# CPSC375Homework 5

Kenn Son, Hamid Suda, Vivian Truong

3/8/2022

1. Consider the two tables shown below called population and countyseats.

population:

```
state <- c("California", "California", "California", "California")
county <- c("Orange", "Orange", "Los Angeles", "Los Angeles")
year <- c(2000,2010,2000,2010)
pop <- c(2846289,3010232,3694820,3792621)
population <- data.frame(state, county, year, "Population" = pop)
x <- as_tibble(population)
x
```

```
## # A tibble: 4 x 4
##   state      county       year Population
##   <chr>      <chr>       <dbl>      <dbl>
## 1 California Orange       2000    2846289
## 2 California Orange       2010    3010232
## 3 California Los Angeles  2000    3694820
## 4 California Los Angeles  2010    3792621
```

countyseats:

```
statename <- c("California", "California", "California", "Oregon")
countyname <- c("Orange", "Los Angeles", "San Diego", "Wasco")
countyseat <- c("Santa Ana", "Los Angeles", "San Diego", "The Dalles")
countyseats <- data.frame(statename, countyname, countyseat)
y <- as_tibble(countyseats)
y
```

```
## # A tibble: 4 x 3
##   statename  countyname  countyseat
##   <chr>      <chr>       <chr>
## 1 California Orange      Santa Ana
## 2 California Los Angeles Los Angeles
## 3 California San Diego   San Diego
## 4 Oregon     Wasco       The Dalles
```

You should be able to calculate the output by hand though you may use R to check your answer. Draw the output table from the following operations (you should be able to calculate the output by hand though you may use R to check your answers).

a) population %>% inner_join(countyseats)
   - Error since we don't know what is being compared
b) population %>% inner_join(countyseats, by=c(state="statename"))

| state | county | year | population | countyname | countyseat |
|---|---|---|---|---|---|
| California | Orange | 2000 | 2846289 | Orange | Santa Ana |
| California | Orange | 2000 | 2846289 | Los Angeles | Los Angeles |
| California | Orange | 2000 | 2846289 | San Diego | San Diego |
| California | Orange | 2010 | 3010232 | Orange | Santa Ana |
| California | Orange | 2010 | 3010232 | Los Angeles | Los Angeles |
| California | Orange | 2010 | 3010232 | San Diego | San Diego |
| California | Los Angeles | 2000 | 3694820 | Orange | Santa Ana |
| California | Los Angeles | 2000 | 3694820 | Los Angeles | Los Angeles |
| California | Los Angeles | 2000 | 3694820 | San Diego | San Diego |
| California | Los Angeles | 2010 | 3792621 | Orange | Santa Ana |
| California | Los Angeles | 2010 | 3792621 | Los Angeles | Los Angeles |
| California | Los Angeles | 2010 | 3792621 | San Diego | San Diego |

c) population %>% inner_join(countyseats, by=c(state="statename", county="countyname"))

| state | county | year | population | countyseat |
|---|---|---|---|---|
| California | Orange | 2000 | 2846289 | Santa Ana |
| California | Orange | 2010 | 3010232 | Santa Ana |
| California | Los Angeles | 2000 | 3694820 | Los Angeles |
| California | Los Angeles | 2010 | 3792621 | Los Angeles |

d) population %>% inner_join(countyseats, by=c(state="statename", county="countyname", year="countyseat"))

| state | county | year | population | countyseat |
|---|---|---|---|---|
| California | Orange | 2000 | 2846289 | Santa Ana |
| California | Orange | 2010 | 3010232 | Santa Ana |
| California | Los Angeles | 2000 | 3694820 | Los Angeles |
| California | Los Angeles | 2010 | 3792621 | Los Angeles |

2. Consider the billboard dataset that is supplied with the tidyverse which shows the Billboard top 100 song rankings in the year 2000. Apply the tidyverse's data wrangling verbs to answer these questions. For each question, give only the code.

a) Show for each track, how many weeks it spent on the chart

```
billboard %>% select(-artist) %>% select(-date.entered) %>%
  pivot_longer(-track, names_to = 'Week',values_to = 'Place',values_drop_na = TRUE) %>%
  group_by(track) %>% summarize('Count'=n())
```

```
## # A tibble: 316 x 2
##    track                        Count
##    <chr>                        <int>
##  1 (Hot S**t) Country G...         34
##  2 3 Little Words              9
##  3 911                        19
##  4 A Country Boy Can Su...     3
##  5 A Little Gasoline           6
##  6 A Puro Dolor (Purest...    26
##  7 Aaron’s Party (Come ...    15
##  8 Absolutely (Story Of...    27
##  9 All Good?                   3
## 10 All The Small Things       23
## # ... with 306 more rows
```

b)List tracks in decreasing order of number of weeks spent on the chart

```
billboard %>% select(-artist) %>% select(-date.entered) %>%
  pivot_longer(-track, names_to = 'Week',values_to = 'Place',values_drop_na = TRUE) %>%
  group_by(track) %>% summarize('Count'=n()) %>% arrange(desc(Count))
```

```
## # A tibble: 316 x 2
##    track                   Count
##    <chr>                   <int>
##  1 Higher                     57
##  2 Amazed                     55
##  3 Breathe                    53
##  4 Kryptonite                 53
##  5 With Arms Wide Open        47
##  6 I Wanna Know               44
##  7 Everything You Want        41
##  8 Bent                       39
##  9 He Wasn’t Man Enough       37
## 10 (Hot S**t) Country G...    34
## # ... with 306 more rows
```

c)Show for each track, its top rank

```
billboard %>% select(-artist) %>% select(-date.entered) %>%
  pivot_longer(-track, names_to = 'Week',values_to = 'Place',values_drop_na = TRUE) %>%
  group_by(track) %>% summarise('TopRank
                                ' = min(Place))
```

```
## # A tibble: 316 x 2
##    track                   ‘TopRank\n                                ‘
##    <chr>                                                         <dbl>
##  1 (Hot S**t) Country G...                                           7
##  2 3 Little Words                                                   89
##  3 911                                                              38
##  4 A Country Boy Can Su...                                          75
##  5 A Little Gasoline                                                75
##  6 A Puro Dolor (Purest...                                          26
##  7 Aaron’s Party (Come ...                                          35
##  8 Absolutely (Story Of...                                           6
##  9 All Good?                                                        96
## 10 All The Small Things                                              6
## # ... with 306 more rows
```

d)List tracks in increasing order of its top rank

```
billboard %>% select(-artist) %>% select(-date.entered) %>%
  pivot_longer(-track, names_to = 'Week',values_to = 'Place',values_drop_na = TRUE) %>%
  group_by(track) %>% summarise('TopRank' = min(Place)) %>%
  arrange(TopRank)
```

```
## # A tibble: 316 x 2
##    track                   TopRank
##    <chr>                     <dbl>
##  1 Amazed                        1
##  2 Be With You                   1
##  3 Bent                          1
##  4 Come On Over Baby (A...       1
##  5 Doesn’t Really Matte...       1
##  6 Everything You Want           1
##  7 I Knew I Loved You            1
##  8 Incomplete                    1
##  9 Independent Women Pa...       1
## 10 It’s Gonna Be Me              1
## # ... with 306 more rows
```

e)Show for each artist, their top rank

```
billboard %>% select(-track, -date.entered) %>%
  pivot_longer(-artist, names_to = 'Week', values_to = 'Place',values_drop_na = TRUE) %>%
  group_by(artist) %>% summarise('TopRank' = min(Place))
```

```
## # A tibble: 228 x 2
##    artist              TopRank
##    <chr>                 <dbl>
##  1 2 Pac                    72
##  2 2Ge+her                  87
##  3 3 Doors Down              3
##  4 504 Boyz                 17
##  5 98^0                      2
##  6 A*Teens                  95
##  7 Aaliyah                   1
##  8 Adams, Yolanda           57
##  9 Adkins, Trace            65
## 10 Aguilera, Christina       1
## # ... with 218 more rows
```

f)List artists in increasing order of their top rank

```
billboard %>% select(-track, -date.entered) %>%
  pivot_longer(-artist, names_to = 'Week', values_to = 'Place',values_drop_na = TRUE) %>%
  group_by(artist) %>% summarise('TopRank' = min(Place)) %>% arrange(TopRank)
```

```
## # A tibble: 228 x 2
##    artist              TopRank
##    <chr>                 <dbl>
##  1 Aaliyah                     1
##  2 Aguilera, Christina         1
##  3 Carey, Mariah               1
##  4 Creed                       1
##  5 Destiny's Child             1
##  6 Iglesias, Enrique           1
##  7 Janet                       1
##  8 Lonestar                    1
##  9 Madonna                     1
## 10 matchbox twenty             1
## # ... with 218 more rows
```

g)List tracks that spent more than 35 weeks in the charts

```
billboard %>% select(-artist) %>% select(-date.entered) %>%
  pivot_longer(-track, names_to = 'Week',values_to = 'Place',values_drop_na = TRUE) %>%
  group_by(track) %>% summarize('Count'=n()) %>% filter(Count > 35)
```

```
## # A tibble: 9 x 2
##   track                 Count
##   <chr>                 <int>
## 1 Amazed                   55
## 2 Bent                     39
## 3 Breathe                  53
## 4 Everything You Want      41
## 5 He Wasn't Man Enough     37
## 6 Higher                   57
## 7 I Wanna Know             44
## 8 Kryptonite               53
## 9 With Arms Wide Open      47
```

h) List tracks that spent more than 35 weeks in the charts along with their artists

```
billboard %>% select(-date.entered) %>%
  pivot_longer(
    -c(artist, track), names_to = 'Week',values_to = 'Place',values_drop_na = TRUE
    ) %>%
  group_by(track, artist) %>% summarize('Count'=n()) %>% filter(Count > 35)
```

```
## `summarise()` has grouped output by 'track'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 9 x 3
## # Groups:   track [9]
##   track                 artist           Count
##   <chr>                 <chr>            <int>
## 1 Amazed                Lonestar            55
## 2 Bent                  matchbox twenty     39
## 3 Breathe               Hill, Faith         53
## 4 Everything You Want   Vertical Horizon    41
## 5 He Wasn't Man Enough Braxton, Toni       37
## 6 Higher                Creed               57
## 7 I Wanna Know         Joe                   44
## 8 Kryptonite           3 Doors Down          53
## 9 With Arms Wide Open  Creed                 47
```

Hint: First, convert to a tidy table. Show code first for this step. All the above questions can then be answered with a single data pipeline.

3. The demographics.csv file (available in the Datasets module on Canvas) gives the proportion of a country's population in different age groups and some other demographic data such as mortality rates and expected lifetime. You can read a CSV file into a tibble using tidyverse's read_csv(), like so: demo <- read_csv("demographics.csv")

```
demo <- read_csv("demographics.csv")
```

```
## Rows: 3885 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Country Name, Country Code, Series Name, Series Code
## dbl (1): YR2015
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

a) The data is not "tidy". In 2-3 sentences, explain why.

   The data is not tidy because the Series Code and YR2015 are messy to read.
   The Series name is also not that helpful as theres its already categorized on with the code.
   The table can group by the country names and then have multiple series code column.

b) Transform the table to tidy data with one country per row. [Give code]

```
new_demo <- demo %>% select(-`Series Name`) %>%
  pivot_wider(names_from = `Series Code`, values_from = YR2015) %>%
  group_by(`Country Name`)
new_demo
```

```
## # A tibble: 259 x 17
## # Groups:   Country Name [259]
##    ‘Country Name‘      ‘Country Code‘ SP.DYN.LE00.IN SP.URB.TOTL SP.POP.TOTL
##    <chr>               <chr>                   <dbl>       <dbl>       <dbl>
##  1 Afghanistan         AFG                      63.4     8535606    34413603
##  2 Albania             ALB                      78.0     1654503     2880703
##  3 Algeria             DZA                      76.1    28146511    39728025
##  4 American Samoa      ASM                      NA         48689       55812
##  5 Andorra             AND                      NA         68919       78011
##  6 Angola              AGO                      59.4    17691524    27884381
##  7 Antigua and Barbuda ATG                      76.5       23392       93566
##  8 Arab World          ARB                      71.2   229821020   396028278
##  9 Argentina           ARG                      76.1    39467043    43131966
## 10 Armenia             ARM                      74.5     1845585     2925553
## # ... with 249 more rows, and 12 more variables: SP.POP.80UP.FE <dbl>,
## #   SP.POP.80UP.MA <dbl>, SP.POP.1564.MA.IN <dbl>, SP.POP.1564.FE.IN <dbl>,
## #   SP.POP.0014.MA.IN <dbl>, SP.POP.0014.FE.IN <dbl>, SP.DYN.AMRT.FE <dbl>,
## #   SP.DYN.AMRT.MA <dbl>, SP.POP.TOTL.FE.IN <dbl>, SP.POP.TOTL.MA.IN <dbl>,
## #   SP.POP.65UP.FE.IN <dbl>, SP.POP.65UP.MA.IN <dbl>
```

c) Add the male/female population numbers together (i.e., ignore sex-related differences). [Hint: You will have to mutate for every pair of columns, e.g., mutate(SP.POP.0014.IN=SP.POP.0014.MA.IN+SP.POP.0014.FE.IN) [Give code]

```
combine_demo <- new_demo %>% mutate(SP.POP.80UP=SP.POP.80UP.MA+SP.POP.80UP.FE) %>%
  mutate(SP.POP.1564=SP.POP.1564.MA.IN+SP.POP.1564.FE.IN) %>%
  mutate(SP.POP.0014.IN=SP.POP.0014.MA.IN+SP.POP.0014.FE.IN) %>%
  mutate(SP.DYN.AMRT=SP.DYN.AMRT.MA+SP.DYN.AMRT.FE) %>%
  mutate(SP.POP.TOTL.IN=SP.POP.TOTL.MA.IN+SP.POP.TOTL.FE.IN) %>%
  mutate(SP.POP.65UP.IN=SP.POP.65UP.MA.IN+SP.POP.65UP.FE.IN) %>%
  select(c(`Country Name`, `Country Code`, SP.DYN.LE00.IN, SP.URB.TOTL,
          SP.POP.80UP, SP.POP.1564,SP.POP.0014.IN,SP.DYN.AMRT,SP.POP.TOTL.IN,
          SP.POP.65UP.IN))
combine_demo
```

```
## # A tibble: 259 x 10
## # Groups:   Country Name [259]
##    ‘Country Name‘      ‘Country Code‘ SP.DYN.LE00.IN SP.URB.TOTL SP.POP.80UP
##    <chr>               <chr>                   <dbl>       <dbl>       <dbl>
##  1 Afghanistan         AFG                      63.4     8535606       85552
##  2 Albania             ALB                      78.0     1654503       66965
##  3 Algeria             DZA                      76.1    28146511      453741
##  4 American Samoa      ASM                      NA         48689          NA
##  5 Andorra             AND                      NA         68919          NA
##  6 Angola              AGO                      59.4    17691524       69363
##  7 Antigua and Barbuda ATG                      76.5       23392        1571
##  8 Arab World          ARB                      71.2   229821020     2689793
##  9 Argentina           ARG                      76.1    39467043     1095211
## 10 Armenia             ARM                      74.5     1845585       77292
## # ... with 249 more rows, and 5 more variables: SP.POP.1564 <dbl>,
## #   SP.POP.0014.IN <dbl>, SP.DYN.AMRT <dbl>, SP.POP.TOTL.IN <dbl>,
## #   SP.POP.65UP.IN <dbl>
```

d) Write code to show the top 5 countries with the lowest proportion of the population below 14 years old (i.e., SP.POP.0014.IN/SP.POP.TOTL) [Code, and list of 5 countries]

```
demo.0014 <- combine_demo %>%
  mutate(`Percent of 14 years and Under` = SP.POP.0014.IN/SP.POP.TOTL.IN) %>%
  select(c(`Country Name`,`Percent of 14 years and Under`)) %>%

    arrange(`Percent of 14 years and Under`)
demo.0014[1:5,] #couldn't figure out top_n
```

```
## # A tibble: 5 x 2
## # Groups:   Country Name [5]
##   ‘Country Name‘       ‘Percent of 14 years and Under‘
##   <chr>                                          <dbl>
## 1 Hong Kong SAR, China                           0.112
## 2 Macao SAR, China                               0.126
## 3 Singapore                                      0.126
## 4 Japan                                          0.130
## 5 Germany                                        0.132
```