SAMSUNG

בס"ד

BI System Specification Document

Date- 12.02.2024

Version: 4.0

Yael Atanelov

SAMSUNG

בס"ד

Table of Contents

1. General

1.1. Project Objective

The goal of the project is to develop a comprehensive Business Intelligence (BI) solution through a dedicated DM using data from the PriorityERP database for Samsung, a global leader in electronics and technology. It is designed to create a solution for Samsung's sales department to meet the company's goals.

The solution will encompass summarized data tables, with a specific focus on sales data, plus customer information, employee records, product details, territories, stores, dates, and other relevant data points. In addition, the project intends to improve data visualization by implementing dashboards and a reporting system. These tools are designed to effectively serve Samsung's management, department heads and sales managers, and provide them with clear insights derived from the PriorityERP database.

By analyzing customer data, this project can support Samsung in gaining a deeper understanding of customer preferences, purchasing behaviors and brand loyalty. This valuable information can be used to tailor more targeted marketing campaigns and strengthen the relationship with customers. In addition, insights into the best-selling products, their brands and colors can optimize production quantities for each model, leading to cost reductions.

1.2. Project Contents

In this project, we will build a Data Mart that will contain information about sales data.

- Main summary tables to be built for the company's demands:
  - Fact Orders – Contains Information about all the orders, including dates, pricing, and quantity per Customer and more. The data loading process for this table will be incremental.
  - Dim Products – Information about the products divided by categories and subcategories.
  - Dim Customers – Information about the company's customers.
  - Dim Employees – Information about the company's employees.
  - Dim Stores- Information about the company's Stores.

- The project will contain reports that will contribute to the achievement of the project's goal:
  - Sales Department Analysis:
    The Sales Department will focus on sales-related data, seller performance, and revenue while referring to different company stores. The company can use this information to identify sales trends by different years, reward productive employees, and compare results to other months to gain a better understanding of goal attainment.
    The report will present the following, among other things:
    - YTD revenue / Quantity
    - Average Revenue / Quantity Per Employee
    - Average Revenue / Quantity Per Order
    - Average Revenue / Quantity Per Month
    - Total Revenue / Quantity from each employee in the last year
    - Total num of Orders

- Customer Sales Analysis:
  This report focuses on client-side activity, providing data on different types of customers. This report enables a comprehensive understanding of customer behavior, including the frequency of returning customers, and their initial product preferences versus subsequent purchases.
  The report will present the following, among other things:
  - The number of new customers added in the last year.
  - The number of returning customers added in the last year.
  - Average Income Per Customer
  - Count Of Second Orders
  - Quantity in first orders
  - Count not Online Orders
- Management Dashboard:
  It will include various data that appear in the other reports that will be created in this project. The data that will be presented in it are the relevant data only for the high management level and its purpose will be to help make strategic decisions in the future.

2. Technical Specification

2.1. Attachment Files

- HLD (High Level Design)- see attachment file
- ERD- see attachment file
- Source To Target file- see attachment file
- Gantt- see attachment file

2.2. Prerequisites

the involved systems during the process

System / Process	Access Information
SQL Server	SQL Server: operational DB PriorityERP- tables, data USERNAME: ABC PASSWORD: 12345678
SSIS	C:\Users\100ya\OneDrive\Desktop\אקספיריס\פרוייקט גמר\הגשה\SSIS-PROD\Samsung_DM_MRR\Samsung_DM_PROD.sln
Data Refreshing	Refreshing processes through an attribute of Employees in SSMS using the JOB definition for each table separately
Power BI	The reports appear in a file named- **Samsung Project Reports.pbix** or in the application- **Samsung Project by Yael Atanelov** and are exposed to the employees of the organization at the administrative levels only.

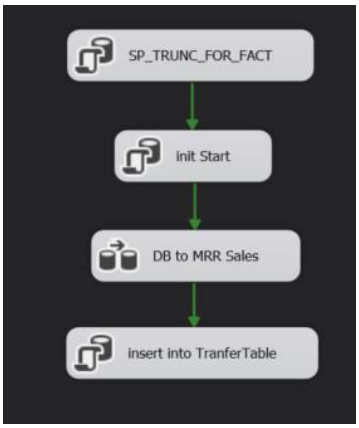
3. Technical Specification

3.1. ETL processes

In SSIS, I have 13 packages for all tables:

3.1.1. Fact Sales Table:

- MRR_Sales package:



in the Truncate SP:

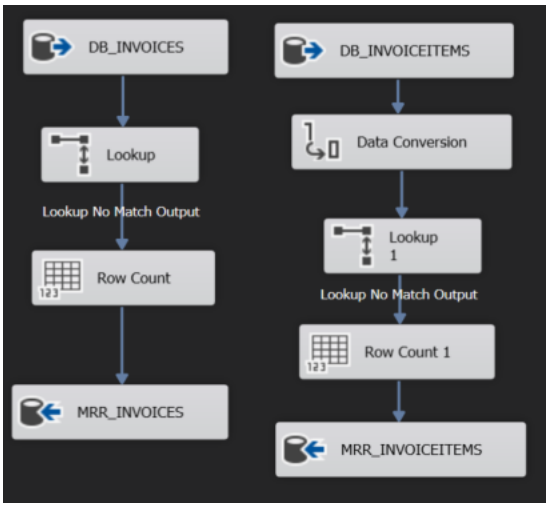
Running a procedure that activates a truncate operation on all the tables that lead to the creation of the **Fact_Sales** table

insert into Tranfer_Table:

The transfer table is a table that contains all the steps of adding from the DB step (OLTP) to the DWH step (OLAP). And in each record there is a record of the stage at which the addition was made.

The 'Init Start' task saves the exact date and time when the other tasks in this phase started running for the Tranfer_Table table.

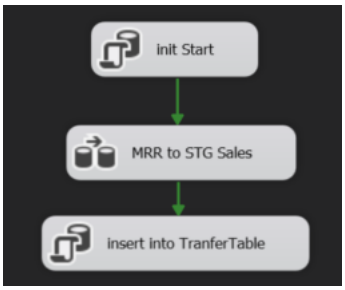
In the Data Flow (DB to MRR Sales):



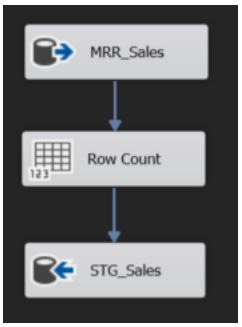
Here we load the data from the OLTP DB into the MRR table, it is about copying the data as it is in the source to MRR table. In the LOOKUP phase we filter the data that already exists in DWH and then count with the help of ROW COUNT the number of new records that will be updated.

The number of new records will be inserted into the Tranfer_Table table.

- STG_Sales package:

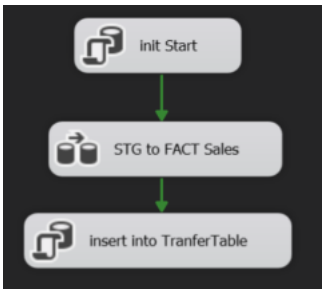


After performing the update in the STG phase, we will insert new records into the Tranfer_Table table that express the number of records transferred in this phase.
The 'Init Start' task saves the exact date and time when the other tasks in this phase started running for the Tranfer_Table table.
In the Data Flow (MRR to STG Sales):



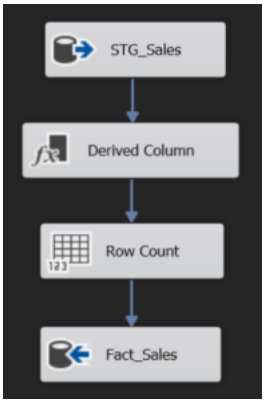
Initially, we perform a JOIN operation on the source that unites the relevant tables. This query generates an initial structure for the creation of the Fact_Sales table in the DW, then we will count the number of records that were transferred for the purpose of correctness checks in the Tranfer_Table and finally we will insert the records into the STG table.

- FACT_Sales package:



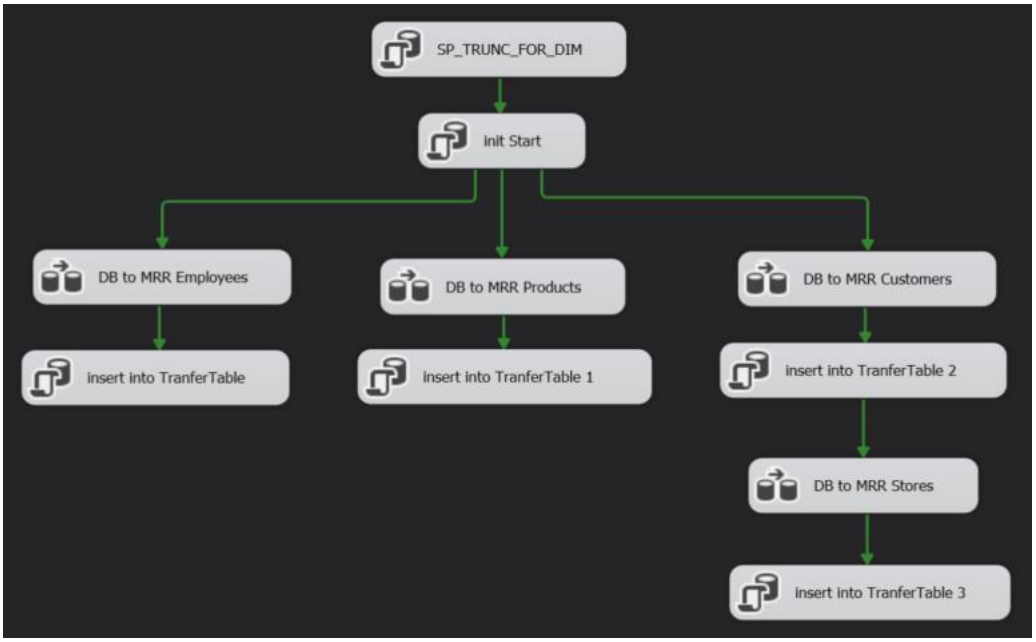
After performing the update in the last phase, we will insert new records into the Tranfer_Table table that express the number of records transferred in this phase.

- In the Data Flow (STG to FACT Sales):



The source table from which we will take the data is the table we created in the STG stage, we will count the records inserted in this stage and we will add columns to the target table using the **Derived Column** operation:
At this stage, a **Total** calculated field is created, which is the total payment for the sale after calculation of tax. In addition, the Data Type of other fields were changed at this stage for them to match the configuration required in Fact_Sales table in the DW.

3.1.2. MRR for all DIMs (MRR_DIMS package):

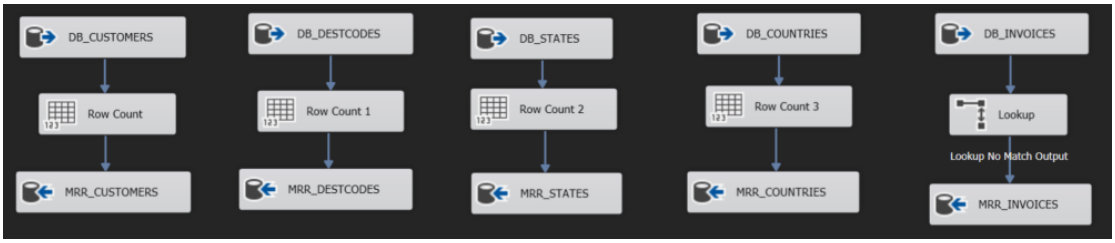


This package includes the initial loading for the four-dimension tables.
Their transition from the OLTP DB to the MRR stage. This is about loading all the fields in the source table as they are into the MRR tables.
For each one, the number of transferred records will be updated separately and at the end of the process we will insert the number of transferred records into the Tranfer_Table table for properness checks.
The 'Init Start' task saves the exact date and time when the other tasks in this phase started running for the Tranfer_Table table.

3.1.3. DIM Customers:

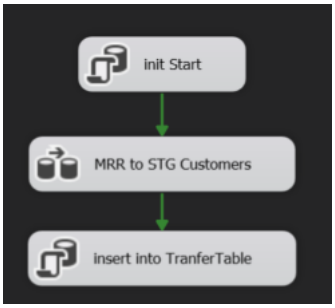
- Inside package MRR_DIMS:

  In the Data Flow (DB to MRR Customers):

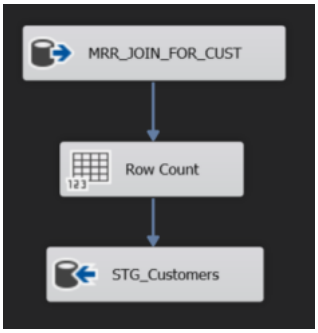


  In the MRR step, the data from the source tables in the OLTP DB is copied to the target tables in the MRR. There are 4 source and destination tables here because the DWH customer table consists of data from these 4 tables.
  Table INVOICES is an auxiliary table whose purpose is to link the 4 tables in the following steps. We will load it incrementally and each time we will load only new records into it.
  For each load for the four tables we will count the number of records transferred for tracking using the Tranfer_Table table.

- STG_ Customers package:

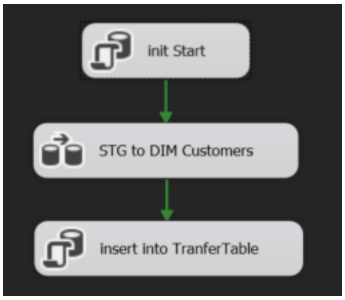


  After performing the update in the STG phase, we will insert new records into the Transfer_Table table expressing the number of records transferred in this phase and the time when the transformation occurred for the purpose of correctness checks.
  In the Data Flow(MRR to STG Customers):



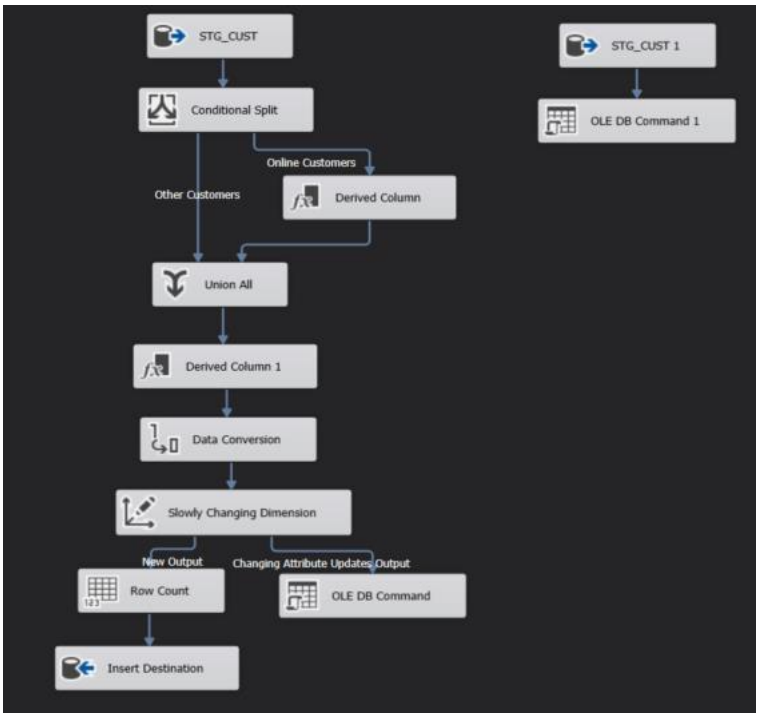
  Initially, we perform a JOIN operation on the source that unites the relevant tables. This query generates an initial structure to create the Dim_Customers table in DW.
  then we will count the number of records that have been transferred for correctness checks and finally we will insert the records into the STG_Customers table.

- DIM_ Customers package:



After performing the update in the DIM phase, we will insert new records into the Tranfer_Table table that express the number of records transferred in this phase for properness checks.

In the Data Flow(STG to DIM Customers):



When moving from the STG phase to the DIM phase, we will want to perform several operations on the data.

- Conditional Split- for separate the regular customers from the online customers.
- Derived Column- define the ID of the online store on the website for the customers who are defined as online customers.
- Union All- union all clients to perform the following actions on all of them.
- Derived Column 1- define a field that says that every record passed is considered an active customer (Is_Active=1).
- Data Conversion- change the DATA TYPE according to the rest of the model in DWH to create one organizational truth.
- Slowly Changing Dimension- Responsible for separating the records into new records and records that one of their fields has been updated.
- Insert Dimension- Inserts new records into the OLAP table.

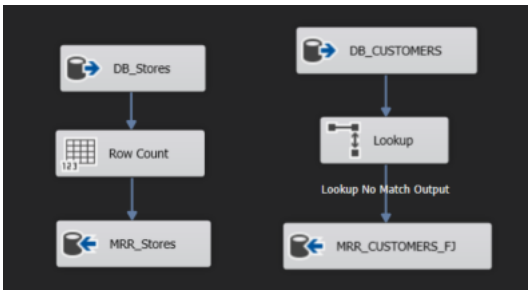
- **OLE DB Command**- Updates the existing records in the OLAP table according to the changes that occurred in the OLTP table.

On the left side you can see the update of records of customers who are not currently relevant (customers who have not placed an order in the last 3 years) in DWH. In the **OLE DB Command** field Is_Active is being updated (Is_Active=0).

3.1.4. DIM Stores:

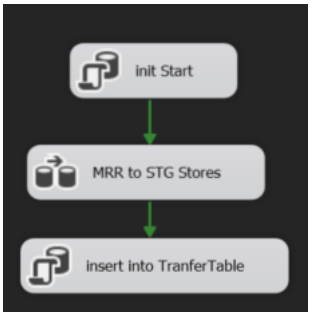
- Inside package MRR_DIMS:

In the Data Flow (DB to MRR Stores):



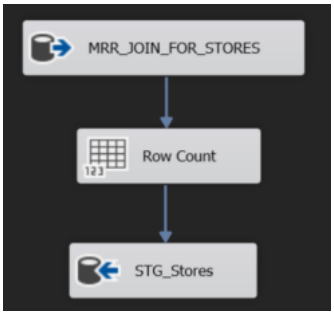
In the MRR step, the data from the source tables in the OLTP DB is copied to the target tables in the MRR. There are 2 source and destination tables here because the DWH customer table consists of data from these 2 tables.
Table COUSTOMERS is an auxiliary table whose purpose is to link the 2 tables in the following steps. We will load it incrementally and each time we will load only new records into it.
For each load for the four tables we will count the number of records transferred for tracking using the Tranfer_Table table.

- STG_ Stores package:



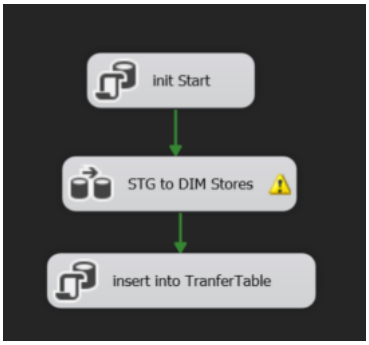
After performing the update in the STG phase, we will insert new records into the Transfer_Table table expressing the number of records transferred in this phase and the time when the transformation occurred for the purpose of correctness checks.

- In the Data Flow(MRR to STG Stores):



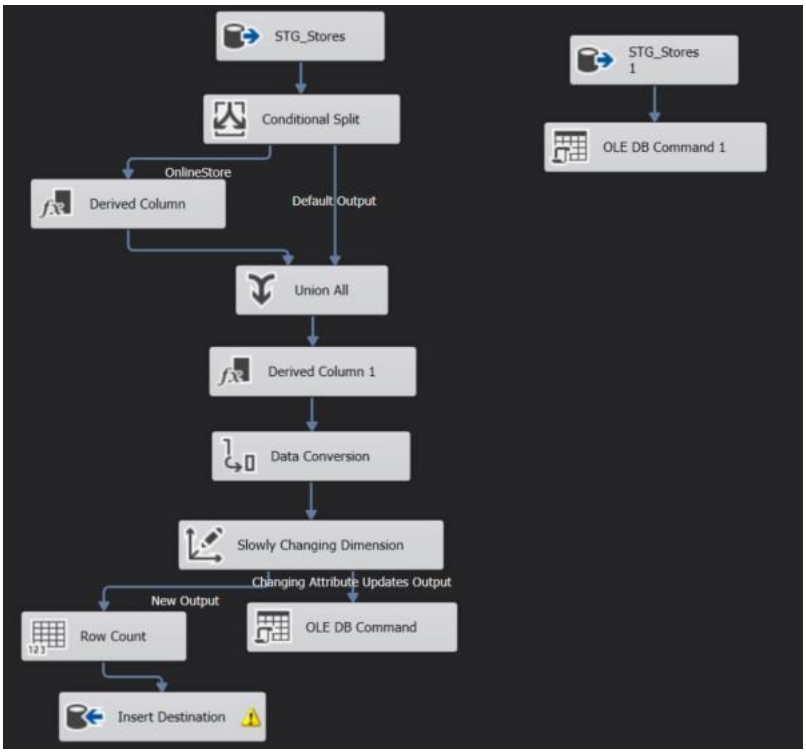
Initially, we perform a JOIN operation on the source that unites the relevant tables. This query generates an initial structure to create the Dim_Stores table in DWH.
then we will count the number of records that have been transferred for correctness checks and finally we will insert the records into the STG_Stores table.

- DIM_ Stores package:



After performing the update in the DIM phase, we will insert new records into the Tranfer_Table table that express the number of records transferred in this phase for properness checks.

In the Data Flow (STG to DIM Stores):



When moving from the STG phase to the DIM phase, we will want to perform several operations on the data.

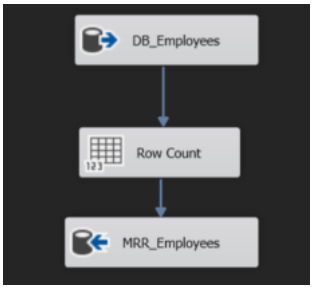
- Conditional Split- for separate the regular stores from the online stores.
- Derived Column- define the country and city fields of the online store to be 'Online'.
- Union All- union all stores to perform the following actions on all of them.
- Derived Column 1- define a field that says that every record passed is considered an active store (Is_Active=1).
- Data Conversion- change the DATA TYPE according to the rest of the model in DWH to create one organizational truth.
- Slowly Changing Dimension- Responsible for separating the records into new records and records that one of their fields has been updated.
- Insert Dimension- Inserts new records into the OLAP table.
- OLE DB Command- Updates the existing records in the OLAP table according to the changes that occurred in the OLTP table.

On the left side you can see the update of records of stores who are not currently relevant at the moment (as long as they exist in the original OLTP database) in DWH. In the OLE DB Command field Is_Active is being updated (Is_Active=0).

3.1.5. DIM Employees:

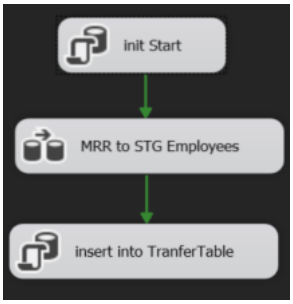
- Inside package MRR_DIMS:

  In the Data Flow (DB to MRR Employees):



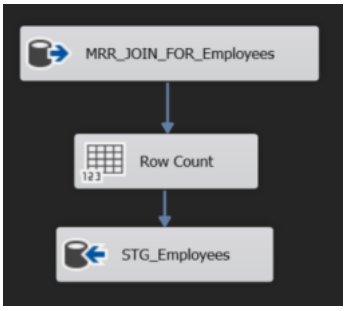
  From the employee table we will only bring the records that have been changed in any way- deleted or updated or new records inserted. This is with the help of the CDC_Employees table that we created, and records are inserted into it using triggers.

- STG_ Employees package:



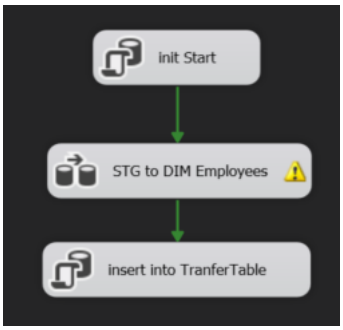
  After performing the update in the STG phase, we will insert new records into the Transfer_Table table expressing the number of records transferred in this phase and the time when the transformation occurred for the purpose of correctness checks.

  In the Data Flow (MRR to STG Employees):



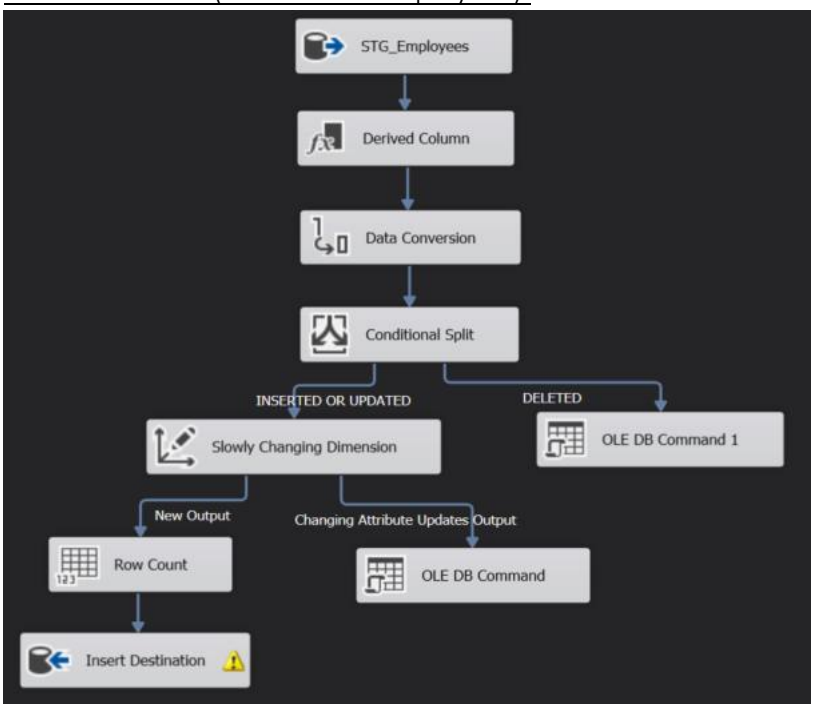
  At this stage we will filter from the MRR table only sales agents who are relevant to the DM we are creating (for a sales department).

DIM_ Employees package:



After performing the update in the DIM phase, we will insert new records into the Tranfer_Table table that express the number of records transferred in this phase for properness checks.

In the Data Flow(STG to DIM Employees):



When moving from the STG phase to the DIM phase, we will want to perform several operations on the data.

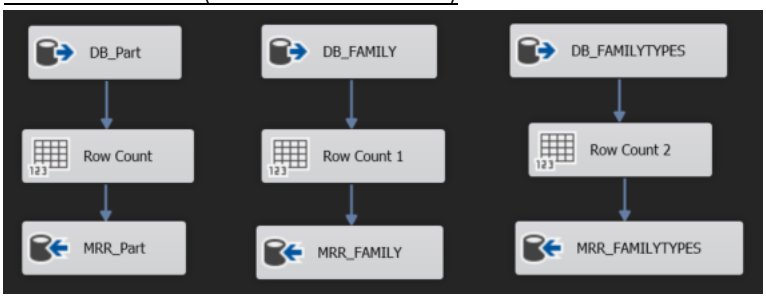
- Derived Column - define a field that says that every record passed is considered an active employee (Is_Active=1).
- Data Conversion- change the DATA TYPE according to the rest of the model in DWH to create one organizational truth.
- Conditional Split- With the help of a field produced in the CDC table that came from the source, it is possible to know the nature of the record change that was made. Accordingly, we can know which records have been deleted and move them to the appropriate path. The rest of the records go to SCD.
- Slowly Changing Dimension- Responsible for separating the records into new records and records that one of their fields has been updated.
- Insert Dimension- Inserts new records into the OLAP table.

- **OLE DB Command**- Updates the existing records in the OLAP table according to the changes that occurred in the OLTP table.
- **OLE DB Command 1**- field Is_Active is being updated (Is_Active=0).

3.1.6. DIM Products:

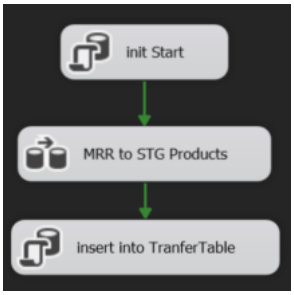
- Inside package MRR_DIMS:

In the Data Flow (DB to MRR Products):



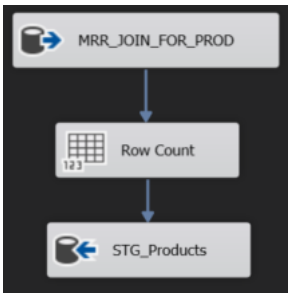
In the MRR step, the data from the source tables in the OLTP DB is copied to the target tables in the MRR. There are 3 source and destination tables here because the DWH customer table consists of data from these 3 tables.
For each load for the 3 tables we will count the number of records transferred for tracking using the Tranfer_Table table.

- STG_ Products package:



After performing the update in the STG phase, we will insert new records into the Transfer_Table table expressing the number of records transferred in this phase and the time when the transformation occurred for the purpose of correctness checks.

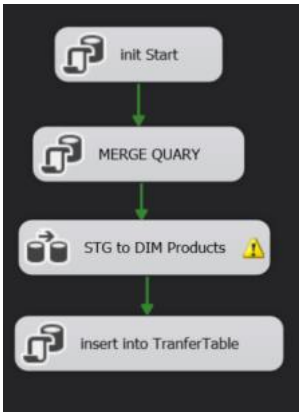
In the Data Flow (MRR to STG Products):



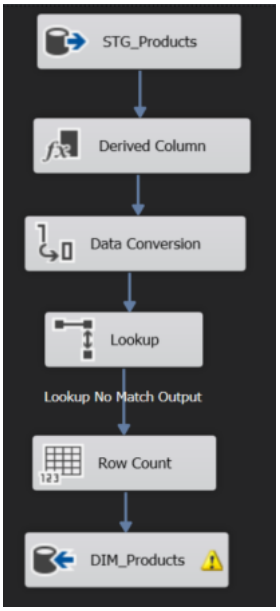
Initially, we perform a JOIN operation on the source that unites the relevant tables. This query generates an initial structure to create the Dim_Products table in DWH.
then we will count the number of records that have been transferred for

correctness checks and finally we will insert the records into the STG_Products table.

- DIM_ Products package:



First, using a MERGE query under MERGE QUERY, the new records that appear in the STG table will be checked.
if records have been deleted (there are records in DIM but not in STG), the deleted product will be updated in the DIM table to be Is_Active = 0. This is actually a FLAG that will indicate that a product is inactive.
If one of the fields in the product table has been updated, the product record will be updated in the DIM table accordingly.
For both cases, a field named DATE_UPDATE will be updated with the current date when the record was last updated.
After performing the update in the last phase, we will insert new records into the Tranfer_Table table that express the number of records transferred in this phase for properness checks.
In the Data Flow (STG to DIM Products):

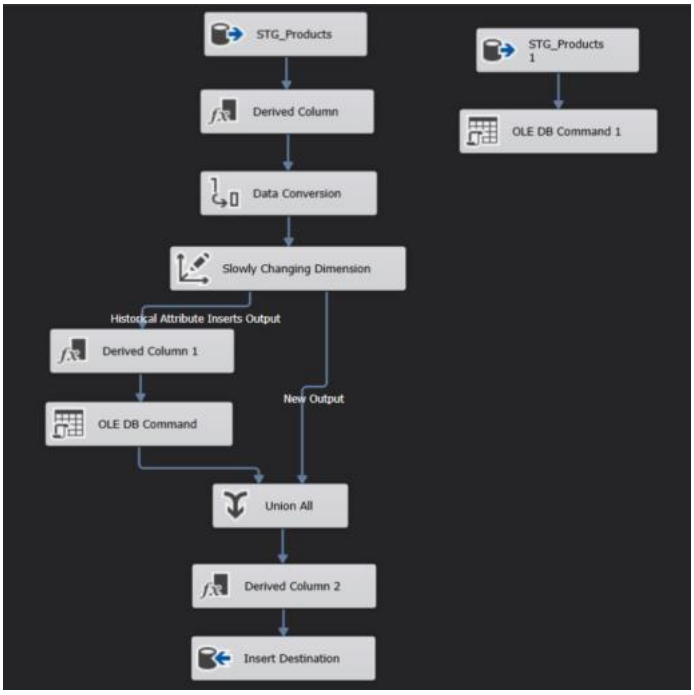


We would like to transfer only new records from the STG table to the DIM table.
In the **LOOKUP** phase, a check will be performed and only records whose ID does not exist in the Dim_Products table will be passed.
In the **Derived Column** step, we will add to the new records an Is_Active=1 field.

After, we will change Data Types according to the rest of the model in DWH to create one organizational truth.

STG_ Products_History package:

In the Data Flow (STG to DIM Products History):



Records are inserted into the dim_Products_History table. The table fields are identical to the fields of the dim_Products table and in addition there are two date fields that specify a start date and an end date for a certain version.
When updating a record in the dim_Products table, the previous version of the record will be saved in the dim_Products_History table together with date fields that constitute the time range in which the version was relevant.

3.2. Tranfer_Table:

The table will be created in the SQL Server by the following query:

```
create table Transfer_Table
Package_Name nvarchar 100,
Table_Name nvarchar 100,
EndDate datetime DEFAULT GETDATE,
NewRows int
)
```

The table is updated during the Control Flow as shown and detailed in the previous screenshots shown in the file.
All the loading steps MRR, STG, DIM/FACT lead to updating records in the Transfer_Table table according to the changes made during the process.

3.3. Defining JOBs in SSMS

To streamline the daily refresh and loading procedures, an SSIS deployment was initiated to SSMS. Following this, a scheduled job was established to run daily at a specific time (SA_MRR) After activating the steps in it, the last step will activate the loading of the next JOB and so on.
Error-handling protocols have been established to suspend the process in the event of an error, guaranteeing data integrity and dependability.
The JOBS listed here are linked to SSIS and will run daily:

The jobs are configured to run packages that appear in SSIS in a predefined order as shown here:

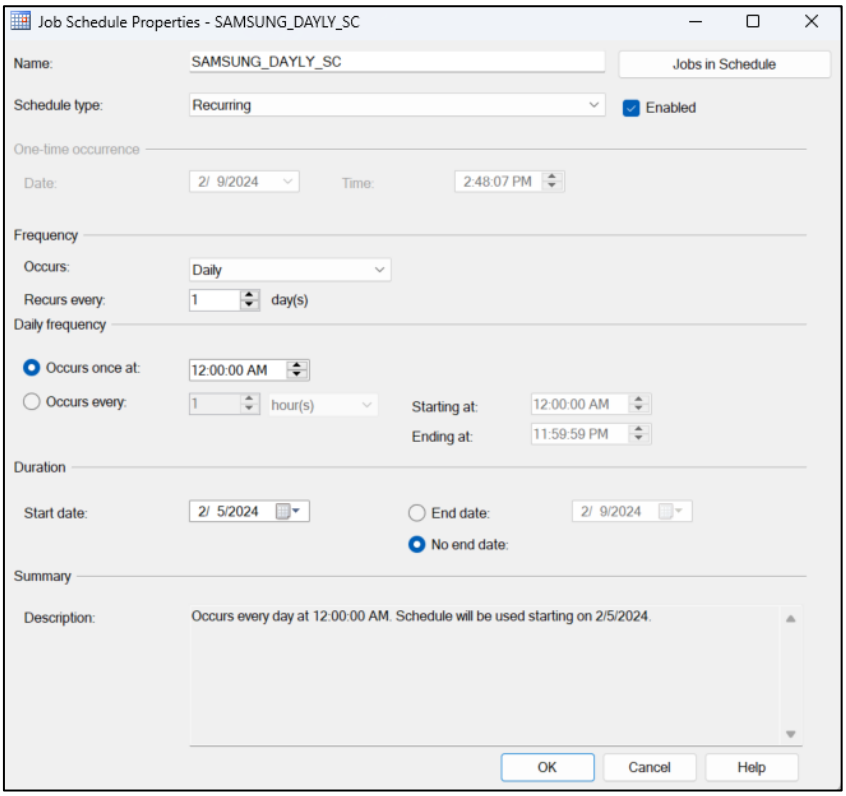
- SA_MRR- contains the package that runs the MRR_DIMS package and MRR_Sales package.
- SA_DIM_Emploees- contains the packages: STG_Employees and Dim_Employees when STG_Employees will run first.
- SA_DIM_Stores- contains the packages: STG_Stores and Dim_Stores when STG_Stores will run first.
- SA_DIM_Customer- contains the packages: STG_Customers and Dim_Customers when STG_Customers will run first.
- SA_DIM_Product- contains the packages: STG_Products, Dim_Products and Dim_Products_History when STG_Products will run first.
- SA_Fact_Sales- contains the packages: STG_Sales and Fact_Sales when STG_Sales will run first.

If the packages do not run as required and the run fails, we will receive a report that will help us understand where the run failed.

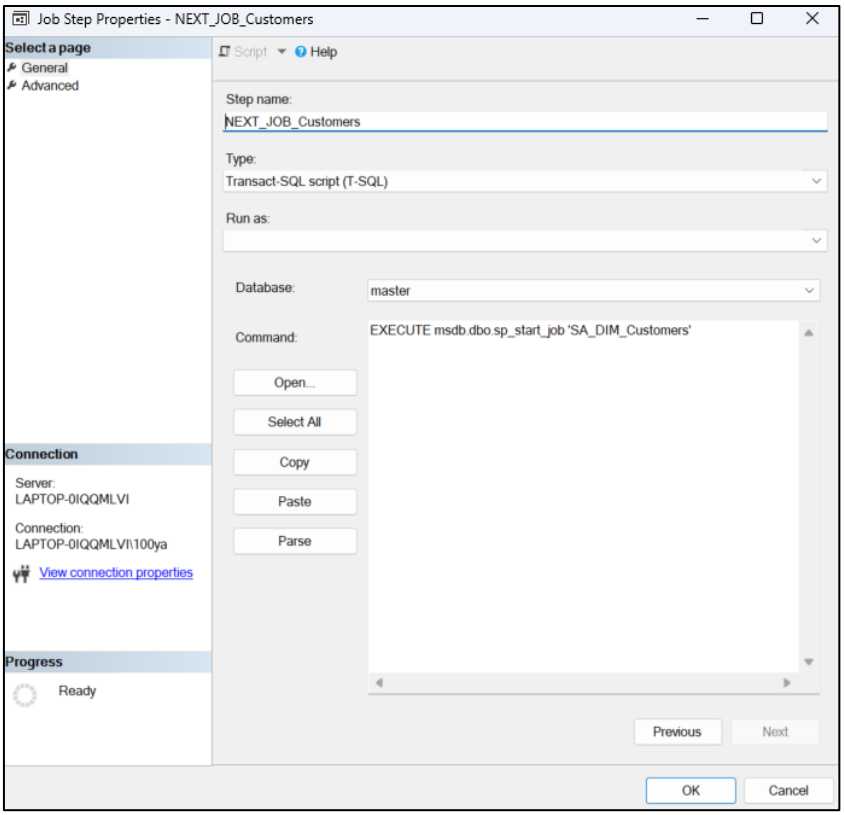
The tables are scheduled for daily updates at midnight (00:00). This involves running packages within SSIS using the SQL Server Agent configured in SSMS. The SQL Server Agent is responsible for executing jobs at 00:00, triggering the packages to run in the specified order.

בס"ד

The definition of scheduling is only in the JOB that will run first- SA_MRRS:

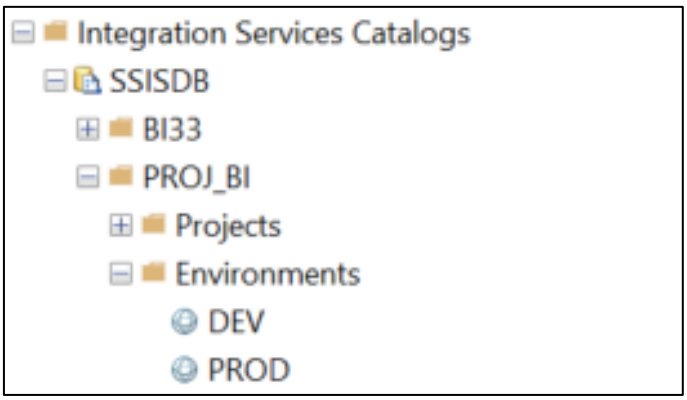


His last step will run the following JOB using the following code:



3.4. Work Environments

The project is divided into two work environments, Production (PROD) environment and Development (DEV) environment.
The defined JOBS will always run on the PROD environment every day according to the settings we made.
The DEV environment is intended for development purposes where we will try to make changes to streamline the processes and find solutions to system users' faults.
Only after in-depth tests of the DEV environment we can move it to the PROD as well as for daily running for the needs of the organization.



3.5. Description of data tables in the Data Warehouse (DWH):

3.5.1. Fact_Sales:

In the Fact_Sales table, we have the following columns:

1. Order_ID: The unique identifier for each order, serving as the primary key for the table.
2. Order_Date: The date when the order was placed, providing a timestamp for each transaction.
3. Customer_ID: The identifier for the customer associated with the order, serving as a foreign key referencing the Dim_Customers table.
4. Agent_ID: The identifier for the employee responsible for making the order, serving as a foreign key referencing the Dim_Employees table.
5. Territory_ID: The identifier for the geographical territory where the order was placed.
6. Product_ID: The identifier for the products purchased in the order, serving as a foreign key referencing the Dim_Products table.
7. Qty: The quantity of each product purchased in the order.
8. Price: The unit price of each product, providing the cost per unit.
9. Discount: The discount applied to each unit, influencing the overall cost.
10. Total: The total cost for each product purchased, inclusive of taxes (calculated as: Price * Qty * (1- Discount) * 1.17). This represents the final amount to be paid for the specified quantity of each product, factoring in the discount and applicable taxes.

3.5.2. Dim_Products:

In the Dim_Products table, we have the following columns.

1. Product_ID: The unique identifier and primary key for each product, serving as the key reference in the table.
2. Product_Name: The name of the product, providing a clear and concise label for each item.
3. Sub_Category_Name: The name of the subcategory to which the product belongs, offering additional categorization.
4. Category_Name: The name of the main category to which the product belongs, providing a higher-level classification.
5. Is_Active: A binary indicator (True/False) denoting the current active status of the product within the organization.

3.5.3. Dim_Products_History:

In the Dim_Products table, we have the following columns.

6. Product_ID: The unique identifier and primary key for each product, serving as the key reference in the table.
7. Product_Name: The name of the product, providing a clear and concise label for each item.
8. Sub_Category_Name: The name of the subcategory to which the product belongs, offering additional categorization.
9. Category_Name: The name of the main category to which the product belongs, providing a higher-level classification.
10. Insert_Date: The date when we made any change to the product, such as inserting a new product, changing a characteristic of this product, or deleting the product.
11. End_Date: The date when a new change occurs. If it is null, that means it is the last update about the product.

3.5.4. Dim_Customers:

In the Dim_Customers table, we have the following columns

1. Customer_ID: The unique identifier and primary key for each customer, serving as a key reference within the table.
2. Name: The full name of the customer, providing comprehensive identification.
3. Store_ID: The identifier for the store associated with the customer, serving as a foreign key referencing the Dim_Stores table.
4. Address: The customer's street address, offering specific location details.
5. City: The city where the customer resides, providing information about the customer's urban location.
6. Region: The region associated with the customer, offering additional geographical context.
7. Country: The country of residence for the customer, indicating the nation from which the customer originates.
8. Is_Active: A binary indicator (True/False) denoting the current active status of the customer within the organization.

3.5.5. Dim_Stores:

In the Dim_Employees table we have the following column:

1. Store_ID: The unique identifier and primary key for each store, serving as a key reference within the table.
2. Store_Name: The name of the store, offering personal identification.
3. Region: The region associated with the customer, offering additional geographical context.
4. Country: The country of residence for the customer, indicating the nation from which the customer originates.
5. Is_Active: A binary indicator (True/False) denoting the current active status of the customer within the organization.

3.5.6. Dim_Employees:

In the Dim_Employees table we have the following column:

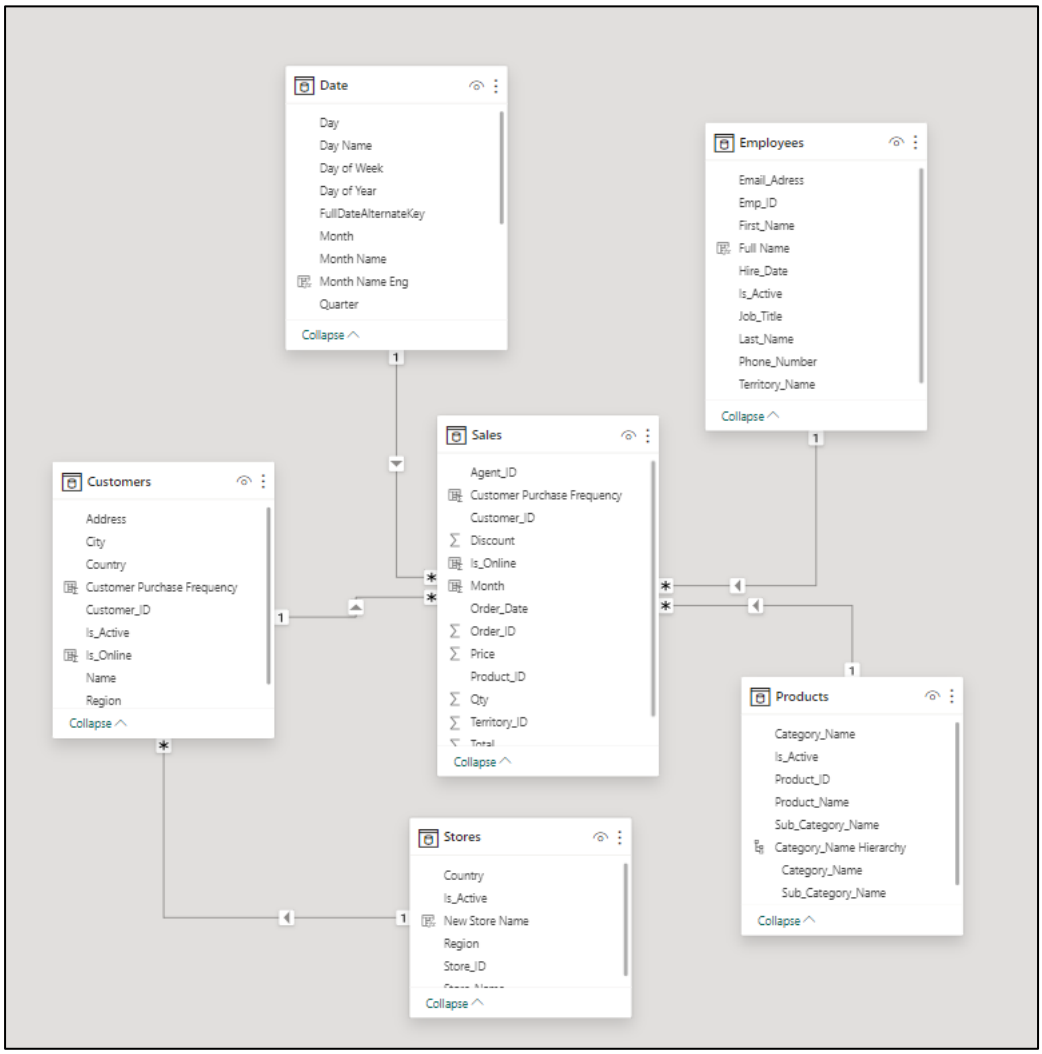
6. Emp_ID: The unique identifier and primary key for each employee, serving as a key reference within the table.
7. First_Name: The first name of the employee, offering personal identification.
8. Last_Name: The last name of the employee, completing the full name for accurate identification.
9. Job_Title: The job title or position held by the employee within the organization.
10. Hire_Date: The date when the employee was hired, providing insight into their tenure.
11. Phone_Number: The contact number associated with the employee, facilitating communication.
12. Email_Address: The email address of the employee, serving as a primary mode of professional communication.
13. Territory_Name: The name of the territory or area associated with the employee's responsibilities.
14. Is_Active: A binary indicator (True/False) denoting the current active status of the employee within the organization.

4. Power BI Reports

After constructing the Data Mart, our next step involves generating reports for the sales department's managers. These reports aim to extract insights from the data, leading organizational changes aligned with our strategic objectives and ultimately improving the company's profitability.

The reporting system includes a central dashboard designed for Samsung's global sales department, supplemented by two additional reports analyzing customer data and stores performance, global and online.

Data Mart's Snow Flack schema:



4.1. Main Measures

In the Power BI program, there are several calculations to create measures in the DAX language for the purpose of presenting the data according to the organization's needs.

1. Average Quantity per Emp = AVERAGEX(FILTER('Sales Aggregated by Emp','Sales Aggregated by Emp'[Year]=2023),'Sales Aggregated by Emp'[Total Qty])
2. Average Quantity per Month = AVERAGEX(FILTER('Sales Aggregated','Sales Aggregated'[Year]=2023),'Sales Aggregated'[Total Qty])
3. Average Quantity per Order = AVERAGEX(FILTER('Sales Aggregated by order','Sales Aggregated by order'[Year]=2023),'Sales Aggregated by order'[Total Qty])
4. Average Revenue per Emp = AVERAGEX(FILTER('Sales Aggregated by Emp','Sales Aggregated by Emp'[Year]=2023),'Sales Aggregated by Emp'[Total Revenue])
5. Average Revenue per Month = AVERAGEX(FILTER('Sales Aggregated','Sales Aggregated'[Year]=2023),'Sales Aggregated'[Total Revenue])
6. Average Revenue per Order = AVERAGEX(FILTER('Sales Aggregated by order','Sales Aggregated by order'[Year]=2023),'Sales Aggregated by order'[Total Revenue])
7. Average Income per Customer =
   VAR TotalIncome = SUM('Sales'[Total])
   VAR DistinctCustomers = DISTINCTCOUNT('Sales'[Customer_ID])
   RETURN
   DIVIDE(TotalIncome, DistinctCustomers)
8. Delta Quantity Months = IF(AND([Total Quantity],[Total Quantity Pervious Month]), ([Total Quantity Pervious Month]-[Total Quantity]))
9. Delta Revenue Months = IF(AND([Total Revenue],[Total Revenue Pervious Month]), ([Total Revenue Pervious Month]-[Total Revenue]))
10. Total Quantity = SUM('Sales'[Qty])
11. Total Quantity 2022 = CALCULATE(SUM('Sales'[Qty]),YEAR('Sales'[Order_Date]) = 2022)
12. Total Quantity 2023 = CALCULATE(SUM('Sales'[Qty]), YEAR('Sales'[Order_Date]) = 2023)
13. Total Quantity Pervious Month = CALCULATE([Total Quantity], DATEADD('Date'[FullDateAlternateKey], -1, MONTH))
14. Total Revenue = SUM('Sales'[Total])
15. Total Revenue 2022 = CALCULATE(SUM('Sales'[Total]),YEAR('Sales'[Order_Date]) = 2022)
16. Total Revenue 2023 = CALCULATE(SUM('Sales'[Total]), YEAR('Sales'[Order_Date]) = 2023)
17. Total Revenue Pervious Month = CALCULATE([Total Revenue], DATEADD('Date'[FullDateAlternateKey], -1, MONTH))
18. YTD Quantity = VAR CurrentYearStartDate = DATE(YEAR(TODAY())-1, 1, 1)
    VAR TotalSales = SUM('Sales'[Qty])
    RETURN CALCULATE(SUM('Sales'[Qty]), FILTER(ALL('Sales'),'Sales'[Order_Date] <= MAX('Sales'[Order_Date]) && 'Sales'[Order_Date] >= CurrentYearStartDate))
19. YTD Revenue = VAR CurrentYearStartDate = DATE(YEAR(TODAY())-1, 1, 1) VAR TotalSales = SUM('Sales'[Total]) RETURNCALCULATE(
    SUM('Sales'[Total]), FILTER(ALL('Sales'),'Sales'[Order_Date] <= MAX('Sales'[Order_Date]) && 'Sales'[Order_Date] >= CurrentYearStartDate))
20. Count not Online Orders = CALCULATE(DISTINCTCOUNT(Sales[Order_ID]),Sales[Is_Online]=0)
21. Count of New Orders =
CALCULATE(DISTINCTCOUNT('Sales Aggregated by order'[Customer_ID]),'Sales Aggregated by order'[Order Number] = 1)
22. Count of Second Orders =
    CALCULATE(COUNTROWS('Sales Aggregated by order'),'Sales Aggregated by order'[Year] = 2023 &&'Sales Aggregated by order'[Order Number] = 2)
23. Count Online Orders =

```
    CALCULATE(DISTINCTCOUNT(Sales[Order_ID]),Sales[Is_Online]=1)
24. Cur Date = MAX(Sales[Order_Date])
25. New Customers =
    VAR CurrentCustomers = VALUES(Sales[Customer_ID])
    VAR PastCustomers = CALCULATETABLE(VALUES(Sales[Customer_ID]),ALL('Date'[Month Name
    Eng],'Date'[Month],'Date'[Year]),'Date'[FullDateAlternateKey] < DATE(max('Date'[Year]),1,1))
    VAR RepeatCustomers = EXCEPT(CurrentCustomers,PastCustomers)
    RETURN
    COUNTROWS(RepeatCustomers)
26. Quantity  not in first order =
    CALCULATE(
  SUM(Sales[Qty]),Sales[Order Number]>1)
27. Quantity in first order =
    CALCULATE(
      SUM(Sales[Qty]),Sales[Order Number]=1)
28. Repeat Customers2 =
    VAR CurrentCustomers = VALUES(Sales[Customer_ID])
    VAR PastCustomers = CALCULATETABLE(VALUES(Sales[Customer_ID]),ALL('Date'[Month Name
    Eng],'Date'[Month],'Date'[Year]),'Date'[FullDateAlternateKey] < DATE(max('Date'[Year]),1,1))
    VAR RepeatCustomers = INTERSECT(CurrentCustomers,PastCustomers)
    RETURN
    COUNTROWS(RepeatCustomers)
29. Revenue in first order =
    CALCULATE(
      SUM(Sales[Total]),Sales[Order Number]=1)
30. Revenue not in first order =
CALCULATE(
      SUM(Sales[Total]),Sales[Order Number]>1)
31. Total num of Orders = DISTINCTCOUNT(Sales[Order_ID])
32. Total returning customers = SUM(Customers[Customer Purchase Frequency])
33. Year For Big Title =
    VAR GetValues= CONCATENATEX(
    VALUES(Sales[Year]), Sales[Year], ", ")
    RETURN
    "Customer Sales Analysis For- " & GetValues & " "
```

4.2. Dashboard and reports

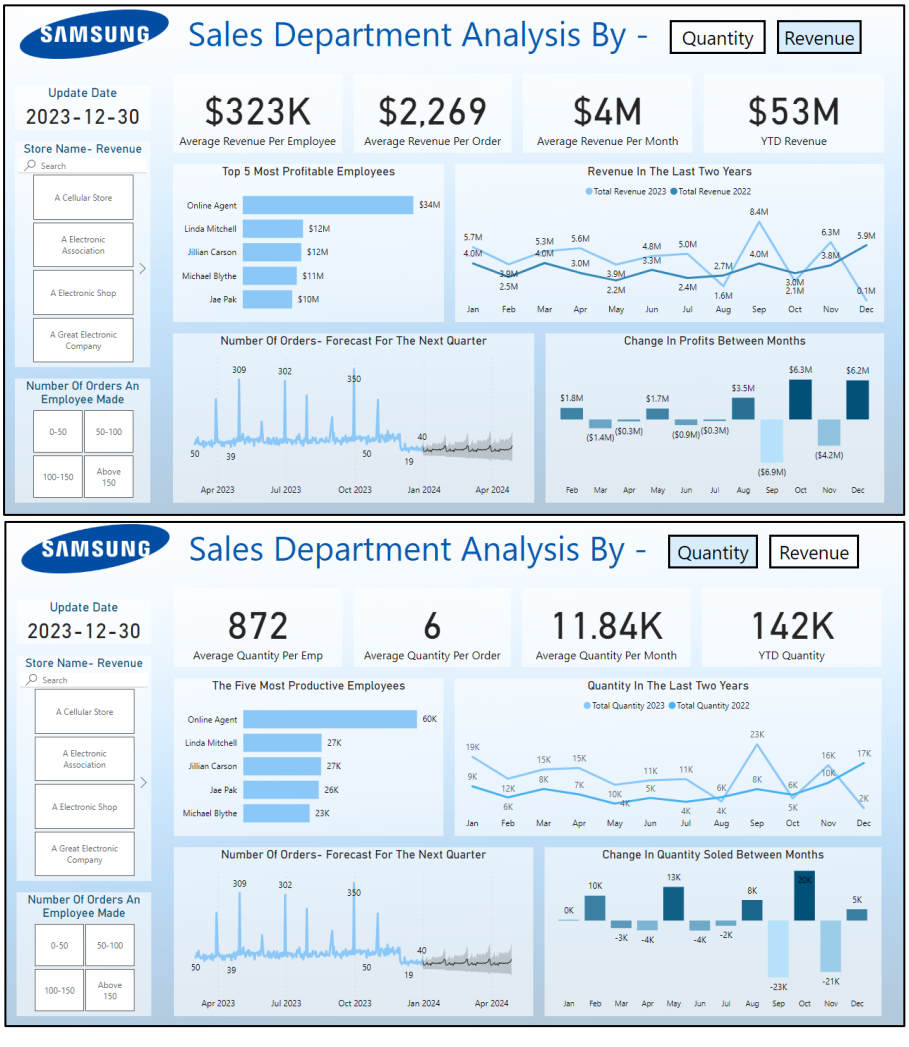
4.2.1. Sales Department Analysis:

The presentation of the relevant information for the sales department is divided into two- the quantity sold and the profit for the company. It is possible and even desirable to examine the two relevant types of information separately when looking at the report by clicking on the buttons at the top of the report.

The purpose of the report is to examine the performance of the company's various stores and their employees. The user can look at the report in a super level and see the performance of all stores together or search for a specific store and examine its performance. In addition, employees can be segmented according to the number of orders they made in the last year.

The main measures appearing in the report deal with the average profit/quantity in the last year, average profit/quantity for one order, average profit/quantity per month in the last year and what profit/quantity we have reached since the beginning of this year.

We will see in the report the forecast for the number of orders that will be placed in the next quarter, what were the differences between the months since the beginning of the year, differences between the sales trend this year and the previous year for the purpose of analyzing and checking seasonality and the most productive employees in the organization for the purpose of business decisions such as granting a promotion or raising the salary accordingly.



4.2.2. Customer Sales Analysis:

The display of information for the company's customers.

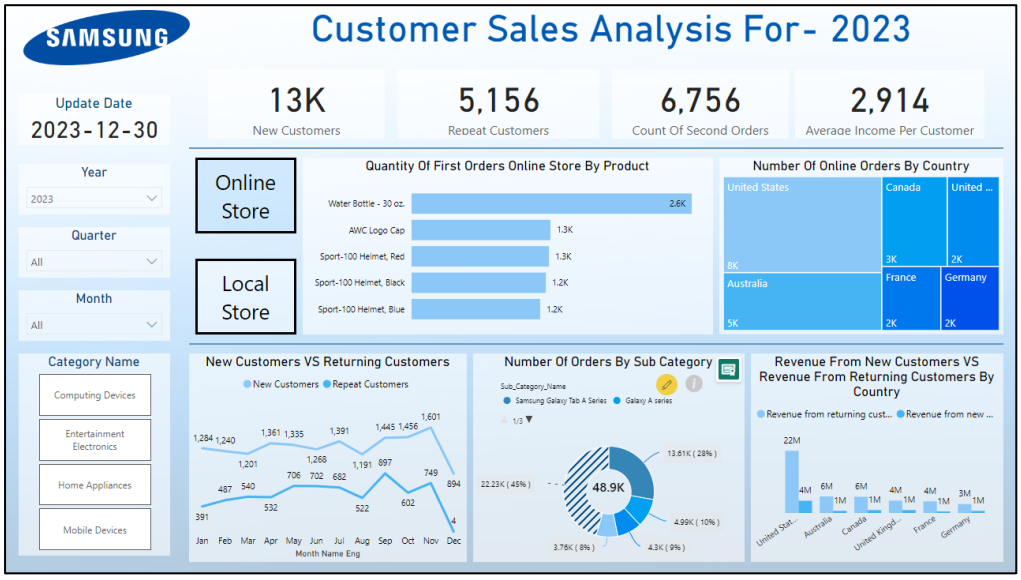
By choosing a specific year and segmenting according to the various categories, we can see how many new customers bought products from this category, how many customers bought from the category in a purchase that was not their first purchase, the number of second orders that were made and the average income from a customer.

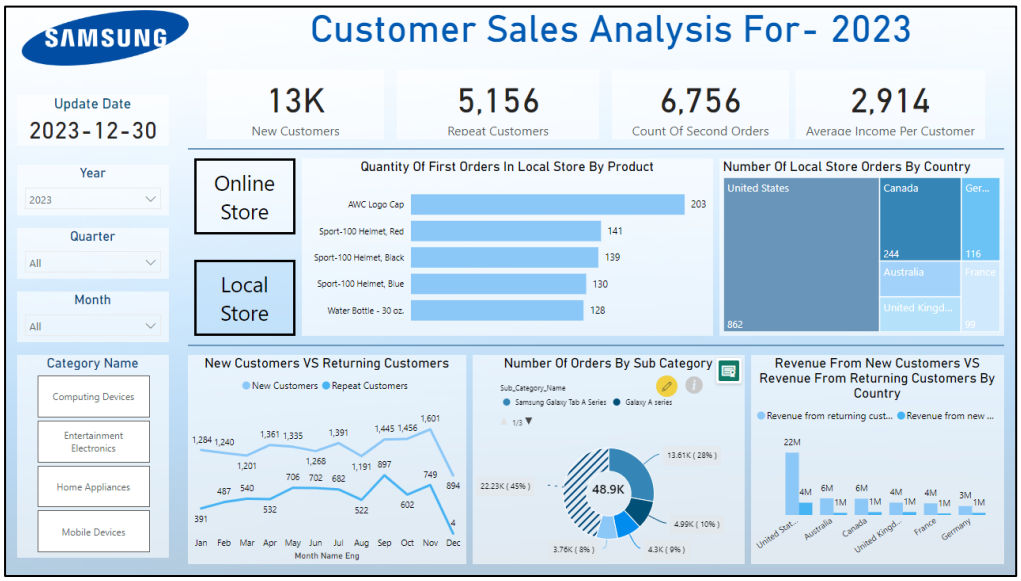
We would like to analyze the preferences of customers in different countries and which products they prefer to buy in the online store compared to a local store.

We can check by segmenting categories- which are the preferred subcategories.

And we would like to see a trend in the number of new customers versus how many returning customers bought Samsung products.

The data can be presented in the report according to a specific year, quarter, month, and category according to needs.





4.2.3. Management Dashboard:

The management report will present an overview of all the company's KPIs:

Total Quantity, Total number of orders, Average revenue per order, ARPU (Average Revenue Per User), ARR (Annual Recurring Revenue), MRR (Monthly Recurring Revenue), New customers, Total revenue, Top profitable products.

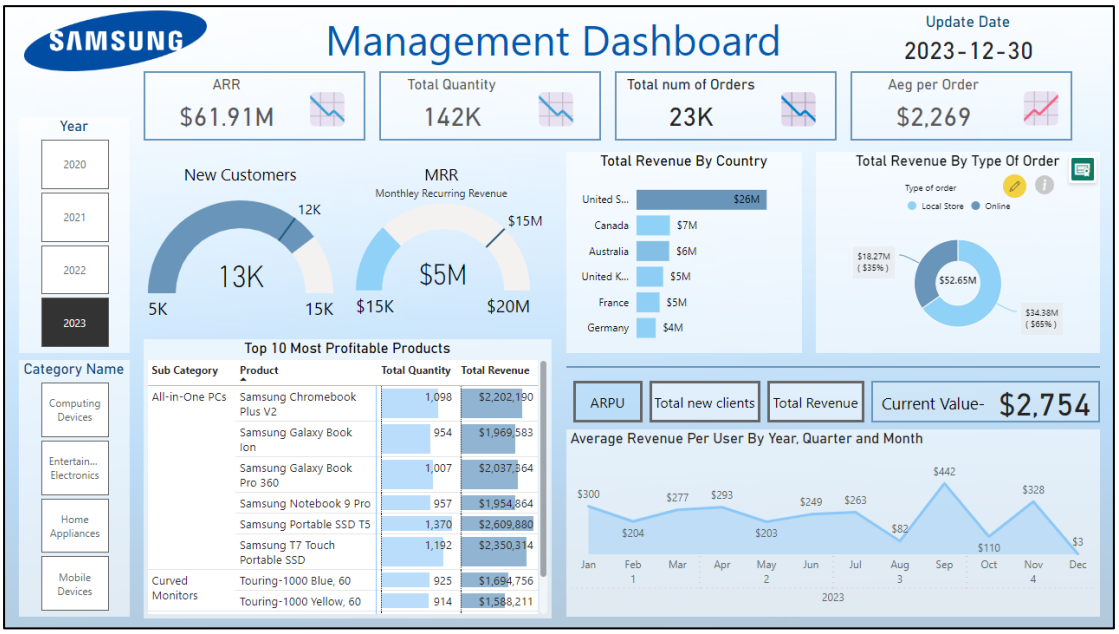
Some of the measures in the report will show the general trend of the index in the last six months- is the measure in a decreasing or increasing state.

In the MRR measure and the number of new customers we can see the actual amount against the company's target, the darker the color the better the situation.

For the measures APRU, Total new clients, Total revenue we can see a graphic display that shows a trend and the ability to see each index separately and the current value of the index (according to Bookmarks settings). The top ten profitable products are shown, what is the profit and what is the quantity bought from them.

We will be able to understand in which countries the products of the Samsung company are more popular and whether the customers prefer to purchase the products in the online store or in a physical store.

The entire report can be displayed for a cross-section of years or for a product category.

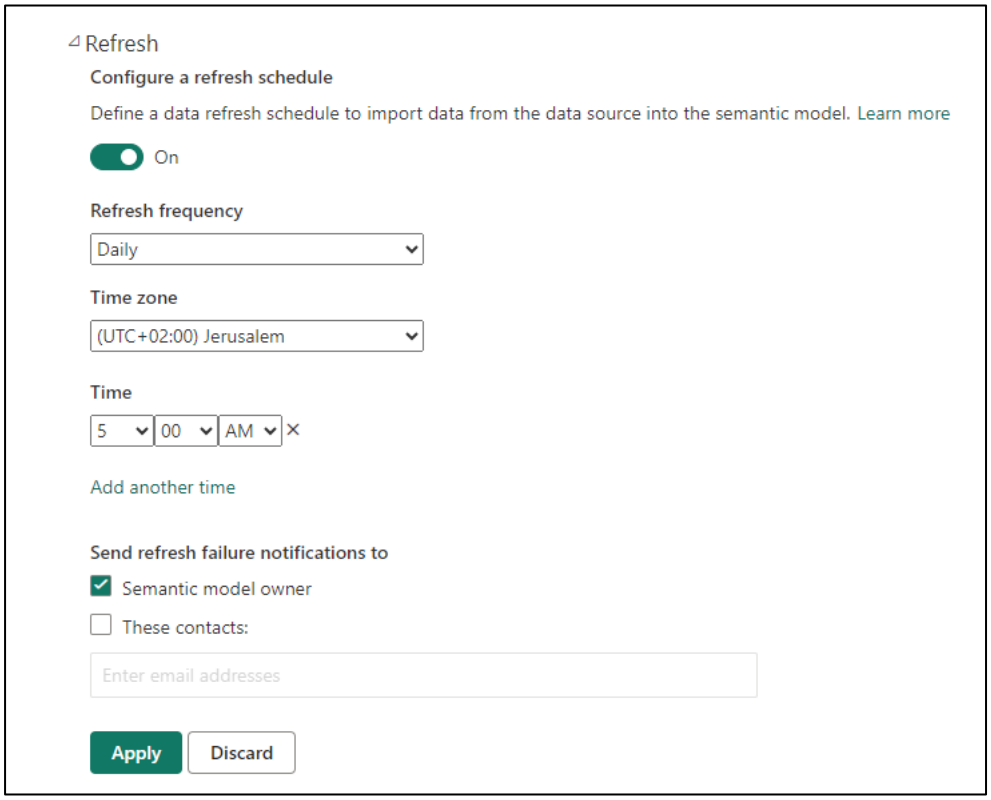


4.3. Updating reports and dashboards

After refreshing the data in the Data Mart we created, we will also want to refresh the data in the reports.

The refresh will also be carried out daily and thus there will always be a match at the beginning of the working day between the Power BI reports and the existing data in the DM source.

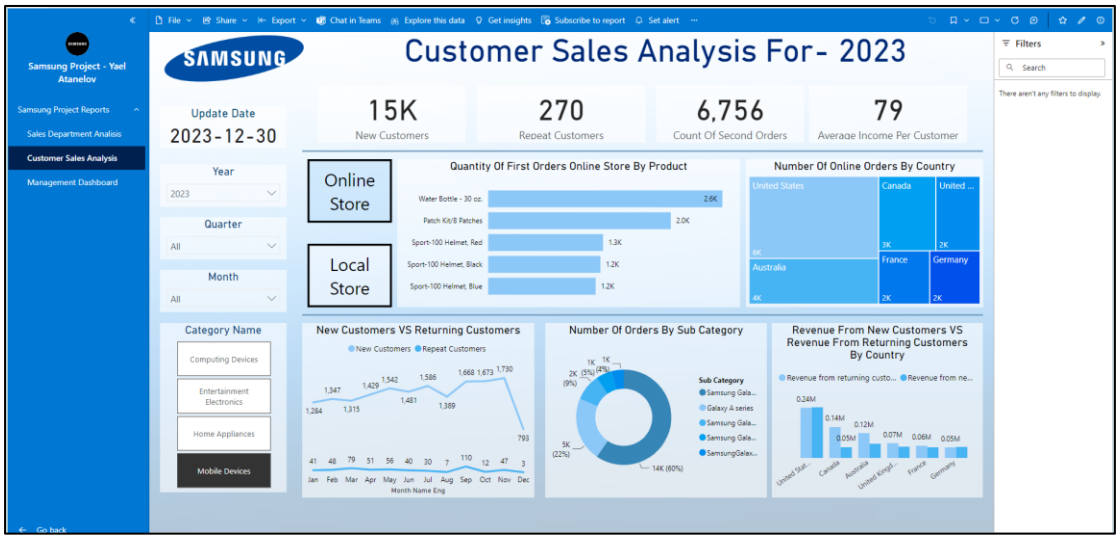
The report will be updated once a day at 5:00 am. The process of refreshing the data in DM is carried out at 00:00 and after taking a safety margin, a loading time for the Data in the reports was set so that there would be no overlap between the loadings.



4.4. Online APP

For the benefit of the users in the organization and for future use of the project's products, the reports and the dashboard will be available to the users under the APP – **Samsung Project - Yael Atanelov**

Link to the app- [Samsung Project by Yael Atanelov]



The application is available on mobile devices either through direct access to the provided link or by utilizing the link within the POWER BI application.