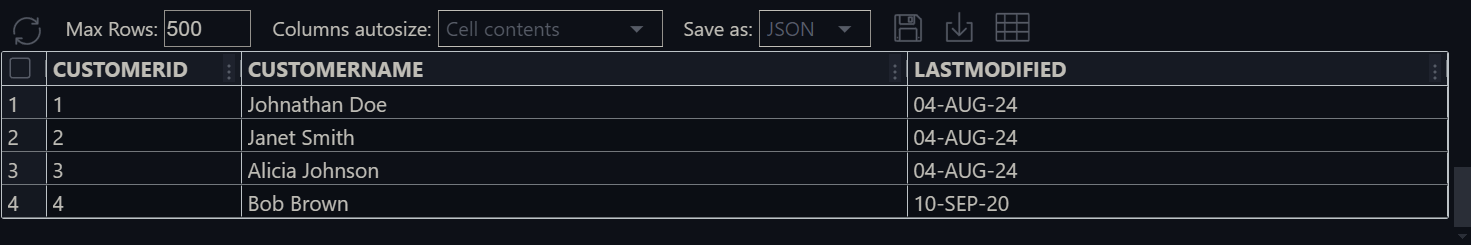
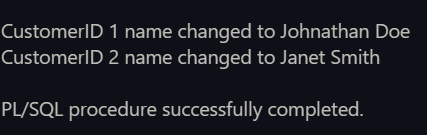
**Exercise 5: Triggers**

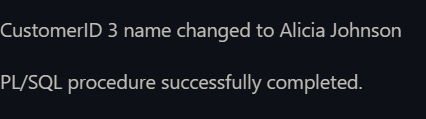
* **Scenario 1:** Automatically update the last modified date when a customer's record is updated.
* **Question:** Write a trigger ***UpdateCustomerLastModified*** that updates the *LastModified* column of the Customers table to the current date whenever a customer's record is updated.
* To implement the `UpdateCustomerLastModified` trigger, begin by connecting to an Oracle database as `SYSDBA` to grant necessary permissions and configure the schema. Specifically, grant `EXECUTE` permission on `DBMS\_LOCK` to a non-SYS user and set unlimited quota on the user’s schema. Then, create the `Customer` table with columns `CustomerID`, `CustomerName`, and `LastModified`, ensuring it doesn't already exist. This setup prepares the database for automatically updating records.
* Next, define the trigger `UpdateCustomerLastModified` to automatically update the `LastModified` column with the current date whenever a record in the `Customer` table is updated. After establishing the trigger, insert several records into the `Customer` table and update some of these records to test the functionality. Verify the correct update of the `LastModified` column by executing SQL statements and using `DBMS\_OUTPUT.PUT\_LINE` for output messages. Use `DBMS\_LOCK.SLEEP` to observe timestamp differences, ensuring the trigger functions as intended.

**The OUTPUT :**

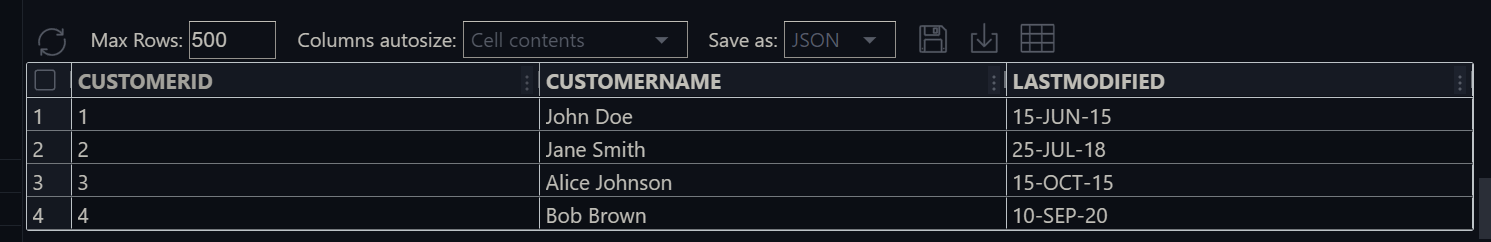


* Modify the *CustomerID* 1, 2, 3



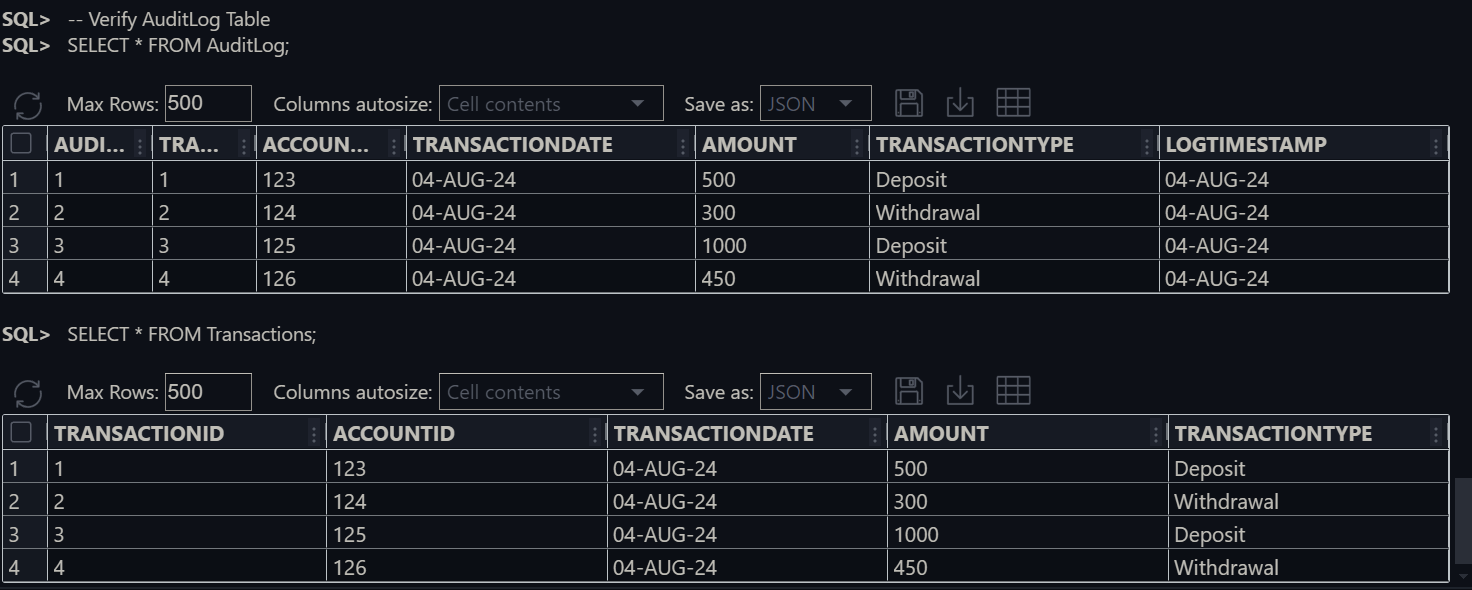


After Modification –

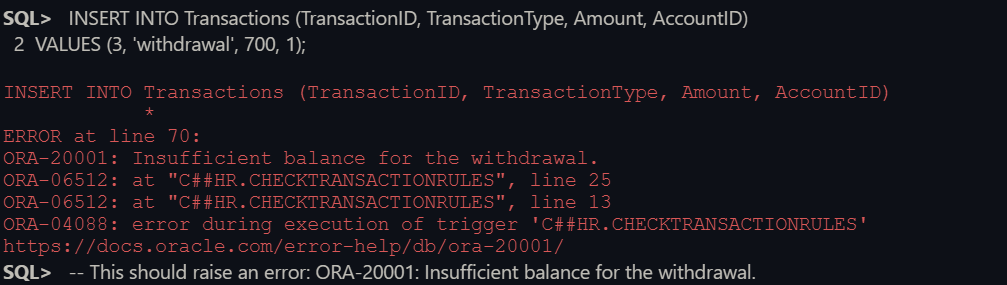
****

* **Scenario 2:** Maintain an audit log for all transactions.
* **Question:** Write a trigger ***LogTransaction*** that inserts a record into an *AuditLog* table whenever a transaction is inserted into the Transactions table.
* To set up a system for logging transactions, start by connecting to the Oracle database as `SYSDBA`. Grant unlimited quota on the `USERS` tablespace to the user specified by `&&DB\_USER` and then reconnect as `&&DB\_USER`. This allows the user to manage their schema effectively. Next, create two tables: `AuditLog` and `Transactions`. The `AuditLog` table will store audit logs, while the `Transactions` table will hold transaction records.
* After setting up the tables, define a sequence named `AuditLog\_seq` to generate unique `AuditID` values for the `AuditLog` table. Then, create a trigger called `LogTransaction`, which will automatically insert a record into `AuditLog` whenever a new row is added to the `Transactions` table. Finally, insert sample transaction data into the `Transactions` table and select all records from both `AuditLog` and `Transactions` to verify that the trigger is functioning correctly, ensuring that each transaction is appropriately logged in the `AuditLog`.

**The OUTPUT :** Transaction and AuditLog table respectively—



* **Scenario 3:** Enforce business rules on deposits and withdrawals.
* **Question:** Write a trigger ***CheckTransactionRules*** that ensures withdrawals do not exceed the balance and deposits are positive before inserting a record into the Transactions table.
* To begin implementing transaction management, connect to the Oracle database as `SYSDBA`. Grant `EXECUTE` permissions on `DBMS\_LOCK` to `DB\_USER` and provide unlimited quota on the `USERS` tablespace for `DB\_USER`. This setup ensures the user has the necessary permissions and resources. Next, create two tables: `Accounts`, which stores account information, and `Transactions`, which records transactions. Implement foreign key constraints to link the `Transactions` table to the `Accounts` table, establishing a relational structure between the two.
* After creating the tables, insert sample records into the `Accounts` table and display its contents for verification. Define a trigger named `CheckTransactionRules` to enforce rules on transactions before insertion. This trigger checks for sufficient balances for withdrawals and ensures deposit amounts are positive, raising errors if any rules are violated. Attempt to insert transactions into the `Transactions` table to validate the trigger’s functionality. The trigger should generate error messages for invalid operations, such as insufficient balance and non-positive deposit amounts. Finally, display the contents of both tables to verify that only valid transactions were inserted.

**The OUPUT :**this error is shown because of insufficient balance for withdrawl-

The account and transaction table respectively –

