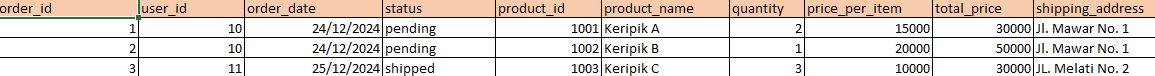
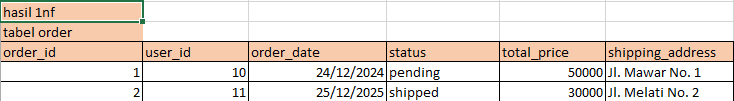
**Laporan Implementasi Database**

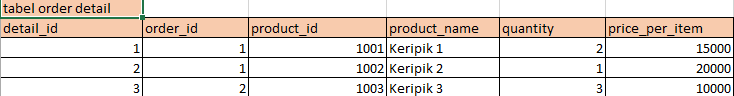
**Aplikasi Penjualan Keripik**

Tabel Awal

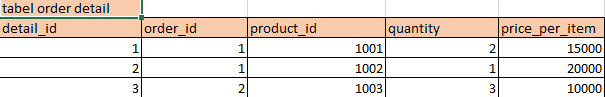


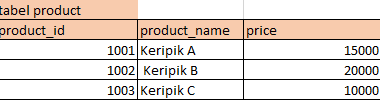
Tabel 1NF



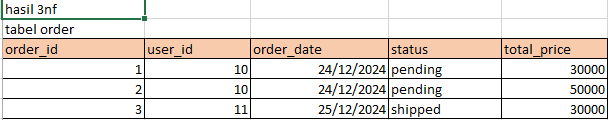


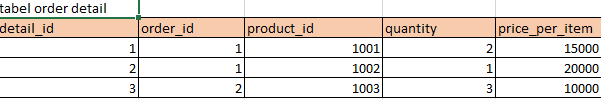
Tabel 2nF

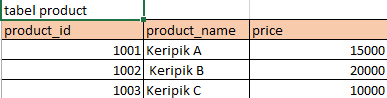


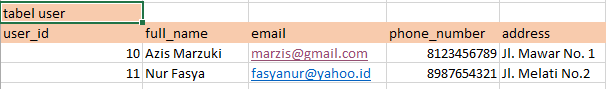


Tabel 3nF









**Struktur Tabel**

1. Users: Menyimpan data pelanggan
2. Products: Menyimpan data produk
3. Orders: Menyimpan data transaksi utama
4. Order\_Details: Menyimpan detail produk dalam setiap transaksi

**Skema Tabel**

Tabel user

CREATE TABLE users (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

phone\_number VARCHAR(20),

address TEXT

);

Tabel products

CREATE TABLE products (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100) NOT NULL,

price DECIMAL(10, 2) NOT NULL

);

**Tabel orders**

CREATE TABLE orders (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

order\_date DATETIME NOT NULL,

status ENUM('pending', 'paid', 'shipped', 'completed', 'canceled') NOT NULL,

total\_price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

Tabel order details

CREATE TABLE order\_details (

detail\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT NOT NULL,

product\_id INT NOT NULL,

quantity INT NOT NULL,

price\_per\_item DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

Menampilkan Detail Pesanan dengan Informasi Pelanggan

SELECT

orders.order\_id,

orders.order\_date,

users.full\_name AS customer\_name,

users.email AS customer\_email,

users.address AS shipping\_address,

orders.status,

orders.total\_price

FROM orders

JOIN users ON orders.user\_id = users.user\_id;

Menampilkan Detail Produk dalam Pesanan

SELECT

orders.order\_id,

products.product\_name,

order\_details.quantity,

order\_details.price\_per\_item,

(order\_details.quantity \* order\_details.price\_per\_item) AS total\_price\_per\_product

FROM order\_details

JOIN products ON order\_details.product\_id = products.product\_id

JOIN orders ON order\_details.order\_id = orders.order\_id;

Pelanggan dengan Total Pembelian Tertinggi

SELECT

users.user\_id,

users.full\_name,

users.email,

total\_spent.total\_price AS total\_spent

FROM users

JOIN (

SELECT

user\_id,

SUM(total\_price) AS total\_price

FROM orders

GROUP BY user\_id

) AS total\_spent ON users.user\_id = total\_spent.user\_id

ORDER BY total\_spent.total\_price DESC

LIMIT 1;

Produk yang Belum Pernah Dibeli

SELECT

products.product\_id,

products.product\_name,

products.price

FROM products

WHERE product\_id NOT IN (

SELECT DISTINCT product\_id

FROM order\_details

);

Menghitung Total Harga Pesanan

DELIMITER $$

CREATE FUNCTION calculate\_order\_total(orderID INT)

RETURNS DECIMAL(10, 2)

DETERMINISTIC

BEGIN

DECLARE total DECIMAL(10, 2);

SELECT SUM(quantity \* price\_per\_item) INTO total

FROM order\_details

WHERE order\_id = orderID;

RETURN total;

END $$

DELIMITER ;

Mengupdate Total Harga Pesanan

DELIMITER $$

CREATE TRIGGER update\_order\_total

AFTER INSERT ON order\_details

FOR EACH ROW

BEGIN

UPDATE orders

SET total\_price = (SELECT calculate\_order\_total(NEW.order\_id))

WHERE order\_id = NEW.order\_id;

END $$

DELIMITER ;

Menghapus Pesanan yang Tidak Dibayar dalam 7 Hari

DELIMITER $$

CREATE EVENT delete\_old\_pending\_orders

ON SCHEDULE EVERY 1 DAY

DO

BEGIN

DELETE FROM orders

WHERE status = 'pending' AND order\_date < NOW() - INTERVAL 7 DAY;

END $$

DELIMITER ;

Kesimpulan

Desain database ini mencakup kebutuhan untuk mendukung aplikasi penjualan online, mulai dari pengelolaan data pelanggan, produk, pesanan, hingga detail transaksi. Fungsi, trigger, dan event yang disediakan membantu dalam otomasi perhitungan dan pemeliharaan data, sehingga meningkatkan efisiensi dan konsistensi sistem..