```
// created on:  2024-03-21
// @author:     Jasper Heuer, based on Gyula Mate Kovács
// use:         collect and cloud mask Landsat 5/7/8/9 imagery
// comment:     2 geometries: mask_geometry is the outline of the glacier, export_geometry is the
//              extent of the study area

// get data ====================================================================================

// var dataset = ee.ImageCollection("LANDSAT/LT05/C02/T1_L2")
// var dataset = ee.ImageCollection("LANDSAT/LE07/C02/T1_L2")
// var dataset = ee.ImageCollection("LANDSAT/LC08/C02/T1_L2")
var dataset = ee.ImageCollection("LANDSAT/LC09/C02/T1_L2")
  .filterDate("2021-01-01", "2024-12-31")
  .filter(ee.Filter.calendarRange(8, 9, "month"))
  .filterBounds(mask_geometry); // filter by extent of glacier

print(dataset); // to check the number of images in unfiltered collection

// define functions ============================================================================

// define cloud function:
function createSnowMask(image) {
  var qa = image.select('QA_PIXEL'); // extract QA_PIXEL band

  // create masks for snow, cloud, and cloud shadow:
  var snowMask = qa.bitwiseAnd(1 << 5).neq(0).rename('snowmask');
  var cloudMask = qa.bitwiseAnd(1 << 3).neq(0).rename('cloudmask');
  var cloudShadowMask = qa.bitwiseAnd(1 << 4).neq(0).rename('shadowmask');

  // return image with the snow, cloud, and cloud shadow masks as bands:
  return image.addBands([snowMask, cloudMask, cloudShadowMask]).clip(export_geometry);
}

// define scaling function:
function applyScaleFactors(image) {
  // scale optical bands and thermal band:
  var opticalBands = image.select('SR_B.').multiply(0.0000275).add(-0.2);

  // add scaled bands to the image:
  return image.addBands(opticalBands, null, true)
              .clip(export_geometry);
}

// define cloud ratio function:
function cloudRatio(image) {
  // count cloud pixels:
  var count = image.select("cloudmask").reduceRegion({
    reducer: ee.Reducer.histogram(),
    geometry: mask_geometry, // only ratio above the glacier is of interest
    scale: 30,
    maxPixels: 1e10
  });

  // get histogram values:
  var histogram = count.get("cloudmask").getInfo();
  var ratio = null; // initialize ratio as null

  // handle cloud free/completely cloudy images (histrogram length = 1) issue:
  if (histogram !== null) { // check if histogram exists
    var vals = ee.List(count.get("cloudmask").getInfo()["histogram"]);
    // check if cloud free/completely cloudy and set ratio accordingly for special cases:
    if (vals.size().getInfo() === 1) {
      ratio = vals.get(0).getInfo() === 0 ? 1 : vals.get(0).getInfo() === vals.get(0).getInfo ? 0 : 0;
      print(ratio);
    } else {
      // calculate cloudiness ratio:
      var number_of_0_pixels = vals.get(0).getInfo(); // cloud free pixels
      var number_of_1_pixels = vals.get(1).getInfo(); // cloudy pixels
      ratio = number_of_1_pixels / (number_of_1_pixels + number_of_0_pixels);
    }
  }

  // set cloud ratio as image property:
  return image.set("CLOUD_RATIO", ratio);
}

```

```
// apply masking ═══════════════════════════════════════════════════════════════

var dataset = dataset.map(applyScaleFactors).map(createSnowMask); // scale imagery

// visualize imagery ═══════════════════════════════════════════════════════════

var visualization = {
  bands: ['SR_B3', 'SR_B2', 'SR_B1'],
  min: 0.0,
  max: 0.5,
};

// visualize true color image, snow mask and cloud mask:
Map.addLayer(dataset.first(), visualization, 'True Color');
Map.addLayer(dataset.first().select("snowmask"), {min: 0, max: 1, palette:['black', 'white']}, 'Snow Mask');
Map.addLayer(dataset.first().select("cloudmask"), {min: 0, max:1, palette:['black', 'red']}, 'Cloud Mask');

// count cloud pixels ══════════════════════════════════════════════════════════

// create list of images:
var n_img = dataset.size().getInfo();
var image_list = dataset.toList(n_img);
var image_list2 = ee.List([]); // create empty list to store images with cloud ratio

// loop through images:
for(var i = 0; i < n_img; i++) {
  var image_i = ee.Image(image_list.get(i));
  var with_ratio = cloudRatio(image_i);
  image_list2 = image_list2.add(with_ratio);
}

// convert list of cloud ratio imagery into collection:
var cloudratio_dataset = ee.ImageCollection.fromImages(image_list2);

// filter collection by cloud ratio ⩽ 0.1:
var filtered_dataset = cloudratio_dataset.filter(ee.Filter.lte("CLOUD_RATIO", 0.1));

print(filtered_dataset); // check number of images again

// export imagery ══════════════════════════════════════════════════════════════

var id = filtered_dataset.aggregate_array("system:index");

id.evaluate(function(list){
  list.map(function(id){
    var image = filtered_dataset.filter(ee.Filter.eq("system:index", id)).first();
    var mask = filtered_dataset.filter(ee.Filter.eq("system:index", id)).first()
      .select("cloudmask").lt(1); // reverse mask values
    var masked_img = image.updateMask(mask);

    Export.image.toDrive({
      image: masked_img.select(["SR_B1", "SR_B2", "SR_B3", "SR_B4", "SR_B5", "SR_B7"]),
      scale: 30,
      region: export_geometry,
      crs: "EPSG:4326",
      maxPixels: 1e13,
      folder: "MITTIVAKKAT_cloud_mask", // change output folder here, if needed
      description: id,
      formatOptions: {cloudOptimized: true}
    });
  });
});

```