```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input: N x M matrix containing grades;                                      %
%        Rows = N students                                                    %
%        Cols = M assignments                                                 %
%                                                                             %
% Produces 2 plots in a single window.                                        %
% 'grades' is assumed to contains values that are already rounded.            %
%                                                                             %
% In this implementation we chose to not close the plots when generating      %
% new ones, to give the user an option to compare them if they desire to.     %
% It is possible to close all the figures open at the moment of the call      %
% of the function or only specific figures before generating new ones by      %
% implementing a piece of code that uses the command 'close'.                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function gradesPlot(grades)
%-----------------------------------------------------------------------------
%                            1 - Bar Plot
%-----------------------------------------------------------------------------

    % cnt is used to count the number of students for each of the 7
    % possible grades; g will be used for the x-axis; the mean grades for
    % each of the N students are stored in 'avg'
    cnt = zeros(1,7);
    g   = [-3 0 2 4 7 10 12];
    avg = computeFinalGrades(grades);

    % All the occurrences of the i-th grade are summed up and the result is
    % divided by the grade value itself, giving the number of occurrences;
    % 0 is a special case since dividing by 0 is not possible and the
    % sum of zeros would be 0 so that case is treated separately by simply
    % incrementing by 1 for every instance of 0
    cnt(1) = sum(avg(avg == g(1))) / g(1);
    cnt(2) = sum(avg(avg == 0) + 1);

    % for the rest of the elements of 'avg' the computation is the same so
    % a for-loop is used
    for i = 3:7
        cnt(i) = sum(avg(avg == g(i))) / g(i);
    end

    % A new figure is created everytime the function is called
    figure ('Name', 'Grades Plots');
    subplot(2 , 1 , 1);
    bar(g , cnt);

    % Aspect of the plot 1
    colormap(summer);
    title('Student Final Grades');
    grid on
    xlabel('Final Grades');
    xlim([-4 , 13]);
    ylabel('Number of Students');
    ylim([0 , (max(cnt) + 1)]);
    set(gca,'YTick',[0:1:max(cnt)]);
```

```
%-------------------------------------------------------------------------
%                     2 - Plot Grades vs Assignment
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%                 2.1 - Single Grades for each assignment
%-------------------------------------------------------------------------


    [N , M] = size(grades);

    % Considering M assignments one by one
    for i = 1:M

        % Extracting grades for the i-th assignment (i.e. i-th coloumn of
        % 'grades'); the vector 'x' will be used for the x-axis of the plot
        y = grades(: , i)';
        x = ones(1 , N) * i;

        % Convert 7-step-scale to 1-7-scale (NB the actual converting
        % scheme is shorter but more complicated to implement e.g. nested
        % conditions to control at every loop)
        y1(y == -3) = 1;
        y1(y == 0)  = 2;
        y1(y == 2)  = 3;
        y1(y == 4)  = 4;
        y1(y == 7)  = 5;
        y1(y == 10) = 6;
        y1(y == 12) = 7;

        % Grade frequency and computation of points to plot in case of
        % multiple grade occurrences.
        % NB 'cnt' can be used again at this point
        for i = 1:7

            % Grade Frequency
            cnt(i) = sum(y1(y1 == i)) / i;

            % if a grade occurs more than once a small random number is
            % added to both the x and y values but at least the first
            % occurrence is plotted as it is
            if (cnt(i) > 1)
                indeces = find(y1 == i);
                y(indeces(1)) = y(indeces(1));
                y(indeces(2:end))= y(indeces(2:end)) - 0.1 + 0.2*rand(1,(cnt(i)-1));
                x(indeces(2:end))= x(indeces(2:end)) - 0.1 + 0.2*rand(1,(cnt(i)-1));
            end

        end
        subplot(2 , 1 , 2);

        % Personal Note: plotting dots makes multiple grades barely visible
        % "plot(x, y,'*b')" produces a somewhat more clear result but
        % I don't know whether I am allowed to stray from this requirements
        % or not.
        plot(x , y , '.b');
        hold on
    end
```

```
%-------------------------------------------------------------------------
%               2.2 - Mean grade for each of the assignment
%-------------------------------------------------------------------------

    % Now 'avg' contains the mean grade of every coloumn, that is the mean
    % grade for each assignment.
    % NB In the case of 1 student (i.e. 'grades' is a row vector),
    % the mean-function would return a single number and the plot would not
    % draw a line between the single points, additionally in this case the
    % computation of the mean values is not even necessary! Therefore this
    % case is considered separately (when the condition in the the
    % if-clause true)

    if (length(avg) == 1)
        p = plot (1:M, grades , '-rs' , 'DisplayName' , 'Mean Grades');
    else
        avg = mean(grades);
        p = plot (1:M, avg, '-rs' , 'DisplayName' , 'Mean Grades');
    end

    % Aspect of the plots 1.2 and 2.2
    title('Grades vs Assignments');
    grid on
    legend(p , 'Mean Grades' , 'Location' , 'EastOutside');
    ylabel('Grades');
    ylim([-3.5 , 12.5]);
    set(gca , 'Ytick' , [-3:1:12]);
    xlabel('Assignments');
    xlim([0.5 , (M + 0.5)]);
    set(gca, 'Xtick' ,[0:1:M]);

end
```