UNIVERSITY OF TECHNOLOGY

HOCHIMINH NATIONAL UNIVERSITY

Faculty of Computer Science and Engineering

Course: Data Warehouses and Decision Support Systems (CO4031)

Assignment
Customer Segmentation

CLASS: CC01 – HK251
Lecturer: PHAN TRỌNG NHÂN
TEAM: 7

No.	ID	NAME
1	2252545	Nguyễn Bình Nguyên
2	2052506	Đỗ Hoàng Thái Hưng
3	2053437	Nguyễn Tiến Thành
4	2211720	Khưu Vĩnh Kiên
5	2211078	Lê Hồng Anh Hoàng

Thành phố Hồ Chí Minh – 2025

Table content

I. Introduction

Customer segmentation is a critical component of modern data-driven decision-making, enabling organizations to understand behavioral patterns, identify high-value customers, and tailor marketing strategies accordingly. However, achieving effective segmentation requires a robust analytical foundation—something that traditional transactional systems (OLTP) are not designed to provide. In the current environment, customer-related data resides across multiple schemas and tables, such as Sales, Person, Production, and Address, resulting in fragmentation and inconsistency. This scattered data structure prevents managers from obtaining a unified view of customer behavior.

Furthermore, OLTP systems focus primarily on daily operations and lack the analytical capabilities needed to compute essential metrics such as Recency, Frequency, Monetary value (RFM), Customer Lifetime Value (CLV), or behavioral indicators. The absence of a centralized analytical platform also means that insights cannot be visualized intuitively, making it difficult to support strategic decision-making. As a result, segmentation efforts become manual, inconsistent, and challenging to replicate.

To address these issues, our project proposes a comprehensive approach built on two key pillars: a Data Warehouse (DW) designed using a Star Schema to integrate and restructure customer-related data, and a Decision Support System (DSS) that transforms this unified data into actionable insights. By implementing a structured ETL pipeline, creating analytical metrics, and developing an automated segmentation workflow, we aim to overcome the limitations of the current system and provide a practical, scalable solution for customer segmentation.

II. How to Solve the Goal Challenges

1. Customer data is scattered across multiple systems

Customer information is distributed across various schemas such as Person, Sales, Production, and Address, making it difficult to obtain a unified analytical view. To address this, we designed a centralized Data Warehouse using a Star Schema consisting of FactSales and four core Dimensions (Customer, Product, Geography, Date). This structure consolidates all relevant data into a single repository, reduces the complexity of the original OLTP model, standardizes business keys, and provides a stable foundation for multi-dimensional analysis and customer segmentation.

2. OLTP structure is not optimized for analytical workloads

The highly normalized OLTP database requires complex joins and cannot efficiently support analytical computations. To solve this issue, we implemented an ETL pipeline using SSIS to extract, clean, and transform data before loading it into the Data Warehouse. Through operations such as Sort, Merge Join, Conditional Split, and Derived Columns, the ETL process removes irrelevant records, standardizes data formats, and restructures relationships to fit the DW design. This transformation ensures that transactional data is optimized for analytical queries and subsequent segmentation tasks.

3. Difficulty computing analytical metrics (RFM, behavior indicators)

The operational database cannot directly provide essential analytical metrics required for segmentation. Using the Data Warehouse as the foundation, we developed an analytical metrics layer that computes Recency, Frequency, and Monetary (RFM) scores, along with behavioral features such as geographic patterns, product preferences, and weekend shopping behavior derived from DimDate. These metrics offer a comprehensive understanding of customer behavior and enable the classification of customers into meaningful segments such as VIP, Loyal, At-risk, or Low-value.

4. Lack of visualization and decision-making support

Without a proper visualization layer, managers cannot easily interpret customer behavior or track key performance indicators. To overcome this, we developed a Decision Support System using BI tools like Power BI or Tableau to present data through interactive dashboards and analytical reports. DSS enables users to visualize trends across time, regions, and product categories, analyze RFM segments, and perform drill-down operations for deeper insights. This enhances data-driven decision-making and improves the effectiveness of marketing and customer retention strategies.

III. Data warehouse design and management

1. Executive Summary

The objective of this phase was to transition the transactional data (OLTP) from the CompanyX database into a denormalized Data Warehouse (OLAP) optimized for decision support. The primary goal is to enable Customer Segmentation analysis (clustering, RFM analysis) by restructuring complex relational tables into a Star Schema.

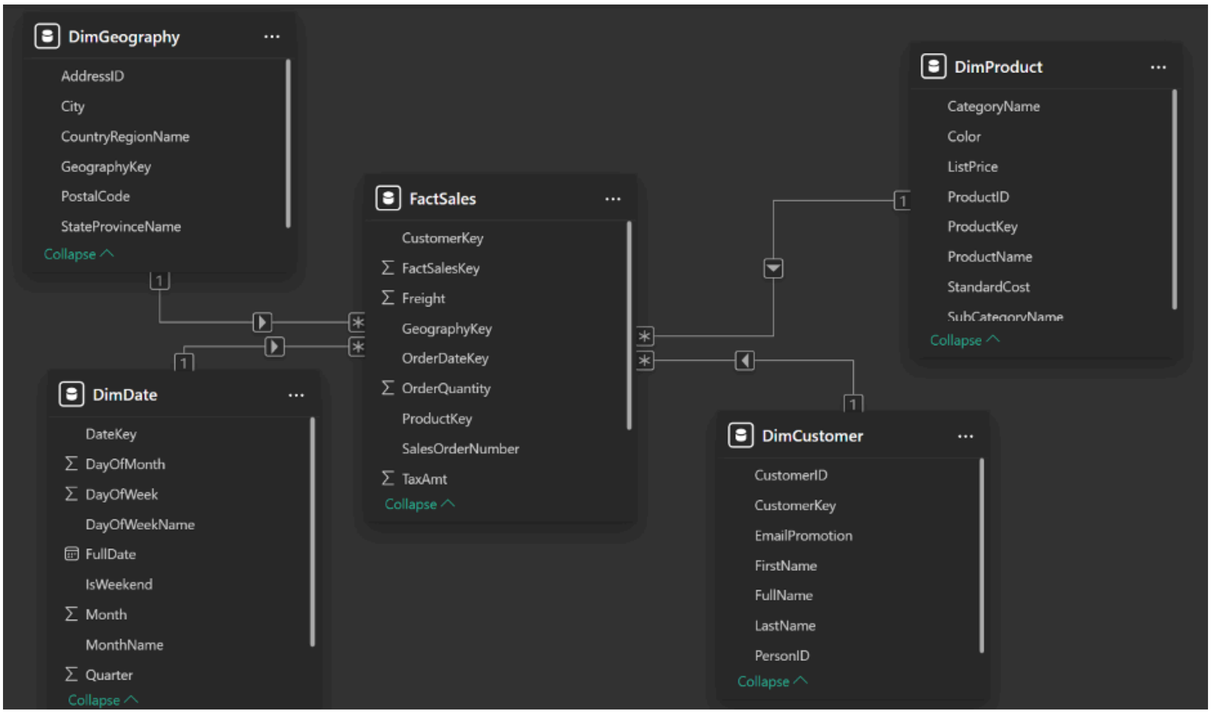
2. Data Warehouse Design (Star Schema)

To facilitate high-performance reporting on customer behavior, we designed a Star Schema centered around sales transactions.

2.1 The Schema Architecture

- Fact Table: FactSales (Contains quantitative metrics: Revenue, Quantity, Counts).

- Dimension Tables: DimCustomer, DimProduct, DimGeography, DimDate.



2.2 Dimensional Modeling Logic

- DimCustomer: Aggregated data from Sales.Customer and Person.Person. We focused on "Individual Customers" rather than Stores to support B2C segmentation.

- DimProduct: Flattened the snowflake hierarchy (Product -> Subcategory -> Category) into a single table to allow analysis by Category (e.g., "Bike Buyers") without complex joins in the visualization layer.

- DimGeography: Consolidated City, State, and Country data to enable geographic clustering of high-value customers.

- DimDate: A comprehensive time dimension procedurally generated (covering 2010–2040) rather than extracted from the source. It utilizes an Integer Smart Key (Format: YYYYMMDD) to optimize join performance with the Fact table. It includes derived attributes such as IsWeekend and DayOfWeekName to support behavioral segmentation (e.g., identifying customers who primarily shop on weekends).

3. ETL Process Implementation (SSIS)

We utilized SQL Server Integration Services (SSIS) in Visual Studio to extract, transform, and load (ETL) the data. We prioritized using SSIS Toolbox components (Merge Join, Sort, Derived Column) over raw SQL queries to ensure visual data flow management.

3.1 Connection Managers

- We established OLE DB connections using the Microsoft OLE DB Driver for SQL Server to ensure modern driver compatibility.

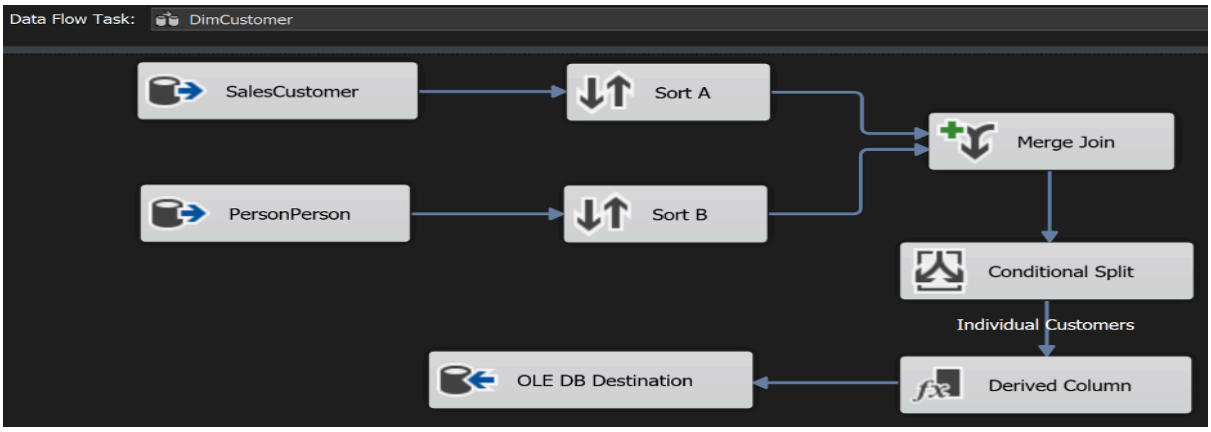
- Source: CompanyX (OLTP)

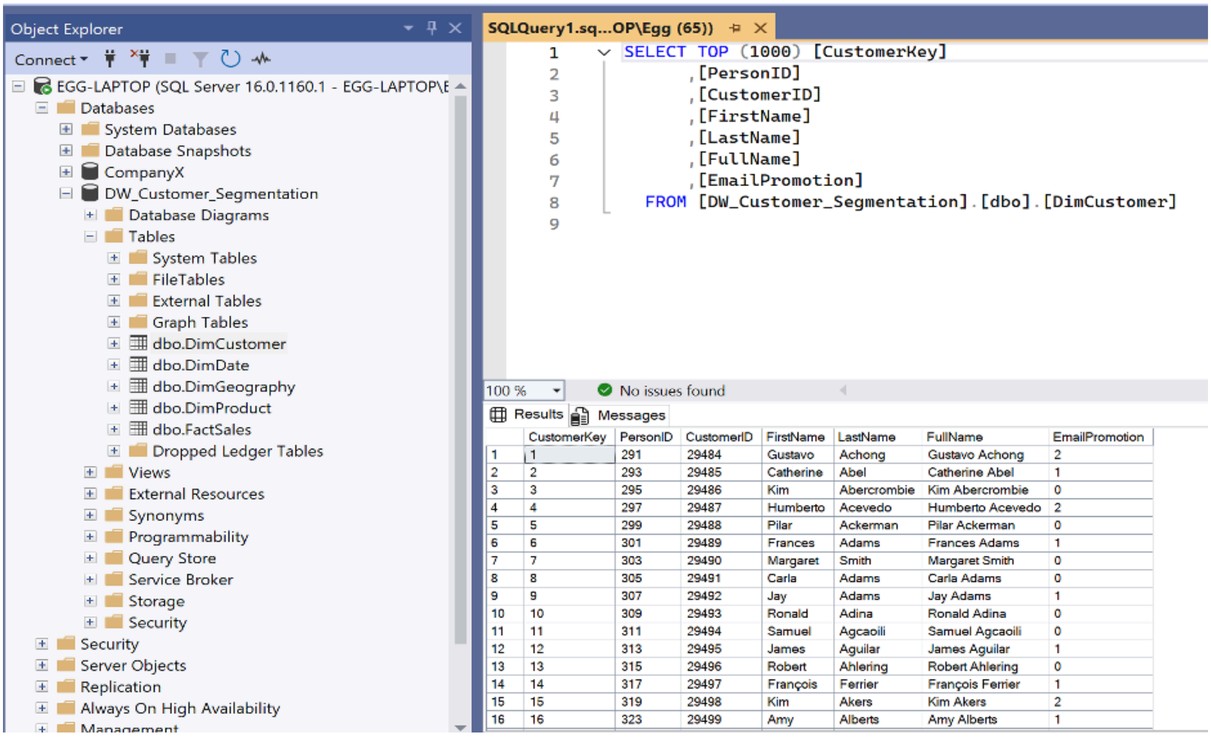
- Destination: DW_Customer_Segmentation (OLAP)

3.2 Dimension Packages

A. DimCustomer

- Extracted data from Person and Customer tables.
- Applied Sort transformations on BusinessEntityID and PersonID.
- Performed an Inner Join to merge attributes.
- Used Derived Column to concatenate FirstName + LastName into FullName.
- Used Conditional Split to filter out non-individual customers (Stores).





```sql
SELECT TOP (1000) [CustomerKey]
      ,[PersonID]
      ,[CustomerID]
      ,[FirstName]
      ,[LastName]
      ,[FullName]
      ,[EmailPromotion]
  FROM [DW_Customer_Segmentation].[dbo].[DimCustomer]
```

	CustomerKey	PersonID	CustomerID	FirstName	LastName	FullName	EmailPromotion
1	1	291	29484	Gustavo	Achong	Gustavo Achong	2
2	2	293	29485	Catherine	Abel	Catherine Abel	1
3	3	295	29486	Kim	Abercrombie	Kim Abercrombie	0
4	4	297	29487	Humberto	Acevedo	Humberto Acevedo	2
5	5	299	29488	Pilar	Ackerman	Pilar Ackerman	0
6	6	301	29489	Frances	Adams	Frances Adams	1
7	7	303	29490	Margaret	Smith	Margaret Smith	0
8	8	305	29491	Carla	Adams	Carla Adams	0
9	9	307	29492	Jay	Adams	Jay Adams	1
10	10	309	29493	Ronald	Adina	Ronald Adina	0
11	11	311	29494	Samuel	Agcaoili	Samuel Agcaoili	0
12	12	313	29495	James	Aguilar	James Aguilar	1
13	13	315	29496	Robert	Ahlering	Robert Ahlering	0
14	14	317	29497	François	Ferrier	François Ferrier	1
15	15	319	29498	Kim	Akers	Kim Akers	2
16	16	323	29499	Amy	Alberts	Amy Alberts	1

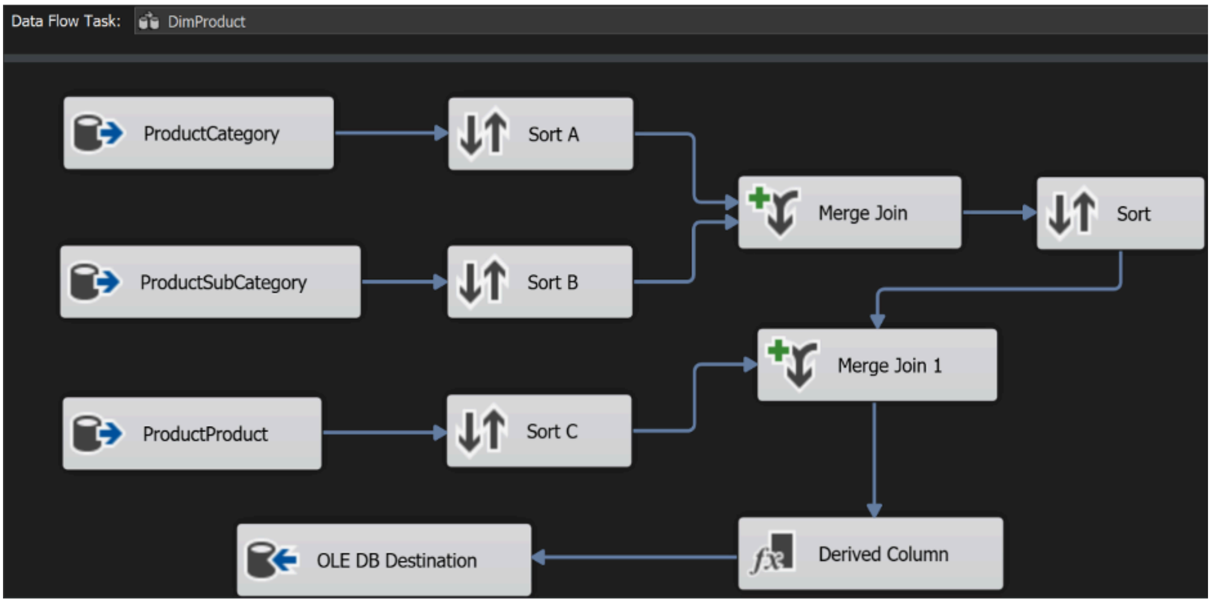
B. DimProduct

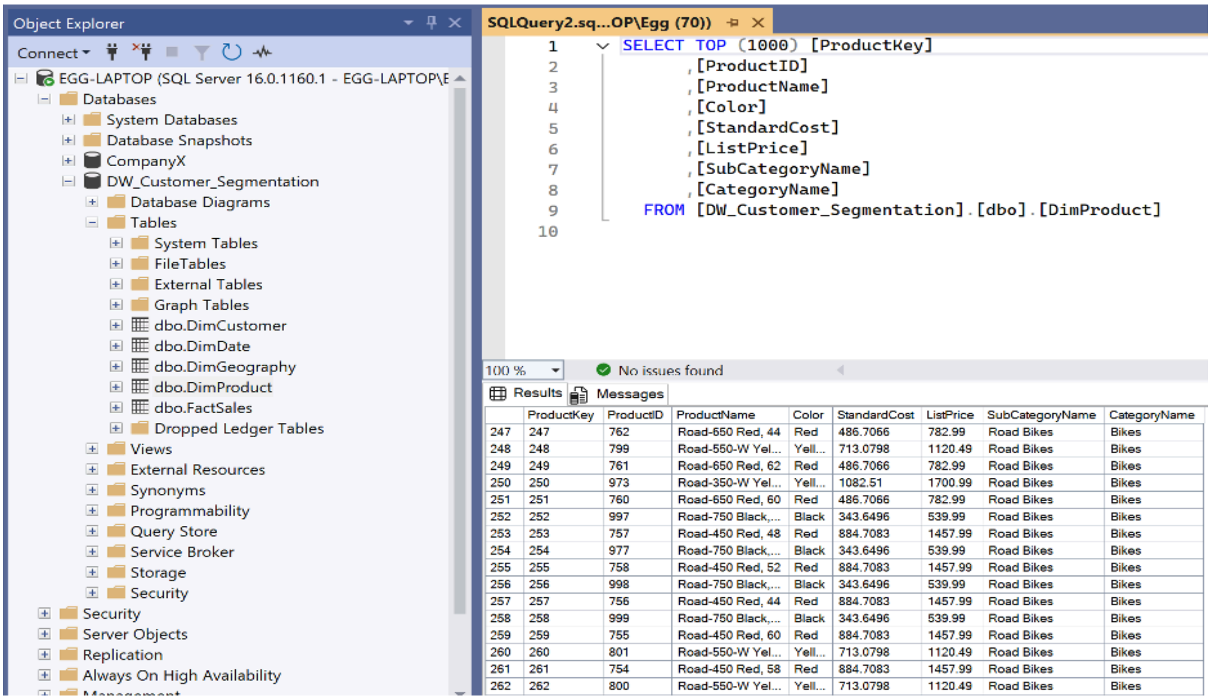
- Merged Category and Subcategory.

- Re-sorted the output.

- Merged the result with Product.

- Used Left Joins to ensure products with missing categories were not dropped.



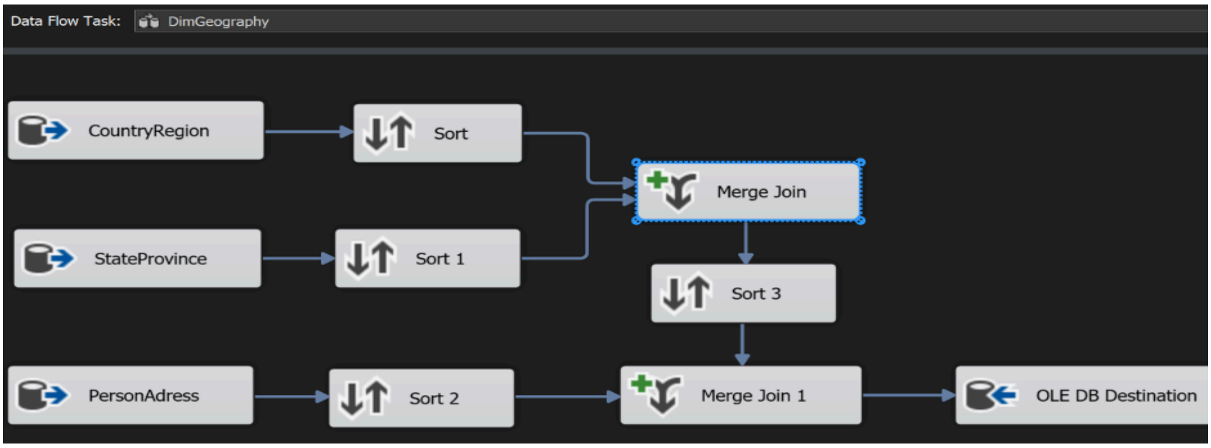


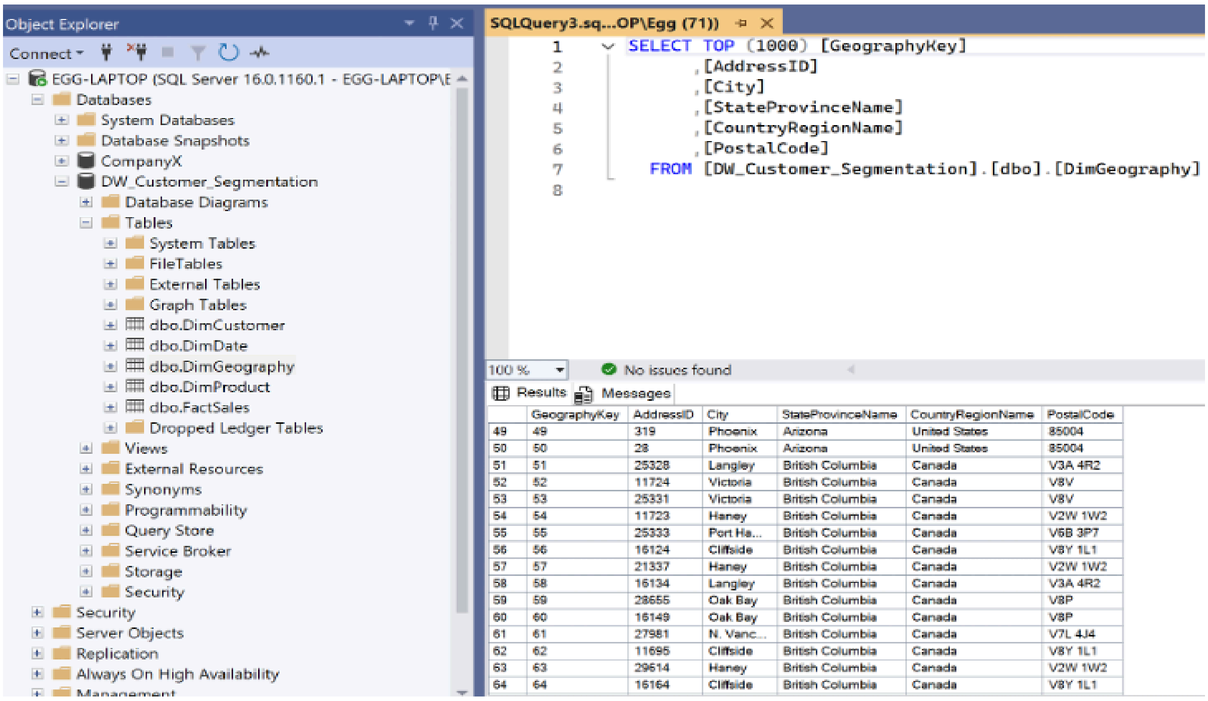
```
SELECT TOP (1000) [ProductKey]
      ,[ProductID]
      ,[ProductName]
      ,[Color]
      ,[StandardCost]
      ,[ListPrice]
      ,[SubCategoryName]
      ,[CategoryName]
  FROM [DW_Customer_Segmentation].[dbo].[DimProduct]
```

	ProductKey	ProductID	ProductName	Color	StandardCost	ListPrice	SubCategoryName	CategoryName
247	247	762	Road-650 Red, 44	Red	486.7066	782.99	Road Bikes	Bikes
248	248	799	Road-550-W Yel...	Yell...	713.0798	1120.49	Road Bikes	Bikes
249	249	761	Road-650 Red, 62	Red	486.7066	782.99	Road Bikes	Bikes
250	250	973	Road-350-W Yel...	Yell...	1082.51	1700.99	Road Bikes	Bikes
251	251	760	Road-650 Red, 60	Red	486.7066	782.99	Road Bikes	Bikes
252	252	997	Road-750 Black,...	Black	343.6496	539.99	Road Bikes	Bikes
253	253	757	Road-450 Red, 48	Red	884.7083	1457.99	Road Bikes	Bikes
254	254	977	Road-750 Black,...	Black	343.6496	539.99	Road Bikes	Bikes
255	255	758	Road-450 Red, 52	Red	884.7083	1457.99	Road Bikes	Bikes
256	256	998	Road-750 Black,...	Black	343.6496	539.99	Road Bikes	Bikes
257	257	756	Road-450 Red, 44	Red	884.7083	1457.99	Road Bikes	Bikes
258	258	999	Road-750 Black,...	Black	343.6496	539.99	Road Bikes	Bikes
259	259	755	Road-450 Red, 60	Red	884.7083	1457.99	Road Bikes	Bikes
260	260	801	Road-550-W Yel...	Yell...	713.0798	1120.49	Road Bikes	Bikes
261	261	754	Road-450 Red, 58	Red	884.7083	1457.99	Road Bikes	Bikes
262	262	800	Road-550-W Yel...	Yell...	713.0798	1120.49	Road Bikes	Bikes

C. DimGeography

- First Merge: Joined Person.CountryRegion and Person.StateProvince using the CountryRegionCode.
- Second Merge: Joined the output with Person.Address using the StateProvinceID.
- Optimization: Included AddressID in the final output. This Business Key acts as the critical link to the Fact table (via BillToAddressID), allowing us to map every sale to a specific city and region accurately.





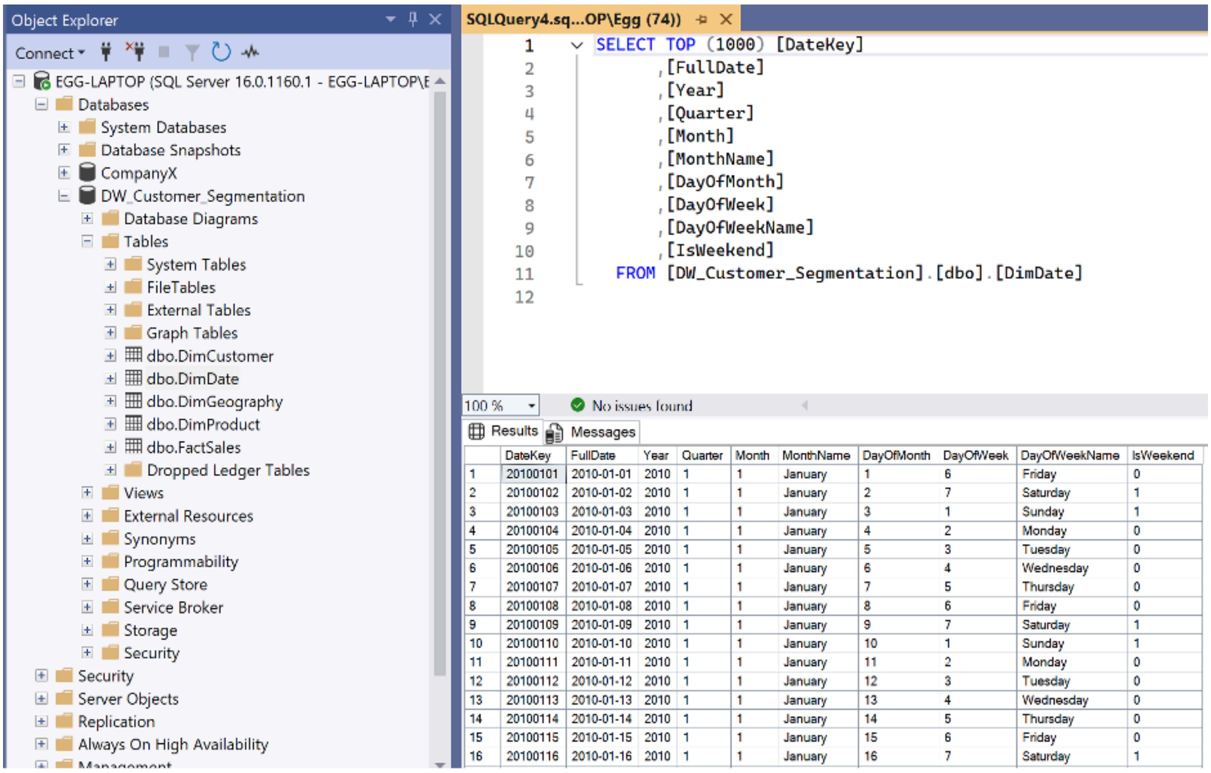
```sql
SELECT TOP (1000) [GeographyKey]
      ,[AddressID]
      ,[City]
      ,[StateProvinceName]
      ,[CountryRegionName]
      ,[PostalCode]
  FROM [DW_Customer_Segmentation].[dbo].[DimGeography]
```

	GeographyKey	AddressID	City	StateProvinceName	CountryRegionName	PostalCode
49	49	319	Phoenix	Arizona	United States	85004
50	50	28	Phoenix	Arizona	United States	85004
51	51	25328	Langley	British Columbia	Canada	V3A 4R2
52	52	11724	Victoria	British Columbia	Canada	V8V
53	53	25331	Victoria	British Columbia	Canada	V8V
54	54	11723	Haney	British Columbia	Canada	V2W 1W2
55	55	25333	Port Ha...	British Columbia	Canada	V6B 3P7
56	56	16124	Cliffside	British Columbia	Canada	V8Y 1L1
57	57	21337	Haney	British Columbia	Canada	V2W 1W2
58	58	16134	Langley	British Columbia	Canada	V3A 4R2
59	59	28655	Oak Bay	British Columbia	Canada	V8P
60	60	16149	Oak Bay	British Columbia	Canada	V8P
61	61	27981	N. Vanc...	British Columbia	Canada	V7L 4J4
62	62	11695	Cliffside	British Columbia	Canada	V8Y 1L1
63	63	29614	Haney	British Columbia	Canada	V2W 1W2
64	64	16164	Cliffside	British Columbia	Canada	V8Y 1L1

D. DimDate

We generated a comprehensive date table (2010–2040) using a SQL script to populate DimDate. This table uses an Integer Key (e.g., 20251129) for performance optimization during joining.

```sql
USE DW_Customer_Segmentation;
GO

-- 1. Create the Table (if it doesn't exist yet)
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'DimDate')
BEGIN
      CREATE TABLE DimDate (
      DateKey INT PRIMARY KEY,      -- Format: 20251126
      FullDate DATE,
      Year INT,
      Quarter INT,
      Month INT,
      MonthName NVARCHAR(15),
      DayOfMonth INT,
      DayOfWeek INT,
      DayOfWeekName NVARCHAR(15),
      IsWeekend BIT                 -- Useful for segmentation!
      );
END

-- 2. Clear existing data to prevent duplicates
TRUNCATE TABLE DimDate;

-- 3. Generate and Insert Data (2010 to 2040)
DECLARE @StartDate DATE = '2010-01-01';
DECLARE @EndDate DATE = '2040-12-31';

WITH DateSequence AS (
      SELECT @StartDate AS DateValue
      UNION ALL
      SELECT DATEADD(day, 1, DateValue)
      FROM DateSequence
      WHERE DateValue < @EndDate
)
INSERT INTO DimDate (
      DateKey,
      FullDate,
      Year,
      Quarter,
      Month,
      MonthName,
      DayOfMonth,
      DayOfWeek,
      DayOfWeekName,
      IsWeekend
)
SELECT
      -- Create Integer Key (e.g., 20251126)
      (YEAR(DateValue) * 10000) + (MONTH(DateValue) * 100) +
DAY(DateValue),

      DateValue,
      YEAR(DateValue),
      DATEPART(qq, DateValue),      -- Quarter (1-4)
      MONTH(DateValue),
      DATENAME(mm, DateValue),      -- Month Name (January, etc.)
      DAY(DateValue),
      DATEPART(dw, DateValue),      -- Day of Week Number
      DATENAME(dw, DateValue),      -- Day Name (Monday, etc.)

      CASE
      WHEN DATENAME(dw, DateValue) IN ('Saturday', 'Sunday') THEN 1
      ELSE 0
      END
FROM DateSequence
OPTION (MAXRECURSION 0); -- Allows more than 100 rows
```



```sql
SELECT TOP (1000) [DateKey]
      ,[FullDate]
      ,[Year]
      ,[Quarter]
      ,[Month]
      ,[MonthName]
      ,[DayOfMonth]
      ,[DayOfWeek]
      ,[DayOfWeekName]
      ,[IsWeekend]
  FROM [DW_Customer_Segmentation].[dbo].[DimDate]
```

No issues found

	DateKey	FullDate	Year	Quarter	Month	MonthName	DayOfMonth	DayOfWeek	DayOfWeekName	IsWeekend
1	20100101	2010-01-01	2010	1	1	January	1	6	Friday	0
2	20100102	2010-01-02	2010	1	1	January	2	7	Saturday	1
3	20100103	2010-01-03	2010	1	1	January	3	1	Sunday	1
4	20100104	2010-01-04	2010	1	1	January	4	2	Monday	0
5	20100105	2010-01-05	2010	1	1	January	5	3	Tuesday	0
6	20100106	2010-01-06	2010	1	1	January	6	4	Wednesday	0
7	20100107	2010-01-07	2010	1	1	January	7	5	Thursday	0
8	20100108	2010-01-08	2010	1	1	January	8	6	Friday	0
9	20100109	2010-01-09	2010	1	1	January	9	7	Saturday	1
10	20100110	2010-01-10	2010	1	1	January	10	1	Sunday	1
11	20100111	2010-01-11	2010	1	1	January	11	2	Monday	0
12	20100112	2010-01-12	2010	1	1	January	12	3	Tuesday	0
13	20100113	2010-01-13	2010	1	1	January	13	4	Wednesday	0
14	20100114	2010-01-14	2010	1	1	January	14	5	Thursday	0
15	20100115	2010-01-15	2010	1	1	January	15	6	Friday	0
16	20100116	2010-01-16	2010	1	1	January	16	7	Saturday	1

3.3 The Fact Table (FactSales)

- The FactSales package is the central pipeline connecting all dimensions.

- Source: Joined SalesOrderHeader and SalesOrderDetail using Merge Join.

- Surrogate Key Replacement (Lookups):

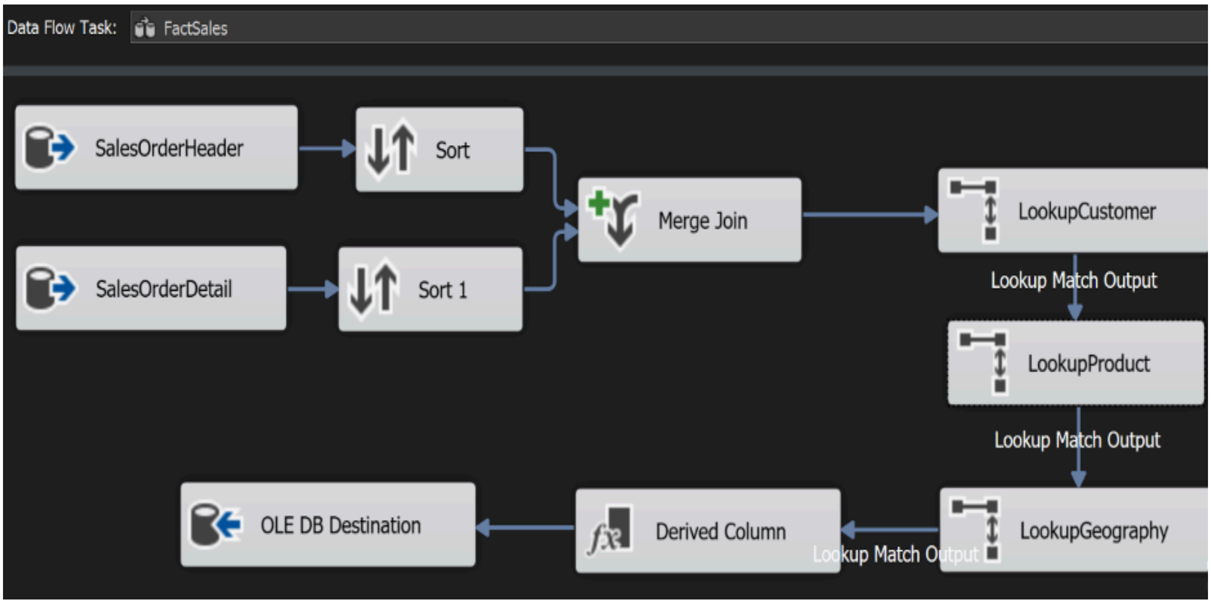
  + Used Lookup Components to replace business keys (e.g., CustomerID) with Data Warehouse keys (CustomerKey).

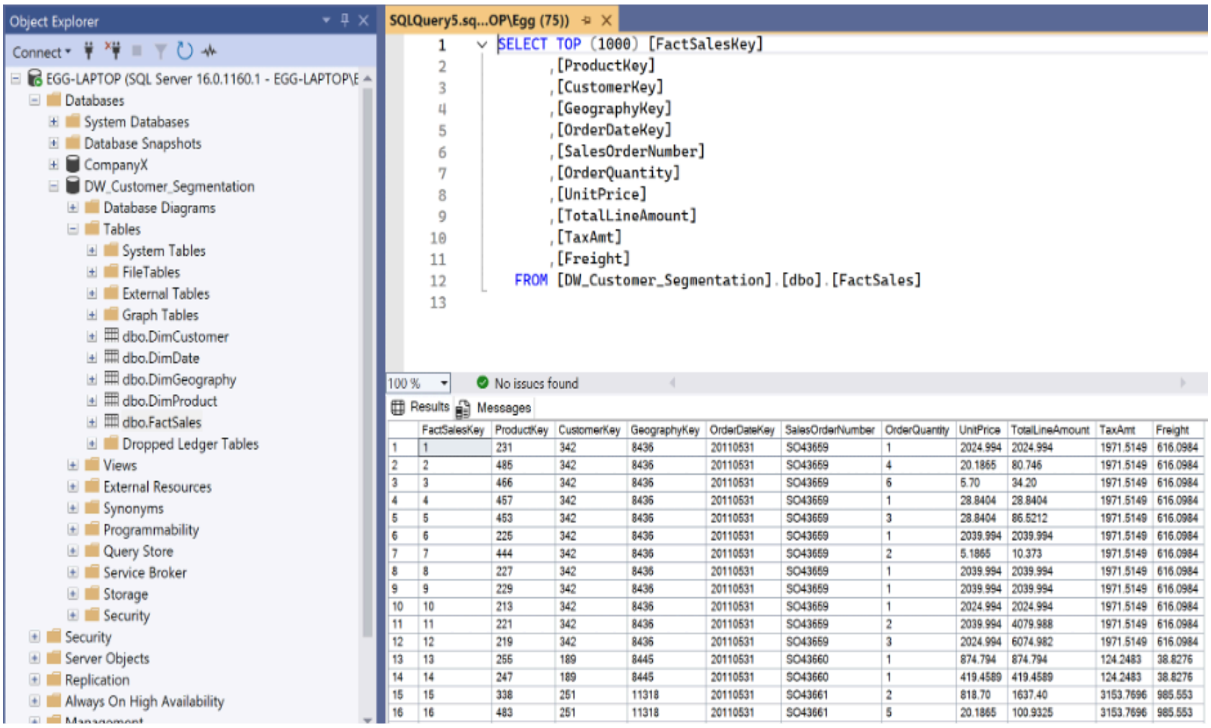
  + Mapped DimCustomer, DimProduct, and DimGeography.

- Date Conversion:

  + Used a Derived Column calculation to convert OrderDate (DateTime) into OrderDateKey (Integer) to match the DimDate logic.

  + Formula:(YEAR(OrderDate)*10000)+(MONTH(OrderDate)* 100) + DAY(OrderDate)





```sql
SELECT TOP (1000) [FactSalesKey]
      ,[ProductKey]
      ,[CustomerKey]
      ,[GeographyKey]
      ,[OrderDateKey]
      ,[SalesOrderNumber]
      ,[OrderQuantity]
      ,[UnitPrice]
      ,[TotalLineAmount]
      ,[TaxAmt]
      ,[Freight]
  FROM [DW_Customer_Segmentation].[dbo].[FactSales]
```

	FactSalesKey	ProductKey	CustomerKey	GeographyKey	OrderDateKey	SalesOrderNumber	OrderQuantity	UnitPrice	TotalLineAmount	TaxAmt	Freight
1	1	231	342	8436	20110531	SO43659	1	2024.994	2024.994	1971.5149	616.0984
2	2	485	342	8436	20110531	SO43659	4	20.1865	80.746	1971.5149	616.0984
3	3	466	342	8436	20110531	SO43659	6	5.70	34.20	1971.5149	616.0984
4	4	457	342	8436	20110531	SO43659	1	28.8404	28.8404	1971.5149	616.0984
5	5	453	342	8436	20110531	SO43659	3	28.8404	86.5212	1971.5149	616.0984
6	6	225	342	8436	20110531	SO43659	1	2039.994	2039.994	1971.5149	616.0984
7	7	444	342	8436	20110531	SO43659	2	5.1865	10.373	1971.5149	616.0984
8	8	227	342	8436	20110531	SO43659	1	2039.994	2039.994	1971.5149	616.0984
9	9	229	342	8436	20110531	SO43659	1	2039.994	2039.994	1971.5149	616.0984
10	10	213	342	8436	20110531	SO43659	1	2024.994	2024.994	1971.5149	616.0984
11	11	221	342	8436	20110531	SO43659	2	2039.994	4079.988	1971.5149	616.0984
12	12	219	342	8436	20110531	SO43659	3	2024.994	6074.982	1971.5149	616.0984
13	13	255	189	8445	20110531	SO43660	1	874.794	874.794	124.2483	38.8276
14	14	247	189	8445	20110531	SO43660	1	419.4589	419.4589	124.2483	38.8276
15	15	338	251	11318	20110531	SO43661	2	818.70	1637.40	3153.7696	985.553
16	16	483	251	11318	20110531	SO43661	5	20.1865	100.9325	3153.7696	985.553

IV. Decision Support System (DSS) Design & Implementation

1. Executive Summary

Following the successful deployment of the Data Warehouse (Phase 1), Phase 2 focused on constructing the **Decision Support System (DSS)**. The objective was to transition from passive data storage to active intelligence. We implemented a **Semantic Layer** using SSAS Tabular to model "Customer Profitability" and "Behavioral Segmentation" (Gold/Silver/Bronze). A rule-based **Action Engine** was developed to automatically recommend specific marketing actions (e.g., "Upsell" vs. "Flag Unprofitable") for every customer, which is visualized in an interactive Power BI dashboard.

2. DSS Design (Logical Architecture)

The Decision Support System (DSS) was engineered using a standard **Microsoft BI 3-Tier Architecture**. This design pattern was selected to decouple data storage from business logic and visualization, ensuring scalability, consistency, and high performance for end-user queries.

2.1 The 3-Tier Architecture

The logical flow of the system moves from raw data processing to actionable intelligence through three distinct layers:

- The Data Foundation Layer (SQL Server):
    + **Role:** Acts as the centralized storage engine. It hosts the Star Schema (FactSales and Dimension tables).
- The Semantic & Logic Layer (SSAS Tabular):

  + **Role:** The "Brain" of the DSS. It holds the in-memory data model, defines the business logic (Measures and Calculated Columns), and handles the aggregation math.
  + This is the most critical design choice. By placing logic here—rather than in Power BI—we create a **"Single Version of the Truth."** Whether a user connects via Excel, Power BI, or a custom app, the definition of "Gold Customer" or "Total Profit" remains identical. It also offloads heavy processing from the reporting tool.
- The Presentation & Decision Layer (Power BI):
  + **Role:** The interactive interface. It queries the Semantic Layer to visualize trends and filter actionable lists.
  + By using a "Live Connection" to SSAS, Power BI remains lightweight. It does not import data; it simply sends DAX queries to the server, allowing for instant filtering of millions of rows without latency.

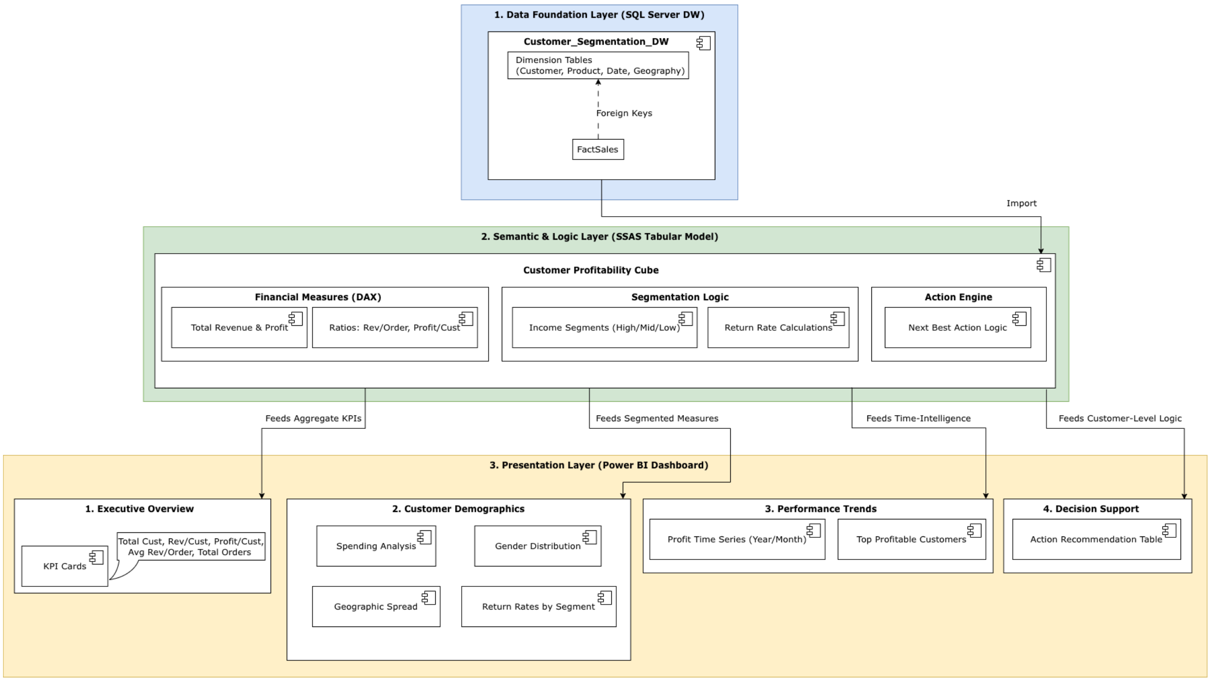
2.2 Technology Selection: Why SSAS Tabular?

- For this specific Customer Segmentation project, we selected **SSAS Tabular Mode** over the traditional **Multidimensional (Cube)** mode.

- Rationale for Tabular:

+ **Columnar Storage:** Tabular uses an in-memory, column-oriented database engine. This is significantly faster for the type of queries used in segmentation (e.g., *distinct counts* of customers, summing specific columns like `TotalProfit`). Multidimensional models struggle with distinct counts on large datasets.

- **DAX vs. MDX:** The logic for our "Action Engine" (nested `SWITCH` statements and `IF` logic) is native to **DAX** (Data Analysis Expressions). Implementing this dynamic row-level segmentation logic in Multidimensional (MDX) would be unnecessarily complex and less performant.
- **Flexibility:** Tabular models allow for agile development. Adding a new Calculated Column (like `SpendingSegment`) is instantaneous, whereas Multidimensional models often require complex reprocessing and structural changes.



3. DSS Implementation (SSAS Tabular)

The implementation was executed in Visual Studio using the **Analysis Services Tabular Project** template.

3.1 Measure Definitions (The Financial Core)

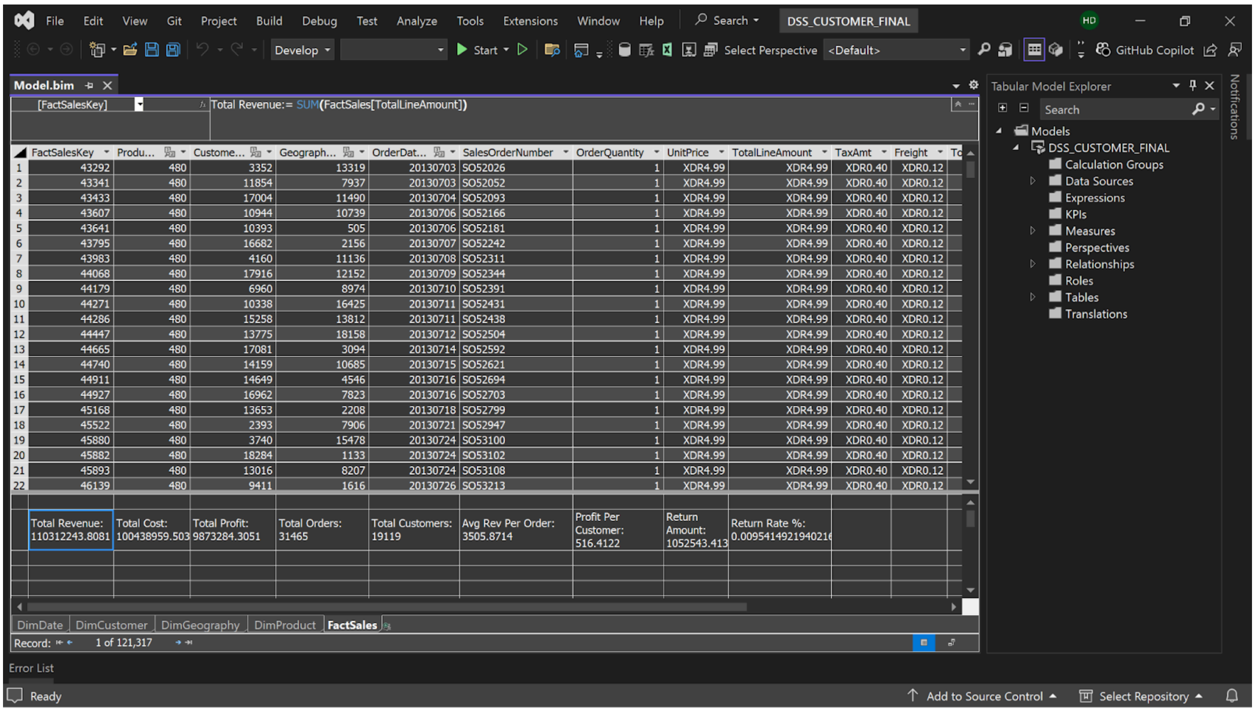
We defined DAX Measures to handle aggregate KPIs. These calculate dynamically based on the user's filter context.

Total Revenue:= SUM(FactSales[TotalLineAmount])

Total Cost:= SUM(FactSales[TotalProductCost])

Total Profit:= [Total Revenue] - [Total Cost]

Return Rate %:= DIVIDE([Return Amount], [Total Revenue], 0)



[FactSalesKey]

```
Total Revenue:= SUM(FactSales[TotalLineAmount])
```

	FactSalesKey	Produ...	Custome...	Geograph...	OrderDat...	SalesOrderNumber	OrderQuantity	UnitPrice	TotalLineAmount	TaxAmt	Freight	To
1	43292	480	3352	13319	20130703	SO52026	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
2	43341	480	11854	7937	20130703	SO52052	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
3	43433	480	17004	11490	20130704	SO52093	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
4	43607	480	10944	10739	20130706	SO52166	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
5	43641	480	10393	505	20130706	SO52181	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
6	43795	480	16682	2156	20130707	SO52242	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
7	43983	480	4160	11136	20130708	SO52311	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
8	44068	480	17916	12152	20130709	SO52344	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
9	44179	480	6960	8974	20130710	SO52391	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
10	44271	480	10338	16425	20130711	SO52431	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
11	44286	480	15258	13812	20130711	SO52438	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
12	44447	480	13775	18158	20130712	SO52504	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
13	44665	480	17081	3094	20130714	SO52592	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
14	44740	480	14159	10685	20130715	SO52621	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
15	44911	480	14649	4546	20130716	SO52694	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
16	44927	480	16962	7823	20130716	SO52703	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
17	45168	480	13653	2208	20130718	SO52799	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
18	45522	480	2393	7906	20130721	SO52947	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
19	45880	480	3740	15478	20130724	SO53100	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
20	45882	480	18284	1133	20130724	SO53102	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
21	45893	480	13016	8207	20130724	SO53108	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	
22	46139	480	9411	1616	20130726	SO53213	1	XDR4.99	XDR4.99	XDR0.40	XDR0.12	

Total Revenue: 110312243.8081	Total Cost: 100438959.503	Total Profit: 9873284.3051	Total Orders: 31465	Total Customers: 19119	Avg Rev Per Order: 3505.8714	Profit Per Customer: 516.4122	Return Amount: 1052543.413	Return Rate %: 0.0095414921940216

3.2 Customer Segmentation (Calculated Columns)

We implemented row-level logic in DimCustomer to classify customers based on their **Lifetime Spend** and **Lifetime Profit**.

- **Spending Segment:** Classifies customers into Gold (>=$5000), Silver (>=$1000), and Bronze (>0).
- **Lifetime Profit:** Calculates the net profit contribution of the customer to detect unprofitable relationships.

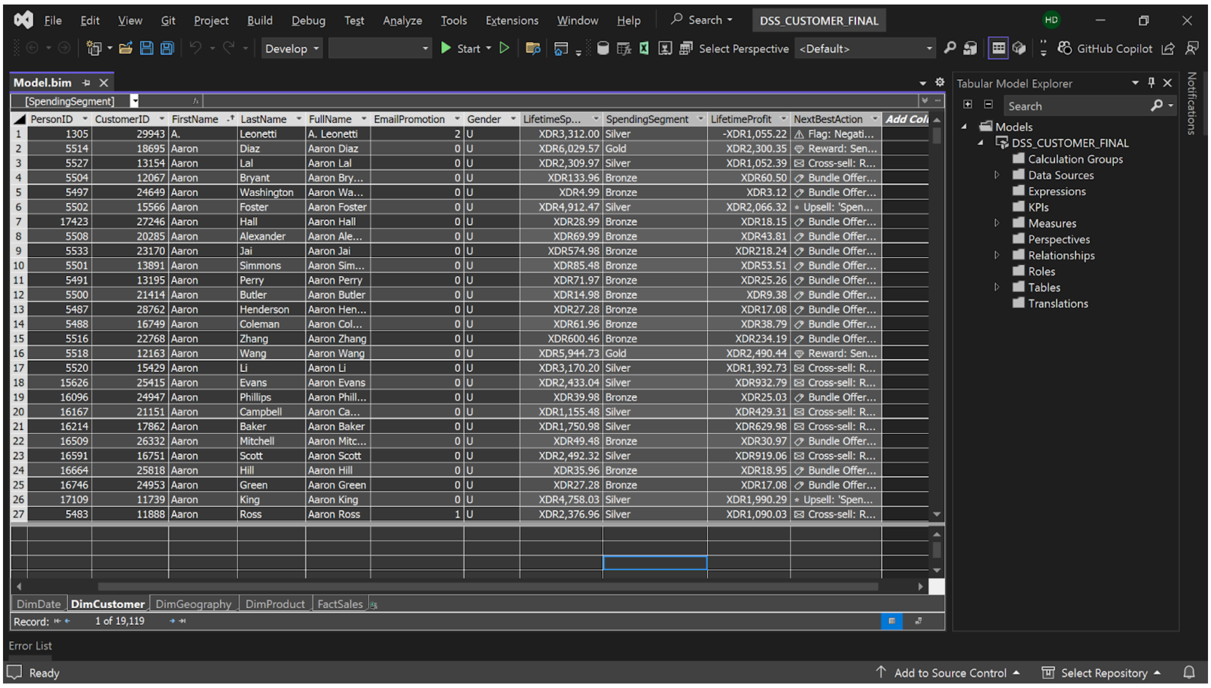
SpendingSegment= SWITCH(TRUE(), [LifetimeSpend] >= 5000, "Gold", [LifetimeSpend] >= 1000, "Silver", [LifetimeSpend] > 0, "Bronze", "Inactive")

LifetimeProfit=CALCULATE([Total Profit])

3.3 The Action Engine (Next Best Action)

The core of the DSS is a SWITCH statement that acts as a rule engine. It evaluates a customer's **Profitability** and **Segment** to assign a prescriptive action.

```
Logic Rules:
Unprofitable: If LifetimeProfit < 0 → Flag (Stop Marketing).
Gold: If Segment = Gold →  Reward (VIP Catalog).
Silver (High Potential): If Segment = Silver AND Spend >= 3500 → Upsell
(Push to Gold).
Silver (Standard): → Cross-sell.
Bronze: → Bundle Offer.
NextBestAction= SWITCH(
    TRUE(),

    // 1. BAD ACTOR: Low spender but returns a lot (Profit killer)
    [LifetimeProfit] < 0,
    "⚠️ Flag: Negative Value (High Returns)",

    // 2. THE VIP: Already at the top
    [SpendingSegment] == "Gold",
    "💎 Reward: Send 'Exclusive 2025 Catalog'",

    // 3. THE CLIMBER: Silver customer close to Gold
    [SpendingSegment] == "Silver" && [LifetimeSpend] >= 3500,
    "⭐ Upsell: 'Spend bit more to hit Gold Status'",

    // 4. THE CORE: Standard Silver customer
    [SpendingSegment] == "Silver",
    "✉️ Cross-sell: Recommend Premium Add-ons",

    // 5. THE NEWBIE: Bronze customer (Low spend)
    [SpendingSegment] == "Bronze",
    "🏷️ Bundle Offer: 'Buy 2 Get 10% Off'",

    // Default
    "Standard Marketing"
)
```



[SpendingSegment]

	PersonID	CustomerID	FirstName	LastName	FullName	EmailPromotion	Gender	LifetimeSp...	SpendingSegment	LifetimeProfit	NextBestAction	Add Col
1	1305	29943	A.	Leonetti	A. Leonetti	2	U	XDR3,312.00	Silver	-XDR1,055.22	⚠ Flag: Negati...	
2	5514	18695	Aaron	Diaz	Aaron Diaz	0	U	XDR6,029.57	Gold	XDR2,300.35	💎 Reward: Sen...	
3	5527	13154	Aaron	Lal	Aaron Lal	0	U	XDR2,309.97	Silver	XDR1,052.39	✉ Cross-sell: R...	
4	5504	12067	Aaron	Bryant	Aaron Bry...	0	U	XDR133.96	Bronze	XDR60.50	🏷 Bundle Offer...	
5	5497	24649	Aaron	Washington	Aaron Wa...	0	U	XDR4.99	Bronze	XDR3.12	🏷 Bundle Offer...	
6	5502	15566	Aaron	Foster	Aaron Foster	0	U	XDR4,912.47	Silver	XDR2,066.32	⋆ Upsell: 'Spen...	
7	17423	27246	Aaron	Hall	Aaron Hall	0	U	XDR28.99	Bronze	XDR18.15	🏷 Bundle Offer...	
8	5508	20285	Aaron	Alexander	Aaron Ale...	0	U	XDR69.99	Bronze	XDR43.81	🏷 Bundle Offer...	
9	5533	23170	Aaron	Jai	Aaron Jai	0	U	XDR574.98	Bronze	XDR218.24	🏷 Bundle Offer...	
10	5501	13891	Aaron	Simmons	Aaron Sim...	0	U	XDR85.48	Bronze	XDR53.51	🏷 Bundle Offer...	
11	5491	13195	Aaron	Perry	Aaron Perry	0	U	XDR71.97	Bronze	XDR25.26	🏷 Bundle Offer...	
12	5500	21414	Aaron	Butler	Aaron Butler	0	U	XDR14.98	Bronze	XDR9.38	🏷 Bundle Offer...	
13	5487	28762	Aaron	Henderson	Aaron Hen...	0	U	XDR27.28	Bronze	XDR17.08	🏷 Bundle Offer...	
14	5488	16749	Aaron	Coleman	Aaron Col...	0	U	XDR61.96	Bronze	XDR38.79	🏷 Bundle Offer...	
15	5516	22768	Aaron	Zhang	Aaron Zhang	0	U	XDR600.46	Bronze	XDR234.19	🏷 Bundle Offer...	
16	5518	12163	Aaron	Wang	Aaron Wang	0	U	XDR5,944.73	Gold	XDR2,490.44	💎 Reward: Sen...	
17	5520	15429	Aaron	Li	Aaron Li	0	U	XDR3,170.20	Silver	XDR1,392.73	✉ Cross-sell: R...	
18	15626	25415	Aaron	Evans	Aaron Evans	0	U	XDR2,433.04	Silver	XDR932.79	✉ Cross-sell: R...	
19	16096	24947	Aaron	Phillips	Aaron Phill...	0	U	XDR39.98	Bronze	XDR25.03	🏷 Bundle Offer...	
20	16167	21151	Aaron	Campbell	Aaron Ca...	0	U	XDR1,155.48	Silver	XDR429.31	✉ Cross-sell: R...	
21	16214	17862	Aaron	Baker	Aaron Baker	0	U	XDR1,750.98	Silver	XDR629.98	✉ Cross-sell: R...	
22	16509	26332	Aaron	Mitchell	Aaron Mitc...	0	U	XDR49.48	Bronze	XDR30.97	🏷 Bundle Offer...	
23	16591	16751	Aaron	Scott	Aaron Scott	0	U	XDR2,492.32	Silver	XDR919.06	✉ Cross-sell: R...	
24	16664	25818	Aaron	Hill	Aaron Hill	0	U	XDR35.96	Bronze	XDR18.95	🏷 Bundle Offer...	
25	16746	24953	Aaron	Green	Aaron Green	0	U	XDR27.28	Bronze	XDR17.08	🏷 Bundle Offer...	
26	17109	11739	Aaron	King	Aaron King	0	U	XDR4,758.03	Silver	XDR1,990.29	⋆ Upsell: 'Spen...	
27	5483	11888	Aaron	Ross	Aaron Ross	1	U	XDR2,376.96	Silver	XDR1,090.03	✉ Cross-sell: R...	

DimDate | DimCustomer | DimGeography | DimProduct | FactSales

Record: 1 of 19,119

Tabular Model Explorer

Search

- Models
  - DSS_CUSTOMER_FINAL
    - Calculation Groups
    - Data Sources
    - Expressions
    - KPIs
    - Measures
    - Perspectives
    - Relationships
    - Roles
    - Tables
    - Translations

Error List

Ready

Add to Source Control

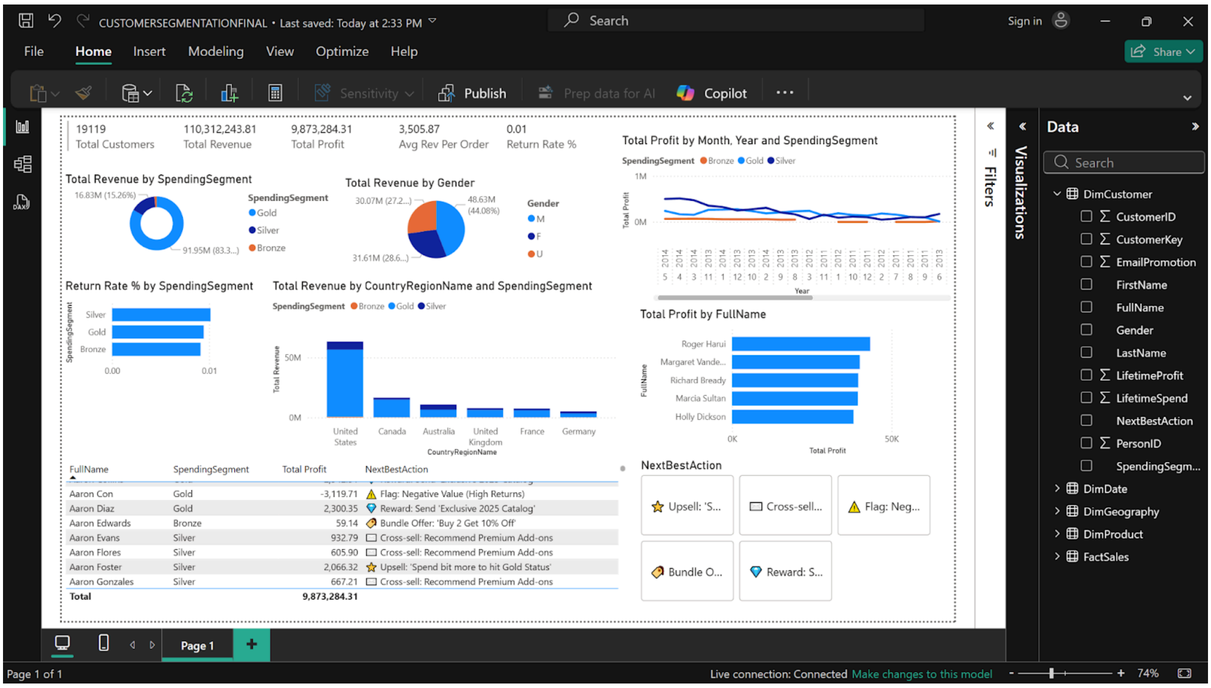
Select Repository

4. Dashboard Visualization (Power BI)

The Presentation Layer connects "Live" to the SSAS model, ensuring a single version of the truth.

- **Executive Scorecard:** Multi-row KPI cards showing Total Revenue, Total Profit, and Average Order Value.
- **Segmentation Analysis:**
  - + *Revenue by Segment (Donut Chart)*: Visualizes the contribution of Gold vs Silver vs Bronze customers.
  - + *Return Rate by Segment (Bar Chart)*: Identifies if higher-tier customers have higher return rates.
  - + *Revenue by Gender (Pie Chart)*: Uses the Title-derived gender logic.
- **Performance Trends:**
  - + *Profit by Month (Line Chart)*: Stacked by Segment to show growth trends over time.
  - + *Top 5 Customers (Bar Chart)*: Filtered by Total Profit.

- **Action Command Center (The DSS Interface):**
  - + **The Slicer:** A button slicer allowing the manager to filter the entire report by **Next Best Action**.
  - + **The List:** A detailed table providing the specific names and profit figures for the selected action group.



19119	110,312,243.81	9,873,284.31	3,505.87	0.01
Total Customers	Total Revenue	Total Profit	Avg Rev Per Order	Return Rate %

FullName	SpendingSegment	Total Profit	NextBestAction
Aaron Con	Gold	-3,119.71	⚠ Flag: Negative Value (High Returns)
Aaron Diaz	Gold	2,300.35	Reward: Send 'Exclusive 2025 Catalog'
Aaron Edwards	Bronze	59.14	Bundle Offer: 'Buy 2 Get 10% Off'
Aaron Evans	Silver	932.79	✉ Cross-sell: Recommend Premium Add-ons
Aaron Flores	Silver	605.90	✉ Cross-sell: Recommend Premium Add-ons
Aaron Foster	Silver	2,066.32	☆ Upsell: 'Spend bit more to hit Gold Status'
Aaron Gonzales	Silver	667.21	✉ Cross-sell: Recommend Premium Add-ons
Total		**9,873,284.31**	

5. Conclusion

The CompanyX DSS successfully transforms raw transaction data into strategic insights. By leveraging **SSAS Tabular** for the semantic layer, we achieved a scalable solution where business logic is centralized. The **Next Best Action** engine provides immediate value by converting complex profit analysis into simple, executable instructions for the marketing team.