SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [cp\_data].[insert\_load\_status]

(

    @Project\_id int,

    @Batch\_id varchar(50),

    @Source\_Dataset varchar(255),

    @Dataset varchar(255),

    @Source\_layer\_id int,

    @Target\_layer\_id int,

    --@Load\_start\_time datetime,

    @Load\_status varchar(20),

    @load\_type varchar(4),

    @Created\_date datetime,

    @Created\_by varchar(50),

    @Source\_id int,

    @event\_id NVARCHAR(255) = null

)

AS

SET NOCOUNT ON;

DECLARE @load\_status\_id INT = NULL;

IF NOT EXISTS

(

    SELECT project\_id, batch\_id, source\_dataset, source\_layer\_id, target\_layer\_id

    FROM cp\_data.load\_status

    WHERE project\_id = @Project\_id

        AND batch\_id = @Batch\_id

        AND dataset = @Dataset

        AND source\_dataset = @Source\_Dataset

        AND source\_layer\_id = @Source\_layer\_id

        AND target\_layer\_id = @Target\_layer\_id

        AND LOWER(load\_status) = 'new'

        AND source\_id = @Source\_id

)

BEGIN

    INSERT INTO cp\_data.load\_status

    (

        project\_id,

        source\_id,

        batch\_id,

        dataset,

        source\_dataset,

        source\_layer\_id,

        target\_layer\_id,

        --load\_start\_time,

        load\_status,

        rerun\_counter,

        created\_date,

        created\_by

    )

    VALUES

    (

        @Project\_id,

        @Source\_id,

        @Batch\_id,

        @Dataset,

        @Source\_Dataset,

        @Source\_layer\_id,

        @Target\_layer\_id,

        --CONVERT(datetime,@Load\_start\_time) ,

        @Load\_status,

        0,

        CONVERT(datetime,@Created\_date),

        @Created\_by

    )

END

SELECT @load\_status\_id = SCOPE\_IDENTITY() --AS load\_status\_id

IF @load\_status\_id IS NOT NULL AND @load\_status\_id > 0

BEGIN

    -- Skip any existing record in load status table

    IF (@Source\_layer\_id=1 AND @Target\_layer\_id=2)

        OR (@Source\_layer\_id=2 AND @Target\_layer\_id=3 AND LOWER(@load\_type) = 'full')

        OR (@Source\_layer\_id=3 AND @Target\_layer\_id=4 AND LOWER(@load\_type) = 'full')

        OR (@Source\_layer\_id=4 AND @Target\_layer\_id=5)

        OR (@Source\_layer\_id=7 AND @Target\_layer\_id=8)

    BEGIN

        UPDATE cp\_data.load\_status

        SET

            load\_status = 'Skipped',

            status\_description = 'ADF SKipped processing this load because a new one has started',

            modified\_date = @Created\_date,

            modified\_by = @Created\_by

        WHERE

            load\_status\_id != @load\_status\_id

            AND project\_id = @Project\_id

            --AND batch\_id = @Batch\_id

            AND dataset = @Dataset

            AND source\_dataset = @Source\_Dataset

            --AND target\_dataset = @Target\_Dataset

            AND source\_layer\_id = @Source\_layer\_id

            AND target\_layer\_id = @Target\_layer\_id

            AND LOWER(load\_status) = 'new'

            AND source\_id = @Source\_id;

    END

END

SELECT @load\_status\_id AS load\_status\_id

If @event\_id is not null and @event\_id <> ''

BEGIN

    exec [cp\_data].[upsert\_EventIngestionLog] @event\_id,@Batch\_id

END

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

DECLARE @count1 int,

        @rerun int;

SET @count1 = (SELECT max(load\_status\_id)

                FROM cp\_data.load\_status

                WHERE project\_id = @Project\_id

                    --AND batch\_id = @Batch\_id

                    AND dataset = @Dataset

                    AND source\_dataset = @Source\_Dataset

                    --AND target\_dataset = @Target\_Dataset

                    AND source\_layer\_id = @Source\_layer\_id

                    AND target\_layer\_id = @Target\_layer\_id

                    AND source\_id = @Source\_id

                );

SET @rerun = (SELECT rerun\_counter FROM cp\_data.load\_status WHERE load\_status\_id = @count1);

IF EXISTS

(

    SELECT project\_id, batch\_id, source\_dataset, source\_layer\_id, target\_layer\_id

    FROM cp\_data.load\_status

    WHERE project\_id = @Project\_id

        --AND batch\_id = @Batch\_id

        AND dataset = @Dataset

        AND source\_dataset = @Source\_Dataset

        --AND target\_dataset = @Target\_Dataset

        AND source\_layer\_id = @Source\_layer\_id

        AND target\_layer\_id = @Target\_layer\_id

        AND load\_status = 'Failed'

        AND source\_id = @Source\_id

)

BEGIN

    UPDATE cp\_data.load\_status

    SET load\_status = 'Rerun',

        rerun\_counter = @rerun +1

    WHERE load\_status\_id = @count1;

END

ELSE IF NOT EXISTS

(

    SELECT project\_id, batch\_id, source\_dataset, source\_layer\_id, target\_layer\_id

    FROM cp\_data.load\_status

    WHERE project\_id = @Project\_id

        --AND batch\_id = @Batch\_id

        AND dataset = @Dataset

        AND source\_dataset = @Source\_Dataset

        --AND target\_dataset = @Target\_Dataset

        AND source\_layer\_id = @Source\_layer\_id

        AND target\_layer\_id = @Target\_layer\_id

        AND load\_status = 'New'

        AND source\_id = @Source\_id

)

    BEGIN

    INSERT INTO cp\_data.load\_status

    (

        project\_id,

        source\_id,

        batch\_id,

        dataset,

        source\_dataset,

--      target\_dataset,

        source\_layer\_id,

        target\_layer\_id,

        load\_start\_time,

        load\_status,

        rerun\_counter,

        created\_date,

        created\_by

    )

    VALUES

    (

        @Project\_id,

        @Source\_id,

        @Batch\_id,

        @Dataset,

        @Source\_Dataset,

--      @Target\_Dataset,

        @Source\_layer\_id,

        @Target\_layer\_id,

        @Load\_start\_time,

        @Load\_status,

        0,

        @Created\_date,

        @Created\_by

    )

    END

    SELECT SCOPE\_IDENTITY() AS load\_status\_id

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*ELSE IF EXISTS

(

    SELECT project\_id, batch\_id, dataset, source\_layer\_id, target\_layer\_id

    FROM cp\_data.load\_status

    WHERE project\_id = @Project\_id

        --AND batch\_id = @Batch\_id

        AND dataset = @Dataset

        AND source\_layer\_id = @Source\_layer\_id

        AND target\_layer\_id = @Target\_layer\_id

        AND load\_status IN ('New','Success')

    --  AND load\_status\_id = @count1

)

BEGIN

    -- NEED TO STOP STORED PROCEDURE EXECUTION

    RETURN

END

ELSE IF EXISTS

(

    SELECT project\_id, batch\_id, source\_dataset, target\_dataset, source\_layer\_id, target\_layer\_id

    FROM cp\_data.load\_status

    WHERE project\_id = @Project\_id

        --AND batch\_id = @Batch\_id

        AND source\_dataset = @Source\_Dataset

        --AND target\_dataset = @Target\_Dataset

        AND source\_layer\_id = @Source\_layer\_id

        AND target\_layer\_id = @Target\_layer\_id

        AND load\_status = 'Failed'

)

BEGIN

    UPDATE cp\_data.load\_status

    SET load\_status = 'Rerun',

        rerun\_counter = @rerun +1

    WHERE load\_status\_id = @count1;

END

\*/

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [cp\_data].[update\_load\_status]

(

    @Load\_status\_id int,

    @Project\_id int,

    @Batch\_id varchar(50),

    @Source\_Dataset varchar(255),

    @Target\_Dataset varchar(255),

    @Source\_layer\_id int,

    @Target\_layer\_id int,

    @Load\_start\_time datetime,

    @Load\_end\_date datetime,

--  @Processing\_time\_in\_minutes float,

    @Source\_record\_count int = 0,

    @Target\_record\_count int = 0,

    @Databytes\_read float = 0,

    @Load\_status varchar(20),

    @Status\_description varchar(max),

/\*  @Rerun\_counter int,

    @Created\_date datetime,

    @Created\_by varchar(50)\*/

    @Modified\_date datetime,

    @Modified\_by varchar(50),

    @Source\_id int,

    @metricLogJson varchar(max) = '',

    @databricksUrl varchar(max) = '',

    @databricksExecTime int = 0,

    @event\_id NVARCHAR(255) = null

)

AS

SET NOCOUNT ON;

--DECLARE @count1 int;

--DECLARE @start\_time datetime;

Declare @w\_watermark\_id int;

DECLARE @l\_dataset nvarchar(255);

DECLARE @w\_dataset nvarchar(255);

DECLARE @w\_source\_schema nvarchar(255);

DECLARE @s\_watermark\_status nvarchar(255);

/\*SET @count1 = (SELECT load\_status\_id

                FROM cp\_data.load\_status

                WHERE project\_id = @Project\_id

                    --AND batch\_id = @Batch\_id

                    AND source\_dataset = @Source\_Dataset

                    --AND target\_dataset = @Target\_Dataset

                    AND source\_layer\_id = @Source\_layer\_id

                    AND target\_layer\_id = @Target\_layer\_id

                );

\*/

/\*SET @start\_time = (SELECT load\_start\_time

                    FROM cp\_data.load\_status

                    WHERE load\_status\_id = @Load\_status\_id

                    );

\*/

IF EXISTS

(

    SELECT project\_id, source\_id, batch\_id, source\_dataset, target\_dataset, source\_layer\_id, target\_layer\_id

    FROM cp\_data.load\_status

    WHERE load\_status\_id = @Load\_status\_id

)

    BEGIN

        IF LOWER(@Load\_status) = 'success'

        BEGIN

            UPDATE cp\_data.load\_status

                SET target\_dataset = @Target\_Dataset,

                    load\_end\_date = @Load\_end\_date,

                    load\_start\_time = @Load\_start\_time,

                    processing\_time\_in\_minutes = DATEDIFF(Minute, @Load\_start\_time, @Load\_end\_date),

                    source\_record\_count = @Source\_record\_count,

                    target\_record\_count = @Target\_record\_count,

                    databytes\_read = @Databytes\_read,

                    load\_status = @Load\_status,

                    status\_description = @Status\_description,

                    modified\_date = @Modified\_date,

                    modified\_by = @Modified\_by

                WHERE load\_status\_id = @Load\_status\_id;

            SELECT @l\_dataset = dataset

            FROM cp\_data.load\_status

            WHERE load\_status\_id = @Load\_status\_id;

            SELECT @w\_watermark\_id = watermark\_id, @w\_dataset = dataset, @w\_source\_schema = source\_schema

            FROM cp\_data.watermark

            WHERE batch\_id = @Batch\_id

            AND project\_id = @Project\_id

            AND source\_id = @Source\_id

            AND dataset = @l\_dataset

            IF @Source\_layer\_id = 2 and @Target\_layer\_id = 3

            BEGIN

            SET @s\_watermark\_status = 'RAW'

            END

            ELSE IF @Source\_layer\_id = 3 and @Target\_layer\_id = 4

            BEGIN

            SET @s\_watermark\_status = 'COMPLETE'

            END

            EXEC cp\_data.usp\_upsert\_watermark @Project\_id,  @Source\_id  , @w\_dataset, @w\_source\_schema, @s\_watermark\_status, @Batch\_id

        END

        ELSE IF LOWER(@Load\_status) = 'failed'

        BEGIN

            UPDATE cp\_data.load\_status

                SET target\_dataset = @Target\_Dataset,

                    load\_end\_date = @Load\_end\_date,

                    load\_start\_time = @Load\_start\_time,

                    processing\_time\_in\_minutes = DATEDIFF(Minute, @Load\_start\_time, @Load\_end\_date),

                    source\_record\_count = @Source\_record\_count,

                    target\_record\_count = @Target\_record\_count,

                    --databytes\_read = @Databytes\_read,

                    load\_status = @Load\_status,

                    status\_description = @Status\_description,

                    modified\_date = @Modified\_date,

                    modified\_by = @Modified\_by

                WHERE load\_status\_id = @Load\_status\_id;

        END

        IF @metricLogJson <> ''

        BEGIN

            EXEC cp\_data.sp\_insert\_load\_status\_metric @Load\_status\_id,

                                                        @Project\_id,

                                                        @Batch\_id,

                                                        @Source\_id,

                                                        @l\_dataset,

                                                        @Source\_layer\_id ,

                                                        @Target\_layer\_id,

                                                        @databricksUrl,

                                                        @databricksExecTime,

                                                        @metricLogJson

        END

    END

If @event\_id is not null and @event\_id <> ''

    BEGIN

            exec [cp\_data].[update\_event\_ingestion\_log] @event\_id,

                                                        @Batch\_id,

                                                        @Source\_layer\_id,

                                                        @Target\_layer\_id,

                                                        @Status\_description,

                                                        @Load\_status

    END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROC [cp\_data].[usp\_load\_metadata\_info]

    @project\_alias NVARCHAR(30)

    , @dataset\_list NVARCHAR(MAX) = ''

    , @source\_list NVARCHAR(MAX) = ''

    , @source\_list\_variant NVARCHAR(MAX) = ''

    , @load\_from\_processing\_log int = 0

    , @processing\_log\_source\_layer\_id int = 0

    , @processing\_log\_target\_layer\_id int = 0

    , @dataset\_seq\_no NVARCHAR(10) = ''

AS

BEGIN

    BEGIN TRY

         --DECLARE @project\_alias NVARCHAR(30);

         --DECLARE @dataset\_list NVARCHAR(MAX) = '';

         --DECLARE @source\_list NVARCHAR(MAX) = 'ENDUR\_SLMT';

         --DECLARE @source\_list\_variant NVARCHAR(MAX) = ''

         --DECLARE @load\_from\_processing\_log int = 0;

         --DECLARE @processing\_log\_source\_layer\_id int = 0;

         --DECLARE @processing\_log\_target\_layer\_id int = 0;

         --DECLARE @dataset\_seq\_no NVARCHAR(10) = '';

         --SET @project\_alias = 'cds'

         --SET @processing\_log\_source\_layer\_id = 1;

         --SET @processing\_log\_target\_layer\_id = 2;

         --SET @load\_from\_processing\_log = 0;

         --SET @dataset\_seq\_no = '1'

--

        DECLARE @source\_list\_initial NVARCHAR(MAX);

        DECLARE @sql\_transpose\_config NVARCHAR(MAX);

        DECLARE @sql\_transpose\_case\_config NVARCHAR(MAX)  = '';

        DECLARE @sql\_transpose\_config\_dataset NVARCHAR(MAX);

        DECLARE @sql\_transpose\_case\_config\_dataset NVARCHAR(MAX)  = '';

        DECLARE @sql\_transpose\_config\_full NVARCHAR(MAX);

        DECLARE @status\_log\_query NVARCHAR(MAX);

        DECLARE @input\_dataset NVARCHAR(MAX) = '';

        DECLARE @sql\_transpose\_watermark NVARCHAR(MAX);

        DECLARE @VariantFilterCondition NVARCHAR(MAX);

        DECLARE @SeqFilterCondition NVARCHAR(MAX);

        DECLARE @DefaultCluster NVARCHAR(50)='INTERACTIVE';

        PRINT('Dataset Seq No : ' + @dataset\_seq\_no)

        --PRINT('Dataset list: ' + @dataset\_list)

        --print('Project Alias: ' + @project\_alias)

        IF @source\_list\_variant is NULL

            BEGIN

                SET @source\_list\_variant=''

            END

        SET @source\_list\_initial = REPLACE(REPLACE(@source\_list, ',', ''','''), ' ', '');

        --SET @source\_list\_variant = REPLACE(REPLACE(@source\_list\_variant, ',', ''','''), ' ', '');

        SET @VariantFilterCondition = ' AND ((''' + REPLACE(REPLACE(@source\_list\_initial, ',', ''','''), ' ', '') +'''<>''' + REPLACE(REPLACE(@source\_list\_variant, ',', ''','''), ' ', '')  +''' AND source\_name\_variant in ('''+  REPLACE(REPLACE(@source\_list\_variant, ',', ''','''), ' ', '') +'''))'

        + 'OR (''' + REPLACE(REPLACE(@source\_list\_initial, ',', ''','''), ' ', '') +'''=''' + REPLACE(REPLACE(@source\_list\_variant, ',', ''','''), ' ', '')

        +'''))'

        print @VariantFilterCondition

        Print('source\_list\_initial :' + @source\_list\_initial)

        IF @dataset\_list IS NOT NULL AND TRIM(@dataset\_list) != ''

        BEGIN

            SET @dataset\_list = ' AND ls.dataset IN (''' + REPLACE(REPLACE(@dataset\_list, ',', ''','''), ' ', '') + ''') ';

            SET @input\_dataset = @dataset\_list;

            print(@dataset\_list)

        END

        ELSE

        BEGIN

            SET @dataset\_list = '';

            SET @input\_dataset = '';

        END

        IF @source\_list IS NOT NULL AND TRIM(@source\_list) != ''

        BEGIN

            SET @source\_list = ' WHERE source\_name IN (''' + REPLACE(REPLACE(@source\_list, ',', ''','''), ' ', '') + ''') ';

        END

        ELSE

        BEGIN

            SET @source\_list = '';

        END

        --PRINT(@source\_list);

        /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START - Generate CASE statements to be used in transpose query for config\_dataset \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

        SET @sql\_transpose\_case\_config\_dataset =    Stuff((SELECT ', '+CONCAT(' MAX(CASE WHEN config\_name=''', config\_name, ''' THEN config\_value ELSE '''' END ) AS ', REPLACE(config\_name, '?', ''))

                                    FROM cp\_data.config\_dataset c

                                    JOIN cp\_data.projects p ON p.project\_id= c.project\_id

                                    WHERE LOWER(p.project\_alias) IN (@project\_alias) AND LOWER(c.status) = 'active'

                                    GROUP BY c.config\_name

                                    FOR XML PATH ('')),1,1,'');

        --PRINT(@sql\_transpose\_case\_config\_dataset)

        IF @load\_from\_processing\_log = 0

        BEGIN

            SET @sql\_transpose\_config\_dataset = N' SELECT

                                        cd.\*

                                        --,cc.cluster\_config\_id

                                        ,cc.driver\_node\_type

                                        ,cc.worker\_node\_type

                                        ,cc.worker\_autoscale

                                        ,cc.spark\_version

                                        ,isnull(cc.cluster\_type,'''+@DefaultCluster+''')  cluster\_type

                                        ,cc.pool\_id

                                        , s.source\_id

                                        , s.source\_name

                                        , s.source\_type

                                        , s.source\_onprem\_cloud

                                        , s.source\_data\_category

                                        , s.source\_server

                                        , s.source\_port\_no

                                        , s.source\_connection\_type

                                        , s.source\_database\_name

                                        , s.source\_username

                                        , s.source\_azure\_kv\_secret\_name

                                        , s.source\_endpoint\_base\_url

                                        , s.source\_endpoint\_relative\_url

                                        , s.source\_endpoint\_tenant\_client\_id

                                        , s.source\_name\_variant

                                        , s.source\_environment

                                        , s.source\_environment\_type

                                    FROM (

                                        SELECT c.project\_id AS project\_id\_config\_dataset, '

                                            + @sql\_transpose\_case\_config\_dataset + '

                                        FROM cp\_data.config\_dataset c

                                        JOIN cp\_data.projects p ON p.project\_id= c.project\_id

                                        WHERE

                                            LOWER(p.project\_alias) IN (''' + @project\_alias + ''')

                                            AND LOWER(c.status) = ''active''

                                            AND c.source\_id IN (SELECT source\_id FROM cp\_data.sources ' + @source\_list + ' AND status=''active'' AND LOWER(project\_alias) IN (''' + @project\_alias + ''')'

                                            + @VariantFilterCondition + ')

                                            ' + REPLACE(@dataset\_list, 'ls.', '') + '

                                        GROUP BY c.project\_id, c.dataset

                                    ) AS cd

                                    LEFT JOIN cp\_data.config\_cluster cc on cd.cluster\_config\_id = cc.cluster\_config\_id

                                    LEFT JOIN cp\_data.sources s ON s.source\_alias = cd.source\_alias AND LOWER(s.status)=''active''

                                    ' + @source\_list + '

                                    ';

        END

        ELSE

        BEGIN

            IF @input\_dataset IS NULL OR TRIM(@input\_dataset) = ''

            BEGIN

                SET @dataset\_list = Stuff((select ',' +  CONCAT(ls.source\_dataset, '')

                                            FROM [cp\_data].[load\_status] ls

                                            JOIN cp\_data.projects p ON p.project\_id= ls.project\_id

                                            WHERE

                                                ls.source\_layer\_id=@processing\_log\_source\_layer\_id

                                                AND ls.target\_layer\_id=@processing\_log\_target\_layer\_id

                                                AND LOWER(load\_status) IN ('new', 'rerun')

                                            GROUP BY ls.source\_dataset

                                            FOR XML PATH ('')),1,1,'');

            END

            ELSE

            BEGIN

                SET @dataset\_list = @input\_dataset;

            END

            --print(@dataset\_list);

            IF (@input\_dataset IS NULL OR TRIM(@input\_dataset) = '') AND @dataset\_list IS NOT NULL AND TRIM(@dataset\_list) != ''

            BEGIN

                --SET @dataset\_list = ' AND source\_dataset IN (''' + REPLACE(REPLACE(@dataset\_list, ',', ''','''), ' ', '') + ''') ';

                --SET @dataset\_list = ' AND source\_dataset IN (''' + REPLACE(@dataset\_list, ',', ''',''') + ''') ';

                SET @dataset\_list = ' AND source\_dataset IN (''' + LTRIM(RTRIM(REPLACE(@dataset\_list, ',', ''','''))) + ''') ';

            END

            SET @sql\_transpose\_config\_dataset = N' SELECT

                                        cd.\*

                                        --,cc.cluster\_config\_id

                                        ,cc.driver\_node\_type

                                        ,cc.worker\_node\_type

                                        ,cc.worker\_autoscale

                                        ,cc.spark\_version

                                        ,isnull(cc.cluster\_type,'''+@DefaultCluster+''')  cluster\_type

                                        ,cc.pool\_id

                                        , ls\_outer.source\_dataset AS source\_dataset

                                        , s.source\_id

                                        , s.source\_name

                                        , s.source\_type

                                        , s.source\_onprem\_cloud

                                        , s.source\_data\_category

                                        , s.source\_server

                                        , s.source\_port\_no

                                        , s.source\_connection\_type

                                        , s.source\_database\_name

                                        , s.source\_username

                                        , s.source\_azure\_kv\_secret\_name

                                        , s.source\_endpoint\_base\_url

                                        , s.source\_endpoint\_relative\_url

                                        , s.source\_endpoint\_tenant\_client\_id

                                        , s.source\_name\_variant

                                        , s.source\_environment

                                        , s.source\_environment\_type

                                    FROM (

                                        SELECT c.project\_id AS project\_id\_config\_dataset, '

                                            + @sql\_transpose\_case\_config\_dataset + '

                                            , MAX(ls.load\_status\_id) AS load\_status\_id

                                            , MAX(ls.load\_status) AS load\_status

                                        FROM cp\_data.config\_dataset c

                                        RIGHT JOIN cp\_data.load\_status ls ON ls.project\_id = c.project\_id AND ls.dataset = c.dataset AND LOWER(ls.load\_status)=''new'' AND ls.source\_layer\_id=' + CONVERT(varchar(5), @processing\_log\_source\_layer\_id) + ' AND ls.target\_layer\_id=' + CONVERT(varchar(5), @processing\_log\_target\_layer\_id) + ' AND ls.source\_id IN (SELECT source\_id FROM cp\_data.sources ' + @source\_list + ' AND status=''active'' AND LOWER(project\_alias) IN (''' + @project\_alias + ''')'+ @VariantFilterCondition +')

                                        JOIN cp\_data.projects p ON p.project\_id= c.project\_id

                                        WHERE

                                            LOWER(p.project\_alias) IN (''' + @project\_alias + ''')

                                            AND LOWER(c.status) = ''active''

                                            AND c.source\_id IN (SELECT source\_id FROM cp\_data.sources ' + @source\_list + ' AND status=''active'' AND LOWER(project\_alias) IN (''' + @project\_alias + ''')'

                                            + @VariantFilterCondition +

                                            ')

                                            ' + @dataset\_list + '

                                        GROUP BY c.project\_id, c.dataset

                                    ) AS cd

                                    LEFT JOIN cp\_data.config\_cluster cc on cd.cluster\_config\_id = cc.cluster\_config\_id

                                    LEFT JOIN cp\_data.sources s ON s.source\_alias = cd.source\_alias AND LOWER(s.status)=''active''

                                    LEFT JOIN cp\_data.load\_status ls\_outer ON ls\_outer.load\_status\_id = cd.load\_status\_id

                                    ' + @source\_list + '

                                    ';

                                        --, STRING\_AGG(CONCAT( CAST(cdc.source\_column\_name AS NVARCHAR(MAX)), '' AS '', REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(cdc.source\_column\_name, '' '', ''\_''), ''%'', ''''), ''('', ''''), '')'', ''''), ''?'', ''''), ''-'', '''') ), '', '') AS source\_column\_name

                                        --LEFT JOIN [cp\_data].[config\_dataset\_columns] cdc ON cdc.project\_id=p.

        END

        --print('Query 2: ' + @sql\_transpose\_config\_dataset)

        -- SET @sql\_transpose\_config\_dataset = N'SELECT c.project\_id AS project\_id\_config\_dataset, dataset

        --                          FROM cp\_data.config\_dataset c

        --                          LEFT JOIN cp\_data.projects p ON p.project\_id= c.project\_id

        --                          WHERE

        --                                  LOWER(p.project\_alias) IN (''' + @project\_alias + ''')

        --                              AND LOWER(c.status) = ''active''

        --                      v       ' + @dataset\_list + '';

        -- PRINT(@sql\_transpose\_config\_dataset)

        -- EXEC(@sql\_transpose\_config\_dataset)

        /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END - Generate CASE statements to be used in transpose query for config\_dataset \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

        /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* START - Generate CASE statements to be used in transpose query for config\_dataset \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

        SET @sql\_transpose\_case\_config =    Stuff((SELECT ', '+CONCAT(' MAX(CASE WHEN config\_name=''', config\_name, ''' THEN config\_value ELSE '''' END ) AS ', REPLACE(config\_name, '?', ''))

                                    FROM cp\_data.config c

                                    JOIN cp\_data.projects p ON p.project\_id= c.project\_id

                                    WHERE LOWER(p.project\_alias) IN (@project\_alias) AND LOWER(c.status) = 'active'

                                    GROUP BY c.config\_name

                                    FOR XML PATH ('')),1,1,'');

        --PRINT(@sql\_transpose\_case\_config)

        -- SET @sql\_transpose\_config = N' SELECT c.project\_id, w.last\_load\_value, ' + @sql\_transpose\_case\_config + '

        --  FROM cp\_data.config c

        --  JOIN cp\_data.projects p ON p.project\_id= c.project\_id

        --  LEFT JOIN cp\_data.watermark w ON p.project\_id= w.project\_id AND c.dataset= w.dataset

        --  WHERE

        --          LOWER(p.project\_alias) IN (''' + @project\_alias + ''') AND LOWER(c.status) = ''active''

        --  GROUP BY c.project\_id';

        SET @sql\_transpose\_config = N' SELECT c.project\_id, ' + @sql\_transpose\_case\_config + '

            FROM cp\_data.config c

            JOIN cp\_data.projects p ON p.project\_id= c.project\_id

            WHERE

                LOWER(p.project\_alias) IN (''' + @project\_alias + ''') AND LOWER(c.status) = ''active''

            GROUP BY c.project\_id';

        --EXEC(@sql\_transpose\_config)

        /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END - Generate CASE statements to be used in transpose query for config\_dataset \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

        SET @sql\_transpose\_watermark = 'SELECT a.project\_id ,a.source\_id ,a.source\_schema ,a.dataset ,max(a.last\_load\_value) as last\_load\_value

                                        FROM (SELECT \* , dense\_rank() over (partition by project\_id ,source\_id ,source\_schema ,dataset  order by watermark\_id desc) as rn

                                        FROM cp\_data.watermark

                                        WHERE watermark\_status = ''COMPLETE''

                                        ) a

                                        WHERE a.rn = 1

                                        GROUP BY a.project\_id ,a.source\_id ,a.source\_schema ,a.dataset

                                       ';

        --PRINT @sql\_transpose\_watermark

        IF @dataset\_seq\_no is null or @dataset\_seq\_no = ''

            BEGIN

                SET @SeqFilterCondition = ' ORDER BY full\_conf.config\_src\_load\_sequence\_no '

            END

        ELSE

            BEGIN

                SET @SeqFilterCondition = ' Where full\_conf.config\_src\_load\_sequence\_no = ' + @dataset\_seq\_no + ' ORDER BY full\_conf.config\_src\_load\_sequence\_no, full\_conf.dataset '

            END

        PRINT('Seq Filter Condition : ' + @SeqFilterCondition)

        SET @sql\_transpose\_config\_full = 'SELECT full\_conf.\*, w.last\_load\_value FROM

                                            (SELECT \* FROM ('

                                                    + @sql\_transpose\_config +

                                            ') AS conf

                                            JOIN (' + @sql\_transpose\_config\_dataset + ') AS conf\_dataset

                                            ON conf.project\_id = conf\_dataset.project\_id\_config\_dataset

                                            ) AS full\_conf

                                            LEFT JOIN ('+ @sql\_transpose\_watermark +'   ) w

                                            ON full\_conf.project\_id = w.project\_id

                                            AND full\_conf.source\_id = w.source\_id

                                            AND full\_conf.dataset = w.dataset

                                            AND full\_conf.config\_source\_schema = w.source\_schema'

                                            + @SeqFilterCondition ;

        PRINT CAST('Query Final : ' + @sql\_transpose\_config\_full AS NTEXT)

        EXEC(@sql\_transpose\_config\_full);

    END TRY

    BEGIN CATCH

        Declare @ErrorMessage nvarchar(4000);

        Declare @ErrorSeverity int;

        SELECT @ErrorMessage = error\_message(), @ErrorSeverity = error\_severity();

        RAISERROR(@ErrorMessage, @ErrorSeverity, 1);

    END CATCH

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- DROP PROCEDURE [cp\_data].[usp\_upsert\_metadata\_info]

CREATE PROC [cp\_data].[usp\_upsert\_metadata\_info]

    @config\_values NVARCHAR(MAX)

    , @project\_alias NVARCHAR(20) = ''

    , @source\_alias\_list NVARCHAR(MAX) = ''

    , @config\_template\_name NVARCHAR(50)

    , @config\_database\_schema NVARCHAR(20) = NULL  AS

BEGIN

BEGIN TRY

    -- DECLARE @config\_values NVARCHAR(MAX);

    -- DECLARE @config\_template\_name NVARCHAR(50);

    -- DECLARE @config\_database\_schema NVARCHAR(20) = NULL

    --

    -- SET @config\_template\_name = 'projects';

    -- SET @config\_values = N'[

    --     {

    --         "project\_id": "1",

    --         "project\_name": "Counter Party Data",

    --         "project\_description": "Counter Party Data",

    --         "project\_alias": "cp\_data",

    --         "modified\_by": "Emmanuel.Edegbo",

    --         "status": "Active",

    --         "config\_environment": "UAT",

    --         "config\_adls\_store": "https://shell01eunadls1lserrccrn.azuredatalakestore.net/webhdfs/v1",

    --         "config\_az\_tenant\_id": "db1e96a8-a3da-442a-930b-235cac24cd5c",

    --         "config\_az\_subscription\_id": "ecb076fa-3003-44cb-bbc5-9b417e7e25da",

    --         "config\_az\_resource\_group": "shell-01-rg-plqgsnrehdmdeajbnkjd",

    --         "config\_adls\_landing\_path": "W00007-TS-DEV/LAND/1ST\_PARTY/Gold\_Tier\_MI/##config\_environment##",

    --         "config\_adls\_raw\_path": "P00004-TS-DEV/RAW/SENS/1ST\_PARTY/Gold\_Tier\_MI/##config\_environment##",

    --         "config\_adls\_delta\_path": "/RAW/W00007-TS-DEV/DELTA/NON\_SENSITIVE/1stParty/GoldTier\_MI",

    --         "config\_sql\_db\_server": "shell-01-eun-sq-dyogurnmurnlyhuuodlt.database.windows.net",

    --         "config\_sql\_db\_port": "3090",

    --         "config\_sql\_db\_username": null,

    --         "config\_sql\_db\_azure\_kv\_secret\_name": null,

    --         "config\_sql\_dwh\_server": null,

    --         "config\_sql\_dwh\_port": null,

    --         "config\_sql\_dwh\_username": null,

    --         "config\_sql\_dwh\_azure\_kv\_secret\_name": null,

    --         "config\_enable\_databricks\_delta": "Y",

    --         "config\_adls\_landing\_file\_format": "avro",

    --         "config\_adls\_raw\_file\_format": "parquet",

    --         "config\_alds\_delta\_file\_format": "delta",

    --         "config\_skip\_adls\_landing": "N",

    --         "config\_skip\_adls\_raw": "N",

    --         "config\_skip\_adls\_delta": "N",

    --         "config\_skip\_dwh": "N"

    --     },

    --     {

    --         "project\_id": null,

    --         "project\_name": null,

    --         "project\_description": null,

    --         "project\_alias": null,

    --         "modified\_by": null,

    --         "status": null,

    --         "config\_environment": null,

    --         "config\_adls\_store": null,

    --         "config\_az\_tenant\_id": null,

    --         "config\_az\_subscription\_id": null,

    --         "config\_az\_resource\_group": null,

    --         "config\_adls\_landing\_path": null,

    --         "config\_adls\_raw\_path": null,

    --         "config\_adls\_delta\_path": null,

    --         "config\_sql\_db\_server": null,

    --         "config\_sql\_db\_port": null,

    --         "config\_sql\_db\_username": null,

    --         "config\_sql\_db\_azure\_kv\_secret\_name": null,

    --         "config\_sql\_dwh\_server": null,

    --         "config\_sql\_dwh\_port": null,

    --         "config\_sql\_dwh\_username": null,

    --         "config\_sql\_dwh\_azure\_kv\_secret\_name": null,

    --         "config\_enable\_databricks\_delta": null,

    --         "config\_adls\_landing\_file\_format": null,

    --         "config\_adls\_raw\_file\_format": null,

    --         "config\_alds\_delta\_file\_format": null,

    --         "config\_skip\_adls\_landing": null,

    --         "config\_skip\_adls\_raw": null,

    --         "config\_skip\_adls\_delta": null,

    --         "config\_skip\_dwh": null

    --     }

    -- ]'

    -- Parameters used in stored procedure only

    DECLARE @config\_table\_name NVARCHAR(50) = @config\_template\_name;

    DECLARE @unique\_columns NVARCHAR(MAX) = '';

    DECLARE @non\_config\_columns NVARCHAR(MAX);

    DECLARE @non\_config\_columns\_cleaned NVARCHAR(MAX) = '';

    DECLARE @column\_update\_list NVARCHAR(MAX) = '';

    DECLARE @column\_update\_exclusion\_list NVARCHAR(MAX) = '';

    DECLARE @insert\_where\_clause NVARCHAR(MAX) = '';

    DECLARE @merge\_insert\_list NVARCHAR(MAX) = '';

    DECLARE @merge\_insert\_exclusion\_list NVARCHAR(MAX) = '';

    DECLARE @column\_update\_exclusion\_list\_new\_records NVARCHAR(MAX) = '';

    DECLARE @upsert\_query NVARCHAR(MAX);

    DECLARE @current\_timestamp nvarchar(20);

    DECLARE @sql\_transpose NVARCHAR(MAX) = '';

    DECLARE @sql\_transpose\_case NVARCHAR(MAX) = '';

    DECLARE @columns\_transposed NVARCHAR(MAX);

    DECLARE @merge\_join\_clause NVARCHAR(MAX) = '';

    IF @config\_database\_schema IS NULL

    BEGIN

        SET @config\_database\_schema = 'cp\_data';

    END

    SELECT @current\_timestamp = REPLACE(convert(VARCHAR, GETDATE(), 120), ' ', 'T');

    print(@current\_timestamp)

    -- DROP temp table if it exists

    IF OBJECT\_ID('#metadata\_config') IS NOT NULL DROP TABLE #metadata\_config

    --GO

    -- Save metadata information into temp table

    SELECT \* INTO #metadata\_config FROM

    (

        SELECT

            TRIM(config\_json.[key]) AS [config\_position]

            --, config\_json.[value].[project\_id] AS [config\_position\_value]

            , REPLACE(TRIM(config\_json\_value.[key]), 'updated\_by', 'modified\_by') AS [config\_name]

            , TRIM(config\_json\_value.[value]) AS [config\_value]

        FROM OPENJSON(@config\_values) AS config\_json

        CROSS APPLY OPENJSON ( config\_json.value) AS config\_json\_value

        WHERE TRIM(config\_json.[key]) IS NOT NULL OR TRIM(config\_json.[key]) != ''

    ) AS X

    --GO

    SELECT \* FROM #metadata\_config;

    -- Generate column names for the table from information schema

    SET @non\_config\_columns =

        --Stuff((SELECT ','+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

        Stuff((SELECT ','+quotename(Column\_Name ) + ''

    FROM INFORMATION\_SCHEMA.COLUMNS

    WHERE

        TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

        AND Column\_Name NOT IN ('source\_id', 'config\_id', 'config\_dataset\_column\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date','ValidTo','ValidFrom')

        ORDER BY ORDINAL\_POSITION

    FOR XML PATH ('')),1,1,'')

    -- Remove square brackets from the table names

    SET @non\_config\_columns\_cleaned = REPLACE(REPLACE(@non\_config\_columns, '[', ''), ']', '')

    PRINT('Column names: ' + @non\_config\_columns);

    PRINT('Column names cleaned: ' + @non\_config\_columns\_cleaned);

    -- Generate columns update list for table from information schema

    SET @column\_update\_list =

        Stuff((SELECT ', '+quotename(REPLACE(TRIM(Column\_Name), 'updated\_by', 'modified\_by') ) + '=new\_records.'+quotename(REPLACE(TRIM(Column\_Name), 'updated\_by', 'modified\_by') ) + ''

        --Stuff((SELECT ',new\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + '='+@config\_database\_schema+'.' + @config\_table\_name + '.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

    FROM INFORMATION\_SCHEMA.COLUMNS

    WHERE

        TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

        AND Column\_Name NOT IN ('project\_id', 'source\_id', 'config\_id', 'config\_dataset\_column\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date','ValidTo','ValidFrom')

        ORDER BY ORDINAL\_POSITION

    FOR XML PATH ('')),1,1,'')

    -- Remove square brackets from the table names

    SET @column\_update\_list = REPLACE(REPLACE(@column\_update\_list, '[', ''), ']', '')

    PRINT('Columns update list: ' + @column\_update\_list);

    -- Generate columns update list for table from information schema

    SET @column\_update\_exclusion\_list =

        Stuff((SELECT ' , TRIM(existing\_records.'+quotename(REPLACE(TRIM(Column\_Name), 'updated\_by', 'modified\_by') ) + ') '

        --Stuff((SELECT ' OR existing\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + '!=new\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

        --Stuff((SELECT ',new\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + '='+@config\_database\_schema+'.' + @config\_table\_name + '.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

    FROM INFORMATION\_SCHEMA.COLUMNS

    WHERE

        TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

        AND Column\_Name NOT IN ('project\_id', 'source\_id', 'config\_id', 'config\_dataset\_column\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date', 'modified\_by','ValidTo','ValidFrom')

        ORDER BY ORDINAL\_POSITION

    FOR XML PATH ('')),1,1,'')

    -- Remove square brackets from the table names

    IF LEN(@column\_update\_exclusion\_list) > 0

    BEGIN

        print(LEN(@column\_update\_exclusion\_list))

        SET @column\_update\_exclusion\_list = REPLACE(REPLACE(@column\_update\_exclusion\_list, '[', ''), ']', '')

        SET @column\_update\_exclusion\_list = RIGHT(@column\_update\_exclusion\_list, LEN(@column\_update\_exclusion\_list)-1)

        SET @column\_update\_exclusion\_list\_new\_records = REPLACE(@column\_update\_exclusion\_list, 'existing\_records', 'new\_records')

    END

    PRINT('Columns update exclusion list: ' + @column\_update\_exclusion\_list);

    -- Generate columns update list for table from information schema

    SET @merge\_insert\_list =

        Stuff((SELECT ', new\_records.'+quotename(REPLACE(TRIM(Column\_Name), 'updated\_by', 'modified\_by') ) + ''

        --Stuff((SELECT ',new\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + '='+@config\_database\_schema+'.' + @config\_table\_name + '.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

    FROM INFORMATION\_SCHEMA.COLUMNS

    WHERE

        TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

        AND Column\_Name NOT IN ('source\_id', 'config\_id', 'config\_dataset\_column\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date','ValidTo','ValidFrom')

        ORDER BY ORDINAL\_POSITION

    FOR XML PATH ('')),1,1,'')

    -- Remove square brackets from the table names

    SET @merge\_insert\_list = REPLACE(REPLACE(@merge\_insert\_list, '[', ''), ']', '')

    PRINT('MERGE INSERT list: ' + @merge\_insert\_list);

    -- Generate case statement for transpose query

    IF @config\_template\_name NOT IN ('config\_dataset', 'config\_dataset\_columns')

        BEGIN

            SET @sql\_transpose\_case =

                Stuff((SELECT ','+CONCAT(' MAX(CASE WHEN TRIM(config\_name)=''', TRIM(Column\_Name), ''' THEN TRIM(config\_value) ELSE '''' END ) AS ', TRIM(Column\_Name))

            FROM INFORMATION\_SCHEMA.COLUMNS

            WHERE

                TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

                AND Column\_Name NOT IN ('source\_id', 'config\_id', 'config\_dataset\_column\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date','ValidTo','ValidFrom')

                ORDER BY ORDINAL\_POSITION

            FOR XML PATH ('')),1,1,'')

        END

    ELSE

        BEGIN

            SET @sql\_transpose\_case =

                Stuff((SELECT ','+CONCAT(' MAX(CASE WHEN TRIM(config\_name)=''', TRIM(Column\_Name), ''' THEN TRIM(config\_value) ELSE '''' END ) AS ', TRIM(Column\_Name))

            FROM INFORMATION\_SCHEMA.COLUMNS

            WHERE

                TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

                AND Column\_Name NOT IN ('source\_id', 'project\_id', 'config\_id', 'config\_dataset\_column\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date','ValidTo','ValidFrom')

                ORDER BY ORDINAL\_POSITION

            FOR XML PATH ('')),1,1,'')

        END

    PRINT('Transcpose CASE statements: ' + @sql\_transpose\_case)

    DECLARE @update\_columns NVARCHAR(MAX) = '';

    -- Define unique column identifiers

    IF @config\_table\_name = 'projects'

    BEGIN

        SET @merge\_insert\_exclusion\_list = ' WHERE project\_id IS NOT NULL AND project\_id != '''' ';

        SET @unique\_columns = ' TRIM(project\_alias), TRIM(status) ';

        SET @insert\_where\_clause = ' WHERE TRIM(project\_alias) NOT IN (SELECT DISTINCT TRIM(project\_alias) FROM ' + @config\_database\_schema + '.' + @config\_table\_name + ') '

        SET @update\_columns = ' TRIM(existing\_records.project\_alias)=TRIM(new\_records.project\_alias) AND existing\_records.project\_id=new\_records.project\_id ';

    END

    IF @config\_table\_name = 'sources'

    BEGIN

        SET @merge\_insert\_exclusion\_list = ' WHERE source\_alias IS NOT NULL AND TRIM(source\_alias) != '''' ';

        -- SET @unique\_columns = 'source\_alias, source\_type, source\_type\_name, source\_onprem\_cloud, source\_internal\_external, source\_server, source\_database\_name, source\_schema, status';

        SET @unique\_columns = ' TRIM(source\_alias), TRIM(source\_type), TRIM(source\_type\_name), TRIM(source\_onprem\_cloud), TRIM(source\_internal\_external), TRIM(source\_server), TRIM(source\_database\_name), TRIM(source\_schema), TRIM(status) ';

        SET @insert\_where\_clause = ' WHERE CONCAT(TRIM(source\_alias), TRIM(source\_type), TRIM(source\_type\_name)) NOT IN (SELECT DISTINCT CONCAT(source\_alias, source\_type, source\_type\_name) FROM ' + @config\_database\_schema + '.' + @config\_table\_name + ')'

        SET @update\_columns = ' TRIM(existing\_records.source\_alias)=TRIM(new\_records.source\_alias) ';

    END

    IF @config\_table\_name = 'config\_descriptions'

    BEGIN

        SET @unique\_columns = 'TRIM(config\_name), TRIM(status)';

        SET @update\_columns = ' TRIM(existing\_records.config\_name)=TRIM(new\_records.config\_name) ';

    END

    PRINT('Unique columns in table ''' + @config\_table\_name + ''': ' + @unique\_columns)

    IF @config\_template\_name NOT IN ('config\_dataset', 'config\_dataset\_columns')

    BEGIN

        IF OBJECT\_ID('#metadata\_nonconfig\_transposed') IS NOT NULL DROP TABLE #metadata\_nonconfig\_transposed

        SET @sql\_transpose = N' SELECT transposed\_data.\* FROM ( SELECT ' + @sql\_transpose\_case + '

        FROM #metadata\_config

        WHERE

                config\_name IN (SELECT TRIM(value) FROM STRING\_SPLIT(''' + @non\_config\_columns\_cleaned + ''', '',''))

        GROUP BY config\_position ) transposed\_data ' + @merge\_insert\_exclusion\_list;

        PRINT('SQL transpose: ' + @sql\_transpose);

        --EXEC(@sql\_transpose)

        SET @upsert\_query = 'MERGE ' + @config\_database\_schema + '.' + @config\_table\_name + ' AS [existing\_records]

            USING (SELECT ' + @non\_config\_columns\_cleaned + ' FROM #metadata\_nonconfig\_transposed) AS [new\_records]

            ON ' + @update\_columns + '

            WHEN MATCHED AND EXISTS(SELECT ' + @column\_update\_exclusion\_list + ' EXCEPT SELECT ' + @column\_update\_exclusion\_list\_new\_records + ') THEN

              UPDATE SET ' + @column\_update\_list + ', modified\_date=''' + @current\_timestamp + '''

            WHEN NOT MATCHED THEN

              INSERT (' + @non\_config\_columns\_cleaned + ', created\_by, created\_date, modified\_date) VALUES (' + @merge\_insert\_list + ', new\_records.modified\_by, ''' + @current\_timestamp + ''', ''' + @current\_timestamp + ''')'

        print('UPSERT query: ' + @upsert\_query)

        EXEC('

            SELECT \* INTO #metadata\_nonconfig\_transposed FROM (' + @sql\_transpose + ') AS transposed\_data;'

            + 'SELECT \* FROM #metadata\_nonconfig\_transposed;'

            + @upsert\_query + ';'

        );

        -- DROP temp table '#metadata\_config\_transposed' if it exists

        IF OBJECT\_ID('#metadata\_nonconfig\_transposed') IS NOT NULL DROP TABLE #metadata\_nonconfig\_transposed

    END

    -- Extract config related columns from 'projects', 'config\_dataset\_columns' and 'config\_dataset' worksheets

    IF @config\_template\_name IN ('projects', 'config\_dataset', 'config\_dataset\_columns')

    BEGIN

        DECLARE @config\_sql\_query NVARCHAR(MAX) = '';

        PRINT(@config\_template\_name)

        IF @config\_template\_name = 'projects'

        BEGIN

            SET @config\_table\_name = 'config'

        END

        PRINT('New table name: ' + @config\_table\_name)

        -- Generate column names for the table from information schema

        SET @non\_config\_columns =

            --Stuff((SELECT ','+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

            Stuff((SELECT ','+quotename(TRIM(Column\_Name) ) + ''

        FROM INFORMATION\_SCHEMA.COLUMNS

        WHERE

            TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

            AND Column\_Name NOT IN ('config\_id', 'config\_dataset\_column\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date','ValidTo','ValidFrom')

            ORDER BY ORDINAL\_POSITION

        FOR XML PATH ('')),1,1,'')

        -- Remove square brackets from the table names

        SET @non\_config\_columns\_cleaned = REPLACE(REPLACE(@non\_config\_columns, '[', ''), ']', '')

        PRINT('Column names2: ' + @non\_config\_columns);

        PRINT('Column names cleaned2: ' + @non\_config\_columns\_cleaned);

        -- Generate columns update list for table from information schema

        SET @column\_update\_list =

            Stuff((SELECT ', '+quotename(REPLACE(TRIM(Column\_Name), 'updated\_by', 'modified\_by') ) + '=new\_records.'+quotename(REPLACE(TRIM(Column\_Name), 'updated\_by', 'modified\_by') ) + ''

            --Stuff((SELECT ',new\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + '='+@config\_database\_schema+'.' + @config\_table\_name + '.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

        FROM INFORMATION\_SCHEMA.COLUMNS

        WHERE

            TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

            AND Column\_Name NOT IN ('config\_id', 'config\_dataset\_column\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date','ValidTo','ValidFrom')

            ORDER BY ORDINAL\_POSITION

        FOR XML PATH ('')),1,1,'')

        -- Remove square brackets from the table names

        SET @column\_update\_list = REPLACE(REPLACE(@column\_update\_list, '[', ''), ']', '')

        PRINT('Columns update list2: ' + @column\_update\_list);

        -- Generate columns update list for table from information schema

        SET @column\_update\_exclusion\_list =

            --Stuff((SELECT ' ,  existing\_records.'+quotename(REPLACE(TRIM(Column\_Name), 'updated\_by', 'modified\_by') ) + ' COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS '

            Stuff((SELECT ' ,  existing\_records.'+quotename(REPLACE(TRIM(Column\_Name), 'updated\_by', 'modified\_by') ) + ' '

            --Stuff((SELECT ' OR existing\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + '!=new\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

            --Stuff((SELECT ',new\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + '='+@config\_database\_schema+'.' + @config\_table\_name + '.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

        FROM INFORMATION\_SCHEMA.COLUMNS

        WHERE

            TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

            --AND Column\_Name NOT IN ('project\_id', 'source\_id', 'config\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date', 'modified\_by')

            AND Column\_Name NOT IN ('project\_id', 'config\_id', 'config\_dataset\_column\_id', 'dataset', 'config\_name', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date', 'modified\_by','ValidTo','ValidFrom')

            ORDER BY ORDINAL\_POSITION

        FOR XML PATH ('')),1,1,'')

        -- Remove square brackets from the table names

        IF LEN(@column\_update\_exclusion\_list) > 0

        BEGIN

            print(LEN(@column\_update\_exclusion\_list))

            SET @column\_update\_exclusion\_list = REPLACE(REPLACE(@column\_update\_exclusion\_list, '[', ''), ']', '')

            SET @column\_update\_exclusion\_list = RIGHT(@column\_update\_exclusion\_list, LEN(@column\_update\_exclusion\_list)-1)

            SET @column\_update\_exclusion\_list\_new\_records = REPLACE(@column\_update\_exclusion\_list, 'existing\_records', 'new\_records')

        END

        PRINT('Columns update exclusion list: ' + @column\_update\_exclusion\_list);

        -- DROP temp table '#metadata\_config\_transposed' if it exists

        IF OBJECT\_ID('#metadata\_config\_transposed') IS NOT NULL DROP TABLE #metadata\_config\_transposed

        IF @config\_template\_name IN ('config\_dataset', 'config\_dataset\_columns')

        BEGIN

            SET @merge\_join\_clause = ' AND existing\_records.project\_id=(SELECT project\_id FROM ' + @config\_database\_schema + '.projects WHERE project\_alias=''' + @project\_alias + ''') AND existing\_records.source\_id IN (SELECT source\_id FROM ' + @config\_database\_schema + '.sources WHERE source\_alias IN (''' + REPLACE(REPLACE(@source\_alias\_list, ',', ''','''), ' ', '') + ''') ) ';

        END

        IF @config\_table\_name = 'config'

        BEGIN

            SET @unique\_columns = ' project\_id, config\_name, status ';

            SET @update\_columns = ' TRIM(existing\_records.config\_name)=TRIM(new\_records.config\_name)  COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS AND existing\_records.project\_id=new\_records.project\_id  COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS ';

            SET @config\_sql\_query = 'SELECT '

                + @non\_config\_columns\_cleaned + '

            FROM

                #metadata\_config t1

            LEFT JOIN (

                SELECT

                    config\_position,

                    config\_value AS project\_id

                FROM

                    #metadata\_config

                WHERE

                    TRIM(config\_name)= ''project\_id''

            ) t2 ON TRIM(t1.config\_position) = TRIM(t2.config\_position)

            LEFT JOIN (

                SELECT

                    config\_position,

                    TRIM(config\_value) AS modified\_by

                FROM

                    #metadata\_config

                WHERE

                    TRIM(config\_name) = ''modified\_by''

            ) t3 ON TRIM(t1.config\_position) = TRIM(t3.config\_position)

            LEFT JOIN (

                SELECT

                    config\_position,

                    COALESCE(TRIM(config\_value), ''Active'') AS status

                FROM

                    #metadata\_config

                WHERE

                    TRIM(config\_name) = ''status''

            ) t4 ON TRIM(t1.config\_position) = TRIM(t4.config\_position)

            WHERE

                TRIM(t1.config\_name) LIKE ''config\_%'' ' + REPLACE(@merge\_insert\_exclusion\_list, 'WHERE', ' AND ')

            + ' AND project\_id IS NOT NULL  '

            print(@config\_sql\_query)

        END

        IF @config\_table\_name IN ('config\_dataset')

        BEGIN

            SET @unique\_columns = ' project\_id, dataset, config\_name, status';

            SET @update\_columns = ' TRIM(existing\_records.config\_name)=TRIM(new\_records.config\_name)  COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS AND existing\_records.project\_id=new\_records.project\_id AND existing\_records.source\_id=new\_records.source\_id AND TRIM(existing\_records.dataset)=TRIM(new\_records.dataset) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS ';

            -- project\_id,dataset,config\_name,config\_value,modified\_by

            SET @config\_sql\_query = 'SELECT '

                + @non\_config\_columns\_cleaned +

            ' FROM

                #metadata\_config t1

            LEFT JOIN (

                SELECT

                    mc.config\_position,

                    TRIM(mc.config\_value) AS project\_alias,

                    p.project\_id

                FROM

                    #metadata\_config mc

                JOIN cp\_data.projects p ON TRIM(p.project\_alias) = TRIM(mc.config\_value)

                WHERE

                    TRIM(config\_name)= ''project\_alias''

            ) t2 ON t1.config\_position = t2.config\_position

            LEFT JOIN (

                SELECT

                    mc2.config\_position,

                    TRIM(mc2.config\_value) AS project\_alias,

                    s.source\_id

                FROM

                    #metadata\_config mc2

                JOIN cp\_data.sources s ON TRIM(s.source\_alias) = TRIM(mc2.config\_value)

                WHERE

                    TRIM(config\_name)= ''source\_alias''

            ) t6 ON t1.config\_position = t6.config\_position

            LEFT JOIN (

                SELECT

                    config\_position,

                    TRIM(config\_value) AS modified\_by

                FROM

                    #metadata\_config

                WHERE

                    TRIM(config\_name) = ''modified\_by''

            ) t3 ON t1.config\_position = t3.config\_position

            LEFT JOIN (

                SELECT

                    config\_position,

                    TRIM(config\_value) AS dataset

                FROM

                    #metadata\_config

                WHERE

                    TRIM(config\_name) = ''dataset''

            ) t4 ON t1.config\_position = t4.config\_position

            LEFT JOIN (

                SELECT

                    config\_position,

                    COALESCE(TRIM(config\_value), ''Active'') AS status

                FROM

                    #metadata\_config

                WHERE

                    TRIM(config\_name) = ''status''

            ) t5 ON t1.config\_position = t5.config\_position ' + REPLACE(@merge\_insert\_exclusion\_list, 'WHERE', ' AND ')

            + ' WHERE project\_id IS NOT NULL  '

        END

        IF @config\_table\_name IN ('config\_dataset\_columns')

        BEGIN

            SET @unique\_columns = ' project\_id, source\_column\_name, dataset, source\_id, status';

            SET @update\_columns = ' TRIM(existing\_records.source\_column\_name)=TRIM(new\_records.source\_column\_name)  COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS AND existing\_records.project\_id=new\_records.project\_id AND existing\_records.source\_id=new\_records.source\_id AND TRIM(existing\_records.dataset)=TRIM(new\_records.dataset) COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS ';

            SET @sql\_transpose = N' SELECT

                                        transposed\_data.\*

                                        , s.source\_id

                                        , p.project\_id

                                    FROM (

                                        SELECT ' + @sql\_transpose\_case + '

                                            , MAX(CASE WHEN TRIM(config\_name)=''project\_alias'' THEN TRIM(config\_value) ELSE '''' END ) AS project\_alias

                                            , MAX(CASE WHEN TRIM(config\_name)=''source\_alias'' THEN TRIM(config\_value) ELSE '''' END ) AS source\_alias

                                        FROM #metadata\_config t1

                                        GROUP BY t1.config\_position

                                    ) transposed\_data

                                    JOIN ' + @config\_database\_schema + '.projects p ON p.project\_alias = transposed\_data.project\_alias

                                    JOIN ' + @config\_database\_schema + '.sources s ON s.source\_alias = transposed\_data.source\_alias ';

                PRINT('SQL transpose 2: ' + @sql\_transpose);

                --EXEC(@sql\_transpose)

            SET @config\_sql\_query = @sql\_transpose;

        END

        PRINT('Config SQL query ' + @config\_sql\_query)

        --EXEC(@config\_sql\_query)

        CREATE CLUSTERED INDEX cx\_configposition ON #metadata\_config (config\_position);

        -- Generate columns update list for table from information schema

        SET @merge\_insert\_list =

            Stuff((SELECT ', new\_records.'+quotename(REPLACE(TRIM(Column\_Name), 'updated\_by', 'modified\_by') ) + ''

            --Stuff((SELECT ',new\_records.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + '='+@config\_database\_schema+'.' + @config\_table\_name + '.'+quotename(REPLACE(Column\_Name, 'updated\_by', 'modified\_by') ) + ''

        FROM INFORMATION\_SCHEMA.COLUMNS

        WHERE

            TABLE\_NAME=@config\_table\_name AND TABLE\_SCHEMA=@config\_database\_schema

            AND Column\_Name NOT IN ('config\_id', 'config\_dataset\_column\_id', 'config\_description\_id', 'created\_date', 'created\_by', 'modified\_date','ValidTo','ValidFrom')

            ORDER BY ORDINAL\_POSITION

        FOR XML PATH ('')),1,1,'')

        -- Remove square brackets from the table names

        SET @merge\_insert\_list = REPLACE(REPLACE(@merge\_insert\_list, '[', ''), ']', '')

        PRINT('MERGE INSERT list2: ' + @merge\_insert\_list);

        SET @upsert\_query = 'MERGE ' + @config\_database\_schema + '.' + @config\_table\_name + ' AS [existing\_records]

        USING (SELECT ' + @non\_config\_columns\_cleaned + ' FROM #metadata\_config\_transposed) AS [new\_records]

        ON ' + @update\_columns + @merge\_join\_clause + '

        WHEN MATCHED AND EXISTS(SELECT ' + @column\_update\_exclusion\_list + ' EXCEPT SELECT ' + @column\_update\_exclusion\_list\_new\_records + ') THEN

          UPDATE SET ' + @column\_update\_list + ', modified\_date=''' + @current\_timestamp + '''

        WHEN NOT MATCHED THEN

          INSERT (' + @non\_config\_columns\_cleaned + ', created\_by, created\_date, modified\_date) VALUES (' + @merge\_insert\_list + ', new\_records.modified\_by, ''' + @current\_timestamp + ''', ''' + @current\_timestamp + ''');'

        print('UPSERT query: ' + @upsert\_query)

        --EXEC('SELECT \* INTO #metadata\_config\_transposed FROM (' + @config\_sql\_query + ') AS transposed\_data;'

        --);

        EXEC('SELECT \* INTO #metadata\_config\_transposed FROM (' + @config\_sql\_query + ') AS transposed\_data;'

            + 'SELECT \* FROM #metadata\_config\_transposed;'

            + @upsert\_query + ';'

        );

        -- DROP temp table '#metadata\_config\_transposed' if it exists

        IF OBJECT\_ID('#metadata\_config\_transposed') IS NOT NULL DROP TABLE #metadata\_config\_transposed

    END

    -- DROP temp table '#metadata\_config' if it exists

    IF OBJECT\_ID('#metadata\_config') IS NOT NULL DROP TABLE #metadata\_config

END TRY

BEGIN CATCH

    Declare @ErrorMessage nvarchar(4000);

    Declare @ErrorSeverity int;

    SELECT @ErrorMessage = error\_message(), @ErrorSeverity = error\_severity();

    RAISERROR(@ErrorMessage, @ErrorSeverity, 1);

END CATCH

END

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROC [cp\_data].[usp\_upsert\_watermark]

    @project\_id int

    , @source\_id int

    , @dataset NVARCHAR(255)

    , @source\_schema NVARCHAR(255)

    , @watermark\_status NVARCHAR(255)  = NULL

    , @batch\_id NVARCHAR(50)

    , @last\_load\_value NVARCHAR(50)  = NULL AS

BEGIN

BEGIN TRY

    IF @last\_load\_value IS NOT NULL AND TRIM(@last\_load\_value) != ''

    BEGIN

        DECLARE @current\_timestamp nvarchar(20);

        DECLARE @ls\_var1 nvarchar(5);

        DECLARE @w\_var1 nvarchar(5);

        DECLARE @ls\_var2 nvarchar(5);

        DECLARE @w\_var2 nvarchar(5);

        DECLARE @ls\_var3 nvarchar(5);

        DECLARE @w\_var3 nvarchar(5);

        SELECT @current\_timestamp = convert(VARCHAR, GETDATE(), 120);

        print('Current timestamp: ' + @current\_timestamp)

        IF EXISTS

            (   SELECT \*

                FROM cp\_data.load\_status

                WHERE batch\_id = @batch\_id

                AND project\_id = @project\_id

                AND source\_id = @source\_id

                AND dataset = @dataset

                AND load\_status = 'Success'

                AND source\_layer\_id = 1 AND target\_layer\_id = 2

            )

        BEGIN

        SET @ls\_var1 = 'y'

        print('ls\_var1 '+@ls\_var1);

        END

        IF NOT EXISTS

            (   SELECT \*

                FROM cp\_data.watermark

                WHERE batch\_id = @batch\_id

                AND project\_id = @project\_id

                AND source\_id = @source\_id

                AND dataset = @dataset

                AND watermark\_status = 'LANDING'

            )

        BEGIN

        SET @w\_var1 = 'n'

        print('w\_var1 '+@w\_var1);

        END

        IF @ls\_var1 = 'y' and @w\_var1 = 'n'

        BEGIN

            INSERT INTO cp\_data.watermark (project\_id, source\_id, source\_schema, dataset, watermark\_status, last\_load\_value, created\_date, modified\_date, batch\_id)

            VALUES (@project\_id, @source\_id, @source\_schema, @dataset, @watermark\_status, @last\_load\_value, @current\_timestamp, @current\_timestamp, @batch\_id);

            print('Inserted in watermark table for project id: '+ str(@project\_id) + ' ,source id:' + str(@source\_id) + ' ,source schema: '+@source\_schema + ' ,dataset: ' + @dataset +'with status '+@watermark\_status);

        END

    END

    ELSE

        BEGIN

        ------------------- Updating watermark status on completion of raw layer---------------

            IF EXISTS

                (

                    SELECT \*

                    FROM cp\_data.load\_status

                    WHERE batch\_id = @batch\_id

                    AND project\_id = @project\_id

                    AND source\_id =  @source\_id

                    AND dataset =    @dataset

                    AND load\_status = 'Success'

                    AND source\_layer\_id = 2 AND target\_layer\_id = 3

                )

            BEGIN

                SET @ls\_var2 = 'y'

                print('ls\_var2 '+@ls\_var2);

            END

            IF NOT EXISTS

                (   SELECT \*

                    FROM cp\_data.watermark

                    WHERE batch\_id = @batch\_id

                    AND project\_id = @project\_id

                    AND source\_id = @source\_id

                    AND dataset = @dataset

                    AND watermark\_status = 'RAW'

                )

            BEGIN

                SET @w\_var2 = 'n'

                print('w\_var2 '+@w\_var2);

            END

            IF @ls\_var2 = 'y' and @w\_var2 = 'n'

                BEGIN

                    SELECT @current\_timestamp = convert(VARCHAR, GETDATE(), 120);

                    print('Current timestamp: ' + @current\_timestamp)

                    UPDATE cp\_data.watermark

                    SET watermark\_status = @watermark\_status, modified\_date = @current\_timestamp

                    WHERE project\_id = @project\_id

                    AND source\_id = @source\_id

                    AND dataset = @dataset

                    AND source\_schema = @source\_schema

                    AND batch\_id = @batch\_id

                    print('Updated in watermark table for project id: '+ str(@project\_id) + ' ,source id:' + str(@source\_id) + ' ,source schema: '+@source\_schema + ' ,dataset: ' + @dataset +'with status '+@watermark\_status);

                END

            ------------------- Updating watermark status on completion of unharm layer---------------

            IF EXISTS

                (

                    SELECT \*

                    FROM cp\_data.load\_status

                    WHERE batch\_id = @batch\_id

                    AND project\_id = @project\_id

                    AND source\_id = @source\_id

                    AND dataset = @dataset

                    AND load\_status = 'Success'

                    AND source\_layer\_id = 3 AND target\_layer\_id = 4

                )

                BEGIN

                    SET @ls\_var3 = 'y'

                    print('ls\_var3 '+@ls\_var3);

                END

            IF NOT EXISTS

                (   SELECT \*

                    FROM cp\_data.watermark

                    WHERE batch\_id = @batch\_id

                    AND project\_id = @project\_id

                    AND source\_id = @source\_id

                    AND dataset = @dataset

                    AND watermark\_status = 'COMPLETE'

                )

            BEGIN

                SET @w\_var3 = 'n'

                print('w\_var3 '+@w\_var3);

            END

        IF @ls\_var3 = 'y' and @w\_var3 = 'n'

            BEGIN

                SELECT @current\_timestamp = convert(VARCHAR, GETDATE(), 120);

                print('Current timestamp: ' + @current\_timestamp)

                UPDATE cp\_data.watermark

                SET watermark\_status = @watermark\_status, modified\_date = @current\_timestamp

                WHERE project\_id = @project\_id

                AND source\_id = @source\_id

                AND dataset = @dataset

                AND source\_schema = @source\_schema

                AND batch\_id = @batch\_id

                print('Updated in watermark table for project id: '+ str(@project\_id) + ' ,source id:' + str(@source\_id) + ' ,source schema: '+@source\_schema + ' ,dataset: ' + @dataset +'with status '+@watermark\_status) ;

            END

        END

END TRY

BEGIN CATCH

    Declare @ErrorMessage nvarchar(4000);

    Declare @ErrorSeverity int;

    SELECT @ErrorMessage = error\_message(), @ErrorSeverity = error\_severity();

    RAISERROR(@ErrorMessage, @ErrorSeverity, 1);

END CATCH

END

GO