Sri Lanka Institute of Information Technology

Faculty of Computing

Data Warehousing and Business Intelligence (IT3021)

Assignment 1

Submitted By:

Pallepitiya N.D.

IT19005386

Y3S1-15(DS-Weekday)

Table of Content

1. Data Set Selection

DonorsChoose was Founded in 2000 by a highschool teacher in Bronx, DonorsChoose empowers school teachers to request needed school materials for their students. There are more than thousands of teacher requests that people can donate to.

This dataset originally consisted with six csv files namely Projects.csv, Resources.csv, Schools.csv, Teachers.csv, Donors.csv, Donations.csv which had millions of records. But due to the analytical purposes of this assignment the csv files were converted to the other formats and five files were chosen among those six files. The chosen files were namely Projects.csv, Schools.csv, Teachers.csv, Donors.csv, Donations.csv. The files namely Projects.csv, Schools.csv, Teachers.csv, Donors.csv, Donations.csv were converted as Projects.db , Teachers.db , Schools.db , Donors.db , Donations.db respectively and the donors addresses were converted as DonorsAddress.xls.

This dataset is used for predict donations of each school,teacher,project,donor etc.

a) Schools.csv

Schools.csv contains information about the schools.

b) Teachers.csv

Teachers.csv contains the prefix of teachers who is in charge of the projects.

c) Projects.csv

Projects.csv contains the details about projects along with dates such as the project posted date, Expiration date and fully funded date. It contains foreign keys to schools.csv and teacher.csv

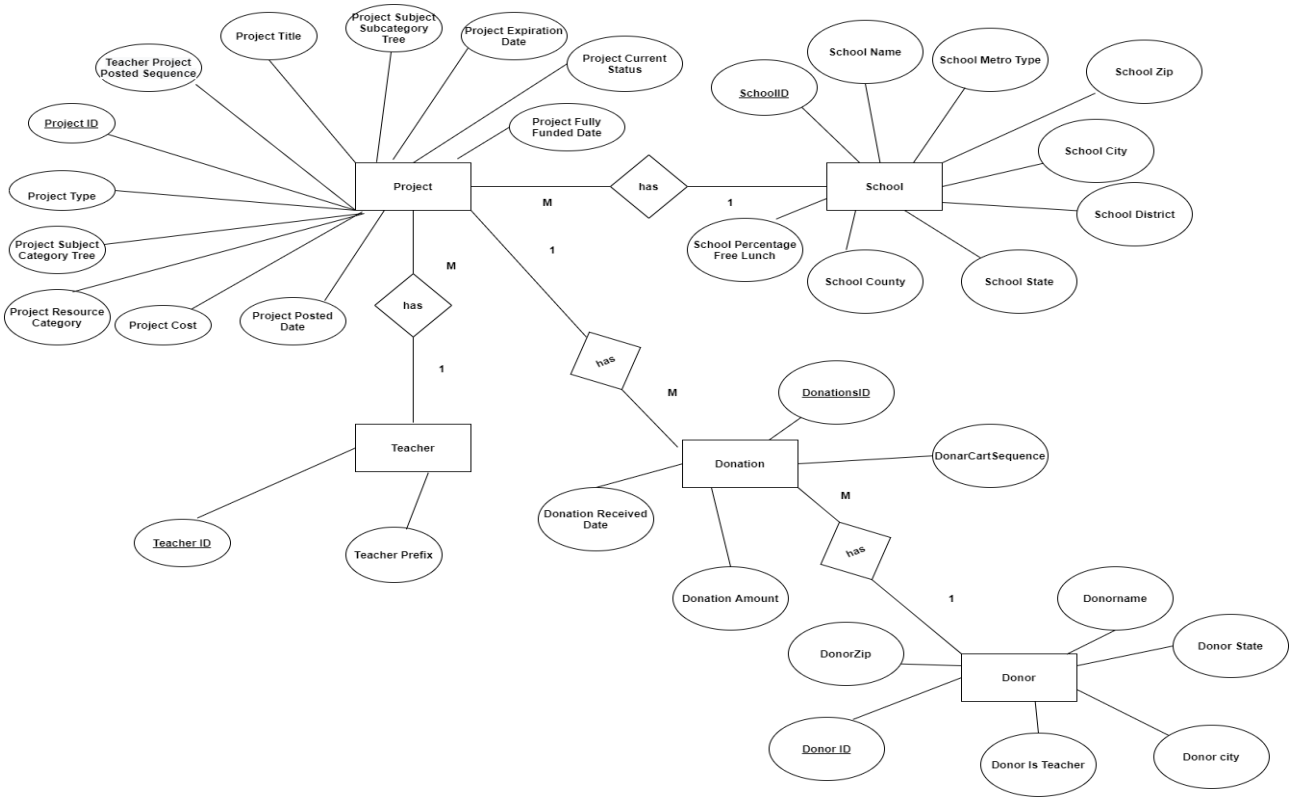
d) Donors.csv

Donors.csv contains information about the donors.

e) Donations.csv

Donations.csv contains information about the donations.

Entity Relationship Diagram



2. Preparation of Data Sources

Original Format	Converted Format
1. Schools.CSV	Schools.db
2. Teachers.CSV	Teachers.db
3. Projects.CSV	Projects.db
4. Donors.CSV	Donors.db
5. Donations.CSV	Donations.db
6. Donors.CSV	DonorsAddress.xls

1. Schools.db
   A) SchoolID –Primary key of a school(Unique).
   B) School Name –Name of the school
   C) School Metro Type – Type of the metro eg:urban,rural,suburban,town,unknown
   D) School Percentage Free Lunch – Percentage of students in the school obtaining free lunch
   E) School State - The state of the school
   F) School Zip - The zip code of the school
   G) School City - The city of the school
   H) School County - The county of the school
   I) School District - The district of the school

   - School ID was generated so that all schools have a unique SchoolID. The School Zip was generated by using excel, so that all the schools have a unique school Zip code. There were more than millions of records in the Schools.csv original datasource but only 2000 out of all were selected and SchoolIDs and School Zip was generated and all the null values were also filled.Then the excel file was imported as a db to ssms.

2. Teachers.db
   A) Teacher ID – Primary key of a teacher.
   B) Teacher Prefix - “Mrs.”, “Ms.”, “Mr.”, “Teacher”

   - Teacher ID was generated using excel so that all teachers have a unique teacher ID.There were million of records in the original datasource and only 3000 were selected and the null values were filled. Then the excel file was imported as a db to ssms.

3. Projects.db
   A) Project ID – Primary key of a Project
   B) School ID - School ID(Foreign key to school db)
   C) Teacher ID – Teacher ID(Foreign key to teacher db)
   D) Teacher Project Posted Sequence – Posted sequence of project
   E) Project Type – Type of the project eg: Teacher-led/Professional
   F) Project Title – Title of the project
   G) Project Subject Category Tree – Subject category of the project
   H) Project Subject Subcategory Tree – Subject subcategory of the project
   I) Project Grade Level Category – Grade of the project
   J) Project Resource Category – Resource of the project
   K) Project Cost – Cost of the project
   L) Project Posted Date – Posted date of the project
   M) Project Expiration Date – Expiration date of the project
   N) Project Current Status – Current status eg: Expired/Fully Funded
   O) Project Fully Funded Date – If fully funded, the date fully funded the project

   - Project ID was generated using excel so that all projects have a unique project ID. There were millions of records in the original datasource and only 11424 records were selected and the null values were filled and generated SchoolIDs and TeacherIDs according to the SchoolIDS and TeacherIDs generated in the above. Then the excel file was imported as a db with relevant foreign keys

4. Donors.db
   A) Donor ID –Primary key of a Donor.
   B) Donor City - City of the Donor.
   C) DonorName-Name of the Donor

   - Donor ID was generated using excel so that all donors have a unique Donor ID. There were millions of records in the original datasource and only 2000 records were selected and the null values were filled. Donor names were not there in the original source. Donor names were selected from another datasource and was filled using excel. Then the excel file was imported as a db to ssms.

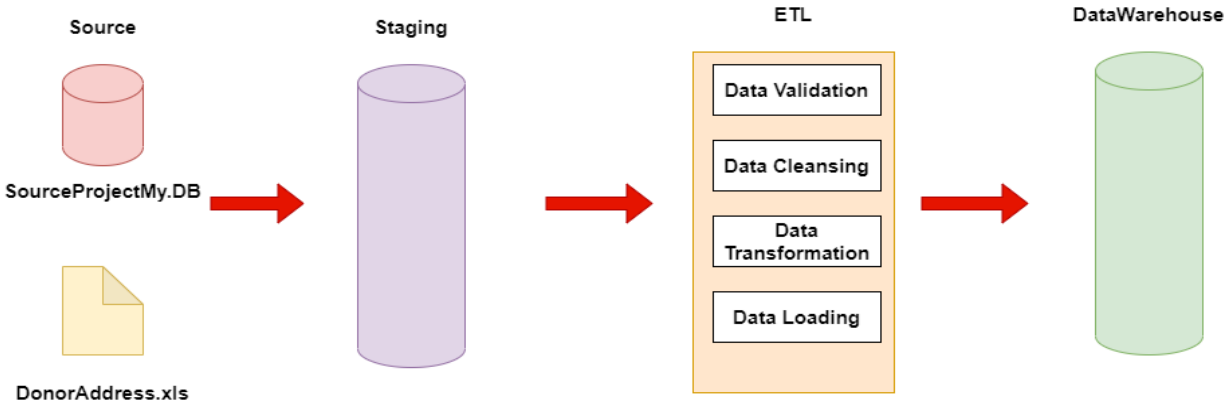
5. DonorsAddress.xls
   A) Donor ID-Primary key of a Donor address
   B) Donor State –State of the Donor.
   C) Donor Is Teacher –Donor is a teacher or not.
   D) Donor Zip – Donors zip code.

- Donor ID was generated using excel so that all donors have a unique Donor ID. There were millions of records in the original datasource and only 2000 records were selected and the null values were filled. The Donor addresses were separated from the Donors.csv and was taken into another excel file. The zip code was generated using excel so that its unique to every donor Accordingly so that a donor has only 1 address.

6. Donations.db
   A) Donation ID – Primary key of a donation
   B) Project ID – ProjectID(Foreign key to Project db)
   C) Donor ID - DonorID(Foreign key to Donor db)
   D) Donation Amount – Amount donated to a project by a donor.
      (Addition of the donations amount given to a single project is the total donation amount for a project)
   E) Donor Cart Sequence – the sequence of the donations given to a single project
   F) Donation Received Date - Date the donation was received

   - Donations ID was generated using excel so that all donations have a unique Donations ID. There were millions of records in the original datasource and only 11075 records were selected and the null values were filled and ProjectID and DonarID was generated using excel from the above created source files. The Doantions excel file was then imported as a db to ssms.

3. Solution Architecture

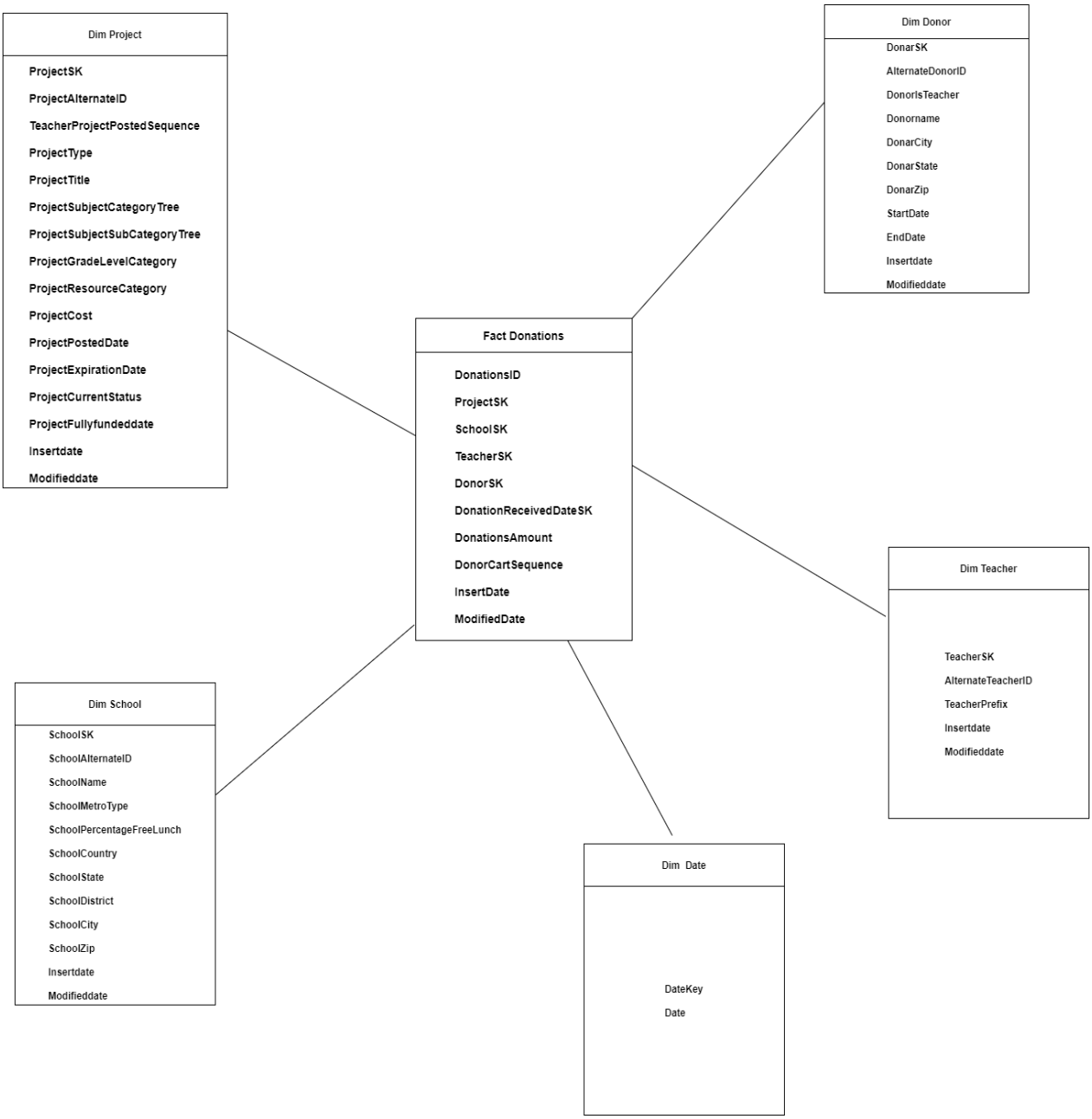


First the source files are loaded into the staging area without any transformations. The following staging tables were created

1. Teacher staging
2. School staging
3. Donor staging
4. DonorAddress staging
5. Project staging
6. Donations staging

Then the transformations are performed, and then data is validated and tested and finally loaded into the datawarehouse. ETL process is carried between the staging and data warehouse layers. After the datwarehouse is created BI results such as OLAP analysis, Reports, Data visualization can be obtained after modifications

4. Data warehouse design and development



Star schema is used to design the data warehouse. In this scenario five dimension tables including DimDate were identified along with one fact table. Details about identified dimension tables and fact tables are mentioned below.

Addresses of the donor were considered as a slowly changing dimension to track the addresses of donor (Type 2).

Fact_Donations:

- DonationID: Unique ID of a donation.
- ProjectSK: Project Dimension SK
- DonorSK: Donor Dimension SK
- TeacherSK: Teacher Dimension SK
- SchoolSK: School Dimension SK
- DonationsAmount: Donations Amout
- DonarCartSequence: Cart sequence
- DonationReceiveddateKey: Date Dimension SK
- InsertDate: Inserted Date
- ModifiedDate: Modified Date

Dim_Teachers:

- TeacherSK: Surrogate key
- TeacherID: TeacherID(Business key)
- TeacherPrefix: Prefix of the teacher
- InsertDate: Insertdate of teacher
- ModifiedDate: Modified date of teacher

Dim_Schools:

- SchoolSK: Surrogate key.
- SchoolID: SchoolID(Business key)
- school_name: Name of the school.
- school_state: State of school
- school_city: The city of the school
- school_metro_type: Metro type of school area ("urban", "suburban", "rural", "town" etc.) .
- school_percentage_free_lunch: Integer describing percentage of students qualifying for free or reduced lunch
- school_zip: Zip code of school
- school county: County where school is situated
- school_district: District where school is situated.
- InsertDate:Insertdate of school
- ModifiedDate:Modified date of school

Dim_Donors:

- DonorSK: Surrogate key
- DonorID:DonorID(Business key)
- DonorName:Name of the Donor
- DonorCity: The donor's city.
- DonorState: The donor's state.
- DonorZip:Zip code of donor
- Donor_Is_Teacher: Whether or not the donor is a teacher.
- StartDate:StartedDate
- EndDate:End Date
- InsertDate:Inserted date
- ModifiedDate:Modified Date

Dim_Project:

- ProjectSK: Surrogate key
- ProjectID:ProjectID(Business key)
- Teacher Project Posted Sequence – Posted sequence of project
- Project Type – Type of the project eg: Teacher-led/Professional
- Project Title – Title of the project
- Project Subject Category Tree – Subject category of the project
- Project Subject Subcategory Tree – Subject subcategory of the project
- Project Grade Level Category – Grade of the project
- Project Resource Category – Resource of the project
- Project Cost – Cost of the project
- Project Posted Date – Posted date of the project
- Project Expiration Date – Expiration date of the project
- Project Current Status – Current status eg: Expired/Fully Funded
- Project Fully Funded Date – If fully funded, the date fully funded the project
- InsertDate:Inserted date
- ModifiedDate:Modified Date

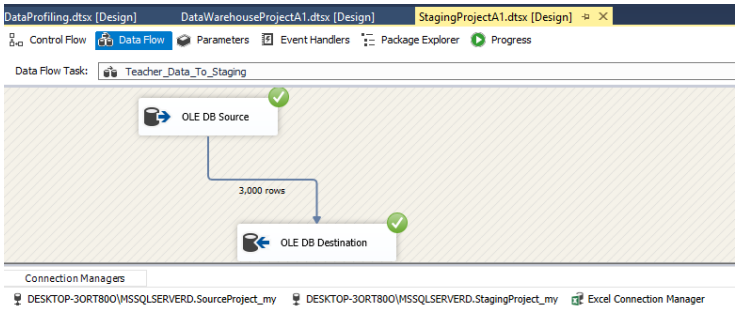
Date dimension was also used to implement this data warehouse.

5. ETL Development

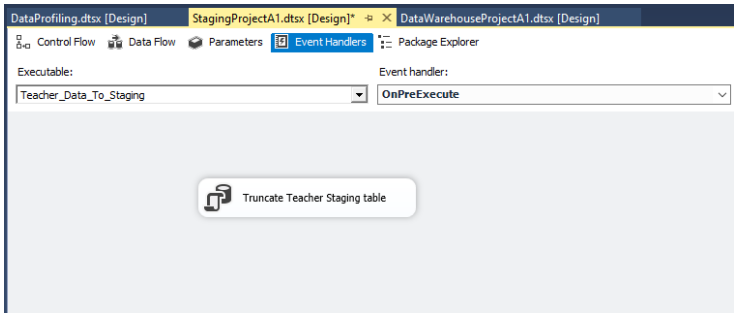
- As the first step data was extracted from the source(DB source and the excel file). For every extraction data flow task was used and was extracted from source to staging. For all the dataflow tasks an event handler was created to truncate the staging table when loading data.
  The screenshots are attached below

Transformations of the successfully Executed dataset

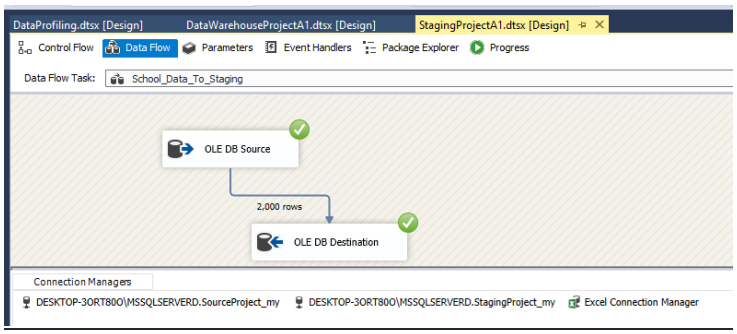
Loading data from source to teacher staging



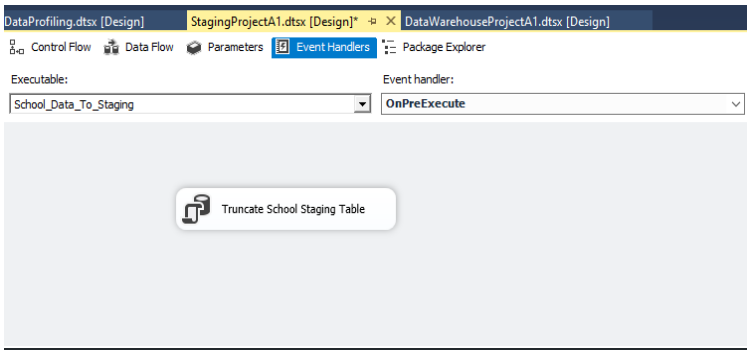
When loading data monthly to prevent saving duplicate values the following Event handler was used.



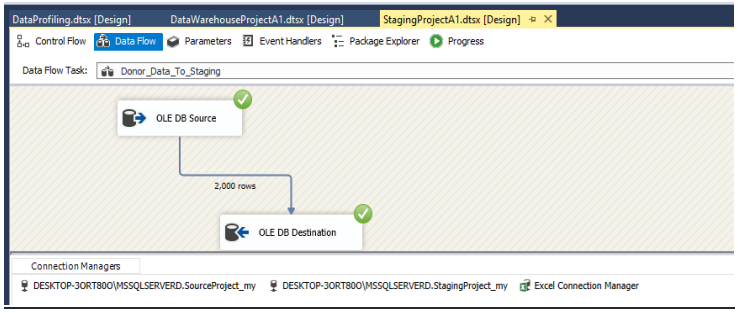
Loading data from source to school staging



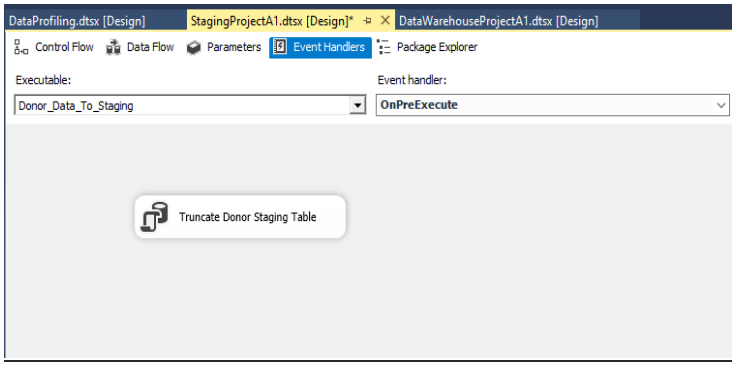
When loading data monthly to prevent saving duplicate values the following Event handler was used.



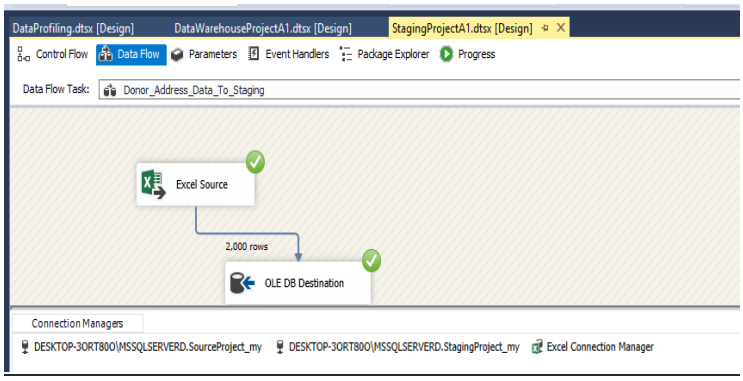
Loading data from source to Donor staging



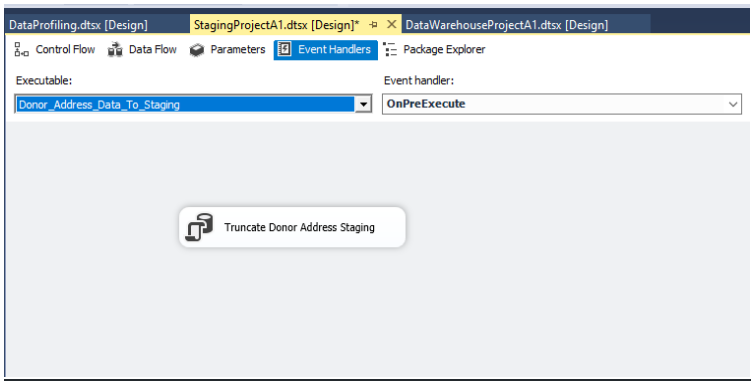
When loading data monthly to prevent saving duplicate values the following Event handler was used.



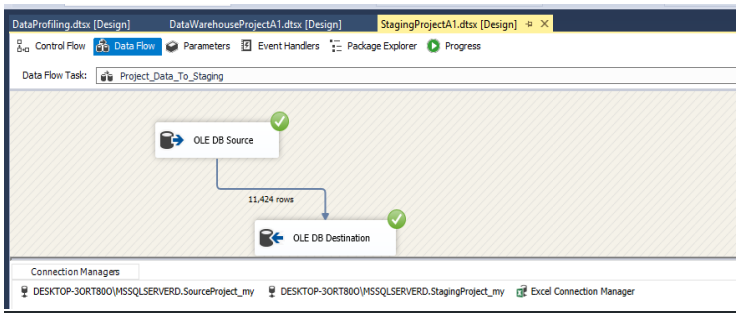
Loading data from source to DonorAddress staging



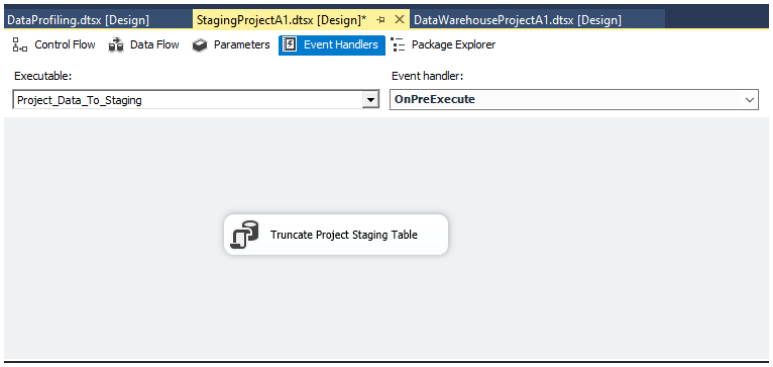
When loading data monthly to prevent saving duplicate values the following Event handler was used.



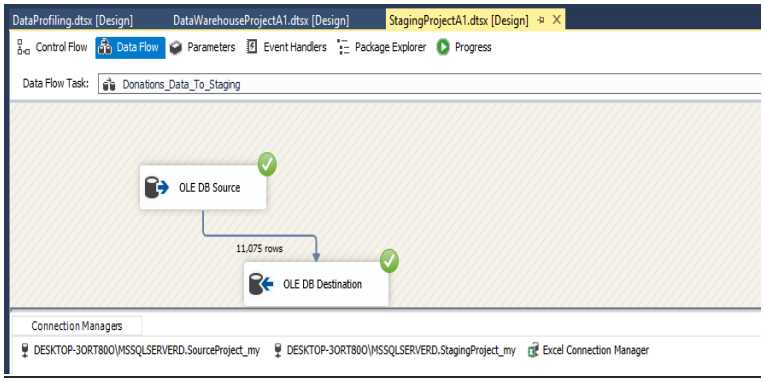
Loading data from source to project staging



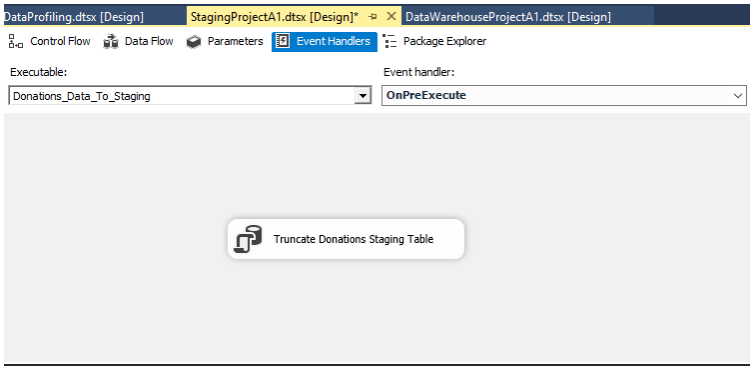
When loading data monthly to prevent saving duplicate values the following Event handler was used.



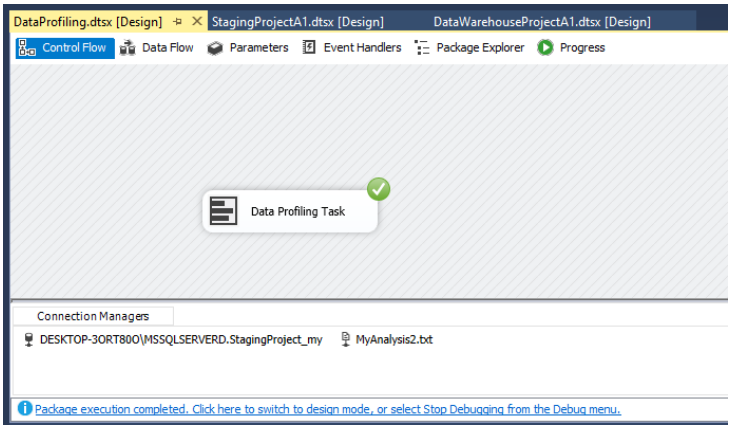
Loading data from source to DonationsStaging



When loading data monthly to prevent saving duplicate values the following Event handler was used.

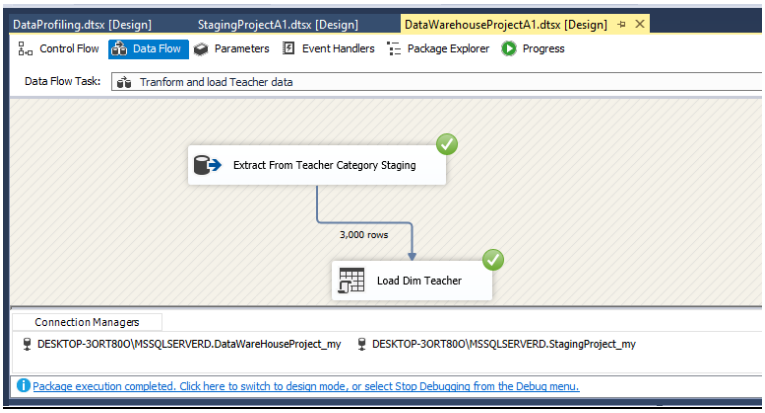


- Then the next step is data profiling, it was done as shown below and the profiled file was saved in a selected location



- As the next step the staging tables were loaded to the datawarehouse. Followed by a stored procedure for every task. The screenshots are attached below.

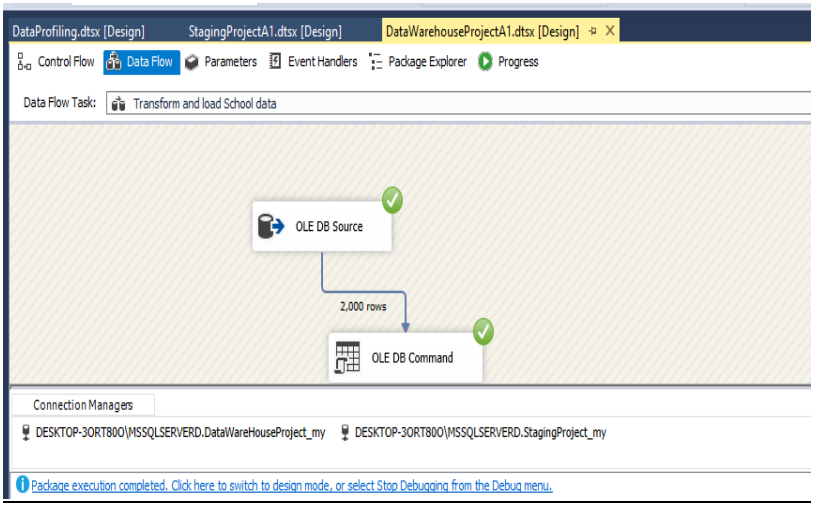
Loading data from staging table to DimTeacher



Update procedure used to update DimTeacher

```
CREATE PROCEDURE dbo.UpdateDimTeacher
@TeacherID nvarchar(255),
@TeacherPrefix nvarchar(255)
AS
BEGIN
if not exists (select TeacherSK
from dbo.DimTeacher
where AlternateTeacherID = @TeacherID)
BEGIN
insert into dbo.DimTeacher (
AlternateTeacherID, TeacherPrefix, insertdate, modifieddate)
values
(@TeacherID, @TeacherPrefix, GETDATE(), GETDATE())
END;
if exists (select TeacherSK
from dbo.DimTeacher
where AlternateTeacherID = @TeacherID)
BEGIN
update dbo.DimTeacher
set TeacherPrefix = @TeacherPrefix,
modifieddate = GETDATE()
where AlternateTeacherID = @TeacherID
END;
END;
```

Loading data from staging table to DimSchool

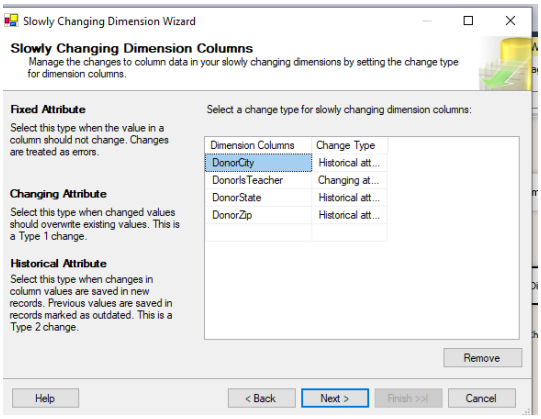


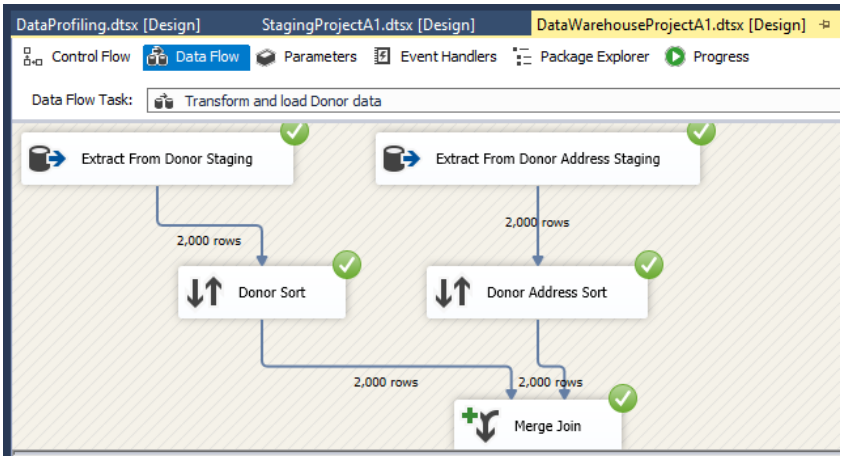
Update procedure used to update DimSchool

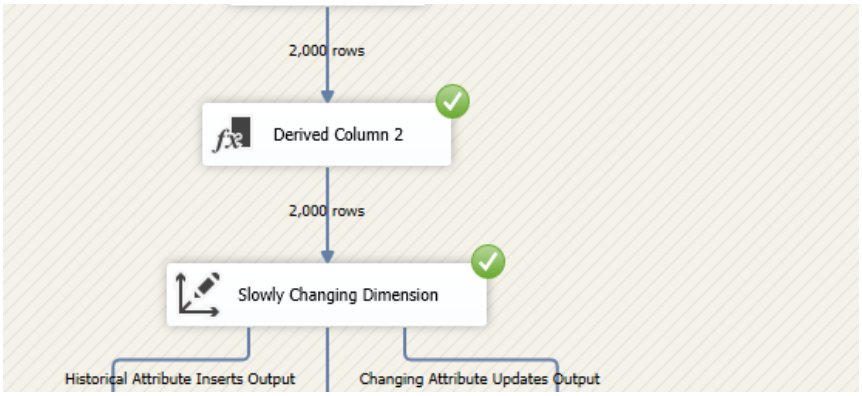
```
CREATE PROCEDURE dbo.UpdateDimSchool
@SchoolID nvarchar(255),
@SchoolName nvarchar(255),
@SchoolMetroType nvarchar(255),
@PercentageFree float,
@SchoolCountry nvarchar(255),
@SchoolState nvarchar(255),
@SchoolDistrict nvarchar(255),
@SchoolCity nvarchar(255),
@SchoolZip nvarchar(255)
AS
BEGIN
if not exists (select SchoolSK
from dbo.DimSchool
where SchoolAlternateID = @SchoolID)
BEGIN
insert into dbo.DimSchool (
SchoolAlternateID, SchoolName,SchoolMetroType,SchoolPercentageFreeLunch,SchoolCountry,SchoolState,SchoolDistrict,SchoolCity,SchoolZip, Insertdate, Modifieddate)
values
(@SchoolID, @SchoolName,@SchoolMetroType,@PercentageFree,@SchoolCountry,@SchoolState,@SchoolDistrict,@SchoolCity,@SchoolZip, GETDATE(), GETDATE())
END;
if exists (select SchoolSK
from dbo.DimSchool
where SchoolAlternateID = @SchoolID)
BEGIN
update dbo.DimSchool
set SchoolName=@SchoolName,SchoolMetroType=@SchoolMetroType,SchoolPercentageFreeLunch=@PercentageFree,SchoolCountry=@SchoolCountry,SchoolState=@SchoolState,SchoolDistrict=@SchoolDistrict,SchoolCity=@SchoolCity,SchoolZip=@SchoolZip,
Modifieddate = GETDATE()
where SchoolAlternateID = @SchoolID
END;
END;
```

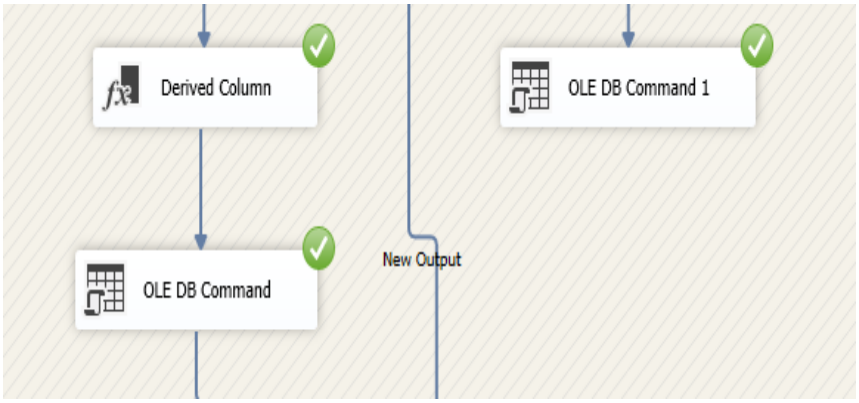
Loading data from staging table to DimDonor

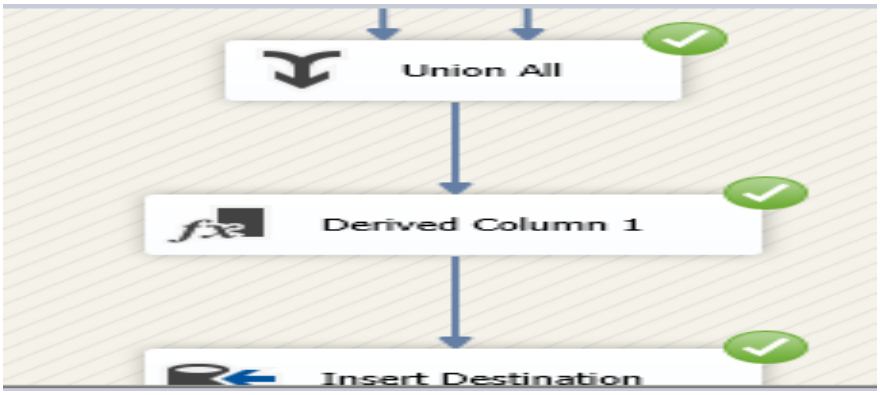
- After extracting data from donor staging table and donor address staging table they were sorted according to the donor id and merged using merge join. Donor city ,Donor state, Donor Zip was identified as slowly changing attributes.
  Screenshots are attached below.


Slowly Changing Dimension Wizard
Slowly Changing Dimension Columns
Manage the changes to column data in your slowly changing dimensions by setting the change type for dimension columns.
Fixed Attribute
Select this type when the value in a column should not change. Changes are treated as errors.
Changing Attribute
Select this type when changed values should overwrite existing values. This is a Type 1 change.
Historical Attribute
Select this type when changes in column values are saved in new records. Previous values are saved in records marked as outdated. This is a Type 2 change.
Select a change type for slowly changing dimension columns:
Dimension Columns
Change Type
DonorCity
Historical att...
DonorIsTeacher
Changing at...
DonorState
Historical att...
DonorZip
Historical att...
Remove
Help
< Back
Next >
Finish >>|
Cancel

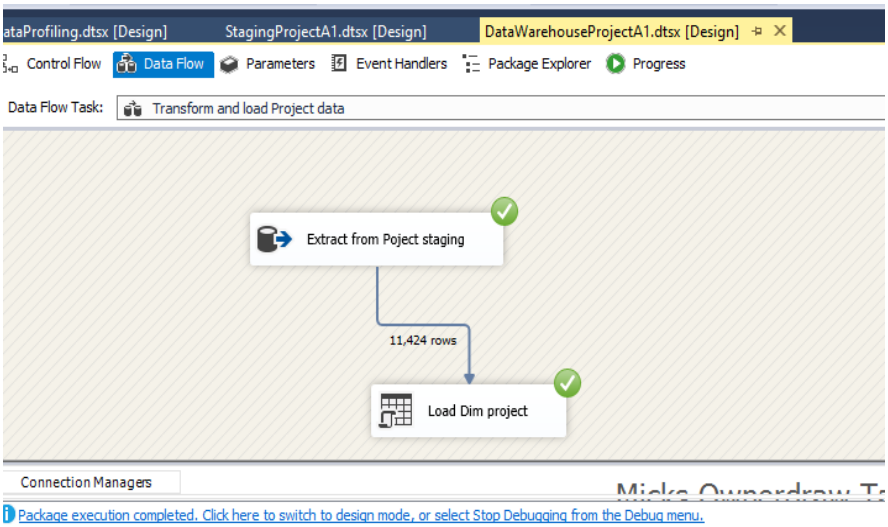

DataProfiling.dtsx [Design]
StagingProjectA1.dtsx [Design]
DataWarehouseProjectA1.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Transform and load Donor data
Extract From Donor Staging
Extract From Donor Address Staging
2,000 rows
2,000 rows
Donor Sort
Donor Address Sort
2,000 rows
2,000 rows
Merge Join


2,000 rows
Derived Column 2
2,000 rows
Slowly Changing Dimension
Historical Attribute Inserts Output
Changing Attribute Updates Output


Derived Column
OLE DB Command 1
OLE DB Command
New Output



Loading data from staging table to DimProject



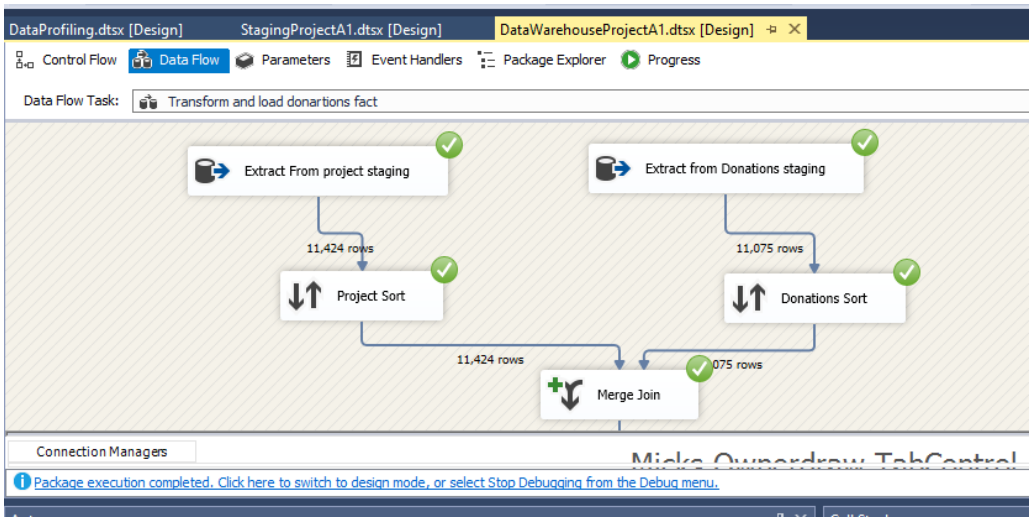
Update procedure used to update DimProject

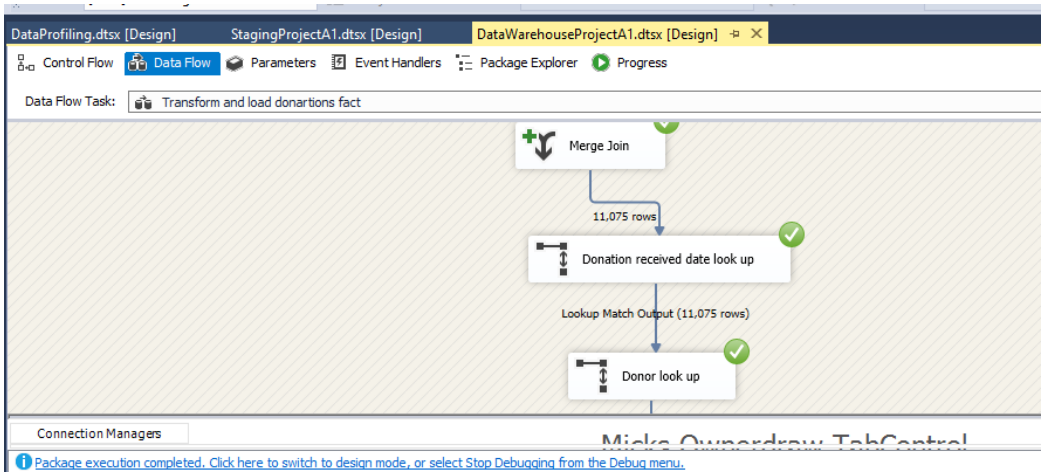
```
CREATE PROCEDURE dbo.UpdateDimProject
@ProjectAlternateID nvarchar(255),
@TeacherProjectPostedSequence nvarchar(255),
@ProjectType nvarchar(255),
@ProjectTitle nvarchar(255),
@ProjectSubjectCategoryTree nvarchar(255),
@ProjectSubjectSubCategoryTree nvarchar(255),
@ProjectGradeLevelCategory nvarchar(255),
@ProjectResourceCategory nvarchar(255),
@ProjectCost float,
@ProjectPostedDate datetime,
@ProjectExpirationDate datetime,
@ProjectCurrentStatus nvarchar(255),
@ProjectFullyfundeddate datetime
 AS BEGIN
 if not exists (select ProjectSK
                           from dbo.DimProject
                           where ProjectAlternateID = @ProjectAlternateID)
BEGIN insert into dbo.DimProject (ProjectAlternateID, TeacherProjectPostedSequence, ProjectType, ProjectTitle, ProjectSubjectCategoryTree,ProjectSubjectSubCategoryTree,ProjectGradeLevelCategory,ProjectResourceCategory,ProjectCost,ProjectPostedDate,ProjectExpirationDat

if exists (select ProjectSK
                      from dbo.DimProject
                      where ProjectAlternateID = @ProjectAlternateID)
BEGIN update dbo.DimProject
set ProjectAlternateID=@ProjectAlternateID, TeacherProjectPostedSequence=@TeacherProjectPostedSequence, ProjectType=@ProjectType, ProjectTitle=@ProjectTitle, ProjectSubjectCategoryTree=@ProjectSubjectCategoryTree,ProjectSubjectSubCategoryTree=@ProjectSubjectSubCategor
where ProjectAlternateID  = @ProjectAlternateID
END;
END;
```

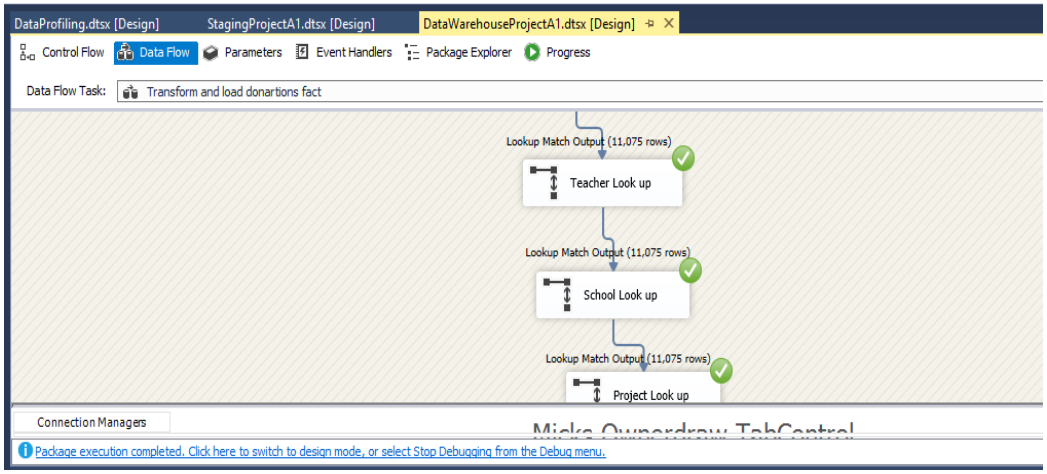
Loading data to fact Donations

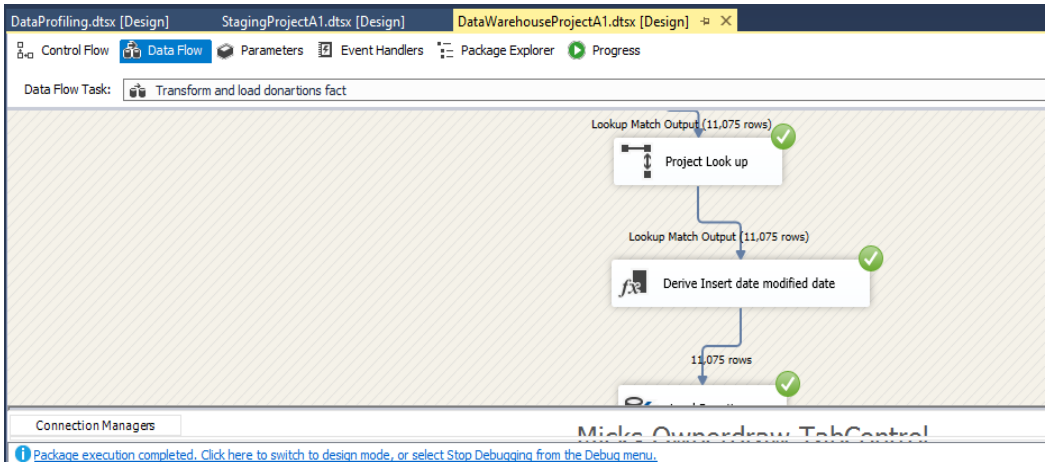
- The project staging table and donations staging table was combined together using ProjectID to create the fact table
- In order to get the teachersk and schoolsk project staging table was merged.
- Donationreceived date SK was set up by a lookup
- TeacherSK was set up by a lookup
- DonarSK was set up by a lookup
- SchoolSK was set up by a lookup
- ProjectSK was set up by a lookup

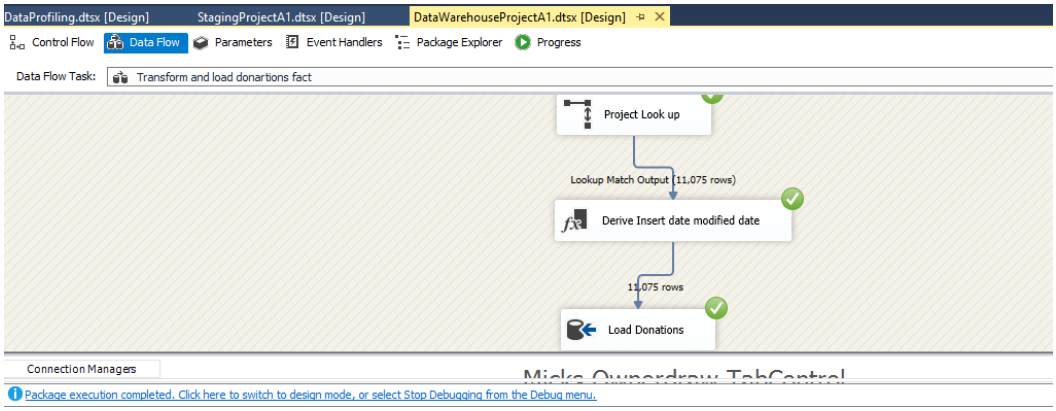
Screenshots are attached below




DataProfiling.dtsx [Design]
StagingProjectA1.dtsx [Design]
DataWarehouseProjectA1.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Transform and load donartions fact
Merge Join
11,075 rows
Donation received date look up
Lookup Match Output (11,075 rows)
Donor look up
Connection Managers
Package execution completed. Click here to switch to design mode, or select Stop Debugging from the Debug menu.


DataProfiling.dtsx [Design]
StagingProjectA1.dtsx [Design]
DataWarehouseProjectA1.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Transform and load donartions fact
Lookup Match Output (11,075 rows)
Teacher Look up
Lookup Match Output (11,075 rows)
School Look up
Lookup Match Output (11,075 rows)
Project Look up
Connection Managers
Package execution completed. Click here to switch to design mode, or select Stop Debugging from the Debug menu.


DataProfiling.dtsx [Design]
StagingProjectA1.dtsx [Design]
DataWarehouseProjectA1.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explorer
Progress
Data Flow Task:
Transform and load donartions fact
Lookup Match Output (11,075 rows)
Project Look up
Lookup Match Output (11,075 rows)
Derive Insert date modified date
11,075 rows
Connection Managers
Package execution completed. Click here to switch to design mode, or select Stop Debugging from the Debug menu.



Query used to create the date dimension is attached below

```
CREATE TABLE    [dbo].[DimDate]
        (       [DateKey] INT primary key,
                [Date] DATETIME,
                [FullDateUK] CHAR(10), -- Date in dd-MM-yyyy format
                [FullDateUSA] CHAR(10),-- Date in MM-dd-yyyy format
                [DayOfMonth] VARCHAR(2), -- Field will hold day number of Month
                [DaySuffix] VARCHAR(4), -- Apply suffix as 1st, 2nd ,3rd etc
                [DayName] VARCHAR(9), -- Contains name of the day, Sunday, Monday
                [DayOfWeekUSA] CHAR(1),-- First Day Sunday=1 and Saturday=7
                [DayOfWeekUK] CHAR(1),-- First Day Monday=1 and Sunday=7
                [DayOfWeekInMonth] VARCHAR(2), --1st Monday or 2nd Monday in Month
                [DayOfWeekInYear] VARCHAR(2),
                [DayOfQuarter] VARCHAR(3),
                [DayOfYear] VARCHAR(3),
                [WeekOfMonth] VARCHAR(1),-- Week Number of Month
                [WeekOfQuarter] VARCHAR(2), --Week Number of the Quarter
                [WeekOfYear] VARCHAR(2),--Week Number of the Year
                [Month] VARCHAR(2), --Number of the Month 1 to 12
                [MonthName] VARCHAR(9),--January, February etc
                [MonthOfQuarter] VARCHAR(2),-- Month Number belongs to Quarter
                [Quarter] CHAR(1),
                [QuarterName] VARCHAR(9),--First,Second..
                [Year] CHAR(4),-- Year value of Date stored in Row
                [YearName] CHAR(7), --CY 2012,CY 2013
                [MonthYear] CHAR(10), --Jan-2013,Feb-2013
                [MMYYYY] CHAR(6),
                [FirstDayOfMonth] DATE,
                [LastDayOfMonth] DATE,
                [FirstDayOfQuarter] DATE,
                [LastDayOfQuarter] DATE,
                [FirstDayOfYear] DATE,
                [LastDayOfYear] DATE,
                [IsHolidaySL] BIT,-- Flag 1=National Holiday, 0-No National Holiday
                [IsWeekday] BIT,-- 0=Week End ,1=Week Day
                [HolidaySL] VARCHAR(50),--Name of Holiday in US
                [isCurrentDay] int, -- Current day=1 else = 0

/*Table Data type to store the day of week count for the month and year*/
DECLARE @DayOfWeek TABLE (DOW INT, MonthCount INT, QuarterCount INT, YearCount INT)

INSERT INTO @DayOfWeek VALUES (1, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (2, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (3, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (4, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (5, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (6, 0, 0, 0)
INSERT INTO @DayOfWeek VALUES (7, 0, 0, 0)

--Extract and assign various parts of Values from Current Date to Variable

DECLARE @CurrentDate AS DATETIME = @StartDate
SET @CurrentMonth = DATEPART(MM, @CurrentDate)
SET @CurrentYear = DATEPART(YY, @CurrentDate)
SET @CurrentQuarter = DATEPART(QQ, @CurrentDate)

/********************************************************************************************/
--Proceed only if Start Date(Current date ) is less than End date you specified above

WHILE @CurrentDate < @EndDate
BEGIN

/*Begin day of week logic*/

         /*Check for Change in Month of the Current date if Month changed then
          Change variable value*/
        IF @CurrentMonth != DATEPART(MM, @CurrentDate)
        BEGIN
                UPDATE @DayOfWeek
                SET MonthCount = 0
                SET @CurrentMonth = DATEPART(MM, @CurrentDate)
        END

        /* Check for Change in Quarter of the Current date if Quarter changed then change
         Variable value*/

        IF @CurrentQuarter != DATEPART(QQ, @CurrentDate)
        BEGIN
                UPDATE @DayOfWeek
                SET QuarterCount = 0
                SET @CurrentQuarter = DATEPART(QQ, @CurrentDate)
        END

        /* Check for Change in Year of the Current date if Year changed then change
         Variable value*/


        IF @CurrentYear != DATEPART(YY, @CurrentDate)
        BEGIN
                UPDATE @DayOfWeek
                SET YearCount = 0
                SET @CurrentYear = DATEPART(YY, @CurrentDate)
        END

        -- Set values in table data type created above from variables

        UPDATE @DayOfWeek
        SET
                MonthCount = MonthCount + 1,
                QuarterCount = QuarterCount + 1,
                YearCount = YearCount + 1
        WHERE DOW = DATEPART(DW, @CurrentDate)

        SELECT
                @DayOfWeekInMonth = MonthCount,
                @DayOfQuarter = QuarterCount,
                @DayOfWeekInYear = YearCount
        FROM @DayOfWeek
        WHERE DOW = DATEPART(DW, @CurrentDate)

/*End day of week logic*/


/* Populate Your Dimension Table with values*/

        INSERT INTO [dbo].[DimDate]
        SELECT

                CONVERT (char(8),@CurrentDate,112) as DateKey,
                @CurrentDate AS Date,
                CONVERT (char(10),@CurrentDate,103) as FullDateUK,
                CONVERT (char(10),@CurrentDate,101) as FullDateUSA,
                DATEPART(DD, @CurrentDate) AS DayOfMonth,
                --Apply Suffix values like 1st, 2nd 3rd etc..
                CASE
                        WHEN DATEPART(DD,@CurrentDate) IN (11,12,13)
                        THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'th'
                        WHEN RIGHT(DATEPART(DD,@CurrentDate),1) = 1
                        THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'st'
                        WHEN RIGHT(DATEPART(DD,@CurrentDate),1) = 2
                        THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'nd'
                        WHEN RIGHT(DATEPART(DD,@CurrentDate),1) = 3
                        THEN CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'rd'
                        ELSE CAST(DATEPART(DD,@CurrentDate) AS VARCHAR) + 'th'
                        END AS DaySuffix,

                DATENAME(DW, @CurrentDate) AS DayName,

                @DayOfWeekInMonth AS DayOfWeekInMonth,
                @DayOfWeekInYear AS DayOfWeekInYear,
                @DayOfQuarter AS DayOfQuarter,
                DATEPART(DY, @CurrentDate) AS DayOfYear,
                DATEPART(WW, @CurrentDate) + 1 - DATEPART(WW, CONVERT(VARCHAR,
                DATEPART(MM, @CurrentDate)) + '/1/' + CONVERT(VARCHAR,
                DATEPART(YY, @CurrentDate))) AS WeekOfMonth,
                (DATEDIFF(DD, DATEADD(QQ, DATEDIFF(QQ, 0, @CurrentDate), 0),
                @CurrentDate) / 7) + 1 AS WeekOfQuarter,
                DATEPART(WW, @CurrentDate) AS WeekOfYear,
                DATEPART(MM, @CurrentDate) AS Month,
                DATENAME(MM, @CurrentDate) AS MonthName,
                CASE
                        WHEN DATEPART(MM, @CurrentDate) IN (1, 4, 7, 10) THEN 1
                        WHEN DATEPART(MM, @CurrentDate) IN (2, 5, 8, 11) THEN 2
                        WHEN DATEPART(MM, @CurrentDate) IN (3, 6, 9, 12) THEN 3
                        END AS MonthOfQuarter,
                DATEPART(QQ, @CurrentDate) AS Quarter,
                CASE DATEPART(QQ, @CurrentDate)
                        WHEN 1 THEN 'First'
                        WHEN 2 THEN 'Second'
                        WHEN 3 THEN 'Third'
                        WHEN 4 THEN 'Fourth'
                        END AS QuarterName,
                DATEPART(YEAR, @CurrentDate) AS Year,
                'CY ' + CONVERT(VARCHAR, DATEPART(YEAR, @CurrentDate)) AS YearName,
                LEFT(DATENAME(MM, @CurrentDate), 3) + '-' + CONVERT(VARCHAR,
                DATEPART(YY, @CurrentDate)) AS MonthYear,
                RIGHT('0' + CONVERT(VARCHAR, DATEPART(MM, @CurrentDate)),2) +
                CONVERT(VARCHAR, DATEPART(YY, @CurrentDate)) AS MMYYYY,
                CONVERT(DATETIME, CONVERT(DATE, DATEADD(DD, - (DATEPART(DD,
                @CurrentDate) - 1), @CurrentDate))) AS FirstDayOfMonth,
                CONVERT(DATETIME, CONVERT(DATE, DATEADD(DD, - (DATEPART(DD,
                (DATEADD(MM, 1, @CurrentDate)))), DATEADD(MM, 1,
                @CurrentDate)))) AS LastDayOfMonth,
                DATEADD(QQ, DATEDIFF(QQ, 0, @CurrentDate), 0) AS FirstDayOfQuarter,
                DATEADD(QQ, DATEDIFF(QQ, -1, @CurrentDate), -1) AS LastDayOfQuarter,
                CONVERT(DATETIME, '01/01/' + CONVERT(VARCHAR, DATEPART(YY,
                @CurrentDate))) AS FirstDayOfYear,
                CONVERT(DATETIME, '12/31/' + CONVERT(VARCHAR, DATEPART(YY,
                @CurrentDate))) AS LastDayOfYear,
                NULL AS IsHolidaySL,
                CASE DATEPART(DW, @CurrentDate)
                        WHEN 1 THEN 0
                        WHEN 2 THEN 1
                        WHEN 3 THEN 1
                        WHEN 4 THEN 1
                        WHEN 5 THEN 1
                        WHEN 6 THEN 1
                        WHEN 7 THEN 0
                        END AS IsWeekday,
                NULL AS HolidaySL, (case when @CurrentDate = convert(date, sysdatetime()) then 1 else 0 end), 0, 0

        SET @CurrentDate = DATEADD(DD, 1, @CurrentDate)
END
```

6. Test Planning and Test Data

Testing is done to ensure that the data that has been successfully loaded from source to destination after the business transformation.
Testing was done in 2 stages as mentioned below

1) Source to Staging
2) Staging to Datawarehouse

Test Plan

Scope

1. Completeness of the data set testing
   To conduct test cases to ensure that there are no data losses between the ETL processes and data is completely loaded

2. Data length testing
   To conduct test cases to ensure that the data lengths are equal after the ETL processes

3. Data type testing
   Data types are tested to ensure that the process is running properly

4. Data Duplicity Testing
   To conduct test cases to ensure that no data is being duplicated at the end of the process

Assumptions

There is no environment downtime during the process

Test Delivarables

- Test cases
- Test plans and Test Results

Test Environment

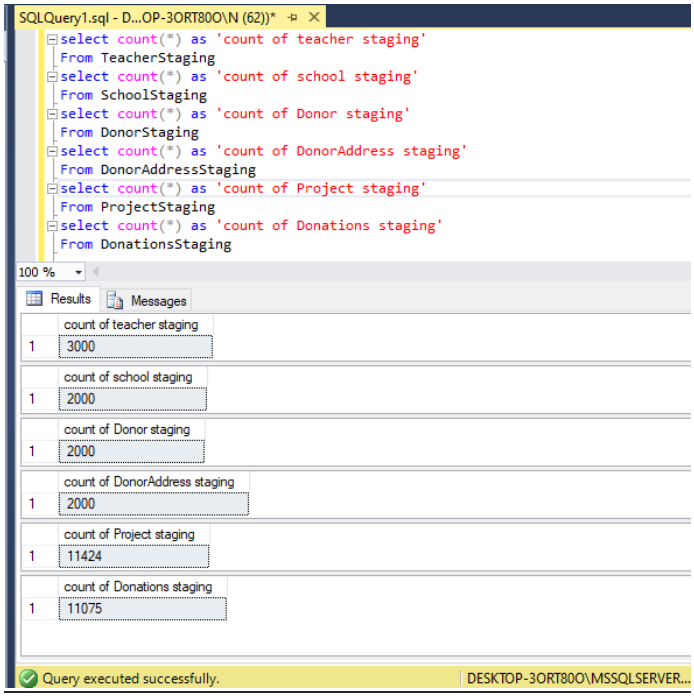
- Microsoft SQL server management studio

Test Tools

- Microsoft SQL server data tools

1)Testing data loaded from Source to Staging

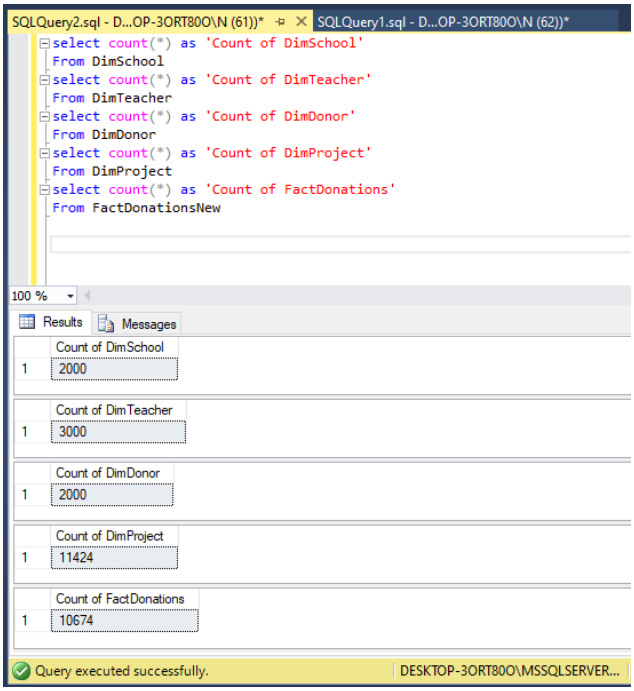
TestCaseID	Action	SQL Query	Expected Output	Actual Output	Test Result	Description
1	Test data passed to Teacher Staging	Select count(*) From TeacherStaging	3000	3000	Pass	Refer 1.1 Attachment
2	Test data passed to School Staging	Select count(*) From SchoolStaging	2000	2000	Pass	Refer 1.1 Attachment
3	Test data passed to Donor Staging	Select count(*) From DonorStaging	2000	2000	Pass	Refer 1.1 Attachment
4	Test data passed to Donoraddres s Staging	Select count(*) From DonorAddressSta ging	2000	2000	Pass	Refer 1.1 Attachment
5	Test data passed to Project Staging	Select count(*) From ProjectStaging	11424	11424	Pass	Refer 1.1 Attachment
6	Test data passed to Donations Staging	Select count(*) From DonationsStaging	11075	11075	Pass	Refer 1.1 Attachment



Attachment 1.1

2)Testing data loaded from Staging to Datawarehouse

TestCaseID	Action	SQL Query	Expected Output	Actual Output	Test Result	Description
1	Test data passed to DimTeacher	Select count(*) From DimTeacher	3000	3000	Pass	Refer 1.2 Attachment
2	Test data passed to DimSchool	Select count(*) From DimSchool	2000	2000	Pass	Refer 1.2 Attachment
3	Test data passed to DimDonor	Select count(*) From DimDonor	2000	2000	Pass	Refer 1.2 Attachment
4	Test data passed to DimProject	Select count(*) From DimProject	11424	11424	Pass	Refer 1.2 Attachment
5	Test data passed to FactDonations	Select count(*) From FactDonations	11075	10674	Fail	Refer 1.2 Attachment



Attachment 1.2