# 2. UNION and UNION all

select \* from inserted

union

select \* from deleted

select convert(char(5),'hi') as Greeting

union all

select convert(char(11),'hello there') as GreetingNow

union all

select convert(char(11),'bonjour')

union all

select convert(char(11),'hi')

select convert(tinyint, 45) as Mycolumn

union

select convert(bigint, 456)

select 4

union

select 'hi there'

# 3. Except and Intersect

select \*, Row\_Number() over(order by (select null)) % 3 as ShouldIDelete

--into tblTransactionNew

from tblTransaction

delete from tblTransactionNew

where ShouldIDelete = 1

update tblTransactionNew

set DateOfTransaction = dateadd(day,1,DateOfTransaction)

Where ShouldIDelete = 2

alter table tblTransactionNew

drop column ShouldIDelete

select \* from tblTransaction -- 2486 rows

intersect--except--union--union all

select \* from tblTransactionNew -- 1657 rows, 829 changed rows, 828 unchanged

order by EmployeeNumber

# 4. CASE

declare @myOption as varchar(10) = 'Option C'

select case when @myOption = 'Option A' then 'First option'

 when @myOption = 'Option B' then 'Second option'

 --else 'No Option'

 END as MyOptions

go

declare @myOption as varchar(10) = 'Option A'

select case @myOption when 'Option A' then 'First option'

 when 'Option B' then 'Second option'

 else 'No Option' END as MyOptions

go

 case when left(EmployeeGovernmentID,1)='A' then 'Letter A'

 when EmployeeNumber<200 then 'Less than 200'

 else 'Neither letter' END + '.' as myCol

 FROM tblEmployee

# 5. Isnull and Coalesce

select \* from tblEmployee where EmployeeMiddleName is null

declare @myOption as varchar(10) = 'Option B'

select isnull(@myOption, 'No Option') as MyOptions

go

declare @myFirstOption as varchar(10) --= 'Option A'

declare @mySecondOption as varchar(10) --= 'Option B'

select coalesce(@myFirstOption, @mySecondOption, 'No option') as MyOptions

go

select isnull('ABC',1) as MyAnswer

select coalesce('ABC',1) as MyOtherAnswer

go

select isnull(null,null) as MyAnswer

select coalesce(null,null) as MyOtherAnswer

go

create table tblExample

(myOption nvarchar(10) null)

go

insert into tblExample (myOption)

values ('Option A')

select coalesce(myOption, 'No option') as MyOptions

into tblIsCoalesce

from tblExample

select case when myOption is not null then myOption else 'No option' end as myOptions from tblExample

go

select isnull(myOption, 'No option') as MyOptions

into tblIsNull

from tblExample

go

drop table tblExample

drop table tblIsCoalesce

drop table tblIsNull

# 7. Let’s build our MERGE statement

BEGIN TRAN

MERGE INTO tblTransaction as T

USING tblTransactionNew as S

ON T.EmployeeNumber = S.EmployeeNumber AND T.DateOfTransaction = S.DateOfTransaction

WHEN MATCHED THEN

 UPDATE SET Amount = T.Amount + S.Amount

WHEN NOT MATCHED BY TARGET THEN

 INSERT ([Amount], [DateOfTransaction], [EmployeeNumber])

 VALUES (S.Amount, S.DateOfTransaction, S.EmployeeNumber);

ROLLBACK TRAN

-- tblTransaction (no) - tblTransactionNew (yes)

-- 1 tblTransaction - 1 tblTransactionNew

-- 1 tblTransaction - multiple rows TblTransactionNew

# 8. Let’s expand our MERGE statement

SELECT DateOfTransaction, EmployeeNumber, COUNT(\*) AS NumberOfRows

FROM tblTransactionNew

GROUP BY DateOfTransaction, EmployeeNumber

HAVING COUNT(\*)>1

BEGIN TRAN

go

DISABLE TRIGGER TR\_tblTransaction ON dbo.tblTransaction

GO

MERGE INTO tblTransaction as T

USING (SELECT DateOfTransaction, EmployeeNumber, MIN(Amount) as Amount

 FROM tblTransactionNew

 GROUP BY DateOfTransaction, EmployeeNumber) as S

ON T.EmployeeNumber = S.EmployeeNumber AND

 T.DateOfTransaction = S.DateOfTransaction

WHEN MATCHED THEN

 UPDATE SET Amount = T.Amount + S.Amount

WHEN NOT MATCHED THEN

 INSERT (Amount, DateOfTransaction, EmployeeNumber)

 VALUES (S.Amount, S.DateOfTransaction, S.EmployeeNumber)

 OUTPUT deleted.\*, inserted.\*;

ROLLBACK TRAN

# 9. Merge with additional column

BEGIN TRAN

ALTER TABLE tblTransaction

ADD Comments varchar(50) NULL

GO -- DDL

MERGE TOP (5) PERCENT INTO tblTransaction as T --DML

USING (select EmployeeNumber, DateOfTransaction, sum(Amount) as Amount

from tblTransactionNew

group by EmployeeNumber, DateOfTransaction) as S

ON T.EmployeeNumber = S.EmployeeNumber AND T.DateOfTransaction = S.DateOfTransaction

WHEN MATCHED AND T.Amount + S.Amount >0 THEN

 UPDATE SET Amount = T.Amount + S.Amount, Comments = 'Updated Row'

WHEN MATCHED THEN

 DELETE

WHEN NOT MATCHED BY TARGET THEN

 INSERT ([Amount], [DateOfTransaction], [EmployeeNumber], Comments)

 VALUES (S.Amount, S.DateOfTransaction, S.EmployeeNumber, 'Inserted Row')

WHEN NOT MATCHED BY SOURCE THEN

 UPDATE SET Comments = 'Unchanged'

OUTPUT inserted.\*, deleted.\* , $action;

--Select \* from tblTransaction ORDER BY EmployeeNumber, DateOfTransaction

ROLLBACK TRAN

# 11. Let’s create our first procedure

create proc NameEmployees as

begin

 select EmployeeNumber, EmployeeFirstName, EmployeeLastName

 from tblEmployee

end

go

NameEmployees

execute NameEmployees

exec NameEmployees

# 12. Ask for a specific employee

--if exists (select \* from sys.procedures where name='NameEmployees')

if object\_ID('NameEmployees','P') IS NOT NULL

drop proc NameEmployees

go

create proc NameEmployees(@EmployeeNumber int) as

begin

 if exists (Select \* from tblEmployee where EmployeeNumber = @EmployeeNumber)

 begin

 select EmployeeNumber, EmployeeFirstName, EmployeeLastName

 from tblEmployee

 where EmployeeNumber = @EmployeeNumber

 end

end

go

NameEmployees 4

execute NameEmployees 223

exec NameEmployees 323

select EmployeeNumber from NameEmployees

DECLARE @EmployeeName int = 123

select @EmployeeName

# 13. Different outcomes

--if exists (select \* from sys.procedures where name='NameEmployees')

if object\_ID('NameEmployees','P') IS NOT NULL

drop proc NameEmployees

go

create proc NameEmployees(@EmployeeNumber int) as

begin

 if exists (Select \* from tblEmployee where EmployeeNumber = @EmployeeNumber)

 begin

 if @EmployeeNumber < 300

 begin

 select EmployeeNumber, EmployeeFirstName, EmployeeLastName

 from tblEmployee

 where EmployeeNumber = @EmployeeNumber

 end

 else

 begin

 select EmployeeNumber, EmployeeFirstName, EmployeeLastName, Department

 from tblEmployee

 where EmployeeNumber = @EmployeeNumber

 select \* from tblTransaction where EmployeeNumber = @EmployeeNumber

 end

 end

end

go

NameEmployees 4

execute NameEmployees 223

exec NameEmployees 324

# 14. Ask for a range of employees

--if exists (select \* from sys.procedures where name='NameEmployees')

if object\_ID('NameEmployees','P') IS NOT NULL

drop proc NameEmployees

go

create proc NameEmployees(@EmployeeNumberFrom int, @EmployeeNumberTo int) as

begin

 if exists (Select \* from tblEmployee where EmployeeNumber between @EmployeeNumberFrom and @EmployeeNumberTo)

 begin

 select EmployeeNumber, EmployeeFirstName, EmployeeLastName

 from tblEmployee

 where EmployeeNumber between @EmployeeNumberFrom and @EmployeeNumberTo

 end

end

go

NameEmployees 4, 5

execute NameEmployees 223, 227

exec NameEmployees @EmployeeNumberFrom = 323, @EmployeeNumberTo = 327

# 15. A different SELECT statement per employee

--if exists (select \* from sys.procedures where name='NameEmployees')

if object\_ID('NameEmployees','P') IS NOT NULL

drop proc NameEmployees

go

create proc NameEmployees(@EmployeeNumberFrom int, @EmployeeNumberTo int) as

begin

 if exists (Select \* from tblEmployee where EmployeeNumber between @EmployeeNumberFrom and @EmployeeNumberTo)

 begin

 declare @EmployeeNumber int = @EmployeeNumberFrom

 while @EmployeeNumber <= @EmployeeNumberTo

 BEGIN

 if exists (Select \* from tblEmployee where EmployeeNumber = @EmployeeNumber)

 select EmployeeNumber, EmployeeFirstName, EmployeeLastName

 from tblEmployee

 where EmployeeNumber = @EmployeeNumber

 SET @EmployeeNumber = @EmployeeNumber + 1

 END

 end

end

go

NameEmployees 4, 5

execute NameEmployees 223, 227

exec NameEmployees @EmployeeNumberFrom = 323, @EmployeeNumberTo = 1327

# 16. Returning values

--if exists (select \* from sys.procedures where name='NameEmployees')

if object\_ID('NameEmployees','P') IS NOT NULL

drop proc NameEmployees

go

create proc NameEmployees(@EmployeeNumberFrom int, @EmployeeNumberTo int, @NumberOfRows int OUTPUT) as

begin

 if exists (Select \* from tblEmployee where EmployeeNumber between @EmployeeNumberFrom and @EmployeeNumberTo)

 begin

 select EmployeeNumber, EmployeeFirstName, EmployeeLastName

 from tblEmployee

 where EmployeeNumber between @EmployeeNumberFrom and @EmployeeNumberTo

 SET @NumberOfRows = @@ROWCOUNT

 RETURN 0

 end

 ELSE

 BEGIN

 SET @NumberOfRows = 0

 RETURN 1

 END

end

go

DECLARE @NumberRows int, @ReturnStatus int

EXEC @ReturnStatus = NameEmployees 4, 5, @NumberRows OUTPUT

select @NumberRows as MyRowCount, @ReturnStatus as Return\_Status

GO

DECLARE @NumberRows int, @ReturnStatus int

execute @ReturnStatus = NameEmployees 4, 327, @NumberRows OUTPUT

select @NumberRows as MyRowCount, @ReturnStatus as Return\_Status

GO

DECLARE @NumberRows int, @ReturnStatus int

exec @ReturnStatus = NameEmployees @EmployeeNumberFrom = 323, @EmployeeNumberTo = 327, @NumberOfRows = @NumberRows OUTPUT

select @NumberRows as MyRowCount, @ReturnStatus as Return\_Status

# 19. Try … Catch

--if exists (select \* from sys.procedures where name='AverageBalance')

if object\_ID('AverageBalance','P') IS NOT NULL

drop proc AverageBalance

go

create proc AverageBalance(@EmployeeNumberFrom int, @EmployeeNumberTo int, @AverageBalance int OUTPUT) as

begin

 SET NOCOUNT ON

 declare @TotalAmount money

 declare @NumOfEmployee int

 begin try

 select @TotalAmount = sum(Amount) from tblTransaction

 where EmployeeNumber between @EmployeeNumberFrom and @EmployeeNumberTo

 select @NumOfEmployee = count(distinct EmployeeNumber) from tblEmployee

 where EmployeeNumber between @EmployeeNumberFrom and @EmployeeNumberTo

 set @AverageBalance = @TotalAmount / @NumOfEmployee

 RETURN 0

 end try

 begin catch

 set @AverageBalance = 0

 SELECT ERROR\_MESSAGE() AS ErrorMessage, ERROR\_LINE() as ErrorLine,

 ERROR\_NUMBER() as ErrorNumber, ERROR\_PROCEDURE() as ErrorProcedure,

 ERROR\_SEVERITY() as ErrorSeverity, -- 0-10 for information

 -- 16 default SQL SERVER log / Windows Application log

 -- 20-25

 ERROR\_STATE() as ErrorState

 RETURN 1

 end catch

end

go

DECLARE @AvgBalance int, @ReturnStatus int

EXEC @ReturnStatus = AverageBalance 4, 5, @AvgBalance OUTPUT

select @AvgBalance as Average\_Balance, @ReturnStatus as Return\_Status

GO

DECLARE @AvgBalance int, @ReturnStatus int

execute @ReturnStatus = AverageBalance 223, 227, @AvgBalance OUTPUT

select @AvgBalance as Average\_Balance, @ReturnStatus as Return\_Status

GO

DECLARE @AvgBalance int, @ReturnStatus int

exec @ReturnStatus = AverageBalance @EmployeeNumberFrom = 323, @EmployeeNumberTo = 327, @AverageBalance = @AvgBalance OUTPUT

select @AvgBalance as Average\_Balance, @ReturnStatus as Return\_Status

SELECT TRY\_CONVERT(int, 'two')

# 21. Print

--if exists (select \* from sys.procedures where name='AverageBalance')

if object\_ID('AverageBalance','P') IS NOT NULL

drop proc AverageBalance

go

create proc AverageBalance(@EmployeeNumberFrom int, @EmployeeNumberTo int, @AverageBalance int OUTPUT) as

begin

 SET NOCOUNT ON

 declare @TotalAmount decimal(5,2)

 declare @NumOfEmployee int

 begin try

 print 'The employee numbers are from ' + convert(varchar(10),@EmployeeNumberFrom)

 + ' to ' + convert(varchar(10),@EmployeeNumberTo)

 select @TotalAmount = sum(Amount) from tblTransaction

 where EmployeeNumber between @EmployeeNumberFrom and @EmployeeNumberTo

 select @NumOfEmployee = count(distinct EmployeeNumber) from tblEmployee

 where EmployeeNumber between @EmployeeNumberFrom and @EmployeeNumberTo

 set @AverageBalance = @TotalAmount / @NumOfEmployee

 RETURN 0

 end try

 begin catch

 set @AverageBalance = 0

 if ERROR\_NUMBER() = 8134 -- @@ERROR

 begin

 set @AverageBalance = 0

 print 'There are no valid employees in this range.'

 Return 8134

 end

 else

 declare @ErrorMessage as varchar(255)

 select @ErrorMessage = error\_Message()

 raiserror (@ErrorMessage, 16, 1)

 --throw 56789, 'Too many flanges', 1

 -- PRINT ERROR\_MESSAGE() AS ErrorMessage, ERROR\_LINE() as ErrorLine, ERROR\_NUMBER() as ErrorNumber, ERROR\_PROCEDURE() as ErrorProcedure, ERROR\_SEVERITY() as ErrorSeverity, -- 0-10 for information

 -- 16 default SQL SERVER log / Windows Application log

 -- 20-25

 -- ERROR\_STATE() as ErrorState

 --RETURN 1

 select 'Hi There'

 end catch

end

go

DECLARE @AvgBalance int, @ReturnStatus int

EXEC @ReturnStatus = AverageBalance 4, 5, @AvgBalance OUTPUT

select @AvgBalance as Average\_Balance, @ReturnStatus as Return\_Status

GO

DECLARE @AvgBalance int, @ReturnStatus int

execute @ReturnStatus = AverageBalance 223, 227, @AvgBalance OUTPUT

select @AvgBalance as Average\_Balance, @ReturnStatus as Return\_Status, 'Error did not stop us' as myMessage

GO

DECLARE @AvgBalance int, @ReturnStatus int

exec @ReturnStatus = AverageBalance @EmployeeNumberFrom = 323, @EmployeeNumberTo = 327, @AverageBalance = @AvgBalance OUTPUT

select @AvgBalance as Average\_Balance, @ReturnStatus as Return\_Status