# 🏥 Medical Tourism in Turkey: A Reddit-Based Analysis

This notebook explores the growing trend of medical tourism in Turkey by analyzing discussions from Reddit. Using natural language processing and data visualization techniques, we investigate user sentiment and popular medical procedures such as hair transplants, cosmetic surgeries, and dental treatments.

---

## 📌 Objectives

- Collect and analyze Reddit posts related to medical tourism in Turkey.
- Identify the most discussed procedures and sentiments.
- Visualize user interest and trends over time.

---

## 🧰 Libraries and Configuration

We begin by importing the necessary libraries for data handling, visualization, and Reddit API access. Additionally, we set visualization parameters to ensure consistent styling throughout the notebook.

In [3]:
```
# Data handling
import pandas as pd
from datetime import datetime
import time
import warnings

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Warnings and aesthetics
warnings.filterwarnings("ignore")
sns.set_theme(style="whitegrid")
plt.rcParams["figure.figsize"] = (10, 6)
pd.set_option('display.max_columns', 500)
```

## 📜 Reddit Data Collection Methodology

Data was collected using the Reddit API through the `praw` library. A wide range of keywords were used to query various subreddits related to health, travel, and personal experiences. Posts and their associated comments were stored in structured dataframes.

The following Python code snippet demonstrates the data collection process used to generate the dataset:

```
import praw
```

```
from datetime import datetime
import time
import pandas as pd

# Initialize Reddit API
reddit = praw.Reddit(
    client_id="yourid",
    client_secret="yourclientsecret",
    user_agent="turkey-health-tourism:v1.0"
)

# Keywords to search for
query_terms = [
    "hair transplant Turkey", "dental implants Turkey", "cosmetic
surgery Turkey",
    "plastic surgery Turkey", "rhinoplasty Turkey", "BBL Turkey",
"liposuction Turkey",
    "boob job Turkey", "IVF Turkey", "weight loss surgery Turkey",
"medical tourism Turkey",
    "is Turkey safe for surgery", "surgery in Turkey", "healthcare
in Turkey",
    "dental treatment in Turkey", "facelift Turkey", "botox in
Turkey",
    "gastric sleeve Turkey", "cheap surgery Turkey", "clinics in
Turkey reviews",
    "dental crowns Turkey", "veneers Turkey", "nose job Turkey",
    "laser eye surgery Turkey", "health insurance Turkey", "breast
implants Turkey",
    "cheap medical care Turkey", "best clinics Turkey", "transplant
reviews Turkey",
    "IVF cost Turkey", "medical travel Turkey"
]

# Targeted subreddits
subreddits = [
    "healthcare", "travel", "AskEurope", "expat", "Turkey",
    "PlasticSurgery", "HairTransplants", "IWantOut", "TravelHacks",
    "Europe", "askmedicine", "LongDistance"
]

posts_data = []
comments_data = []

# Data collection loop
for sub in subreddits:
    subreddit = reddit.subreddit(sub)
    print(f"\n🔍 Subreddit: r/{sub}")
    for query in query_terms:
        print(f"   🔑 Searching for: {query}")
        try:
            for post in subreddit.search(query, sort="relevance",
time_filter="all", limit=200):
                posts_data.append({
                    "query": query,
                    "subreddit": sub,
                    "title": post.title,
                    "text": post.selftext,
```

```
                        "score": post.score,
                        "comments": post.num_comments,
                        "url": post.url,
                        "created_utc":
datetime.utcfromtimestamp(post.created_utc).strftime('%Y-%m-%d'),
                        "id": post.id
                    })

                    # Extract comments
                    try:
                        post.comments.replace_more(limit=0)
                        for comment in post.comments.list():
                            comments_data.append({
                                "post_id": post.id,
                                "comment_text": comment.body,
                                "comment_score": comment.score
                            })
                    except Exception as ce:
                        print(f"    ⚠️ Comment error: {ce}")
                time.sleep(1)  # Prevent rate limiting
            except Exception as e:
                print(f"⚠️ Error ({sub} / {query}): {e}")
                continue

# Convert to DataFrames
posts_df = pd.DataFrame(posts_data)
comments_df = pd.DataFrame(comments_data)

# Save as CSV
posts_df.to_csv("reddit_health_tourism_turkey_posts.csv",
index=False)
comments_df.to_csv("reddit_health_tourism_turkey_comments.csv",
index=False)
```

## 📂 Load Dataset & Initial Exploration

We load the previously collected Reddit data from CSV files. The data consists of two parts:

- **Posts Dataset:** Includes metadata and textual content from Reddit submissions.
- **Comments Dataset:** Contains comments related to each post, including their content and scores.

Let's begin by reading both files and previewing the structure.

In [9]:

```
# Load the datasets
posts_df = pd.read_csv("reddit_health_tourism_turkey_posts.csv")
comments_df = pd.read_csv("reddit_health_tourism_turkey_comments.csv")

# Preview the datasets
print("📝 Posts Dataset:")
display(posts_df.head())

print("\n💬 Comments Dataset:")
display(comments_df.head())
```

Posts Dataset:

| | query | subreddit | title | text |
|---|---|---|---|---|
| 0 | hair transplant Turkey | healthcare | Country medical Specialties - which country ha... | i have noticed memes about turkey with hair tr... |
| 1 | dental implants Turkey | healthcare | [Question - Other] How much is a dental implan... | Long story short, I broke my front tooth and I... |
| 2 | dental implants Turkey | healthcare | Orthodontics - private cosmetics dominated and... | [https://www.facebook.com/groups/orthodonticma... |
| 3 | dental implants Turkey | healthcare | Most affordable state for overseas dental work? | I need a Vivos implant to treat my sleep apnea... |
| 4 | dental implants Turkey | healthcare | [discussion] Dental Health should be included ... | More than 1 in 4 (27%) adults in the United St... |

Comments Dataset:

| | post_id | comment_text | comment_score |
|---|---|---|---|
| 0 | 1k6i84c | For clarity, stem cell therapies are being agg... | 2 |
| 1 | 1k6i84c | Mexico is not standard of care for medical tou... | 1 |
| 2 | 1k6i84c | India | 1 |
| 3 | e1h1xf | This might be a better question for r/healthin... | 1 |
| 4 | e1h1xf | If that includes everything it's a bargain. | 1 |

## 📊 Dataset Summary & Visual Analysis

After loading the dataset, we conduct an initial exploratory analysis to understand its structure and key statistics.

- **Shape**: Number of rows and columns.
- **Missing Values**: Check for null values in each column.
- **Subreddits & Queries**: Count of unique subreddits and query terms.
- **Post Distribution**: Visualizations by year, subreddit, and monthly trend.
- **Engagement**: Top scoring posts and average metrics per subreddit.

In [12]:

In [12]:
```
# Basic overview of the dataset
print("🔢 Dataset Shape:", posts_df.shape)
print("🧱 Column Names:", posts_df.columns.tolist())
```

```
🔢 Dataset Shape: (14428, 9)
🧱 Column Names: ['query', 'subreddit', 'title', 'text', 'score', 'comments',
'url', 'created_utc', 'id']
```

In [14]:
```
# Check for missing values
print("🧹 Missing Values:\n", posts_df.isnull().sum())
```

```
🧹 Missing Values:
 query             0
subreddit         0
title             0
text           1476
score             0
comments          0
url               0
created_utc       0
id                0
dtype: int64
```

In [16]:
```
# Number of unique subreddits and search queries
print("📊 Number of Unique Subreddits:", posts_df['subreddit'].nunique())
print("🔎 Number of Unique Search Queries:", posts_df['query'].nunique())
```

```
📊 Number of Unique Subreddits: 11
🔎 Number of Unique Search Queries: 31
```

In [18]:
```
# Top 10 most active subreddits
print("\n🔝 Top 10 Most Active Subreddits:")
print(posts_df['subreddit'].value_counts().head(10))
```

```
🔝 Top 10 Most Active Subreddits:
subreddit
PlasticSurgery     3893
HairTransplants    3348
Turkey             2129
travel             1500
IWantOut           1032
Europe              771
LongDistance        467
TravelHacks         403
healthcare          398
AskEurope           330
Name: count, dtype: int64
```

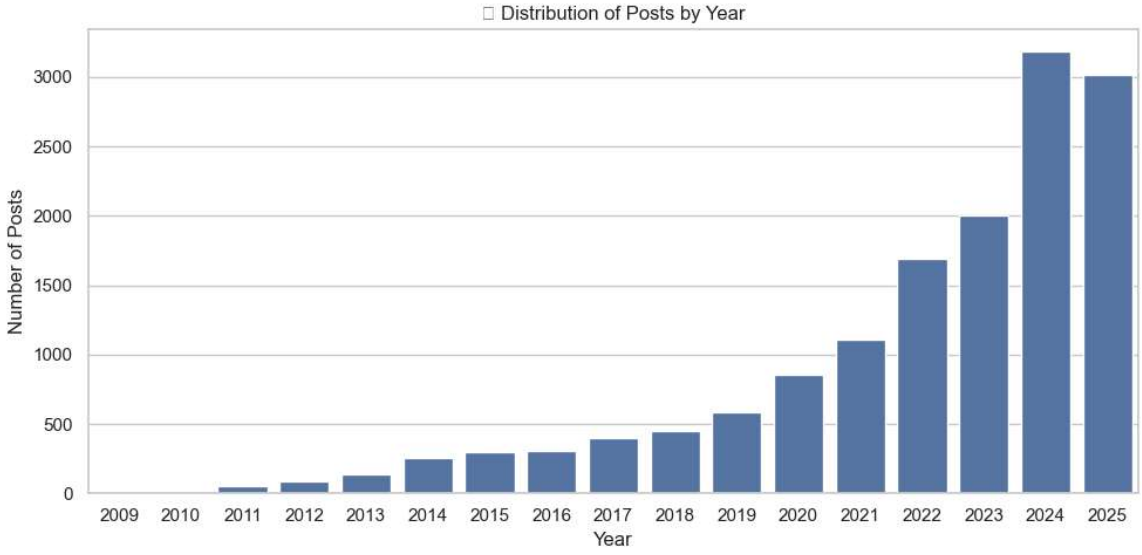
In [20]:
```
# Top 10 most frequent search queries
print("\n🔑 Top 10 Most Frequent Search Queries:")
print(posts_df['query'].value_counts().head(10))
```

```
🔑 Top 10 Most Frequent Search Queries:
query
medical travel Turkey        1664
health insurance Turkey      1649
best clinics Turkey          1132
nose job Turkey               980
boob job Turkey               938
IVF cost Turkey               926
cheap surgery Turkey          896
medical tourism Turkey        862
```

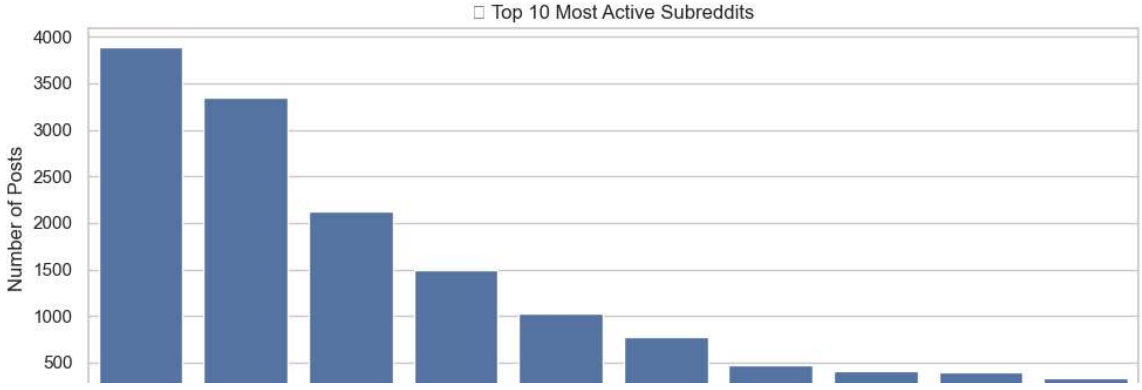
```
transplant reviews Turkey     590
plastic surgery Turkey        545
Name: count, dtype: int64
```

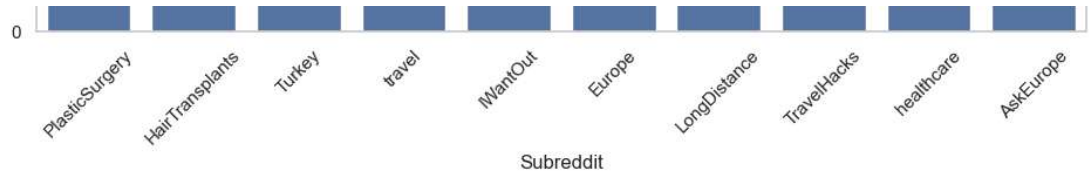
In [24]:
```
posts_df['created_utc'] = pd.to_datetime(posts_df['created_utc'], errors='coe
posts_df['year'] = posts_df['created_utc'].dt.year
posts_df['month'] = posts_df['created_utc'].dt.to_period('M')
```

In [26]:
```
# Distribution of posts by year
yearly_counts = posts_df['year'].value_counts().sort_index()
plt.figure(figsize=(10, 5))
sns.barplot(x=yearly_counts.index, y=yearly_counts.values)
plt.title("🗓️ Distribution of Posts by Year")
plt.xlabel("Year")
plt.ylabel("Number of Posts")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



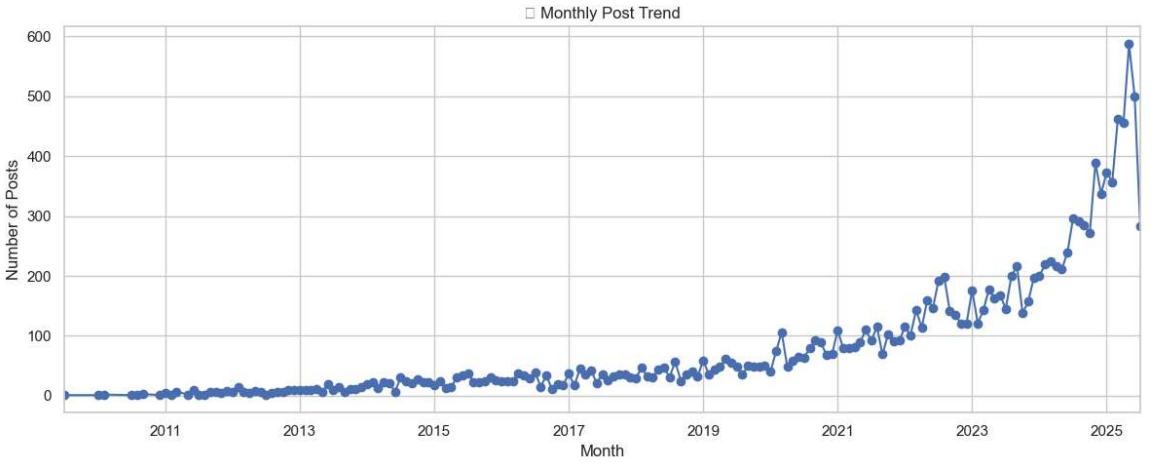

In [28]:
```
# Top 10 subreddits by number of posts
top_subs = posts_df['subreddit'].value_counts().head(10)
plt.figure(figsize=(10, 5))
sns.barplot(x=top_subs.index, y=top_subs.values)
plt.title("🔝 Top 10 Most Active Subreddits")
plt.xlabel("Subreddit")
plt.ylabel("Number of Posts")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```






In [30]:

```
# Monthly post trend
monthly_counts = posts_df['month'].value_counts().sort_index()
plt.figure(figsize=(12, 5))
monthly_counts.plot(marker='o')
plt.title("📈 Monthly Post Trend")
plt.xlabel("Month")
plt.ylabel("Number of Posts")
plt.grid(True)
plt.tight_layout()
plt.show()
```




In [32]:

```
# Top 10 highest scoring posts
top_scored = posts_df[['title', 'score']].sort_values(by='score', ascending=F
print("\n🏆 Top 10 Highest Scoring Posts:")
print(top_scored)
```

🏆 Top 10 Highest Scoring Posts:

| | title | score |
|---|---|---|
| 13407 | Europe In The Style Of Super Mario World | 17704 |
| 13233 | Europe In The Style Of Super Mario World | 17691 |
| 13292 | Dutch And German Citizens Can Not Stay In This... | 13080 |
| 1806 | This summer I took a hot air balloon trip over... | 10251 |
| 1884 | Cappadocia, Turkey. One of the most magical ex... | 7511 |
| 13209 | Turkey has 2d architecture just like Europe co... | 6822 |
| 1749 | *Update* Day: 101. I have now walked over 2300... | 6566 |
| 685 | Samarkand, Uzbekistan | 6268 |
| 1833 | Samarkand, Uzbekistan | 6264 |
| 13475 | Gun Deaths in Europe | 6248 |

In [34]:

```
# Average score and comment count per subreddit
agg_stats = posts_df.groupby('subreddit')[['score', 'comments']].mean().sort_
print("\n📊 Average Score and Comment Count per Subreddit (Top 10):")
print(agg_stats)
```

📊 Average Score and Comment Count per Subreddit (Top 10):

| subreddit | score | comments |
|---|---|---|
| Europe | 261.024643 | 114.212711 |

```
travel            115.299333   60.919333
PlasticSurgery     57.899820   17.928333
Turkey             54.056834   26.616252
IWantOut           36.855620   24.833333
AskEurope          29.754545   57.981818
healthcare         16.972362   13.283920
LongDistance       15.314775    8.783726
HairTransplants    15.271207   21.488650
TravelHacks        15.109181   23.300248
```

## Data Cleaning & Text Preprocessing

To ensure high-quality results in downstream analysis, we perform data cleaning and text preprocessing. The main steps include:

- Removing empty or irrelevant posts
- Lowercasing all text
- Removing URLs, special characters, and emojis
- (Optional) Stopword removal and tokenization for NLP tasks

In [37]:
```
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

In [39]:
```
# 📥 Download necessary NLTK resources (run only once)
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\lofit\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\lofit\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\lofit\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[39]: True

In [41]:
```
# Initialize stopwords and lemmatizer
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
```

In [43]:
```
# 🧹 Text cleaning function
def clean_text(text):
    if pd.isnull(text):
        return ""

    # Convert to lowercase
    text = text.lower()

    # Remove URLs, emails, HTML tags, and non-ASCII characters
    text = re.sub(r"http\S+|www\.\S+", '', text)
```

```
    text = re.sub(r"\S+@\S+", '', text)
    text = re.sub(r"<.*?>", '', text)
    text = re.sub(r"[^\x00-\x7F]+", " ", text)

    # Keep only letters and whitespace
    text = re.sub(r"[^a-z\s]", " ", text)

    # Tokenize
    tokens = nltk.word_tokenize(text)

    # Remove stopwords and lemmatize tokens
    cleaned_tokens = [
        lemmatizer.lemmatize(token)
        for token in tokens
        if token.isalpha() and token not in stop_words
    ]

    return " ".join(cleaned_tokens)
```

In [45]:

```
# 🧪 Apply cleaning to the dataset
posts_df['raw_text'] = posts_df['title'].fillna('') + ' ' + posts_df['text'].
posts_df['clean_text'] = posts_df['raw_text'].apply(clean_text)
```

In [46]:

```
# 🔍 Show sample cleaned text
print("\n✅ Sample Cleaned Texts:")
print(posts_df['clean_text'].sample(5).to_string(index=False))
```

```
✅ Sample Cleaned Texts:
best place turkey plastic surgery hey anyone ad...
thanks bottom heart turkey sends full load medi...
suggestion two week trip anywhere world hi seni...
safe visit turkey unmarried interracial couple ...
rhinoplasty month post op selim turan turkey su...
```

## ☁️ Word Cloud of Most Frequent Terms

To better understand the common language used in Reddit posts related to medical tourism in Turkey, we generate a word cloud based on the cleaned text. This visualization highlights the most frequently used words, excluding stopwords and irrelevant tokens.

In [50]:

```
from wordcloud import WordCloud
```

In [52]:

```
# Join all cleaned text into a single string
all_text = " ".join(posts_df['clean_text'].dropna().tolist())
```

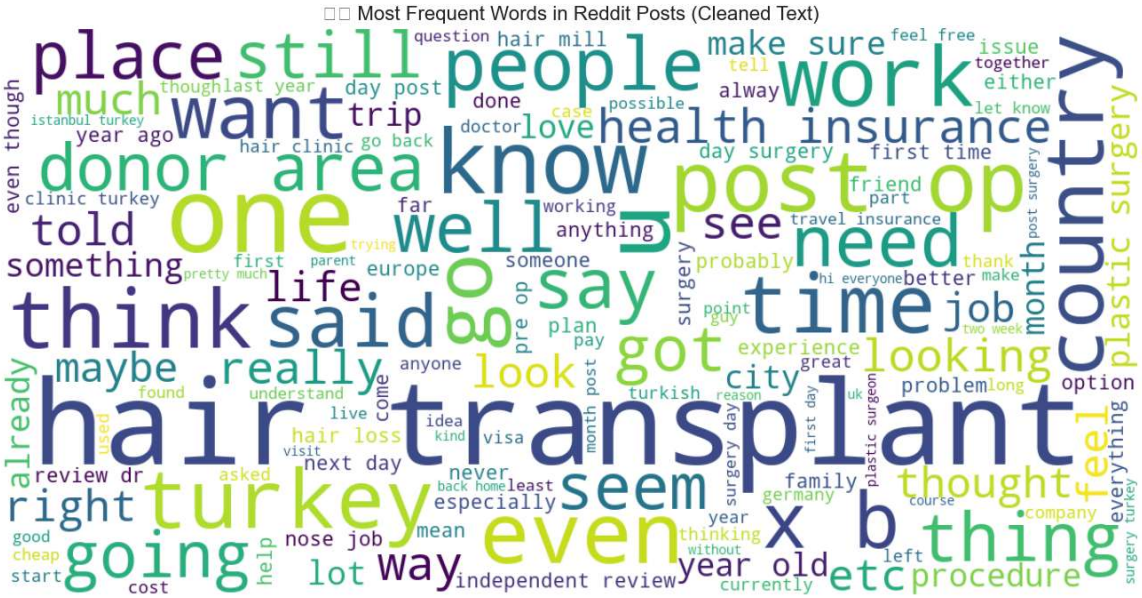
In [54]:

```
# Create a word cloud
wordcloud = WordCloud(
    width=1200,
    height=600,
    background_color='white',
    max_words=150,
    colormap='viridis',  # You can also try: 'plasma', 'magma', 'inferno', et
    contour_color='steelblue',
    contour_width=1
```

```
).generate(all_text)
```

In [56]:

```
# Display the word cloud
plt.figure(figsize=(14, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("☁️ Most Frequent Words in Reddit Posts (Cleaned Text)", fontsize=1
plt.tight_layout()
plt.show()
```




## 📊 Sentiment Analysis of Reddit Posts

Understanding how people feel about medical tourism in Turkey can provide valuable insights into public perception, satisfaction, and trust. In this section, we apply sentiment analysis to the cleaned Reddit posts using VADER (Valence Aware Dictionary and sEntiment Reasoner), a lexicon and rule-based sentiment analysis tool specifically designed for social media text.

Each post is analyzed and scored with a **compound sentiment score**, which we then categorize as:

- **Positive**: compound score ≥ 0.05
- **Negative**: compound score ≤ -0.05
- **Neutral**: otherwise

This helps us quantify the emotional tone of the discussions and visualize the overall sentiment distribution across Reddit.

Key objectives:

- Compute sentiment scores for each post
- Classify posts into positive, negative, or neutral
- Visualize the overall sentiment distribution

In [59]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

In [61]:
```
# 📥 Download VADER lexicon (works in Kaggle)
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\lofit\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

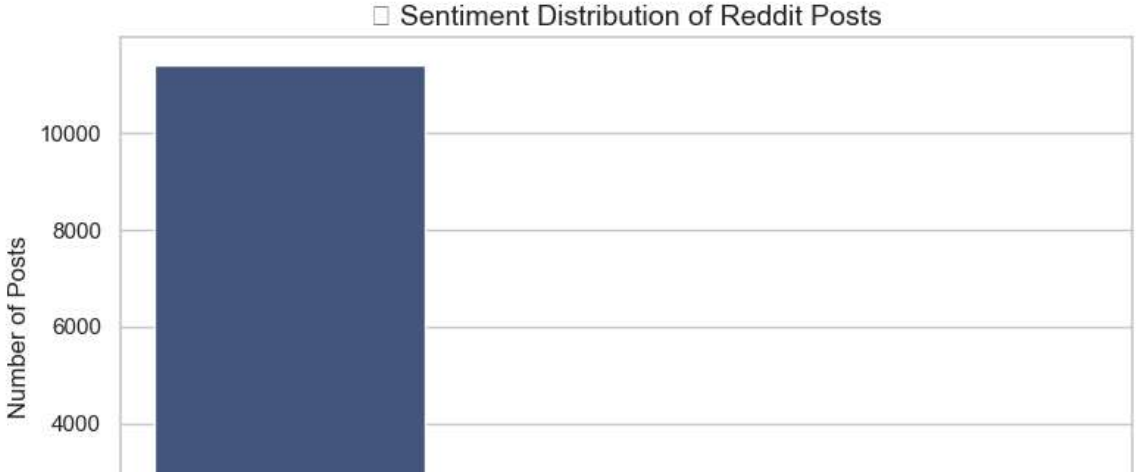
Out[61]: True

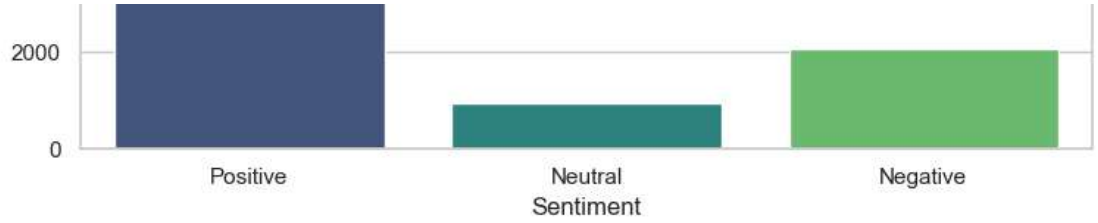
In [63]:
```
# 🔍 Initialize VADER
sia = SentimentIntensityAnalyzer()
```

In [65]:
```
# 🧠 Sentiment analysis function
def analyze_sentiment(text):
    if pd.isnull(text) or text.strip() == "":
        return pd.Series({'sentiment_compound': None, 'sentiment_label': None
    scores = sia.polarity_scores(text)
    return pd.Series({
        'sentiment_compound': scores['compound'],
        'sentiment_label': (
            'Positive' if scores['compound'] >= 0.05 else
            'Negative' if scores['compound'] <= -0.05 else
            'Neutral'
        )
    })
```

In [67]:
```
# 🪄 Apply to cleaned text
sentiment_results = posts_df['clean_text'].apply(analyze_sentiment)
posts_df = pd.concat([posts_df, sentiment_results], axis=1)
```

In [73]:
```
# 📊 Sentiment Distribution
plt.figure(figsize=(8, 5))
sns.countplot(data=posts_df, x='sentiment_label', palette='viridis', order=['
plt.title("🎭 Sentiment Distribution of Reddit Posts", fontsize=14)
plt.xlabel("Sentiment")
plt.ylabel("Number of Posts")
plt.tight_layout()
plt.show()
```



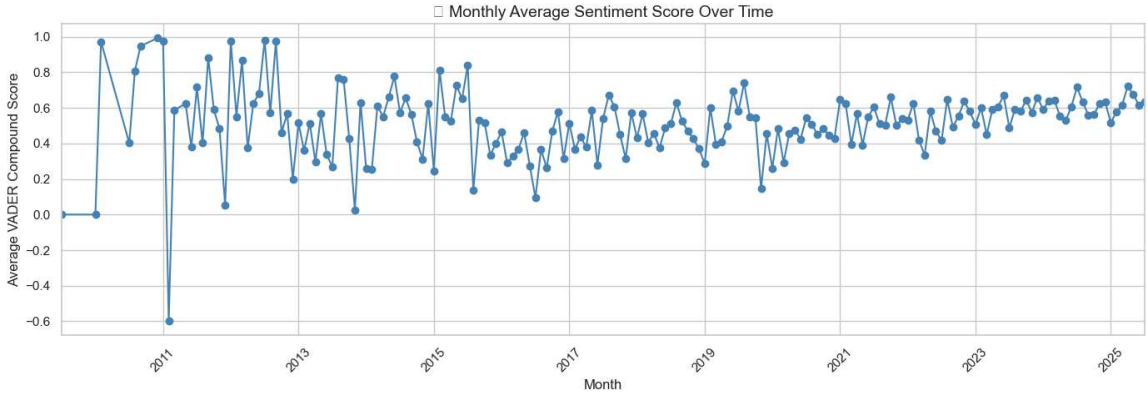



In [69]:

```
# 🕐 Time series: monthly sentiment trend
posts_df['created_utc'] = pd.to_datetime(posts_df['created_utc'])
monthly_sentiment = posts_df.groupby(posts_df['created_utc'].dt.to_period('M'
```

In [71]:

```
plt.figure(figsize=(14, 5))
monthly_sentiment.plot(marker='o', linestyle='-', color='steelblue')
plt.title("📉 Monthly Average Sentiment Score Over Time", fontsize=14)
plt.ylabel("Average VADER Compound Score")
plt.xlabel("Month")
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```




In [75]:

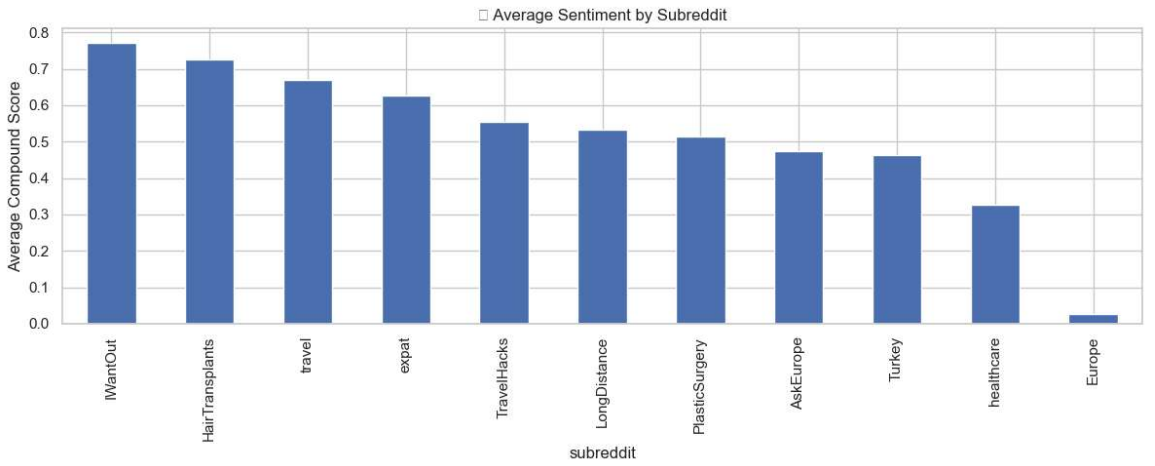
```
print("🏆 Most Positive Posts:")
print(posts_df.sort_values(by='sentiment_compound', ascending=False)[['title'

print("\n💔 Most Negative Posts:")
print(posts_df.sort_values(by='sentiment_compound', ascending=True)[['title',
```

```
🏆 Most Positive Posts:
                                                   title  \
12344  [WeWantOut] 27M US Mechanical Engineer, 27F Br...
6364   An End to End Guide to My Nose Job In Turkey (...
6406   An End to End Guide to My Nose Job In Turkey (...
4750   An End to End Guide to My Nose Job In Turkey (...
7654   An End to End Guide to My Nose Job In Turkey (...

                                              clean_text  sentiment_compound
12344  wewantout u mechanical engineer f brazil spain...             1.0000
6364   end end guide nose job turkey u cost timeline ...             0.9999
6406   end end guide nose job turkey u cost timeline ...             0.9999
4750   end end guide nose job turkey u cost timeline ...             0.9999
7654   end end guide nose job turkey u cost timeline ...             0.9999

💔 Most Negative Posts:
                                                   title  \
4096    Jeremy Salt: The Denial of the Right to Disagree
```

```
13447  Crimes against Serbs in Bosnia 1992–1995 (I) |...
4211   A Turkish Perspective regarding April 24th, an...
3820   A Turkish Perspective regarding April 24th, an...
13910  News recap, 21 March 2022 PDT 07:15, EST 10:15...

                                              clean_text  sentiment_compound
4096   jeremy salt denial right disagree fatma ge g e...             -1.0000
13447  crime serb bosnia naser ori defended podrinje ...             -0.9999
4211   turkish perspective regarding april th foreign...             -0.9998
3820   turkish perspective regarding april th foreign...             -0.9998
13910  news recap march pdt est utc eet observation g...             -0.9997
```

In [77]:

```
subreddit_sentiment = posts_df.groupby('subreddit')['sentiment_compound'].mea
subreddit_sentiment.plot(kind='bar', figsize=(12,5), title="📊 Average Sentim
plt.ylabel("Average Compound Score")
plt.tight_layout()
plt.show()
```




## 🧠 Topic Modeling with LDA (Latent Dirichlet Allocation)

To uncover the main themes discussed in Reddit posts about medical tourism in Turkey, we apply **Latent Dirichlet Allocation (LDA)**—a popular topic modeling algorithm in natural language processing.

By analyzing the frequency and co-occurrence of words in documents, LDA groups posts into topics that share similar vocabularies. This helps us:

- Identify common areas of interest (e.g., procedures, concerns, recovery experiences)
- Discover patterns or trends in discussions
- Summarize large volumes of text data into interpretable themes

### Key Steps:

- Prepare and vectorize text using bag-of-words or TF-IDF
- Train an LDA model with a selected number of topics
- Display top keywords for each topic
- (Optional) Visualize topics interactively using pyLDAvis

In [82]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
```

In [84]:
```
# 1. Create a document-term matrix using CountVectorizer
vectorizer = CountVectorizer(
    max_df=0.95,
    min_df=5,
    stop_words='english',
    lowercase=True
)
doc_term_matrix = vectorizer.fit_transform(posts_df["clean_text"])
```

In [86]:
```
# 2. Fit LDA model with defined number of topics
lda_model = LatentDirichletAllocation(
    n_components=6,       # Number of topics
    max_iter=10,
    learning_method='online',
    random_state=42
)
lda_model.fit(doc_term_matrix)
```

Out[86]:
```
LatentDirichletAllocation(learning_method='online', n_components=6,
                          random_state=42)
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**
**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [90]:
```
# 3. Function to print top keywords per topic
def print_topics(model, vectorizer, top_n=10):
    terms = vectorizer.get_feature_names_out()
    for idx, topic in enumerate(model.components_):
        print(f"\n🔹 Topic #{idx + 1}:")
        print(" ".join([terms[i] for i in topic.argsort()[-top_n:][::-1]]))
```

In [92]:
```
# Display topic keywords
print_topics(lda_model, vectorizer)
```

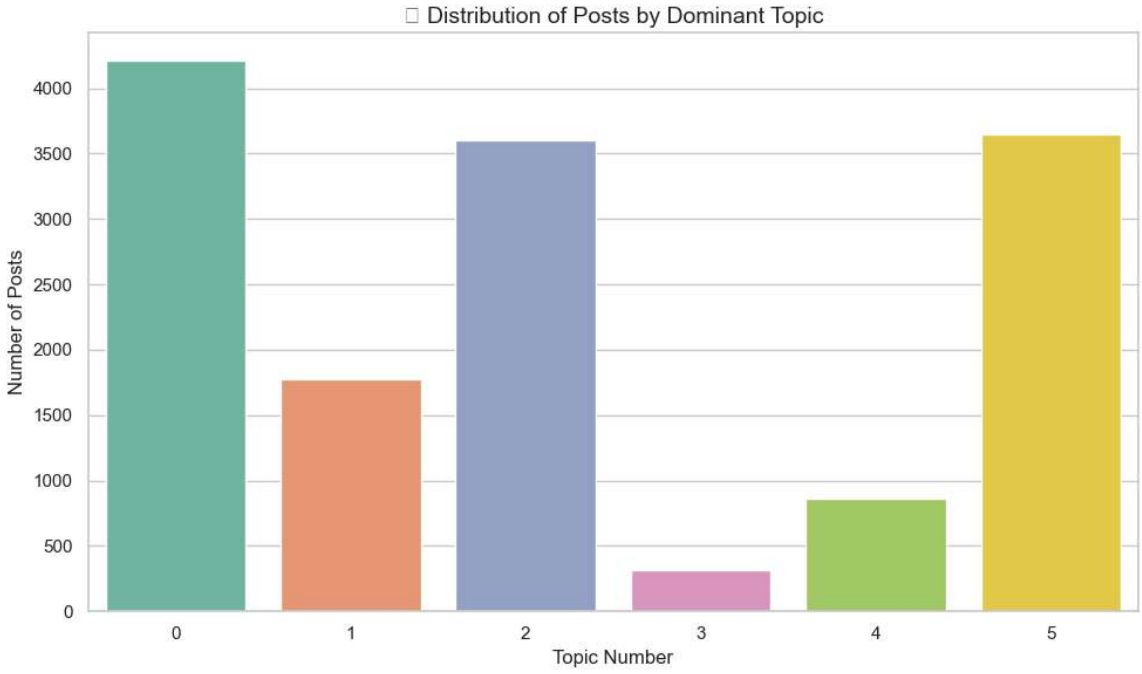
```
🔹 Topic #1:
hair clinic graft transplant turkey dr procedure day time good

🔹 Topic #2:
day hotel time like hour room night trip food week

🔹 Topic #3:
surgery nose surgeon dr turkey like rhinoplasty doctor look post

🔹 Topic #4:
dr review people russian russia list independent ukraine recent turkey

🔹 Topic #5:
travel insurance country health canada visa cost day eu uk

🔹 Topic #6:
year country turkey job like work people want know time
```

In [94]:
```
# 4. Assign dominant topic to each document
```

```
# 4. Assign dominant topic to each document
topic_values = lda_model.transform(doc_term_matrix)
posts_df["dominant_topic"] = topic_values.argmax(axis=1)
```

In [96]:

```
# 5. Visualize topic distribution
plt.figure(figsize=(10, 6))
sns.countplot(x="dominant_topic", data=posts_df, palette="Set2")
plt.title("🔍 Distribution of Posts by Dominant Topic", fontsize=14)
plt.xlabel("Topic Number")
plt.ylabel("Number of Posts")
plt.tight_layout()
plt.show()
```




In [98]:

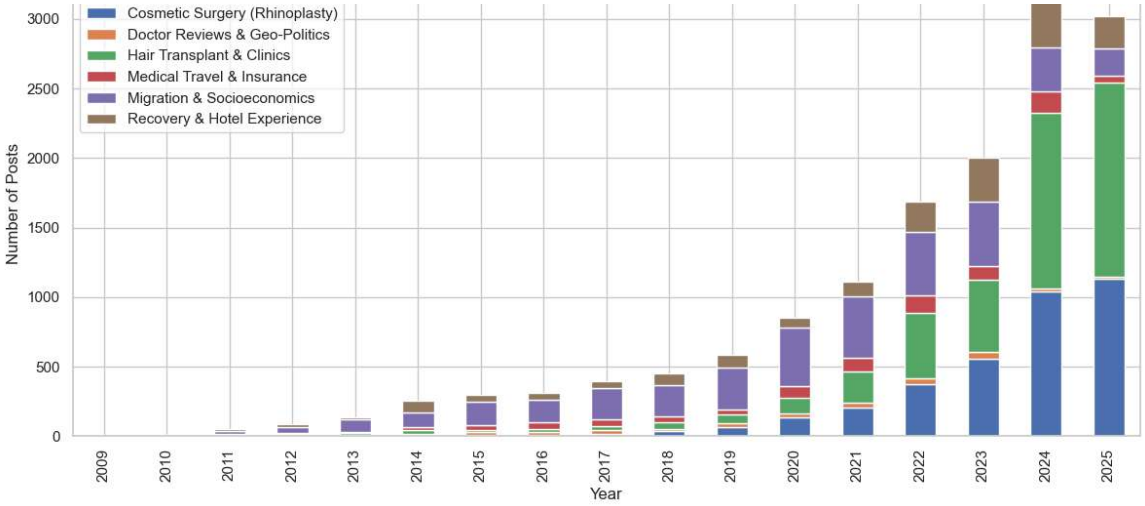
```
# 6. Optional: Assign topic labels (manually interpreted)
topic_labels = {
    0: "Hair Transplant & Clinics",
    1: "Recovery & Hotel Experience",
    2: "Cosmetic Surgery (Rhinoplasty)",
    3: "Doctor Reviews & Geo-Politics",
    4: "Medical Travel & Insurance",
    5: "Migration & Socioeconomics"
}
```

In [100…

```
# Add topic labels to the DataFrame
posts_df['topic_label'] = posts_df['dominant_topic'].map(topic_labels)
```

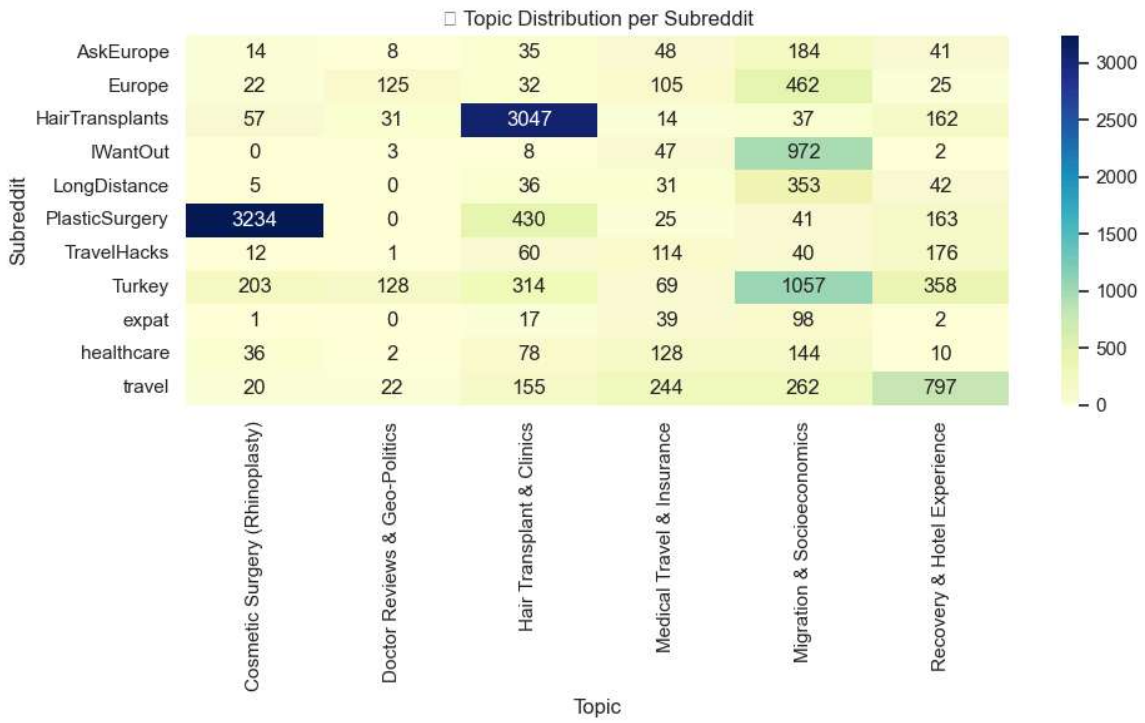
In [102…

```
posts_df['year'] = posts_df['created_utc'].dt.year
topic_by_year = posts_df.groupby(['year', 'topic_label']).size().unstack().fi
topic_by_year.plot(kind='bar', stacked=True, figsize=(12,6))
plt.title("📅 Topic Trends Over Years")
plt.ylabel("Number of Posts")
plt.xlabel("Year")
plt.tight_layout()
plt.show()
```




In [104…

```
topic_sub = pd.crosstab(posts_df['subreddit'], posts_df['topic_label'])
sns.heatmap(topic_sub, cmap="YlGnBu", annot=True, fmt="d")
plt.title("📌 Topic Distribution per Subreddit")
plt.xlabel("Topic")
plt.ylabel("Subreddit")
plt.tight_layout()
plt.show()
```




## 🔗 Topics and Sentiments

To gain deeper insights into how people feel about different topics related to medical tourism in Turkey, we combine **topic modeling** with **sentiment analysis**.

By analyzing the dominant topic of each post alongside its sentiment score and label, we can answer questions like:

- Which topics are associated with the most negative or positive emotions?
- Are certain topics more controversial or polarizing?
- How do sentiments vary across topics over time?

This combined view helps uncover not only *what* people are talking about, but also *how* they feel about it — providing a richer, more contextual understanding of public perception.

## Goals:

- Calculate average sentiment per topic
- Visualize sentiment distribution across topics
- (Optional) Track topic-sentiment trends over time

In [107…

```
# 📊 Count of sentiment labels per topic
topic_sentiment_dist = posts_df.groupby(['topic_label', 'sentiment_label']).s

# 📈 Percentage distribution of sentiments per topic
topic_sentiment_percent = topic_sentiment_dist.div(topic_sentiment_dist.sum(a

# 🔍 Identify most positive and negative topics
most_positive = topic_sentiment_percent['Positive'].idxmax()
most_negative = topic_sentiment_percent['Negative'].idxmax()

print(f"✅ Most positively discussed topic: '{most_positive}' ({topic_sentime
print(f"⚠️ Most negatively discussed topic: '{most_negative}' ({topic_sentime
```

```
✅ Most positively discussed topic: 'Recovery & Hotel Experience' (89.43% posi
tive)
⚠️ Most negatively discussed topic: 'Doctor Reviews & Geo-Politics' (25.00% ne
gative)
```

In [109…

```
# 🗓️ Add time period (monthly)
posts_df['year_month'] = posts_df['created_utc'].dt.to_period('M')
```

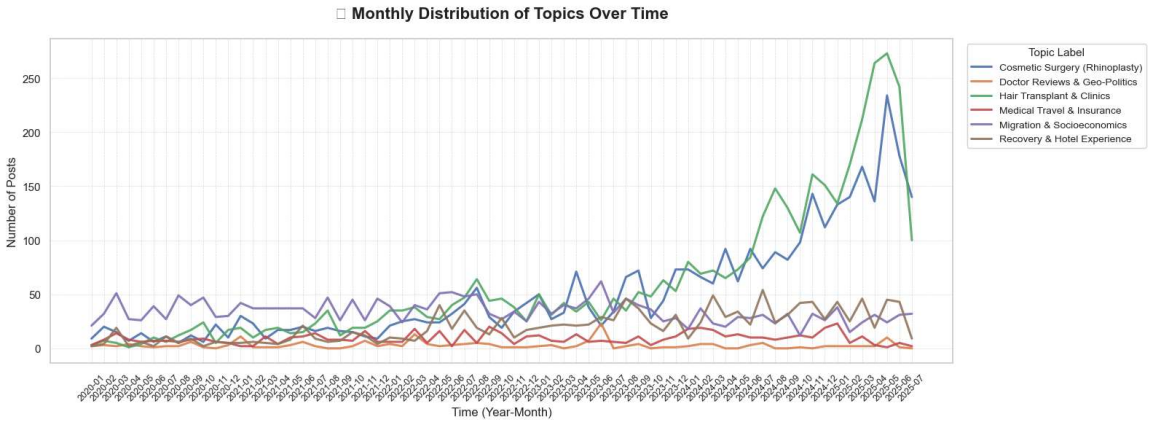
In [111…

```
# 📊 Number of posts per topic per month
topic_time = posts_df.groupby(['year_month', 'topic_label']).size().unstack()

# Convert PeriodIndex to string
topic_time.index = topic_time.index.astype(str)

# Observation
display(topic_time.tail())
```

| topic_label | Cosmetic Surgery (Rhinoplasty) | Doctor Reviews & Geo-Politics | Hair Transplant & Clinics | Medical Travel & Insurance | Migration & Socioeconomics | Recovery & Hotel Experience |
|---|---|---|---|---|---|---|
| year_month | | | | | | |
| 2025-03 | 168.0 | 2.0 | 212.0 | 11.0 | 24.0 | 46.0 |
| 2025-04 | 136.0 | 2.0 | 264.0 | 3.0 | 31.0 | 19.0 |
| 2025-05 | 234.0 | 10.0 | 273.0 | 1.0 | 24.0 | 45.0 |
| 2025-06 | 178.0 | 1.0 | 242.0 | 5.0 | 31.0 | 43.0 |
| 2025-07 | 140.0 | 0.0 | 100.0 | 2.0 | 32.0 | 9.0 |

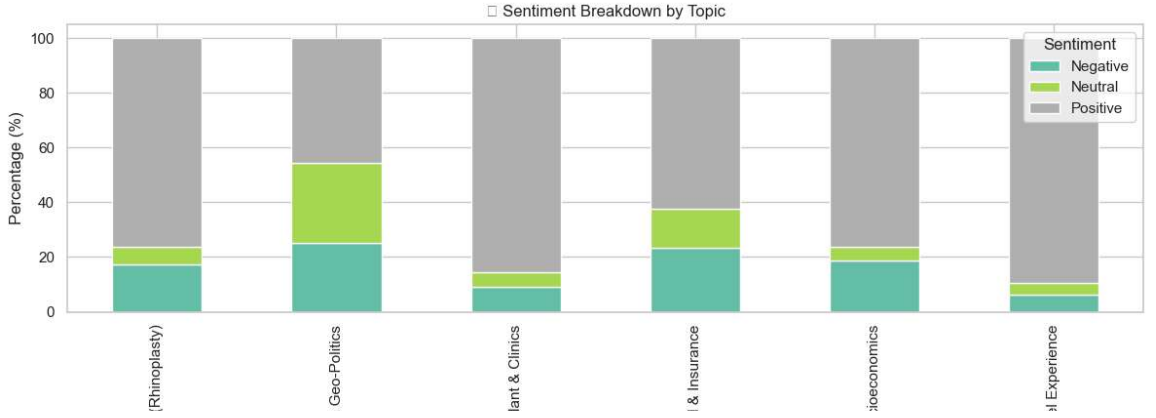
In [113…

```
# Filter to recent years (from 2020 onwards)
filtered_topic_time = topic_time[topic_time.index >= "2020-01"]

# 📈 Plot topic trends over time
plt.figure(figsize=(16, 6))
for column in filtered_topic_time.columns:
    plt.plot(
        filtered_topic_time.index,
        filtered_topic_time[column],
        label=column,
        linewidth=2.2,
        alpha=0.9
    )

plt.title("📈 Monthly Distribution of Topics Over Time", fontsize=16, weight=
plt.xlabel("Time (Year-Month)", fontsize=12)
plt.ylabel("Number of Posts", fontsize=12)
plt.xticks(rotation=45, fontsize=9)
plt.locator_params(axis="x", nbins=15)
plt.legend(title="Topic Label", title_fontsize=11, fontsize=10, bbox_to_ancho
plt.tight_layout()
plt.grid(True, linestyle='--', linewidth=0.5, alpha=0.7)
plt.show()
```



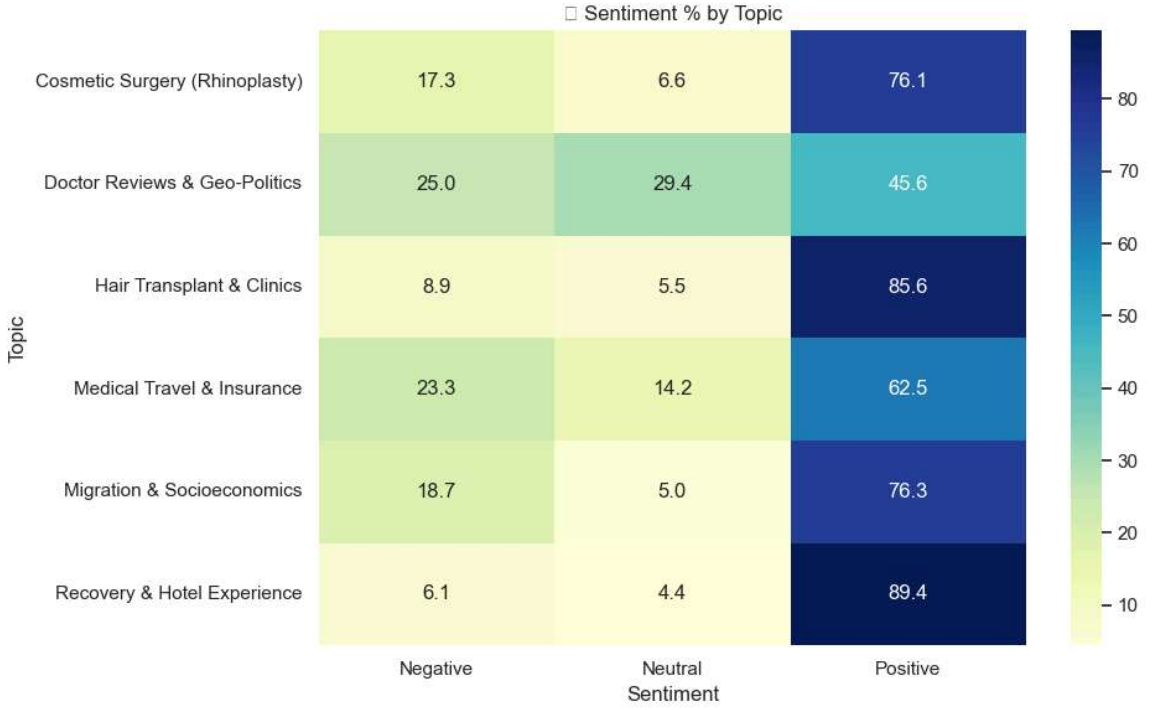

In [115…

```
topic_sentiment_percent.plot(kind='bar', stacked=True, figsize=(12,6), colorm
plt.title("📊 Sentiment Breakdown by Topic")
plt.ylabel("Percentage (%)")
plt.xlabel("Topic")
plt.legend(title="Sentiment")
plt.tight_layout()
plt.show()
```




In [117…

```
sns.heatmap(topic_sentiment_percent, annot=True, cmap="YlGnBu", fmt=".1f")
plt.title("🔆 Sentiment % by Topic")
plt.xlabel("Sentiment")
plt.ylabel("Topic")
plt.tight_layout()
plt.show()
```




## 🗺️ Geographic Analysis

In this section, we explore the geographic dimensions of medical tourism discussions on Reddit. By analyzing location references or user-reported origins, we aim to understand where most interest in Turkey's healthcare services is coming from.

This can help identify:

- Key source countries for medical tourists
- Regional patterns in interest or sentiment
- Cross-border healthcare demand trends

In [140…

```
import pycountry
from collections import Counter
import plotly.express as px
```

In [124…

```
# List of all recognized countries in lowercase
country_list = [country.name.lower() for country in pycountry.countries]
```

In [126…

```
# Function to extract country mentions from text
```

```
def extract_countries(text):
    found = []
    text = re.sub(r'[^a-zA-Z\s]', '', str(text).lower())  # Remove punctuatio
    for country in country_list:
        if country in text:
            found.append(country)
    return found
```

In [128…

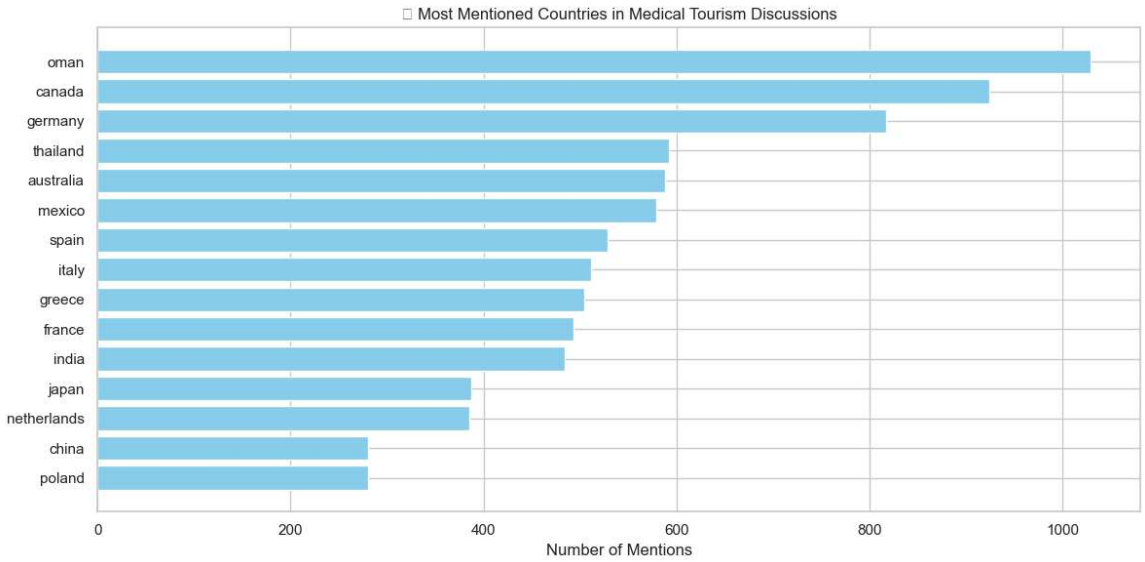
```
# Apply function to clean text column
posts_df['mentioned_countries'] = posts_df['clean_text'].apply(extract_countr
```

In [136…

```
# Flatten and count mentions
all_countries = sum(posts_df['mentioned_countries'], [])
country_counts = Counter(all_countries).most_common(15)
```

In [138…

```
# Plot top mentioned countries
if country_counts:
    countries, counts = zip(*country_counts)
    plt.figure(figsize=(12, 6))
    plt.barh(countries[::-1], counts[::-1], color='skyblue')
    plt.xlabel("Number of Mentions")
    plt.title("🌎 Most Mentioned Countries in Medical Tourism Discussions")
    plt.tight_layout()
    plt.show()
```




In [148…

```
country_counts = Counter(all_countries)
country_df = pd.DataFrame(country_counts.items(), columns=['country', 'mentio
```

In [150…

```
# Convert country names to ISO alpha-3 codes for plotting
def get_country_code(name):
    try:
        return pycountry.countries.lookup(name).alpha_3
    except:
        return None
```

In [152…

```
country_df['iso_alpha'] = country_df['country'].apply(get_country_code)
```

```
country_df = country_df.dropna(subset=['iso_alpha'])
```

In [156…

```
# Create an enhanced choropleth map
fig = px.choropleth(
    country_df,
    locations="iso_alpha",
    color="mentions",
    hover_name="country",
    color_continuous_scale=px.colors.sequential.Plasma,
    title="🌍 Most Mentioned Countries in Reddit Discussions About Medical To
    projection="natural earth",  # More natural map projection
    template="plotly_white",     # Clean background
    width=1200,
    height=700
)

# Styling: remove borders and show countries clearly
fig.update_geos(
    showcoastlines=True,
    coastlinecolor="LightGray",
    showland=True,
    landcolor="whitesmoke",
    showocean=True,
    oceancolor="aliceblue",
    showframe=False
)

# Title and layout settings
fig.update_layout(
    margin={"r":0,"t":60,"l":0,"b":0},
    title_font=dict(size=22, family="Arial", color="black"),
    coloraxis_colorbar=dict(title="Mentions", ticksuffix=""),
)

fig.show()
```

## 📘 Conclusion & Key Takeaways

This project explored the landscape of medical tourism in Turkey by analyzing thousands of Reddit discussions using a mix of natural language processing and data visualization techniques.

By combining sentiment analysis, topic modeling, and geographic insights, we were able to identify:

- 🌍 The countries most interested in Turkey's medical services
- 💬 Common themes such as hair transplants, cosmetic surgery, recovery experiences, and travel logistics
- 📈 Sentiment patterns showing which topics are viewed positively or negatively
- 🧭 Topic trends and shifts over time

These insights offer a valuable snapshot of how global communities perceive and discuss medical tourism in Turkey — from curiosity and satisfaction to skepticism and concern.

---

## 🤝 Project Team & Links

| 👤 Name | 🌐 GitHub | 💼 LinkedIn | 💉 Kaggle |
|---|---|---|---|
| **Abdullah Köse** | GitHub | LinkedIn | Kaggle |
| **Ozan Möhurcu** | GitHub | LinkedIn | Kaggle |
| **Özer Ipek** | GitHub | LinkedIn | Kaggle |

---

> 📣 If you found this notebook useful or interesting, feel free to leave an upvote, fork it, or reach out!