Implementing a Data Warehouse integrated in a full BI flow

Ashok Enukonda

1. Introduction

In this lab, we implement a complete Business Intelligence (BI) flow from start to end. The data comes from AdventureWorks2014, which is a transactional database (OLTP) of a fictional bicycle company. This database contains business data such as sales, products, customers, employees, and other company operations.

The BI flow is implemented in five main steps. First, we create two databases: a Staging database and a Data Warehouse (DW). The Staging database is used as a temporary place to store and prepare the data. Then, we load data from AdventureWorks2014 into the Staging database using SSIS (ETL). After that, we load data from Staging into the Data Warehouse.

The Data Warehouse is designed as a Galaxy schema for the Sales Data Mart. It includes two fact tables: FactInternetSales and FactResellerSales. These fact tables share common dimension tables such as DimDate, DimProduct, DimSalesTerritory, and DimCurrency, and they also have specific dimensions such as DimCustomer (only for Internet Sales) and DimReseller and DimEmployee (only for Reseller Sales).

After the Data Warehouse is populated, we create an SSAS Tabular model (semantic layer) by importing the warehouse tables and deploying the model to Analysis Services. Finally, we create a Power BI report connected to the SSAS cube, so we can analyze sales across multiple filters such as year, month, product categories, countries, continents, and sales modality (Internet vs Reseller).

2. Architecture Overview

The architecture in Figure 1 shows the complete Business Intelligence flow used in this lab. The data starts from the AdventureWorks2014 database, which is the source system.

First, the data is loaded into a Staging database using SSIS. The staging layer is used to temporarily store and prepare the data. Then, the data is moved from the Staging database into the Data Warehouse, where it is organized for analysis.

After that, an SSAS Tabular model is created on top of the Data Warehouse. Finally, Power BI connects to this model to create reports that can be viewed by users.

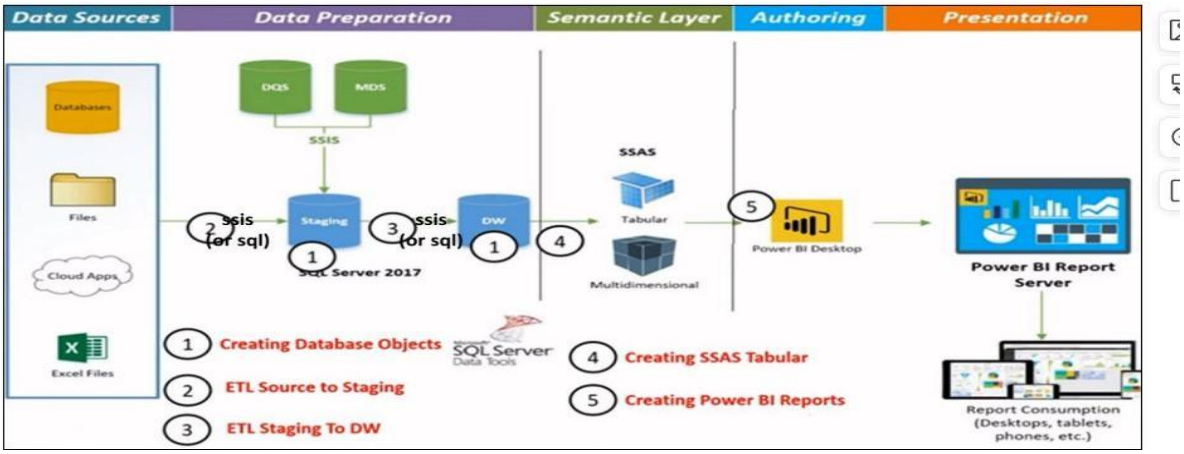


Figure 1: Architecture of the Business Intelligence flow showing data sources, staging area, data warehouse, SSAS Tabular model, and Power BI reporting.

3. Creating Database Objects

In this step, the databases needed for the BI system were created using SQL Server Management Studio. First, the AdventureWorks2014 database was restored. This database is used as the source database and contains the original business data.

Next, the Staging database called AWN_STG_Demo was created by running the SQL script provided for the lab. The staging database is used to temporarily store data coming from the source database. It helps to separate the source system from the Data Warehouse.

After that, the Data Warehouse database called AWN_DW_Demo was created by running the SQL script 2_AWN_DW_Demo only until line 326. This part of the script creates only the schemas and tables of the Data Warehouse. The code related to stored procedures was not executed at this stage.

The Data Warehouse is designed using a Galaxy schema, which includes dimension tables and two fact tables: FactInternetSales and FactResellerSales. At the end of this step, all required databases and tables were successfully created and were ready for the ETL process.

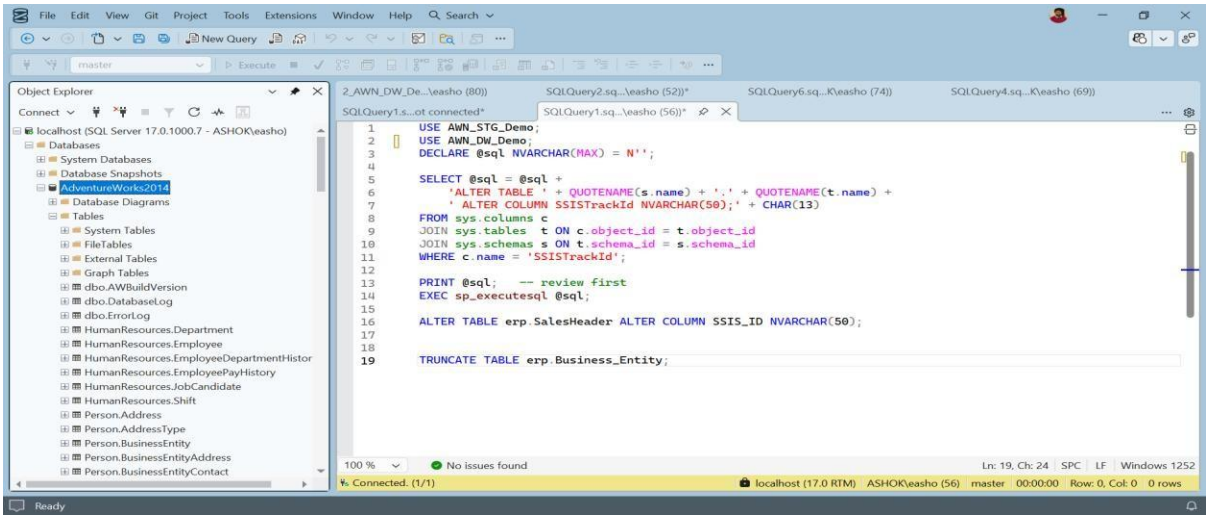


Figure 3.1: AdventureWorks2014 restored in SQL Server Management Studio as the source database.

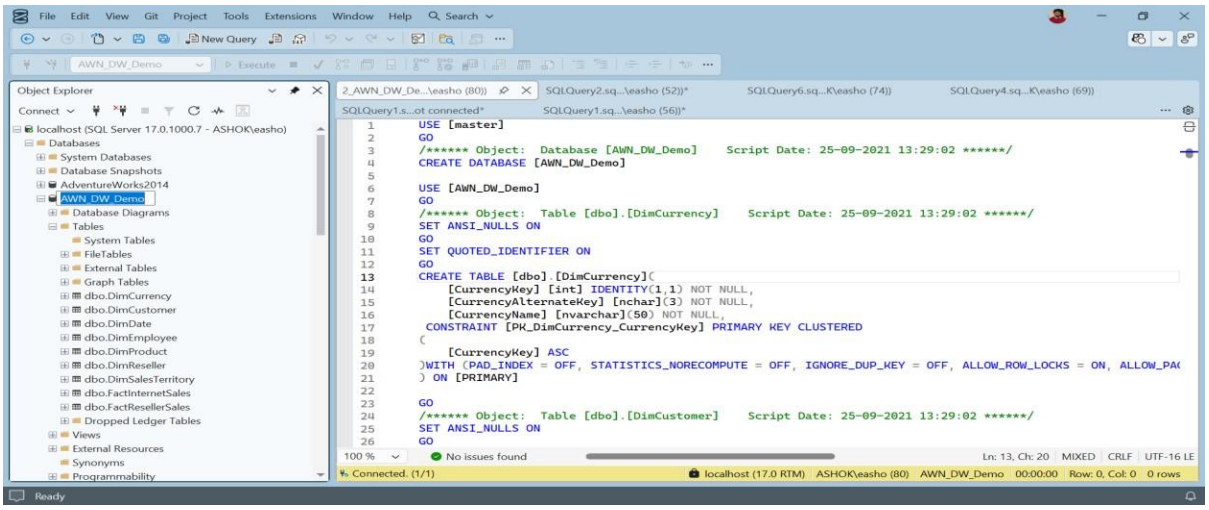


Figure 3.2: Creation of the Data Warehouse database (AWN_DW_Demo) using the SQL script until line 326.

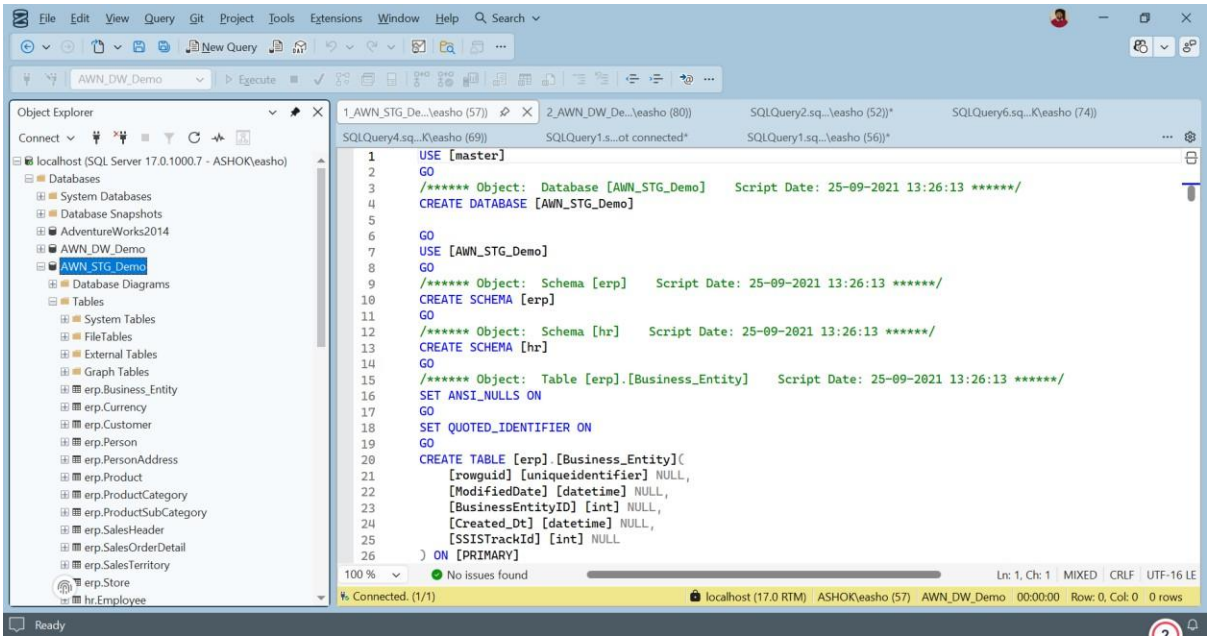


Figure 3.3: Staging database (AWN_STG_Demo) created with ERP and HR tables.

4. ETL - Source to Staging

In this step, SSIS (SQL Server Integration Services) was used to load data from the source database into the staging database. A new SSIS project was created in Visual Studio, and connections were added for both AdventureWorks2014 (source) and AWN_STG_Demo (destination).

Multiple Data Flow Tasks were created, each responsible for loading data into one staging table. Inside each Data Flow Task, OLE DB Source was used to read data from the source tables, and OLE DB Destination was used to insert the data into the staging tables. A Derived Column transformation was used to populate columns such as Created_Dt that do not exist in the source database.

After executing the SSIS package, the staging tables were checked in SQL Server Management Studio and were successfully populated with data.

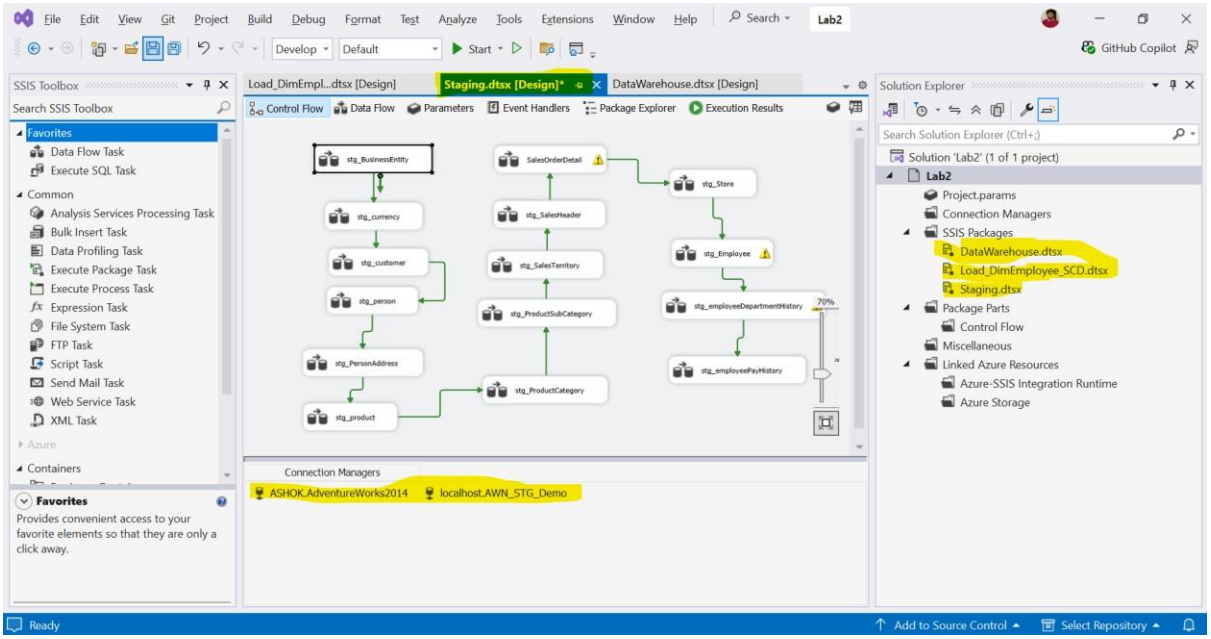


Figure 4.1: SSIS Control Flow showing multiple Data Flow Tasks used to load source data into the staging database.

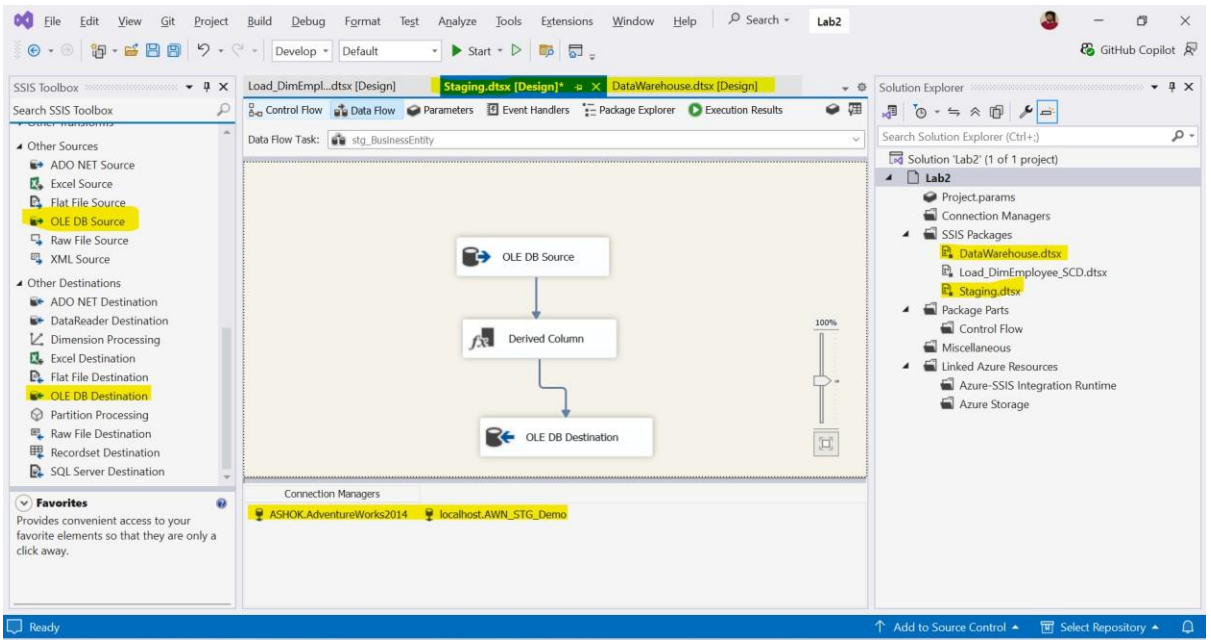


Figure 4.2: SSIS Data Flow Task using OLE DB Source, Derived Column, and OLE DB Destination to load data into the staging database.

5. ETL - Staging to Data Warehouse

In this step, data is moved from the Staging database (AWN_STG_Demo) into the Data Warehouse (AWN_DW_Demo). First, dimension tables are filled, and after that, fact tables are filled. This order is important because fact tables depend on dimension tables.

5.1 Loading Dimension Tables (Stored Procedures)

In this step, all stored procedures created in the SQL script 2_AWN_DW_Demo were executed one by one in SQL Server Management Studio. These procedures load data from the staging database into the dimension tables of the Data Warehouse.

Each procedure was executed manually to make sure that the corresponding dimension table was populated correctly. Dimension tables such as DimDate, DimProduct, DimCurrency, DimCustomer, and DimSalesTerritory were filled using these procedures.

Executing all the procedures first was important because the fact tables depend on the dimension tables.

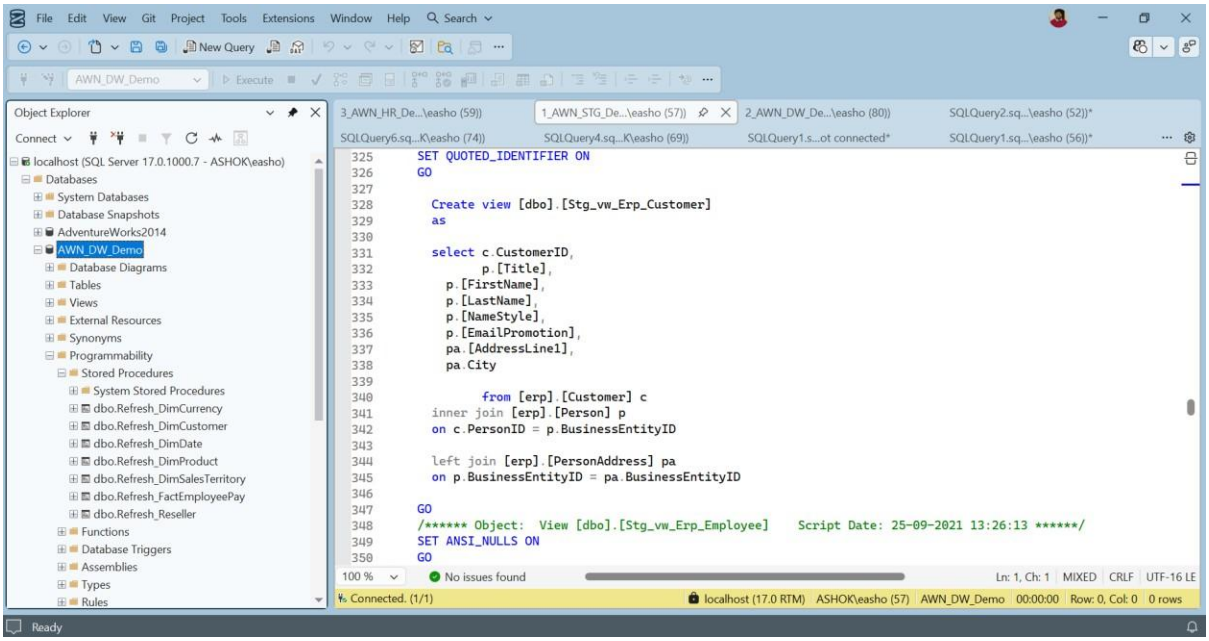


Figure 5.1: Execution of all stored procedures in SSMS to populate dimension tables in the Data Warehouse

5.2 Slowly Changing Dimension – DimEmployee

The DimEmployee table was handled differently because employee information can change over time. To keep the history of changes, a Slowly Changing Dimension (SCD) was used.

An SSIS package was created with a Slowly Changing Dimension transformation. The employee's NationalIDNumber was used as the business key. Attributes such as Title and Department were treated as historical attributes, and MaritalStatus was treated as a changing attribute. This approach allows old records to be closed and new records to be created when changes occur.

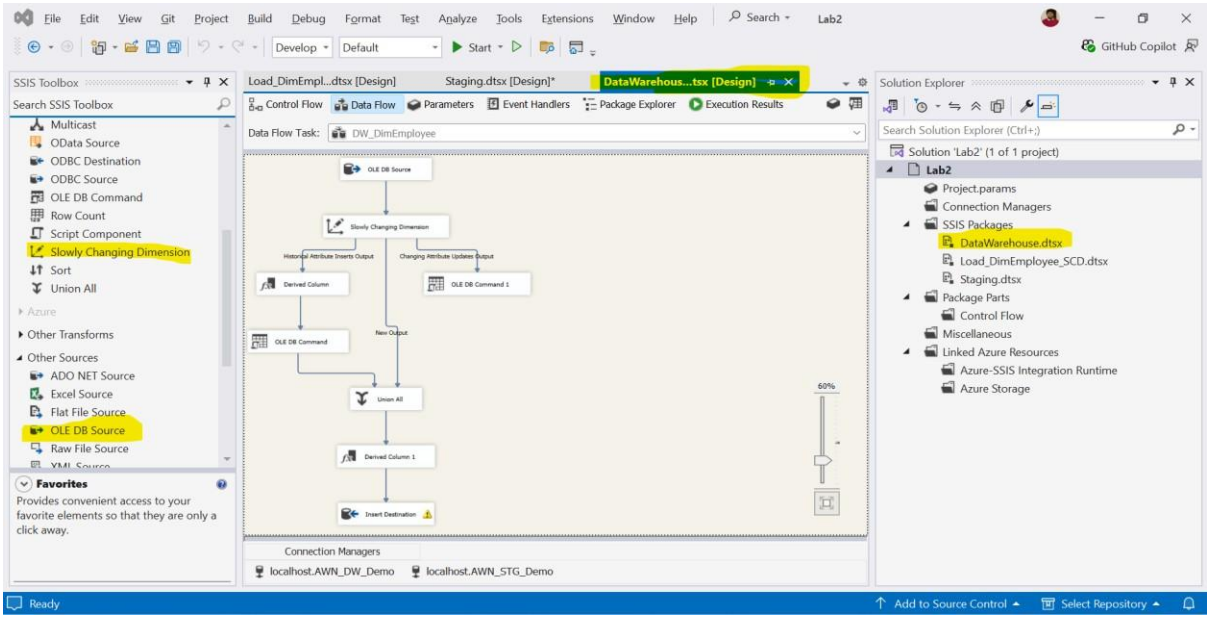


Figure 5.2: Slowly Changing Dimension implementation used to load the DimEmployee table.

5.3 Loading Fact Tables

After loading all dimension tables, the fact tables were populated using SSIS Data Flow Tasks. Separate Data Flow Tasks were created for FactInternetSales and FactResellerSales.

Each fact table was loaded from staging views. Lookup transformations were used to match business keys from the staging data with surrogate keys from the dimension tables. This ensures correct relationships between fact and dimension tables in the Data Warehouse.

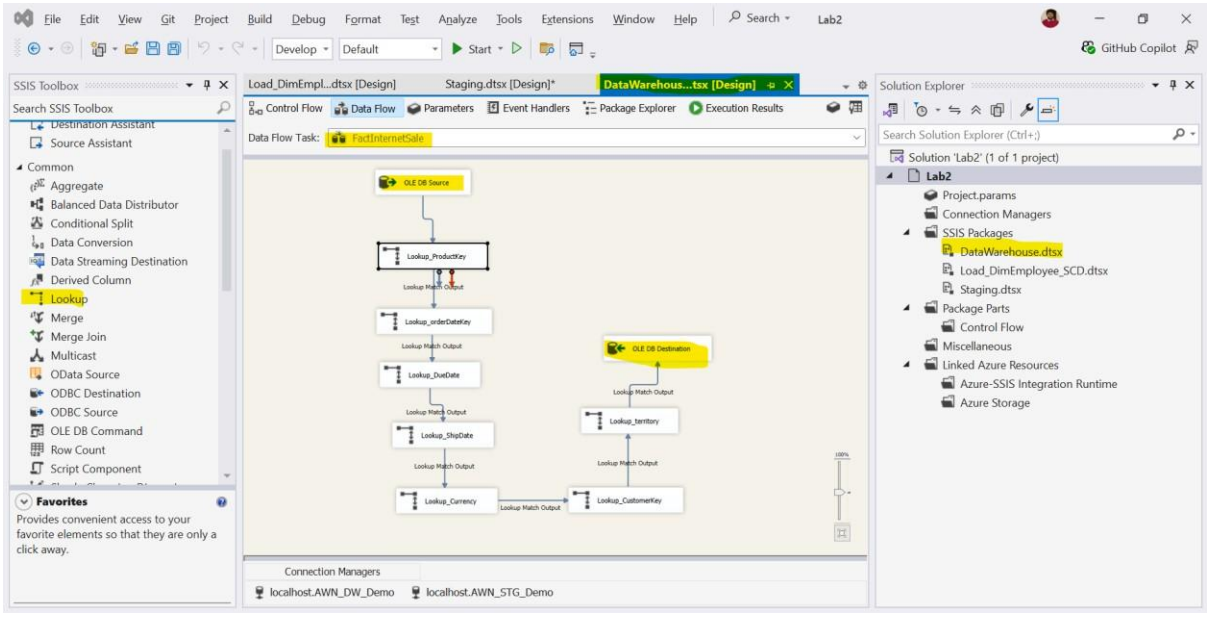


Figure 5.3: SSIS Data Flow used to load the FactInternetSales table using Lookup transformations.

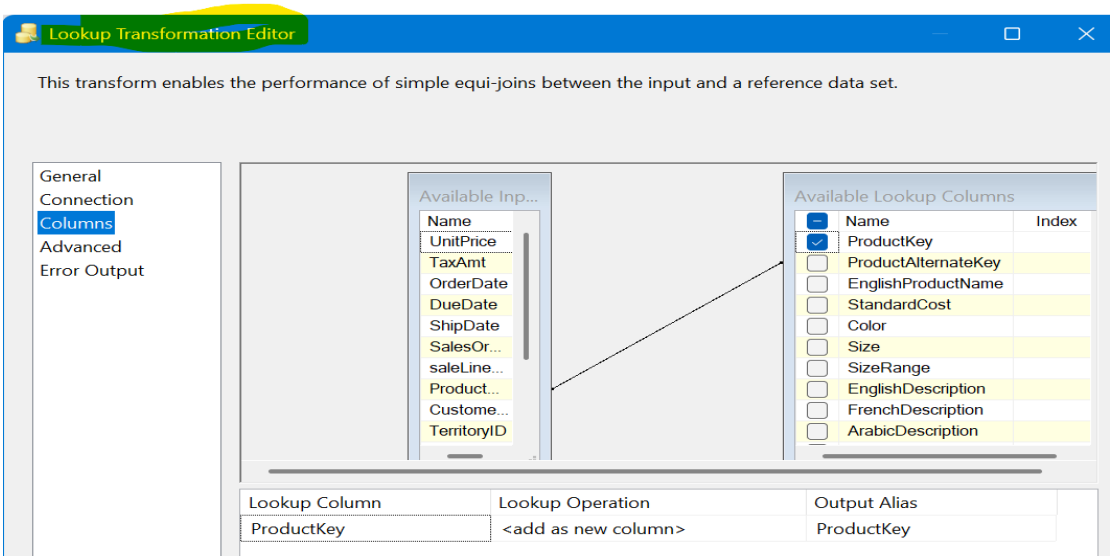


Figure 5.4: Lookup transformation used to map dimension keys while loading fact data.

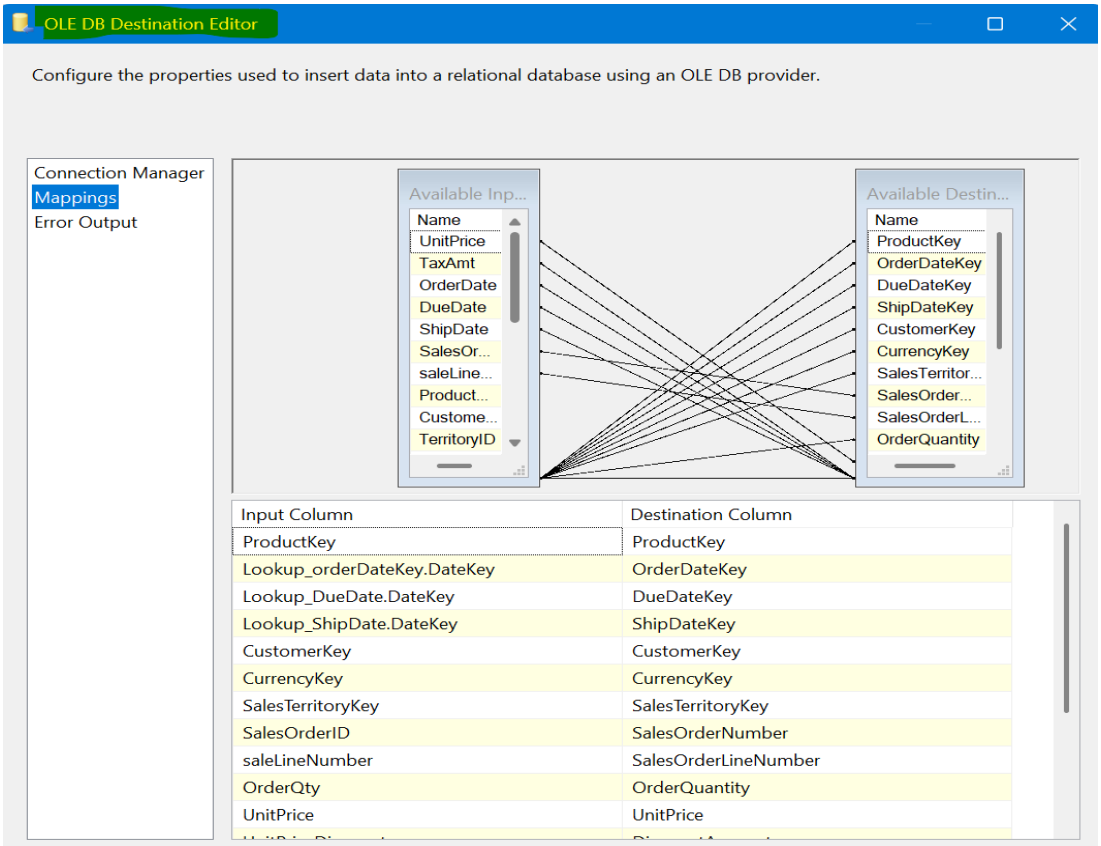


Figure 5.5: Column mapping in OLE DB Destination for FactInternetSales.

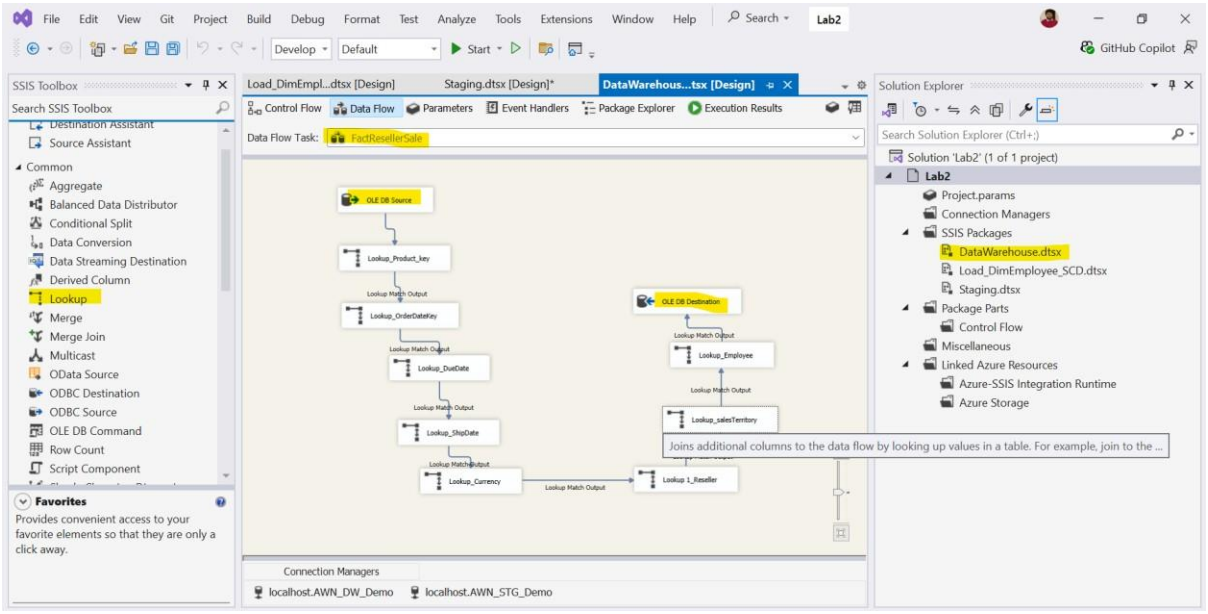


Figure 5.6: SSIS Data Flow used to load the FactResellerSales table.

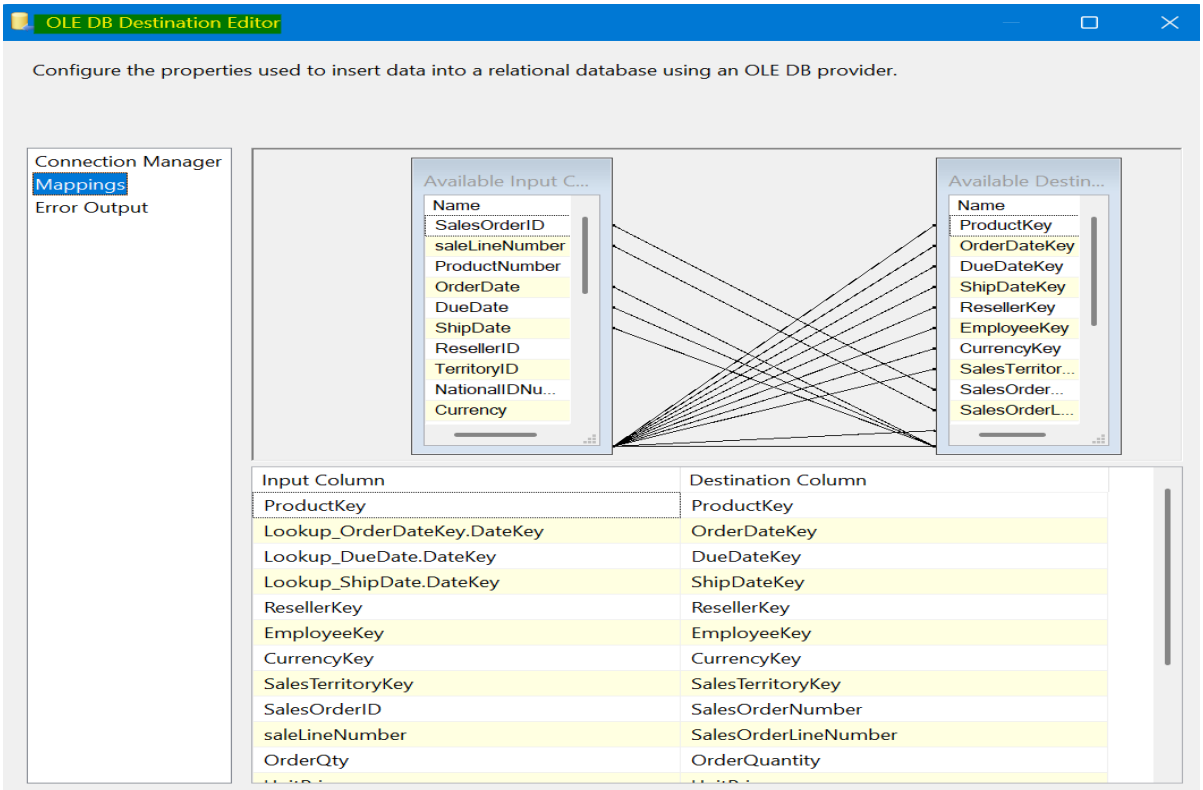


Figure 5.7: Column mapping in OLE DB Destination for FactResellerSales.

6. Creating SSAS Tabular Model

In this step, an SSAS Tabular model was created to make the data easy to use for reports.

First, a new Analysis Services Tabular project was created in Visual Studio. The AWN_DW_Demo database was selected as the data source. All required dimension tables and fact tables were imported into the project.

Before loading the data, the SQL Server Analysis Services account was added as a login in SQL Server Management Studio. This step was needed so that the tabular model could read data from the Data Warehouse.

After the tables were imported, the relationships between fact tables and dimension tables were created automatically. This formed a clear data model.

Finally, the tabular project was built and deployed to the Analysis Services server. After deployment, the model was available and ready to be used in Power BI.

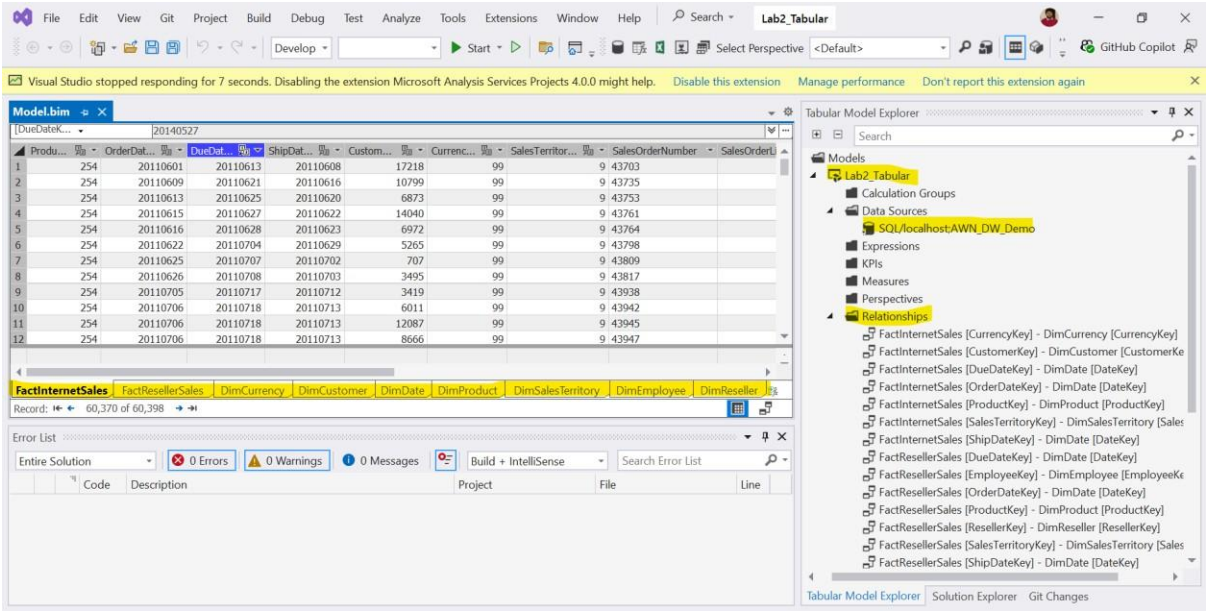


Figure 6.1: SSAS Tabular model created in Visual Studio showing tables, relationships, and data preview.

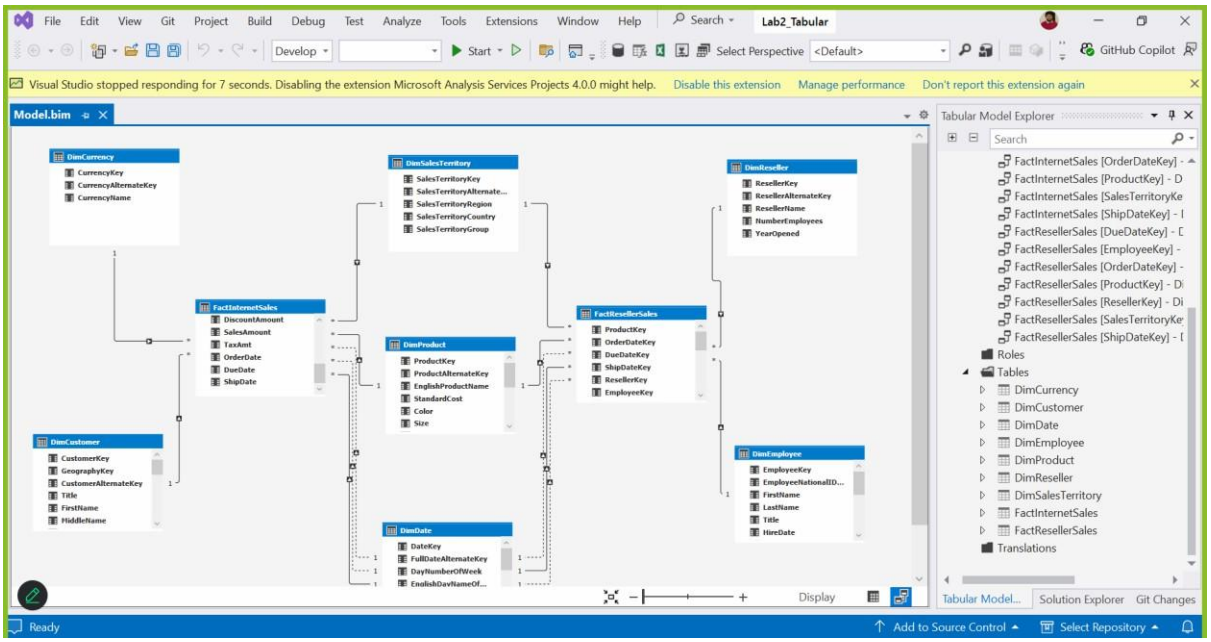


Figure 6.2: Galaxy schema showing FactInternetSales and FactResellerSales with shared and specific dimension tables.

7. Creating Power BI Report

In this step, a Power BI report was created using the SSAS Tabular model. Power BI Desktop was connected to the SSAS model using a live connection. This allows Power BI to read data directly from the tabular model without copying the data.

After connecting, the fact tables and dimension tables were available in Power BI. Different visuals were created to analyze sales data. The report allows users to view sales by year, month, product category, product subcategory, country, continent, and sales type (Internet or Reseller).

Slicers were added to filter the data easily. The report also shows total sales and average sales values, such as daily, weekly, monthly, and quarterly averages. The visuals are interactive, so users can click and filter the data to see different results.

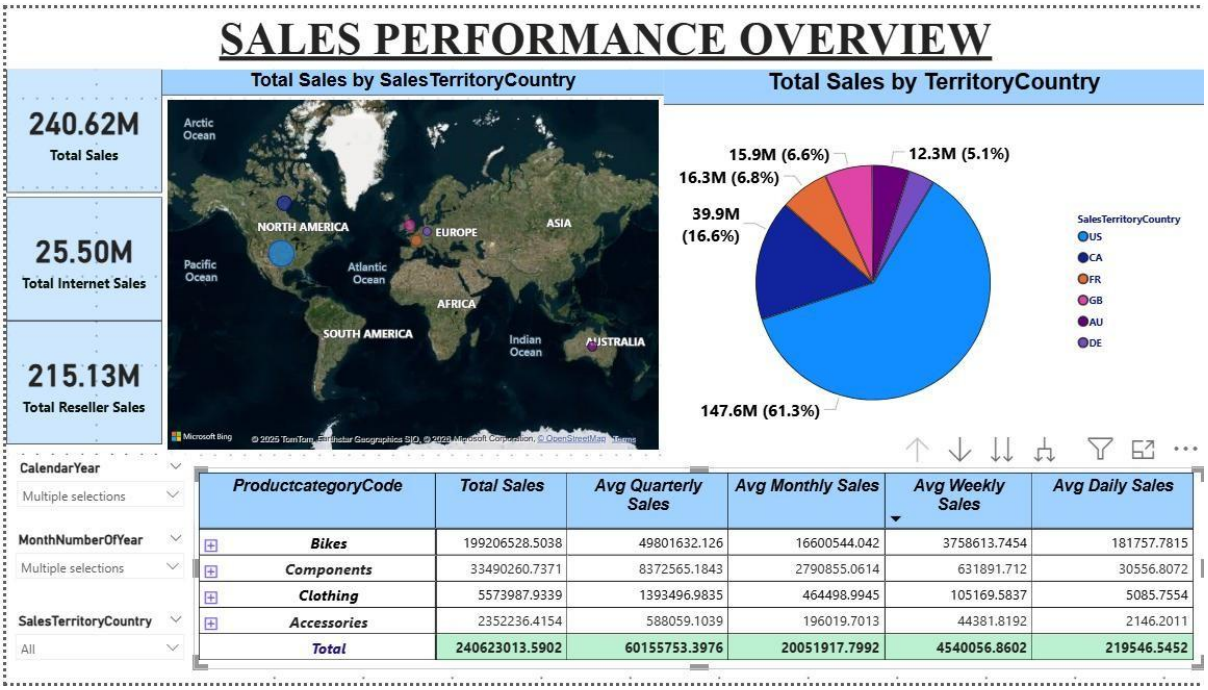


Figure 7.1:-Sales Performance Overview

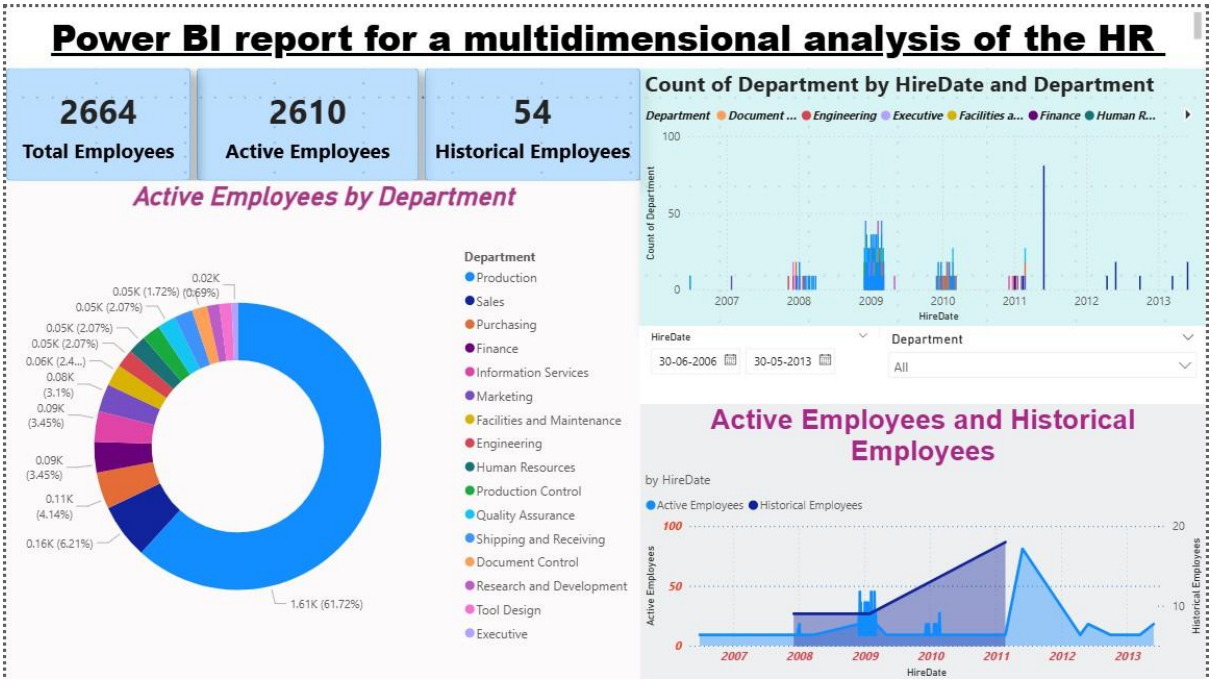


Figure 7.2:- HR Advanced Task (already correct)

8. Advanced Tasks

In addition to the main requirements of the lab, the advanced tasks were also completed to extend the Business Intelligence solution.

HR Data Mart

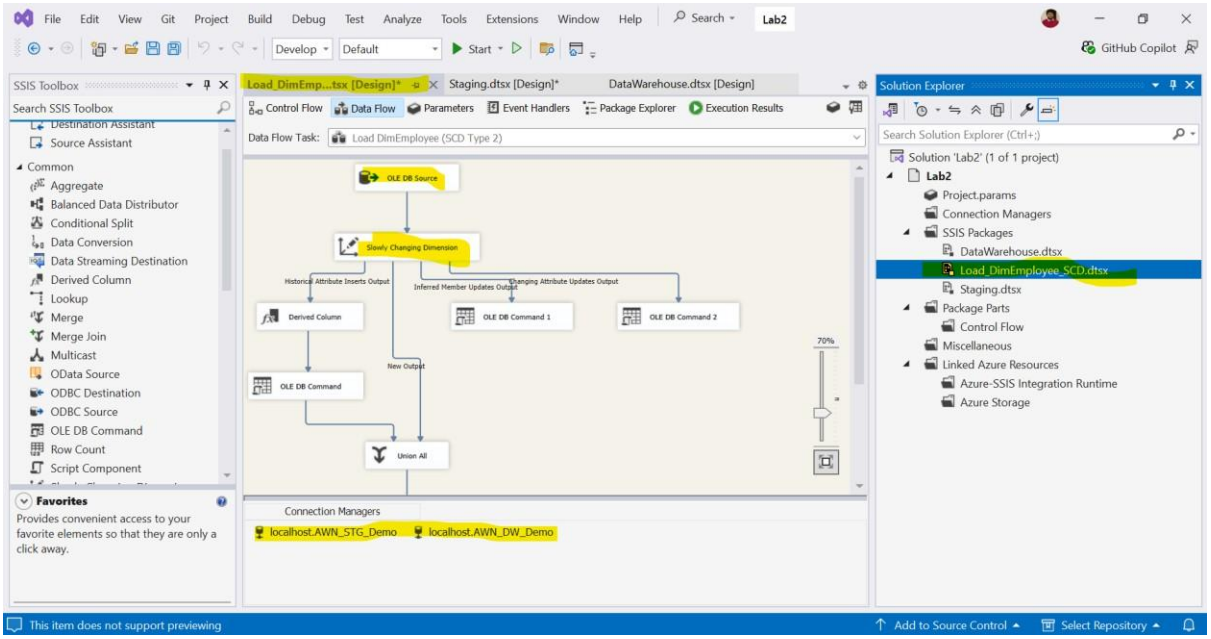
A Human Resources (HR) Data Mart was created in the Data Warehouse. The HR-related data was loaded into the Data Warehouse following the same ETL process used for the Sales Data Mart. This allows analysis of employee-related information in a structured way.

SSIS Deployment and Automation

The SSIS project solutions were deployed successfully. An SQL Server Agent job was created to automate the Data Warehouse update process. This job can be scheduled to run at specific times, which allows the Data Warehouse to be refreshed automatically without manual execution.

Power BI Report for HR Data

A Power BI report was created for the HR Data Mart. This report allows multidimensional analysis of HR data, such as employee information and department-related insights. The report uses interactive visuals and filters to make HR data easy to explore and understand.



Conclusion

In this lab, a complete Business Intelligence solution was built step by step. Data was taken from the source database, loaded into a staging database, and then moved into a Data Warehouse. The Data Warehouse was designed properly to support analysis using fact tables and dimension tables.

An SSAS Tabular model was created to make the data easy to use. Power BI reports were developed to analyze sales data using different filters and visuals. The reports allow users to understand the data clearly and interact with it.

The advanced tasks were also completed. An HR Data Mart was created, the ETL process was automated using SQL Server Agent, and an additional Power BI report was developed for HR analysis.

Overall, this lab helped in understanding how to build an end-to-end BI system, from data loading to reporting, using real tools and real data.

THANK YOU