NYU

# Introduction to Basic SQL

Section Leader: Will Calandra

09.17.24

PART 01

# Data lives in a database

# Assumed “status quo”

- Let’s just use .csv files for everything, store on our file system
  - Pros: easy to use, data will stay the same, tree structure
  - Cons: no forced data types, difficult to read/write concurrently, typically one 2D table at a time
  - Any other problems?
- Doesn’t scale. What if we have big data? Complex relationships? Work together?
- We need **databases** -> *relational databases* (RDBMS)

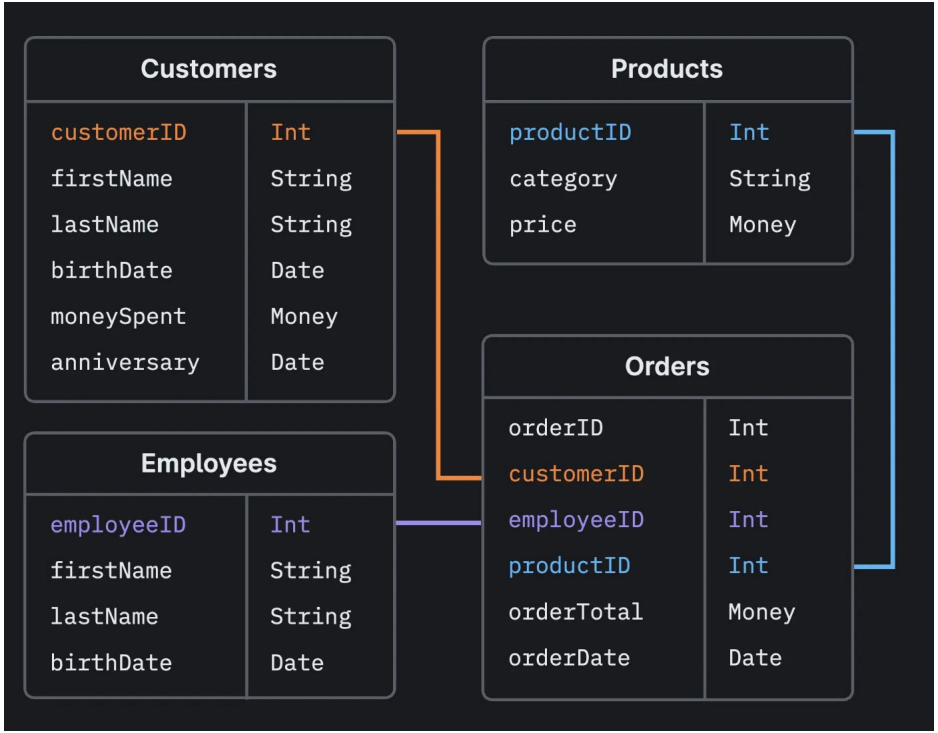
# The industry way

- “Data lives in a database”
- Databases ensure a few things, if done well…
  - **Atomicity**: operations are all-or-nothing (something breaks, it won’t update!)
  - **Consistency**: db is always in a valid state (no corruption)
  - **Isolation**: concurrent operations do not depend on order of execution (locking, independence)
  - **Durability**: completed transactions are permanent (version controls, recover valid states)
- We’ll work with *relational databases*
  - **Tables**: 2D, rows and columns, one entry in each cell
  - **Rows**: represent a unique record/tuple from a relation
  - **Columns**: represent a field of the record/attribute from a relation

# Constraints

- Big idea: induce some constraints on our data to make our programming lives easier
  - Don’t have to explicitly program/check things that db will automate
- We constrain our data through a **schema**
  - Gives us valid rows of data that must fit the schema, or else reject. Helps with data typing and expected inputs
  - Excel fans: “1”, random date casting
- We search data using **keys**
  - **Primary key**: unique id for each row in a table
  - **Foreign key**: column that links to a primary key in another table

# Schema


Customers
customerID Int
firstName String
lastName String
birthDate Date
moneySpent Money
anniversary Date
Products
productID Int
category String
price Money
Orders
orderID Int
customerID Int
employeeID Int
productID Int
orderTotal Money
orderDate Date
Employees
employeeID Int
firstName String
lastName String
birthDate Date

# Keys

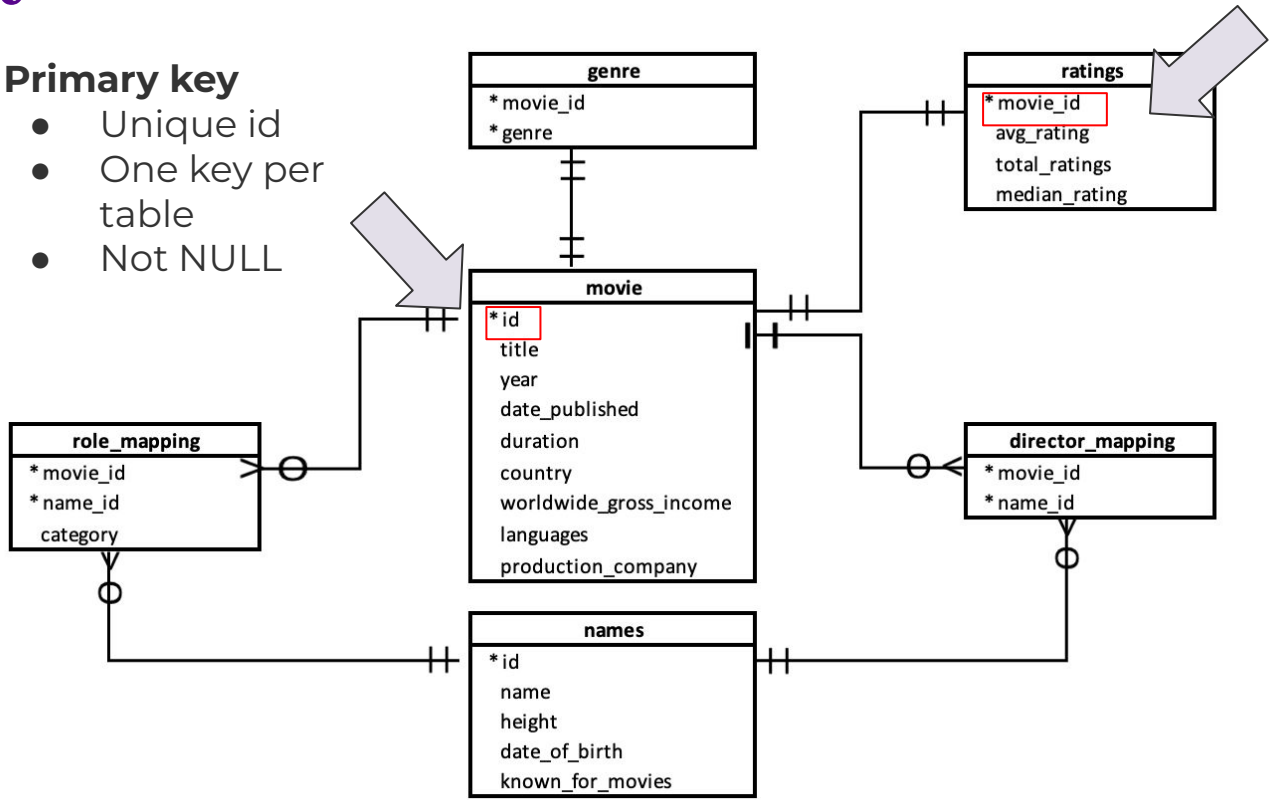
Data source for today

**Primary key**

- Unique id
- One key per table
- Not NULL

**Foreign key**

- Not unique
- Can have multiple in a table
- Can be NULL



PART 01

# What's the big deal with SQL?

# SQL (Structured Query Language)

- **SQL** is how we are able to access and manipulate relational databases
  - **Declarative**: we tell it what to do, not how to do it (not instruction-based like Python or C++)
- We'll see it in a Pythonic way, and we'll use **SQLite**
  - Integrate it into a pipeline, SQLite comes with Python
  - Serverless, don't have to configure/maintain it, but stored locally
  - US Library of Congress trusts it for their digitization. Would you?
- There exist many flavors of SQL (MySQL, PostgreSQL, etc.), but the syntax is mostly the same
- **Industry standard**: not going anywhere!

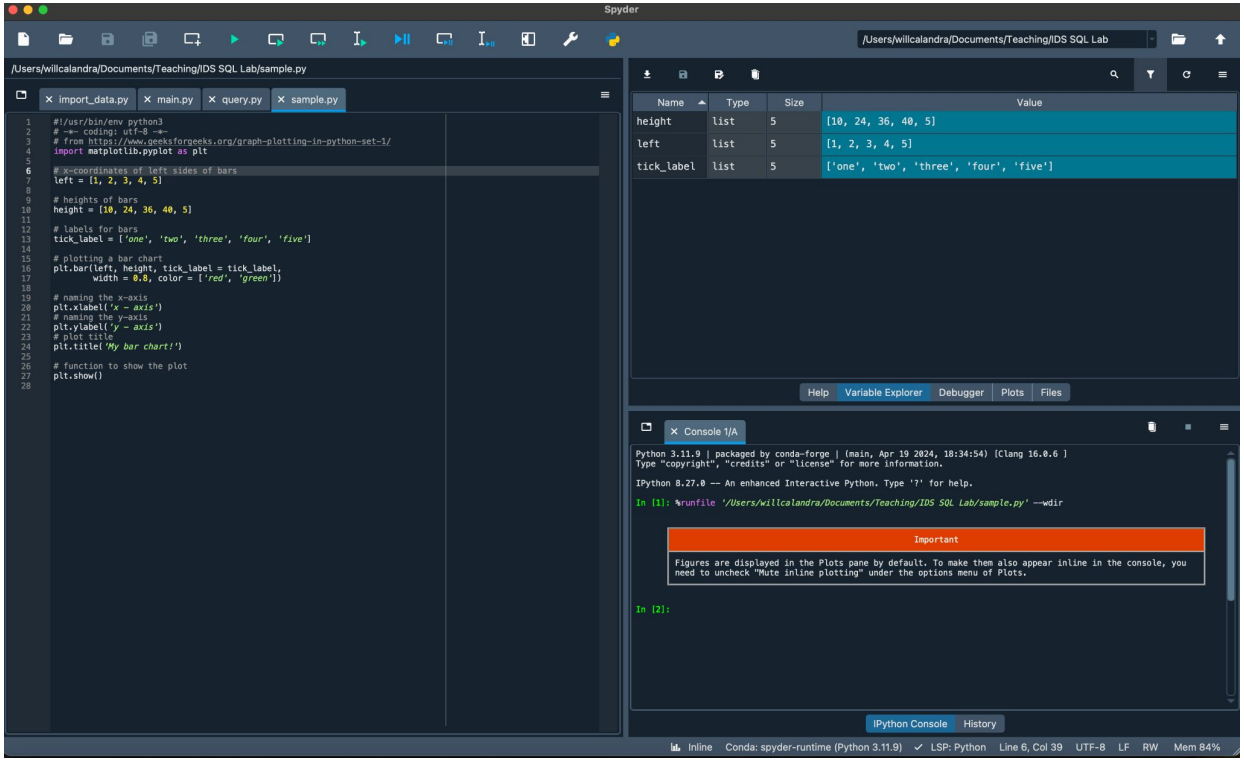
PART 02

# Our host: Spyder IDE

# Spyder IDE

- Installation (try with Anaconda)
- Gemini: 2.7-6.1% of population has arachnophobia. Perhaps you can improve marketing
- **IDE** (Integrated Development Environment) made for data science. “Written in Python, for Python, designed by and for scientists, engineers, and data analysts”
    - A way for developers to code in one GUI (interface)
- Choice of IDE is personal preference
    - Explore extensions (highlighting, dare I say AI)
    - This won’t solve your problems: Microsoft Word anyone?
- **Spyder**: balance of software/scientific scripting
    - Try to stray away from Jupyter notebooks

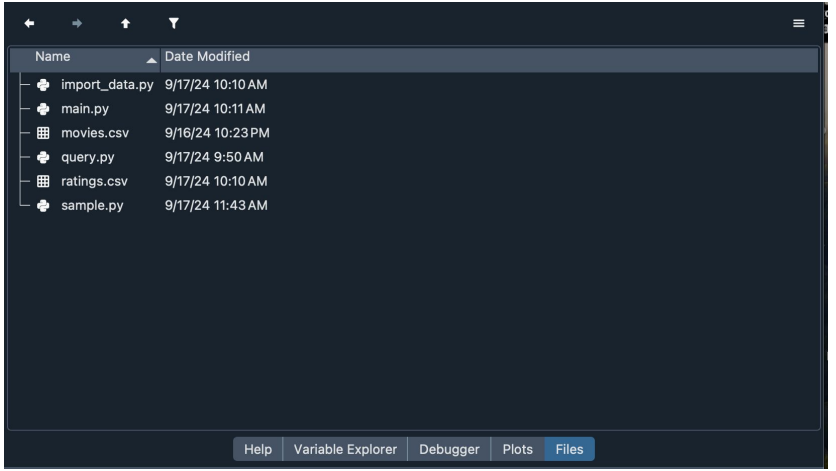
# Spyder IDE




NYU

# Spyder IDE


64 %
My bar chart!
y - axis
x - axis
one
two
three
four
five
Help
Variable Explorer
Debugger
Plots
Files


Name
Date Modified
import_data.py 9/17/24 10:10 AM
main.py 9/17/24 10:11 AM
movies.csv 9/16/24 10:23 PM
query.py 9/17/24 9:50 AM
ratings.csv 9/17/24 10:10 AM
sample.py 9/17/24 11:43 AM
Help
Variable Explorer
Debugger
Plots
Files

PART 03

# SQL demo

# SQL order of operations

## Writing

1. SELECT
2. FROM
3. JOIN, ON
4. WHERE
5. GROUP BY
6. HAVING
7. ORDER BY
8. LIMIT

## Execution

1. FROM
2. JOIN, ON
3. WHERE
4. GROUP BY
5. HAVING
6. SELECT
7. ORDER BY
8. LIMIT

# SQL resources

- SQLite Docs
- W3schools
- DataLemur
- LeetCode Interview Qs
- O'Reilly SQL Books (click O'Reilly link w/ NYU creds)
- Modern Application: Pinterest Text-to-SQL
- CDS students: SQL for Interviewing Workshop next week