AdventureWorks Database Analysis

Quan Phuc Minh Nguyen

2022-08-23

## Introduction

The point of this analysis is to utilize the online and reseller sales data to determine which products or product categories keep customers coming back to AdventureWorks. The company also wants to identify which product generates the highest revenue per one product quantity.

Most data manipulation and transformation process will be done with SQL, while data visualization will be created with R and Tableau.

## Prepare

The prepare phase helps ensure data credibility and data integrity.

### Database location

The AdventureWorks database can be found [here](https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms).

### Duplicate and null values

Checking if the raw data is free of duplicate values allows the data to be unbiased and credible. The following SQL query checks for duplicate values in the reseller sales table - if a sales order number contains more than one product key, it is considered a duplicate.

Duplicate <-"  
SELECT   
 SalesOrderNumber,  
 ProductKey,  
 COUNT(\*)  
FROM   
 AdventureWorksDW2019.dbo.FactResellerSales  
GROUP BY  
 SalesOrderNumber,  
 ProductKey  
HAVING  
 COUNT(\*)>1  
;  
"

## [1] SalesOrderNumber ProductKey   
## <0 rows> (or 0-length row.names)

We use a similar SQL query to check for duplicate values in the online sales and customer tables. Fortunately, based on the result, there are no duplicate orders and no duplicate customer information.

### Non-subcategorized products

Some products do not have a subcategory key, so it is important to take into consideration the orders containing such products. From the table product, the first 209 products labeling 1-209 are not subcategorized. Thus, the following query returns orders with non-subcategorized products.

Unsub <-"  
SELECT   
 COALESCE(Res.ProductKey,Onl.ProductKey) AS ProductKey,  
 COALESCE(Res.SalesOrderNumber,Onl.SalesOrderNumber) AS OrderNumber  
FROM   
 AdventureWorksDW2019.dbo.FactResellerSales AS Res  
 INNER JOIN  
 AdventureWorksDW2019.dbo.FactInternetSales AS Onl   
 ON Res.ProductKey=Onl.ProductKey  
WHERE  
 COALESCE(Res.ProductKey,Onl.ProductKey)<210  
;  
"

## [1] ProductKey OrderNumber  
## <0 rows> (or 0-length row.names)

Results show that there are no such orders. Thus, we can move on to processing the data.

## Process

The process phase allows us to transform the data into meaningful patterns for analysis.

### Best-selling products

The following SQL query returns the top 5 best-selling products of all time. This query also includes subqueries to find the top 5 best-selling products in each division.

BestSellingProduct <-"  
SELECT TOP 5  
 EnglishProductName AS Product,  
 TotalProductsPurchased AS QuantityPurchased  
FROM  
 (  
 SELECT   
 COALESCE(Onl.ProductKey,Res.ProductKey) AS ProductKey,  
 CASE  
 WHEN Onl.QuantityPurchased IS NULL   
 THEN Res.QuantityPurchased  
 WHEN Res.QuantityPurchased IS NULL   
 THEN Onl.QuantityPurchased  
 ELSE Onl.QuantityPurchased+Res.QuantityPurchased   
 END AS TotalProductsPurchased  
 FROM   
 (  
 SELECT  
 ProductKey,  
 COUNT(ProductKey) AS QuantityPurchased  
 FROM   
 AdventureWorksDW2019.dbo.FactInternetSales  
 GROUP BY  
 ProductKey  
 ) AS Onl -- Best-selling products online  
 LEFT JOIN  
 (  
 SELECT  
 ProductKey,  
 COUNT(ProductKey) AS QuantityPurchased  
 FROM   
 AdventureWorksDW2019.dbo.FactResellerSales  
 GROUP BY  
 ProductKey  
 ) AS Res -- Best-selling products from resellers  
 ON Onl.ProductKey=Res.ProductKey  
 ) AS ProductSales  
 LEFT JOIN AdventureWorksDW2019.dbo.DimProduct  
 ON ProductSales.ProductKey=AdventureWorksDW2019.dbo.DimProduct.ProductKey -- Return the product name  
ORDER BY  
 TotalProductsPurchased DESC  
;  
"

## Product QuantityPurchased  
## 1 Water Bottle - 30 oz. 4688  
## 2 Patch Kit/8 Patches 3354  
## 3 Mountain Tire Tube 3095  
## 4 AWC Logo Cap 2650  
## 5 Sport-100 Helmet, Red 2566

The query first generates a table with product keys and the corresponding quantity. Then, it returns the products’ names through a JOIN clause with the product description table.

### Best-selling categories

Each product has a category, and this SQL query reports the number of products purchased under each type.

BestSellingCategories <-"  
SELECT  
 COALESCE(Res.Category, Onl.Category) AS Category,  
 COALESCE(Res.CategoriesPurchased,0) AS ResellerQuantity,  
 COALESCE(Onl.CategoriesPurchased,0) AS OnlineQuantity  
FROM  
 (  
 SELECT   
 EnglishProductCategoryName AS Category,  
 COUNT(EnglishProductCategoryName) AS CategoriesPurchased  
 FROM   
 (  
 SELECT  
 ProductKey,  
 COUNT(ProductKey) AS QuantityPurchased  
 FROM  
 AdventureWorksDW2019.dbo.FactResellerSales  
 GROUP BY  
 ProductKey  
 ) AS TotalNoProducts  
 LEFT JOIN  
 AdventureWorksDW2019.dbo.DimProduct  
 ON TotalNoProducts.ProductKey=AdventureWorksDW2019.dbo.DimProduct.ProductKey  
 LEFT JOIN  
 AdventureWorksDW2019.dbo.DimProductSubcategory  
 ON AdventureWorksDW2019.dbo.DimProduct.ProductSubcategoryKey  
 =AdventureWorksDW2019.dbo.DimProductSubcategory.ProductSubcategoryKey  
 LEFT JOIN  
 AdventureWorksDW2019.dbo.DimProductCategory  
 ON AdventureWorksDW2019.dbo.DimProductSubcategory.ProductCategoryKey  
 =AdventureWorksDW2019.dbo.DimProductCategory.ProductCategoryKey  
 GROUP BY  
 EnglishProductCategoryName  
 ) AS Res  
 FULL OUTER JOIN  
 (  
 SELECT   
 EnglishProductCategoryName AS Category,  
 COUNT(EnglishProductCategoryName) AS CategoriesPurchased  
 FROM   
 (  
 SELECT  
 ProductKey,  
 COUNT(ProductKey) AS QuantityPurchased  
 FROM   
 AdventureWorksDW2019.dbo.FactInternetSales  
 GROUP BY  
 ProductKey  
 ) AS TotalNoProducts  
 LEFT JOIN  
 AdventureWorksDW2019.dbo.DimProduct  
 ON TotalNoProducts.ProductKey=AdventureWorksDW2019.dbo.DimProduct.ProductKey  
 LEFT JOIN  
 AdventureWorksDW2019.dbo.DimProductSubcategory  
 ON AdventureWorksDW2019.dbo.DimProduct.ProductSubcategoryKey  
 =AdventureWorksDW2019.dbo.DimProductSubcategory.ProductSubcategoryKey  
 LEFT JOIN  
 AdventureWorksDW2019.dbo.DimProductCategory  
 ON AdventureWorksDW2019.dbo.DimProductSubcategory.ProductCategoryKey  
 =AdventureWorksDW2019.dbo.DimProductCategory.ProductCategoryKey  
 GROUP BY  
 EnglishProductCategoryName  
 ) AS Onl ON Res.Category=Onl.Category  
;  
"

## Category ResellerQuantity OnlineQuantity  
## 1 Accessories 16 22  
## 2 Bikes 125 116  
## 3 Clothing 43 20  
## 4 Components 150 0

### Best-selling subcategories

We want to further examine each category and find out the exact quantity under each subcategory. Suppose we want to acquire a table with quantity count for each subcategory under the category Bikes that are purchased online. Similar to the previous query, the following one generates the quantity from the sales table and connects to the product tables to return the names for Bikes’ subcategories.

OnlineBikes <-"  
SELECT   
 EnglishProductCategoryName AS Category,  
 EnglishProductSubcategoryName AS Subcategory,  
 COUNT(EnglishProductSubcategoryName) AS QuantitiesPurchasedOnline  
FROM   
 (  
 SELECT  
 ProductKey,  
 COUNT(ProductKey) AS QuantityPurchased  
 FROM   
 AdventureWorksDW2019.dbo.FactInternetSales  
 GROUP BY  
 ProductKey  
 ) AS Onl  
 LEFT JOIN  
 AdventureWorksDW2019.dbo.DimProduct  
 ON Onl.ProductKey=AdventureWorksDW2019.dbo.DimProduct.ProductKey  
 LEFT JOIN  
 AdventureWorksDW2019.dbo.DimProductSubcategory  
 ON AdventureWorksDW2019.dbo.DimProduct.ProductSubcategoryKey  
 =AdventureWorksDW2019.dbo.DimProductSubcategory.ProductSubcategoryKey  
 LEFT JOIN  
 AdventureWorksDW2019.dbo.DimProductCategory  
 ON AdventureWorksDW2019.dbo.DimProductSubcategory.ProductCategoryKey  
 =AdventureWorksDW2019.dbo.DimProductCategory.ProductCategoryKey  
WHERE   
 EnglishProductCategoryName='Bikes'  
GROUP BY  
 EnglishProductCategoryName,   
 EnglishProductSubcategoryName  
;  
"

## Category Subcategory QuantitiesPurchasedOnline  
## 1 Bikes Mountain Bikes 34  
## 2 Bikes Road Bikes 60  
## 3 Bikes Touring Bikes 22

### Total Sales

We would like to analyze total sales to identify patterns and aspects of sales AdventureWorks wants to address. The following SQL query retrieves the sales data in the U.S. territory from 2011 to 2013.

TotalSales <-"  
WITH  
 Res AS (  
 SELECT   
 CASE   
 WHEN CAST(SUBSTRING(CAST(OrderDateKey AS CHAR),5,2) AS INT) BETWEEN 1 AND 3   
 THEN LEFT(OrderDateKey, 4) + ' Q1'  
 WHEN CAST(SUBSTRING(CAST(OrderDateKey AS CHAR),5,2) AS INT) BETWEEN 4 AND 6   
 THEN LEFT(OrderDateKey, 4) + ' Q2'  
 WHEN CAST(SUBSTRING(CAST(OrderDateKey AS CHAR),5,2) AS INT) BETWEEN 7 AND 9   
 THEN LEFT(OrderDateKey, 4) + ' Q3'  
 ELSE LEFT(OrderDateKey, 4) + ' Q4'  
 END AS SalesQuarter,  
 ROUND(SUM(SalesAmount),0) AS Sales  
 FROM AdventureWorksDW2019.dbo.FactResellerSales  
 WHERE SalesTerritoryKey BETWEEN 1 AND 5  
 GROUP BY OrderDateKey  
 ),  
 Onl AS (   
 SELECT   
 CASE   
 WHEN CAST(SUBSTRING(CAST(OrderDateKey AS CHAR),5,2) AS INT) BETWEEN 1 AND 3   
 THEN LEFT(OrderDateKey, 4) + ' Q1'  
 WHEN CAST(SUBSTRING(CAST(OrderDateKey AS CHAR),5,2) AS INT) BETWEEN 4 AND 6   
 THEN LEFT(OrderDateKey, 4) + ' Q2'  
 WHEN CAST(SUBSTRING(CAST(OrderDateKey AS CHAR),5,2) AS INT) BETWEEN 7 AND 9   
 THEN LEFT(OrderDateKey, 4) + ' Q3'  
 ELSE LEFT(OrderDateKey, 4) + ' Q4'  
 END AS SalesQuarter,  
 ROUND(SUM(SalesAmount),0) AS Sales  
 FROM AdventureWorksDW2019.dbo.FactInternetSales  
 WHERE SalesTerritoryKey BETWEEN 1 AND 5  
 GROUP BY OrderDateKey  
 )  
SELECT   
 COALESCE(Res.SalesQuarter,Onl.SalesQuarter) AS SalesPeriod,  
 COALESCE(SUM(Res.Sales), 0) AS ResellerSales,  
 COALESCE(SUM(Onl.Sales), 0) AS OnlineSales,  
 COALESCE(SUM(Onl.Sales), 0)+COALESCE(SUM(Res.Sales), 0) AS TotalSales  
FROM   
 Res   
 FULL OUTER JOIN  
 Onl ON Res.SalesQuarter=Onl.SalesQuarter  
WHERE LEFT(COALESCE(Res.SalesQuarter,Onl.SalesQuarter),4) BETWEEN 2011 AND 2013  
GROUP BY COALESCE(Res.SalesQuarter,Onl.SalesQuarter)  
ORDER BY SalesPeriod;  
"

## SalesPeriod ResellerSales OnlineSales TotalSales  
## 1 2011 Q1 187030299 1258188 188288487  
## 2 2011 Q2 270928880 1332656 272261536  
## 3 2011 Q3 311912777 2217092 314129869  
## 4 2011 Q4 372854769 2454894 375309663  
## 5 2012 Q1 458654560 1339806 459994366  
## 6 2012 Q2 362010714 936483 362947197  
## 7 2012 Q3 334620762 826560 335447322  
## 8 2012 Q4 439135670 1208286 440343956  
## 9 2013 Q1 547751610 2444172 550195782  
## 10 2013 Q2 444136875 4039131 448176006  
## 11 2013 Q3 414926624 4269432 419196056  
## 12 2013 Q4 342567704 3755654 346323358

## Analysis

### Visualization

Data visualization is helpful for identifying patterns and trends. From the TotalSales table, online sales is relatively low compared to the total sales, so we illustrate the data in two bar graphs.

#### Reseller vs. Total & Online

Since the sales data is a discrete variable, it can be ideally represented by a bar chart. The table generated by SQL is a wide-formatted data frame, so we need to reshape it into a long-formatted one for the sake of using R. Using the tidyr package:

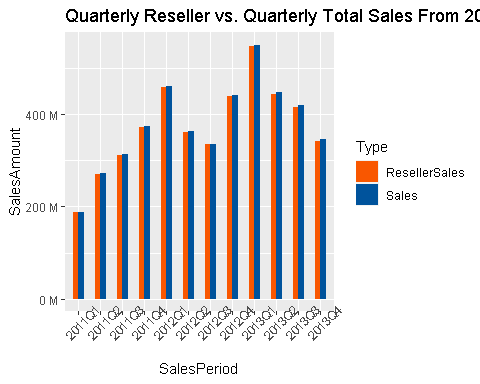
library(tidyr)  
TotalSalesLong <- TotalSales %>% pivot\_longer(cols=c('ResellerSales','OnlineSales','Sales'),  
 names\_to='Type',  
 values\_to='SalesAmount')

Filtering out the online sales data:

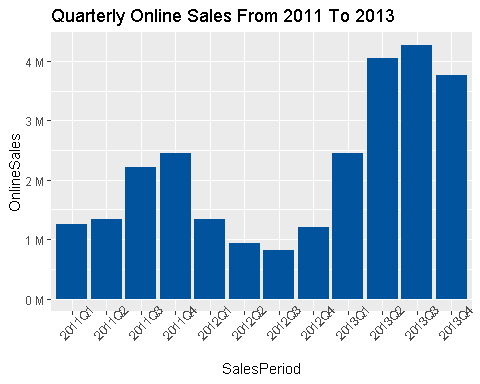
RvT <- TotalSalesLong %>% filter(Type=='ResellerSales'|Type=='Sales')

Now, we can plot our data using ggplot2 package:

library(ggplot2)  
library(scales)  
library(tidyverse)  
  
ggplot(data=RvT)+  
 geom\_bar(mapping=aes(x=SalesPeriod,y=SalesAmount,fill=Type,width=.5),stat='identity',position='dodge')+  
 scale\_y\_continuous(labels = unit\_format(unit = "M", scale = 1e-6))+ # Rescaling metrics  
 scale\_fill\_manual(values = c("#F95700","#00539C"))+ # Contrasting color pallete  
 labs(title='Quarterly Reseller vs. Quarterly Total Sales From 2011 To 2013')+  
 theme(axis.text.x=element\_text(angle=45)) # Make the x-axis readable



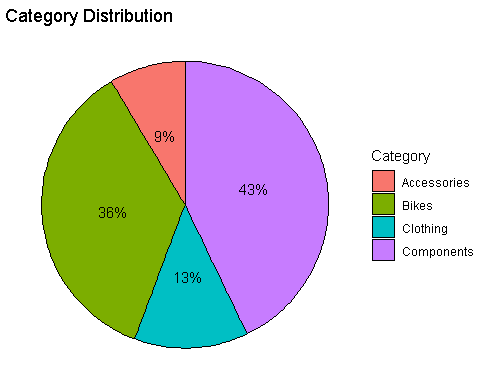
ggplot(data=TotalSales)+  
 geom\_bar(aes(x=SalesPeriod,y=OnlineSales),stat='identity',fill="#00539C")+  
 scale\_y\_continuous(labels = unit\_format(unit = "M", scale = 1e-6))+  
 labs(title='Quarterly Online Sales From 2011 To 2013')+  
 theme(axis.text.x=element\_text(angle=45))



#### Distribution of Categories

The distribution is best illustrated by a pie chart.

ggplot(data=Category,aes(x=" ",y=Quantity, fill=Category))+  
 geom\_bar(stat="identity",color="black")+  
 coord\_polar("y",start=0)+  
 theme\_void()+  
 geom\_text(aes(label = paste0(round(Quantity/sum(Quantity)\*100), "%")),   
 position = position\_stack(vjust = 0.5))+  
 labs(title='Category Distribution')



### Sales insights

In the grouped bar chart, we can notice varying demand patterns within a year, but the average annual sales amount increased every year. Hence, meticulous planning and a flexible manufacturing system would be an excellent solution for AdventureWorks. AdventureWorks should allocate more resources during leaner times of the year for the pre-manufacturing work like the production of components and packages. On the other hand, during busier times, it should focus on manufacturing and distributing products. Since the database only provides two years of sales data, more pattern analysis might be required to foster more effective allocation decisions.

From the online sales chart, we can also see a fluctuating pattern within two years. However, the average sales amount significantly increased in 2013, potentially due to technology and the online platform market improvement. Hence, although the online sales amount was remarkably lower, AdventureWorks should devote more resources to enhancing its online platform, like launching advertising campaigns or building its market app. With more technology becoming readily available, the online platform should be the primary source of sales in no time.

Product category analysis and prospective customer insights can help increase sales and satisfaction and allocate resources to the proper manufacturing sector. The pie plot indicates that components and bikes account for most of the sales. AdventureWorks could utilize this information by planning for product improvements and better promotional strategies.