

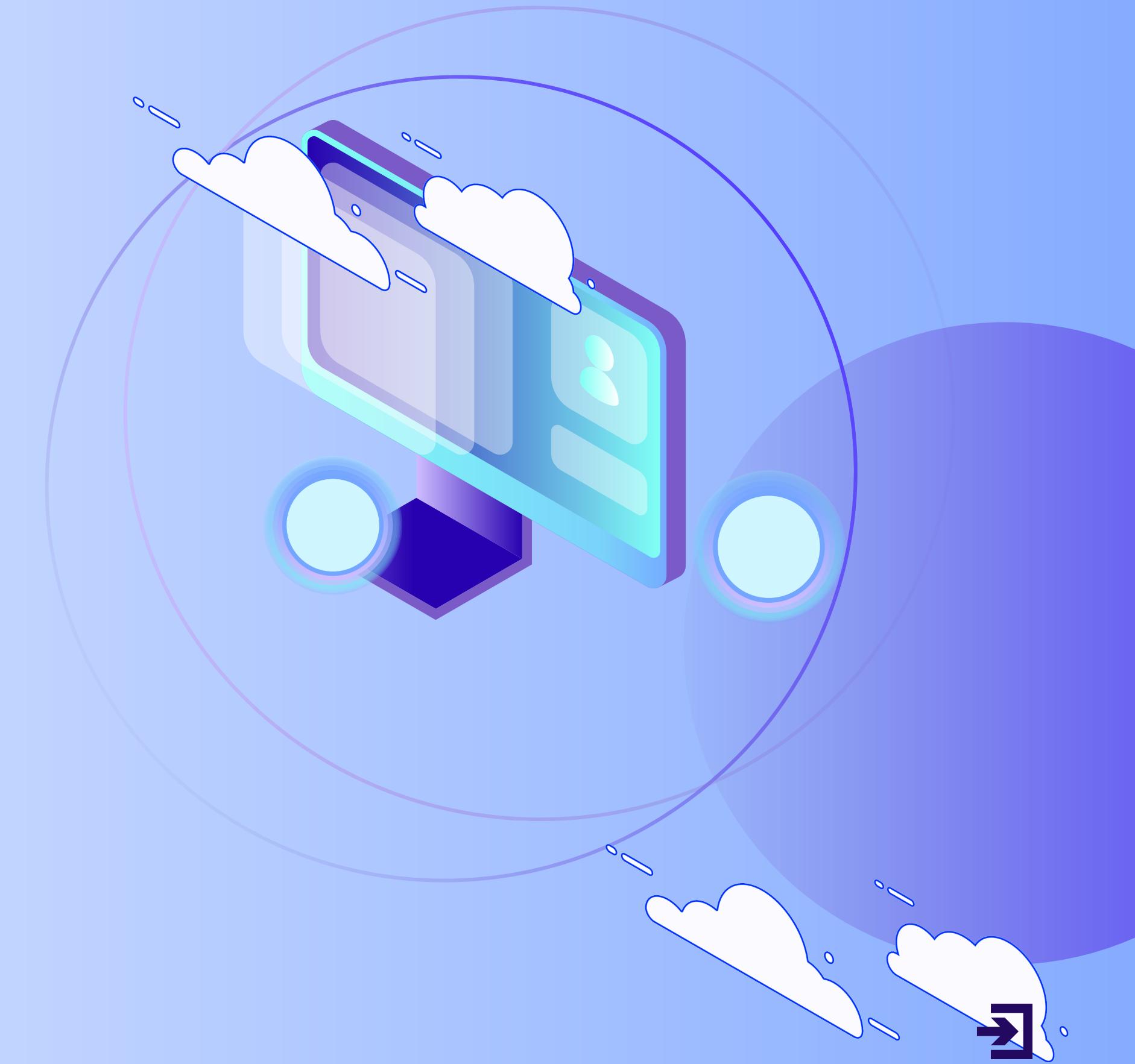


Kelompok 7 (INF-B)

BASIS DATA

Kelompok 7:

1. Syifa Nur Nabila (2023071017)
2. Dandy Tri Widianto (2023071011)

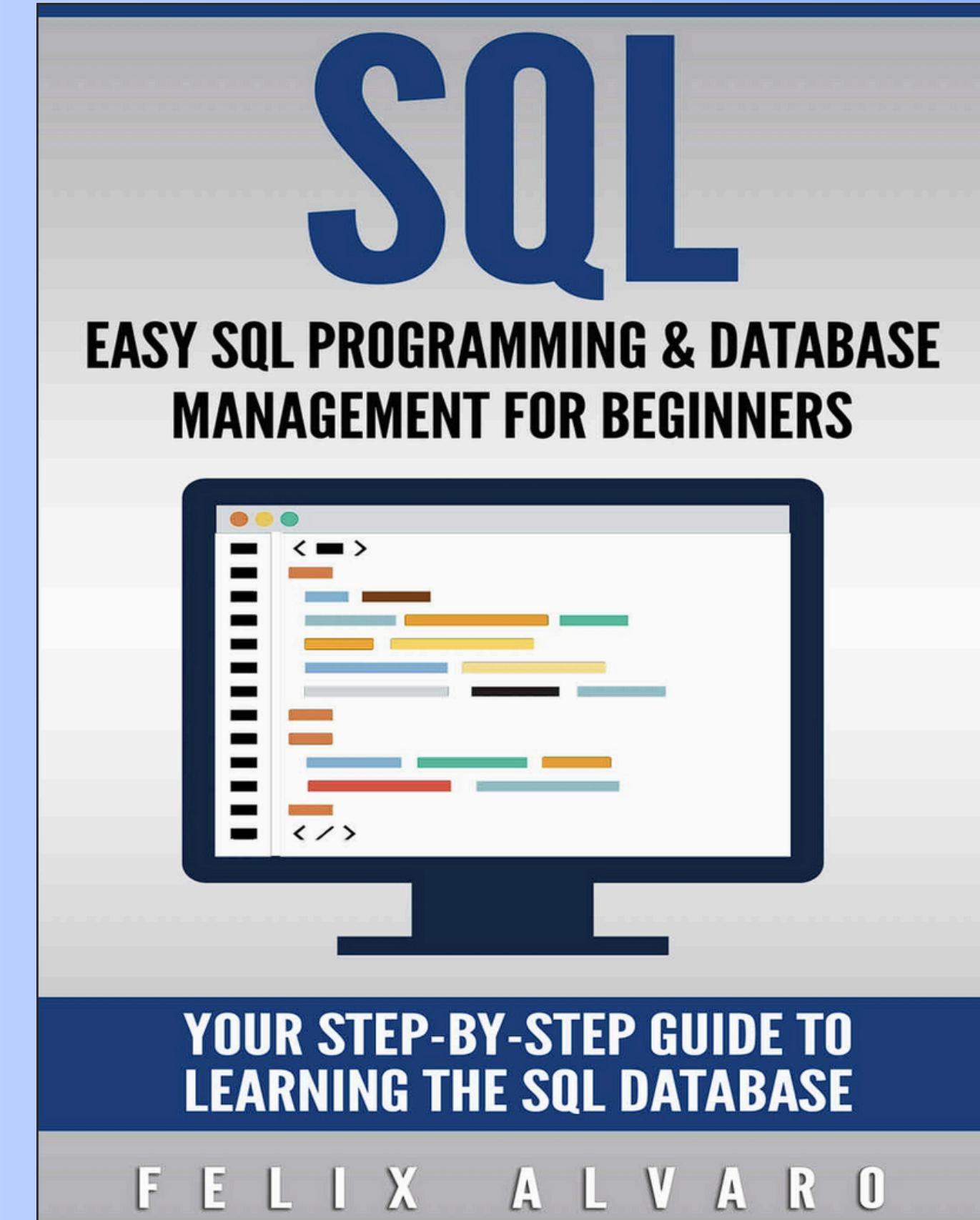


Dosen Pengampu: Riny Nurhajati, S.T., MTI. ×

DATA TYPES & USER-DEFINED DATA TYPE

Data Types & User-Defined Data Type

1. Baca ebook Easy SQL Programming & Database Management For Beginners. Your Step-By- Step Guide To Learning The SQL Database – Felix Alvaro : Halaman 30 – 36 tentang Data Types
2. Buatlah 1 tabel dimana kolom-kolomnya menggunakan seluruh Data Types yang telah dipelajari, beserta User-Defined Data Type



DATA TYPES & USER-DEFINED DATA TYPE

```
CREATE TABLE ContohTipeData (
    id INT PRIMARY KEY,
    nama VARCHAR(100),
    umur SMALLINT,
    tinggi_cm DECIMAL(5,2),
    berat_kg FLOAT,
    tanggal_lahir DATE,
    waktu_daftar TIME,
    terakhir_login TIMESTAMP,
    aktif BOOLEAN,
    bio TEXT,
    foto BLOB,
    preferences JSON,
    metadata TEXT,
    saldo_akun BIGINT,
    rating REAL,
    status_pesanan ENUM('Menunggu', 'Diproses', 'Dikirim', 'Selesai')
);

INSERT INTO ContohTipeData (id, nama, umur, tinggi_cm, berat_kg, tanggal_lahir, waktu_daftar, terakhir_login, aktif, bio, foto, preferences, metadata, saldo_akun, rating, status_pesanan)
VALUES
(1, 'John Doe', 30, 175.5, 70.3, '1994-05-15', '08:30:00', '2024-10-09 14:30:00', TRUE, 'Seorang pengembang web',
NULL, '{"tema": "gelap", "bahasa": "id"}', '<user><level>3</level></user>', 1000000, 4.7, 'Diproses');
```

User-Defined Data Type (UDDT) atau Tipe Data yang Didefinisikan Pengguna adalah fitur dalam beberapa sistem manajemen basis data (DBMS) yang memungkinkan pengguna untuk membuat tipe data baru berdasarkan tipe data yang sudah ada atau kombinasi dari tipe data yang ada.

PELAJARI DAN MEMBUAT SQL SYNTAX

Pelajari dan Membuat SQL Syntax

Data Query Language

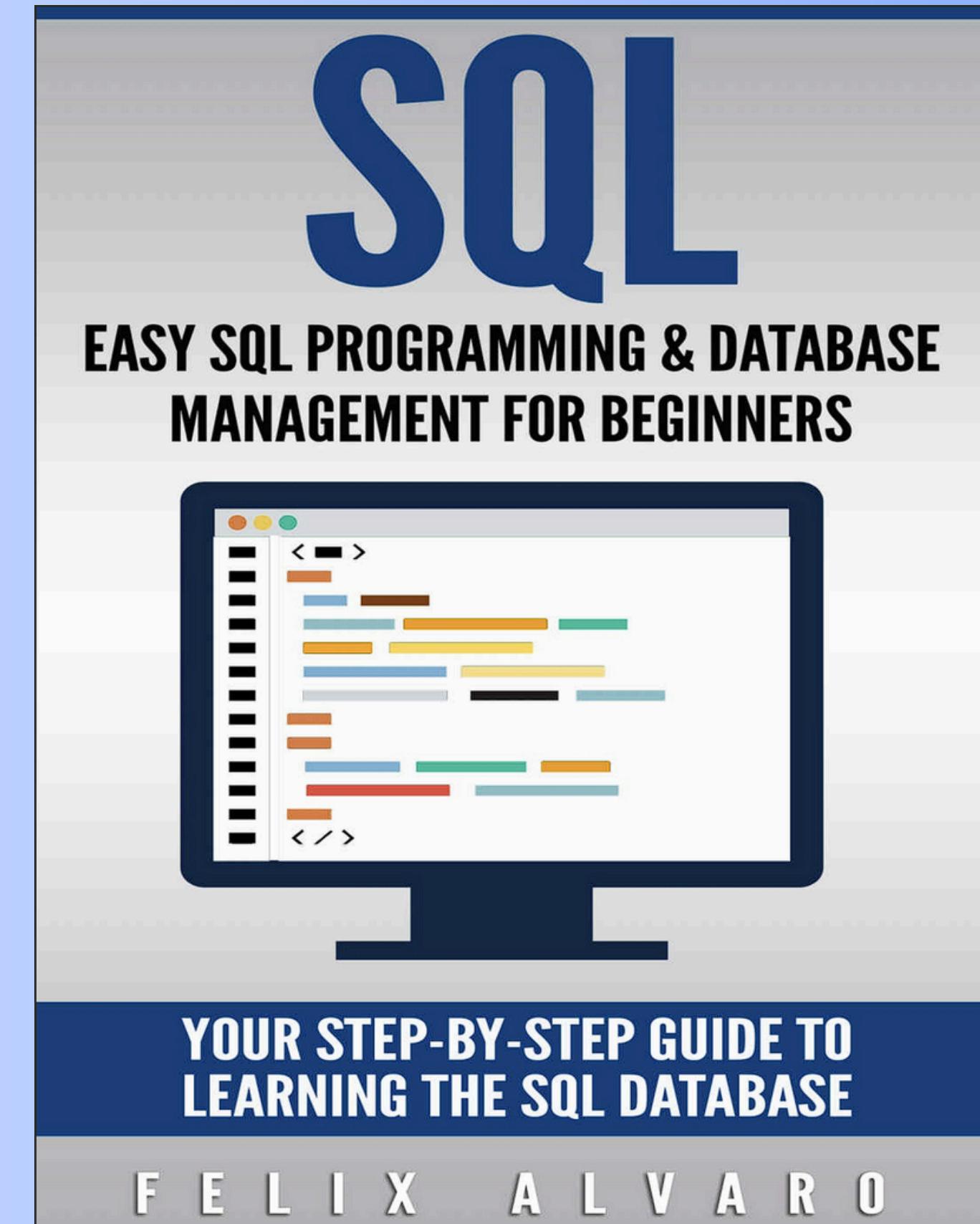
1. Perintah SELECT
2. Perintah WHERE
3. Perintah ORDER BY dan GROUP BY serta Keyword lainnya dalam Data Query Language

Mahasiswa dapat menerapkan perintah SELECT, WHERE, ORDER BY, GROUP BY pada basis data;

Waktu 30 menit pertama setelah penjelasan seluruh materi.

Baca ebook Easy SQL Programming & Database Management For Beginners. Your Step-By- Step Guide To Learning The SQL Database – Felix Alvaro : Halaman 70 – 80

Buat SQL Syntax dan simpan hasil latihan ke dalam file script simpan dan kumpulkan bersama dengan tugas kelompok di akhir sesi.



PELAJARI DAN MEMBUAT SQL SYNTAX

```
INSERT INTO ContohTipeData (id, nama, umur, tinggi_cm, berat_kg, tanggal_lahir, waktu_daftar, terakhir_login, aktif, bio, preferences, saldo_akun, rating, status_pesanan)
VALUES
(2, 'Jane Smith', 28, 165.0, 55.5, '1996-08-20', '10:15:00', '2024-10-08 09:45:00', TRUE, 'Desainer grafis', '{"tema": "terang", "bahasa": "en"}', 750000, 4.9, 'Menunggu'),
(3, 'Bob Johnson', 35, 180.5, 82.1, '1989-03-10', '14:30:00', '2024-10-07 18:20:00', FALSE, 'Manajer proyek', '{"tema": "gelap", "bahasa": "fr"}', 1500000, 4.2, 'Dikirim'),
(4, 'Alice Brown', 22, 170.2, 58.7, '2002-11-30', '09:00:00', '2024-10-09 11:10:00', TRUE, 'Mahasiswa', '{"tema": "terang", "bahasa": "id"}', 250000, 4.5, 'Diproses'),
(5, 'Charlie Davis', 40, 178.0, 75.3, '1984-06-05', '11:45:00', '2024-10-06 20:30:00', TRUE, 'Pengacara', '{"tema": "gelap", "bahasa": "en"}', 2000000, 4.8, 'Selesai');

-- 1. Perintah SELECT dasar
SELECT nama, umur, tinggi_cm FROM ContohTipeData;

-- 2. Perintah WHERE
SELECT nama, saldo_akun FROM ContohTipeData WHERE saldo_akun > 1000000;

-- 3. Perintah ORDER BY
SELECT nama, rating FROM ContohTipeData ORDER BY rating DESC;

-- 4. Perintah GROUP BY
SELECT status_pesanan, COUNT(*) as jumlah FROM ContohTipeData GROUP BY status_pesanan;
```

PELAJARI DAN MEMBUAT SQL SYNTAX

-- 5. Kombinasi WHERE, ORDER BY, dan GROUP BY

```
SELECT status_pesanan, AVG(rating) as rata_rata_rating
FROM ContohTipeData
WHERE aktif = TRUE
GROUP BY status_pesanan
ORDER BY rata_rata_rating DESC;
```

-- 6. Menggunakan HAVING (filter setelah GROUP BY)

```
SELECT status_pesanan, AVG(saldo_akun) as rata_rata_saldo
FROM ContohTipeData
GROUP BY status_pesanan
HAVING rata_rata_saldo > 500000;
```

-- 7. Menggunakan LIMIT

```
SELECT nama, umur FROM ContohTipeData ORDER BY umur DESC LIMIT 3;
```

-- 8. Menggunakan DISTINCT

```
SELECT DISTINCT status_pesanan FROM ContohTipeData;
```

-- 9. Menggunakan fungsi agregasi

```
SELECT
    MIN(umur) as umur_minimum,
    MAX(umur) as umur_maksimum,
    AVG(umur) as umur_rata_rata
FROM ContohTipeData;
```

-- 10. Menggunakan subquery

```
SELECT nama, umur
FROM ContohTipeData
WHERE umur > (SELECT AVG(umur) FROM ContohTipeData);
```

STUDI KASUS

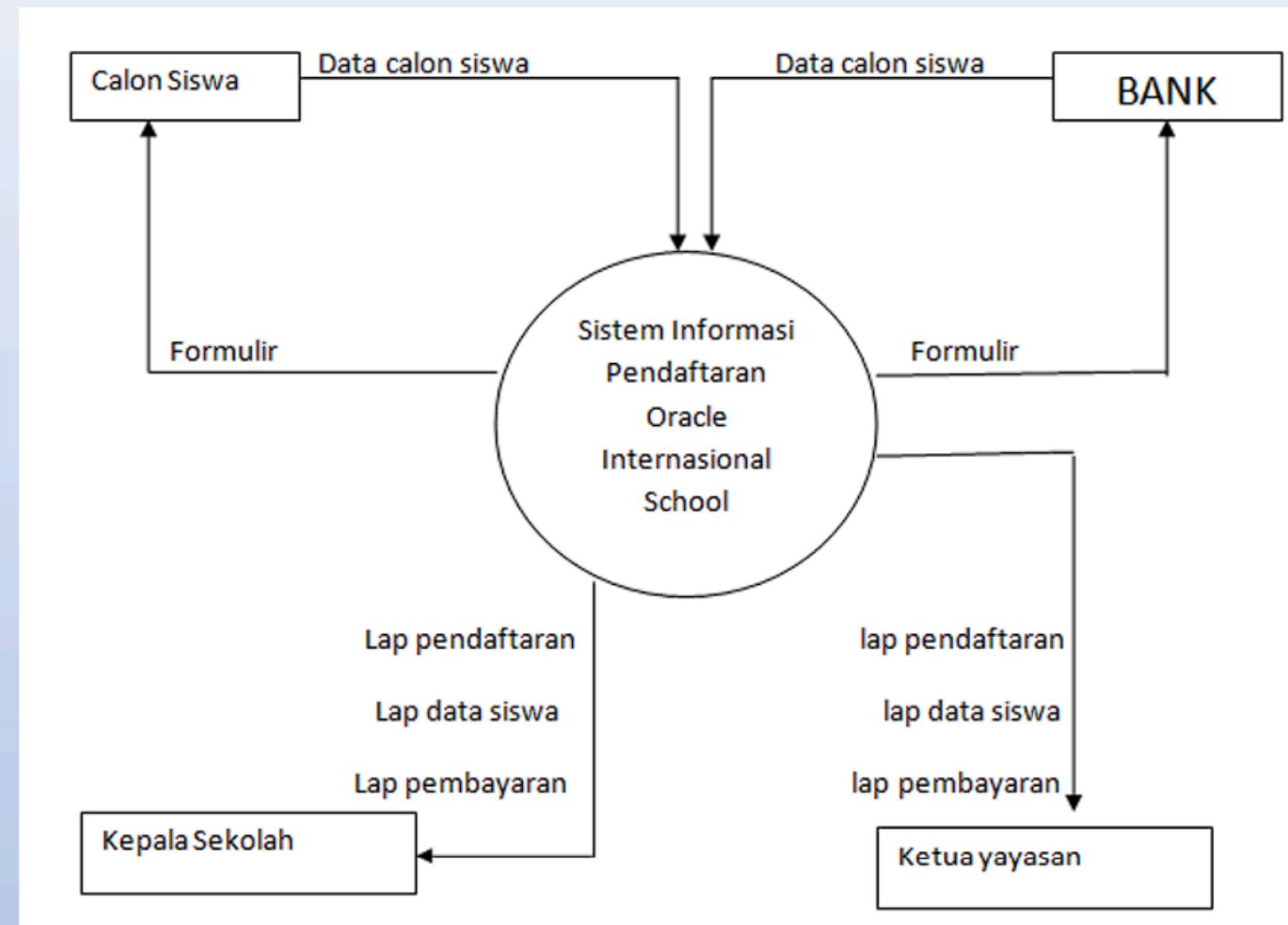
Presentasi Kelompok Pukul 14:30

Buat DDL, DQL, DML dari Contoh DFD

- Pelajari DFD Berikut
- Dari Contoh DFD yang ada maka akan ada banyak proses terkait DML terhadap database : kerjakan sesuai Rancangan Database dari masing-masing kelompok
- Buat list DDL, DQL, DML
- Buatkan masing-masing perintah SQL nya
- Pisahkan DDL dalam 1 script
- Dari DQL dan DML yang ada buatlah kedalam Store Procedure

Dari hasil implementasikan perancangan basis data dari tugas kelompok

1. Pelajari DFD dan buat list dibutuhkan DDL, DQL dan DML apa saja? Gunakan penerapan terhadap project Database dari solusi teknologi yang diajukan setiap kelompok. Maka ada proses apa saja? Ada layar inputan dan output apa saja yang dibutuhkan oleh sistem? Ada kebutuhan DDQI, DQL dan DML apa saja? Buatlah secara rinci. Untuk proses DQL, DML agar dibuatkan kedalam store procedure, sesuai parameter yang dikirimkan.
2. Buatlah perintah SQL terhadap list tersebut dan implementasikan
3. Kumpulkan seluruh query simpan didalam script lengkap dan dikumpulkan disini



- **Diagram nol / Context Diagram** adalah diagram yang menggambarkan proses dari dataflow diagram.
- Diagram nol memberikan pandangan secara menyeluruh mengenai sistem yang ditangani, menunjukkan tentang fungsi-fungsi utama atau proses yang ada, aliran data, dan eksternal entity.
- Pada level ini sudah dimungkinkan adanya/ digembarkannya data store yang digunakan.
- Keseimbangan input dan output (balancing) antara diagram 0 dengan diagram konteks harus terpeliha.

Penjelasan Context Diagram : Sistem Informasi Pendaftaran Oracle Internasional School

- Sistem Informasi menyediakan Formulir untuk pendaftaran
- Formulir Pendaftaran diisi oleh Siswa
- Data Calon Siswa telah diisi dan masuk ke dalam sistem
- Formulir menjadi acuan dalam pembayaran ke Bank
- Bank akan memberikan informasi status pembayaran pada Data Calon Siswa
- Secara periodik sistem akan membuat laporan Pendaftaran, Laporan Data Siswa dan laporan Pembayaran kepada Kepala Sekolah
- Secara periodik sistem akan membuat laporan Pendaftaran, Laporan Data Siswa dan laporan Pembayaran kepada Ketua Yayasan

DDL

```
-- Membuat database
CREATE DATABASE IF NOT EXISTS oracle_international_school;
USE oracle_international_school;

-- Membuat tabel Calon Siswa
CREATE TABLE calon_siswa (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100),
    tanggal_lahir DATE,
    alamat TEXT,
    nomor_telepon VARCHAR(20),
    email VARCHAR(100),
    status_pendaftaran ENUM('Mendaftar', 'Diterima', 'Ditolak') DEFAULT 'Mendaftar',
    status_pembayaran ENUM('Belum Bayar', 'Sudah Bayar') DEFAULT 'Belum Bayar'
);

-- Membuat tabel Formulir Pendaftaran
CREATE TABLE formulir_pendaftaran (
    id INT AUTO_INCREMENT PRIMARY KEY,
    calon_siswa_id INT,
    tanggal_pendaftaran DATE,
    program_yang_dipilih VARCHAR(50),
    FOREIGN KEY (calon_siswa_id) REFERENCES calon_siswa(id)
);
```

```
-- Membuat tabel Pembayaran
CREATE TABLE pembayaran (
    id INT AUTO_INCREMENT PRIMARY KEY,
    calon_siswa_id INT,
    jumlah DECIMAL(10, 2),
    tanggal_pembayaran DATE,
    metode_pembayaran VARCHAR(50),
    status ENUM('Sukses', 'Pending', 'Gagal') DEFAULT 'Pending',
    FOREIGN KEY (calon_siswa_id) REFERENCES calon_siswa(id)
);
```

DDL (Data Definition Language) dalam MySQL adalah kumpulan perintah yang digunakan untuk mendefinisikan atau mengubah struktur database, tabel, dan objek database lainnya.

DQL DAN DML

```
-- Contoh query untuk memasukkan data calon siswa
INSERT INTO calon_siswa (nama, tanggal_lahir, alamat, nomor_telepon, email)
VALUES ('John Doe', '2005-05-15', 'Jl. Contoh No. 123', '081234567890', 'john.doe@example.com');

-- Query untuk mengisi formulir pendaftaran
INSERT INTO formulir_pendaftaran (calon_siswa_id, tanggal_pendaftaran, program_yang_dipilih)
VALUES (1, CURDATE(), 'Program IB');

-- Query untuk mencatat pembayaran
INSERT INTO pembayaran (calon_siswa_id, jumlah, tanggal_pembayaran, metode_pembayaran, status)
VALUES (1, 5000000, CURDATE(), 'Transfer Bank', 'Sukses');

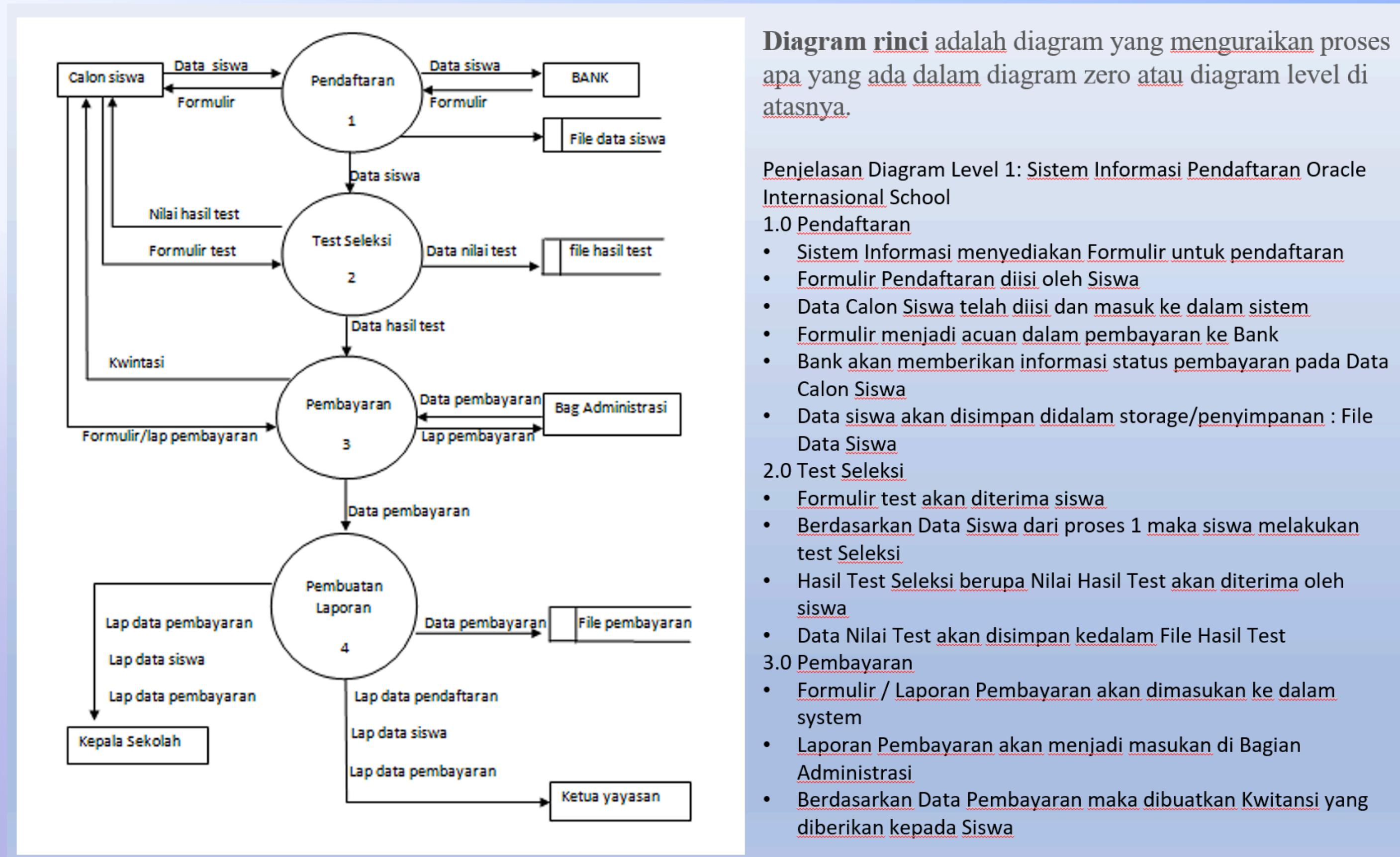
-- Query untuk mengupdate status pembayaran calon siswa
UPDATE calon_siswa
SET status_pembayaran = 'Sudah Bayar'
WHERE id = 1;

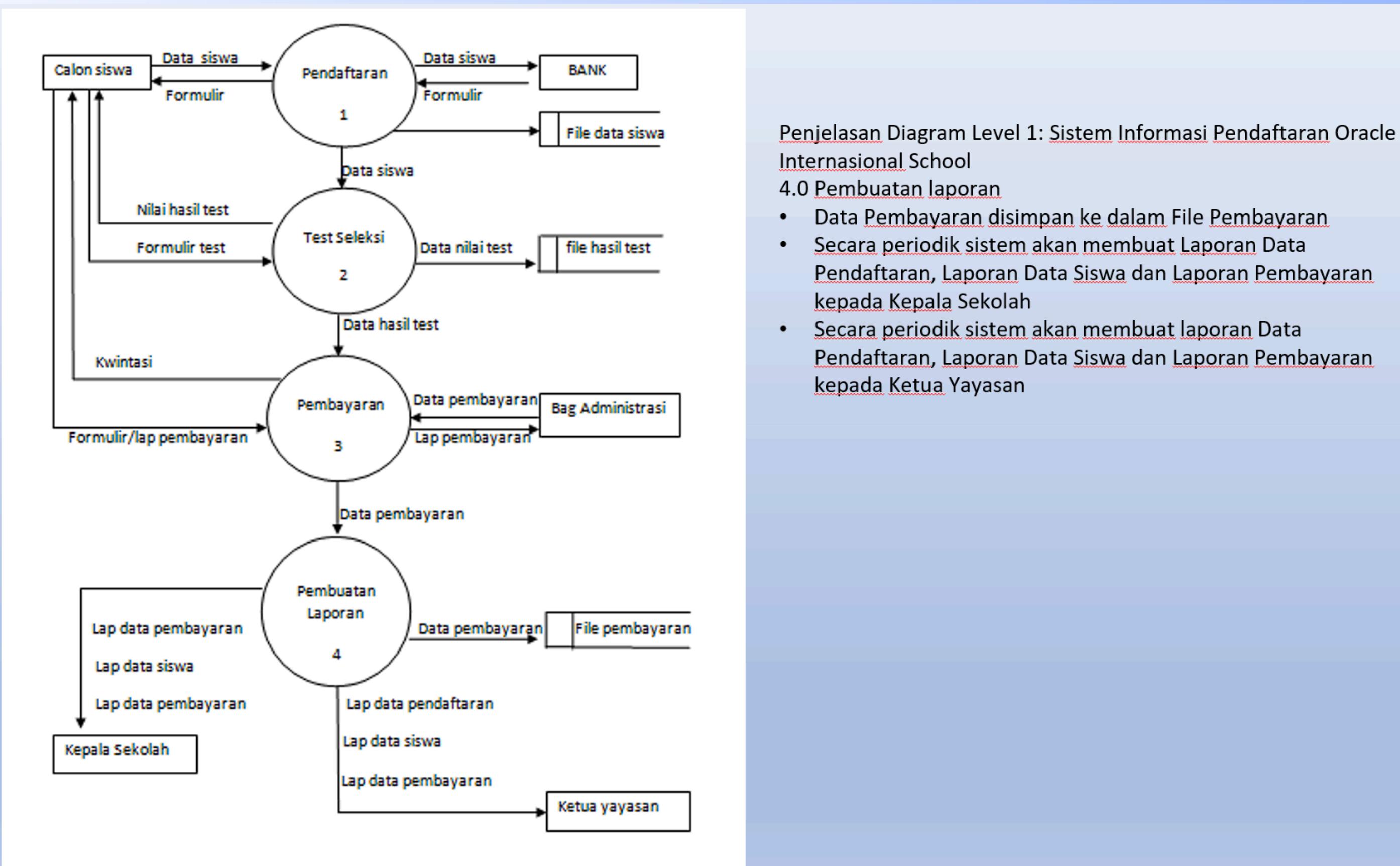
-- Query untuk laporan pendaftaran
SELECT cs.nama, fp.tanggal_pendaftaran, fp.program_yang_dipilih, cs.status_pendaftaran
FROM calon_siswa cs
JOIN formulir_pendaftaran fp ON cs.id = fp.calon_siswa_id;
```

DML (Data Manipulation Language) dalam MySQL adalah kumpulan perintah yang digunakan untuk mengelola dan memanipulasi data yang ada di dalam tabel database.

```
-- Query untuk laporan data siswa
SELECT cs.nama, cs.tanggal_lahir, cs.alamat, cs.nomor_telepon, cs.email, cs.status_pendaftaran, cs.status_pembayaran
FROM calon_siswa cs;

-- Query untuk laporan pembayaran
SELECT cs.nama, p.jumlah, p.tanggal_pembayaran, p.metode_pembayaran, p.status
FROM calon_siswa cs
JOIN pembayaran p ON cs.id = p.calon_siswa_id;
```





DDL

```
CREATE DATABASE oracleintschool;
USE oracleintschool;

CREATE TABLE calon_siswa (
    id_calon_siswa INT PRIMARY KEY,
    nama VARCHAR(100),
    tanggal_lahir DATE
);

CREATE TABLE data_calon_siswa (
    id_data_calon_siswa INT PRIMARY KEY,
    id_calon_siswa INT,
    nilai_ujian INT,
    prestasi VARCHAR(255),
    FOREIGN KEY (id_calon_siswa) REFERENCES calon_siswa(id_calon_siswa)
);

CREATE TABLE pendaftaran (
    id_pendaftaran INT PRIMARY KEY,
    id_calon_siswa INT,
    tanggal_pendaftaran DATE,
    status VARCHAR(50),
    FOREIGN KEY (id_calon_siswa) REFERENCES calon_siswa(id_calon_siswa)
);
```

```
CREATE TABLE test_seleksi (
    id_test INT PRIMARY KEY,
    id_pendaftaran INT,
    nilai_test INT,
    tanggal_test DATE,
    FOREIGN KEY (id_pendaftaran) REFERENCES pendaftaran(id_pendaftaran)
);

CREATE TABLE pembayaran (
    id_pembayaran INT PRIMARY KEY,
    id_pendaftaran INT,
    jumlah_bayar DECIMAL(10,2),
    tanggal_bayar DATE,
    FOREIGN KEY (id_pendaftaran) REFERENCES pendaftaran(id_pendaftaran)
);
```

DDL (Data Definition Language) dalam MySQL adalah kumpulan perintah yang digunakan untuk mendefinisikan atau mengubah struktur database, tabel, dan objek database lainnya.

DQL DAN DML

-- 1. INSERT: Menambahkan data ke tabel calon_siswa

```
INSERT INTO calon_siswa (id_calon_siswa, nama, tanggal_lahir)
VALUES (1, 'John Doe', '2005-05-15'),
       (2, 'Jane Smith', '2006-03-20');
```

-- 2. INSERT: Menambahkan data ke tabel data_calon_siswa

```
INSERT INTO data_calon_siswa (id_data_calon_siswa, id_calon_siswa, nilai_ujian, prestasi)
VALUES (1, 1, 85, 'Juara 1 Olimpiade Matematika'),
       (2, 2, 90, 'Juara 2 Lomba Karya Ilmiah');
```

-- 3. INSERT: Menambahkan data ke tabel pendaftaran

```
INSERT INTO pendaftaran (id_pendaftaran, id_calon_siswa, tanggal_pendaftaran, status)
VALUES (1, 1, '2024-01-15', 'Diterima'),
       (2, 2, '2024-01-16', 'Dalam Proses');
```

-- 4. INSERT: Menambahkan data ke tabel test_seleksi

```
INSERT INTO test_seleksi (id_test, id_pendaftaran, nilai_test, tanggal_test)
VALUES (1, 1, 88, '2024-02-01'),
       (2, 2, 92, '2024-02-01');
```

-- 5. INSERT: Menambahkan data ke tabel pembayaran

```
INSERT INTO pembayaran (id_pembayaran, id_pendaftaran, jumlah_bayar, tanggal_bayar)
VALUES (1, 1, 5000000.00, '2024-02-15'),
       (2, 2, 5000000.00, '2024-02-16');
```

-- 6. UPDATE: Mengubah status pendaftaran

```
UPDATE pendaftaran
SET status = 'Diterima'
WHERE id_pendaftaran = 2;
```

-- 7. UPDATE: Mengubah nilai ujian calon siswa

```
UPDATE data_calon_siswa
SET nilai_ujian = 95
WHERE id_calon_siswa = 2;
```

-- 8. DELETE: Menghapus data pembayaran (contoh saja, hati-hati dalam penggunaan DELETE)

```
DELETE FROM pembayaran
WHERE id_pembayaran = 2;
```

-- DQL (Data Query Language)

-- 9. SELECT: Menampilkan semua data calon siswa

```
SELECT * FROM calon_siswa;
```

-- 10. SELECT dengan JOIN: Menampilkan data calon siswa beserta nilai ujian dan prestasi

```
SELECT cs.id_calon_siswa, cs.nama, cs.tanggal_lahir, dcs.nilai_ujian, dcs.prestasi
FROM calon_siswa cs
JOIN data_calon_siswa dcs ON cs.id_calon_siswa = dcs.id_calon_siswa;
```

-- 11. SELECT dengan WHERE: Menampilkan pendaftaran dengan status 'Diterima'

```
SELECT * FROM pendaftaran WHERE status = 'Diterima';
```

-- 12. SELECT dengan JOIN dan WHERE: Menampilkan data test seleksi untuk pendaftaran yang diterima

```
SELECT ts.id_test, cs.nama, ts.nilai_test, ts.tanggal_test
FROM test_seleksi ts
JOIN pendaftaran p ON ts.id_pendaftaran = p.id_pendaftaran
JOIN calon_siswa cs ON p.id_calon_siswa = cs.id_calon_siswa
WHERE p.status = 'Diterima';
```

-- 13. SELECT dengan GROUP BY dan agregasi: Menghitung jumlah pendaftaran per status

```
SELECT status, COUNT(*) as jumlah_pendaftaran
FROM pendaftaran
GROUP BY status;
```

DQL DAN DML

```
-- 14. SELECT dengan subquery: Menampilkan calon siswa dengan nilai test di atas rata-rata
SELECT cs.nama, ts.nilai_test
FROM calon_siswa cs
JOIN pendaftaran p ON cs.id_calon_siswa = p.id_calon_siswa
JOIN test_seleksi ts ON p.id_pendaftaran = ts.id_pendaftaran
WHERE ts.nilai_test > (SELECT AVG(nilai_test) FROM test_seleksi);

-- 15. SELECT dengan ORDER BY: Menampilkan pembayaran diurutkan berdasarkan tanggal
SELECT p.id_pembayaran, cs.nama, p.jumlah_bayar, p.tanggal_bayar
FROM pembayaran p
JOIN pendaftaran pn ON p.id_pendaftaran = pn.id_pendaftaran
JOIN calon_siswa cs ON pn.id_calon_siswa = cs.id_calon_siswa
ORDER BY p.tanggal_bayar DESC;
```

DML (Data Manipulation Language) dalam MySQL adalah kumpulan perintah yang digunakan untuk mengelola dan memanipulasi data yang ada di dalam tabel database.

STORED PROCEDURE

```
DELIMITER //

-- 1. Stored Procedure untuk menambahkan calon siswa baru
CREATE PROCEDURE sp_tambah_calon_siswa(
    IN p_id_calon_siswa INT,
    IN p_nama VARCHAR(100),
    IN p_tanggal_lahir DATE
)
BEGIN
    INSERT INTO calon_siswa (id_calon_siswa, nama, tanggal_lahir)
    VALUES (p_id_calon_siswa, p_nama, p_tanggal_lahir);
END //

-- 2. Stored Procedure untuk mendaftarkan calon siswa
CREATE PROCEDURE sp_daftar_calon_siswa(
    IN p_id_pendaftaran INT,
    IN p_id_calon_siswa INT,
    IN p_tanggal_pendaftaran DATE,
    IN p_status VARCHAR(50)
)
BEGIN
    INSERT INTO pendaftaran (id_pendaftaran, id_calon_siswa, tanggal_pendaftaran, status)
    VALUES (p_id_pendaftaran, p_id_calon_siswa, p_tanggal_pendaftaran, p_status);
END //
```

```
-- 3. Stored Procedure untuk mencatat hasil test seleksi
CREATE PROCEDURE sp_catat_hasil_test(
    IN p_id_test INT,
    IN p_id_pendaftaran INT,
    IN p_nilai_test INT,
    IN p_tanggal_test DATE
)
BEGIN
    INSERT INTO test_seleksi (id_test, id_pendaftaran, nilai_test, tanggal_test)
    VALUES (p_id_test, p_id_pendaftaran, p_nilai_test, p_tanggal_test);
END //

-- 4. Stored Procedure untuk mencatat pembayaran
CREATE PROCEDURE sp_catat_pembayaran(
    IN p_id_pembayaran INT,
    IN p_id_pendaftaran INT,
    IN p_jumlah_bayar DECIMAL(10,2),
    IN p_tanggal_bayar DATE
)
BEGIN
    INSERT INTO pembayaran (id_pembayaran, id_pendaftaran, jumlah_bayar, tanggal_bayar)
    VALUES (p_id_pembayaran, p_id_pendaftaran, p_jumlah_bayar, p_tanggal_bayar);
END //

-- 5. Stored Procedure untuk mengupdate status pendaftaran
CREATE PROCEDURE sp_update_status_pendaftaran(
    IN p_id_pendaftaran INT,
    IN p_status VARCHAR(50)
)
BEGIN
    UPDATE pendaftaran
    SET status = p_status
    WHERE id_pendaftaran = p_id_pendaftaran;
END //
```

STORED PROCEDURE

```
-- 6. Stored Procedure untuk menampilkan data calon siswa beserta nilai ujian dan prestasi
CREATE PROCEDURE sp_tampil_data_calon_siswa()
BEGIN
    SELECT cs.id_calon_siswa, cs.nama, cs.tanggal_lahir, dcs.nilai_ujian, dcs.prestasi
    FROM calon_siswa cs
    JOIN data_calon_siswa dcs ON cs.id_calon_siswa = dcs.id_calon_siswa;
END //

-- 7. Stored Procedure untuk menampilkan data test seleksi untuk pendaftaran yang diterima
CREATE PROCEDURE sp_tampil_hasil_test_diterima()
BEGIN
    SELECT ts.id_test, cs.nama, ts.nilai_test, ts.tanggal_test
    FROM test_seleksi ts
    JOIN pendaftaran p ON ts.id_pendaftaran = p.id_pendaftaran
    JOIN calon_siswa cs ON p.id_calon_siswa = cs.id_calon_siswa
    WHERE p.status = 'Diterima';
END //

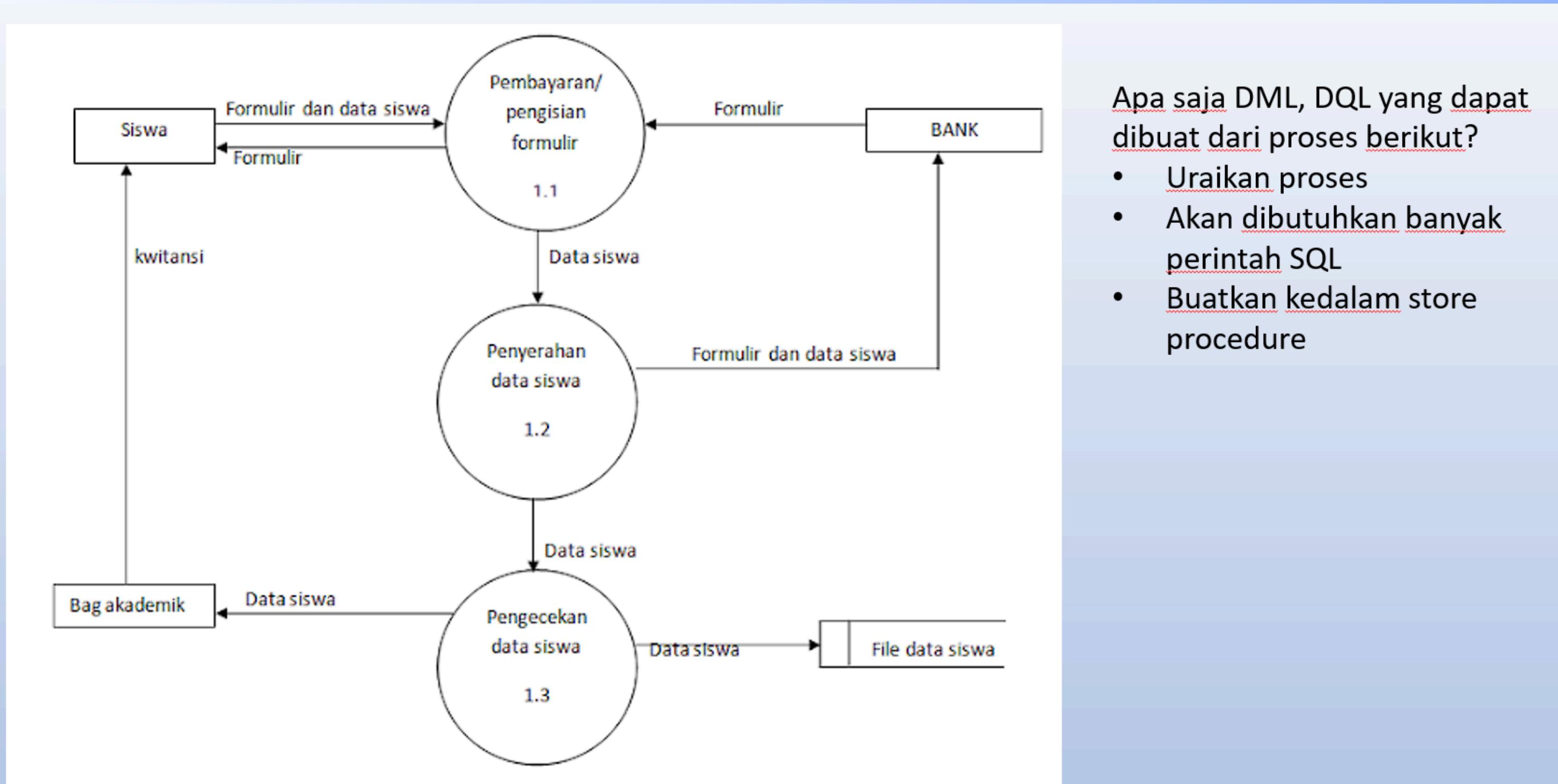
-- 8. Stored Procedure untuk menghitung jumlah pendaftaran per status
CREATE PROCEDURE sp_hitung_pendaftaran_per_status()
BEGIN
    SELECT status, COUNT(*) as jumlah_pendaftaran
    FROM pendaftaran
    GROUP BY status;
END //
```

```
-- 9. Stored Procedure untuk menampilkan calon siswa dengan nilai test di atas rata-rata
CREATE PROCEDURE sp_calon_siswa_nilai_diatas_rata_rata()
BEGIN
    SELECT cs.nama, ts.nilai_test
    FROM calon_siswa cs
    JOIN pendaftaran p ON cs.id_calon_siswa = p.id_calon_siswa
    JOIN test_seleksi ts ON p.id_pendaftaran = ts.id_pendaftaran
    WHERE ts.nilai_test > (SELECT AVG(nilai_test) FROM test_seleksi);
END //

DELIMITER ;

-- Contoh penggunaan Stored Procedure:
CALL sp_tambah_calon_siswa(3, 'Alice Johnson', '2006-08-10');
CALL sp_daftar_calon_siswa(3, 3, '2024-01-17', 'Dalam Proses');
CALL sp_catat_hasil_test(3, 3, 87, '2024-02-02');
CALL sp_catat_pembayaran(3, 3, 5000000.00, '2024-02-17');
CALL sp_update_status_pendaftaran(3, 'Diterima');
CALL sp_tampil_data_calon_siswa();
CALL sp_tampil_hasil_test_diterima();
CALL sp_hitung_pendaftaran_per_status();
CALL sp_calon_siswa_nilai_diatas_rata_rata();
```

Stored Procedure dalam MySQL adalah sekumpulan perintah SQL yang disimpan di dalam database dan dapat dijalankan berulang kali. Ini seperti fungsi yang dapat dipanggil untuk melakukan tugas tertentu di database, seperti menambah, memperbarui, atau mengambil data. Stored procedure memudahkan pengelolaan tugas-tugas berulang dan meningkatkan efisiensi.



Apa saja DML, DQL yang dapat dibuat dari proses berikut?

- Uraikan proses
- Akan dibutuhkan banyak perintah SQL
- Buatkan kedalam store procedure

DFD Level 2 : 1.0 Pendaftaran

DDL

```
-- Membuat database
CREATE DATABASE IF NOT EXISTS sistem_pendaftaran_siswa;
USE sistem_pendaftaran_siswa;

-- Membuat tabel-tabel
CREATE TABLE siswa (
    id_siswa INT PRIMARY KEY AUTO_INCREMENT,
    nama VARCHAR(100),
    alamat TEXT,
    nomor_telepon VARCHAR(15)
);

CREATE TABLE formulir (
    id_formulir INT PRIMARY KEY AUTO_INCREMENT,
    id_siswa INT,
    tanggal_pengisian DATE,
    status_pengisian ENUM('Belum Lengkap', 'Lengkap'),
    FOREIGN KEY (id_siswa) REFERENCES siswa(id_siswa)
);

CREATE TABLE pembayaran (
    id_pembayaran INT PRIMARY KEY AUTO_INCREMENT,
    id_formulir INT,
    jumlah DECIMAL(10, 2),
    tanggal_pembayaran DATE,
    status_pembayaran ENUM('Belum Dibayar', 'Sudah Dibayar'),
    FOREIGN KEY (id_formulir) REFERENCES formulir(id_formulir)
);
```

```
CREATE TABLE data_siswa (
    id_data_siswa INT PRIMARY KEY AUTO_INCREMENT,
    id_siswa INT,
    nilai_ujian DECIMAL(5, 2),
    status_verifikasi ENUM('Belum Diverifikasi', 'Sudah Diverifikasi'),
    FOREIGN KEY (id_siswa) REFERENCES siswa(id_siswa)
);
```

DDL (Data Definition Language) dalam MySQL adalah kumpulan perintah yang digunakan untuk mendefinisikan atau mengubah struktur database, tabel, dan objek database lainnya.

STORED PROCEDURE

```
-- Stored Procedures

DELIMITER //

-- 1. Prosedur untuk pembayaran dan pengisian formulir
CREATE PROCEDURE sp_pembayaran_dan_pengisian_formulir(
    IN p_nama VARCHAR(100),
    IN p_alamat TEXT,
    IN p_nomor_telepon VARCHAR(15),
    IN p_jumlah_pembayaran DECIMAL(10, 2)
)
BEGIN
    DECLARE v_id_siswa INT;
    DECLARE v_id_formulir INT;

    -- Insert data siswa
    INSERT INTO siswa (nama, alamat, nomor_telepon) VALUES (p_nama, p_alamat, p_nomor_telepon);
    SET v_id_siswa = LAST_INSERT_ID();

    -- Insert formulir
    INSERT INTO formulir (id_siswa, tanggal_pengisian, status_pengisian)
    VALUES (v_id_siswa, CURDATE(), 'Belum Lengkap');
    SET v_id_formulir = LAST_INSERT_ID();

    -- Insert pembayaran
    INSERT INTO pembayaran (id_formulir, jumlah, tanggal_pembayaran, status_pembayaran)
    VALUES (v_id_formulir, p_jumlah_pembayaran, CURDATE(), 'Sudah Dibayar');

    -- Generate kwitansi (dalam bentuk output)
    SELECT CONCAT('Kwitansi untuk ', p_nama, ' sejumlah ', p_jumlah_pembayaran) AS kwitansi;
END //
```

```
-- 2. Prosedur untuk penyerahan data siswa
CREATE PROCEDURE sp_penyerahan_data_siswa(
    IN p_id_siswa INT,
    IN p_nilai_ujian DECIMAL(5, 2)
)
BEGIN
    INSERT INTO data_siswa (id_siswa, nilai_ujian, status_verifikasi)
    VALUES (p_id_siswa, p_nilai_ujian, 'Belum Diverifikasi');

    UPDATE formulir SET status_pengisian = 'Lengkap' WHERE id_siswa = p_id_siswa;
END //

-- 3. Prosedur untuk pengecekan data siswa
CREATE PROCEDURE sp_pengecekan_data_siswa(IN p_id_siswa INT)
BEGIN
    UPDATE data_siswa SET status_verifikasi = 'Sudah Diverifikasi' WHERE id_siswa = p_id_siswa;

    -- Mengirim data ke Bag Akademik (simulasi dengan SELECT)
    SELECT * FROM data_siswa WHERE id_siswa = p_id_siswa;

    -- Menyimpan ke file data siswa (simulasi dengan SELECT)
    SELECT * FROM siswa s
    JOIN data_siswa ds ON s.id_siswa = ds.id_siswa
    WHERE s.id_siswa = p_id_siswa;
END //
```

STORED PROCEDURE

```
-- 4. Prosedur untuk mendapatkan laporan pendaftaran
CREATE PROCEDURE sp_laporan_pendaftaran()
BEGIN
    SELECT s.nama, f.tanggal_pengisian, f.status_pengisian, p.status_pembayaran, ds.nilai_ujian, ds.status_verifikasi
    FROM siswa s
    JOIN formulir f ON s.id_siswa = f.id_siswa
    LEFT JOIN pembayaran p ON f.id_formulir = p.id_formulir
    LEFT JOIN data_siswa ds ON s.id_siswa = ds.id_siswa;
END //

DELIMITER ;

-- Contoh penggunaan stored procedures:
CALL sp_pembayaran_dan_pengisian_formulir('John Doe', 'Jl. Contoh No. 123', '08123456789', 1000000.00);
CALL sp_penyerahan_data_siswa(1, 85.5);
CALL sp_pengecekan_data_siswa(1);
CALL sp_laporan_pendaftaran();
```

Stored Procedure dalam MySQL adalah sekumpulan perintah SQL yang disimpan di dalam database dan dapat dijalankan berulang kali. Ini seperti fungsi yang dapat dipanggil untuk melakukan tugas tertentu di database, seperti menambah, memperbarui, atau mengambil data. Stored procedure memudahkan pengelolaan tugas-tugas berulang dan meningkatkan efisiensi.

