HÖGSKOLAN DALARNA

Implementing a Data Warehouse Integrated in a Full BI Flow

Authors

Baranasuriya Patabendige Nadiranga Lakruwan - v25nlbpa@du.se

Contents

Architecture of the Data Warehouse and BI Flow

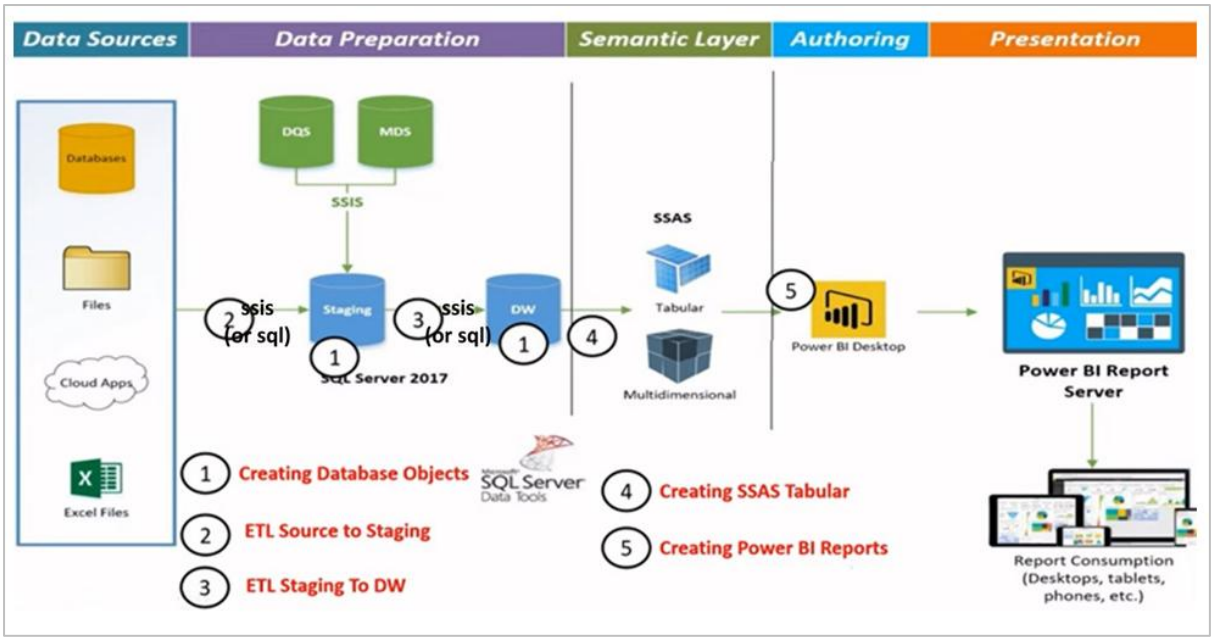


Figure 1 Architecture of the data warehouse and BI flow

The diagram illustrates the overall architecture of the Data Warehouse and Business Intelligence (BI) flow implemented in this project. The architecture is organized into five main stages: Data Sources, Data Preparation, Semantic Layer, Authoring, and Presentation. Each stage plays a critical role in transforming raw data into meaningful business insights.

Data Sources

Data Sources represent the origin of the data used in the BI system. Data can be collected from multiple types of sources, such as relational databases (e.g., SQL Server, Oracle), flat files (CSV, Excel), and cloud-based applications. These sources contain raw transactional data that is not yet suitable for analytical purposes.

Data Preparation

Data Preparation is responsible for processing and organizing raw data. In this stage, data is extracted from the source systems and loaded into a temporary staging area using ETL tools such as SQL Server Integration Services (SSIS). The ETL process consists of three main steps:

- Extract: Retrieve raw data from source systems.
- Transform: Clean, standardize, and prepare the data.
- Load: Store the transformed data in the Data Warehouse for long-term storage and analysis.

The staging area acts as an intermediate layer that enables data validation and transformation before loading it into the Data Warehouse.

Semantic Layer

The Semantic Layer provides a business-friendly view of the data. In this project, SQL Server Analysis Services (SSAS) is used to create a Tabular Model on top of the Data Warehouse. The semantic model organizes data into dimensions, measures, and relationships, allowing users to easily analyze data without dealing with complex database structures. Compared to multidimensional models, tabular models are simpler and more efficient for most analytical scenarios.

Figure 1 illustrates the overall architecture of the Data Warehouse and BI data flow.

Authoring

Authoring refers to the creation of interactive reports and dashboards using Power BI Desktop. At this stage, analytical models created in SSAS are used to build visualizations such as charts, tables, slicers, and KPIs. These visual elements help users explore and understand business data more effectively.

Presentation

The Presentation layer focuses on sharing and consuming reports. Reports and dashboards are published to Power BI Service or Power BI Server, enabling users to access insights from different devices, including desktop computers, tablets, and mobile phones. This ensures that business users can access real-time analytics in a centralized and secure environment.

Summary of Implementation Steps

The implementation of the Data Warehouse and BI solution follows these main steps:

1. Creation of database objects for the staging area and Data Warehouse.
2. ETL process from source systems to the staging area.
3. ETL process from staging area to the Data Warehouse.
4. Creation of SSAS Tabular semantic models.
5. Development of Power BI reports and dashboards for end users.

This structured workflow ensures that raw transactional data is transformed into reliable and meaningful business insights.

Data Source Description

The AdventureWorks2014 database is a sample transactional database provided by Microsoft to simulate a real-world business environment. It represents a fictional bicycle manufacturing company and contains data related to sales, customers, products, employees, and resellers.

In this project, the AdventureWorks2014 database is used as the primary data source for building a Sales Data Mart within the Data Warehouse. The Sales Data Mart is modeled using a Galaxy Schema, which includes two fact tables:

- Internet Sales Fact: Captures sales transactions conducted through online channels.
- Reseller Sales Fact: Captures sales transactions made through resellers.

These fact tables are connected to shared dimension tables, enabling comprehensive sales analysis across multiple business perspectives.

Software applications to install

To successfully implement the Data Warehouse and Business Intelligence solution described in this project, several software applications were installed and configured. These tools support different stages of the BI workflow, including data storage, ETL processing, semantic modeling, and report creation.

Microsoft SQL Server

Microsoft SQL Server was installed to serve as the core database management system for this project. It was used to host the staging database, the data warehouse, and the source transactional database (AdventureWorks2014). SQL Server provides robust support for relational data storage, indexing, and query processing, which is essential for handling large volumes of business data.

SQL Server Management Studio (SSMS)

SQL Server Management Studio (SSMS) was installed as the primary administrative and development tool for SQL Server. SSMS was used to create database objects such as tables, views, stored procedures, and constraints. It also enabled querying and validating data at each stage of the ETL process, ensuring that data was correctly loaded into the staging area and the data warehouse.

SQL Server Integration Services (SSIS)

SQL Server Integration Services (SSIS) was installed and used to implement the ETL (Extract, Transform, Load) processes. SSIS packages were developed to extract data from the AdventureWorks2014 database, transform it according to business requirements, and load it into the staging and data warehouse databases. SSIS was also used to handle advanced ETL tasks such as data cleansing, data type conversion, and Slowly Changing Dimensions (SCD).

SQL Server Analysis Services (SSAS)

SQL Server Analysis Services (SSAS) was installed to create the Semantic Layer of the BI architecture. In this project, an SSAS Tabular model was developed to organize data from the data warehouse into dimensions and fact tables. Measures and calculations were defined within the model to support analytical queries. The SSAS model provides a centralized and reusable analytical layer for reporting tools.

Microsoft Visual Studio (with SSAS and SSIS extensions)

Microsoft Visual Studio was installed and configured with the necessary extensions to support SSIS and SSAS development. Visual Studio was used to create, edit, deploy, and process SSIS packages and SSAS Tabular models. It served as the main development environment for building the semantic model and managing ETL workflows.

Power BI Desktop

Power BI Desktop was installed to create interactive reports and dashboards. Power BI was connected to the SSAS Tabular model using a live connection, allowing the reuse of predefined measures and ensuring consistency in calculations. Power BI Desktop was used to design visualizations such as charts, tables, slicers, and maps for analyzing sales data.

Restore AdventureWorks2014 database backup file

The AdventureWorks2014 database backup file was downloaded from this link https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms and then Microsoft SQL Server Management Studio was opened with the connection of local SQL server. After successful connection with the local SQL server, right click on the Databases and select the 'Restore Database' option, then access the direction where the database backup file downloaded into the computer.

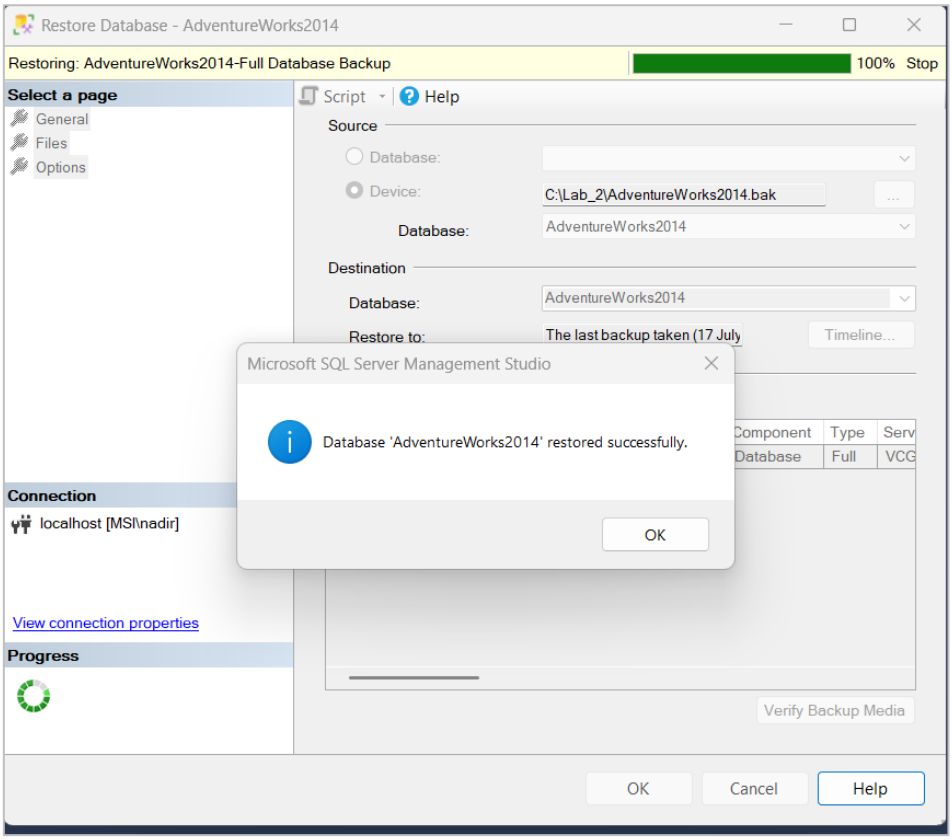


Figure 2 Restore Database backup file

Step 1: Creating Database Objects

Creating 'AWN_STG_Demo' Database

By using below SQL commands, 'AWN_STG_Demo' database was created with two schemas; 'erp' and 'hr', which can help to maintain the source system name.

```
USE [master]
GO
/****** Object:  Database [AWN_STG_Demo]    Script Date: 25-09-2021 13:26:13 ******/
CREATE DATABASE [AWN_STG_Demo]

GO
USE [AWN_STG_Demo]
GO
/****** Object:  Schema [erp]    Script Date: 25-09-2021 13:26:13 ******/
CREATE SCHEMA [erp]
GO
/****** Object:  Schema [hr]    Script Date: 25-09-2021 13:26:13 ******/
CREATE SCHEMA [hr]
GO
```

After that all relevant tables for above 'AWN_STG_Demo' database was created using given SQL script. Then refresh the Databases and it shows the 'AWN_STG_Demo' database as below image, after successful completed all SQL commands.

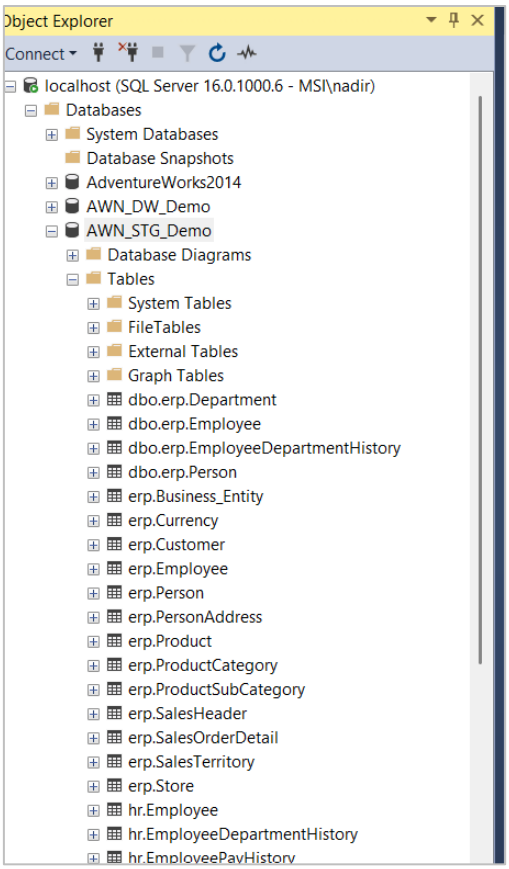


Figure 3 AWN_STG_Demo Database

Creating 'AWN_DW_Demo' Data Warehouse Database

By using below SQL commands, 'AWN_DW_Demo' database was created which was the actual database for this data warehouse project.

```
USE [master]
GO
/****** Object:  Database [AWN_DW_Demo]    Script Date: 25-09-2021 13:29:02 ******/
CREATE DATABASE [AWN_DW_Demo]

USE [AWN_DW_Demo]
GO
```

After that all relevant tables for above 'AWN_DW_Demo' database was created using given SQL script, until the table creation commands (Until line number 326). Then refresh the Databases and it shows the 'AWN_DW_Demo' database as below image, after successful completed all SQL commands.

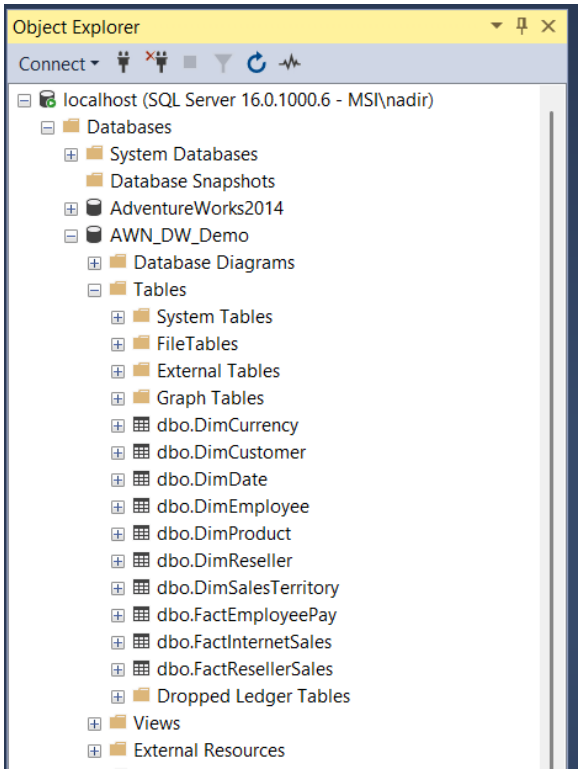


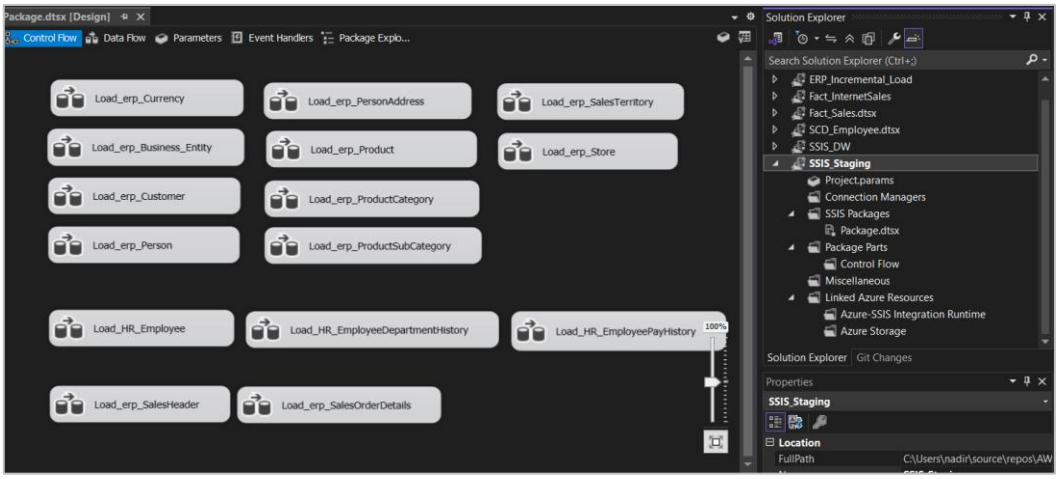
Figure 4 AWN_DW_Demo Database

Step 2: ETL process for Source Data to Staging Database

In this step, a new SQL Server Integration Services (SSIS) project was created using Microsoft Visual Studio. The project was named SSIS_Staging and was used to manage all data extraction and loading activities into the staging database.

As a prerequisite, OLE DB connections were configured through the Connection Manager to establish connectivity with the local SQL Server instance. These connections enabled access to both the AdventureWorks2014 source database and the AWN_STG_Demo staging database.

Within the SSIS_Staging project, a single SSIS package (.dtsx) was developed to handle multiple data loading processes. This package was designed to manage ERP data loading, HR data loading, as well as incremental data loading for the SalesOrderHeader and SalesOrderDetail tables. The use of a unified SSIS package simplified the ETL workflow by centralizing control flow and improving maintainability, while still allowing separate data flows for each data domain. The overall structure of the package is illustrated in the figure below.



For the ERP data, the SSIS package loads data into ten staging tables in the erp schema of the AWN_STG_Demo database:

1. erp.Currency
2. erp.Business_Entity
3. erp.Customer
4. erp.Person
5. erp.PersonAddress
6. erp.Product
7. erp.ProductCategory
8. erp.ProductSubCategory
9. erp.SalesTerritory
10. erp.Store

In addition, the same SSIS package includes Data Flow Tasks to perform a full load of HR data into three staging tables in the hr schema of the AWN_STG_Demo database:

1. hr.Employee
2. hr.EmployeeDepartmentHistory
3. hr.EmployeePayHistory

Load data to erp.Currency table

In the Control Flow tab of the SSIS package, a single Data Flow Task was added to load currency data into the staging database. This task was renamed Load_erp_currency to clearly indicate its purpose. The Data Flow Task is responsible for extracting currency data from the source system and loading it into the erp.Currency table in the AWN_STG_Demo database, as illustrated in the figure below

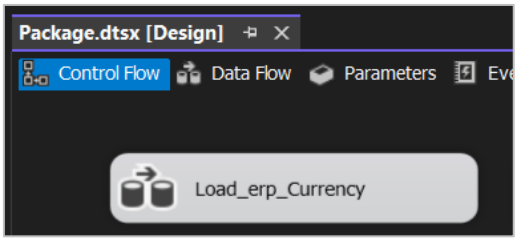


Figure 5 Control Flow for erp.Currency

Inside the Data Flow Task OLE DB Source, OLE DB Destination and Derived Column task were used as shown below in the image.

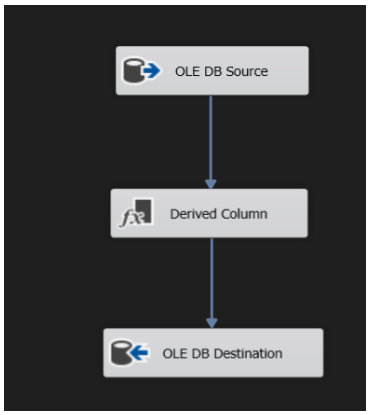


Figure 6 Data flow for erp.Currency

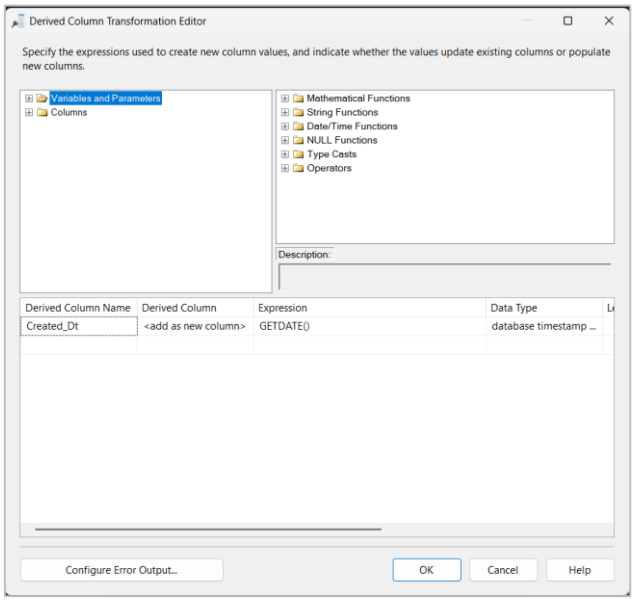


Figure 7 Derived Columns for Currency Data

New technical column ‘Created_Dt’ was created in the Currency table in staging database using this ‘Derived Column’ task for troubleshooting purpose. The configuration of the connection manager at ‘OLE DB Source Editor’, the ‘AdventureWorks2014’ database connection was selected and ‘Sales.Currency’ table was selected as the source table.

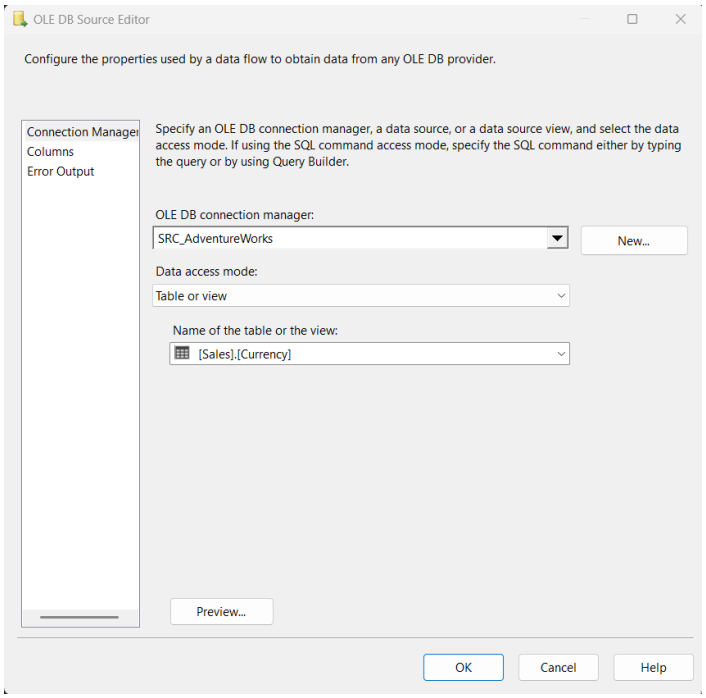


Figure 8 erp.Currency OLE DB Source

The configuration of the connection manager at ‘OLE DB Destination Editor’, the ‘AWN_STG_Demo’ database connection was selected and ‘erp.Currency’ table was selected as the destination table. After that all the input column from the source table and derived columns were mapped with destination columns using the ‘Mappings’ option.

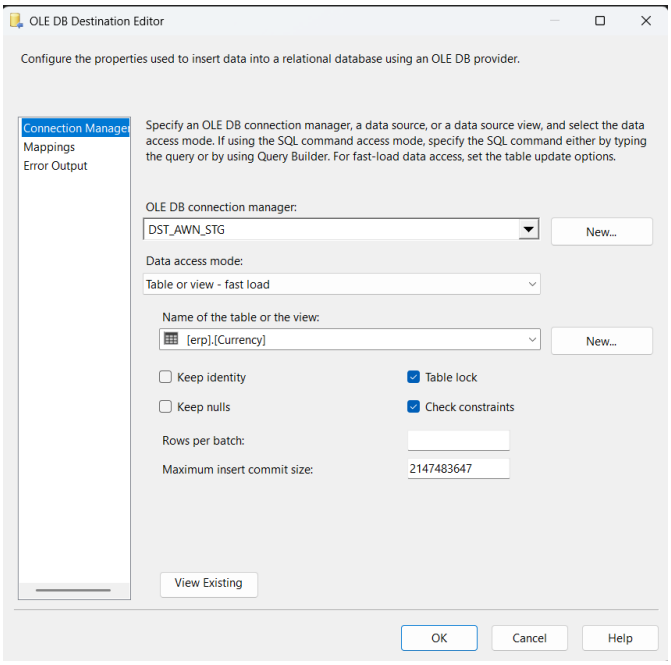


Figure 9 erp.Currency OLE DB Destination

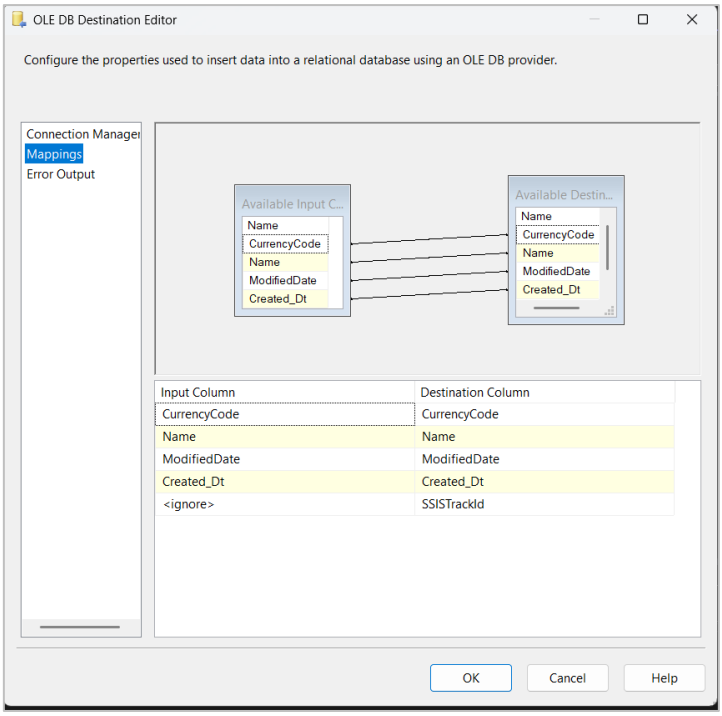


Figure 10 erp.Currency OLE DB Destination mappings

Finally, this control flow was executed and check the data population of the 'erp.Currency' table to verify that task is working correctly as expected.

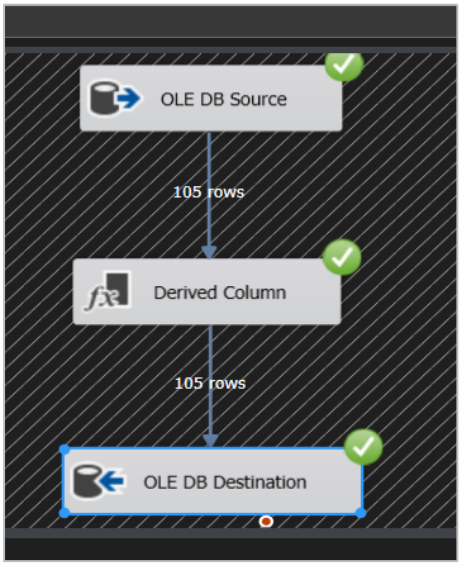


Figure 12 Execution of currency data

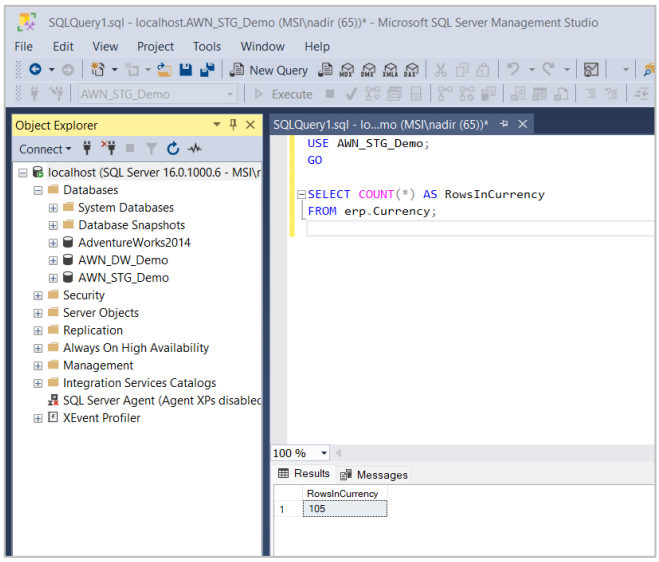


Figure 11 erp.Currency table data

Load data to erp.PersonAddress table

The data flow task for the 'erp.PersonAddress' was configured as below image, due to the 'erp.PersonAddress' data consist with two data sources; 'BusinessEntityAddress' and 'Address' tables. Therefore 'Merge Join Transformation' was used here to join two table data with '**Inner Join'** join type.

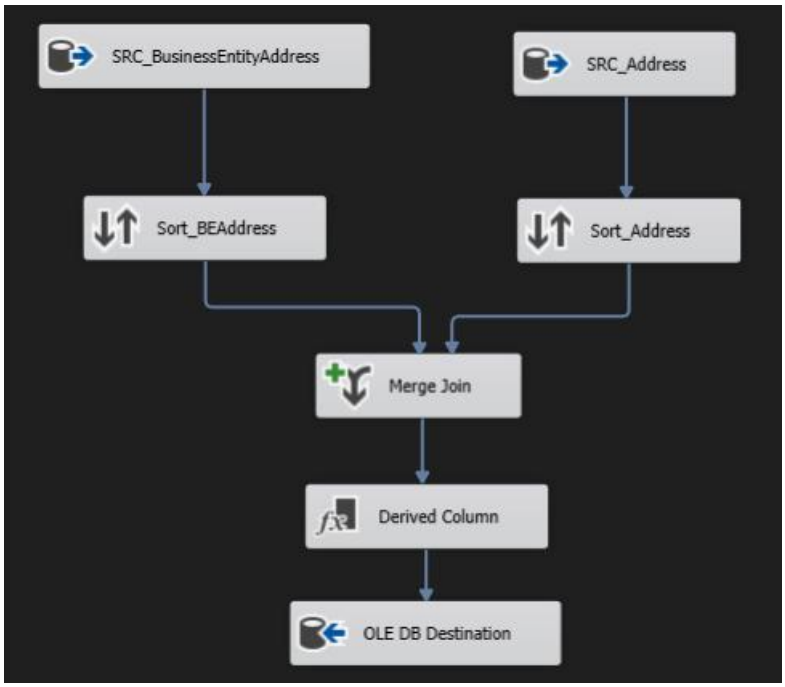


Figure 13 Data Flow Task for erp.PersonAddress

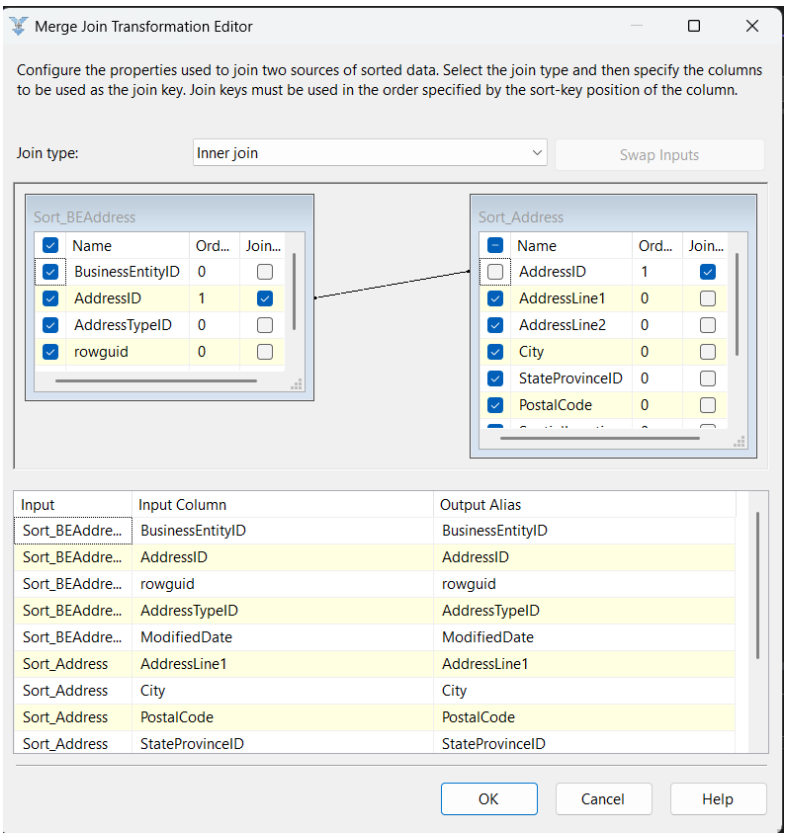


Figure 14 Inner join for erp.PersonAddress

Same configurations were repeatedly done for other ‘erp’ tables and ‘hr’ tables.

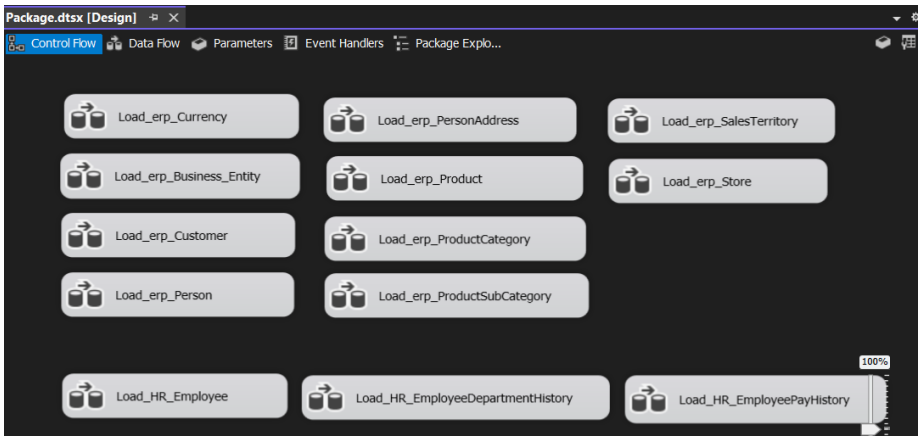


Figure 15 SSIS Package

ERP_Incremental_Load SSIS Package

The ‘SalesHeaderData’ and ‘SalesOrderDetailsData’ were consist of huge data, so it was time consuming to pull huge data from the source tables to target tables. So, data flow task for these tables were configured as below image.

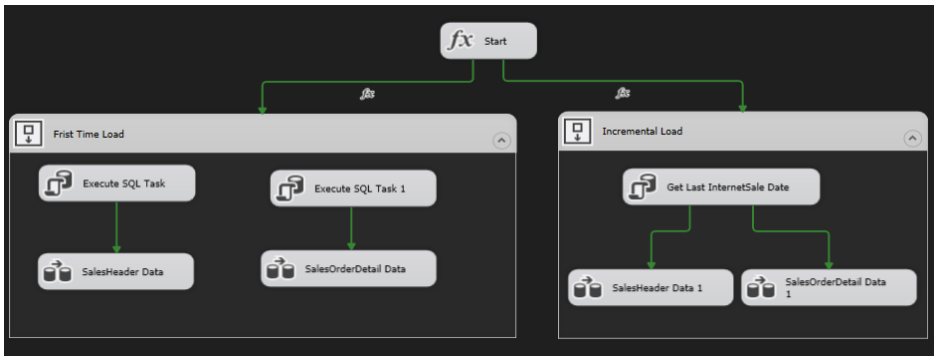


Figure 16 Data flow task for incremental load

The First-time load sequence was same configuration as previous tables. But incremental load start with the ‘Execute SQL Task’ to get the last ‘order Date’ form the ‘dbo.FactInternetSales’

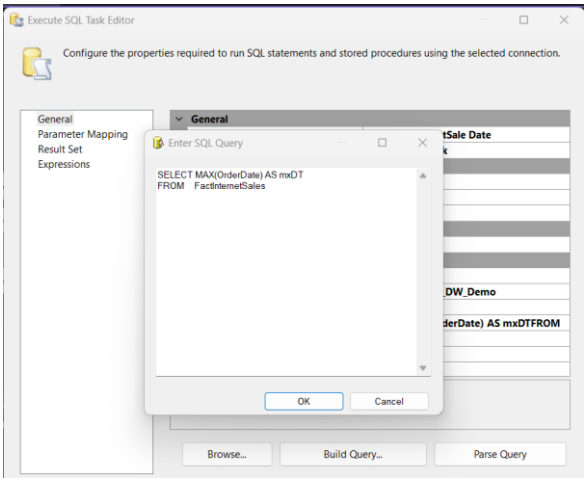


Figure 17 SQL task for get last Order Date

This last order date was stored as ‘Result Set’ in the ‘Execute SQL Task Editor’ which named as ‘mcDT’.

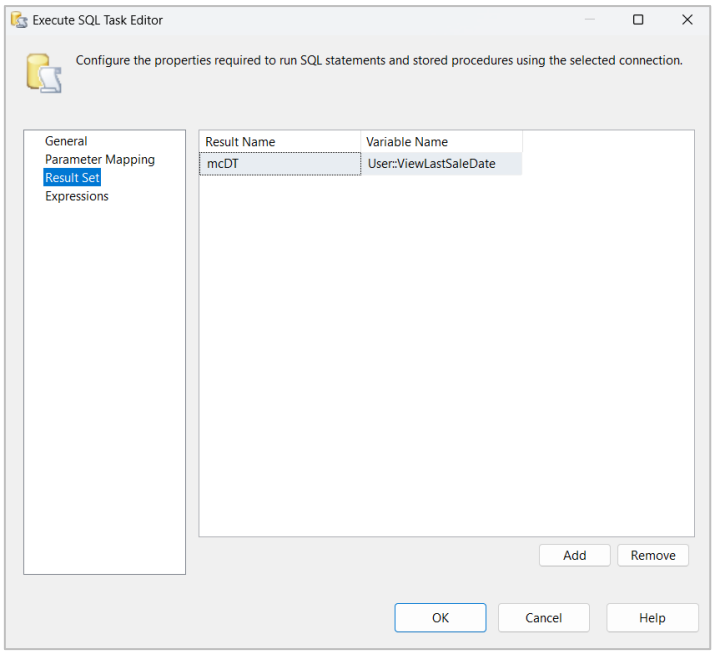


Figure 18 Result set for incremental loading

The ‘SalesOrderHeader’ data source was configured with the SQL command to pull only the data greater than last order date as shown in the below image.

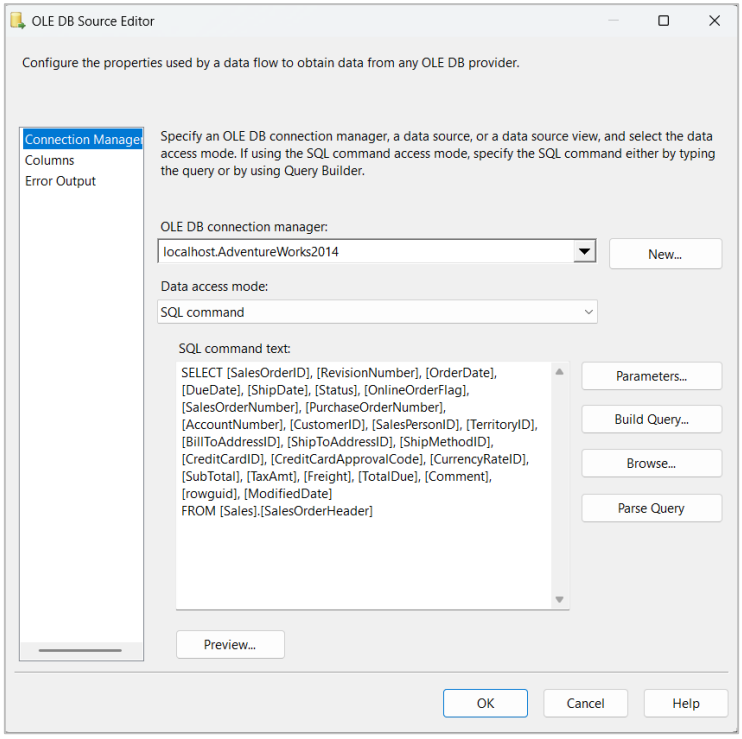


Figure 19 SQL command for SalesOrderHeader

The ‘SalesOrderDetails’ data source was configured with the SQL command to pull only the data greater than last order date as shown in the below image.

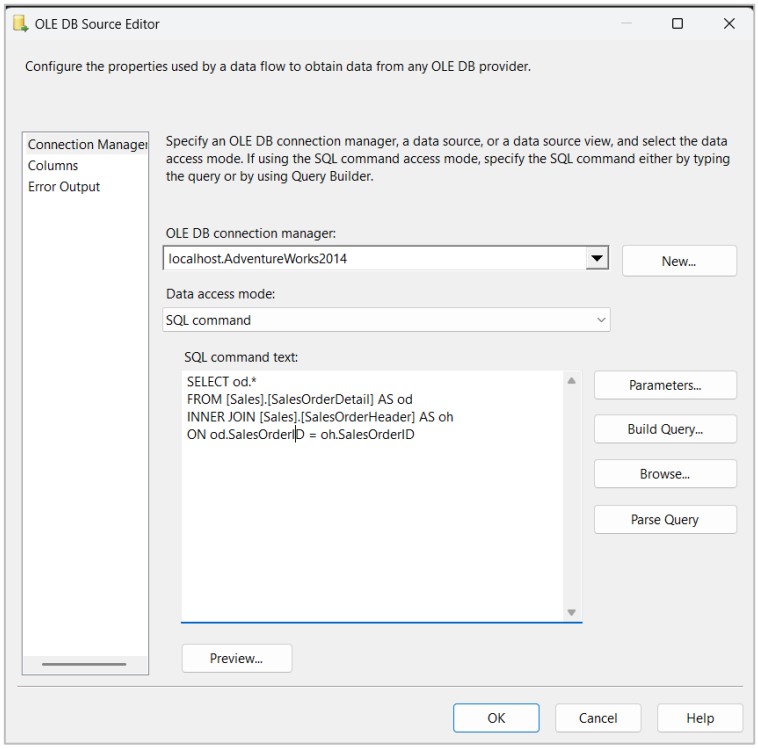


Figure 20 SQL command for SalesOrderDetails

A one project parameter was set to define the data load condition for first time load or incremental load which parameter name set as ‘FirstTimeLoad’. After that using the ‘Expression Task’, the expression in the ‘Precedence Constraint Editor’ was created using that ‘FirstTimeLoad’ parameter. So, if this parameter set to zero, the SSIS package execute the ‘First Time Load’ sequence task only. If this parameter set to one, the SSIS package execute the ‘Incremental Load’ sequence task only.

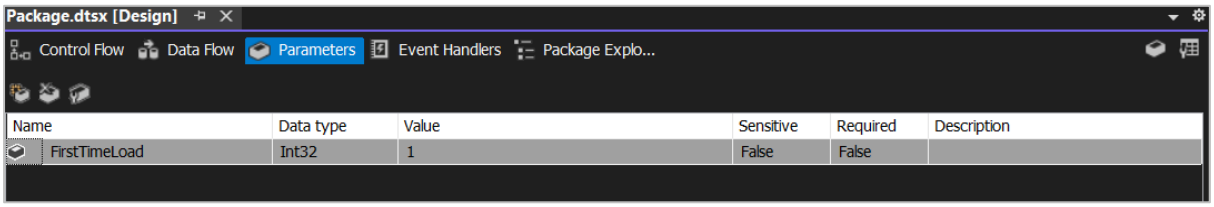


Figure 21 Project parameter for incremental load

Finally, these all SSIS packages were executed and check all the table data in ‘AWN_STG_Demo’ database to verify that step 2 is working correctly as expected.

Step 3: ETL Process for Staging Data to Data Warehouse

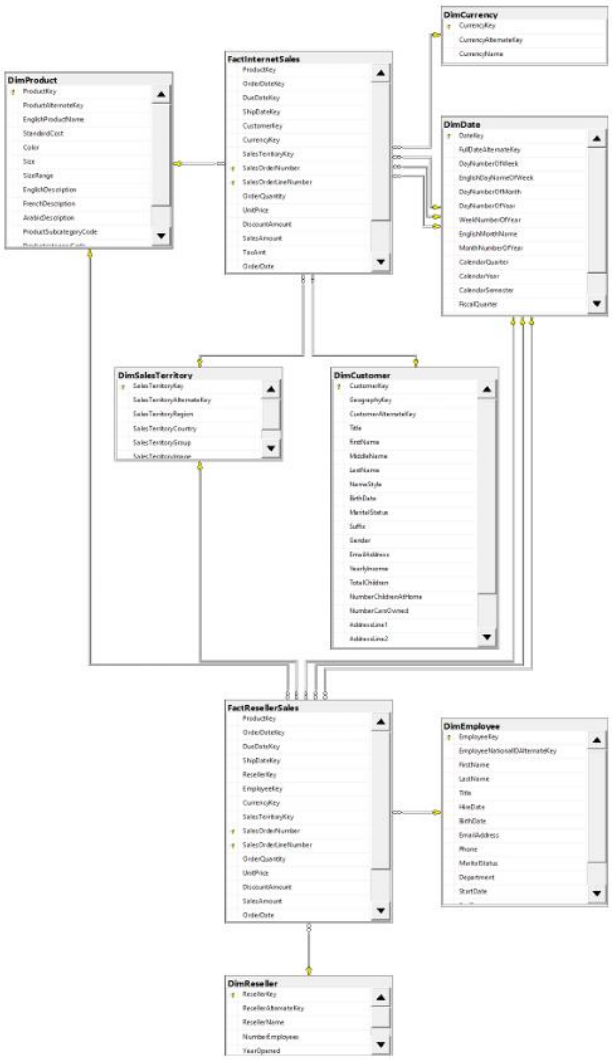


Figure 22 Star schema for AWN_DW_Demo database

As the first step it should be populate data for the dimension tables, because fact tables contain foreign key reference of these dimension tables. So, primary keys in these dimension tables should be populate first. By using the given SQL script, below 7 procedures were created, and these procedures were stored at the 'Stored Procedures' folder at 'Programmability' in 'AWN_DW_Demo' database.

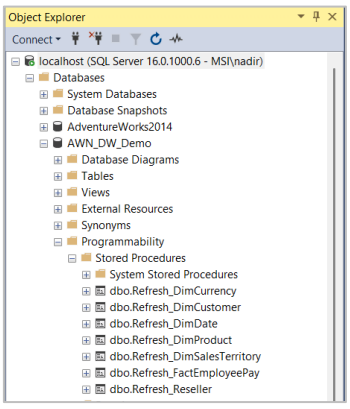


Figure 23 Procedures for populate data for Dim tables

A new integration service projects named ‘SSIS_DW’, ‘Fact_Sales’ and ‘SCD_Employess’ was created to populate data for ‘AWN_DW_Demo’ database tables as shown in the below image. ‘SSIS_DW’ was used to call above 6 procedures for Sales data using ‘Execute SQL Tasks’.

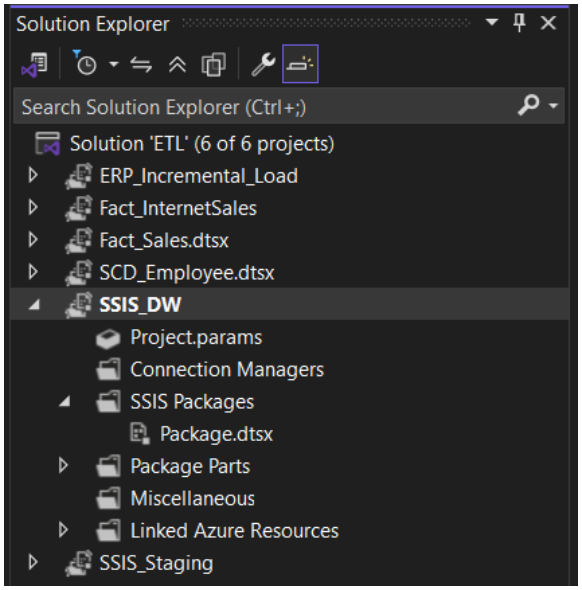


Figure 24 SSIS project for Data warehouse

SSIS_DW (Procedures)

In here, 6 SQL task were created and made connection with the ‘AWN_DW_Demo’ database and configure a SQL statement to call the relevant procedure as shown in the below image.

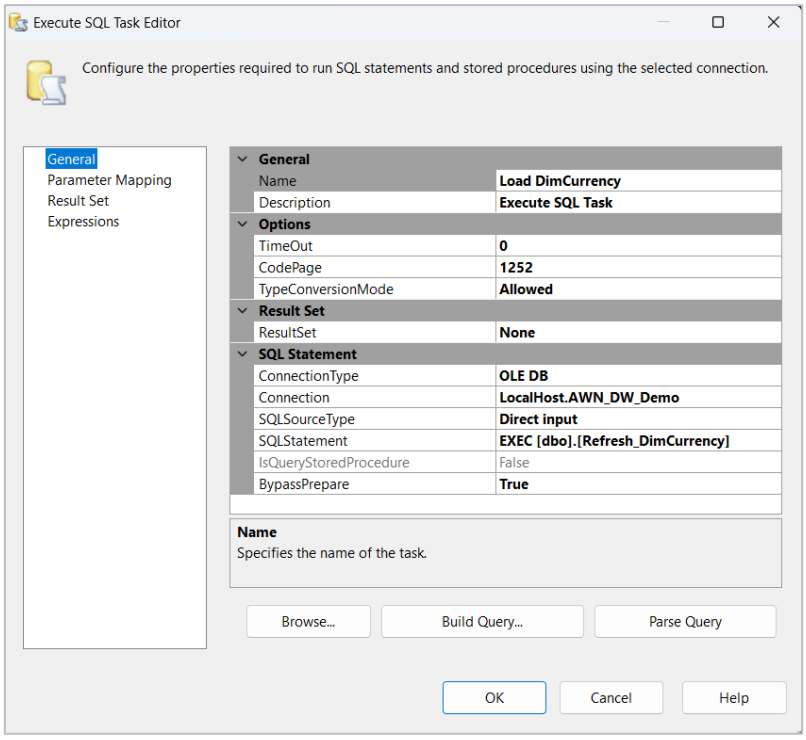


Figure 25 SQL task for call procedures

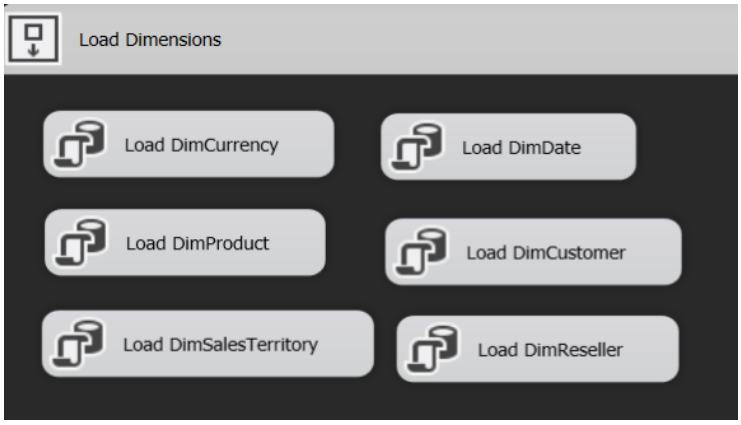


Figure 26 Populate data to 6 Dim tables

SCD_Employee (Slowly changing dimension)

The 'dim.Employee' has to be populate differently, because it needs to maintain history records for Employee 'Title' and 'Department', since those values will change time to time. The 'SCD_Employee' SSIS packge was created as shown in the below image with one 'Data Flow Task' and 'Execute SQL Task'. A one 'Slowly Changing Dimension' wizard was configured inside this 'Data Flow Task' to achieve this goal.

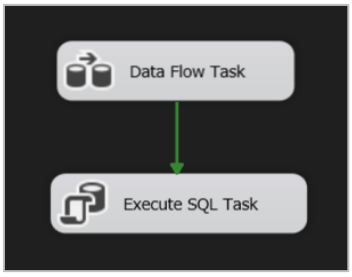


Figure 27 SCD_Employee SSIS Package

As the first step for slowly changing dimension configurations, the 'AWN_DW_Demo' database connection was selected and then 'dbo.DimEmployee' table was selected as the dimension table. After that, 'NationalIDNumber' was selected as the 'Business Key'.

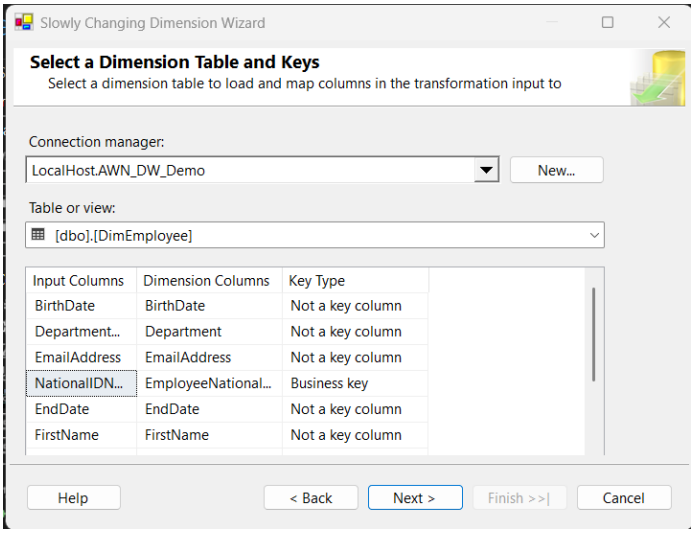


Figure 28 Selecting dim table and business key for SCD

Next in the ‘Slowly Changing Dimension Columns’ window, the change type of the ‘Title’ and ‘Department’ was set as ‘Historical attribute’ while ‘MaritalStatus’ set as ‘Changing attribute’.

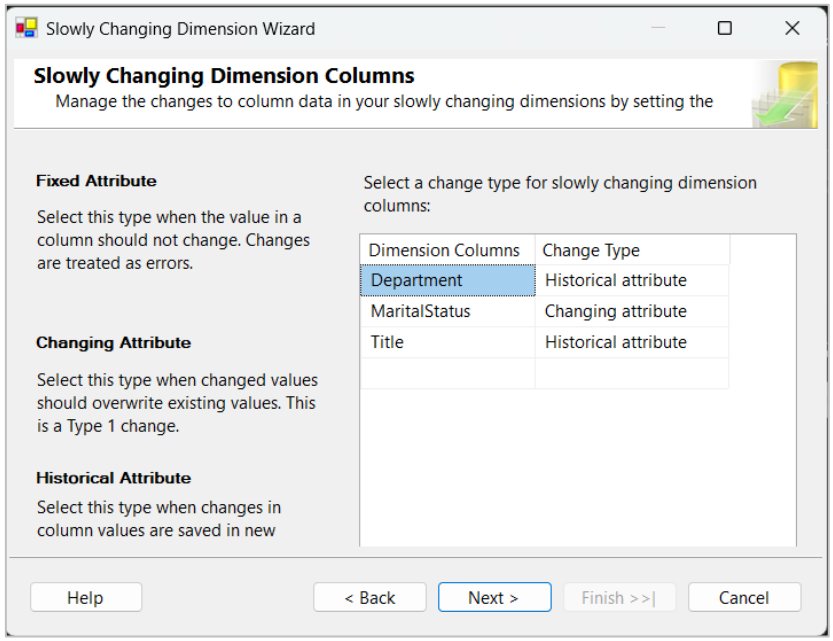


Figure 29 Selecting Historical and Changing attributes

In the next step of ‘Historical Attribute Options’ window, the start date and the end date were selected to identify the current and expired records.

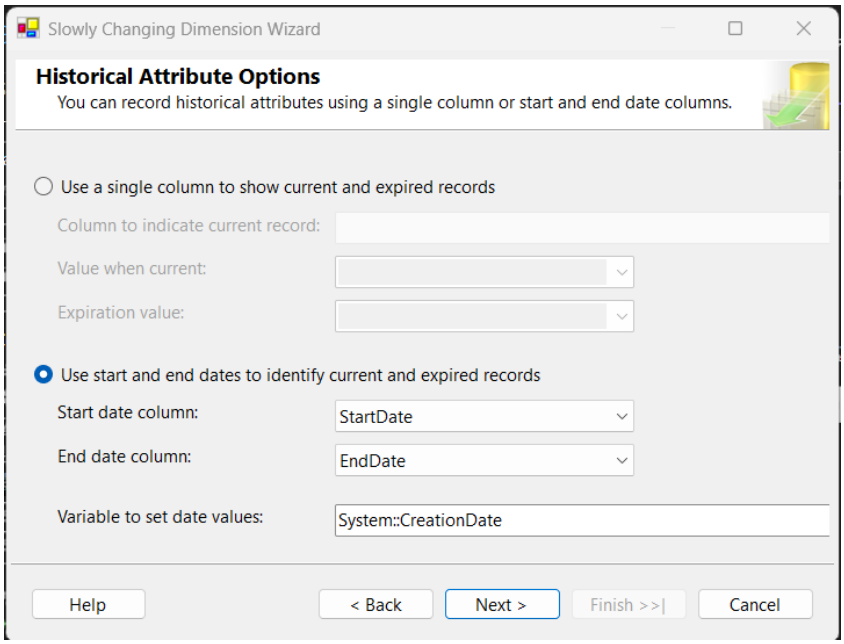


Figure 30 Historical attribute options

Finally, at the end of the configuration of slowly changing dimension task, it will automatically create other sub flow tasks as shown in the below image.

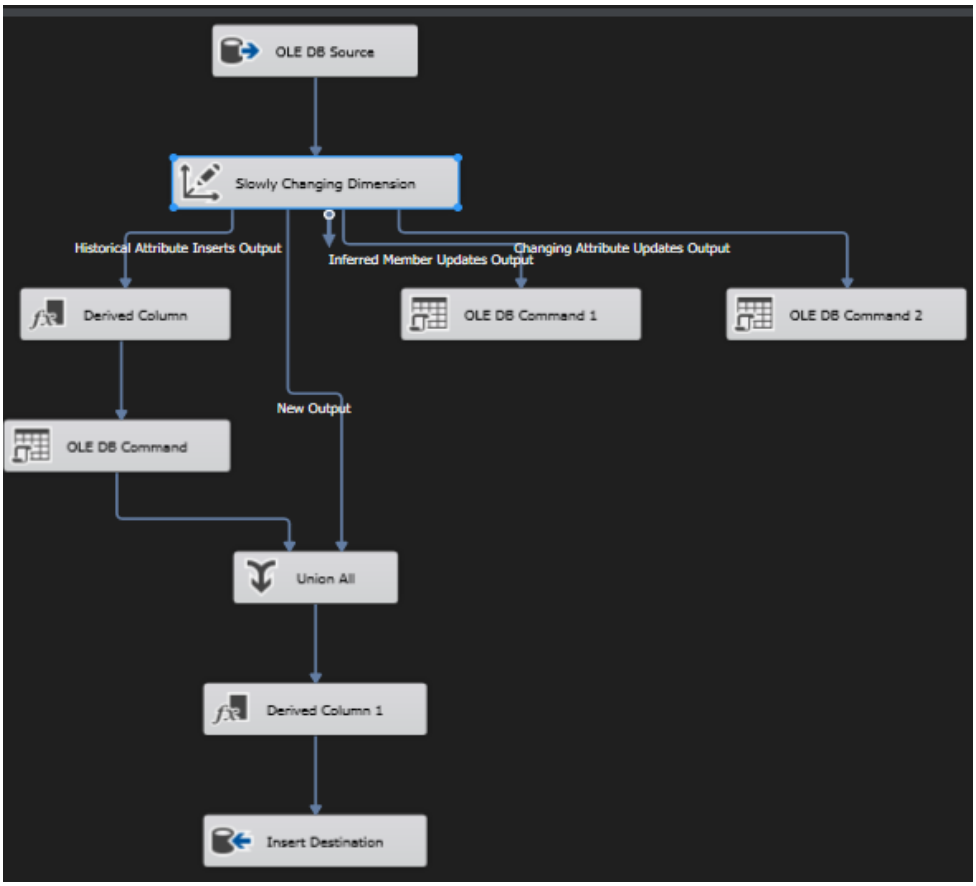


Figure 31 Complete flow of the slowly changing dimension task

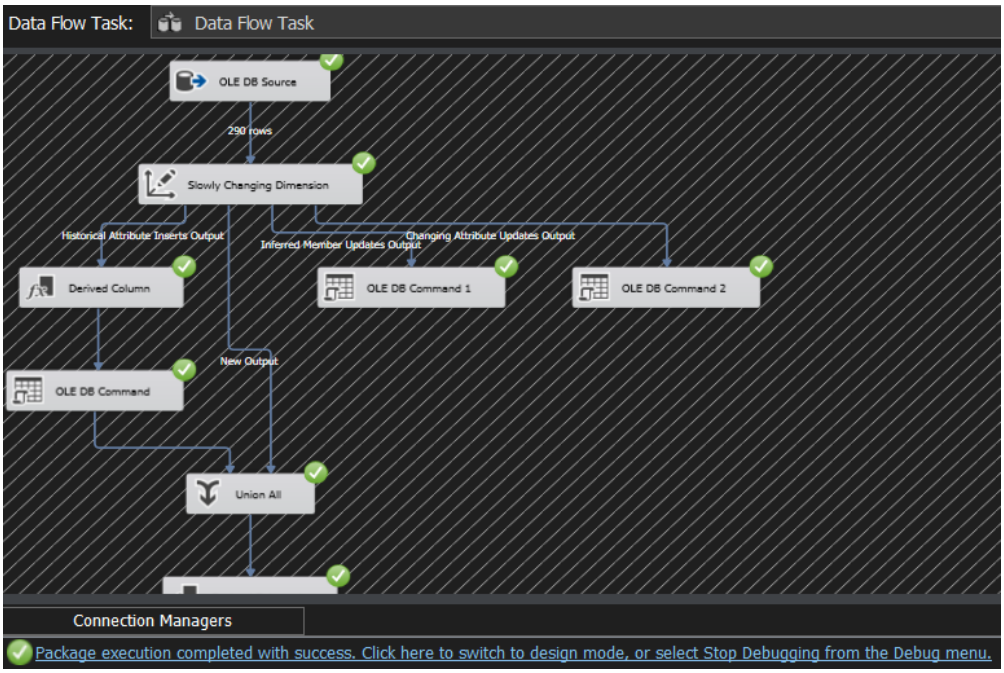


Figure 32 Execute of the slowly changing dimension task

Fact_InternetSales

This SSIS project was created to populate data to the fact table 'FactInternetSales'.

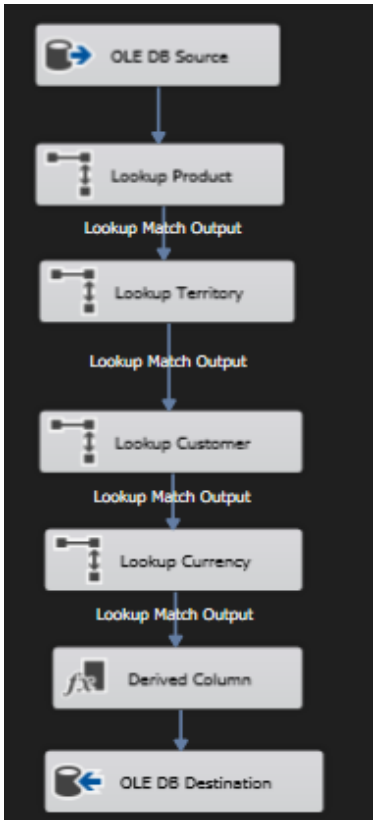


Figure 34 FactInternetSales Data Flow

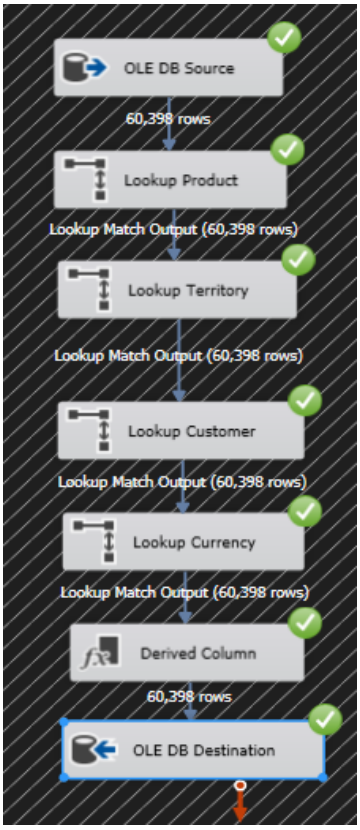


Figure 33 Execute Data Flow

All these input columns were mapped to the 'FactInternetSales' destination columns.

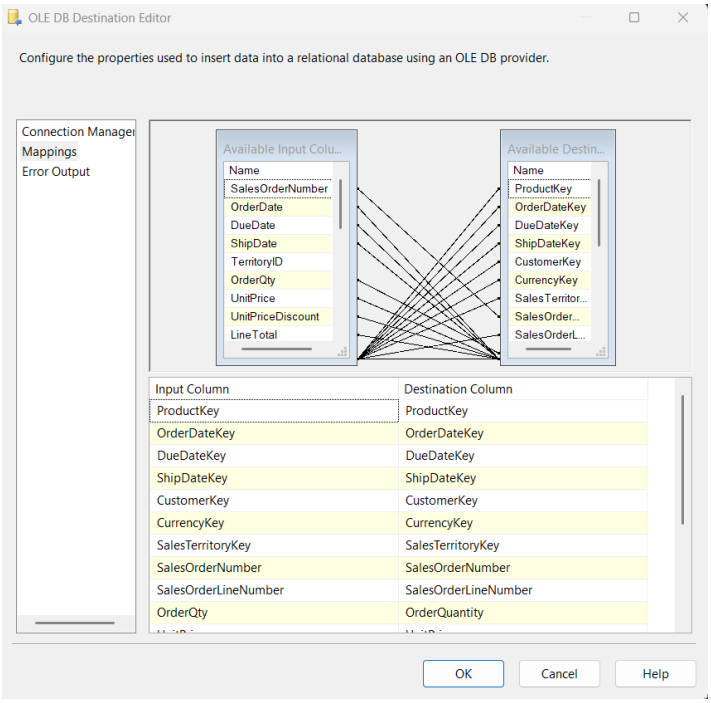


Figure 35 FactInternetSales columns mappings

Fact_Sales

In this SSIS project also configurations for this task were same as previous data load.

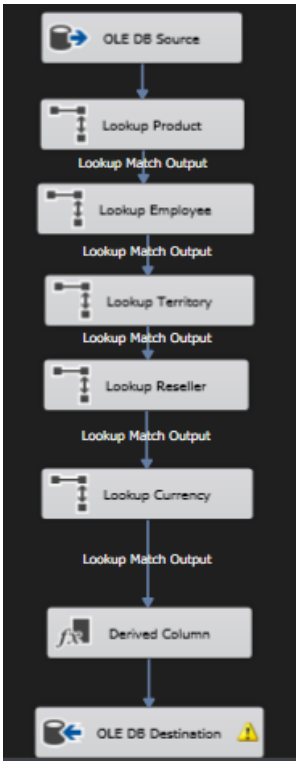


Figure 36 FactSales Data Flow

Step 4: Creating SSAS Tabular

To implement a tabular cube, a new analysis services tabular project named 'ADW_Analysis' was created in the Visual Studio.

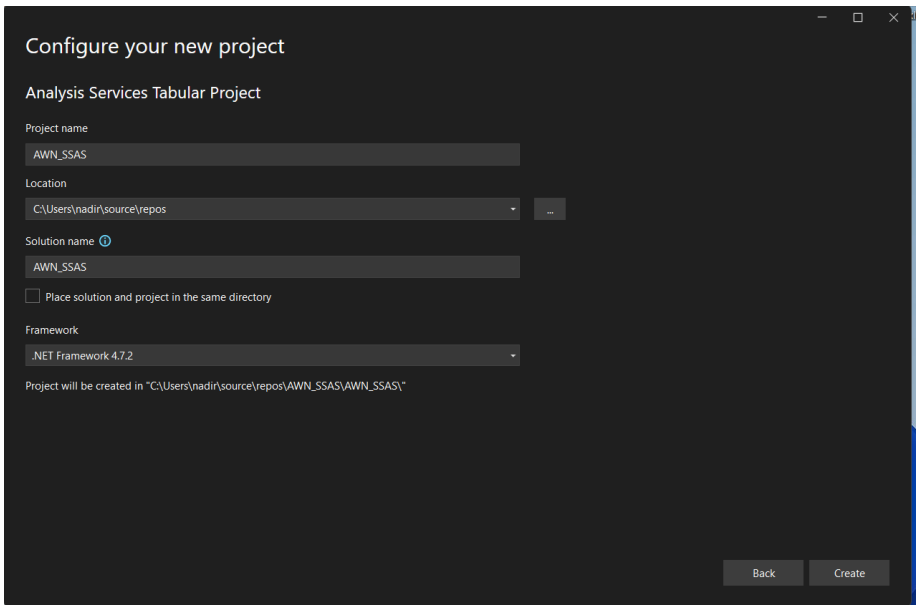


Figure 37 Create a new SSAS project

In the next step, current local server was selected as the workspace server and create the SSAS project.

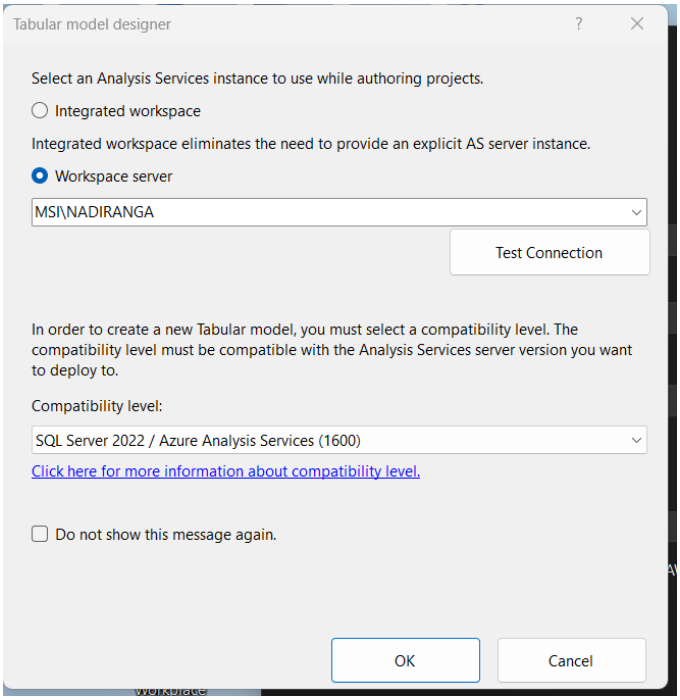
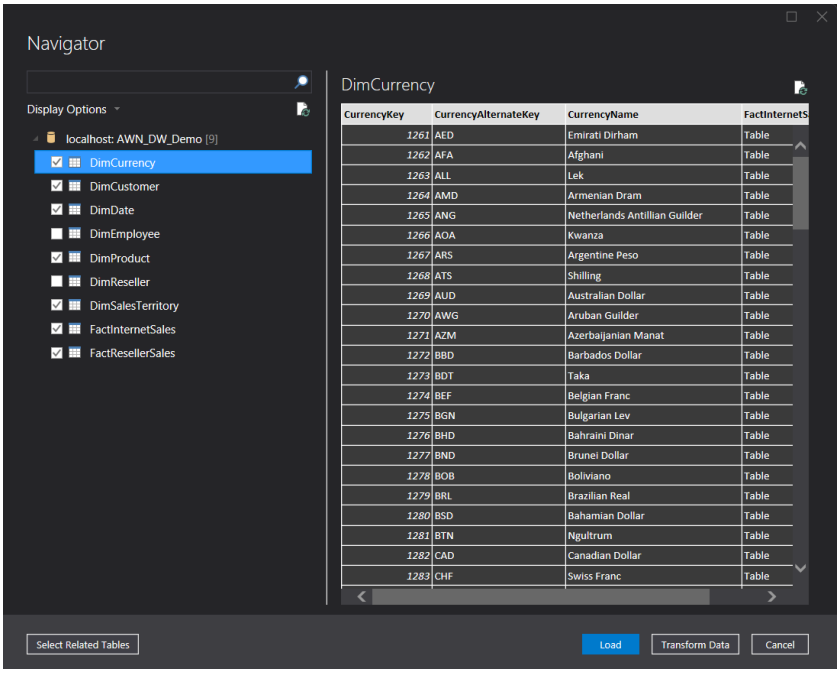


Figure 38 Tabular model designer workspace server selection

After that relevant tables were selected to create a model for sales data mart.



The Power BI report was created using a Live Connection to the SSAS Tabular model. Multiple interactive visualizations were designed to analyze sales performance across different time periods, product categories, and geographical regions. Slicers were added to allow dynamic filtering, enabling users to explore Internet and Reseller sales effectively

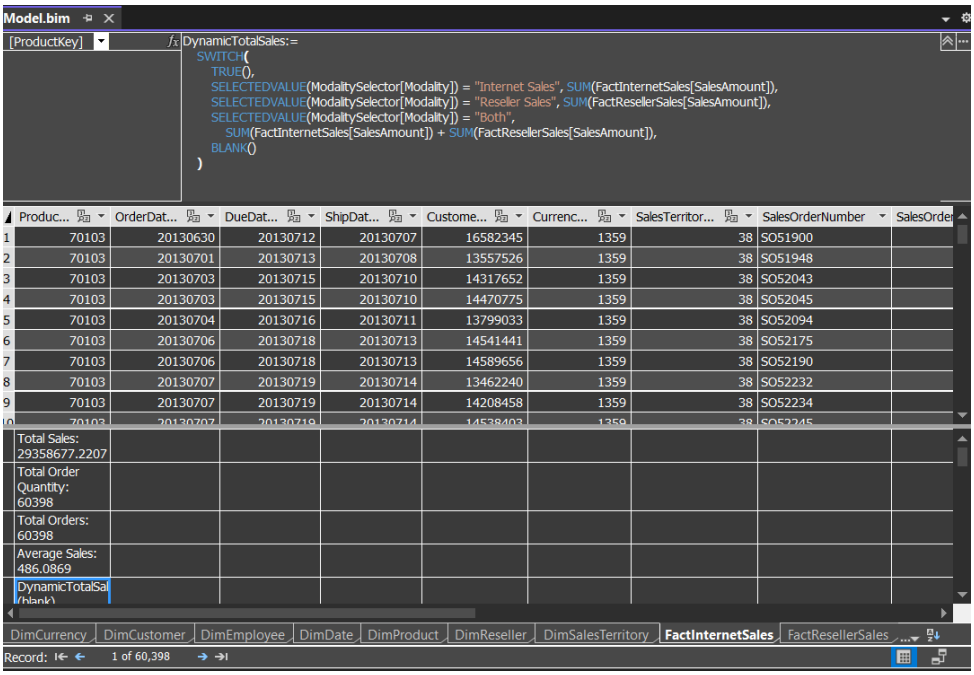


Figure 39 SSAS Tabular Added New Measures

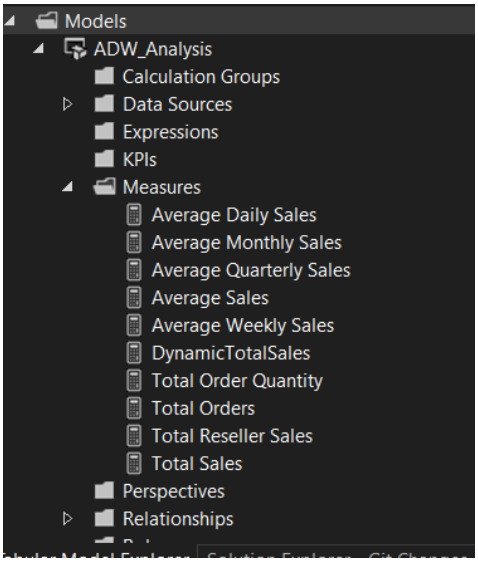


Figure 40 Measures in Model.bim

Star Schema for Sales Data Mart

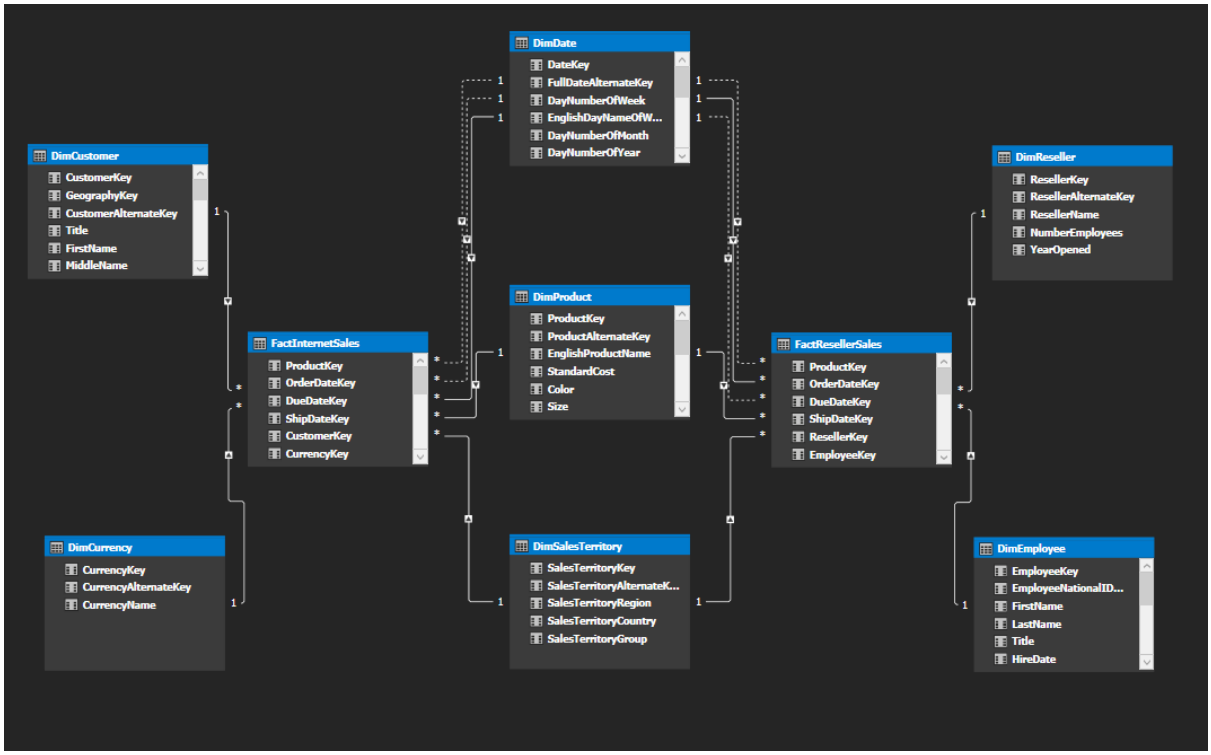


Figure 41 Star schema for sales data mart

Finally, this SSAS project was build and deployed to use in the power BI report.

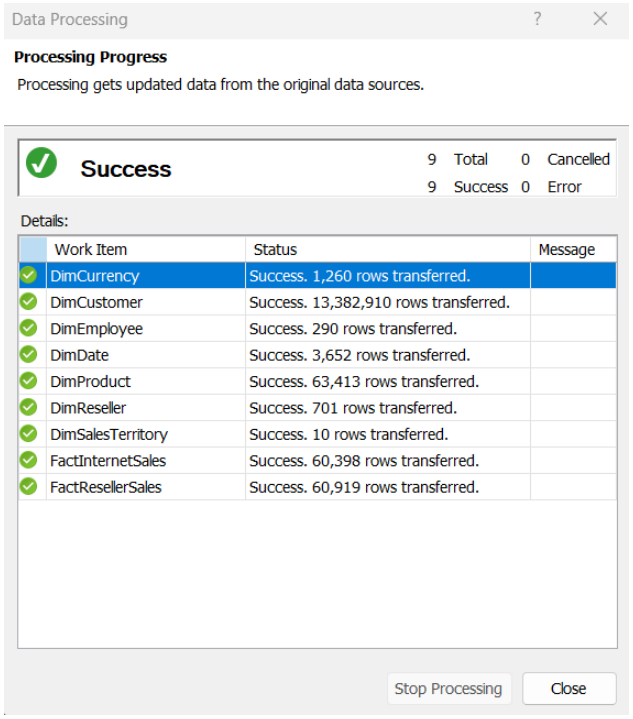


Figure 42 Deploying the SSAS project

Step 5: Creating Power BI Reports

A new Power BI 'Get Data – SQL server analysis services' database project was created and connected with current local SQL analysis server to connect with the model that created for sales data mart. Finally, a 'AdventureWork' Power BI sales report was created using the dim tables and fact tables using appropriate visualization and measures.

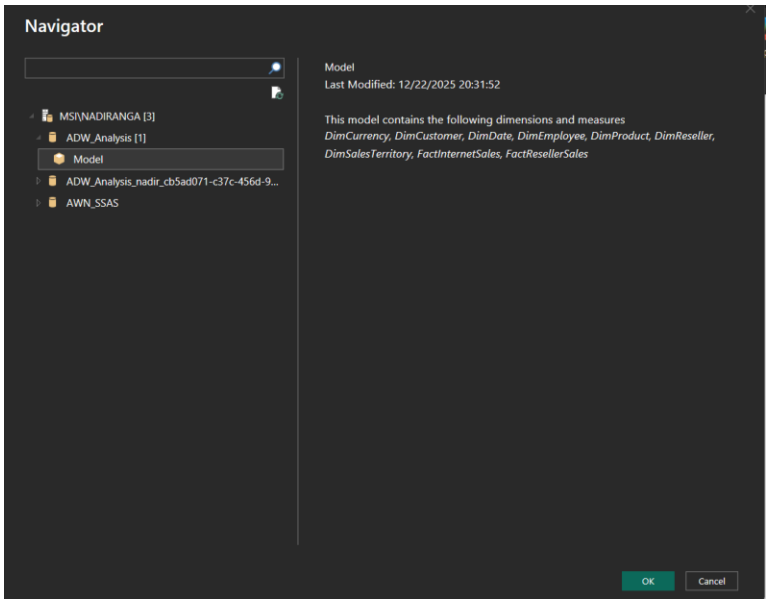


Figure 43 Select the model created for sales data mart

'AdventureWork' Power BI Sales Report

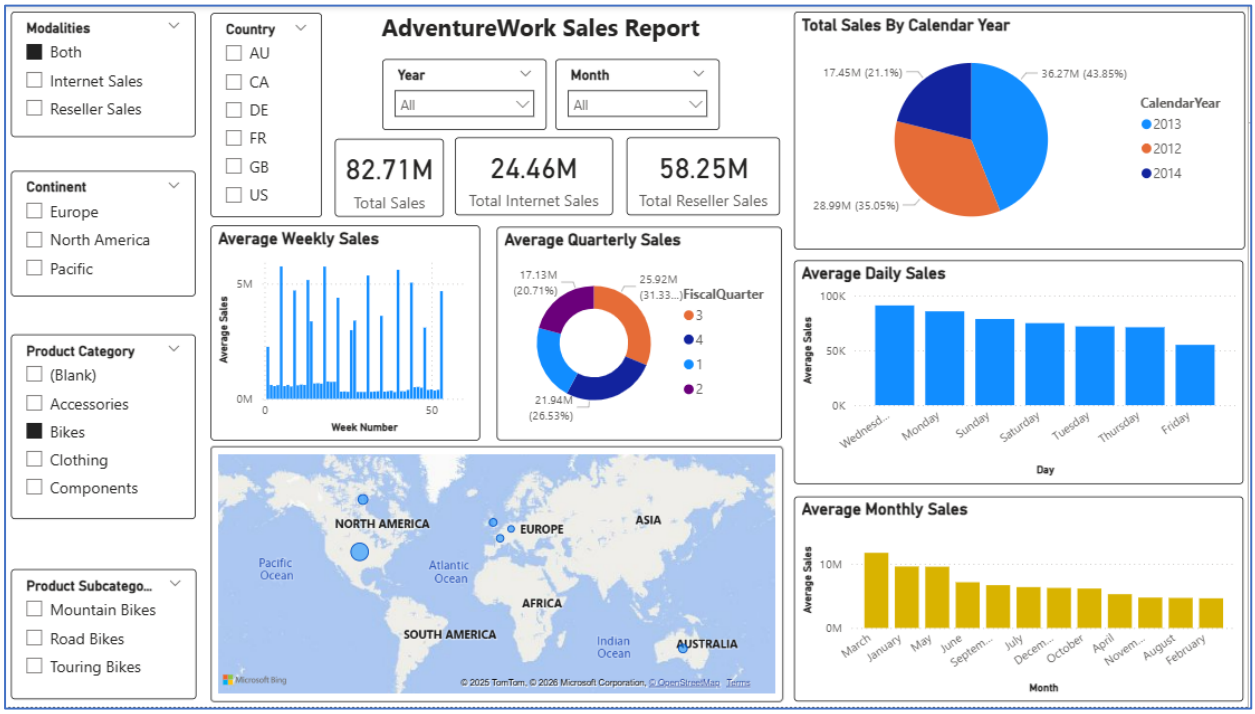


Figure 44 AdventureWork Power BI Sales Report