```
/*
*       Google sheets script to comb through ticket archives for all of the Urgent status
tickets from 2015 to current
*       This accesses the FreshDesk API using GET to retrieve ticket data, then outputs
to a Google Sheet based on certain
*       criteria being met (time frames, dates, ticket status, etc.)
*/

function sheetOutput(ticket_count, id, created_at, subject) {
    // Spreadsheet output information
    var ss = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = ss.getSheets()[0];
    var row = ticket_count + 1;
    var column_ID = 6;
    var column_created_at = 8;
    var column_subject = 9;

    sheet.getRange(row, column_ID).setValue(id);
    sheet.getRange(row, column_created_at).setValue(created_at);
    sheet.getRange(row, column_subject).setValue(subject);

    return;
}

function getTickets() {
   // authentication and header information for FreshDesk API
   var API_KEY = '**********';  // Jake's FreshDesk API key
   var headers = {'Content-type': 'application/json','Authorization': 'Basic ' +
Utilities.base64Encode(API_KEY + ':X')};
   var ENDPOINT = 'https://modolabs.freshdesk.com/api/v2';

   // used for full ticket search count comparison and to calculate total results for
computing total pages
   var query ='query=' +                        // start of query string construction

            encodeURIComponent("\"") +        // encoded start " symbol for UrlFetchApp

            'priority:4' +                    // priority string for query

            encodeURIComponent(" AND ") +     // encoded ' AND ' (with spaces) for
UrlFetchApp()

            'created_at:' +                   // create_by string for query

            encodeURIComponent(">") +         // encoded '>' symbol for UrlFetchApp()

            "'2014-01-01'" +                  // date string (prior to 1st ticket) for query
```

```
            encodeURIComponent("\"");        // encoded end " symbol for UrlFetchApp()

  var url = ENDPOINT + '/search/tickets?' + query;
  var options = {'method': 'get', muteHttpExceptions: true,'headers': headers};
  var response = UrlFetchApp.fetch(url, options);
  var data = JSON.parse(response);
  //Logger.log(query);
  //Logger.log("Total priority 4 tickets since 2014-01-01: " + data.total);

  var total_tickets = data.total;    // count of all priority 4 tickets since 2014
  var ticket_count = 0;              // tracks total tickets that meet all criteria in loops below
  var per_page = 30;                 // default per_page value returned by API used for page
count calculation
  var true_count = 0;                // used to cycle through 30 tickets per page and compare
to last partial page count
  var pages = parseInt((total_tickets / per_page),10);

  if (total_tickets % per_page != 0)
  {
    pages += 1; // add 1 for last partial page
  }

 // cycle through year 2015 to 2019 (current)
  for (var query_date = 2015; query_date <= 2019; query_date++)
  {
     // reset per cycle/year
     total_tickets = 0;

     // 1st half year query cycle - searches priority 4 tickest between 1-1-2015 to
7-1-2015
     var query1 = 'query=' +
           encodeURIComponent("\"") +
           'priority:4' +
           encodeURIComponent(" AND ") +
           '(created_at:' +
           encodeURIComponent(">") +
           encodeURIComponent("\"") +
             + query_date + '-01-01' +  // uses 00:00:00
           encodeURIComponent("\"") +
           encodeURIComponent(" AND ") +
           'created_at:' +
           encodeURIComponent("<") +
           encodeURIComponent("\"") +
             + query_date + '-07-01' +  // uses 23:59:59
           encodeURIComponent("\"") +
           ")" +
```

```
          encodeURIComponent("\"");

    var url1 = ENDPOINT + '/search/tickets?' + query1;
    response = UrlFetchApp.fetch(url1, options);
    data = JSON.parse(response);

    // used in page/per page calculations and looping (t and i loops)
    total_tickets = data.total;
    per_page = 30;
    var pg_count = per_page;
    pages = parseInt((total_tickets / per_page),10);
    true_count += data.total;

    // used to calculate last, partial page if applicable
    if (total_tickets % per_page != 0)
    {
      pages += 1; // add 1 for last partial page
    }

    // loops through each page of results returned per 6 month period (query1, 1st half
year)
    for (var t = 1; t <= pages; t++)
    {
      url1 = ENDPOINT + '/search/tickets?page=' + t + "&" + query1;
      response = UrlFetchApp.fetch(url1, options);
      data = JSON.parse(response);

      // used to compute the total tickets on last partial page
      if (t == pages)
      {
        pg_count = total_tickets - ((pages - 1) * per_page);
      }

      // loops through 'per_page' ticket count on each page (30 default, last page is
partial)
      for (var i = 0; i < pg_count; i++)
      {
        true_count += 1;
        var date = new Date(data.results[i].created_at);
        var check_day = date.getDay();
        var AM_check = new Date(date.getYear(), date.getMonth(), date.getDate(),
"09","00","00");
        var PM_check = new Date(date.getYear(), date.getMonth(), date.getDate(),
"17","59","59");

        // captures Sunday tickets
        if (date.getDay() == 0)
```

```
        {
            ticket_count += 1;
            sheetOutput(ticket_count, data.results[i].id, date, data.results[i].subject);
        }
        // captures Saturday tickets
        else if (date.getDay() == 6)
        {
            ticket_count += 1;
            sheetOutput(ticket_count, data.results[i].id, date, data.results[i].subject);
        }
        // captures M-F tickets created before 9AM
        else if (date <= AM_check)
        {
            ticket_count += 1;
            sheetOutput(ticket_count, data.results[i].id, date, data.results[i].subject);
        }
       // captures M-F tickets created after 6PM
        else if (date >= PM_check)
        {
            ticket_count += 1;
            sheetOutput(ticket_count, data.results[i].id, date, data.results[i].subject);
        }
      }
    }

// Same setup as query1 above used for 2nd half of year, query2

    // 2nd half year query cycle
    var query2 = 'query=' +
          encodeURIComponent("\"") +
          'priority:4' +
          encodeURIComponent(" AND ") +
          '(created_at:' +
          encodeURIComponent(">") +
          encodeURIComponent("\"") +
            + query_date + '-07-02' +  // uses 00:00:00
          encodeURIComponent("\"") +
          encodeURIComponent(" AND ") +
          'created_at:' +
          encodeURIComponent("<") +
          encodeURIComponent("\"") +
            + (1 + query_date) + '-01-01' +  // uses 23:59:59
          encodeURIComponent("\"") +
          ")" +
          encodeURIComponent("\"");

    var url2 = ENDPOINT + '/search/tickets?' + query2;
```

```
    response = UrlFetchApp.fetch(url2, options);
    data = JSON.parse(response);

    total_tickets = data.total;
    per_page = 30;
    var pg_count = per_page;
    pages = parseInt((total_tickets / per_page),10);
    true_count += data.total;

    if (total_tickets % per_page != 0)
    {
      pages += 1; // add 1 for last partial page
    }

    for (var t = 1; t <= pages; t++)
    {
      url2 = ENDPOINT + '/search/tickets?page=' + t + "&" + query2;
      response = UrlFetchApp.fetch(url2, options);
      data = JSON.parse(response);

      if (t == pages)
      {
        pg_count = total_tickets - ((pages - 1) * per_page);
      }

      for (var i = 0; i < pg_count; i++)
      {
        true_count += 1;
        var date = new Date(data.results[i].created_at);
        var check_day = date.getDay();
        var AM_check = new Date(date.getYear(), date.getMonth(), date.getDate(),
"09","00","00");
        var PM_check = new Date(date.getYear(), date.getMonth(), date.getDate(),
"18","00","00");

        if (date.getDay() == 0)
        {
          ticket_count += 1;
          sheetOutput(ticket_count, data.results[i].id, date, data.results[i].subject);
        }
        else if (date.getDay() == 6)
        {
          ticket_count += 1;
          sheetOutput(ticket_count, data.results[i].id, date, data.results[i].subject);
        }
        else if (date <= AM_check)
        {
```

```
            ticket_count += 1;
            sheetOutput(ticket_count, data.results[i].id, date, data.results[i].subject);
          }
          else if (date >= PM_check)
          {
            ticket_count += 1;
            sheetOutput(ticket_count, data.results[i].id, date, data.results[i].subject);
          }
        }
      }
    }
    // used for validation purposes
    //Logger.log("Total On Call, priority 4 tickets: " + ticket_count);
};



/* Commented out Ticket fields for log simplicity

        Logger.log("cc_emails: " + data.results[i].cc_emails);
        Logger.log("fwd_emails: " + data.results[i].fwd_emails);
        Logger.log("reply_cc_emails:" + data.results[i].reply_cc_emails);
        Logger.log("ticket_cc_emails:" + data.results[i].ticket_cc_emails);
        Logger.log("fr_escalated: " + data.results[i].fr_escalated);
        Logger.log("spam: " + data.results[i].spam);
        Logger.log("email_config_id: " + data.results[i].email_config_id);
        Logger.log("group_id: " + data.results[i].group_id);
        Logger.log("priority: " + data.results[i].priority);
        Logger.log("requester_id: " + data.results[i].requester_id);
        Logger.log("responder_id: " + data.results[i].responder_id);
        Logger.log("source: " + data.results[i].source);
        Logger.log("company_id: " + data.results[i].company_id);
        Logger.log("status: " + data.results[i].status);
        Logger.log("subject: " + data.results[i].subject);
        Logger.log("association_type: " + data.results[i].association_type);
        Logger.log("to_emails: " + data.results[i].to_emails);
        Logger.log("product_id: " + data.results[i].product_id);
        Logger.log("id: " + data.results[i].id);
        Logger.log("type: " + data.results[i].type);
        Logger.log("due_by: " + data.results[i].due_by);
        Logger.log("fr_due_by: " + data.results[i].fr_due_by);
        Logger.log("is_escalated: " + data.results[i].is_escalated);
        Logger.log("description: " + data.results[i].description);
        Logger.log("description_text: " + data.results[i].description_text);
        Logger.log("custom_fields: phase: " + data.results[i].custom_fields.phase);
        Logger.log("               : platform: " + data.results[i].custom_fields.platform);
        Logger.log("               : device_os_version: " +
```

```
data.results[i].custom_fields.device_os_version);
        Logger.log("            : resolve_ticket_after_7_days: " +
data.results[i].custom_fields.resolve_ticket_after_7_days);
        Logger.log("            : steps_to_reproduce: " +
data.results[i].custom_fields.steps_to_reproduce);
        Logger.log("            : environment_test_production_admin: " +
data.results[i].custom_fields.environment_test_production_admin);

//        custom_fields.function is an invalid name for java/google scripting
//        Logger.log("             : function: " + data.results[i].custom_fields.function);

        Logger.log("            : audience_internal_only_or_external: " +
data.results[i].custom_fields.audience_internal_only_or_external);
        Logger.log("            : additional_documentation_required_in_support_center: " +
data.results[i].custom_fields.additional_documentation_required_in_support_center);
        Logger.log("created_at: " + data.results[i].created_at);
        Logger.log("updated_at: " + data.results[i].updated_at);
        Logger.log("associated_tickets_count: " +
data.results[i].associated_tickets_count);
        Logger.log("tags: " + data.results[i].tags);
*/
//        }
//    }
```