**Week 5: SQL and Power BI Report Development**

1. Create a table named ‘Product’ based on the ‘DimProduct’, ‘DimProductCategory’, and ‘DimProdustSubcategory’ tables.

SQL:

-- Create the Product table

CREATE TABLE dbo.Product (

ProductKey INT PRIMARY KEY,

ProductName NVARCHAR(255),

Manufacturer NVARCHAR(255),

BrandName NVARCHAR(255),

ClassName NVARCHAR(255),

ColorName NVARCHAR(255),

Size NVARCHAR(255),

UnitCost MONEY,

UnitPrice MONEY,

ProductSubcategoryName NVARCHAR(255),

ProductCategoryName NVARCHAR(255)

);

-- Insert data into Product table

INSERT INTO dbo.Product (

ProductKey,

ProductName,

Manufacturer,

BrandName,

ClassName,

ColorName,

Size,

UnitCost,

UnitPrice,

ProductSubcategoryName,

ProductCategoryName

)

SELECT

p.ProductKey,

p.ProductName,

p.Manufacturer,

p.BrandName,

c.ClassName,

co.ColorName,

p.Size,

p.StandardCost AS UnitCost,

p.ListPrice AS UnitPrice,

ps.ProductSubcategoryName,

pc.ProductCategoryName

FROM ContosoRetailDW.dbo.DimProduct p

JOIN ContosoRetailDW.dbo.DimProductCategory pc

ON p.ProductCategoryKey = pc.ProductCategoryKey

JOIN ContosoRetailDW.dbo.DimProductSubcategory ps

ON p.ProductSubcategoryKey = ps.ProductSubcategoryKey

JOIN ContosoRetailDW.dbo.DimProductClass c

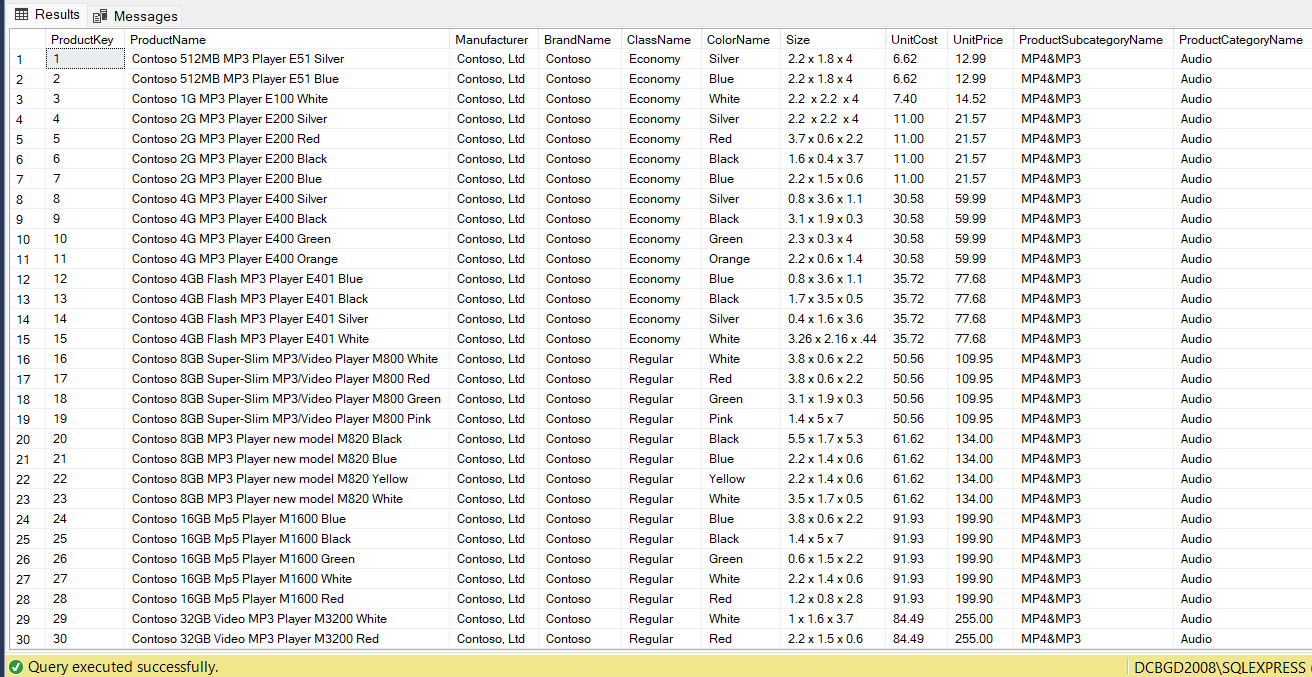
ON p.ProductClassKey = c.ProductClassKey

JOIN ContosoRetailDW.dbo.DimColor co

ON p.ColorKey = co.ColorKey;

GO

Screenshot:



1. Create a view named ‘Channel’ that should have the following columns.

* ChannelKey (Primary key)
* ChannelName

SQL:

-- Create the Channel view

CREATE VIEW dbo.Channel AS

SELECT

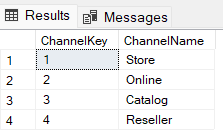
ChannelKey,

ChannelName

FROM ContosoRetailDW.dbo.DimChannel;

GO

select \* from Channel

Screenshot:

1. Crate a ‘Store’ table based on ‘DimStore’, ‘DimEntity’, ‘DimGeography’, ‘DimSalesTerritory’, and ‘DimEmployee’.

SQL:

-- Create the Store table

CREATE TABLE dbo.Store (

StoreKey INT PRIMARY KEY,

StoreType NVARCHAR(15),

StoreName NVARCHAR(100),

ContinentName NVARCHAR(50),

CityName NVARCHAR(100),

StateProvinceName NVARCHAR(100),

RegionCountryName NVARCHAR(100),

SalesTerritoryRegion NVARCHAR(50),

TerritoryManagerName NVARCHAR(100),

StoreManagerName NVARCHAR(100)

);

-- Insert data into Store table

INSERT INTO dbo.Store (

StoreKey,

StoreType,

StoreName,

ContinentName,

CityName,

StateProvinceName,

RegionCountryName,

SalesTerritoryRegion,

TerritoryManagerName,

StoreManagerName

)

SELECT

s.StoreKey,

s.StoreType,

s.StoreName,

g.ContinentName,

g.CityName,

g.StateProvinceName,

g.RegionCountryName,

t.SalesTerritoryRegion,

e.FirstName + ' ' + e.LastName AS TerritoryManagerName,

sm.FirstName + ' ' + sm.LastName AS StoreManagerName

FROM ContosoRetailDW.dbo.DimStore s

JOIN ContosoRetailDW.dbo.DimGeography g

ON s.GeographyKey = g.GeographyKey

JOIN ContosoRetailDW.dbo.DimSalesTerritory t

ON s.SalesTerritoryKey = t.SalesTerritoryKey

LEFT JOIN ContosoRetailDW.dbo.DimEmployee e

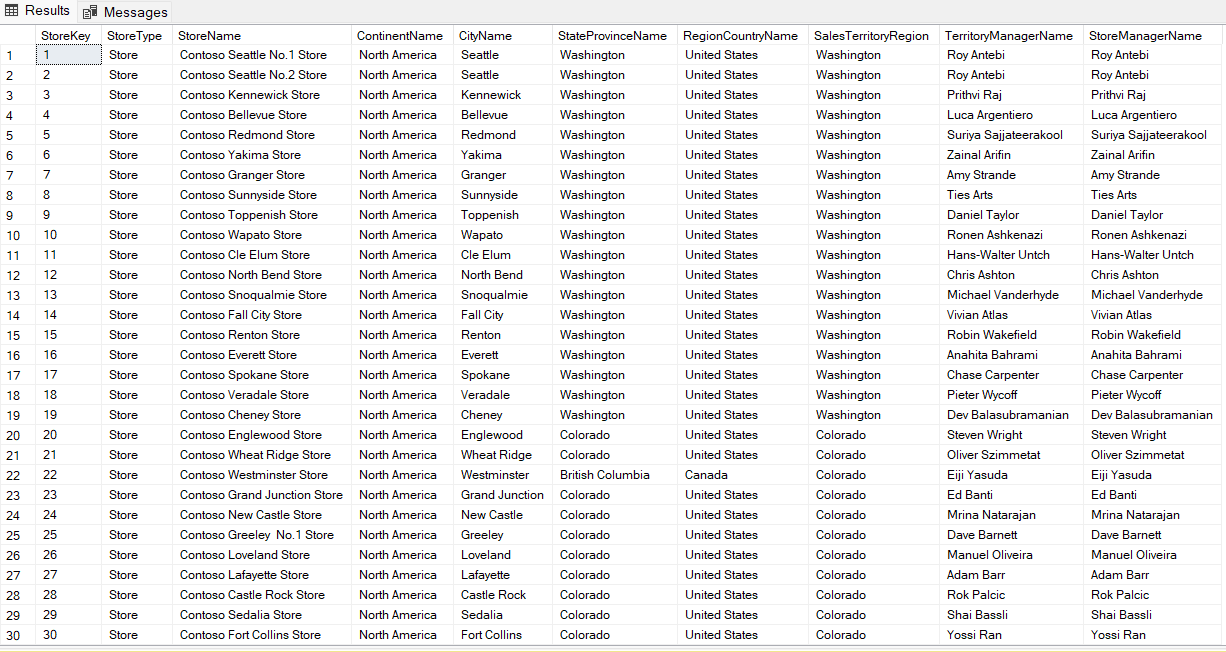
ON t.SalesTerritoryManager = e.EmployeeKey

LEFT JOIN ContosoRetailDW.dbo.DimEmployee sm

ON s.StoreManager = sm.EmployeeKey;

GO

Screenshot:



1. Create a ‘Date’ table

SQL:

-- Create the Date table

CREATE TABLE dbo.Date (

DateKey BIGINT PRIMARY KEY,

FullDateLabel NVARCHAR(20),

DateDescription NVARCHAR(20),

CalendarYear INT,

CalendarYearLabel NVARCHAR(20),

CalendarHalfYear INT,

CalendarHalfYearLabel NVARCHAR(20),

CalendarQuarter INT,

CalendarQuarterLabel NVARCHAR(20),

CalendarMonth INT,

CalendarMonthLabel NVARCHAR(20),

CalendarWeek INT,

CalendarWeekLabel NVARCHAR(20),

CalendarDayOfWeek INT,

CalendarDayOfWeekLabel NVARCHAR(10),

FiscalYear INT,

FiscalYearLabel NVARCHAR(20),

FiscalHalfYear INT,

FiscalHalfYearLabel NVARCHAR(20),

FiscalQuarter INT,

FiscalQuarterLabel NVARCHAR(20),

FiscalMonth INT,

FiscalMonthLabel NVARCHAR(20),

IsWorkDay NVARCHAR(20),

IsHoliday INT,

HolidayName NVARCHAR(20),

EuropeSeason NVARCHAR(50),

NorthAmericaSeason NVARCHAR(50),

AsiaSeason NVARCHAR(50)

);

-- Insert data into Date table

INSERT INTO dbo.Date (

DateKey,

FullDateLabel,

DateDescription,

CalendarYear,

CalendarYearLabel,

CalendarHalfYear,

CalendarHalfYearLabel,

CalendarQuarter,

CalendarQuarterLabel,

CalendarMonth,

CalendarMonthLabel,

CalendarWeek,

CalendarWeekLabel,

CalendarDayOfWeek,

CalendarDayOfWeekLabel,

FiscalYear,

FiscalYearLabel,

FiscalHalfYear,

FiscalHalfYearLabel,

FiscalQuarter,

FiscalQuarterLabel,

FiscalMonth,

FiscalMonthLabel,

IsWorkDay,

IsHoliday,

HolidayName,

EuropeSeason,

NorthAmericaSeason,

AsiaSeason

)

SELECT

CAST(REPLACE(FORMAT(d.DateKey, 'yyyyMMddHHmmss'), '-', '') AS BIGINT) AS DateKey,

d.FullDateLabel,

d.DateDescription,

d.CalendarYear,

d.CalendarYearLabel,

d.CalendarHalfYear,

d.CalendarHalfYearLabel,

d.CalendarQuarter,

d.CalendarQuarterLabel,

d.CalendarMonth,

d.CalendarMonthLabel,

d.CalendarWeek,

d.CalendarWeekLabel,

d.CalendarDayOfWeek,

d.CalendarDayOfWeekLabel,

d.FiscalYear,

d.FiscalYearLabel,

d.FiscalHalfYear,

d.FiscalHalfYearLabel,

d.FiscalQuarter,

d.FiscalQuarterLabel,

d.FiscalMonth,

d.FiscalMonthLabel,

d.IsWorkDay,

d.IsHoliday,

d.HolidayName,

d.EuropeSeason,

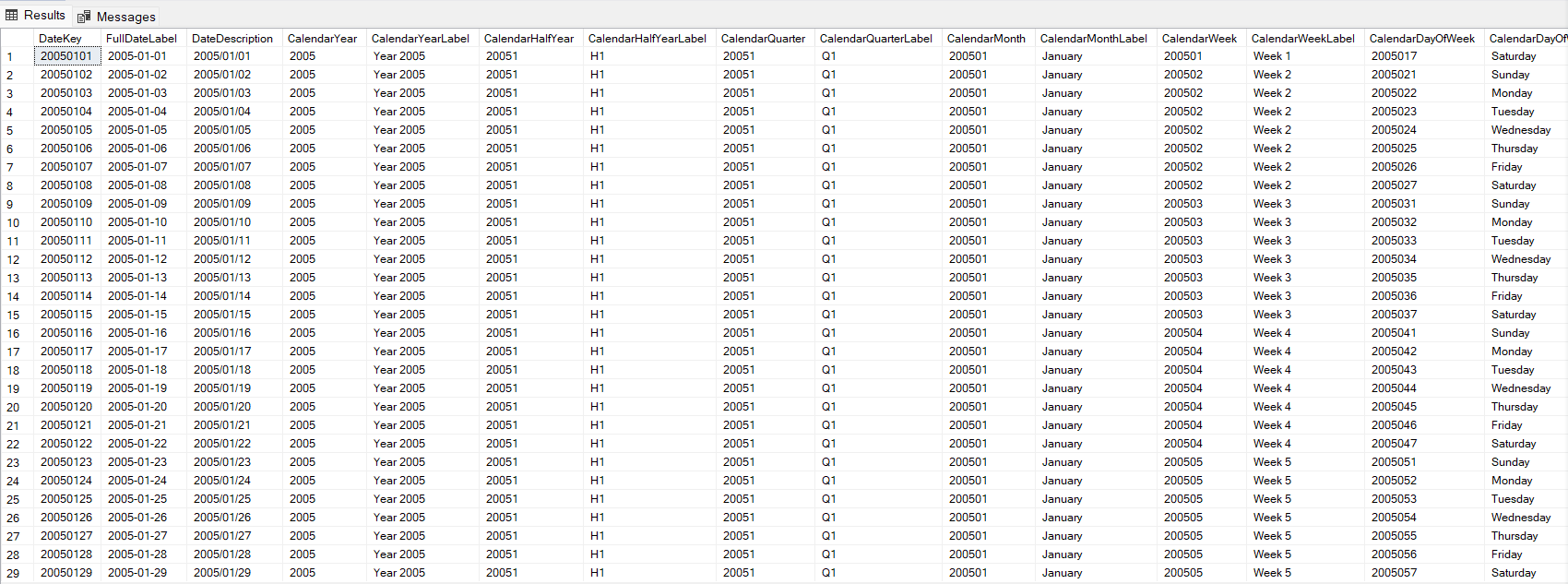
d.NorthAmericaSeason,

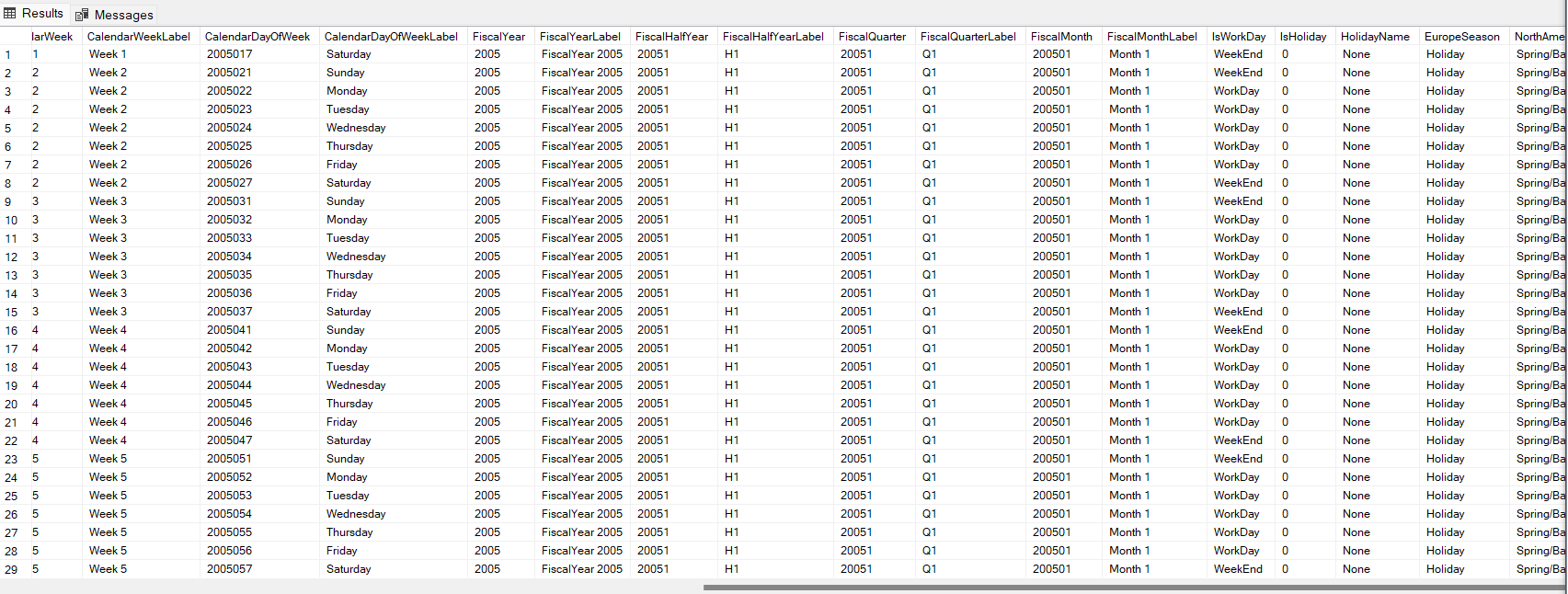
d.AsiaSeason

FROM ContosoRetailDW.dbo.DimDate d;

GO

Screenshot:





1. Create a ‘Budget’ table based on ‘FactSalesQuota’ and ‘DimScenario’ with the following columns.

SQL:

USE UpdateContosoRetailDW;

-- Drop the existing Budget table if it exists

IF OBJECT\_ID('dbo.Budget', 'U') IS NOT NULL

DROP TABLE dbo.Budget;

-- Create the Budget table with combined foreign key references

CREATE TABLE dbo.Budget (

SalesQuotaKey INT PRIMARY KEY NOT NULL,

ChannelKey INT ,

StoreKey INT FOREIGN KEY REFERENCES UpdateContosoRetailDW.dbo.Store(StoreKey),

ProductKey INT FOREIGN KEY REFERENCES UpdateContosoRetailDW.dbo.Product(ProductKey),

DateKey BIGINT FOREIGN KEY REFERENCES UpdateContosoRetailDW.dbo.Date(DateKey),

ScenarioKey INT ,

SalesQuantityQuota DECIMAL(18, 2) NULL,

SalesAmountQuota DECIMAL(18, 2) NULL,

GrossMarginQuota DECIMAL(18, 2) NULL

);

-- Insert data into the Budget table

INSERT INTO dbo.Budget (

SalesQuotaKey,

ChannelKey,

StoreKey,

ProductKey,

DateKey,

ScenarioKey,

SalesQuantityQuota,

SalesAmountQuota,

GrossMarginQuota

)

SELECT

FactSalesQuota.SalesQuotaKey,

FactSalesQuota.ChannelKey,

FactSalesQuota.StoreKey,

FactSalesQuota.ProductKey,

CAST(REPLACE(REPLACE(REPLACE(CONVERT(VARCHAR(50), FactSalesQuota.DateKey, 120), '-', ''), ':', ''), ' ', '') AS BIGINT) AS DateKey,

FactSalesQuota.ScenarioKey,

FactSalesQuota.SalesQuantityQuota,

FactSalesQuota.SalesAmountQuota,

FactSalesQuota.GrossMarginQuota

FROM

ContosoRetailDW.dbo.FactSalesQuota AS FactSalesQuota

JOIN

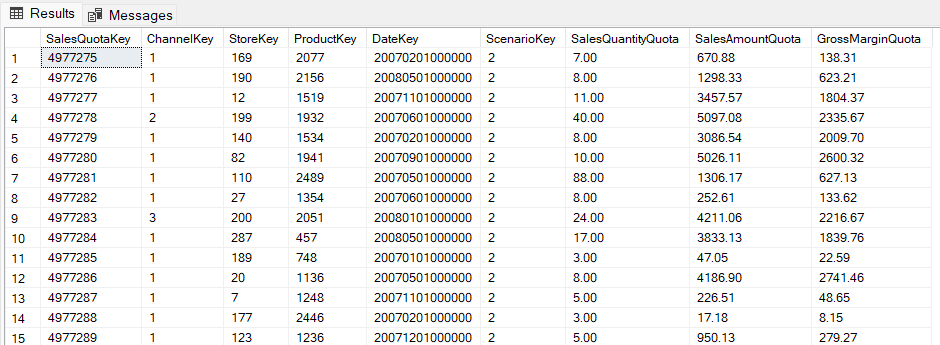
ContosoRetailDW.dbo.DimScenario AS DimScenario

ON FactSalesQuota.ScenarioKey = DimScenario.ScenarioKey

WHERE

DimScenario.ScenarioDescription = 'Budget';

Select \* from Budget

Screenshot:

1. Create a ‘Sales’ table with the following columns. (Note: You should remove spaces, and any special characters from the existing DateKey column. For example, convert ‘2005-01-01 00:00:00.000’ to ‘20050101000000000’.)

SQL:

USE UpdateContosoRetailDW;

-- Drop the existing Sales table if it exists

IF OBJECT\_ID('dbo.Sales', 'U') IS NOT NULL

DROP TABLE dbo.Sales;

-- Create the Sales table with foreign key references

CREATE TABLE dbo.Sales (

SalesKey INT PRIMARY KEY NOT NULL,

DateKey BIGINT NOT NULL FOREIGN KEY REFERENCES UpdateContosoRetailDW.dbo.Date(DateKey),

ChannelKey INT NOT NULL, -- Added ChannelKey as a regular column

StoreKey INT NOT NULL FOREIGN KEY REFERENCES UpdateContosoRetailDW.dbo.Store(StoreKey),

ProductKey INT NOT NULL FOREIGN KEY REFERENCES UpdateContosoRetailDW.dbo.Product(ProductKey),

UnitCost DECIMAL(18, 2) NULL,

UnitPrice DECIMAL(18, 2) NULL,

SalesQuantity INT NULL,

ReturnQuantity INT NULL,

ReturnAmount DECIMAL(18, 2) NULL,

DiscountQuantity INT NULL,

DiscountAmount DECIMAL(18, 2) NULL,

TotalCost DECIMAL(18, 2) NULL,

SalesAmount DECIMAL(18, 2) NULL,

);

-- Insert data into the Sales table

INSERT INTO dbo.Sales (

SalesKey,

DateKey,

ChannelKey,

StoreKey,

ProductKey,

UnitCost,

UnitPrice,

SalesQuantity,

ReturnQuantity,

ReturnAmount,

DiscountQuantity,

DiscountAmount,

TotalCost,

SalesAmount

)

SELECT

FactSales.SalesKey,

CAST(REPLACE(REPLACE(REPLACE(CONVERT(VARCHAR(50), FactSales.DateKey, 120), '-', ''), ':', ''), ' ', '') AS BIGINT) AS DateKey,

FactSales.ChannelKey, -- ChannelKey in the FactSales table

FactSales.StoreKey,

FactSales.ProductKey,

FactSales.UnitCost,

FactSales.UnitPrice,

FactSales.SalesQuantity,

FactSales.ReturnQuantity,

FactSales.ReturnAmount,

FactSales.DiscountQuantity,

FactSales.DiscountAmount,

FactSales.TotalCost,

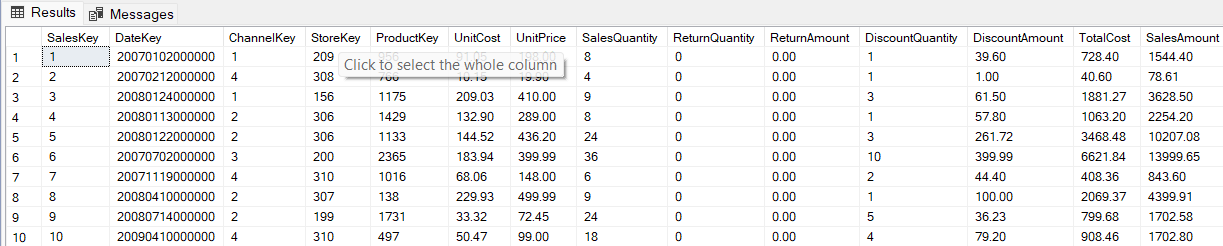
FactSales.SalesAmount

FROM

ContosoRetailDW.dbo.FactSales AS FactSales

-- Select all data from the Sales table

select \* from Sales

Screenshot:

1. You are recommended to find the answers to the following questions with help of both SQL and Power BI Desktop.
2. Find the channel-wise sales variance in descending order.

Note: Sales variance = ((Sales – Budget)/Budget) \* 100

SQL:

-- Calculate channel-wise sales variance

SELECT

c.ChannelKey,

c.ChannelName,

SUM(s.SalesAmount) AS TotalSales,

SUM(b.SalesAmountQuota) AS TotalBudget,

(CAST((SUM(s.SalesAmount) - SUM(b.SalesAmountQuota)) AS FLOAT) / SUM(b.SalesAmountQuota)) \* 100 AS SalesVariance

FROM

Sales AS s

JOIN

Budget AS b

ON s.ChannelKey = b.ChannelKey

AND s.DateKey = b.DateKey

AND s.StoreKey = b.StoreKey

AND s.ProductKey = b.ProductKey

JOIN

Channel AS c

ON s.ChannelKey = c.ChannelKey

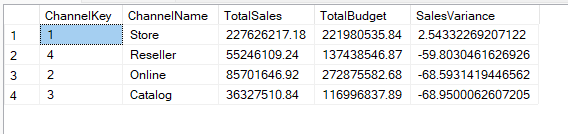
GROUP BY

c.ChannelKey, c.ChannelName

ORDER BY

SalesVariance DESC;

Screenshot:



1. Find product wise total sales in ascending order where sales are greater than 10000

SQL:

-- Find product-wise total sales in ascending order where sales are greater than 10,000

SELECT

p.ProductName,

SUM(s.SalesAmount) AS TotalSales

FROM

Sales AS s

JOIN

Product AS p

ON s.ProductKey = p.ProductKey

GROUP BY

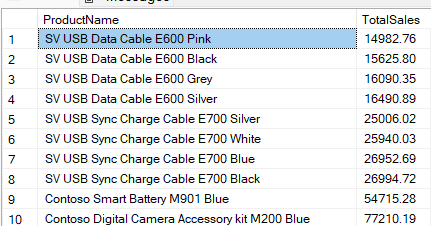
p.ProductKey, p.ProductName

HAVING

SUM(s.SalesAmount) > 10000

ORDER BY

TotalSales ASC;

Screenshot:

1. Create a function that returns the lowest performing channel

SQL:

ALTER FUNCTION dbo.GetLowestPerformingChannelByProfit()

RETURNS TABLE

AS

RETURN

(

SELECT TOP 1

c.ChannelKey,

c.ChannelName,

SUM(s.SalesAmount - s.TotalCost) AS TotalProfit

FROM

Sales AS s

JOIN

Channel AS c

ON s.ChannelKey = c.ChannelKey

GROUP BY

c.ChannelKey, c.ChannelName

ORDER BY

TotalProfit ASC

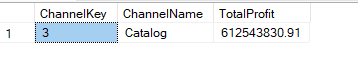
);

GO

SELECT \*

FROM dbo.GetLowestPerformingChannelByProfit();

Screenshot:



1. Which product has been returned most frequently?

SQL:

-- Find the product that has been returned most frequently

SELECT TOP 1

p.ProductName,

SUM(s.ReturnQuantity) AS TotalReturns

FROM

Sales AS s

JOIN

Product AS p

ON s.ProductKey = p.ProductKey

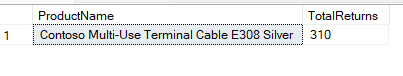
GROUP BY

p.ProductKey, p.ProductName

ORDER BY

TotalReturns DESC;

Screenshot:



1. Find the store where the sales variance is maximum?

SQL:

-- Find the store with the maximum sales variance

SELECT TOP 1

st.StoreKey,

st.StoreName,

SUM(s.SalesAmount) AS TotalSales,

SUM(b.SalesAmountQuota) AS TotalBudget,

(CAST((SUM(s.SalesAmount) - SUM(b.SalesAmountQuota)) AS FLOAT) / SUM(b.SalesAmountQuota)) \* 100 AS SalesVariance

FROM

Sales AS s

JOIN

Budget AS b

ON s.StoreKey = b.StoreKey

AND s.DateKey = b.DateKey

AND s.ProductKey = b.ProductKey

AND s.ChannelKey = b.ChannelKey

JOIN

Store AS st

ON s.StoreKey = st.StoreKey

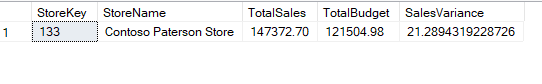
GROUP BY

st.StoreKey, st.StoreName

ORDER BY

SalesVariance DESC;

Screenshot:



1. What is the highest selling product, in terms of sales amount?

SQL:

-- Find the highest-selling product in terms of sales amount

SELECT TOP 1

p.ProductKey,

p.ProductName,

SUM(s.SalesAmount) AS TotalSalesAmount

FROM

Sales AS s

JOIN

Product AS p

ON s.ProductKey = p.ProductKey

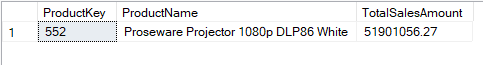
GROUP BY

p.ProductKey, p.ProductName

ORDER BY

TotalSalesAmount DESC;

Screenshot:



1. Create a function that returns the top performing product category

SQL:

-- Drop the function if it already exists

IF OBJECT\_ID('dbo.GetTopPerformingProductCategoryByProfit', 'IF') IS NOT NULL

DROP FUNCTION dbo.GetTopPerformingProductCategoryByProfit;

GO

-- Create the function

CREATE FUNCTION dbo.GetTopPerformingProductCategoryByProfit()

RETURNS TABLE

AS

RETURN

(

SELECT TOP 1

p.ProductCategoryName AS ProductCategory,

SUM(s.SalesAmount - s.TotalCost) AS TotalProfit

FROM

dbo.Sales AS s

JOIN

dbo.Product AS p

ON s.ProductKey = p.ProductKey

GROUP BY

p.ProductCategoryName

ORDER BY

TotalProfit DESC

);

GO

-- Call the function

SELECT \* FROM dbo.GetTopPerformingProductCategoryByProfit();

Screenshot:



1. Which North American store generated the highest revenue?

SQL:

-- Find the North American store with the highest revenue

SELECT TOP 1

st.StoreKey,

st.StoreName,

st.ContinentName,

st.RegionCountryName,

SUM(s.SalesAmount) AS TotalRevenue

FROM

dbo.Store AS st

JOIN

dbo.Sales AS s

ON st.StoreKey = s.StoreKey

WHERE

st.ContinentName = 'North America'

GROUP BY

st.StoreKey, st.StoreName, st.ContinentName, st.RegionCountryName

ORDER BY

TotalRevenue DESC;

select \* from Store

Screenshot:



1. What is the highest selling product subcategory in terms of quantity?

SQL:

-- Find the highest-selling product subcategory in terms of quantity

SELECT TOP 1

p.ProductSubcategoryName,

SUM(s.SalesQuantity) AS TotalQuantitySold

FROM

dbo.Product AS p

JOIN

dbo.Sales AS s

ON p.ProductKey = s.ProductKey

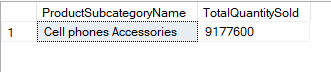
GROUP BY

p.ProductSubcategoryName

ORDER BY

TotalQuantitySold DESC;

Screenshot:



1. Beijing is the city where total sales are maximum. True or False. If false, which city has the maximum total sales?

SQL:

-- Find the city with the maximum total sales and check if it's Beijing

SELECT TOP 1

CASE

WHEN CityName = 'Beijing' THEN 'True'

ELSE 'False'

END AS IsBeijingMaxSales,

CityName,

SUM(SalesAmount) AS TotalSales

FROM dbo.Store s

JOIN dbo.Sales sa

ON s.StoreKey = sa.StoreKey

GROUP BY CityName

ORDER BY TotalSales DESC;

select \* from Store

Screenshot:

