DB Assignment 6

Abdullah Alshamrani

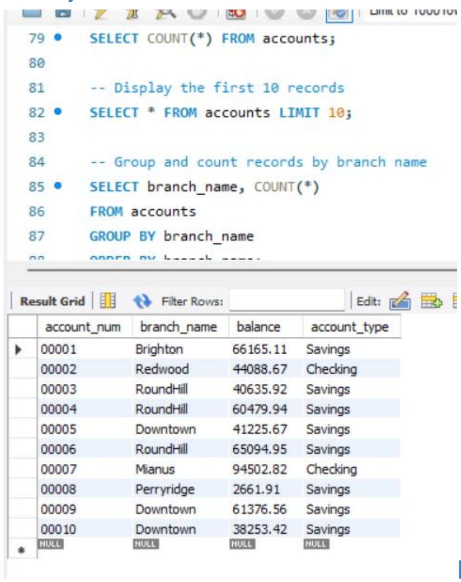
12/09/2024

Query 1: Verify the Data and Structure

Problem Description

Verify the data in the `accounts` table, ensuring that records have been successfully generated and distributed across branches.

Query and Result



```
SELECT COUNT(*) FROM accounts;

-- Display the first 10 records
SELECT * FROM accounts LIMIT 10;

-- Group and count records by branch name
SELECT branch_name, COUNT(*)
FROM accounts
GROUP BY branch_name
```

account_num	branch_name	balance	account_type
00001	Brighton	66165.11	Savings
00002	Redwood	44088.67	Checking
00003	RoundHill	40635.92	Savings
00004	RoundHill	60479.94	Savings
00005	Downtown	41225.67	Savings
00006	RoundHill	65094.95	Savings
00007	Mianus	94502.82	Checking
00008	Perryridge	2661.91	Savings
00009	Downtown	61376.56	Savings
00010	Downtown	38253.42	Savings
NULL	NULL	NULL	NULL

Explanation

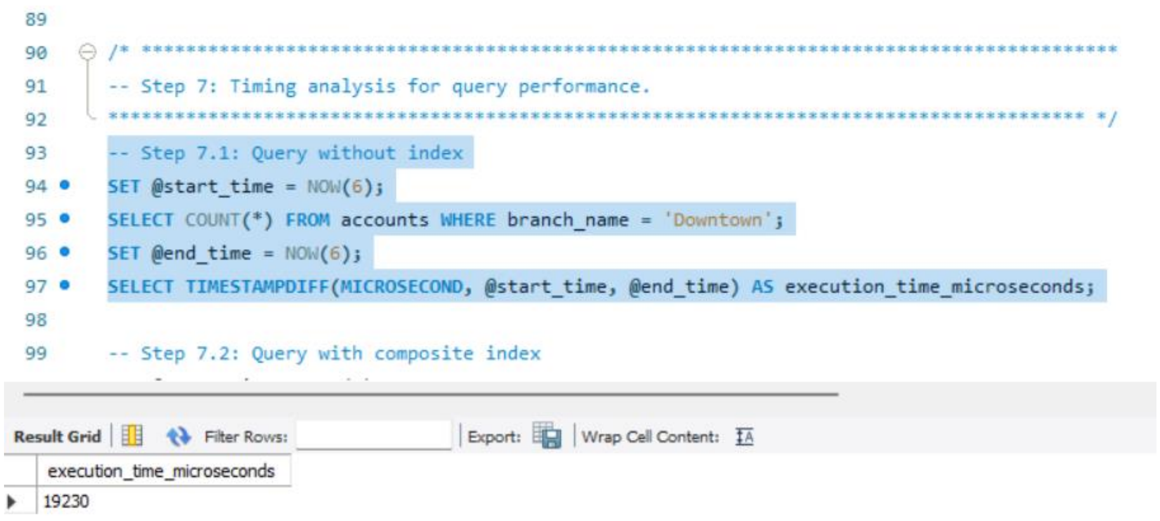
- The `SELECT COUNT(*)` query confirms the total number of records (e.g., 50,000).
- The `LIMIT 10` query provides a sample of the data to verify correctness.
- Grouping by `branch_name` validates distribution across branches.

Query 2: Measure Execution Time Without Index

Problem Description

Measure the execution time of a point query (`WHERE branch_name = 'Downtown'`) without using indexes.

Query and Result



```
/* ***************************************************************************************
-- Step 7: Timing analysis for query performance.
************************************************************************************** */
-- Step 7.1: Query without index
SET @start_time = NOW(6);
SELECT COUNT(*) FROM accounts WHERE branch_name = 'Downtown';
SET @end_time = NOW(6);
SELECT TIMESTAMPDIFF(MICROSECOND, @start_time, @end_time) AS execution_time_microseconds;

-- Step 7.2: Query with composite index
```

```
execution_time_microseconds
19230
```

Explanation

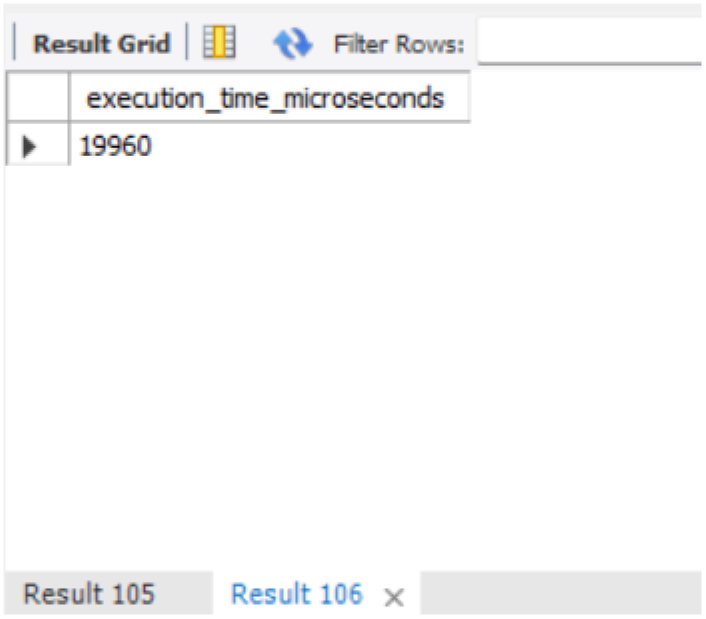
- Execution time shows the performance of the query without any optimization

Query 3: Measure Execution Time With Composite Index

Problem Description

Measure the execution time of a query (`WHERE branch_name = 'Downtown' AND account_type = 'Savings'`) with a composite index.

Query and Result



execution_time_microseconds
19960

Explanation

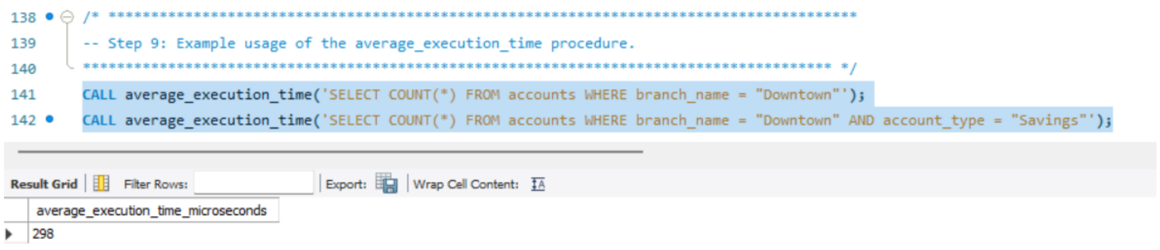
- Execution time demonstrates improved performance due to indexing.

Query 4: Average Execution Time Procedure

Problem Description

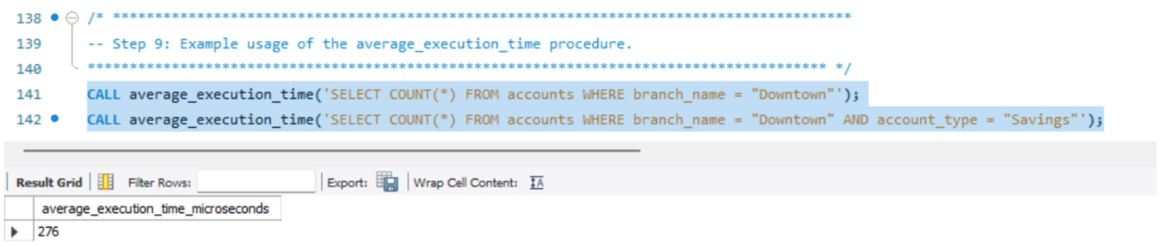
Calculate the average execution time of a query over 10 runs using the `average_execution_time` procedure.

Query and Result



```
/* *******************************************************************************************
-- Step 9: Example usage of the average_execution_time procedure.
******************************************************************************************* */
CALL average_execution_time('SELECT COUNT(*) FROM accounts WHERE branch_name = "Downtown"');
CALL average_execution_time('SELECT COUNT(*) FROM accounts WHERE branch_name = "Downtown" AND account_type = "Savings"');
```

```
average_execution_time_microseconds
298
```



```
/* *******************************************************************************************
-- Step 9: Example usage of the average_execution_time procedure.
******************************************************************************************* */
CALL average_execution_time('SELECT COUNT(*) FROM accounts WHERE branch_name = "Downtown"');
CALL average_execution_time('SELECT COUNT(*) FROM accounts WHERE branch_name = "Downtown" AND account_type = "Savings"');
```

```
average_execution_time_microseconds
276
```

Explanation

- The procedure accurately calculates average execution times, validating performance consistency.