SQL Day 3

The Data Analytics Bootcamp

Goals

By the end of the class, you will be able to:

□ Normalize Data

□ Understand Data Relationships

□ Create Entity Relationship Diagrams

Data Normalization

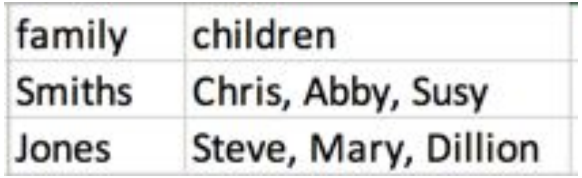
Data Normalization

- Process of restructure data to a set of “normal forms”
- Reduce and eliminate data redundancy and inconsistencies.
- Three most common forms:
  - First normal form (1NF)
  - Second normal form (2NF)
  - Third normal form (3NF)
- There are even more levels!

First Normal Form (1NF)

- Each field in a table row should contain a single value
- Each row is unique
  - ○ Rows can have a fields that repeat
  - ○ But whole rows do not fully match

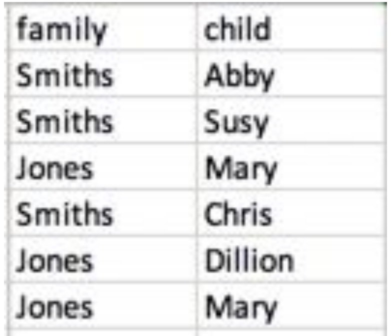
Raw Data



family	children
Smiths	Chris, Abby, Susy
Jones	Steve, Mary, Dillion

Normalization

First Normal Form

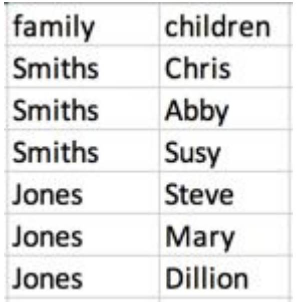


family	child
Smiths	Abby
Smiths	Susy
Jones	Mary
Smiths	Chris
Jones	Dillion
Jones	Mary

Second Normal Form (2NF)

- Be in First Normal Form
- Single Column Primary Key
  - Primary Key
  - Identifies the table and row uniquely
- Generally there could be a need to create a new table

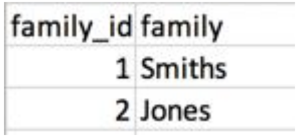
Data in 1NF



family	children
Smiths	Chris
Smiths	Abby
Smiths	Susy
Jones	Steve
Jones	Mary
Jones	Dillion

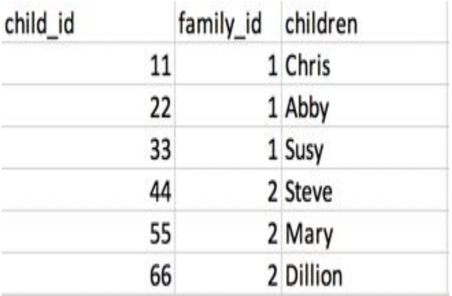
2NF Normalization

Family Table



family_id	family
1	Smiths
2	Jones

Child Table



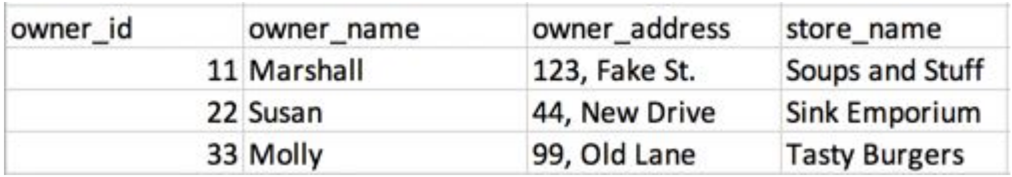
child_id	family_id	children
11	1	Chris
22	1	Abby
33	1	Susy
44	2	Steve
55	2	Mary
66	2	Dillion

Transitive Dependence

- Transitive Dependence is the reliance of a column's value on another column through a third column.
- Transitive
  - If X>Y and Y>Z then A>Z.
- Dependence
  - One value relies on another.
  - City relies on ZIP code; age depends on birthday.
- For example:
  - Say you have three columns: StoreName, OwnerAddress, OwnerName.
  - OwnerName and OwnerAddress rely on the the StoreName
  - OwnerAddress also relies on the OwnerName.
  - So OwnerAddress relies on the StoreName via the OwnerName

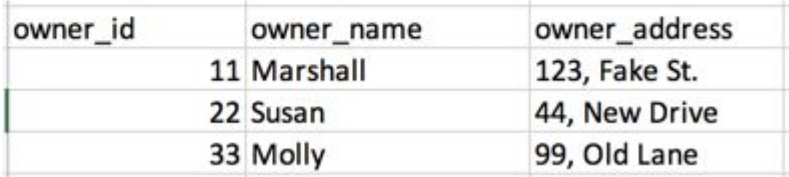
Third Normal Form

- Must be in Second Normal Form
- Contain non-transitively dependent columns

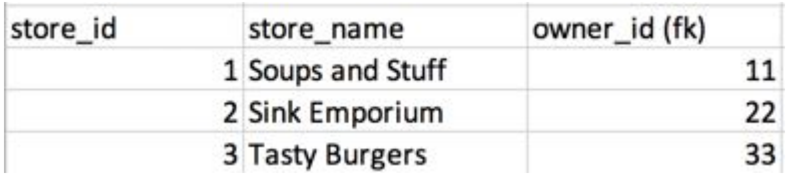


owner_id	owner_name	owner_address	store_name
11	Marshall	123, Fake St.	Soups and Stuff
22	Susan	44, New Drive	Sink Emporium
33	Molly	99, Old Lane	Tasty Burgers

3NF Normalization



owner_id	owner_name	owner_address
11	Marshall	123, Fake St.
22	Susan	44, New Drive
33	Molly	99, Old Lane



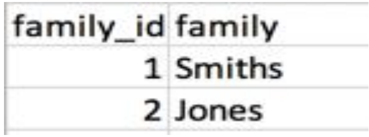
store_id	store_name	owner_id (fk)
1	Soups and Stuff	11
2	Sink Emporium	22
3	Tasty Burgers	33

Foreign Keys

Foreign Keys

- Foreign Keys reference the primary key of another table
  - Can have a different name
  - Do not need to be unique

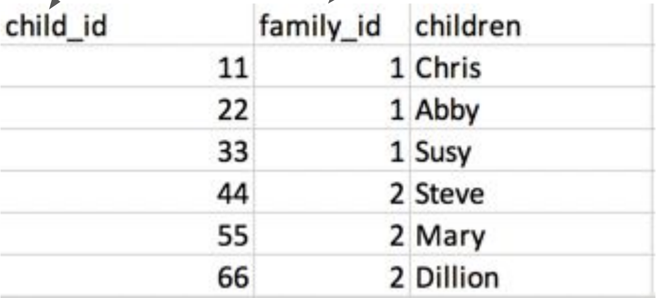
Primary Key



family_id	family
1	Smiths
2	Jones

Primary Key

Foreign Key



child_id	family_id	children
11	1	Chris
22	1	Abby
33	1	Susy
44	2	Steve
55	2	Mary
66	2	Dillion

Goals Review

Goals

By the end of the class, you will be able to:

☑ **Normalize Data**

□ Understand Data Relationships

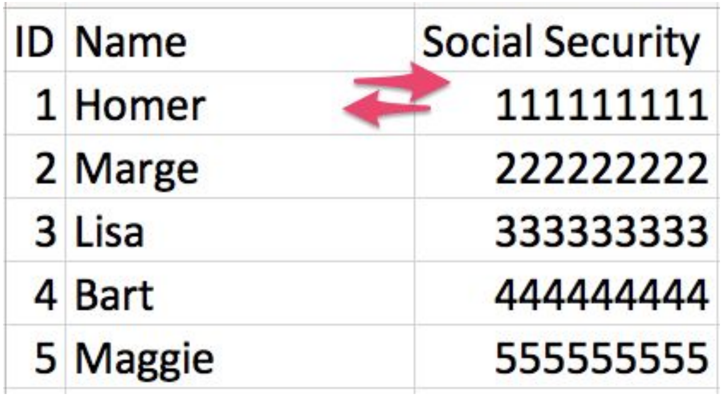
□ Create Entity Relationship Diagrams

Data Relationships

Data Relationships

- One-to-One
- One-to-Many
- Many-to-Many

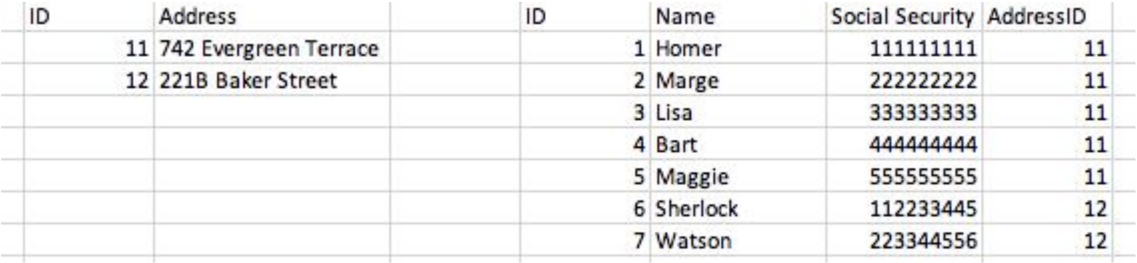
One-to-One Relationship



ID	Name	Social Security
1	Homer	111111111
2	Marge	222222222
3	Lisa	333333333
4	Bart	444444444
5	Maggie	555555555

- Each item in one column is linked to only one other item from the other column.
- Here, each person in the Simpsons family can have only one social security number.
- Each social security number can be assigned only to one person.

One-To-Many

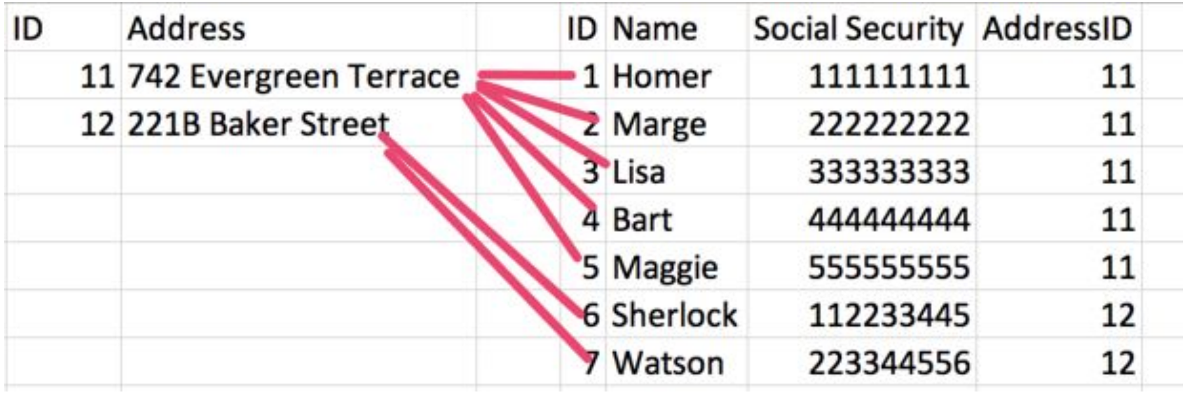


ID	Address
11	742 Evergreen Terrace
12	221B Baker Street

ID	Name	Social Security	AddressID
1	Homer	111111111	11
2	Marge	222222222	11
3	Lisa	333333333	11
4	Bart	444444444	11
5	Maggie	555555555	11
6	Sherlock	112233445	12
7	Watson	223344556	12

- Two tables: one for people, another for addresses
- Each person has only one address
- But each address can be associated with multiple people

One-To-Many

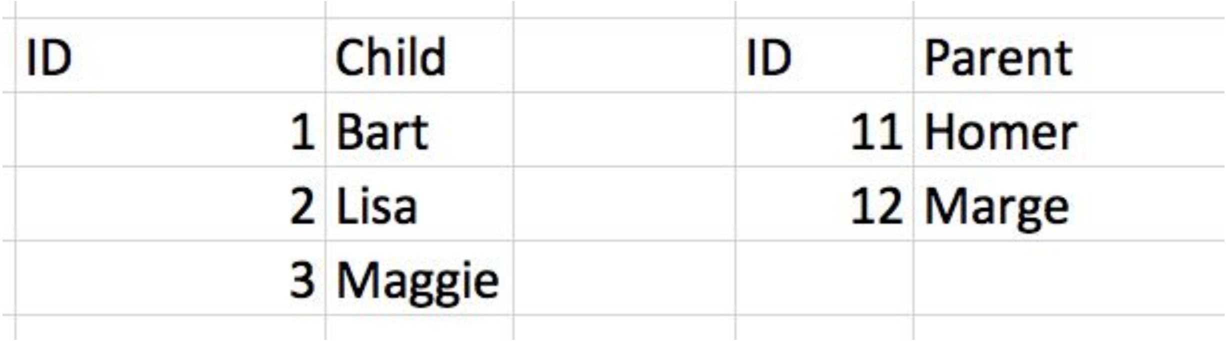


ID	Address
11	742 Evergreen Terrace
12	221B Baker Street

ID	Name	Social Security	AddressID
1	Homer	111111111	11
2	Marge	222222222	11
3	Lisa	333333333	11
4	Bart	444444444	11
5	Maggie	555555555	11
6	Sherlock	112233445	12
7	Watson	223344556	12

- The two tables, joined, would look like this.
- Each person has an address.
- Each address can be associated with multiple people.

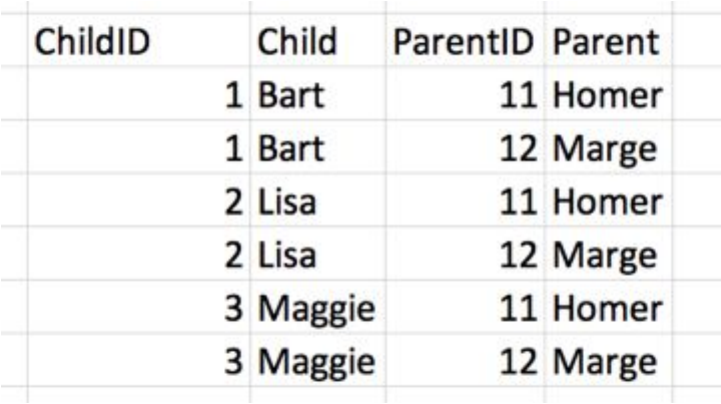
Many-To-Many



ID	Child		ID	Parent
1	Bart		11	Homer
2	Lisa		12	Marge
3	Maggie			

- Each child can have more than one parent.
- Each parent can have more than one child.

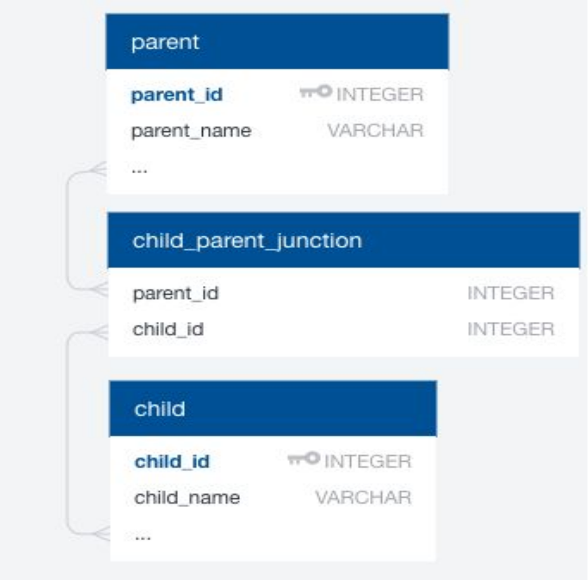
Many-To-Many



ChildID	Child	ParentID	Parent
1	Bart	11	Homer
1	Bart	12	Marge
2	Lisa	11	Homer
2	Lisa	12	Marge
3	Maggie	11	Homer
3	Maggie	12	Marge

- Each child can have more than one parent.
- Each parent can have more than one child.
- The two tables are joined in a junction table.

Junction Table



parent

parent_id INTEGER
parent_name VARCHAR
...

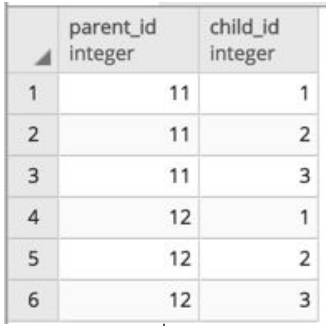
child_parent_junction

parent_id INTEGER
child_id INTEGER

child

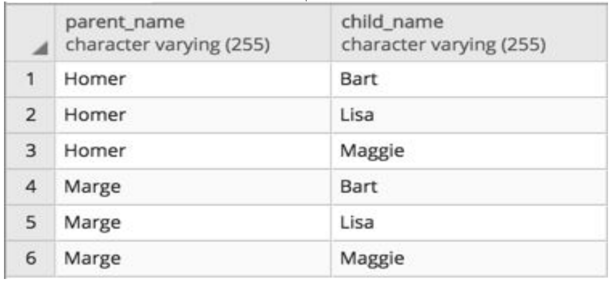
child_id INTEGER
child_name VARCHAR
...

- The Junction table contains many parent_ids and many child_ids



	parent_id integer	child_id integer
1	11	1
2	11	2
3	11	3
4	12	1
5	12	2
6	12	3

Join child and parent table to junction table



	parent_name character varying (255)	child_name character varying (255)
1	Homer	Bart
2	Homer	Lisa
3	Homer	Maggie
4	Marge	Bart
5	Marge	Lisa
6	Marge	Maggie

Entity Relationship Diagrams

ERDs

An **E**ntity **R**elationship **D**iagram provides a visual method of modeling data.

Entities, their data types, and relationships are all illustrated in the diagram.

There are three models used when creating diagrams:

- **Conceptual**: basic information containing table and column names.
- **Logical**: slightly more complex than conceptual models with IDs and data types defined.
- **Physical**: the blueprint of the database, reflecting physical relationships between entities.

Goals Review

Goals

By the end of the class, you will be able to:

☑ Normalize Data

☑ Understand Data Relationships

☑ Create Entity Relationship Diagrams