Exercise 2 starter

Load data

Load the following data: + applications from `app_data_sample.parquet` + edges from `edges_sample.csv`

```
# change to your own path!
data_path <- "/Users/kaz/Desktop/MMA - WINTER Code/"
applications <- read_feather(paste0(data_path,"app_data_starter.feather"))

applications
```

```
## # A tibble: 2,018,477 x 21
##    application_number filing_date examiner_name_last examiner_name_first
##    <chr>              <date>      <chr>              <chr>
##  1 08284457           2000-01-26  HOWARD             JACQUELINE
##  2 08413193           2000-10-11  YILDIRIM           BEKIR
##  3 08531853           2000-05-17  HAMILTON           CYNTHIA
##  4 08637752           2001-07-20  MOSHER             MARY
##  5 08682726           2000-04-10  BARR               MICHAEL
##  6 08687412           2000-04-28  GRAY               LINDA
##  7 08716371           2004-01-26  MCMILLIAN          KARA
##  8 08765941           2000-06-23  FORD               VANESSA
##  9 08776818           2000-02-04  STRZELECKA         TERESA
## 10 08809677           2002-02-20  KIM                SUN
## # i 2,018,467 more rows
## # i 17 more variables: examiner_name_middle <chr>, examiner_id <dbl>,
## #   examiner_art_unit <dbl>, uspc_class <chr>, uspc_subclass <chr>,
## #   patent_number <chr>, patent_issue_date <date>, abandon_date <date>,
## #   disposal_type <chr>, appl_status_code <dbl>, appl_status_date <chr>,
## #   tc <dbl>, gender <chr>, race <chr>, earliest_date <date>,
## #   latest_date <date>, tenure_days <dbl>
```

Get gender for examiners

We'll get gender based on the first name of the examiner, which is recorded in the field `examiner_name_first`. We'll use library `gender` for that, relying on a modified version of their own example.

Note that there are over 2 million records in the applications table – that's because there are many records for each examiner, as many as the number of applications that examiner worked on during this time frame. Our first step therefore is to get all *unique* names in a separate list `examiner_names`. We will then guess gender for each one and will join this table back to the original dataset. So, let's get names without repetition:

```
library(gender)
#install_genderdata_package() # only run this line the first time you use the package, to get data for
```

```
# get a list of first names without repetitions
examiner_names <- applications %>%
  distinct(examiner_name_first)

examiner_names
```

```
## # A tibble: 2,595 x 1
##    examiner_name_first
##    <chr>
##  1 JACQUELINE
##  2 BEKIR
##  3 CYNTHIA
##  4 MARY
##  5 MICHAEL
##  6 LINDA
##  7 KARA
##  8 VANESSA
##  9 TERESA
## 10 SUN
## # i 2,585 more rows
```

Now let's use function `gender()` as shown in the example for the package to attach a gender and probability to each name and put the results into the table `examiner_names_gender`

```
# get a table of names and gender

examiner_names_gender <- examiner_names %>%
  do(results = gender(.$examiner_name_first, method = "ssa")) %>%
  unnest(cols = c(results), keep_empty = TRUE) %>%
  select(
    examiner_name_first = name,
    gender,
    proportion_female
  )

examiner_names_gender
```

```
## # A tibble: 1,822 x 3
##    examiner_name_first gender proportion_female
##    <chr>               <chr>              <dbl>
##  1 AARON               male              0.0082
##  2 ABDEL               male              0
##  3 ABDOU               male              0
##  4 ABDUL               male              0
##  5 ABDULHAKIM          male              0
##  6 ABDULLAH            male              0
##  7 ABDULLAHI           male              0
##  8 ABIGAIL             female            0.998
##  9 ABIMBOLA            female            0.944
## 10 ABRAHAM             male              0.0031
## # i 1,812 more rows
```

Finally, let's join that table back to our original applications data and discard the temporary tables we have just created to reduce clutter in our environment.

```
# remove extra colums from the gender table
examiner_names_gender <- examiner_names_gender %>%
  select(examiner_name_first, gender)

# joining gender back to the dataset
applications <- applications %>%
  left_join(examiner_names_gender, by = "examiner_name_first")

# cleaning up
rm(examiner_names)
rm(examiner_names_gender)
gc()
```

```
##              used  (Mb) gc trigger  (Mb) limit (Mb)  max used  (Mb)
## Ncells  4474566 239.0    7448462 397.8         NA   4923668 263.0
## Vcells 59553815 454.4  119586570 912.4      16384 103999063 793.5
```

```
# colsum na
colSums(is.na(applications))
```

```
##   application_number          filing_date   examiner_name_last
##                    0                    0                    0
##  examiner_name_first examiner_name_middle          examiner_id
##                    0               471770                 9229
##    examiner_art_unit           uspc_class        uspc_subclass
##                    0                    4                 1677
##        patent_number    patent_issue_date         abandon_date
##               931651               931178              1417057
##        disposal_type     appl_status_code     appl_status_date
##                    0                 4609                 4610
##                   tc             gender.x                 race
##                    0               303859                    0
##        earliest_date          latest_date          tenure_days
##                    0                    0                    0
##             gender.y
##               303859
```

Guess the examiner's race

We'll now use package wru to estimate likely race of an examiner. Just like with gender, we'll get a list of unique names first, only now we are using surnames.

```
library(wru)

examiner_surnames <- applications %>%
  select(surname = examiner_name_last) %>%
  distinct()

examiner_surnames
```

```
## # A tibble: 3,806 x 1
##    surname
##    <chr>
##  1 HOWARD
##  2 YILDIRIM
##  3 HAMILTON
##  4 MOSHER
##  5 BARR
##  6 GRAY
##  7 MCMILLIAN
##  8 FORD
##  9 STRZELECKA
## 10 KIM
## # i 3,796 more rows
```

We'll follow the instructions for the package outlined here https://github.com/kosukeimai/wru.

```
examiner_race <- predict_race(voter.file = examiner_surnames, surname.only = T) %>%
  as_tibble()
```

```
## Warning: Unknown or uninitialised column: ‘state‘.

## Proceeding with last name predictions...

## i All local files already up-to-date!

## 701 (18.4%) individuals’ last names were not matched.
```

```
examiner_race
```

```
## # A tibble: 3,806 x 6
##    surname    pred.whi pred.bla pred.his pred.asi pred.oth
##    <chr>         <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
##  1 HOWARD       0.597   0.295    0.0275   0.00690   0.0741
##  2 YILDIRIM     0.807   0.0273   0.0694   0.0165    0.0798
##  3 HAMILTON     0.656   0.239    0.0286   0.00750   0.0692
##  4 MOSHER       0.915   0.00425  0.0291   0.00917   0.0427
##  5 BARR         0.784   0.120    0.0268   0.00830   0.0615
##  6 GRAY         0.640   0.252    0.0281   0.00748   0.0724
##  7 MCMILLIAN    0.322   0.554    0.0212   0.00340   0.0995
##  8 FORD         0.576   0.320    0.0275   0.00621   0.0697
##  9 STRZELECKA   0.472   0.171    0.220    0.0825    0.0543
## 10 KIM          0.0169  0.00282  0.00546  0.943     0.0319
## # i 3,796 more rows
```

As you can see, we get probabilities across five broad US Census categories: white, black, Hispanic, Asian and other. (Some of you may correctly point out that Hispanic is not a race category in the US Census, but these are the limitations of this package.)

Our final step here is to pick the race category that has the highest probability for each last name and then join the table back to the main applications table. See this example for comparing values across columns: https://www.tidyverse.org/blog/2020/04/dplyr-1-0-0-rowwise/. And this one for `case_when()` function: https://dplyr.tidyverse.org/reference/case_when.html.

```
examiner_race <- examiner_race %>%
  mutate(max_race_p = pmax(pred.asi, pred.bla, pred.his, pred.oth, pred.whi)) %>%
  mutate(race = case_when(
    max_race_p == pred.asi ~ "Asian",
    max_race_p == pred.bla ~ "black",
    max_race_p == pred.his ~ "Hispanic",
    max_race_p == pred.oth ~ "other",
    max_race_p == pred.whi ~ "white",
    TRUE ~ NA_character_
  ))

examiner_race
```

```
## # A tibble: 3,806 x 8
##    surname    pred.whi pred.bla pred.his pred.asi pred.oth max_race_p race
##    <chr>         <dbl>    <dbl>    <dbl>    <dbl>    <dbl>      <dbl> <chr>
##  1 HOWARD       0.597   0.295    0.0275   0.00690  0.0741      0.597 white
##  2 YILDIRIM     0.807   0.0273   0.0694   0.0165   0.0798      0.807 white
##  3 HAMILTON     0.656   0.239    0.0286   0.00750  0.0692      0.656 white
##  4 MOSHER       0.915   0.00425  0.0291   0.00917  0.0427      0.915 white
##  5 BARR         0.784   0.120    0.0268   0.00830  0.0615      0.784 white
##  6 GRAY         0.640   0.252    0.0281   0.00748  0.0724      0.640 white
##  7 MCMILLIAN    0.322   0.554    0.0212   0.00340  0.0995      0.554 black
##  8 FORD         0.576   0.320    0.0275   0.00621  0.0697      0.576 white
##  9 STRZELECKA   0.472   0.171    0.220    0.0825   0.0543      0.472 white
## 10 KIM          0.0169  0.00282  0.00546  0.943    0.0319      0.943 Asian
## # i 3,796 more rows
```

Let's join the data back to the applications table.

```
# removing extra columns
examiner_race <- examiner_race %>%
  select(surname,race)

applications <- applications %>%
  left_join(examiner_race, by = c("examiner_name_last" = "surname"))

rm(examiner_race)
rm(examiner_surnames)
gc()
```

```
##            used  (Mb) gc trigger  (Mb) limit (Mb)  max used  (Mb)
## Ncells  4660653 249.0    7448462 397.8         NA   6801083 363.3
## Vcells 61897961 472.3  119586570 912.4      16384 119334815 910.5
```

Examiner's tenure

To figure out the timespan for which we observe each examiner in the applications data, let's find the first and the last observed date for each examiner. We'll first get examiner IDs and application dates in a separate table, for ease of manipulation. We'll keep examiner ID (the field `examiner_id`), and earliest and latest dates for each application (`filing_date` and `appl_status_date` respectively). We'll use functions in package `lubridate` to work with date and time values.

```
library(lubridate) # to work with dates

examiner_dates <- applications %>%
  select(examiner_id, filing_date, appl_status_date)

examiner_dates
```

```
## # A tibble: 2,018,477 x 3
##    examiner_id filing_date appl_status_date
##          <dbl> <date>      <chr>
##  1       96082 2000-01-26  30jan2003 00:00:00
##  2       87678 2000-10-11  27sep2010 00:00:00
##  3       63213 2000-05-17  30mar2009 00:00:00
##  4       73788 2001-07-20  07sep2009 00:00:00
##  5       77294 2000-04-10  19apr2001 00:00:00
##  6       68606 2000-04-28  16jul2001 00:00:00
##  7       89557 2004-01-26  15may2017 00:00:00
##  8       97543 2000-06-23  03apr2002 00:00:00
##  9       98714 2000-02-04  27nov2002 00:00:00
## 10       65530 2002-02-20  23mar2009 00:00:00
## # i 2,018,467 more rows
```

The dates look inconsistent in terms of formatting. Let's make them consistent. We'll create new variables `start_date` and `end_date`.

```
examiner_dates <- examiner_dates %>%
  mutate(start_date = ymd(filing_date), end_date = as_date(dmy_hms(appl_status_date)))
```

Let's now identify the earliest and the latest date for each examiner and calculate the difference in days, which is their tenure in the organization.

```
examiner_dates <- examiner_dates %>%
  group_by(examiner_id) %>%
  summarise(
    earliest_date = min(start_date, na.rm = TRUE),
    latest_date = max(end_date, na.rm = TRUE),
    tenure_days = interval(earliest_date, latest_date) %/% days(1)
    ) %>%
  filter(year(latest_date)<2018)

examiner_dates
```

```
## # A tibble: 5,625 x 4
##    examiner_id earliest_date latest_date tenure_days
##          <dbl> <date>        <date>            <dbl>
##  1       59012 2004-07-28    2015-07-24         4013
##  2       59025 2009-10-26    2017-05-18         2761
##  3       59030 2005-12-12    2017-05-22         4179
##  4       59040 2007-09-11    2017-05-23         3542
##  5       59052 2001-08-21    2007-02-28         2017
##  6       59054 2000-11-10    2016-12-23         5887
##  7       59055 2004-11-02    2007-12-26         1149
##  8       59056 2000-03-24    2017-05-22         6268
##  9       59074 2000-01-31    2017-03-17         6255
## 10       59081 2011-04-21    2017-05-19         2220
## # i 5,615 more rows
```

Joining back to the applications data.

```
applications <- applications %>%
  left_join(examiner_dates, by = "examiner_id")

rm(examiner_dates)
gc()
```

```
##            used  (Mb) gc trigger   (Mb) limit (Mb)  max used  (Mb)
## Ncells  4669144 249.4    7448462  397.8         NA   7448462 397.8
## Vcells 67973006 518.6  145755310 1112.1      16384 121396092 926.2
```

Save file as processed variables, to skip these steps in the following exercises.

```
write_feather(applications, paste0(data_path,"app_data_starter_coded.feather"))
```

Rest of the exercise

```
# load data
data_path <- "/Users/kaz/Desktop/MMA - WINTER Code/"
applications <- read_feather(paste0(data_path,"app_data_starter_coded.feather"))
```

Create Variables

```
library(tidyverse)
library(lubridate)

# Convert filing_date to Date format and create a quarter variable
applications$filing_date <- as.Date(applications$filing_date)
applications$quarter <- paste0(year(applications$filing_date), "/", quarter(applications$filing_date))

# Aggregate applications by quarter and examiner
applications <- applications %>%
  group_by(quarter, examiner_id) %>%
  mutate(new_applications = n_distinct(application_number)) %>%
  ungroup()

applications <- applications %>%
  group_by(quarter, examiner_id) %>%
  mutate(ISSUED_applications = sum(disposal_type == "ISS" & !duplicated(application_number)))

applications <- applications %>%
  group_by(quarter, examiner_id) %>%
  mutate(abn_applications = sum(disposal_type == "ABN" & !duplicated(application_number)))

applications <- applications %>%
  group_by(quarter, examiner_id) %>%
  mutate(PEN_applications = sum(disposal_type == "PEND" & !duplicated(application_number)))

applications <- applications %>%
  group_by(quarter,examiner_art_unit) %>%
  mutate(examiner_art_unit_num =  n_distinct(examiner_id))%>%
  ungroup()

applications <- applications %>%
  group_by(quarter, examiner_art_unit) %>%
  mutate(women_in_art_unit  = sum(gender.y == "female" & !duplicated(examiner_id)))

applications <- applications %>%
  group_by(quarter, examiner_art_unit) %>%
  mutate(Asian_in_art_unit  = sum(race.y == "Asian" & !duplicated(examiner_id)))

applications <- applications %>%
  group_by(quarter, examiner_art_unit) %>%
  mutate(Black_in_art_unit  = sum(race.y == "black" & !duplicated(examiner_id)))


applications <- applications %>%
  group_by(quarter, examiner_art_unit) %>%
  mutate(Hispanic_in_art_unit  = sum(race.y == "Hispanic" & !duplicated(examiner_id)))

applications <- applications %>%
  group_by(quarter, examiner_art_unit) %>%
  mutate(Other_in_art_unit  = sum(race.y == "other" & !duplicated(examiner_id)))

applications <- applications %>%
  group_by(quarter, examiner_art_unit) %>%
  mutate(White_in_art_unit  = sum(race.y == "white" & !duplicated(examiner_id)))
```

Creating separation and au indicator

```
# sort by examiner_id and quarter
applications <- applications %>%
  arrange(examiner_id, quarter)
```

- Drop duplicated columns after merging

```
applications_selected <- applications %>%
  select(
    application_number,
    examiner_id,
    examiner_name_first,
    examiner_name_middle,
    examiner_name_last,
    tc,
    quarter,
    new_applications,
    ISSUED_applications,
    abn_applications,
    PEN_applications,
    examiner_art_unit,
    women_in_art_unit,
    Asian_in_art_unit,
    Black_in_art_unit,
    Other_in_art_unit,
    White_in_art_unit,
    ends_with(".x")  # Select columns that end with '_x'
  ) %>%
  rename_with(~ str_remove(., ".x"), ends_with(".x"))  # Remove the '_x' suffix
```

in order to add separation we must know when each of employee's max quarter and compare it to max quarter in the dataset

```
# find the latest time quarter for each examiner
applications_selected %>%
  group_by(examiner_id) %>%
  mutate(max_quarter = max(quarter))
```

```
## # A tibble: 2,018,477 x 23
## # Groups:   examiner_id [5,649]
##    application_number examiner_id examiner_name_first examiner_name_middle
##    <chr>                    <dbl> <chr>               <chr>
##  1 10901322                 59012 ALBERT              <NA>
##  2 10595152                 59012 ALBERT              <NA>
##  3 10578100                 59012 ALBERT              <NA>
##  4 11396590                 59012 ALBERT              <NA>
##  5 11427049                 59012 ALBERT              <NA>
##  6 11473554                 59012 ALBERT              <NA>
##  7 10593607                 59012 ALBERT              <NA>
##  8 11466665                 59012 ALBERT              <NA>
##  9 11516177                 59012 ALBERT              <NA>
## 10 11520094                 59012 ALBERT              <NA>
## # i 2,018,467 more rows
## # i 19 more variables: examiner_name_last <chr>, tc <dbl>, quarter <chr>,
## #   new_applications <int>, ISSUED_applications <int>, abn_applications <int>,
## #   PEN_applications <int>, examiner_art_unit <dbl>, women_in_art_unit <int>,
## #   Asian_in_art_unit <int>, Black_in_art_unit <int>, Other_in_art_unit <int>,
## #   White_in_art_unit <int>, gender <chr>, race <chr>, earliest_date <date>,
## #   latest_date <date>, tenure_days <dbl>, max_quarter <chr>
```

```
# unique quarters values and count  (when is the latest quarter in the dataset?)
applications_selected %>%
  group_by(quarter) %>%
  summarise(n = n_distinct(examiner_id)) %>%
  arrange(desc(quarter)) %>% head(5)
```

```
## # A tibble: 5 x 2
##   quarter     n
##   <chr>   <int>
## 1 2017/2     68
## 2 2017/1   1866
## 3 2016/4   2728
## 4 2016/3   2879
## 5 2016/2   3004
```

```
overall_max_quarter <- "2017/1"

# Create the separation_indicator
applications_selected <- applications_selected %>%
  group_by(examiner_id) %>%
  mutate(max_quarter_examiner = max(quarter)) %>%
  ungroup() %>%
  mutate(separation_indicator = if_else(max_quarter_examiner < overall_max_quarter, 1, 0))

applications_selected
```

```
## # A tibble: 2,018,477 x 24
##    application_number examiner_id examiner_name_first examiner_name_middle
##    <chr>                    <dbl> <chr>               <chr>
##  1 10901322                 59012 ALBERT              <NA>
##  2 10595152                 59012 ALBERT              <NA>
##  3 10578100                 59012 ALBERT              <NA>
##  4 11396590                 59012 ALBERT              <NA>
##  5 11427049                 59012 ALBERT              <NA>
##  6 11473554                 59012 ALBERT              <NA>
##  7 10593607                 59012 ALBERT              <NA>
##  8 11466665                 59012 ALBERT              <NA>
##  9 11516177                 59012 ALBERT              <NA>
## 10 11520094                 59012 ALBERT              <NA>
## # i 2,018,467 more rows
## # i 20 more variables: examiner_name_last <chr>, tc <dbl>, quarter <chr>,
## #   new_applications <int>, ISSUED_applications <int>, abn_applications <int>,
## #   PEN_applications <int>, examiner_art_unit <dbl>, women_in_art_unit <int>,
## #   Asian_in_art_unit <int>, Black_in_art_unit <int>, Other_in_art_unit <int>,
## #   White_in_art_unit <int>, gender <chr>, race <chr>, earliest_date <date>,
## #   latest_date <date>, tenure_days <dbl>, max_quarter_examiner <chr>, ...
```

- Our separation data shoudl look like 0 0 0 0 0 0 1 -> this one indicates that the employee has left the company
- if 0 0 0 0 0 0 0 -> this one indicates that the employee is still working for the company

Add AU move Indicator

```
applications_selected <- applications_selected %>%
  group_by(examiner_id) %>%
  mutate(au_move_indicator = if_else(examiner_art_unit != lag(examiner_art_unit), 1, 0)) %>%
  ungroup()

# Fill NA for the au_move_indicator
```

```
applications_selected <- applications_selected %>%
  mutate(au_move_indicator = if_else(is.na(au_move_indicator), 0, au_move_indicator))

applications_selected
```

```
## # A tibble: 2,018,477 x 25
##    application_number examiner_id examiner_name_first examiner_name_middle
##    <chr>                    <dbl> <chr>               <chr>
##  1 10901322                 59012 ALBERT              <NA>
##  2 10595152                 59012 ALBERT              <NA>
##  3 10578100                 59012 ALBERT              <NA>
##  4 11396590                 59012 ALBERT              <NA>
##  5 11427049                 59012 ALBERT              <NA>
##  6 11473554                 59012 ALBERT              <NA>
##  7 10593607                 59012 ALBERT              <NA>
##  8 11466665                 59012 ALBERT              <NA>
##  9 11516177                 59012 ALBERT              <NA>
## 10 11520094                 59012 ALBERT              <NA>
## # i 2,018,467 more rows
## # i 21 more variables: examiner_name_last <chr>, tc <dbl>, quarter <chr>,
## #   new_applications <int>, ISSUED_applications <int>, abn_applications <int>,
## #   PEN_applications <int>, examiner_art_unit <dbl>, women_in_art_unit <int>,
## #   Asian_in_art_unit <int>, Black_in_art_unit <int>, Other_in_art_unit <int>,
## #   White_in_art_unit <int>, gender <chr>, race <chr>, earliest_date <date>,
## #   latest_date <date>, tenure_days <dbl>, max_quarter_examiner <chr>, ...
```

Not sure what we are expected to create here: some employees change art unit multiple times in one quarter (most likely because they have multiple ongoing projects). However, summing them would make the "indicator" greater than one. Yet, this sum should indicate how many times one moved art unit in one quarter.

Some other cleaning

```
# drop columns: assumed we don't need them anymore
applications_selected <- applications_selected %>%
  select(-c(max_quarter_examiner, earliest_date, latest_date, tc))

# fill NA for the woman_in_art_unit
applications_selected <- applications_selected %>%
  mutate(women_in_art_unit = if_else(is.na(women_in_art_unit), 0, women_in_art_unit))

# info about the dataset
glimpse(applications_selected)
```

```
## Rows: 2,018,477
## Columns: 21
## $ application_number   <chr> "10901322", "10595152", "10578100", "11396590", "~
## $ examiner_id          <dbl> 59012, 59012, 59012, 59012, 59012, 59012, 59012, ~
## $ examiner_name_first  <chr> "ALBERT", "ALBERT", "ALBERT", "ALBERT", "ALBERT",~
## $ examiner_name_middle <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ examiner_name_last   <chr> "HILTON", "HILTON", "HILTON", "HILTON", "HILTON",~
## $ quarter              <chr> "2004/3", "2006/1", "2006/2", "2006/2", "2006/2",~
## $ new_applications     <int> 1, 1, 4, 4, 4, 4, 5, 5, 5, 5, 5, 9, 9, 9, 9, 9, 9~
## $ ISSUED_applications  <int> 0, 1, 3, 3, 3, 3, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 4~
## $ abn_applications     <int> 1, 0, 1, 1, 1, 1, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5~
## $ PEN_applications     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ examiner_art_unit    <dbl> 1717, 1716, 1716, 1716, 1716, 1716, 1717, 1716, 1~
## $ women_in_art_unit    <dbl> 2, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 6, 0, 0, 6~
## $ Asian_in_art_unit    <int> 0, 3, 3, 3, 3, 3, 0, 3, 20, 3, 3, 20, 3, 3, 0, 0,~
## $ Black_in_art_unit    <int> 0, 0, 0, 0, 0, 0, 1, 0, 3, 0, 0, 2, 0, 0, 1, 1, 0~
## $ Other_in_art_unit    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ White_in_art_unit    <int> 7, 13, 13, 13, 13, 13, 8, 14, 66, 14, 14, 63, 13,~
## $ gender               <chr> "male", "male", "male", "male", "male", "male", "~
## $ race                 <chr> "white", "white", "white", "white", "white", "whi~
## $ tenure_days          <dbl> 4013, 4013, 4013, 4013, 4013, 4013, 4013, 4013, 4~
## $ separation_indicator <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ au_move_indicator    <dbl> 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1~
```

Create a quarterly aggregated panel dataset - how do we aggregate columns like number of race in art unit? because some examiner changes art unit within each quarter - again how should we deal with art unit column? -> next file

```
write_feather(applications_selected, paste0(data_path,"app_applications_starter_coded2.feather"))
```

“‘