual time=0.030..1.332 rows=488 loops=1)
                        -> Index range scan on City using idx_citydemographics_median_age over (0.26 < medianAge), with index cond
ition: (City.medianAge > 0.26)  (cost=219.86 rows=488) (actual time=0.029..1.278 rows=488 loops=1)
        -> Covering index lookup on City using PRIMARY (cityName=c.city)  (cost=0.25 rows=1) (actual time=0.004..0.004 rows=1 loop
s=15)
```

The EXPLAIN ANALYZE command was used to measure the performance before and after adding indices. The index on the city median age performed the best as it had the lowest cost at 51594.32. However, it still performed worse than the query without indices, at a cost of 38971.16.

Query 4: Show how many Alcohol and Drug Crimes were committed before 4 pm in each city.

```
EXPLAIN ANALYZE
SELECT cityName, COUNT(c.crimeId) AS CrimeCount
FROM Crime c JOIN City ON City.cityName = c.city
WHERE c.crimeType IN ('Alcohol', 'Drugs') AND occurredTime < ‘16:00:00’
GROUP BY cityName;
```



```
mysql> EXPLAIN ANALYZE SELECT cityName, COUNT(c.crimeId) AS CrimeCount FROM Crime c JOIN City ON City.cityName = c.city WHERE c.cr
imeType IN ('Alcohol', 'Drugs') AND occurredTime < '16:00:00' GROUP BY cityName;
+--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------+
| EXPLAIN

                    |
+--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------+
| -> Table scan on <temporary>  (actual time=1.144..1.145 rows=3 loops=1)
    -> Aggregate using temporary table  (actual time=1.143..1.143 rows=3 loops=1)
        -> Nested loop inner join  (cost=161.16 rows=147) (actual time=0.081..1.036 rows=144 loops=1)
            -> Filter: ((c.occurredTime < TIME'16:00:00') and (c.city is not null))  (cost=110.31 rows=143) (actual time=0.065..0.
623 rows=144 loops=1)
                -> Index range scan on c using idx_crime_crimeType over (crimeType = 'Alcohol') OR (crimeType = 'Drugs'), with ind
ex condition: (c.crimeType in ('Alcohol','Drugs'))  (cost=110.31 rows=244) (actual time=0.062..0.591 rows=244 loops=1)
            -> Covering index lookup on City using PRIMARY (cityName=c.city)  (cost=0.25 rows=1) (actual time=0.002..0.003 rows=1
loops=144)
```

We experimented with the following indexing designs:

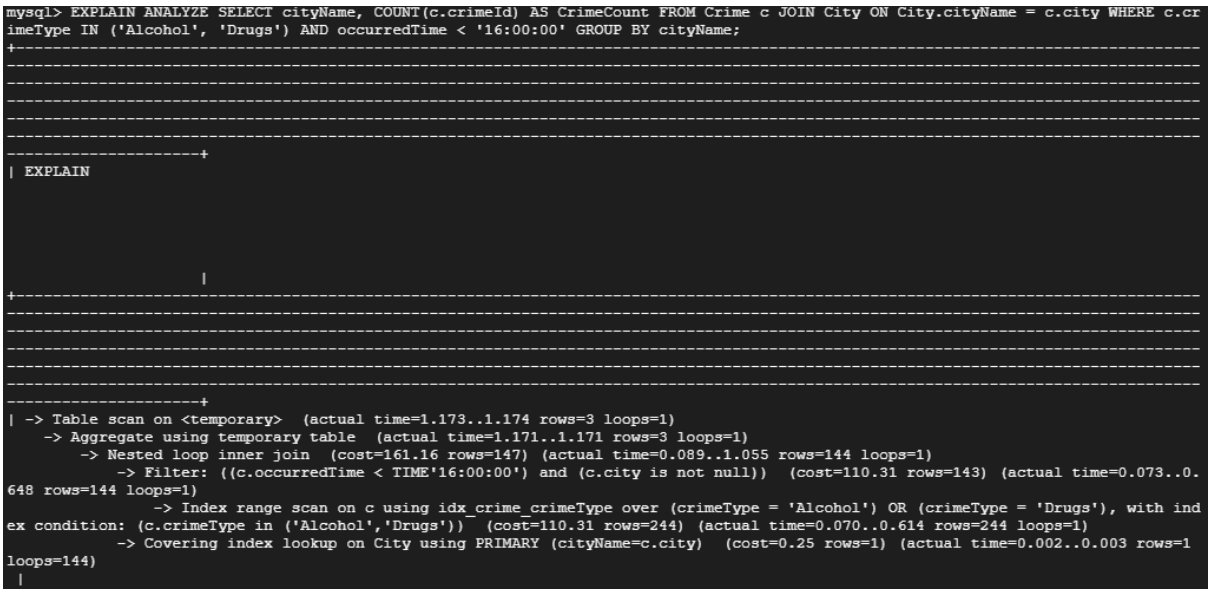
1. CREATE INDEX idx_crime_occurredTime_alcohol_drugs ON Crime (occurredTime);

```
mysql> EXPLAIN ANALYZE SELECT cityName, COUNT(c.crimeId) AS CrimeCount FROM Crime c JOIN City ON City.cityName = c.city WHERE c.cr
imeType IN ('Alcohol', 'Drugs') AND occurredTime < '16:00:00' GROUP BY cityName;
+------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN
                    |
+------------------------------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=1.177..1.178 rows=3 loops=1)
    -> Aggregate using temporary table  (actual time=1.175..1.175 rows=3 loops=1)
        -> Nested loop inner join  (cost=161.16 rows=147) (actual time=0.093..1.057 rows=144 loops=1)
            -> Filter: ((c.occurredTime < TIME'16:00:00') and (c.city is not null))  (cost=110.31 rows=143) (actual time=0.075..0.
620 rows=144 loops=1)
                -> Index range scan on c using idx_crime_crimeType over (crimeType = 'Alcohol') OR (crimeType = 'Drugs'), with ind
ex condition: (c.crimeType in ('Alcohol','Drugs'))  (cost=110.31 rows=244) (actual time=0.071..0.582 rows=244 loops=1)
            -> Covering index lookup on City using PRIMARY (cityName=c.city)  (cost=0.25 rows=1) (actual time=0.002..0.003 rows=1
loops=144)
```

2. CREATE INDEX idx_crime_crimeType_occurredTime ON Crime (crimeType, occurredTime);

```
mysql> EXPLAIN ANALYZE SELECT cityName, COUNT(c.crimeId) AS CrimeCount FROM Crime c JOIN City ON City.cityName = c.city WHERE c.cr
imeType IN ('Alcohol', 'Drugs') AND occurredTime < '16:00:00' GROUP BY cityName;
+------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN
                                                                                                                               |
+------------------------------------------------------------------------------------------------------------------------------+
| -> Table scan on <temporary>  (actual time=0.976..0.977 rows=3 loops=1)
    -> Aggregate using temporary table  (actual time=0.974..0.974 rows=3 loops=1)
        -> Nested loop inner join  (cost=116.55 rows=149) (actual time=0.064..0.858 rows=144 loops=1)
            -> Filter: (c.city is not null)  (cost=65.31 rows=144) (actual time=0.045..0.445 rows=144 loops=1)
                -> Index range scan on c using idx_crime_crimeType_occurredTime over (crimeType = 'Alcohol' AND NULL < occurredTim
e < '16:00:00') OR (crimeType = 'Drugs' AND NULL < occurredTime < '16:00:00'), with index condition: ((c.crimeType in ('Alcohol','
Drugs')) and (c.occurredTime < TIME'16:00:00'))  (cost=65.31 rows=144) (actual time=0.043..0.430 rows=144 loops=1)
            -> Covering index lookup on City using PRIMARY (cityName=c.city)  (cost=0.25 rows=1) (actual time=0.002..0.003 rows=1
loops=144)
```

3. CREATE INDEX idx_crime_universityId_occurredTime ON Crime (universityID, occurredTime);



The EXPLAIN ANALYZE command was used to measure the performance before and after adding indices. The index on crimeType and occurredTime yielded the lowest cost at 116.55, which is lower than the cost without indices at 161.16.

Summary of Results:

The selected indices provided significant performance improvements for the queries involving filtering and joins. The trade-offs in terms of storage and slightly slower insert/update operations were considered acceptable given the performance gains in query execution. The following indices were chosen as the final design:

1. `idx_crime_occurred_date` on `Crime(occurredDate)`

2. `idx_crime_crimeType` on `Crime(crime_type)`

3. `idx_citydemographics_median_age ON City(medianAge)`

4. `idx_crime_universityId_occurredTime ON Crime (universityID, occurredTime)`

These indices ensure optimal performance for the given queries while balancing storage and maintenance costs.