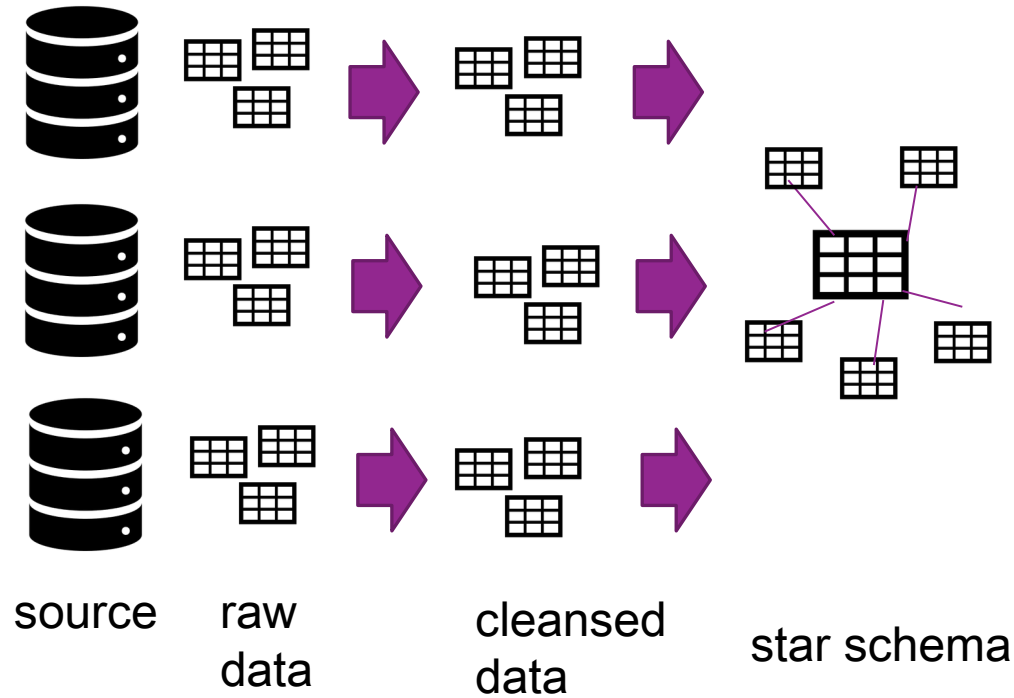


An Introduction to Data Analysis

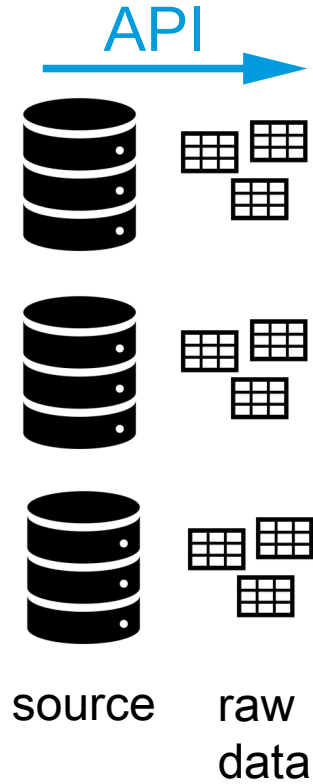
Josh Bodyfelt, Ph.D.

- Nov. 6 – CRUD'ing with SQLite

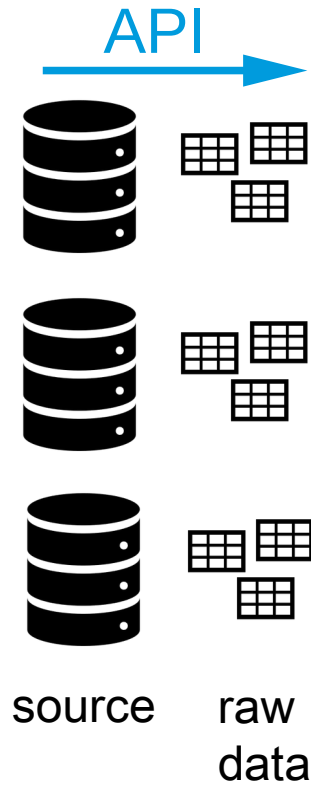
Recall from Last Lecture...



Practice Sources: Datasets



Practice Sources: Governments

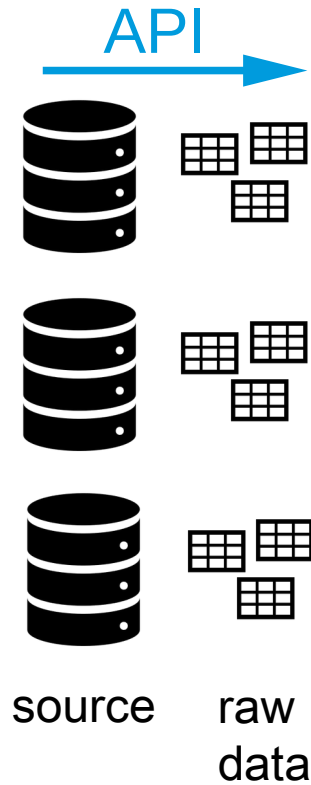


data.govt.nz



DATA.GOV

Practice Sources: Public Datasets



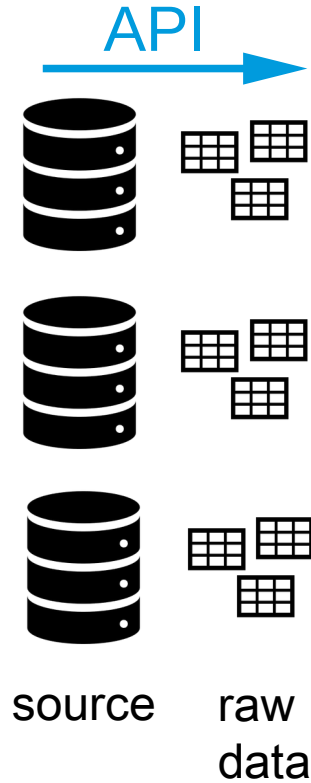
data.govt.nz



Google Cloud Platf



Practice Sources: Financials



data.govt.nz



 **DATA.GOV**



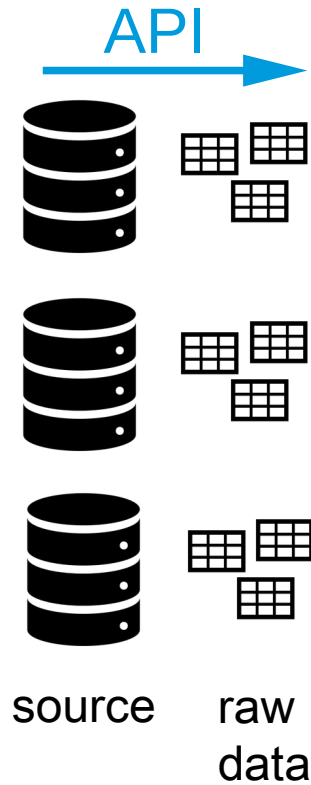
Google Cloud Platf



Quandl



Practice Sources: Machine Learning



data.govt.nz



 **DATA.GOV**



Google Cloud Platf



Quandl



kaggle

Ongoing Example: Movie Dataset



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())
```

```
Martha Williamson\, \gender\': 1, \department\': \Writing\, \job\': \Writer\, \credit_id\': \52fe4df3c3a36847f8275eb1\, \id\': 166873}, {\name\': \Martha Williamson\, \gender\': 1, \depar
tment\': \Production\, \job\': \Executive Producer\, \credit_id\': \58332c6ac3a3686753001dcf\, \id\': 166873}, {\name\': \Joel S. Rice\, \gender\': 2, \department\': \Production\, \job\':
\Executive Producer\, \credit_id\': \554ea9ddc3a36879530001d9\, \id\': 174065}, {\name\': \Jackie Lind\, \gender\': 1, \department\': \Production\, \job\': \Casting\, \credit_id\': \583
3288e92514103450018f0\, \id\': 435006}, {\name\': \Shana Landsburg\, \gender\': 0, \department\': \Production\, \job\': \Casting\, \credit_id\': \5833289ac3a36867500019b1\, \id\': 551521},
{\name\': \Renee Read\, \gender\': 1, \department\': \Art\, \job\': \Production Design\, \credit_id\': \52fe4df3c3a36847f8275ec9\, \id\': 1012891}, {\name\': \Hal Beckett\, \gender\': 2
, \department\': \Sound\, \job\': \Original Music Composer\, \credit_id\': \52fe4df3c3a36847f8275ec3\, \id\': 1099146}, {\name\': \Scott Smith\, \gender\': 0, \department\': \Directing\,
\job\': \Director\, \credit_id\': \52fe4df3c3a36847f8275eab\, \id\': 1219158}, {\name\': \Scott Smith\, \gender\': 0, \department\': \Production\, \job\': \Executive Producer\, \credit_i
d\': \58332c3f9251410348001bf2\, \id\': 1219158}, {\name\': \Lisa Binkley\, \gender\': 0, \department\': \Editing\, \job\': \Editor\, \credit_id\': \52fe4df3c3a36847f8275ebd\, \id\': 1271
268}], Scott Smith\n4801,0,,http://shanghaicalling.com/,126186,,en,Shanghai Calling,"When ambitious New York attorney Sam is sent to Shanghai on assignment, he immediately stumbles into a legal mess that
could end his career. With the help of a beautiful relocation specialist, a well-connected old-timer, a clever journalist, and a street-smart legal assistant, Sam might just save his job, find romance, an
d learn to appreciate the beauty and wonders of Shanghai. Written by Anonymous (IMDB.com).",0.857008,[,{"iso_3166_1": "US", "name": "United States of America"}, {"iso_3166_1": "CN", "name
": "China"}],2012-05-03,0.98,0,[{"iso_639_1": "en", "name": "English"}],Released,A New Yorker in Shanghai,Shanghai Calling,5,7,7,Daniel Henney Eliza Coupe Bill Paxton Alan Ruck Zhu Shimao,"[
{\name\': \Daniel Hsia\, \gender\': 2, \department\': \Directing\, \job\': \Director\, \credit_id\': \52fe4ad9c3a368484e16a36b\, \id\': 208138}, {\name\': \Daniel Hsia\, \gender\': 2, \
department\': \Writing\, \job\': \Writer\, \credit_id\': \52fe4ad9c3a368484e16a371\, \id\': 208138}],Daniel Hsia\n4802,0,Documentary,,25975,obsession camcorder crush dream girl,en,My Date with D
rew,"Ever since the second grade when he first saw her in E.T. The Extraterrestrial, Brian Herzlinger has had a crush on Drew Barrymore. Now, 20 years later he's decided to try to fulfill his lifelong dr
eam by asking her for a date. There's one small problem: She's Drew Barrymore and he's, well, Brian Herzlinger, a broke 27-year-old aspiring filmmaker from New Jersey.",1.9298830000000002,[{"name":
"rusty bear entertainment", "id": "87986", "name": "lucky crow films", "id": "87987"}], [{"iso_3166_1": "US", "name": "United States of America"}, {"iso_639_1": "en", "name": "English"}],Released,My Date with Drew,6,3,16,Drew Barrymore Brian Herzlinger Corey Feldman Eric Roberts Griffin Dunne,[{\name\': \Clark Peterson\, \gender\': 2, \department\': \
Production\, \job\': \Executive Producer\, \credit_id\': \58ce021b9251415a390165d9\, \id\': 6888}, {\name\': \Andrew Reimer\, \gender\': 2, \department\': \Production\, \job\': \Executive
Producer\, \credit_id\': \58ce0232c3a36850e90157da\, \id\': 61051}, {\name\': \Brian Herzlinger\, \gender\': 2, \department\': \Directing\, \job\': \Director\, \credit_id\': \52fe44e8c3a368484e03da8d\, \id\': 85563}, {\name\': \Jon Gunn\, \gender\': 2, \department\': \Directing\, \job\': \Director\, \credit_id\': \52fe44e8c3a368484e03da87\, \id\': 94471}, {\name\': \Bre
tt Winn\, \gender\': 0, \department\': \Directing\, \job\': \Director\, \credit_id\': \52fe44e8c3a368484e03da97\, \id\': 997560}],Brian Herzlinger\n
```

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



→ 6.5 Million Titles ~29 GB
+ 10.4 Million Personalities
+ 83 Million Users

Create: Our First Database

```
import sqlite3
from sqlite3 import Error

#####

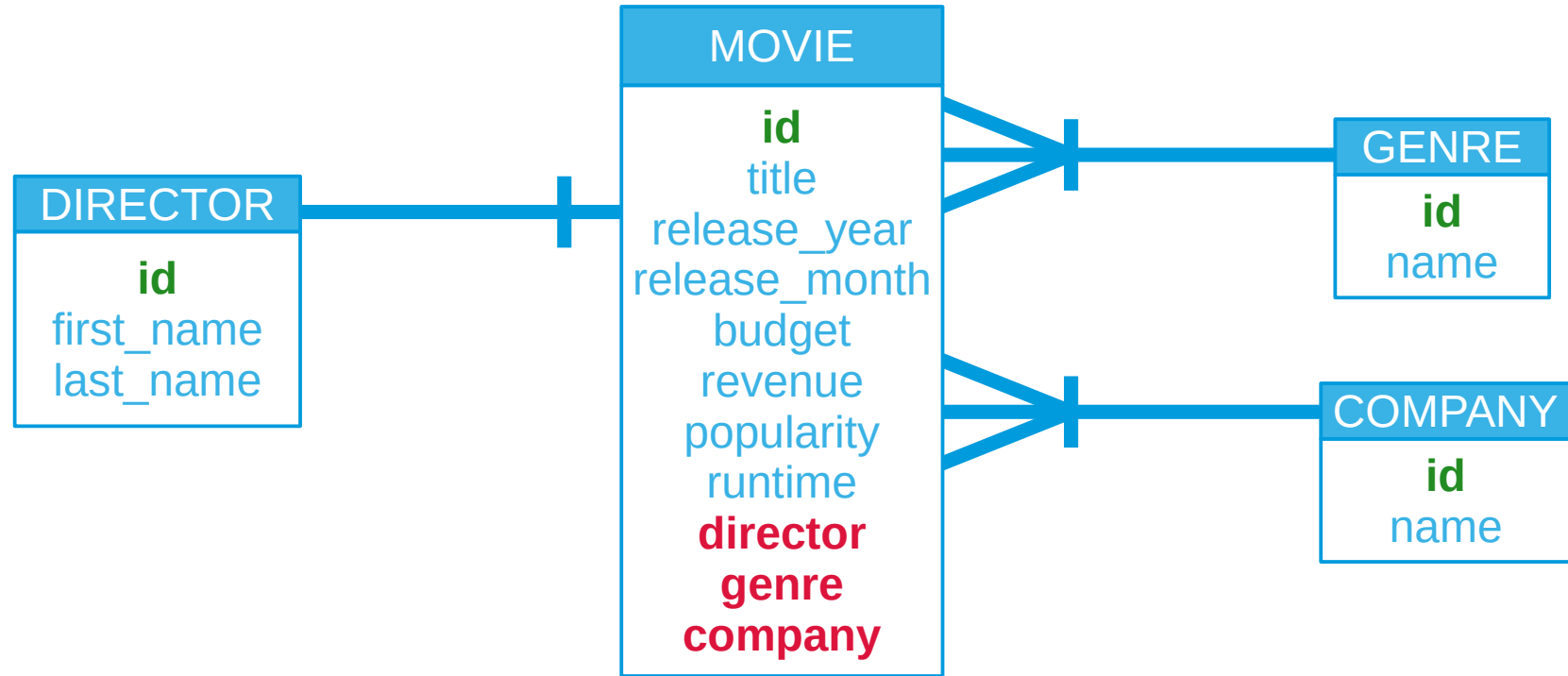
def create_database_connection(db_file):
    """ connect to SQLite database in db_file
    :param db_file: database filename
    :return: Connection object or None
    """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)
   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

DIRECT

id

first_name

last_name

GENRE

id

name

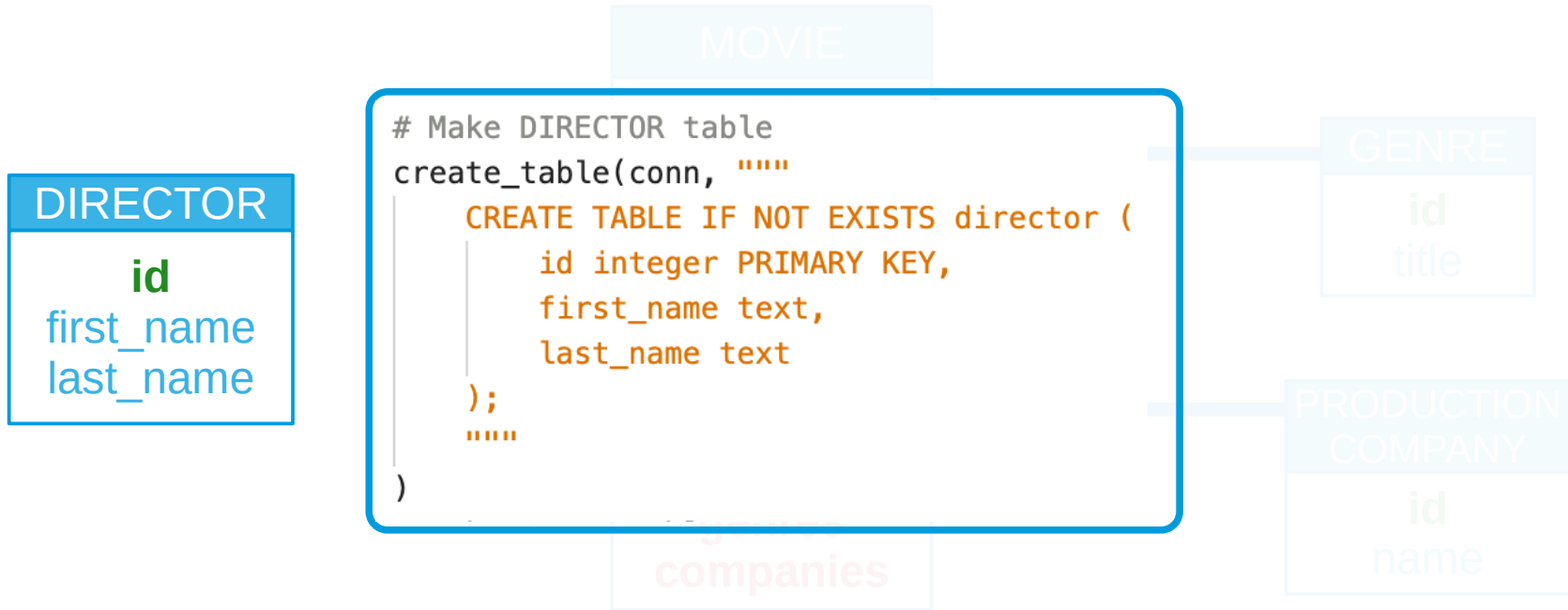
COMPANY

id

name

company

Create: Table DDLs



Create: Table DDLs

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

GENRE

id
name

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

COMPANY

id
name

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



Database Navigator x Projects SQLite - movies.db < N/A > Auto

company director movie x

Enter a part of table name here

- SQLite - movies.db
 - Tables
 - company
 - director
 - genre
 - movie
 - Views
 - Indexes
 - Sequences
 - Table Triggers
 - Data Types

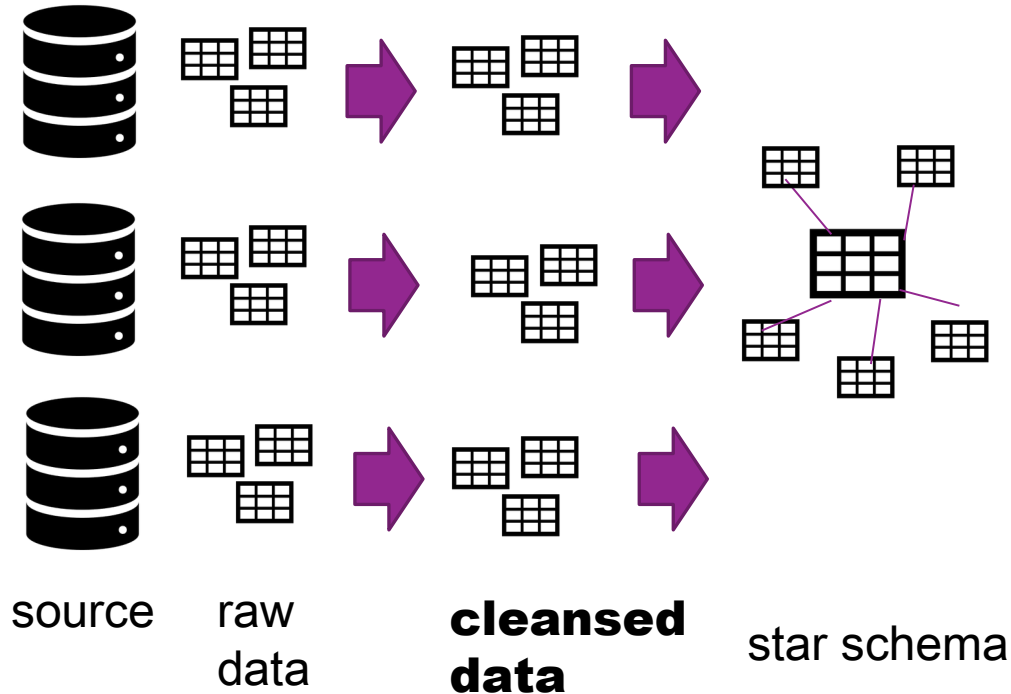
Properties Data ER Diagram

Table Name: movie Table Type: TABLE

Table Description:

	Column Name	#	Data Type	Length	Not Null	Auto Increment	Default	Description
Columns	id	1	INTEGER	-1	<input type="checkbox"/>	<input type="checkbox"/>		
Unique Keys	title	2	TEXT	-1	<input type="checkbox"/>	<input type="checkbox"/>		
Foreign Keys	release_year	3	INTEGER	-1	<input type="checkbox"/>	<input type="checkbox"/>		
Indexes	release_month	4	INTEGER	-1	<input type="checkbox"/>	<input type="checkbox"/>		
References	budget	5	INTEGER	-1	<input type="checkbox"/>	<input type="checkbox"/>		
Triggers	revenue	6	INTEGER	-1	<input type="checkbox"/>	<input type="checkbox"/>		
DDL	popularity	7	REAL	-1	<input type="checkbox"/>	<input type="checkbox"/>		
Virtual	runtime	8	INTEGER	-1	<input type="checkbox"/>	<input type="checkbox"/>		
	director	9	INTEGER	-1	<input type="checkbox"/>	<input type="checkbox"/>		
	genres	10	INTEGER	-1	<input type="checkbox"/>	<input type="checkbox"/>		
	companies	11	INTEGER	-1	<input type="checkbox"/>	<input type="checkbox"/>		

Create (Insert): Record SQLs



4803 Movies

2350 Directors

"Eugene Lourie"

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

"Eugene Lourie"

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!

