**DSLS Mini Project Report**

**Data Engineering**

Gatum Erlangga

# **Intermediate Queries**

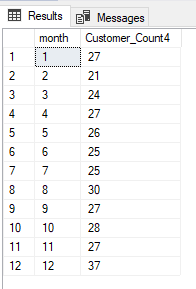
1. Tulis query untuk mendapatkan jumlah customer tiap bulan yang melakukan order pada tahun 1997.

SELECT Month(OrderDate) month, COUNT(DISTINCT CustomerID) Customer\_Count

FROM Orders

WHERE Year(OrderDate) = 1997

GROUP BY Month(OrderDate)

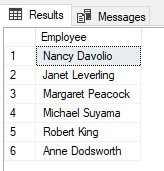


1. Tulis query untuk mendapatkan nama employee yang termasuk Sales Representative.

SELECT CONCAT(FirstName, ' ', LastName) Employee

FROM Employees

WHERE Title = 'Sales Representative'



1. Tulis query untuk mendapatkan top 5 nama produk yang quantitynya paling banyak diorder pada bulan Januari 1997.

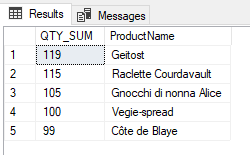
SELECT TOP 5 SUM(od.Quantity) QTY\_SUM, p.ProductName

FROM ([Order Details] od INNER JOIN Products p on od.ProductID = p.ProductID) INNER JOIN Orders o on o.OrderID = od.OrderID

WHERE YEAR(o.OrderDate) = 1997 and MONTH(o.OrderDate) = 1

GROUP BY p.ProductName

ORDER BY SUM(od.Quantity) DESC



1. Tulis query untuk mendapatkan nama company yang melakukan order Chai pada bulan Juni 1997.

SELECT DISTINCT c.CompanyName

FROM [Order Details] od INNER JOIN (Orders o INNER JOIN Customers c on o.CustomerID = c.CustomerID) on od.OrderID = o.OrderID

WHERE YEAR(o.OrderDate) = 1997 and MONTH(o.OrderDate) = 6



1. Tulis query untuk mendapatkan jumlah OrderID yang pernah melakukan sales (unit\_price dikali quantity) <=100, 100<x<=250, 250<x<=500, dan >500.

SELECT count(sc.SalesCategory) as SalesCategoryCount, sc.SalesCategory

FROM (

SELECT \*, (UnitPrice \* Quantity) Sales,

CASE

WHEN (UnitPrice \* Quantity) <=100 THEN '<=100'

WHEN (UnitPrice \* Quantity) BETWEEN 100 and 251 THEN '100<x<=250'

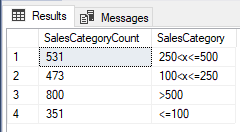
WHEN (UnitPrice \* Quantity) BETWEEN 250 and 501 THEN '250<x<=500'

ELSE '>500'

END as SalesCategory

FROM [Order Details]) sc

GROUP BY sc.SalesCategory



1. Tulis query untuk mendapatkan Company name yang melakukan sales di atas 500 pada tahun 1997.

SELECT c.CompanyName, SUM(s.Sales)

FROM

(SELECT SUM(UnitPrice \* Quantity) Sales, o.OrderID

FROM [Order Details] od INNER JOIN Orders o on od.OrderID = o.OrderID

WHERE YEAR(o.OrderDate) = 1997

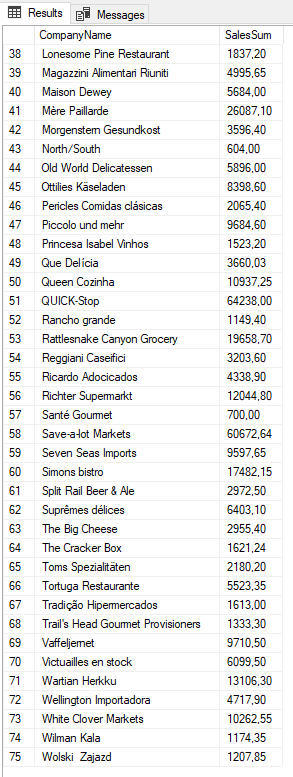
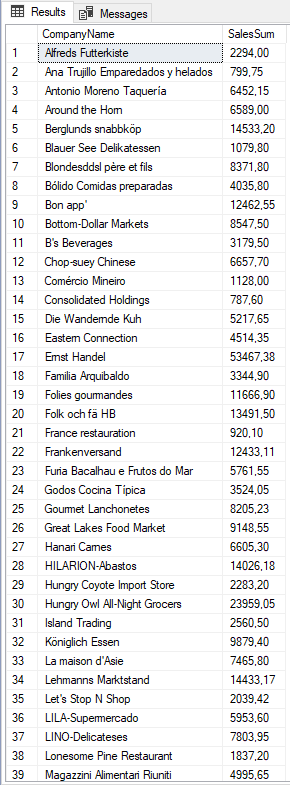
GROUP BY o.OrderID) s

INNER JOIN (Orders o

INNER JOIN Customers c on c.CustomerID = o.CustomerID) on s.OrderID = o.OrderID

GROUP BY c.CompanyName

HAVING SUM(s.Sales) > 500



1. Tulis query untuk mendapatkan nama produk yang merupakan Top 5 sales tertinggi tiap bulan di tahun 1997.

SELECT RS.\*, p.ProductName

FROM

(SELECT

\*,

ROW\_NUMBER() OVER (PARTITION BY ps.month Order by Sales DESC) AS Rank

FROM (

SELECT SUM(Quantity \* UnitPrice) as Sales, YEAR(o.OrderDate) year, MONTH(o.OrderDate) month, od.ProductID

FROM

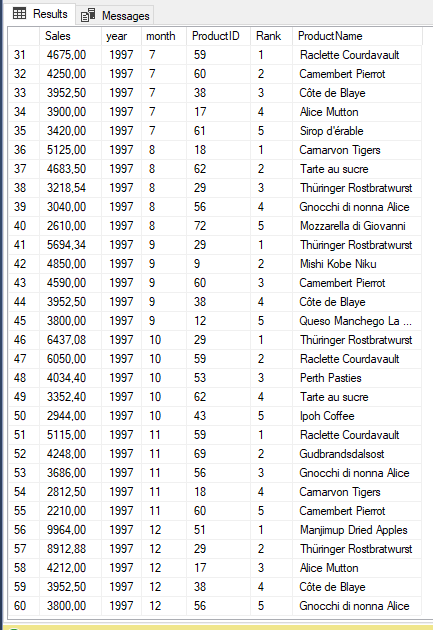
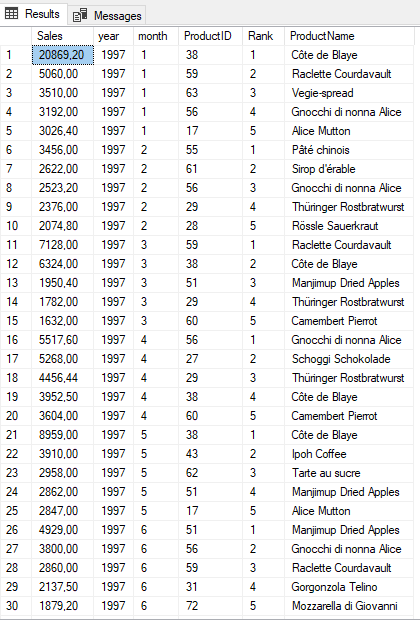
[Order Details] od INNER JOIN Orders o on od.OrderID = o.OrderID

WHERE YEAR(OrderDate) = 1997

GROUP BY od.ProductID, YEAR(o.OrderDate), MONTH(o.OrderDate)) ps) RS

INNER JOIN Products p on RS.ProductID = p.ProductID

WHERE RS.Rank <= 5



1. Buatlah view untuk melihat Order Details yang berisi OrderID, ProductID, ProductName, UnitPrice, Quantity, Discount, Harga setelah diskon.

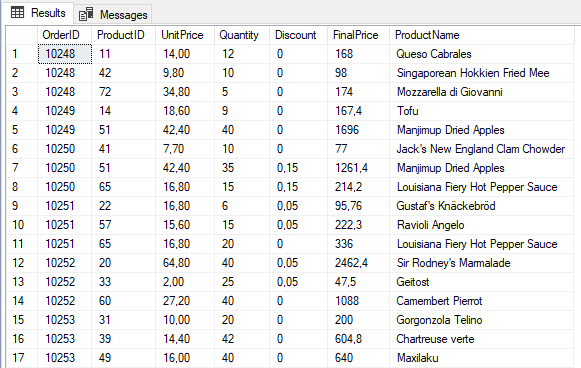
CREATE VIEW OrderDetailsView

AS

SELECT od.\*, od.UnitPrice \* od.Quantity \* (1-od.Discount) as FinalPrice, p.ProductName

FROM [Order Details] od INNER JOIN Products p on od.ProductID = p.ProductID

GO



1. Buatlah procedure Invoice untuk memanggil CustomerID, CustomerName, OrderID, OrderDate, RequiredDate, ShippedDate jika terdapat inputan CustomerID tertentu.

CREATE PROCEDURE Invoice @id NVARCHAR(30)

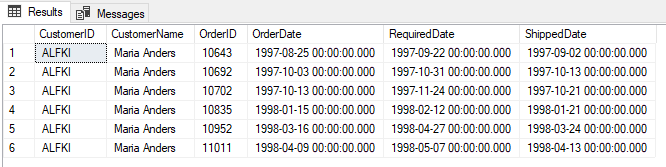
AS

SELECT c.CustomerID, c.ContactName as CustomerName, o.OrderID, o.OrderDate, o.RequiredDate, o.ShippedDate

FROM Orders o INNER JOIN Customers c on o.CustomerID = c.CustomerID

WHERE c.CustomerID = @id;

EXEC Invoice 'ALFKI';



# **Case Studies**

### **Customer Analysis: Loyal Customer**

**Description**

Pada case study ini, penulis ingin melihat apakah ada transaksi yang konsisten dilakukan oleh customer pada setiap bulan. Tabel yang digunakan dalam penelusuran ini adalah Order Details, Orders dan Customer. Ketiga tabel ini digabungkan dan diaggregasi untuk mendapatkan total Sales (quantity \* unit price), bulan, dan customer id dari customer yang melakukan transaksi.

**Query**

SELECT ODR.\*

FROM (

SELECT SUM(ods.Quantity \* ods.UnitPrice) Sales, MONTH(ods.OrderDate) Month, ods.CustomerID,

ROW\_NUMBER() OVER (PARTITION BY MONTH(ods.OrderDate) ORDER BY SUM(ods.Quantity \* ods.UnitPrice) desc) as Rank

FROM (

SELECT o.OrderID, od.UnitPrice, od.Quantity, o.CustomerID, o.EmployeeID, o.OrderDate

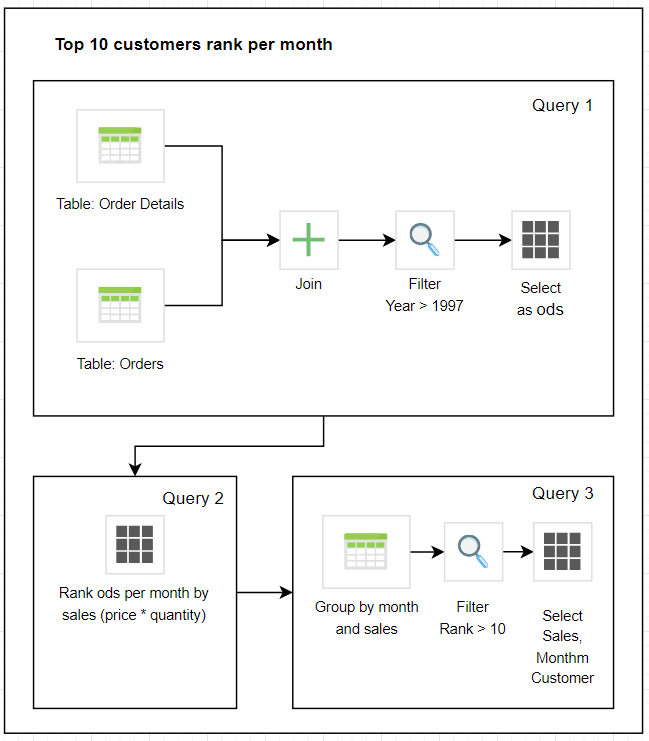
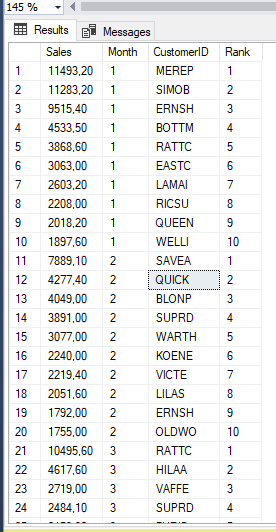
FROM [Order Details] od INNER JOIN Orders o on od.OrderID = o.OrderID

WHERE YEAR(o.OrderDate) = 1997

) ods

GROUP BY MONTH(ods.OrderDate), ods.CustomerID ) ODR

WHERE ODR.Rank <= 10



**Sample Output (kiri) dan Flowchart (kanan)**

### **Product Analysis: Product Category Trends**

**Description**

Penelusuran ini bertujuan untuk melihat trend penjualan produk berdasarkan kategorinya dengan batasan tahun 1997 dan 1998 pada setiap bulan. Tabel yang digunakan adalah Order Details, Products, dan Categories. Seluruh tabel ini diaggregasi untuk mendapatkan category name, tahun, bulan, dan jumlah quantity per bulan untuk setiap kategori produk.

**Query**

-- Tren Jenis produk setiap bulan 1997 - 1998

SELECT c.CategoryName, os.Year, os.Month, SUM(OS.Quantity) Quantity

FROM

(SELECT od.ProductID, SUM(od.Quantity) AS Quantity, YEAR(o.OrderDate) AS Year, MONTH(o.OrderDate) as Month

FROM [Order Details] od INNER JOIN Orders o on od.OrderID = o.OrderID

WHERE YEAR(o.OrderDate) = 1997 or YEAR(o.OrderDate) = 1998

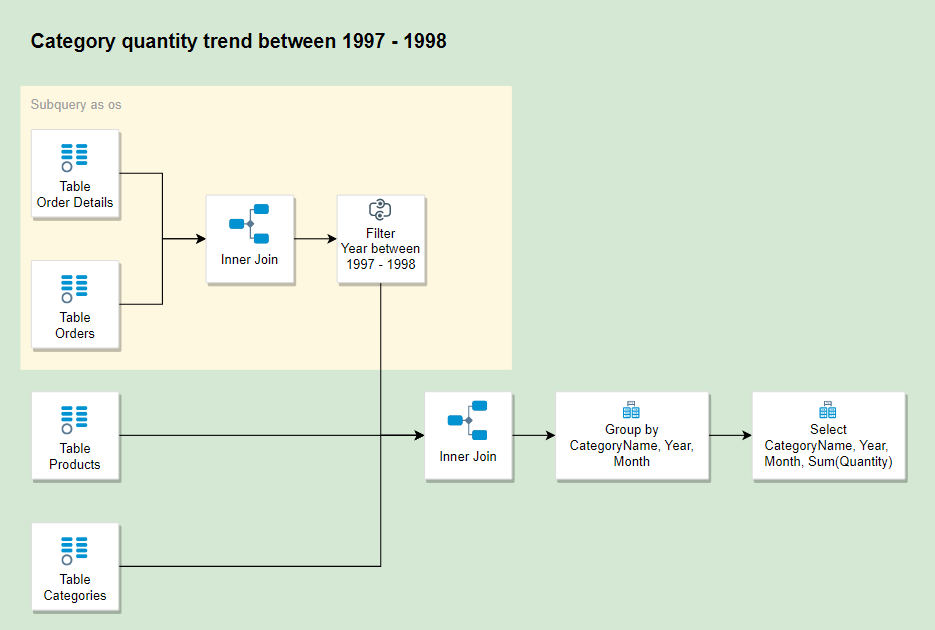
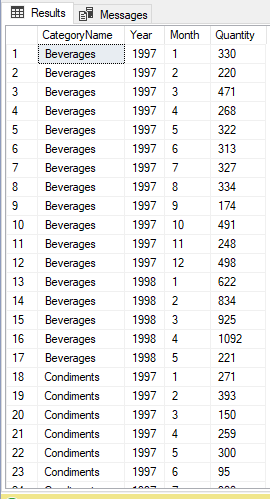
GROUP BY od.ProductID, YEAR(o.OrderDate), MONTH(o.OrderDate)) os

INNER JOIN Products p on p.ProductID = os.ProductID

INNER JOIN Categories c on c.CategoryID = p.CategoryID

GROUP BY c.CategoryName, os.Year, os.Month

ORDER BY c.CategoryName, os.Year, os.Month

**Sample Output (kiri) dan Flowchart (kanan)**

### **Employee Analysis**

**Description**

Perlunya mengetahui beban kerja dari setiap role akan membantu perusahaan dalam menentukan kapan harus menambah pegawai dan tidak. Selain itu, tugas dari setiap pegawai juga dapat lebih disesuaikan dengan beban kerja. Dari permasalahan tersebut, perlu dilihat beban kerja dari setiap role/title pegawai yang sudah ada saat ini. Untuk mendapatkan beban kerja berdasarkan jumlah order, maka diperlukan tabel Employee dan Orders. Dari kedua tabel ini akan didapatkan jumlah order untuk setiap title pegawai.

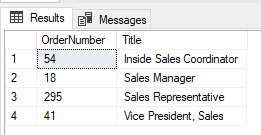
**Query**

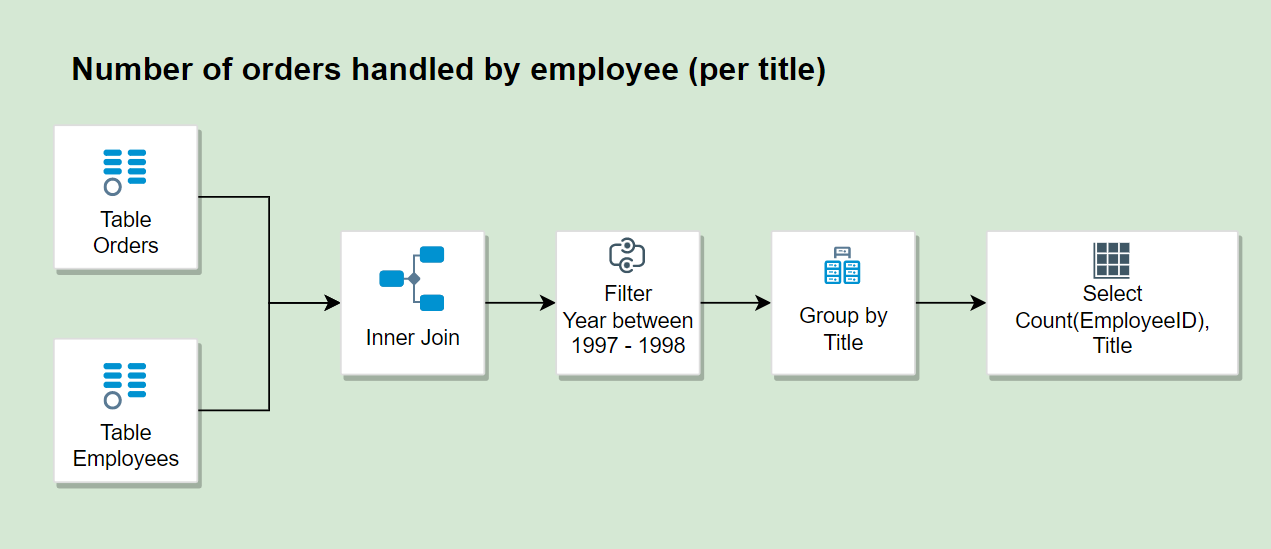
SELECT COUNT(o.EmployeeID) OrderNumber, e.Title

FROM Orders o INNER JOIN Employees e on o.EmployeeID = e.EmployeeID

WHERE YEAR(o.OrderDate) = 1997

GROUP BY e.Title





**Sample Output (atas) dan Flowchart (bawah)**

# **Analysis Report**

#### **Presentation**

<https://docs.google.com/presentation/d/1h4Pudoa-DEFtCDyC3aLapeHo9WknWnOHVGriPzzMGqY/edit?usp=sharing>

#### **Python Code**

<https://colab.research.google.com/drive/1VT2JXbJy_z9z6-U30WwfibvpQo3ARCXs?usp=share_link>

#### **Project’s GitHub Link**

<https://github.com/erlanggagatum/dsls-mini-project-de>