GENERAL ASSEMBLY

INTRO TO DATA SCIENCE

LECTURE 4: DATABASES

LAST TIME:

- INTRO TO MACHINE LEARNING
- PYTHON LIBRARIES

QUESTIONS?

AGENDA

I. INTRO TO DATABASES
II. RELATIONAL DATABASES
III.NOSQL DATABASES

EXERCISES:
III. MYSQL AND MONGO TUTORIALS

I. INTRO TO DATABASES

Databases are a **structured** data source optimized for efficient **retrieval and storage.**

Databases are a **structured** data source optimized for efficient **retrieval and persistent storage.**

structured: we have to pre-define organization strategy

retrieval: the ability to read data out

storage: the ability to write data and save it

Databases are a **structured** data source optimized for efficient **retrieval and persistent storage.**

structured: we have to pre-define organization strategy

retrieval: the ability to read data out

storage: the ability to write data and save it

REVIEW

1. For a database, what is retrieval and storage, and how could they be important?
2. What are the two current main “forms” of databases?

II. RELATIONAL DATABASES

Relational databases are traditionally organized in the following manner:

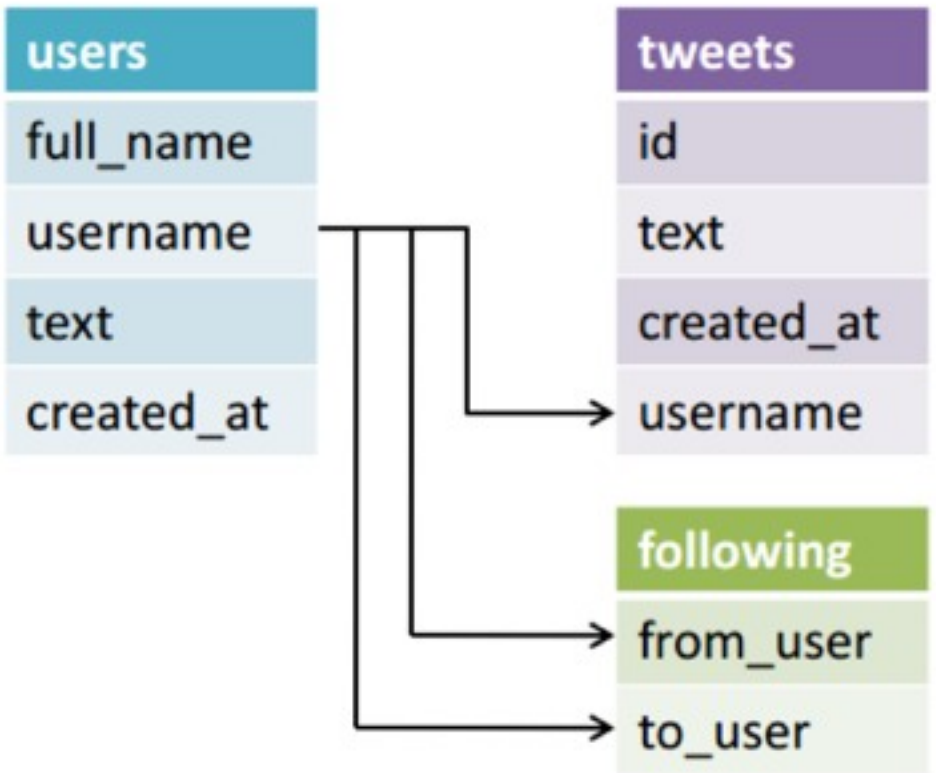
A database has tables which represent individual entities or objects

Tables have predefined schema – rules that tell it what the data will look like

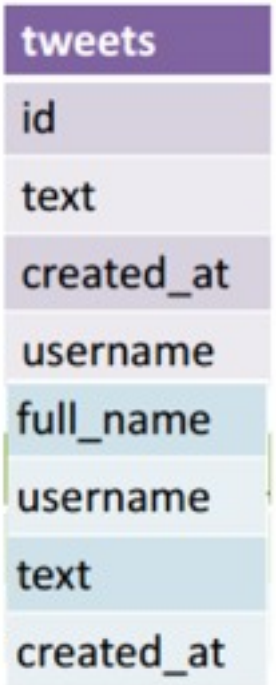
Each table should have a **primary key** column – a unique identifier for that row

Each table should have a **primary key** column – a unique identifier for that row

Additionally each table can have a **foreign key** column – an id that links this to another table.


users
full_name
username
text
created_at
tweets
id
text
created_at
username
following
from_user
to_user

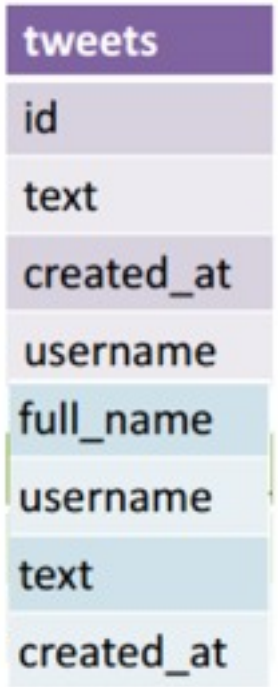
We could have had a table structure as follows:
Why is this different?



We could have had a table structure as follows:
Why is this different?

We would repeat the user information in each row.

This is called **denormalization**.



Normalized Data: Many tables to reduce redundant or repeated data in a table

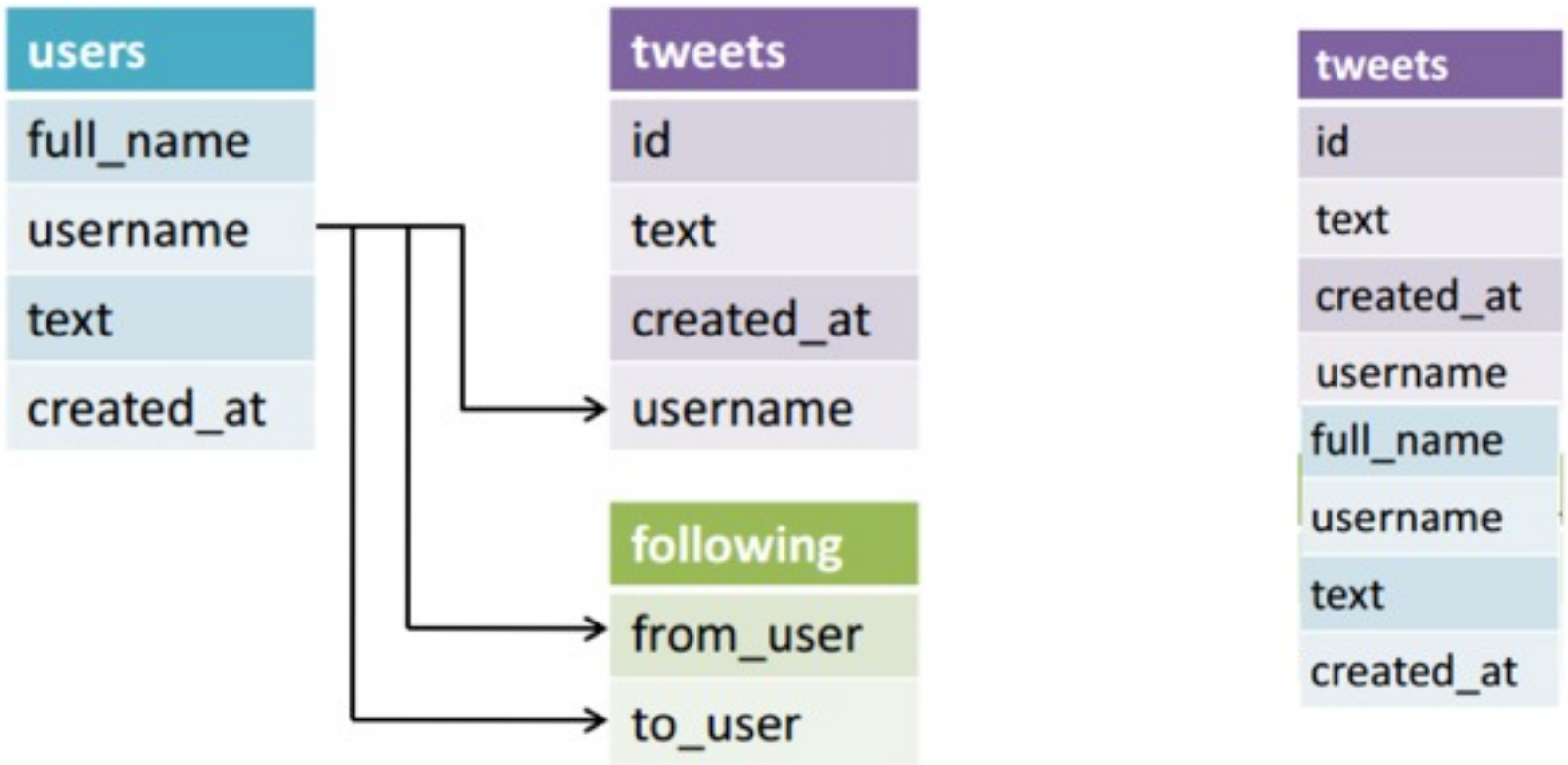
Denormalized Data: Wide data with fields often repeated but removes the need to join together multiple tables

Normalized Data: Many tables to reduce redundant or repeated data in a table

Denormalized Data: Wide data with fields often repeated but removes the need to join together multiple tables

This is a trade off of speed vs storage.

NORMALIZED VS DENONORMALIZED

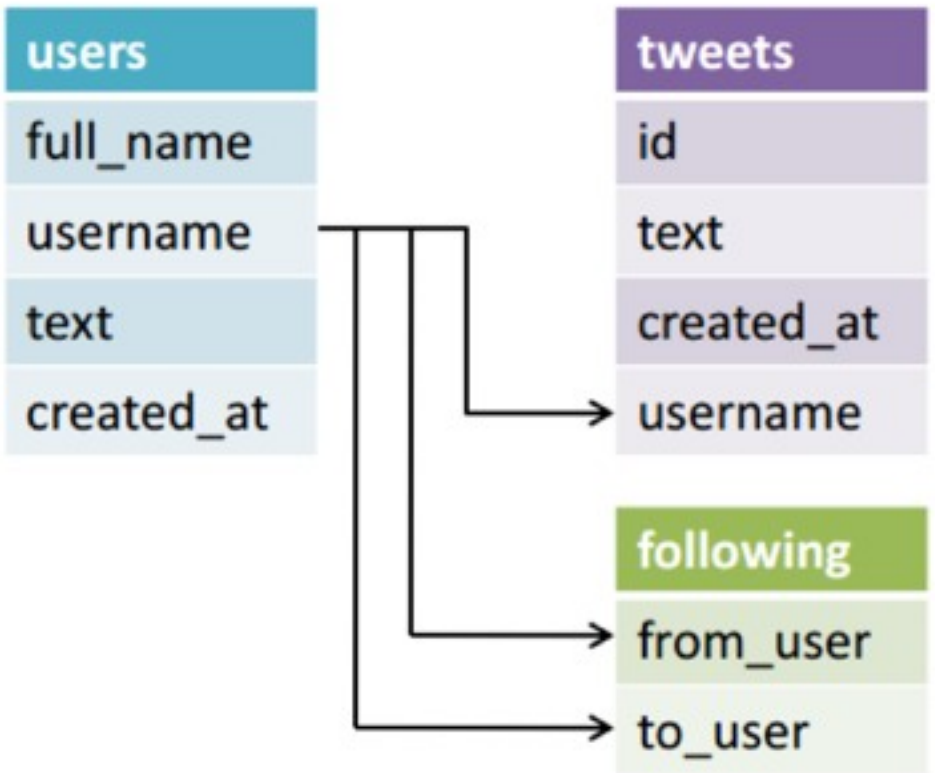


Q: How do we commonly evaluate databases?

read-speed vs write-speed
space considerations
(and many, many other criteria)

Q: Why are normalized tables (possibly) slower to read?

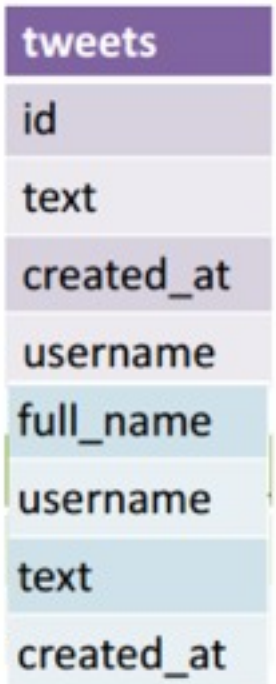
A: We'll have to get data from multiple tables to answer some questions


users
full_name
username
text
created_at
tweets
id
text
created_at
username
following
from_user
to_user

Q: Why are denormalized tables (possibly) slower to write?

A: We'll have to write more information on each write

NORMALIZED VS DENONORMALIZED



SQL is a query language to load, retrieve, and update data in relational databases

Most commonly known SQL-like Databases include:
Oracle
MySQL
PostgreSQL

SELECT: Allows you to retrieve information from a table

Syntax:

```
SELECT col1, col2
FROM table WHERE [some condition]
```

Example:

SELECT poll_title, poll_date FROM polls WHERE romney_pct > obama_pct

GROUP BY: Allows you to aggregate information.

Syntax:

```
SELECT col1, AVG(col2)
FROM table GROUP BY col1
```

Example:

```
SELECT poll_date, AVG(obama_pct)
FROM polls GROUP BY poll_date
```

GROUP BY: Allows you to aggregate information.

Syntax:

```
SELECT col1, AVG(col2)
FROM table GROUP BY col1
```

Example:

```
SELECT poll_date, AVG(obama_pct)
FROM polls GROUP BY poll_date
```

GROUP BY: Allows you to aggregate information.

Syntax:

```
SELECT col1, AVG(col2)
FROM table GROUP BY col1
```

There are usually a few common built-in operations:

SUM, AVG, MIN, MAX, COUNT

JOIN: Allows you to combine multiple tables

Syntax:

```
SELECT t1.c1, t1.c2, t2.c2
FROM t1 JOIN t2 ON t1.c1 = t2.c2
```

JOIN: Allows you to combine multiple tables

Syntax:

```
SELECT t1.c1, t1.c2, t2.c2
FROM t1 JOIN t2 ON t1.c1 = t2.c2
```

INSERT: Allows you to **add** data to tables

Syntax:

```
INSERT INTO table1 (col1, col2)
VALUES (...)

INSERT INTO classroom (first_name, last_name)
VALUES (‘John’, ‘Doe’)
```

LAB: MYSQL QUERYING

III. NO-SQL DATABASES

NoSQL databases are a new trend in databases

NoSQL databases are a new trend in databases

The name **NoSQL** refers to the lack of a relational structure between stored objects.

Most importantly they attempt to minimize the need for **JOIN** operations, or solve other data needs

Memcached

Apache HBase

Cassandra

MongoDB

Hadoop

```
Memcached       ::    Livejournal

Apache HBase    ::    Google BigTable

Cassandra       ::    Amazon Dynamo

MongoDB         ::    10Gen

Hadoop          ::    Google MapReduce
```

Memcached was:

- Developed by LiveJournal
- Distributed key-value store (like a Python Dict)
- Supports two **very fast** operations: **get** and **set**

Memcached is best used for storing application configuration settings, and essential **caching** those settings.

Cassandra was:

- Developed by Facebook
- Messages application and Inbox Search
- Key-Value (ish)
  - Supports query by key or key range
- Very fast writing speeds
- Useful for record keeping, logging

Mongo was:

- Developed by 10Gen (now MongoDB, Inc)
- Document and Collection Structure
- BSON (JSON-like) Storage system
- Aggregation Framework

IV. APIS AND JSON

Mongo's document structure is highly based off of JSON.

JSON (JavaScript Object Notation) is a borrowed JavaScript form turned into a string that can be passed between applications.

JSON are passed through applications as **strings**, and converted into native objects per language.

```
>>> someFile = open('/Users/epodojil/GA_Data_Science/a.json').read()
>>> print json.dumps(someFile)
"{\n    \"glossary\": {\n        \"title\": \"example glossary\",\n    \"GlossDiv\": {\n            \"title\": \"S\",\n      \"
\"SGML\",\n          \"SortAs\": \"SGML\",\n          \"GlossTerm\": \"Standard Generalized Markup Language\",\n          \"Acr
ef\": {\n                        \"para\": \"A meta-markup language, used to create markup languages such as DocBook.\",\n
\"GlossSee\": \"markup\"\n                }\n            }\n        }\n    }\n}"
>>> print someFile
{
    "glossary": {
        "title": "example glossary",
    "GlossDiv": {
            "title": "S",
      "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
                    },
          "GlossSee": "markup"
                }
            }
        }
    }
}
>>> print json.loads(someFile)
{u'glossary': {u'GlossDiv': {u'GlossList': {u'GlossEntry': {u'GlossDef': {u'GlossSeeAlso': [u'GML', u'XML'], u'para': u'A meta-
': u'markup', u'Acronym': u'SGML', u'GlossTerm': u'Standard Generalized Markup Language', u'Abbrev': u'ISO 8879:1986', u'SortAs
```

```
>>> someFile = open('/Users/epodojil/GA_Data_Science/a.json').read()
>>> print json.dumps(someFile)
"{\n    \"glossary\": {\n        \"title\": \"example glossary\",\n    \"GlossDiv\": {\n            \"title\": \"S\",\n      \"
\"SGML\",\n          \"SortAs\": \"SGML\",\n          \"GlossTerm\": \"Standard Generalized Markup Language\",\n          \"Acr
ef\": {\n                    \"para\": \"A meta-markup language, used to create markup languages such as DocBook.\",\n
\"GlossSee\": \"markup\"\n                }\n            }\n        }\n    }\n}"
>>> print someFile
{
    "glossary": {
        "title": "example glossary",
    "GlossDiv": {
            "title": "S",
      "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
                    },
          "GlossSee": "markup"
                }
            }
        }
    }
}
>>> print json.loads(someFile)
{u'glossary': {u'GlossDiv': {u'GlossList': {u'GlossEntry': {u'GlossDef': {u'GlossSeeAlso': [u'GML', u'XML'], u'para': u'A meta-
': u'markup', u'Acronym': u'SGML', u'GlossTerm': u'Standard Generalized Markup Language', u'Abbrev': u'ISO 8879:1986', u'SortAs
```

String

Object

```
>>> someFile = open('/Users/epodojil/GA_Data_Science/a.json').read()
>>> print json.dumps(someFile)
"{\n    \"glossary\": {\n        \"title\": \"example glossary\",\n    \"GlossDiv\": {\n            \"title\": \"S\",\n      \"
\"SGML\",\n          \"SortAs\": \"SGML\",\n          \"GlossTerm\": \"Standard Generalized Markup Language\",\n          \"Acr
ef\": {\n                    \"para\": \"A meta-markup language, used to create markup languages such as DocBook.\",\n
\"GlossSee\": \"markup\"\n                }\n            }\n        }\n    }\n}"
>>> print someFile
{
    "glossary": {
        "title": "example glossary",
    "GlossDiv": {
            "title": "S",
      "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
                    },
          "GlossSee": "markup"
                }
            }
        }
    }
}
>>> print json.loads(someFile)
{u'glossary': {u'GlossDiv': {u'GlossList': {u'GlossEntry': {u'GlossDef': {u'GlossSeeAlso': [u'GML', u'XML'], u'para': u'A meta-
': u'markup', u'Acronym': u'SGML', u'GlossTerm': u'Standard Generalized Markup Language', u'Abbrev': u'ISO 8879:1986', u'SortAs
```

String

Object

Python Dict

APIs (Application Programming Interface) allow people to interact with the structures of an application to get, put, delete, or update data.

APIs (Application Programming Interface) allow people to interact with the structures of an application to get, put, delete, or update data.

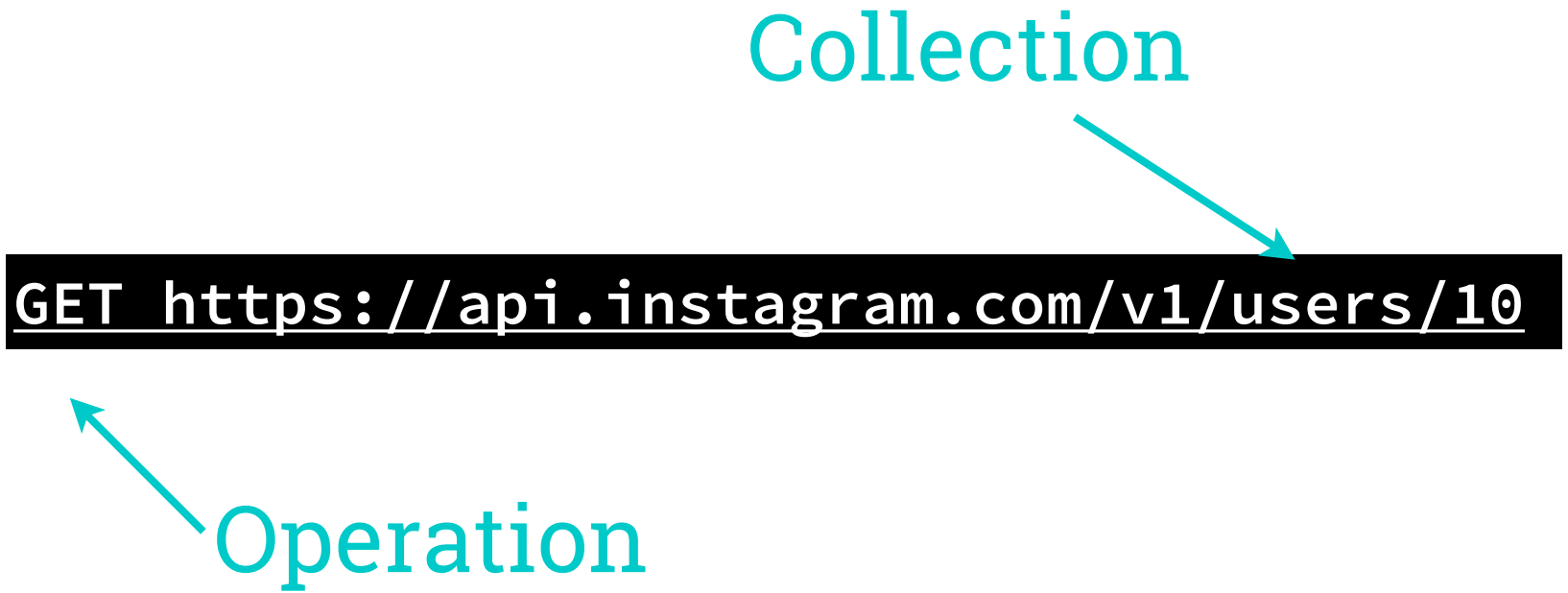
Best practices for APIs are to use RESTful principles.

RESTful APIs include:

The Base URL and collection.
An interactive media type (usually JSON)
Operations (GET, PUT, POST, DELETE)
Driven by Hypertext (http requests)

RESTful APIs include:

The Base URL and collection.
An interactive media type (usually JSON)
Operations (GET, PUT, POST, DELETE)
Driven by Hypertext (http requests)


Collection
GET https://api.instagram.com/v1/users/10
Operation

```
GET https://api.instagram.com/v1/users/
search/?q=andy
```

Querystring

RESTful APIs can always be accessed using cURL requests: hence why hypertext access is a requirement!

Most have language libraries to make it easier to access through the language of your choice.

http://www.pythonapi.com/

DISCUSSION

1. We have processed data now through a variety of different ways (native Python, PANDAS and numpy, MySQL). What seem to be the advantages and tradeoffs of each?

2. This wraps up our “Intro” unit of the course. What remaining questions do you have before we move on?

INTRO TO DATA SCIENCE

HOMEWORK

Journal Review #1 will be due on Tuesday, next week (midnight). Article link is included in the Schoology submission.
See Expectations at the Wiki: "Appendix_01: Reading Journal Articles"

INTRO TO DATA SCIENCE

NEXT CLASS SUBJECT: LINEAR REGRESSION AND GENERALIZATION