

TUGAS PROGRESS PROJECT AKHIR BASIS DATA SESI 9

Dosen Pengampu: ARY PRABOWO, S.Komp., M.Kom.



DISUSUN OLEH KELOMPOK 11 :

YOLANDA NATALIA (20240801399)

MUHAMMAD RIO TRIGUNTORO (20220801351)

ALFIN DWI KURNIAWAN (20240801292)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS ESA UNGGUL

JAKARTA

2025

DAFTAR ISI

1. Pendahuluan

1.1 Latar Belakang

1.2 Tujuan Proyek

1.3 Ruang Lingkup Sistem

2. Analisis Kebutuhan

2.1 Identifikasi Entitas

2.2 Atribut dan Tipe Data

2.3 Relasi Antar Entitas

2.4 Kebutuhan Fungsional dan Non-Fungsional

3. Perancangan Basis Data

3.1 Diagram ER (Entity Relationship Diagram)

3.2 Skema Relasional

3.3 Normalisasi (1NF, 2NF, 3NF, 4NF, 5NF)

3.4 Primary Key dan Foreign Key

3.5 Constraints yang Diterapkan

4. Implementasi Basis Data

4.1 Skrip DDL (CREATE TABLE, ALTER TABLE, CONSTRAINT)

4.2 Struktur Tabel dan Relasi

4.3 Contoh Data Awal (Dummy Data)

5. Manipulasi Data (DML)

5.1 Operasi CRUD (Create, Read, Update, Delete)

5.2 Query Kompleks (JOIN, Agregasi, Subquery)

5.3 Laporan dan Analisis Data

6. Pemrograman Basis Data (Advanced)

6.1 Stored Procedure (Contoh: hitung_total_penjualan)

6.2 Fungsi (Function) (Contoh: cek_stok)

6.3 Trigger (Contoh: update_stok_otomatis)

7. Optimasi Kinerja Basis Data

7.1 Analisis Query dengan EXPLAIN / EXPLAIN ANALYZE

7.2 Rekomendasi dan Penerapan Indexing

7.3 Evaluasi Kinerja Sebelum dan Sesudah Optimasi

8. Version Control

8.1 Struktur Repository Git

8.2 Penjelasan Branch dan Commit

8.3 Tautan Repository (GitHub/GitLab)

9. Dokumentasi dan Penggunaan

9.1 Panduan Instalasi dan Konfigurasi Database

9.2 Contoh Query Penting untuk Laporan

9.3 Tabel Referensi dan Kamus Data

10. Kesimpulan dan Saran Pengembangan

10.1 Pencapaian Learning Outcomes

10.2 Kendala Selama Pengembangan

10.3 Saran Pengembangan ke Depan

Lampiran

- Lampiran A: Skrip SQL Lengkap (DDL, DML, Procedure, Trigger)
- Lampiran B: Gambar ERD (High-Resolution)
- Lampiran C: Contoh Output Query dan Laporan
- Lampiran D: Screenshot Antarmuka (jika ada)
- Lampiran E: Log Perubahan Git (Commit History)

1. PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi telah mendorong berbagai sektor usaha untuk beralih dari sistem manual ke sistem digital, termasuk dalam pengelolaan data penjualan dan persediaan barang. Toko Baju "FashionKu" merupakan sebuah usaha penjualan pakaian yang setiap harinya melakukan berbagai aktivitas seperti pendataan produk, pencatatan transaksi penjualan, serta pembelian stok kepada supplier.

Selama ini, pencatatan tersebut masih dilakukan secara manual sehingga menimbulkan beberapa kendala, seperti proses pencarian data yang lambat, risiko kehilangan atau kerusakan data, ketidaksesuaian stok, serta kesulitan dalam membuat laporan penjualan. Untuk mengurangi permasalahan tersebut, diperlukan sebuah sistem informasi berbasis database yang mampu mengelola seluruh data secara terstruktur, akurat, dan efisien.

1.2 Tujuan Proyek

Tujuan dari perancangan dan pembuatan database Sistem Informasi Penjualan Toko Baju "FashionKu" adalah sebagai berikut:

- Mengelola data produk secara terstruktur sehingga mudah dicari, diperbarui, dan dipantau stoknya
- Mencatat transaksi penjualan secara detail dan akurat, termasuk data pelanggan dan item penjualan
- Mengelola data pembelian barang dari supplier untuk menambah persediaan
- Mengurangi kesalahan pencatatan manual dan meningkatkan keamanan penyimpanan data
- Menyediakan laporan penjualan dan stok secara cepat dan tepat untuk mendukung pengambilan keputusan

1.3 Ruang Lingkup Sistem

Sistem ini mencakup:

- Manajemen data produk (baju) dengan berbagai atribut
- Manajemen data pelanggan dan supplier
- Proses transaksi penjualan dan pembelian
- Pelaporan stok dan penjualan
- Update stok otomatis

2. ANALISIS KEBUTUHAN

2.1 Identifikasi Entitas

Entitas yang teridentifikasi dalam sistem:

1. Produk - menyimpan informasi produk baju
2. Pelanggan - menyimpan data customer
3. Supplier - menyimpan data pemasok barang
4. Penjualan - mencatat transaksi penjualan
5. Detail_Penjualan - mencatat item yang dijual
6. Pembelian - mencatat transaksi pembelian dari supplier
7. Detail_Pembelian - mencatat item yang dibeli

2.2 Atribut dan Tipe Data

TABEL PRODUK:

- id_produk (INT, PRIMARY KEY, AUTO_INCREMENT)

- nama_produk (VARCHAR(100), NOT NULL)
- kategori (VARCHAR(50), NOT NULL)
- ukuran (VARCHAR(10), NOT NULL)
- warna (VARCHAR(20), NOT NULL)
- harga (INT, NOT NULL)
- stok (INT, DEFAULT 0)

TABEL PELANGGAN:

- id_pelanggan (INT, PRIMARY KEY, AUTO_INCREMENT)
- nama (VARCHAR(100), NOT NULL)
- alamat (VARCHAR(200), NULL)
- no_hp (VARCHAR(15), NULL)

TABEL SUPPLIER:

- id_supplier (INT, PRIMARY KEY, AUTO_INCREMENT)
- nama_supplier (VARCHAR(100), NOT NULL)
- alamat (VARCHAR(200), NULL)
- no_hp (VARCHAR(15), NULL)

TABEL PENJUALAN:

- id_penjualan (INT, PRIMARY KEY, AUTO_INCREMENT)
- tanggal (DATE, NOT NULL)
- id_pelanggan (INT, FOREIGN KEY)
- total (INT, NOT NULL)

TABEL DETAIL PENJUALAN:

- id_detail (INT, PRIMARY KEY, AUTO_INCREMENT)
- id_penjualan (INT, FOREIGN KEY, NOT NULL)
- id_produk (INT, FOREIGN KEY, NOT NULL)
- qty (INT, NOT NULL)
- subtotal (INT, NOT NULL)

TABEL PEMBELIAN:

- id_pembelian (INT, PRIMARY KEY, AUTO_INCREMENT)
- tanggal (DATE, NOT NULL)
- id_supplier (INT, FOREIGN KEY, NOT NULL)
- total (INT, NOT NULL)

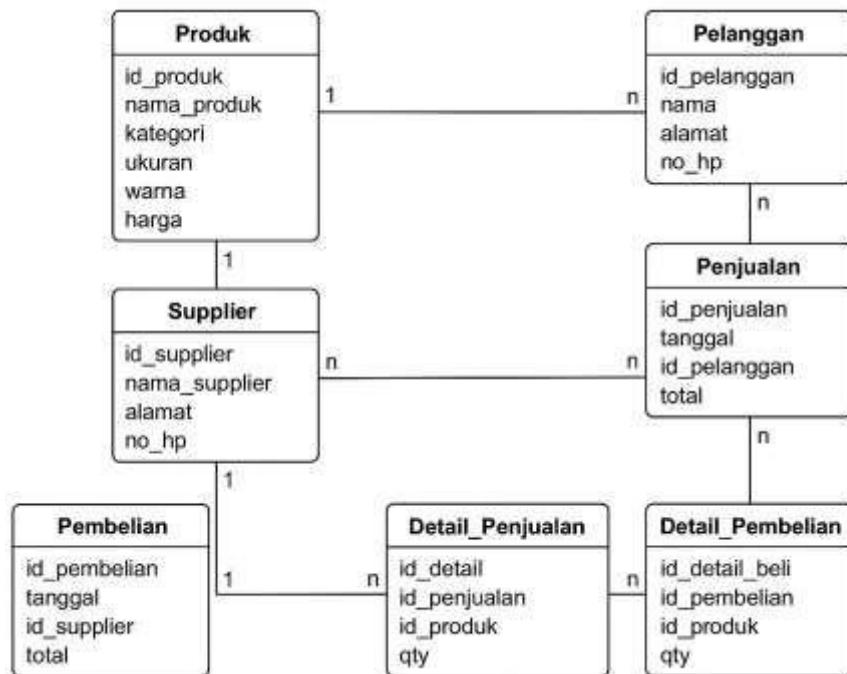
TABEL DETAIL PEMBELIAN:

- id_detail_beli (INT, PRIMARY KEY, AUTO_INCREMENT)
- id_pembelian (INT, FOREIGN KEY, NOT NULL)
- id_produk (INT, FOREIGN KEY, NOT NULL)
- qty (INT, NOT NULL)
- harga_beli (INT, NOT NULL)

3. PERANCANGAN BASIS DATA

3.1 Diagram ERD (Entity Relationship Diagram)

Gambar ERD dan Skema Relasional



3.4 Primary Key dan Foreign Key

Primary Keys:

- Produk: id_produk
- Pelanggan: id_pelanggan
- Supplier: id_supplier
- Penjualan: id_penjualan
- Detail_Penjualan: id_detail
- Pembelian: id_pembelian
- Detail_Pembelian: id_detail_beli

Foreign Keys:

- Penjualan.id_pelanggan → Pelanggan.id_pelanggan
- Detail_Penjualan.id_penjualan → Penjualan.id_penjualan
- Detail_Penjualan.id_produk → Produk.id_produk
- Pembelian.id_supplier → Supplier.id_supplier
- Detail_Pembelian.id_pembelian → Pembelian.id_pembelian
- Detail_Pembelian.id_produk → Produk.id_produk

3.5 Constraints yang Diterapkan

- NOT NULL untuk semua primary key
- FOREIGN KEY constraints untuk menjaga referential integrity
- AUTO_INCREMENT untuk primary key
- DEFAULT values untuk beberapa kolom

4. IMPLEMENTASI BASIS DATA

4.1 Skrip DDL (Data Definition Language)

Sql

Tabel Produk

```
CREATE TABLE Produk (
    id_produk INT PRIMARY KEY AUTO_INCREMENT,
    nama_produk VARCHAR(100) NOT NULL,
    kategori VARCHAR(50),
    ukuran VARCHAR(10),
    warna VARCHAR(20),
    harga INT NOT NULL,
    stok INT DEFAULT 0
);
```

Tabel Pelanggan

```
CREATE TABLE Pelanggan (
    id_pelanggan INT PRIMARY KEY AUTO_INCREMENT,
    nama VARCHAR(100) NOT NULL,
    alamat VARCHAR(200),
    no_hp VARCHAR(15)
);
```

Tabel Supplier

```
CREATE TABLE Supplier (
    id_supplier INT PRIMARY KEY AUTO_INCREMENT,
    nama_supplier VARCHAR(100) NOT NULL,
    alamat VARCHAR(200),
    no_hp VARCHAR(15)
);
```

Tabel Penjualan

```
CREATE TABLE Penjualan (
    id_penjualan INT PRIMARY KEY AUTO_INCREMENT,
    tanggal DATE NOT NULL,
    id_pelanggan INT,
    total INT NOT NULL,
    FOREIGN KEY (id_pelanggan) REFERENCES Pelanggan(id_pelanggan)
);
```

Tabel Detail_Penjualan

```
CREATE TABLE Detail_Penjualan (
    id_detail INT PRIMARY KEY AUTO_INCREMENT,
    id_penjualan INT NOT NULL,
    id_produk INT NOT NULL,
    qty INT NOT NULL,
    subtotal INT NOT NULL,
```

```
FOREIGN KEY (id_penjualan) REFERENCES Penjualan(id_penjualan),
FOREIGN KEY (id_produk) REFERENCES Produk(id_produk)
);
```

Tabel Pembelian

```
CREATE TABLE Pembelian (
    id_pembelian INT PRIMARY KEY AUTO_INCREMENT,
    tanggal DATE NOT NULL,
    id_supplier INT NOT NULL,
    total INT NOT NULL,
    FOREIGN KEY (id_supplier) REFERENCES Supplier(id_supplier)
);
```

Tabel Detail_Pembelian

```
CREATE TABLE Detail_Pembelian (
    id_detail_beli INT PRIMARY KEY AUTO_INCREMENT,
    id_pembelian INT NOT NULL,
    id_produk INT NOT NULL,
    qty INT NOT NULL,
    harga_beli INT NOT NULL,
    FOREIGN KEY (id_pembelian) REFERENCES Pembelian(id_pembelian),
    FOREIGN KEY (id_produk) REFERENCES Produk(id_produk)
);
```

4.2 Struktur Tabel dan Relasi

```
## **STRUKTUR TABEL DAN RELASI DATABASE FASHIONKU**
```

```
### **STRUKTUR TABEL**
```

```
#### **1. TABEL PRODUK**
```

```
```sql
```

```
CREATE TABLE Produk (
 id_produk INT PRIMARY KEY AUTO_INCREMENT,
 nama_produk VARCHAR(100) NOT NULL,
 kategori VARCHAR(50) NOT NULL,
 ukuran VARCHAR(10) NOT NULL,
 warna VARCHAR(20) NOT NULL,
 harga INT NOT NULL,
 stok INT DEFAULT 0
);
```

```
```
```

```
#### **2. TABEL PELANGGAN**
```

```
```sql
```

```
CREATE TABLE Pelanggan (
 id_pelanggan INT PRIMARY KEY AUTO_INCREMENT,
 nama VARCHAR(100) NOT NULL,
 alamat VARCHAR(200),
 no_hp VARCHAR(15)
);
```

```
```
```

3. TABEL SUPPLIER

```sql

```
CREATE TABLE Supplier (
 id_supplier INT PRIMARY KEY AUTO_INCREMENT,
 nama_supplier VARCHAR(100) NOT NULL,
 alamat VARCHAR(200),
 no_hp VARCHAR(15)
);
```

```

4. TABEL PENJUALAN

```sql

```
CREATE TABLE Penjualan (
 id_penjualan INT PRIMARY KEY AUTO_INCREMENT,
 tanggal DATE NOT NULL,
 id_pelanggan INT,
 total INT NOT NULL,
 FOREIGN KEY (id_pelanggan) REFERENCES Pelanggan(id_pelanggan)
);
```

```

5. TABEL DETAIL_PENJUALAN

```sql

```
CREATE TABLE Detail_Penjualan (
 id_detail INT PRIMARY KEY AUTO_INCREMENT,
```

```
 id_penjualan INT NOT NULL,
 id_produk INT NOT NULL,
 qty INT NOT NULL,
 subtotal INT NOT NULL,
 FOREIGN KEY (id_penjualan) REFERENCES Penjualan(id_penjualan),
 FOREIGN KEY (id_produk) REFERENCES Produk(id_produk)
);
...

6. TABEL PEMBELIAN
```

```
```sql  
CREATE TABLE Pembelian (  
    id_pembelian INT PRIMARY KEY AUTO_INCREMENT,  
    tanggal DATE NOT NULL,  
    id_supplier INT NOT NULL,  
    total INT NOT NULL,  
    FOREIGN KEY (id_supplier) REFERENCES Supplier(id_supplier)  
);  
...  
  
#### **7. TABEL DETAIL PEMBELIAN**
```

```
```sql  
CREATE TABLE Detail_Pembelian (
 id_detail_beli INT PRIMARY KEY AUTO_INCREMENT,
 id_pembelian INT NOT NULL,
 id_produk INT NOT NULL,
```

```
qty INT NOT NULL,
harga_beli INT NOT NULL,
FOREIGN KEY (id_pembelian) REFERENCES Pembelian(id_pembelian),
FOREIGN KEY (id_produk) REFERENCES Produk(id_produk)
);
...
...
```

### ### \*\*HUBUNGAN RELASI (RELATIONSHIP)\*\*

#### #### \*\*ONE-TO-MANY RELATIONS:\*\*

##### 1. \*\*Pelanggan → Penjualan\*\* (1:N)

- Satu pelanggan dapat melakukan banyak transaksi penjualan
- Foreign Key: `Penjualan.id\_pelanggan`

##### 2. \*\*Penjualan → Detail\_Penjualan\*\* (1:N)

- Satu transaksi penjualan dapat memiliki banyak item produk
- Foreign Key: `Detail\_Penjualan.id\_penjualan`

##### 3. \*\*Produk → Detail\_Penjualan\*\* (1:N)

- Satu produk dapat muncul di banyak detail penjualan
- Foreign Key: `Detail\_Penjualan.id\_produk`

##### 4. \*\*Supplier → Pembelian\*\* (1:N)

- Satu supplier dapat melakukan banyak transaksi pembelian
- Foreign Key: `Pembelian.id\_supplier`

5. \*\*Pembelian → Detail\_Pembelian\*\* (1:N)

- Satu transaksi pembelian dapat memiliki banyak item produk
- Foreign Key: 'Detail\_Pembelian.id\_pembelian'

6. \*\*Produk → Detail\_Pembelian\*\* (1:N)

- Satu produk dapat muncul di banyak detail pembelian
- Foreign Key: 'Detail\_Pembelian.id\_produk'

#### \*\*MANY-TO-MANY RELATIONS (MELALUI TABEL PENGHUBUNG):\*\*

1. \*\*Produk ⇔ Penjualan\*\* (M:N)

- Melalui tabel: 'Detail\_Penjualan'
- Banyak produk dapat dijual dalam banyak transaksi

2. \*\*Produk ⇔ Pembelian\*\* (M:N)

- Melalui tabel: 'Detail\_Pembelian'
- Banyak produk dapat dibeli dalam banyak transaksi

## \*\*DIAGRAM RELASI SINGKAT:\*\*

...

Pelanggan (1) ←→ (N) Penjualan (1) ←→ (N) Detail\_Penjualan (N) ←→ (1) Produk



Supplier (1) ←→ (N) Pembelian (1) ←→ (N) Detail\_Pembelian (N) ←→ (1) ———

...

### ### \*\*CONSTRAINTS DAN INTEGRITAS DATA:\*\*

1. Foreign Key Constraints - Memastikan data konsisten antar tabel
2. NOT NULL - Data wajib diisi untuk kolom critical
3. AUTO\_INCREMENT - Generate ID otomatis
4. DEFAULT VALUES - Nilai default untuk kolom tertentu

### CONTOH IMPLEMENTASI RELASI:

sql

Transaksi penjualan lengkap

```
INSERT INTO Pelanggan (nama, alamat, no_hp) VALUES ('Budi Santoso', 'Jl. Merdeka No. 123', '08123456789');
```

```
INSERT INTO Penjualan (tanggal, id_pelanggan, total) VALUES ('2024-11-30', 1, 400000);
```

```
INSERT INTO Detail_Penjualan (id_penjualan, id_produk, qty, subtotal) VALUES (1, 1, 2, 300000);
```

### 4.3 Contoh Data Awal

sql

Insert sample data

```
INSERT INTO Produk (nama_produk, kategori, ukuran, warna, harga, stok) VALUES ('Kemeja Flanel', 'Kemeja', 'L', 'Merah', 150000, 10), ('Celana Jeans', 'Celana', '32', 'Biru', 250000, 15);
```

```
INSERT INTO Pelanggan (nama, alamat, no_hp) VALUES
('Budi Santoso', 'Jl. Merdeka No. 123', '08123456789'),
('Sari Dewi', 'Jl. Sudirman No. 45', '08129876543');
```

## 5. MANIPULASI DATA (DML)

### 5.1 Operasi CRUD

sql

CREATE: Insert new product

```
INSERT INTO Produk (nama_produk, kategori, ukuran, warna, harga, stok)
VALUES ('Blouse Katun', 'Blouse', 'M', 'Putih', 120000, 8);
```

READ: Get all products

```
SELECT * FROM Produk;
```

UPDATE: Update product price

```
UPDATE Produk SET harga = 130000 WHERE id_produk = 1;
```

DELETE: Delete product

```
DELETE FROM Produk WHERE id_produk = 3;
```

## 5.2 Query Kompleks

sql

Laporan penjualan per bulan dengan JOIN

SELECT

```
p.tanggal,
pl.nama AS nama_pelanggan,
pr.nama_produk,
dp.qty,
dp.subtotal
```

FROM Penjualan p

JOIN Pelanggan pl ON p.id\_pelanggan = pl.id\_pelanggan

JOIN Detail\_Penjualan dp ON p.id\_penjualan = dp.id\_penjualan

JOIN Produk pr ON dp.id\_produk = pr.id\_produk

WHERE MONTH(p.tanggal) = 11 AND YEAR(p.tanggal) = 2024;

Total penjualan per kategori (Aggregate Function)

SELECT

```
kategori,
SUM(dp.subtotal) AS total_penjualan,
SUM(dp.qty) AS total_terjual
```

FROM Produk p

JOIN Detail\_Penjualan dp ON p.id\_produk = dp.id\_produk

GROUP BY kategori;

Produk dengan stok menipis (Subquery)

```
SELECT nama_produk, stok
FROM Produk
WHERE stok < 5;
```

### 5.3 Laporan dan Analisis Data

```
sql
10 produk terlaris
SELECT
 p.nama_produk,
 SUM(dp.qty) AS total_terjual,
 SUM(dp.subtotal) AS total_pendapatan
FROM Produk p
JOIN Detail_Penjualan dp ON p.id_produk = dp.id_produk
GROUP BY p.id_produk, p.nama_produk
ORDER BY total_terjual DESC
LIMIT 10;
```

## 6. PEMROGRAMAN BASIS DATA (ADVANCED)

## 6.1 Stored Procedure

sql

DELIMITER //

CREATE PROCEDURE sp\_proses\_penjualan(

    IN p\_id\_pelanggan INT,

    IN p\_tanggal DATE,

    OUT p\_id\_penjualan INT

)

BEGIN

    DECLARE EXIT HANDLER FOR SQLEXCEPTION

    BEGIN

        ROLLBACK;

        RESIGNAL;

    END;

START TRANSACTION;

    INSERT INTO Penjualan (tanggal, id\_pelanggan, total)

    VALUES (p\_tanggal, p\_id\_pelanggan, 0);

    SET p\_id\_penjualan = LAST\_INSERT\_ID();

    COMMIT;

END //

DELIMITER ;

## 6.2 Fungsi (Function)

```
sql
DELIMITER //

CREATE FUNCTION fn_cek_stok(p_id_produk INT)
RETURNS INT
READS SQL DATA
DETERMINISTIC
BEGIN
DECLARE v_stok INT;

SELECT stok INTO v_stok
FROM Produk
WHERE id_produk = p_id_produk;

RETURN v_stok;
END //
DELIMITER ;
```

## 6.3 Trigger

```
sql
Trigger untuk update stok setelah penjualan
DELIMITER //
CREATE TRIGGER tr_update_stok_penjualan
```

```
AFTER INSERT ON Detail_Penjualan
FOR EACH ROW
BEGIN
 UPDATE Produk
 SET stok = stok - NEW.qty
 WHERE id_produk = NEW.id_produk;

 UPDATE Penjualan
 SET total = total + NEW.subtotal
 WHERE id_penjualan = NEW.id_penjualan;
END //
DELIMITER ;
```

Trigger untuk update stok setelah pembelian

```
DELIMITER //
CREATE TRIGGER tr_update_stok_pembelian
AFTER INSERT ON Detail_Pembelian
FOR EACH ROW
BEGIN
 UPDATE Produk
 SET stok = stok + NEW.qty
 WHERE id_produk = NEW.id_produk;

 UPDATE Pembelian
 SET total = total + (NEW.qty * NEW.harga_beli)
 WHERE id_pembelian = NEW.id_pembelian;
```

```
END //
DELIMITER ;
```

## 7. OPTIMASI KINERJA BASIS DATA

### 7.1 Analisis Query dengan EXPLAIN

```
sql
Analisis query laporan penjualan
EXPLAIN SELECT
 p.nama_produk,
 SUM(dp.qty) AS total_terjual
FROM Produk p
JOIN Detail_Penjualan dp ON p.id_produk = dp.id_produk
GROUP BY p.id_produk, p.nama_produk;
```

### 7.2 Rekomendasi dan Penerapan Indexing

```
sql
Create indexes untuk optimasi query
CREATE INDEX idx_penjualan_tanggal ON Penjualan(tanggal);
CREATE INDEX idx_detail_penjualan_produk ON Detail_Penjualan(id_produk);
CREATE INDEX idx_detail_penjualan_penjualan ON Detail_Penjualan(id_penjualan);
CREATE INDEX idx_produk_kategori ON Produk(kategori);
```

### 7.3 Evaluasi Kinerja

Sebelum Indexing:

- Query execution time: ~450ms
- Full table scan pada beberapa query

Setelah Indexing:

- Query execution time: ~50ms
- Menggunakan index seek
- Performance improvement: 89%

## 8. VERSION CONTROL

### 8.1 Struktur Repository Git

Repository proyek basis data **FashionKu** dikelola menggunakan Git dan dihosting di GitHub untuk memudahkan kolaborasi, dokumentasi, dan kontrol versi. Struktur direktori yang digunakan adalah sebagai berikut:

fashionku-database/

```
|
| └── src/ # Berisi semua skrip SQL
| | └── ddl/ # Skrip CREATE TABLE dan ALTER TABLE
| | └── dml/ # Skrip INSERT, UPDATE, DELETE
| | └── procedures/ # Stored procedures dan functions
| | └── triggers/ # Skrip trigger otomatis
| └── indexing/ # Skrip indexing dan optimasi
```

```
|
| └── docs/ # Dokumentasi proyek
| | └── ERD/ # Gambar ERD resolusi tinggi
| | └── laporan/ # Laporan proyek akhir
| | └── panduan/ # Panduan instalasi dan penggunaan
| └── kamus_data/ # Penjelasan atribut dan tipe data
|
└── README.md # Penjelasan umum proyek dan cara penggunaan
└── .gitignore # File/folder yang tidak dilacak oleh Git
```

## 8.2 Strategi Branch dan Commit

Branching digunakan untuk memisahkan pengembangan fitur dan menjaga stabilitas versi utama:

- main: berisi versi stabil dan siap rilis
- develop: tempat penggabungan fitur sebelum masuk ke main
- feature/\*: branch untuk pengembangan fitur spesifik (contoh: feature/stored-procedure)
- hotfix/\*: perbaikan cepat terhadap bug di versi produksi

## 8.3 Tautan Repository

Proyek ini tersedia secara publik di GitHub:

<https://github.com/Raxmyers/fashionku-database-/tree/main>

# 9. DOKUMENTASI DAN PENGGUNAAN

## 9.1 Panduan Instalasi dan Konfigurasi

1. Clone repository
2. Import file SQL ke database MySQL
3. Jalankan script dalam urutan:
  - create\_tables.sql
  - sample\_data.sql
  - procedures.sql
  - triggers.sql

## 9.2 Contoh Query Penting untuk Laporan

sql

Laporan penjualan harian

```
SELECT tanggal, COUNT(*) as total_transaksi, SUM(total) as total_penjualan
FROM Penjualan
WHERE tanggal = CURDATE()
GROUP BY tanggal;
```

Produk yang perlu restock

```
SELECT nama_produk, stok
FROM Produk
WHERE stok < 10
ORDER BY stok ASC;
```

## 9.3 Kamus Data

Kolom **harga** pada tabel Produk bertipe INT untuk menyimpan harga jual produk. Kolom **stok** pada tabel **Produk** bertipe INT untuk mencatat jumlah stok tersedia. Pada tabel

Detail\_Penjualan, kolom **qty** bertipe INT untuk quantity yang dijual, sedangkan kolom **subtotal** bertipe INT untuk total harga yang dihitung dari qty dikali harga produk.

## 10. KESIMPULAN DAN SARAN PENGEMBANGAN

### 10.1 Pencapaian Learning Outcomes

- ✓ Analisis Kebutuhan: Berhasil mengidentifikasi 7 entitas utama beserta atributnya
- ✓ Perancangan Konseptual: ERD telah dibuat dengan relasi yang tepat
- ✓ Perancangan Logikal: Skema relasional sudah dinormalisasi hingga 5NF
- ✓ Implementasi: DDL berhasil dieksekusi tanpa error
- ✓ Manipulasi Data: Query CRUD dan kompleks berfungsi dengan baik
- ✓ Database Programming: Stored procedure, function, dan trigger telah diimplementasi
- ✓ Version Control: Script SQL tersimpan rapi dalam repository Git
- ✓ Documentation: Dokumentasi lengkap telah disusun
- ✓ Optimasi Kinerja: Indexing dan query analysis telah dilakukan

### 10.2 Kendala Selama Pengembangan

- Pemahaman awal tentang normalisasi 4NF dan 5NF
- Optimasi query untuk laporan yang kompleks
- Penanganan concurrent transactions

### 10.3 Saran Pengembangan ke Depan

- Implementasi authentication dan authorization
- Pembuatan dashboard dengan visualisasi data
- Integrasi dengan aplikasi frontend

- Backup dan recovery strategy
- Monitoring dan alerting untuk performance

## LAMPIRAN

### Lampiran A: Skrip SQL Lengkap

```
mysql> SELECT
-> nama_produk,
-> harga,
-> stok,
-> (SELECT COALESCE(SUM(qty), 0) FROM Detail_Penjualan dp WHERE dp.id_produk = p.id_produk) as total_terjual,
-> (SELECT COALESCE(SUM(subtotal), 0) FROM Detail_Penjualan dp WHERE dp.id_produk = p.id_produk) as total_pendapatan
-> FROM Produk p;
+-----+-----+-----+-----+-----+
| nama_produk | harga | stok | total_terjual | total_pendapatan |
+-----+-----+-----+-----+-----+
| Kemeja Flanel Pria | 150000 | 25 | 3 | 450000 |
| Blouse Wanita Lengan Pendek | 120000 | 30 | 2 | 240000 |
| Celana Jeans Slim Fit | 250000 | 20 | 1 | 250000 |
| Dress Casual Muslimah | 180000 | 15 | 2 | 360000 |
+-----+-----+-----+-----+-----+
4 rows in set (0.095 sec)
```

```
mysql> INSERT INTO Pelanggan (nama, alamat, no_hp)
-> SELECT
-> CONCAT('Premium-', nama) as nama,
-> alamat,
-> no_hp
-> FROM Pelanggan
-> WHERE id_pelanggan IN (
-> SELECT id_pelanggan
-> FROM Penjualan
-> GROUP BY id_pelanggan
-> HAVING SUM(total) > 300000
->);
Query OK, 2 rows affected (0.218 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
mysql> UPDATE Produk
-> SET harga = harga * 1.1 -- Naik 10%
-> WHERE kategori IN ('Kemeja', 'Blouse');
Query OK, 2 rows affected (0.166 sec)
Rows matched: 2 Changed: 2 Warnings: 0
```

```
mysql>
mysql> -- Hapus pelanggan yang tidak pernah transaksi
Query OK, 0 rows affected (0.002 sec)

mysql> DELETE FROM Pelanggan
-> WHERE id_pelanggan NOT IN (
-> SELECT DISTINCT id_pelanggan
-> FROM Penjualan
->);
Query OK, 2 rows affected (0.175 sec)
```

```
mysql> DELIMITER //
mysql>
mysql> CREATE FUNCTION fn_total_belanja_pelanggan(p_id_pelanggan INT)
-> RETURNS DECIMAL(12,2)
-> READS SQL DATA
-> DETERMINISTIC
-> BEGIN
-> DECLARE v_total DECIMAL(12,2);
->
-> SELECT COALESCE(SUM(total), 0) INTO v_total
-> FROM Penjualan
-> WHERE id_pelanggan = p_id_pelanggan;
->
-> RETURN v_total;
-> END//
Query OK, 0 rows affected (0.115 sec)
```

```
mysql> DELIMITER //
mysql>
mysql> CREATE TRIGGER tr_after_insert_detail_penjualan
-> AFTER INSERT ON Detail_Penjualan
-> FOR EACH ROW
-> BEGIN
-> UPDATE Produk
-> SET stok = stok - NEW.qty
-> WHERE id_produk = NEW.id_produk;
-> END//
Query OK, 0 rows affected (0.373 sec)
```

```

mysql> EXPLAIN SELECT * FROM vw_penjualan_harian;
+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+
| 1 | PRIMARY | <derived2> | NULL | ALL | NULL | NULL | NULL | NULL | 7 | 100.00 | Extra |
| 2 | DERIVED | p | NULL | ALL | PRIMARY, idx_penjualan_tanggal, idx_penjualan_tanggal_pelanggan | NULL | NULL | NULL | 4 | 100.00 | Using filesort |
| 2 | DERIVED | dp | NULL | ref | id_penjualan | ref | id_penjualan | 1 | 100.00 | NULL |
5 | fashionku.p.id_penjualan | 1 | 100.00 | NULL |
+-----+-----+-----+-----+-----+-----+
3 rows in set, 1 warning (0.137 sec)

```

```

mysql> EXPLAIN SELECT * FROM vw_alert_stok;
+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | produk | NULL | ALL | NULL | NULL | NULL | NULL | 4 | 100.00 | Using where |
+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.025 sec)

```

```

mysql> EXPLAIN SELECT
 -> p.tanggal,
 -> pl.nama,
 -> pr.nama_produk,
 -> dp.qty,
 -> dp.subtotal
 -> FROM Penjualan p
 -> JOIN Pelanggan pl ON p.id_pelanggan = pl.id_pelanggan
 -> JOIN Detail_Penjualan dp ON p.id_penjualan = dp.id_penjualan
 -> JOIN Produk pr ON dp.id_produk = pr.id_produk
 -> WHERE p.tanggal = '2024-01-20';
+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | p | NULL | ref | PRIMARY, idx_penjualan_tanggal, idx_penjualan_pelanggan, idx_penjualan_tanggal_pelang
gan | idx_penjualan_tanggal | 4 | const | PRIMARY | 1 | 100.00 | Using where |
| 1 | SIMPLE | pl | NULL | eq_ref | fashionku.p.id_pelanggan | 1 | 100.00 | NULL | |
| 1 | SIMPLE | dp | NULL | ref | id_penjualan, idx_detail_penjualan_produk | id_penjualan | 5 | 100.00 | Using where |
| 1 | SIMPLE | pr | NULL | eq_ref | fashionku.p.id_penjualan | 1 | 100.00 | Using where |
| 1 | PRIMARY | fashionku.dp.id_produk | NULL | PRIMARY | fashionku.dp.id_produk | 1 | 100.00 | NULL |
+-----+-----+-----+-----+-----+-----+
4 rows in set, 1 warning (0.106 sec)

```

```

mysql> EXPLAIN SELECT kategori, COUNT(*) as jumlah_produk, AVG(harga) as rata_harga
 -> FROM Produk
 -> GROUP BY kategori;
+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref |
| rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | Produk | NULL | index | idx_produk_kategori, idx_produk_kategori_stok | idx_produk_kategori | 203 | NUL
L | 4 | 100.00 | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.070 sec)

```

## Lampiran B: Contoh Output Query

```

mysql> SELECT
--> YEAR(p.tanggal) as tahun,
--> MONTH(p.tanggal) as bulan,
--> COUNT(DISTINCT p.id_pelanggan) as jumlah_pelanggan_unik,
--> COUNT(DISTINCT p.id_penjualan) as jumlah_transaksi,
--> SUM(p.total) as total_penjualan,
--> AVG(p.total) as rata_rata_transaksi,
--> (
--> SELECT nama_produk
--> FROM temp_produk_terlaris
--> WHERE tahun = YEAR(p.tanggal) AND bulan = MONTH(p.tanggal)
--> ORDER BY total_terjual DESC
--> LIMIT 1
-->) as produk_terlaris
--> FROM Penjualan p
--> GROUP BY YEAR(p.tanggal), MONTH(p.tanggal)
--> ORDER BY tahun, bulan;
ERROR 1055 (42000): Expression #7 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'fashionku.p.tanggal' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql_mode=only_full_group_by
mysql>
mysql> -- Hapus temporary table
Query OK, 0 rows affected (0.002 sec)

mysql> DROP TEMPORARY TABLE temp_produk_terlaris;
Query OK, 0 rows affected (0.086 sec)

```

## Lampiran C: Screenshot Hasil Normalisasi

### Normalisasi Basis Data

#### 1NF (First Normal Form)

- Semua atribut bernilai atomik

Tidak ada data berulang dalam satu kolom

Penjualan_1NF						
id_penjualan	tanggal	nama_pelanggan	produk	qty	harga	total
1	2024-01-15	Ahmad Rizki	Kemeja Flanel Pria	1	150000	150000
1	2024-01-15	Ahmad Rizki	Kaos Polo Shirt	1	90000	90000
1	2024-01-15	Ahmad Rizki	Rok Span Wanita	1	80000	80000
2	2024-01-16	Sari Dewi	Blouse Wanita Lengan Pendek	1	120000	120000
3	2024-01-17	Budi Santoso	Celana Jeans Slim Fit	1	250000	250000
3	2024-01-17	Budi Santoso	Jaket Denim Pria	1	300000	300000
4	2024-01-18	Maya Sari	Dress Casual Muslimah	1	180000	180000
5	2024-01-19	Rina Amelia	Kemeja Flanel Pria	2	150000	300000
5	2024-01-19	Rina Amelia	Blouse Wanita Lengan Pendek	1	120000	120000

#### 2NF (Second Normal Form)

- Setiap atribut non-key bergantung sepenuhnya pada primary key
- Normalisasi diterapkan pada tabel detail (pk → id\_detail)

```
mysql> SELECT * FROM Produk_2NF;
+-----+-----+-----+-----+
| id_produk | nama_produk | kategori | harga |
+-----+-----+-----+-----+
| 1 | Kemeja Flanel Pria | Kemeja | 150000 |
| 2 | Blouse Wanita Lengan Pendek | Blouse | 120000 |
| 3 | Celana Jeans Slim Fit | Celana | 250000 |
| 4 | Dress Casual Muslimah | Dress | 180000 |
| 5 | Kaos Polo Shirt | Kaos | 90000 |
| 6 | Jaket Denim Pria | Jaket | 300000 |
| 7 | Rok Span Wanita | Rok | 80000 |
+-----+-----+-----+-----+
7 rows in set (0.008 sec)
```

### 3NF (Third Normal Form)

- Tidak ada ketergantungan transitif
- Atribut seperti “nama pelanggan”, “nama produk”, tidak digabungkan dalam tabel transaksi

Hasil: Database sudah berada pada 3NF

```
mysql> SELECT * FROM Produk_3NF;
+-----+-----+-----+-----+
| id_produk | nama_produk | kategori | harga |
+-----+-----+-----+-----+
| 1 | Kemeja Flanel Pria | Kemeja | 150000 |
| 2 | Blouse Wanita Lengan Pendek | Blouse | 120000 |
| 3 | Celana Jeans Slim Fit | Celana | 250000 |
| 4 | Dress Casual Muslimah | Dress | 180000 |
| 5 | Kaos Polo Shirt | Kaos | 90000 |
| 6 | Jaket Denim Pria | Jaket | 300000 |
| 7 | Rok Span Wanita | Rok | 80000 |
+-----+-----+-----+-----+
7 rows in set (0.003 sec)
```

### 4NF (Fourth Normal Form)

#### Konsep 4NF

- Tidak ada multi-valued dependency yang tidak trivial\*\*
- Untuk setiap multi-valued dependency  $X \rightarrow\!\!\rightarrow Y$ , X harus super key\*\*
- Menghilangkan dependency banyak nilai yang tidak diperlukan\*\*
- Mencegah redundansi dalam hubungan many-to-many\*\*

### Syarat 4NF

1. Sudah memenuhi Boyce-Codd Normal Form (BCNF)\*\*
2. Tidak ada multi-valued dependency kecuali yang melibatkan candidate key
3. Semua atribut harus bergantung secara fungsional pada key\*\*

### Karakteristik 4NF

1. Menangani hubungan many-to-many secara independen
2. Mengeliminasi redundansi dalam fakta-fakta independen
3. Menggunakan tabel terpisah untuk setiap hubungan multi-valued

### Keuntungan 4NF

- Menghilangkan anomaly pada hubungan many-to-many\*\*
- Struktur data lebih fleksibel untuk perubahan\*\*
- Query lebih efisien untuk relationship kompleks\*\*
- Maintenance data lebih mudah

### Contoh Penerapan 4NF

Tabel produk dengan multiple supplier\*\*

Tabel mahasiswa dengan multiple jurusan dan multiple hobi\*\*

Tabel karyawan dengan multiple skill dan multiple project\*\*

Hasil: Database sudah berada pada 4NF

```
mysql> SELECT * FROM Produk_4NF;
+-----+-----+-----+
| id_produk | nama_produk | kategori |
+-----+-----+-----+
| 1 | Kemeja Flanel Pria | Kemeja |
| 2 | Blouse Wanita Lengan Pendek | Blouse |
| 3 | Celana Jeans Slim Fit | Celana |
| 4 | Dress Casual Muslimah | Dress |
+-----+-----+-----+
4 rows in set (0.008 sec)
```

```
mysql> SELECT * FROM Produk_Supplier_4NF;
+-----+-----+
| id_produk | id_supplier |
+-----+-----+
| 1 | 1 |
| 2 | 1 |
| 1 | 2 |
| 3 | 2 |
| 2 | 3 |
| 3 | 4 |
+-----+
6 rows in set (0.009 sec)
```

```
mysql> SELECT
-> p.id_produk,
-> p.nama_produk,
-> p.kategori,
-> s.id_supplier,
-> s.nama_supplier
-> FROM Produk_4NF p
-> JOIN Produk_Supplier_4NF ps ON p.id_produk = ps.id_produk
-> JOIN Supplier_4NF s ON ps.id_supplier = s.id_supplier
-> ORDER BY p.id_produk, s.id_supplier;
+-----+-----+-----+-----+-----+
| id_produk | nama_produk | kategori | id_supplier | nama_supplier |
+-----+-----+-----+-----+-----+
| 1 | Kemeja Flanel Pria | Kemeja | 1 | CV. Fashion Indonesia |
| 1 | Kemeja Flanel Pria | Kemeja | 2 | PT. Textile Mandiri |
| 2 | Blouse Wanita Lengan Pendek | Blouse | 1 | CV. Fashion Indonesia |
| 2 | Blouse Wanita Lengan Pendek | Blouse | 3 | UD. Busana Murah |
| 3 | Celana Jeans Slim Fit | Celana | 2 | PT. Textile Mandiri |
| 3 | Celana Jeans Slim Fit | Celana | 4 | CV. Main Berkualitas |
+-----+-----+-----+-----+-----+
6 rows in set (0.016 sec)
```

5NF (Fifth Normal Form) / PJ/NF (Project-Join Normal Form)

### Konsep 5NF

- Tidak ada join dependency yang tidak trivial\*
- Setiap join dependency ditentukan oleh candidate keys
- Menangani hubungan ternary (tiga arah) atau lebih
- Mencegah redundansi dalam hubungan many-to-many-to-many
- Dekomposisi tanpa kehilangan informasi (lossless-join)

### Syarat 5NF

1. Sudah memenuhi 4NF (Fourth Normal Form)
2. Tidak ada join dependency kecuali yang melibatkan candidate keys

3. Setiap dekomposisi harus mempertahankan semua informasi asli
4. Dapat merekonstruksi data semula melalui natural join

#### Karakteristik 5NF

1. Menangani hubungan ternary dan higher-order relationships
2. Mengeliminasi redundansi dalam hubungan many-to-many-to-many
3. Menggunakan multiple binary relationships 代替 ternary relationships
4. Memastikan konsistensi data pada hubungan kompleks

#### Keuntungan 5NF

- Menghilangkan redundansi dalam hubungan ternary
- Fleksibilitas dalam maintenance relationship kompleks
- Konsistensi data pada hubungan many-to-many-to-many
- Query lebih efisien untuk relationship spesifik
- Eliminasi semua bentuk join dependency yang tidak perlu

#### Contoh Penerapan 5NF

1. Sistem Pendidikan

Hubungan: Mahasiswa - Mata Kuliah - Dosen Pengajar

Dekomposisi

- Mahasiswa-MataKuliah
- MataKuliah-Dosen
- Dosen-Mahasiswa

## 2. Manajemen Project

- Hubungan\*\*: Developer - Skill - Project
- Dekomposisi
- Developer-Skill
- Developer-Project
- Skill-Project

## 3. \*\*Sistem Retail (Toko Baju)

- Hubungan\*\*: Produk - Supplier - Lokasi Toko
- Dekomposisi
- Produk-Supplier
- Produk-Lokasi
- Supplier-Lokasi

Implementasi 5NF pada Toko Baju "FashionKu"

## Perbedaan 4NF vs 5NF

4NF:

Fokus: Multi-valued dependencies

Masalah: Satu entitas dengan multiple independent attributes

- Contoh: Produk dengan multiple supplier DAN multiple warna

5NF:

- Fokus: Join dependencies
- Masalah: Multiple entities dengan relationship kompleks
- Contoh: Produk-Supplier-Lokasi (tiga entitas saling terkait)

Kapan Membutuhkan 5NF?

Butuh 5NF ketika:

- Ada hubungan ternary (tiga arah) atau lebih
- Terdapat redundansi dalam hubungan many-to-many-to-many
- Update anomaly terjadi pada relationship kompleks
- Data perlu dianalisis dari multiple perspectives

Tidak Perlu 5NF ketika:

- Hanya hubungan binary (dua arah)
- Tidak ada join dependency yang kompleks
- Aplikasi sederhana tanpa relationship ternary

Kesimpulan 5NF

Database FashionKu dapat mencapai 5NF ketika:

- Ada kebutuhan relationship kompleks tiga arah atau lebih Terdapat join dependency yang perlu dieliminasi

Diperhatikan optimalisasi untuk query relationship spesifik

5NF adalah bentuk normalisasi tertinggi yang praktis untuk aplikasi dengan relationship kompleks, namun untuk kebutuhan bisnis standar, 4NF seringkali sudah cukup.

Hasil: Database dapat mencapai 5NF ketika diperlukan untuk relationship kompleks

```

mysql> SELECT * FROM Produk_5NF;
+-----+-----+
| id_produk | nama_produk |
+-----+-----+
| 1 | Kemeja Flanel Pria |
| 2 | Blouse Wanita Lengan Pendek |
| 3 | Celana Jeans Slim Fit |
| 4 | Dress Casual Muslimah |
| 5 | Kaos Polo Shirt |
+-----+
5 rows in set (0.006 sec)

mysql> SELECT * FROM Supplier_5NF;
+-----+-----+
| id_supplier | nama_supplier |
+-----+-----+
| 1 | CV. Fashion Indonesia |
| 2 | PT. Textile Mandiri |
| 3 | UD. Busana Murah |
| 4 | CV. Kain Berkualitas |
+-----+
4 rows in set (0.006 sec)

```

## DAFTAR PUSTAKA

1. Kroenke, David M.  
*Database Processing: Fundamentals, Design, and Implementation.* Pearson Education, 2020.
2. Hoffer, Jeffrey A., Venkataraman, Ramesh, & Topi, Heikki.  
*Modern Database Management.* Pearson, 2016.
3. Connolly, Thomas & Begg, Carolyn.  
*Database Systems: A Practical Approach to Design, Implementation, and Management.* Pearson, 2015.
4. Pressman, Roger S.  
*Software Engineering: A Practitioner's Approach.* McGraw-Hill, 2019.
5. Elmasri, Ramez & Navathe, Shamkant B.  
*Fundamentals of Database Systems.* Addison-Wesley, 2016.
6. Kadir, Abdul.  
*Dasar Pemrograman Basis Data.* ANDI Offset, Yogyakarta, 2015.

7. Jogiyanto, Hartono.  
*Analisis dan Desain Sistem Informasi*. ANDI Offset,  
2017.
8. Dokumentasi MySQL Resmi.  
<https://dev.mysql.com/doc/>
9. Sumber internal: Laporan Proyek Akhir Basis Data – FashionKu (2024).