```
                # CODE FOR KMEANS
import pandas as pd
from sklearn preprocessing import Standard Scaler
from sklearn.model_selection import KMeans
import numpy as np

csv_path=csv_path = "/content/country_misinfo_vax_percent_clean (2).csv" # same columns: country, misinfo_percent, vax_percent
k = 3
df = pd.read_csv(csv_path)
df.head()
X=df[["misinfo_percent","vax_percent"]].astype(float)
X_scaled=StandardScaler().fit_transform(X)
lables=KMeans(n_clusters=k).fit_predict(X_scaled)
df["cluster"]=lables
plt.figure(figsize=(12,8))
sc=plt.scatter(df["misinfo_percent"],df["vax_percent"]
c=df["cluster"],cmap='viridis')
for r in df.iterrows():
plt.text(r[1]["misinfo_percent"]+0.2,r["misinfo_percent"]=0.2,r["country"],fontsize=8)
plt.xlabel("Vaccination %")
plt.ylabel("Misinformation %")
plt.title("Vaccination vs Misinformation by Country")
plt.colorbar(sc)
plt.show()
```

```
                # SPATIAL MAPPING AND MISINFORMATION DETECTION

    import pandas as pd
    import ploly.express as px
    from scipy.stats import pearsonr
    # step 1 load the all tweet files
    tweet_files=[
        "/content/Tweets_part1.csv",
        "/content/Tweets_part2.csv",
        "/content/Tweets_part3.csv",
        "/content/Tweets_part4.csv",
        "/content/Tweets_part5.csv",
        ]

    tweets_parts=[]
    for f in tweet_files:
      try:
        df=pd.read_csv(f,engine='python',quotechar='"',doublequote=True,on_bad_lines"skip')
        tweet_parts.append(df)
      tweets=pd.concat(tweets_parts,ignore_index=True
    #Step 2 load the vaccination data
    vax=pd.read_csv('/content/vaccinations.csv')
    vax_latest=(
        vax.sort_values(['location', 'date'])
           .groupby('location', as_index=False)
           .last()[['location','people_vaccinated_per_hundred']]
           .rename(columns={'location':'country',
                            'people_vaccinated_per_hundred':'vax_percent'})
    )
    all_countries = vax_latest['country'].unique()

    #step 3 matching the tweet
    # STEP 3: match tweet user location to countries
    def match_country(loc):
        if pd.isna(loc):
            return None
        loc = str(loc).lower()
        for c in all_countries:
            if c.lower() in loc:
                return c
        return None
    tweets['country']=tweets['user_location'].apply(match_country)
    tweets=tweets[tweets['country'].notnull()]
    #step4 flag misinformation tweets
    misinfo_words=[]'microchip','hoax','plandemic','5g','fake vaccine','dna change','magnet','tracking device']

    def is_misinfo(txt):
        if pd.isna(txt):
            return False
        txt = txt.lower()
        return any(w in txt for w in misinfo_words)

    tweets['is_misinformation'] = tweets['text'].apply(is_misinfo)
    #step 5 calulate misinformation % per country
    Summary = (tweets.groupby('country')
                   .agg(num_misinfo=('is_misinformation','sum'),
                        total_tweets=('text','count'))
                   .reset_index())
    summary['misinfo_percent'] = (summary['num_misinfo'] / summary['total_tweets']) * 100
    #step 6 merge with vaccination data
    final=pd.merge(vax_latest,summmary[['country','misinfo_percent']].on='country',how='left')
    final['misinfo_percent']=final['misinfo-percent'].fillna(0)
    #step 7 choropleth map
    fig_map = px.choropleth(
        final,
        locations='country',
        locationmode='country names',
        color='misinfo_percent',
        hover_name='country',
        hover_data={'misinfo_percent':':.2f','vax_percent':':.2f'},
        color_continuous_scale=['#f7fcf5','#e5f5e0','#c7e9c0','#a1d99b',
                                '#74c476','#41ab5d','#238b45','#006d2c','#00441b'],
        range_color=(0,100),
        title='Percentage of Misinformation Tweets by Country'
    )
    fig_map.update_layout(
```

```
        geo=dict(showframe=False, showcoastlines=True, projection_type='equirectangular', bgcolor='white'),
        coloraxis_colorbar=dict(title="Misinformation (%)", tickformat=".0f"),
        font=dict(size=14)
    )
    fig_map.show()

    # STEP 8: scatterplot
    fig_scatter = px.scatter(
        final,
        x='misinfo_percent',
        y='vax_percent',
        text='country',
        trendline='ols',
        title="Country-level % Misinformation Tweets vs. % Vaccinated",
        labels={'misinfo_percent':'% Misinformation Tweets',
                'vax_percent':'% Vaccinated'},
        color='misinfo_percent',
        color_continuous_scale='Greens'
    )
    fig_scatter.update_traces)textposition='top center',marker=dict(size=12,line=dict(width=1,color='DarkSlateGrey'))
    fig_scatter.update_layout(template='plotly_white',width=900,height=550,
                              font=dict(size=15),margin=dict(l=60,r=20,t=70,b=60),
                              coloraxis_colorbar=dict(title='Misinformation(%)'))

    fig_scatter.show()
    #step 9 pearson corelation
    corr, pval = pearsonr(final['misinfo_percent'], final['vax_percent'])

    # step 10 saving reults into adn .csv file
    final[['country','misinfo_percent,"vax_percent']].to_csv('/content/country_misinfo_vax_percent_clean.csv', index=False)

    )
```

```
                # CODE FOR VISUALISATION AND PEARSON CORELATION #
import pandas as pd#
import matplotlib.pyplot as plt
import seaborn as sns

#loading the data set
df=pd.read_csv("/content/vaccination_and_sentiment_analysis_ireland.csv")


#calculate 7-day moving averages to smooth out daily fluctuations
df['vaccinated_7d']=df['daily_vaccinated'].rolling(window=7).mean()
df['positive_7d']=df['daily_positive'].rolling(window=7).mean()
df['negative_7d']=df['negative'].rolling(window=7).mean()

# plotting vaccinations vs sentimnet over time

sns.set(style="whitegrid")

fig,ax1=plt.subplots(figsize=(16,6))

#Left Y-Axis daily vaccinations (bars)
ax1.bar(df['date'],df['vaccinated_7d'],color='orange',alpha=0.6,label='vaccinated (7-day Avg)')
ax1.set_ylabel("Daily Vaccinated",color='orange',fontsize=12)
ax1.tick_params(axis='y',labelcolor='orange')
ax1.tick_xlabel("Date",fontsize=12)
ax1.tick_params(axis='x',rotation=45)

#Right Y-axis sentiment trends
ax2=ax1.twinx()
ax2.plot(df['date'],df['positive_7d'],label='positive (7-day Avg)',color='green',linewidth=2
ax2.plot(df['date'],df['negative_7d'],label='negative (7-day Avg)',color='gray'.linewidth=2)
ax2.plot(df['date'],df['neutral_7d'],label='neutral (7-day Avg)',color='red',linewidth=2)
ax2.set_ylabel("Sentiment",color='black',fontsize=12)
ax2.tick_params(axis='y',labelcolor='black')

#title and styling
plt.title("COVID-19 Vaccination vs Twitter Sentiment(Ireland)"),
         fontsize=14,weight='bold')
ax1.grid(True,linestyle='--',alpha=0.5)

# combining
lines1.labels1=ax1.get_legend_handles_lables()
lines2,labelss2=ax2.get_legend_handles_lables()
ax1.legend(lines1+lines2,labels1+labels2,loc='upper left')
plt.tight_layout()
plt.show()
```

```
# CODE FOR THE VACCINATIONS AND SENTIMENT ANALYSIS
import pandas as pd
# here we need to load our data sets

tweet_files=[
    "/content/Tweets_part1.csv",
    "/content/Tweets_part2.csv",
    "/content/Tweets_part3.csv",
    "/content/Tweets_part4.csv",
    "/content/Tweets_part5.csv",
]

# in this step we need to load all files and merge
tweets_list=[]
for fp in tweets_files:
  df=pd.read_csv(fp)
  tweets_list.append(df)

tweets_df=pd.concat(tweets_list)

# in this we need to load the vaccintion data
vaccination_df=pd.read_csv("/content/vaccination_data.csv")

tweets_df["created_at"]=pd.to_datetime(tweets["created_at"])
tweets_df=tweets_df.dropna(subset=["created_at"]).copy()
tweets_df["date"]=tweets_df["created_at"].dt.date

tweets_df["polarity"]=pd.to_numeric(tweets["polarity"])
tweets_df=tweets.dropna(subset=["polarity "]).copy()

# vacciations
vaccination_df["date"]=pd.to_datetime(vaccination_df["date"])
vaccination_df=vaccination_df.dropna(subset=["date"]).copy()
vaccintion_df["date"]=vaccination["date"].dt.date

# user inputs

print("Enter details for the analysis:")
country=input("Countrt name  : ").strip()
start_date=pd.to_datetime(input("Start date (YYYY-MM-DD) : ").strip())
end_date=pd.to_datetime(input("End date (YYYY-MM-DD) : ").strip().date()

# filter data
vaccination_sel=vaccination_df[(vaccination_df["location"]==country) &
              (vaccination_df["date"]>=start_date) &
              (vaccination_df["date"]<=end_date)].copy()

tweets_sel=tweets_df
 [(tweets["date"]>=start_date)
&(tweets["date"]<=end_date)
].copy()

if vaccination_sel.empty:
            raise RuntimeError(f"No vaccaintion data for '{country}' in that date range.")
if tweets_sel.empty:
            raise RuntimeError(f"No tweets data for that date range.")

#
rows=[]
for d in pd.date_range(start_date,end_date):
  day=d.date()

# Tweets for this day
tday=tweets_sel[tweets_sel["date"]==day]
pos=(tday["polarity"]>0).sum()
neu=(tday["polarity"]==0).sum()
neg=(tday["polarity"]<0).sum()
total=pos+neu+neg

# vaccinations for this day
vday=vaccination_sel[vaccination_sel["date"]==day]
daily_vaccinated=int(vday["daily_vaccinations"].sum() if not v day.empty else 0

rows.append({
    "date":day,
    "positive":round((pos /total)*100,2) if total else 0.0,
```

```
        "neutral":round((neu /total))*100,2) if total else 0.0,
        "negative":round((neg/total)) *100,2) if total else 0.0,
        "daily_vaccinated": daily-vaccinated
        })
    result=pd.DataFrame(rows)

    # save the file
    outfile = f"/content/vaccination_and_sentiment_{safe_country}_{start_date}_to_{end_date}.csv"
    print(f"Saving to {outfile}...")
```