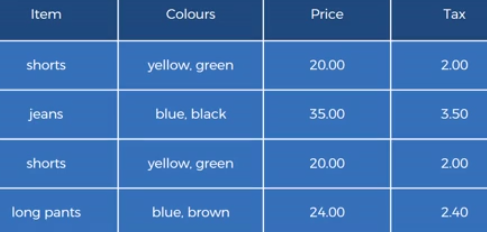
***Database Design***

*Normalization*

* **Normalization –** process of organizing columns/attributes + tables/relations or an RDB to reduce data redundancy and improve data integrity
* Means to better structure DB’s to remove the potential for any UPDATE/INSERT/DELETE anomalies + speed up the process of how the DB is utilized



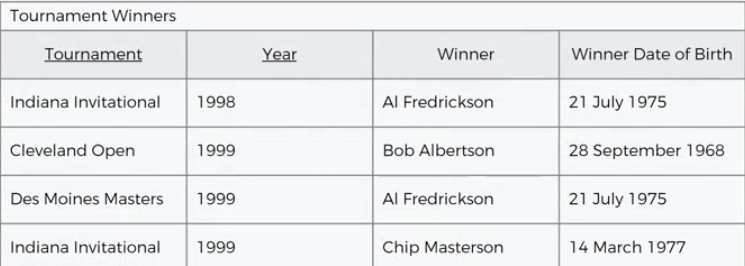
* See 2 values in each cell for Colours, + we just want 1 value for an attribute
* Could see 2 identical rows (shorts) 🡪 2nd row adds no value 🡪 don’t want
* Let’s say color doesn’t impact price in any way + we want to manually add a new row w/ purple jeans
* Should be same price + tax as 1st row w/ jeans, b/c the new color doesn’t affect them
* Now we’re duplicating price + tax info for jeans 🡪 doing unnecessary work to get info that’s already present + wasting time + resources to write it again
* Also raises potential for error if we manually input the wrong price + tax for jeans
* DB Design is important for DS b/c by learning it, we will be able to:
* *Understand HOW + WHY* a DB we’re working w/ was designed in a certain way
* Understand how to navigate around the DB structure quicker
* See + understand potential vulnerabilities + flaws in the DB + data
* Communicate effectively w/ data custodians/stewards/engineers/architects/engineers

*Prime + Non-Prime Attributes*

* Reminder: a PK is a type of candidate key that is chosen to uniquely ID a row



* Employee ID = PK (an employee can only work in 1 dept. = assumption)
* **Prime attribute =** all columns that are part of a candidate key 🡪 Employee ID
* **Non-prime attributes** = columns that are NOT a part of ANY candidate key 🡪 Name, Dept. ID, Dept. name



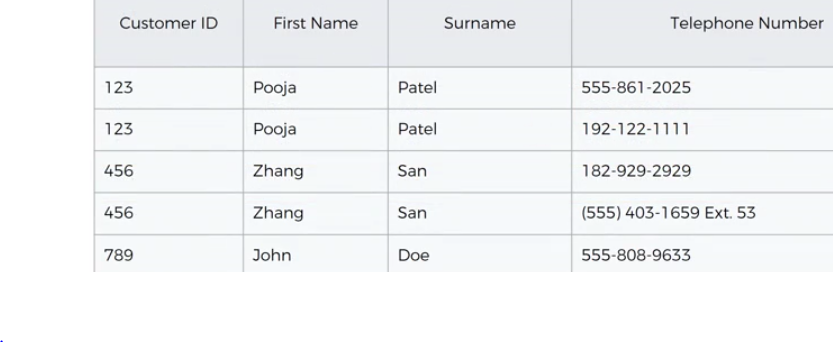
* Candidate Key = Tournament + Year 🡪 prime attributes
* Non-prim attributes = winner + winner DOB



* Prime attributes (candidate key) = Manufacturer + Model AND Model Full Name (2 CK’s)
* Model Full Name does not prevent Manufacturer + Model from being a CK on their own
* Non-prime attribute 🡪 Manufacturer Country

*First Normal Form (1NF)*

* 1NF 🡪 table that doesn’t contain duplicate rows + every cell contains only 1 value
* How to fix 🡪 remove the duplicate row or split out a cell w/ 2 values into 2 rows



* Pneumonic phrase to remember normal forms: **Every non-prime attribute must provide a fact about the key, the whole key, and nothing but the key**
* About the key = 1NF, whole key = 2NF, nothing but the key = 3NF
* “Doesn’t contain duplicate rows” = table has a key

*2NF*

* 2NF 🡪 table is already in 1NF + every *non-prime attribute* of the table is dependent on the whole of EVERY candidate key



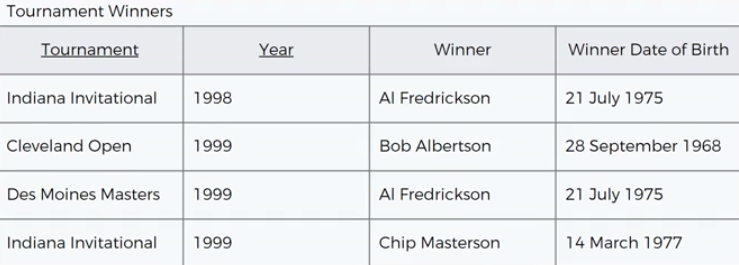
* Prime 🡪 *Model Full* *Name* and *Manufacturer + Model*
* Non-prime 🡪 Manufacturer Country
* Have to check that this is dependent on the whole of both candidate keys
* Is dependent on whole of Model Full Name b/c we cannot break it up, so this column ID’s each row uniquely
* Is NOT dependent on the *WHOLE* of Manufacturer + Model 🡪 it’s not necessary to have both Manufacturer AND model to find out country 🡪 only need Manufacturer
* Forte is always in Italy, Hoch is always is Germany, etc.
* Manufacture on its own ID’s Manufacture Country
* Why is 2NF important?
* In practice, w/out it, DB is more prone to errors + its slows down process of adding info into DB
* Whenever we add in a new row (new model for Forte) 🡪 new value for manufacturer, new value for model, 3rd col is a combo created, and 4th value is based on manufacturer
* This is *duplicating* some info + doing extra work b/c we’re writing in the country even though it’s already present in the DB
* Takes up a lot of time if inserting a lot of values
* May add Forte + “Germany” while inserting data
* Solution 🡪 new table, one for Manufacture + Model + Model Full Name and another **reference table** for Manufacturer + Manufacturer Country, linked by Manufacturer



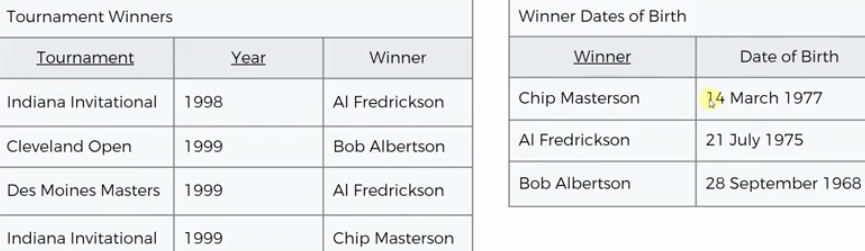
* When adding new row to right table, don’t need to add Country b/c we get it from reference table based on Manufacturer
* 1st table now has no non-prime attributes, and our reference table has a 1 prime attribute/candidate key of manufacturer (can have multiple values of country)

*3NF*

* IN 3NF if in 2NF + every non-prime attribute is **non-transitively** dependent on EVERY candidate key



* No duplicate rows = 1NF
* Candidate key/primary attributes = Tournament + Year
* Non-prime attributes = winner + Winner DOB
* Need both Tournament + Year to get Winner + Winner DOB/both are dependent on whole of Tournament + Year
* BUT winner DOB isn’t *directly* dependent on Tournament + Year 🡪 is also dependent/uniquely IDed by the “Winner” 🡪 a transition from candidate key to Winner to Winner DOB
* Why is 3NF important?
* More prone to duplicate errors 🡪 if we want to add another Tournament + Year w/ another Winner, if a Winner has already won, that DOB is already in the DB and we’re replicating it again
* Unnecessary to have that DOB in that table twice
* Slows processing speed for the DB when inserting those types of rows (massive difference w/ large inserts)
* Chance of errors of inputting wrong DOB for a winner if doing manually
* How to solve 🡪 Add a reference table for the Winner + Winner DOB, linked by Winner



* Only have 1 non-primary attribute in each table, so they must be transitively dependent on their candidate key(s) (**functional dependency**)
* 3NF of DB’s is very powerful + one of the top normalization forms of DB’s (there are, in fact, higher levels, but are unnecessarily complex compared to value added)
* Free from INSERT/UPDATE/DELETE errors
* Ideal normal form for dealing w/ OLTP applications (very high computation requirement to input millions + millions of transactions in short periods of time)