SOURCE CODE

```
//for extracting customer details by giving account number

package crm;


import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStreamReader;

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

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

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


public class Details {


   public static String getDetails(int acc)throws IOException, NullPointerException,
FileNotFoundException

   {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

       //System.out.println("Please enter the account number!!");

       //int a = Integer.parseInt(br.readLine());


       //for loading corresponding data file in excel format.

       FileInputStream file = new FileInputStream(new File("C:\\Users\\vibhati
joshi\\Documents\\NetBeansProjects\\Apriori\\apriori.xls"));


       //creating new workbook to assign the file
```

```
HSSFWorkbook detail = new HSSFWorkbook(file);

//get sheet in  which data is present

HSSFSheet detailsheet = detail.getSheetAt(0);

String set = null;

for(int i =2;i<2938;i++)

{

   HSSFCell detailcell = detailsheet.getRow(i).getCell(0);

   if((int)detailcell.getNumericCellValue()== acc)

   {

      set = "The details of the required customer are: \n";

      //System.out.println("The details of the requested customer are : ");

    for(int j = 0;j<4;j++)

      {

       detailcell = detailsheet.getRow(i).getCell(j);


       switch (j)

        {

           case 0:

              int no = (int)detailcell.getNumericCellValue();

              //System.out.println("Account number : " + no);

              set = (set + "Account number: " + no + ".\n");

              break;


           case 1:

              String name = detailcell.getStringCellValue();

              //System.out.println("Name of customer : " + name);

              set = (set + "Name of customer: " + name + ".\n");
```

```
                    break;


                  case 2:

                    long number =(long)detailcell.getNumericCellValue();

                    //System.out.println("Contact Number : " + number);

                    set = (set + "Contact Number: " + number + ".\n");

                    break;


                  case 3:

                    String gender = detailcell.getStringCellValue();

                    //System.out.println("Gender : " + gender);

                    set = (set + "Gender: " + gender + ".\n");

                    break;


                }

            }


        }


        }

            return set;

        }

    }
```

```
//to execute k means algorithm to create clusters

package crm;

import java.io.File;
import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

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

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

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

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

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


public class K {

   public static String kexecute(String s) throws FileNotFoundException, IOException,
NullPointerException {


       String s1;

       System.out.println("hi!!");

       //try

      //{

          //Reading Excel file

         FileInputStream file = new FileInputStream(new File(s));

         HSSFWorkbook wbin = new HSSFWorkbook(file);

         HSSFSheet sheet = wbin.getSheetAt(1);


          //Creating new excel workbook

          HSSFWorkbook wbout = new HSSFWorkbook();


          //Creating new sheet for the output data

          HSSFSheet sheetout = wbout.createSheet("Sheet Cluster");
```

```
 //sample extraction
 HSSFCell cell = sheet.getRow(1).getCell(3);
 double age = cell.getNumericCellValue();
 System.out.println("First age is " + age);
 //HSSFCell cell = null;
//extract value of k from UI
  int k = 4 ;
  double c[] = new double [k];

  //for assigning initial centroid values
  System.out.println("Initial centroids: ");
  for(int i = 1;i<=k;i++)
  {
     cell = sheet.getRow(i).getCell(3);
     c[i-1] = cell.getNumericCellValue();
     System.out.print(c[i-1] + " ");
  }

  //for storing differences
  double d[] = new double [k]; //for storing differences
  int w=0;
  do{
  int num = 0;
  System.out.println();
   System.out.println("Here now!!");
  //JUMBO for loop
```

```
for(int i=(1);i<=400;i++)
{
   HSSFRow row = sheetout.createRow(num);
   int l=0;
   int col =0;
   //for determining differences
   for(int j =0;j<k;j++)
   {
   cell = sheet.getRow(i).getCell(3);
   d[j]=Math.abs(cell.getNumericCellValue()-c[j]);


   }
   double min = d[0];
   for(l=0;l<d.length;l++)
{
   if (d[l]<min)
   {
      min = d[l];
      col = l;
   }


}
   //extracted data has to be stored in corresponding cluster
   int start = 0;
   start = 5*col;
   int st = start;
```

```
 System.out.println("Start: " + start);



for(int m = 0;m<=4;m++)

{

   cell = sheet.getRow(i).getCell(m);


   //double s = cell.getNumericCellValue();

   //System.out.println("Value cell: " + s);

   //checking type of cell data

   switch (cell.getCellType())

   {

      case Cell.CELL_TYPE_NUMERIC:

         double no = cell.getNumericCellValue();

         System.out.println(no);

         row.createCell(start).setCellValue(no);

         start = start+1;

         break;


      case Cell.CELL_TYPE_STRING:


         String name = cell.getStringCellValue();

         System.out.println(name);

         row.createCell(start).setCellValue(name);

         start = start +1;

         break;

    }
```

```
             }

             //for filling buffer values in order to calculate new mean

            for(int z = 3;z<(k*5);z=z+5)


             {

                if(z!=st && z!=st+1 && z!=st+2 && z!=st+3 && z!=st+4)

                    row.createCell(z).setCellValue(0);

             }


           num = num+1;


           }

           s1 = "C:\\Users\\vibhati joshi\\Documents\\vibhati\\Database\\Output\\output.xls";

           System.out.println(s1+ "in kexecute");

            FileOutputStream fileOut = new FileOutputStream("C:\\Users\\vibhati
joshi\\Documents\\vibhati\\Database\\Output\\output.xls");

            wbout.write(fileOut);

            fileOut.close();


            System.out.println("Output file has been succesfully created");


            //computing new centroids

            int counter[] = new int[k];

            int cnew[] = new int[k];


            System.out.println("The new centroids and corresponding counters are: " );//to see new
centroids

            int p = 0;
```

```
for(int y = 3; y<k*5 ; y=y+5)
{
  for(int x = 0; x<400;x++)
  {
     cell = sheetout.getRow(x).getCell(y);
     if(cell.getNumericCellValue()!=0)
        counter[p]++;
     cnew[p] += (int)cell.getNumericCellValue();


  }
  cnew[p] = cnew[p]/counter[p];


  p++;


}
for(int o:cnew)
  System.out.print(o+"\t");
System.out.println();
for(int o:counter)
   System.out.print(o+"\t");



for(int u = 0;u<k;u++)
{
   if(cnew[u]==c[u])
   {
      w++;
```

```
                continue;
            }
            else
                c[u]=cnew[u];

        }
        System.out.println();
        System.out.println("Updated Centroids: ");
        for(int q=0;q<k;q++)
        {
            System.out.print(c[q] + " ");
        }
            /*if(w!=3)
            {
                System.out.println();
                System.out.println("inside if");

            }*/

    }while(w!=k);
            System.out.println("Centroids are matching!!");

            return s1;

    }
}
```

```
//main module for execution of apriori algorithm for generation of associations


package crm;


import java.io.*;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Vector;

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

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

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

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


public class AprioriExecute {


   public static void doApriori(int a1[], int x)throws IOException, FileNotFoundException,
NullPointerException  {

       /*

       System.out.println("Enter the number of fixed tables!!");

       BufferedReader br =  new BufferedReader(new InputStreamReader(System.in));

       int n = Integer.parseInt(br.readLine());


       System.out.println("Select the sheet numbers from below menu!!");

       System.out.println("1. Savings!");

       System.out.println("2. Home Loan!");
```

```
 System.out.println("3. FD!");

 System.out.println("4. Credit Card!");

 System.out.println("5. Car Loan!");

 */

 //array for storing sheet indices

 int a[] = new int[a1.length];

 a = a1;


/* for(int i=0;i<n;i++)

 {

   a[i] = Integer.parseInt(br.readLine());

 }*/


  System.out.println("The array of excel sheet indices is: ");

  a = sort(a);

  for(int i:a)

     System.out.print(i+"  ");

  System.out.println();


 // System.out.println("Enter the non fixed indices!!");

  int b[] = new int[10];

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

  {

     //b[i]= Integer.parseInt(br.readLine());

     b[i] = (i+1);

  }
```

```
        //call method to get intersection table of fixed sheets

        Transaction.intersect(a);

        //System.out.println("Program end!!");


        //for reading table apriori

        FileInputStream apfile = new FileInputStream(new File("C:\\Users\\vibhati
joshi\\Documents\\vibhati\\Database\\apriori.xls"));

        HSSFWorkbook apin = new HSSFWorkbook(apfile);

        HSSFSheet apsheet = null;


        //to acces intersection file for further transactional file

        FileInputStream apfile1 = new FileInputStream(new File("C:\\Users\\vibhati
joshi\\Documents\\vibhati\\Database\\Transaction\\transaction.xls"));

        HSSFWorkbook apin1 = new HSSFWorkbook(apfile1);

        HSSFSheet apsheet1 = apin1.getSheetAt(0);


        //for final transaction table in excel

        HSSFWorkbook apout2 = new HSSFWorkbook();

        HSSFSheet apsheet2 = apout2.createSheet("Transaction");


        int z=2;


        for(int i=2;i<apsheet1.getLastRowNum();i++)

        {

          int column =0;

          HSSFCell apcell1 = apsheet1.getRow(i).getCell(0);

          int compare = (int)apcell1.getNumericCellValue();
```

```
   HSSFRow aprow2 = apsheet2.createRow(z);

   aprow2.createCell(0).setCellValue(compare);

   column++;


for(int j=0;j<b.length;j++)

{

   if(Transaction.search(compare,b[j]))

   {

      apsheet = apin.getSheetAt(b[j]);

     HSSFCell apcell = apsheet.getRow(0).getCell(0);

     String table = apcell.getStringCellValue();

     /*switch (table)

     {

        case "SAVINGS":

           column = 1;

           break;


        case "HOME LOAN":

           column = 2;

           break;


        case "FD":

           column = 3;

           break;


        case "CREDIT CARD":

           column = 4;
           break;

        case "CAR LOAN":
           column = 5;
           break;

      case "EDUCATION LOAN":
           column = 6;
           break;

      case "PERSONAL LOAN":
           column = 7;
           break;

        case "RECURRING":
           column = 8;
           break;

      case "MUTUAL FUND":
           column = 9;
           break;

      case "INSURANCE":
           column = 10;
           break;
     }*/
     aprow2.createCell(column).setCellValue(table);
```

```
            column++;
          }
        }
        z++;
        }

        //Creation of final transaction database file in excel.
        FileOutputStream apfileOut = new FileOutputStream("C:\\Users\\vibhati
joshi\\Documents\\vibhati\\Database\\Transaction\\finaltransaction.xls");
        apout2.write(apfileOut);
        apfileOut.close();
        System.out.println("finaltransaction.xls has been created!!");
        //String transaction = "C:\\Users\\vibhati
joshi\\Documents\\NetBeansProjects\\Apriori\\Transaction\\finaltransaction.xls";



        //Calculating support count for the algorithm further.
        int last = apsheet1.getLastRowNum();
        System.out.println(last);
        int supp = Parameters.support(last-2,x);
        System.out.println("The support count for our transaction database is: " + supp);


        //to get frequencies
       ResultApriori r = new ResultApriori();
       ResultApriori rfinal = new ResultApriori();
       //r = Execute.count(supp);
       //r.display();
```

```
Vector v = new Vector();

for(int i=0;i<10;i++)

{

   apsheet = apin.getSheetAt(i+1);

   HSSFCell apcell = apsheet.getRow(0).getCell(0);

   v.add(i, apcell.getStringCellValue());

}

for(int i = 0; i<5;i++)

{

 rfinal = r;

 r = Execute.check(v, supp);

if(r.pass.size()!=0)

{

   System.out.println("Modified: ");

   r.display();

}

else

{

   r = rfinal;

   System.out.println("Final Vector: ");

   r.display();

   break;

}


//new support count

//supp = Parameters.support(r.pass.size());

//System.out.println("Revised support: " + supp);
```

```
    v = Execute.create(r,i);

    if(v.size()!=0)

    {

        System.out.println("Combinations: " + v);

    }

    else break;


    }



}


public static int[] sort(int s[])

{

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

    {

        for(int j=i+1;j<s.length;j++)

        {

            if(s[i]>s[j])

            {

                int temp=s[j];

                s[j]=s[i];

                s[i]=temp;

            }

        }

    }
```

```
        return s;
    }
}
```

//class Result for creation of result object to store pass and failed associations at every stage

```
package crm;
import java.util.Vector;

public class ResultApriori {

    Vector pass;
    Vector fail;

    ResultApriori()
    {
        pass = new Vector();
        fail = new Vector();
    }

    public void display()
    {
        apriori.jTextArea1.append(pass  + "\n");
        apriori.jTextArea2.append(fail + "\n");

    }
}
```

```
//calculating support count
package crm;


import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;


public class Parameters {


   public static int support(int l, int x)throws IOException
   {
      /*
      System.out.println("Enter the support % for association rules!!");
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int supp = Integer.parseInt(br.readLine());*/


      int supp;
      //System.out.println(supp);
      supp =(int)((x*(l))/100);
      return supp;


   }
```

```
//creating intermediate intersection and final transaction database
package crm;
```

```
import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

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

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

import com.lowagie.text.Document;

import com.lowagie.text.DocumentException;

import com.lowagie.text.Paragraph;

import com.lowagie.text.Phrase;

import com.lowagie.text.pdf.PdfPCell;

import com.lowagie.text.pdf.PdfPTable;

import com.lowagie.text.pdf.PdfWriter;

import java.io.FileOutputStream;

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

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

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

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


public class Conversion {


   public static String kconvert(String s)throws IOException, FileNotFoundException,
DocumentException, NullPointerException

   {

       FileInputStream file = new FileInputStream(new File(s));

       String s1 = s.replaceAll(".xls", ".pdf");

       int index = s.indexOf(".xls");
```

```
System.out.println("index : "+ index);

System.out.println("in conversion" + s1);

HSSFWorkbook wbin = new HSSFWorkbook(file);

//wbin.save("abc.pdf", SaveFormat.PDF);

HSSFSheet sheet = wbin.getSheetAt(0);

HSSFCell cell;

HSSFRow row;


Document doc = new Document();

PdfWriter.getInstance(doc, new FileOutputStream(s1));

doc.open();

System.out.println("pdf run");


//PdfPTable table = new PdfPTable(5);

PdfPCell tablecell;

int clusters =4;


for(int i =0; i<clusters;i++)

{

    int st = 5*i;

    doc.add(new Paragraph("Cluster " + (i+1) + "\n"));

   PdfPTable table = new PdfPTable(5);




   there:for(int j = 0;j<sheet.getLastRowNum();j++)

    {
```

```
    for(int k=st;k<(st+5);k++)
    {

      cell = sheet.getRow(j).getCell(k,Row.RETURN_BLANK_AS_NULL);
      if(cell==null)
        continue there;
      else
      {
        switch(cell.getCellType())
        {
          case Cell.CELL_TYPE_NUMERIC:
            int num = (int)cell.getNumericCellValue();
            String set = (" " + num + " ");
            tablecell = new PdfPCell(new Phrase(set));
            table.addCell(tablecell);
            break;

          case Cell.CELL_TYPE_STRING:

            tablecell = new PdfPCell(new Phrase(cell.getStringCellValue()));
            table.addCell(tablecell);
            break;
        }
      }
    }
  }
```

```
            doc.add(table);

        }

        doc.close();
        file.close();

        return s1;
    }

    public static String aconvert(String transaction)throws IOException, FileNotFoundException,
DocumentException, NullPointerException
    {

        FileInputStream file = new FileInputStream(new File(transaction));
        String pdf = transaction.replaceAll(".xls", ".pdf");

        HSSFWorkbook wbin = new HSSFWorkbook(file);
        HSSFSheet sheet = wbin.getSheetAt(0);
        HSSFCell cell;
        HSSFRow row;

        Document doc = new Document();
        PdfWriter.getInstance(doc, new FileOutputStream(pdf));
        doc.open();
        PdfPTable table = new PdfPTable(7);
        PdfPCell tablecell;
        doc.add(new Paragraph("Transaction Database:  \n"));
```

```
there: for(int i=2;i<sheet.getLastRowNum();i++)

{

  //PdfPTable table = new PdfPTable(6);
  here: for(int j=0;j<7;j++)
  {
    cell = sheet.getRow(i).getCell(j,Row.RETURN_BLANK_AS_NULL);

    if (cell==null)
     {
     table.addCell("");
      continue here;
     }
    else
    {

        switch(cell.getCellType())
        {
          case Cell.CELL_TYPE_NUMERIC:
            int num = (int)cell.getNumericCellValue();
            String set = (" " + num + " ");
            tablecell = new PdfPCell(new Phrase(set));
            table.addCell(tablecell);
            break;
```

```
                   case Cell.CELL_TYPE_STRING:

                       tablecell = new PdfPCell(new Phrase(cell.getStringCellValue()));
                       table.addCell(tablecell);
                       break;

                  /* case Cell.CELL_TYPE_BLANK:
                       table.addCell("");
                       break;*/
                }
            }

          }

          //doc.add(table);
          }

            doc.add(table);
            doc.close();
            file.close();

        return pdf;
    }
}
```

//creating combinations and checking associations for support count at each stage

```
package crm;


import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.util.Vector;

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

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

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

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


public class Execute {


    public static Vector create(ResultApriori rcreate, int z)

    {

        Vector v1 = new Vector();

        int count;


        if(z==0)

        {

        for(int i=0;i<rcreate.pass.size();i++)

        {

            for(int j=i+1;j<rcreate.pass.size();j++)

            {


                v1.addElement(rcreate.pass.elementAt(i).toString() + "-" +
rcreate.pass.elementAt(j).toString());
```

```
    }
}
}

else
{
    for(int i = 0;i<rcreate.pass.size();i++)
    {
        String temp1 = rcreate.pass.get(i).toString();
        String a1[] = temp1.split("-");

        for(int j = i+1; j<rcreate.pass.size();j++)
        {
            count = 0;
            String temp2 = rcreate.pass.get(j).toString();
            String a2[] = temp2.split("-");

            for(int k=0; k<z;k++)
            {
                if(a1[k].equals(a2[k]))
                {
                    count++;
                    continue;
                }
                else
                    break;
                }


                if(count==z)

              {

               v1.addElement(rcreate.pass.get(i).toString() + "-" + a2[z]);


              }

              }




            }

        }




        return v1;

    }


    public static ResultApriori check(Vector v, int s)throws IOException, FileNotFoundException

    {

        ResultApriori r = new ResultApriori();

        FileInputStream apfileOut = new FileInputStream(new File("C:\\Users\\vibhati
joshi\\Documents\\vibhati\\Database\\Transaction\\finaltransaction.xls"));

        HSSFWorkbook apout2 = new HSSFWorkbook(apfileOut);

        HSSFSheet apsheet2 = apout2.getSheetAt(0);

        HSSFRow aprow2 = null;

        HSSFCell apcell2 = null;


        String store[];
String temp;

int counter[] = new int[v.size()];

System.out.println("Store: ");

for(int i=0;i<v.size();i++)

{

   temp = v.get(i).toString();

   store = temp.split("-");

   counter[i]=Transaction.search(store);

}

System.out.println("Count: ");

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

{

   System.out.print(counter[i] + " ");

   if(counter[i]>=s)


   {

      r.pass.addElement(v.get(i));


   }

   else

   {

      r.fail.addElement(v.get(i));

   }

}

System.out.println();

return r;
```

```
    }

}
```

```
//converting output excel file to PDF

package crm;


import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

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

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

import com.lowagie.text.Document;

import com.lowagie.text.DocumentException;

import com.lowagie.text.Paragraph;

import com.lowagie.text.Phrase;

import com.lowagie.text.pdf.PdfPCell;

import com.lowagie.text.pdf.PdfPTable;

import com.lowagie.text.pdf.PdfWriter;

import java.io.FileOutputStream;

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

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

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

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

```
public class Conversion {


  public static String kconvert(String s)throws IOException, FileNotFoundException,
DocumentException, NullPointerException

  {

    FileInputStream file = new FileInputStream(new File(s));

    String s1 = s.replaceAll(".xls", ".pdf");

    int index = s.indexOf(".xls");

    System.out.println("index : "+ index);

    System.out.println("in conversion" + s1);

    HSSFWorkbook wbin = new HSSFWorkbook(file);

    //wbin.save("abc.pdf", SaveFormat.PDF);

    HSSFSheet sheet = wbin.getSheetAt(0);

    HSSFCell cell;

    HSSFRow row;


    Document doc = new Document();

    PdfWriter.getInstance(doc, new FileOutputStream(s1));

    doc.open();

    System.out.println("pdf run");


    //PdfPTable table = new PdfPTable(5);

    PdfPCell tablecell;

    int clusters =4;


    for(int i =0; i<clusters;i++)

    {

      int st = 5*i;
```

```
 doc.add(new Paragraph("Cluster " + (i+1) + "\n"));
PdfPTable table = new PdfPTable(5);



there:for(int j = 0;j<sheet.getLastRowNum();j++)
 {

    for(int k=st;k<(st+5);k++)
    {

      cell = sheet.getRow(j).getCell(k,Row.RETURN_BLANK_AS_NULL);
      if(cell==null)
         continue there;
      else
      {
         switch(cell.getCellType())
         {
            case Cell.CELL_TYPE_NUMERIC:
               int num = (int)cell.getNumericCellValue();
               String set = (" " + num + " ");
               tablecell = new PdfPCell(new Phrase(set));
               table.addCell(tablecell);
               break;

            case Cell.CELL_TYPE_STRING:

               tablecell = new PdfPCell(new Phrase(cell.getStringCellValue()));
```

```
                    table.addCell(tablecell);

                    break;

                }

              }

            }

          }


            doc.add(table);


        }


        doc.close();

        file.close();


        return s1;

    }


    public static String aconvert(String transaction)throws IOException, FileNotFoundException,
DocumentException, NullPointerException

    {


        FileInputStream file = new FileInputStream(new File(transaction));

        String pdf = transaction.replaceAll(".xls", ".pdf");


        HSSFWorkbook wbin = new HSSFWorkbook(file);

        HSSFSheet sheet = wbin.getSheetAt(0);

        HSSFCell cell;

        HSSFRow row;
```

```
Document doc = new Document();
PdfWriter.getInstance(doc, new FileOutputStream(pdf));
doc.open();
PdfPTable table = new PdfPTable(7);
PdfPCell tablecell;
doc.add(new Paragraph("Transaction Database:  \n"));

there: for(int i=2;i<sheet.getLastRowNum();i++)

{

   //PdfPTable table = new PdfPTable(6);
   here: for(int j=0;j<7;j++)
   {
      cell = sheet.getRow(i).getCell(j,Row.RETURN_BLANK_AS_NULL);

      if (cell==null)
       {
       table.addCell("");
        continue here;
       }



      else
      {
        switch(cell.getCellType())

        {

          case Cell.CELL_TYPE_NUMERIC:

            int num = (int)cell.getNumericCellValue();

            String set = (" " + num + " ");

            tablecell = new PdfPCell(new Phrase(set));

            table.addCell(tablecell);

            break;


          case Cell.CELL_TYPE_STRING:


            tablecell = new PdfPCell(new Phrase(cell.getStringCellValue()));

            table.addCell(tablecell);

            break;


         /* case Cell.CELL_TYPE_BLANK:

            table.addCell("");

            break;*/

        }

   }


}


//doc.add(table);

}
```

```
            doc.add(table);
             doc.close();
             file.close();


        return pdf;
    }
}
```