Working on *guided projects* gives you hands on experience with real world examples, which also means they'll be more challenging than lessons. However, keep in mind that now you have more tools you can use to clean and transform data than you did at the beginning of this course, including:

- Vectorized string methods to clean string columns
- The apply(), map(), and applymap() methods to transform data
- The fillna(), dropna(), and drop() methods to drop missing or unnecessary values
- The melt() function to reshape data
- The concat() and merge() functions to combine data

If you get stuck, don't be afraid to:

- Review previous lessons.
- Use Google and StackOverflow to search for answers.

It's very difficult to understand every concept and memorize syntax the first time you complete a course, so don't be discouraged if you need to review lessons again or use Google to search for answers.

In this guided project, we'll work with exit surveys from employees of the Department of Education, Training and Employment (DETE) and the Technical and Further Education (TAFE) institute in Queensland, Australia. You can find the DETE exit survey data here. The original TAFE exit survey data is no longer available. We've made some slight modifications to the original datasets to make them easier to work with, including changing the encoding to UTF-8 (the original ones are encoded using cp1252.)

In this project, we'll play the role of data analyst and pretend our stakeholders want to know the following:

- Are employees who only worked for the institutes for a short period of time resigning due to some kind of dissatisfaction? What about employees who have been there longer?
- Are younger employees resigning due to some kind of dissatisfaction? What about older employees?

They want us to combine the results for *both* surveys to answer these questions. However, although both used the same survey template, one of them customized some of the answers. In the guided steps, we'll aim to do most of the data cleaning and get you started analyzing the first question.

A data dictionary wasn't provided with the dataset. In a job setting, we'd make sure to meet with a manager to confirm the definitions of the data. For this project, we'll use our general knowledge to define the columns.

Below is a preview of a couple columns we'll work with from the dete_survey.csv:

- ID: An id used to identify the participant of the survey
- SeparationType: The reason why the person's employment ended
- Cease Date: The year or month the person's employment ended
- DETE Start Date: The year the person began employment with the DETE

Below is a preview of a couple columns we'll work with from the tafe_survey.csv:

- Record ID: An id used to identify the participant of the survey
- Reason for ceasing employment: The reason why the person's employment ended
- LengthofServiceOverall. Overall Length of Service at Institute (in years): The length of the person's employment (in years)

Let's start by reading the datasets into pandas and exploring them.

Instructions

- Start by writing a paragraph in a markdown cell introducing the project and the dataset.
- Import the pandas and NumPy libraries.
- Read the dete_survey.csv CSV file into pandas, and assign it to the variable name dete_survey.
- Read the tafe_survey.csv CSV file into pandas, and assign it to the variable name tafe_survey.
- Use the DataFrame.info() and DataFrame.head() methods to print information about both dataframes, as well as the first few rows. Use other data exploration methods such as the Series.value_counts() and DataFrame.isnull() methods to explore the data and figure out some next steps.
    - Write a markdown cell briefly describing your observations.

Often, figuring out the steps you need to take to clean and reshape your data is the hardest part. If you couldn't find a clear path forward in the last screen, don't worry! We'll lay out the steps for you, but give you some room to start making your own decisions.

From our work in the previous screen, we can first make the following observations:

- The dete_survey dataframe contains 'Not Stated' values that indicate values are missing, but they aren't represented as NaN.
- Both the dete_survey and tafe_survey dataframes contain many columns that we don't need to complete our analysis.
- Each dataframe contains many of the same columns, but the column names are different.
- There are multiple columns/answers that indicate an employee resigned because they were dissatisfied.

To start, we'll handle the first two issues. Recall that we can use the pd.read_csv() function to specify values that should be represented as NaN. We'll use this function to fix the missing values first. Then, we'll drop columns we know we don't need for our analysis.

Instructions

- Read the dete_survey.csv CSV file into pandas again, but this time read the Not Stated values in as NaN.
    - To read Not Stated in as NaN, set the na_values parameter to Not Stated in the pd.read_csv() function.
    - Assign the result to the variable name dete_survey.
- Then, let's drop some columns from each dataframe that we won't use in our analysis to make the dataframes easier to work with.
    - Use the DataFrame.drop() method to drop the following columns from dete_survey: dete_survey.columns[28:49]. Remember to set the axis parameter equal to 1.
        - Assign the result to dete_survey_updated.
    - Use the DataFrame.drop() method to drop the following columns from tafe_survey: tafe_survey.columns[17:66]. Remember to set the axis parameter equal to 1.
        - Assign the result to tafe_survey_updated.
- Write a markdown cell explaining the changes you made and why.

Next, let's turn our attention to the column names. Each dataframe contains many of the same columns, but the column names are different. Below are some of the columns we'd like to use for our final analysis:

dete_survey	tafe_survey	Definition
ID	Record ID	An id used to identify the participant of the survey
SeparationType	Reason for ceasing employment	The reason why the participant's employment ended
Cease Date	CESSATION YEAR	The year or month the participant's employment ended
DETE Start Date		The year the participant began employment with the DETE
	LengthofServiceOverall. Overall Length of Service at Institute (in years)	The length of the person's employment (in years)
Age	CurrentAge. Current Age	The age of the participant
Gender	Gender. What is your Gender?	The gender of the participant

Because we eventually want to combine them, we'll have to standardize the column names. Recall that we can use the DataFrame.columns attribute along with vectorized string methods to update all of the columns at once. Here's an example from the last lesson:

happiness2017.columns = happiness2017.columns.str.replace('.', ' ').str.replace('\s+', ' ').str.strip().str.upper()

Instructions

- Rename the remaining columns in the dete_survey_updated dataframe.
    - Use the following criteria to update the column names:
        - Make all the capitalization lowercase.
        - Remove any trailing whitespace from the end of the strings.
        - Replace spaces with underscores ('_').
    - As an example, Cease Date should be updated to cease_date.
    - Remember you can use the DataFrame.columns attribute to print an array of the existing column names.

- Use the DataFrame.rename() method to update the columns below in tafe_survey_updated. Don't worry about the rest of the column names right now - we'll handle them later.
    - 'Record ID': 'id'
    - 'CESSATION YEAR': 'cease_date'
    - 'Reason for ceasing employment': 'separationtype'
    - 'Gender. What is your Gender?': 'gender'
    - 'CurrentAge. Current Age': 'age'
    - 'Employment Type. Employment Type': 'employment_status'
    - 'Classification. Classification': 'position'
    - 'LengthofServiceOverall. Overall Length of Service at Institute (in years)': 'institute_service'
    - 'LengthofServiceCurrent. Length of Service at current workplace (in years)': 'role_service'
- Use the DataFrame.head() method to look at the current state of the dete_survey_updated and tafe_survey_updated dataframes and make sure your changes look good.
- Write a markdown cell explaining the changes you made and why.

In the last screen, we renamed the columns that we'll use in our analysis. Next, let's remove more of the data we don't need.

Recall that our end goal is to answer the following question:

- Are employees who have only worked for the institutes for a short period of time resigning due to some kind of dissatisfaction? What about employees who have been at the job longer?

If we look at the unique values in the separationtype columns in each dataframe, we'll see that each contains a couple of different separation types. For this project, we'll only analyze survey respondents who *resigned*, so their separation type contains the string 'Resignation'.

If you're interested in a challenge, try to complete the project using *all* of the separation types instead - you'll find more issues to work through in the data cleaning process.

Note that dete_survey_updated dataframe contains multiple separation types with the string 'Resignation':

- Resignation-Other reasons
- Resignation-Other employer
- Resignation-Move overseas/interstate

Instructions

- Use the Series.value_counts() method to review the unique values in the separationtype column in both dete_survey_updated and tafe_survey_updated.
- In each of dataframes, select only the data for survey respondents who have a Resignation separation type.
    - Remember that the dete_survey_updated dataframe contains three Resignation separation types. We want to select all of them.
    - Use the DataFrame.copy() method on the result to avoid the SettingWithCopy Warning.
    - Assign the result for dete_survey_updated to dete_resignations.
    - Assign the result for tafe_survey_updated to tafe_resignations.
- Write a markdown paragraph explaining the changes you made and why.

Now, before we start cleaning and manipulating the rest of our data, let's verify that the data doesn't contain any major inconsistencies (to the best of our knowledge). When you're working with real world data, don't assume that the data you're analyzing isn't corrupted in some way!

It may not always be possible to catch all of these errors, but by making sure the data seems reasonable to the best of our knowledge, we can stop ourselves from completing a data analysis project that winds up being useless because of bad data.

In this step, we'll focus on verifying that the years in the cease_date and dete_start_date columns make sense. However, we encourage you to check the data for other issues as well!

- Since the cease_date is the last year of the person's employment and the dete_start_date is the person's first year of employment, it wouldn't make sense to have years after the current date.
- Given that most people in this field start working in their 20s, it's also unlikely that the dete_start_date was before the year 1940.

If we have many years higher than the current date or lower than 1940, we wouldn't want to continue with our analysis, because it could mean there's something very wrong with the data. If there are a small amount of values that are unrealistically high or low, we can remove them.

Instructions

- Check the years in each dataframe for logical inconsistencies.
    - First, clean the cease_date column in dete_resignations.
        - Use the Series.value_counts() method to view the unique values in the cease_date column.
        - Use vectorized string methods to extract the year. As a reminder, here is the full list.
        - Use the Series.astype() method method to convert the type to a float.
    - Use the Series.value_counts() to check the values in the cease_date and dete_start_date columns in dete_resignations and the cease_date column in tafe_resignations.
        - Because Series.value_counts() returns a series, we can use Series.sort_index() method with ascending= True or False to view the highest and lowest values with their counts.
    - You can also plot the values of any numeric columns with a boxplot to identify any values that look wrong.
- Write a markdown paragraph explaining your findings.

From the work we did in the last screen, we can verify:

1. There aren't any major issues with the years.
2. The years in each dataframe don't span quite the same number of years. We'll leave it up to your discretion to drop any years you don't think are needed for the analysis.

Now that we've verified the years in the dete_resignations dataframe, we'll use them to create a new column. Recall that our end goal is to answer the following question:

- Are employees who have only worked for the institutes for a short period of time resigning due to some kind of dissatisfaction? What about employees who have been at the job longer?

In the Human Resources field, the length of time an employee spent in a workplace is referred to as their years of *service*.

You may have noticed that the tafe_resignations dataframe already contains a "service" column, which we renamed to institute_service. In order to analyze both surveys together, we'll have to create a corresponding institute_service column in dete_resignations.

Do we have data that can be used to calculate the length of time the employee spent in their workplace? Take a minute to review dete_resignations once more and see if you can answer this question before moving on.

Instructions

- Create an institute_service column in dete_resignations
    - Create a new column named institute_service in dete_resignations.
        - Subtract the dete_start_date from the cease_date. Assign the result to a new column named institute_service.
- Write a markdown paragraph explaining the changes you made and why.

In the last screen, we created a new institute_service column that we'll use to analyze survey respondents according to their length of employment.Next, we'll identify any employees who resigned because they were dissatisfied.

Below are the columns we'll use to categorize employees as "dissatisfied" from each dataframe. If you disagree, feel free to modify them! Just make sure you explain *why* you made that decision.

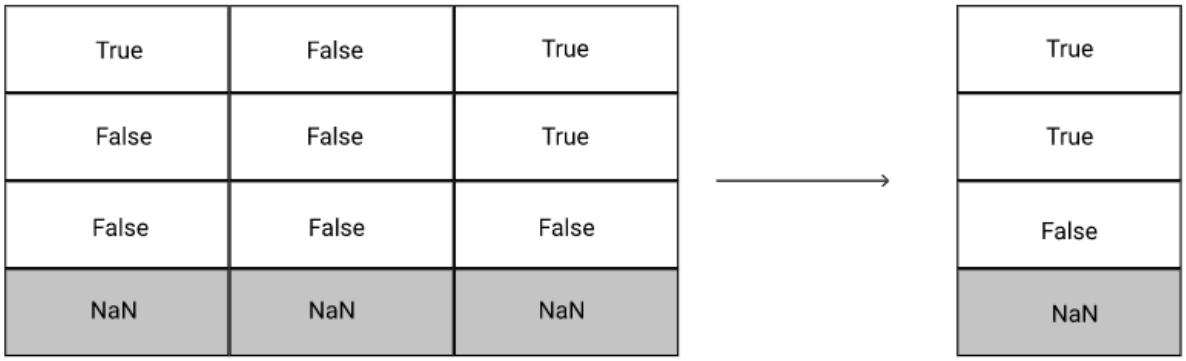
1. tafe_survey_updated:
    - Contributing Factors. Dissatisfaction
    - Contributing Factors. Job Dissatisfaction
2. dete_survey_updated:
    - job_dissatisfaction
    - dissatisfaction_with_the_department
    - physical_work_environment
    - lack_of_recognition
    - lack_of_job_security
    - work_location
    - employment_conditions
    - work_life_balance
    - workload

If the employee indicated any of the factors above caused them to resign, we'll mark them as dissatisfied in a new column.

To create the new column, we'll do the following:

1. Convert the values in the 'Contributing Factors. Dissatisfaction' and 'Contributing Factors. Job Dissatisfaction' columns in the tafe_resignations dataframe to True, False, or NaN values.
2. If any of the columns listed above contain a True value, we'll add a True value to a new column named dissatisfied. To accomplish this, we'll use the DataFrame.any() method to do the following:
    - Return True if *any* element in the selected columns above is True
    - Return False if *none* of the elements in the selected columns above is True

- Return NaN if the value is NaN



Here's the syntax we can use:

df.any(axis=1, skipna=False)

After our changes, the new dissatisfied column will contain just the following values:

- True: indicates a person resigned because they were dissatisfied with the job
- False: indicates a person resigned because of a reason other than dissatisfaction with the job
- NaN: indicates the value is missing

Instructions

- Use the Series.value_counts() method to view the values in the 'Contributing Factors. Dissatisfaction' and 'Contributing Factors. Job Dissatisfaction' in the tafe_resignations dataframe.
- Update the values in the 'Contributing Factors. Dissatisfaction' and 'Contributing Factors. Job Dissatisfaction' in the tafe_resignations dataframe so that each contains only True, False, or NaN values.
    - Write a function named update_vals that makes the following changes:
        - If the value is NaN, return np.nan. You can use the following criteria to check that a value is NaN: pd.isnull(val).
        - If the value is '-', return False.
        - For any other value, return True.
    - Use the DataFrame.applymap() method to apply the function above to the 'Contributing Factors. Dissatisfaction' and 'Contributing Factors. Job Dissatisfaction' in the tafe_resignations dataframe.
        - Remember that we need to pass the update_vals function into the df.applymap() method *without* parentheses.
- Use the df.any() method as described above to create a dissatisfied column in BOTH the tafe_resignations and dete_resignations dataframes.
- Use the df.copy() method to create a copy of the results and avoid the SettingWithCopy Warning. Assign the results to dete_resignations_up and tafe_resignations_up.
- Write a markdown paragraph explaining the changes you made and why.

To recap, we've accomplished the following:

- Renamed our columns
- Dropped any data not needed for our analysis
- Verified the quality of our data
- Created a new institute_service column
- Cleaned the Contributing Factors columns
- Created a new column indicating if an employee resigned because they were dissatisfied in some way

Now, we're finally ready to combine our datasets! Our end goal is to aggregate the data according to the institute_service column, so when you combine the data, think about how to get the data into a form that's easy to aggregate.

Instructions

- First, let's add a column to each dataframe that will allow us to easily distinguish between the two.
    - Add a column named institute to dete_resignations_up. Each row should contain the value DETE.
    - Add a column named institute to tafe_resignations_up. Each row should contain the value TAFE.
- Combine the dataframes. Assign the result to combined.
- Recall that we still have some columns left in the dataframe that we don't need to complete our analysis. Use the DataFrame.dropna() method to drop any columns with less than 500 non null values.
    - Remember that you can drop columns with less than a certain number of non null values with the thresh parameter.
    - Assign the result to combined_updated.
- Write a markdown paragraph explaining the changes you made and why.

Now that we've combined our dataframes, we're almost at a place where we can perform some kind of analysis! First, though, we'll have to clean up the institute_service column. This column is tricky to clean because it currently contains values in a couple different forms:

NaN 88

Less than 1 year 73

1-2 64

3-4 63

5-6 33

11-20 26

5.0 23

1.0 22

7-10 21

0.0 20

...

To analyze the data, we'll convert these numbers into categories. We'll base our analysis on this article, which makes the argument that understanding employee's needs according to career stage instead of age is more effective.

We'll use the slightly modified definitions below:

- New: Less than 3 years at a company
- Experienced: 3-6 years at a company
- Established: 7-10 years at a company
- Veteran: 11 or more years at a company

Let's categorize the values in the institute_service column using the definitions above.

Instructions

- First, we'll extract the years of service from each value in the institute_service column.
    - Use the Series.astype() method to change the type to 'str'.
    - Use vectorized string methods to extract the years of service from each pattern. You can find the full list of vectorized string methods here.
    - Double check that you didn't miss extracting any digits.
    - Use the Series.astype() method to change the type to 'float'.
- Next, we'll map each value to one of the career stage definitions above.
    - Create a function that maps each year value to one of the career stages above.
        - Remember that you'll have to handle missing values separately. You can use the following code to check if a value is NaN where val is the name of the value: pd.isnull(val).
    - Use the Series.apply() method to apply the function to the institute_service column. Assign the result to a new column named service_cat.
- Write a markdown paragraph explaining the changes you made and why.

In the last screen, we created a service_cat column, that categorizes employees according to the amount of years spent in their workplace:

- New: Less than 3 years at a company
- Experienced: 3-6 years at a company
- Established: 7-10 years at a company
- Veteran: 11 or more years at a company

Now, let's finally do our first piece of analysis! We'll help you fill in missing values in the dissatisfied column and then aggregate the data to get you started, but note that we still have additional missing values left to deal with. This is meant to be an initial introduction to the analysis, *not* the final analysis.

Recall that the dissatisfied column consists of Boolean values, meaning they're either True or False. Methods such as the df.pivot_table() method actually treat Boolean values as integers, so a True value is considered to be 1 and a False value is considered to be 0. That means that we can aggregate the dissatisfied column and calculate the number of people in each group, the percentage of people in each group, etc.

Instructions

- Use the Series.value_counts() method to confirm if the number of True and False in the dissatisfied column. Set the dropna parameter to False to also confirm the number of missing values.
- Use the DataFrame.fillna() method to replace the missing values in the dissatisfied column with the value that occurs most frequently in this column, either True or False.
- Use the DataFrame.pivot_table() method to calculate the percentage of dissatisfied employees in each service_cat group.
    - Since a True value is considered to be 1, calculating the mean will also calculate the percentage of dissatisfied employees. The default aggregation function is the mean, so you can exclude the aggfunc argument.
- Use the DataFrame.plot() method to plot the results. Set the kind parameter equal to bar to create a bar chart.
    - Make sure to run %matplotlib inline beforehand to show your plots in the notebook.
- Write a markdown paragraph briefly describing your observations.

In this guided project, we experienced that in order to extract any meaningful insights from our data, we had to perform many data cleaning tasks. In order to create one visualization (and not even the final one), we completed the following tasks:

- Explored the data and figured out how to prepare it for analysis
- Corrected some of the missing values
- Dropped any data not needed for our analysis
- Renamed our columns
- Verified the quality of our data
- Created a new institute_service column
- Cleaned the Contributing Factors columns
- Created a new column indicating if an employee resigned because they were dissatisfied in some way
- Combined the data
- Cleaned the institute_service column
- Handled the missing values in the dissatisfied column
- Aggregated the data

Our work here is far from done! We recommend that you continue with the following steps:

- Decide how to handle the rest of the missing values. Then, aggregate the data according to the service_cat column again. How many people in each career stage resigned due to some kind of dissatisfaction?
- Clean the age column. How many people in each age group resgined due to some kind of dissatisfaction?
- Instead of analyzing the survey results together, analyze each survey separately. Did more employees in the DETE survey or TAFE survey end their employment because they were dissatisfied in some way?