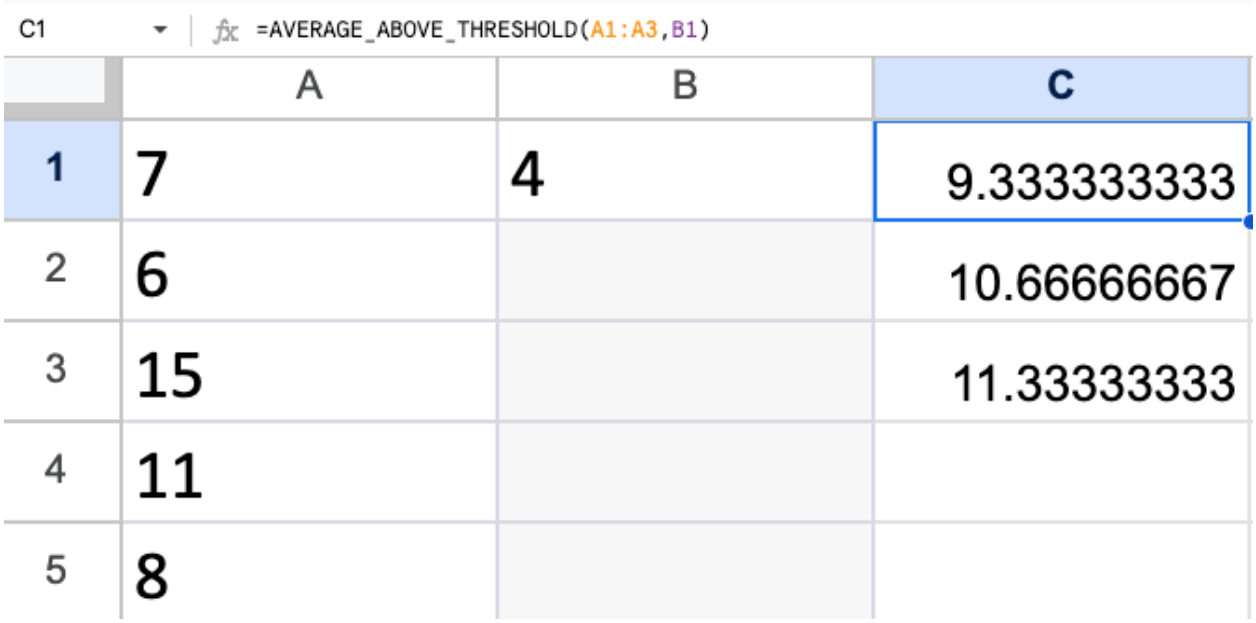
Google Sheets Formulas - Examples and Code Explanations

Calculate average from range above a threshold 2
AVERAGE_ABOVE_THRESHOLD 2
Calculate a weighted Average from a range of values 8
WEIGHTED_AVERAGE 8
Get the total cost custom sheets formula calculate values across ranges 12
CALC_TOTAL_COST 12
Calculate total with discount using a custom Apps Script formula 18
CALC_TOTAL_AMOUNT 18
How to get the total amount of items including the tax rate 24
CALC_TOTAL_AMOUNT 24
How to use data from two sheets to create a custom Apps Script function 31
CALC_TOTAL_COST_WITH_TAX 31

Calculate average from range above a threshold

AVERAGE_ABOVE_THRESHOLD



7	4	9.333333333
6		10.66666667
15		11.33333333
11		
8		

In this example, we'll create a custom formula that calculates the average of a range of numbers, excluding any values that are less than a specified threshold.

Scenario: You have a range of numbers in column A and a threshold value in cell B1. You want to calculate the average of the numbers in column A that are greater than or equal to the threshold value in B1.

Here are the steps to achieve this:

Step 1: Setting up the Spreadsheet

Create a new Google Sheets document.

Enter your data in column A, starting from A2 (A1 will be used for the threshold value).

Enter the threshold value in cell B1.

Step 2: Writing the Google Apps Script Code

Click on "Extensions" in the top menu, then select "Apps Script".

Delete any default code and replace it with the following script:

```
// Custom formula to calculate the average of values above a certain threshold
function AVERAGE_ABOVE_THRESHOLD(range,threshold){
let sum = 0;
let count = 0;
for (let i = 0; i < range.length; i++) {
if (range[i] >= threshold) {
sum += range[i];
count++;
}
}
```

```
if(count>0){
return sum /count;
}else{
return "No Values above Threshold";
}
}
```

Step 3: Using the Custom Formula in Google Sheets

Go back to your Google Sheets document.

In a cell where you want the result to appear (let's say cell C1), enter the following formula:

=AVERAGE_ABOVE_THRESHOLD(A2:A, B1)

Explanation of the Code:

The function AVERAGE_ABOVE_THRESHOLD takes two parameters: range and threshold.

- range: This parameter represents the range of cells containing the numbers you want to consider for the average calculation.
- threshold: This parameter is the threshold value specified in cell B1.

Inside the function, we initialize sum to store the sum of values and count to keep track of the number of values above the threshold.

We loop through each value in the range and check if it's greater than or equal to the threshold. If it is, we add it to the sum and increment the count.

After looping through all values, we calculate the average by dividing the sum by the count.

If no values are above the threshold, the function returns the message "No values above the threshold".

Step 4: Testing the Custom Formula

Enter some numbers in column A starting from A2.

Enter a threshold value in cell B1.

Use the custom formula in cell C1 to calculate the average of values above the threshold.

Remember that using custom formulas with Google Apps Script requires enabling the "Google Apps Script" extension in your Google Sheets document. Also, make sure that your custom function name (AVERAGE_ABOVE_THRESHOLD) matches exactly with what you use in the formula.

Sheet Name: Let's assume you're working in the default "Sheet1".

Column A: This column will contain the list of numbers you want to calculate the average of. Let's say you have the following numbers in column A, starting from A2:

A
10
15
5
20

| 8 |

Cell B1: This cell will contain the threshold value. For this example, let's assume you have set the threshold to 10 in cell B1.

B
10

Cell C1: This is where you will use the custom formula to calculate the average of values above the threshold. Enter the formula below in cell C1:

=AVERAGE_ABOVE_THRESHOLD(A2:A, B1)

With this setup, the custom formula will calculate the average of numbers that are greater than or equal to the threshold (10 in this case), which should result in an average of 15.
Please note that for the custom formula to work, you need to follow the steps outlined in the previous response to create the Apps Script code and enable the extension in your Google Sheets.

Sheet Name: Sheet1

Column A	Column B

10	10
15	
5	
20	
8	

Explanation:

- Column A contains the list of numbers you want to calculate the average of.
- Column B contains the threshold value, which is 10 in this example.

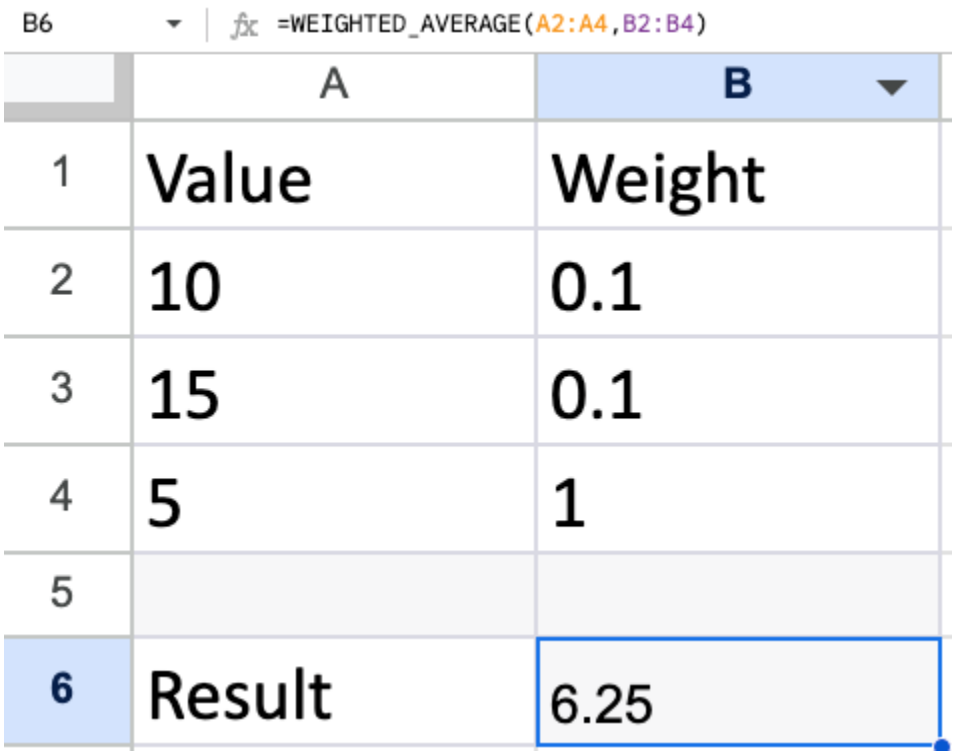
Formula: In cell C1, you will use the custom formula to calculate the average of values above the threshold.

Cell C1
=AVERAGE_ABOVE_THRESHOLD(A2:A, B1)

With this setup, the custom formula will calculate the average of numbers in Column A that are greater than or equal to the threshold (10). The result should be an average of 15 ([(10 + 15 + 20) / 3]).

Calculate a weighted Average from a range of values

WEIGHTED_AVERAGE



Value	Weight
10	0.1
15	0.1
5	1
Result	6.25

Scenario: You want to create a custom Google Sheets formula that calculates the weighted average of values in a specified range, where each value is multiplied by its corresponding weight.

Data Table:

	A	B	C
1	Value	Weight	
2	10	0.3	
3	15	0.5	
4	5	0.2	
5			
6	Result		

Step 1: Set Up the Spreadsheet

Open a new Google Sheets document.

Enter the data table as shown above in columns A and B.

Leave cells C2 and C6 empty for now.

Step 2: Write the Google Apps Script Code

Click on "Extensions" in the top menu, then select "Apps Script".

Replace the default code with the following script:

```
// Custom formula to calculate the weighted average of values
function WEIGHTED_AVERAGE(values,weights){
```

```
let sum = 0;
let totalWeight = 0;
for(let i=0;i<values.length;i++){
sum += values[i] * weights[i];
totalWeight += weights[i];
}
return sum/totalWeight;
}
```

Step 3: Use the Custom Formula in Google Sheets

Go back to your Google Sheets document.

In cell C6, enter the following formula:

=WEIGHTED_AVERAGE(A2:A4, B2:B4)

Explanation:

The WEIGHTED_AVERAGE function takes two parameters: values and weights. Both parameters represent ranges of cells that hold the values and weights, respectively.

Inside the function, we initialize the sum variable to store the cumulative product of each value multiplied by its corresponding weight, and the totalWeight variable to store the sum of weights.

The for loop iterates through each value in the values range. For each value, it multiplies the value by its corresponding weight from the weights range and adds it to the sum. Additionally, it adds the weight to the totalWeight. After looping through all values, the function returns the weighted average by dividing the sum by the totalWeight.

Step 4: Testing the Custom Formula

Enter the values and weights in columns A and B, respectively.

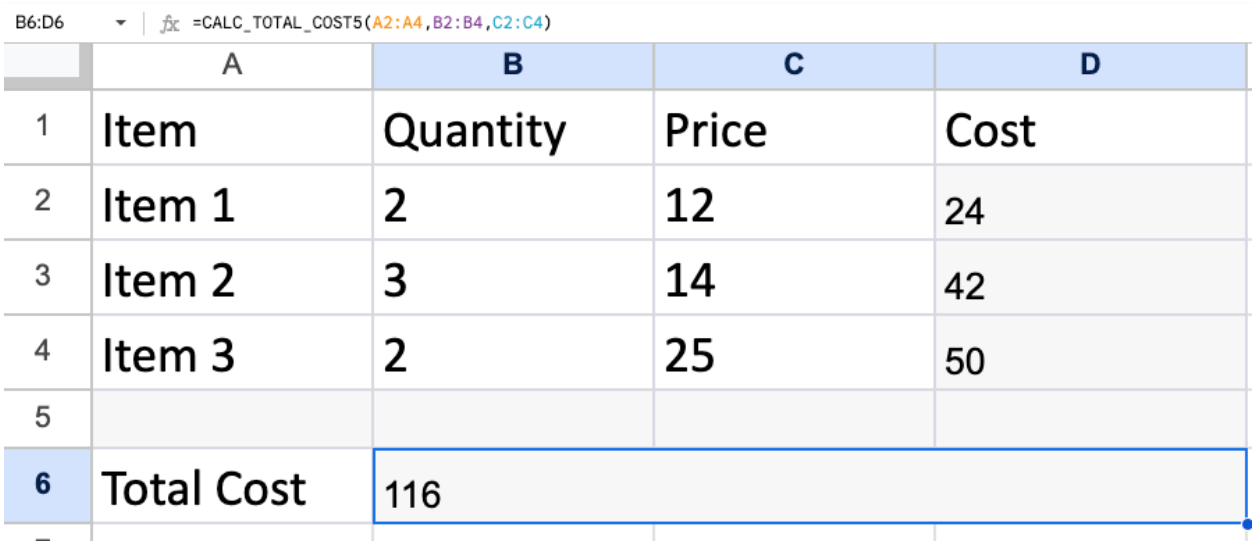
Use the custom formula in cell C6 to calculate the weighted average.

For example, if you input the values 10, 15, and 5 in cells A2:A4 and the weights 0.3, 0.5, and 0.2 in cells B2:B4, the calculated weighted average in cell C6 should be approximately 11.67.

Remember that you need to enable the "Google Apps Script" extension and use the exact function name (WEIGHTED_AVERAGE) in your formula.

Get the total cost custom sheets formula calculate values across ranges

CALC_TOTAL_COST



Item	Quantity	Price	Cost
Item 1	2	12	24
Item 2	3	14	42
Item 3	2	25	50
Total Cost	116		

Scenario: You have a list of items with their quantities and prices in columns A, B, and C. You want to create a custom formula that calculates the total cost for each item (quantity * price) and provides the overall total of all items' costs.

Here are the steps to achieve this:

Step 1: Setting up the Spreadsheet

Create a new Google Sheets document.

Enter your data in columns A, B, and C starting from row 2.

Leave a blank cell in column D for the calculated cost for each item.

Leave another blank cell in column E for the custom formula that calculates the total cost.

Data Table:

A	B	C	D	E
Item	Quantity	Price	Cost	
Item 1	5	10		
Item 2	3	15		
Item 3	2	20		
Total Cost			cuntio	

Step 2: Writing the Google Apps Script Code

Click on "Extensions" in the top menu, then select "Apps Script".

Delete any default code and replace it with the following script:

```
// Custom formula to calculate the total cost for each item and overall total
function CALC_TOTAL_COST(items, quantities, prices) {
```

```
var totalCost = 0;
for (var i = 0; i < items.length; i++) {
var cost = quantities[i] * prices[i];
totalCost += cost;
items[i][3] = cost;
}
return totalCost;
}
```

Step 3: Using the Custom Formula in Google Sheets

Go back to your Google Sheets document.

In a cell where you want the total cost to appear (let's say cell E6), enter the following formula:

=CALC_TOTAL_COST(A2:A4, B2:B4, C2:C4)

B6:D6 =CALC_TOTAL_COST(A2:A4,B2:B4,C2:C4)

	A	B	C	D
1	Item	Quantity	Price	Cost
2	Item 1	5	10	50
3	Item 2	3	15	45
4	Item 3	2	20	40
5				
6	Total Cost	Item 1 * 5, Item 2 * 3, Item 3 * 2, total cost is $135		

```
function CALC_SUB(price,qty){
return price * qty;
}
function CALC_TOTAL_COST(items, quantities, prices) {
let totalCost = 0;
let output = '';
for(let i=0;i<items.length;i++){
const cost = quantities[i] * prices[i];
output += `${items[i]} * ${quantities[i]}, `;
totalCost += cost;
}
return `${output} total cost is $${totalCost}`;
}
```

Explanation of the Code:

The function CALC_TOTAL_COST takes three parameters: items, quantities, and prices.

- items: This parameter represents the range of cells containing the items' names.
- quantities: This parameter represents the range of cells containing the quantities for each item.
- prices: This parameter represents the range of cells containing the prices for each item.

Inside the function, we initialize the totalCost variable to keep track of the overall cost.

The for loop iterates through each item. For each item, it calculates the cost by multiplying the quantity and price, and then adds this cost to the totalCost.

Additionally, it updates the corresponding cell in column D (the "Cost" column) with the calculated cost for each item.

The function returns the overall totalCost.

Step 4: Testing the Custom Formula

Enter the items, quantities, and prices in columns A, B, and C starting from row 2.

Use the custom formula in cell E6 to calculate the overall total cost of all items.

For example, if you have entered the data as shown in the data table, the calculated total cost in cell E6 should be 110.

Remember to enable the "Google Apps Script" extension and use the exact function name (CALC_TOTAL_COST) in your formula.

CALC_SUB(price, qty): This function takes in two parameters, price and qty (quantity). It calculates the subtotal cost of a product by multiplying the given price with the given quantity. The function simply returns the result of this multiplication.

```
function CALC_SUB(price,qty){
return price * qty;
}
```

CALC_TOTAL_COST(items, quantities, prices): This function is more complex and takes three arrays as parameters: items, quantities, and prices. It calculates the total cost for a list of items based on their quantities and prices. The function iterates over each item using a for loop and calculates the cost of each item by multiplying its quantity with its corresponding price.

The function maintains two variables: totalCost to keep track of the cumulative cost of all items, and output to store a string that records the calculation for each item.

Inside the loop, the cost of the current item is calculated using quantities[i] * prices[i], and then the item's description along with its quantity is added to the output string. The calculated cost is also added to the totalCost.

After iterating through all items, the function returns a string containing the calculations for each item and the total cost.

```
function CALC_TOTAL_COST(items, quantities, prices) {
```

```
let totalCost = 0;
let output = '';
for(let i=0;i<items.length;i++){
const cost = quantities[i] * prices[i];
output += `${items[i]} * ${quantities[i]}, `;
totalCost += cost;
}
return `${output} total cost is $${totalCost}`;
}
```

// Output: "Apple * 5, Banana * 3, Orange * 2, total cost is $7.1"

The CALC_TOTAL_COST function is designed to calculate the total cost of purchasing multiple items with their respective quantities and prices. The function returns a formatted string that includes both the item calculations and the total cost.

Calculate total with discount using a custom Apps Script formula

```
CALC_TOTAL_AMOUNT
```

=CALC_TOTAL_AMOUNT(A2:A4,B2:B4,C2:C4,D2:D4)

A	B	C	D	E	F
Item	Quantity	Price	Discount	Total Amount	
Item 1	5	10	0.1	45	45
Item 2	2	15	0.05	28.5	28.5
Item 3	3	20	0.2	48	48

Item	Quantity	Price	Discount	Total Amount	
Item 1	5	10	0.1	45	45
Item 2	2	15	0.05	28.5	28.5
Item 3	3	20	0.2	48	48

CALC_DIS(qty, cost, discount): This function calculates the total cost for a particular quantity of items, given their individual cost and a discount rate. It takes three parameters: qty (quantity), cost (individual cost of each item), and discount (discount rate as a decimal between 0 and 1).

Inside the function, a variable named total is initialized to 0. Then, the total cost is calculated by multiplying the qty with the cost, and then further reducing the result by the product of qty, cost, and discount. The 1 - discount factor is used to apply the discount rate.

The calculated total is returned by the function.

```
function CALC_DIS(qty,cost,discount){
let total = 0;
total = qty * cost * (1-discount);
return total;
```

```
}
```

```
const totalCost = CALC_DIS(5, 10, 0.1); // Calculates (5 * 10 * 0.9) = 45
console.log(totalCost); // Output: 45
```

CALC_TOTAL_AMOUNT(items, quantities, prices, discounts): This function calculates the total amount for each item in a list, considering their quantities, individual prices, and discount rates. It takes four arrays as parameters: items (item names), quantities (quantities of each item), prices (individual prices of each item), and discounts (discount rates for each item).

The function iterates over each item using a for loop. Inside the loop, it extracts the corresponding quantity, price, and discount for the current item. The total amount for the item is then calculated by multiplying quantity, price, and 1 - discount. This amount is added to the totalAmounts array.

After iterating through all items, the function returns an array (totalAmounts) containing the calculated total amounts for each item.

```
function CALC_TOTAL_AMOUNT(items,quantities,prices,discounts){
let totalAmounts = [];
for(let i=0;i<items.length;i++){
let quantity = quantities[i];
let price = prices[i];
let discount = discounts[i];
const itemAmount = quantity * price * (1-discount);
totalAmounts.push([itemAmount]);
}
```

```
return totalAmounts;
}
```

```
const items = ['Apple', 'Banana', 'Orange'];
const quantities = [5, 3, 2];
const prices = [1, 0.5, 0.8];
const discounts = [0.1, 0.2, 0.15];

const totalAmountsArray = CALC_TOTAL_AMOUNT(items, quantities, prices,
discounts);
console.log(totalAmountsArray);
// Output: [[4.5], [1.2], [1.36]]
```

The CALC_DIS function calculates the total cost considering the quantity, individual cost, and discount. The CALC_TOTAL_AMOUNT function calculates the total amount for each item in a list, accounting for quantities, prices, and discounts, and returns an array of these amounts.

In this example, we'll create a custom formula that calculates the total amount of a purchase with discounts based on the quantity of items purchased.
Scenario: You want to create a custom formula that calculates the total amount of a purchase with discounts based on the quantity of items purchased.
Data Table:

A	B	C	D	E

Item	Quantity	Price	Discount	Total Amount
Item 1	5	10	0.1	
Item 2	2	15	0.05	
Item 3	3	20	0.2	
Total				

Step 1: Setting up the Spreadsheet

Create a new Google Sheets document.

Enter the item names, quantities, prices, and discounts in columns A to D starting from row 2.

Leave cells in column E empty for now.

Step 2: Writing the Google Apps Script Code

Click on "Extensions" in the top menu, then select "Apps Script".

Delete any default code and replace it with the following script:

```
// Custom formula to calculate the total amount with discounts
function CALC_TOTAL_AMOUNT(items, quantities, prices, discounts) {
var totalAmounts = [];

for (var i = 0; i < items.length; i++) {
var quantity = quantities[i];
var price = prices[i];
```

```
var discount = discounts[i];

var itemAmount = quantity * price * (1 - discount);
totalAmounts.push([itemAmount]);
}

return totalAmounts;
}
```

Step 3: Using the Custom Formula in Google Sheets

Go back to your Google Sheets document.

In cell E2, enter the following formula:

=CALC_TOTAL_AMOUNT(A2:A, B2:B, C2:C, D2:D)

Explanation of the Code:

The function CALC_TOTAL_AMOUNT calculates the total amount of a purchase with discounts based on the quantity of items purchased.

It takes four parameters: items, quantities, prices, and discounts.

- items: The range of cells containing the item names.
- quantities: The range of cells containing the quantities of items purchased.
- prices: The range of cells containing the prices per item.
- discounts: The range of cells containing the discounts for each item.

Inside the function, a loop iterates through each item.

The total amount for each item is calculated by multiplying the quantity, price, and the complement of the discount.

The calculated total amounts are stored in the totalAmounts array.

The function returns the array of calculated total amounts.

Step 4: Testing the Custom Formula

Enter item names, quantities, prices, and discounts in columns A to D starting from row 2.

Use the custom formula in cell E2 to calculate the total amount of the purchase with discounts for each item.

For example, if you have entered the item details as shown in the data table, the calculated total amounts after applying discounts should appear in column E. Remember to enable the "Google Apps Script" extension and use the exact function name (CALC_TOTAL_AMOUNT) in your formula.

How to get the total amount of items including the tax rate

CALC_TOTAL_AMOUNT

F2 =CALC_TAX_TOTAL_AMOUNT(A2:A4,B2:B4,C2:C4,D2:D4)

	A	B	C	D	E	F
1	Product	Price	Quantity	Tax Rate	Subtotal	Total Amount
2	Product A	10	5	0.08	4	54
3	Product B	20	3	0.05	3	63
4	Product C	15	2	0.1	3	33
5					Total	150

Product	Price	Quantity	Tax Rate	Subtotal	Total Amount
Product A	10	5	0.08	4	54
Product B	20	3	0.05	3	63
Product C	15	2	0.1	3	33
				Total	150

In this example, we'll create a custom formula that calculates the total amount of a sales transaction based on different products and their quantities.

Scenario: You want to create a custom formula that calculates the total amount of a sales transaction based on different products and their quantities, considering the tax rate.

Data Table:

A	B	C	D	E	F
Product	Price	Quantity	Tax Rate	Subtotal	Total Amount
Product A	10	5	0.08		
Product B	20	3	0.05		
Product C	15	2	0.10		
Total					

Step 1: Setting up the Spreadsheet

Create a new Google Sheets document.

Enter the product names, prices, quantities, and tax rates in columns A to D starting from row 2.

Leave cells in columns E and F empty for now.

Step 2: Writing the Google Apps Script Code

Click on "Extensions" in the top menu, then select "Apps Script".

Delete any default code and replace it with the following script:

Step 3: Using the Custom Formula in Google Sheets

Go back to your Google Sheets document.

In cell F2, enter the following formula:

=CALC_TOTAL_AMOUNT(A2:A, B2:B, C2:C, D2:D)

```
function CALC_TAX_TOTAL(price,qty,tax){
const total = price * qty * tax;
return total;
}
function CALC_TAX_TOTAL_AMOUNT(items,price,qty,tax){
let total = 0;
const arr = [];
for(let i=0;i<items.length;i++){
const tot = price[i] * qty[i];
const subTotal = tot * tax[i];
arr.push([subTotal+tot]);
```

```
total += subTotal+tot;
}
arr.push(total);
return arr;
}
```

CALC_TAX_TOTAL(price, qty, tax): This function calculates the total tax amount for a given price, quantity, and tax rate. It takes three parameters: price (individual price), qty (quantity), and tax (tax rate as a decimal between 0 and 1).
Inside the function, a constant named total is calculated by multiplying the price and qty, and then further multiplying the result by the tax. This calculates the tax amount for the specified quantity of items at the given price.
The calculated total tax amount is returned by the function.

```
function CALC_TAX_TOTAL(price, qty, tax) {
  const total = price * qty * tax;
  return total;
}
```

Example usage:

```
const taxAmount = CALC_TAX_TOTAL(10, 5, 0.1); // Calculates (10 * 5 * 0.1) = 5
console.log(taxAmount); // Output: 5
```

CALC_TAX_TOTAL_AMOUNT(items, price, qty, tax): This function calculates the total amount, including tax, for a list of items with their respective prices,

quantities, and tax rates. It takes four arrays as parameters: items (item names), price (individual prices), qty (quantities of each item), and tax (tax rates for each item).

The function initializes two variables: total to keep track of the cumulative total amount, and an array arr to store the subtotals and the final total.

It then iterates over each item using a for loop. Inside the loop, it calculates the total amount for the current item by multiplying its price and quantity. It also calculates the subtotal by multiplying the total amount with the tax rate.

The subtotal, when added to the total amount, gives the total amount including tax for the item. This value is added to the arr array, and the total variable is updated.

After iterating through all items, the final total is appended to the arr array. The function then returns the arr array containing the subtotals for each item and the final total.

```
function CALC_TAX_TOTAL_AMOUNT(items, price, qty, tax) {
  let total = 0;
  const arr = [];
  for (let i = 0; i < items.length; i++) {
    const tot = price[i] * qty[i];
    const subTotal = tot * tax[i];
    arr.push([subTotal + tot]);
    total += subTotal + tot;
  }
  arr.push(total);
```

```
  return arr;
}
```

Example usage:

```
const items = ['Apple', 'Banana', 'Orange'];
const prices = [1, 0.5, 0.8];
const quantities = [5, 3, 2];
const taxes = [0.1, 0.15, 0.08];
const totalAmountsArray = CALC_TAX_TOTAL_AMOUNT(items, prices, quantities, taxes);
console.log(totalAmountsArray);
// Output: [[5.5], [0.975], [1.36], 7.835]
```

The CALC_TAX_TOTAL function calculates the tax amount for a specified price and quantity. The CALC_TAX_TOTAL_AMOUNT function calculates the total amount including tax for each item in a list, considering their prices, quantities, and tax rates, and returns an array containing subtotals and the final total.

Explanation of the Code:

The function CALC_TOTAL_AMOUNT calculates the total amount of a sales transaction based on different products, their prices, quantities, and tax rates.

It takes four parameters: products, prices, quantities, and taxRates.

- products: The range of cells containing the product names.
- prices: The range of cells containing the prices of each product.

- quantities: The range of cells containing the quantities of each product.
- taxRates: The range of cells containing the tax rates for each product.

Inside the function, a loop iterates through each product.

For each product, it calculates the subtotal by multiplying the price and quantity.

It then calculates the tax amount by multiplying the subtotal with the tax rate.

The item total is calculated by adding the subtotal and tax.

The calculated item totals are accumulated to calculate the total amount of the sales transaction.

The function returns the total amount of the sales transaction.

Step 4: Testing the Custom Formula

Enter product names, prices, quantities, and tax rates in columns A to D starting from row 2.

Use the custom formula in cell F2 to calculate the total amount of the sales transaction.

For example, if you have entered the product details as shown in the data table, the calculated total amount of the sales transaction should appear in cell F2. Remember to enable the "Google Apps Script" extension and use the exact function name (CALC_TOTAL_AMOUNT) in your formula.

How to use data from two sheets to create a custom Apps Script function

CALC_TOTAL_COST_WITH_TAX

B6 =CALC_FINAL_TOTAL(B2:B4,C2:C4,D2:D4)

	A	B	C	D	E
1	Item	Quantity	Price	Category	Cost
2	Item 1	5	10	Category 3	54
3	Item 2	3	15	Category 2	51.75
4	Item 3	2	20	Category 1	44
5					
6	Total Cost	149.75			

E2 =CALC_TOTAL_TAX(A2,B2,C2,D2)

	A	B	C	D	E
1	Item	Quantity	Price	Category	Cost
2	Item 1	5	10	Category 3	54
3	Item 2	3	15	Category 2	51.75
4	Item 3	2	20	Category 1	44
5					
6	Total Cost	149.75			

	A	B
1	Category	Tax Rate
2	Category 1	0.1
3	Category 2	0.15
4	Category 3	0.08

```
function getTaxRate(category) {
const sheets = SpreadsheetApp.getActive().getSheetByName('TaxTable');
const taxTable = sheets.getDataRange().getValues();
let tax = 0
taxTable.forEach(val =>{
if (val[0] == category) {
tax = val[1];
}

})
return tax;
}

function test(){
Logger.log(getTaxRate('Category 1'))
}
```

Category	Tax Rate
Category 1	0.1
Category 2	0.15
Category 3	0.08

Item	Quantity	Price	Category	Cost
Item 1	5	10	Category 3	54
Item 2	3	15	Category 2	51.75
Item 3	2	20	Category 1	44
Total Cost	149.75			

getTaxRate(category): This function retrieves the tax rate for a specific category from a sheet named 'TaxTable' within the active spreadsheet. Here's a detailed breakdown of how this function works:

- const sheets = SpreadsheetApp.getActive().getSheetByName('TaxTable');: This line gets the active spreadsheet and fetches the sheet named 'TaxTable' using the getSheetByName method. The variable sheets now holds a reference to this sheet.
- const taxTable = sheets.getDataRange().getValues();: Here, the getDataRange() method is used to get the entire range of data in the 'TaxTable' sheet. The getValues() method is then used to retrieve the values within that range. This produces a two-dimensional array called taxTable, where each row represents a row in the sheet, and each column represents a cell value in that row.
- let tax = 0;: A variable named tax is initialized to 0. This variable will store the tax rate for the specified category.

- taxTable.forEach(val => {...}: This line initiates a loop that iterates through each row of the taxTable array. For each row (represented by the array val), the following code block is executed:
    - if (val[0] == category) { tax = val[1]; }: This condition checks whether the value in the first column of the current row (val[0]) matches the provided category. If there's a match, it means the tax rate for the specified category has been found in the second column of that row (val[1]). The tax rate is assigned to the tax variable.
- Finally, the function returns the value of the tax variable, which is either the tax rate found for the specified category or the default value of 0 if no match was found.

test(): This function is a test function that logs the result of calling the getTaxRate(category) function. It's meant to showcase how the getTaxRate function works by providing an example usage.

- Logger.log(getTaxRate('Category 1')): This line calls the getTaxRate function with the argument 'Category 1'. The result, which is the tax rate associated with this category, is then logged to the execution log using the Logger.log() method.

In summary, the code retrieves tax rates from a 'TaxTable' sheet based on specified categories and provides a test function to demonstrate its functionality. The getTaxRate(category) function looks up the tax rate associated with a given category, and the test() function showcases this behavior by logging the tax rate for the category 'Category 1'.

In this example, we'll create a custom formula that calculates the total cost of items based on their categories, with different tax rates applied to each category. Scenario: You have a list of items with their quantities, prices, and categories in columns A, B, and C. Additionally, you have a separate table of categories and their corresponding tax rates in columns F and G. You want to create a custom formula that calculates the total cost for each item, considering the tax rates based on their categories.

Data Table:

A	B	C	D	E	F	G
Item	Quantity	Price	Category	Cost		
Item 1	5	10	Category 1			
Item 2	3	15	Category 2			
Item 3	2	20	Category 1			
Total Cost						

Tax Rate Table:

F	G
Category	Tax Rate
Category 1	0.1

Category 2	0.15
Category 3	0.08

Step 1: Setting up the Spreadsheet

Create a new Google Sheets document.

Enter your data in columns A to C starting from row 2.

Leave a blank cell in column D for the calculated cost for each item.

Leave another blank cell in column E for the custom formula that calculates the total cost.

Enter the tax rate table data in columns F and G.

Step 2: Writing the Google Apps Script Code

Click on "Extensions" in the top menu, then select "Apps Script".

Delete any default code and replace it with the following script:

```
function CALC_TOTAL_TAX(item,qty,price,cat){
const taxRate = getTaxRate(cat);
const subTotal = qty * price;
const total = subTotal * taxRate + subTotal;
return total;
}

function CALC_FINAL_TOTAL(qty,price,cat){
let total = 0;
for(let i=0;i<qty.length;i++){
const taxRate = getTaxRate(cat[i]);
```

```
const subTotal = qty[i] * price[i];
total += subTotal * taxRate + subTotal;
}
return total;
}
```

Step 3: Using the Custom Formula in Google Sheets

Go back to your Google Sheets document.

In a cell where you want the total cost to appear (let's say cell E6), enter the following formula:

CALC_TOTAL_TAX(item, qty, price, cat): This function calculates the total cost of an item, including tax, based on its quantity, price, and category. Here's a detailed breakdown of how this function works:

- const taxRate = getTaxRate(cat);: This line calls the getTaxRate(category) function to retrieve the tax rate associated with the given category.
- const subTotal = qty * price;: Calculates the subtotal by multiplying the quantity and the price of the item.
- const total = subTotal * taxRate + subTotal;: Calculates the total cost by adding the tax amount to the subtotal. The tax amount is obtained by multiplying the subtotal by the tax rate.
- Finally, the function returns the calculated total cost.

CALC_FINAL_TOTAL(qty, price, cat): This function calculates the total cost for multiple items, considering their quantities, prices, and categories. It iterates

through each item using a for loop, and for each item, it calculates the total cost including tax using a similar approach to the CALC_TOTAL_TAX function. The calculated costs for each item are accumulated to calculate the final total cost. The function returns this final total.

In summary, these functions work together to calculate the total cost of items, considering their quantities, prices, and tax rates based on categories. The CALC_TOTAL_TAX function calculates the total cost for a single item, and the CALC_FINAL_TOTAL function calculates the total cost for multiple items. The getTaxRate function retrieves the tax rate for a given category from a spreadsheet, and the test function is a demonstration of how to use the getTaxRate function.