SQL

* SQL (Structured Query Language)
* The most popular programming language for dealing with databases.
* Been around for 50 years
  + Very well established
  + A bunch of different dialects of SQL
    - Slightly different versions of the Syntax
      * OracleSQL
      * MySQL
      * MariaDB
      * Postgres
      * Microsoft SQL server
* SQL is a scripting language
  + A scripting language is a programming language that tells a machine what to do.
  + A script is a series of actions you want the machine to take.
    - These actions can lead to permanent changes on the machine.

Database

* Database is a way to persist information.
  + Data persistence means to physically save the data in the world.
  + The data is no longer ephemeral.
  + You can turn off the power to a database and your data still exists.
* Types of Databases
  + NoSQL
    - Usually use documents and JSON trees to store information
  + Graph based databases
    - They use complicated connections between nodes to store information
  + Relational Databases
    - The most popular type of database
    - All relational databases use SQL

Relational Database

* Tables are used to store information.
  + Columns/attributes
  + Rows/records

|  |  |  |
| --- | --- | --- |
| Employee\_id | First\_name | Last\_name |
| 101 | Adam | Ranieri |
| 102 | Steven | Smith |

* Tables reference each other to better store information and avoid being redundant.
  + Done using foreign keys.

**Normalization**

* Process by which we eliminate redundancy.
* There are 7 normalized forms.
* 1nf
  + All records in a table must be uniquely identifiable.
    - Primary key
  + All columns must be atomic.
    - The data in it could not be broken down into something more meaningful.
    - Name and address are not atomic
      * Name could be broken down into first and last
      * Address could be broken down into local address, state, zip code

|  |  |  |
| --- | --- | --- |
| Employee\_id | name | address |
| 101 | Adam Ranieri | 111 morgantown wv |
| 202 | Steven Kelsey | 333 Tampa Fl |
|  |  |  |

* 2nf
  + You are already in 1nf.
  + You have no functional dependencies.
    - You cannot calculate one column using the values of other columns
    - Shooting\_percentange is a functional dependency
    - Could be calculated by using the other columns

|  |  |  |  |
| --- | --- | --- | --- |
| Player\_id | Shots\_taken | Shots\_made | Shooting\_percentage |
| 101 | 100 | 51 | 51 |
| 202 | 150 | 100 | 66.666 |
|  |  |  |  |

* 3nf
  + In 2nf
  + No transitive dependencies
    - Counter example
    - If the player table had a field saying who their head coach was
      * Transitive dependency because you could find a player’s team
      * Look at that team’s coaches and see who the head coach was
    - If you can find the data anywhere else in the database you have a transitive dependency
* Normalization does not mean better.
  + Highly normalized is not always preferrable.
  + Highly normalized databases can be hard to query.
    - Difficult to write complex join for
  + OLTP (transaction databases)
    - Highly normalized
    - Lot of updates and insertions
  + OLAP (analytical databases)
    - Denormalized
    - Very read heavy.
* A join is a denormalization of your data.
  + Left join
    - All records on the left table plus matching records
  + Right join
    - All records on the right table plus matching records
  + Inner join
    - Only matching records
  + Cross join
    - Every record matched with every record