בס"ד

Qognity Sales Data Mart

BI System Specifications Document

Date: 15/07/2024

Written by: Hodaya Bilya

בס"ד

Content

1. General

1.1. Project Objective

This project's objective is the creation of a full BI solution for Qognify's sales department, to support Qognify's growth in products sales.
The project was designed according to Qognify's sales department KPIs and is aimed at increasing the company's overall.

Qognify is a company specializing in security and safety software solutions.
It offers advanced systems for video monitoring and management (VMS), incident management, and physical security information management (PSIM).
Qognify's products are primarily used in sensitive locations such as airports, train stations, industrial facilities, and large public buildings.

The Data Mart creation will be done using information derived from the PriorityERP database , the Qognify's operational database.
The solution will include summarized data tables, focusing on products sales data, as well as data regarding Qognify's customers, salespersons, and stores.
In addition, the BI solution will include costumed reports containing sales analysis, customer analysis, and executive dashboard.
These reports will be tailored for the sales departments' needs and will contribute to Qognify's products sales growth.

1.2. Project Contents

The project includes the building of a Data Mart which contains sales data.
The data will be transferred through an ETL process from the PriorityERP operational database to the Data Mart – QognifyDM.

ERD model of the QognifySalesDM database: ERD Link

1.2.1. The Data Mart will include 1 fact table and 4 dimension tables, and 2 history table:

- FactSales – Data regarding all sales, including the id of the order, products bought, quantities, and prices. Data loading process for this table will be incremental.
- DimCustomers – Data regarding the company's customers.
- DimStores – Data regarding the company's stores.
- DimEmployees – Data regarding the company's employees.
- DimProducts - Data regarding the company's products.
- DimEmployeesHistory – Historic data regarding the company's employees.

Source To Target Link

2. Gantt

Gantt_Qognify_SalesDM

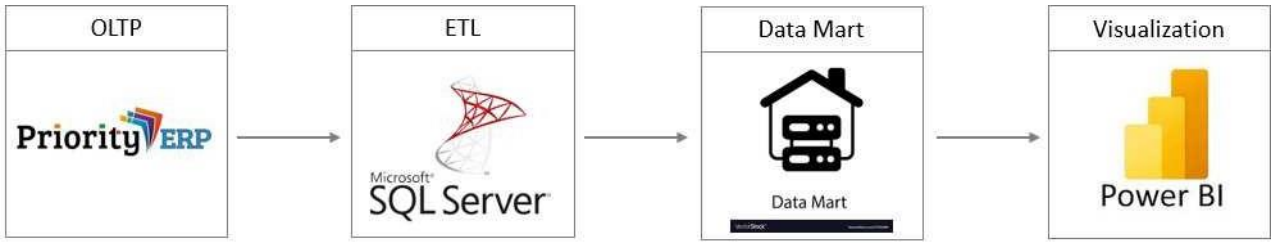
3. Technical Specification

3.1 Prerequisites

SQL Server	ERP system in the operational DB (PriorityERP) - tables, data (SQL files).
SSIS	ETL processes using SSIS in Visual Studio.
Data refresh processes	Definition of JOBS in SSMS.
Power BI	Creating reports and dashboards using Power BI.

3.2 Solution Architecture

3.2.1 High Level Design:



The ETL process, which includes arranging the data into a Data Mart will be performed in SQL Server using SSIS.
After the Data Mart creation, reports will be created using Power BI.

3.2.2. Power BI Reports:

3.2.2.1. The report for the sales department will consist of:

- Total sales
- Total orders
- Average Monthly Revenue
- Total units
- Total sales and year over year growth (this graph can change to orders, units)
- Total sales by month and day - online vs physical stores (this graph can change to orders, units)
- Orders by Year and Store Name
- Sales by country (this graph can change to orders, units)

3.2.2.2. The report for the customer department will consist of:

- Total number of customers
- Number of new customers
- Average Monthly Revenue
- Total units
- Total customers and new customers by month and Year
- Customers by Category
- Number Of Customers by Country
- Revenue by Country
- Selling Products by number of customers.
- Average Revenue per Customer by Month (Drill Mode) vs. Previous Year

3.2.2.3. The executive dashboard will consist of:

- Total sales
- Total orders
- Average Monthly Revenue
- Total units
- Total sales and month over month growth by quarter and month
- Customer Is Active
- Selling products by Number of Customers
- Revenue by country
- Customers by Year/ Month

4. Functional Specification

4.1. ETL processes

4.1.1 Mrr_Tables package:

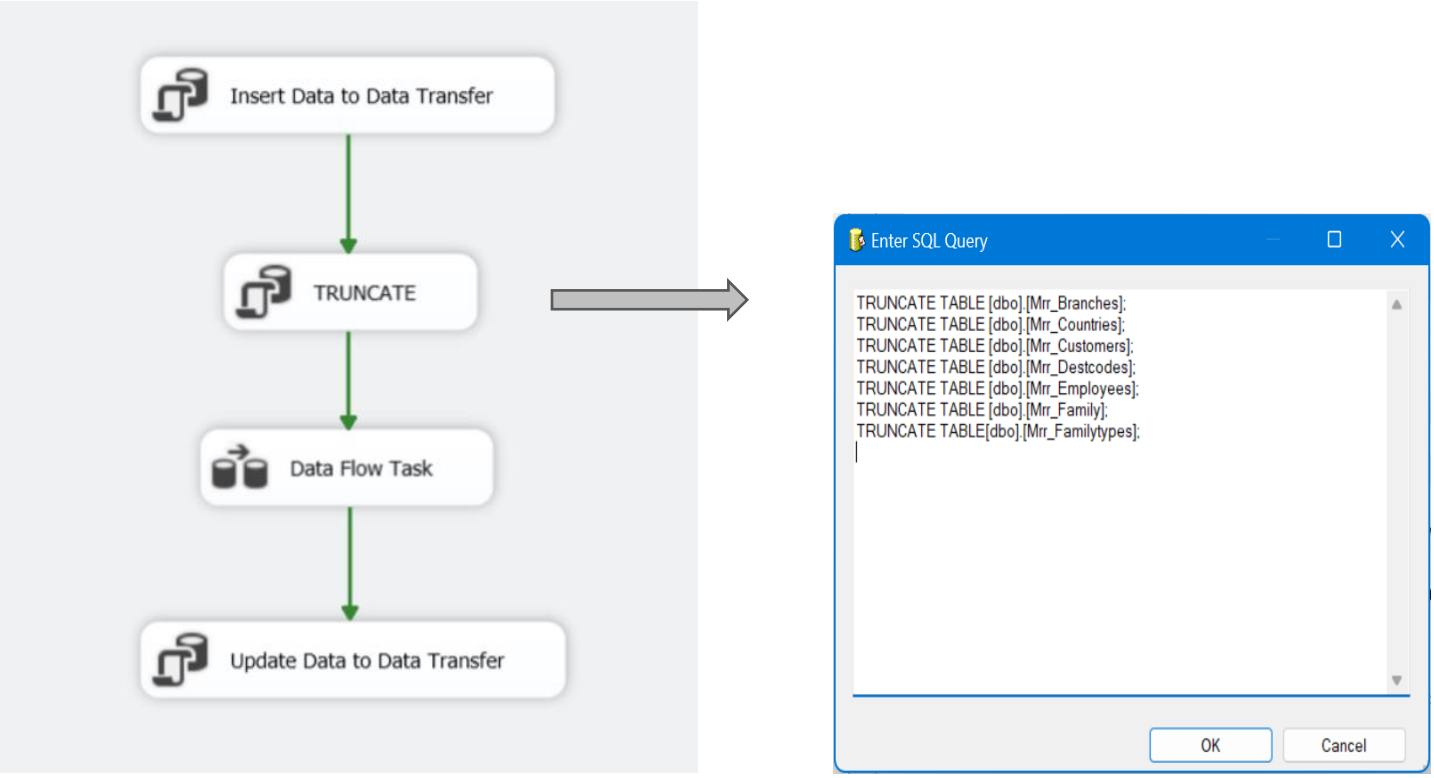
This package is responsible for loading data from PriorityERP tables (OLTP) to all mirror tables relevant for the dim tables .

Execute SQL Task - TRUNCATE:

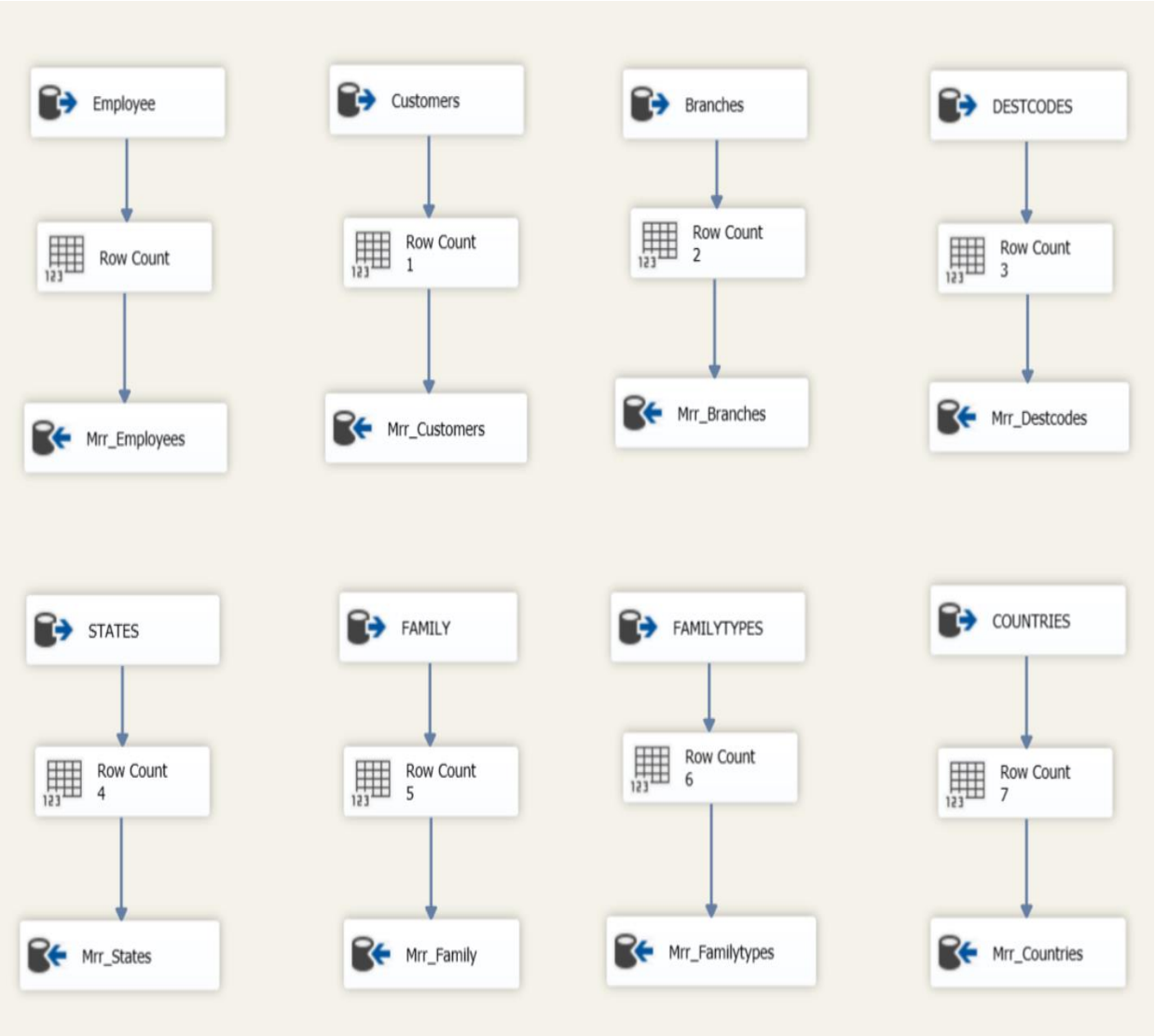
This stored procedure, named TRUNCATE, is designed to truncate (delete all rows) from multiple tables.

Finally, we insert the values to the transfer table.

The TransferTable serves as a comprehensive log, meticulously capturing every update and insertion step as data moves through the stages from the database (DB) to the Data Mart.

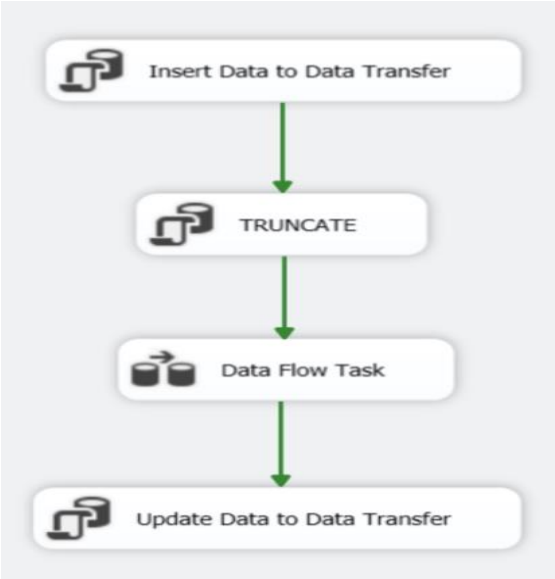


In the data flow:

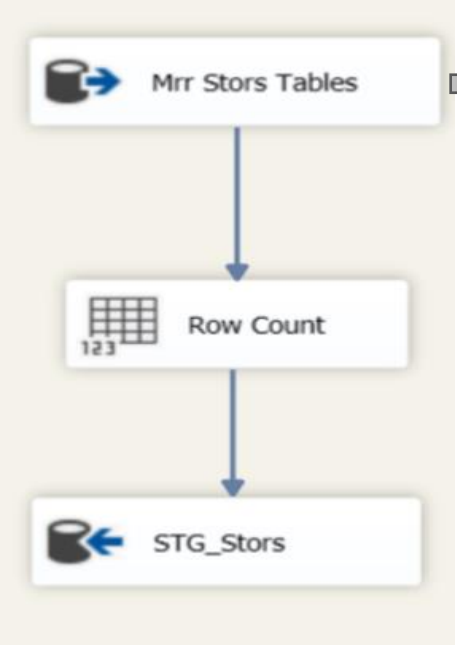


- **Dim_Stores Table:**
  - **o <u>STG_Stores package:</u>**

    STG_Stores table is truncated, and the mirror tables are joined and loaded using a data flow task.



In the data flow, the 4 mirror tables (Mrr_Invoices, Mrr_destcodes, Mrr_states,Mrr_countries) are joined, and the data is loaded to STG_Stores table.

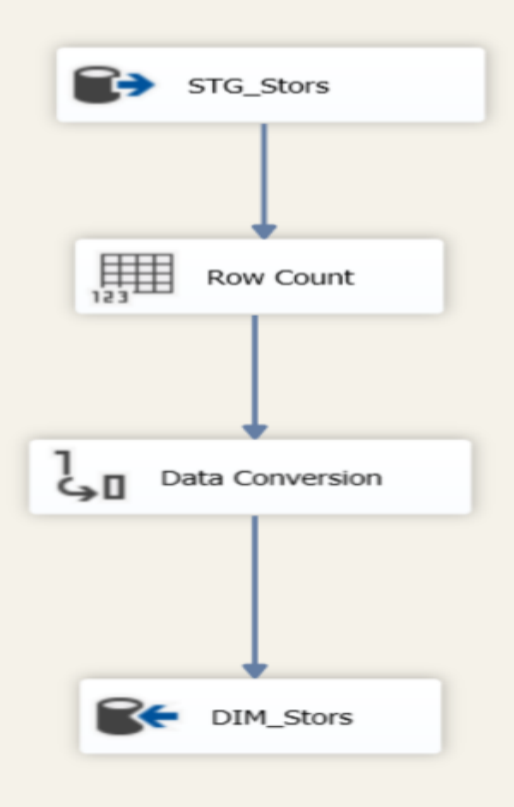



```
SELECT distinct b.branch as StorID, b.branchname as StorName ,
dc.ADDRESS,dc.city,s.STATENAME as Region , c.COUNTRY
FROM Mrr_Branches b
JOIN mrr_invoices i on b.branch = i.branch
LEFT JOIN mrr_destcodes dc on b.branch = dc.DESTCODE
LEFT JOIN mrr_states s on dc.state = s.stateid
LEFT JOIN mrr_countries c on s.country = c.COUNTRYCODE
```

o DWH_Stores package:

Data is loaded from STG_Stors to DWH_Stors, and a Total column is added.

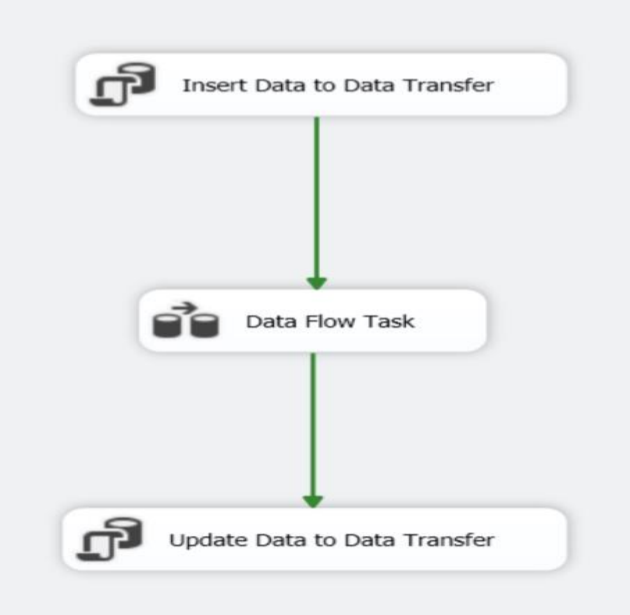
In the Data Conversion, we converted the columns as required for the DIM_Stors table.



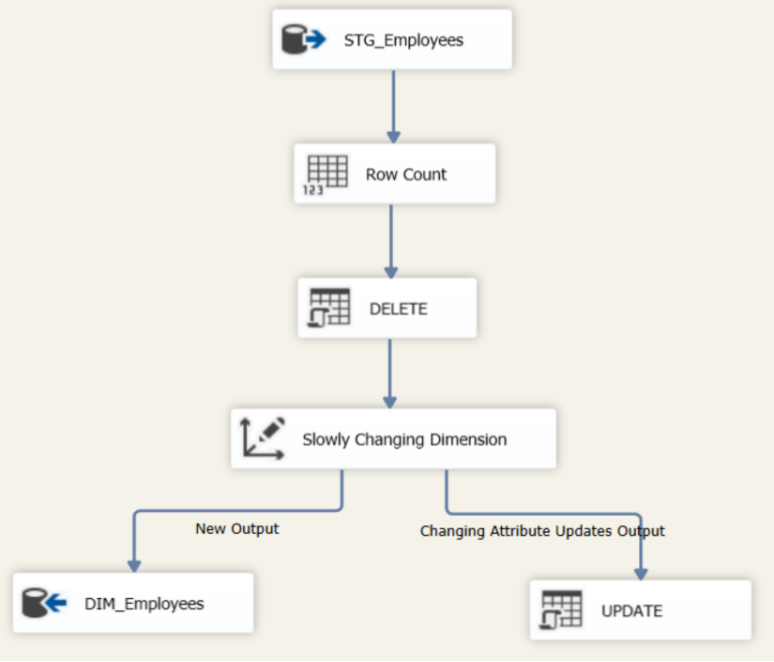
Input Column	Output Alias	Data Type	Length	Precision	Scale	Co
ADDRESS	Copy of ADDRESS	Unicode string [DT_WSTR]	50			
city	Copy of city	Unicode string [DT_WSTR]	40			
COUNTRY	Copy of COUNTRY	Unicode string [DT_WSTR]	50			
StorName	Copy of StorName	Unicode string [DT_WSTR]	50			

- **Dim_Employees Table:**
  - **o <u>STG_Employees package:</u>**

    STG_Employees table is truncated, and the mirror tables are joined and loaded using a data flow task.

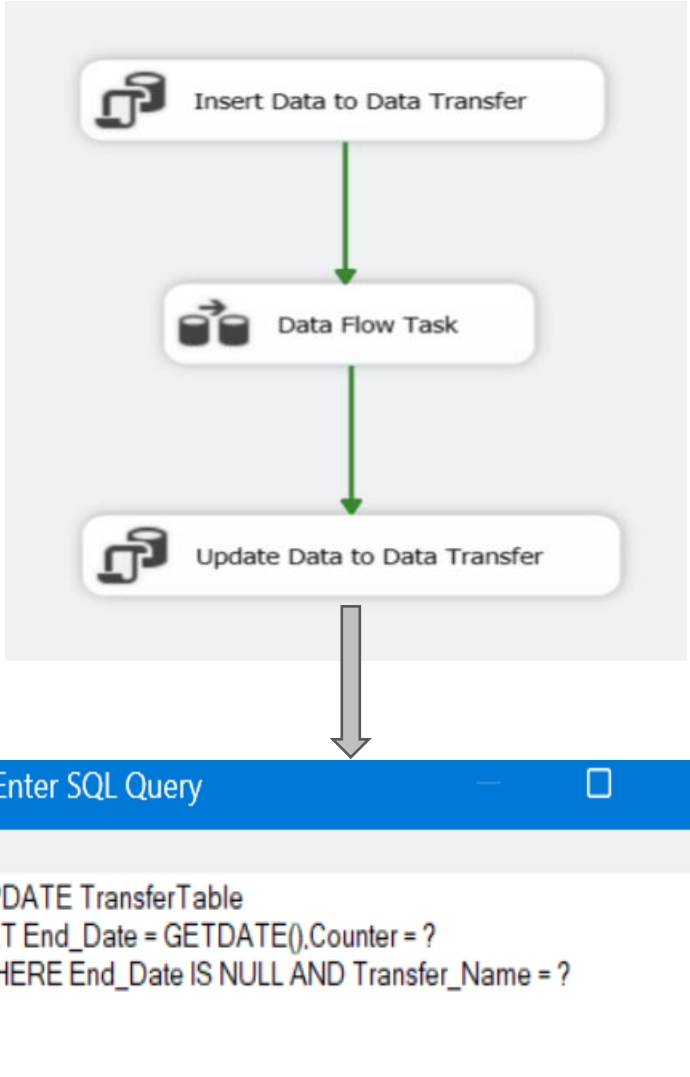


In the data flow, this process updates the `DIM_Employees` table with new and modified employee records, while removing outdated ones as needed. It uses Slowly Changing Dimensions to handle updates to existing records.

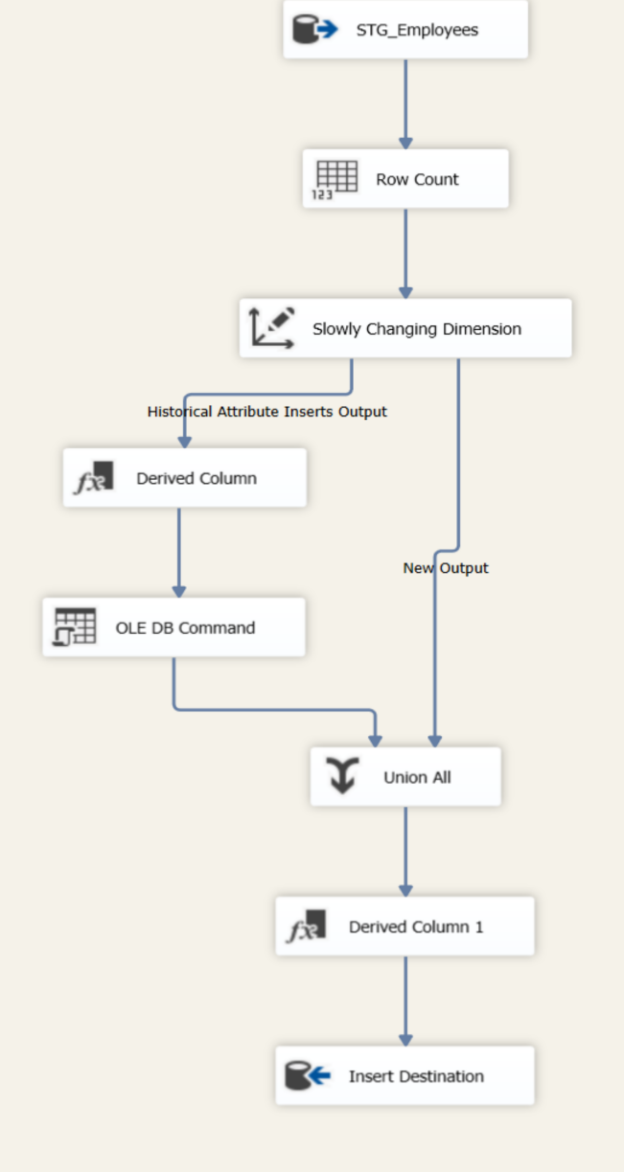


- **Dim_EmployeesHistory Table:**
  - **o <u>DWH_EmployeesHistory package:</u>**

    DWH_EmployeesHistory table is truncated, and the mirror tables are joined and loaded using a data flow task.



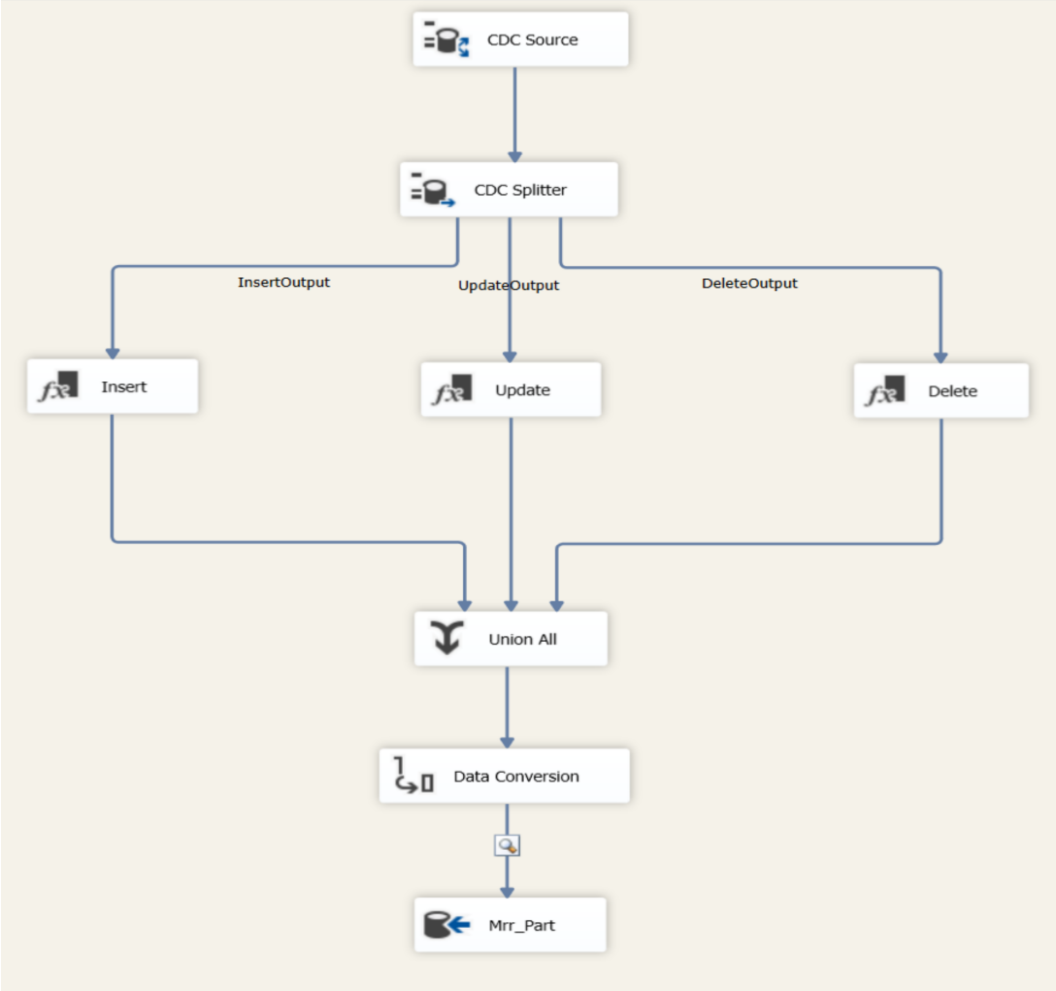
In the data flow, this ETL process uses a Slowly Changing Dimension (SCD) approach to handle new records and updates, including historical data updates. It ensures that employee records are accurately inserted and updated in the destination table.



- **Dim_Products Table:**

  - **Mrr_Products package:**

    In Mrr_Products, a CDC process was carried out.
    Below is an explanation of the process:



  - **STG_Products package:**

    - CDC Source:
      This component serves as the data source and outputs the changes made to the data source (new records, updates, and deletions).
    - CDC Splitter:
      This component splits the changes received from the CDC Source into three different outputs: InsertOutput (inserts), UpdateOutput (updates), and DeleteOutput (deletes).

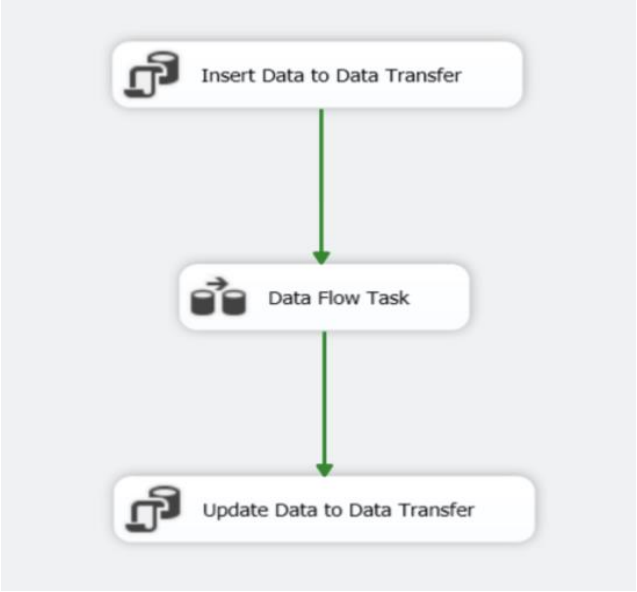
- Union All:
  After the insert, update, and delete operations, the records from all three outputs are combined into a Union All component.
- Data Conversion:
  Next, all records pass through a Data Conversion component, which converts data types as required for the target table.

- Mrr_Part:
  Finally, the records are loaded into the target table named Mrr_Part.

  The entire process is based on the principle of tracking changes in the source data and accurately applying those changes to the target table while ensuring data type compatibility.

o <u>DWH_Products package:</u>

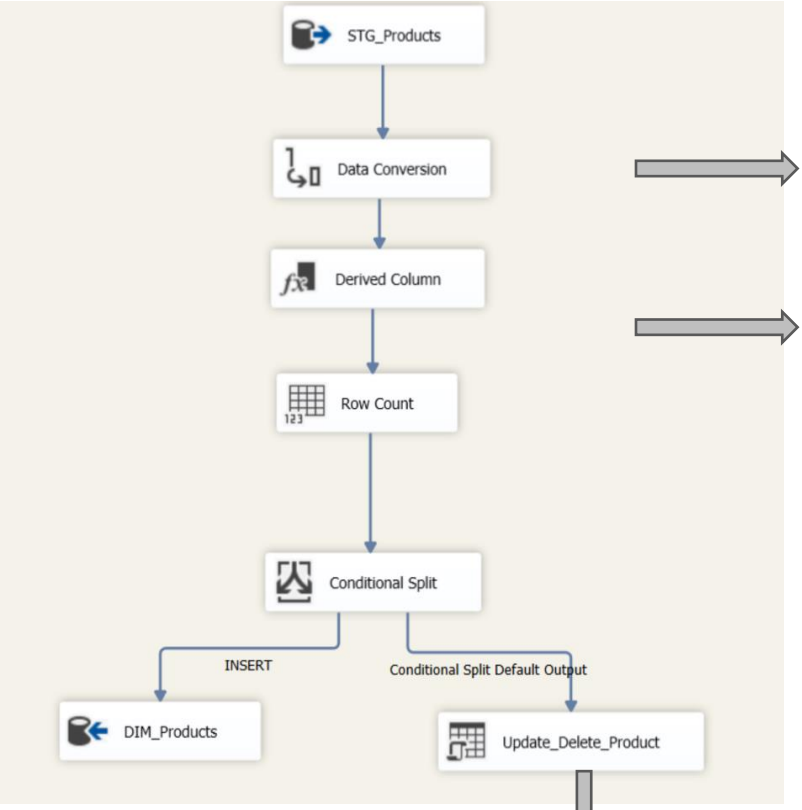
Data is incrementally loaded and updated in DIM_Products. Deleted records are updated in DIM_Products using an Execute SQL task.s



In the Data Flow, the process extracts product data from the staging table (STG_Products), applies transformations (such as data type conversions and derived column calculations), counts the rows, and then uses a conditional split to either insert

new products into the DIM_Products table or update/delete existing products in the Update_Delete_Product step.

This ETL process ensures that the DIM_Products table in the data warehouse is kept up to date with the latest product information from the staging area.



Input Column	Output Alias	Data Type	Length	Pre...	Sca...
Category	Copy of Category	Unicode string [DT_WSTR]	20		
Sub_Category	Copy of Sub_Catego...	Unicode string [DT_WSTR]	50		
Cost	Copy of Cost	numeric [DT_NUMERIC]		8	2
ProductName	Copy of ProductNa...	Unicode string [DT_WSTR]	50		

Derived Column Name	Derived Column	Expression	Data Type
IsActive	<add as new column>	1	four-byte signed integer

```
CREATE PROCEDURE [dbo].[SP_Update_Delete_Product_From_DWH_Qgnify]
AS
    MERGE [dbo].[DIM_Products] AS Target
    USING [dbo].[STG_Products] AS Source
    ON [Target].[ProductID] = [Source].[ProductID]
    WHEN MATCHED THEN UPDATE
    SET [Target].[ProductName] = [Source].[ProductName],
    [Target].[Sub_Category] = [Source].[Sub_Category],
    [Target].[Category] = [Source].[Category],
    [Target].[Price] = [Source].[Price],
    [Target].[Cost] = [Source].[Cost],
    [Target].[IsActive] = IIF([Source].[Status]=2,1,0);
GO
```

This stored procedure, `SP_Update_Delete_Product_From_DWH_Qognify`, updates product data in the target table `DIM_Products` based on data from the staging table `STG_Products`. Here's a breakdown of the process:

1. Merge Operation:
   - The target table is `DIM_Products`, and the source table is `STG_Products`.
   - The records are matched based on the `ProductID` column.

2. When Matched - Update:
   - If a product ID from `STG_Products` matches a product ID in `DIM_Products`, the procedure updates the target table with values from the source table:
     - `ProductName`, `Sub_Category`, `Category`, `Price`, and `Cost` columns are updated with the respective values from the source.
     - The `IsActive` column is set based on the `Status` value in the source table. If `Status` equals 2, `IsActive` is set to 1 (indicating active); otherwise, it is set to 0 (indicating inactive).

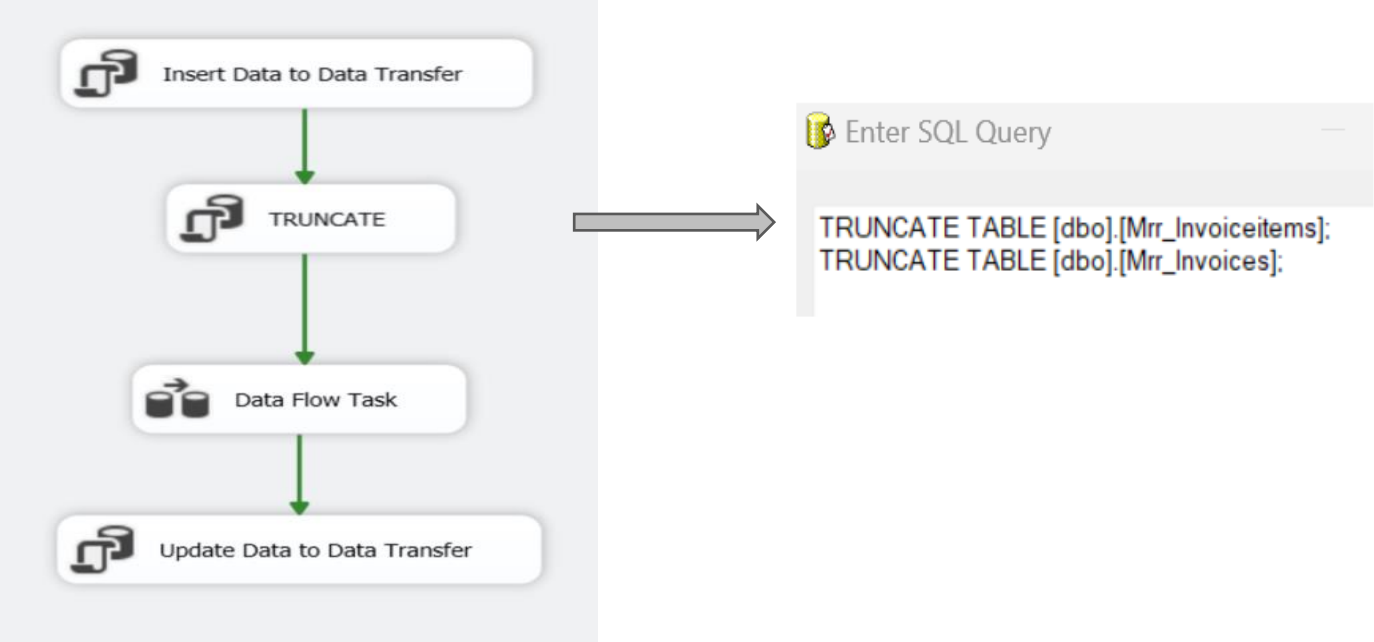
This process ensures that the target product table is kept current with any updates from the source data.

- **FactSales Table:**

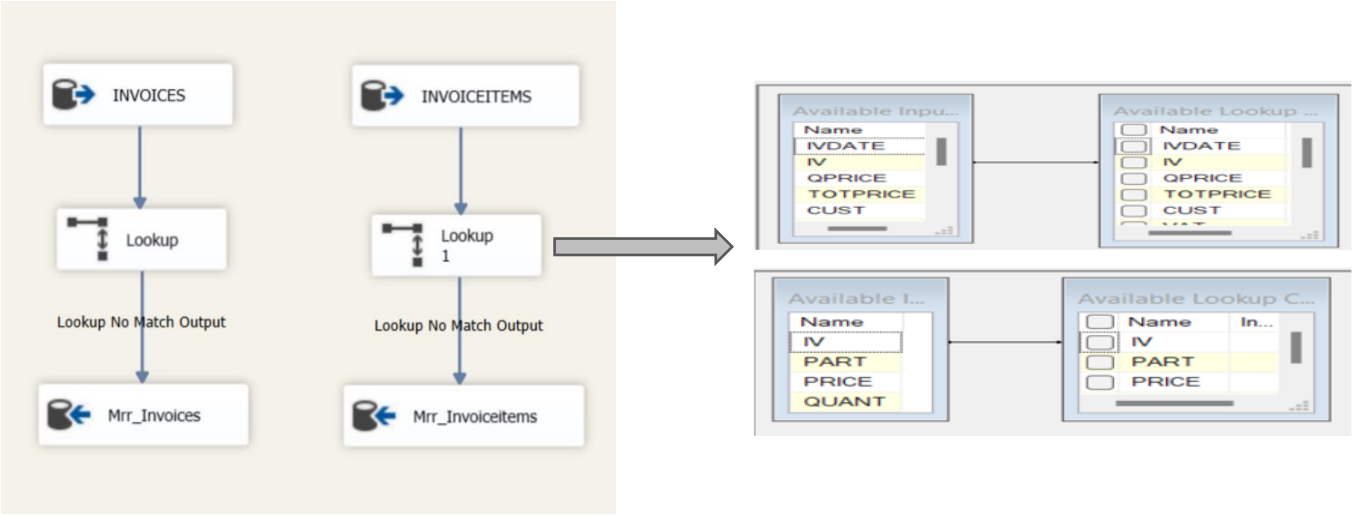
o **MRR_Sales package:**

Invoices & InvoiceItems :

This Execute SQL Task, named TRUNCATE, is designed to truncate (delete all rows) from multiple tables.

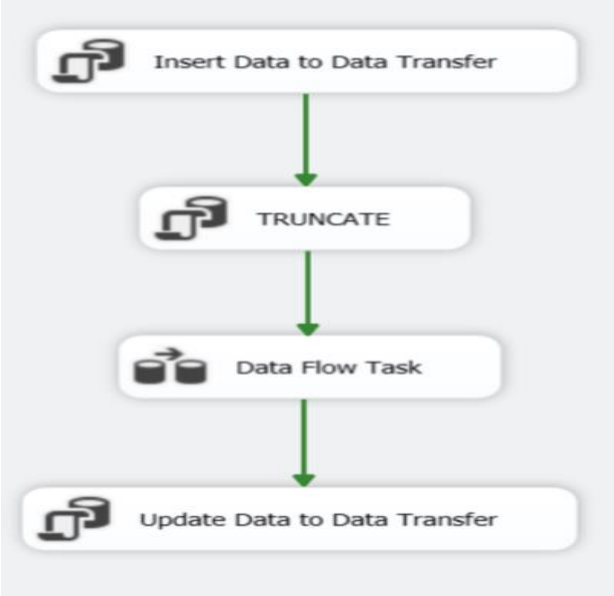


The Mrr_Invoices table is loaded incrementally using a Lookup transformation to compare incoming records with existing ones in MrrInvoicesDim. Unlike other mirror tables, Mrr_Invoices is not truncated during this process.

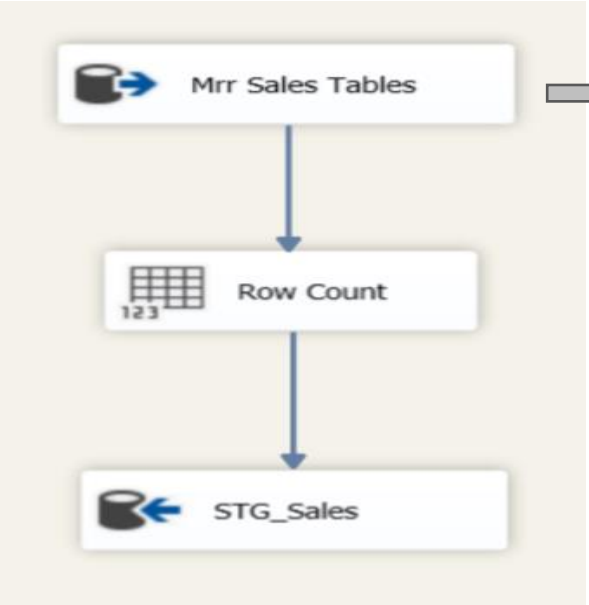


o <u>STG_Sales package:</u>

STG_Sales table is truncated, and the mirror tables are joined and loaded using a data flow task.



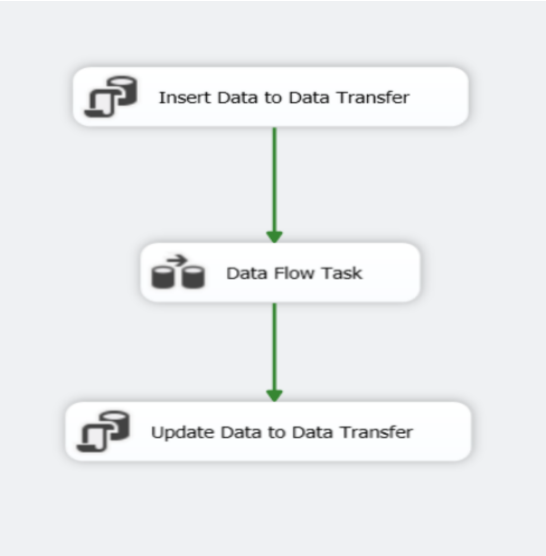
In the data flow, the mirror tables (MrrInvoices and MrrInvoiceItems) are joined, and the resulting data is loaded into the StgSales table.




```
SELECT I.IV AS OrderID,I.AGENT AS EmployeeID, I.CUST AS
CustomerID,DATEADD(DAY,IVDATE,'01-01-1910')AS
OrderDate,
II.PART AS ProductID,I.BRANCH AS StorID, II.QUANT AS Qty,
II.PRICE,1-(I.TOTPRICE/(I.QPRICE*1.17)) AS
DISCOUNT,I.TOTPRICE AS Total
FROM Mrr_Invoices I
JOIN Mrr_Invoiceitems II ON i.IV = II.IV
```

o DWH_Sales package:

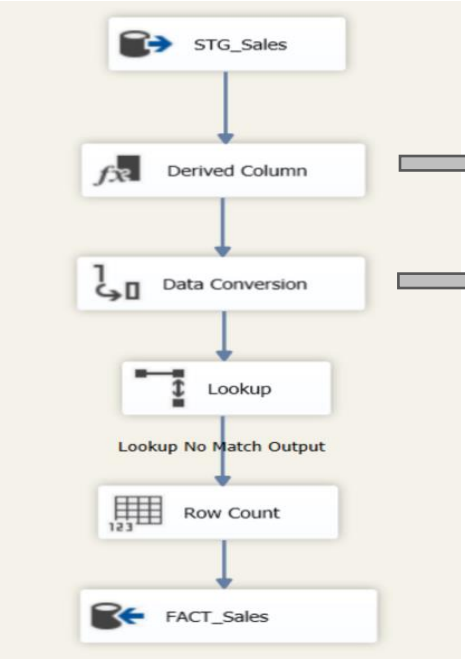
Data is loaded from Stg_Sales to Fact_Sales, and a Total column is added.



In the data flow, a check is performed to determine whether the purchase was made online or in a physical store.

If the purchase was made online, the value 1 is entered into the IsOnline column otherwise, the value 0 is entered.

In the Data Conversion, we converted the columns as required for the FACT_Sales table.

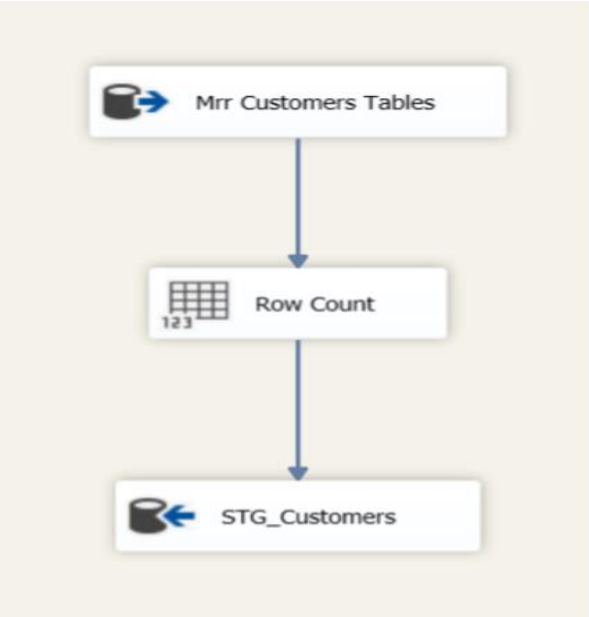


Derived Column Name	Derived Column	Expression	Data Type
IsOnline	<add as new column>	EmployeeID == 77777 ? 1 : 0	four-byte signed inte...

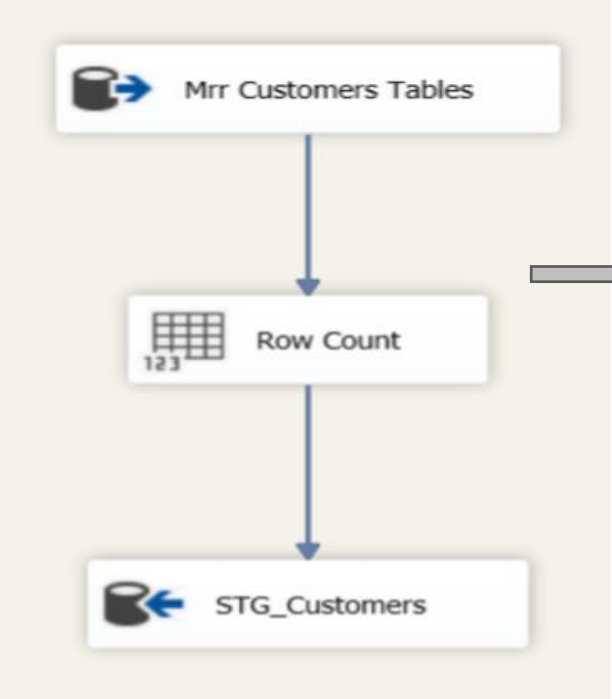
Input Column	Output Alias	Data Type	Length	Precision	Scale
EmployeeID	Copy of EmployeeID	four-byte signed integer [DT_I4]			
CustomerID	Copy of CustomerID	four-byte signed integer [DT_I4]			
ProductID	Copy of ProductID	four-byte signed integer [DT_I4]			
StorID	Copy of StorID	four-byte signed integer [DT_I4]			
Qty	Copy of Qty	four-byte signed integer [DT_I4]			
PRICE	Copy of PRICE	decimal [DT_DECIMAL]			2
DISCOUNT	Copy of DISCOUNT	decimal [DT_DECIMAL]			2
Total	Copy of Total	decimal [DT_DECIMAL]			2
IsOnline	Copy of IsOnline	Boolean [DT_BOOL]			

o <u>STG_Customers package:</u>

STG_Customers table is truncated, and the mirror tables are joined and loaded using a data f low task.



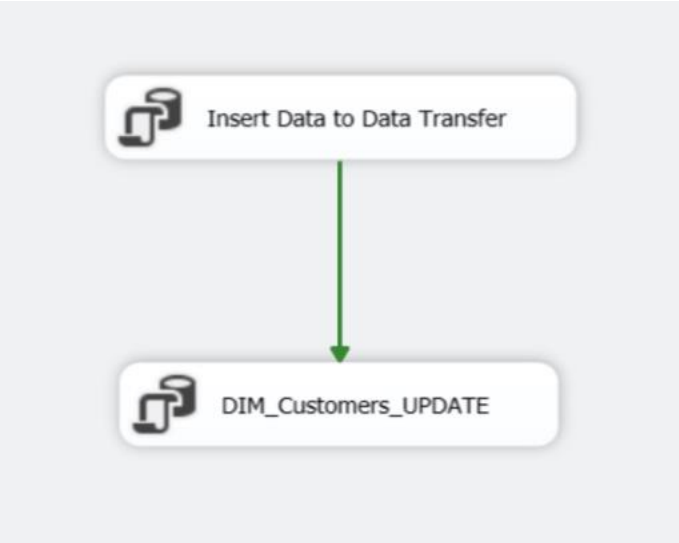
In the data flow, the 5 mirror tables (Mrr_Customers , Mrr_Invoices, Mrr_Destcodes, Mrr_States, Mrr_Countries) are joined, and the data is loaded to STG_Customers table.




```
SELECT DISTINCT I.CUST AS CusromerID, C.CUSTNAME AS
CustomerName, D.ADDRESS, D.CITY,CO.COUNTRY
FROM Mrr_CUSTOMERS C
RIGHT JOIN Mrr_INVOICES I ON c.CUST = I.CUST
LEFT JOIN Mrr_DESTCODES D ON I.DESTCODE = D.DESTCODE
LEFT JOIN Mrr_STATES S ON D.STATE = S.STATEID
LEFT JOIN Mrr_COUNTRIES CO ON S.COUNTRY =
CO.COUNTRYCODE
```

o DMH_Customers package:

Data is incrementally loaded and updated in Dim_Customers.




```
CREATE PROCEDURE [dbo].[SP_Update_DWH_Customers_Q]

AS
    MERGE [dbo].[DIM_Customers]  AS Target
    USING [dbo].[STG_Customers]  AS Source
    ON [Target].[CustomerID]= [Source].[CustomerID]

    WHEN MATCHED AND [Target].[IsActive] = 1 AND
    (ISNULL([Target].[CustomerName],'') <> ISNULL([Source].[CustomerName],'')
    OR ISNULL([Target].[Address],'') <>ISNULL([Source].[Address],'')
    OR ISNULL([Target].[City],'') <> ISNULL([Source].[City],'')
    OR ISNULL([Target].[Country],'') <> ISNULL([Source].[Country],''))
    THEN UPDATE
    SET [Target].[CustomerName] = [Source].[CustomerName],
    [Target].[Address] =[Source].[Address],
    [Target].[City] =[Source].[City] ,
    [Target].[Country] =[Source].[Country],
    [Target].[UpdateDate] = GETDATE()

    WHEN NOT MATCHED BY Target THEN INSERT
    VALUES([Source].[CustomerID],[Source].[CustomerName],[Source].[Country],[Source].[Address],[Source].[City],
    1,GETDATE())

    WHEN NOT MATCHED BY Source AND [Target].[IsActive] = 1 THEN UPDATE SET [Target].[IsActive] = 0;

    UPDATE TransferTable
    SET Counter = @@ROWCOUNT, End_Date = GETDATE(),Total_Time = DATEDIFF(second,Start_Date,GETDATE())
    WHERE End_Date IS NULL AND Transfer_Name = 'DWH_Customers';
```

A merge stored procedure is executed in the Execute SQL Task, the merge statement works according to the following rational:

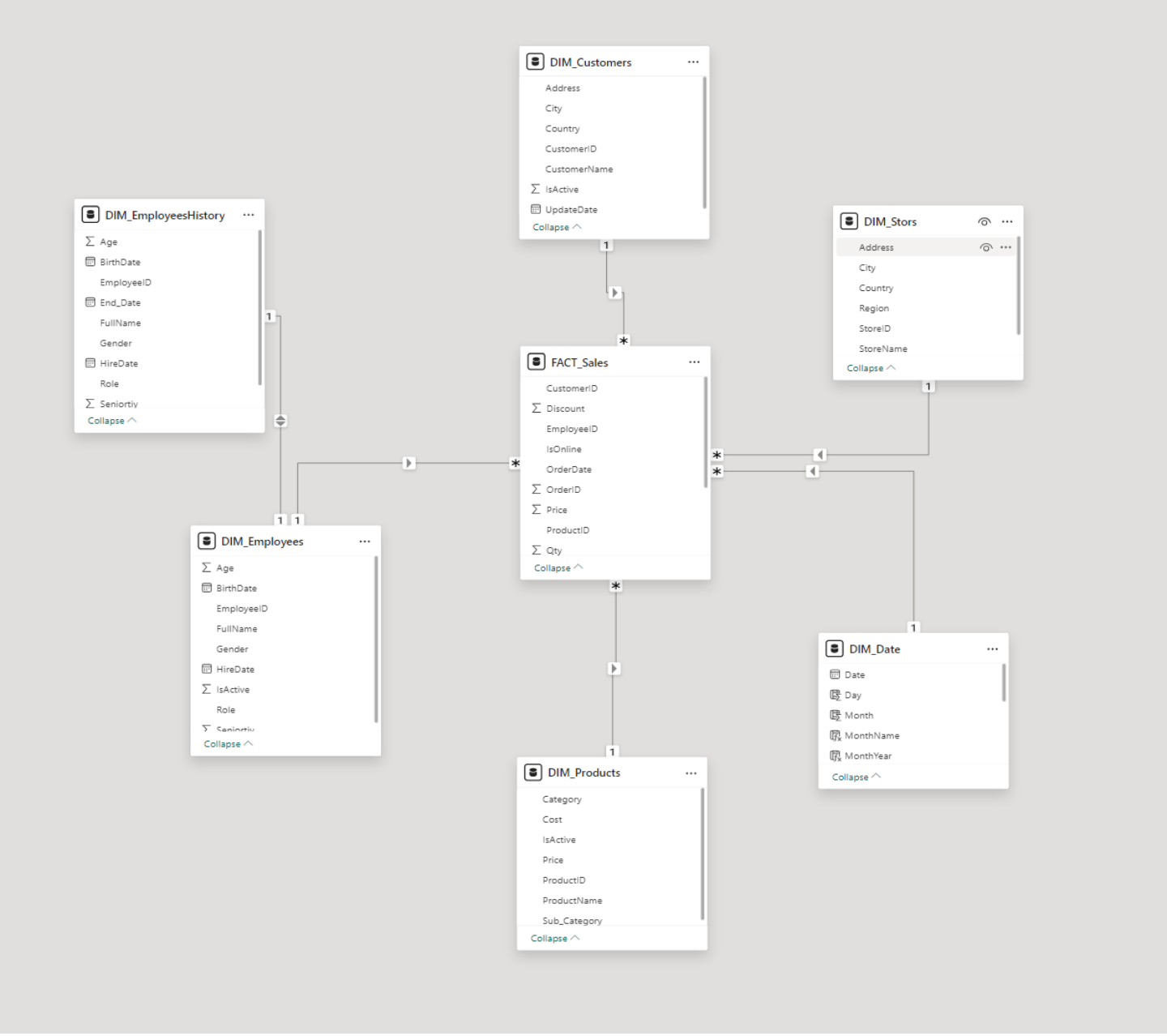
What this process does:

1. **Update existing customer records**: If there is an active customer with updated information, the relevant details are synchronized between the STG_Customers and DIM_Customers tables.
2. **Insert new customer records**: New customers from the STG_Customers table are inserted into the DIM_Customers table.
3. **Deactivate missing customers**: Customers found in the DIM_Customers table but no longer present in the STG_Customers table are marked as inactive.
4. **Log the operation**: The TransferTable logs the operation's details, including the number of records updated and the time taken.

4.3. Visualization in Power BI:

4.3.1. The reports were created using Power BI Desktop and were published to Power BI Service.

The model in the Power BI includes the Fact table and the 5 Dimension tables (not including the product history table). To these tables, a Dim_Date table.

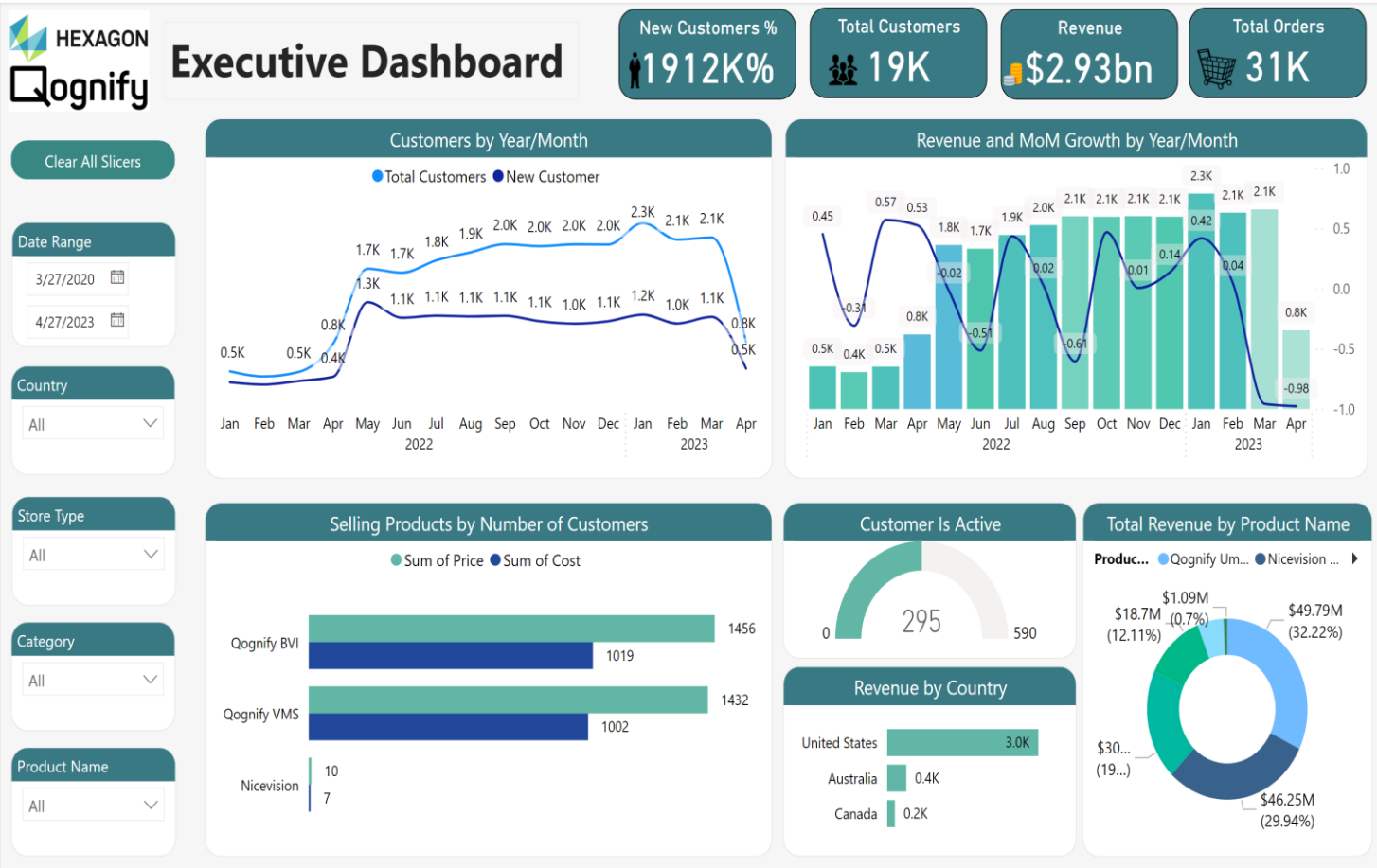


4.3.3. Reports:

The project includes 3 reports: Executive Dashboard, Sales Analysis, Customer Analysis.

4.3.3.1. Executive Dashboard:

This report was created to provide a broader look at the company's status, it includes the main KPIs, sales performance vs. goals, and general graphs.



KPI Cards:

- Revenue
- Total Orders
- Total Customers
- New Customers %

Graphs:

- Revenue and MoM Growth by Year /Month
- Revenue by Product name
- Selling Products by Number of Customers
- Customers by Year/ Month
- Revenue by Country
- Customer Is Active

Slicers:

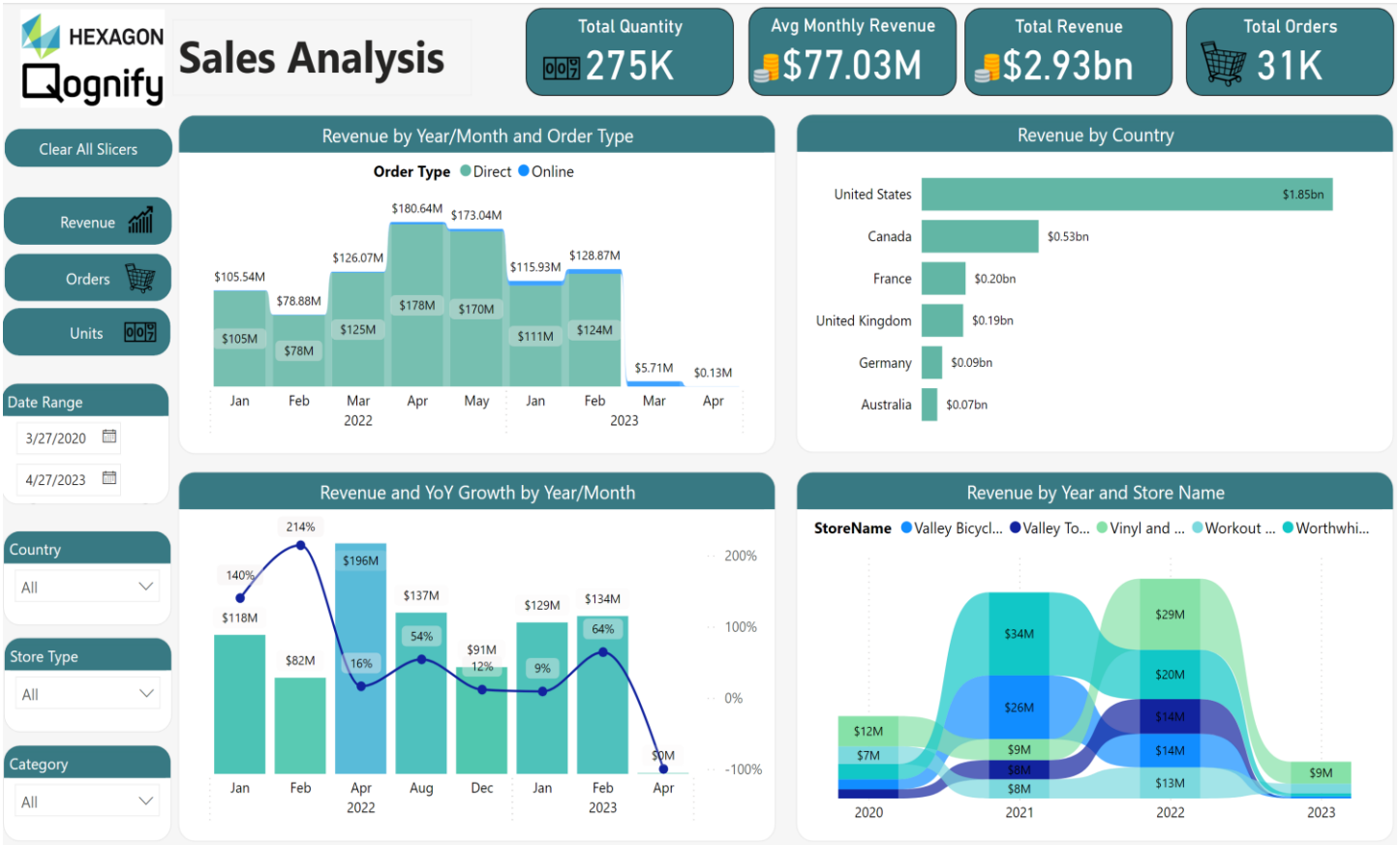
- Date Range
- Country
- Store Type
- Category
- Product Name

4.3.3.2. Sales Analysis:

This report was created for the sales department to follow and understand

sales performance to achieve the department's goals.
In its initial state, the graphs present revenue data.
Using the three buttons on the to right, the user can control the data shown in the graphs and change it to orders data, and units data.

Revenue State:



KPI Cards:

- Total Revenue
- Total Orders
- Average Monthly Revenue
- Total Units

Slicers:

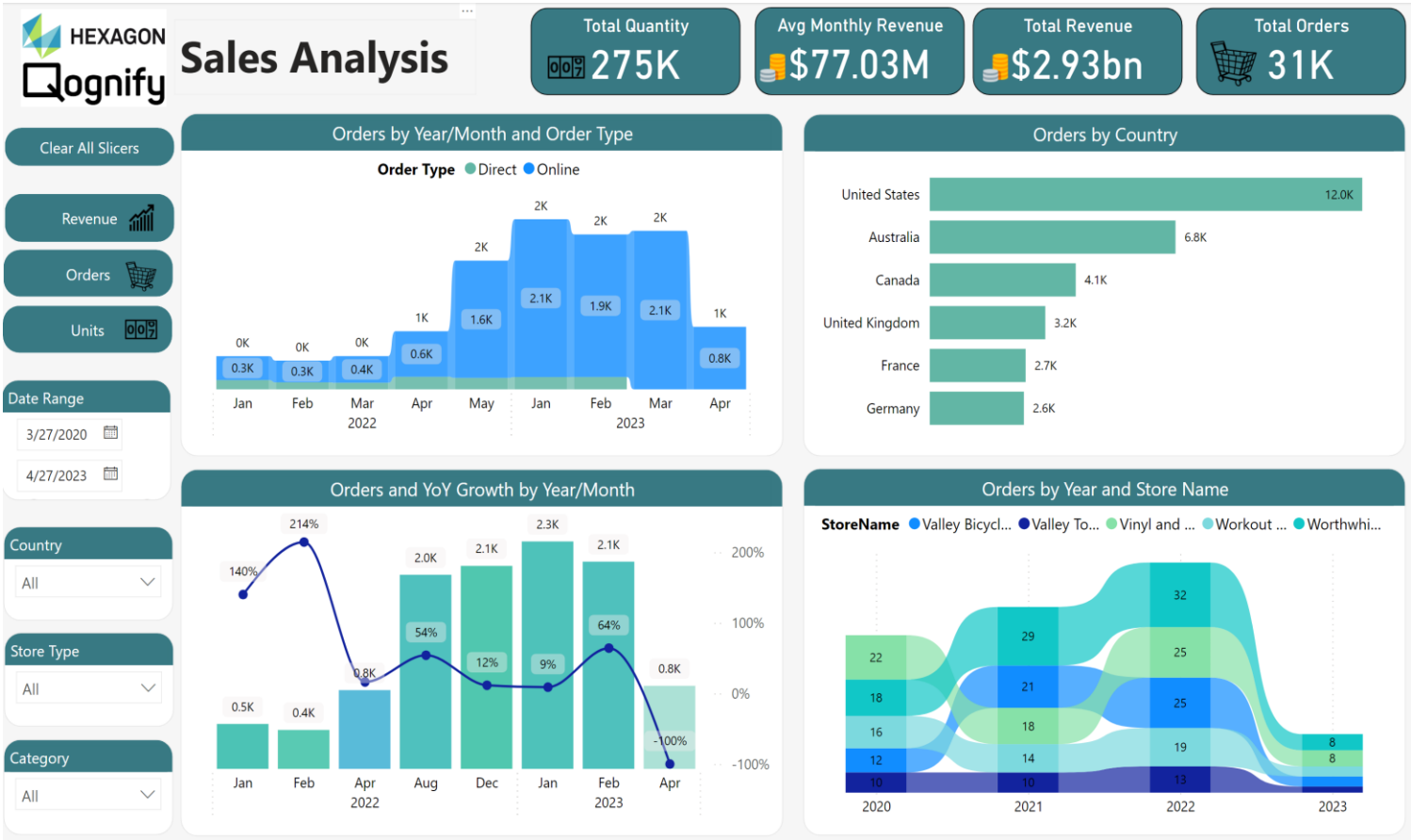
- Date Range

- Country
- Category
- Store Type

Graphs (revenue state):

- Revenue and YoY Growth by Year/ Month (Drill Mode)
- Revenue by Year/ Month (Drill Mode) and by Order Type
- Revenue by Product Country (Drill Mode)
- Revenue By Stores Name

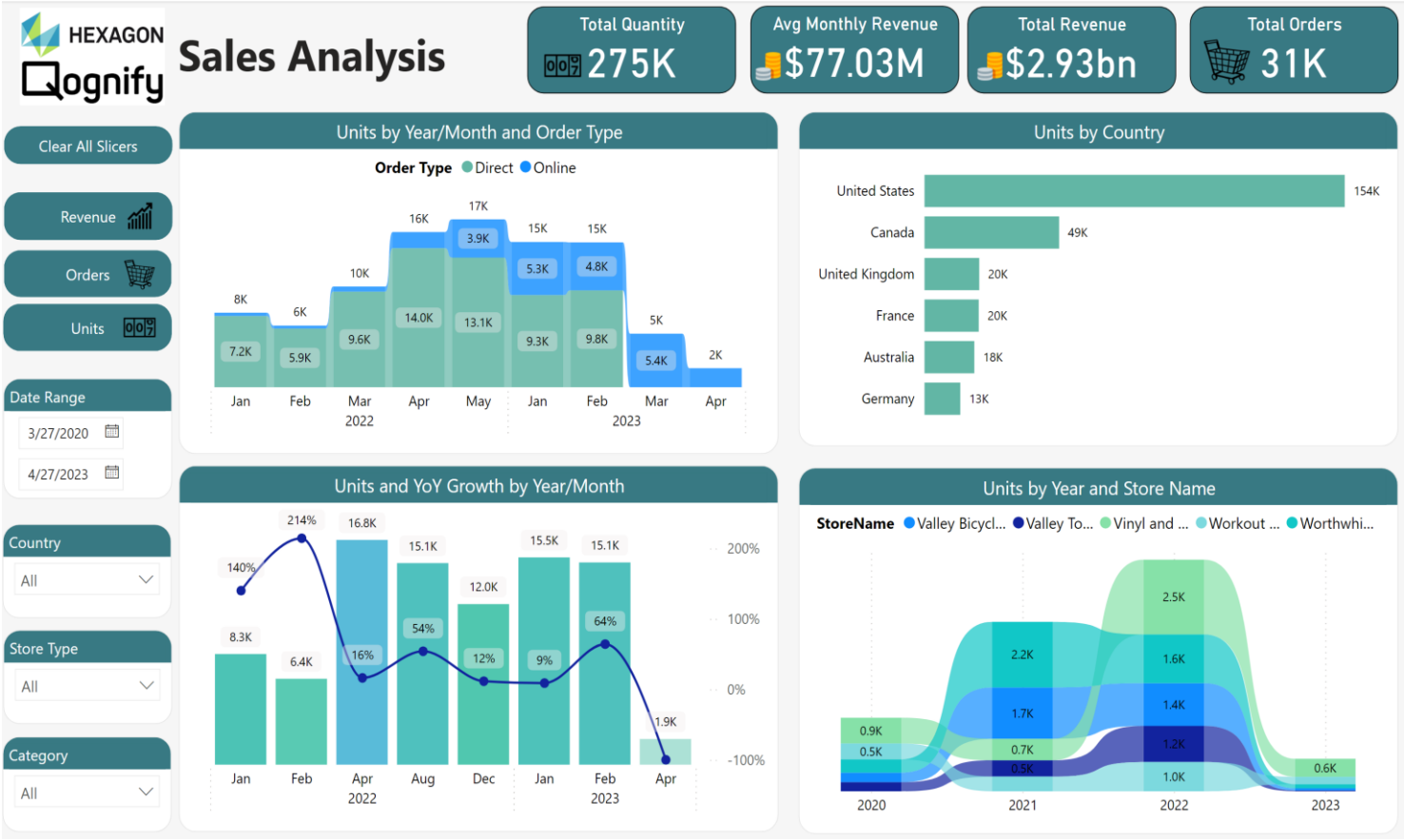
Orders State:



Graphs (Orders state):

- Orders and YoY Growth by Year/ Month (Drill Mode)
- Orders by Year/ Month (Drill Mode) and by Order Type
- Orders by Product Country (Drill Mode)
- Orders By Stores Name

Units State:

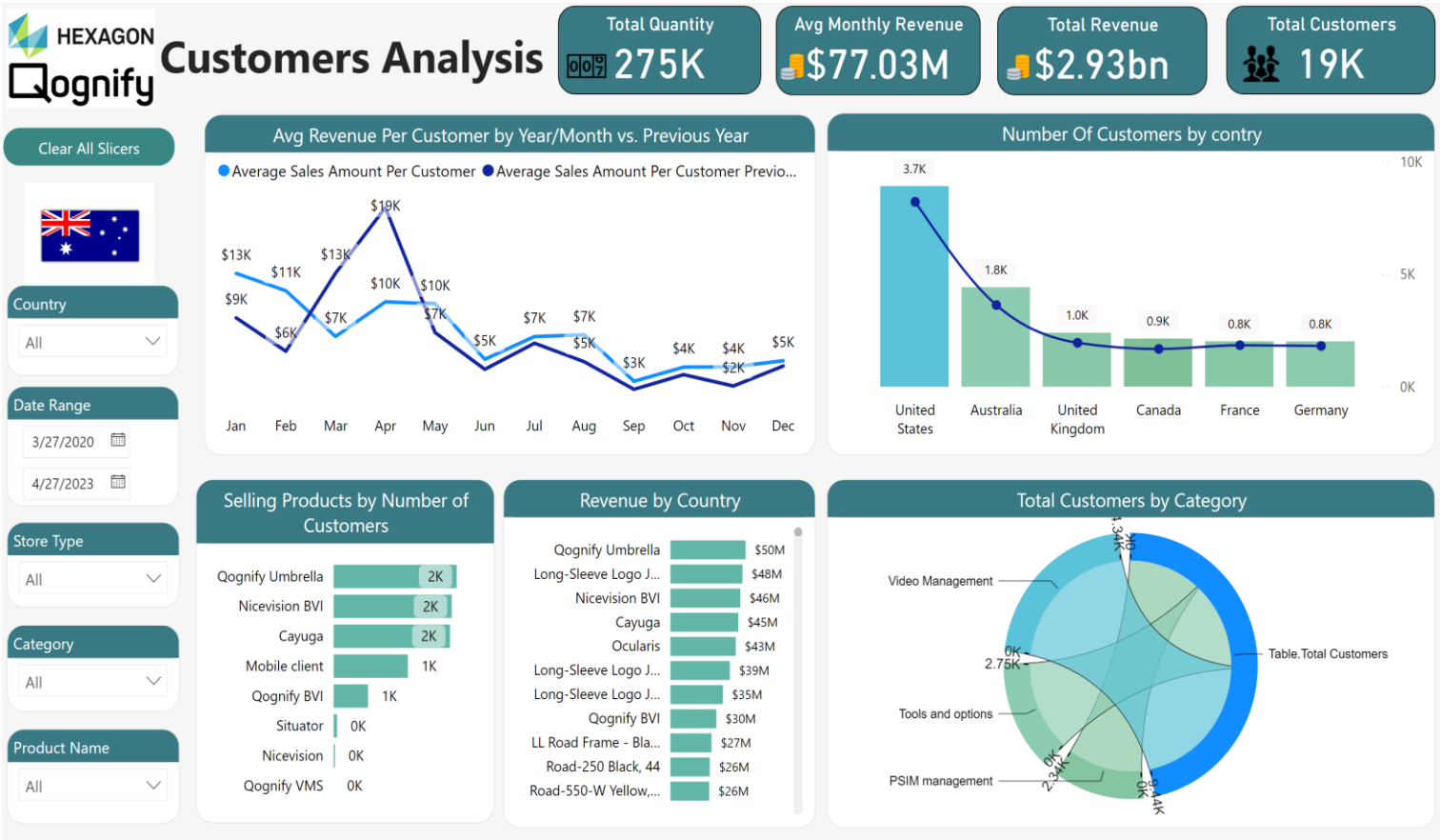


Graphs (Units state):

- Units and YoY Growth by Year/ Month (Drill Mode)
- Units by Year/ Month (Drill Mode) and by Order Type
- Units s by Product Country (Drill Mode)
- Units By Stores Name

4.3.3.3. Customer Analysis:

This report was created for the customers department to better understand Qgnify's customer behavior to achieve the department's goals.



KPI Cards:

- Total Revenue
- Total Orders
- Average Monthly Revenue
- Total Units

Graphs:

- Customers by Category
- Number Of Customers by Country
- Revenue by Country
- Selling Products by number of customers.
- Average Revenue per Customer by Month (Drill Mode) vs. Previous Year

Slicers:

- Date Range
- Country
- Product Name
- Category
- Store Type