Title: DB Assignment 6

Name: Aidan Elm

Date: 2024-12-10

Question 1

```
delimiter $$
create procedure generate_accounts()
begin
        create table accounts (
        account_num int(5),
        branch_name varchar(255),
        balance decimal(10, 2),
        account_type varchar(50)
        );
end $$
delimiter ;

call generate_accounts();
```

This question, for the most part, follows what was in the slides from Module 3. The create table is simply wrapped in a procedure called generate_accounts.

Question 2

```
delimiter $$
create procedure populate_accounts(record_count int)
begin
        declare i int;
        set i = 1;
        while i <= record_count do
        insert into accounts (account_num, branch_name, balance, account_type)
        values (i, concat('branch_', i), 1000.00, 'Checking');
        set i = i + 1;
```

```
        end while;
end $$
delimiter ;

call populate_accounts(5000);

-- Computer couldn't handle any of these :(
-- call populate_accounts(50000);
-- call populate_accounts(100000);
-- call populate_accounts(150000);
```

Again, in this one, we are creating a procedure. This time, we are inserting into the table we created in Problem 1. Note: My computer was only able to handle 5,000 rows of this, which will affect things in the next few questions. (I'm sure this has *nothing* to do with how efficiently I wrote that insert statement… Right, Mike?)

Question 3

```
create index index_branch_name on accounts (branch_name);
create index index_account_type on accounts (account_type);
create index index_account_type_balance on accounts (account_type, balance);
```

Created the indexes on different columns as mentioned in the question.

Question 4

```
select count(*) from accounts where balance = 1000.00;
select count(*) from accounts where balance between 500.00 and 25000.00;
```

The first query is a point query, while the second is a range query.

Questions 5-8

```
delimiter $$
create procedure measure_avg_execution_time(query_text varchar(1000))
begin
	declare start_time datetime;
	declare end_time datetime;
	declare total_time bigint;
	declare avg_time bigint;
	declare i int;

	set i = 1;
	set total_time = 0;

	-- Execute the query 10 times
	while i <= 10 do
	set start_time = now();
	set @dynamic_query = query_text; -- Apparently this is a MySQL thing?
	prepare stmt from @dynamic_query;
	execute stmt;
	deallocate prepare stmt;
	set end_time = now();

	-- Calculate execution time and add to total
	set total_time = total_time + TIMESTAMPDIFF(MICROSECOND, start_time,
end_time); -- From Dr. F instructions
	set i = i + 1;
	end while;

	-- Return average execution time and total execution time
```

```
        set avg_time = total_time / 10;
        select avg_time, total_time;
end $$
delimiter ;

-- With indexing
call measure_avg_execution_time('select count(*) from accounts where account_type = "Checking" and balance = 1000.00;');
call measure_avg_execution_time('select count(*) from accounts where branch_name = "branch_1" and balance = 1000.00;');
call measure_avg_execution_time('select count(*) from accounts where account_type = "Checking" and balance between 500.00 and 2500.00;');
call measure_avg_execution_time('select count(*) from accounts where branch_name = "branch_1" and balance between 500.00 and 2500.00;');

-- Without indexing
alter table accounts drop index index_branch_name;
alter table accounts drop index index_account_type;
alter table accounts drop index index_account_type_balance;
call measure_avg_execution_time('select count(*) from accounts where account_type = "Checking" and balance = 1000.00;');
call measure_avg_execution_time('select count(*) from accounts where branch_name = "branch_1" and balance = 1000.00;');
call measure_avg_execution_time('select count(*) from accounts where account_type = "Checking" and balance between 500.00 and 2500.00;');
call measure_avg_execution_time('select count(*) from accounts where branch_name = "branch_1" and balance between 500.00 and 2500.00;');
```

- **For Question 5: As mentioned above, I was limited to 5000 queries on my computer.**
- **For Question 6: Combined with Question 7**

- **For Question 7: Created a procedure as in the previous questions to execute the queries from Question 6. The queries for Question 6 are below this procedure, first with indexing and then without.**
- **For Question 8: As mentioned, the maximum number of rows I was able to create was 5,000. All of the times that were returned were zero seconds.**