Laporan Dokumentasi Proses

Pembuatan Database untuk E-commerce

- **Perancangan Struktur Database**

Pada tahap perancangan struktur database, kami mendesain database untuk mendukung aplikasi e-commerce. Berikut adalah tahapan yang dilakukan:

1. Normalisasi

Proses normalisasi dilakukan untuk memastikan bahwa data yang tersimpan dalam database tidak memiliki redundansi dan bergantung pada aturan yang benar (hingga bentuk normal ke-3). Berikut adalah langkah-langkah yang diterapkan:

1. **Tabel Categories:** Berfungsi untuk menyimpan kategori produk (misalnya Laptop, Smartphone, dll).
2. **Tabel Products:** Menyimpan informasi produk seperti nama, harga, deskripsi, dan kategori produk.
3. **Tabel Customers:** Menyimpan informasi pelanggan, seperti nama dan email.
4. **Tabel Orders:** Menyimpan informasi tentang pesanan yang dilakukan oleh pelanggan.

2. Entity Relationship Diagram (ERD)

ERD menggambarkan relasi antar entitas dalam database. Relasi utama antara tabel adalah:

- **Customers ↔ Orders** (Satu pelanggan bisa melakukan banyak pesanan)
- **Products ↔ Orders** (Satu produk bisa dipesan banyak kali)
- **Categories ↔ Products** (Satu kategori bisa memiliki banyak produk)

3. Pembuatan Struktur Tabel

Berikut adalah SQL untuk pembuatan tabel yang telah diimplementasikan:

```sql
-- Tabel Categories
CREATE TABLE Categories (
    CategoryID INT AUTO_INCREMENT PRIMARY KEY,
    CategoryName VARCHAR(100) NOT NULL
);
```

```sql
-- Tabel Products
CREATE TABLE Products (
    ProductID INT AUTO_INCREMENT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL,
    Price DECIMAL(10, 2) NOT NULL,
    Description TEXT,
    CategoryID INT,
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);
```

```sql
-- Tabel Customers
CREATE TABLE Customers (
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) NOT NULL UNIQUE
);
```

```sql
-- Tabel Orders
CREATE TABLE Orders (
    OrderID INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT NOT NULL,
    ProductID INT NOT NULL,
    OrderDate DATETIME NOT NULL,
    Quantity INT NOT NULL CHECK (Quantity > 0),
    TotalPrice DECIMAL(10, 2) NOT NULL CHECK (TotalPrice >= 0),
    OrderStatus ENUM('Pending', 'Shipped', 'Completed', 'Cancelled') DEFAULT 'Pending',
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

- **Query SQL untuk Proses Data**

Berikut adalah contoh query SQL yang digunakan untuk memproses data pada e-commerce Tokopedia.

a. Menggunakan JOIN untuk Menggabungkan Tabel

Query ini digunakan untuk menggabungkan data dari tabel **Orders**, **Customers**, dan **Products** untuk mendapatkan informasi lengkap tentang pesanan, nama pelanggan, dan produk yang dipesan.

```sql
SELECT
    o.OrderID,
    c.Name AS CustomerName,
    p.ProductName AS ProductName,
    o.OrderDate,
    o.Quantity,
    o.TotalPrice,
    o.OrderStatus
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID;
```

Penjelasan Fungsionalitas Query:

- **JOIN** digunakan untuk menggabungkan tabel **Orders** dengan **Customers** dan **Products**.
- Menampilkan data pesanan lengkap, termasuk nama pelanggan dan nama produk yang dipesan.

Hasil Output:

OrderID	CustomerName	ProductName	OrderDate	Quantity	TotalPrice	OrderStatus
1	John Doe	MacBook	2024-12-01 14:30:00	1	15000000.00	Completed
2	Jane Smith	Gaming Laptop	2024-12-03 10:45:00	2	30000000.00	Shipped
3	Alice Brown	Flip Laptop	2024-12-05 16:20:00	1	17000000.00	Pending

b. Menggunakan Subquery untuk Penyelesaian Kebutuhan yang Lebih Kompleks

Query ini digunakan untuk menampilkan pesanan yang memiliki total harga lebih tinggi dari rata-rata harga pesanan.

```sql
SELECT
    o.OrderID,
    c.Name AS CustomerName,
    p.ProductName AS ProductName,
    o.OrderDate,
    o.Quantity,
    o.TotalPrice,
    o.OrderStatus
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID
WHERE o.TotalPrice > (
    SELECT AVG(TotalPrice) FROM Orders
);

```

Penjelasan Fungsionalitas Query:

- Subquery digunakan untuk menghitung rata-rata harga total pesanan dan menampilkan pesanan yang melebihi nilai rata-rata tersebut.

Hasil Output (Contoh):

OrderID	CustomerName	ProductName	OrderDate	Quantity	TotalPrice	OrderStatus
2	Jane Smith	Gaming Laptop	2024-12-03 10:45:00	2	30000000.00	Shipped

- **Implementasi Database Objects**

Berikut adalah implementasi minimal 3 jenis **Database Object** dan penjelasan penggunaannya.

1. View

Untuk membuat **View** yang menampilkan semua pesanan lengkap dengan informasi pelanggan dan produk yang dipesan:

```
CREATE VIEW OrderDetails AS
SELECT
    o.OrderID,
    c.Name AS CustomerName,
    p.ProductName AS ProductName,
    o.OrderDate,
    o.Quantity,
    o.TotalPrice,
    o.OrderStatus
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
JOIN Products p ON o.ProductID = p.ProductID;
```

Kasus Penggunaan:

- **View** ini digunakan untuk menampilkan detail pesanan secara mudah tanpa perlu menjalankan query join setiap kali.

Manfaat dalam Proses Bisnis:

- Mempermudah pemantauan dan laporan pesanan.

2. Stored Procedure

Stored Procedure untuk memperbarui status pesanan berdasarkan OrderID:

```
CREATE PROCEDURE UpdateOrderStatus(IN orderID INT, IN newStatus ENUM('Pending', 'Shipped',
'Completed', 'Cancelled'))
BEGIN
    UPDATE Orders
    SET OrderStatus = newStatus
    WHERE OrderID = orderID;
END);

```

Kasus Penggunaan:

- Digunakan untuk memperbarui status pesanan berdasarkan ID pesanan.

Manfaat dalam Proses Bisnis:

- Mempercepat pembaruan status pesanan tanpa harus menjalankan query manual.

3. Trigger

Trigger untuk mengupdate TotalPrice pada tabel Orders setiap kali data baru dimasukkan atau diperbarui:

Kasus Penggunaan:

- **Trigger** ini secara otomatis menghitung dan memperbarui nilai total harga saat pesanan baru dimasukkan ke dalam tabel Orders.

Manfaat dalam Proses Bisnis:

- Mengurangi kesalahan manual dalam perhitungan harga total dan otomatisasi pengolahan data.

Kesimpulan

Proses perancangan database dan implementasi objek database ini memberikan pondasi yang kuat untuk mendukung aplikasi e-commerce. Setiap objek database seperti **View**, **Stored Procedure**, dan **Trigger** berfungsi untuk meningkatkan efisiensi, mengurangi redundansi, dan memastikan proses bisnis berjalan otomatis dan terstruktur dengan baik.