--Welcome to JBooks--

This document is designed to hold information for a small online bookstore, and provides tables based on various inputs such as books (including genres), authors, customers, and orders.

Down below are the Tables described, they will be shown as 1NF, 2NF, and 3NF. They will also be given a description on how they meet those requirements, an explanation of the relationship between each of the following tables, and an explanation of the benefits of using a database for this set of data.

Entity-Relationship Diagram

Those with the same colour background (aside from Authors, Books, and Genres) will be shared across tables.

Authors	Books	Genres
author_id (PK)	book_id	genre_id
author_name	book_title	genre_name
author_birth	isbn	
	publish_year	
	price	
	author_id (FK)	

Customers	Orders	Order Details	Book Genres
cust_id (PK)	order_id (PK)	order_id (FK)	book_id (FK)
cust_name	cust_id (FK)	book_id (FK)	genre_id (FK)
cust_email	order_date	quantity	
cust_address	status	price	

In these two diagrams, this shows two things:

The One-to-Many relationship (Authors -> Books, Customers -> Orders

The Many-to-Many relationship (Books <-> Genres, Orders <-> Books)

The One-to-Many relationship is an association between two tables in a database, where a record of information in one table can be associated to other records in different tables.

An example of this is author_id in the Authors table, being brought over to the Books table under FK (Foreign Key)

The Many-to-Many relationship, however, a complex interaction between multiple data entities in a database.

An example for this is Books and Genres, where a single book can belong to multiple genres, and a genre can contain many books. To handle that type of relationship, that's why the Book Genre table was created.

-- SECTION: NORMALIZATION --

This section is to provide information of each of the three normalization groups, how and why each of the tables meet 1NF, 2NF, and 3NF.

In 1NF, each table cell should generally contain a single value, and each column, a unique name. The first normal helps to eliminate duped data and simplifies queries. (order of stored data does not matter)

2NF is when all non-key attributes are fully dependant on the primary key (each column should be directly related to the PK and no other columns.)

3NF builds on 2NF in the sense that it requires all non-key attributes be independent of one another. In other words, each column should be related directly to the primary key, and not to any other columns in the same table

The following shows each table and whether it is in any of the three (or all of the three) normalization forms:

Books

-1NF: Yes (all columns are atomic)

-2NF: Yes (no partial dependencies and all non-key attributes depend on the book_id PK)

-3NF: Yes (zero transitive dependencies listed)

Authors

-1NF: Yes (all atomic columns)

-2NF: Yes (no partial dependencies and all non-key attributes depend on the author_id)

-3NF: Yes (zero transitive dependencies listed)

Genres

-1NF: Yes (all columns are atomic)

-2NF: Yes (no partial dependencies)

-3NF: Yes (zero transitive dependencies listed)

Customers

-1NF: Yes (all columns are atomic)

-2NF: Yes (no partial dependencies)

-3NF: Yes (zero transitive dependencies listed)

Orders

-1NF: Yes (all columns are atomic)

-2NF: Yes (no partial dependencies)

-3NF: Yes (zero transitive dependencies listed)

Book Genres

-1NF: Yes (all columns are atomic)

-2NF: Yes (no partial dependencies; both the book_id and genre_id are needed to identify a record)

-3NF: Yes (zero transitive dependencies listed)

Order Details

-1NF: Yes (all columns are atomic)

-2NF: Yes (no partial dependencies and all non-key attributes depend on order_id and book_id composite keys)

-3NF: Yes (zero transitive dependencies listed)

-- BENEFITS OF USING DATABASE IN A SET OF DATA --

Databases provide a structured way to store and manage data, keeping all relevant information (like books, authors, customers, and order details) is organized in each of the related tables. This makes it much faster to find what you need! Not only that, but the consistency and scalability is important and can be maintained easier using these tables (especially if the store grows in popularity or size). Lastly, being able to back-up and

recovery data is very important in item management in case of failures or crashes.