Index Analysis:

Query 2:

```
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;
```

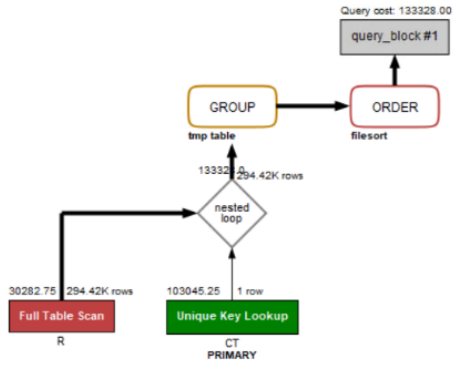
Initial performance:

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		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=133328.00 rows=294415) (actual time=0.107..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..155.893 rows=197212 loops=1)
                        -> Table scan on R  (cost=30282.75 rows=294415) (actual time=0.090..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
```

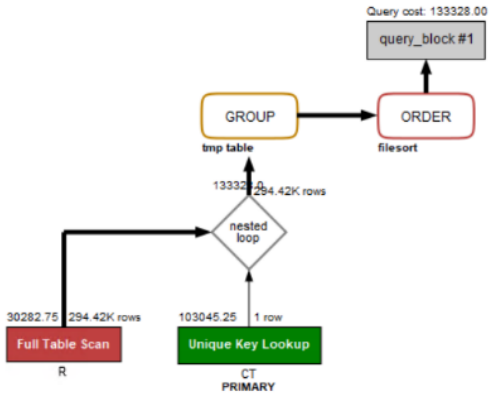


Add index #1 on Record.DateOcc:

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	NULL	NULL		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
Record	1	idx_dateocc	1	DateOcc	A	583	NULL	NULL	YES	BTREE			YES	NULL

```
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=0.063..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



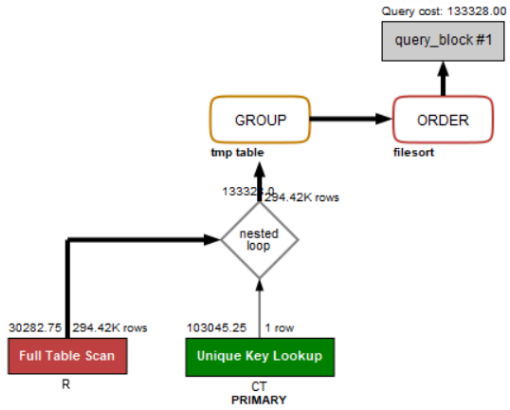
By checking the analysis result, we can find that the new index on Record.DateOcc doesn't help to improve this query. One possible reason is that the query is using "YEAR(R.DateOcc) = 2020" to select and group data entries. However, the index is created based on the date. When the query tries to filter 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	NULL	NULL		BTREE			YES	NULL
CrimeType	1	idx_crimedesc	1	CrimeTypeDesc	A	126	NULL	NULL	YES	BTREE			YES	NULL

```
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.

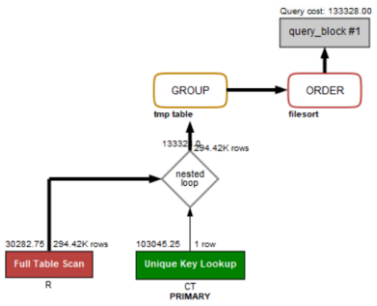
Add index #3 on both Record.DateOcc and CrimeType.CrimeTypeDesc:

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	NULL	NULL		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
Record	1	idx_dateocc	1	DateOcc	A	583	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	Expressi
CrimeType	0	PRIMARY	1	CrimeTypeId	A	126	NULL	NULL		BTREE			YES	NULL
CrimeType	1	idx_crimedesc	1	CrimeTypeDesc	A	126	NULL	NULL	YES	BTREE			YES	NULL

```
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.

Query 3:

```
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;
```

Index Analysis:

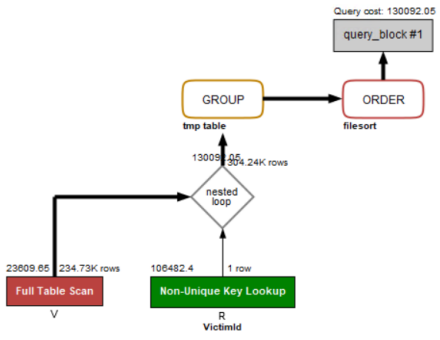
Initial performance:

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		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
Victim	0	PRIMARY	1	VictimId	A	234734	NULL	NULL		BTREE			YES	NULL

```
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
```

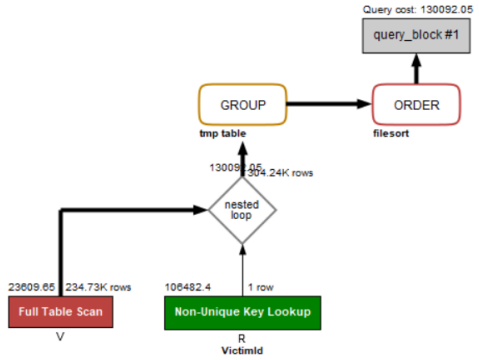


Add index #1 on Victim.Age;

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	NULL	NULL		BTREE			YES	NULL
Victim	1	idx_age	1	Age	A	83	NULL	NULL	YES	BTREE			YES	NULL

```
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



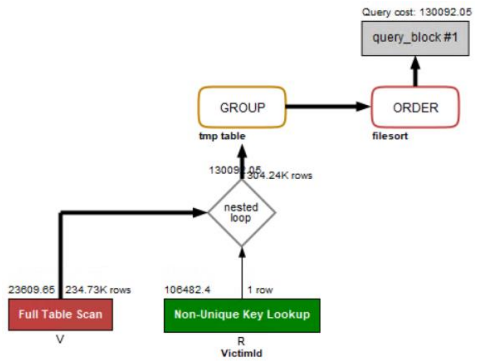
By checking the analysis result, we can find that the new index on Victim.Age doesn't help to improve this query. One possible reason is that this attribute is not used to select the data entries. Instead, its value is used to rename the AgeGroup as an output attribute. So this attribute actually doesn't 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	NULL	NULL		BTREE			YES	NULL
Victim	1	idx_sex	1	Sex	A	4	NULL	NULL	YES	BTREE			YES	NULL

```
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

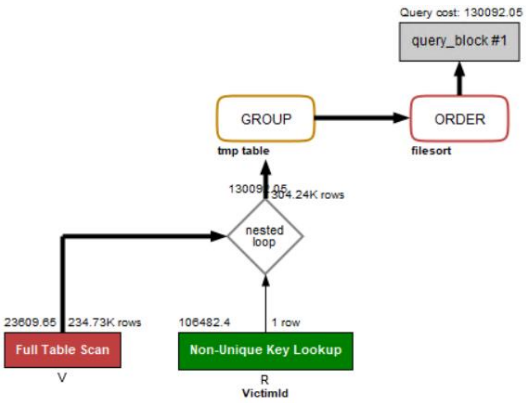
query. One possible reason is that since the results are grouped by two attributes, a table scan is always required for the AgeGroup output attribute. Even though we add a index on the Sex, we cannot eliminate the table scan cost.

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	NULL	NULL		BTREE			YES	NULL
Victim	1	idx_sex	1	Sex	A	4	NULL	NULL	YES	BTREE			YES	NULL
Victim	1	idx_age	1	Age	A	83	NULL	NULL	YES	BTREE			YES	NULL

```
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.