# Homework 6

Prof. Sarah Cannon

Due Thursday 10/17/2024

## Question 1 (4 points)

> **Read the ACM Code of Ethics at https://www.acm.org/code-of-ethics (https://www.acm.org/code-of-ethics) in its entirety. In 1-2 paragraphs below, summarize which parts of the code of ethics are most relevant for a data scientist and how you plan to ensure you adhere to this code of ethics in any future data science (or data science-adjacent) work you do.**

The ACM Code of Ethics is important for data scientists as it emphasizes principles like contributing to society, avoiding harm, and respecting privacy. For data scientists, these principles translate into ensuring that data collection and analysis benefit society, do not cause harm, and protect individuals' privacy. I plan to implement robust data privacy measures, ensure transparency in data usage, and prioritize ethical considerations in all data-driven decisions. Additionally, I will engage in continuous learning to stay updated on best practices and ethical standards in data science.

## Question 2 (4 points)

> **Consider the sets A = {red, green, blue} and B = {purple, blue, pink, yellow}.**

> a. **For these sets A and B, what is the union of A and B?**

{red, green, blue, purple, pink, yellow}

> b. **For these sets A and B, what is the intersection of A and B?**

{blue}

> c. **For these sets A and B, what is set difference A - B?**

{red, green}

> d. **For these sets A and B, what is the Cartesian Product A $\times$ B?**

{(red, purple), (red, blue), (red, pink), (red, yellow), (green, purple), (green, blue), (green, pink), (green, yellow), (blue, purple), (blue, pink), (blue, yellow)}

# Question 3 (8 points)

> **This question deals with the Lahman package, which has several tables related to baseball. *MAKE SURE NO MORE THAN 10 ROWS OF ANY LIST OR TABLE PRINT IN YOUR KNITTED FILE*.**

> a. **What column makes a primary key in the People table? Explain how you know this is a valid key.**

```
head(People, 10)
```

```
##     playerID birthYear birthMonth birthDay    birthCity birthCountry birthState death
Year deathMonth deathDay deathCountry deathState
## 1  aardsda01      1981         12       27       Denver          USA         CO
NA         NA       NA         <NA>       <NA>
## 2  aaronha01      1934          2        5       Mobile          USA         AL
2021          1       22          USA         GA
## 3  aaronto01      1939          8        5       Mobile          USA         AL
1984          8       16          USA         GA
## 4   aasedo01      1954          9        8       Orange          USA         CA
NA         NA       NA         <NA>       <NA>
## 5   abadan01      1972          8       25   Palm Beach          USA         FL
NA         NA       NA         <NA>       <NA>
## 6   abadfe01      1985         12       17    La Romana         D.R.  La Romana
NA         NA       NA         <NA>       <NA>
## 7  abadijo01      1850         11        4 Philadelphia          USA         PA
1905          5       17          USA         NJ
## 8  abbated01      1877          4       15      Latrobe          USA         PA
1957          1        6          USA         FL
## 9  abbeybe01      1869         11       11        Essex          USA         VT
1962          6       11          USA         VT
## 10 abbeych01      1866         10       14   Falls City          USA         NE
1926          4       27          USA         CA
##          deathCity nameFirst    nameLast        nameGiven weight height bats throws
debut   bbrefID  finalGame  retroID  deathDate
## 1             <NA>     David     Aardsma      David Allan    215     75    R      R 2
004-04-06 aardsda01 2015-08-23 aardd001       <NA>
## 2          Atlanta      Hank       Aaron      Henry Louis    180     72    R      R 1
954-04-13 aaronha01 1976-10-03 aaroh101 2021-01-22
## 3          Atlanta    Tommie       Aaron       Tommie Lee    190     75    R      R 1
962-04-10 aaronto01 1971-09-26 aarot101 1984-08-16
## 4             <NA>       Don        Aase   Donald William    190     75    R      R 1
977-07-26  aasedo01 1990-10-03 aased001       <NA>
## 5             <NA>      Andy        Abad    Fausto Andres    184     73    L      L 2
001-09-10  abadan01 2006-04-13 abada001       <NA>
## 6             <NA>  Fernando        Abad Fernando Antonio    235     74    L      L 2
010-07-28  abadfe01 2021-10-01 abadf001       <NA>
## 7        Pemberton      John      Abadie          John W.    192     72    R      R 1
875-04-26 abadijo01 1875-06-10 abadj101 1905-05-17
## 8  Fort Lauderdale        Ed Abbaticchio     Edward James    170     71    R      R 1
897-09-04 abbated01 1910-09-15 abbae101 1957-01-06
## 9       Colchester      Bert       Abbey        Bert Wood    175     71    R      R 1
892-06-14 abbeybe01 1896-09-23 abbeb101 1962-06-11
## 10   San Francisco   Charlie       Abbey       Charles S.    169     68    L      L 1
893-08-16 abbeych01 1897-08-19 abbec101 1926-04-27
##     birthDate
## 1  1981-12-27
## 2  1934-02-05
## 3  1939-08-05
## 4  1954-09-08
## 5  1972-08-25
## 6  1985-12-17
## 7  1850-11-04
```

```
## 8  1877-04-15
## 9  1869-11-11
## 10 1866-10-14
```

The primary key in the People table is the playerID column, as it uniquely identifies each row.

> b. **Explain why the pair of columns { nameFirst, nameLast } aren't a key for the People table. Give an example of specific entries in the table that support your explanation.**

The pair of columns { nameFirst, nameLast } are not a primary key for the People table because multiple people can have the same first and last name.

```
# find all people with the same first and last name
People %>% group_by(nameFirst, nameLast) %>% summarize(n = n()) %>% filter(n > 1)
```

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

```
## # A tibble: 568 × 3
## # Groups:   nameFirst [166]
##    nameFirst nameLast     n
##    <chr>     <chr>    <int>
##  1 Abraham   Nunez        2
##  2 Adam      Eaton        2
##  3 Adam      Peterson     2
##  4 Al        Martin       2
##  5 Al        Shaw         2
##  6 Al        Smith        3
##  7 Al        Wright       2
##  8 Alberto   Castillo     2
##  9 Alex      Gonzalez     2
## 10 Alex      Sanchez      2
## # ℹ 558 more rows
```

> c. **Is the column you identified in part (a) a primary key in the Batting table? Explain why or why not.**

```
head(Batting, 10)
```

```
##     playerID yearID stint teamID lgID   G  AB   R   H X2B X3B HR RBI SB CS BB SO IBB H
BP SH SF GIDP
## 1  aardsda01   2004     1    SFN   NL  11   0  0   0   0   0  0   0  0  0  0  0   0
0  0  0    0
## 2  aardsda01   2006     1    CHN   NL  45   2  0   0   0   0  0   0  0  0  0  0   0
0  1  0    0
## 3  aardsda01   2007     1    CHA   AL  25   0  0   0   0   0  0   0  0  0  0  0   0
0  0  0    0
## 4  aardsda01   2008     1    BOS   AL  47   1  0   0   0   0  0   0  0  0  0  1   0
0  0  0    0
## 5  aardsda01   2009     1    SEA   AL  73   0  0   0   0   0  0   0  0  0  0  0   0
0  0  0    0
## 6  aardsda01   2010     1    SEA   AL  53   0  0   0   0   0  0   0  0  0  0  0   0
0  0  0    0
## 7  aardsda01   2012     1    NYA   AL   1   0  0   0   0   0  0   0  0  0  0  0   0
0  0  0    0
## 8  aardsda01   2013     1    NYN   NL  43   0  0   0   0   0  0   0  0  0  0  0   0
0  0  0    0
## 9  aardsda01   2015     1    ATL   NL  33   1  0   0   0   0  0   0  0  0  0  1   0
0  0  0    0
## 10 aaronha01   1954     1    ML1   NL 122 468 58 131  27   6 13  69  2  2 28 39  NA
3  6  4   13
```

No, the playerID column is not a primary key in the Batting table because players have multiple rows in the Batting table, with each row representing a different season or year of the player's career.

> d. **Is the column you identified in part (a) a foreign key in the Batting table? Explain why or why not.**

Yes, the playerID column is a foreign key in the Batting table because it is a primary key in the People table and it is used to link the Batting table to the People table.

# Question 4 (6 points)

> **Consider the following table of bird sightings; more information about this data is available at https://github.com/rfordatascience/tidytuesday/blob/master/data/2023/2023-01-10/readme.md (https://github.com/rfordatascience/tidytuesday/blob/master/data/2023/2023-01-10/readme.md).**

```
## Rows: 100000 Columns: 22
## ── Column specification ────────────────────────────────────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (8): loc_id, subnational1_code, entry_technique, sub_id, obs_id, PROJ_PERIOD_ID, species_code, Data_Entry_Method
## dbl (14): latitude, longitude, Month, Day, Year, how_many, valid, reviewed, day1_am, day1_pm, day2_am, day2_pm, effort_hrs_atleast, snow_dep_a...
##
## 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. **Explain in your own words what the distinct() function does, and use it to explain how you know the bird_observations table doesn't have any duplicate rows.**

The distinct() function is used to return a tibble with duplicate rows removed. It is used to find the unique rows in a tibble.

```
original_row_count <- nrow(bird_observations)
distinct_row_count <- nrow(distinct(bird_observations))
original_row_count
```

```
## [1] 100000
```

```
distinct_row_count
```

```
## [1] 100000
```

```
original_row_count == distinct_row_count
```

```
## [1] TRUE
```

Since the original row count is equal to the distinct row count, the bird_observations table doesn't have any duplicate rows.

> b. **Because the bird_observations table doesn't have any duplicate rows, it must have at least one key. What do you think is the best set of columns to choose to serve as a primary key for this table? Explain how you know it is a valid key. Hint: Think about what the observations are; you should not have more than 5 columns in your key. There is more than one correct answer.**

These columns together can form a composite key because: loc_id ensures the uniqueness of the location. sub_id and obs_id ensure the uniqueness of the checklist and observation within that location. species_code specifies the species being observed. proj_period_id provides the temporal context of the observation.

We can check if the primary key is unique by using the nrow() and n_distinct() functions.

```
# Create a primary key
bird_observations <- bird_observations %>%
  mutate(primary_key = paste(loc_id, sub_id, obs_id, species_code, PROJ_PERIOD_ID, sep =
"_"))

# Check if the primary key is unique
is_unique <- nrow(bird_observations) == n_distinct(bird_observations$primary_key)
is_unique
```

```
## [1] TRUE
```

# Question 5 (6 points)

> a. **Explain why the diamonds data set doesn't meet the three assumptions we discussed in class on 9-30; be specific about which assumption(s) it violates.**

The assumptions are:

1. Data is tidy
2. Order of rows does not matter
3. There are no identical rows

Lets examine the diamonds data set:

```
head(diamonds, 10)
```

```
## # A tibble: 10 × 11
##    carat cut       color clarity depth table price     x     y     z primary_key
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>       <int>
##  1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43           1
##  2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31           2
##  3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31           3
##  4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63           4
##  5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75           5
##  6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48           6
##  7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47           7
##  8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53           8
##  9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49           9
## 10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39          10
```

```
# Check if the order of rows matters
order_column_exists <- any(names(diamonds) %in% "order")
order_column_exists
```

```
## [1] FALSE
```

```
# Check for duplicate rows
duplicate_rows_exist <- nrow(diamonds) != nrow(distinct(diamonds))
duplicate_rows_exist
```

```
## [1] FALSE
```

```
# Identify duplicate rows
duplicate_rows <- diamonds %>%
  group_by(across(everything())) %>%
  filter(n() > 1) %>%
  ungroup()

# Display the duplicate rows
duplicate_rows
```

```
## # A tibble: 0 × 11
## # ℹ 11 variables: carat <dbl>, cut <ord>, color <ord>, clarity <ord>, depth <dbl>, ta
ble <dbl>, price <int>, x <dbl>, y <dbl>, z <dbl>,
## #   primary_key <int>
```

The diamonds data set violates the third assumption because there are duplicate rows.

> b. **Suppose you have good documentation for the diamonds data set and know that: every row corresponds to a different diamond; there were no mistakes in collecting/entering this data; and the order of the diamonds doesn't matter. Modify the diamonds data set in an appropriate way to make sure it satisfies the three assumptions.**

I will add a primary key to the diamonds data set.

```
diamonds <- diamonds %>%
  mutate(primary_key = row_number())
```

# Question 6 (21 points)

> **Consider the following tibbles:**

> a. **Suppose the order of the cities in CityInfo list matters. Modify your table in an appropriate way, and explain why this is a good thing to do.**

I will add a primary key to the CityInfo table. This is a good thing to do because it ensures that the order of the cities in the CityInfo table is preserved.

```
CityInfo <- CityInfo %>%
  mutate(primary_key = row_number())

head(CityInfo, 10)
```

```
## # A tibble: 8 × 4
##   City           Country    Population primary_key
##   <chr>          <chr>           <dbl>       <int>
## 1 Boston         USA            650706           1
## 2 San Jose       Costa Rica     339581           2
## 3 Toronto        Canada        2930000           3
## 4 Rio de Janeiro Brazil        6211000           4
## 5 Cartago        Costa Rica     160457           5
## 6 Vancouver      Canada         675218           6
## 7 Buenos Aires   Argentina     3121000           7
## 8 Los Angeles    USA           3822000           8
```

> b. **Join these tibbles according to Country using an inner join. Which city/cities appear in two different rows, which city/cities appear only in one row, and which city/cities don't appear in this tibble? Explain why this is.**

```
inner_join(CityInfo, Regions, by = "Country")
```

```
## Warning in inner_join(CityInfo, Regions, by = "Country"): Detected an unexpected many
-to-many relationship between `x` and `y`.
## ℹ Row 2 of `x` matches multiple rows in `y`.
## ℹ Row 2 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship = "many-to-many"` to
silence this warning.
```

```
## # A tibble: 9 × 5
##   City           Country    Population primary_key Region
##   <chr>          <chr>           <dbl>       <int> <chr>
## 1 Boston         USA            650706           1 North America
## 2 San Jose       Costa Rica     339581           2 Central America
## 3 San Jose       Costa Rica     339581           2 North America
## 4 Toronto        Canada        2930000           3 North America
## 5 Rio de Janeiro Brazil        6211000           4 South America
## 6 Cartago        Costa Rica     160457           5 Central America
## 7 Cartago        Costa Rica     160457           5 North America
## 8 Vancouver      Canada         675218           6 North America
## 9 Los Angeles    USA           3822000           8 North America
```

Costa Rica appears in two different rows because it is a country that is both in Central America and North America. The cities that appear in two different rows are San Jose and Cartago. The cities that appear only in one row are Boston, Toronto, Rio de Janeiro, Vancouver, and Los Angeles. The cities that don't appear in this tibble are Panama City and Santiago. This is because Panama and Chile are not listed in the Regions tibble.

> c. **Joining these tables with a left_join rather than an inner_join results in a tibble with one more row than in part (b). Which additional row is present here and why?**

```
left_join(CityInfo, Regions, by = "Country")
```

```
## Warning in left_join(CityInfo, Regions, by = "Country"): Detected an unexpected many-
to-many relationship between `x` and `y`.
## ℹ Row 2 of `x` matches multiple rows in `y`.
## ℹ Row 2 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship = "many-to-many"` to
silence this warning.
```

```
## # A tibble: 10 × 5
##    City           Country    Population primary_key Region
##    <chr>          <chr>           <dbl>       <int> <chr>
##  1 Boston         USA            650706           1 North America
##  2 San Jose       Costa Rica     339581           2 Central America
##  3 San Jose       Costa Rica     339581           2 North America
##  4 Toronto        Canada        2930000           3 North America
##  5 Rio de Janeiro Brazil        6211000           4 South America
##  6 Cartago        Costa Rica     160457           5 Central America
##  7 Cartago        Costa Rica     160457           5 North America
##  8 Vancouver      Canada         675218           6 North America
##  9 Buenos Aires   Argentina     3121000           7 <NA>
## 10 Los Angeles    USA           3822000           8 North America
```

The additional row is Argentina. This is because Argentina is a country that is only in the CityInfo table.

> d. **Joining these tables with a right_join rather than an inner_join results in a tibble with two more rows than in part (b). Which additional rows are present here and why?**

```
right_join(CityInfo, Regions, by = "Country")
```

```
## Warning in right_join(CityInfo, Regions, by = "Country"): Detected an unexpected many
-to-many relationship between `x` and `y`.
## ℹ Row 2 of `x` matches multiple rows in `y`.
## ℹ Row 2 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship = "many-to-many"` to
silence this warning.
```

```
## # A tibble: 11 × 5
##    City           Country    Population primary_key Region
##    <chr>          <chr>           <dbl>       <int> <chr>
##  1 Boston         USA            650706           1 North America
##  2 San Jose       Costa Rica     339581           2 Central America
##  3 San Jose       Costa Rica     339581           2 North America
##  4 Toronto        Canada        2930000           3 North America
##  5 Rio de Janeiro Brazil        6211000           4 South America
##  6 Cartago        Costa Rica     160457           5 Central America
##  7 Cartago        Costa Rica     160457           5 North America
##  8 Vancouver      Canada         675218           6 North America
##  9 Los Angeles    USA           3822000           8 North America
## 10 <NA>           Panama             NA          NA Central America
## 11 <NA>           Chile              NA          NA South America
```

The additional rows are Panama and Chile. This is because Panama and Chile are countries that are only in the Regions tibble.

> e. **Joining these tables with a full_join rather than an inner_join results in a tibble with three more rows than in part (b). Which additional rows are present here and why?**

```
full_join(CityInfo, Regions, by = "Country")
```

```
## Warning in full_join(CityInfo, Regions, by = "Country"): Detected an unexpected many-
to-many relationship between `x` and `y`.
## ℹ Row 2 of `x` matches multiple rows in `y`.
## ℹ Row 2 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship = "many-to-many"` to
silence this warning.
```

```
## # A tibble: 12 × 5
##    City           Country    Population primary_key Region
##    <chr>          <chr>           <dbl>       <int> <chr>
##  1 Boston         USA            650706           1 North America
##  2 San Jose       Costa Rica     339581           2 Central America
##  3 San Jose       Costa Rica     339581           2 North America
##  4 Toronto        Canada        2930000           3 North America
##  5 Rio de Janeiro Brazil        6211000           4 South America
##  6 Cartago        Costa Rica     160457           5 Central America
##  7 Cartago        Costa Rica     160457           5 North America
##  8 Vancouver      Canada         675218           6 North America
##  9 Buenos Aires   Argentina     3121000           7 <NA>
## 10 Los Angeles    USA           3822000           8 North America
## 11 <NA>           Panama             NA          NA Central America
## 12 <NA>           Chile              NA          NA South America
```

The additional rows are Panama, Chile, and Argentina. This is because Panama and Chile are countries that are only in the Regions tibble, and Argentina is a country that is only in the CityInfo table.

> f. **Join these tibbles according to Country using a semi_join(). Which row(s) and column(s) appear in the resulting table? Explain why this is.**

```
semi_join(CityInfo, Regions, by = "Country")
```

```
## # A tibble: 7 × 4
##   City           Country    Population primary_key
##   <chr>          <chr>           <dbl>       <int>
## 1 Boston         USA            650706           1
## 2 San Jose       Costa Rica     339581           2
## 3 Toronto        Canada        2930000           3
## 4 Rio de Janeiro Brazil        6211000           4
## 5 Cartago        Costa Rica     160457           5
## 6 Vancouver      Canada         675218           6
## 7 Los Angeles    USA           3822000           8
```

The rows that appear in the resulting table are those from CityInfo where the Country also appears in the Regions table. The columns in the resulting table are the same as those in the CityInfo table. This is because semi_join() filters CityInfo to only include rows with a Country that is present in Regions.

> g. **Join these tibbles according to Country using an anti_join(). Which row(s) and column(s) appear in the resulting table? Explain why this is.**

```
anti_join(CityInfo, Regions, by = "Country")
```

```
## # A tibble: 1 × 4
##   City         Country   Population primary_key
##   <chr>        <chr>          <dbl>       <int>
## 1 Buenos Aires Argentina    3121000           7
```

The rows that appear in the resulting table are those from CityInfo where the Country does not appear in the Regions table. The columns in the resulting table are the same as those in the CityInfo table. This is because anti_join() filters CityInfo to exclude rows with a Country that is present in Regions.

# Question 7 (6 points)

> **Consider the following two tibbles.**

> a. **Join these tibbles by the species column using a full_join. Explain why doing this join is probably a bad idea.**

```
full_join(October_Pets, Pet_Average_Weights, by = "species")
```

```
## Warning in full_join(October_Pets, Pet_Average_Weights, by = "species"): Detected an
unexpected many-to-many relationship between `x` and `y`.
## ℹ Row 1 of `x` matches multiple rows in `y`.
## ℹ Row 3 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship = "many-to-many"` to
silence this warning.
```

```
## # A tibble: 20 × 6
##    name    species age_months arrival_day sex    avg_weight_lbs
##    <chr>   <chr>        <dbl>       <dbl> <chr>           <dbl>
##  1 Sparky  Dog             31           3 Female           45
##  2 Sparky  Dog             31           3 Male             50
##  3 Fido    Dog             29          11 Female           45
##  4 Fido    Dog             29          11 Male             50
##  5 Fluffy  Cat             78           4 Female            9.4
##  6 Fluffy  Cat             78           4 Male             10.1
##  7 Lassie  Dog             98          28 Female           45
##  8 Lassie  Dog             98          28 Male             50
##  9 Patches Cat            115          14 Female            9.4
## 10 Patches Cat            115          14 Male             10.1
## 11 Spot    Dog              7          12 Female           45
## 12 Spot    Dog              7          12 Male             50
## 13 Socks   Cat              4          17 Female            9.4
## 14 Socks   Cat              4          17 Male             10.1
## 15 Buddy   Dog             15          15 Female           45
## 16 Buddy   Dog             15          15 Male             50
## 17 Lizzie  Lizard           2           1 Female            0.4
## 18 Lizzie  Lizard           2           1 Male              0.3
## 19 Tweety  Bird             6           2 Female            0.8
## 20 Tweety  Bird             6           2 Male              0.9
```

This join is probably a bad idea because it is a many-to-many relationship. This is because there are multiple species in the October_Pets table that have the same species in the Pet_Average_Weights table.

> b. **Explain why you have the number of rows that you do in your join in the previous part.**

The number of rows in the join is 20 because each row in the October_Pets table is matched with each corresponding row in the Pet_Average_Weights table based on the species column. This results in a many-to-many relationship due to the gender differences, leading to multiple rows for each species and gender combination.

# Question 8 (9 points)

> a. (6 points) **Add to the flights data set the altitude of the origin airports and the altitude of the destination airports. That is, each row should now have 2 more additional columns, which you should name origin_alt and dest_alt. Move your columns for origin, destination, and their altitudes to the front of your data set, with the remaining columns displayed after them.**

```
flights <- flights %>%
  left_join(select(airports, faa, alt), by = c("origin" = "faa")) %>%
  left_join(select(airports, faa, alt), by = c("dest" = "faa"), suffix = c("_origin", "_
dest"))

flights <- flights %>%
  select(origin, dest, alt_origin, alt_dest, everything())

head(flights, 10)
```

```
## # A tibble: 10 × 29
##    origin dest  alt_origin alt_dest  year month   day dep_time sched_dep_time dep_del
ay arr_time sched_arr_time arr_delay carrier flight tailnum
##    <chr>  <chr>      <dbl>    <dbl> <int> <int> <int>    <int>          <int>     <db
l>    <int>          <int>     <dbl> <chr>    <int> <chr>
##  1 EWR    IAH           18       97  2013     1     1      517            515
2      830            819        11 UA        1545 N14228
##  2 LGA    IAH           22       97  2013     1     1      533            529
4      850            830        20 UA        1714 N24211
##  3 JFK    MIA           13        8  2013     1     1      542            540
2      923            850        33 AA        1141 N619AA
##  4 JFK    BQN           13       NA  2013     1     1      544            545
-1     1004           1022       -18 B6          725 N804JB
##  5 LGA    ATL           22     1026  2013     1     1      554            600
-6      812            837       -25 DL          461 N668DN
##  6 EWR    ORD           18      668  2013     1     1      554            558
-4      740            728        12 UA         1696 N39463
##  7 EWR    FLL           18        9  2013     1     1      555            600
-5      913            854        19 B6          507 N516JB
##  8 LGA    IAD           22      313  2013     1     1      557            600
-3      709            723       -14 EV         5708 N829AS
##  9 JFK    MCO           13       96  2013     1     1      557            600
-3      838            846        -8 B6           79 N593JB
## 10 LGA    ORD           22      668  2013     1     1      558            600
-2      753            745         8 AA          301 N3ALAA
## # ℹ 13 more variables: air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time
_hour <dttm>, alt_origin_origin <dbl>, alt_dest_dest <dbl>,
## #   alt_origin_origin_origin <dbl>, alt_dest_dest_dest <dbl>, alt_origin_origin_origi
n_origin <dbl>, alt_dest_dest_dest_dest <dbl>,
## #   alt_origin_origin_origin_origin_origin <dbl>, alt_dest_dest_dest_dest_dest <dbl>
```

> b. (3 points) **The following command attaches plane information to the flights tibble, for all flights where the tail number appears in the planes tibble. There's over 284,000 such flights:**

```
inner_join(flights, planes, by = "tailnum")
```

```
## # A tibble: 284,170 × 37
##    origin dest  alt_origin alt_dest year.x month   day dep_time sched_dep_time dep_de
lay arr_time sched_arr_time arr_delay carrier flight tailnum
##    <chr>  <chr>      <dbl>    <dbl>  <int> <int> <int>    <int>          <int>     <d
bl>    <int>          <int>     <dbl> <chr>    <int> <chr>
##  1 EWR    IAH           18       97   2013     1     1      517            515
2      830            819        11 UA        1545 N14228
##  2 LGA    IAH           22       97   2013     1     1      533            529
4      850            830        20 UA        1714 N24211
##  3 JFK    MIA           13        8   2013     1     1      542            540
2      923            850        33 AA        1141 N619AA
##  4 JFK    BQN           13       NA   2013     1     1      544            545
-1     1004           1022       -18 B6         725 N804JB
##  5 LGA    ATL           22     1026   2013     1     1      554            600
-6      812            837       -25 DL         461 N668DN
##  6 EWR    ORD           18      668   2013     1     1      554            558
-4      740            728        12 UA        1696 N39463
##  7 EWR    FLL           18        9   2013     1     1      555            600
-5      913            854        19 B6         507 N516JB
##  8 LGA    IAD           22      313   2013     1     1      557            600
-3      709            723       -14 EV        5708 N829AS
##  9 JFK    MCO           13       96   2013     1     1      557            600
-3      838            846        -8 B6          79 N593JB
## 10 JFK    PBI           13       19   2013     1     1      558            600
-2      849            851        -2 B6          49 N793JB
## # ℹ 284,160 more rows
## # ℹ 21 more variables: air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time
_hour <dttm>, alt_origin_origin <dbl>, alt_dest_dest <dbl>,
## #   alt_origin_origin_origin <dbl>, alt_dest_dest_dest <dbl>, alt_origin_origin_origi
n_origin <dbl>, alt_dest_dest_dest_dest <dbl>,
## #   alt_origin_origin_origin_origin_origin <dbl>, alt_dest_dest_dest_dest_dest <dbl>,
year.y <int>, type <chr>, manufacturer <chr>, model <chr>,
## #   engines <int>, seats <int>, speed <int>, engine <chr>
```

> **When we remove the “by” argument, we get a tibble with fewer than 5000 rows. Explain what’s happening here, and why these particular rows have been included in this tibble.**

```
inner_join(flights, planes)
```

```
## Joining with `by = join_by(year, tailnum)`
```

```
## # A tibble: 4,630 × 36
##    origin dest  alt_origin alt_dest  year month   day dep_time sched_dep_time dep_del
ay arr_time sched_arr_time arr_delay carrier flight tailnum
##    <chr>  <chr>      <dbl>    <dbl> <int> <int> <int>    <int>          <int>     <db
l>    <int>          <int>     <dbl> <chr>    <int> <chr>
##  1 EWR    FLL           18        9  2013     1    18     1846           1810
36     2156           2120        36 UA        1292 N37465
##  2 JFK    BOS           13       19  2013    10     1      647            655
-8      744            809       -25 B6         318 N355JB
##  3 EWR    LAX           18      126  2013    10     1      652            652
0      921            954       -33 UA        1439 N37471
##  4 JFK    MSP           13      841  2013    10     1      755            800
-5      954           1013       -19 9E        3538 N292PQ
##  5 JFK    HOU           13       46  2013    10     1      813            820
-7     1050           1110       -20 B6         281 N354JB
##  6 JFK    SYR           13      421  2013    10     1      925            930
-5     1025           1038       -13 B6         116 N373JB
##  7 JFK    IAD           13      313  2013    10     1     1113           1120
-7     1215           1230       -15 B6        1307 N374JB
##  8 JFK    ROC           13      559  2013    10     1     1426           1429
-3     1535           1548       -13 B6         286 N368JB
##  9 LGA    CLT           22      748  2013    10     1     1446           1450
-4     1635           1652       -17 US        1995 N156UW
## 10 JFK    MSY           13        4  2013    10     1     1454           1455
-1     1751           1718        33 B6         575 N374JB
## # ℹ 4,620 more rows
## # ℹ 20 more variables: air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time
_hour <dttm>, alt_origin_origin <dbl>, alt_dest_dest <dbl>,
## #   alt_origin_origin_origin <dbl>, alt_dest_dest_dest <dbl>, alt_origin_origin_origi
n_origin <dbl>, alt_dest_dest_dest_dest <dbl>,
## #   alt_origin_origin_origin_origin_origin <dbl>, alt_dest_dest_dest_dest_dest <dbl>,
type <chr>, manufacturer <chr>, model <chr>, engines <int>,
## #   seats <int>, speed <int>, engine <chr>
```

When we remove the “by” argument, we get a tibble with fewer than 5000 rows because the join is performed using all columns that have the same names in both tibbles. The rows that are included in this tibble are the ones where all corresponding columns in both tibbles match.

# Question 9 (18 points)

> **This question considers the following three data sets, from a sentiment analysis for African languages. More information about this data set can be found at https://github.com/rfordatascience/tidytuesday/blob/master/data/2023/2023-02-28/readme.md (https://github.com/rfordatascience/tidytuesday/blob/master/data/2023/2023-02-28/readme.md).**

```
afrisenti <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2023/2023-02-28/afrisenti.csv")
```

```
## Rows: 111720 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (4): language_iso_code, tweet, label, intended_use
##
## 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.
```

```
languages <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2023/2023-02-28/languages.csv")
```

```
## Rows: 14 Columns: 2
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): language_iso_code, language
##
## 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.
```

```
language_countries <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2023/2023-02-28/language_countries.csv")
```

```
## Rows: 23 Columns: 2
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): language_iso_code, country
##
## 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. **Are there any language iso codes that appear in the afrisenti table but not in the languages table, or vice versa? Explain how you know.**

```
missing_in_languages <- afrisenti %>%
  anti_join(languages, by = "language_iso_code")
missing_in_languages
```

```
## # A tibble: 0 × 4
## # ℹ 4 variables: language_iso_code <chr>, tweet <chr>, label <chr>, intended_use <chr
>
```

```
missing_in_afrisenti <- languages %>%
  anti_join(afrisenti, by = "language_iso_code")
missing_in_afrisenti
```

```
## # A tibble: 0 × 2
## # ℹ 2 variables: language_iso_code <chr>, language <chr>
```

No, there are no language iso codes that appear in the afrisenti table but not in the languages table, or vice versa because the anti_join() function returns an empty tibble.

> b. **Explain why a left_join, right_join, inner_join, and full_join of the afrisenti and languages data tables will all produce the same result.**

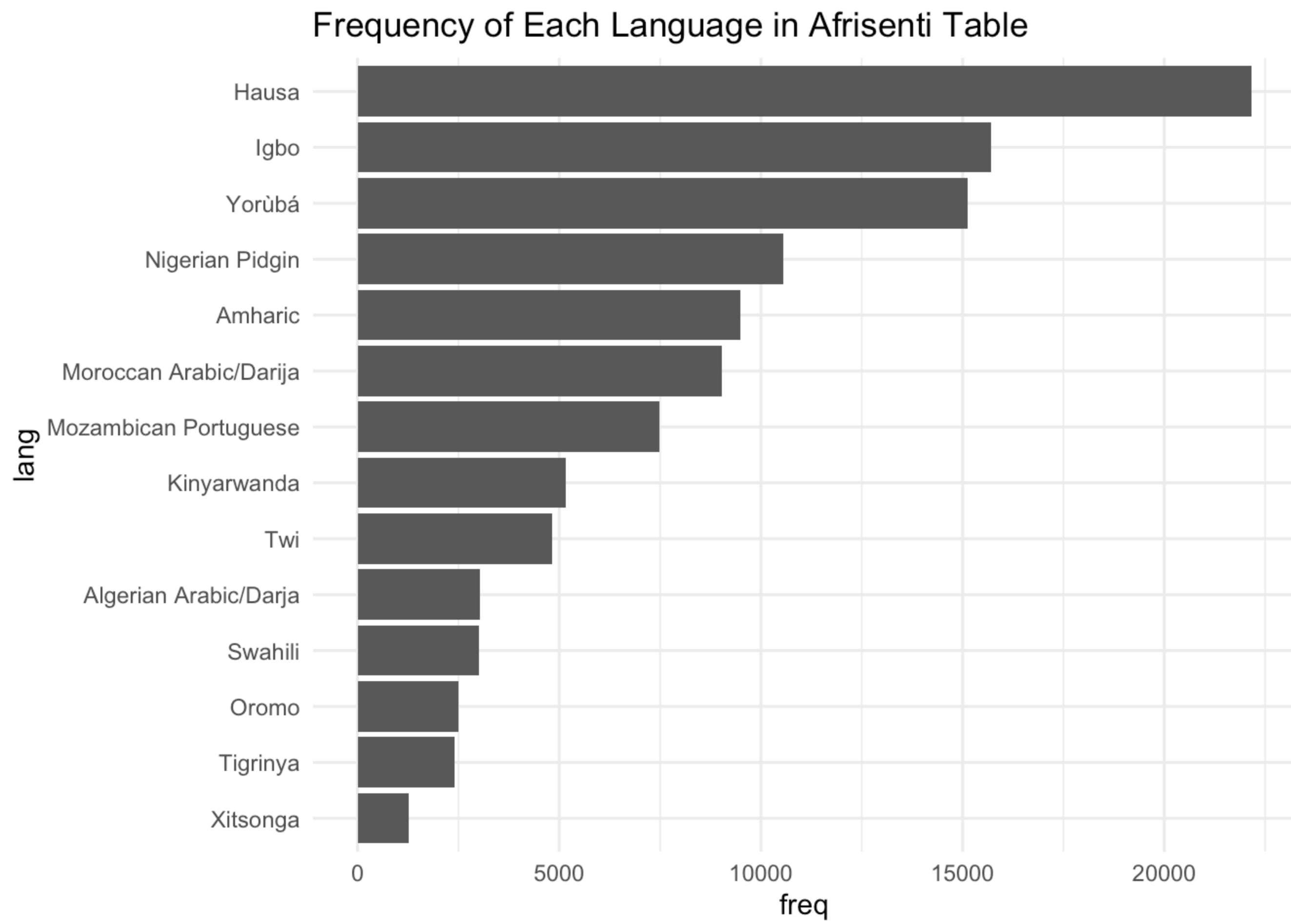
A left_join, right_join, inner_join, and full_join of the afrisenti and languages tables will all produce the same result because the language iso codes match in both tables.

> c. **Make a barchart showing how frequently each language appears in the afrisenti table. Your plot should use the full name of all the languages, not the iso abbreviations for the languages.**

```
afrisenti_full <- afrisenti %>%
  left_join(languages, by = "language_iso_code")

language_counts <- afrisenti_full %>%
  count(language)

ggplot(language_counts, aes(x = reorder(language, n), y = n)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(
    title = "Frequency of Each Language in Afrisenti Table",
    x = "lang",
    y = "freq"
  ) +
  theme_minimal()
```




> d. **Join the afrisenti and language_countries data sets using any join type you'd like. Explain why, although the afrisenti table has 111,720 rows, the new joined table has 186,941 rows. *Note: Some systems may have trouble knitting the 'tweet' column due to the special characters present, so if this applies to you, feel free to add %>% select(-tweet) to your answer.***

```
inner_join(afrisenti %>% select(-tweet), language_countries)
```

```
## Joining with `by = join_by(language_iso_code)`
```

```
## Warning in inner_join(afrisenti %>% select(-tweet), language_countries): Detected an
unexpected many-to-many relationship between `x` and `y`.
## ℹ Row 21542 of `x` matches multiple rows in `y`.
## ℹ Row 1 of `y` matches multiple rows in `x`.
## ℹ If a many-to-many relationship is expected, set `relationship = "many-to-many"` to
silence this warning.
```

```
## # A tibble: 186,941 × 4
##    language_iso_code label    intended_use country
##    <chr>             <chr>    <chr>        <chr>
##  1 amh               negative dev          Ethiopia
##  2 amh               negative dev          Ethiopia
##  3 amh               negative dev          Ethiopia
##  4 amh               negative dev          Ethiopia
##  5 amh               negative dev          Ethiopia
##  6 amh               negative dev          Ethiopia
##  7 amh               negative dev          Ethiopia
##  8 amh               negative dev          Ethiopia
##  9 amh               negative dev          Ethiopia
## 10 amh               negative dev          Ethiopia
## # ℹ 186,931 more rows
```

The increase in the number of rows from 111,720 to 186,941 is due to the many-to-many relationship between the afrisenti and language_countries datasets. Each language in afrisenti can be associated with multiple countries in language_countries, leading to multiple rows in the joined dataset for each original row in afrisenti.

> e. **Make a table consisting only of the 8 languages appearing most frequently in the afrisenti table. You table should only have 8 rows, one for each of these languages.**

```
afrisenti_full <- afrisenti %>%
  left_join(languages, by = "language_iso_code")

language_counts <- afrisenti_full %>%
  count(language)

language_counts %>%
  arrange(desc(n)) %>%
  head(8)
```

```
## # A tibble: 8 × 2
##   language                   n
##   <chr>                  <int>
## 1 Hausa                  22152
## 2 Igbo                   15715
## 3 Yorùbá                 15127
## 4 Nigerian Pidgin        10556
## 5 Amharic                 9480
## 6 Moroccan Arabic/Darija  9038
## 7 Mozambican Portuguese   7492
## 8 Kinyarwanda             5155
```

f. **Filter the afrisenti table, using a join we learned this week, to only keep rows corresponding to one of the 8 languages that appears most frequently in the table. Hint: Use your table from the previous part. *Note: Some systems may have trouble knitting the 'tweet' column due to the special characters present, so feel free to add %>% select(-tweet) to your answer.***

```
language_counts <- afrisenti %>%
  count(language_iso_code, sort = TRUE)

top_languages <- language_counts %>%
  top_n(8, n) %>%
  select(language_iso_code)

filtered_afrisenti <- afrisenti %>%
  semi_join(top_languages, by = "language_iso_code") %>%
  select(-tweet)

filtered_afrisenti
```

```
## # A tibble: 94,715 × 3
##    language_iso_code label    intended_use
##    <chr>             <chr>    <chr>
##  1 amh               negative dev
##  2 amh               negative dev
##  3 amh               negative dev
##  4 amh               negative dev
##  5 amh               negative dev
##  6 amh               negative dev
##  7 amh               negative dev
##  8 amh               negative dev
##  9 amh               negative dev
## 10 amh               negative dev
## # ℹ 94,705 more rows
```

# Question 10 (6 points)

**Filter the flights data set to only contain flights along the 20 routes with the largest average arrival delays (of the flights that took off), where a route consists of both the origin airport and the destination airport. Hint: You may want to make an intermediate table to help you.**

```
flights_with_avg_delay <- flights %>%
  group_by(origin, dest) %>%
  summarise(avg_arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'origin'. You can override using the `.groups` ar
gument.
```

```
top_routes <- flights_with_avg_delay %>%
  arrange(desc(avg_arr_delay)) %>%
  slice_max(order_by = avg_arr_delay, n = 20)

filtered_flights <- flights %>%
  semi_join(top_routes, by = c("origin", "dest"))

head(filtered_flights)
```

```
## # A tibble: 6 × 29
##   origin dest  alt_origin alt_dest  year month   day dep_time sched_dep_time dep_dela
y arr_time sched_arr_time arr_delay carrier flight tailnum
##   <chr>  <chr>      <dbl>    <dbl> <int> <int> <int>    <int>          <int>     <dbl
>    <int>          <int>     <dbl> <chr>    <int> <chr>
## 1 EWR    MEM           18      341  2013     1     1      812            814        -
2     1040           1017        23 EV        4537 N17108
## 2 EWR    JAC           18     6451  2013     1     1      848            851        -
3     1155           1136        19 UA        1741 N27724
## 3 EWR    DCA           18       15  2013     1     1      929            929
0     1028           1042       -14 EV        4636 N11551
## 4 EWR    MKE           18      723  2013     1     1     1044           1045        -
1     1231           1212        19 EV        4322 N15555
## 5 EWR    PWM           18       77  2013     1     1     1056           1059        -
3     1203           1209        -6 EV        4479 N11544
## 6 LGA    CAK           22     1228  2013     1     1     1147           1155        -
8     1335           1327         8 FL         353 N932AT
## # ℹ 13 more variables: air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time
_hour <dttm>, alt_origin_origin <dbl>, alt_dest_dest <dbl>,
## #   alt_origin_origin_origin <dbl>, alt_dest_dest_dest <dbl>, alt_origin_origin_origi
n_origin <dbl>, alt_dest_dest_dest_dest <dbl>,
## #   alt_origin_origin_origin_origin_origin <dbl>, alt_dest_dest_dest_dest_dest <dbl>
```

# Question 11 (6 points)

> a. **Explain what R's intersect() function is, explain how it is different from an inner_join, and give a real-world example of when you might want to use it.**

The intersect() function returns common rows between two tables, unlike inner_join which merges columns. An example of when you might want to use it is when you have two tables of voters and you want to find the voters that are present in both tables.

> b. **Explain what R's setdiff() function is, explain how it is different from an anti_join, and give a real-world example of when you might want to use it.**

The setdiff() function returns rows in one table that are not present in another table, unlike anti_join which filters rows. An example of when it would be useful is when you have two tables of trinkets and you want to find the trinkets that are present in one table but not in the other.

# Question 12 (6 points)

> **Consider the following three data sets with more detailed information about three of the African languages considered above. More information about this data can be found at https://github.com/afrisenti-semeval/afrisent-semeval-2023/tree/main/data_with_annotators_labels#readme (https://github.com/afrisenti-semeval/afrisent-semeval-2023/tree/main/data_with_annotators_labels#readme).**

```
morrocan_arabic <- read_csv("https://raw.githubusercontent.com/afrisenti-semeval/afrisent-semeval-2023/main/data_with_annotators_labels/morrocan_arabic_individual_labels.csv")
```

```
## Rows: 6999 Columns: 8
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (4): text, label_1, label_2, label_3
## dbl (4): text_id, annotator_1, annotator_2, annotator_3
##
## 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.
```

```
algerian_arabic <- read_csv("https://raw.githubusercontent.com/afrisenti-semeval/afrisent-semeval-2023/main/data_with_annotators_labels/algerian_arabic_individual_labels.csv")
```

```
## Rows: 3097 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): annotator1, annotator2, annotator3
## dbl (1): tweet_id
##
## 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.
```

```
hausa <- read_csv("https://raw.githubusercontent.com/afrisenti-semeval/afrisent-semeval-2023/main/data_with_annotators_labels/hausa_individual_labels.csv")
```

```
## Rows: 30000 Columns: 8
## ── Column specification ────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────
## Delimiter: ","
## chr (4): text, label_1, label_2, label_3
## dbl (4): text_id, annotator_1, annotator_2, annotator_3
##
## 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. **Combine the morrocan_arabic and hausa data sets together into a single table in an appropriate way. You should be sure your resulting table contains information about which rows are observations about Morroccan Arabic and which rows are observations about Hausa.**

```
combined_data <- bind_rows(
  morrocan_arabic %>% mutate(language = "Morrocan Arabic"),
  hausa %>% mutate(language = "Hausa")
)

combined_data
```

```
## # A tibble: 36,999 × 9
##    text                                                               text_id annotat
or_1 annotator_2 annotator_3 label_1 label_2 label_3 language
##    <chr>                                                                <dbl>       <
dbl>       <dbl>       <dbl> <chr>   <chr>   <chr>   <chr>
##  1 "#nada0074 Jomo3a mobaraka inchallah 3liya we 3la la famille dyal… 7.18e17
79          72          73 Positi… Positi… Positi… Morroca…
##  2 "#WhatAboutArabArmy Lay lay lay lay lay lay la la la lay lay 😭😭… 1.13e18
79          72          73 Negati… Indete… Indete… Morroca…
##  3 "@nohita123 @Anyssa_Ch la daba homa li ghadi ykhtaro hna khas nsd… 5.72e17
72          79          73 Neutral Neutral Positi… Morroca…
##  4 "@sansuuna matbkhlich 3lina wakha ma3rt fin kati7o 3la had la9ata… 6.78e17
72          79          73 Neutral Neutral Positi… Morroca…
##  5 "@aminattttta o soltana fella wa3dat ibtissam boghniya"            5.99e17
73          72          79 Neutral Neutral Neutral Morroca…
##  6 "@Ihab_Amir ihaaab nta tstaleel la9ab o nta charaftii l maghreeb … 6.83e17
72          79          73 Positi… Positi… Positi… Morroca…
##  7 "@greenadilaida @FatihiW Merehba khouya adil chi poisson au four … 1.19e18
79          73          72 Neutral Positi… Positi… Morroca…
##  8 "Fach l prof dyal communication katsm3ek glti \"un video\" https:… 1.11e18
73          79          72 Neutral Neutral Neutral Morroca…
##  9 "@91Grosminey @IbtissamTiskat @fatiinatiskat @ali__shaddad @Fulla… 5.98e17
73          72          79 Negati… Negati… Negati… Morroca…
## 10 "@_BigBen__ @L7argouss @ucef79 kayn Chi man9diw ? Sme3t sou9 rass… 7.42e17
72          79          73 Neutral Neutral Neutral Morroca…
## # ℹ 36,989 more rows
```

> b. **Explain why you can't combine the morrocan_arabic and algerian_arabic tables together in the same way you did in the previous part for morrocan_arabic and hausa.**

You can't combine the morrocan_arabic and algerian_arabic tables together in the same way you did in the previous part for morrocan_arabic and hausa because the columns in the two tables are different.