Nama : Firza Ridwan Hafidz

NIM : 2023071014

**TUGAS PRAKTIKUM (NON COURSERA)**

**Praktikum 1**

**1. Mengisi tabel jurusan dan biodata**

sql

-- Membuat tabel jurusan

CREATE TABLE jurusan (

kode\_jurusan VARCHAR(5) PRIMARY KEY,

nama\_jurusan VARCHAR(50),

ketua\_jurusan VARCHAR(50)

);

-- Mengisi tabel jurusan

INSERT INTO jurusan (kode\_jurusan, nama\_jurusan, ketua\_jurusan)

VALUES

('KD01', 'Sistem Informasi', 'Harnaningrum, S.Si'),

('KD02', 'Teknik Informatika', 'Enny Sela, S.Kom., M.Kom'),

('KD03', 'Teknik Komputer', 'Berta Bednar, S.Si., M.T');

-- Membuat tabel biodata

CREATE TABLE biodata (

no\_mahasiswa VARCHAR(10) PRIMARY KEY,

kode\_jurusan VARCHAR(5),

nama\_mahasiswa VARCHAR(50),

alamat VARCHAR(50),

ipk FLOAT,

FOREIGN KEY (kode\_jurusan) REFERENCES jurusan(kode\_jurusan)

);

-- Mengisi tabel biodata

INSERT INTO biodata (no\_mahasiswa, kode\_jurusan, nama\_mahasiswa, alamat, ipk)

VALUES

('210089', 'KD01', 'Rina Gunawan', 'Denpasar', 3.0),

('210090', 'KD03', 'Gani Suprapto', 'Singaraja', 3.5),

('210012', 'KD02', 'Alexandra', 'Nusa Dua', 3.0),

('210099', 'KD02', 'Nadine', 'Gianyar', 3.2),

('210002', 'KD01', 'Rizal Samurai', 'Denpasar', 3.7);

**2. Menambahkan data baru pada tabel biodata dengan kode jurusan KD04**

sql

-- Menambahkan data baru

INSERT INTO biodata (no\_mahasiswa, kode\_jurusan, nama\_mahasiswa, alamat, ipk)

VALUES ('210100', 'KD04', 'Eka Putri', 'Badung', 3.4);

**3. Menggunakan perintah UPDATE**

sql

-- Mengganti nama mahasiswa "Rina Gunawan" menjadi "Rina Gunawan Astuti"

UPDATE biodata

SET nama\_mahasiswa = 'Rina Gunawan Astuti'

WHERE nama\_mahasiswa = 'Rina Gunawan';

-- Mengganti kode jurusan pada tabel jurusan "KD01" menjadi "KM01"

UPDATE jurusan

SET kode\_jurusan = 'KM01'

WHERE kode\_jurusan = 'KD01';

-- Mengganti no mahasiswa "210089" menjadi "210098"

UPDATE biodata

SET no\_mahasiswa = '210098'

WHERE no\_mahasiswa = '210089';

-- Mengganti nilai IPK "3" menjadi "3.3"

UPDATE biodata

SET ipk = 3.3

WHERE ipk = 3.0;

-- Mengganti kode jurusan pada tabel biodata "KD03" menjadi "KD05"

UPDATE biodata

SET kode\_jurusan = 'KD05'

WHERE kode\_jurusan = 'KD03';

**4. Kesimpulan**

* **Menambah Data:** Penambahan data pada tabel dilakukan menggunakan perintah INSERT INTO.
* **Mengubah Data:** Pengubahan data dilakukan dengan menggunakan perintah UPDATE dengan kriteria tertentu pada klausa WHERE.

**Praktikum 2**

**a. Cartesian Products**

sql

-- Menggunakan CROSS JOIN

SELECT barang.id\_barang, pembelian.id\_pembeli, pembelian.nama\_pembeli, barang.nama\_barang

FROM barang

CROSS JOIN pembelian;

-- Menggunakan sintaks lain (tanpa CROSS JOIN)

SELECT barang.id\_barang, pembelian.id\_pembeli, pembelian.nama\_pembeli, barang.nama\_barang

FROM barang, pembelian;

**b. INNER JOIN**

sql

-- INNER JOIN antara tabel barang dan pembelian

SELECT barang.id\_barang, pembelian.id\_pembeli, pembelian.nama\_pembeli, barang.nama\_barang

FROM barang

INNER JOIN pembelian

ON barang.id\_barang = pembelian.id\_barang;

**c. LEFT JOIN**

sql

-- LEFT JOIN antara tabel barang dan pembelian

SELECT barang.id\_barang, pembelian.id\_pembeli, pembelian.nama\_pembeli, barang.nama\_barang

FROM barang

LEFT JOIN pembelian

ON barang.id\_barang = pembelian.id\_barang;

**d. RIGHT JOIN**

sql

-- RIGHT JOIN antara tabel barang dan pembelian

SELECT barang.id\_barang, pembelian.id\_pembeli, pembelian.nama\_pembeli, barang.nama\_barang

FROM barang

RIGHT JOIN pembelian

ON barang.id\_barang = pembelian.id\_barang;

**Tugas 1: Cobalah masing-masing perintah JOIN di atas dengan menggunakan database Mahasiswa yang Anda buat!**

**Contoh query:**

sql

-- Cartesian Product

SELECT biodata.no\_mahasiswa, jurusan.kode\_jurusan, jurusan.nama\_jurusan

FROM biodata, jurusan;

-- INNER JOIN

SELECT biodata.no\_mahasiswa, biodata.nama\_mahasiswa, jurusan.nama\_jurusan

FROM biodata

INNER JOIN jurusan

ON biodata.kode\_jurusan = jurusan.kode\_jurusan;

-- LEFT JOIN

SELECT biodata.no\_mahasiswa, biodata.nama\_mahasiswa, jurusan.nama\_jurusan

FROM biodata

LEFT JOIN jurusan

ON biodata.kode\_jurusan = jurusan.kode\_jurusan;

-- RIGHT JOIN

SELECT biodata.no\_mahasiswa, biodata.nama\_mahasiswa, jurusan.nama\_jurusan

FROM biodata

RIGHT JOIN jurusan

ON biodata.kode\_jurusan = jurusan.kode\_jurusan;

**Tugas 2: Kesimpulan perbedaan antara RIGHT JOIN dan LEFT JOIN**

* **LEFT JOIN:** Menampilkan semua data dari tabel sebelah kiri (kiri JOIN), meskipun tidak ada data yang cocok di tabel sebelah kanan. Nilai NULL akan ditampilkan untuk kolom tabel kanan yang tidak cocok.
* **RIGHT JOIN:** Menampilkan semua data dari tabel sebelah kanan (kanan JOIN), meskipun tidak ada data yang cocok di tabel sebelah kiri. Nilai NULL akan ditampilkan untuk kolom tabel kiri yang tidak cocok

**Praktikum 3**

**Single Row Functions**

1. **LOWER**: Mengubah nama mahasiswa menjadi huruf kecil semua.

sql

SELECT LOWER(nama\_mahasiswa) AS huruf\_kecil

FROM biodata;

1. **UPPER**: Mengubah nama mahasiswa menjadi huruf besar semua.

sql

SELECT UPPER(nama\_mahasiswa) AS huruf\_besar

FROM biodata;

1. **SUBSTRING**: Mengambil nama mahasiswa mulai dari huruf kedua sebanyak 5 huruf.

sql

SELECT SUBSTRING(nama\_mahasiswa, 2, 5) AS ambil\_karakter

FROM biodata;

1. **LTRIM**: Menghilangkan spasi di sebelah kiri pada nama mahasiswa.

sql

SELECT LTRIM(' Rina Gunawan') AS hapus\_spasi;

1. **RTRIM**: Menghilangkan spasi di sebelah kanan pada nama mahasiswa.

sql

SELECT RTRIM('Rina Gunawan ') AS hapus\_spasi;

1. **RIGHT**: Mengambil 3 karakter terakhir dari nama mahasiswa.

sql

SELECT RIGHT(nama\_mahasiswa, 3) AS ambil\_karakter

FROM biodata;

1. **LEFT**: Mengambil 3 karakter pertama dari nama mahasiswa.

sql

SELECT LEFT(nama\_mahasiswa, 3) AS ambil\_karakter

FROM biodata;

1. **LENGTH**: Menghitung panjang nama mahasiswa termasuk spasi.

sql

SELECT nama\_mahasiswa, LENGTH(nama\_mahasiswa) AS panjang

FROM biodata;

1. **REVERSE**: Membalik nama mahasiswa.

sql

SELECT nama\_mahasiswa, REVERSE(nama\_mahasiswa) AS balik

FROM biodata;

1. **SPACE**: Memberikan 40 spasi kosong.

sql

SELECT SPACE(40) AS spasi;

**Tugas 1: Buatlah perintah SQL yang menggunakan single row function (masing-masing 1)!**

Berikut adalah contoh penggunaan:

1. Menggunakan LOWER:

sql

SELECT LOWER(ketua\_jurusan) AS huruf\_kecil

FROM jurusan;

1. Menggunakan UPPER:

sql

SELECT UPPER(ketua\_jurusan) AS huruf\_besar

FROM jurusan;

1. Menggunakan SUBSTRING:

sql

SELECT SUBSTRING(nama\_mahasiswa, 1, 4) AS ambil\_karakter

FROM biodata;

1. Menggunakan LENGTH:

sql

SELECT nama\_mahasiswa, LENGTH(nama\_mahasiswa) AS panjang

FROM biodata;

**Praktikum 4**

**ORDER BY**

1. **Urutan ASC (menaik):** Menampilkan data mahasiswa berdasarkan nama secara alfabetik.

sql

SELECT \* FROM biodata

ORDER BY nama\_mahasiswa ASC;

1. **Urutan DESC (menurun):** Menampilkan data mahasiswa berdasarkan nama secara alfabetik terbalik.

sql

SELECT \* FROM biodata

ORDER BY nama\_mahasiswa DESC;

**Aggregate Functions**

1. **AVG:** Menghitung rata-rata IPK mahasiswa.

sql

SELECT AVG(ipk) AS nilai\_rata

FROM biodata;

1. **COUNT:** Menghitung jumlah data pada tabel biodata.

sql

SELECT COUNT(\*) AS jumlah\_data

FROM biodata;

1. **SUM:** Menjumlahkan semua nilai IPK mahasiswa.

sql

SELECT SUM(ipk) AS total\_ipk

FROM biodata;

1. **MAX:** Menampilkan IPK tertinggi.

sql

SELECT MAX(ipk) AS max\_ipk

FROM biodata;

1. **MIN:** Menampilkan IPK terendah.

sql

SELECT MIN(ipk) AS min\_ipk

FROM biodata;

**GROUP BY dan HAVING**

1. **GROUP BY:** Mengelompokkan mahasiswa berdasarkan kode jurusan dan menghitung jumlah mahasiswa di setiap jurusan.

sql

SELECT kode\_jurusan, COUNT(\*) AS jumlah\_mahasiswa

FROM biodata

GROUP BY kode\_jurusan;

1. **HAVING:** Menampilkan jurusan dengan rata-rata IPK mahasiswa lebih dari 3.2.

sql

SELECT kode\_jurusan, AVG(ipk) AS rata\_rata\_ipk

FROM biodata

GROUP BY kode\_jurusan

HAVING AVG(ipk) > 3.2;

**Tugas**

1. **Perintah SQL menggunakan Aggregate Functions (masing-masing 1):**
   * **COUNT:** Menghitung jumlah mahasiswa di jurusan tertentu.

sql

SELECT COUNT(\*) AS jumlah\_mahasiswa

FROM biodata

WHERE kode\_jurusan = 'KD02';

* + **MAX:** Menampilkan IPK tertinggi dari mahasiswa yang tinggal di "Denpasar".

sql

SELECT MAX(ipk) AS max\_ipk

FROM biodata

WHERE alamat = 'Denpasar';

1. **Perintah SQL dengan klausa ORDER BY, GROUP BY, dan HAVING:**
   * **ORDER BY:** Mengurutkan mahasiswa berdasarkan IPK secara menurun.

sql

SELECT \* FROM biodata

ORDER BY ipk DESC;

* + **GROUP BY:** Mengelompokkan mahasiswa berdasarkan alamat dan menghitung jumlah mahasiswa di setiap alamat.

sql

SELECT alamat, COUNT(\*) AS jumlah\_mahasiswa

FROM biodata

GROUP BY alamat;

* + **HAVING:** Menampilkan alamat dengan jumlah mahasiswa lebih dari 1.

sql

SELECT alamat, COUNT(\*) AS jumlah\_mahasiswa

FROM biodata

GROUP BY alamat

HAVING COUNT(\*) > 1;

**Praktikum 5**

**Subqueries**

1. **Menampilkan data pembelian dengan jumlah pembelian terbanyak:**

sql

SELECT id\_barang, tanggal\_beli, nama\_pembeli, jumlah\_pembelian

FROM pembelian

WHERE jumlah\_pembelian = (SELECT MAX(jumlah\_pembelian) FROM pembelian);

1. **Menampilkan data pembelian dengan jumlah pembelian paling sedikit:**

sql

SELECT \*

FROM pembelian

WHERE jumlah\_pembelian = (SELECT MIN(jumlah\_pembelian) FROM pembelian);

1. **Menampilkan Nama\_Barang, Tanggal\_terima, dan Stok\_Barang pada tabel barang dengan jumlah pembelian lebih dari satu:**

sql

SELECT nama\_barang, tanggal\_terima, stok\_barang

FROM barang

WHERE id\_barang IN (

SELECT id\_barang

FROM pembelian

WHERE jumlah\_pembelian > 1

);

**UNION dan UNION ALL**

1. **Menampilkan Nama\_Pembeli dan Jumlah\_Pembelian pada tabel pembelian serta Nama\_Barang dan Stok\_Barang pada tabel barang:**

sql

SELECT nama\_pembeli, jumlah\_pembelian

FROM pembelian

UNION

SELECT nama\_barang, stok\_barang

FROM barang;

1. **Menampilkan Nama\_Pembeli dan Jumlah\_Pembelian dengan UNION ALL:**

sql

SELECT nama\_pembeli, jumlah\_pembelian

FROM pembelian

UNION ALL

SELECT nama\_barang, stok\_barang

FROM barang;

1. **Menampilkan data pembelian dengan jumlah pembelian = 1 dan Nama\_Pembeli mengandung huruf "ma":**

sql

SELECT tanggal\_beli, nama\_pembeli, jumlah\_pembelian

FROM pembelian

WHERE jumlah\_pembelian = 1

UNION ALL

SELECT tanggal\_beli, nama\_pembeli, jumlah\_pembelian

FROM pembelian

WHERE nama\_pembeli LIKE '%ma%';

**Tugas**

1. **Menggunakan subqueries dan set operation pada database mahasiswa:**
   * **Subquery:** Menampilkan mahasiswa dengan IPK tertinggi.

sql

SELECT \*

FROM biodata

WHERE ipk = (SELECT MAX(ipk) FROM biodata);

* + **Set Operation:** Menampilkan data mahasiswa dengan IPK > 3.5 atau tinggal di "Denpasar".

sql

SELECT no\_mahasiswa, nama\_mahasiswa, ipk

FROM biodata

WHERE ipk > 3.5

UNION

SELECT no\_mahasiswa, nama\_mahasiswa, ipk

FROM biodata

WHERE alamat = 'Denpasar';

1. **Menghasilkan INTERSECT dan EXCEPT dengan UNION:**
   * **INTERSECT:** Mahasiswa yang tinggal di "Denpasar" dan memiliki IPK > 3.5.

sql

SELECT no\_mahasiswa, nama\_mahasiswa

FROM biodata

WHERE alamat = 'Denpasar'

INTERSECT

SELECT no\_mahasiswa, nama\_mahasiswa

FROM biodata

WHERE ipk > 3.5;

* + **EXCEPT:** Mahasiswa di "Denpasar" yang IPK-nya bukan 3.5.

sql

SELECT no\_mahasiswa, nama\_mahasiswa

FROM biodata

WHERE alamat = 'Denpasar'

EXCEPT

SELECT no\_mahasiswa, nama\_mahasiswa

FROM biodata

WHERE ipk = 3.5;

1. **Kesimpulan:**
   * **INTERSECT:** Menampilkan data yang ada di kedua query.
   * **EXCEPT:** Menampilkan data yang ada di query pertama tetapi tidak ada di query kedua.

**Praktikum 6**

**View**

1. **Membuat View pada database "Mahasiswa":**

sql

CREATE VIEW laporan\_mahasiswa AS

SELECT biodata.no\_mahasiswa, biodata.nama\_mahasiswa, jurusan.nama\_jurusan, biodata.ipk

FROM biodata

INNER JOIN jurusan

ON biodata.kode\_jurusan = jurusan.kode\_jurusan;

1. **Melihat isi View:**

sql

SELECT \* FROM laporan\_mahasiswa;

1. **Menghapus View:**

sql

DROP VIEW laporan\_mahasiswa;

**Control Flow Function**

1. **Membuat Control Flow Function menggunakan CASE:**

sql

SELECT ipk,

CASE

WHEN ipk >= 3.5 THEN 'Cumlaude'

WHEN ipk >= 3.0 THEN 'Baik'

ELSE 'Cukup'

END AS kategori\_ipk

FROM biodata;

1. **Menggunakan IF:**

sql

SELECT IF(ipk > 3.5, 'IPK Tinggi', 'IPK Normal') AS status\_ipk

FROM biodata;

1. **Menggunakan IFNULL:**

sql

SELECT nama\_mahasiswa, IFNULL(alamat, 'Alamat Tidak Tersedia') AS alamat\_mahasiswa

FROM biodata;

**Temporary Table**

1. **Membuat Tabel Temporary:**

sql

CREATE TEMPORARY TABLE temp\_mahasiswa (

no\_mahasiswa VARCHAR(10),

nama\_mahasiswa VARCHAR(50),

ipk FLOAT

);

1. **Mengisi Tabel Temporary:**

sql

INSERT INTO temp\_mahasiswa (no\_mahasiswa, nama\_mahasiswa, ipk)

SELECT no\_mahasiswa, nama\_mahasiswa, ipk

FROM biodata;

1. **Melihat isi Tabel Temporary:**

sql

SELECT \* FROM temp\_mahasiswa;

1. **Menghapus Tabel Temporary (otomatis dihapus saat sesi berakhir):** Tidak diperlukan perintah manual; tabel temporary akan otomatis dihapus saat koneksi database ditutup.

**Tugas**

1. **Fungsi View:**
   * Mempermudah pengelolaan query kompleks dengan menyederhanakannya menjadi satu entitas.
   * Meningkatkan keamanan data dengan menyembunyikan detail tabel asli.
2. **View pada database "Mahasiswa":**

sql

CREATE VIEW ipk\_mahasiswa AS

SELECT no\_mahasiswa, nama\_mahasiswa, ipk

FROM biodata

WHERE ipk > 3.0;

1. **Control Flow Function dan Temporary Table:**
   * **Control Flow Function:**

sql

SELECT ipk,

CASE

WHEN ipk >= 3.5 THEN 'Cumlaude'

ELSE 'Non-Cumlaude'

END AS status\_ipk

FROM biodata;

* + **Temporary Table:**

sql

CREATE TEMPORARY TABLE temp\_biodata AS

SELECT \* FROM biodata;

SELECT \* FROM temp\_biodata;

**Praktikum 7**

**User-Defined Functions**

1. **Membuat fungsi untuk menjumlahkan dua angka:**

sql

DELIMITER //

CREATE FUNCTION tambah(angka1 INT, angka2 INT)

RETURNS INT

BEGIN

RETURN angka1 + angka2;

END; //

DELIMITER ;

1. **Memanggil fungsi:**

sql

SELECT tambah(10, 20) AS hasil\_penjumlahan;

**Stored Procedures**

1. **Stored Procedure untuk menampilkan semua data mahasiswa berdasarkan ketua jurusan:**

sql

DELIMITER //

CREATE PROCEDURE tampilkan\_mahasiswa()

BEGIN

SELECT biodata.no\_mahasiswa, biodata.nama\_mahasiswa, jurusan.nama\_jurusan

FROM biodata

INNER JOIN jurusan

ON biodata.kode\_jurusan = jurusan.kode\_jurusan;

END; //

DELIMITER ;

1. **Memanggil Stored Procedure:**

sql

CALL tampilkan\_mahasiswa();

1. **Stored Procedure untuk mencari ketua jurusan berdasarkan nama tertentu:**

sql

DELIMITER //

CREATE PROCEDURE cari\_ketua\_jurusan(nama\_jurusan\_input VARCHAR(50))

BEGIN

SELECT ketua\_jurusan

FROM jurusan

WHERE nama\_jurusan = nama\_jurusan\_input;

END; //

DELIMITER ;

1. **Memanggil Stored Procedure:**

sql

CALL cari\_ketua\_jurusan('Teknik Informatika');

**Trigger**

1. **Membuat Trigger untuk mengupdate IPK mahasiswa otomatis jika ada perubahan nilai:**

sql

DELIMITER //

CREATE TRIGGER update\_ipk

AFTER UPDATE ON biodata

FOR EACH ROW

BEGIN

UPDATE biodata

SET ipk = NEW.ipk

WHERE no\_mahasiswa = NEW.no\_mahasiswa;

END; //

DELIMITER ;

1. **Memasukkan data yang memicu trigger:**

sql

UPDATE biodata

SET ipk = 3.8

WHERE no\_mahasiswa = '210002';

**Tugas**

1. **Stored Procedure untuk menampilkan semua data mahasiswa berdasarkan ketua jurusan:**

sql

DELIMITER //

CREATE PROCEDURE tampilkan\_mahasiswa\_by\_ketua(ketua VARCHAR(50))

BEGIN

SELECT biodata.no\_mahasiswa, biodata.nama\_mahasiswa, jurusan.nama\_jurusan

FROM biodata

INNER JOIN jurusan

ON biodata.kode\_jurusan = jurusan.kode\_jurusan

WHERE jurusan.ketua\_jurusan = ketua;

END; //

DELIMITER ;

1. **Stored Procedure untuk mencari nama ketua jurusan tertentu:**

sql

DELIMITER //

CREATE PROCEDURE cari\_ketua(nama\_jurusan\_input VARCHAR(50))

BEGIN

SELECT ketua\_jurusan

FROM jurusan

WHERE nama\_jurusan = nama\_jurusan\_input;

END; //

DELIMITER ;

1. **Trigger untuk database "Mahasiswa":**

sql

DELIMITER //

CREATE TRIGGER auto\_update\_ipk

BEFORE UPDATE ON biodata

FOR EACH ROW

BEGIN

IF NEW.ipk > 4 THEN

SET NEW.ipk = 4;

END IF;

END; //

DELIMITER ;

1. **Perbedaan antara Stored Procedure dan Fungsi:**
   * **Stored Procedure:** Tidak wajib mengembalikan nilai; digunakan untuk operasi lebih kompleks.
   * **Fungsi:** Wajib mengembalikan nilai; sering digunakan dalam query SELECT.

**Praktikum 8**

**Menciptakan Query**

1. **Menciptakan Query SQL menggunakan editor:**

sql

SELECT \*

FROM biodata

WHERE ipk > 3.0;

**Menciptakan dan Menghapus Tabel**

1. **Membuat tabel jurusan:**

Sql

CREATE TABLE jurusan (

kd\_jurusan INT PRIMARY KEY,

nama VARCHAR(50)

);

1. **Menghapus tabel jurusan:**

sql

DROP TABLE jurusan;

**Penambahan Data**

1. **Menambahkan data ke tabel biodata:**

sql

INSERT INTO biodata (no\_mahasiswa, kode\_jurusan, nama\_mahasiswa, alamat, ipk)

VALUES ('210200', 'KD02', 'Tania Putri', 'Jakarta', 3.6);

**Modifikasi Data**

1. **Mengubah data di tabel biodata:**

sql

UPDATE biodata

SET ipk = 3.9

WHERE no\_mahasiswa = '210200';

**Seleksi Data**

1. **Menampilkan semua data dari tabel biodata:**

sql

SELECT \* FROM biodata;

1. **Menampilkan mahasiswa dengan IPK lebih besar dari 3.5:**

sql

SELECT \*

FROM biodata

WHERE ipk > 3.5;

1. **Menampilkan mahasiswa berdasarkan IPK secara menurun:**

sql

SELECT \*

FROM biodata

ORDER BY ipk DESC;

**Seleksi dengan Kriteria**

1. **Menampilkan mahasiswa dengan alamat mengandung kata "Jakarta":**

sql

SELECT \*

FROM biodata

WHERE alamat LIKE '%Jakarta%';

1. **Menampilkan mahasiswa dengan alamat tidak "Denpasar":**

sql

SELECT \*

FROM biodata

WHERE alamat <> 'Denpasar';

**Pengurutan Data**

1. **Mengurutkan data mahasiswa berdasarkan nama secara menaik:**

sql

SELECT \*

FROM biodata

ORDER BY nama\_mahasiswa ASC;

1. **Mengurutkan data mahasiswa berdasarkan nama secara menurun:**

sql

SELECT \*

FROM biodata

ORDER BY nama\_mahasiswa DESC;

**Tugas**

1. **Query SQL untuk menciptakan tabel mahasiswa\_nim:**

sql

CREATE TABLE mahasiswa\_210089 (

nim VARCHAR(10) PRIMARY KEY,

nama VARCHAR(50),

jenis\_kelamin VARCHAR(10),

tempat\_lahir VARCHAR(50),

tanggal\_lahir DATE,

alamat VARCHAR(100)

);

1. **Mengisi data ke tabel mahasiswa\_nim:**

sql

INSERT INTO mahasiswa\_210089 (nim, nama, jenis\_kelamin, tempat\_lahir, tanggal\_lahir, alamat)

VALUES ('210089', 'Rina Gunawan', 'Perempuan', 'Denpasar', '2000-01-01', 'Denpasar');

1. **Menampilkan data mahasiswa dengan tahun angkatan = 16 (NIM 2 digit pertama):**

sql

SELECT \*

FROM mahasiswa\_210089

WHERE LEFT(nim, 2) = '16'

ORDER BY nim ASC;

1. **Menampilkan data mahasiswa dengan kode fakultas = 05 dan jenis kelamin laki-laki:**

sql

SELECT \*

FROM mahasiswa\_210089

WHERE SUBSTRING(nim, 3, 2) = '05' AND jenis\_kelamin = 'Laki-laki'

ORDER BY nim DESC;

1. **Menampilkan data mahasiswa dengan kode tahun angkatan = 14 dan alamat bukan "Jalan Mangga":**

sql

SELECT nim, nama, jenis\_kelamin, alamat

FROM mahasiswa\_210089

WHERE LEFT(nim, 2) = '14' AND alamat <> 'Jalan Mangga';

1. **Menampilkan data mahasiswa beserta umur:**

sql

SELECT nim, nama, tanggal\_lahir,

TIMESTAMPDIFF(YEAR, tanggal\_lahir, CURDATE()) AS umur

FROM mahasiswa\_210089;

**Praktikum 9**

**1. Menciptakan Tabel Mahasiswa2 dan Mengisi Data**

1. **Membuat tabel mahasiswa2:**

sql

CREATE TABLE mahasiswa2 (

nim VARCHAR(10) PRIMARY KEY,

nama VARCHAR(50),

jenis\_kelamin VARCHAR(10),

alamat VARCHAR(100)

);

1. **Menambahkan 10 data pada tabel mahasiswa2:**

sql

INSERT INTO mahasiswa2 (nim, nama, jenis\_kelamin, alamat)

VALUES

('210001', 'Ani', 'Perempuan', 'Jakarta'),

('210002', 'Budi', 'Laki-laki', 'Bandung'),

('210003', 'Citra', 'Perempuan', 'Surabaya'),

('210004', 'Dedi', 'Laki-laki', 'Yogyakarta'),

('210005', 'Eka', 'Perempuan', 'Medan'),

('210006', 'Fandi', 'Laki-laki', 'Makassar'),

('210007', 'Gita', 'Perempuan', 'Denpasar'),

('210008', 'Hadi', 'Laki-laki', 'Semarang'),

('210009', 'Intan', 'Perempuan', 'Malang'),

('210010', 'Joko', 'Laki-laki', 'Bali');

**2. Mendapatkan Data Matakuliah dengan SKS 1 atau 2**

sql

SELECT kode\_mk, nama\_mk, sks, semester

FROM matakuliah

WHERE sks IN (1, 2);

**3. Mendapatkan Data Matakuliah dengan Nama Depan Tidak Sama dengan "Praktikum"**

sql

SELECT kode\_mk, nama\_mk, sks, semester

FROM matakuliah

WHERE nama\_mk NOT LIKE 'Praktikum%';

**Tugas Tambahan**

1. **Menampilkan data mahasiswa dengan kode tahun angkatan = 16 dan alamat bukan "Jakarta":**

sql

SELECT nim, nama, alamat

FROM mahasiswa2

WHERE LEFT(nim, 2) = '16' AND alamat <> 'Jakarta';

1. **Menghitung jumlah mahasiswa berdasarkan jenis kelamin:**

sql

SELECT jenis\_kelamin, COUNT(\*) AS jumlah

FROM mahasiswa2

GROUP BY jenis\_kelamin;

1. **Mengurutkan mahasiswa berdasarkan alamat secara menurun:**

sql

SELECT \*

FROM mahasiswa2

ORDER BY alamat DESC;

**Praktikum 10**

**1. Membuat Tabel mahasiswa\_nim**

sql

CREATE TABLE mahasiswa\_210089 (

nim VARCHAR(10) PRIMARY KEY,

nama VARCHAR(50),

jenis\_kelamin VARCHAR(10),

tempat\_lahir VARCHAR(50),

tanggal\_lahir DATE,

alamat VARCHAR(100)

);

**2. Mengisi Data ke Tabel mahasiswa\_nim**

sql

INSERT INTO mahasiswa\_210089 (nim, nama, jenis\_kelamin, tempat\_lahir, tanggal\_lahir, alamat)

VALUES

('210089', 'Rina Gunawan', 'Perempuan', 'Denpasar', '2000-01-01', 'Denpasar'),

('210090', 'Gani Suprapto', 'Laki-laki', 'Singaraja', '2000-05-15', 'Singaraja'),

('210091', 'Nadine', 'Perempuan', 'Gianyar', '1999-12-12', 'Gianyar');

**3. Menampilkan Data Mahasiswa dengan Kode Tahun Angkatan = 16**

sql

SELECT \*

FROM mahasiswa\_210089

WHERE LEFT(nim, 2) = '16'

ORDER BY nim ASC;

**4. Menampilkan Data Mahasiswa dengan Kode Fakultas = 05 dan Jenis Kelamin Laki-laki**

sql

SELECT \*

FROM mahasiswa\_210089

WHERE SUBSTRING(nim, 3, 2) = '05' AND jenis\_kelamin = 'Laki-laki'

ORDER BY nim DESC;

**5. Menampilkan Data Mahasiswa dengan Kode Tahun Angkatan = 14 dan Alamat Bukan "Jalan Mangga"**

sql

SELECT nim, nama, jenis\_kelamin, alamat

FROM mahasiswa\_210089

WHERE LEFT(nim, 2) = '14' AND alamat <> 'Jalan Mangga';

**6. Menampilkan Data Mahasiswa Beserta Umur**

sql

SELECT nim, nama, tanggal\_lahir,

TIMESTAMPDIFF(YEAR, tanggal\_lahir, CURDATE()) AS umur

FROM mahasiswa\_210089;

**7. Analisis Query**

* **Query nomor 3:** Menggunakan LEFT() untuk memfilter tahun angkatan (2 digit pertama NIM).
* **Query nomor 4:** Menggunakan SUBSTRING() untuk memfilter kode fakultas (digit ke-3 dan ke-4).
* **Query nomor 5:** Memanfaatkan operator logika (<>) untuk memfilter alamat yang tidak sesuai.
* **Query nomor 6:** Menggunakan fungsi TIMESTAMPDIFF() untuk menghitung umur berdasarkan tanggal lahir.

**Praktikum 11**

**1. Membuat Database**

1. **Membuat database db\_latihan1:**

sql

CREATE DATABASE db\_latihan1;

1. **Melihat daftar database:**

sql

SHOW DATABASES;

**2. Membuat Tabel Sederhana**

1. **Mengaktifkan database db\_latihan1:**

sql

USE db\_latihan1;

1. **Membuat tabel mhs:**

sql

CREATE TABLE mhs (

nim VARCHAR(10) PRIMARY KEY,

nama VARCHAR(50),

jurusan VARCHAR(50)

);

1. **Melihat daftar tabel:**

sql

SHOW TABLES;

1. **Melihat struktur tabel:**

sql

DESCRIBE mhs;

**3. Pengelolaan Tabel**

1. **Menambahkan data ke tabel mhs:**

sql

INSERT INTO mhs (nim, nama, jurusan)

VALUES

('210001', 'Ani', 'Sistem Informasi'),

('210002', 'Budi', 'Teknik Informatika');

1. **Mengubah nama tabel mhs menjadi mahasiswa:**

sql

RENAME TABLE mhs TO mahasiswa;

1. **Menambahkan kolom angkatan ke tabel mahasiswa:**

sql

ALTER TABLE mahasiswa ADD angkatan YEAR;

1. **Menghapus kolom angkatan:**

sql

ALTER TABLE mahasiswa DROP COLUMN angkatan;

1. **Menghapus tabel mahasiswa:**

sql

DROP TABLE mahasiswa;

**4. Tugas Tambahan**

1. **Membuat tabel jurusan dengan struktur:**

sql

CREATE TABLE jurusan (

kd\_jurusan INT PRIMARY KEY,

nama\_jurusan VARCHAR(50)

);

1. **Mengisi data ke tabel jurusan:**

sql

INSERT INTO jurusan (kd\_jurusan, nama\_jurusan)

VALUES

(1, 'Sistem Informasi'),

(2, 'Teknik Informatika');

1. **Menghapus database db\_latihan1:**

sql

DROP DATABASE db\_latihan1;

**Praktikum 12**

**Hak Akses User**

1. **Hak Akses Global**
   * Memberikan hak akses global kepada user:

sql

GRANT SELECT ON \*.\* TO 'user'@'localhost';

1. **Hak Akses Level Database**
   * Memberikan hak akses pada database universitas:

sql

GRANT SELECT ON universitas.\* TO 'user'@'localhost';

1. **Hak Akses Level Tabel**
   * Memberikan hak akses pada tabel mahasiswa dalam database universitas:

sql

GRANT SELECT ON universitas.mahasiswa TO 'user'@'localhost';

1. **Hak Akses Level Kolom**
   * Memberikan hak akses pada kolom nama dan umur dalam tabel mahasiswa:

sql

GRANT SELECT (nama, umur) ON universitas.mahasiswa TO 'user'@'localhost';

**Menggunakan Perintah GRANT**

1. **Memberikan Hak Akses User untuk Melakukan SELECT pada Database db\_latihan:**

sql

GRANT SELECT ON db\_latihan.\* TO 'student'@'localhost' IDENTIFIED BY 'password123';

1. **Memberikan Hak Akses INSERT pada Tabel mahasiswa:**

sql

GRANT INSERT ON db\_latihan.mahasiswa TO 'student'@'localhost';

1. **Melihat Hak Akses User:**

sql

SHOW GRANTS FOR 'student'@'localhost';

**Menghapus Hak Akses User**

1. **Menghapus Hak Akses pada Database db\_latihan:**

sql

REVOKE ALL PRIVILEGES ON db\_latihan.\* FROM 'student'@'localhost';

1. **Menghapus User:**

sql

DROP USER 'student'@'localhost';

**Tugas**

1. **Memberikan Hak Akses SELECT dan UPDATE pada Database mahasiswa\_db kepada User admin:**

sql

GRANT SELECT, UPDATE ON mahasiswa\_db.\* TO 'admin'@'localhost' IDENTIFIED BY 'admin123';

1. **Melihat Hak Akses yang Dimiliki User admin:**

sql

SHOW GRANTS FOR 'admin'@'localhost';

1. **Menghapus Hak Akses UPDATE dari User admin:**

sql

REVOKE UPDATE ON mahasiswa\_db.\* FROM 'admin'@'localhost';

1. **Menghapus User admin:**

sql

DROP USER 'admin'@'localhost';

**Praktikum 13**

**Data Definition Language (DDL)**

1. **Membuat Database**
   * Membuat dua database:

sql

CREATE DATABASE komik\_coba1;

CREATE DATABASE komik\_cobalagi;

* + Menampilkan daftar database:

sql

SHOW DATABASES;

* + Menghapus database komik\_cobalagi:

sql

DROP DATABASE komik\_cobalagi;

1. **Membuat Tabel anggota**

sql

CREATE TABLE anggota (

id\_anggota INT PRIMARY KEY,

nama\_anggota VARCHAR(50) NOT NULL,

alamat VARCHAR(100) NOT NULL

);

1. **Menambahkan Kolom dan Mengubah Nama Tabel**
   * Menambahkan kolom tgl\_daftar:

sql

ALTER TABLE anggota ADD tgl\_daftar DATE;

* + Mengubah nama tabel anggota menjadi anggota\_warnet:

sql

RENAME TABLE anggota TO anggota\_warnet;

1. **Mengubah dan Menghapus Kolom**
   * Menambahkan kolom status ke tabel komik:

sql

ALTER TABLE komik ADD status VARCHAR(10);

* + Menghapus kolom jenis\_komik dari tabel komik:

sql

ALTER TABLE komik DROP COLUMN jenis\_komik;

1. **Menghapus Primary Key dan Mengganti Kolom**
   * Menghapus primary key pada tabel anggota\_warnet:

sql

ALTER TABLE anggota\_warnet DROP PRIMARY KEY;

* + Mengganti kolom jumlah pada tabel transaksi menjadi jml\_komik:

sql

ALTER TABLE transaksi CHANGE jumlah jml\_komik INT(4);

**Tugas Tambahan**

1. **Membuat Tabel komik dan pinjam**
   * **Tabel komik:**

sql

CREATE TABLE komik (

id\_komik INT PRIMARY KEY,

nama\_komik VARCHAR(50) NOT NULL,

jenis\_komik VARCHAR(50)

);

* + **Tabel pinjam:**

sql

CREATE TABLE pinjam (

no\_pinjam INT PRIMARY KEY,

id\_anggota INT NOT NULL,

id\_komik INT NOT NULL,

FOREIGN KEY (id\_anggota) REFERENCES anggota\_warnet(id\_anggota),

FOREIGN KEY (id\_komik) REFERENCES komik(id\_komik)

);

1. **Menghapus Database komik\_coba1:**

sql

DROP DATABASE komik\_coba1;

**Praktikum 14**

**Data Manipulation Language (DML)**

1. **Membuat Database db\_latihan\_dml:**

sql

CREATE DATABASE db\_latihan\_dml;

1. **Mengaktifkan Database:**

sql

USE db\_latihan\_dml;

1. **Membuat Tabel buku:**

sql

CREATE TABLE buku (

kode\_buku VARCHAR(10) PRIMARY KEY,

judul\_buku VARCHAR(100),

penulis VARCHAR(50),

penerbit VARCHAR(50),

thn\_terbit YEAR

);

1. **Mengisi Data ke Tabel buku:**

sql

INSERT INTO buku (kode\_buku, judul\_buku, penulis, penerbit, thn\_terbit)

VALUES

('BK01', 'Basis Data', 'Dewi Lestari', 'Abadi Jaya', 2000),

('BK02', 'Pemrograman Python', 'Raden Putra', 'Tekno Pustaka', 2010),

('BK03', 'Jaringan Komputer', 'Budi Sudarsono', 'Informatika Press', 2015),

('BK04', 'Multimedia', 'Tania Rahma', 'Media Kreatif', 2020),

('BK05', 'SQL Mastery', 'Dewi Lestari', 'Abadi Jaya', 2021);

1. **Menampilkan Semua Data:**

sql

SELECT \* FROM buku;

1. **Menampilkan Data dengan Kriteria**
   * **Buku dengan penerbit "Abadi Jaya":**

sql

SELECT \* FROM buku

WHERE penerbit = 'Abadi Jaya';

* + **Buku dengan tahun terbit sebelum 2005:**

sql

SELECT \* FROM buku

WHERE thn\_terbit < 2005;

* + **Buku dengan judul mengandung kata "SQL":**

sql

SELECT \* FROM buku

WHERE judul\_buku LIKE '%SQL%';

1. **Mengurutkan Data:**
   * **Berdasarkan penerbit secara menaik (ascending):**

sql

SELECT \* FROM buku

ORDER BY penerbit ASC;

* + **Berdasarkan penerbit secara menurun (descending):**

sql

SELECT \* FROM buku

ORDER BY penerbit DESC;

1. **Menampilkan Data dengan Pengelompokan:**
   * **Pengelompokan berdasarkan penerbit:**

sql

SELECT penerbit, COUNT(\*) AS jumlah\_buku

FROM buku

GROUP BY penerbit;

1. **Menampilkan Data Teratas:**
   * **Menampilkan 3 data teratas:**

sql

SELECT \* FROM buku

LIMIT 3;

**Tugas**

1. **Query untuk Mengubah Data:**
   * **Mengubah judul buku "Multimedia" menjadi "Desain Multimedia":**

sql

UPDATE buku

SET judul\_buku = 'Desain Multimedia'

WHERE judul\_buku = 'Multimedia';

* + **Mengubah penulis "Dewi Lestari" menjadi "Lestari Dewi":**

sql

UPDATE buku

SET penulis = 'Lestari Dewi'

WHERE penulis = 'Dewi Lestari';

1. **Query untuk Menghapus Data:**
   * **Menghapus buku dengan kode\_buku "BK05":**

sql

DELETE FROM buku

WHERE kode\_buku = 'BK05';

1. **Menampilkan Data Setelah Penghapusan:**

sql

SELECT \* FROM buku;

**Praktikum 15**

**Operator Basis Data**

1. **Operator Aritmatika**
   * **Penjumlahan (+):**

sql

SELECT 10 + 5 AS hasil\_penjumlahan;

* + **Pengurangan (-):**

sql

SELECT 10 - 5 AS hasil\_pengurangan;

* + **Perkalian (\*):**

sql

SELECT 10 \* 5 AS hasil\_perkalian;

* + **Pembagian (/):**

sql

SELECT 10 / 5 AS hasil\_pembagian;

* + **Modulus (%):**

sql

SELECT 10 % 3 AS hasil\_modulus;

1. **Operator Pembanding**
   * **Sama dengan (=):**

sql

SELECT \* FROM buku WHERE penerbit = 'Abadi Jaya';

* + **Tidak sama dengan (!= atau <>):**

sql

SELECT \* FROM buku WHERE penerbit != 'Abadi Jaya';

* + **Kurang dari (<):**

sql

SELECT \* FROM buku WHERE thn\_terbit < 2010;

* + **Antara dua nilai (BETWEEN):**

sql

SELECT \* FROM buku WHERE thn\_terbit BETWEEN 2005 AND 2020;

1. **Operator Logika**
   * **AND:**

sql

SELECT \* FROM buku WHERE penerbit = 'Abadi Jaya' AND thn\_terbit > 2005;

* + **OR:**

sql

SELECT \* FROM buku WHERE penerbit = 'Abadi Jaya' OR thn\_terbit < 2010;

* + **NOT:**

sql

SELECT \* FROM buku WHERE NOT penerbit = 'Abadi Jaya';

**Tugas**

1. **Membuat Tabel member:**

sql

CREATE TABLE member (

id\_member INT PRIMARY KEY,

nama\_member VARCHAR(50),

semester INT,

usia INT,

alamat VARCHAR(100)

);

1. **Mengisi Data ke Tabel member:**

sql

INSERT INTO member (id\_member, nama\_member, semester, usia, alamat)

VALUES

(1, 'Rina', 2, 19, 'Jakarta'),

(2, 'Budi', 4, 21, 'Bandung'),

(3, 'Citra', 3, 20, 'Surabaya'),

(4, 'Dedi', 5, 22, 'Medan');

1. **Query untuk Seleksi Data:**
   * **Menampilkan member dengan usia di bawah 21 tahun:**

sql

SELECT \* FROM member WHERE usia < 21;

* + **Menampilkan member yang bukan "Rina":**

sql

SELECT \* FROM member WHERE nama\_member <> 'Rina';

* + **Mengurutkan data berdasarkan semester secara menaik:**

sql

SELECT \* FROM member ORDER BY semester ASC;

* + **Menampilkan 2 record pertama:**

sql

SELECT \* FROM member LIMIT 2;

**Praktikum 17**

**Fungsi Agregasi**

1. **AVG (Average):** Menghitung nilai rata-rata.

sql

SELECT AVG(usia) AS rata\_rata\_usia

FROM member;

1. **COUNT:** Menghitung jumlah data.

sql

SELECT COUNT(\*) AS jumlah\_member

FROM member;

1. **MAX:** Menampilkan nilai terbesar.

sql

SELECT MAX(usia) AS usia\_tertinggi

FROM member;

1. **MIN:** Menampilkan nilai terkecil.

sql

SELECT MIN(usia) AS usia\_terendah

FROM member;

1. **SUM:** Menjumlahkan nilai dalam sebuah kolom.

sql

SELECT SUM(semester) AS total\_semester

FROM member;

**GROUP BY**

1. **Mengelompokkan data berdasarkan semester dan menghitung jumlah member di setiap semester:**

sql

SELECT semester, COUNT(\*) AS jumlah\_member

FROM member

GROUP BY semester;

**HAVING**

1. **Menampilkan semester dengan jumlah member lebih dari 1:**

sql

SELECT semester, COUNT(\*) AS jumlah\_member

FROM member

GROUP BY semester

HAVING COUNT(\*) > 1;

**DISTINCT**

1. **Menampilkan daftar alamat unik (tanpa duplikasi):**

sql

SELECT DISTINCT alamat

FROM member;

**Alias**

1. **Menggunakan alias untuk kolom:**

sql

SELECT nama\_member AS Nama, usia AS Umur

FROM member;

1. **Menggunakan alias untuk tabel:**

sql

SELECT m.nama\_member, m.usia

FROM member AS m;

**Tugas**

1. **Menampilkan jumlah total usia semua member:**

sql

SELECT SUM(usia) AS total\_usia

FROM member;

1. **Menampilkan rata-rata usia member yang berada di semester lebih dari 2:**

sql

Copy code

SELECT AVG(usia) AS rata\_rata\_usia

FROM member

WHERE semester > 2;

1. **Mengelompokkan data berdasarkan alamat dan menampilkan jumlah member di setiap alamat:**

sql

SELECT alamat, COUNT(\*) AS jumlah\_member

FROM member

GROUP BY alamat;

1. **Menampilkan jumlah member yang usia di bawah 21 tahun menggunakan HAVING:**

sql

SELECT alamat, COUNT(\*) AS jumlah\_member

FROM member

GROUP BY alamat

HAVING AVG(usia) < 21;