


# Percabangan & Perulangan Pada MySQL

BY SATRIA ARDI PERDANA

# OVERVIEW

- Deklarasi Variabel Pada MySQL
  - Statement Percabangan
  - Perulangan
  - Tanya Jawab
  - Challenge / Quiz
- 

# Deklarasi Variable Di MySQL

- Umumnya variable merupakan containers (wadah) yang menyimpan informasi pada suatu program.
- Nilai atau value dari suatu variable dapat diubah-ubah sesuai yang diinginkan atau dibutuhkan.
- Setiap variable memiliki tipe data untuk menyimpan nilai sesuai tipe data dari variabel tersebut.
- Tipe data pada variable misalnya String, integer dan lain sebagainya.

# Tipe Variable Di MySQL

1. User-Defined Variable
2. Local Variable
3. MySQL local variable

# Tipe Variable Di MySQL

## 1. User-Defined Variable

Variabel yang dibuat User yang memungkinkan user untuk menyimpan nilai/value dalam suatu statement dan kemudian merefer ke statement yang lain.

Syntax :

```
SELECT @variable_name = value;
```

# Tipe Variable Di MySQL

## 2. Local Variable

- MySQL local variable dapat dideklarasikan menggunakan perintah **DECLARE**.
- Ketika mendeklarasikan Local variable symbol @ tidak digunakan sebagai prefix.
- Local Variable merupakan variable tipe kuat, yang berarti tipe datanya harus di declare secara pasti. Nilai variable pada saat didefinisikan bersifat optional, yang berarti jika tidak didefinisikan nilainya = null.

**NOTE: NULL beda dengan 0.**

Syntax local variable:

```
DECLARE variable_name1, variable_name2, ...  
data_type [DEFAULT default_value];
```

# Tipe Variable Di MySQL

## 3. System Variable

- System variable sudah ditentukan oleh MySQL. Variable tersebut berisi data yang diperlukan ketika kita bekerja dengan database. Setiap variabelnya memiliki nilai default.
- Untuk merubah nilai defaultnya bisa digunakan perintah SET
- Pada system variabel ada 2 scope variable command untuk menampilkan variable:
  - a. GLOBAL → Variabel GLOBAL aktif sepanjang siklus hidup.
  - b. SESSION → Variabel SESSION hanya tersedia di sesi saat ini.
- Command display semua sistem variable di MySQL:  
**SHOW [GLOBAL | SESSION] VARIABLES;**



# Contoh User-Defined Variable:

Contoh mengisi value dengan perintah **SET**:

```
SET @Name = 'Satria'; -- set value
```

Dengan **SELECT** statement, kita dapat menampilkan value dari @name variable dengan query berikut:

```
SELECT @Name; -- get value
```

Bisa Juga dengan memasukan nilai variable secara langsung menggunakan perintah berikut:

```
SELECT @Name := 'Satria';
```



# Contoh Local Variable:

Contoh local variable dalam store procedure gaji

```
CREATE PROCEDURE gaji()  
BEGIN  
  DECLARE Robert INT;  
  DECLARE Mila INT DEFAULT 3000000;  
  DECLARE Toni INT;  
  DECLARE Total INT;  
  SET Robert = 2000000;  
  SET Toni = 2900000;  
  SET Total = Robert+Mila+Toni;  
  SELECT Total,Robert+Mila+Toni;  
END;
```

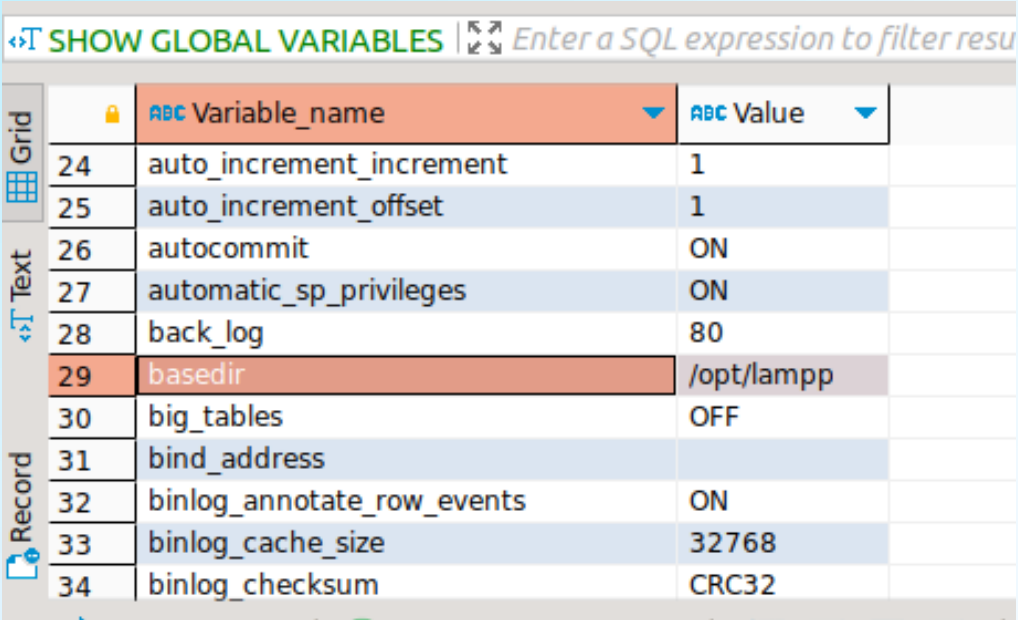
Kita panggil dengan perintah:

```
CALL gaji();
```

# Contoh System Variable:

## Global:

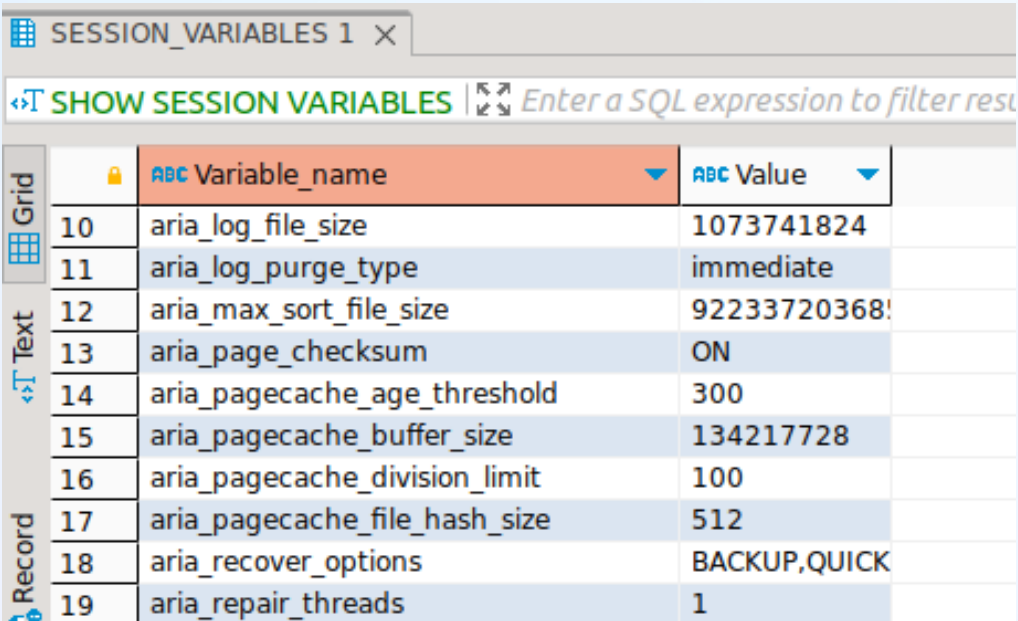
**SHOW GLOBAL VARIABLES;**



	Variable_name	Value
24	auto_increment_increment	1
25	auto_increment_offset	1
26	autocommit	ON
27	automatic_sp_privileges	ON
28	back_log	80
29	basedir	/opt/lampp
30	big_tables	OFF
31	bind_address	
32	binlog_annotate_row_events	ON
33	binlog_cache_size	32768
34	binlog_checksum	CRC32

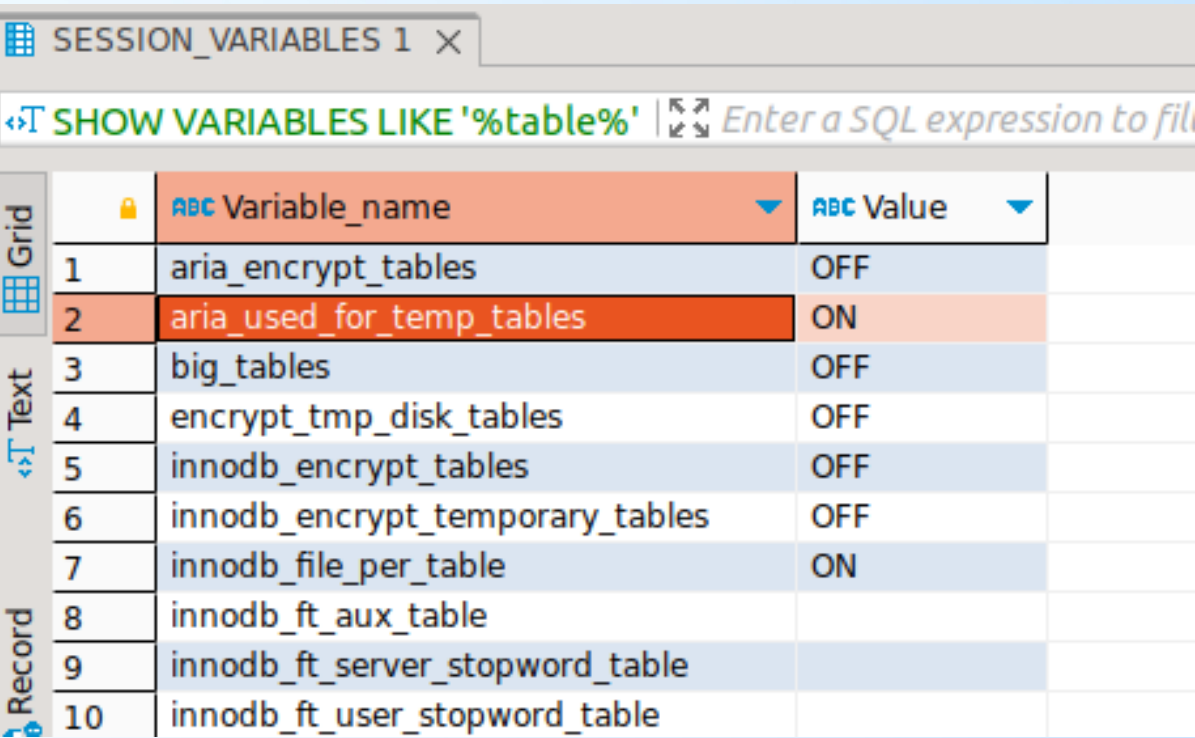
## Session

**SHOW SESSION VARIABLES;**



	Variable_name	Value
10	aria_log_file_size	1073741824
11	aria_log_purge_type	immediate
12	aria_max_sort_file_size	92233720368
13	aria_page_checksum	ON
14	aria_pagecache_age_threshold	300
15	aria_pagecache_buffer_size	134217728
16	aria_pagecache_division_limit	100
17	aria_pagecache_file_hash_size	512
18	aria_recover_options	BACKUP,QUICK
19	aria_repair_threads	1

**SHOW VARIABLES LIKE '%table%';**



	Variable_name	Value
1	aria_encrypt_tables	OFF
2	aria_used_for_temp_tables	ON
3	big_tables	OFF
4	encrypt_tmp_disk_tables	OFF
5	innodb_encrypt_tables	OFF
6	innodb_encrypt_temporary_tables	OFF
7	innodb_file_per_table	ON
8	innodb_ft_aux_table	
9	innodb_ft_server_stopword_table	
10	innodb_ft_user_stopword_table	

# Percabangan

- suatu control untuk mengecek suatu kondisi yang dilakukan sebelum statement-statement dalam sebuah blok PL/SQL dieksekusi.
- Satu atau lebih body percabangan akan dieksekusi bila kondisi bernilai **TRUE**.
- Satu atau lebih body percabangan tidak akan pernah di eksekusi bila kondisinya bernilai **FALSE**.

# Percabangan

Statement yang digunakan untuk percabangan ada 2:

## 1. IF-THEN-ELSE Statement

```
IF kondisi THEN statemen  
    statemen  
END IF
```

## 2. Struktur IF → bisa 1 kondisi, 2 kondisi, 3 kondisi atau lebih

```
IF kondisi THEN statemen  
    ELSE IF kondisi THEN statemen  
    ...  
    ELSE statemen  
END IF
```

## 3. CASE Statement

```
CASE (ekspresi)  
    WHEN nilai_1 THEN statemen_1;  
    WHEN nilai_2 THEN statemen_2;  
    ...  
    WHEN nilai_n THEN statemen_n;  
    ELSE statemen_lain;  
END CASE
```

# Percabangan

## IF-THEN-ELSE Statement

### IF Satu Kondisi

- Hanya mempunyai satu kondisi, jika kondisi bernilai **TRUE** body percabangan akan dieksekusi, sebaliknya jika **FALSE**, tidak akan pernah dieksekusi.
- Structure if pada query MySQL:

```
IF(ekspresi, true, false)
```

# Percabangan

## IF-THEN-ELSE Statement

Contoh IF Satu Kondisi pada MySQL SELECT Statement:

```
SELECT
    *,
    IF (tbmb_harga < 5000, 'Murah', 'Mahal') AS Harga
FROM tb_master_barang;
```

Contoh diatas hanya dapat dijalankan di **MySQL**. Dicoba jalankan di database lain (PostgreSQL atau lainnya) error.

Solusi supaya tidak error (bisa di execute di semua product database, bisa gunakan CASE):

```
SELECT *,
CASE
    WHEN tbmb_harga < 5000 THEN 'Murah'
    ELSE 'Mahal'
END AS Harga
FROM tb_master_barang;
```



# Percabangan

## IF-THEN-ELSE Statement

Contoh IF satu kondisi (**IF-THEN**) dalam Store Procedure:

```
CREATE OR REPLACE PROCEDURE salary(v_salary INT)
BEGIN
DECLARE v_hasil varchar(200);
  IF v_salary < 5000000 THEN SET v_hasil = 'Salary is less than 5000000.';
  END IF;
  SELECT v_hasil;
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL salary(5500000); -- > 5000
-- OR
CALL salary(4500000); -- < 5000
```



# Percabangan

## IF-THEN-ELSE Statement

Contoh IF dua kondisi (**IF-THEN-ELSE**) dalam Store Procedure:

```
CREATE OR REPLACE PROCEDURE salary_2_kondisi(v_salary INT)
BEGIN
DECLARE v_hasil varchar(200);
  IF v_salary < 5000000 THEN SET v_hasil = 'Salary is less than 5000000.';
  ELSE SET v_hasil = 'Salary is more than 5000000.';
  END IF;
  SELECT v_hasil;
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL salary_2_kondisi(5500000); -- > 5000
-- OR
CALL salary_2_kondisi(4500000); -- < 5000
```

# Percabangan

## IF-THEN-ELSE Statement

Contoh IF tiga kondisi (**IF-THEN-ELSE IF-ELSE**) dalam Store Procedure:

```
CREATE OR REPLACE PROCEDURE salary_3_kondisi(v_salary INT)
BEGIN
DECLARE v_hasil varchar(200);
  IF      v_salary < 5000000 THEN SET v_hasil = 'Salary is less than 5000000.';
  ELSEIF  v_salary = 5000000 THEN SET v_hasil = 'Salary is equal to 5000000.';
  ELSE SET v_hasil = 'Salary is more than 5000000.';
  END IF;
  SELECT v_hasil;
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL salary_3_kondisi(5500000); -- > 5000000
-- OR
CALL salary_3_kondisi(4500000); -- < 5000000
-- OR
CALL salary_3_kondisi(5000000); -- = 5000000
```

# Percabangan

## IF-THEN-ELSE Statement

Contoh IF lebih dari tiga kondisi dalam Store Procedure:

```
CREATE OR REPLACE PROCEDURE bulan(v_bulan INT)
BEGIN
  DECLARE v_nama_bulan varchar(20);
  IF v_bulan = 1 THEN SET v_nama_bulan = 'Januari';
  ELSEIF v_bulan = 2 THEN SET v_nama_bulan = 'Februari';
  ELSEIF v_bulan = 3 THEN SET v_nama_bulan = 'Maret';
  ELSEIF v_bulan = 4 THEN SET v_nama_bulan = 'April';
  ELSEIF v_bulan = 5 THEN SET v_nama_bulan = 'Mei';
  ELSEIF v_bulan = 6 THEN SET v_nama_bulan = 'Juni';
  ELSEIF v_bulan = 7 THEN SET v_nama_bulan = 'Juli';
  ELSEIF v_bulan = 8 THEN SET v_nama_bulan = 'Agustus';
  ELSEIF v_bulan = 9 THEN SET v_nama_bulan = 'September';
  ELSEIF v_bulan = 10 THEN SET v_nama_bulan = 'Oktober';
  ELSEIF v_bulan = 11 THEN SET v_nama_bulan = 'November';
  ELSEIF v_bulan = 12 THEN SET v_nama_bulan = 'Desember';
  ELSE SET v_nama_bulan = 'Not Found';
  END IF;
  SELECT v_nama_bulan;
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL bulan(1);
-- OR
CALL bulan(12);
-- OR
CALL bulan(13);
```

# Percabangan

## CASE WHEN Statement

Contoh CASE satu kondisi dalam Store Procedure:

```
CREATE OR REPLACE PROCEDURE salary_case(v_salary INT)
BEGIN
  DECLARE v_hasil varchar(200);
  CASE
    WHEN v_salary < 5000000 THEN SET v_hasil = 'Salary is less than 5000000.';
  END CASE;
  SELECT v_hasil;
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL salary_case(4000000);
```

# Percabangan

## CASE WHEN Statement

Contoh CASE dua kondisi dalam Store Procedure:

```
CREATE OR REPLACE PROCEDURE salary_case_2_kondisi(v_salary INT)
BEGIN
  DECLARE v_hasil varchar(200);
  CASE
    WHEN v_salary < 5000000 THEN SET v_hasil = 'Salary is less than 5000000.';
    ELSE SET v_hasil = 'Salary is equal or more than 5000000.';
  END CASE;
  SELECT v_hasil;
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL salary_case(4000000);
```

OR

```
CALL salary_case(6000000);
```

# Percabangan

## CASE WHEN Statement

Contoh CASE tiga kondisi dalam Store Procedure:

```
CREATE OR REPLACE PROCEDURE salary_case_3_kondisi(v_salary INT)  
BEGIN  
DECLARE v_hasil varchar(200);  
    CASE  
        WHEN v_salary < 5000000 THEN SET v_hasil = 'Salary is less than 5000000.';  
        WHEN v_salary = 5000000 THEN SET v_hasil = 'Salary is equal to 5000000.';  
        ELSE SET v_hasil = 'Salary is more than 5000000.';  
    END CASE;  
    SELECT v_hasil;  
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL salary_case_3_kondisi(4000000);  
OR  
CALL salary_case_3_kondisi(6000000);  
OR  
CALL salary_case_3_kondisi(5000000);
```



# Percabangan

## CASE WHEN Statement

Contoh CASE lebih dari tiga kondisi dalam Store Procedure:

```
CREATE OR REPLACE PROCEDURE case_bulan(v_bulan INT)
BEGIN
  DECLARE v_nama_bulan varchar(20);
  CASE
    WHEN v_bulan = 1 THEN SET v_nama_bulan = 'Januari';
    WHEN v_bulan = 2 THEN SET v_nama_bulan = 'Februari';
    WHEN v_bulan = 3 THEN SET v_nama_bulan = 'Maret';
    WHEN v_bulan = 4 THEN SET v_nama_bulan = 'April';
    WHEN v_bulan = 5 THEN SET v_nama_bulan = 'Mei';
    WHEN v_bulan = 6 THEN SET v_nama_bulan = 'Juni';
    WHEN v_bulan = 7 THEN SET v_nama_bulan = 'Juli';
    WHEN v_bulan = 8 THEN SET v_nama_bulan = 'Agustus';
    WHEN v_bulan = 9 THEN SET v_nama_bulan = 'September';
    WHEN v_bulan = 10 THEN SET v_nama_bulan = 'Oktober';
    WHEN v_bulan = 11 THEN SET v_nama_bulan = 'November';
    WHEN v_bulan = 12 THEN SET v_nama_bulan = 'Desember';
    ELSE SET v_nama_bulan = 'Not Found';
  END CASE;
  SELECT v_nama_bulan;
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL case_bulan(1);
```



# Perulangan

- Pada **MySQL** terdapat Perulangan atau **LOOP Statement** yang digunakan untuk menjalankan blok Procedure / SQL secara berulang-ulang sampai mencapai kondisi tertentu.
- Ada 3 macam perulangan yang digunakan pada **MySQL: LOOP, WHILE, REPEAT.**

Struktur perulangan bisa salah satu atau gabungan dari statement berikut:

1. Loop Statement
2. While Statement
3. Repeat Statement
4. Iterate Statement
5. Leave Statement
6. Return Statement

# Perulangan

## Loop Statement

Pada MySQL, **LOOP** statement digunakan untuk menjalankan blok Procedure / SQL secara berulang-ulang sampai mencapai kondisi tertentu. Syntax:

```
Loop_label: LOOP  
    -- Statements...  
  
END LOOP label_name ;
```

- **Optional** pada **label\_name**.
- **label\_name** digunakan ketika mengeksekusi sebuah **ITERATE statement** atau **LEAVE statement**.
- **LOOP** dapat diakhiri dengan **RETURN statement** atau **LEAVE statement**.

# Perulangan

## While Statement (WHILE - LOOP)

Pada MySQL, **WHILE** statement juga digunakan untuk menjalankan blok Procedure / SQL secara berulang-ulang sampai mencapai kondisi tertentu. Syntax:

```
[ label_name: ] WHILE condition  
DO  
    statements...;  
END WHILE [ label_name ];
```

- **Optional** pada **label\_name**.
- **WHILE** akan menguji setiap proses yang melewatinya. Jika **true**, **LOOP** body akan dieksekusi. Sebaliknya jika **FALSE** proses **WHILE - LOOP** akan dihentikan.

# Perulangan

## REPEAT Statement (REPEAT-UNTIL-LOOP)

Pada MySQL, **REPEAT** statement juga digunakan untuk menjalankan blok Procedure / SQL secara berulang-ulang sampai mencapai kondisi tertentu. Syntax:

```
[ label_name: ] REPEAT  
    ...statements...  
UNTIL condition  
END REPEAT [ label_name ];
```

- **Optional** pada **label\_name**.
- **REPEAT** akan menguji setiap proses yang melewatinya pada blok statement.
- **UNTIL** condition digunakan untuk terminat proses **REPEAT**.

# Perulangan

## ITERATE Statement

Pada MySQL, **ITERATE** statement digunakan ketika kita menginginkan **LOOP** body dieksekusi ulang (berulang). **ITERATE** bisa digunakan pada **LOOP** statement, **WHILE** statement dan **REPEAT** statement. Syntax:

```
ITERATE label_name;
```

# Perulangan

## LEAVE Statement

Pada MySQL, **LEAVE** statement digunakan ketika kita menginginkan keluar / terminate suatu proses dari blok code yang diidentifikasi dengan **nama\_label**. Bisa digunakan pada **LOOP**, **WHILE** atau **REPEAT** statement. Syntax:

```
LEAVE label_name;
```



# Perulangan

## RETURN Statement

Pada MySQL, **RETURN** statement digunakan ketika ingin keluar dari suatu fungsi ataupun LOOP dan mengembalikan hasil atau nilai dari fungsi tersebut.

```
RETURN result;
```



# Contoh Perulangan 1

Dengan LOOP, LEAVE

```
CREATE OR REPLACE PROCEDURE test_loop_leave()  
BEGIN  
    DECLARE x int;  
    DECLARE output varchar(50);  
  
    SET x = 1;  
    SET output = "";  
  
    loop_label_name: LOOP  
        IF x > 5 THEN  
            LEAVE loop_label_name;  
        END IF;  
        SET output = CONCAT(output,x," ");  
        SET x = x + 1;  
  
    END LOOP loop_label_name;  
    SELECT output;  
  
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL test_loop_leave();
```

# Contoh Perulangan 2

Dengan LOOP, REPEAT, UNTIL

```
CREATE OR REPLACE PROCEDURE test_loop_repeat(IN batas
INT)
BEGIN
    DECLARE i int;
    DECLARE hasil varchar(50) DEFAULT ' ';
    SET i = 1;

    REPEAT
        SET hasil = CONCAT(hasil, i, ' ');
        SET i = i + 1;
    UNTIL i > batas
    END REPEAT;
    SELECT hasil;
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL test_loop_repeat(10);
```

# Contoh Perulangan 3

Dengan WHILE

```
CREATE OR REPLACE PROCEDURE test_loop_while(IN batas INT)
BEGIN
    DECLARE i int;
    DECLARE hasil varchar(50) DEFAULT '';
    SET i = 1;
    WHILE i < batas
    DO
        SET hasil = CONCAT(hasil, i, ' ');
        SET i = i + 1;
    END WHILE;
    SELECT hasil;
END;
```

Cobalah panggil Procedure dengan perintah:

```
CALL test_loop_while(10);
```

# Contoh Perulangan 4

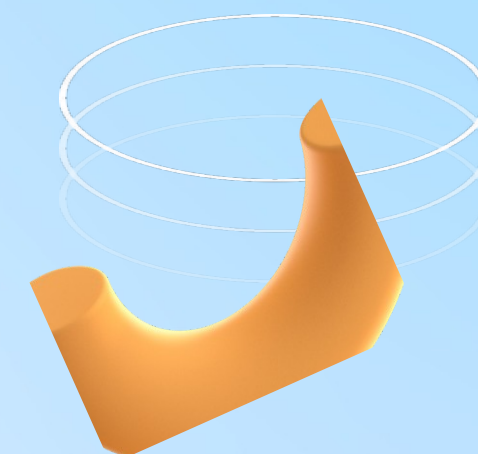
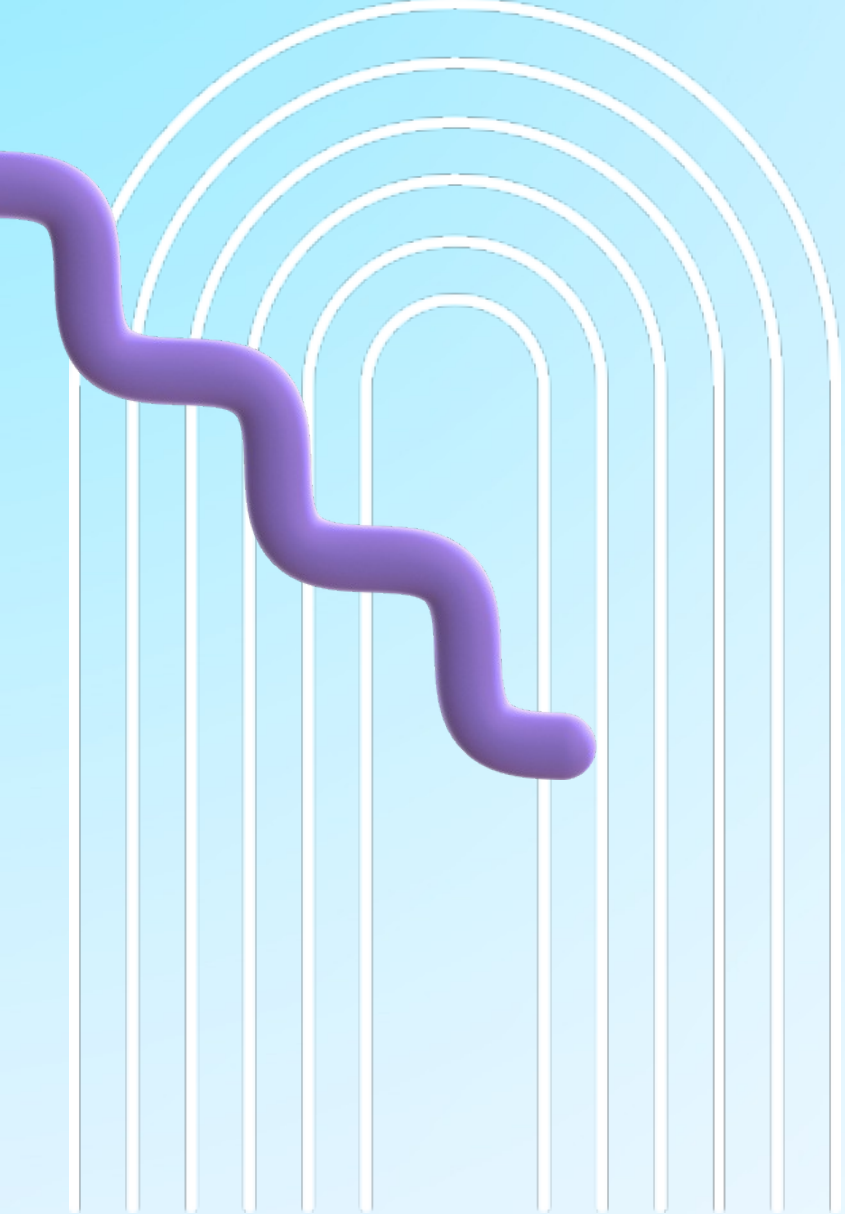
Dengan LOOP, LEAVE, ITERATE menampilkan genap

```
CREATE OR REPLACE PROCEDURE test_loop_leave_iterate(IN batas INT)
BEGIN
    DECLARE i int;
    DECLARE hasil varchar(50) DEFAULT '';
    SET i = 1;
label_ulang: LOOP
    IF i > batas THEN
        LEAVE label_ulang;
    END IF;
    SET i = i + 1;
    IF(i mod 2 !=0) THEN
        ITERATE label_ulang;
    ELSE
        SET hasil = CONCAT(hasil, i, ' ');
    END IF;
END LOOP;
SELECT hasil;
END;
```

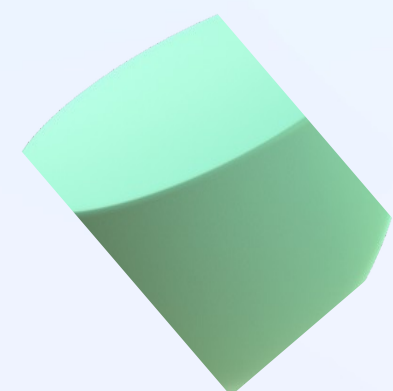
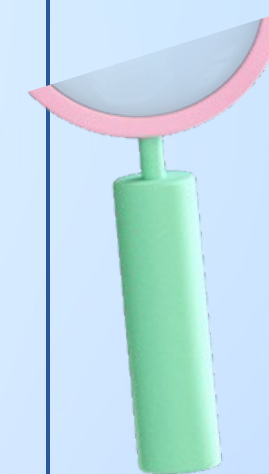
Cobalah panggil Procedure dengan perintah:

```
CALL test_loop_while(10);
```

DO YOU HAVE  
ANY QUESTION?



# Challenge Soal / Task





# SOAL

Pada contoh procedure bulan dibawah ini, buatlah tabel **tb\_hasil** dengan perintah DDL untuk menyimpan v\_nama\_bulan kedalam kolom **tbh\_hasil** dalam tabel **tb\_hasil**.

```
CREATE OR REPLACE PROCEDURE bulan(v_bulan INT)
BEGIN
DECLARE v_nama_bulan varchar(20);
    IF v_bulan = 1 THEN SET v_nama_bulan = 'Januari';
    ELSEIF v_bulan = 2 THEN SET v_nama_bulan = 'Februari';
    ELSEIF v_bulan = 3 THEN SET v_nama_bulan = 'Maret';
    ELSEIF v_bulan = 4 THEN SET v_nama_bulan = 'April';
    ELSEIF v_bulan = 5 THEN SET v_nama_bulan = 'Mei';
    ELSEIF v_bulan = 6 THEN SET v_nama_bulan = 'Juni';
    ELSEIF v_bulan = 7 THEN SET v_nama_bulan = 'Juli';
    ELSEIF v_bulan = 8 THEN SET v_nama_bulan = 'Agustus';
    ELSEIF v_bulan = 9 THEN SET v_nama_bulan = 'September';
    ELSEIF v_bulan = 10 THEN SET v_nama_bulan = 'Oktober';
    ELSEIF v_bulan = 11 THEN SET v_nama_bulan = 'November';
    ELSEIF v_bulan = 12 THEN SET v_nama_bulan = 'Desember';
    ELSE SET v_nama_bulan = 'Not Found';
    END IF;
    SELECT v_nama_bulan;

END;
```

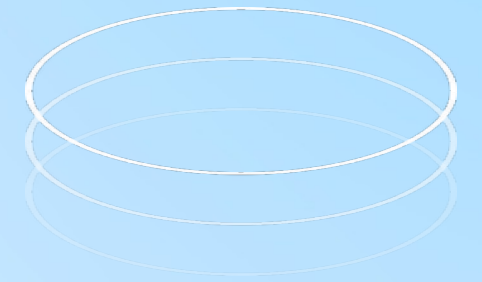
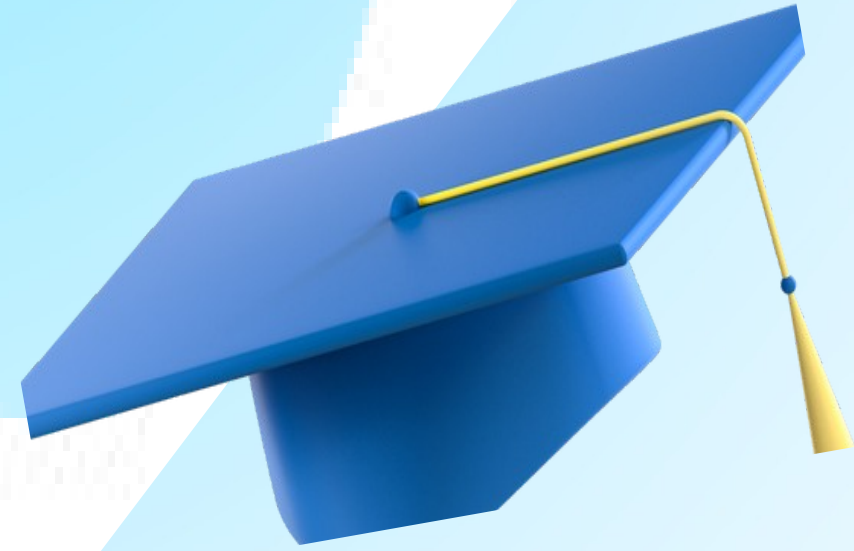
Cobalah panggil Procedure dengan perintah:

```
CALL bulan(13);
```



Jawaban soal bisa di upload pada google form berikut:

<https://forms.gle/FsaeeEKQEyTtrZ9h7>



# THANK YOU

FOR YOUR ATTENTION

