Overview of Books Database with Practice Questions (and Answers)

Contents

[Overview of Books Database with Practice Questions (and Answers) 1](#_Toc162862474)

[Overview 2](#_Toc162862475)

[RDBMS Software Products and different versions of SQL 3](#_Toc162862476)

[The following are some popular SQL products: 3](#_Toc162862477)

[Web resources for learning/using SQL 3](#_Toc162862478)

[Descriptions of each table 4](#_Toc162862479)

[The Data 5](#_Toc162862480)

[What are “primary” keys? What are “foreign” keys? 6](#_Toc162862481)

[Intro to “Entity Relationship Diagrams” (ERDs) 6](#_Toc162862482)

["Crows Foot" format 7](#_Toc162862483)

[Microsoft format 7](#_Toc162862484)

[“One To Many” relationships vs “Many To Many” relationships 8](#_Toc162862485)

[“One to Many” relationships 8](#_Toc162862486)

[“Many to Many” relationships 8](#_Toc162862487)

[A 1 to many relationship requires 2 tables; A many to many relationship requires 3 tables. 9](#_Toc162862488)

[Other columns in the title\_authors table 9](#_Toc162862489)

[One to Many relationships are highlighted on the ERD 10](#_Toc162862490)

[What are the symbols on the lines connecting the tables? 10](#_Toc162862491)

[Maximum Cardinalities and Minimum cardinalities. 10](#_Toc162862492)

[Maximum cardinalities 10](#_Toc162862493)

[Minimum cardinalities 11](#_Toc162862494)

[Questions 12](#_Toc162862495)

[Single Table queries 12](#_Toc162862496)

[Single table queries with aggregate functions but no “group by” (i.e. these will return EXACTLY one row for each query). 12](#_Toc162862497)

[Single table queries with calculated values 12](#_Toc162862498)

[Single table queries with “group by”. These can use aggregate functions but will return at most one row in the output for each “group” of rows as defined in the “group by” 12](#_Toc162862499)

[Multiple table queries with group by 13](#_Toc162862500)

[Multiple table queries 13](#_Toc162862501)

[Subqueries, outer joins, unions and other stuff … 14](#_Toc162862502)

[*ANSWERS* 16](#_Toc162862503)

[*Single Table queries* 16](#_Toc162862504)

[*Single table queries with aggregate functions but no “group by” (i.e. these will return EXACTLY one row for each query).* 17](#_Toc162862505)

[*Single table queries with calculated values* 17](#_Toc162862506)

[*Single table queries with “group by”. These can use aggregate functions but will return at most one row in the output for each “group” of rows as defined in the “group by”* 18](#_Toc162862507)

[*Multiple table queries with group by* 18](#_Toc162862508)

[*Multiple table queries* 19](#_Toc162862509)

[*Subqueries, outer joins, unions and other stuff …* 23](#_Toc162862510)

[Other examples - How to figure out many to many vs one to many 27](#_Toc162862511)

# Overview

A "relational database" is a collection of one or more "tables" that contain data about a particular topic. Each "table" contains rows and columns of data.

This file contains a description of a "relational database" that contains information about books. The data is spread among five different tables where each table contains a specific subset of the data. The tables are: "titles", "authors", "publishers", "royalties" and "title\_authors". The sections below describe this database in much more detail.

SQL (i.e. Structured Query Language) is a standard programming language used for manipulating the contents of "relational databases". Using SQL commands, you can retrieve data from a database in all sorts of custom arrangements (see more in the examples below).

At the end of this file are numerous questions that ask you to write SQL code to retrieve data from the books database in a very specific ways.

NOTE:
The terminology used to describe relational databases can be confusing because different books and tutorials refer to the same concepts by different names. For example, "tables" are sometimes alternatively called "entities" or "objects". The names of the "columns" in a table are sometimes referred to as "fields" or "attributes" of the table.

# RDBMS Software Products and different versions of SQL

The software that is used for managing a relational database is known as a “Relational DataBase Management System” or and RDBMS. There are many different competing RDBMS products from different manufacturers. All of these products are similar. You can use the SQL language with all of these products.

In order to increase the compatibility between different SQL products, the American National Standards Institute publishes a SQL Standard (i.e. the ANSI SQL Standard). This standard undergoes revisions every few years. The SQL

The basic SQL syntax is very similar for all SQL products. The SQL used with most products deviates a little from the SQL standard. Alternatively, many SQL products have add-ons to the language that are not (yet) part of the standard. This allows the products to differentiate themselves from the competition. Once you learn one “flavor” of SQL it is very easy to pick up any other “flavor” of SQL. However, it is helpful when learning SQL to know where to look for the official documentation for the flavor of SQL that you will be learning.

## The following are some popular SQL products:

**SQLite**: A lightweight, self-contained SQL database engine that is widely used small to medium-sized web applications for its simplicity and ease of integration.

**MySQL**: Very popular open source product. Widely used for web applications.

**MariaDB**: A fork of MySQL, designed to maintain compatibility with MySQL, while also focusing on innovation and performance improvements. It's widely adopted by those looking for an alternative to MySQL.

**PostgreSQL**: An open-source, object-relational database system support for advanced data types.

**Oracle Database**: Known for its robust feature set, scalability, and performance. A a popular choice for large enterprises and critical applications.

**Microsoft SQL Server:** Widely used in enterprise environments, especially those that rely on other Microsoft technologies.

**IBM Db2**: Used in critical enterprise environments, offering advanced data management and analytics capabilities.

There are many many others …

# Web resources for learning/using SQL

General SQL

* <https://www.db-fiddle.com/> - create an use SQL commands – can specify which SQL flavor you want to use
* <https://www.w3schools.com/sql/> - general SQL tutorial – can practice right online
* <https://sqlzoo.net/wiki/SQL_Tutorial> - general SQL tutorial – can practice right online

SQLite

* <https://www.sqlitetutorial.net/> - Overview of SQLite flavor of SQL
* <https://sqliteviewer.app/> and <https://alpha.sqliteviewer.app/> - view contents of SQLite files – can’t use actual SQL commands though

MySQL

* <https://www.mysqltutorial.org/> - overview of MySQL flavor of SQL
* <https://www.w3schools.com/MySQL> - tutorial for MySQL flavor of SQL

# Descriptions of each table

The following shows the names of the tables and columns (AKA fileds) in each table.
For each column, the name of the column is shown as well as the type of data in the column and a description of the data.
The actual data in the tables is shown in the next section.

|  |  |  |
| --- | --- | --- |
| **authors table:** stores one row for each author. |  | **publishers table:** stores one row for each publisher. |
| **titles table:** stores one row for each book (AKA title).**royalties table:** This table the information about royalties that the publisher will pay to the author(s) of a book. "advance" is the fixed amount that the publisher pays the author when (s)he starts to write the book. "royalty\_rate" is a decimal value that contains the percent of each book's sale price that is paid to the author(s). If there is more than one author then the royalties and advaces are split between them |  | **title\_authors table:** Each row in this table represents a relationship between an author and a book that (s)he wrote. If a book has 3 authors, there will be 3 rows in this table for that book. This table has a ***composite primary key*** *(title\_id, au\_id).* The "royalty\_share" field represents the percent of the royalties that this author receives. If there is only one author for a book then this will be 1. If there are two authors for the book, the "royalty\_share" could be the same for both or it could be different (e.g. 0.6 and 0.4). |
|  |  |  |

# The Data

Authors:


Titles:



Publishers:



|  |
| --- |
| Title\_Authors:Royalties: |

|  |
| --- |
|  |

# What are “primary” keys? What are “foreign” keys?

A “***primary key***” of a table is a set of 1 or more columns whose contents are guaranteed to be unique for every row that is currently in the table or will ever by added to the table in the future. For example, the “title\_id” column is a “primary key” for the titles table. The values in the title\_id column are unique for every row in the titles table. If in the future we add any new rows to the titles table, we will need to come up with a new title\_id for that new title. Similarly, the “pub\_id” column is a primary key for publishers table. Each of these primary keys contain just one column of information. We’ll see below, when we discuss the “title\_authors” table that a primary key can be composed of more than one column. Note that a primary key column may NOT contain blank (i.e. NULL – to be described more later) values. Notice that the pub\_id column in the titles table is NOT a primary key. This is obvious because there are multiple rows in the titles table that contain the same pub\_id value. Rather the pub\_id column in the titles table is known as a “foreign key” which will be explained next …

A ***foreign key*** is a set of 1 or more columns in a table that stores the values of a primary key from a different table. Again, an example will help to explain this. Again, notice that the pub\_id column in the titles table is not a primary key. This is obvious because there are multiple rows in the titles table that contain the same pub\_id value. The pub\_id column in the titles table identifies the publisher for a particular title. To investigate the details about that publisher (e.g.the name of the publisher and the city they are located in) you need to find the row in the publishers table whose pub\_id matches. For example, the row in the titles table for T02, i.e. “200 Years of German Humor” shows that this title was published by publisher P03. To see the name and address of that publisher you need to find the row for P03 in the publishers table. By doing that we see that the book “200 Years of German Humor” was published by “Schandenfreude Press” which is located in Hamburg Germany.

Matching-up a “foreign key” in one table to a “primary key” in another table is fundamental to how “relational databases” are designed. We say that the titles table and the publishers table are “related” to each other. It is helpful to see these relationships visually. Information about these “relationships” are diagrammed on an “Entity Relationship Diagram” or ERD.

# Intro to “Entity Relationship Diagrams” (ERDs)

The structure of the tables and the details of these relationships are captured in diagrams known as an "Entity Relationship Diagrams" (or ERD) (a "table" is also known as an "entity"). Entity Relationship Diagrams come in several different formats. Some formats are more popular than others. Each format basically describes the same information. However, some formats include slightly more information than others. Below are two different formats of Entity Relationship Diagrams (ERD) for the "books database".

The 1st diagram below is arguably the most popular format for ERDs. It is known as a "crows foot ERD” (or just “crows foot diagram”). The name "crows foot" comes from the symbol (which looks like the foot of a bird) which often appears on this type of diagram. Every symbol that appears on these diagrams has a meaning and should be understood.

The 2nd diagram below shows the ERD in a format that was made popular by Microsoft software. It is very similar to the crows foot ERD. We will not discuss it further in this document.

The “crows foot format” and the Microsoft format of the diagram are only two of several different forms of "Entity Relationship Diagrams". There are other variations on this type of diagram. See this page for an overview of various other formats for ERD's: <https://www.gleek.io/blog/er-symbols-notations>

An Entity Relationship Diagram (ERD) is in essence a map of the database tables and is extremely useful in trying to understand how the tables in a database are related. However, the actual rows of data are NOT shown in the diagram. In order to really understand what is contained in a database it is best to analyze BOTH the ERD and a sampling of the actual data for the different tables.

Each "box" in an "Entity Relationshiop Diagram (ERD)", represents one of the tables in the database. The name of the table appears at the top of the box. Below that are the names of the columns in the table. A primary key field is shown with “PK” next to it and a “foreign key” field is shown with “FK” or “FK*n*” next to it (where *n* is a number). We’ll discuss later why FK is followed by a number but PK is not. A line connecting two tables demonstrates that the two tables are related to each other via a primary key (PK) in one table matching up with a foreign key (FK) in the other table.

Take some time to examine the “crows foot ERD” for the books database. The lines connecting the tables have different symbols at the ends of the lines. These symbols have important meanings which we will discuss in the next section.

See these pages for more information about the crows foot notation:

<https://vertabelo.com/blog/crow-s-foot-notation/>

<https://www.freecodecamp.org/news/crows-foot-notation-relationship-symbols-and-how-to-read-diagrams/>

<https://mermaid.js.org/syntax/entityRelationshipDiagram.html>

##

## "Crows Foot" format

PK stands for "primary key". A table may have only one primary key. The title\_authors table contains two PK fields, but these are both part of a single “composite” primary key. The primary key for a table that contains two or more PK fields is known as a "composite primary key".

FK stands for “foreign key”. A “foreign key” is a column or a set of columns that stores the primary key values from a different table. A table MAY contain more than one foreign key. FK1, FK2, etc. identify different foreign keys in the same table.

The lines between the tables and the symbols on those lines will be explained later.



## Microsoft format



**Crows Foot format *(this version was generated by the "mermaid" charting language - see https://mermaid.js.org/syntax/entityRelationshipDiagram.html)***



# “One To Many” relationships vs “Many To Many” relationships

##  “One to Many” relationships

In order to really understand the structure of a relational database it is important to understand more about the details of the relationships between the items that are represented in the database. Let’s take a moment and not focus on the technology of databases. Rather, let’s just focus on how the world in general works. For example, let’s analyze the relationship between titles and publishers. I think that most people who understand how the publishing industry works would agree to the following:

* *A specific ti*tle can only be published by **one** publisher.
For example, the book , “Harry Potter and the Sorcerer’s Stone” was published in the USA by “Scholastic Corporation”.
Similarly in our database, the title T02, “200 Years of German Humor” is published by P03, “Schadenfreude Press”.
* *A specific publisher* can publish **many** different titles.
For example “Scholastic Corporation” has published many books over the years.
Similarly, in our database, Schadenfreude Press, in addition to publishing T02, “200 Years of German Humor” also published T07, “I Blame My Mother” and T13, “What are the Civilian Applications?”

In technical terms we say that there is a “**one** to **many**” relationship between publishers and titles. This relationship has nothing to do with databases. It is just a fact of life based on how titles and publishers are related to each other in the world.

In order to represent this fact in our database you must realize that each row in the titles table stores information about a single title and each row in the publishers table has information about a single publisher. We design this “one to many” relationship into the structure of the tables by putting the pub\_id, which is the “primary key” of the publishers table, as one of the columns in the titles table (i.e. pub\_id becomes a “foreign key” in the titles table). This allows for many different rows in the titles table to all list P03 as the publisher. However, because there is only one pub\_id for each row in the titles table, a specific title cannot relate to more than one publisher.

## “Many to Many” relationships

When it comes to authors and titles things are a little different. Some books are written in partnership by more than one author. This is often the case for textbooks. Similarly it is usually the case that authors write more than one book. We say that authors and titles have a “**many** to **many**” relationship since *one author* can write **many** books and *one title* can be written by **many** authors. To summarize:

* *A specific author* can write **many** titles
* *A specific title* can be written by more than one, i.e. “**many**” authors

We say that there is a “**many** to **many**” relationship between authors and titles.

To represent the many-many relationship between authors and titles, in addition to the “authors” table and the “titles” table, we also make use of a third table, the “title\_authors” table. Notice that there is no mention in the authors table about which titles an author wrote nor is there any mention in the titles table about who wrote a particular title. The information about which author wrote which title is instead located in the title\_authors table. Each row in the title\_authors table records the fact that a particular author had a part in writing a particular title. If a title was written in collaboration by 3 authors, then there will be 3 rows in the title\_authors table for that title.

This all becomes clear if you analyze the data for the titles, title\_authors and authors tables. For example, T11 was written in collaboration by three different authors, A03, A04 and A06. Therefore there are 3 different rows in the title\_authors table, one for T11,A03, another row for T11,A04 and a third row for T11,A06.. If you want to know the title\_name or number of pages for T11 you can find that information in the row in the titles table for T11. If you want to know the names or any other personal info about the authors, you need to find the rows in the authors table for A03, A04 and A06.

Similarly, if a particular author wrote more than one title, there will be a separate row in the title\_authors table for each title that the author wrote. Each of those rows will record the author’s au\_id paired with the title\_id of one of the books he/she wrote. For example author, A02 wrote or had a part in writing each of the titles T06, T07, T10 and T12. This is represented by four different rows in the title\_authors table. One row for A02,T06, another row for A02,T07, a third row for A02,T10 and a fourth row for A02,T12. If you want to know more details about who A02 is, look in the authors table row for A02. To learn more details about the titles T06, T07, T10 or T12, look in the corresponding row in the titles table.

# A "one to many" relationship requires 2 tables; A "many to many" relationship requires 3 tables.

As we saw above with the publishers and titles tables, a 1 to many relationship is always created between 2 tables with a primary-key/foreign-key pair. More specifically, the primary key from the table for the 1 side of the relationship is stored as a foreign key in the table for the many side of the relationship. In the specific case we just discussed, the primary key for the publishers table (the 1 side of the relationship) is placed into the titles table (the many side of the relationship) as a foreign key.

As we saw above with the authors and titles tables, a many to many relationship is implemented by setting up a 3rd table. The 3rd table contains two foreign keys. One foreign key for each of the tables in the many to many relationship. In our case, the title\_authors table contains a foreign key to the titles table and another foreign key to the authors table. The two foreign key columns together become a “composite primary key” of the “middle” table.

### Other columns in the title\_authors table

As we’ve mentioned, each row in the titles\_authors table records information about the “relationship” between a particular title and a particular author. The most important part of that relationship is simply to know that a particular author wrote a particular title. However, anything else about the details relating to the association of a particular author with a particular title will also be recorded in the title\_authors table. Specifically, for titles that were written in collaboration by more than one author, the values in the au\_order column identify which author appears first, second, third, etc. on the title page. Similarly for titles written by more than one author, the royalty\_share shows the percent of the royalties that each author gets for that title.

For example, the following three rows appear in title\_authors table:

|  |  |  |  |
| --- | --- | --- | --- |
| title\_id | au\_id | au\_order | royalty\_share |
| T11 | A03 | 2 | 0.30 |
| T11 | A04 | 3 | 0.30 |
| T11 | A06 | 1 | 0.40 |

These rows indicate that

* T11 was written by A03,A04 and A06.
* A06 is listed first on the title page and receives 40% of the royalties.
* A03 and A04 are listed 2nd and 3rd on the title page and each receives only 30% of the royalties.

# One to Many relationships are highlighted on the ERD

## What are the symbols on the lines connecting the tables?

A “one to many” relationship is highlighted with special symbols on the Entity Relationship Diagram. For example, let’s examine the line on the ERD that connects the publishers table and the titles table. It is copied below for reference.

|  |  |
| --- | --- |
| **ERD for publishers and titles** | **Zoom in on the line** |
|  | A diagram of a crow foot  Description automatically generated |

Let’s start by clearly pointing out where the symbols are and what they look like. Later, after we recognize what the symbols look like and where they are on the diagram we’ll discuss what these symbols actually mean. First of all, you should recognize that there are actually 4 symbols on this line.

Two of the 4 symbols are very close to the publishers table. **(a)** The symbol that is closest to the publishers table looks like a number “1”. **(b)** Immediately to the right of the “1” symbol is a circle that represents the number “0”. The other two symbols are very close to the titles table. **(c)** The symbol that is touching the titles table looks like a “crows foot”– notice that there are 3 lines that make up the crows foot (i.e. there are “many” lines). **(d)** Immediately to the left of the crow’s foot symbol is another circle that also represents the number zero.

We refer to the symbols that are closest to the table’s “boxes” as the outer symbols and the two symbols that are next to the outer symbols as the “inner” symbols on the line. To summarize, there are 4 symbols on this line. They are:

* Two “outer symbols” on the line:
	+ Closest to the publishers table – a vertical line representing the number 1
	+ Closest to the titles table - crows foot representing “many”
	+ These represent the “maximum cardinalities” (see below for more info)
* Two “inner symbols” on the line:
	+ closer to the publishers table – a circle representing the number 0
	+ closer to the titles table – another circle, also representing the number 0
	+ These represent the “minimum cardinalities” (see below for more info)

Now we’ll discuss the meanings of these symbols

## Maximum Cardinalities and Minimum cardinalities.

### Maximum cardinalities

As you might have already guessed, the “outer symbols” represent the “1 to many” relationship between the publishers and titles tables. These are known as the “**maximum cardinality symbols**”. These symbols represent the following two facts:

* The “1” symbol next to the publisher’s table means the following: Given a specific title, that title can only be published by ONE publisher. When thinking about this it is helpful think about one specific title in your mind – e.g. “200 years of German Humor”. Then ask yourself, in theory, what is the MAXIMUM number of publishers that can publish that title.
* The “crows foot” symbol next to the titles table means the following: Given a specific publisher, that publisher in theory can publish more than one title (this should be obvious to anyone who understands how book publishing works). Again it helps to think about a specific publisher – e.g. “AAA Press” and ask yourself – in theory – what is the maximum number of titles that this publisher could publish.

In summary, the maximum cardinalities between two tables represent the answers to the following two questions

* if given a random single row from the 1st table, what are the MAXIMUM number of rows that can be related to it in the 2nd table. The answer can only be “one row” or “more than one row (i.e. many rows)”.
* if given a random single row from the 2nd table, what are the MAXIMUM number of rows that can be related to it in the 1st table. Again, the answer can only be “one row” or “more than one row (i.e. many rows)”

### Minimum cardinalities

The two “inner” symbols on the line that connect two tables represent the “MINIUMUM cardinalities”.

The minimum cardinalities between two tables represent the answers to the following two questions

* if given a random single row from the 1st table, what are the MINIMUM number of rows that can be related to it in the 2nd table. The answer may only be “zero rows” or “one row”.
* if given a random single row from the 2nd table, what are the MINIMUM number of rows that can be related to it in the 1st table. Again, the answer can only be “zero rows” or “one row”.

Our ERD shows that it is possible for a title to exist in the titles table that has not been published yet. Perhaps this is a work in progress. The ERD shows that our database allows for such titles to exist in the database. Similarly our ERD shows that it is possible for a publisher to exist in the publishers table who did not publish any titles yet. For example, this may be a new publishing business that is just getting started and hasn’t released any titles yet.

Minimum cardinality rules are often choices that are made by the database designer. We could have alternatively decided that we don’t want to allow titles in the database unless they have been published. In that case the minimum carinality symbol next to the publishers table would be “1” and not “0”. Similarly we could have decided that we will not enter any publishers unless they have published at least one title. In that case the minimum cardinality symbol next to the titles table would be a “1” and not a “0”.

In general, “minimum cardinalies” are different from “maximum cardinalities” in the following way. Maximum cardinality rules usually reflect natural rules based on how the world works. For example, just about everyone in the world would agree that publishers can publish more than one title and that titles are only published by one publisher. These rules naturally are enforced by the choice of which foreign keys should be put in which tables. However, minimum cardinalities can be chosen by a database designer based on the rules of a particular business (i.e. the business rules). There is nothing about PKs and FKs that imply the minimum cardinalities. The minimum cardinality rules can be often enforced in the database using other mechanisms but that goes beyond the scope of what we will cover in this document. However, the ERD clearly shows the minimum cardinality rules.

# Questions

Create the following queries for the “books” database. For each query make sure to only include those tables that are necessary to answer the question. NOTE: the answers shown below are just some possible answers. For some questions there may be more than one possible answer.

## Single Table queries

1. List all books (title and number sold) that sold more than 1000 copies. List the books with the most sales at the top.
2. List all authors who are live either in NY or CA and whose last name begins with a "K".
3. List the first and last names of all authors whose last name starts with a letter from A through J
(HINT: In the where clause, make sure that the first letter of the last name is both >="A" and also <"K")
(HINT: another possible solution is to use the LIKE several times - once for A, once for B, once for C, etc up to J. Each LIKE should be separated from the others by OR's)
4. For each author, show their first initial, followed by a period and a space, followed by their last name. In the 2nd column show the author's state. Show the column heading for the first column as 'AuthorName'. Order the results in alphabetical order based on the full name of the person.
5. Show the titles of all books. Also show the length of the title, i.e. how many characters, including spaces and punctuation. Display the 2nd column with the name 'TitleLength'. Sort the output so that the shortest titles are listed first. If two titles are the same length then sort those titles alphabetically.

## Single table queries with aggregate functions but no “group by” (i.e. these will return EXACTLY one row for each query).

1. List the average price of all books.
2. List the average price of history books.
3. List the number of pages in the longest and shortest books (don't list the actual title of the book).

## Single table queries with calculated values

1. List the title\_name and total revenue for each book. (Revenue for a book is the number sold times the price of the book.)
2. List title of each book and the sale price for the book where the sale price is 10% off of the original price.

## Single table queries with “group by”. These can use aggregate functions but will return at most one row in the output for each “group” of rows as defined in the “group by”

1. For each "type" of book (e.g. biography, children, etc) list the number of pages in the shortest book of that type and the number of pages in the longest book of that type. Sort the results alphabetically by the type of book.
2. List the types of books (e.g. history, biography, etc) and the average price of those books for which the average price of books in that category is at least $12.00.
3. For each type of book (e.g. biography, children, etc), show the number of those books that are 450 pages or longer. The column heading should be '# of long books'. If there are no titles of a particular type that are so long, then don't show that type at all.
4. Modify the answer to the previous question so that only those types for which there are at least 2 long books are displayed.

## Multiple table queries with group by

1. For each publisher, list the name of the publisher and total number of pages that the publisher has published in all their books (ie. add up all the pages in all books for each publisher). Show the publishers who have published the most pages at the top of the result list.
2. Modify the previous query so that only publishers who have published at least 1250 pages will show up.
3. List each publisher's name and the numbers of pages in their longest and shortest books. Sort the results by the publisher's name.

## Multiple table queries

1. List the authors first and last names, the author's state, the title name, the publisher name and the publishers state for all books.
2. List the title, number of pages, authors first and last names and author's state for all books that are longer than 150 pages and whose author lives in NY or CA.
3. List the authors first and last names, the author's state, the title name, the publisher name and the publishers state only for those books where the author and publisher are from the same state. (HINT: specify authors.state [note: there is a period between author and state] in the criteria for the publisher's state).
4. List the title and authors of all books published by "Abatis Publishers" sort the results in alphabetical order by title and then by the au\_order column. Do NOT show that au\_order column in the output.
5. List the publisher name, author name, and title name for all books that have at least 100 pages and that sold at least 200 copies. Sort the results in alphabetical order first by publisher and then by author.
6. List authors first and last names and the titles they wrote for publishers whose name (ie. the publisher's name) begins with "A", "B" or "C"
7. List the publisher name, author's name and phone number for all authors that have written books for the publisher's whose name begins with a letter from A through J. (HINT: specify the publisher's name twice but only show one of them. In the criteria for one of them list >="A" and in the criteria for the other list <"K").
8. List the titles of all books and the number of authors for the book (NOT the actual author names). Sort the results so that the books with the most authors appear on top. All books that have the same number of authors should be listed in alphabetical order.
9. List the titles of all books that have 2 or more authors and that are longer than 50 pages.
10. List all publishers and the number of books that the publisher has published (not the actual titles, just the number of books). Sort the list so that the publisher with the most books appears on top.
11. List the publisher name and total revenue for each for each publisher. (Total revenue for a publisher is the total revenue for all books published by that publisher. Revenue for a book is the number sold times the price of the book.)
12. Modify the previous query to only list those publishers who have at least ten thousand dollars in total revenue.
13. List the title and author of each book. For each author for a particular book, list the amount of the "advance" that (s)he got for that book.
14. List the title and author of each book. For each author for a particular book, list the total amount in royalties that (s)he got for that book.
Hint: the royalty dollars for a particular book is the "price" times the "sales" times the "royalty\_rate" times the "royalty\_share". Sort the results so that the authors who made the most royalty dollars are listed at the top. (NOTE: make sure to read the database description at the top of this document to understand the fields ROYALTIES.ROYALTY\_RATE and the field TITLE\_AUTHORS.ROYALTY\_SHARE).
15. List the name of each author and the total amount of royalty dollars that they received for all of their books (name this column “RoyaltyDollars”). Hint: the royalty dollars for a particular book is the "price" times the "sales" times the "royalty\_rate" times the "royalty\_share". Sort the results so that the authors who made the most royalty dollars are listed at the top.
16. Modify the query from the previous question so that the list only includes authors who made more than $50,000.00 in royalties.

## Subqueries, outer joins, unions and other stuff …

1. Show the title names and number of pages for those books that are longer than (ie. more pages) the average length of all books.
2. Show the publisher's name and number of authors who have written books for that publisher:
3. Show the pub\_name and the number of titles published by that publisher. Only include publishers for whom at least 5 authors have worked.
4. Show the names of publishers who did not publish any books (I guess they are just getting started ☺ )
5. List the name of each publisher and the total number of books that each publisher has published. Sort the results so that the publishers who published the most books appear at the top. If two or more publishers published the same number of books then they should be listed in alphabetical order.
	1. The answer should only include publisher who have published some books.
	2. This time make sure to also include publishers who have published zero books

**Additional questions**

1. Show the names of the authors and the publishers who published their books.
Only show those authors/publishers where the author lives in the same state as the publisher who published the book.
Do not show any author/pbulisher names twice.
2. The exchange rate for NIS/USD (i.e. New Israeli Shekel / US Dollar) is 3.3.
Write a query that shows the titles and prices for biographies that cost between 10 and 20 dollars.
Show two columns for the prices. One column for USD and one column for NIS.
Name the columns "price in USD" and "price in NIS".
Sort the results so that the most expensive books are displayed first.
3. Show those titles for which the price of the book is greater than the average price of all books and the number of pages of the book is less than the average number of pages for all books.
4. Show the authors who have not written any books
5. In each row show an author's name and the number of books that author wrote.
Only show those authors who wrote at least two books.
Sort the results alphabetically (last name, first name).
6. Same as previous question. However, this time, only show those authors who wrote biographies, the count of the number of biographies they wrote and only show those authors who wrote at least 2 biographies.
7. Suppose this database is being used by a literary agent who matches up publishers and authors for future projects.
The agent is trying to figure out the authors he works with who might be a good match for the publishers he works with.
The agent thinks that authors who write books that sell for an above average price are good matches for publishers who publish books that sell for an above average price.
The agent wants to see a list of the potential match-ups between such authors and such publishers.
Write a query that does the following:

	* Each row of your output should show the name of an author and a publisher.
	The author may have, or may not have, already written a book for that publisher.
	* The authors should be those authors whose average book price for books they wrote is above the average price of all books in the database.
	* The publishers should be those publishers whose average book price for books they published is above the average price of all books in the database.
8. Same as above, except that the authors should be those authors who have written at least one book for which the price is above the average price of books of the same type.

# *ANSWERS*

## *Single Table queries*

1. ***List all books (title and number sold) that sold more than 1000 copies. List the books with the most sales at the top.

 ANSWER:
 select title\_name, sales
 from titles
 where sales > 1000
 order by sales desc;***
2. ***List all authors who are live either in NY or CA and whose last name begins with a "K".

ANSWER:
 select au\_fname, au\_lname
 from authors
 where (state = 'NY' or state='CA') and substr(au\_lname,1,1) = 'K'; -- (parentheses are required
 -- since AND is done before OR by default)***
3. ***List the first and last names of all authors whose last name starts with a letter from A through J
(HINT: In the where clause, make sure that the first letter of the last name is both >="A" and also <"K")
(HINT: another possible solution is to use the LIKE several times - once for A, once for B, once for C, etc up to J. Each LIKE should be separated from the others by OR's)

 ANSWER:
 select au\_fname, au\_lname
 from authors
 where substr(au\_lname,1,1) >= 'A' and substr(au\_lname,1,1) < 'K';

 ANOTHER ANSWER
 select au\_fname, au\_lname
 from authors
 where au\_lname like 'A%' or au\_lname like 'B%' or au\_lname like 'C%' or
 au\_lname like 'D%' or au\_lname like 'E%' or au\_lname like 'F%' or
 au\_lname like 'G%' or au\_lname like 'H%' or au\_lname like 'I%' or
 au\_lname like 'J%'***
4. ***For each author, show their first initial, followed by a period and a space, followed by their last name. In the 2nd column show the author's state. Show the column heading for the first column as 'AuthorName'. Order the results in alphabetical order based on the full name of the person.

 ANSWER:
 select substr(au\_fname,1,1) || '. ' || au\_lname as 'AuthorName', state
 from authors
 order by au\_lname, au\_fname;***
5. ***Show the titles of all books. Also show the length of the title, i.e. how many characters, including spaces and punctuation. Display the 2nd column with the name 'TitleLength'. Sort the output so that the shortest titles are listed first. If two titles are the same length then sort those titles alphabetically.

 ANSWER
 select title\_name, length(title\_name) as 'TitleLength'
 from titles
 order by length(title\_name), title\_name;***

## *Single table queries with aggregate functions but no “group by” (i.e. these will return EXACTLY one row for each query).*

1. ***List the average price of all books.

 ANSWER:
 select avg(price)
 from titles;***
2. ***List the average price of history books.

 ANSWER:
 select avg(price)
 from titles
 where type = 'history';***
3. ***List the number of pages in the longest and shortest books (don't list the actual title of the book).

 ANSWER:
 select min(pages) , max(pages) from titles;***

## *Single table queries with calculated values*

1. ***List the title\_name and total revenue for each book. (Revenue for a book is the number sold times the price of the book.)

 ANSWER:
 select title\_name, sales \* price as 'Total Revenue'
 from titles;***
2. ***List title of each book and the sale price for the book where the sale price is 10% off of the original price.

 ANSWER:
 select title\_name, price \* 0.90 as 'Sale Price'
 from titles;***

## *Single table queries with “group by”. These can use aggregate functions but will return at most one row in the output for each “group” of rows as defined in the “group by”*

1. ***For each "type" of book (e.g. biography, children, etc) list the number of pages in the shortest book of that type and the number of pages in the longest book of that type. Sort the results alphabetically by the type of book.***

 ***ANSWER:
 select type, min(pages) AS 'Length of Shortest' , max(pages) as 'Length of Longest'
 from titles
 group by type
 order by type;***

1. ***List the types of books (e.g. history, biography, etc) and the average price of those books for which the average price of books in that category is at least $12.00.

 ANSWER:
 select type, avg(price)
 from titles
 group by type
 having avg(price) >= 12;***
2. ***For each type of book (e.g. biography, children, etc), show the number of those books that are 450 pages or longer. The column heading should be '# of long books'. If there are no titles of a particular type that are so long, then don't show that type at all.

 ANSWER
 select type, count(\*) as '# of long books'
 from titles
 where pages >= 450
 group by type;***
3. ***Modify the answer to the previous question so that only those types for which there are at least 2 long books are displayed.

ANSWER - NOTE: the only change to the above answer is the addition of the HAVING clause at the end.

 select type, count(\*) as '# of long books'
 from titles
 where pages >= 450
 group by type
 having count(\*) >= 2; -- this is the only change.***

## *Multiple table queries with group by*

1. ***For each publisher, list the name of the publisher and total number of pages that the publisher has published in all their books (ie. add up all the pages in all books for each publisher). Show the publishers who have published the most pages at the top of the result list.

 ANSWER
 select pub\_name, sum(pages)
 from publishers join titles on publishers.pub\_id = titles.pub\_id
 group by pub\_name
 order by sum(pages) desc;***
2. ***Modify the previous query so that only publishers who have published at least 1250 pages will show up.

 ANSWER
 select … -- same as previous question
 from … -- same as previous question
 group … -- same as previous question***

 ***having sum(pages) >= 1250 -- THIS LINE IS THE ONLY CHANGE
 order … -- same as previous question***

1. ***List each publisher's name and the numbers of pages in their longest and shortest books. Sort the results by the publisher's name.

 ANSWER:
 select pub\_name, min(pages) AS 'Length of Shortest' , max(pages) as 'Length of Longest'
 from titles join publishers on titles.pub\_id = publishers.pub\_id
 group by pub\_name
 order by pub\_name;***

## *Multiple table queries*

1. ***List the authors first and last names, the author's state, the title name, the publisher name and the publishers state for all books.

 ANSWER:
 select au\_fname, au\_lname, authors.state, title\_name, pub\_name, publishers.state
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join publishers on publishers.pub\_id = titles.pub\_id;***
2. ***List the title, number of pages, authors first and last names and author's state for all books that are longer than 150 pages and whose author lives in NY or CA.

 ANSWER:
 select title\_name, pages, au\_fname, au\_lname, authors.state
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 where pages > 150 and (state = 'NY' or state = 'CA') ; -- (parentheses ARE necessary since by default AND is done before OR)***
3. ***List the authors first and last names, the author's state, the title name, the publisher name and the publishers state only for those books where the author and publisher are from the same state. (HINT: specify authors.state [note: there is a period between author and state] in the criteria for the publisher's state).

 ANSWER:
 select au\_fname, au\_lname, authors.state, title\_name, pub\_name, publishers.state
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join publishers on publishers.pub\_id = titles.pub\_id
 where authors.state = publishers.state;***
4. ***List the title and authors of all books published by "Abatis Publishers" sort the results in alphabetical order by title and then by the au\_order column. Do NOT show that au\_order column in the output.

 ANSWER:
 select au\_fname, au\_lname, title\_name, pub\_name
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join publishers on publishers.pub\_id = titles.pub\_id
 where pub\_name = 'Abatis Publishers'
 order by title\_name, au\_order;***
5. ***List the publisher name, author name, and title name for all books that have at least 100 pages and that sold at least 200 copies. Sort the results in alphabetical order first by publisher and then by author.***

 ***ANSWER:
 select pub\_name, au\_fname, au\_lname, title\_name
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join publishers on publishers.pub\_id = titles.pub\_id
 where pages >= 100 and sales >= 200
 order by pub\_name, au\_lname, au\_fname;***

1. ***List authors first and last names and the titles they wrote for publishers whose name (ie. the publisher's name) begins with "A", "B" or "C"

ANSWER:
 select au\_fname, au\_lname, title\_name, pub\_name
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join publishers on publishers.pub\_id = titles.pub\_id
 where substr(pub\_name, 1,1) IN ('A','B','C');***

***ANOTHER ANSWER: Same as above but use the following WHERE clause instead:
 select … # same as above
 from …. # same as above
 where substr(pub\_name, 1,1) = 'A' or substr(pub\_name, 1,1) = 'B' or
 substr(pub\_name, 1,1) = 'C';***

***ANOTHER ANSWER: Same as above but use the following WHERE clause instead:

 select … # same as above
 from …. # same as above
 where pub\_name like 'A%' or pub\_name like 'B%' or pub\_name like 'C%';***

1. ***List the publisher name, author's name and phone number for all authors that have written books for the publisher's whose name begins with a letter from A through J. (HINT: specify the publisher's name twice but only show one of them. In the criteria for one of them list >="A" and in the criteria for the other list <"K").

 ANSWER:
 select pub\_name, au\_fname, au\_lname, phone
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join publishers on publishers.pub\_id = titles.pub\_id
 where substr(pub\_name,1,1) >= 'A' and substr(pub\_name,1,1) < 'K';***
2. ***List the titles of all books and the number of authors for the book (NOT the actual author names). Sort the results so that the books with the most authors appear on top. All books that have the same number of authors should be listed in alphabetical order.***

 ***ANSWER:
 select title\_name, count(\*) as 'Number of Authors'
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 group by titles.title\_id, title\_name
 order by count(\*) desc, title\_name;***

1. ***List the titles of all books that have 2 or more authors and that are longer than 50 pages.

 ANSWER:
 select title\_name, count(\*) as 'Number of Authors'
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 where pages >= 50
 group by titles.title\_id, title\_name
 having count(\*) >= 2
 order by count(\*) desc, title\_name;***
2. ***List all publishers and the number of books that the publisher has published (not the actual titles, just the number of books). Sort the list so that the publisher with the most books appears on top.

 ANSWER:
 select pub\_name, count(\*) as '# of Books'
 from titles join publishers on publishers.pub\_id = titles.pub\_id
 group by publishers.pub\_id, pub\_name
 order by count(\*) desc;***
3. ***List the publisher name and total revenue for each for each publisher. (Total revenue for a publisher is the total revenue for all books published by that publisher. Revenue for a book is the number sold times the price of the book.)***

 ***ANSWER:
 select pub\_name, sum(sales \* price) as 'Total Revenue'
 from titles join publishers on publishers.pub\_id = titles.pub\_id
 group by publishers.pub\_id, pub\_name ;***

1. ***Modify the previous query to only list those publishers who have at least ten thousand dollars in total revenue.

 ANSWER:
 select pub\_name, sum(sales \* price) as 'Total Revenue'
 from titles join publishers on publishers.pub\_id = titles.pub\_id
 group by publishers.pub\_id, pub\_name ;
 having sum(sales \* price) >= 10000;***
2. ***List the title and author of each book. For each author for a particular book, list the amount of the "advance" that (s)he got for that book.***

 ***ANSWER:
 select title\_name, au\_fname, au\_lname, advance
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join royalties on royalties.title\_id = titles.title\_id;***

1. ***List the title and author of each book. For each author for a particular book, list the total amount in royalties that (s)he got for that book.
Hint: the royalty dollars for a particular book is the "price" times the "sales" times the "royalty\_rate" times the "royalty\_share". Sort the results so that the authors who made the most royalty dollars are listed at the top. (NOTE: make sure to read the database description at the top of this document to understand the fields ROYALTIES.ROYALTY\_RATE and the field TITLE\_AUTHORS.ROYALTY\_SHARE).

 ANSWER:
 select title\_name, au\_fname, au\_lname, royalty\_share \* royalty\_rate \* sales \* price as 'Total Royalties for Author'
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join royalties on royalties.title\_id = titles.title\_id;***
2. ***List the name of each author and the total amount of royalty dollars that they received for all of their books (name this column “RoyaltyDollars”). Hint: the royalty dollars for a particular book is the "price" times the "sales" times the "royalty\_rate" times the "royalty\_share". Sort the results so that the authors who made the most royalty dollars are listed at the top.

 ANSWER:
 select au\_fname, au\_lname, sum(royalty\_share \* royalty\_rate \* sales \* price) as 'Total Royalties for Author'
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join royalties on royalties.title\_id = titles.title\_id
 group by authors.au\_id, au\_fname, au\_lname
 order by sum(royalty\_share \* royalty\_rate \* sales \* price) desc;***
3. ***Modify the query from the previous question so that the list only includes authors who made more than $50,000.00 in royalties.

 ANSWER:
 select au\_fname, au\_lname, sum(royalty\_share \* royalty\_rate \* sales \* price) as 'Total Royalties for Author'
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join royalties on royalties.title\_id = titles.title\_id
 group by authors.au\_id, au\_fname, au\_lname
 having sum(royalty\_share \* royalty\_rate \* sales \* price) > 50000
 order by sum(royalty\_share \* royalty\_rate \* sales \* price) desc;***

## *Subqueries, outer joins, unions and other stuff …*

1. ***Show the title names and number of pages for those books that are longer than (ie. more pages) the average length of all books.

ANSWER:
 select title\_name , pages
 from titles
 where pages >= (select avg(pages) from titles);***
2. ***Show the publisher's name and number of authors who have written books for that publisher:

ANSWER:
 select pub\_name, count(DISTINCT authors.au\_id) as '# of authors'
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join publishers on publishers.pub\_id = titles.pub\_id
 group by pub\_id, pub\_name;***
3. ***Show the pub\_name and the number of titles published by that publisher. Only include publishers for whom at least 5 authors have worked.

ANSWER:
 select pub\_name, count(\*) as 'Number of Titles'
 from publishers as pub1 join titles on pub1.pub\_id = titles.pub\_id
 group by pub1.pub\_id, pub\_name
 having 5 <= ( select count(DISTINCT authors.au\_id)
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join publishers as pub2 on pub2.pub\_id = titles.pub\_id
 where pub2.pub\_id = pub1.pub\_id ) ;***
4. ***Show the names of publishers who did not publish any books (I guess they are just getting started ☺ )

ANSWER – with a subquery

 select pub\_name
 from publishers as pub1
 where not exists
 ( select \*
 from publishers as pub2 join titles on pub2.pub\_id = titles.pub\_id
 where pub2.pub\_id = pub1.pub\_id );

ANSWER – with a left join
 select pub\_name from publishes left join titles on publishers.pub\_id = titles.pub\_id
 where title\_id is NULL;***
5. ***List the name of each publisher and the total number of books that each publisher has published. Sort the results so that the publishers who published the most books appear at the top. If two or more publishers published the same number of books then they should be listed in alphabetical order.***
	1. ***The answer should only include publisher who have published some books.

	 ANSWER:
	 select pub\_name, count(\*)
	 from titles join publishers on publishers.pub\_id = titles.pub\_id
	 group by pub\_name
	 order by count(\*) desc, pub\_name;***
	2. ***This time make sure to also include publishers who have published zero books***

***ANSWER:
 select pub\_name, count(titles.title\_id)
 from publishers left join titles on publishers.pub\_id = titles.pub\_id
 group by pub\_name
 order by count(\*) desc, pub\_name;

ANSWER – with a UNION – however the previous answer is shorter and more to the point***

 ***select pub\_name , 0 as NumTitles
 from publishes left join titles on publishers.pub\_id = titles.pub\_id
 where title\_id is NULL

UNION

 select pub\_name, count(\*) as NumTitles
 from titles join publishers on publishers.pub\_id = titles.pub\_id
 group by pub\_name;***

**Additional questions**

1. ***# Show the names of the authors and the publishers who published their books.
# Only show those authors/publishers where the author lives in the same state
# as the publisher who published the book.
# Do not show any author/pbulisher names twice.

# ANSWER

sqldf("
select distinct au\_fname, au\_lname, pub\_name, authors.state, publishers.state
from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
 join publishers on titles.pub\_id = publishers.pub\_id
where authors.state = publishers.state
 ")***
2. ***# QUESTION
#
# The exchange rate for nis/usd is 3.3
# Write a query that shows the titles and prices for biographies that cost between
# 10 and 20 dollars. Show two columns for the prices. One column for USD
# and one column for NIS. Name the columns "price in USD" and "price in NIS".
# Sort the results so that the most expensive books are displayed first.

# ANSWER

sqldf("
select title\_name, price \* 3.3 as 'price in NIS', price as 'price in USD', type
from titles
where type = 'biography' and
 price >= 10 and price <= 20
order by price desc
 ")***
3. ***# Show those titles for which the price of the book is greater than the
# average price of all books and the number of pages of the book is less
# than the average number of pages for all books.
sqldf("
select title\_name, price, pages
from titles
where price > (select avg(price) from titles) and
 pages < (select avg(pages) from titles)
 ")

sqldf("select avg(price) from titles")
sqldf("select avg(pages) from titles")***
4. ***# Show the authors who have not written any books
sqldf("
select au\_fname, au\_lname
from authors left join title\_authors on authors.au\_id = title\_authors.au\_id
where title\_authors.au\_id is null
")***
5. ***# In each row show an author's name and the number of books that author wrote.
# Only show those authors who wrote at least two books.
# Sort the results alphabetically (last name, first name).

sqldf("
select au\_fname, au\_lname, count(\*)
from authors join title\_authors on authors.au\_id = title\_authors.au\_id
group by au\_fname, au\_lname
having count(\*) >= 2
 ")***
6. ***# Same as previous question. However, this time, only show those authors
# who wrote biographies, the count of the number of biographies they wrote
# and only show those authors who wrote at least 2 biographies.
sqldf("
select au\_fname, au\_lname, count(\*)
from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 join titles on title\_authors.title\_id = titles.title\_id
where type = 'biography'
group by au\_fname, au\_lname
having count(\*) >= 2
")***
7. ***# Each row of your output should show the name of an author and a publisher.
# The author did not necessarily write a book for that publisher.
# The authors should be those authors whose average book price for books they
# published is above average.
# The publishers should be those publishers whose average book price for books
# they published is above average.***
8. ***# Show the names of the most prolific authors, i.e. the authors
# who have written the most books. Note that there could be
# several different authors who are "tied" for the most number of books
# written. Sort the names in alphabetical order (last name, first name)
# Also show the au\_id and the number of titles that the author wrote.

# This is a rather complex answer.
# This approach uses a subquery within a subquery.
# There may be other approaches too.
# There may be simpler answers too.
#
# To explain we will show the subqueries and what they produce.
# Then we will show the full query that includes the subquery.
#
# The following query is NOT the answer. It is simply a query that
# shows the number of books that were written by each author. This
# will be used as a subquery in the answer shown below.
sqldf("
 select authors.au\_id, count(\*) as numTitles
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 group by authors.au\_id, au\_fname, au\_lname
")

# We'd like to get the max number of titles that an author wrote.
# This amounts to getting the highest number from the numTitles column
# in the previous query. We can do that by using the above query
# as a subquery in the from clause.

sqldf("
 select max(numTitles2)
 from
 (select authors.au\_id, count(\*) as numTitles2
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 group by authors.au\_id, au\_fname, au\_lname)
 ")

# Finally we can put this all together to answer our question.
# The following is the FULL ANSWER to the original question.
# It uses the above query as a subquery in the having clause (see
# the code below). Note that in the following code there is a
# subquery inside of a subquery.
#
# FINAL ANSWER:

sqldf("
select authors.au\_id, au\_fname, au\_lname, count(\*) as numTitles
from authors join title\_authors on authors.au\_id = title\_authors.au\_id
group by authors.au\_id, au\_fname, au\_lname
having numTitles =
 (select max(numTitles2)
 from
 (select authors.au\_id, count(\*) as numTitles2
 from authors join title\_authors on authors.au\_id = title\_authors.au\_id
 group by authors.au\_id, au\_fname, au\_lname))
order by au\_lname, au\_fname
")***

END OF DOCUMENT

The rest of this document is “under construction”. Students can ignore this information. I kept it in the document so that I don’t “misplace” the info.

-Prof. Rosenthal

## Other examples - How to figure out many to many vs one to many

Each row in a relational database table represents one item. For example, suppose we have a database that stores information about the real estate owned by a university. The university owns many buildings. Each building has many rooms. The database might have one table that stores information about the buildings such as the addresses and the unique “code” for each building. There may be another table that stores information about rooms in those buildings. This table could store the room number, max number of people that can fit in each room, the type of room (e.g. dorm room, office, classroom, social hall, etc), whether the room has a audio visual system, floor in the building, etc.

In the books database, the titles T04, T07 and T11 were each written by more than one author. For example, T04 was written by A03 and A04, i.e. “Hallie Hull” and “Klee Hull”.

Similarly a single

 In this way, the nature of the relationship between titles and publishers is not symmetric.

For example,

***The following is material that was cut from the document. It wasn’t deleted since it might be used in later revisions and I didn’t want to just remove it entirely. You can safely ignore the following material***

The symbols on the lines

The lines that connect tables on the ERD have different symbols at the ends of the lines. For example, between the following shows just a snippet from the ERD that contains the tables titles, publishers and title\_authors. There is a line that connects publishers with titles and a line that connects “title\_authors” with titles



We’ll get to the exact meanings of these symbols soon. However, let’s first point out that on every line connect two tables, there are actually 4 different symbols. These symbols are used to represent the numbers 0, 1 and more than one (i.e. “many”). The technical term for “more than one” is “many”.

The following is the line that connects the publishers table with the titles table: The vertical line closest to the publishers table represents the number 1. The crows foot closest to the titles table represents “many”. Each of these symbols means something very specific.The “1” next to the publishers table carries the meaning that a single title can only be published by at most one publisher. The “crows foot” next to the titles table carries the meaning that a single publisher can publish many titles.

indicate that a single publisher can publish many titles.



The following symbols appear on the line connecting publishers with titles, closest to the publishers table: 

This is actually two different symbols. Closest to the publishers table is a vertical line. This represents the number 1. Right next to that vertical line is a circle – this represents the number zero.

On the same line closer to the titles table are the following two symbols: 

The symbol on the left is another zero. To the right of the zero is this symbol  which we call a “crows foot”. It is comprised of 3 lines – i.e. more than one line (or many lines). We call this a “many” symbol.

 the side of the line touching publishers contains the following symbols closest to the publishers table:

On the side closer to publishers:

On the side closer to titles: 

To figure out the maximum cardinalities between two tables ask yourself the above questions. The “crows foot style of the ERD” only allows for one of two different answers to each question. The answer to each question must be “1” or “more than one” (which we call “many” and symbolize with a crows foot on the diagram).

***Answers to the first 13 questions using R and dplyr***

##############################################################################.

##############################################################################.

# dplyr homework - SOLUTION

#

# There could be other valid solutions.

#

# The code below shows solutions to questions 1-14 in BOOKS DATABASE questions

# in both SQL and dplyr format.

##############################################################################.

##############################################################################.

require(tidyverse)

require(sqldf)

#############################################################################.

#############################################################################.

# Read in the data

#############################################################################.

#############################################################################.

titles = read\_csv("titles.csv", na="NULL", show\_col\_types=FALSE)

authors = read\_csv("authors.csv", na="NULL", show\_col\_types=FALSE)

publishers = read\_csv("publishers.csv", na="NULL", show\_col\_types=FALSE)

title\_authors = read\_csv("title\_authors.csv", na="NULL", show\_col\_types=FALSE)

royalties = read\_csv("royalties.csv", na="NULL", show\_col\_types=FALSE)

#############################################################################.

#############################################################################.

# Answers to the questions

#############################################################################.

#############################################################################.

#---------------------------------------------------------------------------.

# Question 1

#

# List all books (title and number sold) that sold more than 1000 copies. List

# the books with the most sales at the top.

#---------------------------------------------------------------------------.

sqlAnswer =

 sqldf("

 select title\_name, sales

 from titles

 where sales > 1000

 order by sales desc

 ")

dplyrAnswer =

titles %>%

 filter(sales > 1000) %>%

 select(title\_name, sales) %>%

 arrange(desc(sales))

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 2.

#

# List all authors who are live either in NY or CA and whose last name begins

# with a "K".

#---------------------------------------------------------------------------.

sqlAnswer =

 sqldf("

 select au\_fname, au\_lname

 from authors

 where (state = 'NY' or state='CA') and substr(au\_lname,1,1) = 'K';

 ")

dplyrAnswer =

 authors %>%

 filter((state == 'NY' | state == 'CA') & substr(au\_lname, 1, 1) == 'K') %>%

 select(au\_fname, au\_lname)

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 3.

#

# List the first and last names of all authors whose last name starts with a

# letter from A through J

#

# (HINT: In the where clause, make sure that the first

# letter of the last name is both >="A" and also <"K")

#

# (HINT: another possible solution is to use the LIKE several times - once

# for A, once for B, once for C, etc up to J. Each LIKE should be separated

# from the others by OR's)

#---------------------------------------------------------------------------.

sqlAnswer =

sqldf("

 select au\_fname, au\_lname

 from authors

 where substr(au\_lname,1,1) >= 'A' and substr(au\_lname,1,1) < 'K';

 ")

dplyrAnswer =

 authors %>%

 filter(substr(au\_lname, 1, 1) >= 'A' & substr(au\_lname, 1, 1) < 'K') %>%

 select(au\_fname, au\_lname)

sqlAnswer

dplyrAnswer

#3 - another answer

sqlAnswer =

 sqldf("

 select au\_fname, au\_lname

 from authors

 where au\_lname like 'A%' or au\_lname like 'B%' or au\_lname like 'C%' or

 au\_lname like 'D%' or au\_lname like 'E%' or au\_lname like 'F%' or

 au\_lname like 'G%' or au\_lname like 'H%' or au\_lname like 'I%' or

 au\_lname like 'J%'

 ")

dplyrAnswer =

 authors %>%

 filter(grepl("^[A-J]", au\_lname)) %>%

 select(au\_fname, au\_lname)

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 4.

#

# For each author, show their first initial, followed by a period and a space,

# followed by their last name. In the 2nd column show the author's state. Show

# the column heading for the first column as 'AuthorName'. Order the results in

# alphabetical order based on the full name of the person.

#---------------------------------------------------------------------------.

sqlAnswer =

sqldf("

 select substr(au\_fname,1,1) || '. ' || au\_lname as 'AuthorName', state

 from authors

 order by au\_lname, au\_fname;

 ")

dplyrAnswer =

 authors %>%

 arrange(au\_lname, au\_fname) %>%

 mutate(AuthorName = paste0(substr(au\_fname, 1, 1), ". ", au\_lname)) %>%

 select(AuthorName, state)

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 5.

#

# Show the titles of all books. Also show the length of the title, i.e. how many

# characters, including spaces and punctuation. Display the 2nd column with the

# name 'TitleLength'. Sort the output so that the shortest titles are listed

# first. If two titles are the same length then sort those titles

# alphabetically.

#---------------------------------------------------------------------------.

sqlAnswer =

 sqldf("

 select title\_name, length(title\_name) as 'TitleLength'

 from titles

 order by length(title\_name), title\_name;

 ")

dplyrAnswer =

 titles %>%

 mutate(TitleLength = nchar(title\_name)) %>%

 select(title\_name, TitleLength) %>%

 arrange(TitleLength, title\_name)

#############################################################################.

# Single table queries with aggregate functions but no “group by”

# (i.e. these will return EXACTLY one row for each query).

#############################################################################.

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 6.

#

# List the average price of all books.

#---------------------------------------------------------------------------.

sqlAnswer =

sqldf("

 select avg(price)

 from titles;

 ")

dplyrAnswer =

 titles %>%

 summarize(AveragePrice = mean(price, na.rm = TRUE))

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 7.

#

# List the average price of history books.

#---------------------------------------------------------------------------.

sqlAnswer =

 sqldf("

 select avg(price)

 from titles

 where type = 'history';

 ")

dplyrAnswer =

 titles %>%

 filter(type == 'history') %>%

 summarize(AveragePrice = mean(price, na.rm = TRUE))

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 8.

#

# List the number of pages in the longest and shortest books (don't list the

# actual title of the book).

#---------------------------------------------------------------------------.

sqlAnswer =

sqldf("

 select min(pages) , max(pages) from titles;

 ")

dplyrAnswer =

 titles %>%

 summarize(MinPages = min(pages, na.rm = TRUE),

 MaxPages = max(pages, na.rm = TRUE))

sqlAnswer

dplyrAnswer

#############################################################################.

# Single table queries with calculated values

#############################################################################.

#---------------------------------------------------------------------------.

# Question 9.

#

# List the title\_name and total revenue for each book. (Revenue for a book is

# the number sold times the price of the book.)

#---------------------------------------------------------------------------.

sqlAnswer =

 sqldf("

 select title\_name, sales \* price as 'Total Revenue'

 from titles;

")

dplyrAnswer =

 titles %>%

 mutate(TotalRevenue = sales \* price) %>%

 select(title\_name, TotalRevenue)

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 10.

#

# List title of each book and the sale price for the book where the sale price

# is 10% off of the original price.

#---------------------------------------------------------------------------.

sqlAnswer =

sqldf("

 select title\_name, price \* 0.90 as 'Sale Price'

 from titles;

")

dplyrAnswer =

 result <- titles %>%

 mutate(SalePrice = price \* 0.90) %>%

 select(title\_name, SalePrice)

sqlAnswer

dplyrAnswer

#############################################################################.

# Single table queries with “group by”.

# These can use aggregate functions but will return at most one

# row in the output for each “group” of rows as defined in the “group by”

#############################################################################.

#---------------------------------------------------------------------------.

# Question 11.

#

# For each "type" of book (e.g. biography, children, etc) list the number of

# pages in the shortest book of that type and the number of pages in the longest

# book of that type. Sort the results alphabetically by the type of book.

#---------------------------------------------------------------------------.

sqlAnswer =

 sqldf("

 select type, min(pages) AS 'Length of Shortest' , max(pages) as 'Length of Longest'

 from titles

 group by type

 order by type;

 ")

dplyrAnswer =

 titles %>%

 group\_by(type) %>%

 summarize(LengthOfShortest = min(pages, na.rm = TRUE),

 LengthOfLongest = max(pages, na.rm = TRUE)) %>%

 arrange(type)

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 12.

#

# List the types of books (e.g. history, biography, etc) and the average price

# of those books for which the average price of books in that category is at

# least $12.00.

#---------------------------------------------------------------------------.

sqlAnswer =

sqldf("

 select type, avg(price)

 from titles

 group by type

 having avg(price) >= 12;

 ")

dplyrAnswer =

 titles %>%

 group\_by(type) %>%

 summarize(AveragePrice = mean(price, na.rm = TRUE)) %>%

 filter(AveragePrice >= 12)

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 13.

#

# For each type of book (e.g. biography, children, etc), show the number of

# those books that are 450 pages or longer. The column heading should be '# of

# long books'. If there are no titles of a particular type that are so long,

# then don't show that type at all.

#---------------------------------------------------------------------------.

sqlAnswer =

 sqldf("

 select type, count(\*) as '# of long books'

 from titles

 where pages >= 450

 group by type;

 ")

dplyrAnswer =

 titles %>%

 filter(pages >= 450) %>%

 group\_by(type) %>%

 summarize(NumOfLongBooks = n())

sqlAnswer

dplyrAnswer

#---------------------------------------------------------------------------.

# Question 14.

#

# Modify the answer to the previous question so that only those types for which

# there are at least 2 long books are displayed.

#---------------------------------------------------------------------------.

sqlAnswer =

sqldf("

 select type, count(\*) as '# of long books'

 from titles

 where pages >= 450

 group by type

 having count(\*) >= 2;

 ")

dplyrAnswer =

 titles %>%

 filter(pages >= 450) %>%

 group\_by(type) %>%

 summarize(NumOfLongBooks = n()) %>%

 filter(NumOfLongBooks >= 2)

sqlAnswer

dplyrAnswer