```
import numpy as np
import pandas as pd
```

DATASET : BOSTON : 100

```
# Load the CSV files into dataframes
users_df = pd.read_csv('/content/users2.csv')
repositories_df = pd.read_csv('/content/repositories2.csv')
```

```
# Display the first few rows to confirm
users_df.head(20)
```

	login	name	company	location	email	hireable	bio	public_repos	followers	following
0	brianyu28	Brian Yu	NaN	Boston, MA	brian@brianyu.me	NaN	Software developer and educator	35	13203	13
1	PatrickAlphaC	Patrick Collins	Cyfrin	Boston	NaN	NaN	Smart Contract Engineer, Auditor, and Educator	272	9670	43
2	KeithGalli	Keith Galli	NaN	Boston, MA	NaN	True	YouTube Content Creator :).	53	5679	1
3	CharlesCreativeContent	Shawn Charles	Amazon	Boston, MA	NaN	True	Software Engineer building Tech Communities	83	5054	1092
4	timbl	Tim Berners-Lee	@inrupt	Boston MA USA	timbl@w3.org	NaN	NaN	18	4850	69
5	bahmutov	Gleb Bahmutov	NaN	Boston, MA	gleb.bahmutov@gmail.com	NaN	JavaScript ninja, image processing expert, sof...	1245	4796	25
6	migueldeicaza	Miguel de Icaza	Xibbon	Boston, MA.	miguel@gnome.org	NaN	NaN	193	4692	69
7	rwaldron	Rick Waldron	NaN	Boston, MA	waldron.rick@gmail.com	NaN	He/him.	799	4407	53
8	nikomatsakis	Niko Matsakis	NaN	Boston, MA	niko@alum.mit.edu	NaN	NaN	253	3910	0
9	lh3	Heng Li	DFCI & Harvard University	Boston, MA, USA	lh3@me.com	NaN	NaN	122	3875	8
10	cowboy	Ben Alman	NaN	Boston, MA	cowboy@rj3.net	NaN	pronoun.is/he/him	134	3542	19
11	jlooper	Jen Looper	@Amazon	Boston	NaN	NaN	Head of Academic Advocacy, AWS. Author with Wi...	133	3179	35
12	ccoenraets	Christophe Coenraets	Salesforce.com	Boston	ccoenraets@gmail.com	NaN	NaN	145	2973	0
13	rstudio	RStudio	NaN	Boston, MA	info@rstudio.org	NaN	NaN	352	2698	0
14	pluskid	Chiyuan Zhang	MIT	Boston, MA	pluskid@gmail.com	NaN	NaN	30	2520	0
15	leonnoel	Leon Noel	Resilient Coders	Boston	NaN	NaN	Managing Director Of Engineering @ Resilient C...	129	2362	44

Next steps: Generate code with `users_df` | View recommended plots | New interactive sheet

```
# Display the first few rows to confirm
repositories_df.head(20)
```

	login	full_name	created_at	stargazers_count	watchers_count	language	has_projects	has_wiki	license_name
0	brianyu28	brianyu28/accompaniment	2017-10-31T20:20:05Z	0	0	TeX	True	True	NaN
1	brianyu28	brianyu28/alliance	2016-07-03T23:08:52Z	2	2	CSS	True	True	NaN
2	brianyu28	brianyu28/authorship	2018-09-06T20:46:18Z	15	15	Python	True	True	NaN
3	brianyu28	brianyu28/blender	2020-05-10T23:04:07Z	9	9	Python	True	True	NaN
4	brianyu28	brianyu28/brianyu28	2021-04-18T23:10:34Z	11	11	NaN	True	True	NaN
5	brianyu28	brianyu28/byd3	2017-05-17T18:09:51Z	1	1	JavaScript	True	True	NaN
6	brianyu28	brianyu28/chronology	2018-08-15T02:01:27Z	12	12	TypeScript	True	True	NaN
7	brianyu28	brianyu28/classroometrics	2022-08-10T00:35:48Z	15	15	Python	True	True	NaN
8	brianyu28	brianyu28/countdowns	2021-05-09T02:33:40Z	11	11	TypeScript	True	True	NaN
9	brianyu28	brianyu28/courseboards	2017-07-28T00:49:29Z	5	5	JavaScript	True	True	NaN
10	brianyu28	brianyu28/cs50	2019-09-17T15:00:36Z	19	19	HTML	True	True	NaN
11	brianyu28	brianyu28/csguidebook	2018-07-02T02:30:02Z	8	8	TypeScript	True	True	NaN
12	brianyu28	brianyu28/dispatch	2017-08-	39	39	Rust	True	True	GNU General

Next steps: Generate code with repositories_df | View recommended plots | New interactive sheet

```
users_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 469 entries, 0 to 468
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   login         469 non-null    object
 1   name          467 non-null    object
 2   company       304 non-null    object
 3   location      469 non-null    object
 4   email         236 non-null    object
 5   hireable      112 non-null    object
 6   bio           315 non-null    object
 7   public_repos  469 non-null    int64 
 8   followers     469 non-null    int64 
 9   following     469 non-null    int64 
 10  created_at    469 non-null    object
dtypes: int64(3), object(8)
memory usage: 40.4+ KB
```

```
repositories_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42498 entries, 0 to 42497
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   login             42498 non-null  object
 1   full_name         42498 non-null  object
 2   created_at        42498 non-null  object
 3   stargazers_count  42498 non-null  int64 
 4   watchers_count    42498 non-null  int64 
 5   language          32300 non-null  object
 6   has_projects      42498 non-null  bool  
 7   has_wiki          42498 non-null  bool  
 8   license_name      22323 non-null  object
dtypes: bool(2), int64(2), object(5)
memory usage: 2.4+ MB
```

DATA CLEANING

```
# Clean up the 'company' field
users_df['company'] = users_df['company'].str.strip()        # Remove whitespace
users_df['company'] = users_df['company'].str.lstrip('@')    # Strip leading '@'
users_df['company'] = users_df['company'].str.upper()        # Convert to uppercase

# Format the 'hireable' column specifically as 'true', 'false', or empty string if null
users_df['hireable'] = users_df['hireable'].apply(lambda x: 'true' if x is True else ('false' if x is False else ''))
```

```
# Save the cleaned file to confirm changes
users_df.to_csv('cleaned_users2.csv', index=False)

# Display a sample of the cleaned data to verify
users_df.head()
```

	login	name	company	location	email	hireable	bio	public_repos	followers	following	created_at
0	brianyu28	Brian Yu	NaN	Boston, MA	brian@brianyu.me		Software developer and educator	35	13203	13	2015-11-29T07:25:29Z
1	PatrickAlphaC	Patrick Collins	CYFRIN	Boston	NaN		Smart Contract Engineer, Auditor, and Educator	272	9670	43	2019-08-19T14:13:41Z
		Keith		Boston			YouTube				2013-12-

Next steps: Generate code with users_df | View recommended plots | New interactive sheet

```
#  Format boolean fields to be 'true', 'false', or empty string for nulls
repositories_df['has_projects'] = repositories_df['has_projects'].apply(lambda x: 'true' if x is True else ('false' if x is False else ''))
repositories_df['has_wiki'] = repositories_df['has_wiki'].apply(lambda x: 'true' if x is True else ('false' if x is False else ''))

# Save the cleaned file to confirm changes
repositories_df.to_csv('cleaned_repositories2.csv', index=False)

# Display a sample to confirm the output
repositories_df.head()
```

	login	full_name	created_at	stargazers_count	watchers_count	language	has_projects	has_wiki	license_name
0	brianyu28	brianyu28/accompaniment	2017-10-31T20:20:05Z	0	0	TeX	true	true	NaN
1	brianyu28	brianyu28/alliance	2016-07-03T23:08:52Z	2	2	CSS	true	true	NaN
2	brianyu28	brianyu28/authorship	2018-09-06T20:46:18Z	15	15	Python	true	true	NaN
3	brianyu28	brianyu28/blender	2020-05-10T23:04:07Z	9	9	Python	true	true	NaN
4	brianyu28	brianyu28/brianyu28	2021-04-18T23:10:34Z	11	11	NaN	true	true	NaN

Next steps: Generate code with repositories_df | View recommended plots | New interactive sheet

QUESTIONS

1. Who are the top 5 users in Boston with the highest number of followers? List their login in order, comma-separated.

```
top_5_users = users_df.nlargest(5, 'followers')['login'].tolist()
top_5_users_str = ','.join(top_5_users)
top_5_users_str
```

2. Who are the 5 earliest registered GitHub users in Boston? List their login in ascending order of created_at, comma-separated.

```
users_df['created_at'] = pd.to_datetime(users_df['created_at'])
earliest_5_users = users_df.nsmallest(5, 'created_at')['login'].tolist()
earliest_5_users_str = ','.join(earliest_5_users)
earliest_5_users_str
```

3. What are the 3 most popular license among these users? Ignore missing licenses. List the license_name in order, comma-separated.

```
top_3_licenses = repositories_df['license_name'].dropna().value_counts().head(3).index.tolist()
top_3_licenses_str = ','.join(top_3_licenses)
top_3_licenses_str
```

4. Which company do the majority of these developers work at?

```
most_common_company = users_df['company'].dropna().mode()[0]
most_common_company
```

5. Which programming language is most popular among these users?

```
most_popular_language = repositories_df['language'].dropna().mode()[0]
most_popular_language
```

6. Which programming language is the second most popular among users who joined after 2020?

```
second_most_popular_language = repositories_df[repositories_df['created_at'] > '2020-01-01']['language'].dropna().value_counts().index[1]
second_most_popular_language
```

7. Which language has the highest average number of stars per repository?

```
highest_avg_stars_language = repositories_df.groupby('language')['stargazers_count'].mean().idxmax()
highest_avg_stars_language
```

8. Let's define leader_strength as followers / (1 + following). Who are the top 5 in terms of leader_strength? List their login in order, comma-separated.

```
users_df['leader_strength'] = users_df['followers'] / (1 + users_df['following'])
top_5_leader_strength = users_df.nlargest(5, 'leader_strength')['login'].tolist()
','.join(top_5_leader_strength)
```

9. What is the correlation between the number of followers and the number of public repositories among users in Boston? Correlation between followers and repos (to 3 decimal places, e.g. 0.123 or -0.123)

```
correlation_followers_repos = users_df['followers'].corr(users_df['public_repos'])
round(correlation_followers_repos, 3)
```

```
0.168
```

10. Does creating more repos help users get more followers? Using regression, estimate how many additional followers a user gets per additional public repository. Regression slope of followers on repos (to 3 decimal places, e.g. 0.123 or -0.123)

```
from sklearn.linear_model import LinearRegression

X = users_df['public_repos'].values.reshape(-1, 1)
y = users_df['followers'].values

model = LinearRegression().fit(X, y)
slope = round(model.coef_[0], 3)
slope
```

```
1.192
```

11. Do people typically enable projects and wikis together? What is the correlation between a repo having projects enabled and having wiki enabled? Correlation between projects and wiki enabled (to 3 decimal places, e.g. 0.123 or -0.123)

```
# Create duplicate columns for computation
repositories_df['has_projects_computed'] = repositories_df['has_projects'].apply(lambda x: 1 if x == 'true' else 0)
repositories_df['has_wiki_computed'] = repositories_df['has_wiki'].apply(lambda x: 1 if x == 'true' else 0)

correlation_projects_wiki = repositories_df['has_projects_computed'].corr(repositories_df['has_wiki_computed'])
round(correlation_projects_wiki, 3)
```

```
0.334
```

12. Do hireable users follow more people than those who are not hireable? Average of following per user for hireable=true minus the average following for the rest (to 3 decimal places, e.g. 12.345 or -12.345)

```
hireable_following_avg = users_df[users_df['hireable'] == 'true']['following'].mean()
non_hireable_following_avg = users_df[users_df['hireable'] != 'true']['following'].mean()

difference_following_avg = round(hireable_following_avg - non_hireable_following_avg, 3)
difference_following_avg
```

```
111.969
```

13. Some developers write long bios. Does that help them get more followers? What's the impact of the length of their bio (in Unicode words, split by whitespace) with followers? (Ignore people without bios) Regression slope of followers on bio word count (to 3 decimal places, e.g. 12.345 or -12.345)

```
from sklearn.linear_model import LinearRegression

users_with_bios = users_df[users_df['bio'].notna()].copy()
users_with_bios.loc[:, 'bio_word_count'] = users_with_bios['bio'].apply(lambda x: len(x.split()))

X = users_with_bios['bio_word_count'].values.reshape(-1, 1)
y = users_with_bios['followers'].values

model = LinearRegression().fit(X, y)
bio_slope = round(model.coef_[0], 3)
bio_slope
```

```
-5.301
```

14. Who created the most repositories on weekends (UTC)? List the top 5 users' login in order, comma-separated Users login

```
repositories_df['created_at'] = pd.to_datetime(repositories_df['created_at'])
weekend_repos = repositories_df[repositories_df['created_at'].dt.dayofweek >= 5]
top_5_weekend_creators = weekend_repos['login'].value_counts().head(5).index.tolist()
top_5_users_str = ','.join(top_5_weekend_creators)
top_5_users_str
```

15. Do people who are hireable share their email addresses more often? [fraction of users with email when hireable=true] minus [fraction of users with email for the rest] (to 3 decimal places, e.g. 0.123 or -0.123)

```
hireable_with_email_fraction = users_df[users_df['hireable'] == 'true']['email'].notna().mean()
non_hireable_with_email_fraction = users_df[users_df['hireable'] != 'true']['email'].notna().mean()
email_share_difference = round(hireable_with_email_fraction - non_hireable_with_email_fraction, 3)
email_share_difference
```

```
0.113
```

16. Let's assume that the last word in a user's name is their surname (ignore missing names, trim and split by whitespace.) What's the most common surname? (If there's a tie, list them all, comma-separated, alphabetically) Most common surname(s)

```
users_with_names = users_df[users_df['name'].notna()].copy()
users_with_names['surname'] = users_with_names['name'].str.strip().apply(lambda x: x.split()[-1])
surname_counts = users_with_names['surname'].value_counts()
max_count = surname_counts.max()
most_common_surnames = surname_counts[surname_counts == max_count].index.tolist()
','.join(sorted(most_common_surnames))
```

```
'Williams'
```

The End

Start coding or generate with AI.