const INPUT\_HARE = "晴れ";

const INPUT\_AME = "雨";

const INPUT\_KAMINARI = "雷";

const FORM\_SHEETNAME = "フォームの回答 1";

const RESULT\_SHEETNAME = "結果";

function performInspection() {

  try {

    var spreadsheet = SpreadsheetApp.getActiveSpreadsheet();

    var resultSheet = spreadsheet.getSheetByName(RESULT\_SHEETNAME);

    if (resultSheet === null) {

      resultSheet = spreadsheet.insertSheet(RESULT\_SHEETNAME);

    }

    var sheet1 = spreadsheet.getSheetByName(FORM\_SHEETNAME);

    var filterMode = sheet1.getFilter() !== null;

    if (!filterMode) {

      sheet1.getRange("A1:K").getDataRegion().createFilter();

    }

    resultSheet.clear();

    setResultsHeader(resultSheet);

    var dataRange = sheet1.getRange(4, 1, sheet1.getLastRow() - 1, sheet1.getLastColumn());

    var dataValues = dataRange.getValues();

    var startDate = new Date();

    // 1. 直近3回「雨」または「雷」が連続で選択された回答をフィルタリング

    var filteredData = filterRecentConsecutiveResponses(dataValues);

    writeFilteredData(resultSheet, filteredData, 1);

    // 2. 直近30日間の範囲内かつ「雨」または「雷」が5回以上選択された回答をフィルタリング

    var filtered30Data = filterRecentCountResponses(dataValues, startDate);

    writeFilteredData(resultSheet, filtered30Data, 2);

    // 3. 前回の回答は「晴れ」だったが今回は「雨」又は「雷」を選択した児童生徒

    var filteredHareData = filterPreviousHareResponses(dataValues);

    writeFilteredData(resultSheet, filteredHareData, 3);

    // 4. 前回体調がよかったのに悪くなってしまった子供の抽出

    var filteredCompData = filterDeterioratedResponses(dataValues);

    writeFilteredData(resultSheet, filteredCompData, 4);

  } catch (e) {

    Logger.log("エラーが発生しました: " + e.toString());

  }

}

function setResultsHeader(sheet) {

  var headerValues = [

    ["直近3日間、心の天気で「雨」又は「雷」を選択した児童生徒",

    "心の天気で「雨」又は「雷」を選択する傾向が強い児童生徒(直近30日間で5回以上)",

    "心の天気で前回の回答は「晴れ」だったが今回は「雨」又は「雷」を選択した児童生徒",

    "前回の回答から体調が大幅に悪化した児童生徒"]

  ];

  sheet.getRange(1, 1, 1, 4).setValues(headerValues);

  sheet.getRange("A:D").setWrap(true);

  sheet.setColumnWidths(1, 4, 300);

  sheet.setRowHeight(1, 34.2);

  sheet.getRange("A:D").setVerticalAlignment("middle");

  sheet.getRange("A1:D1").setFontWeight("bold");

  sheet.getRange("A1:D1").setBackground("#fff2cc");

}

function filterRecentConsecutiveResponses(dataValues) {

  var studentData = {};

  // 生徒データの作成

  dataValues.forEach(function(row) {

    var classname = row[2];

    var shusseki = row[3];

    var studentId = classname + '-' + shusseki; // 生徒のID

    var studentName = row[4];

    var timestamp = row[1]; // 日付

    var tenkiResponse = row[6]; // 回答がある列のインデックス

    if (!studentData.hasOwnProperty(studentId)) {

      studentData[studentId] = [];

    }

    studentData[studentId].push({

      timestamp: timestamp,

      response: tenkiResponse,

      studentName: studentName

    });

  });

  var filteredData = [];

  for (var studentId in studentData) {

    var submissions = studentData[studentId];

    var rainOrThunderCount = 0;

    var validSubmissions = [];

    // 日付の新しい順にソート

    submissions.sort(function(a, b) {

      return new Date(b.timestamp) - new Date(a.timestamp);

    });

    //for (var i = submissions.length - 1; i >= 0 ; i--) {

    for (var i = 0; i < submissions.length; i++) {

      var submission = submissions[i];

      if (submission.response === INPUT\_AME || submission.response === INPUT\_KAMINARI) {

        rainOrThunderCount++;

        validSubmissions.push(submission);

        if (rainOrThunderCount === 3) {

          filteredData.push([studentId, submission.studentName]);

          break;

        }

      }

    }

  }

  return filteredData;

}

function filterRecentCountResponses(dataValues, startDate) {

  startDate.setDate(startDate.getDate() - 29);

    var filteredData = dataValues.filter(function(row) {

      var timestamp = row[1];

      var tenkiResponse = row[6];

      var classname = row[2];

      var shusseki = row[3];

      var count = dataValues.filter(function(row) {

        return row[2] === classname && row[3] === shusseki;

      }).length;

      return timestamp >= startDate &&

        (tenkiResponse === INPUT\_AME || tenkiResponse === INPUT\_KAMINARI)

      && count >= 5;

    }).map(function(row) {

      return [row[2],row[3],row[4]];

    });

  return filteredData;

}

function filterPreviousHareResponses(dataValues) {

  var countByPerson = {};

    var filteredData = dataValues.filter(function(row) {

      var tenkiResponse = row[6];

      var classname = row[2];

      var shusseki = row[3];

      var key = classname + "-" + shusseki;

      if (!countByPerson[key]) {

        countByPerson[key] = {

          count: 1,

          previousResponse: tenkiResponse

        };

      } else {

        countByPerson[key].count++;

      }

      // 前回は「はれ」だったが今回は「あめ」または「かみなり」の場合に抽出

      if (

        countByPerson[key].count >= 2 &&

        (countByPerson[key].previousResponse.toString() === INPUT\_HARE) && (tenkiResponse === INPUT\_AME || tenkiResponse === INPUT\_KAMINARI)

      ) {

        countByPerson[key].previousResponse = tenkiResponse;

        return true;

      }

      countByPerson[key].previousResponse = tenkiResponse;

      return false;}).map(function(row) {

      return [row[2], row[3], row[4]];

    });

  return filteredData;

}

function filterDeterioratedResponses(dataValues) {

  var countByPersonComp = {};

    var filteredData = dataValues.filter(function(row) {

      var genkiresponse = row[5];

      var classname = row[2];

      var shusseki = row[3];

      var key = classname + "-" + shusseki;

      if (!countByPersonComp[key]) {

        countByPersonComp[key] = {

          count: 1,

          previousResponse: genkiresponse

        };

      } else {

        countByPersonComp[key].count++;

      }

      // 前回は「5」だったが今回は「1」の場合に抽出

      if (countByPersonComp[key].count >= 2 && countByPersonComp[key].previousResponse === 5 && genkiresponse === 1) {

        countByPersonComp[key].previousResponse = genkiresponse;

        return true; // 抽出対象としてフィルタリング

      }

      countByPersonComp[key].previousResponse = genkiresponse;

      return false;

    }).map(function(row) {

      return [row[2],row[3], row[4]];

    });

  return filteredData;

}

function writeFilteredData(sheet, filteredData, column) {

  if (filteredData.length > 0) {

    var uniqueData = [];

      filteredData.forEach(function(row) {

        var key = row[0] + '-' + row[1] + '-' + row[2];

        if (!uniqueData.includes(key)) {

          uniqueData.push(key);

        }

      });

      var uniqueFilteredData = uniqueData.map(function(key) {

        var info = key.split("-");

        return [info[0] + ' ' + info[1] + '番 ' + info[2]];

      });

      if (uniqueFilteredData.length > 0) {

        sheet.getRange(2, column, uniqueFilteredData.length, uniqueFilteredData[0].length).setValues(uniqueFilteredData);

      } else {

        Logger.log("抽出されたデータはありません。");

      }

    } else {

      Logger.log("該当する回答がありません。");

    }

}