# Function to mimic API1 in R

API1 <- function(in\_newname, in\_date, param\_precinct) {

# Placeholder for out\_winner

out\_winner <- NA

# Check if timestamp is correct

var\_check2 <- sum(Penna$timestamp == in\_date)

if (var\_check2 <= 0) {

out\_winner <- 0

stop("incorrect timestamp")

}

# Check if precinct is correct

var\_check1 <- sum(Penna$precinct == param\_precinct)

if (var\_check1 <= 0) {

out\_winner <- 0

stop("incorrect precinct")

}

# Extract max and min timestamps

v\_maxtimestamp <- max(Penna$Timestamp[Penna$Timestamp == as.Date(in\_date)])

v\_mintimestamp <- min(Penna$Timestamp[Penna$Timestamp == as.Date(in\_date)])

# Check if the date is within the valid range

if (as.Date(in\_date) < v\_mintimestamp) {

out\_winner <- 0

stop("incorrect candidate")

}

# Check candidate and select the winner

if (in\_newname == 'Biden') {

out\_winner <- Penna$Biden[Penna$Timestamp == v\_maxtimestamp & Penna$precinct == param\_precinct]

} else if (in\_newname == 'Trump') {

out\_winner <- Penna$Trump[Penna$Timestamp == v\_maxtimestamp & Penna$precinct == param\_precinct]

} else {

out\_winner <- 0

stop("incorrect candidate")

}

return(out\_winner)

}

# Sample Data (Assuming Penna is a data frame)

Penna <- data.frame(

Timestamp = as.POSIXct(c('2020-11-06 19:10:53', '2020-11-06 19:10:53')),

precinct = c('abcddee', 'Hanover'),

Biden = c(1, 2),

Trump = c(3, 4)

)

# Call with wrong precinct

qty\_wrong\_precinct <- API1('Biden', '2020-11-06 19:10:53', 'abcddee')

print(qty\_wrong\_precinct)

# Call with correct precinct

qty\_correct\_precinct <- API1('Biden', '2020-11-06 19:10:53', 'Hanover')

print(qty\_correct\_precinct)

# Query the data of the table

selected\_data <- Penna[Penna$Timestamp == '2020-11-06 19:10:53' & Penna$precinct == 'Hanover', ]

print(selected\_data)

# API 2 in R

API2 <- function(param\_date) {

# Placeholder for out\_precinct

out\_precinct <- NA

# Extract max timestamp

max\_timestamp <- max(as.Date(Penna$Timestamp[Penna$Timestamp == as.Date(param\_date)]))

# Calculate sums for Biden and Trump

result <- aggregate(cbind(Biden, Trump) ~ Timestamp, data = Penna[Penna$Timestamp == max\_timestamp, ], sum)

# Extract values

v\_timestamp <- as.character(result$Timestamp)

v\_Biden <- sum(result$Biden)

v\_Trump <- sum(result$Trump)

# Determine winner

if (v\_Biden < v\_Trump) {

out\_precinct <- paste("the last timestamp:", v\_timestamp, ", Trump‘s votes:", v\_Trump)

} else {

out\_precinct <- paste("the last timestamp:", v\_timestamp, ", Biden‘s votes:", v\_Biden)

}

return(out\_precinct)

}

# Sample Call for API 2

out\_precinct\_API2 <- API2('2020-11-06')

print(out\_precinct\_API2)

# API 3 in R

API3 <- function(param\_precinct) {

# Placeholder for out\_precinct

out\_precinct <- NA

if (param\_precinct == 'Biden') {

result <- aggregate(totalvotes ~ precinct, data = Penna[Biden > Trump, ], max)

result <- result[order(result$totalvotes, decreasing = TRUE), ][1:10, ]

out\_precinct <- result

} else if (param\_precinct == 'Trump') {

result <- aggregate(totalvotes ~ precinct, data = Penna[Biden < Trump, ], max)

result <- result[order(result$totalvotes, decreasing = TRUE), ][1:10, ]

out\_precinct <- result

} else {

out\_precinct <- 'incorrect candidate'

stop('incorrect candidate')

}

return(out\_precinct)

}

# Sample Calls for API 3

out\_precinct\_API3\_Biden <- API3('Biden')

print(out\_precinct\_API3\_Biden)

out\_precinct\_API3\_Trump <- API3('Trump')

print(out\_precinct\_API3\_Trump)

# API 4 in R

API4 <- function(param\_precinct) {

# Placeholder for out\_winner

out\_winner <- NA

# Check if precinct is correct

var\_check1 <- sum(Penna$precinct == param\_precinct)

if (var\_check1 <= 0) {

out\_winner <- 0

stop("incorrect precinct")

}

# Calculate sums and percentages

result <- aggregate(cbind(Biden, Trump, totalvotes) ~ precinct, data = Penna[Penna$precinct == param\_precinct, ], sum)

result$Bidenpercent <- round(result$Biden / result$totalvotes \* 100, 2)

result$Trumppercent <- round(result$Trump / result$totalvotes \* 100, 2)

# Determine winner

if (result$Biden < result$Trump) {

out\_winner <- paste(result$precinct, 'winner: Trump, percentage of total:', result$Trumppercent, '%')

} else {

out\_winner <- paste(result$precinct, 'winner: Biden, percentage of total:', result$Bidenpercent, '%')

}

return(out\_winner)

}

# Sample Calls for API 4

out\_winner\_wrong\_precinct <- API4('Lower')

print(out\_winner\_wrong\_precinct)

out\_winner\_correct\_precinct1 <- API4('Lower Providence 1-1')

print(out\_winner\_correct\_precinct1)

out\_winner\_correct\_precinct2 <- API4('Lower Providence 2-4')

print(out\_winner\_correct\_precinct2)

# API 5 in R

API5 <- function(param\_precinct) {

# Placeholder for out\_winner2

out\_winner2 <- NA

# Check if precinct is correct

var\_check1 <- sum(grepl(param\_precinct, Penna$precinct))

if (var\_check1 <= 0) {

out\_winner2 <- 0

stop("incorrect precinct")

}

# Calculate sums for Biden and Trump

result <- aggregate(cbind(Biden, Trump) ~ precinct, data = Penna[grepl(param\_precinct, Penna$precinct), ], sum)

v\_Biden <- sum(result$Biden)

v\_Trump <- sum(result$Trump)

# Determine winner

if (v\_Biden < v\_Trump) {

out\_winner2 <- paste(param\_precinct, 'winner: Trump, how many votes:', v\_Trump)

} else {

out\_winner2 <- paste(param\_precinct, 'winner: Biden, how many votes:', v\_Biden)

}

return(out\_winner2)

}

# Sample Calls for API 5

out\_winner\_wrong\_precinct <- API5('abccddeeee')

print(out\_winner\_wrong\_precinct)

out\_winner\_correct\_precinct <- API5('Township')

print(out\_winner\_correct\_precinct)

# Create newPenna table

newPenna <- data.frame(

precinct = character(),

Timestamp = as.POSIXct(character()),

newvotes = integer(),

new\_Trump = integer(),

new\_Biden = integer(),

stringsAsFactors = FALSE

)

# newPenna procedure in R

newPenna <- function(in\_newPenna, in\_date) {

# Check if precinct is correct

var\_check1 <- sum(Penna$precinct == in\_newPenna)

if (var\_check1 <= 0) {

stop("incorrect precinct")

}

# Initialize variables

var\_date <- var\_totalvotes <- var\_Trump <- var\_Biden <- old\_totalvotes <- old\_Trump <- old\_Biden <- NA

out\_totalvotes <- out\_new\_Trump <- out\_new\_Biden <- NA

# Fetch data using cursor

for (row in Penna[Penna$timestamp < in\_date & Penna$precinct == in\_newPenna &

as.Date(Penna$timestamp) == as.Date(in\_date),]) {

var\_date <- row$timestamp

old\_totalvotes <- row$totalvotes

old\_Trump <- row$Trump

old\_Biden <- row$Biden

}

# Get values for the specified timestamp and precinct

selected\_row <- Penna[Penna$timestamp == in\_date & Penna$precinct == in\_newPenna &

as.Date(Penna$timestamp) == as.Date(in\_date), ]

var\_totalvotes <- selected\_row$totalvotes

var\_Trump <- selected\_row$Trump

var\_Biden <- selected\_row$Biden

# Calculate differences

out\_totalvotes <- var\_totalvotes - old\_totalvotes

out\_new\_Trump <- var\_Trump - old\_Trump

out\_new\_Biden <- var\_Biden - old\_Biden

# Insert into newPenna table

newPenna <<- rbind(newPenna, data.frame(

precinct = in\_newPenna,

Timestamp = in\_date,

newvotes = out\_totalvotes,

new\_Trump = out\_new\_Trump,

new\_Biden = out\_new\_Biden

))

}

# Sample Calls for newPenna procedure

newPenna('abcdeeff', '2020-11-06 19:10:53')

print(newPenna)

newPenna('Hanover', '2020-11-06 19:10:53')

print(newPenna)

# Switch procedure in R

Switch <- function() {

# Get data for the switch

switch\_data <- Penna[Penna$timestamp < max(Penna$timestamp) &

as.Date(Penna$timestamp) >= (max(Penna$timestamp) - 24 \* 60 \* 60) &

Penna$Biden < Penna$Trump,]

# Create the result dataframe

result <- data.frame(

precinct = switch\_data$precinct,

Timestamp = switch\_data$timestamp,

fromCandidate = 'Trump',

toCandidate = 'Biden'

)

return(result)

}

# Sample Calls for Switch procedure

switch\_result <- Switch()

print(switch\_result)

Part 3:

# newP3CHECK procedure in R

newP3CHECK <- function() {

# Define local variables

var\_check1 <- var\_check2 <- var\_check3 <- var\_check4 <- var\_check5 <- 0

is\_check <- TRUE

# Check constraint 1

if (sum(Penna$totalvotes < (Penna$Trump + Penna$Biden)) > 0) {

is\_check <- FALSE

stop("FALSE (constraint is not satisfied)")

}

# Check constraint 2

if (sum(Penna$timestamp > '2020-11-11 23:59:59') > 0) {

is\_check <- FALSE

stop("FALSE (constraint is not satisfied)")

}

# Check constraint 3

if (sum(as.Date(Penna$timestamp) > '2020-11-11') > 0) {

is\_check <- FALSE

stop("FALSE (constraint is not satisfied)")

}

# Check constraint 4

if (sum(as.Date(Penna$timestamp) < '2020-10-03') > 0) {

is\_check <- FALSE

stop("FALSE (constraint is not satisfied)")

}

# Check constraint 5

result <- merge(

subset(Penna, as.Date(Penna$timestamp) == '2020-11-05'),

subset(Penna, as.Date(Penna$timestamp) == '2020-11-05'),

by = 'precinct'

)

if (sum(result$maxtotal < result$mintotal) > 0) {

is\_check <- FALSE

stop("FALSE (constraint is not satisfied)")

}

return(is\_check)

}

# Sample Call for newP3CHECK procedure

is\_check <- newP3CHECK()

print(is\_check)

Part 4:

# Define data frames for each table

Updated\_Tuples <- data.frame(

ID = integer(),

Timestamp = as.POSIXct(character()),

state = character(),

locality = character(),

precinct = character(),

geo = character(),

totalvotes = integer(),

Biden = integer(),

Trump = integer(),

filestamp = character(),

stringsAsFactors = FALSE

)

Inserted\_Tuples <- data.frame(

ID = integer(),

Timestamp = as.POSIXct(character()),

state = character(),

locality = character(),

precinct = character(),

geo = character(),

totalvotes = integer(),

Biden = integer(),

Trump = integer(),

filestamp = character(),

stringsAsFactors = FALSE

)

Deleted\_Tuples <- data.frame(

ID = integer(),

Timestamp = as.POSIXct(character()),

state = character(),

locality = character(),

precinct = character(),

geo = character(),

totalvotes = integer(),

Biden = integer(),

Trump = integer(),

filestamp = character(),

stringsAsFactors = FALSE

)

BAK\_Tuples <- data.frame(

ID = integer(),

Timestamp = as.POSIXct(character()),

state = character(),

locality = character(),

precinct = character(),

geo = character(),

totalvotes = integer(),

Biden = integer(),

Trump = integer(),

filestamp = character(),

stringsAsFactors = FALSE

)

# Insert Trigger in R

insert\_trigger <- function(new\_data) {

Inserted\_Tuples <<- rbind(Inserted\_Tuples, new\_data)

}

# Update Trigger in R

update\_trigger <- function(old\_data) {

Updated\_Tuples <<- rbind(Updated\_Tuples, old\_data)

}

# Delete Trigger in R

delete\_trigger <- function(old\_data) {

Deleted\_Tuples <<- rbind(Deleted\_Tuples, old\_data)

}

# MoveVotes procedure in R

MoveVotes <- function(param\_precinct, in\_date, in\_newname, Number\_of\_Moved\_Votes) {

# Declare local variables

var\_check1 <- var\_check2 <- var\_check3 <- 0

v\_Biden <- v\_Trump <- 0

out\_return <- ""

# Check timestamp

if (sum(Penna$timestamp == in\_date) <= 0) {

out\_return <- "Unknown Timestamp"

stop(out\_return)

}

# Check precinct

if (sum(Penna$precinct == param\_precinct) <= 0) {

out\_return <- "wrong precinct name"

stop(out\_return)

}

# Check votes

if (Number\_of\_Moved\_Votes <= 0) {

out\_return <- "Not enough votes"

stop(out\_return)

}

# Get candidate votes

if (in\_newname == 'Biden') {

v\_Biden <- Penna$Biden[Penna$timestamp == in\_date & Penna$precinct == param\_precinct]

if (v\_Biden < Number\_of\_Moved\_Votes) {

out\_return <- "Not enough votes"

stop(out\_return)

}

} else if (in\_newname == 'Trump') {

v\_Trump <- Penna$Trump[Penna$timestamp == in\_date & Penna$precinct == param\_precinct]

if (v\_Trump < Number\_of\_Moved\_Votes) {

out\_return <- "Not enough votes"

stop(out\_return)

}

} else {

out\_return <- "wrong candidate name"

stop(out\_return)

}

# Update Data

if (in\_newname == 'Biden') {

Penna$Trump[Penna$timestamp >= in\_date & Penna$precinct == param\_precinct] <-

Penna$Trump[Penna$timestamp >= in\_date & Penna$precinct == param\_precinct] + Number\_of\_Moved\_Votes

Penna$Biden[Penna$timestamp >= in\_date & Penna$precinct == param\_precinct] <-

Penna$Biden[Penna$timestamp >= in\_date & Penna$precinct == param\_precinct] - Number\_of\_Moved\_Votes

out\_return <- "Biden update true"

} else if (in\_newname == 'Trump') {

Penna$Trump[Penna$timestamp >= in\_date & Penna$precinct == param\_precinct] <-

Penna$Trump[Penna$timestamp >= in\_date & Penna$precinct == param\_precinct] - Number\_of\_Moved\_Votes

Penna$Biden[Penna$timestamp >= in\_date & Penna$precinct == param\_precinct] <-

Penna$Biden[Penna$timestamp >= in\_date & Penna$precinct == param\_precinct] + Number\_of\_Moved\_Votes

out\_return <- "Trump update true