

This query provides a list of the top 15 police stations with the highest number of reported crimes, along with their respective locations and the total crime counts.

```
SELECT P.StationId, P.Location, COUNT(*) AS TotalCrimes
FROM Los_Angeles_Crime_Data.Record R
JOIN Los_Angeles_Crime_Data.District D ON R.DistrictId = D.DistrictId
JOIN Los_Angeles_Crime_Data.PoliceStation P ON P.StationId = D.StationId
GROUP BY P.StationId, P.Location
ORDER BY TotalCrimes DESC
LIMIT 15;
```

The screenshot displays the Google Cloud Platform Cloud SQL console interface. The left sidebar shows the 'SCHEMAS' section with a tree view of the database structure, including tables like 'Los\_Angeles\_Crime\_Data', 'District', and 'PoliceStation'. The main area shows a SQL query in the 'Query Editor' tab, which is the same query provided in the text block. Below the query editor, the 'Result Grid' shows the top 15 police stations with their locations and total crime counts. The 'Output' tab at the bottom shows the execution log with timestamps and row counts for each step of the query execution.

StationId	Location	TotalCrimes
12	7600 S. BROADWAY	20961
14	12312 CLAVER BLVD.	19115
1	251 E. 6TH ST.	18810
3	1546 MARTIN LUTHER KING JR. BLVD.	17568
6	1358 N. WILCOX AVE.	16977
18	145 W. 108TH ST.	16965
15	11640 BURBANK BLVD.	16258
20	1130 S. VERNON AVE.	15650
13	3400 S. CENTRAL AVE.	15648
8	1663 BUTLER AVE.	14837
7	4861 VENICE BLVD.	14631
2	1401 W. 6TH ST.	14432
5	2175 JOHN S. GIBSON BLVD.	14129
11	3353 SAN FERNANDO RD.	13798
9	6240 SYLMAR AVE.	13692

This query provides a list of the top 15 crime types that occurred in Los Angeles in 2020, ranked by their frequency, giving a clear picture of the most prevalent crimes in that year.

```
SELECT YEAR(R.DateOcc) AS Year,
       CT.CrimeTypeDesc,
       COUNT(*) AS TotalCrimes
FROM Los_Angeles_Crime_Data.Record R
JOIN Los_Angeles_Crime_Data.CrimeType CT ON R.CrimeTypeId = CT.CrimeTypeId
WHERE YEAR(R.DateOcc) = 2020
GROUP BY YEAR(R.DateOcc), CT.CrimeTypeDesc
ORDER BY TotalCrimes DESC
LIMIT 15;
```

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Schemas' tree with 'Los\_Angeles\_Crime\_Data' expanded, showing tables like 'Record' and 'CrimeType'. The central pane shows the following SQL query:

```
1 SELECT YEAR(R.DateOcc) AS Year,
2       CT.CrimeTypeDesc,
3       COUNT(*) AS TotalCrimes
4 FROM Los_Angeles_Crime_Data.Record R
5 JOIN Los_Angeles_Crime_Data.CrimeType CT ON R.CrimeTypeId = CT.CrimeTypeId
6 WHERE YEAR(R.DateOcc) = 2020
7 GROUP BY YEAR(R.DateOcc), CT.CrimeTypeDesc
8 ORDER BY TotalCrimes DESC
9 LIMIT 15;
```

The right pane shows the 'Result Grid' with 16 rows of data. The columns are 'Year', 'CrimeTypeDesc', and 'TotalCrimes'. The data shows the top 15 crime types for the year 2020, with 'VEHICLE - STOLEN' having the highest frequency (20,725).

Year	CrimeTypeDesc	TotalCrimes
2020	VEHICLE - STOLEN	20725
2020	BATTERY - SIMPLE ASSAULT	16290
2020	VANDALISM - FELONY (\$400 & OVER, ALL CH...	12888
2020	BURGLARY	12774
2020	BURGLARY FROM VEHICLE	12680
2020	ASSAULT WITH DEADLY WEAPON, AGGRAVATE...	11556
2020	INTIMATE PARTNER - SIMPLE ASSAULT	10795
2020	THEFT PLAIN - PETTY (\$950 & UNDER)	10788
2020	THEFT FROM MOTOR VEHICLE - PETTY (\$950 &...	9690
2020	THEFT OF IDENTITY	7891
2020	VANDALISM - MISDEMEANOR (\$399 OR UNDER)	6973
2020	ROBBERY	6891
2020	THEFT - GRAND (\$950.01 & OVER/EXCEPT GUNG...	5461
2020	THEFT FROM MOTOR VEHICLE - GRAND (\$400 ...	4768
2020	CRIMINAL THREATS - NO WEAPON DISPLAYED	4184

Index Analysis:

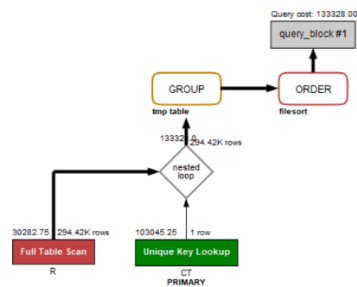
Initial performance:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Record	0	PRIMARY	1	RecordId	A	294415	NULL	NULL	YES	BTREE			YES	NULL
Record	1	DistrictId	1	DistrictId	A	1027	NULL	NULL	YES	BTREE			YES	NULL
Record	1	CrimeTypeId	1	CrimeTypeId	A	93	NULL	NULL	YES	BTREE			YES	NULL
Record	1	PremiseId	1	PremiseId	A	384	NULL	NULL	YES	BTREE			YES	NULL
Record	1	WeaponId	1	WeaponId	A	61	NULL	NULL	YES	BTREE			YES	NULL
Record	1	VictimId	1	VictimId	A	227157	NULL	NULL	YES	BTREE			YES	NULL
Record	1	UserName	1	UserName	A	1	NULL	NULL	YES	BTREE			YES	NULL

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
CrimeType	0	PRIMARY	1	CrimeTypeId	A	126	NULL	NULL		BTREE			YES	NULL

```
EXPLAIN
--> Limit: 15 row(s) (actual time=713.396..713.398 rows=15 loops=1)
--> Sort: TotalCrimes DESC, limit input to 15 row(s) per chunk (actual time=713.395..713.396 rows=15 loops=1)
--> Table scan on <temporary> (actual time=713.318..713.357 rows=129 loops=1)
--> Aggregate using temporary table (actual time=713.312..713.312 rows=129 loops=1)
--> Nested loop inner join (cost=133328.80 rows=294415) (actual time=0.107..1.341,383 rows=197212 loops=1)
--> Filter: ((year(Los_Angeles_Crime_Data.R.DateOcc) = 2020) and (Los_Angeles_Crime_Data.R.CrimeTypeId is not null)) (cost=30282.75 rows=294415) (actual time=0.093..1.155,893 rows=197212 loops=1)
--> Table scan on R (cost=30282.75 rows=294415) (actual time=0.090..1.115,039 rows=317804 loops=1)
--> Single-row index lookup on CT using PRIMARY (CrimeTypeId=Los_Angeles_Crime_Data.R.CrimeTypeId) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=197212)
```

23:40:50 EXPLAIN ANALYZE SELECT YEAR(R.DateOcc) AS Year, CT.CrimeTypeDesc, COUNT(\*) AS TotalCrim... 1 row(s) returned 0.765 sec / 0.000 sec



Add index #1 on Record.DateOcc:

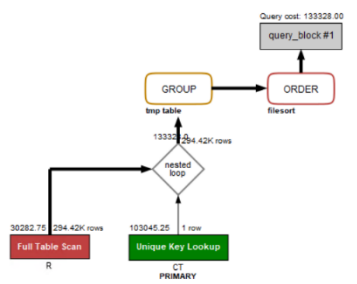
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Record	0	PRIMARY	1	RecordId	A	294415				BTREE			YES	
Record	1	DistrictId	1	DistrictId	A	1027			YES	BTREE			YES	
Record	1	CrimeTypeId	1	CrimeTypeId	A	93			YES	BTREE			YES	
Record	1	PremiseId	1	PremiseId	A	384			YES	BTREE			YES	
Record	1	WeaponId	1	WeaponId	A	61			YES	BTREE			YES	
Record	1	VictimId	1	VictimId	A	227157			YES	BTREE			YES	
Record	1	UserName	1	UserName	A	1			YES	BTREE			YES	
Record	1	idx_dateocc	1	DateOcc	A	583			YES	BTREE			YES	

```

EXPLAIN
--> Limit: 15 row(s) (actual time=732.109..732.111 rows=15 loops=1)
--> Sort: TotalCrimes DESC, limit input to 15 row(s) per chunk (actual time=732.108..732.109 rows=15 loops=1)
--> Table scan on <temporary> (actual time=732.030..732.074 rows=129 loops=1)
--> Aggregate using temporary table (actual time=732.024..732.024 rows=129 loops=1)
--> Nested loop inner join (cost=133328.00 rows=294415) (actual time=0.863..349.785 rows=197212 loops=1)
--> Filter: ((year(Los_Angeles_Crime_Data.R.DateOcc) = 2020) and (Los_Angeles_Crime_Data.R.CrimeTypeId is not null)) (cost=30282.75 rows=294415) (actual time=0.052..158.945 rows=197212 loops=1)
--> Table scan on R (cost=30282.75 rows=294415) (actual time=0.049..118.770 rows=317804 loops=1)
--> Single-row index lookup on CT using PRIMARY (CrimeTypeId=Los_Angeles_Crime_Data.R.CrimeTypeId) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=197212)

```

23:56:37 EXPLAIN ANALYZE SELECT YEAR(R.DateOcc) AS Year, CT.CrimeTypeDesc, COUNT(\*) AS TotalCrim... 1 row(s) returned 0.750 sec / 0.000 sec



By checking the analysis result, we can find that the new index on Record.DateOcc doesn't help to improve this query. One possible reason is that the query is using "YEAR(R.DateOcc) = 2020" to select and group data entries. However, the index is created based on the date. When the query tries to filter the entries, the processor still needs to check the YEAR attribute of each date. So it's still a table scan. Also, another possible reason is that no matter whether there is an index on DateOcc or not, the Record table needs to be scanned to check whether "CrimeTypeId is not null". So a table scan is always required.

Add index #2 on CrimeType.CrimeTypeDesc:

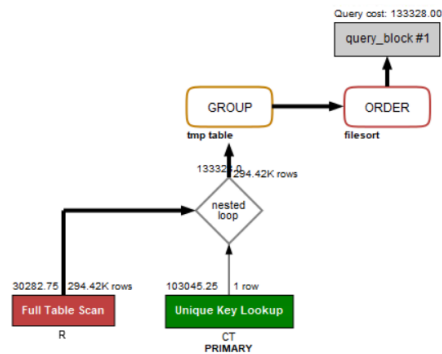
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expressi
CrimeType	0	PRIMARY	1	CrimeTypeId	A	126				BTREE			YES	
CrimeType	1	idx_crimedesc	1	CrimeTypeDesc	A	126			YES	BTREE			YES	

```

EXPLAIN
--> Limit: 15 row(s) (actual time=725.936..725.938 rows=15 loops=1)
--> Sort: TotalCrimes DESC, limit input to 15 row(s) per chunk (actual time=725.935..725.936 rows=15 loops=1)
--> Table scan on <temporary> (actual time=725.858..725.896 rows=129 loops=1)
--> Aggregate using temporary table (actual time=725.852..725.852 rows=129 loops=1)
--> Nested loop inner join (cost=133328.00 rows=294415) (actual time=0.074..349.001 rows=197212 loops=1)
--> Filter: ((year(Los_Angeles_Crime_Data.R.DateOcc) = 2020) and (Los_Angeles_Crime_Data.R.CrimeTypeId is not null)) (cost=30282.75 rows=294415) (actual time=0.058..159.965 rows=197212 loops=1)
--> Table scan on R (cost=30282.75 rows=294415) (actual time=0.055..118.903 rows=317804 loops=1)
--> Single-row index lookup on CT using PRIMARY (CrimeTypeId=Los_Angeles_Crime_Data.R.CrimeTypeId) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=197212)

```

00:41:15 EXPLAIN ANALYZE SELECT YEAR(R.DateOcc) AS Year, CT.CrimeTypeDesc, COUNT(\*) AS TotalCrim... 1 row(s) returned 0.750 sec / 0.000 sec



By checking the analysis result, we can find that the new index on CrimeType.CrimeTypeDesc doesn't help to improve this query. This is because the CrimeTypeDesc attribute is a string. Adding an index on it won't help to improve the grouping efficiency since each group has exactly one unique string.

Add index #3 on both Record.DateOcc and CrimeType.CrimeTypeDesc:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Record	0	PRIMARY	1	RecordId	A	294415				BTREE			YES	
Record	1	DistrictId	1	DistrictId	A	1027			YES	BTREE			YES	
Record	1	CrimeTypeId	1	CrimeTypeId	A	93			YES	BTREE			YES	
Record	1	PremiseId	1	PremiseId	A	384			YES	BTREE			YES	
Record	1	WeaponId	1	WeaponId	A	61			YES	BTREE			YES	
Record	1	VictimId	1	VictimId	A	227157			YES	BTREE			YES	
Record	1	UserName	1	UserName	A	1			YES	BTREE			YES	
Record	1	idx_dateocc	1	DateOcc	A	583			YES	BTREE			YES	

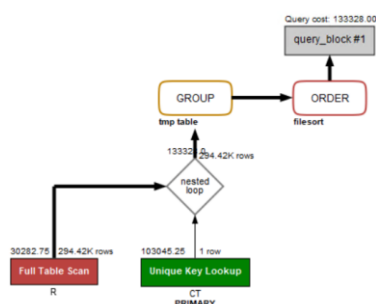
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expressi
CrimeType	0	PRIMARY	1	CrimeTypeId	A	126				BTREE			YES	
CrimeType	1	idx_crimedesc	1	CrimeTypeDesc	A	126			YES	BTREE			YES	

```

EXPLAIN
--> Limit: 15 row(s) (actual time=725.475..725.477 rows=15 loops=1)
--> Sort: TotalCrimes DESC, limit input to 15 row(s) per chunk (actual time=725.474..725.475 rows=15 loops=1)
--> Table scan on <temporary> (actual time=725.395..725.434 rows=129 loops=1)
--> Aggregate using temporary table (actual time=725.388..725.388 rows=129 loops=1)
--> Nested loop inner join (cost=133328.00 rows=294415) (actual time=0.070..347.157 rows=197212 loops=1)
--> Filter: ((year(Los_Angeles_Crime_Data.R.DateOcc) = 2020) and (Los_Angeles_Crime_Data.R.CrimeTypeId is not null)) (cost=30282.75 rows=294415) (actual time=0.058..158.847 rows=197212 loops=1)
--> Table scan on R (cost=30282.75 rows=294415) (actual time=0.055..117.821 rows=317804 loops=1)
--> Single-row index lookup on CT using PRIMARY (CrimeTypeId=Los_Angeles_Crime_Data.R.CrimeTypeId) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=197212)

```

00:49:46 EXPLAIN ANALYZE SELECT YEAR(R.DateOcc) AS Year, CT.CrimeTypeDesc, COUNT(\*) AS TotalCrim... 1 row(s) returned 0.750 sec / 0.000 sec



By checking the analysis result, we can find that these two indexes on Record.DateOcc and CrimeType.CrimeTypeDesc doesn't help to improve this query. For the index on Record.DateOcc, it cannot improve the efficiency of the filter "YEAR(R.DateOcc) = 2020" and the group by operation, while for the index on CrimeType.CrimeTypeDesc, it cannot help the group by operation as well.

Summary:

Since these index designs cannot help to improve the query performance, we decide to use the initial design without any additional indexes for this query.

This query highlights which specific demographics (in terms of age and sex) were most impacted by crime in Los Angeles in 2021.

```
SELECT V.Sex, CASE
    WHEN V.Age BETWEEN 0 AND 10 THEN '0-10'
    WHEN V.Age BETWEEN 10 AND 20 THEN '10-20'
    WHEN V.Age BETWEEN 20 AND 30 THEN '20-30'
    WHEN V.Age BETWEEN 30 AND 40 THEN '30-40'
    WHEN V.Age BETWEEN 40 AND 50 THEN '40-50'
    WHEN V.Age BETWEEN 50 AND 60 THEN '50-60'
    WHEN V.Age BETWEEN 60 AND 70 THEN '60-70'
    ELSE '70+'
END AS AgeGroup,
COUNT(*) AS NumberOfCrimes
FROM Los_Angeles_Crime_Data.Record R
JOIN Los_Angeles_Crime_Data.Victim V ON R.VictimId = V.VictimId
WHERE YEAR(R.DateOcc) = 2021
GROUP BY V.Sex, AgeGroup
ORDER BY NumberOfCrimes DESC
LIMIT 15;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the query from the previous block. The Results window displays the following data:

Sex	AgeGroup	NumberOfCrimes
M	30-40	12433
F	20-30	11969
M	20-30	11400
F	30-40	10770
M	40-50	8610
F	40-50	7466
M	50-60	6855
F	50-60	5314
M	60-70	3997
F	10-20	2851
F	60-70	2836
M	10-20	2232
M	70+	1708
F	70+	1555
X	10-20	633

The bottom status bar indicates the query was executed successfully, returning 0 rows.

Index Analysis:

Initial performance:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Record	0	PRIMARY	1	RecordId	A	294415	HULL	HULL		BTREE			YES	HULL
Record	1	DistrictId	1	DistrictId	A	1027	HULL	HULL	YES	BTREE			YES	HULL
Record	1	CrimeTypeId	1	CrimeTypeId	A	93	HULL	HULL	YES	BTREE			YES	HULL
Record	1	PremiseId	1	PremiseId	A	384	HULL	HULL	YES	BTREE			YES	HULL
Record	1	WeaponId	1	WeaponId	A	61	HULL	HULL	YES	BTREE			YES	HULL
Record	1	VictimId	1	VictimId	A	227157	HULL	HULL	YES	BTREE			YES	HULL
Record	1	UserName	1	UserName	A	1	HULL	HULL	YES	BTREE			YES	HULL

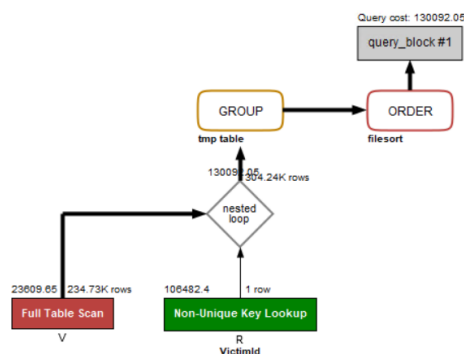
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Victim	0	PRIMARY	1	VictimId	A	234734	HULL	HULL		BTREE			YES	HULL

```

EXPLAIN
--> Limit: 15 row(s) (actual time=1059.378..1059.380 rows=15 loops=1)
--> Sort: NumberOfCrimes DESC, limit input to 15 row(s) per chunk (actual time=1059.378..1059.379 rows=15 loops=1)
--> Table scan on <temporary> (actual time=1059.344..1059.352 rows=30 loops=1)
--> Aggregate using temporary table (actual time=1059.342..1059.342 rows=30 loops=1)
--> Nested loop inner join (cost=130092.05 rows=304235) (actual time=565.315..978.786 rows=91482 loops=1)
--> Table scan on V (cost=23609.65 rows=234734) (actual time=0.035..73.783 rows=240614 loops=1)
--> Filter: (year(Los_Angeles_Crime_Data.R.DateOcc) = 2021) (cost=0.32 rows=1) (actual time=0.003..0.004 rows=0 loops=240614)
--> Index lookup on R using VictimId (VictimId=Los_Angeles_Crime_Data.V.VictimId) (cost=0.32 rows=1) (actual time=0.003..0.003 rows=1 loops=240614)

```

01:07:49 EXPLAIN ANALYZE SELECT V.Sex, CASE WHEN V.Age BETWEEN 0 AND 10 THEN '0-10' W... 1 row(s) returned 1.078 sec / 0.000 sec



Add index #1 on Victim.Age;

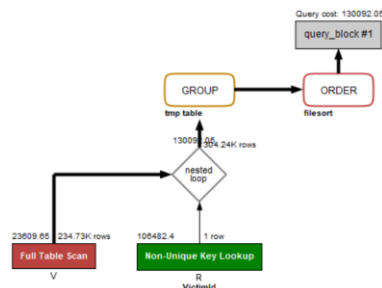
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Victim	0	PRIMARY	1	VictimId	A	234734	HULL	HULL		BTREE			YES	HULL
Victim	1	idx_age	1	Age	A	83	HULL	HULL	YES	BTREE			YES	HULL

```

EXPLAIN
--> Limit: 15 row(s) (actual time=1054.524..1054.526 rows=15 loops=1)
--> Sort: NumberOfCrimes DESC, limit input to 15 row(s) per chunk (actual time=1054.524..1054.525 rows=15 loops=1)
--> Table scan on <temporary> (actual time=1054.493..1054.500 rows=30 loops=1)
--> Aggregate using temporary table (actual time=1054.491..1054.491 rows=30 loops=1)
--> Nested loop inner join (cost=130092.05 rows=304235) (actual time=554.715..972.357 rows=91482 loops=1)
--> Table scan on V (cost=23609.65 rows=234734) (actual time=0.050..75.764 rows=240614 loops=1)
--> Filter: (year(Los_Angeles_Crime_Data.R.DateOcc) = 2021) (cost=0.32 rows=1) (actual time=0.003..0.004 rows=0 loops=240614)
--> Index lookup on R using VictimId (VictimId=Los_Angeles_Crime_Data.V.VictimId) (cost=0.32 rows=1) (actual time=0.003..0.003 rows=1 loops=240614)

```

01:11:22 EXPLAIN ANALYZE SELECT V.Sex, CASE WHEN V.Age BETWEEN 0 AND 10 THEN '0-10' W... 1 row(s) returned 1.078 sec / 0.000 sec



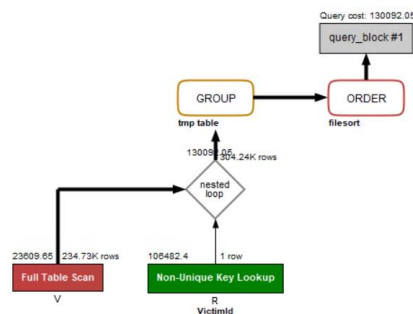
By checking the analysis result, we can find that the new index on Victim.Age doesn't help to improve this query. One possible reason is that this attribute is not used to select the data entries. Instead, its value is used to rename the AgeGroup as an output attribute. So this attribute actually doesn't participate in the query process.

Add index #2 on Victim.Sex:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Victim	0	PRIMARY	1	VictimId	A	234734				BTREE			YES	
Victim	1	idx_sex	1	Sex	A	4			YES	BTREE			YES	

```
EXPLAIN
-> Limit: 15 row(s) (actual time=1041.694..1041.696 rows=15 loops=1)
-> Sort: NumberOfCrimes DESC, limit input to 15 row(s) per chunk (actual time=1041.692..1041.693 rows=15 loops=1)
-> Table scan on <temporary> (actual time=1041.660..1041.668 rows=30 loops=1)
-> Aggregate using temporary table (actual time=1041.658..1041.658 rows=30 loops=1)
-> Nested loop inner join (cost=130092.05 rows=304235) (actual time=566.371..964.241 rows=91482 loops=1)
-> Table scan on V (cost=23609.65 rows=234734) (actual time=0.030..73.826 rows=240614 loops=1)
-> Filter: (year(Los_Angeles_Crime_Data.R.DateOcc) = 2021) (cost=0.32 rows=1) (actual time=0.003..0.004 rows=0 loops=240614)
-> Index lookup on R using VictimId (VictimId=Los_Angeles_Crime_Data.V.VictimId) (cost=0.32 rows=1) (actual time=0.003..0.003 rows=1 loops=240614)
```

01:25:58 EXPLAIN ANALYZE SELECT V.Sex, CASE WHEN V.Age BETWEEN 0 AND 10 THEN '0-10' W... 1 row(s) returned 1.062 sec / 0.000 sec



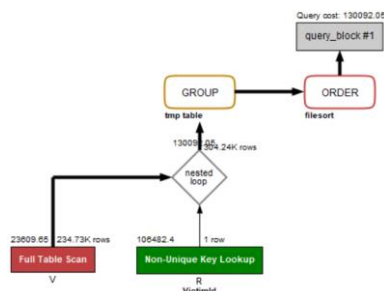
By checking the analysis result, we can find that the new index on Victim.Sex doesn't help to improve this query. One possible reason is that since the results are grouped by two attributes, a table scan is always required for the AgeGroup output attribute. Even though we add a index on the Sex, we cannot eliminate the table scan cost.

Add index #3 on both Victim.Age and Victim.Sex:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Victim	0	PRIMARY	1	VictimId	A	234734				BTREE			YES	
Victim	1	idx_sex	1	Sex	A	4			YES	BTREE			YES	
Victim	1	idx_age	1	Age	A	83			YES	BTREE			YES	

```
EXPLAIN
-> Limit: 15 row(s) (actual time=1083.374..1083.376 rows=15 loops=1)
-> Sort: NumberOfCrimes DESC, limit input to 15 row(s) per chunk (actual time=1083.373..1083.374 rows=15 loops=1)
-> Table scan on <temporary> (actual time=1083.342..1083.350 rows=30 loops=1)
-> Aggregate using temporary table (actual time=1083.340..1083.340 rows=30 loops=1)
-> Nested loop inner join (cost=130092.05 rows=304235) (actual time=575.655..1000.059 rows=91482 loops=1)
-> Table scan on V (cost=23609.65 rows=234734) (actual time=0.043..75.374 rows=240614 loops=1)
-> Filter: (year(Los_Angeles_Crime_Data.R.DateOcc) = 2021) (cost=0.32 rows=1) (actual time=0.003..0.004 rows=0 loops=240614)
-> Index lookup on R using VictimId (VictimId=Los_Angeles_Crime_Data.V.VictimId) (cost=0.32 rows=1) (actual time=0.003..0.003 rows=1 loops=240614)
```

01:44:23 EXPLAIN ANALYZE SELECT V.Sex, CASE WHEN V.Age BETWEEN 0 AND 10 THEN '0-10' W... 1 row(s) returned 1.109 sec / 0.000 sec



By checking the analysis result, we can find that these two indexes on Victim.Age and Victim.Sex doesn't help to improve this query. For the index on Victim.Age, it doesn't participate in the query process and

cannot improve the renaming efficiency, while for the index on Victim.Sex, it cannot help the group by operation.

Summary:

Since these index designs cannot help to improve the query performance, we decide to use the initial design without any additional indexes for this query.