Basic Databases – Report03

Wroclaw University of Science and Technology, Date: March 23, 2022

Student:	----------	Grade
Identifier	245784	
First name	Rahul	
Last name	Vijayvargiya	

This laboratory assignment consists of 3 task. If you cannot solve the task, try to give at least a partial solution or justification for the reason for the lack of a solution.

All tasks assume that you are using the AdventureWorks OLTP database. Note that an example data dictionary is available at: https://dataedo.com/download/AdventureWorks.pdf.

Data Source: SalesOrderHeader

---------- Part I ---------

Task 1

Let us focus on selected detailed querying capabilities of SQL. In particular, we are interested in accessing summary information from a strictly operational database. Let us assume that we want to assess the yearly performance of individual sales representatives working in AdventureWorks company. As such, we will focus on defining a set of basic SQL queries, which retrieve all of the required information, that can be further reported to the management.

The major goal of this task is to investigate, in a real example, how easy is to run analytical processing tasks, here a particular query, on a strictly transactional database. Moreover, we would also like to study the performance of such a query. As such, we will dive a bit into the area of query performance analysis. In particular, you will introduce three different SQL implementations, all capable of providing identical results, and study their performance. Additional resources regarding needed T-SQL clauses are available below:

- *Overview: https://docs.microsoft.com/en-us/sql/t-sql/queries/queries?view=sql-server-ver15*
- *GROUP BY ROLL UP, CUBE and GROUPING SETS - https://docs.microsoft.com/en-us/sql/t-sql/queries/select-group-by-transact-sql?view=sql-server-ver15*
- *OVER - https://docs.microsoft.com/en-us/sql/t-sql/queries/select-over-clause-transact-sql?view=sql-server-ver15*

It should be stressed that query performance is one of the most important aspects of data analysis. Therefore, most of the available database management tools provide dedicated ways to study the performance of individual queries. For instance, in MS SQL Management Studio you can visual query execution plans, that is, there are three available options: The Estimated Execution Plan (the compiled plan, as produced by the Query Optimizer based on estimations, plan that is stored in the plan cache), The Actual Execution Plan (the compiled plan plus its execution context, available after the query execution has completed and includes actual runtime information), and The Live Query Statistics (the compiled plan plus its execution context, updated every second; includes runtime information such as the actual number of rows flowing through the operators, elapsed time, and the estimated query progress). Additional resources regarding execution plans available in MS SQL Server Management Studio are available below:

- *https://docs.microsoft.com/en-us/sql/relational-databases/performance/execution-plans?view=sql-server-ver15*
- *https://www.sqlshack.com/execution-plans-in-sql-server/*

Prepare a report (using proper SQL queries) to assess the yearly performance of individual sales representatives working in the AdventureWorks company. The key metrics that we focus on are total sales and the number of orders made by employees. The report should contain the data as shown in Table 1. SQL query + fragment of the result (4 records from ?)

Table 1.1. Result structure for task 1.1

Sales Person	Employee ID	Year	Sub Total	Number of orders
Jiang, Stephen	274	2011	28926.25	4
Jiang, Stephen	274	2012	453524.52	22

…	…	…	…	…

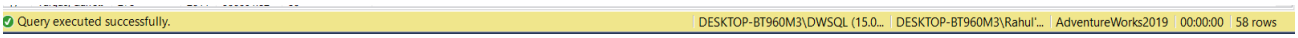
1.1. Prepare the report without using windowed functions (OVER clause).

Solution:

```
SELECT Min(PP.lastname + ', ' + PP.firstname) AS "Sales Person",
       SOH.salespersonid                      AS "Employee ID",
       Year(SOH.orderdate)                    AS "Year",
       Round(Sum(SOH.subtotal), 2)            AS "Sub Total",
       Count(SOH.salesorderid)                "Number of orders"
FROM   [sales].[salesorderheader] AS SOH
       JOIN person.person AS PP
         ON PP.businessentityid = SOH.salespersonid
GROUP  BY Year(SOH.orderdate),
          SOH.salespersonid
ORDER  BY "Employee ID", "Year";
```

Rec: 4/58

Sales Person	Employee ID	Year	Sub Total	Number of orders
Jiang, Stephen	274	2011	28926.25	4
Jiang, Stephen	274	2012	453524.52	22
Jiang, Stephen	274	2013	431088.72	14
Jiang, Stephen	274	2014	178584.36	8



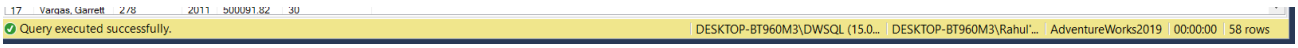
1.2. Prepare the report using windowed functions (OVER clause).

Solution:

```
SELECT DISTINCT PP.lastname + ', ' + PP.firstname
                AS
                "Sales Person",
                SOH.salespersonid
                AS "Employee ID",
                Year(SOH.orderdate)
                AS "Year",
                Round(Sum(SOH.subtotal)
                        OVER(
                          partition BY SOH.salespersonid, Year(SOH.orderdate)),
                2) AS
                "Sub Total",
                Count(SOH.salesorderid)
                  OVER(
                    partition BY SOH.salespersonid, Year(SOH.orderdate))
                AS "Number of orders"
FROM   [sales].[salesorderheader] AS SOH
       INNER JOIN person.person AS PP
               ON PP.businessentityid = SOH.salespersonid
ORDER  BY "Employee ID",
          "Year";
```

Rec: 4/58

Sales Person	Employee ID	Year	Sub Total	Number of orders
Jiang, Stephen	274	2011	28926.25	4
Jiang, Stephen	274	2012	453524.52	22
Jiang, Stephen	274	2013	431088.72	14
Jiang, Stephen	274	2014	178584.36	8

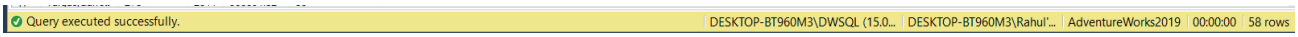


1.3. Prepare the report using CTE, where first you aggregate the sales data to establish yearly performance metrics, and only then attaching the details of the sales person.

Solution:

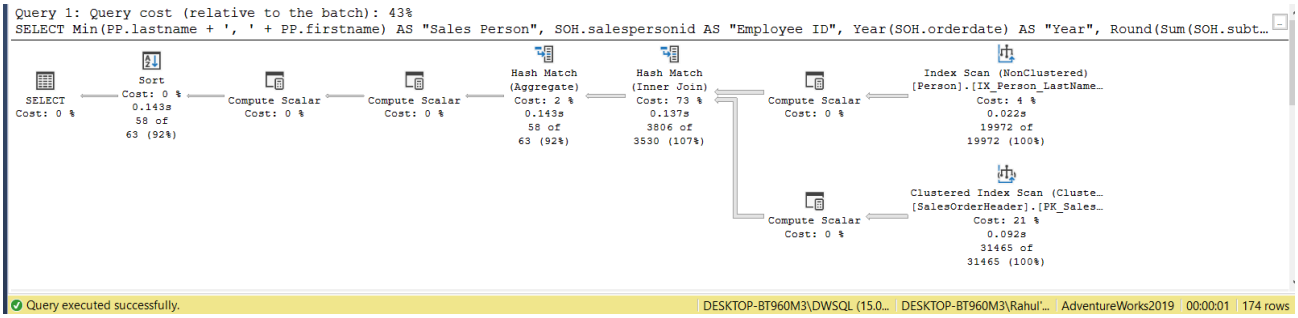
```
WITH sales_cte (salespersonid, salesyear, subtotal, numberoforder)
     AS (SELECT salespersonid,
                Year(order date)          AS "Year",
                Round(Sum(subtotal), 2) AS "Sub Total",
                Count(salesorderid)     AS "Number of Order"
         FROM   [Sales].[salesorderheader]
         WHERE  salespersonid IS NOT NULL
         GROUP  BY Year(orderdate),
                   salespersonid)
SELECT PP.lastname + ',' + PP.firstname "Sales Person",
       salespersonid                    AS "Employee ID",
       salesyear                        AS "Year",
       subtotal                         AS "Sub Total",
       numberoforder                    AS "Number of Orders"
FROM   sales_cte
       JOIN person.person AS PP
         ON sales_cte.salespersonid = PP.businessentityid
ORDER  BY salespersonid,
          salesyear;
```

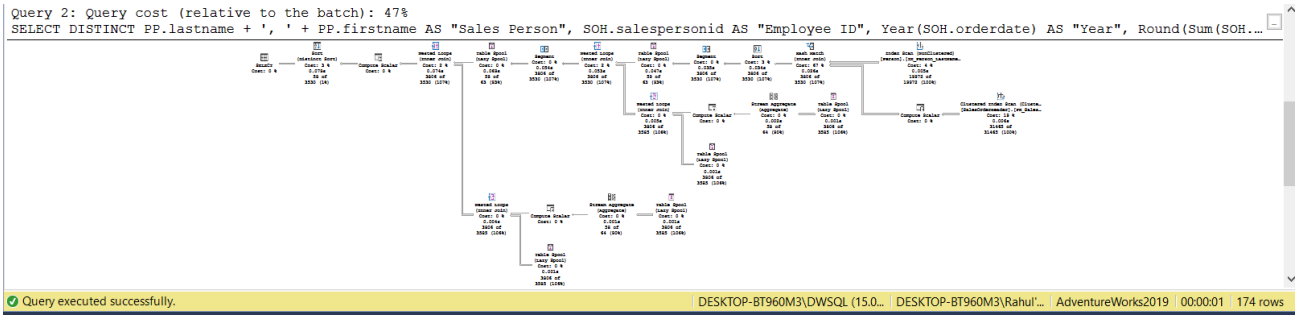
Sales Person	Employee ID	Year	Sub Total	Number of Orders
Jiang,Stephen	274	2011	28926.25	4
Jiang,Stephen	274	2012	453524.52	22
Jiang,Stephen	274	2013	431088.72	14
Jiang,Stephen	274	2014	178584.36	8

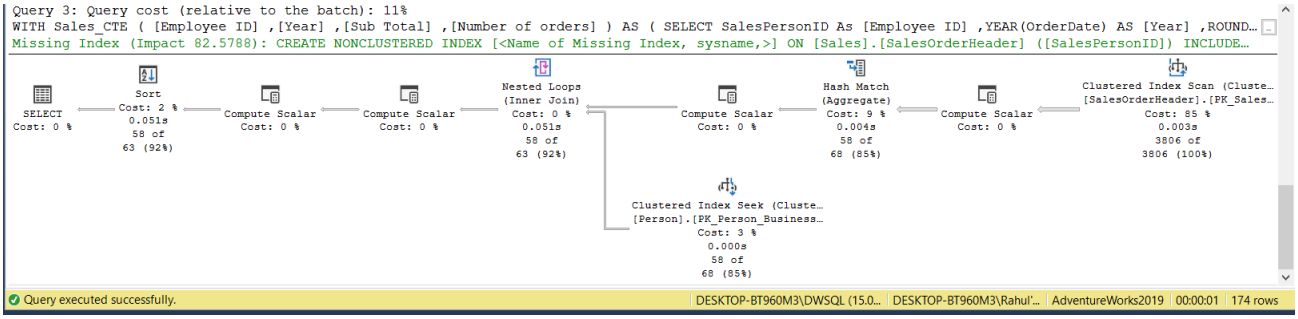


1.4. Assess the quality of the solutions proposed in the previous two tasks. Which one is more advantageous and why, utilise the execution plans for the three queries.

Solution:







1.5. Pick one of the introduced solutions (1-3) and modify it to include basic data summaries, total subtotal value, and total number of orders for each year and for each employee

Solution:

Three Different Queries and Three Different Execution Cost, with First Query we had 43% and 2nd Query with 47%
And 3rd query 11%,

43% which was lower than using over clause because over clause uses nested loops which significantly increases execution cost on the other side,
we have CTE which reduces our execution cost significantly as it is more readable and arranged than subqueries

CONCLUSIONS:

Use this section to provide your conclusions:

Over Clause with partition cost a lot, with CTE we have most control over data, readability and modification is easy and maintaining such queries are bit easier as well

---

---------- Part II ---------

Task 2

Assume you are a data engineer working in Adventure Works Cycles Company, you are responsible for preparing the data for the analyst who is interested in studying basic sales information from the product and customer perspective. Assume that they will use basic reporting and data visualisation tools, such as pivot tables in Excel and dashboards in Tableau. As such, your task is to prepare a set of new, dedicated, relational structures capable of storing data for basic OLAP processing and further move the data from available transactional sources (AdventureWorks database) using SQL statements. Additional resources on using T-SQL statements to create structures and to move data around different tables and databases are available below:

- *Schema creation - https://msdn.microsoft.com/en-us/library/ms189462.aspx*
- *INSERT INTO...SELECT - https://docs.microsoft.com/en-us/sql/t-sql/statements/insert-transact-sql?view=sql-server-ver15*
- *SELECT INTO clause - http://msdn.microsoft.com/en-us/library/ms188029.aspx*
- *ALTER TABLE clause - https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-table-transact-sql*

--- Organize the data into subjects – customer, product, and orders. ---

2.1. Create a database (if it does not exist) with a name that matches your student ID, and then create tables (using CREATE TABLE script) according to the following specification:

a. Dim_Customer (**CustomerID**, FirstName, LastName, TerritoryName, CounrtyRegionName, Group (Geography))

Source data (tables): SalesTerritory, Customer, and Person

b. Dim_Product (**ProductID**, Name, FinishedGoodsFlag, Class, ListPrice, Color, SubCategoryName, CategoryName)

Source data (tables): Product, ProductSubcategory, and ProductCategory

c. Fact_Orders (ProductID, CustomerID, OrderDate, ShipDate, OrderQty, UnitPrice, UnitPriceDiscount, LineTotal)

Source data (tables): SalesOrderDetail, and SalesOrderHeader

Solution:

Creating Database:

```
IF NOT EXISTS (SELECT *
               FROM   sys.databases
               WHERE  NAME = '253961')
  CREATE DATABASE [245784];

GO

USE [245784];

GO
```

Dim_Customer table:

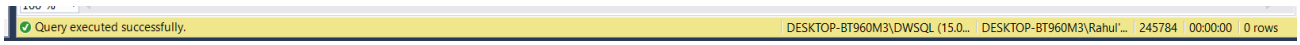
```
CREATE TABLE Dim_Customer (
CustomerID INT NOT NULL CONSTRAINT PK_Dim_Customer PRIMARY KEY,
FirstName NVARCHAR(50) NOT NULL,
LastName NVARCHAR(50) NOT NULL,
TerritoryName NVARCHAR(50) NOT NULL,
CountryRegionCode NVARCHAR(3) NOT NULL,
[Group] NVARCHAR(50) NOT NULL);
```

Dim_Product table:

```
CREATE TABLE Dim_Product (
ProductID INT NOT NULL CONSTRAINT PK_Dim_Product PRIMARY KEY,
[Name] NVARCHAR(50) NOT NULL,
[FinishedGoodsFlag] BIT NOT NULL DEFAULT 1,
[Class] [nchar](2) NULL,
ListPrice MONEY NOT NULL,
Color NVARCHAR(15) NULL,
SubCategoryName NVARCHAR(50) NOT NULL,
CategoryName NVARCHAR(50) NOT NULL
);
```

Fact_orders table:

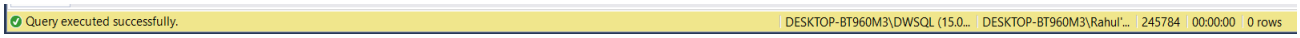
```
CREATE TABLE Fact_Orders (
ProductID INT NOT NULL,
CustomerID INT NOT NULL,
OrderDate DATETIME NOT NULL,
ShipDate DATETIME NULL,
OrderQty SMALLINT NOT NULL,
UnitPrice MONEY NOT NULL,
UnitPriceDiscount MONEY NOT NULL,
LineTotal AS ISNULL(([UnitPrice]*((1.0)-[UnitPriceDiscount]))*[OrderQty],(0.0))
);
```



2.2. Define referential integrity constraints for the Fact_Orders table

Solution:

```
ALTER TABLE dbo.Fact_Orders
ADD
CONSTRAINT FK_ProductID FOREIGN KEY (ProductID) REFERENCES Dim_Product(ProductID),
CONSTRAINT FK_CustomerID FOREIGN KEY (CustomerID) REFERENCES
Dim_Customer(CustomerID);
```



2.3. Fill the created tables with data using the data in the available source tables

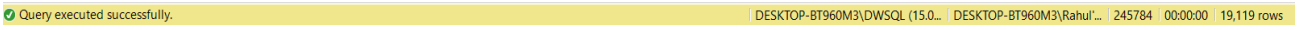
Solution:

Dim_Customer table data Insertion:

```
INSERT INTO [245784].dbo.Dim_Customer
SELECT C.CustomerID,
PP.FirstName,
PP.LastName,
ST.[Name],
ST.[CountryRegionCode],
ST.[Group]
FROM AdventureWorks2019.Sales.Customer AS C
JOIN AdventureWorks2019.Person.Person AS PP
ON PP.BusinessEntityID = C.PersonID
JOIN AdventureWorks2019.Sales.SalesTerritory AS ST
ON ST.TerritoryID = C.TerritoryID;
```

Rec: 4/19119

Customerid	Firstname	lastname	territoryname	countryregioncode	Group
11000	Jon	Yang	Australia	AU	Pacific
11001	Eugene	Huang	Australia	AU	Pacific
11002	Ruben	Torres	Australia	AU	Pacific
11003	Christy	Zhu	Australia	AU	Pacific



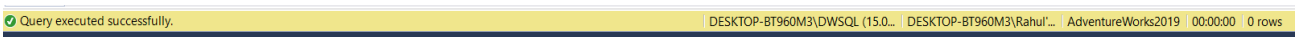
Dim_Product table data insertion:

```
INSERT INTO [245784].dbo.Dim_Product
SELECT [Production].[Product].[ProductID]
       ,[Production].[Product].[Name]
       ,[Production].[Product].[FinishedGoodsFlag]
       ,[Production].[Product].[Class]
       ,[Production].[Product].[ListPrice]
       ,[Production].[Product].[Color]
       ,[Production].[ProductSubcategory].[Name] AS [SubCategoryName]
       ,[Production].[ProductCategory].[Name] AS [CategoryName]
FROM [Production].[Product]
INNER JOIN Production.ProductSubcategory ON Production.Product.ProductSubcategoryID =
Production.ProductSubcategory.ProductSubcategoryID
```

```
INNER JOIN Production.ProductCategory ON Production.ProductSubcategory.ProductCategoryID =
Production.ProductCategory.ProductCategoryID;
```

Rec 4/295

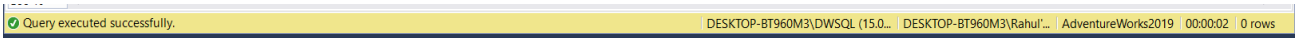
ProductID	Name	FinishedGoodsFlag	Class	ListPrice	Color	SubCategoryName	CategoryName
680	HL Road Frame - Black, 58	1	H	1431.50	Black	Road Frames	Components
706	HL Road Frame - Red, 58	1	H	1431.50	Red	Road Frames	Components
707	Sport-100 Helmet, Red	1	NULL	34.99	Red	Helmets	Accessories
708	Sport-100 Helmet, Black	1	NULL	34.99	Black	Helmets	Accessories



Fact_orders table data Insertion:

```
INSERT INTO [245784].dbo.Fact_Orders
SELECT SOD.ProductID,
SOH.CustomerID,
SOH.OrderDate,
SOH.ShipDate,
SOD.OrderQty,
```

```
SOD.UnitPrice,
SOD.UnitPriceDiscount
FROM AdventureWorks2019.Sales.SalesOrderHeader AS SOH
JOIN AdventureWorks2019.Sales.SalesOrderDetail AS SOD
ON SOH.SalesOrderID = SOD.SalesOrderID;
```



Rec 4/121317

ProductID	CustomerID	OrderDate	ShipDate	OrderQty	UnitPrice	UnitPriceDiscount	LineTotal
707	29614	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	2	20.1865	0.00	40.373000
707	29824	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	1	20.1865	0.00	20.186500
707	29844	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	4	20.1865	0.00	80.746000
707	29580	2011-05-31 00:00:00.000	2011-06-07 00:00:00.000	1	20.1865	0.00	20.186500

Task 3

Let us now look at some reporting capabilities of the prepared data model and the Tableau tool. In particular, we plan to use the order data to create a simple set of visualisations in the aid to analysis of sales from two basic points of view: focus on product and customer. In particular, you are now the BI user, and your goal is to consume the available data to gather some insights about the performance of individual customers. Consider a situation in which you are trying to identify the most prominent customers and their purchase preferences.

Using the prepared data (from the previous task), use Microsoft SQL Server Management Studio (alternatively Azure Data Studio) and Tableau to generate a set of basic analytical reports.

3.1 Prepare an SQL query that results in a report of sales made by individual customers by different products defined as:

Order("Lastname, Firstname", Product Category, Product Name, Sales value)

Solution (Query + 4 records of query execution**):**

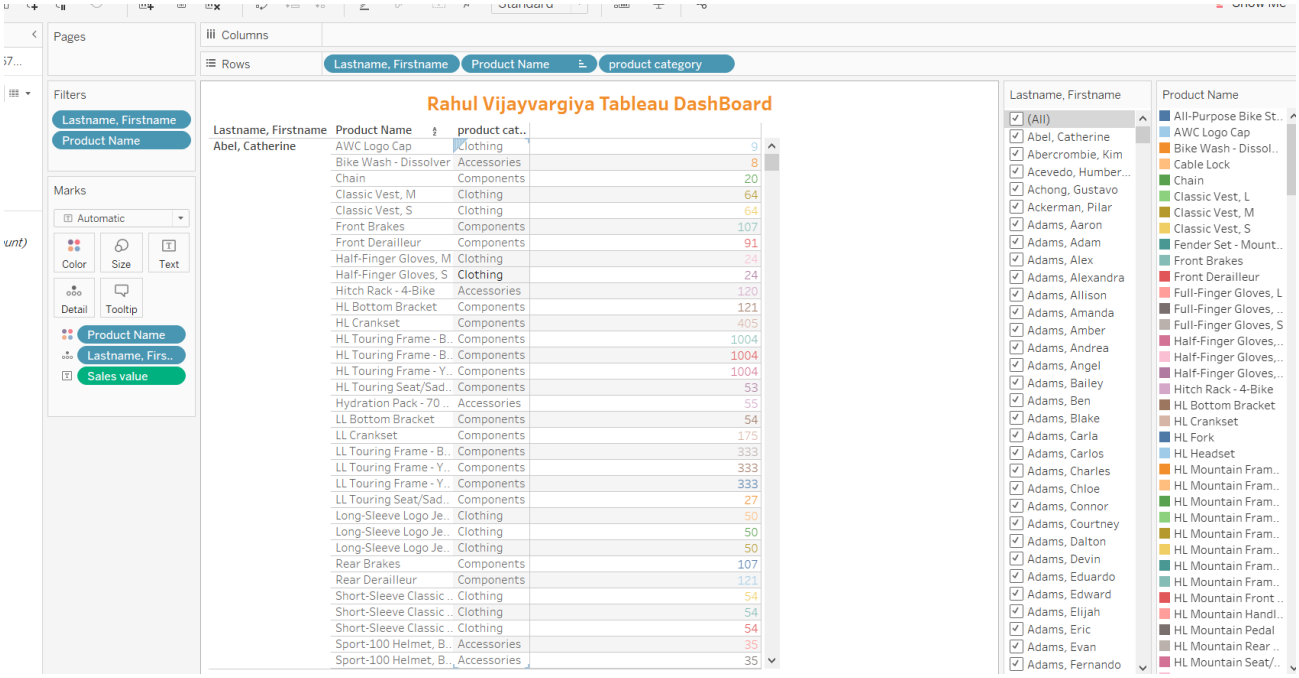
```
SELECT DC.LastName + ', ' + DC.FirstName AS "Lastname, Firstname",
DP.CategoryName "product category",
DP.[Name] "Product Name",
DP.ListPrice "Sales value"
FROM [245784].dbo.Fact_Orders AS FO
JOIN [245784].dbo.Dim_Customer AS DC
ON DC.CustomerID = FO.CustomerID
JOIN [245784].dbo.Dim_Product AS DP
ON DP.ProductID = FO.ProductID;
```

4/121317 rows

Lastname, Firstname	product category	Product Name	Sales value
Calafato, Ryan	Accessories	Sport-100 Helmet, Red	34.99
Heaney, Brenda	Accessories	Sport-100 Helmet, Red	34.99
Hirota, Nancy	Accessories	Sport-100 Helmet, Red	34.99
Bready, Richard	Accessories	Sport-100 Helmet, Red	34.99

3.1 Prepare the same report using Tableau

Solution (a single sheet)**:**



3.2 Prepare a dashboard (arranged and interconnected collection of sheets) to help identify the most prominent customers

Solution (Dashboard containing 2-3 sheets)

CONCLUSIONS:

Use this section to provide your conclusions:

1. Over Clause Along with Round to Filter Data, Round the Result as well
2. The Tableau let us Visualize data and let's us have more control over the data
3. When filling the Dim_Customer JOIN table, we must do BusinessEntityID and PersonID (and not CustomerID), otherwise it will turn out that not all individuals are left saved

---

REMARKS:

- *A report without final conclusions will not be checked and results in a negative score!*
- *The report file should be named **Rep03DW_StudentID_Last name-2022**, please use the PDF format*
- *You should use MS SQL SERVER 2019 (or 2017) and Tableau Desktop (available at https://www.tableau.com/academic/students).*