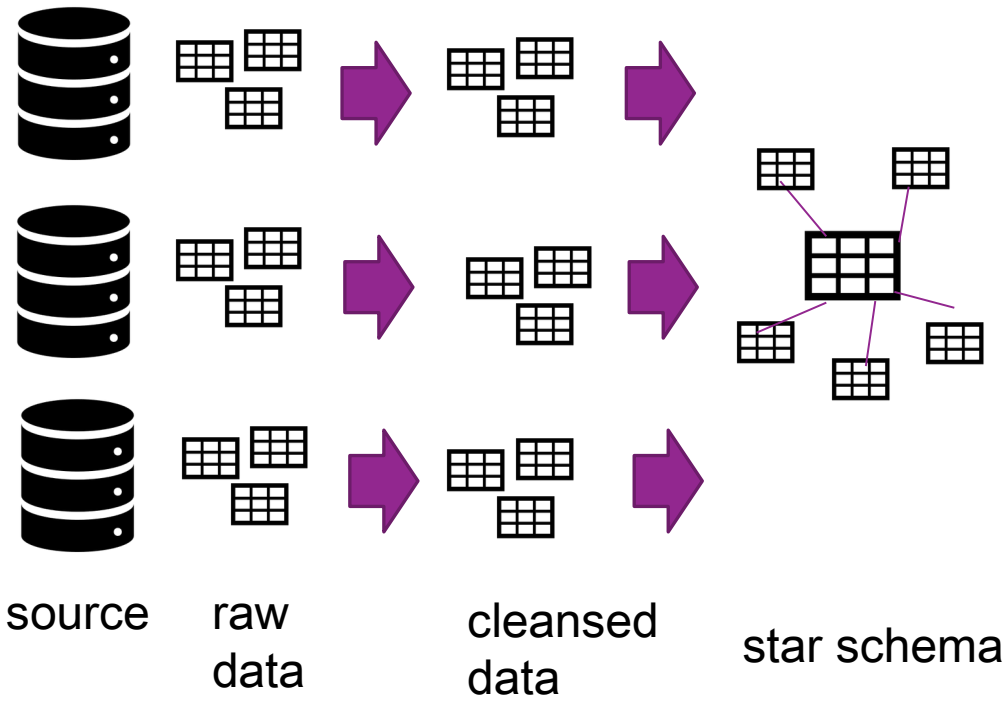
An Introduction to Data Analysis

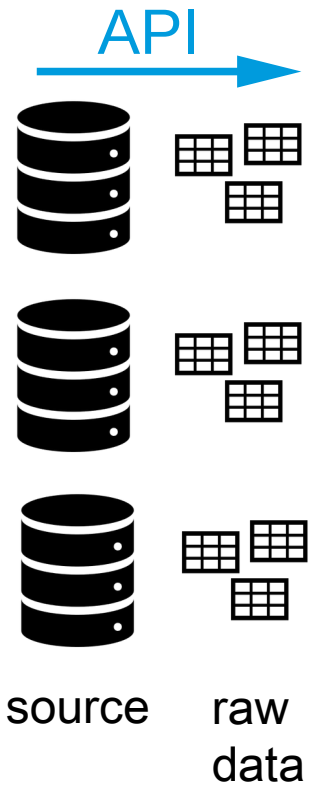
Josh Bodyfelt, Ph.D.

- Nov. 6 – CRUD’ing with SQLite

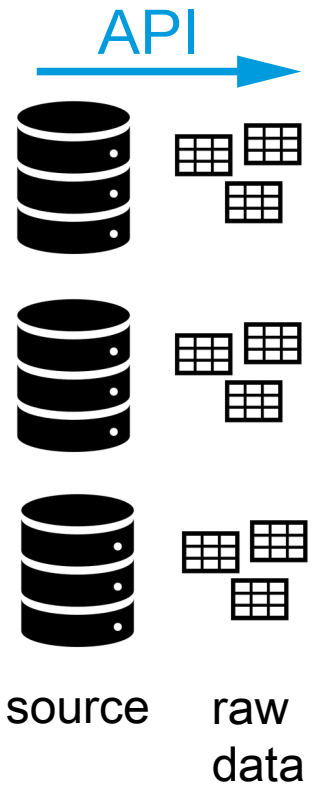
Recall from Last Lecture...



Practice Sources: **Datasets**



Practice Sources: **Governments**

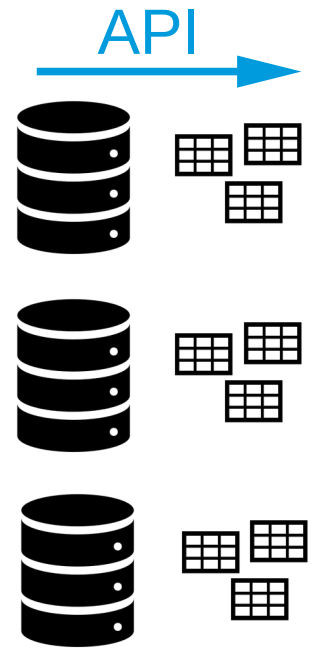


data.govt.nz

data.gov.au

DATA.GOV

Practice Sources: **Public Datasets**



data.govt.nz

data.gov.au

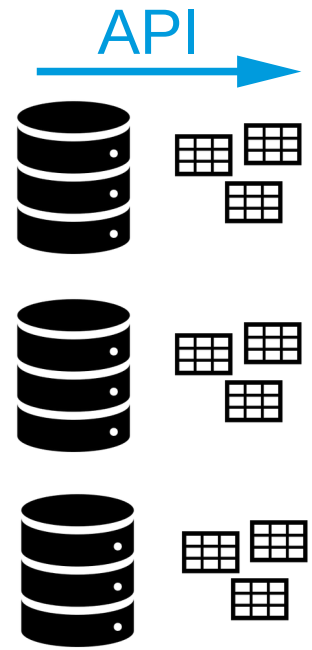
DATA.GOV

Google Cloud Platf

amazon web services™

GitHub

Practice Sources: **Financials**



data.govt.nz

data.gov.au

DATA.GOV

Google Cloud Platf

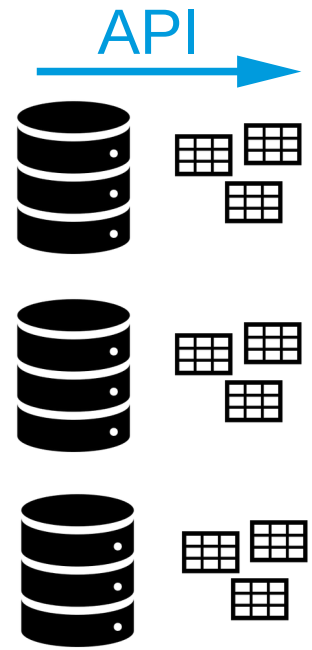
amazon web services™

Quandl

GitHub

Quantopian

Practice Sources: **Machine Learning**



data.govt.nz

data.gov.au

DATA.GOV

Google Cloud Platf

amazon web services™

Quandl

GitHub

Quantopian

UCI Machine Learning Repository

kaggle

Ongoing Example: Movie Dataset

GitHub

Ingesting Data

```
import urllib.request

myURL = "https://raw.githubusercontent.com/rashida048/Datasets/master/movie_dataset.csv"
response = urllib.request.urlopen(myURL)
print(response.read())
```

Ingesting Data

```
import urllib.request

myURL = "https://raw.githubusercontent.com/rashida048/Datasets/master/movie_dataset.csv"
response = urllib.request.urlopen(myURL)
print(response.read())
```

REQUIRED PARSING!

A Secret Weapon to Data Analytics

PANDAS: A Secret Weapon to Data Analytics

```
import pandas as pd

myURL = "https://raw.githubusercontent.com/rashida048/Datasets/master/movie_dataset.csv"
csv = pd.read_csv(myURL)
print(csv.head())
```

```
   index     budget                                  genres  ...                                               cast                                               crew           director
0      0  237000000  Action Adventure Fantasy Science Fiction  ...  Sam Worthington Zoe Saldana Sigourney Weaver S...  [{'name': 'Stephen E. Rivkin', 'gender': 0, 'd...      James Cameron
1      1  300000000                 Adventure Fantasy Action  ...  Johnny Depp Orlando Bloom Keira Knightley Stel...  [{'name': 'Dariusz Wolski', 'gender': 2, 'depa...      Gore Verbinski
2      2  245000000                   Action Adventure Crime  ...  Daniel Craig Christoph Waltz L\u00e9a Seydoux ...  [{'name': 'Thomas Newman', 'gender': 2, 'depar...          Sam Mendes
3      3  250000000               Action Crime Drama Thriller  ...  Christian Bale Michael Caine Gary Oldman Anne ...  [{'name': 'Hans Zimmer', 'gender': 2, 'departm...  Christopher Nolan
4      4  260000000          Action Adventure Science Fiction  ...  Taylor Kitsch Lynn Collins Samantha Morton Wil...  [{'name': 'Andrew Stanton', 'gender': 2, 'depa...      Andrew Stanton

[5 rows x 24 columns]
```

PANDAS: A Secret Weapon to Data Analytics

```
import pandas as pd

myURL = "https://raw.githubusercontent.com/rashida048/Datasets/master/movie_dataset.csv"
csv = pd.read_csv(myURL)
print(csv.head())
```

```
   index     budget                                  genres  ...                                               cast                                               crew           director
0      0  237000000  Action Adventure Fantasy Science Fiction  ...  Sam Worthington Zoe Saldana Sigourney Weaver S...  [{'name': 'Stephen E. Rivkin', 'gender': 0, 'd...      James Cameron
1      1  300000000                 Adventure Fantasy Action  ...  Johnny Depp Orlando Bloom Keira Knightley Stel...  [{'name': 'Dariusz Wolski', 'gender': 2, 'depa...      Gore Verbinski
2      2  245000000                   Action Adventure Crime  ...  Daniel Craig Christoph Waltz L\u00e9a Seydoux ...  [{'name': 'Thomas Newman', 'gender': 2, 'depar...          Sam Mendes
3      3  250000000                Action Crime Drama Thriller  ...  Christian Bale Michael Caine Gary Oldman Anne ...  [{'name': 'Hans Zimmer', 'gender': 2, 'departm...  Christopher Nolan
4      4  260000000           Action Adventure Science Fiction  ...  Taylor Kitsch Lynn Collins Samantha Morton Wil...  [{'name': 'Andrew Stanton', 'gender': 2, 'depa...      Andrew Stanton

[5 rows x 24 columns]
```

```
print(csv.columns.values)
csv.to_csv("movie_dataset.csv")
```

```
['index' 'budget' 'genres' 'homepage' 'id' 'keywords' 'original_language'
 'original_title' 'overview' 'popularity' 'production_companies'
 'production_countries' 'release_date' 'revenue' 'runtime'
 'spoken_languages' 'status' 'tagline' 'title' 'vote_average' 'vote_count'
 'cast' 'crew' 'director'] _
```

Why not just use a flat file?!

movie_dataset.csv → 4803 Titles ~22 MB

IMDb → 6.5 Million Titles ~29 GB

+ 10.4 Million Personalities

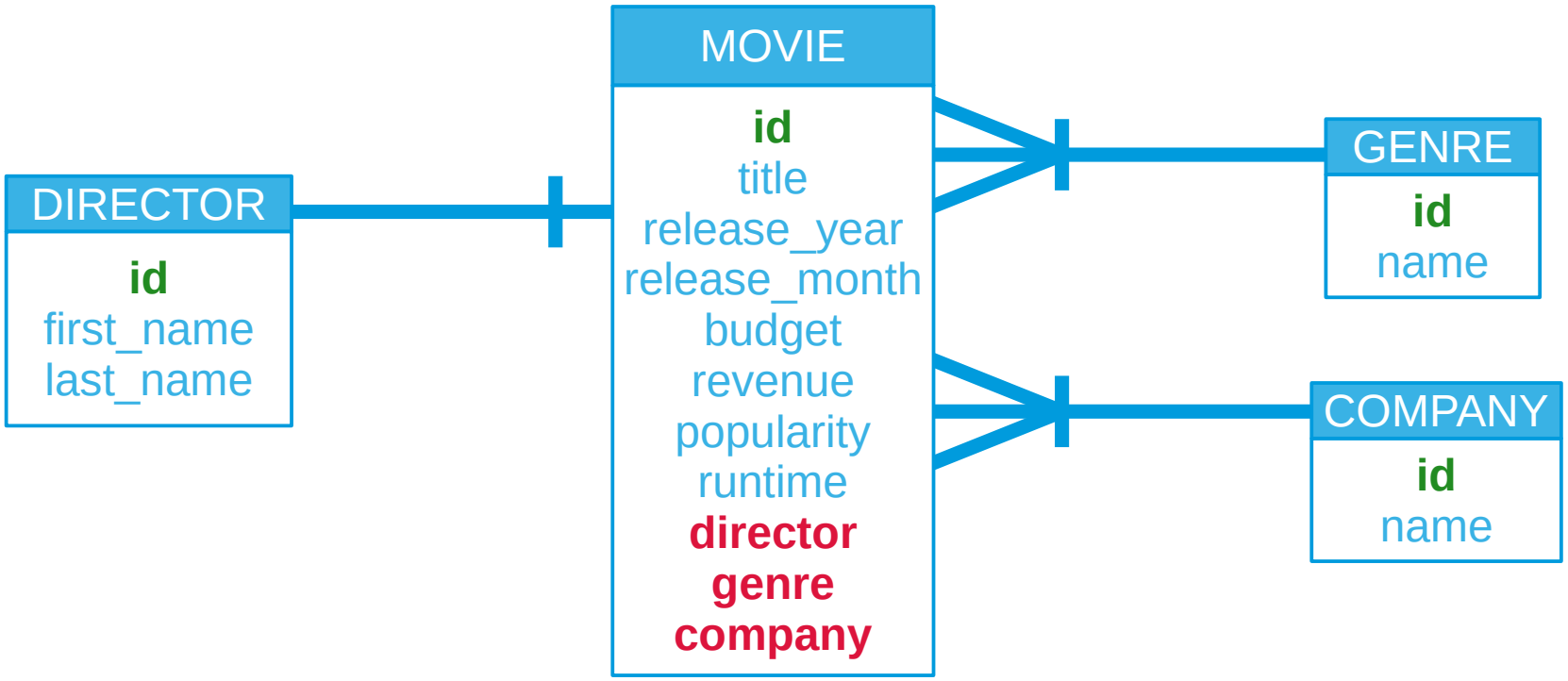
+ 83 Million Users

Create: Our First Database

```
import sqlite3
from sqlite3 import Error
##################################################
def create_database_connection(db_file):
    """ connect to SQLite database in db_file
    :param db_file: database filename
    :return: Connection object or None
    """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)
    return conn
##################################################
conn = create_database_connection("movies.db")
```

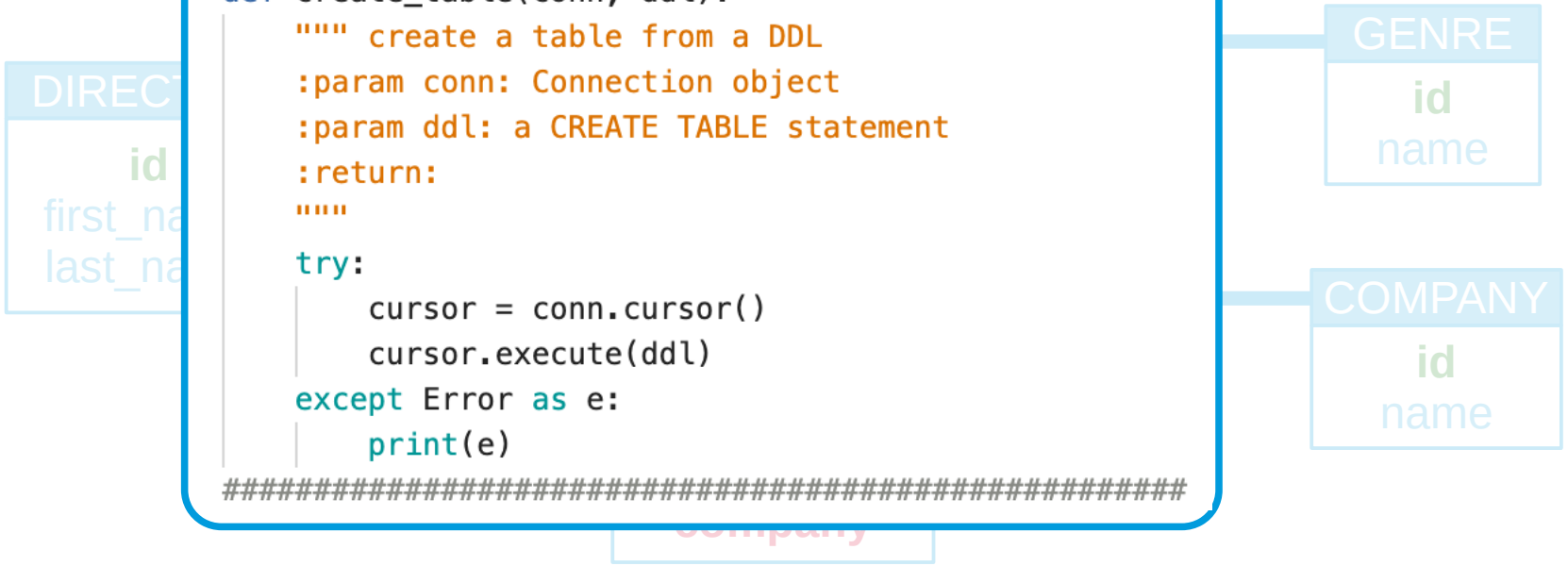
Storing my functions in *movies.py* module...

Entity-Relation Diagram: Our First Database



Create: Table DDLs

```
##########################################################
def create_table(conn, ddl):
    """ create a table from a DDL
    :param conn: Connection object
    :param ddl: a CREATE TABLE statement
    :return:
    """
    try:
        cursor = conn.cursor()
        cursor.execute(ddl)
    except Error as e:
        print(e)
##########################################################
```



Create: Table DDLs


```
# Make DIRECTOR table
create_table(conn, """
    CREATE TABLE IF NOT EXISTS director (
        id integer PRIMARY KEY,
        first_name text,
        last_name text
    );
    """
)
```

Create: Table DDLs

```
# Make GENRE table
create_table(conn, """
    CREATE TABLE IF NOT EXISTS genre (
        id INTEGER PRIMARY KEY,
        name TEXT
    );
    """
)
```

```
# Make COMPANY table
create_table(conn, """
    CREATE TABLE IF NOT EXISTS company (
        id INTEGER PRIMARY KEY,
        name TEXT
    );
    """
)
```

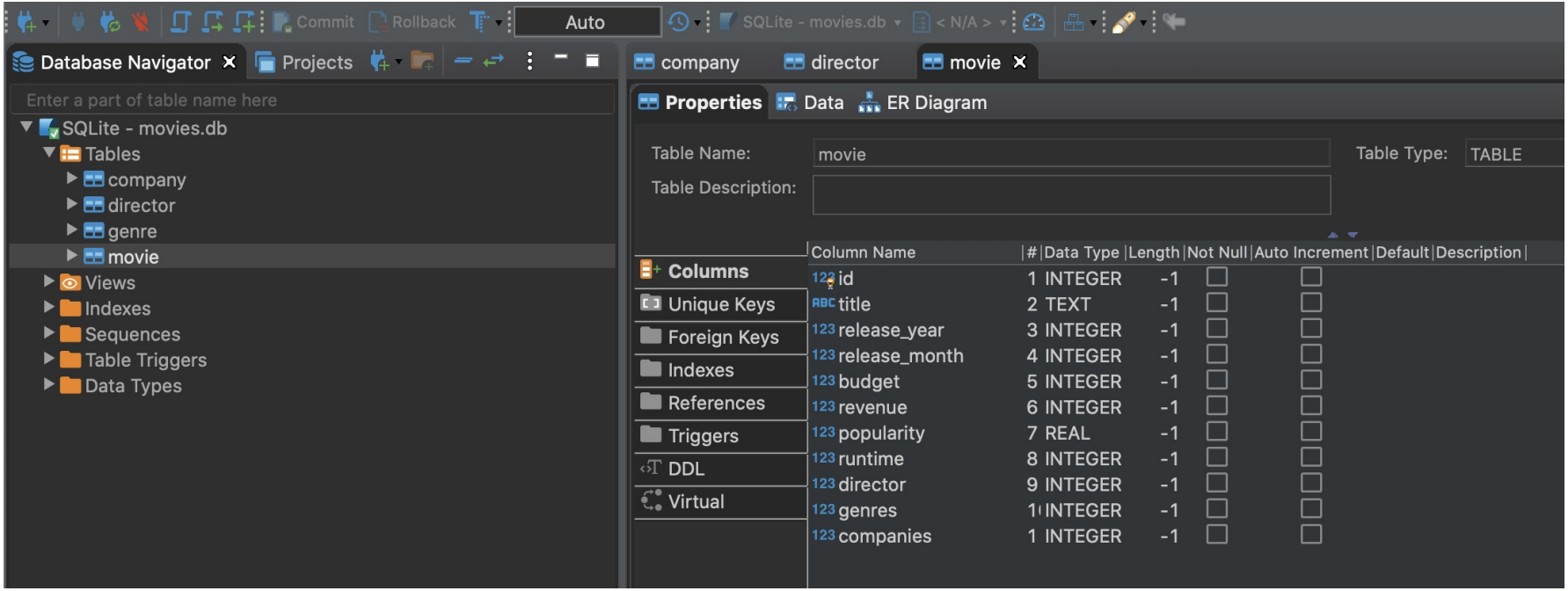
Create: Table DDLs

```
# Make MOVIE table
create_table(conn, """
    CREATE TABLE IF NOT EXISTS movie (
        id INTEGER PRIMARY KEY,
        title TEXT,
        release_year INTEGER,
        release_month INTEGER,
        budget INTEGER,
        revenue INTEGER,
        popularity REAL,
        runtime INTEGER,
        director INTEGER,
        genres INTEGER,
        companies INTEGER,
        FOREIGN KEY(director) REFERENCES director(id),
        FOREIGN KEY(genres) REFERENCES genre(id),
        FOREIGN KEY(companies) REFERENCES company(id)
    );
    """
)
```

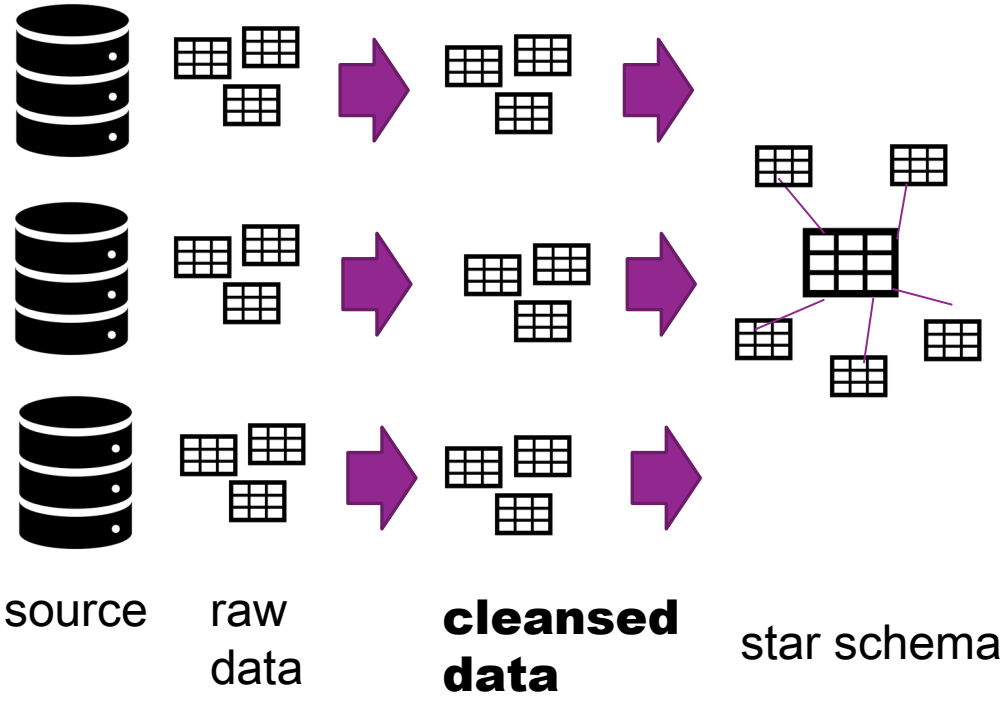
MOVIE
id
title
release_year
release_month
budget
revenue
popularity
runtime
director
genres
companies

A Double Check: Database Explorers

DBeaver



Create (Insert): Record SQLs



4803 Movies

2350 Directors

"Eug\\u00e8ne Louri\\u00e9"

21 Genres

"Western Science Fiction TV Movie"

CLEAN FIRST!

Create (Insert): Record SQLs

```
################################################
def clean_genres(series):
    # Cleaning Genre Series
    cleaned = []
    old_list = ['Science Fiction', 'TV Movie']
    new_list = ['SciFi', 'TV_Movie']
    for genres in series:
        if(type(genres)) == float:
            genres = "Unknown"
        for old, new in zip(old_list, new_list):
            genres = genres.replace(old, new)
        for genre in genres.split(" "):
            cleaned.append(genre)
    return sorted(list(set(cleaned)))
################################################
data = pd.read_csv('movie_dataset.csv')
genres = clean_genres(data["genres"])
```

4803 Movies

2350 Directors

"Eug\\u00e8ne Louri\\u00e9"

21 Genres

"Western Science Fiction TV Movie"

CLEAN FIRST!

Create (Insert): Record SQLs

```
##################################################
def insert_record(table, record):
    """ insert a table record with an SQL
    :param conn: Connection object
    :param table: an existing table name (string)
    :param record: dictionary of record
    :return sql: resulting sql command
    """
    sql = f"INSERT INTO {table}("
    for key in record.keys():
        sql += f"{key}, "
    sql = f"{sql[:-2]}) VALUES ("
    for val in record.values():
        sql += f"{val}, "
    sql = f"{sql[:-2]})"
    return sql
##################################################
```

```
INSERT INTO genre(id, name) VALUES (0, Action)
INSERT INTO genre(id, name) VALUES (1, Adventure)
INSERT INTO genre(id, name) VALUES (2, Animation)
INSERT INTO genre(id, name) VALUES (3, Comedy)
INSERT INTO genre(id, name) VALUES (4, Crime)
INSERT INTO genre(id, name) VALUES (5, Documentary)
INSERT INTO genre(id, name) VALUES (6, Drama)
INSERT INTO genre(id, name) VALUES (7, Family)
INSERT INTO genre(id, name) VALUES (8, Fantasy)
INSERT INTO genre(id, name) VALUES (9, Foreign)
INSERT INTO genre(id, name) VALUES (10, History)
INSERT INTO genre(id, name) VALUES (11, Horror)
INSERT INTO genre(id, name) VALUES (12, Music)
INSERT INTO genre(id, name) VALUES (13, Mystery)
INSERT INTO genre(id, name) VALUES (14, Romance)
INSERT INTO genre(id, name) VALUES (15, SciFi)
INSERT INTO genre(id, name) VALUES (16, TV_Movie)
INSERT INTO genre(id, name) VALUES (17, Thriller)
INSERT INTO genre(id, name) VALUES (18, Unknown)
INSERT INTO genre(id, name) VALUES (19, War)
INSERT INTO genre(id, name) VALUES (20, Western)
```

Aim for REUSABILITY!

Create (Insert): Record SQLs

```
import pandas as pd
from movies import *

data = pd.read_csv('movie_dataset.csv')
genres = clean_genres(data["genres"])

conn = create_database_connection('movies.db')
cursor = conn.cursor()
for k, genre in enumerate(genres):
    sql = insert_record("genre", {'id': k, 'name': f"\'{genre}\'"})
    cursor.execute(sql)
conn.commit()
conn.close()
```

Aim for REUSABILITY!

Read: Getting EXISTING Records

```
0 Action
1 Adventure
2 Animation
3 Comedy
4 Crime
5 Documentary
6 Drama
7 Family
8 Fantasy
9 Foreign
10 History
11 Horror
12 Music
13 Mystery
14 Romance
15 SciFi
16 TV_Movie
17 Thriller
18 Unknown
19 War
20 Western
```

```
genres = cursor.execute('SELECT * FROM genre').fetchall()
for (k, genre) in genres:
    print(k, genre)
```

Read: Getting EXISTING Records

```
0 Action
1 Adventure
2 Animation
3 Comedy
4 Crime
5 Documentary
6 Drama
7 Family
8 Fantasy
9 Foreign
10 History
11 Horror
12 Music
13 Mystery
14 Romance
15 SciFi
16 TV_Movie
17 Thriller
18 Unknown
19 War
20 Western
```

```
genres = cursor.execute('SELECT * FROM genre').fetchall()
for (k, genre) in genres:
    print(k, genre)
```

```
sql = "SELECT * FROM genre WHERE name LIKE 'A%'"
for (k,genre) in cursor.execute(sql).fetchall():
   print(k, genre)
```

```
0 Action
1 Adventure
2 Animation
```

Update: Changing EXISTING Record

```
0 Action
1 Adventure
2 Animation
3 Comedy
4 Crime
5 Documentary
6 Drama
7 Family
8 Fantasy
9 Foreign
10 History
11 Horror
12 Music
13 Mystery
14 Romance
15 SciFi
16 TV_Movie
17 Thriller
18 Unknown
19 War
20 Western
```

```
sql = "UPDATE genre SET name = 'TV Movie' WHERE name = 'TV_Movie'"
cursor.execute(sql)
conn.commit()
# Print to check
for (k, genre) in cursor.execute("SELECT * FROM genre").fetchall():
    print(k, genre)
```

```
0 Action
1 Adventure
2 Animation
3 Comedy
4 Crime
5 Documentary
6 Drama
7 Family
8 Fantasy
9 Foreign
10 History
11 Horror
12 Music
13 Mystery
14 Romance
15 SciFi
16 TV Movie
17 Thriller
18 Unknown
19 War
20 Western
```

Delete: Remove EXISTING Record

```
0 Action
1 Adventure
2 Animation
3 Comedy
4 Crime
5 Documentary
6 Drama
7 Family
8 Fantasy
9 Foreign
10 History
11 Horror
12 Music
13 Mystery
14 Romance
15 SciFi
16 TV_Movie
17 Thriller
18 Unknown
19 War
20 Western
```

```
sql = "DELETE FROM genre WHERE name LIKE 'F%'"
cursor.execute(sql)
conn.commit()
# Print to check
for (k, genre) in cursor.execute("SELECT * FROM genre").fetchall():
    print(k, genre)
```

```
0 Action
1 Adventure
2 Animation
3 Comedy
4 Crime
5 Documentary
6 Drama
10 History
11 Horror
12 Music
13 Mystery
14 Romance
15 SciFi
16 TV Movie
17 Thriller
18 Unknown
19 War
20 Western
```

Exercise Time!

- Clean & Insert (*Inject*) the remaining three tables:

  director – watch your encodings! Try to use *replace* to modify non-English characters...

  company – raw data is a *stringed* dictionary. Parse it to extract names...

  movie – take care in your one-to-many relationships...

- Review lecture links – including below tutorial site!