

	Source Head	Source	Week	Qualified Leads
1	Digital	Facebook	1	954.4752186
2	Digital	Google	1	647.1807580
3	Digital	Affiliate	1	437.6618075
4	Digital	Digital Others	1	367
5	Digital	Live Chat	1	59
6	Digital	Youtube	1	1202

show a list of source and the cost

Analyzed ^

```
python
Always show details  Copy
import pandas as pd

# Load the uploaded Excel file
file_path = "/mnt/data/Clean Data.xlsx"
xls = pd.ExcelFile(file_path)

# Display sheet names to understand the structure of the file
xls.sheet_names

Result
['Sheet1']
```

Analyzed ^

```
python
Always show details  Copy
# Load the data from the first sheet
df = pd.read_excel(xls, sheet_name='Sheet1')

# Display the first few rows to understand the structure
df.head()

Result
   Source Head      Source  Week  Qualified Leads  Landed Conversions
0     Digital    Facebook    1      954.475219    188.153061
1     Digital      Google    1      647.180758    172.030612
2     Digital    Affiliate    1      437.661808    127.265306
3     Digital  Digital Others    1      367.000000      0.000000
4     Digital    Live Chat    1       59.000000      0.000000

Cost
0  226868
1  361681
2  303000
3      0
```

```
3      0
4      0
```

Analyzed ^

python Always show details Copy

```
# Select and display the required columns: Source and Cost
# Display the dataframe
import ace_tools as tools
tools.display_dataframe_to_user(name="Source and Cost Data", dataframe=source_cost_df)
```

Result

	Source	Cost
0	Facebook	226868
1	Google	361681
2	Affiliate	303000
3	Digital Others	0
4	Live Chat	0

Source And Cost Data

	Source	Cost
1	Facebook	226868
2	Google	361681
3	Affiliate	303000
4	Digital Others	0
5	Live Chat	0
6	Youtube	0

I have displayed the list of sources along with their associated costs for you. Let me know if you need any further analysis or modifications. [x]

remove the sources with 0 cost

Analyzed ^

python Always show details Copy

```
# Remove sources with 0 cost
filtered_source_cost_df = source_cost_df[source_cost_df['Cost'] > 0]
```

Display the updated dataframe

```
tools.display_dataframe_to_user(name="Filtered Source and Cost Data", dataframe=filtered_source_cost_df)
```

Result

	Source	Cost
0	Facebook	226868
1	Google	361681

```
2 Affiliate 303000
6 Facebook 436133
7 Google 502500
```

Filtered Source And Cost Data			
		Source	Cost
1	0	Facebook	226868
2	1	Google	361681
3	2	Affiliate	303000
4	6	Facebook	436133
5	7	Google	502500
6	8	Affiliate	251600

I have filtered out the sources with zero cost and displayed the updated data. Let me know if you need any further modifications or analysis. [x]

i want a list of unique sources

Analyzed ^

```
python
# Get the unique sources
unique_sources = filtered_source_cost_df['Source'].unique()

# Convert to DataFrame for better display
unique_sources_df = pd.DataFrame(unique_sources, columns=['Unique Sources'])

# Display the unique sources
tools.display_dataframe_to_user(name="Unique Sources List", dataframe=unique_sources_df)
```

Result

```
Unique Sources
0    Facebook
1     Google
2   Affiliate
3   Care World
4   Cinema TV
```

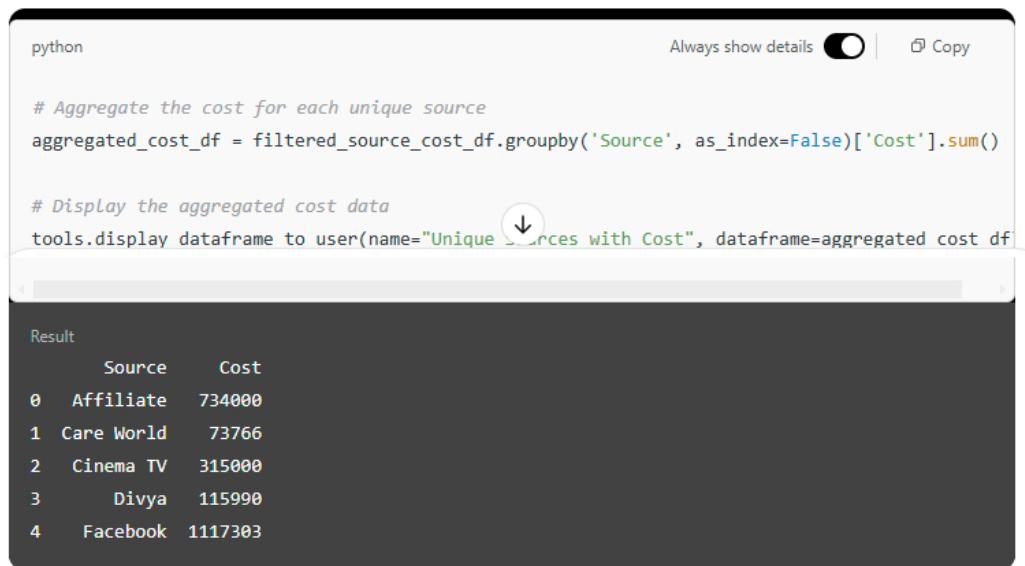
Unique Sources List

	Unique Sources
1	Facebook
2	Google
3	Affiliate
4	Care World
5	Cinema TV

I have displayed the list of unique sources for you. Let me know if you need any further refinements or analysis. [x]

along with their cost

Analyzed ^



```
python
# Aggregate the cost for each unique source
aggregated_cost_df = filtered_source_cost_df.groupby('Source', as_index=False)[['Cost']].sum()

# Display the aggregated cost data
tools.display_dataframe_to_user(name="Unique Sources with Cost", dataframe=aggregated_cost_df)
```

Result

	Source	Cost
0	Affiliate	734000
1	Care World	73766
2	Cinema TV	315000
3	Divya	115990
4	Facebook	1117303

Unique Sources With Cost		
	Source	Cost
1	Affiliate	734000
2	Care World	73766
3	Cinema TV	315000
4	Divya	115990
5	Facebook	1117303
6	Google	1160941

I have displayed the unique sources along with their aggregated costs. Let me know if you need any further analysis or modifications. [x]

let us start afresh
from the original data
group the total cost for all unique sources

Analyzed ^



```
python
# Reload the original data
df = pd.read_excel(xls, sheet_name='Sheet1')
```

```
# Group by 'Source' and sum the 'Cost' column
total_cost_per_source = df.groupby('Source', index=False)[['Cost']].sum()

# Display the grouped data
tools.display_dataframe_to_user(name="Total Cost per Source", dataframe=total_cost_per_source)
```

Result

	Source	Cost
0	Affiliate	734000
1	Care World	221298
2	Cinema TV	560000
3	Digital Others	0
4	Divya	347970

Total Cost Per Source		
	Source	Cost
10	News-18 Lokmat	318000
11	Sadhna X 2	1184400
12	TV-Show-Others	0
13	Tez	1225980
14	Youtube	0
15	Zee Hindustan	1162014

I have displayed the total cost for all unique sources from the original data. Let me know if you need any further modifications or analysis. [a-]

this is perfect
remove the 0 sources

Analyzed ^

```
python
# Remove sources with 0 cost
filtered_total_cost_df = total_cost_per_source[total_cost_per_source['Cost'] > 0]

# Display the updated data
tools.display_dataframe_to_user(name="Filtered Total Cost per Source", dataframe=filtered_total_cost_df)
```

Result

	Source	Cost
0	Affiliate	734000
1	Care World	221298
2	Cinema TV	560000
4	Divya	347970
5	Facebook	1117303

Filtered Total Cost Per Source

		Source	Cost	
6	6	Google	1160941	
7	8	News 24	1218000	
8	9	News-18 Lokmat	318000	↓
9	10	Jamia Zakir Hussain	1104400	
10	12	Tez	1225980	
11	14	Zee Hindustan	1162014	

I have removed the sources with zero cost and displayed the updated data. Let me know if you need any further analysis or modifications. [x]

also show the qualified lead and conversion columns

Analyzed ^

```
python
Always show details  Copy

# Group by 'Source' and sum the 'Cost', 'Qualified Leads', and 'Landed Conversions' columns
aggregated_data = df.groupby('Source', as_index=False).agg({
    'Cost': 'sum',
    'Qualified Leads': 'sum',
    'Landed Conversions': 'sum'
})

# Remove sources with 0 cost
filtered_aggregated_data = aggregated_data[aggregated_data['Cost'] > 0]

# Display the updated data
tools.display_dataframe_to_user(name="Aggregated Data per Source", dataframe=filtered_aggregated_data)
```

Result

	Source	Cost	Qualified Leads	Landed Conversions
0	Affiliate	734000	1128.621749	332.923201
1	Care World	221298	588.662932	167.632078
2	Cinema TV	560000	1097.271373	238.816307
4	Divya	347970	742.488662	168.019012
5	Facebook	1117303	3098.284257	677.452800

		Source	Cost	Qualified Leads
1	0	Affiliate	734000	1128.621749
2	1	Care World	221298	588.662932
3	2	Cinema TV	560000	1097.271372
4	4	Divya	347970	742.488662
5	5	Facebook	1117303	3098.284256
6	6	Google	1160941	2122.965314

I have displayed the aggregated data per source, including total cost, qualified leads, and landed conversions. Let me know if you need any further refinements or analysis. [x]

Analyzed ^

↓

python Always show details |

```
# Calculate CAC (Customer Acquisition Cost) as Cost per Landed Conversion
filtered_aggregated_data['CAC'] = filtered_aggregated_data['Cost'] / filtered_aggregated_data['Landed Conversions']

# Calculate Conversion Rate as (Landed Conversions / Qualified Leads) * 100
filtered_aggregated_data['Conversion Rate (%)'] = (filtered_aggregated_data['Landed Conversions'] / filtered_aggregated_data['Qualified Leads']) * 100

# Display the updated data
tools.display_dataframe_to_user(name="Aggregated Data with CAC and Conversion Rate", dataframe=filtered_aggregated_data)
```

STDOUT/STDERR

```
/tmp/ipykernel_12/3569805993.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-dc
    filtered_aggregated_data['CAC'] = filtered_aggregated_data['Cost'] / fi
/tmp/ipykernel_12/3569805993.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = val instead
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-dc

```
    filtered_aggregated_data['Conversion Rate (%)'] = (filtered_aggregated_
    ↓
```

Result

	Source	Cost	Qualified Leads	Landed Conversions	CAC
0	Affiliate	734000	1128.621749	332.923201	2204.712671
1	Care World	221298	588.662932	167.632078	1320.141123
2	Cinema TV	560000	1097.271373	238.816307	2344.898495
4	Divya	347970	742.488662	168.019012	2071.015630
5	Facebook	1117303	3098.284257	677.452800	1649.270620

Conversion Rate (%)

	Conversion Rate (%)
0	29.498209
1	28.476751
2	21.764562
4	22.629169
5	21.865418

↓

Aggregated Data With CAC And Conversion Rate

	Source	Cost	Qualified Leads
6	Google	1160941	2122.965314
7	News 24	1018000	2813.174217

		News Cat	Total Leads	Cost / %
8	9	News-18 Lokmat	318000	443.2958192
9	10	Sadhna X 2	1184400	3187.766330
10	12	Tez	1225980	4713.246520
11	14	Zee Hindustan	1162014	1792.222824

I have added the **Customer Acquisition Cost (CAC)** and **Conversion Rate (%)** columns to the dataset and displayed the updated data. Let me know if you need any further analysis or modifications. [x]

let us do min max normalization
before doing it show me how you plan to do it



Plan for Min-Max Normalization

Min-Max Normalization scales values to a range of **0 to 1** using the following formula:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

where:

- X is the original value,
- X_{min} is the minimum value in the column,
- X_{max} is the maximum value in the column.

Steps to Implement:

1. Identify the numeric columns for normalization:

- Cost
- Qualified Leads
- Landed Conversions
- CAC
- Conversion Rate (%)



2. Apply Min-Max Normalization to these columns.
3. Retain categorical columns (`Source`) so that only numerical values are transformed.
4. Display the final normalized dataset.

Would you like to keep the original values alongside the normalized values, or should I replace them?

i want you to apply it to cost, conversion and cac
but first tell me how will you write the formulas

Min-Max Normalization Formulas for Selected Columns

We will apply the **Min-Max Normalization** formula:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

to the following columns:

1. Cost Normalized:

$$\text{Cost}_{\text{norm}} = \frac{\text{Cost} - \min(\text{Cost})}{\max(\text{Cost}) - \min(\text{Cost})}$$

2. CAC Normalized (Customer Acquisition Cost):

$$\text{CAC}_{\text{norm}} = \frac{\text{CAC} - \min(\text{CAC})}{\max(\text{CAC}) - \min(\text{CAC})}$$

3. Conversion Rate Normalized:

$$\text{ConversionRate}_{\text{norm}} = \frac{\text{Conversion Rate} - \min(\text{Conversion Rate})}{\max(\text{Conversion Rate}) - \min(\text{Conversion Rate})}$$

Let me know if you're good with this approach, and I'll apply the normalization now.

this looks good
but a higher CAC and higher cost are bad
but higher conversion rate is good

< 2/2 >

You're absolutely right! Since **higher CAC and higher Cost are bad** (we want them lower), and **higher Conversion Rate is good** (we want it higher), we need to modify the normalization approach accordingly:

1. For Cost and CAC (where lower is better):

- We will use **inverse normalization** so that lower values get higher scores:

$$\text{Cost}_{\text{norm}} = 1 - \frac{\text{Cost} - \min(\text{Cost})}{\max(\text{Cost}) - \min(\text{Cost})}$$

$$\text{CAC}_{\text{norm}} = 1 - \frac{\text{CAC} - \min(\text{CAC})}{\max(\text{CAC}) - \min(\text{CAC})}$$

- This way, **lower cost and CAC will have higher scores (better)**, and **higher ones will have lower scores (worse)**.

2. For Conversion Rate (where higher is better):

- We keep the standard Min-Max normalization:

$$\text{ConversionRate}_{\text{norm}} = \frac{\text{Conversion Rate} - \min(\text{Conversion Rate})}{\max(\text{Conversion Rate}) - \min(\text{Conversion Rate})}$$

- This ensures **higher conversion rates get higher scores**.

Now, I'll apply this logic and display the results.

Analyzed ^

```
python
# Apply Min-Max Normalization

# Normalization for Cost and CAC (inverted because lower is better)
filtered_aggregated_data['Cost_norm'] = 1 - (filtered_aggregated_data['Cost'] - filtered_aggregated_data['Cost'].min()) / (filtered_aggregated_data['Cost'].max() - filtered_aggregated_data['Cost'].min())
filtered_aggregated_data['CAC_norm'] = 1 - (filtered_aggregated_data['CAC'] - filtered_aggregated_data['CAC'].min()) / (filtered_aggregated_data['CAC'].max() - filtered_aggregated_data['CAC'].min())

# Normalization for Conversion Rate (higher is better, so normal Min-Max is used)
filtered_aggregated_data['ConversionRate_norm'] = (filtered_aggregated_data['Conversion Rate'] - filtered_aggregated_data['Conversion Rate'].min()) / (filtered_aggregated_data['Conversion Rate'].max() - filtered_aggregated_data['Conversion Rate'].min())

Always show details  Copy
```

```
# Display the normalized dataset
tools.display_dataframe_to_user(name="Normalized Aggregated Data", dataframe=filtered_aggrega
```

STDOUT/STDERR

```
/tmp/ipykernel_12/1148537863.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-dc
    filtered_aggregated_data['Cost_norm'] = 1 - (filtered_aggregated_data['
/tmp/ipykernel_12/1148537863.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-dc
    filtered_aggregated_data['CAC_norm'] = 1 - (filtered_aggregated_data['C
/tmp/ipykernel_12/1148537863.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-dc
    filtered_aggregated_data['ConversionRate_norm'] = (filtered_aggregated_
```

Result

	Source	Cost	Qualified Leads	Landing Page	Conversions	CAC
0	Affiliate	754000	1128.021749	552.923201	2204.712071	
1	Care World	221298	588.662932	167.632078	1320.141123	
2	Cinema TV	560000	1097.271373	238.816307	2344.898495	
4	Divya	347970	742.488662	168.019012	2071.015630	
5	Facebook	1117303	3098.284257	677.452800	1649.270620	

```
    Conversion Rate (%)  Cost_norm  CAC_norm  Conversion Rate_norm \
0           29.498209  0.489687  0.481647          0.957236
1           28.476751  1.000000  0.984026          0.852838
2           21.764562  0.662876  0.402031          0.166815
4           22.629169  0.873918  0.557578          0.255182
5           21.865418  0.108171  0.797102          0.177123
```

```
    ConversionRate_norm
0           0.957236
1           0.852838
2           0.166815
4           0.255182
5           0.177123
```

Normalized Aggregated Data

		Source	Cost	Qualified Leads
8	9	News-18 Lokmat	318000	443.2958192
9	10	Sadhna X 2	1184400	3187.766330
10	12	Tez	1225980	4713.246520

I have applied Min-Max normalization with **inverse scaling for Cost and CAC** (lower is better) and **direct scaling for Conversion Rate** (higher is better). The normalized values are now displayed alongside the original data. Let me know if you need any further modifications or insights! [x]

this is perfect

↓ just realized that we had removed the 0 sources
can you not do that and repeat all the steps

Analyzed ^

```
python Always show details ⚡ | ⌂ Copy

# Reload the original data without removing 0-cost sources
df = pd.read_excel(xls, sheet_name='Sheet1')

# Group by 'Source' and aggregate necessary columns
aggregated_data = df.groupby('Source', as_index=False).agg({
    'Cost': 'sum',
    'Qualified Leads': 'sum',
    'Landed Conversions': 'sum'
})

# Calculate CAC (Customer Acquisition Cost) as Cost per Landed Conversion
aggregated_data['CAC'] = aggregated_data['Cost'] / aggregated_data['Landed Conversions']

# Calculate Conversion Rate as (Landed Conversions / Qualified Leads) * 100
aggregated_data['Conversion Rate (%)'] = (aggregated_data['Landed Conversions'] / aggregated_data['Qualified Leads']) * 100

# Apply Min-Max Normalization for all sources including those with 0 cost
aggregated_data['Cost_norm'] = 1 - (aggregated_data['Cost'] - aggregated_data['Cost'].min()) / (aggregated_data['Cost'].max() - aggregated_data['Cost'].min())
aggregated_data['CAC_norm'] = 1 - (aggregated_data['CAC'] - aggregated_data['CAC'].min()) / (aggregated_data['CAC'].max() - aggregated_data['CAC'].min())
aggregated_data['ConversionRate_norm'] = (aggregated_data['Conversion Rate (%)'] - aggregated_data['Conversion Rate (%)'].min()) / (aggregated_data['Conversion Rate (%)'].max() - aggregated_data['Conversion Rate (%)'].min())

# Display the final dataset including sources with 0 cost
tools.display_dataframe_to_user(name="Normalized Aggregated Data (Including 0 Cost Sources)",
```

STDOUT/STDERR

```
/home/sandbox/.local/lib/python3.11/site-packages/pandas/core/internals/t
values = values.astype(str)
```

Result

	Source	Cost	Qualified Leads	Landed Conversions	C
0	Affiliate	734000	1128.621749	332.923201	2204.7126
1	Care World	221298	588.662932	167.632078	1320.1411
2	Cinema TV	560000	1097.271373	238.816307	2344.8984
3	Digital Others	0	1034.000000	0.000000	N
4	Divya	347970	742.488662	168.019012	2071.0156

	Conversion Rate (%)	Cost_norm	CAC_norm	ConversionRate_norm
0	29.498209	0.401295	0.481647	0.986014
1	28.476751	0.819493	0.984026	0.951871
2	21.764562	0.543223	0.402031	0.727507
3	0.000000	1.000000	NaN	0.000000
4	22.629169	0.716170	0.557578	0.756408

Normalized Aggregated Data (Including 0 Cost Sources)				
	e (%)	Cost_norm	CAC_norm	ConversionRate_norm
1	5320	0.4012952902983735	0.481647271883921	0.986014164635271
2	5874	0.8194929770469339	0.9840259309490929	0.9518706743025067
3	1419	0.5432225648052986	0.4020309123862572	0.7275074370991372
4		1.0		0.0
5	5215	0.7161699212058924	0.5575784275746119	0.7564079890795029
1985		0.088615002365451	0.797101919718183	0.730878672706215

I have now included all sources, even those with **zero cost**, and repeated all the previous steps, including normalization. Let me know if you need any further refinements or insights! [→]

thanks

now add weights to normalized values
30% to conv, 30% to cac and 40% to cost

Analyzed ^

```
python
# Apply weights to normalized values
aggregated_data['Weighted_Score'] = (
    (aggregated_data['ConversionRate_norm'] * 0.30) +
    (aggregated_data['CAC_norm'] * 0.30) +
    (aggregated_data['Cost_norm'] * 0.40)
)

# Display the updated dataset with weighted scores
tools.display_dataframe_to_user(name="Weighted Score Data", dataframe=aggregated_data)

STDOUT/STDERR
/home/sandbox/.local/lib/python3.11/site-packages/pandas/core/internals/b
values = values.astype(str)

Result
      Source   Cost  Qualified Leads  Landed Conversions      C
0   Affiliate  734000       1128.621749     332.923201  2204.7126
1   Care World  221298        588.662932     167.632078  1320.1411
2   Cinema TV  560000       1097.271373     238.816307  2344.8984
3   Digital Others      0.000000       0.000000      N
4      Divya  347970       742.488662     168.019012  2071.0156

      Conversion Rate (%)  Cost_norm  CAC_norm  ConversionRate_norm \
0            29.498209  0.401295  0.481647           0.986014
1            28.476751  0.819493  0.984026           0.951871
2            21.764562  0.543223  0.402031           0.727507
3            0.000000  1.000000      NaN           0.000000
4            22.629169  0.716170  0.557578           0.756408
```

	Weighted_Score
0	0.600817
1	0.908566
2	0.556151
3	NaN
4	0.680664

Weighted Score Data				
	Source	Cost	Qualified Leads	Landed Conv
11	Sadhna X 2	1184400	3187.766330836337	723.3720307
12	TV-Show-Others	0	2839.0	0.0
13	Tez	1225980	4713.24652095176	948.8902676
14	Youtube	0	3514.0	0.0
15	Zee Hindustan	1162014	175 ↓ .2824652488	380.6411162

I have applied the **weighted scoring** using 40% weight for Cost, 30% for CAC, and 30% for Conversion Rate. The updated dataset with the **Weighted Score** is now available. Let me know if you need any further analysis! [→]

combine all this into 1 final code

Analyzed ^

```
python
# Reload the original data without removing 0-cost sources
df = pd.read_excel(xls, sheet_name='Sheet1')

# Group by 'Source' and aggregate necessary columns
aggregated_data = df.groupby('Source', as_index=False).agg({
    'Cost': 'sum',
    'Qualified Leads': 'sum',
    'Landed Conversions': 'sum'
})

# Calculate CAC (Customer Acquisition Cost) as Cost per Landed Conversion
aggregated_data['CAC'] = aggregated_data['Cost'] / aggregated_data['Landed Conversions']

# Calculate Conversion Rate as (Landed Conversions / Qualified Leads) * 100
aggregated_data['Conversion Rate (%)'] = (aggregated_data['Landed Conversions'] / aggregated_data['Qualified Leads']) * 100

# Apply Min-Max Normalization for all sources including those with 0 cost
aggregated_data['Cost_norm'] = 1 - (aggregated_data['Cost'] - aggregated_data['Cost'].min()) / (aggregated_data['Cost'].max() - aggregated_data['Cost'].min())
aggregated_data['CAC_norm'] = 1 - (aggregated_data['CAC'] - aggregated_data['CAC'].min()) / (aggregated_data['CAC'].max() - aggregated_data['CAC'].min())
aggregated_data['ConversionRate_norm'] = (aggregated_data['Conversion Rate (%)']) - aggregated_data['Conversion Rate (%)'].min()

# Apply weights to normalized values: 40% Cost, 30% CAC, 30% Conversion Rate
aggregated_data['Weighted_Score'] = (
    aggregated_data['ConversionRate_norm'] * 0.30) +
    (aggregated_data['CAC_norm'] * 0.30) +
    (aggregated_data['Cost_norm'] * 0.40)
```

```

        (aggregated_data['Cost_norm'] * 0.40)
    )

# Display the final dataset
tools.display_dataframe_to_user(name="Final Processed Data", dataframe=aggregated_data)

```

STDOUT/STDERR

```

/home/sandbox/.local/lib/python3.11/site-packages/pandas/core/internals/b
    values = values.astype(str)

```

Result

	Source	Cost	Qualified Leads	Landed Conversions	C
0	Affiliate	734000	1128.621749	332.923201	2204.7126
1	Care World	221298	588.662932	167.632078	1320.1411
2	Cinema TV	560000	1097.271373	238.816307	2344.8984
3	Digital Others	0	1034.000000	0.000000	N
4	Divya	347970	742.488662	168.019012	2071.0156

	Conversion Rate (%)	Cost_norm	CAC_norm	ConversionRate_norm
0	29.498209	0.401295	0.481647	0.986014
1	28.476751	0.819493	0.984026	0.951871
2	21.764562	0.543223	0.402031	0.727507
3	0.000000	1.000000	NaN	0.000000
4	22.629169	0.716170	0.557578	0.756408

	Weighted_Score
0	0.600817
1	0.908566
2	0.556151
3	NaN
4	0.680664

Final Processed Data				
	Source	Cost	Qualified Leads	Landed Conv
11	Sadhna X 2	1184400	3187.766330836337	723.3720307
12	TV-Show-Others	0	2839.0	0.0
13	Tez	1225980	4713.24652095176	948.8902676
14	Youtube	0	3514.0	0.0
15	Zee Hindustan	1162014	179.2824652488	380.64111623

I have combined all steps into one final code execution, ensuring that all sources (including those with 0 cost) are included. The final dataset with cost, CAC, conversion rate, normalized values, and weighted scores is now available. Let me know if you need any further refinements or insights! [→]

this is perfect

convert it to a code i can use in google colab

Here's the full Python code you can use in Google Colab to process your dataset exactly as we did here.

This code will:

- Load an Excel file from Google Drive or local upload
- Perform aggregation, CAC & conversion rate calculation
- Apply Min-Max Normalization (with inverse normalization for cost & CAC)
- Calculate weighted scores
- Display the processed DataFrame



You can copy and run this directly in [Google Colab](#):

```
</> Google Colab Data Processing

47     (aggregated_data['CAC_norm'] * 0.30) +
48     (aggregated_data['Cost_norm'] * 0.40)
49 )
50
51 # Display the processed DataFrame
52 from IPython.display import display
53 display(aggregated_data)
54
55 # Save to a new Excel file
56 download_filename = "Processed_Data.xlsx"
57 aggregated_data.to_excel(download_filename, index=False)
58 print(f"File saved as {download_filename}. You can download it from the files section")
59
```

This Python script is now ready to be used in [Google Colab](#). It will:

1. Prompt you to upload an Excel file.
2. Process the data by aggregating costs, calculating CAC & conversion rates, applying normalization, and computing weighted scores.
3. Display the processed data.
4. Save and allow you to download the final processed Excel file.

Let me know if you need any modifications! ↗

thanks

give me the tableau formulas required for the noramlziation and weights

Here are the Tableau formulas for Normalization and Weighted Score Calculation:

1. Min-Max Normalization

We normalize Cost (inverse), CAC (inverse), and Conversion Rate (direct):



Cost Normalization (Lower is Better)

sql

Copy Edit

```
1 - ( [Cost] - WINDOW_MIN([Cost]) ) / ( WINDOW_MAX([Cost]) - WINDOW_MIN([Cost]) )
```

CAC Normalization (Lower is Better)

```
sql
```

Copy Edit

```
1 - ( [CAC] - WINDOW_MIN([CAC]) ) / ( WINDOW_MAX([CAC]) - WINDOW_MIN([CAC]) )
```

Conversion Rate Normalization (Higher is Better)

```
sql
```

Copy Edit

```
( [Conversion Rate] - WINDOW_MIN([Conversion Rate]) ) / ( WINDOW_MAX([Conversion Rate]) - WIN
```



2. Weighted Score Calculation

We apply 40% weight to Cost, 30% to CAC, and 30% to Conversion Rate:

```
sql
```

Copy Edit

```
([Conversion Rate Normalized] * 0.30) + ([CAC Normalized] * 0.30) + ([Cost Normalized] * 0.40)
```

How to Use in Tableau

1. Create calculated fields for each normalization formula.
2. Use a Table Calculation:
 - Go to Edit Table Calculation.
 - Choose "Compute using Table (or Specific Dimension)".
 - This ensures values are computed dynamically across the dataset.
3. Create a final calculated field for Weighted Score.

This will ensure that lower cost and CAC get better scores, and higher conversion rates are rewarded.

Let me know if you need any clarifications!

Ask anything



Search



ChatGPT can make mistakes. OpenAI doesn't use Havish M Consulting workspace data to train its models.

?