Homework 4

Prepare your answers as a **single PDF file**.
Group work: You may work in groups of 1-3. Include all group member names in the PDF file. You may work with students in both sections (375-01, -02). Only one person in the group should submit to Canvas.
Due: check on Canvas.

1. Load the `nycflights13` library (will have to install the `nycflights13` package first) which contains flight arrival and departure data in a table called `flights`. Apply the tidyverse's data wrangling verbs to answer these questions. For each question, **give only the (one line as a data pipeline) code beginning with `flights %>%` ….**

1. Show the average arrival delay for all flights.
    a. `flights %>% summarise(mean(arr_delay, na.rm = TRUE))`
2. Show the average arrival delay for every departure city.
    a. `flights %>% group_by(origin) %>% summarise(mean(arr_delay, na.rm = TRUE))`
3. Show the average arrival delay for every departure-arrival city pair.
    a. `flights %>% group_by(origin, dest) %>% summarise(mean(arr_delay, na.rm = TRUE))`

2. Consider the two tables shown below called *bands* and *instruments*.

bands:

name	***lastname***	***band***	***year***
Mick	Jagger	Stones	1962
John	Lennon	Beatles	1960
Paul	McCartney	Beatles	1960
Paul	McCartney	Wings	1971

instruments:

artist	***artistname***	***plays***	***model***
John	Lennon	guitar	Gibson
Paul	McCartney	bass	Hofner
Keith	Richards	guitar	Fender
Paul	McCartney	bass	Hofner

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) `bands %>% inner_join(instruments)` [Hint: this is a trick question!]
    a) N/A, no common variables

b) `bands %>% inner_join(instruments, by=c(name="artist"))`
    a)

name	***lastname***	***band***	***year***	***artistname***	***plays***	***model***
John	Lennon	Beatles	1960	Lennon	guitar	Gibson

Paul	McCartney	Beatles	1960	McCartney	bass	Hofner
Paul	McCartney	Beatles	1960	McCartney	bass	Hofner
Paul	McCartney	Wings	1971	McCartney	bass	Hofner
Paul	McCartney	Wings	1971	McCartney	bass	Hofner

c) `bands %>% inner_join(instruments, by=c(name="artist", lastname="artistname"))`

a)

name	***lastname***	***band***	***year***	***plays***	***model***
John	Lennon	Beatles	1960	guitar	Gibson
Paul	McCartney	Beatles	1960	bass	Hofner
Paul	McCartney	Beatles	1960	bass	Hofner
Paul	McCartney	Wings	1971	bass	Hofner
Paul	McCartney	Wings	1971	bass	Hofner

d) `bands %>% left_join(instruments, by=c(name="artist", lastname="artistname"))`

a)

name	***lastname***	***band***	***year***	***plays***	***model***
Mick	Jagger	Stones	1962	N/A	N/A
John	Lennon	Beatles	1960	guitar	Gibson
Paul	McCartney	Beatles	1960	bass	Hofner
Paul	McCartney	Beatles	1960	bass	Hofner
Paul	McCartney	Wings	1971	bass	Hofner
Paul	McCartney	Wings	1971	bass	Hofner

e) `bands %>% inner_join(instruments, by=c(name="artist", lastname="artistname", year="plays"))`

a) N/A

3. 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**. **Hint**: ***First***, convert to a tidy table. All the questions can then be answered with a single data pipeline.

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

a) The billboard data is not tidy because each variable does not have its own column and each observation must have its own row. Also, each value needs its own cell. The week and its ranking should each have their own column in order to make this dataset tidy.

b) Give the code to convert `billboard` to a tidy table and store it in a tibble called `billboard_tidy`.

```
billboard_tidy <- billboard %>%
    pivot_longer(
        cols = starts_with("wk"),
        names_to = "week",
        names_prefix = "wk",
        values_to = "rank",
        values_drop_na = TRUE
    )
```

Answer the following questions with a single data pipeline beginning with `billboard_tidy %>% billboard`

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

```
billboard_tidy %>% group_by(track) %>% summarise(week = n()) %>%
arrange(-week) %>% View()
```

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

```
billboard_tidy %>% group_by(track) %>% summarise(week = n()) %>%
arrange(-week) %>% View()
```

e) Show for each track, its top rank

```
billboard_tidy %>% group_by(track) %>% summarise(top_rank = min(rank)) %>%
View()
```

f) List tracks in increasing order of its top rank

a) IF increasing means it increases as it goes down the table

i)
```
billboard_tidy %>% group_by(track) %>% summarise(top_rank =
min(rank)) %>% arrange(top_rank) %>% View()
```

b) IF increasing means it starts at its highest point and decreases as it goes down the table

i)
```
billboard_tidy %>% group_by(track) %>% summarise(top_rank =
min(rank)) %>% arrange(-top_rank) %>% View()
```

g) Show for each artist, their top rank

```
billboard_tidy %>% group_by(artist) %>% arrange(-rank) %>% summarise(top_rank
= min(rank)) %>% View()
```

h) List artists in increasing order of their top rank

a) IF increasing means it increases as it goes down the table

i) `billboard_tidy %>% group_by(artist) %>% summarise(top_rank = min(rank)) %>% arrange(top_rank) %>% View()`

b) IF increasing means it starts at its highest point and decreases as it goes down the table

i) `billboard_tidy %>% group_by(artist) %>% summarise(top_rank = min(rank)) %>% arrange(-top_rank) %>% View()`

i) List tracks that only spent one week in the charts

```
billboard_tidy %>% group_by(track) %>% summarise(week=n()) %>%
filter(week==1) %>% View()
```

j) List tracks that only spent one week in the charts along with its artist

```
billboard_tidy %>% group_by(artist, track) %>% summarise(week=n()) %>%
filter(week==1) %>% View()
```

4. [THIS TOPIC WILL BE COVERED ON TUE 2/28. ATTEMPT AFTER TUESDAY'S CLASS]

Consider the attached .csv file, "insurance_premiums.csv," which contains the Average Annual Single Premium per Enrolled Employee For Employer-Based Health Insurance[1]

Location	2013__Employee_Contribution	2013__Employer_Contribution	...	2018__Employee_Contribution	2018__Employer_Contribution
United States	1170	4401	...	1427	5288
Alabama	1379	3825	...	1453	4636
Alaska	1078	6291	...	1154	7278

The first few rows are shown above.

(a) The data is not "tidy". In 2-3 sentences, explain why.
The employee and employer contributions are separated and have their own columns for each year. This leads to having too many columns and a messy dataset.

(b) The goal is to convert this dataset to the tidy format shown below.

Location	Year	Employee_Contribution	Employer_Contribution
United States	2013	1170	4401
United States	2014	1234	4598
United States	2015	1255	4708
United States	2016	1325	4776
United States	2017	1415	4953
United States	2018	1427	5288

[1] Dataset from: https://www.kff.org/other/state-indicator/single-coverage/

Alabama	2013	1379	3825
Alabama	2014	1362	4164
Alabama	2015	1228	4505
Alabama	2016	1510	4026
Alabama	2017	1593	4482
Alabama	2018	1453	4636
…	…	…	…

Pivot **all** columns into **two** names_to columns, then pivot again! Specifically, do the following steps. *Show your code for each step.*

(i) Read the .csv file using **read_csv** (NOT read.csv) and store it in a table.

Hint: `premiums <- read_csv("insurance_premiums.csv")`

```
premiums <- read_csv("insurance_premiums.csv")
```

(ii) Pivot_longer all columns except Location into *two* `names_to` columns. This requires a `names_sep` to be specified. Read the help for pivot_longer().

```
premiums %>% pivot_longer(-Location, names_to = c("Year", "Contribution"),
names_sep = "__")
```

(iii) Pivot_wider a pair of columns from the previous step.

```
premiums %>% pivot_longer(-Location, names_to = c("Year", "Contribution"),
names_sep = "__") %>% pivot_wider(names_from = "Contribution")
```

(iv) Check that the resulting table matches the desired tidy format. Show the output of `str()` for the tidy table.

Hint: show output of `… %>% str()`

```
> premiums %>% pivot_longer(-Location, names_to = c("Year", "Contribution"), names_sep = "__") %>% pivot_wider(names_from =
"Contribution") %>% str()
tibble [312 × 4] (S3: tbl_df/tbl/data.frame)
 $ Location            : chr [1:312] "United States" "United States" "United States" "United States" ...
 $ Year                : chr [1:312] "2013" "2014" "2015" "2016" ...
 $ Employee_Contribution: num [1:312] 1170 1234 1255 1325 1415 ...
 $ Employer_Contribution: num [1:312] 4401 4598 4708 4776 4953 ...
```

Optional, no extra credit: Perform the above transformation using only *one* pivot_longer. Hint: this will use the special ".value" entry in the names_to argument. Read the help for pivot_longer().