**Python Code for performing Sentiment Analysis**

# pip install pandas nltk pyodbc sqlalchemy

import pandas as pd

import pyodbc

import nltk

from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Download the VADER lexicon for sentiment analysis if not already present.

nltk.download('vader\_lexicon')

# Define a function to fetch data from a SQL database using a SQL query

def fetch\_data\_from\_sql():

# Define the connection string with parameters for the database connection

conn\_str = (

"Driver={SQL Server};" # Specify the driver for SQL Server

"Server=ALI-LT2024\\SQLEXPRESS;" # Specify your SQL Server instance

"Database=PortfolioProject\_MarketingAnalytics;" # Specify the database name

"Trusted\_Connection=yes;" # Use Windows Authentication for the connection

)

# Establish the connection to the database

conn = pyodbc.connect(conn\_str)

# Define the SQL query to fetch customer reviews data

query = "SELECT ReviewID, CustomerID, ProductID, ReviewDate, Rating, ReviewText FROM fact\_customer\_reviews"

# Execute the query and fetch the data into a DataFrame

df = pd.read\_sql(query, conn)

# Close the connection to free up resources

conn.close()

# Return the fetched data as a DataFrame

return df

# Fetch the customer reviews data from the SQL database

customer\_reviews\_df = fetch\_data\_from\_sql()

# Initialize the VADER sentiment intensity analyzer for analyzing the sentiment of text data

sia = SentimentIntensityAnalyzer()

# Define a function to calculate sentiment scores using VADER

def calculate\_sentiment(review):

# Get the sentiment scores for the review text

sentiment = sia.polarity\_scores(review)

# Return the compound score, which is a normalized score between -1 (most negative) and 1 (most positive)

return sentiment['compound']

# Define a function to categorize sentiment using both the sentiment score and the review rating

def categorize\_sentiment(score, rating):

# Use both the text sentiment score and the numerical rating to determine sentiment category

if score > 0.05: # Positive sentiment score

if rating >= 4:

return 'Positive' # High rating and positive sentiment

elif rating == 3:

return 'Mixed Positive' # Neutral rating but positive sentiment

else:

return 'Mixed Negative' # Low rating but positive sentiment

elif score < -0.05: # Negative sentiment score

if rating <= 2:

return 'Negative' # Low rating and negative sentiment

elif rating == 3:

return 'Mixed Negative' # Neutral rating but negative sentiment

else:

return 'Mixed Positive' # High rating but negative sentiment

else: # Neutral sentiment score

if rating >= 4:

return 'Positive' # High rating with neutral sentiment

elif rating <= 2:

return 'Negative' # Low rating with neutral sentiment

else:

return 'Neutral' # Neutral rating and neutral sentiment

# Define a function to bucket sentiment scores into text ranges

def sentiment\_bucket(score):

if score >= 0.5:

return '0.5 to 1.0' # Strongly positive sentiment

elif 0.0 <= score < 0.5:

return '0.0 to 0.49' # Mildly positive sentiment

elif -0.5 <= score < 0.0:

return '-0.49 to 0.0' # Mildly negative sentiment

else:

return '-1.0 to -0.5' # Strongly negative sentiment

# Apply sentiment analysis to calculate sentiment scores for each review

customer\_reviews\_df['SentimentScore'] = customer\_reviews\_df['ReviewText'].apply(calculate\_sentiment)

# Apply sentiment categorization using both text and rating

customer\_reviews\_df['SentimentCategory'] = customer\_reviews\_df.apply(

lambda row: categorize\_sentiment(row['SentimentScore'], row['Rating']), axis=1)

# Apply sentiment bucketing to categorize scores into defined ranges

customer\_reviews\_df['SentimentBucket'] = customer\_reviews\_df['SentimentScore'].apply(sentiment\_bucket)

# Display the first few rows of the DataFrame with sentiment scores, categories, and buckets

print(customer\_reviews\_df.head())

# Save the DataFrame with sentiment scores, categories, and buckets to a new CSV file

customer\_reviews\_df.to\_csv('fact\_customer\_reviews\_with\_sentiment.csv', index=False)