```
unit PWS_Sample_Bins;

interface

uses SysUtils, Classes, LCRGlobals, System.Generics.Collections;

type
  TPWSSampleBins = class
  public

    constructor Create;
    destructor Destroy; override;

    procedure ReadBinFile(IsBaseline: boolean; LSLLevel, SmallProxyLabel: string);
    function GetData(PWSId: string): TStringList;
    function GetDataValue(VarLabel: string; SLData: TStringList): string;
  private
    SLLabels: TStringList;
    SavedDataDict: TDictionary<string, string>;
    SLData: TStringList;
  end;

implementation

uses Dialogs;

{ TPWSSampleBins }

constructor TPWSSampleBins.Create;
begin
  SLLabels := TStringList.Create;
  SLLabels.Delimiter := ',';
  SLLabels.StrictDelimiter := true;
  SLLabels.CaseSensitive := false;

  SavedDataDict := TDictionary<string, string>.Create(200000);
  SLData := TStringList.Create;
  SLData.Delimiter := ',';
  SLData.StrictDelimiter := true;
end;

destructor TPWSSampleBins.Destroy;
begin
  SavedDataDict.Clear;
  SLLabels.Free;
  SavedDataDict.Free;
  SLData.Free;
```

```
  inherited;
end;

function TPWSSampleBins.GetData(PWSId: string): TStringList;
var
  cBin: string;
  sData: string;
begin
  if SavedDataDict.TryGetValue(PWSId, sData) then
  begin
    SLData.CommaText := sData;
    Result := SLData;
  end
  else
    Result := nil;
end;

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

procedure TPWSSampleBins.ReadBinFile(IsBaseline: boolean; LSLLevel, SmallProxyLabel:
string);
var
  Reader: TStreamReader;
  Writer: TStreamWriter;
  filename: string;

  iCnt: integer;
  S: string;
  PWSId: string;
begin
  iCnt := 0;

  if IsBaseline then
  begin
    filename := 'Baseline_Bins_' + LSLLevel + '_' + SmallProxyLabel + '.csv';
    if not FileExists(datapath + filename) then
      raise Exception.Create('Cannot find file: ' + datapath + filename);
  end
  else
  begin
    filename := 'Option_Bins_' + LSLLevel + '_' + SmallProxyLabel + '.csv';
    if not FileExists(datapath + filename) then
      raise Exception.Create('Cannot find file: ' + datapath + filename);
  end;
```

```
  SavedDataDict.Clear;

  Reader := TStreamReader.Create(datapath + filename);

  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 = 1 then
    begin
      SLLabels.CommaText := S;
    end
    else
    begin
      PWSId := Copy(S, 1, AnsiPos(',', S) - 1);

      try
        SavedDataDict.Add(PWSId, S);
      except
        on E: exception do
        begin
          ShowMessage(E.Message);
          ShowMessage('iCnt: ' + iCnt.toString + ' S: ' + S);
          SavedDataDict.Free;
        end;
      end;
    end;
  end;

  Reader.Free;
end;

end.
```