STATS 507
Data Analysis in Python

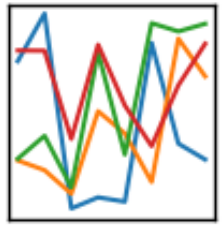
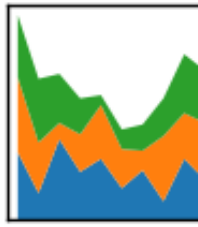
Week10-2: Intro to Database with SQL

Dr. Xian Zhang

Recap: Pandas

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

Yet another **open-sourced**, **practical**, modern data science tool in Python…

- **Database-like structures**, largely similar to those available in R
- Well integrated with `numpy/scipy`
- Supports read/write for a wide range of different file format.
- Flexible and efficient
- Operations: indexing/slicing/functions/group/aggregation/filtering…

https://pandas.pydata.org/docs/user_guide/cookbook.html#cookbook

Intro: why resurgence of DL?

1952 Stochastic gradient descent

1958 Perceptron
- Learnable weights

⋮

1986 Back propagation
- Multi-Layer perceptron

1995 Deep convolutional NN

⋮

ChatGPT… (why now)

1. Big data

- Larger dataset
- Easier collection and storage

IMAGENET

Wikipedia The Free Encyclopedia

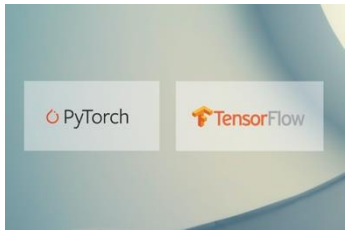
Open Dataset

2. Hardware

- GPUs
- Massively parallelizable

3. Software

- Improved techniques
- New models
- Toolboxes



Slide Credit: Alexander Amini

Modified from MIT open course: 6.S191

Isn’t a spreadsheet enough?

Recall with Pandas, we can already…

```
df = pd.read_csv("forestfires.csv")
df
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00
…	…	…	…	…	…	…	…	…	…	…	…	…	…
512	4	3	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44
513	2	4	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29
514	7	4	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16
515	1	4	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00
516	6	3	nov	tue	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00

517 rows × 13 columns

1. Indexing/slicing
2. Statistical computation
3. Add / delete / sort data
4. Reorganizing / filtering Data

…

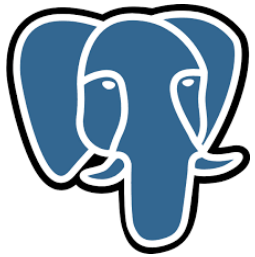
From Spreadsheet to Database

Apple Numbers

Google Sheets

Microsoft Excel

1. Scale
2. Update Frequency
3. Speed



Postgresql

ORACLE DATABASE

…

Database: A collection of data organized for creating, reading, updating and deleting.

What is a (relational) Database?

Relational Database: a collection of tables (tabular data)

id	*name*	*address*	*class*	*dob*
12345678	Jill Jones	Canaday C-54	2011	3/10/85
25252525	Alan Turing	Lowell House F-51	2008	2/7/88
33566891	Audrey Chu	Pfoho, Moors 212	2009	10/2/86
45678900	Jose Delgado	Eliot E-21	2009	7/13/88
66666666	Count Dracula	The Dungeon	2007	11/1431
...	...	...	...	...

Each **row** in a table holds data that describes either:

- An entity
- A relationship between two or more entities

Each **column** in a table represents one **attribute** of an entity

- Each column has a domain of possible values

How can we interact with Database?

Database management systems (DBMS): software via which you can interact with a database.

Public/Open-source options:

MySQL, PostgreSQL, **SQLite**

Proprietary:

IBM Db2, Oracle, SAP, SQL Server (Microsoft)

Using SQL with Python: SQLite

What is SQL?

(originally SEQUEL, from IBM)

Structured Query Language: A language via which you can create, read, update, and delete data in a database.

Structured: it does have **key** words you can use to interact with the database

Query: it is a query language it can be used to ask questions of data inside a database.

Writing Querying / Asking questions

Is our number of daily users growing or shrinking?

Which songs are most like the song the user just player?

What is the most popular carrier at each of the three NYC airports?

SQL `SELECT` Statements

Basic Syntax: `SELECT [column names] FROM [table]`

Table student

id	name	gpa	major	birth_year	pets	favorite_color
101010	John Bardeen	3.1	Electrical Engineering	1908	2	Blue
314159	Albert Einstein	4.0	Physics	1879	0	Green
999999	Jerzy Neyman	3.5	Statistics	1894	1	Red
112358	Ky Fan	3.55	Mathematics	1914	2	Green

```
SELECT id, name, birth_year FROM student
```

id	name	birth_year
101010	John Bardeen	1908
314159	Albert Einstein	1879
999999	Jerzy Neyman	1894
112358	Ky Fan	1914

SQL `WHERE` Statements

To further filter the data: `SELECT [column names] FROM [table] WHERE [filter]`

Table student

id	name	gpa	major	birth_year	pets	favorite_color
101010	John Bardeen	3.1	Electrical Engineering	1908	2	Blue
314159	Albert Einstein	4.0	Physics	1879	0	Green
999999	Jerzy Neyman	3.5	Statistics	1894	1	Red
112358	Ky Fan	3.55	Mathematics	1914	2	Green

```
SELECT id, name FROM student WHERE birth_year >1900
```

id	name
101010	John Bardeen
112358	Ky Fan

Aggregating: `GROUP BY`

Example: I have a DB of transactions at my internet business, and I want to know how much each customer has spent in total.

```
SELECT customer_id,SUM(dollar_amount) FROM transaction GROUP BY customer_id
```

customer_id	customer	order_id	dollar_amount
101	Amy	0023	25
200	Bob	0101	10
315	Cathy	0222	50
200	Bob	0120	12
310	Bob	0429	100
315	Cathy	0111	33
101	Amy	0033	25
315	Cathy	0504	70

customer_id	dollar_amount
101	50
200	22
310	100
315	153

`GROUP BY field_x` combines the rows with the same value in the field `field_x` (Similar to the concept in Pandas)

Other aggregate functions

`GROUP BY` supports other operations in addition to `SUM`:

`COUNT, AVG, MIN, MAX`

The `AS` keyword just lets us give a nicer name to the aggregated field.

```
SELECT customer_id, SUM(dollar_amount) AS total_dollar FROM transaction GROUP
BY customer_id HAVING total_dollar > 50
```

customer_id	dollar_amount
101	50
200	22
310	100
315	40
315	100

customer_id	total_dollar
310	100
315	140

Note: the difference between the `HAVING` keyword and the `WHERE` keyword is that `HAVING` operates *after* applying filters and `GROUP BY`.

Ordering: `ORDER BY` Statements

```
SELECT [columns] FROM [table] ORDER BY [column] [ASC|DESC]
```

Table student

id	name	gpa	major	birth_year	pets	favorite_color
101010	John Bardeen	3.1	Electrical Engineering	1908	2	Blue
314159	Albert Einstein	4.0	Physics	1879	0	Green
999999	Jerzy Neyman	3.5	Statistics	1894	1	Red
112358	Ky Fan	3.55	Mathematics	1914	2	Green

```
SELECT id, name, gpa FROM student ORDER BY gpa DESC
```

id	name	
314159	Albert E	
112358	Ky Fan	
999999	Jerzy Neyman	3.5
101010	John Bardeen	3.1

Note: most implementations order ascending by default, but best to always specify, for your sanity and that of your colleagues!

A quick intro to SQLite

The `sqlite3` module provides an **SQL interface** **in Python** compliant with the database.

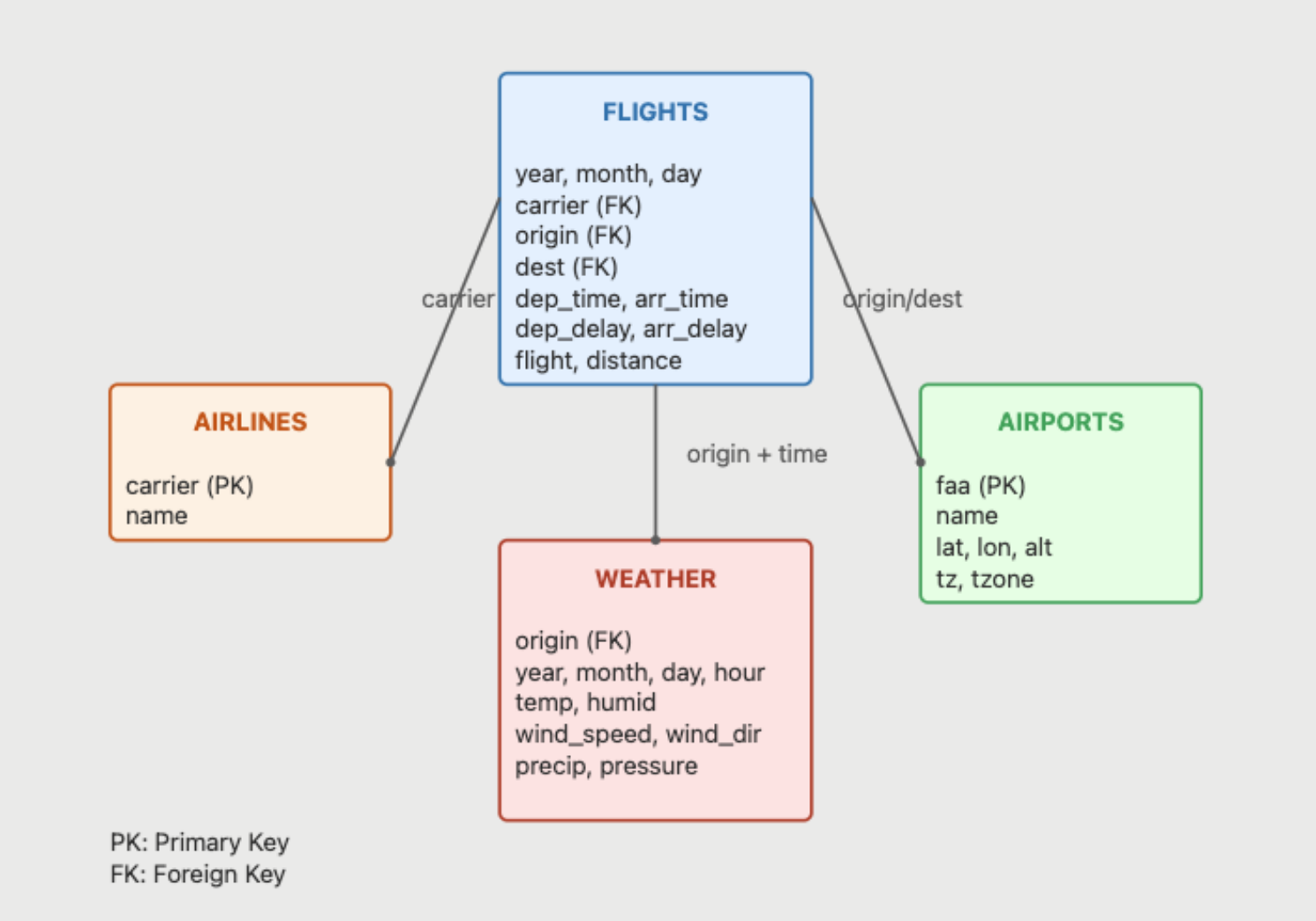
Represents the connection to the on-disk database.

In order to execute SQL statements and fetch results from SQL queries, we will need to use a database cursor.

```
res = cur.execute(QUERY)
```

https://docs.python.org/3/library/sqlite3.html

In class practice



More on `WHERE` Statements

`WHERE` keyword supports all the natural comparisons one would want

(Numerical) Operation	Symbol/keyword
Equal	=
Not equal	<>, !=
Less than	<
Less than or equal to	<=
Greater than	>
Greater than or equal to	>=
Within a range	BETWEEN … AND …

Examples:

```
SELECT id from t_student WHERE …

        … gpa>=3.2

        … pets=1

        … gpa BETWEEN 2.9 AND 3.1

        … birth_year > 1900

        … pets <> 0
```

Caution: different implementations define `BETWEEN` differently (i.e., inclusive vs exclusive)! Be sure to double check!

More filtering: `DISTINCT` Keyword

To remove repeats from a set of returned results:

```
SELECT DISTINCT [columns] FROM [table]
```

Table `student`

id	name	gpa	major	birth_year	pets	favorite_color
101010	John Bardeen	3.1	Electrical Engineering	1908	2	Blue
314159	Albert Einstein	4.0	Physics	1879	0	Green
999999	Jerzy Neyman	3.5	Statistics	1894	1	Red
112358	Ky Fan	3.55	Mathematics	1914	2	Green

```
SELECT DISTINCT pets FROM student ORDER BY pets ASC
```

Test your understanding: what should this return?

Table `student`

id	name	gpa	major	birth_year	pets	favorite_color
101010	John Bardeen	3.1	Electrical Engineering	1908	2	Blue
314159	Albert Einstein	4.0	Physics	1879	0	Green
999999	Jerzy Neyman	3.5	Statistics	1894	1	Red
112358	Ky Fan	3.55	Mathematics	1914	2	Green

```
SELECT DISTINCT pets FROM student ORDER BY pets ASC
```

pets
0
1
2

Relational DBs: pros and cons

Pros:

Natural for the vast majority of applications

Numerous tools for managing and querying

Cons:

Not well-suited to some data (e.g., networks, unstructured text)

Fixed schema (i.e., hard to add columns)

Expensive to maintain when data gets large (e.g., many TBs of data)

Other things

HW7 is out

We will expect you learn LineCharts, BarCharts and CatPlot on your own for HW7. Read the tutorials!

Pick up your midterms

Coming next:

- More on SQL and Regular expressions
- Final project guideline (next Monday)