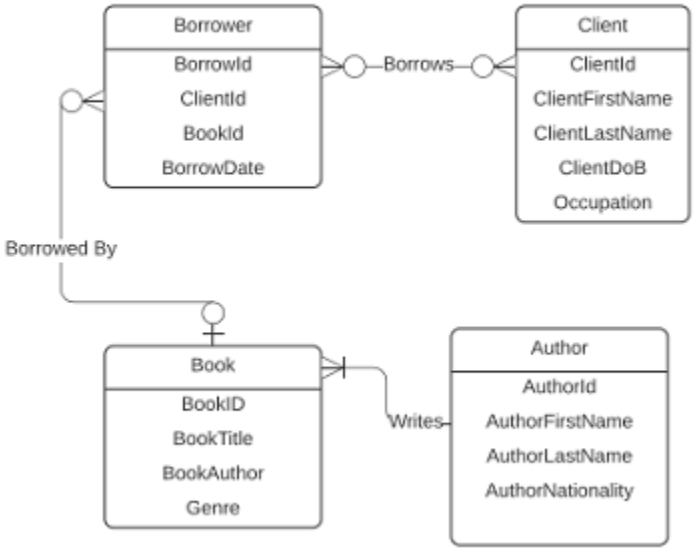
Scott Hurwitz
Computer Science 204: database programming
SQL Assignment

A. Steps A and B grouped together under the following bullet ‘B’
B. I used google sheets to format the data, namely the borrow date field which was in the incorrect format, and then exported to csv. Then I used the following statements to create tables in postgresql and import the data for four tables: author, book, client, borrower. I added primary keys and foreign keys to model the relationship in the diagram:



a. **Author**:

```
CREATE TABLE author ( /*creates new table*/
  authorid SERIAL,
  authorfirstname VARCHAR(50),
  authorlastname VARCHAR(50),
  Authornationality varchar(50),
  PRIMARY KEY (authorid)
);

COPY author /*populates table*/
FROM '/Users/scottsmac/Downloads/204 - author.csv'
DELIMITER ','
CSV HEADER;
```

b. **Book**:

```
CREATE TABLE book (
  bookid SERIAL,
  booktitle VARCHAR(50),
```

```
  bookauthor int,
  genre varchar(50),
  PRIMARY KEY (bookid),
constraint fk_author foreign key(bookauthor)
        references author(authorid)
);

COPY book
FROM '/Users/scottsmac/Downloads/204 - book.csv'
DELIMITER ','
CSV HEADER;
```

c. **Client**:

```
CREATE TABLE client (
  clientid SERIAL,
  clientfirstname VARCHAR(50),
  Clientlastname VARCHAR(50),
  clientdob int,
  occupation varchar(50),
  PRIMARY KEY (clientid),
);

COPY client
FROM '/Users/scottsmac/Downloads/204 - client.csv'
DELIMITER ','
CSV HEADER;
```

d. **Borrower**:

```
CREATE TABLE borrower (
  borrowid SERIAL,
  clientid VARCHAR(50),
  bookid VARCHAR(50),
  Borrowdate date,
  PRIMARY KEY (borrowid),
  constraint fk_client foreign key(clientid)
        references client(clientid),
constraint fk_book foreign key(bookid)
        references book(bookid)
)

COPY borrower
FROM '/Users/scottsmac/Downloads/204 - borrower.csv'
DELIMITER ','
CSV HEADER;
```

C. Note: Postgresql automatically creates a unique index when a unique constraint or primary key is defined for a table

1. Display all contents of the Clients table

   Query:

   Select * from client;

   Result:

```
 clientid | clientfirstname | clientlastname | clientdob |        occupation
----------+-----------------+----------------+-----------+---------------------------
        1 | Kaiden          | Hill           |      2006 | Student
        2 | Alina           | Morton         |      2010 | Student
        3 | Fania           | Brooks         |      1983 | Food Scientist
        4 | Courtney        | Jensen         |      2006 | Student
        5 | Brittany        | Hill           |      1983 | Firefighter
        6 | Max             | Rogers         |      2005 | Student
        7 | Margaret        | McCarthy       |      1981 | School Psychologist
        8 | Julie           | McCarthy       |      1973 | Professor
        9 | Ken             | McCarthy       |      1974 | Securities Clerk
       10 | Britany         | O'Quinn        |      1984 | Violinist
       11 | Conner          | Gardner        |      1998 | Licensed Massage Therapist
       12 | Mya             | Austin         |      1960 | Parquet Floor Layer
       13 | Thierry         | Rogers         |      2004 | Student
       14 | Eloise          | Rogers         |      1984 | Computer Security Manager
       15 | Gerard          | Jackson        |      1979 | Oil Exploration Engineer
       16 | Randy           | Day            |      1986 | Aircraft Electrician
       17 | Jodie           | Page           |      1990 | Manufacturing Director
       18 | Coral           | Rice           |      1996 | Window Washer
       19 | Ayman           | Austin         |      2002 | Student
       20 | Jaxson          | Austin         |      1999 | Repair Worker
       21 | Joel            | Austin         |      1973 | Police Officer
       22 | Alina           | Austin         |      2010 | Student
       23 | Elin            | Austin         |      1962 | Payroll Clerk
       24 | Ophelia         | Wolf           |      2004 | Student
       25 | Eliot           | McGuire        |      1967 | Dentist
       26 | Peter           | McKinney       |      1968 | Professor
       27 | Annabella       | Henry          |      1974 | Nurse
       28 | Anastasia       | Baker          |      2001 | Student
       29 | Tyler           | Baker          |      1984 | Police Officer
       30 | Lilian          | Ross           |      1983 | Insurance Agent
       31 | Thierry         | Arnold         |      1975 | Bus Driver
```

32 | Angelina | Rowe | 1979 | Firefighter
33 | Marcia | Rowe | 1974 | Health Educator
34 | Martin | Rowe | 1976 | Ship Engineer
35 | Adeline | Rowe | 2005 | Student
36 | Colette | Rowe | 1963 | Professor
37 | Diane | Clark | 1975 | Payroll Clerk
38 | Caroline | Clark | 1960 | Dentist
39 | Dalton | Clayton | 1982 | Police Officer
40 | Steve | Clayton | 1990 | Bus Driver
41 | Melanie | Clayton | 1987 | Computer Engineer
42 | Alana | Wilson | 2007 | Student
43 | Carson | Byrne | 1995 | Food Scientist
44 | Conrad | Byrne | 2007 | Student
45 | Ryan | Porter | 2008 | Student
46 | Elin | Porter | 1978 | Computer Programmer
47 | Tyler | Harvey | 2007 | Student
48 | Arya | Harvey | 2008 | Student
49 | Serena | Harvey | 1978 | School Teacher
50 | Lilly | Franklin | 1976 | Doctor
51 | Mai | Franklin | 1994 | Dentist
52 | John | Franklin | 1999 | Firefighter
53 | Judy | Franklin | 1995 | Firefighter
54 | Katy | Lloyd | 1992 | School Teacher
55 | Tamara | Allen | 1963 | Ship Engineer
56 | Maxim | Lyons | 1985 | Police Officer
57 | Allan | Lyons | 1983 | Computer Engineer
58 | Marc | Harris | 1980 | School Teacher
59 | Elin | Young | 2009 | Student
60 | Diana | Young | 2008 | Student
61 | Diane | Young | 2006 | Student
62 | Alana | Bird | 2003 | Student
63 | Anna | Becker | 1979 | Security Agent
64 | Katie | Grant | 1977 | Manager
65 | Joan | Grant | 2010 | Student
66 | Bryan | Bell | 2001 | Student
67 | Belle | Miller | 1970 | Professor
68 | Peggy | Stevens | 1990 | Bus Driver
69 | Steve | Williamson | 1975 | HR Clerk
70 | Tyler | Williamson | 1999 | Doctor
71 | Izabelle | Williamson | 1990 | Systems Analyst
72 | Annabel | Williamson | 1960 | Cashier
73 | Mohamed | Waters | 1966 | Insurance Agent
74 | Marion | Newman | 1970 | Computer Programmer
75 | Ada | Williams | 1986 | Computer Programmer

76 | Sean | Scott | 1983 | Bus Driver
77 | Farrah | Scott | 1974 | Ship Engineer
78 | Christine | Lambert | 1973 | School Teacher
79 | Alysha | Lambert | 2007 | Student
80 | Maia | Grant | 1984 | School Teacher

2. First names, last names, ages and occupations of all clients

   Query:

   select clientfirstname, clientlastname, occupation from client;

   Result:

```
clientfirstname | clientlastname |        occupation
----------------+----------------+--------------------------
Kaiden          | Hill           | Student
Alina           | Morton         | Student
Fania           | Brooks         | Food Scientist
Courtney        | Jensen         | Student
Brittany        | Hill           | Firefighter
Max             | Rogers         | Student
Margaret        | McCarthy       | School Psychologist
Julie           | McCarthy       | Professor
Ken             | McCarthy       | Securities Clerk
Britany         | O'Quinn        | Violinist
Conner          | Gardner        | Licensed Massage Therapist
Mya             | Austin         | Parquet Floor Layer
Thierry         | Rogers         | Student
Eloise          | Rogers         | Computer Security Manager
Gerard          | Jackson        | Oil Exploration Engineer
Randy           | Day            | Aircraft Electrician
Jodie           | Page           | Manufacturing Director
Coral           | Rice           | Window Washer
Ayman           | Austin         | Student
Jaxson          | Austin         | Repair Worker
Joel            | Austin         | Police Officer
Alina           | Austin         | Student
Elin            | Austin         | Payroll Clerk
Ophelia         | Wolf           | Student
Eliot           | McGuire        | Dentist
Peter           | McKinney       | Professor
Annabella       | Henry          | Nurse
Anastasia       | Baker          | Student
```

Tyler	Baker	Police Officer
Lilian	Ross	Insurance Agent
Thierry	Arnold	Bus Driver
Angelina	Rowe	Firefighter
Marcia	Rowe	Health Educator
Martin	Rowe	Ship Engineer
Adeline	Rowe	Student
Colette	Rowe	Professor
Diane	Clark	Payroll Clerk
Caroline	Clark	Dentist
Dalton	Clayton	Police Officer
Steve	Clayton	Bus Driver
Melanie	Clayton	Computer Engineer
Alana	Wilson	Student
Carson	Byrne	Food Scientist
Conrad	Byrne	Student
Ryan	Porter	Student
Elin	Porter	Computer Programmer
Tyler	Harvey	Student
Arya	Harvey	Student
Serena	Harvey	School Teacher
Lilly	Franklin	Doctor
Mai	Franklin	Dentist
John	Franklin	Firefighter
Judy	Franklin	Firefighter
Katy	Lloyd	School Teacher
Tamara	Allen	Ship Engineer
Maxim	Lyons	Police Officer
Allan	Lyons	Computer Engineer
Marc	Harris	School Teacher
Elin	Young	Student
Diana	Young	Student
Diane	Young	Student
Alana	Bird	Student
Anna	Becker	Security Agent
Katie	Grant	Manager
Joan	Grant	Student
Bryan	Bell	Student
Belle	Miller	Professor
Peggy	Stevens	Bus Driver
Steve	Williamson	HR Clerk
Tyler	Williamson	Doctor
Izabelle	Williamson	Systems Analyst
Annabel	Williamson	Cashier

Mohamed | Waters | Insurance Agent
Marion | Newman | Computer Programmer
Ada | Williams | Computer Programmer
Sean | Scott | Bus Driver
Farrah | Scott | Ship Engineer
Christine | Lambert | School Teacher
Alysha | Lambert | Student
Maia | Grant | School Teacher

3. First and last names of clients that borrowed books in March 2018

   Query:
   /*joins borrower and client tables to execute query*/
   Select c.clientfirstname, c.clientlastname
   from client as c inner join borrower as b on c.clientid = b.clientid
   where (select extract(year from b.borrowdate)) = 2018 and (select extract(month from b.borrowdate)) = 3;

   Result:
   clientfirstname | clientlastname
   ----------+----------
   Gerard | Jackson
   Tyler | Baker
   Angelina | Rowe
   Marcia | Rowe
   Carson | Byrne
   Katy | Lloyd
   Alysha | Lambert
   Maia | Grant

4. First and last names of the top 5 authors clients borrowed in 2017

Query:
/*joins book, author and borrower tables to execute query using aggregation*/
Select a.authorfirstname, a.authorlastname, count(a.authorid)
from author as a inner join book as b on a.authorid = b.bookauthor
Inner join borrower as c on c.bookid = b.bookid
Where (select extract(year from c.borrowdate)) = 2017
Group by a.authorid
Order by count desc
Limit 5;

Result:

```
 authorfirstname | authorlastname | count
-----------------+----------------+-------
 Logan           | Moore          |     7
 Elena           | Martin         |     7
 Sofia           | Smith          |     7
 Maria           | Brown          |     6
 Zoe             | Roy            |     5
```

5. Nationalities of the least 5 authors that clients borrowed during the years 2015-2017

Query:

```
/*joins author, book, and borrower tables to execute query using aggregation*/
Select count(a.authorid), a.authornationality
from author as a inner join book as b on a.authorid = b.bookauthor
Inner join borrower as c on c.bookid = b.bookid
Where (select extract(year from c.borrowdate)) between 2015 and 2017
Group by a.authorid
Order by count asc
Limit 5;
```

Result:

```
 count | authornationality
-------+-------------------
     3 | Spain
     5 | USA
     5 | Canada
     6 | USA
     6 | USA
```

6. The book that was most borrowed during the years 2015-2017

Query:

```
/*joins author, book, and borrower tables to execute query using aggregation*/
Select count(b.bookid), b.booktitle
from author as a inner join book as b on a.authorid = b.bookauthor
Inner join borrower as c on c.bookid = b.bookid
Where (select extract(year from c.borrowdate)) between 2015 and 2017
Group by b.bookid
Order by count desc
Limit 1;
```

Result:

```
count |     booktitle
-------+-------------------
    13 | The perfect match
```

7. Top borrowed genres for client born in years 1970-1980

Query:

```
/*joins author, borrower, client, and book tables to execute query using aggregation*/
Select count(b.genre), b.genre
from author as a inner join book as b on a.authorid = b.bookauthor
Inner join borrower as c on c.bookid = b.bookid
Inner join client as d on d.clientid = c.clientid
Where clientdob between 1970 and 1980
Group by b.genre
Order by count desc;
```

Result:

```
 count |   genre
-------+------------
    24 | Science
    16 | Fiction
    13 | Well being


     5 | Humor
     4 | Society
     3 | History
     3 | Children
     3 | Literature
     3 | Law
     2 | well being
```

8. Top 5 occupations that borrowed the most in 2016

Query:

```
/*joins all tables to execute query using aggregation*/
Select count(d.occupation), d.occupation
from author as a inner join book as b on a.authorid = b.bookauthor
Inner join borrower as c on c.bookid = b.bookid
Inner join client as d on d.clientid = c.clientid
Where (select extract(year from c.borrowdate)) = 2016
```

```
Group by d.occupation
Order by count desc
Limit 5;
```

Result:

```
 count |     occupation
-------+---------------------
    32 | Student
     8 | Bus Driver
     6 | Dentist
     6 | Computer Programmer
     5 | Firefighter
```

9. Average number of borrowed books by job title

Query:

```
 /*creates view for all tables joined on pk fk relationships*/
Create or replace view view1 as
Select *
from author as a inner join book as b on a.authorid = b.bookauthor
Inner join borrower as c using (bookid)
Inner join client as d using (clientid);

/*creates second view to form aggregations*/
Create or replace view view2 as
Select occupation, count(occupation) as book_count, count(distinct clientid) as job_title_count
From view1
Group by occupation;

/*casts integer values to float to perform average calculation*/
Select occupation, (book_count::float/job_title_count::float) as  books_per_job_title_holder
From view2;
```

10. Create a VIEW and display the titles that were borrowed by at least 20% of clients

/*uses view that joined all tables*/ /*20 percent of clients is 16*/

Query:

```
Select booktitle, count(clientid)
From view1
Group by booktitle
Having count(clientid) >= 16;
```

Result:

```
        booktitle         | count
--------------------------+-------
 Electrical transformers |    18
```

11. The top month of borrows in 2017

Query:

```
/*uses view that joined all tables, extract clause for datetime manipulation, and aggregation*/
Select extract(month from borrowdate) as month, count(*)
From view1
Where extract(year from borrowdate) = 2017
Group by month
Order by count(*) desc
Limit 3;
```

Result: (three months were tied for most borrows)

```
 month | count
-------+-------
     7 |    10
     8 |    10
    10 |    10
```

12. Average number of borrows by age

```
/*uses view that joined all tables, type casting, calculated field, and aggregation*/
```

Query:

```
Select (2022-clientdob) as age,
(/*age_count*/count((2022-clientdob))::float/ /*borrow count*/count(distinct clientid)::float) as
borrows_per_person_by_age
From view1
Group by age;
```

Result:

```
 age | borrows_per_person_by_age
-----+---------------------------
  12 |        2.3333333333333335
  14 |                         6
  15 |                         5
```

16	5.5
17	4.5
18	3
19	5
20	2
21	4.5
23	3.6666666666666665
24	2
26	2
27	4.5
28	10
30	3
32	5.5
35	2
36	3
37	4
38	5.5
39	3.75
40	3
41	2
42	1
43	4.333333333333333
44	5.5
45	3
46	3.5
47	2.6666666666666665
48	3.25
49	3.6666666666666665
52	4.5
54	4
55	3
56	1
59	5
60	3
62	3.6666666666666665

13. The oldest and the youngest clients of the library

Query:

```
/*uses view that joined all tables, calculated field, and aggregation*/
Select clientfirstname, clientlastname, (2022-clientdob) as age
From view1
Where (2022-clientdob) = (select min(2022-clientdob) from view1)
```

Or (2022-clientdob) = (select max(2022-clientdob) from view1);

Result:

```
 clientfirstname | clientlastname | age
-----------------+----------------+-----
 Annabel         | Williamson     |  62
 Mya             | Austin         |  62
 Joan            | Grant          |  12
 Alina           | Morton         |  12
 Joan            | Grant          |  12
 Alina           | Austin         |  12
 Annabel         | Williamson     |  62
 Joan            | Grant          |  12
 Caroline        | Clark          |  62
 Annabel         | Williamson     |  62
 Alina           | Austin         |  12
 Alina           | Morton         |  12
 Caroline        | Clark          |  62
 Caroline        | Clark          |  62
 Annabel         | Williamson     |  62
 Mya             | Austin         |  62
 Caroline        | Clark          |  62
 Annabel         | Williamson     |  62
```

14. First and last names of authors that wrote books in more than one genre

Select authorid, authorfirstname, authorlastname, count(distinct genre) as genre_count
From view1
Group by authorid, authorfirstname, authorlastname
Having count(distinct genre) > 1;

Result: the following result is given however, this is not valid since the query is picking up a difference in genre case rather than actual difference in genre i.e “well being” vs “Well being”. There are no authors that wrote books in more than one genre.

~~authorid | authorfirstname | authorlastname | genre_count~~
~~---------+-----------------+----------------+-------------~~
~~20 | Helena | Adams | 2~~