

Let's start a chat for my Week 8 Independent Lab

Sure—let's get started on your Week 8 Independent Lab. What are the specific tasks or questions you need to work through? If you have the lab instructions or any code/data you're working with, please share them and we'll tackle each step together.



Here are the instructions. Do not solve problems or output any code unless i explicitly ask you to: "#
Independent Lab: Advanced Data Manipulation

In this assignment, you will use advanced functions in **pandas** to query and filter data. These functions reduce workload by simplifying and reducing code.

Please use the [gh_turnover_four_projects.csv]
(data/gh_turnover_four_projects.csv) file provided in the data folder to complete the tasks outline below.

***Data Summary*:** This dataset contains information about four projects hosted on GitHub and the contributors to those projects over a period of time. The dataset represents a subset of data that has been combined and aggregated from the data published by Vasilescu et al. in [their 2015 Mining Software Repositories conference paper] (<http://doi.org/10.1109/MSR.2015.77>). The data in gh_turnover_four_projects.csv were prepared by me (Olivia B. Newton) for an analysis of turnover in GitHub projects. To make the dataset more manageable for the assignment, I have selected four projects from my original dataset which includes data for over 22,000 projects.

Rename Columns

Import the file `gh_turnover_four_projects.csv`.
Once imported, rename columns like so:

- * domain change to `project_domain`
- * language change to `project_language`
- * windows change to `project_age`
- * window_idx change to `quarter`
- * num_team change to `team_size`
- * num_commits change to `project_commits`
- * blau_gender change to `gender_blau`
- * Gini_gh_ten change to `gh_tenure_gini`

- * Core1 change to core_dev
- * commits change to user_commits
- * propCommits change to commit_proportion
- * github_tenure change to gh_tenure
- * leavesNextQ change to leaves_next_q
- * Leave all other column names as is

Data Wrangling

It is time to practice your data wrangling skills with Python! Please perform the following tasks. ****Run all code in your notebook, then save your notebook with output****.

Note, some of the tasks ask you to use regular expressions. We have not covered this in class, so I have provided the regex syntax. If parentheses are included as part of the regex, include it. For example, I provide `(^p)` below. Thus, your code would be `.filter(regex = ' (^p) ')`.

1. Calculate the mean of user_commits.
2. Calculate the median of user_commits.
3. Select all columns that start with a `*p*` (i.e., `(^p)`) ****or**** contain an `*g*` (i.e., `(g)`). Save it as a new DataFrame named `turnover_new`. Output the columns in your notebook.
4. Using your newly created DataFrame `turnover_new`, select rows in which `project_commits` is greater than 9,000.
5. Using pandas piping notation, perform the previous two operations together and save it as a new DataFrame `turnover_newer`. This means you should select columns that start with a `*p*` or contain an `*g*` and select rows in which `project_commits` is greater than 9,000.

For the following tasks, ***do not use*** `turnover_new` or `turnover_newer`. Instead, use the original DataFrame you initially created from the provided file.

1. Using pandas piping notation, select all columns that end with the letter `*e*` (i.e., `e$`) or contain the letter `*s*` (i.e., `s`). Additionally, select rows in which `gh_tenure` is greater than or equal to 1,991.
2. Use the query you just performed. You will calculate the mean and median of `user_commits`. How does it compare to the answers above in which you calculated the mean and median? Please provide your answer in a Markdown cell within your notebook."

Understood—I won't provide any code or solutions until you explicitly ask. Which step would you like to tackle first? For example, shall we start with the column-renaming instructions or move straight to computing the mean and median of `user_commits`?



```
"Please use the [gh_turnover_four_projects.csv]
(data/gh_turnover_four_projects.csv) file provided in
the data folder to complete the tasks outline below. "
What's wrong with my code? "import csv
```

```
path =
'/Users/AlexMcKenzie/Library/CloudStorage/GoogleDriv
e-alex.c.mckenzie@gmail.com/My Drive/Education/UM
Graduate
School/BGEN632/week8labs/data/gh_turnover_four_pro
jects.csv'

turnover_df.columns = (
    teams_df.columns
    .str.strip()
    .str.strip('\"')
)

print(turnover_df.columns.tolist())"
```

Thought for a few seconds >

There are a few issues here:

1. You never read the CSV into a pandas DataFrame

You've only imported the built-in `csv` module, but then you immediately try to use pandas-style operations (`.str.strip()`) on a DataFrame that doesn't exist yet.

2. You're referencing the wrong variable name

In your assignment you write

```
python  Copy  Edit

turnover_df.columns = ( teams_df.columns .str.strip()
.str.strip('\"') )
```

but you never defined `teams_df` anywhere — you meant to use `turnover_df` , not `teams_df` .

3. You need to import and use pandas

To load and manipulate a CSV with a DataFrame you should do something like:

```
python  Copy 
```

```
import pandas as pd
turnover_df = pd.read_csv(path)
turnover_df.columns = (turnover_df.columns .str.strip()
                        .str.strip('\"'))
print(turnover_df.columns.tolist())
```

Once you

- `import pandas as pd`
- actually call `pd.read_csv(path)` into `turnover_df`
- and reference `turnover_df.columns` instead of `teams_df.columns`

your column-stripping logic will run without errors.



forget about stripping anything -- that was a code snippet from a prior assignment. I just want to load `"/Users/AlexMcKenzie/Library/CloudStorage/GoogleDrive-alex.c.mckenzie@gmail.com/My Drive/Education/UM Graduate School/BGEN632/week8labs/data/gh_turnover_four_projects.csv"`

Thought for 4 seconds >

Here's the minimal code to load that CSV into a pandas DataFrame (no stripping or extra steps):

python



```
import pandas as pd
path = '/Users/AlexMcKenzie/Library/CloudStorage/GoogleDrive-alex.c.mckenzie@gmail.com/My Drive/Education/UM Graduate School/BGEN632/week8labs/data/gh_turnover_four_projects.csv'
turnover_df = pd.read_csv(path) # to verify it loaded correctly
turnover_df.head()
```



"We have not covered this in class, so I have provided the regex syntax. If parentheses are included as part of the regex, include it. For example, I provide `(^p)` below. Thus, your code would be `.filter(regex = '^p')`."

1. Calculate the mean of `user_commits`. "

Thought for 6 seconds >

You can get the mean of that column with pandas' built-in `.mean()` method. For example, once you've loaded your DataFrame as `turnover_df`:

python



```
turnover_df['user_commits'].mean()
```

This will return the average number of commits per user. Let me know if you'd like the full notebook cell or further explanation on how to interpret the result!



```
"-----  
-----"
```

```
KeyError                                Traceback (most recent call  
last)
```

```
File /opt/anaconda3/lib/python3.12/site-  
packages/pandas/core/indexes/base.py:3805, in  
Index.get_loc(self, key)
```

```
    3804 try:  
-> 3805     return self._engine.get_loc(casted_key)  
    3806 except KeyError as err:
```

```
File index.pyx:167, in  
pandas._libs.index.IndexEngine.get_loc()
```

```
File index.pyx:196, in  
pandas._libs.index.IndexEngine.get_loc()
```

```
File pandas/_libs/hashtable_class_helper.pxi:7081, in  
pandas._libs.hashtable.PyObjectHashTable.get_item()
```

```
File pandas/_libs/hashtable_class_helper.pxi:7089, in  
pandas._libs.hashtable.PyObjectHashTable.get_item()
```

```
KeyError: 'user_commits'
```

The above exception was the direct cause of the following exception:

```
KeyError                                Traceback (most recent call  
last)
```

```
Cell In[10], line 1  
----> 1 turnover_df['user_commits'].mean()
```

```
File /opt/anaconda3/lib/python3.12/site-  
packages/pandas/core/frame.py:4102, in  
DataFrame.__getitem__(self, key)
```

```
    4100 if self.columns.nlevels > 1:  
    4101     return self._getitem_multilevel(key)  
-> 4102 indexer = self.columns.get_loc(key)  
    4103 if is_integer(indexer):  
    4104     indexer = [indexer]
```

```

File /opt/anaconda3/lib/python3.12/site-
packages/pandas/core/indexes/base.py:3812, in
Index.get_loc(self, key)
    3807     if isinstance(casted_key, slice) or (
    3808         isinstance(casted_key, abc.Iterable)
    3809         and any(isinstance(x, slice) for x in
casted_key)
    3810     ):
    3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
    3813 except TypeError:
    3814     # If we have a listlike key,
_check_indexing_error will raise
    3815     # InvalidIndexError. Otherwise we fall through
and re-raise
    3816     # the TypeError.
    3817     self._check_indexing_error(key)

KeyError: 'user_commits'

```

Thought for 7 seconds >

The `KeyError: 'user_commits'` means your DataFrame doesn't actually have a column by that name yet. You loaded the CSV but haven't renamed the original "commits" column to "user_commits," so pandas can't find it.

To confirm, you can inspect your columns:

python

 Copy  Edit

```
turnover_df.columns.tolist()
```

You'll see that the contributor-level commits column is still called "commits". You have two options:

1. **Rename first** (per the lab's instructions), then call `.mean()` on `user_commits`.
2. **Skip renaming** and just compute the mean of the existing "commits" column:

python

 Copy  Edit

```
turnover_df['commits'].mean()
```

Either approach will give you the average commits per user—but until you rename, there is no "user_commits" column to reference.

```

['project_id',
 'domain',
 'language',
 'windows',

```

```
'window_idx',
'num_team',
'num_commits',
'blau_gender',
'Gini_gh_ten',
'user_id',
'Core1',
'commits',
'propCommits',
'gender',
'github_tenure',
'leavesNextQ']
```

Thought for 7 seconds >

Your DataFrame still has the original column name ("commits") rather than "user_commits" , so pandas can't find `turnover_df['user_commits']` .

You have two paths forward:

1. Calculate the mean on the existing column

Simply use the "commits" column for now:

python



Copy



Edit

```
turnover_df['commits'].mean()
```

2. Rename then calculate

First rename "commits" → "user_commits" , then call `.mean()` on it:

python



Copy



Edit

```
turnover_df.rename(columns={'commits':'user_commits'},
inplace=True) turnover_df['user_commits'].mean()
```

Let me know which approach you'd like (or if you want the exact code cell), and I'll provide it.



```
but i changed all the columns names with "new_df =
turnover_df.rename(columns={
    'domain':'project_domain',
    'language':'project_language',
    'windows':'project_age',
    'window_idx':'quarter',
    'num_team':'team_size',
    'num_commits':'project_commits',
    'blau_gender':'gender_blau',
    'Gini_gh_ten':'gh_tenure_gini',
    'Core1':'core_dev',
```

```

'commits':'user_commits',
'propCommits':'commit_proportion',
'github_tenure':'gh_tenure',
'leavesNextQ':'leaves_next_q'
})
new_df". And it seemed to confirm the changes: "
project_id    project_domain    project_language
project_age    quarter    team_size
project_commits    gender_blau    gh_tenure_gini
user_id    core_dev    user_commits    commit_proportion
gender    gh_tenure    leaves_next_q
0    1101    OTHER    Python    23    22    26    22688
0.000    0.422259    1614129    1    22138
0.975758    male    426    0
1    1101    OTHER    Python    23    22    26    22688
0.000    0.422259    1614101    0    124    0.005465
male    46    0
2    1101    OTHER    Python    23    22    26    22688
0.000    0.422259    94412    0    53    0.002336
male    941    1
3    1101    OTHER    Python    23    22    26    22688
0.000    0.422259    88459    0    49    0.002160
male    1870    0
4    1101    OTHER    Python    23    22    26    22688
0.000    0.422259    738787    0    45    0.001983
male    781    0
...    ...    ...    ...    ...    ...    ...    ...    ...    ...
...    ...    ...    ...    ...    ...
468    24293    DEV_FRAME    Python    24    23    12
1655    0.153    0.292578    938914    0    1
0.000604    male    326    1
469    24293    DEV_FRAME    Python    24    24    12
1224    0.000    0.229926    11648    0    1
0.000817    male    893    0
470    24293    DEV_FRAME    Python    24    24    12
1224    0.000    0.229926    179354    0    1
0.000817    male    1238    0
471    24293    DEV_FRAME    Python    24    24    12
1224    0.000    0.229926    438642    0    1
0.000817    male    1582    0
472    24293    DEV_FRAME    Python    24    24    12
1224    0.000    0.229926    5268    0    1
0.000817    male    1903    0"

```

Your `KeyError` is happening because you only created the `user_commits` column on `new_df` , not on your original `turnover_df` . So when you do

```
python
```



Copy



Edit


```
turnover_df['user_commits'].mean()
```

there is no such column on `turnover_df` . You have two simple fixes:

1. **Compute the mean on** `new_df` (which does have `user_commits`):

python



Copy



Edit

```
new_df['user_commits'].mean()
```

2. **Rename in place** on your original DataFrame and then call `.mean()` on it:

python



Copy



Edit

```
turnover_df.rename(columns={'commits':'user_commits'},  
inplace=True) turnover_df['user_commits'].mean()
```

Either approach will give you the mean of the (now-renamed) commits column. Let me know which you'd like to use, or if you'd like the full cell written out!



```
"turnover_df.rename(columns={  
    'domain':'project_domain',  
    'language':'project_language',  
    'windows':'project_age',  
    'window_idx':'quarter',  
    'num_team':'team_size',  
    'num_commits':'project_commits',  
    'blau_gender':'gender_blau',  
    'Gini_gh_ten':'gh_tenure_gini',  
    'Core1':'core_dev',  
    'commits':'user_commits',  
    'propCommits':'commit_proportion',  
    'github_tenure':'gh_tenure',  
    'leavesNextQ':'leaves_next_q'  
})  
new_df" project_id    project_domain  
project_language project_age quarter team_size  
project_commits  gender_blau gh_tenure_gini  
user_id  core_dev user_commits commit_proportion  
gender  gh_tenure  leaves_next_q  
0    1101 OTHER  Python    23    22    26    22688  
0.000    0.422259    1614129  1    22138  
0.975758 male    426  0  
1    1101 OTHER  Python    23    22    26    22688  
0.000    0.422259    1614101  0    124  0.005465  
male    46  0  
2    1101 OTHER  Python    23    22    26    22688  
0.000    0.422259    94412    0    53  0.002336  
male    941  1
```

```

3    1101 OTHER   Python   23    22    26    22688
0.000    0.422259    88459    0    49    0.002160
male    1870    0
4    1101 OTHER   Python   23    22    26    22688
0.000    0.422259    738787    0    45    0.001983
male    781    0

```

```

...    ...    ...    ...    ...    ...    ...    ...    ...    ...
...    ...    ...    ...    ...    ...
"turnover_df['user_commits'].mean()" "-----
-----
---

```

KeyError Traceback (most recent call last)

File /opt/anaconda3/lib/python3.12/site-packages/pandas/core/indexes/base.py:3805, in Index.get_loc(self, key)

```

    3804 try:
-> 3805     return self._engine.get_loc(casted_key)
    3806 except KeyError as err:

```

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas/_libs/hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas/_libs/hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'user_commits'

The above exception was the direct cause of the following exception:

KeyError Traceback (most recent call last)

Cell In[13], line 1
 ----> 1 turnover_df['user_commits'].mean()

File /opt/anaconda3/lib/python3.12/site-packages/pandas/core/frame.py:4102, in DataFrame.__getitem__(self, key)

```

    4100 if self.columns.nlevels > 1:
    4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
    4103 if is_integer(indexer):
    4104     indexer = [indexer]

```

```

File /opt/anaconda3/lib/python3.12/site-
packages/pandas/core/indexes/base.py:3812, in
Index.get_loc(self, key)
    3807     if isinstance(casted_key, slice) or (
    3808         isinstance(casted_key, abc.Iterable)
    3809         and any(isinstance(x, slice) for x in
casted_key)
    3810     ):
    3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
    3813 except TypeError:
    3814     # If we have a listlike key,
_check_indexing_error will raise
    3815     # InvalidIndexError. Otherwise we fall through
and re-raise
    3816     # the TypeError.
    3817     self._check_indexing_error(key)

KeyError: 'user_commits'

```

The reason you're still getting a `KeyError` is that your call to

```
python
```



Copy



Edit

```
turnover_df.rename(columns={...})
```

does *not* modify `turnover_df` in place (and you didn't reassign it). You instead captured the renamed DataFrame in `new_df`, but left `turnover_df` untouched — so it still has no `user_commits` column.

You have two quick fixes:

1. Compute on your renamed DataFrame

```
python
```



Copy



Edit

```

# you already have this new_df = turnover_df.rename(columns={ ... })
# now call mean on new_df new_df['user_commits'].mean()

```

2. Rename in place (or reassign) and then use turnover_df

```
python
```



Copy



Edit

```

# either... turnover_df.rename(columns={'commits':'user_commits',
..., inplace=True) # or... turnover_df = turnover_df.rename(columns=
{'commits':'user_commits', ...}) # now this will work
turnover_df['user_commits'].mean()

```

Pick whichever fits your workflow.





```
"turnover_df = turnover_df.rename(columns=
{'commits': 'user_commits', ...})" "Cell In[17], line 1
    turnover_df = turnover_df.rename(columns=
{'commits': 'user_commits', ...})
                                                                    ^
SyntaxError: invalid character '...' (U+2026)"
```

The problem is that the “...” character (U+2026) isn’t valid Python syntax—it was just a placeholder in my example. Python expects a real dict of column mappings, not an ellipsis.

If you only need to rename the `commits` column, you can do:

```
python
```

 Copy  Edit

```
turnover_df = turnover_df.rename(columns={'commits': 'user_commits'})
```

Or, to rename in place:

```
python
```

 Copy  Edit

```
turnover_df.rename(columns={'commits': 'user_commits'}, inplace=True)
```

If you want to rename all of your columns at once, you must spell out each mapping, for example:

```
python
turnover_df = turnover_df.rename(columns={ 'domain':'project_domain',
'language':'project_language', 'windows':'project_age',
'window_idx':'quarter', 'num_team':'team_size',
'num_commits':'project_commits', 'blau_gender':'gender_blau',
'Gini_gh_ten':'gh_tenure_gini', 'Core1':'core_dev',
'commits':'user_commits', 'propCommits':'commit_proportion',
'github_tenure':'gh_tenure', 'leavesNextQ':'leaves_next_q' })
```

Make sure you remove any "..." placeholders and include only real key:value pairs.



Select all columns that start with a *p* (i.e., (^p))
 or contain an *g* (i.e., (g)). Save it as a new
 DataFrame named `turnover_new`. Output the columns
 in your notebook.

Thought for 10 seconds >

python Copy Edit

```
# Select all columns that start with "p" or contain "g" turnover_new =  
new_df.filter(regex='(^p)|(g)') # Output the resulting column names  
turnover_new.columns.tolist()
```



Using your newly created DataFrame `turnover_new`, select rows in which `project_commits` is greater than 9,000.

Thought for 5 seconds >

python

Copy Edit

```
# Select rows in turnover_new where project_commits > 9000  
turnover_new_filtered = turnover_new[turnover_new['project_commits'] >  
9000] # Display the filtered DataFrame turnover_new_filtered
```



Using pandas piping notation, perform the previous two operations together and save it as a new DataFrame `turnover_newer`. This means you should select columns that start with a `*p*` or contain an `*g*` and select rows in which `project_commits` is greater than 9,000.

Thought for 11 seconds >

python

Copy Edit

```
turnover_newer = ( new_df .filter(regex='(^p)|(g)') .loc[lambda df:  
df['project_commits'] > 9000] ) turnover_newer
```



For the following tasks, *do not use* `turnover_new` or `turnover_newer`. Instead, use the original DataFrame you initially created from the provided file.

1. Using pandas piping notation, select all columns that end with the letter `*e*` (i.e., `e$`) or contain the letter `*s*` (i.e., `s`). Additionally, select rows in which `gh_tenure` is greater than or equal to 1,991.
2. Use the query you just performed. You will calculate the mean and median of `user_commits`. How does it compare to the answers above in which you calculated the mean and median? Please provide your answer in a Markdown cell within your notebook.

Thought for 5 seconds >

python

 Copy  Edit

```
# 1. Select columns ending with "e" or containing "s", and rows with
gh_tenure ≥ 1991 filtered_df = ( turnover_df .filter(regex='(e$)|s')
.loc[lamba df: df['gh_tenure'] >= 1991] ) # 2. Calculate mean and
median of user_commits on this filtered subset mean_filtered =
filtered_df['user_commits'].mean() median_filtered =
filtered_df['user_commits'].median() mean_filtered, median_filtered
```

markdown

 Copy  Edit

```
**Comparison of `user_commits` Statistics** - **Overall (all rows)** -
Mean of `user_commits`: *<overall_mean>* - Median of `user_commits`: *
<overall_median>* - **Filtered (gh_tenure ≥ 1991)** - Mean of
`user_commits`: *{mean_filtered}* - Median of `user_commits`: *
{median_filtered}* Compared to the overall statistics, the filtered
dataset's mean is *higher/lower* and its median is *higher/lower*,
indicating that contributors with at least 1991 days of GitHub tenure
tend to have *more/fewer* commits on average.
```