```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from prophet import Prophet
import geopandas as gpd
import plotly.express as px

# Load dataset
df = pd.read_json("cleaned_data.json")

### 📊 USER ENGAGEMENT ANALYSIS ###
df["engagement_score"] = df["score"] + (df["num_comments"] * 2)  # Weight comments more

plt.figure(figsize=(10, 5))
sns.histplot(df["engagement_score"], bins=50, kde=True)
plt.title("User Engagement Score Distribution")
plt.xlabel("Engagement Score")
plt.ylabel("Frequency")
plt.show()

### 🌍 LOCATION ANALYSIS ###
df["location"] = df["location"].fillna("Unknown")  # Handle missing locations

# Count posts by location
location_counts = df["location"].value_counts().reset_index()
location_counts.columns = ["Location", "Count"]

fig = px.bar(location_counts, x="Location", y="Count", title="Post Count by Location")
fig.show()

# If you want to visualize location data on a map:
```

```
# Ensure you have a dataset that maps locations to lat/lon

# Example: Indian states map (Uncomment below if applicable)

# india_map = gpd.read_file("india_states.geojson")  # Load India map GeoJSON

# india_map = india_map.merge(location_counts, left_on="state_name", right_on="Location", how="left")

# india_map.plot(column="Count", cmap="Blues", legend=True)

# plt.title("Engagement by State")

# plt.show()


### 📈 TREND ANALYSIS ###

df["created_utc"] = pd.to_datetime(df["created_utc"])  # Convert to datetime


# Group by date and aggregate scores to analyze trends

time_series = df.groupby(df["created_utc"].dt.date)["score"].sum().reset_index()

time_series.columns = ["ds", "y"]


# Apply Prophet for trend forecasting

model = Prophet()

model.fit(time_series)

future = model.make_future_dataframe(periods=30)  # Predict for next 30 days

forecast = model.predict(future)


fig = model.plot(forecast)

plt.title("Trend Analysis of Post Scores Over Time")

plt.show()




### 🌍 CATEGORY VS LOCATION ANALYSIS ###
```

```
df["location"] = df["location"].fillna("Unknown")  # Handle missing locations
df["category"] = df["category"].fillna("Uncategorized")  # Handle missing categories

# Count posts by category and location
category_location_counts = df.groupby(["location", "category"]).size().reset_index(name="Count")

# Plot category-wise distribution across locations
fig = px.bar(
    category_location_counts,
    x="location",
    y="Count",
    color="category",
    title="Category Distribution by Location",
    text="Count",
    barmode="stack"
)

fig.update_traces(textposition="outside")
fig.show()
```

📈 CATEGORY-WISE TREND ANALYSIS

```
df["created_utc"] = pd.to_datetime(df["created_utc"])  # Convert to datetime
df["engagement_score"] = df["score"] + (df["num_comments"] * 2)  # Calculate engagement score

# Aggregate engagement scores by date and category
category_trend = df.groupby([df["created_utc"].dt.date,
"category"])["engagement_score"].sum().reset_index()
```

```
category_trend.columns = ["Date", "Category", "Engagement_Score"]

# Plot category-wise engagement trends
fig = px.line(
    category_trend,
    x="Date",
    y="Engagement_Score",
    color="Category",
    title="Category-Wise Engagement Trends Over Time"
)

fig.show()
```