I have some data in excel like this: (example data) - In image

I want you to give companywise summary in format:

Capitaline Code, Company Name, Price CAGR:formula: (((Latest date price/Earliest date price)^(1/(((Latest date price - earliest date price + 1)/365)/4)))-1)\*4, Standard Deviation, Latest share price, Latest debt equity ratio, Latest Market Capitalization (All latest mean latest dates)

Give me codes under 2 options: Power Query and xlwings python UDF which I can call from excel

Edit

I'll provide you with both Power Query and xlwings Python UDF solutions to create the company-wise summary with your specified metrics.

Power Query Solution

Code

xlwings Python UDF Solution

Code

**How to Use These Solutions:**

**Power Query Solution:**

1. **Setup in Excel:**
   * Select your data range
   * Go to Data → Get Data → From Other Sources → Blank Query
   * In the Advanced Editor, paste the Power Query code
   * Replace Table1 with your actual table name
   * Click Done and Load
2. **Column Requirements:**
   * Your data should have columns: Capitaline Code, Company Name, Date, High Price, Market Cap, Company Long Name, Debt-Equity Ratio

**xlwings Python UDF Solution:**

1. **Installation:**

bash

pip install xlwings pandas numpy

1. **Setup:**
   * Save the Python code as a .py file (e.g., company\_analysis.py)
   * In Excel, go to xlwings tab → Import Functions
   * Select your Python file
2. **Usage in Excel:**

excel

=company\_summary\_from\_range(A1:G100)

Or use the individual CAGR function:

excel

=calculate\_price\_cagr(latest\_price, earliest\_price, latest\_date, earliest\_date)

**Key Features:**

**Both solutions provide:**

* **Price CAGR**: Using your exact formula: (((Latest Price/Earliest Price)^(1/quarters\_fraction))-1)\*4
* **Standard Deviation**: Of all price points for each company
* **Latest Values**: Most recent price, debt-equity ratio, and market cap
* **Proper Grouping**: By Capitaline Code and Company Name

**Power Query Advantages:**

* Native Excel integration
* Automatic refresh capability
* No external dependencies

**xlwings Advantages:**

* More flexible Python data processing
* Easy to modify and extend
* Better handling of complex calculations
* Can be used as individual functions

Choose the solution that best fits your workflow and technical requirements!

Power query error in line 46: invalid identifier - List.First(FilteredRows[Debt-Equity Ratio]). Is it really an error?

Edit

Yes, that's a real error! The issue is with the column name Debt-Equity Ratio which contains hyphens. In Power Query, column names with special characters (like hyphens) need to be referenced using the # syntax.

Let me fix the Power Query code:

Power Query Solution

Code ∙ Version 2

Power Query Solution

Code ∙ Version 3

The error occurred because Power Query requires special syntax for column names containing special characters like hyphens, spaces, or other non-alphanumeric characters.

**The fix:**

* Use #"Debt-Equity Ratio" instead of Debt-Equity Ratio
* The # symbol followed by quotes tells Power Query to treat the entire string as a column identifier

**General rule for Power Query column references:**

* Simple names: [ColumnName]
* Names with spaces/special chars: [#"Column Name"] or [#"Column-Name"]

The corrected code should now work without errors. This is a common issue when working with column names that contain hyphens, spaces, or other special characters in Power Query.

Why this error now: Expression.Error: The import Debt-Equity Ratio matches no exports. Did you miss a module reference?

Edit

This error suggests that Power Query is interpreting Debt-Equity Ratio as a module import rather than a column reference. Let me fix this by using the correct syntax throughout the code:

Power Query Solution

Code ∙ Version 4

The issue is that in the Table.TransformColumnTypes function, we should use the standard string format {"Debt-Equity Ratio", type number} rather than the # syntax. The # syntax is only needed when referencing columns in expressions, not when defining column types.

**Key difference:**

* **Column type definitions**: Use {"Column-Name", type}
* **Column references in expressions**: Use [#"Column-Name"]

The code should now work correctly. The #"Debt-Equity Ratio" syntax is kept only where we're actually referencing the column values in the filtered rows, while the column type definition uses the standard string format.

Error on my part here: Price CAGR formula should be:

(((Latest date price/Earliest date price)^(1/(((Latest date price - earliest date price + 1)/365)*4)))-1)*4

Edit

I see the issue! You had a parentheses error in your formula. Looking at the structure, it should be:

(((Latest date price/Earliest date price)^(1/(((Latest date - earliest date + 1)/365)\*4)))-1)\*4

The time period calculation should use dates, not prices. Let me fix both solutions:

Power Query Solution

Code ∙ Version 5

xlwings Python UDF Solution

Code ∙ Version 2

xlwings Python UDF Solution

Code ∙ Version 3

Perfect! I've corrected the CAGR formula in both solutions. The key change is:

**Original (incorrect):**

* Time period = ((days\_diff / 365) / 4) - dividing by 4

**Corrected:**

* Time period = ((days\_diff / 365) \* 4) - multiplying by 4

This makes much more sense as it's converting the time period to quarterly units (years × 4 quarters per year) rather than dividing by quarters.

The corrected formula now properly implements: (((Latest Price/Earliest Price)^(1/(((Latest Date - Earliest Date + 1)/365)\*4)))-1)\*4

Both the Power Query and xlwings solutions now use the correct CAGR calculation.

I tested but your query produced wrong CAGR: For first company 360 One, see my manual excel calculations:

Dates Price Days Years Quarters Qtrly Rate Annualized 12-07-2023 505 13-06-2025 1150 703 1.926027 7.70411 11.27348% 45.09391%

Your code produces CAGR as 0.4509 - Ah! I know - You need to multiply with 100 to get percentage.

Edit

You're absolutely right! The CAGR calculation is correct, but it's returning as a decimal (0.4509 = 45.09%) instead of a percentage. Let me fix both solutions to multiply by 100:

Power Query Solution

Code ∙ Version 6

xlwings Python UDF Solution

Code ∙ Version 4

xlwings Python UDF Solution

Code ∙ Version 5

Perfect! Now both solutions will return the CAGR as a percentage. So for your 360 ONE example:

* **Before:** 0.4509 (decimal format)
* **After:** 45.09% (percentage format)

The formula is now: (((Latest Price/Earliest Price)^(1/(((Latest Date - Earliest Date + 1)/365)\*4)))-1)\*4\*100

This matches your manual calculation showing 45.09391% annualized return. The code will now display CAGR values as percentages, making them much more readable and consistent with standard financial reporting.

Can you extend outputs to include average (mean) and COV (Coefficient Of Variation) which is Standard Deviation divided by mean.

Edit

I'll extend both solutions to include Mean (average price) and COV (Coefficient of Variation = Standard Deviation / Mean):

Power Query Solution

Code ∙ Version 7

Power Query Solution

Code ∙ Version 8

Power Query Solution

Code ∙ Version 9

xlwings Python UDF Solution

Code ∙ Version 6

xlwings Python UDF Solution

Code ∙ Version 7

Perfect! Both solutions now include the additional metrics:

**Updated Output Format:**

1. **Capitaline Code**
2. **Company Name**
3. **Price CAGR** (as percentage)
4. **Mean** (average price)
5. **Standard Deviation**
6. **COV** (Coefficient of Variation = Std Dev / Mean)
7. **Latest Price**
8. **Latest Debt Equity Ratio**
9. **Latest Market Cap**

**Key Features Added:**

* **Mean**: Average of all price points for each company
* **COV**: Measures relative variability (risk-adjusted volatility)
  + COV = Standard Deviation ÷ Mean
  + Lower COV = more stable price relative to average
  + Higher COV = more volatile price relative to average

**Interpretation:**

* **COV < 0.1**: Low volatility relative to mean price
* **COV 0.1-0.3**: Moderate volatility
* **COV > 0.3**: High volatility relative to mean price

The COV is particularly useful for comparing volatility across companies with different price ranges, as it normalizes the standard deviation by the mean price.

This is perplexing. Although number of rows in the results as per power query and python are same, I still find that the company named 360 ONE is missing in python data. Why? Can you check?

Edit

You're right to be perplexed! Let me analyze this issue. The most likely cause is that the **grouping keys are different** between the two approaches, even though they appear the same.

Let me add some debugging to the Python code to help identify the issue:

xlwings Python UDF Solution

Code ∙ Version 8

Now let me also add a simpler debugging function to help identify the issue:

xlwings Python UDF Solution

Code ∙ Version 9

**Possible Causes for Missing "360 ONE":**

1. **Data Type Issues**:
   * Capitaline Code might be stored as text in some rows, numbers in others
   * Company names might have trailing/leading spaces
2. **Column Mapping Issues**:
   * Excel column order might be different than expected
   * Column headers might not match exactly
3. **Data Filtering**:
   * Some rows might have NaN/null values that cause them to be dropped during processing

**To Debug:**

1. **First, run the debug function**:

excel

=debug\_companies(A1:G100)

This will show you exactly what companies are found in your data.

1. **Check the console output** when running the main function - it will now print detailed information about what companies are being processed.
2. **Common fixes to try**:
   * Make sure your Excel data doesn't have extra spaces in company names
   * Ensure Capitaline Code column contains only numbers
   * Check that all required columns are present and in the expected order

Run the debug function first and let me know what output you get. This will help identify exactly why "360 ONE" is missing from the Python results while appearing in Power Query.

I don't see that company in xlwings UDF server console outputs:

Number of groups: 481 Processing: Code=6, Company='ACC', Rows=237 Processing: Code=9, Company='Aegis Logistics', Rows=290 Processing: Code=23, Company='Apollo Tyres', Rows=169 Processing: Code=51, Company='Bajaj Electrical', Rows=743 Processing: Code=55, Company='Balkrishna Inds', Rows=59 Processing: Code=60, Company='Bata India', Rows=741 Processing: Code=64, Company='Berger Paints', Rows=528 Processing: Code=69, Company='Bharat Forge', Rows=743 Processing: Code=77, Company='Birla Corpn.', Rows=743 Processing: Code=82, Company='Blue Star', Rows=633 Processing: Code=92, Company='Borosil Renew.', Rows=494 Processing: Code=102, Company='Graphite India', Rows=331 Processing: Code=103, Company='Carborundum Uni.', Rows=743 Processing: Code=104, Company='CEAT', Rows=210 Processing: Code=105, Company='ITD Cem', Rows=226 Processing: Code=107, Company='A B Real Estate', Rows=505

That should have been very first company - 360 ONE

Edit

The issue is clear now! The company "360 ONE" with Capitaline Code 66008 is not appearing in the Python processing, even though it should be the first one (codes 6, 9, 23... but no 66008).

This suggests a **data filtering or conversion issue**. Let me add more specific debugging to catch this:

xlwings Python UDF Solution

Code ∙ Version 10

This enhanced debugging will help us identify exactly where the "360 ONE" data is getting lost. The most likely causes are:

1. **Data Type Conversion Issues**: The Capitaline Code 66008 might be stored as text and failing conversion to integer
2. **NaN Values**: Essential fields might have NaN values causing rows to be dropped
3. **Date Conversion Issues**: Date parsing might be failing for 360 ONE rows
4. **Column Mapping**: The data might not be reaching the function correctly

Run this enhanced version and check the console output. It will show:

* Whether 360 ONE data exists in the initial DataFrame
* What happens during each cleaning step
* Whether it survives the data type conversions
* If it makes it to the final grouping stage

This should pinpoint exactly where the data is being filtered out. Let me know what the detailed debug output shows!