FIT1043 Assignment 3
Renee Yeo Shu Ting
33518904

MacBook User.

TASK A (1)
Download file from Desktop: `cd ~/Desktop`

```
(base) reneeyeo@Renees-MacBook-Pro ~ % cd ~/Desktop
```

Shell command used:
`ls -lh corona_tweets.csv.gz`

Explanation:
- `ls` is used to list files and directories
- `-l` enables long list format with detailed information
- `h` provides readable file sizes
- `corona_tweets.csv.gz` is the file that I selected to perform action

Output:

```
(base) reneeyeo@Renees-MacBook-Pro Desktop % ls -lh corona_tweets.csv.gz
-rw-r--r--@ 1 reneeyeo  staff   118M May 20 14:21 corona_tweets.csv.gz
```

Answer: 118 Megabytes

TASK A (2)

One line code:
`gzcat corona_tweets.csv.gz | head -n 1 | tr '\t' '\n'`

Explanation:
- `gzcat corona_tweets.csv.gz` is to display the contents of the selected file without decompressing
- `head` is to extract the first few lines, `-n 1` is to specify only 1st line of the input
- `tr` is used for character translation or deletion, `'\t'` and `'\n'` means it transforms the tab-separated values in the first line into separate lines, making each string appear on a new line

Output:

```
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | head -n 1 | tr '\t' '\n'
Created
Tweet_ID
Text
User_ID
User
User_Location
Followers_Count
Friends_Count
Geo
Place_Type
Place_Name
Place_Country
Language
```

TASK A (3)

One line code:
`gzcat corona_tweets.csv.gz | wc -l`

Explanation:
- `gzcat corona_tweets.csv.gz` is to display the contents of the selected file without decompressing

- `wc -l` counts the number of lines that matches

Output:

```
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | wc -l
 1143559
```

TASK B (1)

One line code:
`gzcat corona_tweets.csv.gz | awk -F'\t' '{print $4}' | sort | uniq | wc -l`

Explanation:
- `gzcat corona_tweets.csv.gz` is to read the compressed file 'corona_tweets.csv.gz' and outputs the uncompressed contents

- `awk` is a command-line tool that checks for text on a line-by-line basis

- `-F'\t'` sets a tab field separator ('\t'), to specify the files are separated by tabs

- `'{print $4}'` is an instruction from the 'awk' statement. The purpose is to print the datasets from the fourth line of the file – to represent Twitter users.

- `sort` is used to ensure that Twitter users are sorted in ascending order

- `uniq` filters out any duplicates Twitter users ID and keeps the unique one

- `wc -l` counts the number of lines that has unique Twitter users

Output:

```
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{print $4}' | sort | uniq | wc -l
  641976
```

TASK B (2)

a) One line code:

```
gzcat corona_tweets.csv.gz | awk -F '\t' '{print $3}' | grep -i -w
"death" | wc -l
```

Explanation:

- `gzcat corona_tweets.csv.gz` is to read the compressed file 'Desktop/corona_tweets.csv.gz' and outputs the uncompressed contents

- `awk` is a command-line tool that checks for text on a line-by-line basis

- `-F'\t'` sets a tab field separator ('\t'), to specify the files are separated by tabs

- `'{print $3}'` is an instruction from the 'awk' statement. The purpose is to print the datasets from the third line of the file

- `grep -i -w "death"` searches for the word "death" in case of sensitive manner, capturing any combination of uppercase or lowercase letters that only match the whole word of "death"

- `wc -l` counts the number of lines that matches the search for number of tweets mentioning the word "death"

Output:

```
(base) reneeyeo@Renees-MacBook-Pro Assignment 3 % gzcat corona_tweets.csv.gz | a
wk -F '\t' '{print $3}' | grep -i -w "death" | wc -l
   19426
```

b) One line code:

```
gzcat corona_tweets.csv.gz | grep -i -w -E
'.*\b(deaths?|death[[:alpha:]]+)\b.*' | grep -vE '(^|[^A-Za-
z])(death|deaths|Death|Deaths)($|[^A-Za-z])' | wc -l
```

Explanation:

- `gzcat corona_tweets.csv.gz` is to read the compressed file 'corona_tweets.csv.gz' and outputs the uncompressed contents

- `grep` is a command tool used for searching and matching

- `-i` performs case-insensitive match, `-w` matches the whole words, and `-E` uses extended regular expressions for pattern matching

- `'.*\b(deaths?|death[[:alpha:]]+)\b.*'` matches any line that contains "death" or "deaths" with any combination of letters before or after it

- `grep -vE` inverts the match pattern of extended regular expressions

- `'(^|[^A-Za-z])(death|deaths|Death|Deaths)($|[^A-Za-z])'` matches the lines that do not have any non-alphabetic character before or after the word match

- `wc -l` counts the number of lines

Output:

```
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | grep -
i -w -E '.*\b(deaths?|death[[:alpha:]]+)\b.*' | grep -vE '(^|[^A-Za-z])(death|de
aths|Death|Deaths)($|[^A-Za-z])' | wc -l
     603
```

c) One line code:

```
gzcat corona_tweets.csv.gz | grep -i -w -E
'.*\b(deaths?|death[[:alpha:]]+)\b.*' | grep -vE '(^|[^A-Za-
z])(death|deaths|Death|Deaths)($|[^A-Za-z])' > myText.txt
```

Explanation:

- `> myText.txt` redirects the output of the previous command to a file named "myText.txt"

Output:

A file named "myText.txt" is created in my Desktop.

myText.txt

```
Fri May 15 04:28:05 +0000 2020  1261151340664930300     I AM STILL WAITING FOR HOLLY RAIN AGAINST
CORONA VIRUS DEATH ..WHICH IS ONLY ANSWER AFTER DOING ALL KIND OF RESEARCH .    1108753156845043700
        kamrantahawar2          563     4024    null    null                    en
Fri May 15 04:28:07 +0000 2020  1261151350420721700     RT @LisaMei62: 3. Clapper acting like he's
deathly afraid for his life re: exposure to COVID-19. What he's deathly afraid of is exposure of…
        1251676607682470000     Sunny38Suh      대한민국 서울       557     907     null    null
                en
Fri May 15 04:28:24 +0000 2020  1261151421778493400     I AM STILL WAITING FOR HOLLY RAIN AGAINST
CORONA VIRUS DEATH ..WHICH IS ONLY ANSWER AFTER DOING ALL KIND OF RESEARCH .... 1108753156845043700
        kamrantahawar2          563     4024    null    null                    en
Fri May 15 04:28:29 +0000 2020  1261151441239986200     @MaryMastersMS @ccmalinda77 @greggutfeld
Here's the Medical Director for New York City, NO WHERE TO BE FOUND! She needs to be held responsible
PERIOD! Her decision to put OLDER COVID PATIENTS IN NURSING HOMES CAUSING 33% of all NYC CV DEATHS
BECAUSE OF THAT DECISION. SHE HAS BLOOD ON HER HANDS!   2847719549      Bill Poremba    Las Vegas
        35      152     null    null                    en
Fri May 15 04:29:27 +0000 2020  1261151685893697500     RT @LisaMei62: 3. Clapper acting like he's
deathly afraid for his life re: exposure to COVID-19. What he's deathly afraid of is exposure of…
        3722160913      DeeAn           11048   9256    null    null                    en
Fri May 15 04:29:36 +0000 2020  1261151722958962700     @FreezyWriter Covid19 won't kill me or anyone
around me who isn't on their deathbed https://t.co/vMn3CCh35S   1077342658438332400     Dumb Trucker
                39      209     null    null                    en
Fri May 15 04:33:01 +0000 2020  1261152583328002000     @indivisible_oc @joinandrewdo
@TaxFighterSteel @MichelleSteelCA @OCSupBartlett @DougChaffee2010 TRUMP AND THE TRUMP MOVEMENT WERE
BEHIND \nTHE LIBERTY/FREEDOM PROPAGANDA PUNKS THAT\nBROKE DOWN THE STAY SAFELY AT HOME WAYS..\nNOW
THE DEATHS ARE GOING UP WITH TALK OF IT\nLASTING A LONG TIME 🆘 85,600 USA COVID19 \nDEATHS SINCE
FEBRUARY...TOO BAD‼️ https://t.co/o99vx5fRSk    3398590505      Lampits Richart         42      134
        null    null                    en
Fri May 15 04:33:11 +0000 2020  1261152626495893500     RT @lizzkatherine_: BREAKING NEWS: Louisiana
just released COVID-19 data which shows that African-Americans account for 70% of ALL DEATHS i…
1151639499245535200     Producer Named MegaMell         13      11      null    null
```

```
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | grep -
i -w -E '.*\b(deaths?|death[[:alpha:]]+)\b.*' | grep -vE '(^|[^A-Za-z])(death|de
aths|Death|Deaths)($|[^A-Za-z])' > myText.txt
```

TASK C (1)

a)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 <= 1000) print $4}'
| sort -u
```

b)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 >= 1001 && $7 <=
2000) print $4}' | sort -u
```

c)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 >= 2001 && $7 <=
3000) print $4}' | sort -u
```

d)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 >= 3001 && $7 <=
4000) print $4}' | sort -u
```

e)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 >= 4001 && $7 <=
5000) print $4}' | sort -u
```

f)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 >= 5001 && $7 <=
6000) print $4}' | sort -u
```

g)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 >= 6001 && $7 <=
7000) print $4}' | sort -u
```

h)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 >= 7001 && $7 <=
8000) print $4}' | sort -u
```

i)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 >= 8001 && $7 <=
9000) print $4}' | sort -u
```

j)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 >= 9001 && $7 <=
10000) print $4}' | sort -u
```

k)
```
gzcat corona_tweets.csv.gz | awk -F'\t' '{if ($7 > 10000) print $4}'
| sort -u
```

Output:

The above codes only show how do we group them.

Now I am showing the number of lines that exists from the above codes.

```
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 <= 1000) print $4}' | sort -u | wc -l
  455758
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 >= 1001 && $7 <= 2000) print $4}' | sort -u | wc -l
   68574
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 >= 2001 && $7 <= 3000) print $4}' | sort -u | wc -l
   31281
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 >= 3001 && $7 <= 4000) print $4}' | sort -u | wc -l
   18803
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 >= 4001 && $7 <= 5000) print $4}' | sort -u | wc -l
   11699
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 >= 5001 && $7 <= 6000) print $4}' | sort -u | wc -l
    7985
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 >= 6001 && $7 <= 7000) print $4}' | sort -u | wc -l
    5875
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 >= 7001 && $7 <= 8000) print $4}' | sort -u | wc -l
    4368
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 >= 8001 && $7 <= 9000) print $4}' | sort -u | wc -l
    3525
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 >= 9001 && $7 <= 10000) print $4}' | sort -u | wc -l
    2738
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk -F
'\t' '{if ($7 > 10000) print $4}' | sort -u | wc -l
   31473
```

TASK C (2)

Code:

```
#!/bin/bash

# Define the path to the uncompressed data file
data_file="$HOME/Desktop/corona_tweets.csv.gz"

# Define the output CSV file
file="TaskC2output.csv"

# Define the ranges
ranges=("<=1000" "1001-2000" "2001-3000" "3001-4000" "4001-5000"
"5001-6000" "6001-7000" "7001-8000" "8001-9000" "9001-10000"
">10000")

# Create or overwrite the output CSV file and write the header
echo "Range,Number of Twitter Users" > "$file"

# Iterate over the ranges
for range in "${ranges[@]}"; do
    # Perform the count using awk, sort, and wc
    if [ "$range" == "<=1000" ]; then
        count=$(gzcat "$data_file" | awk -F'\t' '{if ($7 <= 1000)
print $4}' | sort -u | wc -l)
    elif [ "$range" == ">10000" ]; then
        count=$(gzcat "$data_file" | awk -F'\t' '{if ($7 > 10000)
print $4}' | sort -u | wc -l)
    else
        lower=$(echo "$range" | cut -d'-' -f1)
        upper=$(echo "$range" | cut -d'-' -f2)
        count=$(gzcat "$data_file" | awk -F'\t' -v lower="$lower" -v
upper="$upper" '{if ($7 >= lower && $7 <= upper) print $4}' | sort -
u | wc -l)
    fi

    # Append the range and count to the output CSV file
    echo "$range,$count" >> "$file"
done
```

Terminal:

```
bash twitter_followers.sh
(base) reneeyeo@Renees-MacBook-Pro Desktop % bash twitter_followers.sh
```

Output:

	A	B	C	D
1	Range	Number of Twitter Users		
2	<=1000	455758		
3	1001-2000	68574		
4	2001-3000	31281		
5	3001-4000	18803		
6	4001-5000	11699		
7	5001-6000	7985		
8	6001-7000	5875		
9	7001-8000	4368		
10	8001-9000	3525		
11	9001-10000	2738		
12	>10000	31473		
13				
14				
15				

TASK C (3)

Code:

```
getwd()
setwd("/Users/reneeyeo/Desktop")

# Task C3
# Read the output file into a data frame
data1 <- read.csv("TaskC2output.csv",header=TRUE)

# Print the data frame
print(data1)
```

Output:

```
> getwd()
[1] "/Users/reneeyeo/Desktop"
> setwd("/Users/reneeyeo/Desktop")
> # Task C3
> # Read the output file into a data frame
> data1 <- read.csv("TaskC2output.csv",header=TRUE)
> # Print the data frame
> print(data1)
        Range Number.of.Twitter.Users
1      <=1000                 455758
2   1001-2000                  68574
3   2001-3000                  31281
4   3001-4000                  18803
5   4001-5000                  11699
6   5001-6000                   7985
7   6001-7000                   5875
8   7001-8000                   4368
9   8001-9000                   3525
10 9001-10000                   2738
11     >10000                  31473
```

TASK C (4)

Code:

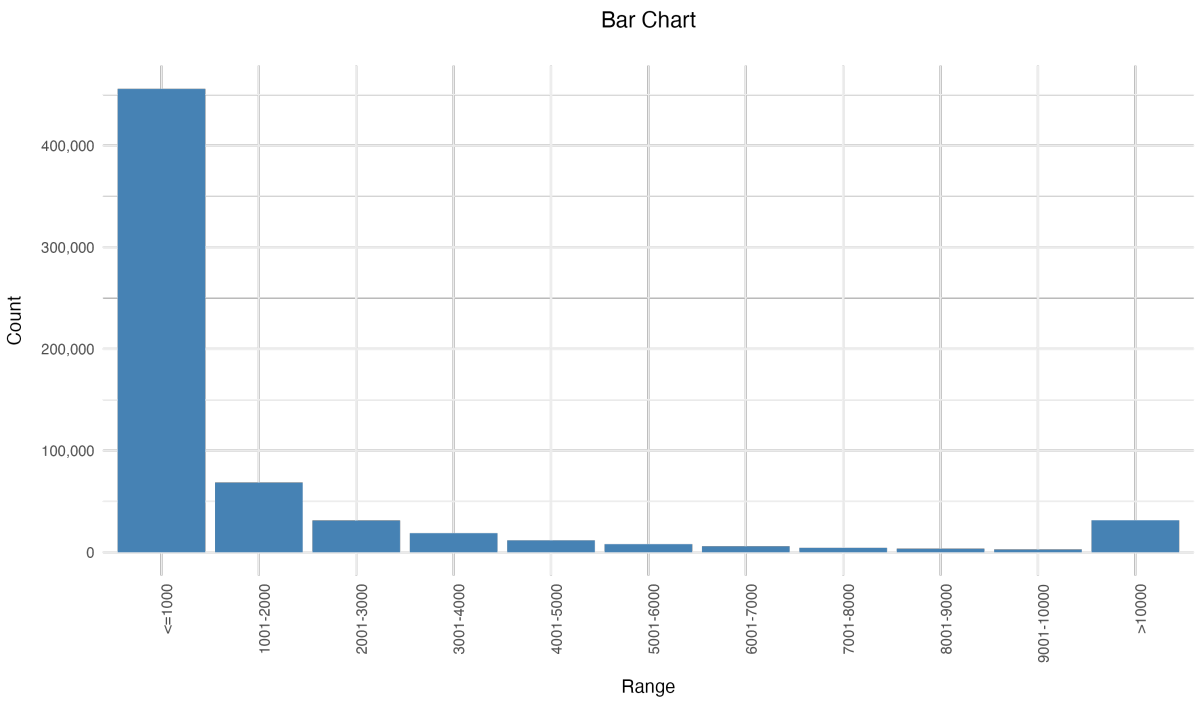
```
# Task C4
library(ggplot2)

ggplot(data1, aes(x = Range, y = Number.of.Twitter.Users)) +
  geom_bar(stat = "identity", fill = "#4682B4") +
  labs(x = "Range", y = "Count") +
  ggtitle("Bar Chart") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 90, hjust = 1),
    axis.title.x = element_text(margin = margin(t = 10)),  # Adjust
distance for x-axis label
    axis.title.y = element_text(margin = margin(r = 10)),  # Adjust
distance for y-axis label
    plot.title = element_text(hjust = 0.5, margin = margin(b = 20)),
# Center the title and margin distance
  ) +
  scale_x_discrete(limits = c("<=1000", "1001-2000", "2001-3000",
"3001-4000", "4001-5000","5001-6000", "6001-7000", "7001-8000",
"8001-9000", "9001-10000", ">10000")) +
  scale_y_continuous(labels = scales::comma)

# Save barchart as a PNG file
ggsave("TaskC4.png")
```

Output:

```
> # Task C4
> library(ggplot2)
> ggplot(data1, aes(x = Range, y = Number.of.Twitter.Users)) +
+   geom_bar(stat = "identity", fill = "#4682B4") +
+   labs(x = "Range", y = "Count") +
+   ggtitle("Bar Chart") +
+   theme_minimal() +
+   theme(
+     axis.text.x = element_text(angle = 90, hjust = 1),
+     axis.title.x = element_text(margin = margin(t = 10)),  # Adjust distance for x-axis label
+     axis.title.y = element_text(margin = margin(r = 10)),  # Adjust distance for y-axis label
+     plot.title = element_text(hjust = 0.5, margin = margin(b = 20)),  # Center the title and margin distance
+   ) +
+   scale_x_discrete(limits = c("<=1000", "1001-2000", "2001-3000", "3001-4000", "4001-5000","5001-6000", "6001-7000", "7001-8000", "8001-9000", "9001-10000", ">10000")) +
+   scale_y_continuous(labels = scales::comma)
> # Save barchart as a PNG file
> ggsave("Task(4.png")
Saving 9.96 x 5.82 in image
```



TASK D (1)

Single line code:

```
gzcat corona_tweets.csv.gz | awk '!/RT @/' | gzip >
filtered_tweets.gz
```

Explanation:

- `gzcat corona_tweets.csv.gz` is to read the compressed file 'corona_tweets.csv.gz' and outputs the uncompressed contents

- `awk` is a command of a versatile text processing tool

- `!` represent 'not', `'!/RT @/'` means it excludes the line that contains the substring 'RT @'

- `gzip` means the output taken from the previous codes is being compressed

- `>` symbol means to redirect the compressed output to a new file named 'filtered_tweets.gz' (`gzip > filtered_tweets.gz`)

Output:

```
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat corona_tweets.csv.gz | awk '!
 /RT @/' | gzip > filtered_tweets.gz
```

TASK D (2)

Step 1: Group them by the number of followers into their following ranges (11 codes)

```
gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 <= 1000) print $4}' |
sort -u

gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 >= 1001 && $7 <=
2000) print $4}' | sort -u

gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 >= 2001 && $7 <=
3000) print $4}' | sort -u

gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 >= 3001 && $7 <=
4000) print $4}' | sort -u

gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 >= 4001 && $7 <=
5000) print $4}' | sort -u

gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 >= 5001 && $7 <=
6000) print $4}' | sort -u

gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 >= 6001 && $7 <=
7000) print $4}' | sort -u
```

```
gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 >= 7001 && $7 <=
8000) print $4}' | sort -u

gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 >= 8001 && $7 <=
9000) print $4}' | sort -u

gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 >= 9001 && $7 <=
10000) print $4}' | sort -u

gzcat filtered_tweets.gz | awk -F'\t' '{if ($7 > 10000) print $4}' |
sort -u
```

Based on the above codes, I will output the number of lines that exists in the groupings:

```
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 <= 1000) print $4}' | sort -u | wc -l
  142249
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 >= 1001 && $7 <= 2000) print $4}' | sort -u | wc -l
   24039
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 >= 2001 && $7 <= 3000) print $4}' | sort -u | wc -l
   11771
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 >= 3001 && $7 <= 4000) print $4}' | sort -u | wc -l
    7348
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 >= 4001 && $7 <= 5000) print $4}' | sort -u | wc -l
    4726
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 >= 5001 && $7 <= 6000) print $4}' | sort -u | wc -l
    3462
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 >= 6001 && $7 <= 7000) print $4}' | sort -u | wc -l
    2498
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 >= 7001 && $7 <= 8000) print $4}' | sort -u | wc -l
    1908
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 >= 8001 && $7 <= 9000) print $4}' | sort -u | wc -l
    1611
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 >= 9001 && $7 <= 10000) print $4}' | sort -u | wc -l
    1279
(base) reneeyeo@Renees-MacBook-Pro Desktop % gzcat filtered_tweets.gz | awk -F'\
t' '{if ($7 > 10000) print $4}' | sort -u | wc -l
   17017
```

Step 2: Modify the codes and include them in a shell script to output a CSV file

Code:

```
#!/bin/bash

# Define the path to the uncompressed data file
data_file="$HOME/Desktop/filtered_tweets.gz"

# Define the output CSV file
file="TaskD2output.csv"

# Define the ranges
ranges=("<=1000" "1001-2000" "2001-3000" "3001-4000" "4001-5000"
"5001-6000" "6001-7000" "7001-8000" "8001-9000" "9001-10000"
">10000")

# Create or overwrite the output CSV file and write the header
echo "Range,Number of Twitter Users" > "$file"

# Iterate over the ranges
for range in "${ranges[@]}"; do
    # Perform the count using awk, sort, and wc
    if [ "$range" == "<=1000" ]; then
        count=$(gzcat "$data_file" | awk -F'\t' '{if ($7 <= 1000)
print $4}' | sort -u | wc -l)
    elif [ "$range" == ">10000" ]; then
        count=$(gzcat "$data_file" | awk -F'\t' '{if ($7 > 10000)
print $4}' | sort -u | wc -l)
    else
        lower=$(echo "$range" | cut -d'-' -f1)
        upper=$(echo "$range" | cut -d'-' -f2)
        count=$(gzcat "$data_file" | awk -F'\t' -v lower="$lower" -v
upper="$upper" '{if ($7 >= lower && $7 <= upper) print $4}' | sort -
u | wc -l)
    fi

    # Append the range and count to the output CSV file
    echo "$range,$count" >> "$file"
done
```

Terminal:

```
bash filtered_tweets.sh
(base) reneeyeo@Renees-MacBook-Pro Desktop % bash filtered_tweets.sh
```

Output:

	A	B	C
1	Range	Number of Twitter Users	
2	<=1000	142249	
3	1001-2000	24039	
4	2001-3000	11771	
5	3001-4000	7348	
6	4001-5000	4726	
7	5001-6000	3462	
8	6001-7000	2498	
9	7001-8000	1908	
.0	8001-9000	1611	
.1	9001-10000	1279	
.2	>10000	17017	
.3			
.4			

Step 3: By using the output above, read it in R

Code:

```
# Task D2
# Read the output file into a data frame
data2 <- read.csv("TaskD2output.csv",header=TRUE)

# Print the data frame
print(data2)
```

Output:

```
> # Task D2
> # Read the output file into a data frame
> data2 <- read.csv("TaskD2output.csv",header=TRUE)
> # Print the data frame
> print(data2)
        Range Number.of.Twitter.Users
1      <=1000                 142249
2   1001-2000                  24039
3   2001-3000                  11771
4   3001-4000                   7348
5   4001-5000                   4726
6   5001-6000                   3462
7   6001-7000                   2498
8   7001-8000                   1908
9   8001-9000                   1611
10 9001-10000                   1279
11     >10000                  17017
>
```

TASK D (3)

Code:

```
# Task D3
library(ggplot2)

# Rename the columns in each data frame
data1_renamed <- data1
data2_renamed <- data2
names(data1_renamed)[names(data1_renamed) ==
"Number.of.Twitter.Users"] <- "Number.of.Twitter.Users.RT"
names(data2_renamed)[names(data2_renamed) ==
"Number.of.Twitter.Users"] <- "Number.of.Twitter.Users.NoRT"

# Check whether is it renamed
print(data1_renamed)
print(data2_renamed)

# Merge the two data frames by the "Range" column
merged_data <- merge(data1_renamed, data2_renamed, by = "Range")

# Reshape the data to long format
merged_data_long <- tidyr::pivot_longer(merged_data, cols =
c(Number.of.Twitter.Users.RT, Number.of.Twitter.Users.NoRT),
names_to = "Group", values_to = "Number_of_Followers")

# Create the side-by-side bar chart
ggplot(merged_data_long, aes(x = Range, y = Number_of_Followers,
fill = Group)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Range", y = "Count", fill = "Group") +
  theme_minimal() +
  theme(legend.position = "top",
        axis.text.x = element_text(angle = 90, hjust = 1),
        axis.title.x = element_text(margin = margin(t = 10)),  #
Adjust distance for x-axis label
        axis.title.y = element_text(margin = margin(r = 10)),  #
Adjust distance for y-axis label
        plot.title = element_text(hjust = 0.5, margin = margin(b =
20)),  # Center the title and margin distance
        ) +
  scale_x_discrete(limits = c("<=1000", "1001-2000", "2001-3000",
"3001-4000", "4001-5000","5001-6000", "6001-7000", "7001-8000",
"8001-9000", "9001-10000", ">10000")) +
  scale_y_continuous(labels = scales::comma)

# Save barchart as a PNG file
ggsave("TaskD3.png")
```

Output:

```
> # Task D3
> library(ggplot2)
> # Rename the columns in each data frame
> data1_renamed <- data1
> data2_renamed <- data2
> names(data1_renamed)[names(data1_renamed) == "Number.of.Twitter.Users"] <- "Number.of.Twitter.Users.RT"
> names(data2_renamed)[names(data2_renamed) == "Number.of.Twitter.Users"] <- "Number.of.Twitter.Users.NoRT"
> # Check whether is it renamed
> print(data1_renamed)
       Range Number.of.Twitter.Users.RT
1      <=1000                    455758
2   1001-2000                     68574
3   2001-3000                     31281
4   3001-4000                     18803
5   4001-5000                     11699
6   5001-6000                      7985
7   6001-7000                      5875
8   7001-8000                      4368
9   8001-9000                      3525
10 9001-10000                      2738
11     >10000                     31473
> print(data2_renamed)
       Range Number.of.Twitter.Users.NoRT
1      <=1000                     142249
2   1001-2000                      24039
3   2001-3000                      11771
4   3001-4000                       7348
5   4001-5000                       4726
6   5001-6000                       3462
7   6001-7000                       2498
8   7001-8000                       1908
9   8001-9000                       1611
10 9001-10000                       1279
11     >10000                      17017
```

```
> # Merge the two data frames by the "Range" column
> merged_data <- merge(data1_renamed, data2_renamed, by = "Range")
> # Reshape the data to long format
> merged_data_long <- tidyr::pivot_longer(merged_data, cols = c(Number.of.Twitter.Users.RT, Number.of.Twitter.Users.NoRT), names_to = "Group", values_to = "Number_of_Followers")
> # Create the side-by-side bar chart
> ggplot(merged_data_long, aes(x = Range, y = Number_of_Followers, fill = Group)) +
+   geom_bar(stat = "identity", position = "dodge") +
+   labs(x = "Range", y = "Count", fill = "Group") +
+   theme_minimal() +
+   theme(legend.position = "top",
+         axis.text.x = element_text(angle = 90, hjust = 1),
+         axis.title.x = element_text(margin = margin(t = 10)),  # Adjust distance for x-axis label
+         axis.title.y = element_text(margin = margin(r = 10)),  # Adjust distance for y-axis label
+         plot.title = element_text(hjust = 0.5, margin = margin(b = 20)),  # Center the title and margin distance
+         ) +
+   scale_x_discrete(limits = c("<=1000", "1001-2000", "2001-3000", "3001-4000", "4001-5000","5001-6000", "6001-7000", "7001-8000", "8001-9000", "9001-10000", ">10000")) +
+   scale_y_continuous(labels = scales::comma)
> # Save barchart as a PNG file
> ggsave("TaskD3.png")
Saving 7.42 x 5.82 in image
```