

**TUGAS INDIVIDU**  
**CRUD DATABASE**  
**PEMROGRAMAN BERORIENTASI OBJEK**



**DISUSUN OLEH:**  
**AJI KHARISMA ATMAJA (5230411292)**

**JURUSAN INFORMATIKA**  
**FAKULTAS SAINS DAN TEKNOLOGI**  
**UNIVERSITAS TEKNOLOGI YOGYAKARTA**  
**2024**

## 1. IMPORT YANG DIBUTUHKAN

```
import mysql.connector
from mysql.connector import Error
from datetime import datetime
from tabulate import tabulate
```

Saya mengimport mysql.connector agar terhubung dengan mysql. Kemudian import error untuk memberi tahu error yang berhubungan langsung dengan mysql. Import datetime untuk membuat tanggal otomatis. Yang terakhir import tabulate untuk membuat table agar lebih rapi.

## 2. MEMBUAT CLASS PENJUALAN

```
class penjualan:
    def __init__(self, host="localhost", user="root", password="",
database="penjualan"):
        self.conn = mysql.connector.connect(
            host=host,
            user=user,
            password=password,
            database=database
        )
        self.cur = self.conn.cursor()
```

Dimana dalam class ini ada fungsi konstruktor yang mendeskripsikan tentang database mysql seperti host, user, password dan nama database kita. Yang kemudian menggunakan self.cur untuk mengkoneksikan ke database mysql.

## 3. MEMBUAT DATABASE

```
def create_database(self):
    try:
        self.cur.execute("CREATE DATABASE penjualan")
        print("Database 'penjualan' berhasil dibuat.")
    except mysql.connector.Error as e:
        print(f"Terjadi kesalahan saat membuat database: {e}")
```

Fungsi ini untuk membuat database yang bernama penjualan dan ada infonya jika berhasil serta error jika ada yang error. Hasilnya :



## 4. MEMBUAT TABEL-TABEL DALAM DATABASE

```
def create_pegawai_table(self):
    try:
        self.cur.execute("""
            CREATE TABLE Pegawai (
                NIK CHAR(50) NOT NULL PRIMARY KEY,
                Nama_Pegawai VARCHAR(15),
                Alamat VARCHAR(255)
            )
        """)
        print("Tabel 'Pegawai' berhasil dibuat.")
    except mysql.connector.Error as e:
        print(f"Terjadi kesalahan saat membuat tabel Pegawai: {e}")
```

```

# Fungsi untuk membuat tabel Transaksi
def create_transaksi_table(self):
    try:
        self.cur.execute("""
            CREATE TABLE Transaksi (
                No_Transaksi CHAR(4) NOT NULL PRIMARY KEY,
                Detail_Transaksi VARCHAR(255)
            )
        """)
        print("Tabel 'Transaksi' berhasil dibuat.")
    except mysql.connector.Error as e:
        print(f"Terjadi kesalahan saat membuat tabel Transaksi: {e}")

# Fungsi untuk membuat tabel Produk
def create_produk_table(self):
    try:
        self.cur.execute("""
            CREATE TABLE Produk (
                Kode_Produk CHAR(4) NOT NULL PRIMARY KEY,
                Nama_Produk VARCHAR(25),
                Jenis_Produk VARCHAR(20),
                Harga INT(30)
            )
        """)
        print("Tabel 'Produk' berhasil dibuat.")
    except mysql.connector.Error as e:
        print(f"Terjadi kesalahan saat membuat tabel Produk: {e}")

# Fungsi untuk membuat tabel Struk
def create_struk_table(self):
    try:
        self.cur.execute("""
            CREATE TABLE Struk (
                No_Transaksi CHAR(4) NOT NULL PRIMARY KEY,
                Nama_Pegawai VARCHAR(15),
                Nama_Produk VARCHAR(25),
                Jumlah_Produk INT(30),
                Total_Harga INT(30)
            )
        """)
        print("Tabel 'Struk' berhasil dibuat.")
    except mysql.connector.Error as e:
        print(f"Terjadi kesalahan saat membuat tabel Struk: {e}")

```

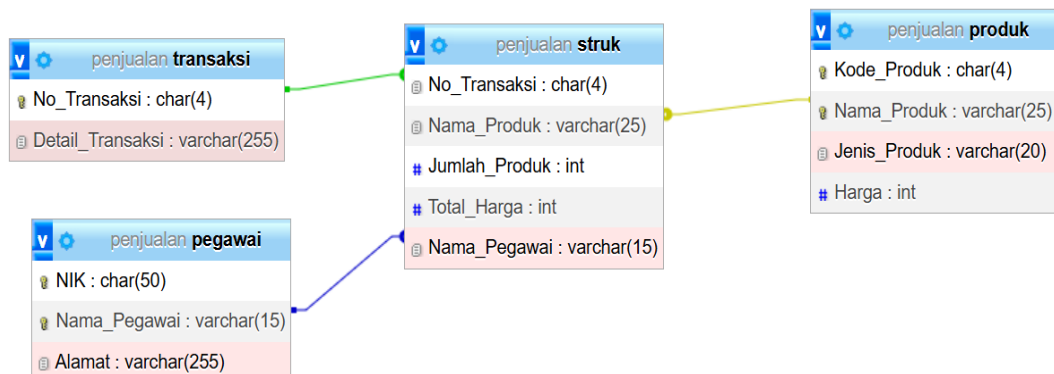
Fungsi-fungsi diatas berfungsi untuk membuat table-table pada database kita. Dimana didalamnya juga ada nama-nama kolom dari setiap table dan tipe-tipe data serta Panjang data yang ditentukan pada setiap kolom. Jika berhasil dibuat ada info berhasil dibuat jika error ada pesan error nya. Hasilnya :

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> pegawai	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_0900_ai_ci	16.0 KiB	-
<input type="checkbox"/> produk	★ Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_0900_ai_ci	16.0 KiB	-
<input type="checkbox"/> struk	★ Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_0900_ai_ci	64.0 KiB	-
<input type="checkbox"/> transaksi	★ Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_0900_ai_ci	16.0 KiB	-
4 tables	Sum	13	InnoDB	utf8mb4_0900_ai_ci	112.0 KiB	0 B

## 5. MEMBUAT RELASI ANTAR TABEL

```
def add_foreign_keys(self):
    try:
        self.cur.execute("""
            ALTER TABLE Struk
            ADD FOREIGN KEY (Nama_Pegawai)
            REFERENCES Pegawai(Nama_Pegawai)
        """)
        self.cur.execute("""
            ALTER TABLE Struk
            ADD FOREIGN KEY (Nama_Produk)
            REFERENCES Produk(Nama_Produk)
        """)
        self.cur.execute("""
            ALTER TABLE Struk
            ADD FOREIGN KEY (No_Transaksi)
            REFERENCES Transaksi(No_Transaksi)
        """)
        print("Foreign keys berhasil ditambahkan.")
    except mysql.connector.Error as e:
        print(f"Terjadi kesalahan saat menambahkan foreign keys: {e}")
```

Fungsi diatas berfungsi untuk membuat relasi antar table dalam database. Yang pertama ada table pegawai berelasi dengan table struk dengan foreign keynya adalah nama\_pegawai, Kedua ada table produk yang berelasi dengan table struk dimana foreign key nya adalah nama\_produk. Terakhir ada table transaksi yang berelasi dengan table struk dimana foreign key nya adalah no\_transaksi. Hasilnya:



## 6. FUNGSI TAMBAH PEGAWAI

```
def tambah_pegawai(self, nik, nama_pegawai, alamat):
    try:
        if not nik or not nama_pegawai or not alamat:
```

```

        print("Semua inputan harus diisi. Gagal menambahkan
Pegawai.")
        return

        self.cur.execute("SELECT COUNT(*) FROM Pegawai WHERE NIK =
%s", (nik,))
        result = self.cur.fetchone()

        if result[0] > 0:
            print(f"NIK {nik} sudah terdaftar. Gagal menambahkan
Pegawai.")
        else:
            query = "INSERT INTO Pegawai (NIK, Nama_Pegawai,
Alamat) VALUES (%s, %s, %s)"
            self.cur.execute(query, (nik, nama_pegawai, alamat))
            self.conn.commit()
            print("Data Pegawai berhasil ditambahkan.")
        except Error as e:
            print(f"Terjadi kesalahan saat menambahkan pegawai: {e}")

```

Kode itu berfungsi untuk menambahkan data pegawai kedalam table pegawai. Yang ditambahkan adalah nik, nama pegawai, dan alamat. Semua inputan harus diisi jika tidak akan ada pesan seperti ini (Semua inputan harus diisi. Gagal menambahkan Pegawai). Jika ada nik yang double maka itu tidak bisa dan nik setiap pegawai harus berbeda. Jika semua kondisi terpenuhi maka data pegawai dapat ditambahkan ke dalam table pegawai.

## 7. FUNGSI LIHAT PEGAWAI

```

def lihat_pegawai(self):
    try:
        self.cur.execute("SELECT * FROM Pegawai")
        return self.cur.fetchall()
    except Error as e:
        print(f"Terjadi kesalahan saat membaca data pegawai: {e}")

```

Kode itu berfungsi untuk melihat daftar pegawai yang ada dalam table pegawai di database kita.

## 8. FUNGSI UPDATE PEGAWAI

```

def update_pegawai(self, nik, nama_pegawai=None, alamat=None):
    try:
        query = "UPDATE Pegawai SET"
        updates = []
        values = []

        if nama_pegawai:
            updates.append("Nama_Pegawai = %s")
            values.append(nama_pegawai)

        if alamat:
            updates.append("Alamat = %s")
            values.append(alamat)

```

```

        if updates:
            query += " " + ", ".join(updates) + " WHERE NIK = %s"
            values.append(nik)
            self.cur.execute(query, tuple(values))
            self.conn.commit()
            print("Data Pegawai berhasil diperbarui.")
        else:
            print("Tidak ada data yang diperbarui.")
    except Error as e:
        print(f"Terjadi kesalahan saat memperbarui pegawai: {e}")

```

Kode itu berfungsi untuk mengupdate data pegawai dalam table pegawai. Updates adalah list untuk menampung data mana yang akan diperbarui dan values adalah list untuk menampung data baru. Yang dapat diupdate hanya nama dan alamat, bisa keduanya atau salah satu saja. Data akan diperbarui sesuai nik yang akan dipilih oleh user.

## 9. FUNGSI HAPUS PEGAWAI

```

def hapus_pegawai(self, nik):
    try:
        self.cur.execute("DELETE FROM Pegawai WHERE NIK = %s",
(nik,))
        self.conn.commit()
        print("Data Pegawai berhasil dihapus.")
    except Error as e:
        print(f"Terjadi kesalahan saat menghapus pegawai: {e}")

```

Kode itu berfungsi untuk menghapus data dalam table pegawai. Tapi jika data nya sudah berelasi dengan table lain itu akan error dan ada pesan kesalahan saat menghapus jadi yang bisa dihapus hanya data yang belum berelasi dengan table lain.

## 10. FUNGSI TAMBAH PRODUK

```

def tambah_produk(self, nama_produk, jenis_produk, harga):
    try:
        self.cur.execute("SELECT COUNT(*) FROM Produk")
        result = self.cur.fetchone()
        product_count = result[0] + 1
        kode_produk = f"P{product_count:02d}"

        query = "INSERT INTO Produk (Kode_Produk, Nama_Produk,
Jenis_Produk, Harga) VALUES (%s, %s, %s, %s)"
        self.cur.execute(query, (kode_produk, nama_produk,
jenis_produk, harga))
        self.conn.commit()
        print(f"Data Produk dengan Kode {kode_produk} berhasil
ditambahkan.")
    except Error as e:
        print(f"Terjadi kesalahan saat menambahkan produk: {e}")

```

Kode itu untuk menambahkan data produk ke dalam table produk. Dimana yang ditambahkan adalah kode produk tapi kode produk ini tidak diinputkan oleh user melainkan sudah otomatis dibuat, nama produk, jenis produk dan harga.

## 11. FUNGSI LIHAT PRODUK

```
def lihat_produk(self):
    try:
        self.cur.execute("SELECT * FROM Produk")
        return self.cur.fetchall()
    except Error as e:
        print(f"Terjadi kesalahan saat membaca data produk: {e}")
```

Kode itu berfungsi melihat data dalam table produk dan akan menampilkan semua data yang ada.

## 12. FUNGSI UPDATE PRODUK

```
def update_produk(self, kode_produk, nama_produk=None,
jenis_produk=None, harga=None):
    try:
        query = "UPDATE Produk SET"
        updates = []
        values = []

        if nama_produk:
            updates.append("Nama_Produk = %s")
            values.append(nama_produk)

        if jenis_produk:
            updates.append("Jenis_Produk = %s")
            values.append(jenis_produk)

        if harga is not None:
            updates.append("Harga = %s")
            values.append(harga)

        if updates:
            query += " " + ", ".join(updates) + " WHERE Kode_Produk"
            = %s"
            values.append(kode_produk)
            self.cur.execute(query, tuple(values))
            self.conn.commit()
            print("Data Produk berhasil diperbarui.")
        else:
            print("Tidak ada data yang diperbarui.")
    except Error as e:
        print(f"Terjadi kesalahan saat memperbarui produk: {e}")
```

Dalam kode itu sebenarnya sama saja dengan yang di pegawai. Perbedaannya hanya kode itu mengupdate ditabel pegawai dan yang dapat diupdate adalah nama produk, jenis dan harganya bisa ketiganya atau salah dua atau salah satunya.

## 13. FUNGSI HAPUS PRODUK

```
def hapus_produk(self, kode_produk):
    try:
        self.cur.execute("DELETE FROM Produk WHERE Kode_Produk = %s", (kode_produk,))
```

```

        self.conn.commit()
        print("Data Produk berhasil dihapus.")
    except Error as e:
        print(f"Terjadi kesalahan saat menghapus produk: {e}")

```

Kode itu untuk menghapus data daam table produk,sama seperti pegawai tadi yang dihapus adalah data yang belum memiliki relasi saja jika sudah memiliki maka akan ada pesan error.

#### 14. FUNGSI TAMBAH TRANSAKSI

```

def tambah_transaksi(self):
    try:
        # Membuat No Transaksi otomatis (contoh: T01, T02, ...)
        self.cur.execute("SELECT COUNT(*) FROM Transaksi")
        result = self.cur.fetchone()
        transaksi_count = result[0] + 1 # Menghitung transaksi
        no_transaksi = f"T{transaksi_count:02d}" # Format No
        Transaksi

        # Menentukan detail transaksi (tanggal dan waktu saat ini)
        tanggal_transaksi = datetime.now().strftime("%Y-%m-%d
%H:%M:%S")
        detail_transaksi = tanggal_transaksi # Menggunakan tanggal
        dan waktu saat transaksi

        # Input dari pengguna
        nama_pegawai = input("Masukkan nama pegawai: ").strip()
        if not nama_pegawai:
            print("Nama pegawai tidak boleh kosong.")
            return

        # Menambahkan produk ke dalam transaksi
        produk_list = []
        while True:
            nama_produk = input("Masukkan nama produk: ").strip()
            if not nama_produk:
                print("Nama produk tidak boleh kosong.")
                return

            jumlah_produk_input = input("Masukkan jumlah produk:
").strip()
            if not jumlah_produk_input.isdigit() or
int(jumlah_produk_input) <= 0:
                print("Jumlah produk harus berupa angka positif.")
                return
            jumlah_produk = int(jumlah_produk_input)

            # Cek apakah produk tersedia dalam tabel Produk

```



```

        self.cur.execute("SELECT Harga FROM Produk WHERE
Nama_Produk = %s", (nama_produk,))
        result_produk = self.cur.fetchone()

        if result_produk:
            harga_produk = result_produk[0]
            total_harga = harga_produk * jumlah_produk
            produk_list.append((nama_produk, jumlah_produk,
total_harga))
        else:
            print(f"Produk '{nama_produk}' tidak ditemukan
dalam database.")

        # Tanya apakah ingin menambah produk lagi
        tambah_lagi = input("Apakah ingin menambah produk lagi?
(ya/tidak): ").strip().lower()
        if tambah_lagi != 'ya':
            break

        # Cek apakah pegawai tersedia dalam tabel Pegawai
        self.cur.execute("SELECT * FROM Pegawai WHERE Nama_Pegawai
= %s", (nama_pegawai,))
        result_pegawai = self.cur.fetchone()

        if result_pegawai:
            # Menambahkan data ke tabel Transaksi
            query_transaksi = "INSERT INTO Transaksi (No_Transaksi,
Detail_Transaksi) VALUES (%s, %s)"
            self.cur.execute(query_transaksi, (no_transaksi,
detail_transaksi))
            self.conn.commit()

            # Menambahkan data ke tabel Struk untuk setiap produk
yang dibeli
            for produk in produk_list:
                nama_produk, jumlah_produk, total_harga = produk
                # Pastikan bahwa setiap produk dimasukkan dengan
entri unik ke tabel Struk
                query_struk = """INSERT INTO Struk (No_Transaksi,
Nama_Pegawai, Nama_Produk, Jumlah_Produk, Total_Harga)
VALUES (%s, %s, %s, %s, %s)"""
                self.cur.execute(query_struk, (no_transaksi,
nama_pegawai, nama_produk, jumlah_produk, total_harga))
                self.conn.commit()

            print(f"Transaksi dan struk dengan No Transaksi
{no_transaksi} berhasil ditambahkan.")
        else:

```

```

        print(f"Pegawai '{nama_pegawai}' tidak ditemukan dalam
database.")
    except Error as e:
        print(f"Terjadi kesalahan saat menambahkan transaksi: {e}")

```

Kode diatas untuk menambahkan transaksi dan menambahkannya kedalam table transaksi.No transaksi akan dibuat secara otomatis tanpa inputan dari user.Pertama user akan input nama pegawai dulu dan nama pegawai tidak boleh kosong dan tidak boleh tidak ada dari dalam table.Kemudian nama produk ini juga harus yang ada dalam database dan jumlah produk yang dimana mengambil dari table produk untuk harganya kemudian di totalkan dengan cara harga dikali jumlah.User dapat menginputkan lebih dari 1 transaksi.Jika semua sudah berhasil semua inputan tadi akan masuk kedalam table transaksi.

## 15. FUNGSI CETAK STRUK

```

def cetak_struk(self):
    try:
        # Menampilkan struk yang baru saja ditambahkan (struk
        terakhir berdasarkan No Transaksi)
        self.cur.execute("""
            SELECT s.No_Transaksi, s>Nama_Pegawai, s>Nama_Produk,
            s.Jumlah_Produk, s.Total_Harga
            FROM Struk s
            WHERE s.No_Transaksi = (SELECT MAX(No_Transaksi) FROM
Struk)
        """)

        rows = self.cur.fetchall()

        if rows:
            total_harga = 0
            print("=" * 30)
            print("                STRUK TRANSAKSI")
            print("=" * 30)

            for row in rows:
                no_transaksi = row[0]
                nama_pegawai = row[1]
                nama_produk = row[2]
                jumlah_produk = row[3]
                total_harga_produk = row[4]

                total_harga += total_harga_produk # Menambahkan
                harga total per produk ke total harga keseluruhan

            # Mencetak detail produk per transaksi
            print(f>Nama Produk      : {nama_produk}")
            print(f>Jumlah Produk   : {jumlah_produk}")
            print(f>Total Per Item  : Rp
{total_harga_produk:,.0f}")

```

```

        print("-" * 30)

        # Menampilkan total harga
        print(f"Total Harga      : Rp {total_harga:,.0f}")
        print("-" * 30)
    else:
        print("Tidak ada data struk.")
except Error as e:
    print(f"Terjadi kesalahan saat membaca data struk: {e}")

```

Kode ini akan mencetak hasil tambah transaksi yang baru saja ditambahkan. Semua data yang berelasi dengan table struk ini akan ditampilkan dalam bentuk struk sederhana. Didalamnya juga ada perhitungan total harga barang yang dibeli.

## 16. FUNGSI MENUTUP KONEKSI

```

def close_connection(self):
    try:
        self.cur.close()
        self.conn.close()
    except Error as e:
        print(f"Terjadi kesalahan saat menutup koneksi: {e}")

```

## 17. MENU-MENU

```

if __name__ == "__main__":
    crud = penjualan()

    while True:
        print("\nMenu Warung Penjualan Sederhana:")
        print("1. Pegawai")
        print("2. Produk")
        print("3. Transaksi")
        print("4. Struk")
        print("5. Exit")

        pilih = input("Pilih menu: ")

        if pilih == "1":
            while True:
                print("\nMenu Pegawai:")
                print("1. Tambah Pegawai")
                print("2. Lihat Pegawai")
                print("3. Update Pegawai")
                print("4. Hapus Pegawai")
                print("5. Kembali ke Menu Utama")

                pilihan = input("Pilih menu pegawai: ")

                if pilihan == "1":
                    nik = input("Masukkan NIK: ")
                    nama = input("Masukkan Nama Pegawai: ")

```

```

        alamat = input("Masukkan Alamat: ")
        crud.tambah_pegawai(nik, nama, alamat)
    elif pilihan == "2":
        pegawai = crud.lihat_pegawai()
        print(tabulate(pegawai, headers=["NIK",
>Nama_Pegawai", "Alamat"], tablefmt="grid"))
    elif pilihan == "3":
        nik = input("Masukkan NIK yang akan diupdate: ")
        nama = input("Masukkan Nama baru (kosongkan jika
tidak ingin diubah): ")
        alamat = input("Masukkan Alamat baru (kosongkan
jika tidak ingin diubah): ")
        crud.update_pegawai(nik, nama if nama else None,
alamat if alamat else None)
    elif pilihan == "4":
        nik = input("Masukkan NIK yang akan dihapus: ")
        crud.hapus_pegawai(nik)
    elif pilihan == "5":
        break
    else:
        print("Pilihan tidak valid(Pilih Menu Yang
Sesuai).")

elif pilih == "2":
    while True:
        print("\nMenu Produk:")
        print("1. Tambah Produk")
        print("2. Lihat Produk")
        print("3. Update Produk")
        print("4. Hapus Produk")
        print("5. Kembali ke Menu Utama")

        pilihan = input("Pilih menu produk: ")

        if pilihan == "1":
            nama = input("Masukkan Nama Produk: ")
            jenis = input("Masukkan Jenis Produk: ")
            harga = float(input("Masukkan Harga Produk: "))
            crud.tambah_produk(nama, jenis, harga)
        elif pilihan == "2":
            produk = crud.lihat_produk()
            print(tabulate(produk, headers=["Kode_Produk",
>Nama_Produk", "Jenis_Produk", "Harga"], tablefmt="grid"))
        elif pilihan == "3":
            kode_produk = input("Masukkan Kode Produk yang akan
diupdate: ")
            nama = input("Masukkan Nama baru (kosongkan jika
tidak ingin diubah): ")

```

```

        jenis = input("Masukkan Jenis baru (kosongkan jika
tidak ingin diubah): ")
        harga = input("Masukkan Harga baru (kosongkan jika
tidak ingin diubah): ")
        crud.update_produk(kode_produk, nama if nama else
None, jenis if jenis else None, float(harga) if harga else None)
        elif pilihan == "4":
            kode_produk = input("Masukkan Kode Produk yang akan
dihapus: ")
            crud.hapus_produk(kode_produk)
        elif pilihan == "5":
            break
        else:
            print("Pilihan tidak valid (Masukkan Menu Yang
Sesuai).")

    elif pilih == "3":
        while True:
            print("\nMenu Transaksi:")
            print("1. Tambah Transaksi")
            print("2. Kembali ke Menu Utama")

            pilihan = input("Pilih menu transaksi: ")

            if pilihan == "1":
                crud.tambah_transaksi()
            elif pilihan == "2":
                break
            else:
                print("Pilihan tidak valid (Masukkan Menu Yang
Sesuai).")

    elif pilih == "4":
        while True:
            print("\nMenu Struk:")
            print("1. Lihat Struk Terbaru")
            print("2. Kembali ke Menu Utama")

            pilihan = input("Pilih menu struk: ")

            if pilihan == "1":
                crud.cetak_struk()
            elif pilihan == "2":
                break
            else:
                print("Pilihan tidak valid (Masukkan Menu Yang
Sesuai).")

```

```

elif pilih == "5":
    crud.close_connection()
    break
else:
    print("Pilihan tidak valid (Masukkan Menu Yang Sesuai).")

```

Kode diatas adalah menu yang akan ditampilkan dan user akan menginputkan semuanya yang ingin iinputkan.

## 18. HASIL RUNNING

```

Menu Warung Penjualan Sederhana:
1. Pegawai
2. Produk
3. Transaksi
4. Struk
5. Exit
Pilih menu: 1

```

```

Menu Pegawai:
1. Tambah Pegawai
2. Lihat Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Kembali ke Menu Utama
Pilih menu pegawai: 1

```

```

Menu Pegawai:
1. Tambah Pegawai
2. Lihat Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Kembali ke Menu Utama
Pilih menu pegawai: 1
Masukkan NIK: 432
Masukkan Nama Pegawai: joko
Masukkan Alamat: bantul
Data Pegawai berhasil ditambahkan.

```

```

Menu Pegawai:
1. Tambah Pegawai
2. Lihat Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Kembali ke Menu Utama
Pilih menu pegawai: 1
Masukkan NIK: 123
Masukkan Nama Pegawai: jojo
Masukkan Alamat: sleman
NIK 123 sudah terdaftar. Gagal menambahkan Pegawai.

```

```

Menu Pegawai:
1. Tambah Pegawai
2. Lihat Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Kembali ke Menu Utama
Pilih menu pegawai: 2

```

NIK	Nama_Pegawai	Alamat
123	aji	jombor
321	kharisma	sleman
432	joko	bantul

```

Menu Pegawai:
1. Tambah Pegawai
2. Lihat Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Kembali ke Menu Utama
Pilih menu pegawai: 3
Masukkan NIK yang akan diupdate: 123
Masukkan Nama baru (kosongkan jika tidak ingin diubah):
Masukkan Alamat baru (kosongkan jika tidak ingin diubah): jogja
Data Pegawai berhasil diperbarui.

```

```

Menu Pegawai:
1. Tambah Pegawai
2. Lihat Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Kembali ke Menu Utama
Pilih menu pegawai: 2

```

NIK	Nama_Pegawai	Alamat
123	aji	jogja
321	kharisma	sleman
432	joko	bantul

```

Menu Pegawai:
1. Tambah Pegawai
2. Lihat Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Kembali ke Menu Utama
Pilih menu pegawai: 4
Masukkan NIK yang akan dihapus: 123
Terjadi kesalahan saat menghapus pegawai: 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`penjualan`.`struk`, CONSTRAINT `struk_ibfk_2` FOREIGN KEY (`Nama_Pegawai`) REFERENCES `pegawai` (`Nama_Pegawai`))

Menu Pegawai:
1. Tambah Pegawai
2. Lihat Pegawai
3. Update Pegawai
4. Hapus Pegawai
5. Kembali ke Menu Utama
Pilih menu pegawai: 4
Masukkan NIK yang akan dihapus: 432
Data Pegawai berhasil dihapus.

```

				NIK	Nama_Pegawai	Alamat
<input type="checkbox"/>		Edit		Copy		Delete
				123	aji	jogja
<input type="checkbox"/>		Edit		Copy		Delete
				321	kharisma	sleman

```
Menu Produk:
1. Tambah Produk
2. Lihat Produk
3. Update Produk
4. Hapus Produk
5. Kembali ke Menu Utama
Pilih menu produk: 1
Masukkan Nama Produk: nasgor
Masukkan Jenis Produk: makanan
Masukkan Harga Produk: 10000
Data Produk dengan Kode P03 berhasil ditambahkan.
```

```
Menu Produk:
1. Tambah Produk
2. Lihat Produk
3. Update Produk
4. Hapus Produk
5. Kembali ke Menu Utama
Pilih menu produk: 2
+-----+-----+-----+-----+
| Kode_Produk | Nama_Produk | Jenis_Produk | Harga |
+-----+-----+-----+-----+
| P01         | golda       | minuman     | 3000  |
+-----+-----+-----+-----+
| P02         | taro        | snack       | 5000  |
+-----+-----+-----+-----+
| P03         | nasgor      | makanan     | 10000 |
+-----+-----+-----+-----+
```

```
Menu Produk:
1. Tambah Produk
2. Lihat Produk
3. Update Produk
4. Hapus Produk
5. Kembali ke Menu Utama
Pilih menu produk: 3
Masukkan Kode Produk yang akan diupdate: P02
Masukkan Nama baru (kosongkan jika tidak ingin diubah):
Masukkan Jenis baru (kosongkan jika tidak ingin diubah):
Masukkan Harga baru (kosongkan jika tidak ingin diubah): 3500
Data Produk berhasil diperbarui.
```













```
Menu Produk:
1. Tambah Produk
2. Lihat Produk
3. Update Produk
4. Hapus Produk
5. Kembali ke Menu Utama
Pilih menu produk: 4
Masukkan Kode Produk yang akan dihapus: P03
Data Produk berhasil dihapus.
```

```
Menu Produk:
1. Tambah Produk
2. Lihat Produk
3. Update Produk
4. Hapus Produk
5. Kembali ke Menu Utama
Pilih menu produk: 2
+-----+-----+-----+-----+
| Kode_Produk | Nama_Produk | Jenis_Produk | Harga |
+-----+-----+-----+-----+
| P01         | golda       | minuman     | 3000  |
+-----+-----+-----+-----+
| P02         | taro        | snack       | 3500  |
+-----+-----+-----+-----+
| P03         | nasgor      | makanan     | 10000 |
+-----+-----+-----+-----+
```

```
Menu Produk:
1. Tambah Produk
2. Lihat Produk
3. Update Produk
4. Hapus Produk
5. Kembali ke Menu Utama
Pilih menu produk: 2
+-----+-----+-----+-----+
| Kode_Produk | Nama_Produk | Jenis_Produk | Harga |
+-----+-----+-----+-----+
| P01         | golda       | minuman     | 3000  |
+-----+-----+-----+-----+
| P02         | taro        | snack       | 3500  |
+-----+-----+-----+-----+
```

					Kode_Produk	Nama_Produk	Jenis_Produk	Harga
<input type="checkbox"/>		Edit		Copy		Delete		
					P01	golda	minuman	3000
<input type="checkbox"/>		Edit		Copy		Delete		
					P02	taro	snack	3500

```
Menu Transaksi:
1. Tambah Transaksi
2. Kembali ke Menu Utama
Pilih menu transaksi: 1
Masukkan nama pegawai: aji
Masukkan nama produk: taro
Masukkan jumlah produk: 2
Apakah ingin menambah produk lagi? (ya/tidak): ya
Masukkan nama produk: golda
Masukkan jumlah produk: 1
Apakah ingin menambah produk lagi? (ya/tidak): tidak
Transaksi dan struk dengan No Transaksi T04 berhasil ditambahkan.
```

<div>←T→</div>				No_Transaksi	Detail_Transaksi
<input type="checkbox"/>	 Edit	 Copy	 Delete	T01	2024-12-12 22:47:43
<input type="checkbox"/>	 Edit	 Copy	 Delete	T02	2024-12-12 22:54:40
<input type="checkbox"/>	 Edit	 Copy	 Delete	T03	2024-12-12 22:57:50
<input type="checkbox"/>	 Edit	 Copy	 Delete	T04	2024-12-13 00:04:55

#### Menu Transaksi:

1. Tambah Transaksi

2. Kembali ke Menu Utama

Pilih menu transaksi: 1

Masukkan nama pegawai: jojo

Masukkan nama produk: golda

Masukkan jumlah produk: 2

Apakah ingin menambah produk lagi? (ya/tidak): tidak

Pegawai 'jojo' tidak ditemukan dalam database.

#### Menu Warung Penjualan Sederhana:

1. Pegawai

2. Produk

3. Transaksi

4. Struk

5. Exit

Pilih menu: 4

#### Menu Struk:

1. Lihat Struk Terbaru

2. Kembali ke Menu Utama

Pilih menu struk: 1

```
=====
                        STRUK TRANSAKSI
=====
Nama Produk           : taro
Jumlah Produk         : 2
Total Per Item        : Rp 7,000
-----
Nama Produk           : golda
Jumlah Produk         : 1
Total Per Item        : Rp 3,000
-----
Total Harga           : Rp 10,000
=====
```



No_Transaksi	Nama_Produk	Jumlah_Produk	Total_Harga	Nama_Pegawai
T01	golda	3	9000	aji
T01	taro	2	10000	aji
T02	golda	2	6000	aji
T02	taro	1	5000	aji
T03	golda	2	6000	aji
T03	taro	1	5000	aji
T04	taro	2	7000	aji
T04	golda	1	3000	aji