TABLE EXPRESSION

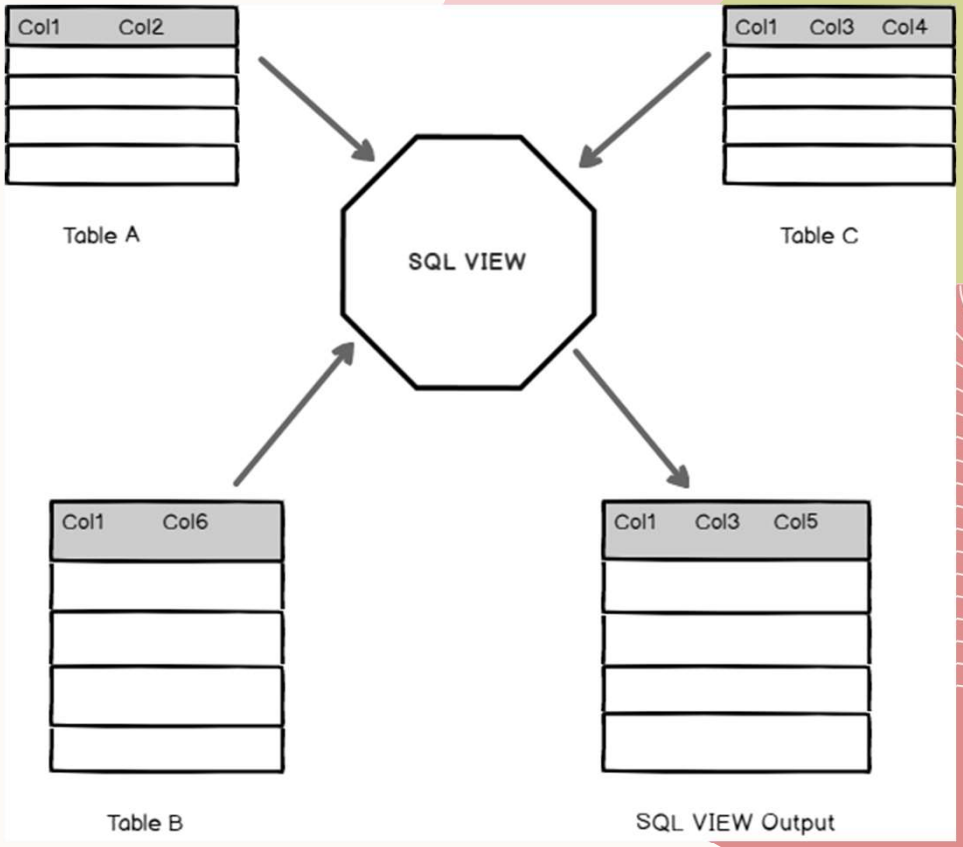
MINGGU KE – 6
BASIS DATA LANJUT

TEAM TEACHING BASIS DATA LANJUT
JURUSAN TEKNOLOGI INFORMASI (TA 2023/2024)

Table Expression

MEMBUAT VIEW SEDERHANA

View adalah QUERY yang disimpan dalam database oleh administrator



MEMBUAT VIEW SEDERHANA

- View adalah QUERY yang disimpan dalam database oleh administrator dan developer
- View didefinisikan dengan satu pernyataan SELECT
- ORDER BY tidak diizinkan dalam definisi View tanpa menggunakan TOP, OFFSET/FETCH, atau FOR XML
- Untuk menyortir output, gunakan ORDER BY di luar QUERY

```
CREATE VIEW HumanResources.EmployeeList
AS
SELECT BusinessEntityID, JobTitle,
HireDate, VacationHours
FROM HumanResources.Employee;

SELECT * FROM HumanResources.EmployeeList
```

MEMBUAT INLINE TABLE-VALUED FUNCTIONS

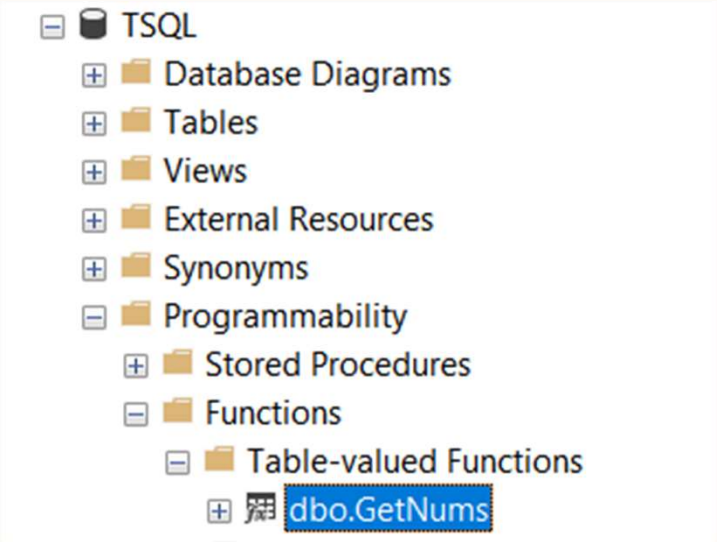


TABLE-VALUED FUNCTIONS dibuat oleh administrator dan developers
TABLE VALUE FUNCTION.

Fungsi tersebut akan **mengembalikan nilai berupa table yang berisi record-record dari hasil eksekusi QUERY yang didefinisikan didalam fungsi** tersebut

MEMBUAT INLINE TABLE-VALUED FUNCTIONS

Membuat dan beri nama fungsi dan parameter opsional dengan CREATE FUNCTION

Nyatakan return type Sebagai TABLE

Nyatakan inline SELECT statement **setelah RETURN**

```
CREATE FUNCTION Sales.fn_LineTotal (@SalesOrderID INT)
RETURNS TABLE
AS
RETURN
   SELECT SalesOrderID,
   CAST((OrderQty * UnitPrice * (1 - SpecialOfferID))
   AS DECIMAL(8, 2)) AS LineTotal
   FROM    Sales.SalesOrderDetail
   WHERE   SalesOrderID = @SalesOrderID ;
```

Perintah yang mendefinisikan TABLE VALUE FUNCTION yaitu perintah RETURNS TABLE. Setelah itu QUERY nya di buat didalam perintah RETURN setelah perintah AS.

MENULIS QUERIES DENGAN DERIVED TABLES

DERIVED TABLES adalah QUERY yang diberi nama, dibuat dalam outer SELECT statement
Tidak disimpan pada database – merepresentasikan virtual relational table
Memungkinkan kita menulis modular QUERIES

```
SELECT <column_list>
FROM  (
   <derived_table_definition>
   ) AS <derived_table_alias>;
```

Lingkup dari DERIVED TABLE adalah pada QUERY dimana DERIVED TABLE tersebut dibuat

GUIDELINES FOR DERIVED TABLES

DERIVED TABLES Must	DERIVED TABLES May
• Have an alias • Have names for all columns • Have unique names for all columns • Not use an ORDER BY clause (without TOP or OFFSET/FETCH) • Not be referred to multiple times in the same QUERY	• Use internal or external aliases for columns • Refer to parameters and/or VARIABLES • Be nested within other DERIVED TABLES

PASSING ARGUMENTS TO DERIVED TABLES

Tabel TURUNAN dapat merujuk pada argumen
Argumen dapat berupa:

VARIABEL yang dideklarasikan dalam batch yang sama seperti pernyataan SELECT

Parameter diteruskan ke fungsi tabel-VALUED atau prosedur tersimpan

```
DECLARE @emp_id INT = 9;
SELECT orderyear, COUNT(DISTINCT custid) AS
cust_count
FROM (
    SELECT YEAR(orderdate) AS orderyear,
custid
    FROM Sales.Orders
    WHERE empid=@emp_id
) AS derived_year
GROUP BY orderyear;
```

CTE

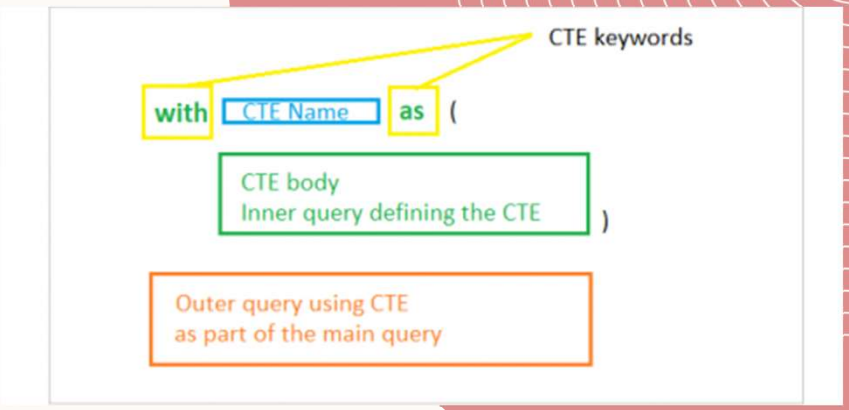
```
WITH avg_price_brand AS
    (SELECT brand, AVG(price) AS average_for_brand
     FROM cameras
     GROUP BY brand)
SELECT c.id, c.brand, c.model, c.price, avg.average_for_brand
FROM cameras c
JOIN avg_price_brand avg
ON c.brand = avg.brand;
```

WITH keyword

CTE Name

CTE body

Use of CTE

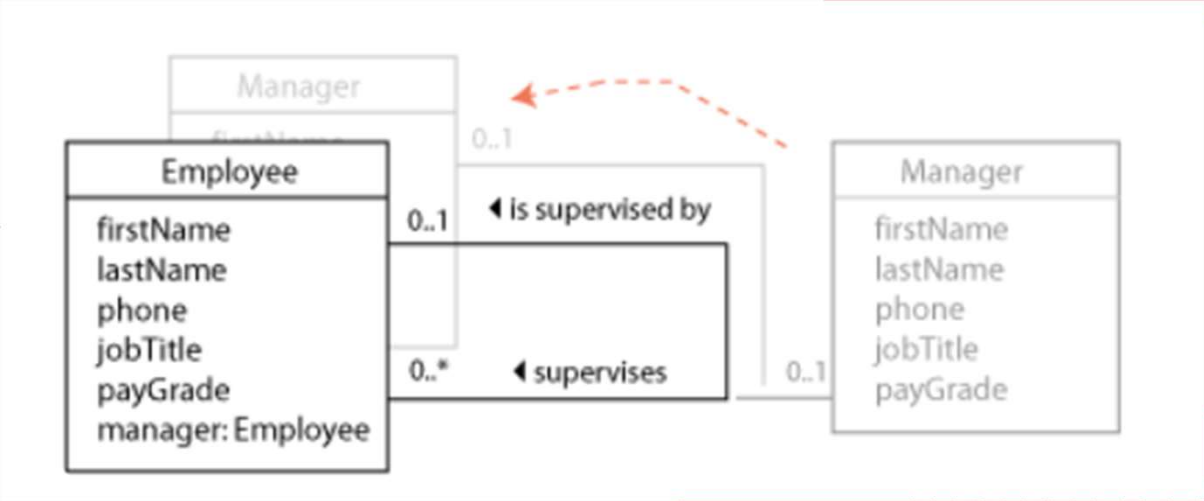


MEMBUAT QUERIES DENGAN CTE

Common Table Expression (CTE) adalah salah satu bentuk query SQL yang digunakan **untuk menyederhanakan JOIN pada SQL kedalam subqueries dan mampu memberikan query yang bersifat hiearki.**

Hirarki dengan CTE

contohnya adalah design pada masalah employee-manager



MEMBUAT QUERIES DENGANCOMMON TABLE EXPRESSIONS

Use WITH clause to create a CTE:

- Definisikan CTE dengan WITH
- Panggil CTE di outer QUERY
- berikan alias (inline or external)
- Berikan arguments jika diinginkan

```
WITH CTE_year AS
(
SELECT YEAR(OrderDate) AS OrderYear, customerID
FROM Sales.SalesOrderHeader
)
SELECT orderyear, COUNT(DISTINCT CustomerID) AS CustCount
FROM CTE_year
GROUP BY OrderYear;
```

TERIMA KASIH