Week 4 - Declarative SQL Analysis

Where do we get data from??

- provided (csv, excel, xml, json) datasets.
  by organisation / available from online
- web scraping

Relational databases

relation: named, two-dimensional table data

This week's objective

RDBMS + Python X analysis with SQL

CSV → DB via python

then use SQL to analyse that data.

SQL - structured query language.

- DDL, DML

create, drop ~~update~~ alter regarding tables

retrieve information

insert, update, ~~delete~~ delete, select from

Table constraints & relational key

- ~~pk~~ unique, not null
- also check(), default

] can be composite

- FK: refers to candidate key of parent table.

{ DML stuffs

Loading data into SQL

Approach A: \copy

(in psql)

\COPY <table> FROM <filename> CSV [HEADER]

pro: fast, no programming

con: only supports 1:1 mapping (from only one file source)

no data cleaning / transformation ∴ prone to error, crash

Approach B: Python

1. create connection (pgconnect) ← manually-defined function
2. create tables (conn.execute)
3. df = pd.read_csv()
4. df.to_sql on conn

pro: flexible data cleaning / transformation

con: coding required

```
df.to_sql(table_name, conn, if_exists='replace')
```

Data cleaning with Python

- type conversion
  - int(), float()
  - datetime.strptime()
- missing values
  - to ~~None~~ or NaN

For every row in a specified table,
- check row is one of the special values
- check casting runs w/o error
  - → if error, default value.

split: np.nan