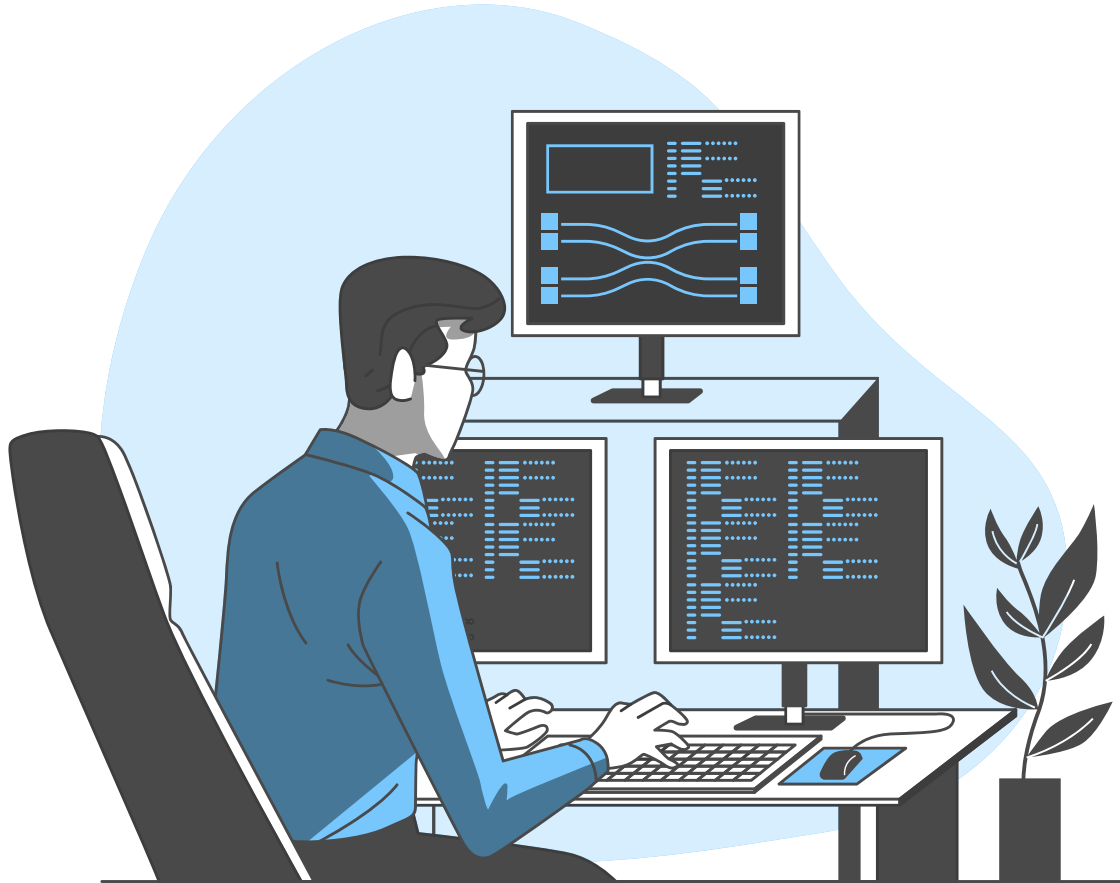
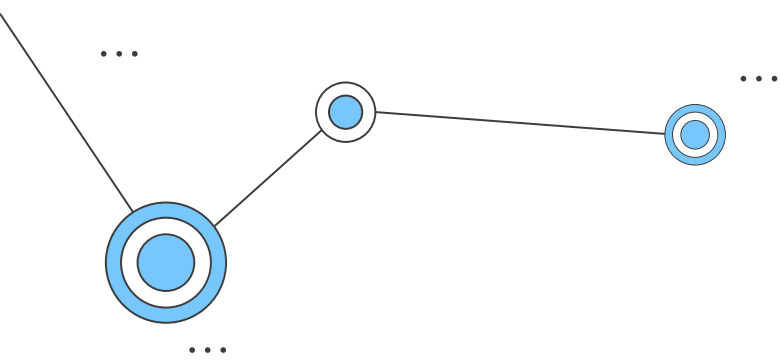


# BASIS DATA LANJUT

**Pertemuan 3**

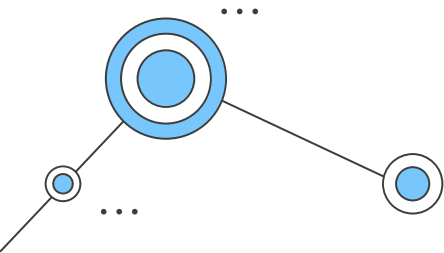
TIPE DATA FUNGSI-FUNGSI BAWAAN

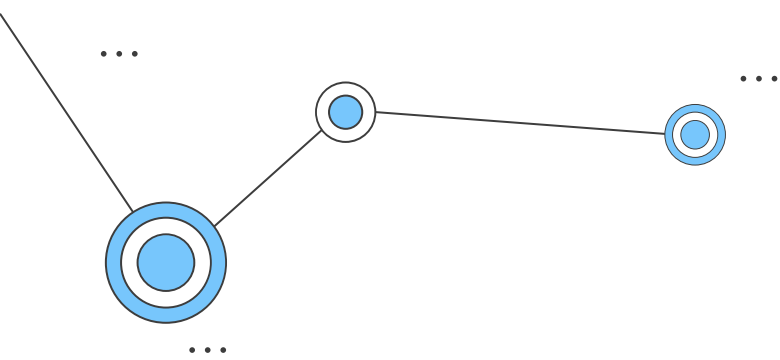




# OUTLINE

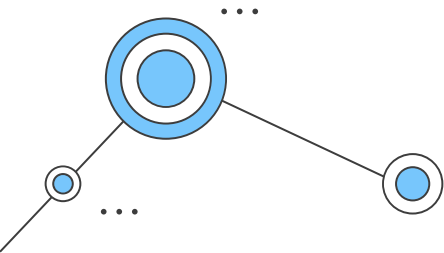
- Tipe Data
- Date & Time
- Fungsi Date & Time
- Tipe Data Karakter
- Fungsi Karakter

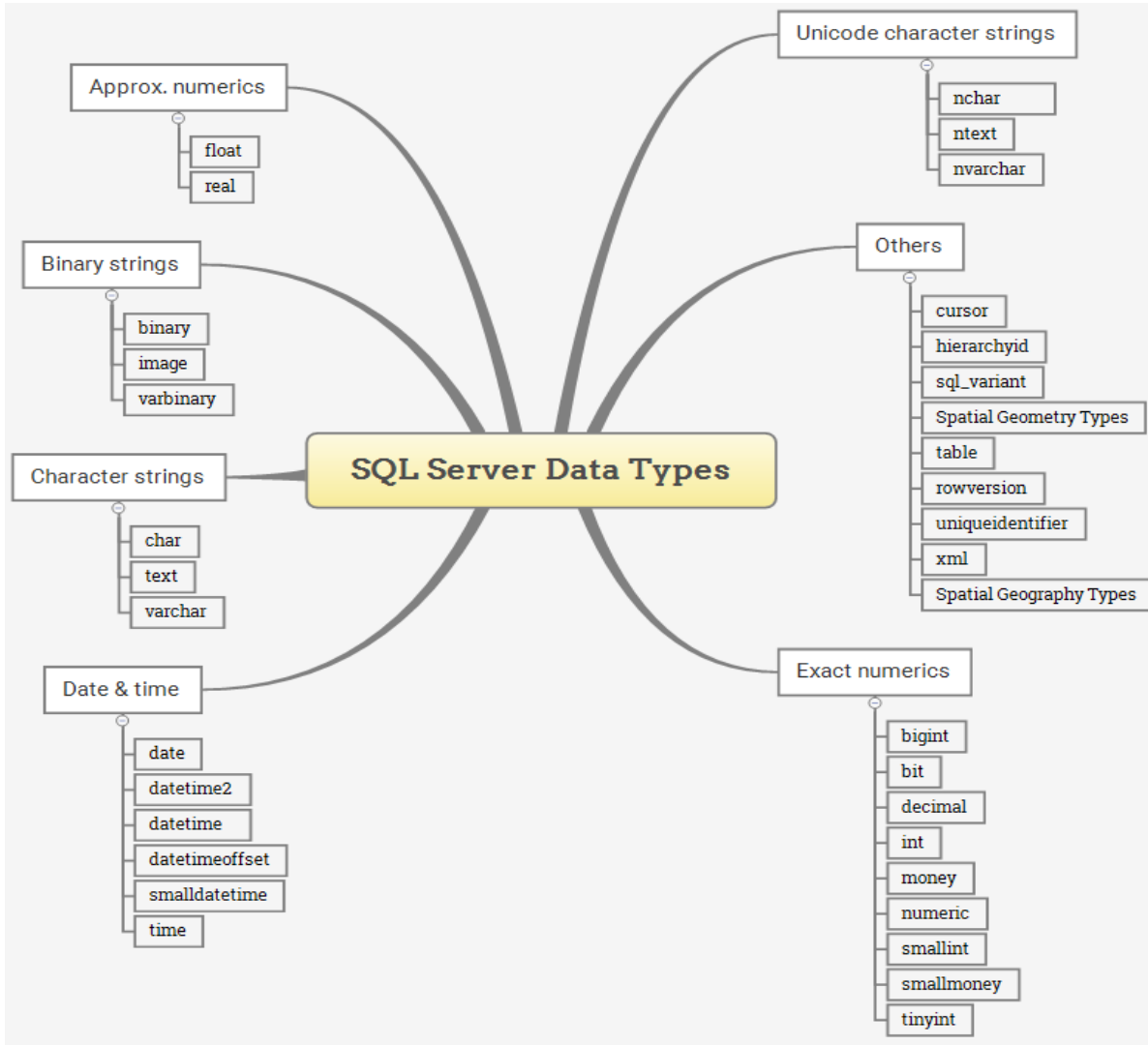




# Tipe Data

- Mendefinisikan suatu kolom
- Menentukan batasan/kontrol terhadap data
- Menentukan memori yang digunakan







# TIPE DATA YANG TEPAT

- Menjaga database lebih reliable → memastikan data yang masuk sesuai standar
- Efisiensi → menghemat storage
- Pertimbangkan perubahan yang akan datang
- Contoh:
  - Untuk menyimpan tanggal, gunakan tipe data datetime, sehingga dapat dilakukan operasi waktu, misalnya menghitung rentang waktu

# Tipe Data NUMERIC



# TIPE DATA NUMERIK

- Numerik adalah tipe data berupa **angka** yang mengizinkan operasi aritmatika.
- Data numerik ini dapat berupa bilangan positif dan negatif maupun bilangan bulat dan pecahan.
- Untuk menyimpan data dalam bentuk bilangan bulat negatif ataupun positif dapat menggunakan **INT**.

Data type	Range	Storage
<b>BIGINT</b>	$-2^{63}$ (-9,223,372,036,854,775,808) to $2^{63}-1$ (9,223,372,036,854,775,807)	8 Bytes
<b>INT</b>	$-2^{31}$ (-2,147,483,648) to $2^{31}-1$ (2,147,483,647)	4 Bytes
<b>SMALLINT</b>	$-2^{15}$ (-32,768) to $2^{15}-1$ (32,767)	2 Bytes
<b>TINYINT</b>	0 to 255	1 Byte



# DECIMAL & NUMERIC

- Decimal dan numeric adalah tipe data yang memiliki fixed precision dan scale
- Numeric adalah sinonim dari decimal

```
CREATE TABLE dbo.MyTable (  
    MyDecimalColumn DECIMAL(5, 2),  
    MyNumericColumn NUMERIC(10, 5)  
);  
GO
```

```
INSERT INTO dbo.MyTable  
VALUES (123, 12345.12);  
GO
```

```
SELECT MyDecimalColumn, MyNumericColumn  
FROM dbo.MyTable;
```

	MyDecimalColumn	MyNumericColumn
1	123.00	12345.12000



# Tipe Data DATE & TIME

# TYPE DATA DATE & TIME

Data type	Format	Range	Accuracy	Storage size (bytes)	User-defined fractional second precision	Time zone offset
time	hh:mm:ss[.nnnnnnn]	00:00:00.0000000 through 23:59:59.9999999	100 nanoseconds	3 to 5	Yes	No
date	YYYY-MM-DD	0001-01-01 through 9999-12-31	1 day	3	No	No
smalldatetime	YYYY-MM-DD hh:mm:ss	1900-01-01 through 2079-06-06	1 minute	4	No	No
datetime	YYYY-MM-DD hh:mm:ss[.nnn]	1753-01-01 through 9999-12-31	0.00333 second	8	No	No
datetime2	YYYY-MM-DD hh:mm:ss[.nnnnnnn]	0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999	100 nanoseconds	6 to 8	Yes	No
datetimeoffset	YYYY-MM-DD hh:mm:ss[.nnnnnnn] [+ -]hh:mm	0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999 (in UTC)	100 nanoseconds	8 to 10	Yes	Yes



# FUNGSI DATE & TIME

<b>Fungsi</b>	<b>Deskripsi</b>
GETDATE()	Mengembalikan tanggal dan waktu
DATEPART()	Mengembalikan satu bagian dari tanggal / waktu
DATEADD()	Menambahkan atau mengurangi interval waktu yang ditentukan dari tanggal
DATEDIFF()	Mengembalikan waktu antara dua tanggal
CONVERT()	tanggal Menampilkan / data time dalam format yang berbeda

# CONTOH QUERY DENGAN DATE

```
SELECT OrderID, CustomerID, EmployeeID, OrderDate
FROM Sales.Orders
WHERE OrderDate = '20070825';
--WHERE OrderDate = '2007-08-25';
--WHERE OrderDate = '2007/08/25';
```

	OrderID	CustomerID	EmployeeID	OrderDate
1	10643	1	6	2007-08-25 00:00:00.000
2	10644	88	3	2007-08-25 00:00:00.000



# CONTOH QUERY DENGAN DATE

```
SELECT LastName, BirthDate
FROM HR.Employees
WHERE YEAR(BirthDate) = 1973;
```

	LastName	BirthDate
1	Lew	1973-08-30 00:00:00.000
2	Suurs	1973-07-02 00:00:00.000

# CURRENT\_TIMESTAMP

```
SELECT CURRENT_TIMESTAMP
```

Results		Messages
	(No column name)	
1	2022-09-10 05:46:43.380	
✓ Query executed successfully.   DESKTOP-EIPTP8V (15.0 RTM)		

- Mengembalikan nilai berupa tanggal dan waktu saat ini
- Merupakan fungsi SQL ANSI



# DATEADD()

SELECT

    Lastname

    ,HireDate

    ,DATEADD(YEAR, 1, HireDate) AS AssesmentDate

FROM HR.Employees

	Lastname	HireDate	AssesmentDate
1	Davis	2002-05-01 00:00:00.000	2003-05-01 00:00:00.000
2	Funk	2002-08-14 00:00:00.000	2003-08-14 00:00:00.000
3	Lew	2002-04-01 00:00:00.000	2003-04-01 00:00:00.000
4	Peled	2003-05-03 00:00:00.000	2004-05-03 00:00:00.000
5	Buck	2003-10-17 00:00:00.000	2004-10-17 00:00:00.000
6	Suurs	2003-10-17 00:00:00.000	2004-10-17 00:00:00.000
7	King	2004-01-02 00:00:00.000	2005-01-02 00:00:00.000
8	Cameron	2004-03-05 00:00:00.000	2005-03-05 00:00:00.000
9	Dolgopyatova	2004-11-15 00:00:00.000	2005-11-15 00:00:00.000



# DATEDIFF()

SELECT

    LastName

    ,BirthDate

    ,DATEDIFF(YEAR, BirthDate, GETDATE()) AS Age

FROM HR.Employees

	LastName	BirthDate	Age
1	Davis	1958-12-08 00:00:00.000	66
2	Funk	1962-02-19 00:00:00.000	62
3	Lew	1973-08-30 00:00:00.000	51
4	Peled	1947-09-19 00:00:00.000	77
5	Buck	1965-03-04 00:00:00.000	59
6	Suurs	1973-07-02 00:00:00.000	51
7	King	1970-05-29 00:00:00.000	54
8	Cameron	1968-01-09 00:00:00.000	56
9	Dolgopyatova	1976-01-27 00:00:00.000	48



# CAST & CONVERT



# CAST() & CONVERT()

- CAST() dan CONVERT() adalah fungsi digunakan untuk konversi ekspresi dari satu jenis tipe data ke tipe data lainnya secara eksplisit
- CAST() → SQL Standard
- CONVERT() → T-SQL
- CAST disarankan untuk konversi dasar, CONVERT disarankan spesifik untuk datetime



# CAST

- Syntax:

**CAST(<value> AS <datatype>)**

**SELECT CAST(25.65 AS int);**



# CONVERT

- Untuk konversi:
  - Date
  - Time
  - Numeric
  - XML
- Syntax:

**CONVERT(<datatype>,<value>,<optional\_style\_number>)**

```
SELECT CONVERT(int, 25.65);
```

# KONVERSI FORMAT DATE

Without century	With century	Input/Output	Standard
0	100	mon dd yyyy hh:miAM/PM	Default
1	101	mm/dd/yyyy	US
2	102	yyyy.mm.dd	ANSI
3	103	dd/mm/yyyy	British/French
4	104	dd.mm.yyyy	German
5	105	dd-mm-yyyy	Italian
6	106	dd mon yyyy	-
7	107	Mon dd, yyyy	-
8	108	hh:mm:ss	-
9	109	mon dd yyyy hh:mi:ss:mmmAM (or PM)	Default + millisec

[https://www.w3schools.com/sql/func\\_sqlserver\\_convert.asp](https://www.w3schools.com/sql/func_sqlserver_convert.asp)



# CAST & CONVERT

```
SELECT CAST('20220910' AS DATE)
```

```
SELECT CONVERT(DATETIME, '09/22/2022')
```

Results		Messages
	(No column name)	
1	2022-09-10	
	(No column name)	
1	2022-09-22 00:00:00.000	
✓ Query executed successfully.		DESKTOP-EIPTP8V (15.0 RTM)



# CAST & CONVERT

```
SELECT LastName
       ,BirthDate
       ,CONVERT(varchar, BirthDate, 101) AS BirthDate1
       ,CONVERT(varchar, BirthDate, 106) AS BirthDate2
       ,CONVERT(varchar, BirthDate, 105) AS BirthDate3
FROM [HR].[Employees]
```

	LastName	BirthDate	BirthDate1	BirthDate2	BirthDate3
1	Davis	1958-12-08 00:00:00.000	12/08/1958	08 Dec 1958	08-12-1958
2	Funk	1962-02-19 00:00:00.000	02/19/1962	19 Feb 1962	19-02-1962
3	Lew	1973-08-30 00:00:00.000	08/30/1973	30 Aug 1973	30-08-1973
4	Peled	1947-09-19 00:00:00.000	09/19/1947	19 Sep 1947	19-09-1947
5	Buck	1965-03-04 00:00:00.000	03/04/1965	04 Mar 1965	04-03-1965
6	Suurs	1973-07-02 00:00:00.000	07/02/1973	02 Jul 1973	02-07-1973
7	King	1970-05-29 00:00:00.000	05/29/1970	29 May 1970	29-05-1970
8	Cameron	1968-01-09 00:00:00.000	01/09/1968	09 Jan 1968	09-01-1968
9	Dolgopyatova	1976-01-27 00:00:00.000	01/27/1976	27 Jan 1976	27-01-1976

# TIPE DATA KARAKTER





# TIPE DATA KARAKTER

Data type	Description	Max size	Storage
char(n)	Fixed width character string	8,000 characters	Defined width
varchar(n)	Variable width character string	8,000 characters	2 bytes + number of chars
varchar(max)	Variable width character string	1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string	2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string	4,000 characters	Defined width x 2
nvarchar	Variable width Unicode string	4,000 characters	
nvarchar(max)	Variable width Unicode string	536,870,912 characters	
ntext	Variable width Unicode string	2GB of text data	
binary(n)	Fixed width binary string	8,000 bytes	
varbinary	Variable width binary string	8,000 bytes	
varbinary(max)	Variable width binary string	2GB	
image	Variable width binary string	2GB	



# CONCATENATION

Menggabungkan karakter

- Menggunakan operator '+'

```
SELECT EmployeeID, FirstName + ' ' + LastName AS FullName  
FROM HR.Employees;
```

- Menggunakan CONCAT()

```
SELECT EmployeeID, CONCAT (FirstName, ' ', LastName) AS FullName  
FROM HR.employees;
```

# CONCATENATION

```
SELECT City
       ,Region
       ,City + ' ' + Region as Contoh1
       ,CONCAT(City, ',', Region) as Contoh2
FROM [Sales].[Customers]
```

	City	Region	Contoh1	Contoh2
1	Berlin	NULL	NULL	Berlin,
2	México D.F.	NULL	NULL	México D.F.,
3	México D.F.	NULL	NULL	México D.F.,
4	London	NULL	NULL	London,
5	Luleå	NULL	NULL	Luleå,
6	Mannheim	NULL	NULL	Mannheim,
7	Strasbourg	NULL	NULL	Strasbourg,
8	Madrid	NULL	NULL	Madrid,
9	Marseille	NULL	NULL	Marseille,
10	Tsawassen	BC	Tsawassen BC	Tsawassen, BC

# SUBSTRING()

Mengembalikan substring berdasarkan *starting point* dan jumlah karakter yang dikembalikan

```
SUBSTRING(expression, start, length)
```

Contoh:

```
SELECT
```

```
    SUBSTRING ('Microsoft SQL SERVER', 11, 3) AS Contoh1  
    , SUBSTRING ('Microsoft SQL SERVER', 11, 100) AS Contoh2  
    , SUBSTRING ('Microsoft SQL SERVER', 110, 100) AS Contoh3
```

	Contoh1	Contoh2	Contoh3
1	SQL	SQL SERVER	

# LEFT() & RIGHT()

Mengembalikan substring dalam jumlah tertentu, dimulai dari karakter string paling kiri/kanan

LEFT(character\_expression , integer\_expression)

RIGHT(character\_expression , integer\_expression)

Contoh:

SELECT

LEFT('Microsoft SQL SERVER', 9) AS Contoh1

, RIGHT('ID-12345', 5) AS Contoh2;

	Contoh1	Contoh2
1	Microsoft	12345

# LEN() & DATALENGTH()

- LEN(String) → Mengembalikan jumlah karakter (exclude trailing blanks)
- DATALENGTH(String) → Mengembalikan jumlah byte data

Contoh:

```
SELECT Name, LEN(Name) as Length1, DATALENGTH(Name) as Length2  
FROM Production.Product
```


	Name	Length1	Length2
1	Adjustable Race	15	30
2	All-Purpose Bike Stand	22	44
3	AWC Logo Cap	12	24
4	BB Ball Bearing	15	30
5	Bearing Ball	12	24



# LIKE

- Digunakan untuk membandingkan karakter berdasarkan pola tertentu
- Digunakan pada statement WHERE
- Simbol % merepresentasikan string

```
SELECT CategoryID, Description  
FROM Production.Categories  
WHERE Description LIKE '%ee%';
```



	CategoryID	Description
1	1	Soft drinks, coffees, teas, beers, and ales
2	2	Sweet and savory sauces, relishes, spreads, and ...
3	3	Desserts, candies, and sweet breads
4	4	Cheeses
5	8	Seaweed and fish



# NULL

- NULL merepresentasikan nilai yang kosong
- Hasil semua ekspresi yang memuat nilai NULL adalah NULL
  - $2 + \text{NULL} = \text{NULL}$
  - `'MyString:' + NULL = NULL`
- Perbandingan nilai
  - NULL = NULL returns *false*
  - NULL **IS NULL** returns *true*





# ISNULL()

ISNULL() menggantikan nilai NULL dengan nilai lain

**ISNULL(<check\_expression>,<replacement\_value>);**

Argumen	Deskripsi
check_expression	Ekspresi yang perlu dievaluasi
replacement_value	Nilai kembalian jika ekspresi bernilai NULL

# ISNULL()

```
SELECT CustomerID, City, Region, Country
FROM [Sales].[Customers]
```

	CustomerID	City	Region	Country
1	1	Berlin	NULL	Germany
2	2	México D.F.	NULL	Mexico
3	3	México D.F.	NULL	Mexico
4	4	London	NULL	UK
5	5	Luleå	NULL	Sweden
6	6	Mannheim	NULL	Germany
7	7	Strasbourg	NULL	France
8	8	Madrid	NULL	Spain
9	9	Marseille	NULL	France
10	10	Tsawassen	BC	Canada

```
SELECT CustomerID
      ,City
      ,ISNULL(Region, 'N/A') as Region
      ,Country
FROM [Sales].[Customers]
```

	CustomerID	City	Region	Country
1	1	Berlin	N/A	Germany
2	2	México D.F.	N/A	Mexico
3	3	México D.F.	N/A	Mexico
4	4	London	N/A	UK
5	5	Luleå	N/A	Sweden
6	6	Mannheim	N/A	Germany
7	7	Strasbourg	N/A	France
8	8	Madrid	N/A	Spain
9	9	Marseille	N/A	France
10	10	Tsawassen	BC	Canada

# LOGICAL FUNCTION



# IIF()

- Seleksi kondisi dengan IIF
- Seperti ekspresi CASE dengan 2 kemungkinan kembalian
- Syntax:

```
SELECT IIF(<boolean expression>,<value_if_TRUE>,  
          <value_if_FALSE_or_UNKNOWN>);
```



# IIF()

```
SELECT ProductID
       ,UnitPrice
       ,IIF(UnitPrice > 50, 'High', 'Low') AS PricePoint
FROM Production.Products;
```

	ProductID	UnitPrice	PricePoint
1	1	18.00	Low
2	2	19.00	Low
3	3	10.00	Low
4	4	22.00	Low
5	5	21.35	Low
6	6	25.00	Low
7	7	30.00	Low
8	8	40.00	Low
9	9	97.00	High
10	10	31.00	Low

# IIF()

```
SELECT ProductID
       ,UnitPrice
       ,IIF(UnitPrice > 50, 'High', IIF(UnitPrice > 20, 'Middle','Low')) AS PricePoint
FROM Production.Products;
```

	ProductID	UnitPrice	PricePoint
1	1	18.00	Low
2	2	19.00	Low
3	3	10.00	Low
4	4	22.00	Middle
5	5	21.35	Middle
6	6	25.00	Middle
7	7	30.00	Middle
8	8	40.00	Middle
9	9	97.00	High
10	10	31.00	Middle



# CHOOSE()

- CHOOSE mengembalikan sebuah item pada list yang dipilih berdasarkan nilai index tertentu
- 1-based index
- Syntax:

```
SELECT CHOOSE(<index_value>,<item1>, <item2>[,...]);
```

- Contoh:

SELECT

```
    CHOOSE(StatusCode, 'Pending', 'Approved', 'Rejected') AS Status  
FROM Orders;
```

```
SELECT CHOOSE(3, 'Monday', 'Tuesday', 'Wednesday', 'Thursday',  
'Friday') AS DayOfWeek;
```

# Thanks!

Do you have any questions?



Team Teaching Matakuliah Basis Data Lanjut  
JTI POLINEMA