A logo with text on it

Description automatically generated

**ANL503**

**DATA WRANGLING**

**END-OF-COURSE ASSIGNMENT**

**JAN 2024 PRESENTATION**

**Submission date: 13 Apr 2024**

Name: Sally Marcellina Yeo   
PI No.: Y2410435

# Q1a

A screenshot of a computer

Description automatically generated

import os # to set working directory

import re # for regex

import pandas as pd

import pymysql

from sqlalchemy import create\_engine

from sqlalchemy.engine.url import URL

# Define regex for parsing

INIT\_regex = re.compile(r'^WEBVTT\s\*$')

SNo\_regex = re.compile(r'^(\d+)\s\*$')

Time\_regex = re.compile(r'^(\d{2}:\d{2}:\d{2}\.\d{3}) --> (\d{2}:\d{2}:\d{2}\.\d{3})\s\*$')

NameUtt\_regex = re.compile(r'^([^:]+):\s?(.\*)$')

# Set your working directory

os.chdir('/Users/sallyyeo/Desktop/ANL503ECA')

# Read the VTT file, with handle file closure

vttpath = 'captured\_dialogue.vtt'

with open(vttpath) as vtt:

all\_lines = vtt.readlines()

# Initialize lists to store extracted values

sno\_list, time\_from\_list, time\_to\_list, reg\_name\_list, utterance\_list = [], [], [], [], []

# Initialize an unknown speaker

unknown\_speaker = None

lookingFor = 'INIT'

for i, current\_line in enumerate(all\_lines):

if lookingFor == 'INIT':

mo = INIT\_regex.search(current\_line)

if mo:

lookingFor = 'SNo'

continue

elif lookingFor == 'SNo':

mo = SNo\_regex.search(current\_line)

if mo:

sno = int(mo.group(1)) # Retrieve only the seq number

lookingFor = 'TimeFrom'

continue

elif lookingFor == 'TimeFrom':

mo = Time\_regex.search(current\_line)

if mo:

time\_from, time\_to = mo.groups()

lookingFor = 'RegName'

continue

elif lookingFor == 'RegName':

mo = NameUtt\_regex.search(current\_line)

if mo:

reg\_name, utterance = mo.groups()

else:

# If the current line does not match NameUtt\_regex,

# check if it's because it's a continuation of the last utterance or

# has no named speaker.

if current\_line.strip() == "":

# Skip empty lines

continue

elif not unknown\_speaker:

# If there's no speaker, assign UNKNOWN

reg\_name = "UNKNOWN"

utterance = current\_line.strip()

# Append values to lists

sno\_list.append(sno)

time\_from\_list.append(time\_from)

time\_to\_list.append(time\_to)

reg\_name\_list.append(reg\_name)

utterance\_list.append(utterance)

# At the end of dialogue block, next thing look for SNo

lookingFor = 'SNo'

# Create DataFrame from lists

df = pd.DataFrame({

'SNo': sno\_list,

'TimeFrom': time\_from\_list,

'TimeTo': time\_to\_list,

'RegName': reg\_name\_list,

'Utterance': utterance\_list

})

# Print for debugging

print(df)

# SQLAlchemy engine URL for MySQL

# to handle special characters in password

engine\_url = URL.create(

"mysql+pymysql",

username="root",

password="Ihtrmp@2405",

host="localhost",

database="anl503eca"

)

engine = create\_engine(engine\_url)

# Create table in MySQL

df.to\_sql('vtt', con=engine, if\_exists='replace', index=False)

# Q1b

A screenshot of a computer

Description automatically generated

-- Check data type of vtt table DESCRIBE vtt;

-- Convert TimeFrom and TimeTo to TIME type

ALTER TABLE vtt MODIFY TimeFrom TIME(3);

ALTER TABLE vtt MODIFY TimeTo TIME(3);

-- Use TIMESTAMPDIFF to calculate the difference between two datetime

-- Specify the unit as microsecond, the two datetime: TimeFrom and TimeTo

CREATE TABLE vttclean AS

SELECT

\*,

TIMESTAMPDIFF(

MICROSECOND, TimeFrom, TimeTo)

DIV 1000 AS milliseconds

FROM

vtt;

-- To see result

SELECT

\*

FROM

vttclean;

# Q1c

A graph of a number of blue bars

Description automatically generated with medium confidence

require(RMySQL)

require(ggplot2)

require(scales)

# Connect to database

con = dbConnect(MySQL(), dbname='anl503eca')

vttclean = suppressWarnings(expr=dbReadTable(conn=con, name="vttclean"))

dbDisconnect(conn=con)

# Filter out non-student entries

student\_only <- vttclean[vttclean$RegName != "INSTRUCTOR" & vttclean$RegName != "UNKNOWN",]

# Calculate total airtime for each student using sum

total\_airtime\_per\_student <- aggregate(milliseconds ~ RegName, data = student\_only, sum)

# Renaming columns

names(total\_airtime\_per\_student) <- c("Student", "TotalMilliseconds")

# Creating the barchart with improved label positioning

ggplot(total\_airtime\_per\_student, aes(x = reorder(Student, -TotalMilliseconds), y = TotalMilliseconds)) +

geom\_bar(stat = "identity", fill = "navy") +

geom\_text(aes(label = scales::comma(TotalMilliseconds)), # Comma as thousand separator

position = position\_nudge(y = 0.5), # Nudge labels to the right of the bar

hjust = -0.1, # Adjust horizontal position to ensure labels are outside the bars

size = 3.5, # Text size

color = "black") + # Text color

theme\_minimal() +

labs(title = "Total Airtime per Student",

x = "Student",

y = "Total Airtime (milliseconds)") +

scale\_y\_continuous(labels = label\_number(big.mark = ",")) + # Scale and put comma as thousand separator

coord\_flip() # Flipping the coordinates for better readability

**End of Report**