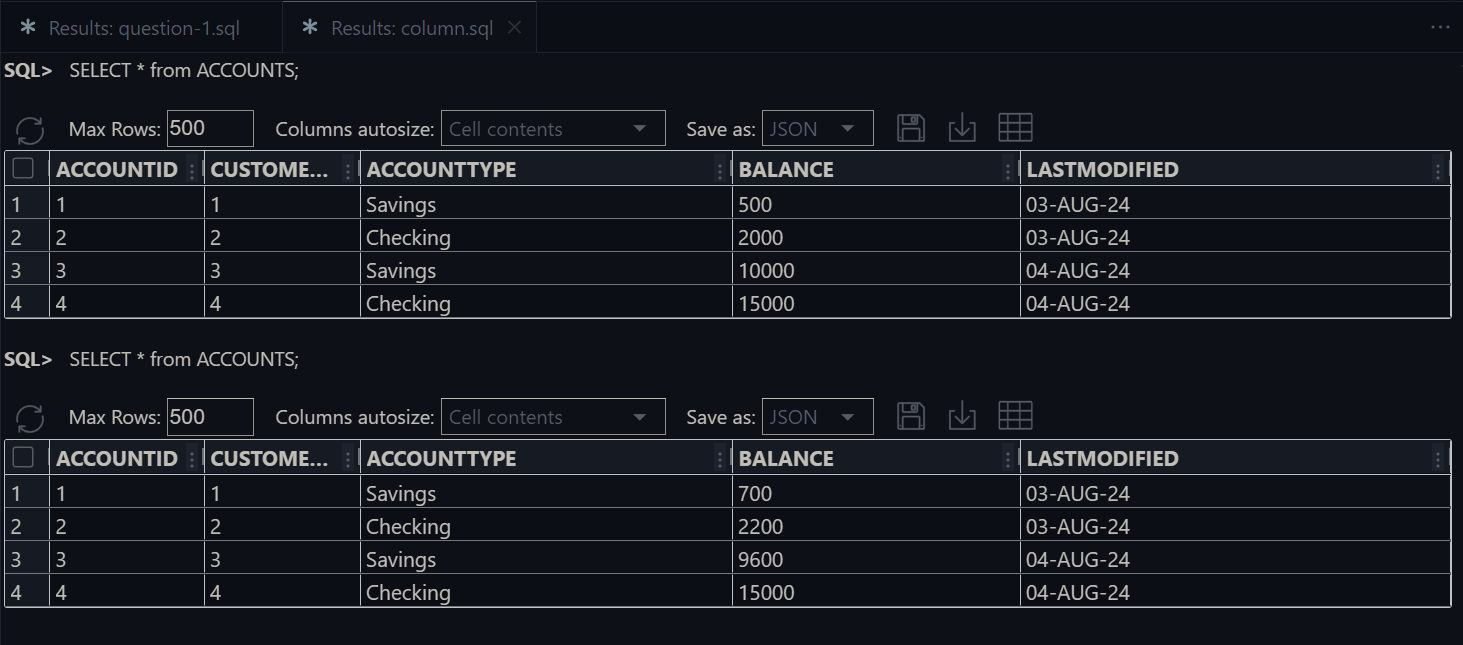
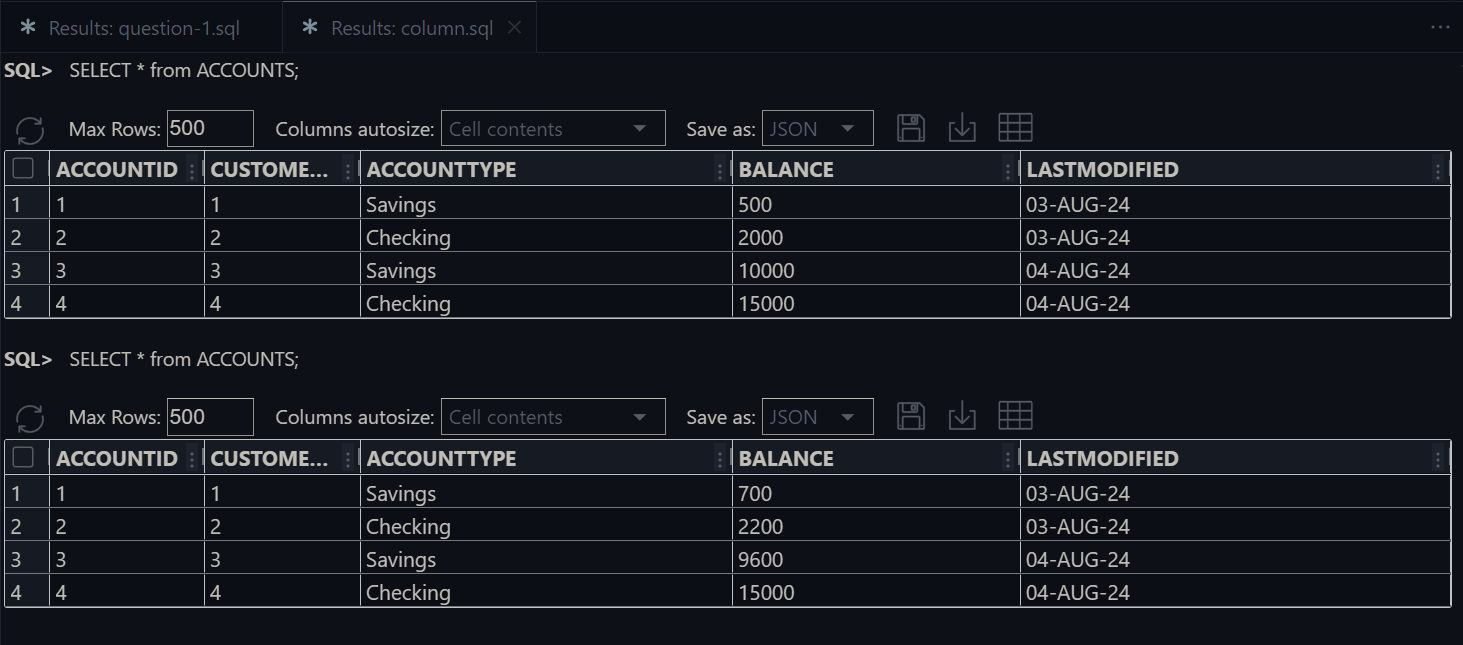
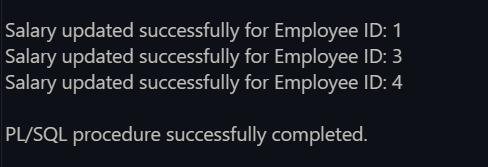
**Exercise 2: Error Handling**

* **Scenario 1:** Handle exceptions during fund transfers between accounts.
* **Question:** Write a stored procedure ***SafeTransferFunds*** that transfers funds between two accounts. Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.
* **Procedure Declaration:** Defines *SafeTransferFunds* with parameters for source account ID, target account ID, and transfer amount.
* **Balance Check:** Retrieves and locks the source account's balance; raises *insufficient*\_*funds* exception if the balance is inadequate.
* **Update Statements:** Deducts the amount from the source account and adds it to the target account.
* **Transaction Control:** Commits the transaction if updates succeed.
* **Exception Handling:** Rolls back the transaction and logs appropriate error messages for exceptions like *insufficient\_funds* and *invalid\_account*.
* **Sample Calls:** Tests the procedure with valid and invalid scenarios to verify successful and failed transfers.
* **The output of the code –**

Before transaction

After transaction:

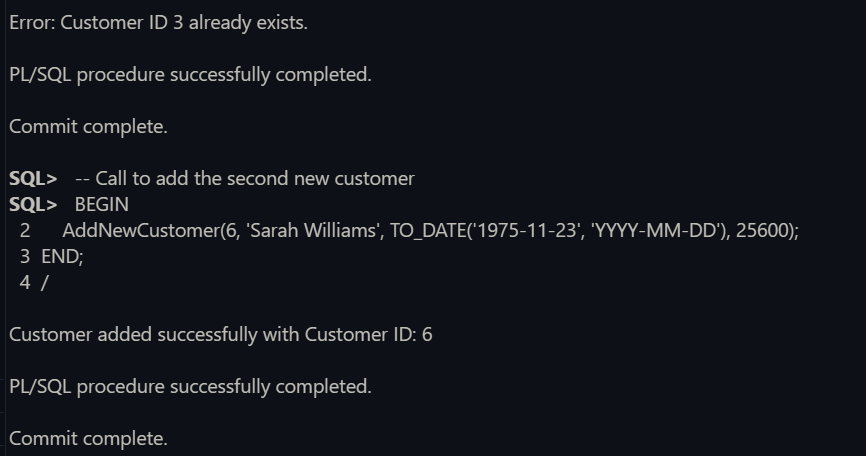
* **Scenario 2:** Manage errors when updating employee salaries.
* **Question:** Write a stored procedure **UpdateSalary** that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.
* **Fetch Current Salary**: The procedure attempts to retrieve the current salary of the employee using the provided employee ID.
* **Update Salary**: If the employee exists, the procedure updates the salary by adding the specified percentage increase.
* **Commit and Handle Exceptions**: It commits the transaction if successful and handles exceptions by logging appropriate error messages and rolling back the transaction if the employee ID does not exist or other errors occur.
* **Output Messages**: The procedure outputs messages indicating whether the salary was successfully updated or if an error occurred.

**The output of the code:**



Before Salary update

After Salary update

* **Scenario 3:** Ensure data integrity when adding a new customer.
* **Question:** Write a stored procedure **AddNewCustomer** that inserts a new customer into the Customers table. If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.
* **Objective**: Insert a new customer into the *Customers* table.
* **Integrity Check**: Ensure no customer with the same ID already exists.
* **Exception Handling**: Handle duplicate ID errors by logging an appropriate message.
* **Transaction Control**: Prevent insertion if a duplicate ID is found, ensuring data integrity.

**Output of the code:**

Here add a Customer in ID 3 and ID 6. The ID 3 already have a customer detail so it does not change the ID 3 and add new Details in ID 6

