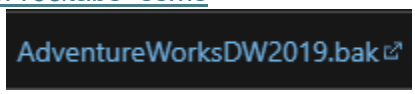
Project:

AdventureWorks2019 Internet Sales Analytics

- Project Aim: Explore online sales data using the AdventureWorks2019 dataset, extracting valuable insights to boost performance and refine strategic decisions.

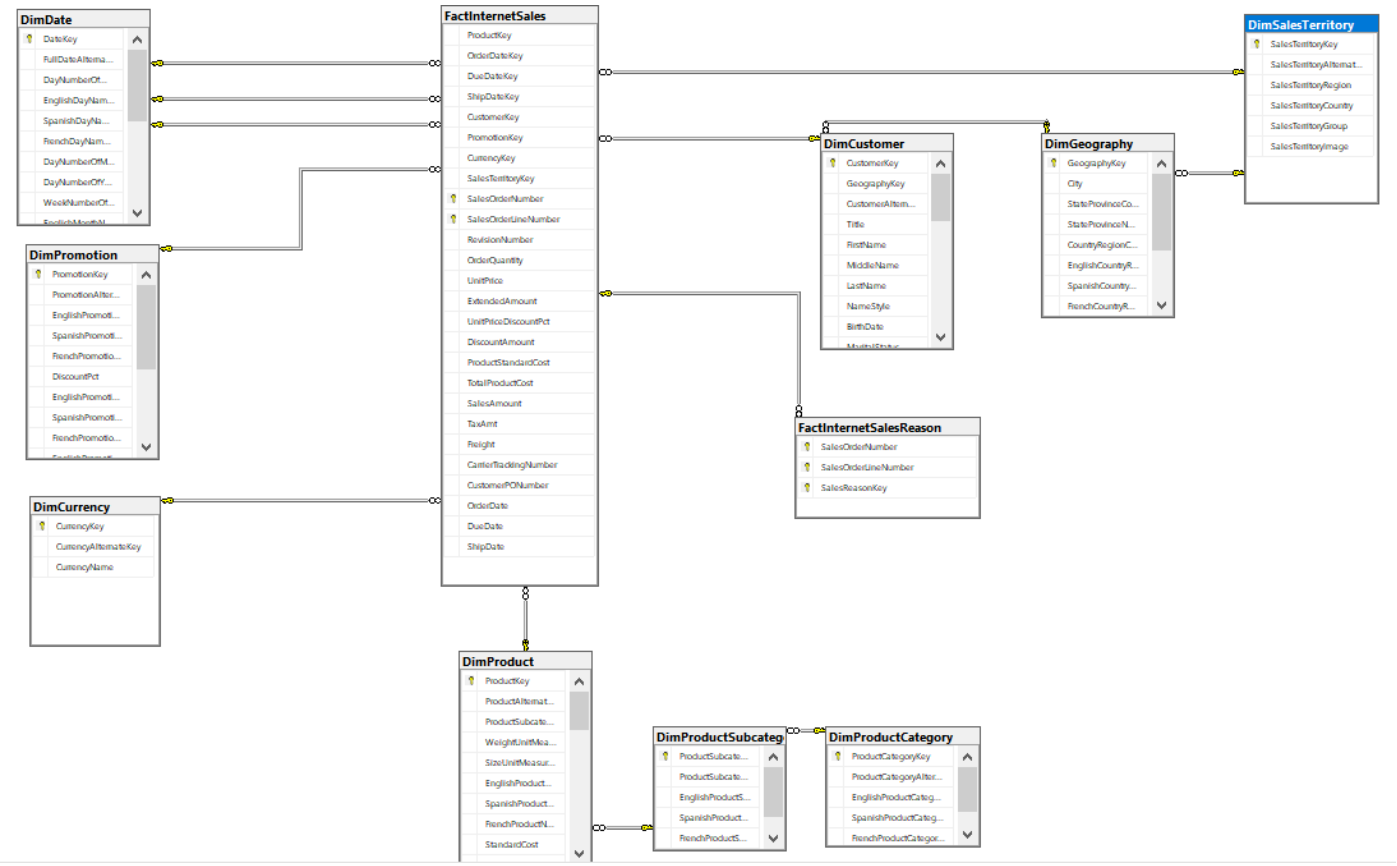
- Data source: https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms



Contents

1. Data Cleaning

- The purpose of this project is to analyze the internet sales performance so the focus is on the FactInternetSales table. There are many unnecessary tables and columns in *AdventureWorksDW2019* database and we consider this as a source so we need a new database for analytics and connecting to Power BI. First, we will query each table and see which tables, columns to keep and what to add or transform. This section results a new database schema and subqueries. The new database schema is for understanding tables structure and the relationships between tables. And the subqueries will be used for inserting data to the new database.



a) Dimension Date table:

- Leave out name of month, week in other languages rather than English.

```
--Dimension Date table
SELECT [DateKey]
      ,[FullDateAlternateKey] AS Date
      ,[DayNumberOfWeek] AS Week_day_number
      ,[EnglishDayNameOfWeek] AS Week_day
      --,[SpanishDayNameOfWeek]...
      ,[EnglishMonthName] AS Month_name
      --,[SpanishMonthName]...
      ,[CalendarQuarter] AS Quarter
      --,[CalendarYear] ...
  FROM [AdventureWorksDW2019].[dbo].[DimDate]
```

- Result:

Results | Messages

	DateKey	Date	Week_day_number	Week_day	Month_name	Quarter
1	20050101	2005-01-01	7	Saturday	January	1
2	20050102	2005-01-02	1	Sunday	January	1
3	20050103	2005-01-03	2	Monday	January	1
4	20050104	2005-01-04	3	Tuesday	January	1

00:00:00 | 3,652 rows

b) Customer Dimension table:

- All emails has @adventure-works.com domain

```
--check email domain
SELECT EmailAddress
FROM [AdventureWorksDW2019].[dbo].[DimCustomer]
WHERE SUBSTRING(EmailAddress, CHARINDEX('@', EmailAddress) + 1,
                LEN(EmailAddress) - CHARINDEX('@', EmailAddress)) NOT LIKE '%adventure-works.com%'
```

EmailAddress

- Leave out unnecessary columns: Since we'll focus on general information about customers like geography, incomes, gender,.. So we'll leave out information like name, specific address, phone, email,…
- Transform values:
  - Convert values of columns 'MaritalStatus, Gender, HouseOwnerFlag' for a clearer understanding

```
,CASE WHEN MaritalStatus = 'M' THEN 'Married'
      WHEN MaritalStatus = 'S' THEN 'Single'
      END AS MaritalStatus
 ,CASE WHEN [Gender] = 'M' THEN 'Male'
       WHEN Gender = 'F' THEN 'Female'
       END AS Gender
 ,CASE WHEN [HouseOwnerFlag] = 0 THEN 'No'
       WHEN HouseOwnerFlag = 1 THEN 'Yes'
       END AS Is_house_owner
```

  - Convert numerical values to categorical values for better segmentation.

```
 ,CASE WHEN NumberCarsOwned = 0 THEN 'No'
       WHEN NumberCarsOwned > 0 THEN 'Yes'
       END AS Is_car_owner
 ,CASE WHEN [TotalChildren] > 0 THEN 'Yes'
       WHEN TotalChildren = 0 THEN 'No'
       END AS Is_parent
```

- Result:

Results | Messages

	CustomerKey	GeographyKey	BirthDate	MaritalStatus	Gender	YearlyIncome	Is_parent	Education_level	Job	Is_house_owner	Is_car_owner	DateFirstPurchase
1	11000	26	1971-10-06	Married	Male	90000.00	Yes	Bachelors	Professional	Yes	No	2011-01-19
2	11001	37	1976-05-10	Single	Male	60000.00	Yes	Bachelors	Professional	No	Yes	2011-01-15
3	11002	31	1971-02-09	Married	Male	60000.00	Yes	Bachelors	Professional	Yes	Yes	2011-01-07
4	11003	11	1973-08-14	Single	Female	70000.00	No	Bachelors	Professional	No	Yes	2010-12-29

Query executed successfully. | master | 00:00:00 | 18,484 rows

c) Dimension Product Category table:

- Leave out columns: unnecessary columns like alternative key (same values as primary key) and category's names in other languages.

```
--DIMENSION PRODUCT CATEGORY TABLE
SELECT [ProductCategoryKey]
    --,[ProductCategoryAlternateKey]
    ,[EnglishProductCategoryName] AS CategoryName
    --,[SpanishProductCategoryName]
    --,[FrenchProductCategoryName]
FROM [AdventureWorksDW2019].[dbo].[DimProductCategory];
```

- Result:

Results | Messages

	ProductCategoryKey	CategoryName
1	1	Bikes
2	2	Components
3	3	Clothing
4	4	Accessories

[redacted] | 00:00:00 | 4 rows

d) Dimension Product Subcategory table:

- Leave out columns: unnecessary columns like alternative key (same values as primary key) and category's names in other languages.

```
--DIMENSION PRODUCT SUB CATEGORY TABLE
SELECT [ProductSubcategoryKey]
    --,[ProductSubcategoryAlternateKey]
    ,[EnglishProductSubcategoryName] AS SubcategoryName
    --,[SpanishProductSubcategoryName]
    --,[FrenchProductSubcategoryName]
    ,[ProductCategoryKey]
FROM [AdventureWorksDW2019].[dbo].[DimProductSubcategory]
```

- Result:

Results | Messages

	ProductSubcategoryKey	SubcategoryName	ProductCategoryKey
1	1	Mountain Bikes	1
2	2	Road Bikes	1
3	3	Touring Bikes	1
4	4	Handlebars	2
5	5	Bottom Brackets	2

[redacted] | 00:00:00 | 37 rows

e) Currency Dimension table:

- Since there are different currencies, the exchange rate is different for each currency, so we'll create a exchange rate to USD to compare the sales, profits easier. And get only the currency that relates to sales fact table.

```sql
--DIMENSION CURRENCY TABLE
SELECT DISTINCT c.CurrencyKey,
                CurrencyAlternateKey AS CurrencyAbbreviation,
                CurrencyName,
       CASE WHEN CurrencyAlternateKey = 'DEM' THEN 0.56
            WHEN CurrencyAlternateKey = 'AUD' THEN 0.68
            WHEN CurrencyAlternateKey = 'GBP' THEN 1.32
            WHEN CurrencyAlternateKey = 'CAD' THEN 0.73
            WHEN CurrencyAlternateKey = 'FRF' THEN 0.16
            WHEN CurrencyAlternateKey = 'USD' THEN 1
            END AS To_USD_Rate
FROM AdventureWorksDW2019.dbo.DimCurrency c
RIGHT JOIN AdventureWorksDW2019.dbo.FactInternetSales s
ON c.CurrencyKey = s.CurrencyKey;
```

- Result:

Results | Messages

	CurrencyKey	CurrencyAbbreviation	CurrencyName	To_USD_Rate
1	100	USD	US Dollar	1.00
2	29	DEM	Deutsche Mark	0.56
3	6	AUD	Australian Dollar	0.68
4	39	FRF	French Franc	0.16
5	98	GBP	United Kingdom Pound	1.32
6	19	CAD	Canadian Dollar	0.73

00:00:00 | 6 rows

f) Dimension Product table:

- The main focus are on the price, cost and the product's category
- Leave out some columns that are specific characteristics of the product like size, color, reorderPoint,...
  - The products in sales are only finished as well so I will also leave out FinishedGoodsFlag column

```sql
--check finished goods product
SELECT COUNT(DISTINCT p.ProductKey) AS in_sales_product,
       (SELECT COUNT(DISTINCT p.ProductKey)
          FROM [AdventureWorksDW2019].[dbo].[DimProduct] p
          JOIN AdventureWorksDW2019.dbo.FactInternetSales s
          ON p.ProductKey = s.ProductKey) AS finished_goods
FROM [AdventureWorksDW2019].[dbo].[DimProduct] p
JOIN AdventureWorksDW2019.dbo.FactInternetSales s
ON p.ProductKey = s.ProductKey
WHERE FinishedGoodsFlag <> 0;
```

Results | Messages

	in_sales_product	finished_goods
1	158	158

  - StartDate and EndDate don't really make sense since all startDate > endDate. So column EndDate won't be chosen.

```sql
-- check start, end date
SELECT StartDate, EndDate
FROM AdventureWorksDW2019.dbo.DimProduct
WHERE StartDate < EndDate
```

StartDate	EndDate

- Transform NULL value of column status from NULL to Outdated since these NULL values indicate that the expire date of the product runs out.

```sql
,CASE WHEN [Status] IS NULL THEN 'Outdated'
      ELSE Status
      END AS Status
```

- Result:

Results | Messages

	ProductKey	ProductSubcategoryKey	ProductName	StandardCost	ListPrice	DealerPrice	ModelName	StartDate	EndDate	Status
1	214	31	Sport-100 Helmet, Red	13.0863	34.99	20.994	Sport-100	2013-07-01 00:00:00.000	NULL	Current
2	217	31	Sport-100 Helmet, Black	13.0863	34.99	20.994	Sport-100	2013-07-01 00:00:00.000	NULL	Current
3	222	31	Sport-100 Helmet, Blue	13.0863	34.99	20.994	Sport-100	2013-07-01 00:00:00.000	NULL	Current
4	225	19	AWC Logo Cap	6.9223	8.99	5.394	Cycling Cap	2013-07-01 00:00:00.000	NULL	Current
5	228	21	Long-Sleeve Logo Jersey, S	38.4923	49.99	29.994	Long-Sleeve Logo Jersey	2013-07-01 00:00:00.000	NULL	Current

Query executed successfully. [redacted] 00:00:00 | 158 rows

g) Dimension Geography table:

- Leave out columns like name in other languages, location code, IP address

```sql
--DIMENSION Geography TABLE
SELECT [GeographyKey]
    ,[City]
    --,[StateProvinceCode]
    ,[StateProvinceName]
    ,[CountryRegionCode]
    ,[EnglishCountryRegionName] AS CountryRegionName
    --,[SpanishCountryRegionName]
    --,[FrenchCountryRegionName]
    --,[PostalCode]
    ,[SalesTerritoryKey]
    --,[IpAddressLocator]
FROM [AdventureWorksDW2019].[dbo].[DimGeography];
```

- Result:

Results | Messages

	GeographyKey	City	StateProvinceName	CountryRegionCode	CountryRegionName	SalesTerritoryKey
1	1	Alexandria	New South Wales	AU	Australia	9
2	2	Coffs Harbour	New South Wales	AU	Australia	9
3	3	Darlinghurst	New South Wales	AU	Australia	9
4	4	Goulburn	New South Wales	AU	Australia	9

Query executed su... [redacted] 00:00:00 | 655 rows

h) Dimension Sales Territory Table:

- Leave out column that is unrelated like Image, column that is duplicated to primary key like SalesTerritoryAlternateKey
- Clear missing row

Results | Messages

	SalesTerritoryKey	SalesTerritoryAlternateKey	SalesTerritoryRegion	SalesTerritoryCountry	SalesTerritoryGroup
8	8	8	Germany	Germany	Europe
9	9	9	Australia	Australia	Pacific
10	10	10	United Kingdom	United Kingdom	Europe
11	11	0	NA	NA	NA

```
--DIMENSION SALES TERRITORY TABLE
SELECT [SalesTerritoryKey]
    --,[SalesTerritoryAlternateKey]
    ,[SalesTerritoryRegion] AS Region
    ,[SalesTerritoryCountry] AS Country
    ,[SalesTerritoryGroup] AS [Group]
    --,[SalesTerritoryImage]
FROM [AdventureWorksDW2019].[dbo].[DimSalesTerritory]
WHERE SalesTerritoryRegion !=  'NA';
```

- Result:

Results | Messages

	SalesTerritoryKey	Region	Country	Group
1	1	Northwest	United States	North America
2	2	Northeast	United States	North America
3	3	Central	United States	North America
4	4	Southw...	United States	North America

00:00:00 | 10 rows

i) Fact Internet Sales table:

- Leave out columns:
  - All the observations have quantity = 0 and discount = 0 so I don't choose these columns and the unnecessary columns to product's sales performance.

```
SELECT PromotionKey, DiscountAmount
FROM AdventureWorksDW2019.dbo.FactInternetSales
WHERE DiscountAmount <> 0;
```

109 %

Results | Messages

PromotionKey	DiscountAmount

  - ➔ So the table DimPromotion and FactInternetSalesReason will be dropped due to their lack of meaning.
- Add column:
  - profit = sales amount – total cost

```
SalesAmount - TotalProductCost AS Profit
```

  - profit margin = (sales amount - total cost) / sales amount
- Keep important column for measuring total sales and total costs. And keys to reference information from other dimensions tables.
- This table primary keys are: [SalesOrderNumber] ,[SalesOrderLineNumber]
- Result: This is the largest table with 60,398 rows and 12 columns. Remember that there are 6 currencies so we'll add cost, sales, profit columns with USD unit in Power BI later.

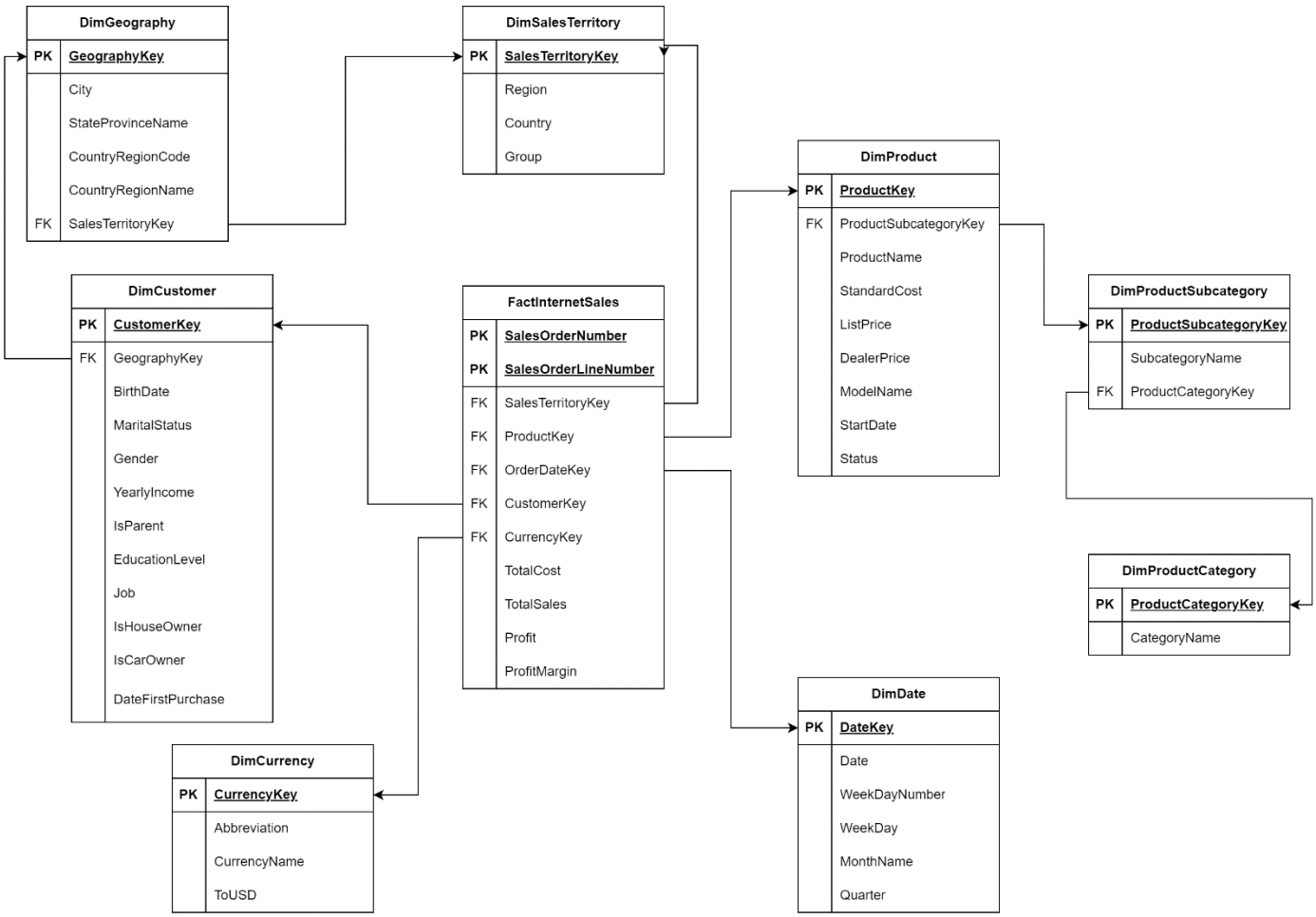
Results | Messages

	ProductKey	OrderDateKey	CustomerKey	PromotionKey	CurrencyKey	SalesTerritoryKey	SalesOrderNumber	SalesOrderLineNumber	Total_cost	Total_sales	Profit	Profit_margin
1	310	20101229	21768	1	19	6	SO43697	1	2171.2942	3578.27	1406.9758	0.3932
2	346	20101229	28389	1	39	7	SO43698	1	1912.1544	3399.99	1487.8356	0.4375
3	346	20101229	25863	1	100	1	SO43699	1	1912.1544	3399.99	1487.8356	0.4375
4	336	20101229	14501	1	100	4	SO43700	1	413.1463	699.0982	285.9519	0.409
5	346	20101229	11003	1	6	9	SO43701	1	1912.1544	3399.99	1487.8356	0.4375
6	311	20101230	27645	1	100	4	SO43702	1	2171.2942	3578.27	1406.9758	0.3932
7	310	20101230	16624	1	6	9	SO43703	1	2171.2942	3578.27	1406.9758	0.3932
8	351	20101230	11005	1	6	9	SO43704	1	1898.0944	3374.99	1476.8956	0.4375
9	344	20101230	11011	1	6	9	SO43705	1	1912.1544	3399.99	1487.8356	0.4375
10	312	20101231	27621	1	100	4	SO43706	1	2171.2942	3578.27	1406.9758	0.3932
11	312	20101231	27616	1	100	4	SO43707	1	2171.2942	3578.27	1406.9758	0.3932

Query executed successfully. [redacted] 00:00:00 60,398 rows

j) New database schema:

- Database schema of the new dimensional data store:



- The subqueries are stored in *cleanData.sql* file.

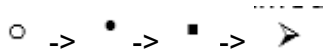
2. New Dimensional Database

a) Create tables

- From the new schema, we will create new tables, columns with the data type that fits the table of subquery.
- The creation sequence will be from the tables that are referenced and don't have foreign key(parent table). Then to the tables that are referenced and also have foreign key to other tables(parent table and also child table). And finally the Fact table, the table that only has foreign key but isn't referenced(child table).
    - Creation sequence structure:
        - DimDate
        - DimProductCategory
            - DimProductSubcategory

        - DimProduct
    - DimCurrency
    - DimSalesTerritory
        - DimGeography
            - DimCustomer
                - FactInternetsales

Note: The sequence order can be adjusted as long as fact table is created last and the order follows this point level:



- Example of creating product category and subcategory as parent table and child table:

```
-- Dimension product category table
IF NOT EXISTS (SELECT * FROM information_schema.tables WHERE table_schema = 'AdventureWorks2019_DDS'
                AND table_name = 'DimProductCategory')
    BEGIN
    CREATE TABLE DimProductCategory (
        ProductCategoryKey int NOT NULL PRIMARY KEY,
        CategoryName varchar(50),
    );
    END;
```

then

```
-- Dimension product subcategory table
IF NOT EXISTS (SELECT * FROM information_schema.tables WHERE table_schema = 'AdventureWorks2019_DDS'
                AND table_name = 'DimProductSubcategory')
    BEGIN
    CREATE TABLE DimProductSubcategory (
        ProductSubcategoryKey int NOT NULL PRIMARY KEY,
        ProductCategoryKey int FOREIGN KEY REFERENCES DimProductCategory(ProductCategoryKey),
        SubcategoryName varchar(50),
    );
    END;
```

- The code blocks is run sequently until the end of file *createTables.sql*

b) Insert data

- The running sequence is similar to table creation section.
- For inserting data, we have to check if the primary key(s) of the table exists before inserting. If it doesn't, then we'll insert data from the subquery(section 2.) of source database to new table. E.g:

```
--DIM PRODUCT CATEGORY
MERGE AdventureWorks2019_DDS.dbo.DimProductCategory AS dest
USING (
  SELECT [ProductCategoryKey]
      ,[EnglishProductCategoryName] AS CategoryName
  FROM [AdventureWorksDW2019].[dbo].[DimProductCategory]
) AS src
ON dest.[ProductCategoryKey] = src.[ProductCategoryKey]
WHEN NOT MATCHED BY target THEN
    INSERT ([ProductCategoryKey], CategoryName)
    VALUES (src.[ProductCategoryKey], src.CategoryName);
```

- File: *insertData.sql*

3. Data Exploratory Analysis

- Before analyzing, convert all money columns to match USD currency:

```
--convert money to usd (run once)
UPDATE  s
SET s.TotalCost = s.TotalCost * c.ToUSD,
    s.TotalSales = s.TotalSales * c.ToUSD,
    s.Profit = s.Profit * c.ToUSD
FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
JOIN AdventureWorks2019_DDS.dbo.DimCurrency c
ON s.CurrencyKey = c.CurrencyKey;
```

- Total sales:

```
-- total internet sales, total profit, total cost
SELECT SUM(TotalSales) AS Total_sales,
        SUM(Profit) AS Total_profit,
        SUM(TotalCost) AS Total_cost
FROM AdventureWorks2019_DDS.dbo.FactInternetSales
```

	Total_sales	Total_profit	Total_cost
1	26802228.49	11038817.13	15763402.05

- Average sales of order:

```
--average sales of an order
WITH sub AS(
    SELECT SUM(s.TotalSales) AS mean_sales
    FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
    GROUP BY s.SalesOrderNumber
)
SELECT AVG(mean_sales) AS avg_order_sales
FROM sub;
```

	avg_order_sales
1	969.023771285987

- Sales, profit performance by product category:

```
--sales by product category
SELECT c.CategoryName, ROUND(SUM(s.TotalSales), 2) AS total_sales,
        ROUND(SUM(s.Profit), 2) AS total_profit,
        ROUND(AVG(s.ProfitMargin), 2) AS mean_profit_margin
FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
JOIN AdventureWorks2019_DDS.dbo.DimProduct p
ON s.ProductKey = p.ProductKey
JOIN AdventureWorks2019_DDS.dbo.DimProductSubcategory sc
ON p.ProductSubcategoryKey = sc.ProductSubcategoryKey
JOIN AdventureWorks2019_DDS.dbo.DimProductCategory c
ON sc.ProductCategoryKey = c.ProductCategoryKey
GROUP BY c.CategoryName
ORDER BY total_sales DESC, total_profit DESC;
```

	CategoryName	total_sales	total_profit	mean_profit_margin
1	Bikes	25831624.76	10502621.34	0.4
2	Accessories	655793.96	410473.3	0.63
3	Clothing	314809.77	125722.49	0.39

- Bike category brings the most profit even with almost lowest profit margin. And even though Accessories has the highest profit margin, it's much smaller than Bikes because of the difference in price, but it's still 4 times better in total profit than Clothing.

- Sales by product subcategory:

```
--sales by product subcategory
SELECT c.CategoryName, sc.SubcategoryName, ROUND(SUM(s.TotalSales), 2) AS total_sales,
        ROUND(SUM(profit), 2) AS total_profit,
        ROUND(AVG(s.profitMargin), 2) AS mean_profit_margin
FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
JOIN AdventureWorks2019_DDS.dbo.DimProduct p
ON s.ProductKey = p.ProductKey
JOIN AdventureWorks2019_DDS.dbo.DimProductSubcategory sc
ON p.ProductSubcategoryKey = sc.ProductSubcategoryKey
JOIN AdventureWorks2019_DDS.dbo.DimProductCategory c
ON sc.ProductCategoryKey = c.ProductCategoryKey
GROUP BY c.CategoryName, sc.SubcategoryName
ORDER BY c.CategoryName, total_sales DESC;
```

	CategoryName	SubcategoryName	total_sales	total_profit	mean_profit_margin
1	Accessories	Tires and Tubes	229381.52	143572.61	0.63
2	Accessories	Helmets	213166.8	133423.16	0.63
3	Accessories	Bottles and Cages	53402.43	33418.25	0.63
4	Accessories	Fenders	43182.27	27030.44	0.63
5	Accessories	Bike Racks	36777.6	23022.88	0.63
6	Accessories	Bike Stands	36674.94	22957.67	0.63
7	Accessories	Hydration Packs	36624.32	22926.29	0.63
8	Accessories	Cleaners	6584.08	4122	0.63
9	Bikes	Road Bikes	13010267.81	4958206.76	0.38
10	Bikes	Mountain Bikes	9200573.61	4174310.15	0.45
11	Bikes	Touring Bikes	3620783.34	1370104.43	0.38
12	Clothing	Jerseys	161293.82	37099.26	0.23
13	Clothing	Shorts	65007.91	40691.68	0.63
14	Clothing	Vests	32789.7	20526.32	0.63
15	Clothing	Gloves	32057.38	20065.97	0.63
16	Clothing	Caps	18886.38	4348.86	0.23
17	Clothing	Socks	4774.58	2990.4	0.63

- Most subcategories from Accessories has equal mean profit margin = 0.63, but the product that brings the highest product sales and profit is 'Tires and Tubes'.
- For Bike subcategories, even though mean profit margin of Road Bikes and Touring Bikes have mean profit margin = 0.38, Road Bikes has total profit about 4 times better than Touring Bikes. And 'Road Bikes' is also the best selling subcategory.
- For Clothing, 'Shorts' is the most profitable subcategory even though 'Jerseys' is the best selling subcategory. Even though 'Socks' has a high profit margin, it doesn't have good sales result so it has the least sales and profit among all subcategories.

- Sales by quarter, year:

```sql
--sales by quarter, year
SELECT DATEPART(YEAR, d.[Date]) AS [year], DATEPART(QUARTER, d.[Date]) AS [quarter],
        SUM(s.TotalSales) AS total_sales,
        SUM(SUM(s.TotalSales)) OVER(PARTITION BY DATEPART(YEAR, d.[Date])) AS year_sales
FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
    JOIN AdventureWorks2019_DDS.dbo.DimDate d
    ON s.OrderDateKey = d.DateKey
GROUP BY DATEPART(YEAR, d.[Date]), DATEPART(QUARTER, d.[Date])
ORDER BY [year] , [quarter];
```

	year	quarter	total_sales	year_sales
1	2010	4	33131.48	33131.48
2	2011	1	1145208.72	6053495.43
3	2011	2	1466496.65	6053495.43
4	2011	3	1625064.78	6053495.43
5	2011	4	1816725.28	6053495.43
6	2012	1	1275080.34	5320142.91
7	2012	2	1200605.46	5320142.91
8	2012	3	1304816.3	5320142.91
9	2012	4	1539640.81	5320142.91
10	2013	1	2517857.37	15349763.95
11	2013	2	3742616.41	15349763.95
12	2013	3	4079968.73	15349763.95
13	2013	4	5009321.44	15349763.95
14	2014	1	45694.72	45694.72

- 2013 has the highest total sales with it's 4th quarter with 5 million $, almost equal to total sales of the whole year 2012. However, there is a sudden drop in 1st quarter of 2014, so we may look into this further.

- Sales by prodct category through quarters:

```sql
--sales by product category through quarters
WITH sub AS (
    SELECT p.ProductKey, c.CategoryName
    FROM AdventureWorks2019_DDS.dbo.DimProduct p
        JOIN AdventureWorks2019_DDS.dbo.DimProductSubcategory sc
        ON p.ProductSubcategoryKey = sc.ProductSubcategoryKey
        JOIN AdventureWorks2019_DDS.dbo.DimProductCategory c
        ON c.ProductCategoryKey = sc.ProductCategoryKey
)
SELECT DATEPART(YEAR, d.[Date]) AS [year], DATEPART(QUARTER, d.[Date]) AS [quarter],
        SUM(CASE WHEN sub.CategoryName = 'Bikes'
                THEN s.TotalSales ELSE 0 END) AS Bikes_Sales,
        SUM(CASE WHEN sub.CategoryName = 'Accessories'
                THEN s.TotalSales ELSE 0 END) AS Accessories_Sales,
        SUM(CASE WHEN sub.CategoryName = 'Clothing'
                THEN s.TotalSales ELSE 0 END) AS Clothing_Sales
FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
    JOIN AdventureWorks2019_DDS.dbo.DimDate d
    ON s.OrderDateKey = d.DateKey
    JOIN sub
    ON s.ProductKey = sub.ProductKey
GROUP BY DATEPART(YEAR, d.[Date]), DATEPART(QUARTER, d.[Date])
ORDER BY [year] , [quarter];
```

	year	quarter	Bikes_Sales	Accessories_Sales	Clothing_Sales
1	2010	4	33131.48	0	0
2	2011	1	1145208.72	0	0
3	2011	2	1466496.65	0	0
4	2011	3	1625064.78	0	0
5	2011	4	1816725.28	0	0
6	2012	1	1275080.34	0	0
7	2012	2	1200605.46	0	0
8	2012	3	1304816.3	0	0
9	2012	4	1536995.39	2037.09	608.33
10	2013	1	2353255.87000001	112377.560000003	52223.9399999998
11	2013	2	3506077.80000008	160841.050000001	75697.5600000002
12	2013	3	3836410.01000009	163982.700000001	79576.0200000005
13	2013	4	4731756.68000016	186184.209999997	91380.5500000015
14	2014	1	0	30371.3500000008	15323.3699999999

- So the sudden drop in sales maybe due to that the Bikes category isn't sold. So we may look into if the Bikes are not in stock. And total sales in accessories and clothing also decrease in 1st quarter of 2014.
- Accessories and Clothing also take part in the increasing of sales from the 4th quarter of 2012 to 4th quarter of 2013.

- Sales by weekday:

```
-- sales by weekdays
SELECT d.[WeekDay],
        SUM(s.TotalSales) AS total_sales
FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
JOIN AdventureWorks2019_DDS.dbo.DimDate d ON s.OrderDateKey = d.DateKey
GROUP BY d.[WeekDay], d.WeekDayNumber
ORDER BY d.WeekDayNumber;
```

	WeekDay	total_sales
1	Sunday	3772883.92
2	Monday	3858949.39
3	Tuesday	3927373.9
4	Wednesday	3834993.29
5	Thursday	3820966.95
6	Friday	3805347.38
7	Saturday	3781713.66

- Sales over weekdays don't have significant differences. Customers seem to shop more in weekdays rather than weekends. And Tuesday is the most idea day for shoppers.

- Sales by month:

```
-- sales by months
SELECT d.[MonthName], SUM(s.TotalSales) AS total_sales
FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
    JOIN AdventureWorks2019_DDS.dbo.DimDate d
    ON s.OrderDateKey = d.DateKey
GROUP BY d.[MonthName], DATEPART(MONTH, d.[Date])
ORDER BY DATEPART(MONTH, d.[Date]);
```

	MonthName	total_sales
1	January	1673269.84
2	February	1593463.42
3	March	1717107.89
4	April	1749506.77
5	May	1981867.32
6	June	2678344.43
7	July	2198831.65
8	August	2477189.39
9	September	2333828.77
10	October	2665030.17
11	November	2760430.12
12	December	2973358.72

- December has the highest sales. Months in winter have better sales performance than months in spring. June also has high sales, it's ranked 3rd in the year, behind November.

- Sales by territory group and country:

```
-- sales by territory group & country
SELECT t.[Group], t.Country, ROUND(SUM(s.TotalSales),2) AS sales,
        SUM(ROUND(SUM(s.TotalSales), 2)) OVER(PARTITION BY t.[Group]) AS group_sales
FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
    JOIN AdventureWorks2019_DDS.dbo.DimSalesTerritory t
    ON s.SalesTerritoryKey = t.SalesTerritoryKey
GROUP BY t.[Group], t.Country
```

	Group	Country	sales	group_sales
1	Europe	France	2492684.51	9758022.15
2	Europe	Germany	2789686.77	9758022.15
3	Europe	United Kingdom	4475650.87	9758022.15
4	North America	Canada	1490304.82	10879863.97
5	North America	United States	9389559.15	10879863.97
6	Pacific	Australia	6164342.37	6164342.37

- United States outsells all other sales territories with more than 9.3 million dollars. Pacific only has 1 territory in Australia but it's has about 6 million dollars, and ranked 2nd , only after United States. The UK has the highest sales in Europe. Even though being in the same group with the US, Canada only has nearly 1.5 million, which is ranked last. We may examine why.

```
--count territories
SELECT [Group], Country, COUNT(*) number_of_territories
FROM AdventureWorks2019_DDS.dbo.DimSalesTerritory
GROUP BY [Group], Country
```

	Group	Country	number_of_territories
1	Pacific	Australia	1
2	North America	Canada	1
3	Europe	France	1
4	Europe	Germany	1
5	Europe	United Kingdom	1
6	North America	United States	5

- So one of the reasons United States has the highest sales is that it has 5 sales territories over the country.

- Customers:

```
--count customer in each category
SELECT g.CountryRegionName, COUNT(*) AS number_of_customers
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.DimGeography g
    ON c.GeographyKey = g.GeographyKey
GROUP BY g.CountryRegionName
```

	CountryRegionName	number_of_customers
1	Australia	3591
2	Canada	1571
3	France	1810
4	Germany	1780
5	United Kingdom	1913
6	United States	7819

- Australia has a large number of customers and is ranked 2nd , this also explains why it's ranked 2nd in sales. Canada has about 350 less customers than United Kingdom but Canada sales is nearly 3 times smaller than United Kindom's sales.

- Sales by customers gender:

```
--sales by customer gender
SELECT c.Gender, COUNT(*) AS number_of_customers, SUM(s.TotalSales) AS total_sales
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.FactInternetSales s
    ON c.CustomerKey = s.CustomerKey
GROUP BY c.Gender;
```

	Gender	number_of_customers	total_sales
1	Male	30381	13327642.770002
2	Female	30017	13474585.720002

- There's not much difference between male and female purchase in sales.

- Sales by customer's marital status:

```
--sales by customer marital status
SELECT c.MaritalStatus, COUNT(*) AS number_of_customers,
        ROUND(SUM(s.TotalSales), 2) AS total_sales
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.FactInternetSales s
    ON c.CustomerKey = s.CustomerKey
GROUP BY c.MaritalStatus;
```

	MaritalStatus	number_of_customers	total_sales
1	Married	33273	14091622.44
2	Single	27125	12710606.05

- There are more married customers and the sales also larger for married ones.

- Sales on if a customer is a parent:

```sql
--sales on if customer a parent status
SELECT c.IsParent, COUNT(*) AS number_of_customers,
        ROUND(SUM(s.TotalSales), 2) AS total_sales
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.FactInternetSales s
    ON c.CustomerKey = s.CustomerKey
GROUP BY c.IsParent;
```

	IsParent	number_of_customers	total_sales
1	Yes	43350	19249332.1
2	No	17048	7552896.39

- So there's about 70% of customers are parents. And sales of parent almost 3 times larger than customers that are note parents.

- Sales performance on if customers owns a car

```sql
--sales on if customer owns a car
SELECT c.IsCarOwner, COUNT(*) AS number_of_customers,
        ROUND(SUM(s.TotalSales), 2) AS total_sales
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.FactInternetSales s
    ON c.CustomerKey = s.CustomerKey
GROUP BY c.IsCarOwner;
```

	IsCarOwner	number_of_customers	total_sales
1	Yes	46330	19046642.88
2	No	14068	7755585.61

- Almost similar with parent status, car owner status also has 3:1 ratio for car owner customers and customers that don't own car.

- Sales on if customer is a house owner

```sql
--sales on if customer owns a house
SELECT c.IsHouseOwner, COUNT(*) AS number_of_customers,
        ROUND(SUM(s.TotalSales), 2) AS total_sales
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.FactInternetSales s
    ON c.CustomerKey = s.CustomerKey
GROUP BY c.IsHouseOwner;
```

	IsHouseOwner	number_of_customers	total_sales
1	Yes	41699	18779347.82
2	No	18699	8022880.67

- Sales by customer's educational level

```sql
--sales on customer's educational level
SELECT c.EducationLevel, COUNT(*) AS number_of_customers,
        ROUND(SUM(s.TotalSales), 2) AS total_sales
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.FactInternetSales s
    ON c.CustomerKey = s.CustomerKey
GROUP BY c.EducationLevel;
```

	EducationLevel	number_of_customers	total_sales
1	High School	10320	4138609.37
2	Partial High School	4708	1478315.19
3	Bachelors	18144	8821793.47
4	Graduate Degree	10603	5140175.98
5	Partial College	16623	7223334.48

- Abc

- Sales by customer's job:

```
--sales by customer's job
SELECT c.Job, COUNT(*) AS number_of_customers,
        ROUND(SUM(s.TotalSales), 2) AS total_sales
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.FactInternetSales s
    ON c.CustomerKey = s.CustomerKey
GROUP BY c.Job;
```

	Job	number_of_customers	total_sales
1	Management	10594	4766056.5
2	Manual	6924	2786012.37
3	Skilled Manual	14261	5860463.83
4	Clerical	9624	4673965.37
5	Professional	18995	8715730.42

- Customers with professional roles have the largest number and sales purchase.

- Sales by customer age group:

```
--sales by customer's age
WITH age_group_sub AS(
    SELECT CustomerKey,
           CASE WHEN '2014'- DATEPART(YEAR, BirthDate) BETWEEN 28 AND 38 THEN '28-38'
                WHEN '2014'- DATEPART(YEAR, BirthDate) BETWEEN 38 AND 48 THEN '38-48'
                WHEN '2014'- DATEPART(YEAR, BirthDate) BETWEEN 48 AND 58 THEN '48-58'
                WHEN '2014'- DATEPART(YEAR, BirthDate) BETWEEN 58 AND 68 THEN '58-68'
                WHEN '2014'- DATEPART(YEAR, BirthDate) BETWEEN 68 AND 78 THEN '68-78'
                WHEN '2014'- DATEPART(YEAR, BirthDate) BETWEEN 78 AND 88 THEN '78-88'
                WHEN '2014'- DATEPART(YEAR, BirthDate) BETWEEN 88 AND 98 THEN '88-98'
           END AS age_group
    FROM AdventureWorks2019_DDS.dbo.DimCustomer
)
SELECT sub.age_group,
       COUNT(*) AS number_of_customers,
       ROUND(SUM(s.TotalSales), 2) AS total_sales
FROM AdventureWorks2019_DDS.dbo.FactInternetSales s
    JOIN age_group_sub sub
    ON s.CustomerKey = sub.CustomerKey
GROUP BY sub.age_group
ORDER BY sub.age_group;
```

	age_group	number_of_customers	total_sales
1	28-38	20315	8785787.99
2	38-48	19158	9281804.55
3	48-58	12262	5670301.03
4	58-68	6563	2524181.12
5	68-78	1889	511750.95
6	78-88	158	27121.42
7	88-98	53	1281.43

- There's no customer that is under 28. Most customers are from 28 to 58 years old. With group 38-48 has the highest sales

- Sales by customer's yearly income:

```
--sales by customer's yearly income
SELECT c.YearlyIncome, COUNT(s.CustomerKey) AS customer_count,
        ROUND(AVG(s.TotalSales), 2) AS mean_sales,
        ROUND(SUM(s.TotalSales), 2) AS total_sales
        --RANK() OVER(ORDER BY s.TotalSales) as
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.FactInternetSales s
    ON c.CustomerKey = s.CustomerKey
GROUP BY c.YearlyIncome
ORDER BY c.YearlyIncome;
```

	YearlyIncome	customer_count	mean_sales	total_sales
1	10000	3352	364.2	1220799.17
2	20000	5128	379.12	1944144.13
3	30000	6994	417.12	2917354.59
4	40000	9181	462.18	4243233.36
5	50000	1977	389.54	770112.38
6	60000	9470	392.16	3713802.01
7	70000	8267	464.9	3843341.31
8	80000	4848	412.1	1997862.01
9	90000	3143	525.9	1652897.31
10	100000	1955	441.02	862188.25
11	110000	1715	492.22	844158.82
12	120000	1242	531.25	659810.94
13	130000	1906	643.58	1226661.5
14	150000	385	700.14	269554.59
15	160000	359	662.68	237903.16
16	170000	476	836.99	398404.96

- Customers with highest sales has yearly income = 40000.

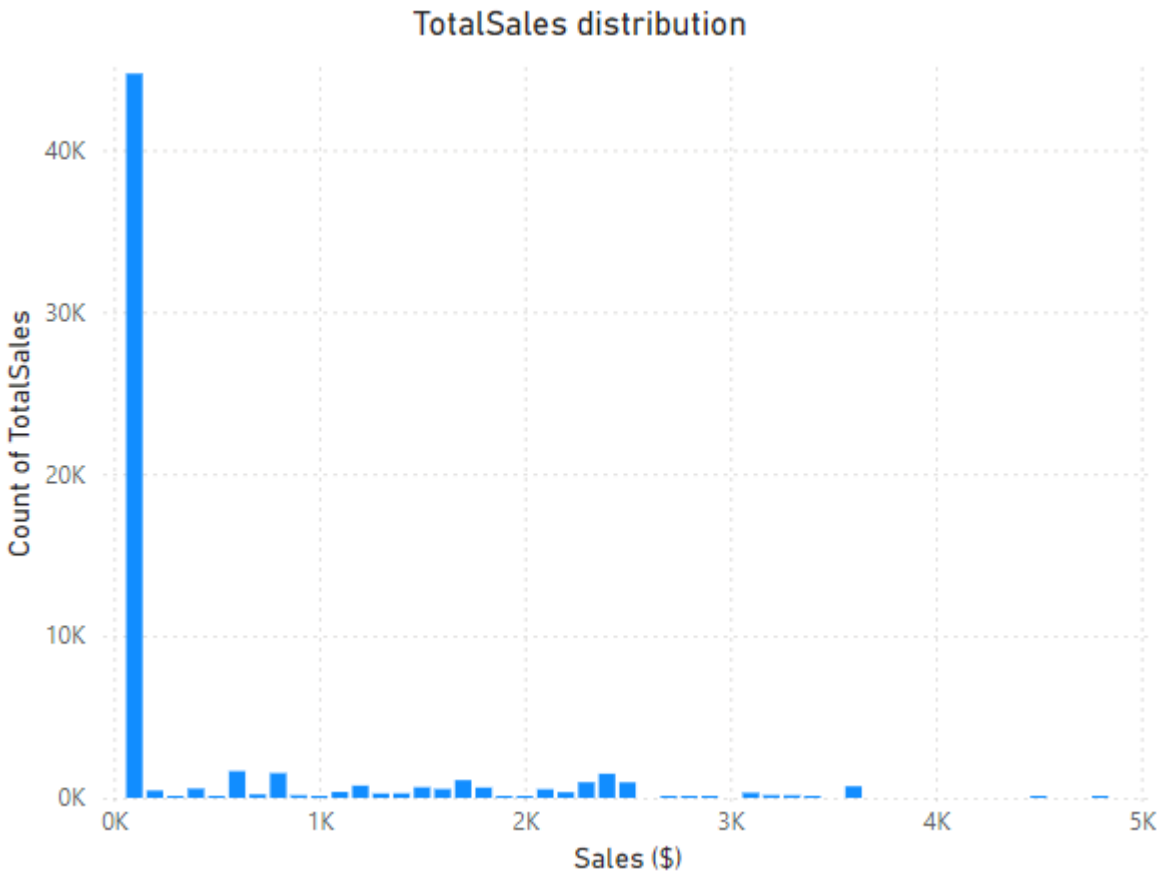
- Sales by customer's quarters joined:

```
--sales by customer's first date purchase
SELECT DATEDIFF(QUARTER, c.DateFirstPurchase, '2014-02-01') AS quarters_joined, --2014-01-28 is last purchase
        ROUND(AVG(s.TotalSales), 2) AS mean_sales,
        ROUND(SUM(s.TotalSales), 2) AS total_sales
        --RANK() OVER(ORDER BY s.TotalSales) as
FROM AdventureWorks2019_DDS.dbo.DimCustomer c
    JOIN AdventureWorks2019_DDS.dbo.FactInternetSales s
    ON c.CustomerKey = s.CustomerKey
GROUP BY DATEDIFF(QUARTER, c.DateFirstPurchase, '2014-02-01')
ORDER BY quarters_joined;
```

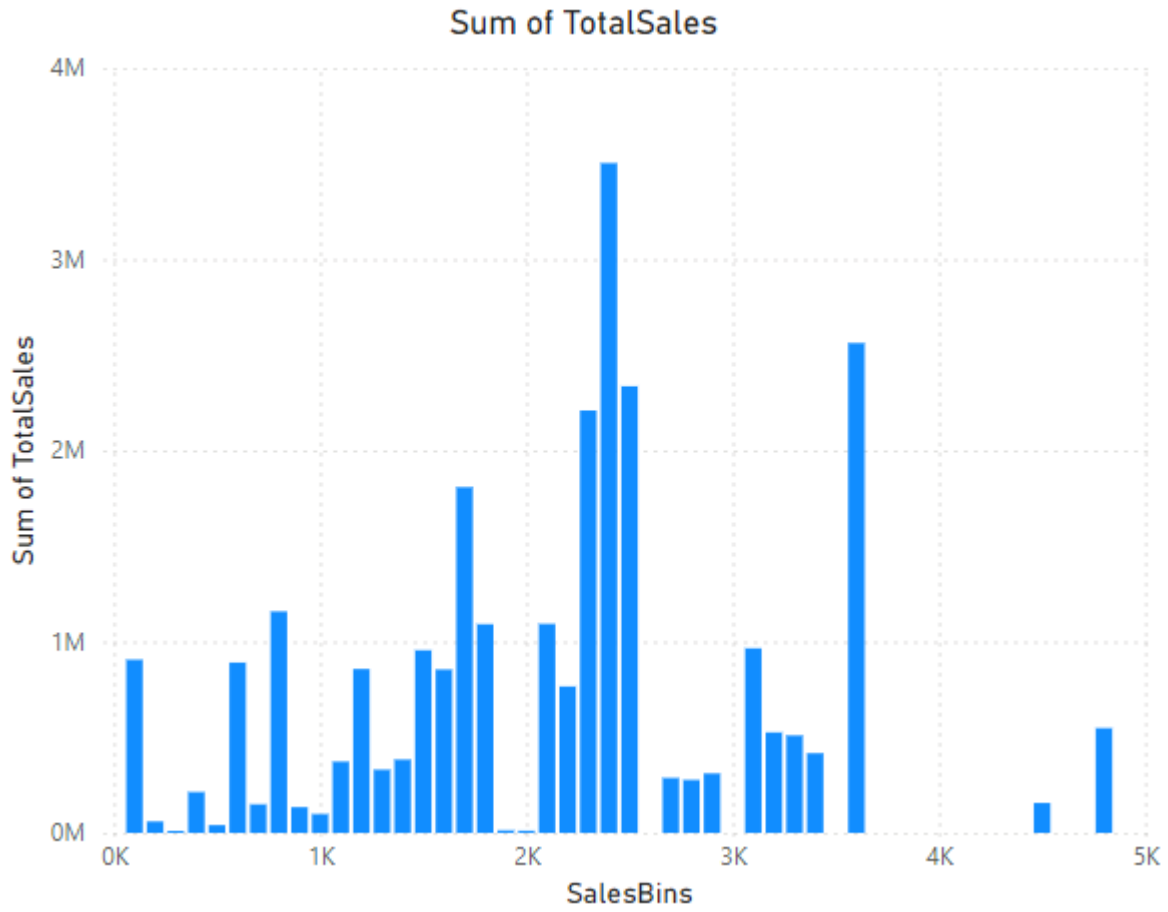
	quarters_joined	mean_sales	total_sales
1	0	22.09	26089.46
2	1	235.72	2063528.98
3	2	137.17	1180655.23
4	3	147.65	1479597.7
5	4	106.85	1038125.31
6	5	830.95	3036297.8
7	6	783.88	2731048.6
8	7	823.22	2570905.24
9	8	893.57	2603870.82
10	9	1097.24	3098600.62
11	10	1186.53	2645967.93
12	11	1129.63	2327041.47
13	12	1091.35	1939329.41
14	13	1154.15	61169.92

- Customers who joined earlier tend to purchase more

- Sales statistics:

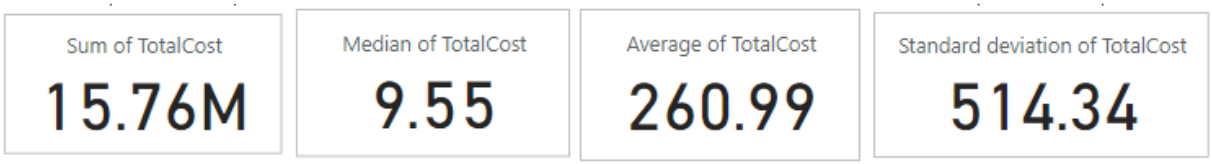


- The distribution of TotalSales is left skewed heavily, with a large standard deviation.



- Even though most customers purchased with order from 0-100$, the highest sum of sales is ranged from 2300$ to 2600$ orders with sum upto 3.5 million dollars.

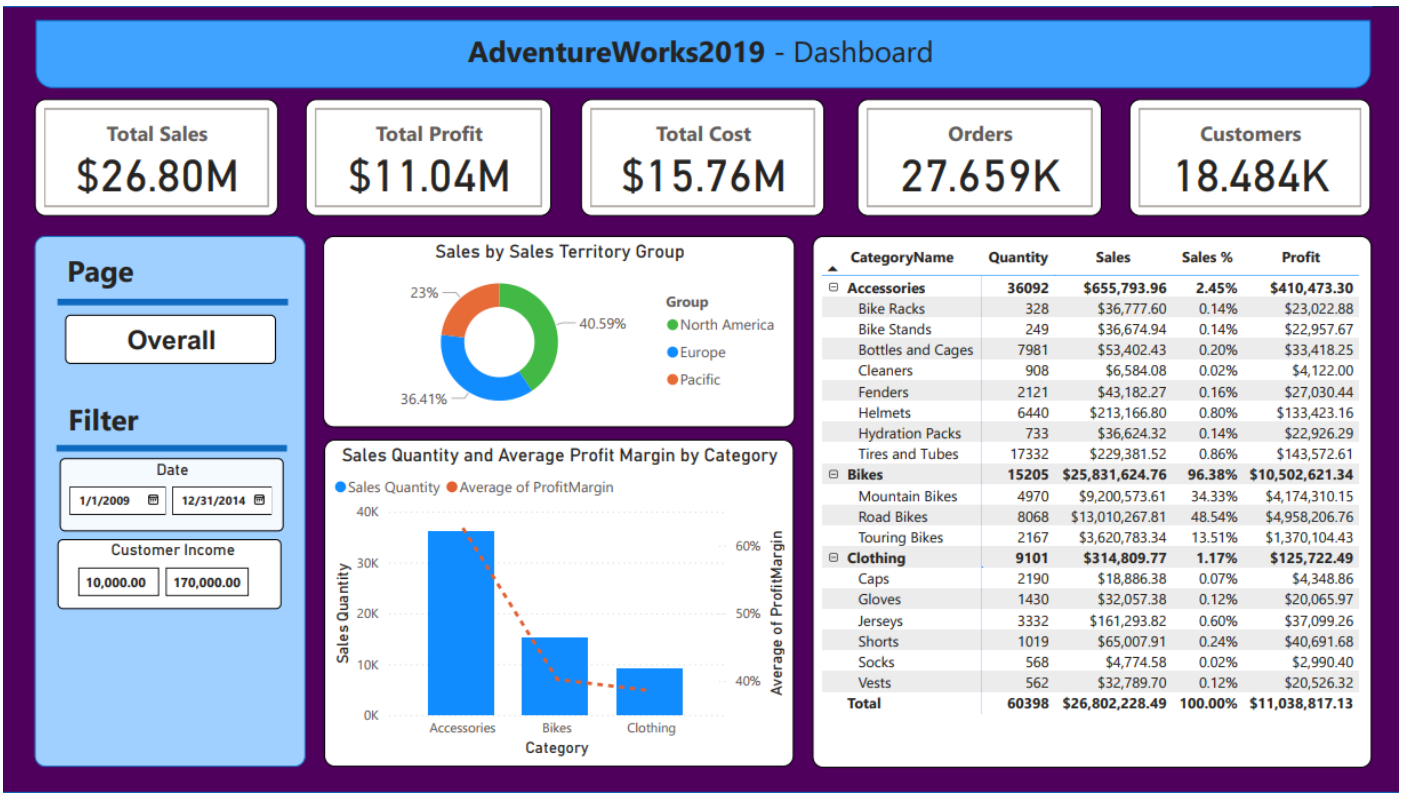
- Total cost statistics:



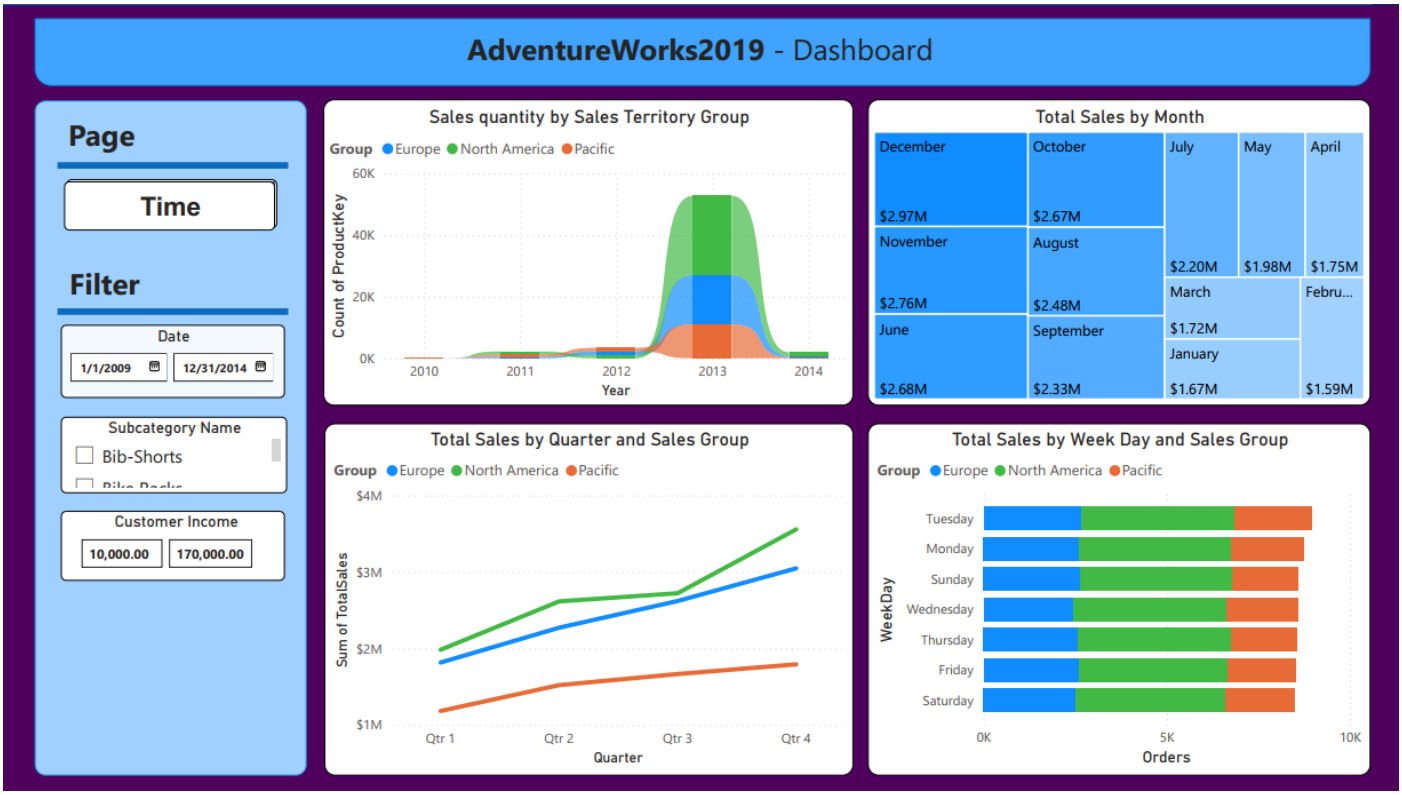
4. Power BI Dashboard

- First, connect Power BI to the new database ‘AdventureWorks2019_DDS’

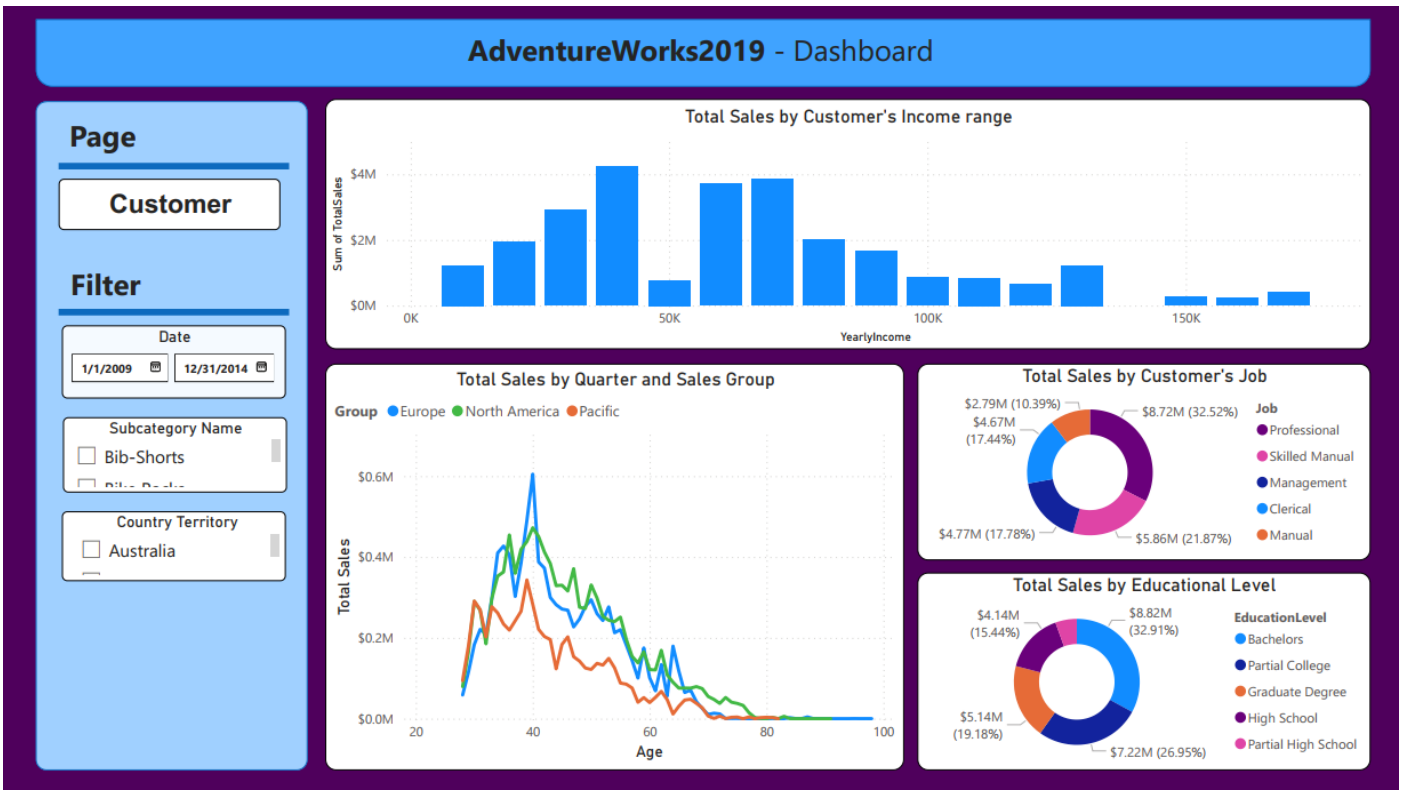
- Dasboard design: contains 4 pages of metric summaries about sales, profit, cost,… with page switch, filters
- Overall sales:



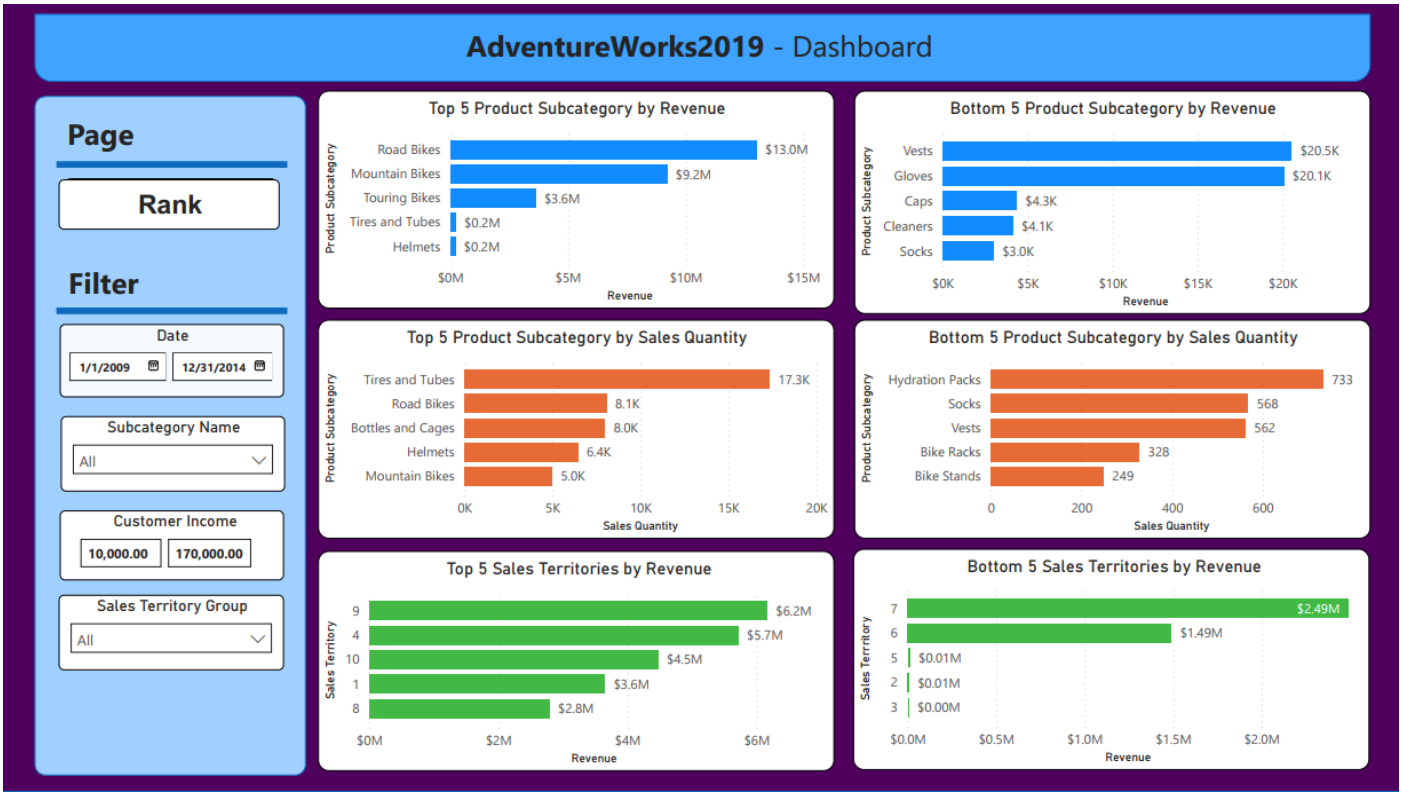
- Sales over time:



- Sales based on customer:



- Top highest and lowest:



-

5. Project Summary

Insights