Database Normalization

ISTA 322 - Data Engineering

What is normalization?

- I’ve referred to it a lot but haven’t directly explained.
- Normalization is a way to organize data in a RDB.
- Follows a set of rules
  - Increase flexibility
  - Reduce redundancy
  - Consistent format and dependency
- The rules make up a series of ‘normal forms’
  - Three main normal forms we’ll focus on

Normal forms

- 1st normal form
  - Remove repeating groups within tables
  - Separate tables for related data
  - Each set of related data should have a **primary key**

Normal forms

- 1st normal form
  - Remove repeating groups within tables
  - Separate tables for related data
  - Each set of related data should have a **primary key**
- 2nd normal form
  - Separate tables for values that apply to multiple records
  - These are related to others with a **foreign key**

Normal forms

- 1st normal form
  - Remove repeating groups within tables
  - Separate tables for related data
  - Each set of related data should have a **primary key**
- 2nd normal form
  - Separate tables for values that apply to multiple records
  - These are related to others with a **foreign key**
- 3rd normal form
  - Remove fields that don’t rely on the key

Let’s look at a simple example - Data

Table: top_track_info

artist_name	artist_id	song_id	song_name	streams	genre_1	genre_2	followers
Dance w/t Dead	x88928	a99189	diabolic	75092	synthwave	darkwave	156000
Dance w/t Dead	x88928	a73198	invader	93910	synthwave	darkwave	156000
Frank Sinatra	z99029	a23812	witchcraft	21048820	traditional		12020900
Frank Sinatra	z99029	a83012	angel eyes	14029901	traditional		12020900
ODESZA	y88420	a01818	Meridian	5401928	electronic	indie	6800700
ODESZA	y88420	a01912	Bloom	5691321	electronic	indie	6800700

Sample non-normal data

- Might be good for analysis
- Common format for CSV/excel
- But lots and lots of extra info
- Difficult to update/change

Let's look at a simple example - 1NF

Table: top_track_info

artist_name	artist_id	song_id	song_name	streams	genre_1	genre_2	followers
Dance w/t Dead	x88928	a99189	diabolic	75092	synthwave	darkwave	156000
Dance w/t Dead	x88928	a73198	invader	93910	synthwave	darkwave	156000
Frank Sinatra	z99029	a23812	witchcraft	21048820	traditional		12020900
Frank Sinatra	z99029	a83012	angel eyes	14029901	traditional		12020900
ODESZA	y88420	a01818	Meridian	5401928	electronic	indie	6800700
ODESZA	y88420	a01912	Bloom	5691321	electronic	indie	6800700

Repeating groups

- List of data in a column/field
- Or multiple fields with the same data

- Must 'stack' columns

Let’s look at a simple example - 1NF

Table: top_track_info

artist_name	artist_id	song_id	song_name	streams	genre	followers
Dance w/t Dead	x88928	a99189	diabolic	75092	synthwave	156000
Dance w/t Dead	x88928	a73198	invader	93910	synthwave	156000
Dance w/t Dead	x88928	a99189	diabolic	75092	darkwave	156000
Dance w/t Dead	x88928	a73198	invader	93910	darkwave	156000
Frank Sinatra	z99029	a23812	witchcraft	21048820	traditional	12020900
Frank Sinatra	z99029	a83012	angel eyes	14029901	traditional	12020900
ODESZA	y88420	a01818	Meridian	5401928	electronic	6800700
ODESZA	y88420	a01912	Bloom	5691321	electronic	6800700
ODESZA	y88420	a01818	Meridian	5401928	indie	6800700
ODESZA	y88420	a01912	Bloom	5691321	indie	6800700

Repeating groups

- List of data in a column/field
- Or multiple fields with the same data

- Must ‘stack’ columns

Let’s look at a simple example - 1NF

Table: top_track_info						
artist_name	**artist_id**	**song_id**	**song_name**	**streams**	**genre**	**followers**
Dance w/t Dead	x88928	a99189	diabolic	75092	synthwave	156000
Dance w/t Dead	x88928	a73198	invader	93910	synthwave	156000
Dance w/t Dead	x88928	a99189	diabolic	75092	darkwave	156000
Dance w/t Dead	x88928	a73198	invader	93910	darkwave	156000
Frank Sinatra	z99029	a23812	witchcraft	21048820	traditional	12020900
Frank Sinatra	z99029	a83012	angel eyes	14029901	traditional	12020900
ODESZA	y88420	a01818	Meridian	5401928	electronic	6800700
ODESZA	y88420	a01912	Bloom	5691321	electronic	6800700
ODESZA	y88420	a01818	Meridian	5401928	indie	6800700
ODESZA	y88420	a01912	Bloom	5691321	indie	6800700

Table: artist_genre	
artist_id	**genre**
x88928	synthwave
x88928	darkwave
z99029	traditional
y88420	electronic
y88420	indie

Repeating groups

- Break into own table

Let’s look at a simple example - 1NF

Table: top_track_info					
artist_name	**artist_id**	**song_id**	**song_name**	**streams**	**followers**
Dance w/t Dead	x88928	a99189	diabolic	75092	156000
Dance w/t Dead	x88928	a73198	invader	93910	156000
Frank Sinatra	z99029	a23812	witchcraft	21048820	12020900
Frank Sinatra	z99029	a83012	angel eyes	14029901	12020900
ODESZA	y88420	a01818	Meridian	5401928	6800700
ODESZA	y88420	a01912	Bloom	5691321	6800700

Table: artist_genre	
artist_id	**genre**
x88928	synthwave
x88928	darkwave
z99029	traditional
y88420	electronic
y88420	indie

Repeating groups

- Break into own table
- Reduce other table

Let's look at a simple example

Table: top_track_info					
artist_name	**artist_id**	**song_id**	**song_name**	**streams**	**followers**
Dance w/t Dead	x88928	a99189	diabolic	75092	156000
Dance w/t Dead	x88928	a73198	invader	93910	156000
Frank Sinatra	z99029	a23812	witchcraft	21048820	12020900
Frank Sinatra	z99029	a83012	angel eyes	14029901	12020900
ODESZA	y88420	a01818	Meridian	5401928	6800700
ODESZA	y88420	a01912	Bloom	5691321	6800700

Foreign key

Table: artist_genre	
artist_id	**genre**
x88928	synthwave
x88928	darkwave
z99029	traditional
y88420	electronic
y88420	indie

Foreign key

Repeating groups

- Break into own table
- Reduce other table
- Foreign key in artist_info
- Foreign key in top_track_info

Let’s look at a simple example

Table: top_track_info

artist_name	artist_id	song_id	song_name	streams	followers
Dance w/t Dead	x88928	a99189	diabolic	75092	156000
Dance w/t Dead	x88928	a73198	invader	93910	156000
Frank Sinatra	z99029	a23812	witchcraft	21048820	12020900
Frank Sinatra	z99029	a83012	angel eyes	14029901	12020900
ODESZA	y88420	a01818	Meridian	5401928	6800700
ODESZA	y88420	a01912	Bloom	5691321	6800700

Foreign key

Primary key

Foreign key

Table: artist_genre

artist_id	genre
x88928	synthwave
x88928	darkwave
z99029	traditional
y88420	electronic
y88420	indie

Repeating groups

- Break into own table
- Reduce other table
- Foreign key in artist_info
- Foreign key in top_track_info

Let’s look at a simple example - 2NF

Table: top_track_info

artist_name	artist_id	song_id	song_name	streams	followers
Dance w/t Dead	x88928	a99189	diabolic	75092	156000
Dance w/t Dead	x88928	a73198	invader	93910	156000
Frank Sinatra	z99029	a23812	witchcraft	21048820	12020900
Frank Sinatra	z99029	a83012	angel eyes	14029901	12020900
ODESZA	y88420	a01818	Meridian	5401928	6800700
ODESZA	y88420	a01912	Bloom	5691321	6800700

Redundant info

- Artist info isn’t dependent on song info

Table: artist_genre

artist_id	genre
x88928	synthwave
x88928	darkwave
z99029	traditional
y88420	electronic
y88420	indie

Let’s look at a simple example - 2NF

Table: top_track_info				
artist_id	**song_id**	**song_name**	**streams**	**followers**
x88928	a99189	diabolic	75092	156000
x88928	a73198	invader	93910	156000
z99029	a23812	witchcraft	21048820	12020900
z99029	a83012	angel eyes	14029901	12020900
y88420	a01818	Meridian	5401928	6800700
y88420	a01912	Bloom	5691321	6800700

Redundant info

- Artist info isn’t dependent on song info
- Split off

Table: artist_genre	
artist_id	**genre**
x88928	synthwave
x88928	darkwave
z99029	traditional
y88420	electronic
y88420	indie

Table: artist_info	
artist_name	**artist_id**
Dance w/t Dead	x88928
Frank Sinatra	z99029
ODESZA	y88420

Let’s look at a simple example - 3NF

Table: top_track_info				
artist_id	**song_id**	**song_name**	**streams**	**followers**
x88928	a99189	diabolic	75092	156000
x88928	a73198	invader	93910	156000
z99029	a23812	witchcraft	21048820	12020900
z99029	a83012	angel eyes	14029901	12020900
y88420	a01818	Meridian	5401928	6800700
y88420	a01912	Bloom	5691321	6800700

Remove data not dependent on key

- Followers isn’t dependent on key of song_id
- Make own table?
- Add to artist_info

Table: artist_genre	
artist_id	**genre**
x88928	synthwave
x88928	darkwave
z99029	traditional
y88420	electronic
y88420	indie

Table: artist_info	
artist_name	**artist_id**
Dance w/t Dead	x88928
Frank Sinatra	z99029
ODESZA	y88420

Let’s look at a simple example - 3NF

Table: top_track_info

artist_id	song_id	song_name	streams
x88928	a99189	diabolic	75092
x88928	a73198	invader	93910
z99029	a23812	witchcraft	21048820
z99029	a83012	angel eyes	14029901
y88420	a01818	Meridian	5401928
y88420	a01912	Bloom	5691321

Table: artist_genre

artist_id	genre
x88928	synthwave
x88928	darkwave
z99029	traditional
y88420	electronic
y88420	indie

Table: artist_info

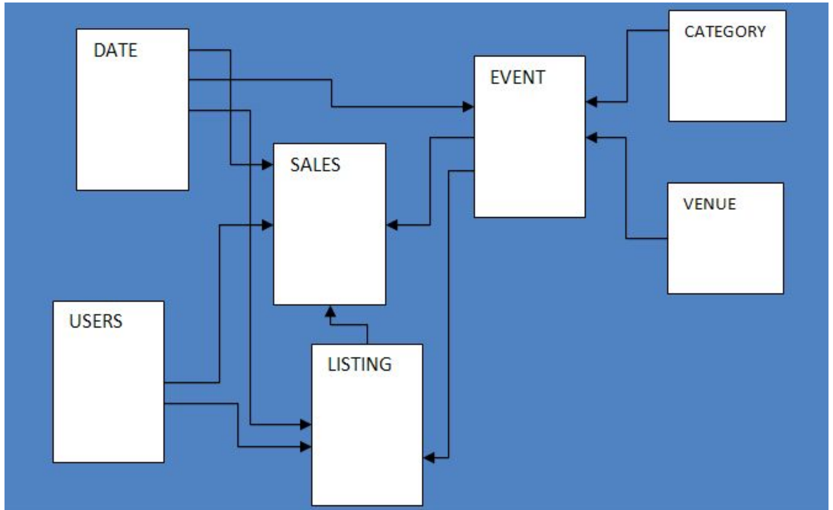
artist_name	artist_id	followers
Dance w/t Dead	x88928	156000
Frank Sinatra	z99029	12020900
ODESZA	y88420	6800700

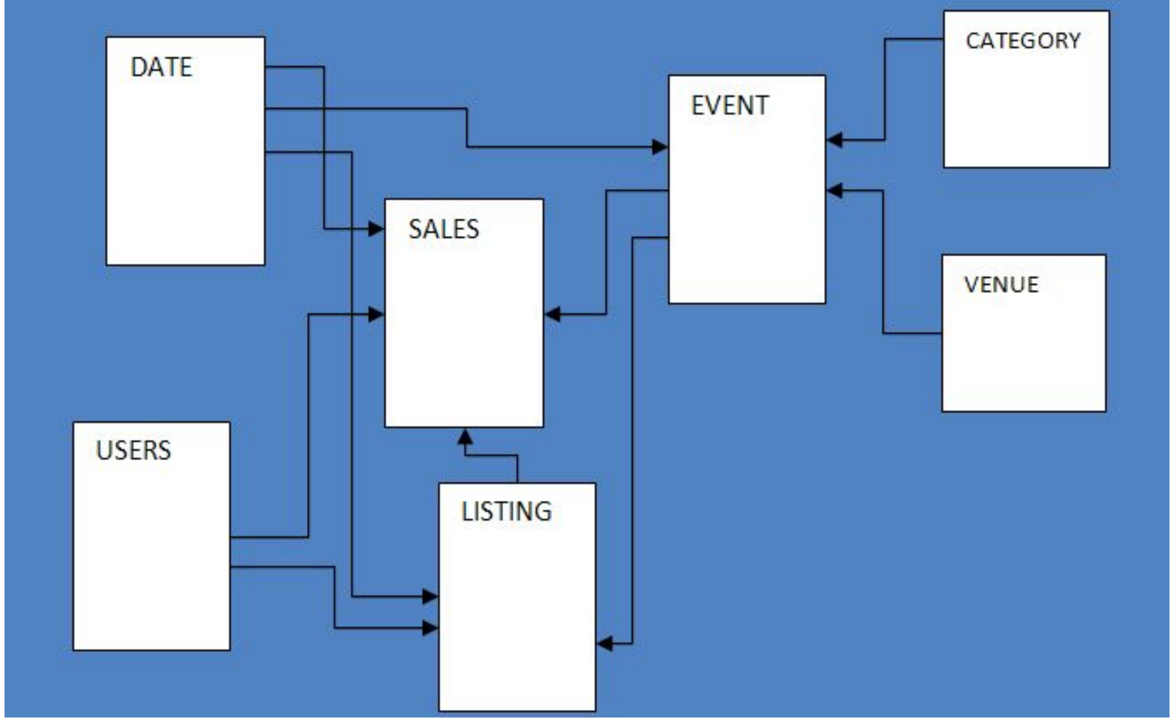
Remove data not dependent on key

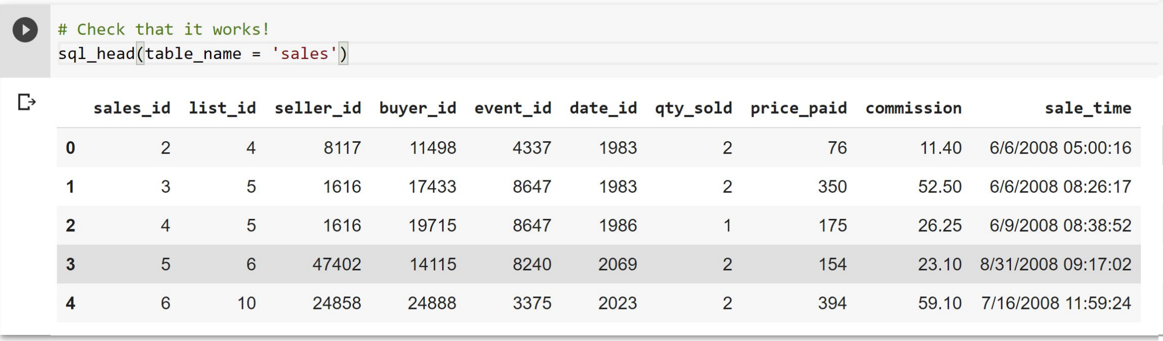
- Followers isn’t dependent on key of song_id
- Make own table?
- Add to artist_info?

Creating a schema

- Arrangement of tables is called a schema
- Look at head of raw data
- Hand draw tables to normalize
- Or list columns
- Schema of sales data

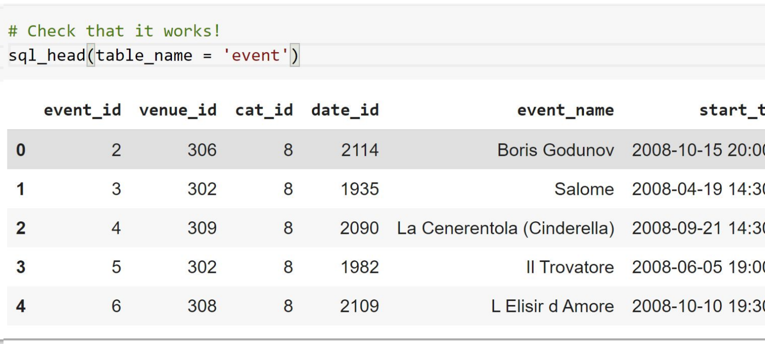






```
# Check that it works!
sql_head(table_name = 'sales')
```

	sales_id	list_id	seller_id	buyer_id	event_id	date_id	qty_sold	price_paid	commission	sale_time
0	2	4	8117	11498	4337	1983	2	76	11.40	6/6/2008 05:00:16
1	3	5	1616	17433	8647	1983	2	350	52.50	6/6/2008 08:26:17
2	4	5	1616	19715	8647	1986	1	175	26.25	6/9/2008 08:38:52
3	5	6	47402	14115	8240	2069	2	154	23.10	8/31/2008 09:17:02
4	6	10	24858	24888	3375	2023	2	394	59.10	7/16/2008 11:59:24



```
# Check that it works!
sql_head(table_name = 'event')
```

	event_id	venue_id	cat_id	date_id	event_name	start_t
0	2	306	8	2114	Boris Godunov	2008-10-15 20:0
1	3	302	8	1935	Salome	2008-04-19 14:3
2	4	309	8	2090	La Cenerentola (Cinderella)	2008-09-21 14:3
3	5	302	8	1982	Il Trovatore	2008-06-05 19:0
4	6	308	8	2109	L Elisir d Amore	2008-10-10 19:3

Doing this in Python

- Split into different dataframes by selecting columns
- Reduce rows if needed!
  - drop_duplicates() is helpful!
- Generate keys before or after split?
- Connect to RDB and upload

Doing this in Python

Table: top_track_info

artist_name	artist_id	song_id	song_name	streams	genre	followers
Dance w/t Dead	x88928	a99189	diabolic	75092	synthwave	156000
Dance w/t Dead	x88928	a73198	invader	93910	synthwave	156000
Dance w/t Dead	x88928	a99189	diabolic	75092	darkwave	156000
Dance w/t Dead	x88928	a73198	invader	93910	darkwave	156000
Frank Sinatra	z99029	a23812	witchcraft	21048820	traditional	12020900
Frank Sinatra	z99029	a83012	angel eyes	14029901	traditional	12020900
ODESZA	y88420	a01818	Meridian	5401928	electronic	6800700
ODESZA	y88420	a01912	Bloom	5691321	electronic	6800700
ODESZA	y88420	a01818	Meridian	5401928	indie	6800700
ODESZA	y88420	a01912	Bloom	5691321	indie	6800700

Table: artist_genre

artist_id	genre
x88928	synthwave
x88928	darkwave
z99029	traditional
y88420	electronic
y88420	indie

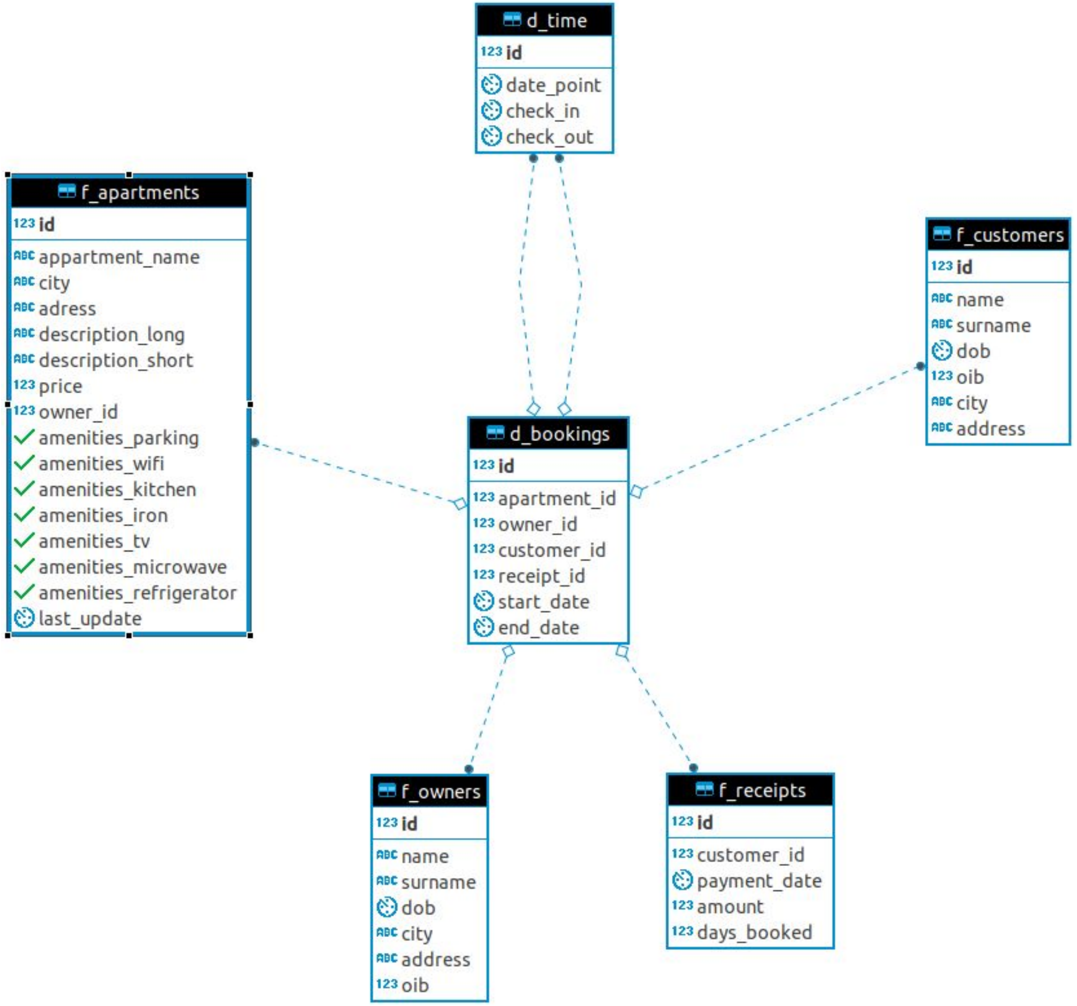
```
artist_genre =
top_track_info[[
    ‘artist_id’,
    ‘genre’]]

artist_genre =
    artist_genre.
    drop_duplicates()
```

Normalization isn’t the only way

- Star and Snowflake schema
- Fact table
  - Core measurements
  - e.g. Sales/bookings/listens
  - Highly granular
  - Foreign keys to dimension tables
- Dimension table
  - Descriptions of elements in fact table
- Often used in data warehouses

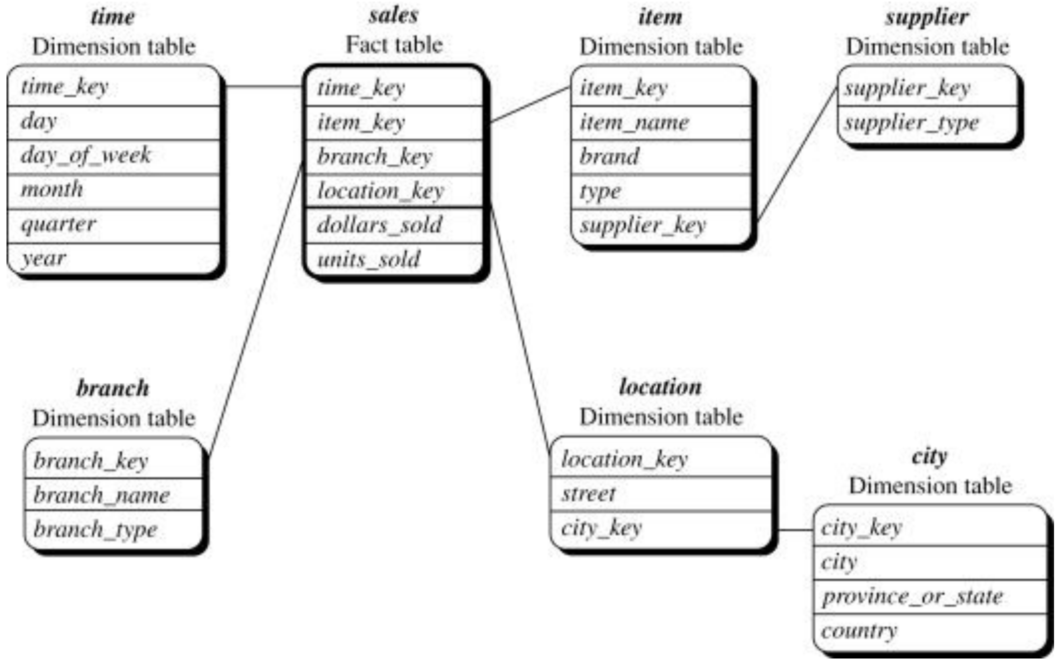
Star Schema

- AirBnB
- Fact
  - bookings
- Dimensions
  - apartments
  - time
  - customer



Snowflake Schema

- Similar to Star
- Further normalization in dimension tables



Wrapping up Normalization

- Normalization is key to making a database stable, maintainable, and efficient.
- Tradeoffs are made regarding how far you want to normalize
  - Could balloon into tons of tables
- Standard normalization is not the only way!
- This is not a RDB design class.