Here’s the implementation in Java using Test-Driven Development (TDD). Below is the breakdown of the solution:

Steps to Implement

1. Define the Data Model: Represent customers in a Java class.

2. Create Unit Tests: Use JUnit for testing the implementation.

3. ETL Process Implementation: Create Java methods to:

Parse the input file.

Filter and transform the data.

Distribute data into country-specific tables.

Implementation

1. Define the Data Model

```
public class Customer {
    private String customerName;
    private String customerId;
    private String openDate;
    private String lastConsultedDate;
    private String vaccinationId;
    private String doctorName;
    private String state;
    private String country;
    private String dob;
    private char isActive;
    private int age;
    private int daysSinceLastConsulted;

    // Constructors, Getters, and Setters
```

```
    public Customer(String customerName, String customerId, String openDate, String
lastConsultedDate,
                    String vaccinationId, String doctorName, String state, String country,
                    String dob, char isActive) {
        this.customerName = customerName;
        this.customerId = customerId;
        this.openDate = openDate;
        this.lastConsultedDate = lastConsultedDate;
        this.vaccinationId = vaccinationId;
        this.doctorName = doctorName;
        this.state = state;
        this.country = country;
        this.dob = dob;
        this.isActive = isActive;
    }

    public void calculateDerivedFields() {
        this.age = calculateAge(this.dob);
        this.daysSinceLastConsulted = calculateDaysSinceLastConsulted(this.lastConsultedDate);
    }

    private int calculateAge(String dob) {
        // Assuming dob is in format YYYYMMDD
        int birthYear = Integer.parseInt(dob.substring(0, 4));
        int currentYear = java.time.LocalDate.now().getYear();
        return currentYear - birthYear;
    }

    private int calculateDaysSinceLastConsulted(String lastConsultedDate) {
        java.time.LocalDate lastConsulted = java.time.LocalDate.parse(lastConsultedDate,
java.time.format.DateTimeFormatter.ofPattern("yyyyMMdd"));
        java.time.LocalDate today = java.time.LocalDate.now();
        return (int) java.time.temporal.ChronoUnit.DAYS.between(lastConsulted, today);
    }

    // Override toString() for easy debugging
    @Override
    public String toString() {
        return String.format("Customer{name='%s', id='%s', country='%s', age=%d,
daysSinceLastConsulted=%d}",
                             customerName, customerId, country, age, daysSinceLastConsulted);
    }
}
```

---

2. Create Unit Tests

Using JUnit, test each functionality.

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class CustomerTest {

    @Test
    public void testCalculateAge() {
        Customer customer = new Customer("Alex", "123457", "20101012", "20121013",
                                         "MVD", "Paul", "SA", "USA", "19870603", 'A');
        customer.calculateDerivedFields();
        assertEquals(37, customer.getAge(), "Age calculation is incorrect");
    }

    @Test
    public void testCalculateDaysSinceLastConsulted() {
        Customer customer = new Customer("Alex", "123457", "20101012", "20231101",
                                         "MVD", "Paul", "SA", "USA", "19870603", 'A');
        customer.calculateDerivedFields();
        assertTrue(customer.getDaysSinceLastConsulted() > 30, "Days since last consulted
calculation is incorrect");
    }

    @Test
    public void testCountryValidation() {
        Customer customer = new Customer("John", "123458", "20101012", "20121013",
                                         "MVD", "Paul", "TN", "IND", "19870603", 'A');
        assertEquals("IND", customer.getCountry(), "Country validation failed");
    }
}
```

---

3. Implement ETL Process

```
import java.io.*;
import java.util.*;
```

```
import java.util.stream.Collectors;

public class ETLProcess {

    public List<Customer> parseInputFile(String filePath) throws IOException {
        List<Customer> customers = new ArrayList<>();
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                if (line.startsWith("|D|")) {
                    String[] parts = line.substring(3).split("\\|");
                    Customer customer = new Customer(parts[0], parts[1], parts[2], parts[3], parts[4],
                                                     parts[5], parts[6], parts[7], parts[8], parts[9].charAt(0));
                    customer.calculateDerivedFields();
                    customers.add(customer);
                }
            }
        }
        return customers;
    }

    public Map<String, List<Customer>> distributeByCountry(List<Customer> customers) {
        return customers.stream().collect(Collectors.groupingBy(Customer::getCountry));
    }

    public void saveToCountryTables(Map<String, List<Customer>> countryData) {
        for (Map.Entry<String, List<Customer>> entry : countryData.entrySet()) {
            String country = entry.getKey();
            List<Customer> customers = entry.getValue();
            saveToTable(country, customers);
        }
    }

    private void saveToTable(String country, List<Customer> customers) {
        System.out.println("Saving data for country: " + country);
        for (Customer customer : customers) {
            if (customer.getDaysSinceLastConsulted() > 30) {
                System.out.println(customer);
            }
        }
    }

    public static void main(String[] args) throws IOException {
        ETLProcess etl = new ETLProcess();
```

```
        List<Customer> customers = etl.parseInputFile("CustomerData.txt");
        Map<String, List<Customer>> countryData = etl.distributeByCountry(customers);
        etl.saveToCountryTables(countryData);
    }
}
```

---

4. Explanation of Code

1. Customer Class:

Represents customer data.

Includes methods to calculate derived fields (age, days since last consulted).

2. Unit Tests:

Validate age calculation, days since last consulted, and country filtering.

3. ETLProcess:

parseInputFile: Reads the file and converts it to Customer objects.

distributeByCountry: Groups customers by country.

saveToCountryTables: Filters customers based on business rules and saves to respective "tables" (simulated as console output).

This setup ensures the program is testable, modular, and scalable, adhering to TDD principles.