# *Retail Analytics Mastery*

DEPI FINAL PROJECT

17 October, 2024

# *Team Members*

Omar Ayman

Youssef Mohamed

Mostafa Alaa

Ahmed Ashraf

Abdrehman Mohamed

# *Introduction*

# *Project Overview*

This project aims to analyze and explore key performance metrics related to sales, inventory management, and customer behavior using the AdventureWorks database. By leveraging data from sales transactions, customer information, and inventory records, the project will provide insights into sales trends, customer demographics, inventory optimization, and business performance.

# *Problem Statement*

AdventureWorks, a manufacturing company, wants to improve its decision-making by gaining deeper insights into its sales performance, customer preferences, and inventory management. However, the current data is not fully utilized, and critical business questions remain unanswered:

- Sales Performance: How can we optimize sales strategies across different regions and customer segments?
- Inventory: Is the current inventory management process efficient, or are there areas for improvement to reduce stockouts or overstocking?
- Customer Analysis: What are the key characteristics of the most profitable customers, and how can we tailor marketing efforts to attract similar profiles?

# *Objectives and Goals*

- **Sales Analysis**:
  - Identify sales trends over time, product performance, and regional sales distribution.
  - Track revenue growth and key sales metrics such as average order size and order frequency.
- **Inventory Management:**
  - Analyze inventory turnover, stock levels, and potential inefficiencies.
  - Identify products that are consistently overstocked or understocked to optimize inventory processes.
- **Customer Insights:**
  - Explore customer demographics (age, location, income) to identify key segments.
  - Analyze customer lifetime value (CLV) and purchasing behaviors to enhance customer retention.

# Technologies Reviewed

For this project, several technologies were reviewed to determine the most suitable tools for analyzing sales, inventory, an customer data:

1. SQL Server:
   - For building Staging and DataWarehouse associated with its constraints
2. SSIS (SQL Server Integration Services):
   - For ETL processes to load data into a data warehouse from various sources, including transactional sales data, inventory records, and customer information.
3. Power BI / Tableau:
   - Used as visualization tools for creating dashboards and reports on sales, inventory trends, and customer insights.
4. Python:
   - used ML libraries and models to predict sales, segment customers based on features, etc.... on Colab

5. Azure:
   - Upload our final data on storage account to derive from it our analysis for Dashboard on Synapse.

# 2. Milestones and Timeline

### Milestone 1: Project Planning and Requirement Gathering (3 days)

- Define the project scope, goals, and deliverables.

### Milestone 2: Data Extraction and ETL Setup (Weeks 1)

- Implement the ETL pipeline using SSIS to extract data from the AdventureWorks database.
- Design transformation rules and load data into a staging area.

### Milestone 3: Data Warehouse Design (Weeks 2)

- Create the star or snowflake schema for the data warehouse.
- Implement fact tables (e.g., FactInternetSales, FactResellerSales) and dimension tables (e.g., DimCustomer, DimProduct, DimDate).
- Load transformed data into the data warehouse.

### Milestone 4: Dashboard and Reporting Development (Weeks 3-4)

- Develop interactive dashboards for sales, inventory, and customer analytics using Power BI
- Create reports for sales trends, inventory levels, and customer demographics.

### Milestone 5: Implement ML models (Weeks 3-4)

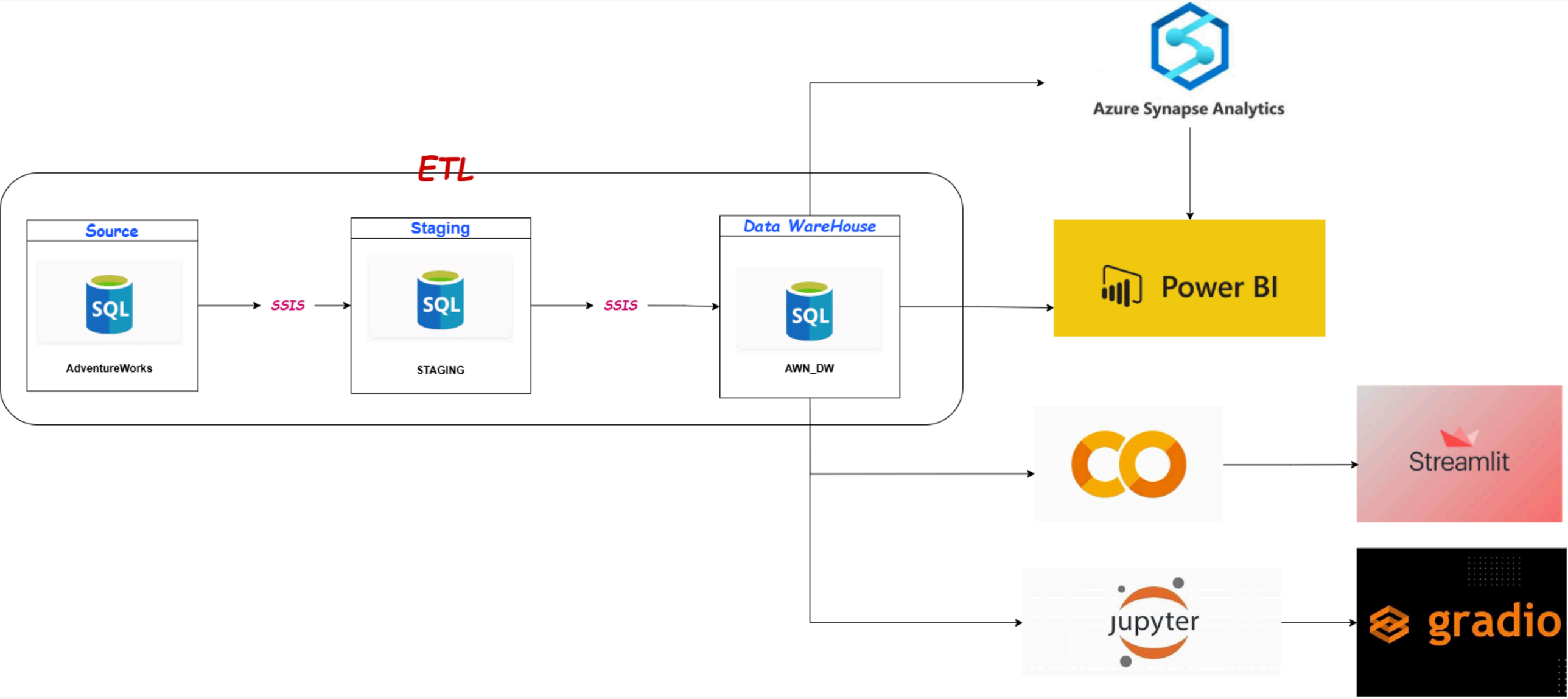
- Develop and train the chosen ML models for sales forecasting, inventory optimization, and customer segmentation.
- Evaluate the models using performance metrics like RMSE for forecasting, precision/recall for classification, and silhouette score for clustering.

### Milestone 6: Final Delivery and Documentation (Week 3-4)

- Finalize the system and deliver it to the stakeholders.
- Provide documentation on how to use the reports and dashboards.

# *Implemetation*

# *Data Pipeline*



# *Data flow using SSIS*

# *Data Warehouse using Microsoft SQL*

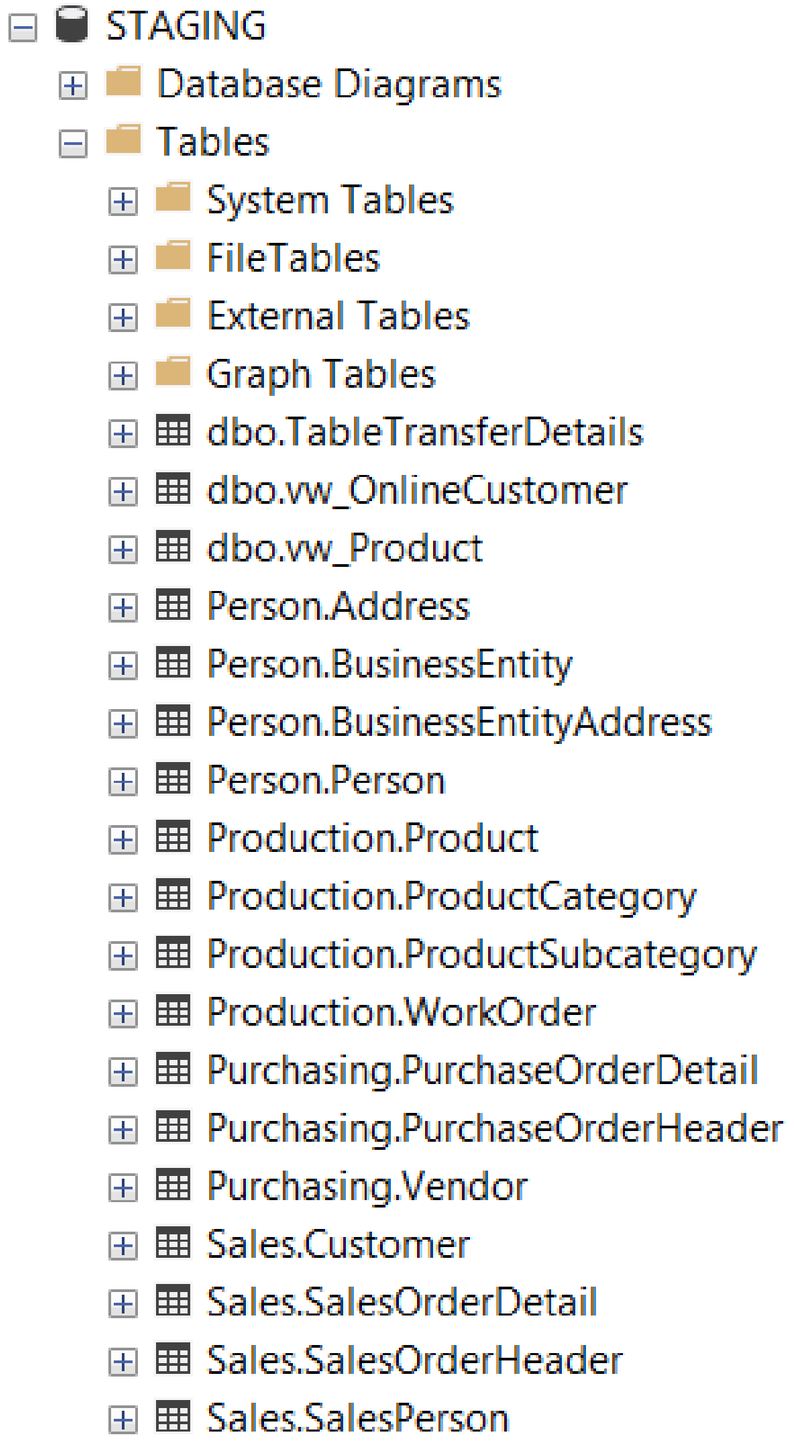
- ***Create Staging database to store the raw data extracted from AdventureWorks***

- ***Create only the tables and schema we will need to extract (those who are related to sales, Customers, inventory)***

```
CREATE DATABASE STAGING;
GO

USE STAGING;
GO

-- Creating schemas and tables

CREATE SCHEMA Person;
GO

-- Create table ADDRESS
CREATE TABLE [Person].[Address](
    [AddressID] [int] ,
    [AddressLine1] [nvarchar](60) NOT NULL,
    [AddressLine2] [nvarchar](60) NULL,
    [City] nvarchar(30) NOT NULL,
    [StateProvinceID] [int] NOT NULL,
    [PostalCode] nvarchar(15) NOT NULL,
    [SpatialLocation] VARBINARY(MAX) NULL,
    [rowguid] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
    [ModifiedDate] [datetime] NOT NULL,
 CONSTRAINT [PK_Address_AddressID] PRIMARY KEY CLUSTERED
(
    [AddressID] ASC
)
);
GO



 CREATE TABLE [Person].[Person](
```

- ***Those are the final tables for staging database***

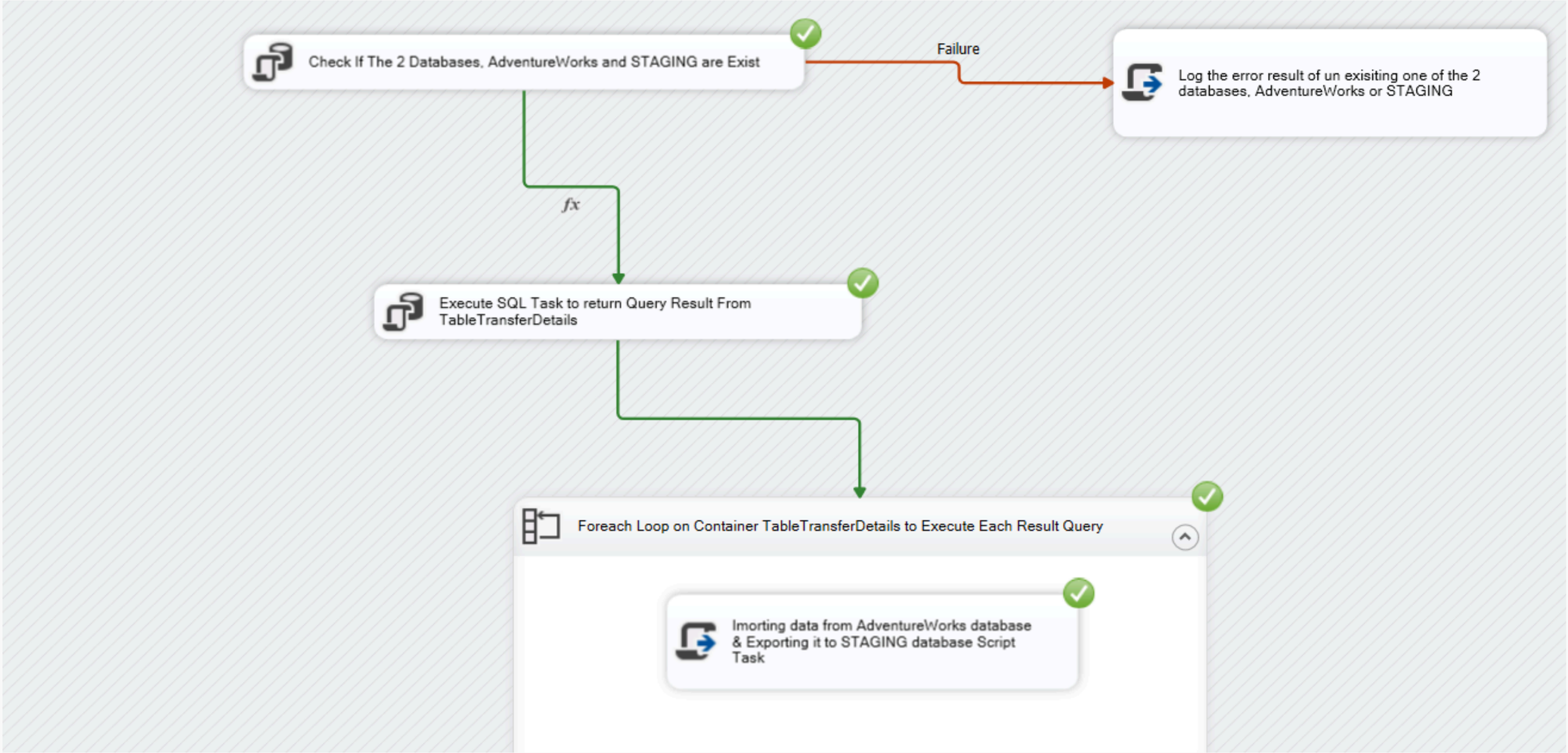
- ***There are other tables we create to help us later to increase performance and decrease time effort***



# *1st package: Extract-to-Staging*

***In this package:***

1. ***We Extract the tables we need from source (AW) to Staging database (STAGING)***
2. ***We add constraints and validataions test to ensure performance of dataflow***
3. ***We do the transfer using dynamic mapping by appling data flow programmatically using Script Task***



## 2. *We add constraints and validataions test to ensure performance of dataflow*:

- ***We check if the 2 databases are exixst***
  - ***on False return 0***
  - ***on True return 1***

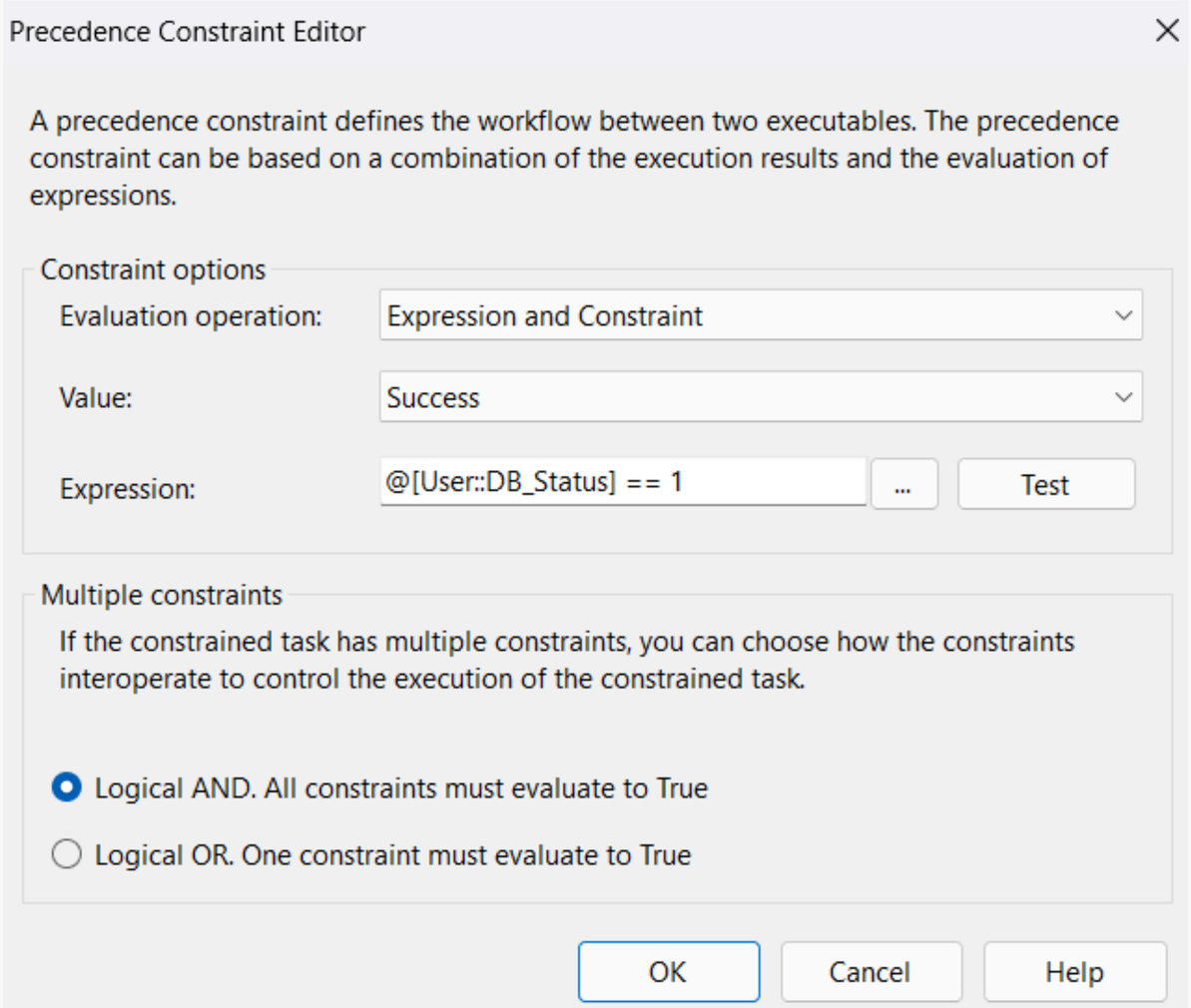
Check If 2 DBs Exi...-1V4QN0F\user (55))

```
IF EXISTS (SELECT * FROM sys.databases WHERE name = 'AdventureWorks')
    AND EXISTS (SELECT * FROM sys.databases WHERE name = 'STAGING')

  BEGIN
    SELECT 1 AS Status; -- Indicates both databases exist
    PRINT 'Both databases exist. Continuing with the process.';
  END

ELSE

  BEGIN
    SELECT 0 AS Status; -- Indicates one or both databases do not exist
    PRINT 'one or both databases do not exist. Wrute into the logging file.';
  END
```

# 2. We add constraints and validataions test to ensure performance of dataflow

***The status of databases are assigned to variable DB_Status:***

1. ***If the status is 1 then the 2 database exist and we can progress on to the dataflow***
2. ***If the status is 0 then one or both of the 2 database are missing and we will log it to the file***



## *3. We do the transfer using dynamic mapping by appling data flow programmatically using Script Task:*

Data Flow Task:

- Dynamic Mapping: Limited. You can use expressions or configurations, but it isn't flexible for handling frequent changes in column names or structures without manual adjustments.

- Manual Mapping: Direct and easy using "Source" and "Destination" components. Column mappings are set up manually in the designer, but any schema changes require manual remapping.

Script Task:

- Dynamic Mapping: Highly flexible. You can write custom C#/VB code to programmatically handle dynamic column names, allowing you to map columns based on metadata or external sources at runtime.

- Manual Mapping: Requires coding to explicitly map each column, making it more complex compared to Data Flow Task

# 3. We do the transfer using dynamic mapping by appling data flow programmatically using Script Task:

- We create TableTransferDetails that holds the connection to sourrce and destination, query on source table, destination table to put this query result and optional query on detination table

```
CREATE TABLE dbo.TableTransferDetails (
    SourceConnection NVARCHAR(255),
    DestinationConnection NVARCHAR(255),
    SourceQuery NVARCHAR(MAX),
    DestinationTable NVARCHAR(255),
    PreExecuteQuery NVARCHAR(MAX) -- Optional for truncating, etc.
);


INSERT INTO dbo.TableTransferDetails
(SourceConnection, DestinationConnection, SourceQuery, DestinationTable, PreExecuteQuery)
VALUES
(
 'Data Source=DESKTOP-1V4QN0F;Initial Catalog=AdventureWorks;Integrated Security=True',
 'Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGING;Integrated Security=True',
 'SELECT [BusinessEntityID],[PersonType],[NameStyle],[Title],[FirstName],[MiddleName],[LastName],[Suffix],[EmailPromotion],CAST(Addition
 '[STAGING].[Person].[Person]',
 'TRUNCATE TABLE [STAGING].[Person].[Person]');



INSERT INTO dbo.TableTransferDetails
(SourceConnection, DestinationConnection, SourceQuery, DestinationTable, PreExecuteQuery)
VALUES
(
 'Data Source=DESKTOP-1V4QN0F;Initial Catalog=AdventureWorks;Integrated Security=True',
 'Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGING;Integrated Security=True',
 'SELECT [AddressID],[AddressLine1],[AddressLine2],[City],[StateProvinceID],[PostalCode],CAST(SpatialLocation AS VARBINARY(MAX)) AS Spat
 '[STAGING].[Person].[Address]',
```

# *3. We do the transfer using dynamic mapping by appling data flow programmatically using Script Task:*

- ***We use a sql task to return this result to variable Var_System_Object of type Object and make it FullResultSet to hold all the result***

```
SELECT SourceConnection, DestinationConnection, SourceQuery, DestinationTable, PreExecuteQuery FROM dbo.TableTransferDetails;
```

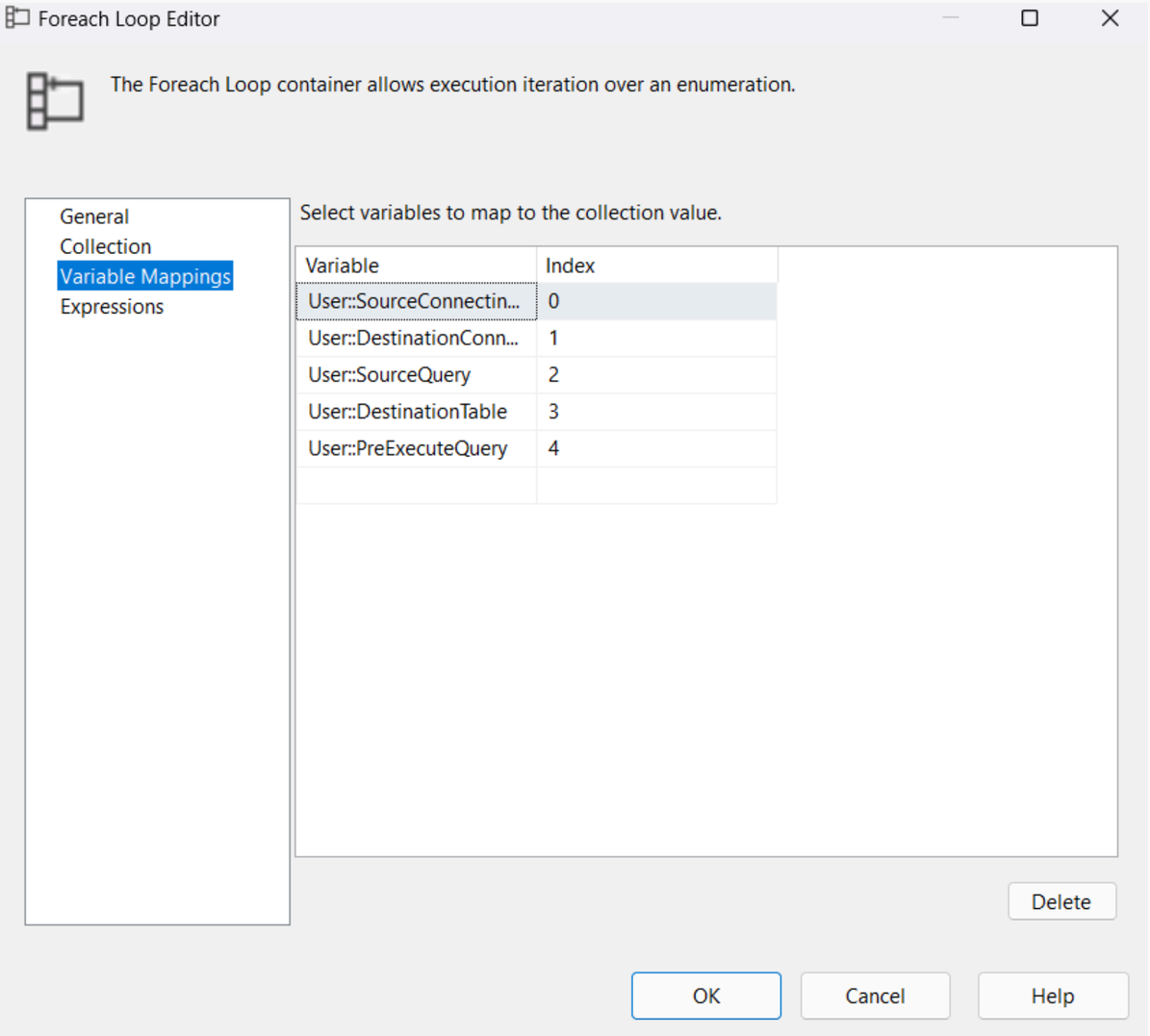
111 %

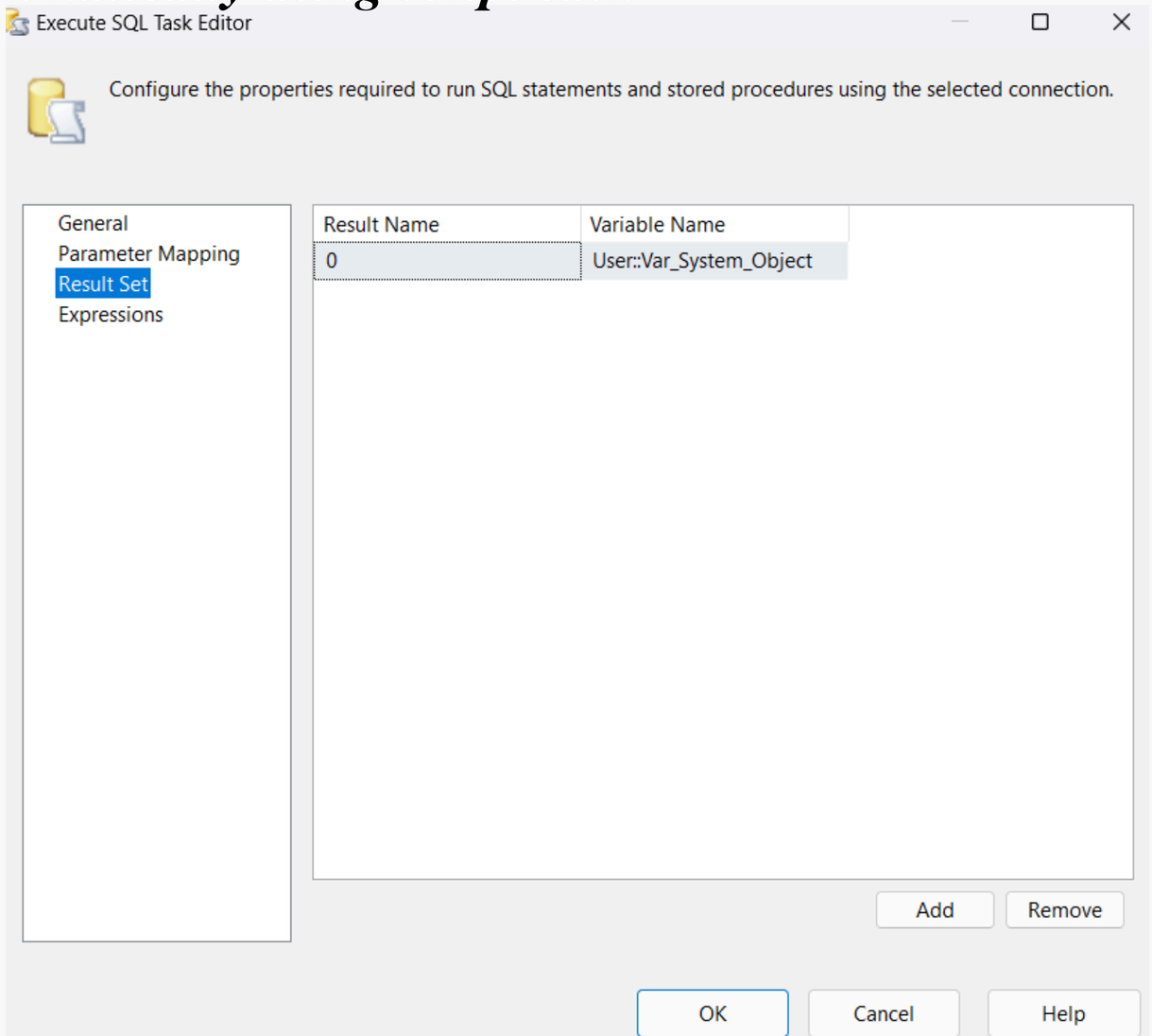
Results | Messages

| | SourceConnection | DestinationConnection | SourceQuery | DestinationTable | PreExecuteQue |
|---|---|---|---|---|---|
| 1 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT * FROM [AdventureWorks].[Sales].[Store] | [STAGING].[Sales].[Store] | TRUNCATE TA |
| 2 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT [CustomerID],[PersonID],[StoreID],[TerritoryID... | [STAGING].[Sales].[Customer] | TRUNCATE TA |
| 3 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT * FROM [AdventureWorks].[Person].[Business... | [STAGING].[Person].[BusinessEntityAddress] | TRUNCATE TA |
| 4 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT * FROM [AdventureWorks].[Person].[Business... | [STAGING].[Person].[BusinessEntity] | TRUNCATE TA |
| 5 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT [BusinessEntityID],[PersonType],[NameStyle],... | [STAGING].[Person].[Person] | TRUNCATE TA |
| 6 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT [AddressID],[AddressLine1],[AddressLine2],[C... | [STAGING].[Person].[Address] | TRUNCATE TA |
| 7 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT * FROM [AdventureWorks].[Production].[Prod... | [STAGING].[Production].[Product] | TRUNCATE TA |
| 8 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT * FROM [AdventureWorks].[Production].[Prod... | [STAGING].[Production].[ProductCategory] | TRUNCATE TA |
| 9 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT * FROM [AdventureWorks].[Production].[Prod... | [STAGING].[Production].[ProductSubCategory] | TRUNCATE TA |
| 10 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT [WorkOrderID],[ProductID],[OrderQty],[Scrapp... | [STAGING].[Production].[WorkOrder] | TRUNCATE TA |
| 11 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT [PurchaseOrderID],[PurchaseOrderDetailID],[... | [STAGING].[Purchasing].[PurchaseOrderDetail] | TRUNCATE TA |
| 12 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT [PurchaseOrderID],[RevisionNumber],[Status]... | [STAGING].[Purchasing].[PurchaseOrderHeader] | TRUNCATE TA |
| 13 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT * FROM [AdventureWorks].[Purchasing].[Ven... | [STAGING].[Purchasing].[Vendor] | TRUNCATE TA |
| 14 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT [SalesOrderID],[SalesOrderDetailID],[CarrierT... | [STAGING].[Sales].[SalesOrderDetail] | TRUNCATE TA |
| 15 | Data Source=DESKTOP-1V4QN0F;Initial Catalog=Adve... | Data Source=DESKTOP-1V4QN0F;Initial Catalog=STAGI... | SELECT [SalesOrderID],[RevisionNumber],[OrderDate... | [STAGING].[Sales].[SalesOrderHeader] | TRUNCATE TA |

# 3. We do the transfer using dynamic mapping by appling data flow programmatically using Script Task:

***On each iteration we assign one row of result set that is stored in Var_System_obeject to those 5 variables and do the data flow programmatically using script task***



```
public void Main()
{
    // SSIS Variables
    string sourceConnection = Dts.Variables["User::SourceConnectingString"].Value.ToString();
    string destinationConnection = Dts.Variables["User::DestinationConnectionString"].Value.ToString();
    string sourceQuery = Dts.Variables["User::SourceQuery"].Value.ToString();
    string destinationTable = Dts.Variables["User::DestinationTable"].Value.ToString();
    string preExecuteQuery = Dts.Variables["User::PreExecuteQuery"].Value.ToString();

    // Execute PreExecuteQuery if present (e.g., TRUNCATE TABLE)
    if (!string.IsNullOrEmpty(preExecuteQuery))
    {
        ExecuteNonQuery(destinationConnection, preExecuteQuery);
    }
    //MessageBox.Show("The Table truncated Successfully");

    // Load data from source and insert into destination
    DataTable sourceData = LoadDataFromSource(sourceConnection, sourceQuery);
    //MessageBox.Show("Data has been Imported Successfuly");
    BulkInsertIntoDestination(destinationConnection, destinationTable, sourceData);
    MessageBox.Show($"Data has been Loaded successfuly to table {destinationTable}");
    Dts.TaskResult = (int)ScriptResults.Success;
}
```

(2)

```
private DataTable LoadDataFromSource(string connectionString, string query)
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
        DataTable data = new DataTable();
        adapter.Fill(data);
        return data;
    }
}
```

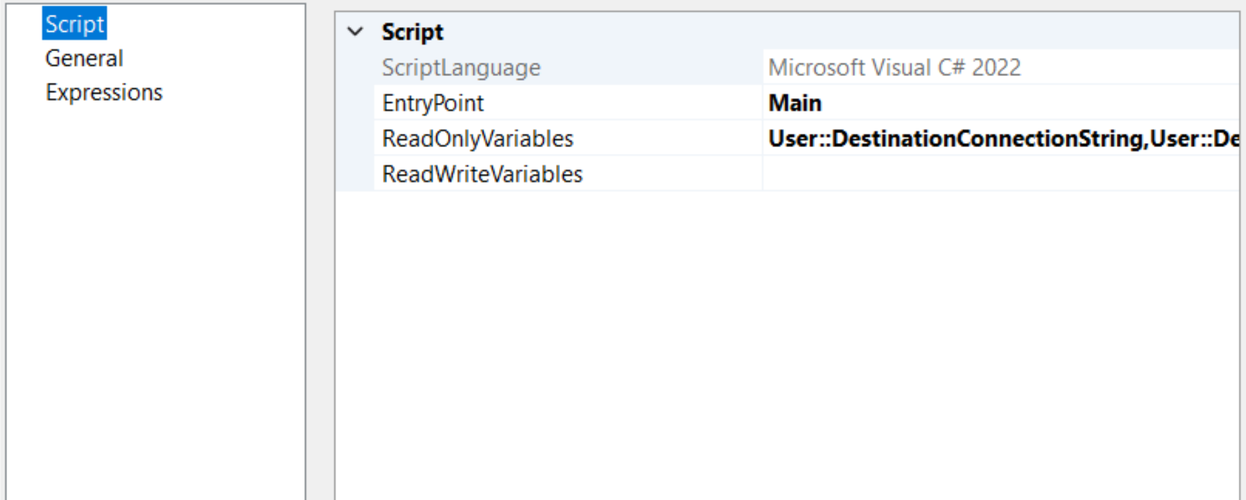
(3)

```
1 reference
private void BulkInsertIntoDestination(string connectionString, string tableName, DataTable da
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        SqlBulkCopy bulkCopy = new SqlBulkCopy(conn);
        bulkCopy.DestinationTableName = tableName;
        conn.Open();
        bulkCopy.WriteToServer(data);
        conn.Close();
    }
}
```

```
1 reference
private void ExecuteNonQuery(string connectionString, string query)
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        SqlCommand cmd = new SqlCommand(query, conn);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
    }
}
```

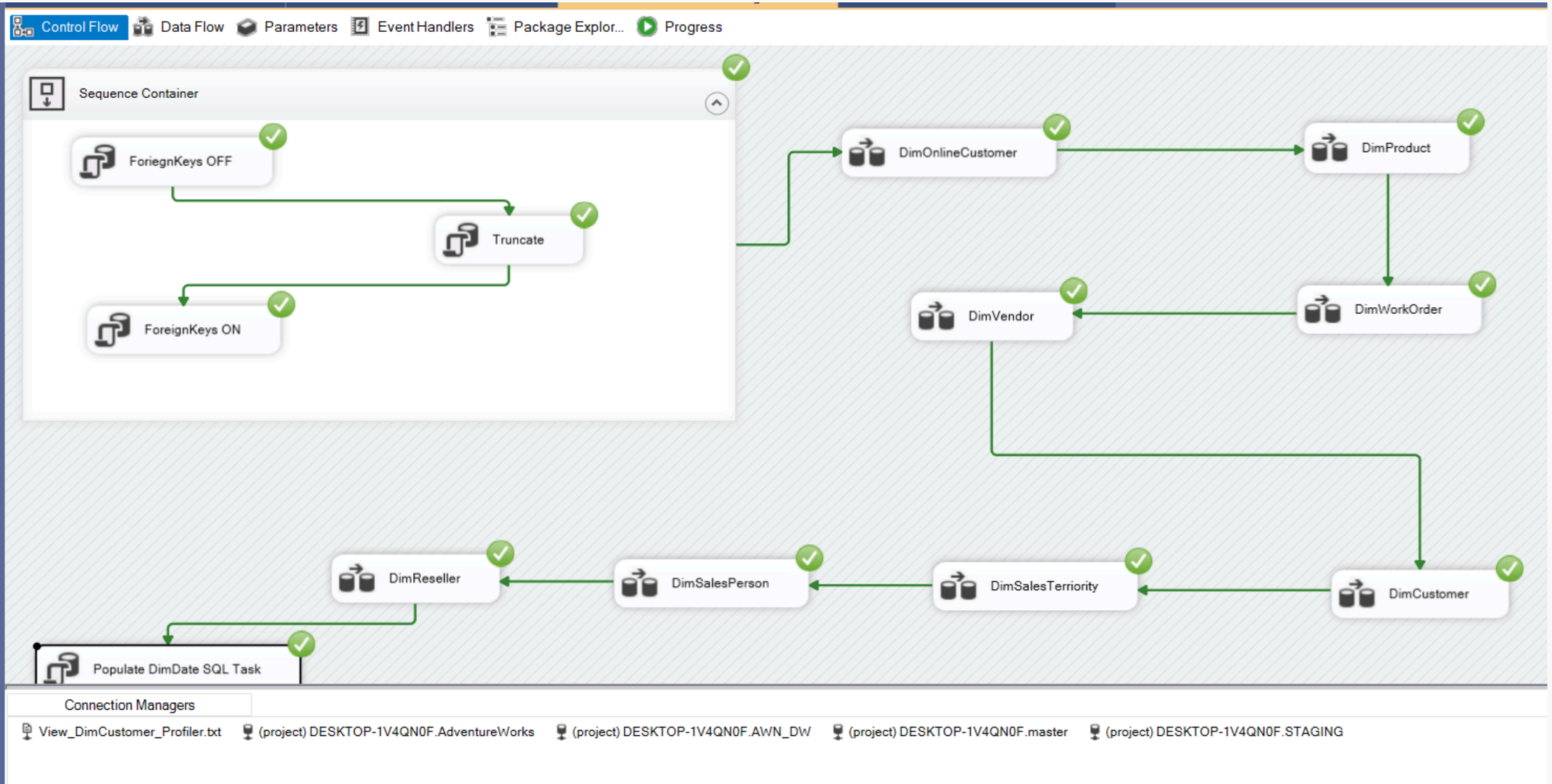
Access Microsoft Visual Studio Tools for Applications (VSTA) to write scripts using the Visual Basic 2022 or Visual C# 2022, and configure the task's properties.

(1)



# *2nd Package: Load-Views-Dim:*

- *Disable foriegnKeys , Truncate and Enable it again (Data WareHouse)*
- *Do flow task to each Dimension table in DW from staging tables or views*
- *Build and populate our DimDate*



# *Building Data WareHouse (1)Dimension tables and (2)Fact tables*

*(1)*

```
Create Database AWN_DW
Use AWN_DW
GO

CREATE TABLE [dbo].[DimReseller] (
    ResellerSurrogateKey INT IDENTITY(1,1) ,    -- Surrogate key
    ResellerID INT UNIQUE NOT NULL,
    SalesPersonID INT,
    CustomerKey Int,
    ResellerName NVARCHAR(100),
    ModifiedDate datetime
 CONSTRAINT [PK_DimReseller_ResellerKey] PRIMARY KEY CLUSTERED
(
    ResellerSurrogateKey ASC,
    ResellerID ASC
)
);
GO

CREATE TABLE [dbo].[DimCustomer](
    [CustomerSurrogateKey] [int] identity(1,1),
    [CustomerID] [int]  UNIQUE NOT NULL,
    [PersonID] [int] NULL,
    [ResellerID] [int] NULL,
    [TerritoryID] [int] NULL,
    [rowguid] [uniqueidentifier]   NOT NULL,
    [ModifiedDate] [datetime] NOT NULL,
 CONSTRAINT [PK_DimCustomer_CustomerKey] PRIMARY KEY CLUSTERED
(
    [CustomerSurrogateKey] ASC,
    [CustomerID] ASC
)
);
GO

CREATE TABLE [dbo].[DimOnlineCustomer](
    [OnlineCustomerSurrogateKey] [int] IDENTITY(1,1) NOT NULL,
    [OnlineCustomerID] INT UNIQUE NOT NULL,
```

*(2)*

```
CREATE TABLE [dbo].[FactInternetSales](
    [ProductKey] [int] NOT NULL,
    [OrderDateKey_Internet] [int],
    OnlineCustomerID int,
    [CustomerID] [int] ,
    [SalesTerritoryKey] [int] ,
    [SalesOrderNumber] [nvarchar](20) NOT NULL, --SALESORDERID
    [SalesOrderLineNumber] [tinyint] NOT NULL,
    LineTotal money,
    [OrderQuantity] [smallint] NOT NULL,
    [UnitPrice] [money] NOT NULL,
    UnitPriceDiscount money,
    SubTotal money,
    TaxAmt money,
    Freight money,
    TotalDue money,
    [OrderDate] [datetime] NULL,
    [DueDate] [datetime] NULL,
    [ShipDate] [datetime] NULL,
    ModifiedDate datetime
 CONSTRAINT [PK_FactInternetSales_SalesOrderNumber_SalesOrderLineNumber] PRIMARY KEY CLUSTERED
(
    [SalesOrderNumber] ASC,
    [SalesOrderLineNumber] ASC
)
);
GO

CREATE TABLE [dbo].[FactPurchaseOrder] (
    PurchaseOrderID INT NOT NULL,
    PurchaseOrderDetailID INT NOT NULL,
```

# Building DimTable

(1)

```sql
USE AWN_DW


-- Create DimDate dimension table
CREATE TABLE [dbo].[DimDate] (
    [DateKey]               [int] NOT NULL,
    [FullDateAlternateKey] [date] NULL,
    [DayNumberOfWeek]       [tinyint] NULL,
    [DayNumberOfMonth]      [tinyint] NULL,
    [DayNumberOfYear]       [smallint] NULL,
    [WeekNumberOfYear]      [tinyint] NULL,
    [MonthNumberOfYear]     [tinyint] NULL,
    [CalendarQuarter]       [tinyint] NULL,
    [CalendarYear]          [smallint] NULL,
    [CalendarSemester]      [tinyint] NULL,
    [FiscalQuarter]         [tinyint] NULL,
    [FiscalYear]            [smallint] NULL,
    [FiscalSemester]        [tinyint] NULL,
    CONSTRAINT [PK_DimDate_DateKey] PRIMARY KEY CLUSTERED (
        [DateKey] ASC
    ) WITH (
        PAD_INDEX = OFF,
        STATISTICS_NORECOMPUTE = OFF,
        IGNORE_DUP_KEY = OFF,
        ALLOW_ROW_LOCKS = ON,
        ALLOW_PAGE_LOCKS = ON
    ) ON [PRIMARY]
) ON [PRIMARY];
GO
```

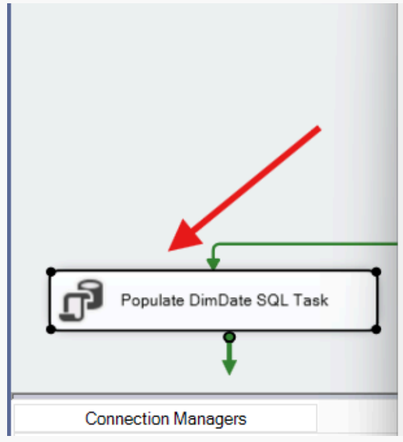
(2)

```sql
CREATE PROCEDURE [dbo].[Refresh_DimDate]
AS
BEGIN
    DECLARE @startdate date = '2001-01-01', @enddate date = '2014-12-31';

    IF @startdate IS NULL
    BEGIN
        SELECT TOP 1 @startdate = FullDateAlternateKey
        FROM DimDate
        ORDER BY DateKey ASC;
    END

    DECLARE @datelist TABLE (FullDate date);

    -- Loop through date range
    WHILE @startdate <= @enddate
    BEGIN
        INSERT INTO @datelist (FullDate)
        SELECT @startdate;

        SET @startdate = DATEADD(dd, 1, @startdate);
    END

    -- Insert new records into DimDate
    INSERT INTO dbo.DimDate (
        DateKey,
        FullDateAlternateKey,
        DayNumberOfWeek,
        DayNumberOfMonth,
        DayNumberOfYear,
        WeekNumberOfYear,
        MonthNumberOfYear,
        CalendarQuarter,
        CalendarYear,
        CalendarSemester,
```

(3)

```sql
    SELECT
        CONVERT(int, CONVERT(varchar, dl.FullDate, 112)) AS DateKey,
        dl.FullDate,
        DATEPART(dw, dl.FullDate) AS DayNumberOfWeek,
        DATEPART(d, dl.FullDate) AS DayNumberOfMonth,
        DATEPART(dy, dl.FullDate) AS DayNumberOfYear,
        DATEPART(wk, dl.FullDate) AS WeekNumberOfYear,
        MONTH(dl.FullDate) AS MonthNumberOfYear,
        DATEPART(qq, dl.FullDate) AS CalendarQuarter,
        YEAR(dl.FullDate) AS CalendarYear,
        CASE DATEPART(qq, dl.FullDate)
            WHEN 1 THEN 1
            WHEN 2 THEN 1
            WHEN 3 THEN 2
            WHEN 4 THEN 2
        END AS CalendarSemester,
        CASE DATEPART(qq, dl.FullDate)
            WHEN 1 THEN 3
            WHEN 2 THEN 4
            WHEN 3 THEN 1
            WHEN 4 THEN 2
        END AS FiscalQuarter,
        CASE DATEPART(qq, dl.FullDate)
            WHEN 1 THEN YEAR(dl.FullDate)
            WHEN 2 THEN YEAR(dl.FullDate)
            WHEN 3 THEN YEAR(dl.FullDate) + 1
            WHEN 4 THEN YEAR(dl.FullDate) + 1
        END AS FiscalYear,
        CASE DATEPART(qq, dl.FullDate)
            WHEN 1 THEN 2
            WHEN 2 THEN 2
            WHEN 3 THEN 1
            WHEN 4 THEN 1
        END AS FiscalSemester
    FROM @datelist dl
    LEFT JOIN DimDate dd ON dl.FullDate = dd.FullDateAlternateKey
    WHERE dd.FullDateAlternateKey IS NULL;
END;
GO
```

(4)




| | |
|---|---|
| ConnectionType | **OLE DB** |
| Connection | **DESKTOP-1V4QN0F.AWN_DW** |
| SQLSourceType | **Direct input** |
| SQLStatement | **EXEC [dbo].[Refresh_DimDate]** |
| IsQueryStoredProcedure | False |
| BypassPrepare | **False** |

**Name**
Specifies the name of the task.

Browse... | Build Query... | Parse Query

# Building Constraints and ForeignKeys

- Between Facts and DimTables
- Between Facts and DimDate
- Between Dims and Dims

(1)

```
USE AWN_DW

-- FactInternetSales related to DimCustomer, DimProduct, DimSalesTerritory
ALTER TABLE [dbo].[FactInternetSales]  WITH CHECK ADD  CONSTRAINT [FK_FactInternetSales_DimOnlineCustomer] FOREIGN KEY([OnlineCustomerID])
REFERENCES [dbo].[DimOnlineCustomer] ([OnlineCustomerID])
GO
ALTER TABLE [dbo].[FactInternetSales] CHECK CONSTRAINT [FK_FactInternetSales_DimOnlineCustomer]
GO


ALTER TABLE [dbo].[FactInternetSales]  WITH CHECK ADD  CONSTRAINT [FK_FactInternetSales_DimCustomer] FOREIGN KEY([CustomerID])
REFERENCES [dbo].[DimCustomer] ([CustomerID])
GO
ALTER TABLE [dbo].[FactInternetSales] CHECK CONSTRAINT [FK_FactInternetSales_DimCustomer]
GO


ALTER TABLE [dbo].[FactInternetSales]  WITH CHECK ADD  CONSTRAINT [FK_FactInternetSales_DimProduct] FOREIGN KEY([ProductKey])
REFERENCES [dbo].[DimProduct] ([ProductID])
GO
ALTER TABLE [dbo].[FactInternetSales] CHECK CONSTRAINT [FK_FactInternetSales_DimProduct]
GO


ALTER TABLE [dbo].[FactInternetSales]  WITH CHECK ADD  CONSTRAINT [FK_FactInternetSales_DimSalesTerritory] FOREIGN KEY([SalesTerritoryKey])
REFERENCES [dbo].[DimSalesTerritory] ([SalesTerritoryID])
GO
ALTER TABLE [dbo].[FactInternetSales] CHECK CONSTRAINT [FK_FactInternetSales_DimSalesTerritory]
GO
```

(2)

```
--Dimdate linked to facts
-- Check if the foreign key constraint on FactInternetSales exists before adding
IF NOT EXISTS (
    SELECT 1
    FROM sys.foreign_keys
    WHERE name = 'FK_FactInternetSales_DimDate'
)
BEGIN
    -- Add foreign key constraint on FactInternetSales
    ALTER TABLE [dbo].[FactInternetSales] WITH CHECK ADD CONSTRAINT [FK_FactInternetSales_DimDate]
    FOREIGN KEY (OrderDateKey_Internet) REFERENCES [dbo].[DimDate] (DateKey);

    -- Check the constraint to make sure it is enforced
    ALTER TABLE [dbo].[FactInternetSales] CHECK CONSTRAINT [FK_FactInternetSales_DimDate];
END;
GO
```

(3)

```
tResellerSales related to DimProduct, DimSalesTerritory, DimReseller
ALTER TABLE [dbo].[FactResellerSales]  WITH CHECK ADD  CONSTRAINT [FK_FactResellerSales_DimCustomer] FOREIGN KEY([CustomerID])
REFERENCES [dbo].[DimCustomer] ([CustomerID])
GO
ALTER TABLE [dbo].[FactInternetSales] CHECK CONSTRAINT [FK_FactInternetSales_DimCustomer]
GO


ALTER TABLE [dbo].[FactResellerSales]  WITH CHECK ADD  CONSTRAINT [FK_FactResellerSales_DimProduct] FOREIGN KEY([ProductKey])
REFERENCES [dbo].[DimProduct] ([ProductID])
GO
ALTER TABLE [dbo].[FactResellerSales] CHECK CONSTRAINT [FK_FactResellerSales_DimProduct]
GO


ALTER TABLE [dbo].[FactResellerSales]  WITH CHECK ADD  CONSTRAINT [FK_FactResellerSales_DimSalesTerritory] FOREIGN KEY([SalesTerritoryKey])
REFERENCES [dbo].[DimSalesTerritory] ([SalesTerritoryID])
GO
ALTER TABLE [dbo].[FactResellerSales] CHECK CONSTRAINT [FK_FactResellerSales_DimSalesTerritory]
GO


ALTER TABLE [dbo].[FactResellerSales]  WITH CHECK ADD  CONSTRAINT [FK_reseller] FOREIGN KEY([ResellerKey])
REFERENCES [dbo].[DimReseller] (ResellerID)
GO
ALTER TABLE [dbo].[FactResellerSales] CHECK CONSTRAINT [FK_reseller]
GO
```

(4)

```
-- FactPurchaseOrder related to DimVendor, DimShipMethod
ALTER TABLE [dbo].[FactPurchaseOrder] WITH CHECK ADD CONSTRAINT [FK_FactPurchaseOrder_DimVendor] FOREIGN KEY(VendorID)
REFERENCES [dbo].[DimVendor] ([VendorID])
GO
ALTER TABLE [dbo].[FactPurchaseOrder] CHECK CONSTRAINT [FK_FactPurchaseOrder_DimVendor]


ALTER TABLE [dbo].[FactPurchaseOrder] WITH CHECK ADD CONSTRAINT [FK_FactPurchaseOrder_DimProduct] FOREIGN KEY(ProductID)
REFERENCES [dbo].[DimProduct] ([ProductID])
GO
ALTER TABLE [dbo].[FactPurchaseOrder] CHECK CONSTRAINT [FK_FactPurchaseOrder_DimProduct]
GO


-- DimReseller related to SalesPerson
ALTER TABLE [dbo].[DimReseller] ADD CONSTRAINT [FK_DimReseller_DimSalesPerson] FOREIGN KEY([SalesPersonID])
REFERENCES [dbo].[DimSalesPerson] ([SalesPersonID])
GO
ALTER TABLE [dbo].[DimReseller] CHECK CONSTRAINT [FK_DimReseller_DimSalesPerson]


--- DimProduct related to WorkOrder
ALTER TABLE [dbo].[DimWorkOrder] ADD CONSTRAINT [FK_DimWorkOrder_DimProduct] FOREIGN KEY([ProductID])
REFERENCES [dbo].[DimProduct] ([ProductID])
GO

ALTER TABLE [dbo].[DimWorkOrder] CHECK CONSTRAINT [FK_DimWorkOrder_DimProduct]
```

## *Do we now just do data flow ???*

- *17 tables are many and maybe some of them has few rows or just to link some tables*
- *Data WareHouse is OLAP*
- *We have to do DeNormalization*

***In Data WareHouse Performance is more important that storage***

```sql
SELECT *
INTO [dbo].[vw_Product]
FROM (
        SELECT
            p.ProductID,
            p.ProductNumber,
            p.[Name],
            sc.[Name] AS ProductSubcategoryName,
            c.[Name] AS ProductCategoryName,
            p.MakeFlag,
            p.FinishedGoodsFlag,
            p.Color,
            p.SafetyStockLevel,
            p.ReorderPoint,
            p.StandardCost,
            p.ListPrice,
            p.Size,
            p.SizeUnitMeasureCode,
            p.[Weight],
            p.WeightUnitMeasureCode,
            p.DaysToManufacture,
            p.ProductLine,
            p.Class,
            p.Style,
            p.SellStartDate,
            p.SellEndDate,
            p.DiscontinuedDate,
            p.rowguid,
            p.ModifiedDate,
            ROW_NUMBER() OVER (PARTITION BY p.ProductID ORDER BY p.ProductID ASC) AS row_num
        FROM
            Production.Product AS p
        LEFT JOIN
            Production.ProductSubCategory AS sc ON p.ProductSubcategoryID = sc.ProductSubcategoryID
        LEFT JOIN
            Production.ProductCategory AS c ON sc.ProductCategoryID = c.ProductCategoryID
    )AS RankedSales
WHERE row_num = 1 and ProductID IS NOT NULL
order by ProductID;
```

```sql
-- View for FactOnlineSales
CREATE VIEW [dbo].[FactOnlineSales] AS
SELECT
    sod.SalesOrderID,
    sod.SalesOrderDetailID,
    ROW_NUMBER() OVER (PARTITION BY sod.SalesOrderID ORDER BY soh.ModifiedDate) AS saleLineNumber,
    CAST(soh.OrderDate AS DATE) AS OrderDate,
    CAST(soh.DueDate AS DATE) AS DueDate,
    CAST(soh.ShipDate AS DATE) AS ShipDate,
    sod.CarrierTrackingNumber,
    sod.OrderQty,
    sod.ProductID,
    sod.UnitPrice,
    sod.UnitPriceDiscount,
    soh.CustomerID,
    c.PersonID,
    soh.TerritoryID,
    soh.SubTotal,
    soh.TaxAmt,
    soh.Freight,
    soh.ModifiedDate
FROM
    Sales.SalesOrderHeader AS soh
LEFT JOIN
    Sales.SalesOrderDetail AS sod ON soh.SalesOrderID = sod.SalesOrderID
LEFT JOIN
    Production.[Product] AS p ON sod.ProductID = p.ProductID
LEFT JOIN
    Sales.Customer AS c ON soh.CustomerID = c.CustomerID
WHERE
    c.PersonID IS NOT NULL   -- Include only online sales
GO
```

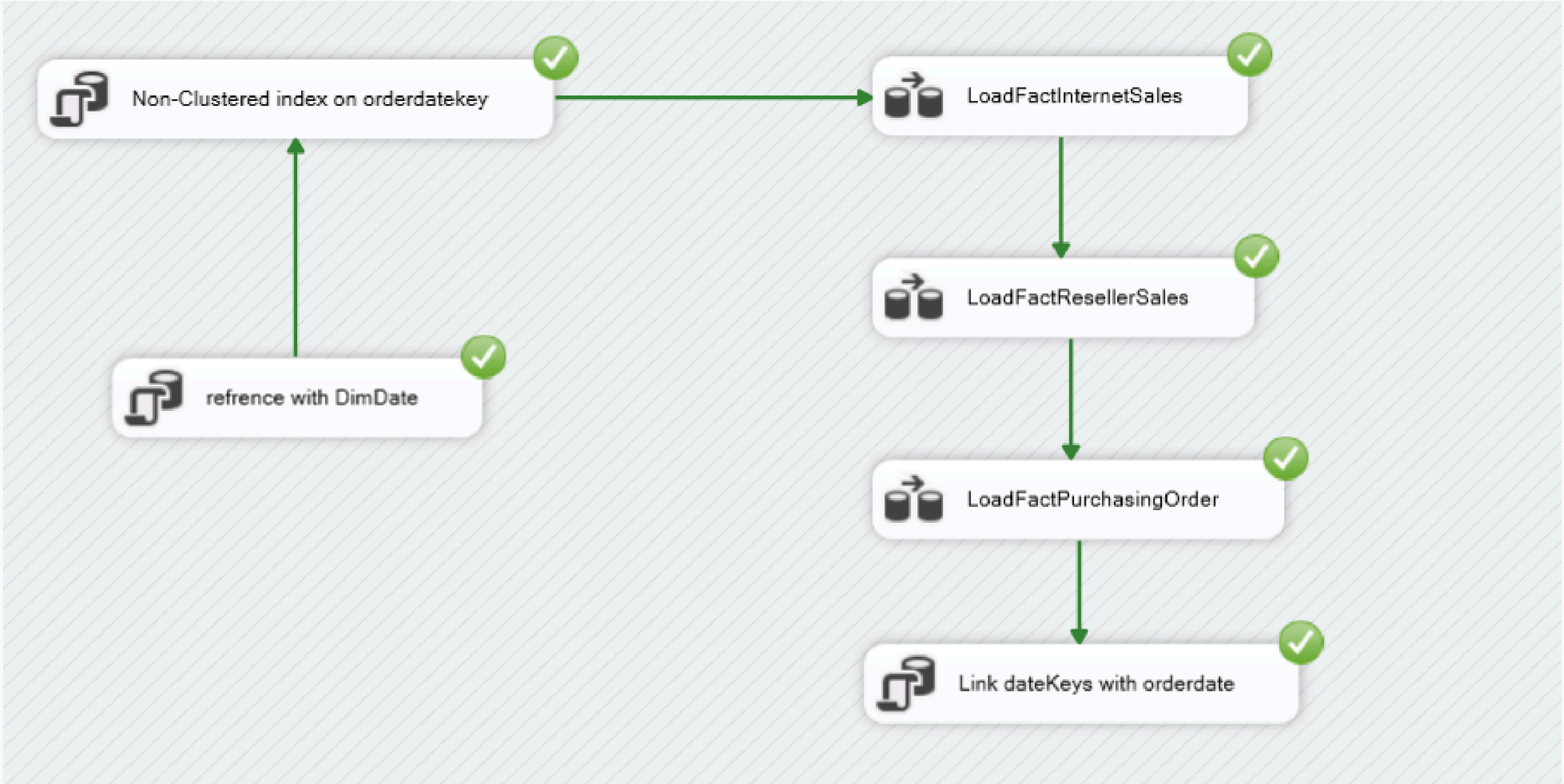
- ***Fact tables need many joins: so to increase the Performance we do it using Views ( Temp tables) instead of tables***

```sql
SELECT *
INTO vw_OnlineCustomer -- Name of the new table
FROM (
    SELECT
        c.PersonID,
        c.CustomerID,
        p.NameStyle,
        p.Title,
        p.FirstName,
        p.MiddleName,
        p.LastName,
        p.EmailPromotion,
        p.Suffix,
        p.AdditionalContactInfo,
        p.Demographics,
        p.rowguid,
        p.ModifiedDate AS PersonModifiedDate,
        a.AddressLine1,
        a.AddressLine2,
        a.City,
        a.ModifiedDate AS AddressModifiedDate,
        ROW_NUMBER() OVER (PARTITION BY c.PersonID ORDER BY c.PersonID ASC) AS row_num
    FROM Sales.Customer AS c
    LEFT JOIN
        Person.Person AS p ON c.PersonID = p.BusinessEntityID
    LEFT JOIN
        Person.BusinessEntityAddress AS bea ON p.BusinessEntityID = bea.BusinessEntityID
    LEFT JOIN
        Person.Address AS a ON bea.AddressID = a.AddressID
) AS RankedSales
WHERE row_num = 1 and PersonID IS NOT NULL
order by PersonID;
```

- ***We create new invented table for Readability and Clarity***

# *3rd package: Load Fact Tables*

- ***Refrence Fact tables with DimDate***
- ***Make Clustered Index on these keys***
- ***Load Fact Tables and Update it with DateKey***



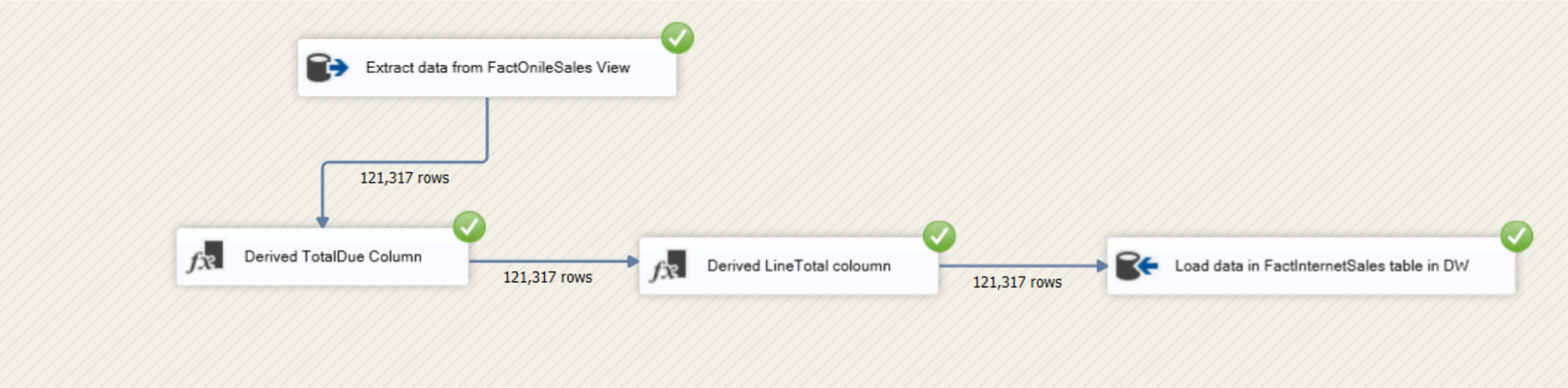
# *Make foreignKeys if not Exist*

```sql
--Dimdate linked to facts
-- Check if the foreign key constraint on FactInternetSales exists before adding
IF NOT EXISTS (
    SELECT 1
    FROM sys.foreign_keys
    WHERE name = 'FK_FactInternetSales_DimDate'
)
BEGIN
    -- Add foreign key constraint on FactInternetSales
    ALTER TABLE [dbo].[FactInternetSales] WITH CHECK ADD CONSTRAINT [FK_FactInternetSales_DimDate]
    FOREIGN KEY (OrderDateKey_Internet) REFERENCES [dbo].[DimDate] (DateKey);

    -- Check the constraint to make sure it is enforced
    ALTER TABLE [dbo].[FactInternetSales] CHECK CONSTRAINT [FK_FactInternetSales_DimDate];
END;
GO

-- Check if the foreign key constraint on FactResellerSales exists before adding
IF NOT EXISTS (
    SELECT 1
    FROM sys.foreign_keys
    WHERE name = 'FK_FactResellerSales_DimDate'
)
BEGIN
    -- Add foreign key constraint on FactResellerSales
    ALTER TABLE [dbo].[FactResellerSales] WITH CHECK ADD CONSTRAINT [FK_FactResellerSales_DimDate]
    FOREIGN KEY (OrderDateKey_Reseller) REFERENCES [dbo].[DimDate] (DateKey);

    -- Check the constraint to make sure it is enforced
    ALTER TABLE [dbo].[FactResellerSales] CHECK CONSTRAINT [FK_FactResellerSales_DimDate];
END;
GO
```

# *Make Clustered Index if not exist*

```sql
--- clusteredIndex on DimDate
-- Check if the index on FactInternetSales for OrderDateKey_Internet exists before creating
IF NOT EXISTS (
    SELECT 1
    FROM sys.indexes
    WHERE name = 'IX_FactInternetSales_OrderDateKey'
    AND object_id = OBJECT_ID('dbo.FactInternetSales')
)
BEGIN
    -- Create index on FactInternetSales for OrderDateKey_Internet
    CREATE NONCLUSTERED INDEX [IX_FactInternetSales_OrderDateKey] ON [dbo].[FactInternetSales] (
        [OrderDateKey_Internet] ASC
    );
END;
GO

-- Check if the index on FactResellerSales for OrderDateKey_Reseller exists before creating
IF NOT EXISTS (
    SELECT 1
    FROM sys.indexes
    WHERE name = 'IX_FactResellerSales_OrderDateKey'
    AND object_id = OBJECT_ID('dbo.FactResellerSales')
)
BEGIN
    -- Create index on FactResellerSales for OrderDateKey_Reseller
    CREATE NONCLUSTERED INDEX [IX_FactResellerSales_OrderDateKey] ON [dbo].[FactResellerSales] (
        [OrderDateKey_Reseller] ASC
    );
END;
GO
```

# *Extract data from tables and views of Staging database to Fact Tables in DW*




Derived Column Transformation Editor

Specify the expressions used to create new column values, and indicate whether the values update existing columns or popu
new columns.

- Variables and Parameters
- Columns

- Mathematical Functions
- String Functions
- Date/Time Functions
- NULL Functions
- Type Casts
- Operators

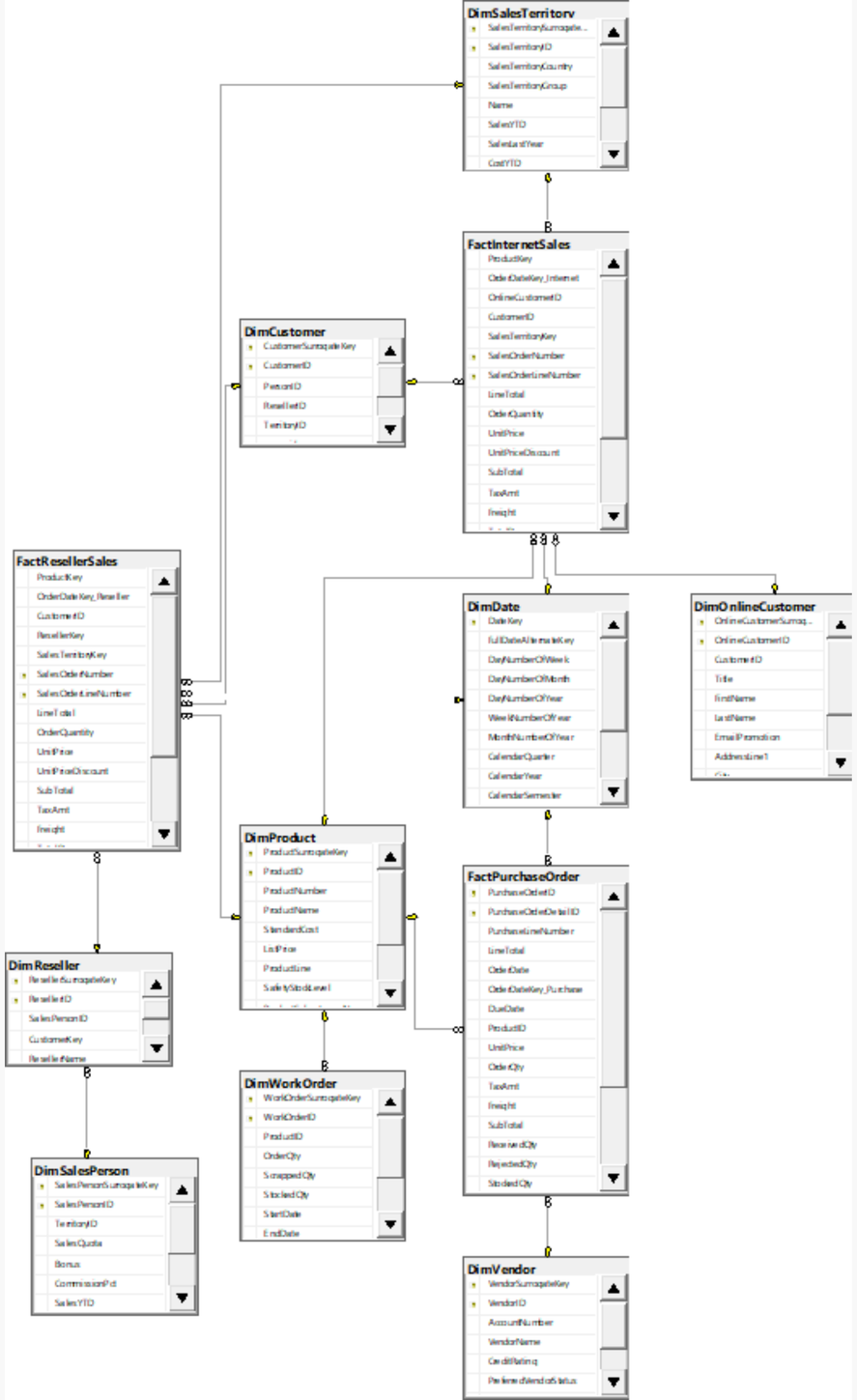
Description:

| Derived Column Name | Derived Column | Expression | Data Type |
|---|---|---|---|
| LineTotal | <add as new column> | (DT_DECIMAL,2)REPLACENULL(OrderQty * UnitPrice,(0.00)) | decimal [D |

# *We update DateKey coloumn in all fact tables*

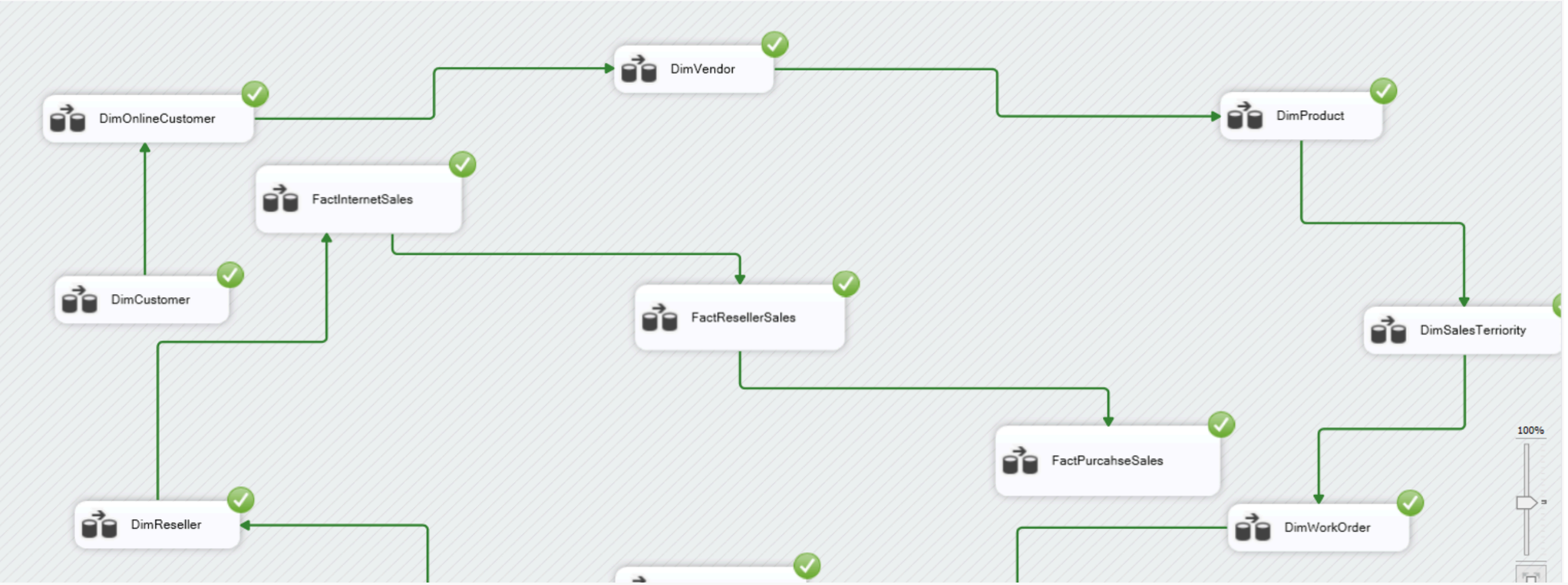
```
---update fact with datekey
-- Update the OrderDateKey_Internet in FactInternetSales by looking up DimDate
UPDATE fis
SET fis.OrderDateKey_Internet = dd.DateKey
FROM [dbo].[FactInternetSales] fis
JOIN [dbo].[DimDate] dd
  ON fis.OrderDate = dd.FullDateAlternateKey;

-- Update the ShipDateKey in FactInternetSales by looking up DimDate
UPDATE frs
SET frs.OrderDateKey_Reseller = dd.DateKey
FROM [dbo].[FactResellerSales] frs
JOIN [dbo].[DimDate] dd
  ON frs.OrderDate = dd.FullDateAlternateKey;

UPDATE fps
SET fps.OrderDateKey_Purchase = dd.DateKey
FROM [dbo].[FactPurchaseOrder] fps
JOIN [dbo].[DimDate] dd
ON fps.OrderDate = dd.FullDateAlternateKey;
```

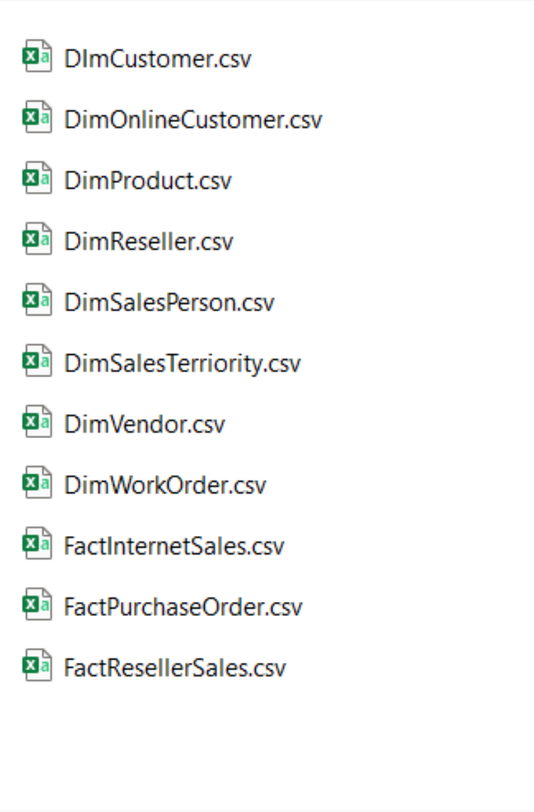
# *SnowFlake Schema*



# *4th Package: Export-Dims-files*

- ***Exporting the dataWarehouse tables to CSV files to be ready for future Analysis or ML models***



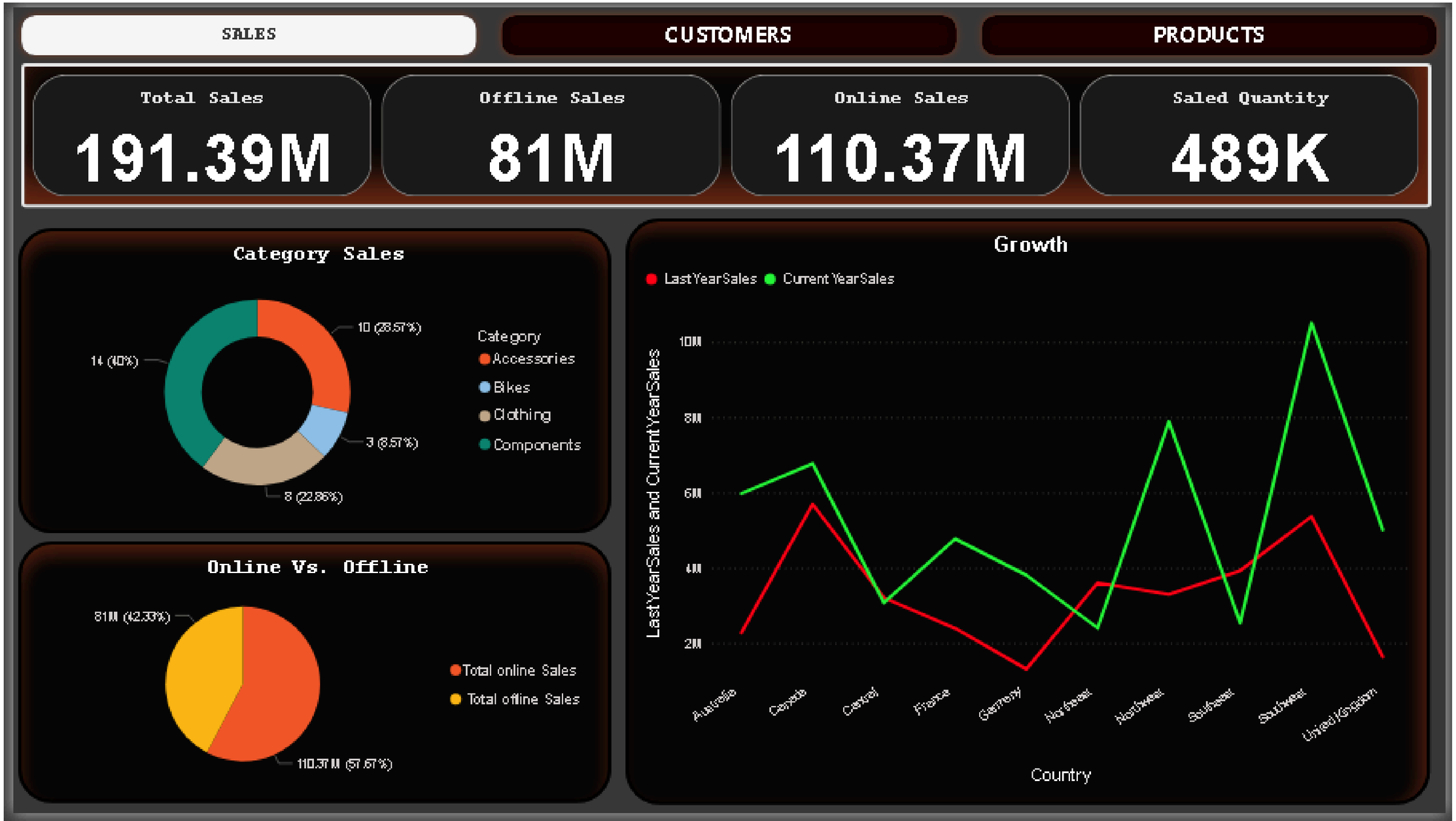



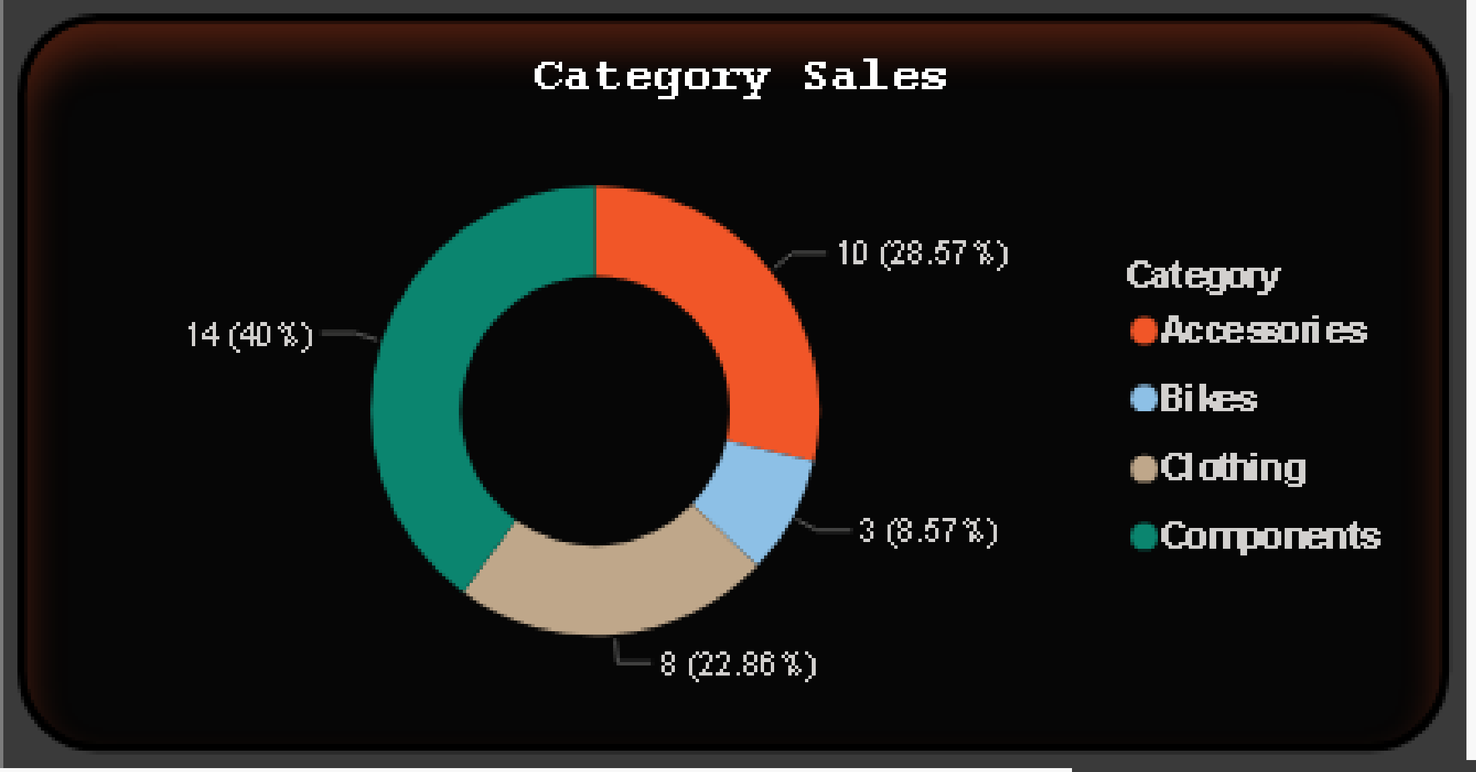
# *Future Enhancement*

- ***Make pipeline Automated***
- ***Implement CDC or track change to capture Changes happens to source***
- ***Implement SCD to capture Changes happens to Dimension***
- ***Ensure data Quality***

# *Data Analysis Using Power BI*

*We made 3 Dashboards that we toggle between them using Dynamic buttons , Each button has the name specified to Certain analysis*

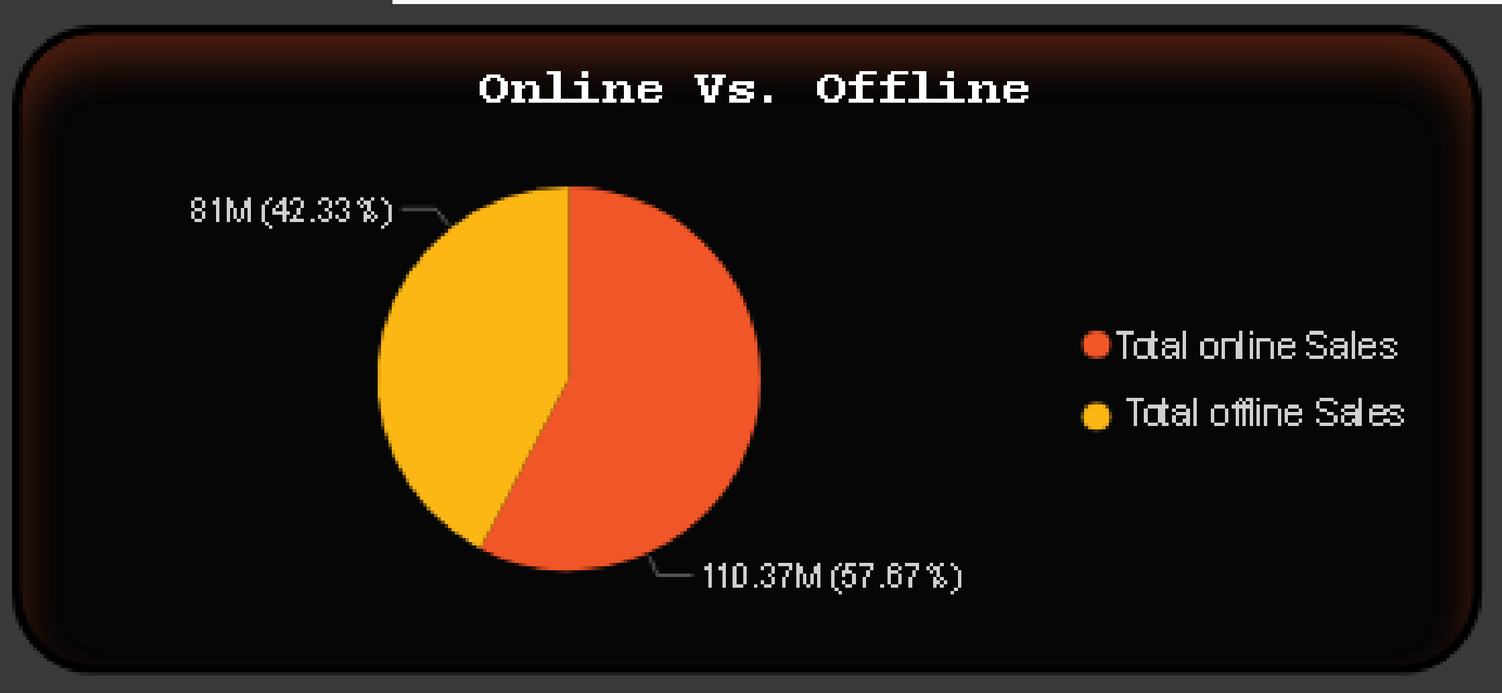




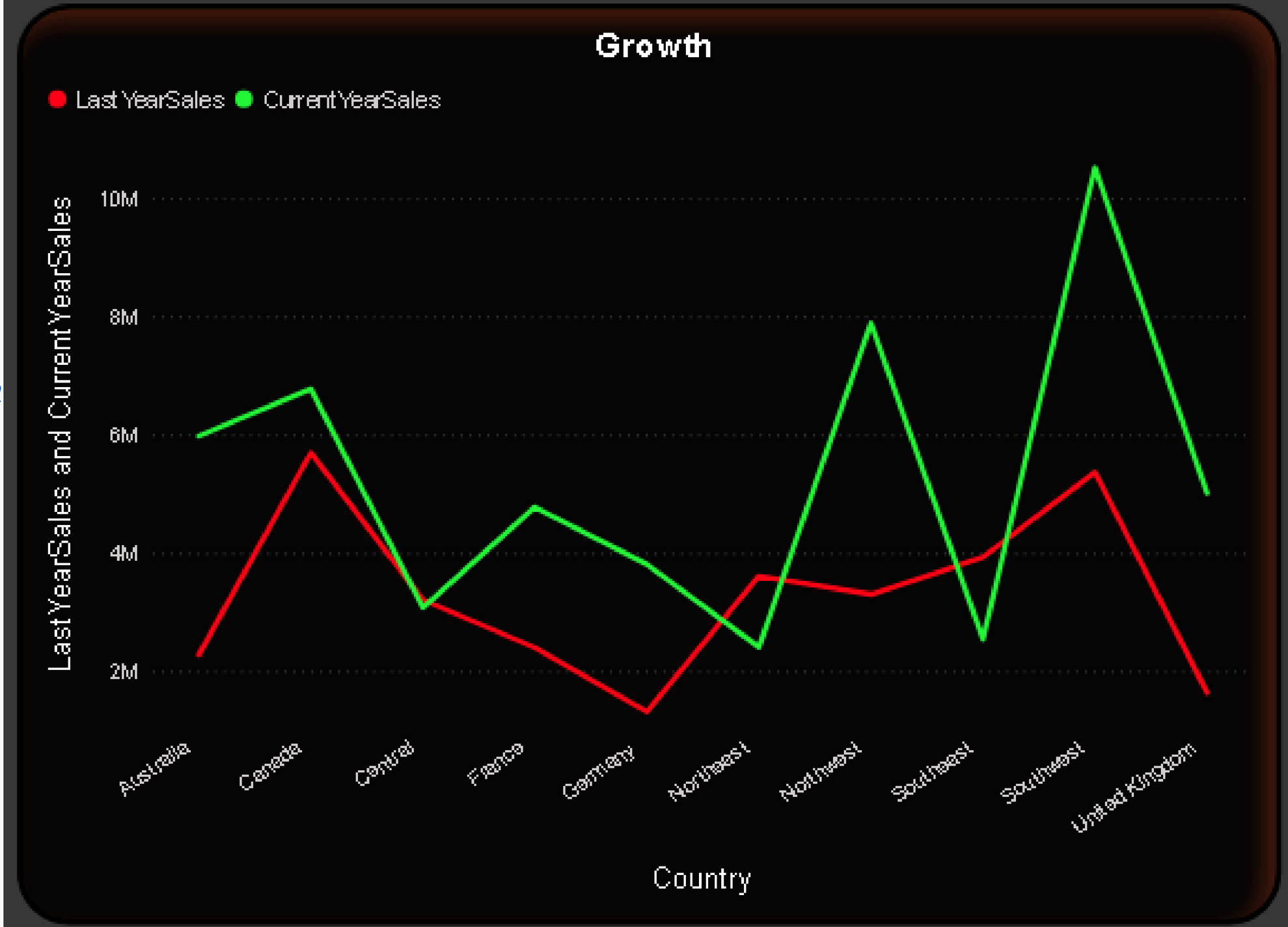

## *TotalSales made by each ProductSubcategory*

## *TotalSales made by:*

1. *OnlineSalse (Person Tables)*
2. *OfflineSales (Reseller tables)*



***Comparison in Total Sales of each country between last and current years***



# DashBoard on Customers

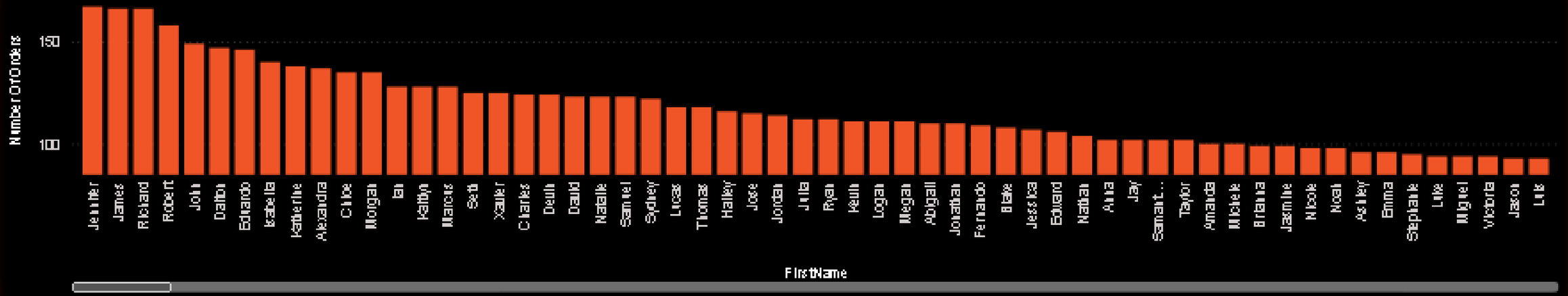
SALES | CUSTOMERS | PRODUCTS

**19.82K**
Customer

**130K**
Order

## Customers Sigmentaition


## Customers Orders




## *On Hovering on Map the data shows:*

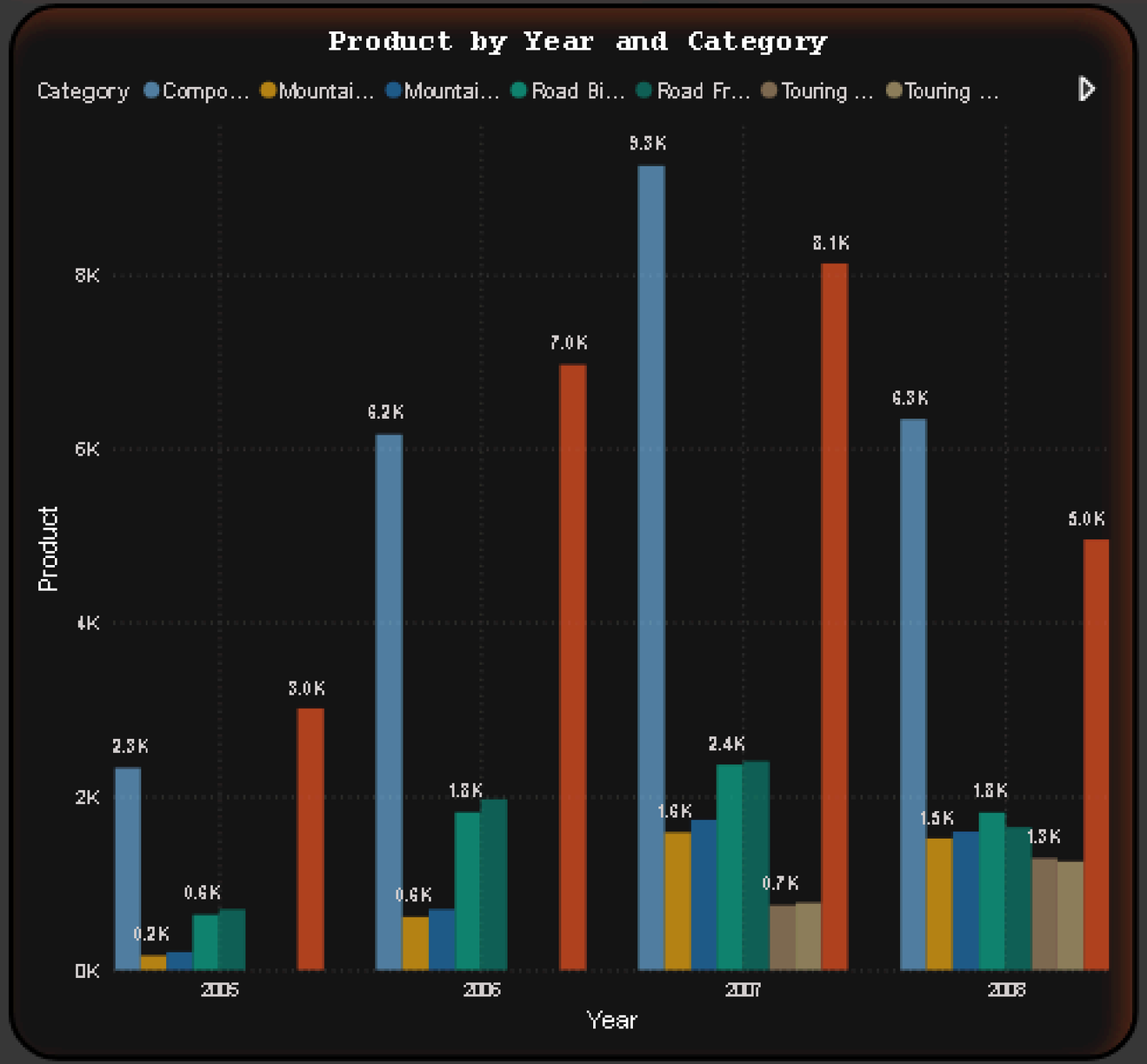
- *City name*
- *Number of terriority in this City*
- *Number of customers in this City*
- *Number of OnlineCustomers*

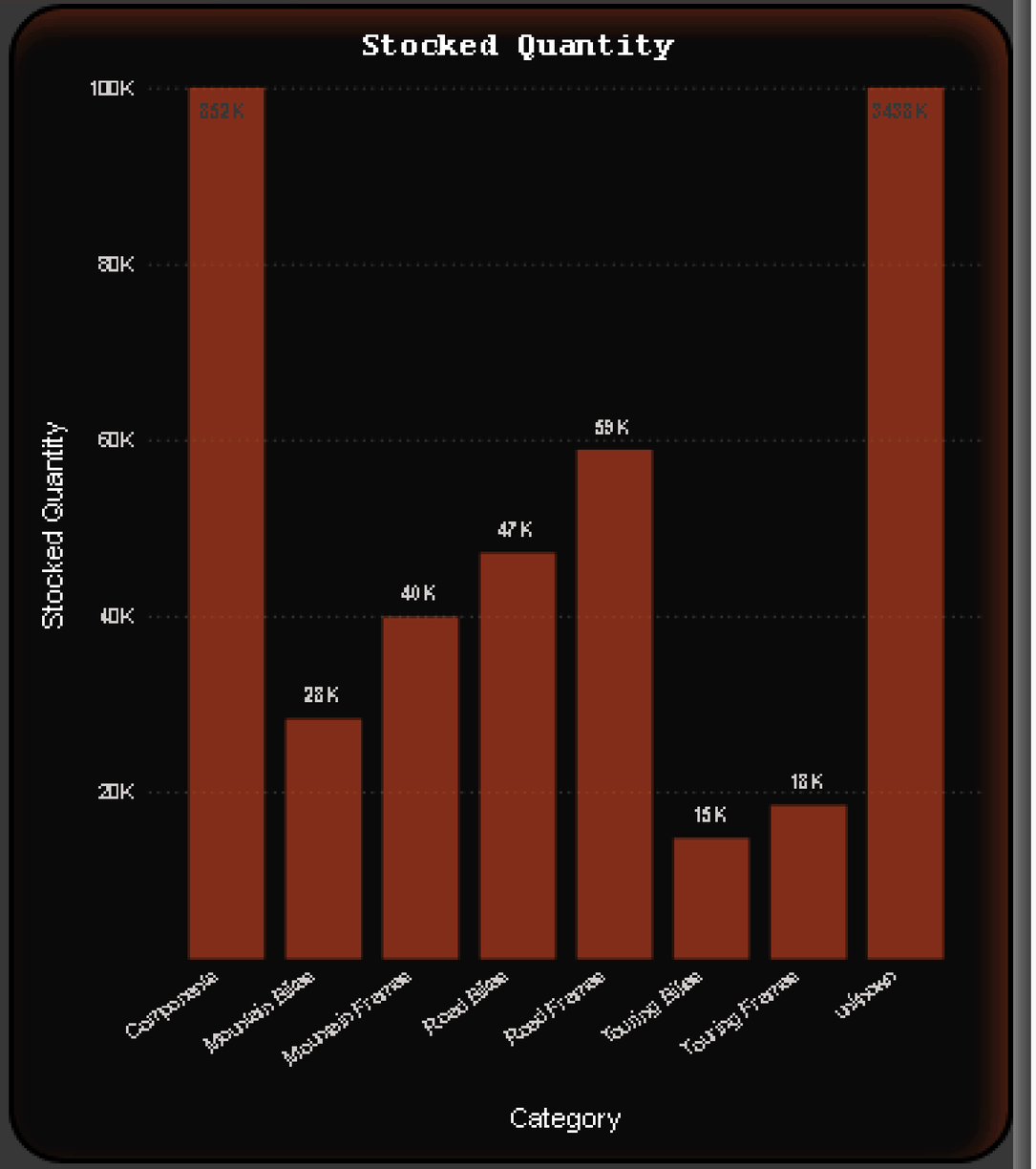
## *On hovering the data shows:*

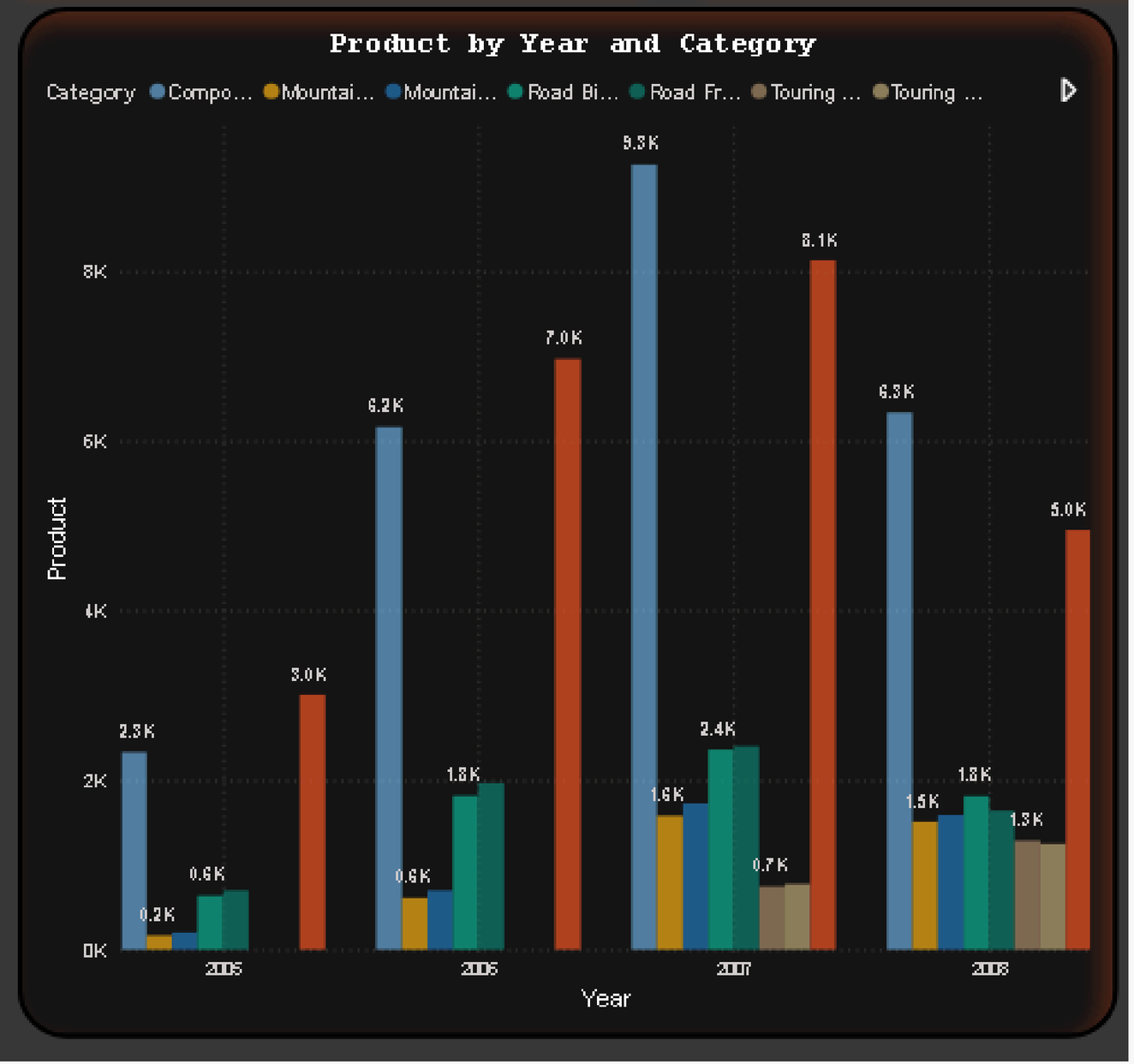
- *name of Customer*
- *number of his orders*

# DashBoard on Inventory

SALES | CUSTOMERS | PRODUCTS

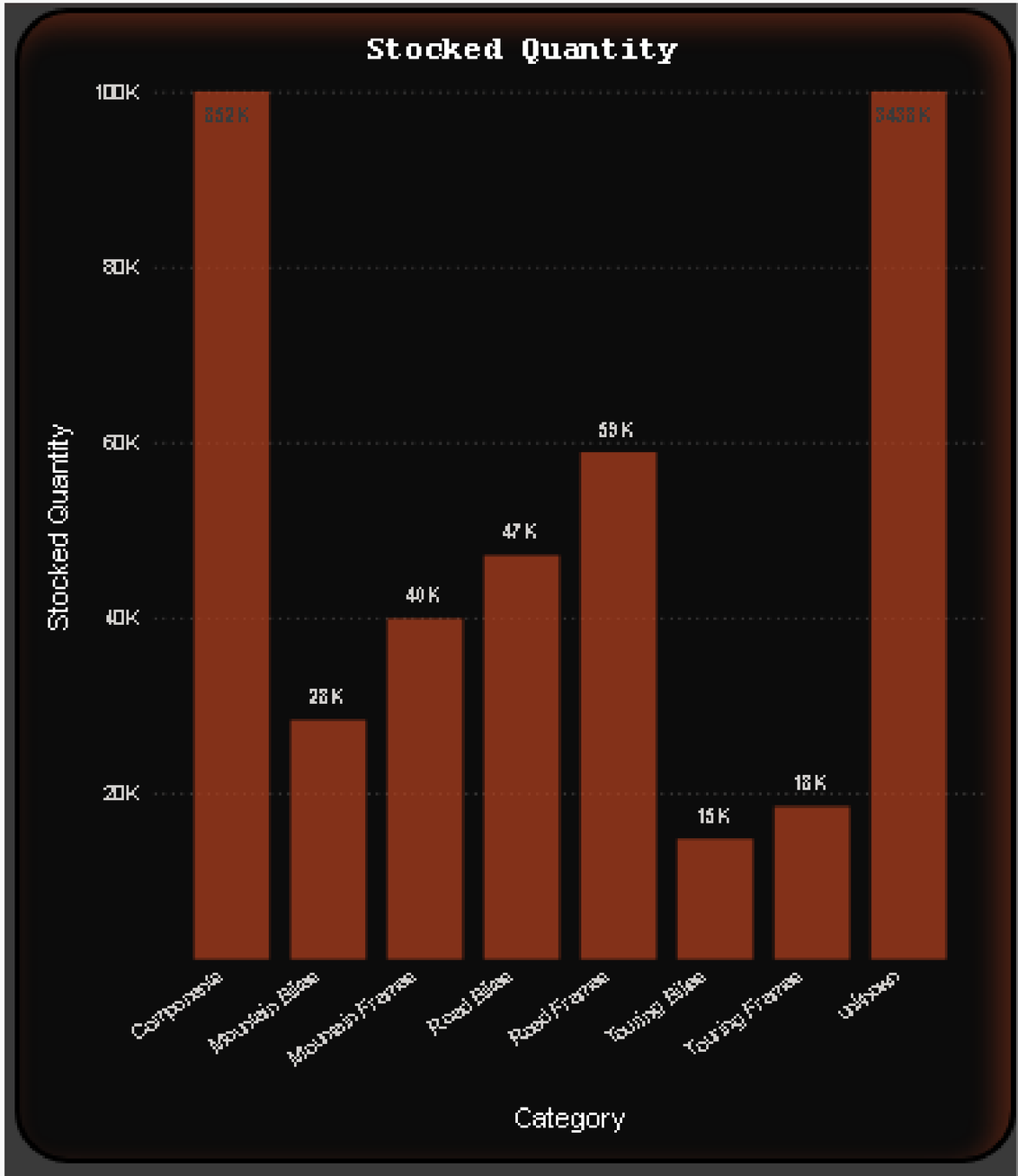





# *Stocked Products in each year*

# *Stocked Quantity on each Product Category*



# *Machine Learning Models*

# *Customer Segmentation model*

***Devolping features from the current data to help the model create different segments***

```
[6]  import datetime
     current_date = data['OrderDate'].max()
     year = current_date.year
     today =datetime.date(year, 12, 31)
     today=pd.to_datetime(today)

     # current_date = pd.to_datetime('2008-12-31')

     #Aggregate Features
     rfm = data.groupby('CustomerID').agg({
         'OrderDate': lambda x: (today - x.max()).days,  # Recency
         'TotalDue': ['count', 'sum']  # Frequency and Monetary
     }).reset_index()

     # Rename columns
     rfm.columns = ['CustomerID', 'Recency', 'Frequency', 'Monetary']
     temp=rfm.copy()
     rfm.head()
```

| | CustomerID | Recency | Frequency | Monetary |
|---|---|---|---|---|
| 0 | 11000 | 423 | 8 | 22784.02 |
| 1 | 11001 | 202 | 11 | 22376.68 |
| 2 | 11002 | 492 | 4 | 11639.06 |
| 3 | 11003 | 416 | 9 | 24704.66 |
| 4 | 11004 | 425 | 6 | 16982.72 |

***Scaling the data to avoid any parameter being neglected and aid the model to converge faster***

```
[7]  from sklearn.preprocessing import StandardScaler

     # Scale the RFM data: Models (use ditance between two points) Converge faster when the data is scaled.
     scaler = StandardScaler()
     rfm_scaled = scaler.fit_transform(rfm[['Recency', 'Frequency', 'Monetary']])

     rfm_scaled[:5]
```

# *Kmeans model to predict the customer segmentations from 0 to 2*

```python
from sklearn.cluster import KMeans

# Choose an optimal number of clusters, here we start with 4
n_clusters=3
kmeans = KMeans(n_clusters=n_clusters, random_state=7200)
rfm['Segment'] = kmeans.fit_predict(rfm_scaled)
rfm.head()
```

| | CustomerID | Recency | Frequency | Monetary | Segment |
|---|---|---|---|---|---|
| 0 | 11000 | 423 | 8 | 22784.02 | 0 |
| 1 | 11001 | 202 | 11 | 22376.68 | 0 |
| 2 | 11002 | 492 | 4 | 11639.06 | 0 |
| 3 | 11003 | 416 | 9 | 24704.66 | 0 |
| 4 | 11004 | 425 | 6 | 16982.72 | 0 |

## *Calculate mean of each parameter to observe differentiable between each one*

```
mean_segments = rfm.groupby('Segment')[['Recency', 'Frequency', 'Monetary']].mean()
mean_segments.head(n_clusters) # there are only four segments as we assigned before
```

| Segment | Recency | Frequency | Monetary |
|---|---|---|---|
| 0 | 326.695955 | 4.286850 | 2.617735e+04 |
| 1 | 283.070922 | 276.276596 | 1.658196e+07 |
| 2 | 994.014463 | 6.365702 | 2.165063e+05 |

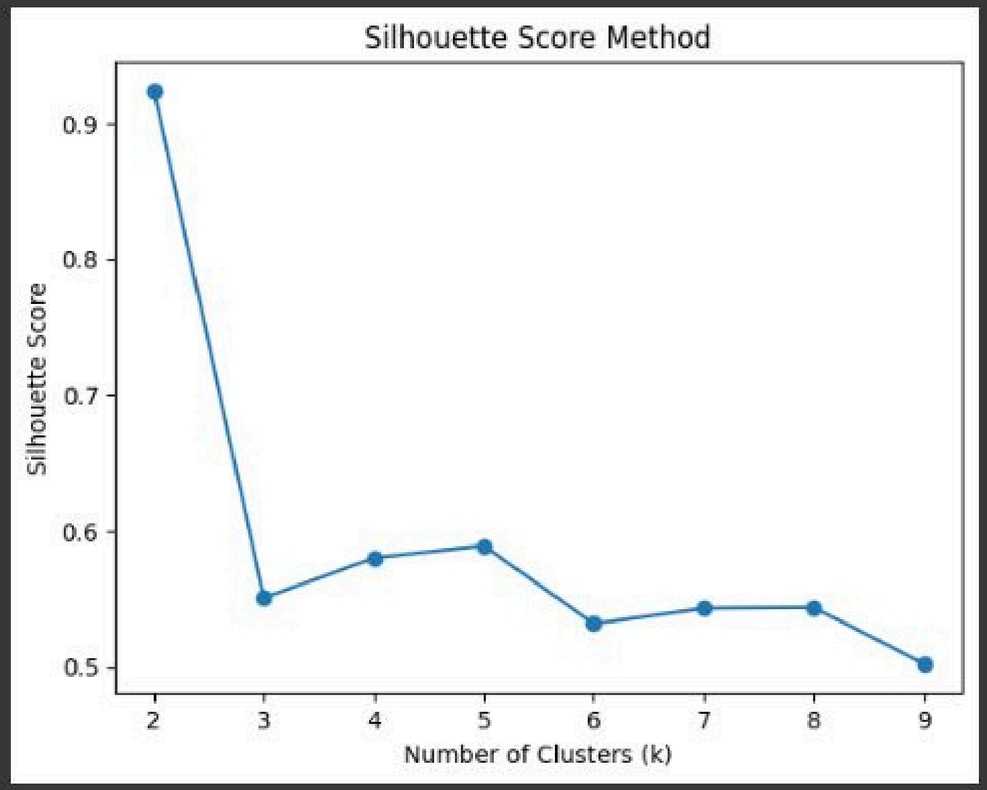
## *Silhouette Score to measure how similar  an object its own segment that other segments*

```
#Measures how similar an object is to its own cluster compared to other clusters. A higher score indicates better-defined clusters.
from sklearn.metrics import silhouette_score
silhouette_avg = silhouette_score(rfm_scaled, kmeans.labels_)
print("Silhouette Score:", silhouette_avg)
```

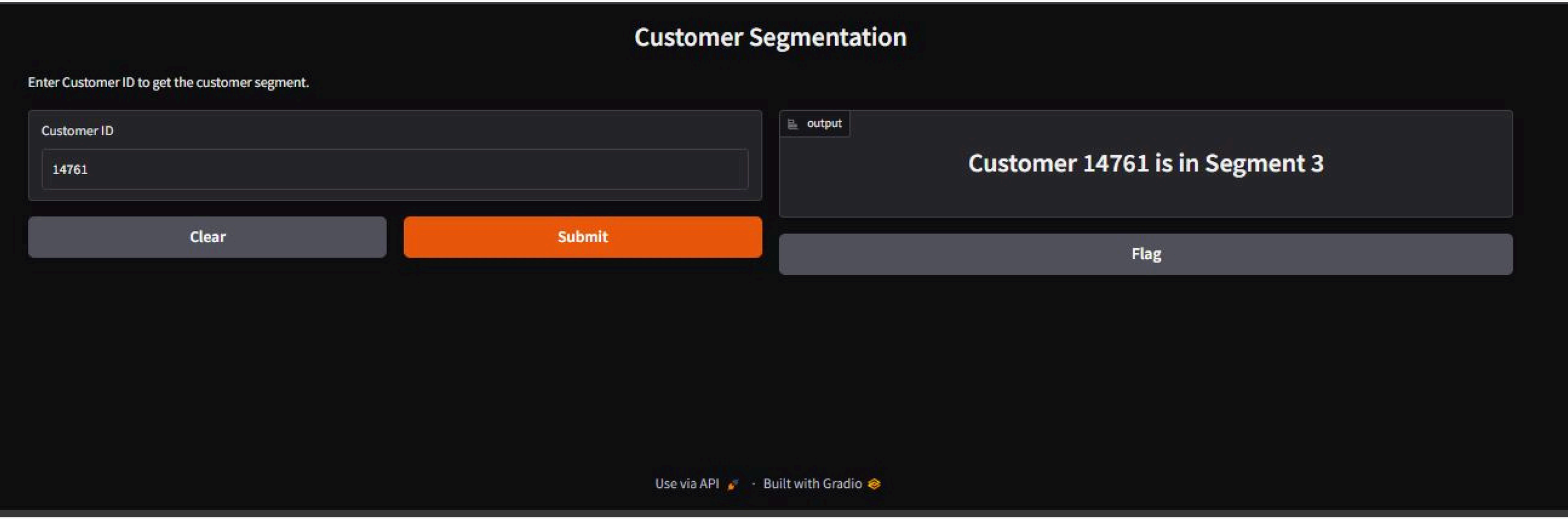
```
Silhouette Score: 0.7919463977515007
```

***Observing the Impact of increasing number of segments (Clusters) on the Silhouette Score***

```
[14] #Measures how similar an object is to its own cluster compared to other clusters. A higher score indicates better-defined cluste

     from sklearn.metrics import silhouette_score

     silhouette_scores = []

     for k in k_values:
         kmeans = KMeans(n_clusters=k, random_state=42)
         kmeans.fit(rfm_scaled)
         score = silhouette_score(rfm_scaled, kmeans.labels_)
         silhouette_scores.append(score)


     plt.plot(k_values, silhouette_scores, marker='o')
     plt.xlabel('Number of Clusters (k)')
     plt.ylabel('Silhouette Score')
     plt.title('Silhouette Score Method')
     plt.show()
```



# *Interface using Gradio library*



# *Customer top Recommendation:*

- *Assuming the rating range of the data*
- *Adding purchase column (=1) to the table*
- *Pivoting the data CustomerID against each product (fill_value=0 for not found*

```
[6] reader=Reader(rating_scale=(0,1))
    data['Purchased']=1
    user_product_matrix = data.pivot_table(index='CustomerID', columns='ProductKey', values='Purchased', fill_value=0)
    user_product_matrix.head()
```

| ProductKey | 707 | 708 | 709 | 710 | 711 | 712 | 713 | 714 | 715 | 716 | ... | 989 | 990 | 991 | 992 | 993 | 994 | 996 | 997 | 998 | 999 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CustomerID | | | | | | | | | | | | | | | | | | | | | |
| 11000 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 11001 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 |
| 11002 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 11003 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 11004 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

5 rows × 266 columns

# *Creating a table includes all products with CustomerID and and purchasing status*

```
[10] ratings=[]
     for user_id, row in user_product_matrix.iterrows():
       for product_id, purchase in row.items():
         ratings.append((user_id,product_id,purchase))
```

```
ratings_df=pd.DataFrame(ratings, columns=['CustomerID','ProductID','Purchased'])
temp=ratings_df.copy()
ratings_df.head()
```

| | CustomerID | ProductID | Purchased |
|---|---|---|---|
| 0 | 11000 | 707 | 1.0 |
| 1 | 11000 | 708 | 0.0 |
| 2 | 11000 | 709 | 0.0 |
| 3 | 11000 | 710 | 0.0 |
| 4 | 11000 | 711 | 0.0 |

***function to include two parameters CustomerID and number of required products to recommend using SVD (Singular Value Decomposition)***

```python
def get_product_recommendations(customer_id, n=50): # n: number of products that not Purchased
    if customer_id not in user_product_matrix.index:
        print(f"Customer ID {customer_id} not found in the data.")
        return []

    customer_products = user_product_matrix.loc[customer_id]
    unPurchased_products = customer_products[customer_products == 0].index.tolist()  # Get unPurchased products

    # Predict ratings for unPurchased products
    recommendations = []
    for product_id in unPurchased_products:
        est_rating = model.predict(customer_id, product_id).est
        recommendations.append((product_id, est_rating))

    # Sort recommendations by estimated rating
    recommendations.sort(key=lambda x: x[1], reverse=True) #x[1] to sort on the product_id column

    return recommendations[:n]
```

```python
customer_id=29672
recommended_products=get_product_recommendations(customer_id,n=50)
print(f"Top recommendations for Customer {customer_id}:")
for product_id, rating in recommended_products:
    print(f"Product ID: {product_id}, Estimated Rating: {rating}")
```

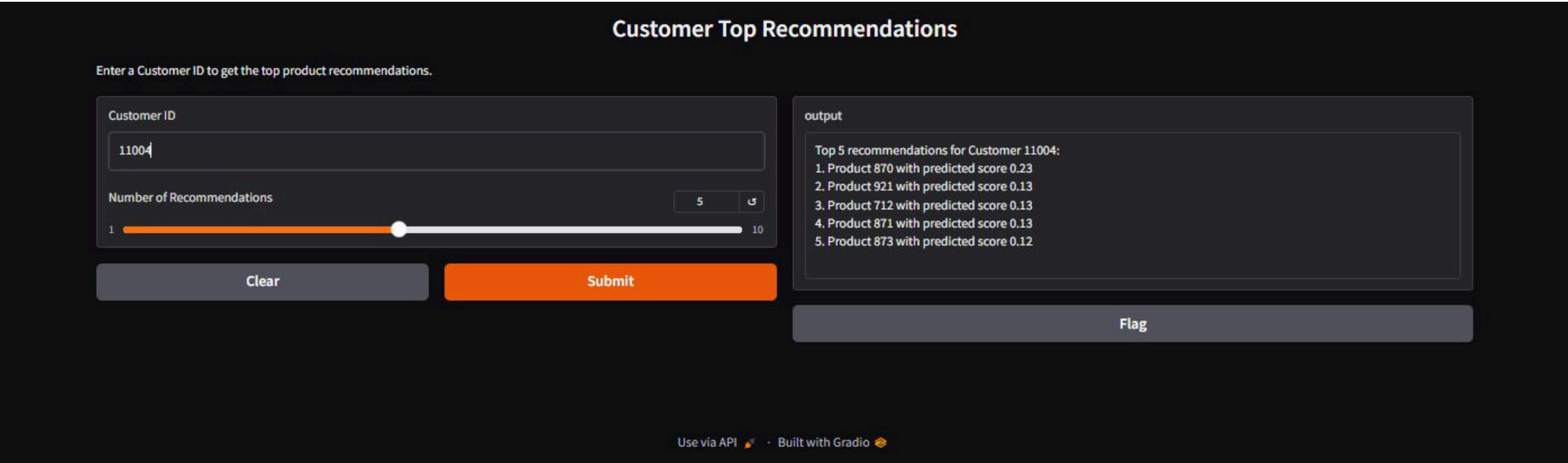
```
Top recommendations for Customer 29672:
Product ID: 870, Estimated Rating: 0.22556232247646732
Product ID: 921, Estimated Rating: 0.12867218675154357
Product ID: 712, Estimated Rating: 0.12793207564294576
Product ID: 871, Estimated Rating: 0.12585960742789734
Product ID: 873, Estimated Rating: 0.12377554087529982
Product ID: 922, Estimated Rating: 0.10166291031218368
Product ID: 708, Estimated Rating: 0.10135651624078025
Product ID: 711, Estimated Rating: 0.09302650563127182
Product ID: 878, Estimated Rating: 0.08747268055220002
Product ID: 872, Estimated Rating: 0.0797381908136001
Product ID: 930, Estimated Rating: 0.07011026872409298
```

```
[13] surprise_data=Dataset.load_from_df(ratings_df[['CustomerID','ProductID','Purchased']],reader)
     trainset, testset = train_test_split(surprise_data, test_size=0.2,random_state=300)
```

```
[14] model = SVD(reg_all=0.2)#req_all: controls data overfitting lowest:OverFitting,, Highesr::UnderFitting

     model.fit(trainset)

     predictions = model.test(testset)

     print("RMSE:", accuracy.rmse(predictions))
```

```
RMSE: 0.1174
```

# *Customer top Recommendation Interface*



# Customer Top Recommendations

Customer ID

11001

Number of Recommendations

5

1 10

Get Recommendations

Top 5 recommendations for Customer 11001:

1. Product 870 with predicted score 0.24
2. Product 921 with predicted score 0.14
3. Product 712 with predicted score 0.14
4. Product 871 with predicted score 0.14
5. Product 873 with predicted score 0.14

# Reseller Churn

## Adding some features to the data we have to enhance model accuracy

```
[5] #Features:

    #1- Recency: How many days since last Order
    import datetime
    year = last_order_date.year
    current_date =datetime.date(year, 12, 31)
    current_date=pd.to_datetime(current_date)
    # current_date = pd.to_datetime('2008-12-31')
    last_order_dates['DaysSinceLastPurchase'] = (current_date - last_order_dates['OrderDate']).dt.days
    last_order_dates.head(10)
```

| | ResellerKey | OrderDate | Churn | DaysSinceLastPurchase |
|---|---|---|---|---|
| 0 | 292 | 2007-06-01 | 0 | 579 |
| 1 | 294 | 2008-06-01 | 0 | 213 |
| 2 | 296 | 2008-06-01 | 0 | 213 |
| 3 | 298 | 2008-06-01 | 0 | 213 |
| 4 | 300 | 2008-06-01 | 0 | 213 |
| 5 | 302 | 2008-04-01 | 0 | 274 |
| 6 | 304 | 2007-03-01 | 1 | 671 |
| 7 | 306 | 2006-04-01 | 1 | 1005 |
| 8 | 308 | 2008-06-01 | 0 | 213 |
| 9 | 310 | 2006-05-01 | 1 | 975 |

```
#2- Frequency: Number of Orders per Reseller
Reseller_order_counts=data.groupby('ResellerKey')['SalesOrderNumber'].nunique().reset_index()
Reseller_order_counts.columns=['ResellerKey','Frequency']
Reseller_order_counts.head()
```

*(1)*

| | ResellerKey | Frequency |
|---|---|---|
| 0 | 292 | 7 |
| 1 | 294 | 4 |
| 2 | 296 | 12 |
| 3 | 298 | 11 |
| 4 | 300 | 4 |

Next steps: Generate code with Reseller_order_counts | View recommended plots | New interactive

# *Merging all features*

```
[7] #3- Monetary: Total SalesAmount per Reseller
reseller_sales_amount=data.groupby('ResellerKey')['TotalDue'].sum().reset_index()
reseller_sales_amount.columns = ['ResellerKey', 'Monetary']
reseller_sales_amount.head()
```

| | ResellerKey | Monetary |
|---|---|---|
| 0 | 292 | 4416690.45 |
| 1 | 294 | 5272394.94 |
| 2 | 296 | 19411014.86 |
| 3 | 298 | 500533.19 |
| 4 | 300 | 12230554.97 |

```
final_data = pd.merge(features, last_order_dates[['ResellerKey', 'Churn','DaysSinceLastPurchase']].drop_duplicates(),
temp=final_data.copy()
final_data.head()
```

*(2)*

| | ResellerKey | Frequency | Monetary | Churn | DaysSinceLastPurchase |
|---|---|---|---|---|---|
| 0 | 292 | 7 | 4416690.45 | 0 | 579 |
| 1 | 294 | 4 | 5272394.94 | 0 | 213 |
| 2 | 296 | 12 | 19411014.86 | 0 | 213 |
| 3 | 298 | 11 | 500533.19 | 0 | 213 |
| 4 | 300 | 4 | 12230554.97 | 0 | 213 |

***Splitting data with 80%, 20% train, test to help the model predict and then calculating accuracy.***

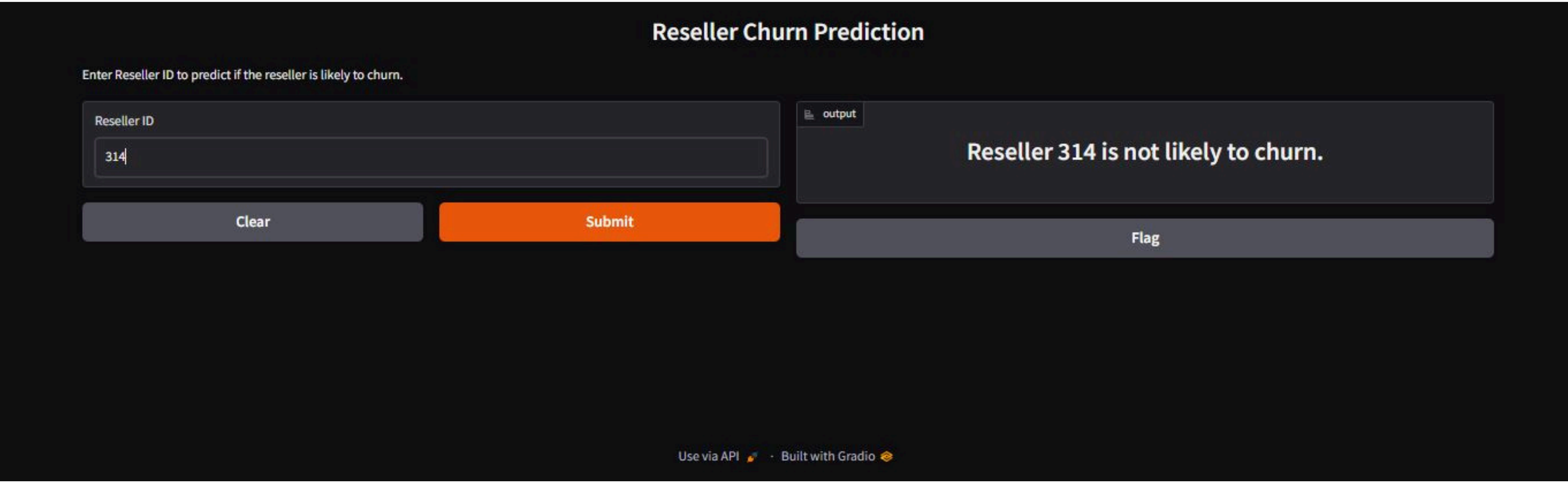
```
#Train Data
X = final_data.drop(columns=['Churn'],axis=1)
y = final_data['Churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Double-click (or enter) to edit

```
[11] #Predictions

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print(f"Accuracy: {accuracy*100} %")
```

```
Accuracy: 96.06299212598425 %
```

# *Reseller churn Interface*



# *Microsoft Azure*

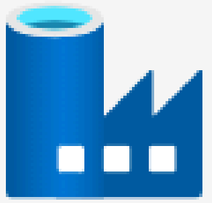
# AZURE MAP

## Create

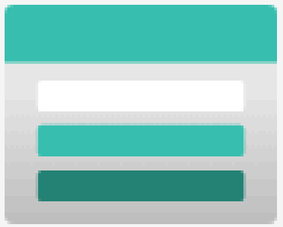
Resource Group


SQL Server


SQL Database

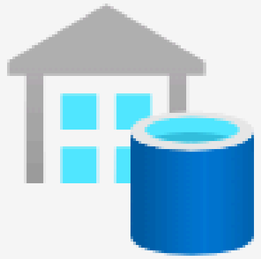


Data Factory



Storage Account

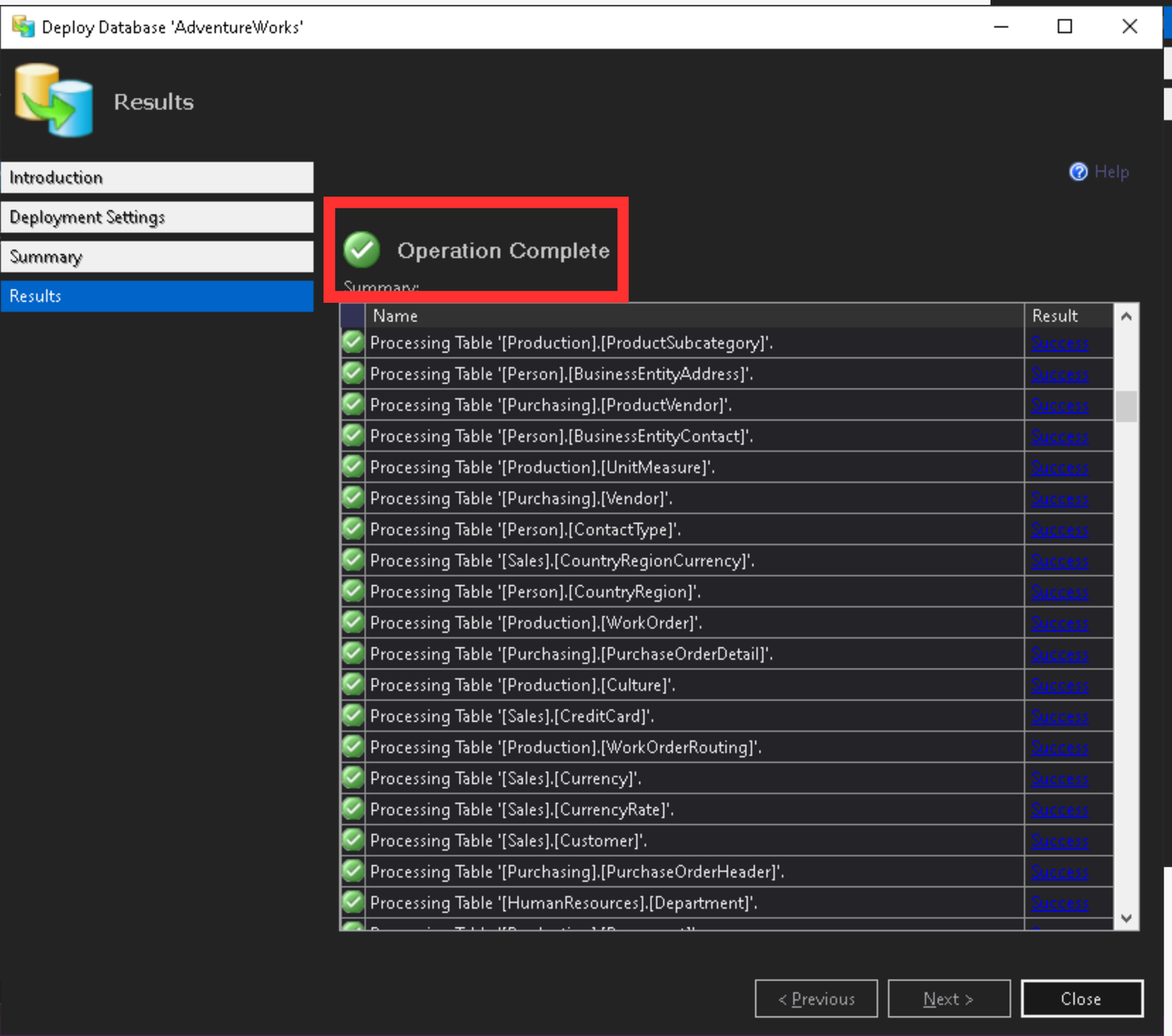
Synapse Analytics

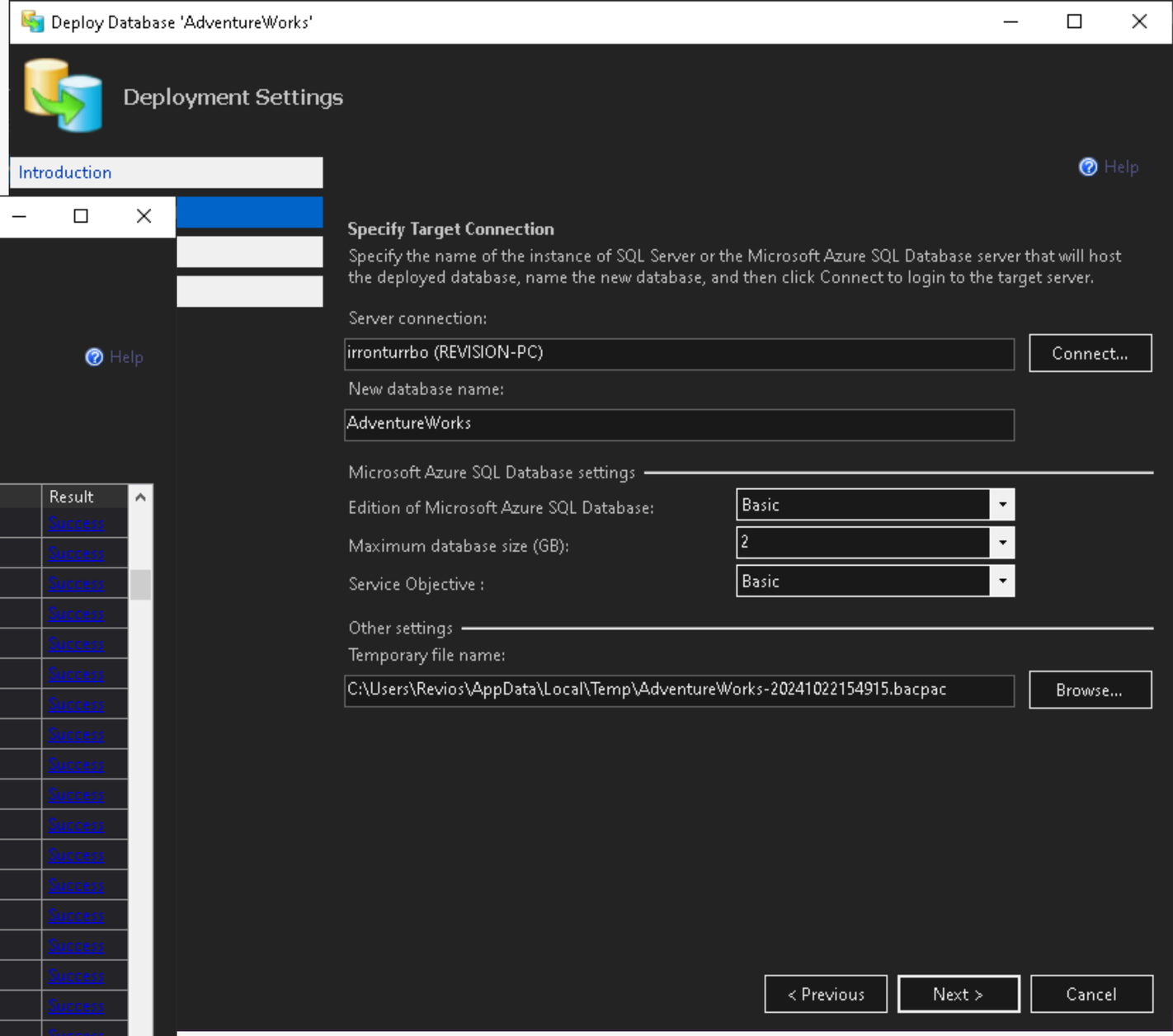


Dedicated SQL Pool

Integration Runtime

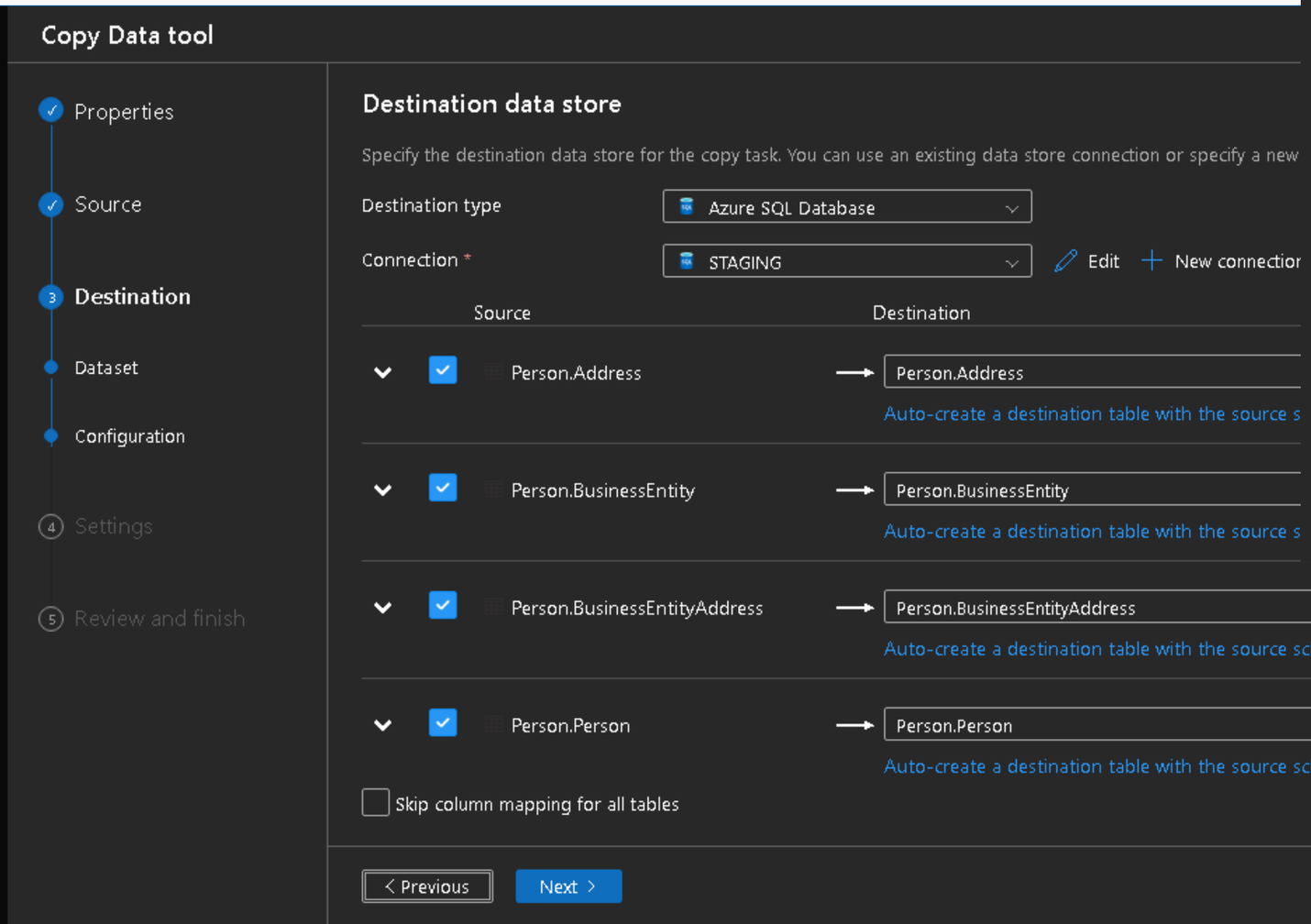
# First, We upload database (AdventureWorks) on Azure

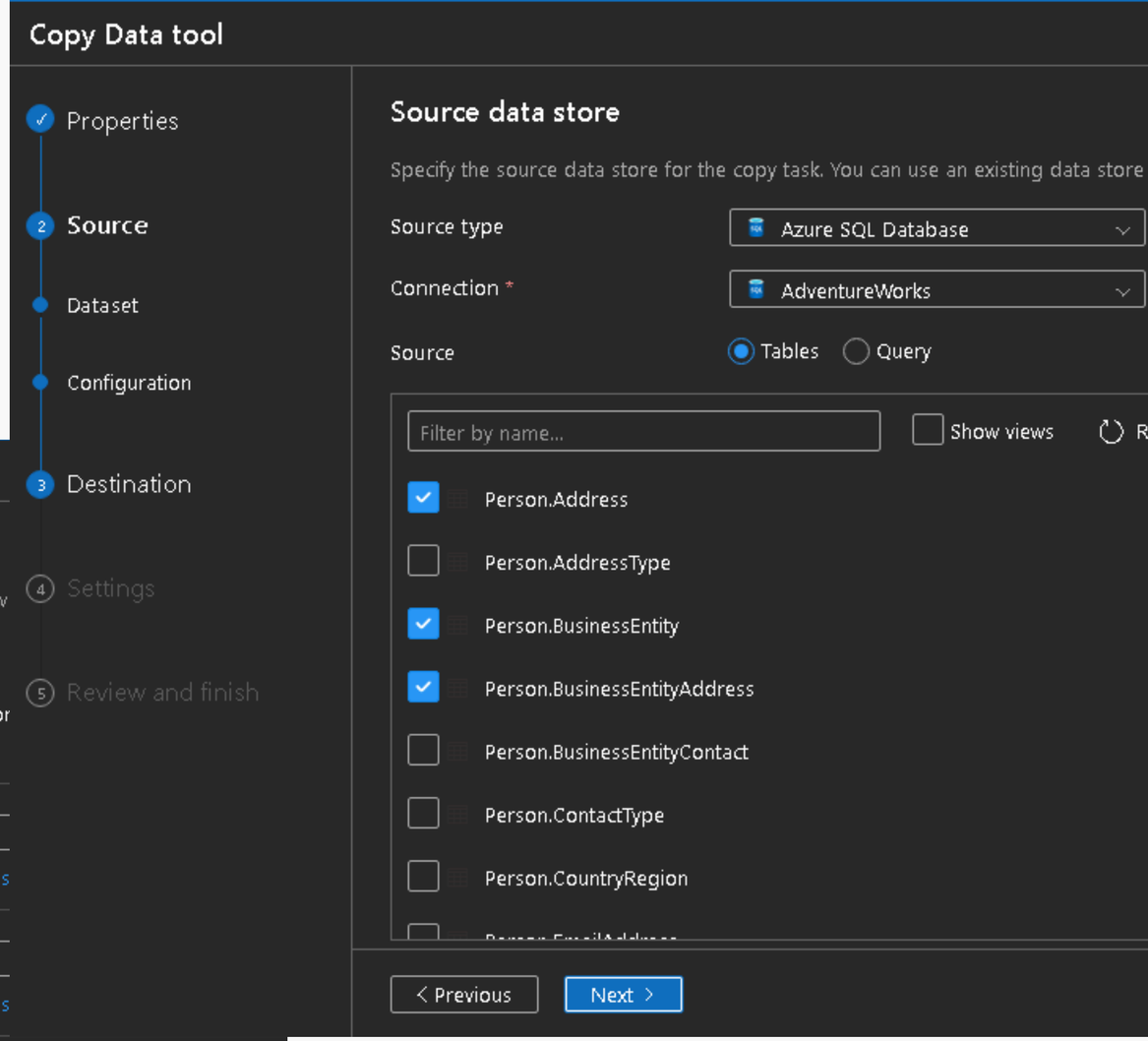



# Using Deploy Wizard

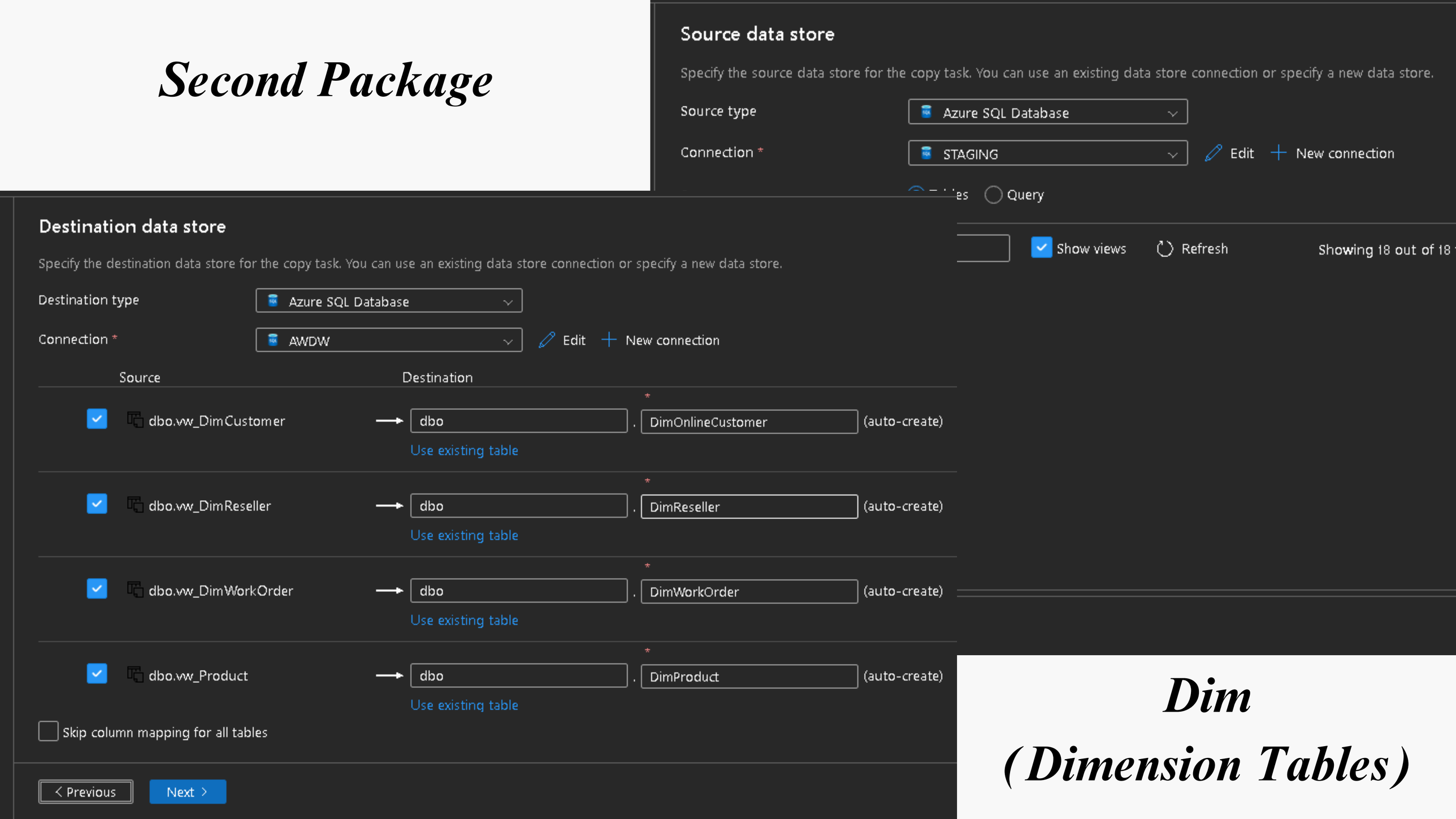
# *Then, We make the First SSIS Package the STAGING database*




# *Using Copy Data Tool from Data Factory*

# *Second Package*




# *Dim*
# *(Dimension Tables)*

## Source data store

Specify the source data store for the copy task. You can use an existing data store connection or specify a new data store.

Source type: Azure SQL Database

Connection *: STAGING  Edit  New connection

Source: Tables  Query

Filter by name...  Show views  Refresh  Showing 18 out of 18 tables, 7 out of 7 views (3 selected)

- Select all
- dbo.FactOnlineSales
- dbo.FactPurchaseOrder
- dbo.FactResellerSales
- dbo.TableTransferDetails
- dbo.vw_DimCustomer
- dbo.vw_DimReseller

Previous  Next

# *Third and Last Package*

## Destination data store

Specify the destination data store for the copy task. You can use an existing data store connection or specify a new data store.

Destination type: Azure SQL Database

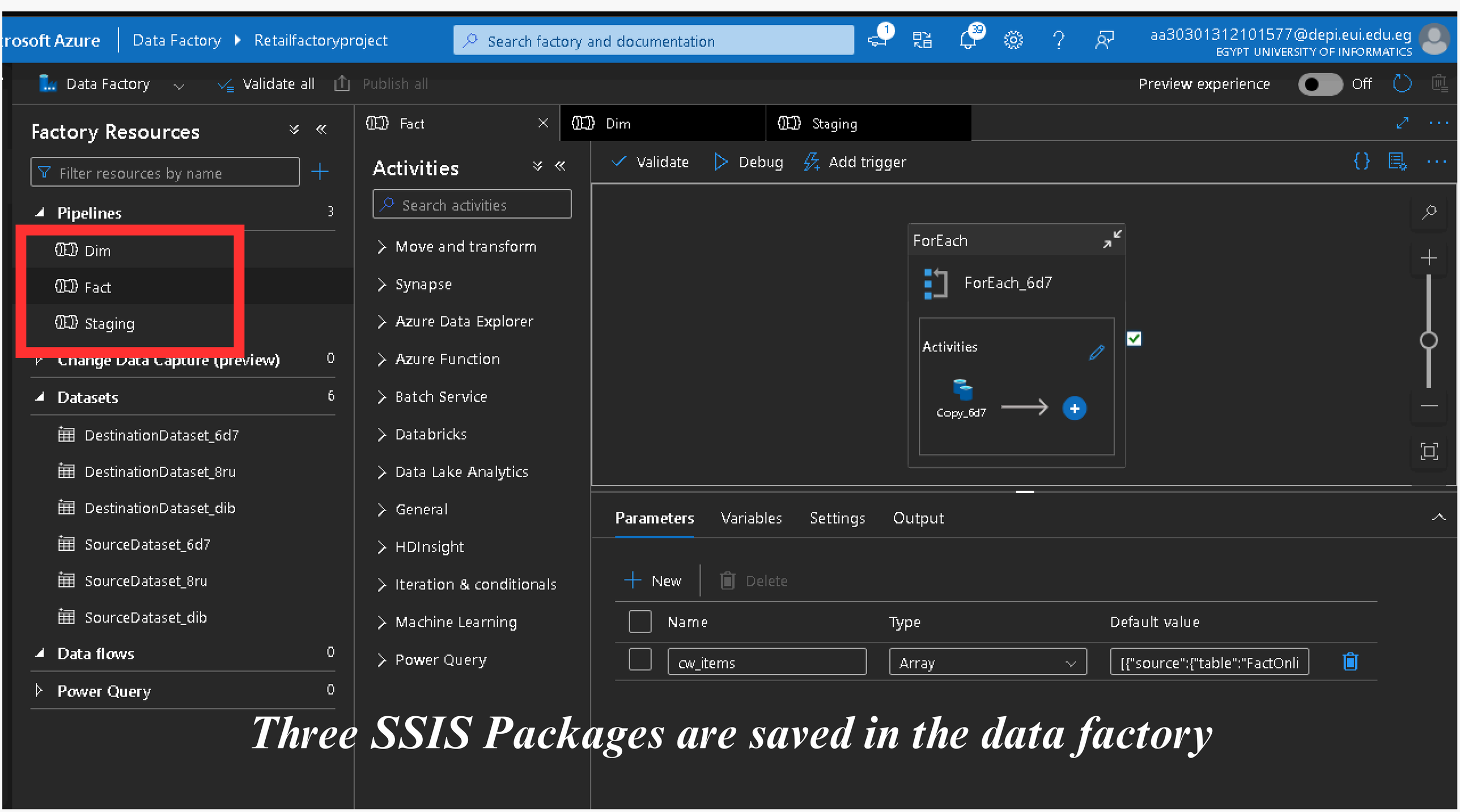
Connection *: AWDW  Edit  New connection

| Source | Destination | |
|---|---|---|
| dbo.FactOnlineSales | dbo . FactOnlineSales | (auto-create) |
| | Use existing table | |
| dbo.FactPurchaseOrder | dbo . FactPurchaseOrder | (auto-create) |
| | Use existing table | |
| dbo.FactResellerSales | dbo . FactResellerSales | (auto-create) |
| | Use existing table | |

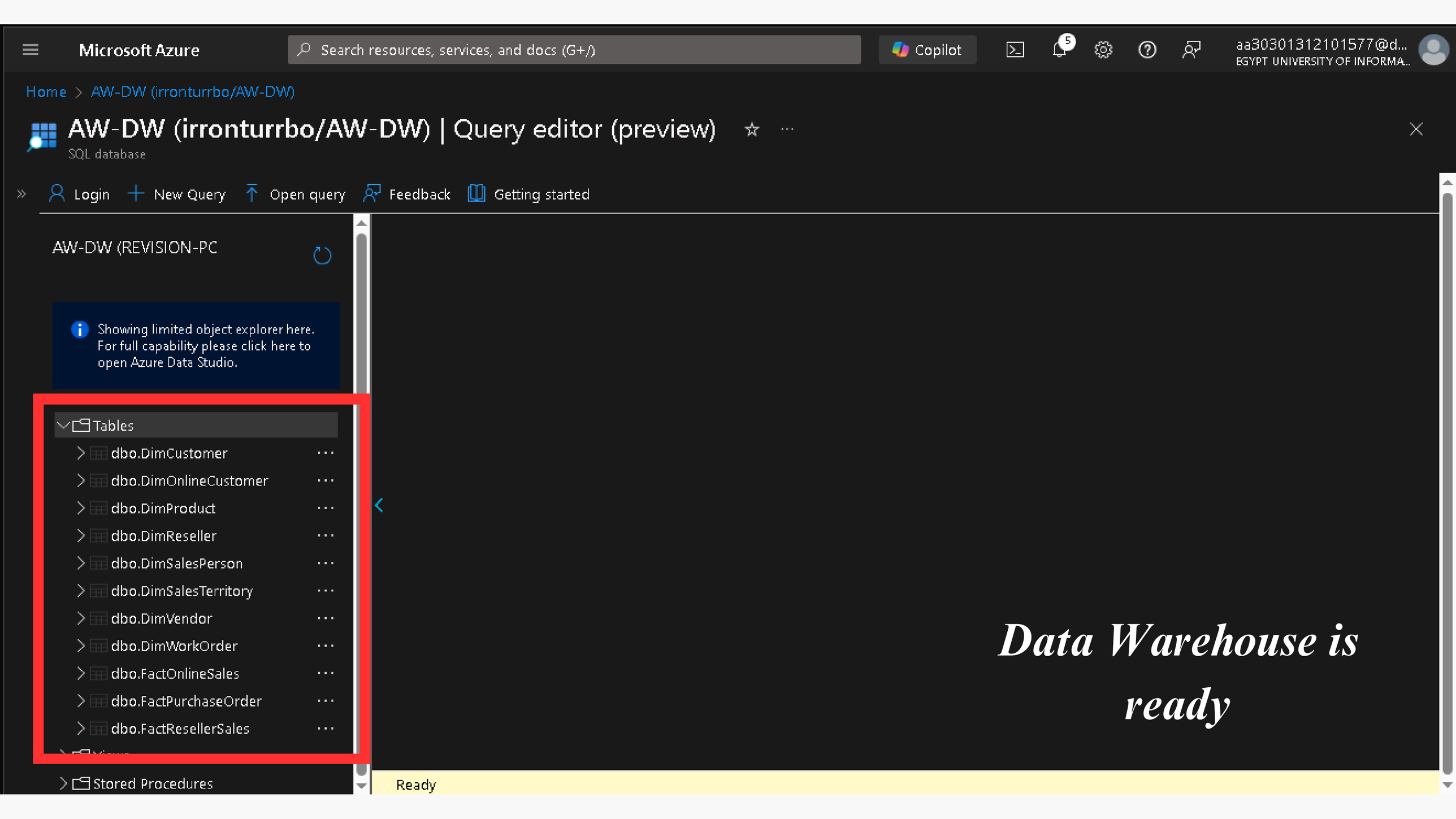
Skip column mapping for all tables

Previous  Next  Cancel

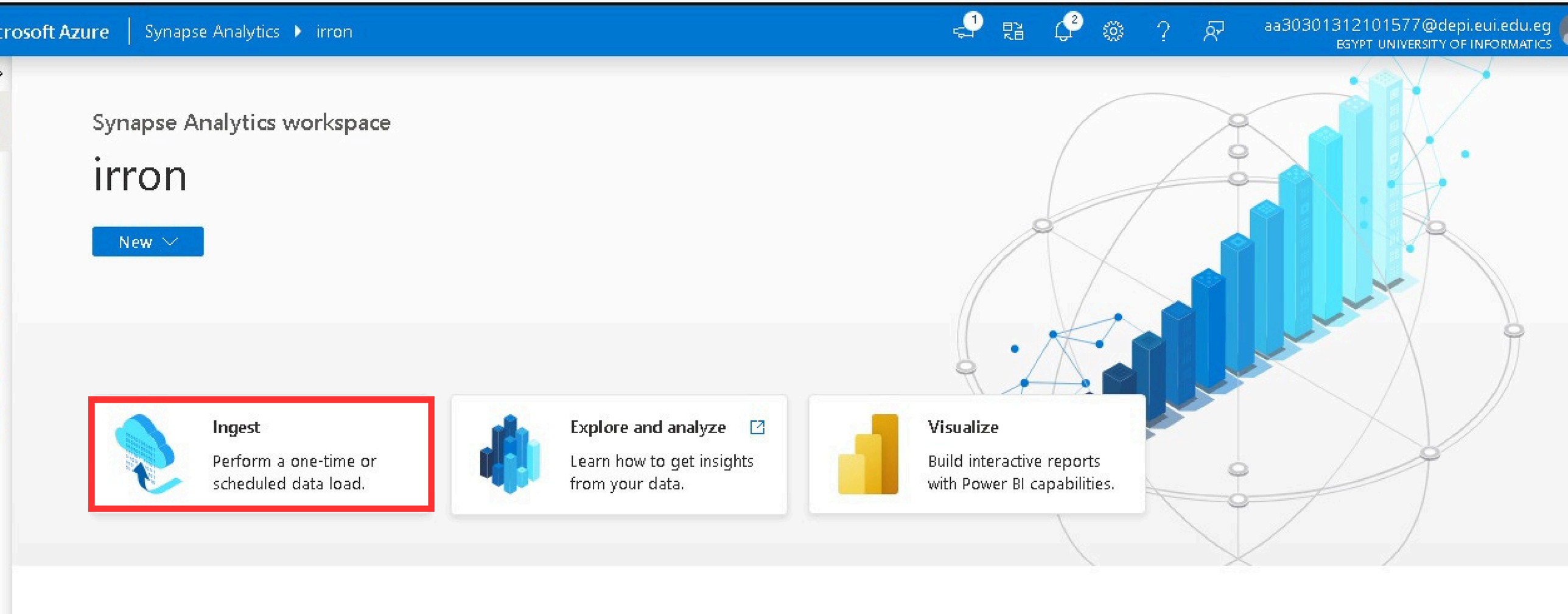
# *Fact(Fact Tables)*




***Three SSIS Packages are saved in the data factory***


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
aa30301312101577@d...
EGYPT UNIVERSITY OF INFORMA...
Home > AW-DW (irronturrbo/AW-DW)
AW-DW (irronturrbo/AW-DW) | Query editor (preview)
SQL database
Login
New Query
Open query
Feedback
Getting started
AW-DW (REVISION-PC
Showing limited object explorer here. For full capability please click here to open Azure Data Studio.
Tables
dbo.DimCustomer
dbo.DimOnlineCustomer
dbo.DimProduct
dbo.DimReseller
dbo.DimSalesPerson
dbo.DimSalesTerritory
dbo.DimVendor
dbo.DimWorkOrder
dbo.FactOnlineSales
dbo.FactPurchaseOrder
dbo.FactResellerSales
Stored Procedures
Ready
Data Warehouse is ready

# *Now, We can use Ingest (Copy Data Tool) from Synapse Analytics to copy data from the data warehouse to Synapse SQl pool for further Analysis*



Microsoft Azure | Synapse Analytics ▸ irron

Search

aa30301312101577@depi.eui.edu.eg
EGYPT UNIVERSITY OF INFORMATICS

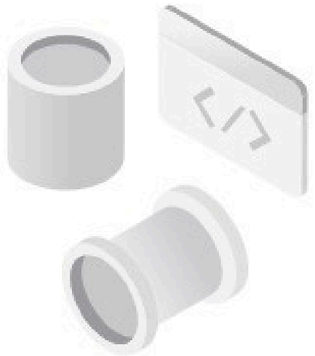
Synapse live | Validate all | Publish all

## Data

Workspace | Linked

Filter resources by name

- SQL database
  - pool1 (SQL)
    - Tables
      - dbo.DimCustomer
      - dbo.DimDate
      - dbo.DimOnlineCustomer
      - dbo.DimProduct
      - dbo.DimReseller
      - dbo.DimSalesPerson
      - dbo.DimSalesTerritory
      - dbo.DimVendor
      - dbo.DimWorkOrder
      - dbo.FactInternetSales
      - dbo.FactPurchaseOrder
      - dbo.FactResellerSales
      - dbo.OfflineSales



Select an item

Use the resource explorer to select or create a new item

***Data is ready to be queried on it in Synapse***

```
103     SELECT      --OFFLINE
104         [ProductcategoryName],
105         [ProductSubcategoryName],
106         TotalSales
107     FROM
108         OfflineSales
109     )AS TOTAL
110 GROUP BY
111     [ProductcategoryName],
112     [ProductSubcategoryName]
113 ORDER BY
114     TotalSales DESC;
```

Results Messages

Select Query 4 View Table Chart Export results

Search

| ProductcategoryName | ProductSubcategoryName | TotalSales |
|---|---|---|
| Bikes | Road Bikes | 73436398.4000 |
| Bikes | Mountain Bikes | 63291855.8100 |
| Bikes | Touring Bikes | 25245017.0500 |

00:00:09 Query executed successfully.

```
11      (([SalesYTD]-[SalesLastYear])/[SalesLastYear])*100 AS PercentageGrowth
12  FROM
13      [dbo].[DimSalesTerritory]
14  ORDER BY
15      PercentageGrowth DESC;
16
17
18
19  --  Customers Distribution
```

Results Messages

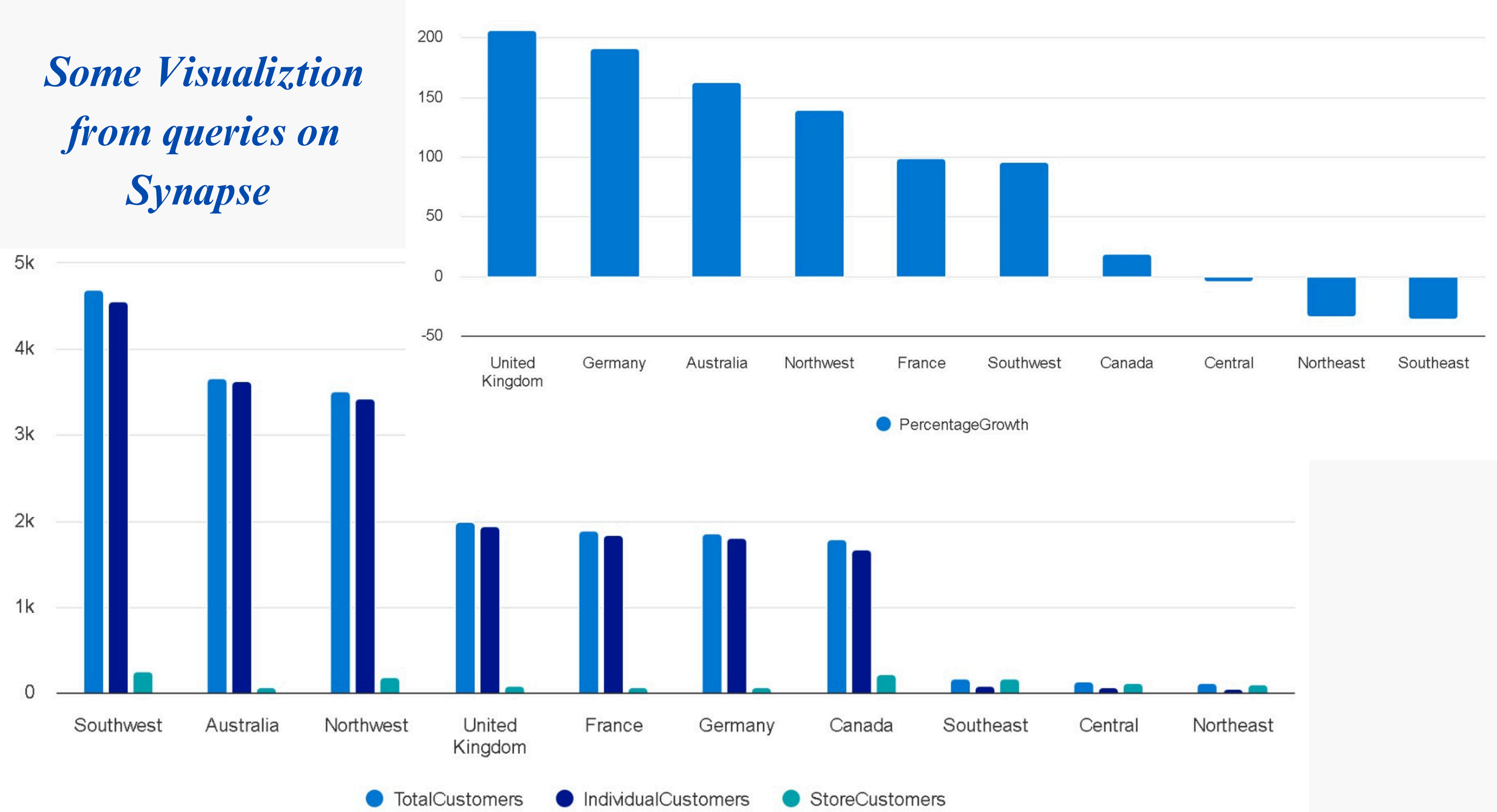
Select Query 0 View Table Chart Export results

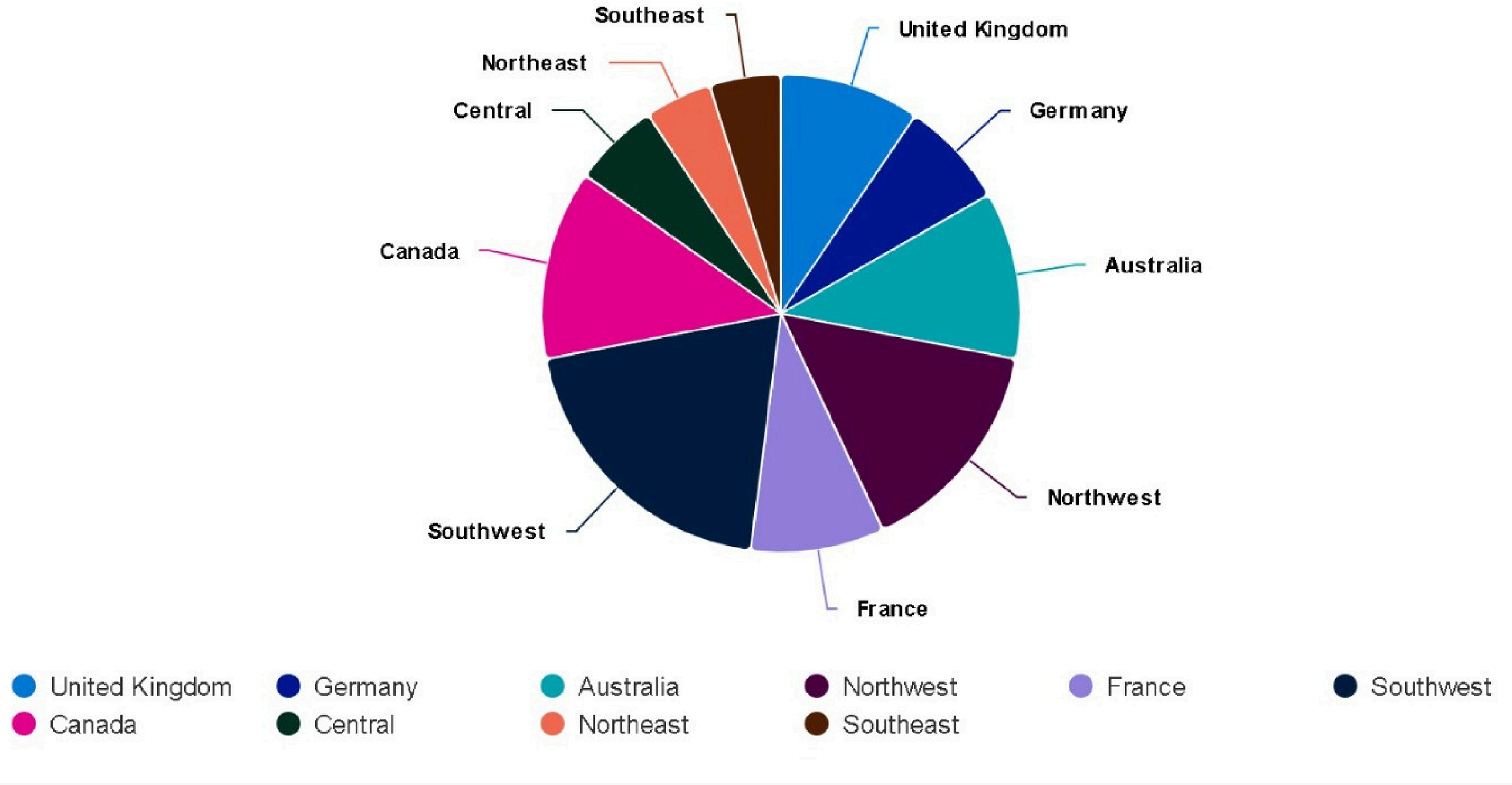
Search

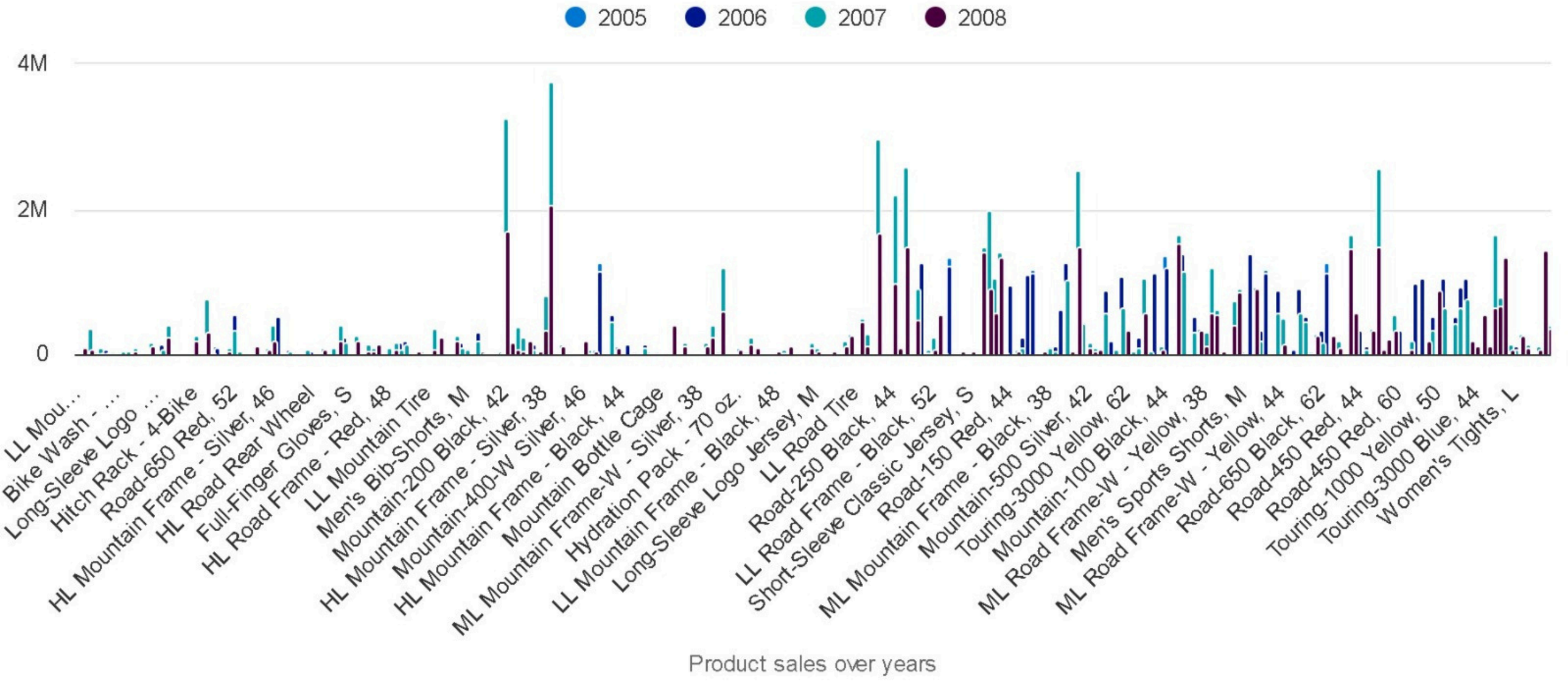
| Country | Continent | CurrentYearSales | LastYearSales |
|---|---|---|---|
| United Kingdom | Europe | 5012905.3656 | 1635823.3967 |
| Germany | Europe | 3805202.3478 | 1307949.7917 |
| Australia | Pacific | 5977814.9154 | 2278548.9776 |
| Northwest | North America | 7887186.7882 | 3298694.4938 |
| France | Europe | 4772398.3078 | 2396539.7601 |

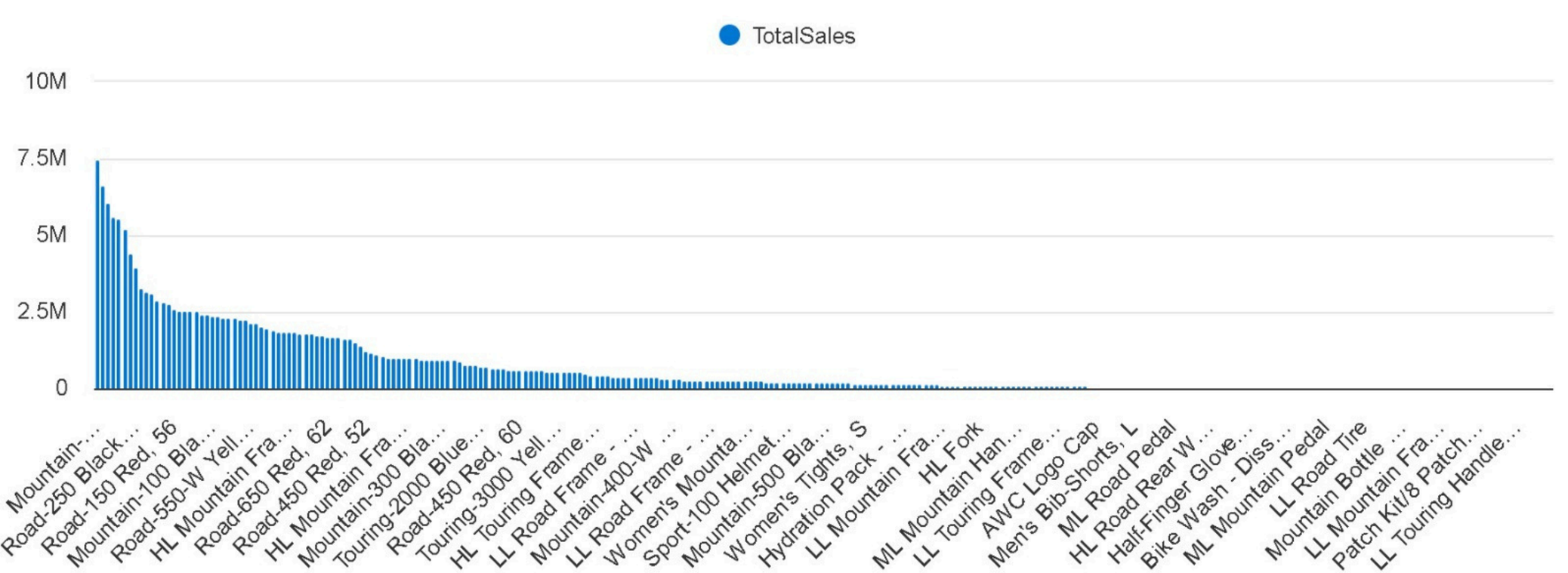
***Some Results from queries on Synapse***

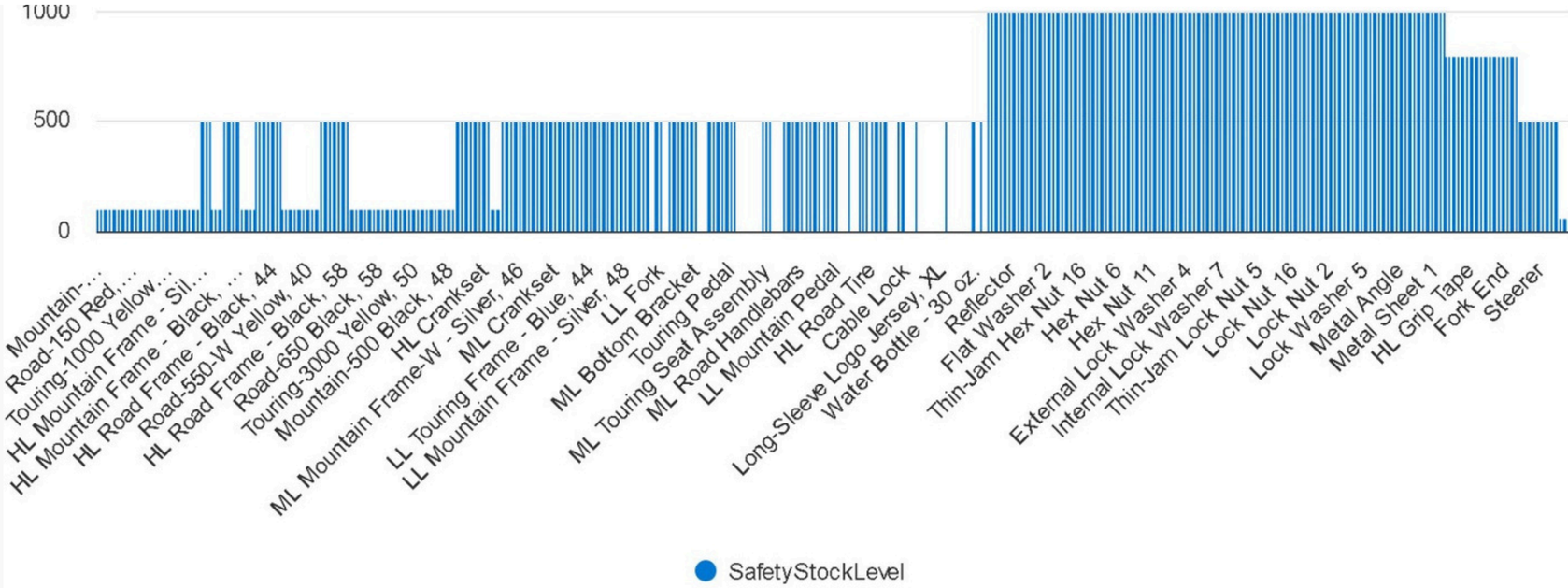
# *Some Visualiztion from queries on Synapse*




Southeast
United Kingdom
Northeast
Germany
Central
Australia
Canada
Northwest
Southwest
France
United Kingdom
Germany
Australia
Northwest
France
Southwest
Canada
Central
Northeast
Southeast


2005
2006
2007
2008
4M
2M
0
LL Mou...
Bike Wash - ...
Long-Sleeve Logo ...
Hitch Rack - 4-Bike
Road-650 Red, 52
HL Mountain Frame - Silver, 46
HL Road Rear Wheel
Full-Finger Gloves, S
HL Road Frame - Red, 48
LL Mountain Tire
Men's Bib-Shorts, M
Mountain-200 Black, 42
HL Mountain Frame - Silver, 38
Mountain-400-W Silver, 46
ML Mountain Frame - Black, 44
Mountain Bottle Cage
HL Mountain Frame-W - Silver, 38
Hydration Pack - 70 oz.
LL Mountain Frame - Black, 48
Long-Sleeve Logo Jersey, M
LL Road Tire
Road-250 Black, 44
LL Road Frame - Black, 52
Short-Sleeve Classic Jersey, S
Road-150 Red, 44
ML Mountain Frame - Black, 38
Mountain-500 Silver, 42
Touring-3000 Yellow, 62
Mountain-100 Black, 44
ML Road Frame-W - Yellow, 38
Men's Sports Shorts, M
ML Road Frame-W - Yellow, 44
Road-650 Black, 62
Road-450 Red, 44
Road-450 Red, 60
Touring-1000 Yellow, 50
Touring-3000 Blue, 44
Women's Tights, L
Product sales over years


TotalSales
10M
7.5M
5M
2.5M
0
Mountain-...
Road-250 Black...
Road-150 Red, 56
Mountain-100 Bla...
Road-550-W Yell...
HL Mountain Fra...
Road-650 Red, 62
Road-450 Red, 52
HL Mountain Fra...
Mountain-300 Bla...
Touring-2000 Blue...
Road-450 Red, 60
Touring-3000 Yell...
HL Touring Frame...
LL Road Frame - ...
Mountain-400-W ...
LL Road Frame - ...
Women's Mounta...
Sport-100 Helmet...
Mountain-500 Bla...
Women's Tights, S
Hydration Pack - ...
LL Mountain Fra...
HL Fork
ML Mountain Han...
LL Touring Frame...
AWC Logo Cap
Men's Bib-Shorts, L
ML Road Pedal
HL Road Rear W...
Half-Finger Glove...
Bike Wash - Diss...
ML Mountain Pedal
LL Road Tire
Mountain Bottle ...
LL Mountain Fra...
Patch Kit/8 Patch...
LL Touring Handle...


1000
500
0
Mountain-...
Road-150 Red,...
Touring-1000 Yellow...
HL Mountain Frame - Sil...
HL Mountain Frame - Black, ...
HL Road Frame - Black, 44
Road-550-W Yellow, 40
HL Road Frame - Black, 58
Road-650 Black, 58
Touring-3000 Yellow, 50
Mountain-500 Black, 48
HL Crankset
ML Mountain Frame-W - Silver, 46
ML Crankset
LL Touring Frame - Blue, 44
LL Mountain Frame - Silver, 48
LL Fork
ML Bottom Bracket
Touring Pedal
ML Touring Seat Assembly
ML Road Handlebars
LL Mountain Pedal
HL Road Tire
Cable Lock
Long-Sleeve Logo Jersey, XL
Water Bottle - 30 oz.
Reflector
Flat Washer 2
Thin-Jam Hex Nut 16
Hex Nut 6
Hex Nut 11
External Lock Washer 4
Internal Lock Washer 7
Thin-Jam Lock Nut 5
Lock Nut 16
Lock Nut 2
Lock Washer 5
Metal Angle
Metal Sheet 1
HL Grip Tape
Fork End
Steerer
SafetyStockLevel

*Thank you*