Apple's Sales Data Mart

BI System Specifications Document

Date: 11/02/2024

Version: 5.00

Written by: Adi Aljadeff

Content

1. General

1.1. Project Objective

This project aims to create a full BI solution for Apple's sales department to support Apple's growth in device sales. The project was designed according to Apple's sales department KPIs and is aimed at increasing the company's overall (and the devices sales department in particular) ROI.

Apple Inc. is a multinational technology company headquartered in Cupertino, California. It designs, develops, and sells consumer electronics, computer software, and online services. The company's best-known devices include the iPhone, iPad, Mac, Apple Watch, and Apple TV. Apple's devices are sold worldwide through Apple's physical stores, Apple's online store, and authorized resellers (via physical and online stores). This project will focus only on sales performed by Apple stores (physical and online).

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

1.2. Project Contents

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

ERD model of the AppleSalesDM database: ERD Link

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

- FactSales – Data regarding all sales, including the order ID, products bought, quantities, and prices. The 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.

- DimProductsHistory – Historical data regarding the company's products.

  Source To Target Link

  The tables will be updated daily at 04:00:00 using an automated process configured in the SQL Server Management Studio.

1.2.2. The reports will include data visualizations that will support the project's objective in the following ways:

- Sales Analysis:

  The sales report will include data about sales (revenue, number of orders, and number of units) by date, country, product, store (online vs. physical), and salespersons, which will help the department assess the performance of all the parts needed for sales growth. The reports will help to identify sale trends like seasonality and trending product categories, analyze product orders and revenue, spot top-performing salespersons, and analyze the differences in behavior between the online store and

physical stores. All of these will support data-driven strategic decision-making, which can lead to growth in sales and revenue.

- Customer Analysis:

  The customer analysis report will include data regarding Apple's customers by date, country, store, product, and category. This report is aimed at helping Apple's customer department better understand their customers' behavior, like what (products), where (countries and stores), and when they shop. This is vital to retain current customers and reach new ones.

- Executive Dashboard:

  The dashboard will include key visuals from the two reports. The dashboard will allow a broader perspective on the data and integrate measures from sales and customer analysis.

2. Gantt

[Gantt Link](#)

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
- YTD sales
- Total orders

- Average price per order

- Total units

- Average revenue per customer

- 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)

- Top stores by sales (this graph can change to orders, units)

- Sales by country (this graph can change to orders, units)

- Sales by category and subcategory (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

- Percentage of new customers from total

- YTD customers

- Average revenue per customer

- Average number of orders per customer

- Total customers and new customers by month and day

- Customers by country

- Customers by number of orders per customer

- Top products by customers

- Average revenue per customer compared to previous year by month and day

3.2.2.3. The executive dashboard will consist of:

- Total sales

- Average monthly revenue

- Total orders

- Total customers

- Percentage of new customers from total

- YTD sales

- MTD sales

- Total sales and month-over-month growth by quarter and month

- Revenue by online vs physical stores

- Top selling products by revenue

- Total customers by quarter and month

- Revenue by country

4. Functional Specification

4.1. Creation of final Source to Target and ERD models.

4.1.1. Source to Target

- Source To Target link

A total of 11 tables from the operational database will be used.

4.1.2. ERD model of the AppleSalesDM database

- ERD link

4.2. ETL processes

The ETL process was done in SSIS using 13 packages.

All the packages include 2 reoccurring Execute SQL tasks (Update StartTime Var and Insert Data to DataTransfer) and Row Count transformations, which oversee updating the DataTransfer table. These will be explained later in the DataTransfer table section.

- **FactSales Table:**

  o MRR_Sales package:

  Mirror tables are truncated (using a stored procedure), and data is loaded from the PriorityERP database (INVOICES and INVOICEITMES) to AppleSalesDM mirror tables.
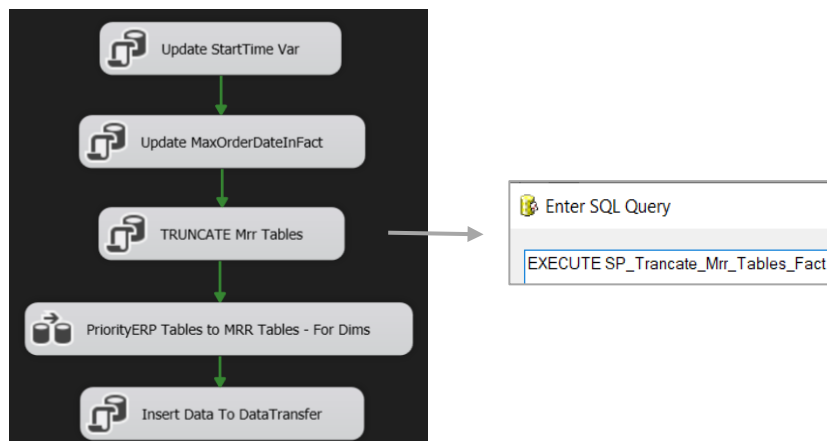
In the data flow, data is incrementally loaded using lookup transformations, meaning only new transactions that cannot be found in the FactSales table are loaded:
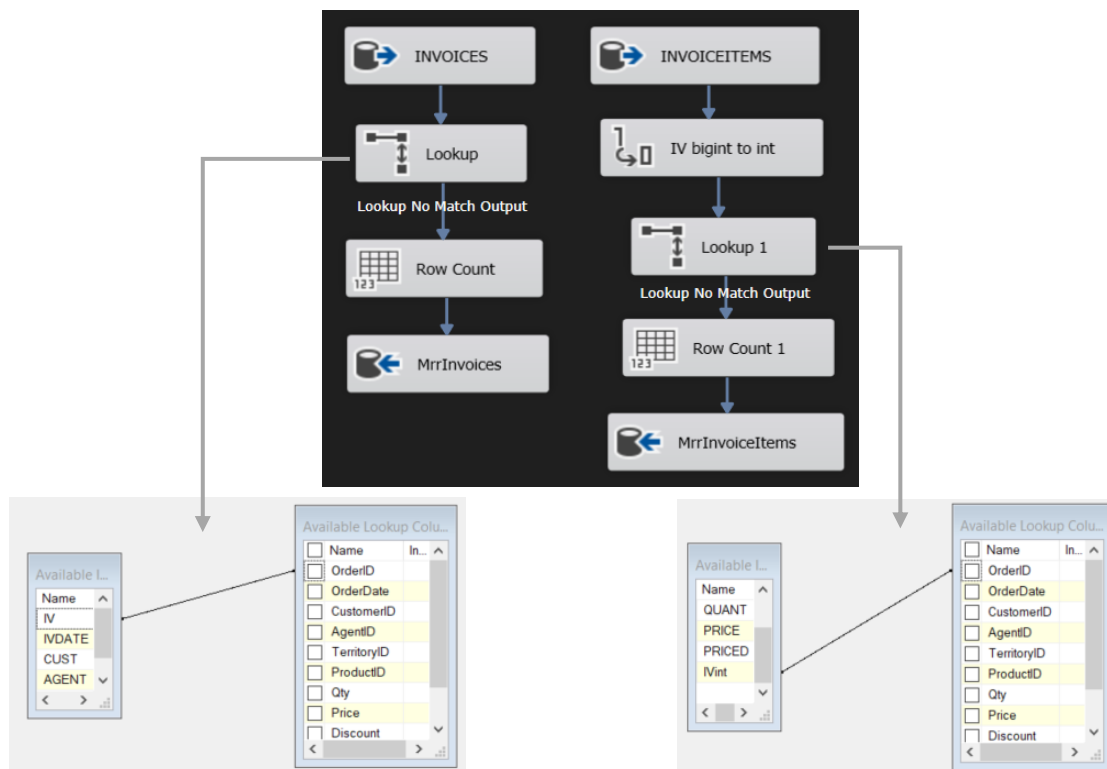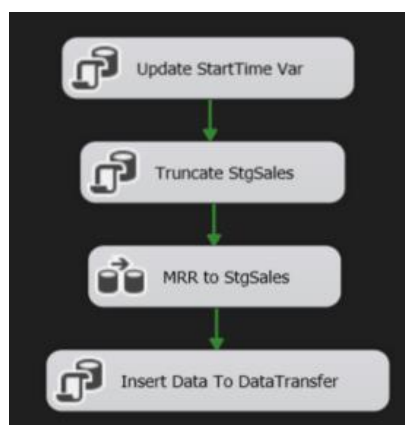

Data Conversion transformation is used to convert the IV column from bigint to int (DT_I4). This conversion is necessary for the lookup transformation:

Input Column	Output Alias	Data Type
IV	IVint	four-byte signed integer [DT_I4]

o STG_Sales package:

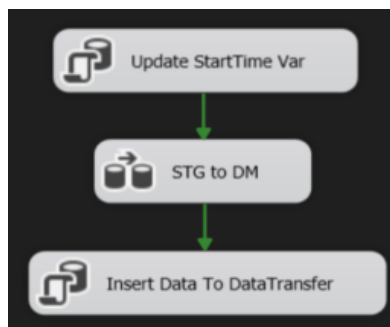The stgSales table is truncated, and the mirror tables are joined and loaded using a data flow task.

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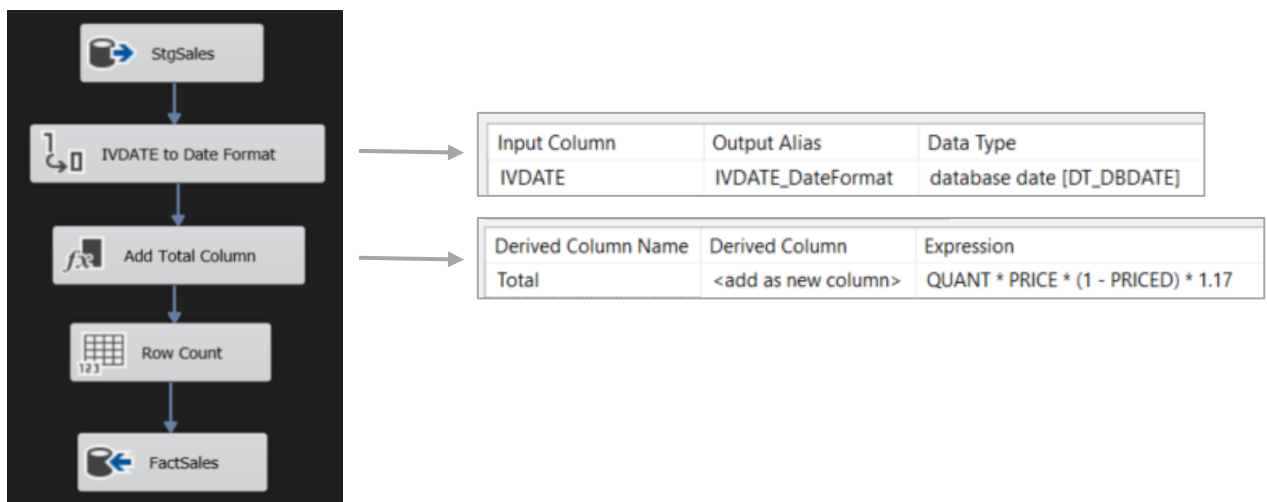

SELECT i.IV, i.IVDATE, CUST, i.AGENT, i.ZONE, ii.PART, QUANT, PRICE, PRICED
FROM MrrInvoices i
JOIN MrrInvoiceItems ii ON i.IV = ii.IV
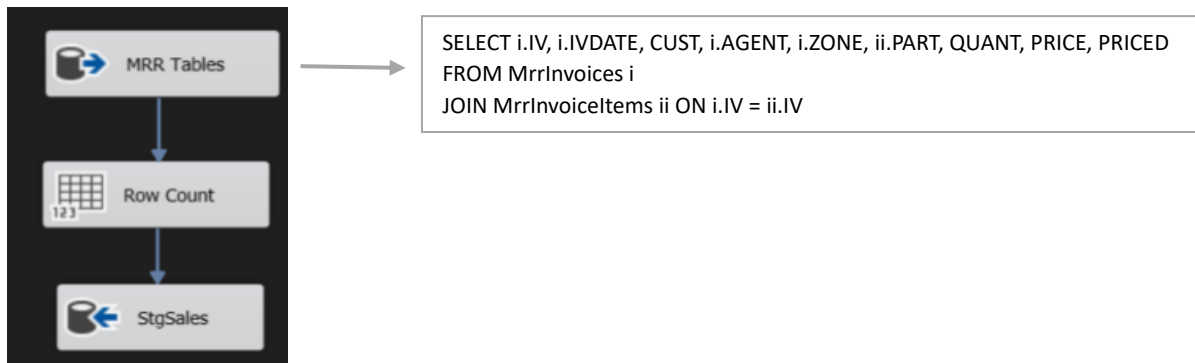
- o DM_Sales package:

Data is loaded from StgSales to FactSales, and a Total column is added.

In the data flow, the IVDATE column is converted from string to date format (DT_DBDATE) in a data conversion transformation. A Total column specifies the total price for the product in a sale and is calculated and added using a derived column transformation.
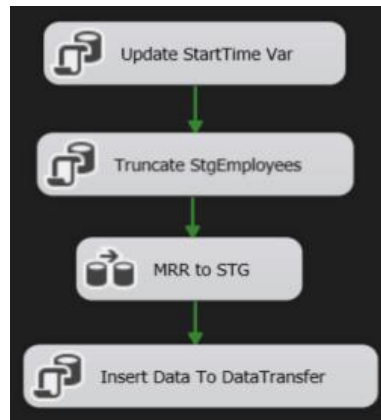

Input Column	Output Alias	Data Type
IVDATE	IVDATE_DateFormat	database date [DT_DBDATE]

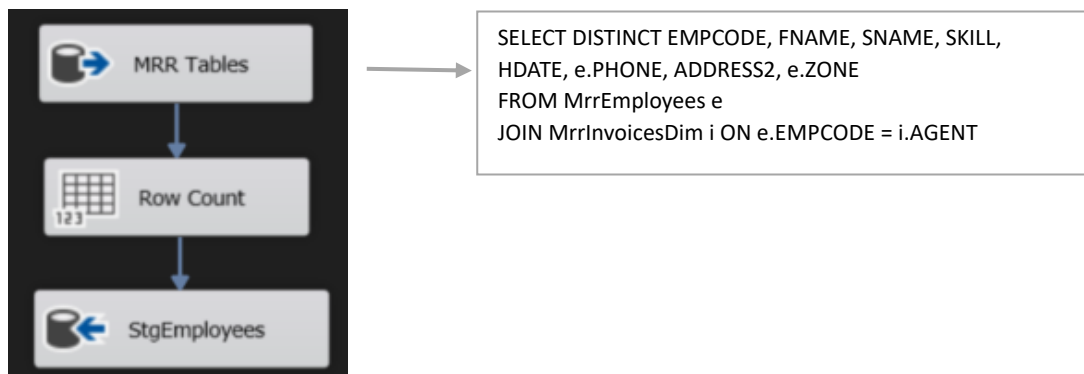| Derived Column Name	Derived Column	Expression
Total	<add as new column>	QUANT * PRICE * (1 - PRICED) * 1.17

- **DimEmployees Table**:
- o STG_Employees package:

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


In the data flow, the mirror tables (MrrEmployees and MrrInvoicesDim) are joined, and the data is loaded to the StgEmployees table.


```
SELECT DISTINCT EMPCODE, FNAME, SNAME, SKILL,
HDATE, e.PHONE, ADDRESS2, e.ZONE
FROM MrrEmployees e
JOIN MrrInvoicesDim i ON e.EMPCODE = i.AGENT
```

This Join's objective is to exclude irrelevant employees from the sales data mart, i.e., include only employees who made a sale (thus are included in the INVOICES table).

- DM_Employees package:

    Data is incrementally loaded and updated in the DimEmployees.
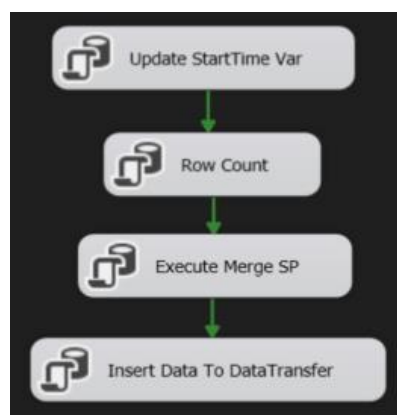
A merge stored procedure is executed in the Execute SQL Task. The merge statement works according to the following rationale:

1. For **new** employees (EmployeeID exists in StgEmployees but not in DimEmployees): Insert new records to DimEmployees.
2. For **updated** employees (EmployeeID exists in StgEmployees and in DimEmployees, but one or more of the other columns does not match): Update the record in DimEmployees.
3. For **deleted** employees (EmployeeID exists in DimEmployees but not in StgEmployees): Update the IsActive column in DimEmployees to 'N'.
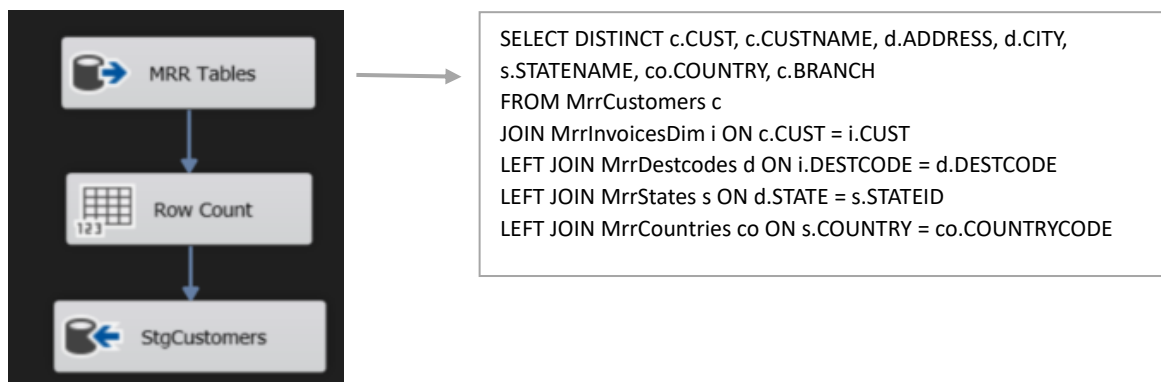
- **DimCustomers Table**:

  o <u>STG_Customers package</u>:

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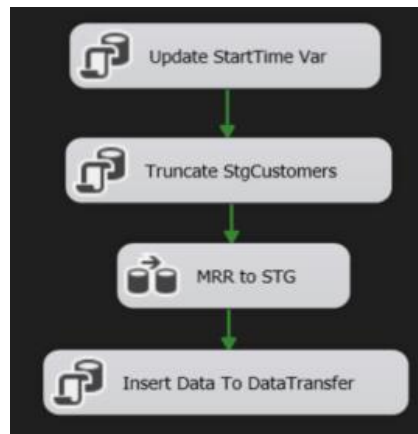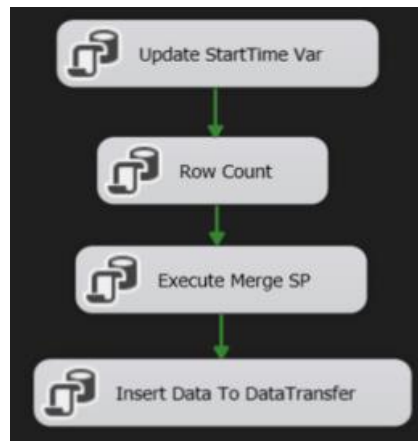

  In the data flow, the 5 mirror tables (MrrCustomers, MrrInvoicesDim, MrrDestcodes, MrrStates, MrrCountries) are joined, and the data is loaded to the StgCustomers table.


```
SELECT DISTINCT c.CUST, c.CUSTNAME, d.ADDRESS, d.CITY,
s.STATENAME, co.COUNTRY, c.BRANCH
FROM MrrCustomers c
JOIN MrrInvoicesDim i ON c.CUST = i.CUST
LEFT JOIN MrrDestcodes d ON i.DESTCODE = d.DESTCODE
LEFT JOIN MrrStates s ON d.STATE = s.STATEID
LEFT JOIN MrrCountries co ON s.COUNTRY = co.COUNTRYCODE
```

  o <u>DM_Customers package</u>:

Data is incrementally loaded and updated in DimCustomers.
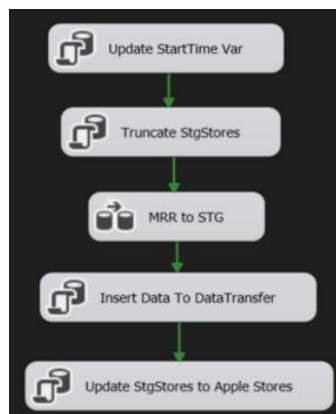

A merge stored procedure is executed in the Execute SQL Task. The merge statement works according to the following rationale:

1. For **new** customers (CustomerID exists in stgCustomers but not in DimCustomers): Insert new records to DimCustomers.
2. For **updated** customers (CustomerID exists in stgCustomers and DimCustomers, but one or more of the other columns does not match): Update the record in DimCustomers.
3. For **deleted** customers (CustomerID exists in DimCustomers but not in StgCustomers): Updates the IsActive column in DimCustomers to 'N'.
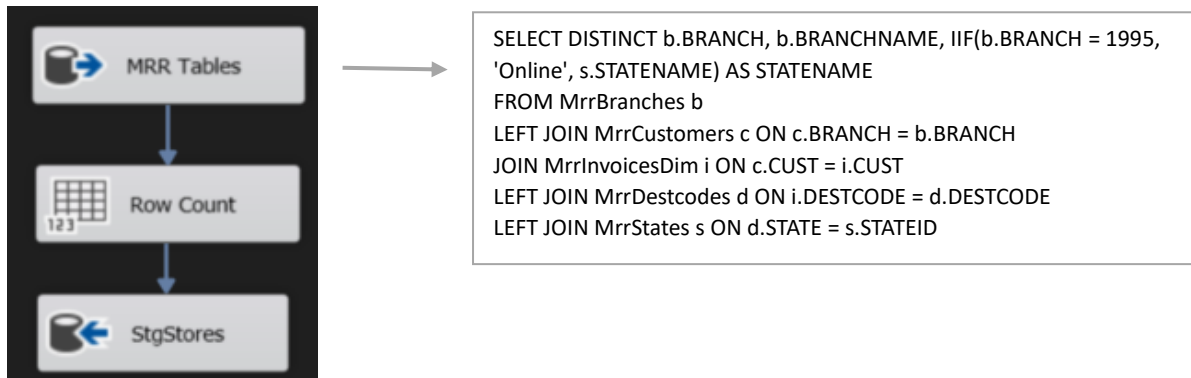
- **DimStores Table**:

  o STG_Stores package:

  The stgStores table is truncated, and the mirror tables are joined and loaded using a data flow task. Store names are updated using a stored procedure executed in an Execute SQL
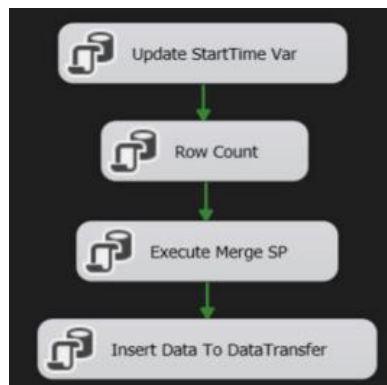

  task.

In the data flow, the 3 mirror tables (MrrBranches, MrrCustomers, MrrInvoicesDim) are joined, and the data is loaded to the StgStores table.


```
SELECT DISTINCT b.BRANCH, b.BRANCHNAME, IIF(b.BRANCH = 1995,
'Online', s.STATENAME) AS STATENAME
FROM MrrBranches b
LEFT JOIN MrrCustomers c ON c.BRANCH = b.BRANCH
JOIN MrrInvoicesDim i ON c.CUST = i.CUST
LEFT JOIN MrrDestcodes d ON i.DESTCODE = d.DESTCODE
LEFT JOIN MrrStates s ON d.STATE = s.STATEID
```

- o DM_Stores package:

    Data is incrementally loaded and updated in DimCustomers.

    A merge stored procedure is executed in the Execute SQL Task. The merge statement works according to the following rationale:

    1. For **new** stores (storeID exists in StgStores but not in DimStores): Insert new records to DimStores.
    2. For **updated** stores (storeID exists in StgStores and DimStores but one or more of the other columns does not match): Update the record in DimStores.
    3. For **deleted** stores (storeID exists in DimStores but not in StgStore): Update the IsActive column in DimStores to 'N'.
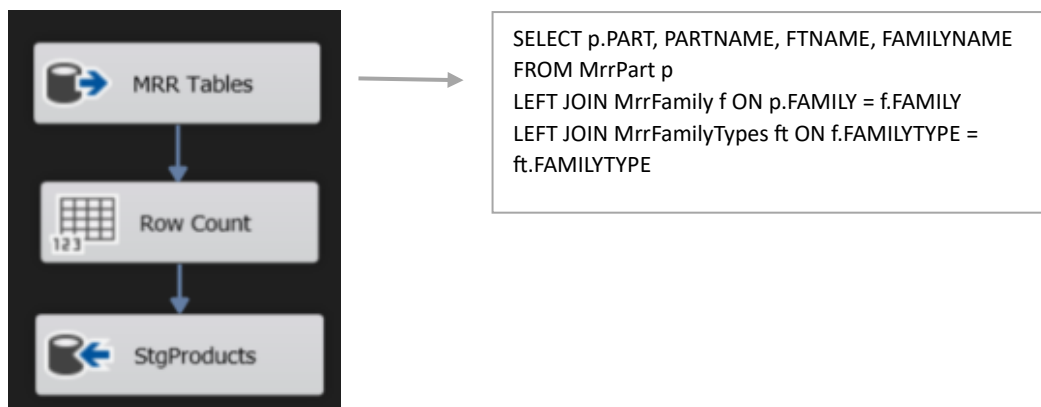
- **DimProducts Table**:
- o STG_Products package:

The stgProducts table is truncated, and the mirror tables are joined and loaded using a data flow task. Product names, categories, and subcategories are updated using a stored procedure executed in an Execute SQL task.
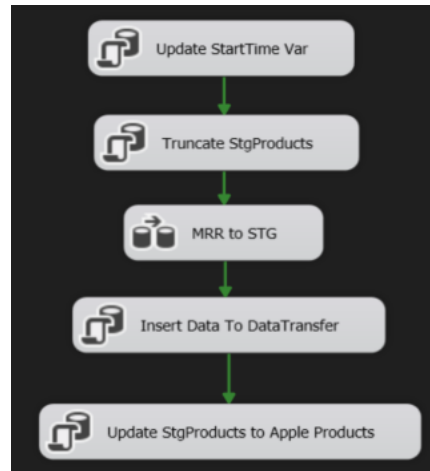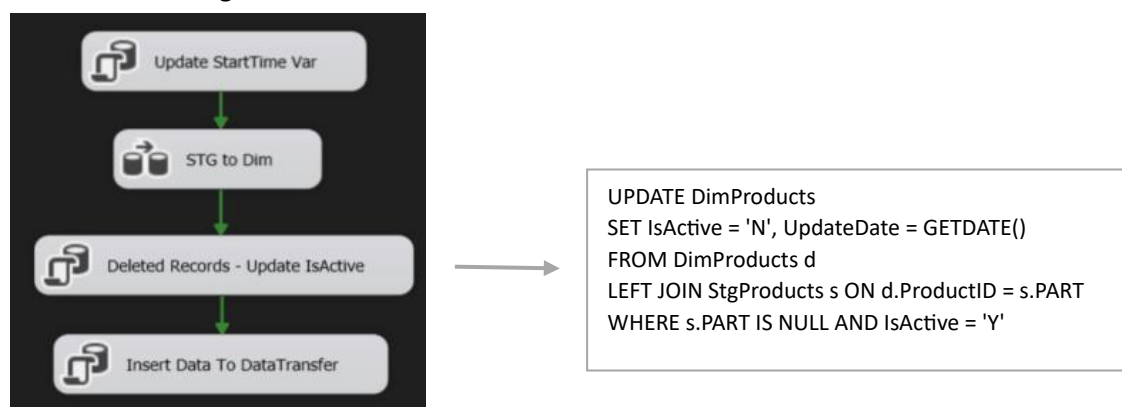

In the data flow, the 3 mirror tables (MrrPart, MrrFamily, MrrFamilyTypes) are joined, and the data is loaded to the StgProducts table.


```
SELECT p.PART, PARTNAME, FTNAME, FAMILYNAME
FROM MrrPart p
LEFT JOIN MrrFamily f ON p.FAMILY = f.FAMILY
LEFT JOIN MrrFamilyTypes ft ON f.FAMILYTYPE =
ft.FAMILYTYPE
```

o DM_Products package:
Data is incrementally loaded and updated in DimProducts. Deleted records are updated in DimProducts using an Execute SQL task.


```
UPDATE DimProducts
SET IsActive = 'N', UpdateDate = GETDATE()
FROM DimProducts d
LEFT JOIN StgProducts s ON d.ProductID = s.PART
WHERE s.PART IS NULL AND IsActive = 'Y'
```

Deleted records are located using Left Join between DimProducts and StgProducts using ProductID as the key while filtering records to products where ProductID in StgProducts is

NULL (i.e., products which cannot be found in StgProducts) and products where IsActive in DimProducts is 'Y' (i.e., they were not updated as deleted yet).

In the Data Flow, Incremental load to the DimProducts table is done using the Slowly Changing Dimension transformation (change type: Changing Attribute). 3 column data types are converted to Unicode (DT_WSTR) using data conversion transformation to match destination data types.
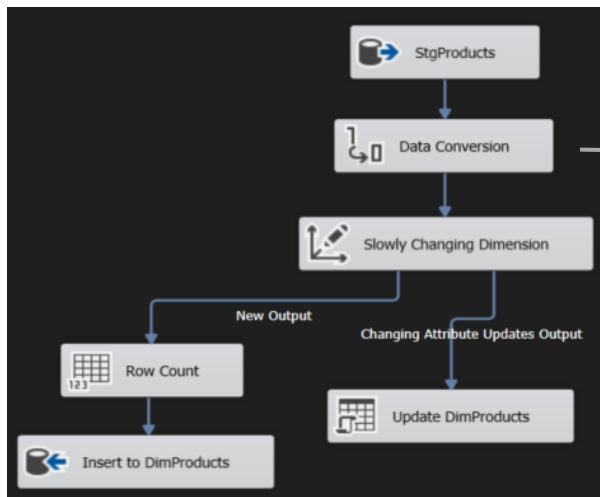

Input Column	Output Alias	Data Type	Length
PARTNAME	PARTNAME-unicode	Unicode string [DT_WSTR]	40
FAMILYNAME	FAMILYNAME-unicode	Unicode string [DT_WSTR]	35
FTNAME	FTNAME-unicode	Unicode string [DT_WSTR]	35

- **DimProductsHistory Table**:
 - o DM_ProductsHistory package:

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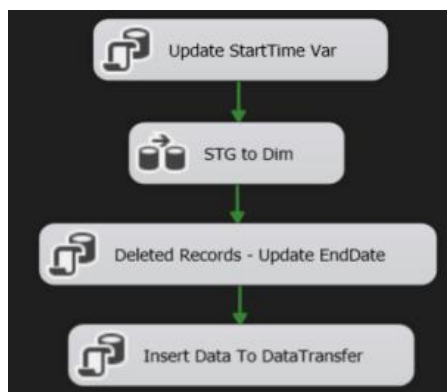

```
UPDATE DimProductsHistory
SET EndDate = GETDATE()
FROM DimProductsHistory d
LEFT JOIN StgProducts s ON d.ProductID = s.PART
WHERE s.PART IS NULL AND d.EndDate IS NULL
```

    Deleted records are located in the same manner as in the DM_Products package but using the EndDate column instead of IsActive.

    In the Data Flow, Incremental load to the DimProductsHistory table is done using the Slowly Changing Dimension transformation (change type: Historical Attribute). 3 column data types are converted to Unicode (DT_WSTR) using data conversion transformation to match destination data types.
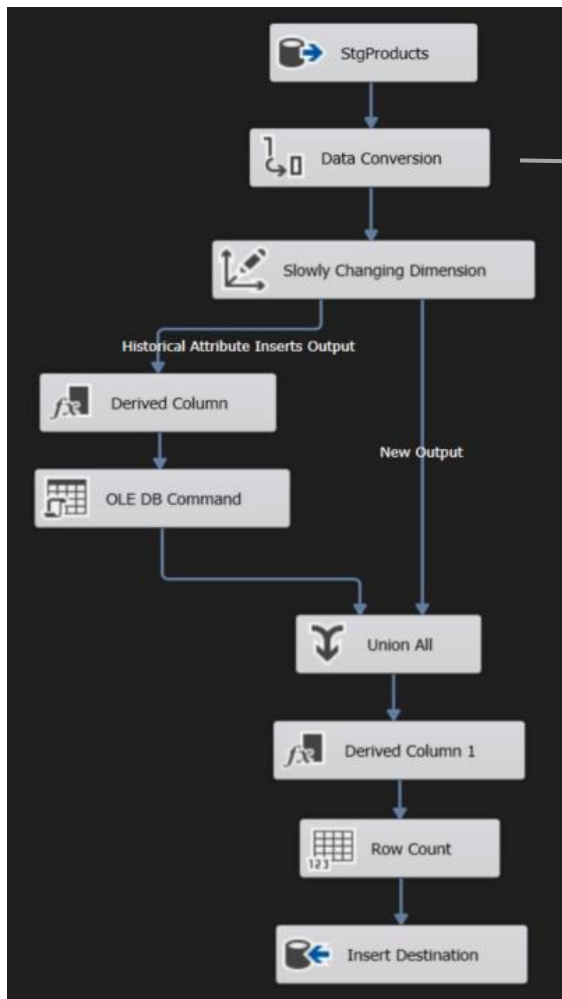
Input Column	Output Alias	Data Type	Length
PARTNAME	PARTNAME-unicode	Unicode string [DT_WSTR]	40
FAMILYNAME	FAMILYNAME-unicode	Unicode string [DT_WSTR]	35
FTNAME	FTNAME-unicode	Unicode string [DT_WSTR]	35

- **All Dim Tables – Mirror step**

  o <u>MRR_Dim package</u>:

  This package is responsible for loading data from PriorityERP tables to all mirror tables relevant to the dim tables (10 tables in total). All mirror tables (except MrrInvoicesDim) are truncated using a stored procedure.
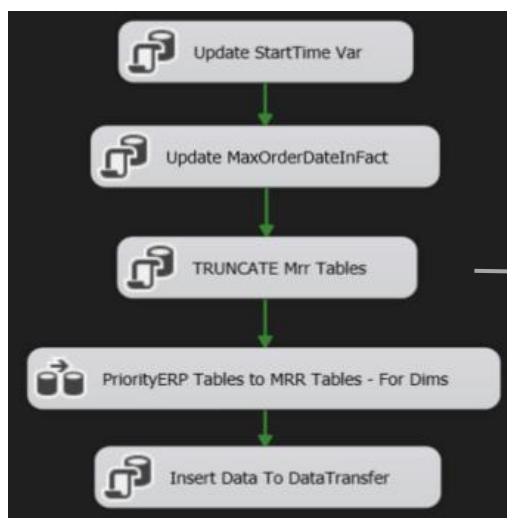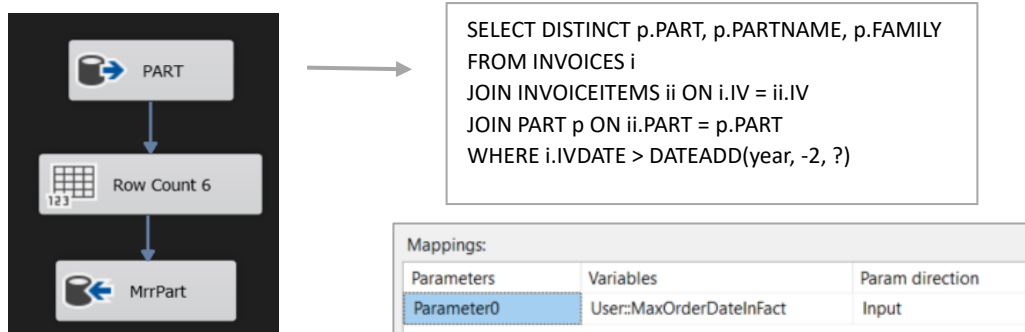
In the data flow:

Loading to MrrInvoicesDim is incremental and is performed using Lookup transformation (comparing records to MrrInvoicesDim). The MrrInvoicesDim is not truncated (compared to other mirror tables):
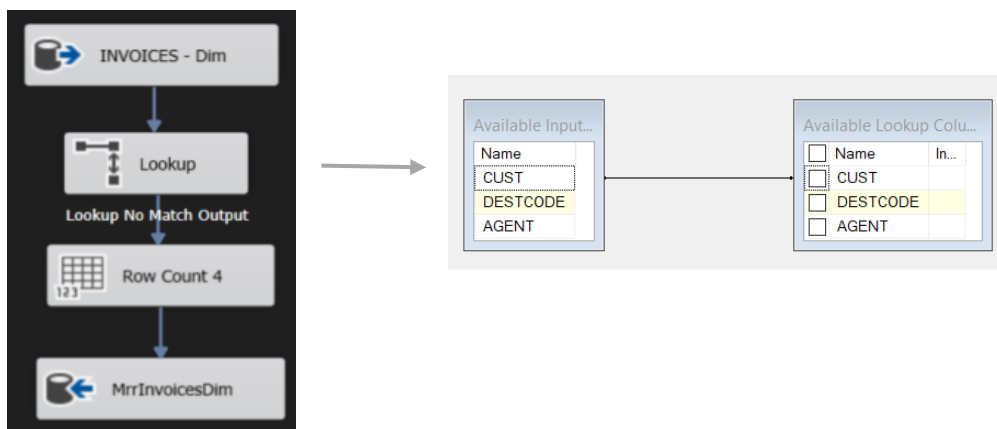
To filter out irrelevant products while loading to MrrPart, only products sold in the last 2 years are loaded to the table. This is done using a variable containing the max order date in the FactSales table:


```
SELECT DISTINCT p.PART, p.PARTNAME, p.FAMILY
FROM INVOICES i
JOIN INVOICEITEMS ii ON i.IV = ii.IV
JOIN PART p ON ii.PART = p.PART
WHERE i.IVDATE > DATEADD(year, -2, ?)
```

Mappings:

| Parameters	Variables	Param direction
Parameter0	User::MaxOrderDateInFact	Input

- **DataTransfer Table**

To monitor the ETL process, a dataTransfer table was created documenting each data insert: which table was updated and in which package, how many rows were inserted, and start and end times. The tasks and transformation in charge of the updates are included in all of the packages.

Example from STG_Stores package:

In the control flow user variable, StartTime is updated in the first task, and an insert statement is executed in the last task, inserting the values of the user variables: PackageName, TableName, StartTime, RowCount (which is updated in the data flow), with GETDATE() as EndTime.
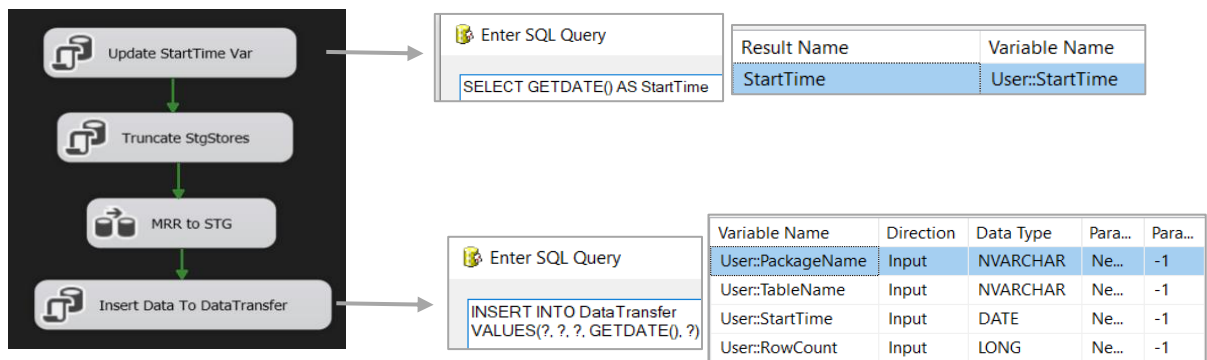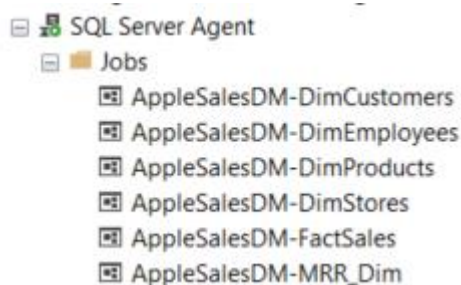

In the data flow, the user variable RowCount is updated using a Row Count transformation.
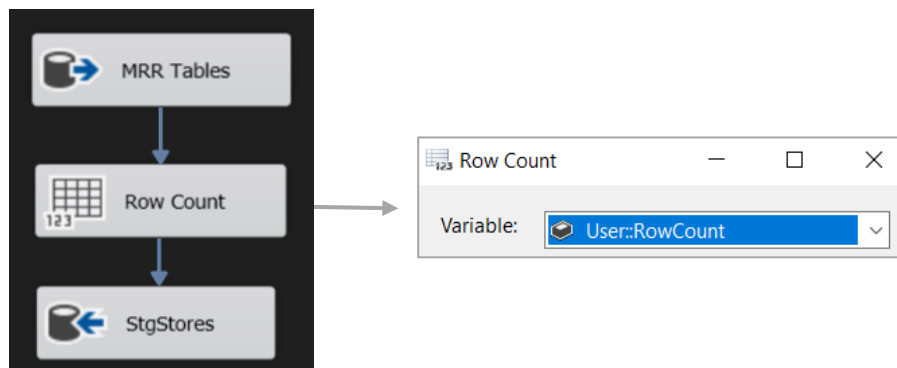
- **Automatic Processing:**

The data is automatically refreshed daily at 4:00:00 using SQL Agent jobs. The first job executed is the AppleSalesDM-MRR_DIM, this job executes the next job, and so on:

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 DimDate table was added, together with 3 more tables:

- Orders Sums – an aggregated table calculating the sum for each order. This table was used in a slicer slicing orders by sum groups (e.g. $0-$500, $500-$1,000 etc.). This table was related to FactSales using OrderID.

- Yearly Goals – a table specifying the sales goals for each year (relying on the previous year's sales). This table was not connected to the model.

- Monthly Goals - a table specifying the sales goals for each month (relying on sales of the same month in the previous year).
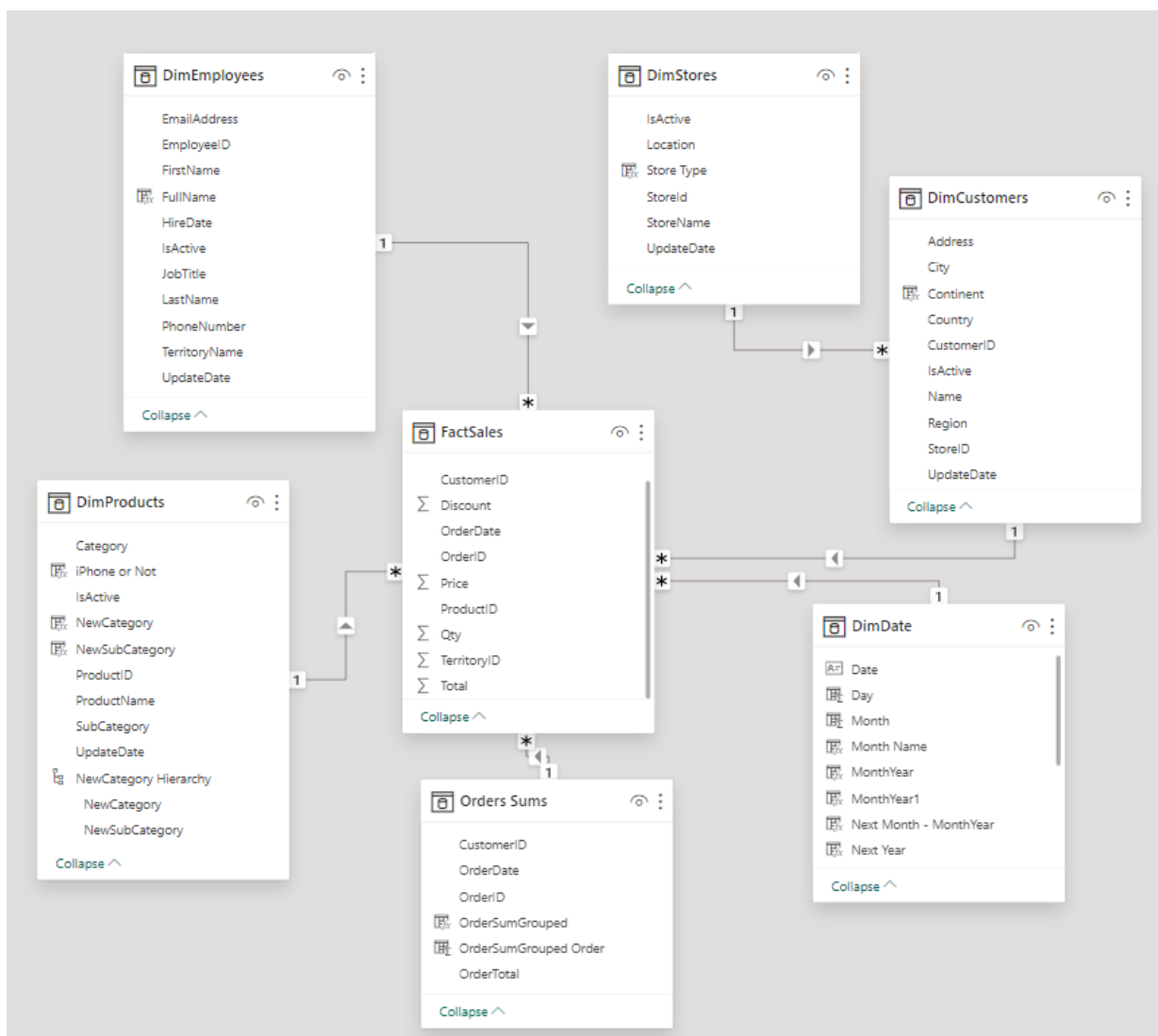
4.3.2. To create the visuals, the following measures were created in DAX:

<u>Totals:</u>

- Total Customers = IF(ISBLANK(DISTINCTCOUNT(FactSales[CustomerID])), 0, DISTINCTCOUNT(FactSales[CustomerID]))

- Total Orders = IF(ISBLANK(DISTINCTCOUNT(FactSales[OrderID])), 0, DISTINCTCOUNT(FactSales[OrderID]))

- Total Revenue = IF(ISBLANK(SUM(FactSales[Total])), 0, SUM(FactSales[Total]))

- Total Units = IF(ISBLANK(SUM(FactSales[Qty])), 0, SUM(FactSales[Qty]))

Previous Year/Month:

- Previous Year Orders = CALCULATE([Total Orders], SAMEPERIODLASTYEAR(DimDate[Date]))

- Previous Year Sales = CALCULATE([Total Revenue], SAMEPERIODLASTYEAR(DimDate[Date]))

- Previous Year Units = CALCULATE([Total Units], SAMEPERIODLASTYEAR(DimDate[Date]))

- Previous Month Sales = IF(ISBLANK(CALCULATE([Total Revenue], PREVIOUSMONTH(DimDate[Date]))), BLANK(), CALCULATE([Total Revenue], PREVIOUSMONTH(DimDate[Date])))

Change From Last Year/Month:

- YoY Growth = IFERROR([Total Revenue]/[Previous Year Sales] - 1, "NULL")

- YoY Growth Orders = IFERROR([Total Orders]/[Previous Year Orders] - 1, "NULL")

- YoY Growth Units = IFERROR([Total Units]/[Previous Year Units] - 1, "NULL")

- MoM Growth = IFERROR([Total Revenue]/[Previous Month Sales] - 1, BLANK())

YTD and MTD:

- YTD Customers = TOTALYTD([Total Customers], DimDate[Date])

- YTD Sales = TOTALYTD([Total Revenue], DimDate[Date])

- MTD Customers = TOTALMTD([Total Customers], DimDate[Date])

- MTD Sales = TOTALMTD([Total Revenue], DimDate[Date])

Averages:

- Average Monthly Revenue = IF(ISERROR(AVERAGEX(VALUES(DimDate[MonthYear]), [Total Revenue])), 0, AVERAGEX(VALUES(DimDate[MonthYear]), [Total Revenue]))

- Average Orders Per Customer = IF(ISERROR([Total Orders]/[Total Customers]), 0, [Total Orders]/[Total Customers])

- Average Sales Amount Per Customer = IF(ISERROR([Total Revenue] / DISTINCTCOUNT(DimCustomers[CustomerID])), 0, [Total Revenue] / DISTINCTCOUNT(DimCustomers[CustomerID]))

- Average Sales Amount Per Customer Previous Year = IF(ISERROR([Previous Year Sales] / DISTINCTCOUNT(DimCustomers[CustomerID])), 0, [Previous Year Sales] / DISTINCTCOUNT(DimCustomers[CustomerID]))

- Average Sales Amount Per Order = IF(ISERROR([Total Revenue] / DISTINCTCOUNT(FactSales[OrderID])), 0, [Total Revenue] / DISTINCTCOUNT(FactSales[OrderID]))

- Average Units Per Customer = IF(ISERROR([Total Units]/[Total Customers]), 0, [Total Units]/[Total Customers])

- Average Units Per Order = IF(ISERROR([Total Units] / DISTINCTCOUNT(FactSales[OrderID])), 0, [Total Units] / DISTINCTCOUNT(FactSales[OrderID]))

New Customers:

- New Customers =

  VAR currentCustomers = VALUES(FactSales[CustomerID])

  VAR currentDate = MIN(DimDate[Date])
  VAR pastCustomers =
  CALCULATETABLE(VALUES(FactSales[CustomerID]),ALL(DimDate[Date]),DimDate[Date]<currentDate)

  VAR newCustomers = EXCEPT(currentCustomers,pastCustomers)

  RETURN

  IF(ISBLANK(COUNTROWS(newCustomers)), 0, COUNTROWS(newCustomers))

- New Customers % = IF(ISERROR([New Customers]/[Total Customers]), 0, [New Customers]/[Total Customers])

  Measures for visuals including monthly and yearly goals:

- Max Month = FORMAT(MAXX(DimDate, DimDate[Date]), "mmm-yyyy")

- Max Year = Year(MAXX(DimDate, DimDate[Date]))

- Monthly Goal = CALCULATE(SUM('Monthly Goals'[Goal]), FILTER('Monthly Goals', 'Monthly Goals'[Next Month - MonthYear] = [Max Month]))

- Monthly Goal Max For Gauge = [Monthly Goal] * 1.2

- Yearly Goal = CALCULATE(SUM('Yearly Goals'[Goal]), FILTER('Yearly Goals', 'Yearly Goals'[Next Year] = [Max Year]))

- Yearly Goal Max For Gauge = [Yearly Goal] * 1.2

- Count of Customers with that Many Orders Grouped =
  VAR _CustomerOrders =
      SUMMARIZE(
      FactSales,
      FactSales[CustomerID],
      "How Many Orders",
      DISTINCTCOUNT(FactSales[OrderID])
)
  RETURN
      COUNTROWS(
      FILTER(
      _CustomerOrders,
      [How Many Orders] >= SELECTEDVALUE(Segment[Bottom]) && [How Many Orders] <= SELECTEDVALUE(Segment[Top])
   ))

4.3.3. Reports:

The project includes 3 reports: Executive Dashboard, Sales Analysis, and Customer Analysis. All three reports include only data from the last year (i.e., 365 earlier than the max order date)

4.3.3.1. Executive Dashboard:

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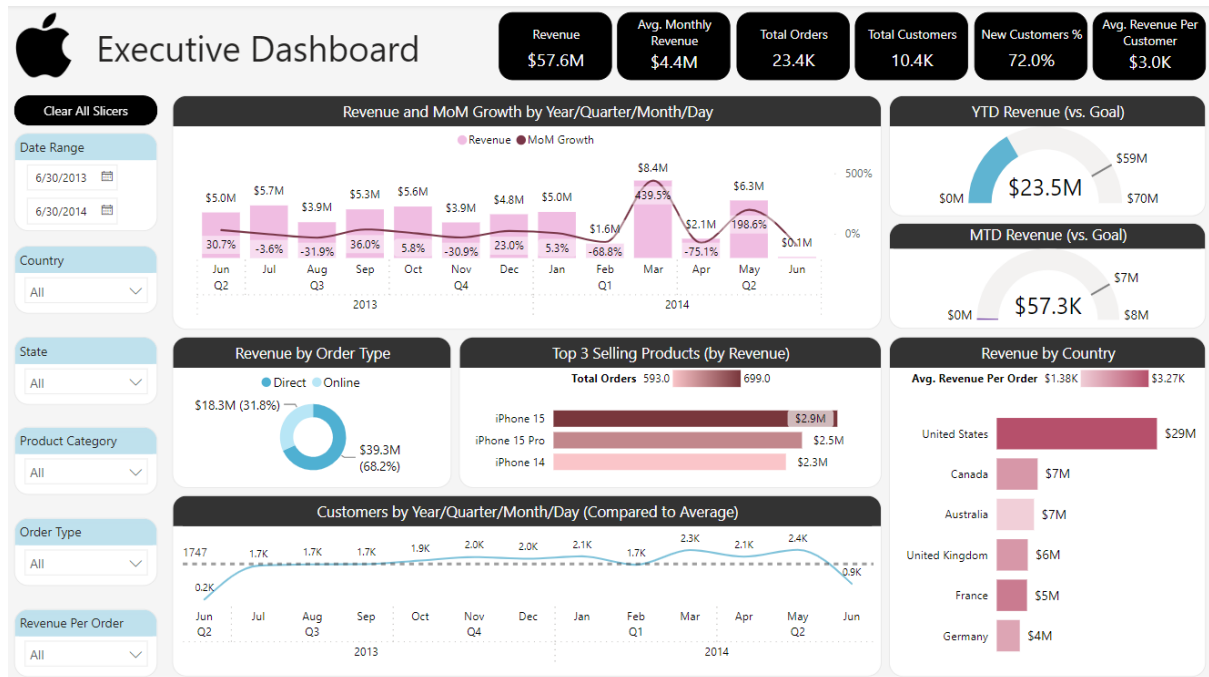
KPI Cards:

- Revenue
- Average Monthly Revenue
- Total Orders
- Total Customers
- New Customers %
- Average Revenue Per Customer

Graphs:

- Revenue and MoM Growth by Year/Quarter/Month/Day (Drill Mode)
- YTD Revenue compared to yearly goal – goal and max value on the gauge, change according to the time range.
- Month Revenue compared to monthly goal - goal and max value on the gauge, change according to the time range.
- Revenue by Order Type
- Top 3 Selling Products by Revenue
- Customers by Year/Quarter/Month/Day (Drill Mode) Compared to Average
- Revenue by Country – Color by Average Revenue per Order

Slicers:

- Date Range
- Country
- State
- Product Category
- Order Type

- Revenue Per Order (grouped)

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 top right, the user can control the data shown in the graphs and change it to orders and units data.
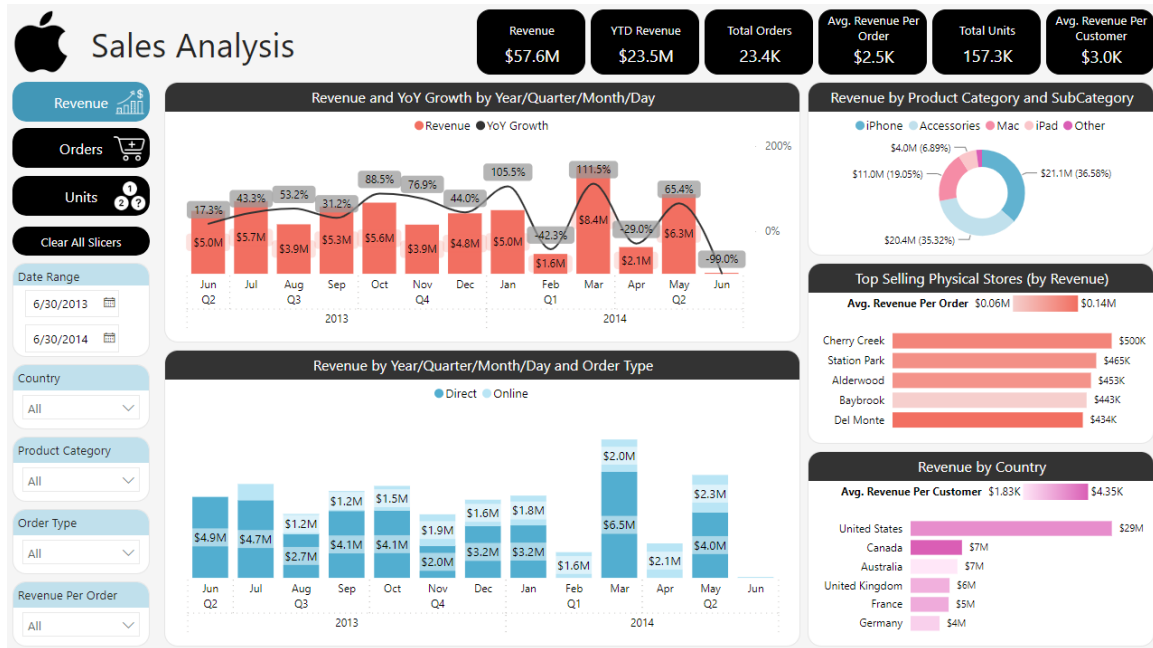
KPI Cards (same for all 3 states):

- Revenue
- YTD Revenue
- Total Orders
- Average Revenue per Order
- Total Units
- Average Revenue Per Customer

Slicers (same for all 3 states):

- Date Range
- Country
- Product Category
- Order Type
- Revenue Per Order (grouped)

Graphs (revenue state):

- Revenue and YoY Growth by Year/Quarter/Month/Day (Drill Mode)
- Revenue by Year/Quarter/Month/Day (Drill Mode) and by Order Type
- Revenue by Product Category/SubCategory (Drill Mode)
- Top 5 Selling Stores (by Revenue)
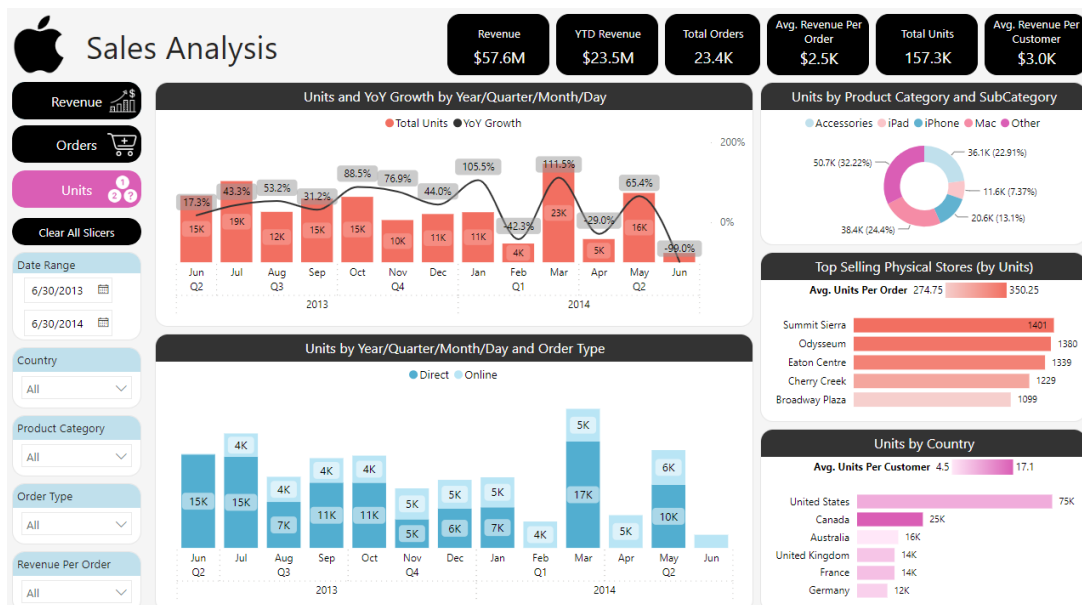- Revenue by Country – Color gradient by Average Revenue per Customer

Orders State:
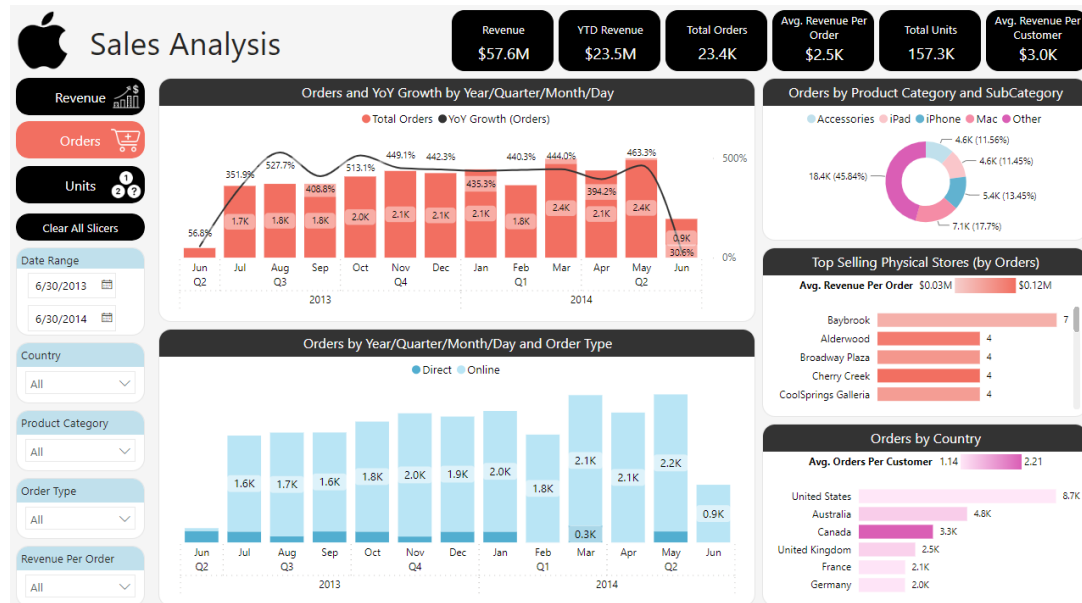
Graphs (orders state):

- Orders and YoY Growth by Year/Quarter/Month/Day (Drill Mode)
- Orders by Year/Quarter/Month/Day (Drill Mode) and by Order Type
- Orders by Product Category/SubCategory (Drill Mode)
- Top 5 Selling Stores (by Orders)
- Revenue by Country – Color gradient by Average Orders per Customer

Units State:

Graphs (units state):

- Units and YoY Growth by Year/Quarter/Month/Day (Drill Mode)
- Units by Month/Day (Drill Mode) and by Order Type
- Units by Product Category/SubCategory (Drill Mode)

- Top 5 Selling Stores (by Units)
- Revenue by Country – Color gradient by Average Units per Customer

4.3.3.3. Customer Sales Analysis:

This report was to understand better Apple's customer sales behavior to achieve the department's goals.
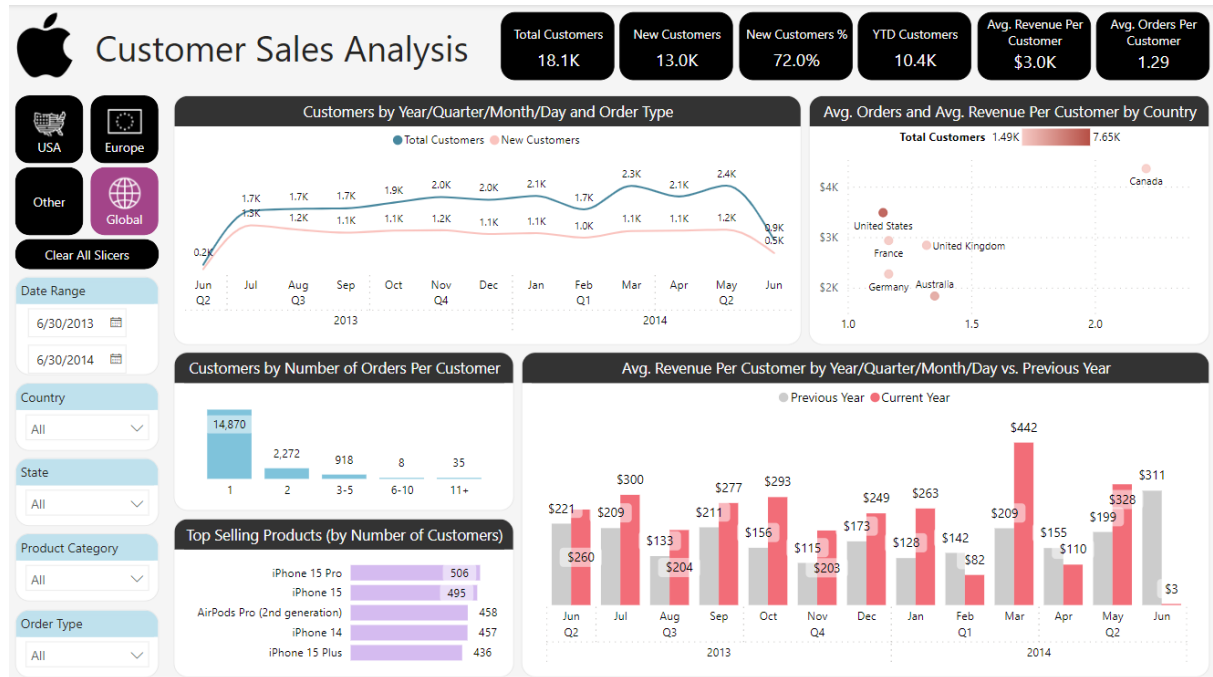
KPI Cards:

- Total Customers
- New Customers
- New Customers %
- YTD Customers
- Average Revenue per Customer
- Average Orders per Customer

Graphs:

- Customers by Year/Quarter/Month/Day (Drill Mode)
- Average Orders per Customer and Average Revenue Per Customer by Country – Color gradient by Total Customers
- Customers by Number of Orders per Customer – This bar chart, specified for the chosen time range (can be changed using the Date Range slicer), how many customers ordered once, twice, 3-5 times, 6-10 times, and more than 10.
- Top 5 Selling Products by number of customers.
- Average Revenue per Customer by Month/Day (Drill Mode) vs. Previous Year

Slicers:

- Date Range
- Country
- State

- Product Category
- Order Type

Using the 4 buttons on the top left, the user can slice the data according to significant customer populations: USA, Europe (France, Germany, and the United Kingdom), Other (Australia and Canada), and global – presenting data from all countries.
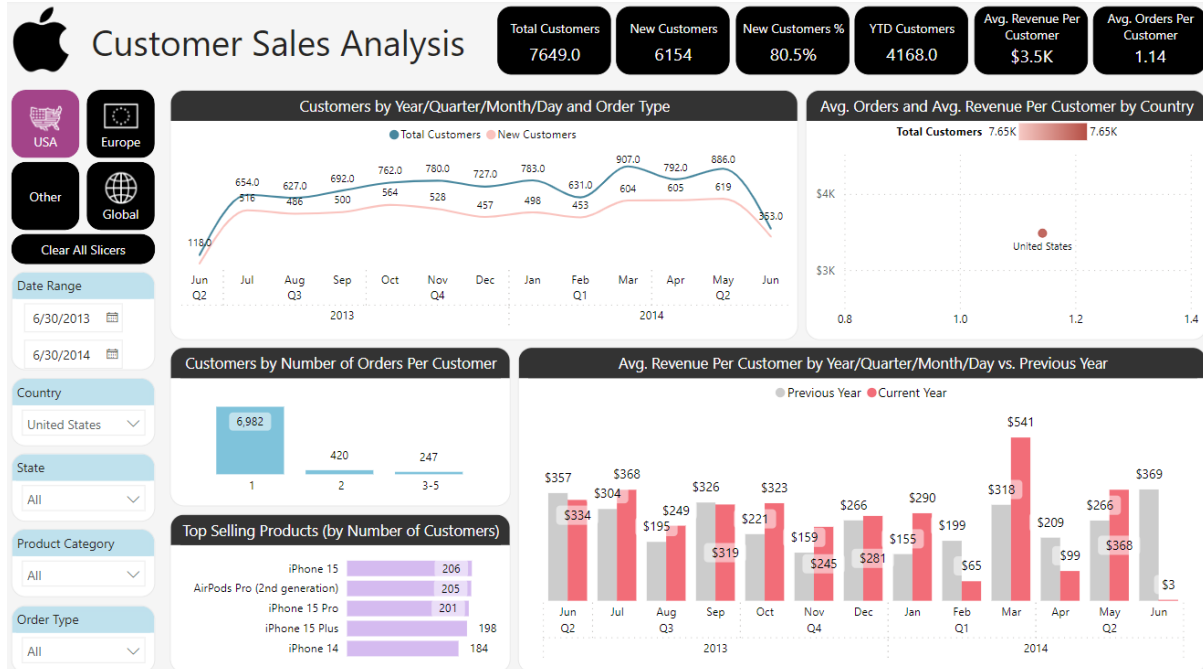
4.3.4. After creating the reports in Power BI Desktop, they were published to Power BI Service, and an app was created - <u>Final Project – Adi Aljadeff - Apple</u>.
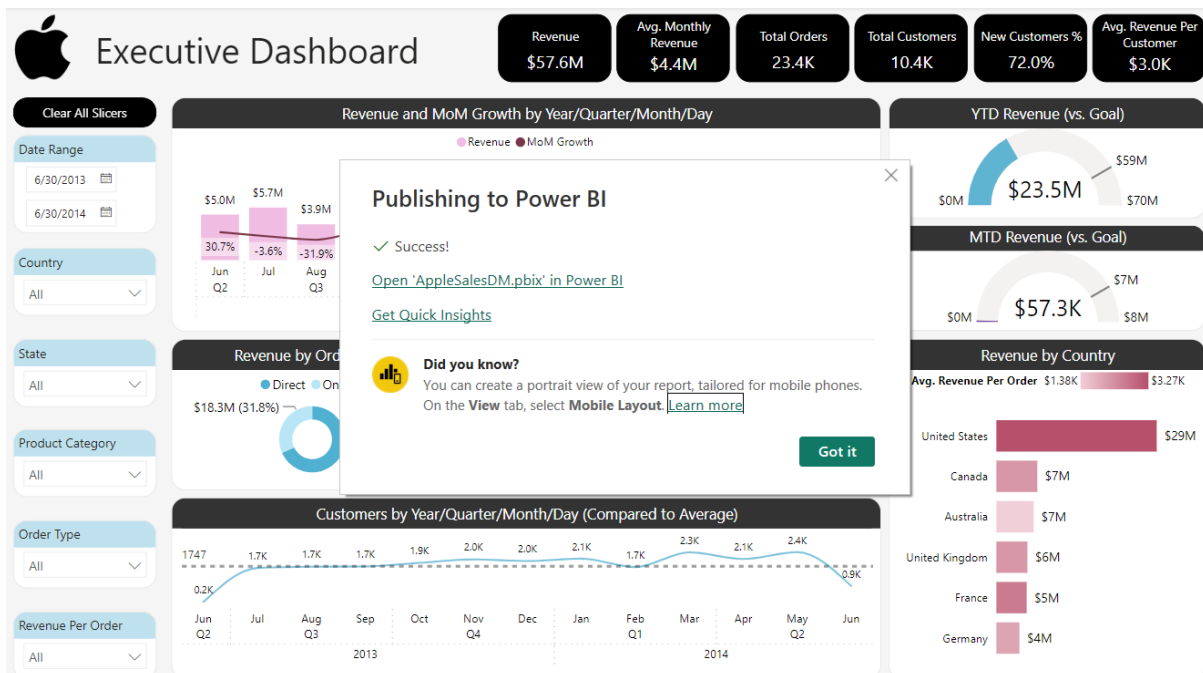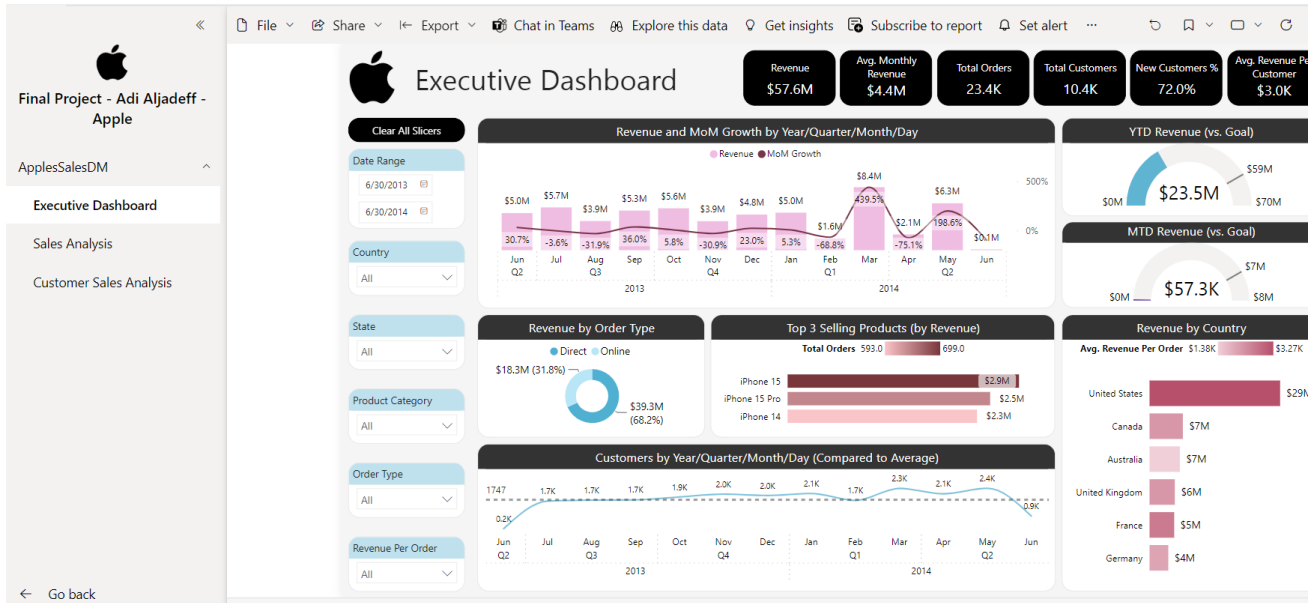
4.3.5. The data is refreshed daily at 5:00:00 (after the refresh of the data mart occurs). For this purpose, a personal gateway was created: