Assignment 2:

User Assignment-2

Problem Statement:

1. To access the postgre mysql, create a docker image with postgre mysql and run it to perform the database operations.

2. In the same program read the data which is inserted in the above from the postgre mysql and display on the console.

Source Code:

Assign2.java

```
import org.hibernate.Session;

public class assign2 {
  private static SessionFactory factory;

  public static void main(String args[]) throws Exception {
        setUp();
        getCustomer();
        URL file_path = Main.class.getClassLoader().getResource("customers.json");
        JSONProcessor jsonProcessor = new JSONProcessor (file_path.getPath(0);
        List<Customer> customer = jsonProcessor.parseFile();
        customer.forEach(Main::addCustomer);

  private static void setUp() {
        factory - new Configuration()
                .addAnnotatedClass (Customer.class)
                 .configure()
                 .buildsessionFactory();

  private static Integer addCustomer(Customer customer) {
    Session session = factory.openSession();
    Transaction tx = session.beginTransaction();
    Integer customerId = (Integer) session.save(customer);
    tx.commit();
    return customerId;
  }

  private static List<Customer> getCustomer () {
        Session session - factory.openSession();
        Transaction tx = session.beginTransaction();
        List<Customer> customer = session.createQuery("FROM Customer").list();
        System.out.println();
        return null;
}
}
```

Customer.java

```
import javax.persistence.*;

@Entity
@Table(name = "colibri.customer")
public class Car {
  eColumn(name = "cusId")

  private String cusID;

  eColumn (name = "cusName")

  private String cusName;
  @Column(name = "city")
  private String city;
  @Column(name - "pin")
  private int pin;

  eld
    eGeneratedValue(strategy = GenerationType. IDENTITY)

  @column(name = "id")
  private int id;

  public Customer (String cusID, String cusName, String city,int pin) {
        this.cusID = cusID;
        this.cusName = cusName;
        this.city = city;
       this.pin = pin;
   }

  public Customer()
    {}

  public int getPin() {
    return pin;
  }

  public String getCity() {
    return city;
  }

  public String getcusName() {
    return cusName;
  }

  public String getcusID() {
    return cusID;
  }

  public String getCity() {
    return city;
  }
```

```
  public String getcusName() {
    return cusName;
  }

  public String getcusID() {
    return cusID;
  }

  public String toCSVString() {
    return cusID + ", " + cusName + "," + city + "," + pin;
  }

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public void setcusID(String cusID) {
    this.cusID = cusID;
  }

  public void setcusName(String cusName) {
    this.cusName = cusName;
  }

  public void setcity(String city) {
    this.city = city;
  }

  public void setpin(int pin) {
    this.pin = pin;
  }
}
```

Customers.json

```
{
  "customers": [
    {
    "custId": "101",
    "custName": "Akshit",
    "city": "Bangalore",
    "pin": 100001
  },
  {
    "custId": "102",
    "custName": "Amisha Sahu",
    "city": "Delhi",
    "pin": 100002
```

```
  },
  {
    "custId": "102",
    "custName": "Kamal",
    "city": "Mumbai",
    "pin": 100003
  }
]
}
```

JsonProcessor.java

```
import org.json.simple.JSONArray;

public class JSONProcessor {
  private final String targetFilePath;

  JSONProcessor(String targetFilePath) {
    this.targetFilePath = targetFilePath;
  }

  public List<Customer> parseFile() throws IOException, ParseException {
    JSONParser parser = new JSONParser();
    JSONObject json = (JSONObject) parser.parse(new FileReader(targetFilePath));
    JSONArray customers = (JSONArray) json.get("customers");
    List<JSONObject> customerList = (List<JSONObject>)
customers.stream().collect(Collectors.tolist());
    return customerList.stream()
        .map(x -> new Customer((String) x.get("cusId"), (String) x.get("cusName"),
(String) x.get("city"),
            (Double) x.get("pin")))
        .collect(Collectors.tolist());
  }
}
```