D. Saul Jameson

https://www.linkedin.com/in/dsauljameson/

```
import pandas as pd
import numpy as np
from google.cloud import bigquery
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk

# Ensure necessary nltk resources are available
nltk.download('vader_lexicon')

# Initialize sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Ensure we have a valid BigQuery client
client = bigquery.Client()

# Define dataset and table names
dataset_id = "news_dataset"  # Replace with your actual dataset ID
original_table_id = "news_headlines"  # Original table with headlines
sentiment_table_id = "headlines_with_sentiment"  # New table for sentiment data

# Query to retrieve headlines from the original table
query = """
SELECT
    DISTINCT(url),
    title,
    source,
    description,
    published_at,
    category,
    sentiment
FROM `lofty-fragment-454812-h4.news_dataset.news_headlines`
WHERE DATE(published_at) BETWEEN DATE_SUB(CURRENT_DATE(), INTERVAL 8 DAY) AND CURRENT_DATE()
"""

# Load the original headlines into a DataFrame
df_original = client.query(query).to_dataframe()

# Function to get sentiment
def get_sentiment(text):
    """Analyze sentiment of a given text."""
    if not isinstance(text, str) or text.strip() == "":
        return "neutral"  # Default to neutral for empty or invalid text

    scores = sia.polarity_scores(text)
    compound = scores['compound']

    # Define thresholds
    if compound >= 0.05:
        return "positive"
    elif compound <= -0.05:
        return "negative"
    else:
        return "neutral"

# Apply sentiment analysis
df_original["sentiment"] = df_original["title"].apply(get_sentiment)

print(df_original.dtypes)

# Ensure 'published_at' is a proper timestamp
df_original["published_at"] = pd.to_datetime(df_original["published_at"], errors="coerce")

# Ensure 'category' remains a STRING (as required by BigQuery)
df_original["category"] = df_original["category"].astype(str)

# Convert all other columns to STRING except 'published_at'
for col in df_original.columns:
    if col != "published_at":
        df_original[col] = df_original[col].astype(str)

# Remove duplicates based on URL
df_original.drop_duplicates(subset=["url"], keep="last", inplace=True)
```

```
df_original.drop_duplicates(subset=['url'], keep='last', inplace=True)

# Define BigQuery table reference
table_ref = client.dataset(dataset_id).table(sentiment_table_id)

# Load job configuration (append new rows)
job_config = bigquery.LoadJobConfig(write_disposition="WRITE_APPEND")

# Push data to BigQuery
job = client.load_table_from_dataframe(df_original, table_ref, job_config=job_config)
job.result()

print("✅ Upload successful: Data appended to `headlines_with_sentiment`.")
```