WEEK 2 – HOMEWORK 3

INTERMEDIATE SQL

By

Anugrah Yazid Ghani

DATA SCIENCE - BATCH 11

1. *Window Function* dan contohnya

Window functions menyediakan kemampuan untuk melakukan penghitungan di seluruh rangkaian baris yang terkait dengan baris *query* saat ini. *General-Purpose Window Functions* tercantum dalam tabel di bawah ini.

Tabel *General-Purpose Window Functions*

Function	Return Type	Description
row_number()	bigint	number of the current row within its partition, counting from 1
rank()	bigint	rank of the current row with gaps; same as row_number of its first peer
dense_rank()	bigint	rank of the current row without gaps; this function counts peer groups
percent_rank()	double precision	relative rank of the current row: (rank - 1) / (total rows - 1)
cume_dist()	double precision	relative rank of the current row: (number of rows preceding or peer with current row) / (total rows)
ntile(num_buckets integer)	integer	integer ranging from 1 to the argument value, dividing the partition as equally as possible
lag(value any [, offset integer [, default any]])	same type as value	returns value evaluated at the row that is offset rows before the current row within the partition; if there is no such row, instead return default. Both offset and default are evaluated with respect to the current row. If omitted, offset defaults to 1 and default to null
lead(value any [, offset integer [, default any]])	same type as value	returns value evaluated at the row that is offset rows after the current row within the partition; if there is no such row, instead return default. Both offset and default are evaluated with respect to the current row. If omitted, offset defaults to 1 and default to null
first_value(value any)	same type as value	returns value evaluated at the row that is the first row of the window frame
last_value(value any)	same type as value	returns value evaluated at the row that is the last row of the window frame
nth_value(value any, nth integer)	same type as value	returns value evaluated at the row that is the nth row of the window frame (counting from 1); null if no such row

Perhatikan bahwa fungsi-fungsi ini harus dipanggil menggunakan sintaks *window functions*; itu adalah klausa OVER diperlukan.

Selain fungsi-fungsi ini, semua fungsi agregat bawaan atau yang ditentukan pengguna dapat digunakan sebagai *window functions*. Namun tidak seperti fungsi agregat biasa, penggunaan fungsi jendela tidak menyebabkan baris dikelompokkan menjadi satu baris keluaran, akan tetapi baris mempertahankan identitasnya yang terpisah. Fungsi agregat bertindak sebagai *window functions* hanya ketika klausa OVER mengikuti panggilan; jika tidak mereka bertindak sebagai agregat biasa.

Contoh:

Window functions memungkinkan akses ke data dalam catatan tepat sebelum dan sesudah catatan saat ini. *Window functions* mendefinisikan bingkai atau baris *window* dengan panjang tertentu di sekitar baris saat ini, dan melakukan perhitungan di seluruh kumpulan data di *window*.

```
Current row

      NAME |
------------
      Akbar| <-- Preceding (unbounded) <-- 1st preceding row
    Anugrah|
     Amelia|
     Farhan|
      Fikri|
     Firman|
        Nia|
       Nita|
      Paula|
       Zoro| <-- Following (unbounded) <-- 1st following row
```

Dalam tabel di atas, *query* berikutnya mengekstrak untuk setiap baris nilai *window* dengan satu baris sebelumnya (*preceding row*) dan satu baris berikutnya (*following row*):

```
SELECT
  LAG(name, 1)
    OVER(ORDER BY name) "prev",
  name,
  LEAD(name, 1)
    OVER(ORDER BY name) "next"
 FROM people
 ORDER BY name
```

Output:

```
|      PREV |      NAME |      NEXT |
|-----------|-----------|-----------|
|    (null)|      Akbar|    Anugrah|
|      Akbar|    Anugrah|     Amelia|
|    Anugrah|     Amelia|     Farhan|
|     Amelia|     Farhan|      Fikri|
|     Farhan|      Fikri|     Firman|
|      Fikri|     Firman|        Nia|
|     Firman|        Nia|       Nita|
|        Nia|       Nita|      Paula|
|       Nita|      Paula|       Zoro|
|      Paula|       Zoro|    (null)|
```

2. Penggunaan WHERE didalam fungsi SELECT

Pernyataan/klausa WHERE sering kita temui pada query-query yang kita lakukan. baik untuk mengupdate atau menghapus data. Perintah tersebut wajib menggunakan klausa WHERE dimana query akan dijalankan jika memenuhi kondisi tertentu yang telah ditetapkan. WHERE dalam SELECT digunakan untuk memfilter hasil SELECT dengan mengekstrak record yang memenuhi persyaratan tertentu.

Berikut adalah sintak sederhana dari pernyataan WHERE dalam SELECT:

```
SELECT nama_kolom
FROM nama_tabel
WHERE kondisi_tertentu
```

Penjelasan:

- Pernyataan SELECT akan menampilkan record dari kolom yang ingin ditapilkan. Jika kita ingin menampilkan semua kolom bisa menggunakan * .
- Pernyataan FROM mengidentifikasi sumber data dari suatu tabel.
- Pernyataan WHERE akan memfilter hasil baris data. Hanya record/data yang memenuhi saja yang akan ditampilkan/eksekusi.

Operator pada klausa WHERE di SQL

Operator digunakan untuk menfilter data disesuaikan dengan operator yang digunakan. Bisa berupa operator pembanding, logika, dan operator lainnya. Berikut ini adalah operator-operator yang dapat digunakan dengan pernyataan WHERE:

OPERATOR	KETERANGAN
=	Sama dengan
>	Lebih besar dari
<	Kurang dari
>=	Lebih besar dari atau sama dengan
<=	Kurang dari atau sama dengan
<> atau !=	Tidak sama dengan *(Operator tergantung pada versi SQL yang digunakan)
BETWEEN	Berada dalam rentang tertentu
LIKE	kata kunci kriteria tertentu
IN	Menentukan beberapa nilai yang berada dalam sebuah kolom
NOT IN	Menentukan beberapa nilai yang tidak berada dalam sebuah kolom

Contoh: Data simpel Mahasiswa

Tabel mahasiswa

id	nama	universitas	jurusan	umur
1	Anugrah	ITB	KL	24
2	Dita	UPI	TF	25
3	Annisa	ITB	TI	22
4	Bayu	ITB	KL	19
5	Rini	UNPAD	EK	24
6	Tita	UNPAD	TF	23
7	Dewi	UPI	EK	22
8	Putri	UNPAD	KL	20
9	Ihsan	ITB	TI	26
10	Ridwan	UPI	TI	21

- **Operator sama dengan (=)**

```
SELECT * FROM mahasiswa
WHERE universitas = 'ITB'
```

Output:

id	nama	universitas	jurusan	umur
1	Anugrah	ITB	KL	24
3	Annisa	ITB	TI	22
4	Bayu	ITB	KL	19
9	Ihsan	ITB	TI	26

- **Operator lebih besar dari (>)**

```
SELECT * FROM mahasiswa
WHERE umur > 24
```

Output:

id	nama	universitas	jurusan	umur
2	Dita	UPI	TF	25
9	Ihsan	ITB	TI	26

- **Operator kurang dari (<)**

```
SELECT * FROM mahasiswa
WHERE umur < 24
```

Output:

id	nama	universitas	jurusan	umur
3	Annisa	ITB	TI	22
4	Bayu	ITB	KL	19
6	Tita	UNPAD	TF	23
7	Dewi	UPI	EK	22
8	Putri	UNPAD	KL	20
10	Ridwan	UPI	TI	21

- **Operator lebih besar dari atau sama dengan (>=)**

```
SELECT * FROM mahasiswa
WHERE umur >= 24
```

Output:

id	nama	universitas	jurusan	umur
1	Anugrah	ITB	KL	24
2	Dita	UPI	TF	25
5	Rini	UNPAD	EK	24
9	Ihsan	ITB	TI	26

- **Operator kurang dari atau sama dengan (<=)**

```
SELECT * FROM mahasiswa
WHERE umur <= 24
```

Output:

id	nama	universitas	jurusan	umur
1	Anugrah	ITB	KL	24
3	Annisa	ITB	TI	22
4	Bayu	ITB	KL	19
5	Rini	UNPAD	EK	24
6	Tita	UNPAD	TF	23
7	Dewi	UPI	EK	22

8	Putri	UNPAD	KL	20
10	Ridwan	UPI	TI	21

- **Operator tidak sama dengan (<> atau !=)**

```
SELECT * FROM mahasiswa
WHERE jurusan <> 'KL'

OR

SELECT * FROM mahasiswa
WHERE jurusan != 'KL'
```

Output:

id	nama	universitas	jurusan	umur
2	Dita	UPI	TF	25
3	Annisa	ITB	TI	22
5	Rini	UNPAD	EK	24
6	Tita	UNPAD	TF	23
7	Dewi	UPI	EK	22
9	Ihsan	ITB	TI	26
10	Ridwan	UPI	TI	21

- **Operator BETWEEN untuk mencari rentang tertentu**

```
SELECT * FROM mahasiswa
WHERE umur between 24 and 26
```

Output:

id	nama	universitas	jurusan	umur
1	Anugrah	ITB	KL	24
2	Dita	UPI	TF	25
5	Rini	UNPAD	EK	24
9	Ihsan	ITB	TI	26

- **Operator LIKE untuk mencari kata kunci tertentu**

```
SELECT * FROM mahasiswa
WHERE nama like '_nug%'
```

Output:

id	nama	universitas	jurusan	umur
1	Anugrah	ITB	KL	24

- **Operator IN**

```
SELECT * FROM mahasiswa
WHERE jurusan in (‘KL’, ‘TI’)
```

Output:

id	nama	universitas	jurusan	umur
1	Anugrah	ITB	KL	24
3	Annisa	ITB	TI	22
4	Bayu	ITB	KL	19
8	Putri	UNPAD	KL	20
9	Ihsan	ITB	TI	26
10	Ridwan	UPI	TI	21

- **Operator NOT IN**

```
SELECT * FROM mahasiswa
WHERE jurusan not in (‘KL’, ‘TI’)
```

Output:

id	nama	universitas	jurusan	umur
2	Dita	UPI	TF	25
5	Rini	UNPAD	EK	24
6	Tita	UNPAD	TF	23
7	Dewi	UPI	EK	22

Perbedaan *Teks Field* dan *Numerik Field*

Dapat dilihat pada contoh-contoh penggunaan operator pada WHERE diatas, ketika menggunakan pernyataan WHERE, field atau kolom yang berupa teks harus wajib menggunakan tanda petik tunggal ”. Sementara untuk field yang menggunakan tipe data numerik seperti integer tidak boleh menggunakan petik tunggal seperti pada field teks tersebut.

Wildcard Characters in* SQL *Server

Dapat dilihat dari contoh LIKE diatas terdapat simbol _ dan % untuk mencari kata kunci tertentu, berikut tabel *Wildcard Characters* pada Server SQL:

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents any single character within the specified range	c[a-b]t finds cat and cbt

Semua *Wildcard Characters* juga dapat dikombinasikan !