Transactions In SQL
Steps Of Transaction Process

1. Begin Transactions
2. Process the commands
3. Check for errors
   1. If error occur rollback the transaction
4. Commit the transaction
Example

- Step 1: Update the product 1 qty to new value

  ```
  Begin Transaction
  Update Tr_Prodcuts set QtyAva=300 where ProductID =1;
  ```

- Step 2: The value of the product a will change
Example

Step 3: Open a new connection and try to retrieve data:

```sql
select * from Tr_Products
```

Step 2: The result is in the processing:

Executing query...
What is the reason?

The reason here is although we (user 1) have initiated the transaction it is not rollback or committed. In SQL server environment by default transaction isolation level is set to read committed data. Therefore the second user will not be able to read the data process by the user 1

As a solution we can put the state to read uncommitted data

```sql
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
```
ROLLBACK TRANSACTION

<table>
<thead>
<tr>
<th>ProductID</th>
<th>Name</th>
<th>UnitPrice</th>
<th>QtyAv</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Samsung Mobiles</td>
<td>2300</td>
<td>300</td>
</tr>
<tr>
<td>2</td>
<td>Huawei Mobiles</td>
<td>3500</td>
<td>50</td>
</tr>
</tbody>
</table>

Begin Transaction

Update Tr_Products set QtyAv = 300 where ProductID = 1;

ROLLBACK TRANSACTION
Commit Transaction

-Example 2

1. Now Commit the transaction

   ```
   Begin Transaction
   Update Tr_Products set QtyAva=400 where ProductID =1;
   COMMIT TRANSACTION
   ```

2. Retrieve the information

3. The result Will be 400 as the new data is committed
Commit Transaction

-Example 2

1. Try to rollback the transaction

```
Begin Transaction
Update Tr_Prodcuts set QtyAva=400 where ProductID =1;
--COMMIT TRANSACTION
ROLLBACK TRANSACTION
```

1. retrieve the information

2. The result is now not initial 100, it is committed 400 (as the new value is committed)

<table>
<thead>
<tr>
<th>ProductID</th>
<th>Name</th>
<th>UnitPrice</th>
<th>QtyAva</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Samsung Mobiles</td>
<td>2300</td>
<td>400</td>
</tr>
<tr>
<td>2</td>
<td>Huawei Mobiles</td>
<td>3500</td>
<td>50</td>
</tr>
</tbody>
</table>
Example

- Create Two Tables as Phy_Add and Mail_Add

  | AddressID (nchar(10), null) |
  | EmpNum (nchar(10), null)   |
  | City (nchar(10), null)     |
```sql
select * from Mail_Add;
select * from Phy_Add;
```
Create Stored Procedure to Execute transaction

Create Procedure Trasnaction_Error_Check
AS
BEGIN
BEGIN TRY
BEGIN TRANSACTION
Update Phy_Add set City='Kandy'
where AddressID='2'And EmpNum ='300'

Update Mail_Add set City='Kandy'
where AddressID='2'And EmpNum = '300'

Commit Transaction
  Print 'Transaction Committed'
End Try

Begin Catch
  Rollback Transaction
  Print 'Transaction Rollback'
End Catch
End

Without Error
(1 row(s) affected)

(1 row(s) affected)

Transaction Committed

<table>
<thead>
<tr>
<th>AddressID</th>
<th>EmpNum</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Kandy</td>
</tr>
</tbody>
</table>
Create Stored Procedure to Execute transaction

```
Alter Procedure Transaction_Error_Check
AS
BEGIN
BEGIN TRY
    BEGIN TRANSACTION
    Update Phy_Add set City='Matale'
    where AddressID='2' And EmpNum = '300'
    Update Mail_Add set City='Ratmalana MountLav'
    where AddressID='2' And EmpNum = '300'
    Commit Transaction
    Print 'Transaction Committed'
    End Try
    Begin Catch
    Rollback Transaction
    Print 'Transaction Rollback'
    End Catch
END
```
(1 row(s) affected)

(0 row(s) affected)

Transaction Rollback

<table>
<thead>
<tr>
<th>AddressID</th>
<th>EmpNum</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>300</td>
</tr>
</tbody>
</table>