```
unit CSVSample;

interface

uses SysUtils, Classes, StrUtils, System.Character,
     System.Generics.Collections;

type
  TCSVSample = class
  public
    Filename, Column: string;
    SLColNames: TStringList;
    ExtractName: string;

    slValues: TStringList;
    slFreqs: TStringList;

    constructor Create;
    destructor Destroy; override;

    procedure readSample;
    procedure WriteCols;
  private
    procedure GetColNames;
    function GetDataValue(VarLabel: string; SLData: TStringList): string;
    function GetDataValues(VarLabels: string; SLData: TStringList): string;
  end;

implementation

{ TCSVSample }

constructor TCSVSample.Create;
begin
  SLColNames := TStringList.Create;
  SLColNames.Delimiter := ',';
  SLColNames.CaseSensitive := false;
  SLColNames.StrictDelimiter := true;

  // distinct list of column values
  slValues := TStringList.Create;
  slValues.Sorted := true;
  slValues.Duplicates := dupIgnore;
end;

destructor TCSVSample.Destroy;
begin
  SLColNames.Free;
  slValues.Free;
```

```
  inherited;
end;

procedure TCSVSample.GetColNames;
var
  Reader: TStreamReader;
  sLine: string;
begin
  Reader := TStreamReader(Filename);
  sLine := Reader.ReadLine;
  SLColNames.CommaText := sLine;
  Reader.Free;
end;

function TCSVSample.GetDataValue(VarLabel: string;
  SLData: TStringList): string;
begin
  Result := SLData.Strings[SLColNames.IndexOf(VarLabel)];
end;

function TCSVSample.GetDataValues(VarLabels: string;
  SLData: TStringList): string;
var
  SL: TStringList;
  i: integer;
  sValues: string;
begin
  SL := TStringList.Create;
  SL.Delimiter := ',';
  SL.StrictDelimiter := true;
  SL.CommaText := VarLabels;

  sValues := '';
  for i := 0 to SL.Count-1 do
    sValues := sValues + SLData.Strings[SLColNames.IndexOf(SL.Strings[i])] + ',';

  Result := Copy(sValues, 1, Length(sValues)-1);
  SL.Free;
end;

procedure TCSVSample.readSample;
var
  Reader: TStreamReader;
  iCnt: integer;
  S: String;
  slData: TStringList;
  i: integer;

  varname: string;
  vname: string;
```

```
  curval, rawval: double;

  sColName: string;
begin
  Reader := TStreamReader.Create(Filename);

  slData := TStringList.Create;
  SLData.Delimiter := ',';
  SLData.StrictDelimiter := true;

  iCnt := 0;

  while not Reader.EndOfStream do
  begin
    try
      S := Reader.ReadLine;
      inc(iCnt);
    except
      on E: exception do
      begin
        //ShowMessage(E.Message);
        //ShowMessage('nCnt: ' + nCnt.toString + ' S: ' + S);
        //Reader.Free;
      end;
    end;

    //if iCnt > 25 then break;

    if iCnt = 1 then
    begin
      slColNames.CommaText := S;
    end
    else
    begin
      slData.CommaText := S;

      if slData.Count <> slColNames.Count then
        raise Exception.Create('Mismatch data and column labels');

      slValues.Add(GetDataValues(Column, slData));
    end;
  end;

  // accumulate counts

  slData.Free;
  Reader.Free;
end;

procedure TCSVSample.WriteCols;
```

```
var
  Reader: TStreamReader;
  Writer: TStreamWriter;
  iCnt: integer;
  S, SOut: String;
  slData: TStringList;
  i: integer;
begin
  Reader := TStreamReader.Create(Filename);
  Writer := TStreamWriter.Create(ExtractName);

  slData := TStringList.Create;
  SLData.Delimiter := ',';
  SLData.StrictDelimiter := true;

  iCnt := 0;

  while not Reader.EndOfStream do
  begin
    try
      S := Reader.ReadLine;
      inc(iCnt);
    except
      on E: exception do
      begin
        //ShowMessage(E.Message);
        //ShowMessage('nCnt: ' + nCnt.toString + ' S: ' + S);
        //Reader.Free;
      end;
    end;

    //if iCnt > 25 then break;

    if iCnt = 1 then
    begin
      slColNames.CommaText := S;
      for i := 0 to slColNames.Count-1 do
      begin
        if (AnsiUpperCase(slColNames.Strings[i]) = 'PWSID') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'BASELINEPH_WOCCT') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'BASELINEPH_WOPH') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'NUM_PROXIES') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'SMALL_CORRECT') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'PP_OVERLAP_FIND_FIX')
        then
          SOut := SOut + slColNames.Strings[i] + ',';
      end;
      Writer.WriteLine(SOut);
    end
    else
```

```
    begin
      slData.CommaText := S;

      if slData.Count <> slColNames.Count then
        raise Exception.Create('Mismatch data and column labels');

      sOut := '';

      for i := 0 to slColNames.Count-1 do
      begin
        if (AnsiUpperCase(slColNames.Strings[i]) = 'PWSID') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'BASELINEPH_WOCCT') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'BASELINEPH_WOPH') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'NUM_PROXIES') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'SMALL_CORRECT') or
           (AnsiUpperCase(slColNames.Strings[i]) = 'PP_OVERLAP_FIND_FIX')
        then
          SOut := SOut + slData.Strings[i] + ',';
      end;

      Writer.WriteLine(SOut);
    end;
  end;

  slData.Free;
  Reader.Free;
  Writer.Free;
end;

end.
```