Database Design Cheatsheet

Identify the domain entities. For example, for an online store:

- Customers
- Products
- Reviews

If there is data that doesn't fit in this entities categories, probably it's an attribute.

Customers

Products

Reviews

1. Define Domain Entities

Define the attributes for each entity.

For example:

- Customer's attributes can be: email, first_name, last_name, etc.
- Product attributes: id, name, description, price

Decide what primary keys to use: email for a customer (or assign an unique number id), id for a product (which could be the barcode number, or an unique auto generated number)

Customers			
id	email	first_name	last_name

Products			
id	name	description	price

Reviews			
customer_id	product_id	rating	comments

2. Define Attributes & Primary Keys

1NF

Atomic values for each attribute. No values like: "'phoneX, tabletY, laptopZ" for a 'favorite_product' attribute in Customers table.

Non Repeating groups: No columns (attributes) like: favorite_product1, favorite_product2, favorite_product3 in the Customers table

Records (rows) are unique. A customer with id = 111, appears only once in the Customers Table

2NF

Non-key (non prime) attributes are depended on the whole composite key. Eg: Column 'price' in Reviews table breaks 2NF. It is depended only on the product_id.

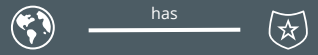
All tables that have only **one attribute as primary key are in 2NF** by default.

3NF

[Every] non-key [attribute] must be depended on the key, the whole key, and nothing but the key.
Adding 'brand' and 'brand_description' in Products table breaks 3NF. 'brand_description' is depended on 'brand', which is not a key.

3. Apply Normalization 1FN, 2FN, 3FN

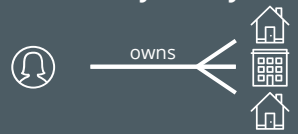
One-to-one



One Country — **One Capital City**

- Foreign Key can be on each side.
- It can be forced using an UNIQUE constraint.

One-to-Many / Many-to-one



One (Rich) Person — **Many Houses**

- Foreign Key is on the Houses side referencing a person's id (Houses.person_id)

Many-to-Many

A customer reviews many products — **A product is reviewed by many customers**

- Use intermediary table that holds the foreign keys.

5. Build the tables relationships

Filter out redundancies.

Examine the Database design and if redundancies are found, remove them.

Review with Client

Review the specifications with your client to ensure that the database model supports the data requirements.

6. Polish the design

The Complete Database Design & Modeling Beginners Tutorial (https://goo.gl/nBK357)