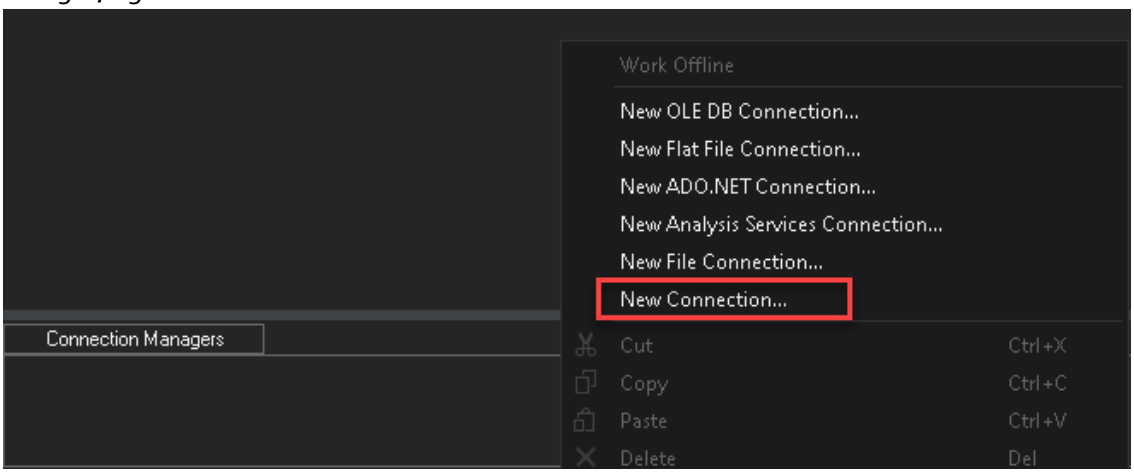
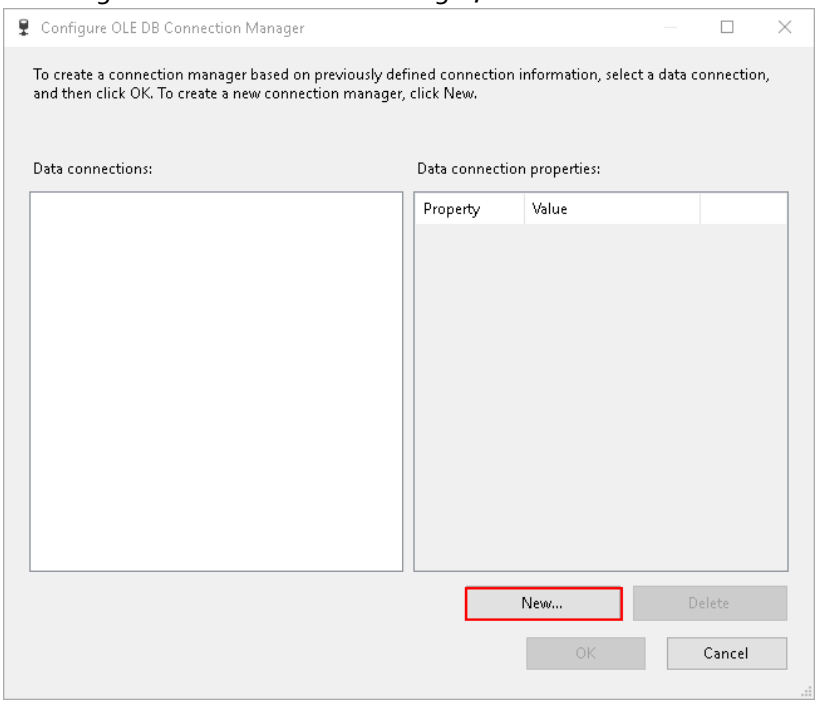
SQL SERVER INTEGRATION SERVICES

MODULE 04 – LAB 03 – EXERCISE 01: CHECKPOINTS

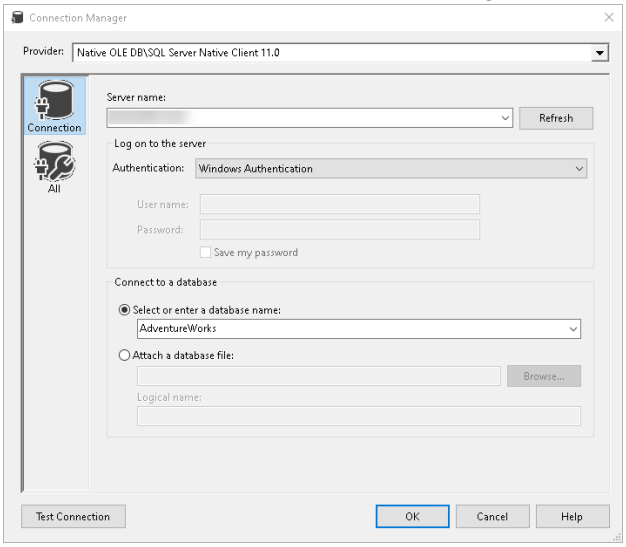
1. Create new integration services project.
2. Set up a connection manager for the database. In the bottom center pane under Connection Manger, right-click select New Ole-DB Connection.



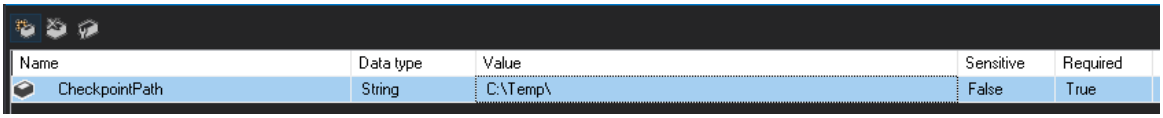
3. In Configure OLE DB Connect Manager, click New.



4. Setup connection to SQL Server pointing to the AdventureWorks database.

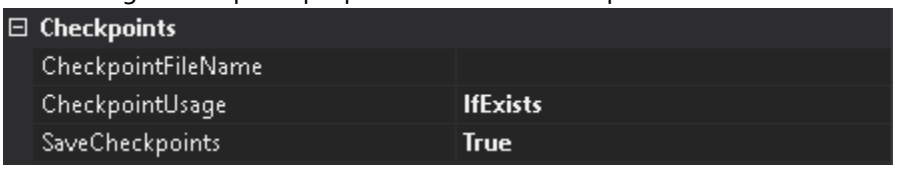


5. Create a new Project Parameter, "CheckpointPath". Set it to String value with default value C:\Temp\ (confirm the path exists, if not create it).



Name	Data type	Value	Sensitive	Required
CheckpointPath	String	C:\Temp\	False	True

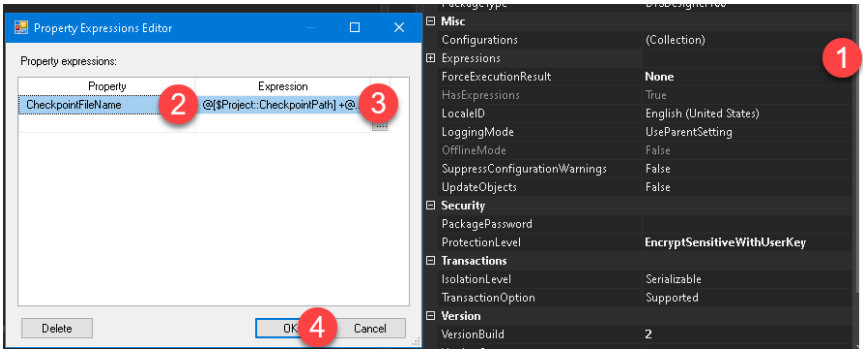
6. Set Package Checkpoint properties – Leave CheckpointFileName as blank.



Checkpoints	
CheckpointFileName	
CheckpointUsage	IfExists
SaveCheckpoints	True

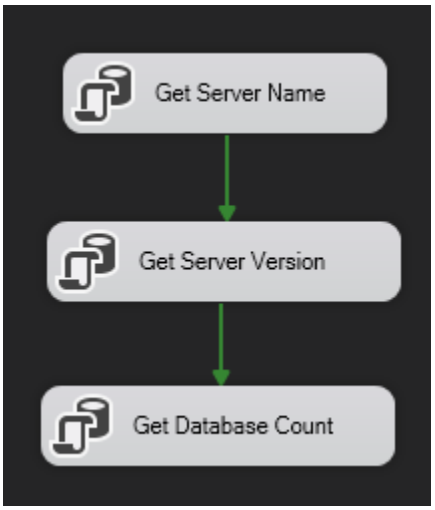
7. The CheckpointFileName should be dynamic, therefore we must use expressions to set the value.

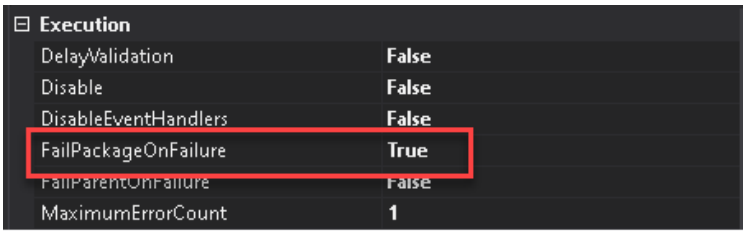
   Expression: @[$Project::CheckpointPath] +@[System::PackageName] + "_SSIS_Checkpoint.xml"



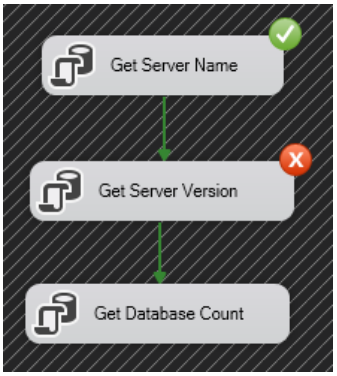
8. Create three "Execute SQL Task" and respective SQL code and link it to connection created in step #1. Set the property "FailPackageOnFailure" to true for each task.

Task Name	SQL Code
Get Server Name	SELECT @@ServerName
Get Server Version	SELECT @@Versions
Get Database Count	SELECT COUNT(*) AS DBCount FROM sys.databases





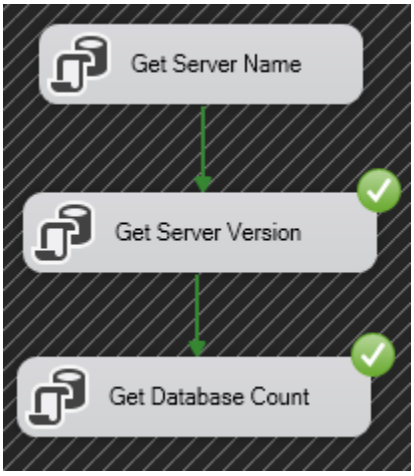
9. Execute the task, it should fail at "Get Server Version."



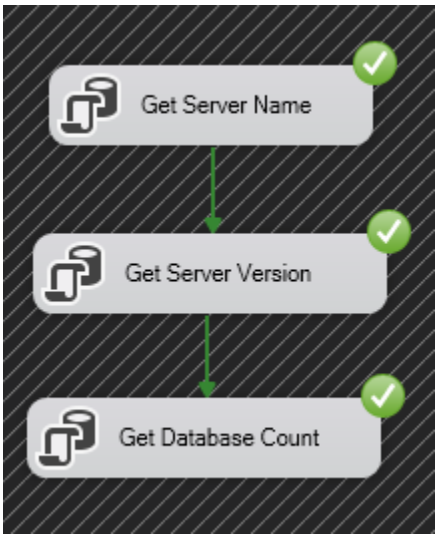
10. Review the checkpoint file that got created in C:\Temp\ (reformatted for lab).

```
<DTS:Checkpoint xmlns:DTS="www.microsoft.com/SqlServer/Dts" DTS:PackageID="{F93B1C31-D6D7-4210-AAA0-4B9AC0C0EB0D}">
    <DTS:Variables DTS:ContID="{F93B1C31-D6D7-4210-AAA0-4B9AC0C0EB0D}"/>
    <DTS:Container DTS:ContID="{0A386E59-1745-4279-A219-83F1A1E92669}" DTS:Result="0" DTS:PrecedenceMap=""/>
</DTS:Checkpoint>
```

11. The package failed because "Get Server Version" SQL code is incorrect. Update the code to "SELECT @@Version" and rerun. Notice this time, it skips the first SQL Task due to check point.



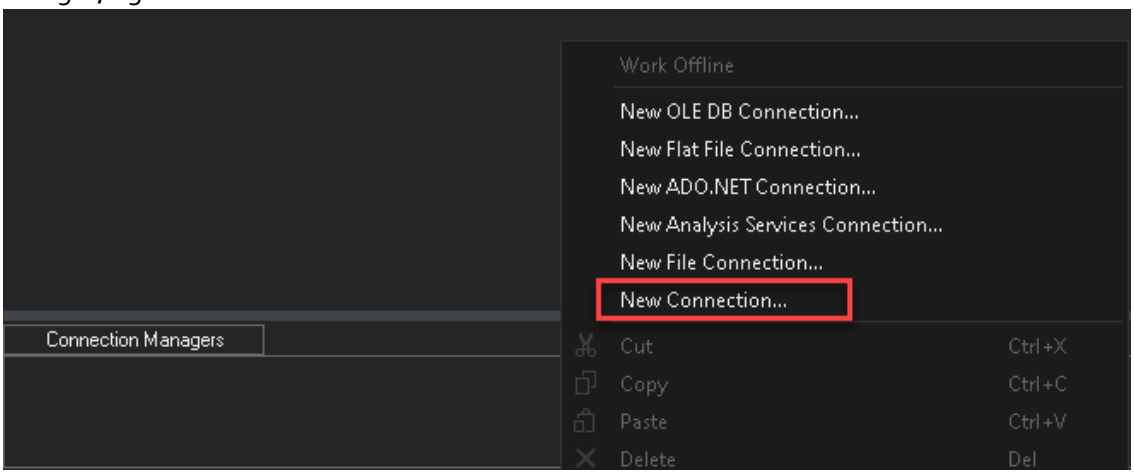
12. Review the C:\Temp\ notice the checkpoint should have been deleted at the end of execution.
13. Re-run the package. This time all three steps run, because the check point file was deleted after previous execution.



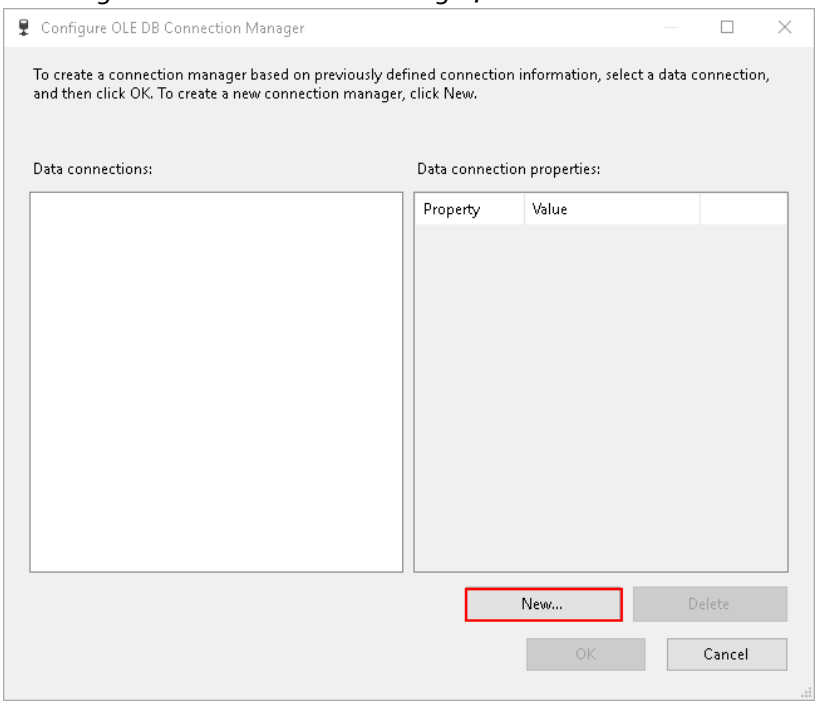
MODULE 04 – LAB 03 – EXERCISE 02: TRANSACTIONS

Note: Instructions in this exercise are not as detailed intentionally. Review previous labs if unsure how to complete some steps.

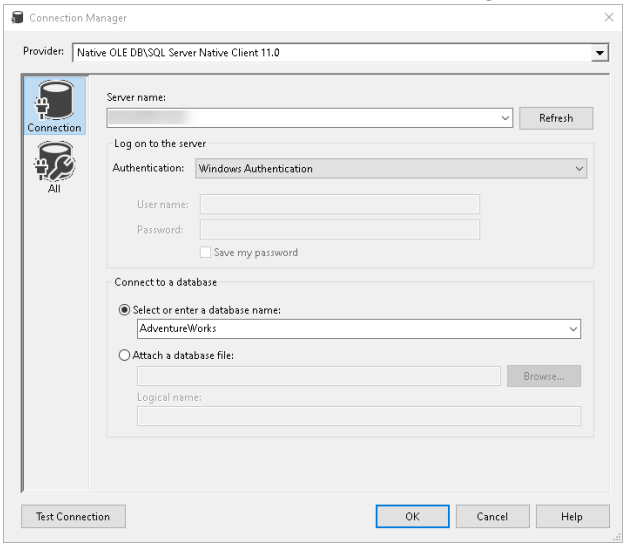
1. Create new integration services project.
2. Set up a connection manager for the database. In the bottom center pane under Connection Manger, right-click select New Ole-DB Connection.



3. In Configure OLE DB Connect Manager, click New.



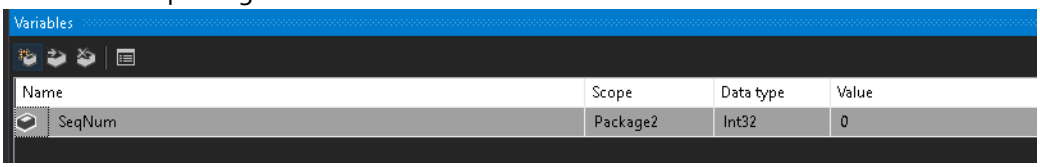
4. Setup connection to SQL Server pointing to the AdventureWorks database.



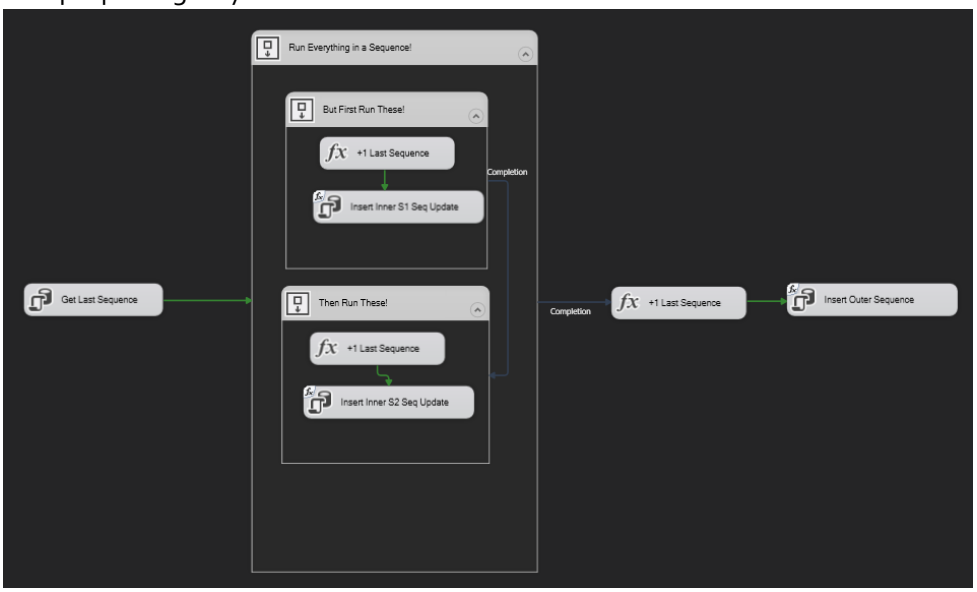
5. Create a new table in SQL Server.

```
CREATE TABLE dbo.TxSequences (SeqNum INT, CallLoc VARCHAR(255), DateSaved
DATETIME)
```

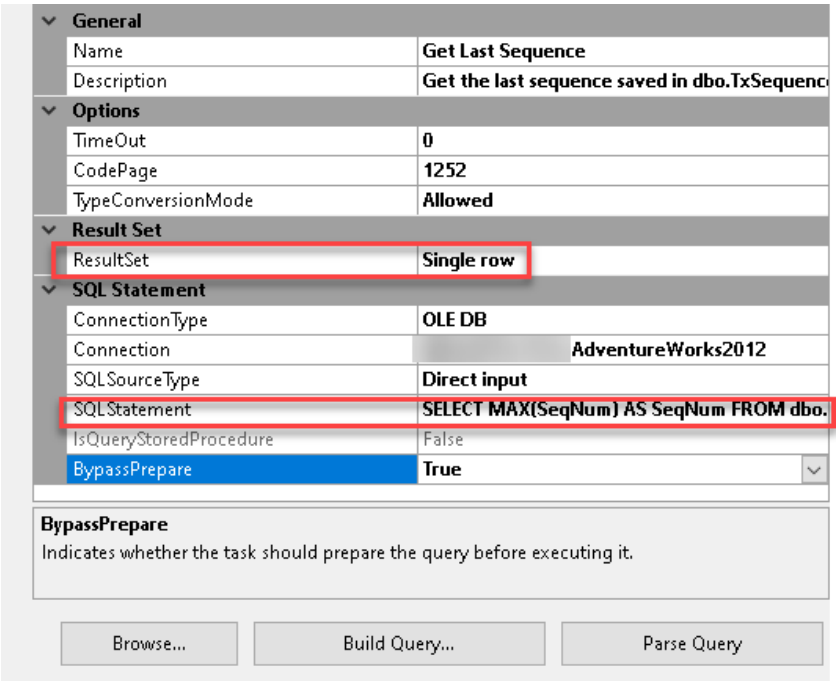
6. Create a new package variable.



7. Setup a package layout as below.

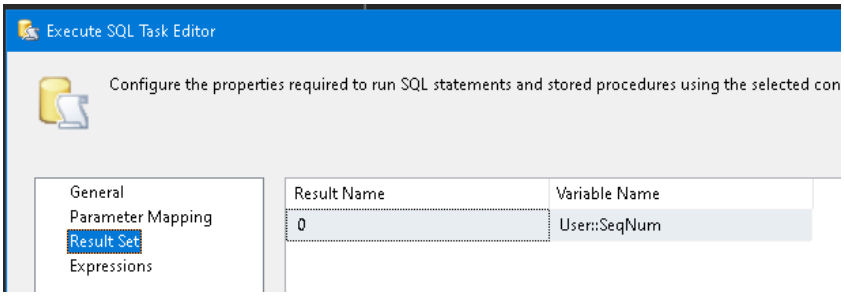


8. Update "Get Last Sequence".




```
SELECT MAX(SeqNum) AS SeqNum FROM dbo.TxSequences
```

9. Output results to variable created in step #6.



10. Update each Expression Tasks (three in total) to following expression.

```
@[User::SeqNum] = @[User::SeqNum]+1
```

11. Update each Execute SQL Task (excluding Get Last Sequence) to use expression for SQL string. Use the following expression.

```
"INSERT INTO dbo.TxSequences VALUES (" + (DT_WSTR,5) @[User::SeqNum] + ", 'Inner S1', GETDATE())"
```

12. Execute the package after all the updates are completed. The default execution should run without problem.

13. Verify the results in the SQL Server with the following code.

```
SELECT *
  FROM dbo.TxSequences

DELETE
  FROM dbo.TxSequences
```

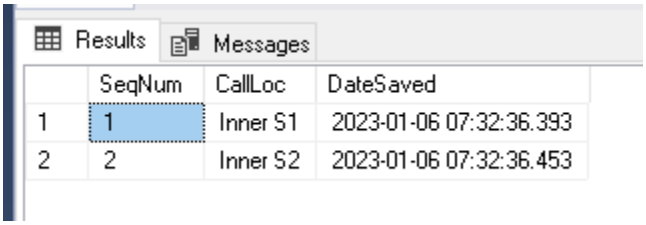
Results | Messages

	SeqNum	CallLoc	DateSaved
1	1	Inner S1	2023-01-06 07:29:46.643
2	2	Inner S2	2023-01-06 07:29:46.690
3	3	Outer	2023-01-06 07:29:46.730

14. Next modify the "Insert Outer Sequence" SQL string expression to the following value.

```
"INSERT INTO dbo.TxSequences VALUES (1," + (DT_WSTR,5) @[User::SeqNum] + ",
'Outer', GETDATE())"
```

15. Execute the package and review SQL data as in step #13. We should get only two rows. Because by default SSIS each container/task is their own transactions.


Results Messages

	SeqNum	CallLoc	DateSaved
1	1	Inner S1	2023-01-06 07:32:36.393
2	2	Inner S2	2023-01-06 07:32:36.453

16. Next change the package property **TransactionOption** to **Required** and rerun the package. Review SQL data as in step #13. This time we get no rows because the entire package is built as a single transaction.
17. Next change the package property **TransactionOption** back to **Supported** but change the outer sequence "Run Everything in a Sequence!" **TransactionOption** to **Required**. Rerun the package and review the results. We get results like #15.

.