

**DATABASE IN DUMDUM**

Group 4

Names : Leann Nataly Kenan Pakpahan

Salma Aufa Ramadhanti Iswara

Techi Kariska Sari

Class : 1CC8

**CEP CCIT**

**FAKULTAS TEKNIK UNIVERSITAS INDONESIA**

**2023**

**Developed by**

1. **Leann Nataly Kenan Pakpahan**
2. **Salma Aufa Ramadhanti Iswara**
3. **Techi Kariska Sari**

****

**PROJECT ON**

**DATABASE IN DUMDUM**

**Database in DumDum**

Batch Code : 1CC8

Start Date : December 14, 2023  
End Date : December 28, 2023

Name of Faculty : Tri Agus Riyadi, S.Kom, M.Kom

Names of Developer :

1. Leann Nataly Kenan Pakpahan
2. Salma Aufa Ramadhanti Iswara
3. Techi Kariska Sari

Date of Submission: December 28, 2023

**CERTIFICATE**

This is to certify that this report titled “Database in DumDum” embodies the original work done by Leann Nataly Kenan Pakpahan, Salma Aufa Ramadhanti Iswara and Techi Kariska Sari. Project in partial fulfillment of their course requirement at NIIT.

Coordinator:

CEP CCIT

****

**ACKNOWLEDGEMENT**

We are grateful to Allah, the Almighty God, for giving us the abilities to fulfill our work without incident. Blessings and greetings are also sent to Muhammad SAW, our final and greatest prophet. Not to mention, we are grateful for our parents who have continued to support us both morally and financially till this point. We would like to thank Tri Agus Riyadi, S.Kom, M.Kom. as our supervisor who always helps us to finish this paper. Also acknowledged are our friends who have assisted us up until quite recently.

This paper was prepared to fulfill our project assignment as CEP CCIT FTUI students. The topic that will be discussed in this paper mechanism Database in DumDum works such as Entity Relationship Diagram, Scematic Diagram, Table Design etc. We hope this paper will be beneficial to all those who read, especially in education of information technology.

The author realizes that this paper is far from perfect, so it will be very helpful if colleagues and supervisors are willing to provide suggestions and criticism for the perfection of this paper.

**SYSTEM ANALYSIS**

**System Summary:**

Database is a management process that usually used in an organization, office, business and others. In a project entitled “Database in DumDum “. We will show to readers the different database-related products we have developed for lecture systems. ERD, Database Design, Schematic Diagrams, Table Design, and Conducted Validations are the topics we’ll cover.

**System Process:**

We will start by giving an ERD perspective, ERD is a visual representation of the relationship between the tables in this project. Following that, we created a design for a database that includes equation relationship in a table with the main view consisting of things that may be put in a table. Start creating a Schematic Diagram after the Database Design is complete. This schematic diagram serves as a display of a database’s grouping and sorting process in terms of flow or sequence. So that it will be simpler for us to modify a database’s work with this methodical approach. The next page is Validation Performed, where a system for validating data that will be entered into the database is displayed.

**Databases Name : DUMDUM**

**Databases Schema : DUMDUM**

**Number of Schema : 2**

Schema Human

Schema Product

**Number of Tables :6**

Table employee

Table Customer

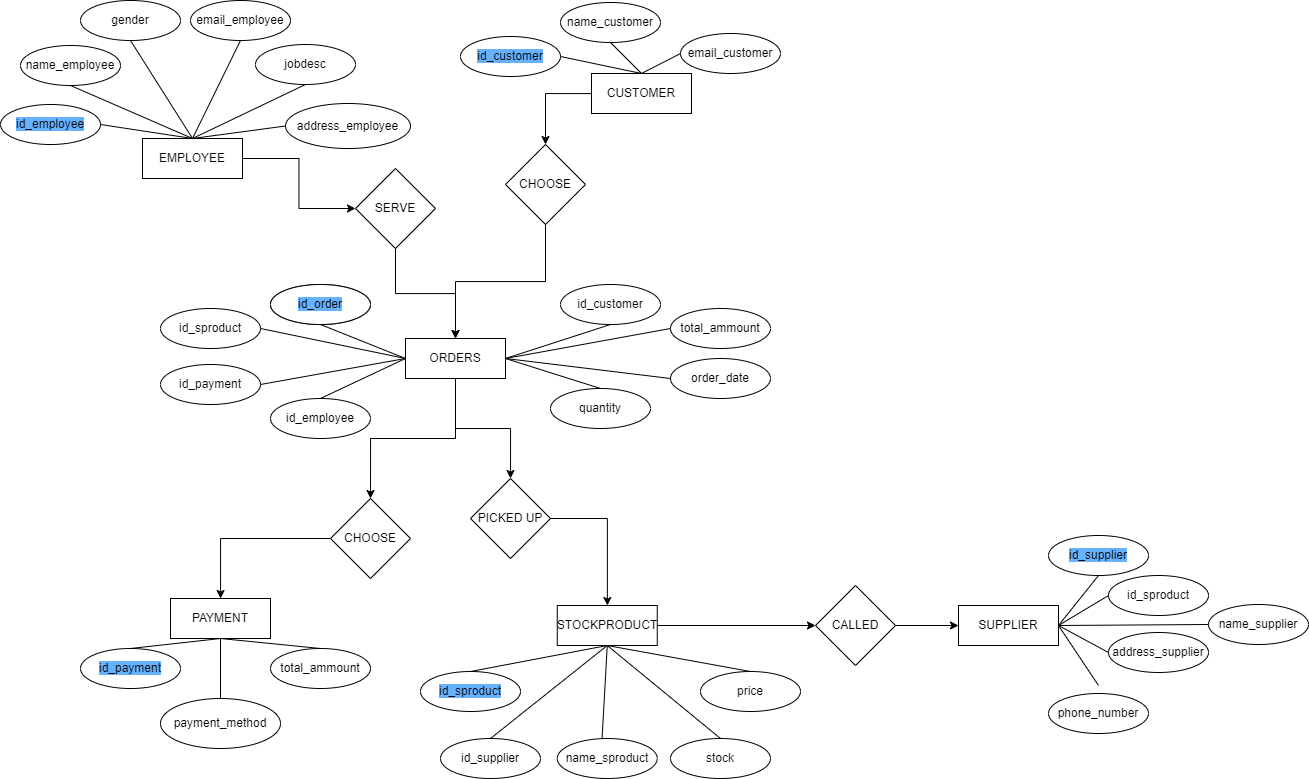
Table Order

Table Payment

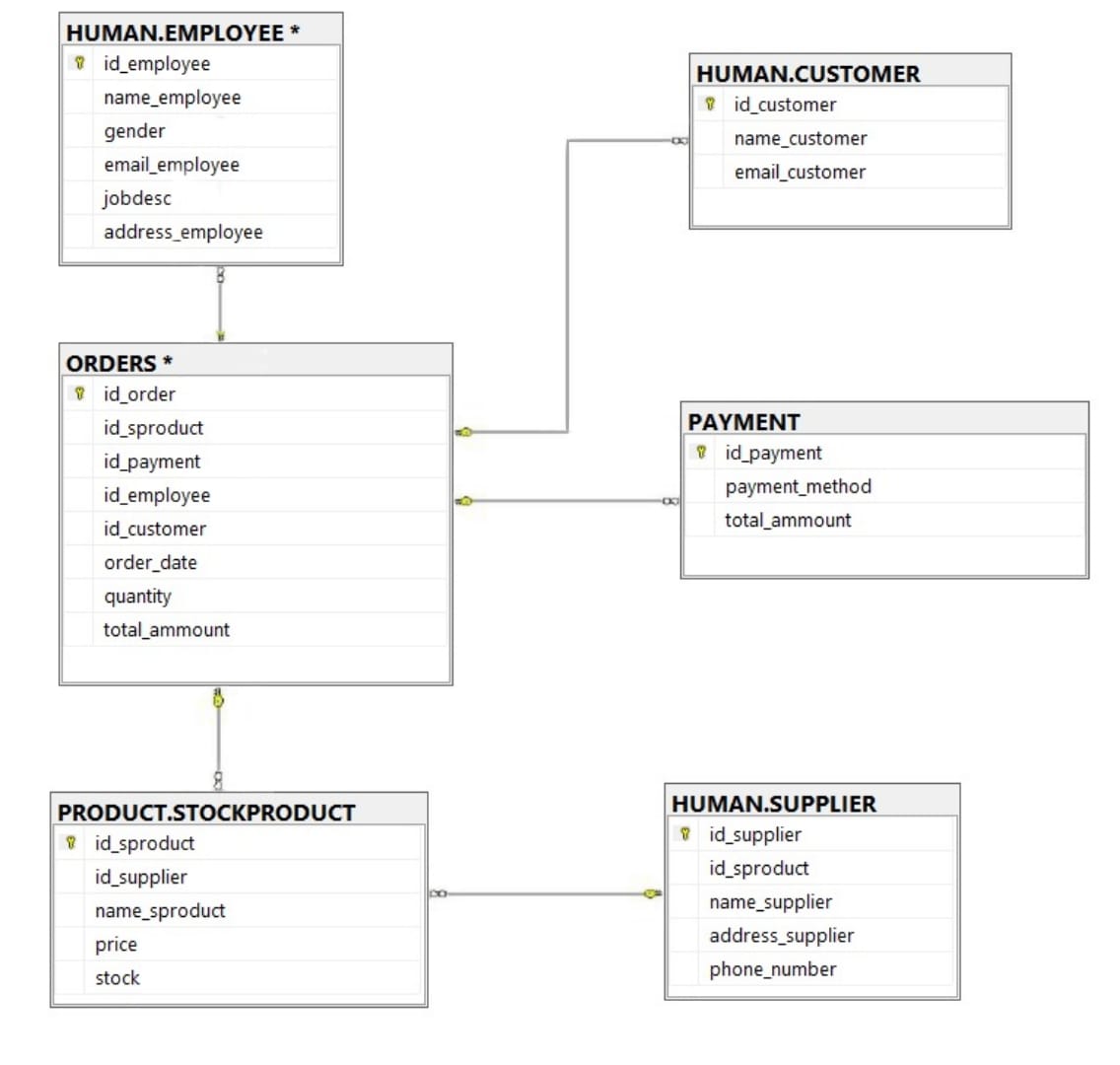
Table Stockproduct

Table Supplier

**DATABASE DESIGN**



**ENTITY RELATIONSHIP DIAGRAM**



**SCEMATIC DIAGRAM**

**TABLE DESIGN**

**Employee Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Width** | **Description** |
| id\_employee | VARCHAR | 15 | Identify of the Employee |
| name\_employee | VARCHAR | 15 | Name of the Employee |
| gender | VARCHAR | 50 | Gender of the Employee |
| email\_employee | VARCHAR | 50 | Email of the Employee |
| jobdesc | VARCHAR | 50 | Jobdesc of the Employee |
| address\_employee | VARCHAR | 60 | Address of the Employee |

**Customer Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Width** | **Description** |
| id\_customer | VARCHAR | 15 | Identify of the Customer |
| name\_customer | VARCHAR | 50 | Name of the Customer |
| email\_customer | VARCHAR | 50 | Email of the Customer |

**Payment Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Name** | **Width** | **Description** |
| id\_payment | VARCHAR | 15 | Identify of the Payment |
| payment\_method | VARCHAR | 15 | Method of the Payment |
| total\_ammount | VARCHAR | 255 | Total Ammount of the Payment |

**TABLE DESIGN**

**Stock Product Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Name** | **Width** | **Description** |
| id\_sproduct | VARCHAR | 15 | Identify of the Stock Product |
| id\_supplier | VARCHAR | 50 | Identify of the supplier |
| name\_sproduct | VARCHAR | 300 | Name of the Stock Product |
| price | VARCHAR | 15 | Price of the Stok Product |

**Order Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Name** | **Width** | **Description** |
| id\_order | VARCHAR | 15 | Identify of the Order |
| id\_sproduct | VARCHAR | 15 | Identify of the Stock Product |
| id\_payment | VARCHAR | 15 | Identify of the Payment |
| id\_employee | VARCHAR | 15 | Identify of the Employee |
| id\_customer | VARCHAR | 15 | Identify of the Customer |
| total\_ammount | VARCHAR | 255 | Total Ammount of the Order |

**Supplier Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Name** | **Width** | **Description** |
| id\_supplier | VARCHAR | 15 | Identify of the supplier |
| id\_sproduct | VARCHAR | 15 | Identify of the Stock Product |
| name\_supplier | VARCHAR | 50 | Name of the Supplier |
| address\_supplier | VARCHAR | 50 | Address of the Supplier |

**EMPLOYEE TABLE**

|  |  |
| --- | --- |
| **Validation Required** | **Method Use Validation** |
| id\_employee | Primary Key |
| name\_employee | NOT NULL |
| gender | NOT NULL |
| email\_employee | NOT NULL |
| jobdesc | NOT NULL |
| address\_employee | NOT NULL |

**CUSTOMER Table**

|  |  |
| --- | --- |
| **Validation Required** | **Method Use Validation** |
| id\_customer | Primary Key |
| name\_customer | NOT NULL |
| email\_customer | NOT NULL |

**PAYMENT Table**

|  |  |
| --- | --- |
| **Validation Required** | **Method Use Validation** |
| id\_payment | Primary Key |
| payment\_method | NOT NULL |
| total\_ammount | NOT NULL |

**VALIDATIONS PERFORMED**

**VALIDATIONS PERFORMED**

**STOCKPRODUCT Table**

|  |  |
| --- | --- |
| **Validation Required** | **Method Use Validation** |
| id\_sproduct | Primary Key |
| id\_supplier | NOT NULL |
| name\_sproduct | NOT NULL |
| price | NOT NULL |

**ORDER Table**

|  |  |
| --- | --- |
| **Validation Required** | **Method Use Validation** |
| id\_order | Primary Key |
| id\_sproduct | NOT NULL |
| id\_payment | NOT NULL |
| id\_employee | NOT NULL |
| id\_customer | NOT NULL |
| total\_ammount | NOT NULL |

**SUPPLIER Table**

|  |  |
| --- | --- |
| **Validation Required** | **Method Use Validation** |
| id\_supplier | Primary Key |
| id\_sproduct | NOT NULL |
| name\_supplier | NOT NULL |
| address\_supplier | NOT NULL |
| phone\_number | NOT NULL |

**SCRIPT**

create database DUMDUM

USE DUMDUM

-- CREATE SCHEMA

CREATE SCHEMA HUMAN

CREATE SCHEMA PRODUCT

--CREATE TABLE EMPLOYEE ON SCHEMA HUMAN

create table HUMAN.EMPLOYEE(

id\_employee VARCHAR (15) PRIMARY KEY NOT NULL, --PK

CONSTRAINT pkid\_employee

CHECK (id\_employee LIKE ('[E][0-9][0-9][0-9]')), --VALIDASI

name\_employee VARCHAR (15) NOT NULL,

gender VARCHAR (50) NOT NULL,

email\_employee VARCHAR (50) NOT NULL,

CHECK (email\_employee LIKE '\_\_%@gmail.com'), --VALIDASI EMAIL

jobdesc VARCHAR (50) NOT NULL,

address\_employee VARCHAR (60) NOT NULL,

);

INSERT INTO HUMAN.EMPLOYEE(id\_employee, name\_employee, gender, email\_employee, jobdesc, address\_employee)

VALUES

('E001', 'Jean', 'Male', 'je4n@gmail.com', 'Barista', 'Puri Street'),

('E002', 'Natasya', 'Female', 'natt@gmail.com', 'Cashier', 'Gading Street'),

('E003', 'Kevin', 'Male', 'vinn@gmail.com', 'Office Boy', 'Kemang Street'),

('E004', 'Aldo', 'Male', '44ld000@gmail.com', 'Barista', 'Serpong Street'),

('E005', 'Dapa', 'Male', 'dafxz@gmail.com', 'Cashier', 'Margonda Street');

-- CREATE TABLE CUSTOMER ON SCHEMA HUMAN --

create TABLE HUMAN.CUSTOMER(

id\_customer VARCHAR (15) PRIMARY KEY NOT NULL, --PK

CONSTRAINT pkid\_customer

CHECK (id\_customer LIKE ('[C][0-9][0-9][0-9]')), --VALIDASI

name\_customer VARCHAR (50) NOT NULL,

email\_customer VARCHAR (50) NOT NULL,

CHECK (email\_customer LIKE '\_\_%@gmail.com'), --VALIDASI GMAIL

);

INSERT INTO HUMAN.CUSTOMER (id\_customer, name\_customer, email\_customer)

VALUES

('C001', 'Stark','tonyystark@gmail.com'),

('C002', 'Steve','rogers5@gmail.com');

--CREATE TABLE PAYMENT--

create TABLE PAYMENT(

id\_payment VARCHAR (15) PRIMARY KEY NOT NULL, --PK,

CONSTRAINT pkid\_payment

**SCRIPT**

CHECK (id\_payment LIKE ('[P][0-9][0-9][0-9]')), --VALIDASI

payment\_method VARCHAR (15),

total\_ammount VARCHAR (255)

);

INSERT INTO PAYMENT (id\_payment, payment\_method, total\_ammount)

VALUES

('P001', 'Debit', 'Rp168.000');

('P002', 'Cash', 'Rp96.000');

--CREATE TABLE STOCKPRODUCT ON SCHEMA PRODUCT--

create table PRODUCT.STOCKPRODUCT(

id\_sproduct VARCHAR (15) primary key not null, --PK

CONSTRAINT pkid\_sproduct

CHECK (id\_sproduct LIKE ('[SPR][0-9][0-9][0-9]')), --VALIDASI

id\_supplier varchar (50) NOT NULL,

name\_sproduct varchar (300) NOT NULL,

price VARCHAR (15) NOT NULL,

stock int

);

INSERT INTO PRODUCT.STOCKPRODUCT (id\_sproduct, id\_supplier, name\_sproduct, price, stock)

VALUES

('SPR001', 'S001', 'Thai Tea Original', 'Rp24.000', 50),

('SPR002', 'S002', 'Thai Green Tea', 'Rp28.000', 50),

('SPR003', 'S003', 'Roasted Thai Tea', 'Rp32.000', 35),

('SPR004', 'S004', 'Dumilo Dinosaur', 'Rp25.000', 45),

('SPR005', 'S005', 'Aren Thai Coffee', 'Rp35.000', 32),

('SPR006', 'S006', 'Dark Chocolate', 'Rp34.000', 30),

);

--CREATE TABLE ORDERS--

create TABLE ORDERS (

id\_order VARCHAR (15) primary key not null, --PK,

id\_sproduct VARCHAR (15) FOREIGN KEY REFERENCES PRODUCT.STOCKPRODUCT(id\_sproduct),

id\_payment VARCHAR (15) FOREIGN KEY REFERENCES payment(id\_payment),

id\_employee VARCHAR (15) FOREIGN KEY REFERENCES HUMAN.EMPLOYEE(id\_employee),

id\_customer VARCHAR (15) FOREIGN KEY REFERENCES HUMAN.CUSTOMER(id\_customer),

order\_date DATE,

quantity int,

total\_ammount VARCHAR (255)

);

--INPUT DATA ORDERS--

INSERT INTO ORDERS (id\_order, id\_sproduct, id\_payment, id\_employee, id\_customer, order\_date, quantity, total\_ammount)

VALUES

(1, 'SPR001', 'P001', 'E002', 'C001', '2023-12-19', 7, (SELECT price FROM PRODUCT.STOCKPRODUCT WHERE id\_sproduct = 'SPR001') \* 7);

(2, 'SPR003', 'P002', 'E005', 'C002', '2023-12-19', 4, (SELECT price FROM PRODUCT.STOCKPRODUCT WHERE id\_sproduct = 'SPR003') \* 3);

**SCRIPT**

(1, 'SPR001', 'P001', 'E002', 'C001', '2023-12-19', 7, (SELECT price FROM PRODUCT.STOCKPRODUCT WHERE id\_sproduct = 'SPR001') \* 7);

(2, 'SPR003', 'P002', 'E005', 'C002', '2023-12-19', 4, (SELECT price FROM PRODUCT.STOCKPRODUCT WHERE id\_sproduct = 'SPR003') \* 3);

--CREATE TABLE SUPPLIER ON SCHEMA HUMAN

create table HUMAN.SUPPLIER(

id\_supplier VARCHAR (15) PRIMARY KEY NOT NULL, --PK

CONSTRAINT pkid\_supplier

CHECK (id\_supplier LIKE ('[S][0-9][0-9][0-9]')), --VALIDASI

id\_sproduct VARCHAR (15) NOT NULL,

name\_supplier VARCHAR (50) NOT NULL,

address\_supplier VARCHAR (50) NOT NULL,

phone\_number VARCHAR (50) NOT NULL,

CHECK (phone\_number LIKE '%08\_\_'), --VALIDASI PHONE NUMBER

);

INSERT INTO HUMAN.SUPPLIER(id\_supplier,id\_sproduct, name\_supplier,address\_supplier,phone\_number)

VALUES

('S001', 'SPR006', 'Jimin', 'Senayan Street', '08197298291'),

('S002', 'SPR005','Jungkook', 'Sudirman Street', '086193026678'),

('S003', 'SPR003', 'Taehyung', 'Semanggi Street', '088390552813'),

('S003', 'SPR002', 'Jin', 'Cipete Street', '080639271025'),

('S003', 'SPR004', 'Suga', 'Lebak Bulus Street', '089188224813'),

('S003', 'SPR001', 'Namseok', 'Tebet Street', '081824627702');

--CREATE STORED PROCEDURE--

--PROCEDURE TO SEE ALL TABLES--

CREATE PROCEDURE SPALL

AS

BEGIN

SELECT \* FROM HUMAN.EMPLOYEE

SELECT \* FROM HUMAN.CUSTOMER

SELECT \* FROM ORDER

SELECT \* FROM PAYMENT

SELECT \* FROM PRODUCT.STOCKPRODUCT

SELECT \* FROM HUMAN.SUPPLIER

END

EXEC SPALL

--CREATE VIEW FOR INFORMATION MENU--

CREATE VIEW Menu AS

SELECT

o.id\_order,

p.name\_product AS product\_name,

p.price

FROM

ORDERS o

JOIN PRODUCT.STOCKPRODUCT p ON o.id\_sproduct = p.id\_sproduct;

--SHOW DATA FROM TABLE THAT HAS BEEN CREATED--

SELECT \* FROM Menu;

--PROCEDURE TO INPUT NEW DATA ON TABLE CUSTOMER--

CREATE PROCEDURE InsertCustomer

@id\_customer VARCHAR (15),

@name\_customer VARCHAR (50),

@email\_customer VARCHAR (50),

AS

BEGIN

INSERT INTO HUMAN.CUSTOMER (id\_customer, name\_customer, email\_customer)

VALUES (@id\_customer, @name\_customer, @email\_customer);

END;

EXEC InsertCustomer 'C003', 'Irish', '1rish@gmail.com';

--PROCEDURE UPDATE DATA ON TABLE CUSTOMER--

CREATE PROCEDURE sp\_UpdateCustomer

@id\_customer VARCHAR (15),

@new\_name\_customer VARCHAR (50),

@new\_email\_customer VARCHAR (50),

**SCRIPT**

JOIN PRODUCT.STOCKPRODUCT p ON o.id\_sproduct = p.id\_sproduct;

--SHOW DATA FROM TABLE THAT HAS BEEN CREATED--

SELECT \* FROM Menu;

--PROCEDURE TO INPUT NEW DATA ON TABLE CUSTOMER--

CREATE PROCEDURE InsertCustomer

@id\_customer VARCHAR (15),

@name\_customer VARCHAR (50),

@email\_customer VARCHAR (50),

AS

BEGIN

INSERT INTO HUMAN.CUSTOMER (id\_customer, name\_customer, email\_customer)

VALUES (@id\_customer, @name\_customer, @email\_customer);

END;

EXEC InsertCustomer 'C003', 'Irish', '1rish@gmail.com';

--PROCEDURE UPDATE DATA ON TABLE CUSTOMER--

CREATE PROCEDURE sp\_UpdateCustomer

@id\_customer VARCHAR (15),

@new\_name\_customer VARCHAR (50),

@new\_email\_customer VARCHAR (50),

AS

BEGIN

UPDATE HUMAN.CUSTOMER

SET

name\_customer = @new\_name\_customer,

email\_customer = @new\_email\_customer

WHERE

id\_customer = @id\_customer;

END;

EXEC sp\_UpdateCustomer 'C002', 'Steve', 'St3ve254@gmail.com';

--PROCEDURE DELETE DATA BASE ON id\_customer IN TABLE CUSTOMER--

CREATE PROCEDURE DeleteCustomerByID

@ID\_Reservasi varchar(15)

AS

BEGIN

DELETE FROM HUMAN.CUSTOMER

WHERE id\_customer = @id\_customer;

END;

EXEC DeleteCustomerByID 'C002'

--PROCEDURE DELETE EMPLOYEE

create procedure DELETEEMPLOYEE

@id\_employee INT

AS

**SCRIPT**

DELETE FROM HUMAN.EMPLOYEE WHERE @id\_employee = id\_employee

-- PROCEDURE UPDATE STOCK

create procedure UPDATESTOCKPRODUCT

@id\_sproduct VARCHAR (50)

@id\_supplier VARCHAR (50)

@name\_product VARCHAR (50)

@price VARCHAR (50)

@stock INT (50)

@change\_timestamp TIMESTAMP

AS

BEGIN

INSERT INTO stock\_change\_log (id\_sproduct, change\_quantity, new\_stock, change\_timestamp)

VALUES (p\_id\_sproduct, p\_change\_quantity, p\_new\_stock, p\_change\_timestamp);

END

--CREATE TRIGGER NOTIFICATION FOR EVERY INPUT DATA TABLE STOCKPRODUCT--

CREATE TRIGGER TRINSERTPRODUCT

ON PRODUCT.STOCKPRODUCT

FOR INSERT

AS

BEGIN

DECLARE @id\_sproduct VARCHAR (50),

@id\_supplier VARCHAR (50),

@name\_product VARCHAR (50),

@price VARCHAR (50),

@stock INT;

SELECT @id\_sproduct = id\_sproduct, @id\_supplier = id\_supplier, @name\_product = name\_product, @price = price, @stock = stock

FROM INSERTED;

PRINT '===================NOTIFICATION======================'

PRINT 'id sproduct : ' + @id\_sproduct

PRINT 'id supplier : ' + @id\_supplier

PRINT 'name\_product : ' + @name\_product

PRINT 'price : ' + @price

PRINT 'stock : ' + CONVERT(INT, @stock)

PRINT ''

PRINT 'THE DATA HAS BEEN INSERTED '

PRINT '===================NOTIFICATION======================'

END;

--AFTER INSERT

-- CREATE TRIGGER AFTER INSERT FOR TABLE STOCKPRODUCT

CREATE TRIGGER name\_product\_AfterInsert

ON PRODUCT.STOCKPRODUCT

AFTER INSERT

AS

BEGIN

UPDATE PRODUCT.STOCKPRODUCT

SET name = + inserted.name

FROM PRODUCT.STOCKPRODUCT

INNER JOIN inserted ON PRODUCT.STOCKPRODUCT.stock = inserted.stock;

END;

INSERT INTO PRODUCT.STOCKPRODUCT (id\_sproduct, id\_supplier, name\_product, price, stock)

VALUES

('SPR007', 'S007', 'Bubble Tea', 'Rp31.000', 24);

**SCRIPT**

UPDATE PRODUCT.STOCKPRODUCT

SET name = + inserted.name

FROM PRODUCT.STOCKPRODUCT

INNER JOIN inserted ON PRODUCT.STOCKPRODUCT.stock = inserted.stock;

END;

INSERT INTO PRODUCT.STOCKPRODUCT (id\_sproduct, id\_supplier, name\_product, price, stock)

VALUES

('SPR007', 'S007', 'Bubble Tea', 'Rp31.000', 24);

--AFTER UPDATE

-- CREATE TABLE LOG FOR SAVING CHANGES IN TABLE STOCKPRODUCT --

CREATE TABLE PRODUCT.STOCKPRODUCT\_Log (

LogID INT PRIMARY KEY IDENTITY(1,1),

id\_sproduct varchar(15),

id\_supplier varchar(15),

name\_product varchar(15),

price varchar (30),

stock int,

UpdateTimestamp datetime DEFAULT GETDATE()

);

--CREATE TRIGGER AFTER UPDATE FOR TABLE STOCK PRODUCT

CREATE TRIGGER UPDATEPRODUCT\_Log

ON PRODUCT.STOCKPRODUCT

AFTER UPDATE

AS

BEGIN

INSERT INTO PRODUCT.STOCKPRODUCT\_Log (id\_sproduct, id\_supplier, name\_product, price, stock)

SELECT id\_sproduct, id\_supplier, name\_product, price, stock

FROM inserted;

END;

UPDATE PRODUCT.STOCKPRODUCT

SET name\_product = 'Bubble Tea'

WHERE id\_sproduct = 'SPR007'

SELECT \* FROM PRODUCT\_STOCKPRODUCT\_log;

--AFTERDELETE

-- CREATE TABLE LOG FOR SAVING CHANGES IN TABLE STOCKPRODUCT

CREATE TABLE PRODUCT.DELETEPRODUCT\_Log (

id\_Log INT PRIMARY KEY IDENTITY(1,1),

id\_sproduct varchar(15),

id\_supplier varchar(15),

name\_product varchar(15),

price varchar (30),

stock int,

Deleted\_Date DATETIME

);

-- CREATE TRIGGER AFTER DELETE FOT TABLE STOCKPRODUCT

CREATE TRIGGER DeleteSTOCKPRODUCT\_Log1

ON PRODUCT.STOCKPRODUCT

AFTER DELETE

AS

BEGIN

INSERT INTO PRODUCT.DeleteSTOCKPRODUCT\_Log (id\_sproduct, id\_supplier, name\_product, price, stock, Deleted\_Date)

SELECT id\_sproduct, id\_supplier, name\_product, price, stock, GETDATE()

FROM deleted;

END;

DELETE FROM PRODUCT.STOCKPRODUCT

WHERE id\_sproduct = 'SPR007'

SELECT \* FROM PRODUCT.DeleteSTOCKPRODUCT\_Log;

**SCRIPT**

-- CREATE TRIGGER AFTER DELETE FOT TABLE STOCKPRODUCT

CREATE TRIGGER DeleteSTOCKPRODUCT\_Log1

ON PRODUCT.STOCKPRODUCT

AFTER DELETE

AS

BEGIN

INSERT INTO PRODUCT.DeleteSTOCKPRODUCT\_Log (id\_sproduct, id\_supplier, name\_product, price, stock, Deleted\_Date)

SELECT id\_sproduct, id\_supplier, name\_product, price, stock, GETDATE()

FROM deleted;

END;

DELETE FROM PRODUCT.STOCKPRODUCT

WHERE id\_sproduct = 'SPR007'

SELECT \* FROM PRODUCT.DeleteSTOCKPRODUCT\_Log;

**Hardware :** Acer Aspire A715 – 76G 12th Gen Intel(R) Core(TM) i5-12450H

2.00 GHz 8,00 GB 64-bit operating system

**Operating System :** Windows 11 Home Single Language

**Software :**  Microsoft SQL Server Management Studio, Draw.io, Google Chrome, Microsoft Word

|  |  |  |
| --- | --- | --- |
| **PROJECT FILE DETAILS** | | |
| **No.** | **File Name** | **Remarks** |
| 1. | ERD.jpeg | Made with draw.io |
| 2. | DUMDUM.sql | Made with Microsoft SSMS |
| 3. | Schematic Diagram.jpeg | Made with Microsoft SSMS |

**CONFIGURATION**