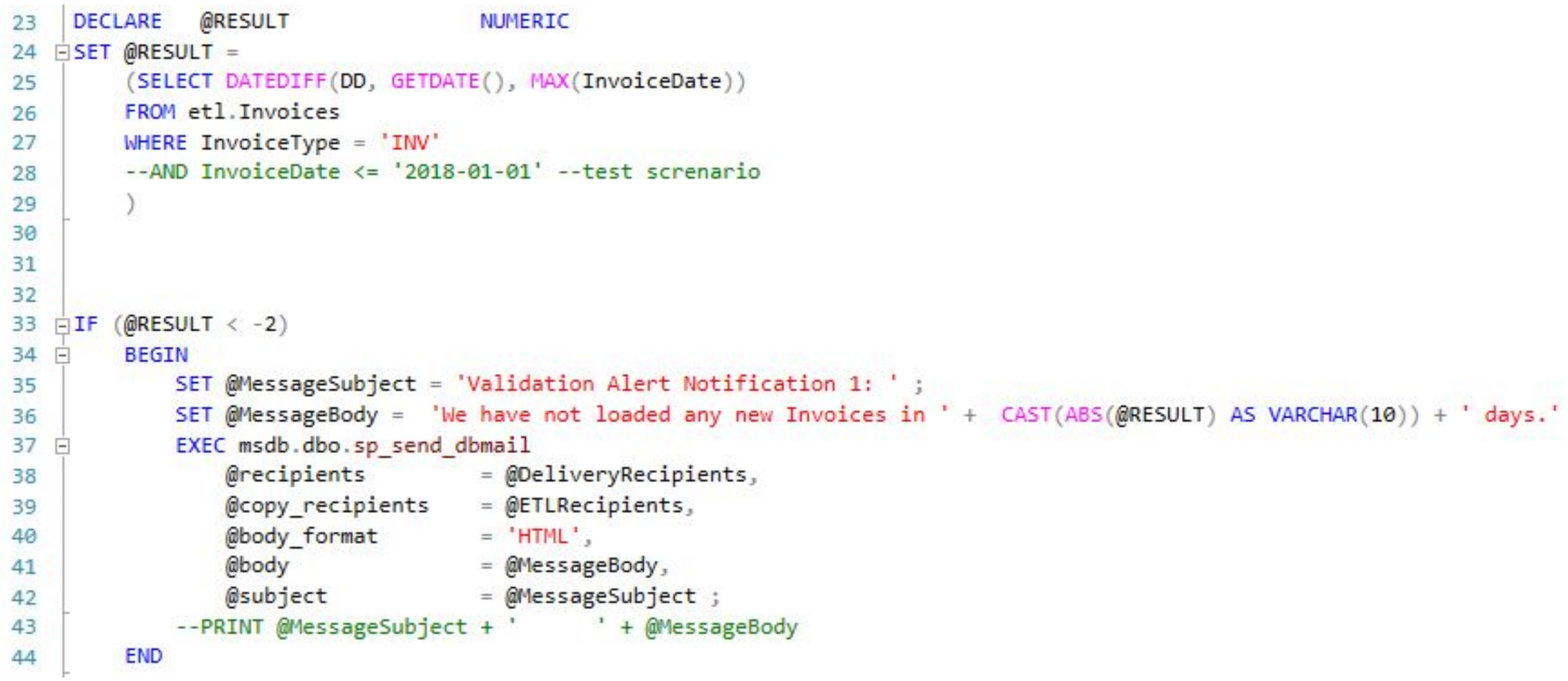
The following is a capstone presentation that showcases some of my work in creating a data validation process for automated data loads in SQL Server. Some information has been redacted to protect client confidentiality.

INTRODUCTION

Overview and Goals

Sample Procedure

- Current procedure had three validation measures
  - Invoice Date
  - Price Changes
  - Revenue



```
DECLARE   @RESULT                NUMERIC
SET @RESULT =
    (SELECT DATEDIFF(DD, GETDATE(), MAX(InvoiceDate))
    FROM etl.Invoices
    WHERE InvoiceType = 'INV'
    --AND InvoiceDate <= '2018-01-01' --test screnario
    )



IF (@RESULT < -2)
    BEGIN
        SET @MessageSubject = 'Validation Alert Notification 1: ' ;
        SET @MessageBody =  'We have not loaded any new Invoices in ' +  CAST(ABS(@RESULT) AS VARCHAR(10)) + ' days.'
        EXEC msdb.dbo.sp_send_dbmail
            @recipients         = @DeliveryRecipients,
            @copy_recipients    = @ETLRecipients,
            @body_format        = 'HTML',
            @body               = @MessageBody,
            @subject            = @MessageSubject ;
        --PRINT @MessageSubject + '      ' + @MessageBody
    END
```

Research

Client Team Meetings

- Figure out the metrics that are important to each client
- Establish thresholds and determine action upon failure

Data Lead Meeting

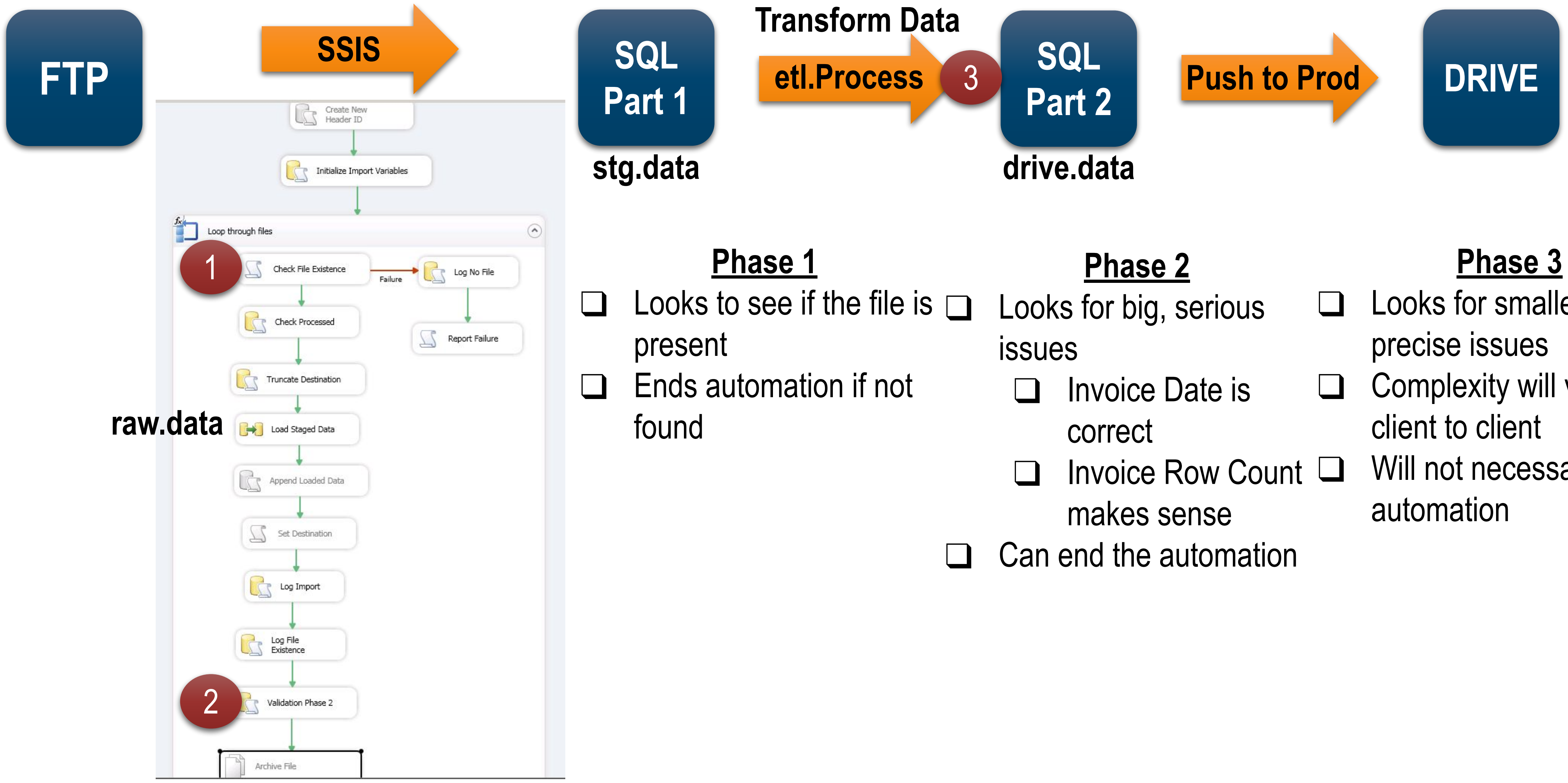
- Gain deeper understanding of the structure of each database
- Go over clarifications specific to each database

Preparation of Structure

- Individual creation of automated validation process
- Reach agreement on joint procedures

THE PROCESS

General Data Automation Process with Validation Phases



Phase 1

- ❑ Looks to see if the file is present
- ❑ Ends automation if not found

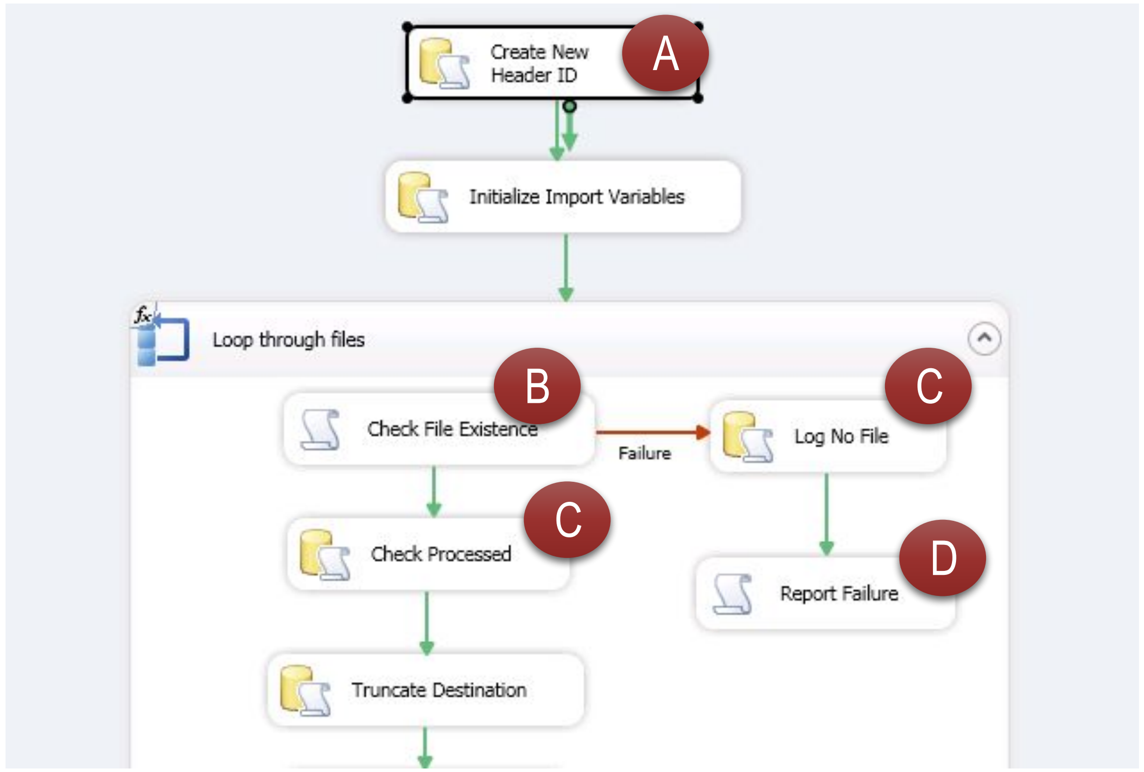
Phase 2

- ❑ Looks for big, serious issues
  - ❑ Invoice Date is correct
  - ❑ Invoice Row Count makes sense
- ❑ Can end the automation

Phase 3

- ❑ Looks for smaller, more precise issues
- ❑ Complexity will vary from client to client
- ❑ Will not necessarily end automation

Phase One



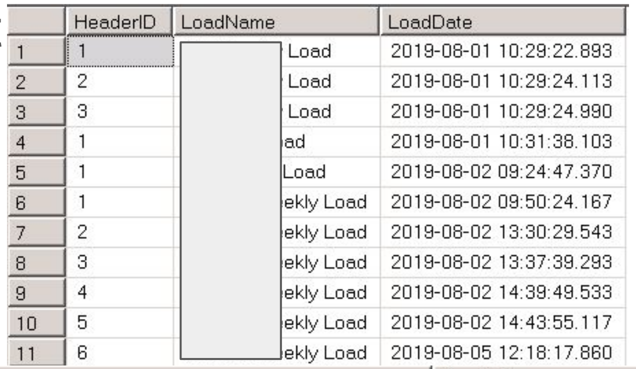
A. Create new entry in the Load ID table

B. Check to see if file exists in the specified location

C. If success – continue with the load. If failure – log failure

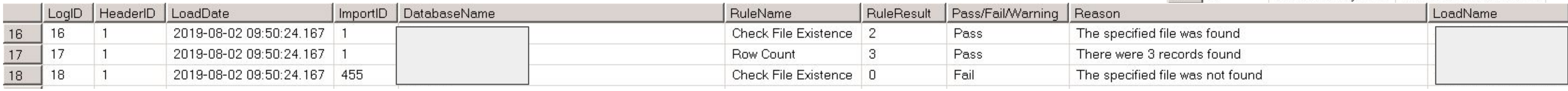
D. On failure – report failure, exit job, send email reporting that the file does not exist

Load ID Table



	HeaderID	LoadName	LoadDate
1	1	[redacted] Load	2019-08-01 10:29:22.893
2	2	[redacted] Load	2019-08-01 10:29:24.113
3	3	[redacted] Load	2019-08-01 10:29:24.990
4	1	[redacted] ad	2019-08-01 10:31:38.103
5	1	[redacted] Load	2019-08-02 09:24:47.370
6	1	[redacted] ekly Load	2019-08-02 09:50:24.167
7	2	[redacted] ekly Load	2019-08-02 13:30:29.543
8	3	[redacted] ekly Load	2019-08-02 13:37:39.293
9	4	[redacted] ekly Load	2019-08-02 14:39:49.533
10	5	[redacted] ekly Load	2019-08-02 14:43:55.117
11	6	[redacted] ekly Load	2019-08-05 12:18:17.860

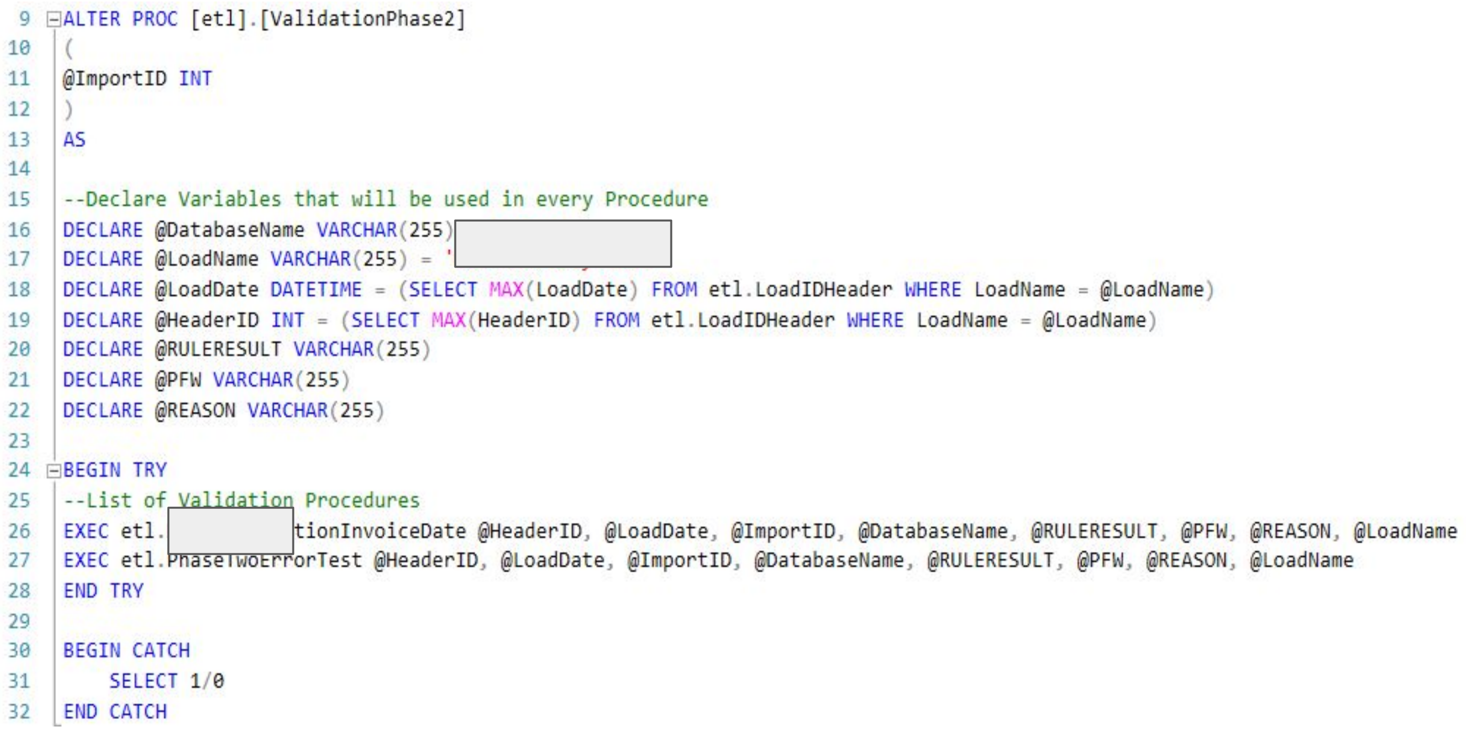
Validation Log Table



	LogID	HeaderID	LoadDate	ImportID	DatabaseName	RuleName	RuleResult	Pass/Fail/Warning	Reason	LoadName
16	16	1	2019-08-02 09:50:24.167	1	[redacted]	Check File Existence	2	Pass	The specified file was found	[redacted]
17	17	1	2019-08-02 09:50:24.167	1	[redacted]	Row Count	3	Pass	There were 3 records found	[redacted]
18	18	1	2019-08-02 09:50:24.167	455	[redacted]	Check File Existence	0	Fail	The specified file was not found	[redacted]

Phase Two of Validation

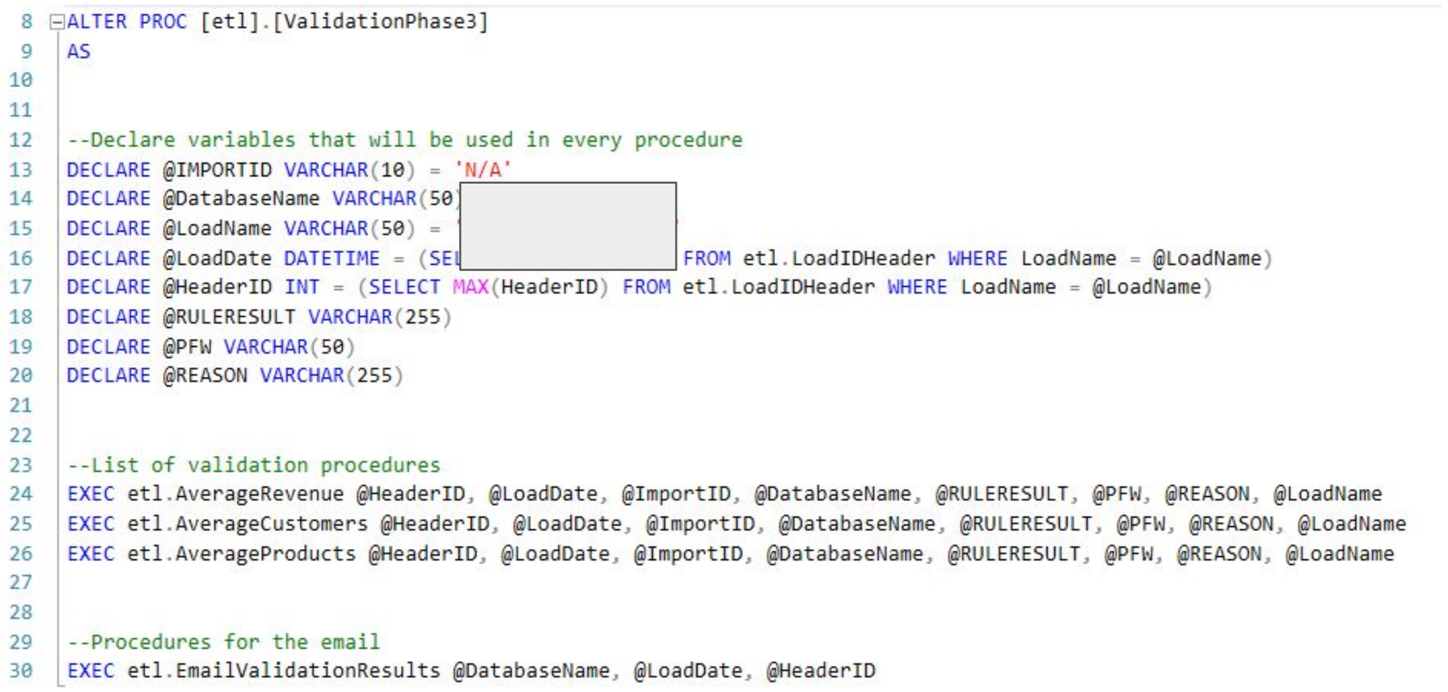
- Found in the SSIS Package
- Looks for big issues where the data load would end if these issues are found
  - Invoice Date is incorrect
  - Invoice Row Count is severely off
- If one procedure fails, it will not go on to the next
- If you are using a SQL Agent Job, this procedure will cause the SSIS package to error out, thus causing the Agent Job to fail too
- Email is sent out if any of the validation procedures fail



```sql
ALTER PROC [etl].[ValidationPhase2]
(
@ImportID INT
)
AS

--Declare Variables that will be used in every Procedure
DECLARE @DatabaseName VARCHAR(255)[redacted]
DECLARE @LoadName VARCHAR(255) = '
DECLARE @LoadDate DATETIME = (SELECT MAX(LoadDate) FROM etl.LoadIDHeader WHERE LoadName = @LoadName)
DECLARE @HeaderID INT = (SELECT MAX(HeaderID) FROM etl.LoadIDHeader WHERE LoadName = @LoadName)
DECLARE @RULERESULT VARCHAR(255)
DECLARE @PFW VARCHAR(255)
DECLARE @REASON VARCHAR(255)

BEGIN TRY
--List of Validation Procedures
EXEC etl.[redacted]tionInvoiceDate @HeaderID, @LoadDate, @ImportID, @DatabaseName, @RULERESULT, @PFW, @REASON, @LoadName
EXEC etl.PhaseTwoErrorTest @HeaderID, @LoadDate, @ImportID, @DatabaseName, @RULERESULT, @PFW, @REASON, @LoadName
END TRY

BEGIN CATCH
    SELECT 1/0
END CATCH
```

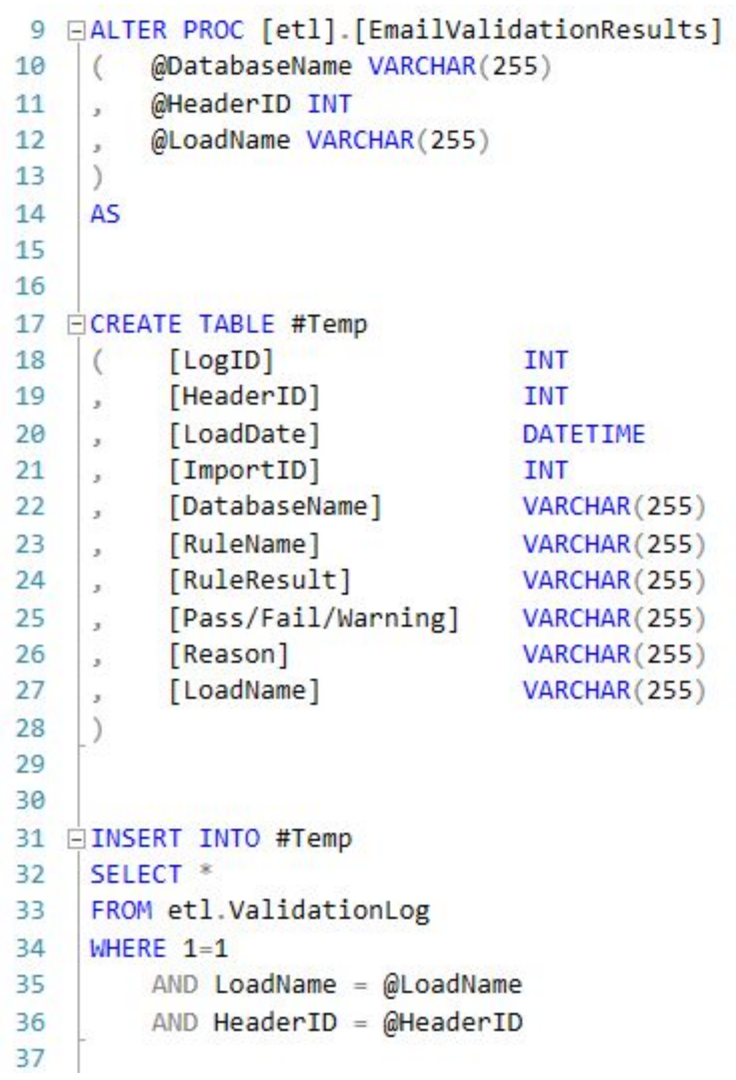
Phase Three of Validation

- Found at the end of or after etl.Process
- Looks for smaller issues in the data load
  - Revenue is 10% less or greater than the previous week's average
  - Number of customers is 10% less or greater than the previous week's average
- If one of the procedures finds an issue, it will log that result as a warning and move on to the next procedure
- Will not necessarily end the data load
- Email is sent out at the very end of this procedure



```
ALTER PROC [etl].[ValidationPhase3]
AS


--Declare variables that will be used in every procedure
DECLARE @IMPORTID VARCHAR(10) = 'N/A'
DECLARE @DatabaseName VARCHAR(50[redacted]
DECLARE @LoadName VARCHAR(50) = '
DECLARE @LoadDate DATETIME = (SEL FROM etl.LoadIDHeader WHERE LoadName = @LoadName)
DECLARE @HeaderID INT = (SELECT MAX(HeaderID) FROM etl.LoadIDHeader WHERE LoadName = @LoadName)
DECLARE @RULERESULT VARCHAR(255)
DECLARE @PFW VARCHAR(50)
DECLARE @REASON VARCHAR(255)


--List of validation procedures
EXEC etl.AverageRevenue @HeaderID, @LoadDate, @ImportID, @DatabaseName, @RULERESULT, @PFW, @REASON, @LoadName
EXEC etl.AverageCustomers @HeaderID, @LoadDate, @ImportID, @DatabaseName, @RULERESULT, @PFW, @REASON, @LoadName
EXEC etl.AverageProducts @HeaderID, @LoadDate, @ImportID, @DatabaseName, @RULERESULT, @PFW, @REASON, @LoadName


--Procedures for the email
EXEC etl.EmailValidationResults @DatabaseName, @LoadDate, @HeaderID
```

Email



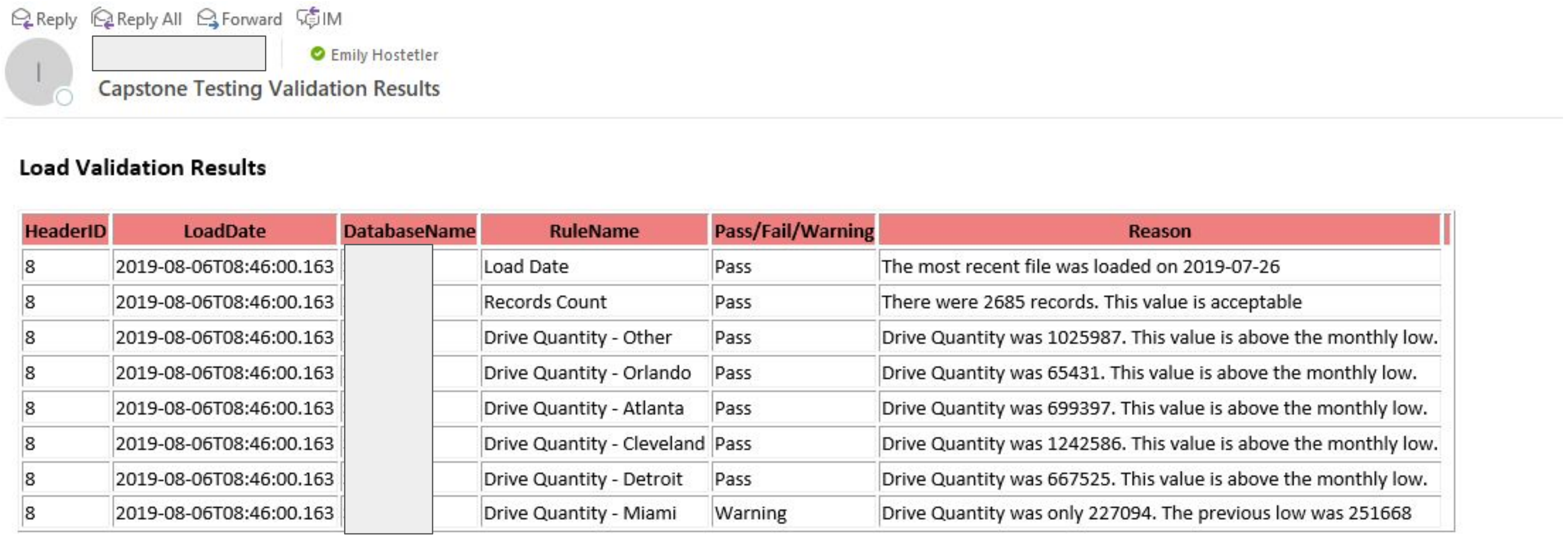
```
ALTER PROC [etl].[EmailValidationResults]
(    @DatabaseName VARCHAR(255)
,    @HeaderID INT
,    @LoadName VARCHAR(255)
)
AS


CREATE TABLE #Temp
(     [LogID]                   INT
,     [HeaderID]                INT
,     [LoadDate]                DATETIME
,     [ImportID]                INT
,     [DatabaseName]            VARCHAR(255)
,     [RuleName]                VARCHAR(255)
,     [RuleResult]              VARCHAR(255)
,     [Pass/Fail/Warning]       VARCHAR(255)
,     [Reason]                  VARCHAR(255)
,     [LoadName]                VARCHAR(255)
)


INSERT INTO #Temp
SELECT *
FROM etl.ValidationLog
WHERE 1=1
    AND LoadName = @LoadName
    AND HeaderID = @HeaderID

```

Selects all entries in the Validation Log Table where the Load Name and Header ID are a match

Inserts those entries into a temporary table

Passes the temporary table through HTML and allows user to set the subject, body, and recipients

Sample email sent after load fails.



Reply Reply All Forward IM

[redacted] Emily Hostetler

Capstone Testing Validation Results

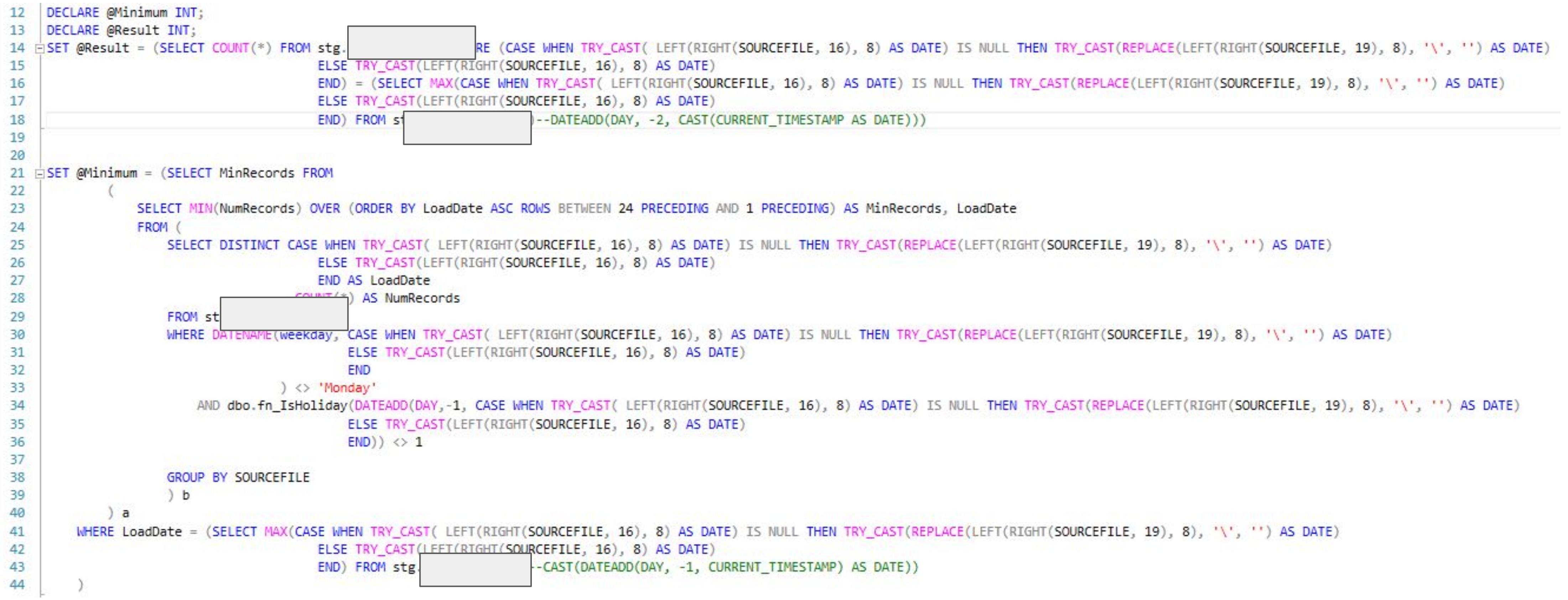
Load Validation Results

HeaderID	LoadDate	DatabaseName	RuleName	Pass/Fail/Warning	Reason
8	2019-08-06T08:46:00.163	[redacted]	Load Date	Pass	The most recent file was loaded on 2019-07-26
8	2019-08-06T08:46:00.163	[redacted]	Records Count	Pass	There were 2685 records. This value is acceptable
8	2019-08-06T08:46:00.163	[redacted]	Drive Quantity - Other	Pass	Drive Quantity was 1025987. This value is above the monthly low.
8	2019-08-06T08:46:00.163	[redacted]	Drive Quantity - Orlando	Pass	Drive Quantity was 65431. This value is above the monthly low.
8	2019-08-06T08:46:00.163	[redacted]	Drive Quantity - Atlanta	Pass	Drive Quantity was 699397. This value is above the monthly low.
8	2019-08-06T08:46:00.163	[redacted]	Drive Quantity - Cleveland	Pass	Drive Quantity was 1242586. This value is above the monthly low.
8	2019-08-06T08:46:00.163	[redacted]	Drive Quantity - Detroit	Pass	Drive Quantity was 667525. This value is above the monthly low.
8	2019-08-06T08:46:00.163	[redacted]	Drive Quantity - Miami	Warning	Drive Quantity was only 227094. The previous low was 251668

CUSTOM PROCEDURES

Custom Procedures – [redacted] Row Count

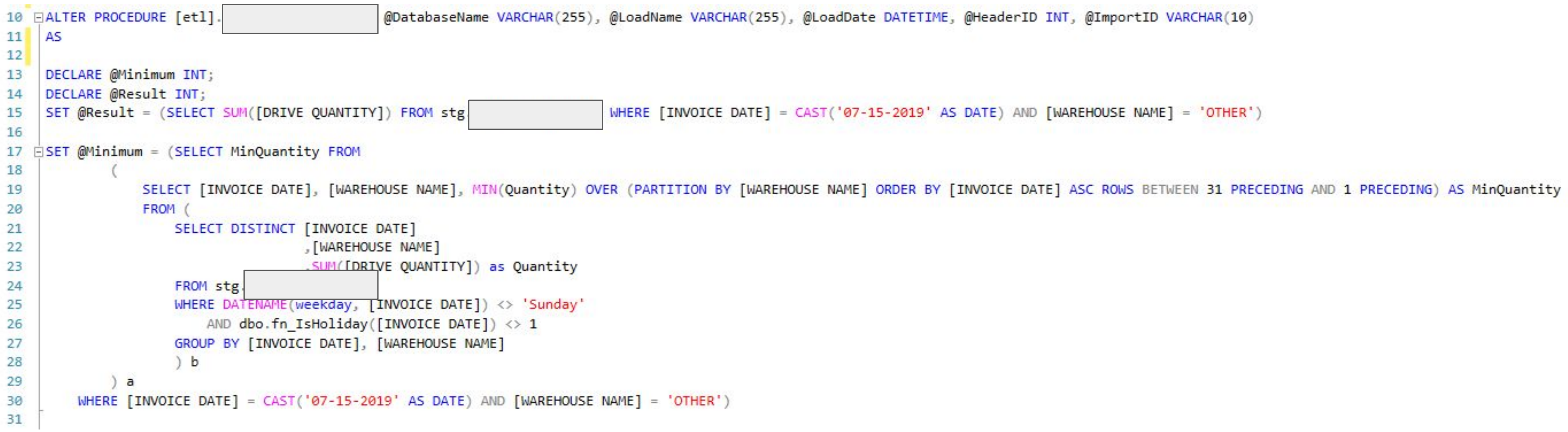
- Goal: Check that the row count is above the minimum for the last 30 days
  - On Pass – record pass and continue
  - On Warn – record warning and continue



```
DECLARE @Minimum INT;
DECLARE @Result INT;
SET @Result = (SELECT COUNT(*) FROM stg.[redacted]RE (CASE WHEN TRY_CAST( LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE) IS NULL THEN TRY_CAST(REPLACE(LEFT(RIGHT(SOURCEFILE, 19), 8), '\', '') AS DATE)
                                    ELSE TRY_CAST(LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE)
                                    END) = (SELECT MAX(CASE WHEN TRY_CAST( LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE) IS NULL THEN TRY_CAST(REPLACE(LEFT(RIGHT(SOURCEFILE, 19), 8), '\', '') AS DATE)
                                    ELSE TRY_CAST(LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE)
                                    END) FROM st[redacted])--DATEADD(DAY, -2, CAST(CURRENT_TIMESTAMP AS DATE)))


SET @Minimum = (SELECT MinRecords FROM
        (
            SELECT MIN(NumRecords) OVER (ORDER BY LoadDate ASC ROWS BETWEEN 24 PRECEDING AND 1 PRECEDING) AS MinRecords, LoadDate
            FROM (
                SELECT DISTINCT CASE WHEN TRY_CAST( LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE) IS NULL THEN TRY_CAST(REPLACE(LEFT(RIGHT(SOURCEFILE, 19), 8), '\', '') AS DATE)
                                    ELSE TRY_CAST(LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE)
                                    END AS LoadDate
                                        ) AS NumRecords
                FROM st[redacted]
                WHERE DATENAME(weekday, CASE WHEN TRY_CAST( LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE) IS NULL THEN TRY_CAST(REPLACE(LEFT(RIGHT(SOURCEFILE, 19), 8), '\', '') AS DATE)
                                        ELSE TRY_CAST(LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE)
                                        END
                               ) <> 'Monday'
                    AND dbo.fn_IsHoliday(DATEADD(DAY,-1, CASE WHEN TRY_CAST( LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE) IS NULL THEN TRY_CAST(REPLACE(LEFT(RIGHT(SOURCEFILE, 19), 8), '\', '') AS DATE)
                                        ELSE TRY_CAST(LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE)
                                        END)) <> 1

                GROUP BY SOURCEFILE
                ) b
        ) a
    WHERE LoadDate = (SELECT MAX(CASE WHEN TRY_CAST( LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE) IS NULL THEN TRY_CAST(REPLACE(LEFT(RIGHT(SOURCEFILE, 19), 8), '\', '') AS DATE)
                                    ELSE TRY_CAST(LEFT(RIGHT(SOURCEFILE, 16), 8) AS DATE)
                                    END) FROM stg.[redacted]-CAST(DATEADD(DAY, -1, CURRENT_TIMESTAMP) AS DATE))
    )
```

Custom Procedures – [redacted] Quantity

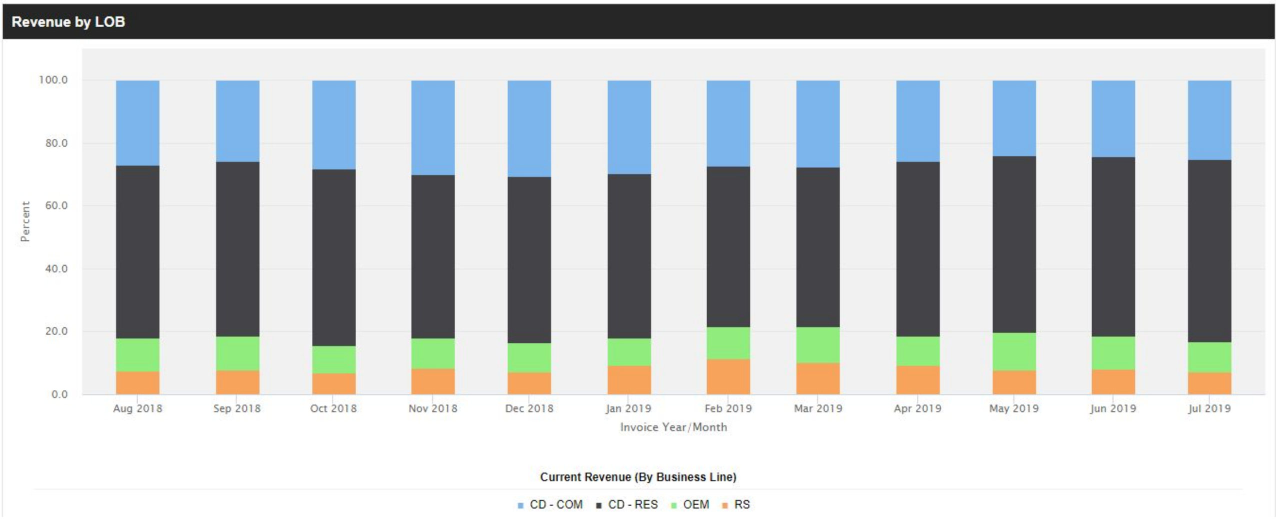
- Goal: Check that the Quantity for each branch or warehouse is above the minimum for the last 30 days
- If the given day is a holiday it doesn't count that as being part of the count.
    - On Pass – Log pass and continue
    - On Warning – Log warning and continue



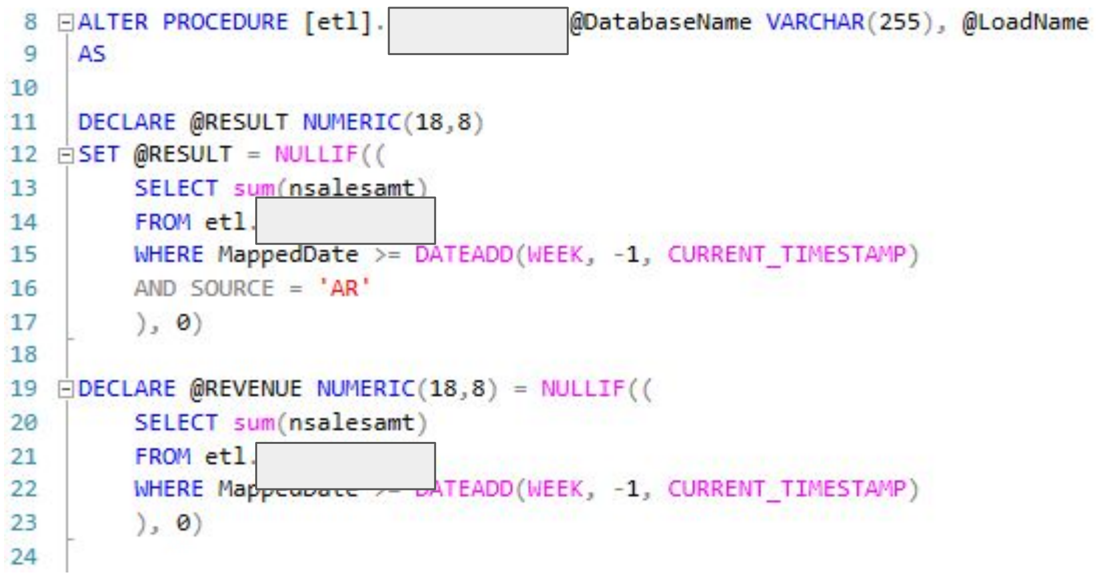
```sql
ALTER PROCEDURE [etl].[redacted] @DatabaseName VARCHAR(255), @LoadName VARCHAR(255), @LoadDate DATETIME, @HeaderID INT, @ImportID VARCHAR(10)
AS

DECLARE @Minimum INT;
DECLARE @Result INT;
SET @Result = (SELECT SUM([DRIVE QUANTITY]) FROM stg[redacted] WHERE [INVOICE DATE] = CAST('07-15-2019' AS DATE) AND [WAREHOUSE NAME] = 'OTHER')

SET @Minimum = (SELECT MinQuantity FROM
        (
            SELECT [INVOICE DATE], [WAREHOUSE NAME], MIN(Quantity) OVER (PARTITION BY [WAREHOUSE NAME] ORDER BY [INVOICE DATE] ASC ROWS BETWEEN 31 PRECEDING AND 1 PRECEDING) AS MinQuantity
            FROM (
                SELECT DISTINCT [INVOICE DATE]
                    ,[WAREHOUSE NAME]
                    ,SUM([DRIVE QUANTITY]) as Quantity
                FROM stg.[redacted]
                WHERE DATENAME(weekday, [INVOICE DATE]) <> 'Sunday'
                    AND dbo.fn_IsHoliday([INVOICE DATE]) <> 1
                GROUP BY [INVOICE DATE], [WAREHOUSE NAME]
                ) b
        ) a
    WHERE [INVOICE DATE] = CAST('07-15-2019' AS DATE) AND [WAREHOUSE NAME] = 'OTHER')
```

Custom Procedures - [redacted]

- Goal: Check Percent of Revenue by Business Line
- The percent of revenue should be pretty consistent across all months for the last year



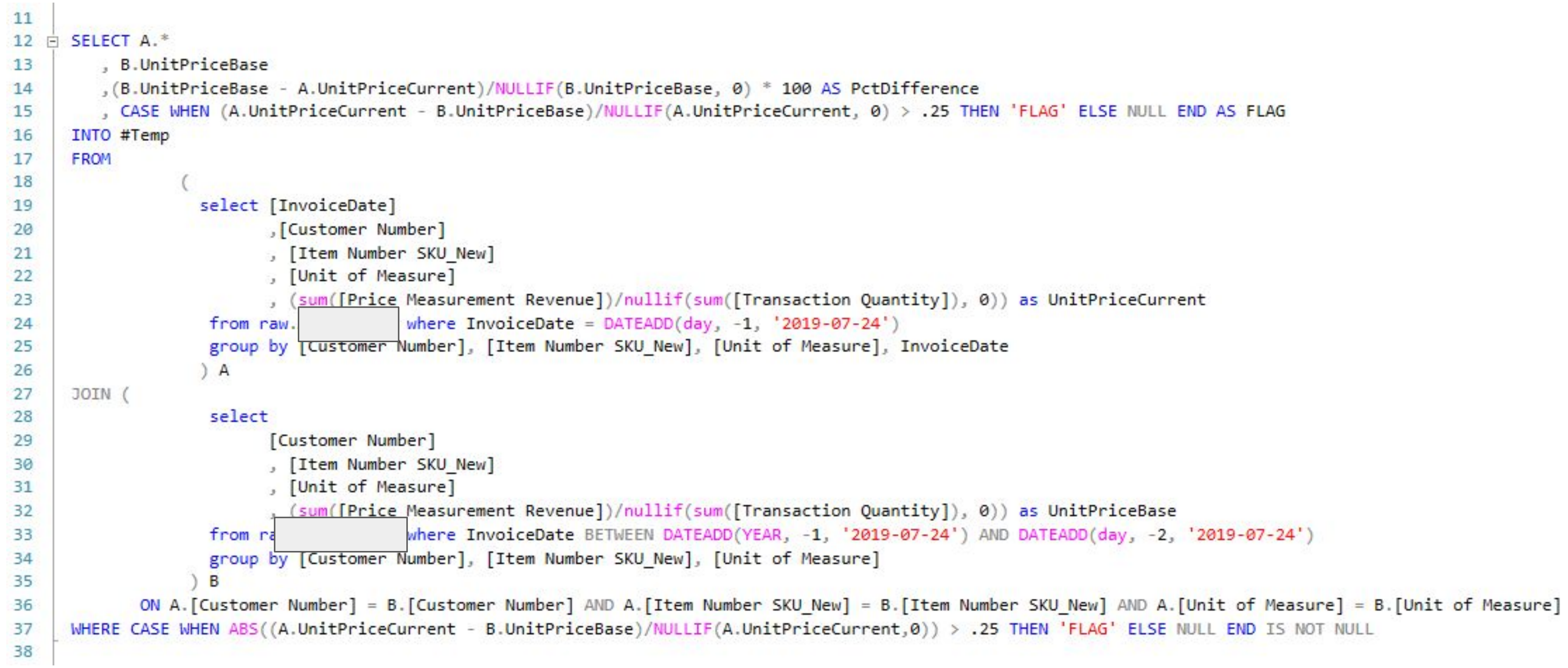
- Goal: Check if the source AR is present
  - If more than 5% is AR – Fail and exit load
  - Else warn and continue



```
ALTER PROCEDURE [etl].[redacted]@DatabaseName VARCHAR(255), @LoadName
AS

DECLARE @RESULT NUMERIC(18,8)
SET @RESULT = NULLIF((
    SELECT sum(nsalesamt)
    FROM etl.[redacted]
    WHERE MappedDate >= DATEADD(WEEK, -1, CURRENT_TIMESTAMP)
    AND SOURCE = 'AR'
    ), 0)

DECLARE @REVENUE NUMERIC(18,8) = NULLIF((
    SELECT sum(nsalesamt)
    FROM etl.[redacted]
    WHERE Map[redacted]ATEADD(WEEK, -1, CURRENT_TIMESTAMP)
    ), 0)

```

Custom Procedures – [redacted]

- Goal: Check that the Unit of Measure recorded in the invoice for a customer/product combination is correct



```

SELECT A.*
     , B.UnitPriceBase
     ,(B.UnitPriceBase - A.UnitPriceCurrent)/NULLIF(B.UnitPriceBase, 0) * 100 AS PctDifference
     , CASE WHEN (A.UnitPriceCurrent - B.UnitPriceBase)/NULLIF(A.UnitPriceCurrent, 0) > .25 THEN 'FLAG' ELSE NULL END AS FLAG
INTO #Temp
FROM
           (
             select [InvoiceDate]
                    ,[Customer Number]
                    , [Item Number SKU_New]
                    , [Unit of Measure]
                    , (sum([Price Measurement Revenue])/nullif(sum([Transaction Quantity]), 0)) as UnitPriceCurrent
              from raw.[redacted] where InvoiceDate = DATEADD(day, -1, '2019-07-24')
              group by [Customer Number], [Item Number SKU_New], [Unit of Measure], InvoiceDate
             ) A
JOIN (
              select
                    [Customer Number]
                    , [Item Number SKU_New]
                    , [Unit of Measure]
                    , (sum([Price Measurement Revenue])/nullif(sum([Transaction Quantity]), 0)) as UnitPriceBase
              from ra[redacted]where InvoiceDate BETWEEN DATEADD(YEAR, -1, '2019-07-24') AND DATEADD(day, -2, '2019-07-24')
              group by [Customer Number], [Item Number SKU_New], [Unit of Measure]
            ) B
       ON A.[Customer Number] = B.[Customer Number] AND A.[Item Number SKU_New] = B.[Item Number SKU_New] AND A.[Unit of Measure] = B.[Unit of Measure]
WHERE CASE WHEN ABS((A.UnitPriceCurrent - B.UnitPriceBase)/NULLIF(A.UnitPriceCurrent,0)) > .25 THEN 'FLAG' ELSE NULL END IS NOT NULL

```