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;

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

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

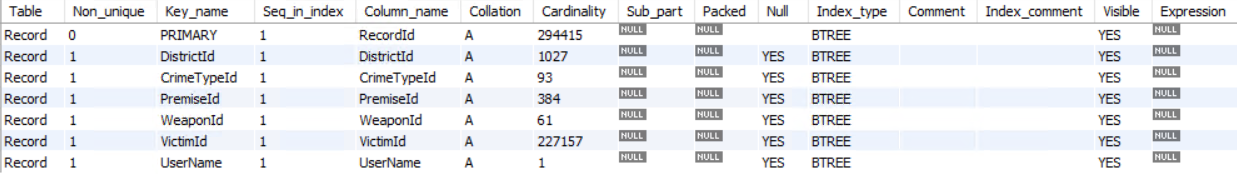
LIMIT 15;

图形用户界面, 文本, 应用程序

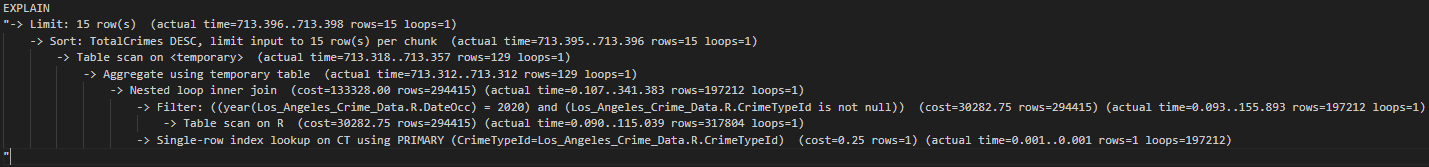
描述已自动生成

Index Analysis:

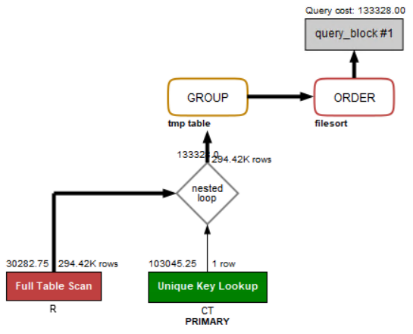
Initial performance:



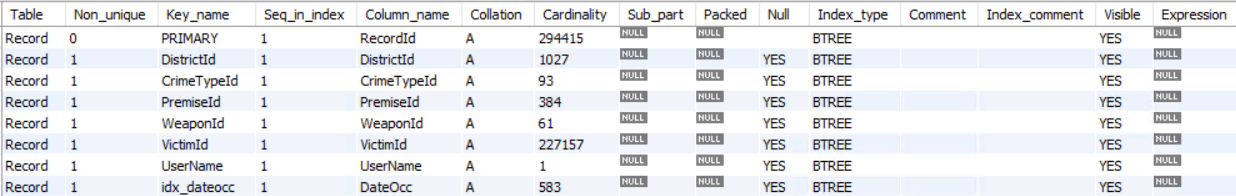


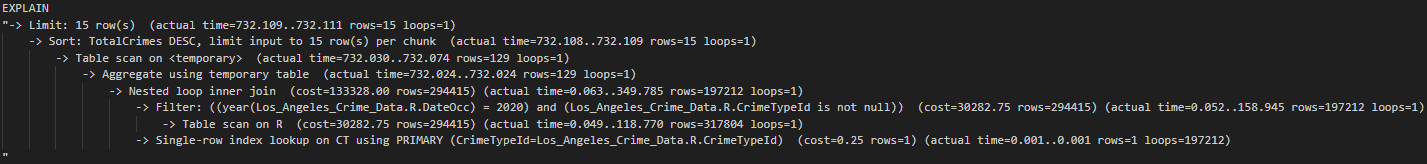




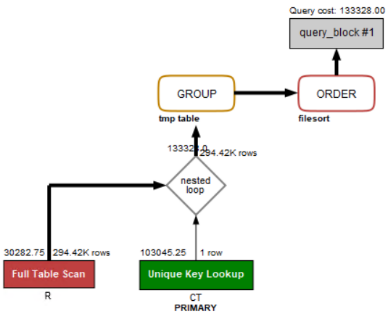


Add index #1 on Record.DateOcc:



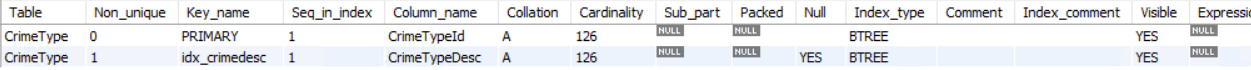


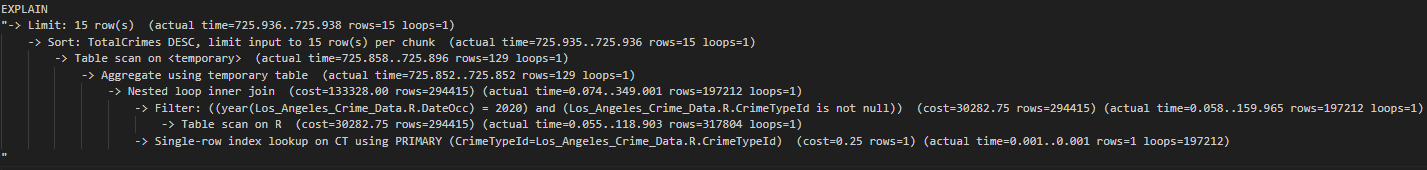




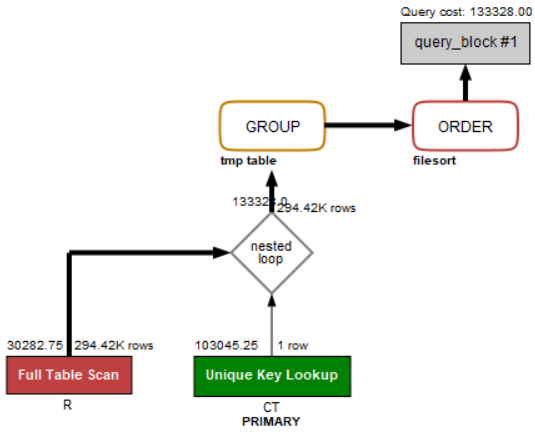
By checking the analysis result, we can find that the new index on Record.DateOcc doesn’t help to improve this query. One possible reason is that the query is using “YEAR(R.DateOcc) = 2020” to select and group data entries. However, the index is created based on the date. When the query tries to filter 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:



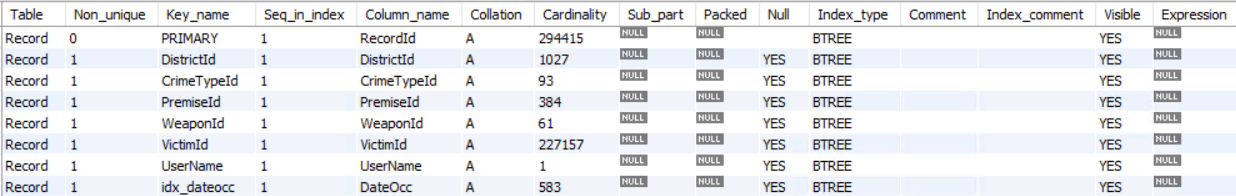


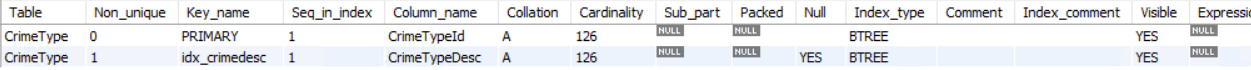


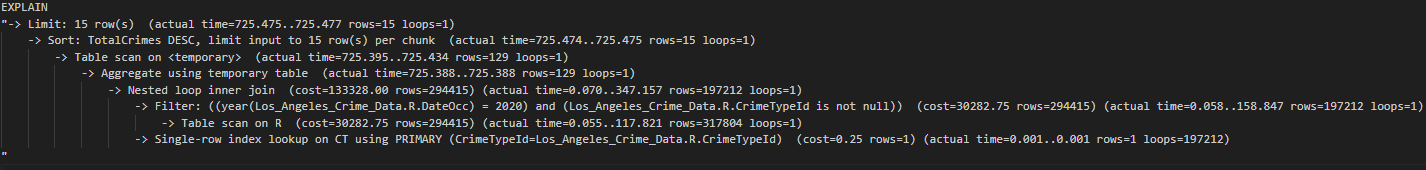


By checking the analysis result, we can find that the new index on CrimeType.CrimeTypeDesc doesn’t help to improve this query. This is because the CrimeTypeDesc attribute is a string. Adding an index on it won’t help to improve the grouping efficiency since each group has exactly one unique string.

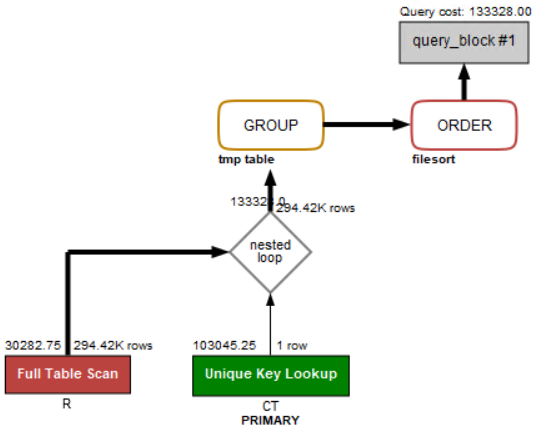
Add index #3 on both Record.DateOcc and CrimeType.CrimeTypeDesc:











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

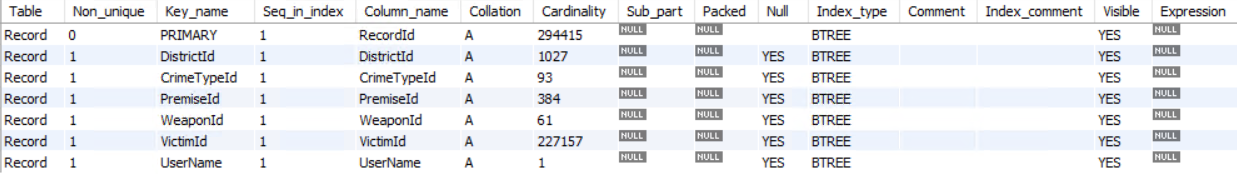
LIMIT 15;

图形用户界面, 文本, 应用程序

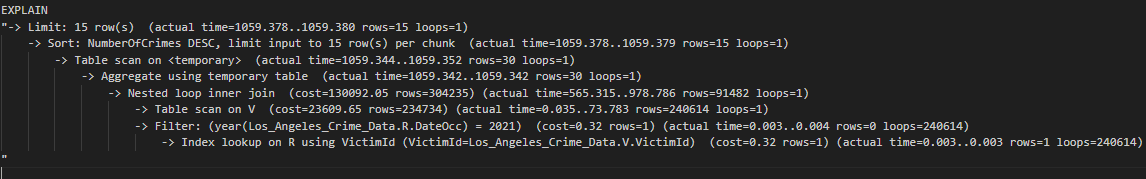
描述已自动生成

Index Analysis:

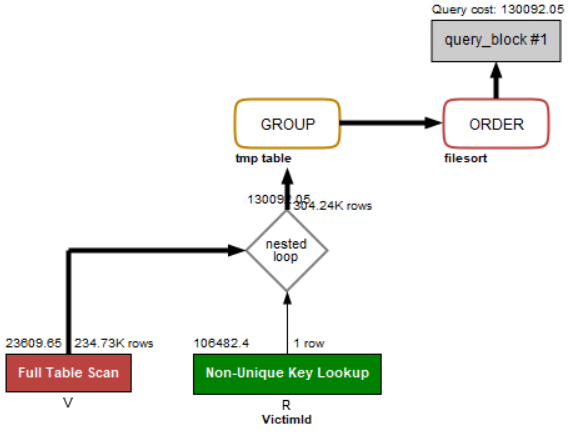
Initial performance:



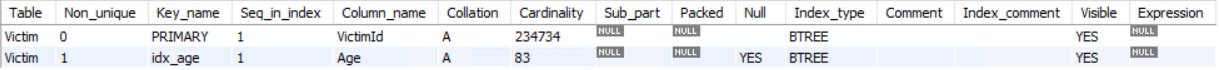


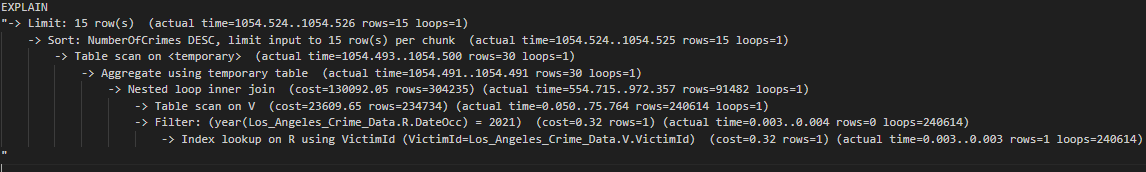




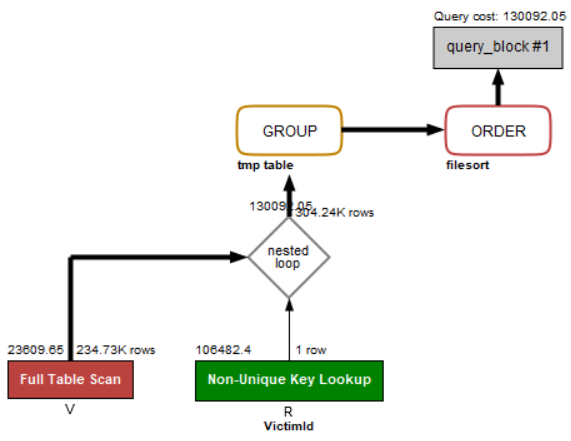


Add index #1 on Victim.Age;





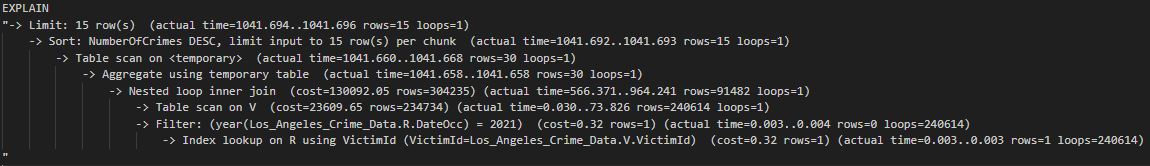




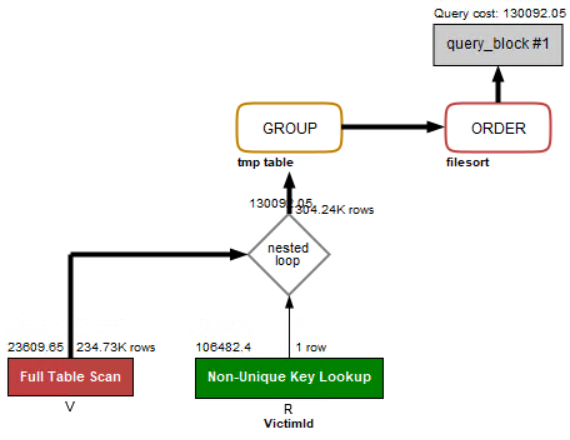
By checking the analysis result, we can find that the new index on Victim.Age doesn’t help to improve this query. One possible reason is that this attribute is not used to select the data entries. Instead, its value is used to rename the AgeGroup as an output attribute. So this attribute actually doesn’t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

Add index #2 on Victim.Sex:



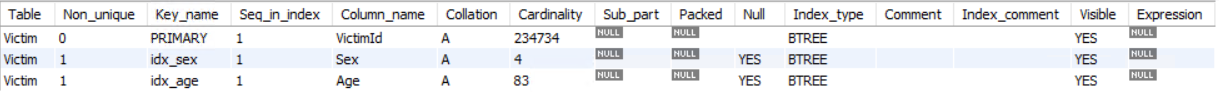


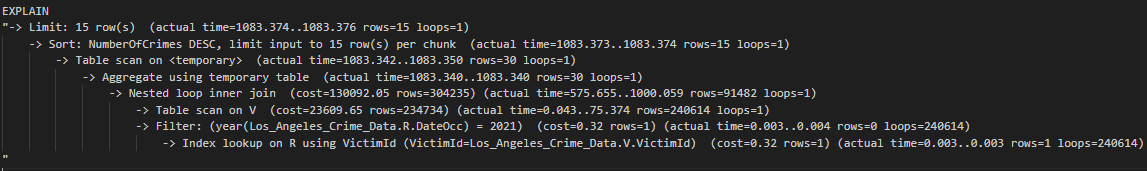




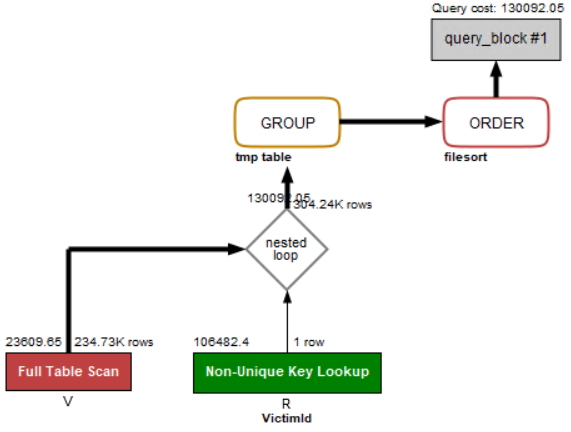
By checking the analysis result, we can find that the new index on Victim.Sex doesn’t help to improve this query. One possible reason is that since the results are grouped by two attributes, a table scan is always required for the AgeGroup output attribute. Even though we add a index on the Sex, we cannot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

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









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.