Experiment: Develop Emerging Analytics programs based on social and mobile applications.

Aim:
To analyze and compare the performance of social media posts (Tweets) and blogs by simulating realistic datasets and calculating key performance metrics such as Engagement Rate, Amplification Rate, Ripple Index, and Author Contribution.

Objectives
To simulate 100 tweet and blog records using synthetic data.
To calculate the following metrics:
Tweets: Engagement Rate, Amplification Rate, Conversation Rate
Blogs: Engagement Rate, Ripple Index, Author Contribution, Author Efficiency
To group metrics by source (e.g., Mobile, Web, RSS, etc.) and observe trends.
To compare the average performance of Tweets vs. Blogs.

Pseudocode (for Lab Observation Book)

```
import pandas as pd
import numpy as np

np.random.seed(42)

# Simulate 100 Tweets
tweets = pd.DataFrame({
  "Tweet_ID": np.arange(1, 101),
  "Likes": np.random.randint(10000, 100000, 100),
  "Retweets": np.random.randint(5000, 50000, 100),
  "Replies": np.random.randint(2000, 20000, 100),
  "Follower_Count": np.random.randint(100000, 300000, 100),
  "Traffic_Source": np.random.choice(["Mobile", "Web", "3rd party App"], 100, p=[0.6, 0.3,
0.1])
})

tweets["Engagement_Rate"] = ((tweets["Likes"] + tweets["Retweets"] + tweets["Replies"]) /
tweets["Follower_Count"]) * 100
tweets["Amplification_Rate"] = (tweets["Retweets"] / tweets["Follower_Count"]) * 100
```

```
tweets["Conversation_Rate"] = (tweets["Replies"] / tweets["Follower_Count"]) * 100

tweet_by_source = tweets.groupby("Traffic_Source")[["Engagement_Rate",
"Amplification_Rate", "Conversation_Rate"]].mean().reset_index()

# Simulate 100 Blogs
blogs = pd.DataFrame({
   "Blog_ID": np.arange(1, 101),
   "Views": np.random.randint(10000, 100000, 100),
   "Comments": np.random.randint(100, 1000, 100),
   "Shares": np.random.randint(500, 5000, 100),
   "Traffic_Source": np.random.choice(["RSS", "Direct", "Search", "Social"], 100, p=[0.25,
0.25, 0.3, 0.2])
})

total_word_counts = np.random.randint(800, 2000, 100)
author_word_counts = [np.random.randint(300, total + 1) for total in total_word_counts]
blogs["Total_Word_Count"] = total_word_counts
blogs["Author_Word_Count"] = author_word_counts

blogs["Engagement_Rate"] = ((blogs["Comments"] + blogs["Shares"]) / blogs["Views"]) *
100
blogs["Ripple_Index"] = blogs["Shares"] / blogs["Comments"]
blogs["Author_Contribution"] = (blogs["Author_Word_Count"] /
blogs["Total_Word_Count"]) * 100
blogs["Author_Efficiency"] = blogs["Engagement_Rate"] * (blogs["Author_Contribution"] /
100)

blog_by_source = blogs.groupby("Traffic_Source")[["Engagement_Rate", "Ripple_Index",
"Author_Contribution"]].mean().reset_index()

tweet_avg = tweets[["Engagement_Rate", "Amplification_Rate",
"Conversation_Rate"]].mean().rename(lambda x: "Tweet_" + x)
blog_avg = blogs[["Engagement_Rate", "Ripple_Index",
"Author_Contribution"]].mean().rename(lambda x: "Blog_" + x)
comparison = pd.concat([tweet_avg, blog_avg], axis=0).round(2)
```

Full program

```
import pandas as pd
import numpy as np

np.random.seed(42)

# ----------------------------
# Simulate 100 Tweets
# ----------------------------
tweets = pd.DataFrame({
    "Tweet_ID": np.arange(1, 101),
    "Likes": np.random.randint(10000, 100000, 100),
    "Retweets": np.random.randint(5000, 50000, 100),
    "Replies": np.random.randint(2000, 20000, 100),
    "Follower_Count": np.random.randint(100000, 300000, 100),
    "Traffic_Source": np.random.choice(["Mobile", "Web", "3rd party App"], 100, p=[0.6, 0.3,
0.1])
})

# Correct Tweet Metrics (as percentages)
tweets["Engagement_Rate"] = ((tweets["Likes"] + tweets["Retweets"] + tweets["Replies"]) /
tweets["Follower_Count"]) * 100
tweets["Amplification_Rate"] = (tweets["Retweets"] / tweets["Follower_Count"]) * 100
tweets["Conversation_Rate"] = (tweets["Replies"] / tweets["Follower_Count"]) * 100

# Avg Tweet Metrics by Source
tweet_by_source = tweets.groupby("Traffic_Source")[["Engagement_Rate",
"Amplification_Rate", "Conversation_Rate"]].mean().reset_index()

# ----------------------------
# Simulate 100 Blogs
# ----------------------------
blogs = pd.DataFrame({
    "Blog_ID": np.arange(1, 101),
    "Views": np.random.randint(10000, 100000, 100),
    "Comments": np.random.randint(100, 1000, 100),
    "Shares": np.random.randint(500, 5000, 100),
    "Traffic_Source": np.random.choice(["RSS", "Direct", "Search", "Social"], 100, p=[0.25, 
0.25, 0.3, 0.2])
})

# Ensure realistic Author Contribution
total_word_counts = np.random.randint(800, 2000, 100)
author_word_counts = [np.random.randint(300, total + 1) for total in total_word_counts]
blogs["Total_Word_Count"] = total_word_counts
blogs["Author_Word_Count"] = author_word_counts

# Correct Blog Metrics
blogs["Engagement_Rate"] = ((blogs["Comments"] + blogs["Shares"]) / blogs["Views"]) * 
100
blogs["Ripple_Index"] = blogs["Shares"] / blogs["Comments"]
blogs["Author_Contribution"] = (blogs["Author_Word_Count"] / 
blogs["Total_Word_Count"]) * 100
blogs["Author_Efficiency"] = blogs["Engagement_Rate"] * (blogs["Author_Contribution"] / 
100)

# Avg Blog Metrics by Source
blog_by_source = blogs.groupby("Traffic_Source")[["Engagement_Rate", "Ripple_Index", 
"Author_Contribution"]].mean().reset_index()

# ----------------------------
# Final Comparison
# ----------------------------
tweet_avg = tweets[["Engagement_Rate", "Amplification_Rate", 
"Conversation_Rate"]].mean().rename(lambda x: "Tweet_" + x)
blog_avg = blogs[["Engagement_Rate", "Ripple_Index", 
"Author_Contribution"]].mean().rename(lambda x: "Blog_" + x)
comparison = pd.concat([tweet_avg, blog_avg], axis=0).round(2)

# ----------------------------
# Display All Outputs
# ----------------------------
print("\nAverage Metrics (Tweets vs Blogs) — All Rates in %:")
for metric, value in comparison.items():
    if "Ripple_Index" in metric:
        print(f"{metric}: {value}")
    else:
        print(f"{metric}: {value}%")

print("\nAverage Tweet Metrics by Source (in %):")
print(tweet_by_source.round(2).to_string(index=False))

print("\nAverage Blog Metrics by Traffic Source (in % and ratio):")
blog_by_source["Engagement_Rate"] = blog_by_source["Engagement_Rate"].round(2)
blog_by_source["Author_Contribution"] = blog_by_source["Author_Contribution"].round(2)
print(blog_by_source)
```