

**ANL503 Data Wrangling**

**End-of-Course Assessment Jan Semester 2024**

**Name: Chan Wen Le (B2411245)**

**Group: T01**

Question 1a

|  |
| --- |
| # Import relevant libraries  import re  import pandas as pd  import pymysql  from sqlalchemy import create\_engine  # -- set up regular expressions patterns ----  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\*$')  NameUtt\_regex = re.compile(r'^([^:]+):\s?(.\*)$')  UttOnly\_regex = re.compile(r'^([^:]+)$')  # -- read in vtt file ----  vttpath = 'captured\_dialogue.vtt'  vtt = open(vttpath)  all\_lines = vtt.readlines()  vtt.close()  # -- create an empty DataFrame with five columns ----  colnames = ['SNo','TimeFrom','TimeTo','RegName','Utterance']  df = pd.DataFrame(columns=colnames)  lookingFor = 'INIT'  for i, current\_line in enumerate(all\_lines):  if lookingFor == 'INIT':  mo = INIT\_regex.search(current\_line)  if mo!= None:  lookingFor = 'SNo'  elif lookingFor == 'SNo':  mo = SNo\_regex.match(current\_line)  if mo:  SNo = mo.group(1)  lookingFor = 'Time'  elif lookingFor == 'Time':  mo = Time\_regex.match(current\_line)  if mo:  TimeFrom = mo.group(1)  TimeTo = mo.group(2)  lookingFor = 'NameUtt\_or\_UttOnly'  elif lookingFor == 'NameUtt\_or\_UttOnly':  mo\_nameutt = NameUtt\_regex.match(current\_line)  mo\_utt\_only = UttOnly\_regex.match(current\_line)  if mo\_nameutt:  RegName = mo\_nameutt.group(1)  Utterance = mo\_nameutt.group(2)  df.loc[len(df)] = [SNo, TimeFrom, TimeTo, RegName, Utterance]  lookingFor = 'SNo'  elif mo\_utt\_only:  RegName = None # Assign None or a default speaker name  Utterance = mo\_utt\_only.group(1)  df.loc[len(df)] = [SNo, TimeFrom, TimeTo, RegName, Utterance]  lookingFor = 'SNo'  print(df)  from sqlalchemy import create\_engine  engine = create\_engine('mysql+pymysql://root:111111@localhost/anl503')  df.to\_sql('vtt', con=engine, index=False,if\_exists='replace') |

Question 1b

|  |
| --- |
| -- Step 1: Duplicate the table 'vtt' to 'vttclean'  CREATE TABLE vttclean AS SELECT \* FROM vtt;  -- Step 2: Alter data type for timeto and timefrom  ALTER TABLE vtt  MODIFY COLUMN timefrom TIME(3),  MODIFY COLUMN timeto TIME(3);  -- Step 3: Perform the substraction  SELECT \*, TIMESTAMPDIFF(MICROSECOND, timefrom, timeto) / 1000 AS milliseconds  FROM vtt;  -- Step 4: Add milliseconds into the table  ALTER TABLE vttclean ADD COLUMN milliseconds INT;  -- Step 5: Update and save the function permenantly  UPDATE vttclean  SET milliseconds = TIMESTAMPDIFF(MICROSECOND, timefrom, timeto) / 1000; |

Question 1c

|  |
| --- |
| # Deploy library  library(RMySQL)  library(dplyr)  library(forcats)  library(ggplot2)  # Link to SQL  con = dbConnect(MySQL(), dbname="anl503", user="root", password="111111")  df = suppressWarnings(dbReadTable(conn=con, name="vttclean"))  dbDisconnect(conn=con)  # Calculate the sum of milliseconds for each RegName  result <- df %>%  group\_by(RegName) %>%  summarise(TotalDuration = sum(milliseconds))  # View the result  print(result)  # Drop rows where RegName is 'INSTRUCTOR' & RegName is NA  result <- result[!(result$RegName == 'INSTRUCTOR' | is.na(result$RegName)), ]  #Arrange the sequence of the bar as descending order of Total Duration  result$RegName <- fct\_reorder(result$RegName, result$TotalDuration, .desc = TRUE)  # Plot out the bar chart  ggplot(result, aes(x = RegName, y=TotalDuration)) +  geom\_bar(stat = 'identity',fill = "blue", color = "black")+  geom\_text(aes(label = TotalDuration),vjust = -0.3, color = "black", size = 2.1) +  scale\_y\_continuous(labels = scales::comma) +  labs(x = "Name of Students", y = "Total Duration(milliseconds)", title = "Accumulated Airtime for Students who Spoke Up ")+  theme(axis.text.x = element\_text(angle = 45, hjust = 1),plot.title = element\_text(hjust = 0.5)) |

Result:

