

Najla Humaira Desni  
2211521002  
PBO Kelas B

Instruction :

Carilah materi tentang JDBC (Java Database Connectivity).  
Buatlah sebuah program yang merupakan lanjutan dari tugas *Sting and Date*. Dari tugas sebelumnya anda tambahkan JDBC sampai dengan CRUD (Create, Read, Update, Delete).

Jawab :

A. Kodingan

a.Class App

```
package demo;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class App
{
    private static boolean login(String correctUsername, String
correctPassword, String inputUsername, String inputPassword) {
        return correctUsername.equals(inputUsername) &&
correctPassword.equals(inputPassword);
    }

    private static String generateCaptcha() {
        return "jdhsqw";
    }

    private static boolean checkCaptcha(String correctCaptcha, String
inputCaptcha) {
        return correctCaptcha.equalsIgnoreCase(inputCaptcha);
    }

    private static void tambahData(Connection connection) {
        try {
            Scanner scanner = new Scanner(System.in);
```

```

        System.out.print("Masukkan Nama: ");
        String nama = scanner.nextLine();

        System.out.print("Masukkan Nomor HP: ");
        String nomorHp = scanner.nextLine();

        System.out.print("Masukkan Total Transaksi: ");
        double totalTransaksi = scanner.nextDouble();
        scanner.nextLine(); // Membersihkan buffer

        // Menyiapkan pernyataan SQL
        String sql = "INSERT INTO transaksi (nama, nomorHp,
totalTransaksi) VALUES (?, ?, ?)";
        try (PreparedStatement preparedStatement =
connection.prepareStatement(sql)) {
            // Mengatur parameter
            preparedStatement.setString(1, nama);
            preparedStatement.setString(2, nomorHp);
            preparedStatement.setDouble(3, totalTransaksi);

            // Menjalankan pernyataan
            int affectedRows = preparedStatement.executeUpdate();
            if (affectedRows > 0) {
                System.out.println("Data berhasil ditambahkan!");
            } else {
                System.out.println("Gagal menambahkan data.");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void tampilkanData(Connection connection) {
    try {
        // Menyiapkan pernyataan SQL
        String sql = "SELECT * FROM transaksi";
        try (Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(sql)) {

            // Menampilkan hasil query
            System.out.println("Data dalam tabel transaksi:");
            while (resultSet.next()) {
                String nama = resultSet.getString("nama");
                String nomorHp = resultSet.getString("nomorHp");
                double totalTransaksi =
resultSet.getDouble("totalTransaksi");

```

```

        System.out.println("Nama: " + nama + ", Nomor HP: " +
        nomorHp + ", Total Transaksi: " + totalTransaksi);
    }
}
} catch (SQLException e) {
    e.printStackTrace();
}
}

private static void hapusData(Connection connection, String nama) {
    try {
        // Menyiapkan pernyataan SQL DELETE
        String sql = "DELETE FROM transaksi WHERE nama = ?";
        try (PreparedStatement preparedStatement =
        connection.prepareStatement(sql)) {
            // Mengatur parameter
            preparedStatement.setString(1, nama);

            // Menjalankan pernyataan
            int affectedRows = preparedStatement.executeUpdate();
            if (affectedRows > 0) {
                System.out.println("Data berhasil dihapus!");
            } else {
                System.out.println("Data tidak ditemukan atau gagal
        dihapus.");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void updateData(Connection connection, String nama, double
totalTransaksiBaru) {
    try {
        // Menyiapkan pernyataan SQL UPDATE
        String sql = "UPDATE transaksi SET totalTransaksi = ? WHERE nama =
        ?";
        try (PreparedStatement preparedStatement =
        connection.prepareStatement(sql)) {
            // Mengatur parameter
            preparedStatement.setDouble(1, totalTransaksiBaru);
            preparedStatement.setString(2, nama);

            // Menjalankan pernyataan
            int affectedRows = preparedStatement.executeUpdate();
            if (affectedRows > 0) {
                System.out.println("Data berhasil diupdate!");
            } else {

```

```

        System.out.println("Data tidak ditemukan atau gagal
diupdate.");
    }
}
} catch (SQLException e) {
    e.printStackTrace();
}
}

public static void main( String[] args ) throws SQLException
{
    String url = "jdbc:mysql://localhost:3306/market_pbo";
    String dbuser = "root";
    String password = "";
    try {
        Connection connection = DriverManager.getConnection(url, dbuser,
password);
        System.out.println("Koneksi ke database berhasil!");
        // Jangan lupa untuk menutup koneksi setelah selesai menggunakan
connection.close();
    } catch (SQLException e) {
        System.err.println("Gagal terhubung ke database!");
        e.printStackTrace();
    }

    Scanner scanner = new Scanner(System.in);

    String username = "nana";
    String pass = "nana1234";
    boolean lgn=false;

    while (!lgn) {
        System.out.print("Masukkan username: ");
        String inputUsername = scanner.nextLine();

        System.out.print("Masukkan password: ");
        String inputPassword = scanner.nextLine();

        if (login(username, pass, inputUsername, inputPassword)) {
            String captcha = generateCaptcha();
            System.out.println("CAPTCHA: " + captcha);

            System.out.print("Masukkan CAPTCHA: ");
            String inputCaptcha = scanner.nextLine();

            if (checkCaptcha(captcha, inputCaptcha)) {
                System.out.println("Login berhasil!");
            }
        }
    }
}

```

```

        lgn=true;
    } else {
        System.out.println("Login gagal. CAPTCHA salah.");
    }
} else {
    System.out.println("Login gagal. Periksa kembali username dan
password.");
}
}
Date tanggal = new Date();
SimpleDateFormat dateFormat = new SimpleDateFormat("EEEE,
dd/MM/yyyy");
String tanggalTransaksi = dateFormat.format(tanggal);

SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm:ss z");
String waktuTransaksi = timeFormat.format(tanggal);

try {

    System.out.println("");
    MemberReguler tes1 = new MemberReguler(){};
    System.out.print("Masukkan Nama Pelanggan\t: ");
    tes1.namaPelanggan = scanner.nextLine();

    System.out.print("Masukkan Nomor HP\t: ");
    tes1.noHp = scanner.nextLine();

    System.out.print("Masukkan Alamat\t\t: ");
    tes1.alamat = scanner.nextLine();

    System.out.print("Masukkan Kode Barang\t: ");
    tes1.kodeBarang = scanner.nextLine();

    System.out.print("Masukkan Nama Barang\t: ");
    tes1.namaBarang = scanner.nextLine();

    System.out.print("Masukkan Harga Barang\t: ");
    tes1.hargaBarang = scanner.nextLong();

    System.out.print("Masukkan Jumlah Barang\t: ");
    tes1.jumlahBeli = scanner.nextLong();

    // Menghitung total bayar
    tes1.totalBayar = tes1.hargaBarang * tes1.jumlahBeli;

    // Menampilkan hasil
    System.out.println("");
    System.out.println("\t\033[1m\033[31mTres Mart\033[0m");
}

```

```

        System.out.println("Tanggal \t:" + tanggalTransaksi);
        System.out.println("Waktu \t\t:" + waktuTransaksi);
        System.out.println("=====");
        System.out.println("DATA PELANGGAN");
        System.out.println("-----");
        System.out.println("Nama Pelanggan\t:" +
tes1.namaPelanggan.toUpperCase());
        System.out.println("Nomor Hp\t:" + tes1.noHp);
        System.out.println("Alamat\t\t:" + tes1.alamat.toLowerCase());
        System.out.println("+++++");
        System.out.println("DATA PEMBELIAN BARANG");
        System.out.println("-----");
        System.out.println("Kode Barang\t:" +
tes1.kodeBarang.toUpperCase());
        System.out.println("Nama Barang\t:" + tes1.namaBarang);
        System.out.println("Harga Barang\t:" + tes1.hargaBarang);
        System.out.println("Jumlah Beli\t:" + tes1.jumlahBeli);
        System.out.println("TOTAL BAYAR\t:" + tes1.totalBayar);
        System.out.println("+++++");
        System.out.println("Kasir\t\t:" + tes1.namaKasir.toUpperCase());
        System.out.println("");

    } catch (java.util.InputMismatchException e) {
        System.out.println("Maaf, input tidak valid. Pastikan Anda
memasukkan nilai numerik untuk harga dan jumlah barang.");
    }

    try (Connection connection = DriverManager.getConnection(url, dbuser,
password)) {

        while (true) {
            System.out.println("Pilih operasi:");
            System.out.println("1. Tambah Data");
            System.out.println("2. Tampilkan Data");
            System.out.println("3. Hapus Data");
            System.out.println("4. Update Data");
            System.out.println("5. Keluar");
            System.out.print("Masukkan pilihan Anda: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Membersihkan buffer

            switch (choice) {
                case 1:
                    tambahData(connection);
                    break;

```

```

        case 2:
            tampilkanData(connection);
            break;
        case 3:
            System.out.print("Masukkan nama untuk menghapus data:
");
            String namaHapus = scanner.nextLine();
            hapusData(connection, namaHapus);
            break;
        case 4:
            System.out.print("Masukkan nama untuk mengupdate data:
");
            String namaUpdate = scanner.nextLine();

            System.out.print("Masukkan total transaksi baru: ");
            double totalTransaksiBaru = scanner.nextDouble();
            scanner.nextLine(); // Membersihkan buffer

            updateData(connection, namaUpdate,
totalTransaksiBaru);
            break;
        case 5:
            System.out.println("Program selesai.");
            System.exit(0);
        default:
            System.out.println("Pilihan tidak valid. Silakan coba
lagi.");
            break;
    }
}
} catch (SQLException e) {
    e.printStackTrace();
}finally {
    // Menutup scanner setelah selesai digunakan
    scanner.close();
}
}
}

```

b. class Member

```

package demo;

public abstract class Member {
    protected Integer poin = 0;
    String namaPelanggan;
    String kodeBarang;
}

```

```

String namaBarang;
Long hargaBarang;
Long jumlahBeli;
Long totalBayar;
String noHp;
String alamat;
String namaKasir="Nana";

public Integer getPoin(){
    return poin;
}

public Integer redeemPoin(Integer hargaSouvenir){
    return this.poin = this.poin - hargaSouvenir;
}

protected Integer getBonusPoin(Integer totalBayar){
    Integer poin = (int) (totalBayar / 10000);
    return poin;
}
}

```

c.class CanGetDiskon

```

package demo;

public interface CanGetDiskon {
    public Integer hitungTotalBayar(Integer jumlahBelanja);
}

```

d.class MemberReguler

```

package demo;

public class MemberReguler extends Member implements CanGetDiskon{

    public MemberReguler(Integer poin){
        this.poin=poin;
    }

    public MemberReguler(){}

    /*----- */
    ----- */
    public Integer hitungTotalBayar(Integer jumlahBelanja){
        double diskon = 0;
        if (jumlahBelanja < 500000) {

```



```

        int newPoin = (jumlahBelanja/ 10000);
        this.poin = (int) (this.poin + newPoin);
        return (int) jumlahBelanja;
    }
    else {
        if (jumlahBelanja >= 500000 && jumlahBelanja <= 1000000) {
            diskon = jumlahBelanja * 0.99;
            int newPoin = (int) (diskon/ 10000);
            this.poin = (int) (this.poin + newPoin);

        }
        else if (jumlahBelanja >= 1000000 && jumlahBelanja <= 7000000) {
            diskon = jumlahBelanja * 0.98;
            int newPoin = (int) (diskon/ 10000);
            this.poin = (int) (this.poin + newPoin);
        }
        else {
            diskon = jumlahBelanja * 0.97;
            int newPoin = (int) (diskon/ 10000);
            this.poin = (int) (this.poin + newPoin);
        }
    }
    return (int) diskon;
};

/*----- */
----- */

@Override
public Integer redeemPoin(Integer hargaSouvenir){
    return this.poin = this.poin - hargaSouvenir;
}

@Override
public Integer getPoin(){
    return this.poin;
}

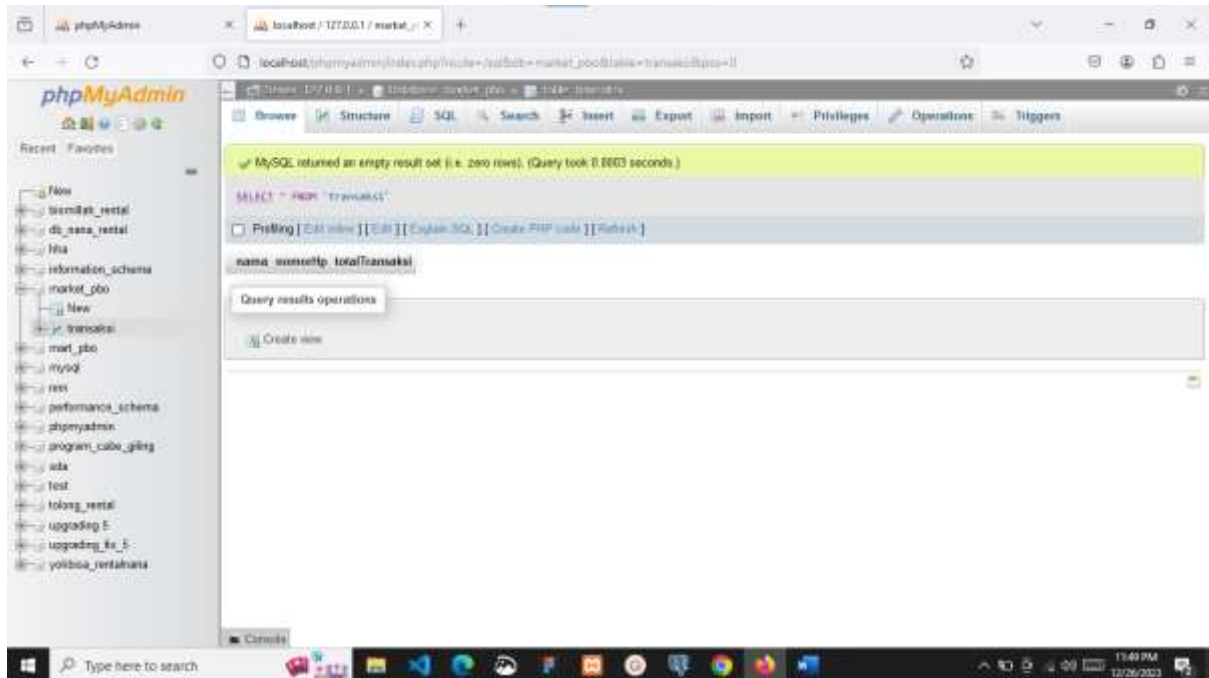
@Override
protected Integer getBonusPoin(Integer totalBayar){
    Integer poin = (int) (totalBayar / 10000);
    return poin;
}

}

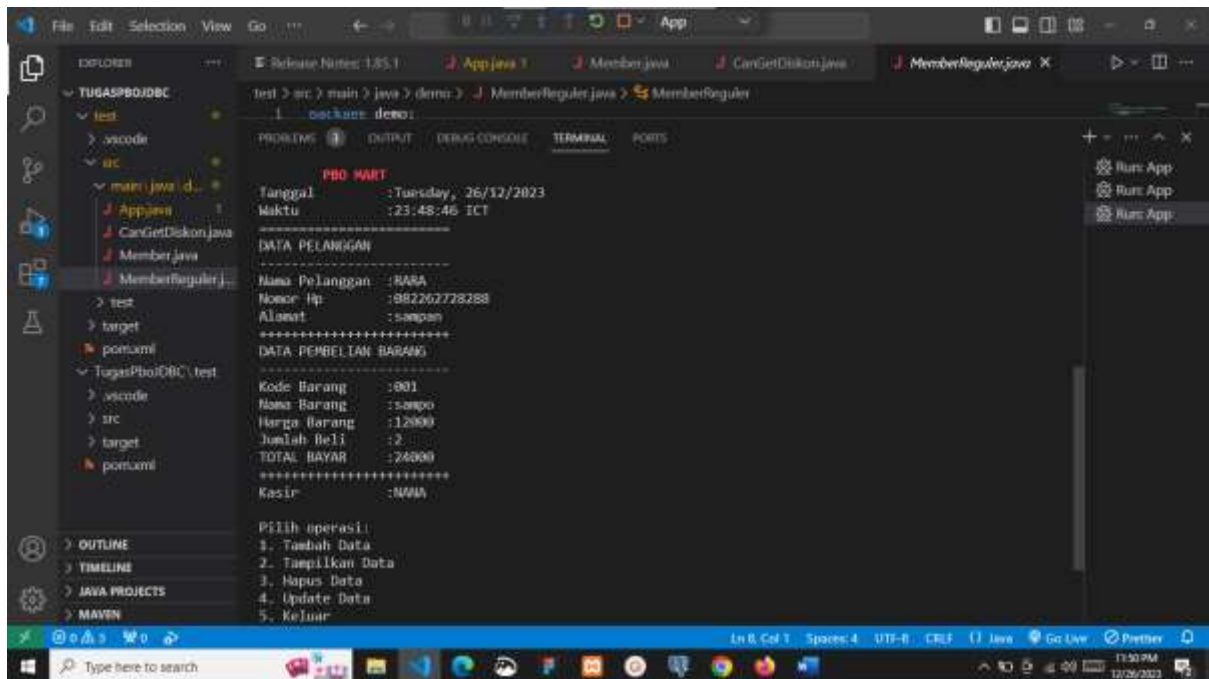
```

## B. Output program

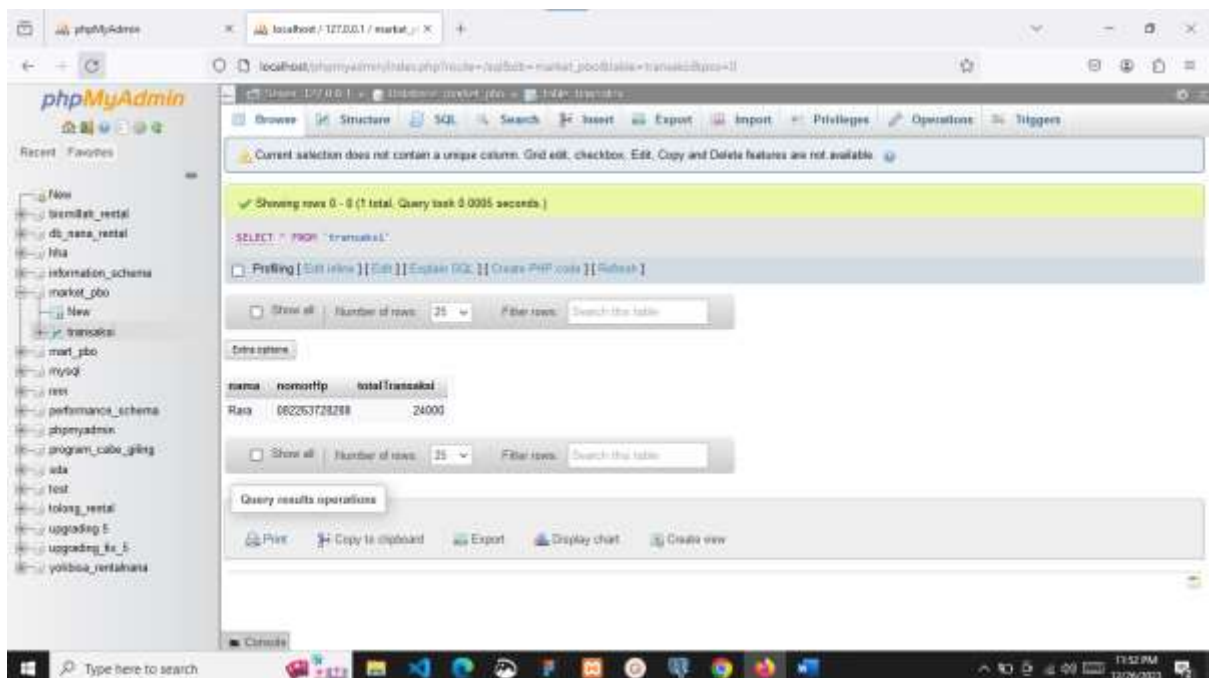
1. saat di run akan melakukan seperti sebelumnya yaitu login dan mengisi data transaksi pelanggan namun di awal akan terlihat bahwa koneksi ke database berhasil, namun database saat ini masih kosong



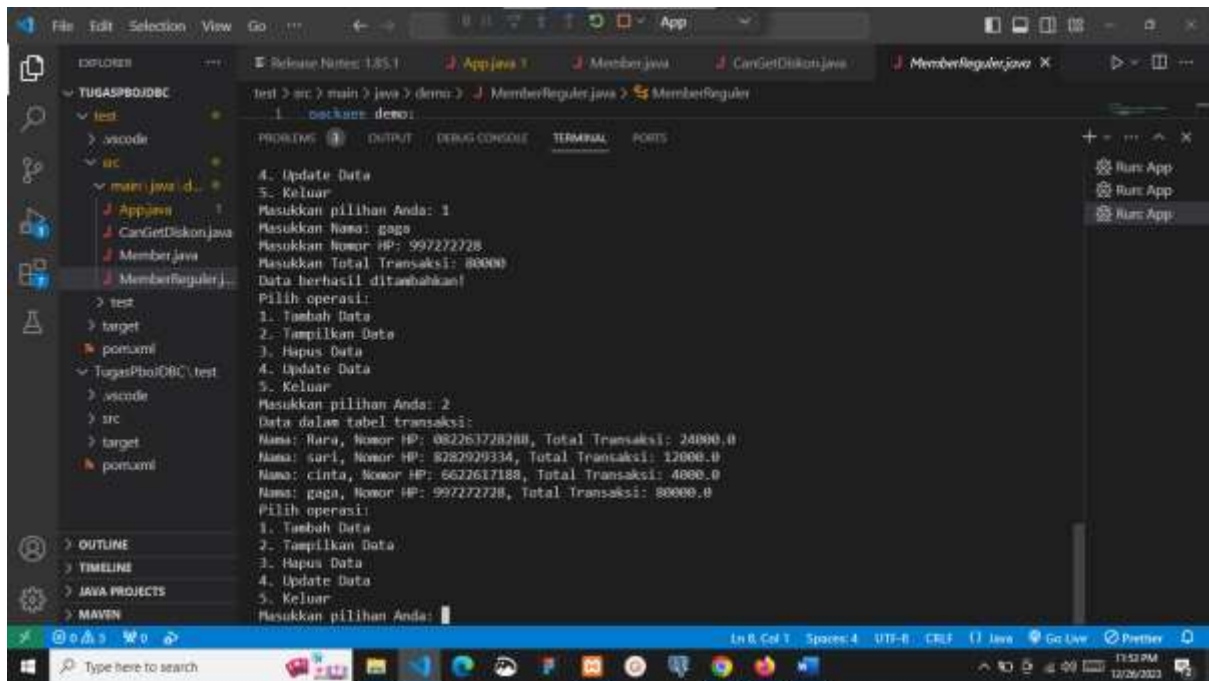
2. Akan terdapat 4 pilihan untuk JDBC yang akan muncul setelah tampil Struk transaksi, terlebih dahulu kita akan menambahkan data pelanggan sebelumnya dengan menginputkan nama, no hp dan total transaksi sesuai Struk transaksi



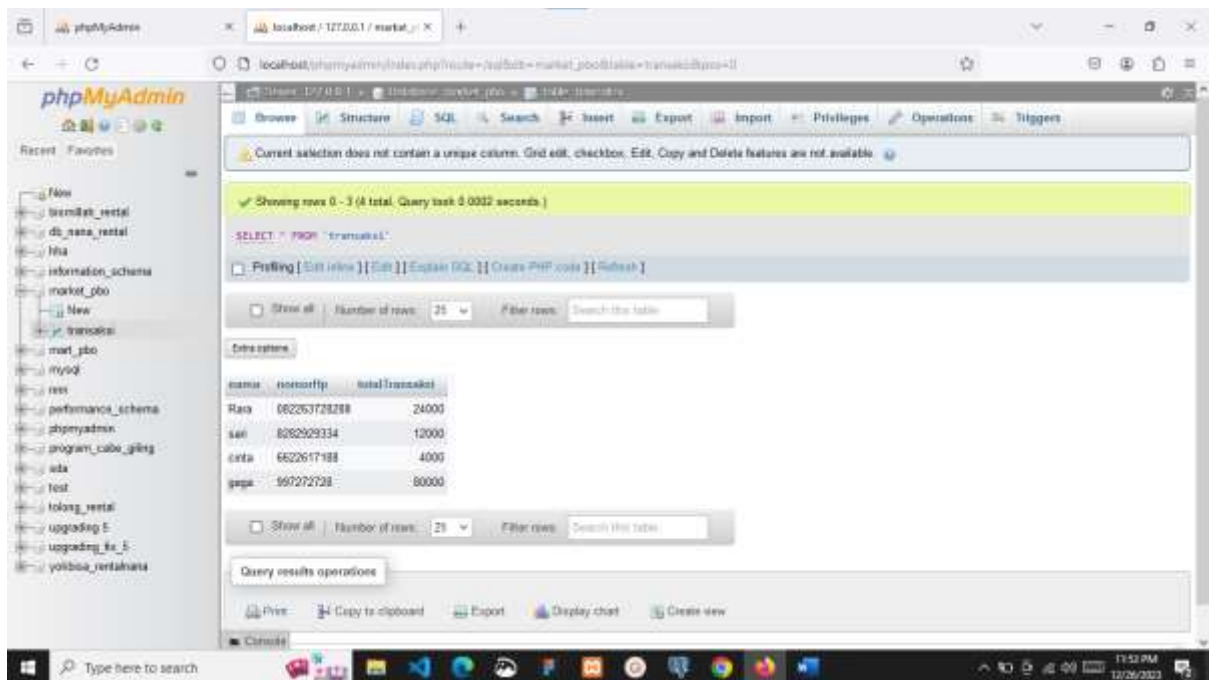
Saat data berhasil ditambahkan akan muncul pemberitahuan “Data berhasil ditambahkan !” dan akan muncul kembali menu sebelumnya, disini kita juga dapat memunculkan data yang telah ditambahkan, serta data yang ditambahkan akan tersimpan di database



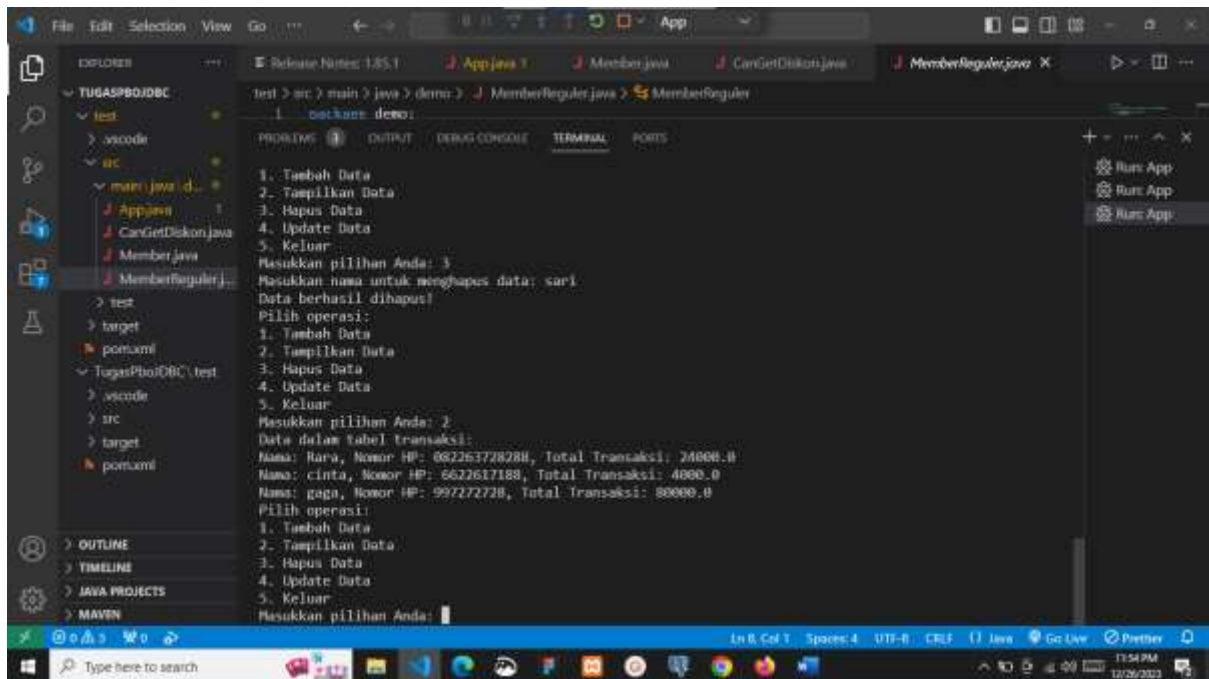
3. kita juga bisa menambahkan data selain dari yang transaksi



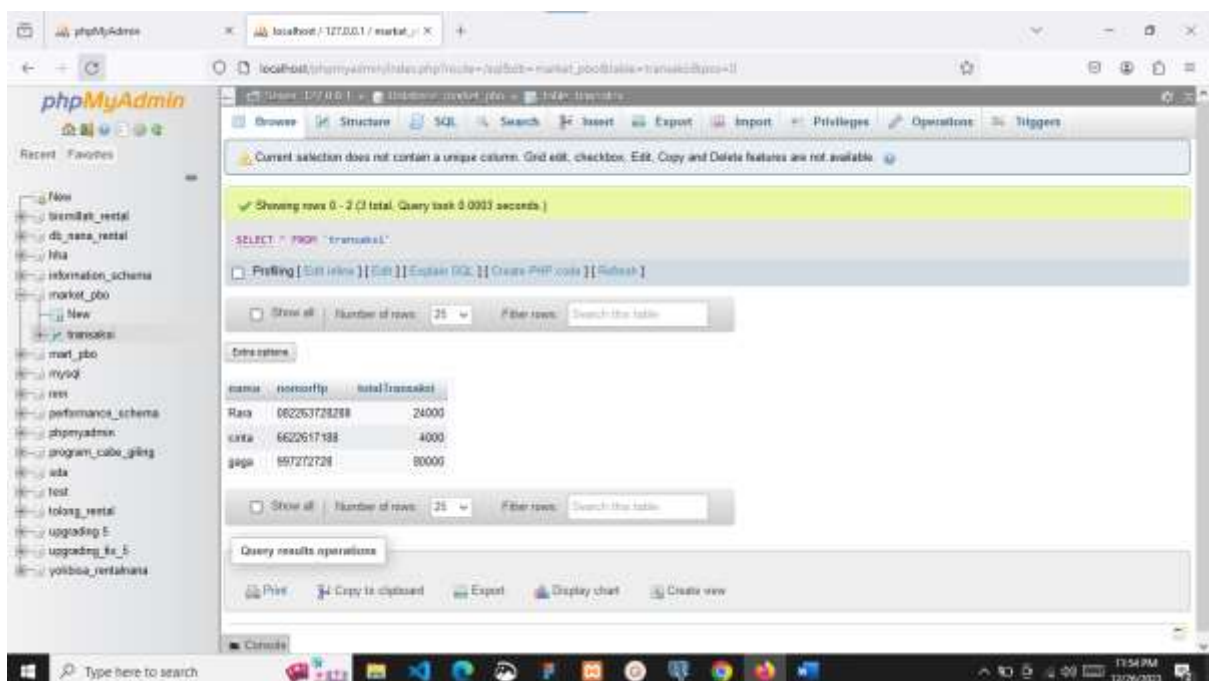
Data yang kita tambahkan akan dtambahkan juga ke database seperti yang terlihat berikut



4. Kita juga bisa menghapus data yang sudah kita tambahkan

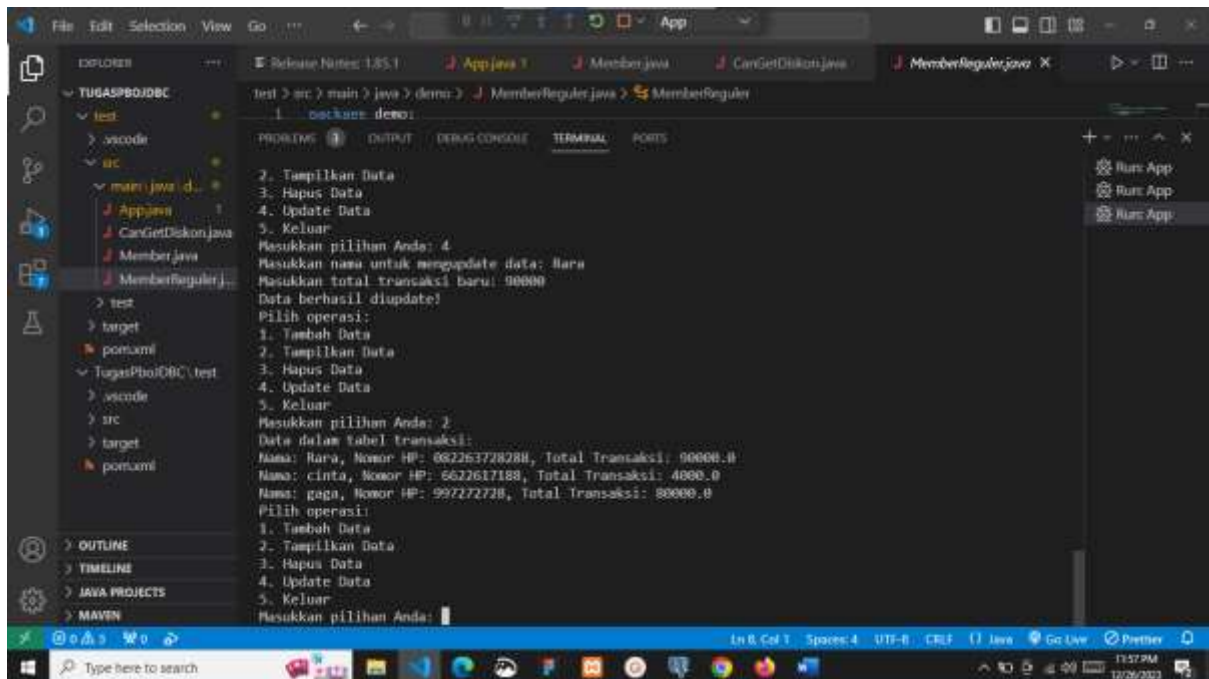


Disini kita menghapus data sari, sebelumnya ada 4 data dan setelah dihapus tersisa 3 data, di database juga akan mengikuti



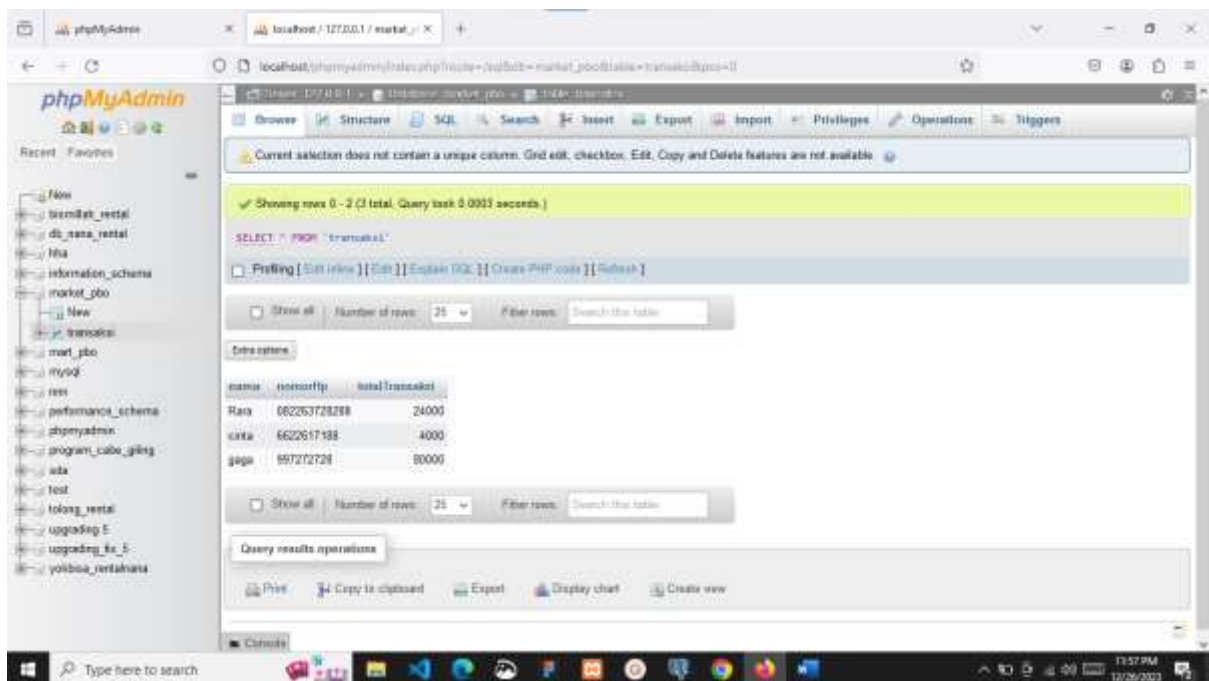
5. jika nama yang kita masukan untuk dihapus tidak ada akan muncul pesan “Data tidak ditemukan atau gagal dihapus.” Dicontoh ini kita menghapus data sari yang memang tidak ada sehingga muncul pesan tersebut

6. Disini juga bisa mengupdate data



```
test > src > main > java > demo > MemberReguler.java > MemberReguler
1. back: demo:
2. Tampilkan Data
3. Hapus Data
4. Update Data
5. Keluar
Masukkan pilihan Anda: 4
Masukkan nama untuk mengupdate data: Rara
Masukkan total transaksi baru: 90000
Data berhasil diupdate!
Pilih operasi:
1. Tambah Data
2. Tampilkan Data
3. Hapus Data
4. Update Data
5. Keluar
Masukkan pilihan Anda: 2
Data dalam tabel transaksi:
Nama: Rara, Nomor HP: 082263728288, Total Transaksi: 90000.0
Nama: cinta, Nomor HP: 6622617188, Total Transaksi: 4000.0
Nama: gaga, Nomor HP: 997272728, Total Transaksi: 80000.0
Pilih operasi:
1. Tambah Data
2. Tampilkan Data
3. Hapus Data
4. Update Data
5. Keluar
Masukkan pilihan Anda: 
```

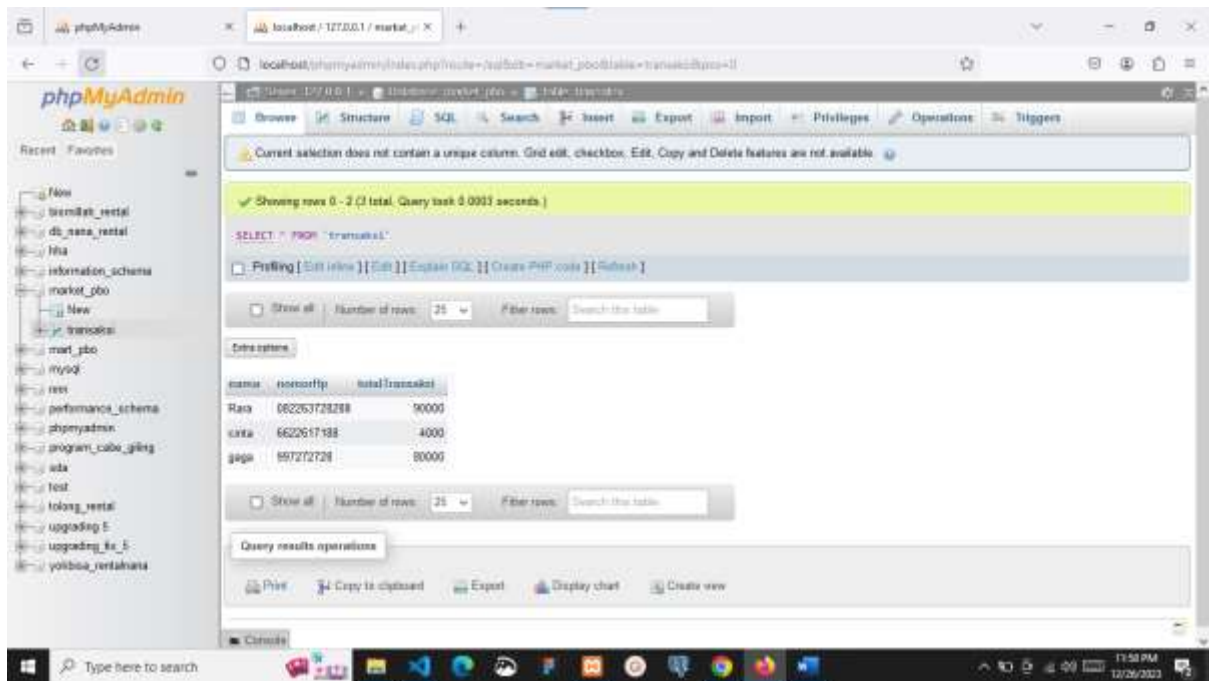
Sebelum update data total transaksi rara 24.000 setelah update menjadi 90.000



nama	nomorhp	totaltransaksi
Rara	082263728288	24000
cinta	6622617188	4000
gaga	997272728	80000

Setelah update





Disini kita juga bisa Menutup programnya dengan memilih keluar dan muncul program selesai.

