Sonia Berrios
CS 204:Database Programming
Study.com

Creating and Manipulating a Database

Create the database

1. CREATE DATABASE Reading_Interests;
2. SHOW DATABASES;
3. USE Reader_Data
4. -- With the Code above the database “Reading_Interests” was created.
    i.

Create the tables

5. CREATE TABLE `Author` (
6. `AuthorID` int NOT NULL,
7. `AuthorFirstName` varchar(45) DEFAULT NULL,
8. `AuthorLastName` varchar(45) DEFAULT NULL,
9. `AuthorNationality` varchar(45) DEFAULT NULL,
10. PRIMARY KEY (`AuthorID`)
11. ----- Below is the code to verify that the table ‘Author’ has been made.
12. SELECT *
13. FROM Author;

14. CREATE TABLE `Book` (
15. `BookID` int NOT NULL,
16. `BookTitle` varchar(45) DEFAULT NULL,
17. `BookAuthor` varchar(45) DEFAULT NULL,
18. `Genre` varchar(45) DEFAULT NULL,
19. PRIMARY KEY (`BookID`)
20. ----- Below is the code to verify that the table ‘Book’ has been made.
21. SELECT *
22. FROM Book;
23. ------* Book has been altered twice to be given a foreign key.
24. ALTER TABLE `Reader_Data`.`Book`
25. CHANGE COLUMN `BookAuthor` `BookAuthor` INT NULL DEFAULT NULL ;
26.
27. ALTER TABLE `Reader_Data`.`Book`
28. ADD INDEX `BookAuthor_idx` (`BookAuthor` ASC) VISIBLE;
29. ;
30. ALTER TABLE `Reader_Data`.`Book`
31. ADD CONSTRAINT `BookAuthor`

```
  FOREIGN KEY (`BookAuthor`)
  REFERENCES `Reader_Data`.`Author` (`AuthorID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;


CREATE TABLE `Borrower` (
 `BorrowID` int NOT NULL,
 `ClientID` varchar(45) DEFAULT NULL,
 `BookID` varchar(45) DEFAULT NULL,
`BorrowDate` varchar(45) DEFAULT NULL,
 PRIMARY KEY (`BorrowID`
----- Below is the code to verify that the table 'Borrower' has been made.
SELECT *
FROM Borrower;
----* Borrower has been altered twice please see the updates below.
ALTER TABLE `Reader_Data`.`Borrower`
CHANGE COLUMN `ClientID` `ClientID` INT NULL DEFAULT NULL ,
CHANGE COLUMN `BookID` `BookID` INT NULL DEFAULT NULL ,
CHANGE COLUMN `BorrowDate` `BorrowDate` DATETIME NULL DEFAULT NULL ;

ALTER TABLE `Reader_Data`.`Borrower`
ADD INDEX `ClientID_idx` (`ClientID` ASC) VISIBLE,
ADD INDEX `BookID_idx` (`BookID` ASC) VISIBLE;
;
ALTER TABLE `Reader_Data`.`Borrower`
ADD CONSTRAINT `ClientID`
  FOREIGN KEY (`ClientID`)
  REFERENCES `Reader_Data`.`Client` (`ClientID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
ADD CONSTRAINT `BookID`
  FOREIGN KEY (`BookID`)
  REFERENCES `Reader_Data`.`Book` (`BookID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION;

CREATE TABLE `Client` (
 `ClientID` int NOT NULL,
`ClientFirstName` varchar(45) DEFAULT NULL,
 `ClientLastName` varchar(45) DEFAULT NULL,
```

```
`ClientDoB` varchar(45) DEFAULT NULL,
`Occupation` varchar(45) DEFAULT NULL,
 PRIMARY KEY (`ClientID`
----- Below is the code to verify that the table 'Client' has been made.
SELECT *
FROM Client;
```

Populate Data with Tables

```
INSERT INTO Author (AuthorFirstName,AuthorLastName,AuthorNationality)
VALUES ('Sofia','Smith','Canada'),
('Maria','Brown','Brazil'),
('Elena','Martin','Mexico'),
 ('Zoe','Roy','France'),
('Sebastian','Lavoie','Canada'),
 ('Dylan','Garcia','Spain'),
 ('Ian','Cruz','Mexico'),
 ('Lucas','Smith','USA'),
 ('Fabian','Wilson','USA'),
('Liam','Taylor','Canada'),
('William','Thomas','Great Britain'),
 ('Logan','Moore','Canada'),
 ('Oliver','Martin','France'),
('Alysha','Thompson','Canada'),
('Isabelle','Lee','Canada'),
 ('Emily','Clark','USA'),
 ('John','Young','China'),
 ('David','Wright','Canada'),
 ('Thomas','Scott','Canada'),
 ('Helena','Adams','Canada'),
 ('Sofia','Carter','USA'),
 ('Liam','Parker','Canada'),
     ('Emily','Murphy','USA');
       ----- Below is the code to verify that the data for 'Author' has been made.
    SELECT *
    FROM Author;

    INSERT INTO Book (BookTitle,BookAuthor,Genre)
    VALUES ('Build your database system','1','Science'),
    ('The red wall','2','Fiction'),
    ('The perfect match','3','Fiction'),
    ('Digital Logic','4','Science'),
    ('How to be a great lawyer','5','Law'),
    ('Manage successful negotiations','6','Society'),
```

112. ('Pollution today','7','Science'),
113. ('A gray park','2','Fiction'),
114. ('How to be rich in one year','8','Humor'),
115. ('Their bright fate','9','Fiction'),
116. ('Black lines','10','Fiction'),
117. ('History of theater','11','Literature'),
118. ('Electrical transformers','12','Science'),
119. ('Build your big data system','1','Science'),
120. ('Right and left','13','Children'),
121. ('Programming using Python','1','Science'),
122. ('Computer networks','14','Science'),
123. ('Performance evaluation','15','Science'),
124. ('Daily exercise','16','Well being'),
125. ('The silver uniform','17','Fiction'),
126. ('Industrial revolution','18','History'),
127. ('Green nature','19','Well being'),
128. ('Perfect football','20','Well being'),
129. ('The chocolate love','21','Humor'),
130. ('Director and leader','22','Society'),
131. ('Play football every week','20','well being'),
132. ('Maya the bee','13','Children'),
133. ('Perfect rugby','20','Well being'),
134. ('The end','23','Fiction'),
135. ('Computer security','1','Science'),
136. ('Participate','22','Society'),
137. ('Positive figures','3','Fiction');
138. ----- Below is the code to verify that the data for ‘Book’ has been made.
    i. SELECT *
    ii. FROM Book;

139. INSERT INTO Borrower(ClientId,BookId,BorrowDate)
140. VALUES (35,17,'2016-07-20'),
141. (1,3,'2017-04-19'),
142. (42,8,'2016-10-03'),
143. (62,16,'2016-04-05'),
144. (53,13,'2017-01-17'),
145. (33,15,'2015-11-26'),
146. (40,14,'2015-01-21'),
147. (64,2,'2017-09-10'),
148. (56,30,'2017-08-02'),
149. (23,2,'2018-06-28'),
150. (46,19,'2015-11-18'),

```
151.    (61,20,'2015-11-24'),
152.    (58,7,'2017-06-17'),
153.    (46,16,'2017-02-12'),
154.    (80,21,'2018-03-18'),
155.    (51,23,'2015-09-01'),
156.    (49,18,'2015-07-28'),
157.    (43,18,'2015-11-04'),
158.    (30,2,'2018-08-10'),
159.    (48,24,'2015-05-13'),
160.    (71,5,'2016-09-05'),
161.    (35,3,'2016-07-03'),
162.    (57,1,'2015-03-17'),
163.    (23,25,'2017-08-16'),
164.    (20,12,'2018-07-24'),
165.    (25,7,'2015-01-31'),
166.    (72,29,'2016-04-10'),
167.    (74,20,'2017-07-31'),
168.    (53,14,'2016-02-20'),
169.    (32,10,'2017-07-24'),
170.    (12,15,'2018-04-25'),
171.    (77,13,'2017-06-09'),
172.    (30,4,'2017-10-24'),
173.    (37,24,'2016-01-14'),
174.    (27,26,'2017-06-05'),
175.    (1,16,'2018-05-06'),
176.    (21,9,'2016-03-19'),
177.    (69,28,'2017-03-29'),
178.    (17,19,'2017-03-14'),
179.    (8,9,'2016-04-22'),
180.    (63,18,'2015-01-25'),
181.    (65,20,'2016-10-10'),
182.    (51,19,'2015-07-28'),
183.    (23,12,'2017-01-25'),
184.    (17,4,'2017-04-18'),
185.    (68,5,'2016-09-06'),
186.    (46,13,'2017-09-30'),
187.    (15,13,'2017-07-05'),
188.    (11,19,'2017-12-14'),
189.    (78,15,'2017-01-26'),
190.    (47,9,'2015-03-03'),
191.    (68,7,'2016-05-26'),
192.    (37,26,'2017-02-06'),
```

193. (48,27,'2015-12-30'),
194. (9,21,'2017-10-21'),
195. (29,8,'2018-04-01'),
196. (64,18,'2017-08-29'),
197. (61,26,'2018-02-21'),
198. (39,28,'2016-07-26'),
199. (73,18,'2018-08-22'),
200. (11,13,'2018-01-17'),
201. (45,6,'2016-07-20'),
202. (33,13,'2018-03-18'),
203. (10,17,'2016-06-06'),
204. (28,18,'2017-02-17'),
205. (51,3,'2016-12-09'),
206. (29,2,'2015-09-18'),
207. (28,30,'2017-09-14'),
208. (74,20,'2015-12-12'),
209. (15,22,'2015-01-14'),
210. (57,8,'2017-08-20'),
211. (2,5,'2015-01-18'),
212. (74,12,'2018-04-14'),
213. (51,10,'2016-02-25'),
214. (25,17,'2015-02-24'),
215. (45,21,'2017-02-10'),
216. (27,25,'2016-08-03'),
217. (32,28,'2016-06-15'),
218. (71,21,'2017-05-21'),
219. (75,26,'2016-05-03'),
220. (56,32,'2015-12-23'),
221. (26,32,'2015-05-16'),
222. (66,32,'2015-05-30'),
223. (57,18,'2017-09-15'),
224. (40,15,'2016-09-02'),
225. (65,4,'2017-08-17'),
226. (54,7,'2015-12-19'),
227. (29,4,'2017-07-22'),
228. (44,9,'2017-12-31'),
229. (56,31,'2015-06-13'),
230. (17,4,'2015-04-01'),
231. (35,16,'2018-07-19'),
232. (22,18,'2017-06-22'),
233. (39,24,'2015-05-29'),
234. (63,14,'2018-01-20'),

```
235.  (53,21,'2016-07-31'),
236.  (40,9,'2016-07-10'),
237.  (52,4,'2017-04-05'),
238.  (27,20,'2016-09-04'),
239.  (72,29,'2015-12-06'),
240.  (49,16,'2017-12-19'),
241.  (6,12,'2016-12-04'),
242.  (74,31,'2016-07-27'),
243.  (48,32,'2016-06-29'),
244.  (69,2,'2016-12-27'),
245.  (60,32,'2017-10-29'),
246.  (45,22,'2017-06-12'),
247.  (42,15,'2017-05-14'),
248.  (79,8,'2016-10-13'),
249.  (70,18,'2016-12-04'),
250.  (34,8,'2016-03-06'),
251.  (43,8,'2015-12-19'),
252.  (42,32,'2016-04-20'),
253.  (67,5,'2017-03-06'),
254.  (80,25,'2015-06-23'),
255.  (54,11,'2017-05-03'),
256.  (34,28,'2017-08-30'),
257.  (65,20,'2017-08-26'),
258.  (61,19,'2018-01-05'),
259.  (38,12,'2018-01-17'),
260.  (51,4,'2016-05-13'),
261.  (7,16,'2016-03-17'),
262.  (46,16,'2016-11-25'),
263.  (75,30,'2018-08-12'),
264.  (72,32,'2015-03-12'),
265.  (44,17,'2015-06-15'),
266.  (68,15,'2016-02-21'),
267.  (21,1,'2016-06-19'),
268.  (14,25,'2016-10-10'),
269.  (68,21,'2016-05-27'),
270.  (35,20,'2015-03-19'),
271.  (16,27,'2016-08-08'),
272.  (79,31,'2018-03-07'),
273.  (14,17,'2018-04-28'),
274.  (29,28,'2018-03-11'),
275.  (41,4,'2018-08-08'),
276.  (42,3,'2016-02-23'),
```

277. (45,3,'2017-07-10'),
278. (36,16,'2018-07-19'),
279. (36,30,'2015-08-07'),
280. (54,32,'2018-03-14'),
281. (61,15,'2017-03-28'),
282. (1,13,'2018-05-17'),
283. (43,1,'2015-05-14'),
284. (37,14,'2015-07-30'),
285. (62,17,'2015-09-19'),
286. (50,22,'2016-12-02'),
287. (45,1,'2016-07-24'),
288. (32,17,'2018-03-10'),
289. (13,28,'2016-02-14'),
290. (15,9,'2018-08-11'),
291. (10,19,'2018-08-29'),
292. (66,3,'2016-11-27'),
293. (68,29,'2017-07-12'),
294. (21,14,'2018-06-27'),
295. (35,9,'2016-01-22'),
296. (17,24,'2016-08-25'),
297. (40,21,'2015-07-09'),
298. (1,24,'2016-03-28'),
299. (70,27,'2015-07-10'),
300. (80,26,'2016-04-24'),
301. (29,5,'2015-10-18'),
302. (76,12,'2018-04-25'),
303. (22,4,'2016-12-24'),
304. (2,2,'2017-10-26'),
305. (35,13,'2016-02-28'),
306. (40,8,'2017-10-02'),
307. (68,9,'2016-01-03'),
308. (32,5,'2016-11-13'),
309. (34,17,'2016-09-15'),
310. (34,16,'2018-04-13'),
311. (80,30,'2016-10-13'),
312. (20,32,'2015-11-17'),
313. (36,10,'2017-09-01'),
314. (78,12,'2018-06-27'),
315. (57,8,'2016-03-22'),
316. (75,11,'2017-06-27'),
317. (71,10,'2015-08-01'),
318. (48,22,'2015-09-29'),

(19,16,'2016-02-21'),
(79,30,'2018-08-20'),
(70,13,'2016-09-16'),
(30,6,'2017-02-10'),
(45,12,'2017-10-12'),
(30,27,'2016-11-23'),
(26,3,'2016-08-13'),
(66,6,'2017-01-14'),
(47,15,'2016-02-10'),
(53,30,'2018-08-08'),
(80,16,'2016-03-31'),
(70,13,'2018-02-03'),
(14,25,'2016-03-27'),
(46,22,'2016-01-13'),
(30,32,'2015-08-06'),
(60,14,'2016-11-27'),
(14,13,'2018-05-23'),
(71,15,'2016-06-22'),
(38,21,'2015-12-27'),
(69,30,'2017-04-29'),
(49,31,'2018-06-03'),
(28,28,'2015-05-29'),
(49,3,'2016-08-30'),
(75,1,'2015-10-29'),
(78,3,'2017-05-12'),
(43,18,'2015-03-25'),
(27,21,'2016-02-22'),
(64,22,'2015-04-03'),
(21,11,'2017-12-09'),
(66,29,'2016-12-20'),
(45,13,'2017-04-15'),
(48,30,'2015-01-31'),
(20,25,'2017-12-20'),
(41,20,'2018-01-29'),
(51,12,'2015-07-05'),
(5,1,'2015-04-12'),
(40,3,'2018-02-24'),
(79,4,'2018-06-27'),
(15,10,'2016-11-01'),
(42,22,'2016-12-28'),
(17,9,'2018-01-29'),
(38,13,'2016-05-09'),

(79,2,'2017-12-06'),
(74,3,'2015-12-07'),
(46,8,'2016-06-05'),
(78,22,'2018-08-11'),
(45,2,'2015-04-20'),
(72,31,'2015-11-11'),
(18,17,'2015-03-21'),
(29,3,'2017-08-13'),
(66,11,'2018-06-05'),
(36,16,'2016-04-28'),
(26,2,'2016-10-23'),
(32,1,'2017-10-31'),
(62,14,'2017-07-25'),
(12,4,'2015-07-08'),
(38,32,'2015-02-24'),
(29,16,'2016-07-28'),
(36,25,'2017-05-07'),
(76,7,'2015-06-13'),
(28,16,'2016-08-15'),
(60,13,'2016-08-26'),
(8,3,'2017-07-28'),
(25,1,'2016-07-30'),
(62,29,'2018-08-24'),
(51,8,'2016-09-01'),
(27,23,'2015-02-08'),
(69,12,'2018-06-25'),
(51,12,'2015-07-04'),
(7,4,'2015-05-01'),
(31,15,'2017-10-29'),
(14,23,'2015-01-15'),
(14,1,'2018-05-21'),
(39,25,'2015-12-26'),
(79,24,'2016-05-31'),
(40,15,'2016-03-18'),
(51,13,'2018-04-13'),
(61,1,'2015-02-11'),
(15,24,'2018-03-02'),
(10,22,'2018-01-21'),
(67,10,'2017-07-08'),
(79,11,'2016-12-11'),
(19,32,'2016-05-04'),
(35,11,'2017-08-01'),

```
403.    (27,13,'2017-12-15'),
404.    (30,22,'2015-12-22'),
405.    (8,7,'2015-06-26'),
406.    (70,9,'2016-03-20'),
407.    (56,18,'2016-01-29'),
408.    (13,19,'2015-03-06'),
409.    (61,2,'2016-06-18'),
410.    (47,13,'2017-09-18'),
411.    (30,22,'2016-02-19'),
412.    (18,22,'2016-12-31'),
413.    (34,29,'2017-10-27'),
414.    (32,21,'2015-06-03'),
415.    (9,28,'2016-03-30'),
416.    (62,24,'2015-03-23'),
417.    (44,22,'2017-04-29'),
418.    (27,5,'2015-03-25'),
419.    (61,28,'2017-07-14'),
420.    (5,13,'2016-12-04'),
421.    (43,19,'2018-03-15'),
422.    (34,19,'2016-06-05'),
423.    (35,5,'2018-02-19'),
424.    (13,12,'2016-09-23'),
425.    (74,18,'2016-12-26'),
426.    (70,31,'2017-08-15'),
427.    (42,17,'2016-06-15'),
428.    (51,24,'2018-07-30'),
429.    (45,30,'2015-01-15'),
430.    (70,17,'2017-10-07'),
431.    (77,7,'2017-01-06'),
432.    (74,25,'2015-09-25'),
433.    (47,14,'2018-02-01'),
434.    (10,2,'2017-04-18'),
435.    (16,21,'2016-10-03'),
436.    (48,5,'2016-09-17'),
437.    (72,3,'2017-02-10'),
438.    (26,23,'2016-03-01'),
439.    (49,23,'2016-10-25');
440.     ----- Below is the code to verify that the data for ‘Borrower’ has been made.
441.    SELECT *
442.    FROM Borrower;
443.
444.    INSERT INTO Client(ClientFirstName,ClientLastName,ClientDoB,Occupation)
```

```
445.    VALUES  ('Kaiden','Hill',2006,'Student'),
446.     ('Alina','Morton',2010,'Student'),
447.     ('Fania','Brooks',1983,'Food Scientist'),
448.     ('Courtney','Jensen',2006,'Student'),
449.     ('Brittany','Hill',1983,'Firefighter'),
450.     ('Max','Rogers',2005,'Student'),
451.     ('Margaret','McCarthy',1981,'School Psychologist'),
452.     ('Julie','McCarthy',1973,'Professor'),
453.     ('Ken','McCarthy',1974,'Securities Clerk'),
454.     ('Britany','O''Quinn',1984,'Violinist'),
455.     ('Conner','Gardner',1998,'Licensed Massage Therapist'),
456.     ('Mya','Austin',1960,'Parquet Floor Layer'),
457.     ('Thierry','Rogers',2004,'Student'),
458.     ('Eloise','Rogers',1984,'Computer Security Manager'),
459.     ('Gerard','Jackson',1979,'Oil Exploration Engineer'),
460.     ('Randy','Day',1986,'Aircraft Electrician'),
461.     ('Jodie','Page',1990,'Manufacturing Director'),
462.     ('Coral','Rice',1996,'Window Washer'),
463.     ('Ayman','Austin',2002,'Student'),
464.     ('Jaxson','Austin',1999,'Repair Worker'),
465.     ('Joel','Austin',1973,'Police Officer'),
466.     ('Alina','Austin',2010,'Student'),
467.     ('Elin','Austin',1962,'Payroll Clerk'),
468.     ('Ophelia','Wolf',2004,'Student'),
469.     ('Eliot','McGuire',1967,'Dentist'),
470.     ('Peter','McKinney',1968,'Professor'),
471.     ('Annabella','Henry',1974,'Nurse'),
472.     ('Anastasia','Baker',2001,'Student'),
473.     ('Tyler','Baker',1984,'Police Officer'),
474.     ('Lilian','Ross',1983,'Insurance Agent'),
475.     ('Thierry','Arnold',1975,'Bus Driver'),
476.     ('Angelina','Rowe',1979,'Firefighter'),
477.     ('Marcia','Rowe',1974,'Health Educator'),
478.     ('Martin','Rowe',1976,'Ship Engineer'),
479.     ('Adeline','Rowe',2005,'Student'),
480.     ('Colette','Rowe',1963,'Professor'),
481.     ('Diane','Clark',1975,'Payroll Clerk'),
482.     ('Caroline','Clark',1960,'Dentist'),
483.     ('Dalton','Clayton',1982,'Police Officer'),
484.     ('Steve','Clayton',1990,'Bus Driver'),
485.     ('Melanie','Clayton',1987,'Computer Engineer'),
486.     ('Alana','Wilson',2007,'Student'),
```

```
('Carson','Byrne',1995,'Food Scientist'),
('Conrad','Byrne',2007,'Student'),
('Ryan','Porter',2008,'Student'),
('Elin','Porter',1978,'Computer Programmer'),
('Tyler','Harvey',2007,'Student'),
('Arya','Harvey',2008,'Student'),
('Serena','Harvey',1978,'School Teacher'),
('Lilly','Franklin',1976,'Doctor'),
('Mai','Franklin',1994,'Dentist'),
('John','Franklin',1999,'Firefighter'),
('Judy','Franklin',1995,'Firefighter'),
('Katy','Lloyd',1992,'School Teacher'),
('Tamara','Allen',1963,'Ship Engineer'),
('Maxim','Lyons',1985,'Police Officer'),
('Allan','Lyons',1983,'Computer Engineer'),
('Marc','Harris',1980,'School Teacher'),
('Elin','Young',2009,'Student'),
('Diana','Young',2008,'Student'),
('Diane','Young',2006,'Student'),
('Alana','Bird',2003,'Student'),
('Anna','Becker',1979,'Security Agent'),
('Katie','Grant',1977,'Manager'),
('Joan','Grant',2010,'Student'),
('Bryan','Bell',2001,'Student'),
('Belle','Miller',1970,'Professor'),
('Peggy','Stevens',1990,'Bus Driver'),
('Steve','Williamson',1975,'HR Clerk'),
('Tyler','Williamson',1999,'Doctor'),
('Izabelle','Williamson',1990,'Systems Analyst'),
('Annabel','Williamson',1960,'Cashier'),
('Mohamed','Waters',1966,'Insurance Agent'),
('Marion','Newman',1970,'Computer Programmer'),
('Ada','Williams',1986,'Computer Programmer'),
('Sean','Scott',1983,'Bus Driver'),
('Farrah','Scott',1974,'Ship Engineer'),
('Christine','Lambert',1973,'School Teacher'),
('Alysha','Lambert',2007,'Student'),
('Maia','Grant',1984,'School Teacher');
----- Below is the code to verify that the data for ‘Client’ has been made.
SELECT *
FROM Client;
```

Indexes

```
------The following Indexes were created before the query process in order to ease the query process.

CREATE INDEX Client_idx
ON Client (ClientID,ClientFirstName,ClientLastName,ClientDoB,Occupation);

 CREATE INDEX Borrower_idx
 ON Borrower (BorrowId ,ClientId,BookId,BorrowDate);
```

Queries

```
Query 1: Display all contents of the Clients table

--- This was a simple query that required the a simple code. The result grid provided all  the information for the Client table  organized by ClientID.

SELECT *
FROM Client;


Query 2: First names, last names, ages and occupations of all clients
---Becuase there is no age option in the information that was initial given. The age is calculated the current year minus the clients date of birth. The code (YEAR(NOW())-ClientDoB) is the solution to the age equation. The code also provides an Alias for Age.
SELECT ClientFirstName,ClientLastName,(YEAR(NOW())-ClientDoB) AS Age,Occupation
FROM Client;


Query 3:  First and last names of clients that borrowed books in March 2018
---This Query was a little difficult. I tried a couple of queries and non gave me the results that I needed.  I narrowed the results to the Clients first and last name.  The Inner Join glued together the search and as it narrowed it down to the borrowers. The Where clause was needed for the date range.

SELECT ClientFirstName,ClientLastName
FROM Client
INNER JOIN Borrower
```

```
ON Client.ClientId = Borrower.ClientId
WHERE MONTH(BorrowDate) = 3 AND YEAR(BorrowDate) = 2018
GROUP BY Client.ClientId;

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

--- For this query I needed to find the TOPS five authors that readers (Clients) borrowed. This means that would need information from the Author, Book, and Borrower tables and for this I would need the inner join and connect them with the respective ON clauses. The where claused was used to filter the year 2017 and the Order By Count was used to filter the top 5 authors.

SELECT AuthorFirstName, AuthorLastName

FROM Author

INNER JOIN Book

ON Book.BookAuthor = Author.AuthorId

INNER JOIN Borrower

ON Book.BookId = Borrower.BookId

WHERE YEAR(BorrowDate) = 2017

GROUP BY Borrower.BookId

ORDER BY COUNT(Borrower.BookId) DESC LIMIT 5;

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

----  I used INNER JOIN to connect the tables Author, Book, and Borrower with their respective ON clauses. The WHERE clause was used to filter the years the books were borrowed. The GROUP By clause was to search for the authors nationality. ORDER BY clause  was used to filter the least 5 authors.

SELECT AuthorNationality

FROM Author

INNER JOIN Book

ON Book.BookAuthor = Author.AuthorId

INNER JOIN Borrower
```

```
574.  ON Book.BookId = Borrower.BookId
575.  WHERE YEAR(BorrowDate) BETWEEN 2015 AND 2017
576.  GROUP BY AuthorNationality
577.  ORDER BY COUNT(Borrower.BookId) ASC LIMIT 5;
578.
```

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

580. ---- INNER JOIN was used to connect the tables Book and Borrower with Book.BookId = Borrower.BookId, as I needed the information from both tables. WHERE specifies the year the books were borrowed and the BETWEEN clause uses the range of years used to gather the informaiton. GROUP BY COUNT the descending order to.

```
581.  SELECT BookTitle
582.  FROM Book
583.  INNER JOIN Borrower
584.  ON Book.BookId = Borrower.BookId
585.  WHERE YEAR(BorrowDate) BETWEEN 2015 AND 2017
586.  GROUP BY Borrower.BookId
587.  ORDER BY COUNT(Borrower.BookId) DESC LIMIT 1;
588.
```

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

590. ---INNER JOIN was used to connect the table Borrower and Book.BookId = Borrower.BookId, and it was used to connect the Client and ClientId = Client.ClientId. The WHERE clause was used to filter the clientsDOB between the years 1970 and 1980. It was all wall filtered by Genres.

```
591.  SELECT Genre
592.  FROM Book
593.  INNER JOIN Borrower
594.  ON Book.BookId = Borrower.BookId
595.  INNER JOIN Client
596.  ON Borrower.ClientId = Client.ClientId
```

597. WHERE ClientDoB BETWEEN 1970 AND 1980

598. GROUP BY Genre;

599.

600. Query 8: Top 5 occupations that borrowed the most in 2016

601. -----The INNER JOIN was used to connect the table borrower with the ON clause. The WHERE clause was used to filter to the date 2016. The ORDER BY COUNT clause was limited to decensd to 5 and stop at just 5.

602. SELECT Occupation

603. FROM Client

604. INNER JOIN Borrower

605. ON Borrower.ClientId = Client.ClientId

606. WHERE YEAR(BorrowDate) = 2016

607. GROUP BY Occupation

608. ORDER BY COUNT(Borrower.ClientId) DESC LIMIT 5;

609.

610. Query 9:Average number of borrowed books by job title

611. --- Because I was averaging the number of books by job title I used ROUND, Average_ Borrowed, and COUNT for Client to calculate the averages. Like the queries before INNER JOIN was used to connect Borrower and the ON clause.

612. SELECT Client.Occupation, ROUND(COUNT(Client.Occupation)/ COUNT(DISTINCT Borrower.ClientId))AS Average_Borrowed

613. FROM Client

614. INNER JOIN Borrower

615. ON Borrower.ClientId = Client.ClientId

616. GROUP BY Client.Occupation;

617.

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

619. ---This was to challenging query. Like always I used the INNER JOIN to connect the table Borrower with the ON clause Borrower.BookId = Book.BookId. I used the GROUP

BY clause to organize everything by BookTitle. I used the HAVING cause since I used the function COUNT.

```
620.  CREATE VIEW [TwentyPBooks] AS
621.  SELECT BookTitle
622.  FROM Book
623.  INNER JOIN Borrower
624.  ON Borrower.BookId = Book.BookId
625.  GROUP BY BookTitle
626.  HAVING  (COUNT(Borrower.ClientId)>0.2);
627.
```

628. Query 11: The top month of borrows in 2017

629. --- I used the SELECT to select the borrow dates/months count of the borrow dates/months and months. In the SELECT statement the AS function was also used to for the Alias of the BorrowDate which is the Month. The WHERE cause was used to filter to the year 2017. The ORDER BY clause was used in descending orer for a limit of 5.

```
630.  SELECT MONTH(BorrowDate) AS Top_Month, COUNT(MONTH(BorrowDate))
AS Amount_of_books_borrowed
631.   FROM Borrower
632.   WHERE YEAR(BorrowDate) = 2017
633.   GROUP BY MONTH(BorrowDate)
634.   ORDER BY Amount_of_books_borrowed DESC LIMIT 5;
635.
```

636. Query 12: Average number of borrows by age

637. ---(YEAR(NOW())-Client.ClientDoB) was used to find the ages of the Clients. The function was used to minus the current year with the date of birth. The AS AGE function was used to input results as alias for AGE. ROUND(COUNT(Client.ClientId)/ COUNT(DISTINCT Borrower.ClientId)) was used to calculate the number of books borrowed.

```
638.  SELECT DISTINCT (YEAR(NOW())-Client.ClientDoB) AS AGE,
ROUND(COUNT(Client.ClientId)/ COUNT(DISTINCT Borrower.ClientId))AS
Average_Borrowed
```

```
639.    FROM Client
640.    INNER JOIN Borrower
641.    ON Borrower.ClientId = Client.ClientId
642.    GROUP BY AGE;
643.
644.    Query 13: The oldest and the youngest clients of the library
645.    --- Max and Min function were used for highest and lowest number of ages. The
        (YEAR(NOW())-Client.ClientDoB) were used to calculate the ages. The AS function
        were used to create alias.
646.    SELECT MAX(YEAR(NOW())-Client.ClientDoB) AS
        Oldest,MIN(YEAR(NOW())-Client.ClientDoB)AS Youngest
647.    FROM Client
648.    INNER JOIN Borrower
649.    ON Borrower.ClientId = Client.ClientId;
650.
651.    Query 14: First and last names of authors that wrote books in more than one genre
652.    --- The Select statement was used to select the Author Firstname and Author
        Lastname.  This information from
653.    SELECT Author.AuthorFirstName AS First_Name, Author.AuthorLastName AS
        Last_Name. The function  (COUNT(DISTINCT Book.Genre) > 1) was used to count the
        number of booker > than 1.
654.    FROM Book
655.    INNER JOIN Author
656.    ON Author.AuthorId = Book.BookAuthor
657.    GROUP BY Author.AuthorId
658.    HAVING (COUNT(DISTINCT Book.Genre) > 1);
659.
660.
```

661.