|  |
| --- |
|  |

**LAB 8: End-to-End Telecom Data Query Optimization**

|  |  |
| --- | --- |
| **Track** | Data & AI |
| **Role** | Software Development - Backend |
| **Level** | Level 2 |
| **Course** | Intermediate SQL |
| **Scope** | * End-to-end query optimization. * Index creation and usage. * Query refactoring techniques. * Performance analysis with EXPLAIN. * Telecom-specific query optimization. |
| **Prerequisite** | * Basic SQL knowledge. * Familiarity with PostgreSQL and MySQL. * Understanding of indexing and Experience with subqueries. |

**Objective**

To optimize complex queries in a telecom dataset using indexing, query refactoring, and performance analysis tools such as EXPLAIN to improve query execution time.

**Tasks**

**Dataset (also available in resources): Telecom Insights.xlsx**

1. **Identifying Slow Queries**
   * Find and Optimize a Slow Query
2. **Using Indexes for Query Optimization**
   * Create and Use a Composite Index
3. **Query Refactoring for Better Performance**
   * Optimize a Query Using Subqueries
4. **Utilizing Partial Indexes for Specific Queries**
   * Use a Partial Index to Optimize Queries for Long Calls
   * Create a Partial Index for Specific Rows
5. **Analyzing and Tuning Query Performance**
   * Analyze Query Performance with EXPLAIN
6. **Combining Indexes and Query Refactoring**
   * Optimize Sorting and Filtering Queries
7. **Optimizing Complex Nested Queries**

* Optimized Query (Using HAVING Clause)
* Optimize a Nested Query Using a Common Table Expression (CTE)

1. **Using Temporary Tables for Performance Improvement**

* Create A temporary Table

1. **Advanced Telecom Data Query Optimization with Indexing and Query Refactoring**

* Implement Indexes on Frequently Used WHERE Clauses
* Run ANALYZE on Telecom Tables
* Use SELECT with LIMIT for Paginated Data Retrieval

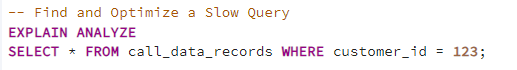
**Outcomes**

* Faster query execution.
* Efficient use of indexes.
* Reduced query complexity.
* Improved performance for nested queries.
* Better understanding of telecom data optimization.

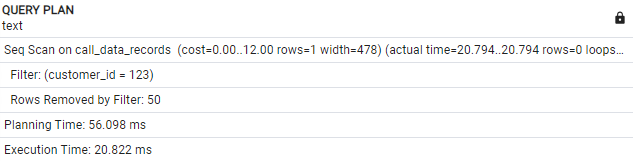
**Solution**

1. **Advanced Subquery Implementation for Telecom Data**

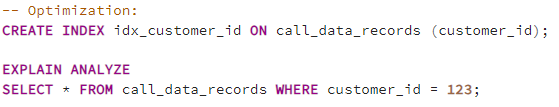
* Identifying Slow Queries
  + Use the EXPLAIN ANALYZE command to identify a slow-running query in the call\_data\_records table, then refactor or index the query to improve performance.

****

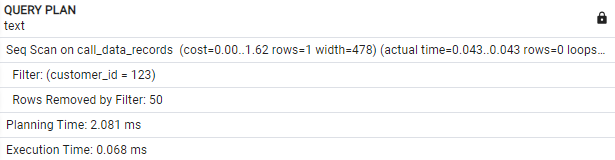
* Output:



* + Optimization:

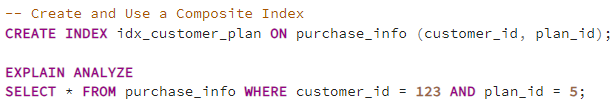


* + - Output:

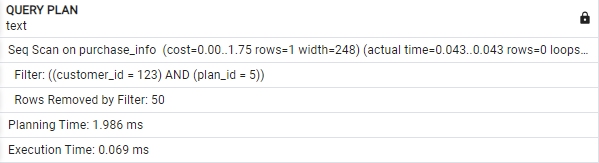


1. **Using Indexes for Query Optimization**

* Create and Use a Composite Index
  + Create a composite index on the purchase\_info table to optimize queries that filter by both customer\_id and plan\_id.
  + Run a query that filters both by customer and plan to test the performance with the new index.

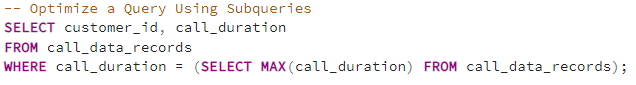


* Output:



1. **Query Refactoring for Better Performance**

* Optimize a Query Using Subqueries
  + Refactor a query that uses a subquery to retrieve customers who have made the longest calls, and optimize it by indexing or removing unnecessary subqueries.
  + Before Optimization



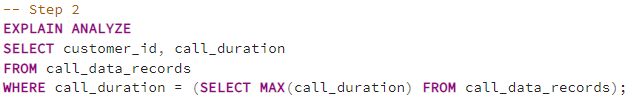
* Output:



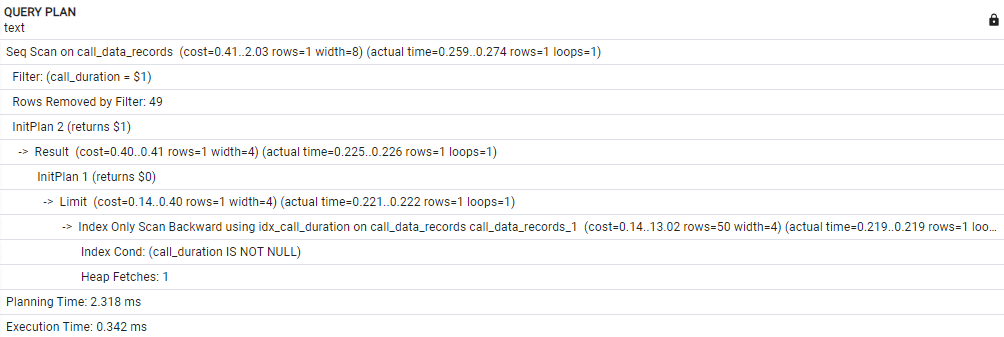
* Optimization:
  + **Step 1**: Create an index on the call\_duration column to optimize the subquery.



* + **Step 2**: Re-run the query and analyze its performance.

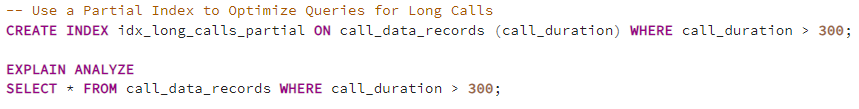


* Output:

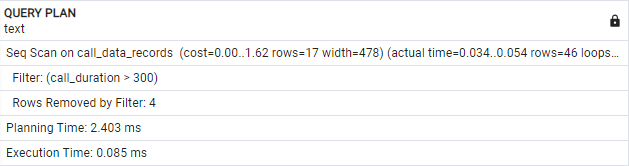


1. **Utilizing Partial Indexes for Specific Queries**

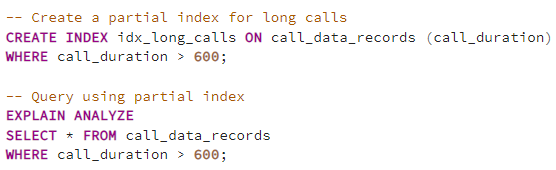
* Use a Partial Index to Optimize Queries for Long Calls
  + Use a partial index to optimize queries that search for call durations longer than 5 minutes.



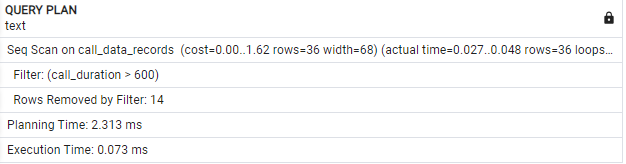
* Output:



* create a partial index on calls that lasted more than 10 minutes (600 seconds).

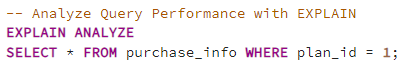


* Output:

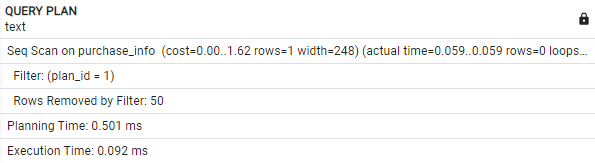


1. **Analyzing and Tuning Query Performance**

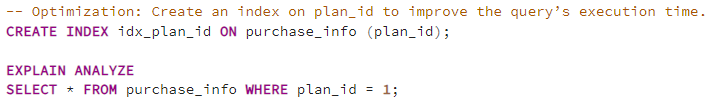
* Analyze Query Performance with EXPLAIN
  + Use EXPLAIN and EXPLAIN ANALYZE to identify bottlenecks in query performance, then apply indexing or query rewriting techniques to improve execution time.



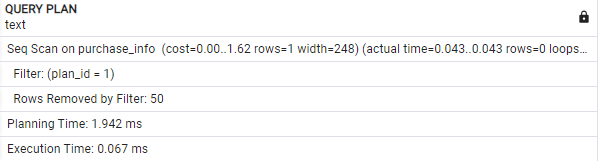
* Output:



* + **Optimization**: Create an index on plan\_id to improve the query’s execution time.



* Output:



1. **Combining Indexes and Query Refactoring**

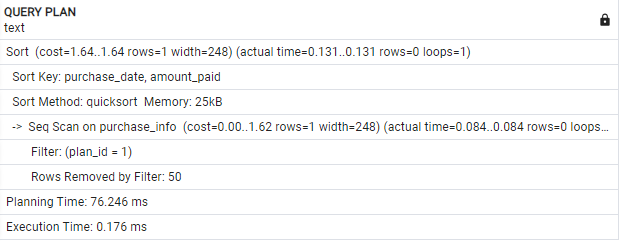
* Optimize Sorting and Filtering Queries
  + Optimize queries that involve both filtering and sorting by creating an index that covers both the purchase\_date and amount\_paid columns.



* + Test the performance of a query that filters by a plan and sorts by purchase\_date and amount\_paid.

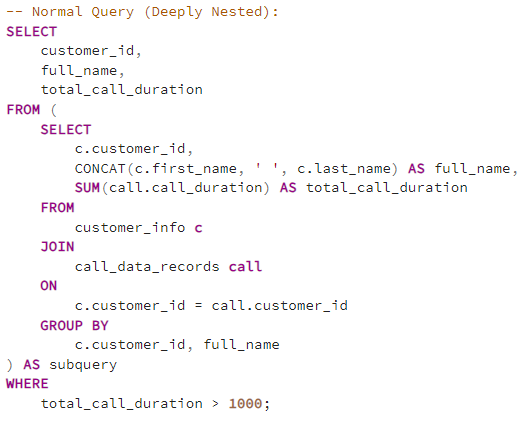


* Output:

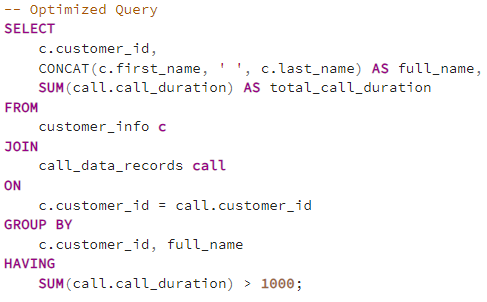


1. **Optimizing Complex Nested Queries**

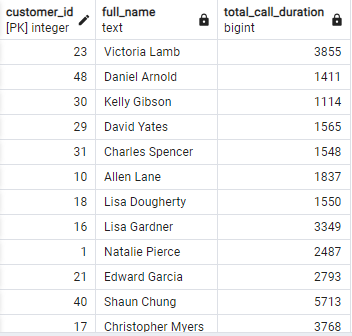
* Normal Query (Deeply Nested):
  + optimize complex queries in a telecom database by avoiding nested subqueries and using the HAVING clause to filter aggregated results.



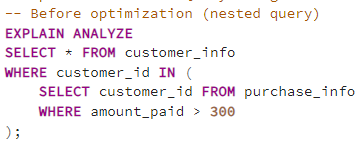
* + Optimized Query



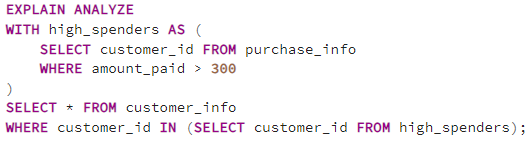
* Output:



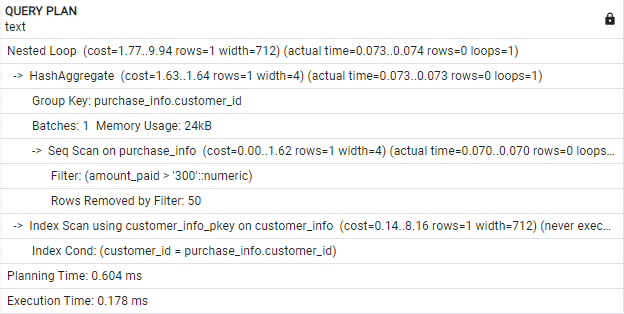
* nested query that retrieves customers who have spent more than $300 using a CTE to improve readability and performance.



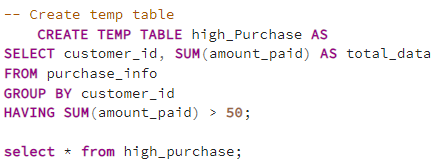
* + Optimized Query



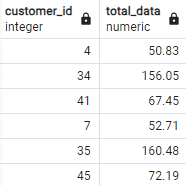
* Output:



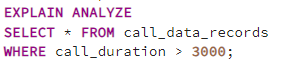
1. **Using Temporary Tables for Performance Improvement**
   * + Create Temporary table
       - Use temporary tables to store intermediate results for queries that retrieve customers with high purchase, improving performance when multiple queries are executed on the same data.



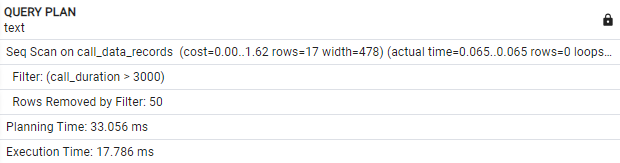
* Output:



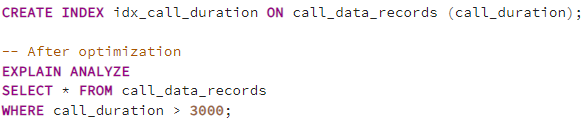
1. **Advanced Telecom Data Query Optimization**
   * + To facilitate the analysis of long-duration calls, an index will be created on the call\_duration column, as queries are often filtered by call\_duration in the call\_data\_records table.
       - Before Optimization:



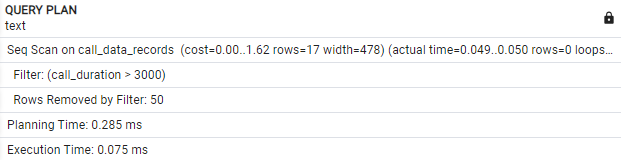
* Output:



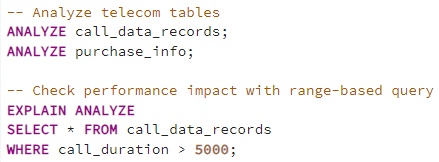
* + - * After Optimization:



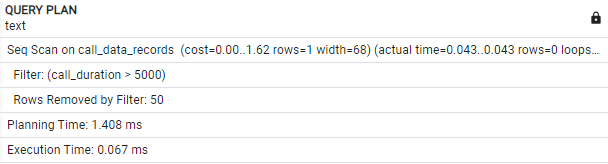
* Output:



* + - To improve the performance of range-based queries, we will run the ANALYZE command on key tables to update statistics.



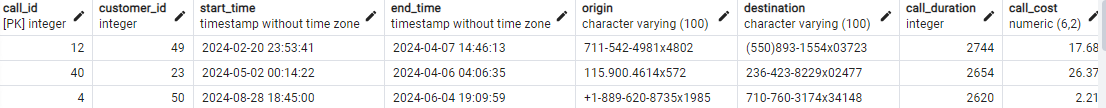
* Output:



* + - Retrieving only a limited set of rows at a time can enhance performance.



* Output:



# (Additional) Task to do

1. Create an index on a frequently used WHERE clause column, and use EXPLAIN to compare query times before and after.
2. Refactor a multi-table join query to reduce unnecessary joins and improve execution time.
3. Run ANALYZE on telecom tables and check how it impacts the performance of range-based queries.
4. Create a partial index for specific rows (e.g., by region or date) and test its impact on query speed.
5. Use EXPLAIN on a nested query, refactor it using a CTE, and compare the performance before and after.