

DATABASE CHALLENGE
PEMROGRAMAN BERORIENTASI OBJEK

T. A. Semester Ganjil 2024/2025



5230411302 AYU KUSUMA SETYANINGSIH

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA
YOGYAKARTA

2024

PENDAHULUAN

Sistem penjualan adalah komponen penting dalam berbagai jenis bisnis yang bertujuan untuk mengelola data transaksi penjualan secara efektif dan efisien. Dalam tugas ini, kami mengembangkan sistem penjualan berbasis database menggunakan MySQL untuk penyimpanan data dan Python untuk pengolahan data. Sistem ini dirancang untuk mengelola data produk, pegawai, transaksi penjualan, dan struk yang dihasilkan dari transaksi tersebut.

Tujuan dari proyek ini adalah untuk mengimplementasikan prinsip-prinsip pemrograman berorientasi objek dalam konteks pengelolaan database dengan memanfaatkan MySQL sebagai sistem manajemen basis data. Dengan pendekatan ini, kami dapat menangani berbagai operasi penting dalam sistem penjualan, seperti menambah produk, mencatat transaksi, dan menghasilkan struk secara terstruktur.

Melalui tugas ini, kami berfokus pada penerapan query SQL untuk menangani data transaksi dan produk, serta pengorganisasian kode program dengan memanfaatkan objek-objek yang mewakili entitas dalam sistem penjualan. Selain itu, kami juga mengimplementasikan prosedur untuk mengelola data penjualan dan memberikan laporan transaksi yang dilakukan oleh pelanggan.

Dengan menggunakan Python sebagai bahasa pemrograman dan MySQL sebagai basis data, proyek ini bertujuan untuk memberikan solusi yang efisien bagi pengelolaan penjualan dan menjadi dasar dalam pengembangan sistem yang lebih kompleks di masa mendatang.

DESAIN DAN IMPLEMENTASI KODE PROGRAM

1. Menghubungkan Aplikasi Python Dengan Database Mysql

```
import mysql.connector
from tabulate import tabulate

conn = mysql.connector.connect(
    user = "root",
    host = "localhost",
    password = "",
    database = "pbo_penjualan",
    port=3307
)

cur = conn.cursor()
```

Kode ini digunakan untuk menghubungkan aplikasi Python dengan database MySQL. Dengan menggunakan `mysql.connector`, kode ini mengatur koneksi ke database bernama `pbo_penjualan` di server lokal pada port 3307, menggunakan username "root" dan tanpa password. Setelah koneksi berhasil, sebuah cursor dibuat dengan `conn.cursor()` untuk memungkinkan eksekusi query SQL pada database tersebut. Selain itu, modul `tabulate` diimpor untuk memformat hasil query menjadi tabel yang rapi saat ditampilkan.

2. Membuat Table Pegawai

```
# Membuat Table Pegawai
cur.execute("""CREATE TABLE IF NOT EXISTS Pegawai (
            NIK CHAR(4) NOT NULL PRIMARY KEY,
            Nama VARCHAR(50),
            Alamat VARCHAR(255)
        ) """)
```

Kode ini digunakan untuk membuat tabel baru bernama "Pegawai" dalam database MySQL jika tabel tersebut belum ada. Tabel ini memiliki tiga kolom: NIK yang bertipe data CHAR(4) dan berfungsi sebagai primary key, Nama yang bertipe VARCHAR(50), serta Alamat yang bertipe VARCHAR(255). Dengan menggunakan perintah CREATE TABLE IF NOT EXISTS, tabel hanya akan dibuat jika belum ada tabel dengan nama yang sama.

3. Membuat Table Transaksi

```
# Membuat Table Transaksi
cur.execute("""CREATE TABLE IF NOT EXISTS Transaksi (
                No_Transaksi CHAR(4) NOT NULL PRIMARY KEY,
                Detail_Transaksi VARCHAR(255)
            ) """)
```

Kode ini digunakan untuk membuat tabel "Transaksi" dalam database MySQL jika tabel tersebut belum ada. Tabel ini memiliki dua kolom: No_Transaksi yang bertipe data CHAR(4) dan berfungsi sebagai primary key, serta Detail_Transaksi yang bertipe VARCHAR(255) untuk menyimpan informasi rincian transaksi. Perintah CREATE TABLE IF NOT EXISTS memastikan bahwa tabel hanya dibuat jika belum ada tabel dengan nama yang sama.

4. Membuat Table Struk

```
# Membuat Table Struk
cur.execute("""CREATE TABLE IF NOT EXISTS Struk (
                No_Transaksi CHAR(4) NOT NULL,
                Nama_Pegawai VARCHAR(50),
                Nama_Produk VARCHAR(50),
                Jumlah_Produk INT,
                Total_Harga FLOAT
            ) """)
```

Kode ini digunakan untuk membuat tabel "Struk" dalam database MySQL jika tabel tersebut belum ada. Tabel ini memiliki lima kolom: No_Transaksi yang bertipe CHAR(4), Nama_Pegawai dan Nama_Produk yang bertipe VARCHAR(50),

Jumlah_Produk yang bertipe INT, serta Total_Harga yang bertipe FLOAT. Tabel ini dirancang untuk menyimpan informasi tentang transaksi, pegawai, produk yang dibeli, jumlah produk, dan total harga. Perintah CREATE TABLE IF NOT EXISTS memastikan bahwa tabel hanya dibuat jika belum ada tabel dengan nama yang sama.

5. Membuat Table Produk

```
# Membuat Table Produk
cur.execute("""CREATE TABLE IF NOT EXISTS Produk (
            Kode_Produk INT NOT NULL PRIMARY KEY,
            Nama_Produk VARCHAR(50),
            Jenis_Produk VARCHAR(20),
            Harga_Produk DECIMAL(10, 2)
        ) """)
```

Kode ini digunakan untuk membuat tabel "Produk" dalam database MySQL jika tabel tersebut belum ada. Tabel ini memiliki empat kolom: Kode_Produk yang bertipe INT dan berfungsi sebagai primary key, Nama_Produk yang bertipe VARCHAR(50), Jenis_Produk yang bertipe VARCHAR(20), serta Harga_Produk yang bertipe DECIMAL(10, 2) untuk menyimpan harga produk dengan presisi dua angka desimal. Perintah CREATE TABLE IF NOT EXISTS memastikan bahwa tabel hanya dibuat jika belum ada tabel dengan nama yang sama.

6. Relasi Struk – Transaksi

```
cur.execute("""ALTER TABLE Struk
ADD CONSTRAINT FK_StrukTransaksi FOREIGN KEY (No_Transaksi) REFERENCES
Transaksi(No_Transaksi);
""")
```

Kode ini digunakan untuk menambahkan relasi foreign key pada tabel "Struk". Dengan perintah ini, kolom No_Transaksi di tabel "Struk" dihubungkan dengan kolom No_Transaksi di tabel "Transaksi" sebagai referensinya. Relasi ini memastikan bahwa setiap nilai di kolom No_Transaksi tabel "Struk" harus sesuai dengan nilai yang ada di kolom No_Transaksi tabel "Transaksi," sehingga menjaga integritas data antar tabel.

7. Relasi Transaksi – Pegawai

```
cur.execute("""ALTER TABLE Transaksi
ADD COLUMN NIK CHAR(4),
ADD CONSTRAINT FK_TransaksiPegawai FOREIGN KEY (NIK) REFERENCES
Pegawai (NIK);
""")
```

Kode ini digunakan untuk menambahkan kolom NIK ke tabel "Transaksi" dan menetapkan kolom tersebut sebagai foreign key yang merujuk ke kolom NIK di tabel "Pegawai." Relasi ini memastikan bahwa setiap nilai NIK di tabel "Transaksi" harus sesuai dengan nilai yang ada di tabel "Pegawai," sehingga menghubungkan transaksi dengan data pegawai yang bertanggung jawab atas transaksi tersebut.

8. Relasi Transaksi – Produk

```
cur.execute("""ALTER TABLE Transaksi
ADD COLUMN Kode_Produk INT,
ADD CONSTRAINT FK_TransaksiProduk FOREIGN KEY (Kode_Produk) REFERENCES
Produk (Kode_Produk);
""")
```

Kode ini digunakan untuk menambahkan kolom Kode_Produk ke tabel "Transaksi" dan menetapkan kolom tersebut sebagai foreign key yang merujuk ke kolom Kode_Produk di tabel "Produk." Relasi ini memastikan bahwa setiap nilai Kode_Produk di tabel "Transaksi" harus sesuai dengan nilai yang ada di tabel "Produk," sehingga menghubungkan transaksi dengan produk terkait.

9. Data Produk

```
produk = {
    "snack": {
        "P001": ("Chitato", 10000),
        "P002": ("Lays", 15000),
        "P003": ("Potabee", 12000),
    },
    "minuman": {
        "M001": ("Aqua", 3000),
        "M002": ("Jus", 7000),
        "M003": ("Susu", 5000),
    },
    "makanan": {
        "F001": ("Ayam Goreng", 8000),
        "F002": ("Nasi Goreng", 12000),
    },
}
```

```

        "F003": ("Mie Ayam", 15000),
    },
}

```

Kode ini mendefinisikan sebuah dictionary bernama produk yang mengorganisasi data produk berdasarkan kategori, yaitu "snack," "minuman," dan "makanan." Setiap kategori memiliki kode produk sebagai kunci (misalnya, "P001") dengan nilai berupa tuple yang mencakup nama produk (seperti "Chitato") dan harga produk (seperti 10000). Struktur ini memudahkan pengelolaan dan pencarian informasi produk berdasarkan kategori dan kode.

10. While True

```

while True:
    print("\n===== Menu Sistem Penjualan =====")
    print("1. Tampilkan Data Produk")
    print("2. Masukkan Data Pegawai")
    print("3. Tambahkan Produk")
    print("4. Masukkan Data Transaksi")
    print("5. Tampilkan Struk")
    print("6. Ubah Data")
    print("7. Hapus Data")
    print("0. Keluar")
    menu = input("Pilih menu: ")

```

Setiap pilihan pengguna ditangkap melalui input menu, yang nantinya akan digunakan untuk menjalankan fungsi atau logika tertentu berdasarkan opsi yang dipilih. Program ini berguna untuk mengelola berbagai aspek dari sistem penjualan secara interaktif.

11. Menu 1: Data Produk

```

if menu == '1':
    # Menampilkan Data Produk
    print("\n===== Data Produk =====")
    for kategori, items in produk.items():
        print(f"\nKategori: {kategori.capitalize()}")
        table = [[Kode, nama, f"Rp {harga:,}"] for Kode, (nama,
harga) in items.items()]
        print(tabulate(table, headers=["Kode", "Produk", "Nama
Produk", "Harga"], tablefmt="grid"))

```

Kode ini menangani opsi menu untuk menampilkan data produk. Ketika pengguna memilih menu 1, program akan mencetak semua data produk yang dikelompokkan berdasarkan kategori seperti "snack," "minuman," dan "makanan." Setiap kategori

ditampilkan dengan format tabel menggunakan modul tabulate. Tabel berisi kolom "Kode Produk," "Nama Produk," dan "Harga," di mana harga diformat dengan tanda pemisah ribuan. Fungsi ini memudahkan pengguna untuk melihat informasi lengkap tentang produk dalam tampilan yang terorganisir dan mudah dibaca.

12. Menu 2: Input Sata Pegawai

```
elif menu == '2':
    # Input Data Pegawai
    print("\n===== Input Data Pegawai =====")
    NIK = input("Masukkan NIK Pegawai: ")
    Nama_Pegawai = input("Masukkan Nama Pegawai: ")
    Alamat = input("Masukkan Alamat Pegawai: ")

    # Cek apakah NIK sudah ada
    cur.execute("SELECT * FROM Pegawai WHERE NIK = %s", [NIK])
    if cur.fetchone():
        print("NIK sudah ada. Tidak menambahkan data Pegawai.")
    else:
        try:
            cur.execute("""INSERT INTO Pegawai (NIK, Nama, Alamat)
VALUES (%s, %s, %s)""",
                        [NIK, Nama_Pegawai, Alamat])
            conn.commit()
            print("Data Pegawai berhasil ditambahkan!")
        except mysql.connector.Error as err:
            print(f"Error: {err}")
```

Kode ini menangani input data pegawai baru. Pengguna diminta memasukkan NIK, nama, dan alamat pegawai. Program memeriksa apakah NIK sudah ada di tabel "Pegawai." Jika NIK sudah ada, data tidak akan ditambahkan. Jika belum, data pegawai baru disimpan ke database menggunakan perintah INSERT, dan perubahan dikonfirmasi dengan conn.commit(). Jika terjadi kesalahan selama proses, pesan error ditampilkan.

13. Menu 3: Tambahkan Produk

```
elif menu == '3':
    # Tambahkan Produk
    print("\n===== Tambahkan Produk =====")

    # Memilih Jenis Produk
    print("Pilih Jenis Produk:")
    print("1. Snack")
    print("2. Minuman")
    print("3. Makanan")
    pilihan_jenis = input("Pilih jenis produk (1/2/3): ")
```



```

        if pilihan_jenis == '1':
            jenis_produk = "snack"
        elif pilihan_jenis == '2':
            jenis_produk = "minuman"
        elif pilihan_jenis == '3':
            jenis_produk = "makanan"
        else:
            print("Pilihan tidak valid!")
            break

        # Menampilkan Produk Berdasarkan Pilihan Jenis
        print(f"\nPilih Produk {jenis_produk.capitalize()}:")
        for kode, (nama, harga) in produk[jenis_produk].items():
            print(f"{kode}: {nama} - Rp{harga}")

        # Memilih Produk
        pilihan_produk = input(f"Masukkan kode produk yang dipilih: ")

        # Mengecek apakah produk yang dipilih valid
        if pilihan_produk in produk[jenis_produk]:
            nama_produk, harga_produk = produk[jenis_produk][pilihan_produk]
            print(f"\nProduk: {nama_produk}\nHarga: Rp{harga_produk}")
        else:
            print("Produk tidak valid!")
            break

        Nama_Produk = nama_produk
        Harga_Produk = harga_produk

        try:
            cur.execute("""INSERT INTO Produk (Kode_Produk,
            Nama_Produk, Jenis_Produk, Harga_Produk)
            VALUES (%s, %s, %s, %s)""",
            [pilihan_produk, Nama_Produk,
            jenis_produk, Harga_Produk])
            conn.commit()
            print("Produk berhasil ditambahkan!")
        except mysql.connector.Error as err:
            print(f"Error: {err}")

```

Kode ini menangani penambahan data produk ke database. Pengguna diminta memilih jenis produk (snack, minuman, makanan) dan produk tertentu berdasarkan kode. Setelah validasi kode produk berhasil, detail produk (nama dan harga) ditampilkan. Data produk kemudian dimasukkan ke tabel "Produk" di database menggunakan perintah INSERT. Jika proses berhasil, perubahan disimpan dengan conn.commit(). Jika terjadi kesalahan, pesan error akan ditampilkan.

14. Menu 4: Input Data Transaksi

```
elif menu == '4':
    # Masukkan Data Transaksi
    print("\n===== Input Data Transaksi =====")
    No_Transaksi = input("Masukkan No Transaksi: ")
    Detail_Transaksi = input("Masukkan Detail Transaksi: ")
    NIK = input("Masukkan NIK Pegawai: ")
    Kode_Produk = input("Masukkan Kode Produk: ")
    Jumlah_Produk = int(input("Masukkan Jumlah Produk: "))

    # Ambil data produk dari tabel Produk
    cur.execute("SELECT Nama_Produk, Harga_Produk FROM Produk
WHERE Kode_Produk = %s", [Kode_Produk])
    produk = cur.fetchone()

    if produk:
        Nama_Produk = produk[0]
        Harga_Produk = produk[1]
        Total_Harga = Harga_Produk * Jumlah_Produk

        # Menyimpan Data Transaksi
        try:
            cur.execute("""INSERT INTO Transaksi (No_Transaksi,
Detail_Transaksi, NIK, Kode_Produk)
VALUES (%s, %s, %s, %s)""",
[No_Transaksi, Detail_Transaksi, NIK,
Kode_Produk])
            conn.commit()

            # Menyimpan Data Struk
            cur.execute("""INSERT INTO Struk (No_Transaksi,
Nama_Pegawai, Nama_Produk, Jumlah_Produk, Total_Harga)
VALUES (%s, %s, %s, %s, %s)""",
[No_Transaksi, NIK, Nama_Produk,
Jumlah_Produk, Total_Harga])
            conn.commit()

            print("Transaksi berhasil ditambahkan!")
        except mysql.connector.Error as err:
            print(f"Error: {err}")
        else:
            print("Produk tidak ditemukan!")
```

Kode ini menangani input data transaksi baru. Pengguna memasukkan nomor transaksi, detail transaksi, NIK pegawai, kode produk, dan jumlah produk. Program mengambil data produk dari tabel "Produk" berdasarkan kode produk. Jika produk valid, nama dan harga produk digunakan untuk menghitung total harga. Data transaksi disimpan ke tabel "Transaksi," dan data terkait seperti struk disimpan ke tabel "Struk." Perubahan disimpan dengan `conn.commit()`, dan jika ada kesalahan, pesan error akan ditampilkan. Jika produk tidak ditemukan, pesan kesalahan ditampilkan.

15. Menu 5: Tampilkan Struk

```
elif menu == '5':
    # Tampilkan Struk
    print("\n===== Tampilkan Struk =====")
    No_Transaksi = input("Masukkan No Transaksi untuk menampilkan struk: ")

    # Query untuk mengambil data struk
    cur.execute("""
        SELECT      s.No_Transaksi,      p>Nama      AS      Nama_Pegawai,
s>Nama_Produk, s.Jumlah_Produk, s.Total_Harga
        FROM Struk s
        JOIN Pegawai p ON p>Nama = s>Nama_Pegawai # Menghubungkan
Nama di tabel Pegawai dan Nama_Pegawai di Struk
        WHERE s.No_Transaksi = %s
        """, [No_Transaksi])

    struk = cur.fetchall()

    if struk:
        print("\n===== Struk Transaksi =====")
        print(tabulate(struk, headers=["No Transaksi", "Nama Pegawai", "Nama Produk", "Jumlah Produk", "Total Harga"],
            tablefmt="grid"))
    else:
        print("Struk tidak ditemukan!")
```

Kode ini menangani penampilan struk transaksi berdasarkan nomor transaksi yang dimasukkan pengguna. Program menjalankan query untuk mengambil data dari tabel "Struk," termasuk informasi pegawai, produk, jumlah produk, dan total harga, dengan menghubungkan tabel "Struk" dan "Pegawai." Jika data ditemukan, struk ditampilkan dalam format tabel menggunakan modul tabulate. Jika tidak ada data yang sesuai, pesan "Struk tidak ditemukan" akan ditampilkan.

16. Menu 6: Ubah Data

```
elif menu == '6':
    # Ubah Data
    print("\n===== Ubah Data =====:")
    print("1. Ubah Pegawai")
    print("2. Ubah Produk")
    print("3. Ubah Transaksi")
    pilihan_ubah = input("Pilih yang ingin diubah: ")
    if pilihan_ubah == '1':
        NIK = input("Masukkan NIK Pegawai yang ingin diubah: ")
        Nama_Baru = input("Masukkan Nama Baru: ")
        Alamat_Baru = input("Masukkan Alamat Baru: ")
        # Update data pegawai di database
```

```

        cur.execute("""UPDATE Pegawai SET Nama = %s, Alamat = %s
WHERE NIK = %s""",
                    [Nama_Baru, Alamat_Baru, NIK])
        conn.commit()
        print(f>Data Pegawai dengan NIK {NIK} berhasil diubah.")
    elif pilihan_ubah == '2':
        Kode_Produk = input("Masukkan Kode Produk yang ingin
diubah: ")
        Nama_Produk_Baru = input("Masukkan Nama Produk Baru: ")
        # Update data produk di database
        cur.execute("""UPDATE Produk SET Nama_Produk = %s WHERE
Kode_Produk = %s""",
                    [Nama_Produk_Baru, Kode_Produk])
        conn.commit()
        print(f>Data Produk dengan Kode {Kode_Produk} berhasil
diubah.")
    elif pilihan_ubah == '3':
        No_Transaksi = input("Masukkan No Transaksi yang ingin
diubah: ")
        Detail_Transaksi_Baru = input("Masukkan Detail Transaksi
Baru: ")
        # Update data transaksi di database
        cur.execute("""UPDATE Transaksi SET Detail_Transaksi = %s
WHERE No_Transaksi = %s""",
                    [Detail_Transaksi_Baru, No_Transaksi])
        conn.commit()
        print(f>Data Transaksi dengan No {No_Transaksi} berhasil
diubah.")
    else:
        print("Pilihan tidak valid.")

```

Kode ini memungkinkan pengguna untuk mengubah data pada tiga entitas: pegawai, produk, dan transaksi. Pengguna memilih jenis data yang ingin diubah (pegawai, produk, atau transaksi), kemudian memasukkan informasi baru. Program kemudian menjalankan query SQL untuk memperbarui data yang dipilih dalam database. Setelah perubahan berhasil, sistem memberi tahu pengguna bahwa data telah diperbarui. Jika pilihan yang dimasukkan tidak valid, program akan menampilkan pesan kesalahan.

17. Menu 7: Hapus Data

```
elif menu == '7':
    # Hapus Data
    print("\nHapus Data:")
    print("1. Hapus Pegawai")
    print("2. Hapus Produk")
    print("3. Hapus Transaksi")
    pilihan_hapus = input("Pilih yang ingin dihapus: ")
    if pilihan_hapus == '1':
        NIK = input("Masukkan NIK Pegawai yang ingin dihapus: ")
        # Hapus data pegawai di database
        cur.execute("""DELETE FROM Pegawai WHERE NIK = %s""",
[NIK])
        conn.commit()
        print(f>Data Pegawai dengan NIK {NIK} berhasil dihapus.")
    elif pilihan_hapus == '2':
        Kode_Produk = input("Masukkan Kode Produk yang ingin
dihapus: ")
        # Hapus data produk di database
        cur.execute("""DELETE FROM Produk WHERE Kode_Produk =
%s""", [Kode_Produk])
        conn.commit()
        print(f>Data Produk dengan Kode {Kode_Produk} berhasil
dihapus.")
    elif pilihan_hapus == '3':
        No_Transaksi = input("Masukkan No Transaksi yang ingin
dihapus: ")
        # Hapus data transaksi di database
        cur.execute("""DELETE FROM Transaksi WHERE No_Transaksi =
%s""", [No_Transaksi])
        conn.commit()
        print(f>Data Transaksi dengan No {No_Transaksi} berhasil
dihapus.")
    else:
        print("Pilihan tidak valid.")
```

Kode ini memungkinkan pengguna untuk menghapus data pada tiga entitas: pegawai, produk, dan transaksi. Pengguna memilih jenis data yang ingin dihapus, kemudian memasukkan identifier (seperti NIK, Kode Produk, atau No Transaksi). Program kemudian menjalankan query SQL untuk menghapus data yang dipilih dari database. Setelah penghapusan berhasil, sistem memberi tahu pengguna bahwa data telah dihapus. Jika pilihan yang dimasukkan tidak valid, program akan menampilkan pesan kesalahan.

PENGUJIAN DAN HASIL

1. Menu Sistem Penjualan

```
===== Menu Sistem Penjualan =====
1. Tampilkan Data Produk
2. Masukkan Data Pegawai
3. Tambahkan Produk
4. Masukkan Data Transaksi
5. Tampilkan Struk
6. Ubah Data
7. Hapus Data
0. Keluar
```

2. Data Produk

```
Pilih menu: 1

===== Data Produk =====

Kategori: Snack
+-----+-----+-----+
| Kode Produk | Nama Produk | Harga   |
+=====+=====+=====+
| P001        | Chitato    | Rp 10,000 |
+-----+-----+-----+
| P002        | Lays       | Rp 15,000 |
+-----+-----+-----+
| P003        | Potabee    | Rp 12,000 |
+-----+-----+-----+







Kategori: Minuman
+-----+-----+-----+
| Kode Produk | Nama Produk | Harga   |
+=====+=====+=====+
| M001        | Aqua       | Rp 3,000 |
+-----+-----+-----+
```

3. Input Data Pegawai

```
Pilih menu: 2

===== Input Data Pegawai =====
Masukkan NIK Pegawai: 1345
Masukkan Nama Pegawai: Mark Lee
Masukkan Alamat Pegawai: Jl. Kenangan Hitam
Data Pegawai berhasil ditambahkan!
```

Hasil di Database

	NIK	Nama	Alamat
<input type="checkbox"/>  Edit  Copy  Delete	1290	Na Jaemin	Jl. Merpati Unggu
<input type="checkbox"/>  Edit  Copy  Delete	1345	Mark Lee	Jl. Kenangan Hitam

4. Tambahkan Produk


```
Pilih menu: 3

===== Tambahkan Produk =====
Pilih Jenis Produk:
1. Snack
2. Minuman
3. Makanan
Pilih jenis produk (1/2/3): 1

Pilih Produk Snack:
P001: Chitato - Rp10000
P002: Lays - Rp15000
P003: Potabee - Rp12000
Masukkan kode produk yang dipilih: P002

Produk: Lays
Harga: Rp15000
Produk berhasil ditambahkan!
```

Hasil di Database


	Kode_Produk	Nama_Produk	Jenis_Produk	Harga_Produk
<input type="checkbox"/>  Edit  Copy  Delete	1	Susu	minuman	5000.00
<input type="checkbox"/>  Edit  Copy  Delete	2	Lays	snack	15000.00

5. Input Data Transaksi

```
Pilih menu: 4

===== Input Data Transaksi =====
Masukkan No Transaksi: TRX2
Masukkan Detail Transaksi: Pembelian Snack
Masukkan NIK Pegawai: 1345
Masukkan Kode Produk: P002
Masukkan Jumlah Produk: 2
```

Hasil di Database

	No_Transaksi	Detail_Transaksi	NIK	Kode_Produk
<input type="checkbox"/>  Edit  Copy  Delete	TRX1	Pembelian Minuman	1290	0

6. Tampilkan Struk

```
Pilih menu: 5

===== Tampilkan Struk =====
Masukkan No Transaksi untuk menampilkan struk: TRX2
```

Hasil di Dataset

No_Transaksi	Nama_Pegawai	Nama_Produk	Jumlah_Produk	Total_Harga
TRX1	1290	Susu	3	15000

7. Ubah Data

```
Pilih menu: 6

===== Ubah Data =====
1. Ubah Pegawai
2. Ubah Produk
3. Ubah Transaksi
Pilih yang ingin diubah: 1
Masukkan NIK Pegawai yang ingin diubah: 1290
Masukkan Nama Baru: Huang Renjun
Masukkan Alamat Baru: Jl. Merpati Hijau
Data Pegawai dengan NIK 1290 berhasil diubah.
```

Hasil di Dataset









				NIK	Nama	Alamat
<input type="checkbox"/>		Edit		Copy		Delete
				1290	Huang Renjun	Jl. Merpati Hijau
<input type="checkbox"/>		Edit		Copy		Delete
				1345	Mark Lee	Jl. Kenangan Hitam

8. Hapus Data

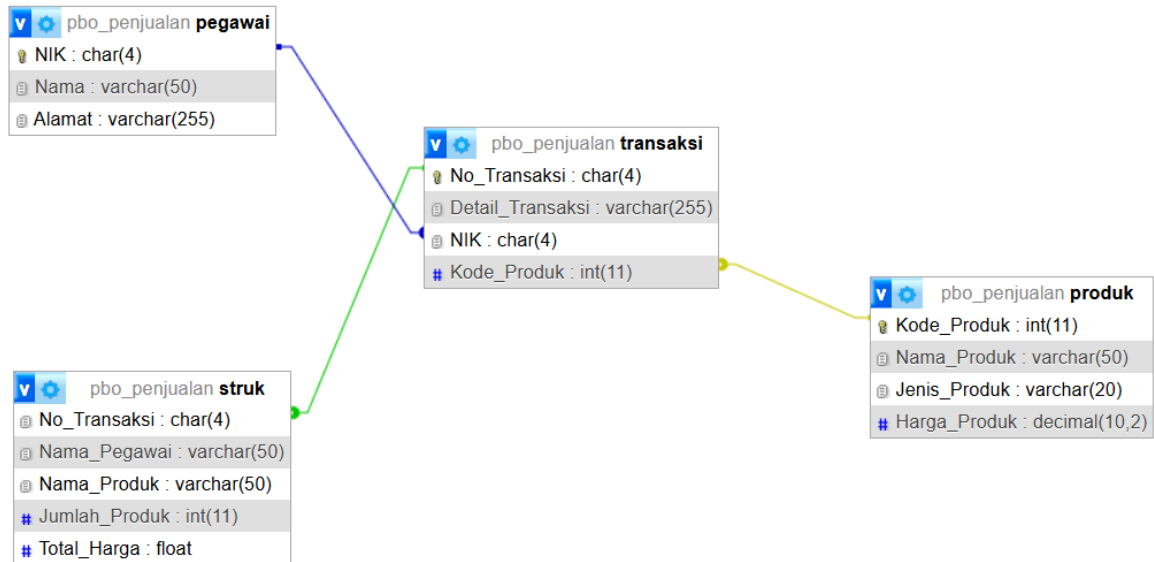
```
Pilih menu: 7

===== Hapus Data =====
1. Hapus Pegawai
2. Hapus Produk
3. Hapus Transaksi
Pilih yang ingin dihapus: 1
Masukkan NIK Pegawai yang ingin dihapus: 1345
Data Pegawai dengan NIK 1345 berhasil dihapus.
```

Hasil di Dataset

				NIK	Nama	Alamat
<input type="checkbox"/>		Edit		Copy		Delete
				1290	Huang Renjun	Jl. Merpati Hijau
	<input type="checkbox"/>	Check all	With selected:			Edit
						Copy
						Delete
						Export

TABEL RELASI



KESIMPULAN

Dalam implementasi sistem database untuk penjualan produk ini, kita telah berhasil membangun struktur tabel yang mendukung penyimpanan data produk, termasuk menampilkan, menambah, mengubah, dan menghapus data terkait produk, pegawai, dan transaksi.

Program ini menggunakan antarmuka berbasis teks yang memungkinkan pengguna untuk berinteraksi dengan database melalui pilihan menu. Setiap operasi dilakukan dengan query SQL yang sesuai untuk memastikan data ditangani dengan benar, seperti menambahkan produk, pegawai, atau transaksi, serta memodifikasi atau menghapus data yang sudah ada.

Secara keseluruhan, sistem ini memberikan solusi yang efisien dan terstruktur untuk mengelola data produk dalam sebuah bisnis penjualan. Dengan perbaikan dan pengembangan lebih lanjut, sistem ini dapat digunakan untuk mendukung skala yang lebih besar, termasuk integrasi dengan modul-modul lain seperti pengelolaan stok dan laporan penjualan.