* Denormalized data contains repetitions that can cause anomalies
* Rules exist to normalize data
* First Normal Form:
  + - Single-valued columns
    - No repeating columns
    - Consistent data across a column
    - Uniquely identify a row
* Second Normal Form: No partial dependencies
* Third Normal Form: No transitive dependencies
* Sometimes, it's OK to violate normal forms; use your best judgement

| **Key Term** | **Definition** |
| --- | --- |
| First Normal Form (1NF) | Shaping data to eliminate inconsistencies, and allowing unique identification of each row |
| Second Normal Form (2NF) | Extends 1NF by removing partial dependencies |
| Third Normal Form (3NF) | Extends 2NF by removing transitive dependencies |
| One-One Relationship | When one entity "has one" of another entity, and that second entity "belongs to" only the first. For example, entity "user" and "home address" have a one-one relationship |
| One-Many Relationship | When one entity "has many" of another entity, and that second entity "belongs to" only the first. For example, entity "user" and "email address" have a one-many relationship, because a user can have many email addresses, but each email address belongs to only one user |
| Many-Many Relationship | When two entities are related in such a way where many links can exist on both sides. For example, entities "books" and "categories" have a many-many relationship, because a book can have multiple categories, and a category can belong to multiple books |
| Candidate Key | A set of one or more columns that can uniquely identify a row in a database table |
| Primary Key | The key from the set of candidate keys that we actually choose in order to uniquely identify a row in that table |
| Composite Key | A key that is composed of more than one column |
| Partial Dependency | When a non-key column depends on only part of the primary key |
| Transitive Dependency | When a non-key column depends on the primary key through another non-key column |

| **Key Term** | **Definition** |
| --- | --- |
| CREATE TABLE | Allows you to create tables in Postgres. |
| INTEGER | A numeric data type that stores numbers without decimals. |
| SERIAL | A numeric data type stored as an INTEGER, but that is incremented by one automatically by Postgres each time a new row is added. |
| REAL | An inexact numeric data type, with precision to six decimal points. |
| DOUBLE PRECISION | An inexact numeric data type, with precision to 15 decimal points. |
| DECIMAL | An exact numeric data type that includes decimal values; also known as NUMERIC. |
| CHARACTER(n) | Stores text data of length (n). Anything shorter is padded. |
| VARCHAR | Stores text of varying data lengths (can be limited by an “n”). |
| TEXT | Stores much longer texts of varying data lengths. |
| TIMESTAMP | Stores data and time data. Use WITH/WITHOUT TIME ZONE to determine whether a time zone is associated with it (which Postgres updates to match the database’s time zone). |
| DATE | Stores date data. |
| TIME | Stores time data. |
| JSONB | Store more complex data in key-value pairs. |
| ALTER TABLE | Change table structure to add/remove columns or change the data type of a column. |
| DROP | Remove a table from the database |
| TRUNCATE | Remove all data in a table |
| COMMENT | Add a text comment to a table or column |

| **Key Term** | **Definition** |
| --- | --- |
| INSERT ... VALUES | One form of adding data into a table; used when introducing new data in a table. This data would come from an external source like an application. |
| INSERT ... SELECT | One form of adding data into a table; used when taking already existing data from a table and migrating it — most often with some modifications or clean-ups — into an already existing table. |
| UPDATE | Used to update rows of data within a given column with new values. |
| DELETE | Used to delete some portion of data from a table. |
| BEGIN | Starts a transaction. |
| COMMIT | Tells the system to attempt to complete the transaction (make the requested changes). Similar functionality is also achieved with END. |
| ROLLBACK | Tells the system to not commit any changes as part of the current transaction, discarding the changes. |
| ACID | An acronym that describes the transactional guarantees provided by a relational database. |
| Atomicity | The database guarantees that a transaction will either register all the commands in a transaction, or none of them. |
| Consistency | The database guarantees that a successful transaction will leave the data in a consistent state, one that obeys all the rules that you've setup. We've seen simple rules like limiting the number of characters in a VARCHAR column, and we'll see many more in the next lesson. |
| Isolation | The database guarantees that concurrent transactions don't "see each other" until they are committed. Committing a transaction is a command that tells the database to execute all the commands we passed to it since we started that transaction. |
| Durability | The database guarantees that once it accepts a transaction and returns a success, the changes introduced by the transaction will be permanently stored on disk, even if the database crashes right after the success response. |

| **Key Term** | **Definition** |
| --- | --- |
| Constraint | A rule that can be added at the table or column level to restrict insertion and updates of data based on business rules. |
| Unique Constraint | Ensures that a column or set of columns are unique across all the rows of the table. |
| Primary Key Constraint | Like a unique constraint, it enforces unique values across a column or set of columns. In addition to that, it also enforces a NOT NULL, which is another database constraint that can be used by itself to ensure that a column's values cannot be null. Lastly, there can only be one of these for a given table. |
| Surrogate Key | A primary key that is composed of a value not present in the business data, an artificial value created only for the purpose of uniquely identifying the rest of the data. It is not exposed to users of the system. |
| Natural key | When a primary key is composed of a value that is present in the business data and exposed to users of the system. |
| Foreign Key Constraint | Restricts the values in a column to only values that appear in another column. They're often used to relate IDs in relationships between tables, thereby preserving what we call "referential integrity". In many cases, the foreign key will refer to a primary key in another table, but that is not necessary. Any column can be referenced by a foreign key constraint. |
| Referential Integrity | The property of columns referencing other entities to be consistent and valid, only referring to existing data. |
| ON DELETE CASCADE | When the referenced data gets deleted, the referencing rows of data will be automatically deleted as well. |
| ON DELETE SET NULL | When the referenced data gets deleted, the referring column will have its value set to NULL. Since NULL is a special value, it won't break the foreign key constraint because it will be clear that the row of data is now referencing absolutely nothing. |
| Check Constraint | Allows one to implement custom business rules at the level of the database, such as "a product can't have a negative quantity". |