Rebuilding the BIClass DB

GROUP#1: Jaskaran, David, Ashly, Maitri, Richard and William

What Are NACE Competencies?

- NACE 1: Career and self development
- NACE 2: Leadership
- NACE 3: Communication
- NACE 4: Professionalism
- NACE 5: Critical Thinking
- NACE 6: Teamwork
- NACE 7: Equity & Inclusion
- NACE 8: Technology

QC CS Department NACE Competencies - TechSmith Screencast

How NACE Competencies helped us in this project?

- The NACE competencies played a crucial role in the successful execution of our project. Critical thinking and problem-solving were key as we designed the schema, developed stored procedures, and implemented logic for surrogate key generation using SQL Server sequences. Teamwork and collaboration were evident throughout the project, as we divided responsibilities, coordinated tasks like schema setup and data loading, and worked closely to maintain consistent documentation. Our communication skills helped us clearly articulate technical decisions in both code and presentations, while professionalism ensured that all deliverables were completed with accuracy and within deadlines. We leveraged our technological skills by utilizing advanced features of SQL Server Management Studio (SSMS) and optimizing the ETL workflows. Additionally, leadership and initiative were demonstrated in designing workflow logging and metadata tracking, and we continuously improved by reflecting on feedback and learning new techniques throughout the project. Overall, these competencies not only enhanced our individual contributions but also strengthened the overall effectiveness of the team.

Overview:

Part 1: Non-technical Project Objective

Part 2: Voice annotation describing the work

Part 1: Non-technical Project Objective

The Effectiveness of Meetings with an Agenda and Attendance

Meetings that include a clear agenda and confirmed attendance are significantly more effective, as they promote structure, focus, and accountability. An agenda ensures that all participants are aware of the topics to be discussed and can come prepared, which minimizes off-topic conversations and maximizes productive use of time. Knowing who will attend allows for better coordination and ensures that the right stakeholders are present to make decisions or provide necessary input. This combination leads to more efficient discussions, quicker resolutions, and a greater likelihood of achieving meeting

Contributions:

Jaskaren – Primary Project Leader

Ash – Backup Project Leader

Maitri - Secretary/Meeting Documentations

David - Notebook organization/parsed file

Richard – PowerPoint/Video editing

William - Powerpoint/Planning

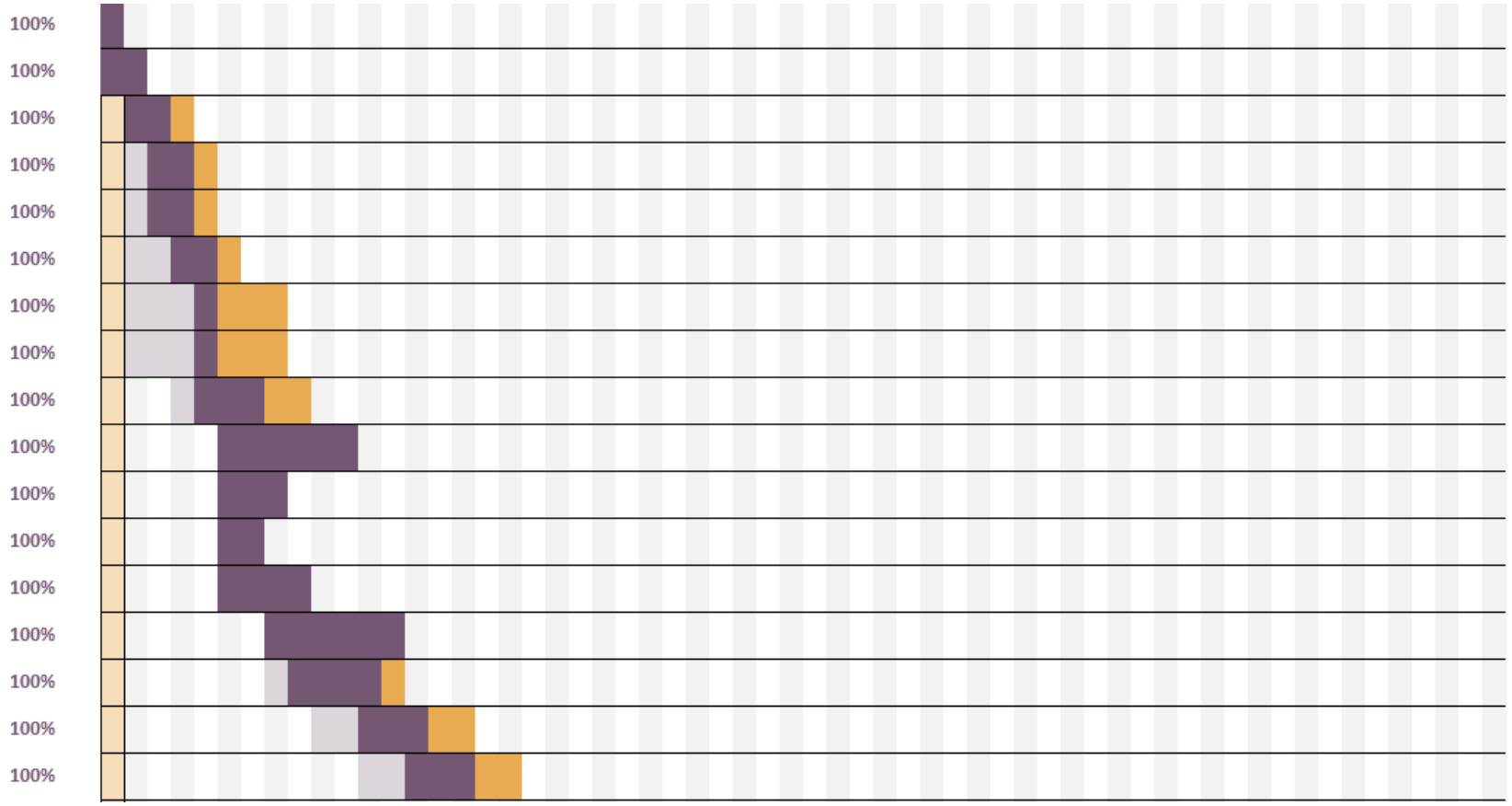
To-do List for Project #2

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Notes
100%	Assigning Work	04/11/25			In a day	Jasky assigned work
100%	Overview of the Project	04/11/25			In a day	Jasky summarized the project and made steps
100%	Created tables: UserAuthorization and WorkflowSteps	04/12/25			In a day	Ashly created the tables
100%	Created procedure: TrackWorkFlow	04/12/25			In a day	Ashly created the procedure
100%	Added 2 new tables: ProductCategory and Subcategory	04/12/25			In a day	Ashly added the 2 new tables
100%	Alter every table to include 3 new columns, the userAuth keys, the date added, and the date of last update	04/12/25			In a day	Ashly did
100%	Drop all pri keys and foreign key relations	04/12/25			In a day	Ashly did

100%	Sequence Objects	04/12/25	In a day	Jasky and David did
100%	Truncate tables	04/12/25	In a day	Ashly did
100%	Create status function	04/13/25	In a day	Jasky and David
100%	Meeting documentations	04/13/25	1 Day	Maitri did
100%	Created a PowerPoint that consists of two parts	04/13/25	In a day	Ashly and David did
100%	First part of the PowerPoint	04/13/25	In a day	William and Maitri did
100%	Create loading functions	04/13/25	In a day	Ashly, David and Maitri did
100%	To do list	04/13/25	In a day	William did
100%	Gantt Chart	04/13/25	In a day	William did
100%	Parsing Objects	04/13/25	In a day	David, Maitri and William
100%	Star schema	04/13/25	In a day	Jasky and Richard
100%	Second part of the powerpoint	04/13/25	In a day	Richard and Maitri did
100%	Recording	04/13/25	In a day	Richard did
100%	Voice annotation	04/13/25	In a day	Richard did

Gantt Chart (Project Planner)

Planning	1	1	1	1
Assigning work	1	2	1	2
Create tables	2	2	2	3
Create procedures	2	3	3	3
Create more tables	2	3	3	3
Alter every table	2	4	4	3
Drop relations	2	4	5	4
Truncate tables	2	4	5	4
Status Function	4	4	5	5
PowerPoint	6	6	6	6
To do list	6	3	6	3
Gantt Chart	6	2	6	2
Parsing Objects	6	4	6	4
Star Schemas	8	6	8	6
Voice annotation	8	5	9	5
Recording	10	5	12	5
Github	12	5	14	5



Project Plan

1. Meeting
2. Distribute workloads
3. Project Planner
4. To do list
5. Create Table User Authorization
6. Create Table WorkflowSteps
7. Create Procedure TrackWorkFlow
8. Add Tables Dim.ProductCategory and Subcategory
9. Alter every table to include 3 new columns
10. Drop all priv key and foreign key relations
11. Truncate Tables
12. Create Status Function
13. Create Loading Function
14. Sequence Objects
15. Star Schema
16. PowerPoint
17. Voice Annotation
18. Recording
19. Github

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Ashly Felix and Jaskaran Bains
-- Create date: 2025-04-12
-- Description: Creates the DbSecurity schema
-- =============================================
CREATE PROCEDURE [Project2].[CreateSchemaDbSecurity]
AS
BEGIN
    SET NOCOUNT ON;


    -- Create the Process schema if it doesn't exist
    IF NOT EXISTS (SELECT 1 FROM sys.schemas WHERE name = 'DbSecurity')
    BEGIN
        -- Use dynamic SQL to isolate CREATE SCHEMA in its own batch
        EXEC sp_executesql N'CREATE SCHEMA DbSecurity';
    END
END;
GO
```

```
In [ ]:
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Ashly Felix
-- Create date: 2025-04-12
-- Description: Creates the Process schema
-- =============================================
CREATE PROCEDURE [Project2].[CreateSchemaProcess]
AS
BEGIN
    SET NOCOUNT ON;

    -- Create the Process schema if it doesn't exist
    IF NOT EXISTS (SELECT 1 FROM sys.schemas WHERE name = 'Process')
    BEGIN
        -- Use dynamic SQL to isolate CREATE SCHEMA in its own batch
        EXEC sp_executesql N'CREATE SCHEMA Process';
    END
END;
GO
```

Create Schemas for dbsecurity and process.

CreateSchemaDbSecurity: Creates the DbSecurity schema if it doesn't already exist. It will be used for tables dealing with user authorization/security.

CreateSchemaProcess: Creates the Process schema to store logs and workflow steps.

CreatePkSequenceSchema: Creates the PkSequence schema to store SQL sequences (used for generating surrogate keys). It also logs this action using the tracking procedure.

Create the User Authorization Table

Drops the existing table if it exists.

Recreates DbSecurity.UserAuthorization, which stores information about each user or team member working on the database.

Inserts predefined rows with names and roles.

Logs the creation using usp_TrackWorkFlow.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Ashly Felix
-- Create date: 2025-04-12
-- Description: Creates the DbSecurity.UserAuthorization table
-- ============================================
CREATE PROCEDURE [Project2].[CreateUserAuthorizationTable]
    @UserAuthorizationKey INT -- To log who executed this
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Drop the table if it exists
    IF EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.TABLES
               WHERE TABLE_SCHEMA = 'DbSecurity' AND TABLE_NAME = 'UserAuthorization')
    BEGIN
        DROP TABLE DbSecurity.UserAuthorization;
    END;

    -- Create the DbSecurity.UserAuthorization table
    CREATE TABLE DbSecurity.UserAuthorization (
        UserAuthorizationKey INT NOT NULL PRIMARY KEY,
        ClassTime NCHAR(5) NULL CHECK (ClassTime IN ('10:45')),
        IndividualProject NVARCHAR(60) NULL DEFAULT 'PROJECT 2 RECREATE THE BICLASS DATABASE STAR SCHEMA',
        GroupMemberLastName NVARCHAR(35) NOT NULL,
        GroupMemberFirstName NVARCHAR(25) NOT NULL,
        GroupName NVARCHAR(20) NOT NULL,
        DateAdded DATETIME2 NULL DEFAULT SYSDATETIME(),
        DateOfLastUpdate DATETIME2 NULL DEFAULT SYSDATETIME()
    );

    -- Insert initial data
    INSERT INTO DbSecurity.UserAuthorization (
        UserAuthorizationKey,
        ClassTime,
        IndividualProject,
        GroupMemberLastName,
        GroupMemberFirstName,
        GroupName,
        DateAdded,
        DateOfLastUpdate
    )
    VALUES
        (1, '10:45', 'create store procedures', 'Bains', 'Jasky', 'Group#1', DEFAULT, DEFAULT),
        (2, '10:45', 'drop foreign keys', 'Vathada', 'Mai', 'Group#1', DEFAULT, DEFAULT),
        (3, '10:45', 'load data', 'Garnica', 'Richard', 'Group#1', DEFAULT, DEFAULT),
        (4, '10:45', 'truncate tables', 'Wang', 'William', 'Group#1', DEFAULT, DEFAULT),
        (5, '10:45', 'Create first tables', 'Felix', 'Ashly', 'Group#1', DEFAULT, DEFAULT),
        (6, '10:45', 'load data', 'Cristobal', 'David', 'Group#1', DEFAULT, DEFAULT),
        (7, '10:45', 'default', 'default', 'default', 'default', DEFAULT, DEFAULT);

    -- Compute row count for logging
    DECLARE @RowCount INT;
    SET @RowCount = (SELECT COUNT(*) FROM DbSecurity.UserAuthorization);

    -- Log the operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Created DbSecurity.UserAuthorization table',
        @WorkFlowStepTableRowCount = @RowCount,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

Create the Workflow Tracking Table & Procedure

CreateWorkFlowTable:

Creates Process.WorkflowSteps, a log table that stores:

-Description of work

-When it started and ended

-Which user ran it

usp_TrackWorkFlow:

A helper stored procedure to write logs into the WorkflowSteps table anytime a task is performed.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Ashly Felix
-- Create date: 2025-04-12
-- Description: Create the Process.WorkflowSteps Table
-- ============================================
CREATE PROCEDURE [Project2].[CreateWorkFlowTable]
    @UserAuthorizationKey INT -- To log who executed this
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @StartTime DATETIME2 = SYSDATETIME();

CREATE TABLE Process.WorkflowSteps (
    WorkFlowStepKey INT NOT NULL PRIMARY KEY,
    WorkFlowStepDescription NVARCHAR(100) NOT NULL,
    WorkFlowStepTableRowCount INT NULL DEFAULT 0,
    StartingDateTime DATETIME2 NULL DEFAULT SYSDATETIME(),
    EndingDateTime DATETIME2 NULL DEFAULT SYSDATETIME(),
    ClassTime VARCHAR(5) NULL DEFAULT '10:45',
    UserAuthorizationKey INT NOT NULL,
    CONSTRAINT FK_WorkflowSteps_UserAuthorization FOREIGN KEY (UserAuthorizationKey)
        REFERENCES DbSecurity.UserAuthorization (UserAuthorizationKey)
);

EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Create the Process.WorkflowSteps Table',
        @WorkFlowStepTableRowCount = 0,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

```sql
-- =============================================
-- Author: Ashly Felix
-- Procedure: Process.usp_TrackWorkFlow
-- Create date: 2025-04-12
-- Description: Logs a workflow step into Process.WorkflowSteps with description, row count, and timing details.
-- =============================================
CREATE PROCEDURE Process.usp_TrackWorkFlow
    @StartTime DATETIME2,
    @WorkFlowDescription NVARCHAR(100),
    @WorkFlowStepTableRowCount INT,
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Insert workflow step details
    INSERT INTO Process.WorkflowSteps (
        WorkFlowStepKey,
        WorkFlowStepDescription,
        WorkFlowStepTableRowCount,
        StartingDateTime,
        EndingDateTime,
        ClassTime,
        UserAuthorizationKey
    )
    VALUES (
        (SELECT ISNULL(MAX(WorkFlowStepKey), 0) + 1 FROM Process.WorkflowSteps),
        @WorkFlowDescription,
        @WorkFlowStepTableRowCount,
        @StartTime,
        SYSDATETIME(),
        (SELECT ClassTime FROM DbSecurity.UserAuthorization WHERE UserAuthorizationKey = @UserAuthorizationKey),
        @UserAuthorizationKey
    );
END;
GO
```

Create Dimension Tables

CreateProductCategory:

Creates the [CH01-01-Dimension].DimProductCategory table:

Has a surrogate key, category name, and user tracking info.

Logged using usp_TrackWorkFlow.

CreateProductSubcategory:

Creates the DimProductSubcategory table:

Links to the DimProductCategory table via foreign key.

Tracks which user created it.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Ashly Felix
-- Create date: 4/12/25
-- Description: Creates the DimProductCategory table
-- =============================================
CREATE PROCEDURE [Project2].[CreateProductCategory]
    @UserAuthorizationKey INT -- To log who executed this
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Drop the table if it exists
    IF EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.TABLES
               WHERE TABLE_SCHEMA = 'CH01-01-Dimension' AND TABLE_NAME = 'DimProductCategory')
    BEGIN
        DROP TABLE [CH01-01-Dimension].DimProductCategory;
    END;

    -- Create the DimProductCategory table
    CREATE TABLE [CH01-01-Dimension].DimProductCategory (
        ProductCategoryKey INT NOT NULL PRIMARY KEY,
        ProductCategory NVARCHAR(50) NOT NULL, -- Corrected column name to match diagram
        UserAuthorizationKey INT NOT NULL,
        DateAdded DATETIME2 NULL DEFAULT SYSDATETIME(),
        DateOfLastUpdate DATETIME2 NULL DEFAULT SYSDATETIME(),
        CONSTRAINT FK_DimProductCategory_UserAuthorization
            FOREIGN KEY (UserAuthorizationKey)
            REFERENCES DbSecurity.UserAuthorization (UserAuthorizationKey)
    );

    -- Log the operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Created Table DimProductCategory',
        @WorkFlowStepTableRowCount = 0,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Ashly Felix
-- Create date: 4/12/25
-- Description: Creates the DimProductSubcategory table
-- =============================================
CREATE PROCEDURE [Project2].[CreateProductSubcategory]
    @UserAuthorizationKey INT -- To log who executed this
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Drop the table if it exists
    IF EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.TABLES
               WHERE TABLE_SCHEMA = 'CH01-01-Dimension' AND TABLE_NAME = 'DimProductSubcategory')
    BEGIN
        DROP TABLE [CH01-01-Dimension].DimProductSubcategory;
    END;

    -- Create the DimProductSubcategory table
    CREATE TABLE [CH01-01-Dimension].DimProductSubcategory (
        ProductSubcategoryKey INT NOT NULL PRIMARY KEY,
        ProductSubcategory NVARCHAR(50) NOT NULL, -- Changed from SubcategoryName to ProductSubcategory
        ProductCategoryKey INT NOT NULL,
        UserAuthorizationKey INT NOT NULL,
        DateAdded DATETIME2 NULL DEFAULT SYSDATETIME(),
        DateOfLastUpdate DATETIME2 NULL DEFAULT SYSDATETIME(),
        CONSTRAINT FK_DimProductSubcategory_ProductCategory
            FOREIGN KEY (ProductCategoryKey)
            REFERENCES [CH01-01-Dimension].DimProductCategory (ProductCategoryKey),
        CONSTRAINT FK_DimProductSubcategory_UserAuthorization
            FOREIGN KEY (UserAuthorizationKey)
            REFERENCES DbSecurity.UserAuthorization (UserAuthorizationKey)
    );

    -- Log the operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Created Table DimProductSubcategory',
        @WorkFlowStepTableRowCount = 0,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

Add User Authorization Tracking to All Tables

AddUserAuthorizationKeyToTables:

Adds a UserAuthorizationKey column to every dimension and fact table.

The default value is 7, which seems to refer to a "default" user. This will help us when adding a column to the tables which will automatically give a 7/default table.

Ensures all data changes are tied to a user for auditing.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Ashly Felix
-- Create date: 2025-04-12
-- Description: Adds UserAuthorizationKey column to star schema tables
-- =============================================
CREATE PROCEDURE [Project2].[AddUserAuthorizationKeyToTables]
    @UserAuthorizationKey INT -- To log who executed this
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Add UserAuthorizationKey to dimension tables in CH01-01-Dimension schema
    ALTER TABLE [CH01-01-Dimension].DimCustomer
    ADD UserAuthorizationKey INT NOT NULL DEFAULT 7;

    ALTER TABLE [CH01-01-Dimension].DimGender
    ADD UserAuthorizationKey INT NOT NULL DEFAULT 7;

    ALTER TABLE [CH01-01-Dimension].DimMaritalStatus
    ADD UserAuthorizationKey INT NOT NULL DEFAULT 7;

    ALTER TABLE [CH01-01-Dimension].DimOccupation
    ADD UserAuthorizationKey INT NOT NULL DEFAULT 7;

    ALTER TABLE [CH01-01-Dimension].DimOrderDate
    ADD UserAuthorizationKey INT NOT NULL DEFAULT 7;

    ALTER TABLE [CH01-01-Dimension].DimProduct
    ADD UserAuthorizationKey INT NOT NULL DEFAULT 7;

    ALTER TABLE [CH01-01-Dimension].DimTerritory
    ADD UserAuthorizationKey INT NOT NULL DEFAULT 7;

    ALTER TABLE [CH01-01-Dimension].SalesManagers
    ADD UserAuthorizationKey INT NOT NULL DEFAULT 7;

    -- Add UserAuthorizationKey to fact table in CH01-01-Fact schema
    ALTER TABLE [CH01-01-Fact].Data
    ADD UserAuthorizationKey INT NOT NULL DEFAULT 7;

    -- Log the operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Added UserAuthorizationKey column to star schema tables',
        @WorkFlowStepTableRowCount = 0,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

Drop Foreign Keys

DropForeignKeysFromStarSchemaData:

Removes all foreign key constraints (the links between fact and dimension tables).

This is often done before reloading data or restructuring tables.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author:              Ashly Felix
-- Create date: 4/12/25
-- Description: Drop the Foreign Keys From the Star Schema
-- ============================================
CREATE PROCEDURE [Project2].[DropForeignKeysFromStarSchemaData]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

        DECLARE @StartTime DATETIME2 = SYSDATETIME();
    DECLARE @SQL NVARCHAR(MAX);

ALTER TABLE [CH01-01-Fact].[Data] DROP CONSTRAINT FK_Data_SalesManagers;
ALTER TABLE [CH01-01-Fact].[Data] DROP CONSTRAINT FK_Data_DimGender;
ALTER TABLE [CH01-01-Dimension].[DimProductSubcategory] DROP CONSTRAINT FK_DimProductSubcategory_ProductCategory;
ALTER TABLE [CH01-01-Fact].[Data] DROP CONSTRAINT FK_Data_DimMaritalStatus;
ALTER TABLE [CH01-01-Fact].[Data] DROP CONSTRAINT FK_Data_DimOccupation;
ALTER TABLE [CH01-01-Fact].[Data] DROP CONSTRAINT FK_Data_DimOrderDate;
ALTER TABLE [CH01-01-Fact].[Data] DROP CONSTRAINT FK_Data_DimTerritory;
ALTER TABLE [CH01-01-Fact].[Data] DROP CONSTRAINT FK_Data_DimProduct;
ALTER TABLE [CH01-01-Fact].[Data] DROP CONSTRAINT FK_Data_DimCustomer;
ALTER TABLE [CH01-01-Dimension].[DimProductCategory] DROP CONSTRAINT FK_DimProductCategory_UserAuthorization;
ALTER TABLE [CH01-01-Dimension].[DimProductSubcategory] DROP CONSTRAINT FK_DimProductSubcategory_UserAuthorization;

    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Dropped foreign keys from star schema',
        @WorkFlowStepTableRowCount = 0,
        @UserAuthorizationKey = @UserAuthorizationKey;

END;
GO
```

Remove Identity Property from Keys

RemoveIdentityFromStarSchemaTables:

Not fully visible, but the idea is:

Remove the auto-increment identity property from surrogate key columns.

Store original values in a backup table (OriginallyLoadedData).

This is useful if you're reloading or resetting data and need control over the key values.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Jaskaran Bains
-- Create date: 2025-04-12
-- Description: Removes identity property from star schema tables, storing keys in ParsedFileUpload.OriginallyLoadedData
-- =============================================
CREATE PROCEDURE [Project2].[RemoveIdentityFromStarSchemaTables]
    @UserAuthorizationKey INT -- To log who executed this
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure ParsedFileUpload.OriginallyLoadedData exists
    IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.TABLES
                   WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'OriginallyLoadedData')
    BEGIN
        SELECT *
        INTO ParsedFileUpload.OriginallyLoadedData
        FROM FileUpload.OriginallyLoadedData;
    END;

    -- Step 1: Add columns to ParsedFileUpload.OriginallyLoadedData and copy identity key values

    -- SalesManagers
    IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
                   WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'OriginallyLoadedData' AND COLUMN_NAME = 'Sales
    BEGIN
        ALTER TABLE ParsedFileUpload.OriginallyLoadedData
        ADD SalesManagerKey INT;
    END;
    UPDATE pfd
    SET pfd.SalesManagerKey = t.SalesManagerKey
    FROM ParsedFileUpload.OriginallyLoadedData pfd
    INNER JOIN [CH01-01-Dimension].SalesManagers t
    ON t.SalesManager = pfd.SalesManagerParsed;

    -- DimGender
    IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
                   WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'OriginallyLoadedData' AND COLUMN_NAME = 'Gende
    BEGIN
        ALTER TABLE ParsedFileUpload.OriginallyLoadedData
        ADD GenderKey INT;
    END;
```

```
UPDATE pfd
SET pfd.MaritalStatusKey = t.MaritalStatusKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
INNER JOIN [CH01-01-Dimension].DimMaritalStatus t
ON t.MaritalStatusDescription = pfd.MaritalStatusParsed;

-- DimOccupation
IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
               WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'Originally
BEGIN
    ALTER TABLE ParsedFileUpload.OriginallyLoadedData
    ADD OccupationKey INT;
END;
UPDATE pfd
SET pfd.OccupationKey = t.OccupationKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
INNER JOIN [CH01-01-Dimension].DimOccupation t
ON t.Occupation = pfd.OccupationParsed;

-- DimOrderDate
IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
               WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'Originally
BEGIN
    ALTER TABLE ParsedFileUpload.OriginallyLoadedData
    ADD OrderDateKey INT;
END;
UPDATE pfd
SET pfd.OrderDateKey = t.OrderDateKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
INNER JOIN [CH01-01-Dimension].DimOrderDate t
ON t.OrderDate = pfd.OrderDateFormatted;

-- DimTerritory
IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
               WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'Originally
BEGIN
    ALTER TABLE ParsedFileUpload.OriginallyLoadedData
    ADD TerritoryKey INT;
END;
UPDATE pfd
SET pfd.TerritoryKey = t.TerritoryKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
INNER JOIN [CH01-01-Dimension].DimTerritory t
ON t.TerritoryRegion = pfd.TerritoryRegionParsed
AND t.TerritoryCountry = pfd.TerritoryCountryParsed
AND t.TerritoryGroup = pfd.TerritoryGroupParsed;
```

```
-- DimProduct
IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
               WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'OriginallyLoadedData' AND COLUMN_NAME = 'Produ
BEGIN
    ALTER TABLE ParsedFileUpload.OriginallyLoadedData
    ADD ProductKey INT;
END;
UPDATE pfd
SET pfd.ProductKey = t.ProductKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
INNER JOIN [CH01-01-Dimension].DimProduct t
ON t.ProductName = pfd.ProductName;

-- DimCustomer
IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
               WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'OriginallyLoadedData' AND COLUMN_NAME = 'Custo
BEGIN
    ALTER TABLE ParsedFileUpload.OriginallyLoadedData
    ADD CustomerKey INT;
END;
UPDATE pfd
SET pfd.CustomerKey = t.CustomerKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
INNER JOIN [CH01-01-Dimension].DimCustomer t
ON t.CustomerName = pfd.CustomerNameParsed;

-- ProductCategories
IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
               WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'OriginallyLoadedData' AND COLUMN_NAME = 'Produ
BEGIN
    ALTER TABLE ParsedFileUpload.OriginallyLoadedData
    ADD ProductCategoryKey INT;
END;
UPDATE pfd
SET pfd.ProductCategoryKey = t.ProductCategoryKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
INNER JOIN [CH01-01-Dimension].ProductCategories t
ON t.ProductCategory = pfd.ProductCategoryParsed;

-- ProductSubcategories
IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
               WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'OriginallyLoadedData' AND COLUMN_NAME = 'Produ
BEGIN
    ALTER TABLE ParsedFileUpload.OriginallyLoadedData
    ADD ProductSubcategoryKey INT;
END;
UPDATE pfd
SET pfd.ProductSubcategoryKey = t.ProductSubcategoryKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
INNER JOIN [CH01-01-Dimension].ProductSubcategories t
ON t.ProductSubcategory = pfd.ProductSubcategoryParsed;
```

```
-- Data (fact table)
IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS
               WHERE TABLE_SCHEMA = 'ParsedFileUpload' AND TABLE_NAME = 'OriginallyLoadedData' AND COLUMN_NAME = 'Sale
BEGIN
    ALTER TABLE ParsedFileUpload.OriginallyLoadedData
    ADD SalesKey INT;
END;
UPDATE pfd
SET pfd.SalesKey = t.SalesKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
INNER JOIN [CH01-01-Fact].Data t
ON t.SalesAmount = pfd.SalesAmount; -- Adjust join condition as needed

-- Step 2: Remove identity property from each table
-- SalesManagers
ALTER TABLE [CH01-01-Dimension].SalesManagers
DROP CONSTRAINT PK_SalesManagers;
ALTER TABLE [CH01-01-Dimension].SalesManagers
DROP COLUMN SalesManagerKey;
ALTER TABLE [CH01-01-Dimension].SalesManagers
ADD SalesManagerKey INT NOT NULL;
UPDATE [CH01-01-Dimension].SalesManagers
SET SalesManagerKey = pfd.SalesManagerKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].SalesManagers.SalesManager = pfd.SalesManagerParsed;
ALTER TABLE [CH01-01-Dimension].SalesManagers
ADD CONSTRAINT PK_SalesManagers PRIMARY KEY (SalesManagerKey);

-- DimGender
ALTER TABLE [CH01-01-Dimension].DimGender
DROP CONSTRAINT PK_DimGender;
ALTER TABLE [CH01-01-Dimension].DimGender
DROP COLUMN GenderKey;
ALTER TABLE [CH01-01-Dimension].DimGender
ADD GenderKey INT NOT NULL;
UPDATE [CH01-01-Dimension].DimGender
SET GenderKey = pfd.GenderKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].DimGender.GenderDescription = pfd.GenderParsed;
ALTER TABLE [CH01-01-Dimension].DimGender
ADD CONSTRAINT PK_DimGender PRIMARY KEY (GenderKey);

-- DimMaritalStatus
ALTER TABLE [CH01-01-Dimension].DimMaritalStatus
DROP CONSTRAINT PK_DimMaritalStatus;
ALTER TABLE [CH01-01-Dimension].DimMaritalStatus
DROP COLUMN MaritalStatusKey;
ALTER TABLE [CH01-01-Dimension].DimMaritalStatus
ADD MaritalStatusKey INT NOT NULL;
UPDATE [CH01-01-Dimension].DimMaritalStatus
SET MaritalStatusKey = pfd.MaritalStatusKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].DimMaritalStatus.MaritalStatusDescription = pfd.MaritalStatusParsed;
ALTER TABLE [CH01-01-Dimension].DimMaritalStatus
ADD CONSTRAINT PK_DimMaritalStatus PRIMARY KEY (MaritalStatusKey);
```

```
-- DimOccupation
ALTER TABLE [CH01-01-Dimension].DimOccupation
DROP CONSTRAINT PK_DimOccupation;
ALTER TABLE [CH01-01-Dimension].DimOccupation
DROP COLUMN OccupationKey;
ALTER TABLE [CH01-01-Dimension].DimOccupation
ADD OccupationKey INT NOT NULL;
UPDATE [CH01-01-Dimension].DimOccupation
SET OccupationKey = pfd.OccupationKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].DimOccupation.Occupation = pfd.OccupationParsed;
ALTER TABLE [CH01-01-Dimension].DimOccupation
ADD CONSTRAINT PK_DimOccupation PRIMARY KEY (OccupationKey);

-- DimOrderDate
ALTER TABLE [CH01-01-Dimension].DimOrderDate
DROP CONSTRAINT PK_DimOrderDate;
ALTER TABLE [CH01-01-Dimension].DimOrderDate
DROP COLUMN OrderDateKey;
ALTER TABLE [CH01-01-Dimension].DimOrderDate
ADD OrderDateKey INT NOT NULL;
UPDATE [CH01-01-Dimension].DimOrderDate
SET OrderDateKey = pfd.OrderDateKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].DimOrderDate.OrderDate = pfd.OrderDateFormatted;
ALTER TABLE [CH01-01-Dimension].DimOrderDate
ADD CONSTRAINT PK_DimOrderDate PRIMARY KEY (OrderDateKey);

-- DimTerritory
ALTER TABLE [CH01-01-Dimension].DimTerritory
DROP CONSTRAINT PK_DimTerritory;
ALTER TABLE [CH01-01-Dimension].DimTerritory
DROP COLUMN TerritoryKey;
ALTER TABLE [CH01-01-Dimension].DimTerritory
ADD TerritoryKey INT NOT NULL;
UPDATE [CH01-01-Dimension].DimTerritory
SET TerritoryKey = pfd.TerritoryKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].DimTerritory.TerritoryRegion = pfd.TerritoryRegionParsed
AND [CH01-01-Dimension].DimTerritory.TerritoryCountry = pfd.TerritoryCountryParsed
AND [CH01-01-Dimension].DimTerritory.TerritoryGroup = pfd.TerritoryGroupParsed;
ALTER TABLE [CH01-01-Dimension].DimTerritory
ADD CONSTRAINT PK_DimTerritory PRIMARY KEY (TerritoryKey);

-- DimProduct
ALTER TABLE [CH01-01-Dimension].DimProduct
DROP CONSTRAINT PK_DimProduct;
ALTER TABLE [CH01-01-Dimension].DimProduct
DROP COLUMN ProductKey;
ALTER TABLE [CH01-01-Dimension].DimProduct
ADD ProductKey INT NOT NULL;
UPDATE [CH01-01-Dimension].DimProduct
SET ProductKey = pfd.ProductKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].DimProduct.ProductName = pfd.ProductName;
ALTER TABLE [CH01-01-Dimension].DimProduct
ADD CONSTRAINT PK_DimProduct PRIMARY KEY (ProductKey);
```

```
-- DimCustomer
ALTER TABLE [CH01-01-Dimension].DimCustomer
DROP CONSTRAINT PK_DimCustomer;
ALTER TABLE [CH01-01-Dimension].DimCustomer
DROP COLUMN CustomerKey;
ALTER TABLE [CH01-01-Dimension].DimCustomer
ADD CustomerKey INT NOT NULL;
UPDATE [CH01-01-Dimension].DimCustomer
SET CustomerKey = pfd.CustomerKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].DimCustomer.CustomerName = pfd.CustomerNameParsed;
ALTER TABLE [CH01-01-Dimension].DimCustomer
ADD CONSTRAINT PK_DimCustomer PRIMARY KEY (CustomerKey);

-- ProductCategories
ALTER TABLE [CH01-01-Dimension].ProductCategories
DROP CONSTRAINT PK_ProductCategories;
ALTER TABLE [CH01-01-Dimension].ProductCategories
DROP COLUMN ProductCategoryKey;
ALTER TABLE [CH01-01-Dimension].ProductCategories
ADD ProductCategoryKey INT NOT NULL;
UPDATE [CH01-01-Dimension].ProductCategories
SET ProductCategoryKey = pfd.ProductCategoryKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].ProductCategories.ProductCategory = pfd.ProductCategoryParsed;
ALTER TABLE [CH01-01-Dimension].ProductCategories
ADD CONSTRAINT PK_ProductCategories PRIMARY KEY (ProductCategoryKey);

-- ProductSubcategories
ALTER TABLE [CH01-01-Dimension].ProductSubcategories
DROP CONSTRAINT PK_ProductSubcategories;
ALTER TABLE [CH01-01-Dimension].ProductSubcategories
DROP COLUMN ProductSubcategoryKey;
ALTER TABLE [CH01-01-Dimension].ProductSubcategories
ADD ProductSubcategoryKey INT NOT NULL;
UPDATE [CH01-01-Dimension].ProductSubcategories
SET ProductSubcategoryKey = pfd.ProductSubcategoryKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Dimension].ProductSubcategories.ProductSubcategory = pfd.ProductSubcategoryParsed;
ALTER TABLE [CH01-01-Dimension].ProductSubcategories
ADD CONSTRAINT PK_ProductSubcategories PRIMARY KEY (ProductSubcategoryKey);

-- Data
ALTER TABLE [CH01-01-Fact].Data
DROP CONSTRAINT PK_Data;
ALTER TABLE [CH01-01-Fact].Data
DROP COLUMN SalesKey;
ALTER TABLE [CH01-01-Fact].Data
ADD SalesKey INT NOT NULL;
UPDATE [CH01-01-Fact].Data
SET SalesKey = pfd.SalesKey
FROM ParsedFileUpload.OriginallyLoadedData pfd
WHERE [CH01-01-Fact].Data.SalesAmount = pfd.SalesAmount;
ALTER TABLE [CH01-01-Fact].Data
ADD CONSTRAINT PK_Data PRIMARY KEY (SalesKey);
```

```
    -- Log the operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Removed identity property from star schema tables and stored keys in ParsedFileUpload.Orig
        @WorkFlowStepTableRowCount = 0,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

Truncate Tables

Project2.TruncateStarSchemaTables

Purpose: Clears out (truncates) all records from the star schema tables before loading new data.

Steps:

Records the start time.

Executes TRUNCATE TABLE on each dimension and fact table under the schemas CH01-01-Dimension and CH01-01-Fact.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Ashly Felix
-- Create date: 4/12/25
-- Description: Truncates all star schema tables
-- =============================================
CREATE PROCEDURE [Project2].[TruncateStarSchemaTables]
    @UserAuthorizationKey INT -- To log who executed this
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Truncate dimension tables in CH01-01-Dimension schema
    TRUNCATE TABLE [CH01-01-Dimension].[DimCustomer];
    TRUNCATE TABLE [CH01-01-Dimension].[DimGender];
    TRUNCATE TABLE [CH01-01-Dimension].[DimMaritalStatus];
    TRUNCATE TABLE [CH01-01-Dimension].[DimOccupation];
    TRUNCATE TABLE [CH01-01-Dimension].[DimOrderDate];
    TRUNCATE TABLE [CH01-01-Dimension].[DimProduct];
    TRUNCATE TABLE [CH01-01-Dimension].[DimProductCategory];
    TRUNCATE TABLE [CH01-01-Dimension].[DimProductSubcategory];
    TRUNCATE TABLE [CH01-01-Dimension].[DimTerritory];
    TRUNCATE TABLE [CH01-01-Dimension].[SalesManagers];
    TRUNCATE TABLE [CH01-01-Fact].[Data];

    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Truncated all star schema tables',
        @WorkFlowStepTableRowCount = 0,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

Row Status Function

Project2.ShowTableStatusRowCount

Purpose: Displays the number of rows in each star schema table.

Steps:

Outputs the TableStatus (passed as a parameter), table name, and row count using COUNT(*).

Logs the action using the tracking procedure.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Richard Garnica
-- Create date: 2025-04-13
-- Description: Shows amount of rows in the tables
-- ============================================
CREATE PROCEDURE [Project2].[ShowTableStatusRowCount]
@TableStatus VARCHAR(64),
@UserAuthorizationKey INT

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;


        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.DimCustomer', COUNT(*) FROM [CH01-01-Dimension].D
        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.DimGender', COUNT(*) FROM [CH01-01-Dimension].Dim
        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.DimMaritalStatus', COUNT(*) FROM [CH01-01-Dimensi
        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.DimOccupation', COUNT(*) FROM [CH01-01-Dimension]
        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.DimOrderDate', COUNT(*) FROM [CH01-01-Dimension].I
        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.DimProduct', COUNT(*) FROM [CH01-01-Dimension].Di
        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.DimProductCategory', COUNT(*) FROM [CH01-01-Dimen
        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.DimProductSubcategory', COUNT(*) FROM [CH01-01-Di
        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.DimTerritory', COUNT(*) FROM [CH01-01-Dimension].I
        select TableStatus = @TableStatus, TableName ='CH01-01-Dimension.SalesManagers', COUNT(*) FROM [CH01-01-Dimension]
        select TableStatus = @TableStatus, TableName ='CH01-01-Fact.Data', COUNT(*) FROM [CH01-01-Fact].Data

        -- Log the operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Dynamically reported row counts for star schema tables',
        @WorkFlowStepTableRowCount = 0,
        @UserAuthorizationKey = @UserAuthorizationKey;

END
GO
```

Loading/ Filling Functions for Truncated Tables

Codes will appear in the next few pages

LoadDimCustomer

Loads distinct customers.

Uses a sequence object PkSequence.DimCustomerSequenceObject to generate surrogate keys.

Groups by CustomerName.

LoadDimGender

Loads unique gender values (M, F, etc.).

Adds a descriptive label like Male, Female, or Other.

LoadDimMaritalStatus

Loads marital statuses (M = Married, S = Single, else Other).

Grouped for uniqueness.

LoadDimOccupation

Loads distinct occupations using a subquery (SELECT DISTINCT).

Simple load of raw values with surrogate key.

LoadDimOrderDate

Loads dates from OriginallyLoadedData, and includes:

MonthName, MonthNumber, and Year.

Uses GROUP BY on all four columns to avoid duplicates.

LoadDimCustomer

Loads distinct customers.

Uses a sequence object PkSequence.DimCustomerSequenceObject to generate surrogate keys.

Groups by CustomerName.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Ashly Felix
-- Create date: 2025-04-13
-- Description: Loads distinct customers into DimCustomer using surrogate keys from sequence object
-- ============================================
CREATE PROCEDURE [Project2].[LoadDimCustomer]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'DimCustomerSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.DimCustomerSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.DimCustomerSequenceObject RESTART WITH 1;
    END;

    -- Insert distinct customers
    INSERT INTO [CH01-01-Dimension].[DimCustomer] (
        CustomerKey,
        CustomerName,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.DimCustomerSequenceObject,
        d.CustomerName,
        @UserAuthorizationKey
    FROM FileUpload.OriginallyLoadedData d
    WHERE d.CustomerName IS NOT NULL
    GROUP BY d.CustomerName;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded DimCustomer with surrogate keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

LoadDimGender

Loads unique gender values (M, F, etc.).

Adds a descriptive label like Male, Female, or Other.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Ashly Felix
-- Create date: 2025-04-13
-- Description: Loads distinct gender values into DimGender using surrogate keys from sequence object
-- ============================================
CREATE PROCEDURE [Project2].[LoadDimGender]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'DimGenderSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.DimGenderSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.DimGenderSequenceObject RESTART WITH 1;
    END;

    -- Insert distinct gender values with surrogate key
    INSERT INTO [CH01-01-Dimension].[DimGender] (
        GenderKey,
        Gender,
        GenderDescription,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.DimGenderSequenceObject,
        d.Gender,
        CASE
            WHEN d.Gender = 'M' THEN 'Male'
            WHEN d.Gender = 'F' THEN 'Female'
            ELSE 'Other'
        END,
        @UserAuthorizationKey
    FROM FileUpload.OriginallyLoadedData d
    WHERE d.Gender IS NOT NULL
    GROUP BY d.Gender;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded DimGender with surrogate keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

LoadDimMartialStatus

Loads marital statuses (M = Married, S = Single, else Other).

Grouped for uniqueness.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Ashly Felix
-- Create date: 2025-04-13
-- Description: Loads distinct marital statuses into DimMaritalStatus using surrogate keys from sequence object
-- ============================================
CREATE PROCEDURE [Project2].[LoadDimMaritalStatus]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'DimMaritalStatusSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.DimMaritalStatusSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.DimMaritalStatusSequenceObject RESTART WITH 1;
    END;

    -- Insert distinct marital statuses with surrogate key
    INSERT INTO [CH01-01-Dimension].[DimMaritalStatus] (
        MaritalStatusKey,
        MaritalStatus,
        MaritalStatusDescription,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.DimMaritalStatusSequenceObject,
        d.MaritalStatus,
        CASE
            WHEN d.MaritalStatus = 'M' THEN 'Married'
            WHEN d.MaritalStatus = 'S' THEN 'Single'
            ELSE 'Other'
        END,
        @UserAuthorizationKey
    FROM FileUpload.OriginallyLoadedData d
    WHERE d.MaritalStatus IS NOT NULL
    GROUP BY d.MaritalStatus;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded DimMaritalStatus with surrogate keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

LoadDimOccupation

Loads distinct occupations using a subquery (SELECT DISTINCT).

Simple load of raw values with surrogate key.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Ashly Felix
-- Create date: 2025-04-13
-- Description: Loads distinct occupations into DimOccupation using surrogate keys via sequence object
-- =============================================
CREATE PROCEDURE [Project2].[LoadDimOccupation]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'DimOccupationSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.DimOccupationSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.DimOccupationSequenceObject RESTART WITH 1;
    END;

    -- Insert distinct occupations using surrogate keys
    INSERT INTO [CH01-01-Dimension].[DimOccupation] (
        OccupationKey,
        Occupation,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.DimOccupationSequenceObject,
        d.Occupation,
        @UserAuthorizationKey
    FROM (
        SELECT DISTINCT d.Occupation
        FROM FileUpload.OriginallyLoadedData d
        WHERE d.Occupation IS NOT NULL
    ) AS UniqueOccupations;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded DimOccupation with surrogate keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

LoadDimOrderDate

Loads dates from OriginallyLoadedData, and includes:

MonthName, MonthNumber, and Year.

Uses GROUP BY on all four columns to avoid duplicates.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Ashly Felix
-- Create date: 2025-04-13
-- Description: Loads distinct order dates into DimOrderDate using surrogate keys from sequence object
-- ============================================
CREATE PROCEDURE [Project2].[LoadDimOrderDate]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'DimOrderDateSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.DimOrderDateSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.DimOrderDateSequenceObject RESTART WITH 1;
    END;

    -- Insert distinct order dates
    INSERT INTO [CH01-01-Dimension].[DimOrderDate] (
        OrderDateKey,
        OrderDate,
        MonthName,
        MonthNumber,
        Year,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.DimOrderDateSequenceObject,
        d.OrderDate,
        d.MonthName,
        d.MonthNumber,
        d.Year,
        @UserAuthorizationKey
    FROM FileUpload.OriginallyLoadedData d
    WHERE d.OrderDate IS NOT NULL
    GROUP BY d.OrderDate, d.MonthName, d.MonthNumber, d.Year;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded DimOrderDate with surrogate keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

[Project2].[LoadDimProduct]
Purpose:
Load distinct product records from raw data into [CH01-01-Dimension].[DimProduct] using surrogate keys.

Ensures existence of PkSequence.DimProductSequenceObject; creates or restarts it.
Uses ROW_NUMBER() to ensure one record per ProductCode (prefers first alphabetical ProductName).
Inserts deduplicated records with:
Auto-generated ProductKey
NULL for ProductSubcategoryKey (possibly to be updated later)
All other product attributes + UserAuthorizationKey.
Logs the workflow execution via Process.usp_TrackWorkFlow.

```sql
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: Ashly Felix
-- Create date: 2025-04-13
-- Description: Loads distinct products into DimProduct with surrogate keys via sequence object
-- =============================================
CREATE PROCEDURE [Project2].[LoadDimProduct]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'DimProductSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.DimProductSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.DimProductSequenceObject RESTART WITH 1;
    END;

    -- Load distinct products by ProductCode (1st name per code)
    INSERT INTO [CH01-01-Dimension].[DimProduct] (
        ProductKey,
        ProductSubcategoryKey,
        ProductCategory,
        ProductSubcategory,
        ProductCode,
        ProductName,
        Color,
        ModelName,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.DimProductSequenceObject,
        NULL AS ProductSubcategoryKey,
        ProductCategory,
        ProductSubcategory,
        ProductCode,
        ProductName,
        Color,
        ModelName,
        @UserAuthorizationKey
    FROM (
        SELECT *,
               ROW_NUMBER() OVER (PARTITION BY ProductCode ORDER BY ProductName) AS rn
        FROM FileUpload.OriginallyLoadedData
        WHERE ProductCode IS NOT NULL
    ) AS Ranked
    WHERE rn = 1;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded DimProduct with surrogate keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
```

[Project2].[LoadDimTerritory]
Purpose:
Populate [CH01-01-Dimension].[DimTerritory] with unique territory combinations.

Checks/creates/restarts PkSequence.DimTerritorySequenceObject.
Extracts distinct TerritoryGroup, TerritoryCountry, and TerritoryRegion from source.
Assigns surrogate TerritoryKey for each combination.
Logs execution stats via usp_TrackWorkFlow.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Maitri Vathada
-- Create date: 2025-04-13
-- Description: Loads distinct territory combinations into DimTerritory using surrogate keys from a sequence object
-- ============================================
CREATE PROCEDURE [Project2].[LoadDimTerritory]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'DimTerritorySequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.DimTerritorySequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.DimTerritorySequenceObject RESTART WITH 1;
    END;

    -- Insert distinct territory values
    INSERT INTO [CH01-01-Dimension].[DimTerritory] (
        TerritoryKey,
        TerritoryGroup,
        TerritoryCountry,
        TerritoryRegion,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.DimTerritorySequenceObject,
        d.TerritoryGroup,
        d.TerritoryCountry,
        d.TerritoryRegion,
        @UserAuthorizationKey
    FROM (
        SELECT DISTINCT TerritoryGroup, TerritoryCountry, TerritoryRegion
        FROM FileUpload.OriginallyLoadedData
        WHERE TerritoryGroup IS NOT NULL
          AND TerritoryCountry IS NOT NULL
          AND TerritoryRegion IS NOT NULL
    ) AS d;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded DimTerritory with surrogate keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
```

[Project2].[LoadSalesManagers]
Purpose:
Load unique sales manager + category pairs into [CH01-01-Dimension].[SalesManagers] and assign office locations.

Checks/creates/restarts SalesManagersSequenceObject.
Groups by SalesManager and ProductCategory to remove duplicates.
Adds business rule logic:
Assigns "Redmond" if name starts with Marco or Maurizio.
Assigns "Seattle" for Alberto or others.
Adds surrogate key + UserAuthorizationKey.
Tracks via usp_TrackWorkFlow.

```sql
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Jaskaran Bains
-- Create date: 2025-04-13
-- Description: Loads distinct sales managers into Dim table using surrogate key sequence and conditional office logic
-- ============================================
CREATE PROCEDURE [Project2].[LoadSalesManagers]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'SalesManagersSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.SalesManagersSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.SalesManagersSequenceObject RESTART WITH 1;
    END;

    -- Insert distinct sales managers
    INSERT INTO [CH01-01-Dimension].SalesManagers (
        SalesManagerKey,
        Category,
        SalesManager,
        Office,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.SalesManagersSequenceObject,
        ProductCategory,
        SalesManager,
        CASE
            WHEN SalesManager LIKE 'Marco%' THEN 'Redmond'
            WHEN SalesManager LIKE 'Alberto%' THEN 'Seattle'
            WHEN SalesManager LIKE 'Maurizio%' THEN 'Redmond'
            ELSE 'Seattle'
        END,
        @UserAuthorizationKey
    FROM FileUpload.OriginallyLoadedData
    WHERE SalesManager IS NOT NULL AND ProductCategory IS NOT NULL
    GROUP BY ProductCategory, SalesManager;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded SalesManagers with surrogate keys from sequence object and assigned office values',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END
GO
```

[Project2].[LoadProductCategories]
Purpose:
Populate [CH01-01-Dimension].[ProductCategories] with unique product categories.

Ensures ProductCategoriesSequenceObject exists/restarts.
Extracts distinct ProductCategory values from source.
Assigns new surrogate keys as ProductCategoryKey.
Logs workflow with usp_TrackWorkFlow.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Jaskaran Bains
-- Create date: 2025-04-13
-- Description: Loads distinct product categories into Dim table using surrogate key sequence
-- ============================================
CREATE PROCEDURE [Project2].[LoadProductCategories]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'ProductCategoriesSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.ProductCategoriesSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.ProductCategoriesSequenceObject RESTART WITH 1;
    END;

    -- Insert distinct product categories
    INSERT INTO [CH01-01-Dimension].[ProductCategories] (
        ProductCategoryKey,
        ProductCategory,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.ProductCategoriesSequenceObject,
        d.ProductCategory,
        @UserAuthorizationKey
    FROM FileUpload.OriginallyLoadedData d
    WHERE d.ProductCategory IS NOT NULL
    GROUP BY d.ProductCategory;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded ProductCategories with surrogate keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

[Project2].[LoadProductSubcategories]
Purpose:
Populate [CH01-01-Dimension].[ProductSubcategories] with unique product subcategories and their corresponding category foreign keys.

Manages ProductSubcategoriesSequenceObject (check/create/restart).
Selects distinct (ProductCategory, ProductSubcategory) pairs.
Joins with [ProductCategories] to get foreign key ProductCategoryKey.
Inserts deduplicated records with surrogate key + relationship.
Tracks insert activity with usp_TrackWorkFlow.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Jaskaran Bains
-- Create date: 2025-04-13
-- Description: Loads distinct product subcategories into Dim table using surrogate keys
-- ============================================
CREATE PROCEDURE [Project2].[LoadProductSubcategories]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'ProductSubcategoriesSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.ProductSubcategoriesSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.ProductSubcategoriesSequenceObject RESTART WITH 1;
    END;

    -- Insert distinct product subcategories with resolved ProductCategoryKey
    INSERT INTO [CH01-01-Dimension].[ProductSubcategories] (
        ProductSubcategoryKey,
        ProductCategoryKey,
        ProductSubcategory,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.ProductSubcategoriesSequenceObject,
        pc.ProductCategoryKey,
        d.ProductSubcategory,
        @UserAuthorizationKey
    FROM (
        SELECT DISTINCT ProductCategory, ProductSubcategory
        FROM FileUpload.OriginallyLoadedData
        WHERE ProductSubcategory IS NOT NULL AND ProductCategory IS NOT NULL
    ) d
    INNER JOIN [CH01-01-Dimension].[ProductCategories] pc
        ON d.ProductCategory = pc.ProductCategory;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded ProductSubcategories with surrogate keys and foreign keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

[Project2].[Load Data]

Purpose: Load fact table with data from dimensions using a surrogate key sequence.

Creates or resets PkSequence.DataSequenceObject.

Joins dimension tables to enrich OriginallyLoadedData.

Uses NEXT VALUE FOR to assign SalesKey.

Logs the workflow.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Ashly Felix
-- Create date: 2025-04-13
-- Description: Loads fact table Data using surrogate keys from sequence object
-- ============================================
CREATE PROCEDURE [Project2].[LoadData]
    @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Ensure sequence exists and restart it
    IF NOT EXISTS (
        SELECT 1 FROM sys.sequences
        WHERE name = 'DataSequenceObject'
          AND SCHEMA_NAME(schema_id) = 'PkSequence'
    )
    BEGIN
        CREATE SEQUENCE PkSequence.DataSequenceObject
            AS INT START WITH 1 INCREMENT BY 1 MINVALUE 1 MAXVALUE 2147483647 CACHE;
    END
    ELSE
    BEGIN
        ALTER SEQUENCE PkSequence.DataSequenceObject RESTART WITH 1;
    END;

    -- Insert data into the fact table by joining with dimension tables
    INSERT INTO [CH01-01-Fact].[Data] (
        SalesKey,
        SalesManagerKey,
        OccupationKey,
        TerritoryKey,
        ProductKey,
        CustomerKey,
        ProductCategory,
        SalesManager,
        ProductSubcategory,
        ProductCode,
        ProductName,
        Color,
        ModelName,
        OrderQuantity,
        UnitPrice,
        ProductStandardCost,
        SalesAmount,
        OrderDate,
        MonthName,
        MonthNumber,
        Year,
        CustomerName,
        MaritalStatus,
        Gender,
        Education,
        Occupation,
```

```
        TerritoryRegion,
        TerritoryCountry,
        TerritoryGroup,
        UserAuthorizationKey
    )
    SELECT
        NEXT VALUE FOR PkSequence.DataSequenceObject AS SalesKey,
        sm.SalesManagerKey,
        occ.OccupationKey,
        terr.TerritoryKey,
        prod.ProductKey,
        cust.CustomerKey,
        f.ProductCategory,
        f.SalesManager,
        f.ProductSubcategory,
        f.ProductCode,
        f.ProductName,
        f.Color,
        f.ModelName,
        f.OrderQuantity,
        f.UnitPrice,
        f.ProductStandardCost,
        f.SalesAmount,
        f.OrderDate,
        f.MonthName,
        f.MonthNumber,
        f.Year,
        f.CustomerName,
        f.MaritalStatus,
        f.Gender,
        f.Education,
        f.Occupation,
        f.TerritoryRegion,
        f.TerritoryCountry,
        f.TerritoryGroup,
        @UserAuthorizationKey
    FROM FileUpload.OriginallyLoadedData f
    LEFT JOIN [CH01-01-Dimension].SalesManagers sm
        ON f.SalesManager = sm.SalesManager AND f.ProductCategory = sm.Category
    LEFT JOIN [CH01-01-Dimension].DimOccupation occ
        ON f.Occupation = occ.Occupation
    LEFT JOIN [CH01-01-Dimension].DimTerritory terr
        ON f.TerritoryRegion = terr.TerritoryRegion
        AND f.TerritoryCountry = terr.TerritoryCountry
        AND f.TerritoryGroup = terr.TerritoryGroup
    LEFT JOIN [CH01-01-Dimension].DimProduct prod
        ON f.ProductCode = prod.ProductCode
    LEFT JOIN [CH01-01-Dimension].DimCustomer cust
        ON f.CustomerName = cust.CustomerName;

    -- Log operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Loaded Data fact table with surrogate keys from sequence object',
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

[Project2].[CreateParsedFileUpload]

Purpose: Creates staging schema and table with all dimension data combinations.

Creates ParsedFileUpload schema if missing.

Drops and recreates OriginallyLoadedData.

Cross joins all dimension tables (generates all combos).

Logs the workflow.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Ashly Felix
-- Create date: 2025-04-13
-- Description: Creates the ParsedFileUpload schema and populates OriginallyLoadedData table with data from dimension tabl
-- ============================================
CREATE PROCEDURE [Project2].[CreateParsedFileUpload]
    @UserAuthorizationKey INT -- To log who executed this
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Create the ParsedFileUpload schema if it doesn't exist
    IF SCHEMA_ID('ParsedFileUpload') IS NULL
    BEGIN
        EXEC sp_executesql N'CREATE SCHEMA ParsedFileUpload';
    END;

    -- Drop the table if it exists
    IF OBJECT_ID('ParsedFileUpload.OriginallyLoadedData') IS NOT NULL
    BEGIN
        DROP TABLE ParsedFileUpload.OriginallyLoadedData;
    END;

    -- Populate ParsedFileUpload.OriginallyLoadedData by joining dimension tables
    SELECT
        cust.CustomerKey,
        cust.CustomerName,
        gen.GenderKey,
        gen.Gender,
        gen.GenderDescription,
        ms.MaritalStatusKey,
        ms.MaritalStatus,
        ms.MaritalStatusDescription,
        occ.OccupationKey,
        occ.Occupation,
        od.OrderDateKey,
        od.OrderDate,
        od.MonthName,
        od.MonthNumber,
        od.Year,
        terr.TerritoryKey,
        terr.TerritoryRegion,
        terr.TerritoryCountry,
        terr.TerritoryGroup,
        sm.SalesManagerKey,
        sm.SalesManager,
        sm.Category AS ProductCategory,
        sm.Office,
        pc.ProductCategoryKey,
        pc.ProductCategory,
        psc.ProductSubcategoryKey,
        psc.ProductSubcategory,
        prod.ProductKey,
        prod.ProductCode,
        prod.ProductName,
        prod.Color,
        prod.ModelName
```

```
    INTO ParsedFileUpload.OriginallyLoadedData
    FROM [CH01-01-Dimension].DimCustomer cust
    CROSS JOIN [CH01-01-Dimension].DimGender gen
    CROSS JOIN [CH01-01-Dimension].DimMaritalStatus ms
    CROSS JOIN [CH01-01-Dimension].DimOccupation occ
    CROSS JOIN [CH01-01-Dimension].DimOrderDate od
    CROSS JOIN [CH01-01-Dimension].DimTerritory terr
    CROSS JOIN [CH01-01-Dimension].SalesManagers sm
    CROSS JOIN [CH01-01-Dimension].ProductCategories pc
    CROSS JOIN [CH01-01-Dimension].ProductSubcategories psc
    CROSS JOIN [CH01-01-Dimension].DimProduct prod;

    -- Log the operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Created ParsedFileUpload schema and populated OriginallyLoadedData table with dimension da
        @WorkFlowStepTableRowCount = @@ROWCOUNT,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

[Project2].[AddForeignKeysToStarSchemaData]

Purpose: Restores foreign key constraints in the star schema.

Adds FKs between fact and dimension tables.

Ensures data integrity.

Logs the workflow.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author: Jaskaran Bains
-- Create date: 2025-04-13
-- Description: Restores foreign keys to star schema tables
-- ============================================
CREATE PROCEDURE [Project2].[AddForeignKeysToStarSchemaData]
      @UserAuthorizationKey INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @StartTime DATETIME2 = SYSDATETIME();

    -- Restore foreign keys in ProductSubcategories
    ALTER TABLE [CH01-01-Dimension].ProductSubcategories
    ADD CONSTRAINT FK_ProductSubcategory_DimProductCategory
    FOREIGN KEY (ProductCategoryKey)
    REFERENCES [CH01-01-Dimension].ProductCategories (ProductCategoryKey);

    -- Restore foreign keys in DimProduct
    ALTER TABLE [CH01-01-Dimension].DimProduct
    ADD CONSTRAINT FK_DimProduct_ProductSubcategory
    FOREIGN KEY (ProductSubcategoryKey)
    REFERENCES [CH01-01-Dimension].ProductSubcategories (ProductSubcategoryKey);

    -- Restore foreign keys in Data
    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_SalesManagers
    FOREIGN KEY (SalesManagerKey)
    REFERENCES [CH01-01-Dimension].SalesManagers (SalesManagerKey);

    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_DimGender
    FOREIGN KEY (GenderKey)
    REFERENCES [CH01-01-Dimension].DimGender (GenderKey);

    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_DimMaritalStatus
    FOREIGN KEY (MaritalStatusKey)
    REFERENCES [CH01-01-Dimension].DimMaritalStatus (MaritalStatusKey);

    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_DimOccupation
    FOREIGN KEY (OccupationKey)
    REFERENCES [CH01-01-Dimension].DimOccupation (OccupationKey);

    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_DimOrderDate
    FOREIGN KEY (OrderDateKey)
    REFERENCES [CH01-01-Dimension].DimOrderDate (OrderDateKey);

    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_DimTerritory
    FOREIGN KEY (TerritoryKey)
    REFERENCES [CH01-01-Dimension].DimTerritory (TerritoryKey);

    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_DimProduct
    FOREIGN KEY (ProductKey)
    REFERENCES [CH01-01-Dimension].DimProduct (ProductKey);

    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_DimCustomer
    FOREIGN KEY (CustomerKey)
    REFERENCES [CH01-01-Dimension].DimCustomer (CustomerKey);
```

```
    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_ProductCategories
    FOREIGN KEY (ProductCategoryKey)
    REFERENCES [CH01-01-Dimension].ProductCategories (ProductCategoryKey);

    ALTER TABLE [CH01-01-Fact].Data
    ADD CONSTRAINT FK_Data_ProductSubcategories
    FOREIGN KEY (ProductSubcategoryKey)
    REFERENCES [CH01-01-Dimension].ProductSubcategories (ProductSubcategoryKey);

    -- Log the operation
    EXEC Process.usp_TrackWorkFlow
        @StartTime = @StartTime,
        @WorkFlowDescription = 'Restored foreign keys to star schema tables',
        @WorkFlowStepTableRowCount = 0,
        @UserAuthorizationKey = @UserAuthorizationKey;
END;
GO
```

Systems Life Cycle

Project Tracking Techniques

The following sides will contain information about our meetings and who was assigned for what reponsiblities

Recreating BIClass Star Schema Project – Team meeting notes

Meeting 1: Preparing Project

April 11th
Time: 11 am – 11 pm (12 hours)
Location - Discord voice chat

Attendance:

Name	Present?
Ashly	Yes
David	Yes
Jasky	Yes
Maitri	Yes
Richard	Yes
William	Yes

Agenda:

1. Review overall project goals and structure
2. Assign roles and responsibilities
3. Set up a shared folder for collaboration
4. Agree on communication method (Discord)

Discussion Notes:

1. The project will be split up into three days
    a. Day 1: Going over requirements and possibly starting the project
    b. Day 2: Must start project (if you have not done so)
    c. Day 3: Finishing touches on the project.
        i. Includes creation of PowerPoints
2. Assign roles
    a. Manager: Jasky
    b. Backup Manager: Ashly
    c. notebook Organization/parsed file: David
    d. secretary/vhdx: Maitri
    e. Format meeting notes/ life cycle: Richard
    f. PowerPoint/video editing: William
3. Agree on forms of communication and hours
    a. With some exceptions
4. Shared work folder: Google Drive

Project Tracking Techniques

Information about our meeting that occurred on day 2

Meeting 3: Final day of Project (PPT)

April 13th
Time: 11 am -11 pm (12 hours)
Location - Discord voice chat

Attendance:

Name	Present?
Ashly	Yes
David	Yes
Jasky	Yes
Maitri	Yes
Richard	Yes
William	Yes

Agenda:

1. Everyone is on point and has completed their assigned tasks
2. Finish remaining tasks
    a. SQL tasks and PowerPoint Tasks
3. Submit project

Discussion Notes:

4. Everyone is on point and has completed their assigned tasks
5. Tasks left for SQL
    a. Create Status Function (Jasky and David)
    b. Parsing Objects (David, Maitri, and William
    c. Star Schema (Jasky and Richard)
    d. Touching up code
6. Remaining task (non-SQL)
    a. Create PPTs
    b. Finish up To-Do List
    c. Finish up Gnatt Chart
    d. Recording of PPT and Voice Annotations
7. Submit project

Project Tracking Techniques

Information about our meeting that occurred on day 3

Meeting 2: Start of Project (SQL start)

April 12th
Time: 11 am – 11 pm (12 hours)
Location - Discord voice chat

Attendance:

Name	Present?
Ashly	Yes
David	Yes
Jasky	Yes
Maitri	Yes
Richard	Yes
William	Yes

Agenda:

1. Begin database project
2. Create Schemas and tables
3. Assign stored procedures and functions
4. Handle truncation logic and necessary key drops
5. Create sequences, columns, etc. (reusable parts of code)

Discussion Notes:

1. Ashly has already started the project
    a. Created the necessary Schemas for associated tables
        i. DbSecurity Schema
        ii. Process Schema
    b. Created the following tables using those Schema
        i. DbSecurity.UserAuthorization table
        ii. Process.WorkflowSteps table
    c. Dropped all primary keys and foreign key relations
2. Create procedure (Ashly)
    a. TrackWorkFlow
3. Create two new tables (Ashly)
    a. ProductCategory
    b. Subcategory
4. Alter all tables to include the three new columns (Ashly)
    a. UserAuthorizationKey
    b. DateAdded
    c. DateOfLastUpdate
5. Sequence objects (Jasky and David)
    a. Create Schema PkSequence
    b. Create sequence called PkSequence
        i. Should be used on every dimension and fact table
6. Truncate Tables (Ashly)
    a. Record row count before truncating to verify the final result is correct after truncating has taken place and data was reloaded
7. Status function (Jasky and David)
    a. Task for April 13th

Thanks for reading!