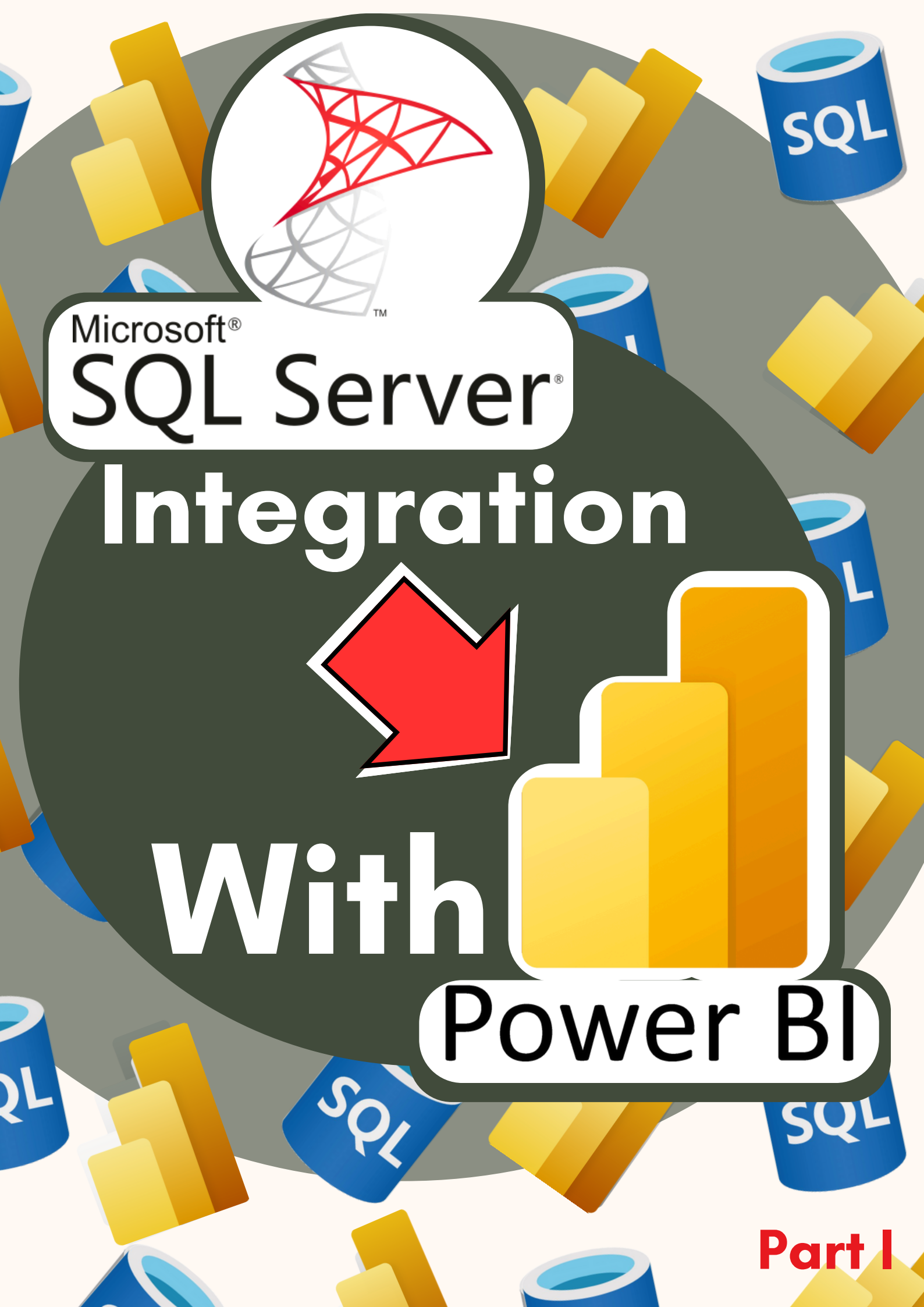

Microsoft®
SQL Server®
Integration
With
Power BI
Part I

This document outlines the process of creating a Business Intelligence (BI) project that uses Microsoft SQL Server as its database and Power BI for data visualization and analysis. The goal is to extract and transform data, define key performance indicators (KPIs), and create an interactive dashboard for sales analysis.

1. Context and Overview

The project aims to analyze sales data from the AdventureWorks 2022 database, a sample database widely used for learning purposes. The focus will be on Internet sales analysis, covering important metrics such as revenue, profit, and quantity of products sold, as well as segmentations by product category, customers, gender, and country.

AdventureWorks2022.bak

2. Environment Setup

The project aims to analyze sales data from the AdventureWorks 2022 database, a sample database widely used for learning purposes. The focus will be on Internet sales analysis, covering important metrics such as revenue, profit, and quantity of products sold, as well as segmentations by product category, customers, gender, and country.

AdventureWorks2022.bak

2. Environment Setup

The project aims to analyze sales data from the AdventureWorks 2022 database, a sample database widely used for learning purposes. The focus will be on Internet sales analysis, covering important metrics such as revenue, profit, and quantity of products sold, as well as segmentations by product category, customers, gender, and country.

AdventureWorks2022.bak

3. Defining Project Key Performance Indicators (KPIs)

To guide the analysis and ensure the dashboard meets business objectives, the following KPIs have been defined. They will be grouped into two main tabs in the Power BI dashboard:

General and Customers

3. General Tab

Total Revenue: Sum of all sales

Quantity Sold: Total number of items sold

Total Product Categories:
Count of product categories

Number of Customers:
Count of unique customers

Total Revenue and Total Profit by Month:
Analysis of sales and profit evolution over time

Profit Margin: Profit percentage relative to revenue

Quantity Sold by Month:
Analysis of monthly sales volume.

Profit by Country:
Comparison of profitability across different countries

3. Customers Tab

Sales by Country:
Distribution of sales by geographic territory

Customers by Country:
Distribution of the customer base by geographic territory.

Sales by Gender:
Analysis of sales segmented by customer gender

Sales by Category:
Analysis of revenue generated by each product category

4. Data Extraction and Modeling

For Power BI to perform analyses efficiently, it's crucial to extract the data optimally. Instead of importing all tables and performing joins in Power BI the strategy is to create a view in Microsoft SQL Server This centralizes the business logic at the data source, ensuring better performance and reusability.

4. Selected Tables and Columns

The following tables and columns have been identified as essential for calculating the KPIs:

FactInternetSales: Contains sales information

- SalesOrderNumber
- OrderDate
- OrderQuantity
- SalesAmount
- TotalProductCost

4. Selected Tables and Columns

The following tables and columns have been identified as essential for calculating the KPIs:

DimProductCategory: Contains product categories

EnglishProductCateg

DimCustomer: Contains customer information

CustomerKey

Gender

FirstName and LastName

DimGeography: Contains customer location

EnglishCountryRegionName

5. Creating the View

Microsoft SQL Server

The RESULTADOS_ADW view is the central piece of the project. It joins the necessary tables and pre-calculates some columns, simplifying the analysis in Power BI

View Code

SQLQuery4.sql - L...M4NC\samsung (51))*

```sql
CREATE OR ALTER VIEW RESULTS_ADW AS
SELECT
    fis.SalesOrderNumber AS 'Order No.',
    DATEADD(YEAR, 9, fis.OrderDate) AS 'Order Date',
    dpc.EnglishProductCategoryName AS 'Product Category',
    fis.CustomerKey AS 'Customer ID',
    dc.FirstName + ' ' + dc.LastName AS 'Customer Name',
    REPLACE(REPLACE(dc.Gender, 'M', 'Male'), 'F', 'Female') AS 'Gender',
    dg.EnglishCountryRegionName AS 'Country',
    fis.OrderQuantity AS 'Quantity Sold',
    fis.SalesAmount AS 'Sales Revenue',
    fis.TotalProductCost AS 'Sales Cost',
    fis.SalesAmount - fis.TotalProductCost AS 'Sales Profit'
FROM FactInternetSales fis
INNER JOIN DimProduct dp ON fis.ProductKey = dp.ProductKey
    INNER JOIN DimProductSubcategory dps ON dp.ProductSubcategoryKey = dps.ProductSubcategoryKey
        INNER JOIN DimProductCategory dpc ON dps.ProductCategoryKey = dpc.ProductCategoryKey
INNER JOIN DimCustomer dc ON fis.CustomerKey = dc.CustomerKey
    INNER JOIN DimGeography dg ON dc.GeographyKey = dg.GeographyKey
```

Note: The join between FactInternetSales and DimProductCategory is a chain relationship, passing through DimProduct and DimProductSubcategory. This ensures the category information is correct.

5. Creating the View

Microsoft SQL Server

View Code

```
CREATE OR ALTER VIEW RESULTS_ADW AS
SELECT
   fis.SalesOrderNumber AS 'ORDER NUMBER',
   DATEADD(YEAR, 9, fis.OrderDate) AS 'Order Date',  -- add 9 years
   dpc.EnglishProductCategoryName AS 'PRODUCT CATEGORY',
   fis.CustomerKey AS 'CUSTOMER ID',
   dc.FirstName + ' ' + dc.LastName AS 'CUSTOMER NAME',
   REPLACE(REPLACE(dc.Gender, 'M', 'Male'), 'F', 'Female') AS 'GENDER',
   dg.EnglishCountryRegionName AS 'COUNTRY',
   fis.OrderQuantity AS 'QUANTITY SOLD',
   fis.SalesAmount AS 'SALES REVENUE',
   fis.TotalProductCost AS 'SALES COST',
   fis.SalesAmount - fis.TotalProductCost AS 'SALES PROFIT'
FROM FactInternetSales fis
INNER JOIN DimProduct dp ON fis.ProductKey = dp.ProductKey
   INNER JOIN DimProductSubcategory dps ON dp.ProductSubcategoryKey =
dps.ProductSubcategoryKey
      INNER JOIN DimProductCategory dpc ON dps.ProductCategoryKey =
dpc.ProductCategoryKey
INNER JOIN DimCustomer dc ON fis.CustomerKey = dc.CustomerKey
   INNER JOIN DimGeography dg ON dc.GeographyKey = dg.GeographyKey;
```

Note: The join between FactInternetSales and DimProductCategory is a chain relationship, passing through DimProduct and DimProductSubcategory. This ensures the category information is correct.

5. Creating the View

Microsoft® SQL Server

Note: The join between FactInternetSales and DimProductCategory is a chain relationship, passing through DimProduct and DimProductSubcategory. This ensures the category information is correct.

INNER JOIN

FactInternetSales → **DimProduct** → **DimProduct Subcategory** → **DimProductCategory**

FactInternetSales
- ProductKey

→

DimProduct
- ProductKey
- ProductSubcategoryKey

ProductKey

DimProduct
- ProductKey
- ProductSubcategoryKey

→

DimProductSubcategory
- ProductSubcategoryKey
- ProductCategoryKey

ProductSubcategoryKey

DimProductSubcategory
- ProductSubcategoryKey
- ProductCategoryKey

→

DimProductCategory
- ProductCategoryKey

ProductCategoryKey

[FactInternetSales] --ProductKey--> [DimProduct] --ProductSubcategoryKey--> [DimProductSubcategory] --ProductCategoryKey--> [DimProductCategory]

(dashed line: indirect logical relation — chain)

6. Additional Analysis: The Year 2021

For a more specific case study, a second view, VENDAS_INTERNET, was created to focus only on sales made in the year 2021. This allows for a deeper analysis of a specific period.

View Code

SQLQuery4.sql - L...M4NC\samsung (51))*

```sql
CREATE OR ALTER VIEW ONLINE_SALE AS
SELECT
    fis.SalesOrderNumber AS 'Order No.',
    CAST(DATEADD(YEAR, 9, fis.OrderDate) AS DATE) AS 'Order Date',
    dpc.EnglishProductCategoryName AS 'Product Category',
    dc.FirstName + ' ' + dc.LastName AS 'Customer Name',
    dst.SalesTerritoryCountry AS 'Country',
    fis.OrderQuantity AS 'Quantity Sold',
    fis.TotalProductCost AS 'Sales Cost',
    fis.SalesAmount AS 'Sales Revenue'
FROM FactInternetSales fis
INNER JOIN DimProduct dp ON fis.ProductKey = dp.ProductKey
    INNER JOIN DimProductSubcategory dps ON dp.ProductSubcategoryKey = dps.ProductSubcategoryKey
        INNER JOIN DimProductCategory dpc ON dps.ProductCategoryKey = dpc.ProductCategoryKey
INNER JOIN DimCustomer dc ON fis.CustomerKey = dc.CustomerKey
INNER JOIN DimSalesTerritory dst ON fis.SalesTerritoryKey = dst.SalesTerritoryKey
WHERE YEAR(CAST(DATEADD(YEAR, 9, fis.OrderDate) AS DATE)) = 2021
```

6. Additional Analysis: The Year 2021

View Code

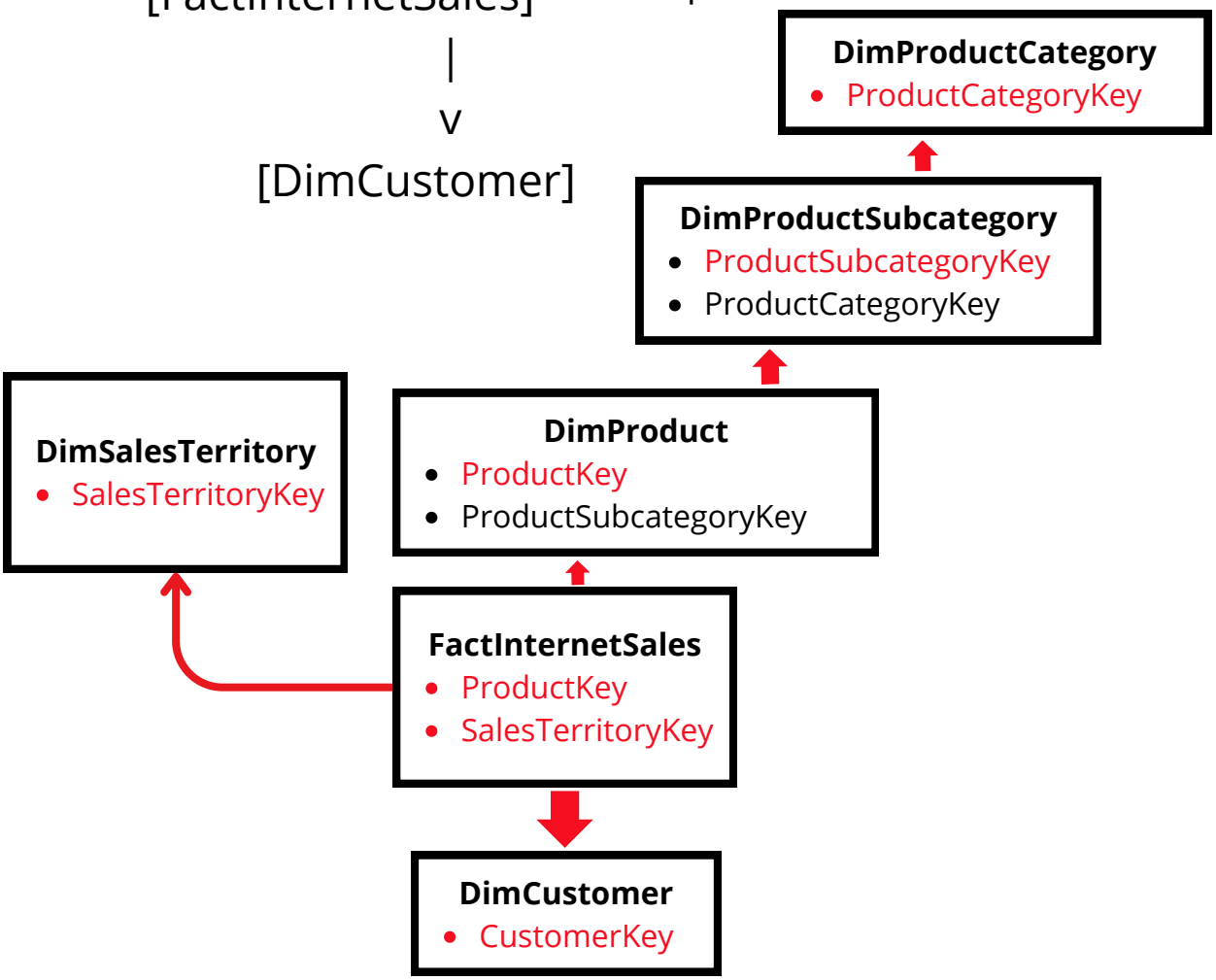
```
CREATE OR ALTER VIEW ONLINE_SALES AS
SELECT
 fis.SalesOrderNumber AS 'Order Number',
 DATEADD(YEAR, 9, fis.OrderDate) AS 'Order Date',  -- add 9 years
 dpc.EnglishProductCategoryName AS 'Product Category',
 dc.FirstName + ' ' + dc.LastName AS 'Customer Name',
 dst.SalesTerritoryCountry AS 'Country',
 fis.OrderQuantity AS 'Quantity Sold',
 fis.TotalProductCost AS 'Sales Cost',
 fis.SalesAmount AS 'Sales Revenue'
FROM FactInternetSales AS fis
INNER JOIN DimProduct AS dp ON fis.ProductKey = dp.ProductKey
 INNER JOIN DimProductSubcategory AS dps ON dp.ProductSubcategoryKey =
dps.ProductSubcategoryKey
  INNER JOIN DimProductCategory AS dpc ON dps.ProductCategoryKey =
dpc.ProductCategoryKey
INNER JOIN DimCustomer AS dc ON fis.CustomerKey = dc.CustomerKey
INNER JOIN DimSalesTerritory AS dst ON fis.SalesTerritoryKey = dst.SalesTerritoryKey
WHERE YEAR(CAST(DATEADD(YEAR, 9, fis.OrderDate) AS DATE)) = 2021

SELECT * FROM ONLINE_SALES;
```

6. Additional Analysis: The Year 2021

```
                [DimProductCategory]
                         ^
                         |
            [DimProductSubcategory]
                         ^
                         |
[DimSalesTerritory]  [DimProduct]
          ^              ^
          |              |
 +--------------[FactInternetSales]----------+
                         |
                         v
                  [DimCustomer]
```

Note: The join between FactInternetSales and DimCustomer is a direct relationship through CustomerKey, ensuring customer details are correctly linked. Similarly, the join with DimSalesTerritory via SalesTerritoryKey provides accurate sales territory information.



7. Data Optimization and Update

Power BI process is not static. As new data arrives, it's important to maintain consistency.
The example below shows a simple transaction (UPDATE) that can be used to simulate data updates in the database, followed by a check.

View Code

SQLQuery2.sql - L...M4NC\samsung (71))*

```sql
BEGIN TRANSACTION T1

    UPDATE FactInternetSales
    SET OrderQuantity = 20
    WHERE ProductKey = 361        -- Bike Category

COMMIT TRANSACTION T1

SELECT * FROM FactInternetSales
```

This code updates the quantity sold for a specific product (with ProductKey = 361) to 20. The use of BEGIN TRANSACTION and COMMIT TRANSACTION ensures that the operation is executed atomically (either everything happens, or nothing happens).

Next Steps

With the views created in Microsoft SQL Server the next step is to connect Power BI to the database and import the RESULTS_ADW view. From there, it will be possible to create the necessary visualizations, charts, and slicers to build the sales analysis dashboard, using the KPIs defined in section 3.