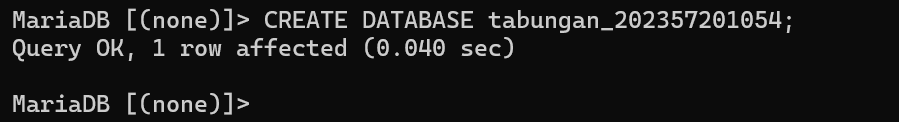
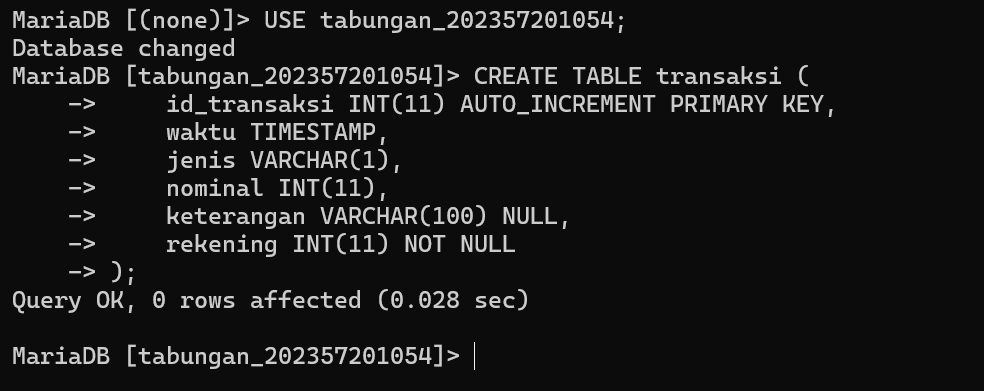
**PRAKTIKUM BASIS DATA II**

**TRANSACTION A. PERSIAPAN**

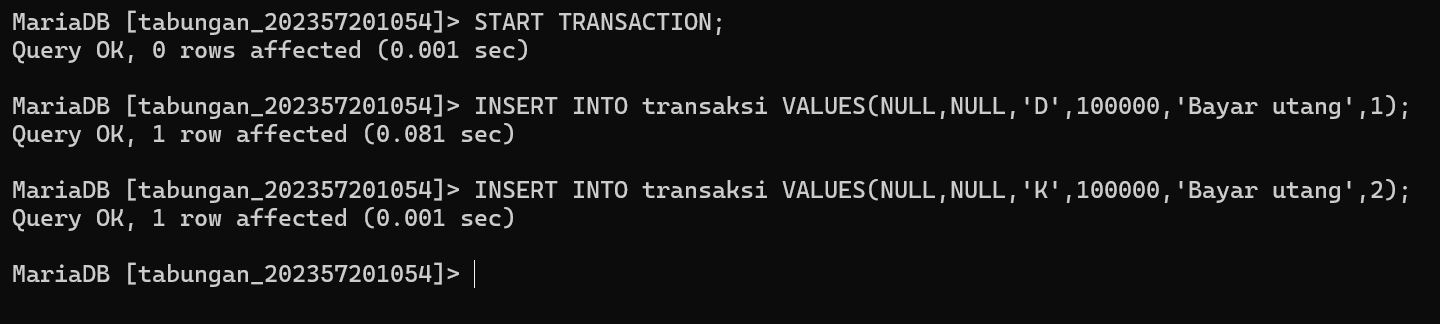
1. Buat basis data dengan nama ‘tabungan’



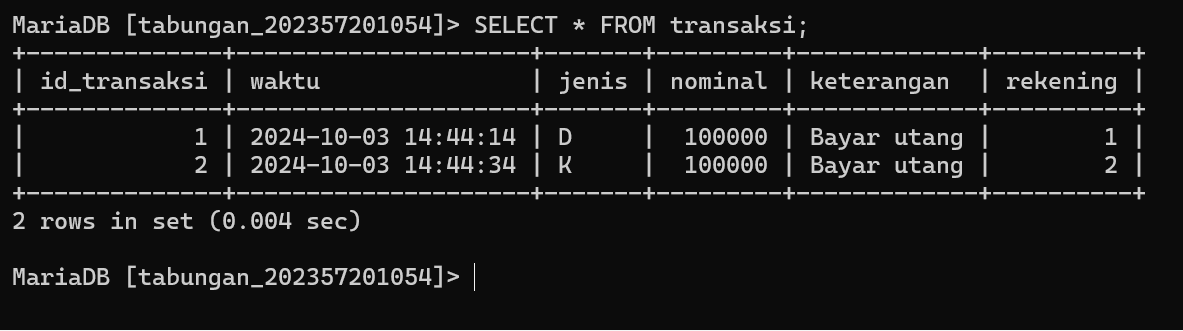
1. Buat tabel ‘transaksi’



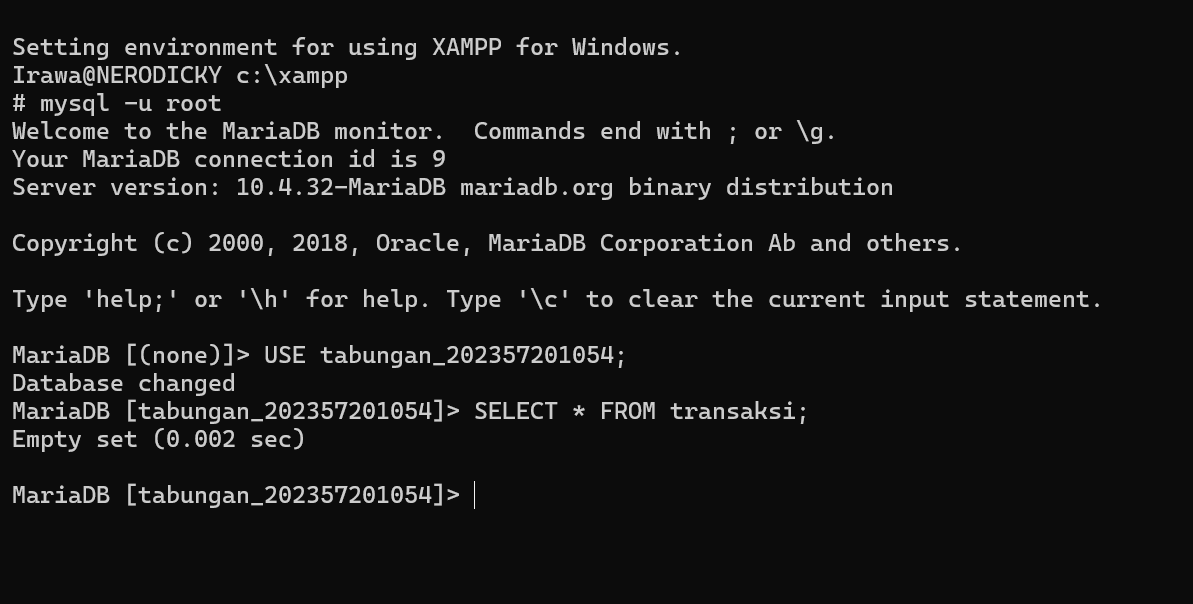
1. **UJI COBA TRANSAKSI (COMMIT)** 
   * 1. Jalankan perintah **INSERT** di bawah ini:



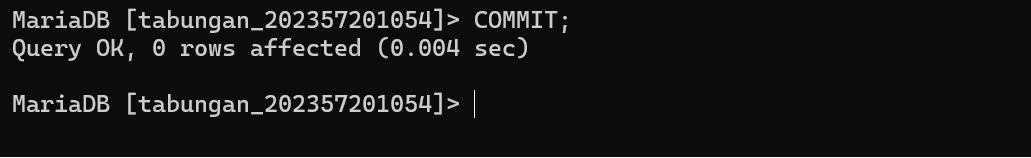
* + 1. Cek isi tabel **transaksi**, perhatikan kolom **waktu**



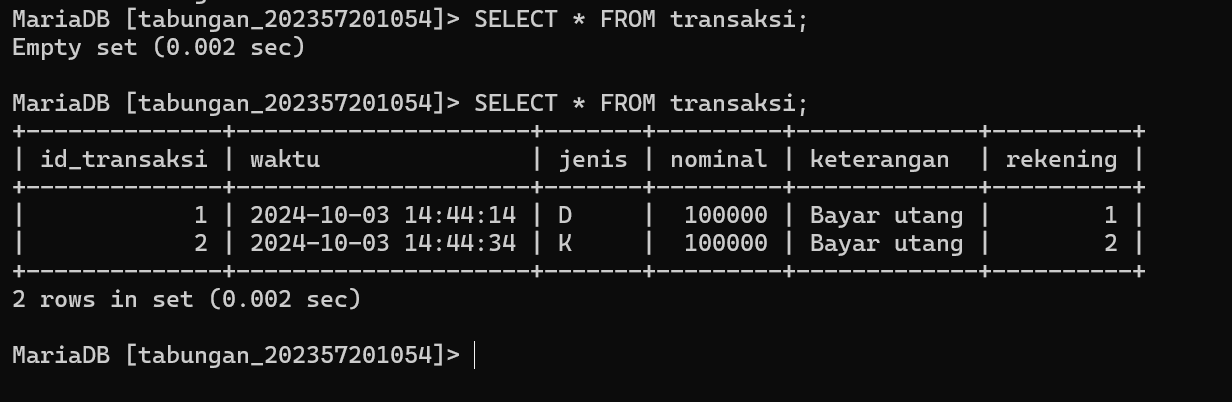
* + 1. Buka jendela baru shell/command promt, buka sesi baru mysql dan cek isi tabel **transaksi** pada sesi baru



* + 1. Kembalilah ke sesi pertama mysql Anda, kemudian jalankan perintah di bawah ini:



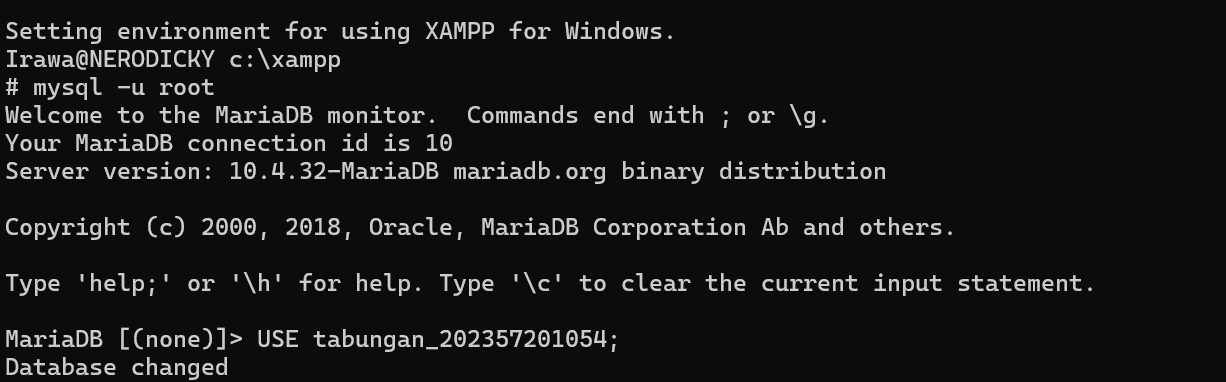
* + 1. Sekarang pindah ke sesi kedua mysql dan cek lagi isi tabel **transaksi**



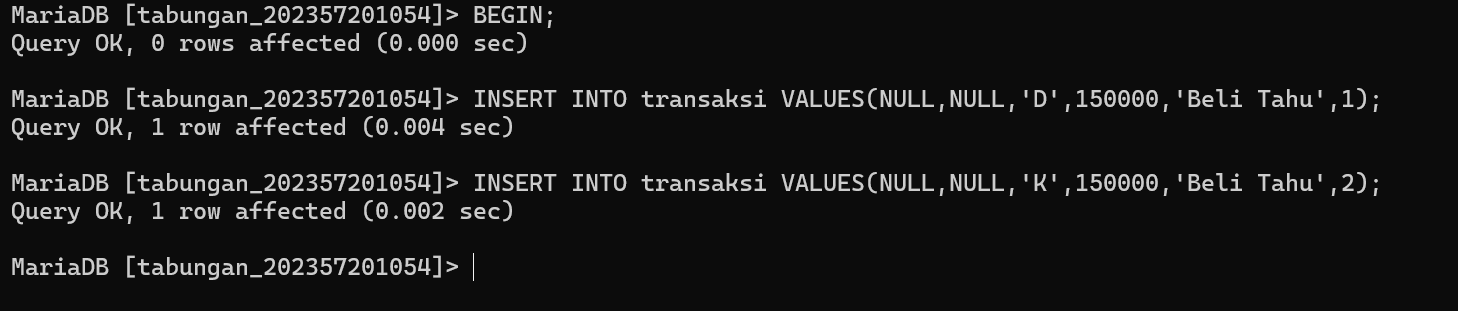
Apa yang bisa Anda simpulkan dari uji coba ini?

Dalam uji coba transaksi (COMMIT) pada MariaDB, transaksi dimulai dengan perintah START TRANSACTION; dan diikuti oleh dua operasi INSERT yang menambahkan data ke tabel transaksi. Hasilnya, kedua baris data hanya terlihat dalam sesi yang sama hingga transaksi tersebut di-commit. Ketika membuka sesi baru sebelum commit, tabel transaksi tampak kosong, menegaskan bahwa perubahan hanya terlihat di sesi yang mengontrol transaksi. Setelah menjalankan COMMIT; di sesi asli, kedua baris data menjadi terlihat di semua sesi, memverifikasi bahwa perubahan sudah tersimpan secara keseluruhan.

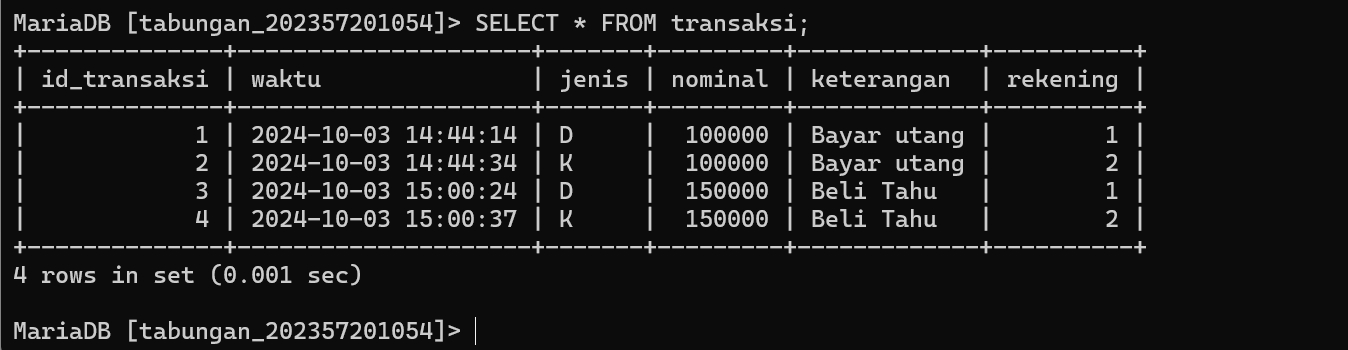
1. **UJI COBA TRANSAKSI (ROLLBACK)** 
   * 1. Buka sesi baru mysql, kemudian pilih basis data **tabungan**



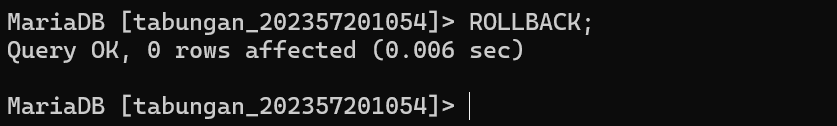
* + 1. Jalankan perintah transaksi di bawah ini:



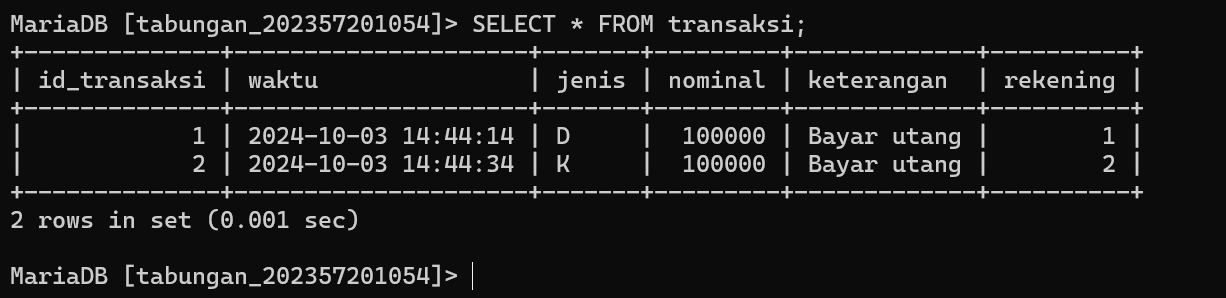
* + 1. Cek isi tabel **transaksi**



* + 1. Jalankan perintah **rollback**

****

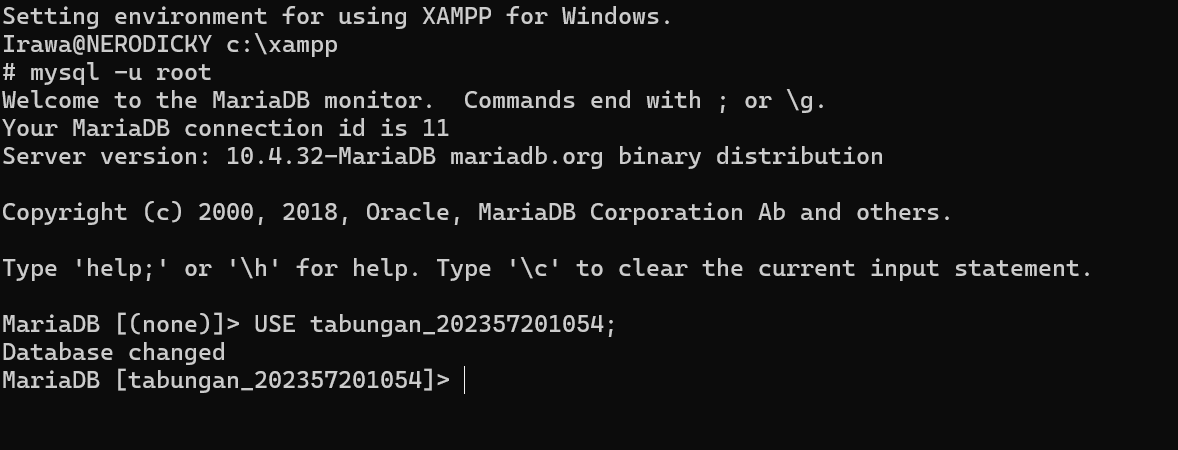
* + 1. Cek isi tabel **transaksi**

****

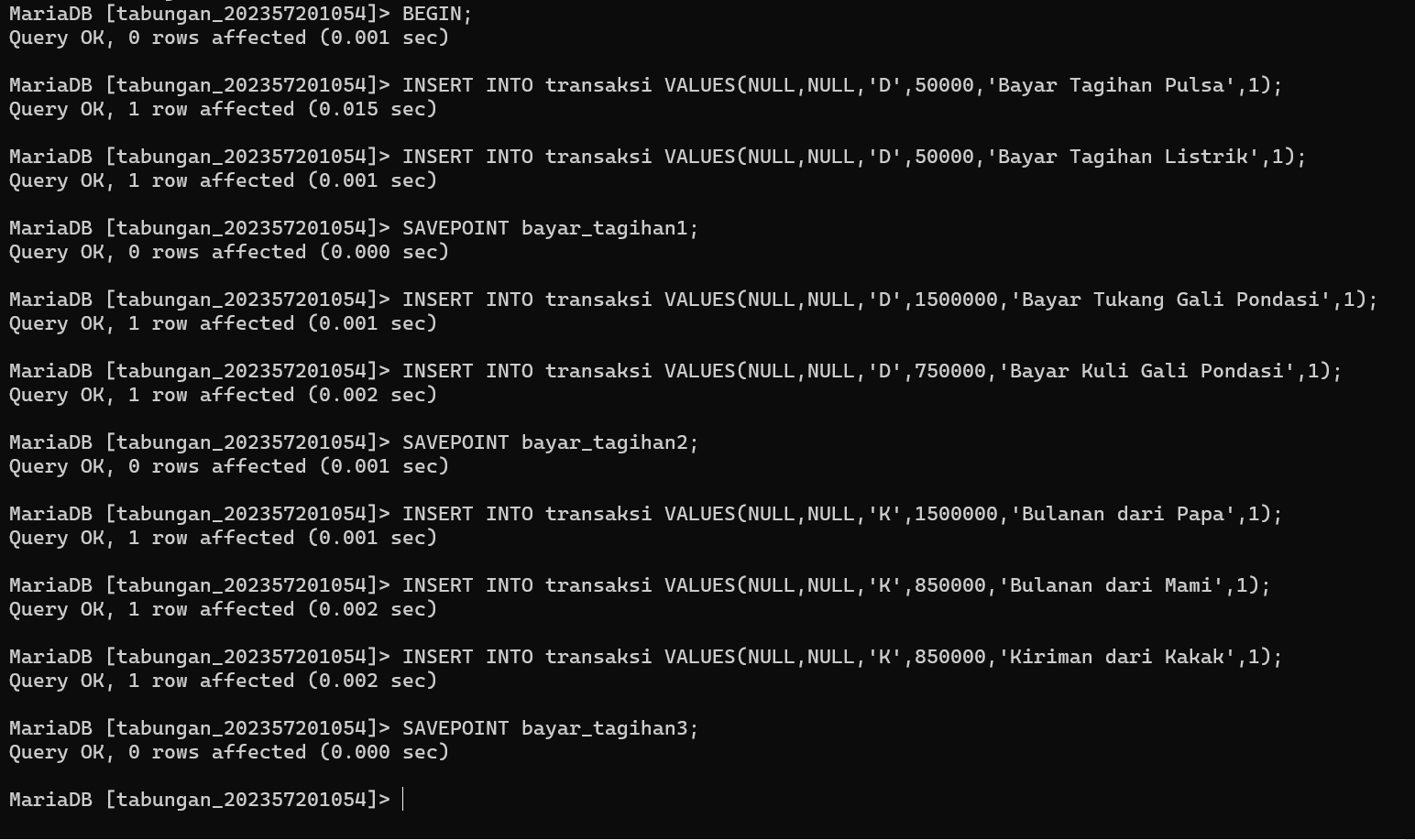
Apa yang bisa Anda simpulkan dari uji coba ini?

Dalam uji coba transaksi (ROLLBACK) menggunakan MariaDB, transaksi dimulai dengan perintah BEGIN, yang menandai awal dari serangkaian operasi yang bertujuan untuk memodifikasi tabel transaksi. Selama sesi ini, dua perintah INSERT secara berhasil menambahkan total empat baris data dengan detail transaksi keuangan ke dalam tabel. Semua empat baris data ini terlihat saat menjalankan query SELECT \* FROM transaksi; yang menunjukkan bahwa data telah berhasil ditambahkan sementara dalam sesi tersebut. Namun, untuk menguji kemampuan mengembalikan kondisi sebelum perubahan, perintah ROLLBACK dijalankan, yang bertujuan untuk membatalkan semua operasi yang belum di-commit. Hasilnya, ketika tabel diperiksa kembali, hanya dua baris data awal yang muncul, menunjukkan bahwa ROLLBACK telah efektif menghapus dua baris yang baru ditambahkan dan memulihkan tabel ke keadaan sebelum transaksi dimulai. Uji coba ini menggambarkan pentingnya mekanisme ROLLBACK dalam mengelola transaksi untuk mencegah perubahan yang tidak diinginkan menjadi permanen, serta memastikan stabilitas dan keandalan data dalam basis data.

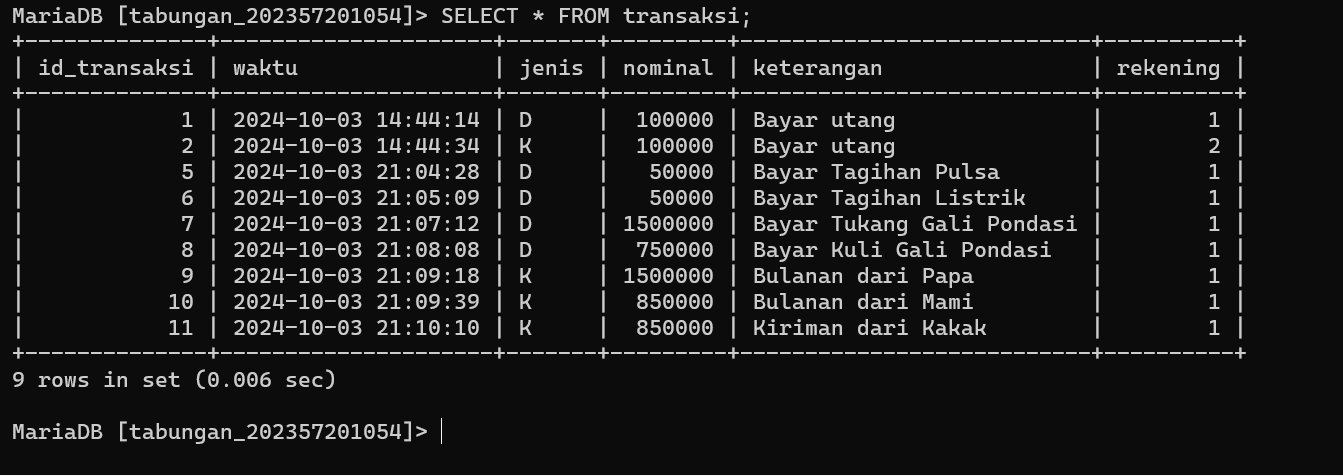
1. **UJI COBA TRANSAKSI (SAVEPOINT)** 
   1. Buka sesi baru mysql dan pilih basis data **tabungan**



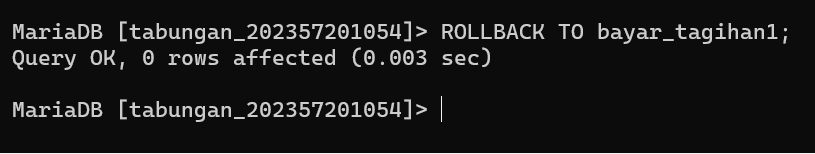
* 1. Jalankan perintah **transaksi** di bawah ini



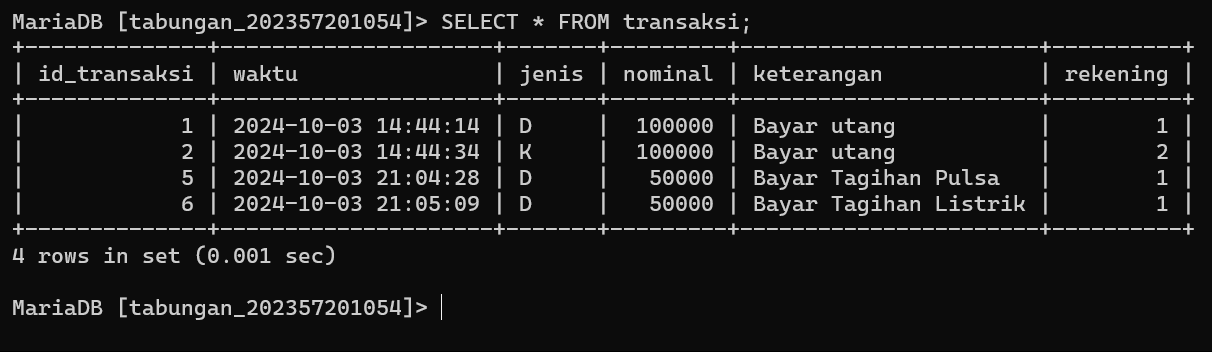
* 1. Cek isi tabel **transaksi**



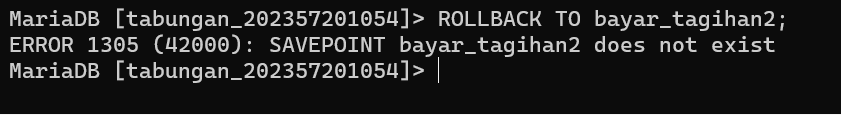
* 1. Rollback ke savepoint **bayar\_tagihan1**



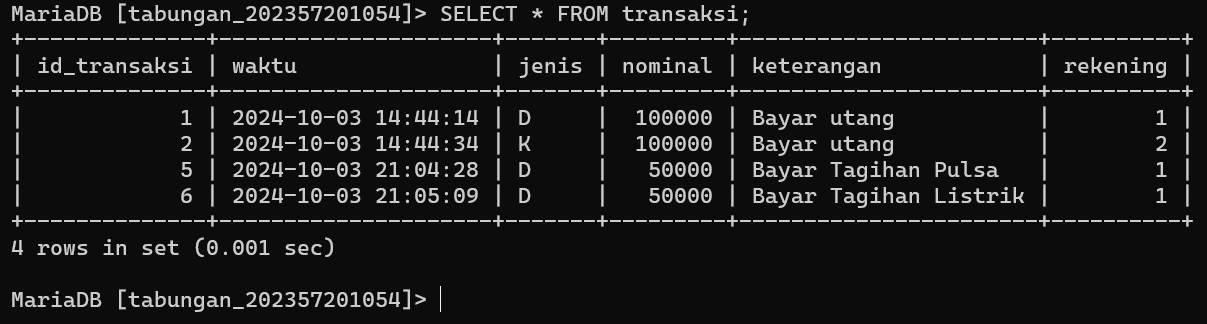
* 1. Cek isi tabel **transaksi**



* 1. Rollback ke savepoint **bayar\_tagihan2**



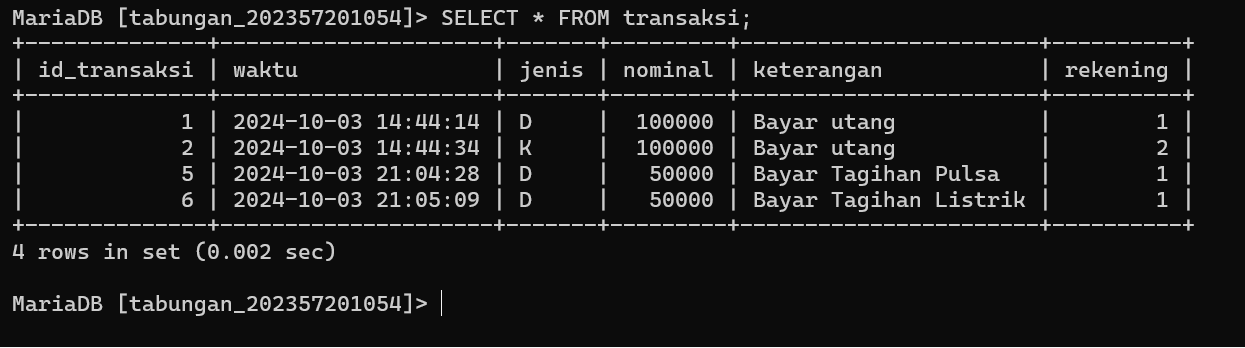
* 1. Cek isi tabel **transaksi**



* 1. Commit



* 1. Cek isi tabel **transaksi**



* 1. Apa yang bisa Anda simpulkan dari uji coba ini?

Dari uji coba yang dilakukan menggunakan transaksi dengan savepoint pada MariaDB, saya dapat menyimpulkan beberapa hal penting tentang pengelolaan transaksi dan penggunaan savepoint:

1. **Penggunaan Savepoint**

Dalam transaksi, savepoint digunakan untuk menandai titik-titik tertentu yang bisa digunakan untuk mengembalikan (rollback) sebagian dari transaksi tanpa membatalkan seluruh transaksi. Dalam uji coba ini, savepoint bayar\_tagihan1, bayar\_tagihan2 dan bayar\_tagihan3 ditetapkan setelah beberapa operasi INSERT.

1. **Fungsi Rollback ke Savepoint**

Rollback ke bayar\_tagihan1 berhasil dilakukan, yang mengembalikan transaksi ke keadaan setelah savepoint tersebut dibuat, menghilangkan perubahan yang dilakukan setelahnya. Hal ini mengurangi jumlah baris pada tabel dari semula sembilan baris menjadi empat baris, hanya mempertahankan transaksi yang terjadi sebelum dan sesaat setelah savepoint bayar\_tagihan1 dibuat.

1. **Kegagalan Rollback ke Savepoint yang Sudah Hilang**

Ketika mencoba melakukan rollback ke bayar\_tagihan2 setelah rollback ke bayar\_tagihan1, operasi tersebut gagal karena savepoint bayar\_tagihan2 tidak lagi eksis setelah rollback ke savepoint sebelumnya. Ini menunjukkan bahwa savepoint yang dipulihkan (rollback) ke savepoint sebelumnya akan menghilangkan savepoint yang dibuat setelahnya.

1. **Pengaruh Commit**

Akhirnya, COMMIT dijalankan untuk mempertahankan perubahan yang tersisa (empat baris data). Setelah commit, tidak ada lagi kemungkinan untuk rollback perubahan yang telah di-commit, memastikan bahwa data yang tersimpan adalah data yang terakhir disetujui setelah serangkaian operasi yang diinginkan.

**Kesimpulan Akhir:**

Uji coba ini menunjukkan bahwa savepoint sangat berguna dalam mengelola transaksi kompleks di mana beberapa sub-operasi bisa dibatalkan secara selektif tanpa mempengaruhi seluruh transaksi. Ini memperlihatkan fleksibilitas dalam mengelola kesalahan atau perubahan keputusan dalam batch operasi yang besar sebelum akhirnya memutuskan untuk commit atau rollback seluruh transaksi. Ini membuktikan pentingnya mekanisme savepoint dalam sistem manajemen basis data untuk memperbaiki atau membatalkan sebagian transaksi sesuai dengan kebutuhan operasional dan bisnis.