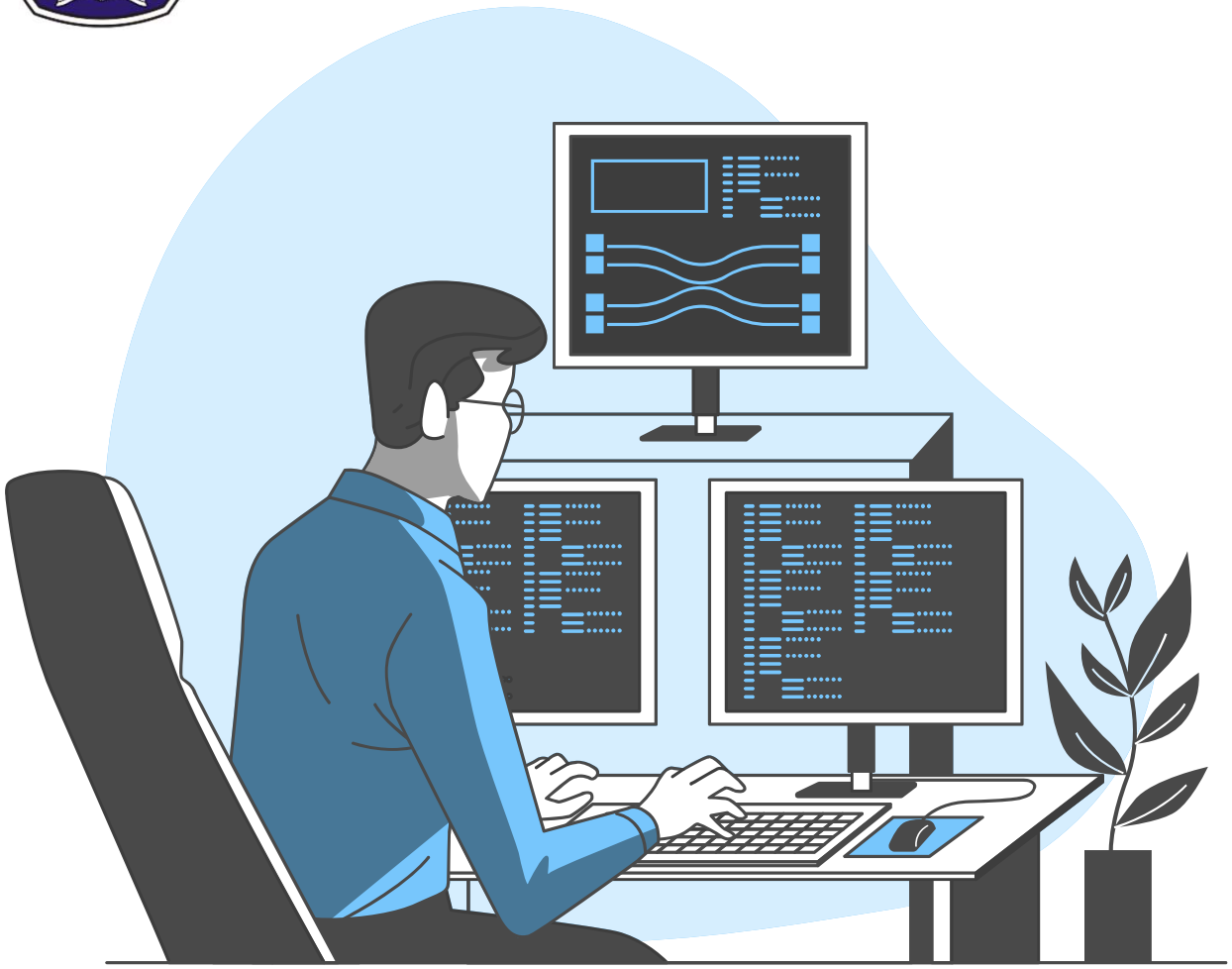
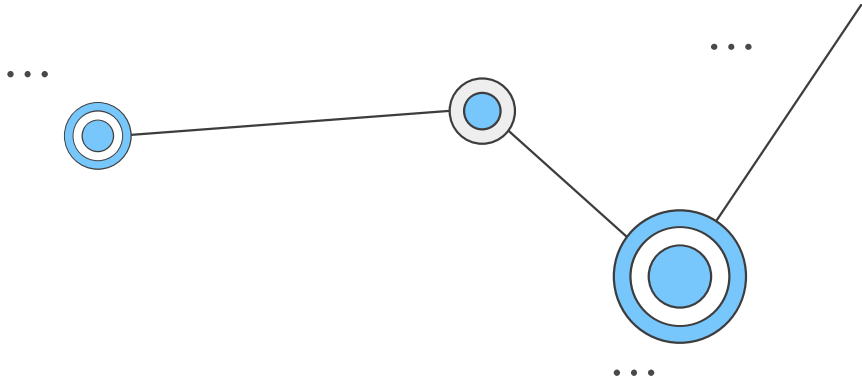


BASIS DATA LANJUT

Pertemuan 4

Sub-queries, groupping, & Aggregating

TIM AJAR BASIS DATA LANJUT
JTI POLINEMA



OUTLINE

Aggregating

- Fungsi-fungsi agregasi

Groupping

- GROUP BY
- HAVING

Sub-queries

T-SQL

Fungsi Aggregasi

Fungsi Agregasi

Agregasi → Total, rangkuman, rekap

Fungsi agregasi → Fungsi yang mengembalikan hasil hitungan **rekapitulasi** dari banyak nilai/baris menjadi 1 nilai

<u>Contoh</u>:
Daftar nilai mahasiswa: 3, 3, 2, 3.5
Diagregasi secara:

- Rata-rata: (3 + 3 + 2.5 + 3.5) / 4 = 3
- Total: (3 + 3 + 3 + 2.5 + 3.5) = 12
- Jumlah data: 4
- Nilai terkecil: 2.5
- Dst...dst...

Fungsi-fungsi agregasi bawaan SQL Server

Common	Statistical	Other
• SUM • MIN • MAX • AVG • COUNT • COUNT_BIG	• STDEV • STDEVP • VAR • VARP	• CHECKSUM_AGG • GROUPING • GROUPING_ID

Menggunakan Fungsi Agregasi

Fungsi Agregasi:

- Mengembalikan nilai tunggal (skalar) (tanpa nama kolom)
- Mengabaikan NULLs kecuali pada fungsi COUNT(*)

Dapat digunakan pada klausa:

- SELECT, HAVING, dan ORDER BY
- Sering kali digunakan pada klausa GROUP BY

```
SELECT COUNT (DISTINCT SalesOrderID) AS
UniqueOrders,
AVG(UnitPrice) AS Avg_UnitPrice,
MIN(OrderQty)AS Min_OrderQty,
MAX(LineTotal) AS Max_LineTotal
FROM Sales.SalesOrderDetail;
```

```
UniqueOrders Avg_UnitPrice Min_OrderQty Max_LineTotal
------------- ------------ ------------ -------------
31465         465.0934         1         27893.619000
```

Fungsi agregasi + ‘Distinct’

Gunakan DISTINCT pada fungsi agregasi untuk meng-agregasikan nilai yang unik saja
Agregasi dengan DISTINCT hanya *menghilangkan nilai yang sama*, BUKAN baris yang sama (tidak seperti SELECT DISTINCT)
Seperti pada contoh berikut(hasil tidak ditampilkan semua):

```
SELECT SalesPersonID, YEAR(OrderDate) AS OrderYear,
COUNT(CustomerID) AS All_Custs,
COUNT(DISTINCT CustomerID) AS Unique_Custs
FROM Sales.SalesOrderHeader
GROUP BY SalesPersonID, YEAR(OrderDate);
```

```
SalesPersonID  OrderYear  All_Custs  Unique_custs
-----------   ----------- ----------- ------------
289            2006        84          48
281            2008        52          27
285            2007         9           8
277            2006       140          57
```

Klausa Group by

GROUP BY mengelompokkan nilai/baris yang diproses oleh fungsi agregasi berdasarkan kondisi/kolom tertentu.

```
SELECT <select_list>
FROM <table_source>
WHERE <search_condition>
GROUP BY <group_by_list>;
```

GROUP BY mengelompokkan dulu nilai-nilai, setelah itu diberikan ke fungsi agregasinya
Misal: Menghitung berapa jumlah mahasiswa berdasarkan nilai hurufnya.

```
SELECT NilaiHuruf, COUNT(*) AS JumlahMhs
FROM Nilai
GROUP BY NilaiHuruf;
```

Baris-baris yang lain akan “hilang” setelah GROUP BY diproses

Klausa Group by

NIM	NamaMahasiswa	NilaiHuruf
1	Adi	A
2	Budi	C
3	Charlie	B
4	Doni	A
5	Evan	A
6	Fuad	C

Count(*) & GROUP BY NilaiHuruf

NilaiHuruf	JumlahMhs
A	3
B	1
C	2

T-SQL

Group BY & Having

‘Group by’ dan prioritas Logis operasi

HAVING, SELECT, dan ORDER BY harus mengembalikan sebuah nilai tunggal per group
Semua kolom pada SELECT, HAVING, dan ORDER BY harus berada dalam klausa GROUP BY atau menjadi input dari fungsi/ekspresi agregasi

Logical Order	Phase	Comments
5	SELECT	
1	FROM	
2	WHERE	
3	GROUP BY	Creates groups
4	HAVING	Operates on groups
6	ORDER BY	

‘Group by’ dan prioritas Logis operasi

SQL berikut akan mengembalikan error:

```
SELECT empid, custid, COUNT(*) AS cnt
FROM Sales.Order
GROUP BY empid;
```

Errornya:
Msg 8120, Level 16, State 1, Line 3
Column 'Sales.Orders.**custid**' is invalid in the select list because it is *not contained in either an aggregate function or the GROUP BY* clause.
Mengapa?
Karena kolom **custid** tidak ada di klausa GROUP BY

Menggunakan ‘group by’ bersama fungsi agregasi

Fungsi agregasi umum digunakan di statement SELECT, untuk menghitung berdasarkan kelompok:

```
SELECT CustomerID, COUNT(*) AS cnt
FROM Sales.SalesOrderHeader
GROUP BY CustomerID;
```

Fungsi agregasi boleh meng-agregasi kolom apa saja. Tidak hanya yang di dalam GROUP BY saja.

```
SELECT productid, MAX(OrderQty) AS largest_order
FROM Sales.SalesOrderDetail
GROUP BY productid;
```

Menyaring kelompok data dengan klausa 'Having'

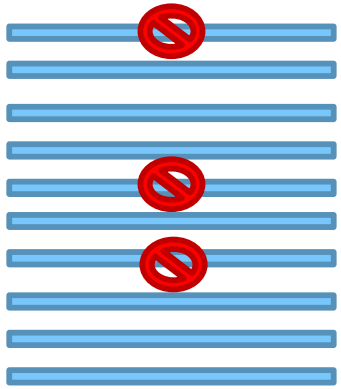
Klausa HAVING membantu kita untuk memfilter grup-grup yang didapat dari GROUP BY berdasarkan kondisi tertentu.
Klausa HAVING diproses setelah GROUP BY
Misal: Mencari pelanggan yang sudah pernah belanja lebih dari 10x...

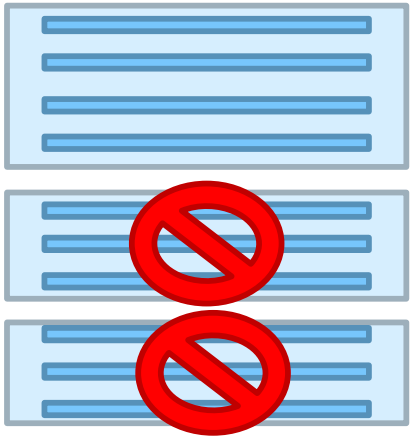
NoStruk	IdPelanggan	TotalBelanja
1192	Ani	90.000
1193	Budi	100.000
1194	Ani	25.000
Dst...	Dst...	Dst...

```
SELECT IdPelanggan, COUNT(*) AS
Jumlah_Transaksi
FROM Penjualan
GROUP BY IdPelanggan
HAVING COUNT(*) > 10;
```

Klausa ‘Having’ vs. ‘WHERE’

WHERE memfilter baris-baris sebelum grup dibuat
Memilih baris mana yang ditampilkan/dimasukkan grup/dikembalikan
HAVING filters groups
Memilih grup mana saja yang ditampilkan/dikembalikan





Klausa 'Having' vs. 'WHERE'

Menggunakan fungsi/ekspresi COUNT(*) pada klausa HAVING sangat berguna untuk menyelesaikan permasalahan bisnis umum:
Misal: Tampilkan customer yang sudah pesan lebih dari sekali:

```
SELECT Cust.Customerid, COUNT(*) AS cnt
FROM Sales.Customer AS Cust
JOIN Sales.SalesOrderHeader AS Ord ON Cust.CustomerID =
ORD.CustomerID
GROUP BY Cust.CustomerID
HAVING COUNT(*) > 1;
```

Misal: Tampilkan produk yang sudah lebih dari 10x dipesan

```
SELECT Prod.ProductID, COUNT(*) AS cnt
FROM Production.Product AS Prod
JOIN Sales.SalesOrderDetail AS Ord ON Prod.ProductID = Ord.ProductID
GROUP BY Prod.ProductID
HAVING COUNT(*) >= 10;
```

Contoh HAVING

```sql
SELECT
    column_name1,
    column_name2,
    aggregate_function (column_name3) column_alias
FROM
    table_name
GROUP BY
    column_name1,
    column_name2
HAVING
    column_alias > value;
```

Instead, you must use the aggregate function expression in the `HAVING` clause explicitly as follows:

```sql
SELECT
    column_name1,
    column_name2,
    aggregate_function (column_name3) alias
FROM
    table_name
GROUP BY
    column_name1,
    column_name2
HAVING
    aggregate_function (column_name3) > value;
```

[SQL Server HAVING Clause (sqlservertutorial.net)]

T-SQL

SUB QUERIES

Bekerja dengan sub-query

Sub-query adalah query bersarang, atau *query didalam query*.

Hasil dari kueri yang lebih ‘dalam’ (*inner query*) dilanjutkan ke query yang lebih ‘luar’ (diatasnya, *outer query*)
Inner query menjadi seperti expression dari perspektif outer query

Subqueries dapat berupa **self-contained** or **correlated**
Subquery yang Self-contained → Tidak tergantung pada outer query-nya.
Subquery yang Correlated → Bergantung pada nilai dari outer query.

Subqueries bisa scalar, multi-valued, atau table-valued

Menulis sub-query skalar

Subquery Scalar → Mengembalikan 1 nilai ke outer query-nya.
Dapat diletakkan dimana saja sebuah *single-valued expression* bisa diletakkan: SELECT, WHERE, etc.

```
SELECT SalesOrderID, ProductID, UnitPrice, OrderQty
FROM Sales.SalesOrderDetail
WHERE SalesOrderID =
(SELECT MAX(SalesOrderID) AS LastOrder
FROM Sales.SalesOrderHeader);
```

Jika Inner query tidak mengembalikan apa-apa, maka hasil yang dilempar ke outer querynya dianggap sebagai NULL

Penyusunan outer query menentukan apakah inner query harus mengembalikan nilai tunggal atau tidak.

Menulis sub-query multivalue

Subquery multivalue → Mengembalikan lebih dari 1 nilai sebagai 1 set kolom ke outer querynya.

Digunakan pada predicate **IN**

Jika ada nilai yang cocok pada hasil subquery, maka predicate IN-nya akan mengembalikan nilai TRUE

```
SELECT CustomerID, SalesOrderId,TerritoryID
FROM Sales.SalesorderHeader
WHERE CustomerID IN (
SELECT CustomerID
FROM Sales.Customer
WHERE TerritoryID = 10);
```

May also be expressed as a JOIN (test both for performance)

Sub-queries dengan EXISTS terhadap Sub-queries

EXIST → Mengecek nilai ‘yang ada di dalam hasil inner-query’
Keyword EXIST tidak diikuti oleh nama kolom atau ekspresi lain.
SELECT didalam EXISTS umumnya hanya menggunakan SELECT bintang/asterisk(*).

```
SELECT CustomerID, PersonID
FROM Sales.Customer AS Cust
WHERE EXISTS (
SELECT *
FROM Sales.SalesOrderHeader AS Ord
WHERE Cust.CustomerID = Ord.CustomerID);
```

```
SELECT CustomerID, PersonID
FROM Sales.Customer AS Cust
WHERE NOT EXISTS (
SELECT *
FROM Sales.SalesOrderHeader AS Ord
WHERE Cust.CustomerID = Ord.CustomerID);
```

Contoh SUBQUERY

```sql
SELECT
    order_id,
    order_date,
    customer_id
FROM
    sales.orders
WHERE
    customer_id IN (
        SELECT
            customer_id
        FROM
            sales.customers
        WHERE
            city = 'New York'
    )
ORDER BY
    order_date DESC;
```

outer query

subquery

[The Ultimate Guide To SQL Server Subquery (sqlservertutorial.net)]

Thanks!

Do you have any questions?

Team Teaching Matakuliah Basis Data Lanjut
JTI POLINEMA