**Hurtownie danych – Spr. 3.**

PWr. WIZ, Data: 21-22.03.2022

|  |  |  |
| --- | --- | --- |
| Student | ------------------------------------------------------------- | Ocena |
| Indeks | 256305 |  |
| Imię | Grzegorz |
| Nazwisko | Dzikowski |

Zestaw składa się z 4 zadań. Jeżeli nie potrafisz rozwiązać zadania, to próbuj podać, chociaż częściowe rozwiązanie lub uzasadnienie przyczyny braku rozwiązania. Pamiętaj o podaniu nr. indeksu oraz imienia i nazwiska.

Baza danych: **AdventureWorks**

**Zad. 1.**

Należy przygotować raport umożliwiający ocenę pracowników na podstawie sumarycznej sprzedaży zrealizowanej przez pracowników w poszczególnych latach.

Raport powinien zawierać dane zgodnie z tabelą 1. Raport należy przygotować bez użycia funkcji okienkowych (klauzula OVER)

Tab. 1. Wykaz pracowników oraz wypracowane przez nich kwoty w poszczególnych latach wraz z liczbą obsłużonych zamówień

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pracownik | pracID | Rok zamówienia | Kwota | Liczba zamówień |
| Jiang, Stephen | 274 | 2011 | 28926.25 | 4 |
| Jiang, Stephen | 274 | 2012 | 453524.52 | 22 |
| … | … | … | … | … |
| Blythe, Michael | 275 | 2011 | 875823.83 | 65 |
| Blythe, Michael | 275 | 2012 | 3375456.89 | 148 |
| … | … | … | … | … |

Rek: ?/?

**Rozwiązanie:**

USE AdventureWorks2019;

SELECT

CONCAT\_WS(', ', LastName, FirstName) as [Pracownik],

SalesPersonID as pracID,

YEAR(Sales.SalesOrderHeader.OrderDate) as [Rok zamówienia],

SUM(Sales.SalesOrderHeader.SubTotal) as [Kwota],

COUNT(SalesOrderID) as [Liczba zamówień]

FROM Sales.SalesPerson

JOIN Sales.SalesOrderHeader

ON Sales.SalesOrderHeader.SalesPersonID = Sales.SalesPerson.BusinessEntityID

JOIN Person.Person

ON Person.Person.BusinessEntityID = Sales.SalesOrderHeader.SalesPersonID

GROUP BY

SalesPersonID,

YEAR(Sales.SalesOrderHeader.OrderDate),

LastName,

FirstName

ORDER BY SalesPersonID, [Rok zamówienia];

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pracownik | pracID | Rok zamówienia | Kwota | Liczba zamówień |
| Jiang, Stephen | 274 | 2011 | 28926.2465 | 4 |
| Jiang, Stephen | 274 | 2012 | 453524.5233 | 22 |
| Jiang, Stephen | 274 | 2013 | 431088.7238 | 14 |
| Jiang, Stephen | 274 | 2014 | 178584.3625 | 8 |

Rek 4/58

**Zad 2.**

Należy przygotować raport umożliwiający ocenę pracowników na podstawie sumarycznej sprzedaży zrealizowanej przez pracowników w poszczególnych latach (raport powinien zawierać dane zgodnie z tabelą 1.). Raport należy przygotować wykorzystując klauzule OVER

**Rozwiązanie:**

USE AdventureWorks2019;

WITH Raport\_Oceny(Pracownik, pracID, [Rok zamówienia], Kwota, [Liczba zamówień])

AS

(

SELECT

CONCAT\_WS(', ', LastName, FirstName),

SalesPersonID,

YEAR(OrderDate),

SUM(SubTotal) OVER (PARTITION BY SalesPersonID, YEAR(OrderDate)),

COUNT(SalesOrderID) OVER (PARTITION BY SalesPersonID, YEAR(OrderDate))

FROM Sales.SalesOrderHeader

JOIN Person.Person

ON Sales.SalesOrderHeader.SalesPersonID = Person.Person.BusinessEntityID

)

SELECT

DISTINCT \*

FROM Raport\_Oceny

ORDER BY pracID, [Rok zamówienia]

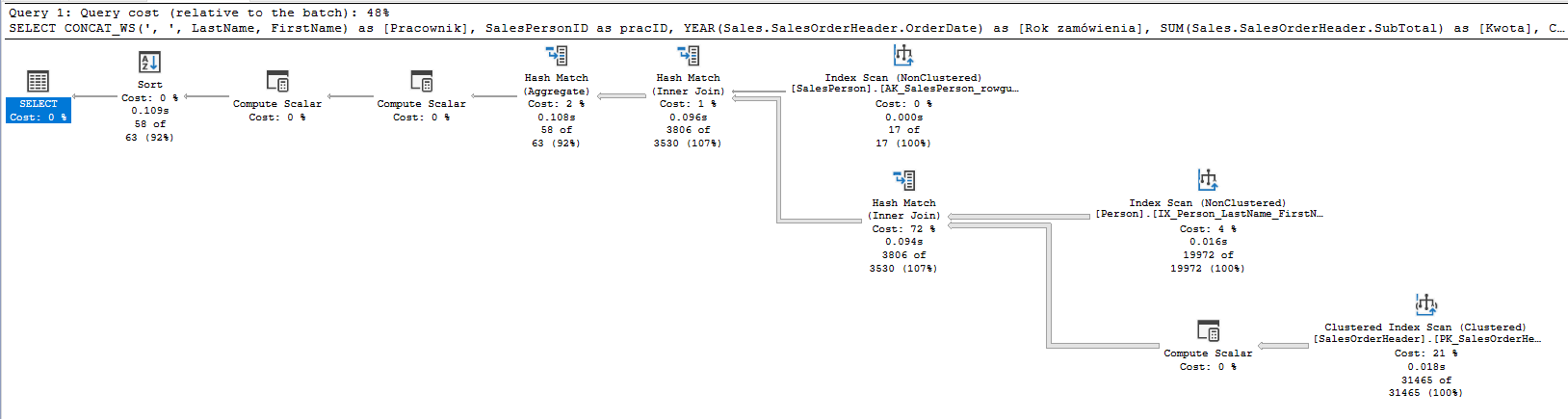
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pracownik | pracID | Rok zamówienia | Kwota | Liczba zamówień |
| Jiang, Stephen | 274 | 2011 | 28926.2465 | 4 |
| Jiang, Stephen | 274 | 2012 | 453524.5233 | 22 |
| Jiang, Stephen | 274 | 2013 | 431088.7238 | 14 |
| Jiang, Stephen | 274 | 2014 | 178584.3625 | 8 |

Rek 4/58

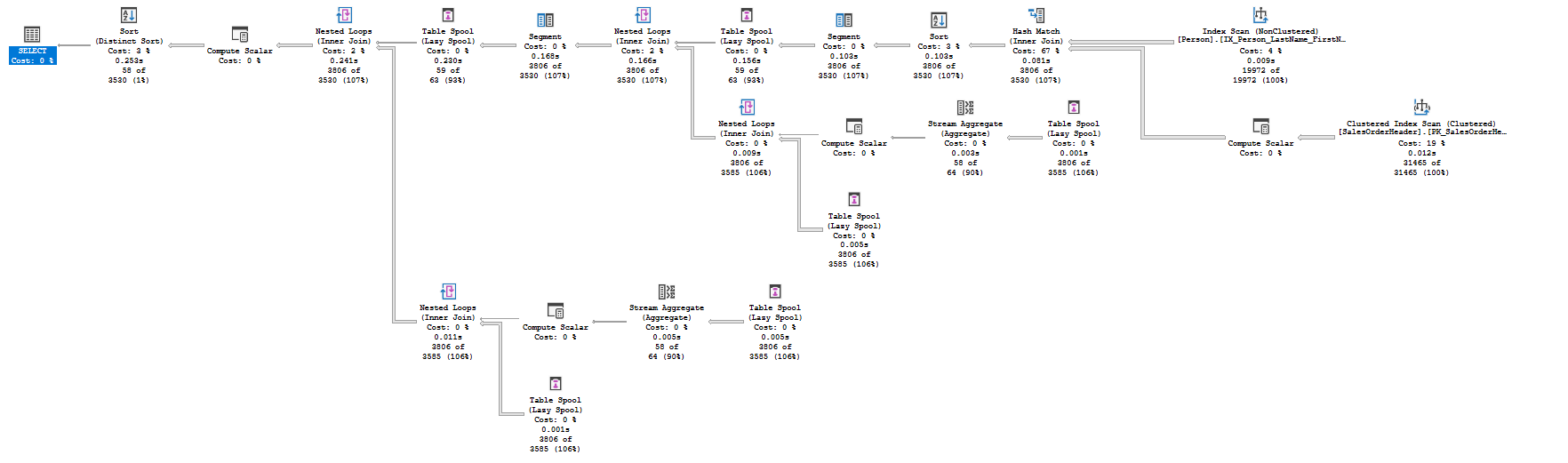
**Zad 3.**

Ocenić jakość rozwiązań zaproponowanych w poprzednich dwóch zadaniach. Które z nich jest korzystniejsze i dlaczego, biorąc poda uwagę plany wykonania obu zapytań

**Rozwiązanie:**



Execution plan zapytania z Group By



Execution plan zapytania z Partition Over

W obu zapytaniach większość kosztu zapytania zabiera Query z obu tabel, oraz JOIN ich obu.

GROUP BY jest realizowany przez jedną operację „Hash Match ‘Aggregate’”. W przypadku partition by, mamy dużo więcej operacji, dużo zagnieżdżonych pętli, oraz kilkukrotne odniesienie do tabeli wejściowej. Całość wykonuje się w takim samym czasie przy tej ilości danych.

Partition by specyfikuje kolumnę do agregacji, i zawsze zwraca tyle samo elementów, ile wejście. Natomiast group by automatycznie zwraca tylko 1 rekord per wartość agregowana.

Według mnie group by sprawdza się lepiej w tradycyjnych agregacjach, gdzie chcemy widzieć tylko jedną wartość agregracji. Natomiast partition by przydaje się w bardziej zaawansowanych zastosowaniach, np. szczegółowe raporty

**Zad. 4.**

Źródło danych:

Baza danych: **AdventureWorks**

Narzędzia**:**

* **Microsoft SQL Server Management Studio, Tableau**

**Etap 1.**

Utworzyć bazę danych (o ile nie istnieje) o nazwie zgodnej z identyfikatorem studenta, a następnie utworzyć tabele (skrypt CREATE TABLE) zgodnie z poniższą specyfikacją:

* 1. Dim\_Customer (CustomerID, FirstName, LastName, TerritoryName, CounrtyRegionCode, Group)

Tabele źródłowe:

* SalesTerritory
* Customer
* Person

**Rozwiązanie:**

USE [256305];

WITH Customer\_Source

(

CustomerID,

FirstName,

LastName,

TerritoryName,

CountryRegionCode,

[Group]

) AS

(

SELECT

CustomerID,

FirstName,

LastName,

[AdventureWorks2019].[Sales].[SalesTerritory].[Name],

[AdventureWorks2019].[Sales].[SalesTerritory].[CountryRegionCode],

[AdventureWorks2019].[Sales].[SalesTerritory].[Group]

FROM [AdventureWorks2019].[Sales].[Customer]

LEFT JOIN [AdventureWorks2019].[Person].[Person]

ON [AdventureWorks2019].[Person].[Person].BusinessEntityID = [AdventureWorks2019].[Sales].[Customer].CustomerID

LEFT JOIN [AdventureWorks2019].[Sales].[SalesTerritory]

ON [AdventureWorks2019].[Sales].[SalesTerritory].TerritoryID = [AdventureWorks2019].[Sales].[Customer].TerritoryID

)

SELECT \* INTO [dbo].[Dim\_Customer] FROM Customer\_Source;

ALTER TABLE dbo.Dim\_Customer

ADD CONSTRAINT Constraint\_PK\_CustomerID PRIMARY KEY(CustomerID);

Wynik:

(19820 rows affected)

* 1. Dim\_Product (**ProductID**, Name, ListPrice, Color, SubCategoryName, CategoryName)

Tabele źródłowe:

* Production.Product
* Production.ProductSubcategory
* Production.ProductCategory

**Rozwiązanie:**

USE [256305];

WITH Product\_Source

(

ProductID,

[Name],

ListPrice,

Color,

SubCategoryName,

CategoryName

)

AS

(

SELECT

ProductID,

[AdventureWorks2019].[Production].[Product].[Name],

[AdventureWorks2019].[Production].[Product].[ListPrice],

[AdventureWorks2019].[Production].[Product].[Color],

[AdventureWorks2019].[Production].[ProductSubcategory].[Name],

[AdventureWorks2019].[Production].[ProductCategory].[Name]

FROM [AdventureWorks2019].[Production].[Product]

JOIN [AdventureWorks2019].[Production].[ProductSubcategory]

ON [AdventureWorks2019].[Production].[Product].ProductSubcategoryID = [AdventureWorks2019].[Production].[ProductSubcategory].ProductSubcategoryID

JOIN [AdventureWorks2019].[Production].[ProductCategory]

ON [AdventureWorks2019].[Production].[ProductCategory].ProductCategoryID = [AdventureWorks2019].[Production].[ProductSubcategory].ProductCategoryID

)

SELECT \* INTO [dbo].[Dim\_Product] FROM Product\_Source;

ALTER TABLE dbo.Dim\_Product

ADD CONSTRAINT PK\_DimProduct\_ProductID PRIMARY KEY(ProductID)

ALTER TABLE dbo.Dim\_Product

ADD CONSTRAINT Check\_Product\_ListPrice CHECK ([ListPrice]>=(0.00))

Wynik:

(295 rows affected)

* 1. Fact\_Orders (ProductID, CustomerID, OrderDate, ShipDate, OrderQty, UnitPrice, UnitPriceDiscount, LineTotal)

Tabele źródłowe:

* Sales.SalesOrderDetail
* Sales.SalesOrderHeader

**Rozwiązanie:**

USE [256305];

WITH Orders\_Source

(

ProductID,

CustomerID,

OrderDate,

ShipDate,

OrderQty,

UnitPrice,

UnitPriceDiscount,

LineTotal

)

AS

(

SELECT

ProductID,

CustomerID,

OrderDate,

ShipDate,

OrderQty,

UnitPrice,

UnitPriceDiscount,

LineTotal

FROM [AdventureWorks2019].[Sales].[SalesOrderHeader]

JOIN [AdventureWorks2019].[Sales].[SalesOrderDetail]

ON [AdventureWorks2019].[Sales].[SalesOrderDetail].SalesOrderID = [AdventureWorks2019].[Sales].[SalesOrderHeader].SalesOrderID

)

SELECT \* INTO [dbo].[Fact\_Orders] FROM Orders\_Source;

ALTER TABLE [dbo].[Fact\_Orders]

ADD CONSTRAINT [Check\_FO\_UnitPrice] CHECK ([UnitPrice]>=(0.00));

ALTER TABLE [dbo].[Fact\_Orders]

ADD CONSTRAINT [Check\_FO\_UnitPriceDiscount] CHECK ([UnitPriceDiscount]>=(0.00));

ALTER TABLE [dbo].[Fact\_Orders]

ADD CONSTRAINT [Check\_FO\_OrderQty] CHECK ([OrderQty]>(0));

ALTER TABLE [dbo].[Fact\_Orders]

ADD CONSTRAINT [Check\_FO\_ShipDate] CHECK (([ShipDate]>=[OrderDate] OR [ShipDate] IS NULL))

ALTER TABLE [dbo].[Fact\_Orders]

ADD CONSTRAINT [Check\_FO\_OrderDateDefault] DEFAULT (getdate()) FOR [OrderDate];

(121317 rows affected)

1. Zdefiniować ograniczenia referencyjne dla tabeli Fact\_Orders

**Rozwiązanie:**

USE [256305];

ALTER TABLE [dbo].[Fact\_Orders]

ADD CONSTRAINT [FK\_FactOrders\_To\_DimCustomer]

FOREIGN KEY([CustomerID])

REFERENCES [dbo].[Dim\_Customer] ([CustomerID]);

ALTER TABLE [dbo].[Fact\_Orders]

ADD CONSTRAINT [FK\_FactOrders\_To\_DimProduct]

FOREIGN KEY([ProductID])

REFERENCES [dbo].[Dim\_Product] ([ProductID]);

Wynik:

Commands completed successfully.

1. Utworzone tabele napełnić danymi wykorzystując dane w podanych tabelach źródłowych
2. Wygenerować raport z danych w utworzonej bazie danych zgodnie z podaną definicją:

**Zamówienia(„Nazwisko, imię”, Kategoria produktu, Nazwa produktu, Cena)**

Rozwiązanie przedstawić w formie tabelarycznej (definicja zapytania i kopia wyniku) oraz w formie wizualnej (2-3 diagramy przygotowane w Tableau wraz z analizą wyników i wnioskami).

**Rozwiązanie 1. (zapytanie SQL + fragment wyniku 4-5 rekordów):**

USE [256305];

WITH Order\_Source

(

[KlientName],

[Kategoria produktu],

[Nazwa Produktu],

[Cena]

)

AS

(

SELECT

CONCAT\_WS(', ', LastName, FirstName),

CategoryName,

[dbo].[Dim\_Product].[Name],

LineTotal

FROM dbo.Fact\_Orders

JOIN dbo.Dim\_Customer

ON dbo.Dim\_Customer.CustomerID = dbo.Fact\_Orders.CustomerID

JOIN dbo.Dim\_Product

ON dbo.Dim\_Product.ProductID = dbo.Fact\_Orders.ProductID

)

SELECT

IIF(KlientName ='', ' Anonimowy', KlientName) as Klient,

[Kategoria produktu],

[Nazwa Produktu],

[Cena]

FROM Order\_Source

ORDER BY

IIF(KlientName = '', 0, 1) DESC,

Klient,

[Kategoria produktu],

[Nazwa Produktu],

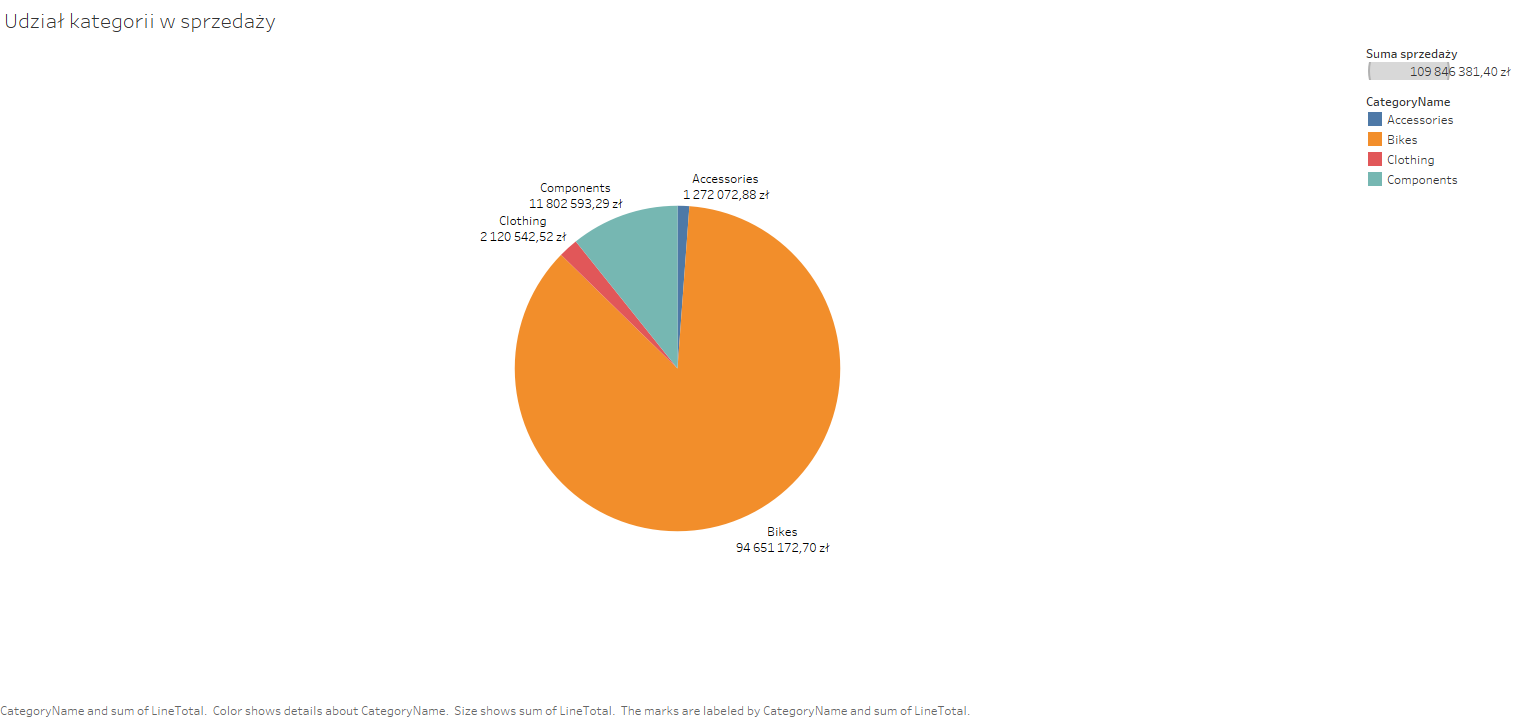
[Cena];

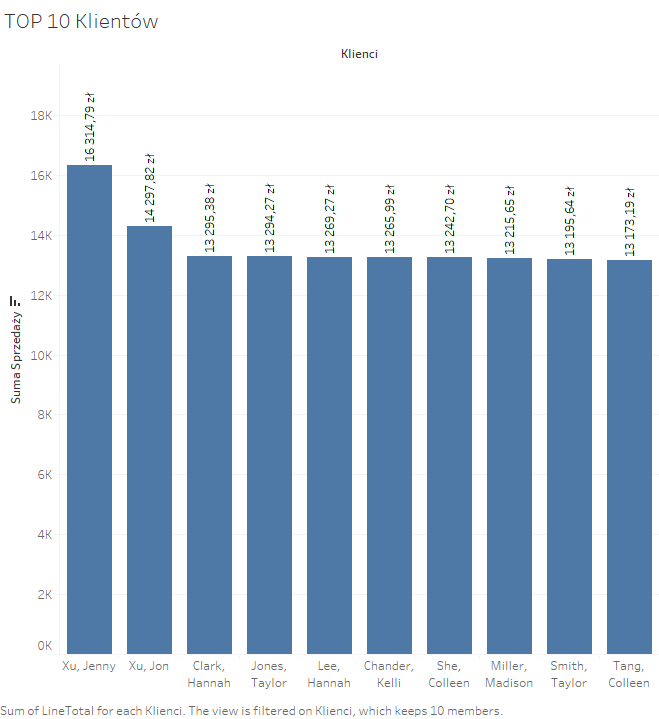
Używam takiej składni, żeby klienci anonimowi (bez imienia nazwiska, regionu) byli na samym końcu tabelki

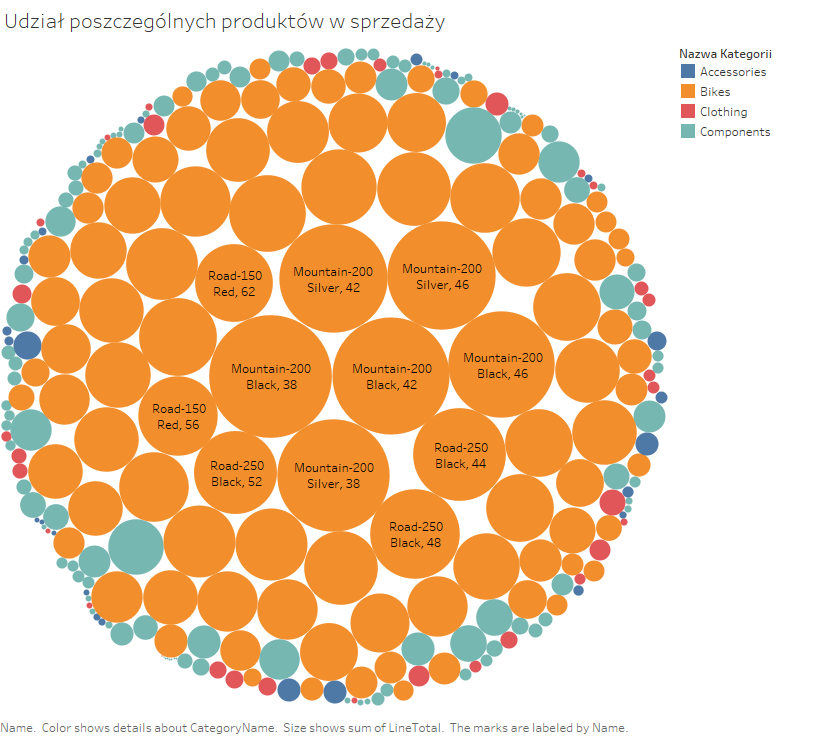
|  |  |  |  |
| --- | --- | --- | --- |
| Klient | Kategoria produktu | Nazwa Produktu | Cena |
| Adams, Aaron | Accessories | Fender Set - Mountain | 21.980000 |
| Adams, Aaron | Accessories | Mountain Bottle Cage | 9.990000 |
| Adams, Aaron | Accessories | Water Bottle - 30 oz. | 4.990000 |
| Adams, Aaron | Bikes | Mountain-200 Black, 46 | 2294.990000 |

**Rek: 4/121317**

**Rozwiązanie 2. (prezentacja wyników analizy danych - Tableau):**

****





**Wnioski:**

Partition over przydaje się przy bardzo niestandardowych kwerendach agregacyjnych i prezentacji wyników, jednak do standardowego agregowania danych group by wystarcza z nadmiarem. Kopiowanie danych przed analizą jest ważne, aby po pierwsze, pracować na kopii danych, a nie na produkcji, a po drugie, pozwala na wykonania wycinka rzeczywistości w danym momencie czasu. Tablau w bardzo przyjazny sposób pozwala analizować dane z bazy danych, i graficznie przedstawiać je na wykresach, pie-chartach czy nawet chmurze

*Uwaga:*

* Sprawozdanie bez wniosków końcowych nie będzie sprawdzane i tym samym ocena jest negatywna!