Individual Final

This "final" exam will be relatively brief and, hopefully, pretty straightforward. Your task will be to follow the instructions below to explore a dataset summarizing job postings from various companies around the US. All of the job postings have something to do with data, and most of them are data science jobs.

Unlike the other individual assignments you've completed this semester, this assignment is to be completed on your own (though you are welcome to use the internet as well as any AI tools).

Most of the tasks you'll be asked to do should be pretty easy for you at this point in the semester. But, as usual, there are some new things I'm going to have you learn-by-doing in this assignment to give you a few more tools in your R tool belt. I'll describe these first in this summary section, and if you can get comfortable with each of these, each of the tasks you'll accomplish below will be easy.

There are 4 new things you'll need to be able to do, all of which are `ggplot` tricks that you should be able to find lots of examples of around the web. I'll list them and give you some background that should guide you:

1. `geom_col()` is a geom that we didn't learn how to use during the visualization section of the class. We did use a closely related one, though: `geom_bar()`, and the usage is nearly the same. The main difference (and the reason you'll use it extensively in this assignment) is that, by default, a `geom_col()` will display a bar chart derived directly from *pre-summarized* column values in the dataset, and you'll need to map both an `x` and a `y` aesthetic or else it will complain at you. (By contrast, the `geom_bar()` that we used in the past would do the summarizing for us, based on just an `x` aesthetic mapping of a category variable.) In practice, this makes `geom_col()` really useful when you've already summarized your data (either with a `group_by()` or even just with a `count()`), especially when you want to display a bar chart that summarizes something *other* than a simple count of each of the category values (for which `geom_bar()` is more convenient). Anyway, you can read more about the two functions and their differences in the `ggplot` documentation if you need some help.
2. You'll also need to hide the legend in each of the plots I ask you to create. This can be accomplished using a `legend.position` theme element.
3. In order to display the plots in the same way I did, you'll also need to learn about how to order the bars according a separate column rather than the category defining the bars. (You'll notice my sample plots are ordered from largest to smallest rather than alphabetically.)
4. For the few plots summarizing salaries (which are measured in dollars), you'll want to look into the `label_dollar()` label formatter from the `scales` package. Documentation here.

Aside from those new things, you should already have a good handle on everything else you'll be asked to do. I'll just note the other two things that I've added to every plot in my output so that you can easily match the look and feel:

- I added `theme_bw()` as usual.
- I also used `labs()` to add the title and to label the x and y axes.

(Also note: after reading in the data using the code provided, you won't need to clean or otherwise mess with the data in `job_data`.)

First, you'll need to create a summary tibble that shows the top 10 companies in the dataset in terms of the number of job postings they have listed in this dataset. (Note that there are 3 ties for the 10th spot, so your

tibble and the subsequent plot will actually show 12 companies.) The result should look like the preview below:

```
# A tibble: 12 × 2
   company                    n
   <chr>                  <int>
 1 I28 Technologies          63
 2 TikTok                    58
```

Save that summarized view of the data to a new tibble called `top_hiring_companies`, then use it to create a plot that matches the one found in `plots/plot_1.png`. Save this plot as `plot_1`.

1. You should now have both `top_hiring_companies` and `plot_1` in your environment.

Next, you'll need to create a summary tibble that shows the count of each of the simplified job titles in the dataset. (Note that there are two columns in the data related to job titles; you're aiming for the `job_simpl` one.) The result should look like the preview below:

```
# A tibble: 5 × 2
  job_simpl                   n
  <chr>                   <int>
1 data scientist           1197
2 data analyst              328
```

Save that summarized view of the data to a new tibble called `top_job_titles`, then use it to create a plot that matches the one found in `plots/plot_2.png`. Save this plot as `plot_2`.

2. You should now have both `top_job_titles` and `plot_2` in your environment.

Next, you'll need to create a summary tibble that sums each of the `_yn` skills columns. Each of those columns is a binary indicator of whether a given job description mentioned python, spark, etc. The result should look like the preview below:

```
# A tibble: 6 × 2
  skill     count
  <chr>     <dbl>
1 python_yn  1485
2 spark_yn    244
```

Save that summarized view of the data to a new tibble called `skill_counts`, then use it to create a plot that matches the one found in `plots/plot_3.png`. Save this plot as `plot_3`.

3. You should now have both `skill_counts` and `plot_3` in your environment.

Next, you'll need to create a summary tibble that shows the top 10 locations in the dataset in terms of the number of job postings associated with that location. (The reason there are 11 rather than 10 in the preview below is because there is a tie for the 10th spot.) The result should look like the preview below:

```
# A tibble: 11 × 2
   location              n
```

```
  <chr>             <int>
1 Remote              565
2 New York, NY        160
```

Save that summarized view of the data to a new tibble called `top_10_job_locations`, then use it to create a plot that matches the one found in `plots/plot_4.png`. Save this plot as `plot_4`.

4. You should now have both `top_10_job_locations` and `plot_4` in your environment.

Next, you'll need to create a plot (with no associated summary dataset because it can be derived directly from the `job_data` tibble) that summarizes the distribution of salaries by (simplified) job title, similar to the one found in `plots/plot_5.png`. Save this plot as `plot_5`.

5. You should now have `plot_5` in your environment.

Next, you'll need to create a summary tibble that shows the mean and median salaries by industry. (You can filter out any `NA`s in the `company_industry` column.) The result should look like the preview below:

```
# A tibble: 65 × 3
   company_industry                median_salary mean_salary
   <chr>                                   <dbl>       <dbl>
 1 Accounting & Tax                        62500      65591.
 2 Advertising & Public Relations          91214      95261.
```

Save that summarized view of the data to a new tibble called `industry_salary`, then use it to create a plot that matches the one found in `plots/plot_6.png`. Save this plot as `plot_6`. (Note that the tibble doesn't necessarily need to be ordered by either of the salary columns, but the bars on the plot do.)

6. You should now have both `industry_salary` and `plot_6` in your environment.

Next, you'll need to create a summary tibble that shows the mean and median salaries by location, but only for the top 10 locations (In terms of number of postings, not in terms of salaries, which means there will again be 11 locations in your results because of the tie for 10th place). The result should look like the preview below:

```
# A tibble: 11 × 3
   location          median_salary mean_salary
   <chr>                     <dbl>       <dbl>
 1 Alexandria, VA           102817     109075.
 2 Austin, TX               115874     103778.
```

Save that summarized view of the data to a new tibble called `top_locations_salary`, then use it to create a plot that matches the one found in `plots/plot_7.png`. Save this plot as `plot_7`. (Note that the tibble doesn't necessarily need to be ordered by either of the salary columns, but the bars on the plot do.)

7. You should now have both `top_locations_salary` and `plot_7` in your environment.

Final Cleanup and Submission

Final Run

Please do the following to make sure your code runs without errors and passes the naming checks by doing the following:

1. Restart your R session (Session >> Restart R).
2. Run your entire script from the beginning, watching for any errors along the way.
3. After your script is finished, make sure that you pass all tests in the code in the "Naming Checks" section at the bottom of your script.

Housekeeping

Once you have completed the steps in the prior section, please check all of the following and make adjustments as needed. (Failure to do the housekeeping steps below will likely cause an error in our grading process, and that will make it hard to give you the right credit for your hard work.)

1. If you used the `setwd()` command near the beginning of the script, please COMMENT OUT that line before committing and pushing your code.
2. Please also comment out any code where you are either using the `View()` or `glimpse()` functions. These cause issues with our grading procedures.
3. If you installed any new packages as you completed the assignment (using `install.packages()`), please comment those installation commands out as well.
4. Lastly, please ensure that you have **NO** absolute references in your code that are not commented out. (An absolute reference is something like: `/Users/YourName/Documents/GitHub/...` or `C:/GitHubProjects/...` .) These also throw errors and make it hard for us to grade your work.

Save, Commit, Push

You're now ready to do all three of the following:

1. Save your script.
2. Do a final commit with your git tool to stage your work for submission.
3. Push your changes up to your repository. And you're done!