Assignment 01 From CSD3848(Αντώνης Συκουτρής)

A)

I correctly installed the Oracle Database Express Edition and also the Oracle Developer SQL which visualizes the tables and it makes the homework much more enjoyable. In the first question we are asked to create a table with certain columns and a constraint. Below is the proof that I made such one(and I added 4 dummy account to be able to do transactions).

	ACCID	BALANCE
1	1	950
2	2	3000.55
3	3	1743.27
4	4	256.47

B)

In the second question we were asked to create a transaction between 2 accounts(without the try catch).Below is the picture of the SQL quire and the updated table.

Worksheet | Query Builder

```sql
--Create the table
CREATE TABLE accounts (
accId INTEGER NOT NULL PRIMARY KEY,
balance DECIMAL(11,2) NOT NULL,
CONSTRAINT empty_account CHECK (balance >= 0.00)
);

--Add a few dummy accounts
INSERT INTO SYS.accounts (accId, balance) VALUES (1, 950.00);
INSERT INTO SYS.accounts (accId, balance) VALUES (2, 3000.55);
INSERT INTO SYS.accounts (accId, balance) VALUES (3, 1743.27);
INSERT INTO SYS.accounts (accId, balance) VALUES (4, 256.47);

--We are Moving to Question B
UPDATE SYS.accounts SET balance = balance - 300 WHERE accid = 2;
UPDATE SYS.accounts SET balance = balance + 300 WHERE accid = 1;
```

Script Output × | Script Output 1 ×

Task completed in 0.123 seconds

```
INSERT INTO SYS.accounts (accId, balance) VALUES (4, 256.47)
Error report -
ORA-00001: unique constraint (SYS.SYS_C007358) violated


1 row updated.


1 row updated.
```

	ACCID	BALANCE
1	1	1250
2	2	2700.55
3	3	1743.27
4	4	256.47

Afterwards we were asked to examine the different situations. For example if an error triggers an automatic

ROLLBACK or the transaction continues even after an error.

In all of the cases I will show that the transaction was successful and didn’t stop.

<u>Add money to an account that doesn’t exist.</u>

Worksheet | Query Builder

```
--Create the table
CREATE TABLE accounts (
accId INTEGER NOT NULL PRIMARY KEY,
balance DECIMAL(11,2) NOT NULL,
CONSTRAINT empty_account CHECK (balance >= 0.00)
);

--Add a few dummy accounts
INSERT INTO SYS.accounts (accId, balance) VALUES (1, 950.00);
INSERT INTO SYS.accounts (accId, balance) VALUES (2, 3000.55);
INSERT INTO SYS.accounts (accId, balance) VALUES (3, 1743.27);
INSERT INTO SYS.accounts (accId, balance) VALUES (4, 256.47);

--We are Moving to Question B
UPDATE SYS.accounts SET balance = balance - 300 WHERE accid = 2;
UPDATE SYS.accounts SET balance = balance + 300 WHERE accid = 7;
```

Script Output × | Script Output 1 ×

Task completed in 0.084 seconds

```
INSERT INTO SYS.accounts (accId, balance) VALUES (4, 256.47)
Error report -
ORA-00001: unique constraint (SYS.SYS_C007358) violated


1 row updated.


0 rows updated.
```

Sort.. | F

	ACCID	BALANCE
1	1	950
2	2	2700.55
3	3	1743.27
4	4	256.47

Take Money from an account that doesn't have 300.

Worksheet | Query Builder

```
--Create the table
CREATE TABLE accounts (
accId INTEGER NOT NULL PRIMARY KEY,
balance DECIMAL(11,2) NOT NULL,
CONSTRAINT empty_account CHECK (balance >= 0.00)
);

--Add a few dummy accounts
INSERT INTO SYS.accounts (accId, balance) VALUES (1, 950.00);
INSERT INTO SYS.accounts (accId, balance) VALUES (2, 3000.55);
INSERT INTO SYS.accounts (accId, balance) VALUES (3, 1743.27);
INSERT INTO SYS.accounts (accId, balance) VALUES (4, 256.47);

--We are Moving to Question B
UPDATE SYS.accounts SET balance = balance - 300 WHERE accid = 4;
UPDATE SYS.accounts SET balance = balance + 300 WHERE accid = 1;
```

Script Output | Script Output 1

Task completed in 0.069 seconds

```
Error starting at line : 15 in command -
UPDATE SYS.accounts SET balance = balance - 300 WHERE accid = 4
Error report -
ORA-02290: check constraint (SYS.EMPTY_ACCOUNT) violated


1 row updated.
```

	ACCID	BALANCE
1	1	1250
2	2	3000.55
3	3	1743.27
4	4	256.47

The changes were permanent because the transaction was finished with a commit(F11 commits) so the changes you see are in the session of Oracle Developer and in SQL Plus. I showed you screenshots as well from SQL Plus to prove that.

C)

Now we are introduced to error handlers and that Is going to be a long list. First I will show you the **<u>PROCEDURE</u>** I made to be able to do what was asked.

Worksheet | Query Builder

```oracle_procedure
create or replace NONEDITIONABLE PROCEDURE ACC_TR
(
  P_ACID1 IN NUMBER
, P_ACID2 IN NUMBER
) AS
        acc_check INTEGER;
        bal_check INTEGER;
        e_invalid_bal EXCEPTION;
        e_acc_same    EXCEPTION;
BEGIN
--CHECKING FOR POTENSIAL WRONG ACCOUNTS
    SELECT accId INTO acc_check FROM SYS.accounts WHERE accid = p_acid1;
    SELECT accId INTO acc_check FROM SYS.accounts WHERE accid = p_acid2;
--CHECKING FOR THE SAME ACCOUNTS
    IF p_acid1 = p_acid2 THEN
        RAISE e_acc_same;
    END IF;

--CHECK IF THE BALANCE OF THE GIVER IS APPROPRIATE
    SELECT balance INTO bal_check FROM SYS.accounts WHERE accId = p_acid1;

--IF IT IS NOT RAISE THE EXCEPTION FLAGS
    IF bal_check < 300 THEN
        RAISE e_invalid_bal;
    END IF;

--IF EVERYTHING IS OKEY THEN PROCCED WITH THE TRANSACTION
    UPDATE SYS.ACCOUNTS SET balance = balance - 300 WHERE accid = P_ACID1;
    UPDATE SYS.ACCOUNTS SET balance = balance + 300 WHERE accid = P_ACID2;
    COMMIT;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('ERROR: NO RECORDS');
        ROLLBACK;
    WHEN TOO_MANY_ROWS THEN --THIS IS NOT NESSESARY BUT I DO IT FOR CHECKING PERPUSES OF THE DATABASE
        DBMS_OUTPUT.PUT_LINE('ERROR: More than 1 Records found');
        ROLLBACK;
    WHEN INVALID_NUMBER THEN
        DBMS_OUTPUT.PUT_LINE('ERROR: Invalid Number.YOU HAD ONE JOB');
        ROLLBACK;
    WHEN e_invalid_bal THEN
        DBMS_OUTPUT.PUT_LINE('ERROR: Balance is less than 300');
        ROLLBACK;
    WHEN e_acc_same THEN
```

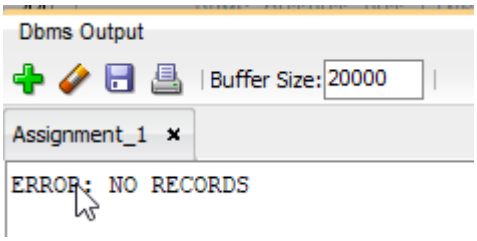
```
    WHEN e_acc_same THEN
        DBMS_OUTPUT.PUT_LINE('ERROR: Try a transaction with a different account');
        ROLLBACK;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE ('ERROR: Unexpected error');
        RAISE;
        ROLLBACK;

END ACC_TR;
/

COMMIT;
```

Now I will explain how I handled the different exceptions. I will show screenshots that prove that the different cases are being handled correctly.

<u>FIRST CASE</u>:

If the user gives wrong accounts(accounts that do not have a valid accId, that do not exist in the table) using the <u>NO_DATA_FOUND</u> exception I handle it.

```
EXEC    ACC_TR(0, 3);
```



	ACCID	BALANCE
1	1	950
2	2	3000.55
3	3	1743.27
4	4	256.47

SECOND CASE:

If the accounts are the same the transaction shouldn't be made possible because it occupies time of the database to update the values.

```
EXEC    ACC_TR(1, 1);
```

Dbms Output

Buffer Size: 20000

Assignment_1

```
ERROR: Try a transaction with a different account
```

	ACCID	BALANCE
1	1	950
2	2	3000.55
3	3	1743.27
4	4	256.47

THIRD CASE:

If the user that gives 300 doesn't have that much(I purposely add one account to have 256.47) the transaction should stop.

```
EXEC    ACC_TR(4, 2);
```

Dbms Output

Buffer Size: 20000

Assignment_1

```
ERROR: Balance is less than 300
```

Columns | Data | Model | Constraints | Gra

Sort..

	ACCID	BALANCE
1	1	950
2	2	3000.55
3	3	1743.27
4	4	256.47

Those were the main exceptions. If there is another exception then This will handle it:

```
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE ('ERROR: Unexpected error');
```

Else the transaction will proceed normally and I will post screenshots of such a transaction.

BEFORE

	ACCID	BALANCE
1	1	950
2	2	3000.55
3	3	1743.27
4	4	256.47

```
EXEC    ACC_TR(2, 3);
```

D)

AFTER

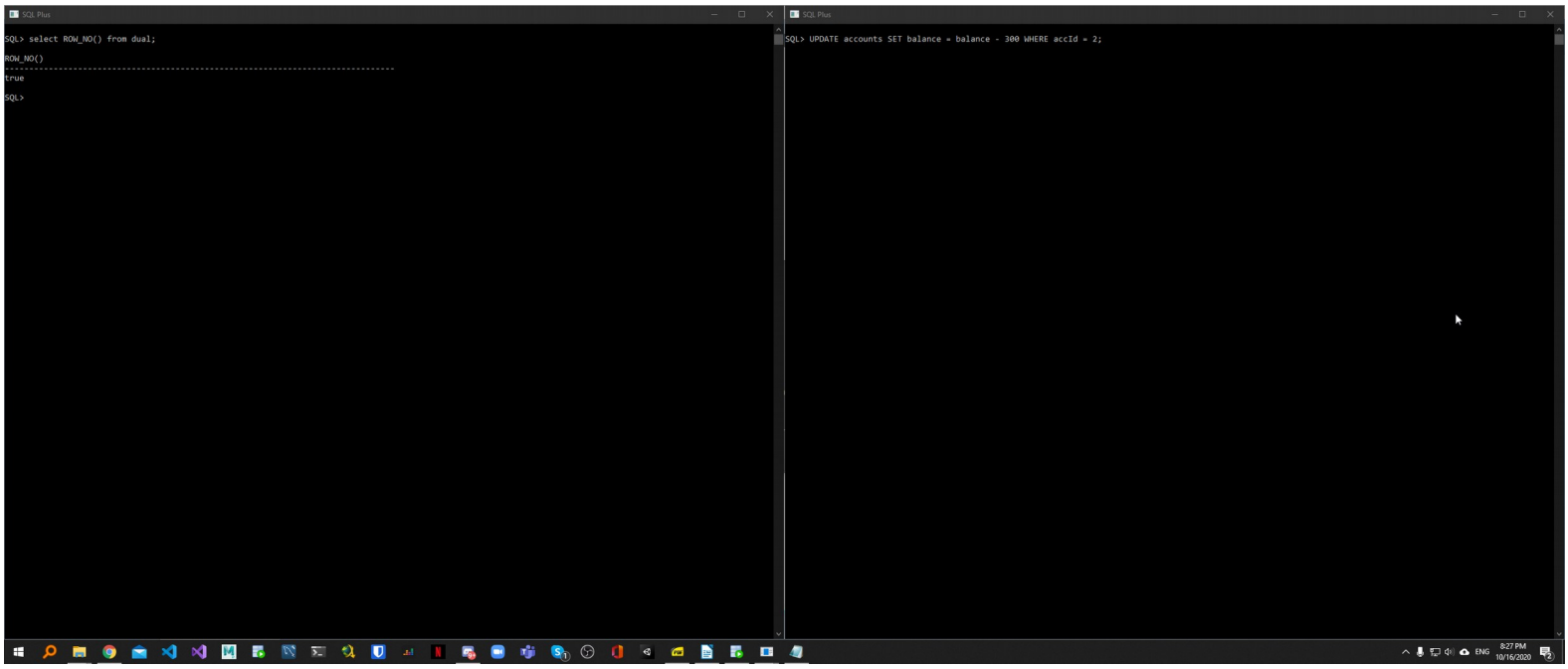
	ACCID	BALANCE
1	1	950
2	2	2700.55
3	3	2043.27
4	4	256.47

```
CREATE OR REPLACE FUNCTION ROW_NO
RETURN BOOLEAN AS

e_row_lock EXCEPTION;
BEGIN

LOCK TABLE accounts IN SHARE ROW EXCLUSIVE MODE NOWAIT;
RETURN TRUE;


EXCEPTION
WHEN e_row_lock THEN
    DBMS_OUTPUT.PUT_LINE('ERROR: It is locked');
RETURN FALSE;

END ROW_NO;

COMMIT;
```

D)

I created a function that only gives a lock to a session and if it cant it throws an exception.

To demonstrate it I will make 2 sessions using SQL plus and I will give a lock to one and you will see that the other is waiting until a commit or a rollback.

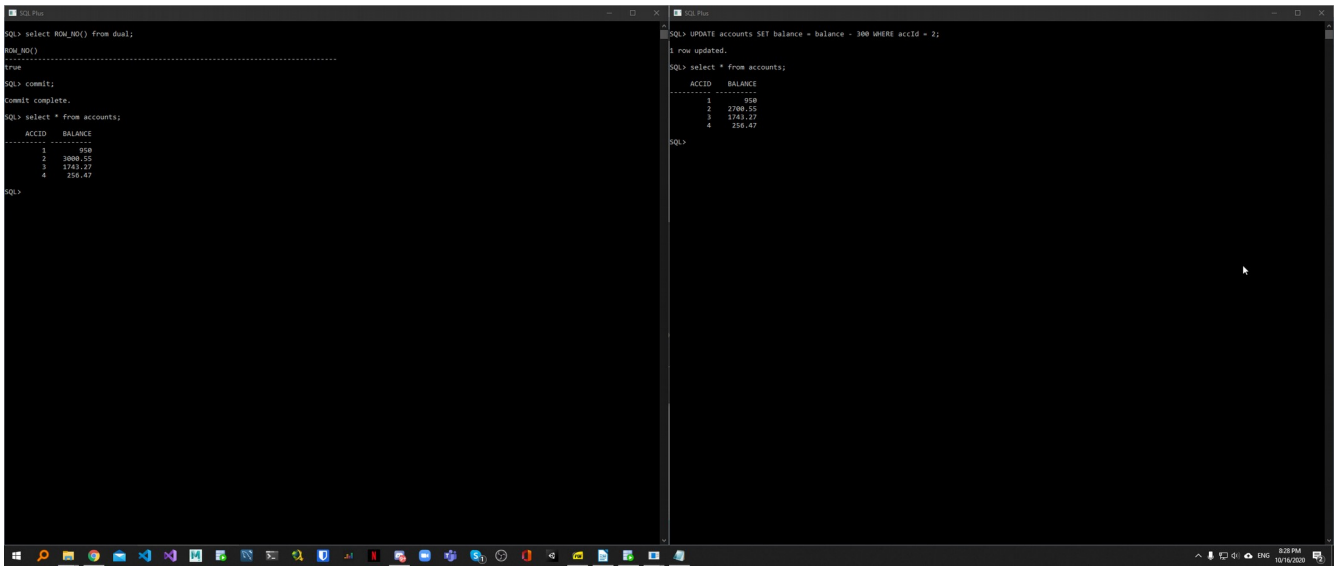


SQL Plus

```
SQL> select ROW_NO() from dual;

ROW_NO()
--------------------------------------------------------------------------------
true

SQL>
```

SQL Plus

```
SQL> UPDATE accounts SET balance = balance - 300 WHERE accId = 2;
```



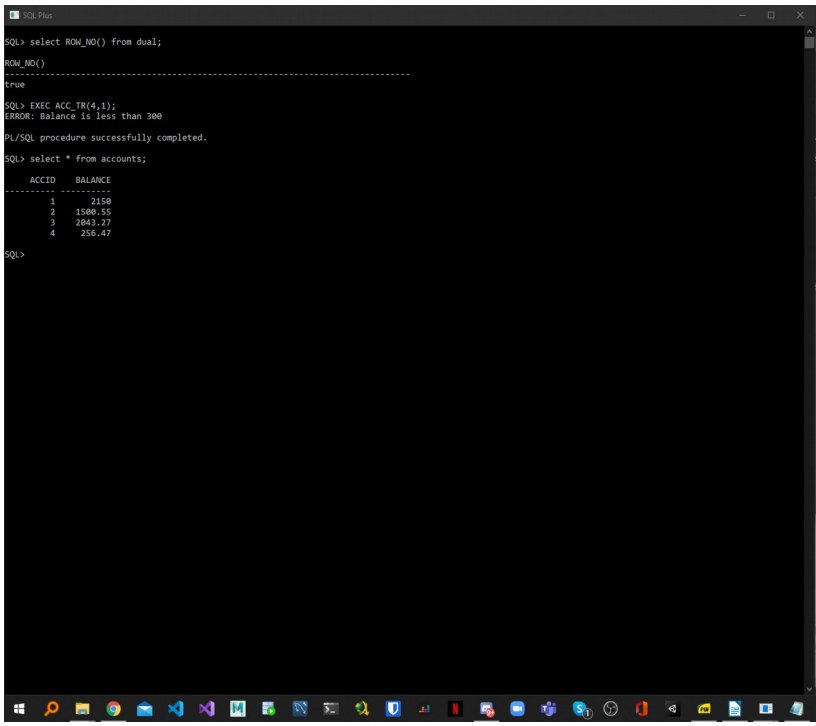
```
SQL> select ROW_NO() from dual;

ROW_NO()
--------------------------------------------------------------------------------
true

SQL> EXEC ACC_TR(3,2);

PL/SQL procedure successfully completed.

SQL>
```

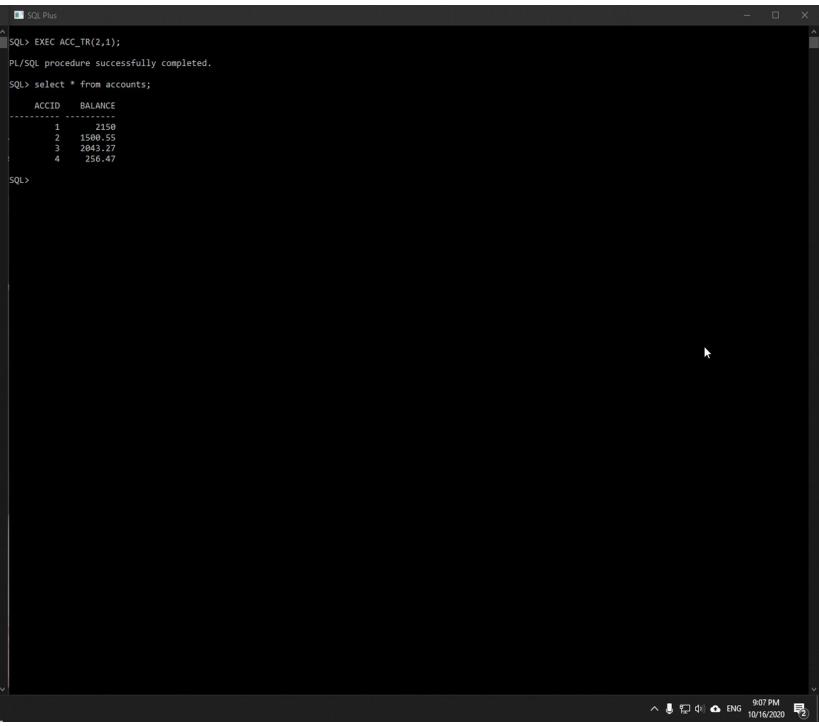
```
SQL> EXEC ACC_TR(2,3);

PL/SQL procedure successfully completed.

SQL>
```

```
SQL> select ROW_NO() from dual;

ROW_NO()
--------------------------------------------------------------------------------
true

SQL> EXEC ACC_TR(3,2);

PL/SQL procedure successfully completed.

SQL> select * from accounts;

     ACCID    BALANCE
---------- ----------
         1       1250
         2    2400.55
         3    2043.27
         4     256.47

SQL>
```

```
SQL> EXEC ACC_TR(2,3);

PL/SQL procedure successfully completed.

SQL> select * from accounts;

     ACCID    BALANCE
---------- ----------
         1       1250
         2    2400.55
         3    2043.27
         4     256.47

SQL>
```

Here I demonstrate that if you do a transaction (try to take money from an account < 300) it will throw an exception and it will rollback.





BONUS

A,B)

	accId	balance
▶	1	950.00
	2	3000.55
	3	1743.27
	4	256.47
✱	NULL	NULL

```
CREATE TABLE accounts (
accId INTEGER NOT NULL PRIMARY KEY,
balance DECIMAL(11,2) NOT NULL,
CONSTRAINT empty_account CHECK (balance >= 0.00)
);

insert into accounts (accId, balance) values (1, 950.00);
insert into accounts (accId, balance) values (2, 3000.55);
insert into accounts (accId, balance) values (3, 1743.27);
insert into accounts (accId, balance) values (4, 256.47);

select * from accounts;
```

I created the table that was given in MySQL WORKBENCH and I inserted 4 accounts with the values shown in the picture above.

The problem with MySQL is that the constraints are not even checked so I can have a negative balance...Of course I could add an if else but the point is to see what the default MySQL do.

accId	balance
1	950.00
2	3000.55
3	1743.27
4	-43.53
NULL	NULL

```
UPDATE accounts SET balance = balance - 300 WHERE accId = 4;

select * from accounts;
```

C)

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `ACC_TR`(IN P_ACID1 int, IN P_ACID2 int)
BEGIN

start transaction;

IF (SELECT accId FROM ACCOUNTS WHERE accId = P_ACID1) IS NULL THEN
SELECT 'First Parameter Does not Exist in Table Exception OCCURED';
rollback;

ELSEIF (SELECT accId FROM ACCOUNTS WHERE accId = P_ACID2) IS NULL THEN
SELECT 'Second Parameter Does not Exist in Table Exception OCCURED';
rollback;


ELSEIF P_ACID1 = P_ACID2 THEN
SELECT 'Same Parameter Exception OCCURED';
rollback;


ELSEIF (SELECT balance FROM ACCOUNTS WHERE accId = P_ACID1) < 300 THEN
SELECT 'First Parameter Does not Have Enough Balance';
rollback;

ELSE
UPDATE ACCOUNTS SET balance = balance - 300 WHERE accId = P_ACID1;
UPDATE ACCOUNTS SET balance = balance + 300 WHERE accId = P_ACID2;
commit;
END IF;
END
```

I named the MySQL procedure(above picture) as I named the oracle one (also the parameters) in order to have easier time with all those variables. Below I will show examples of using it with the (call).

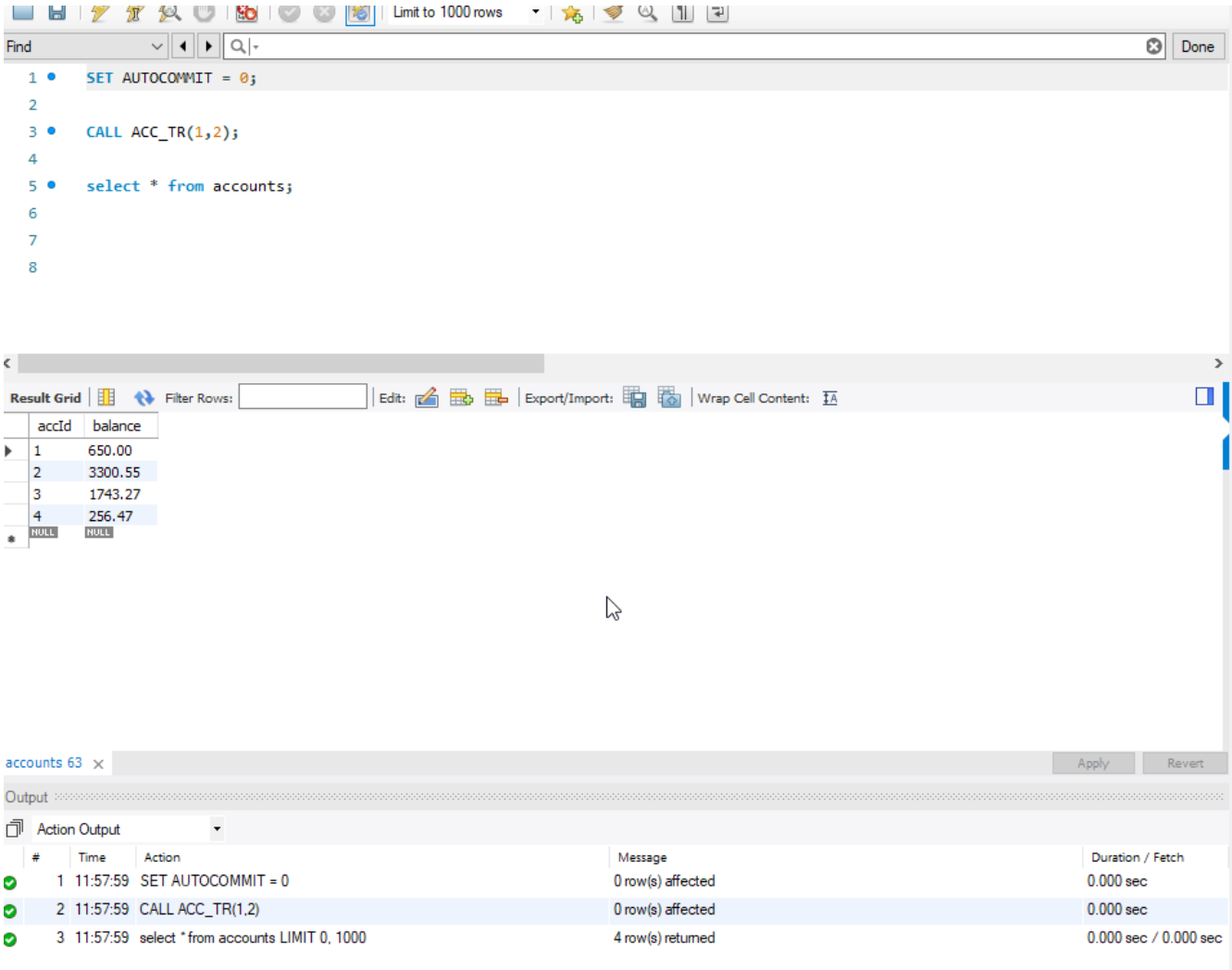
```
SET AUTOCOMMIT = 0;

CALL ACC_TR(4,3);

select * from accounts;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

accId	balance
1	950.00
2	3000.55
3	1743.27
4	256.47
NULL	NULL

First Parameter Does not Have Enough Balance
First Parameter Does not Have Enough Balance

If I try to do a valid transaction then



D)