

Sri Lanka Institute of Information Technology



IT3021- Data Warehousing and Business Intelligence

Assignment 01

Student Registration No: IT20188054

Student Name: Benthota Arachchi B.A.T.P.

Batch: Y3.S1.WE.DS.04

Table of Contents

Step 1: Data Set Selection	3
Step 2: Preparation of Data Sources	4
Step 3: Solution Architecture	7
Data Sources	7
Staging Area	8
Data Warehouse	8
Step 4: Data Warehouse Design & Development	9
Step 5: ETL development	12
5.1. ETL Process to Staging Database	12
5.1. Overall Control Flow of Extraction	14
5.1.1.1 Consumer data source to stgConsumer staging	14
5.1.2.1 Consumer address data source to stgConsumerAddress staging	15
5.1.3.1 Consumer type data source to stgConsumerType staging	16
5.1.4.1 Consumption unit price data source to stgConsumptionUnit staging	17
5.1.5.1 Install service data source to stgInstallService staging	18
5.1.6.1 Power consumption data sources to stgPowerConsumption staging	19
5.1.7.1 Power distribution data sources to stgPowerDistribution staging	20
5.1.8.1 Power generator data sources to stgPowerGeneratorInfo staging	21
5.1.9.1 Power Supplier data sources to stgPowerSupplierCompany staging	22
5.2 ETL Process to Data Warehouse	24
5.2 Overall Control Flow of Extraction	26
5.2.1 Transform and load power supplier data to DimPowerSupplier	27
5.2.2 Transform and load power plant data to DimPowerPlant	28
5.2.3 Transform and load power distribution data to DimPowerDistribution	29
5.2.4 Transform and load power unit data to DimPowerUnit	31
5.2.5 Transform and load consumer data to DimConsumer	32
5.2.6 Transform and load data to fact table.	33
5.2.7 Creation of Date Dimension	34
Step 6: ETL development – Accumulating fact tables	38

Step 1: Data Set Selection

Data Set: Energy consumption and management of the Netherlands,

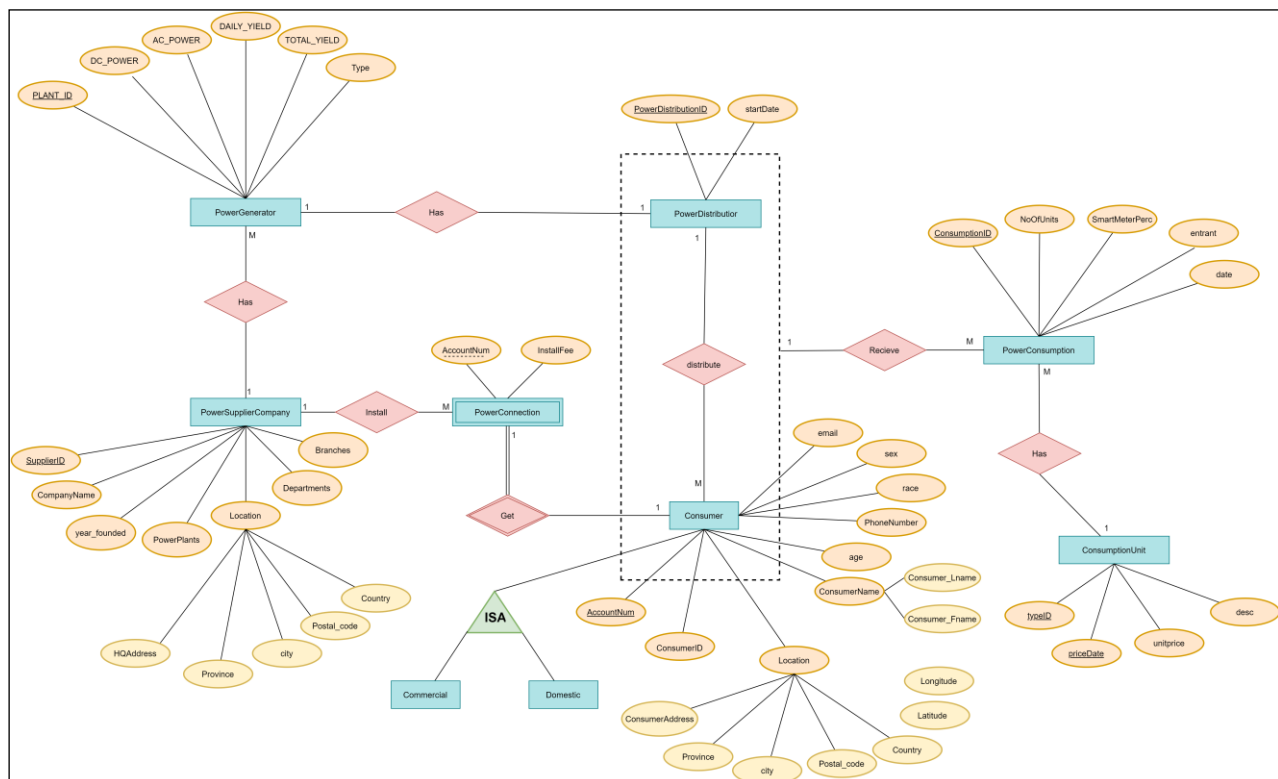
This data set contains the Electricity Management dataset of the Netherlands. This power generation is done by private companies in the Netherlands by distributed to the consumer in several cities whose are consume electricity for both commercial and domestic purposes. For those who get the electricity connection companies will charge a fee for the installation of the meter and other appliances.

These datasets consist of ~50,000 of power consumption data of consumers for the years 2016 and 2017 and power generation, power distribution, and power plants data of private companies, and all the other necessary data needed for electricity management.

This dataset contains,

- Power Suppliers Details
- Power Plants Details
- Power Distribution Details
- Power Consumption Details
- Unit price of the electricity throughout the year
- Connection Install payments Details
- Consumer Details
- Consumer Addresses

The following ER- diagram will describe the scenario of the selected dataset.



Hierarchical Data

- Power Generation data – Power Supplier → Power Plants → Power Distribution
- Consumer data – country → province → city → ZIP code → address

Step 2: Preparation of Data Sources

Database, Text, Excel, and csv were used as the data sources,

1. Database(.bak)

- *Electricity_Consumer.bak*
 - ConsumerDetails
 - ConsumerType
- *Electricity_PowerSuppliers.bak*
 - SupplierCompany
 - InstallServiceInfo

2. Text(.txt)

- ConsumerAddress.txt

3. Excel 97-2003 Worksheet(.xls)










- PowerDistributionInfo.xls
- PowerConsumption 2017.xls

4. Excel Woksheet(.xlsx)

- ConsumptionUnit.xlsx

5. Comma Separated Values (.csv)

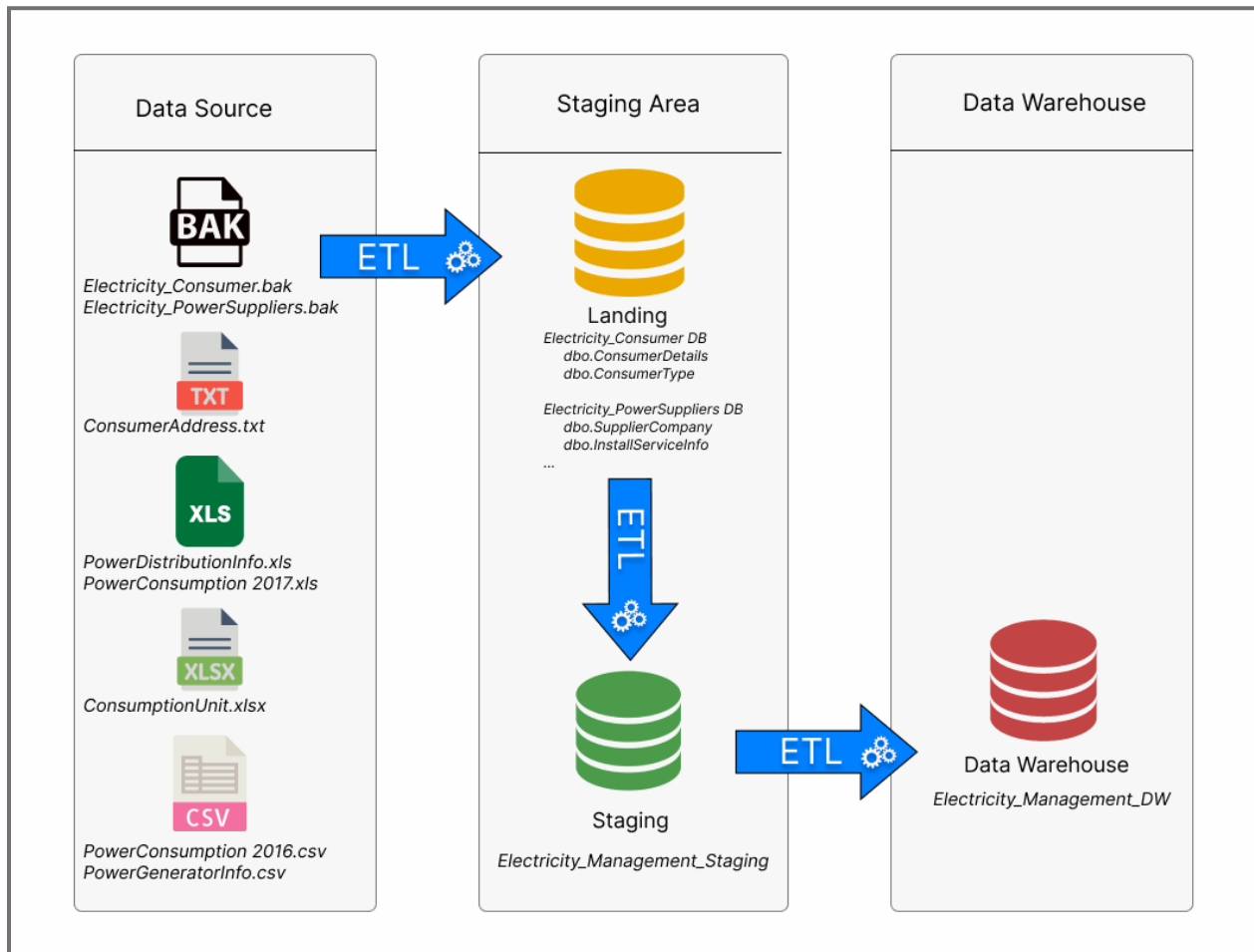
- PowerConsumption 2016.csv
- PowerGeneratorInfo.csv

Name	Date modified	Type	Size
 Completetime.csv	14/05/2022 08:56 PM	Microsoft Excel Co...	1,308 KB
 ConsumerAddress.txt	04/05/2022 07:11 PM	Text Document	215 KB
 ConsumptionUnit.xlsx	10/05/2022 08:58 PM	Microsoft Excel W...	38 KB
 Electricity_Consumer.bak	10/05/2022 11:10 PM	BAK File	13,496 KB
 Electricity_PowerSuppliers.bak	06/05/2022 10:11 PM	BAK File	3,993 KB
 PowerConsumption 2016.csv	07/05/2022 10:47 PM	Microsoft Excel Co...	704 KB
 PowerConsumption 2017.xls	07/05/2022 10:50 PM	Microsoft Excel 97...	2,047 KB
 PowerDistributionInfo.xls	07/05/2022 07:03 PM	Microsoft Excel 97...	38 KB
 PowerGeneratorInfo.csv	07/05/2022 06:47 PM	Microsoft Excel Co...	19 KB

Data Source Type	Source Name	Column Name	Data Type	Description
Database File(.bak)	ConsumerDetails	ConsumerID	smallint	Unique Id of the consumer
		AccountNum	smallint	Unique Account Number of the consumer
		Consumer_Fname	nvarchar(50)	First name of the consumer
		Consumer_Lname	nvarchar(50)	Last name of the consumer
		age	tinyint	Age of the consumer
		PhoneNumber	nvarchar(50)	Phone number of the consumer
		race	nvarchar(50)	Ethnicity of the consumer
		sex	nvarchar(50)	Gender of the consumer
		Type	tinyint	Type of Consumer
		email	nvarchar(50)	Email of the consumer
		Did	int	Distribution Cluster ID
	ConsumerType	typeID	tinyint	Unique ID for type
		ConsumerType	nvarchar(50)	Name of the consumer type
	SupplierCompany	SupplierID	tinyint	Unique ID for the supplier
		CompanyName	nvarchar(50)	Power supplier company name
		year_founded	smallint	Year of the company started
		HQAddress	nvarchar(50)	Address of the company headquarters
		Province	nvarchar(50)	Province of the HQ
		City	nvarchar(50)	City location
		PostalCode	smallint	Postal code of location
		Country	nvarchar(50)	Name of the county
		Departments	tinyint	Number of departments under the company
		PowerPlants	smallint	Number of power plants under the company
		Branches	tinyint	Number of branches under the company
	InstallServiceInfo	ConsumerID	smallint	Id of the consumer
		InstallFee	float	Connection Install fee
		PowerSupplierID	tinyint	ID of the supplier
Text(.txt)	ConsumerAddress.txt	ConsumerID	smallint	Unique ID of the Consumer
		Consumer_Street	nvarchar(50)	Address of the consumer
		Province	nvarchar(50)	Province name of consumer
		city	nvarchar(50)	City name of the consumer
		Postal_code	nvarchar(50)	Postal code of the consumer
		Country	nvarchar(50)	Country name of the consumer
		Latitude	float	Latitude of the consumer address
		Longitude	float	Longitude of the consumer address
Excel 97-2003 Worksheet(.xls)	PowerDistributionInfo.xls	PowerDistributionID	int	Unique ID for power distribution
		PowerGenID	int	Power plant ID
		StartDate	date	Start date of the distribution
	PowerConsumption 2017.xls	ConsumptionID	int	Unique ID of the power consumption of consumers
		AccountNum	smallint	Account number of the consumer
		NoOfUnits	float	No of units consume by user
		SmartMeterPerc	float	Incentive rate of the consumer
		entrant	int	Shows the consumer is a new user or not
		Date	date	Date of power consumption
Excel Woksheet(.xlsx)	ConsumptionUnit.xlsx	TypeID	tinyint	Type of the consumer

Comma Separated Values (.csv)		unitPrice	float	Unit price of the electricity for the day
		priceDate	date	Unique date for the price of the electricity market
		consumerType	nvarchar(255)	Name of the type of the user
	PowerConsumption 2016.csv	ConsumptionID	int	Unique ID of the power consumption of consumers
		AccountNum	smallint	Account number of the consumer
		NoOfUnits	float	No of units consume by user
		SmartMeterPerc	float	Incentive rate of the consumer
		entrant	int	Shows the consumer is a new user or not
	PowerGeneratorInfo.csv	Date	date	Date of power consumption
		PLANT_ID	int	Unique ID for power plant
		DC_POWER	real	DC power capacity of power plant
		AC_POWER	real	AC power capacity of power plant
		DAILY_YIELD	real	Daily generated electricity
		TOTAL_YIELD	real	Total Generated electricity
		Type	nvarchar(20)	Type of the electricity source
		power_supplierID	tinyint	ID of the power supplier

Step 3: Solution Architecture



The above architecture shows the high-level DW & BI solution design.

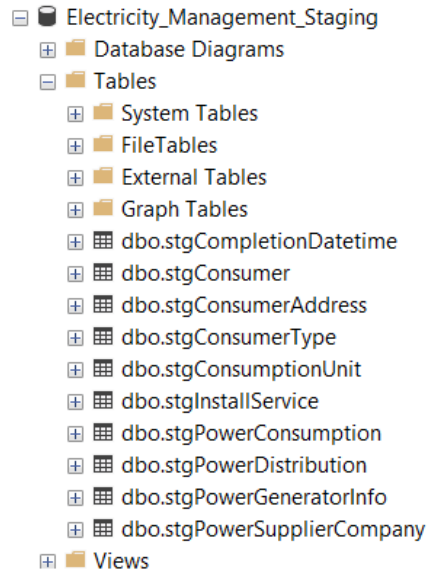
Data Sources

‘.txt’ component represents Text files, ‘.xls’ and ‘xlsx’ component is used to represent versions of Excel files, ‘.csv’ component is used to display Comma Separated files and ‘.bak’ component represents database files.

Staging Area

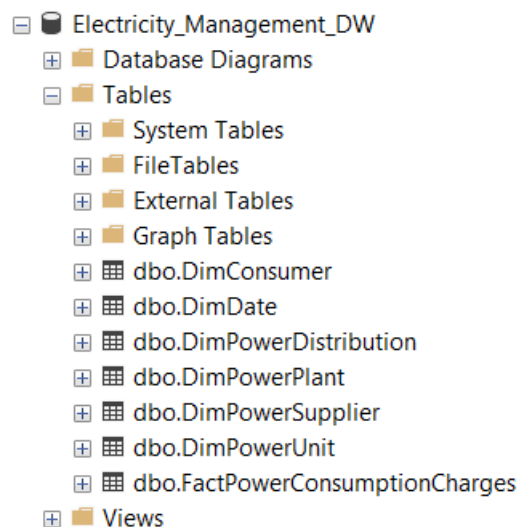
Landing DB component represents the process of the creating database *Electricity_Consumer* DB *Electricity_PowerSupplier* DB. *Consumer*, *ConsumerType*, *SupplierCompany*, and *InstallServiceInfo* were imported to the database and were used to create the tables. And these tables were used as the DB source data.

The staging DB component represents creating staging level tables by extracting and loading the data from various data sources.



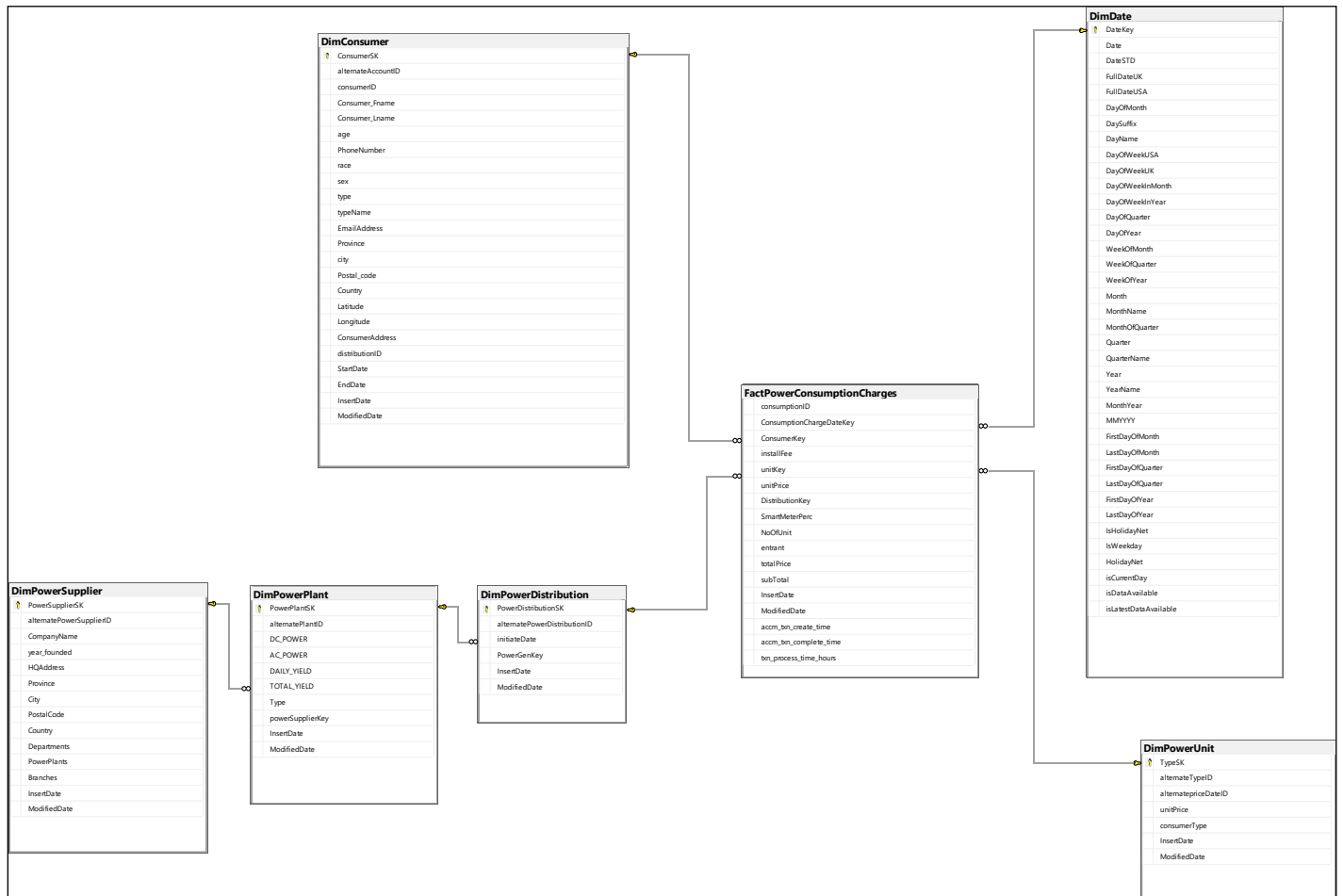
Data Warehouse

The data warehouse DB component is used to display the formation of dimension tables and fact tables in the warehouse by extracting, transforming, and loading the data into the data warehouse from the staging



Step 4: Data Warehouse Design & Development

The following figure will show how the fact table and dimension tables were combined rationally.



Schema Type

For this scenario, the **snowflake** schema type was used.

Dimensions and Fact Table

- FactPowerConsumptionCharges – Fact Table

The fact table contains all the information on power consumption and charges

- DimPowerDistribution

All the power distribution collections of power generation

- DimPowerPlant

All the power generation information related to the power plant

- DimPowerSupplier

All the information about power suppliers who contribute electricity to the power generation.

- DimPowerUnit

All the unit prices of the electricity market throughout the year

- DimConsumer

All the commercial and domestic consumer details who consume electricity

- DimDate – Date dimension

Dimension Types

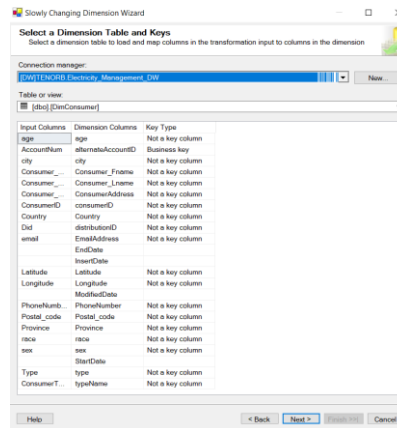
- Hierarchical Dimension
 - Power Generation: DimPowerSupplier→ DimPowerPlant→DimPowerDistribution
 - DimConsumer: Country → Province → City → Postal code → Address
 - Date – all the hierarchies in date
- Slowly Changing Dimension
 - DimConsumer – Consumer dimension used type 2.
 - Following columns were set as changing attributes.
 - Email
 - PhoneNumber
 - Type
 - typeName

- Fact Table
 - Numerical columns – installFee, unitPrice, SmartMeterPerc, NoOfUnit
 - Derived columns in Fact table,
 - $\text{totalPrice} = \text{unitPrice} * \text{NoOfUnit}$
 - $\text{subtotal} = \text{totalPrice} * \left(1 - \frac{\text{SmartMeterPerc}}{100}\right)$
 - FK – Consumer Key, ConsumptionChargeDate Key, Unit Key, Distribution Key

Assumptions and Justifications

- The consumer dimension was considered as a slowly changing dimension.
- SmartMeterPerc – Consumers who save the electricity units with respect to the previous month will be rewarded with an incentive percentage which will be decreased from the current charge. (Percentage was given by the smart meter installed in consumer)

Dimension Columns	Change Type
age	Changing a...
city	Historical a...
Consumer_Fname	Historical a...
Consumer_Lname	Historical a...
ConsumerAddress	Historical a...
Country	Historical a...
distributionID	Historical a...
EmailAddress	Changing a...
Latitude	Historical a...
Longitude	Historical a...
PhoneNumber	Changing a...
Postal_code	Historical a...
Province	Historical a...
race	Historical a...
sex	Historical a...
type	Changing a...
typeName	Changing a...



Step 5: ETL development

5.1. ETL Process to Staging Database

Extraction from Data sources:

In this step, All the data sources were extracted and loaded to the staging tables by using the relevant Data connection. Flat file connection was used for text files and CSV files, Excel file connections for excel files, and DB source connection for DB files.

All those tables and source types that were imported to the Electricity_Management_Staging, which contains the below tables,

#	Table name	Source type	Source
1	stgConsumer	DB source	Electricity_Consumer.dbo.ConsumerDetails
2	stgConsumerAddress	Text File	ConsumerAddress.txt
3	stgConsumerType	DB source	Electricity_Consumer.dbo.ConsumerType
4	stgConsumptionUnit	Excel (xlsx)	ConsumptionUnit.xlsx
5	stgInstallService	DB Source	Electricity_PowerSuppliers.dbo.InstallServiceInfo
6	stgPowerConsumption	Excel(xls)	PowerConsumption 2017.xls
		CSV	PowerConsumption 2016.csv
7	stgPowerDistribution	Excel(xls)	PowerDistributionInfo.xls
8	stgPowerGeneratorInfo	CSV	PowerGeneratorInfo.csv
9	stgPowerSupplierCompany	CSV	Electricity_PowerSuppliers.dbo.SupplierCompany

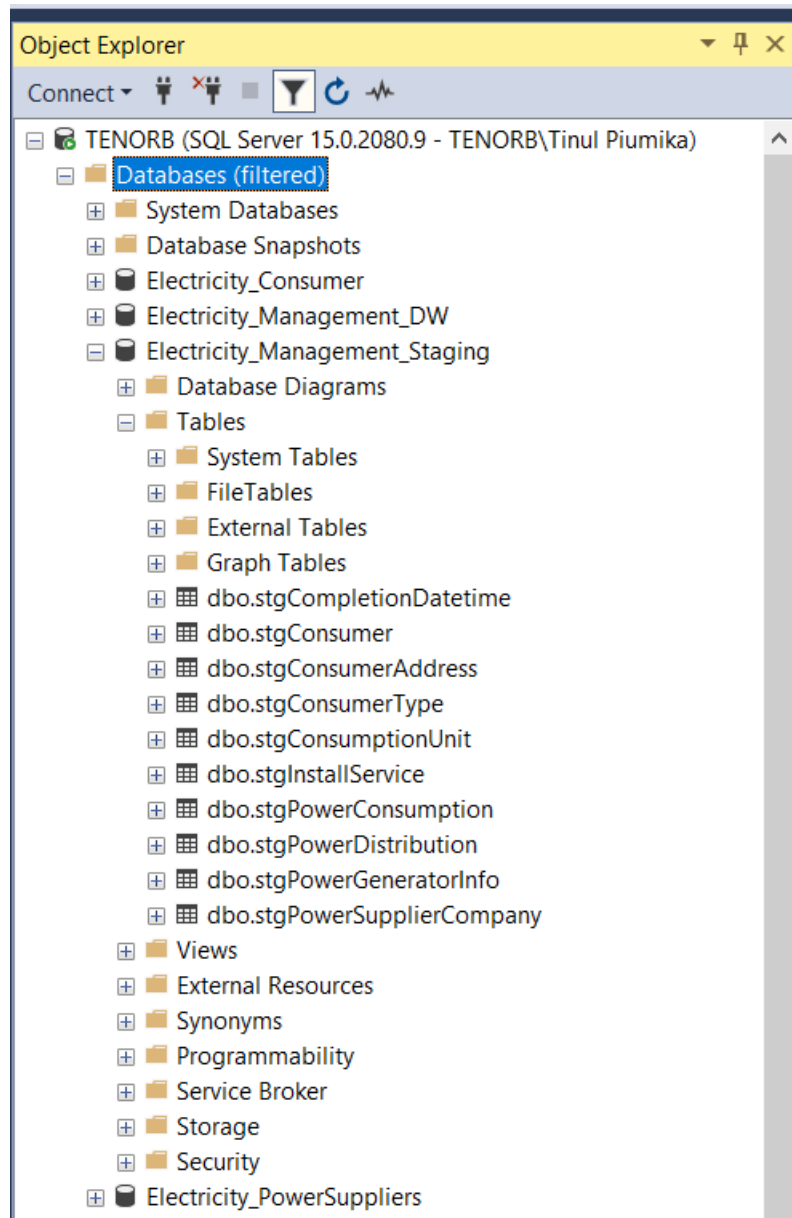
Transform and Loading to Staging Database:

In transformation, a set of rules or functions are applied to the extracted data to convert it into a single standard format. It may involve the following processes/tasks:

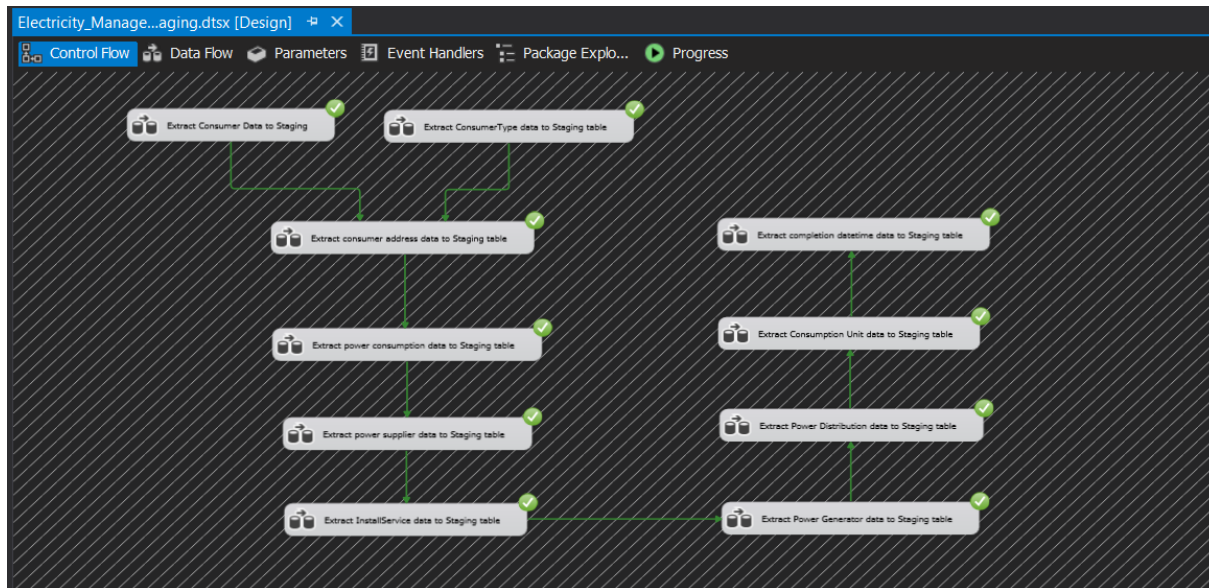
- Filtering – loading only relevant attributes into the data warehouse.
- Joining – Joining the union of data sources PowerConsumption2016.xls and PowerConsumption2017.csv into Power consumption staging

At last, all the extracted and transformed data is finally loaded into the staging tables.

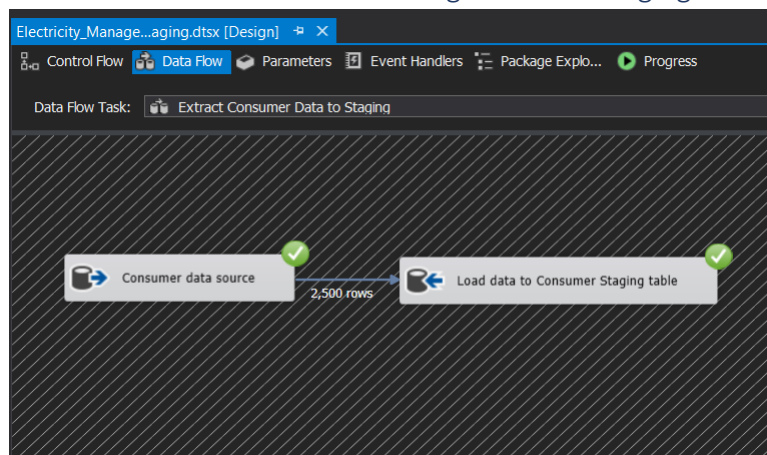
- **Snapshot of SSMS Staging Database**



5.1. Overall Control Flow of Extraction



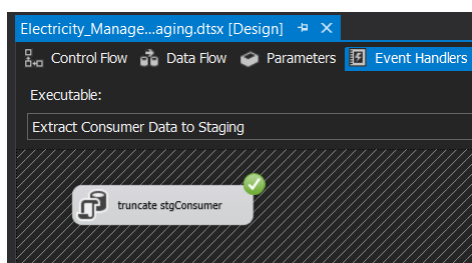
5.1.1.1 Consumer data source to stgConsumer staging



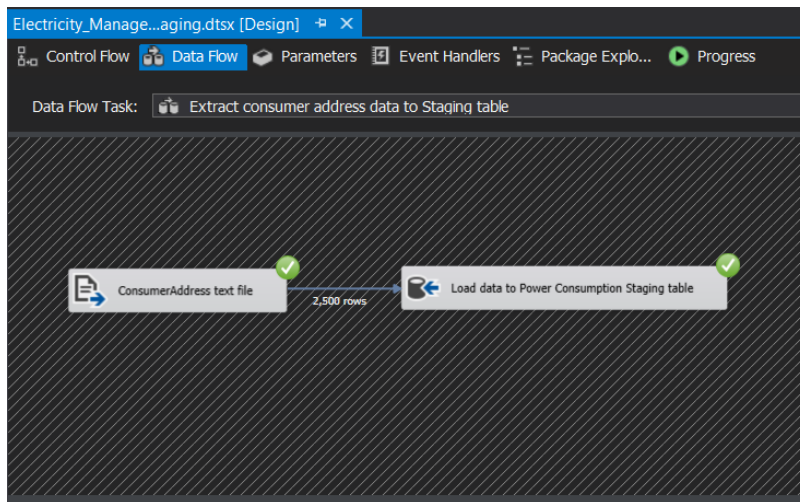
Consumer data in Electricity Consumer database has been extracted and loaded to stgConsumer Staging table

5.1.1.2 Event Handler

Before executing the 'Extract Consumer Data to Staging', existing data in the staging layer has been truncated.



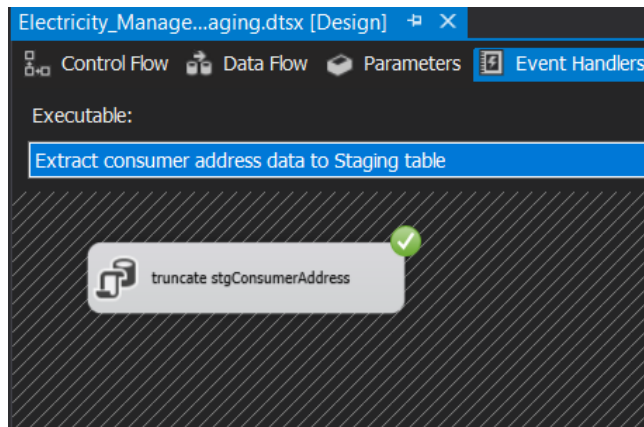
5.1.2.1 Consumer address data source to stgConsumerAddress staging



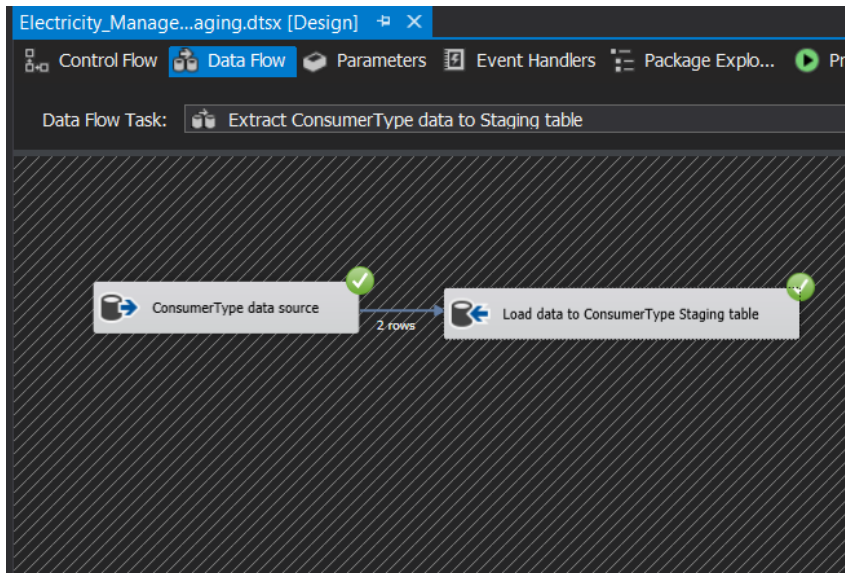
Consumer address data in Consumer address text file has been extracted and loaded to stgConsumerAddress Staging table

5.1.2.2 Event Handler

Before executing the 'Extract Consumer address Data to Staging', existing data in the staging layer has been truncated.



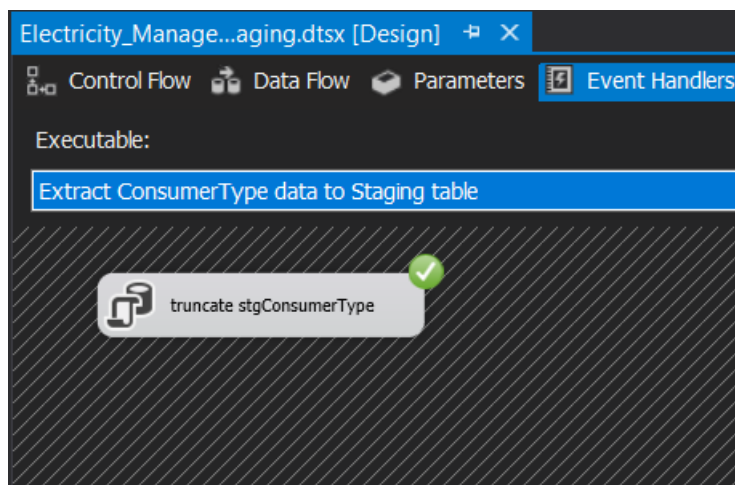
5.1.3.1 Consumer type data source to stgConsumerType staging



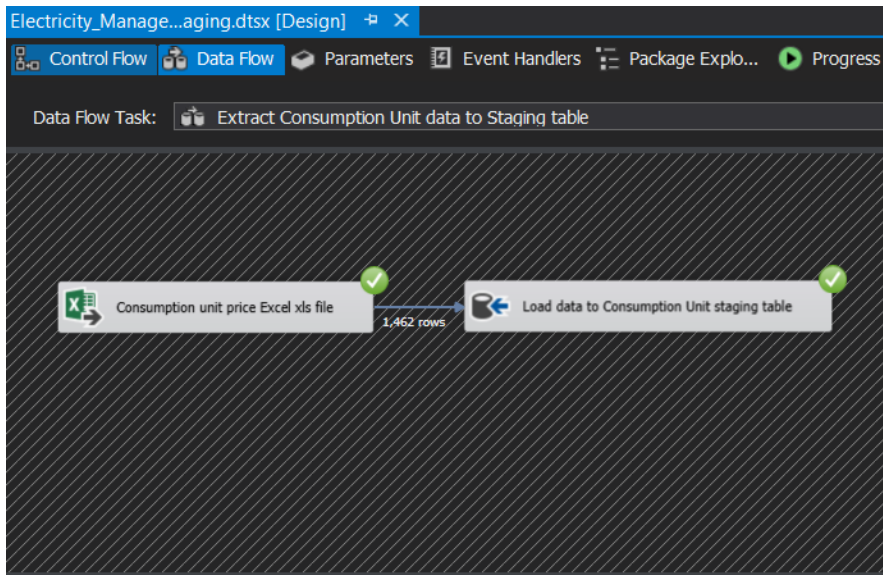
Consumer type data in Electricity Consumer database has been extracted and loaded to stgConsumerType Staging table

5.1.3.2 Event Handler

Before executing the 'Extract Consumer type Data to Staging', existing data in the staging layer has been truncated.



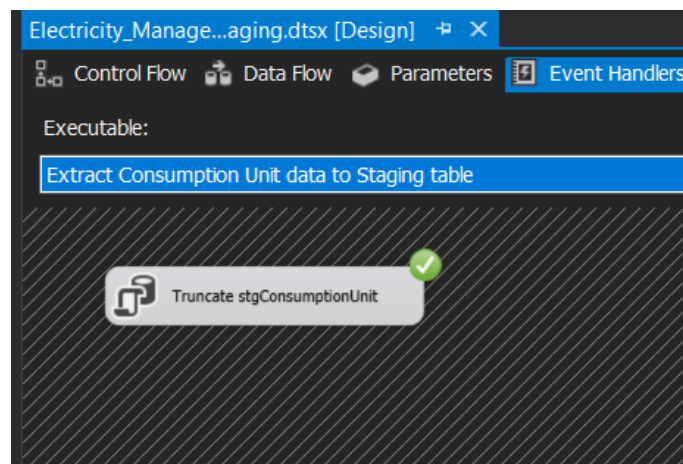
5.1.4.1 Consumption unit price data source to stgConsumptionUnit staging



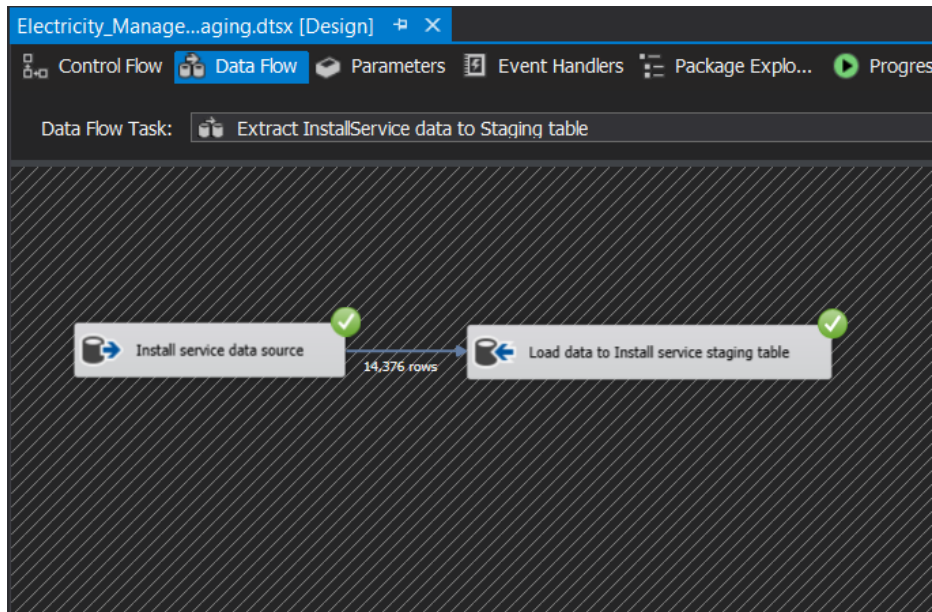
Consumption unit data in consumer unit Excel file has been extracted and loaded to stgConsumptionUnit Staging table

5.1.4.2 Event Handler

Before executing the 'Extract Consumption unit data to Staging', existing data in the staging layer has been truncated.



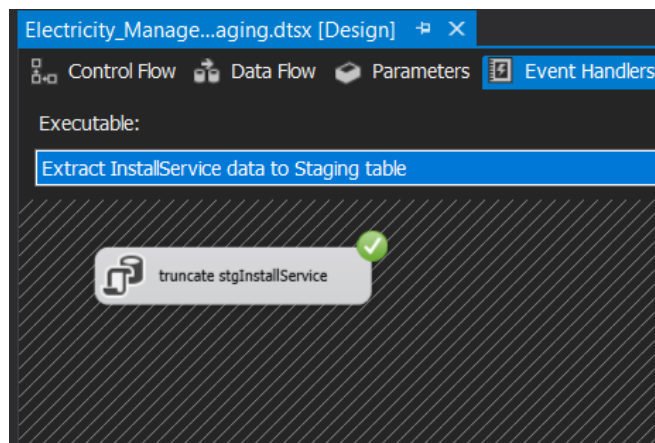
5.1.5.1 Install service data source to stgInstallService staging



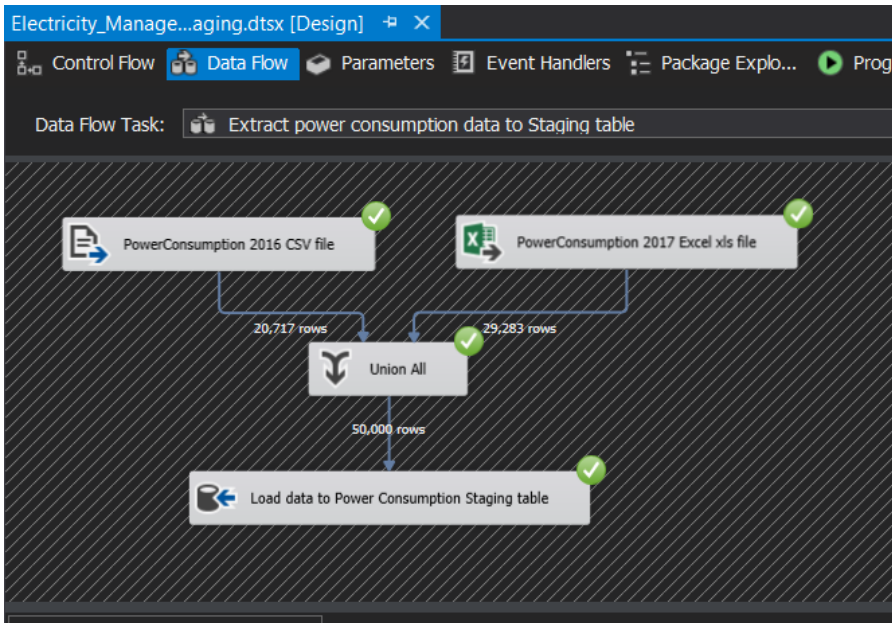
Install service data in consumer database has been extracted and loaded to stgInstallService Staging table

5.1.5.2 Event Handler

Before executing the 'Extract Install service data to Staging', existing data in the staging layer has been truncated.



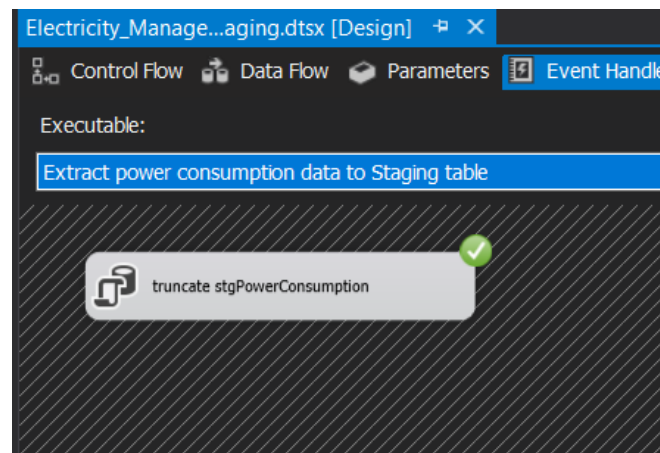
5.1.6.1 Power consumption data sources to stgPowerConsumption staging



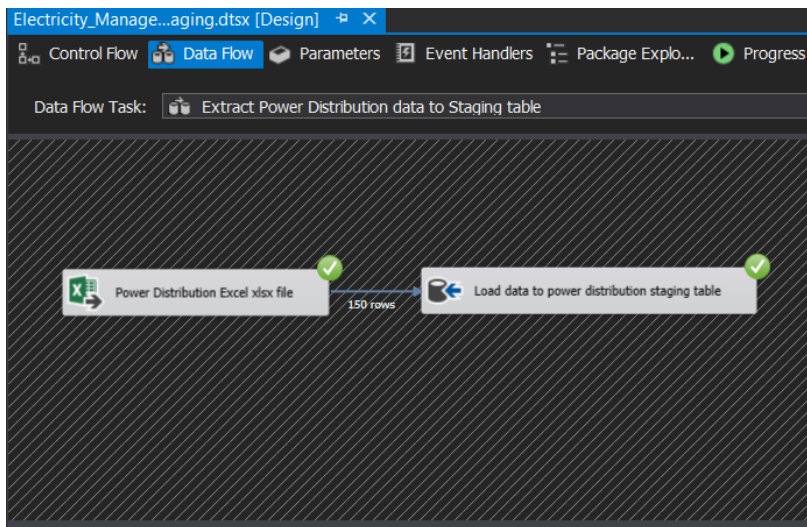
Power consumption data in Power consumption Excel and CSV file has been extracted, Union merge and loaded to stgPowerConsumption Staging table

5.1.6.2 Event Handler

Before executing the 'Extract Consumption data to Staging', existing data in the staging layer has been truncated.



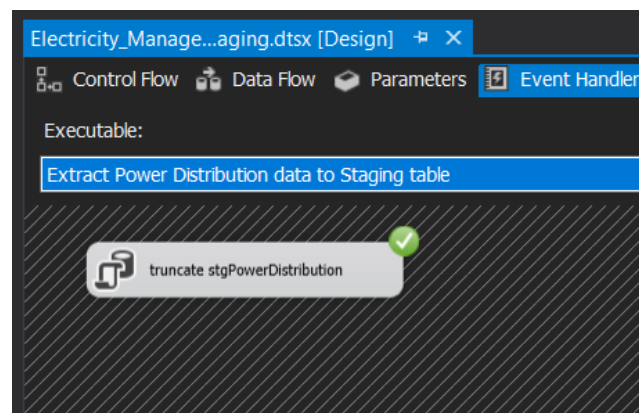
5.1.7.1 Power distribution data sources to stgPowerDistribution staging



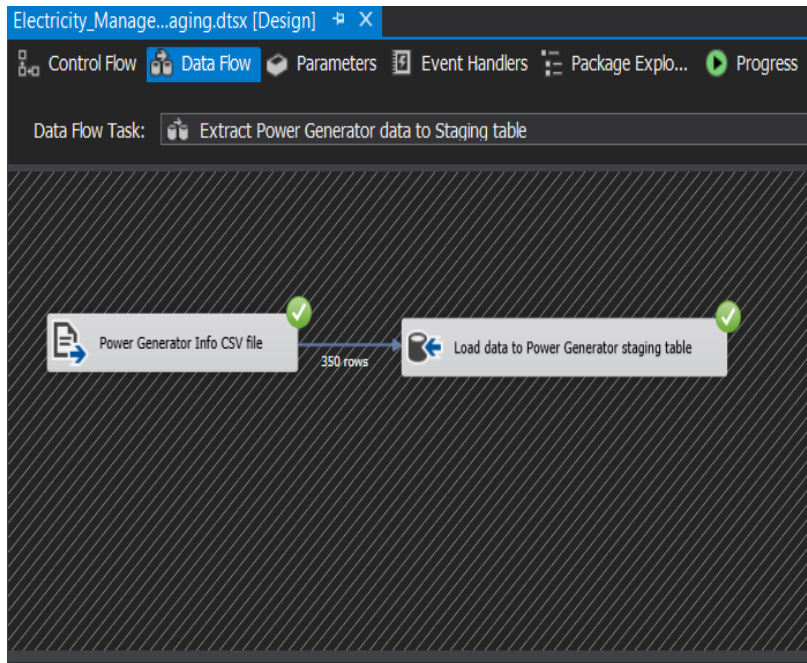
Power Distribution data in power distribution excel file has been extracted and loaded to stgPowerDistribution Staging table

5.1.7.2 Event Handler

Before executing the 'Extract Power distribution data to Staging', existing data in the staging layer has been truncated.



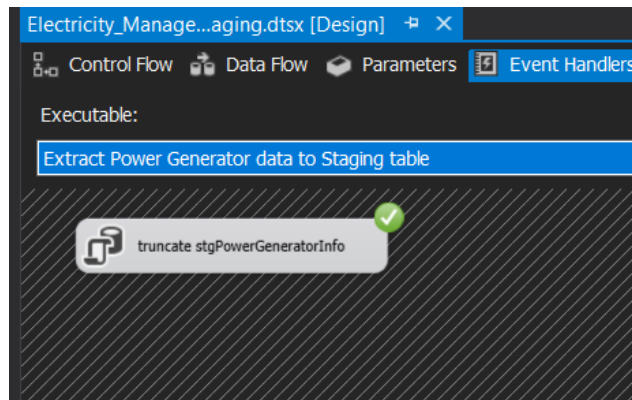
5.1.8.1 Power generator data sources to stgPowerGeneratorInfo staging



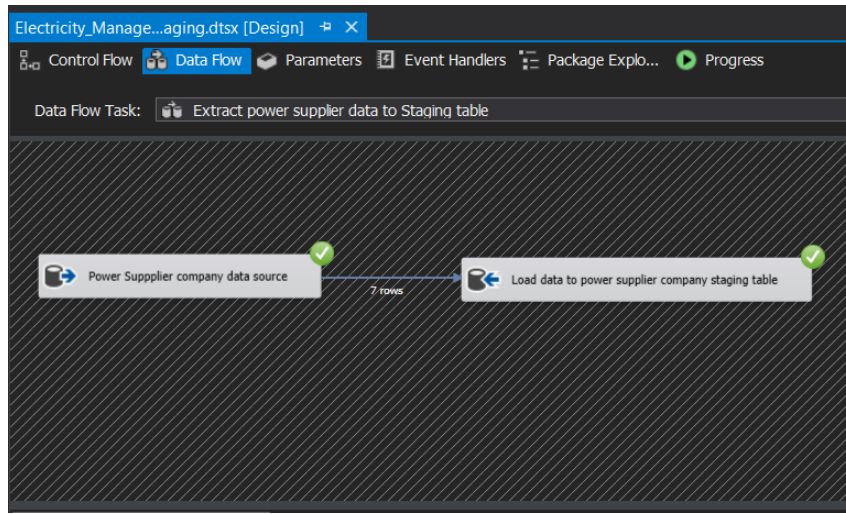
Power generator data in power generator info CSV file has been extracted and loaded to stgPowerGeneratorInfo Staging table

5.1.8.2 Event Handler

Before executing the 'Extract Power generator data to Staging', existing data in the staging layer has been truncated.



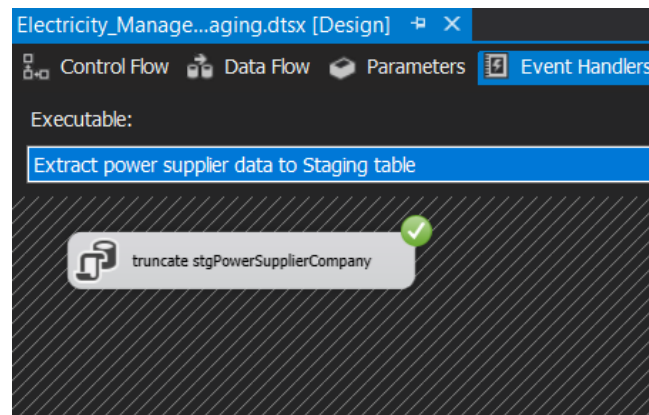
5.1.9.1 Power Supplier data sources to stgPowerSupplierCompany staging



Power Supplier data in power Electricity Power Supplier has been extracted and loaded to stgPowerSupplierCompany Staging table

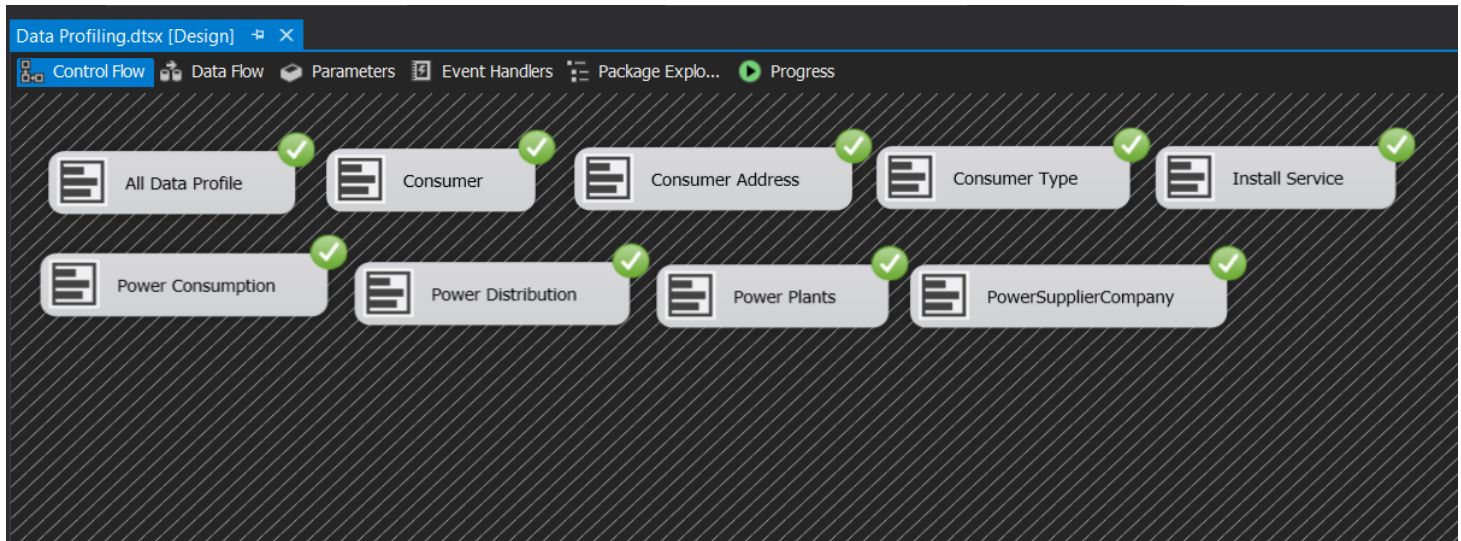
5.1.9.2 Event Handler

Before executing the 'Extract Power supplier data to Staging', existing data in the staging layer has been truncated.



- **Data profiling**

Used Data_Profiling package to profiling the staging tables



5.2 ETL Process to Data Warehouse

Extract

All the data were extracted from staging tables by using the Database connection.

Transform & Load

All the extracted datasets were Transformed and Loaded into the Data warehouse. As the first step, the Dimension tables and fact tables in the Datawarehouse were created. In transformation, a set of rules or functions are applied to the extracted data to convert it into a single standard format. It may involve the following processes/tasks:

Used Transformation Tasks

1. Lookups

- In Power plant dimensions' PowerPlantID is looked at when Power distribution Staging is loading to the Power distribution dimension table (DimPowerDistribution)
- In Power Plant Dimensions' PowerPlantID is looked at when Power distribution staging is loading to the Power Distribution Dimension
- Lookups are used when getting the surrogate keys from dimensions to the fact table

2. Derived Columns

- Replace Null race types in the Consumer dimension table
- Calculate the total price from the unit price from DimPowerUnit and no of units from Power consumption staging and assign the sum into the derived column and load it into the Fact table.

$$\text{totalPrice} = \text{unitPrice} * \text{NoOfUnit}$$

- Calculate the subtotal using the previous derived column total price and incentive percentage from Power consumption staging and assign the sum into the derived column and load it into the Fact table.

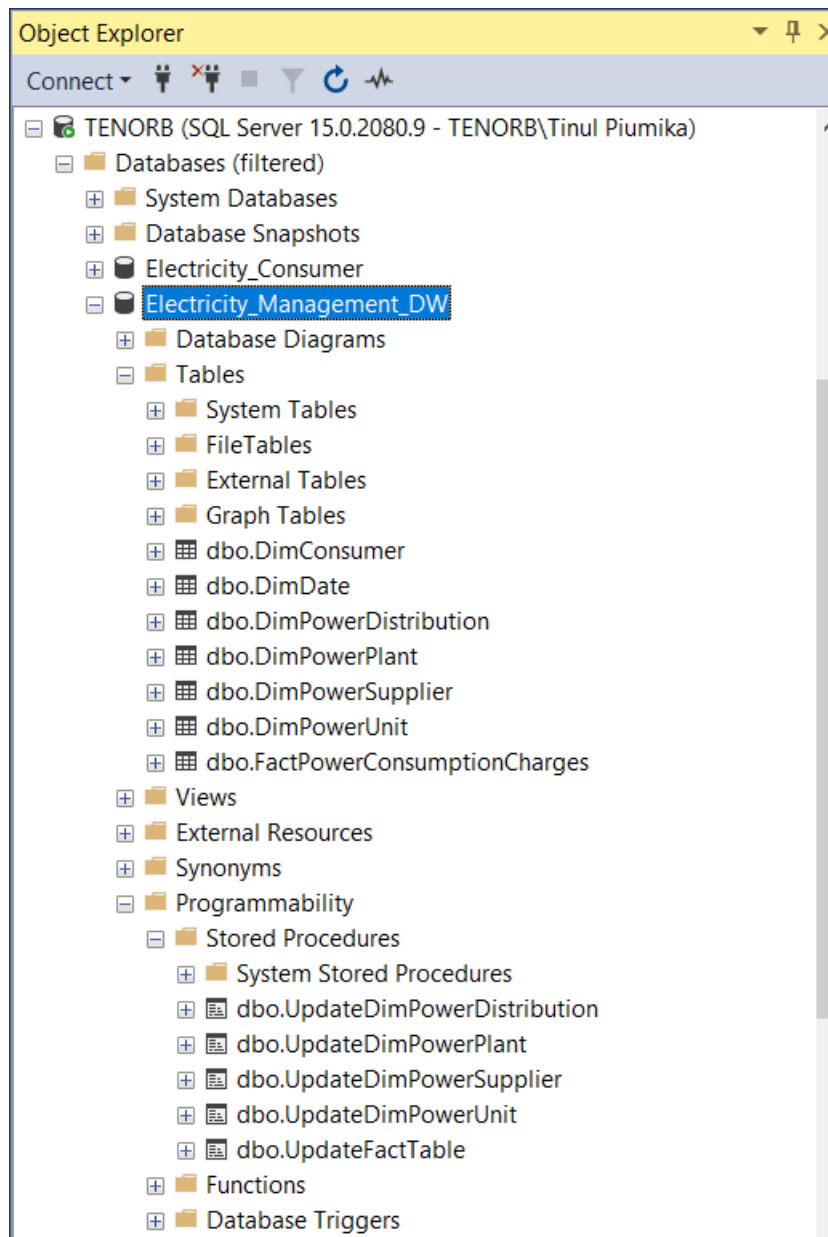
$$\text{subtotal} = \text{totalPrice} * \left(1 - \frac{\text{SmartMeterPerc}}{100}\right)$$

- ##### 3. Union - Union is used to transform and combines all the data from both Consumer staging and consumer address staging loading into DimConsumer.

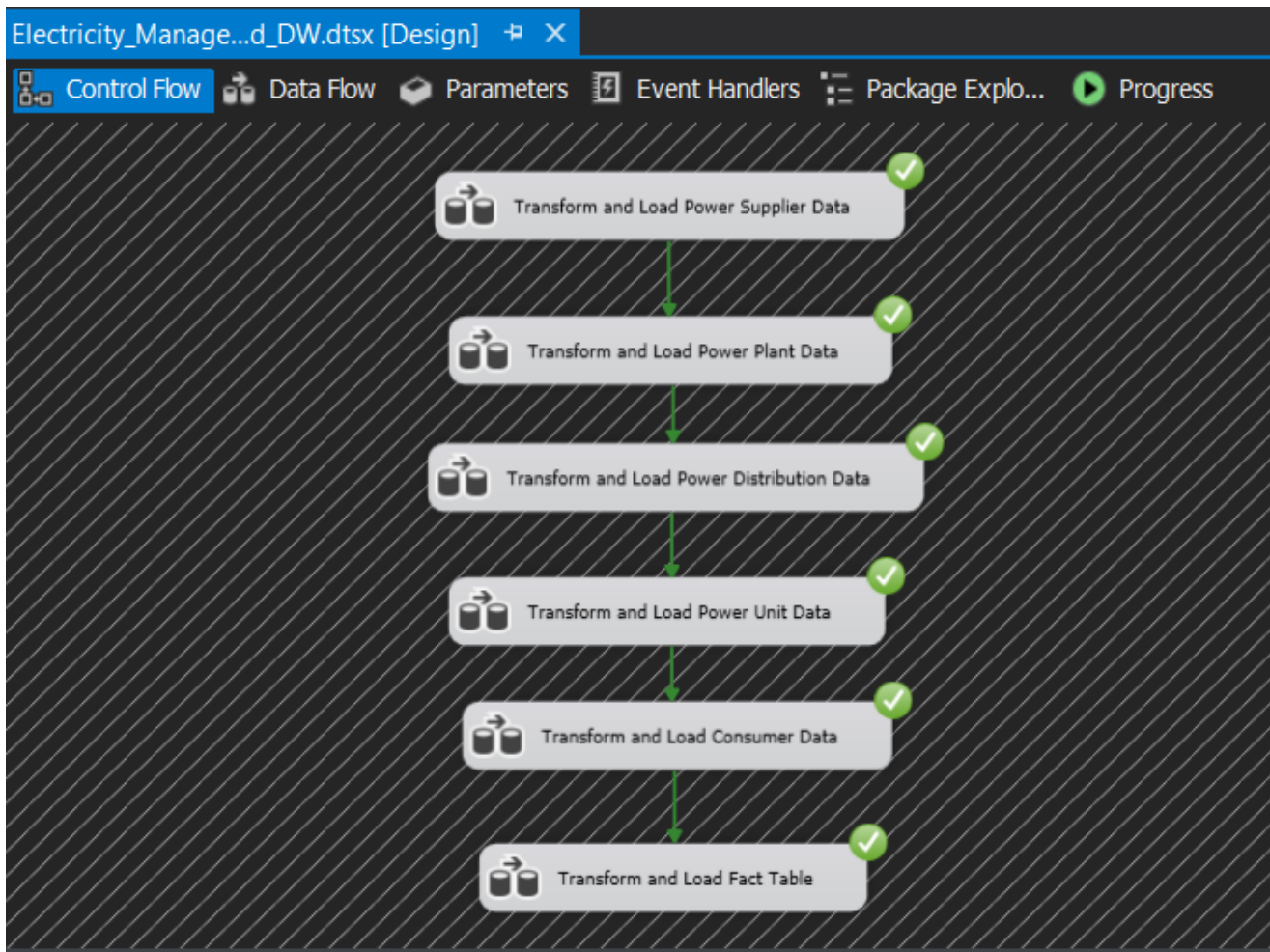
4. Sort and Merge

- When transforming and loading Power Plant Data into the data warehouse, Supplier ID was sorted in both stgPowerGeneratorInfo table (Power Plant Staging table) data source and PowerSupplier Dimension data source and merge those data using 'Merge Join' Tool which left outer join the both tables
- When transforming and loading consumer details into the data warehouse, Consumer ID was sorted in both stgConsumerAddress table data source and stgConsumer table data source and merge those data using 'Merge Join' Tool which left outer join the both tables

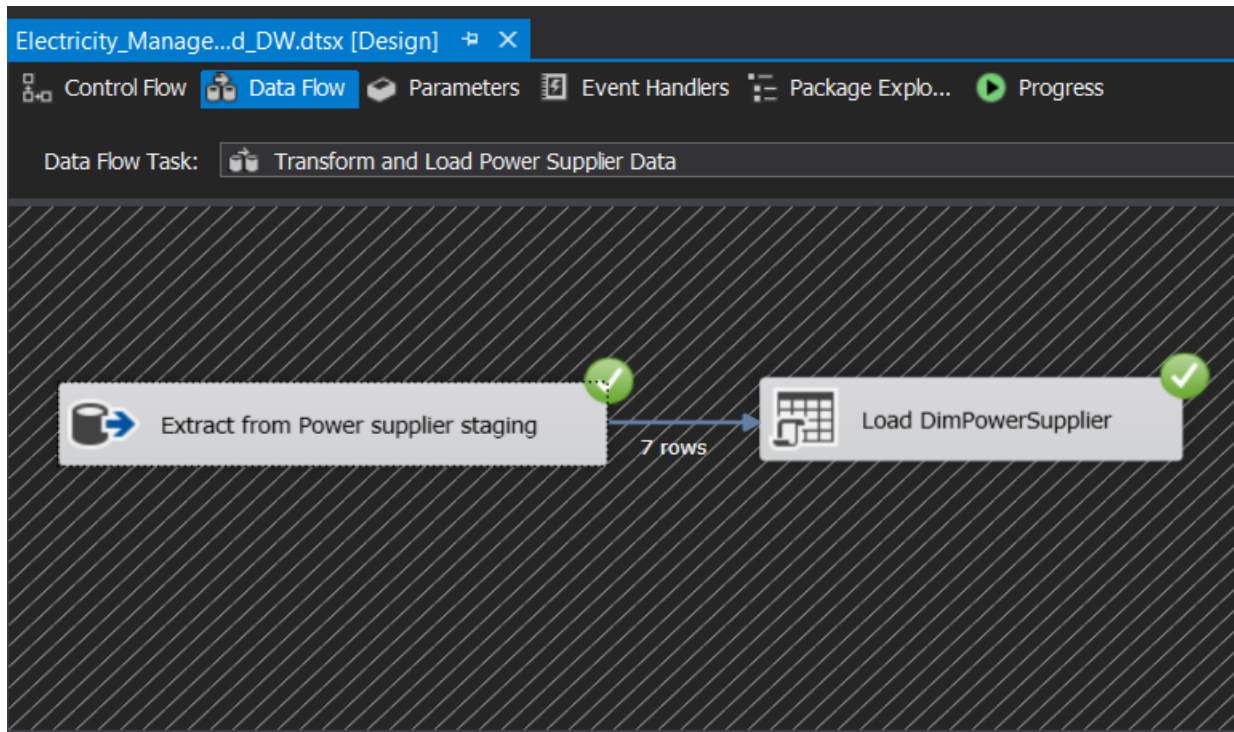
- Snapshot of SQL server Data warehouse Database



5.2 Overall Control Flow of Extraction



5.2.1 Transform and load power supplier data to DimPowerSupplier



ETL procedure when loading DimPowerSupplier

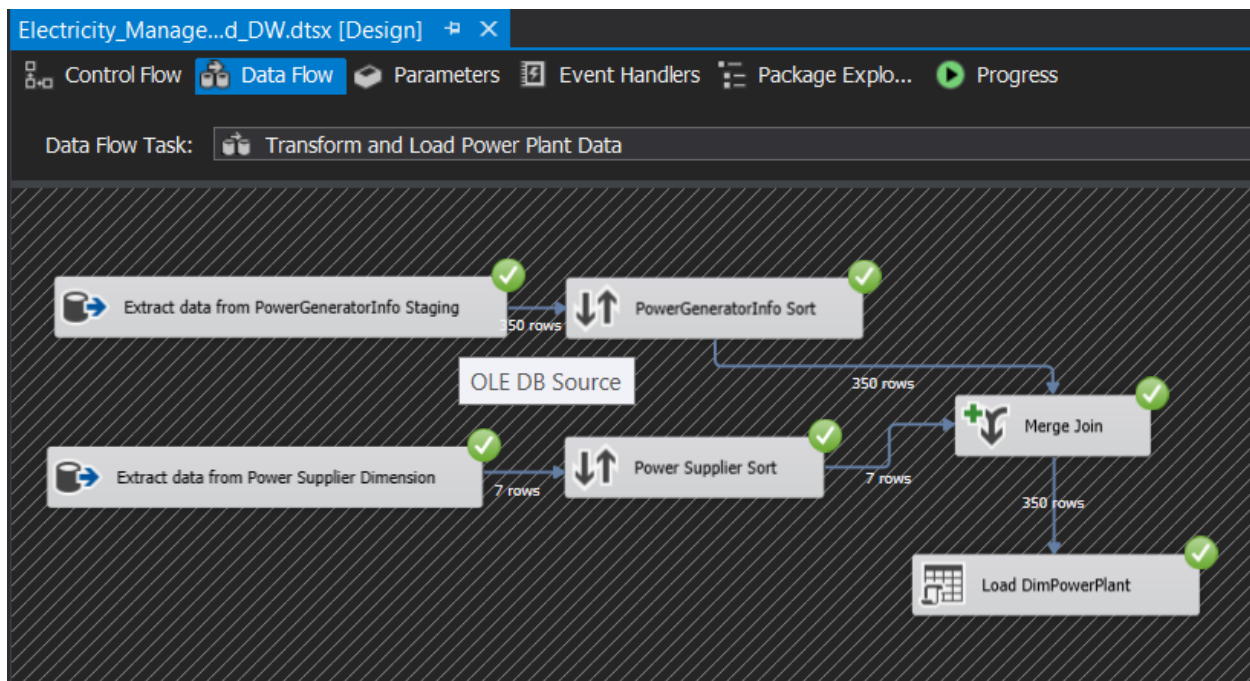
```
DROP PROCEDURE if exists dbo.UpdateDimPowerSupplier;
CREATE PROCEDURE dbo.UpdateDimPowerSupplier
@powerSupplierID int,
@CompanyName nvarchar(50),
@year_founded nvarchar(50),
@HQAddress nvarchar(50),
@Province nvarchar(50),
@City nvarchar(50),
@PostalCode smallint,
@Country nvarchar(50),
@Departments tinyint,
@PowerPlants smallint,
@Branches tinyint
AS BEGIN
    if not exists (select PowerSupplierSK
from dbo.DimPowerSupplier
where alternatePowerSupplierID = @powerSupplierID)
    BEGIN
        insert into dbo.DimPowerSupplier (alternatePowerSupplierID, CompanyName,
year_founded, HQAddress, Province, City, PostalCode,Country,
Departments,PowerPlants, Branches, InsertDate, ModifiedDate)
values(@powerSupplierID, @CompanyName, @year_founded, @HQAddress,@Province,
@City, @PostalCode,@Country, @Departments, @PowerPlants, @Branches,
GETDATE(), GETDATE())
    END;
END;
```

```

if exists (select PowerSupplierSK
from dbo.DimPowerSupplier
where alternatePowerSupplierID = @powerSupplierID)
BEGIN
    update dbo.DimPowerSupplier
    set alternatePowerSupplierID = @powerSupplierID, CompanyName=@CompanyName,
    year_founded=@year_founded, HQAddress=@HQAddress, Province=@Province,
    City=@City, PostalCode=@PostalCode, Country=@Country,
    Departments=@Departments, PowerPlants=@PowerPlants, Branches=@Branches,
    ModifiedDate = GETDATE()
    where alternatePowerSupplierID = @powerSupplierID
END;
END;

```

5.2.2 Transform and load power plant data to DimPowerPlant



ETL procedure when loading DimPowerPlant

```

DROP PROCEDURE if exists dbo.UpdateDimPowerPlant;
CREATE PROCEDURE dbo.UpdateDimPowerPlant
@alternatePlantID int,
@DC_POWER real,
@AC_POWER real,
@DAILY_YIELD real,
@TOTAL_YIELD real,
@Type nvarchar(20),
@powerSupplierKey int
AS BEGIN
    if not exists (select PowerPlantSK
from dbo.DimPowerPlant
where alternatePlantID = @alternatePlantID )

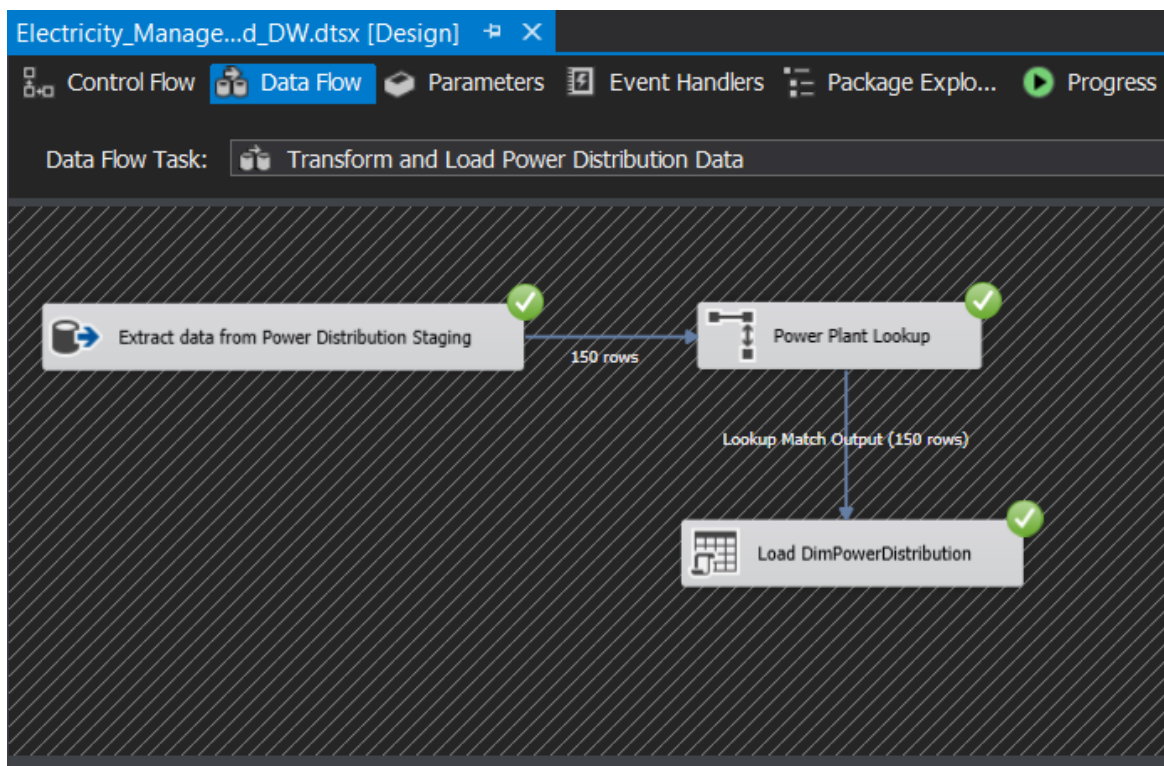
```

```

BEGIN
    insert into dbo.DimPowerPlant (alternatePlantID, DC_POWER, AC_POWER,
    DAILY_YIELD,TOTAL_YIELD,Type, powerSupplierKey, InsertDate, ModifiedDate)
    values(@alternatePlantID, @DC_POWER, @AC_POWER, @DAILY_YIELD,@TOTAL_YIELD,
    @Type, @powerSupplierKey, GETDATE(), GETDATE())
END;
if exists (select PowerPlantSK
from dbo.DimPowerPlant
where alternatePlantID = @alternatePlantID )
BEGIN
    update dbo.DimPowerPlant
    set alternatePlantID = @alternatePlantID, DC_POWER=@DC_POWER,
    AC_POWER=@AC_POWER,DAILY_YIELD = @DAILY_YIELD, TOTAL_YIELD=@TOTAL_YIELD,
    Type=@Type, powerSupplierKey=@powerSupplierKey, ModifiedDate = GETDATE()
    where alternatePlantID = @alternatePlantID
END;
END;

```

5.2.3 Transform and load power distribution data to DimPowerDistribution

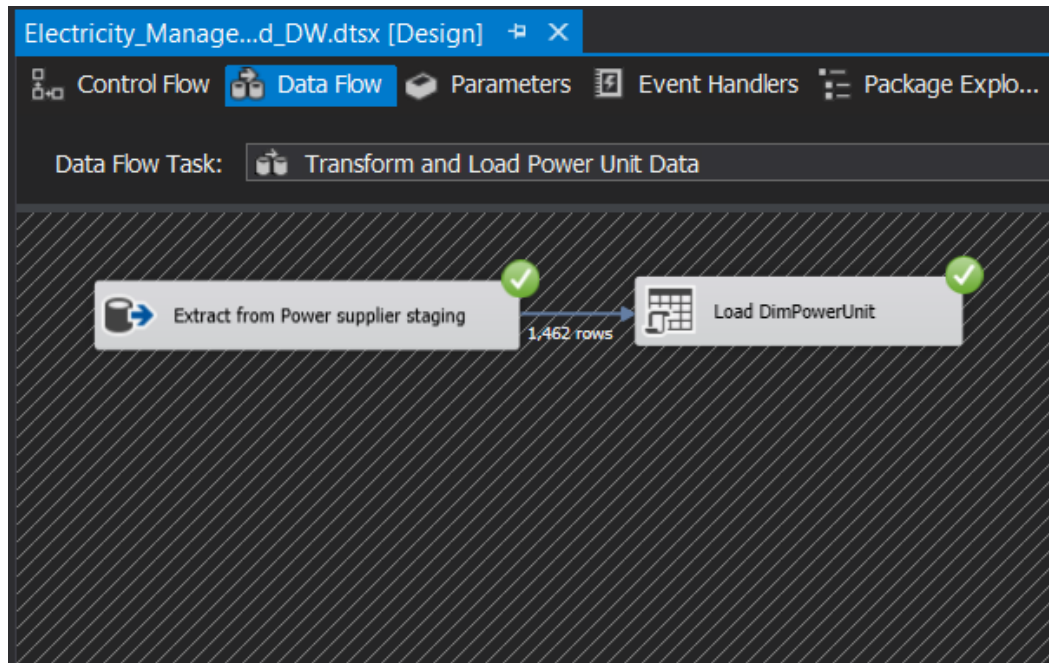


ETL procedure when loading DimPowerDistribution

```
DROP PROCEDURE if exists dbo.UpdateDimPowerPlant;
CREATE PROCEDURE dbo.UpdateDimPowerDistribution
@alternatePowerDistributionID int,
@initiateDate date,
@PowerGenKey int
AS BEGIN
    if not exists (select PowerDistributionSK
from dbo.DimPowerDistribution
where alternatePowerDistributionID = @alternatePowerDistributionID )
        BEGIN
            insert into dbo.DimPowerDistribution (alternatePowerDistributionID,
            initiateDate, PowerGenKey, InsertDate, ModifiedDate)
            values(@alternatePowerDistributionID, @initiateDate, @PowerGenKey,
            GETDATE(), GETDATE())
        END;

    if exists (select PowerDistributionSK
from dbo.DimPowerDistribution
where alternatePowerDistributionID = @alternatePowerDistributionID )
        BEGIN
            update dbo.DimPowerDistribution
            set alternatePowerDistributionID = @alternatePowerDistributionID,
            initiateDate=@initiateDate, PowerGenKey=@PowerGenKey,
            ModifiedDate = GETDATE()
            where alternatePowerDistributionID = @alternatePowerDistributionID
        END;
END;
```

5.2.4 Transform and load power unit data to DimPowerUnit



ETL procedure when loading DimPowerUnit

```
DROP PROCEDURE if exists dbo.UpdateDimPowerUnit;
CREATE PROCEDURE dbo.UpdateDimPowerUnit
@alternateTypeID tinyint,
@alternatepriceDateID date,
@unitPrice float,
@consumerType nvarchar(225)
AS BEGIN
    if not exists (select TypeSK
from dbo.DimPowerUnit
where alternatepriceDateID = @alternatepriceDateID AND alternateTypeID =
@alternateTypeID)
        BEGIN
            insert into dbo.DimPowerUnit (alternateTypeID, alternatepriceDateID,
unitPrice, consumerType, InsertDate, ModifiedDate)
values(@alternateTypeID, @alternatepriceDateID, @unitPrice, @consumerType,
GETDATE(), GETDATE())
        END;

    if exists (select TypeSK
from dbo.DimPowerUnit
where alternatepriceDateID = @alternatepriceDateID AND alternateTypeID =
@alternateTypeID)
        BEGIN
            update dbo.DimPowerUnit
set alternateTypeID = @alternateTypeID,
alternatepriceDateID=@alternatepriceDateID, unitPrice=@unitPrice,
consumerType=@consumerType, ModifiedDate = GETDATE()
        END
END;
```

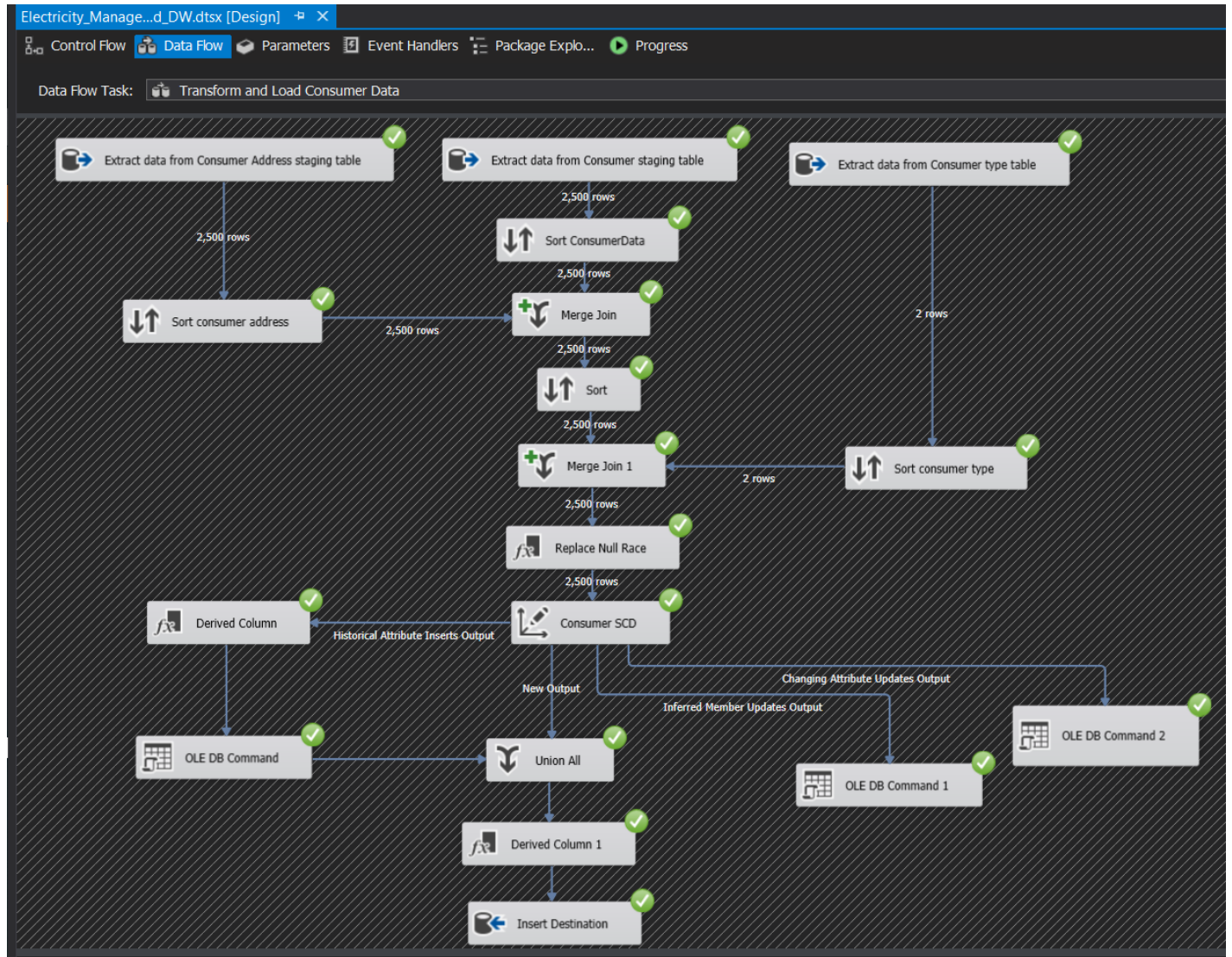


```

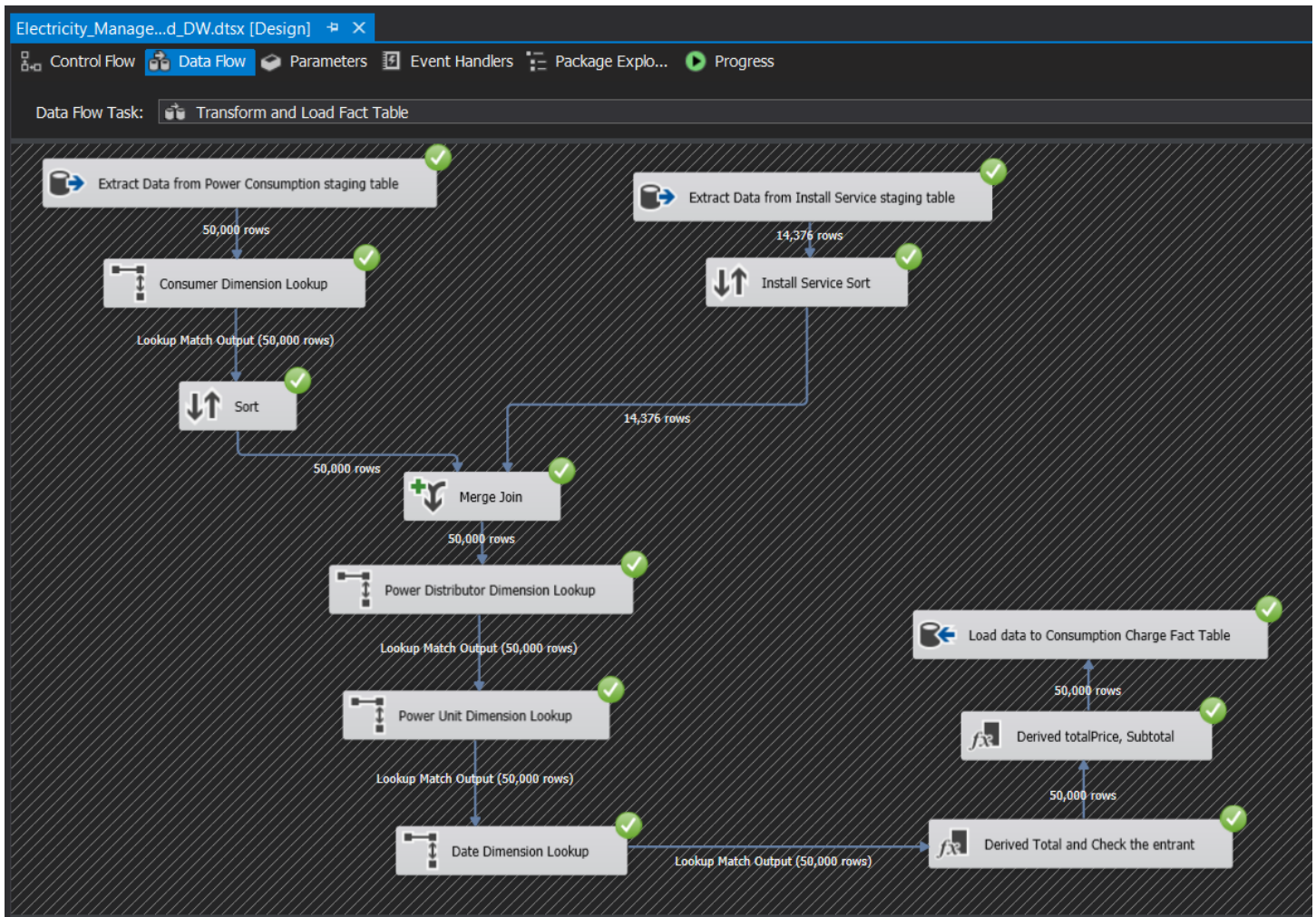
where alternatepriceDateID = @alternatepriceDateID AND
alternateTypeID = @alternateTypeID
END;
END;

```

5.2.5 Transform and load consumer data to DimConsumer



5.2.6 Transform and load data to fact table.



5.2.7 Creation of Date Dimension

```
BEGIN TRY
    DROP TABLE [dbo].[DimDate]
END TRY

BEGIN CATCH
END CATCH

CREATE TABLE [dbo].[DimDate]
(
    [DateKey] INT primary key,
    [Date] DATETIME,
    [DateSTD] DATE,
    [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 2013a
    [MonthYear] CHAR(10), --Jan-2013, Feb-2013
    [MMYYYY] CHAR(6),
    [FirstDayOfMonth] DATE,
    [LastDayOfMonth] DATE,
    [FirstDayOfQuarter] DATE,
    [LastDayOfQuarter] DATE,
    [FirstDayOfYear] DATE,
    [LastDayOfYear] DATE,
    [IsHolidayNet] BIT, -- Flag 1=National Holiday, 0-No National Holiday
    [IsWeekday] BIT, -- 0=Week End ,1=Week Day
    [HolidayNet] VARCHAR(50), --Name of Holiday in US
    [isCurrentDay] int, -- Current day=1 else = 0
    [isDataAvailable] int, -- data available for the day = 1, no data available
    [isLatestDataAvailable] int
)

GO

DECLARE @StartDate DATETIME = '01/01/1990' --Starting value of Date Range
DECLARE @EndDate DATETIME = '01/01/2099' --End Value of Date Range
```

```

DECLARE
    @DayOfWeekInMonth INT,
    @DayOfWeekInYear INT,
    @DayOfQuarter INT,
    @WeekOfMonth INT,
    @CurrentYear INT,
    @CurrentMonth INT,
    @CurrentQuarter INT
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)

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

WHILE @CurrentDate < @EndDate
BEGIN

    IF @CurrentMonth != DATEPART(MM, @CurrentDate)
    BEGIN
        UPDATE @DayOfWeek
        SET MonthCount = 0
        SET @CurrentMonth = DATEPART(MM, @CurrentDate)
    END

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

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

    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)

INSERT INTO [dbo].[DimDate]
SELECT

    CONVERT (char(8),@CurrentDate,112) as DateKey,
    @CurrentDate AS Date,
    convert(DATE,@CurrentDate) AS DateSTD,
    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,
    DATEPART(DW, @CurrentDate) AS DayOfWeekUSA,

    -- check for day of week as Per US and change it as per UK format
    CASE DATEPART(DW, @CurrentDate)
        WHEN 1 THEN 7
        WHEN 2 THEN 1
        WHEN 3 THEN 2
        WHEN 4 THEN 3
        WHEN 5 THEN 4
        WHEN 6 THEN 5
        WHEN 7 THEN 6
    END
    AS DayOfWeekUK,

    @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(YEAR, @CurrentDate)) AS MonthYear,
    RIGHT('0' + CONVERT(VARCHAR, DATEPART(MM, @CurrentDate)), 2) +
    CONVERT(VARCHAR, DATEPART(YEAR, @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(YEAR,
    @CurrentDate))) AS FirstDayOfYear,
    CONVERT(DATETIME, '12/31/' + CONVERT(VARCHAR, DATEPART(YEAR,
    @CurrentDate))) AS LastDayOfYear,
    NULL AS IsHolidayNet,
    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 HolidayNet, (case when @CurrentDate = convert(date, sysdatetime())
then 1 else 0 end), 0, 0

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

```

Step 6: ETL development – Accumulating fact tables

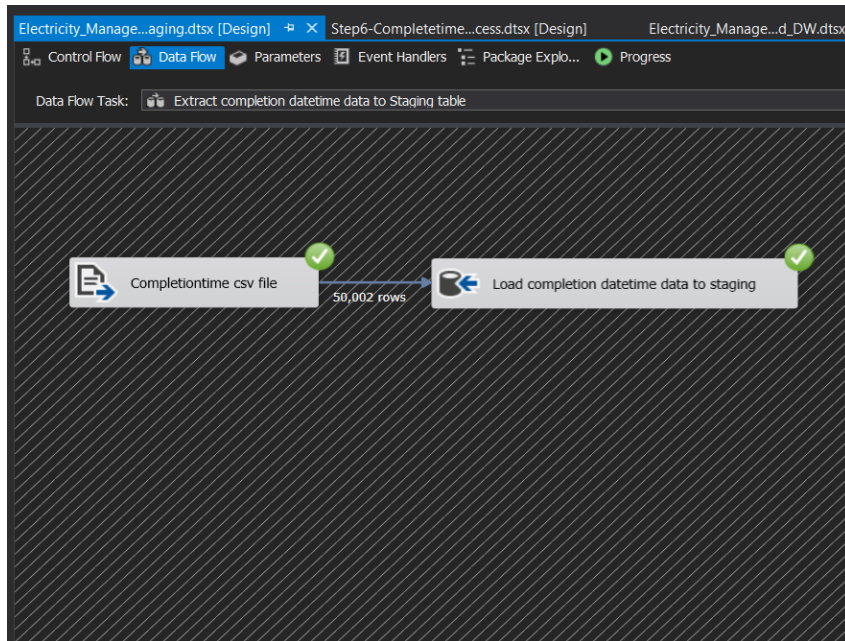
The fact table was extended by adding 3 new columns as shown below.

TENORB.Electricit...ConsumptionCharges			
	Column Name	Data Type	Allow Nulls
▶	consumptionID	int	<input checked="" type="checkbox"/>
	ConsumptionChargeDateKey	int	<input checked="" type="checkbox"/>
	ConsumerKey	int	<input checked="" type="checkbox"/>
	installFee	float	<input checked="" type="checkbox"/>
	unitKey	int	<input checked="" type="checkbox"/>
	unitPrice	float	<input checked="" type="checkbox"/>
	DistributionKey	int	<input checked="" type="checkbox"/>
	SmartMeterPerc	decimal(10, 2)	<input checked="" type="checkbox"/>
	NoOfUnit	decimal(10, 2)	<input checked="" type="checkbox"/>
	entrant	tinyint	<input checked="" type="checkbox"/>
	totalPrice	float	<input checked="" type="checkbox"/>
	subTotal	float	<input checked="" type="checkbox"/>
	InsertDate	datetime	<input checked="" type="checkbox"/>
	ModifiedDate	datetime	<input checked="" type="checkbox"/>
	accm_txn_create_time	datetime	<input checked="" type="checkbox"/>
	accm_txn_complete_time	datetime	<input checked="" type="checkbox"/>
	txn_process_time_hours	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Then a separate data source (csv file) named Completetime.csv was created and the structure of this file is shown below

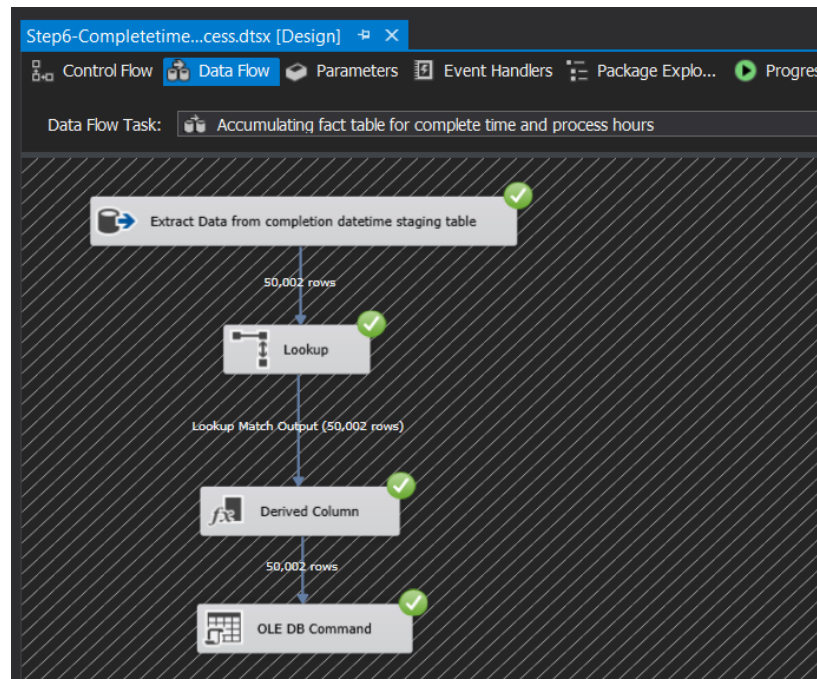
ConsumptionID	completedatetime
1	18/05/2022 04:10
2	17/05/2022 13:19
3	18/05/2022 00:09
4	18/05/2022 23:53
5	17/05/2022 12:33
6	18/05/2022 18:28
7	18/05/2022 18:29
8	19/05/2022 17:52
9	20/05/2022 00:09
10	19/05/2022 07:39
11	17/05/2022 22:16

Then, the data from excel file was loaded to the stgCompletionDatetime staging table



Data from the Completion Time staging table extracted and lookup the fact table “FactPowerConsumptionCharges” using the fields consumptionID field.

Finally, the after derived the txn_process_time_hours column data was loaded to the FactPowerConsumptionCharges fact table in the data warehouse.



A Derived Columns task has been used to derive the values for txn_process_time_hours column by getting the date difference of accn_txn_complete_time & accn_txn_create_time.

Derived Column Name	Derived Column	Expression	Data Type
accm_txn_complete_t...	Replace 'accm_txn_co...	completedatetime	database timestam
txn_process_time_ho...	Replace 'txn_process_t...	DATEDIFF("hh",accm_txn_create_time,accm_txn_com...	four-byte signed int

<>

Configure Error Output...

OK

Cancel

Help