In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [ ]:
```
search = pd.read_csv('/content/drive/My Drive/project dataset/google_search_data.csv')
search
```

Out[ ]:

	date	platform	searchVolume	Claim_ID	week_number	year_new
0	05-01-2014	google	349	916	1	2014
1	06-01-2014	google	349	916	2	2014
2	07-01-2014	google	697	916	2	2014
3	10-01-2014	google	349	916	2	2014
4	20-01-2014	google	697	916	4	2014
...	...	...	...	...	...	...
181560	05-09-2019	walmart	125	980	36	2019
181561	22-09-2019	walmart	84	980	38	2019
181562	26-09-2019	walmart	42	980	39	2019
181563	15-07-2019	walmart	42	622	29	2019
181564	02-09-2019	walmart	42	689	36	2019

181565 rows × 6 columns

In [ ]:
```
manufactur = pd.read_csv('/content/drive/My Drive/project dataset/product_manufacturer_list.csv')
manufactur
```

Out[ ]:

	PRODUCT_ID	Vendor
0	1	Others
1	2	Others
2	3	Others
3	4	Others
4	5	Others
...	...	...
67170	67171	Private Label
67171	67172	Private Label
67172	67173	Private Label
67173	67174	Private Label
67174	67175	Private Label

67175 rows × 2 columns

In [ ]:
```
sales = pd.read_csv('/content/drive/My Drive/project dataset/sales_data.csv')
sales
```

Out[ ]:

	system_calendar_key_N	product_id	sales_dollars_value	sales_units_value	sales_lbs_value
0	20160109	1	13927.0	934	18680
1	20160109	3	10289.0	1592	28646
2	20160109	4	357.0	22	440
3	20160109	6	23113.0	2027	81088
4	20160109	7	23177.0	3231	58164
...	...	...	...	...	...
4526177	20181027	47536	8.0	2	3
4526178	20181027	47539	391.0	39	68
4526179	20181027	47543	105.0	59	48
4526180	20181027	47544	3720.0	1246	4361
4526181	20181027	47545	1729.0	2016	378

4526182 rows × 5 columns

In [ ]:
```
sm = pd.read_csv('/content/drive/My Drive/project dataset/social_media_data.csv')
sm
```

Out[ ]:

	Theme Id	published_date	total_post
0	148.0	10/1/2015	76
1	148.0	10/10/2015	31
2	148.0	10/11/2015	65
3	148.0	10/12/2015	88
4	148.0	10/13/2015	85
...	...	...	...
533385	876.0	9/5/2019	4658
533386	876.0	9/6/2019	3731
533387	876.0	9/7/2019	2336
533388	876.0	9/8/2019	1374
533389	876.0	9/9/2019	1442

533390 rows × 3 columns

In [ ]:
```
theme = pd.read_csv('/content/drive/My Drive/project dataset/Theme_list.csv')
theme
```

Out[ ]:

	CLAIM_ID	Claim Name
0	0	No Claim
1	8	low carb
2	15	beans
3	16	cocoa
4	26	vanilla
...	...	...
203	508	cola
204	769	shortbread
205	949	passion fruit
206	521	blood orange
207	876	caramel

208 rows × 2 columns

In [ ]:
```
themeprod = pd.read_csv('/content/drive/My Drive/project dataset/Theme_product_list.csv')
themeprod
```

Out[ ]:

	PRODUCT_ID	CLAIM_ID
0	26	8
1	29	8
2	48	81
3	50	81
4	74	227
...	...	...
91480	8158	0
91481	45183	0
91482	25690	0
91483	46085	0
91484	34907	0

91485 rows × 2 columns

Shape before clean

In [ ]:
```
search.shape
```

Out[ ]:
```
(181565, 6)
```

In [ ]:
```
manufactur.shape
```

Out[ ]:
```
(67175, 2)
```

In [ ]:
```
sales.shape
```
Out[ ]:
```
(4526182, 5)
```
In [ ]:
```
sm.shape
```
Out[ ]:
```
(533390, 3)
```
In [ ]:
```
theme.shape
```
Out[ ]:
```
(208, 2)
```
In [ ]:
```
themeprod.shape
```
Out[ ]:
```
(91485, 2)
```

Task 1

For this task we will individually find out the number of missing and duplicated data in all datasets and print out the percentages of them

In [ ]:
```
search.isnull().sum()
```
Out[ ]:
```
date            0
platform        0
searchVolume    0
Claim_ID        0
week_number     0
year_new        0
dtype: int64
```
In [ ]:
```
manufactur.isnull().sum()
```
Out[ ]:
```
PRODUCT_ID    0
Vendor        0
dtype: int64
```
In [ ]:
```
sales.isnull().sum()
```
Out[ ]:
```
system_calendar_key_N    0
product_id               0
sales_dollars_value      0
sales_units_value        0
sales_lbs_value          0
dtype: int64
```
In [ ]:
```
sm.isnull().sum()
```
Out[ ]:
```
Theme Id          218511
published_date         0
total_post             0
dtype: int64
```
In [ ]:
```
theme.isnull().sum()
```
Out[ ]:
```
CLAIM_ID      0
Claim Name    0
dtype: int64
```
In [ ]:
```
themeprod.isnull().sum()
```
Out[ ]:
```
PRODUCT_ID    0
CLAIM_ID      0
dtype: int64
```

In [ ]:
```
dups_search = search.duplicated().sum()
print(dups_search)
```

```
40
```

In [ ]:
```
dups_manu = manufactur.duplicated().sum()
print(dups_manu)
```

```
0
```

In [ ]:
```
dups_sales = sales.duplicated().sum()
print(dups_sales)
```

```
0
```

In [ ]:
```
dups_sm = sm.duplicated().sum()
print(dups_sm)
```

```
26299
```

In [ ]:
```
dups_theme = theme.duplicated().sum()
print(dups_theme)
```

```
0
```

In [ ]:
```
dups_themeprod = themeprod.duplicated().sum()
print(dups_themeprod)
```

```
0
```

In [ ]:
```
perct = (dups_search/len(search))*100
print('The percentage of duplicated data is ' ,perct)
```

```
The percentage of duplicated data is  0.022030677718723322
```

In [ ]:
```
perct = (search.isnull().sum()/len(search))*100
print('The percentage of missing data is\n' ,perct)
```

```
The percentage of missing data is
 date            0.0
platform        0.0
searchVolume    0.0
Claim_ID        0.0
week_number     0.0
year_new        0.0
dtype: float64
```

In [ ]:
```
perct = ((dups_sales)/len(sales))*100
print('The percentage of duplicated data is' ,perct)
```

```
The percentage of duplicated data is 0.0
```

In [ ]:
```
perct = (sales.isnull().sum()/len(sales))*100
print('The percentage of missing data is\n' ,perct)
```

```
The percentage of missing data is
 system_calendar_key_N    0.0
product_id               0.0
sales_dollars_value      0.0
sales_units_value        0.0
sales_lbs_value          0.0
dtype: float64
```

In [ ]:
```
percit = (dups_sm/len(sm))*100
print('The percentage of duplicated data is' ,percit)
```

```
The percentage of duplicated data is 4.930538630270534
```

In [ ]:
```
perciti = (sm.isnull().sum()/len(sm))*100
print('The percentage of missing data is\n' ,perciti)
```

```
The percentage of missing data is
 Theme Id          40.96646
published_date     0.00000
total_post         0.00000
dtype: float64
```

In [ ]:
```
percit = (dups_manu/len(manufactur))*100
print('The percentage of duplicated data is' ,percit)
```

```
The percentage of duplicated data is 0.0
```

In [ ]:
```
perciti = (manufactur.isnull().sum()/len(manufactur))*100
print('The percentage of missing data is\n' ,perciti)
```

```
The percentage of missing data is
 PRODUCT_ID    0.0
Vendor        0.0
dtype: float64
```

In [ ]:
```
percit = (dups_theme/len(theme))*100
print('The percentage of duplicated data is' ,percit)
```

```
The percentage of duplicated data is 0.0
```

In [ ]:
```
perciti = (theme.isnull().sum()/len(theme))*100
print('The percentage of missing data is\n' ,perciti)
```

```
The percentage of missing data is
 CLAIM_ID      0.0
Claim Name    0.0
dtype: float64
```

In [ ]:
```
percit = (dups_themeprod/len(themeprod))*100
print('The percentage of duplicated data is' ,percit)
```

```
The percentage of duplicated data is 0.0
```

In [ ]:
```
perciti = (themeprod.isnull().sum()/len(themeprod))*100
print('The percentage of missing data is\n' ,perciti)
```

```
The percentage of missing data is
 PRODUCT_ID    0.0
CLAIM_ID      0.0
dtype: float64
```

Shape after clean

In [ ]:
```
clean = sm.drop_duplicates(inplace = True ,keep = False)
drop = sm.dropna(inplace=True)
sm
```

Out[ ]:

	Theme Id	published_date	total_post
0	148.0	10/1/2015	76
1	148.0	10/10/2015	31
2	148.0	10/11/2015	65
3	148.0	10/12/2015	88
4	148.0	10/13/2015	85
...	...	...	...
533385	876.0	9/5/2019	4658
533386	876.0	9/6/2019	3731
533387	876.0	9/7/2019	2336
533388	876.0	9/8/2019	1374
533389	876.0	9/9/2019	1442

314873 rows × 3 columns

In [ ]:

```
clean = search.drop_duplicates(inplace = True ,keep = False)
search
```

Out[ ]:

	date	platform	searchVolume	Claim_ID	week_number	year_new
0	05-01-2014	google	349	916	1	2014
1	06-01-2014	google	349	916	2	2014
2	07-01-2014	google	697	916	2	2014
3	10-01-2014	google	349	916	2	2014
4	20-01-2014	google	697	916	4	2014
...	...	...	...	...	...	...
181560	05-09-2019	walmart	125	980	36	2019
181561	22-09-2019	walmart	84	980	38	2019
181562	26-09-2019	walmart	42	980	39	2019
181563	15-07-2019	walmart	42	622	29	2019
181564	02-09-2019	walmart	42	689	36	2019

181485 rows × 6 columns

In [ ]:

```
sales.shape
```

Out[ ]:

```
(4526182, 5)
```

In [ ]:

```
search.shape
```

Out[ ]:

```
(181485, 6)
```

In [ ]:

```
search.rename(columns={
    'Claim_ID':'Theme_ID'
},inplace=True)
search
```

Out[ ]:

	date	platform	searchVolume	Theme_ID	week_number	year_new
0	05-01-2014	google	349	916	1	2014
1	06-01-2014	google	349	916	2	2014
2	07-01-2014	google	697	916	2	2014
3	10-01-2014	google	349	916	2	2014
4	20-01-2014	google	697	916	4	2014
...	...	...	...	...	...	...
181560	05-09-2019	walmart	125	980	36	2019
181561	22-09-2019	walmart	84	980	38	2019
181562	26-09-2019	walmart	42	980	39	2019
181563	15-07-2019	walmart	42	622	29	2019
181564	02-09-2019	walmart	42	689	36	2019

181485 rows × 6 columns

In [ ]:
```
sm.rename(columns={
    'Theme Id':'Theme_ID'
},inplace=True)
```

In [ ]:
```
theme.rename(columns={
    'CLAIM_ID':'Theme_ID'
},inplace=True)
```

In [ ]:
```
themeprod.rename(columns={
    'CLAIM_ID':'Theme_ID'
},inplace=True)
```

In [ ]:
```
sales.rename(columns={
    'product_id':'PRODUCT_ID'
},inplace=True)
```

Task2

In [ ]:
```
# Task 2.1: Number of themes present
num_themesis = len(theme)
print('The number of themes present is',num_themesis)
```

```
The number of themes present is 208
```

In [ ]:
```
# Task 2.2(a): Number of themes mentioned in social media
num_themes_social_media = len(sm['Theme_ID'].unique())
print('The number of themes mentioned in social media is',num_themes_social_media)
#Task 2.2 (b): how may themes have no publicity
publicized_themes = sm['Theme_ID'].unique()
all_themes = theme['Theme_ID'].unique()
not_publicized_themes = [theme_id for theme_id in all_themes if theme_id not in publicized_themes]

print("Themes that do not get publicity:")
for theme_id in not_publicized_themes:
    theme_name = theme[theme['Theme_ID'] == theme_id]['Claim Name'].values[0]
    print(theme_name)
```

```
The number of themes mentioned in social media is 193
Themes that do not get publicity:
No Claim
cocoa
stroganoff
buckwheat
tutti frutti
brown ale
whitebait
french
cookie
pollock
pizza
american southwest style
tilapia
apple cinnamon
dha
```

In [ ]:
```
# Task 2.3: Number of themes searched in google
google_data = search[search['platform'] == 'google']
num_themes_google = google_data['Theme_ID'].nunique()
print('The number of themes searched in google is',num_themes_google)
```

```
The number of themes searched in google is 159
```

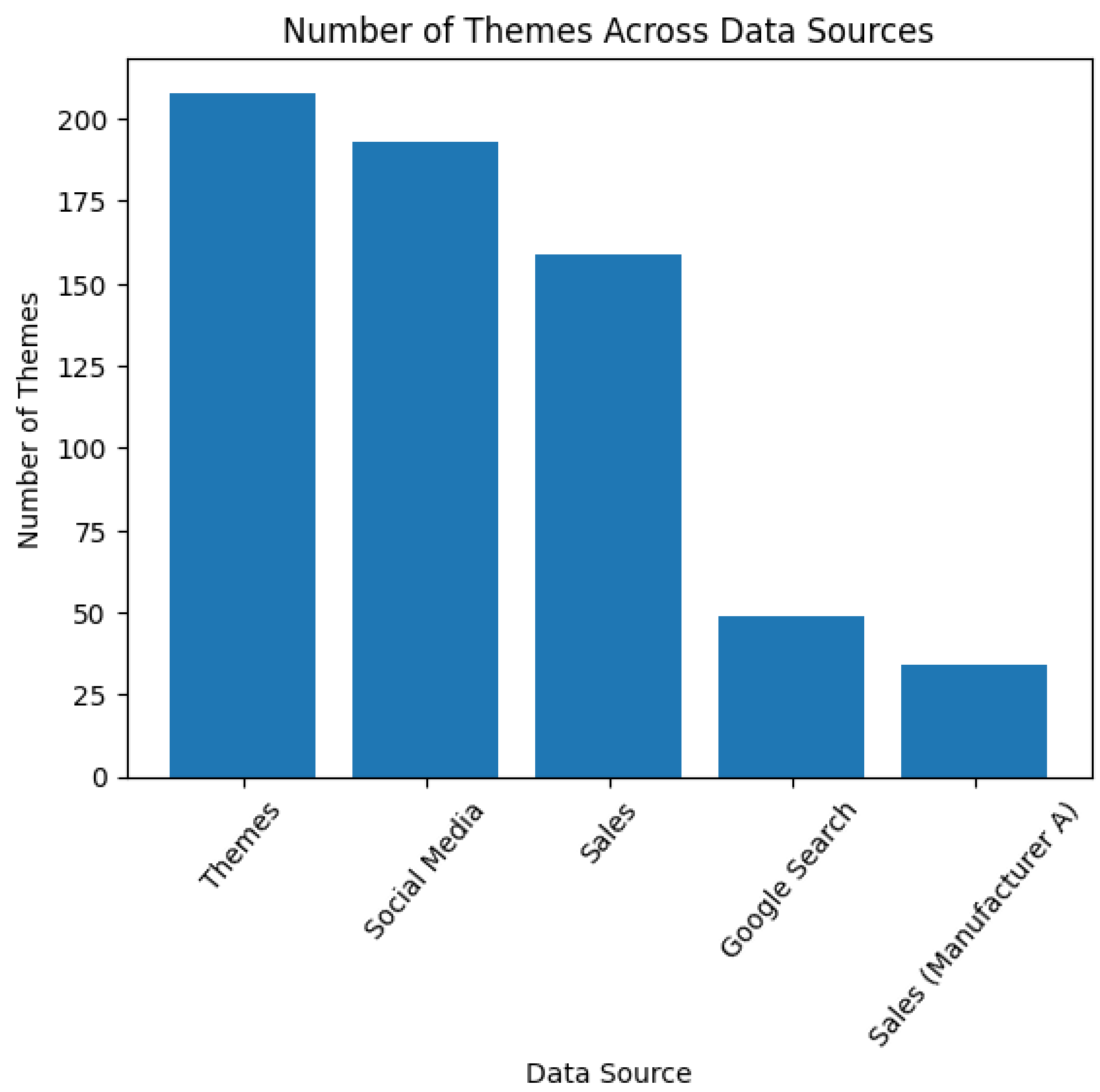
In [ ]:
```
# Task 2.4: Number of themes contributing to sales
#merging these 3 datasets allows us to easily filter out the number of themes in the sales dataset
df = pd.merge(themeprod, sales, on='PRODUCT_ID', how='inner')
df2 = pd.merge(df,theme, on='Theme_ID',how='inner')
num_themes_sales = len(df['Theme_ID'].unique())
print('The number of themes contributing to sales',num_themes_sales)
```

```
The number of themes contributing to sales 49
```

In [ ]:
```
# Task 2.5: Number of themes Manufacturer A has business in
d1 = pd.merge(manufactur,themeprod, on='PRODUCT_ID', how='inner')
num_themes_manufacturer_A = len(d1[d1['Vendor'] == 'A']['Theme_ID'].unique())
print('The number of themes Manufacturer A has business in is',num_themes_manufacturer_A)
```

```
The number of themes Manufacturer A has business in is 34
```

In [ ]:
```
# Create a bar plot to show the results of tasks 2.1 to 2.5
data_sources = ['Themes', 'Social Media','Sales','Google Search', 'Sales (Manufacturer A)']
num_themes = [num_themesis, num_themes_social_media, num_themes_google, num_themes_sales, num_themes_manufacturer_A]
#data_sources is a summarized version of the tasks output
#num_themes is the variables that conatin the answers
plt.bar(data_sources, num_themes)
plt.xlabel('Data Source')
plt.ylabel('Number of Themes')
plt.title('Number of Themes Across Data Sources')
plt.xticks(rotation=50)
plt.show()
```



Task 3

For task 3 i feel that it is best to leave out theme_id 0 also known as no claim as there is no definitive theme anme for no claim thus making it redundant in the analysis of this task

In [ ]:
```
# Task 3.1: Top 5 themes preferred by consumers as per sales
top_5_themes_sales = df2[df2['Theme_ID'] != 0].groupby('Claim Name')['sales_units_value'].sum().nlargest(5)
print('The top 5 themes are: \n', top_5_themes_sales)
```

```
The top 5 themes are:
 Claim Name
low carb                      2627421936
no additives/preservatives    1559993572
stroganoff                    1391218733
apple cinnamon                 805116403
soy foods                      449104186
Name: sales_units_value, dtype: int64
```

In [ ]:
```
# Task 3.2: Top 5 themes preferred by consumers as per social media data
#by merging sm and theme dataset we can easily accomplish the task
df3 = pd.merge(theme,sm,on='Theme_ID',how='inner')
top_5_themes_social_media = df3[df3['Theme_ID'] != 0].groupby('Claim Name')['total_post'].sum().nlargest(5)
print('The top 5 themes preferred by customers are: \n',top_5_themes_social_media)
```

```
The top 5 themes preferred by customers are:
 Claim Name
health (passive)    5329592
boar                4609405
rabbit              2821690
probiotic           2462718
pumpkin             2417031
Name: total_post, dtype: int64
```

In [ ]:

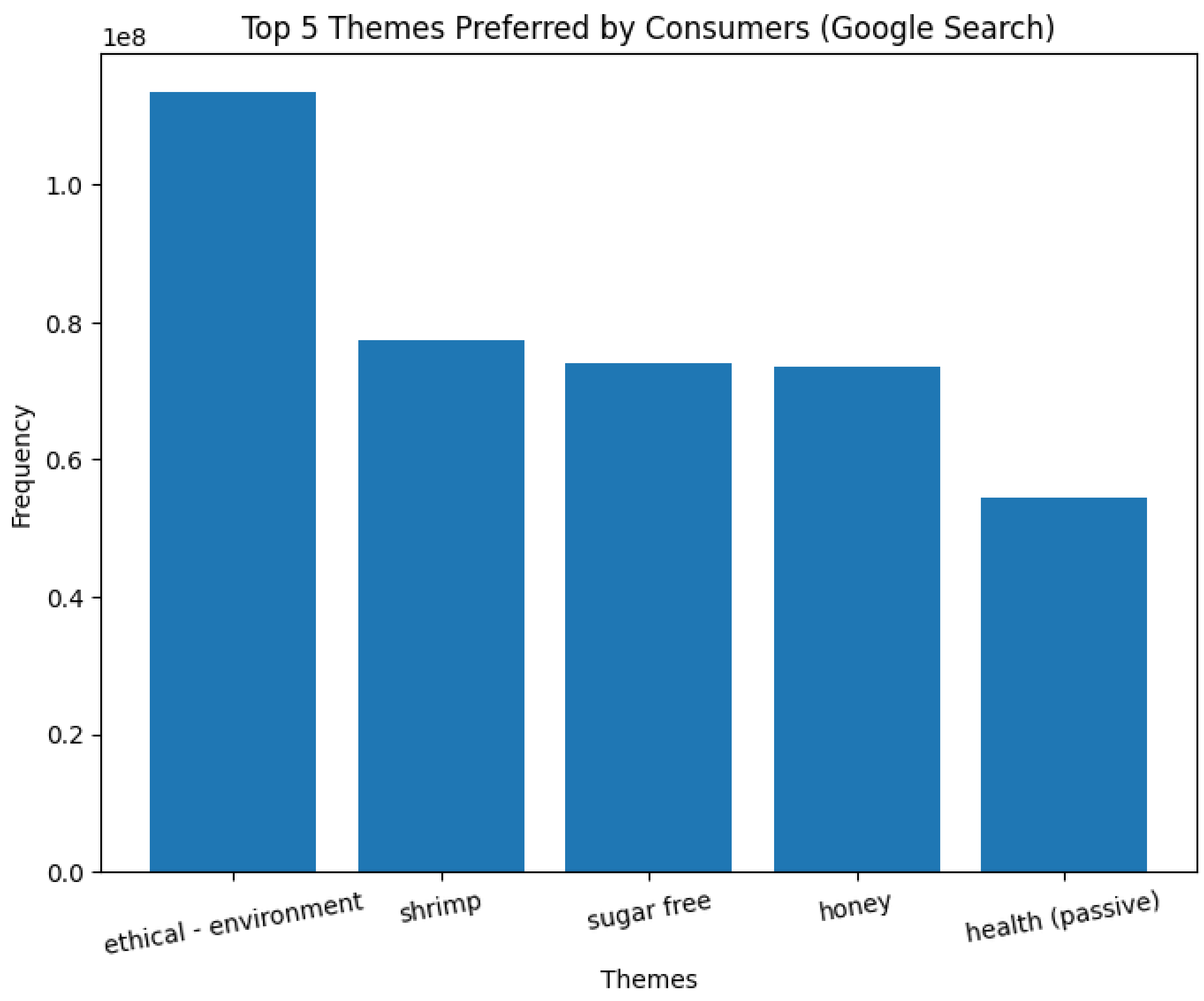
```python
# Task 3.3: Top 5 themes preferred by consumers as per google search data
#merging search and theme set will make the easier to filter out google in the merged dataset
#Allowing us to get the Claim Names for the themes and doing the task
df4 = pd.merge(theme,search,on='Theme_ID',how='inner')
google_dataa = df4[df4['platform'] == 'google']
top_5_themes_google = google_dataa[google_dataa['Theme_ID'] != 0].groupby('Claim Name')['searchVolume'].sum().nlargest(5)
print('The top 5 themes preferred by customer search data is:\n',top_5_themes_google)
```

```
The top 5 themes preferred by customer search data is:
 Claim Name
ethical - environment    113482508
shrimp                    77498586
sugar free                74104963
honey                     73600640
health (passive)          54400741
Name: searchVolume, dtype: int64
```

In [ ]:

```python
# Create a single plot
fig, ax = plt.subplots(figsize=(8, 6))

# Plot the top 5 themes preferred by consumers as per google search data
ax.bar(top_5_themes_google.index, top_5_themes_google.values)
ax.set_xlabel('Themes')
ax.set_ylabel('Frequency')
ax.set_title('Top 5 Themes Preferred by Consumers (Google Search)')
ax.set_xticklabels(top_5_themes_google.index, rotation=10)

# Show the plot
plt.show()
```

```
<ipython-input-57-6d4741b0e12f>:9: UserWarning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels(top_5_themes_google.index, rotation=10)
```



Task 4

My reasoning for selecting monthly granularity is as follows:

Smoothing and Seasonality: Monthly granularity smoothes out short-term swings while capturing underlying patterns and seasonality in the data. Many corporate processes and consumer habits, such as monthly sales cycles, seasonal trends, and monthly budgeting or planning cycles, display monthly patterns.

Sufficient data: Monthly granularity gives enough data to evaluate and comprehend trends and patterns within a particular month. It finds a compromise between capturing the intricacies of shorter time periods (such as daily or weekly) and avoiding the unnecessary noise or data volume that comes with finer granularities.

Practicality and interpretability: Monthly data is reasonably simple to manipulate and interpret. It is well-aligned with typical company reporting and decision-making processes, making communication and analysis easier.Monthly data points are less detailed than daily or hourly data, allowing for more succinct and informative trend portrayal.

Adequate Sample Size: Because monthly data has a relatively big sample size, it allows for more reliable statistical analysis, forecasting, and modeling. With more data points than quarterly or yearly data, statistical metrics such as mean, variance, or correlation may be estimated more accurately.

Long-term Patterns: Because of the monthly granularity, long-term patterns and trends may be identified and analyzed. It aids in capturing long-term changes, growth rates, and alterations in consumer behavior or market dynamics.

Overall, monthly granularity provides a reasonable mix between capturing relevant patterns and trends and retaining a practical degree of information and interpretability. However, the granularity selection depends on the unique environment, type of the data, and the study or commercial aims. When choosing the right granularity for time series analysis, it is critical to examine the features of the data as well as the specific analysis needs.

In [ ]:
```
# Find the theme_id for GMO Free
theme_GMO_free = theme[theme['Claim Name'] == 'gmo free']["Theme_ID"].values
print(theme_GMO_free)
```

```
# Use the recommended time granularity, analyse the time series trend among the sales, social media,
# and google search data for the GMO free theme and show if there is any correlation
```

```
[81]
```

In [ ]:

```
def group_data_by_month(data, date_column, value_column):
    data[date_column] = pd.to_datetime(data[date_column])
    grouped_data = data.groupby(pd.Grouper(key=date_column, freq='M'))[value_column].sum()
    return grouped_data

# Filter sales data for the GMO free theme

gmo_free_sales = df2[df2['Theme_ID'].isin(theme_GMO_free)]

# Filter social media data for the GMO free theme
gmo_free_social_media = sm[sm['Theme_ID'].isin(theme_GMO_free)]

# Filter Google search data for the GMO free theme
gmo_free_google_search = search[search['Theme_ID'].isin(theme_GMO_free)]
```

In [ ]:

```
start_year = 2016
# Set the time granularity to monthly
time_granularity = 'M'
```

In [ ]:

```
#group sales data by month
gmo_free_sales['system_calendar_key_N'] = pd.to_datetime(gmo_free_sales['system_calendar_key_N'], format='%Y%m%d')
#make this date column the index in order to plot the graph
gmo_free_sales.set_index('system_calendar_key_N', inplace=True)
grouped_sales = gmo_free_sales[gmo_free_sales.index.year >= start_year].groupby(pd.Grouper(freq=time_granularity)).sum()['sales_units_value']
```

```
<ipython-input-61-6ad644b669a0>: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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  gmo_free_sales['system_calendar_key_N'] = pd.to_datetime(gmo_free_sales['system_calendar_key_N'], format='%Y%m%d')
<ipython-input-61-6ad644b669a0>:5: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_o
nly or select only columns which should be valid for the function.
  grouped_sales = gmo_free_sales[gmo_free_sales.index.year >= start_year].groupby(pd.Grouper(freq=time_granularity)).sum()['sales_units_value']
```

In [ ]:

```
# Group social media data by month
gmo_free_social_media.loc[:, 'published_date'] = pd.to_datetime(gmo_free_social_media['published_date'])
grouped_social_media = gmo_free_social_media[gmo_free_social_media['published_date'].dt.year >= start_year].groupby(pd.Grouper(key='published_date', freq=time_granularity)).sum()['total_post']
```

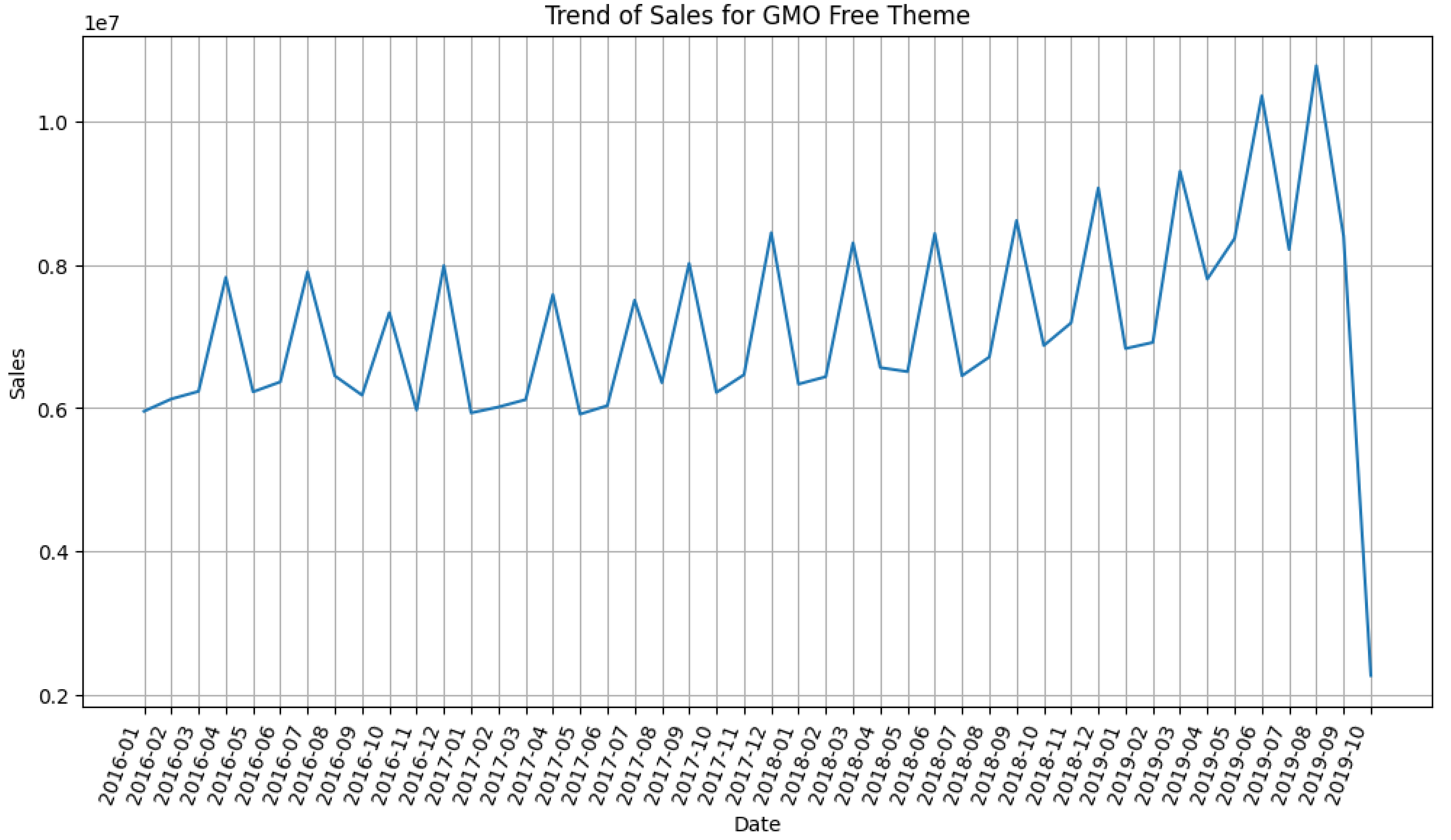
```
<ipython-input-62-beb4025080c2>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  gmo_free_social_media.loc[:, 'published_date'] = pd.to_datetime(gmo_free_social_media['published_date'])
<ipython-input-62-beb4025080c2>:2: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inplace instead of always setting a new array. To retain the old behavi
or, use either `df[df.columns[i]] = newvals` or, if columns are non-unique, `df.isetitem(i, newvals)`
  gmo_free_social_media.loc[:, 'published_date'] = pd.to_datetime(gmo_free_social_media['published_date'])
```

In [ ]:

```
# Group Google search data by month
gmo_free_google_search['date'] = pd.to_datetime(gmo_free_google_search['date'])
grouped_google_search = gmo_free_google_search[gmo_free_google_search['date'].dt.year >= start_year].groupby(pd.Grouper(key='date', freq=time_granularity)).sum()['searchVolume']
```

```
<ipython-input-63-0f2837671583>:2: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to ensure consistent parsing.
  gmo_free_google_search['date'] = pd.to_datetime(gmo_free_google_search['date'])
<ipython-input-63-0f2837671583>: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-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  gmo_free_google_search['date'] = pd.to_datetime(gmo_free_google_search['date'])
<ipython-input-63-0f2837671583>:3: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
  grouped_google_search = gmo_free_google_search[gmo_free_google_search['date'].dt.year >= start_year].groupby(pd.Grouper(key='date', freq=time_granularity)).sum()['searchVolume']
```

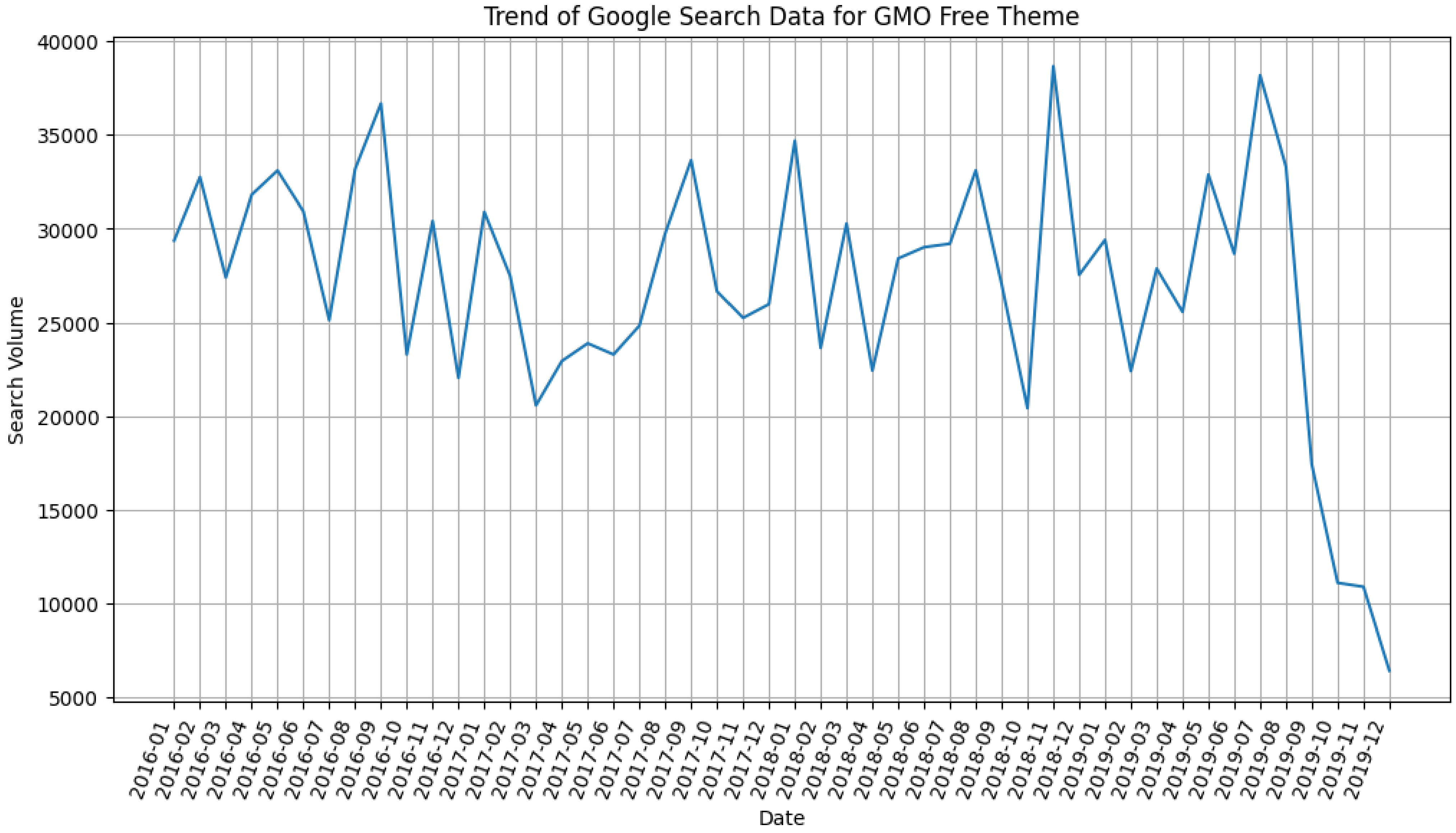
In [ ]:
```
plt.figure(figsize=(12, 6))
plt.plot(grouped_sales.index.strftime('%Y-%m'), grouped_sales.values)
plt.title('Trend of Sales for GMO Free Theme')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.grid(True)
plt.xticks(rotation=70, ha='right')
plt.show()
```



In this graph we can see that the general trend is where the numbers is around 0.6 10^7 to 0.8 10^7 from 2016 january to 2019 july.From August 2019 onwards, the trend is a downward spike from there to 2019 october January.

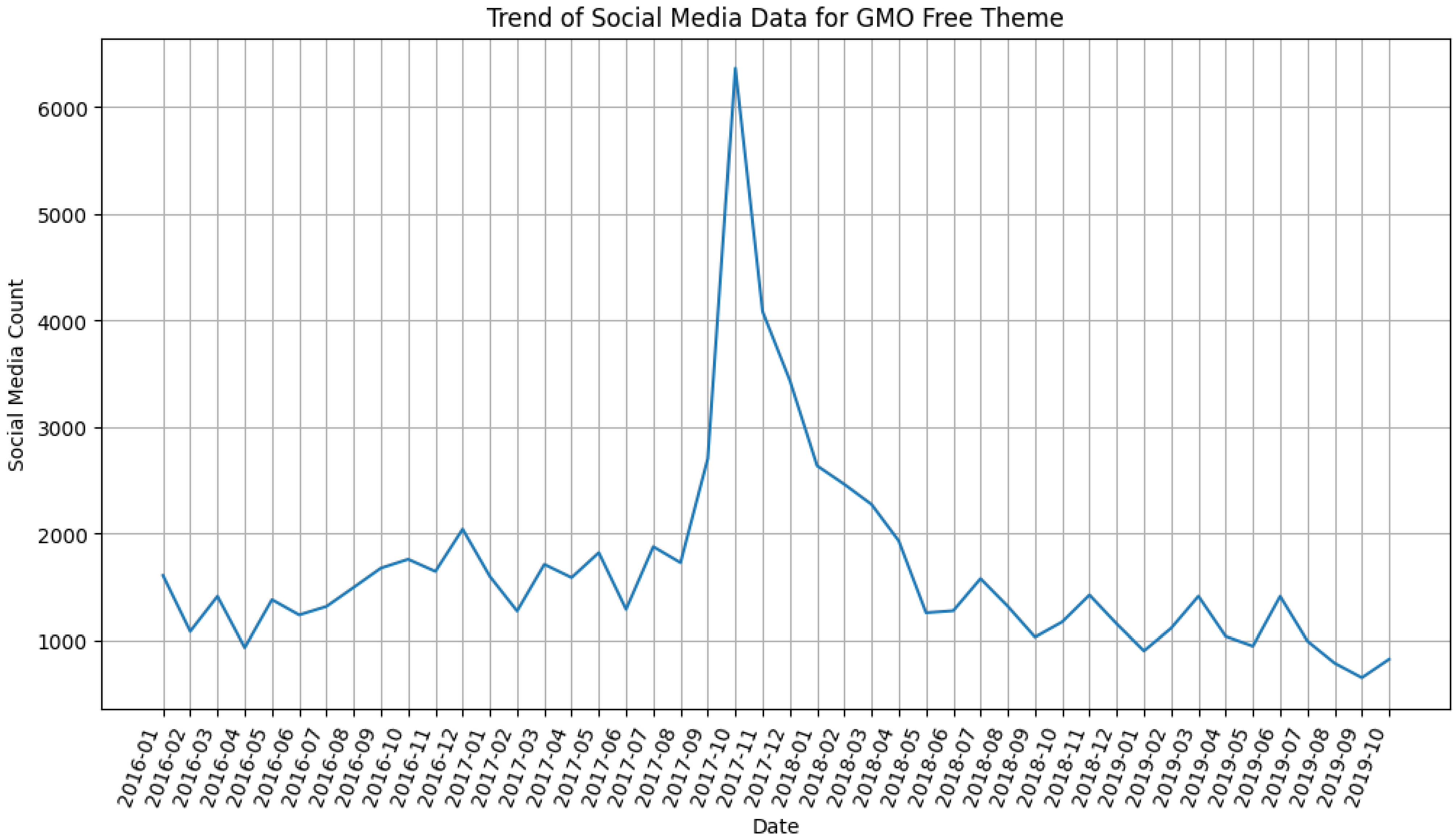
In [ ]:
```
# Plot the trend of Google search data
plt.figure(figsize=(12, 6))
plt.plot(grouped_google_search.index.strftime('%Y-%m'), grouped_google_search.values)
plt.title('Trend of Google Search Data for GMO Free Theme')
```

```
plt.xlabel('Date')
plt.ylabel('Search Volume')
plt.grid(True)
plt.xticks(rotation=70, ha='right')
plt.show()
```



For this graph, we can infer that the general trend is around 30000 with a few exceptions of 35000 and above in late 2016 and 2018 and a few lows of around 20000 and as with the first graph there is a sharp downturn from late 2019.

In [ ]:
```
# Plot the trend of social media data
plt.figure(figsize=(12, 6))
plt.plot(grouped_social_media.index.strftime('%Y-%m'), grouped_social_media.values)
plt.title('Trend of Social Media Data for GMO Free Theme')
plt.xlabel('Date')
plt.ylabel('Social Media Count')
plt.grid(True)
plt.xticks(rotation=70,ha='right')
plt.show()
```



For the third graph, the main trend is quite low at around 1000 to 2000 and in between 2017-7 to 2018-01 there is a sharp incline in post count in late 2017 to early 2018

One thing which is noticeable is the fact that there is a massive decline in trends towards the end of 2019 for the sales and search graphs causing the numbers to be low thus indicating an obvious correlation between the 2 graphs.

Task 5

In [ ]:
```
# Task 5C: Show the total sales unit of various vendors
dd = pd.merge(sales,manufactur,on='PRODUCT_ID',how='inner')
total_sales_unit_vendors = dd.groupby('Vendor')['sales_units_value'].sum()
print(total_sales_unit_vendors)
```

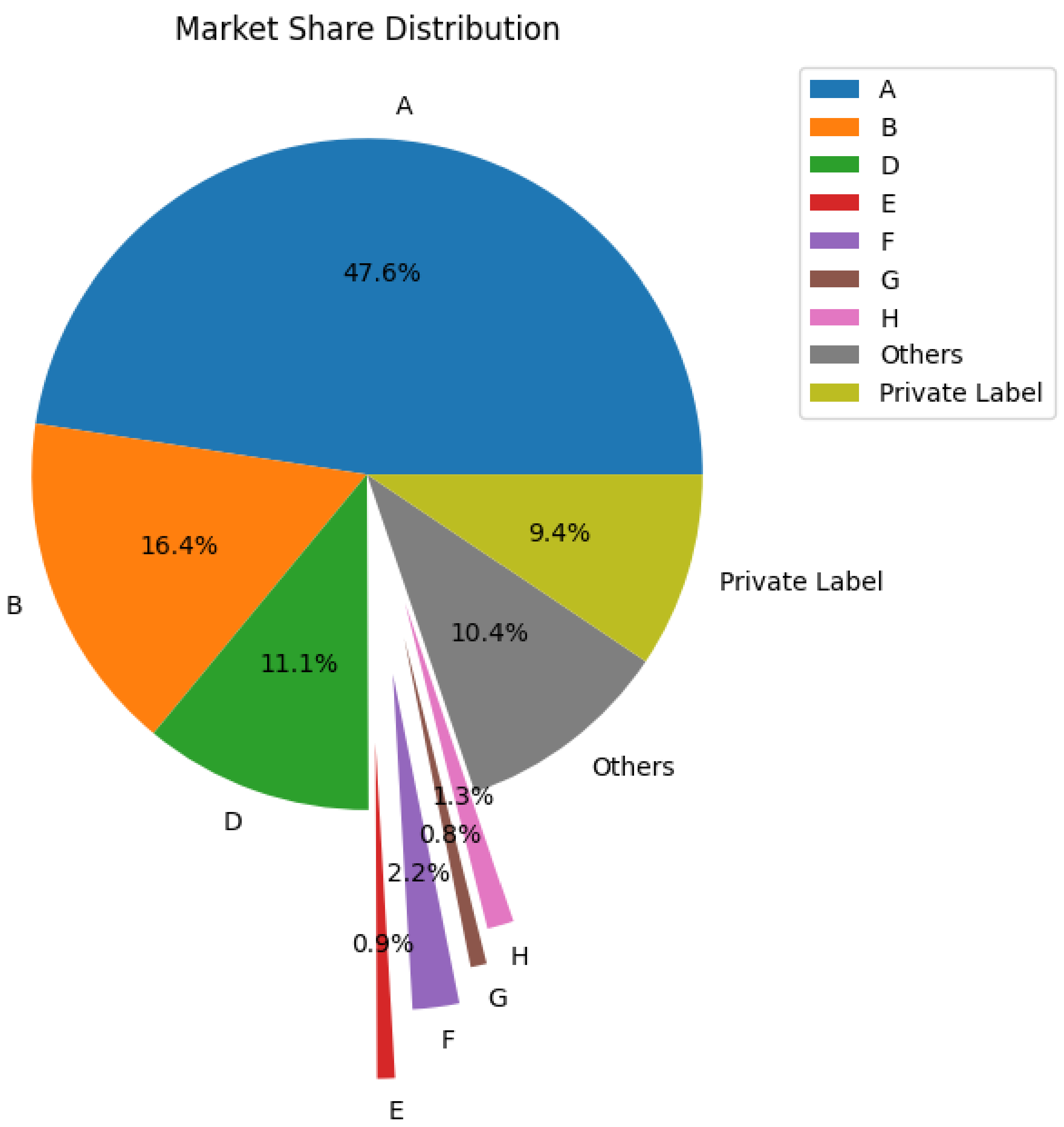
```
Vendor
A                11110417674
B                 3841860718
D                 2582299912
E                  200362303
F                  525090112
G                  181286878
H                  294479885
Others            2421832452
Private Label     2197710799
Name: sales_units_value, dtype: int64
```

In [ ]:
```
# Task 5D: Who are the top 3 manufacturers in overall sales unit?
top_3_manufacturers_sales_unit = dd.groupby('Vendor')['sales_units_value'].sum().nlargest(3)
print(top_3_manufacturers_sales_unit)
```

```
Vendor
A    11110417674
B     3841860718
D     2582299912
Name: sales_units_value, dtype: int64
```

In [ ]:

```
# Task 5C: Plot the percentage distribution of market shares in terms of sales unit value among all vendors
market_share_sales_unit = total_sales_unit_vendors / total_sales_unit_vendors.sum() * 100

# Set explode values for all sections to prevent smaller percentages values from overlapping with each other
explode = [0, 0, 0, 0.8, 0.6, 0.5, 0.4, 0, 0]

# Increase the overall size of the pie chart
plt.figure(figsize=(10, 6))

# Create the pie chart
#_, _, autotexts = plt.pie(...) is used to unpack the third returned value from the plt.pie() function call.
plt.pie(
    market_share_sales_unit,
    labels=market_share_sales_unit.index,
    autopct='%1.1f%%',
    explode=explode
)


# Set the legend position
plt.legend(bbox_to_anchor=(1, 1), loc='upper left')

# Add a title to the pie chart
plt.title("Market Share Distribution")

# Display the pie chart
plt.show()
```



Task 6

In [ ]:
```
# Function to filter competitors based on theme and manufacturer
df5 = pd.merge(manufactur,sales,on='PRODUCT_ID')
df6 = pd.merge(df5,themeprod,on='PRODUCT_ID')
def find_competitors(target_theme, manufacturer):
    # Convert the "Claim Name" column to lowercase for case-insensitive comparison
    lower_theme_df = theme.copy()
    lower_theme_df["Claim Name"] = lower_theme_df["Claim Name"].apply(lambda x: x.lower())
    # Get the relevant Theme_ID for the given theme
    theme_id = lower_theme_df.loc[lower_theme_df["Claim Name"] == target_theme.lower(), "Theme_ID"].values[0]
    # Filter themeprod_df to get PRODUCT_IDs for the given theme
    theme_products = themeprod.loc[themeprod["Theme_ID"] == theme_id, "PRODUCT_ID"]
    # Filter manufactur_df to get potential competitors for the given theme and manufacturer
    competitors = df6.loc[(df6["PRODUCT_ID"].isin(theme_products)) & (df6["Vendor"] != 'A') & (df6["sales_units_value"] > 0), "Vendor"].unique()
    return competitors

# Find potential competitors for the "GMO free" theme against manufacturer "A"
gmo_free_competitors = find_competitors("gmo free", "A")
# Print the potential competitors for the "GMO free" theme against manufacturer "A"
print("Potential competitors for GMO free against manufacturer A:")
for competitor in gmo_free_competitors:
    print(competitor)
```

```
Potential competitors for GMO free against manufacturer A:
Others
F
H
B
Private Label
```

Part 2

For finding themes with high business oppotunities we need to merge sales and themeproductid dataset and filter according to the theme_id and sales_units_value columns

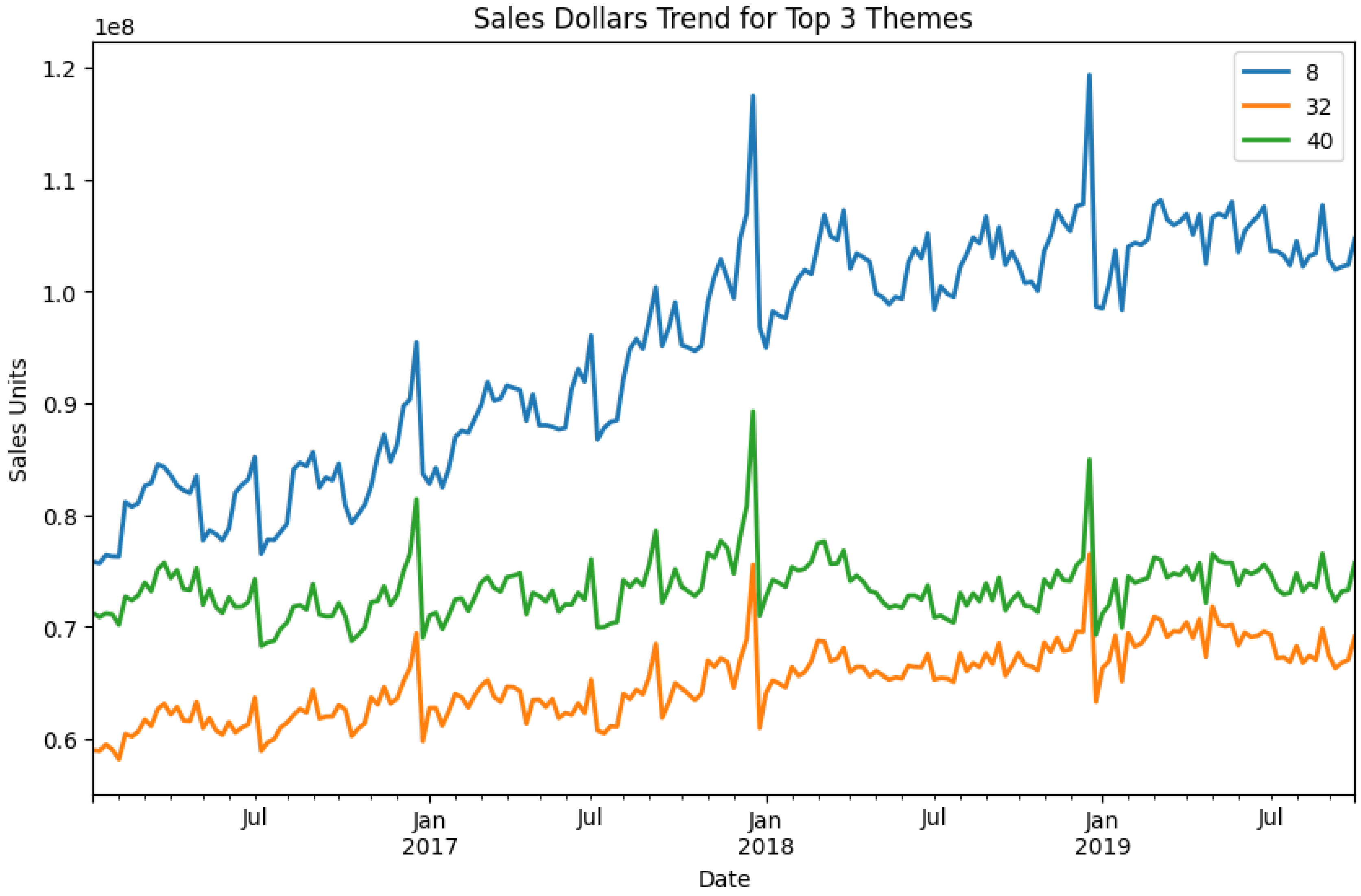
In [ ]:
```python
# Mkae the system_calendar_key_N column in the df2 dataset the index to produce a decent graph
df2['system_calendar_key_N'] = pd.to_datetime(df2['system_calendar_key_N'], format='%Y%m%d')
df2.set_index('system_calendar_key_N', inplace=True)

# Group the merged data by theme and calculate total sales units
lol = df2[df2['Theme_ID'] != 0]
theme_sale = lol.groupby('Theme_ID')['sales_dollars_value'].sum()
theme_sales = lol.groupby('Claim Name')['sales_dollars_value'].sum()
# Sort the themes in descending order of sales units
sorted_themes = theme_sales.sort_values(ascending=False)
sorted_theme = theme_sale.sort_values(ascending=False)
# Select the top three themes
top_3_themes = sorted_themes.nlargest(3)
top_3_theme = sorted_theme.nlargest(3)
# Print the top three themes and their corresponding sales units
print('Top 3 Themes with High Business Opportunity:')
print('Based on revenue generated')
print(top_3_themes)
print(top_3_theme)
```

```
Top 3 Themes with High Business Opportunity:
Based on revenue generated
Claim Name
low carb                       1.862566e+10
no additives/preservatives     1.438821e+10
stroganoff                     1.275780e+10
Name: sales_dollars_value, dtype: float64
Theme_ID
8     1.862566e+10
40    1.438821e+10
32    1.275780e+10
Name: sales_dollars_value, dtype: float64
```

Based on the themes shown we shall find their trends to see how good they are for business growth for Manufacturer A

In [ ]:
```python
# Filter the merged data for the top three themes

filtered_data = df2[df2['Theme_ID'].isin(top_3_theme.index)]

# Group the filtered data by theme and date, calculate monthly sales units
grouped_data = filtered_data.groupby(['Theme_ID', 'system_calendar_key_N'])['sales_dollars_value'].sum().reset_index()

# Pivot the data to have themes as columns and dates as rows
pivot_data = grouped_data.pivot_table(index='system_calendar_key_N', columns='Theme_ID', values='sales_dollars_value', aggfunc='sum')

# Plot the trendline for each theme
pivot_data.plot(figsize=(10, 6), linewidth=2)

# Set plot title and axis labels
plt.title('Sales Dollars Trend for Top 3 Themes')
plt.xlabel('Date')
plt.ylabel('Sales Units')
```

```python
# Display the legend
plt.legend()

# Show the plot
plt.show()
```



As seen by the graph, the sales amount for all 3 themes is on a steady trend of increasing which is good for business growth as Manufacturer A can earn alot of money by investing into the aforementioned themes for sales. However getting a share of this huge booming market may prove to be quite competitve as there may be more manufacturers in this space

In [ ]:
```python
# Make the system_calendar_key_N column in the df2 dataset the index to produce a decent graph

# Group the merged data by theme and calculate total sales units
lol = df2[df2['Theme_ID'] != 0]
theme_sale = lol.groupby('Theme_ID')['sales_units_value'].sum()
theme_sales = lol.groupby('Claim Name')['sales_units_value'].sum()
# Sort the themes in descending order of sales units
sorted_themes = theme_sales.sort_values(ascending=False)
sorted_theme = theme_sale.sort_values(ascending=False)
# Select the top three themes
top_3_themes = sorted_themes.nlargest(3)
top_3_theme = sorted_theme.nlargest(3)
# Print the top three themes and their corresponding sales units
print('Top 3 Themes with High Business Opportunity:')
print('Based on units sold')
print(top_3_themes)
print(top_3_theme)
```

```
Top 3 Themes with High Business Opportunity:
Based on units sold
Claim Name
low carb                        2627421936
no additives/preservatives      1559993572
stroganoff                      1391218733
Name: sales_units_value, dtype: int64
Theme_ID
8     2627421936
40    1559993572
32    1391218733
Name: sales_units_value, dtype: int64
```

In [ ]:

```
# Group the merged data by theme and calculate total sales units
lol = df3[df3['Theme_ID'] != 0]
#theme_sale = lol.groupby('Theme_ID')['total_post'].sum()
theme_sales = lol.groupby('Claim Name')['total_post'].sum()
# Sort the themes in descending order of sales units
#sorted_themes = theme_sales.sort_values(ascending=False)
sorted_theme = theme_sale.sort_values(ascending=False)
# Select the top three themes
top_3_themes = sorted_themes.nlargest(3)
top_3_theme = sorted_theme.nlargest(3)
# Print the top three themes and their corresponding sales units
print('Top 3 Themes with High Business Opportunity:')
print('Based on Social Media posts')
print(top_3_themes)
#print(top_3_theme)
```

```
Top 3 Themes with High Business Opportunity:
Based on Social Media posts
Claim Name
low carb                        2627421936
no additives/preservatives      1559993572
stroganoff                      1391218733
Name: sales_units_value, dtype: int64
```

We use social media posts as a factor of consideration as this may attract more consumers especially the social media savvy people to buy the products.There is also a obvious correlation with social media posts and dollars generated as the exact same themes appear.

In [ ]:

```
# Task 2: Calculate market share for each theme
total_sales = df2['sales_units_value'].sum()
theme_market_share = theme_sale / total_sales

# Task 3: Identify themes with high growth potential
theme_growth = df2.groupby('Claim Name')['sales_dollars_value'].mean().pct_change()

# Identify the top 3 themes based on sales volume, market share, and growth potential
top_3_themes_market_share = theme_market_share.nlargest(3)
top_3_themes_growth = theme_growth.nlargest(3)

# Print the recommended themes
print("Top 3 Themes with High Business Opportunity:")
print("Based on Market Share:")
print(top_3_themes_market_share)
print("Based on Growth Potential:")
print(top_3_themes_growth)
```

```
Top 3 Themes with High Business Opportunity:
Based on Market Share:
Theme_ID
8     0.093683
40    0.055623
32    0.049605
Name: sales_units_value, dtype: float64
Based on Growth Potential:
Claim Name
vegetarian                    122.202870
no additives/preservatives     21.277672
soy foods                      20.926419
Name: sales_dollars_value, dtype: float64
```

We also look at market share to see how distributed the 3 themes are. Growth Potential is based on the .pct_change() function which is a pandas method that calculates the percentage change between consecutive elements in a series or column. When applied to a series of values, it computes the percentage change from the previous element to the current element. This will help us see which themes have the biggest percentage changes based on dollar value.

All in all , I feel that the 3 best themes for business growth is low carb,no additives/preservatives and stroganoff due to their overwhelming presence in the business,consumer and media markets