Minimarket Database Design Using Plan with MySQL Workbench

Group 8

Names : Haikal Bagaskara Syopian

Muhammad Nur Falah

Muhammad Iqbal

Class : ICC2

CEP CCIT

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

2024

PROJECT ON

Minimarket Database Design Using Plan with MySQL Workbench

Developed by

1. **Haikal Bagaskara Syopian**
2. **Muhammad Nur Falah**
3. **Muhammad Iqbal**

Minimarket Database Design Using Plan with MySQL Workbench

Batch Code : ICC2

Start Date : 20 December 2024

End Date : 30 December 2024

Name of Faculty : Muhammad Riza Iqbal

Names of Developer:

1. Haikal Bagaskara Syopian
2. Muhammad Nur Falah
3. Muhammad Iqbal

Date of Submission:30 December 2024

CERTIFICATE

This is to certify that this report titled “Minimarket Database Design Using Plan with MySQL Workbench” embodies the original work done by Muhammad Nur Falah, Haikal Bagaskara Syopian, and Muhammad Iqbal. Project in partial fulfillment of their course requirement at CEP-CCIT FTUI.

Coordinator:
Muhammad Riza Iqbal

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this professional paper project. First and foremost, I extend my deepest appreciation to my lecturer Muhammad Riza Iqbal for their invaluable guidance, support, and mentorship throughout this endeavor. Their expertise and feedback were instrumental in shaping this project. I am also thankful to my colleagues and peers who provided insightful discussions and constructive criticism, helping me refine my ideas and arguments.

Lastly, I owe a debt of gratitude to my family and friends for their unwavering encouragement and understanding during the time I dedicated to this project. Without their support, this accomplishment would not have been possible. Thank you all for your contributions, encouragement, and support.

Depok, 30 December 2024

Author

BACKGROUND

In the modern era, the retail industry has undergone a dramatic transformation driven by technological advancements, shifting consumer behaviours, and changing economic landscapes. Among the various forms of retail, minimarkets have emerged as one of the most popular formats, offering a wide range of convenience products in urban and suburban areas. These stores, which typically operate in smaller spaces, are characterized by their quick service, extensive product assortment, and strategic locations.

The rise of minimarkets can be attributed to several factors, with consumer convenience being at the forefront. In a fast-paced world where time is often limited, consumers increasingly prefer the ability to make quick, one-stop shopping trips to buy everyday items, such as groceries, snacks, beverages, and personal care products. Minimarkets meet this demand by offering a compact yet diverse selection of goods, typically located near residential areas, offices, and busy commercial zones.

SYSTEM ANALYSIS

Minimarket are essential in this era of convenient market with variable stock to help community around them. But to run a decent market, you need a system. This system will decide how a minimarket can improve and stay in business. That's why our group decide to create a Minimarket System about the most important part of minimarket, which is transaction between a minimarket to customer.

Transaction contains many elements, some of them are Customer, Employees, Product, Transactions, and Detail Transaction. In this report we will show how these elements can create a fully functional transaction for everyday minimarket activities.

DATABASE DESIGN

Database Name : db_minimarket_system

Number of Tables : 5
- customer
- employees
- products
- transaction_details
- transactions

Number of View : 3
- vw_nota_TransactionDetails
- vw_DataPenjualanProduct
- vw_DataPenjualanPerhari

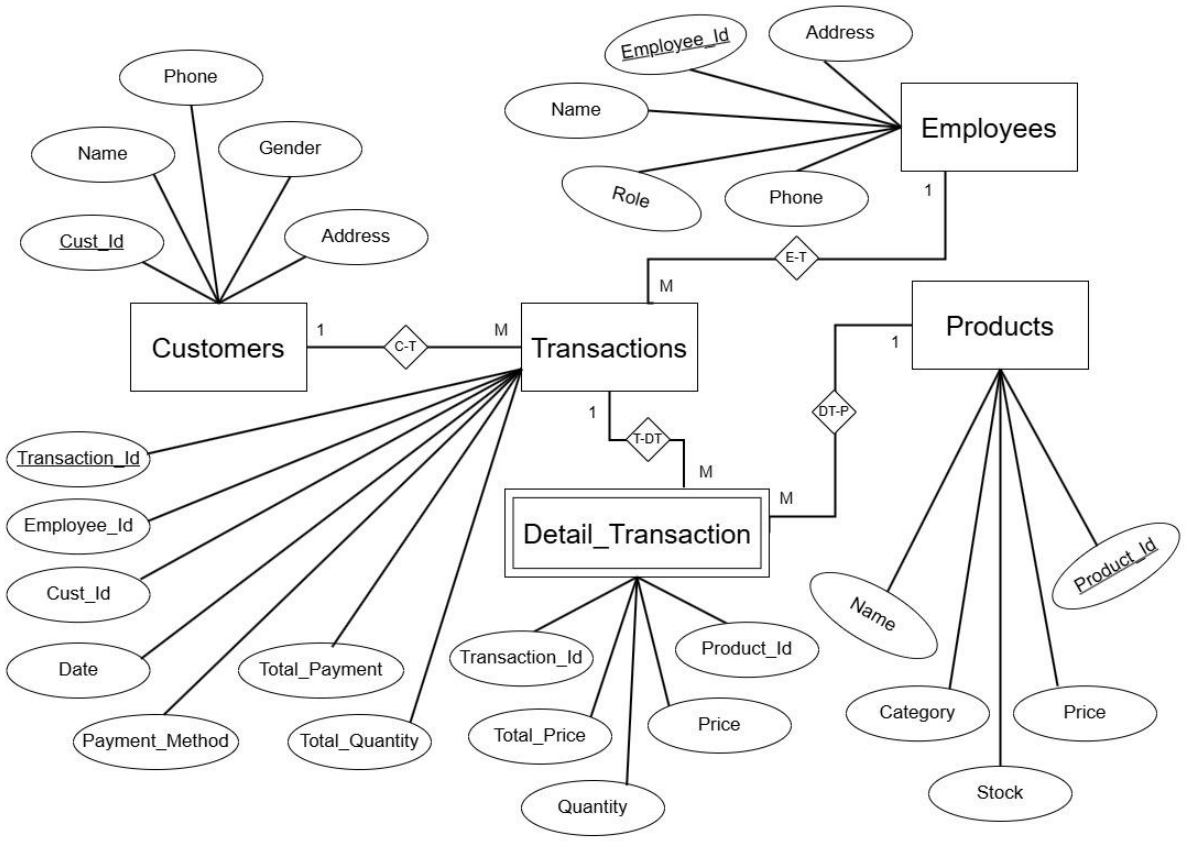
Number of Procedure : 3
- proc_AddTransaction
- proc_AddProduct
- proc_AddStock
- proc_AddCustomer

Number of Triggers : 4
- Trg_UpdateTransactionTotals
- Trg_ValidateTransactionData

Number of Index : 2
- idx_name
- ft_idx_name

ERD (Entity Relationship Diagram)

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. And here is the ERD of the Minimarket database.



NORMALIZATION TABLES

In ERD there is normalization where each element depends on the primary key to connect them between each table. The following picture are Normalization Tables from our Minimarket System.

3NF

Customer Table

Cust_ID	Cust_Name	Cust_Add	Cust_Phone_num	Gender
C0001	JOHN	JAKSEL	87656574322	MEN
C0002	NANA	BOGOR	87665543212	WOMEN
C0003	DEDE	JAKSEL	89765431122	MEN

Employee Table

Employee_ID	Employee_Name	Employee_Add	Employee_Phone_Num	Role
E0001	MIKAEL	JAKSEL	89765331122	Supervisor
E0002	NISA	JAKPUT	89865331112	Kasir
E0003	ASEP	DEPOK	86815331212	Kasir

Transaction Table

Transaction_ID	Employee_ID	Cust_ID	Date
T0001	E0001	C0001	12/28/2024 22.40
T0002	E0002	C0002	12/28/2024 22.42
T0003	E0003	C0003	12/28/2024 22.44

Transaction_ID	Total_Quantity	Total_Payment	Payment_Method
T0001	3	11000	Cash
T0002	12	49000	Debit
T0003	5	25000	Qris

Detail Trans

Transaction_ID	Product_ID	Price	Quantity	Total_Price
T0001	P0001	3000	2	6000
T0001	P0002	5000	1	5000
T0002	P0001	3000	3	9000
T0002	P0002	5000	4	20000
T0002	P0004	4000	5	20000
T0003	P0002	5000	5	25000

Product Table

Product_ID	Product_Name	Stock	Category	price
P0001	MIE INSTAN	90	MAKANAN	3000
P0002	AIR MINERAL	135	MINUMAN	5000
P0003	SUSU UHT	50	MINUMAN	12000
P0004	TISU	14	KEBUTUHAN RUMAH	4000

ERD DESIGN

In the database there is a top down approach, namely ERD tables, which is to implement the Database. The following picture are ERD Tables from our Minimarket System.

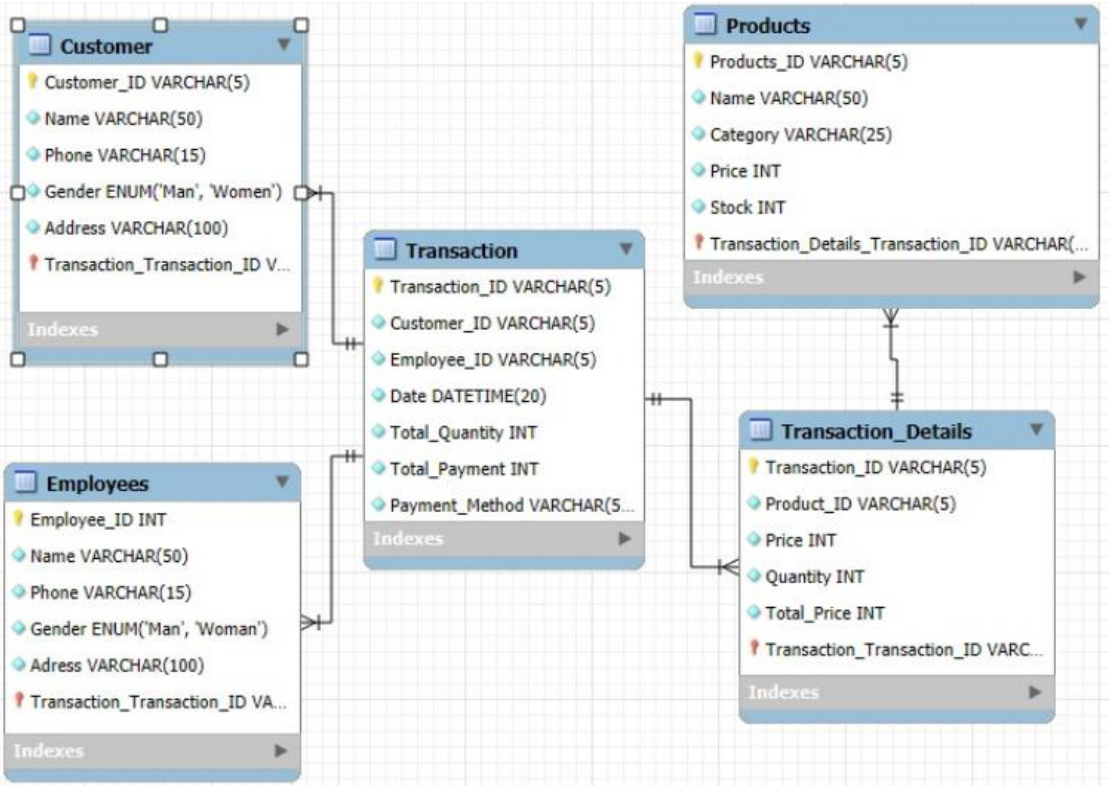


TABLE DESIGN

The table design of the coffee shop database is as follows:

Table Customer

Field	Type	Null	Key	Default	Extra
cust_id	varchar(5)	NO	PRI	NULL	
name	varchar(50)	YES		NULL	
phone	varchar(15)	YES		NULL	
gender	enum('Men','Women')	NO		NULL	
address	varchar(100)	YES		NULL	

Table Employees

Field	Type	Null	Key	Default	Extra
employee_id	varchar(5)	NO	PRI	NULL	
name	varchar(50)	NO		NULL	
role	varchar(20)	NO		NULL	
phone	varchar(15)	NO		NULL	
address	varchar(100)	NO		NULL	

Table Product

Field	Type	Null	Key	Default	Extra
product_id	varchar(5)	NO	PRI	NULL	
name	varchar(50)	NO		NULL	
category	varchar(25)	NO		NULL	
price	int	NO		NULL	
stock	int	NO		NULL	

TABLE DESIGN

Table Transaction_Details

Field	Type	Null	Key	Default	Extra
transaction_id	varchar(5)	NO	MUL	NULL	
product_id	varchar(5)	NO	MUL	NULL	
price	int	NO		NULL	
quantity	int	NO		NULL	
total_price	int	NO		NULL	

Table Transactions

Field	Type	Null	Key	Default	Extra
transaction_id	varchar(5)	NO	PRI	NULL	
cust_id	varchar(5)	NO	MUL	NULL	
employee_id	varchar(5)	NO	MUL	NULL	
date	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
total_quantity	int	NO		NULL	
total_payment	int	NO		NULL	
payment_method	varchar(50)	NO		NULL	

VALIDATION PERFORM

Customer Table

Column	Method(s) for user validation	Description
Customer_ID **(PK)**	Primary Key, **NOT NULL**	**This column contain Primary Key, cannot be empty**
Customer_Name	**NOT NULL**	**This column cannot be empty**
Customer_Address	**NOT NULL**	**This column cannot be empty**
Customer_Phone_Number	**NOT NULL**	**This column cannot be empty**
Customer_Gender	**NOT NULL**	**This column cannot be empty**

Transaction Table

Column	Method(s) for user validation	Description
Transaction_ID **(PK)**	Primary Key, **NOT NULL**	**This column contain Primary Key, cannot be empty**
Customer_ID **(FK)**	Foreign Key, **NOT NULL**	**This column contain Foreign Key, references from Employees table cannot be empty**
Employee_ID **(FK)**	Foreign Key, **NOT NULL**	**This column contain Foreign Key, references from Customers table cannot be empty**
Date	**NOT NULL**	**This column cannot be empty**
Payment_Method	**NOT NULL**	**This column cannot be empty**
Total_Quantity	**NOT NULL**	**This column cannot be empty**

Products Table

Column	Method(s) for user validation	Description
Product_ID **(PK)**	Primary Key, **NOT NULL**	**This column contain Primary Key, cannot be empty**
Product_Name	**NOT NULL**	**This column cannot be empty**
Price	**NOT NULL**	**This column cannot be empty**
Category	**NOT NULL**	**This column cannot be empty**
Stock	**NOT NULL**	**This column cannot be empty**

VALIDATION PERFORM

Employees Table

Column	Method(s) for user validation	Description
Employee_ID **(PK)**	Primary Key, **NOT NULL**	**This column contain Primary Key, cannot be empty**
Employee_Name	**NOT NULL**	**This column cannot be empty**
Employee_Address	**NOT NULL**	**This column cannot be empty**
Employee_Phone_Number	**NOT NULL**	**This column cannot be empty**
Role	**NULL**	**This column can be empty**

Detail_Transaction Table

Column	Method(s) for user validation	Description
Transaction_ID **(PK)**	Primary Key, **NOT NULL**	**This column contain Primary Key, cannot be empty**
Product_ID **(FK)**	Foreign Key, **NOT NULL**	**This column contain Foreign Key, references from Product table cannot be empty**
Price	**NOT NULL**	**This column cannot be empty**
Quantity	**NOT NULL**	**This column cannot be empty**

STORE PROCEDURE

Query of Procedure Add Transaction:

```
-- Create Prosedure add transaction
DELIMITER //
create procedure proc_AddTransaction (
    in p_transaction_id varchar(5),
    in p_cust_id varchar(5),
    in p_employee_id varchar(5),
    in p_payment_method varchar(50),
    in p_product_ids text, -- Format: 'P0001,P0002'
    in p_quantities text   -- Format: '2,1'
)
BEGIN
    declare current_product_id varchar(10);
    declare current_quantity int;
    declare current_price int;
    declare current_stock int;
    declare total_price int;
    declare total_quantity int default 0;
    declare total_payment int default 0;
```

STORE PROCEDURE

```
-- Simpan salinan awal p_product_ids dan p_quantities

  declare original_product_ids text default p_product_ids;

  declare original_quantities text default p_quantities;

-- Loop untuk validasi produk dan stok sebelum insert ke table
transactions

  while length(p_product_ids) > 0 DO

-- Ambil produk pertama dan jumlahnya

  set current_product_id = SUBSTRING_INDEX(p_product_ids, ',',
1);

  set current_quantity = CAST(SUBSTRING_INDEX(p_quantities, ',',
1) as unsigned);


-- Potong produk dan jumlah yang telah diproses

  set p_product_ids = if(LENGTH(p_product_ids) =
LENGTH(current_product_id), '', SUBSTRING(p_product_ids,
LENGTH(current_product_id) + 2));

  set p_quantities = if(LENGTH(p_quantities) =
LENGTH(SUBSTRING_INDEX(p_quantities, ',', 1)), '',
SUBSTRING(p_quantities, LENGTH(SUBSTRING_INDEX(p_quantities,
',', 1)) + 2));


-- Ambil harga dan stok produk

  select price, stock into current_price, current_stock

  from products

  where product_id = current_product_id;
```

STORE PROCEDURE

```
-- Validasi apakah produk ditemukan
  if current_price is null then
  set @error_message = CONCAT('Product ID not found: ',
current_product_id);
  signal sqlstate '45000'
  set message_text = @error_message;
  end if;


-- Validasi apakah stok mencukupi
  if current_stock < current_quantity then
  set @error_message = CONCAT('Insufficient stock for product
ID: ', current_product_id,
                            '. Available stock: ', current_stock);
   signal sqlstate '45000'
   set message_text = @error_message;
   end if;


-- Update total quantity dan payment sementara
   set total_quantity = total_quantity + current_quantity;
   set total_payment = total_payment + (current_price *
current_quantity);
   end while;
```

STORE PROCEDURE

```
-- Jika total_quantity = 0 setelah validasi semua produk

  if total_quantity = 0 then

  signal sqlstate '45000'

  set message_text = 'Transaction failed: no valid products or
insufficient stock.';

  end if;


-- Insert ke tabel transactions setelah validasi

  insert into transactions (transaction_id, cust_id,
employee_id, total_quantity, total_payment, payment_method)

  values (p_transaction_id, p_cust_id, p_employee_id, 0, 0,
p_payment_method);


-- Gunakan kembali salinan asli untuk loop kedua

  set p_product_ids = original_product_ids;

  set p_quantities = original_quantities;

-- Loop kedua untuk memasukkan detail transaksi

  while LENGTH(p_product_ids) > 0 do

-- Ambil produk pertama dan jumlahnya

  set current_product_id = SUBSTRING_INDEX(p_product_ids, ',',
1);

  set current_quantity = CAST(SUBSTRING_INDEX(p_quantities, ',',
1) as unsigned);
```

STORE PROCEDURE

```
-- Potong produk dan jumlah yang telah diproses

  set p_product_ids = if(LENGTH(p_product_ids) =
LENGTH(current_product_id), '',

  SUBSTRING(p_product_ids, LENGTH(current_product_id) + 2));

  set p_quantities = if(LENGTH(p_quantities) =
LENGTH(SUBSTRING_INDEX(p_quantities, ',', 1)), '',

  SUBSTRING(p_quantities, LENGTH(SUBSTRING_INDEX(p_quantities,
',', 1)) + 2));


-- Ambil harga produk

  select price into current_price

  from products

  where product_id = current_product_id;


-- Hitung total harga

  set total_price = current_price * current_quantity;


-- Masukkan ke tabel transaction_details

  insert into transaction_details (transaction_id, product_id,
price, quantity, total_price)

  values (p_transaction_id, current_product_id, current_price,
current_quantity, total_price);
```

STORE PROCEDURE

```
-- Update stok produk
  update products
  set stock = stock - current_quantity
  where product_id = current_product_id;
  end while;


-- Update tabel transactions dengan total quantity dan payment
  update transactions
  set total_quantity = total_quantity, total_payment =
total_payment
  where transaction_id = p_transaction_id;
END //
DELIMITER ;
```

STORE PROCEDURE

Query of Procedure Add Product:

```
-- Create Procedure Add Product
DELIMITER $$
create procedure proc_AddProduct(
    in p_product_id varchar(5),
    in p_name varchar(50),
    in p_category varchar(25),
    in p_price int,
    in p_stock int
)
BEGIN
-- 1. Validasi apakah ID produk sudah ada
  if exists (select 1 from products where product_id =
p_product_id) then
  signal sqlstate '45000' set message_text = 'Produk dengan ID
tersebut sudah ada.';
  end if;


-- 2. Validasi apakah nama produk sudah ada
  if exists (select 1 from products where name = p_name) then
  signal sqlstate '45000' set message_text = 'Produk dengan nama
tersebut sudah ada.';
  end if;
```

STORE PROCEDURE

```
-- 3. Validasi apakah stok produk valid (tidak kosong atau
negatif)

  if p_stock <= 0 then

  signal sqlstate '45000' set message_text = 'Stok produk tidak
boleh kosong atau negatif.';

  end if;


-- 4. Validasi apakah harga produk valid (tidak negatif)

    if p_price <= 0 then

    signal sqlstate '45000' set message_text = 'Harga produk
tidak boleh negatif.';

    end if;

-- Menambahkan produk jika tidak ada error

  insert into products (product_id, name, category, price,
stock)

  values (p_product_id, p_name, p_category, p_price, p_stock);

END $$

DELIMITER ;
```

STORE PROCEDURE

Query of Procedure Add Stock:

```
-- Create Procedure Add Stock Product

DELIMITER ++

create procedure proc_AddStock(

    in p_product_id varchar(5),

    in p_add_stock int

)

BEGIN

-- 1. Validasi apakah ID produk ada dalam tabel products

  if not exists (select 1 from products where product_id =
p_product_id) then

  signal sqlstate '45000' set message_text = 'Produk dengan ID
tersebut tidak ditemukan.';

  end if;


 -- 2. Validasi apakah jumlah stok yang akan ditambahkan valid
(tidak negatif atau kosong)

  IF p_add_stock <= 0 THEN

  signal sqlstate '45000' set message_text = 'Jumlah stok yang
ditambahkan tidak boleh kosong atau negatif.';

  end if;
```

STORE PROCEDURE

```
-- 3. Menambahkan stok ke produk yang valid
  update products
  set stock = stock + p_add_stock
  where product_id = p_product_id;
END ++
DELIMITER ;
```

STORE PROCEDURE

Query of Procedure Add Customer:

```
-- Create Procedure Add Cust
DELIMITER \\
create procedure proc_AddCustomer(
   in p_cust_id varchar(5),
   in p_name varchar(50),
   in p_phone varchar(15),
   in p_gender enum('Men', 'Women'),
   in p_address varchar(100)
)
BEGIN
-- 1. Validasi apakah ID pelanggan sudah ada
  if exists (select 1 from customers where cust_id = p_cust_id)
then
  signal sqlstate '45000' set message_text = 'ID pelanggan
sudah digunakan.';
  end if;
-- 2. Validasi apakah nama pelanggan sudah ada
  if exists (select 1 from customers where name = p_name) then
  signal sqlstate '45000' set message_text = 'Nama pelanggan
sudah digunakan.';
  end if;
```

STORE PROCEDURE

```
-- 3. Validasi apakah nomor telepon pelanggan sudah ada
  if exists (select 1 from customers where phone = p_phone) then
  signal sqlstate '45000' set message_text = 'Nomor telepon
pelanggan sudah digunakan.';
  end if;


-- 4. Validasi apakah gender sesuai dengan nilai yang
diperbolehkan
  if p_gender not in ('Men', 'Women') then
  signal sqlstate '45000' set message_text= 'Gender hanya boleh
Men atau Women.';
  end if;


-- 5. Menambahkan pelanggan baru ke tabel customers
  insert into customers (cust_id, name, phone, gender, address)
  values (p_cust_id, p_name, p_phone, p_gender, p_address);
END \\
DELIMITER ;
```

VIEW

In this database there is also a view named "vw_nota_TransactionDetails".

```
select * from vw_nota_TransactionDetails;
```

Kode Transaksi	Nama Pembeli	Tanggal Transaksi	Kode Product	Nama Product	Harga Product	Jumlah Product	Total Harga	Nama Yang Melayani	Jaba
T0001	John	2024-12-30 07:40:30	P0001	Mie Instan	3000	2	11000	Mikael	Super
T0001	John	2024-12-30 07:40:30	P0002	Air Mineral	5000	1	11000	Mikael	Super
T0002	Nana	2024-12-30 07:40:55	P0001	Mie Instan	3000	3	49000	Nisa	Kasir
T0002	Nana	2024-12-30 07:40:55	P0002	Air Mineral	5000	4	49000	Nisa	Kasir
T0002	Nana	2024-12-30 07:40:55	P0004	Tisu Gulung	4000	5	49000	Nisa	Kasir
T0003	Dede	2024-12-30 07:40:56	P0002	Air Mineral	5000	5	25000	Nisa	Kasir
T0004	Dodi	2024-12-30 07:41:00	P0001	Mie Instan	3000	5	52000	Anna	Kasir
T0004	Dodi	2024-12-30 07:41:00	P0009	Pocari	6000	2	52000	Anna	Kasir
T0004	Dodi	2024-12-30 07:41:00	P0005	Keripik Singkong	5000	5	52000	Anna	Kasir
T0005	Keke	2024-12-30 07:41:01	P0002	Air Mineral	5000	5	25000	Anna	Kasir

Query:

```
create view vw_nota_TransactionDetails as
select  t.transaction_id as 'Kode Transaksi',
        c.name as 'Nama Pembeli',
        t.date as 'Tanggal Transaksi',
        dt.product_id as 'Kode Product',
        p.name as 'Nama Product',
        p.price as 'Harga Product',
        dt.quantity as 'Jumlah Product',
        t.total_payment as 'Total Harga',
        e.name as 'Nama Yang Melayani',
        e.role as 'Jabatan' from transactions t
join customers c on c.cust_id = t.cust_id
join employees e on e.employee_id = t.employee_id
join transaction_details dt on dt.transaction_id =
t.transaction_id
join products  p on p.product_id = dt.product_id;
```

VIEW

In this database there is also a view named "vw_DataPenjualanProduct".

```
select * from vw_DataPenjualanProduct;
```

Kode Product	Nama Product	Harga Product	Jumlah Terjual	Total Penjualan
P0001	Mie Instan	3000	10	30000
P0002	Air Mineral	5000	15	75000
P0004	Tisu Gulung	4000	5	20000
P0009	Pocari	6000	2	12000
P0005	Keripik Singkong	5000	5	25000

Query:

```
create view vw_DataPenjualanProduct as
select  p.product_id as 'Kode Product',
        p.name as 'Nama Product',
        p.price as 'Harga Product',
        sum(dt.quantity) as 'Jumlah Terjual',
        sum(dt.quantity * dt.price) as 'Total Penjualan' from
products p
join transaction_details dt on dt.product_id = p.product_id
group by p.product_id, p.name, p.price;
```

VIEW

In this database there is also a view named "vw_DataPenjualanPerhari".

```
select * from vw_DataPenjualanPerhari;
```

	Tanggal Tranasaksi	Jumlah Transaksi	Jumlah Product Terjual	Total Pendapatan
▶	2024-12-30	5	37	162000

Query:

```
create view vw_DataPenjualanPerhari as
select  date(t.date) as 'Tanggal Tranasaksi',
        count(t.transaction_id) as 'Jumlah Transaksi',
        sum(t.total_quantity) as 'Jumlah Product Terjual',
        sum(t.total_payment) as 'Total Pendapatan' from
transactions t
group by date(t.date) order by 'Tanggal Tranasaksi';
```

FULLTEXT SEARCH

Fulltext Search

```
alter table products
add index idx_name (name);

alter table products
add fulltext index ft_idx_name (name);

select*from products where match(name)
against('sabun' in natural language mode);
```

Index

	product_id	name	category	price	stock
▶	P0013	Sabun Mandi	Produk Rumah Tangga	4000	50
	P0014	Sabun Cuci Piring	Produk Rumah Tangga	12000	45
*	NULL	NULL	NULL	NULL	NULL

TRIGGER

Trigger update transactions after transaction_details insert

```
-- Create Trigger
DELIMITER |
-- Trigger untuk menghitung ulang total transaksi setelah INSERT atau UPDATE pada transaction_details
create trigger trg_UpdateTransactionTotals
after insert on transaction_details
for each row
BEGIN
    declare calculated_quantity int default 0;
    declare calculated_payment int default 0;

    -- Hitung ulang total kuantitas dan pembayaran
    select SUM(quantity), SUM(total_price)
    into calculated_quantity, calculated_payment
    from transaction_details
    where transaction_id = new.transaction_id;

    -- Update tabel transactions
    update transactions
    set total_quantity = calculated_quantity, total_payment = calculated_payment
    where transaction_id = new.transaction_id;
END |
DELIMITER ;
```

Trigger Validation Before transactions insert

```
DELIMITER ||
-- Trigger untuk validasi data transaksi sebelum INSERT atau UPDATE pada transactions
create trigger trg_ValidateTransactionData
before insert on transactions
for each row
BEGIN
    -- Validasi ID transaksi
    if CHAR_LENGTH(NEW.transaction_id) != 5 then
        signal sqlstate '45000' set message_text = 'Format ID transaksi tidak valid (harus 5 karakter).';
    end if;

    -- Validasi ID pelanggan
    if not exists (select 1 from customers where cust_id = new.cust_id) then
        signal sqlstate '45000' set message_text = 'ID pelanggan tidak ditemukan.';
    end if;

    -- Validasi ID karyawan
    if not exists (select 1 from employees where employee_id = new.employee_id) then
        signal sqlstate '45000' set message_text = 'ID karyawan tidak ditemukan.';
    end if;

    -- Validasi total pembayaran
    if new.total_payment < 0 then
        signal sqlstate '45000' set message_text = 'Total pembayaran tidak boleh negatif.';
    end if;
END ||
DELIMITER ;
```

CONFIGURATION

Hardware : Laptop

Operating System : Windows 11

Software : Microsoft Word 2019
MySQL Workbench

Project File Details		
No	**File Name**	**Remarks**
1	Project SQL.pdf	PDF that contains report
2	db_minimarket_system.sql	Database File
3	MINIMARKET DB PROJECT.pdf	PDF that contains presentation Slide