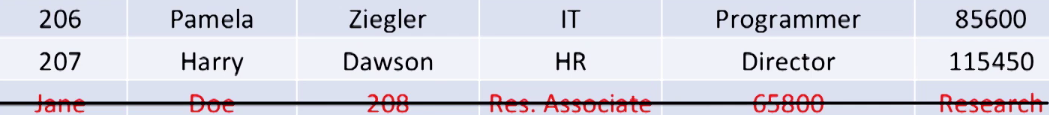
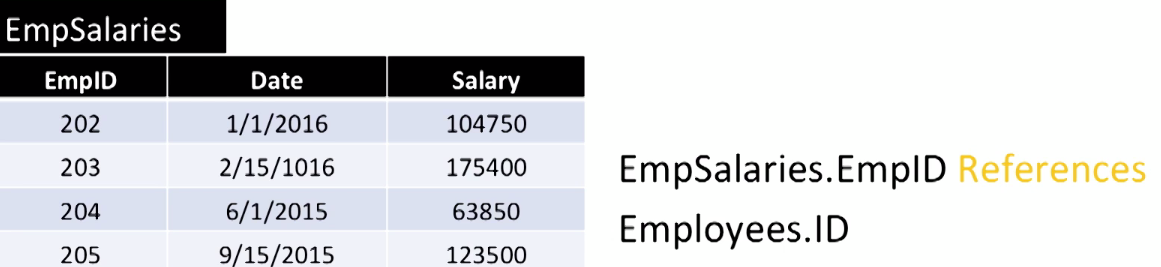
**I. RELATIONAL DATA MODEL**

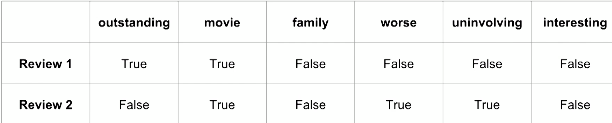
* Primary data structure for a **relational model** is a **table**, which is very similar in structure to a Python DataFrame (which can be used to load **relations** (relational tables) into Python)
* provides a way to describe unique relationships between entities and data records
* Pandas DataFrames implement some features from relational models to make it easier to work w/ RDBs
* Tables represent a set of **relational tuples** as *rows* (records)
* A table of 6 rows is a **set** (collection of *distinct* elements of the same type) of 6 tuples
* Many relational systems allows duplicate tuples, but mechanisms are provided to prevent these is a user chooses so
* Also cannot add a tuple w/ info in the wrong order
* 
* 
* The header (1st row) is part of the **schema** of a table = columns in a Pandas DataFrame
* Schemas can also specify keys (ID as **primary key =** uniquely ID a record/tuple and uniquely know the other 5 attributes for that employee)
* Tables w/ primary keys prevent duplicate records via a **uniqueness constraint**
* **Foreign Keys** refer to primary keys



* **“References” =** values in the EmpSalaries.EmpID column can only exist if the same values appear in Employees.ID (exists in *table being referenced* = **parent table**)
* **Natural Join** = Inner Join
* Joins are one of the most expensive (in terms of time and space) consuming operations, especially w/ very large tables
* For Data Science involving big data, it’s very important to choose a suitable RDBMS to make this operation efficient
* **Data retrieval** = the way you specify how to get desired data out of a relational data store, + the internal processing that occurs w/in the data management system to compute/evaluate that specified retrieval request.
* Ex: Database Schemas:
* Bars(name, addr, license) Beers(name, manf) Sells(bar, beer, price)
* Sells records which bar sells which beer at what price (differences in establishment costs)

***Natural Language Processing (NLP)***

* **NLP** = interaction of CPU’s + the natural language that humans use
* Very difficult task b/c human language is ambiguous
* Context, slang, metaphors, etc. are easily understood by humans, not so much by CPU’s
* Understanding context of speech and linking it to understanding on the concepts around it is difficult to do algorithmically
* Field of NLP is trying to improve
* Applications: summarize text, generate keywords, ID sentiment of text
* Examples: Virtual assistants (), online summaries of news, most popular Twitter topics via keywords
* Speech recognition engines (Siri, Google Now, Alexa) are designed to learn what + how a human talks over time + works to improve accuracy of their understanding
* Translators (Google or Facebook) use NLP w/ newer, effective neural network techniques that take not only words + phrases into account, but also context by looking at words surrounding the text they’re translating
* Chatbots use NLP engines to process + categorize questions + match them to existing answers to questions
* **nltk** = Python NLP toolkit
* works w/ human language data w/ 50+ datasets
* library w/ easy-to-use algorithms for importing, cleaning, pre-processing text data, as well as applying computational linguistic algorithms (ex: sentiment analysis) to it
* nltk depends on large amounts of text/other linguistic data
* **Corpora =** digital collections of large amounts of text/linguistic data
* **Tokenizing =** splitting text into words (1st step in NLP)
* Very tedious to handle **corner cases** = inconsistent punctuation use or contractions, shortened words (abbreviations), hyphenated words, etc.
* Simplest model for analyzing text 🡪 Just think about text as an *unordered collection of words* = **bag-of-words**
* Bag-of-words is a body of text represented as a loose set of words that flattens any text into an unordered collection
* This generally allows us to infer the category, topic, or sentiment from the text
* Pretty useful technique to ID topic or sentiment in text even though we’ve disregarded sentence structure associated to the words
* Can use words in a feature matrix where each words = column + each text body (ex: movie review) is a rows, and we have Boolean values if a word appears in the text body



* Just by looking at the reviews, we can ID that the topic of the reviews are “movies” (col2) + review 1 is positive while review 2 is negative
* It’s common to filter **stopwords** (ex: *the, that, is, + other words* that appear a lot but don’t have significance in IDing context) and maybe even punctuation from the bag-of-words before analysis
* **Sentiment Analysis** = IDing the attitude or emotion encoded in a body of text (product review, literature, etc.)
* **Naïve Bayes Classifier** = simple classifier based on **conditional probabilities**
* Training = detects probability that each feature (word) appears in a category (pos or neg)
* Once trained, it collects “votes” for all words in a new text body + finds the most probable label
* i.e. the relationships between input features + class labels are expressed as a probability
* Given input features, probability for each class is estimated and the class w/ the highest probability determine the label for the sample