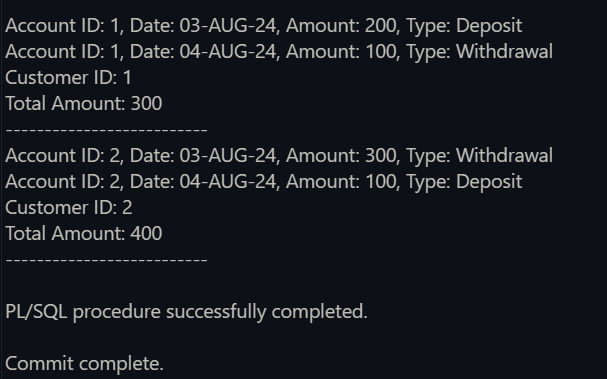
**Exercise 6: Cursors**

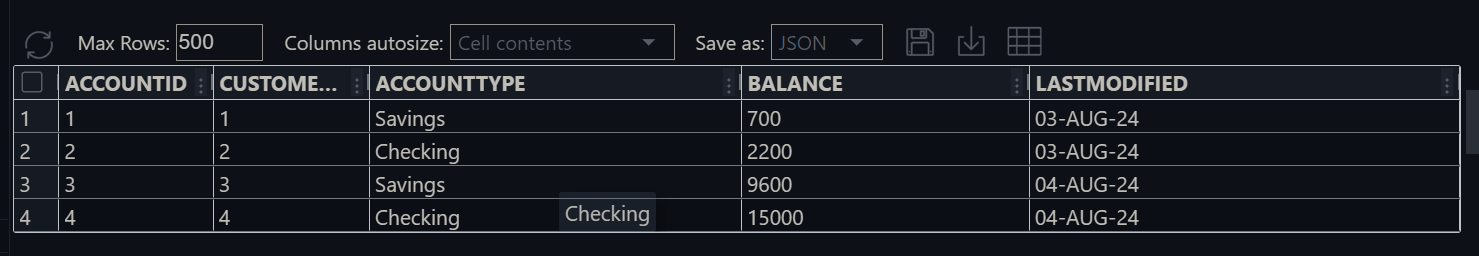
* **Scenario 1:** Generate monthly statements for all customers.
* **Question:** Write a PL/SQL block using an explicit cursor ***GenerateMonthlyStatements*** that retrieves all transactions for the current month and prints a statement for each customer.
* The code defines a cursor named `GenerateMonthlyStatements`, which is used to fetch transaction data for the current month. This data is joined with account information and ordered by `CUSTOMERID` and `TRANSACTIONDATE`, allowing for organized processing of transactions. The code declares variables to store the fetched data and control variables to track the current customer and calculate the total transaction amount for each customer. This setup ensures that transactions can be processed and summarized efficiently.
* Once the cursor is opened, the code iterates through each fetched transaction. It checks if the current transaction belongs to a new customer, resetting totals and printing the previous customer's statement if necessary. For each transaction, the code prints the details and accumulates the total transaction amount for the current customer. After processing all transactions, it prints the statement for the last customer and closes the cursor, ensuring that all resources are properly released and the output is complete.

**The OUTPUT :**



* **Scenario 2:** Apply annual fee to all accounts.
* **Question:** Write a PL/SQL block using an explicit cursor ***ApplyAnnualFee*** that deducts an annual maintenance fee from the balance of all accounts.
* The code begins by declaring a cursor named `ApplyAnnualFee`, which selects the `ACCOUNTID` and `BALANCE` from the `ACCOUNTS` table. This cursor is used to iterate over each account to update the balance. The code also initializes variables `v\_account\_id` and `v\_balance` to store the fetched `ACCOUNTID` and `BALANCE`, respectively. A constant `v\_annual\_fee` is set to 50, representing the annual fee that will be deducted from each account's balance.
* After opening the cursor `ApplyAnnualFee`, the code enters a loop to process each row. For each account, it fetches the data, deducts the annual fee from the balance, updates the balance in the `ACCOUNTS` table, and optionally logs the deduction for record-keeping. Once all rows have been processed, the cursor is closed to release resources, ensuring that the operation is completed efficiently and cleanly.

**The OUPUT :** Before using the cursor –

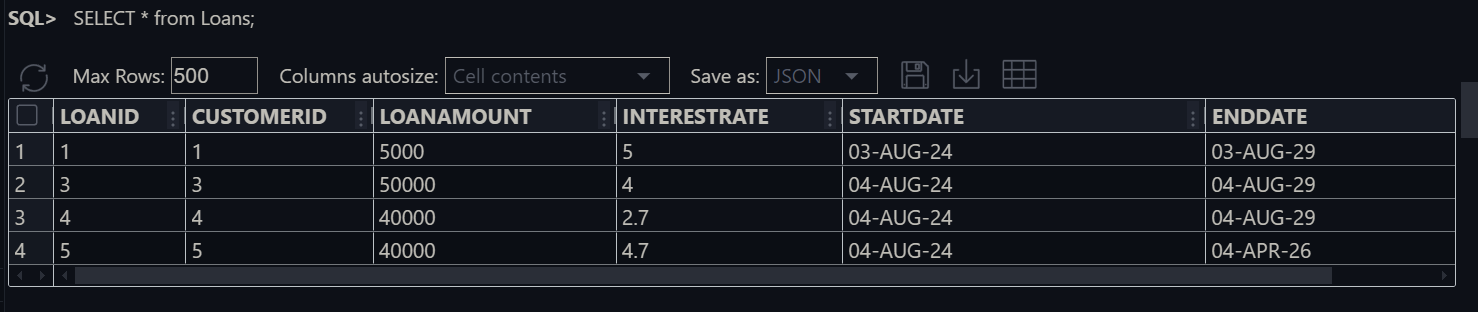
****

After using the cursor—

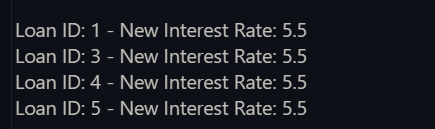


* **Scenario 3:** Update the interest rate for all loans based on a new policy.
* **Question:** Write a PL/SQL block using an explicit cursor ***UpdateLoanInterestRates*** that fetches all loans and updates their interest rates based on the new policy.
* The code defines a cursor named `UpdateLoanInterestRates`, which selects `LOANID` and `INTERESTRATE` from the `LOANS` table while locking the `INTERESTRATE` column for updates. It declares variables `v\_loan\_id` and `v\_interest\_rate` to store the cursor output. Additionally, a constant `v\_new\_interest\_rate` is set to 5.5, indicating the new interest rate to be applied to each loan.
* After opening the `UpdateLoanInterestRates` cursor, the code enters a loop to process each record. For each loan, it fetches the data into the variables, applies the new interest rate, updates the `LOANS` table, and optionally logs the changes for record-keeping. Once all records have been processed, the cursor is closed to ensure that all resources are properly released and the updates are complete.

**The OUTPUT :** Before applying the rates—

****

Apply the rates –

****

After updating the interest rate –

