gdpr\_status.jsp :

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>GDPR Status</title>

<style type="text/css">

.center {

margin: auto;

width: 30%;

border: 3px solid #73AD21;

padding: 10px;

}

</style>

</head>

<body>

<br><br><br><br>

<h1> <center> GDPR Status Page</h1>

<div class="center">

<form action="http://localhost:8080/GDPR\_Status/NonUSMemberPortalServlet" method="POST">

<input type="checkbox" name="appname" value="International Member Portal (Non-US Expat)"> International Member Portal (Non-US Expat) <br>

<input type="checkbox" name="appname" value="International Member Portal (US Expat)"> International Member Portal (US Expat) <br>

<input type="checkbox" name="appname" value="International Mobile Assistant- Mobile"> International Mobile Assistant- Mobile <br>

<input type="checkbox" name="appname" value="International Online Enrollment Portal(Non-US Expat)"> International Online Enrollment Portal(Non-US Expat) <br>

<input type="checkbox" name="appname" value="International Online Enrollment Portal(US Expat)"> International Online Enrollment Portal(US Expat) <br>

<input type="checkbox" name="appname" value="International Broker Portal"> International Broker Portal <br>

<input type="checkbox" name="appname" value="International Provider Portal"> International Provider Portal <br>

<input type="checkbox" name="appname" value="ISJ Telesales (Individual Sales Journey)"> ISJ Telesales (Individual Sales Journey) <br>

<input type="checkbox" name="appname" value="International Provider Mobile App"> International Provider Mobile App <br>

<input type="checkbox" name="appname" value="AetnaInternational.com"> AetnaInternational.com <br>

<input type="checkbox" name="appname" value="AI Member Portal Personalization Tool"> AI Member Portal Personalization Tool <br>

<input type="checkbox" name="appname" value="Americas Trust Quoting Tool"> Americas Trust Quoting Tool <br>

<input type="checkbox" name="appname" value="GOP Admin Tool">GOP Admin Tool <br>

<input type="checkbox" name="appname" value="Passport Benefit App Tool ">Passport Benefit App Tool <br>

<input type="checkbox" name="appname" value="Other/Non-Applicable"> Other/Non-Applicable<br>

<br>

<center> <input type="submit" name="flow1" value="Submit"> </center>

</form>

<div></div>

</body>

</html>

gdprAppInput.jsp:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>GDPR Application Input Page</title>

<style type="text/css">

.center {

margin: auto;

width: 60%;

border: 3px solid #73AD21;

padding: 10px;

}

</style>

</head>

<body>

<h1> <center> GDPR Input page Page </h1>

<div class="center">

<%

String[] apptype=request.getParameterValues("appname");

for (int i = 0; i < apptype.length; i++) {

%>

<form action="http://localhost:8080/GDPR\_Status/NonUSMemberPortalServlet" method=POST>

<p> <%=(i+1)%>.) <%=apptype[i] %> </p>

<p> <% if(apptype[i].equals("International Member Portal (Non-US Expat)")) { %>

Member ID: <br>

<input type="text" name="membidNonUsExpat">

<input type="hidden" name="appType" value="International Member Portal (Non-US Expat)">

<% } %>

<p> <% if(apptype[i].equals("International Member Portal (US Expat)")) { %>

Member ID: <br>

<input type="text" name="membidUsExpat"><br>

User Name: <br>

<input type="text" name="usernameUsExpat">

<% } %>

<p> <% if(apptype[i].equals("International Mobile Assistant- Mobile")) { %>

Member ID: <br>

<input type="text" name="membidIMA"><br>

User Name: <br>

<input type="text" name="usernameIMA">

<% } %>

<p> <% if(apptype[i].equals("International Online Enrollment Portal(Non-US Expat)")) { %>

TBD

<% } %>

<p> <% if(apptype[i].equals("International Online Enrollment Portal(US Expat)")) { %>

AGBCont ID: <br>

<input type="text" name="agbcontId"><br>

Username: <br>

<input type="text" name="usernameabgcont">

<% } %>

<p> <% if(apptype[i].equals("International Broker Portal")) { %>

Broker registration Key:<br>

<input type="text" name="brregkey"><br>

User Name: <br>

<input type="text" name="usernameBrokerPortal">

<% } %>

<p> <% if(apptype[i].equals("International Provider Portal")) { %>

Provider TIN: <br>

<input type="text" name="prtin"><br>

Provider User Name: <br>

<input type="text" name="prusername">

<% } %>

<p> <% if(apptype[i].equals("ISJ Telesales (Individual Sales Journey)")) { %>

Salesforce: <br>

<input type="text" name="salesforce"><br>

Lead name: <br>

<input type="text" name="leadname"><br>

User Name: <br>

<input type="text" name="username">

<% } %>

<p> <% if(apptype[i].equals("International Provider Mobile App")) { %>

Provider TIN: <br>

<input type="text" name="prtinapp"><br>

Provider User Name: <br>

<input type="text" name="prusernameapp">

<% } %>

<p> <% if(apptype[i].equals("AetnaInternational.com")) { %>

N/A

<% } %>

<p> <% if(apptype[i].equals("AI Member Portal Personalization Tool")) { %>

Member ID: <br>

<input type="text" name="membidAIMemberPortal"><br>

<% } %>

<p> <% if(apptype[i].equals("Americas Trust Quoting Tool")) { %>

TBD:

<% } %>

<p> <% if(apptype[i].equals("GOP Admin Tool")) { %>

Member ID: <br>

<input type="text" name="membidGOP"><br>

<% } %>

<p> <% if(apptype[i].equals("Passport Benefit App Tool")) { %>

TBD

<% } %>

<p> <% if(apptype[i].equals("Other/Non-Applicable")) { %>

N/A

<% } %>

<%

}

%>

<br><center> <input type="submit" name="flow2" value="Submit"> </center>

</form>

</div>

</html>

gdprStatusServlet :

package com.gdprstatus.servlet;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.Map;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import com.gdprstatus.USMemberPortal.dao.USMemberPortalDAO;

import com.gdprstatus.brokerPortal.DAO.BrokerPortalDAO;

import com.gdprstatus.employerportalUS.dao.EmployerPortalUsDAO;

import com.gdprstatus.excel.toExcelExport;

import com.gdprstatus.imaMemberPortal.dao.IMAMemberPortalDAO;

import com.gdprstatus.individualSalesJourney.dao.ISJportalDAO;

import com.gdprstatus.internationalproviderportal.dao.InternationalProviderPortalDAO;

import com.gdprstatus.nonUSMemberPortal.dao.NonUSMemberPortalDAO;

import com.gdprstatus.nonUSMemberPortal.pojo.GdprStatusJPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberEnrollmentPOJO;

@WebServlet("/NonUSMemberPortalServlet")

public class GPDRStatusServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public GPDRStatusServlet() {

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

System.out.println("doPOSt in");

GdprStatusJPOJO gdprStatusPOJO = new GdprStatusJPOJO();

if (request.getParameter("flow1") != null) {

String appType[]=request.getParameterValues("appname");

for (String string : appType) {

System.out.println("Flow1 + "+string);

}

// Forwarded to get the input parameters from user

RequestDispatcher rd = request.getRequestDispatcher("GDPRAppInput.jsp");

rd.forward(request, response);

}

if (request.getParameter("flow2") != null) {

System.out.println("Flow two in");

String appType = request.getParameter("appType");

// getting inpur parameters from user and setting it to POJO object

String memberIdNonUsExpat = request.getParameter("membidNonUsExpat");

String memberIdUsExpat = request.getParameter("membidUsExpat");

String userNameUsExpat = request.getParameter("usernameUsExpat");

String membIdIMA = request.getParameter("membidIMA");

String userNameIMA = request.getParameter("usernameIMA");

String agbContId = request.getParameter("agbcontId");

String userNameAbgCont = request.getParameter("usernameabgcont");

String brRegkey = request.getParameter("brregkey");

String userNameBrokerPortal = request.getParameter("usernameBrokerPortal");

String prTin = request.getParameter("prtin");

String prUserName = request.getParameter("prusername");

String salesForce = request.getParameter("salesforce");

String leadName = request.getParameter("leadname");

String userName = request.getParameter("username");

String prTinApp = request.getParameter("prtinapp");

String prUserNameApp = request.getParameter("prusernameapp");

String membIdAIMemberPortal = request.getParameter("membidAIMemberPortal");

String membIdGOP = request.getParameter("membidGOP");

gdprStatusPOJO.setAppType(appType);

gdprStatusPOJO.setMembidNonUsExpat(memberIdNonUsExpat);

gdprStatusPOJO.setMembidUsExpat(memberIdUsExpat);

gdprStatusPOJO.setUsernameUsExpat(userNameUsExpat);

gdprStatusPOJO.setMembidIMA(membIdIMA);

gdprStatusPOJO.setUsernameIMA(userNameIMA);

gdprStatusPOJO.setAgbcontId(agbContId);

gdprStatusPOJO.setUsernameabgcont(userNameAbgCont);

gdprStatusPOJO.setBrregkey(brRegkey);

gdprStatusPOJO.setUsernameBrokerPortal(userNameBrokerPortal);

gdprStatusPOJO.setPrtin(prTin);

gdprStatusPOJO.setPrusername(prUserName);

gdprStatusPOJO.setSalesforce(salesForce);

gdprStatusPOJO.setLeadname(leadName);

gdprStatusPOJO.setUsername(userName);

gdprStatusPOJO.setPrtinapp(prTinApp);

gdprStatusPOJO.setPrusernameapp(prUserNameApp);

gdprStatusPOJO.setMembidAIMemberPortal(membIdAIMemberPortal);

gdprStatusPOJO.setMembidGOP(membIdGOP);

request.setAttribute("gdprStatusPOJO", gdprStatusPOJO);

System.out.println("befpre hashmap");

LinkedHashMap<String, ArrayList<?>> nonUsMap = new LinkedHashMap<>();

LinkedHashMap<String, ArrayList<?>> imaMap = new LinkedHashMap<>();

LinkedHashMap<String, ArrayList<?>> brokerMap = new LinkedHashMap<>();

LinkedHashMap<String, ArrayList<?>> ipProviderMap = new LinkedHashMap<>();

// getting details from different DBs based on user input

if(memberIdNonUsExpat!=null) {

nonUsMap = NonUSMemberPortalDAO.fromDB(gdprStatusPOJO);

request.setAttribute("nonUsMap", nonUsMap);

}

if(membIdIMA!=null) {

//IMA queries same as nonUSMember Portal. So we are calling that method.

imaMap = IMAMemberPortalDAO.fromDB(gdprStatusPOJO);

request.setAttribute("IMAQueries", imaMap);

}

if(brRegkey!=null && userNameBrokerPortal!=null) {

brokerMap = BrokerPortalDAO.fromDB(gdprStatusPOJO);

request.setAttribute("BrokerMap", brokerMap);

}

if(prTin!=null) {

ipProviderMap = InternationalProviderPortalDAO.fromDB(gdprStatusPOJO);

request.setAttribute("IPProviderMap", ipProviderMap);

}

/\*

LinkedHashMap<String, ArrayList<?>> isjMap = ISJportalDAO.fromDB(gdprStatusPOJO);

request.setAttribute("ISJMap", isjMap);

LinkedHashMap<String, ArrayList<?>> usMap = USMemberPortalDAO.fromDB(gdprStatusPOJO);

request.setAttribute("usMap", usMap);

LinkedHashMap<String, ArrayList<?>> employerMap = EmployerPortalUsDAO.fromDB(gdprStatusPOJO);

request.setAttribute("EmployerMap", employerMap);

\*/

System.out.println("before toExcel");

toExcelExport.toExcelReport(ipProviderMap);

System.out.println("After hashmap");

// Forwarding the response to result page to show the tables

RequestDispatcher rd = request.getRequestDispatcher("success.jsp");

rd.forward(request, response);

}

}

}

NonUSMEmberPortalDAO :

package com.gdprstatus.nonUSMemberPortal.dao;

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.Properties;

import com.gdprstatus.nonUSMemberPortal.pojo.GdprStatusJPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberAdressInformationDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberBrokerConsentDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberClaimDetailsBothOnlineOfflineclaimsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberClaimEOBDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberDependentInformationDetailsConsentDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberDependentInformationDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberEOBEmailNotificationDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberEOBMobileSMSDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberEnrollmentPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberOnlineClaimAuditDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberOnlineClaimSubmissionDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberPolicyGroupInformationDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberPolicyInformationDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberRREBankInformationDetailsPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberRegistrationPOJO;

import com.gdprstatus.nonUSMemberPortal.pojo.MemberSecurityQuestionAnswerDetailsPOJO;

public class NonUSMemberPortalDAO {

public static LinkedHashMap<String, ArrayList<?>> fromDB(GdprStatusJPOJO gdprStatusPOJO) throws IOException {

LinkedHashMap<String, ArrayList<?>> hmap = new LinkedHashMap<>();

Connection con = null;

try {

FileInputStream fis=new FileInputStream("C:\\Users\\n081070\\Desktop\\GPDR\\workspace\_1\\GDPR Status\\AGBMBP-dao.PROD.properties");

Properties p=new Properties ();

p.load (fis);

String conDriver= (String) p.get ("MS\_SQL2005\_ETLA.connection.driver\_class");

String conURL= (String) p.get ("MS\_SQL2005\_ETLA.connection.url");

String conUsername= (String) p.get ("MS\_SQL2005\_ETLA.connection.username");

String conPassword= (String) p.get ("MS\_SQL2005\_ETLA.connection.password");

String decryptedPassword="G$C1E$0E";

Class.forName(conDriver);

try {

con = DriverManager.getConnection(conURL,conUsername,decryptedPassword);

if(con!=null) {

System.out.println("After creating connection 111");

Statement st = con.createStatement();

DateFormat df = new SimpleDateFormat("MM/dd/yyyy");

// For International Member Portal (Non-US Expat) Portal

// Retrieving Table 1

BufferedReader br = null;

String strLine = "";

String query="";

int i = 1;

try {

br = new BufferedReader(new FileReader("C:\\Users\\n081070\\Desktop\\GPDR\\workspace\_1\\GDPR Status\\NON US MEMBER PORTAL.sql"));

while ((strLine = br.readLine()) != null) {

String replace = strLine.replace("@MEMBER\_ID","'"+ gdprStatusPOJO.getMembidNonUsExpat()+"'");

String parts[] = replace.split(";");

for (String string : parts) {

//.contains method has to come here

if((string.startsWith("--") && string.endsWith("--")) || string.contains("DECLARE") || string.contains("SET") || string.contains("OPEN") || string.contains("FETCH") || string.contains("CLOSE") || string.contains("DEALLOCATE") || string.contains("/\*") || string.contains("\*/") || string.contains("REGUALTION") || string.contains("Prepared") || string.contains("FileName") || (string.equals("")) )

continue;

query+=string;

if(string.endsWith(")")) {

System.out.println("Query to execute = "+query);

ResultSet rs = st.executeQuery(query);

if (i == 1) {

if (rs != null) {

System.out.println("Inside 1st table");

ArrayList<MemberEnrollmentPOJO> memberEnrollmentList = memberEnrollmentList(rs);

hmap.put("MemberEnrollmentTable", memberEnrollmentList);

}

}

if(i==2) {

if (rs != null) {

System.out.println("Inside 2nd table");

ArrayList<MemberRegistrationPOJO> memberRegistrationList = memberRegistrationList(rs);

hmap.put("MemberRegistrationDetailsTable", memberRegistrationList);

}

}

if(i==3) {

if (rs != null) {

System.out.println("Inside 3rd table");

ArrayList<MemberAdressInformationDetailsPOJO> memberAdressInformationDetailsList = memberAdressInformationDetailsList(rs);

hmap.put("MemberAdressInformationDetailsTable", memberAdressInformationDetailsList);

}

}

if(i==4) {

if (rs != null) {

System.out.println("Inside 4th table");

ArrayList<MemberSecurityQuestionAnswerDetailsPOJO> memberSecurityQuestionAnswerDetailsList = memberSecurityQuestionAnswerDetailsList(rs);

hmap.put("MemberSecurityQuestionAnswerDetailsTable", memberSecurityQuestionAnswerDetailsList);

}

}

if(i==5) {

if (rs != null) {

System.out.println("Inside 5th table");

ArrayList<MemberPolicyInformationDetailsPOJO> memberPolicyInformationDetailsList = memberPolicyInformationDetailsList(rs);

hmap.put("MemberPolicyInformationDetailsTable", memberPolicyInformationDetailsList);

}

}

if(i==6) {

if (rs != null) {

System.out.println("Inside 6th table");

ArrayList<MemberPolicyGroupInformationDetailsPOJO> MemberPolicyGroupInformationDetailsList = memberPolicyGroupInformationDetailsList(rs);

hmap.put("MemberPolicyGroupInformationDetailsTable", MemberPolicyGroupInformationDetailsList);

}

}

if(i==7) {

if (rs != null) {

System.out.println("Inside 7th table");

ArrayList<MemberDependentInformationDetailsPOJO> memberDependentInformationDetailsList =memberDependentInformationDetailsList(rs);

hmap.put("MemberDependentInformationDetailsTable", memberDependentInformationDetailsList);

}

}

if(i==8) {

if (rs != null) {

System.out.println("Inside 8th table");

ArrayList<MemberDependentInformationDetailsConsentDetailsPOJO> memberDependentInformationDetailsConsentDetailsList = memberDependentInformationDetailsConsentDetailsList(rs);

hmap.put("MemberDependentInformationDetailsConsentDetailsTable", memberDependentInformationDetailsConsentDetailsList);

}

}

if(i==9) {

if (rs != null) {

System.out.println("Inside 9th table");

ArrayList<MemberOnlineClaimSubmissionDetailsPOJO> memberOnlineClaimSubmissionDetailsList =memberOnlineClaimSubmissionDetailsList(rs);

hmap.put("MemberOnlineClaimSubmissionDetailsTable", memberOnlineClaimSubmissionDetailsList);

}

}

if(i==10) {

if (rs != null) {

System.out.println("Inside 10th table");

ArrayList<MemberOnlineClaimAuditDetailsPOJO> memberOnlineClaimAuditDetailsList = memberOnlineClaimAuditDetailsList(rs);

hmap.put("MemberOnlineClaimAuditDetailsTable", memberOnlineClaimAuditDetailsList);

}

}

if(i==11) {

if (rs != null) {

System.out.println("Inside 11 table");

ArrayList<MemberClaimDetailsBothOnlineOfflineclaimsPOJO> memberClaimDetailsBothOnlineOfflineclaimsList = memberClaimDetailsBothOnlineOfflineclaimsList(rs);

hmap.put("MemberClaimDetailsBothOnlineOfflineclaimsTable", memberClaimDetailsBothOnlineOfflineclaimsList);

}

}

if(i==12) {

if (rs != null) {

System.out.println("Inside 12th table");

ArrayList<MemberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO> memberClaimLineItemDetailsBothOnlineOfflineClaimsList = memberClaimLineItemDetailsBothOnlineOfflineClaimsList(rs);

hmap.put("MemberClaimLineItemDetailsBothOnlineOfflineClaimsTable", memberClaimLineItemDetailsBothOnlineOfflineClaimsList);

}

}

if(i==13) {

if (rs != null) {

System.out.println("Inside 13th table");

ArrayList<MemberRREBankInformationDetailsPOJO> memberRREBankInformationDetailsList = memberRREBankInformationDetailsList(rs);

hmap.put("MemberRREBankInformationDetailsTable", memberRREBankInformationDetailsList);

}

}

if(i==14) {

if (rs != null) {

System.out.println("Inside 14th table");

ArrayList<MemberBrokerConsentDetailsPOJO> memberBrokerConsentDetailsList = memberBrokerConsentDetailsList(rs);

hmap.put("MemberBrokerConsentDetailsTable", memberBrokerConsentDetailsList);

}

}

if(i==15) {

if (rs != null) {

System.out.println("Inside 15th table");

ArrayList<MemberClaimEOBDetailsPOJO> memberClaimEOBDetailsList = memberClaimEOBDetailsList(rs);

hmap.put("MemberClaimEOBDetailsTable", memberClaimEOBDetailsList);

}

}

if(i==16) {

if (rs != null) {

System.out.println("Inside 16th table");

ArrayList<MemberEOBEmailNotificationDetailsPOJO> memberEOBEmailNotificationDetailsList =memberEOBEmailNotificationDetailsList(rs);

hmap.put("MemberEOBEmailNotificationDetailsTable", memberEOBEmailNotificationDetailsList);

}

}

if(i==17) {

if (rs != null) {

System.out.println("Inside 17th table");

ArrayList<MemberEOBMobileSMSDetailsPOJO> memberEOBMobileSMSDetailsList = memberEOBMobileSMSDetailsList(rs);

hmap.put("MemberEOBMobileSMSDetailsTable", memberEOBMobileSMSDetailsList);

}

}

if(i==18) {

if (rs != null) {

System.out.println("Inside 18th table");

ArrayList<MemberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO> memberCertificateOfInsuranceDetailsOnlyforActisureUsersList = memberCertificateOfInsuranceDetailsOnlyforActisureUsersList(rs);

hmap.put("MemberCertificateOfInsuranceDetailsOnlyforActisureUsersTable", memberCertificateOfInsuranceDetailsOnlyforActisureUsersList);

break;

}

}

i++;

query="";

}

}

}//braces need to come here

} catch (FileNotFoundException e) {

System.err.println("Unable to find the file: fileName");

} catch (IOException e) {

System.err.println("Unable to read the file IO exception: fileName");

}

}

} catch (SQLException e) {

System.out.println("Exception here= "+e);

} finally {

if (con != null) {

try {

con.close();

} catch (SQLException ex) {

System.out.println("Exception over here= "+ex);

}

}

}

} catch (ClassNotFoundException e) {

System.out.println("CLass not found "+e);

}

return hmap;

}

public static ArrayList<MemberEnrollmentPOJO> memberEnrollmentList(ResultSet rs) throws SQLException {

//table 1

ArrayList<MemberEnrollmentPOJO> memberEnrollmentList = new ArrayList<>();

while (rs.next()) {

MemberEnrollmentPOJO memberEnrollmentPOJO = new MemberEnrollmentPOJO();

String memberID=rs.getString(1);

String title=rs.getString(2);

String firstname=rs.getString(3);

String surname=rs.getString(4);

String initials=rs.getString(5);

//String dob=df.format(rs.getTimestamp(6));

String dob=rs.getString(6);

if(dob!=null) {

String dobSplit[]=dob.split("\\s+");

memberEnrollmentPOJO.setDateOfBirth(dobSplit[0]);

}

else {

memberEnrollmentPOJO.setDateOfBirth(dob);

}

String email=rs.getString(7);

String isActive=rs.getString(8);

String instanceID=rs.getString(9);

String branchID=rs.getString(10);

String countryOfResidence =rs.getString(11);

String memberPhone=rs.getString(12);

String nationality=rs.getString(13);

String occupation=rs.getString(14);

String passportID=rs.getString(15);

// String passportExpiryDate=df.format(rs.getTimestamp(16));

String passportExpiryDate=rs.getString(16);

if(passportExpiryDate!=null) {

String passportExpiryDateSplit[]=passportExpiryDate.split("\\s+");

memberEnrollmentPOJO.setPassportExpiryDate(passportExpiryDateSplit[0]);

}

else {

memberEnrollmentPOJO.setPassportExpiryDate(passportExpiryDate);

}

memberEnrollmentPOJO.setMemberID(memberID);

memberEnrollmentPOJO.setTitle(title);

memberEnrollmentPOJO.setFirstname(firstname);

memberEnrollmentPOJO.setSurname(surname);

memberEnrollmentPOJO.setInitials(initials);

memberEnrollmentPOJO.setEmail(email);

memberEnrollmentPOJO.setIsActive(isActive);

memberEnrollmentPOJO.setInstanceID(instanceID);

memberEnrollmentPOJO.setBranchID(branchID);

memberEnrollmentPOJO.setCountryOfResidence(countryOfResidence);

memberEnrollmentPOJO.setMemberPhone(memberPhone);

memberEnrollmentPOJO.setNationality(nationality);

memberEnrollmentPOJO.setOcccupation(occupation);

memberEnrollmentPOJO.setPassportID(passportID);

System.out.println(memberEnrollmentPOJO);

memberEnrollmentList.add(memberEnrollmentPOJO);

}

return memberEnrollmentList;

}

public static ArrayList<MemberRegistrationPOJO> memberRegistrationList(ResultSet rs) throws SQLException {

//table 2

ArrayList<MemberRegistrationPOJO> memberRegistrationList=new ArrayList<>();

while (rs.next()) {

MemberRegistrationPOJO memberRegistrationPOJO = new MemberRegistrationPOJO();

String userGuideID = rs.getString(1);

String userTypeCD = rs.getString(2);

String userCreateDT = rs.getString(3);

if(userCreateDT!=null) {

String userCreateDTplit[]=userCreateDT.split("\\s+");

memberRegistrationPOJO.setUserCreateDT(userCreateDTplit[0]);

}

else {

memberRegistrationPOJO.setUserCreateDT(userCreateDT);

}

String userUpdateDT = rs.getString(4);

if(userUpdateDT!=null) {

String userUpdateDTsplit[]=userUpdateDT.split("\\s+");

memberRegistrationPOJO.setUserUpdateDT(userUpdateDTsplit[0]);

}

else {

memberRegistrationPOJO.setUserUpdateDT(userUpdateDT);

}

String userEmailTxt = rs.getString(5);

String userLstNm = rs.getString(6);

String userFrstNm = rs.getString(7);

String userBrthDt = rs.getString(8);

if(userBrthDt!=null) {

String userBrthDtsplit[]=userBrthDt.split("\\s+");

memberRegistrationPOJO.setUserBrthDt(userBrthDtsplit[0]);

}

else {

memberRegistrationPOJO.setUserBrthDt(userBrthDt);

}

String userRefNmbr = rs.getString(9);

String userCurrpwdTxt = rs.getString(10);

String userStatCd = rs.getString(11);

String userLoginTxt = rs.getString(12);

String userFlpwdattCnt = rs.getString(13);

String userFlpwdattstrtDt = rs.getString(14);

if(userFlpwdattstrtDt!=null) {

String userFlpwdattstrtDtsplit[]=userFlpwdattstrtDt.split("\\s+");

memberRegistrationPOJO.setUserFlpwdattstrtDt(userFlpwdattstrtDtsplit[0]);

}

else {

memberRegistrationPOJO.setUserFlpwdattstrtDt(userFlpwdattstrtDt);

}

String userFlchlgansCnt = rs.getString(15);

String userFlchlgansDt = rs.getString(16);

if(userFlchlgansDt!=null) {

String userFlchlgansDtsplit[]=userFlchlgansDt.split("\\s+");

memberRegistrationPOJO.setUserFlchlgansDt(userFlchlgansDtsplit[0]);

}

else {

memberRegistrationPOJO.setUserFlchlgansDt(userFlchlgansDt);

}

String userLstloginDt = rs.getString(17);

if(userLstloginDt!=null) {

String userLstloginDtsplit[]=userLstloginDt.split("\\s+");

memberRegistrationPOJO.setUserLstloginDt(userLstloginDtsplit[0]);

}

else {

memberRegistrationPOJO.setUserLstloginDt(userLstloginDt);

}

String userLstlockutDt = rs.getString(18);

if(userLstlockutDt!=null) {

String userLstlockutDtsplit[]=userLstlockutDt.split("\\s+");

memberRegistrationPOJO.setUserLstlockutDt(userLstlockutDtsplit[0]);

}

else {

memberRegistrationPOJO.setUserLstlockutDt(userLstlockutDt);

}

memberRegistrationPOJO.setUserGuideID(userGuideID);

memberRegistrationPOJO.setUserTypeCD(userTypeCD);

memberRegistrationPOJO.setUserEmailTxt(userEmailTxt);

memberRegistrationPOJO.setUserLstNm(userLstNm);

memberRegistrationPOJO.setUserFrstNm(userFrstNm);

memberRegistrationPOJO.setUserRefNmbr(userRefNmbr);

memberRegistrationPOJO.setUserCurrpwdTxt(userCurrpwdTxt);

memberRegistrationPOJO.setUserStatCd(userStatCd);

memberRegistrationPOJO.setUserLoginTxt(userLoginTxt);

memberRegistrationPOJO.setUserFlpwdattCnt(userFlpwdattCnt);

memberRegistrationPOJO.setUserFlchlgansCnt(userFlchlgansCnt);

System.out.println(memberRegistrationPOJO);

memberRegistrationList.add(memberRegistrationPOJO);

}

return memberRegistrationList;

}

public static ArrayList<MemberAdressInformationDetailsPOJO> memberAdressInformationDetailsList(ResultSet rs) throws SQLException {

//table 3

ArrayList<MemberAdressInformationDetailsPOJO> memberAdressInformationDetailsList=new ArrayList<>();

while (rs.next()) {

MemberAdressInformationDetailsPOJO memberAdressInformationDetailsPOJO =new MemberAdressInformationDetailsPOJO();

String memberAddInfoId=rs.getString(1);

String memberId=rs.getString(2);

String passportNumber=rs.getString(3);

String martialStatusId=rs.getString(4);

String martialStatusDesc=rs.getString(5);

String emirateId=rs.getString(6);

String emirateDesc=rs.getString(7);

String planSponSorTypeId=rs.getString(8);

String planSponSorDesc=rs.getString(9);

String resLocationId=rs.getString(10);

String resLocationName=rs.getString(11);

String workLocationId=rs.getString(12);

String workLocationDesc=rs.getString(13);

String salaryBandId=rs.getString(14);

String salaryBandDesc=rs.getString(15);

String incomeKey=rs.getString(16);

String incomeDesc=rs.getString(17);

String emiratesAppId=rs.getString(18);

String suId=rs.getString(19);

String sSponsoruId=rs.getString(20);

String lMemberContactKey=rs.getString(21);

String familyMemberId=rs.getString(22);

String familyMemberDesc=rs.getString(23);

String sEmail=rs.getString(24);

String steamEmail=rs.getString(25);

String originalJoinDate=rs.getString(26);

if(originalJoinDate!=null) {

String originalJoinDatesplit[]=originalJoinDate.split("\\s+");

memberAdressInformationDetailsPOJO.setOriginalJoinDate(originalJoinDatesplit[0]);

}

else {

memberAdressInformationDetailsPOJO.setOriginalJoinDate(originalJoinDate);

}

String memberFrom=rs.getString(27);

if(memberFrom!=null) {

String memberFromtsplit[]=memberFrom.split("\\s+");

memberAdressInformationDetailsPOJO.setMemberFrom(memberFromtsplit[0]);

}

else {

memberAdressInformationDetailsPOJO.setMemberFrom(memberFrom);

}

String memberCancelled=rs.getString(28);

if(memberCancelled!=null) {

String memberCancelledsplit[]=memberCancelled.split("\\s+");

memberAdressInformationDetailsPOJO.setMemberCancelled(memberCancelledsplit[0]);

}

else {

memberAdressInformationDetailsPOJO.setMemberCancelled(memberCancelled);

}

String extPolicyNumber=rs.getString(29);

String contactReference=rs.getString(30);

String contactMethodId=rs.getString(31);

String contactMethodDesc=rs.getString(32);

String employeeNumber=rs.getString(33);

String mobilePhone=rs.getString(34);

String homePhone=rs.getString(35);

memberAdressInformationDetailsPOJO.setMemberAddInfoId(memberAddInfoId);

memberAdressInformationDetailsPOJO.setMemberId(memberId);

memberAdressInformationDetailsPOJO.setPassportNumber(passportNumber);

memberAdressInformationDetailsPOJO.setMartialStatusId(martialStatusId);

memberAdressInformationDetailsPOJO.setMartialStatusDesc(martialStatusDesc);

memberAdressInformationDetailsPOJO.setEmirateId(emirateId);

memberAdressInformationDetailsPOJO.setEmirateDesc(emirateDesc);

memberAdressInformationDetailsPOJO.setPlanSponSorTypeId(planSponSorTypeId);

memberAdressInformationDetailsPOJO.setPlanSponSorDesc(planSponSorDesc);

memberAdressInformationDetailsPOJO.setResLocationId(resLocationId);

memberAdressInformationDetailsPOJO.setResLocationName(resLocationName);

memberAdressInformationDetailsPOJO.setWorkLocationId(workLocationId);

memberAdressInformationDetailsPOJO.setWorkLocationDesc(workLocationDesc);

memberAdressInformationDetailsPOJO.setSalaryBandId(salaryBandId);

memberAdressInformationDetailsPOJO.setSalaryBandDesc(salaryBandDesc);

memberAdressInformationDetailsPOJO.setIncomeKey(incomeKey);

memberAdressInformationDetailsPOJO.setIncomeDesc(incomeDesc);

memberAdressInformationDetailsPOJO.setEmiratesAppId(emiratesAppId);

memberAdressInformationDetailsPOJO.setSuId(suId);

memberAdressInformationDetailsPOJO.setsSponsoruId(sSponsoruId);

memberAdressInformationDetailsPOJO.setlMemberContactKey(lMemberContactKey);

memberAdressInformationDetailsPOJO.setFamilyMemberId(familyMemberId);

memberAdressInformationDetailsPOJO.setFamilyMemberDesc(familyMemberDesc);

memberAdressInformationDetailsPOJO.setsEmail(sEmail);

memberAdressInformationDetailsPOJO.setSteamEmail(steamEmail);

memberAdressInformationDetailsPOJO.setExtPolicyNumber(extPolicyNumber);

memberAdressInformationDetailsPOJO.setContactReference(contactReference);

memberAdressInformationDetailsPOJO.setContactMethodId(contactMethodId);

memberAdressInformationDetailsPOJO.setContactMethodDesc(contactMethodDesc);;

memberAdressInformationDetailsPOJO.setEmployeeNumber(employeeNumber);

memberAdressInformationDetailsPOJO.setMobilePhone(mobilePhone);

memberAdressInformationDetailsPOJO.setHomePhone(homePhone);

System.out.println(memberAdressInformationDetailsPOJO);

memberAdressInformationDetailsList.add(memberAdressInformationDetailsPOJO);

}

return memberAdressInformationDetailsList;

}

public static ArrayList<MemberSecurityQuestionAnswerDetailsPOJO> memberSecurityQuestionAnswerDetailsList(ResultSet rs) throws SQLException {

//table 4

ArrayList<MemberSecurityQuestionAnswerDetailsPOJO> memberSecurityQuestionAnswerDetailsList = new ArrayList<>();

while (rs.next()) {

MemberSecurityQuestionAnswerDetailsPOJO memberSecurityQuestionAnswerDetailsPOJO = new MemberSecurityQuestionAnswerDetailsPOJO();

String userGuidId=rs.getString(1);

String usrqstnId=rs.getString(2);

String usrqstnDispordNbr=rs.getString(3);

String usrqstnTxt=rs.getString(4);

String usrqstnAnswerTxt=rs.getString(5);

memberSecurityQuestionAnswerDetailsPOJO.setUserGuidId(userGuidId);

memberSecurityQuestionAnswerDetailsPOJO.setUsrqstnId(usrqstnId);

memberSecurityQuestionAnswerDetailsPOJO.setUsrqstnDispordNbr(usrqstnDispordNbr);

memberSecurityQuestionAnswerDetailsPOJO.setUsrqstnTxt(usrqstnTxt);

memberSecurityQuestionAnswerDetailsPOJO.setUsrqstnAnswerTxt(usrqstnAnswerTxt);

System.out.println(memberSecurityQuestionAnswerDetailsPOJO);

memberSecurityQuestionAnswerDetailsList.add(memberSecurityQuestionAnswerDetailsPOJO);

}

return memberSecurityQuestionAnswerDetailsList;

}

public static ArrayList<MemberPolicyInformationDetailsPOJO> memberPolicyInformationDetailsList(ResultSet rs) throws SQLException {

//table 5

ArrayList<MemberPolicyInformationDetailsPOJO> memberPolicyInformationDetailsList = new ArrayList<>();

while (rs.next()) {

MemberPolicyInformationDetailsPOJO memberPolicyInformationDetailsPOJO = new MemberPolicyInformationDetailsPOJO();

String memberId=rs.getString(1);

String policyId=rs.getString(2);

String policyNumber=rs.getString(3);

String policyStatus=rs.getString(4);

String employerId=rs.getString(5);

String employerName=rs.getString(6);

String policyHolderName=rs.getString(7);

String joinDate=rs.getString(8);

if(joinDate!=null) {

String joinDatesplit[]=joinDate.split("\\s+");

memberPolicyInformationDetailsPOJO.setJoinDate(joinDatesplit[0]);

}

else {

memberPolicyInformationDetailsPOJO.setJoinDate(joinDate);

}

String inceptionDate=rs.getString(9);

if(inceptionDate!=null) {

String inceptionDatesplit[]=inceptionDate.split("\\s+");

memberPolicyInformationDetailsPOJO.setInceptionDate(inceptionDatesplit[0]);

}

else {

memberPolicyInformationDetailsPOJO.setInceptionDate(inceptionDate);

}

String effectiveDate=rs.getString(10);

if(effectiveDate!=null) {

String effectiveDatesplit[]=effectiveDate.split("\\s+");

memberPolicyInformationDetailsPOJO.setEffectiveDate(effectiveDatesplit[0]);

}

else {

memberPolicyInformationDetailsPOJO.setEffectiveDate(effectiveDate);

}

String renewalDate=rs.getString(11);

if(renewalDate!=null) {

String renewalDatesplit[]=renewalDate.split("\\s+");

memberPolicyInformationDetailsPOJO.setRenewalDate(renewalDatesplit[0]);

}

else {

memberPolicyInformationDetailsPOJO.setRenewalDate(renewalDate);

}

String terminationDate=rs.getString(12);

if(terminationDate!=null) {

String terminationDatesplit[]=terminationDate.split("\\s+");

memberPolicyInformationDetailsPOJO.setTerminationDate(terminationDatesplit[0]);

}

else {

memberPolicyInformationDetailsPOJO.setTerminationDate(terminationDate);

}

String cancellationDate=rs.getString(13);

if(cancellationDate!=null) {

String cancellationDatesplit[]=cancellationDate.split("\\s+");

memberPolicyInformationDetailsPOJO.setCancellationDate(cancellationDatesplit[0]);

}

else {

memberPolicyInformationDetailsPOJO.setCancellationDate(cancellationDate);

}

String instanceId=rs.getString(14);

String productId=rs.getString(15);

String productName=rs.getString(16);

String productPrefix=rs.getString(17);

String originalJoinDate=rs.getString(18);

if(originalJoinDate!=null) {

String originalJoinDatesplit[]=originalJoinDate.split("\\s+");

memberPolicyInformationDetailsPOJO.setOriginalJoinDate(originalJoinDatesplit[0]);

}

else {

memberPolicyInformationDetailsPOJO.setOriginalJoinDate(originalJoinDate);

}

String policyType=rs.getString(19);

String planId=rs.getString(20);

String planName=rs.getString(21);

String frontingPartnerId=rs.getString(22);

String frontingPartnerName=rs.getString(23);

String policyFolderKey=rs.getString(24);

String policyStatusOhPlus=rs.getString(25);

String sReference=rs.getString(26);

String underWritingType=rs.getString(27);

String isMainMedicalPolicy=rs.getString(28);

String categoryId=rs.getString(29);

String categoryName=rs.getString(30);

String currencyCode=rs.getString(31);

String mobileDocInd=rs.getString(32);

String isDpiExcluded=rs.getString(33);

String actCountryOfResidence=rs.getString(34);

memberPolicyInformationDetailsPOJO.setMemberId(memberId);

memberPolicyInformationDetailsPOJO.setPolicyId(policyId);

memberPolicyInformationDetailsPOJO.setPolicyNumber(policyNumber);

memberPolicyInformationDetailsPOJO.setPolicyStatus(policyStatus);

memberPolicyInformationDetailsPOJO.setEmployerId(employerId);

memberPolicyInformationDetailsPOJO.setEmployerName(employerName);

memberPolicyInformationDetailsPOJO.setPolicyHolderName(policyHolderName);

memberPolicyInformationDetailsPOJO.setInstanceId(instanceId);

memberPolicyInformationDetailsPOJO.setProductId(productId);

memberPolicyInformationDetailsPOJO.setProductName(productName);

memberPolicyInformationDetailsPOJO.setProductPrefix(productPrefix);

memberPolicyInformationDetailsPOJO.setPolicyType(policyType);

memberPolicyInformationDetailsPOJO.setPlanId(planId);

memberPolicyInformationDetailsPOJO.setPlanName(planName);

memberPolicyInformationDetailsPOJO.setFrontingPartnerId(frontingPartnerId);

memberPolicyInformationDetailsPOJO.setFrontingPartnerName(frontingPartnerName);

memberPolicyInformationDetailsPOJO.setPolicyFolderKey(policyFolderKey);

memberPolicyInformationDetailsPOJO.setPolicyStatusOhPlus(policyStatusOhPlus);

memberPolicyInformationDetailsPOJO.setsReference(sReference);

memberPolicyInformationDetailsPOJO.setUnderWritingType(underWritingType);

memberPolicyInformationDetailsPOJO.setIsMainMedicalPolicy(isMainMedicalPolicy);

memberPolicyInformationDetailsPOJO.setCategoryId(categoryId);

memberPolicyInformationDetailsPOJO.setCategoryName(categoryName);

memberPolicyInformationDetailsPOJO.setCurrencyCode(currencyCode);

memberPolicyInformationDetailsPOJO.setMobileDocInd(mobileDocInd);

memberPolicyInformationDetailsPOJO.setIsDpiExcluded(isDpiExcluded);

memberPolicyInformationDetailsPOJO.setActCountryOfResidence(actCountryOfResidence);

System.out.println(memberPolicyInformationDetailsPOJO);

memberPolicyInformationDetailsList.add(memberPolicyInformationDetailsPOJO);

}

return memberPolicyInformationDetailsList;

}

public static ArrayList<MemberPolicyGroupInformationDetailsPOJO> memberPolicyGroupInformationDetailsList(ResultSet rs) throws SQLException {

//table 6

ArrayList<MemberPolicyGroupInformationDetailsPOJO> MemberPolicyGroupInformationDetailsList = new ArrayList<>();

while (rs.next()) {

MemberPolicyGroupInformationDetailsPOJO memberPolicyGroupInformationDetailsPOJO = new MemberPolicyGroupInformationDetailsPOJO();

String memberPolicyExtKey=rs.getString(1);

String memberID=rs.getString(2);

String policyID=rs.getString(3);

String branchID=rs.getString(4);

String webRegionDescription=rs.getString(5);

String opsCenterDescr=rs.getString(6);

String employeesTaffid=rs.getString(7);

String issuingIsuredid=rs.getString(8);

String coverageArea=rs.getString(9);

String networkProviderTier=rs.getString(10);

String correspondenceAddr=rs.getString(11);

String memberRelgflg=rs.getString(12);

String dExcess=rs.getString(13);

String sCurrency=rs.getString(14);

String sumType=rs.getString(15);

String smsTNumber=rs.getString(16);

String bsendMBrid=rs.getString(17);

String coiAddress=rs.getString(18);

String dCoinsoopMax=rs.getString(19);

String sDirectBilling=rs.getString(20);

String sRoutineDental=rs.getString(21);

String dRoutineDentalCoins=rs.getString(22);

String smarjRestOrative=rs.getString(23);

String dmajRestCoins=rs.getString(24);

String sDental=rs.getString(25);

String dDentalCOins=rs.getString(26);

String dStandardCoins=rs.getString(27);

String dCoPay=rs.getString(28);

String dOutPatientCopay=rs.getString(29);

String schronic=rs.getString(30);

String schronicTier=rs.getString(31);

String dMaternityWaitingGperiod=rs.getString(32);

String idCardDescription1=rs.getString(33);

String idCardDescription2=rs.getString(34);

String brokerId=rs.getString(35);

String brokerName=rs.getString(36);

String PlanType=rs.getString(37);

String sBrokerPhone=rs.getString(38);

String sPlanSponsorPhone=rs.getString(39);

String sExcessDeductiblefg=rs.getString(40);

String dExcessPermedCond=rs.getString(41);

String dOutpatientCoinerPClaim=rs.getString(42);

String dRoutineMajorResdental=rs.getString(43);

String tier4Dubai=rs.getString(44);

String outPatientDirectBilling=rs.getString(45);

String soptrRegChildBenefit=rs.getString(46);

String dPremiumPayment=rs.getString(47);

String lMainMedicalPolicyID=rs.getString(48);

String networkProvider=rs.getString(49);

memberPolicyGroupInformationDetailsPOJO.setMemberPolicyExtKey(memberPolicyExtKey);

memberPolicyGroupInformationDetailsPOJO.setMemberID(memberID);

memberPolicyGroupInformationDetailsPOJO.setPolicyID(policyID);

memberPolicyGroupInformationDetailsPOJO.setBranchID(branchID);

memberPolicyGroupInformationDetailsPOJO.setWebRegionDescription(webRegionDescription);

memberPolicyGroupInformationDetailsPOJO.setOpsCenterDescr(opsCenterDescr);

memberPolicyGroupInformationDetailsPOJO.setEmployeesTaffid(employeesTaffid);

memberPolicyGroupInformationDetailsPOJO.setIssuingIsuredid(issuingIsuredid);

memberPolicyGroupInformationDetailsPOJO.setCoverageArea(coverageArea);

memberPolicyGroupInformationDetailsPOJO.setNetworkProviderTier(networkProviderTier);

memberPolicyGroupInformationDetailsPOJO.setCorrespondenceAddr(correspondenceAddr);

memberPolicyGroupInformationDetailsPOJO.setMemberRelgflg(memberRelgflg);

memberPolicyGroupInformationDetailsPOJO.setdExcess(dExcess);

memberPolicyGroupInformationDetailsPOJO.setsCurrency(sCurrency);

memberPolicyGroupInformationDetailsPOJO.setSumType(sumType);

memberPolicyGroupInformationDetailsPOJO.setSmsTNumber(smsTNumber);

memberPolicyGroupInformationDetailsPOJO.setBsendMBrid(bsendMBrid);

memberPolicyGroupInformationDetailsPOJO.setCoiAddress(coiAddress);

memberPolicyGroupInformationDetailsPOJO.setdCoinsoopMax(dCoinsoopMax);

memberPolicyGroupInformationDetailsPOJO.setsDirectBilling(sDirectBilling);

memberPolicyGroupInformationDetailsPOJO.setsRoutineDental(sRoutineDental);

memberPolicyGroupInformationDetailsPOJO.setdRoutineDentalCoins(dRoutineDentalCoins);

memberPolicyGroupInformationDetailsPOJO.setSmarjRestOrative(smarjRestOrative);

memberPolicyGroupInformationDetailsPOJO.setDmajRestCoins(dmajRestCoins);

memberPolicyGroupInformationDetailsPOJO.setsDental(sDental);

memberPolicyGroupInformationDetailsPOJO.setdDentalCOins(dDentalCOins);

memberPolicyGroupInformationDetailsPOJO.setdStandardCoins(dStandardCoins);

memberPolicyGroupInformationDetailsPOJO.setdCoPay(dCoPay);

memberPolicyGroupInformationDetailsPOJO.setdOutPatientCopay(dOutPatientCopay);

memberPolicyGroupInformationDetailsPOJO.setSchronic(schronic);

memberPolicyGroupInformationDetailsPOJO.setSchronicTier(schronicTier);

memberPolicyGroupInformationDetailsPOJO.setdMaternityWaitingGperiod(dMaternityWaitingGperiod);

memberPolicyGroupInformationDetailsPOJO.setIdCardDescription1(idCardDescription1);

memberPolicyGroupInformationDetailsPOJO.setIdCardDescription2(idCardDescription2);

memberPolicyGroupInformationDetailsPOJO.setBrokerId(brokerId);

memberPolicyGroupInformationDetailsPOJO.setBrokerName(brokerName);

memberPolicyGroupInformationDetailsPOJO.setPlanType(PlanType);

memberPolicyGroupInformationDetailsPOJO.setsBrokerPhone(sBrokerPhone);

memberPolicyGroupInformationDetailsPOJO.setsPlanSponsorPhone(sPlanSponsorPhone);

memberPolicyGroupInformationDetailsPOJO.setsExcessDeductiblefg(sExcessDeductiblefg);

memberPolicyGroupInformationDetailsPOJO.setdExcessPermedCond(dExcessPermedCond);

memberPolicyGroupInformationDetailsPOJO.setdOutpatientCoinerPClaim(dOutpatientCoinerPClaim);

memberPolicyGroupInformationDetailsPOJO.setdRoutineMajorResdental(dRoutineMajorResdental);

memberPolicyGroupInformationDetailsPOJO.setTier4Dubai(tier4Dubai);

memberPolicyGroupInformationDetailsPOJO.setOutPatientDirectBilling(outPatientDirectBilling);

memberPolicyGroupInformationDetailsPOJO.setSoptrRegChildBenefit(soptrRegChildBenefit);

memberPolicyGroupInformationDetailsPOJO.setdPremiumPayment(dPremiumPayment);

memberPolicyGroupInformationDetailsPOJO.setlMainMedicalPolicyID(lMainMedicalPolicyID);

memberPolicyGroupInformationDetailsPOJO.setNetworkProvider(networkProvider);

System.out.println(memberPolicyGroupInformationDetailsPOJO);

MemberPolicyGroupInformationDetailsList.add(memberPolicyGroupInformationDetailsPOJO);

}

return MemberPolicyGroupInformationDetailsList;

}

public static ArrayList<MemberDependentInformationDetailsPOJO> memberDependentInformationDetailsList(ResultSet rs) throws SQLException {

//table 7

ArrayList<MemberDependentInformationDetailsPOJO> memberDependentInformationDetailsList = new ArrayList<>();

while (rs.next()) {

MemberDependentInformationDetailsPOJO memberDependentInformationDetailsPOJO = new MemberDependentInformationDetailsPOJO();

String memberID=rs.getString(1);

String dependantMemberID=rs.getString(2);

String consent=rs.getString(3);

String instanceID=rs.getString(4);

String relationshipStartDate=rs.getString(5);

if(relationshipStartDate!=null) {

String relationshipStartDatesplit[]=relationshipStartDate.split("\\s+");

memberDependentInformationDetailsPOJO.setRelationshipStartDate(relationshipStartDatesplit[0]);

}

else {

memberDependentInformationDetailsPOJO.setRelationshipStartDate(relationshipStartDate);

}

String relationshipEndDate=rs.getString(6);

if(relationshipEndDate!=null) {

String relationshipEndDatesplit[]=relationshipEndDate.split("\\s+");

memberDependentInformationDetailsPOJO.setRelationshipEndDate(relationshipEndDatesplit[0]);

}

else {

memberDependentInformationDetailsPOJO.setRelationshipEndDate(relationshipEndDate);

}

String relationShipType=rs.getString(7);

memberDependentInformationDetailsPOJO.setMemberID(memberID);

memberDependentInformationDetailsPOJO.setDependantMemberID(dependantMemberID);

memberDependentInformationDetailsPOJO.setConsent(consent);

memberDependentInformationDetailsPOJO.setInstanceID(instanceID);

memberDependentInformationDetailsPOJO.setRelationShipType(relationShipType);

System.out.println(memberDependentInformationDetailsPOJO);

memberDependentInformationDetailsList.add(memberDependentInformationDetailsPOJO);

}

return memberDependentInformationDetailsList;

}

public static ArrayList<MemberDependentInformationDetailsConsentDetailsPOJO> memberDependentInformationDetailsConsentDetailsList(ResultSet rs) throws SQLException {

//table 8

ArrayList<MemberDependentInformationDetailsConsentDetailsPOJO> memberDependentInformationDetailsConsentDetailsList = new ArrayList<>();

while (rs.next()) {

MemberDependentInformationDetailsConsentDetailsPOJO memberDependentInformationDetailsConsentDetailsPOJO = new MemberDependentInformationDetailsConsentDetailsPOJO();

String memberID=rs.getString(1);

String title=rs.getString(2);

String firstname=rs.getString(3);

String surname=rs.getString(4);

String initials=rs.getString(5);

//String dob=df.format(rs.getTimestamp(6));

String dob=rs.getString(6);

if(dob!=null) {

String dobSplit[]=dob.split("\\s+");

memberDependentInformationDetailsConsentDetailsPOJO.setDateOfBirth(dobSplit[0]);

}

else {

memberDependentInformationDetailsConsentDetailsPOJO.setDateOfBirth(dob);

}

String email=rs.getString(7);

String isActive=rs.getString(8);

String instanceID=rs.getString(9);

String branchID=rs.getString(10);

String countryOfResidence =rs.getString(11);

String memberPhone=rs.getString(12);

String nationality=rs.getString(13);

String occupation=rs.getString(14);

String passportID=rs.getString(15);

// String passportExpiryDate=df.format(rs.getTimestamp(16));

String passportExpiryDate=rs.getString(16);

if(passportExpiryDate!=null) {

String passportExpiryDateSplit[]=passportExpiryDate.split("\\s+");

memberDependentInformationDetailsConsentDetailsPOJO.setPassportExpiryDate(passportExpiryDateSplit[0]);

}

else {

memberDependentInformationDetailsConsentDetailsPOJO.setPassportExpiryDate(passportExpiryDate);

}

memberDependentInformationDetailsConsentDetailsPOJO.setMemberID(memberID);

memberDependentInformationDetailsConsentDetailsPOJO.setTitle(title);

memberDependentInformationDetailsConsentDetailsPOJO.setFirstname(firstname);

memberDependentInformationDetailsConsentDetailsPOJO.setSurname(surname);

memberDependentInformationDetailsConsentDetailsPOJO.setInitials(initials);

memberDependentInformationDetailsConsentDetailsPOJO.setEmail(email);

memberDependentInformationDetailsConsentDetailsPOJO.setIsActive(isActive);

memberDependentInformationDetailsConsentDetailsPOJO.setInstanceID(instanceID);

memberDependentInformationDetailsConsentDetailsPOJO.setBranchID(branchID);

memberDependentInformationDetailsConsentDetailsPOJO.setCountryOfResidence(countryOfResidence);

memberDependentInformationDetailsConsentDetailsPOJO.setMemberPhone(memberPhone);

memberDependentInformationDetailsConsentDetailsPOJO.setNationality(nationality);

memberDependentInformationDetailsConsentDetailsPOJO.setOcccupation(occupation);

memberDependentInformationDetailsConsentDetailsPOJO.setPassportID(passportID);

System.out.println(memberDependentInformationDetailsConsentDetailsPOJO);

memberDependentInformationDetailsConsentDetailsList.add(memberDependentInformationDetailsConsentDetailsPOJO);

}

return memberDependentInformationDetailsConsentDetailsList;

}

public static ArrayList<MemberOnlineClaimSubmissionDetailsPOJO> memberOnlineClaimSubmissionDetailsList(ResultSet rs) throws SQLException {

//table 9

ArrayList<MemberOnlineClaimSubmissionDetailsPOJO> memberOnlineClaimSubmissionDetailsList = new ArrayList<>();

while (rs.next()) {

MemberOnlineClaimSubmissionDetailsPOJO memberOnlineClaimSubmissionDetailsPOJO = new MemberOnlineClaimSubmissionDetailsPOJO();

String onlclmsbmId=rs.getString(1);

String onlclmsbmClmtrckId=rs.getString(2);

String onlclmsbmReftrckId=rs.getString(3);

String onlclmsbmEeNm=rs.getString(4);

String onlclmsbmMbrId=rs.getString(5);

String onlclmsbmInstId=rs.getString(6);

String onlclmsbmPolicyId=rs.getString(7);

String onlclmsbmPolhldrNm=rs.getString(8);

String onlclmsbmEmailTxt=rs.getString(9);

String onlclmsbmEefirstNm=rs.getString(10);

String onlclmsbmEelastNm=rs.getString(11);

String onlclmsbmSubmfirstNm=rs.getString(12);

String onlclmsbmSubmlastNm=rs.getString(13);

String onlclmsbmCntctemailTxt=rs.getString(14);

String onlclmsbmSubmId=rs.getString(15);

String onlclmsbmRefileNm=rs.getString(16);

String onlclmsbmSubmsttsCd=rs.getString(17);

String onlclmsbmValidattchmntCnt=rs.getString(18);

String onlclmsbmBtchsttsCd=rs.getString(19);

String onlclmsbmBtchDt=rs.getString(20);

String onlclmsbmDelDt=rs.getString(21);

String onlclmsbmSubmDts=rs.getString(22);

String onlclmsbmWqId=rs.getString(23);

String onlclmsbmClaimType=rs.getString(24);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmId(onlclmsbmId);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmClmtrckId(onlclmsbmClmtrckId);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmReftrckId(onlclmsbmReftrckId);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmEeNm(onlclmsbmEeNm);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmMbrId(onlclmsbmMbrId);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmInstId(onlclmsbmInstId);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmPolicyId(onlclmsbmPolicyId);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmPolhldrNm(onlclmsbmPolhldrNm);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmEmailTxt(onlclmsbmEmailTxt);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmEefirstNm(onlclmsbmEefirstNm);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmEelastNm(onlclmsbmEelastNm);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmSubmfirstNm(onlclmsbmSubmfirstNm);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmSubmlastNm(onlclmsbmSubmlastNm);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmCntctemailTxt(onlclmsbmCntctemailTxt);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmSubmId(onlclmsbmSubmId);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmRefileNm(onlclmsbmRefileNm);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmSubmsttsCd(onlclmsbmSubmsttsCd);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmValidattchmntCnt(onlclmsbmValidattchmntCnt);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmBtchsttsCd(onlclmsbmBtchsttsCd);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmBtchDt(onlclmsbmBtchDt);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmDelDt(onlclmsbmDelDt);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmSubmDts(onlclmsbmSubmDts);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmWqId(onlclmsbmWqId);

memberOnlineClaimSubmissionDetailsPOJO.setOnlclmsbmClaimType(onlclmsbmClaimType);

System.out.println(memberOnlineClaimSubmissionDetailsPOJO);

memberOnlineClaimSubmissionDetailsList.add(memberOnlineClaimSubmissionDetailsPOJO);

}

return memberOnlineClaimSubmissionDetailsList;

}

public static ArrayList<MemberOnlineClaimAuditDetailsPOJO> memberOnlineClaimAuditDetailsList(ResultSet rs) throws SQLException {

//table 10

ArrayList<MemberOnlineClaimAuditDetailsPOJO> memberOnlineClaimAuditDetailsList = new ArrayList<>();

while (rs.next()) {

MemberOnlineClaimAuditDetailsPOJO memberOnlineClaimAuditDetailsPOJO = new MemberOnlineClaimAuditDetailsPOJO();

String onlclmauditId=rs.getString(1);

String mbrId=rs.getString(2);

String mbrFname=rs.getString(3);

String mbrLname=rs.getString(4);

String patAetnaId=rs.getString(5);

String patName=rs.getString(6);

String patDob =rs.getString(7);

if(patDob!=null) {

String patDobsplit[]=patDob.split("\\s+");

memberOnlineClaimAuditDetailsPOJO.setPatDob(patDobsplit[0]);

}

else {

memberOnlineClaimAuditDetailsPOJO.setPatDob(patDob);

}

String patGender=rs.getString(8);

String policyNum=rs.getString(9);

String groupNm=rs.getString(10);

String correspondenceAddr=rs.getString(11);

String postalCd=rs.getString(12);

String town=rs.getString(13);

String countryId=rs.getString(14);

String patEmail=rs.getString(15);

String patMobilePhone=rs.getString(16);

String patAlternatePhone=rs.getString(17);

String instanceId=rs.getString(18);

String onlclmsbmClmtrckId=rs.getString(19);

String createdDt=rs.getString(20);

if(createdDt!=null) {

String createdDtsplit[]=createdDt.split("\\s+");

memberOnlineClaimAuditDetailsPOJO.setCreatedDt(createdDtsplit[0]);

}

else {

memberOnlineClaimAuditDetailsPOJO.setCreatedDt(createdDt);

}

String createdBy=rs.getString(21);

String onlclmsbmReftrckId=rs.getString(22);

String trtmntMnldChina=rs.getString(23);

String branchId=rs.getString(24);

String lstStepVisit=rs.getString(25);

String outpatientClaim=rs.getString(26);

String addressLine2=rs.getString(27);

String countyState=rs.getString(8);

memberOnlineClaimAuditDetailsPOJO.setOnlclmauditId(onlclmauditId);

memberOnlineClaimAuditDetailsPOJO.setMbrId(mbrId);

memberOnlineClaimAuditDetailsPOJO.setMbrFname(mbrFname);

memberOnlineClaimAuditDetailsPOJO.setMbrLname(mbrLname);

memberOnlineClaimAuditDetailsPOJO.setPatAetnaId(patAetnaId);

memberOnlineClaimAuditDetailsPOJO.setPatName(patName);

memberOnlineClaimAuditDetailsPOJO.setPatGender(patGender);

memberOnlineClaimAuditDetailsPOJO.setPolicyNum(policyNum);

memberOnlineClaimAuditDetailsPOJO.setGroupNm(groupNm);

memberOnlineClaimAuditDetailsPOJO.setCorrespondenceAddr(correspondenceAddr);

memberOnlineClaimAuditDetailsPOJO.setPostalCd(postalCd);

memberOnlineClaimAuditDetailsPOJO.setTown(town);

memberOnlineClaimAuditDetailsPOJO.setCountryId(countryId);

memberOnlineClaimAuditDetailsPOJO.setPatEmail(patEmail);

memberOnlineClaimAuditDetailsPOJO.setPatMobilePhone(patMobilePhone);

memberOnlineClaimAuditDetailsPOJO.setPatAlternatePhone(patAlternatePhone);

memberOnlineClaimAuditDetailsPOJO.setInstanceId(instanceId);

memberOnlineClaimAuditDetailsPOJO.setOnlclmsbmClmtrckId(onlclmsbmClmtrckId);

memberOnlineClaimAuditDetailsPOJO.setCreatedBy(createdBy);

memberOnlineClaimAuditDetailsPOJO.setOnlclmsbmReftrckId(onlclmsbmReftrckId);

memberOnlineClaimAuditDetailsPOJO.setTrtmntMnldChina(trtmntMnldChina);

memberOnlineClaimAuditDetailsPOJO.setBranchId(branchId);

memberOnlineClaimAuditDetailsPOJO.setLstStepVisit(lstStepVisit);

memberOnlineClaimAuditDetailsPOJO.setOutpatientClaim(outpatientClaim);

memberOnlineClaimAuditDetailsPOJO.setAddressLine2(addressLine2);

memberOnlineClaimAuditDetailsPOJO.setCountyState(countyState);

System.out.println(memberOnlineClaimAuditDetailsPOJO);

memberOnlineClaimAuditDetailsList.add(memberOnlineClaimAuditDetailsPOJO);

}

return memberOnlineClaimAuditDetailsList;

}

public static ArrayList<MemberClaimDetailsBothOnlineOfflineclaimsPOJO> memberClaimDetailsBothOnlineOfflineclaimsList(ResultSet rs) throws SQLException {

//table 11

ArrayList<MemberClaimDetailsBothOnlineOfflineclaimsPOJO> memberClaimDetailsBothOnlineOfflineclaimsList = new ArrayList<>();

while (rs.next()) {

MemberClaimDetailsBothOnlineOfflineclaimsPOJO memberClaimDetailsBothOnlineOfflineclaimsPOJO = new MemberClaimDetailsBothOnlineOfflineclaimsPOJO();

String claimID=rs.getString(1);

String memberID=rs.getString(2);

String policyID=rs.getString(3);

String policyHolderName=rs.getString(4);

String startDateOfService=rs.getString(5);

if(startDateOfService!=null) {

String startDateOfServicestartDateOfServicesplit[]=startDateOfService.split("\\s+");

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setStartDateOfService(startDateOfServicestartDateOfServicesplit[0]);

}

else {

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setStartDateOfService(startDateOfService);

}

String endDateOfService=rs.getString(6);

if(endDateOfService!=null) {

String endDateOfServicesplit[]=endDateOfService.split("\\s+");

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setEndDateOfService(endDateOfServicesplit[0]);

}

else {

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setEndDateOfService(endDateOfService);

}

String firstName=rs.getString(7);

String surName=rs.getString(8);

String dateOfBirth=rs.getString(9);

if(dateOfBirth!=null) {

String dateOfBirthsplit[]=dateOfBirth.split("\\s+");

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setDateOfBirth(dateOfBirthsplit[0]);

}

else {

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setDateOfBirth(dateOfBirth);

}

String claimStatusID=rs.getString(10);

String providerID=rs.getString(11);

String provider=rs.getString(12);

String totalSubmittedIC=rs.getString(13);

String totalSubmittedPBC=rs.getString(14);

String totalPaidIC=rs.getString(15);

String totalPaidPC=rs.getString(16);

String invoiceCurrencyID=rs.getString(17);

String policyBasedCurrencyID=rs.getString(18);

String paymentCurrencyID=rs.getString(19);

String totalDiscountedAmountIC=rs.getString(20);

String totalDiscountedAmountPBC=rs.getString(21);

String outOfPocketIC=rs.getString(22);

String outOfPocketPBC=rs.getString(23);

String amalgamatedAmountPC=rs.getString(24);

String payeeID=rs.getString(25);

String payee=rs.getString(26);

String paymentType=rs.getString(27);

String paymentReferenceNumber=rs.getString(28);

String paymentDate=rs.getString(29);

if(paymentDate!=null) {

String paymentDatesplit[]=paymentDate.split("\\s+");

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPaymentDate(paymentDatesplit[0]);

}

else {

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPaymentDate(paymentDate);

}

String payeeAddress1=rs.getString(30);

String payeeAddress2=rs.getString(31);

String payeeAddress3=rs.getString(32);

String payeeAddress4=rs.getString(33);

String payeePostalCode=rs.getString(34);

String payeeCountry=rs.getString(35);

String providerInvoiceNumber=rs.getString(36);

String instanceID=rs.getString(37);

String paymentStatus=rs.getString(38);

String paymentReference=rs.getString(39);

String claimStatusOHPlus=rs.getString(40);

String sReference=rs.getString(41);

String sPaymentReference=rs.getString(42);

String citiPaymentAggregationKey=rs.getString(43);

String submittedDate=rs.getString(44);

if(submittedDate!=null) {

String submittedDatesplit[]=submittedDate.split("\\s+");

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setSubmittedDate(submittedDatesplit[0]);

}

else {

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setSubmittedDate(submittedDate);

}

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setClaimID(claimID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setMemberID(memberID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPolicyID(policyID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPolicyHolderName(policyHolderName);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setFirstName(firstName);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setSurName(surName);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setClaimStatusID(claimStatusID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setProviderID(providerID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setProvider(provider);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setTotalSubmittedIC(totalSubmittedIC);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setTotalSubmittedPBC(totalSubmittedPBC);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setTotalPaidIC(totalPaidIC);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setTotalPaidPC(totalPaidPC);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setInvoiceCurrencyID(invoiceCurrencyID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPolicyBasedCurrencyID(policyBasedCurrencyID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPaymentCurrencyID(paymentCurrencyID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setTotalDiscountedAmountIC(totalDiscountedAmountIC);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setTotalDiscountedAmountPBC(totalDiscountedAmountPBC);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setOutOfPocketIC(outOfPocketIC);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setOutOfPocketPBC(outOfPocketPBC);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setAmalgamatedAmountPC(amalgamatedAmountPC);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPayeeID(payeeID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPayee(payee);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPaymentType(paymentType);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPaymentReferenceNumber(paymentReferenceNumber);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPayeeAddress1(payeeAddress1);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPayeeAddress2(payeeAddress2);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPayeeAddress3(payeeAddress3);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPayeeAddress4(payeeAddress4);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPayeePostalCode(payeePostalCode);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPayeeCountry(payeeCountry);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setProviderInvoiceNumber(providerInvoiceNumber);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setInstanceID(instanceID);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPaymentStatus(paymentStatus);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setPaymentReference(paymentReference);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setClaimStatusOHPlus(claimStatusOHPlus);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setsReference(sReference);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setsPaymentReference(sPaymentReference);

memberClaimDetailsBothOnlineOfflineclaimsPOJO.setCitiPaymentAggregationKey(citiPaymentAggregationKey);

System.out.println(memberClaimDetailsBothOnlineOfflineclaimsPOJO);

memberClaimDetailsBothOnlineOfflineclaimsList.add(memberClaimDetailsBothOnlineOfflineclaimsPOJO);

}

return memberClaimDetailsBothOnlineOfflineclaimsList;

}

public static ArrayList<MemberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO> memberClaimLineItemDetailsBothOnlineOfflineClaimsList(ResultSet rs) throws SQLException {

//table 12

ArrayList<MemberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO> memberClaimLineItemDetailsBothOnlineOfflineClaimsList = new ArrayList<>();

while (rs.next()) {

MemberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO = new MemberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO();

String claimID=rs.getString(1);

String lineID=rs.getString(2);

String dateOfService=rs.getString(3);

if(dateOfService!=null) {

String dateOfServicesplit[]=dateOfService.split("\\s+");

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setDateOfService(dateOfServicesplit[0]);

}

else {

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setDateOfService(dateOfService);

}

String serviceProvider=rs.getString(4);

String chargesSubmittedIC=rs.getString(5);

String discountedAmountIC=rs.getString(6);

String chargesSubmittedPBC=rs.getString(7);

String discountedAmountPBC=rs.getString(8);

String paidAmountIC=rs.getString(9);

String paidAmountPBC=rs.getString(10);

String coInsuranceAmountPBC=rs.getString(11);

String policyExpressAmountPBC=rs.getString(12);

String benefitAccessAmountPBC=rs.getString(13);

String deniedAmountIC=rs.getString(14);

String outOfPocketIC=rs.getString(15);

String outOfPocketPBC=rs.getString(16);

String invoiceCurrencyID=rs.getString(17);

String policyBasedCurrencyID=rs.getString(18);

String remarkTest=rs.getString(19);

String remarkID=rs.getString(20);

String instanceID=rs.getString(21);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setClaimID(claimID);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setLineID(lineID);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setServiceProvider(serviceProvider);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setChargesSubmittedIC(chargesSubmittedIC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setDiscountedAmountIC(discountedAmountIC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setChargesSubmittedPBC(chargesSubmittedPBC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setDiscountedAmountPBC(discountedAmountPBC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setPaidAmountIC(paidAmountIC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setPaidAmountPBC(paidAmountPBC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setCoInsuranceAmountPBC(coInsuranceAmountPBC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setPolicyExcessAmountPBC(policyExpressAmountPBC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setBenefitAccessAmountPBC(benefitAccessAmountPBC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setDeniedAmountIC(deniedAmountIC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setOutOfPocketIC(outOfPocketIC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setOutOfPocketPBC(outOfPocketPBC);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setInvoiceCurrencyID(invoiceCurrencyID);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setPolicyBasedCurrencyID(policyBasedCurrencyID);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setRemarkTest(remarkTest);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setRemarkID(remarkID);

memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO.setInstanceID(instanceID);

System.out.println(memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO);

memberClaimLineItemDetailsBothOnlineOfflineClaimsList.add(memberClaimLineItemDetailsBothOnlineOfflineClaimsPOJO);

}

return memberClaimLineItemDetailsBothOnlineOfflineClaimsList;

}

public static ArrayList<MemberRREBankInformationDetailsPOJO> memberRREBankInformationDetailsList(ResultSet rs) throws SQLException {

//table 13

ArrayList<MemberRREBankInformationDetailsPOJO> memberRREBankInformationDetailsList = new ArrayList<>();

while (rs.next()) {

MemberRREBankInformationDetailsPOJO memberRREBankInformationDetailsPOJO = new MemberRREBankInformationDetailsPOJO();

String rReSubmissionId=rs.getString(1);

String memberId=rs.getString(2);

String instanceId=rs.getString(3);

String submissionState=rs.getString(4);

String submissionDate=rs.getString(5);

if(submissionDate!=null) {

String submissionDatesplit[]=submissionDate.split("\\s+");

memberRREBankInformationDetailsPOJO.setSubmissionDate(submissionDatesplit[0]);

}

else {

memberRREBankInformationDetailsPOJO.setSubmissionDate(submissionDate);

}

String paymentMethod=rs.getString(6);

String destCurrenceId=rs.getString(7);

String achBankCode=rs.getString(8);

String achBranchCode=rs.getString(9);

String accountNumber=rs.getString(10);

String achCheckDigits=rs.getString(11);

String accountTypeId=rs.getString(12);

String postBankTownName=rs.getString(13);

String iban=rs.getString(14);

String bankIdentCodeTypeId=rs.getString(15);

String bankIdentCode=rs.getString(16);

String destBankName=rs.getString(17);

String interBankIdentCodeTypeId=rs.getString(8);

String interBankIdentCode=rs.getString(9);

String interBankName=rs.getString(20);

String beneficiaryFirstName=rs.getString(21);

String beneficiaryMiddleName=rs.getString(22);

String beneficiaryLastName=rs.getString(23);

String beneficiaryPhone=rs.getString(24);

String beneficiaryEmail=rs.getString(25);

String checkClearingCountryId=rs.getString(26);

String comments=rs.getString(27);

String pdfCreateDt=rs.getString(28);

if(pdfCreateDt!=null) {

String pdfCreateDtsplit[]=pdfCreateDt.split("\\s+");

memberRREBankInformationDetailsPOJO.setPdfCreateDt(pdfCreateDtsplit[0]);

}

else {

memberRREBankInformationDetailsPOJO.setPdfCreateDt(pdfCreateDt);

}

String dataCleanupDt=rs.getString(29);

if(dataCleanupDt!=null) {

String dataCleanupDtsplit[]=dataCleanupDt.split("\\s+");

memberRREBankInformationDetailsPOJO.setDataCleanupDt(dataCleanupDtsplit[0]);

}

else {

memberRREBankInformationDetailsPOJO.setDataCleanupDt(dataCleanupDt);

}

String destBankAddressId=rs.getString(30);

String interBankAddressId=rs.getString(31);

String beneficiaryAddressId=rs.getString(32);

String onlclmpaydtlId=rs.getString(33);

String onlclmsbmClmtrckId=rs.getString(34);

String onlclmsbmReftrckId=rs.getString(35);

String rreWqId=rs.getString(36);

String ohDestCountryId=rs.getString(37);

String ohDestCurrencyId=rs.getString(38);

String ohDestBankAddressId=rs.getString(39);

String ohInterBankAddressId=rs.getString(40);

memberRREBankInformationDetailsPOJO.setrReSubmissionId(rReSubmissionId);

memberRREBankInformationDetailsPOJO.setMemberId(memberId);

memberRREBankInformationDetailsPOJO.setInstanceId(instanceId);

memberRREBankInformationDetailsPOJO.setSubmissionState(submissionState);

memberRREBankInformationDetailsPOJO.setPaymentMethod(paymentMethod);

memberRREBankInformationDetailsPOJO.setDestCurrenceId(destCurrenceId);

memberRREBankInformationDetailsPOJO.setAchBankCode(achBankCode);

memberRREBankInformationDetailsPOJO.setAchBranchCode(achBranchCode);

memberRREBankInformationDetailsPOJO.setAccountNumber(accountNumber);

memberRREBankInformationDetailsPOJO.setAchCheckDigits(achCheckDigits);

memberRREBankInformationDetailsPOJO.setAccountTypeId(accountTypeId);

memberRREBankInformationDetailsPOJO.setPostBankTownName(postBankTownName);

memberRREBankInformationDetailsPOJO.setIban(iban);

memberRREBankInformationDetailsPOJO.setBankIdentCodeTypeId(bankIdentCodeTypeId);

memberRREBankInformationDetailsPOJO.setBankIdentCode(bankIdentCode);

memberRREBankInformationDetailsPOJO.setDestBankName(destBankName);

memberRREBankInformationDetailsPOJO.setInterBankIdentCodeTypeId(interBankIdentCodeTypeId);

memberRREBankInformationDetailsPOJO.setInterBankIdentCode(interBankIdentCode);

memberRREBankInformationDetailsPOJO.setInterBankName(interBankName);

memberRREBankInformationDetailsPOJO.setBeneficiaryFirstName(beneficiaryFirstName);

memberRREBankInformationDetailsPOJO.setBeneficiaryMiddleName(beneficiaryMiddleName);

memberRREBankInformationDetailsPOJO.setBeneficiaryLastName(beneficiaryLastName);

memberRREBankInformationDetailsPOJO.setBeneficiaryPhone(beneficiaryPhone);

memberRREBankInformationDetailsPOJO.setBeneficiaryEmail(beneficiaryEmail);

memberRREBankInformationDetailsPOJO.setCheckClearingCountryId(checkClearingCountryId);

memberRREBankInformationDetailsPOJO.setComments(comments);

memberRREBankInformationDetailsPOJO.setDestBankAddressId(destBankAddressId);

memberRREBankInformationDetailsPOJO.setInterBankAddressId(interBankAddressId);

memberRREBankInformationDetailsPOJO.setBeneficiaryAddressId(beneficiaryAddressId);

memberRREBankInformationDetailsPOJO.setOnlclmpaydtlId(onlclmpaydtlId);

memberRREBankInformationDetailsPOJO.setOnlclmsbmClmtrckId(onlclmsbmClmtrckId);

memberRREBankInformationDetailsPOJO.setOnlclmsbmReftrckId(onlclmsbmReftrckId);

memberRREBankInformationDetailsPOJO.setRreWqId(rreWqId);

memberRREBankInformationDetailsPOJO.setOhDestCountryId(ohDestCountryId);

memberRREBankInformationDetailsPOJO.setOhDestCurrencyId(ohDestCurrencyId);

memberRREBankInformationDetailsPOJO.setOhDestBankAddressId(ohDestBankAddressId);

memberRREBankInformationDetailsPOJO.setOhInterBankAddressId(ohInterBankAddressId);

System.out.println(memberRREBankInformationDetailsPOJO);

memberRREBankInformationDetailsList.add(memberRREBankInformationDetailsPOJO);

}

return memberRREBankInformationDetailsList;

}

public static ArrayList<MemberBrokerConsentDetailsPOJO> memberBrokerConsentDetailsList(ResultSet rs) throws SQLException {

//table 14

ArrayList<MemberBrokerConsentDetailsPOJO> memberBrokerConsentDetailsList = new ArrayList<>();

while (rs.next()) {

MemberBrokerConsentDetailsPOJO memberBrokerConsentDetailsPOJO = new MemberBrokerConsentDetailsPOJO();

String brkrgmbrelID=rs.getString(1);

String brkrgProflRegKey=rs.getString(2);

String mbrID=rs.getString(3);

String brkrgmbrelSttsDesc=rs.getString(4);

String brkrgmbrelStartDTS=rs.getString(5);

String brkrgmbrelEndDTS=rs.getString(6);

String createdUsrID=rs.getString(7);

String createdDt=rs.getString(8);

String lstUpdUsrID=rs.getString(9);

String lstUpdDt=rs.getString(10);

String brkrgmbrelCnsntType=rs.getString(11);

memberBrokerConsentDetailsPOJO.setBrkrgmbrelID(brkrgmbrelID);

memberBrokerConsentDetailsPOJO.setBrkrgProflRegKey(brkrgProflRegKey);

memberBrokerConsentDetailsPOJO.setMbrID(mbrID);

memberBrokerConsentDetailsPOJO.setBrkrgmbrelSttsDesc(brkrgmbrelSttsDesc);

memberBrokerConsentDetailsPOJO.setBrkrgmbrelStartDTS(brkrgmbrelStartDTS);

memberBrokerConsentDetailsPOJO.setBrkrgmbrelEndDTS(brkrgmbrelEndDTS);

memberBrokerConsentDetailsPOJO.setCreatedUsrID(createdUsrID);

memberBrokerConsentDetailsPOJO.setCreatedDt(createdDt);

memberBrokerConsentDetailsPOJO.setLstUpdUsrID(lstUpdUsrID);

memberBrokerConsentDetailsPOJO.setLstUpdDt(lstUpdDt);

memberBrokerConsentDetailsPOJO.setBrkrgmbrelCnsntType(brkrgmbrelCnsntType);

System.out.println(memberBrokerConsentDetailsPOJO);

memberBrokerConsentDetailsList.add(memberBrokerConsentDetailsPOJO);

}

return memberBrokerConsentDetailsList;

}

public static ArrayList<MemberClaimEOBDetailsPOJO> memberClaimEOBDetailsList(ResultSet rs) throws SQLException {

//table 15

ArrayList<MemberClaimEOBDetailsPOJO> memberClaimEOBDetailsList = new ArrayList<>();

while (rs.next()) {

MemberClaimEOBDetailsPOJO memberClaimEOBDetailsPOJO = new MemberClaimEOBDetailsPOJO();

String filterID=rs.getString(1);

String claimID=rs.getString(2);

String instanceID=rs.getString(3);

String memberID=rs.getString(4);

String paymentSttsDesc=rs.getString(5);

String paymentTypeCD=rs.getString(6);

String claimSttsCD=rs.getString(7);

String paymentAmt=rs.getString(8);

String claimStartDt=rs.getString(9);

if(claimStartDt!=null) {

String claimStartDtsplit[]=claimStartDt.split("\\s+");

memberClaimEOBDetailsPOJO.setClaimStartDt(claimStartDtsplit[0]);

}

else {

memberClaimEOBDetailsPOJO.setClaimStartDt(claimStartDt);

}

String claimEndDt=rs.getString(10);

if(claimEndDt!=null) {

String claimEndDtsplit[]=claimEndDt.split("\\s+");

memberClaimEOBDetailsPOJO.setClaimEndDt(claimEndDtsplit[0]);

}

else {

memberClaimEOBDetailsPOJO.setClaimEndDt(claimEndDt);

}

String notificationSttsCD=rs.getString(11);

String createdDTS=rs.getString(12);

String notificationUpdatedDTS=rs.getString(13);

String birthDt=rs.getString(14);

if(birthDt!=null) {

String birthDtsplit[]=birthDt.split("\\s+");

memberClaimEOBDetailsPOJO.setBirthDt(birthDtsplit[0]);

}

else {

memberClaimEOBDetailsPOJO.setBirthDt(birthDt);

}

memberClaimEOBDetailsPOJO.setFilterID(filterID);

memberClaimEOBDetailsPOJO.setClaimID(claimID);

memberClaimEOBDetailsPOJO.setInstanceID(instanceID);

memberClaimEOBDetailsPOJO.setMemberID(memberID);

memberClaimEOBDetailsPOJO.setPaymentSttsDesc(paymentSttsDesc);

memberClaimEOBDetailsPOJO.setPaymentTypeCD(paymentTypeCD);

memberClaimEOBDetailsPOJO.setClaimSttsCD(claimSttsCD);

memberClaimEOBDetailsPOJO.setPaymentAmt(paymentAmt);

memberClaimEOBDetailsPOJO.setNotificationSttsCD(notificationSttsCD);

memberClaimEOBDetailsPOJO.setCreatedDTS(createdDTS);

memberClaimEOBDetailsPOJO.setNotificationUpdatedDTS(notificationUpdatedDTS);

System.out.println(memberClaimEOBDetailsPOJO);

memberClaimEOBDetailsList.add(memberClaimEOBDetailsPOJO);

}

return memberClaimEOBDetailsList;

}

public static ArrayList<MemberEOBEmailNotificationDetailsPOJO> memberEOBEmailNotificationDetailsList(ResultSet rs) throws SQLException {

//table 16

ArrayList<MemberEOBEmailNotificationDetailsPOJO> memberEOBEmailNotificationDetailsList = new ArrayList<>();

while (rs.next()) {

MemberEOBEmailNotificationDetailsPOJO memberEOBEmailNotificationDetailsPOJO = new MemberEOBEmailNotificationDetailsPOJO();

String alertID=rs.getString(1);

String claimID=rs.getString(2);

String instanceID=rs.getString(3);

String messageTargetVal=rs.getString(4);

String messageTypeCD=rs.getString(5);

String templateID=rs.getString(6);

String memberID=rs.getString(7);

String memberTypeCD=rs.getString(8);

String timeZoneVal=rs.getString(9);

String notificationSttsCD=rs.getString(10);

String createdDTS=rs.getString(11);

String notificationUpdatedDTS=rs.getString(12);

String filterID=rs.getString(13);

String mobileCountryCD=rs.getString(14);

String countryID=rs.getString(15);

String timeZoneID=rs.getString(16);

String userLstNm=rs.getString(17);

String userFrstNm=rs.getString(18);

String eobEmailNotifyIND=rs.getString(19);

String eobSMSNotifyIND=rs.getString(20);

String messageContentTxt=rs.getString(21);

memberEOBEmailNotificationDetailsPOJO.setAlertID(alertID);

memberEOBEmailNotificationDetailsPOJO.setClaimID(claimID);

memberEOBEmailNotificationDetailsPOJO.setInstanceID(instanceID);

memberEOBEmailNotificationDetailsPOJO.setMessageTargetVal(messageTargetVal);

memberEOBEmailNotificationDetailsPOJO.setMessageTypeCD(messageTypeCD);

memberEOBEmailNotificationDetailsPOJO.setTemplateID(templateID);

memberEOBEmailNotificationDetailsPOJO.setMemberID(memberID);

memberEOBEmailNotificationDetailsPOJO.setMemberTypeCD(memberTypeCD);

memberEOBEmailNotificationDetailsPOJO.setTimeZoneVal(timeZoneVal);

memberEOBEmailNotificationDetailsPOJO.setNotificationSttsCD(notificationSttsCD);

memberEOBEmailNotificationDetailsPOJO.setCreatedDTS(createdDTS);

memberEOBEmailNotificationDetailsPOJO.setNotificationUpdatedDTS(notificationUpdatedDTS);

memberEOBEmailNotificationDetailsPOJO.setFilterID(filterID);

memberEOBEmailNotificationDetailsPOJO.setMobileCountryCD(mobileCountryCD);

memberEOBEmailNotificationDetailsPOJO.setCountryID(countryID);

memberEOBEmailNotificationDetailsPOJO.setTimeZoneID(timeZoneID);

memberEOBEmailNotificationDetailsPOJO.setUserLstNm(userLstNm);

memberEOBEmailNotificationDetailsPOJO.setUserFrstNm(userFrstNm);

memberEOBEmailNotificationDetailsPOJO.setEobEmailNotifyIND(eobEmailNotifyIND);

memberEOBEmailNotificationDetailsPOJO.setEobSMSNotifyIND(eobSMSNotifyIND);

memberEOBEmailNotificationDetailsPOJO.setMessageContentTxt(messageContentTxt);

System.out.println(memberEOBEmailNotificationDetailsPOJO);

memberEOBEmailNotificationDetailsList.add(memberEOBEmailNotificationDetailsPOJO);

}

return memberEOBEmailNotificationDetailsList;

}

public static ArrayList<MemberEOBMobileSMSDetailsPOJO> memberEOBMobileSMSDetailsList(ResultSet rs) throws SQLException {

//table 17

ArrayList<MemberEOBMobileSMSDetailsPOJO> memberEOBMobileSMSDetailsList = new ArrayList<>();

while (rs.next()) {

MemberEOBMobileSMSDetailsPOJO memberEOBMobileSMSDetailsPOJO = new MemberEOBMobileSMSDetailsPOJO();

String mobileAlertId=rs.getString(1);

String claimId=rs.getString(2);

String instanceId=rs.getString(3);

String mobileCntryId=rs.getString(4);

String mobileNo=rs.getString(5);

String messageTypeCD=rs.getString(6);

String templateId=rs.getString(7);

String memberId=rs.getString(8);

String timeZoneVal=rs.getString(9);

String memberTypeCD=rs.getString(10);

String filterId=rs.getString(12);

String mobileNotificationSttsCd=rs.getString(13);

String createdDts=rs.getString(14);

String notificationUpdatedDTS=rs.getString(15);

String alertId=rs.getString(16);

String responseCD=rs.getString(17);

String requestId=rs.getString(18);

String responseId=rs.getString(19);

String countryId=rs.getString(20);

String timeZoneId=rs.getString(21);

String userLstNm=rs.getString(22);

String userFrstNm=rs.getString(23);

String eobEmailNotifyIND=rs.getString(24);

String eobSmsNotifyIND=rs.getString(25);

String messageCOntentTxt=rs.getString(26);

memberEOBMobileSMSDetailsPOJO.setMobileAlertId(mobileAlertId);

memberEOBMobileSMSDetailsPOJO.setClaimId(claimId);

memberEOBMobileSMSDetailsPOJO.setInstanceId(instanceId);

memberEOBMobileSMSDetailsPOJO.setMobileCntryId(mobileCntryId);

memberEOBMobileSMSDetailsPOJO.setMobileNo(mobileNo);

memberEOBMobileSMSDetailsPOJO.setMessageTypeCD(messageTypeCD);

memberEOBMobileSMSDetailsPOJO.setTemplateId(templateId);

memberEOBMobileSMSDetailsPOJO.setMemberId(memberId);

memberEOBMobileSMSDetailsPOJO.setTimeZoneVal(timeZoneVal);

memberEOBMobileSMSDetailsPOJO.setMemberTypeCD(memberTypeCD);

memberEOBMobileSMSDetailsPOJO.setFilterId(filterId);

memberEOBMobileSMSDetailsPOJO.setMobileNotificationSttsCd(mobileNotificationSttsCd);

memberEOBMobileSMSDetailsPOJO.setCreatedDts(createdDts);

memberEOBMobileSMSDetailsPOJO.setNotificationUpdatedDTS(notificationUpdatedDTS);

memberEOBMobileSMSDetailsPOJO.setAlertId(alertId);

memberEOBMobileSMSDetailsPOJO.setResponseCD(responseCD);

memberEOBMobileSMSDetailsPOJO.setRequestId(requestId);

memberEOBMobileSMSDetailsPOJO.setResponseId(responseId);

memberEOBMobileSMSDetailsPOJO.setCountryId(countryId);

memberEOBMobileSMSDetailsPOJO.setTimeZoneId(timeZoneId);

memberEOBMobileSMSDetailsPOJO.setUserLstNm(userLstNm);

memberEOBMobileSMSDetailsPOJO.setUserFrstNm(userFrstNm);

memberEOBMobileSMSDetailsPOJO.setEobEmailNotifyIND(eobEmailNotifyIND);

memberEOBMobileSMSDetailsPOJO.setEobSmsNotifyIND(eobSmsNotifyIND);

memberEOBMobileSMSDetailsPOJO.setMessageCOntentTxt(messageCOntentTxt);

System.out.println(memberEOBMobileSMSDetailsPOJO);

memberEOBMobileSMSDetailsList.add(memberEOBMobileSMSDetailsPOJO);

}

return memberEOBMobileSMSDetailsList;

}

public static ArrayList<MemberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO> memberCertificateOfInsuranceDetailsOnlyforActisureUsersList(ResultSet rs) throws SQLException {

//table 18

ArrayList<MemberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO> memberCertificateOfInsuranceDetailsOnlyforActisureUsersList = new ArrayList<>();

while (rs.next()) {

MemberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO = new MemberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO();

String mbroutpdetMbrId=rs.getString(1);

String mbroutpdetInstanceId=rs.getString(2);

String mbroutpdetDoccatCD=rs.getString(3);

String mbroutpdetFileNm=rs.getString(4);

String mbroutpdetFilepathNm=rs.getString(5);

String mbroutpdetOutputDt=rs.getString(6);

if(mbroutpdetOutputDt!=null) {

String mbroutpdetOutputDtsplit[]=mbroutpdetOutputDt.split("\\s+");

memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO.setMbroutpdetOutputDt(mbroutpdetOutputDtsplit[0]);

}

else {

memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO.setMbroutpdetOutputDt(mbroutpdetOutputDt);

}

memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO.setMbroutpdetMbrId(mbroutpdetMbrId);

memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO.setMbroutpdetDoccatCD(mbroutpdetDoccatCD);

memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO.setMbroutpdetInstanceId(mbroutpdetInstanceId);

memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO.setMbroutpdetFileNm(mbroutpdetFileNm);

memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO.setMbroutpdetFilepathNm(mbroutpdetFilepathNm);

System.out.println(memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO);

memberCertificateOfInsuranceDetailsOnlyforActisureUsersList.add(memberCertificateOfInsuranceDetailsOnlyforActisureUsersPOJO);

}

return memberCertificateOfInsuranceDetailsOnlyforActisureUsersList;

}

}

toExcel :