

**LAPORAN TUGAS BESAR
PROYEK BASIS DATA LANJUT**



Disusun Oleh :

- | | |
|--------------------------------|-------------|
| 1. Najwa nabilah Wibisono | (G1A023065) |
| 2. Abim Bintang Audio | (G1A023073) |
| 3. Khaylilla Shafaraly Irnanda | (GA1023079) |
| 4. Ajis saputra Hidayah | (G1A023083) |

Nama Asisten Dosen :

- | | |
|--------------------------------|-------------|
| 1. Merly Yuni Purnama | (G1A022006) |
| 2. Reksi Hendra Pratama | (G1A022032) |
| 3. Sinta Ezra Wati Gulo | (G1A022040) |
| 4. Fadlan Dwi Febrio | (G1A022051) |
| 5. Torang Four Yones Manullang | (G1A022052) |
| 6. Wahyu Ozorah Manurung | (G1A022060) |
| 7. Shalaudin Muhammad Sah | (G1A022070) |
| 8. Dian Ardiyanti Saputri | (G1A022084) |

Dosen Pengampu :

- | |
|---|
| 1. Dr. Endina Putri Purwandari, Dr.,S.T.,M.Kom. |
| 2. Ir.Tiara Eka Putri, S.T., M.Kom. |

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU**

2025

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga penulis bersama tim dapat menyelesaikan laporan *Tugas Besar Proyek Basis Data Lanjut* ini dengan baik dan tepat waktu. Laporan ini disusun sebagai salah satu syarat untuk memenuhi mata kuliah *Proyek Basis Data Lanjut* pada Program Studi Informatika, Fakultas Teknik, Universitas Bengkulu.

Laporan ini berisi perancangan dan implementasi sistem basis data untuk platform e-commerce, yang mencakup pembuatan struktur tabel, penggunaan relasi antar entitas, hingga penerapan *query* SQL tingkat lanjut seperti *JOIN*, *stored procedure*, dan *subquery*. Melalui proyek ini, penulis dan tim memperoleh pengalaman praktis dalam membangun sistem basis data yang terstruktur, efisien, dan mendukung kebutuhan operasional sebuah sistem digital secara nyata.

Ucapan terima kasih penulis sampaikan kepada:

- Ibu Dr. Endina Putri Purwandari, Dr., S.T., M.Kom. dan Ibu Ir. Tiara Eka Putri, S.T., M.Kom., selaku dosen pengampu mata kuliah, atas ilmu dan bimbingan yang diberikan selama perkuliahan berlangsung.
- Para asisten dosen, khususnya Bang Wahyu Ozorah Manurung dan Bang Torang Four Yones Manullang, atas arahan dan bantuan teknis selama praktikum.
- Serta semua pihak yang telah memberikan bantuan, baik secara langsung maupun tidak langsung, dalam proses penyusunan laporan ini.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa yang akan datang

Bengkulu, 08 Mei 2025

Penulis

DAFTAR ISI

DAFTAR ISI	2
DAFTAR GAMBAR	Error! Bookmark not defined.
DAFTAR TABEL	Error! Bookmark not defined.
BAB I	7
PENDAHULUAN	7
1.1 Latar belakang	7
1.2 Rumusan Masalah	8
1.3 Tujuan Proyek	8
BAB II	9
LANDASAN TEORI	9
BAB IV	Error! Bookmark not defined.
PEMBAHASAN	11
BAB V	61
KESIMPULAN DAN SARAN	61
A. KESIMPULAN	61
B. SARAN	61
DAFTAR PUSTAKA	62

DAFTAR GAMBAR

Gambar 4.1 ERD E-commerce	11
Gambar 4.2 Source Code Create Database dan Tabel	12
Gambar 4.3 Source Code Create Tabel Product dan Cart.....	12
Gambar 4.4 Source Code Create Tabel Review.....	12
Gambar 4.5 Source Code Create Tabel Payment dan Shipping	13
Gambar 4.6 Source Code Create Tabel Orders dan Orders_Item	13
Gambar 4.7 Output Create Database dan Tabel.....	17
Gambar 4.8 Menambahkan Data Pada Tabel User dan Category.....	18
Gambar 4.9 Menambahkan Data Pada Tabel Produk Dan Cart.....	18
Gambar 4.10 Menambahkan Data Pada Tabel Review	18
Gambar 4.11 Menambahkan Data Pada Tabel Payment dan Shipping	19
Gambar 4. 12 Menambah Data Pada Tabel Orders Dan Order_Item	19
Gambar 4.13 Output Tabel User	24
Gambar 4.14 Output Tabel Category	24
Gambar 4.15 Output Tabel Order	24
Gambar 4.16 Output Tabel Order_Item	25
Gambar 4.17 Output Tabel Payment.....	25
Gambar 4.18 Output Tabel Shipping	25
Gambar 4.19 Output Tabel Review	25
Gambar 4. 20 Source Code Inner Join	26
Gambar 4.21 Output Inner Join.....	27
Gambar 4.22 Source Code Left Join	27
Gambar 4.23 Output Left Join	28
Gambar 4.24 Source Code Right Join.....	28
Gambar 4.25 Output Right Join	29
Gambar 4.26 Source Code Full Join	29
Gambar 4.27 Output Full Join.....	30
Gambar 4.28 Sorce Code Alias dan Operator	30
Gambar 4.29 Output AS dan Operator.....	31
Gambar 4. 30 Source Code AS dan Operator	31
Gambar 4.31 Output AS dan Operator.....	32

Gambar 4.32 Source Code AS dan Operator	32
Gambar 4.33 Output AS dan Operator.....	33
Gambar 4. 34 Source Code AS dan Operator	34
Gambar 4.35 Output AS dan Operator.....	34
Gambar 4.36 Source Code Alias dan Operator.....	35
Gambar 4. 37 Output AS dan Operator.....	35
Gambar 4.38 Operator AS dan Operator.....	36
Gambar 4. 39 Output AS dan Operator.....	36
Gambar 4.40 Source Code AS dan Operator	37
Gambar 4.41 Output AS dan Operator.....	38
Gambar 4.42 Output AS dan Operator.....	39
Gambar 4.43 Source Code AS dan Operator	39
Gambar 4.44 Output AS dan Operator.....	40
Gambar 4.45 Source Code AS dan Operator	40
Gambar 4.46 Source Code AS dan Operator	41
Gambar 4.47 Source Code Grouping	41
Gambar 4.48 Output Grouping	42
Gambar 4.49 Source Code Grouping	43
Gambar 4. 50 Source Code Grouping	44
Gambar 4.51 Source Code Grouping	44
Gambar 4.52 Output Grouping	45
Gambar 4.53 Source Code Grouping	46
Gambar 4.54 Output Grouping	47
Gambar 4.55 Source Code Grouping	47
Gambar 4.56 Ouput Grouping	48
Gambar 4. 57 Source Code Stored Procedure.....	49
Gambar 4.58 Output Stored Procedure	50
Gambar 4.59 Source Code Stored Procedure.....	51
Gambar 4.60 Output Stored Procedure	52
Gambar 4.61 Source Nested.....	53
Gambar 4.62 Output Nested.....	54
Gambar 4.63 Source Code Nested	54

Gambar 4.64 Output Nested.....	55
Gambar 4.65 Trigger Cek Stok	56
Gambar 4.66 Trigger Pengurangan Stok.....	57
Gambar 4. 67 Output Pengurangan Stok	58
Gambar 4.68 Source Kode DCL	59
Gambar 4.69 Output Grant.....	60

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga penulis bersama tim dapat menyelesaikan laporan *Tugas Besar Proyek Basis Data Lanjut* ini dengan baik dan tepat waktu. Laporan ini disusun sebagai salah satu syarat untuk memenuhi mata kuliah Proyek Basis Data Lanjut pada Program Studi Informatika, Fakultas Teknik, Universitas Bengkulu.

Laporan ini berisi perancangan dan implementasi sistem basis data untuk platform *e-commerce*, yang mencakup pembuatan struktur tabel, penggunaan relasi antar entitas, hingga penerapan *query* SQL tingkat lanjut seperti *JOIN*, *stored procedure*, dan *subquery*. Melalui proyek ini, penulis dan tim memperoleh pengalaman praktis dalam membangun sistem basis data yang terstruktur, efisien, dan mendukung kebutuhan operasional sebuah sistem digital secara nyata. Ucapan terima kasih penulis sampaikan kepada:

- Ibu Dr. Endina Putri Purwandari, Dr., S.T., M.Kom. dan Ibu Ir. Tiara Eka Putri, S.T., M.Kom., selaku dosen pengampu mata kuliah, atas ilmu dan bimbingan yang diberikan selama perkuliahan berlangsung.
- Para asisten dosen, khususnya Abang Wahyu Ozorah Manurung dan Abang Torang Four Yones Manullang, atas arahan dan bantuan selama praktikum.
- Serta semua pihak yang telah memberikan bantuan, baik secara langsung maupun tidak langsung, dalam proses penyusunan laporan ini.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa yang akan datang

Bengkulu, 08 Mei 2025

Penulis

BAB I

PENDAHULUAN

1.1 Latar belakang

MySQL adalah sistem manajemen basis data relasional (RDBMS) yang pertama kali dikembangkan pada tahun 1994 oleh Michael Widenius (Monty), David Axmark, dan perusahaan Swedia bernama MySQL AB. Awalnya, mereka menggunakan sistem database mSQL (Mini SQL), namun karena keterbatasannya, mereka menciptakan MySQL sebagai solusi yang lebih cepat dan efisien, terutama untuk aplikasi web. Versi pertama MySQL dirilis pada tahun 1995 dan dengan cepat mendapat popularitas karena sifatnya yang open-source, ringan, dan sangat cocok digunakan bersama PHP dan Apache dalam stack LAMP (Linux, Apache, MySQL, PHP/Perl/Python). Pada periode 2000 hingga 2008, MySQL mengalami banyak perkembangan, seperti dukungan untuk mesin penyimpanan InnoDB, fitur transaksi, *foreign key*, *view*, *trigger*, dan *stored procedure*.

Pada tahun 2008, perusahaan MySQL AB diakuisisi oleh Sun Microsystems, dan dua tahun kemudian, Sun sendiri diakuisisi oleh Oracle Corporation. Hal ini menimbulkan kekhawatiran di kalangan komunitas *open-source* tentang masa depan MySQL, yang mendorong lahirnya proyek fork seperti MariaDB oleh Monty dan Percona Server oleh Percona. Meski begitu, Oracle tetap melanjutkan pengembangan MySQL dan terus merilis versi-versi terbaru dengan peningkatan fitur dan performa. Versi MySQL 8.0 yang dirilis pada tahun 2018 menjadi tonggak penting dengan hadirnya fitur modern seperti dukungan JSON, window functions, dan *Common Table Expressions* (CTE).

Perkembangan teknologi informasi yang pesat telah mendorong digitalisasi dalam berbagai aspek kehidupan, termasuk dalam bidang perdagangan. Salah satu bentuk nyata dari digitalisasi tersebut adalah hadirnya platform *e-commerce* yang memungkinkan proses jual beli dilakukan secara daring dengan lebih cepat, praktis, dan fleksibel. Dalam mendukung operasional sebuah sistem *e-commerce*, dibutuhkan perancangan basis data yang handal dan terstruktur agar seluruh proses, mulai dari pengelolaan data pengguna, produk, transaksi, hingga pengiriman, dapat berjalan dengan lancar dan efisien. Basis data berperan penting sebagai inti dari sistem karena menyimpan seluruh informasi yang dibutuhkan.

Namun, merancang struktur basis data yang baik tidak bisa dilakukan secara sembarangan. Diperlukan pendekatan sistematis melalui analisis kebutuhan, perancangan entitas dan relasi, hingga implementasi struktur tabel yang saling terhubung. Oleh karena itu, dalam proyek ini dilakukan perancangan dan implementasi basis data *e-commerce* dengan tujuan untuk mengelola seluruh data yang berkaitan dengan pengguna, produk, kategori, keranjang belanja, pesanan, pembayaran, pengiriman, dan ulasan pelanggan secara terintegrasi. Dengan struktur database yang baik, sistem diharapkan mampu menjaga integritas data, meningkatkan efisiensi transaksi, serta memberikan pengalaman yang lebih baik bagi pengguna.

1.2 Rumusan Masalah

1. Bagaimana merancang struktur basis data yang mampu mengelola data pengguna, produk, kategori, keranjang belanja, dan pesanan secara efisien dan terintegrasi?
2. Bagaimana mengatur relasi antar tabel seperti antara pengguna, produk, dan transaksi agar data tetap konsisten dan mendukung integritas referensial?
3. Bagaimana mengelola proses transaksi pembelian, mulai dari keranjang belanja, pemesanan, pembayaran, hingga pengiriman, menggunakan sistem basis data yang dirancang?
4. Bagaimana merancang sistem *review* produk oleh pengguna tanpa merusak struktur data utama?
5. Bagaimana menyimpan dan mengelompokkan data produk berdasarkan kategori agar mudah dicari dan ditampilkan pada aplikasi pengguna?

1.3 Tujuan Proyek

Tujuan dari proyek ini adalah untuk merancang dan membangun sistem basis data yang dapat mendukung pengelolaan data dalam sebuah platform *e-commerce* secara efisien dan terstruktur. Sistem ini dirancang agar mampu menangani berbagai komponen penting, seperti pengelolaan data pengguna, produk, kategori produk, keranjang belanja, proses pemesanan, pembayaran, pengiriman, serta ulasan dari pelanggan. Dengan basis data yang terintegrasi dan relasi antar tabel yang dirancang secara tepat, memastikan bahwa setiap proses transaksi dari awal hingga akhir dapat berjalan dengan lancar, aman, dan konsisten.

BAB II

LANDASAN TEORI

MySQL adalah sebuah *Database Management System* (DBMS) populer yang memiliki fungsi sebagai *Relational Database Management System* (RDBMS). Selain itu MySQL software merupakan suatu aplikasi yang sifatnya *open source* serta server basis data MySQL memiliki kinerja sangat cepat, *reliable*, dan mudah untuk digunakan serta bekerja dengan arsitektur *client server* atau *embedded systems*. Tujuan utamanya adalah untuk menyusun data dengan cara yang memudahkan dan mempercepat akses bagi para manajer guna mengambil keputusan. Tujuan dari sistem manajemen basis data (*Database Management System-DBMS*) adalah untuk menyediakan lingkungan yang efisien, andal, aman, dan mudah diakses untuk menyimpan, mengelola, dan mengambil data secara efisien.

Bahasa SQL tersusun atas 3 kelompok pernyataan berdasarkan fungsi dari pernyataan tersebut yaitu. *Data Definition Language* (DDL) Mendefinisikan jenis data yang akan dibuat (dapat berupa angka atau huruf), cara relasi data, validasi data dan lainnya. *Data Manipulation Language* (DML) Data yang telah dibuat dan didefinisikan tersebut akan dilakukan beberapa pengerjaan, seperti menyaring data, melakukan proses *query*. *Data Control Language* (DCL) Bagian ini berkenaan dengan cara mengendalikan data, seperti siapa saja yang bisa melihat isi data, bagaimana data bisa digunakan oleh banyak user.

Perkembangan dunia teknologi informasi memang sedang bergairah pesat, tak terkecuali perkembangan di cabang-cabang ilmu dari teknologi informasi lainnya, salah satunya adalah teknologi sistem basis data. Perkembangan tentang teknologi database saat ini, tentu tidak lepas dari teknologi database yang diciptakan oleh MySQL, karena sampai saat ini MySQL masih merupakan teknologi database yang sangat populer.

ERD (*Entity Relationship Diagram*) adalah sebuah diagram yang digunakan untuk menggambarkan struktur logis dari sebuah sistem basis data dengan menampilkan entitas, atribut, dan hubungan antar entitas tersebut. Entitas dalam ERD mewakili objek nyata atau konsep penting yang memiliki data yang disimpan, seperti pengguna, produk, atau pesanan. Setiap entitas memiliki atribut, yaitu data

atau informasi yang menjelaskan karakteristik dari entitas tersebut. Selain itu, ERD juga menunjukkan relasi antar entitas, seperti hubungan antara pengguna dengan pesanan atau produk dengan kategori. Dengan menggunakan ERD, perancang sistem dapat memahami bagaimana data saling terkait dan merancang struktur database yang efisien.

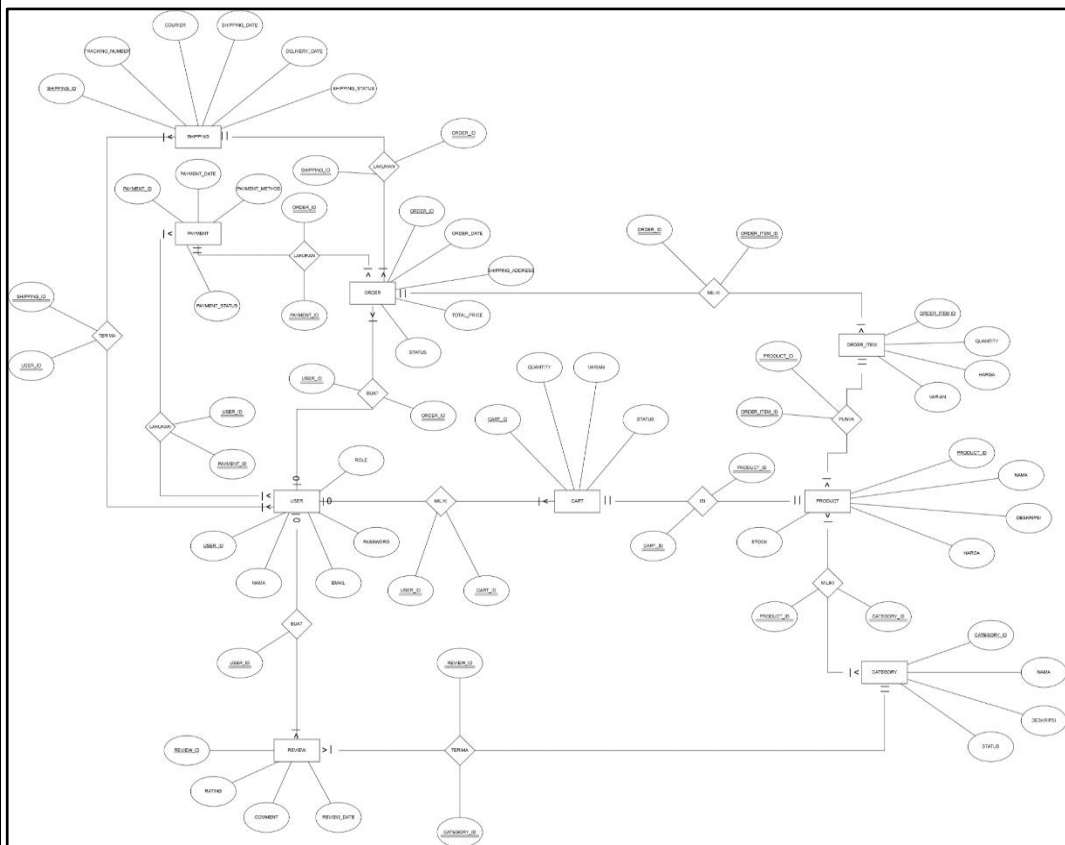
Dalam SQL, JOIN digunakan untuk menggabungkan data dari dua atau lebih tabel berdasarkan kolom yang saling berhubungan. Terdapat beberapa jenis JOIN, seperti INNER JOIN yang hanya mengambil data yang cocok di kedua tabel, LEFT JOIN yang mengambil semua data dari tabel kiri dan yang cocok dari tabel kanan, serta RIGHT JOIN yang merupakan kebalikannya. Alias digunakan untuk memberi nama sementara pada tabel atau kolom agar penulisan *query* lebih ringkas dan mudah dibaca, biasanya dengan kata kunci AS. Operator dalam SQL seperti =, <, >, AND, OR, dan LIKE digunakan untuk membandingkan nilai dan membuat kondisi logika dalam klausa WHERE. Fungsi (function) seperti COUNT(), SUM(), AVG(), MIN(), dan MAX() digunakan untuk melakukan perhitungan pada data, baik kelompok maupun keseluruhan data.

Untuk mengelompokkan data berdasarkan nilai tertentu, digunakan GROUP BY, biasanya dikombinasikan dengan fungsi agregat seperti COUNT atau SUM. Sorting atau pengurutan data dilakukan dengan perintah ORDER BY, baik secara naik (ASC) maupun turun (DESC). VIEW adalah tabel virtual yang dibuat berdasarkan *query* tertentu; meskipun bukan tabel fisik, view bisa digunakan seperti tabel biasa untuk menyederhanakan *query* yang kompleks. Stored Procedure adalah kumpulan perintah SQL yang disimpan di database dan dapat dipanggil berulang kali, sering digunakan untuk mengotomatisasi proses tertentu. Nested *Query* atau subquery adalah *query* yang ditulis di dalam *query* lain, biasanya dalam klausa SELECT, WHERE, atau FROM, untuk mengambil data berdasarkan hasil *query* lain. Trigger adalah prosedur otomatis yang dijalankan saat terjadi aksi tertentu dalam tabel, seperti INSERT, UPDATE, atau DELETE, dan berguna untuk menjaga konsistensi data. Terakhir, DCL (*Data Control Language*) mencakup perintah seperti GRANT dan REVOKE, yang digunakan untuk mengatur hak akses pengguna terhadap objek-objek di dalam database.

BAB IV PEMBAHASAN

Pada proyek ini kami membuat sebuah database *e-commerce* untuk membantu mengelola dan menyimpan berbagai informasi penting yang berkaitan dengan operasional toko *online* secara terstruktur dan efisien. Dalam sebuah platform *e-commerce*, terdapat banyak data yang harus diatur, seperti data produk, pelanggan, transaksi, pengiriman, dan pembayaran. Dengan adanya database, informasi produk seperti nama, harga, deskripsi, stok, dan gambar dapat disimpan dengan rapi sehingga memudahkan dalam pengelolaan dan pencarian data. Berikut ini merupakan penjelasan masing-masing kode yang telah kami buat.

1. ERD E-Commerce



Gambar 4.1 ERD *E-commerce*

2. Membuat database dan tabel

Printscreen Source Code :

```

create database ecommerce;
use ecommerce;

CREATE TABLE User (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    password VARCHAR(100),
    role VARCHAR(20)
);

CREATE TABLE Category (
    category_id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100),
    deskripsi TEXT,
    status VARCHAR(20)
);

```

Gambar 4.2 *Source Code Create Database dan Tabel*

```

CREATE TABLE Product (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100),
    deskripsi TEXT,
    harga INT,
    stock INT,
    category_id INT,
    FOREIGN KEY (category_id) REFERENCES Category(category_id)
);

CREATE TABLE Cart (
    cart_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    product_id INT,
    quantity INT,
    varian VARCHAR(50),
    status VARCHAR(20),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
);

```

Gambar 4.3 *Source Code Create Tabel Product dan Cart*

```

CREATE TABLE Review (
    review_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    category_id INT,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    comment TEXT,
    review_date DATE,
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (category_id) REFERENCES Category(category_id)
);

```

Gambar 4.4 *Source Code Create Tabel Review*

```

CREATE TABLE Payment (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    user_id INT,
    payment_method VARCHAR(50),
    payment_date DATE,
    payment_status VARCHAR(20),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);

CREATE TABLE Shipping (
    shipping_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    user_id INT,
    courier VARCHAR(50),
    tracking_number VARCHAR(100),
    shipping_status VARCHAR(50),
    shipping_date DATE,
    delivery_date DATE,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);

```

Gambar 4.5 Source Code Create Tabel Payment dan Shipping

```

CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    total_price INT,
    status VARCHAR(20),
    order_date DATE,
    shipping_address VARCHAR(50),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);

CREATE TABLE Order_Item (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    harga INT,
    varian VARCHAR(50),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Product(product_id)
);

```

Gambar 4.6 Source Code Create Tabel Orders dan Orders_Item

Source Code :

```

create database e-commerce;
use e-commerce;

```

```

CREATE TABLE User (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100),
    email VARCHAR(100) UNIQUE,

```

```

password VARCHAR(100),
role VARCHAR(20)
);
CREATE TABLE Category (
category_id INT AUTO_INCREMENT PRIMARY KEY,
nama VARCHAR(100),
deskripsi TEXT,
status VARCHAR(20)
);
CREATE TABLE Product (
product_id INT AUTO_INCREMENT PRIMARY KEY,
nama VARCHAR(100),
deskripsi TEXT,
harga INT,
stock INT,
category_id INT,
FOREIGN KEY (category_id) REFERENCES Category(category_id)
);
CREATE TABLE Cart (
cart_id INT AUTO_INCREMENT PRIMARY KEY,
user_id INT,
product_id INT,
quantity INT,
varian VARCHAR(50),
status VARCHAR(20),
FOREIGN KEY (user_id) REFERENCES User(user_id),
FOREIGN KEY (product_id) REFERENCES Product(product_id)
);
CREATE TABLE Orders (
order_id INT AUTO_INCREMENT PRIMARY KEY,
user_id INT,
total_price INT,

```

```

status VARCHAR(20),
order_date DATE,
shipping_address VARCHAR(50),
FOREIGN KEY (user_id) REFERENCES User(user_id)
);
CREATE TABLE Order_Item (
order_item_id INT AUTO_INCREMENT PRIMARY KEY,
order_id INT,
product_id INT,
quantity INT,
harga INT,
varian VARCHAR(50),
FOREIGN KEY (order_id) REFERENCES Orders(order_id),
FOREIGN KEY (product_id) REFERENCES Product(product_id)
);
CREATE TABLE Payment (
payment_id INT AUTO_INCREMENT PRIMARY KEY,
order_id INT,
user_id INT,
payment_method VARCHAR(50),
payment_date DATE,
payment_status VARCHAR(20),
FOREIGN KEY (order_id) REFERENCES Orders(order_id),
FOREIGN KEY (user_id) REFERENCES User(user_id)
);
CREATE TABLE Shipping (
shipping_id INT AUTO_INCREMENT PRIMARY KEY,
order_id INT,
user_id INT,
courier VARCHAR(50),
tracking_number VARCHAR(100),
shipping_status VARCHAR(50),

```



```

shipping_date DATE,
delivery_date DATE,
FOREIGN KEY (order_id) REFERENCES Orders(order_id),
FOREIGN KEY (user_id) REFERENCES User(user_id)
);

CREATE TABLE Review (
review_id INT AUTO_INCREMENT PRIMARY KEY,
user_id INT,
category_id INT,
rating INT CHECK (rating BETWEEN 1 AND 5),
comment TEXT,
review_date DATE,
FOREIGN KEY (user_id) REFERENCES User(user_id),
FOREIGN KEY (category_id) REFERENCES Category(category_id)
);

```

Pembahasan :

Struktur database e-commerce terdiri dari beberapa tabel utama yang saling terhubung dan berfungsi untuk mendukung sistem toko online secara menyeluruh. Pertama, database dibuat dengan perintah **CREATE DATABASE e-commerce**; lalu diaktifkan menggunakan **USE e-commerce**; Di dalamnya terdapat **CREATE TABLE User** yang menyimpan informasi pengguna seperti admin, penjual, dan pembeli, dengan kolom `user_id` sebagai identitas unik, serta data penting lainnya seperti nama, email (unik), password, dan peran (role) pengguna dalam sistem. Selanjutnya, **CREATE TABLE Category** digunakan untuk mengelompokkan produk ke dalam kategori tertentu, agar lebih mudah dicari dan dikelola. Setiap kategori memiliki `category_id`, nama, deskripsi, dan status aktif/tidak aktif. Lalu terdapat kode **CREATE TABLE Product** menyimpan data produk yang dijual, dengan kolom `product_id`, nama produk, deskripsi, harga, stok, dan `category_id` sebagai *foreign key* untuk menghubungkan produk ke kategori yang sesuai. Lalu ada **CREATE TABLE Cart** yang berisi informasi keranjang belanja milik pengguna, mencakup

cart_id, user_id, product_id, jumlah (quantity), varian, dan status item di keranjang. **CREATE TABLE User Orders** mencatat pesanan pengguna dengan kolom order_id, user_id, total harga, status pesanan, tanggal pemesanan, dan alamat pengiriman. Lalu terdapat kode untuk membuat tabel order item **CREATE TABLE Order_Item** yang berisi order_item_id, order_id, product_id, jumlah produk, harga satuan saat dipesan, dan varian jika ada. Selanjutnya terdapat kode untuk membuat tabel pembayaran **CREATE TABLE Payment**, mencakup payment_id, order_id, user_id, metode pembayaran, tanggal pembayaran, dan status pembayaran seperti berhasil, gagal, atau menunggu. Untuk membuat tabel pengiriman digunakan kode **CREATE TABLE Shipping**, dengan kolom shipping_id, order_id, user_id, nama ekspedisi, nomor resi, status pengiriman, tanggal pengiriman, dan tanggal barang sampai. Terakhir terdapat kode **CREATE TABLE Review** untuk menyimpan ulasan dari pengguna, dengan review_id, user_id, dan category_id. Ulasan ini bersifat umum terhadap kategori, bukan produk spesifik. Sehingga sistem *e-commerce* data pengguna, produk, pesanan, pembayaran, pengiriman, dan ulasan.

Printscreen Output :

	Tables_in_ecommerce
►	cart
	category
	order_item
	orders
	payment
	product
	review
	shipping
	user

Gambar 4.7 Output *Create Database dan Tabel*

Pembahasan :

Output dari perintah SQL tersebut adalah sebuah database e-commerce yang memiliki struktur lengkap untuk mengelola sistem toko online. Database ini mencakup beberapa tabel penting seperti User, Category, Product, Cart, Orders, Order_Item, Payment, Shipping, dan Review. Masing-masing tabel memiliki kolom dan relasi yang dirancang untuk menyimpan serta mengatur data pengguna, produk, pesanan, pembayaran, pengiriman, hingga ulasan. Dengan struktur ini, sistem *e-commerce* dapat berjalan secara terintegrasi, dan menjaga konsistensi data.

3. Menambahkan data pada tabel

Printscreen Source Code :

```
INSERT INTO User (nama, email, password, role) VALUES
('Andi Saputra', 'andisaputra@gmail.com', 'password1', 'admin'),
('Budi Santoso', 'budisantoso@gmail.com', 'password2', 'customer'),
('Citra Lestari', 'citralestari@gmail.com', 'password3', 'customer'),
('Dewi Maharani', 'dewimaharani@gmail.com', 'password4', 'customer'),
('Eko Prasetyo', 'ekoprasetyo@gmail.com', 'password5', 'customer'),
('Fitriani Ayu', 'fitrianiayu@gmail.com', 'password6', 'customer'),
('Gilang Ramadhan', 'gilangramadhan@gmail.com', 'password7', 'customer'),
('Hesti Wulandari', 'hestiwulandari@gmail.com', 'password8', 'customer'),
('Imam Hidayat', 'imamhidayat@gmail.com', 'password9', 'customer'),
('Joko Suharto', 'jokosuharto@gmail.com', 'password10', 'customer');

INSERT INTO Category (nama, deskripsi, status) VALUES
('Elektronik', 'Perangkat dan gadget seperti ponsel, laptop, dan aksesoris', 'tersedia'),
('Pakaian', 'Pakaian untuk pria, wanita, dan anak-anak', 'tersedia'),
('Rumah & Dapur', 'Peralatan rumah tangga dan perlengkapan dapur', 'tersedia'),
('Buku', 'Fiksi, non-fiksi, edukasi, dan lainnya', 'tersedia'),
('Kecantikan', 'Produk perawatan kulit, makeup, dan perawatan diri', 'tersedia'),
('Olahraga', 'Peralatan olahraga, kebugaran, dan kegiatan luar ruangan', 'tersedia'),
('Mainan', 'Mainan dan permainan untuk segala usia', 'tersedia'),
('Otomotif', 'Aksesoris mobil dan peralatan otomotif', 'tersedia'),
('Kesehatan', 'Produk perawatan kesehatan dan suplemen', 'tersedia'),
('Perlengkapan Kantor', 'Alat tulis, elektronik kantor, dan furnitur kantor', 'tersedia');
```

Gambar 4.8 Menambahkan Data Pada Tabel User dan Category

```
INSERT INTO Product (nama, deskripsi, harga, stock, category_id) VALUES
('iPhone 14', 'Smartphone Apple dengan chip A15 Bionic dan penyimpanan 128GB', 23000000, 20, 1),
('Jaket Denim Pria', 'Jaket denim pria yang stylish dan tahan lama', 320000, 38, 2),
('Air Fryer', 'Penggoreng tanpa minyak untuk memasak sehat', 700000, 66, 3),
('The Great Gatsby', 'Novel klasik karya F. Scott Fitzgerald', 90000, 57, 4),
('Krim Pelembab', 'Krim hidrasi untuk kulit kering dan sensitif', 120000, 18, 5),
('Matras Yoga', 'Matras anti slip untuk yoga dan olahraga', 175000, 53, 6),
('Set LEGO City', 'Mainan balok bangunan kreatif untuk anak usia 6+', 350000, 37, 7),
('Vacuum Cleaner Mobil', 'Penyedot debu portabel untuk interior mobil', 450000, 64, 8),
('Tablet Vitamin C', 'Tingkatkan daya tahan tubuh dengan vitamin C harian', 75000, 45, 9),
('Kursi Kantor Ergonomis', 'Kursi untuk kenyamanan kerja', 900000, 47, 10);

INSERT INTO Cart (user_id, product_id, quantity, varian, status) VALUES
(2, 7, 1, 'Standard', 'aktif'),
(4, 8, 2, 'Hitam', 'aktif'),
(6, 9, 1, '1000mg', 'aktif'),
(8, 10, 4, 'Hitam', 'aktif'),
(10, 4, 1, 'Softcover', 'aktif'),
(1, 5, 1, '100ml', 'aktif'),
(3, 2, 1, 'XL', 'aktif'),
(5, 3, 2, '3.5L', 'aktif'),
(7, 6, 1, 'Ungu', 'aktif'),
(9, 1, 1, '128GB', 'aktif');
```

Gambar 4.9 Menambahkan Data Pada Tabel Produk Dan Cart

```
INSERT INTO Review (user_id, category_id, rating, comment, review_date) VALUES
(7, 2, 5, 'Produk sangat bagus dan berkualitas!', '2025-05-02'),
(8, 4, 4, 'Sangat nyaman dipakai, namun agak ketat.', '2025-05-02'),
(9, 6, 3, 'Cukup baik, namun saya mengharapkan lebih.', '2025-04-30'),
(10, 8, 5, 'Sangat puas dengan pengirimannya, produk sesuai deskripsi!', '2025-04-29'),
(4, 10, 4, 'Buku ini sangat menarik, tapi agak lambat pengirimannya.', '2025-05-01'),
(5, 1, 3, 'Krim pelembab ini oke, tapi kurang cocok untuk kulit saya.', '2025-05-02'),
(2, 3, 5, 'Jaket ini sangat stylish dan nyaman.', '2025-05-01'),
(3, 5, 4, 'Penggorengan tanpa minyak ini sangat memudahkan.', '2025-05-02'),
(6, 7, 4, 'Matras ini cukup tebal dan nyaman untuk yoga.', '2025-04-30'),
(1, 9, 5, 'iPhone 14 sangat cepat dan layarnya tajam!', '2025-05-02');
```

Gambar 4.10 Menambahkan Data Pada Tabel Review

```

INSERT INTO Payment (order_id, user_id, payment_method, payment_date, payment_status) VALUES
(4, 2, 'E-Wallet', '2025-05-02', 'berhasil'),
(1, 4, 'Kartu Kredit', '2025-05-02', 'berhasil'),
(10, 6, 'Transfer Bank', '2025-04-30', 'berhasil'),
(9, 8, 'E-Wallet', '2025-04-29', 'berhasil'),
(2, 10, 'Kartu Kredit', '2025-05-01', 'berhasil'),
(8, 1, 'Transfer Bank', '2025-05-02', 'gagal'),
(7, 3, 'E-Wallet', '2025-05-01', 'berhasil'),
(6, 5, 'COD', '2025-05-02', 'berhasil'),
(5, 7, 'Kartu Kredit', '2025-04-30', 'berhasil'),
(3, 9, 'E-Wallet', '2025-05-02', 'berhasil');

INSERT INTO Shipping (order_id, user_id, courier, tracking_number, shipping_status, shipping_date, delivery_date) VALUES
(6, 2, 'JNE', 'JNE234567', 'diproses', '2025-05-02', NULL),
(7, 4, 'SiCepat', 'SC890123', 'dikirim', '2025-05-02', NULL),
(8, 6, 'J&T', 'JT456789', 'sampai', '2025-04-30', '2025-05-04'),
(9, 8, 'Tiki', 'TK678901', 'sampai', '2025-04-29', '2025-05-04'),
(10, 10, 'POS Indonesia', 'POS012345', 'diproses', NULL, NULL),
(1, 1, 'JNE', 'JNE987654', 'dibatalkan', NULL, NULL),
(2, 3, 'SiCepat', 'SC345678', 'sampai', '2025-05-01', '2025-05-07'),
(3, 5, 'J&T', 'JT567890', 'dikirim', '2025-05-02', NULL),
(4, 7, 'Tiki', 'TK789012', 'sampai', '2025-04-30', '2025-05-05'),
(5, 9, 'POS Indonesia', 'POS234567', 'diproses', NULL, NULL);

```

Gambar 4.11 Menambahkan Data Pada Tabel Payment dan Shipping

```

INSERT INTO Orders (user_id, total_price, status, order_date, shipping_address) VALUES
(2, 350000, 'diproses', '2025-05-02', 'Jl. Cendana No. 3'),
(4, 450000, 'dikirim', '2025-05-02', 'Jl. Sawo No. 7'),
(6, 75000, 'selesai', '2025-04-30', 'Jl. Nangka No. 12'),
(8, 900000, 'selesai', '2025-04-29', 'Jl. Rambutan No. 20'),
(10, 90000, 'diproses', '2025-05-01', 'Jl. Pepaya No. 5'),
(1, 120000, 'dibatalkan', '2025-05-02', 'Jl. Jeruk No. 15'),
(3, 320000, 'selesai', '2025-05-01', 'Jl. Kedondong No. 8'),
(5, 700000, 'dikirim', '2025-05-02', 'Jl. Semangka No. 4'),
(7, 175000, 'selesai', '2025-04-30', 'Jl. Belimbing No. 9'),
(9, 23000000, 'diproses', '2025-05-02', 'Jl. Durian No. 18');

INSERT INTO Order_Item (order_id, product_id, quantity, harga, varian) VALUES
(2, 7, 1, 350000, 'Standard'),
(10, 8, 2, 450000, 'Hitam'),
(8, 9, 1, 75000, '1000mg'),
(3, 10, 4, 900000, 'Hitam'),
(9, 4, 1, 90000, 'Softcover'),
(5, 5, 1, 120000, '100ml'),
(7, 2, 1, 320000, 'XL'),
(4, 3, 2, 700000, '3.5L'),
(6, 6, 1, 175000, 'Ungu'),
(1, 1, 1, 23000000, '128GB');

```

Gambar 4. 12 Menambah Data Pada Tabel Orders Dan Order_Item

Source Code :

```

INSERT INTO User (nama, email, password, role) VALUES
('Andi Saputra', 'andisaputra@gmail.com', 'password1', 'admin'),
('Budi Santoso', 'budisantoso@gmail.com', 'password2', 'customer'),
('Citra Lestari', 'citralestari@gmail.com', 'password3', 'customer'),
('Dewi Maharani', 'dewimaharani@gmail.com', 'password4', 'customer'),
('Eko Prasetyo', 'ekoprasetyo@gmail.com', 'password5', 'customer'),
('Fitriani Ayu', 'fitrianiayu@gmail.com', 'password6', 'customer'),
('Gilang Ramadhan', 'gilangramadhan@gmail.com', 'password7', 'customer'),
('Hesti Wulandari', 'hestiwulandari@gmail.com', 'password8', 'customer'),
('Imam Hidayat', 'imamhidayat@gmail.com', 'password9', 'customer'),
('Joko Suharto', 'jokosuharto@gmail.com', 'password10', 'customer');

```

```
INSERT INTO Category (nama, deskripsi, status) VALUES
('Elektronik', 'Perangkat dan gadget seperti ponsel, laptop, dan aksesori',
'tersedia'),
('Pakaian', 'Pakaian untuk pria, wanita, dan anak-anak', 'tersedia'),
('Rumah & Dapur', 'Peralatan rumah tangga dan perlengkapan dapur',
'tersedia'),
('Buku', 'Fiksi, non-fiksi, edukasi, dan lainnya', 'tersedia'),
('Kecantikan', 'Produk perawatan kulit, makeup, dan perawatan diri', 'tersedia'),
('Olahraga', 'Peralatan olahraga, kebugaran, dan kegiatan luar ruangan',
'tersedia'),
('Mainan', 'Mainan dan permainan untuk segala usia', 'tersedia'),
('Otomotif', 'Aksesori mobil dan peralatan otomotif', 'tersedia'),
('Kesehatan', 'Produk perawatan kesehatan dan suplemen', 'tersedia'),
('Perlengkapan Kantor', 'Alat tulis, elektronik kantor, dan furnitur kantor',
'tersedia');
```

```
INSERT INTO Product (nama, deskripsi, harga, stock, category_id) VALUES
('iPhone 14', 'Smartphone Apple dengan chip A15 Bionic dan penyimpanan
128GB', 23000000, 20, 1),
('Jaket Denim Pria', 'Jaket denim pria yang stylish dan tahan lama', 320000, 38,
2),
('Air Fryer', 'Penggoreng tanpa minyak untuk memasak sehat', 700000, 66, 3),
('The Great Gatsby', 'Novel klasik karya F. Scott Fitzgerald', 90000, 57, 4),
('Krim Pelembab', 'Krim hidrasi untuk kulit kering dan sensitif', 120000, 18, 5),
('Matras Yoga', 'Matras anti slip untuk yoga dan olahraga', 175000, 53, 6),
('Set LEGO City', 'Mainan balok bangunan kreatif untuk anak usia 6+', 350000,
37, 7),
('Vacuum Cleaner Mobil', 'Penyedot debu portabel untuk interior mobil',
450000, 64, 8),
('Tablet Vitamin C', 'Tingkatkan daya tahan tubuh dengan vitamin C harian',
75000, 45, 9),
('Kursi Kantor Ergonomis', 'Kursi untuk kenyamanan kerja', 900000, 47, 10);
```

```

INSERT INTO Cart (user_id, product_id, quantity, varian, status) VALUES
(2, 7, 1, 'Standard', 'aktif'),
(4, 8, 2, 'Hitam', 'aktif'),
(6, 9, 1, '1000mg', 'aktif'),
(8, 10, 4, 'Hitam', 'aktif'),
(10, 4, 1, 'Softcover', 'aktif'),
(1, 5, 1, '100ml', 'aktif'),
(3, 2, 1, 'XL', 'aktif'),
(5, 3, 2, '3.5L', 'aktif'),
(7, 6, 1, 'Ungu', 'aktif'),
(9, 1, 1, '128GB', 'aktif');

```

```

INSERT INTO Orders (user_id, total_price, status, order_date,
shipping_address) VALUES
(2, 350000, 'diproses', '2025-05-02', 'Jl. Cendana No. 3'),
(4, 450000, 'dikirim', '2025-05-02', 'Jl. Sawo No. 7'),
(6, 75000, 'selesai', '2025-04-30', 'Jl. Nangka No. 12'),
(8, 900000, 'selesai', '2025-04-29', 'Jl. Rambutan No. 20'),
(10, 90000, 'diproses', '2025-05-01', 'Jl. Pepaya No. 5'),
(1, 120000, 'dibatalkan', '2025-05-02', 'Jl. Jeruk No. 15'),
(3, 320000, 'selesai', '2025-05-01', 'Jl. Kedondong No. 8'),
(5, 700000, 'dikirim', '2025-05-02', 'Jl. Semangka No. 4'),
(7, 175000, 'selesai', '2025-04-30', 'Jl. Belimbing No. 9'),
(9, 23000000, 'diproses', '2025-05-02', 'Jl. Durian No. 18');

```

```

INSERT INTO Order_Item (order_id, product_id, quantity, harga, varian)
VALUES
(2, 7, 1, 350000, 'Standard'),
(10, 8, 2, 450000, 'Hitam'),
(8, 9, 1, 75000, '1000mg'),
(3, 10, 4, 900000, 'Hitam'),
(9, 4, 1, 90000, 'Softcover'),

```

```
(5, 5, 1, 120000, '100ml'),  
(7, 2, 1, 320000, 'XL'),  
(4, 3, 2, 700000, '3.5L'),  
(6, 6, 1, 175000, 'Ungu'),  
(1, 1, 1, 23000000, '128GB');
```

```
INSERT INTO Payment (order_id, user_id, payment_method, payment_date,  
payment_status) VALUES
```

```
(4, 2, 'E-Wallet', '2025-05-02', 'berhasil'),  
(1, 4, 'Kartu Kredit', '2025-05-02', 'berhasil'),  
(10, 6, 'Transfer Bank', '2025-04-30', 'berhasil'),  
(9, 8, 'E-Wallet', '2025-04-29', 'berhasil'),  
(2, 10, 'Kartu Kredit', '2025-05-01', 'berhasil'),  
(8, 1, 'Transfer Bank', '2025-05-02', 'gagal'),  
(7, 3, 'E-Wallet', '2025-05-01', 'berhasil'),  
(6, 5, 'COD', '2025-05-02', 'berhasil'),  
(5, 7, 'Kartu Kredit', '2025-04-30', 'berhasil'),  
(3, 9, 'E-Wallet', '2025-05-02', 'berhasil');
```

```
INSERT INTO Shipping (order_id, user_id, courier, tracking_number,  
shipping_status, shipping_date, delivery_date) VALUES
```

```
(6, 2, 'JNE', 'JNE234567', 'diproses', '2025-05-02', NULL),  
(7, 4, 'SiCepat', 'SC890123', 'dikirim', '2025-05-02', NULL),  
(8, 6, 'J&T', 'JT456789', 'sampai', '2025-04-30', '2025-05-04'),  
(9, 8, 'Tiki', 'TK678901', 'sampai', '2025-04-29', '2025-05-04'),  
(10, 10, 'POS Indonesia', 'POS012345', 'diproses', NULL, NULL),  
(1, 1, 'JNE', 'JNE987654', 'dibatalkan', NULL, NULL),  
(2, 3, 'SiCepat', 'SC345678', 'sampai', '2025-05-01', '2025-05-07'),  
(3, 5, 'J&T', 'JT567890', 'dikirim', '2025-05-02', NULL),  
(4, 7, 'Tiki', 'TK789012', 'sampai', '2025-04-30', '2025-05-05'),  
(5, 9, 'POS Indonesia', 'POS234567', 'diproses', NULL, NULL);
```

```
INSERT INTO Review (user_id, category_id, rating, comment, review_date)
VALUES
```

```
(7, 2, 5, 'Produk sangat bagus dan berkualitas!', '2025-05-02'),
(8, 4, 4, 'Sangat nyaman dipakai, namun agak ketat.', '2025-05-02'),
(9, 6, 3, 'Cukup baik, namun saya mengharapkan lebih.', '2025-04-30'),
(10, 8, 5, 'Sangat puas dengan pengirimannya, produk sesuai deskripsi!', '2025-04-29'),
(4, 10, 4, 'Buku ini sangat menarik, tapi agak lambat pengirimannya.', '2025-05-01'),
(5, 1, 3, 'Krim pelembab ini oke, tapi kurang cocok untuk kulit saya.', '2025-05-02'),
(2, 3, 5, 'Jaket ini sangat stylish dan nyaman.', '2025-05-01'),
(3, 5, 4, 'Penggorengan tanpa minyak ini sangat memudahkan.', '2025-05-02'),
(6, 7, 4, 'Matras ini cukup tebal dan nyaman untuk yoga.', '2025-04-30'),
(1, 9, 5, 'iPhone 14 sangat cepat dan layarnya tajam!', '2025-05-02');
```

Pembahasan:

Perintah SQL yang diatas digunakan untuk mengisi data awal ke dalam sistem *e-commerce*. Bagian pertama menambahkan sepuluh pengguna ke tabel User dengan kode `INSERT INTO User (nama, email, password, role)`. Selanjutnya, sepuluh kategori produk dimasukkan ke tabel Category, dengan kode `INSERT INTO Category (nama, deskripsi, status)`. Tabel Product juga diisi dengan sepuluh produk dengan menggunakan kode `INSERT INTO Product (nama, deskripsi, harga, stock, category_id)`. Lalu terdapat kode `INSERT INTO Cart (user_id, product_id, quantity, varian, status)` untuk memasukkan data keranjang belanja ke tabel Cart. Lalu untuk memasukkan data ke dalam Tabel Orders digunakan kode `INSERT INTO Orders (user_id, total_price, status, order_date, shipping_address)` sedangkan Order_Item `INSERT INTO Order_Item (order_id, product_id, quantity, harga, varian)`. Untuk menambahkan data pada tabel Payment menggunakan kode `INSERT INTO Payment (order_id, user_id, payment_method, payment_date, payment_status)`. Data pada tabel Shipping dimasukkan dengan kode `INSERT INTO Shipping (order_id, user_id, courier, tracking_number, shipping_status,`

shipping_date, delivery_date). Terakhir untuk memasukkan data pada tabel Review digunakan kode INSERT INTO Review (user_id, category_id, rating, comment, review_date).

Printscreen Output :

	user_id	nama	email	password	role
▶	1	Andi Saputra	andisaputra@gmail.com	password1	admin
	2	Budi Santoso	budisantoso@gmail.com	password2	customer
	3	Citra Lestari	citralestari@gmail.com	password3	customer
	4	Dewi Maharani	dewimaharani@gmail.com	password4	customer
	5	Eko Prasetyo	ekoprasetyo@gmail.com	password5	customer
	6	Fitriani Ayu	fitrianiayu@gmail.com	password6	customer
	7	Gilang Ramadhan	gilangramadhan@gmail.com	password7	customer
	8	Hesti Wulandari	hestiwulandari@gmail.com	password8	customer
	9	Imam Hidayat	imamhidayat@gmail.com	password9	customer
	10	Joko Suharto	jokosuharto@gmail.com	password10	customer
*	NULL	NULL	NULL	NULL	NULL

Gambar 4.13 Output Tabel User

	category_id	nama	deskripsi	status
▶	1	Elektronik	Perangkat dan gadget seperti ponsel, laptop, d...	tersedia
	2	Pakaian	Pakaian untuk pria, wanita, dan anak-anak	tersedia
	3	Rumah & Dapur	Peralatan rumah tangga dan perlengkapan dapur	tersedia
	4	Buku	Fiksi, non-fiksi, edukasi, dan lainnya	tersedia
	5	Kecantikan	Produk perawatan kulit, makeup, dan perawata...	tersedia
	6	Olahraga	Peralatan olahraga, kebugaran, dan kegiatan lu...	tersedia
	7	Mainan	Mainan dan permainan untuk segala usia	tersedia
	8	Otomotif	Aksesori mobil dan peralatan otomotif	tersedia
	9	Kesehatan	Produk perawatan kesehatan dan suplemen	tersedia
	10	Perlengkapan Kantor	Alat tulis, elektronik kantor, dan furnitur kantor	tersedia
*	NULL	NULL	NULL	NULL

Gambar 4.14 Output Tabel Category

	order_id	user_id	total_price	status	order_date	shipping_address
▶	1	2	350000	diproses	2025-05-02	Jl. Cendana No. 3
	2	4	450000	dikirim	2025-05-02	Jl. Sawo No. 7
	3	6	75000	selesai	2025-04-30	Jl. Nangka No. 12
	4	8	900000	selesai	2025-04-29	Jl. Rambutan No. 20
	5	10	90000	diproses	2025-05-01	Jl. Pepaya No. 5
	6	1	120000	dibatalkan	2025-05-02	Jl. Jeruk No. 15
	7	3	320000	selesai	2025-05-01	Jl. Kedondong No. 8
	8	5	700000	dikirim	2025-05-02	Jl. Semangka No. 4
	9	7	175000	selesai	2025-04-30	Jl. Belimbing No. 9
	10	9	23000000	diproses	2025-05-02	Jl. Durian No. 18
*	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 4.15 Output Tabel Order

	order_item_id	order_id	product_id	quantity	harga	varian
▶	1	2	7	1	350000	Standard
	2	10	8	2	450000	Hitam
	3	8	9	1	75000	1000mg
	4	3	10	4	900000	Hitam
	5	9	4	1	90000	Softcover
	6	5	5	1	120000	100ml
	7	7	2	1	320000	XL
	8	4	3	2	700000	3.5L
	9	6	6	1	175000	Ungu
	10	1	1	1	23000000	128GB
*	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 4.16 Output Tabel Order_Item

	payment_id	order_id	user_id	payment_method	payment_date	payment_status
▶	1	4	2	E-Wallet	2025-05-02	berhasil
	2	1	4	Kartu Kredit	2025-05-02	berhasil
	3	10	6	Transfer Bank	2025-04-30	berhasil
	4	9	8	E-Wallet	2025-04-29	berhasil
	5	2	10	Kartu Kredit	2025-05-01	berhasil
	6	8	1	Transfer Bank	2025-05-02	gagal
	7	7	3	E-Wallet	2025-05-01	berhasil
	8	6	5	COD	2025-05-02	berhasil
	9	5	7	Kartu Kredit	2025-04-30	berhasil
	10	3	9	E-Wallet	2025-05-02	berhasil
*	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 4.17 Output Tabel Payment

	shipping_id	order_id	user_id	courier	tracking_number	shipping_status	shipping_date	delivery_date
▶	1	6	2	JNE	JNE234567	diproses	2025-05-02	NULL
	2	7	4	SiCepat	SC890123	dikirim	2025-05-02	NULL
	3	8	6	J&T	JT456789	sampai	2025-04-30	2025-05-04
	4	9	8	Tiki	TK678901	sampai	2025-04-29	2025-05-04
	5	10	10	POS Indonesia	POS012345	diproses	NULL	NULL
	6	1	1	JNE	JNE987654	dibatalkan	NULL	NULL
	7	2	3	SiCepat	SC345678	sampai	2025-05-01	2025-05-07
	8	3	5	J&T	JT567890	dikirim	2025-05-02	NULL
	9	4	7	Tiki	TK789012	sampai	2025-04-30	2025-05-05
	10	5	9	POS Indonesia	POS234567	diproses	NULL	NULL

Gambar 4.18 Output Tabel Shipping

	review_id	user_id	category_id	rating	comment	review_date
▶	1	7	2	5	Produk sangat bagus dan berkualitas!	2025-05-02
	2	8	4	4	Sangat nyaman dipakai, namun agak ketat.	2025-05-02
	3	9	6	3	Cukup baik, namun saya mengharapkan lebih.	2025-04-30
	4	10	8	5	Sangat puas dengan pengirimannya, produk se...	2025-04-29
	5	4	10	4	Buku ini sangat menarik, tapi agak lambat pengi...	2025-05-01
	6	5	1	3	Krim pelembab ini oke, tapi kurang cocok untuk ...	2025-05-02
	7	2	3	5	Jaket ini sangat stylish dan nyaman.	2025-05-01
	8	3	5	4	Pengorengan tanpa minyak ini sangat muda...	2025-05-02
	9	6	7	4	Matras ini cukup tebal dan nyaman untuk yoga.	2025-04-30
	10	1	9	5	iPhone 14 sangat cepat dan layarnya tajam!	2025-05-02
*	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 4.19 Output Tabel Review

Pembahasan :

Pada gambar Output diatas merupakan data awal yang berhasil dimasukkan ke dalam masing-masing tabel di database *e-commerce*. Setiap tabel seperti User, Category, Product, Cart, Orders, Order_Item, Payment, Shipping, dan Review akan berisi sepuluh baris data sesuai dengan perintah INSERT, yang mencerminkan struktur dan relasi antar data dalam sistem sehingga siap digunakan untuk proses transaksi, manajemen produk, ulasan pelanggan, dan pelacakan pengiriman.

4. Menampilkan data dengan join**Printscreen Source Code :**

```
SELECT o.order_id, u.nama AS nama_user, p.nama AS nama_produk, oi.quantity, oi.harga
FROM Orders o
INNER JOIN User u ON o.user_id = u.user_id |
INNER JOIN Order_Item oi ON o.order_id = oi.order_id
INNER JOIN Product p ON oi.product_id = p.product_id;
```

Gambar 4. 20 *Source Code* Inner Join

Source Code:

```
SELECT o.order_id, u.nama AS nama_user, p.nama AS nama_produk,
oi.quantity, oi.harga
FROM Orders o
INNER JOIN User u ON o.user_id = u.user_id
INNER JOIN Order_Item oi ON o.order_id = oi.order_id
INNER JOIN Product p ON oi.product_id = p.product_id;
```

Pembahasan :

Kode ini `SELECT o.order_id, u.nama AS nama_user, p.nama AS nama_produk, oi.quantity, oi.harga` digunakan untuk menampilkan data pesanan dari pengguna termasuk nama pengguna, nama produk, jumlah yang dipesan, dan harga. Data tersebut diambil dari empat tabel yang dihubungkan dengan `INNER JOIN` berdasarkan relasi antar-ID, yakni tabel Orders `FROM Orders o`, User `INNER JOIN User u ON o.user_id = u.user_id`, Order_Item `INNER JOIN Order_Item oi ON o.order_id = oi.order_id`, dan Product `INNER JOIN Product p ON oi.product_id = p.product_id`.

Printscreen Output:

	order_id	nama_user	nama_produk	quantity	harga
▶	6	Andi Saputra	Matras Yoga	1	175000
	1	Budi Santoso	iPhone 14	1	23000000
	7	Citra Lestari	Jaket Denim Pria	1	320000
	2	Dewi Maharani	Set LEGO City	1	350000
	8	Eko Prasetyo	Tablet Vitamin C	1	75000
	3	Fitriani Ayu	Kursi Kantor Ergonomis	4	900000
	9	Gilang Ramadhan	The Great Gatsby	1	90000
	4	Hesti Wulandari	Air Fryer	2	700000
	10	Imam Hidayat	Vacuum Cleaner Mobil	2	450000
	5	Joko Suharto	Krim Pelembab	1	120000

Gambar 4.21 Output Inner Join

Pembahasan :

Output dari kode SQL ini adalah daftar pesanan lengkap yang menampilkan nama pengguna, nama produk yang dipesan, jumlah barang, dan harga. Informasi ini disatukan dari beberapa tabel melalui relasi ID, sehingga memudahkan dalam melihat detail setiap pesanan secara menyeluruh dalam satu tampilan.

Printscreen Source Code :

```
SELECT u.user_id, u.nama, o.order_id, o.total_price
FROM User u LEFT JOIN Orders o ON u.user_id = o.user_id;
```

Gambar 4.22 Source Code Left Join

Source Code :

```
SELECT u.user_id, u.nama, o.order_id, o.total_price
FROM User u LEFT JOIN Orders o ON u.user_id = o.user_id;
```

Pembahasan :

Kode SQL ini digunakan untuk menampilkan daftar semua pengguna beserta nama dari pengguna, serta order_id dan total_price dengan menggunakan kode berupa `SELECT u.user_id, u.nama, o.order_id, o.total_price`. Dengan menggunakan kueri left join untuk menghubungkan antara tabel user dan orders `FROM User u LEFT JOIN Orders o ON u.user_id = o.user_id`, kueri ini memastikan bahwa semua data dari tabel user tetap ditampilkan.

Printscreen Output:

	user_id	nama	order_id	total_price
▶	1	Andi Saputra	6	120000
	2	Budi Santoso	1	350000
	3	Citra Lestari	7	320000
	4	Dewi Maharani	2	450000
	5	Eko Prasetyo	8	700000
	6	Fitriani Ayu	3	75000
	7	Gilang Ramadhan	9	175000
	8	Hesti Wulandari	4	900000
	9	Imam Hidayat	10	23000000
	10	Joko Suharto	5	90000

Gambar 4.23 Output Left Join

Pembahasan :

Output dari kode SQL ini adalah daftar semua pengguna beserta informasi pesanan mereka. Jika seorang pengguna belum pernah melakukan pemesanan, kolom `order_id` dan `total_price` akan bernilai NULL, namun data pengguna tetap ditampilkan.

Printscreen Source Code :

```
SELECT p.product_id, p.nama AS nama_produk, oi.order_id, oi.quantity  
FROM Product p RIGHT JOIN Order_Item oi ON p.product_id = oi.product_id;
```

Gambar 4.24 Source Code Right Join

Source Code :

```
SELECT p.product_id, p.nama AS nama_produk, oi.order_id, oi.quantity  
FROM Product p RIGHT JOIN Order_Item oi ON p.product_id = oi.product_id;
```

Pembahasan :

Kode SQL tersebut digunakan untuk menampilkan data produk yang dipesan oleh pelanggan. Di mana kode `SELECT p.product_id, p.nama AS nama_produk, oi.order_id, oi.quantity` akan menampilkan data `product_id`, nama produk, `order_id`, dan `quantity` dari item pesanan. Lalu tabel `product` dan `order item` di hubungkan dengan `right join` menggunakan kode `FROM Product p RIGHT JOIN Order_Item oi ON p.product_id = oi.product_id`. Kode ini mencocokkan `id product` pada tabel `order item` dan `product`.

Printscreen Output:

	product_id	nama_produk	order_id	quantity
▶	7	Set LEGO City	2	1
	8	Vacuum Cleaner Mobil	10	2
	9	Tablet Vitamin C	8	1
	10	Kursi Kantor Ergonomis	3	4
	4	The Great Gatsby	9	1
	5	Krim Pelembab	5	1
	2	Jaket Denim Pria	7	1
	3	Air Fryer	4	2
	6	Matras Yoga	6	1
	1	iPhone 14	1	1

Gambar 4.25 Output Right Join

Pembahasan :

Pada gambar ini menggunakan *right join*, semua baris dari Order_Item akan ditampilkan, dan data produk akan diisi jika cocok; jika tidak, kolom terkait produk akan bernilai NULL. Ini berguna untuk melacak pesanan meskipun produk terkait mungkin telah dihapus atau tidak tercatat lengkap.

Printscreen Source Code :

```
SELECT u.user_id, u.nama, o.order_id, o.total_price
FROM User u LEFT JOIN Orders o ON u.user_id = o.user_id
UNION
SELECT u.user_id, u.nama, o.order_id, o.total_price
FROM User u RIGHT JOIN Orders o ON u.user_id = o.user_id;
```

Gambar 4.26 Source Code Full Join

Source Code :

```
SELECT u.user_id, u.nama, o.order_id, o.total_price
FROM User u LEFT JOIN Orders o ON u.user_id = o.user_id
UNION
SELECT u.user_id, u.nama, o.order_id, o.total_price
FROM User u RIGHT JOIN Orders o ON u.user_id = o.user_id;
```

Pembahasan :

Kode SQL `SELECT u.user_id, u.nama, o.order_id, o.total_price` tersebut digunakan untuk menampilkan data dari tabel User dan Orders berupa id user, nama, order id dan, total price. *Query* ini menggunakan dua bagian yang

digabung dengan *union*, pertama *left join* antara User dan Orders **FROM User u LEFT JOIN Orders o ON u.user_id = o.user_id** untuk menampilkan semua pengguna beserta pesanan mereka, dan kedua *right join* **FROM User u RIGHT JOIN Orders o ON u.user_id = o.user_id** untuk menampilkan semua pesanan meskipun tidak ada pengguna yang terkait.

Printscreen Output:

	user_id	nama	order_id	total_price
▶	1	Andi Saputra	6	120000
	2	Budi Santoso	1	350000
	3	Citra Lestari	7	320000
	4	Dewi Maharani	2	450000
	5	Eko Prasetyo	8	700000
	6	Fitriani Ayu	3	75000
	7	Gilang Ramadhan	9	175000
	8	Hesti Wulandari	4	900000
	9	Imam Hidayat	10	23000000
	10	Joko Suharto	5	90000

Gambar 4.27 Output Full Join

Pembahasan :

Output dari kode SQL ini berupa daftar gabungan yang menampilkan seluruh data pengguna dan pesanan, baik yang saling berhubungan maupun tidak. Artinya, output akan mencakup pengguna yang memiliki pesanan, pengguna yang belum pernah melakukan pesanan (dengan kolom pesanan kosong), serta pesanan yang tidak terkait dengan data pengguna mana pun (dengan kolom pengguna kosong). Dengan menggunakan LEFT JOIN dan RIGHT JOIN yang digabungkan melalui UNION, hasil akhirnya menampilkan informasi user_id, nama pengguna, order_id, dan total_price secara lengkap tanpa ada data yang diabaikan.

5. Alias dan Operator

Printscreen Source Code :

```
SELECT p.nama AS product_name, SUM(oi.quantity) AS total_sold
FROM Product p JOIN Order_Item oi ON p.product_id = oi.product_id
GROUP BY p.nama ORDER BY total_sold DESC;
```

Gambar 4.28 Sorce Code Alias dan Operator

Source Code :

```
SELECT p.nama AS product_name, SUM(oi.quantity) AS total_sold  
FROM Product p JOIN Order_Item oi ON p.product_id = oi.product_id  
GROUP BY p.nama ORDER BY total_sold DESC;
```

Pembahasan :

Kode SQL ini mengambil data dari tabel Product dan Order_Item, lalu menghubungkannya menggunakan **JOIN** berdasarkan product_id. Selanjutnya, data dikelompokkan **GROUP BY** berdasarkan nama produk p.nama dan dijumlahkan **SUM** berdasarkan jumlah pembelian oi.quantity. Hasil akhir diurutkan **ORDER BY** dari produk yang paling banyak terjual ke yang paling sedikit **DESC**.

Printscreen Output:

	product_name	total_sold
►	Kursi Kantor Ergonomis	4
	Air Fryer	2
	Vacuum Cleaner Mobil	2
	iPhone 14	1
	Jaket Denim Pria	1
	The Great Gatsby	1
	Krim Pelembab	1
	Matras Yoga	1
	Set LEGO City	1
	Tablet Vitamin C	1

Gambar 4.29 Output AS dan Operator

Pembahasan :

Output dari *query* ini adalah daftar produk beserta total jumlah yang terjual untuk masing-masing produk. Hasilnya menampilkan nama produk (product_name) dan total jumlah unit yang terjual (total_sold), yang diurutkan dari produk dengan penjualan terbanyak hingga yang terendah. Data ini memberikan gambaran tentang produk mana yang paling populer atau banyak dibeli di sistem *e-commerce*.

Printscreen Source Code :

```
SELECT u.nama AS customer_name, COUNT(o.order_id) AS number_of_orders, SUM(o.total_price) AS total_spent  
FROM User u JOIN Orders o ON u.user_id = o.user_id  
GROUP BY u.nama HAVING total_spent > 1000000;
```

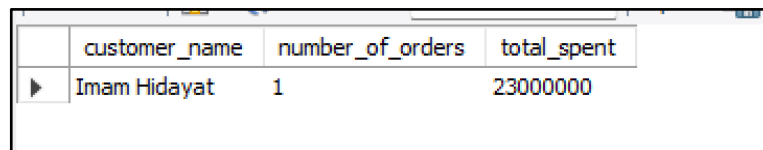
Gambar 4. 30 Source Code AS dan Operator

Source Code :

```
SELECT u.nama AS customer_name, COUNT(o.order_id) AS  
number_of_orders, SUM(o.total_price) AS total_spent  
FROM User u JOIN Orders o ON u.user_id = o.user_id  
GROUP BY u.nama HAVING total_spent > 1000000;
```

Pembahasan :

Kode SQL ini menampilkan daftar nama pelanggan, jumlah pesanan yang mereka buat, dan total uang. Data diambil dari tabel User dan Orders dengan kode `FROM User u JOIN Orders o ON u.user_id = o.user_id` yang dihubungkan melalui `user_id`, lalu dikelompokkan berdasarkan nama pelanggan. Hanya pelanggan dengan total belanja lebih dari 1.000.000 yang ditampilkan, menggunakan klausa `HAVING`.

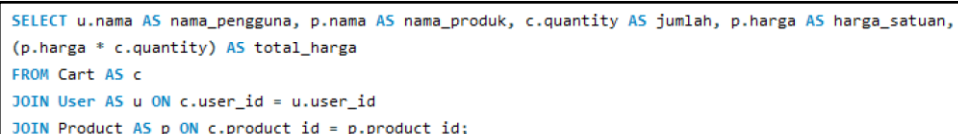
Printscreen Output:

	customer_name	number_of_orders	total_spent
▶	Imam Hidayat	1	23000000

Gambar 4.31 Output AS dan Operator

Pembahasan :

Output dari *query* ini adalah daftar nama pelanggan, jumlah pesanan yang telah mereka buat, dan total uang yang telah mereka belanjakan. Hanya pelanggan yang total belanjanya melebihi 1.000.000 yang akan ditampilkan. Data ini menunjukkan siapa saja pelanggan yang melakukan pembelian dalam jumlah besar, serta memberikan informasi tentang frekuensi dan besaran pengeluaran mereka di sistem *e-commerce*.

Printscreen Source Code :

```
SELECT u.nama AS nama_pengguna, p.nama AS nama_produk, c.quantity AS jumlah, p.harga AS harga_satuan,  
(p.harga * c.quantity) AS total_harga  
FROM Cart AS c  
JOIN User AS u ON c.user_id = u.user_id  
JOIN Product AS p ON c.product_id = p.product_id;
```

Gambar 4.32 Source Code AS dan Operator

Source Code :

```
SELECT u.nama AS nama_pengguna, p.nama AS nama_produk, c.quantity AS  
jumlah, p.harga AS harga_satuan,
```

```
(p.harga c.quantity) AS total_harga
FROM Cart AS c
JOIN User AS u ON c.user_id = u.user_id
JOIN Product AS p ON c.product_id = p.product_id;
```

Pembahasan :

Kode SQL tersebut `SELECT u.nama AS nama_pengguna, p.nama AS nama_produk, c.quantity AS jumlah, p.harga AS harga_satuan, (p.harga c.quantity) AS total_harga` digunakan untuk menampilkan isi keranjang belanja pengguna. *Query* ini mengambil nama pengguna, nama produk, jumlah produk yang dibeli, harga satuan, dan total harga (hasil perkalian jumlah dengan harga). Data diambil dari tabel Cart lalu digabung dengan tabel User `FROM Cart AS c JOIN User AS u ON c.user_id = u.user_id` dan `JOIN Product AS p ON c.product_id = p.product_id` Product berdasarkan `user_id` dan `product_id`. Hasilnya menunjukkan rincian belanja masing-masing pengguna dalam keranjang.

Printscreen Output:

	nama_pengguna	nama_produk	jumlah	harga_satuan	total_harga
►	Budi Santoso	Set LEGO City	1	350000	350000
	Dewi Maharani	Vacuum Cleaner Mobil	2	450000	900000
	Fitriani Ayu	Tablet Vitamin C	1	75000	75000
	Hesti Wulandari	Kursi Kantor Ergonomis	4	900000	3600000
	Joko Suharto	The Great Gatsby	1	90000	90000
	Andi Saputra	Krim Pelembab	1	120000	120000
	Citra Lestari	Jaket Denim Pria	1	320000	320000
	Eko Prasetyo	Air Fryer	2	700000	1400000
	Gilang Ramadhan	Matras Yoga	1	175000	175000
	Imam Hidayat	iPhone 14	1	23000000	23000000

Gambar 4.33 Output AS dan Operator

Pembahasan :

Output dari *query* ini menampilkan rincian keranjang belanja masing-masing pengguna, termasuk nama pengguna, nama produk yang ada dalam keranjang, jumlah produk yang dibeli, harga satuan produk, dan total harga (jumlah dikalikan harga satuan). Data ini diambil dari tabel Cart dan digabungkan dengan tabel User dan Product, memberikan gambaran lengkap mengenai barang-barang yang dipilih oleh pengguna untuk dibeli, beserta jumlah dan harga totalnya.

Printscreen Source Code :

```
SELECT o.order_id AS id_pesanan, u.nama AS nama_pelanggan, o.status AS status_pesanan  
FROM Orders AS o JOIN User AS u ON o.user_id = u.user_id  
WHERE o.status = 'diproses' OR o.status = 'dikirim';
```

Gambar 4. 34 Source Code AS dan Operator

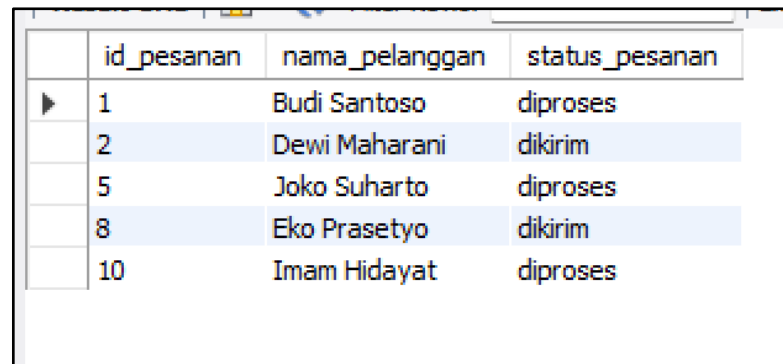
Source Code :

```
SELECT o.order_id AS id_pesanan, u.nama AS nama_pelanggan, o.status AS  
status_pesanan  
FROM Orders AS o JOIN User AS u ON o.user_id = u.user_id  
WHERE o.status = 'diproses' OR o.status = 'dikirim';
```

Pembahasan :

Kode SQL ini menampilkan order_id, nama pelanggan, dan status pesanan SELECT o.order_id AS id_pesanan, u.nama AS nama_pelanggan, o.status AS status_pesanan dari tabel Orders yang digabung dengan tabel User FROM Orders AS o JOIN User AS u ON o.user_id = u.user_id berdasarkan user_id. Filter WHERE o.status = 'diproses' OR o.status = 'dikirim' memastikan hanya pesanan dengan status 'diproses' atau 'dikirim' yang ditampilkan.

Printscreen Output:



	id_pesanan	nama_pelanggan	status_pesanan
▶	1	Budi Santoso	diproses
	2	Dewi Maharani	dikirim
	5	Joko Suharto	diproses
	8	Eko Prasetyo	dikirim
	10	Imam Hidayat	diproses

Gambar 4.35 Output AS dan Operator

Pembahasan :

Output dari query ini menampilkan daftar pesanan yang statusnya sedang diproses atau psudah dikirim. Informasi yang ditampilkan mencakup order_id, nama pelanggan, dan status pesanan. Data ini diperoleh dari penggabungan tabel Orders dengan tabel User berdasarkan user_id, dengan filter yang memastikan hanya pesanan dengan status 'diproses' atau 'dikirim' yang ditampilkan.

Printscreen Source Code :

```
SELECT c.nama AS nama_kategori, SUM(p.harga * oi.quantity) AS total_pendapatan
FROM Order_Item AS oi
JOIN Product AS p ON oi.product_id = p.product_id
JOIN Category AS c ON p.category_id = c.category_id
GROUP BY c.nama;
```

Gambar 4.36 *Source Code* Alias dan Operator

Source Code :

```
SELECT c.nama AS nama_kategori, SUM(p.harga * oi.quantity) AS
total_pendapatan
FROM Order_Item AS oi
JOIN Product AS p ON oi.product_id = p.product_id
JOIN Category AS c ON p.category_id = c.category_id
GROUP BY c.nama;
```

Pembahasan :

Kode SQL ini digunakan untuk menghitung total pendapatan dari setiap kategori produk. Data diambil dari tabel **Order_Item**, **Product**, dan **Category** yang dihubungkan berdasarkan **product_id** dan **category_id**. *Query* menjumlahkan hasil perkalian antara harga produk dan jumlah yang dipesan, lalu mengelompokkannya berdasarkan nama kategori. Hasilnya menunjukkan pendapatan total untuk tiap kategori produk.

Printscreen Output:

	nama_kategori	total_pendapatan
▶	Mainan	350000
	Otomotif	900000
	Kesehatan	75000
	Perlengkapan Kantor	3600000
	Buku	90000
	Kecantikan	120000
	Pakaian	320000
	Rumah & Dapur	1400000
	Olahraga	175000
	Elektronik	23000000

Gambar 4. 37 Output AS dan Operator

Pembahasan :

Output dari query ini menampilkan total pendapatan yang diperoleh dari setiap kategori produk. Pendapatan dihitung dengan mengalikan harga produk dengan jumlah yang dipesan, dan kemudian mengelompokkan hasilnya berdasarkan nama kategori produk. Data ini diperoleh dari penggabungan tabel Order_Item, Product, dan Category, yang dihubungkan berdasarkan product_id dan category_id. Hasilnya menunjukkan total pendapatan untuk masing-masing kategori produk.

Printscreen Source Code :

```
SELECT o.order_id AS id_pesanan, u.nama AS nama_pelanggan, o.order_date AS tanggal_pesanan
FROM Orders AS o JOIN User AS u ON o.user_id = u.user_id
WHERE o.order_date BETWEEN '2025-05-01' AND '2025-05-03';
```

Gambar 4.38 Operator AS dan Operator

Source Code :

```
SELECT o.order_id AS id_pesanan, u.nama AS nama_pelanggan, o.order_date
AS tanggal_pesanan
FROM Orders AS o JOIN User AS u ON o.user_id = u.user_id
WHERE o.order_date BETWEEN '2025-05-01' AND '2025-05-03';
```

Pembahasan :

Kode SQL ini menampilkan daftar pesanan yang dilakukan antara tanggal 1 hingga 3 Mei 2025. *Query* mengambil **ID pesanan**, nama pelanggan, dan tanggal pesanan dari tabel Orders yang digabung dengan tabel User berdasarkan **user_id**. Filter **WHERE** dengan **BETWEEN** digunakan untuk membatasi hasil hanya pada pesanan yang dibuat dalam rentang tanggal tersebut.

Printscreen Output:

	id_pesanan	nama_pelanggan	tanggal_pesanan
▶	1	Budi Santoso	2025-05-02
	2	Dewi Maharani	2025-05-02
	5	Joko Suharto	2025-05-01
	6	Andi Saputra	2025-05-02
	7	Citra Lestari	2025-05-01
	8	Eko Prasetyo	2025-05-02
	10	Imam Hidayat	2025-05-02

Gambar 4. 39 Output AS dan Operator

Pembahasan :

Pada output menggabungkan tabel Orders dan User berdasarkan user_id untuk menampilkan informasi pesanan pelanggan dalam rentang tanggal tertentu. Kueri ini menampilkan ID pesanan, nama pelanggan, dan tanggal pesanan untuk pesanan yang dibuat antara tanggal 1 Mei 2025 dan 3 Mei 2025 (inklusif).

Printscreen Source Code :

```
SELECT
  p.payment_id AS id_pembayaran,
  u.nama AS nama_pelanggan,
  CASE
    WHEN p.payment_status = 'berhasil' THEN 'Pembayaran Sukses'
    WHEN p.payment_status = 'gagal' THEN 'Pembayaran Gagal'
    ELSE 'Status Tidak Diketahui'
  END AS deskripsi_status
FROM Payment AS p
JOIN User AS u ON p.user_id = u.user_id;
```

Gambar 4.40 Source Code AS dan Operator

Source Code :

```
SELECT
  p.payment_id AS id_pembayaran,
  u.nama AS nama_pelanggan,
  CASE
    WHEN p.payment_status = 'berhasil' THEN 'Pembayaran Sukses'
    WHEN p.payment_status = 'gagal' THEN 'Pembayaran Gagal'
    ELSE 'Status Tidak Diketahui'
  END AS deskripsi_status
FROM Payment AS p
JOIN User AS u ON p.user_id = u.user_id;
```

Pembahasan :

Kode SQL ini menampilkan daftar pembayaran beserta nama pelanggan dan deskripsi status pembayarannya. Data diambil dari tabel Payment yang digabung dengan tabel User berdasarkan user_id. Kolom **payment_status** diterjemahkan menggunakan **CASE** menjadi keterangan yang lebih jelas:

'berhasil' menjadi *Pembayaran Sukses*, 'gagal' menjadi *Pembayaran Gagal*, dan status lainnya ditampilkan sebagai *Status Tidak Diketahui*.

Printscreen Output:

	id_pembayaran	nama_pelanggan	deskripsi_status
▶	1	Budi Santoso	Pembayaran Sukses
	2	Dewi Maharani	Pembayaran Sukses
	3	Fitriani Ayu	Pembayaran Sukses
	4	Hesti Wulandari	Pembayaran Sukses
	5	Joko Suharto	Pembayaran Sukses
	6	Andi Saputra	Pembayaran Gagal
	7	Citra Lestari	Pembayaran Sukses
	8	Eko Prasetyo	Pembayaran Sukses
	9	Gilang Ramadhan	Pembayaran Sukses
	10	Imam Hidayat	Pembayaran Sukses

Gambar 4.41 Output AS dan Operator

Pembahasan :

Pada output ini menggabungkan tabel Payment dan User berdasarkan `user_id` untuk menampilkan informasi pembayaran pelanggan. Kueri akan menampilkan **ID pembayaran**, nama pelanggan, dan deskripsi status pembayaran yang diubah dari nilai aslinya ('berhasil' atau 'gagal') menjadi format yang lebih deskriptif.

Printscreen Source Code :

```
SELECT p.nama AS nama_produk, p.harga AS harga_produk
FROM Product AS p
WHERE p.nama LIKE '%Kursi%';
```

Gambar 5.15 Filter Mencari Nama Kursi

Source Code :

```
SELECT p.nama AS nama_produk, p.harga AS harga_produk
FROM Product AS p
WHERE p.nama LIKE '%Kursi%';
```

Pembahasan :

Dalam kode perintah sql ini kolom nama dari tabel Product akan ditampilkan sebagai `nama_produk`, dan kolom harga_produk `SELECT p.nama AS nama_produk, p.harga AS harga_produk`. Kemudian, klausa `WHERE` digunakan untuk menyaring data agar hanya baris yang memenuhi kondisi

tertentu yang ditampilkan, yaitu nama produk (**p.nama**) yang mengandung kata **Kursi**. Tanda persenpada pola pencarian **Kursi** menandakan bahwa terdapat sebuah kata kursi yang dapat berada di awal, tengah, atau akhir nama produk, sehingga dalam pencarian bersifat fleksibel dan tidak tergantung pada sebuah posisi kata tersebut dalam teks.

Printscreen Output:

	nama_produk	harga_produk
▶	Kursi Kantor Ergonomis	900000

Gambar 4.42 Output AS dan Operator

Pembahasan :

Output dari query ini menampilkan daftar produk yang nama produknya mengandung kata "Kursi". Kolom nama produk akan ditampilkan dengan alias "nama_produk" dan kolom harga dengan alias "harga_produk". Penggunaan klausa WHERE dengan operator LIKE memungkinkan pencarian produk yang namanya mengandung kata "Kursi" di posisi mana pun dalam nama produk. Dengan kata lain, query ini menyaring dan menampilkan produk yang relevan dengan kriteria pencarian tersebut.

Printscreen Source Code :

```
SELECT o.order_id AS id_pesanan, o.status AS status_pesanan
FROM Orders AS o
WHERE o.status IN ('diproses', 'dikirim', 'selesai');
```

Gambar 4.43 Source Code AS dan Operator

Source Code :

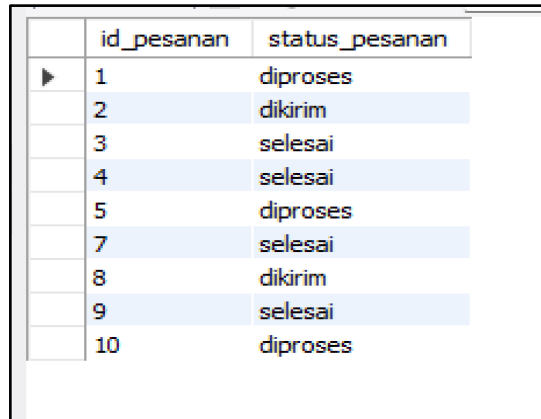
```
SELECT o.order_id AS id_pesanan, o.status AS status_pesanan
FROM Orders AS o
WHERE o.status IN ('diproses', 'dikirim', 'selesai');
```

Pembahasan :

Kode SQL ini digunakan untuk menampilkan daftar pesanan yang memiliki status tertentu, yaitu "diproses", "dikirim", atau "selesai". Dalam query ini, kolom **order_id** dari tabel Orders ditampilkan dengan nama alias **id_pesanan**, dan kolom status ditampilkan sebagai **status_pesanan**. Penggunaan klausa

WHERE berfungsi untuk memfilter data sehingga hanya pesanan dengan salah satu dari tiga status tersebut yang akan ditampilkan dalam hasil output. Tujuannya adalah untuk melihat daftar pesanan yang sedang berjalan atau sudah selesai, dan mengecualikan pesanan yang dibatalkan atau belum diproses.

Printscreen Output:



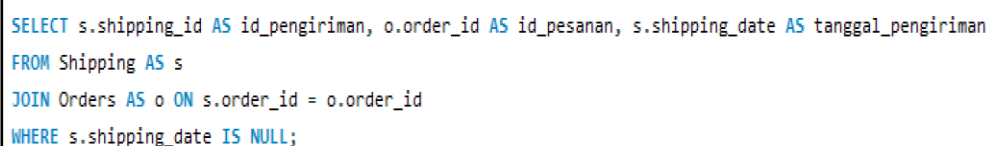
	id_pesanan	status_pesanan
▶	1	diproses
	2	dikirim
	3	selesai
	4	selesai
	5	diproses
	7	selesai
	8	dikirim
	9	selesai
	10	diproses

Gambar 4.44 Output AS dan Operator

Pembahasan :

Pada gambar output ini memberikan gambaran lengkap mengenai ID pesanan dan status masing-masing pesanan yang tercatat dalam tabel Orders. Dengan memilih kolom **order_id** sebagai **id_pesanan** dan kolom status sebagai **status_pesanan** dari tabel Orders. Kueri ini menampilkan semua baris dari tabel Orders karena klausa **WHERE** mencakup semua nilai status yang ada dalam tabel.

Printscreen Source Code :



```
SELECT s.shipping_id AS id_pengiriman, o.order_id AS id_pesanan, s.shipping_date AS tanggal_pengiriman
FROM Shipping AS s
JOIN Orders AS o ON s.order_id = o.order_id
WHERE s.shipping_date IS NULL;
```

Gambar 4.45 Source Code AS dan Operator

Source Code :

```
SELECT s.shipping_id AS id_pengiriman, o.order_id AS id_pesanan,
s.shipping_date AS tanggal_pengiriman
FROM Shipping AS s
JOIN Orders AS o ON s.order_id = o.order_id
WHERE s.shipping_date IS NULL;
```

Pembahasan :

Kode SQL ini digunakan untuk menampilkan daftar pengiriman yang belum memiliki tanggal pengiriman. *Query* mengambil **shipping_id**, **order_id**, dan **shipping_date** dari tabel Shipping yang digabung dengan tabel Orders berdasarkan **order_id**. Filter **WHERE** memastikan hanya pengiriman yang belum memiliki tanggal pengiriman (nilai **shipping_date** adalah NULL) yang ditampilkan.

Printscreen Output:

	id_pengiriman	id_pesanan	tanggal_pengiriman
▶	5	10	NULL
	6	1	NULL
	10	5	NULL

Gambar 4.46 *Source Code* AS dan Operator

Pembahasan :

Pada menampilkan data dari tabel Shipping yang digabungkan dengan tabel Orders berdasarkan **order_id**. Hasil kueri ini difilter untuk hanya menampilkan baris di mana nilai pada kolom tanggal_pengiriman (yang berasal dari kolom **shipping_date** pada tabel Shipping) adalah NULL. gambar ini memperlihatkan tiga catatan pengiriman dengan **ID pengiriman** masing-masing 5, 6, dan 10, yang terkait dengan **ID pesanan** 10, 1, dan 5. Yang penting untuk diperhatikan adalah bahwa semua pengiriman yang ditampilkan ini belum memiliki tanggal pengiriman yang terisi (NULL).

6. Function, Grouping dan Sorting

Printscreen Source Code :

```
SELECT
    DATE_FORMAT(order_date, '%Y-%m') AS month,
    COUNT(*) AS order_count
FROM
    Orders
GROUP BY
    DATE_FORMAT(order_date, '%Y-%m')
ORDER BY
    month;
```

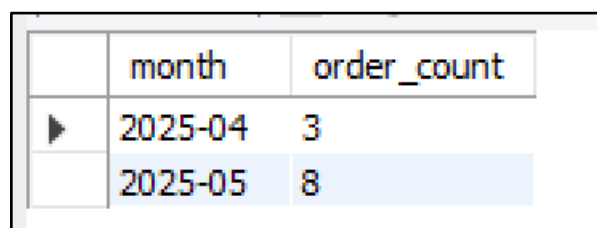
Gambar 4.47 *Source Code* Grouping

Source Code :

```
SELECT
    DATE_FORMAT(order_date, '%Y-%m') AS month,
    COUNT() AS order_count
FROM
    Orders
GROUP BY
    DATE_FORMAT(order_date, '%Y-%m')
ORDER BY
    month;
```

Pembahasan :

Query SQL ini dipakai untuk menampilkan berapa banyak pesanan yang terjadi setiap bulan dari tabel Orders. Pertama, bagian `DATE_FORMAT(order_date, '%Y-%m')` digunakan untuk mengambil data tanggal, lalu mengubahnya jadi format bulan dan tahun (misalnya jadi “2025-01”). Setelah itu, `COUNT()` digunakan untuk menghitung jumlah pesanan di setiap bulan. Data pesanan kemudian dikelompokkan berdasarkan bulan menggunakan `GROUP BY`, jadi semua pesanan yang terjadi di bulan yang sama akan dihitung bersama. Terakhir, hasilnya diurutkan dari bulan paling awal ke bulan paling akhir dengan `ORDER BY month`.

Printscreen Output :

	month	order_count
▶	2025-04	3
	2025-05	8

Gambar 4.48 Output *Grouping*

Pembahasan :

Output dari query ini akan menampilkan dua kolom: satu kolom berisi bulan dan tahun (misalnya "2025-01"), dan kolom lainnya menunjukkan jumlah total pesanan yang dibuat pada bulan tersebut. Setiap baris dalam hasil merepresentasikan jumlah pesanan yang terjadi di bulan tertentu, yang diurutkan secara kronologis dari bulan paling awal ke bulan paling akhir.

Printscreen Source Code :

```
SELECT
  c.nama AS category_name,
  AVG(r.rating) AS average_rating
FROM
  Review r
  JOIN Category c ON r.category_id = c.category_id
GROUP BY
  c.nama
ORDER BY
  average_rating DESC;
```

Gambar 4.49 *Source Code* Grouping

Source Code :

```
SELECT
  c.nama AS category_name,
  AVG(r.rating) AS average_rating
FROM
  Review r
  JOIN Category c ON r.category_id = c.category_id
GROUP BY
  c.nama
ORDER BY
  average_rating DESC;
```

Pembahasan ;

Kode SQL ini digunakan untuk menampilkan rata-rata rating dari setiap kategori berdasarkan data ulasan yang ada. Caranya, *query* ini menggabungkan dua tabel: tabel Review yang berisi data rating dari pengguna, dan tabel Category yang berisi nama-nama kategori produk atau layanan. Penggabungan dilakukan berdasarkan `category_id`, yaitu kolom yang menjadi penghubung di kedua tabel tersebut. Setelah digabungkan, *query* menghitung rata-rata rating (`AVG(r.rating)`) untuk setiap kategori, lalu menampilkan nama kategorinya (`c.nama`) sebagai `category_name`, dan rata-rata rating-nya sebagai `average_rating`. Hasil akhirnya dikelompokkan berdasarkan nama kategori (`GROUP BY c.nama`) dan diurutkan dari rating tertinggi ke terendah (`ORDER BY average_rating DESC`).

Printscreen Output :

	category_name	average_rating
▶	Pakaian	5.0000
	Rumah & Dapur	5.0000
	Otomotif	5.0000
	Kesehatan	5.0000
	Buku	4.0000
	Kecantikan	4.0000
	Mainan	4.0000
	Perlengkapan Kantor	4.0000
	Elektronik	3.0000
	Olahraga	3.0000

Gambar 4. 50 *Source Code Grouping*

Pembahasan :

Output dari *query* ini akan menampilkan daftar kategori produk atau layanan beserta rata-rata rating yang diberikan oleh pengguna untuk setiap kategori. Setiap baris menampilkan nama kategori dan nilai rata-rata ratingnya, yang diurutkan dari kategori dengan rating tertinggi ke yang terendah. Ini membantu mengetahui kategori mana yang paling disukai berdasarkan penilaian pengguna.

Printscreen Source Code :

```
SELECT
  c.nama AS category_name,
  SUM(p.stock) AS total_stock
FROM
  Product p
  JOIN Category c ON p.category_id = c.category_id
GROUP BY
  c.nama
ORDER BY
  total_stock DESC;
```

Gambar 4.51 *Source Code Grouping*

Source Code :

```
SELECT
  c.nama AS category_name,
  AVG(r.rating) AS average_rating
FROM
  Review r
```

JOIN Category c ON r.category_id = c.category_id

GROUP BY

c.nama

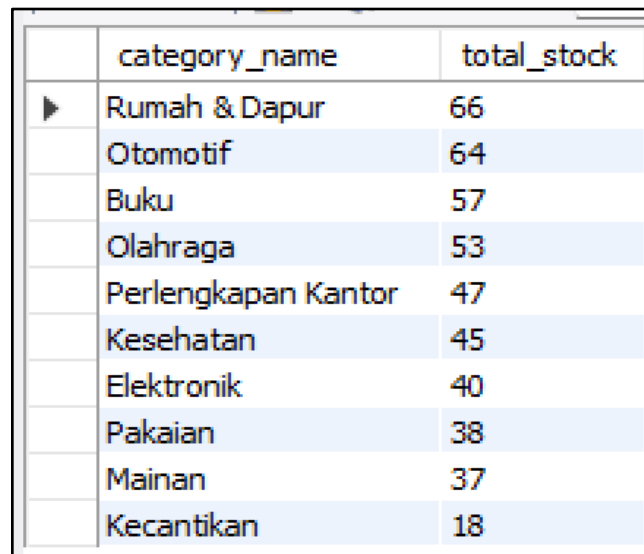
ORDER BY

average_rating DESC;

Pembahasan :

Query SQL ini digunakan untuk melihat rata-rata rating dari setiap kategori produk atau layanan yang ada. Caranya, *query* ini mengambil data dari dua tabel, yaitu tabel Review yang berisi rating dari pengguna dan tabel Category yang berisi daftar nama kategori. Data dari kedua tabel ini digabungkan dengan menggunakan kolom category_id yang ada di masing-masing tabel. Setelah itu, *query* ini menghitung rata-rata rating untuk setiap kategori dengan menggunakan fungsi **AVG()**. Hasilnya, setiap kategori akan ditampilkan bersama dengan rating rata-ratanya, dan data tersebut diurutkan dari kategori dengan rating tertinggi ke terendah.

Printscreen Output :



	category_name	total_stock
▶	Rumah & Dapur	66
	Otomotif	64
	Buku	57
	Olahraga	53
	Perlengkapan Kantor	47
	Kesehatan	45
	Elektronik	40
	Pakaian	38
	Mainan	37
	Kecantikan	18

Gambar 4.52 Output Grouping

Pembahasan :

Output dari *query* ini menunjukkan kategori produk dan rata-rata rating yang diberikan oleh pelanggan untuk masing-masing kategori. Jadi, setiap baris hasil menunjukkan kategori seperti “Elektronik” atau “Buku” dan berapa rata-rata rating yang diberikan oleh pengguna.

Printscreen Source Code :

```
SELECT
    order_date,
    SUM(total_price) AS daily_revenue
FROM
    Orders
GROUP BY
    order_date
ORDER BY
    order_date;
```

Gambar 4.53 *Source Code* Grouping

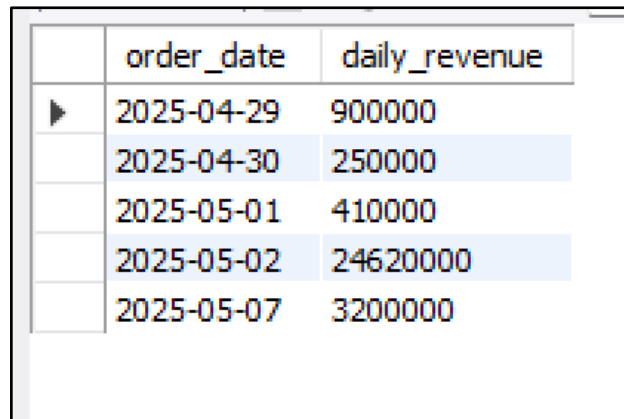
Source Code :

```
SELECT
    order_date,
    SUM(total_price) AS daily_revenue
FROM
    Orders
GROUP BY
    order_date
ORDER BY
    order_date;
```

Pembahasan :

Query SQL ini digunakan untuk menghitung pendapatan harian dari tabel Orders. Setiap baris data diambil berdasarkan tanggal pesanan (*order_date*), dan *query* ini menjumlahkan total harga dari semua pesanan yang dilakukan pada hari tersebut menggunakan fungsi *SUM(total_price)*, yang hasilnya diberi nama alias *daily_revenue*. Data kemudian dikelompokkan berdasarkan tanggal pesanan (*GROUP BY order_date*), sehingga kita mendapatkan total pendapatan untuk setiap hari. Hasilnya kemudian diurutkan berdasarkan tanggal pesanan (*ORDER BY order_date*), dari yang paling awal hingga yang paling terbaru.

Printscreen Output :



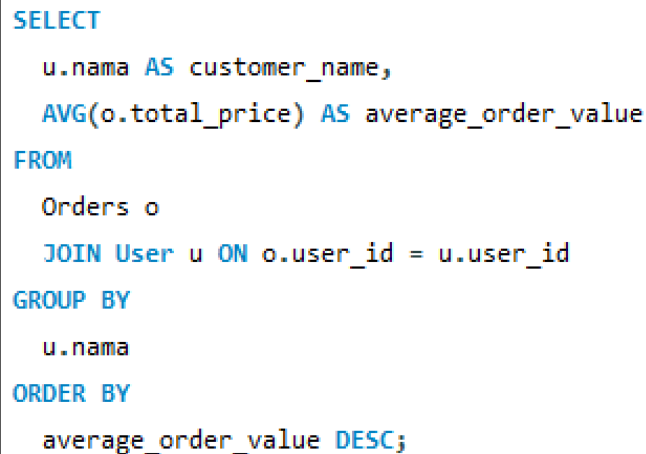
	order_date	daily_revenue
▶	2025-04-29	900000
	2025-04-30	250000
	2025-05-01	410000
	2025-05-02	24620000
	2025-05-07	3200000

Gambar 4.54 Output Grouping

Pembahasan :

Output dari query ini menampilkan daftar tanggal beserta total pendapatan yang diperoleh pada masing-masing tanggal. Setiap baris menunjukkan satu hari dan jumlah total harga pesanan pada hari tersebut, yang membantu dalam menganalisis pendapatan harian secara kronologis.

Printscreen Source Code :



```
SELECT
    u.nama AS customer_name,
    AVG(o.total_price) AS average_order_value
FROM
    Orders o
    JOIN User u ON o.user_id = u.user_id
GROUP BY
    u.nama
ORDER BY
    average_order_value DESC;
```

Gambar 4.55 Source Code Grouping

Source Code :

```
SELECT
    u.nama AS customer_name,
    AVG(o.total_price) AS average_order_value
FROM
    Orders o
```



```
JOIN User u ON o.user_id = u.user_id
```

```
GROUP BY
```

```
u.nama
```

```
ORDER BY
```

```
average_order_value DESC;
```

Pembahasan :

Query ini menggabungkan dua tabel, yaitu tabel Orders yang berisi data pesanan dan tabel User yang berisi data pelanggan. Dengan menggunakan JOIN, data dari kedua tabel ini digabungkan berdasarkan user_id, yang menjadi penghubung antara pesanan dan pelanggan. Setelah digabungkan, *query* ini menghitung rata-rata total harga pesanan untuk setiap pelanggan dengan menggunakan fungsi `AVG(o.total_price)`. Hasilnya adalah nilai rata-rata pesanan yang dilakukan oleh masing-masing pelanggan. Data kemudian dikelompokkan berdasarkan nama pelanggan (`GROUP BY u.nama`), dan hasilnya diurutkan berdasarkan nilai rata-rata pesanan dari yang tertinggi ke yang terendah (`ORDER BY average_order_value DESC`).

Printscreen Output :

	customer_name	average_order_value
►	Imam Hidayat	23000000.0000
	Eko Prasetyo	1950000.0000
	Hesti Wulandari	900000.0000
	Dewi Maharani	450000.0000
	Budi Santoso	350000.0000
	Citra Lestari	320000.0000
	Gilang Ramadhan	175000.0000
	Andi Saputra	120000.0000
	Joko Suharto	90000.0000
	Fitriani Ayu	75000.0000

Gambar 4.56 Ouput Grouping

Pembahasan :

Output dari *query* ini menampilkan daftar nama pelanggan beserta rata-rata nilai pesanan yang mereka buat. Setiap baris menunjukkan satu pelanggan dan rata-rata total harga dari seluruh pesanan yang pernah mereka lakukan, diurutkan dari pelanggan dengan rata-rata belanja tertinggi hingga terendah.

7. Stored Procedure

Prinscreen Source Code ;

```
DELIMITER $$
CREATE PROCEDURE TambahProduk(
    IN p_nama VARCHAR(100),
    IN p_deskripsi TEXT,
    IN p_harga INT,
    IN p_stock INT,
    IN p_category_id INT
)
BEGIN
    INSERT INTO Product (nama, deskripsi, harga, stock, category_id)
    VALUES (p_nama, p_deskripsi, p_harga, p_stock, p_category_id);

    UPDATE Category
    SET status = CASE
        WHEN (SELECT COUNT(*) FROM Product WHERE category_id = p_category_id) > 0 THEN 'tersedia'
        ELSE 'habis'
    END
    WHERE category_id = p_category_id;
END
$$ DELIMITER ;
```

Gambar 4. 57 Source Code Stored Procedure

Source Code :

DELIMITER \$\$

CREATE PROCEDURE TambahProduk(

IN p_nama VARCHAR(100),

IN p_deskripsi TEXT,

IN p_harga INT,

IN p_stock INT,

IN p_category_id INT

)

BEGIN

INSERT INTO Product (nama, deskripsi, harga, stock, category_id)

VALUES (p_nama, p_deskripsi, p_harga, p_stock, p_category_id);

```

UPDATE Category
SET status = CASE
    WHEN (SELECT COUNT() FROM Product WHERE category_id =
p_category_id) > 0 THEN 'tersedia'
    ELSE 'habis'
END
WHERE category_id = p_category_id;
END

```

\$\$ DELIMITER ;

Pembahasan :

Kode SQL ini adalah prosedur yang berfungsi untuk menambahkan produk baru ke dalam tabel Product dan secara otomatis memperbarui status kategori produk di tabel Category. Prosedur yang bernama TambahProduk menerima beberapa input berupa nama produk, deskripsi, harga, stok, dan ID kategori produk. Langkah pertama adalah memasukkan data produk baru ke dalam tabel Product sesuai dengan nilai-nilai yang diterima dari input. Setelah produk berhasil ditambahkan, prosedur kemudian memperbarui status kategori produk tersebut di tabel Category. Status kategori akan diubah menjadi 'tersedia' jika ada produk yang termasuk dalam kategori tersebut, dan 'habis' jika tidak ada produk lagi di kategori tersebut.

Printscreen Output :

	product_id	nama	deskripsi	harga	stock	category_id
▶	11	Smartphone XYZ	Smartphone canggih dengan kamera 48 MP	5000000	20	1
	12	Smartphone XYZ	Smartphone canggih dengan kamera 48 MP	5000000	20	1
✱	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 4.58 Output Stored Procedure

Pembahasan :

Output ini akan menampilkan nama produk, deskripsi, harga, stok, dan kategori produk yang sesuai, misalnya produk dengan harga 5.000.000 dan stok 20 unit. Kedua, setelah produk berhasil ditambahkan, status kategori produk juga akan diperbarui di tabel Category.

Printscreen Source Code :

```
DELIMITER $$
CREATE PROCEDURE BuatPesanan (
    IN p_user_id INT,
    IN p_product_id INT,
    IN p_quantity INT,
    IN p_varian VARCHAR(50),
    IN p_alamat VARCHAR(50)
)
BEGIN
    DECLARE harga_produk INT;
    DECLARE total INT;
    DECLARE last_order_id INT;

    SELECT harga INTO harga_produk FROM Product WHERE product_id = p_product_id;
    SET total = harga_produk * p_quantity;

    INSERT INTO Orders(user_id, total_price, status, order_date, shipping_address)
    VALUES(p_user_id, total, 'diproses', CURDATE(), p_alamat);
    SET last_order_id = LAST_INSERT_ID();

    INSERT INTO Order_Item(order_id, product_id, quantity, harga, varian)
    VALUES(last_order_id, p_product_id, p_quantity, harga_produk, p_varian);
END
$$ DELIMITER ;

CALL BuatPesanan(5, 2, 10, 'M', 'Jl. Melati No. 25, Bengkulu');
```

Gambar 4.59 Source Code Stored Procedure

Source Code :

```
DELIMITER $$
CREATE PROCEDURE BuatPesanan (
    IN p_user_id INT,
    IN p_product_id INT,
    IN p_quantity INT,
    IN p_varian VARCHAR(50),
    IN p_alamat VARCHAR(50)
)
BEGIN
    DECLARE harga_produk INT;
    DECLARE total INT;
    DECLARE last_order_id INT;

    SELECT harga INTO harga_produk FROM Product WHERE product_id =
p_product_id;
```

```
SET total = harga_produk p_quantity;
```

```
INSERT INTO Orders(user_id, total_price, status, order_date,  
shipping_address)
```

```
VALUES(p_user_id, total, 'diproses', CURDATE(), p_alamat);
```

```
SET last_order_id = LAST_INSERT_ID();
```

```
INSERT INTO Order_Item(order_id, product_id, quantity, harga, varian)
```

```
VALUES(last_order_id, p_product_id, p_quantity, harga_produk, p_varian);
```

```
END
```

```
$$ DELIMITER ;
```

```
CALL BuatPesanan(5, 2, 10, 'M', 'Jl. Melati No. 25, Bengkulu');
```

Pembahasan :

Kode SQL ini berfungsi untuk mencatat pesanan baru dari seorang pelanggan secara otomatis ke dalam database. Prosedur bernama BuatPesanan ini menerima lima informasi penting, yaitu ID pengguna (user), ID produk, jumlah produk yang dipesan, varian produk (seperti ukuran atau warna), dan alamat pengiriman. Begitu dijalankan, sistem akan mencari harga produk berdasarkan ID produk yang diberikan, lalu menghitung total harga dengan mengalikan harga satuan dengan jumlah yang dipesan. Setelah total harga dihitung, sistem akan menyimpan data pesanan ke dalam tabel Orders dengan status awal 'diproses', mencatat tanggal hari ini, dan alamat tujuan pengiriman. Setelah itu, sistem mencatat detail isi pesanan (produk apa, jumlahnya berapa, harganya berapa, dan variannya apa) ke dalam tabel Order_Item, menggunakan ID pesanan yang barusan dibuat.

Printscreen Output :

	order_id	user_id	total_price	status	order_date	shipping_address
▶	6	1	120000	dibatalkan	2025-05-02	Jl. Jeruk No. 15
•	NULL	NULL	NULL	NULL	NULL	NULL

Gambar 4.60 Output Stored Procedure

Pembahasan :

data pesanan baru yang berhasil disimpan ke dalam database. Ketika prosedur BuatPesanan dijalankan dengan data seperti `CALL BuatPesanan(5, 2, 10, 'M', 'Jl. Melati No. 25, Bengkulu');`, maka sistem akan otomatis menyimpan informasi bahwa user dengan ID 5 memesan produk dengan ID 2 sebanyak 10 unit, varian ukuran “M”, dan akan dikirim ke alamat “Jl. Melati No. 25, Bengkulu”.

8. Nested

Printscreen Source Code :

```
} SELECT u.user_id, u.nama, (  
    SELECT SUM(o.total_price)  
    FROM Orders o  
    WHERE o.user_id = u.user_id  
- ) AS total_belanja  
FROM User u  
ORDER BY total_belanja DESC;
```

Gambar 4.61 Source Nested

Source Code :

```
SELECT u.user_id, u.nama, (  
    SELECT SUM(o.total_price)  
    FROM Orders o  
    WHERE o.user_id = u.user_id  
) AS total_belanja  
FROM User u  
ORDER BY total_belanja DESC;
```

Pembahasan :

Kode SQL ini digunakan untuk menampilkan daftar semua pengguna (user) di sistem beserta total belanja mereka, yaitu jumlah keseluruhan harga dari semua pesanan yang pernah mereka buat. Di dalamnya, ada *subquery* (*query* di dalam *query*) yang menjumlahkan total belanja (`SUM(total_price)`) dari tabel Orders berdasarkan `user_id` masing-masing. Jadi, setiap baris yang ditampilkan

akan menunjukkan ID pengguna, nama pengguna, dan total belanja mereka. Hasil akhirnya diurutkan dari yang paling besar total belanjanya sampai yang paling kecil.

Printscreen Output :

	user_id	nama	total_belanja
▶	9	Imam Hidayat	23000000
	5	Eko Prasetyo	10300000
	8	Hesti Wulandari	900000
	4	Dewi Maharani	450000
	2	Budi Santoso	350000
	3	Citra Lestari	320000
	7	Gilang Ramadhan	175000
	1	Andi Saputra	120000
	10	Joko Suharto	90000
	6	Fitriani Ayu	75000

Gambar 4.62 Output Nested

Pembahasan :

Output dari query ini menampilkan daftar seluruh pengguna beserta total belanja mereka. Setiap baris mencakup ID dan nama pengguna, serta jumlah keseluruhan uang yang telah mereka keluarkan untuk semua pesanan. Data ditampilkan dari pengguna dengan total belanja tertinggi ke yang terendah.

Printscreen Soure Code :

```
SELECT c.nama, (  
    SELECT AVG(r.rating)  
    FROM Review r  
    WHERE r.category_id = c.category_id  
) AS rata_rating  
FROM Category c  
ORDER BY rata_rating DESC;
```

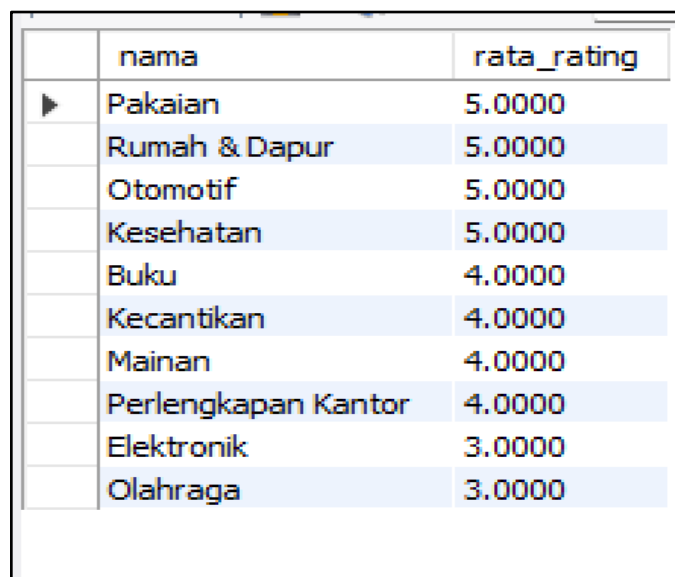
Gambar 4.63 Source Code Nested

Source Code :

```
SELECT c.nama, (  
    SELECT AVG(r.rating)  
    FROM Review r  
    WHERE r.category_id = c.category_id  
)  
AS rata_rating  
FROM Category c  
ORDER BY rata_rating DESC;
```

Pembahasan :

Kode SQL ini digunakan untuk menampilkan daftar kategori produk yang ada di sistem, lalu menunjukkan rata-rata rating (penilaian) dari setiap kategori tersebut. Jadi, sistem akan mencari semua kategori dari tabel Category, lalu untuk setiap kategori itu, sistem akan menghitung berapa rata-rata rating yang diberikan pengguna dari tabel Review, berdasarkan kecocokan category_id.

Printscreen Output :

	nama	rata_rating
▶	Pakaian	5.0000
	Rumah & Dapur	5.0000
	Otomotif	5.0000
	Kesehatan	5.0000
	Buku	4.0000
	Kecantikan	4.0000
	Mainan	4.0000
	Perlengkapan Kantor	4.0000
	Elektronik	3.0000
	Olahraga	3.0000

Gambar 4.64 Output Nested

Pembahasan :

Output dari query ini menampilkan daftar nama kategori produk beserta rata-rata rating yang diberikan oleh pengguna untuk setiap kategori tersebut. Setiap baris menunjukkan satu kategori dan nilai rata-rata rating-nya,

memberikan gambaran kategori mana yang paling disukai berdasarkan penilaian pengguna.

9. Trigger

Printscreen Source Code :

```
-- TRIGGER
DELIMITER $$
CREATE TRIGGER cek_stok
BEFORE INSERT ON Order_Item
FOR EACH ROW
BEGIN
    DECLARE stok_saat_ini INT;

    SELECT stock INTO stok_saat_ini
    FROM Product
    WHERE product_id = NEW.product_id;

    IF stok_saat_ini < NEW.quantity THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Stok tidak mencukupi untuk produk ini';
    END IF;
END
$$ DELIMITER ;
```

Gambar 4.65 Trigger Cek Stok

Source Code:

DELIMITER \$\$

CREATE TRIGGER cek_stok

BEFORE INSERT ON Order_Item

FOR EACH ROW

BEGIN

DECLARE stok_saat_ini INT;

SELECT stock INTO stok_saat_ini

FROM Product

WHERE product_id = NEW.product_id;

IF stok_saat_ini < NEW.quantity THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE_TEXT = 'Stok tidak mencukupi untuk produk ini';

END IF;

END

\$\$

DELIMITER ;

Pembahasan:

Trigger ini menggunakan klausa **BEFORE INSERT**, yang akan dijalankan sebelum data dimasukkan. Di dalam blok **BEGIN** hingga **END**, terdapat deklarasi variabel `stok_saat_ini` untuk menyimpan jumlah stok saat ini dari produk yang ingin dipesan. Dengan perintah **SELECT**, sistem mencari stock dari tabel `Product` berdasarkan `product_id` yang sama dengan data yang baru (`NEW.product_id`). Jika jumlah stok saat ini lebih kecil dari jumlah yang dipesan (`NEW.quantity`), maka trigger akan menghentikan proses **INSERT** dan memunculkan pesan kesalahan menggunakan **SIGNAL SQLSTATE**.

Printscreen Output :



Gambar 9.2 Output Melebihi Stok Produk

Pembahasan:

Output dari trigger ini berupa pesan error yang muncul saat user mencoba memasukkan pesanan melebihi stok produk yang tersedia. Jika stok mencukupi, data akan berhasil disimpan ke tabel ``Order_Item``. Namun, jika jumlah pesanan lebih besar dari stok, maka muncul pesan: "Stok tidak mencukupi untuk produk ini", dan data tidak akan disimpan.

Printscreen Source Code :

```
DELIMITER $$
CREATE TRIGGER kurangi_stok
AFTER INSERT ON Order_Item
FOR EACH ROW
BEGIN
    UPDATE Product
    SET stock = stock - NEW.quantity
    WHERE product_id = NEW.product_id;
END
$$ DELIMITER ;

INSERT INTO Order_Item (order_id, product_id, quantity, harga, varian)
VALUES (1, 2, 2, 100000, 'L');
```

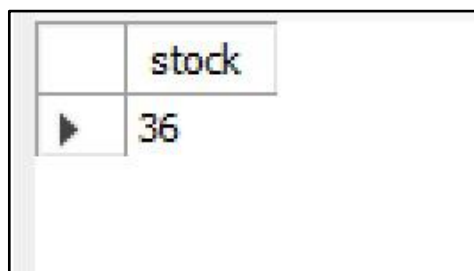
Gambar 4.66 Trigger Pengurangan Stok

Source Code:

```
DELIMITER $$  
CREATE TRIGGER kurangi_stok  
AFTER INSERT ON Order_Item  
FOR EACH ROW  
BEGIN  
    UPDATE Product  
    SET stock = stock - NEW.quantity  
    WHERE product_id = NEW.product_id;  
END  
$$  
DELIMITER ;
```

Pembahasan:

Trigger `kurangi_stok` digunakan untuk secara otomatis mengurangi stok produk setelah pesanan berhasil ditambahkan ke tabel `Order_Item`. Trigger ini menggunakan klausa `AFTER INSERT`, yang akan dijalankan setelah data pesanan berhasil dimasukkan. Di dalam blok `BEGIN` hingga `END`, perintah `UPDATE` akan mengurangi nilai stock di tabel `Product` sebesar jumlah produk (`NEW.quantity`) yang baru saja dipesan. Pemilihan produk ditentukan berdasarkan `product_id` dari baris yang baru dimasukkan.

Printscreen Output :

	stock
▶	36

Gambar 4. 67 Output Pengurangan Stok

Pembahasan:

Output dari trigger ``kurangi_stok`` adalah perubahan pada nilai stok produk di tabel ``Product``. Setelah pesanan baru ditambahkan ke tabel ``Order_Item``, trigger ini akan mengurangi stok produk yang terkait berdasarkan jumlah yang dipesan. Sebagai contoh, jika seorang pelanggan memesan 3 unit

produk dengan `product_id` tertentu, maka stok produk tersebut akan berkurang sebanyak 3 unit di tabel `Product`. Jika stok mencukupi, maka stok akan diperbarui sesuai jumlah pesanan. Jika tidak ada error dalam proses, tidak akan ada pesan yang ditampilkan, tetapi perubahan pada stok akan tercermin di tabel `Product`.

10. DCL

Printscreen Source Code:



```
-- DCL
CREATE USER 'kasir'@'localhost' IDENTIFIED BY 'securepass123';

GRANT SELECT, INSERT ON ecommerce.Orders TO 'kasir'@'localhost';
GRANT SELECT, INSERT ON ecommerce.Order_Item TO 'kasir'@'localhost';

REVOKE UPDATE ON ecommerce.Orders FROM 'kasir'@'localhost';
SHOW GRANTS FOR 'kasir'@'localhost';
```

Gambar 4.68 Source Kode DCL

Source Code:

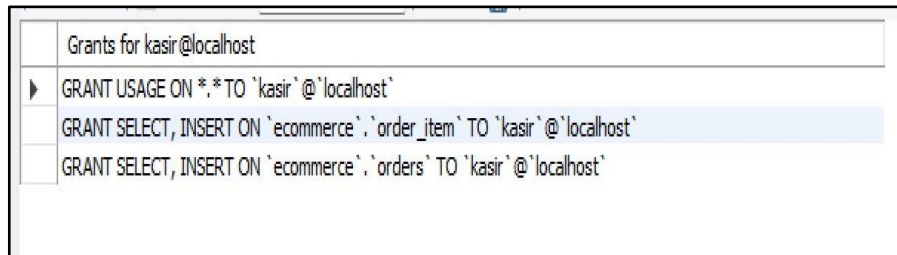
```
CREATE USER 'kasir'@'localhost' IDENTIFIED BY 'securepass123';
GRANT SELECT, INSERT ON e-commerce.Orders TO 'kasir'@'localhost';
GRANT SELECT, INSERT ON e-commerce.Order_Item TO 'kasir'@'localhost';
REVOKE UPDATE ON e-commerce.Orders FROM 'kasir'@'localhost';
SHOW GRANTS FOR 'kasir'@'localhost';
```

Pembahasan:

Kode ini merupakan serangkaian perintah DCL (Data Control Language) yang mengatur hak akses untuk kasir. Perintah **CREATE USER** membuat akun pengguna baru bernama kasir yang hanya bisa digunakan dari komputer lokal (localhost) dengan kata sandi securepass123. Selanjutnya, melalui perintah **GRANT**, user kasir diberi izin untuk melihat (**SELECT**) dan menambahkan (**INSERT**) data ke dalam tabel Orders dan Order_Item pada database *e-commerce*. Ini berarti kasir dapat mencatat transaksi dan melihat data pesanan, namun tidak bisa melakukan perubahan terhadap data yang sudah ada.

Kemudian, dengan **REVOKE**, hak untuk mengubah data (**UPDATE**) di tabel Orders dicabut dari user tersebut, sehingga kasir tidak bisa memodifikasi isi pesanan yang sudah dicatat. Terakhir, perintah **SHOW GRANTS** digunakan untuk melihat dan memastikan hak akses yang dimiliki oleh user kasir, baik untuk keperluan audit maupun pengelolaan keamanan data.

Printscreen Output :



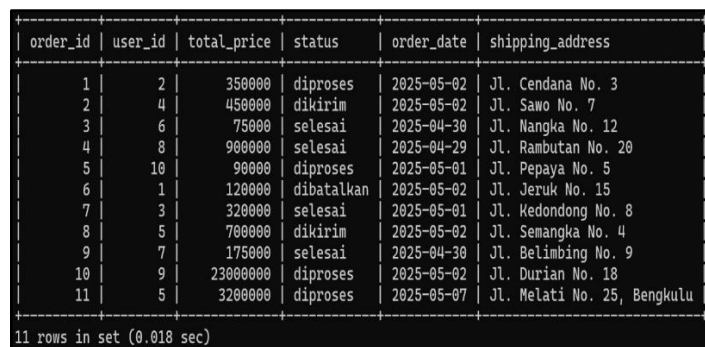
Grants for kasir@localhost	
▶	GRANT USAGE ON *.* TO 'kasir'@'localhost'
	GRANT SELECT, INSERT ON 'ecommerce'. 'order_item' TO 'kasir'@'localhost'
	GRANT SELECT, INSERT ON 'ecommerce'. 'orders' TO 'kasir'@'localhost'

Gambar 4.69 Output Grant

Pembahasan:

Pada gambar ini menunjukkan kasir hanya dapat melakukan **SELECT** dan **INSERT** pada tabel orders dan order_item. Tidak ada hak akses **UPDATE**, **DELETE**, atau hak istimewa lainnya. Pembatasan akses bagi peran kasir, agar mereka hanya bisa melihat dan menambahkan transaksi tanpa dapat mengubah data yang sudah ada.

Printscreen Output :



order_id	user_id	total_price	status	order_date	shipping_address
1	2	350000	diproses	2025-05-02	Jl. Cendana No. 3
2	4	450000	dikirim	2025-05-02	Jl. Sawo No. 7
3	6	75000	selesai	2025-04-30	Jl. Nangka No. 12
4	8	900000	selesai	2025-04-29	Jl. Rambutan No. 20
5	10	90000	diproses	2025-05-01	Jl. Pepaya No. 5
6	1	120000	dibatalkan	2025-05-02	Jl. Jeruk No. 15
7	3	320000	selesai	2025-05-01	Jl. Kedondong No. 8
8	5	700000	dikirim	2025-05-02	Jl. Semangka No. 4
9	7	175000	selesai	2025-04-30	Jl. Belimbing No. 9
10	9	23000000	diproses	2025-05-02	Jl. Durian No. 18
11	5	3200000	diproses	2025-05-07	Jl. Melati No. 25, Bengkulu

11 rows in set (0.018 sec)

Gambar 10.3 Output Grant

Pembahasan:

Pada tabel ini berisi data mengenai transaksi pemesanan dari para pengguna. Setiap baris mewakili satu pesanan, dengan informasi seperti ID pemesanan (order_id), ID pengguna (user_id), total harga pesanan (total_price), status pesanan (status), tanggal pemesanan (order_date), serta alamat pengiriman (shipping_address).

BAB V

KESIMPULAN DAN SARAN

5.1 KESIMPULAN

Melalui proyek ini, telah berhasil dirancang dan diimplementasikan sistem basis data untuk platform *e-commerce* dengan menggunakan MySQL. Sistem ini mencakup berbagai tabel yang saling terhubung untuk mengelola data pengguna, produk, kategori, keranjang belanja, pesanan, pembayaran, pengiriman, serta ulasan pelanggan. Dengan struktur basis data yang terintegrasi, sistem mampu menjaga konsistensi data dan mendukung proses transaksi dari awal hingga akhir.

Penggunaan fitur-fitur lanjutan seperti JOIN, *subquery*, alias, fungsi agregat, serta stored procedure terbukti dapat meningkatkan efisiensi dalam pengelolaan dan penyajian data. Selain itu, proses filtering, pengelompokan data, dan analisis pendapatan dapat dilakukan dengan lebih efektif berkat rancangan basis data yang baik. Secara keseluruhan, proyek ini membuktikan pentingnya desain database yang matang dalam mendukung sistem digital yang kompleks dan dinamis.

5.2 SARAN

Supaya sistem basis data yang sudah dibuat ini bisa berjalan lebih maksimal, ada beberapa hal yang bisa ditingkatkan ke depannya. Pertama, perlu dilakukan pengoptimalan *query* supaya saat data makin banyak, prosesnya tetap cepat dan tidak terhambat. Kedua, soal keamanan juga penting, misalnya dengan mengenkripsi password dan membatasi akses user biar datanya nggak mudah disalahgunakan. Selain itu, sistem bisa ditambah fitur-fitur baru seperti notifikasi otomatis, pelacakan pengiriman yang real-time, atau pengelolaan stok barang yang lebih praktis. Akan lebih baik juga kalau memanfaatkan *view* dan *trigger* untuk bikin sistem lebih rapi dan otomatis. Dan yang nggak kalah penting, sistem ini perlu dites lagi dengan berbagai kondisi biar bisa dipastikan benar-benar stabil saat dipakai beneran.

DAFTAR PUSTAKA

- Pratama, R., & Wijaya, T. (2022). Optimasi Basis Data dengan SQL: Studi Kasus dan Implementasi. Jakarta: Penerbit Informatika.
- Suryani, L. (2021). Pemrograman Basis Data: Konsep dan Aplikasi. Yogyakarta: Andi Offset.
- Wahyudi, F. (2023). Penerapan Data Definition Language dan Data Manipulation Language dalam Sistem Manajemen Basis Data. Surabaya: Pustaka Teknologi
- Ezeria, A., & Siahaan, D. (2022). Laporan Basis Data I: Penerapan DDL dan DML. ResearchGate
- Henderi, Setiyadi, D., & Khasanah, F.N. (2013). Sistem Trigger Database Pada SIAKAD Informatika. Jurnal Sistem dan Teknologi Informasi (JUSTIN), 1(1), 1-10.
- Hidayat, A., & Prasetyo, R. (2019). "Pengaruh Sorting Data dalam Optimalisasi Database". Jurnal Sistem Informasi, 7(2), 55-63.
- Maulana, T. (2020). "Penerapan Grouping dan Agregasi Data dalam Sistem Manajemen Basis Data". Jurnal Informatika, 8(3), 77-85.
- Sari, L. (2021). "Implementasi CASE WHEN dalam Pengolahan Data Akademik". Jurnal Teknologi Informasi, 9(4), 150-160
- Sari, D., Ramadhani, F., & Saputra, H. (2020). Penerapan Stored Procedure untuk Meningkatkan Performa Query dalam E-Commerce. Jurnal Ilmu Komputer, 14(1), 88-99.
- Prasetyo, R., Putri, S., & Yuniarto, B. (2019). Implementasi Join dalam Sistem Basis Data Relasional untuk Pengelolaan Data Mahasiswa. Jurnal Sistem Informasi, 11(3), 34-50

LAMPIRAN

Dokumentasi



Jobdesk:

Laporan dan PPT :

1. Abim Bintang Audio
2. Ajis Saputra Hidayah

Kode dan Laporan:

1. Najwa Nabilah Wibisono
2. Khaylilla Shafaraly Irnanda



1

**KEMENTERIAN RISET, TEKNOLOGI DAN
PENDIDIKAN**

**UNIVERSITAS BENGKULU
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA**

Jl. Wr. Supratman Kandang Limun, Bengkulu Bengkulu
38371 A Telp: (0736) 344087, 22105 – 227

(G1A023065)

- 2. Abim Bintang Audio (G1A023073)
- 3. Khaylilla Shafaraly Irnanda (GA1023079)
- 4. Ajis saputra Hidayah (G1A023083)

Dosen : 1. Dr.Endina Putri Purwandari,Dr.,S.T.,M.Kom
2. Ir. Tiara Eka Putri, S.T.,M.Kom.

Asisten : 1. Merly Yuni Purnama (G1A022006)
2. Reksi Hendra Pratama (G1A022032)
3. Sinta Ezra Wati Gulo (G1A022040)
4. Fadlan Dwi Febrio (G1A022051)
5. Torang Four Yones Manullang (G1A022052)
6. Wahyu Ozorah Manurung (G1A022060)
7. Shalaudin Muhammad Sah (G1A022070)
8. Dian Ardiyanti Saputri (G1A022084)

Laporan Tugas Akhir

Catatan dan Tanda Tangan

Laporan Tugas Besar