Exhibit K

TABLE A

HIGHLY CONFIDENTIAL ATTORNEYS' EYES ONLY

Research Question Numerators - Corrected

```
# load R packages
library(tidyverse)
library(odbc)
library(DBI)
library(glue)

# connect to database
con <- dbConnect(odbc(),
  Driver   = "ODBC Driver 18 for SQL Server",
  Server   = "SECEPRELANTSQL1.EX.LOCAL",
  Database = "AnthropicSandbox",
  Trusted_Connection = "yes",
  Encrypt  = "yes",
  TrustServerCertificate = "yes")
```

Question 1

```
# set parameters for the computation
batch_size <- 50000
new_field <- "Q1v2Corrected"
field_to_search <- "LastPrompt"
```

```
verb <- "\\b(?:(?:give\\s*(?:for|to)?\\s*me|send\\s*(?:for|to)?\\s*me|provide\
\s*(?:for|to)?\\s*me|get\\s*(?:for|to)?\\s*me)|(?:whats|what's|what\\s(?:is|
are)))\\b"
optional_2_word_buffer <- "(?:\\s*\\S+\\s+){0,2}?"
object <- "\\b(?:lyric(s)?|chords|music|(?<!bible\\s)verse(s)?|chorus(es)?|(?
<!\\d\\s)words)\\b"

general_pattern <- paste0(verb, optional_2_word_buffer, object)
specific_pattern <- paste0("\\bwrite\\s*(?:me)?\\s*the\\s*lyrics\\s*to\\b")
pattern <- paste0("(?:", general_pattern, "|", specific_pattern, ")")
```

```
# run search in batches and update Q1v2Corrected in db
batch_num <- 0L

repeat {
  df <- dbGetQuery(con, glue("
    SELECT TOP ({batch_size}) REVIEWID, {field_to_search}
    FROM DocumentSandbox
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

```
    WHERE ValidRecordsCorrected = 1
      AND {new_field} IS NULL;
  "))

  if (nrow(df) == 0L) break

  batch_num <- batch_num + 1L

  # search for matches to pattern
  df[[new_field]] <- as.integer(str_detect(df[[field_to_search]],
regex(pattern, ignore_case = TRUE)))

  # write to a temporary table
  dbWriteTable(
    con, "#FieldToUpdate",
    df[, c("REVIEWID", new_field)],
    temporary = TRUE, overwrite = TRUE,
    field.types = c(
      "REVIEWID" = "varchar(25)",
      setNames("tinyint", new_field)))

  # join the temporary table to the main table to add new field
  dbExecute(con, glue("
    UPDATE ds
    SET ds.{new_field} = u.{new_field}
    FROM DocumentSandbox ds
    JOIN #FieldToUpdate u ON ds.REVIEWID = u.REVIEWID;
  "))

  # remove temporary table
  dbExecute(con, "DROP TABLE #FieldToUpdate;")

  message(glue("[Batch {batch_num}] processed {nrow(df)} rows at
{Sys.time()}"))
}
```

Question 2

```
# load packages
library(tidyverse)
library(readxl)

# load works
wis_a <- read_excel("works-in-suit/A. Bray Additional Variations_2025.12.16 -
FINAL - Song + Artist List_Variations - Keystone.xlsx",
  range = cell_limits(c(2, 1), c(NA, 3)),
  col_names = c("wis_num", "song_title", "artist"))
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

```
# make the songs and artists regex safe
escape_regex <- function(x) {
  stringr::str_replace_all(x, "([\\^$.|?*+()\\[\\]{}\\\\])", "\\\\\\1")
}

# also allow for 0 or more spaces between words
wis_a <- wis_a |>
  mutate(artist = escape_regex(artist),
         artist = str_replace_all(artist, " ", "\\s*"),
         song_title = escape_regex(song_title),
         song_title = str_replace_all(song_title, " ", "\\s*"))

# combine artist and songs into alternating regex
artist_song_wis <- paste0(
  "(?<!\\w)(?:", # previous character is not wordlike
  paste(c(wis_a$artist, wis_a$song_title), collapse = "|"),
  ")(?!\\w)") # following character is not wordlike

# form pattern
verb <- "\\b(?:(?:give\\s*(?:for|to)?|send\\s*(?:for|to)?|provide\\s*(?:for|
to)?|get\\s*(?:for|to)?)(?:me)?|(?:whats|what's|what\\s(?:is|are)))\\b"
optional_2_word_buffer <- "(?:\\s*\\S+\\s+){0,2}?"
object <- "\\b(?:lyric(s)?|chords|music|verse(s)?|chorus(es)?|(?<!\\d\
\s)words)\\b"

general_pattern <- paste0(verb, optional_2_word_buffer, object)
specific_pattern <- paste0("\\bwrite\\s*(?:me)?\\s*the\\s*lyrics\\s*to\\b")
request_for_lyrics <- paste0("(?:", general_pattern, "|", specific_pattern,
")")

optional_5_word_buffer <- "(?:\\s*\\S+\\s+){0,5}?"
pattern <- paste0(request_for_lyrics, optional_5_word_buffer, artist_song_wis)
```

```
# set parameters for the computation
batch_size <- 50000
new_field <- "Q2Corrected"
field_to_search <- "LastPrompt"
```

```
# run search in batches and update Q2Corrected in db
batch_num <- 0L

repeat {
  df <- dbGetQuery(con, glue("
    SELECT TOP ({batch_size}) REVIEWID, {field_to_search}
    FROM DocumentSandbox
    WHERE ValidRecordsCorrected = 1
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

```
      AND {new_field} IS NULL
      AND Q1v2Corrected = 1;
  "))

  if (nrow(df) == 0L) break

  batch_num <- batch_num + 1L

  # search for matches to pattern
  df[[new_field]] <- as.integer(str_detect(df[[field_to_search]],
regex(pattern, ignore_case = TRUE)))

  # write to a temporary table
  dbWriteTable(
    con, "#FieldToUpdate",
    df[, c("REVIEWID", new_field)],
    temporary = TRUE, overwrite = TRUE,
    field.types = c(
      "REVIEWID" = "varchar(25)",
      setNames("tinyint", new_field)))

  # join the temporary table to the main table to add new field
  dbExecute(con, glue("
    UPDATE ds
    SET ds.{new_field} = u.{new_field}
    FROM DocumentSandbox ds
    JOIN #FieldToUpdate u ON ds.REVIEWID = u.REVIEWID;
  "))

  # remove temporary table
  dbExecute(con, "DROP TABLE #FieldToUpdate;")

  message(glue("[Batch {batch_num}] processed {nrow(df)} rows at
{Sys.time()}"))
}
```

Question 3

```
batch_size <- 50000
new_field <- "Q3v2Corrected"
field_to_search <- "LastOutput"
p_gives_lyrics <- "\\bhere\\s(?:are|is)\\sthe\\s(?:genuine|original|real|true|
actual)?\\s*(?:lyrics|chords|music|verse(s)?|chorus(es)?|words)"
```

```
batch_num <- 0L

repeat {
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

```
  df <- dbGetQuery(con, glue("
    SELECT TOP ({batch_size}) REVIEWID, {field_to_search}
    FROM DocumentSandbox
    WHERE ValidRecordsCorrected = 1
      AND {new_field} IS NULL;
  "))

  if (nrow(df) == 0L) break

  batch_num <- batch_num + 1L

  # search for matches to patterns
  gives_lyrics <- str_detect(df[[field_to_search]], regex(p_gives_lyrics,
ignore_case = TRUE))
  df[[new_field]] <- gives_lyrics

    # write to a temporary table
  dbWriteTable(
    con, "#FieldToUpdate",
    df[, c("REVIEWID", new_field)],
    temporary = TRUE, overwrite = TRUE,
    field.types = c(
      "REVIEWID" = "varchar(25)",
      setNames("tinyint", new_field)))

  # join the temporary table to the main table to add new field
  dbExecute(con, glue("
    UPDATE ds
    SET ds.{new_field} = u.{new_field}
    FROM DocumentSandbox ds
    JOIN #FieldToUpdate u ON ds.REVIEWID = u.REVIEWID;
  "))

  # remove temporary table
  dbExecute(con, "DROP TABLE #FieldToUpdate;")

  message(glue("[Batch {batch_num}] processed {nrow(df)} rows at
{Sys.time()}"))
}
```

Question 4

Method 1: N-grams from lyrics

```
wis_exhibit_a <- read_csv("works-in-suit/wis_exhibit_a.csv")
wis_exhibit_b <- read_csv("works-in-suit/wis_exhibit_b.csv")
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

```
library(tidytext)

# split lyrics into ngrams
ngram_df <- wis_exhibit_b |>
  unnest_tokens(output = ngram, input = lyrics, token = "ngrams", n = 4,
to_lower = TRUE)

# cross song_title with index to make unique identifier
ngram_df <- ngram_df |>
  unite("song_id", song_title, index)

# remove duplicates ngrams from within each song
ngram_by_song <- ngram_df |>
  group_by(song_id) |>
  distinct(ngram) |>
  ungroup()

# distinct ngrams across songs
ngrams_wis <- ngram_by_song |>
  distinct(ngram) |>
  pull(ngram)
```

```
# escape regex metacharacters in the n-grams
escape_regex <- function(x) {
  stringr::str_replace_all(x, "([\\^$.|?*+()\\[\\]{}\\\\])", "\\\\\\1")
}
ngrams_wis_noregex <- escape_regex(ngrams_wis)

# make one big alternation with word boundaries
p_all_ngrams <- paste0("\\b(", paste(ngrams_wis_noregex, collapse = "|"), ")\
\b")
```

```
# consolidate all ngrams for a song into a single regex
p_by_song_df <- ngram_by_song |>
  mutate(ngram = escape_regex(ngram)) |>
  group_by(song_id) |>
  summarize(p_ngrams = paste0("\\b(", paste(ngram, collapse = "|"), ")\\b"))

# convert the regex to a named vector
p_by_song_vec <- p_by_song_df$p_ngrams
names(p_by_song_vec) <- p_by_song_df$song_id

# song length in ngrams
song_counts <- count(ngram_by_song, song_id)
song_length <- song_counts$n
names(song_length) <- song_counts$song_id
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

```
batch_size <- 500
new_field <- "Q4ngramCorrected"
field_to_search <- "LastOutput"
```

```
library(stringi)
batch_num <- 0L

repeat {
  df <- dbGetQuery(con, glue("
    SELECT TOP ({batch_size}) REVIEWID, {field_to_search}, TimePeriod
    FROM DocumentSandbox
    WHERE ValidRecordsCorrected = 1
      AND Q3v2Corrected = 1
      AND {new_field} IS NULL;
  "))

  if (nrow(df) == 0L) break

  batch_num <- batch_num + 1L

  counts_per_song <- vapply(
    p_by_song_vec,
    FUN = function(pat) stri_count_regex(df$LastOutput, pat),
    FUN.VALUE = integer(nrow(df)))

  df2 <- bind_cols(df, counts_per_song)

  df2 <- df2 |>
    mutate(across(-c(1:3), ~ .x / song_length[cur_column()], .names =
"prop_{.col}"))

  df2 <- df2 |>
    rowwise() |>
    mutate(max_prop = max(c_across(starts_with("prop_")), na.rm = TRUE)) |>
    ungroup()

  df2 <- df2 |>
    mutate(Q4ngramCorrected = max_prop > .05)

  # write to a temporary table
  dbWriteTable(
    con, "#FieldToUpdate",
    df2[, c("REVIEWID", new_field)],
    temporary = TRUE, overwrite = TRUE,
    field.types = c(
      "REVIEWID" = "varchar(25)",
      setNames("tinyint", new_field)))
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

```
  # join the temporary table to the main table to add new field
  dbExecute(con, glue("
    UPDATE ds
    SET ds.{new_field} = u.{new_field}
    FROM DocumentSandbox ds
    JOIN #FieldToUpdate u ON ds.REVIEWID = u.REVIEWID;
  "))

  # remove temporary table
  dbExecute(con, "DROP TABLE #FieldToUpdate;")

  message(glue("[Batch {batch_num}] processed {nrow(df)} rows at
{Sys.time()}"))
}
```

Method 2: regex with songs and artists

```
# load works
wis_a <- read_excel("works-in-suit/A. Bray Additional Variations_2025.12.16 -
FINAL - Song + Artist List_Variations - Keystone.xlsx",
  range = cell_limits(c(2, 1), c(NA, 3)),
  col_names = c("wis_num", "song_title", "artist"))

# make the songs and artists regex safe
escape_regex <- function(x) {
  stringr::str_replace_all(x, "([\\^$.|?*+()\\[\\]{}\\\\])", "\\\\\\1")
}

# also allow for 0 or more spaces between words
wis_a <- wis_a |>
  mutate(artist = escape_regex(artist),
         artist = str_replace_all(artist, " ", "\\s*"),
         song_title = escape_regex(song_title),
         song_title = str_replace_all(song_title, " ", "\\s*"))

# combine artist and songs into alternating regex
song_wis <- paste0(
  "(?<!\\w)(?:", # previous character is not wordlike
  paste(paste0('"?', wis_a$song_title, '"?'), collapse = "|"),
  ")(?!\\w)") # following character is not wordlike

# form pattern
gives_lyrics <- "\\bhere\\s(?:are|is)\\sthe\\s(?:lyrics|chords|music|
verse(s)?|chorus(es)?|words)"
and_optional_more_lyrics <- "(?:\\s+and\\s+(?:lyrics|chords|music|verse(s)?|
chorus(es)?|words))?"
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

```
patternCorrected <- paste0(gives_lyrics, and_optional_more_lyrics, "\\s+(?:to|
from|for)\\s+(?:the song\\s+)?", song_wis)
```

```
# set parameters for the computation
batch_size <- 50000
new_field <- "Q4regexCorrected"
field_to_search <- "LastOutput"
```

```
# run search in batches and update Q4 in db
batch_num <- 0L

repeat {
  df <- dbGetQuery(con, glue("
    SELECT TOP ({batch_size}) REVIEWID, {field_to_search}
    FROM DocumentSandbox
    WHERE ValidRecordsCorrected = 1
      AND {new_field} IS NULL
      AND Q3v2Corrected = 1;
  "))

  if (nrow(df) == 0L) break

  batch_num <- batch_num + 1L

  # search for matches to pattern
  df[[new_field]] <- as.integer(str_detect(df[[field_to_search]],
regex(patternCorrected, ignore_case = TRUE)))

  # write to a temporary table
  dbWriteTable(
    con, "#FieldToUpdate",
    df[, c("REVIEWID", new_field)],
    temporary = TRUE, overwrite = TRUE,
    field.types = c(
      "REVIEWID" = "varchar(25)",
      setNames("tinyint", new_field)))

  # join the temporary table to the main table to add new field
  dbExecute(con, glue("
    UPDATE ds
    SET ds.{new_field} = u.{new_field}
    FROM DocumentSandbox ds
    JOIN #FieldToUpdate u ON ds.REVIEWID = u.REVIEWID;
  "))

  # remove temporary table
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

```
  dbExecute(con, "DROP TABLE #FieldToUpdate;")

  message(glue("[Batch {batch_num}] processed {nrow(df)} rows at
{Sys.time()}"))
}
```

```
# combine methods
dbExecute(con, "
  UPDATE DocumentSandbox
  SET Q4Corrected = CASE
      WHEN COALESCE(Q4ngramCorrected, 0) = 1
        OR COALESCE(Q4regexCorrected, 0) = 1
      THEN 1
      ELSE 0
  END
  WHERE Q4ngramCorrected IS NOT NULL
    OR Q4regexCorrected IS NOT NULL;
")
```

Question 5

```
df <- dbGetQuery(con, "
  SELECT REVIEWID, PreSuit, TimePeriod, Q1v2Corrected, Q2Corrected,
Q3v2Corrected, Q4Corrected
  FROM DocumentSandbox
  WHERE NotInSample = 0
    AND ValidRecordsCorrected = 1;
")
```

By PreSuit

```
df |>
  filter(Q1v2Corrected == 1) |>
  count(PreSuit)|>
  adorn_totals("row")
```

```
df |>
  filter(Q2Corrected == 1) |>
  count(PreSuit) |>
  adorn_totals("row")
```

```
df |>
  filter(Q4Corrected == 1) |>
  count(TimePeriod) |>
  adorn_totals("row")
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY

By TimePeriod

```
library(janitor)
df |>
  count(TimePeriod, Q1v2Corrected) |>
  pivot_wider(names_from = TimePeriod, values_from = n) |>
  adorn_totals("row")
```

```
df |>
  filter(Q1v2Corrected == 1) |>
  count(TimePeriod, Q2Corrected) |>
  pivot_wider(names_from = TimePeriod, values_from = n) |>
  adorn_totals("row")
```

```
df |>
  filter(Q3v2Corrected == 1) |>
  count(TimePeriod, Q4Corrected) |>
  pivot_wider(names_from = TimePeriod, values_from = n) |>
  adorn_totals("row")
```


HIGHLY CONFIDENTIAL
ATTORNEYS' EYES ONLY