Installation and importing Libraries

```
!pip install --upgrade google-api-python-client textblob vaderSentiment
```

```
Requirement already satisfied: google-api-python-client in /usr/local/lib/python3.11/dist-packages (2.177.0)
Collecting google-api-python-client
  Downloading google_api_python_client-2.178.0-py3-none-any.whl.metadata (7.0 kB)
Requirement already satisfied: textblob in /usr/local/lib/python3.11/dist-packages (0.19.0)
Collecting vaderSentiment
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl.metadata (572 bytes)
Requirement already satisfied: httplib2<1.0.0,>=0.19.0 in /usr/local/lib/python3.11/dist-packages (from google-api-python-client) (0.22.
Requirement already satisfied: google-auth!=2.24.0,!=2.25.0,<3.0.0,>=1.32.0 in /usr/local/lib/python3.11/dist-packages (from google-api-
Requirement already satisfied: google-auth-httplib2<1.0.0,>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from google-api-python-cli
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0,>=1.31.5 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: uritemplate<5,>=3.0.1 in /usr/local/lib/python3.11/dist-packages (from google-api-python-client) (4.2.0)
Requirement already satisfied: nltk>=3.9 in /usr/local/lib/python3.11/dist-packages (from textblob) (3.9.1)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from vaderSentiment) (2.32.3)
Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.56.2 in /usr/local/lib/python3.11/dist-packages (from google-api-core!
Requirement already satisfied: protobuf!=3.20.0,!=3.20.1,!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<7.0.0,>=3.19.5 in /usr/l
Requirement already satisfied: proto-plus<2.0.0,>=1.22.3 in /usr/local/lib/python3.11/dist-packages (from google-api-core!=2.0.*,!=2.1.*
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from google-auth!=2.24.0,!=2.25.0,<3.0
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from google-auth!=2.24.0,!=2.25.0,<3.0.
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.11/dist-packages (from google-auth!=2.24.0,!=2.25.0,<3.0.0,>=1.32
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in /usr/local/lib/python3.11/dist-packages (from http
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk>=3.9->textblob) (4.67.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->vaderSentiment) (3.4.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->vaderSentiment) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->vaderSentiment) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->vaderSentiment) (2025.8.3)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.11/dist-packages (from pyasn1-modules>=0.2.1->google-auth!
Downloading google_api_python_client-2.178.0-py3-none-any.whl (13.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 100.3 MB/s eta 0:00:00
Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.0/126.0 kB 10.4 MB/s eta 0:00:00
Installing collected packages: vaderSentiment, google-api-python-client
  Attempting uninstall: google-api-python-client
    Found existing installation: google-api-python-client 2.177.0
    Uninstalling google-api-python-client-2.177.0:
      Successfully uninstalled google-api-python-client-2.177.0
Successfully installed google-api-python-client-2.178.0 vaderSentiment-3.3.2
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
!pip install qdrant-client
```

```
Collecting qdrant-client
  Downloading qdrant_client-1.15.1-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: grpcio>=1.41.0 in /usr/local/lib/python3.11/dist-packages (from qdrant-client) (1.74.0)
Requirement already satisfied: httpx>=0.20.0 in /usr/local/lib/python3.11/dist-packages (from httpx[http2]>=0.20.0->qdrant-client) (0.28
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.11/dist-packages (from qdrant-client) (2.0.2)
Collecting portalocker<4.0,>=2.7.0 (from qdrant-client)
  Downloading portalocker-3.2.0-py3-none-any.whl.metadata (8.7 kB)
Requirement already satisfied: protobuf>=3.20.0 in /usr/local/lib/python3.11/dist-packages (from qdrant-client) (5.29.5)
Requirement already satisfied: pydantic!=2.0.*,!=2.1.*,!=2.2.0,>=1.10.8 in /usr/local/lib/python3.11/dist-packages (from qdrant-client)
Requirement already satisfied: urllib3<3,>=1.26.14 in /usr/local/lib/python3.11/dist-packages (from qdrant-client) (2.5.0)
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-packages (from httpx>=0.20.0->httpx[http2]>=0.20.0->qdrant-client
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.20.0->httpx[http2]>=0.20.0->qdrant-clie
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.20.0->httpx[http2]>=0.20.0->qdrar
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from httpx>=0.20.0->httpx[http2]>=0.20.0->qdrant-client)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.20.0->httpx[http2]>=0.
Requirement already satisfied: h2<5,>=3 in /usr/local/lib/python3.11/dist-packages (from httpx[http2]>=0.20.0->qdrant-client) (4.2.0)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=2.0.*,!=2.1.*,!=2.2.0,>
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=2.0.*,!=2.1.*,!=2.2.0,>=
Requirement already satisfied: typing-extensions>=4.12.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=2.0.*,!=2.1.*,!=2.2.
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=2.0.*,!=2.1.*,!=2.2.0
Requirement already satisfied: hyperframe<7,>=6.1 in /usr/local/lib/python3.11/dist-packages (from h2<5,>=3->httpx[http2]>=0.20.0->qdrar
```

```
Requirement already satisfied: hpack<5,>=4.1 in /usr/local/lib/python3.11/dist-packages (from h2<5,>=3->httpx[http2]>=0.20.0->qdrant-cli
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio->httpx>=0.20.0->httpx[http2]>=0.20.0-
Downloading qdrant_client-1.15.1-py3-none-any.whl (337 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 337.3/337.3 kB 19.2 MB/s eta 0:00:00
Downloading portalocker-3.2.0-py3-none-any.whl (22 kB)
Installing collected packages: portalocker, qdrant-client
Successfully installed portalocker-3.2.0 qdrant-client-1.15.1
```

```
!pip install keybert sentence-transformers
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.4 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 11.1 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.5 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.9 MB/s eta 0:00:00
Downloading nvidia_nccl_cu12-2.21.5-py3-none-manylinux2014_x86_64.whl (188.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 188.7/188.7 MB 6.4 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 87.8 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu1
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-nccl-cu12
    Found existing installation: nvidia-nccl-cu12 2.23.4
    Uninstalling nvidia-nccl-cu12-2.23.4:
      Successfully uninstalled nvidia-nccl-cu12-2.23.4
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed keybert-0.9.0 nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidi
```

```
import pandas as pd
```

Setting up Youtube API

```
from googleapiclient.discovery import build

# 🔑 Replace with your actual API key
API_KEY = ""(paste your api key here)
```

```
API_KEY = "(paste your api key here)"
youtube = build('youtube', 'v3', developerKey=API_KEY)
```

Extracting relavent Youtube videos from the API

```
def search_youtube(query, total_results=100):
    videos = []
    next_page_token = None

    while len(videos) < total_results:
        remaining = total_results - len(videos)
        max_results = min(50, remaining)

        request = youtube.search().list(
            q=query,
            part='snippet',
            type='video',
            order='viewCount',
            maxResults=max_results,
            pageToken=next_page_token
        )
        response = request.execute()

        for item in response.get('items', []):
            video_id = item.get('id', {}).get('videoId')
            if not video_id:
                continue

            videos.append({
                'video_id': video_id,
                'title': item['snippet'].get('title', ''),
                'description': item['snippet'].get('description', ''),
                'channel': item['snippet'].get('channelTitle', ''),
                'published_at': item['snippet'].get('publishedAt', '')
            })

        next_page_token = response.get('nextPageToken')
        if not next_page_token:
            break  # No more pages

    return pd.DataFrame(videos)


df = search_youtube("atomberg fan", total_results=100)
print(df.shape)
df.head()
```

```
(100, 5)
```

	video_id	title	description	channel	published_at
0	4zy-Z_6j6KE	Atomberg TVC I Power Department comes knocking	ATOM and BERG are back. This time with an 'Oop...	Atomberg Technologies	2023-01-02T13:55:09Z
1	prF2IROgBzg	What a Fan! | Harsha Bhogle x Atomberg	The commentary maven has met many a fan in his...	Atomberg Technologies	2024-02-01T10:23:08Z
2	cnkeSDW2Xps	Atomberg | What a Fan (Bengali) | Harsha x Ato...	The commentary maven has met many a fan in his...	Atomberg Technologies	2024-02-21T09:07:41Z

Obtaining video stats

```
def get_video_stats(video_ids):
    stats = []

    for i in range(0, len(video_ids), 50):  # API allows max 50 at once
        batch_ids = video_ids[i:i+50]
        request = youtube.videos().list(
            part="statistics",
            id=",".join(batch_ids)
```

```
            )
            response = request.execute()

            for item in response.get('items', []):
                video_id = item['id']
                stats_dict = item.get('statistics', {})
                stats.append({
                    'video_id': video_id,
                    'view_count': int(stats_dict.get('viewCount', 0)),
                    'like_count': int(stats_dict.get('likeCount', 0)),
                    'comment_count': int(stats_dict.get('commentCount', 0))
                })

        return pd.DataFrame(stats)



    video_df = search_youtube("Atomberg fan review", total_results=100)
    video_ids = video_df['video_id'].tolist()

    stats_df = get_video_stats(video_ids)

    # Merge both dataframes on 'video_id'
    final_df = pd.merge(video_df, stats_df, on='video_id')

    final_df.head()
```

	video_id	title	description	channel	published_at	view_count	like_count	comment_count
0	TES_KUo50Bo	BLDC fan unit consumption for 24 hours..		Engineering Facts	2024-09-27T13:22:56Z	3560323	134559	499
1	wfeb17_lrEQ	Don't BUY Atomberg BLDC Fan #shorts #ytshorts		Dip Electronics Secret	2023-07-20T10:04:03Z	2955550	65426	1442
2	u_kxhHjHln0	Atomberg Renesa 1400mm	Atomberg Renesa 1400mm mrp 3850 on atomberg we...	AMARDEEP Toor	2022-05-04T17:45:33Z	3172671	30668	379
3	EN7aVw7X9Kc	This is the Smartest Ceiling	Instagram -	Halka Tech	2023-10-	2421742	108660	241

Code for extracting youtube comments of the videos

```
def get_video_comments(video_id, max_comments=50):
    comments = []
    next_page_token = None

    while len(comments) < max_comments:
        try:
            request = youtube.commentThreads().list(
                part="snippet",
                videoId=video_id,
                maxResults=min(100, max_comments - len(comments)),  # API allows max 100
                textFormat="plainText",
                pageToken=next_page_token
            )
            response = request.execute()

            for item in response.get('items', []):
                comment = item['snippet']['topLevelComment']['snippet']
                comments.append({
                    'video_id': video_id,
                    'author': comment.get('authorDisplayName'),
                    'comment': comment.get('textDisplay'),
                    'like_count': comment.get('likeCount'),
                    'published_at': comment.get('publishedAt')
                })

            next_page_token = response.get('nextPageToken')
            if not next_page_token:
                break
        except Exception as e:
```

```
            print(f"Error fetching comments for video {video_id}: {e}")
            break

    return comments



top_videos = final_df.sort_values(by="view_count", ascending=False).head(100)
all_comments = []

for vid in top_videos['video_id']:
    all_comments.extend(get_video_comments(vid, max_comments=20))

# Convert to DataFrame
comments_df = pd.DataFrame(all_comments)
comments_df
```

```
WARNING:googleapiclient.http:Encountered 403 Forbidden with reason "commentsDisabled"
Error fetching comments for video YFkEL1qENJo: <HttpError 403 when requesting https://youtube.googleapis.com/youtube/v3/commentThreads?p
```

	video_id	author	comment	like_count	published_at
0	TES_KUo50Bo	@kesavankesavan4568	Bro how can make to bldc circuit pcb explain...	0	2025-07-25T16:50:24Z
1	TES_KUo50Bo	@Ranganathan-g9v	BLDC fan lifespan is only 3 years then you hav...	1	2025-07-24T04:59:10Z
2	TES_KUo50Bo	@vishnugh1372	Hello thalaivarey\n\nSaffola honey \nDabur hon...	0	2025-07-18T14:31:46Z
3	TES_KUo50Bo	@prasanth5646	Crompton old model 80w fan maadhiri yendha fan...	0	2025-07-08T16:37:42Z
4	TES_KUo50Bo	@liger9336	Bro oru nalla rechargeable fan sollunga bro	0	2025-07-03T03:32:17Z
...	...	...	...	...	...
1836	WlqMeJtOjEw	@Sanju_craftss	Sir sabse best colour konsa rahega black ya wh...	1	2024-05-26T05:23:03Z
1837	WlqMeJtOjEw	@AjaySingh-uj1lk	Renesa ke jagah isko lena thik rahega	1	2024-05-18T06:54:13Z
1838	WlqMeJtOjEw	@mukundtechtips	Can you review renesa zen ?	0	2024-05-16T09:09:53Z
1839	WlqMeJtOjEw	@mohitsaini1971	An urge to you pls dont buy atomberg fans just...	0	2024-05-02T06:39:45Z
1840	WlqMeJtOjEw	@avinashbhatt2317	price...? flipkart and amazon me	0	2024-05-01T12:58:15Z

1841 rows × 5 columns

Adding the brand Names to the database

```
import pandas as pd
import re

# Step 1: Define your brand list
brands = [
    "DREO", "Ahawill", "Orient", "Atomberg", "Crompton", "Havells", "LG", "Panasonic",
    "Dyson", "Ludomide", "Bajaj", "Usha", "GoveeLife", "Hunter", "Modern Forms",
    "iLiving", "Smafan", "GE", "Airbrick", "KASA", "Caledon", "Kichler", "Xiaomi", "Dayton"
]

# Step 2: Function to detect brand mentions in a video
def detect_brands(title, description):
    text = f"{title} {description}".lower()
    found = [brand for brand in brands if re.search(rf"\b{brand.lower()}\b", text)]
    return found

# Step 3: Apply detection to your DataFrame
final_df['mentioned_brands'] = final_df.apply(
    lambda row: detect_brands(row['title'], row['description']), axis=1
)



# Step 4: Explode the DataFrame (one row per brand mention per video)
exploded_df = final_df.explode('mentioned_brands').dropna(subset=['mentioned_brands'])

# Step 5: Compute SoV Metrics
sov_df = exploded_df.groupby('mentioned_brands').agg({
    'video_id': 'count',
```

```
    'view_count': 'sum',
    'like_count': 'sum',
    'comment_count': 'sum'
}).rename(columns={
    'video_id': 'video_mentions',
    'view_count': 'total_views',
    'like_count': 'total_likes',
    'comment_count': 'total_comments'
}).sort_values(by='video_mentions', ascending=False).reset_index()

exploded_df
```

	video_id	title	description	channel	published_at	view_count	like_count	comment_count	mentioned_
1	wfeb17_IrEQ	Don't BUY Atomberg BLDC Fan #shorts #ytshorts		Dip Electronics Secret	2023-07-20T10:04:03Z	2955550	65426	1442	At
2	u_kxhHjHln0	Atomberg Renesa 1400mm	Atomberg Renesa 1400mm mrp 3850 on atomberg we...	AMARDEEP Toor	2022-05-04T17:45:33Z	3172671	30668	379	At
3	FN7aVw7X9Kc	This is the Smartest Ceiling Fan | Atomberg Ar...	Instagram - https://www.instagram.com/hrithika...	Halka Tech	2023-10-14T08:10:16Z	2421742	108660	241	At
4	vIM7Jx036S4	Atomberg Studio Stainless Steel 150mm BLDC Exh...	My advice for you, buy only Atomberg BLDC exha...	Sumit Jaiswal - PRONIX	2021-12-02T13:43:34Z	2107207	12769	101	At
5	2Tx1s2KLM4U	Cheap & Best Smart BLDC celling	smart celling fan with remote with power savin...	My Smart Support	2021-04-25T11:27:24Z	1287439	14366	562	

```
exploded_df_atomberg = exploded_df[
    exploded_df['mentioned_brands'].str.lower() == 'atomberg'
]
exploded_df_atomberg.head()
```

	video_id	title	description	channel	published_at	view_count	like_count	comment_count	mentioned_br
1	wfeb17_IrEQ	Don't BUY Atomberg BLDC Fan #shorts #ytshorts		Dip Electronics Secret	2023-07-20T10:04:03Z	2955550	65426	1442	Ator
2	u_kxhHjHln0	Atomberg Renesa 1400mm	Atomberg Renesa 1400mm mrp 3850 on atomberg we...	AMARDEEP Toor	2022-05-04T17:45:33Z	3172671	30668	379	Ator
3	FN7aVw7X9Kc	This is the Smartest Ceiling	Instagram -	Halka Tech	2023-10-	2421742	108660	241	Ator

```
exploded_df['mentioned_brands'].value_counts()
```

mentioned_brands	count
Atomberg	75
Havells	9
Crompton	8
Orient	5
Usha	2
LG	1
Bajaj	1

dtype: int64

Calculating and prinitng SOV of the the obtained brands

Proposed SoV Metric for Atomberg

Base formula:

*SoV = (Atomberg_Mentions / Total_Brand_Mentions) * 100*

Enhanced (Recommended) Formula — Weighted Share of Positive Voice (SoPV):

*SoPV = ((Positive_Mentions_for_Atomberg * E_avg) / sum_over_brands(Positive_Mentions * E_avg)) * 100*

Where:

Atomberg_Mentions = count of times Atomberg is mentioned in posts/comments

Total_Brand_Mentions = mentions of Atomberg + competitors

Positive_Mentions = count of positive sentiment mentions (from VADER)

E_avg = average engagement (likes, replies, etc.) for those mentions

```
volume_sov = sov_df.copy()
volume_sov['volume_share'] = volume_sov['video_mentions'] / volume_sov['video_mentions'].sum()
```

```
volume_sov
```

	mentioned_brands	video_mentions	total_views	total_likes	total_comments	volume_share
0	Atomberg	75	25790961	368536	12480	0.742574
1	Havells	9	2075762	30294	1185	0.089109
2	Crompton	8	2314574	24247	1435	0.079208
3	Orient	5	643030	5248	458	0.049505
4	Usha	2	362151	5824	275	0.019802
5	Bajaj	1	61112	503	25	0.009901
6	LG	1	1287439	14366	562	0.009901

Engagement Scores

```
sov_df['engagement_score'] = sov_df['total_views'] + 5 * sov_df['total_likes'] + 10 * sov_df['total_comments']
engagement_sov = sov_df[['mentioned_brands', 'engagement_score']].copy()
engagement_sov['engagement_share'] = engagement_sov['engagement_score'] / engagement_sov['engagement_score'].sum()
```

Adding the comments to the Qdrant memory

```
from sentence_transformers import SentenceTransformer
from qdrant_client import QdrantClient
from qdrant_client.models import PointStruct, VectorParams, Distance

model = SentenceTransformer("all-MiniLM-L6-v2")
client = QdrantClient(":memory:")
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secre
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100% 349/349 [00:00<00:00, 13.2kB/s]
config_sentence_transformers.json: 100% 116/116 [00:00<00:00, 6.57kB/s]
README.md: 10.5k/? [00:00<00:00, 655kB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 3.58kB/s]
config.json: 100% 612/612 [00:00<00:00, 34.2kB/s]
model.safetensors: 100% 90.9M/90.9M [00:00<00:00, 114MB/s]
tokenizer_config.json: 100% 350/350 [00:00<00:00, 19.8kB/s]
vocab.txt: 232k/? [00:00<00:00, 6.28MB/s]
tokenizer.json: 466k/? [00:00<00:00, 10.7MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 6.76kB/s]
config.json: 100% 190/190 [00:00<00:00, 8.49kB/s]
```

```
import re

def clean(text):
    text = re.sub(r"https?://\S+", "", text)
    text = re.sub(r"[^\w\s]", "", text)
    text = re.sub(r"\s+", " ", text).strip()
    return text.lower()

comments_df["clean_comment"] = comments_df["comment"].apply(clean)
```

```
comments_df.head()
```

	video_id	author	comment	like_count	published_at	clean_comment
0	TES_KUo50Bo	@kesavankesavan4568	Bro how can make to bldc circuit pcb explain...	0	2025-07-25T16:50:24Z	bro how can make to bldc circuit pcb explain p...
1	TES_KUo50Bo	@Ranganathan-g9v	BLDC fan lifespan is only 3 years then you hav...	1	2025-07-24T04:59:10Z	bldc fan lifespan is only 3 years then you hav...
2	TES_KUo50Bo	@vishnugh1372	Hello thalaivarey\n\nSaffola honey \nDabur hon...	0	2025-07-18T14:31:46Z	hello thalaivarey saffola honey dabur honey an...

```
embeddings = model.encode(comments_df["clean_comment"].tolist())

client.recreate_collection(
    collection_name="comments_collection",
    vectors_config=VectorParams(size=384, distance=Distance.COSINE)
)

points = [
    PointStruct(
        id=i,
        vector=embeddings[i],
        payload={
```

```
            "comment": comments_df.iloc[i]["comment"],
            "video_id": comments_df.iloc[i]["video_id"],
            "author": comments_df.iloc[i]["author"]
        }
    )
    for i in range(len(comments_df))
]

client.upsert(collection_name="comments_collection", points=points)
```

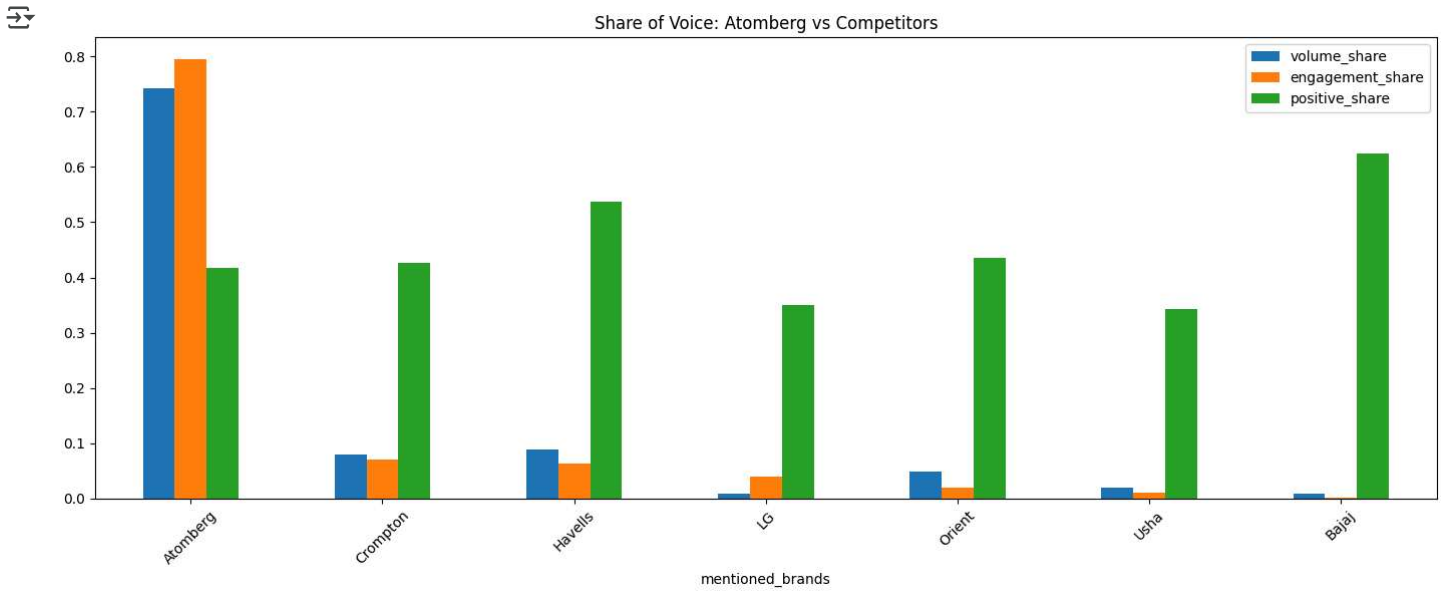
```
/tmp/ipython-input-226704966.py:3: DeprecationWarning: `recreate_collection` method is deprecated and will be removed in the future. Use
  client.recreate_collection(
UpdateResult(operation_id=0, status=<UpdateStatus.COMPLETED: 'completed'>)
```

Analysing the sentiment

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()

def get_vader_sentiment(text):
    score = analyzer.polarity_scores(text)['compound']
    if score >= 0.05:
        return 'positive'
    elif score <= -0.05:
        return 'negative'
    else:
        return 'neutral'

comments_df['sentiment'] = comments_df['clean_comment'].apply(get_vader_sentiment)



# Merge to assign brand to comment
comments_with_brand = pd.merge(comments_df, exploded_df[['video_id', 'mentioned_brands']], on='video_id', how='inner')

sentiment_counts = comments_with_brand.groupby(['mentioned_brands', 'sentiment']).size().unstack(fill_value=0)
sentiment_counts['total'] = sentiment_counts.sum(axis=1)
sentiment_counts['positive_share'] = sentiment_counts['positive'] / sentiment_counts['total']



final_sov = volume_sov[['mentioned_brands', 'volume_share']].merge(
    engagement_sov[['mentioned_brands', 'engagement_share']], on='mentioned_brands'
).merge(
    sentiment_counts[['positive_share']], on='mentioned_brands'
)

final_sov = final_sov.sort_values(by='engagement_share', ascending=False)
```

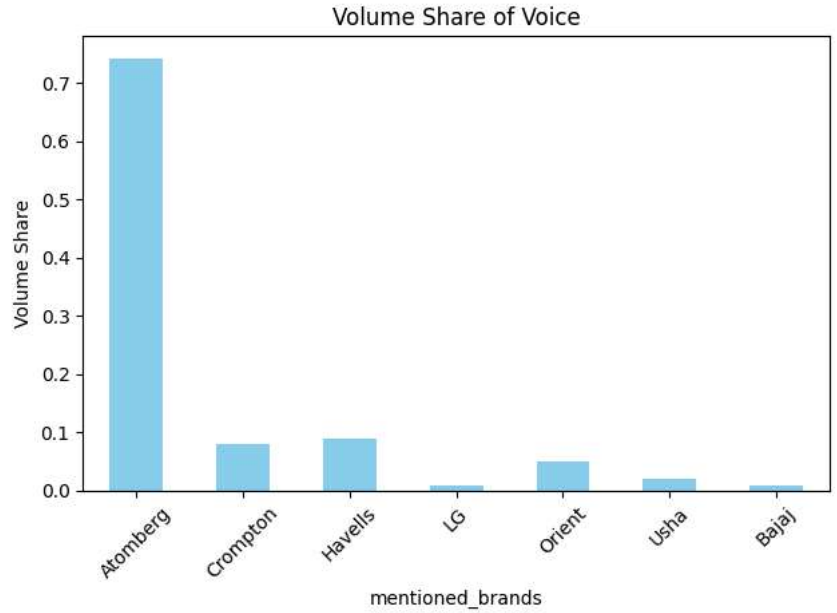
Plotting SOV on different metrics

```
import matplotlib.pyplot as plt



final_sov.plot(x='mentioned_brands', y=['volume_share', 'engagement_share', 'positive_share'],
               kind='bar', figsize=(14,6), title='Share of Voice: Atomberg vs Competitors')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

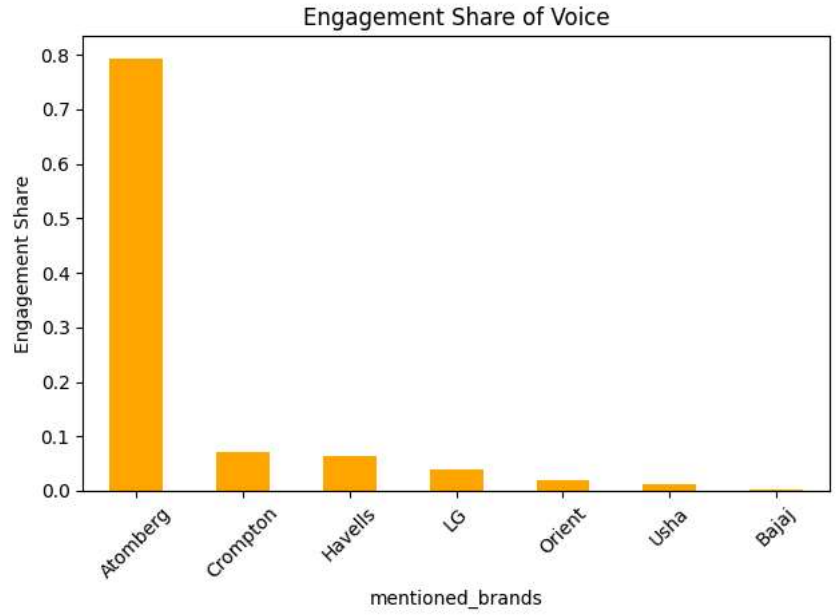



```
import matplotlib.pyplot as plt

# Plot 1: Volume Share
plt.figure(figsize=(12, 5))
final_sov.plot(
    x='mentioned_brands', y='volume_share', kind='bar',
    title='Volume Share of Voice', legend=False, color='skyblue'
)
plt.ylabel('Volume Share')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Plot 2: Engagement Share
plt.figure(figsize=(12, 5))
final_sov.plot(
    x='mentioned_brands', y='engagement_share', kind='bar',
    title='Engagement Share of Voice', legend=False, color='orange'
)
plt.ylabel('Engagement Share')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Plot 3: Positive Share of Voice
plt.figure(figsize=(12, 5))
final_sov.plot(
    x='mentioned_brands', y='positive_share', kind='bar',
    title='Positive Share of Voice', legend=False, color='green'
)
plt.ylabel('Positive Share')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x500 with 0 Axes>
```




```
<Figure size 1200x500 with 0 Axes>
```




```
<Figure size 1200x500 with 0 Axes>
```

Extracting comments from Qdrant and then obtaining KeyWords

```
# Pull all payloads (comments) from Qdrant
retrieved = client.scroll(
    collection_name="comments_collection",
    with_payload=True,
    limit=1000  # Adjust as needed
)

comments = [point.payload["comment"] for point in retrieved[0]]
```

```
from keybert import KeyBERT
from collections import Counter

kw_model = KeyBERT(model="all-MiniLM-L6-v2")

cleaned = [clean(c) for c in comments]

all_keywords = []
for text in cleaned:
```

```
    keywords = kw_model.extract_keywords(text, keyphrase_ngram_range=(1, 2), stop_words='english', top_n=3)
    all_keywords.extend([kw for kw, _ in keywords])

keyword_counts = Counter(all_keywords)
top_keywords = keyword_counts.most_common(100)

top_keywords
```

```
[('bldc fan', 33),
 ('atomberg', 24),
 ('price', 22),
 ('fan', 17),
 ('bldc', 15),
 ('hai', 15),
 ('atomberg fan', 13),
 ('bhai', 12),
 ('fan hai', 10),
 ('hai kya', 9),
 ('rpm', 9),
 ('ceiling fan', 9),
 ('bldc fans', 8),
 ('best', 8),
 ('jata hai', 7),
 ('bakwas hai', 7),
 ('link', 7),
 ('kya hai', 7),
 ('nice', 7),
 ('fan ka', 7),
 ('air flow', 6),
 ('havells', 6),
 ('noise', 6),
 ('warranty', 5),
 ('air', 5),
 ('price kya', 5),
 ('raha hai', 5),
 ('ka hai', 5),
 ('buy atomberg', 5),
 ('good', 5),
 ('kya', 4),
 ('nahi hai', 4),
 ('hota hai', 4),
 ('चल रह', 4),
 ('exhaust', 4),
 ('crompton', 4),
 ('normal fan', 4),
 ('service', 4),
 ('super', 4),
 ('brother', 4),
 ('remote', 4),
 ('kharab ho', 4),
 ('air delivery', 4),
 ('best bldc', 4),
 ('review', 4),
 ('brush', 4),
 ('rod', 4),
 ('rahe hai', 3),
 ('ka fan', 3),
 ('company ka', 3),
 ('sakta hai', 3),
 ('air throw', 3),
 ('सरकट खरब', 3),
 ('speed', 3),
 ('atomberg best', 3),
 ('problem', 3),
 ('ceiling fans', 3),
 ('atomberg bakwas', 3),
```

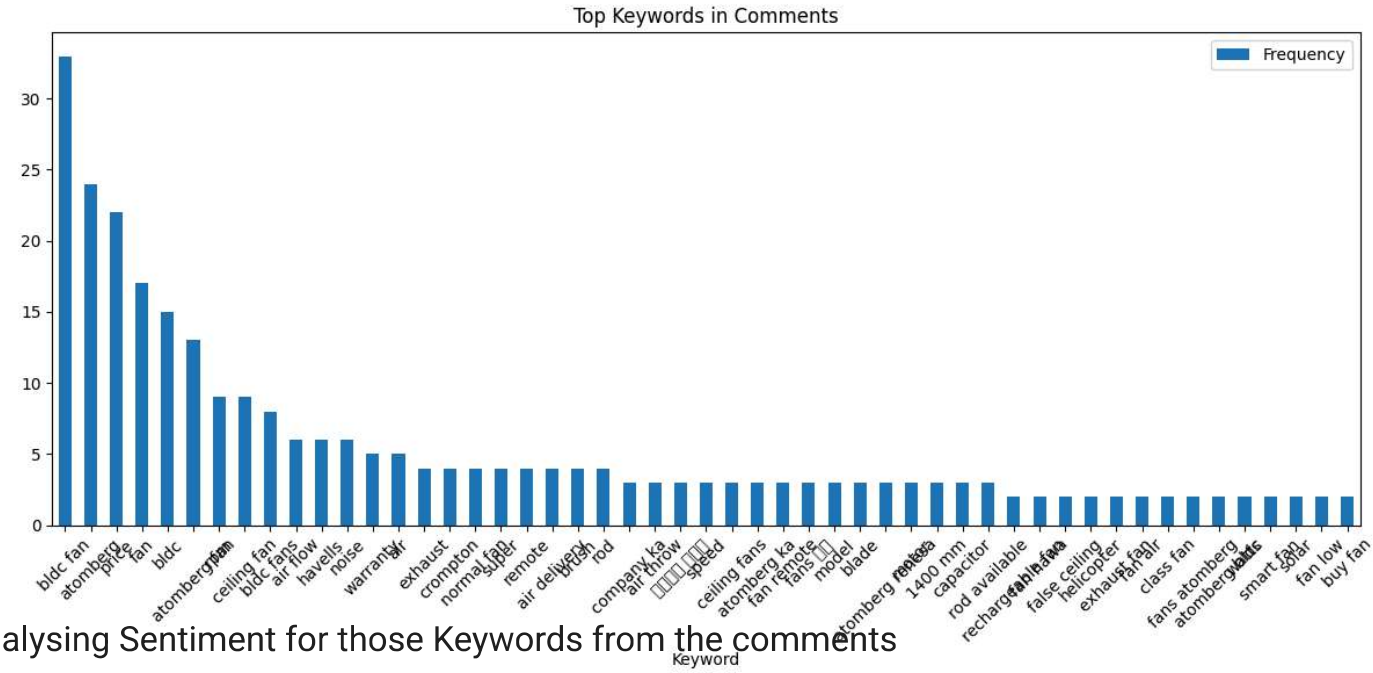
```
non_technical_terms = {
    'hai kya', 'hai', 'fan hai', 'bhai', 'jata hai', 'bakwas hai', 'best', 'good',
    'nice', 'hota hai', 'raha hai', 'ka hai', 'fan ka', 'kya hai', 'kya', 'review',
    'buy atomberg', 'link', 'price kya', 'nahi hai', 'ka fan', 'kharab ho',
    'sir', 'hai fan', 'bro', 'disclaimernever buy', '00zero servicepoor',
    'product 00zero', 'problem', 'service', 'atomberg bakwas',
    'bhai iska', 'hai ki', 'speed kaisa', 'buy fans', 'information',
    'excellent', 'tak engineer', 'rha hai', 'kitna', 'company hai', 'reviews aftersale',
    'disappointed aftersale', 'aftersale service', 'lagta hai', 'rahe hai',
    'चल रह', 'bolta hai', 'thank', 'thank sir', 'hai ye', 'bhi hai', 'dog', 'bara',
    'price koto', 'koto', 'worst fan', 'crompton best', 'atomberg best'
}

# Lowercase for matching
```

```
non_technical_terms = {t.lower() for t in non_technical_terms}

# Filter with partial match
top_keywords = [
    (kw, freq) for kw, freq in top_keywords
    if not any(nt in kw.lower() for nt in non_technical_terms)
]


top_keywords
```

```
[('bldc fan', 33),
 ('atomberg', 24),
 ('price', 22),
 ('fan', 17),
 ('bldc', 15),
 ('atomberg fan', 13),
 ('rpm', 9),
 ('ceiling fan', 9),
 ('bldc fans', 8),
 ('air flow', 6),
 ('havells', 6),
 ('noise', 6),
 ('warranty', 5),
 ('air', 5),
 ('exhaust', 4),
 ('crompton', 4),
 ('normal fan', 4),
 ('super', 4),
 ('remote', 4),
 ('air delivery', 4),
 ('brush', 4),
 ('rod', 4),
 ('company ka', 3),
 ('air throw', 3),
 ('सरकट खरब', 3),
 ('speed', 3),
 ('ceiling fans', 3),
 ('atomberg ka', 3),
 ('fan remote', 3),
 ('fans सल', 3),
 ('model', 3),
 ('blade', 3),
 ('atomberg renesa', 3),
 ('motor', 3),
 ('1400 mm', 3),
 ('capacitor', 3),
 ('rod available', 3),
 ('rechargeable fan', 2),
 ('fan hawa', 2),
 ('false ceiling', 2),
 ('helicopter', 2),
 ('exhaust fan', 2),
 ('fan air', 2),
 ('class fan', 2),
 ('fans atomberg', 2),
 ('atomberg bldc', 2),
 ('watts', 2),
 ('smart fan', 2),
 ('solar', 2),
 ('fan low', 2),
 ('buy fan', 2)]
```

```
import pandas as pd
import matplotlib.pyplot as plt

df_keywords = pd.DataFrame(top_keywords, columns=["Keyword", "Frequency"])
df_keywords.plot(x="Keyword", y="Frequency", kind="bar", figsize=(12,6), title="Top Keywords in Comments")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2115403435.py:7: UserWarning: Glyph 2360 (\N{DEVANAGARI LETTER SA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2115403435.py:7: UserWarning: Matplotlib currently does not support Devanagari natively.
  plt.tight_layout()
/tmp/ipython-input-2115403435.py:7: UserWarning: Glyph 2352 (\N{DEVANAGARI LETTER RA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2115403435.py:7: UserWarning: Glyph 2325 (\N{DEVANAGARI LETTER KA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2115403435.py:7: UserWarning: Glyph 2335 (\N{DEVANAGARI LETTER TTA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2115403435.py:7: UserWarning: Glyph 2326 (\N{DEVANAGARI LETTER KHA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2115403435.py:7: UserWarning: Glyph 2348 (\N{DEVANAGARI LETTER BA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2115403435.py:7: UserWarning: Glyph 2354 (\N{DEVANAGARI LETTER LA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2360 (\N{DEVANAGARI LETTER SA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Matplotlib currently does not support Devanagari
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2352 (\N{DEVANAGARI LETTER RA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2325 (\N{DEVANAGARI LETTER KA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2335 (\N{DEVANAGARI LETTER TTA}) missing fr
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2326 (\N{DEVANAGARI LETTER KHA}) missing fr
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2348 (\N{DEVANAGARI LETTER BA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2354 (\N{DEVANAGARI LETTER LA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
```

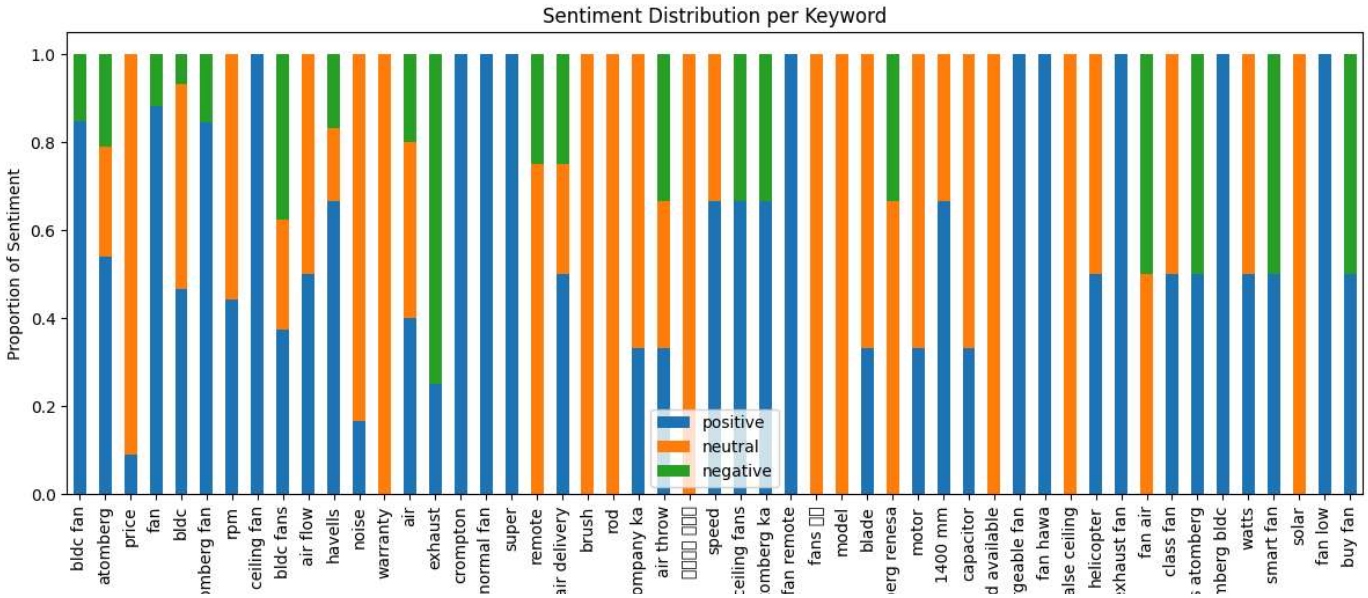


Analysing Sentiment for those Keywords from the comments

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Initialize VADER analyzer
analyzer = SentimentIntensityAnalyzer()

# Analyze sentiment
comment_sentiments = []
for comment in cleaned:
    score = analyzer.polarity_scores(comment)['compound']
    if score >= 0.05:
        sentiment = 'positive'
    elif score <= -0.05:
        sentiment = 'negative'
    else:
        sentiment = 'neutral'
    comment_sentiments.append((comment, sentiment))
```

```
from collections import defaultdict

# Initialize dictionary to count sentiments for each keyword
keyword_sentiment_stats = defaultdict(lambda: {'positive': 0, 'negative': 0, 'neutral': 0})

for (comment, sentiment) in comment_sentiments:
    for keyword, _ in kw_model.extract_keywords(comment, keyphrase_ngram_range=(1, 2), stop_words='english', top_n=3):
        if keyword in dict(top_keywords):
            keyword_sentiment_stats[keyword][sentiment] += 1



import pandas as pd

# Convert to DataFrame
df_keyword_sentiment = pd.DataFrame.from_dict(keyword_sentiment_stats, orient='index')
top_keywords_only = [kw for kw, _ in top_keywords]
df_keyword_sentiment = df_keyword_sentiment.loc[top_keywords_only]

#df_keyword_sentiment = df_keyword_sentiment.loc[top_keywords]
 # Keep only top 30
df_keyword_sentiment['total'] = df_keyword_sentiment.sum(axis=1)

# Optional: Normalize
df_keyword_sentiment_norm = df_keyword_sentiment.div(df_keyword_sentiment['total'], axis=0)

import matplotlib.pyplot as plt

df_keyword_sentiment_norm[['positive', 'neutral', 'negative']].plot(kind='bar', stacked=True, figsize=(12, 6))
plt.title('Sentiment Distribution per Keyword')
plt.ylabel('Proportion of Sentiment')
plt.xlabel('Keyword')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2928991205.py:21: UserWarning: Glyph 2360 (\N{DEVANAGARI LETTER SA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2928991205.py:21: UserWarning: Matplotlib currently does not support Devanagari natively.
  plt.tight_layout()
/tmp/ipython-input-2928991205.py:21: UserWarning: Glyph 2352 (\N{DEVANAGARI LETTER RA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2928991205.py:21: UserWarning: Glyph 2325 (\N{DEVANAGARI LETTER KA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2928991205.py:21: UserWarning: Glyph 2335 (\N{DEVANAGARI LETTER TTA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2928991205.py:21: UserWarning: Glyph 2326 (\N{DEVANAGARI LETTER KHA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2928991205.py:21: UserWarning: Glyph 2348 (\N{DEVANAGARI LETTER BA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/tmp/ipython-input-2928991205.py:21: UserWarning: Glyph 2354 (\N{DEVANAGARI LETTER LA}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2360 (\N{DEVANAGARI LETTER SA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Matplotlib currently does not support Devanagari
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2352 (\N{DEVANAGARI LETTER RA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2325 (\N{DEVANAGARI LETTER KA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2335 (\N{DEVANAGARI LETTER TTA}) missing fr
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2326 (\N{DEVANAGARI LETTER KHA}) missing fr
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2348 (\N{DEVANAGARI LETTER BA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 2354 (\N{DEVANAGARI LETTER LA}) missing fro
  fig.canvas.print_figure(bytes_io, **kw)
```




```
keyword_sentiment_stats
```

```
defaultdict(<function __main__.<lambda>()>,
            {'bldc fan': {'positive': 28, 'negative': 5, 'neutral': 0},
             'bldc': {'positive': 7, 'negative': 1, 'neutral': 7},
             'rechargeable fan': {'positive': 2, 'negative': 0, 'neutral': 0},
             'warranty': {'positive': 0, 'negative': 0, 'neutral': 5},
             'air flow': {'positive': 3, 'negative': 0, 'neutral': 3},
             'air': {'positive': 2, 'negative': 1, 'neutral': 2},
             'fan hawa': {'positive': 2, 'negative': 0, 'neutral': 0},
             'price': {'positive': 2, 'negative': 0, 'neutral': 20},
             'fan': {'positive': 15, 'negative': 2, 'neutral': 0},
             'false ceiling': {'positive': 0, 'negative': 0, 'neutral': 2},
             'helicopter': {'positive': 1, 'negative': 0, 'neutral': 1},
             'exhaust': {'positive': 1, 'negative': 3, 'neutral': 0},
             'exhaust fan': {'positive': 2, 'negative': 0, 'neutral': 0},
             'atomberg': {'positive': 13, 'negative': 5, 'neutral': 6},
             'crompton': {'positive': 4, 'negative': 0, 'neutral': 0},
             'rpm': {'positive': 4, 'negative': 0, 'neutral': 5},
             'fan air': {'positive': 0, 'negative': 1, 'neutral': 1},
             'class fan': {'positive': 1, 'negative': 0, 'neutral': 1},
             'fans atomberg': {'positive': 1, 'negative': 1, 'neutral': 0},
             'atomberg bldc': {'positive': 2, 'negative': 0, 'neutral': 0},
             'watts': {'positive': 1, 'negative': 0, 'neutral': 1},
             'normal fan': {'positive': 4, 'negative': 0, 'neutral': 0},
```

```
'smart fan': {'positive': 1, 'negative': 1, 'neutral': 0},
'solar': {'positive': 0, 'negative': 0, 'neutral': 2},
'fan low': {'positive': 2, 'negative': 0, 'neutral': 0},
'company ka': {'positive': 1, 'negative': 0, 'neutral': 2},
'ceiling fan': {'positive': 9, 'negative': 0, 'neutral': 0},
'super': {'positive': 4, 'negative': 0, 'neutral': 0},
'bldc fans': {'positive': 3, 'negative': 3, 'neutral': 2},
'buy fan': {'positive': 1, 'negative': 1, 'neutral': 0},
'atomberg fan': {'positive': 11, 'negative': 2, 'neutral': 0},
'air throw': {'positive': 1, 'negative': 1, 'neutral': 1},
'remote': {'positive': 0, 'negative': 1, 'neutral': 3},
'सरकट खरब': {'positive': 0, 'negative': 0, 'neutral': 3},
'speed': {'positive': 2, 'negative': 0, 'neutral': 1},
'ceiling fans': {'positive': 2, 'negative': 1, 'neutral': 0},
'atomberg ka': {'positive': 2, 'negative': 1, 'neutral': 0},
'fan remote': {'positive': 3, 'negative': 0, 'neutral': 0},
'fans सल': {'positive': 0, 'negative': 0, 'neutral': 3},
'havells': {'positive': 4, 'negative': 1, 'neutral': 1},
'air delivery': {'positive': 2, 'negative': 1, 'neutral': 1},
'model': {'positive': 0, 'negative': 0, 'neutral': 3},
'blade': {'positive': 1, 'negative': 0, 'neutral': 2},
'noise': {'positive': 1, 'negative': 0, 'neutral': 5},
'atomberg renesa': {'positive': 0, 'negative': 1, 'neutral': 2},
'brush': {'positive': 0, 'negative': 0, 'neutral': 4},
'motor': {'positive': 1, 'negative': 0, 'neutral': 2},
'1400 mm': {'positive': 2, 'negative': 0, 'neutral': 1},
'capacitor': {'positive': 1, 'negative': 0, 'neutral': 2},
'rod': {'positive': 0, 'negative': 0, 'neutral': 4},
'rod available': {'positive': 0, 'negative': 0, 'neutral': 3}})
```

Filtering out Atomberg comments

```
# 1. Filter for Atomberg videos only
atomberg_videos = exploded_df[
    exploded_df['mentioned_brands'].str.lower() == 'atomberg'
]['video_id'].unique()

# 2. Optional: sort by view_count if you want top N Atomberg videos
atomberg_top_videos = final_df[final_df['video_id'].isin(atomberg_videos)] \
    .sort_values(by="view_count", ascending=False) \
    .head(100)

# 3. Extract comments only for Atomberg videos
all_comments_atomberg = []
for vid in atomberg_top_videos['video_id']:
    all_comments_atomberg.extend(get_video_comments(vid, max_comments=20))

# 4. Convert to DataFrame
comments_df_atomberg = pd.DataFrame(all_comments_atomberg)
```

```
WARNING:googleapiclient.http:Encountered 403 Forbidden with reason "commentsDisabled"
Error fetching comments for video YFkEL1qENJo: <HttpError 403 when requesting https://youtube.googleapis.com/youtube/v3/commentThreads?p
```

```
comments_df_atomberg
```

	video_id	author	comment	like_count	published_at
0	u_kxhHjHIn0	@GautamPatel-hb7zk	Speed kaise hai fan ki	1	2025-07-31T15:35:08Z
1	u_kxhHjHIn0	@Littleheart-b4r	😂😂😂😂❤❤❤	1	2025-07-31T10:42:40Z
2	u_kxhHjHIn0	@vish_arjun	10/8❤	1	2025-07-31T09:05:01Z
3	u_kxhHjHIn0	@rajeshbaghel5394	kya itna saund	1	2025-07-20T10:17:06Z
4	u_kxhHjHIn0	@theaarnikshow3057	Atomberg bakwas hai circuit bar bar bigad jata...	0	2025-07-20T06:17:36Z
...	...	...	...	...	...
1385	WlqMeJtOjEw	@Sanju_craftss	Sir sabse best colour konsa rahega black ya wh...	1	2024-05-26T05:23:03Z
1386	WlqMeJtOjEw	@AjaySingh-uj1lk	Renesa ke jagah isko lena thik rahega	1	2024-05-18T06:54:13Z
1387	WlqMeJtOjEw	@mukundtechtips	Can you review renesa zen ?	0	2024-05-16T09:09:53Z
1388	WlqMeJtOjEw	@mohitsaini1971	An urge to you pls dont buy atomberg fans just...	0	2024-05-02T06:39:45Z
1389	WlqMeJtOjEw	@avinashbhatt2317	price...? flipkart and amazon me	0	2024-05-01T12:58:15Z

1390 rows × 5 columns

```
import re

def clean_text(text):
    text = str(text).lower()                        # lowercase
    text = re.sub(r"http\S+", "", text)              # remove URLs
    text = re.sub(r"[^a-z0-9\s]", "", text)          # remove punctuation
    text = re.sub(r"\s+", " ", text).strip()         # normalize spaces
    return text

comments_df_atomberg["clean_comment"] = comments_df_atomberg["comment"].apply(clean_text)
```

Embedding the comments into Qdrant and obtaining Keywords

```
# This assumes you already filtered to comments_df_atomberg
embeddings = model.encode(comments_df_atomberg["clean_comment"].tolist())

client.recreate_collection(
    collection_name="comments_collection_atomberg",
    vectors_config=VectorParams(size=384, distance=Distance.COSINE)
)

points = [
    PointStruct(
        id=i,
        vector=embeddings[i],
        payload={
            "comment": comments_df_atomberg.iloc[i]["comment"],
            "video_id": comments_df_atomberg.iloc[i]["video_id"],
            "author": comments_df_atomberg.iloc[i]["author"]
        }
    )
    for i in range(len(comments_df_atomberg))
]

client.upsert(collection_name="comments_collection_atomberg", points=points)
```

```
/tmp/ipython-input-2828704932.py:4: DeprecationWarning: `recreate_collection` method is deprecated and will be removed in the future. Us
  client.recreate_collection(
UpdateResult(operation_id=0, status=<UpdateStatus.COMPLETED: 'completed'>)
```

```
retrieved = client.scroll(
    collection_name="comments_collection_atomberg",
    with_payload=True,
    limit=1000
)

comments = [point.payload["comment"] for point in retrieved[0]]
cleaned = [clean(c) for c in comments]
```

```
all_keywords = []
for text in cleaned:
    keywords = kw_model.extract_keywords(
        text, keyphrase_ngram_range=(1, 2), stop_words='english', top_n=3
    )
    all_keywords.extend([kw for kw, _ in keywords])

from collections import Counter
top_keywords = Counter(all_keywords).most_common(100)



top_keywords
```

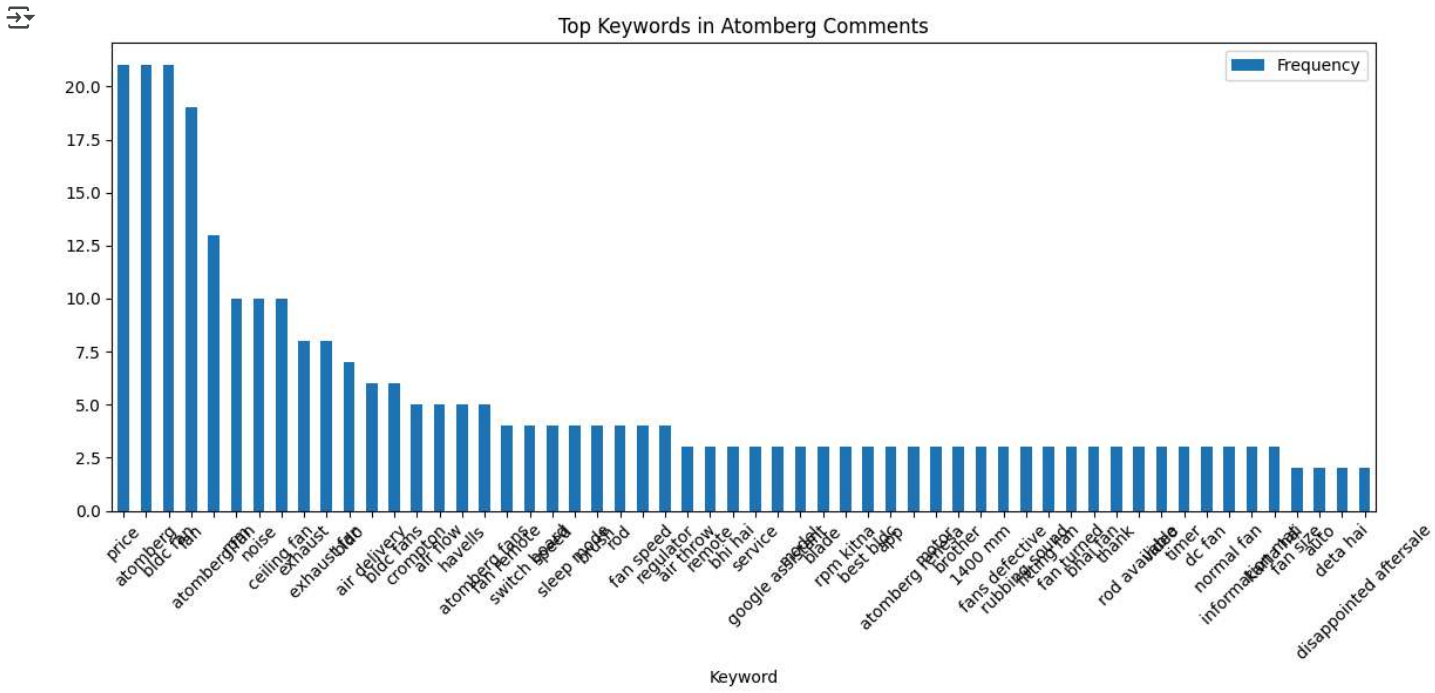
```
[('price', 21),
 ('atomberg', 21),
 ('bldc fan', 21),
 ('fan', 19),
 ('bhai', 15),
 ('hai kya', 14),
 ('hai', 14),
 ('atomberg fan', 13),
 ('fan hai', 11),
 ('rpm', 10),
 ('noise', 10),
 ('ceiling fan', 10),
 ('exhaust', 8),
 ('exhaust fan', 8),
 ('jata hai', 7),
 ('bakwas hai', 7),
 ('best', 7),
 ('bldc', 7),
 ('nice', 7),
 ('hota hai', 6),
 ('raha hai', 6),
 ('ka hai', 6),
 ('fan ka', 6),
 ('good', 6),
 ('air delivery', 6),
 ('bldc fans', 6),
 ('crompton', 5),
 ('buy atomberg', 5),
 ('air flow', 5),
 ('link', 5),
 ('kya hai', 5),
 ('havells', 5),
 ('atomberg fans', 5),
 ('nahi hai', 4),
 ('ka fan', 4),
 ('bro', 4),
 ('kharab ho', 4),
 ('atomberg ka', 4),
 ('fan remote', 4),
 ('switch board', 4),
 ('speed', 4),
 ('sleep mode', 4),
 ('brush', 4),
 ('sir', 4),
 ('review', 4),
 ('rod', 4),
 ('fan speed', 4),
 ('regulator', 4),
 ('hai fan', 3),
 ('kya', 3),
 ('air throw', 3),
 ('disclaimernever buy', 3),
 ('00zero servicepoor', 3),
 ('product 00zero', 3),
 ('remote', 3),
 ('bhi hai', 3),
 ('atomberg best', 3),
 ('problem', 3),
```

```
non_technical_terms = {
    'hai kya', 'hai', 'fan hai', 'bhai', 'jata hai', 'bakwas hai', 'best', 'good',
    'nice', 'hota hai', 'raha hai', 'ka hai', 'fan ka', 'kya hai', 'kya', 'review',
    'buy atomberg', 'link', 'price kya', 'nahi hai', 'ka fan', 'kharab ho',
    'atomberg ka', 'sir', 'hai fan', 'bro', 'disclaimernever buy', '00zero servicepoor',
    'product 00zero', 'atomberg best', 'problem', 'atomberg bakwas',
    'bhai iska', 'hai ki', 'speed kaisa', 'hoti hai', 'buy fans', 'information',
    'excellent', 'tak engineer', 'rha hai', 'kitna', 'company hai', 'reviews aftersale',
```

```
    'aftersale service', 'lagta hai', 'rahe hai',
    'चल रह', 'bolta hai'
}

# Filter out non-technical terms
top_keywords = [
    (kw, freq) for kw, freq in top_keywords
    if kw.lower() not in non_technical_terms
]
top_keywords
```

```
[('price', 21),
 ('atomberg', 21),
 ('bldc fan', 21),
 ('fan', 19),
 ('atomberg fan', 13),
 ('rpm', 10),
 ('noise', 10),
 ('ceiling fan', 10),
 ('exhaust', 8),
 ('exhaust fan', 8),
 ('bldc', 7),
 ('air delivery', 6),
 ('bldc fans', 6),
 ('crompton', 5),
 ('air flow', 5),
 ('havells', 5),
 ('atomberg fans', 5),
 ('fan remote', 4),
 ('switch board', 4),
 ('speed', 4),
 ('sleep mode', 4),
 ('brush', 4),
 ('rod', 4),
 ('fan speed', 4),
 ('regulator', 4),
 ('air throw', 3),
 ('remote', 3),
 ('bhi hai', 3),
 ('service', 3),
 ('google assistant', 3),
 ('model', 3),
 ('blade', 3),
 ('rpm kitna', 3),
 ('best bldc', 3),
 ('app', 3),
 ('atomberg renesa', 3),
 ('motor', 3),
 ('brother', 3),
 ('1400 mm', 3),
 ('fans defective', 3),
 ('rubbing sound', 3),
 ('fitting fan', 3),
 ('fan turned', 3),
 ('bhai fan', 3),
 ('thank', 3),
 ('rod available', 3),
 ('video', 3),
 ('timer', 3),
 ('dc fan', 3),
 ('normal fan', 3),
 ('information mat', 3),
 ('karta hai', 3),
 ('fan size', 2),
 ('auto', 2),
 ('deta hai', 2),
 ('disappointed aftersale', 2)]
```

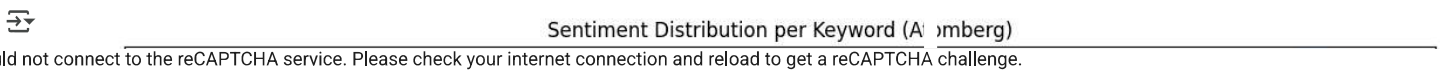
```
df_keywords = pd.DataFrame(top_keywords, columns=["Keyword", "Frequency"])
df_keywords.plot(
    x="Keyword", y="Frequency", kind="bar", figsize=(12, 6),
    title="Top Keywords in Atomberg Comments"
)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Analysing Sentiment for those Key words

```
# 5️⃣ Sentiment analysis
analyzer = SentimentIntensityAnalyzer()
comment_sentiments = []
for comment in cleaned:
    score = analyzer.polarity_scores(comment)['compound']
    if score >= 0.05:
        sentiment = 'positive'
    elif score <= -0.05:
        sentiment = 'negative'
    else:
        sentiment = 'neutral'
    comment_sentiments.append((comment, sentiment))

# 6️⃣ Sentiment per keyword
keyword_sentiment_stats = defaultdict(lambda: {'positive': 0, 'negative': 0, 'neutral': 0}

for (comment, sentiment) in comment_sentiments:
    for keyword, _ in kw_model.extract_keywords(
        comment, keyphrase_ngram_range=(1, 2),
        stop_words='english', top_n=3
    ):
        if keyword in dict(top_keywords):
            keyword_sentiment_stats[keyword][sentiment] += 1

# 7️⃣ Convert to DataFrame & normalize
df_keyword_sentiment = pd.DataFrame.from_dict(keyword_sentiment_stats, orient='index')
top_keywords_only = [kw for kw, _ in top_keywords]
df_keyword_sentiment = df_keyword_sentiment.loc[top_keywords_only]
df_keyword_sentiment['total'] = df_keyword_sentiment.sum(axis=1)
df_keyword_sentiment_norm = df_keyword_sentiment.div(df_keyword_sentiment['total'], axis=0

# 8️⃣ Plot sentiment distribution
df_keyword_sentiment_norm[['positive', 'neutral', 'negative']].plot(
    kind='bar', stacked=True, figsize=(12, 6)
)
plt.title('Sentiment Distribution per Keyword (Atomberg)')
plt.ylabel('Proportion of Sentiment')
plt.xlabel('Keyword')
```

```
plt.tight_layout()
plt.show()
```



Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.