**LAPORAN RESMI**

**MODUL VI**

**STORED PROCEDURE : BRANCING / LOOPING**



**NAMA : ANISYAFAAH**

**N.R.P : 220441100105**

**DOSEN : FITRI DAMAYANTI, S.Kom., M.Kom.**

**ASISTEN : AFFAN MAULANA ZULKARNAIN**

**TGL PRAKTIKUM : 17 MEI 2024**

**Disetujui : Mei 2024**

**Asisten**

**AFFAN MAULANA ZULKARNAIN**

**20.04.411.00052**

**LABORATORIUM BISNIS INTELIJEN SISTEM**

**PRODI SISTEM INFORMASI**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

# **BAB I PENDAHULUAN**

## **Latar Belakang**

Di era digital saat ini, teknologi informasi berkembang pesat dan mengubah cara perusahaan beroperasi. Basis data menjadi salah satu komponen kunci dalam manajemen informasi, dimana data digunakan untuk mendukung keputusan bisnis yang lebih baik. Stored procedure, yang merupakan sekumpulan perintah SQL yang disimpan dan dijalankan di server basis data, memiliki peran penting dalam mengelola dan memproses data dengan efisien. Dalam konteks ini, kemampuan branching dan looping dalam stored procedure memberikan fleksibilitas dan kontrol yang lebih besar dalam operasi basis data.

Penggunaan stored procedure dengan branching (percabangan) dan looping (pengulangan) memungkinkan pengembang untuk menyederhanakan dan mengoptimalkan tugas-tugas yang berulang dan kompleks. Misalnya, dalam kasus pemrosesan batch data atau pengelolaan transaksi yang melibatkan banyak langkah, stored procedure dapat memproses data secara efisien dan konsisten. Branching memungkinkan pengambilan keputusan dinamis, sementara looping memungkinkan eksekusi berulang terhadap sekumpulan data.

Selain itu, stored procedure meningkatkan keamanan dan integritas data. Dengan mengeksekusi logika bisnis di sisi server, risiko kebocoran data dapat diminimalkan karena data tidak perlu ditransfer ke aplikasi klien untuk diproses. Ini juga memungkinkan pengendalian akses yang lebih ketat terhadap data sensitif, karena hanya pengguna yang memiliki izin yang dapat mengeksekusi stored procedure tertentu. Dengan demikian, penggunaan stored procedure yang memanfaatkan branching dan looping dapat memberikan lapisan tambahan perlindungan terhadap ancaman keamanan siber.

## **Tujuan**

* Mampu mengerti tentang Stored Procedure
* Mampu menggunakan Variabel, Brancing/Looping serta Percabangan pada Stored Procedure

# **BAB II DASAR TEORI**

## **Variabel**

Dideklarasikan dengan keyword “DECLARE” kemudian diikuti dengan nama variabel dan tipe data.

Sintaks:

**DECLARE namavariabel TYPE DEFAULT nilai;**

Untuk **DEFAULT** sifatnya opsional

DELIMITER //

**CREATE PROCEDURE** dec1()

**BEGIN**

**DECLARE COUNT INT** DEFAULT0;

**DECLARE x INT**;

**DECLARE** message **VARCHAR**(100);

**END**;

DELIMITER ;

Contoh Cetak Nama :

DELIMITER |

**CREATE PROCEDURE** Excetak\_nm( **IN** namadepan **VARCHAR**(20),

**IN** namabelakang **VARCHAR**(30), **IN** gelardepan **CHAR** (6),

**IN** gelarbalakang **CHAR**(10))

**BEGIN**

**DECLARE** nama **VARCHAR**(50);

**SET** nama =**CONCAT** (gelardepan, " ",namadepan, " ",

namabelakang, " ", gelarbelakang);

**SELECT** nama **AS** hasil;

**END**;

DELIMITER ;

Contoh prosedur untuk mencetak nama :

**call** cetak\_nm("Doni","Abdul","Dr.","S.Kom, M.Kom");

## **Pencabangan dan Pengulangan**

### **Pencabangan**

Penggunaan pernyataan-pernyataan pencabangan ataupun pengulangan di dalam stored procedure merupakan tindakan yang legal. Dengan demikian, kita bisa menghasilkan suatu prosedur yang kompleks. Perintah pemilihan ini berupa statement-statement yang akan mengerjakan instruksi jika kondisi benar/terpenuhi.

Contoh berikut memperlihatkan penggunaan pernyataan IF

Sintak:

**IF** [val] **THEN IF** [val] **THEN**

[result1] [result1]

**END IF ; ELSE**

[result2]

**END IF**

DELIMITER //

**CREATE PROCEDURE** cobaIF(

**IN** bil **INT**(3)

)

**BEGIN**

*/\*Deklarasi Variabel\*/*

**DECLARE** str **VARCHAR**(50);

**if** (bil<0) **then**

**SET** str ='Bilangan Negetif';

**ELSE**

**SET** str='Bilangan Posistif';

**END if**; **SELECT** str;

**END** //

DELIMITER;

Atau

DELIMITER |

**CREATE PROCEDURE** exIF(

**IN** bil **INT**(3)

)

**BEGIN**

*/\*Deklarasi Variabel\*/*

**DECLARE** str **VARCHAR**(50);

**if** (bil<0) **then**

**SET** str ='Bilangan Negetif';

**ELSE**

**SET** str='Bilangan Posistif';

**END if**;

**SELECT** str;

**END**;

|

DELIMITER ;

Untuk memanggilnya:

**call** exIF(7);

### **Pengulangan**

Perintah perulangan dengan menggunakan statement LOOP, WHILE, dan REPEAT., Penggunaan statement LOOP diawali dengan menentukan nama perulangan: LOOP dan diakhiri dengan END LOOP.

Statement WHILE – Statement WHILE melakukan perulangan berdasarkan kondisi tertentu. Perulangan akan dilakukan jika kondisi bernilai benar/true.

**Sintak Loop :**

Loop\_name : **LOOP** [statement1] [statement2]

**END LOOP** loop\_name

**Sintak While :**

Loop\_Name : **WHILE** [condition] **DO**

[statement1] [statement2]

**END WHILE** Loop\_Name;

**Sintak Repeat :**

[begin\_label:] **REPEAT**

statement\_list

**UNTIL** search\_condition

**END REPEAT** [end\_label]

Contoh penggunaan looping adalah sebagai berikut

DELIMITER |

**CREATE PROCEDURE** ExLoop(

**IN** bil **INT**(3)

)

**BEGIN**

*/\*Deklarasi Variabel\*/* **DECLARE** str **VARCHAR**(150); **DECLARE** i **int**(3);

**Set** i=1;

**Set** str= ;

**While** i<= bil **do**

**Set** str=**concat** (str, i, , );

**Set** i=i+1;

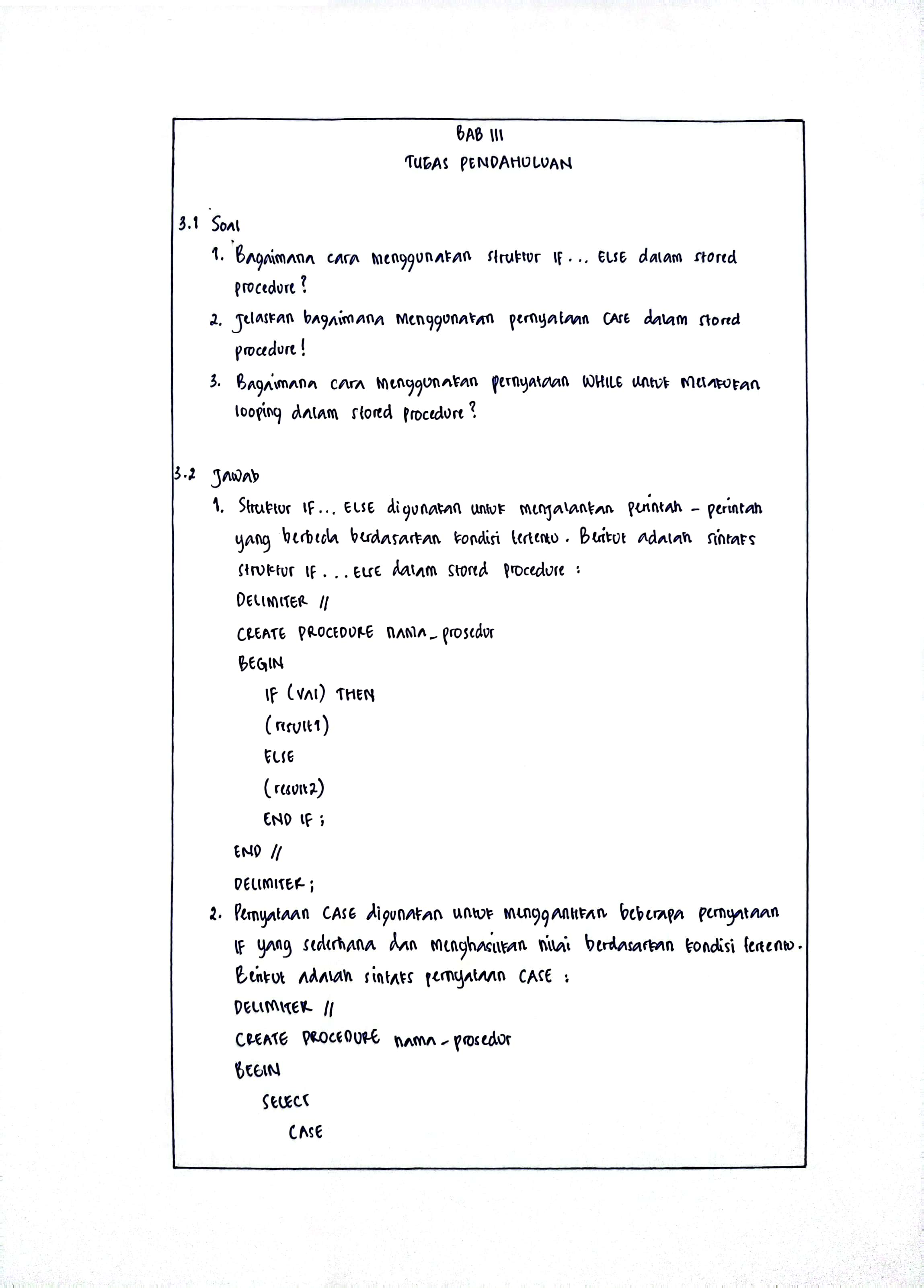
**END WHILE;**

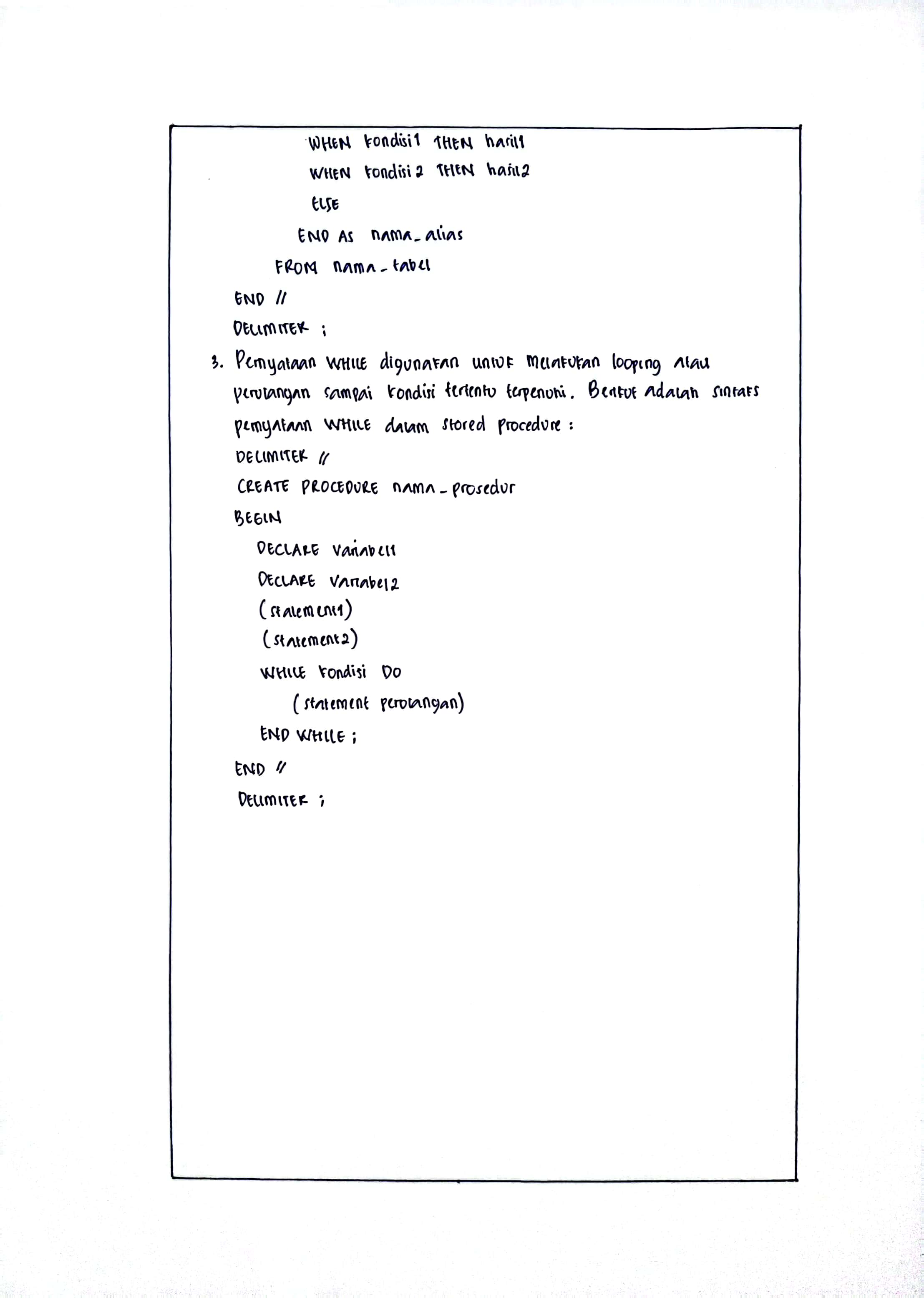
**SELECT** str;

**END**;

|

DELIMITER ;

****

****

# **BAB IV IMPLEMENTASI**

## **Source Code**

### **Mengisi Data Setiap Tabel**

* **Tabel Projects**

1. **Source Code**

|  |
| --- |
| CREATE DATABASE OutSourcingCompany;  USE OutSourcingCompany;  CREATE TABLE Projects (  Project\_ID INT NOT NULL PRIMARY KEY,  Project\_Name VARCHAR(100),  Client\_Company VARCHAR(100),  Start\_Date DATE,  End\_Date DATE  );  INSERT INTO Projects (Project\_ID, Project\_Name, Client\_Company, Start\_Date, End\_Date) VALUES  (1, 'Proyek Alpha', 'Tech Innovators Inc.', '2023-01-10', '2023-06-15'),  (2, 'Proyek Beta', 'Green Energy Solutions', '2023-02-01', '2023-08-01'),  (3, 'Proyek Gamma', 'Healthcare Dynamics', '2023-03-05', '2023-09-05'),  (4, 'Proyek Delta', 'EduTech Global', '2023-04-10', '2023-12-20'),  (5, 'Proyek Epsilon', 'FinanceSecure Ltd.', '2023-05-15', '2023-11-30'),  (6, 'Proyek Zeta', 'AutoDrive Corp.', '2023-06-20', '2023-12-31'),  (7, 'Proyek Eta', 'FoodTech Enterprises', '2023-07-25', '2024-01-30'),  (8, 'Proyek Theta', 'SmartHomes Inc.', '2023-08-10', '2024-02-15'),  (9, 'Proyek Iota', 'TravelPlus Ltd.', '2023-09-05', '2024-03-10'),  (10, 'Proyek Kappa', 'Retail Innovations', '2023-10-10', '2024-04-20'); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat database Out Sourcing Company dan membuat tabel projects serta mengisinya dengan 10 data. Untuk membuat database Out Sourcing Company menggunakan perintah CREATE DATABASE OutSourcingCompany. Untuk membuat tabel projects menggunakan perintah CREATE TABLE Projects (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id project sebagai primary key, nama project, client company, start date, dan end date. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

* **Tabel Employees**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Employees (  Employee\_ID INT NOT NULL PRIMARY KEY,  Name VARCHAR(100),  Position VARCHAR(50),  Salary FLOAT,  Date\_Hired DATE,  Project\_ID INT,  FOREIGN KEY (Project\_ID) REFERENCES Projects (Project\_ID)  );  INSERT INTO Employees (Employee\_ID, Name, Position, Salary, Date\_Hired, Project\_ID) VALUES  (1, 'Alice Johnson', 'Manajer Proyek', 75000, '2023-01-15', 1),  (2, 'Bob Smith', 'Insinyur Perangkat Lunak', 68000, '2023-02-10', 2),  (3, 'Charlie Brown', 'Analis Data', 60000, '2023-03-20', 3),  (4, 'David Wilson', 'Desainer UX', 55000, '2023-04-25', 4),  (5, 'Eva Adams', 'Spesialis Pemasaran', 52000, '2023-05-30', 5),  (6, 'Frank White', 'Insinyur QA', 57000, '2023-06-05', 6),  (7, 'Grace Lee', 'Analis Bisnis', 62000, '2023-07-10', 7),  (8, 'Henry King', 'Insinyur DevOps', 64000, '2023-08-15', 8),  (9, 'Ivy Scott', 'Manajer HR', 58000, '2023-09-20', 9),  (10, 'Jack Miller', 'Akuntan', 56000, '2023-10-25', 10); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel employees dan mengisi data sebanyak 10 data. Untuk membuat tabel supplier menggunakan perintah CREATE TABLE Employees (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id employee sebagai primary key, nama employee, position, salary, date hired, dan id project yang terhubung dengan tabel project menggunakan FOREIGN KEY. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

* **Tabel Certificates**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Certificates (  Certificate\_ID INT NOT NULL PRIMARY KEY,  Employee\_ID INT,  Certificate\_Name VARCHAR(100),  Issue\_Date DATE,  Expiry\_Date DATE,  FOREIGN KEY (Employee\_ID) REFERENCES Employees (Employee\_ID)  );  INSERT INTO Certificates (Certificate\_ID, Employee\_ID, Certificate\_Name, Issue\_Date, Expiry\_Date) VALUES  (1, 1, 'Project Management Professional (PMP)', '2023-02-01', '2026-02-01'),  (2, 2, 'Certified Scrum Master (CSM)', '2023-03-15', '2025-03-15'),  (3, 3, 'Certified Data Analyst (CDA)', '2023-04-20', '2025-04-20'),  (4, 4, 'Sertifikasi Desain UX', '2023-05-10', '2026-05-10'),  (5, 5, 'Sertifikasi Pemasaran Digital', '2023-06-05', '2025-06-05'),  (6, 6, 'Certified Software Quality Analyst (CSQA)', '2023-07-01', '2025-07-01'),  (7, 7, 'Certified Business Analysis Professional (CBAP)', '2023-08-12', '2026-08-12'),  (8, 8, 'AWS Certified DevOps Engineer', '2023-09-20', '2025-09-20'),  (9, 9, 'Professional in Human Resources (PHR)', '2023-10-05', '2026-10-05'),  (10, 10, 'Certified Public Accountant (CPA)', '2023-11-15', '2026-11-15'); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel certificates dan mengisi data sebanyak 10 data. Untuk membuat tabel gudang menggunakan perintah CREATE TABLE Certificates (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id certificate sebagai primary key, employee id ynag terhubung dengan tabel employees menggunakan FOREIGN KEY, certificate name, issue date, dan expiry date. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

* **Tabel Trainings**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Trainings (  Training\_ID INT NOT NULL PRIMARY KEY,  Certificate\_Name VARCHAR(100),  Duration\_In\_Months INT  );  INSERT INTO Trainings (Training\_ID, Certificate\_Name, Duration\_In\_Months) VALUES  (1, 'Project Management Professional (PMP)', 6),  (2, 'Certified Scrum Master (CSM)', 2),  (3, 'Certified Data Analyst (CDA)', 4),  (4, 'Sertifikasi Desain UX', 5),  (5, 'Sertifikasi Pemasaran Digital', 3),  (6, 'Certified Software Quality Analyst (CSQA)', 4),  (7, 'Certified Business Analysis Professional (CBAP)', 6),  (8, 'AWS Certified DevOps Engineer', 5),  (9, 'Professional in Human Resources (PHR)', 3),  (10, 'Certified Public Accountant (CPA)', 6); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel trainings dan mengisi data sebanyak 10 data. Untuk membuat tabel stok menggunakan perintah CREATE TABLE Trainings (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id training sebagai primary key, certificate name, dan duration in month. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

* **Tabel Notifications**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Notifications (  Notification\_ID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,  Employee\_ID INT,  Message TEXT,  Date DATE,  FOREIGN KEY (Employee\_ID) REFERENCES Employees (Employee\_ID)  );  INSERT INTO Notifications (Notification\_ID, Employee\_ID, Message, Date) VALUES  (1, 1, 'Pengingat: Sertifikasi PMP Anda akan segera berakhir. Harap perbarui sebelum tanggal kedaluwarsa.', '2024-01-15'),  (2, 2, 'Selamat telah menyelesaikan pelatihan Certified Scrum Master (CSM)!', '2023-03-16'),  (3, 3, 'Sertifikasi Certified Data Analyst (CDA) Anda telah diterbitkan.', '2023-04-21'),  (4, 4, 'Pengingat: Ujian Sertifikasi Desain UX dijadwalkan minggu depan.', '2023-05-03'),  (5, 5, 'Sertifikasi Pemasaran Digital Anda akan kedaluwarsa dalam enam bulan. Harap perbarui.', '2024-01-05'),  (6, 6, 'Selamat telah meraih sertifikasi Certified Software Quality Analyst (CSQA)!', '2023-07-02'),  (7, 7, 'Sertifikasi CBAP Anda telah berhasil diperbarui.', '2023-08-13'),  (8, 8, 'Sesi pelatihan AWS Certified DevOps Engineer dimulai bulan depan.', '2023-08-20'),  (9, 9, 'Pengingat: Sertifikasi Professional in Human Resources (PHR) Anda akan segera berakhir.', '2026-09-30'),  (10, 10, 'Anda telah berhasil lulus ujian Certified Public Accountant (CPA).', '2023-11-16'); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel notifications dan mengisi data sebanyak 10 data. Untuk membuat tabel karyawan menggunakan perintah CREATE TABLE Notifications (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id notification sebagai primary key, id employee yang terhubung dengan tabel employees menggunakan perintah FOREIGN KEY, message, dan date. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

* **Tabel Companies**

1. **Source Code**

|  |
| --- |
| CREATE TABLE Companies (  Company\_ID INT NOT NULL PRIMARY KEY,  Company\_Name VARCHAR(100),  Address TEXT  );  INSERT INTO Companies (Company\_ID, Company\_Name, Address) VALUES  (1, 'Tech Innovators Inc.', '123 Tech Street, Silicon Valley, CA'),  (2, 'Green Energy Solutions', '456 Renewable Avenue, New York, NY'),  (3, 'Healthcare Dynamics', '789 Medical Center Drive, Chicago, IL'),  (4, 'EduTech Global', '101 Education Boulevard, Boston, MA'),  (5, 'FinanceSecure Ltd.', '202 Financial District, London, UK'),  (6, 'AutoDrive Corp.', '303 Autonomous Road, Tokyo, Japan'),  (7, 'FoodTech Enterprises', '404 Food Plaza, Paris, France'),  (8, 'SmartHomes Inc.', '505 Smart Street, Sydney, Australia'),  (9, 'TravelPlus Ltd.', '606 Travel Avenue, Dubai, UAE'),  (10, 'Retail Innovations', '707 Retail Road, Toronto, Canada'); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat tabel companies dan mengisi data sebanyak 10 data. Untuk membuat tabel transaksi menggunakan perintah CREATE TABLE Companies (namaKolom1, namaKolom2, dst.). Tabel di atas memiliki kolom yang terdiri dari id company sebagai primary keycompany name, dan address. Selanjutnya untuk mengisi data pada setiap kolom menggunakan perintah INSERT INTO.

### **Stored Procedure Menambah Salary Karyawan 10% yang Bekerja Lebih dari Satu Tahun**

1. **Source Code**

|  |
| --- |
| -- Nomor 1  DELIMITER //  CREATE PROCEDURE TambahanBonus()  BEGIN  DECLARE tglSekarang DATE;  SET tglSekarang = CURDATE();  UPDATE Employees  SET Salary = Salary \* 1.10  WHERE DATEDIFF(tglSekarang, Date\_Hired) > 365;  SELECT Employees.Name, Salary, Date\_Hired FROM Employees WHERE DATEDIFF(tglSekarang, Date\_Hired) > 365;  END //  DELIMITER ;  CALL TambahanBonus(); |

1. **Penjelasan**

Kode di atas digunakan untuk membuat sebuah prosedur yang akan menambah gaji karyawan yang bekerja lebih dari satu tahun sebanyak 10%. Kode di atas DECLARE untuk mengatur variabel tglSekarang menjadi tanggal hari ini (CURDATE()). Selanjutnya yaitu perintah untuk mengupdate tabel Employees kolom salary dimana salary awalnya dikali 1.10 dimana dilakukan pengurangan antara tanggal sekarang dan date hirednya lebih dari 365 hari. Untuk menampilkan datanya menggunakan syntax CALL nama\_prosedur.

### **Stored Procedure Perpanjang Project yang Akan Berakhir dalam Kurang dari Satu Bulan**

1. **Source Code**

|  |
| --- |
| -- Nomor 2  DELIMITER //  CREATE PROCEDURE ProyekDiperpanjang()  BEGIN  DECLARE PerpanjangProyek DATE;  SET PerpanjangProyek = CURDATE();    SELECT  Project\_ID AS 'ID Project',  Project\_Name AS 'Nama Project',  End\_Date AS'Tanggal Project Berakhir',  DATE\_ADD(End\_Date, INTERVAL 3 MONTH) AS 'Tanggal Project Diperpanjang'  FROM Projects  WHERE DATEDIFF(End\_Date, PerpanjangProyek) < 30;  END //  DELIMITER ;  CALL ProyekDiperpanjang(); |

1. **Penjelasan**

Kode di atas digunakan untuk memperpanjang project yang akan berakhir dalam kurang dari satu bulan. Dalam SQL syntaxnya perlu dideklarasikan sebuah variabel menjadi tanggal saat ini (CURDATE()). Selanjutnya SELECT id project, nama project, tanggal berakhir ptoject, dari tabel projects. Kemudian menggunakan fungsi DATE\_ADD untuk menambahkan tanggal berakhir project dengan 3 bulan setelahnya jika selisih (DATEDIFF) dari tanggal berakhir dengan tanggal saat ini adalh 30 hari. Hasilnya akan menampilkan beberapa data yang tanggal berakhirnya akan ditambah 3 bulan ke depan.

### **Stored Procedure Memeriksa, Memperbarui, dan Menambah Pelatihan**

1. **Source Code**

|  |
| --- |
| -- Nomor 3  DELIMITER //  CREATE PROCEDURE UpdateSertifikat(  IN EmployeeID INT)  BEGIN  DECLARE BanyakSertifikat INT;  DECLARE HariIni DATE;  DECLARE IDSertifikatBaru INT;  DECLARE IDTrainingBaru INT;    SELECT COUNT(\*) INTO BanyakSertifikat FROM Certificates WHERE Employee\_ID = EmployeeID;  SET HariIni = CURDATE();  IF BanyakSertifikat > 0 THEN  UPDATE Certificates  SET Expiry\_Date = ADDDATE(Expiry\_Date, INTERVAL 1 YEAR)  WHERE Employee\_ID = EmployeeID AND Expiry\_Date < HariIni;  IF ROW\_COUNT() > 0 THEN  SELECT CONCAT('Sertif karyawan ', Employee\_ID, ' sudah diupdate.') AS Message;  INSERT INTO Trainings (Training\_ID, Certificate\_Name, Duration\_In\_Months) VALUES  (IDTrainingBaru, 'Sertifikat Pengembangan Website Menggunakan Laravel', '5');  INSERT INTO Certificates (Certificate\_ID, Employee\_ID, Certificate\_Name, Issue\_Date, Expiry\_Date) VALUES  (id\_sertif\_baruIDSertifikatBaru, EmployeeID, 'Pengembangan Website Menggunakan Laravel', '2024-05-05', '2024-05-20');  ELSE  SELECT CONCAT('Sertifikat karyawan ', EmployeeID, ' tidak ada yang perlu diperbarui.') AS Message;  END IF;  ELSE  SELECT CONCAT('karyawan ', EmployeeID, ' tidak mempunya sertifikat.') AS Message;  END IF;  SELECT IFNULL(MAX(Certificate\_ID), 0) + 1 INTO IDSertifikatBaru FROM Certificates;  SELECT \* FROM Trainings;  SELECT \* FROM Certificates;  END //  DELIMITER ;  CALL UpdateSertifikat(1); |

1. **Penjelasan**

Kode di atas digunakan untuk menambah data baru ke tabel trainings dan certificates. Kode akan mendeklarasikan empat variabel yang masing-masing diatur dan diset nilainya. Kemudian kode akan memeriksa apakah banyak sertifikat lebih dari 0? Jika iya maka kode akan mengupdate data tanggal pada tabel certificates. Kemudian kode di atur agar bisa menginputkan data baru pada tabel trainings dan certificates. Kemudian kode akan menentukan nilai IDSertifikatBaru yang akan digunakan untuk sertifikat baru yang akan dimasukkan ke dalam tabel. Selanjutnya akan menampilkan semua data padata tabel trainings dan certificates.

### **Stored Procedure Mengirim Notifikasi Pelatihan Baru Kepada Semua Karyawan**

1. **Source Code**

|  |
| --- |
| -- Nomor 4  DELIMITER //  CREATE PROCEDURE KirimNotifikasi()  BEGIN  DECLARE HariIni DATE;  DECLARE IdEmployee INT DEFAULT 0;  DECLARE SimpanID INT;  SET HariIni = CURDATE();  SELECT MAX(Employee\_ID) INTO SimpanID FROM Employees;  REPEAT  SET IdEmployee = IdEmployee + 1;  IF EXISTS (SELECT \* FROM Employees WHERE Employee\_ID = IdEmployee) THEN  INSERT INTO Notifications (Employee\_ID, Message, Date)  SELECT IdEmployee, 'Coming Soon! Pelatihan Pengembangan Aplikasi Web Menggunakan JavaScript dan React', HariIni;  END IF;  UNTIL IdEmployee >= SimpanID END REPEAT;    SELECT \* FROM Notifications;  END //  DELIMITER ;  CALL KirimNotifikasi(); |

1. **Penjelasan**

Kode di atas digunakan untuk mengirim notifikasi baru ke semua karyawan. Kode menggunakan tiga variabel untuk mengatur tanggal hari ini, id employee, dan untuk menyimpan nilai max dari id employee dari tabel employees. Kemudian terdapat perulangan REPEAT dimana id employee akan otomatis ditambah satu. Selanjutnya jika id employee yang diinputkan sesuai, maka tambah data pada tabel notifications yang terdiri dari id employee, message, dan tanggalnya. Kemudian akan terus berjalan selama nilai dari variabel IdEmployee masih kurang dari nilai dari variabel SimpanID.

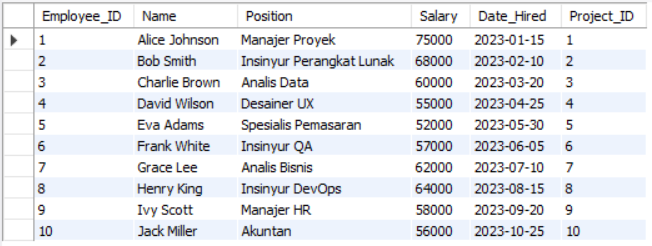
## **Hasil**

### **Data Setiap Tabel**

* **Tabel Projects**



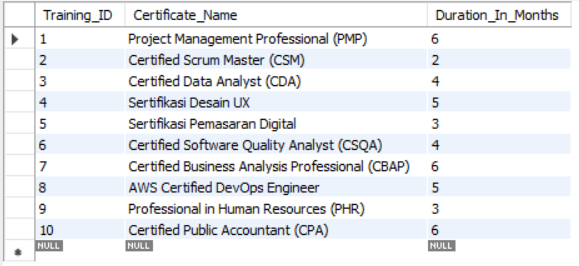
* **Tabel Employees**



* **Tabel Certificates**



* **Tabel Trainings**



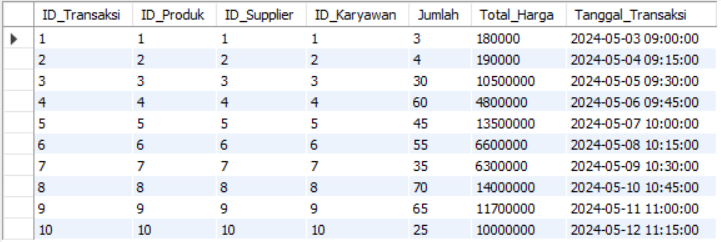
* **Tabel Notifications**



* **Tabel Companies**



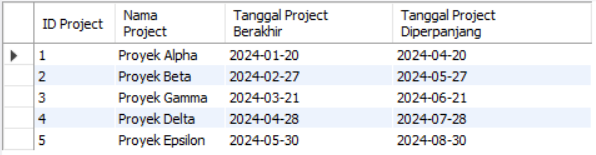
* **Tabel Companies**



### **Stored Procedure Menambah Salary Karyawan 10% yang Bekerja Lebih dari Satu Tahun**

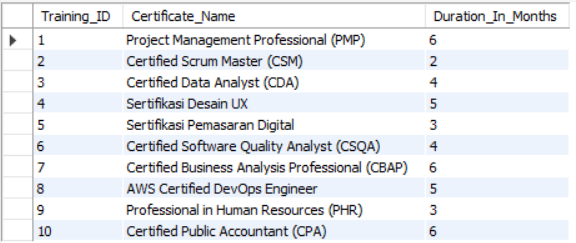


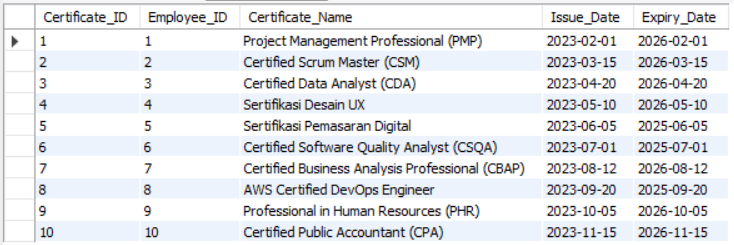
### **Stored Procedure Perpanjang Project yang Akan Berakhir dalam Kurang dari Satu Bulan**



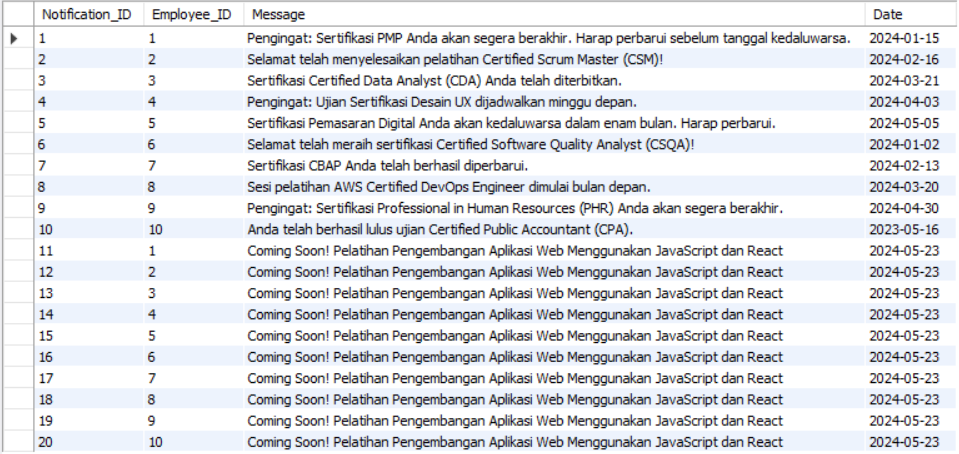
### **Stored Procedure Memeriksa, Memperbarui, dan Menambah Pelatihan**







### **Stored Procedure Mengirim Notifikasi Pelatihan Baru Kepada Semua Karyawan**



# **BAB V PENUTUP**

## **Analisa**

Dari hasil praktikum, praktikan menganalisa bahwa penggunaan stored procedure dengan kemampuan branching dan looping dalam basis data memberikan keuntungan signifikan dalam hal efisiensi dan kinerja sistem. Dengan menyimpan dan menjalankan logika bisnis langsung di server basis data, operasi kompleks dapat dieksekusi lebih cepat dibandingkan jika dilakukan melalui aplikasi klien. Branching memungkinkan sistem untuk membuat keputusan dinamis berdasarkan kondisi tertentu tanpa perlu mengirim data bolak-balik antara klien dan server. Looping memungkinkan pemrosesan batch data atau iterasi melalui dataset besar dengan cara yang efisien, mengurangi kebutuhan untuk menulis kode berulang di sisi klien.

Di sisi lain, stored procedure juga menawarkan peningkatan dalam hal keamanan dan integritas data. Dengan menjalankan logika bisnis di sisi server, data tidak perlu dikirimkan ke klien untuk diproses, yang mengurangi risiko kebocoran data. Penggunaan stored procedure memungkinkan kontrol akses yang lebih ketat, memastikan hanya pengguna dengan izin yang sesuai dapat menjalankan operasi tertentu. Dengan demikian, selain memberikan manfaat kinerja, stored procedure dengan branching dan looping juga berkontribusi pada peningkatan keamanan dan keandalan sistem basis data.

## **Kesimpulan**

* 1. Branching adalah penggunaan pernyataan kondisional (seperti IF...ELSE) untuk membuat keputusan berdasarkan kondisi tertentu.
  2. Looping adalah penggunaan perulangan (seperti WHILE atau FOR) untuk mengeksekusi serangkaian instruksi berulang kali hingga kondisi tertentu terpenuhi.
  3. DECLARE pada sebuah variabel yaitu pernyataan yang digunakan untuk mendefinisikan variabel baru yang dapat digunakan untuk menyimpan dan memanipulasi data selama eksekusi prosedur.Top of Form