

# Automated rules & scripts on Google Ads



Avishay Freund  
Internet Marketing Expert  
[avishay@logos.co.il](mailto:avishay@logos.co.il)



# About me



- Owner of Logos Online Marketing
- Expert on online marketing and SEM/SEO
- Consulting on online marketing to the private and public sector
- Certified Google Advertising Professional
- Senior lecturer at colleges and universities on online marketing and web technologies
- Expert witness on online advertising
- MBA graduate, Ben Gurion University



# **Automated Rules**

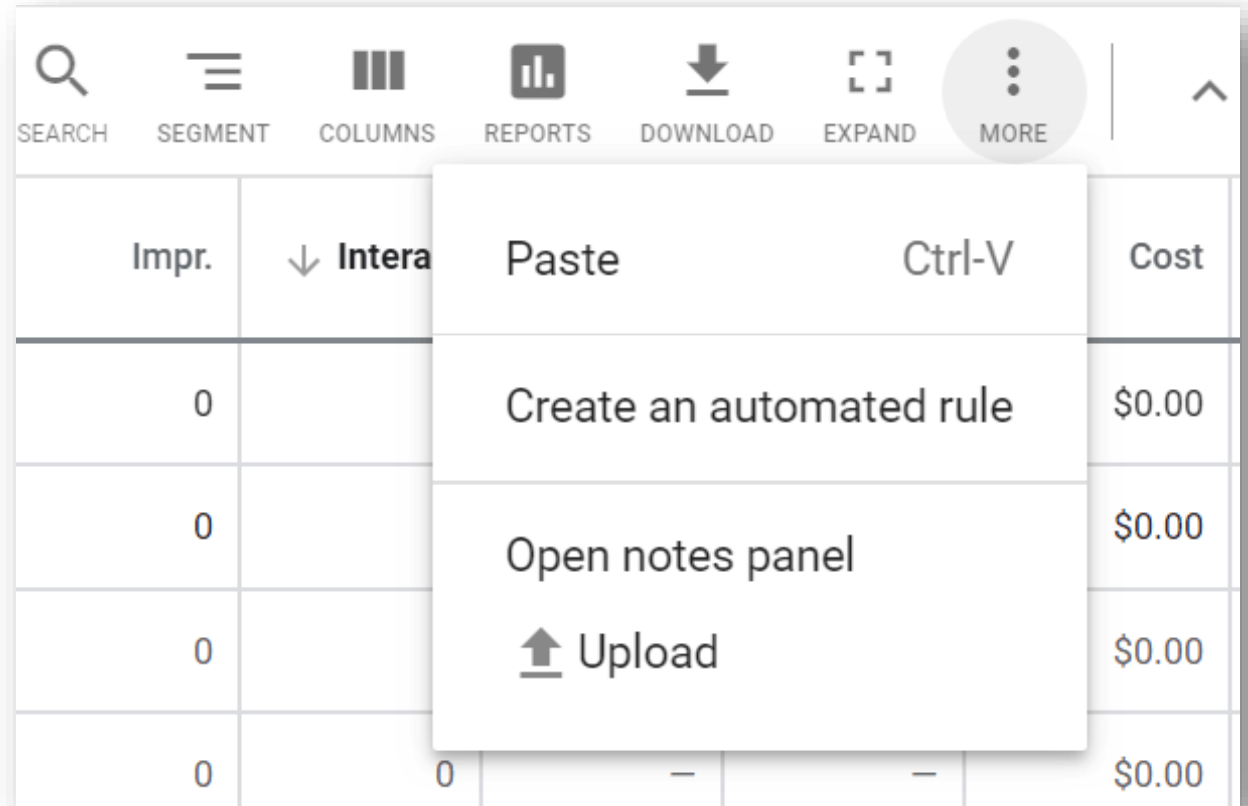
# Automated Rules



- Run automated actions across your account
- Can be scheduled to run daily, weekly or monthly
- Examples:
  - Pause campaigns on a certain spend cap (e.g. 6,000\$)
  - Adjust bids by 15% for keywords under position 4
  - Daily email report on keywords with a low quality score

# Creating a rule

- Navigate to Campaign, Ad Group or Keywords screen
- Select the relevant keywords (one or more)
- Click the three dots to open the menu
- Select "Create an automated rule"



The screenshot shows a mobile application interface with a table. The table has columns for 'Impr.', 'Intera', and 'Cost'. A context menu is open over the table, showing options like 'Paste', 'Create an automated rule', 'Open notes panel', and 'Upload'. The 'MORE' button in the top right corner is highlighted.

Impr.	Intera	Cost
0		\$0.00
0		\$0.00
0		\$0.00
0	0	\$0.00

# Rule #1: Pausing campaigns at a set budget

- This rule will check daily if the account cap of \$6K was met and if so will pause all active campaigns

Create a new rule for campaigns

Type of rule  
Pause campaigns ▼

Apply to campaigns ?

All enabled campaigns  
 All enabled and paused campaigns  
 Select campaigns

Condition ?

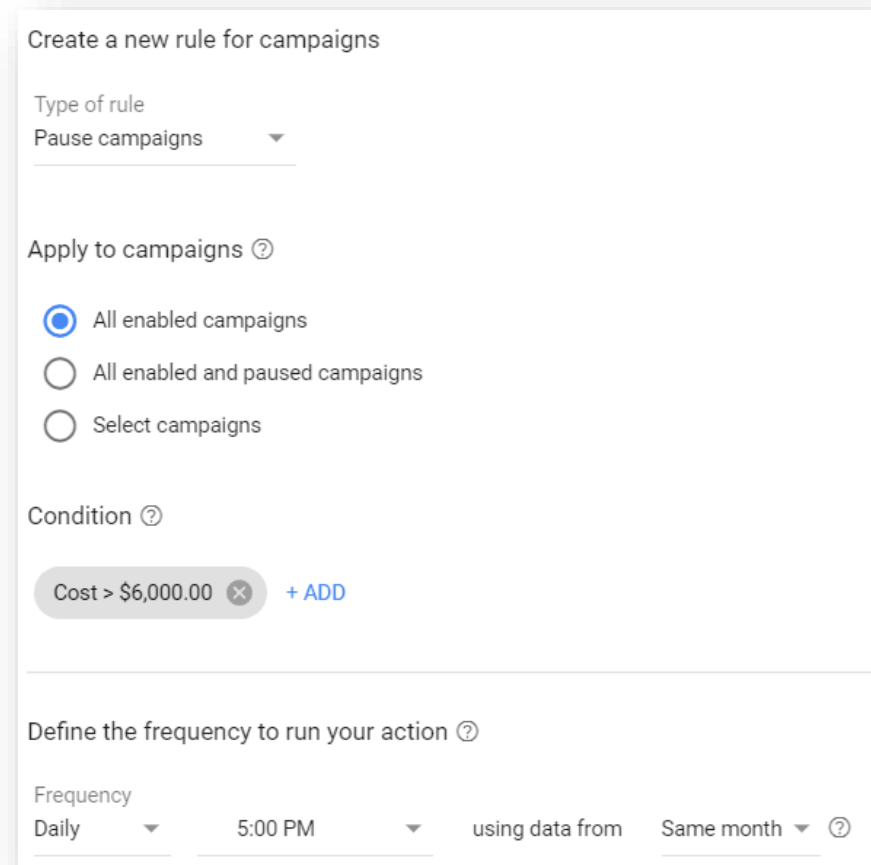
Cost > \$6,000.00 × + ADD

Define the frequency to run your action ?

Frequency  
Daily ▼ 5:00 PM ▼ using data from Same month ▼ ?

# Rule #1: Pausing campaigns at a set budget

- The type of rule is "Pause campaigns"
- You can select to which campaigns this applies – a specific set or all
- The condition is based on the Cost field
- Set the frequency and the lookback window for the Cost data



Create a new rule for campaigns

Type of rule  
Pause campaigns

Apply to campaigns ?

All enabled campaigns  
 All enabled and paused campaigns  
 Select campaigns

Condition ?

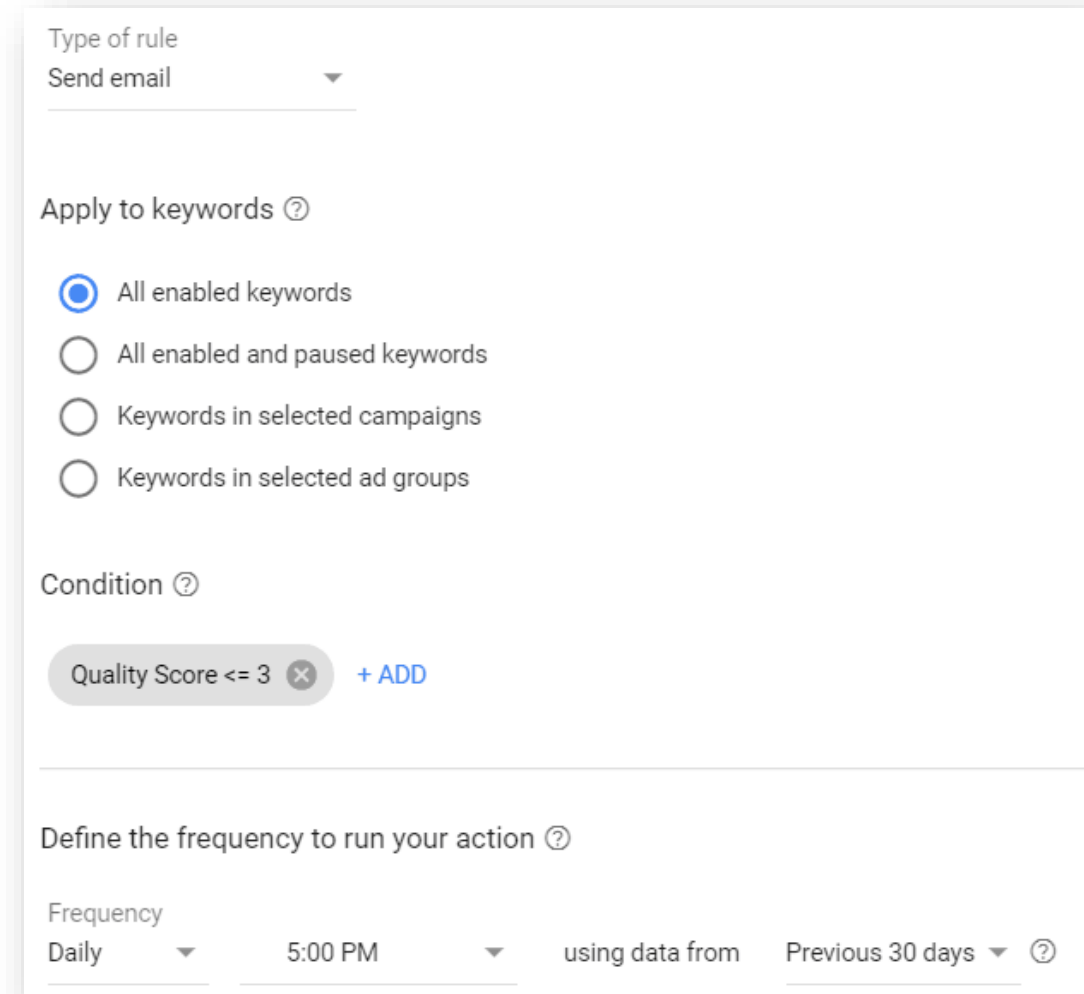
Cost > \$6,000.00

Define the frequency to run your action ?

Frequency  
Daily  5:00 PM  using data from Same month  ?

# Rule #2: Email alerts on low Quality Score

- The type of rule is "Pause campaigns"
- I recommend selecting all keywords
- The condition is Quality Score equal or lower than 3



The screenshot shows the configuration interface for a rule in Google Ads. The rule type is set to "Send email". The rule applies to "All enabled keywords". The condition is "Quality Score <= 3". The frequency is set to "Daily" at "5:00 PM" using data from the "Previous 30 days".

Type of rule  
Send email

Apply to keywords ?

- All enabled keywords
- All enabled and paused keywords
- Keywords in selected campaigns
- Keywords in selected ad groups

Condition ?

Quality Score <= 3 × + ADD

Define the frequency to run your action ?

Frequency  
Daily 5:00 PM using data from Previous 30 days ?



# Notes on Automated Rules



- Sometimes you will want to set a rule to run twice daily – this will require setting up two separate rules
- Rules can be created on the account or MCC level
- Not every scenario can be addressed using a rule, that's what we have Google Ads Scripts for



# Google Ads Scripts

# Google Ads Scripts

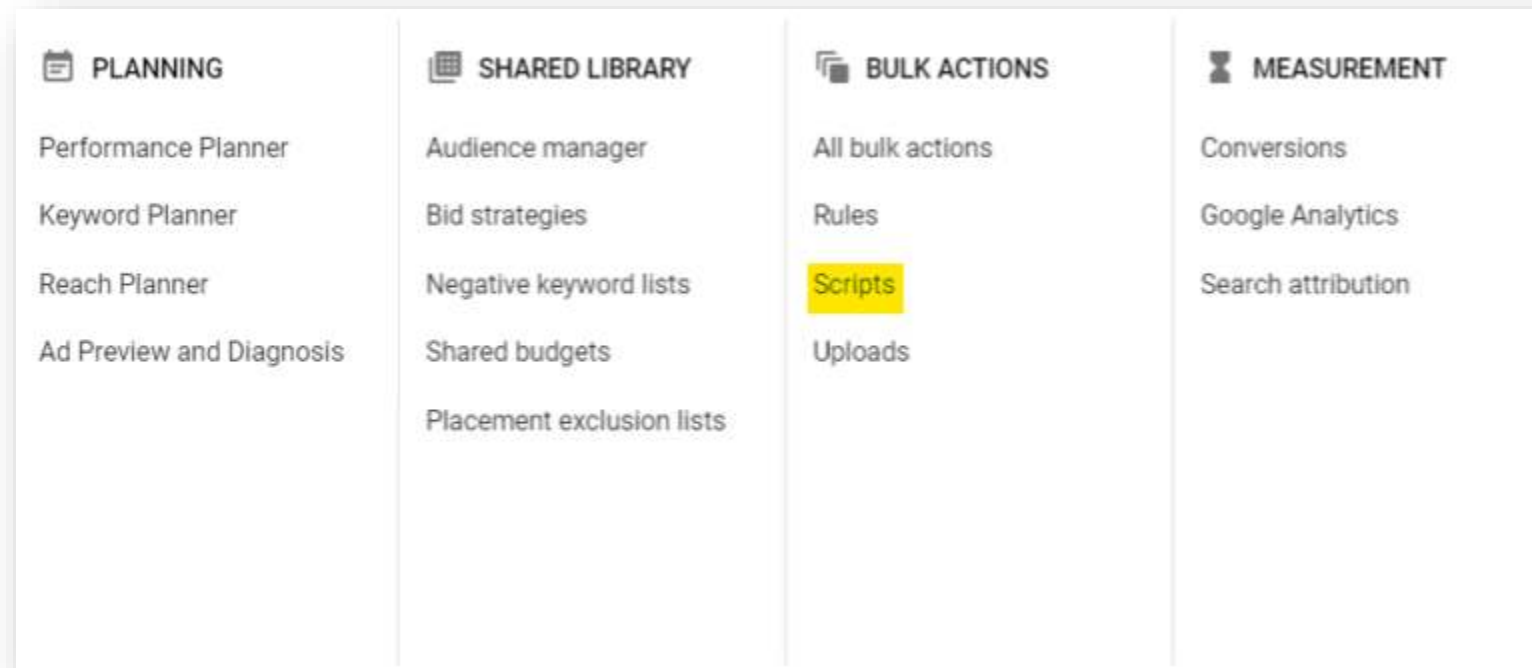
---

- A script is basically a piece of code
- The scripts are hosted on Google's servers
- Scripts can run on an *MCC* or account level

# Creating a script

---

- Open the top menu – Bulk Actions – Scripts
- Click the + icon to create a new script



# Script #1: Pages that don't load



- This script monitors broken pages on your ads (4XX or 5XX responses) and can send an email immediately or even pause the matching ads
- The script's name: (R) Ads Final URL Monitor (Pause + Report)
- Two variables that you can edit in the script:
  - The email that will be notified
  - Should the script pause ads that point to broken pages
- Save the script and set it's run frequency (recommended – Hourly)
- Note: When setting up the script you will be prompted to approve the script to run in your account

# Script #1: Pages that don't load

All bulk actions

Script name: Broken pages

ADVANCED APIS DOCUMENTATION

Rules

Code.gs

Scripts

Uploads

```
1 var EMAIL = 'mail@example.com';
2 var PAUSE = false;
3
4 function main(){
5   var htmlService = HTMLBuilderService();
6
7   var ads = AdWordsApp.ads()
8   .withCondition('CampaignStatus = ENABLED')
9   .withCondition('AdGroupStatus = ENABLED')
10  .withCondition('Status = ENABLED')
11  .withCondition('type NOT_IN [CALL_ONLY_AD]') // IMAGE_AD, MOBILE_AD, MOBILE_IMAGE_AD, PRODUCT_AD, RICH_MEDIA_AD, TEMPLATE_AD, TEXT_AD, CALL_ONLY_AD
12  .withLimit(4)
13  .get();
14
15  Logger.log('%s ads found.', ads.getTotalNumEntities());
16
17  var matches = 0;
18  while(ads.hasNext()){
19    var ad = ads.next();
20    var adURL = ad.getUrl().getFinalUrl();
21    var adHeader = ad.getHeadline();
22    var adType = ad.getType();
23
24    Logger.log('ad: %s (%s)', adHeader, adType);
25
26    var statusCode = -1;
27    try{
28      var response = UrlFetchApp.fetch(adURL, { muteHttpExceptions: true });
29      statusCode = response.getResponseCode();
30      Logger.log('[status %s] %s', statusCode, adURL);
31    }
32    catch(e){
33      Logger.log('[exception %s] %s', e.message, adURL);
34    }
35
36    // OK
37    if(statusCode == 200 || statusCode == -1)
38      continue;
39
40    var adgroup = ad.getAdGroup();
41    var adgroupName = adgroup.getName();
42
43    Logger.log('adgroup: %s', adgroupName);
44    Logger.log('%s [%s]', adURL, statusCode);
45
46    matches++;
47
48    htmlService.add(
49      '<tr>statusCode: ' + statusCode + '</tr>' +
50      '<tr>adgroup: ' + adgroupName + '</tr>' +
51      '<tr>adURL: ' + adURL + '</tr>'
52    );
53
54    if(PAUSE)
55      {
```

+

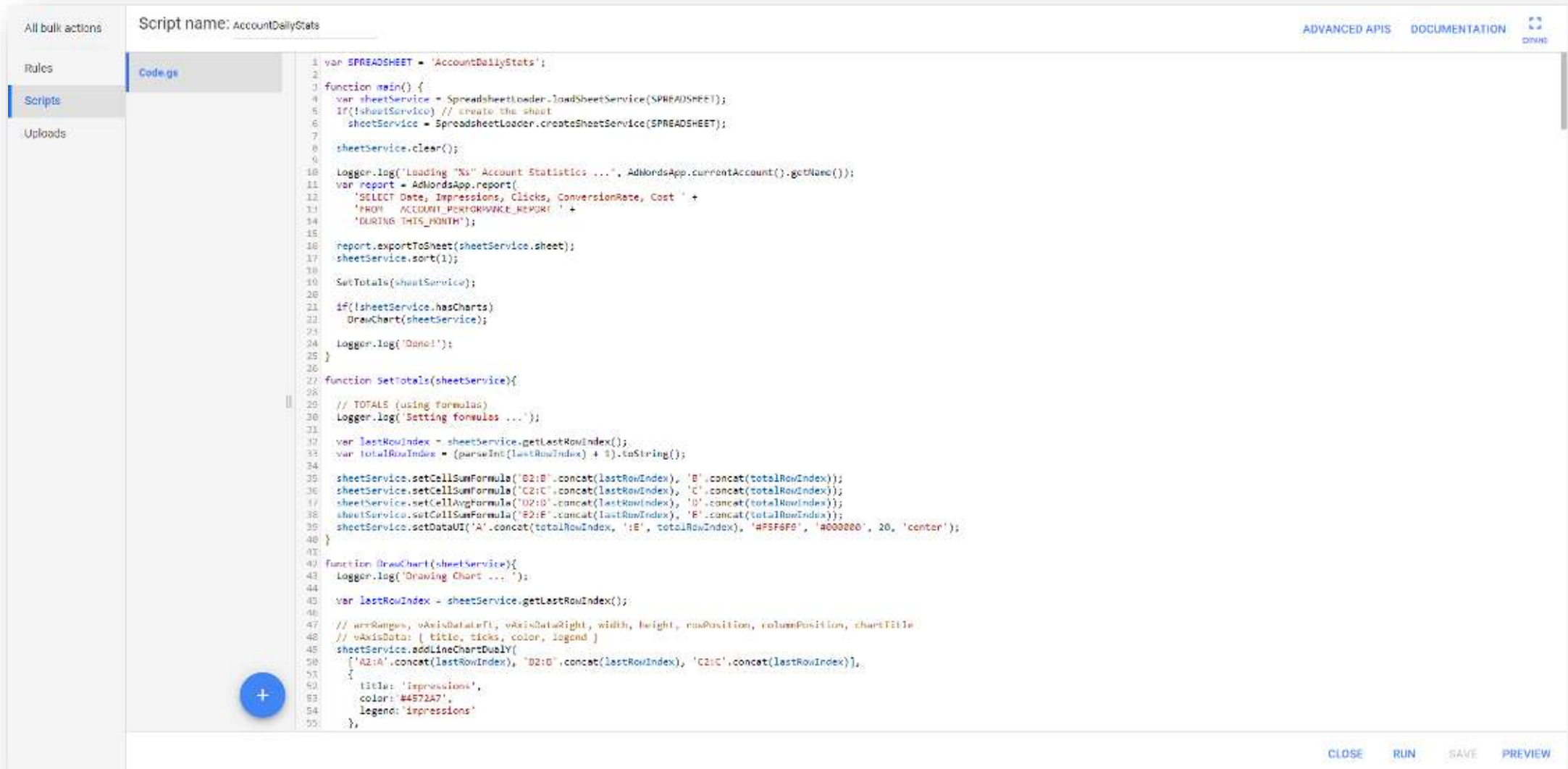
CLOSE RUN SAVE PREVIEW

# Script #2: Self updating Google Sheets Report

---

- Script's name: (R) Account Daily Stats + Charts - Current Month
- This script will create a self updating Google Sheet with a data chart and some graphs
- You can change the name of the file that's created
- If the file already exists, it will be updated. If not, a new file will be created
- This is an account level script

# Script #2: Self updating Google Sheets Report



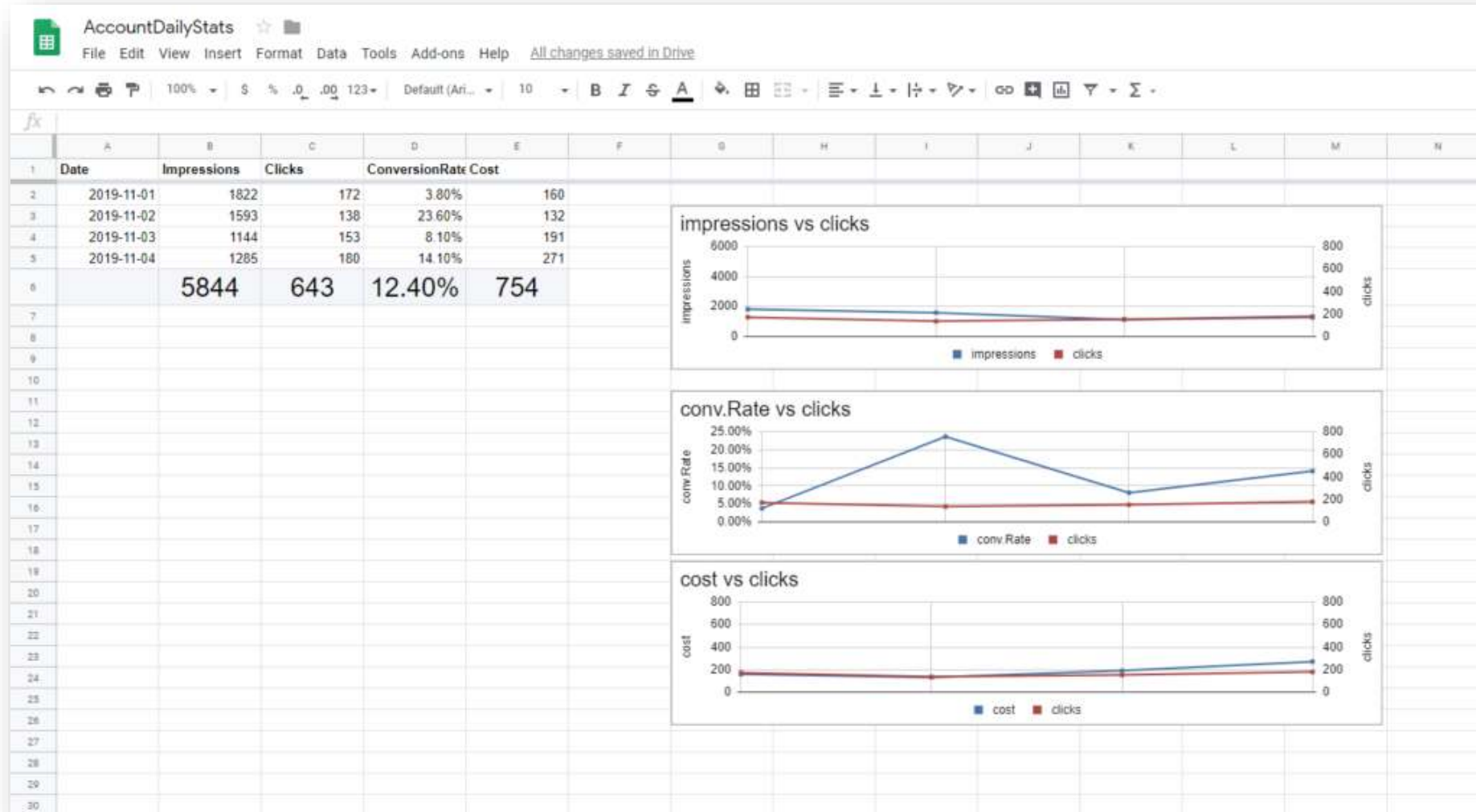
The screenshot shows the Google Apps Script editor interface. On the left, there is a sidebar with navigation options: "All bulk actions", "Rules", "Scripts" (selected), and "Uploads". The main area displays a script named "AccountDailyStats". The script code is as follows:

```
1 var SPREADSHEET = 'AccountDailyStats';
2
3 function main() {
4   var sheetService = SpreadsheetApp.getSpreadsheet(SPREADSHEET);
5   if(!sheetService) // create the sheet
6     sheetService = SpreadsheetApp.createSheet(SPREADSHEET);
7
8   sheetService.clear();
9
10  Logger.log('Loading "Ns" Account Statistics ...', AdWordsApp.currentAccount().getName());
11  var report = AdWordsApp.report(
12    'SELECT Date, Impressions, Clicks, ConversionRate, Cost ' +
13    'FROM ACCOUNT_PERFORMANCE_REPORT ' +
14    'DURING THIS_MONTH');
15
16  report.exportToSheet(sheetService.getActiveSheet());
17  sheetService.sort(1);
18
19  SetTotals(sheetService);
20
21  if(!sheetService.hasCharts)
22    DrawChart(sheetService);
23
24  Logger.log('Done!');
25 }
26
27 function SetTotals(sheetService){
28   // TOTALS (using formulas)
29   Logger.log('Setting formulas ...');
30
31   var lastRowIndex = sheetService.getLastRowIndex();
32   var totalRowIndex = (parseInt(lastRowIndex) + 1).toString();
33
34
35   sheetService.setCellSumFormula('B2:B'+totalRowIndex, 'B'+totalRowIndex);
36   sheetService.setCellSumFormula('C2:C'+totalRowIndex, 'C'+totalRowIndex);
37   sheetService.setCellAvgFormula('D2:D'+totalRowIndex, 'D'+totalRowIndex);
38   sheetService.setCellSumFormula('E2:E'+totalRowIndex, 'E'+totalRowIndex);
39   sheetService.setDataUI('A'+totalRowIndex, 'E'+totalRowIndex, '#FF69B4', '#000000', 20, 'center');
40 }
41
42 function DrawChart(sheetService){
43   Logger.log('Drawing Chart ...');
44
45   var lastRowIndex = sheetService.getLastRowIndex();
46
47   // arrRanges, vAxisDataLeft, vAxisDataRight, width, height, xPos, yPos, columnPosition, chartTitle
48   // vAxisData: { title, ticks, color, legend }
49   sheetService.addLineChartDualY(
50     ['A2:A'+lastRowIndex, 'D2:D'+lastRowIndex, 'C2:C'+lastRowIndex],
51     {
52       title: 'Impressions',
53       color: '#4572A7',
54       legend: 'Impressions'
55     }
56   );
57 }
```

At the bottom right of the editor, there are buttons for "CLOSE", "RUN", "SAVE", and "PREVIEW".



# Script #2: Self updating Google Sheets Report



# Script #3: Pausing Inefficient Ads

---

- Script's name: (R) Pause Non Converting or High Cost Ads
- This script will pause ads that aren't driving enough conversions or are driving conversions at a cost that's significantly higher than the account's average
- You can edit the lookback window, default is All Time
- This is an account level script

# Script #3: Pausing Inefficient Ads

All bulk actions: Script name: Inefficient Ads

ADVANCED APIS DOCUMENTATION

Rules Code.gs

Scripts

Uploads

```
1 var PERIOD = 'ALL_TIME'; // ALL_TIME, LAST_30_DAYS
2
3 function main() {
4   HandleNonConvertingAds();
5   HandleConvertingAds();
6 }
7
8 function HandleNonConvertingAds(){
9   Logger.log('## Non Converting Ads ##')
10  var ads = AdWordsApp.ads()
11    .withCondition('ConvertedClicks = 0')
12    .withCondition('Status = ENABLED')
13    .withCondition('Cost > 0')
14    .forDateRange(PERIOD)
15    .orderBy('Cost ASC')
16    .get();
17
18  while (ads.hasNext()) {
19    var ad = ads.next();
20    var adGroup = ad.getAdGroup();
21    var adGroupCPA = GetAdGroupCPA(adGroup);
22    var adCost = ad.getStatsFor(PERIOD).getCost();
23
24    Logger.log('ad: %s -> cost %s', ad.getHeadline(), adCost);
25    Logger.log('adGroup %s -> CPA %s', adGroup.getName(), adGroupCPA);
26
27    if(adGroupCPA == null || adCost < adGroupCPA * 2) // no changes require
28      continue;
29
30    // adCost is too high - pause the ad
31    if(!AdWordsApp.getExecutionInfo().isPreview())
32      ad.pause();
33    Logger.log('PAUSE THE AD!');
34  }
35 }
36
37 function GetAdGroupCPA(adGroup){
38  var stats = adGroup.getStatsFor(PERIOD);
39  var conversions = stats.getConvertedClicks();
40  return conversions == 0 ? 0 : (stats.getCost() / conversions); // calculate the ad "CPA" - cost per converted
41 }
42
43 function HandleConvertingAds(){
44  Logger.log('## Converting Ads ##')
45  var ads = AdWordsApp.ads()
46    .withCondition('ConvertedClicks > 0')
47    .withCondition('Status = ENABLED')
48    .withCondition('Cost > 0')
49    .forDateRange(PERIOD)
50    .orderBy('ConvertedClicks DESC')
51    .get();
52
53  while (ads.hasNext()) {
54    var ad = ads.next();
55    var adGroup = ad.getAdGroup();
```

+

CLOSE RUN SAVE PREVIEW

# Script #4: Hidden Keywords

---

- Script's name: (R) queries 2 keywords - mcc (Set)
- This script observes the keywords that have driven conversions and adds these as keywords in the relevant campaign.
- These originate from Phrase or Broad Modifier Match (BMM)
- This is an MCC level script

# Script #4: Hidden Keywords

All bulk actions

Script name: Hidden Keywords

ADVANCED APIS DOCUMENTATION

Rules

Code.gs

Scripts

Uploads

```
1 function main() {
2   var accounts = MccApp.accounts().get();
3
4   while (accounts.hasNext()) {
5     var account = accounts.next();
6     var accountName = account.getName() ? account.getName() : '--';
7     logger.log('account #%s %s', account.getCustomerId(), accountName);
8
9     MccApp.select(account);
10    execute();
11  }
12 }
13
14 function execute() {
15   var report = AdWordsApp.report(
16     'SELECT Query,Clicks,ConvertedClicks,AverageCpc ' +
17     'FROM SEARCH_QUERY_PERFORMANCE_REPORT ' +
18     'WHERE ConvertedClicks > 8 ' +
19     'DURING 20000101,' + dateFormat(new Date());
20
21   var rows = report.rows();
22   while(rows.hasNext()) {
23     var row = rows.next();
24
25     var clicks = row['Clicks'];
26     var conversions = row['ConvertedClicks'];
27
28     var query = row['Query'];
29     //logger.log(query + ' ' + clicks);
30     var found_in_my_keywords = AdWordsApp.keywords().withCondition('Text CONTAINS "' + query + '"').get().hasNext();
31     if(found_in_my_keywords)
32     {
33       logger.log('"' + query + '" found in my keywords');
34       continue;
35     }
36
37     var adGroup = AdWordsApp.adGroups().get().next();
38     var maxcpc = row['AverageCpc'];
39
40     if(!AdWordsApp.getExecutionInfo().isPreview())
41       adGroup.createKeyword(query, maxcpc);
42     logger.log('"' + query + '" added as keyword with maxcpc ' + maxcpc)
43   }
44 }
45
46 function dateFormat(date){
47   var year = date.getFullYear().toString();
48   var month = date.getMonth().toString();
49   var day = date.getDate().toString();
50
51   if(month.length == 1) month = '0' + month; // return yyyyMMdd
52   if(day.length == 1) day = '0' + day;
53
54   return year + month + day;
55 }
```

+

CLOSE RUN SAVE PREVIEW

# Other examples for scripts



- Out of stock – Monitoring an ecommerce site by scraping the product page’s HTML to identify out of stock items and pause their ads
- Feed based campaigns – Creating ads for new feed items (e.g. “Exclusive discount for 24 hours”), pausing ads when out of stock, updating product pricing in ads and more
- Budget management – Pausing a campaign or account exactly at a set limit
- Data injection – Infusing external data into the ad account (e.g. weather data)
- Alert system – Triggering alerts on custom events from the account

# Thanks for listening

---



**Avishay Freund**  
CEO

T 03-6443777 | M 054-4690690 | F 03-6448095 | [avishay@logos.co.il](mailto:avishay@logos.co.il)  
1 Adrichal Duv Carmi st, Tel Aviv | [www.logos.co.il](http://www.logos.co.il)