Search Query Report



A good source of both negative and positive keywords is the [Search Query Report](https://support.google.com/adwords/answer/2472708). This report shows the actual search queries that trigger your ads. For example, if your broad match keyword is "pet store", your ads may show when users search for "bad pet store". This type of negative search is not something you want to advertise for. Negative exact keywords allow you to exclude undesired search queries. By looking at performance of various search terms, you can decide which search terms to exclude.

The script uses the [Search Query Performance Report](https://developers.google.com/adwords/api/docs/appendix/reports#search-query) to find undesired search terms and add them as negative (or positive) exact keywords.

How it works

**Request a report with AWQL**

You can request a report using [AWQL](https://developers.google.com/adwords/api/docs/guides/awql) like this:

var report = AdWordsApp.report(  
   "SELECT Query,Clicks,Cost,Ctr,ConversionRate,CostPerConversion,Conversions,CampaignId,AdGroupId " +  
   " FROM SEARCH\_QUERY\_PERFORMANCE\_REPORT " +  
   " WHERE " +  
       " Conversions > 0" +  
       " AND Impressions > " + IMPRESSIONS\_THRESHOLD +  
       " AND AverageCpc > " + AVERAGE\_CPC\_THRESHOLD +  
   " DURING LAST\_7\_DAYS");  
var rows = report.rows();

When you start iterating over the rows in the report, AdWords scripts servers will download the data corresponding to the report you asked for and store it temporarily. As you iterate over the rows, more information is sent to your script.

In the above AWQL, we're requesting the Search Query Performance Report (SEARCH\_QUERY\_PERFORMANCE\_REPORT) and asking for a number of performance fields with data for the last seven days.

**Making decisions with report data**

Next, the script needs to iterate over the rows in the report and decide what to do with our report data:

while(rows.hasNext()) {  
  var row = rows.next();  
  if (parseFloat(row['Ctr']) < CTR\_THRESHOLD) {  
    addToMultiMap(negativeKeywords, row['AdGroupId'], row['Query']);  
    allAdGroupIds[row['AdGroupId']] = true;  
  } else if (parseFloat(row['CostPerConversion']) < COST\_PER\_CONVERSION\_THREADHOLD) {  
    addToMultiMap(positiveKeywords, row['AdGroupId'], row['Query']);  
    allAdGroupIds[row['AdGroupId']] = true;  
  }  
}

Each row in the report is a JavaScript object—an [associative array](https://en.wikipedia.org/wiki/Associative_array) where the key is the field name you requested and the value is a string representation of that field's value. The script then parses the field's value (which will always be a string) into a number and compares it to the CTR\_THRESHOLD. If a keyword's Ctr is below the threshold, the keyword is considered poor and added to the negative keywords list (to be created later). We also record the AdGroupId in a separate object so we can load the ad groups in bulk.

We also keep track of keywords that are above our Ctr threshold and have a reasonable CostPerConversion—these will be added as positive keywords.

**Loading ad groups in bulk**

Using the AdGroupIds we recorded earlier, we can load all the ad groups in a single request:

var adGroupIdList = [];  
for (var adGroupId in allAdGroupIds) {  
  adGroupIdList.push(adGroupId);  
}  
  
var adGroups = AdWordsApp.adGroups().withIds(adGroupIdList).get();

This is significantly more efficient than loading them one-by-one as we see a keyword to add. We strongly encourage the use of IDs to load objects in a batch when working with reports.

**Adding keywords to ad groups**

Next we iterate over all the ad groups (that we loaded in bulk) and add the appropriate keywords:

if (negativeKeywords[adGroup.getId()]) {  
  for (var i = 0; i < negativeKeywords[adGroup.getId()].length; i++) {  
    adGroup.createNegativeKeyword('[' + negativeKeywords[adGroup.getId()][i] + ']');  
  }  
}

This code checks to see if we have negative keywords to add for this group, loops over each one and adds it as an [exact match](https://support.google.com/adwords/answer/2497836) negative keyword to make sure we won't show ads if this exact text is searched for.

Setup

* Create a new script with the source code below.
* When Previewing, the script will probably be able to complete within the 30 second preview window. If not, try changing the condition DURING LAST\_7\_DAYS to DURING YESTERDAY.
* After the script is done previewing, you'll see a list of keywords that would be added as negative (and in some cases positive) keywords.

[Creating an AdWords script](https://developers.google.com/adwords/scripts/docs/solutions/search-query)

Source code

// Minimum number of impressions to consider "enough data"  
var IMPRESSIONS\_THRESHOLD = 100;  
// Cost-per-click (in account currency) we consider an expensive keyword.  
var AVERAGE\_CPC\_THRESHOLD = 1; // $1  
// Threshold we use to decide if a keyword is a good performer or bad.  
var CTR\_THRESHOLD = 0.5; // 0.5%  
// If ctr is above threshold AND our conversion cost isn’t too high, it’ll become a positive keyword.  
var COST\_PER\_CONVERSION\_THRESHOLD = 10; // $10  
  
function main() {  
  var report = AdWordsApp.report(  
      "SELECT Query,Clicks,Cost,Ctr,ConversionRate,CostPerConversion,Conversions,CampaignId,AdGroupId " +  
      " FROM SEARCH\_QUERY\_PERFORMANCE\_REPORT " +  
      " WHERE " +  
          " Conversions > 0" +  
          " AND Impressions > " + IMPRESSIONS\_THRESHOLD +  
          " AND AverageCpc > " + AVERAGE\_CPC\_THRESHOLD +  
      " DURING LAST\_7\_DAYS");  
  var rows = report.rows();  
  
  var negativeKeywords = {};  
  var positiveKeywords = {};  
  var allAdGroupIds = {}  
  // Iterate through search query and decide whether to  
  // add them as positive or negative keywords (or ignore).  
  while (rows.hasNext()) {  
    var row = rows.next();  
    if (parseFloat(row['Ctr']) < CTR\_THRESHOLD) {  
      addToMultiMap(negativeKeywords, row['AdGroupId'], row['Query']);  
      allAdGroupIds[row['AdGroupId']] = true;  
    } else if (parseFloat(row['CostPerConversion']) < COST\_PER\_CONVERSION\_THRESHOLD) {  
      addToMultiMap(positiveKeywords, row['AdGroupId'], row['Query']);  
      allAdGroupIds[row['AdGroupId']] = true;  
    }  
  }  
  
  // Copy all the adGroupIds from the object into an array.  
  var adGroupIdList = [];  
  for (var adGroupId in allAdGroupIds) {  
    adGroupIdList.push(adGroupId);  
  }  
  
  // Add the keywords as negative or positive to the applicable ad groups.  
  var adGroups = AdWordsApp.adGroups().withIds(adGroupIdList).get();  
  while (adGroups.hasNext()) {  
    var adGroup = adGroups.next();  
    if (negativeKeywords[adGroup.getId()]) {  
      for (var i = 0; i < negativeKeywords[adGroup.getId()].length; i++) {  
        adGroup.createNegativeKeyword('[' + negativeKeywords[adGroup.getId()][i] + ']');  
      }  
    }  
    if (positiveKeywords[adGroup.getId()]) {  
      for (var i = 0; i < positiveKeywords[adGroup.getId()].length; i++) {  
        adGroup.createKeyword('[' + positiveKeywords[adGroup.getId()][i] + ']');  
      }  
    }  
  }  
}  
  
function addToMultiMap(map, key, value) {  
  if (!map[key]) {  
    map[key] = [];  
  }  
  map[key].push(value);  
}